SYBASE®

内存数据库用户指南

Adaptive Server® Enterprise

15.5

文档 ID：DC01273-01-1550-01

最后修订日期：2009 年 11 月

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

目录

关于本手册

读者

本手册的目标读者是正在使用内存数据库的系统管理员和数据库所有者。

如何使用本手册

- 第 1 章 “内存数据库”介绍内存数据库和宽松持久性数据库。
- 第 2 章 “管理内存数据库和宽松持久性数据库”介绍创建内存数据库和宽松持久性数据库所需的步骤。
- 第 3 章 “最少日志记录 DML”介绍为 Adaptive Server 数据库提供的不同日志记录级别。
- 第 4 章 “内存数据库的性能和调优”介绍内存数据库和宽松持久性数据库的性能和调优特性。

相关文档

Adaptive Server® Enterprise 文档集包括：

- 针对所用平台的发行公告 — 包含未能及时写入手册的最新信息。

  最新版本的发行公告可能已可获得。若要了解本产品 CD 发行以后增加的重要产品或文档信息，请使用 Sybase® Product Manuals 网站。
- 针对所用平台的安装指南 — 介绍所有 Adaptive Server 产品及相关 Sybase 产品的安装、升级和一些配置过程。
- New Feature Summary （《新增功能摘要》） — 介绍 Adaptive Server 中的新增功能、为支持这些功能所做的系统更改，以及可能会影响现有应用程序的更改。
- Active Messaging Users Guide （《Active Messaging 用户指南》） — 介绍如何使用 Active Messaging 功能捕获 Adaptive Server Enterprise 数据库中的事务 （数据更改），并将它们作为事件实时发送给外部应用程序。
- 《组件集成服务用户指南》— 说明如何使用组件集成服务来连接远程 Sybase 和非 Sybase 数据库。
- 针对所用平台的 Configuration Guide （《配置指南》） — 提供执行特定配置任务的说明。
- 《词汇表》 — 定义 Adaptive Server 文档中使用的技术术语。
- 《Historical Server 用户指南》 — 介绍如何使用 Historical Server 从 Adaptive Server 获取性能信息。

- 《Adaptive Server Enterprise 中的 Java》— 介绍在 Adaptive Server 数据库中如何安装 Java 类以及如何将它们用作数据类型、函数和存储过程。
- 《Job Scheduler 用户指南》 — 提供有关如何使用命令行或图形用户界面 (GUI) 在本地或远程 Adaptive Server 上进行安装和配置以及创建和调度作业的说明。
- 《迁移技术指南》 — 介绍迁移到 Adaptive Server 的其它版本所需的策略和工具。
- 《Monitor Client Library 程序员指南》 — 介绍如何编写可访问 Adaptive Server 性能数据的 Monitor Client Library 应用程序。
- 《Monitor Server 用户指南》— 介绍如何使用 Monitor Server 从 Adaptive Server 获取性能统计信息。
- Monitoring Tables Diagram （《监控表框图》） — 以张贴画格式阐明监控表及其实体关系。大图只在印刷品中提供；采用 PDF 格式时提供缩略图。
- Performance and Tuning Series（《性能和调优系列》）— 是一系列丛书，介绍如何调节 Adaptive Server 以获得最佳性能：
  - Basics（《基础知识》）— 包含了解和研究 Adaptive Server 中的性能问题需具备的基础知识。
  - Improving Performance with Statistical Analysis （《利用统计分析改进性能》） — 介绍 Adaptive Server 如何存储和显示统计信息，以及如何使用 set statistics 命令分析服务器统计信息。
  - Locking and Concurrency Control （《锁定和并发控制》） — 介绍如何使用锁定方案来提高性能，以及如何选择索引以最大限度地减少并发。
  - Monitoring Adaptive Server with sp_sysmon （《使用 sp_sysmon 监控 Adaptive Server》） — 讨论如何使用 sp_sysmon 监控性能。
  - Monitoring Tables （《监控表》） — 介绍如何从 Adaptive Server 监控表中查询统计信息和诊断信息。
  - Physical Database Tuning （《物理数据库调优》） — 介绍如何管理物理数据放置、为数据分配的空间以及临时数据库。
  - Query Processing and Abstract Plans （《查询处理和抽象计划》） — 介绍优化程序如何处理查询以及如何使用抽象计划更改某些优化程序计划。

- 《快速参考指南》 — 这是一本袖珍手册，完整地列出了各种命令、函数、系统过程、扩展系统过程、数据类型和实用程序的名称和语法（该手册在用 PDF 格式查看时采用正常大小）。
- 《参考手册》 — 是一系列丛书，包含详细的 Transact-SQL® 信息：
  - 《构件块》 — 讨论数据类型、函数、全局变量、表达式、标识符、通配符和保留字。
  - 《命令》 — 提供了命令的文档资料。
  - 《过程》 — 介绍系统过程、目录存储过程、系统扩展存储过程和 dbcc 存储过程。
  - 《表》 — 讨论系统表、监控表和 dbcc 表。
- 《系统管理指南》 —
  - 《卷 1》 — 介绍系统管理的基础知识，包括对配置参数、资源问题、字符集、排序顺序的描述以及有关诊断系统问题的说明。《卷 1》的第二部分深入讨论了安全管理。
  - 《卷 2》 — 包括针对管理物理资源、镜像设备、配置内存和数据高速缓存、管理多处理器服务器和用户数据库、装入和卸下数据库、创建和使用段、使用 reorg 命令以及检查数据库一致性的说明和指导。《卷 2》的后半部分介绍如何备份和恢复系统数据库和用户数据库。
- System Tables Diagram（《系统表框图》） — 以张贴画格式阐明系统表及其实体关系。大图只在印刷品中提供；采用 PDF 格式时提供缩略图。
- 《Transact-SQL 用户指南》 — 提供有关 Transact-SQL 这一 Sybase 关系数据库语言增强版的文档资料。此指南可用作数据库管理系统初级用户的教科书，其中还包含有关 pubs2 和 pubs3 示例数据库的详细说明。
- Troubleshooting and Error Messages Guide（《故障排除和错误消息指南》） — 包含有关如何解除最常出现的 Adaptive Server 错误消息的详细说明。
- 《加密列用户指南》 — 介绍如何通过 Adaptive Server 配置和使用加密列。
- 《内存数据库用户指南》 — 介绍如何配置和使用内存数据库。
- Using Adaptive Server Distributed Transaction Management Features（《使用 Adaptive Server 分布式事务管理功能》） — 介绍如何在分布式事务处理环境中配置、使用 Adaptive Server DTM 功能以及如何排除其中的故障。

- 《将 Backup Server 与 IBM® Tivoli® Storage Manager 配合使用》 — 介绍如何设置和使用 IBM Tivoli Storage Manager 以创建 Adaptive Server 备份。
- 《在高可用性系统中使用 Sybase 故障切换》 — 提供有关使用 Sybase 的故障切换功能将 Adaptive Server 配置为高可用性系统中的协同服务器的说明。
- Unified Agent and Agent Management Console （《Unified Agent 和 Agent Management Console》） — 介绍 Unified Agent，它用于提供管理、监控和控制分布式 Sybase 资源的运行期服务。
- 《实用程序指南》 — 提供有关在操作系统级别执行的 Adaptive Server 实用程序 （如 isql 和 bcp）的文档资料。
- 《Web 服务用户指南》 — 介绍如何配置、使用 Adaptive Server 的 Web 服务以及如何排除其中的故障。
- *XA Interface Integration Guide for CICS, Encina, and TUXEDO*（《适用于 CICS、 Encina 和 TUXEDO 的 XA 接口集成指南》） — 提供有关将 Sybase DTM XA 接口与 X/Open XA 事务管理器配合使用的说明。
- 《Adaptive Server Enterprise 中的 XML 服务》 — 介绍 Sybase 本机 XML 处理器和 Sybase 基于 Java 的 XML 支持以及数据库中的 XML，并提供了有关 XML 服务中可用的查询和映射函数的文档资料。

其它信息来源

使用 Sybase Getting Started CD、SyBooks™ CD 和 Sybase Product Manuals 网站可以了解有关产品的更多信息：

- Getting Started CD 包含 PDF 格式的发行公告和安装指南，还可能包含 SyBooks CD 中未收纳的其它文档或更新信息。它随软件一起提供。若要阅读或打印 Getting Started CD 上的文档，需要使用 Adobe Acrobat Reader，该软件可以使用 CD 上提供的链接从 Adobe 网站免费下载。
- SyBooks CD 含有产品手册，它随软件一起提供。基于 Eclipse 的 SyBooks 浏览器使您能够以易于使用的、基于 HTML 的格式阅读手册。

  有些文档可能是以 PDF 格式提供的，您可以通过 SyBooks CD 上的 PDF 目录访问这些文档。若要阅读或打印 PDF 文件，您需要使用 Adobe Acrobat Reader。

  有关安装和启动 SyBooks 的说明，请参见 Getting Started CD 上的 SyBooks Installation Guide （《SyBooks 安装指南》）或 SyBooks CD 上的 *README.txt* 文件。

- Sybase Product Manuals 网站是 SyBooks CD 的联机版本，您可以使用标准 Web 浏览器来访问它。除了产品手册之外，还可以找到“EBF/ 维护”(EBFs/Maintenance)、“技术文档”(Technical Documents)、“案例管理”(Case Management)、“解决的案例”(Solved Cases)、“新闻组”(Newsgroups) 和“Sybase 开发人员网络”(Sybase Developer Network) 的链接。

  若要访问 Sybase Product Manuals 网站，请转到位于 http://www.sybase.com/support/manuals/ 的“产品手册”(Product Manuals)。

Web 上的 Sybase 认证

Sybase 网站上的技术文档在不断地更新。

❖ **查找有关产品认证的最新信息**

1 将 Web 浏览器定位到位于 http://www.sybase.com/support/techdocs/ 的“技术文档”(Technical Documents)。

2 单击“认证报告”(Certification Report)。

3 在“认证报告”(Certification Report) 过滤器中选择相应的产品、平台和时间范围，然后单击“查找”(Go)。

4 单击“认证报告”(Certification Report) 标题显示该报告。

❖ **查找有关组件认证的最新信息**

1 将 Web 浏览器定位到位于 http://certification.sybase.com/ 的“可用性和认证报告”(Availability and Certification Reports)。

2 在“按基本产品搜索”(Search by Base Product) 下选择产品系列和产品，或在“按平台搜索”(Search by Platform) 下选择平台和产品。

3 选择“搜索”(Search) 以显示所选项目的可用性和认证报告。

❖ **创建 Sybase 网站（包括支持页）的个性化视图**

建立 MySybase 配置文件。 MySybase 是一项免费服务，它允许您创建 Sybase 网页的个性化视图。

1 将 Web 浏览器定位到位于 http://www.sybase.com/support/techdocs/ 的“技术文档”(Technical Documents)。

2 单击“Sybase”并创建 MySybase 配置文件。

Sybase EBF 和软件维护

❖ **查找有关 EBF 和软件维护的最新信息**

1 将 Web 浏览器定位到位于 http://www.sybase.com/support 的“Sybase 支持页” (Sybase Support Page)。

2 选择 “EBF/ 维护” (EBFs/Maintenance)。如果出现提示信息，请输入您的 MySybase 用户名和口令。

3 选择一个产品。

4 指定时间范围并单击 “查找” (Go)。即会显示 EBF/ 维护版本的列表。

锁形图标表示因为您没有注册为 “技术支持联系人” (Technical Support Contact)，因此您没有某些 EBF/ 维护版本的下载授权。如果您尚未注册，但拥有 Sybase 代表提供的或通过支持合同获得的有效信息，请单击 “编辑角色” (Edit Roles) 将 “技术支持联系人” (Technical Support Contact) 角色添加到 MySybase 配置文件中。

5 单击信息图标可显示 EBF/ 维护报告，单击产品说明可下载软件。

约定

以下各部分将说明在本手册中使用的约定。

SQL 是一种形式自由的语言。没有规定每一行中的单词数量或者必须换行的地方。不过，为了便于阅读，本手册中的所有示例和大多数语法语句都经过了格式设置，以便语句的每个子句都在一个新行上开始。有多个成分的子句会扩展到其它行，这些行会有缩进。复杂命令使用修正的 Backus Naur Form (BNF) 表示法进行了格式设置。

表 1 说明本手册中出现的语法语句的约定：

表 1：本手册的字体和语法约定

元素	示例
命令名、过程名、实用程序名和其它关键字用 sans serif 字体显示。	select sp_configure
数据库名和数据类型用 sans serif 字体显示。	master 数据库
书名采用正常字体并加书名号；文件名、变量和路径名用斜体显示。	《系统管理指南》 *sql.ini* 文件 *column_name* *$SYBASE/ASE* 目录

变量 （即代表您要填充的值的词语）作为查询或语句的一部分出现时用斜体的 Courier 字体显示。	`select column_name`
	`from table_name`
	`where search_conditions`

输入小括号作为命令的一部分。	compute *row_aggregate* (*column_name*)

元素	示例
双冒号加等号表示语法是用 BNF 表示法编写的。请勿输入此符号。表示“被定义为”。	`::=`
大括号表示必须至少选择括号中的一个选项。请勿输入大括号。	`{cash, check, credit}`
中括号表示可以选择括号中的一个或多个选项，也可不选。请勿输入中括号。	`[cash | check | credit]`
逗号表示可以选择任意多个所显示的选项。可用逗号作为命令的一部分来分隔选项。	`cash, check, credit`
竖线 (|) 表示只可选择所显示的选项中的一个。	`cash | check | credit`
省略号 (...) 表示可以将最后一个单元*重复*任意次。	`buy thing = price [cash | check | credit] [, thing = price [cash | check | credit]]...` 您必须至少购买一件物品，并给出其价格。可以选择一种付款方式：方括号中的项目之一。还可选择购买其它物品：可根据需要购买任意数量的物品。对于要购买的每件物品，给出其名称、价格和付款方式 （可选）。

