IBM® DB2® 通用数据库

管理指南: 性能

版本 7

SB84-0243-00

IBM® DB2® 通用数据库

管理指南: 性能

版本 7

SB84-0243-00

在使用本资料和它支持的产品之前，请参阅第549页的『附录G. 注意事项』中的一般信息。

目录

关于本书

本管理指南在它的三卷中提供了必要的参考信息，以便使用和管理 2000 年就绪的 DB2* 关系数据库管理系统 (RDBMS) 产品，包括:

- 有关数据库设计的信息（可在*管理指南: 计划*中找到）
- 有关实现和管理数据库的信息（可在*管理指南: 实现*中找到）
- 有关配置和调整数据库环境以改善性能的信息（可在*管理指南: 性能*中找到）

本书中描述的许多任务可以使用不同的接口来执行:

- **命令处理器**，它允许您从图形界面存取和操纵数据库。从此接口，您也可执行 SQL 语句和 DB2 实用程序功能。本书中的大多数示例是说明此接口的使用。有关使用命令处理器的详情，参见 *Command Reference*。
- **应用程序设计接口**，它允许您在应用程序内执行 DB2 实用程序功能。有关使用应用程序设计接口的详情，参见 *Administrative API Reference*。
- **控制中心**，它允许您以图形方式执行管理任务，如配置系统、管理目录、备份和恢复系统、调度作业以及管理媒体。控制中心也包含复制管理，以便以图形方式设置系统之间数据的复制。此外，“控制中心”使您能够通过图形用户界面执行 DB2 实用程序功能。要调用控制中心，使用 `db2cc` 命令，或（对于 OS/2）从 DB2 文件夹选择控制中心图符。要获得介绍性帮助，从控制中心窗口的**帮助**下拉菜单中选择**入门**。 **Visual Explain** 和**性能监控程序**工具是从控制中心中调用的。

还有其他工具可用来执行管理任务。它们包括:

- 脚本中心，用于存储称为脚本的小应用程序。这些脚本可包含 DB2 命令以及操作系统命令。
- 警报中心，用于监控其他 DB2 操作产生的信息。
- 工具设置，用于更改“控制中心”、“警报中心”和“复制”的设置。
- 日志，用于调度将以无人照管方式运行的作业。
- “数据仓库中心”，用于管理仓库对象。

谁应使用本书

本书主要面向数据库管理员、系统管理员、安全管理员和系统操作员，他们需要设计、实施和维护本地或远程客户机要存取的数据库。需要了解 DB2 关系数据库管理系统的管理和操作的程序员和其他用户也可使用本书。

本书的结构

本书包含关于以下主要主题的信息:

性能介绍

- 第1部分 性能介绍介绍管理和改进 DB2 UDB 性能的概念和考虑事项。
- 第2章 体系结构和进程概述介绍下层 “DB2 通用数据库” 体系结构和处理。

调整应用程序性能

- 第3章 应用程序考虑事项描述在设计应用程序时改善数据库性能的一些技术。
- 第4章 环境考虑事项描述在设置数据库环境时改善数据库性能的一些技术。
- 第5章 系统目录统计信息描述可以如何收集并使用有关您的数据的统计信息以确保最优性能。
- 第6章 了解 SQL 编译程序描述当使用 SQL 编译程序编译 SQL 语句时所产生的结果。
- 第7章 SQL 解释设施描述解释设施，它允许您检查 SQL 编译程序为存取您的数据所做的选择。

调整和配置系统

- 第8章 操作性能提供有关数据库管理程序如何使用内存及影响运行期性能的其他考虑事项的概述。
- 第9章 使用控制器提供有关如何使用控制器来控制数据库管理的某些方面的介绍。
- 第10章 通过添加处理器调整配置介绍与增加数据库系统的大小相关的一些考虑事项和任务。
- 第11章 将数据重新分布至数据库分区讨论在一个分区数据库环境中将数据重新分布在各分区中所需的任务。
- 第12章 基准测试提供有关基准测试和如何执行基准测试的概述。
- 第13章 配置 DB2讨论数据库管理程序、数据库配置文件和配置参数的值。

附录

- 附录A. DB2 注册表变量和环境变量列出简要表注册表值和环境变量。
- 附录B. 解释表和定义提供有关 DB2 解释设施使用的表和如何创建那些表的信息。
- 附录C. SQL 解释工具提供有关使用 DB2 解释工具: db2expln 和 dynexpln 的信息。
- 附录D. db2exfmt - 解释表格式的工具格式化 DB2 解释表的内容。

- 附录E. 配置 XA 事务管理程序以使用 DB2 UDB描述如何配置特定的产品，以将 DB2 用作资源管理程序。
- 附录F. 使用 DB2 资料库提供关于 DB2 资料库结构的信息，包括向导、联机帮助、信息和书籍。

“管理指南”的其他卷的简要概述

管理指南: 计划

*管理指南: 计划*关心的是数据库设计。它讨论了逻辑和物理设计问题、分布式事务问题以及高可用性主题。下面简要描述该卷中的某些章节和附录:

DB2 通用数据库的世界

- “管理 DB2 通用数据库”提供“DB2 通用数据库”的介绍和概述。

数据库概念

- “基本关系数据库概念”提供数据库对象的概述，包括恢复对象、存储器对象和系统对象。
- “联合体系统”讨论联合体系统，联合体系统是一个数据库管理系统 (DBMS)，它支持应用程序和用户提交特定 SQL 语句，这些 SQL 语句在单条语句中引用两个或多个 DBMS 或数据库。
- “并行数据库系统”提供 DB2 可使用的并行度类型的介绍。
- “关于数据入库”提供数据入库和数据入库任务的概述。
- “关于 Spatial Extender”通过说明 Spatial Extender 的用途和讨论其处理的数据介绍 Spatial Extender。

数据库设计

- “逻辑数据库设计”讨论逻辑数据库设计的概念和准则。
- “物理数据库设计”讨论物理数据库设计的准则，包括与数据存储相关联的考虑事项。

分布式事务处理

- “设计分布式数据库”讨论如何在单个事务中存取多个数据库。
- “针对事务管理程序的设计”讨论如何在分布式事务处理环境（如 CICS）中使用数据库。

高可用性系统

- “针对高可用性的设计”提供 DB2 所提供的高可用性故障恢复支持的概述。

- “AIX 上的高可用性群集多重处理增强可缩放性 (HACMP ES)”讨论 DB2 对 AIX 上的高可用性故障恢复的支持。
- “Windows NT 环境中的高可用性”讨论 DB2 对 Windows NT 上的高可用性故障恢复的支持。
- “Sun Cluster 2.2 上的 DB2 和高可用性”讨论对 Sun Solaris 操作系统上的高可用性故障恢复的 DB2 支持。

附录

- “计划数据库迁移”提供有关将数据库迁移至版本 7 的信息。
- “发行版间的不兼容性”讨论不同发行版（最多到版本 7）之间的不兼容性。
- “国家语言支持 (NLS)”介绍 DB2 的“国家语言支持”，包括有关国家、语言和代码页的信息。

管理指南: 实现

*管理指南: 实现*关心的是数据库设计的实现。下面简要描述该卷中的某些章节和附录:

使用控制中心进行管理

- “使用 GUI 工具管理 DB2”介绍用来管理数据库的“图形用户界面”(GUI) 工具。

实现设计

- “在创建数据库之前”讨论创建数据库之前的前提条件。
- “创建数据库”讨论那些与数据库及相关数据库对象的创建相关联的任务。
- “改变数据库”讨论在改变数据库之前必须完成的工作以及那些与修改或卸下数据库及相关数据库对象相关联的任务。

数据库安全性

- “控制数据库存取”描述如何控制对数据库资源的存取。
- “审核 DB2 活动”描述如何检测和监控对数据的不需要的存取或不期望的存取。

移动数据

- “移动数据实用程序”是一页介绍，它说明移动数据的多种方法并指导您参考 *Data Movement Utilities Guide and Reference* 一书。

恢复

- “恢复数据库”讨论当选择数据库和表空间恢复方法时要考虑的因素，包括备份和复原数据库或表空间以及使用前滚恢复方法。

附录

- “使用分布式计算环境 (DCE) 目录服务”提供有关如何使用“DCE 目录服务”的信息。
- “数据库恢复的用户出口”讨论如何对数据库日志文件使用用户出口程序，并描述一些样本用户出口程序。
- “向多数据库分区服务器发出命令”讨论如何使用 *db2_all* 和 *rah* shell 脚本将命令发送至一个分区数据库环境中的所有分区。
- “DB2 Windows NT 版如何使用 Windows NT 安全性”描述 DB2 如何使用 Windows NT 安全性。
- “使用 Windows NT 性能监控程序”提供有关向“Windows NT 性能监控程序”注册 DB2 和如何使用性能信息的资料。
- “配置多个逻辑节点”描述如何在分区数据库环境中配置多个逻辑节点。
- “使用虚拟接口 (VI) 结构”描述如何启用“虚拟接口结构”以用于“DB2 通用数据库”。
- “轻量级目录存取协议 (LDAP) 目录服务”提供有关如何使用“LDAP 目录服务”的信息。
- “扩展控制中心”提供有关如何通过添加包含新操作的新工具栏按钮、添加新对象定义及添加新操作定义来扩展控制中心的信息。

第1部分 性能介绍

第1章 性能元素

*性能*是计算机系统在特定工作负荷下表现的方式。可通过系统的响应时间、处理能力和可用性来测量性能；性能受以下因素影响:

- 可用的资源
- 如何充分利用和共享这些资源。

一般情况下，如果要改善系统的成本效益比率，应调整性能。具体目标可能包括:

- 处理更大的或更紧迫的工作负荷，而不增加处理成本。(例如，增加工作负荷而不用购买新硬件或占用更多处理器时间。)
- 获得更快的系统响应时间或更高的处理能力，而不增加处理成本。
- 降低处理成本，而不会给用户服务带来负面影响。

将性能从技术指标转换为经济指标比较困难。调整性能一定会提高成本（占用人工时间和处理器时间），因此在进行调整前应衡量其成本和可能带来的效益。其中某些效益是有形的:

- 更有效地利用资源
- 能够向系统添加更多的用户。

其他效益是无形的，如由于响应更快而让用户更加满意。应考虑所有这些效益。

DB2 中集成了向导，它们可帮助您完成某些与性能相关的管理任务。这些任务通常只需花很少时间就可获得明显的性能改善。向导指导您逐步完成每个任务。可通过控制中心和“客户机配置辅助程序”找到向导。

“性能配置”向导通过更新配置参数来辅助您调整数据库性能，以满足企业需要。此向导以及较少使用的“创建数据库”向导，可以辅助您改进数据库的性能。其他向导有助于改进个别表和一般数据的存取性能。相关的向导包括: 创建表格、创建索引和配置多站点更新的向导。这些向导可从控制中心用鼠标右按钮单击一个对象来找到。

调整准则

以下准则可帮助您制定一个调整性能的总体方案。

请记住递减回报定律: 最大的性能收益通常来自最初的努力。以后的更改一般只产生越来越小的效益，并需要越来越多的努力。

不要只为调整而调整：进行调整以释放标识的约束。如果调整的资源不是造成性能问题的主要原因，这种调整对响应时间几乎不产生影响，除非您释放了主要约束，而且这种调整实际上会使后续调整工作更加困难。如果有可能明显改进性能的话，就在于对某些资源的性能改进，这些资源是影响响应时间的主要因素。

考虑整个系统：永远不能片面地调整一个参数或系统。在进行任何调整前，务必考虑它将对整个系统带来的影响。

一次更改一个参数：不要一次更改多个性能调整参数。即使您肯定所有更改都有好处，也没有任何办法来评估每个更改所带来的影响。如果一次更改多个参数，也不能有效地判断所做的更改的利弊。如果每次调整一个参数来改进一方面，几乎总是会影响至少一个您可能没有考虑到的其他方面。

按级别测量和重新配置：由于一次只应更改一个参数，因此一次也只能调整系统的一个级别。可使用以下的系统级别列表作为参考：

- 硬件
- 操作系统
- 应用服务器和请求器
- 数据库
- SQL 语句
- 应用程序

检查是否存在硬件和软件问题：某些性能问题可通过维修硬件和／或修订软件来解决。如果通过维修或修订就可解决问题，就不需要花过多时间来监控和调整系统。

在升级硬件前搞清楚问题：即使增加存储器或提高处理器能力可立即改善性能，也应花时间了解系统的瓶颈所在。可能花钱增加磁盘存储器后，才发现系统没有处理能力或可利用它的通道。

在开始调整前执行备份过程：正如前面所讲，某些调整可能产生无法预测的性能结果。如果此调整使性能降低，应撤消该调整，改试另一种调整。如果保存了以前的设置并可重新调用它，那么撤消不正确的信息将变得非常容易。

磁盘存储器

我们已经说过，构成系统的硬件可能会影响系统的性能。作为硬件影响性能的示例，我们考虑一些与磁盘存储器相关的蕴含内容。

磁盘存储器的管理以四种方式影响性能:

- **如何将存储器分区**:

  如何在索引和数据之间、表空间之间、缓冲池之间划分有限的存储器容量，在很大程度上决定了每一种资源在不同情况下的运行方式。

- **浪费的存储器**:

  浪费的存储器本身可能不影响使用它的系统的性能，但它可能代表可用于改善别处性能的一个资源。

- **分配磁盘 I/O**:

  如何良好地平衡几个磁盘存储设备和控制器上对磁盘 I/O 的需求，可影响数据库管理程序从磁盘检索信息的速度。

- **用完存储器**:

  达到可用存储器极限后，可能会降低整体性能。

性能改进过程

可使用以下过程改进任何系统的性能:

1. 建立性能指标。
2. 定义性能目标。
3. 制定性能监控方案。
4. 实行该方案。
5. 分析度量元素，以确定是否满足了目标。如果已满足，考虑减小度量元素数目，因为性能监控本身也使用系统资源。否则，继续执行下一步。
6. 确定系统中的主要约束。
7. 决定在哪些方面可负担得起性能改进，以及哪些资源可承受附加的负荷。（几乎所有调整都涉及到在系统资源和各种性能元素之间进行折衷。）
8. 调整系统配置。如果您认为更改多个调整选项是可行的话，应一次更改一个。如果在任何级别都没有选项可选，表明已达到资源极限，需要将硬件升级。
9. 返回到上面的步骤 4，继续监控系统。

定期，或在对系统或工作负荷进行重大更改后:

- 返回到上面的步骤 1。

- 重新检查目标和指标。
- 细化监控并调整策略。

可对系统进行多大程度的调整?

系统效率可提高的程度受一定的限制。考虑要投入多少时间和费用来改善系统性能，以及要投入多少额外的时间和费用来帮助系统的用户。

您的系统可能不需要任何调整也可运行得很好，但它可能没有发挥到极限。每个数据库都是唯一的。一旦创建了自己的数据库和使用该数据库的应用程序，应查看可用的调整参数，了解如何定制它们的设置来反映您的情况。在某些情况下，调整系统后只能获得较低的效益；但在大多数情况下，这种效益可能比较明显。

可从控制中心找到向导，以帮助调整数据库参数。从控制中心用鼠标右按钮单击想要调整的数据库，可找到“性能配置”向导。

当系统遇到性能瓶颈时，调整很可能发挥作用。如果已接近性能极限，同时又将系统上的用户数增加了大约 10%，那么响应时间可能远远不止增加 10%。在这种情况下，需要确定如何调整系统来抵消降低的性能。但有一个界限，超出这个界限再调整也没有用。达到该界限时，应考虑修改该环境内的目标和期望值。或者应考虑进行以下更改来更改系统环境：更多的磁盘存储器、更快的 CPU、附加 CPU、更多的主内存、更快的通信链路，或者这些更改的某种组合。

一种不太正式的方法

如果您没有足够的时间来设定性能目标并通过一种完备的方式来监控和调整，可听听用户的意见以改善性能。了解他们是否有与性能相关的问题。通常可提出几个简单的问题，来找到问题或确定从那里开始查找问题。例如，可以询问用户:

- “响应慢”达到何种程度？ 是比预期的慢 10% 还是慢数十倍？
- 问题是何时发现的？ 它是最近出现的，还是一直都存在？
- 是否知道其他用户也有相同的问题？ 投诉这些问题的是一两个人还是整个一组？
- （如果是整个一组用户遇到困难，他们是否与同一个终端控制器连接？）
- 遇到的问题是否与特定事务或应用程序相关？
- 问题是定期（如中午）出现还是持续出现？

综合考虑

因为理解关键概念和过程将有助于您解决其他性能问题，所以 DB2 的下层体系结构很重要。诸如存储器体系结构、数据管理、流程模型以及内存模型之类的主题最初全都在下一章提供。有关详情，参见第9页的『第2章 体系结构和进程概述』。

调整应用程序性能涉及到与您的应用程序相关的那些性能主题以及它们与数据库的交互作用。有一些是针对应用程序本身的主题：并发、锁定、优化级别、控制查询结果集、行锁定、使用复合 SQL。此外，也有一些简短的讨论：字符转换（当它与应用程序性能相关时）；存储过程；数据库激活以及并行处理的优点。有关详情，参见第37页的『第3章 应用程序考虑事项』。

还有针对查询优化的主题：影响查询优化的配置参数、节点组和表空间对查询优化的影响，以及索引对查询优化可能带来的比较大的影响。有关详情，参见第77页的『第4章 环境考虑事项』。

系统目录统计信息对应用程序如何有效地存取数据有很大影响。以下主题与统计信息相关：RUNSTATS 实用程序、分布统计信息、索引统计信息以及用户可更新的那些统计信息。有关详情，参见第93页的『第5章 系统目录统计信息』。

SQL 编译程序会编译每个应用程序并确定该应用程序的最佳存取方案。应用程序中的每个查询要经过评估，并可能执行若干个不同的操作以便最清晰地定义查询的目标。然后复查每个查询的不同存取方法（扫描和连接），以确定检索该查询请求的数据的最快方法。也考虑并行性的影响。有关详情，参见第121页的『第6章 了解 SQL 编译程序』。

该 DB2 产品中提供了几种不同的工具，可帮助了解应用程序的查询过程。这些工具有助于说明哪些因素影响应用程序性能。有关详情，参见第177页的『第7章 SQL 解释设施』。

除调整个别应用程序外，还应考虑那些应用程序运行所在的数据库的性能。数据库性能很大程度上是由内存使用情况确定的。有很多围绕内存的主题，它们都影响性能，如缓冲池、数据的预读取、并行 I/O、排序能力、重组表数据的需要以及数据库代理程序的概念。有关详情，参见第201页的『第8章 操作性能』。

可以建立一个“控制器”，来管理应用程序如何使用该数据库。有关详情，参见第237页的『第9章 使用控制器』。

可通过增加处理器数和数据库分区数来改进数据库性能。有关详情，参见第251页的『第10章 通过添加处理器调整配置』。

一旦增加了数据库分区数，就确保将数据库数据正确地分布到或重新分配到各数据库分区上。有关详情，参见第259页的『第11章 将数据重新分布至数据库分区』。

为了确定数据库的性能，可进行基准测试。基准测试的方法、如何准备基准测试、创建基准测试程序以及运行基准测试都是很重要的主题。有关详情，参见第265页的『第12章 基准测试』。

第277页的『第13章 配置 DB2』中分别讨论了大量的数据库管理程序参数和数据库配置参数

还有一些其他信息与这些性能主题相关。附录包括以下内容:

- 第417页的『附录A. DB2 注册表变量和环境变量』
- 第443页的『附录B. 解释表和定义』
- 第475页的『附录C. SQL 解释工具』
- 第517页的『附录D. db2exfmt - 解释表格式的工具』
- 第519页的『附录E. 配置 XA 事务管理程序以使用 DB2 UDB』
- 第531页的『附录F. 使用 DB2 资料库』

第2章 体系结构和进程概述

使用 DB2 的数据库操作的性能时，需要对基本概念（包括 DB2 体系结构和进程）有一些了解。本章介绍足够的信息以提供有关 “DB2 通用数据库” 如何工作的信息。后面的章节提供了有关此处找到的主题的更多细节，本章显示的内容是创建环境，以便稍后能更好地理解。

第一个图显示 DB2 UDB 的体系结构和进程的概述。

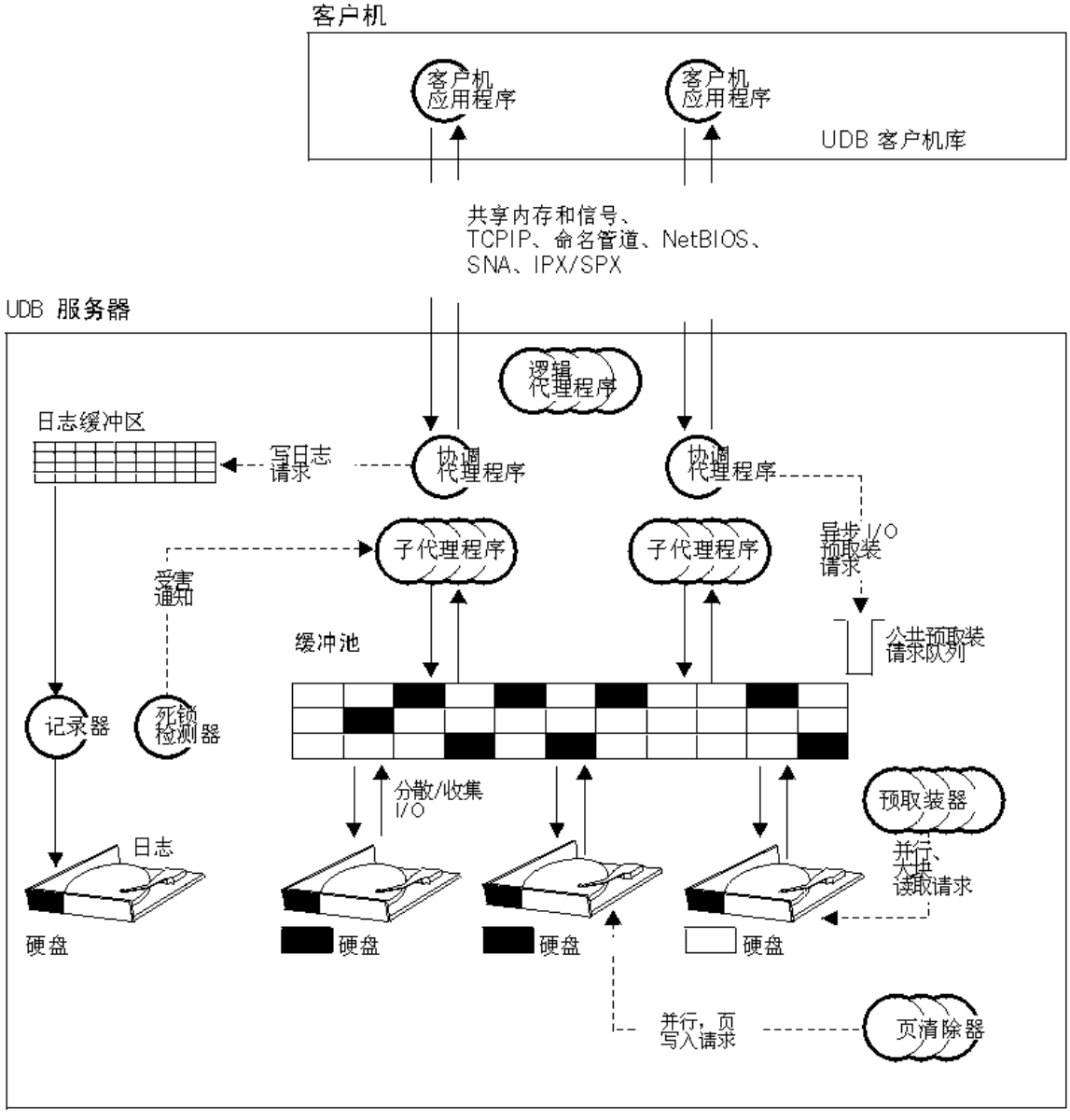


图 1. 体系结构和进程概述

在客户机方面，具有与“DB2 通用数据库”客户机库链接的本地和／或远程应用程序。

在客户机和“DB2 通用数据库”服务器之间是 『cloud』，表示本地或远程客户机与服务器之间的通信方法。本地客户机使用共享内存和信号进行通信；远程客户机使用协议（如命名管道 (NPIPE)、TCP/IP、NetBIOS, IPX/SPX 或 SNA）进行通信。

在服务器方面，活动由**引擎可调度单元** (EDU) 控制。在本章中的所有图形中，EDU 显示成圆或许多组圆。EDU在 Windows NT 和 OS/2 上作为线程实现（全都在单一进程中），在 UNIX 上作为进程实现。最常见的 EDU 类型是 DB2 代理程序。这些 EDU 代替应用程序执行大量 SQL 处理。EDU 的其他示例包括 DB2 预取装器和页清除器，它们负责各种类型的 I/O 处理。有关详情，参见第227页的『数据库代理程序』。

每个客户机应用程序都被指定了一个称为“协调代理程序”的唯一 EDU，它协调该应用程序的处理并与之通信。也可能会一起指定一组子代理程序来处理客户机应用程序请求。可能会指定多个子代理程序，以便当服务器所在的机器带有多个处理器时（如在对称多重处理环境中），客户机应用程序请求可以利用这些处理器。

所有代理程序和子代理程序都是使用特定入池算法来管理的，该算法将 EDU 的创建和/或破坏减至最少。

缓冲池是存储器内存的一个区域，用户表数据、索引数据和目录数据的数据库页被临时地移至该区域，并可能在该处被修改。因为从内存存取数据可以比从磁盘存取数据快得多，所以缓冲池对整体数据库性能起着关键的影响。应用程序所需的数据在缓冲池中存在得越多，与从磁盘存储器中查找出此数据相比，存取此数据所需的时间也越短。有关详情，参见第209页的『管理数据库缓冲池』。

缓冲池的配置与预取装器和页清除器 EDU 一起控制着应用程序所需的数据的可用性。

预取装器用来在应用程序需要数据之前从磁盘检索该数据，并将其移到缓冲池中。例如，如果没有数据预取装器，则需要扫描大量数据的应用程序将必须等待数据从磁盘移到缓冲池中。应用程序的代理程序将异步预读请求发送至公共预取装队列。当预取装器可用时，它们使用大块或分散读取输入操作来将请求的页从磁盘读入缓冲池，从而实现那些请求。使用多个磁盘来存储数据库数据意味着可以将数据分块到多个磁盘上。这种数据分块使预取装器能同时使用多个任务来检索数据。有关详情，参见第214页的『将数据预读取至缓冲池』和第216页的『配置 I/O 服务器启用预读取和并行 I/O』。

使用预取装器来将数据读入缓冲池。使用页清除器来将数据从缓冲池移回到磁盘中。

页清除器是后台 EDU，它独立于应用程序代理程序（这些代理程序在缓冲池中寻找不再需要的页并将这些页写至磁盘）。页清除器可以确保缓冲池中有空间供预取装器正在检索的页使用。

如果不存在独立的预取装器和页清除器 EDU，则应用程序代理程序将必须执行缓冲池与磁盘存储器之间的所有数据读取和写入操作。

当有多个应用程序使用数据库中的数据时，两个或多个应用程序之间有可能发生“死锁”。下图对死锁进行了说明。

死锁概念

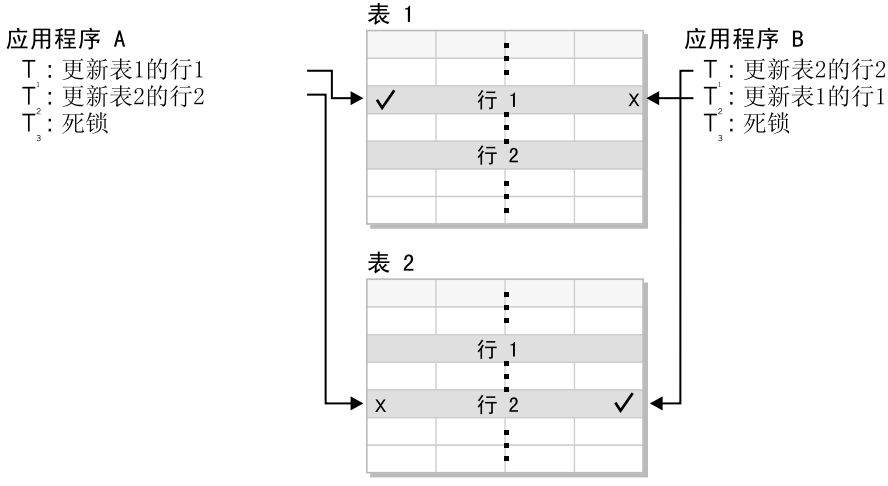


图 2. 死锁检测器

“死锁”表示多个应用程序等待另一应用程序释放数据上的锁定。每一个等待的应用程序都通过锁定持有其他应用程序所需的数据。此锁定的数据是一个或多个其他应用程序所必需的，而这些应用程序又持有其他应用程序所需的数据。相互等待另一应用程序释放所持有的数据上的锁定将导致死锁：应用程序会永远等待下去，直到“另一”应用程序释放所持有的数据上的锁定为止。其他应用程序不会自动释放对它们所需的数据的锁定。需要一个进程来中断这些死锁状态。

就象名称所指的那样，死锁检测器监控关于等待锁定的代理程序的信息。死锁检测器任意选择其中一个处于死锁状态的应用程序，以释放当前由“志愿”应用程序挂起的锁定。通过释放该应用程序的锁定，其他正在等待的应用程序所需的数据便可供使用。于是，以前进行等待的应用程序便可以使用所需的数据来完成对数据库中数据的操作。

对缓冲池中数据页的更改被记录下来。存在一个日志缓冲区，它与一个记录器 EDU 相关联。更新数据库中数据记录的代理程序更新缓冲池中的关联页，并编写日志记录。日志记录包含重做或撤消该更改所必需的信息。缓冲池中的页和日志缓冲区中的日志记录都不会立即写至磁盘，以优化性能。记录器 EDU 和缓冲池管理程序共同实现一个“超前写记录”(WAL) 协议，此协议确保在将数据页的关联日志记录写至日志之前，不将该数据页写至磁盘。WAL 协议确保日志中总是有足够的信息来从崩溃恢复和复原数据库一致性。如果将某一页上未落实的更新写至磁盘，则崩溃恢复使用相关联的日志记录中的撤消信息来撤消该更新。如果未将已落实的更新写至磁盘，则崩溃恢复使用相关联的日志记录中的重做信息来重做该更新。

注: 进行 COMMIT 时，事务中的所有日志记录都被清仓至磁盘（如果它们尚未被清仓的话）。

存储体系结构

在讨论存储器体系结构时，我们将会考虑:

- 『数据库目录』
- 第15页的『表空间』

数据库目录

当创建一个数据库时，关于该数据库的信息（包括缺省信息）放在一个目录中。此目录结构的创建位置依赖于您在 CREATE DATABASE 命令中提供的信息。如果创建数据库时您不指定路径或驱动器的位置，则将使用缺省位置。

我们建议您明确地指出您希望在哪里创建数据库。

在您在 CREATE DATABASE 命令中指定的目录中，将使用实例名创建一个子目录。这个子目录确保不同实例在同一目录中创建的数据库不会使用相同的路径。在实例名子目录下面，将创建一个名为『NODE0000』的子目录。这个子目录用来区分多逻辑分区数据库环境中的分区。在节点目录下面，将创建名为『SQL00001』的子目录。这个子目录使用数据库记号进行命名，表示正在创建的数据库。它还用来区分此实例在 CREATE DATABASE 命令中指定的目录中创建的数据库。

目录结构的外观将类似于:

```
<your_directory>/<your_instance>/NODE0000/SQL00001/
```

数据库目录将包含几个作为 CREATE DATABASE 命令的一部分创建的文件。缓冲池信息包含在文件 SQLBP.1 和 SQLBP.2 中。表空间信息包含在文件 SQLSPCS.1 和 SQLSPCS.2 中。这些文件的每一个都有两份的目的是用来备份这些文件中的信息。

数据库配置信息包含在 SQLDBCON 中。您很容易阅读历史文件 *db2rhist.asc* 及其备份 *db2rhist.bak* 的内容，它们包含关于备份、复原、装入表、重组表、改变表空间以及对数据库的其他更改的历史信息。

日志控制文件 SQLOGCTL.LFH 包含关于活动日志的信息。恢复处理使用此文件中的信息来确定要在日志中后退多远来开始恢复。SQLOGDIR 子目录包含实际的日志文件。

注: 您应确保日志子目录映射至与数据不同的磁盘。这样，磁盘问题就只能影响到数据或日志，而不会同时影响到二者。并且，这能带来实实在在的性能好处，即日志文件与数据库容器不会争用同一磁盘头的移动。您可以使用 *newlogpath* 数据库配置参数来更改日志子目录的位置。

将创建 SQLT* 子目录，它们包含运作的数据库所需的缺省“系统管理空间”(SMS) 表空间。将创建三个缺省表空间:

- SQLT0000.0 子目录包含带有系统目录表的目录表空间。
- SQLT0001.0 子目录包含缺省临时表空间。
- SQLT0002.0 子目录包含缺省用户数据表空间。

当考虑表空间时，也称之为“容器”。对于 SMS 表空间，容器是操作系统目录。

每个子目录或容器中都会创建一个名为 『SQLTAG.NAM』 的文件。这个文件标记该子目录在使用中，因此后续表空间创建不会尝试使用这些子目录。容器子目录下面还会创建其他文件，它们具有不同的扩展名，目的是区分这些文件中存储的数据的类型。这些扩展名包括:

- SQL*.DAT（包含非长型表数据）
- SQL*.LF（包含 LONG VARCHAR 或 LONG VARGRAPHIC 数据）
- SQL*.LB（包含 BLOB、CLOB 或 DBCLOB 数据）
- SQL*.LBA（包含关于 SQL*.LB 文件的分配和空闲空间信息）
- SQL*.INX（包含索引表数据）
- SQL*.DTR（包含用于重组 SQL*.DAT 文件的临时数据）
- SQL*.LFR（包含用于重组 SQL*.LF 文件的临时数据）
- SQL*.RLB（包含用于重组 SQL*.LB 文件的临时数据）
- SQL*.RBA（包含用于重组 SQL*.LBA 文件的临时数据）

表空间

支持两种类型的表空间: “系统管理空间”(SMS) 和“数据库管理空间”(DMS)。它们各有各的特性，适合于不同的环境。有关设计和选择表空间的详情，请参考*管理指南: 计划*。

SMS 表空间

“系统管理空间”(SMS) 表空间存储操作系统文件中的数据。表空间中的数据按系统中所有容器上的数据块进行划分。**数据块**是对数据库定义的一组连续页。表空间中的每个表都被赋予它自己的文件名，此文件名在所有容器中使用。文件扩展名指示了该文件中存储的数据的类型。每个表的起始数据块都以“循环”方式放在容器中。这将空间需求均匀地分布在表空间中的所有容器上。当有非常多的小型表时，这一点非常重要。

当需要附加的空间时，便执行空间分配。在缺省情况下，每次分配一页空间。

DMS 表空间

数据库管理程序通过“数据库管理空间”(DMS) 表空间控制存储空间。当定义 DMS 表空间时，选择设备或文件的列表，使其属于该表空间。那些设备或文件上的空间由 DB2 数据库管理程序管理。对于 SMS 表空间和容器，DMS 表空间和数据库管理程序使用“按数据块分块”来确保数据均匀地分布在所有容器上。

DMS 表空间与 SMS 表空间之间的差异在于，对于 DMS 表空间，空间是在创建表空间时分配的，而不是在需要时分配的。

并且，在这两种类型的表空间上，数据的放置也会有所不同。例如，考虑高效表扫描的这样的一个需要: 重要的是数据块中的页在物理上是连续的。对于 SMS，操作系统的文件系统决定了每个逻辑文件页的物理放置位置。根据文件系统上其他活动的级别以及用来确定放置位置的算法的不同，这些页可能连续分配，也可能不连续分配。然而，对于 DMS，因为数据库管理程序直接与磁盘打交道，所以它可以确保这些页在物理上是连续的。

这一关于页在存储器中的连续放置的一般性陈述有一个例外。当使用 DMS 表空间时，有两个容器选项: 原始设备和文件。当使用文件容器时，数据库管理程序在创建表空间时分配整个容器。这种一开始就分配整个表空间的结果是，即使由文件系统执行分配，物理分配也通常（但不保证）是连续的。当使用原始设备容器时，数据库管理程序控制整个设备，并总是确保数据块中的页是连续的。

与 SMS 表空间不同，组成一个 DMS 表空间的容器的容量不必接近相等。然而，建议让容器的容量相等，或接近相等。并且，如果任何容器已满，则 DMS 表空间可以使用其他容器中的任何可用的空闲空间。