- 语法语句 （显示命令的语法和所有选项）显示如下：

  sp_dropdevice [*device_name*]

  对于具有多个选项的命令：

  select *column_name*
  from *table_name*
  where *search_conditions*

  在语法语句中，关键字 （命令）采用常规字体，而标识符为小写。斜体表示用户提供的内容。

- 说明 Transact-SQL 命令用法的示例显示如下：

  ```
  select * from publishers
  ```

- 计算机输出的示例如下：

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

(3 rows affected)
```

本手册中的大多数示例都用小写字符显示。不过，输入 Transact-SQL 关键字时可以忽略大小写。例如， SELECT、 Select 和 select 是相同的。

Adaptive Server 是否区分数据库对象 （如表名）的大小写取决于安装在 Adaptive Server 上的排序顺序。通过重新配置 Adaptive Server 的排序顺序，可改变单字节字符集的区分大小写设置。请参见 《系统管理指南：卷 1》。

辅助功能特性

此文档具有针对辅助功能进行了专门设计的 HTML 版本。可以利用适应性技术（如屏幕阅读器）浏览 HTML 文档，也可以用屏幕放大器进行查看。

Adaptive Server HTML 文档已经过测试，符合美国政府 “第 508 节辅助功能”的要求。符合 “第 508 节”要求的文档一般也符合美国之外的辅助功能原则，如针对网站的 World Wide Web 协会 (W3C) 原则。

注释 可能需要配置辅助功能工具，以便获得最佳使用效果。某些屏幕阅读器按照大小写来辨别文本，例如将 “ALL UPPERCASE TEXT”看作首字母缩写，而将 “MixedCase Text”看作单词。对工具进行配置，规定语法约定，您可能会感觉更方便。有关工具的信息，请查阅文档。

有关 Sybase 如何支持辅助功能的信息，请参见位于 http://www.sybase.com/accessibility 的 “Sybase 辅助功能” (Sybase Accessibility)。 Sybase Accessibility 站点包括指向 “第 508 节”和 W3C 标准相关信息的链接。

如果需要帮助

对于购买了支持合同的客户安装的每一个 Sybase 产品，都会有一位或多位指定人员获得与 Sybase 技术支持部门联系的授权。如果使用手册或联机帮助不能解决问题，可让指定人员与 Sybase 技术支持部门联系或与所在区域的 Sybase 子公司联系。

第 1 章

内存数据库

内存数据库完全在命名高速缓存中（即，在 Adaptive Server 内存空间中）运行，而不使用磁盘存储空间来存储数据或日志。因为内存数据库不需要 I/O，所以它的性能可以比传统的磁盘驻留式数据库好得多。内存数据库不是为恢复而设计的：它们的事务日志被写入高速缓存而不是磁盘，因而如果数据库发生故障，任何数据更改都将丢失。内存数据库为运行期回退和其它操作（例如，引发触发器、延迟模式更新、复制等）执行事务性记录。

磁盘驻留式数据库写入到磁盘，以确保保持原子性、一致性、完整性和持久性（统称为 ACID 属性）等事务性属性。持久性是指事务在提交后继续存在。传统的 Adaptive Server 数据库（也称为磁盘驻留式数据库）在提交事务时将其事务日志写入磁盘，从而以完全持久性操作。此数据库与定期写入磁盘的数据页一起来确保所有已提交的事务是持久性的。

内存数据库不将数据或日志写入磁盘，以牺牲对事务持久性的保证为代价来换取性能上的改进。如果发生数据库故障，内存数据库将无法恢复。如果您的应用程序要求在服务器发生故障或正常关闭后保证数据可恢复性，请考虑使用传统的 Adaptive Server 数据库。

通过支持宽松持久性， Sybase 使磁盘驻留式数据库具备内存数据库的性能优点。磁盘驻留式数据库以完全持久性方式操作，以保证从服务器故障中进行事务性恢复。宽松持久性数据库以牺牲已提交事务的完全持久性为代价，来提高事务性负载的运行期性能。

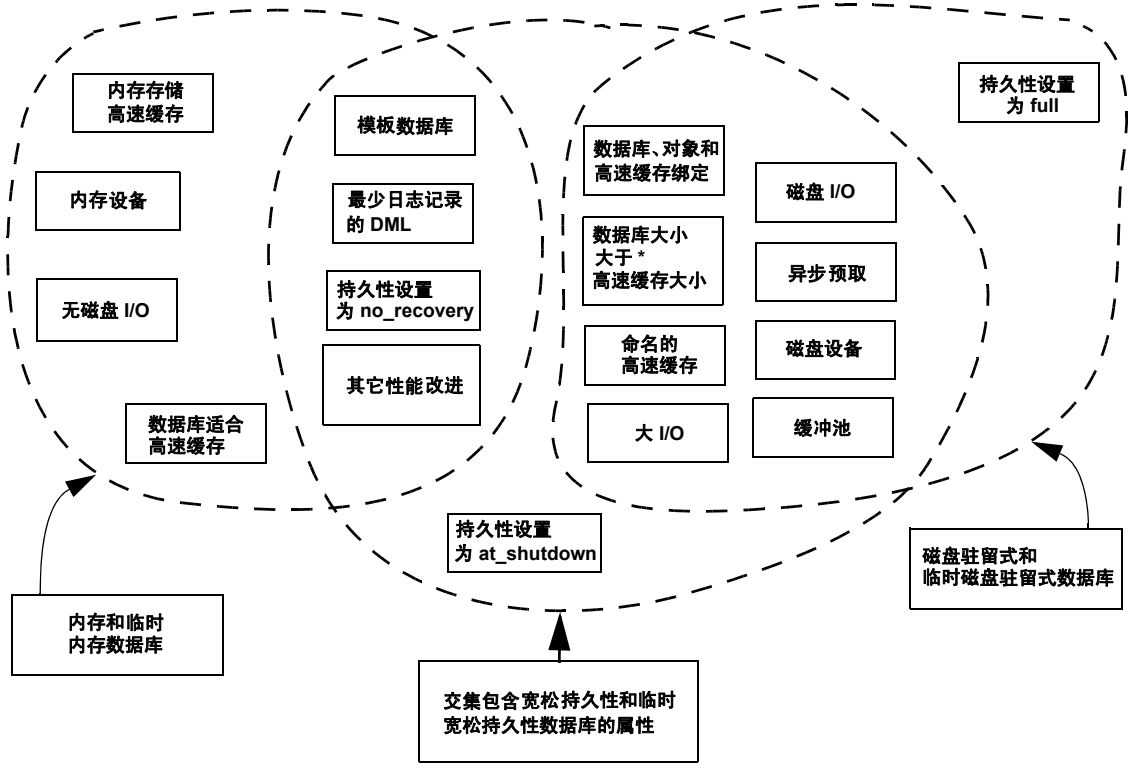
内存数据库和宽松持久性数据库的性能优点包括：

- 内存数据库不等待 I/O。
- 改善了缓冲区和用户日志高速缓存管理，因此，当 Adaptive Server 对相同数据执行并发更新时，您需要增加用户日志高速缓存刷新和缓冲区管理的开销。
- 当事务提交或中止时，运行期策略可以避免在用户日志高速缓存中对事务日志进行刷新。这些策略减少了对内存日志页的争用。
- 支持使用内存日志记录技术进行最少日志记录的 DML 操作，从而提高了大量 DML 操作的性能。

Adaptive Server 15.5 版和更高版本使您可以创建下列类型的数据库 （如下面的图 1-1 所示）：

- 将持久性设置为 full 的磁盘驻留式数据库 （这是缺省的或传统的 Adaptive Server 数据库）
- 用户创建的磁盘驻留式临时数据库
- 将持久性设置为 no_recovery 的内存用户数据库
- 将持久性设置为 no_recovery 的用户创建内存临时数据库
- 将持久性设置为 no_recovery 或 at_shutdown 的磁盘驻留式宽松持久性数据库

图 1-1：内存数据库、宽松持久性数据库和磁盘驻留式数据库的属性



* 大于或等于高速缓存大小

注释　有关在复制环境中使用内存数据库、宽松持久性数据库和 DML 记录的信息，请参见 Replication Server® 文档。

高速缓存和缓冲区支持

承载内存数据库的高速缓存必须足够大，以包含整个数据库，而且该数据库中的每一页都必须驻留在高速缓存中，而不能进行任何缓冲区替换或与磁盘之间进行 I/O 操作。您不能使用已经创建的高速缓存执行下列操作：

- 承载内存数据库以绑定其它数据库或其它对象。
- 绑定其它数据库或对象以承载内存数据库。用于承载内存数据库的命名高速缓存使用不同的结构，并且专用于内存数据库。

使用 sp_cacheconfig 可以为内存数据库创建高速缓存。使用 disk init 可以将高速缓存划分为内存设备，它们类似于磁盘设备并支持段。您可以将一个或多个逻辑段绑定到内存设备，从而将对象绑定到单个段。

在将任何对象绑定到内存数据库或宽松持久性数据库的高速缓存之前，请考虑以下方面：

- 使用命名高速缓存绑定整个宽松持久性或完全持久性数据库。您可以：
  - 将宽松持久性数据库中的各个对象绑定到命名高速缓存，类似于绑定常规数据库中的各个对象
  - 将宽松持久性数据库绑定到命名高速缓存 （例如，缺省数据高速缓存）
  - 内存存储高速缓存类似于命名高速缓存，但是为实现有效的内存访问而进行了配置。将宽松持久性数据库中的各个对象绑定到不同的高速缓存
- 内存存储高速缓存的高速缓存行为类似于常规高速缓存的高速缓存行为。
- 使用用于提高命名高速缓存性能的相同监控工具和调优技术可提高绑定到命名高速缓存的宽松持久性数据库的性能。
- 因为单个内存存储高速缓存承载整个数据库，所以不要将数据库或对象绑定到各个高速缓存。大多数现有高速缓存管理器监控和调优技术都适用于内存高速缓存。请参见 Performance and Tuning Series: Monitoring Adaptive Server with sp_sysmon （《性能和调优系列：使用 sp_sysmon 监控 Adaptive Server》）中的第 2 章“Monitoring Performance with sp_sysmon”（使用 sp_sysmon 监控性能）。

内存数据库必须由单个高速缓存承载，但可以驻留在从该高速缓存创建的多个内存设备中。图 1-2 显示了 imdb_cache 高速缓存，它包含一个内存存储设备 imdb_dev：

图 1-2：承载单个设备的单个高速缓存

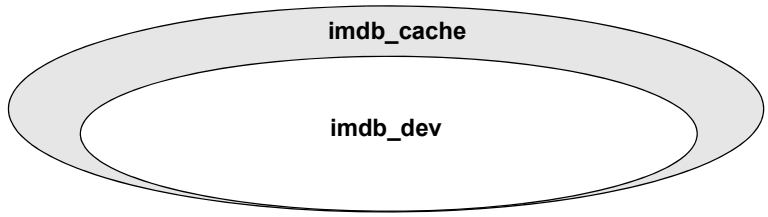
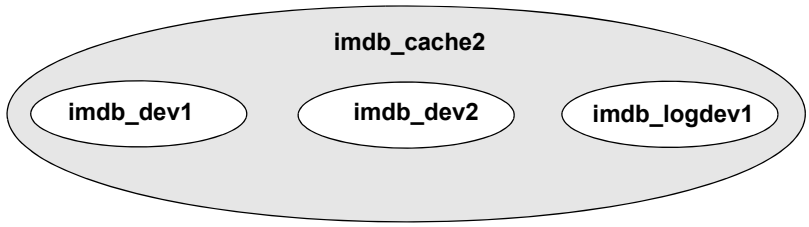


图 1-3 显示了 imdb_cache2 高速缓存，它包括两个内存数据设备（imdb_dev1 和 imdb_dev2）和一个日志设备 (imdb_logdev1)：

图 1-3：承载多个设备的单个高速缓存



使用 create inmemory database 可直接在逻辑设备上创建内存数据库。如果您的安装不使用段来限制对象的空间，请在要用于整个内存存储高速缓存的内存设备上创建数据库。为了在各个对象的空间管理以及阈值过程支持中实现更精细的粒度，请在单独的内存设备上创建数据库和日志。

使用 ddlgen 可为内存数据库和宽松持久性数据库生成数据库和对象定义。请参见 《实用程序指南》。

使用 alter database 可更改现有内存数据库的布局。请参见 《参考手册：命令》。

持久性级别

系统发生故障或执行 shutdown with nowait 后重新启动服务器，在已提交的持久性事务中更改的数据仍然存在。将事务日志和数据库页刷新到磁盘的操作使传统磁盘驻留式数据库中的事务具有了持久性。在服务器发生故障或将服务器强行关闭后，内存数据库不提供事务性持久性。

宽松持久性数据库提供了两个级别的持久性，以供您从中进行选择。第一个级别的持久性类似于内存数据库：如果服务器发生故障，则数据丢失。第二个级别的持久性介于磁盘驻留式数据库的持久性和内存数据库的持久性之间：只有正常关闭服务器，才会完成所有事务并将其永久存储到磁盘中。这使宽松持久性数据库能够利用内存数据库的很多性能优化特性。

如果将数据库的持久性设置为 no_recovery 或 at_shutdown，则无论它们是内存数据库还是磁盘驻留式数据库，都将它们称为低持久性数据库。低持久性数据库中的数据在提交后仍然存在 （前提是您没有重新启动服务器）。

使用 create database with durability=*durability_level* 可设置数据库的持久性级别。Adaptive Server 支持 full、no_recovery 和 at_shutdown 持久性级别。请参见 《参考手册：命令》。

表 1-1 介绍了在每个持久性级别可以执行的操作：

表 1-1：数据库的持久性级别

操作	no_recovery	at_shutdown	full
创建磁盘驻留式数据库	是	是	是
创建内存数据库	是	否	否
运行期回退	是	是	是
故障恢复	否	否	是
在服务器正常关闭和重新启动之后恢复数据库。	否	是	是
将数据库转储到存档或从存档装载数据库。	是	是	是
将事务日志从设备转储到存档。将事务日志从存档装载到设备。	否	否	是

减弱的持久性和提高的性能仅适用于使用 at_shutdown 和 no_recovery 的宽松持久性数据库。您可以将宽松持久性数据库绑定到命名高速缓存，命名高速缓存的大小可以小于数据库大小。

临时数据库和内存临时数据库

临时数据库隐式使用 no_recovery 持久性。您可以用显式 no_recovery 持久性创建临时数据库以增强性能，并且可以在升级后改变现有临时数据库，以将其持久性显式设置为 no_recovery。

您完全在内存中使用的临时数据库显式使用 no_recovery 持久性。

除了使您可以管理缺省的 tempdb 组以外，Adaptive Server 还使您可以创建和管理用户创建的 tempdb 组。用户创建的 tempdb 组可能包括其它用户创建的临时数据库，并支持应用程序和登录名绑定。

您不能从 default 临时数据库组中删除系统 tempdb。您也不能将系统 tempdb 添加到用户创建的任何其它 tempdb 组中。

临时数据库组可以混合磁盘驻留式或内存临时数据库。

您可以指定和管理用户创建的 tempdb 组，以仅包含磁盘驻留式数据库或仅包含内存临时数据库。虽然 Adaptive Server 没有显式施加任何限制，但通过控制 tempdb 组的成员资格，您可以将特定的仅磁盘或仅内存 tempdb 组分配给特定的登录名或应用程序。

多数据库事务和数据库类型

数据库的持久性影响跨越多个数据库的事务。

低持久性数据库 （即用户创建的临时数据库、内存数据库或磁盘驻留式宽松持久性数据库）可以参与多数据库事务，即使协调数据库使用完全持久性也不例外。但是，如果协调数据库使用低持久性，则不允许跨越其它完全持久性数据库的事务。

表 1-2：数据库参与多数据库事务

协调数据库	参与者数据库	多数据库事务
完全持久性数据库	临时、内存、宽松持久性或临时内存数据库	允许
临时、内存、宽松持久性或临时内存数据库	临时、内存、宽松持久性或临时内存数据库	允许
临时、内存、宽松持久性或临时内存数据库	完全持久性数据库	不允许

模板数据库

您可以使用除 model 以外的数据库作为内存数据库的模板，以便将参考数据 （例如，表和存储过程）装载到用模板创建的数据库中。模板数据库必须是具有 full 持久性的现有磁盘驻留式用户数据库。只有在低持久性数据库中，才支持使用模板创建数据库和最少日志记录 DML。

您不能使用模板创建使用 at_shutdown 持久性的宽松持久性数据库。您只能对具有 no_recovery 持久性的数据库使用模板。

使用 create database use *database_name* as template 命令可指定除 model 以外的数据库作为任何已经用 no_recovery 参数创建的数据库的模板。

在创建使用除 model 以外的数据库作为其模板的数据库之后，如果服务器重新启动， Adaptive Server 将使用该模板数据库中的数据重新创建相关数据库。

当您重新启动 Adaptive Server 后，模板数据库将在使用它们作为模板的数据库之前恢复。如果 Adaptive Server 无法恢复模板数据库，则无法重新创建依赖该模板数据库的数据库。

当您创建相关数据库时， Adaptive Server 将应用模板数据库的属性。您将其指定为 create database 命令的一部分的属性将覆盖模板数据库的属性。对于您使用模板创建的数据库而言，当 Adaptive Server 重新启动后，以及 Adaptive Server 从它的模板重新创建该数据库后，该数据库的选项和属性将继续存在。当 Adaptive Server 在后续的重新启动过程中重新创建相关数据库时，将不会使用您对模板数据库的属性所做的任何更改。

您不能删除被任何数据库用作模板的数据库。您必须首先删除所有使用此数据库作为模板的数据库，或使用 alter database 命令将模板数据库与其所有相关数据库进行分离，然后才能删除被其它数据库用作模板的数据库。

运行 sp_helpdb 可报告有关下列数据库的模板的信息：

- 用户数据库 — 确定是否已将该用户数据库用作模板，以及将其用作哪些数据库的模板。 sp_helpdb 输出的以下部分说明 pubs2 数据库被用作 pubs3 和 pubs5 内存数据库的模板：

  ```
  template_for
  --------------------------------------------------
  pubs3
  pubs5
  ```

- 从模板创建的数据库 — 确定哪个数据库在创建过程中被用作模板（如果不是 model 数据库）。 sp_helpdb 输出的以下部分说明您将 pubs2 数据库用作 pubs3 数据库的模板：

  ```
  template
  --------------------------------------------------
  pubs2
  ```

改变数据库以使用新模板

当您使用 alter database 更改模板时， Adaptive Server 不会更改数据库的现有数据。当您重新启动 Adaptive Server 时， Adaptive Server 会使用新模板的数据来重新创建该数据库。

您只能更改内存数据库的模板以及使用 no_recovery 持久性的数据库的模板；您无法更改系统数据库的模板或使用 full 持久性的传统磁盘驻留式数据库的模板。

进行最少日志记录操作的命令

在 Adaptive Server 15.5 版本中，您可以基于数据库、表和会话为数据操作语言 （DML 命令）执行最少日志记录，以便记录最少的行和页更改，以及页的分配和解除分配。

通过配置 create database、 alter database、 create table、 select into、 set DML logging 和 alter table 等命令，可以在数据库、表和会话级别控制 DML 日志记录。

请参见第 3 章 “最少日志记录 DML”。

内存数据库和宽松持久性数据库的限制

内存数据库和宽松持久性数据库包括下列限制：

- Cluster Edition 当前不支持内存数据库、宽松持久性数据库、模板数据库或进行最少日志记录操作的 DML。
- Replication Server 当前不支持内存数据库或将持久性设置为 no_recovery 的数据库的复制。
- 内存数据库不支持使用兼容模式进行查询。在 Adaptive Server 中，请仅对常规磁盘驻留式数据库表使用兼容模式。有关启用兼容模式的信息，请参见 《迁移技术指南》。

  如果您已经启用了兼容模式并且查询涉及到内存数据库中的表，则 Adaptive Server 会使用 “受限兼容模式”恢复到本机 15.0 版查询优化程序和执行引擎。通常情况下，此模式会产生与 Adaptive Server 12.5 版本中的计划类似的查询计划。如果您注意到性能下降， Sybase 建议您对此查询禁用兼容模式。

- 您不能在用于增加数据库大小的同一命令中更改内存数据库或低持久性数据库的持久性或日志记录级别。
- 在使用 alter database 命令更改数据库的 durability 或 minimal_logging 属性时，该命令会自动将该数据库置于单用户模式，而如果该命令无法获得对数据库的独占访问权，该命令就会失败。为了避免失败，请在运行 alter database 之前手动将该数据库置于单用户模式。
- 内存数据库和宽松持久性数据库无法参与分布式事务。
- 内存数据库和宽松持久性数据库无法协调涉及到具有完全持久性的数据库的多数据库事务。如果您执行的系统过程在您从内存数据库或宽松持久性数据库运行存储过程时执行事务性更新，则会出现以下错误：

  ```
  Msg 3952, Level 16, State 2:
  Procedure 'sp_XX', Line 258:
  Command not allowed.You cannot start a multidatabase
  operation in database 'master'
  after starting the master transaction in 'imdb1' as
  it may render the database 'master' unrecoverable.
  ```

  运行系统过程：

  a 从完全持久性数据库中执行该系统过程。例如，若要在 master 中运行 sp_XX，请输入：

  ```
  use master
  go
  exec sp_XX
  ```

  b 使用以下格式从当前数据库 （内存数据库或宽松持久性数据库）中引用该存储过程：`database_name.owner.`***`sp_name`***。

  例如，若要在 imdb_1 内存数据库中运行 sp_XX，请输入：

  ```
  use imdb_1
  go
  exec master.dbo.sp_XX
  ```

- 在内存数据库中，您无法使用代理表或数据库映射到其它内存数据库或磁盘驻留式数据库对象。
- 在磁盘驻留式数据库中，您无法使用代理表或数据库映射到内存数据库或表。

已更改的系统过程

表 1-3 列出了已经更改以支持内存存储高速缓存、内存设备和内存数据库的存储过程。

表 1-3：为内存数据库更改的系统过程

系统过程	注释
sp_addsegment	已更新以管理内存数据库中的空间。
sp_addthreshold	已更新以管理内存数据库中的空间。
sp_bindcache	您无法将对象或数据库绑定到内存存储高速缓存，并且无法将内存数据库或内存数据库中的对象绑定到任何高速缓存。
sp_cacheconfig	创建、删除内存存储高速缓存或扩大其大小。
sp_cachestrategy	prefetch 和 MRU 参数不适用于内存数据库中的表和索引。
sp_dbextend	对于内存数据库，当前不支持自动数据库扩展。
sp_deviceattr	directio 和 dsync device 属性不适用于内存设备。
sp_diskdefault	不能使用 sp_diskdefault 将内存设备指定为缺省设备。
sp_downgrade	支持将包含内存数据库或宽松持久性数据库或包含使用模板或最少日志记录的数据库的 Adaptive Server 降级到以前的版本。
sp_dropdevice	删除从内存存储高速缓存创建的内存设备。
sp_dropsegment	已更新以管理内存数据库中的空间。
sp_dropthreshold	已更新以管理内存数据库中的空间。
sp_extendsegment	已更新以管理内存数据库中的空间。
sp_help	报告表的属性，例如最少日志记录属性。
sp_helpcache	显示内存存储高速缓存的属性、在此高速缓存中创建的内存数据库以及此高速缓存中可用空间的详细信息。
sp_helpdb	报告数据库的属性，例如，持久性、 DML 日志记录级别、是否是内存数据库、模板数据库 （如果有）的使用或是否用作模板数据库。
sp_helpdevice	报告从内存存储高速缓存创建的内存设备属性。
sp_modifythreshold	已更新以管理内存数据库中的空间。
sp_plan_dbccdb	为内存数据库中的 checkstorage 执行设置 dbccdb。
sp_poolconfig	内存数据库不支持大型 I/O 缓冲池。
sp_post_xpload	支持内存数据库的跨平台操作。
sp_tempdb	支持内存临时数据库的登录名或应用程序绑定。
sp_unbindcache、sp_unbindcache_all	不能从宿主内存存储高速缓存中解除对象与内存数据库本身的绑定。

第 2 章

管理内存数据库和宽松持久性数据库

为内存数据库指定命名高速缓存

Sybase 建议您对内存存储高速缓存使用巨内存页。请参见所用平台的 Configuration Guide （《配置指南》）。

容纳内存数据库的高速缓存必须足够大，以包含整个数据库。包含内存数据库的高速缓存称为内存存储，并且：

- 在运行期操作过程中禁用 I/O
- 禁用缓冲区清洗
- 缓冲区替换和清洗

在创建内存存储高速缓存后，可将其划分为一个或多个区段，其中每个区段都容纳一段数据库或日志。请参见第 4 页的 “高速缓存和缓冲区支持”。

使用带 inmemory_storage 参数的 sp_cacheconfig 可创建内存存储高速缓存。请参见 《参考手册：过程》。

注释 在创建内存存储高速缓存之前，请检验 max memory 的值对于指定的高速缓存大小而言是否足够大。如果 max memory 不够大，Adaptive Server 会发出错误消息。

例如，若要创建一个名为 imdb_cache 的内存存储高速缓存，请输入：

```
sp_cacheconfig imdb_cache, '2G', inmemory_storage
```

注释 对于常规命名高速缓存，如果可用内存大小小于高速缓存的请求内存大小，则 Adaptive Server 会用减小的内存大小创建该高速缓存。这意味着高速缓存被成功创建，但具有较小的大小。但是，如果没有足够的空间来创建内存数据库的高速缓存，则该命令将失败。

检验配置文件的更改

检验配置文件 (*$SYBASE/server_name.cfg*) 是否正确指定了内存存储高速缓存信息。每个内存存储高速缓存都在配置文件中包括一个标记为“Named Cache: *cache_name*”的标头。`Named Cache` 条目包括：

- `cache size` — 高速缓存的大小必须足够大，以容纳整个内存数据库。
- `cache status` — 设置为 “in-memory storage cache”。
- `cache replacement policy` — 设置为 “DEFAULT”或 “none”。
- `local cache partition number` — 本地高速缓存分区数或 “DEFAULT”。

一个名为 imdb_cache 的高速缓存的条目如下所示：