当使用 DMS 表空间时，您应考虑将每个容器与不同的磁盘相关联。这使表空间容量可以更大，并且能够利用并行 I/O 操作。

下图显示了 DMS 表空间的逻辑地址映射。

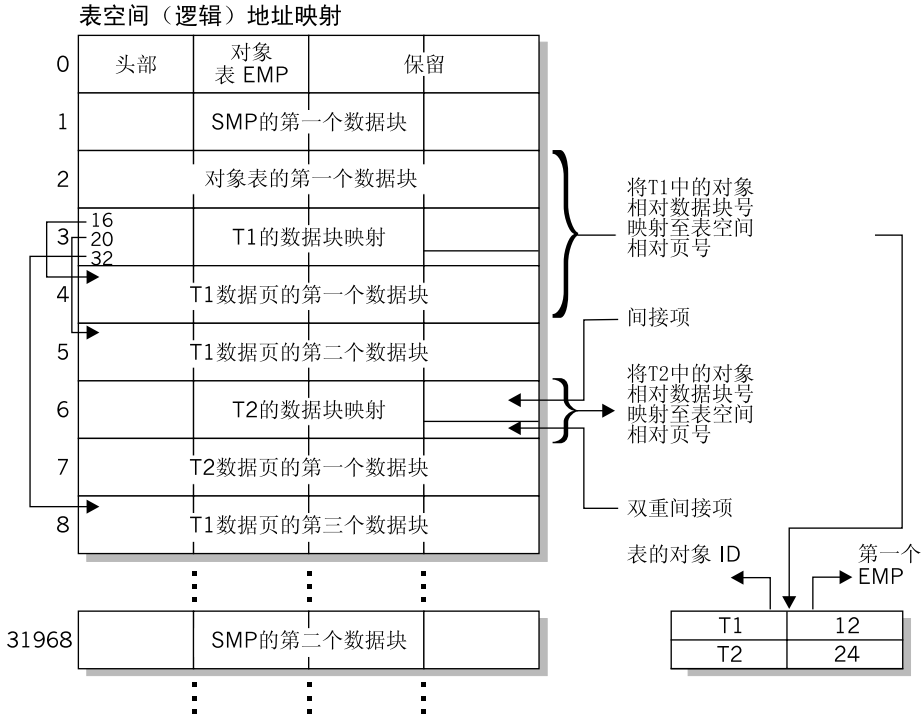


图 3. DMS 表空间

CREATE TABLESPACE 语句在数据库中创建新的表空间，向表空间分配容器，并在目录中记录表空间定义和属性。创建表空间时定义的其中一个属性是数据块大小。数据块是表空间中的空间分配单位。它只是一组连续的页。数据块大小便是连续页的数目。只有一个表（或其他对象，如一个索引）能够使用任何一个数据块中的页。将逻辑表空间地址映射中的数据块分配给表空间中创建的所有对象（表、索引和其他对象）。数据块一次只属于一个对象。数据块分配通过“空间映射页”(SMP) 进行管理。

逻辑表空间地址映射中的第一个数据块是包含内部控制信息的表空间的首部。第二个数据块是表空间的“空间映射页”(SMP) 的第一个数据块。SMP 数据块以固定间隔方式分布在表空间中。每个 SMP 数据块仅仅是当前 SMP 数据块到下一 SMP 数据块的数据块位映射。位映射用来跟踪哪些中间数据块在使用中，哪些未使用。

SMP 后面的一个数据块是表空间的对象表。对象表是一个内部表，它跟踪表空间中存在哪些用户对象，以及它们的第一个“数据块映射页”(EMP) 数据块的位置。每个对象都有其自己的 EMP，它提供了指向该对象的每一页的映射，这些映射存储在逻辑表空间地址映射中。

对象表是一个内部关系表，它将对象标识符映射至该表的第一个 EMP 数据块的位置。这个 EMP 数据块直接或间接地映射出该对象中的所有数据块。每个 EMP 都包含一连串的项。每一项都将对象相对数据块号映射至该对象数据块所在的表空间相对页号。直接 EMP 项直接将对象相对地址映射至表空间相对地址。第一个 EMP 数据块中的最后一个 EMP 页包含间接项。间接 EMP 项映射至 EMP 页，EMP 页然后映射至对象页。第一个 EMP 数据块中最后一个 EMP 页中的最后 16 项包含双重间接项。

逻辑表空间地址映射中的数据块以循环方式在与该表空间相关联的容器上分块。

比较 SMS 和 DMS 表空间

SMS 与 DMS 表空间相比，SMS 表空间特别适合于一般用途。SMS 表空间能提供非常好的性能，且管理成本非常低。如果力求达到最佳性能，则最好选择 DMS 表空间。因为使用文件容器或 SMS 表空间移动数据时会发生双重缓冲，所以设备容器能提供最佳的性能。(当在数据库管理程序级首先缓冲一次数据，然后在文件系统级再缓冲一次数据，便发生了双重缓冲。)

数据管理

在创建数据库、创建表空间、创建表和将数据放入表之后，最为有趣的是了解表是如何组织的，以及如何使用索引来检索该表数据。

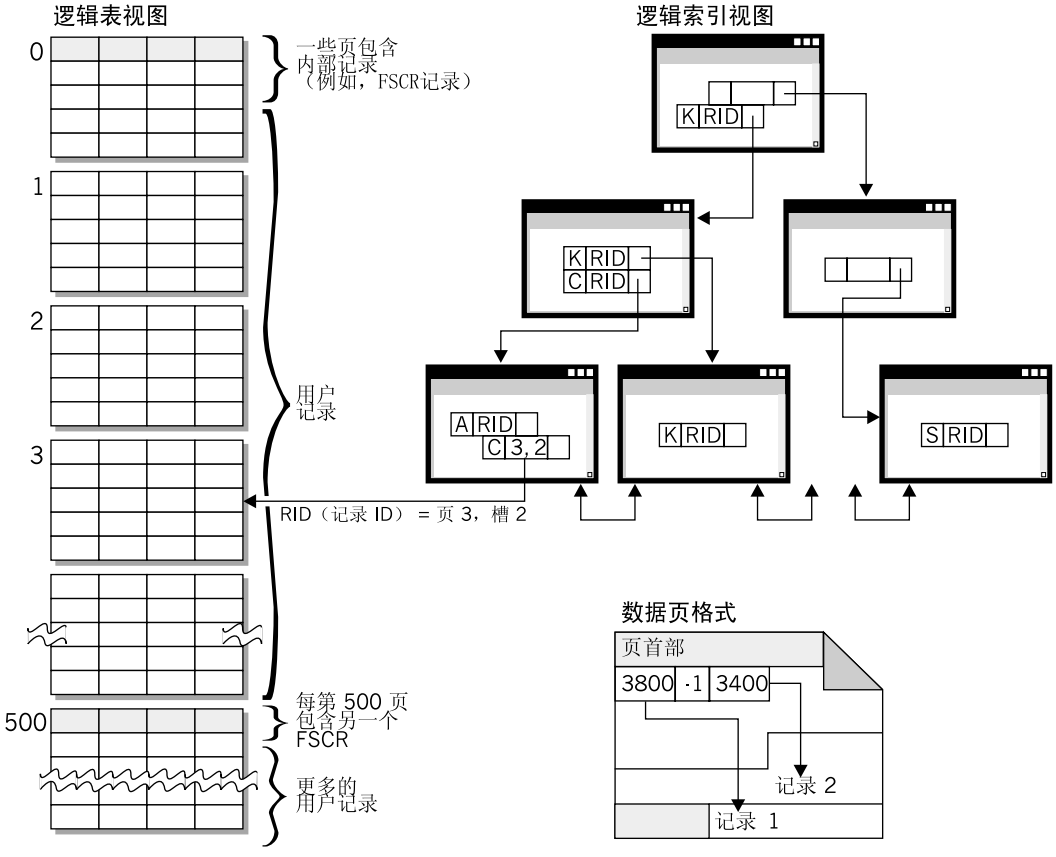


图 4. 表、记录和索引

在逻辑上，表数据组织成数据页列表。这些数据页根据表空间的数据块大小在逻辑上分组在一起。例如，如果数据块大小是 4，第 0 至 3 页是第一个数据块的一部分，则第 4 至 7 页是第二个数据块的一部分，依此类推。

根据数据页大小以及记录大小的不同，每个数据页中所包含的记录数可能会有所变化。一页最多可以放 255个记录。大多数页只包含用户记录。然而，少数页包括特殊的内部记录，DB2 使用那些记录来管理表。例如，每第 500 个数据页上都有一个“空闲空间控制记录”(FSCR)。这些记录映射出下面每 500 个数据页（直到下一 FSCR为止）上存在多少空闲空间可供新记录使用。当将记录插入表时，将使用这部分可用空闲空间。

在逻辑上，索引页组织成 B 树，这可以有效地在表数据中定位带有给定关键字值的记录。索引页上的项数不是固定的，但依赖于关键字的大小。对于 DMS 表空间

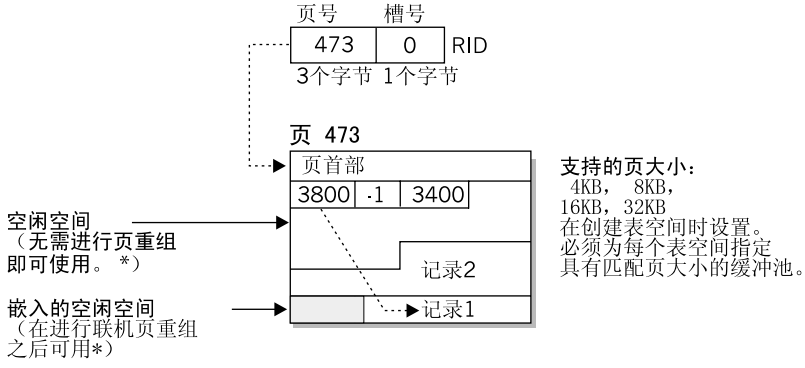
中的表，索引页中的记录标识符 (RID) 使用表空间相对页号，而不是对象相对页号。这使**索引扫描**能够直接存取数据页，而不需要“数据块映射页” (EMP) 来进行映射。

每个数据页都具有以下格式：每个数据页开头都有一个页首部。在页首部后面，有一个槽目录。槽目录中的每一项都与该页中的另一记录相对应。该项本身是数据页中记录开始位置的字节位移。值为 -1的项与已删除的记录相对应。

记录标识符和页

记录标识符 (RID) 由三个字节的页号及随后的一个字节的槽号组成。在使用索引来标识 RID 之后，便使用该 RID 来到达正确的数据页和该页上正确的槽号。槽的内容是页中正在查找的记录的开始位置的字节位移。一旦对记录指定了 RID，在进行表重组之前，该 RID 便不会更改。

数据页和RID格式



* 例外：由未落实的 DELETE 保留的任何空间均不可用。

图 5. 数据页和 RID 格式

重组表时，删除记录后在数据页上留下的嵌入空闲空间被转换成可使用的空闲空间。根据记录在数据页上的移动重新定义 RID，以利用可使用的空闲空间。

DB2 支持不同的页大小。对于有可能连续存取行的工作负荷，请使用较大的页大小。例如，“决策支持”应用程序或大量使用临时表的场合使用的便是顺序存取。对于更有可能进行随机存取的工作负荷，使用较小的页大小。例如，OLTP 环境中使用的便是随机存取。

有关重组表的详情，参见第223页的『重组目录和用户表』。

空间管理

使用 SQL INSERT 语句来将新信息放入表。这样做时，将遵循一个 INSERT 搜索算法来完成工作。首先，使用“空闲空间控制记录”(FSCR) 来查找带有足够空间的页。然而，即使 FSCR 指出某页带有足够的空闲空间，该空间也可能因为被另一事务的未落实的 DELETE 语句所“保留”而不可用。结果是，您应确保所有事务频繁地落实 (COMMIT)，否则，未落实释放的空间将不可用。

并不会搜索表中的所有 FSCR。DB2MAXFSCRSEARCH 注册表变量限制了尝试 INSERT 时要考虑的 FSCR 数。此注册表变量的缺省值是 5。如果在 5 个 FSCR 中找不到空间，则将正在插入的记录追加至表末尾。并且，为了优化 INSERT 速度，还可能会将后续记录追加至表末尾，直到填满了两个数据块为止。在填满了两个数据块之后，下一 INSERT 在上一搜索结束处的 FSCR 继续搜索。

注: DB2MAXFSCRSEARCH 的值很重要。要优化 INSERT 速度（可能的代价是表增长得更加快），请将此注册表变量设置为较小的值。要优化空间重用（可能的代价是 INSERT 速度减慢），请将此注册表变量设置为较大的值。

在搜索了整个表之后，将追加要插入的记录，而不会进行附加的搜索。另外，也不再执行使用 FSCR的搜索，直到在表中某处创建了空间为止（例如，跟随 DELETE）。

另外还有两个搜索算法选项。第一个是 APPEND MODE。在此方式中，总是将新行追加至表末尾。不执行 FSCR 的搜索或维护。此选项使用 ALTER TABLE APPEND ON 语句启用，它可以提高只会增长的表（例如日志）的性能。第二个选项是对表定义分群索引。在这种情况下，数据库管理程序尝试将记录插入到带有类似索引关键字值的其他记录所在的页上。如果该页上没有空间，则尝试将该记录放到周围的页中。如果仍不成功，则使用如上所述的 FSCR 搜索算法 – 只有一个很小的例外：使用最差匹配方法，而不是使用首先匹配方法。这种最差匹配方法往往能选择带有更多空闲空间的页。这样做的目的是为带有此关键字值的行建立新的分群区。

当对表定义分群索引时，请在装入或重组表之前使用 ALTER TABLE... PCTFREE。在装入和重组之后，PCTFREE 子句在该表的数据页上留下给定百分比值的空闲空间。这增加了群集索引操作能在期望的页上找到空闲空间的可能性。

数据页和溢出记录

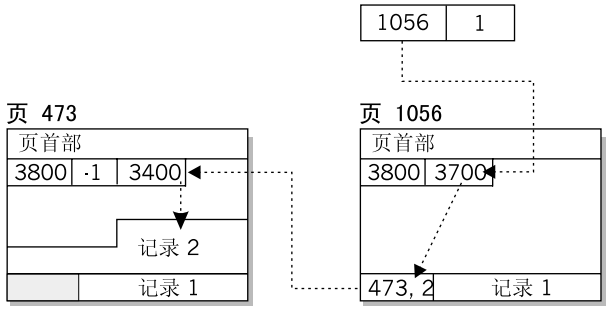


图 6. 数据页和溢出记录

当更新请求放大了现存记录，导致它在当前页中装不下时，便有可能溢出记录。放大后的记录作为溢出记录被插入到另一带有足够空间的页中。原始 RID 被转换为指针记录，它包含该溢出记录的新 RID。表的索引存放原始 RID，要得到请求的数据记录，需要进行附加的页读取操作。溢出记录越多，意味着附加页读取操作越多，存取表的性能也就越低。重组表将除去溢出记录。然而，应尽可能地避免放大记录的更新请求，以避免溢出记录。

索引管理

DB2 索引是最优的 B 树实现，它基于一种高效率且高并行度的索引管理方法，该方法使用超前写记录。

最优 B 树实现的叶子页上带有双向指针，这使单个索引能支持正向或反向扫描（但不是同时支持二者）。索引页分割通常刚好是对半的，但在 high-key 页上除外，在那些页上，使用 90/10分割。即，索引关键字的高 10% 放在新页上。这种类型的索引页分割对于特定工作负荷而言非常有用，这些工作负荷的 INSERT 请求通常使用新的 high-key 完成。

当除去一页上的最后一个索引关键字时，便从索引中释放该页。如果创建索引时选择了 MINPCTUSED 子句，便会发生这一规则的例外情况。使用此子句指示索引可以联机重组；且使用此子句给出的值是索引叶子页上使用的空间的最小百分比阈值。在从索引页中删除关键字之后，如果该页上使用的空间的百分比等于或低于给出的值，则尝试将剩余的关键字与相邻页的关键字合并。如果有足够的空间，则执行合并，并删除一个索引叶子页。使用此子句能改进空间重用情况；然而，如果使用的值太高，则尝试合并所花的时间会增加，但成功的可能性却更低。建议您将此子句的值总是设置为小于 50%。

CREATE INDEX 语句的 INCLUDE 子句允许除了包括关键字列之外，还包括索引叶子页上的指定列。这会增加适合于纯索引存取的查询数。然而，这也会增加索引空间需求，并且，如果频繁地更新所包括的列，则还可能会增加索引维护成本。只使用关键字列，而不使用所包括的列来对 B 树进行定序。

锁定

数据库管理程序提供并行性控制并借助于锁定防止不受控制的资源和数据存取。**锁定**将应用程序与数据库管理程序资源或数据记录相关联。锁定控制其他应用程序可以如何存取同一资源或数据记录。

数据库管理程序根据下列各项来适当地使用记录级锁定或表级锁定:

- 隔离级别是在预编译或应用程序与数据库联编时指定的。隔离级别可以是以下之一: “未落实的读取”(UR)、“游标稳定性”(CS)、“读取稳定性”(RS) 或“可重复的读取”(RR)。使用不同的隔离级别来控制对未落实的数据的存取、防止丢失更新、允许对数据进行不可重复的读取以及防止幻象读取。请使用能够满足应用程序需要的最小隔离级别。
- 优化器选择的存取方案。表扫描、索引扫描和其他数据存取方法其每一个都需要不同的数据存取类型。
- 表的 LOCKSIZE 属性。ALTER TABLE 语句上的 LOCKSIZE 子句指示存取该表时使用的锁定的粒度。选项是 ROW（表示行锁定）或 TABLE（表示表锁定）。对于只读表，使用 ALTER TABLE... LOCKSIZE TABLE。这减少了数据库活动所需的锁定数。
- 专门用于锁定的内存量。专门用于锁定的内存量由 *locklist* 数据库配置参数控制。如果锁定列表已满，则性能可能会因锁定逐步升级以及数据库中共享对象上的并行性降低而下降。如果频繁地遇到锁定逐步升级，则增加 *locklist* 和／或 *maxlocks* 的值。

确保所有事务频繁地进行落实 (COMMIT) 以释放挂起的锁定。

通常，除非下列其中一项为真，否则使用记录级锁定:

- 选择的隔离级别是“未落实的读取”(UR)。
- 选择的隔离级别是“可重复的读取”(RR)，存取方案需要不带谓词的扫描。
- 表的 LOCKSIZE 属性是 『TABLE』。
- 锁定列表已满，发生锁定逐步升级。
- 通过 LOCK TABLE 语句获取了显式的表锁定。LOCK TABLE 语句使并行应用程序进程不能更改表或使用表。

锁定逐步升级指的是将一个或多个记录锁定转换为表锁定。互斥锁定逐步升级是这样的锁定逐步升级：获取的表锁定是互斥锁定。锁定逐步升级降低了并行性，应该避免。

记录锁定的持续时间随使用的隔离级别的不同而有所变化:

- “未落实的读取”(UR) 扫描：除非更改记录，否则不挂起记录锁定。
- “游标稳定性”(CS) 扫描：仅当游标定位在记录上时，才挂起记录锁定。
- “读取稳定性”(RS) 扫描：事务期间只挂起合格的记录锁定。
- “可重复的读取”(RR) 扫描：事务期间挂起所有记录锁定。如果您所处的环境不需要或不想要此隔离级别，则使用 DB2_RR_TO_RS 注册表变量。这告知数据库管理程序避免所需的附加锁定，以启用 RR 语义，结果是，提高了性能。

有关此主题的详情，参见第43页的『锁定』。

记录

有两个记录策略选项:

- 循环记录指的是日志记录填充日志文件，然后覆盖初始日志文件中的初始日志记录。被覆盖的日志记录不可恢复。
- 保留日志记录指的是日志文件一旦填满，便将其归档。使用新日志文件来存放日志记录。保留日志文件启用**前滚恢复**。前滚恢复根据日志中记录的已完成的工作单元（事务）来对数据库重新应用更改。您可以指定前滚恢复是恢复至日志末尾，还是恢复至日志末尾之前的特定时间点。

无论选择了哪一个选项，对正规数据和索引页所作的所有更改都被写至日志缓冲区。日志缓冲区中的数据只有在下面这些时候才会被强制写至磁盘:

- 在将对应的数据页强制写至磁盘之前。这称为“超前写记录”。
- 在执行 COMMIT 时；或在达到“入组的 COMMIT 数”(*mincommit*) 数据库配置参数的值之后。
- 当日志缓冲区变满时。使用双重缓冲来防止 I/O 等待。

注: 当事务使用 COMMIT 语句完成时，所有更改过的页都被清仓至磁盘，以确保可恢复性。

当事务很短时，因为执行落实时对日志执行清仓的频率非常高，所以日志 I/O 会成为“瓶颈”。在这种环境中，将 *mincommit* 配置参数设置为大于 1 的值可以除去“瓶颈”。当使用大于 1 的值时，数个事务的 COMMIT 操作或者是被挂起，或者是“以批处理方式执行”。第一个要执行 COMMIT 的事务等待更多的 (*mincommit* - 1) 事务 COMMIT；然后将日志强制写至磁盘，之后，所有事务对它们的应用程序作出响应。结果是只有一次日志 I/O，而不是发生数次个别的日志 I/O。

为了避免过分降低响应速度，每个事务最多只等待其他要执行 COMMIT 的事务（*mincommit* - 1 个）1 秒钟。如果一秒钟到了，则等待的事务将自行强制写入日志并响应它的应用程序。这允许您设置 *mincommit*，并且，当正在处理少数几个事务时，也不会太关心性能。

通过影子分页跟踪对“大对象”(LOB) 和 LONG VARCHAR 的更改。除非使用日志保留，且在 CREATE TABLE 语句上将 LOB 列定义为不使用 NOT LOGGED 子句，否则不记录 LOB 列更改。与对正规数据页的更改一样，对 LONG 或 LOB 数据类型的分配页的更改被记录下来。

进行更新时发生的情况

代理程序更新页时如何处理日志和数据页？此处描述的协议将事务所必需的 I/O 数降至最低，并且还确保了可恢复性。

首先，使用互斥锁定将要更新的页锁住。将描述如何重做和撤消更改的日志记录写至日志缓冲区。作为此操作的一部分，获取日志序号 (LSN)，并将其存储在正在更新的页的页首部中。然后对该页进行更改。最后，取消锁定该页。因为对该页的更改尚未被写至磁盘，所以该页被认为是“脏”的。日志缓冲区也已被更新。

将需要把日志缓冲区和“脏”数据页中的数据强制写至磁盘。为了提高性能，这些 I/O 被延迟至一个方便的时间（例如，在系统装入时的暂停期间），或延迟至必须确保可恢复性的时间，或延迟至固定的恢复时间。更明确地说，“脏”页在下面这些时候被强制写至磁盘:

- 当另一代理程序选择它作为干扰页的时候。
- 当作为下列各项的结果，页清除器对该页进行操作的时候:
  - 另一代理程序选择它作为干扰页。
  - 超过 *chngpgs_thresh* 数据库配置参数百分比值。一旦超过，异步页清除器便被“唤醒”，并将更改过的页写至磁盘。
  - 超过 *softmax* 数据库配置参数百分比值。一旦超过，异步页清除器便被“唤醒”，并将更改过的页写至磁盘。
- 当作为调用了 NOT LOGGED INITIALLY 子句并发出了 COMMIT 的表的一部分更新该页时。当执行 COMMIT 时，所有已更改的页都被清仓至磁盘，以确保可恢复性。

在下面这些时候，记录器引擎可调度单元 (EDU) 将日志缓冲区强制写至磁盘:

- 在将对应的数据页强制写至磁盘之前。这称为“超前写记录”。
- 在执行 COMMIT 时；或在达到“入组的 COMMIT 数”(*mincommit*) 数据库配置参数的值之后。

- 当日志缓冲区变满时。使用双重缓冲来防止 I/O 等待。

进程模型

本地和远程应用程序进程可以使用同一数据库。远程应用程序是从作为数据库机器的远程机器启用数据库操作的应用程序。本地应用程序直接与服务器机器上的数据库相连。

下图中的每一个圈都表示引擎可调度单元 (EDU), 它在 UNIX 平台上称为 “进程”，在 Windows NT 和 OS/2 平台上称为 “线程”。

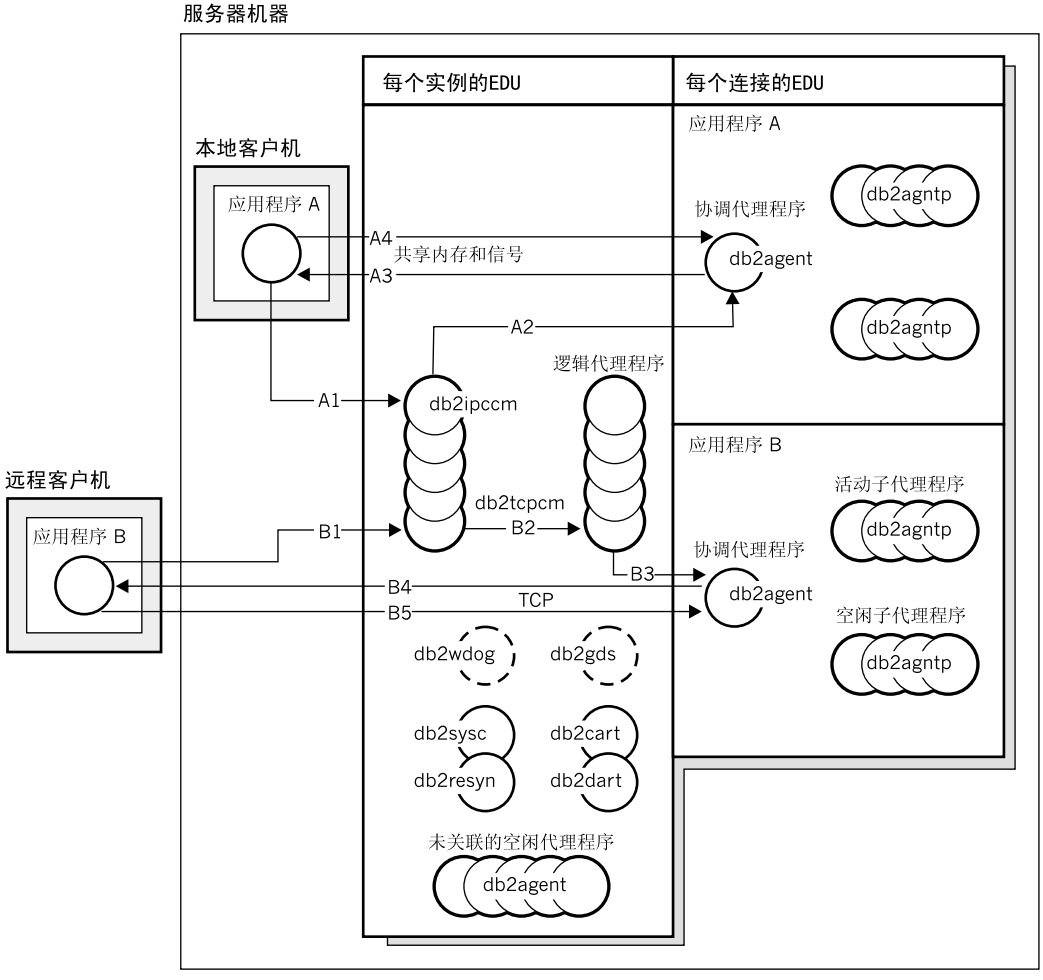


图 7. 进程模型概述

在可以执行应用程序要对数据库执行的工作之前，必须在应用程序与数据库管理程序之间建立一种通信方法。

在上图中的 A1 处，本地客户机首先通过使用 db2ipccm 引擎可调度单元 (EDU) 建立通信。A2 处的此 EDU 使用 db2agent EDU，该 EDU 已成为来自本地客户机的应用程序请求的协调代理程序。该协调代理程序与 A3 处的客户机应用程序联系，并建立共享内存以及客户机应用程序与 A4 处的数据库之间的信号通信。本地客户机上的应用程序与数据库相连。

在上图的 B1 处，远程客户机首先通过使用 db2tcpcm EDU 建立通信。如果选择了另一通信协议，则将使用适当的 EDU。B2 处的 db2tcpcm EDU 使用逻辑代理程序。B3 处的此 EDU 使用 db2agent EDU，该 EDU 已成为来自远程客户机的应用程序请求的协调代理程序。协调代理程序与 B4 处的客户机应用程序联系，并在客户机应用程序与 B5 处数据库之间建立 TCP/IP 通信。远程客户机上的应用程序与数据库相连。

在此图中要注意的其他事项:

- 有两类代理程序: 逻辑代理程序和工作代理程序。逻辑代理程序表示与数据库管理程序相连的应用程序。工作代理程序执行应用程序请求，但并不与任何特定应用程序永久相连。
- 有四种类型的工作代理程序: 活动协调代理程序、子代理程序、不活动代理程序和空闲代理程序。
- 客户机应用程序的每一个由逻辑代理程序所表示的进程或线程都将与一个活动的协调代理程序相链接。
- 在分区数据库环境以及启用的分区内并行性环境中，协调代理程序将数据库请求分配给子代理程序 (db2agntp)。子代理程序为应用程序执行请求。
- 有一个代理程序池 (db2agent)，空闲代理程序在那里等待新工作。
- 还有其他用来管理客户机连接、日志、两阶段 COMMIT、备份和复原任务以及其他任务的 EDU。

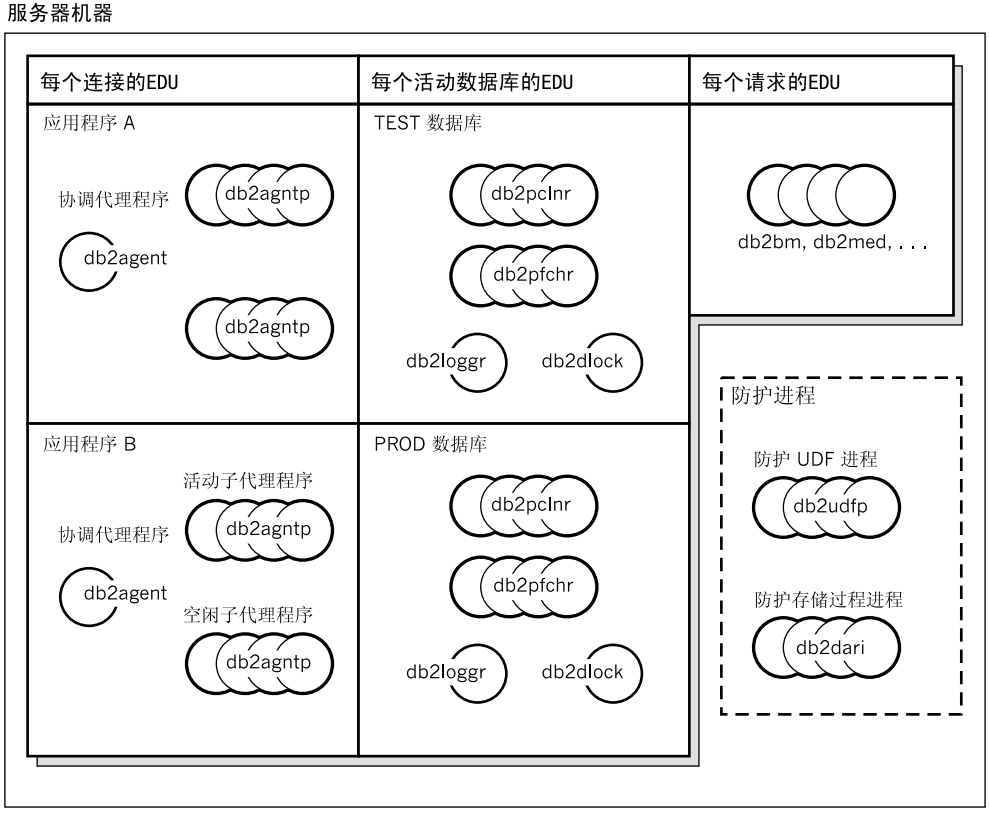


图 8. 进程模型第二部分

此图显示了作为服务器机器环境一部分的附加引擎可调度单元 (EDU)。每个活动数据库都有它自己的预取装器共享池 (db2pfchr) 和页清除器共享池 (db2pclnr)，并有它自己的记录器 (db2loggr) 和死锁检测器 (db2dlock)。

图右下角标记了 『db2udfp』 和 『db2dari』 的圆圈分别表示作为防护 “用户定义函数” (UDF) 和 “存储过程” 在 “DB2 通用数据库” 中运行的进程。管理这些进程的目的是将创建和破坏它们的成本降至最低。 *keepdari* 数据库管理程序配置参数的缺省值是 『YES』，这使该存储过程进程在下一存储过程调用时可供重新使用。

注: 还有直接在代理程序的地址空间中运行的不受防护 UDF 和 “存储过程”。通过以这种方式工作，可以获得更好的性能。然而，因为它们对代理程序的地址空间的存取权不受限制，所以在使用之前，需要对它们进行严格的测试。

有关详情，参考 *Application Development Guide* 中有关存储过程的章节。

多分区处理模型是单分区处理模型的逻辑扩展。实际上，单一公共代码库支持这两种方式的操作。下图用来显示单分区处理模型（如您在前两幅图中所看到的）与多分区处理模型之间存在的相似性与差异。

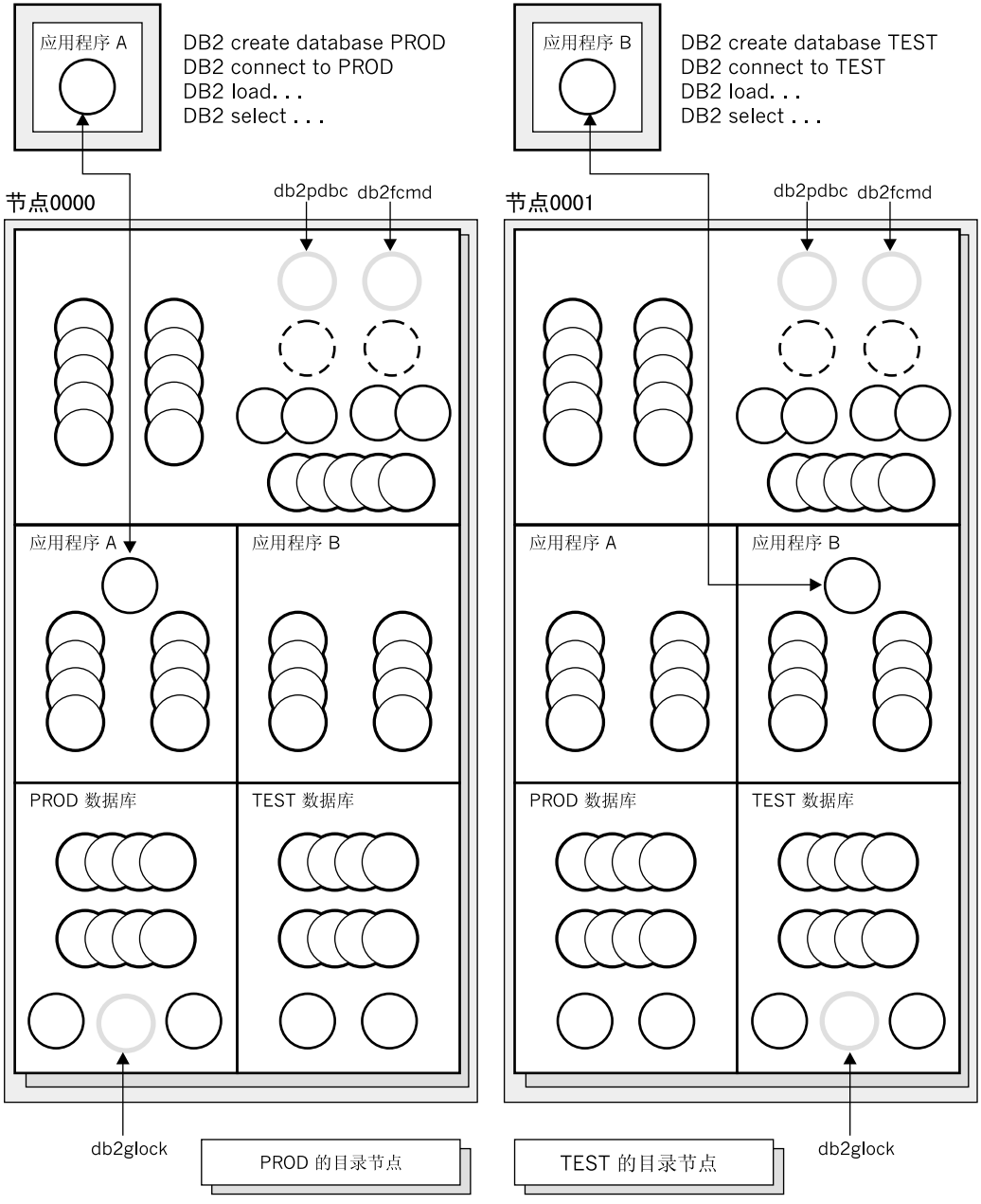


图 9. 进程模型与多分区

在单分区处理模型与多分区处理模型之间，大多数引擎可调度单元 (EDU) 是相同的。

在多分区（或节点）环境中，必须将其中一个分区看作是目录节点。目录存放与数据库中的对象相关的所有信息。

如上图所示，因为“应用程序 A”在 Node0000 上创建 PROD 数据库，所以在此节点上创建 PROD 数据库的目录。同样，因为“应用程序 B”在 Node0001 上创建 TEST 数据库，所以在此节点上创建 TEST 数据库的目录。因为您将总是希望在系统环境中的节点之间对与每个数据库的目录相关联的附加活动进行平衡，所以您可能会希望在不同的节点上创建数据库。

在多分区数据库环境中的每个节点上，可以找到与实例相关联的附加 EDU（db2pdbc 和 db2fcmd）。这些 EDU 是在数据库分区之间协调请求以及启用“快速通信管理器”(FCM) 所需的。

还有一个附加的 EDU (db2glock) 与数据库的目录节点相关联。此 EDU 在活动数据库所在的节点之间控制全局死锁状态。

每个来自应用程序的 CONNECT 都由数据库上的一个逻辑代理程序表示，并生成单一的协调代理程序。协调代理程序存在于与应用程序相连的分区上。此分区因而成为该应用程序的“协调程序节点”。协调程序节点也可以使用 *SET CLIENT CONNECT_NODE* 命令设置。协调程序节点将部分来自应用程序的数据库请求分至其他分区中的子代理程序；在将来自其他分区的所有结果在发送回给应用程序之前，在协调程序节点上对它们进行合并。

发出 CREATE DATABASE 命令的数据库分区称为该数据库的“目录节点”。目录表就存储在此数据库分区上。通常，所有用户表都在一组节点上进行分区。

注： 可以配置任意数目个分区来在同一机器上运行。这称为“多逻辑分区”或“多逻辑节点”配置。在带有非常大容量的主内存的大型对称多重处理器 (SMP) 机器上，这样的配置非常有用。在这种环境中，可以优化分区之间的通信，以使用共享内存和信号。