```
[Named Cache:imdb_cache]
cache size = 2G
cache status = in-memory storage cache
cache replacement policy = none
local cache partition number = 8
```

更改内存数据库的静态配置参数

Adaptive Server 在您重新启动它时对静态配置参数进行更改。因为内存数据库在您重新启动服务器后将重新创建 （而且对这些数据库进行的所有更改都将丢失），所以请执行以下任一操作，以确保内存数据库中的数据在您重新启动 Adaptive Server 后不会丢失：

- 在创建内存数据库之前进行所有静态配置更改，或者，
- 将内存数据库转储到存档，进行静态配置更改，重新启动服务器，然后从存档中装载内存数据库。

创建内存设备

使用 disk init 可将内存存储高速缓存划分为多个较小的区段，这些区段称为内存设备，用于创建内存数据库。 Sybase 建议您对用于磁盘驻留式设备的内存设备使用相同的命名约定。将用户或系统定义的段与内存设备逻辑名进行绑定可以控制绑定到这些段的对象的空间使用情况。

注释　您不能使用常规命名高速缓存来创建内存设备。也就是说，您必须使用 disk init 的 type=inmemory 参数来创建内存设备。

创建内存设备的语法为：

```
disk init name = device_name
   physname = {"physical_name" | "cache_name"}
. . .
[, type = "inmemory"]
```

其中，*device_name* 是内存设备的逻辑名，*cache_name* 是用 sp_cacheconfig 创建的内存存储高速缓存的名称，而 inmemory 表示该设备用于内存数据库。

例如：

```
disk init name = imdb_cache_dev,
physname = "imdb_cache" ,
size = "50M",
type = "inmemory"
```

创建内存数据库

使用 create inmemory database 可创建内存数据库，并且可使用 model 或其它用户数据库作为它的模板。

您还可以将临时数据库创建为完全驻留在内存存储中的内存数据库。但是，您不能为内存临时数据库指定模板数据库。

当您使用用户数据库模板创建内存数据库时，所有的用户、权限、对象和过程都将从模板数据库复制到该内存数据库中。

当您创建内存临时数据库时：

- 将 guest 用户添加到临时数据库。
- 将 create table 特权授予 public。

您不能将系统数据库（例如，sybsecurity）创建为内存数据库，因为一旦 Adaptive Server 发生故障，必须更新这些数据库。

当您创建内存数据库时，您可以指定基于数据库、对象或会话对 insert、update、 delete 和某些批量拷入操作进行最少或完全日志记录。请参见第 3 章 “最少日志记录 DML”。

本示例在 2GB 内存设备上创建一个内存数据库。 with override 使您可以在同一个内存设备上创建数据和日志段（对于内存数据库，所支持的唯一持久性级别是 no_recovery：试图使用其它持久性级别会导致错误）：

```
create inmemory database imdb1
on imdb_data_dev1 = '1.0g'
log on imdb_data_dev1 = '0.5g'
with override, durability = no_recovery
```

用宽松持久性创建磁盘驻留式数据库

将宽松持久性数据库的持久性级别设置为 no_recovery 或 at_shutdown。请参见第 6 页的 “持久性级别”。

您可以在现有磁盘上创建宽松持久性数据库。宽松持久性数据库使用磁盘存储，因此您必须使用 disk init 创建它所驻留的设备。

以下示例用持久性级别 at_shutdown 创建 pubs6 数据库：

```
create database pubs6
on pubs6_dev
with override, durability=at_shutdown
```

管理内存数据库

在创建内存数据库之后，可以通过调整内存存储高速缓存的大小和将其删除、增加它们的大小以及执行转储和装载来对其进行管理。

调整内存存储高速缓存的大小

使用 sp_cacheconfig 可在运行期增加内存存储高速缓存的大小。例如，若要将 imdb_cache 的大小增加到 3GB，请输入：

```
sp_cacheconfig imdb_cache, '3G', inmemory_storage
```

您可以使用相同的过程减小内存存储高速缓存的大小。您只能按照高速缓存中当前未被内存数据库使用的空间量来减小内存存储高速缓存的大小，换句话说，内存存储高速缓存不能小于内存数据库。

重新启动服务器后，高速缓存的大小会减小。

删除内存存储高速缓存

在删除内存存储高速缓存之前，必须删除所有的设备和内存数据库。通过将内存存储高速缓存的大小设置为零将其删除：

```
sp_cacheconfig imdb_cache, '0g'
```

增加内存数据库的大小

使用 alter database 可增加内存数据库的大小。您要增加其大小的设备所属的高速缓存必须与当前承载该数据库所驻留设备的高速缓存相同（也就是说，您不能创建一个附加内存存储高速缓存并增加此存储高速缓存上的现有内存数据库的大小）。

例如，若要将 pubs5 数据库的大小增加 50MB，请输入：

```
alter database pubs5
on imdb_cach = '50M'
```

如果现有的所有内存存储高速缓存都正在使用，并且您无法创建更多高速缓存，则不能使用 disk resize。而应增加高速缓存的大小，在此空间上创建一个新的内存存储高速缓存，然后通过 alter database 添加使用该高速缓存的空间。

转储和装载内存数据库

使用 dump database 和 load database 可转储到内存数据库和宽松持久性数据库的存档设备，或从该存档设备进行装载。转储和装载内存数据库或宽松持久性数据库不需要对 dump 或 load 命令使用特殊参数。

执行 load database 命令将数据库装载到内存数据库中时，可将数据直接装载到内存存储高速缓存中。

您可以：

- 跨持久性级别转储和装载数据库。例如，您可以将持久性级别为 full 的数据库转储到内存数据库中，后者的持久性级别始终为 no_recovery
- 跨平台执行转储和装载。

您不能：

- 从内存数据库或宽松持久性数据库执行 dump transaction，原因是 load transaction 不能使用可能包含未排序的日志记录的事务日志来执行所需的任务。
- 从将持久性设置为 no_recovery 或 at_shutdown 的数据库中将转储信息装载到不支持内存数据库或宽松持久性数据库的 Adaptive Server 版本中。但是，您可以将数据库转储信息从早期版本的 Adaptive Server 装载到内存数据库或宽松持久性数据库中。

注释 您必须使用您用于转储和装载内存数据库或宽松持久性数据库的 Adaptive Server 版本所附带的 Backup Server 版本。

配置 *number of backup connections*

转储和装载内存数据库要求 Backup Server 连接到 Adaptive Server。load 命令使用与设备数相同的连接数。 dump 命令使用的连接数等于设备数加上附加连接数。使用 number of backup connections 可配置 Backup Server 可以使用的最大用户连接数。

请参见 《系统管理指南，卷 1》中的第 5 章 “设置配置参数”。

Backup Server 需要具备下列条件才能连接到 Adaptive Server：

- Backup Server interfaces 文件必须包含有关 Adaptive Sever 的条目。
- Backup Server 用于连接到 Adaptive Server 的用户名和口令与执行 dump 和 load 命令的用户相同。如果连接是使用安全外部鉴定 （例如，Kerberos）建立的，则 Backup Server 不能从 Adaptive Server 检索口令令牌。请使用 sp_remotelogin 为 SYB_BACKUP 定义受托远程登录名，否则 Backup Server 会收到鉴定失败消息。

删除内存数据库

使用 drop database 可删除内存数据库。以下示例删除 pubs6 数据库：

```
drop database pubs6
```

删除内存数据库会将数据库从系统表中删除，并会放弃内存存储高速缓存，但这些设备中的缓冲区和数据仍然保留下来。删除内存数据库不会影响用来创建它的内存存储高速缓存，该高速缓存仍然可以用于其它内存数据库。

删除内存设备

使用 sp_dropdevice 可删除内存设备。仅当内存设备当前未被任何数据库使用时， sp_dropdevice 才能删除它。必须首先删除数据库，然后才能删除内存设备。删除内存设备会从 sysdevices 中删除它的条目，并且将内存归还给用于创建它的高速缓存。此后，您可以将该部分高速缓存用于任何其它目的，包括创建新的内存设备。以下示例删除名为 pubs6_device 的设备：

```
sp_dropdevice 'pubs6_device'
```

第 3 章

最少日志记录 DML

为了优化被刷新到磁盘上的事务日志的日志记录， Adaptive Server 可以在对所有类型的低持久性数据库（例如，内存数据库以及使用 at_shutdown 和 no_recovery 选项的低持久性数据库）执行某些数据操作语言 (DML) 命令 （insert、 update、 delete 和 slow bcp）时执行最少日志记录操作，甚至不执行日志记录操作。您可以基于数据库、表和会话为 DML 执行最少日志记录操作。

DML 日志记录设置的类型

DML 日志记录设置的控制层次为：

- 数据库级日志记录 — 缺省情况下，在数据库级为所有表启用 DML 的日志记录。 DML 日志记录设置仅影响用户和临时表。
- 表级日志记录 — 覆盖在数据库级设置的日志记录级别，具体取决于表是如何创建或改变的。
- 会话级日志记录 — 覆盖在表和数据库级设置的日志记录级别。

数据库级日志记录

对启用和禁用日志记录功能的数据库级支持主要是为临时数据库提供的，在这种情况下，所有不依赖于日志记录功能的应用程序都可以更有效地运行，而无需更改创建临时表的应用程序或过程的代码。

您不能更改系统数据库 （包括 model 数据库）的日志记录模式。您可以将系统 tempdb 和任何用户临时数据库的 DML 日志记录模式更改为最少日志记录模式。在此之前， Sybase 建议您了解更改为最少日志记录模式会对所有使用临时表的应用程序的回退语义产生什么样的影响。请参见第 26 页的 “事务性语义”。

只有在将持久性级别设置为 no_recovery 或 at_shutdown 的数据库中，才允许最少日志记录 DML。您必须将数据库的 select into 选项设置为 on 才能使最少日志记录生效。

您只能从 master 数据库中改变数据库的缺省日志记录模式。此外，您要改变的数据库必须满足下列条件：

- 处于用户数据库的单用户模式
- 对于临时数据库，将 dbo-use only 设置为 true (on)，以便只有数据库所有者可以使用数据库

如果数据库尚未处于必需的模式，则服务器会尝试将数据库置于该模式。如果尝试不成功，服务器会引发错误，并提示用户显式将数据库置于其正确模式。

命令

在数据库级更改 DML 日志记录模式的语法是：

```
create [temporary] database database_name
    [on {default | database_device [= size]
                [, {database_device [= size]...]
    [log on {database_device [= size]
          [, {database_device [= size]]...]
    [with {override | default_location = "pathname"
    | [[,]durability = { no_recovery | at_shutdown | full} ]
    | [[,]dml_logging = {full | minimal} ]
    }...
]
    [for {load | proxy_update}]
```

若要在现有数据库中更改 DML 日志记录的数据库级设置，请使用：

```
alter database dbname
set dml_logging = {full | minimal}
```

请参见 《参考手册：命令》。

表级日志记录

在表级设置 DML 日志记录选项会覆盖数据库级设置，具体取决于创建或改变表的方式。缺省模式是使用与数据库相同的设置。

如果没有在数据库级启用最少日志记录，则：

- 为通过 create table 或 alter table 将 dml_logging 显式设置为 minimal 的表执行最少 DML 日志记录。
- 为所有其它表执行完全日志记录 DML。

如果在数据库级启用了最少日志记录，则：

- 为将 dml_logging 显式设置为 full 的表执行完全 DML 日志记录。
- 为所有其它表执行最少 DML 日志记录。

表的 default 日志记录设置使其可以在执行 DML 时继承日志记录功能的当前数据库级设置。数据库管理员可能希望在数据库级定期禁用日志记录功能，然后重新启用该功能。之后，只需要通过显式表级设置控制那些对 full 或最少 DML 日志记录具有特定需要的表。

仅当数据库启用 select into 数据库选项后，您才能对表执行最少日志记录 DML 命令；否则，将完全记录所有的 DML 命令。

对于已启用相应触发器 （例如，在执行 insert statement 时启用了 insert 触发器）的表上的任何 DML 语句， Adaptive Server 将执行完全日志记录。这避免了以下情况的发生，即触发器实现了业务规则和安全机制，而这些业务规则和安全机制在执行该触发器时需要日志记录。若要执行最少日志记录，在执行 DML 语句之前必须由表所有者禁用特定的触发器。

如果对可以更新的视图执行了 DML 语句，而该语句最终解析为适合于最少日志记录 DML 操作的基表上的 DML 语句，则对该视图执行的 DML 语句会导致该基表上的最少日志记录 DML。若要控制通过视图更新的基表上的日志记录模式，请使用 alter table 或 set dml_logging 设置基础表上的日志记录模式。

您可以使用 alter table 仅更改用户表的日志记录模式，而不更改视图或其它对象的日志记录模式。

命令

若要创建具有完全或最少 DML 日志记录模式的表，请使用：

```
create table tablename (
      <rest of the column list specifications>
)
lock lock_scheme
with { max_rows_per_page = num_rows
      , exp_row_size = num_bytes
      , reservepagegap = num_pages
      , identity_gap = value
      , dml_logging = {full | minimal}
      }
on segment_name
```

若要使用 select into 创建一个表以使目标表启用或禁用 DML 日志记录，请使用：

```
select <column list>
into table_name
      [ <external table specifications> ]
      on segment_name
      [ partition_clause ]
      lock lock_scheme
      with { max_rows_per_page = num_rows
            , exp_row_size = num_bytes
            , reservepagegap = num_pages
            , identity_gap = value
                  , dml_logging = {full | minimal}
            }
[ from_clause ]
[ where_clause ]
...
```

注释 在 15.5 版本中，如果表所有者在创建表时显式禁用了日志记录功能，则不能无条件地启用该表的日志记录功能。这可以防止大型 DML 在所创建的拟使用最少日志记录操作的表上生成数量巨大的日志记录操作。

会话级日志记录

特定会话的日志记录选项设置会覆盖表级和数据库级的日志记录选项设置。

若要在当前会话中启用或禁用 DML 日志记录功能 （甚至当数据库级或表级 DML 日志记录设置为 full 时），请使用：

```
set dml_logging = { full | minimal | default}
```

请参见 《参考手册：命令》。

将 DML 日志记录功能设置为 minimal 只会影响当前会话用户拥有的对象的日志记录模式。如果会话用户具有 sa_role，则所有用户对象的日志记录模式为 minimal。

一旦您将会话特定的 DML 日志记录功能设置为 minimal，set dml_logging default 就会根据表级和数据库级设置，将受影响表当前有效的日志记录模式恢复为该表的缺省日志记录模式。

您可以使用 set dml_logging 命令对表执行最少日志记录，但是，如果数据库所有者或表所有者已经将该表设置为以最少日志记录模式运行，则不能使用该命令执行完全日志记录 DML。

会话特定的日志记录模式设置将最终确定是否日志记录特定对象，但需要遵循前文及下列各节中介绍的各种规则，并且与特定表的日志记录模式的选择有关。

- 在进行成功的 set 命令调用之后，当前会话中的所有表都成为选择日志记录模式时的考虑对象，具体取决于对表拥有的权限和特权。
- 您只能使用 set 命令更改用户表的日志记录模式，而不能更改系统表、视图等其它对象的日志记录模式。
- 最少 DML 日志记录模式要求启用 select into 数据库选项，而这需要数据库所有者或 sa_role 特权。

其它最少日志记录规则

除了与数据库、表和会话相关的基本规则以外，下列规则也会影响最少日志记录模式：

- 您只能在内存数据库或宽松持久性数据库中使用最少日志记录 DML。而不能在具有完全持久性的数据库中使用它们。
- 表在多语句事务中的日志记录模式将在整个事务中保持不变。
- 对于参与声明性或逻辑参照完整性约束的表，将完全记录所有的 DML 命令。
- 您可以将 set dml_login 选项从登录触发器导出到客户端会话。但是，与多数 set 选项不同的是，您不能从存储过程导出 set dml_logging 或向其调用方发出 execute immediate，即使您启用了 set export_options 也不例外。
- 设置保存点后，所有的 DML 命令都以 full 日志记录模式运行，即使该表之前符合最少日志记录要求也是如此。
- 如果表上存在任何活动的触发器，则执行完全日志记录。为了使 DML 在最少日志记录模式下运行，请在执行 DML 语句之前禁用所有触发器。
- 选择延迟模式更新处理的优化程序会覆盖最少 DML 日志记录设置，而 DML 将以 full 日志记录模式执行。
- 为了支持基于日志的复制，复制表上的 DML 总是执行 full 日志记录。

事务性语义

在以最少日志记录模式操作时，在运行期回退之后不能保证事务的原子性。

因为运行期的日志记录操作是不完整的，所以无法完整回退由失败的命令进行的更改。所有的更改都将以提交模式应用于数据库。例如，您不能回退删除很多行的事务；对已删除的行进行的更改已经提交。如果事务从随后已经解除分配的页中删除行，则该页仍然保持解除分配状态。

但是，未对更改进行日志记录不会妨碍将受影响的行或页锁定。即使 DML 命令以无法回退的最少日志记录模式运行，也能够获取受影响行和页上的锁并一直保持到事务结束，并且仅在事务完成后释放该锁，而无论该事务被 rollback 还是被 commit。

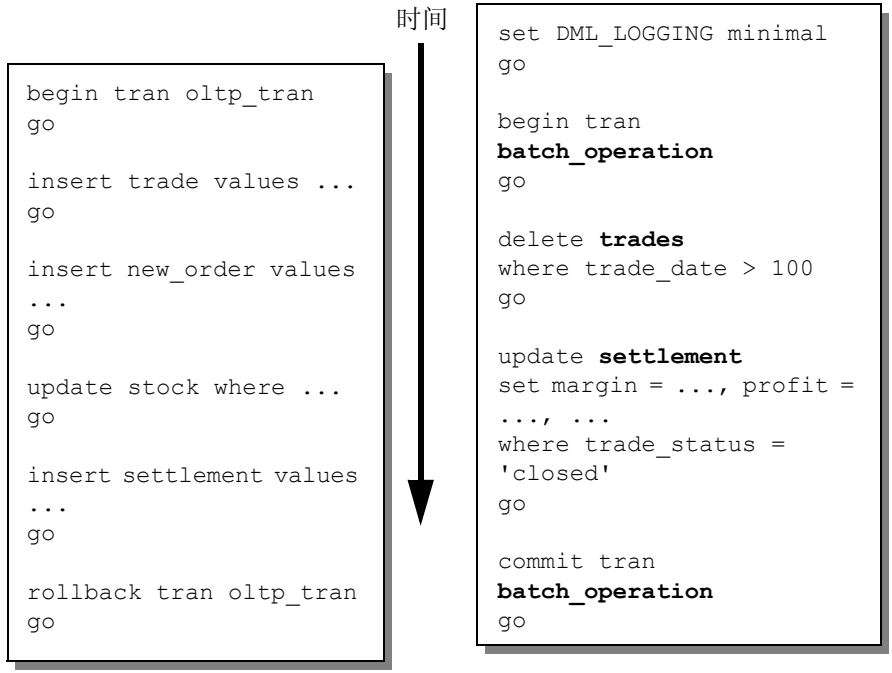
如果在以最少日志记录模式对一个或多个表执行事务时发出 rollback 命令，则将引发警告，说明 Adaptive Server 在遇到回退操作时正在提交事务。

日志记录并发事务

set dml_logging 命令只影响当前会话中的日志记录模式，并且只影响会话用户所拥有的表上的 DML 语句的日志记录模式。这使 DML 命令可以从多个事务中并发执行，其中，在一个与其它会话中的 minimal 日志记录操作并发执行的会话中利用了 full 日志记录模式。以 full 日志记录模式执行的会话中的任何回退都被撤消，但是，更改在另一个会话的回退中未被撤消。

这种用法的一个常见示例是事务性系统的执行，该系统与以最少日志记录模式执行的大型批处理更新或删除操作并发地运行小事务。任意一个会话中的任何错误都不会影响另一个会话的事务一致性。在下面的示例中，OLTP 事务以 full 日志记录模式运行，支持完全可恢复性，而批处理操作以最少记录模式在另一个会话中运行。

图 3-1：采用不同日志记录模式的事务的并发执行



将 *ddl in tran* 设置为 true 的最少日志记录模式

如果将 ddl in tran 数据库选项设置为 true，则对于已经在相同事务中对其执行了任何 DDL 操作的表，不能再以最少日志记录模式对其执行 DML 语句。该事务中的 DML 命令以 full 日志记录模式执行。

在以下事务中，sp_dboption 数据库选项 ddl in tran 被设置为 true，而表 t1 被用最少 DML 日志记录模式进行了配置。但是，在运行期， Adaptive Server 以完全日志记录模式执行 drop index 命令后面的 DML 命令，因为它以完全日志记录模式执行 drop index 命令：

```
sp_dboption pubs1, 'ddl in tran', on
go

begin tran
go

update t1 set ...
go

drop index t1.ind1
go

insert t1 values ...
go

insert t1 values ...
go

delete t1 where ...
go

update t1 where ...
go

rollback tran
go
```

在相同事务中执行时， create index 和 drop index 命令将影响如何选择最少日志记录模式（也就是说，这两个命令中的任何一个都会导致 Adaptive Server 在上述脚本中执行完全日志记录而不是最少日志记录）。

参照完整性约束的影响

当参照完整性约束涉及到表时，用于选择最少日志记录模式的规则为：

- 具有参照完整性约束的表上的所有 DML 语句都总是被完全记录，并且覆盖基于数据库级或会话级设置对该表进行的任何日志记录设置。
- 如果表包括参照完整性约束，则不能使用 alter table 将表的日志记录模式更改为 minimal。
- 您不能在已经为其定义最少日志记录模式的表之间创建任何参照完整性约束。例如，当已经在主表上显式设置最少日志记录模式后，如果从一个表向其主表创建外部参照完整性约束，则会引发错误。

最少日志记录模式下的多语句事务

按照下列规则可在后跟多批 DML 语句的显式 begin transaction 中使用最少日志记录模式 DML。

- 您可以在多语句事务中使用对不同表进行操作的 DML 语句，这些表可以采用完全日志记录模式或最少日志记录模式。您可以使用 set dml_logging 命令在事务处理过程中更改日志记录模式，但该操作对后续 DML 语句的影响随已经在相同事务中执行的其它 DML 操作的不同而不同。假定当您启动事务时日志记录模式是 full，则允许执行以下 SQL 事务：

```
begin tran
go
delete t1 where ...
go
set dml_logging minimal
go
insert t1 values ...
go
update t2 where ...
go
commit tran
go
```

  `delete t1` 以完全日志记录模式执行，但 `update t2` 是以最少日志记录模式执行。如果发生回退，则更新操作不会撤消，但删除操作将回退。

- 一旦您在多语句事务中以 full 或 minimal 日志记录模式对表执行了命令，则对同一个表执行的后续命令必须使用相同的日志记录模式（full 或 minimal），而无论会话的 dml_logging 设置如何。在上面的示例中，insert t1 以 full 日志记录模式执行，这是因为先前对 t1 执行的 delete 操作是以 full 日志记录模式执行的。
- 反之，如果先前在相同事务中以最少日志记录模式对该表进行了操作，则将该会话的日志记录模式重置为缺省模式不会恢复 DML 语句的日志记录模式。
- 如果事务被回退，则混合相同事务中不同表的日志记录模式会对涉及到的表产生不同的影响。对具有完全日志记录模式的表进行的更改不会回退，而对具有最少日志记录模式的表进行的更改仍然提交。
- 存储过程内部的日志记录模式选择规则和 set dml_logging 命令的使用与多语句批处理完全相同。如果：
  - 该过程在显式事务外部运行，则每个语句都作为单个事务执行。
  - 该过程在事务内部运行，则前面介绍的相同规则适用。
- 可以更改事务内部的日志记录模式，然后从先前以不同日志记录模式在相同事务中操作的表中执行 select。对此没有任何限制。 delete t1 以日志记录模式执行，而 update t2 以最少日志记录模式执行。当日志记录模式现在为 minimal 时，如果引用相同的表 t1 以供读取 （该表曾经被以完全日志记录模式操作），则不会产生错误。

```
begin tran
go

delete t1 where ...
go

set dml_logging minimal
go

update t2
where t2.c2 = (select c1 from t1 where ...)
go

commit tran
go
```

存储过程和最少日志记录 DML

会话中的 DML 日志记录设置是从所调用的系统过程中继承的，而在过程中执行的任何 set dml_logging 命令的行为设置都由子过程继承。在退出该过程的作用域后，无论 set export_options 是否为 on，都将恢复父会话或父过程中的 dml_logging 设置。

示例

下列示例说明如何应用上述规则来影响过程内的表的日志记录模式。

示例 1　在本示例中，执行过程的用户也是它的所有者以及受此过程影响的所有表的所有者：

```
create procedure p1 as
begin
    delete t1 where...

    set dml_logging minimal

    update t2 where...
end
go

set dml_logging default
go

exec p1
go
/*
** Exiting from the procedure restores the
** session's setting to what it was before
** calling the procedure, in this case, the
** logging mode will be back to DEFAULT
** (i.e. FULL).
*/
-- This will operate in logged mode now.
delete t2
go
```

1　当执行在过程 p1 中开始时，日志记录模式为 full。

2　`delete t1` 以完全日志记录模式执行，然后，该过程的会话级日志记录模式被更改为 minimal。

3　`update t2` 以最少日志记录模式执行。

4　在从 p1 中退出后，当控制返回到外部 SQL 批处理时，日志记录模式设置恢复为 full。下一个 delete t2 以完全日志记录模式执行。

示例 2 本示例在将会话级日志记录模式设置为 minimal 的情况下执行过程 p1。delete t1 以最少日志记录模式操作，update t2 也是如此。在 p1 完成后，下一个 delete t2 也以最少日志记录模式操作，因为日志记录模式已经恢复为调用 p1 之前的模式。

```
set dml_logging minimal
go

exec p1
go

-- This will operate in minimally logged mode.
delete t2
go
```

因为过程内部 DML 语句的日志记录模式受到调用会话或过程的会话级设置的影响，所以 Sybase 建议您在过程体的开头显式选择所需的日志记录模式。当该过程完成后，您可以根据需要将其重新设置为 default。

```
create procedure p1 as
begin
    set dml_logging minimal

    delete t1 where ...
    update t2 where ...