内存模型

因为内存对数据库中工作的完成方式有着显著的影响，所以它非常重要。您如何在数据库中的那些区域之间划分可用内存是您控制数据库运作的主要方法。您通过配置参数来在不同的堆之间控制这种内存划分。在本节中，我们将呈示关键的配置参数以及它们所控制的内存部分。有关此主题的详情，参见第201页的『DB2 如何使用内存』。

分区中的所有引擎可调度单元 (EDU) 都与“实例共享内存”相连。所有在数据库中执行工作的 EDU 都与该数据库的“数据库共享内存”相连。所有代替特定应用

程序工作的 EDU 都与该应用程序的“应用程序共享内存”区相连。仅当启用了分区内或分区间并行性时，才会分配这种类型的共享内存。最后，每个 EDU 都有它自己的专用内存。

“实例共享内存”（也称为“数据库管理程序共享内存”）是在启动数据库时分配的。从“实例共享内存”中，连接／分配所有其他内存。如果使用“快速通信管理器”(FCM)，则有些缓冲区取自此内存。FCM 用于特定数据库环境中数据库服务器之间及内部的内部通信，主要是信息。当第一个应用程序与一个数据库相连时，要分配“数据库共享内存区”、“应用程序共享内存区”和“代理程序专用内存区”。

“数据库共享内存”（也称为“数据库全局内存”）是在激活数据库或首次连接数据库时分配的。所有可能与该数据库相连的应用程序都使用此内存。数据库共享内存中包含许多不同的内存区，包括:

- 缓冲池
- 锁定列表
- 数据库堆－这包括日志缓冲区和目录高速缓存。
- 实用程序堆
- 程序包高速缓存

数据库管理程序配置参数 *numdb* 指定可以并行活动的本地数据库数。在分区数据库环境中，此参数限制数据库分区服务器上的活动数据库分区数。*numdb* 参数的值可能会影响已分配的总内存量。

“应用程序共享内存”（也称为“应用程序全局内存”）是在应用程序连接数据库时分配的。只有在分区数据库环境中，或当启用了数据库管理程序配置参数 *intra_parallel* 时，才发生此分配。此内存由为该应用程序工作的代理程序使用，以便在它们之间共享数据和协调活动。

数据库配置参数 *maxappls* 设置与一个数据库相连的应用程序数的上限。因为每个与数据库连接的应用程序都导致分配一些专用内存，所以允许大量并行应用程序可能将使用更多内存。

对于特定数据块，最大应用程序数也由数据库管理程序配置参数 *maxagents*（或者，用于并行环境的 *max_coordagents*）控制。 *maxagents* 参数设置分区中数据库管理程序代理程序总数的上限。这些数据库管理程序代理程序包括活动的协调代理程序、子代理程序、不活动代理程序和空闲代理程序。

当分配一个代理程序来为特定应用程序工作时，便分配该代理程序的“代理程序专用内存”。代理程序专用内存是为代理程序分配的，它包含仅供此特定代理程序使用的内存分配，如排序堆和应用程序堆。

有几种特殊类型的共享内存:

- 代理程序 / 本地应用程序共享内存。此内存用于代理程序与其客户机应用程序之间的 SQL 请求和响应通信。
- UDF / 代理程序共享内存。运行受防护 UDF 或“存储过程”的代理程序与此内存相连。它用作通信区。
- 扩充存储器。通常是一块非常大（大于 4 GB）的共享内存区，用作扩充缓冲池。代理程序 / 预取装器 / 页清除器并不是永久地与其相连，而是根据需要与其中个别段相连。

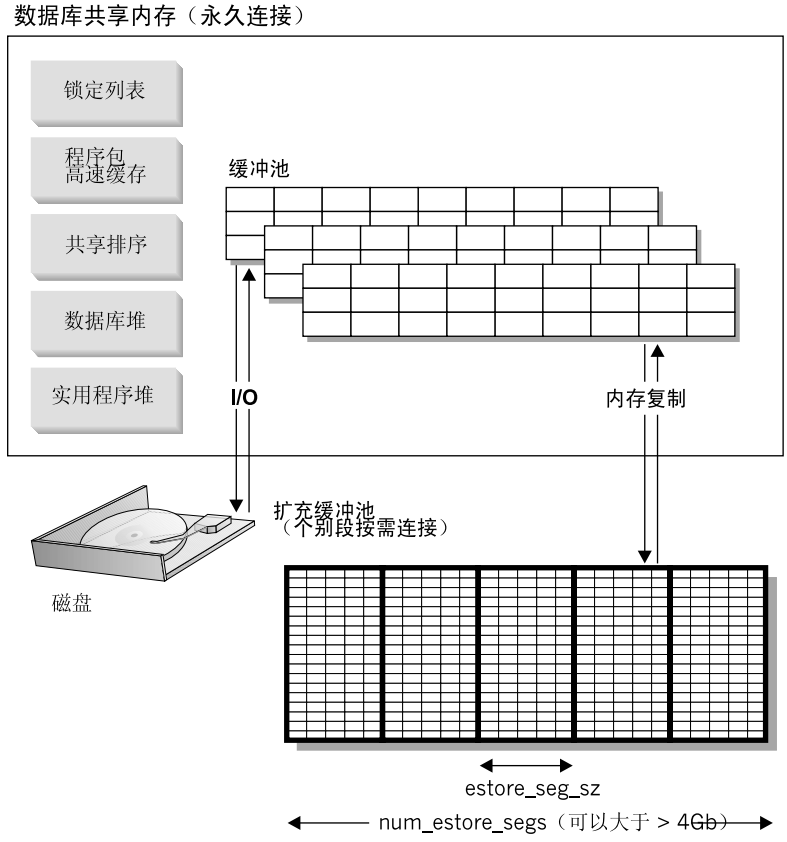


图 10. 缓冲池和扩充存储器

扩充存储器用作主缓冲池的扩充后备缓冲区。有关此主题的详情，参见第234页的『扩充内存』。它必须大于 4 GB，这是一种极好地利用带有大量主存储器的机器的方法。扩充高速缓存是用内存段来定义的。

当决定将一些可寻址实内存用作扩充高速缓存时，您应该知道，此内存不再能够用于机器上的其他用途，如 JFS 高速缓存或进程专用地址空间。向扩充高速缓存分配附加可寻址实内存会导致更高程度的系统分页。

下列数据库配置参数影响可用作扩充存储器的内存容量和大小:

- *num_estore_segs* 定义扩充内存段的数目。
- *estore_seg_sz* 定义每个扩充内存段的大小。

对每个表空间指定一个缓冲池。扩充高速缓存必须始终与一个或多个特定的缓冲池相关。扩充高速缓存的页大小必须与和它相关联的缓冲池的页大小相匹配。

有关扩充高速缓存的详情，参见第234页的『扩充内存』。

第2部分 调整应用程序性能

第3章 应用程序考虑事项

有许多因素可以影响应用程序的运行期性能。本章描述在设计和编码应用程序时应考虑的下列主题:

- 并行性
- 锁定
- 调整优化级别
- 限制结果集以改进性能
- 行分块
- 调整查询
- 复合 SQL
- 性能考虑事项和字符转换
- 存储过程
- 激活数据库
- 应用程序的并行处理。

还应参考 *Application Development Guide* 和 *CLI Guide and Reference*，以获取可影响应用程序性能的其他信息，例如:

- 使用嵌入式静态 SQL 编写程序
- 使用嵌入式动态 SQL 编写程序
- 使用“调用层接口”(CLI) 编写程序。

并行性

在多个用户存取和更改数据时，必须保持关系数据库中数据的完整性。*并行性*指的是可以同时由多个交互式用户或应用程序共享资源。数据库管理程序控制此存取以防止意外的后果，例如:

- *丢失更新*。两个应用程序 A 和 B，可能同时从数据库中读取相同的行并同时根据这些应用程序读取的数据为其中一列计算新的值。如果 A 用它的新值更新该行而 B 随后也更新该行，则由 A 执行的更新将丢失。
- *存取未落实的数据*。应用程序 A 可能更新数据库中的值，而应用程序 B 在该值被落实前可能会读取它。因此，如果 A 的值稍后未被落实，而是被收回，则由 B 执行的计算将基于未落实的（可能为无效的）数据。

- *不可重复读*。某些应用程序涉及到以下一系列事件: 应用程序 A 从数据库中读取一行，然后继续处理其他 SQL 请求。与此同时，应用程序 B 修改或删除该行并落实更改。稍后，如果应用程序 A 试图再次读取原始行，它将接收到已修改的行或发现该原始行已被删除。
- *幻象读现象*。在下列情况下将出现幻象读现象:
  1. 您的应用程序执行一个查询，该查询根据某些搜索标准读取一组行。
  2. 另一个应用程序插入新数据或更新现有的数据以满足您的应用程序查询。
  3. 您的应用程序从步骤 1 开始重复查询（在同一工作单元中）。

  当重复查询（步骤 3）时，某些附加的（“幻象”）行作为结果行的一部分返回，但在第一次执行该查询（步骤 1）时不会返回这些行。

*隔离级别*确定存取数据时如何锁定数据或使数据不受其他进程影响。该隔离级别将在工作单元运行期生效。在执行 OPEN CURSOR 的工作单元期间，使用由 WITH HOLD 子句说明的游标的应用程序将保持选定的隔离级别。（有关详情，参考 *SQL Reference* 手册）。查看第42页的『指定隔离级别』以获取关于如何指定隔离级别的信息。

DB2 支持下列隔离级别:
- 可重复读
- 读稳定性
- 游标稳定性
- 未落实的读。

（注意，某些 DRDA 数据库服务器支持*非落实*隔离级别。对于其他数据库，其特性和未落实的读隔离级别一样。参考 *SQL Reference* 以获取关于此隔离级别的信息。）

另见:
- 第41页的『选择隔离级别』
- 第42页的『指定隔离级别』。

您可能正在联合体数据库系统中工作，该系统支持应用程序和用户提交 SQL 语句，这些 SQL 语句在单个语句中引用两个或多个数据库管理系统或数据库。DB2 联合体系统为数据库对象提供位置透明性。例如，如果移动了有关表和视图的信息，则可更新对该信息（称为别名）的引用，而无需更改请求该信息的应用程序。应用程序存取别名时，DB2 依靠数据源数据库管理程序的并发控制协议来保证隔离级别。（数据源由 DBMS 和数据组成。）DB2 会尝试将数据源的请求隔离级别与等效的逻辑级别匹配；然而，结果可能因数据源的容量而不同。参考 *Application Development Guide* 手册以获取关于编写存取别名的应用程序的信息。

每个隔离级别的详细说明按它们对性能的影响程度的降序排列，但按您存取和更新数据时需要加以关心的程度的升序排列。

可重复读

可重复读 (RR) 会锁定应用程序在工作单元中引用的所有行。利用可重复读，在打开游标的相同工作单元内一个应用程序发出一个 SELECT 语句两次，每次都能获得相同的结果。利用可重复读，不可能出现丢失更新、存取未落实的数据和幻象行的情况。

在该工作单元完成之前，可重复读应用程序可以尽可能多次地检索和操作这些行。然而，在该工作单元完成之前其他应用程序均不能更新、删除或插入可能会影响结果表的行。可重复读的应用程序不能查看其他应用程序的未落实更改。

利用可重复读，将会锁定引用的每一行，而不仅仅是检索的那些行。执行了适当的锁定，以至其他应用程序不能插入或更新行，该行可能要添加到查询所引用的行的列表中（如果重新执行查询的话）。这将防止出现幻象行。这意味着，如果您扫描 10 000 行并对它们进行过滤，尽管只有 10 行满足条件，但仍会锁定全部的 10 000 行。

注: 可重复读隔离级别确保在应用程序*看到*所有返回的数据之前数据保持不变，即使使用了临时表或行分块。

由于可重复读可能获得和持有大量锁定，因此这些锁定可能超出可作为 *locklist* 和 *maxlocks* 配置参数的有效结果的锁定数。（查看第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』和第297页的『锁定列表的最大存储器 (locklist)』。）为了避免锁定逐步升级，优化器在认为很可能会发生锁定逐步升级的时候，可能选择立即获得单个表级别锁定用于索引扫描。（查看第48页的『锁定逐步升级』以获取有关锁定逐步升级的讨论。）这就象数据库管理程序代表您发出了一个 LOCK TABLE 语句一样。如果不想获得表级别锁定，确保有足够的锁定可用于该事务或使用读稳定性隔离级别。

读稳定性

读稳定性 (RS) 只锁定应用程序在工作单元中检索的那些行。它确保在工作单元完成之前，任何在工作单元运行期间的行读取限定不被其它应用程序进程更改，且不会读取另一应用程序进程更改的任何行，直至该进程落实更改。也就是说，不可能出现“不可重复读”情形。

与可重复读不同，在读稳定性级别，如果您的应用程序多次发出相同的查询，则有可能看到附加的*幻象*行（*幻象读现象*）。再次引用扫描 10 000 行的示例时，读稳

定性只锁定限定的行。这样，在读稳定性级别，只检索 10 行，且只对那十行持有锁定。将它与可重复读对比，在此示例中，可重复读会在所有的 10 000 行上持有锁定。持有的锁定可以是共享、下次共享、更新或互斥锁定。（有关锁定属性的详情，参见第44页的『锁定的属性』。）

注: 读稳定性隔离级别确保所有返回的数据在被应用程序*看到*之前保持不变，即使使用了临时表或行分块。

读稳定性隔离级别的其中一个目标是提供高度的并行性以及数据的稳定视图。为了有助于达到此目标，优化器确保在发生锁定逐步升级前不获取表级锁定。（查看第48页的『锁定逐步升级』以获取锁定逐步升级的详情）。

读稳定性隔离级别最适用于包括下列所有特性的应用程序:

- 在并行环境下运行
- 需要限定某些行在工作单元运行期间保持稳定
- 在工作单元中不会多次发出相同的查询，或者在同一工作单元中发出多次查询时并不要求该查询获得相同的回答。

游标稳定性

游标稳定性 (CS) 当在行上定位游标时会锁定任何由应用程序的事务所存取的行。此锁定在读取下一行或终止事务之前有效。然而，如果更改了某一行上的任何数据，则在对数据库落实更改之前必须持有该锁定。

对于具有游标稳定性的应用程序已检索的行，当该行上有任何可更新的游标时，任何其他应用程序都不能更新或删除该。游标稳定性应用程序不能查看其他应用程序的未落实更改。

再次引用扫描 10 000 行的示例，如果使用游标稳定性，将只锁定当前游标位置以下的行。当移离该行时，也就除去了该锁定（除非更新该行）。

使用游标稳定性，可能会出现不可重复读和幻象读现象。游标稳定性是缺省隔离级别，且应在需要最大并行性，但只看到其他应用程序中的已落实行的情况下才使用。

未落实的读

未落实的读 (UR) 允许应用程序存取其他事务的未落实的更改。除非其他应用程序尝试卸下或改变该表，否则该应用程序也不会锁定正读取的行而使其他应用程序不能访问该行。对于只读和可更新的游标，未落实的读的工作方式有所不同。

只读游标可存取大多数其他事务的未落实的更改。然而，当该事务正在处理时，由其他事务创建或卸下的表、视图和索引不能使用。其他事务的任何其他更改在落实或回滚前都可被读取。

注： 在未落实的读隔离级别下，可更新操作的游标将按隔离级别是游标稳定性的方式。

再次引用扫描 10 000 行的示例，如果使用未落实的读，则不需要对任何行锁定。

使用未落实的读，可能出现不可重复读行为和幻象读现象。

未落实的读隔离级别最常用于只读表上的查询，或者若只执行查询语句且不关心是否可从其他应用程序中看到未落实的数据时也最常用。

选择隔离级别

表1根据 *Application Development Guide* 手册中描述的不期望的结果来总结了几种不同的隔离级别。

表 1. 隔离级别摘要

隔离级别	存取未落实的数据	不可重复读	幻象读现象
可重复读 (RR)	不可能	不可能	不可能
读稳定性 (RS)	不可能	不可能	可能
游标稳定性 (CS)	不可能	可能	可能
未落实的读 (UR)	可能	可能	可能

表2提供了简单的启发式方法，可能有助于为应用程序选择初始隔离级别。首先考虑下表列出的方法，并参考先前对各级因素的讨论，可能会找到更适合要求的另一个值。

表 2. 选择隔离级别的准则

应用程序类型	需要高数据稳定性	不需要高数据稳定性
读写事务	RS	CS
只读事务	RR 或 RS	UR

为一个应用程序选择适当的隔离级别，这对于避免该应用程序产生无法容忍的现象很重要。因为获取和释放锁定所需的 CPU 和内存资源随隔离级别的不同而不同，所以此隔离级别不但影响应用程序之间的隔离程度，而且还影响个别应用程序的性能特征。潜在的死锁情况随隔离级别的不同而不同。

指定隔离级别

隔离级别是在预编译或应用程序与数据库联编时指定。对于用受支持的编译语言编写的应用程序，使用命令行处理器 PREP 或 BIND 命令的 ISOLATION 选项。也可使用 PREP 或 BIND API 来指定隔离级别。如果未指定隔离级别，则使用缺省值 - 游标稳定性。

如果在预编译时创建了联编文件，则隔离级别存储在联编文件中。如果在联编时未指定隔离级别，则缺省值是预编译期间使用的隔离级别。

可通过执行下列查询来确定程序包的隔离级别:

```
SELECT ISOLATION FROM SYSCAT.PACKAGES
  WHERE PKGNAME = 'XXXXXXXX'
  AND PKGSCHEMA = 'YYYYYYYY'
```

其中 *XXXXXXXX* 是程序包名，而 *YYYYYYYY* 是程序包的模式名。这两种名称都必须大写。

当创建一个数据库时，会有多个联编文件与该数据库联编（在支持 REXX 的那些服务器上），这些文件用于支持使用 REXX SQL 的不同隔离级别。当创建一个数据库时，其他命令行处理器程序包也会与该数据库联编。参考 *Application Development Guide* 以获取关于这些联编文件的详情。

REXX 和命令行处理器使用缺省隔离级别 - 游标稳定性与数据库连接。更改为不同的隔离级别时并不更改连接状态。必须在 CONNECTABLE AND UNCONNECTED 状态下或 IMPLICITLY CONNECTABLE 状态下执行它。（查看 *SQL Reference* 中的 CONNECT TO 语句以获取关于这些连接状态的细节）。发出此命令时不能与数据库连接。

正在使用的隔离级别可由 REXX 应用程序通过检查 SQLISL REXX 变量的值而得到。每次执行 CHANGE SQLISL 命令时更新该值。

可使用 DB2_RR_TO_RS 简要表注册表变量，来阻止对用户表的“可重复读”(RR) 隔离级别存取。在 RR 隔离语义不是必需的环境中可使用 `db2set` 将此注册表值设置为 『YES』。必须停止然后启动该数据库，此设置才能生效。执行 `db2start` 之后，此更改将影响整个实例。一旦设置后，若接收到使用 RR 存取用户表的请求，则在内部修改该请求以改用“读稳定性”(RS) 隔离级别。当执行此操作时，不会给出警告。

如果正在使用命令行处理器，可以使用 CHANGE ISOLATION LEVEL 命令更改隔离级别。有关详情，参考 *Command Reference* 手册。

对于 DB2 调用层接口 (DB2 CLI)，可以将隔离级别更改为 DB2 CLI 配置的一部分。在运行期的 CLI 中，SQLSetConnectAttr 函数将与 SQL_ATTR_TXN_ISOLATION 属性配合使用。这将设置 *ConnectionHandle* 引用的当前连接的事务隔离级别。在 db2cli.ini 文件中，还可以使用 TXNISOLATION 关键字。

注: JDBC 和 SQLJ 是通过 DB2 上的 CLI 实现的，这意味着 db2cli.ini 设置会影响使用 JDBC 和 SQLJ 写入和运行的内容。有关详情，参考 *CLI Guide and Reference* 手册。

当在运行期使用 JDBC 或 SQLJ 时，可在 java.sql 接口连接中使用 setTransactionIsolation 方法来建立隔离级别。有关详情，参考 *Application Development Guide* 的 『Programming in Java』 一章。

使用 SQLJ 时，如果运行 db2profc SQLJ 优化器，将创建一个程序包。可以指定此程序包的最终选项，以包括要使用的隔离级别。有关详情，参考 *Application Development Guide* 的 『Programming in Java』 一章。

另外，许多商业性应用程序也提供方法供您选择隔离级别。有关详情，参考 *CLI Guide and Reference* 手册。

已说明临时表和并行性

因为已说明临时表只可用于说明它们的应用程序，所以已说明临时表没有并行性问题。这种类型的表从应用程序说明它起开始存在，直到应用程序完成或断开连接为止。

锁定

数据库管理程序提供并行性控制并借助于锁定防止不受控制的存取。*锁定*是将数据库管理程序资源与应用程序关联以控制其他应用程序存取同一资源的方式的一种方法。与资源相关联的应用程序被认为持有或拥有该锁定。

数据库管理程序用锁定来禁止应用程序存取其他应用程序写入的未落实数据（除非使用未落实的读隔离级别）。此原则目的是保护数据的完整性（即，数据的一致性和安全性）。锁定也可以禁止行更新（例如对于可重复读应用程序）。

要满足数据完整性，数据库管理程序应在数据库管理程序控制下隐式获取锁定。除了未落实的读隔离级别之外，应用程序从不需要显式请求锁定来确保未落实数据对其他进程不可见。

由于锁定的基本原理，在大多数情况下不需要采取措施控制锁定。同样，应用程序根据某些通用参数来获取锁定。对本地情况的了解有助于通过更改那些参数来更好地利用系统资源。为了帮助您，下面分别讨论下列关于锁定的题目:

- 锁定的属性
- 锁定与应用程序性能
- 影响锁定的因素
- LOCK TABLE 语句
- CLOSE CURSOR WITH RELEASE
- 锁定考虑事项摘要。

锁定的属性

数据库管理程序锁定具有下列基本属性:

方式 允许锁定拥有者使用的存取类型以及允许锁定对象的并行用户使用的存取类型。有时称之为锁定的*状态*。

对象 正被锁定的资源。唯一可显式锁定的对象类型是表。数据库管理程序也可锁定其他类型的资源，例如行、表和表空间。正被锁定的对象表示锁定的*粒度*。

持续时间
持有锁定的时间长度。在第37页的『并行性』中讨论的隔离级别会影响锁定持续时间。

在下表中，按照对资源的控制性递增的顺序显示锁定方式及其效果:

表 3. 锁定方式摘要

锁定方式	适用的对象类型	说明
IN（无意图）	表空间、表	锁定拥有者可以读取表中的任何数据，包括未落实的数据，但不能更新它的任何数据。锁定拥有者未获取行锁定。其他并行应用程序可以读取或更新表。
IS（有意共享）	表空间、表	锁定拥有者可读取已锁定的表中的数据，但不能更新此数据。当应用程序持有 IS 表锁定时，应用程序获取每个行读取的 S 或 NS 锁定。在任一种情况下，其他应用程序均可以读取或更新表。
NS（下一个关键字共享）	行	锁定拥有者和所有并行应用程序可读取但不能更新锁定的行。对一个表的行获取此锁定，以代替 S 锁定。
S（共享）	行、表	锁定拥有者和所有并行应用程序可读取但不能更新锁定的数据。表的个别行可以是 S 锁定的。如果一个表是 S 锁定的，则行锁定不是必需的。

表 3. 锁定方式摘要 (续)

锁定方式	适用的对象类型	说明
IX（有意互斥）	表空间、表	锁定拥有者和并行应用程序可以读取和更新表中的数据。当锁定拥有者读取数据时，它在每一个行读取上获取 S、NS、X 或 U 锁定。还在锁定拥有者更新的每一行上获取 X 锁定。其他并行应用程序既可以读取也可以更新表。
SIX（在有意互斥下共享）	表	锁定拥有者可以读取和更新表中的数据。锁定拥有者获取对它所更新的行的 X 锁定，但不获取对它所读取的行的锁定。其他并行应用程序可以读取该表。
U（更新）	行、表	锁定拥有者可以更新锁定的行或表中的数据。锁定拥有者在它更新行之前，先在那些行上获取 X 锁定。其他工作单元可以读取锁定的行或表中的数据，但不能尝试更新它。
NX（下一个关键字互斥）	行	锁定拥有者可以读取但不可更新已锁定的行。除了与 NS 锁定兼容这一点外，此方式类似于 X 锁定。
NW（下一个关键字弱互斥）	行	当将行插入非目录表的索引时，即获取对下一行的此种锁定。锁定拥有者可以读取但不可更新已锁定的行。除了与 W 和 NS 锁定兼容这一点外，此方式类似于 X 和 NX 锁定。
X（互斥）	行、表	锁定拥有者既可以读取也可以更新锁定的行或表中的数据。可以“互斥”锁定表，即，不在那些表中的行上获取行锁定。只有未落实的读应用程序可以存取已锁定的表。
W（弱互斥）	行	当将行插入非目录表时，即获取对该行的此种锁定。锁定拥有者可更改已锁定的行。除了与 NW 锁定兼容这一点外，此锁定类似于 X 锁定。只有未落实的读应用程序可以存取已锁定的行。
Z（超互斥）	表空间、表	对表的此种锁定是在一定条件下获取的，例如，当改变或卸下表、创建或卸下表的索引或重组表时。没有其他的并行应用程序可以读取或更新该表。

注: 只有表和表空间将获得“意图”锁定方式。也就是说，不会获得对行的意图锁定。

锁定与应用程序性能

应用程序设计人员需要了解有关锁定的使用以及锁定对应用程序性能的影响的几个相关因素。这些因素包括:

- 并行性与粒度
- 锁定兼容性
- 锁定转换
- 锁定逐步升级
- 锁定等待和超时

- 死锁。

并行性与粒度

一个应用程序持有的锁定可以防止被另一个应用程序存取。因此，要获得最大的并行性，行级别锁定比表锁定好。但锁定需要存储器和处理时间来管理。因此，要最小化存储器和处理时间，单个表锁定比多行锁定更好。

可通过 ALTER TABLE 语句的 LOCKSIZE 子句在行级别或表级别定义锁定的大小（粒度）。缺省情况下，使用行锁定。对于由 ALTER TABLE 定义的永久表锁定，仅使用 S 和 X 表锁定。因为应用程序不需要获取和释放许多行锁定，因此提高了性能。在下列情况下，您可选择使用 ALTER TABLE 语句来获得永久表锁定，而不要选择使用 LOCK TABLE 语句来获得单个事务表锁定:

- 您的表是只读的，您将始终需要 S 锁定。当允许其他用户获得该表的 S 锁定时，表级别锁定的性能将改善。
- 在有限的时间段内，该表将由单个用户存取以进行维护，该用户需要 X 锁定。通过 ALTER TABLE 在表上更改表级别锁定，将为该用户提供表级别的 X 锁定。一旦用户完成操作，就可使用 ALTER TABLE 将表返回至行级别锁定。

使用 ALTER TABLE 语句不会阻止正常锁定逐步升级。

另外，还需注意，使用 ALTER TABLE 将锁定设定为表级别是一种全局处理方法，它会影响到存取该表的所有应用程序和用户。单个应用程序可选择的另一种方法是使用 LOCK TABLE 语句。这允许您在应用程序级别而不是在数据库级别转至表锁定（如上面的第二个强调符所述）。

锁定兼容性

表4指示在一给定状态下，如果另一个进程持有或正在请求对同一个资源的锁定，是否准许锁定请求。**否**指示请求者必须等待，直到所有不兼容的锁定被其他进程释放为止。注意，当等待锁定时，可能出现超时。**是**指示准许该锁定（除非别的人正在等待该资源）。

表 4. 锁定类型兼容性

	占有资源的状态												
请求的状态	**无**	**IN**	**IS**	**NS**	**S**	**IX**	**SIX**	**U**	**NX**	**X**	**Z**	**NW**	**W**
无	是	是	是	是	是	是	是	是	是	是	是	是	是
IN	是	是	是	是	是	是	是	是	是	是	否	是	是
IS	是	是	是	是	是	是	是	是	否	否	否	否	否
NS	是	是	是	是	是	否	否	是	是	否	否	是	否
S	是	是	是	是	是	否	否	是	否	否	否	否	否
IX	是	是	是	否	否	是	否	否	否	否	否	否	否
SIX	是	是	是	否	否	否	否	否	否	否	否	否	否
U	是	是	是	是	是	否	否	否	否	否	否	否	否
NX	是	是	否	是	否	否	否	否	否	否	否	否	否
X	是	是	否	否	否	否	否	否	否	否	否	否	否
Z	是	否	否	否	否	否	否	否	否	否	否	否	否
NW	是	是	否	是	否	否	否	否	否	否	否	否	是
W	是	是	否	否	否	否	否	否	否	否	否	是	否

注:

I 意图
N 无
NS 下一个关键字共享
S 共享
NX 下一个关键字互斥
X 独占
U 更新
Z 超互斥
NW 下一个关键字弱互斥
W 弱互斥

有关这些锁定类型的详细信息，可参考第44页的『锁定的属性』中的讨论。

注:

- 是 - 立即**准许**所请求的锁定
- 否 - **等待**释放持有的锁定或出现超时

假定应用程序 A 持有对某个表的锁定，而应用程序 B 也想要存取该表。数据库管理程序代表应用程序 B，请求某特定方式的锁定。如果由 A 持有的锁定的方式许可 B 请求的锁定，则这两个锁定（或方式）被称为是兼容的。

如果应用程序 B 所请求的锁定方式与应用程序 A 所持有的锁定不兼容，那么应用程序 B 不能继续运行。相反，它要等到应用程序 A 释放其锁定，并且*所有*其他现存不兼容锁定被释放为止。

锁定转换

当一个进程存取它已锁定的数据对象，且存取方式需要比已持有的锁定更严格的锁定时，将进行锁定转换。一个进程在任一时间对数据对象只能持有一个锁定，尽管它可以（通过查询间接地）对同一数据对象多次请求锁定。更改已持有的锁定方式的操作称为*转换*。

行的转换情况很简单：例如，如果需要 X 而持有 S 或 U，则进行转换。

表和行有着不同的锁定方式。然而，在锁定转换这一点上，IX（意图互斥）和 S（共享）锁定是特殊情况。S 和 IX 锁定的严格程度相当，所以如果持有了一个而请求另一个，则结果转换为 SIX（带意图互斥的共享）锁定。所有其他的转换都导致所请求的锁定方式变成所持有的锁定方式（如果所请求的方式不那么严格的话）。

要更新一行的查询也可产生双向转换。假定已通过索引存取读取了该行并且将该行锁定为 S。包含该行的表会具有覆盖意图锁定。假定它是 IS 而不是 IX。这样，如果随后更改了该行，则表锁定转换为 IX，而行锁定则转换为 X。

值得注意的是，锁定通常是在执行查询期间隐式生效。了解不同的查询以及表和索引的组合获得的锁定种类，有助于您设计和调节您的应用程序。有关此主题的详情，参见第52页的『影响锁定的因素』。

锁定逐步升级

锁定逐步升级是减少持有的锁定数的一种内部机制。升级是从多行锁定（单个表中的）至单个表锁定。

如果当前持有太多锁定（任何类型），则将进行锁定逐步升级。

如果一个特定的数据库代理程序超过为它分配的锁定列表，可对该代理程序进行锁定逐步升级（查看第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』）。

这类逐步升级在内部处理；唯一在外部可检测的结果可能是对一个或多个表并行存取的减少。正常情况下，在一个适当配置的数据库中，不频繁进行锁定逐步升级。

例如，当应用程序设计者对大型表使用索引以提高性能与并行性，然而应用程序却存取表中的大部分记录时，可能会发生锁定逐步升级。数据库管理程序无法预计（在此情况下）该表的这么多部分要被锁定，并且分别锁定每个记录而不是只以 S 或 X 方式锁定该表。作为解决方法，数据库设计者在与应用程序设计者磋商之后，可以建议应用程序设计者对此事务使用 LOCK TABLE 语句。

有时，（内部地）接受逐步升级请求的进程持有很少或不持有对任何表的记录锁定。此逐步升级的原因是一个进程（或多个进程）可持有多个锁定（此数量低于数据库配置参数中每个进程的锁定数），但仍不足以触发逐步升级请求。该进程可能无法请求另一个锁定或再次存取该数据库，除非结束事务。这样另一个进程可以请求触发逐步升级请求的一个或多个锁定。

如果锁定逐步升级将并行性减少到不可接受的程度，可以采取如下措施:

- 有关逐步升级的信息，参见 *db2diag.log* 的内容。记录了有关要逐步升级的每个表的信息。记录的信息类型包括:
  - 当前持有的锁定数目。
  - 在锁定逐步升级完成之前所需的锁定数目。
  - 逐步升级的每个表的表标识符信息和表名。
  - 当前持有的非表锁定的数目。
  - 作为逐步升级的一部分，获取新表级别的锁定。一般情况下，它将是 『S』 或共享锁定，或者是 『X』 或互斥锁定。
  - 获取新表锁定级别结果的内部返回码。

  也可能记录当前的动态 SQL 语句。如果是这样，并且**如果** DIAGLEVEL 数据库管理程序配置参数是 4，则在对任何表锁定进行逐步升级前将记录当前的 SQL 语句。如果锁定逐步升级失败，并且**如果** DIAGLEVEL 大于或等于 2，则记录的信息将包括进行逐步升级失败的表和当前的 SQL 语句（如果可找到该语句并且它不是先前编写的）。

  参照此信息，您将能根据下面提到的其他几点来执行适当的操作。

  要开始记录这类信息，应将数据库管理程序配置参数 DIAGLEVEL 设置为缺省值 3 或 4。
- 通过增加数据库配置文件中 *maxlocks* 和／或 *locklist* 参数的值来增加允许的锁定数。（查看第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』和第297页的『锁定列表的最大存储器 (locklist)』。）如果其他进程对该表的并行存取很重要，可以考虑选择此措施。但是，获得记录级别锁定的额外开销可能使其他进程延迟，这些延迟比并行存取表节省下来的延迟更多。（当在分区数据库中更改这些参数时，确保更新了所有分区中的这些参数）。

- 找到并调整违规进程，这些进程可能是或可能不是逐步升级或回滚的进程，同时显式发出 LOCK TABLE 语句。
- 更改隔离度。注意，这可能导致降低并行性。
- 增加落实的频率。这会减少在给定时间存在的锁定数。有关隔离级别和并行性的详情，参见第37页的『并行性』。

锁定等待和超时

在异常情况下，如果没有锁定超时检测，应用程序可能必须无限期地等待释放锁定。例如，如果一个事务正等待另一个用户的应用程序持有的锁定，而该用户已离开工作站，而没有执行某个交互操作来允许应用程序落实事务以释放锁定，则可能出现这种情况。显然，这会使应用程序的性能较差。要避免这种情况*影响*您的程序，可以使用 *locktimeout* 配置参数来设置任何应用程序等待获得锁定的最长时间。（查看第323页的『锁定超时 (locktimeout)』。）

使用此参数有助于避免全局死锁，特别是在分布式工作单元 (DUOW) 应用程序中。如果锁定超时，也就是说，如果锁定请求处于暂挂的时间大于 *locktimeout* 值，那么应用程序将接收到一个错误信息并将事务回滚。例如，如果*程序 1* 试图获取一个已由*程序 2* 持有的锁定，那么*程序 1* 在超时到期时返回 SQLCODE -911 (SQLSTATE 40001)，原因码为 68。

如果数据库管理程序配置参数 *diaglevel* 设置为 4，且锁定请求超时，则可以在 db2diag.log 找到更多信息。在那里找到的信息包括该对象、锁定方式以及对该对象挂起锁定的应用程序。还可能会找到当前的动态 SQL 语句或静态程序包名。

死锁

在数据库管理程序中，使用数据库的多个进程争用锁定可能导致死锁。例如，“进程 1”以 X（互斥）方式锁定表 A 而“进程 2”以 X 方式锁定表 B；然后如果“进程 1”试图以 X 方式锁定表 B 而“进程 2”试图以 X 方式锁定表 A，那么这两个进程将处于死锁状态。在死锁中，两个进程均被暂停，直到它们的第二个锁定请求被准许为止，而在其中一个进程执行落实或回滚之前不会准许这两个请求。此状态一直保持下去，直到一个外部代理程序激活了一个进程，并强制它执行回滚为止。

锁定系统中的死锁在数据库管理程序中由一个称为死锁检测器的异步系统后台进程来处理。死锁检测器按 *dlchktime* 配置参数的定义定期成为活动的（参见第321页的『检查死锁的时间间隔 (dlchktime)』）。当死锁检测器活动时，它检查死锁的锁定系统。如果已经将数据库分区，则每个分区将*锁定图*发送至带有系统目录视图的数据库分区，这就是进行全局死锁检测的地方。

如果发现死锁，则死锁检测器将选择回滚一个死锁的进程。所选择的进程被唤醒，并返回到调用的应用程序，其 SQLCODE 为 -911 (SQLSTATE 40001)，原因码为 2。数据库管理程序自动回滚选择的进程。当完成回滚时，释放属于受害进程的锁定，而该死锁涉及的其他进程最终可以继续进行。

有必要为死锁检测器选择适当的时间间隔以确保良好的性能。过短的时间间隔会导致不必要的额外开销，而太长的时间间隔会使死锁将进程延迟一段不可接受的时间。例如，唤醒时间间隔设置为 30 分钟可能会让一个死锁存在近 30 分钟。应用程序设计员必须在解决死锁时的可能延迟与检测它们的额外开销两者之间进行平衡。

在分区数据库中，该间隔在所有的分区中应是相同的（对所有的分区，必须将 *dlchktime* 配置参数更新为相同的值）。如果在目录节点中的该值小于它在其他分区中的值，那么可能检测到幻象死锁。如果在目录节点中的该值大于其在其他分区中的值，那么在检测到死锁前可能好象经过了两个以上的时间间隔。如果在分区数据库中检测到大量的死锁，应该增加 *dlchktime* 参数的值以解决锁定等待和通信等待。