    -- Optionally, reset upon exit
    set dml_logging default
end
go
```

不过，如果过程内部的 DML 语句多数情况下以单一日志记录模式（例如，full）执行，但偶尔必须以不同的（最少日志记录）模式运行，则 Sybase 建议您从调用过程或 isql 会话中用会话级设置来控制日志记录 DML 语句，而不是在过程体内部包括日志记录模式。

示例 3 DML 日志记录设置只影响会话的用户所拥有的那些表。这会影响必须对某些表执行最少日志记录 DML 但由不拥有那些表的用户使用 exec proc 特权执行的过程。

在本示例中，Joe 执行过程 mary.delete_proc，而该过程对 Mary 所拥有的表执行 delete。 Joe 使用 set 命令请求最少日志记录模式，但这样做会在该过程中影响由 Mary 拥有的表的日志记录模式：

```
isql -Sservername -Umary -Pmaryspwd

create procedure mary.delete_proc as
```

```
begin
    delete mary.large_table where ...
end
go

grant exec on mary.delete_proc to joe
go

isql –Sservername –Ujoe –Pjoespwd

-- User 'joe' executes the following SQL:
--
set dml_logging MINIMAL
go

exec mary.delete_proc
go
```

示例 4　Adaptive Server 不允许过程所有者 Mary 在该过程由不拥有该表的用户执行时同意某些语句采用最少日志记录模式。过程中的 set dml_logging 命令仅适用于会话所有者拥有的那些表。

在本示例中，最少日志记录模式不适用于 `delete mary.large_table`，但是，当用户 Joe 用缺省日志记录设置执行该过程时，该模式的确适用于 `update joe.very_large_table`。

```
isql –Sservername –Umary –Pmaryspwd

create procedure mary.delete_proc2 as
begin
    set dml_logging MINIMAL

    delete mary.large_table where ...

    update joe.very_large_table where ...
end
go

grant exec on mary.delete_proc2 to joe
go

isql –Sservername –Ujoe –Pjoespwd

exec mary.delete_proc2
go
```

如果过程对多个所有者拥有的表执行 DML 语句，则根据执行该过程的用户的不同，被执行最少日志记录 DML 的表集也会发生变化。只有表所有者或具有 sa_role 的用户可以执行对特定表执行最少日志记录 DML 的过程。

重新编译正在运行的过程或先前执行的过程的高速缓存计划不会影响在运行期为出现在过程体中的各个 DML 命令选择的日志记录模式。开始执行 DML 语句时，任何可能更改表的日志记录模式的 set 命令都被加以考虑。

在触发器中包括 *set dml_logging*

如果对于表上的 DML 操作有一个活动的触发器，则 DML 语句被完全记录。如果在表上创建触发器并且禁用日志记录功能，则会引发警告，指示 DML 语句将以完全日志记录模式操作。但是，该触发器被成功创建。

触发器中 DML 语句的日志记录模式随引发触发器的用户的不同而不同。由引发触发器的同一个用户拥有的表可以用最少日志记录模式操作。对不是由引发该触发器的用户拥有的表执行的 DML 语句以完全日志记录模式执行，除非这些表被显式设置为以最少日志记录模式操作。

假设 delete_trig_m1 是用户 Mary 所拥有的对象 m1 上的一个 delete 触发器。此触发器以最少日志记录模式对其它表（例如， Joe.j2 和 Paul.p3）执行 DML 语句。对 Mary.m1 的 delete 特权已被授予用户 Sally。

```
Create trigger Mary.delete_trig_m1 FOR DELETE
On Mary.m1
as
Begin
    Delete Mary.min_logged_table_t3
    WHERE ...

    SET DML_LOGGING MINIMAL

    DELETE Paul.p3 WHERE ...
End
```

当用户 Sally 对 Mary.m1 执行 delete 语句时，将引发触发器 Mary.delete_trig_m1，并将代表 Sally 对相应的表执行特权检查。因为 Mary 拥有该触发器，所以 Adaptive Server 不会对 `delete Mary.min_logged_table_t3` 语句（其中，DML 语句以最少日志记录模式运行）执行权限检查。（已经通过某种其它方式将该表定义为进行最少日志记录。）因为 Sally 没有为 Paul.p3 关闭 dml_logging 的特权，所以 set 命令不会自动生效。下一个语句 `delete Paul.p3 where...` 以完全日志记录模式运行。当用户 Mary 执行该触发器时， DML 日志记录行为是相同的 （完全日志记录）。

但是，如果用户 Paul 对 Mary.m1 执行外部 delete，从而导致引发该触发器，则该触发器体中的两个语句都将以最少日志记录模式执行。

一旦该触发器引发，外部 DML 语句就必须已经完全记录；在触发器体中对同一个外部表执行最少日志记录 DML 的尝试将被忽略，这些 DML 语句将被完全记录。

不能在视图上禁用日志记录模式，因此，当带有 instead of 触发器的视图上的 DML 语句执行时，当前仅在视图上支持的 instead of 触发器不会受到视图所引用的基表的日志记录模式的影响，也不会受到日志记录模式的会话级设置的影响。但是，用于在多个表上选择日志记录模式或最少日志记录模式的规则、触发器体内 set 命令的用法以及事务性语义都适用于 instead of 触发器内部的 DML 语句。

使用延迟更新

如果查询优化程序为适合于最少日志记录操作的表选择延迟模式更新，则会为该语句覆盖最少日志记录设置，该语句将在延迟模式下工作，但采用 full 日志记录模式。用大型事务日志记录操作以延迟模式生成 DML 语句的应用程序不会从最少日志记录模式获益。一旦您为表上的多语句事务执行延迟模式操作，此表上的所有后续 DML 语句就会被完全记录。

获取诊断信息

很多规则相互交互以确定表的日志记录模式。应用程序开发人员可能需要了解 Adaptive Server 当前是否正在为特定表生成最少日志记录 DML 命令，并且需要了解模式类型、约束、会话设置等等。

object_attr 根据会话级、表级和数据库级设置来报告表的当前日志记录模式。请参见 《参考手册：构件块》。

使用 set print_minlogged_mode_override 可确定 Adaptive Server 是否已经覆盖您的日志记录模式选择 （请参见 《参考手册：命令》）。print_minlogged_mode_override 向会话输出生成跟踪信息，从而就已经由其它规则为其覆盖表的最少日志记录模式的语句进行报告。这些跟踪信息包括：是否存在参照完整性约束；延迟模式选择；受影响的表的名称；产生影响的规则的说明等。请在整个服务器范围内启用此开关，以捕获整个应用程序的诊断输出。使用 print_output_to_log 开关可将此输出重定向到错误日志。

第 4 章

内存数据库的性能和调优

本章讨论内存数据库和宽松持久性数据库的性能和调优特性，包括高速缓存、 sp_sysmon 结果、监控计数器和 Backup Server。

配置内存存储高速缓存

Adaptive Server 为所有使用命名高速缓存和数据库绑定的数据库类型提供多种数据库和高速缓存配置支持。在调整高速缓存的大小或绑定高速缓存时，请考虑下列因素：

- 将大型磁盘驻留式数据库绑定到较小的命名高速缓存。这是标准的 Adaptive Server 配置，在该配置中，数据库只有一小部分内容在高速缓存中。根据负载的不同，大部分数据库页空间可能由于普通高速缓存替换策略而得到回收利用。
- 使高速缓存大小类似于数据库大小，这可能会降低高速缓存页的回收利用率。但是，根据负载的不同，磁盘写入操作数可能会很高。
- 将整个临时数据库绑定到大得足以承载整个临时数据库的命名高速缓存。服务器应用于临时数据库的缺省优化（以及 delayed commit 之类策略的使用）使您可以显著提高完全高速缓存临时数据库的性能。

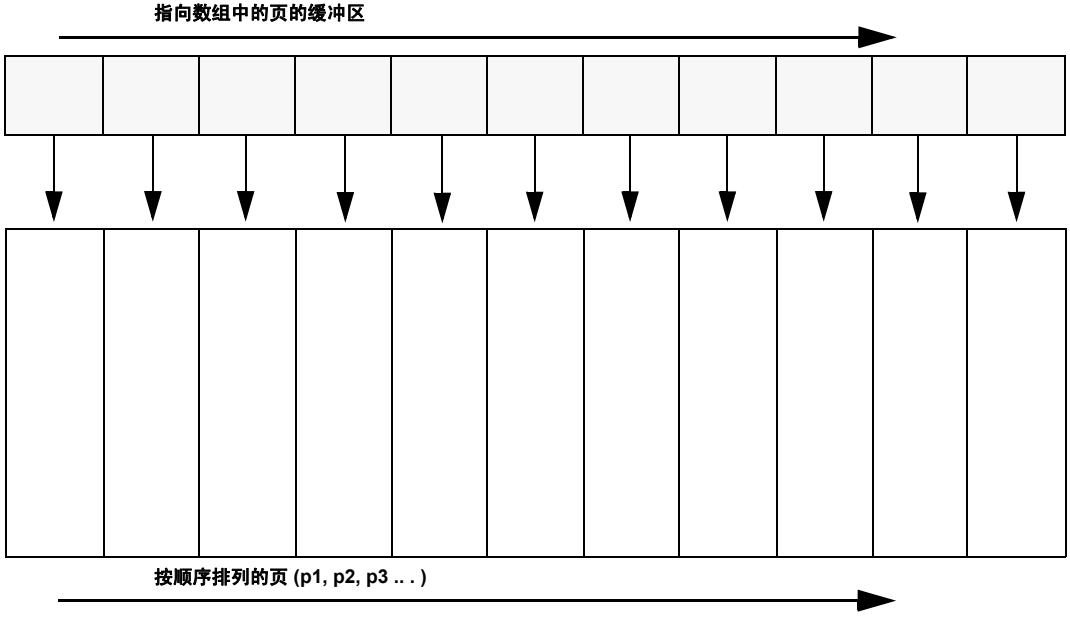
高速缓存布局

宽松持久性数据库以及其中绑定到任何命名高速缓存的对象都使用与常规高速缓存相同的布局。

绑定到高速缓存的对象的访问模式指示常规高速缓存的布局。 Adaptive Server 利用严格的 LRU 高速缓存替换策略来选择最近使用最少的 (LRU) 缓冲区和页对进行替换。对于绑定到命名高速缓存且驻留在磁盘上的对象（由数据库、表和索引 ID 标识）， Adaptive Server 会根据请求将它们的页装载到高速缓存中。在 Adaptive Server 选择这些对象以进行替换之前，它们将一直保留在高速缓存中。 Adaptive Server 根据空闲缓冲区的可用性或根据哪些缓冲区可以替换，来确定将此页读取到高速缓存中的哪个位置，具体取决于 LRU 或最近使用最多的 (MRU) 策略。 Adaptive Server 用于在内存中容纳页的缓冲区可能有所不同，而在此高速缓存配置中，该缓冲区在缓冲区高速缓存中的位置也可能有所不同。

使用内存存储高速缓存的数据库被绑定到高速缓存，此数据库中的所有对象及其索引都使用同一个高速缓存。整个数据库都承载在该高速缓存中。该数据库中的所有页 （包括已分配的和未分配的）都被散列，以便在高速缓存中进行页搜索时总是能够找到所需的页。页按顺序排列（如图 4-1 中所示），而由于所有页都驻留在高速缓存中，因此缓冲区和页对的位置不会改变。

图 4-1：在内存数据库中按顺序排列的页



内存存储高速缓存仅支持缺省池 （对于该页上的任何逻辑读取或写入操作，都使用服务器页大小）。内存存储高速缓存不需要大型缓冲池，这些缓冲池用于与磁盘之间执行大型 I/O 操作。作为一种提高运行期 I/O 性能以及在搜索页时减少高速缓存未命中数的策略，异步预取没有受到内存存储高速缓存的支持。

因为内存存储高速缓存不在磁盘中存储任何数据，所以它具有下列特点：

- 不使用任何替换策略 （对于每个高速缓存，替换策略被定义为 none）。高速缓存中的页从不更改位置。
- 不支持 I/O，因此不支持清洗区和大型池。
- 无需执行缓冲区替换。偶尔会对专用缓冲区使用缺省数据高速缓存。
- 无需动态散列缓冲区。当您创建数据库时，将在散列表中装载和散列所有页。如果您用 alter database 增加数据库的大小，内存数据库可能需要在运行期散列缓冲区。
- 无需执行异步预取，因为所有页都驻留在高速缓存内存中。

内存存储高速缓存支持高速缓存分区。高速缓存分区机制通过将内存数据库中的页子集分配到分区 （其中，每个子集都由一个单独的螺旋锁控制）来减少对单个螺旋锁的争用。

内存数据库的 sp_sysmon 输出

对内存数据库运行 sp_sysmon 可监控它们的性能。

对于内存数据库高速缓存而言， Cache Hits 的值应该总是为 100%，而 Cache Misses 的值应该总是为 0：

```
Cache: imdb
                              per sec      per xact       count   % of total
------------------------   ------------  ------------  ----------  ----------
Spinlock Contention                 n/a           n/a         n/a      50.9 %

Utilization                         n/a           n/a         n/a      99.9 %

Cache Searches
Cache Hits                      16954.4        5735.8     1152897     100.0 %
Found in Wash                       0.0           0.0           0       0.0 %
Cache Misses                        0.0           0.0           0       0.0 %
------------------------   ------------  ------------  ----------  ----------
Total Cache Searches             16954.4         5735.8     1152897
```

内存高速缓存不应当具有清洗区，如下面所报告的那样：

```
Buffer Wash Behavior
   Statistics Not Available - No Buffers Entered Wash Section Yet

Cache Strategy
Cached (LRU) Buffers          21225.3         7180.7     1443321      100.0 %
Discarded (MRU) Buffers           0.0            0.0           0        0.0 %
```

请参见 *Performance and Monitoring Series: Monitoring Adaptive Server with sp_sysmon*（《性能和监控系列：使用 sp_sysmon 监控 Adaptive Server》）。

下列 sp_sysmon 部分不适用于内存存储高速缓存，原因是整个数据库都被绑定到高速缓存，并且所有的数据库页都驻留在高速缓存中：

- Utilization
- Cache Hits
- Found in Wash
- Cache Misses
- Large I/O Usage
- Pool Turnover

下列各部分不适用于内存数据库，原因是它们不执行缓冲区争夺：

- LRU Buffer Grab
- Grabbed Dirty
- Total Cache Turnover

下列各部分不适用于内存数据库，原因是它们未定义清洗区。在稳定状态下，不会发生磁盘读取和写入操作：

- Buffer Wash Behavior
- Cache Strategy

下列各部分不适用于内存数据库，原因是它们不使用大型池：

- Large I/Os Performed
- Large I/Os Denied
- Total Large I/O Requests
- Large I/O Detail

监控缺省数据高速缓存性能

因为所有页都在内存存储高速缓存中进行散列，所以，需要使用少量专用缓冲区来存储中间页（例如，在排序过程中）的任务使用缺省数据高速缓存。使用 buf_imdb_privatebuffer_grab 监控计数器可确定 Adaptive Server 是否暂时使用缺省数据高速缓存对缓冲区进行排序：