当带有多个存取数据库的独立进程的应用程序被以一种很可能产生死锁的方式构造时，可能出现另一个问题。例如，在一个应用程序中的数个进程存取同一个表，对该表进行读取及写入操作。如果这些进程首先执行只读 SQL 查询，然后再对该表执行 SQL 更新，那么由于各个进程间对同一数据潜在的争用使得死锁出现的机会增加。例如，如果两个进程读该表，然后更新该表，那么它们将形成状态“进程 A 试图得到对行的 X 锁定，而进程 B 对该行具有 S 锁定”，反之亦然。结果可能就是死锁。要避免这些死锁，为执行修改而存取数据的应用程序应在执行选择时使用 FOR UPDATE OF 子句。此子句确保当进程 A 试图读取该数据时进行 U 锁定。

注： 可能要考虑定义一个监控程序来记录何时发生死锁。使用 *SQL Reference* 中所描述的 CREATE EVENT 语句来创建该监控程序。

在联合体系统环境中，当应用程序存取别名时，可能会由于数据源处发生死锁而无法获取应用程序请求的数据。当发生这种情况时，DB2 依靠数据源的死锁处理设施解开锁定。一旦多个数据源之间发生死锁，则 DB2 依靠数据源超时工具解开死锁。

如果数据库管理程序配置参数 *diaglevel* 设置为 4，且锁定请求因死锁而失败，则可以在 db2diag.log 找到更多信息。在那里找到的信息包括该对象、锁定方式以及对该对象挂起锁定的应用程序。还可能会找到当前的动态 SQL 语句或静态程序包名。

影响锁定的因素

数据库管理程序锁定的方式和粒度由各种因素综合确定：应用程序执行的处理的类型、它存取数据的方式以及可指定的一些参数。

应用程序处理

为了确定锁定属性，可以将处理分为四种类型:

只读 此类型包括所有这样的选择语句，它们本身是只读的（查看 *SQL Reference* 以获取关于游标的信息），并具有一个显式的 FOR READ ONLY 子句，或者是二义的（但是 SQL 编译程序认为是只读的，因为在 PREP 或 BIND 命令上指定了 BLOCKING 选项的值）。它只要求“共享”锁定（S 或 IS）。

更改意图

此类型包括所有带 FOR UPDATE 子句的选择语句，或由于存在二义性语句的解释 SQL 编译程序认为要更改选择语句。它使用“共享”和“更新”锁定（对于行，为 S、U、及 X；对于表，为 IX、U、X）。

更改 此类型包括 UPDATE、INSERT 及 DELETE，但不包括 UPDATE WHERE CURRENT OF 或 DELETE WHERE CURRENT OF。它要求“互斥”锁定（X 或 IX）。

受控游标

此类型包括 UPDATE WHERE CURRENT OF 和 DELETE WHERE CURRENT OF。它也要求“互斥”锁定（X 或 IX）。

根据子查询语句的结果对目标表进行插入、更新或删除的语句执行两种类型的处理。对在子选择语句中返回的表的锁定由只读处理规则来确定；而对目标表的锁定则由更改处理规则来确定。

存取路径

*存取路径*是优化器选择的一种方法，可通过引用特定的表来检索数据。（查看第134页的『数据存取概念和优化』。） 由优化器选择的存取路径对锁定方式会有显著的影响。例如，当使用索引扫描来查找特定行时，优化器很可能为该表选择行级别锁定 (IS)。此存取类型将用于从 EMPLOYEE 表中选择单个雇员的信息，该表有一个关于雇员编号 (EMPNO) 的索引，以及如下所示的语句:

```
SELECT *
  FROM EMPLOYEE
  WHERE EMPNO = '000310';
```

类似地，当不使用索引时，必须按查找选定行的顺序扫描整个表，并且可获取单个表级锁定 (S)。例如，此存取类型可用于选择所有的男性雇员，使用如下语句（如果 SEX 列上没有索引的话）：

```
SELECT *
  FROM EMPLOYEE
  WHERE SEX = 'M';
```

以下各表概述了对哪一类存取方案获取哪种锁定。有关列标题定义的信息，查看第52页的『应用程序处理』。另见第134页的『数据存取概念和优化』以获取关于存取方法定义的信息。注意，*受控游标*类型处理使用基础游标锁定方式，直到应用程序找到要更新或要删除的行为止。对于此种类型的处理，无论游标的锁定方式是什么，总会获得互斥锁定以执行更新或删除。

在下面的表中，如果只显示一个锁定方式，则它是表级锁定方式。如果显示两个锁定方式，那么第一个是表级锁定方式，第二个是行级锁定方式。

表 5. 表扫描的锁定方式

隔离级别	只读	更改意图	更改
存取方法: ***无谓词的表扫描***			
RR	S	U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
存取方法: ***带谓词的表扫描***			
RR	S	U	U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

表 6. 索引扫描的锁定方式

隔离级别	只读	更改意图	更改
存取方法: ***无谓词的索引扫描***			
RR	S	IX / U	X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
存取方法: ***索引扫描单个限定行***			
RR	IS / S	IX / U	IX / X
RS	IS / NS	IX / U	IX / X

表 6. 索引扫描的锁定方式 (续)

隔离级别	只读	更改意图	更改
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
存取方法: *只带开始和停止谓词的索引扫描*			
RR	IS / S	IX / S	IX / X
RS	IS / NS	IX / U	IX / X
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
存取方法: *无谓词的索引扫描*			
RR	IS / S	IX / S	IX / U
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

表7显示某些情况下的锁定方式，在这些情况中，将数据页的读取延迟以允许将行的列表:

- 使用多个索引进一步限定。有关详情，参见第139页的『多索引存取』。
- 排序以获得有效的预读取。有关详情，参见第216页的『了解列表预读取』。

被延迟的数据页存取意味着行存取分两步进行并导致更复杂的锁定情况。有两个主要类别依赖于隔离级别。由于可重复读隔离级别持有所有已获取的锁定直到事务结束，所以会保持在第一步中所获取的锁定而不必在第二步获取进一步的锁定。对于读稳定性和游标稳定性隔离级别，必须在第二步期间获取锁定。要使并行性达到最大，在第一步期间不获取锁定并依靠重新应用所有谓词来确保只返回限定行。

表 7. 延迟的数据页存取所用的索引扫描的锁定方式

隔离级别	只读	更改意图	更改
存取方法: *无谓词的 索引扫描*			
RR	IS / S	IX / S	X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
存取方法: *延迟的数据页存取，在无谓词的索引扫描之后*			
RR	IN	IX / S	X
RS	IS / NS	IX / U	IX / X

表 7. 延迟的数据页存取所用的索引扫描的锁定方式 (续)

隔离级别	只读	更改意图	更改
CS	IS / NS	IX / U	IX / X
UR	IN	IX / U	IX / X
存取方法: 带谓词的索引扫描			
RR	IS / S	IX / S	IX / S
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
存取方法: 只带开始和停止谓词的索引扫描			
RR	IS / S	IX / S	IX / X
RS	IN	IN	IN
CS	IN	IN	IN
UR	IN	IN	IN
存取方法: 延迟的数据页存取，在无谓词的索引扫描之后			
RR	IN	IX / S	IX / S
RS	IS / NS	IX / U	IX / U
CS	IS / NS	IX / U	IX / U
UR	IN	IX / U	IX / U

存取路径不由用户控制；它由优化器选择。

所使用的存取路径可以影响锁定的方式和粒度。例如，在使用可重复读 (RR) 隔离级别的应用程序中，使用无谓词的表扫描的 UPDATE 查询将对该表使用 X 锁定。如果通过索引来定位行，则数据库管理程序可能选择锁定该表的单个行。

已说明临时表和锁定

因为已说明临时表只可用于说明它们的应用程序，所以不锁定已说明临时表。这种类型的表从应用程序说明它起开始存在，直到应用程序完成或断开连接为止。

LOCK TABLE 语句

可以在应用程序中使用 LOCK TABLE 语句来取代获取初始锁定方式的规则。

该语句锁定整个表。但只锁定 LOCK TABLE 语句中指定的表。不锁定指定表的父表和从属表。必须确定是否需要锁定其他可存取的表以便在并行性与性能方面达到期望的结果。在工作单元被落实或回滚之前不释放锁定。

如果几个用户正常地共享一个表，可能会由于下列原因要锁定它:

以共享方式锁定表

可存取*在时间上一致的*数据；即，表在特定时间点的最新数据。如果表活动频繁，则确保整个表保持稳定的唯一方法是锁定它。例如，应用程序想要抽取表的快照。但是，在应用程序需要处理表的一些行期间，其他应用程序正在更新还没有处理的行。可重复读允许这样的操作，但您不希望这样。

另一种选择是，应用程序可以发出 LOCK TABLE IN SHARE MODE 语句: 无论是否检索了任何行，不能更改它们。然后可以按需要检索任意多行，应了解已经检索的行就在检索它们之前还没有被更改。

使用 LOCK TABLE IN SHARE MODE，其他用户可从表中检索数据，但不能更新、删除表中的行，或将行插入表中。

以互斥方式锁定表

可更新表的大部分。相对于更新表时锁定每一行，然后在落实所有更改时解锁行而言，阻止所有其他用户存取该表的开销少并且更有效。

使用 LOCK TABLE IN EXCLUSIVE MODE，所有其他用户都被锁定在外；没有其他应用程序可以存取该表，除非它们是未落实的读应用程序。

有关 LOCK TABLE 语句的详细信息，可参考 *SQL Reference* 手册。

如果不使用 LOCK TABLE 语句，那么可以使用带 LOCKSIZE 参数的 ALTER TABLE 语句。LOCKSIZE 参数允许选择 ROW 锁定或 TABLE 锁定。无论您选择哪种锁定，都将成为下一次存取该表时选择的锁定粒度。当新建一个表时，选择 ROW 锁定同选择缺省锁定大小无任何区别。使用表级锁定可提高查询性能，方法是限制需要获取和释放的锁定的数目。然而，由于所有锁定都是对整个表持有的，这样可减小并发性。任何一种选择都不会阻止正常的锁定逐步升级。有关 ALTER TABLE 语句的更多详细资料，参考 *SQL Reference* 手册。

CLOSE CURSOR WITH RELEASE

当用包括 WITH RELEASE 子句的 CLOSE CURSOR 语句关闭游标时，数据库管理程序将尝试释放所有为游标持有的读锁定（如果有的话）。读锁定是 IS、S 与 U 表锁定，以及 S、NS 与 U 行锁定。有关锁定方式的详情，参见第44页的『锁定的属性』。

WITH RELEASE 子句对正在 CS 或 UR 隔离级别下操作的游标不起作用。当对正在 RS 或 RR 隔离级别下操作的游标指定 WITH RELEASE 子句，它将结束那些隔离级别的一些保证。明确地说，RS 游标可能经历*不可重复读*现象，而 RR 游标可能经历*不可重复读*或*幻象读*现象。

如果一个原来是 RR 或 RS 的游标通过使用 WITH RELEASE 子句关闭后重新打开，那么将获得新的读锁定。

参见 第66页的『DECLARE CURSOR WITH HOLD 语句』以了解与 CLOSE CURSOR 语句的其他主子句的比较。

可在 CLI 应用程序中使用 DB2 CLI 连接属性 SQL_ATTR_ACCESS_MODE 来获得同样的结果。有关详情，参考 *CLI Guide and Reference* 的 SQLSetConnectAttr() 一节。

锁定考虑事项摘要

应记住下列关于锁定的要点:

- 小的工作单元（频繁的 COMMIT 语句）会使许多用户并行存取数据。当应用程序在逻辑上一致时，即更改的数据一致时，包括 COMMIT 语句。当发出 COMMIT 时，释放锁定（与声明了 WITH HOLD 的游标相关的表锁定除外）。
- 即使应用程序很少读取行，也会获得锁定，所以落实只读工作单元仍很重要。这是因为在只读应用程序中通过可重复读、读稳定性及游标稳定性隔离级别获得共享锁定。借助于可重复读和读稳定性，可持有所有锁定，直到发出 COMMIT，这可阻止其他进程更新锁定的数据，除非您使用 WITH RELEASE 子句关闭您的游标。此外，甚至在使用动态 SQL 的未落实的读应用程序中也要获得目录锁定。
- 数据库管理程序确保应用程序不检索未落实的数据（其他应用程序已经更新但尚未落实的行），除非正在使用未落实的读隔离级别。
- 可以通过发出 LOCK TABLE 语句锁定整个要保护的表:
  - 允许其他应用程序检索，但不更新、删除或插入行
  - 防止其他应用程序（那些带未落实的读隔离级别的应用程序除外）存取表中的行。
- 当用包括 WITH RELEASE 子句的 CLOSE CURSOR 语句关闭游标时，数据库管理程序将尝试释放所有为游标持有的读锁定（如果有的话）。
- 当更改影响分区数据库中的锁定的配置参数时，确保对数据库中所有的分区都作了更改。

调整优化级别

当编译 SQL 查询时，一些优化技术可用于确定对该查询最有效的存取方案。使用更多的优化技术将导致:

1. 运行期性能改善
2. 查询编译时间增加

3. 使用的系统资源增加。

由于此原因，应设置优化级别来限制应用于优化查询的技术的数量。如果您具有以下条件，这可能特别有用:

- 非常小型的数据库或非常简单的动态查询
- 在数据库服务器上编译时只有有限的可用内存
- 减少查询编译（例如，PREPARE）时间的要求。

可以从下面所述的任何查询优化级别中进行选择，级别 0 与级别 9 应该只用于特殊的情况。级别 5 是缺省值。级别 0、1 和 2 使用贪婪联合枚举算法；对于复杂的查询，与级别 3 和更高的级别相比，此算法考虑的替代方案少得多，并且编译时间也显著减少。级别 3 和更高的级别使用“动态规划”联合枚举算法；随着表的数量增加，与级别 0、1 和 2 相比，此算法考虑的替代方案多得多，并且编译时间也显著增加。

0 - 此级别指导优化器使用最少的优化来生成存取方案。例如:

- 优化器不考虑任何非均匀分布统计。
- 仅应用基本的查询重写规则（查看第125页的『由 SQL 编译程序重写查询』以获取关于查询重写的信息）。
- 发生贪婪联合枚举（查看第149页的『选择最优连接的搜索策略』）。
- 仅允许使用嵌套循环连接及索引扫描存取方法（参见第144页的『连接概念』和第134页的『索引扫描概念』）。
- 禁止列表预读取和索引 AND 运算，以使它们不会被用在生成的存取方法中。
- 不考虑星形连接策略。

此级别应该只用于需要最低的查询编译额外开销的特殊情况。完全由很简单的动态 SQL 语句（它们存取经过良好索引的表）组成的应用程序是适用使用查询优化级别 0 的一个很好的例子。

1 - 此级别指导优化器使用一定程度的优化，大致相当于 DB2/6000 版本 1，再加上一些在版本 1 中没有的附加的低成本功能部件。特别是:

- 优化器不考虑任何非均匀分布统计。
- 只应用查询重写规则的一个子集，包括在 DB2/6000 版本 1 中提供的那些规则。
- 贪婪联合枚举（查看第149页的『选择最优连接的搜索策略』。）
- 禁止列表预读取和索引 AND 运算，以使它们不会被用在生成的存取方法中。

  注: 当使用随星型连接一起的半连接时，仍使用索引 AND 运算。

除了“合并扫描”连接及表扫描也可用以外，优化级别 1 十分类似于级别 0。

2 - 此级别指导优化器使用一定程度的优化，它显著改善了级别 1 的优化程度，同时使复杂查询的编译成本显著低于级别 3 及更高级别。特别是:

- 使用了所有可用的统计信息，包括频率和分位数非均匀分布统计信息。
- 应用了所有查询重写规则，只适用于极少情况的密集型计算规则除外。
- 使用了贪婪联合枚举（查看第149页的『选择最优连接的搜索策略』）。
- 考虑了大量的存取方法，包括列表预读取。
- 如果适用的话，考虑星形连接策略。

优化级别 2 除了使用贪婪联合枚举而不是“动态规划”以外，十分类似于级别 5。在所有使用贪婪联合枚举算法的优化级别中，此级别具有最高的优化程度，与级别 3 及更高级别相比，它对复杂查询的替代方案考虑较少，因而消耗的编译时间也少。因此建议将它用于决策支持或联机分析处理 (OLAP) 环境中非常复杂的查询。在这种情况下，很有可能偶尔执行同一查询，因此其存取方案不大可能在高速缓存中停留到出现下一个查询为止。

3 - 此级别请求执行适量的优化以生成存取方案。此级别与 DB2 MVS/ESA 版或 OS/390 版的查询优化特性基本匹配。此优化级别具有下列特性:

- 使用非均匀分布统计，该统计跟踪频繁出现的值（如果可用的话）。
- 应用大部分查询重写规则，包括子查询至连接的转换。
- 动态规划联合枚举（查看第149页的『选择最优连接的搜索策略』）:
  - 组合内部表的有限使用（查看第150页的『组合表』）
  - 涉及“查找”表的星形模式笛卡尔乘积的有限使用（参见第149页的『星形连接的搜索策略』）
- 考虑各种存取方法，包括列表预读取、索引 AND 运算和星形连接。

此级别适用于大量应用程序。使用此级别使优化器更有可能为具有四个或更多连接的查询选择最优的存取方案。但是，优化器可能无法考虑使用缺省查询优化级别选择的更好方案。

5 - 此级别指导优化器使用相当大量的优化来生成存取方案。例如，级别 5 具有下列特性:

- 使用所有可用的统计信息，包括频率和分位数非均匀分布统计信息。
- 除只在极少情况下才适用的那些计算密集型规则外，将应用所有其他查询重写规则，包括摘要表查询的路由选择。

- 动态规划联合枚举（查看第149页的『选择最优连接的搜索策略』）：
  - 组合内部表的有限使用（查看第150页的『组合表』）
  - 涉及“查找”表的星形模式笛卡尔乘积的有限使用（参见第149页的『星形连接的搜索策略』）
- 考虑各种存取方法，包括列表预读取、索引 AND 运算和摘要表路由选择。

当优化器检测到不能保证用于复杂动态 SQL 查询的附加资源和处理时间时，将减少优化。减少的范围或大小取决于机器大小和谓词数目。

当查询优化器减少执行的查询优化量时，它继续应用正常时应用的所有查询重写规则。但是，它的确使用了贪婪联合枚举法并减少了考虑的存取方案的组合数。

对于由事务和复杂查询组成的混合环境，查询优化级别 5 是一个很好的选择。此优化级别被设计为可以用高效的方式应用最有价值的查询转换和其他查询优化技术。

7 - 此级别指导优化器使用相当大量的优化来生成存取方案。级别 7 除了不减少用于复杂动态 SQL 查询的查询优化量以外，它与查询优化级别 5 是相同的。

9 - 此级别指导优化器使用所有可用的优化技术。这些技术包括:

- 所有可用的统计信息
- 所有查询重写规则
- 联合枚举的所有可能性，包括笛卡尔乘积和任意多种组合的内部结构
- 所有存取方法。

此级别可以大大扩展由优化器考虑的可能的存取方案数量。此级别应该用于确定对于使用大表的很复杂、运行时间很长的查询，更全面优化是否可以生成更好的存取方案。应使用解释和性能测量来验证是否已找到更好的方案。

如何设置优化级别?

请求特定查询优化级别的方法取决于您在使用静态 SQL 还是动态 SQL。

- *静态 SQL 语句*使用在 PREP 和 BIND 命令上指定的优化级别。SYSCAT.PACKAGES 目录表中的 QUERYOPT 列记录与程序包联编所用的优化级别。如果以隐式方式或使用 REBIND PACKAGE 命令重新联编程序包，将把这同一个优化级别用于静态 SQL 语句。如果想更改用于这些静态 SQL 语句的优化级别，必须使用 BIND 命令。如果未指定优化级别，则 DB2 使用 *dft_queryopt* 指定的缺省优化级别。

- *动态 SQL 语句*使用由 CURRENT QUERY OPTIMIZATION 专用寄存器指定的优化级别，而该寄存器是用 SQL SET 语句设置的。例如，下列语句将优化级别设置为 1:

  ```
  SET CURRENT QUERY OPTIMIZATION = 1
  ```

  要确保动态 SQL 语句总是使用同一个优化级别，应将此 SET 语句加入应用程序中。有关详情，参考 *SQL Reference*。

  如果尚未设置 CURRENT QUERY OPTIMIZATION 寄存器，将使用缺省查询优化级别联编动态语句。动态和静态 SQL 的缺省值由可配置的数据库参数 DFT_QUERYOPT 的值确定。级别 5 是缺省查询优化级别，除非已经更改了该缺省值。（有关此参数的详情，参见第374页的『缺省查询优化级别 (dft_queryopt)』。）联编选项和专用寄存器的缺省值是从 DFT_QUERYOPT 配置参数得到的。

需要多大级别的优化?

对缺省查询优化级别使用合理数量的资源将使大多数语句充分优化。在一个给定的优化级别中，查询编译时间和资源消耗主要受查询的复杂性特别是连接和子查询的数量影响。然而，编译时间和资源的使用也受对不同优化级别所执行的优化数量影响。对于任何优化级别，可以预料一个非常复杂的查询与一个简单的查询，在查询编译时间和资源消耗方面有着更大的差别。

下面内容可帮助您选择使用哪一个优化级别:

- 通过使用缺省查询优化级别开始。
- 如果不希望使用缺省级别，首先尝试级别 1、2 或 3。
- 对运行时间短（即，不到一秒）的查询，使用低优化级别（0 或 1）。（查看下列讨论以了解关于何时选择低优化级别的附加标准。）
- 如果您有许多表，这些表的同一列有许多连接谓词，并且您又关心编译时间，则使用优化级别 1 或 2。
- 对于运行时间长（即，大于 30 秒）的查询，使用高优化级别（3、5 或 7）。
- 在正常情况下，不应使用优化级别 9。
- 对于长时间运行的查询，使用 db2batch 运行该查询以确定编译和执行各花费了多长时间。
  - 如果大部分时间花费在编译上，那么降低优化级别。
  - 如果大部分时间花费在执行上，那么考虑更高的优化级别。

注意，查询优化级别 1、2、3、5 和 7 都比较通用。

仅当需要进一步减少查询编译时间并且了解将执行的 SQL 语句的种类（例如，非常简单的语句）时，才应考虑级别 0。此 SQL 将有可能具有下列特性:
- 存取单个表或仅存取少量表
- 读取一行或仅读取少量行
- 使用全限定的唯一索引。

这种 SQL 的典型例子是联机事务处理 (OLTP) 事务。

复杂的查询可能需要不同程度的优化，以便选择最佳存取方案。可对具有下列特性的查询考虑使用更高的优化级别:
- 存取大型表
- 大量谓词
- 很多子查询
- 很多连接
- 很多集合运算符，例如 UNION 和 INTERSECT
- 很多限定行
- GROUP BY 和 HAVING 操作
- 嵌套表表达式
- 大量视图。

复杂查询的典型例子是对完全规范化数据库的决策支持查询或月结报告查询，这些复杂查询至少应该使用缺省查询优化级别。

使用较高的查询优化级别的另一个原因是由查询生成程序产生的 SQL。许多查询生成程序创建的 SQL 运行效率不高。编写得很差的查询，包括那些由查询生成程序产生的查询，可能需要另外优化以便尽可能选择一个良好的存取方案。使用查询优化级别 2 和更高的级别可以改进编写得很差的 SQL 查询。

使用静态还是动态 SQL，以及是否重复执行同一个动态 SQL 也是重要的考虑事项。对于静态 SQL，只耗费一次查询编译时间和资源，而产生的方案可以使用许多次。一般来说，静态 SQL 应该总是使用缺省查询优化级别。而动态语句是在运行期联编和执行；因此，应该考虑对动态语句进行另外的优化是否将改善总体性能。但是，如果重复执行相同的动态 SQL 语句， 那么所选择的存取方案将被高速缓存。为了选择查询优化级别，可以象处理静态 SQL 语句那样处理该语句。

（有关何时使用静态和动态 SQL 的信息，参考 *Application Development Guide*。）

如果您认为一个查询可以从附加优化获益，但不能肯定，或者您关心编译时间和资源的使用，那么应该执行一些基准测试。此测试可以帮助您确定从不同的优化级别获得的利益的多少。查看第265页的『第12章 基准测试』以获取关于 db2batch

工具的一般技术和具体使用的信息。当设计和运行基准程序测试时，考虑应用程序中的 SQL 语句是静态的还是动态的:

- 对于**动态** SQL 语句，测试应比较该语句的平均运行时间。可以使用下列公式计算平均运行时间:

$$\frac{\text{编译时间 + 所有重复的执行时间的总和}}{\text{重复次数}}$$

  其中，重复次数表示每次编译 SQL 语句时，期望 SQL 语句执行的次数。

  注: 第一次编译后，如果环境更改而要求再编译动态 SQL 语句时，该语句将被重新编译。一旦将 SQL 语句高速缓存，就不需要再次编译，因为后来的 PREPARE 语句将再使用高速缓存的语句，假定环境没有更改的话。 （查看第293页的『目录高速缓存大小 (catalogcache_sz)』与第299页的『程序包高速缓存大小 (pckcachesz)』以获取关于当使用动态 SQL 时可以改善性能的高速缓存的信息。）

- 对于**静态** SQL 语句，测试应比较语句运行时间。

  注: 您可能也对静态 SQL 的编译时间感兴趣，但该语句的总计（编译与运行）时间没有什么意义。比较总计时间不能识别这样一个事实，即每次将静态 SQL 语句联编时，它可以运行多次，而在运行期，一般不将它联编。

限制结果集以改进性能

SELECT 语句定义满足搜索标准的一组行。DB2 优化器假设应用程序将会检索所有合格的行。此假设最适合于 OLTP 和批处理环境。但是，在“浏览”应用程序中，一个常见的情况是，一个查询定义一个可能很大的答案集，但只检索前几行，通常只检索填满该屏幕所需的那么多行。

优化器为检索所有合格行所作的缺省假设对于那些没有在存储的数据中更新或删除信息的应用程序来说也许并不是最合适的。

有四种方法修改 SELECT 语句，以限制或修改结果表，从而提高性能。它们是:

- FOR UPDATE 子句
- FOR READ/FETCH ONLY 子句
- OPTIMIZE FOR n ROWS 子句
- FETCH FIRST n ROWS ONLY 子句。

FOR UPDATE 子句

FOR UPDATE 子句标识可由后续定位的 UPDATE 语句更新的列。如果不带列名指定 FOR UPDATE 子句，则包括表或视图的全部可更新列。如果指定列名，则每个名称必须是非限定的，且必须标识表或视图的某一列。

当以下任何一种情况为真时，不能使用 FOR UPDATE 子句:

- 不能删除与 SELECT 语句关联的游标。
- 选择的列中至少有一列是目录表的不可更新列，且没有在 FOR UPDATE 子句中排除掉。

可将 DB2 CLI 连接属性 SQL_ATTR_ACCESS_MODE 用于 CLI 应用程序以实现同样的结果。有关详情，参考 *CLI Guide and Reference* 的 SQLSetConnectAttr() 一节。

FOR READ 或 FETCH ONLY 子句

FOR READ ONLY 子句确保结果表是只读的。FOR FETCH ONLY 子句具有相同的含义。

某些结果表被定义为只读。例如，被定义为只读的视图上 SELECT 的结果表。在这种情况下，仍可指定 FOR READ ONLY，但该子句无效。

对于允许更新和删除的结果表，指定 FOR READ ONLY 可能会提高 FETCH 操作的性能。当数据库管理程序可对数据执行分块（而不是互斥锁定）时，才会发生这种可能的性能改进。除了将查询用于定位的 UPDATE 或 DELETE 语句这些情况外，可使用 FOR READ ONLY 子句改进性能。

可将 DB2 CLI 连接属性 SQL_ATTR_ACCESS_MODE 用于 CLI 应用程序以实现同样的结果。有关详情，参考 *CLI Guide and Reference* 的 SQLSetConnectAttr() 一节。

OPTIMIZE FOR n ROWS 子句

OPTIMIZE FOR 子句提供了一个方法，使得一个应用程序可以声明它只想检索结果的一个子集或优先检索前几行。一旦了解此意图，优化器优先选择把检索前几行的响应时间减至最短的存取方案。另外，作为单个块发送到客户机的行数（参见第67页的『行分块』）由 OPTIMIZE FOR 子句中的 『n』 值来限制。因此，OPTIMIZE FOR子句既影响服务器从数据库检索合格行的方式，又影响将合格行返回到客户机的方式。

例如，假设您定期查询雇员表，来查找具有最高工资的雇员。

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
```

您定义了一个基于 SALARY 列的降序索引。但是，由于雇员是按雇员号排序的，所以工资索引可能很难群集。为了尽量避免许多随机的同步 I/O，优化器可能选择使用列表预读取存取方法（参见 第216页的『了解列表预读取』），此方法需要对合格的所有行的行标识符排序。在将前几个合格行返回至应用程序前，这可能导致一个延迟。通过向语句添加 OPTIMIZE FOR 子句，如下所示:

```
SELECT LASTNAME,FIRSTNAME,EMPNO,SALARY
FROM EMPLOYEE
ORDER BY SALARY DESC
OPTIMIZE FOR 20 ROWS
```

优化器将可能选择直接使用 SALARY 索引，如果它了解到在所有情况下将只检索具有最高工资的二十个雇员。不管可以将多少行分块，将只把每二十行组成的一个块返回至客户机。

使用 OPTIMIZE FOR 子句，将使优化器优先选择，可以避免大量操作或避免中断行流动的操作（如排序）的存取方案。使用 OPTIMIZE FOR 1 ROW 最有可能影响存取路径。因此，使用此子句可能有如下效果:

- 与复合内部的连接顺序不大可能，因为它们需要一个临时表。
- 连接方法可能更改。一个嵌套循环连接是非常可能的选择，因为它具有低额外开销成本，并且通常在只想检索少量行时更有效率。
- 更可能选择一个与 ORDER BY 子句匹配的索引。这是因为 ORDER BY 不需要排序。
- 不大可能选择列表预读取，因为此存取方法需要排序。
- DB2 不大可能请求顺序预读取，因为它意味着您只想查看少量的行。
- 在一个连接查询中，在 ORDER BY 子句中包含列的表可能选作外部表，前提是该外部表上有一个索引，该索引可提供 ORDER BY 子句所需的定序。

虽然 OPTIMIZE FOR 子句适用于所有优化级别（参见第57页的『调整优化级别』），但它在优化级别 3 和更高优化级别下工作得最好。在低于 3 的优化级别中，使用贪婪连接枚举方法（参见第149页的『选择最优连接的搜索策略』）有时会产生一个不能使它们自己很快检索前几行的多表连接的存取方案。

OPTIMIZE FOR 子句不阻止您检索全部合格行。但是，检索全部合格行的总经过时间可能大大高于允许优化器为整个答案集进行优化所需的时间。

如果您有一个使用调用层接口（DB2 CLI 或 ODBC）的已封装应用程序，可通过在 db2cli.ini 配置文件中使用 OPTIMIZEFORNROWS 关键字，让 DB2 CLI 自动将一个 OPTIMIZE FOR 子句添加至每个查询语句的末尾。有关其他信息，参考 *CLI Guide and Reference* 手册。

当用别名选择数据时，结果可能随数据源支持的不同而变化。如果该别名引用的数据源支持 OPTIMIZE FOR 子句，且 DB2 优化器将包含该子句的整个查询下推至数据源，那么将在发送到数据源的远程 SQL 中生成该子句。如果数据源不支持此子句，或者如果优化器决定在本地执行该子句（最低成本方案），则在 DB2 中本地应用 OPTIMIZE FOR 子句。在这种情况下，DB2 优化器将继续优先选择把检索某查询前几行的响应时间最小化的存取方案，但可供优化器生成方案的选项略微受到限制，因此从 OPTIMIZE FOR 子句获得的性能改善可能非常小。

如果同时指定了 FETCH FIRST 子句和 OPTIMIZE FOR 子句，则使用这两个值中较小的一个来影响通信缓冲区大小。为了达到最优化，将这两个值看作是互不相关的。有关这两个子句之间交互作用的详情，参见第68页的『使用 SELECT 语句』。

FETCH FIRST n ROWS ONLY 子句

OPTIMIZE FOR *n* ROWS 子句不阻止检索所有合格行。（检索全部合格行的总经过时间可能大大高于允许优化器为整个答案集进行优化所需的时间。）

FETCH FIRST *n* ROWS ONLY 子句设置从 SELECT 语句中可检索的最大行数。将结果表限制为只包含前几行可提高性能。无论结果表中有多少行（这基于没有指定此子句的 SELECT），只检索 n 行。

如果同时指定了 FETCH FIRST 子句和 OPTIMIZE FOR 子句，则使用这两个值中较小的一个来影响通信缓冲区大小。为了达到最优化，将这两个值看作是互不相关的。有关这两个子句之间交互作用的详情，参见第68页的『使用 SELECT 语句』。

DECLARE CURSOR WITH HOLD 语句

如果用包括 WITH HOLD 子句的 DECLARE CURSOR 语句声明游标，在落实该事务时任何打开的游标仍然打开。而且，除保护打开的 WITH HOLD 游标的当前游标位置的锁定之外，会释放所有其他锁定。

如果用包括 WITH HOLD 子句的 DECLARE CURSOR 语句声明游标，在该事务以 ROLLBACK 结束时任何打开的游标将被关闭。此外，释放所有锁定，并释放 LOB 定位器。

参见 第56页的『CLOSE CURSOR WITH RELEASE』 以了解与 CLOSE CURSOR 语句的比较。

可将 DB2 CLI 连接属性 SQL_ATTR_CURSOR_HOLD 用于 CLI 应用程序以实现同样的结果。有关其他信息，参考 *CLI Guide and Reference* 手册中的『SQLSetStmtAttr - 设置与语句相关的选项』一节。

如果有一个封装的应用程序使用调用层接口（DB2 CLI 或 ODBC），可在 `db2cli.ini` 配置文件中使用 CURSORHOLD 关键字，让 DB2 CLI 自动对每个已声明的游标假设应用 WITH HOLD 子句。有关详情，参考 *CLI Guide and Reference* 的事务配置关键字一节。

行分块

行分块是一种技术，它通过在单个操作中检索行*块*来减少数据库管理程序的额外开销。这些行存储在高速缓存中，且应用程序中的每个 FETCH 请求都从该高速缓存中获取下一行。当处理完块中的所有行后，由数据库管理程序检索其他行块。

该高速缓存在应用程序发出 OPEN CURSOR 请求时分配，且在关闭游标时被取消分配。该高速缓存的大小由配置参数确定，该参数用于分配 I/O 块的内存。使用的参数取决于客户机是本地还是远程的:

- 对于*本地应用程序*，使用参数 *aslheapsz* 为行分块分配高速缓存。（查看第313页的『应用程序支持层堆大小 (aslheapsz)』以获取有关此参数的信息。）
- 对于*远程应用程序*，客户机工作站上的参数 *rqrioblk* 用于为行分块分配*高速缓存*。将在数据库客户机上分配该高速缓存。（查看第314页的『客户机 I/O 块大小 (rqrioblk)』以获取关于此参数的信息。）

对于*本地*应用程序，您可使用下列公式来估计每块返回的行数，其中:
- *aslheapsz* 是以页计的内存
- 4 096 是每页的字节数
- *orl* 是以字节为单位的输出行长度:

```
每块行数 = aslheapsz * 4096 / orl
```

对于*远程*应用程序，您可使用下列公式来估计每块返回的行数，其中:
- *rqrioblk* 是以字节为单位的内存
- *orl* 是以字节为单位的输出行长度:

```
每块行数 = rqrioblk / orl
```

注意，如果在 SELECT 语句中使用 FETCH FIRST *n* ROWS ONLY 子句或 OPTIMIZE FOR *n* ROWS 子句，每块的行数将是下列各项中的最小值:

- 上述公式中计算的值
- FETCH FIRST 子句中 *n* 的值
- OPTIMIZE FOR 子句中 *n* 的值

在 PREP 和 BIND 命令上使用 BLOCKING 选项以指定下列其中一种行分块类型:

UNAMBIG
对只读游标和未指定为 『FOR UPDATE OF』 的游标进行分块。将模糊游标当作可更新的游标来处理。

ALL 对只读游标和未指定为 『FOR UPDATE OF』 的游标进行分块。将模糊游标当作只读游标来处理。

NO 不对任何游标进行分块。将模糊游标当作只读游标来处理。

有关这些类型的行分块的细节，参考 *Command Reference* 手册中的 PREP 和 BIND 命令说明。

如果在 PREP 和 BIND 命令中未指定选项，则缺省行分块类型为 UNAMBIG。对于命令行处理器和调用层接口，缺省行分块类型为 ALL。

参考 *SQL Reference* 以获取关于这些游标的详情。

调整查询

本节提供特定的考虑事项和准则，帮助您细调应用程序中的 SQL 语句。作为一个通用规则，这些准则可以有助于设计一个程序，以最大程度地减少系统资源的使用和存取一个非常大的表中的数据所需的时间。根据编译该 SQL 语句时所执行的优化量，您可能不需要细调您的 SQL 语句。该 SQL 编译程序可以将您的 SQL 重写为更有效的形式。参见第125页的『由 SQL 编译程序重写查询』和第57页的『调整优化级别』。

还有一点很重要，即应注意优化器选择的存取方案也受其他因素的影响，这些因素包括环境考虑事项和系统目录统计信息。如果您对应用程序的性能进行基准测试，可进行调整以改善存取方案。

使用 SELECT 语句

SQL 语言是一种非常灵活的高级语言。因此，可以编写不同的 *select 语句*来检索相同的数据。但是，对于不同形式和不同种类的优化，性能可以相差很大。

SQL 编译程序（包括查询重写和优化阶段）将选择一个存取方案，它将对您编码的查询产生结果集。因此，正如下列许多准则中所提到的，您应将您的查询编写为只获取您需要的数据。

使用 SELECT 语句时的准则

使用 *select 语句*的准则包括:

- 只指定在选择列表中需要的那些列。尽管用星号 (*) 指定所有列可能更简单，但会产生不需要的处理并返回不想要的列。
- 通过使用谓词将回答集限制为只包含您必需的那些行，以限制选择的行数。（有关不同类型的谓词以及它们对性能的相对影响的详情，参见第142页的『谓词术语』。）
- 当您要使用的行数大大小于可以返回的总行数时，对 *select 语句*指定 OPTIMIZE FOR 子句。此子句会影响存取方案的选择以及在通信缓冲区中分块的行。（有关详情，参见第67页的『行分块』。）
- 当要检索的行数很小时，不需要指定 OPTIMIZE FOR k ROWS 子句和 FETCH FIRST n ROWS ONLY 子句。然而，如果 n 值很大并且您想要通过快速获取前 k 行（而可能延迟后续 k 行）进行优化，则同时指定两个子句。根据 n 和 k 中的较小者来确定通信缓冲区的大小。

```
SELECT EMPNAME, SALARY FROM EMPLOYEE
   ORDER BY SALARY DESC
   FETCH FIRST 100 ROWS ONLY
   OPTIMIZE FOR 20 ROWS
```

- 指定 FOR READ ONLY （或 FOR FETCH ONLY）子句可改善性能，因为它允许您的查询利用行分块。它也可改善数据并行性，因为永远不会对指定了此子句的查询所检索到的行保持互斥锁。它也允许进行附加的查询重写。指定 FOR READ ONLY（或 FOR FETCH ONLY）子句，同时指定 BLOCKING ALL BIND，可改善联合体系统中对别名查询的性能。
- 对将要更新的游标指定 FOR UPDATE OF 子句也可改善性能，因为它使数据库管理程序在一开始就可选择更适当的锁定级别，从而避免潜在的死锁（参见第50页的『死锁』）和锁定转换（参见第48页的『锁定转换』）。
- 尽可能避免数字数据类型的转换。当比较值时，使用有相同数据类型的项目可能更有效。如果需要转换，则可能由于精度受限制使结果不准确，而且由于运行期转换而使性能降低。

  如果可能的话，使用下列数据类型:
  - 非可变字符，用于较短的列
  - 整数而不是浮点数或小数
  - 日期时间而不是字符
  - 数字而不是字符。
- 包含子句或操作（如 DISTINCT 或 ORDER BY）的 SQL 语句要求对数据排序，才能执行该操作。如果您要降低使用排序操作的可能性，当不需要这些子句时，不要指定它们。

- 要检查一个表中是否存在一些行，不要使用:

  ```
  SELECT COUNT(*) FROM  TABLENAME
  ```

  且不要检查是否有非零值，除非您知道该表将很小。当该表增大时，对所有行计数将影响性能。建议您改试选择单行。为此，可打开一个游标，获取一行，或执行单行 (SELECT INTO) 选择。（记住如果从 *select 语句*中发现多行，则要检查是否有 SQLCODE -811 错误。）
- 如果更新活动较少且您的表较大，则在频繁用作谓词的列上定义索引。
- 如果同一列出现在多个谓词子句中，则考虑使用 IN 列表。
- 对于配合主变量使用的大型 IN 列表，套入主变量子集可能会改进性能。

下列建议只适用于存取几个表的 *select 语句*。

- 连接表时使用连接谓词。（连接谓词是来自一个连接中不同表的两个列之间的比较。）
- 在连接谓词中的列上定义索引，可以更有效地处理连接。这也会改善 UPDATE 和 DELETE 语句的性能，它们包含存取几个表的 select 语句。
- 如果可能的话，避免在连接谓词中使用表达式或 OR 子句。对于这种情况，数据库管理程序不能使用某种连接技术，因此可能无法选择最有效的连接方法。
- 如果可能的话，确保在一个分区数据库环境中按连接列来分区连接的表。

有关详情，参见第144页的『连接概念』。

也可参考 *Application Development Guide*，以获取编写具有连接和子查询的 SQL 语句的详情。

复合 SQL

复合 SQL 允许您将几条 SQL 语句分组为单个的可执行块。包含于块中的 SQL 语句（*子语句*）可以单独执行；但是，通过创建和执行一个语句块，您可以减少数据库管理程序的额外开销。对于远程客户机，复合 SQL 也减少必须通过网络传送的请求的数目。

有两种类型的复合 SQL:

- **基本**

  当所有子语句都成功完成时，或当一个子语句因出错而结束时，应用程序将从数据库管理程序收到一个响应。如果一个子语句因出错而结束，则将整个块视为因出错而结束，并且将回滚在此块中对数据库所做的任何更改。
- **非基本**

当所有子语句都完成时，应用程序将从数据库管理程序接收到一个响应。不管其前面的子语句是否成功执行，一个块中的所有子语句都要执行。仅当回滚包含非基本复合 SQL的工作单元时，才可以回滚此语句组。

- DB2 Connect 不支持基本的复合 SQL
- 存储过程（也称为 DARI 例程）内支持复合 SQL
- 通过下列各项支持复合 SQL:
  - 嵌入式静态 SQL（参考 *SQL Reference* 手册）
  - DB2 调用层接口（参考 *CLI Guide and Reference* 手册）
  - JDBC（参考 *Application Development Guide* 手册）。

性能考虑事项和字符转换

当您的应用程序和数据库未使用相同的代码页时，如果可能的话，将把数据从一个代码页映射到另一代码页。要恰当地在应用程序和数据库的代码页之间映射数据，可能需要一些数据转换。

对于使用与数据库代码页不同的代码页运行的应用程序，此映射和数据转换会给该应用程序的处理时间带来一定量的额外开销。如果应用程序和数据库使用相同的代码页或相同的整理顺序，则可以改进应用程序的性能。

代码页转换

在下列情况下可能要进行字符转换:

- 当存取数据库的客户机或应用程序在与数据库的代码页不同的代码页中运行时。
  - 数据库转换将在数据库服务器设备上进行: 从应用程序代码页转换为数据库代码页; 并从数据库代码页转换为应用程序代码页。
- 当调入（或装入）一个文件的客户机或应用程序使用与调入（或装入）的文件不同的代码页运行时。
- 当 DB2 Connect 用于存取 DRDA 服务器上的数据时。

对于下列各项，将**不**进行字符转换:

- 文件名。
- 将传送至或来自于指定了 FOR BIT DATA 属性的一列的数据，或者在其结果为 FOR BIT 或 BLOB 数据的 SQL 操作中使用的数据。
- 没有安装与 EUC 或 UCS-2 之间进行转换的受支持转换功能的 DB2 产品或平台。当运行您的应用程序时，会接收到 SQLCODE -332 (SQLSTATE 57017)。

有关 EUC 代码页支持和“国家语言支持”(NLS) 考虑事项的详情，*参考管理指南: 计划*。

根据操作系统环境，DB2 数据库管理程序使用转换功能和转换表，或者当转换多字节代码页时使用 DBCS 转换 API。

注: 在多字节代码页之间的字符串转换，如 DBCS 与 EUC 之间，可能引起此字符串长度的增加或减少。

分配给一个国家的 PC DBCS、EUC 和 UCS-2 代码集中不同字符的代码点，在对相同字符排序时也许会产生不同的结果。如果需要对不同国家的代码集排序，应*参考管理指南: 计划*。

扩展 UNIX 代码 (EUC) 代码页支持

在 C 或 C++ 应用程序中使用图形数据的主变量的使用需要一些特殊考虑，包括特殊的预编译程序、应用程序性能和应用程序设计等问题。

如果需要 EUC 代码集来开发应用程序，应当参阅 *Administrative API Reference* 手册。

数据库和客户机应用程序对图形（即，双字节字符）数据的支持，在处理日语和繁体中文的 EUC 代码页中发现的很多字符时，必须克服双字节宽度限制。使用 UCS-2 代码集存储和操纵来自这些 EUC 代码页的图形数据。

存储过程

在数据库应用程序环境中，很多情况会重复出现；例如，接收一个固定的数据集，对一个数据库执行相同的多个请求，或者返回一个固定的数据集。存储过程允许一次调用一个远程数据库来执行一个预编程的过程。一次调用也许意味着对数据库的几次存取。

处理远程数据库的一条 SQL 语句需要发送两次传输: 一次请求和一次接收。然而，一个应用程序可包含很多 SQL 语句。不用存储过程，一个应用程序要完成它的工作就需要许多次传输。

当一个数据库客户机使用一个存储过程时，对于整个过程它只需要两个传输，因此减少了网络传输的数目。要调用一个存储过程，请求的应用程序必须在调用存储过程之前与包含此过程的数据库连接。

通常这些存储过程是在与数据库代理程序在不同的进程中运行。这种独立性要求存储过程进程和代理程序进程必须通过路由器通信。要获得一个存储过程的可能

的最佳性能，可能要将一个存储过程标识为“可信赖的”或“无保护的”，这样可直接在数据库代理程序进程中运行此过程。所谓“可信赖的”和“无保护的”表示什么意思呢？

- *无保护的*是指没有任何东西将存储过程与数据库代理程序所用的数据库控制结构分开这样一个事实。
- *可信赖的*是指作为一个管理员，您相信此存储过程不会意外地或恶意地破坏数据库控制结构。即，您相信它们以一种不危害您的数据库完整性的方式操作。

这两个术语含义相同，即，如果您的存储过程是“无保护的”，则您的存储过程也是“可信赖的”。由于破坏您的数据库会带来相关的危险，您**只应**在需要获得最大可能的性能效益时才使用未保护存储过程。另外，您应确保该过程编写得很严密，并且，在允许该过程作为未保护存储过程运行前已彻底测试过。如果在运行这些未保护存储过程之一时发生一个致命错误，则数据库管理程序将确定此错误是发生在应用程序中还是在数据库管理程序代码中，并进行适当的恢复。

无保护的存储过程有可能在恢复之后损坏数据库管理程序，这可能会导致数据丢失和／或数据库损坏。当运行可信赖的存储过程时，应极为小心。在几乎所有情况下，应用程序的正确性能分析都将得到期望的性能，而不会使用此选项。例如，可以通过使用触发器改进性能。

有两个方法创建无保护的存储过程:

- 使用 CREATE PROCEDURE 命令并指定 NOT FENCED 子句。
- 按所用平台的*快速入门*手册中的定义，将此过程放在一个特殊的目录中。（此方法对 Java 存储过程不起作用。）

要运行一个存储过程，运行调用此过程的应用程序的最终用户必须在运行时具有下列其中一个特权:

- 对与此存储过程相关的程序包的 EXECUTE 或 CONTROL 特权
- SYSADM 或 DBADM 权限

有关使用存储过程编写程序的信息，参考 *Application Development Guide* 手册。

激活数据库

启动一个数据库后，几种类型的数据将被高速缓存。例如，数据缓冲区是在缓冲池中被高速缓存，而程序包和动态 SQL 语句是在程序包高速缓存中进行高速缓存。

如果频繁出现没有用户与数据库连接，并且这些时间段与少数用户连接此数据库时的其他时间段交叉，则高速缓存提供的好处会丧失，因为高速缓存被频繁破坏。为避免这种情况，可考虑发出如下命令来激活数据库:

```
DB2 ACTIVATE DATABASE database
```

此命令激活指定的数据库并启动所有需要的服务，以便该数据库可用于连接，并可以供任何应用程序使用。通过 ACTIVATE DATABASE 初始化的数据库可以由 DEACTIVATE DATABASE 或 *db2stop* 来关闭。有关这些命令的详情，参考 *Command Reference* 手册。

应用程序的并行处理

DB2 支持的一种并行环境是需要对称多处理器 (SMP) 机器的环境。在此环境中，多个处理器共享对数据库的存取。这允许并行执行复杂的 SQL 请求，可将这些请求分开到多个处理器中来执行。

通过使用 CURRENT DEGREE 专用寄存器或 DEGREE 联编选项，可以指定编译您的应用程序时要实施的并行度。“度”仅指一个查询的并行执行部分的数目。在处理器数目和为并行度选择的值之间没有严格的关系。当运行您的应用程序时，不需要请求可用于您的硬件平台的处理器总数；您可以选择大于此数或小于此数。

每个并行度都增加系统内存和 CPU 额外开销。

当利用并行度时，您应该知道，需要修改一些配置参数才能优化性能。在一个具有高度并行性的环境中，应根据需要复查和修改控制共享内存和预读取的量的配置参数。有关与并行操作和分区数据库环境相关的参数列表，参见第388页的『并行』。

您可以使用 3 个配置参数来控制和管理并行度。第一个参数是 *intra_parallel* 数据库管理程序配置参数，它启用或仅用实例并行度支持。第二个是 *max_querydegree* 数据库配置参数，它设置数据库中任何查询的并行度上限。此值覆盖 CURRENT DEGREE 专用寄存器和 DEGREE 联编选项。第三个配置参数是 *dft_degree* 数据库配置参数。它设置 CURRENT DEGREE 专用寄存器和 DEGREE 联编选项的缺省值。

有关使用多个并行度的应用程序使用情况以及内在情况的详情，参考 *Application Development Guide* 手册。

如果以 DEGREE = ANY 来运行一个查询，则数据库管理程序根据许多因素，包括处理器数目和此查询的特征，来选择分区内并行度。运行期使用的实际并行度可能低于处理器数，这取决于这些因素。

并行度是在编译该语句时由 SQL 优化器确定的，并可在执行查询前根据数据库活动进行调整。如果大量使用系统，则并行度可能低于 SQL 优化器选择的并行度。发生这种情况是因为，分区内并行性积极使用系统资源来减少查询的经过时间，而这可能严重影响其他数据库用户的性能。

通过使用“SQL 解释设施”显示存取方案，可找到 SQL 优化器所选择的并行度。可使用数据库系统监控程序找到在运行期使用的并行度。有关“SQL 解释设施”和相关工具的详情，参见第177页的『第7章 SQL 解释设施』 和第475页的『附录C. SQL 解释工具』。有关监控程序的其他信息，参考 *System Monitor Guide and Reference*。

注: 可以不考虑硬件环境来设置并行度。这意味着您可以使用并行度而不必具有 SMP 机器。例如，在一个单处理器机器上的“受 I/O 约束”的查询可由于将并行度说明为 ″2″ 或更高值而受益。在这种情况下，单处理器可能不必等待输入或输出任务完成，就可处理新的查询。说明并行度为 ″2″ 或更高值并不能直接控制一个单处理器机器上的 I/O 并行性。象 LOAD 这样的实用程序可以不依赖这种声明来控制 I/O 并行性。当更改 *dft_degree* 时，也可使用关键字 ANY。使用 ANY 表示由优化器确定分区内并行度。

在很多情况下，使用*数据库代理程序*来协调并行执行。有关详情和影响数据库代理程序的各种数据库管理程序配置参数的列表，参见第227页的『数据库代理程序』。

第4章 环境考虑事项

除了在设计和编写应用程序时应考虑的那些因素（第37页的『第3章 应用程序考虑事项』中所描述的）之外，环境因素也可影响为应用程序选择的存取方案:

- 影响查询优化的配置参数
- 节点组对查询优化的影响
- 表空间对查询优化的影响
- 索引对查询优化的影响
- 影响联合体数据库查询的服务器选项。

有关影响 SQL 优化器因素的详情，还可参考第93页的『第5章 系统目录统计信息』。

当调整您的应用程序和环境时，对以上任何方面进行更改之后，应重新联编应用程序。这确保使用的是最佳存取方案。

影响查询优化的配置参数

有一些配置参数影响 SQL 编译程序选择的存取方案。其中的许多参数都适合单分区数据库，而某些参数仅适合分区数据库。当在一个分区数据库中使用配置参数时，建议每个参数所用的值在所有分区上是相同的。

在联合体系统中工作时，如果大多数查询存取别名，则应考虑在更改环境之前要发送的查询类型。例如，缓冲池不对来自数据源的页进行高速缓存；如果是这样，增加 *buffpage* 的参数值并不能保证在为包含别名的查询创建存取方案时优化器会考虑其他替代项。（数据源是在联合体系统内的 DBMS 和数据。）而且，优化器可以决定数据源表的本地具体化是否是成本最低的方法，或者是否是排序操作的必需步骤。在那种情况下，增加 “DB2 通用数据库” 可用的资源会提高性能。有关其他信息，参见第88页的『影响联合体数据库查询的服务器选项』和第289页的『数据库共享内存』。

以下是影响 SQL 编译程序所选存取方案的配置参数的列表:

- 第290页的『缓冲池大小 (buffpage)』。

  当选择存取方案时，优化器要考虑从磁盘将页读取至缓冲池的 I/O 成本。在优化器的计算中，优化器将估计满足一个查询所必需的 I/O 数。此估计包括预测缓冲

池的使用，因为不需要附加的物理 I/O 来读取已在缓冲池中的页中的行。优化器在估计是否能在缓冲池中找到某页时，要考虑在 BUFFERPOOLS 系统目录表中的 *npages* 列的值。

读取表的 I/O 成本可影响:
– 如何连接两个表，如第148页的『外部与内部的确定』中所述。
– 是否使用非群集索引来读取数据（参见第141页的『群集索引』）。

在一个数据库中可有多个缓冲池。在一个分区数据库中也可有多个缓冲池。可以有选择地将新缓冲池添加至数据库的每个分区中，也可以添加至所有分区。BUFFERPOOLS 和 BUFFERPOOLSNODE 系统目录表中的 *npages* 列用于分区数据库中的估计。

- 第373页的『缺省度 (dft_degree)』。

  dft_degree 配置参数指定 CURRENT DEGREE 专用寄存器和 DEGREE 联编选项的缺省值。值 1 表示无分区内并行性。值 -1 表示优化器根据处理器数目和查询类型来确定分区内并行度。

- 第374页的『缺省查询优化级别 (dft_queryopt)』。

  当编译 SQL 查询时，可使用查询优化级别来指导优化器使用不同程度的优化。有关选择适当的查询优化级别的详情，参见第57页的『调整优化级别』。

- 第332页的『活动应用程序的平均数 (avg_appls)』。

  SQL 优化器使用 *avg_appls* 参数来帮助估计对于所选存取方案，在运行期要使用多大的缓冲池。较高的此参数值可以影响优化器，使它为查询选择一个在缓冲池的使用方面更节省的存取方案。此参数值为 1 将使优化器认为整个缓冲池可供该应用程序使用。

- 第303页的『排序堆大小 (sortheap)』。

  如果排序不需要临时表来存储最终的已排序的数据列表，则可认为该排序是“管道”排序。即，可以按单一的顺序存取方式来读取排序的结果。管道排序的性能优于非管道排序，应尽可能使用它。（参见第159页的『使用优化器排序的影响』以了解非管道排序的定义，与管道排序对比。）

  当选择存取方案时，优化器要通过下列操作来估计排序操作的成本，包括评估是否可使用管道传送排序:
  – 估计要排序的数据量
  – 查看 *sortheap* 参数，以确定是否有足够的空间通过管道传送排序。

- 第297页的『锁定列表的最大存储器 (locklist)』和第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』。

  当所用的隔离级（参见第37页的『并行性』）是**可重复读 (RR)** 时，SQL 优化器将考虑 *locklist* 和 *maxlocks* 参数的值，来确定将行级锁定升级到表级锁定是否可能。如果优化器预测将对表存取执行锁定升级，则它将为该存取方案选择一个表级锁定，这样就不会在执行查询期间产生锁定升级的额外开销。

- 第400页的『CPU 速度 (cpuspeed)』。

  SQL 优化器使用 CPU 速度来估计执行特定操作的成本。优化器使用这些 CPU 成本估计和各种 I/O成本估计，来为查询选择最佳存取方案。

  一台机器的 CPU 速度可显著影响所选的存取方案。当安装或迁移数据库时，会自动将此配置参数设置为一个适当的值。仅当您要在测试系统上为一个生产环境建立模型时，或估计硬件更改的影响时，才应调整此参数。使用此参数为不同的硬件环境建立模型，使您可观察将为该环境选择的存取方案。

- 第305页的『语句堆大小 (stmtheap)』。

  语句堆的大小不影响优化器选择不同的存取路径；但是，它可以影响对复杂的 SQL 语句执行的优化量。

  如果未将 *stmtheap* 参数设置得足够大，您可能接收到 SQL 警告，指示无足够内存用于处理该语句。例如，SQLCODE +437 (SQLSTATE 01602) 可能指示用于编译一个语句的优化量小于指定查询优化级别时请求的量。（有关详情，参见第57页的『调整优化级别』。）

- 第394页的『最大查询并行度 (max_querydegree)』。

  当此参数的值为 "ANY" 时，优化器选择要使用的并行度。如果参数值为 "ANY" 以外的值，则使用用户指定的值来确定要用于应用程序的并行度。

- 第400页的『通信带宽 (comm_bandwidth)』。

  优化器使用通信带宽来确定存取路径。优化器使用此参数中的值来估计在一个分区数据库的各数据库分区服务器之间执行特定操作的成本。

有关其他信息，参见第277页的『调整配置参数』。

节点组对查询优化的影响

在分区数据库中，优化器认可表的并置，并在确定查询的最佳存取方案时使用它。假定在连接查询中频繁涉及的那些表位于一个分区数据库中的各分区中，那么理想情况是要连接的每个表中的行位于同一个数据库分区中。在执行连接操作期间，如果并置了该连接的两个表中的数据，就不需要将数据从一个分区移动至另一个分区。将两个表置于同一个节点组中，可以保证将两个表中的数据并置在一起。

有关并置表的详情，*参考管理指南: 计划*。

而且，在一个分区数据库中，将数据分布在多个分区上可缩短执行查询的估计时间（或成本）。表的数目、那些表中数据的位置以及查询类型（正如以上所述，是否需要连接）都会影响查询的成本。

表空间对查询优化的影响

表空间的某些特征可影响 SQL 编译程序选择的存取方案:

- 容器特性

  容器特征可显著影响执行查询时的相关 I/O 成本。当选择存取方案时，SQL 优化器会考虑这些 I/O 成本，包括存取不同表空间中的数据所产生的任何成本差别。优化器使用 SYSCAT.TABLESPACES 系统目录中的两列来帮助估计存取表空间中的数据的 I/O 成本:

  – OVERHEAD，它估计在将任何数据读入内存之前容器所需的时间（以毫秒计）。此额外开销活动包括容器的 I/O 控制器额外开销以及磁盘等待时间，后者包括磁盘寻道时间。

    可使用以下公式来帮助估计额外开销成本:

    ```
    OVERHEAD = 以毫秒计的平均寻道时间
               +（0.5 * 旋转等待时间）
    ```

    其中:

    - 0.5 表示半个旋转的平均额外开销
    - 对每个完整的旋转计算以毫秒为单位的旋转等待时间，如下所示:

      ```
      (1 / RPM) * 60 * 1000
      ```

      其中:

      - 除以每分钟旋转次数来获得每个旋转所需的分钟数
      - 乘以 60 秒／分
      - 乘以 1000 毫秒／秒。

    例如，假定磁盘每分钟旋转 7 200 次。使用旋转等待时间公式得到如下结果:

    ```
    (1 / 7200) * 60 * 1000 = 8.328 毫秒
    ```

    然后可假定平均寻道时间为 11 毫秒，使用以上结果计算 OVERHEAD 估计值:

    ```
    OVERHEAD = 11 + (0.5 * 8.328)
             = 15.164
    ```

    得出大约 15 毫秒的 OVERHEAD 估计值。

  – TRANSFERRATE，它估计将一页数据读入内存所需的时间（以毫秒计）。

    如果每个表空间容器是单个物理磁盘，那么可使用以下公式来帮助估计传送成本（以毫秒／页计）:

    ```
    TRANSFERRATE = (1 / spec_rate) * 1000 / 1 024 000 * page_size
    ```

    其中:

    - spec_rate 表示磁盘传送速率的技术要求，以 MB／秒计

- 除以 spec_rate 来获得秒 / MB
- 乘以 1000 毫秒 / 秒
- 除以 1 024 000 字节 / MB
- 乘以页的大小（以字节计）（例如，对于 4 KB 页为 4 096 字节 ）

比如，假设磁盘的技术要求速率为 3 MB / 秒。这将得出以下计算

```
TRANSFERRATE = (1 / 3) * 1000 / 1024000 * 4096
             = 1.333248
```

得到大约每页 1.3 毫秒的估计传送速率值。

如果表空间容器不是单个物理磁盘，而是磁盘阵列（如 RAID），则当试图确定要使用的传送速率时还有其他考虑事项。假设瓶颈处于磁盘级，如果阵列相对较小，则可用磁盘数乘以 spec_rate。然而，如果组成容器的阵列中磁盘数目很大，则瓶颈可能不是在磁盘级，而是在其它某个 I/O 子系统部件级别，如磁盘控制器、I/O 总线或系统总线。在这种情况下，不能假设 I/O 的吞吐量能力是 spec_rate 和磁盘数目的乘积。相反，必须在顺序扫描期间测量实际的 I/O 速率（以 MB 计）。例如，一个顺序扫描可能是 select count(*) from big_table，并且它的大小会是几 MB。将该结果除以组成表空间的容器数目，big_table 驻留在这种表空间中。将该结果替换上面给出的公式中的 spec_rate。例如，在扫描含有 4 个容器的表空间中的某个表时，所测得的 100 MB 的顺序 I/O 速率意味着每个容器 25 MB，或传送速率为 (1/25) * 1000 / 1024000 * 4096 = 0.16 毫秒 / 页。

分配给一个表空间的每个容器可驻留在不同的物理磁盘上。要获得最佳结果，用于给定表空间的所有物理磁盘应具有相同的 OVERHEAD 和 TRANSFERRATE 特征。如果这些特征不相同，则当设置 OVERHEAD 和 TRANSFERRATE 的值时您应使用平均值。

您可以从硬件技术说明或通过实验来获得这些列的媒体特定值。可在 CREATE TABLESPACE 和 ALTER TABLESPACE 语句上指定这些值。

在上面提到的把磁盘阵列作为容器的环境中，进行实验变得特别重要。应创建一个移动数据的简单查询，与针对平台的测量实用程序一起使用。然后，用表空间中不同的容器配置重新运行该查询。可使用 CREATE 和 ALTER TABLESPACE 语句来更改在环境中传送数据的方式。

I/O 成本信息可通过这两个值以多种方式影响优化器，包括是否使用索引来存取数据，以及要选择将哪个表用作连接中的内部表和外部表。

- 预读取

当考虑从表空间存取数据的 I/O 成本时，优化器也要考虑从磁盘预读取数据页和索引页可能给查询性能带来的潜在影响。预读取数据页和索引页可减少将数据读入缓冲池所需的额外开销和等待时间。有关详情，参见第214页的『将数据预读取至缓冲池』。

优化器使用 SYSCAT.TABLESPACES 中 PREFETCHSIZE 和 EXTENTSIZE 列的信息，来估计将对一个表空间执行的预读取量。

– 仅当创建表空间（例如，使用 CREATE TABLESPACE 语句）时才可设置 EXTENTSIZE。缺省数据块大小为 32 页（每页 4 KB），且通常足够大。
– 创建表空间并使用 ALTER TABLESPACE 语句时，可设置 PREFETCHSIZE。缺省预读取大小由 DFT_PREFETCH_SZ 数据库配置参数值来确定，该数据库配置参数值随操作系统的不同而不同。应复查第328页的『缺省预读取大小 (dft_prefetch_sz)』说明中调整此参数的建议，并按需要更改参数以改善数据移动。

以下显示更改 RESOURCE 表空间特征的语法示例:

```
ALTER TABLESPACE RESOURCE
  PREFETCHSIZE 64
  OVERHEAD     19.3
  TRANSFERRATE 0.9
```

当对表空间进行任何更改之后，应考虑重新联编应用程序，并使用 RUNSTATS 实用程序来收集有关索引的最新统计信息，以确保使用最佳存取方案。

索引对查询优化的影响

务必记住不要决定何时应使用索引；数据库管理程序可以根据可用的表和索引信息来做出该决策。然而，由于创建了可改善性能的必需索引，您在该过程中起了重要的作用。当创建了索引或更改了预读取大小（如以上所述）后，应收集有关索引的统计信息（使用 RUNSTATS 实用程序），并始终保持该统计信息是最新的，这对您也很重要。这表示您必须了解可以创建的索引种类以及创建它们的方式。

索引与无索引

对于一个数据库查询中引用的每个表，如果该表上不存在索引，则必须对该表执行表扫描。表越大，表扫描所花的时间越长。所谓*表扫描*，即是数据库管理程序顺序地存取一个表的每一行。可将它与*索引扫描*比较，索引扫描是指数据库管理程序使用索引存取数据。（参见第134页的『索引扫描概念』。）

如果优化器估计索引扫描比表扫描快，则将选择索引。索引文件一般较小，因此读取它所需的时间比读取整个表所需的时间要少，尤其在表增大时更是如此。此外，可能不需要扫描整个索引。应用于索引的谓词可减少要从数据页读取的行数。

每个索引项由一个搜索关键字值和一个指针组成，该指针指向包含该值的行。仅当在 CREATE INDEX 语句中指定了 ALLOW REVERSE SCANS 参数时，才能以相反方向搜索那些值。因此，在具有正确谓词的情况下，才可能对搜索分类。也可使用索引来获得已排序的行，使数据库管理程序在从表中读取这些行之后不必对它们排序。指定了 ALLOW REVERSE SCANS 之后，将允许使用索引以正向或反向次序直接获取行。有关其他详情，参考 *SQL Reference*。

唯一的索引除包含搜索关键字值和行指针外，还可能包含 include 列。

注: 您不能控制数据库管理程序是否使用索引。例如，不能仅因为在要查询的表上存在索引而保证可生成一个排序的查询结果。在处理查询期间，数据库管理程序可以使用此索引，但不是一定要使用它。只有存在 ORDER BY 子句时，才可保证结果集的次序。

索引可显著缩短存取时间；但是，索引也可给性能带来负面影响。在创建索引前，考虑多个索引给磁盘空间和处理时间带来的影响:

- 每个索引要占用一定的存储器或磁盘空间。准确的容量取决于表的大小以及在该索引中包括的列的大小和数目。
- 对一个表执行的每个 INSERT 或 DELETE 操作都需要对该表上的每个索引进行额外的更新。对于更改索引关键字的每个 UPDATE 操作，也是如此。
- LOAD 实用程序重建任何现存的索引或追加至现存的索引之后。
- 可在 LOAD 命令上指定 indexfreespace MODIFIED BY 参数，以替换创建索引时使用的索引 PCTFREE。
- 每个索引都有可能对查询添加替代存取路径，优化器将考虑该路径，因此会增加查询编译时间。

应谨慎选择索引来满足应用程序的需要。

要确定在特定的程序包中是否使用索引，可使用 SQL 解释设施，如第177页的『第7章 SQL 解释设施』中所述。

使用索引顾问

“DB2 索引顾问”是一个工具，它辅助您选择最优的表数据索引集合。获得此工具有多种途径:

- 可以通过“控制中心”来存取此工具，方法是选择“索引”文件夹，单击鼠标右按钮，并选择**创建 —> 使用向导创建索引**。

- 可以从命令行输入 db2advis 存取此工具。

有关“DB2 索引顾问”的详情，可在第193页的『SQL 建议设施』中找到。

创建索引准则

应创建哪些索引取决于数据和它的计划用途。以下准则可帮助您确定哪些索引是最有用的:

- 无论应用在何处，都使用 CREATE UNIQUE INDEX 语句定义主关键字和唯一关键字。（有关详情，参考 *SQL Reference*。）唯一索引可帮助优化器避免执行特定的操作，如排序。
- 使用 include 列来定义唯一索引可改善数据检索的性能。如果列满足下列条件，则适合作为唯一索引的 INCLUDE 列:
  - 被频繁存取，因此可从纯索引存取中受益
  - 不需要用来限制索引扫描的范围
  - 不影响索引关键字的排序或唯一性。

  有关 INCLUDE 列的详情，参考*管理指南: 计划*中的“创建索引或索引规范”一章。
- 使用索引可优化对含有较多数据页的表的频繁查询，数据页数由 SYSCAT.TABLES 目录视图中的 NPAGES 列确定:
  - 根据连接表时要使用的任何一列来创建索引。
  - 根据将用于定期搜索特定值的任何列来创建索引。
- 决定对关键字使用升序还是降序，这取决于该次序是主要使用的还是请求的。仅当在 CREATE INDEX 语句中指定了 ALLOW REVERSE SCANS 参数时，才能以相反方向搜索那些值。尽管可按正反两个方向扫描索引，但索引的正向扫描（即创建索引时指定的次序）比索引的反向扫描表现略好些。有关其他详情，参考 *SQL Reference*。
- 避免创建的索引是这些列上其他索引关键字的部分关键字。例如，如果列 a、b 和 c 上有索引，则列 a 和 b 上的第二个索引一般用处不大。
- 使用基于外部关键字的索引可改进对父表执行的删除和更新操作的性能。
- 对将频繁用于数据排序的列使用索引。
- 在创建多列索引时，如果第一个关键字列有多项选择，则首先选择通常用『=』谓词指定的那一列，或指定具有最多相异值的那些列。
- 在所有列中任意选择来创建索引，不仅消耗大量磁盘空间，而且导致准备时间过长。对于复杂的查询更是如此，对复杂的查询使用具有动态规划连接枚举的优化级别。（参见第57页的『调整优化级别』）。
- 下面提供了一个*经验法则*，用来确定将为表定义的索引的典型数目。此数目由数据库主要使用的索引来确定的:
  - 对于联机事务处理 (OLTP) 环境，应只有一个或两个索引

– 对于查询（只读）环境，可有五个以上的索引
– 对于混合查询／OLTP 环境，可有两个至五个的索引。

- 考虑定义一个群集索引，以帮助新插入的行根据该索引进行群集。群集索引可以尽量避免重组表。

  注： 当定义了一个群集索引时，应装入该表，且在每个数据页上保留空闲空间，以允许在这些页上进行插入。（通过在 ALTER TABLE语句上使用 PCTFREE 关键字，或使用 LOAD 命令的 pagefreespace MODIFIED BY 子句，来保留空闲空间。）
- 创建索引时考虑使用 PCTFREE 关键字。PCTFREE 在索引页上保留空间，以便将来对索引进行更新。这可能减少页分割的频率并提高性能。
- 创建索引时考虑使用 MINPCTUSED 选项。MINPCTUSED 指定索引叶页中已使用空间的最小阈值，并启用联机索引重组。这可能减少对数据和索引脱机重组的需要。

注： 已说明临时表不支持索引。

以下是创建索引以改善性能的典型情况:

- 可在最频繁处理的查询和事务的 WHERE 子句中所使用的那些列上创建索引。

  以下的 WHERE 子句:

  ```
  WHERE WORKDEPT='A01' OR WORKDEPT='E21'
  ```

  根据 WORKDEPT 创建索引以改进性能，除非那些值频繁出现。
- 可在按整理顺序对行排序所使用的一列或多列上创建索引。不仅在 ORDER BY 子句中，而且其他功能，如 DISTINCT 和 GROUP BY 子句都需要定序。

  以下示例使用 DISTINCT 子句:

  ```
  SELECT DISTINCT WORKDEPT
    FROM EMPLOYEE
  ```

  数据库管理程序可使用定义为升序或降序的、基于 WORKDEPT 的索引来消除重复值。这同一个索引也可用于 GROUP BY 子句中，以将值分组，如以下示例所示:

  ```
  SELECT WORKDEPT, AVERAGE(SALARY)
    FROM EMPLOYEE
  GROUP BY WORKDEPT
  ```
- 可创建索引来命名一个语句中的每一列。当用此方式指定索引时，生成的纯索引存取表示通过避免表存取可更有效地检索数据。

  例如，假定发出以下 SQL 语句:

```
SELECT LASTNAME
     FROM EMPLOYEE
  WHERE WORKDEPT IN ('A00','D11','D21')
```

如果为 EMPLOYEE 表的 WORKDEPT 和 LASTNAME 列定义了索引，则通过扫描索引而不扫描整个表可能会更有效地处理该语句。注意，因为该谓词基于 WORKDEPT，因此此列应是该索引的第一列。

- 索引上的 Include 列是改进对表使用索引的另一种方法。使用上述示例，可将唯一索引定义为:

  ```
  CREATE UNIQUE INDEX x ON employee (workdept) INCLUDE (lastname)
  ```

  指定 lastname 为 include 列而不是索引关键字的一部分，意味着 lastname 只存储在索引的叶页上。

管理索引的性能提示

下面内容可帮助您了解如何通过适当使用和管理索引来影响性能:

1. **索引创建**

   当在大型表上创建索引且有 SMP 机器时，可考虑将 *intra_parallel* 设置为 YES (1) 或 SYSTEM (-1)，以利用并行性能改进。

   可使用多个处理器来扫描数据并对数据排序。在创建索引期间，仅当 *indexsort* 数据库配置参数为 NO 时（该参数的缺省值是 YES），多个处理器显示不出优势。该参数控制在创建索引期间是否对索引关键字排序。

2. **索引表空间**

   索引可存储在另外的表空间中，与其他表数据分开存储。这样，通过减少读/写磁头的移动，可以更有效地使用磁盘存储器。也可创建您的索引表空间，以便在更快的物理设备上存储它们。

   也可给表空间分配单独的缓冲池，这样索引页不会由于存在大量数据页而被挤出该缓冲区。

   当未将索引放在单独的表空间中时，数据页和索引页使用相同的数据块大小和预读取量。如果将索引放在另外的表空间，则可为一个表空间的所有特征选择不同的值。因为索引一般比表小，且分布在更少的容器中，因此通常只查找较小的数据块大小，如 8 和 16。有关详情，参见第142页的『索引页预读取』。 SQL 优化器将考虑对表空间使用更快的设备，如第80页的『表空间对查询优化的影响』中所述。有关表空间的详情，参阅*管理指南: 计划*。

3. **分组级别**

   如果 SQL 语句需要排序（例如，ORDER BY、GROUP BY、DISTINCT）且有适当的索引来满足该排序，则数据库管理程序可能有机会不选择该索引。这可能在下列情况下发生:

- 索引分组级别较低（查看 SYSCAT.INDEXES 的 CLUSTERRATIO 和 CLUSTERFACTOR 列）
- 表很小，扫描该表并在内存中对回答集排序所消耗的成本更低
- 对存取该表有多个索引可供选择。

建议在创建群集索引后，执行 REORG 或排序，然后执行 LOAD。通常，一个表只能在一个索引上群集。您的表和索引应该按该表的群集索引的顺序来构建。群集索引试图维护数据的特定次序，以改进 RUNSTATS 实用程序收集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。

若在装入或重组一个表之前改变该表，也应考虑使用 PCTFREE。为了维护群集，每个表都需要在每个数据页上有可用的空间，以用于附加的插入。当该空间可用时，附加的插入才可与现存数据进行群集。结果，在为了与附加数据群集而在每一页中留下一定百分比的空闲空间后，将需要考虑向表中装入数据。为此，可首先创建该表，然后使用 PCTFREE 参数改变该表。类似地，在重组数据前，应考虑用 PCTFREE 参数改变该表。否则，如果未设置 PCTFREE，该重组将除去所有多余的空间。

目前在更新期间不维护群集。即，如果某人更新了一条记录，因此就更改了它在群集索引中的关键字值，不必将该记录移至新页以维护群集次序。要维护群集，代替 UPDATE 的另一个方法是使用 DELETE，然后使用 INSERT。

4. **RUNSTATS 实用程序**

   当创建新索引后，您应使用 RUNSTATS 实用程序来收集索引统计信息。这些统计信息允许优化器确定使用索引是否改进存取性能。有关此主题的详情，参见第94页的『使用 RUNSTATS 实用程序收集统计信息』。

5. **重组索引**

   要从索引获得最佳性能，应考虑定期重组索引。更新您的表可能使索引页预读取效率降低。要保持索引页预读取的高效性，必须重组该索引。

   可以通过卸下并重新创建该索引或通过使用 REORG 实用程序来重组索引。有关详情，参见第223页的『重组目录和用户表』。

   为了防止频繁重组，可在创建索引时指定 PCTFREE。在创建索引期间指定 PCTFREE 参数，可以在创建时在每个索引的叶页上保留空闲空间。这样在涉及该索引的后续活动中，可将记录插入索引，而使索引页分割的可能性较小。索引页分割导致索引页既不是连续的也不是按顺序的。这将使执行索引页预读取的能力降低。为索引选择适当的 PCTFREE 可以避免重组或降低重组索引的频率。

   注: 在重组期间重新创建索引时，使用创建该索引时指定的 PCTFREE。

   卸下并重新创建索引可得到大致连续且顺序排列的新的一组页。这可提高索引页预读取的执行性能。

尽管为到达该目的所消耗的成本更高，REORG 实用程序仍确保数据页的群集。这个群集为存取大量数据页的索引扫描带来很大的好处。

如果您在对称多处理器 (SMP) 系统环境中工作，当 *intra_parallel* 是 YES 或 ANY 时，REORG 实用程序将使用多个处理器。

6. **使用 EXPLAIN**

   定期在您最频繁使用的查询上运行 EXPLAIN，并检查每个索引是否至少使用了一次。如果有一个索引未在任何查询中使用，考虑卸下该索引。

   也可使用 EXPLAIN 来了解在大型表上的表扫描是否是作为嵌套循环连接的内部来处理的。这将指示连接谓词列上的索引或者丢失，或者被认为是应用于该连接谓词的效率不高。或者，该连接谓词不存在。

7. **易变的表**

   *易变的*表是在运行时它的内容可能从空变为很大的一种表。生成一个使用易变表的存取方案，可使优化器使用表扫描而不是用索引扫描来存取该易变表。

   使用 ALTER TABLE...VOLATILE 语句声明表是“易变的”，可允许优化器对易变表使用索引扫描。有关此主题的其他信息，*参考管理指南: 计划*或 *SQL Reference*。

影响联合体数据库查询的服务器选项

联合体系统由 DB2 DBMS（联合体数据库）和一个或多个数据源组成。当发出 CREATE SERVER 语句时，要向联合体数据库标识数据源。当发出这些语句时，也可提供服务器选项来细化和控制涉及 DB2 和指定数据源的联合体系统操作的若干方面。稍后可使用 ALTER SERVER 语句更改服务器选项。有关 CREATE SERVER 和 ALTER SERVER 语句的详情，参考 *SQL Reference*。

注: 必须安装分布式连接安装选项，并将数据库管理程序参数 FEDERATED 设置为 YES，然后才能创建并指定服务器选项。

服务器选项及其值用于简化查询下推分析、全局优化和联合体数据库操作的其他方面。例如: 在 CREATE SERVER 语句中，可将特定的性能统计信息指定为服务器选项值。即，可将 *cpu_ratio* 选项设置为指示数据源 CPU 和联合体服务器 CPU 的相对速度的值。而且，可将 *cpu_ratio* 选项设置成指示数据源 I/O 设备和联合体服务器 I/O 设备的相对速率的值。运行 CREATE SERVER 时，此数据被添加到目录视图 SYSCAT.SERVEROPTIONS 中，优化器使用它来制定数据源的存取方案。如果统计信息更改（例如，如果升级了数据源 CPU，则这种情况可能会发生），可使用 ALTER SERVER 语句用这个更改更新 SYSCAT.SERVEROPTIONS。这样，优化器在制定该数据源的下一个存取方案时可使用这些更新。

表 8. 服务器选项及其设置

选项	有效设置	缺省设置
collating_sequence	指定数据源是否使用与联合体数据库相同的缺省整理顺序，这基于代码集和国家信息。如果数据源的整理顺序与 DB2 的整理顺序不同，则不能在数据源处对取决于 DB2 整理顺序的大多数运算进行远程求值。例如，在一个具有不同整理顺序的数据源处对别名字符列执行 MAX 列函数。因为如果是在远程数据源对 MAX 函数求值结果可能会不同，因此 DB2 将在本地执行聚合运算和 MAX 函数。 如果查询包含等号，即使整理顺序不同（设置为 'N'），也可能将那部分查询下推。例如，可能将谓词 C1 = 'A' 下推至某个数据源。当然，当数据源处的整理顺序不区分大小写时，不能下推这类查询。当数据源不区分大小写时，C1= 'A' 和 C1 = 'a' 的结果是相同的，这在区分大小写的环境 (DB2) 中是无法接受的。 管理员可创建具有特定整理顺序的联合体数据库，以使该顺序与数据源整理顺序匹配。如果所有数据源使用相同整理顺序，或者如果大多数或所有列函数都应用于使用相同整理顺序的数据源，则此方法可提高性能。 'Y' 数据源的整理顺序与联合体数据库的整理顺序相同。 'N' 数据源的整理顺序与联合体数据库的整理顺序不同。 'I' 数据源的整理顺序与联合体数据库的整理顺序不同，且不区分大小写（例如，会认为 'TOLLESON' 和 'TolLESon' 是相同的）。	'N'
comm_rate	指定联合体服务器及其相关数据源之间的通信速率。以每秒兆字节表示。 有效值大于 0 且小于 2147483648。值只能表示成整数，例如 12。	2
connectstring	指定与 OLE DB 提供器连接所需的初始化特性。有关连接字符串完整的语法和语义学，参见 *Microsoft OLE DB 2.0 Programmer's Reference and Data Access SDK, Microsoft Press, 1998* 中的 "OLE DB 核心部件的 DataLinks API"。	无
cpu_ratio	指示数据源的 CPU 比联合体服务器的 CPU 快多少或慢多少。 有效值大于 0 且小于 $1x10^{23}$。值可以表示成任何有效双精度表示法，例如 123E10、123 或 1.21E4。	'1.0'

表 8. 服务器选项及其设置 (续)

选项	有效设置	缺省设置
dbname	您希望联合体服务器存取的数据源数据库的名称。此参数是 DB2 系列数据源所必需的；它不适用于 Oracle** 数据源，因为 Oracle 实例只包含一个数据库。对于 DB2，此值与实例中的特定数据库相对应，或者，如果是 DB2 OS/390 版，则与数据库 LOCATION 值相对应。	无。
fold_id（参见此表末尾的注释 1 和 4。）	应用于用户 ID，联合体服务器将它们发送到数据源以便认证。有效的值为: 'U' 联合体服务器在将用户 ID 发送到数据源之前要将它转换为大写。这是 DB2 系列和 Oracle** 数据源的逻辑选项（参见此表末尾的注释 2）。 'N' 联合体服务器在将用户 ID 发送到数据源之前不对它执行任何操作。（参见此表末尾的注释 2）。 'L' 联合体服务器在将用户 ID 发送到数据源之前要将它转换为小写。 如果未使用这些设置，联合体服务器会尝试将用户 ID 以大写形式发送到数据源。如果该用户 ID 失败，服务器再尝试以小写形式发送它。	无。
fold_pw（参见此表末尾的注释 1、3 和 4。）	应用于口令，联合体服务器将它们发送到数据源以便认证。有效的值为: 'U' 联合体服务器在将口令发送到数据源之前要将它转换为大写。这是 DB2 系列和 Oracle** 数据源的逻辑选项。 'N' 联合体服务器在将口令发送到数据源之前不对它执行任何操作。 'L' 联合体服务器在将口令发送到数据源之前要将它转换为小写。 如果未使用这些设置，联合体服务器会尝试将口令以大写形式发送到数据源。如果该口令失败，服务器再尝试以小写形式发送它。	无。
io_ratio	指示数据源的 I/O 系统比联合体服务器的 I/O 系统快多少或慢多少。 有效值大于 0 且小于 $1x10^{23}$。值可以表示成任何有效双精度表示法，例如 123E10、123 或 1.21E4。	'1.0'

表 8. 服务器选项及其设置 (续)

选项	有效设置	缺省设置
node	将数据源定义为 RDBMS 的实例所用的名称。它是所有数据源必需的参数。 对于 DB2 系列数据源，此名称是在联合体数据库的 DB2 节点目录中指定的节点。要查看此目录，发出 **db2 list node directory** 命令。 对于 Oracle** 数据源，此名称是在 Oracle** tnsnames.ora 文件中指定的服务器名称。要在 Windows NT 平台上存取此名称，指定 Oracle** SQL Net Easy 配置工具的**查看配置信息**选项。	无。
口令	指定是否将口令发送至数据源。 'Y' 总是将口令发送至数据源，并验证它们。这是缺省值。 'N' 不将口令发送至数据源（无论是否存在任何用户映射），也不验证口令。 'ENCRYPTION' 总是以加密形式将口令发送至数据源，并验证它们。只对支持加密口令的 DB2 系列数据源有效。	'Y'
plan_hints	指定是否启用方案提示。方案提示是为数据源优化器提供额外信息的语句段。对于特定查询类型，此信息可改进查询性能。方案提示可帮助数据源优化器决定是否使用索引、使用哪个索引或使用哪种表连接顺序。 'Y' 如果数据源支持方案提示，则在数据源处启用方案提示。 'N' 在数据源处不启用方案提示。	'N'
pushdown	'Y' DB2 将考虑允许数据源求值运算。 'N' DB2 只检索远程数据源中的列，而且不允许数据源对其他运算（如连接）求值。	'Y'

表 8. 服务器选项及其设置 (续)

选项	有效设置	缺省设置
varchar_no_trailing_blanks	指定此数据源是否使用非空格填充 varchar 比较语义学。对于不包含尾部空格的变长字符串，某些 DBMS 的非空格填充比较语义学会返回与 DB2 的比较语义学相同的结果。如果确认在一个数据源中的所有 VARCHAR 表／视图列都不包含尾部空格，考虑对数据源将此服务器选项设置为 'Y'。此选项经常用于 Oracle** 数据源。确保考虑可能会有别名的所有对象（包括视图）。 'Y' 此数据源的非空格填充比较语义学与 DB2 的类似。 'N' 此数据源的非空格填充比较语义学与 DB2 的不同。	'N'

第89页的表8的注释:

1. 无论为认证指定了什么值，都要应用此字段。
2. 因为 DB2 以大写形式存储用户 ID，因此值 'N' 和 'U' 在逻辑上是等效的。
3. 当口令的设置为 'N' 时，fold_pw 的设置就没有作用。因为没有发送口令，因此大小写不会带来影响。
4. 避免其中任何一个选项的设置是空值。空设置看起来好象很有用，因为 DB2 将进行多次尝试来分辨用户 ID 和口令；但性能可能会降低（DB2 可能将用户 ID 和口令发送四次，然后才成功地通过数据源认证）。

第5章 系统目录统计信息

当优化 SQL 查询时，SQL 编译程序所作的决策很大程度上受到优化器为数据库内容建立的模型的影响。优化器使用此数据模型来估计可用于解析特定查询的替代存取路径的成本。

数据模型中的一个关键元素是所收集的统计信息的集合，这些信息是关于数据库中包含的数据和系统目录表中存储的数据。它包括表、别名、索引、列和用户定义函数 (UDF) 的统计信息。若数据统计信息发生变化，就会选择不同的存取方案作为存取期望的数据的最有效方法。

有助于对优化器定义数据模型的可用统计信息的示例包括:
- 一个表中的页数和非空页数
- 表中的行从其原始页移动（溢出）至其他页的程度。
- 一个表中的行数
- 一个列中的相异值数
- 一个索引的群集程度。即，一个表中各行的物理顺序顺从一个索引所达到的程度。
- 索引层数和每个索引中的叶页数
- 频繁使用的列值的出现次数（参见第100页的『收集和使用分布统计信息』）
- 列值在该列中给出的值的范围内的分布（参见第100页的『收集和使用分布统计信息』）
- 用户定义函数 (UDF) 的成本估计。

仅当明确请求时，才在系统目录表中更新对象的统计信息。可通过下列操作更新部分或全部统计信息:
- 使用 RUNSTATS（运行统计信息）实用程序（参见第94页的『使用 RUNSTATS 实用程序收集统计信息』）
- 使用 LOAD，并指定统计信息收集选项
- 编写针对一组预定义的目录视图运行的 SQL UPDATE 语句（参见第111页的『用户可更新的目录统计信息』）。注意，必须使用此技术来更新用户定义函数的统计信息（参见第116页的『更新用户定义函数的统计信息』）。除 UDF 外，只应人工更新目录，以便在测试系统上为一个生产环境建立模型，或用于“假设分析”。不应在生产系统上更新统计信息。

在联合体数据库系统中，从数据源收集别名的新统计信息的唯一方法是卸下别名，接着在数据源处运行 RUNSTARTS 的等效命令，然后重建别名。只要创建了别名，就可从数据源目录收集基本表的统计信息。

如果基础表中的数据定义信息更改，必须卸下别名然后重建它们。例如，如果向表定义中添加一列。

另外，如果性能下降，应考虑重建别名。另一个方法是人工更新 SYSSTAT.TABLES 中的统计信息。

创建视图的别名时要小心。统计信息（如此别名返回的行数）可能不能反映评估此视图的实际成本。如果在单个基表上定义视图，且在 SELECT 列表上没有应用列函数，则优化器可利用的统计信息将会很精确。如果视图较复杂，可考虑在联合体数据库系统的 “DB2 通用数据库” 服务器处为视图基表创建基于别名的新视图，以便优化器可生成一个有效的方案来存取数据。

其他信息

SYSCAT 和 SYSSTAT 目录包含有关收集的统计信息的资料。参考 *SQL Reference*:

- 获得有关所有目录视图和它们包含的列的信息。
- 获得有关所有可更新的目录视图和它们包含的列的信息。如果您只对目录表的统计信息列感兴趣，也可参考本节。
- 获得有关表统计信息的资料。
- 获得有关列统计信息的资料。
- 获得有关列分布统计信息的资料。
- 获得有关索引统计信息的资料。
- 获得有关用户定义函数统计信息的资料。

使用 RUNSTATS 实用程序收集统计信息

RUNSTATS 实用程序更新系统目录表中的统计信息，以帮助执行查询优化进程。如果没有这些统计信息，数据库管理程序可能会作出对 SQL 语句性能有负面影响的决策。RUNSTATS 实用程序允许您收集有关表和／或索引中包含的数据的统计信息。

在下列情况下，可使用 RUNSTATS 实用程序收集表和索引数据的统计信息，以便在选择存取方案时提供准确的信息:

- 当一个表已装入数据且已创建适当的索引时。
- 当一个表已用 REORG 实用程序重组时。
- 当发生了影响一个表及其索引的大量更新、删除和插入时。（此处所指的 “大量” 可能表示有 10% 到 20% 的表和索引数据受影响。）
- 在联编其性能至关重要的应用程序之前

- 当期望与先前的统计信息比较时。定期运行统计信息可以尽早发现性能问题，如下所述。
- 当预读取量更改时。
- 当使用了 REDISTRIBUTE NODEGROUP 实用程序时。

当您在一个分区数据库中工作时，通过在单个节点上执行 RUNSTATS 操作来收集与一个表及其索引相关的统计信息。（执行该实用程序所在的节点由发出该命令的节点是否包含表数据来确定。有关详情，参见『执行 RUNSTATS 所在的数据库分区』。）因为假定目录中存储的统计信息表示表级信息，因此数据库管理程序收集的节点级统计信息要适当地乘以包含该表分区的节点的数目。这就提供了一个实际统计信息的近似值，这些实际的统计信息是通过在每个节点执行 RUNSTATS 并聚合这些统计信息而收集到的。

注: DB2 查询优化器假定均匀地将属性值（数据）放在系统的各数据库分区上。如果数据的放置不等，应在您认为具有代表性表所分布的数据库分区上运行此命令。

执行 RUNSTATS 所在的数据库分区

当您在一个表上调用 RUNSTATS 时，必须与存储该表的数据库连接，但是从中发出该命令的数据库分区不必包含此表的一个分区:

- 如果发出 RUNSTATS 的数据库分区包含该表分区，则该实用程序在此数据库分区执行。
- 如果发出 RUNSTATS 的数据库分区不包含表分区，则将请求发送到节点组中保留该表分区的第一个数据库分区。然后，该实用程序在此数据库分区执行。

分析统计信息

分析统计信息可以指示何时需要重组。其中一些指示有:

- 索引的群集

  如果收集了群集率统计信息，则它们的值将在范围 0 至 100 之间。如果收集了群集因子统计信息，则它们的值将是 0 至 1 之间的一个数。这两个群集统计信息中只有一个将记录在 SYSCAT.INDEXES 目录中。通常，表中只有一个索引可以有较高的分组级别。值 -1 用于指示没有统计信息可用。

  如果您希望比较比率值，则将该群集因子乘以 100 来获得表示该群集量的一个百分比值。

  非专用于索引存取的索引扫描在具有较高群集率的情况下可能执行得更好。低的群集率导致此类扫描要执行更多的 I/O，因为在每个数据页经过第一次存取后，下次存取该页时，该页仍在缓冲池中的可能性减小。增加缓冲区大小可改进非群集索引的性能。（有关数据库管理程序可以如何对具有低群集率的索引改

进索引扫描性能的信息，参见第216页的『了解列表预读取』；有关优化器如何使用索引统计信息的资料，参见第141页的『群集索引』。）

如果表数据最初是用某个索引来群集的，而以上的群集信息指示现在使用该相同的索引无法较好地将数据群集，则您可能希望重组该表以使用该索引来重新群集数据。

- 行的溢出

  溢出数指示无法放在其原始页上的行的数目。当使用更长的值更新 VARCHAR 列时，可能发生这种情况。在这种情况下，将一个指针保留在该行的原始位置。这可能降低性能，因为数据库管理程序必须跟随该指针来查找该行的内容，这增加了处理时间并也可能增加 I/O 数。

  随着溢出行数的增多，重组表数据的潜在好处也增大。重组表数据将消除行的溢出。

- 文件页的比较

  可将具有行的页数与一个表包含的总页数比较。在执行表扫描时，将读到空页。当删除整个范围内的行时，可能出现空页。

  随着空页数的增多，对重组表的需求也增加。重组一个表时，可通过收回这些空页来压缩该表使用的空间容量。除了可更有效地使用磁盘空间外，收回未使用的页也可改进表扫描的性能，因为减少了要读入缓冲池中的页数。

- 叶页数

  叶页数预测使用一个索引来进行完整的扫描需要多少索引页 I/O。

  随机的更新活动可使页分割发生，这就增加了索引的大小，并使该大小超出必需的最小空间容量。当在重组一个表期间重建索引时，尽可能使用最小空间容量来构建每个索引是可能的。有关索引的最小空间需求的详情，参见第82页的『索引对查询优化的影响』，或参考*管理指南: 计划*中的“创建索引或索引规范”一节。

  注: 当重建索引时，会在每个索引页上保留百分之十的缺省空闲空间。在第一次创建索引时，可使用 PCTFREE 参数来增加空闲空间容量。然后，每当重组索引时，都使用 PCTFREE 值。如果您希望减少重组索引的次数，则拥有一个大于百分之十的空闲空间可能很重要。空闲空间用于插入附加索引。

RUNSTATS 也可帮助您确定性能如何与数据库中的更改相关。统计信息显示一个表中的数据分布。当例行使用时，RUNSTATS 提供在一段时间内有关该表和索引的数据，这样当您的数据模型随时间变化时，您可标识该模型的性能走向。

理想情况是，您应在运行统计信息之后重新联编应用程序，因为该查询优化器可以根据新的统计信息来选择不同的存取方案。

如果您没有足够的时间一次收集全部的统计信息，可选择定期运行 RUNSTATS，以只更新可收集的那部分统计信息。如果对选择性部分更新运行 RUNSTATS 期间由于表上的活动而产生了不一致性，则发出警告信息（SQL0437W，原因码 6）。例如，首先使用 RUNSTATS 来收集表分布统计信息。然后，使用 RUNSTATS 来收集索引统计信息。如果检测到由于表上的活动产生了不一致性，则删除表分布统计信息，并发出警告信息。建议在发生这种情况时运行 RUNSTATS 来收集表分布统计信息。

应定期使用 RUNSTATS 一次性收集表和索引统计信息，以确保索引统计信息和表统计信息同步。索引统计信息保留自上次运行 RUNSTATS 以来收集的大部分表和列的统计信息。如果自上次收集该表的统计信息以来已对该表做了大量修改，则只收集该表的索引统计信息将使两个统计信息集合不同步。

在下列情况下，您可能希望只根据索引数据来收集统计信息:
- 自执行该实用程序以来已创建了新索引，且您不想重新收集有关该表数据的统计信息。
- 存在影响索引的第一列的大量数据更改。

RUNSTATS 实用程序允许您收集各种级别的统计信息。对于表，您可以收集基本级别的统计信息，您也可收集一个表内列值的分布统计信息（参见第100页的『收集和使用分布统计信息』）。对于索引，可以收集基本级别的统计信息，也可收集详细的统计信息，它们可帮助优化器更好地估计一个索引扫描的 I/O 成本。（参见第141页的『群集索引』以了解这些“详细”统计信息）。

注: 不收集 LONG、大对象 (LOB) 或结构化类型列的统计信息。对于行类型，不收集子表的表级统计信息 NPAGES、FPAGES 和 OVERFLOW。不收集扩充索引的统计信息，也不收集已说明临时表的统计信息。

下面的表显示 RUNSTATS 实用程序更新的目录统计信息:

表 9. 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES）

统计信息	说明	RUNSTATS 选项	
		表	索引
FPAGES	一个表所用的页数	是	是
NPAGES	包含行的页数	是	是
OVERFLOW	溢出的行数	是	否
CARD	表中的行数（基数）	是	是（注 2）

表 9. 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES） (续)

统计信息	说明	RUNSTATS 选项	
		表	索引
注: 1. 对于一个分区数据库，每个统计信息的值是根据该数据库分区上的计数乘以数据库分区数的值来估计的。 2. 如果该表未定义索引，而您请求了索引的统计信息，则不更新新的 CARD 统计信息。而保留先前的 CARD 统计信息。			

表 10. 列统计信息（SYSCAT.COLUMNS 和 SYSSTAT.COLUMNS）

统计信息	说明	RUNSTATS 选项	
		表	索引
COLCARD	列基数	是（注 1）	是（注 2）
AVGCOLLEN	列的平均长度	是	是（注 2）
HIGH2KEY	列中的第二个最大值	是	是（注 2）
LOW2KEY	列中的第二个最小值	是	是（注 2）
NUMNULLS	列中的空值数	是	是（注 2）
注: 1. 要对表中的所有列估计 COLCARD。在一个分区数据库中，如果该列是该表的单列分区关键字，则该计数的值是根据该数据库分区上的计数乘以数据库分区数之积来估计的。 2. 为索引关键字中的第一列收集列统计信息。			

表 11. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES）

统计信息	说明	RUNSTATS 选项	
		表	索引
NLEAF	索引叶页数	否	是（注 3）
NLEVELS	索引层数	否	是
CLUSTERRATIO	表数据的分组级别	否	是（注 2）
CLUSTERFACTOR	细分的分组级别	否	详细的（注 1，2）
DENSITY	SEQUENTIAL_PAGES 与该索引占用的页范围中页数的比（百分比）（注 4）	否	是
FIRSTKEYCARD	在索引的第一列中的相异值数	否	是（注 3）
FIRST2KEYCARD	在索引的前两列中的相异值数	否	是（注 3）

表 11. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES） (续)

统计信息	说明	RUNSTATS 选项	
		表	索引
FIRST3KEYCARD	在索引的前三列中的相异值数	否	是（注 3）
FIRST4KEYCARD	在索引的前四列中的相异值数	否	是（注 3）
FULLKEYCARD	在索引的所有列中的相异值数	否	是（注 3）
PAGE_FETCH_PAIRS	不同缓冲区大小的页读取估计值	否	详细的（注 1，2）
SEQUENTIAL_PAGES	按索引关键字顺序位于磁盘上的叶页的数目，在这些叶页之间很少有或没有大的间隔	否	是

注:

1. 通过在 RUNSTATS 命令上指定 DETAILED 子句，或当调用 RUNSTATS API 时为 `statsopt` 参数指定 `A`、`Y` 或 `X`，来收集详细的索引统计信息。
2. 除非该表很大，否则，不使用 DETAILED 子句收集 CLUSTER_FACTOR 和 PAGE_FETCH_PAIRS。如果该表超过 25 页，则收集 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS 统计信息。在这种情况下，CLUSTERRATIO 是 -1（不收集）。如果该表是一个相当小的表，则 RUNSTATS 只填写 CLUSTERRATIO，而不填写 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS。如果未指定 DETAILED 子句，则只收集 CLUSTERRATIO 统计信息。
3. 对于一个分区数据库，该值是根据在该数据库分区上的计数乘以数据库分区数的值来估计的。
4. 此统计信息测量在包含属于该表的索引的文件中页数的百分比。对于只定义了一个索引的表，DENSITY 通常应为 100。优化器使用 DENSITY 来估计如果预读取了索引页，那么平均可能从其他索引读取多少个不相关的页。

表 12. 列分布统计信息（SYSCAT.COLDIST 和 SYSSTAT.COLDIST）

统计信息	说明	RUNSTATS 选项	
		表	索引
DISTCOUNT	如果 TYPE 是 Q，则是小于或等于 COLVALUE 统计信息的相异值数目	分布（注 2）	否
TYPE	有关行是提供高频值还是分位数统计信息的指示符	分布	否
SEQNO	序号的频率排序，它帮助唯一地标识该表中的行	分布	否

表 12. 列分布统计信息（SYSCAT.COLDIST 和 SYSSTAT.COLDIST） (续)

统计信息	说明	RUNSTATS 选项	
		表	索引
COLVALUE	要收集其频率或分位数统计信息的数据值	分布	否
VALCOUNT	数据值在列中出现的频率，或对于分位数，小于或等于数据值 (COLVALUE) 的值的数目	分布	否

注:

1. 通过在 RUNSTATS 命令上指定 WITH DISTRIBUTION 子句，或当调用 RUNSTATS API 时为 statsopt 参数指定 A、D 或 Y，来收集列分布统计信息。注意，除非列值严重不均匀，否则，可能不收集分布统计信息。
2. 仅对索引中的第一个关键字列收集 DISTCOUNT。
3. 在一个分区数据库中，VALCOUNT 是该数据库分区上的计数乘以数据库分区数的估计值。当 TYPE是 'F' 且该列是表的单列分区关键字时除外，在这种情况下，VALCOUNT 只是该数据库分区上的计数。

有关列分布统计信息的详情，参见『收集和使用分布统计信息』。

RUNSTATS 实用程序不收集用户定义函数的统计信息。您必须人工更新这些函数的统计信息。参见第111页的『用户可更新的目录统计信息』和第116页的『更新用户定义函数的统计信息』。

收集和使用分布统计信息

数据库管理程序可以收集、维护和使用“高频值统计信息”和“分位数”，这两种类型的统计信息使用一种简洁方式来估计在一个列中数据值的分布。优化器使用这些统计信息可以对一个列中满足给定等式或范围谓词的列数给出准确得多的估计。因此，这些更准确的估计可增加优化器将选择一个最优方案的可能性。

您可以在 RUNSTATS 命令上使用 WITH DISTRIBUTION 子句，来收集有关这些数据值分布的统计信息。当收集有关 RUNSTATS 实用程序在附加额外开销方面的这些附加统计结果时，SQL 编译程序可使用此信息来帮助确保选择最佳存取方案。

在某些情况下，数据库管理程序将不收集分布统计信息，也不返回错误。例如:

- 将 *num_freqvalues* 和 *num_quantiles* 配置参数设置为零 (0)，以指示您不想要收集分布统计信息。有关这些参数的详情，参见:
  - 第104页的『您应保留多少个统计信息？』
  - 第375页的『保存的高频值数目 (num_freqvalues)』
  - 第376页的『列的分位数数目 (num_quantiles)』。

- 已知数据的分布情况，不必使用分布统计信息。例如，一列中没有任何数据值出现一次以上，即该列中的每个数据值是唯一的。
- 数据类型是不收集其统计信息的那种类型。即，该列是使用长整数字段或大对象数据类型定义的。
- 对于分位数，该列中只有一个非 NULL 的值。

对于索引的第一列，分布统计信息是准确的。对于每个附加的列，数据库管理程序使用散列和抽样技术来估计分布统计信息，因为计算准确的统计信息将需要太多的时间和内存。这些技术是已接受的统计方法，具有已接受的准确度。

可以通过更新 SYSSTAT.COLDIST，并对不再需要其分布统计信息的列将所有 COLVALUE 和 VALCOUNT 值设置为 0 或 -1 来除去分布统计信息。

下列主题提供的信息有助于您了解和使用这些分布统计信息:
- 了解分布统计信息。
- 何时应使用分布统计信息？
- 您应保留多少个统计信息？
- 优化器如何使用分布统计信息？
- 建立生产数据库的模型。
- 更新列分布统计信息的规则。

了解分布统计信息

对于一个固定的数 N>=1，一列中 *N 个最常用的值*由具有最高频率（即，重复次数）的数据值、具有第二个最高频率的数据值、依此类推，一直到具有第 N 个最高频率的数据值组成。对应的*高频值统计信息*由这 ″N″ 个数据值以及该列中这些值的频率组成。

一列的 *K 分位数*是最小的数据值 V，即至少有 ″K″ 行具有小于或等于 V 的数据值。可以将该列中的所有行按数据值递增的次序来排序，以计算 K 分位数；K 分位数是已排序的列中第 K 行中的数据值。

例如，考虑如下一列数据:

```
 C1
--
 B
 E
 Y
 B
 F
 G
 E
 A
```

```
J
K
E
L
```

可以将此列排序，以获得下列定序的值:

```
C1'
--
 A
 B
 B
 E
 E
 E
 F
 G
 J
 K
 L
 Y
```

在列 C1 中有九个相异的数据值。对于 N = 2，高频值统计信息是:

```
SEQNO    COLVALUE     VALCOUNT
-----    ---------    --------
  1          E            3
  2          B            2
```

如果要收集的分位数数目是 5（参见第376页的『列的分位数数目(num_quantiles)』），则对于 K = 1、3、6、9 和 12，此列的 K 分位数是:

```
SEQNO    COLVALUE     VALCOUNT
-----    ---------    --------
  1          A            1
  2          B            3
  3          E            6
  4          J            9
  5          Y           12
```

在此示例中，分位数 6 等于 E，因为已排序的列中的第六行具有等于 E 的数据值（且原始列中有 6 行具有小于或等于 E 的数据值）。

同一个分位数值可出现多次（如果它是一个高频值）。将存储给定的值的最大两个分位数。这两个分位数中的第一个具有 COLCOUNT，它提供严格小于 COLVALUE 的行的数目，而这两个分位数中的第二个提供小于或等于 COLVALUE 的行的数目。

何时应使用分布统计信息?

要决定是否应保留给定表的分布统计信息，应考虑两个因素:

1. 静态或动态 SQL 的使用。

   不使用主变量的动态 SQL 和静态 SQL 的分布统计信息最有用。当使用具有主变量的 SQL 时，优化器只能有限地利用分布统计信息。

2. 数据分布不太均匀。

   如果该表中至少有一列的数据分布非常“不均匀”，且该列频繁出现在等式或范围谓词中；即，类似如下所示的子句中，则保留分布统计信息是明智的:

   ```
   WHERE C1 = KEY;
   WHERE C1 IN (KEY1, KEY2, KEY3);
   WHERE (C1 = KEY1) OR (C1 = KEY2) OR (C1 = KEY3);
   WHERE C1 <= KEY;
   WHERE C1 BETWEEN KEY1 AND KEY2;
   ```

   在一个数据分布中可能有两类不均匀情况，它们可能一起发生:

   - 一种不均匀发生在当数据未均匀地分布在最高和最低数据值之间，而是群集在某个小间隔中时，如下面一列，在该列中，数据群集在范围 (5,10) 之间:

     ```
       C1
     -----
       0.0
       5.1
       6.3
       7.1
       8.2
       8.4
       8.5
       9.1
      93.6
     100.0
     ```

     当这种不均匀存在时，保留分位数很有用。

     以下示例显示一个查询，可使用它来帮助确定一个列中是否存在高度不均匀。

     ```
     SELECT C1, COUNT(*) AS OCCURRENCES
       FROM T1
     GROUP BY C1
     ORDER BY OCCURRENCES DESC;
     ```

   - 另一种不均匀发生在特定的数据值具有比其他数据值高得多的频率时，如某一列具有下列频率的数据值:

     ```
     数据值       频率
     ----------   ---------
         20           5
         30          10
         40          10
     ```

```
50          25
60          25
70          20
80           5
```

当这种不均匀存在时，保留分位数和高频值统计信息很有用。

通过在 RUNSTATS 命令上指定 WITH DISTRIBUTION 子句；或当调用 RUNSTATS API 时为 statsopt 参数指定 D、E 或 A，来收集分布统计信息。有关应用程序设计接口的详情，参考 *Administrative API Reference* 手册。

您应保留多少个统计信息？

保留大量列分布统计信息可使优化器更好地选择存取方案，但是，收集这些统计信息和编译查询的成本也相应增加。统计信息堆的大小（参见第306页的『统计堆大小 (stat_heap_sz)』）将限制可计算和存储的统计信息的数目。

当请求分布统计信息时，数据库管理程序要存储一列的 10 个最常用的值的缺省值。保留 10 到 100个高频值应足以应付大多数实际情况。理想情况下，应保留足够的高频值统计信息，以便其余值的频率可近似等于最高频值的频率，或者相比之下忽略不计。

要设置将收集的高频值的数目，使用 *num_freqvalues* 配置参数，如第375页的『保存的高频值数目 (num_freqvalues)』中所述。数据库管理程序收集的高频值统计信息的数目可能低于此数，因为只对出现多次的数据值收集这些统计信息。如果只收集分位数统计信息，可将此参数设置为零。

当请求分布统计信息时，数据库管理程序要存储一列的 20 个分位数的缺省值。此值保证对于任何简单的单方范围谓词（>、>=、< 或 <=）的最大估计错误率为 2.5%，而对于任何 BETWEEN 谓词的最大错误率为 5%。确定分位数数目的经验方法为:

- 确定在估计任何范围查询的行数时可允许的最大错误百分比，即 P
- 分位数数目在谓词是 BETWEEN 谓词时应近似为 100/P，在谓词是任何其他类型的范围谓词（<、<=、> 或 >=）时应为 50/P。

例如，25 个分位数导致的最大估计错误对于 BETWEEN 谓词应为 4%，而对于 ″>″ 谓词则应为 2%。通常，应保留至少 10 个分位数，对于极端不均匀的数据才需要 50 个以上的分位数。

要设置分位数数目，使用 *num_quantiles* 配置参数，如第376页的『列的分位数数目 (num_quantiles)』中所述。如果只收集高频值统计信息，可将此参数设置为零。将此参数设置为 ″1″ 也不会导致收集分位数统计信息，因为整个范围的值将适合一个分位数。

优化器如何使用分布统计信息?

为什么要收集和存储分布统计信息？答案是优化器需要估计一列中满足等式或范围谓词的行数以选择成本最低的存取方案。估计越准确，优化器将选择最优存取方案的可能性就越大。例如，考虑如下查询