```
buf_imdb_privatebuffer_grab  buffer_0          244              510525
```

此脚本从所有监控组收集监控计数器。它首先清除所有监控计数器，对监控计数器进行一分钟的取样，然后报告该监控时间间隔内所有监控计数器更新的值。

若要确定 buf_imdb_privatebuffer_grab 的当前值，请运行：

```
dbcc monitor ('clear','all','on')
go
dbcc monitor ('sample','all','on')
go
waitfor delay "00:01:00"
go
dbcc monitor ('sample','all','off')
go
dbcc monitor ('select','all','on')
go
select * from master.dbo.sysmonitors
where field_name = 'buf_imdb_privatebuffer_grab'
go
```

buf_imdb_privatebuffer_grab 的值指示下列查询中的缺省数据高速缓存的缓冲区使用情况：这些查询需要使用临时缓冲区来完成针对内存数据库中的表运行的中间排序。可根据缺省数据高速缓存的大小来评估 buf_imdb_privatebuffer_grab 的值：

- 如果与缺省数据高速缓存的大小相比争夺的缓冲区数量很小，则表明针对内存数据库中的表运行的查询未大量使用临时缓冲区。
- 如果争夺的缓冲区数量占缺省数据高速缓存中缓冲区数量的很大一部分，则表明缺省数据高速缓存需要为提供针对内存数据库运行的查询所使用的缓冲区而承担较重的负载。通常，只有当缺省数据高速缓存非常小（例如，它使用的缺省大小为 8MB）时，才会发生这种情况。

  使用较小的缺省数据高速缓存可能会影响其它依赖于缺省数据高速缓存的应用程序的性能。增加缺省数据高速缓存的大小既可满足内存数据库中对临时缓冲区使用率的请求，又可满足使用同一个高速缓存的其它并发应用程序的需要。

请参见 *Performance and Tuning Series: Monitoring Adaptive Server with sp_sysmon*（《性能和调优系列：使用 sp_sysmon 监控 Adaptive Server》）中的第 2 章“Monitoring Performance with sp_sysmon”（使用 sp_sysmon 监控性能）。

为内存设备组织物理数据

因为内存数据库不使用 I/O，所以，在组织物理数据放置时，您无需考虑设备 I/O 特性（例如， I/O 设备的速度）。您无需考虑将频繁访问的索引页放到快速设备上或者将其它很少利用的对象的页（例如，文本和图像页）放在较慢设备上等问题。但是，由于内存数据库消除了磁盘 I/O 所导致的其它瓶颈，因此，螺旋锁和闩锁争用等其它瓶颈可能很严重。

当您为内存设备组织物理数据时，请考虑下列策略：

- 若要减少闩锁争用，请使用单独的数据和日志设备（也就是说，不要为混合日志和数据配置内存数据库）。
- 若要控制日志空间消耗量，请使用单独的日志设备。
- 若要限制对象空间使用量，请按段组织设备。
- 若要减少页分配争用，请使用绑定到不同设备的段。

低持久性数据库的性能优化

低持久性数据库将其持久性设置为 no_recovery 和 at_shutdown。因为不恢复低持久性数据库，所以它们提供了下列性能优点：

- 低持久性数据库不会由于解锁而刷新用户日志高速缓存 (ULC)，因此增加了事务密集型系统中的事务吞吐量。

  当两个事务试图更新同一个数据行锁定数据页，从而导致第二个事务在使用该数据页之前首先将第一个事务的 ULC 中的日志记录刷新到 syslogs 时，将发生 ULC 解锁。增加到 syslogs 的 ULC 刷新会增加日志争用，从而对事务密集型系统中的事务吞吐量造成不利的影响。

sp_sysmon 输出中的 Transaction Management 部分表明，低持久性数据库的 ULC Flushes to Xact Log、 by Unpin 和 by Single Log Record（所有与解锁相关的操作）的值明显低于完全持久性数据库的相应值。这就增加了事务密集型系统中的事务吞吐量。低持久性数据库的以下 sp_sysmon 输出表明没有由于解锁而产生的 ULC 刷新：

```
ULC Flushes to Xact Log    per sec       per xact    count  % of total
-----------------------  ----------   ------------  -------  ---------
Any Logging Mode DMLs
by End Transaction             40.3            0.0     2416   100.0 %
by Change of Database           0.0            0.0        1     0.0 %
by Unpin                        0.0            0.0        0     0.0 %
by Other                         0.0            0.0        0     0.0 %
```

- 低持久性数据库不在 commit 过程中执行事务日志刷新，原因是事务不写入磁盘。

  对于宽松持久性数据库而言，通常不需要进行事务日志刷新；这些数据库不需要可靠记录的已提交事务，原因是它们不在重新启动过程中恢复。 Adaptive Server 通常在事务完成后刷新事务日志；对于宽松持久性数据库而言，它在缓冲区清洗过程中刷新事务日志。

  sp_sysmon 输出中的 Device Activity Detail 部分显示了宽松持久性数据库日志设备的写入操作次数。它显示了 15.0.3 版磁盘驻留式数据库的 sp_sysmon 输出：

```
Device:
/disk_resident/device/1503esd2/drdb_device.dat
drdb_device                per sec      per xact       count  % of total
-------------------   -----------  ------------  ---------  ----------
Reads
  APF                         0.0           0.0          0      0.0 %
  Non-APF                    14.3           0.0       2380      0.2 %
Writes                     9247.3           1.5    1544304     99.8 %
-------------------   -----------  ------------  ---------   ----------
Total I/Os                 9261.6           1.5    1546684     100.0 %
```

下面显示了 15.5 版 Adaptive Server 的有关宽松持久性数据库的相同输出。因为这是宽松持久性数据库，所以其写入操作数大约占磁盘驻留式数据库写入操作数的 23%：

```
Device:
/relaxed_durability/IMDB/devices/ariesSMP/rddb_device.dat
rddb_device                per sec      per xact      count    % of total
-------------------   -----------  ------------  ---------   ----------
Reads
  APF                         0.0           0.0          0       0.0 %
  Non-APF                    14.7           0.0       1031       0.7 %
Writes                     2106.4           0.1     147446      99.3 %
-------------------   -----------  ------------  ---------   ----------
Total I/Os                 2121.1           0.1     148477      99.9 %
```

- 低持久性数据库不对 ULC 中包含的事务进行日志记录。如果每个事务的所有日志记录都完全包含在 ULC 中，并且每个事务都不需要使用日志记录来进行任何提交后处理，则 Adaptive Server 在提交事务时会放弃低持久性数据库中事务的日志记录。

  因为您无需传输日志记录，所以日志的争用数量减少，从而增加了事务密集型系统中的吞吐量。通常，在 ULC 中记录事务更有利于较小的事务，原因是大型事务在提交时无法在 ULC 中包含其所有日志记录；所记录的记录数大于 ULC 容量的大型事务必须在 ULC 已满时将其内容刷新到 syslogs 中。

  sp_sysmon 的 Transaction Management 部分中的 ULC Flushes to Xact Log 和 by Full ULC 的值表明 Adaptive Server 是否由于 ULC 已满而将其刷新。

  如果这些值很高，则 ULC 无法在提交事务时完全包含该事务，并且由于无法放弃日志记录，因此 Adaptive Server 的性能没有因为不在 ULC 中记录事务日志而获益。请增加 user log cache size 配置参数的大小，然后重新运行 sp_sysmon 以查看 ULC Flushes to Xact Log 的值是否已减小。如果该值已减小，则表明 ULC 不再需要刷新到 syslogs。

  如果小型事务在提交时包含在 ULC 中，则也可能会发生 ULC 刷新，原因是 Adaptive Server 需要使用这些日志记录来完成提交后工作。例如，如果事务在需要完成提交后工作时将数据库中的任何空间解除分配，则也会在事务中发生 ULC 刷新。

- 低持久性数据库不会将记录不全的更改刷新到磁盘。 Adaptive Server 必须偶尔将完全持久性数据库的数据页刷新到磁盘，原因是相应的日志记录不能完全描述所发生的更改。以下是一些示例：
  - 带有聚簇索引的表的任何索引或数据页拆分
  - 排序
  - writetext 命令
  - 快速 bcp
  - alter table 以更改表的锁定方案
  - alter table 以更改表的分区方案或表的模式
  - 全表 reorg rebuild
  - alter table...unparition

  内存数据库无需将数据页刷新到磁盘，原因是它们不使用磁盘。宽松持久性数据库不会将数据页刷新到磁盘，原因是无需进行数据库恢复。当磁盘 I/O 操作数很高 （例如，在 sort 操作结束时将已更改的页刷新到磁盘）时，无需将数据页刷新到磁盘可以显著提高性能。

对低持久性数据库进行的不与其持久性直接相关、在内部发生且无需管理的其它改进：

- 有关更新数据库时间戳 （事务密集型数据库中的频繁操作）的改进
- 对通过对仅数据锁定表使用索引扫描进行删除的改进
- 对向带非唯一索引的仅数据锁定表中进行批量插入的改进

上述改进还适用于将持久性显式设置为 no_recovery （通过 create database 或 alter database）的临时数据库；它们不适用于将持久性显式设置为 no_recovery 的临时数据库。

调整检查点时间间隔

除了确定恢复时间的长短以外， recovery interval in minutes 配置参数还确定 Adaptive Server 对数据库执行检查点的频率。从来不对内存数据库执行检查点，但是 Adaptive Server 确实对宽松持久性数据库执行检查点，并按照 recovery interval in minutes 从磁盘中刷新所有已修改的缓冲区。

使用 recovery interval in minutes 可减少对缓冲区清洗施加的压力，并维护可替换的缓冲区。

recovery interval in minutes 的值越低， Adaptive Server 执行检查点和冲洗所有已更改缓冲区的频率就越高。但是，您必须在执行缓冲区清洗获得的好处与 Adaptive Server 执行检查点时发生的磁盘 I/O 操作数之间取得平衡。

当您将 recovery interval in minutes 设置为较高的值时，您必须在由于执行检查点的频率较低而使磁盘 I/O 操作数减少所带来的好处与缓冲区请求在缓冲区清洗发生时发现缓冲区 “脏”从而延迟其使用的可能性之间取得平衡。

recovery interval in minutes 适用于服务器中的所有数据库，只有在评估此参数的更改会对完全持久性数据库产生的影响之后，才能更改此参数。如果您至少有一个在服务器崩溃后通常需要进行重大恢复的完全持久性数据库，请继续使用及时恢复此数据库所需的值 （通常为缺省值，即 5 分钟）。如果更改对完全持久性数据库几乎没有影响 （也就是说，这些数据库在服务器发生故障后几乎不需要恢复），请首先使用一个比缺省值 5 分钟更长的恢复时间间隔 （例如， 30 分钟）。在进行更改之后，请查看 sp_sysmon 的 “数据高速缓存管理”部分中的 Buffers Grabbed Dirty 值或每个高速缓存的信息。如果 Buffer Grabbed Dirty 值较高，请减小 recovery interval in minutes 的值。

若要减少检查点为宽松持久性数据库执行的磁盘 I/O 操作的数量，同时不影响完全持久性数据库的行为 （无论恢复时间间隔的值是多少），请为宽松持久性数据库将 no chkpt on recovery 数据库选项设置为 true。使用上述方法可评估 Buffers Grabbed Dirty 的值，并检验禁用检查点是否不会对可重用缓冲区的可用性产生不利影响。

最少日志记录 DML

在执行最少日志记录 DML 的过程中， insert、 update、 delete 和慢速 bcp-in 命令在执行时只进行最少量的日志记录或不做日志记录。如果语句失败 （例如，由于发生内部错误或用户发出回退指令），则已经完成的那部分工作仍将提交，而不会回退。但是，在针对最少日志记录 DML 进行配置后， Adaptive Server 必须保持已更改的表的逻辑一致性 （例如，回退数据行插入操作必须导致相关索引行的回退）。为了确保此一致性，将最少日志记录 DML 命令划分为多个名为子命令的基本单元操作，它们全都是执行单个行更改所需的数据更改，包括对索引以及任何文本、图像或行外列的更改。为了在子命令级保持逻辑一致性， Adaptive Server 在用户日志高速缓存 (ULC) 中记录了最少日志记录 DML 命令。在子命令完成后，Adaptive Server 立刻从 ULC 中放弃该子命令的日志记录，因而无需将 ULC 刷新到 syslogs。

如果 ULC 不够大，无法包含所有子命令的日志记录，则 Adaptive Server 可能无法放弃日志记录。如果受到 DML 影响的数据行非常大，或者表包含很多索引，则通常会发生这种情况。如果 Adaptive Server 无法放弃 ULC，则会将 ULC 日志记录刷新到 syslogs，而这会产生下列后果：

- 增加忙碌系统中的日志争用
- 减小事务吞吐量
- 增加所需的日志空间量，从而抵消了使用最少日志记录 DML 所获得的所有好处

请确保 ULC 足够大，以包含大多数子命令的日志记录。 Sybase 建议您的 ULC 的大小为缺省服务器页大小的两倍：