```
SELECT C1, C2
  FROM TABLE1
  WHERE C1 = 'NEW YORK'
  AND C2 <= 10
```

并假定 C1 上有一个索引，C2 上也有一个索引。一个可能的存取方案是使用 C1 上的索引来检索 C1 = 'NEW YORK' 的所有行，然后检查每个检索的行以了解是否 C2 <= 10。一个替代的存取方案是使用 C2 上的索引来检索 C2 <= 10 的所有行，然后检查每个检索的行以了解是否 C1 = 'NEW YORK'。通常，执行以上查询的主要成本是检索那些行的成本，因此期望选择需要最少数目的检索的方案。要选择最佳方案，需要估计满足每个谓词的行的数目。

当您未请求分布统计信息时，优化器只维护列的下列信息：第二个最高的数据值 (HIGH2KEY)、第二个最低的数据值 (LOW2KEY)、相异值数目 (COLCARD) 和行数 (CARD)。然后假定一列中的数据值的频率全部相等，而且数据值均匀地分布在整个间隔 (LOW2KEY, HIGH2KEY) 中，以此为前提估计满足等式或范围谓词的行的数目。特别地，满足等式谓词 C1 = KEY 的行数估计为 CARD/COLCARD，而满足范围谓词 C1 BETWEEN KEY1 AND KEY2 的行数估计为:

```
      KEY2 - KEY1
------------------  x CARD       (1)
 HIGH2KEY - LOW2KEY
```

仅当一列中数据值的真实分布相当均匀时，这些估计才准确。如果分布统计信息不可用，且数据值的频率彼此之间相差很大，或数据值群集在间隔 (LOW_KEY,HIGH_KEY) 中的几个小间隔内，则估计可能偏差很大，且优化器选择的存取方案可能不是最优的。

当分布统计信息可用时，以上描述的错误可通过以下方法来大大减少：使用高频值统计信息来计算满足等式谓词的行数，使用高频值统计信息和分位数来计算满足范围谓词的行数。

对等式谓词的影响示例:

首先考虑格式为 C1 = KEY 的谓词。如果 KEY 是 N 个最高频值之一，则优化器只需使用存储在该目录中的 KEY 的频率。如果 KEY 不是 N 个最高频值之一，则优化器在假定 (COLCARD - N) 个非高频值具有均匀分布的前提下估计满足该谓词的行数。即，估计行数为:

```
CARD - NUM_FREQ_ROWS
--------------------          (2)
   COLCARD - N
```

其中，NUM_FREQ_ROWS 是具有 N 个最高频值之一的行的总数。

例如，考虑具有如下数据值频率的一列 (C):

数据值	频率
1	2
2	3
3	40
4	4
5	1

假定仅基于最高频值（即，N = 1）的高频值统计信息可用。对于此列，CARD = 50，COLCARD = 5。对于谓词 C = 3，有整 40 行满足它。假定数据分布均匀，满足该谓词的行数估计为 50/5 = 10，错误率为 -75%。在使用高频值统计信息的情况下，该行数估计为 40，且没有错误。

类似地，有 2 行满足谓词 C = 1。如果不使用高频值统计信息，则满足该谓词的行数估计为 10，错误率为 400%。您可使用以下公式来计算估计错误率（百分比）：

```
      估计行数 - 实际行数
----------------------------  X 100
         实际行数
```

使用高频值统计信息 (N = 1)，优化器将使用以上给出的公式 (2) 来估计包含此值的行数，例如:

```
(50 - 40)
--------- = 3
 (5 - 1)
```

并大大降低了错误，如下所示:

```
 3 - 2
------- = 50%
   2
```

可以使用分位数来估计满足一个范围谓词的行数，如以下示例所示。考虑下面一列 (C):

```
   C
-------
   0.0
   5.1
   6.3
   7.1
```

```
  8.2
  8.4
  8.5
  9.1
 93.6
100.0
```

并假定对于 K = 1、4、7 和 10，有可用的 K 分位数:

```
 K      K 分位数
---  ----------
 1       0.0
 4       7.1
 7       8.5
10     100.0
```

首先考虑谓词 C <= 8.5。对于以上给出的数据，有整 7 行满足此谓词。假定数据分布均匀并使用上面的公式 (1)，用 LOW2KEY 置换 KEY1，则满足该谓词的行数估计如下:

```
 8.5 - 5.1
---------- x 10 *= 0
93.6 - 5.1
```

其中，*= 表示“近似相等”。此估计中的错误率近似为 -100%。

使用分位数，可通过如下方法来估计满足此同一谓词 (C <= 8.5) 的行数: 查找作为 K 分位数之一的 8.5，并使用 K 的对应值 7 来作为估计值。在这种情况下，错误率降低至 0。

现在考虑谓词 C <= 10。有整 8 行满足此谓词。与上一个示例不同，值 10 不是存储的 K 分位数之一。假定数据分布均匀并使用公式 (1)，则满足该谓词的行数估计为 1，且错误率为 -86%。

使用分位数，优化器估计满足该谓词的行数为 r_1 + r_2，其中，r_1 是满足谓词 C <= 8.5 的行数，而 r_2 是满足谓词 C > 8.5 AND C <= 10 的行。在以上示例中，r_1 = 7。要估计 r_2，优化器使用线形插值:

```
       100.0 - 10.0
r_2 *= ------------ x (具有 > 8.5 且 <= 100.0 的值的行数)
       100.0 - 8.5
       100.0 - 10.0
     = ----------- x (10 - 7)
       100.0 - 8.5
    *= 3
```

最后估计是 r_1 + r_2 *= 10，且绝对误差降低到原来的三分之一以下。

在以上示例中使用分位数提高了估计的准确性的原因是，实数据值“群集”在范围 5-10 之间，但标准估计公式假定数据值均匀地分布在 0 和 100 之间。

当不同数据值的频率有很大差异时，使用分位数也可提高准确性。考虑具有如下数据值频率的一列:

```
   数据值          频率
----------    ---------
   20              5
   30              5
   40             15
   50             50
   60             15
   70              5
   80              5
```

假定对于 K = 5、25、75、95 和 100，有可用的分位数:

```
 K      K 分位数
----   ----------
  5       20
 25       40
 75       50
 95       70
100       80
```

也假定根据 3 个最常用的值可获得高频值统计信息。

考虑谓词 C BETWEEN 20 AND 30。从数据值的分布，您可看到有整 10 行满足此谓词。假定数据分布均匀并使用公式 (1)，则满足该谓词的行数估计如下:

```
30 - 20
-------  x 100 = 25
70 - 30
```

它的错误率为 150%。

使用高频值统计信息和分位数，满足该谓词的行数估计为 r_1 + r_2，其中，r_1 是满足谓词 (C = 20) 的行数，而 r_2 是满足谓词 C > 20 AND C <= 30 的行数。使用公式 (2)， r_1 估计如下:

```
100 - 80
-------- = 5
 7 - 3
```

使用线形插值，r_2 估计如下:

```
  30 - 20
  ------- x （具有 > 20 且 <= 40 的值的行数）
  40 - 20
  30 - 20
= ------- x (25 - 5)
  40 - 20
= 10,
```

得到最终估计值 15，并使错误率降低到原来的三分之一。

收集和使用详细的索引统计信息

作为一个选项，您可收集有关索引的更详细的统计信息，以帮助优化器更好地估计使用该索引存取一个表的成本。可通过以下两种方式之一来完成：第一种方式，可在 RUNSTATS 命令上使用 DETAILED 子句；第二种方式，可在调用 RUNSTATS API 时为 statsopt 参数指定 A、Y 或 X。仅当表有足够的大小（大约 25 页）时，才收集 DETAILED 统计信息 PAGE_FETCH_PAIRS 和 CLUSTERFACTOR。在此示例中，CLUSTERFACTOR 将是在 0 和 1 之间的一个值；而 CLUSTERRATIO 将是 -1（不收集）。对于少于 25 页的表，即使为该表上的索引指定了 DETAILED子句，CLUSTERFACTOR 仍将是 -1（不收集），而 CLUSTERRATIO 仍将是 0 和 100 之间的一个值。

了解详细的索引统计信息

DETAILED 统计信息试图以一种简洁的方式来捕捉，当在不同的缓冲区大小下执行一个完整的索引扫描时，存取一个表的数据页将需要的物理 I/O 的数目。当 RUNSTATS 扫描该索引的页时，它将为不同的缓冲区大小建立模型，并收集一个页故障发生频率的估计值。例如，对于只有一个缓冲区页可用的情况，该索引每次引用新页都会产生一个页故障，还有一种恶劣的情况是，每一行可能引用不同的页，从而导致最大的 CARDINALITY I/O。另一个极端情况是，当该缓冲区大得足以容纳整个表（但不能超过最大缓冲区大小）时，该表的每个 NPAGES 页都将实际只读取一次。因此，物理 I/O 的数目必须是缓冲区大小的单调非递增函数。

RUNSTATS 为这些估计值模拟出一个分段线性曲线，该曲线作为 11 对字符串存储在 PAGE_FETCH_PAIRS 统计信息中。每一对中的第一个值是假设的缓冲区大小，每一对中的第二个值是在该索引的一个完整扫描中读取数据页所需的物理 I/O 的估计数，而且前提是具有该大小的一个缓冲区全部可用于该索引扫描。然后优化器使用 PAGE_FETCH_PAIRS 统计信息来估计在使用该索引的任何完整或部分索引扫描中，读取数据页所用的物理 I/O 的数目。

存储在一个索引的 PAGE_FETCH_PAIRS 中的曲线的形状将取决于该索引的群集行为。

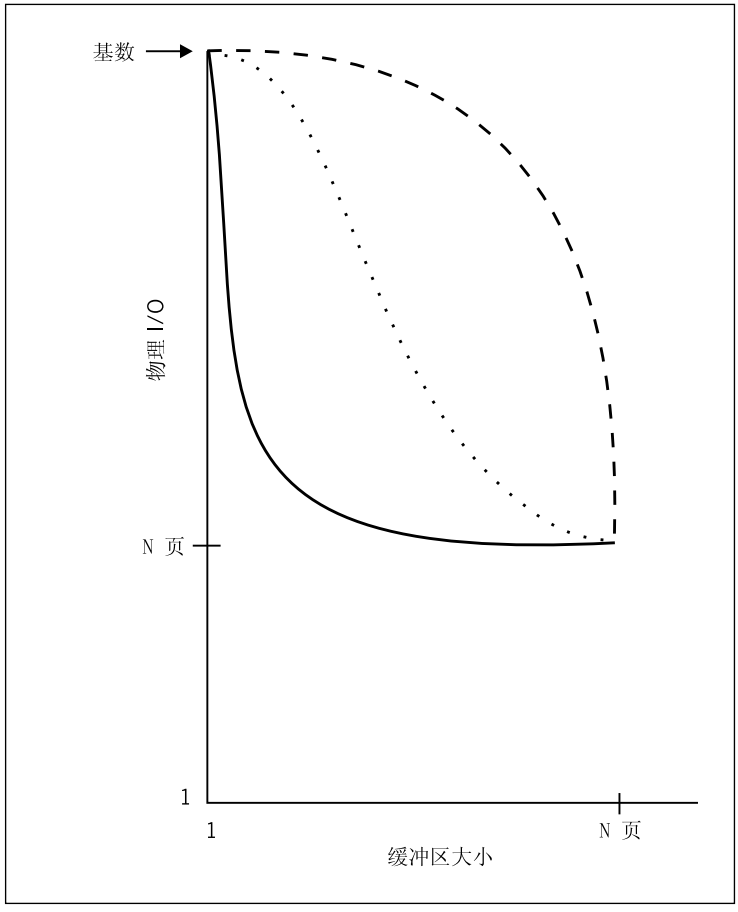


图 11. 群集索引和非群集索引的三条曲线

有三种可能的曲线:

1. 曲线 1（虚线）是高度非群集索引，在缓冲区中找到重新引用的页之前，它需要几乎与该表一样大的一个缓冲区。这表示这样一种情况，即对同一页的引用广泛地分布在该索引的关键字值中，因此一个中等大小的缓冲区不足以避免对同一页重新引用多次。这是最差的方案，因为它需要最大的缓冲区空间才能执行得很好。优化器可对这类索引使用列表预读取存取策略，来尝试群集该索引的合格关键字值所进行的数据页存取。如果此索引使用得很频繁，它应是用于重组的主要候选者。
2. 曲线 2（实线）的本地非群集性更明显。对于很小的缓冲区，它与曲线 1 一样是非群集的，但是一旦有少量的缓冲区页可用于包含最新引用的数据，则数据

页命中率会显著提高。这种情况令人有些欣慰，尽管该索引的群集率并不是特别高，但对相同数据页的引用在该索引的所有关键字值中显得彼此之间很接近。

3. 曲线 3（虚线）介于这两种极端情况之间，当缓冲区增大时，它以均匀的速率增加。对于非群集索引，这种情况通常更常见，表示当不存在 DETAILED 索引时优化器将假定什么值。

何时应使用详细的索引统计信息？

当您的查询引用并不全在该索引中的列时，应使用 DETAILED 索引统计信息。此外，当发生下列情况时，也应使用 DETAILED 索引统计信息:

- 有多个具有不同分组级别的非群集索引
- 各关键字值的群集程度不均匀
- 该索引中的值被不均匀地更新。

如果不具有以上知识，不尝试强制在不同缓冲区大小下进行索引扫描，也不使用监控程序来观察产生的物理 I/O，则可能很难确定这些情况。确定其中任何一个情况是否发生的成本最低的方法可能是，收集一个索引的 DETAILED 统计信息，如果得到的 PAGE_FETCH_PAIRS 是非线性的，则保留它们。

用户可更新的目录统计信息

更新所选择的系统目录统计信息的能力允许您:

- 使用生产系统统计信息为一个开发系统上的查询性能建立模型
- 执行 ″what if″ 查询性能分析。

不应更新一个生产系统上的统计信息，因为您可能阻止优化器为查询查找最佳存取方案。

要更新这些统计列的值，对在 SYSSTAT 模式中定义的视图使用 SQL UPDATE 语句。您可以更新下列项目的统计信息:

- 您对其持有显式 CONTROL 特权的表。也可更新这些表的列和索引的统计信息。
- 在联合体数据库系统中，您对其持有显式 CONTROL 特权的别名。也可更新这些别名的列和索引的统计信息。注意，更新仅影响本地元数据（数据源表统计信息不变）。这些更新只影响由 DB2 优化器生成的全局存取策略。
- 您拥有的用户定义函数 (UDF)（有关指南，参见第116页的『更新用户定义函数的统计信息』）。

如果您的用户 ID 对该数据库具有显式 DBADM 权限，也可更新这些统计信息；即，您的用户 ID 被记录为在 SYSCAT.DBAUTH 表中具有 DBADM 权限。属于一个 DBADM 组，并不会显式提供此权限。

使用这些视图，一个 DBADM 可查看所有用户的统计信息。没有 DBADM 权限的用户只能查看某些行，这些行包含用户具有 CONTROL 特权的对象的统计信息。

下面显示更新 EMPLOYEE 表的表统计信息的一个示例:

```
UPDATE SYSSTAT.TABLES
SET  CARD   = 10000,
     NPAGES = 1000,
     FPAGES = 1000,
     OVERFLOW = 2
WHERE TABSCHEMA = 'userid'
  AND TABNAME   = 'EMPLOYEE'
```

当更新目录统计信息时，您必须小心。任意的更新可对后续查询的性能带来严重的影响。您可能希望使用下列任何一种方法来置换应用于这些表的任何更新:

- ROLLBACK 进行了更改的工作单元（假定该工作单元未落实）。
- 使用 RUNSTATS 实用程序，您可以重新计算并刷新目录统计信息。
- 更新目录统计信息，以指示未收集统计信息。（例如，将列 NPAGES 设置为 -1，指示未收集页数统计信息。）
- 用更新之前它们包含的数据来置换目录统计信息。仅当您使用了 *db2look* 工具（如第118页的『建立生产数据库的模型』中所述）来捕捉您进行任何更改前的统计信息时，才可使用此方法。

在某些情况下，优化器可能确定某个特定的统计值或统计值的组合无效，这时它将使用缺省值并发出警告。但是，这类情况很少，因为该验证的大部分工作是在更新统计信息时执行的。

其他信息: 有关更新目录统计信息的资料，参见:

- 『更新目录统计信息的规则』
- 第113页的『更新表和别名统计信息的规则』
- 第114页的『更新列统计信息的规则』
- 第114页的『更新列分布统计信息的规则』
- 第115页的『更新索引统计信息的规则』
- 第116页的『更新用户定义函数的统计信息』
- 第118页的『建立生产数据库的模型』。

更新目录统计信息的规则

当您更新目录统计信息时，最重要的通用规则是确保各种统计信息的有效值、范围和格式存储在统计信息视图中。保持各种统计信息之间的关系的一致性也很重要。

例如，SYSSTAT.COLUMNS 中的 COLCARD 必须小于 SYSSTAT.TABLES 中的 CARD（一列中的相异值的数目不能大于行数）。假定您想把 COLCARD 从 100减

少到 25，把 CARD 从 200 减少到 50。如果首先更新 SYSCAT.TABLES，会收到一个错误（因为 CARD 应小于 COLCARD）。正确的次序是首先更新 SYSCAT.COLUMNS 中的 COLCARD，然后更新 SYSSTAT.TABLES 中的 CARD。如果您想把 COLCARD 从 100 增加到 250，把 CARD 从 200 增加到 300，则发生相反的情况。在这种情况下，必须首先更新 CARD，然后更新 COLCARD。

当检测到一个已更新的统计信息和另一个统计信息之间存在冲突时，则发出出错通知。但是，并不是在发生冲突时，始终都会发出出错通知。在某些情况下，该冲突难于检测及在出错通知中报告，尤其是两个相关的统计信息在不同目录中时更是如此。鉴于这个原因，您应小心避免导致这类冲突。

在更新目录统计信息前，您应执行的最常见的检查是:

1. 数字统计信息必须是 -1 或大于等于零。
2. 表示百分比的数字统计信息（例如，SYSSTAT.INDEXES 中的 CLUSTERRATIO）必须在 0 和 100之间。

注: 对于行类型，子表的表级统计信息 NPAGES、FPAGES 和 OVERFLOW 是不可更新的。

更新表和别名统计信息的规则

在 SYSTAT.TABLES 中只有四种统计信息值您可以更新: CARD、 FPAGES、NPAGES 和 OVERFLOW。注意:

1. CARD 必须大于对应于该表的 SYSSTAT.COLUMNS 中的所有 COLCARD 值。
2. CARD 必须大于 NPAGES。
3. FPAGES 必须大于 NPAGES。
4. NPAGES 必须小于或等于任何索引的 PAGE_FETCH_PAIRS 列中的任何“读取”值（假定此统计信息与该索引相关）。
5. CARD 不能小于或等于任何索引的 PAGE_FETCH_PAIRS 列中的任何“读取”值（假定此统计信息与该索引相关）。

当在联合体数据库系统中工作时，在人工提供／更新有关远程视图的别名统计信息时务必小心。统计信息（如此别名将返回的行数）可能不能反映评估此远程视图的实际成本，这样就可能误导 DB2 优化器。可从统计信息更新受益的情况包括在某单个基表上定义的远程视图，且未在 SELECT 列表上应用任何列函数。复杂的视图可能需要一个复杂的调整过程，该过程可能要求调整每个查询。考虑创建基于别名的本地视图，以便 DB2 优化器知道如何更准确地推导出该视图的成本。

更新列统计信息的规则

当您更新 SYSSTAT.COLUMNS 中的统计信息时，遵循以下准则。有关更新列分布统计信息的详情，参见『更新列分布统计信息的规则』。

1. HIGH2KEY 和 LOW2KEY（在 SYSSTAT.COLUMNS 中的）必须遵守下列规则:
   - 任何 HIGH2KEY、LOW2KEY 值的数据类型必须对应于此统计信息所属的用户列的数据类型。因为 HIGH2KEY 是 VARCHAR 列，您必须将该值用双引号引起来。例如，要把 INTEGER 用户列的 HIGH2KEY 设置为 25，您的更新语句应包括 SET HIGH2KEY = '25'。
   - HIGH2KEY、LOW2KEY 值的长度必须是 33 或目标列的数据类型的最大长度这两者中的较小者
   - 无论何时在对应的列中有 3 个或更多的相异值时， HIGH2KEY 必须大于 LOW2KEY。对于列中的相异值少于 3 个的情况，HIGH2KEY 可以等于 LOW2KEY。
2. 一列的基数（SYSSTAT.COLUMNS 中的 COLCARD 统计信息）不能大于其对应的表的基数（SYSSTAT.TABLES 中的 CARD 统计信息）。
3. 一列的基数（SYSSTAT.COLUMNS 中的 NUMNULLS 统计信息）不能大于其对应的表的基数（SYSSTAT.TABLES 中的 CARD 统计信息）。
4. 具有如下数据类型的列不支持统计信息: LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DBCLOB。

更新列分布统计信息的规则

第111页的『用户可更新的目录统计信息』提供有关如何更新目录统计信息的一般信息。您可能希望在试图更新列分布统计信息之前参考该节。

为了使目录中的所有统计信息一致，您必须在更新分布统计信息时养成小心的习惯。尤其是对于每一列，常用数据统计信息和分位数的目录项必须满足下列约束:

1. 高频值统计信息（在 SYSSTAT.COLDIST 目录中的）
   - 列 VALCOUNT 中的值必须随 SEQNO 值的递增而不递增。
   - 列 COLVALUE 中的值的数目必须小于或等于该列中的相异值数目，相异值数目存储在目录视图 SYSSTAT.COLUMNS 的列 COLCARD 中。
   - 列 VALCOUNT 中的值的总和必须小于或等于该列中的行数，行数存储在目录视图 SYSSTAT.TABLES 的列 CARD 中。
   - 在大多数情况下，列 COLVALUE 中的值应位于该列的第二个最高和第二个最低的数据值之间，这两个值分别存储在目录视图 SYSSTAT.COLUMNS 的列 HIGH2KEY 和 LOW2KEY 中。可能有一个高频值大于 HIGH2KEY，且有一个高频值小于 LOW2KEY。
2. 分位数（在 SYSSTAT.COLDIST 目录中）

- 列 COLVALUE 中的值必须随 SEQNO 值的递增而不递减
- 列 VALCOUNT 中的值必须严格地随 SEQNO 值的递增而递增
- 列 COLVALUE 中的最大值必须在列 VALCOUNT 中有对应的一项，它等于该列中的行数
- 在大多数情况下，列 COLVALUE 中的值应位于该列的第二个最高和第二个最低的数据值之间，这两个值分别存储在目录视图 SYSSTAT.COLUMNS 的列 HIGH2KEY 和 LOW2KEY 中。

假定具有 ″R″ 行的列 C1 的分布统计信息可供使用，而您希望修改该统计信息以对应于具有相同的数据值相对比例、但具有 ″(F x R)″ 行的列。要将高频值统计信息上调 F 倍，列 VALCOUNT 中的每一项必须乘以 F。类似地，要将分位数上调 F 倍，列 VALCOUNT 中的每一项必须乘以 F。如果不遵循这些规则，则优化器在您运行该查询时可能使用错误的过滤因子，这将导致无法预测的性能。

更新索引统计信息的规则

当您更新 SYSSTAT.INDEXES 中的统计信息时，遵循下面描述的规则:

1. PAGE_FETCH_PAIRS（在 SYSSTAT.INDEXES 中）必须遵守下列规则:
   - PAGE_FETCH_PAIRS 统计信息中的个别值必须由一系列空白定界符分开。
   - PAGE_FETCH_PAIRS 统计信息中的个别值不能超过 10 位，且必须小于最大整数值 (MAXINT = 2147483647)。
   - 如果 CLUSTERFACTOR 大于零，必须始终有一个有效的 PAGE_FETCH_PAIRS 值。
   - 在单个的 PAGE_FETCH_PAIR 统计信息中必须有整 11 对数。
   - PAGE_FETCH_PAIRS 的缓冲区大小项的值必须呈升序。
   - 如果缓冲区大小的值与上一对中的该值相同，则页读取值也必须与上一对中的该值相同。
   - 一个 PAGE_FETCH_PAIRS 项中的任何缓冲区大小值不能大于 MIN(NPAGES, 524287)，其中，NPAGES 是对应表中的页数（在 SYSSTAT.TABLES 中）。
   - PAGE_FETCH_PAIRS 的 “读取” 项的值必须呈降序，且任何个别的 “读取” 项不能小于 NPAGES。PAGE_FETCH_PAIRS 项中的 “读取” 大小值不能大于对应表的 CARD（基数）统计信息。
   - 如果缓冲区大小值在两个连续的对中相同，则页读取值也必须在两个对中相同（在 SYSSTAT.TABLES 中）。

   一个有效的 PAGE_FETCH_UPDATE 是:

   ```
   PAGE_FETCH_PAIRS =
    '100 380 120 360 140 340 160 330 180 320 200 310 220 305 240 300
     260 300 280 300 300 300'
   ```

   其中

```
NPAGES = 300
CARD   = 10000
CLUSTERRATIO  =  -1
CLUSTERFACTOR = 0.9
```

2. CLUSTERRATIO 和 CLUSTERFACTOR（在 SYSSTAT.INDEXES 中）必须遵守下列规则:
   - CLUSTERRATIO 的有效值是 -1 或在 0 和 100 之间。
   - CLUSTERFACTOR 的有效值是 -1 或在 0 和 1 之间。
   - CLUSTERRATIO 和 CLUSTERFACTOR 值中至少一个必须始终是 -1。
   - 如果 CLUSTERFACTOR 为正数，则它必须伴有一个有效的 PAGE_FETCH_PAIR 统计信息。
3. 下列规则适用于 FIRSTKEYCARD、FIRST2KEYCARD、FIRST3KEYCARD、FIRST4KEYCARD 和 FULLKEYCARD:
   - 对于单列索引，FIRSTKEYCARD 必须等于 FULLKEYCARD。
   - 对于对应的列，FIRSTKEYCARD 必须等于 COLCARD。
   - 如果其中任何一个索引统计信息是无关的信息，应将它们设置为 -1。例如，如果您有一个只有 3 列的索引，则将 FIRST4KEYCARD 设置为 -1。
   - 对于多列索引，如果所有统计信息是相关的，则它们之间的关系必须是:

     ```
     FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
       <= FULLKEYCARD <= CARD
     ```
4. 下列规则适用于 SEQUENTIAL_PAGES 和 DENSITY:
   - SEQUENTIAL_PAGES 的有效值是 -1 或在 0 和 NLEAF 之间。
   - DENSITY 的有效值是 -1 或在 0 和 100 之间。

更新用户定义函数的统计信息

使用 SYSSTAT.FUNCTIONS 目录视图，您可以更新用户定义函数 (UDF) 的统计信息。如果这些统计信息是可用的，该优化器将在估计各种存取方案的成本时使用它们。如果统计信息不可用，则统计信息列的值将是 -1，且优化器将使用缺省值，这些缺省值假定一个简单的 UDF。

下表提供有关您可为 UDF 更新的统计信息列的信息:

表 13. 函数统计信息（SYSCAT.FUNCTIONS 和 SYSSTAT.FUNCTIONS）

统计信息	说明
IOS_PER_INVOC	每次执行一个函数时所执行的读 / 写请求的估计数目。
INSTS_PER_INVOC	每次执行一个函数时所执行的机器指令的估计数目。
IOS_PER_ARGBYTE	对每个输入自变量字节所执行的读 / 写请求的估计数目。

表 13. 函数统计信息（SYSCAT.FUNCTIONS 和 SYSSTAT.FUNCTIONS） (续)

统计信息	说明
INSTS_PER_ARGBYTES	对每个输入自变量字节所执行的机器指令的估计数目。
PERCENT_ARGBYTES	该函数将实际处理的输入自变量字节的估计平均百分比。
INITIAL_IOS	仅在第一次／最后一次调用函数时所执行的读／写请求的估计数目。
INITIAL_INSTS	仅在第一次／最后一次调用函数时所执行的机器指令的估计数目。
CARDINALITY	一个表函数生成的估计行数。

例如，考虑一个 UDF (EU_SHOE)，它将美国鞋码转换为等效的欧洲鞋码。（这两个鞋码可以是 UDT。）对于此 UDF，应按如下所示设置统计信息列:

- 应将 INSTS_PER_INVOC 设置为执行下列操作所需的机器指令的估计数目:
  - 调用 EU_SHOE
  - 初始化输出字符串
  - 返回结果。
- 应将 INSTS_PER_ARGBYTE 设置为把输入字符串转换为欧洲鞋码所需的机器指令的估计数目。
- 应将 PERCENT_ARGBYTES 设置为 100，指示要转换整个输入字符串
- 应将 INITIAL_INSTS、IOS_PER_INVOC、IOS_PER_ARGBYTE 和 INITIAL_IOS 全部设置为 0，因为此 UDF 只执行计算。

PERCENT_ARGBYTES 将由一个不是始终处理整个输入字符串的函数使用。例如，考虑一个 UDF (LOCATE)，它接受两个自变量作为输入，并返回第一个自变量在第二个自变量中第一次出现的起始位置。假定第一个自变量的长度小得相对于第二个自变量来说变得不重要，那么平均来说要搜索第二个自变量的 75%。根据此信息，应将 PERCENT_ARGBYTES 设置为 75。以上对平均值 75% 的估计是基于下列的附加假设:

- 当有一半的时间找不到第一个自变量时，将导致搜索整个第二个自变量
- 第一个自变量在第二个自变量中任何地方出现的可能性相同，导致当找到第一个自变量时，（平均）搜索第二个自变量的一半。

可使用 INITIAL_INSTS 或 INITIAL_IOS 来记录仅在第一次或最后一次调用该函数时，所执行的机器指令或读／写请求的估计数目。这也可用于，例如，记录设置一个暂存区的成本。

要获得有关一个用户定义函数使用的 I/O 和指令的信息，您可以使用您的程序设计语言编译程序或可用于您的操作系统的监控工具所提供的输出。

建立生产数据库的模型

有时，您可能希望测试系统包含生产系统数据的一个子集。但是，为这类测试系统选择的存取方案不必与在生产系统上应选择的那些存取方案相同，除非更新该测试系统的目录统计信息和配置参数来匹配生产系统的那些参数。

提供一个生产工具 *db2look*，可对生产数据库运行它，以生成使测试数据库的目录统计信息与生产数据库的目录统计信息匹配所需的更新语句。可在模拟方式（`-m` 选项）下使用 *db2look* 来生成这些更新语句。在这种情况下，*db2look* 将生成一个命令处理器脚本，它包含要模拟生产数据库的目录统计信息所需的所有语句。在测试环境中通过 Visual Explain 来分析 SQL 语句时，它会很有用。

可以使用 *db2look -e* 来提取 DDL 语句，以重新创建数据库数据对象，包括表、视图、索引和数据库中的其他对象。可以对另一个数据库运行用此命令创建的命令处理器脚本，来重新创建该数据库。可同时使用 `-e` 选项和 `-m` 选项。

当对测试系统运行 `db2look` 产生的更新语句后，可使用该测试系统来验证要在生产中生成的存取方案。因为优化器使用表空间的类型和配置来估计 I/O 成本，因此测试系统必须有相同的表空间几何结构或布局。即，同一类型（SMS 或 DMS）的相同数目的容器。

可在 *bin* 子目录下找到 *db2look* 工具。

有关如何使用此生产工具的详情，在命令行上输入如下命令:

```
db2look -h
```

有关此工具的详情，也可参考 *Command Reference* 手册。

控制中心还为 *db2look* 实用程序提供了一个称为“生成 SQL - 对象名”的接口。使用“控制中心”可将实用程序的结果文件集成到“脚本中心”。也可从控制中心调度 *db2look* 命令。使用控制中心的一个不同之处是，这种情况下只能完成一个表分析，而使用 *db2look* 命令时，在单个调用中最多可完成三十个表的分析。还应知道控制中心不支持 LaTex 输出和图形输出。

还可以对 OS/390 数据库运行 `db2look` 实用程序。`db2look` 实用程序抽取 OS/390 对象的 DDL 和 UPDATE 统计语句。如果您希望抽取 OS/390 对象并在“DB2 通用数据库”(UDB) 数据库中重建它们，则此实用程序非常有用。参考 *Command Reference* 以了解有关 `db2look` 实用程序的其他信息。

DB2 UDB 统计信息与 OS/390 统计信息之间存在一些差异。`db2look` 实用程序执行从 DB2 OS/390 版到 DB2 UDB 的适当转换（当适用时），并将不存在其 DB2 OS/390 版对应统计信息的 DB2 UDB 统计信息设置为缺省值 (-1)。以下是 `db2look`

实用程序将 DB2 OS/390 版统计信息映射至 DB2 UDB 统计信息的方法。在下面的讨论中，『UDB_x』 表示 DB2 UDB 统计信息列，而 『S390_x』 表示 DB2 OS/390 版统计信息列。

1. 表层统计信息。

   UDB_CARD = S390_CARDF
   UDB_NPAGES = S390_NPAGES

   没有 S390_FPAGES。但是，DB2 OS/390 版有另一名为 PCTPAGES 的统计信息，它表示包含表行的活动表空间页的百分比。因而，有可能根据 S390_NPAGES 和 S390_PCTPAGES 计算 UDB_FPAGES，如下所示:

   ```
   UDB_FPAGES=(S390_NPAGES * 100)/S390_PCTPAGES
   ```

   没有 S390_OVERFLOW 来映射至 UDB_OVERFLOW。因而，db2look 实用程序只是将 UDB_OVERFLOW 设置为缺省值:

   ```
   UDB_OVERFLOW=-1
   ```

2. 列层统计信息。

   UDB_COLCARD = S390_COLCARDF
   UDB_HIGH2KEY = S390_HIGH2KEY
   UDB_LOW2KEY = S390_LOW2KEY

   没有 S390_AVGCOLLEN 来映射至 UDB_AVGCOLLEN，因而，db2look 实用程序只是将 UDB_AVGCOLLEN 设置为缺省值:

   ```
   UDB_AVGCOLLEN=-1
   ```

3. 索引层统计信息。

   UDB_NLEAF = S390_NLEAF
   UDB_NLEVELS = S390_NLEVELS
   UDB_FIRSTKEYCARD= S390_FIRSTKEYCARD
   UDB_FULLKEYCARD = S390_FULLKEYCARD
   UDB_CLUSTERRATIO= S390_CLUSTERRATIO

   其他没有 OS/390 对应统计信息的统计信息只是设置为缺省值。它们是:

   ```
   UDB_FIRST2KEYCARD = -1
   UDB_FIRST3KEYCARD = -1
   UDB_FIRST4KEYCARD  = -1
   UDB_CLUSTERFACTOR = -1
   UDB_SEQUENTIAL_PAGES = -1
   UDB_DENSITY = -1
   ```

4. 列分布统计信息。

在 DB2 OS/390 版 SYSIBM.SYSCOLUMNS 中，有两种类型的统计信息。类型 『F』表示频率值，类型 『C』 表示基数。只有类型为 『F』 的项才适用于 DB2 UDB 版，这些就是将被考虑到的项。并且，DB2 OS/390 版 SYSIBM.SYSCOLUMNS 中没有 SEQNO 列，但这是 DB2 UDB 版所必需的。因此，db2look 自动生成一个。