```
declare @ulc_size int
select @ulc_size = @@maxpagesize * 2
exec sp_configure "user log cache size", @ulc_size
```

对于已经显式设置持久性的内存临时数据库或宽松持久性临时数据库，可通过创建大小为服务器逻辑页大小两倍的、特定于会话 tempdb 的 ULC，来提高最少日志记录功能的效率并避免向 syslogs 进行 ULC 刷新：

```
declare @ulc_size int
select @ulc_size = @@maxpagesize * 2
exec sp_configure "session tempdb log cache size", @ulc_size
```

通常，上述脚本可对 ULC 的大小进行相应的配置，以使其能够将一条子命令进行的所有更改全部包含在内存中，并在并发 DML 性能方面产生重大改进。

更改 ULC 大小需要您重新启动 Adaptive Server。

若要确定最少日志记录功能是否有效，请查看 sp_sysmon 输出的 Transaction Management 部分。

以下示例显示了有效的最少日志记录 DML，原因是 Adaptive Server 放弃了从子命令生成的大多数日志记录：

```
ML-DMLs ULC Efficiency    per sec      per xact       count  % of total
------------------------  -------  ------------  ----------  ----------
Discarded Sub-Commands    33383.9       11087.8     3071323       100.0
Logged Sub-Commands           0.4           0.1          37         0.0
```

Transaction Detail 部分概括了对于指定示例，以完全日志记录模式和最少日志记录模式执行 DML 命令的对比情况。在此输出中，Adaptive Server 以最少日志记录模式对一个 data-only locked 表执行了几乎所有插入操作：

```
Transaction Detail             per sec      per xact     count  % of total
-------------------------   ------------  ------------  -------   ----------
Inserts
  Fully Logged
   APL Heap Table                 57.8         173.5      694        0.7 %
   APL Clustered Table             0.0           0.0        0        0.0 %
   Data Only Lock Table            0.7           2.0        8        0.0 %
   Fast Bulk Insert                0.0           0.0        0        0.0 %
  Minimally Logged
   APL Heap Table                  0.0           0.0        0        0.0 %
   APL Clustered Table             0.0           0.0        0        0.0 %
   Data Only Lock Table         7775.8       23327.5    93310       99.3 %
```

Transaction Management 部分提供了有关 Adaptive Server 如何记录子命令而不是放弃它们的详细信息。下面的输出说明了导致向事务日志进行 ULC 刷新的事件，同时也说明了由完全日志记录 DML 和最少日志记录 DML 导致的 ULC 刷新中止。对于 Minimally Logged DML 部分，由 Full ULC 和子命令结束导致的刷新操作数几乎相等：

```
ULC Flushes to Xact Log         per sec      per xact     count  % of total
-------------------------   ------------  ------------  -------   ----------
Any Logging Mode DMLs
by End Transaction                  0.3           1.0        4       11.1 %
by Change of Database               0.0           0.0        0        0.0 %
by Unpin                            0.0           0.0        0        0.0 %
by Other                            0.0           0.0        0        0.0 %

Fully Logged DMLs
  by Full ULC                       0.2           0.5        2         5.6 %
  by Single Log Record              0.0           0.0        0         0.0 %

Minimally Logged DMLs
  by Full ULC                       1.3           4.0       16        44.4 %
```

```
  by Single Log Record                0.0            0.0         0          0.0 %
  by Start of Sub-Command             0.0            0.0         0          0.0 %
  by End of Sub-Command               1.2            3.5        14         38.9 %
------------------------     ------------   ------------   -------    ----------
 Total ULC Flushes                    3.0            9.0        36
```

如果 ULC Flushes to Xact Log 中 Minimally Logged DMLs 下 by Full ULC 部分中的 `count` 列的值比受影响的行数大，请增大 user log cache 或 session tempdb log cache size 配置参数的值。

下面的输出表明了最少日志记录命令的 ULC 操作的效率：系统只引起由最少日志记录功能产生的少量日志记录开销，原因是几乎所有日志记录活动都完全包含在 ULC 中，并且几乎没有向 syslogs 进行刷新。

```
ML-DMLs ULC Efficiency          per sec       per xact      count  % of total
-------------------------    ------------  ------------  -------   ----------
Discarded Sub-Commands           7774.7        23324.0      93296        100.0
Logged Sub-Commands                 1.2            3.5         14          0.0
-------------------------    ------------  ------------  -------
Total ML-DML Sub-Commands        7775.8        23327.5      93310
```

转储和装载内存数据库

转储和装载宽松持久性数据库与转储和装载完全持久性数据库相同。在转储宽松持久性数据库的过程中，Backup Server 直接从基于磁盘的数据库设备中读取；在装载过程中，Backup Server 通过从转储存档复制页直接向这些设备进行写入操作。通过对宽松持久性数据库使用分条设备，可以提高转储和装载性能。

因为内存数据库不使用磁盘设备，所以 Backup Server 在转储过程中直接从 Adaptive Server 共享内存读取页，并将其写入存档介质。在装载过程中， Backup Server 从存档介质 （例如，磁带）读取页，然后将负载直接写入内存存储高速缓存的页。因为内存数据库不使用磁盘 I/O 读取或写入服务器页，所以内存数据库的转储和装载性能通常优于具有相同规格的基于磁盘的数据库。

在执行 dump 和 load 命令的过程中， Backup Server 为每个设备打开一个到 Adaptive Server 的 CT-library 连接，并创建一条通信通道以直接从 Adaptive Server 共享内存中读取数据库页。

Backup Server 在直接从其共享内存中读取页时，将直接与 Adaptive Server 中的活动并发任务进行同步。为了支持 load database 恢复，Adaptive Server 在 dump 操作过程中禁用了第 42 页的 “低持久性数据库的性能优化”中介绍的策略。这可能会在 dump database 命令执行过程中降低事务性活动的性能。

螺旋锁争用和网络连接调优

内存数据库不像磁盘驻留式数据库那样有那么多的延迟和争用问题，原因是它们不使用磁盘 I/O。但是，在负载非常高的情况下，内存数据库可能会受到其它螺旋锁争用和内存访问问题的困扰：

- 高速缓存螺旋锁争用 — 在负载非常高的情况下，可能会成为内存高速缓存的瓶颈。请考虑将高速缓存分区数增加到 64 个或更多。大量高速缓存分区所需的额外内存资源微不足道，但通过减少高速缓存管理器螺旋锁争用提高了性能。
- 对象管理器螺旋锁争用 — 如果您的应用程序频繁访问少量对象，您可能会注意到对元数据结构的螺旋锁争用，这种情况由 sp_sysmon 报告。

  使用 dbcc tune 'des_bind' 可绑定频繁访问对象的描述符，以使其永远不会被清除。即使仅绑定几个常用对象的描述符，也可以大大减少总体元数据螺旋锁争用情况并提高性能。

改进对锁管理程序散列表螺旋锁比率的争用

内存数据库的吞吐量可能会产生对锁管理程序散列表螺旋锁比率的争用。表锁散列表以及页和行锁散列表螺旋锁可能会成为产生争用的重要因素。

sp_sysmon 的 Lock Management 部分显示了对于为这些散列表管理散列桶的螺旋锁的争用百分比：

```
Lock Management
---------------

Lock Summary                  per sec     per xact     count     % of total
------------------------  ------------  ------------  --------  ------------
Total Lock Request           285063.3          43.0  17103795           n/a
Avg Lock Contention            1857.3           0.3    111435         0.7 %
Cluster Locks Retained            0.0           0.0         0         0.0 %
Deadlock Percentage               0.1           0.0         8         0.0 %

Lock Detail                   per sec     per xact     count     % of total
------------------------  ------------  ------------  --------  ------------
Table Lock Hashtable
  Lookups                  130160.4          19.6   7809622           n/a
  Avg Chain Length              n/a           n/a   0.00000           n/a
  Spinlock Contention           n/a           n/a       n/a          4.6 %
[...]
Page & Row Lock HashTable
```

```
  Lookups                      268968.1          40.6  16138085          n/a
  Avg Chain Length                n/a           n/a   1.03330          n/a
  Spinlock Contention             n/a           n/a       n/a          4.8 %
```

通常，如果此争用超过大约 4%，请考虑减小这些螺旋锁与散列桶之间的比率。对一个螺旋锁所控制的散列桶个数进行控制的配置选项的缺省值是：

- 如果 lock table spinlock ratio 的缺省值为 20，则会影响 Table Lock Hashtable 输出中的螺旋锁争用。
- 如果 lock spinlock ratio 的缺省值为 85，则会影响 Page & Row Lock Hashtable 输出中的螺旋锁争用。

当螺旋锁争用很严重时，减小 lock table spinlock ratio 和 lock spinlock ratio 的值可以提高运行期性能。用较多螺旋锁来控制较少的散列桶所产生的额外内存开销并不大。开始时，可以将配置参数的值减小一半。如果发生极端螺旋锁争用情况 （超过 10%），则将相应的配置选项减小到非常小的值 （例如， 3 – 5）可能有助于消除螺旋锁开销产生的性能瓶颈。

确定网络连接数

缺省情况下， Adaptive Server 使用单个网络监听器，该监听器可以在任何引擎上运行。当 Adaptive Server 收到连接请求时，接受该连接的引擎将成为该连接的网络引擎，而当相应的任务执行网络 I/O 时，必须迁移到此引擎。

如果同时存在很多个客户端连接， Adaptive Server 可能无法在不同的连接请求之间调度监听器 （除非连接请求由于时间片耗完而放弃）并在同一个引擎上接受所有连接。由于任何相应任务都必须在该引擎上运行以执行网络 I/O，因此该引擎成为瓶颈。使用 sp_sysmon Network I/O Management 部分可确定 Adaptive Server 如何分配网络 I/O。以下示例说明 Adaptive Server 未按比例分配网络 I/O：Engine 2 使用了 88% 以上的网络 I/O，而其它引擎使用了很少的网络 I/O，甚至根本没有使用：

```
Network I/O Management
----------------------

Total Network I/O Requests       7301.5           1.4      438092
n/a
Network I/Os Delayed              0.0             0.0           0
0.0 %

Total TDS Packets Received      per sec     per xact     count      % of total
--------------------------  ------------  ------------  --------    ------------
```

```
Engine 0                          308.8           0.1    18528          7.7 %
Engine 1                          163.6           0.0     9818          4.1 %
Engine 2                         3558.8           0.7   213527         88.3 %
Engine 3                            0.0           0.0        2          0.0 %
Engine 4                            0.0           0.0        0          0.0 %
Engine 5                            0.0           0.0        0          0.0 %
-------------------------  ------------      --------  -------

Total TDS Packets Rec'd          4031.3           0.8   241875

Avg Bytes Rec'd per Packet          n/            n/a      136
n/a
-------------------------------------------------------------------------------



Total TDS Packets Sent           per sec     per xact     count     % of total
-------------------------  ------------  ------------  --------    ------------
Engine 0                          308.8           0.1     18529          7.7 %
Engine 1                          163.6           0.0      9818          4.1 %
Engine 2                         3558.9           0.7    213531         88.3 %
Engine 3                            0.0           0.0         2          0.0 %
Engine 4                            0.0           0.0         0          0.0 %
Engine 5                            0.0           0.0         0          0.0 %
-------------------------  -----------  ------------  --------

Total TDS Packets Sent           4031.3            0.8    241880
```

若要解决不平衡的网络 I/O 使用率，请使用多个网络监听器并将它们绑定到不同的引擎（通常，一个监听器对应一个引擎）。若要确定将多少个客户端绑定到每个网络监听器，请划分客户端连接以便每个监听器接受大约相同数量的连接。例如，如果有 6 个网络监听器和 60 个客户端，可将每组 10 个客户端连接到一个监听器。

在平衡上述网络监听器后，sp_sysmon 输出可能如下所示：

```
Network I/O Management
----------------------

Total Network I/O Requests       8666.5         1.3     519991            n/a
n/a
Network I/Os Delayed                0.0         0.0          0          0.0 %

Total TDS Packets Received       per sec     per xact     count      % of total
-------------------------  ------------  ------------  --------    ------------
Engine 0                          893.4           0.1     53602         17.8 %
Engine 1                          924.5           0.1     55468         18.5 %
Engine 2                          701.9           0.1     42113         14.0 %
Engine 3                          906.0           0.1     54358         18.1 %
```

```
Engine 4                          896.1           0.1     53763         17.9 %
Engine 5                          683.8           0.1     41028         13.7 %
------------------------  ------------      --------  -------
Total TDS Packets Rec'd          5005.5           0.8   300332

Avg Bytes Rec'd per Packet          n/            n/a      136
n/a
-------------------------------------------------------------------------------

Total TDS Packets Sent           per sec     per xact     count     % of total
------------------------  ------------  ------------  --------    ------------
Engine 0                          893.3           0.1     53595         17.8 %
Engine 1                          924.5           0.1     55467         18.5 %
Engine 2                          701.9           0.1     42113         14.0 %
Engine 3                          905.9           0.1     54355         18.1 %
Engine 4                          896.1           0.1     53763         17.9 %
Engine 5                          683.8           0.1     41026         13.7 %
------------------------   -----------  -----------  --------
Total TDS Packets Sent             4031.3         0.8    241880
```

网络监听器不平衡并非内存数据库所特有的问题，在磁盘驻留式数据库中也可能发生。但是，由于内存驻留式数据库不使用磁盘 I/O，因此它们通常具有比磁盘驻留式数据库更高的吞吐量。吞吐量增加以及不存在磁盘 I/O 延迟会使内存数据库完成比磁盘驻留式数据库更多的工作，包括执行更多网络 I/O，而这可能会由于网络监听器负载不平衡而增加瓶颈的严重性。

索引

符号

英文

B

C

D

F

G

H

J

K

L

M

N

P

Q

R

S

W

X

Y

Z