```
UDB_COLVALUE = S390_COLVALUE
UDB_VALCOUNT = S390_FrequencyF * S390_CARD
```

第6章 了解 SQL 编译程序

当编译一个 SQL 查询时，要执行许多步骤，然后才执行“最佳”存取方案，或将该方案存储在系统目录中。

在一个分区数据库环境中，SQL 编译程序对 SQL 查询执行的所有工作是在连接的数据库分区上进行的。在执行之前，将已编译的查询分布至该数据库中的所有数据库分区上。

下列主题提供有关 SQL 编译程序执行的步骤的详情:

- SQL 编译程序概述
- 由 SQL 编译程序重写查询
- 操作合并
- 操作移动
- 谓词转换
- 数据存取概念和优化
- 分区内并行性的优化策略
- 自动摘要表
- 联合体数据库查询编译程序阶段

下列几节也提供有关编译程序的外部因素的信息，这些因素可影响该编译程序产生的结果:

- 第37页的『第3章 应用程序考虑事项』
- 第77页的『第4章 环境考虑事项』
- 第93页的『第5章 系统目录统计信息』。

第177页的『第7章 SQL 解释设施』描述如何检查 SQL 编译程序选择的存取方案。

SQL 编译程序概述

SQL 编译程序在产生您可以执行的存取方案之前，执行几个步骤。这些步骤显示在第122页的图12中。

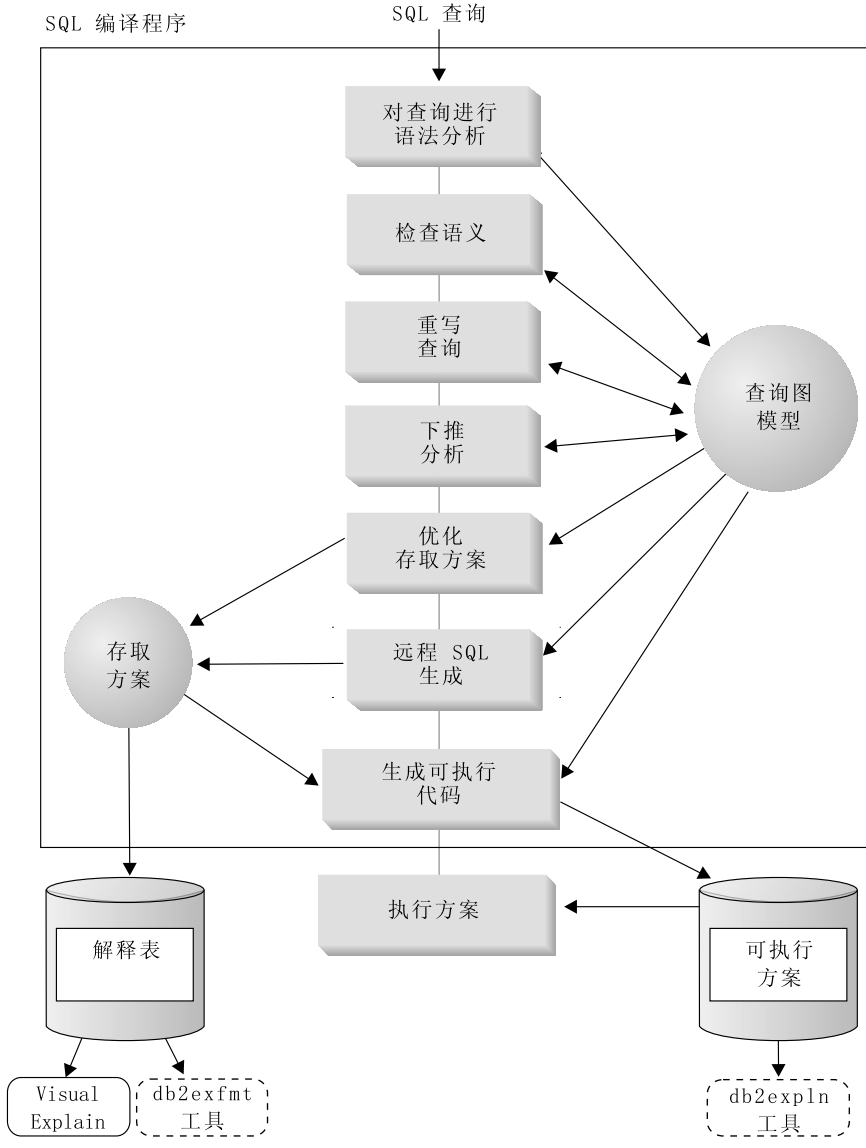


图 12. SQL 编译程序执行的步骤

此图显示**查询图模型**是 SQL 编译程序的一个关键部件。 *查询图模型*是一个内部的、驻留在内存中的数据库，它用于在整个查询编译过程中表示该查询，如下所述:

- **分析查询语法**

SQL 编译程序的第一个任务是分析 SQL 查询以验证其语法。如果检测到任何语法错误，则 SQL 编译程序停止处理，并将适当的 SQL 错误返回至试图编译该 SQL 语句的应用程序。当语法分析完成时，就会创建该查询的内部表示。

- **检查语义**

  编译程序的第二个任务是确保语句的各部分之间不存在不一致。检查语义的一个简单示例是，确保为 YEAR 标量函数指定的列的数据类型是日期时间数据类型。而且在这第二个阶段期间，编译程序将行为语义添加至该查询图模型，包括参考约束、表检查约束、触发器和视图的作用。

  该查询图模型包含查询的所有语义，包括查询块、子查询、关联、派生的表、表达式、数据类型、数据类型转换、代码页转换和分区关键字。

- **重写查询**

  SQL 编译程序的第三个阶段是使用查询图模型中提供的全局语义，将该查询转换为更易于优化的一种形式。例如，编译程序可能会移动谓词，改变其应用级别并有可能提高查询性能。这种类型的操作移动称为*一般谓词下推*。有关详情，参见第125页的『由 SQL 编译程序重写查询』。

  在一个分区数据库环境中工作时，某些查询操作的计算更为集中，象涉及到下列操作的那些查询:

  – 聚合
  – 行的重新分布
  – 关联的子查询。

  相关子查询是包含对该子查询外部表列的引用的子查询。

  在此环境中，使用某些查询时，作为重写该查询的一部分，可能发生取消关联。

  传送的查询存储在“查询图模型”中。

- **下推分析（联合体数据库）**

  此步骤的主要任务是向 DB2 优化器建议是否可在数据源处对一个操作远程求值（“下推”）。这种类型的下推活动仅适用于数据源查询，它是对一般谓词下推操作的扩展。

  除非执行联合体数据库查询，否则绕过此步骤。有关详情，参见第166页的『下推分析』。

- **优化存取方案**

  SQL 编译程序的 SQL 优化器部分使用查询图模型作为输入，并生成许多替代的执行方案来满足用户的请求。它使用有关表、索引、列和函数的统计信息来估

计每个替代方案的执行成本，并选择使用最小估计执行成本的该方案。优化器使用查询图模型来分析查询语义，并获取有关各种因素的信息，包括索引、基表、派生的表、子查询、关联和递归。

该优化器部分也可考虑第三类下推操作：*聚合与排序*，这种操作可将对这些操作求得的值下推到“数据管理服务”部件中来改进性能。有关详情，参见第160页的『聚合和排序下推运算符』。

在确定页面大小选择时，优化器还考虑是否有不同大小的缓冲池。该环境是否包括一个分区数据库，以及是否能够为在对称多处理器 (SMP) 环境中实现分区内并行性改善选择的方案，均属考虑范围。优化器使用此信息来帮助选择该查询的最佳存取方案。有关详情，参见第134页的『数据存取概念和优化』。

SQL 编译程序此步骤的输出是“存取方案”。此存取方案为解释表中捕捉的信息提供了基本资料。可以用解释快照捕捉用于生成该存取方案的信息。（有关“解释”主题的详情，参见第177页的『第7章 SQL 解释设施』。）

- **远程 SQL 生成（联合体数据库）**

  DB2 优化器选择的最后一个方案可由一组可对远程数据源执行的步骤组成。对于每个数据源将执行的那些操作，远程 SQL 生成步骤将根据数据源 SQL 语言创建一个高效的 SQL 语句。

  除非执行联合体数据库查询，否则绕过此步骤。有关详情，参见第172页的『远程 SQL 生成与全局优化』。

- **生成“可执行”代码**

  SQL 编译程序的最后一个步骤使用*存取方案*和查询图模型，为该查询创建一个可执行的存取方案或程序段。此代码生成步骤使用查询图模型中的信息，以避免重复执行只需要对一个查询计算一次的表达式。可以进行此优化的示例包括代码页转换和主变量的使用。

  有关静态 SQL 存取方案的信息存储在系统目录表中。当执行该程序包时，数据库管理程序将使用存储在系统目录表中的信息来确定如何存取该数据，并提供该查询的结果。这就是 *db2expln* 工具所使用的信息。（有关“解释”主题的详情，参见第177页的『第7章 SQL 解释设施』。）

建议在期望获得良好性能的查询中所用的表上定期执行 RUNSTATS。然后根据数据的性质，为优化器配备更适当的相关统计信息。如果未执行 RUNSTATS（或优化器怀疑在空表或几乎是空的表上执行了 RUNSTATS），则优化器可使用缺省值，或尝试根据在磁盘上存储该表所用的文件页的数目 (FPAGES)来得出特定的统计信息。

由 SQL 编译程序重写查询

SQL 编译程序包括一个查询重写阶段，它将 SQL 语句转换为更易于优化的形式，从而改善选择的存取路径。重写查询对于那些非常复杂的查询（包括具有许多子查询或许多连接的查询）特别重要。查询发生器工具通常会创建这些很复杂的查询。

可更改优化级别来影响应用于 SQL 语句的查询重写规则的数目（参见第57页的『调整优化级别』）。

可使用解释设施或 Visual Explain 来查看查询重写的一些结果。

SQL 编译程序可执行三种主要类别的重写:

- 操作合并
- 操作移动
- 谓词转换。

操作合并

SQL 编译程序将重写查询来合并查询操作，以试图构造该查询，使它具有最少的操作，尤其是 SELECT 操作。提供以下示例，以举例说明 SQL 编译程序可合并的一些操作:

- 示例 - 视图合并

  在 SELECT 语句中使用视图，可限制表的连接次序，也可能引入表的冗余连接。通过在查询重写期间合并视图，可撤消这些限制。

- 示例 - 子查询至连接的转换

  如果优化器在 SELECT 语句中找到子查询，则它在选择表的次序处理方面可能会受到限制。

- 示例 - 消除冗余连接

  在查询重写期间，可除去冗余连接，以进一步简化将优化的 SELECT 语句。

- 示例 - 共享聚合

  当使用不同的函数时，重写查询可减少需要执行的计算数。

示例 - 视图合并

假定您可存取 EMPLOYEE 表的以下两个视图，一个显示受过高等教育的雇员，另一个视图显示工资超过 $35,000 的雇员:

```
CREATE VIEW EMP_EDUCATION (EMPNO, FIRSTNME, LASTNAME, EDLEVEL) AS
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL
  FROM EMPLOYEE
```

```
 WHERE EDLEVEL > 17
CREATE VIEW EMP_SALARIES (EMPNO, FIRSTNAME, LASTNAME, SALARY) AS
SELECT EMPNO, FIRSTNME, LASTNAME, SALARY
  FROM EMPLOYEE
 WHERE SALARY > 35000
```

现在，假定您执行以下查询，来列出既受过高等教育而且工资又超过 $35,000 的雇员:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMP_EDUCATION E1,
       EMP_SALARIES  E2
 WHERE E1.EMPNO = E2.EMPNO
```

在查询重写期间，可合并这两个视图来创建以下查询:

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

通过使用用户编写的 SELECT 语句来合并这两个视图中的 SELECT 语句，优化器在选择存取方案时可考虑更多选项。此外，如果已合并的这两个视图使用相同的基表，还可执行重写，如『示例 - 消除冗余连接』中所述。

示例 - 子查询至连接的转换

SQL 编译程序将执行一个包含子查询的查询，如:

```
SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO
FROM EMPLOYEE
WHERE WORKDEPT IN
      (SELECT DEPTNO
         FROM DEPARTMENT
        WHERE DEPTNAME = 'OPERATIONS')
```

并将它转换为如下形式的连接查询:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME, PHONENO
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNAME = 'OPERATIONS'
```

一般情况下，连接的执行效率比子查询高很多。

示例 - 消除冗余连接

有时编写或生成的查询可能没有需要的连接。在查询重写阶段期间，也可能产生类似如下所示的查询，如第125页的『示例 - 视图合并』中所述。

```
SELECT E1.EMPNO, E1.FIRSTNME, E1.LASTNAME, E1.EDLEVEL, E2.SALARY
  FROM EMPLOYEE E1,
       EMPLOYEE E2
 WHERE E1.EMPNO = E2.EMPNO
   AND E1.EDLEVEL > 17
   AND E2.SALARY  > 35000
```

在此查询中，SQL 编译程序可消除该连接并将查询简化为:

```
SELECT EMPNO, FIRSTNME, LASTNAME, EDLEVEL, SALARY
  FROM EMPLOYEE
 WHERE EDLEVEL > 17
   AND SALARY  > 35000
```

另一个示例假设具有该部门号的 EMPLOYEE 和 DEPARTMENT 样本表之间存在参考约束。首先，创建一个视图。

```
CREATE VIEW PEPLVIEW
   AS SELECT FIRSTNME, LASTNAME, SALARY, DEPTNO, DEPTNAME, MGRNO
         FROM EMPLOYEE E DEPARTMENT D
         WHERE E.WORKDEPT = D.DEPTNO
```

接着，如下的查询:

```
SELECT LASTNAME, SALARY
   FROM PEPLVIEW
```

变成

```
SELECT LASTNAME, SALARY
   FROM EMPLOYEE
   WHERE WORKDEPT NOT NULL
```

注意在这种情况下，即使用户知道可重新编写查询，他们可能也无法这样做，因为他们没有基表的存取权。他们可能只有视图（上面显示的）的存取权。因此，必须在数据库管理程序内执行这种类型的优化。

只要符合以下条件，参考完整性连接中就可能存在冗余:

- 视图是通过连接定义的
- 查询是自动生成的。

  例如，查询管理器提供了一些自动化工具，它们会阻止用户编写优化的查询。

示例 - 共享聚合

在一个查询中使用多个函数可生成几次计算，这会占用时间。将要在查询中执行的计算数减少可改进方案。SQL 编译程序执行使用多个函数的一个查询，如:

```
SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  AVG(SALARY+BONUS+COMM) AS OAVG,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE;
```

用下列方式转换该查询:

```
SELECT OSUM,
  OSUM/OCOUNT
  OCOUNT
FROM (SELECT SUM(SALARY+BONUS+COMM) AS OSUM,
  COUNT(*) AS OCOUNT
FROM EMPLOYEE) AS SHARED_AGG;
```

此重写将该查询从 2 次求和 2 次计数减少为 1 次求和 1 次计数。

操作移动

SQL 编译程序将重写查询以移动查询操作，来尝试构造具有最少数目的操作和谓词的查询。提供以下示例，以举例说明 SQL 编译程序可移动的一些操作:

- 示例 - 消除 DISTINCT

  在查询重写期间，优化器可移动执行 DISTINCT 操作的位置，以减少此操作的成本。在提供的示例中，全部除去了 DISTINCT 操作。

- 示例 - 一般谓词下推

  在查询重写期间，可更改应用谓词的次序，以便将选择性更强的谓词尽早地应用于该查询。

- 示例 - 解相关

  当在一个分区数据库环境中时，在数据库分区之间移动结果集的成本很高。减小必须广播至其他数据库分区的信息的大小和 / 或减少广播次数，是重写查询的目标之一。

示例 - 消除 DISTINCT

如果 EMPNO 列被定义为 EMPLOYEE 表的主关键字，则以下查询:

```
SELECT DISTINCT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

将通过除去 DISTINCT 子句来重写:

```
SELECT EMPNO, FIRSTNME, LASTNAME
  FROM EMPLOYEE
```

在以上示例中，因为选择了主关键字，因此 SQL 编译程序知道返回的每一行已经是唯一的。在这种情况下，DISTINCT 关键字是冗余的。如果未重写该查询，则优化器将进行必要的处理（例如，排序）来构建一个方案，以确保这些列是相异的。

示例 - 一般谓词下推

改变谓词通常应用的级别可改善性能。例如，假定以下视图，它提供部门 『D11』 中所有雇员的列表:

```
CREATE VIEW D11_EMPLOYEE
 (EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM)
AS SELECT EMPNO, FIRSTNME, LASTNAME, PHONENO, SALARY, BONUS, COMM
     FROM EMPLOYEE
    WHERE WORKDEPT = 'D11'
```

并假定以下查询:

```
SELECT FIRSTNME, PHONENO
  FROM D11_EMPLOYEE
 WHERE LASTNAME = 'BROWN'
```

该编译程序的查询重写阶段将把谓词 LASTNAME = 'BROWN' 下推到视图 D11_EMPLOYEE 中。这使得可以更快并可能更有效地应用该谓词。在此示例中可能执行的实际查询为:

```
SELECT FIRSTNME, PHONENO
  FROM EMPLOYEE
 WHERE LASTNAME = 'BROWN'
   AND WORKDEPT = 'D11'
```

谓词的下推并不限于视图。可以下推谓词的其他情况包括 UNION、GROUP BY 和派生的表（嵌套表表达式或公共表表达式）。

示例 - 解相关

在一个分区数据库环境中，SQL 编译程序可重写以下查询:

查找正在从事程序设计项目并被少付工资的所有雇员。

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
  FROM EMPLOYEE E, PROJECT P
 WHERE P.EMPNO = E.EMPNO
   AND P.PROJNAME LIKE '%PROGRAMMING%'
   AND E.SALARY+E.BONUS+E.COMM <
     (SELECT AVG(E1.SALARY+E1.BONUS+E1.COMM)
        FROM EMPLOYEE E1, PROJECT P1
        WHERE P1.PROJNAME LIKE '%PROGRAMMING%'
          AND P1.PROJNO = A.PROJNO
          AND E1.EMPNO = P1.EMPNO)
```

因为此查询是相关的，而且因为 PROJECT 和 EMPLOYEE 不可能根据 PROJNO 来分区，因此将每个项目广播至每个数据库分区是可能的。此外，子查询将不得不求值多次。

SQL 编译程序可以重写该查询，如下所示:

- 确定从事程序设计项目的雇员的相异值列表，并将其称为 DIST_PROJS。它必须是相异的，才能确保只对每个项目执行一次聚合:

```
WITH DIST_PROJS(PROJNO, EMPNO) AS
(SELECT DISTINCT PROJNO, EMPNO
 FROM PROJECT P1
 WHERE P1.PROJNAME LIKE '%PROGRAMMING%')
```

- 使用从事程序设计项目的雇员的相异值列表，将该列表与雇员表连接，以获得每个项目的平均赔偿额 AVG_PER_PROJ:

```
AVG_PER_PROJ(PROJNO, AVG_COMP) AS
(SELECT P2.PROJNO, AVG(E1.SALARY+E1.BONUS+E1.COMM)
 FROM EMPLOYEE E1, DIST_PROJS P2
 WHERE E1.EMPNO = P2.EMPNO
 GROUP BY P2.PROJNO)
```

- 因此新查询将是:

```
SELECT P.PROJNO, E.EMPNO, E.LASTNAME, E.FIRSTNAME,
       E.SALARY+E.BONUS+E.COMM AS COMPENSATION
 FROM PROJECT P, EMPLOYEE E, AVG_PER_PROG A
WHERE P.EMPNO = E.EMPNO
  AND P.PROJNAME LIKE '%PROGRAMMING%'
  AND P.PROJNO = A.PROJNO
  AND E.SALARY+E.BONUS+E.COMM < A.AVG_COMP
```

重写的 SQL 查询计算每个项目的 AVG_COMP (AVG_PRE_PROJ)，然后将该结果广播至包含 EMPLOYEE 表的所有数据库分区。

谓词转换

SQL 编译程序将重写查询，以便把特定查询的现存谓词转换为更优的谓词。提供以下示例，以举例说明 SQL 编译程序可转换的一些谓词:

- 示例 - 添加隐含谓词

  在查询重写期间，可以将谓词添加至查询，以允许优化器在选择该查询的最佳存取方案时考虑附加表连接。

- 示例 - OR 至 IN 的转换

  在查询重写期间，可以将一个 OR 谓词转换为一个 IN 谓词，以允许选择更有效的存取方案。SQL 编译程序也可将一个 IN 谓词转换为一个 OR 谓词，但前提是此转换使得可以选择一个更有效的存取方案。

示例 - 添加隐含谓词

以下查询产生一个其部门是向 『E01』 报告的经理的列表，以及那些经理所负责的项目的列表:

```
SELECT DEPT.DEPTNAME DEPT.MGRNO, EMP.LASTNAME, PROJ.PROJNAME
  FROM DEPARTMENT DEPT,
       EMPLOYEE   EMP,
       PROJECT    PROJ
 WHERE DEPT.ADMRDEPT = 'E01'
   AND DEPT.MGRNO = EMP.EMPNO
   AND EMP.EMPNO  = PROJ.RESPEMP
```

该查询重写将添加以下隐含的谓词:

```
DEPT.MGRNO = PROJ.RESPEMP
```

作为此重写的结果，优化器在尝试选择该查询的最佳存取方案时可以考虑附加连接。

除以上的谓词传递闭包外，查询重写还将根据等式谓词隐含的传递性派生附加的本地谓词。例如，以下查询列出部门（部门号大于 『E00』）的名称和在该部门工作的雇员的名称。

```
SELECT EMPNO, LASTNAME, FIRSTNAME, DEPTNO, DEPTNAME
  FROM EMPLOYEE EMP,
       DEPARTMENT DEPT
 WHERE EMP.WORKDEPT = DEPT.DEPTNO
   AND DEPT.DEPTNO > 'E00'
```

对于此查询，该重写阶段将添加以下隐含的谓词:

```
EMP.WORKDEPT > 'E00'
```

作为此重写的结果，优化器可减少要连接的行数。

示例 - OR 至 IN 的转换

假定一个 OR 子句根据同一列将两个或更多个简单的等式谓词相连，如以下示例所示:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO = 'D11'
    OR DEPTNO = 'D21'
    OR DEPTNO = 'E21'
```

如果 DEPTNO 列上没有索引，将 OR 子句转换为以下的 IN 谓词将允许更有效地处理该查询:

```
SELECT *
  FROM EMPLOYEE
 WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

注: 在某些情况下，数据库管理程序可以将一个 IN 谓词转换为一组 OR 子句，以便可执行索引 OR 运算。有关索引 OR 运算的详情，参见第139页的『多索引存取』。

考虑列关联

您的应用程序可能包含由连接构成的查询，这些连接具有将两个表相连的多个连接谓词。这似乎很复杂，但这种情况并不是什么异乎寻常的事情，在这种情况中，可尝试确定各表中相似及相关的列之间的关系。

例如，制造商利用各种颜色、弹性和质量的原料生产产品。制成品与所用的原料有相同的颜色和弹性。制造商发出查询:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY FROM PRODUCT, RAWMATERIAL
   WHERE PRODUCT.COLOR       =  RAWMATERIAL.COLOR
     AND PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

此查询返回所有产品的名称和原料质量。有两个连接谓词:

```
PRODUCT.COLOR       =  RAWMATERIAL.COLOR
PRODUCT.ELASTICITY  =  RAWMATERIAL.ELASTICITY
```

当 DB2 UDB 优化器选择一个方案来执行此查询时，它要计算两个谓词中每一个的选择性如何，并假定它们是独立的，即，对于每种颜色存在各种弹性；相反，对于每一级的弹性，存在每种颜色的原料。然后，它使用有关每个表中的弹性级别数和不同颜色数的统计信息，来计算谓词对的总体选择性。例如，根据此总体选择性，它可选择“嵌套循环连接”，而不是“合并连接”，反之亦然。

但是，有可能这两个谓词都不是独立的。例如，有可能只有几种颜色的高弹性材料可用，且只有其他几种颜色（与弹性材料的颜色不同）的弹性很低的材料可用。于是，谓词的组合选择性较低（排除更少的行），因此查询将返回更多行。为了明白这一点，想象最极端的情况，即对于每种颜色，只有一种级别的弹性，反之亦然。现在，两个谓词中任何一个在逻辑上可以完全省略，因为它被另一个谓词隐含。优化器选择的方案可能不再是最好的，例如，可能选择了“嵌套循环”连接方案，但“合并连接”将会更快。

对于其他数据库产品，数据库管理员已尝试通过更新目录中的统计信息以使其中一个谓词的选择性较低，来解决此性能问题，但是此方法可能对其他查询产生不期望的负面影响。

如果执行了以下操作，DB2 UDB 的优化器将尝试检测并补偿连接谓词的相关性:

1. 在关联的列（即，一个表中出现在关联谓词中的那些列）上定义唯一索引。
2. 不要将注册表变量 DB2_CORRELATED_PREDICATES 设置为 『NO』。

在以上的示例中，您可以定义涉及到:

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

或

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

的唯一索引。

为了检测关联，此索引的非包括列必须是关联的列，不能有其他列。该索引可以选择是否包含包括列。

通常，在连接谓词中可能有 2 个以上的关联列，因此，您应确保定义唯一的索引来涵盖所有关联的列。

在许多情况下，一个表中的关联列构成其主关键字。主关键字总是唯一的，因此如果相关列上有主关键字，则无需定义另一个唯一索引。

这样做之后，确保表的统计信息是最新的，且不因任何原因（例如，试图影响优化器）改变它们的真实值。

优化器将使用唯一索引统计信息的 FIRSTnKEYCARD 和 FULLKEYCARD 信息来检测关联情况，并动态调整关联谓词的组合选择性，因而获得对连接大小和成本的更准确的估计。

除了 JOIN 谓词相关之外，优化器还使用类型为 COL = 『constant』 的简单相等谓词来解释相关。例如，考虑不同轿车类型的表，每一种轿车都有 MAKE（即制造商）、MODEL、STYLE（即，私家车、旅行车、赛车）、YEAR 和 COLOR。因为几乎每个制造商每年生产的各种型号和式样的轿车都具有相同的标准颜色，所以 COLOR 上的谓词很可能独立于 MAKE、MODEL、STYLE 或 YEAR 上的谓词。然而，因为只有一个轿车制造商才会生产具有特定名称的型号，所以谓词 MAKE 和 MODEL 当然不是独立的。两个或多个轿车制造商几乎没有可能会使用完全相同的星号名，当然轿车制造商也不想这样做。如果 MAKE 和 MODEL 这两列上存在索引，则优化器将使用该索引中的统计信息来确定组合的相异值数，并调整两两列之间的相关的选择性或基数估计。对于这种并非连接谓词的谓词，优化器并不需要唯一索引即可进行调整。

当编译一条 SQL 语句时，SQL 优化器要评估满足请求的不同方法的执行成本。根据此评估，优化器选择它认为的最优存取方案。*存取方案*指定解析一条 SQL 语句所需的操作次序。当联编一个应用程序时，会创建一个*程序包*。此程序包包含该应用程序中所有静态 SQL 语句的存取方案。动态 SQL 语句的存取方案在执行应用程序时创建。

有两种方法来存取一个表中的数据：直接读取该表（关系扫描），或首先存取该表上的索引（索引扫描）。

*关系扫描*在数据库管理程序顺序存取一个表的每一行时执行。参见『索引扫描概念』，以了解索引扫描的原理，参见第142页的『关系扫描与索引扫描』，以了解在哪些条件下使用哪种类型的扫描。

下列主题描述可在存取方案中用来存取表中数据、并产生查询结果的其他方法：

- 第142页的『谓词术语』
- 第144页的『连接概念』
- 第152页的『分区数据库中的连接策略』
- 第159页的『使用优化器排序的影响』。

其他相关主题：

- 第57页的『调整优化级别』，提供有关控制 SQL 编译程序评估的替代存取方案数目的信息。
- 第177页的『第7章 SQL 解释设施』，提供有关如何获取 SQL 编译程序选择的存取方案信息的资料。

索引扫描概念

在数据库管理程序存取一个索引来执行下列任何一个或全部操作时将执行*索引扫描*：

- 在存取基表前，缩小合格行的集合（通过扫描该索引特定范围内的行）。索引*扫描范围*（扫描的起点和终点）由比较索引列的查询中的值确定。
- 将输出定序。
- 全面检索请求的数据。如果所有请求的数据都可用该索引找到，将不存取基表。这称为*纯索引存取*。

也可能按与定义的方向相反的方向来扫描索引。有关详情，参考 *SQL Reference* 中 CREATE INDEX 语句的 ALLOW REVERSE SCANS 选项。

提供下列附加主题:

- 索引结构
- 定界范围的索引扫描
- 对数据定序的索引扫描
- 纯索引存取
- 多索引存取
- 群集索引
- 索引页预读取。

索引结构

数据库管理程序使用 B+ 树结构来存储其索引。一个 B+ 树有一层或多层，如下图中所示（其中，RID 表示行 ID）：

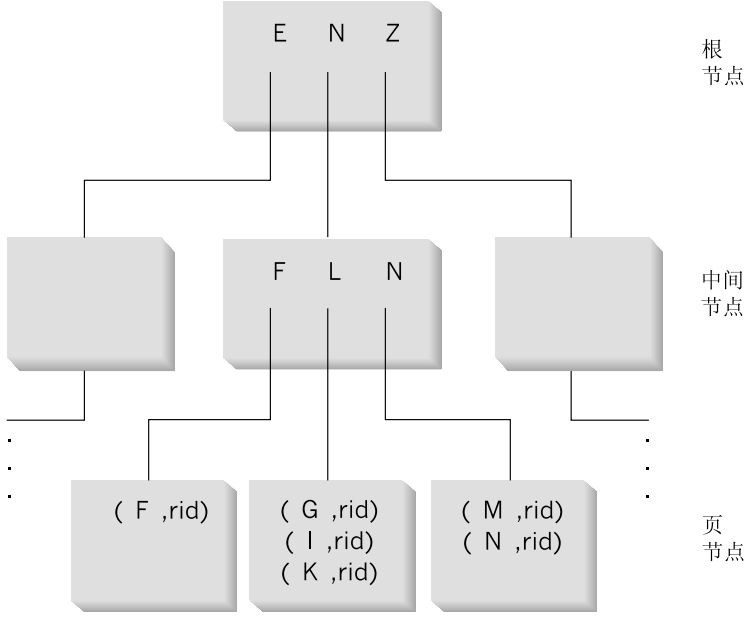


图 13. B+ 树结构

顶层称为*根节点*。底层由*叶节点*组成，底层存储了实际的索引关键字值，并有一个指针指向表中的实际行。根节点层和叶节点层之间的那些层称为*中间节点*。

当查找特定的索引关键字值时，“索引管理器”会从根节点开始搜索该索引树。对于下一层的每个节点根都包含一个关键字。每个关键字的值是下一层中对应节点的最大现存关键字值。例如，如果一个索引有三层，如图13中所示，那么，要查找一个索引关键字值，“索引管理器”将搜索根节点，以查找大于或等于要查

找的关键字的第一个关键字值。此根节点关键字将指向特定的中间节点。对该中间节点执行相同的过程，来确定要转至的叶节点。将在叶节点中找到最后的索引关键字。在第135页的图13中，要查找的关键字是 『I』。在根节点中大于或等于 『I』 的第一个关键字是 『N』。它指向下一层的中间节点。在该中间节点中大于或等于 『I』 的第一个关键字是 『L』。它指向发现有 『I』 的索引关键字及其对应的行 ID （基表中对应行的行 ID）的特定叶节点。

注: 在叶节点层，可能有先前的叶指针。这会有很大的好处，因为在遍历树时一旦发现索引中特定的关键字值，“索引管理器”可按两种方向扫描叶节点来检索值的范围。按任一方向扫描的这种能力仅在索引是使用 ALLOW REVERSE SCANS 参数创建时才可能。

有关详情，参考 *SQL Reference* 中 CREATE INDEX 语句的选项。

定界范围的索引扫描

在确定是否可对特定的查询使用一个索引时，优化器从该索引的第一列开始对索引的每一列求值，以了解它是否可用于满足:

- 该语句的 WHERE 子句中的任何 EQUAL 谓词
- WHERE 子句中任何其他谓词。

*谓词*是 WHERE 子句中搜索条件的元素，它表达或隐含比较运算。可用于定界索引扫描范围的谓词是那些涉及到满足下列其中一个条件的索引列的谓词:

- 正在测试索引列是否等于一个常数、主变量、值为常数的表达式或关键字
- 对索引列进行的测试是 『IS NULL』 或 『IS NOT NULL』
- 测试目的为，检查是否等于基本子查询（即，不包含 ANY、ALL 或 SOME 的子查询），且该子查询没有关联的列引用其直接父查询块（即，此子查询是其子选择的 SELECT）。
- 测试是满足以下描述的条件的不等式谓词。

例如，假定一个索引具有下列定义:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

下列谓词可用于定界索引 IX1 的扫描范围:

```
WHERE  NAME = :hv1
  AND  DEPT = :hv2
```

或

```
WHERE  MGR  = :hv1
  AND  NAME = :hv2
  AND  DEPT = :hv3
```

注意在第二个示例中，WHERE 谓词不必按关键字列在索引中出现的相同次序来指定。而且，尽管示例中使用了主变量，但参数标志符、表达式或常数也会有相同的效果。

在 CREATE INDEX 语句上用 ALLOW REVERSE SCANS 参数创建的单个索引可正向或反向扫描。即这类索引支持以创建索引时定义的方向扫描，也支持以相反的方向扫描。该语句可能类似于如下所示:

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

在这种情况下，根据 cname 中的降序值来构建索引 (iname)。允许反向扫描后，尽管列中的索引被定义为按降序扫描，但也可以按升序进行扫描。当创建和考虑存取方案时，索引的双向实际使用不是由您控制，而是由优化器控制。

在如下的 WHERE 子句中，将只使用 NAME 和 DEPT 的谓词来定界索引扫描的范围，而不使用 SALARY 或 YEARS 的谓词:

```
WHERE  NAME   = :hv1
  AND  DEPT   = :hv2
  AND  SALARY = :hv4
  AND  YEARS  = :hv5
```

这是因为有一个关键字列 (MGR) 将这些列与最前面两个索引关键字列分开，因此定序关闭。但是，一旦该范围由 NAME = :hv1 和 DEPT = :hv2 谓词确定，则可根据其余索引关键字列来对其余谓词求值。

除以上描述的等式谓词外，也可使用特定的不等式谓词来定界索引扫描的范围。下面讨论两类不等式谓词: 严格不等式和相容不等式。

严格不等式谓词: 可用作范围定界谓词的严格不等式运算符是 > 和 <。

要定界索引扫描的范围，将只考虑一个具有严格不等式谓词的列。在以下示例中，NAME 和 DEPT 列上的谓词可用于定界该范围，但不能使用 MGR 列上的谓词。

```
WHERE  NAME   = :hv1
  AND  DEPT   > :hv2
  AND  DEPT   < :hv3
  AND  MGR    < :hv4
```

相容不等式谓词: 以下是可用作范围定界谓词的相容不等式运算符:

- >= 和 <=

- BETWEEN
- LIKE

要定界索引扫描的范围，将考虑多个具有相容不等式谓词的列。在以下示例中，所有谓词都可用于定界索引扫描的范围:

```
WHERE  NAME   = :hv1
  AND  DEPT  >= :hv2
  AND  DEPT  <= :hv3
  AND  MGR   <= :hv4
```

要进一步说明此示例，假定 :hv2 = 404、:hv3 = 406 和 :hv4 = 12345。数据库管理程序将使用该索引扫描部门 404 和 405 的全部，但当它扫描到雇员号（MGR 列）大于 12345 的第一个经理时它将停止扫描部门 406。

有关其他信息，参见第143页的『范围定界谓词和索引 SARGable 谓词』。

对数据定序的索引扫描

如果查询涉及到定序，而且，如果定序列从第一个索引关键字列开始连续出现在该索引中，则可使用索引来对数据定序。（定序或排序可由类似如下的操作产生: ORDER BY、DISTINCT、GROUP BY、『= ANY』 子查询、『> ALL』 子查询、『< ALL』 子查询、INTERSECT 或 EXCEPT、UNION。）一个例外是当比较索引关键字列是否等于“常数值”（即，值为常数的任何表达式）时。在这种情况下，定序列不能是第一个索引关键字列。例如，在如下查询中:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

可使用索引来对行定序，因为 NAME 和 DEPT 将始终是相同的值，因此将被定序。换言之，前面的 WHERE 和 ORDER BY 子句等效于:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY NAME, DEPT, MGR
```

也可使用唯一索引来降低定序要求。例如，给定下列索引定义和 order by 子句:

```
UNIQUE INDEX IX0:  PROJNO  ASC
SELECT PROJNO, PROJNAME, DEPTNO
  FROM PROJECT
ORDER BY PROJNO, PROJNAME
```

不需要根据 PROJNAME 列再进行定序，因为 IX0 索引确保 PROJNO 是唯一的。此唯一性确保对于每个 PROJNO 值只有一个 PROJNAME 值。

纯索引存取

在某些情况下，可从索引检索所有必需的数据，而不用存取表。这称为*纯索引存取*。

要举例说明纯索引存取，考虑下列索引定义:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

仅存取该索引而不用读取基表，就可满足以下查询:

```
SELECT NAME, DEPT, MGR, SALARY
  FROM EMPLOYEE
 WHERE NAME = 'SMITH'
```

在其他情况下，可能有一些列未出现在该索引中。要获取这些列的数据，必须读取基表的行。例如，假设有 IX1 索引，以下查询需要存取该基表来获取 PHONENO 和 HIREDATE 列数据:

```
SELECT NAME, DEPT, MGR, SALARY, PHONENO, HIREDATE
  FROM EMPLOYEE
  WHERE NAME = 'SMITH'
```

通过创建具有包括列的唯一索引，并增加仅通过索引进行的存取尝试次数，可提高数据检索的性能。

要举例说明包括列的使用，考虑以下索引定义:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

它创建了一个唯一索引，以实现 NAME 列的唯一性，同时存储和维护 DEPT、MGR、SALARY 和 YEARS 列的数据。

仅存取该索引而不用读取基表，就可满足以下查询:

```
SELECT NAME, DEPT, MGR, SALARY
   FROM EMPLOYEE
   WHERE NAME='SMITH'
```

多索引存取

在以上所有示例中，仅执行了单索引扫描来产生结果。要满足 WHERE 子句的谓词，优化器可以选择扫描多个索引。例如，假定有下列两个索引定义:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

可使用这两个索引来解析下列谓词:

```
WHERE DEPT = :hv1
  OR (JOB   = :hv2
  AND YEARS >= :hv3)
```

在此示例中，扫描索引 IX2 将产生满足 DEPT = :hv1 谓词的行 ID (RID) 的列表。扫描索引 IX3 将产生满足 JOB = :hv2 AND YEARS >= :hv3 谓词的 RID 的列表。在存取该表之前，可组合 RID 的这两个列表，并除去重复值。这称为*索引 OR 运算*。

索引 OR 运算也可用于使用 IN 表达式的谓词，如下所示:

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

索引 OR 运算的目标是消去重复的 RID；然而，*索引 AND 运算*的目标是查找公共 RID。可在符合下列条件的应用程序内执行索引 AND 运算：在同一个表内的对应列上有多个索引，且对该表运行使用多个 『and』 谓词的一个查询。在这类查询中对具有索引的每一列执行多索引扫描，将产生被散列以创建位图的值。第二个位图用于探测第一个位图，以生成合格行，可读取这些行以创建最终返回的数据集。

例如，假定有下列两个索引定义:

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

可使用这两个索引来解析下列谓词:

```
WHERE SALARY BETWEEN 20000 AND 30000
   AND COMM BETWEEN 1000 AND 3000
```

在此示例中，扫描索引 IX4 产生一个满足 SALARY BETWEEN 20000 AND 30000 谓词的位图。扫描 IX5 和探测 IX4 的位图产生满足这两个谓词的合格 RID 的列表。这称为 “动态位图 AND”。此操作仅在满足下列条件时才执行：该表有足够大的基数，且列在合格范围内有足够多的值，或者如果使用等式谓词，则列具有足够多的重复值。

注: 在存取任何单个表时，DB2 不将索引 AND 运算和索引 OR 运算组合在一起。

群集索引

当选择存取方案时，优化器要考虑从磁盘将页读取至缓冲池的 I/O 成本。在优化器的计算中，优化器将估计满足一个查询所必需的 I/O 数。此估计包括预测缓冲池的使用，因为不需要附加 I/O来读取已在缓冲池内一页中的行。

对于索引扫描，优化器使用系统目录表 (SYSCAT.INDEXES) 中的信息来帮助估计将数据页读入缓冲池的 I/O 成本。使用 SYSCAT.INDEXES 表中的以下列:

- CLUSTERRATIO，指示与此索引相关的表数据的群集程度。高数值表示在数据页上的行已按索引关键字顺序排序。因此，当一个数据页在缓冲区中时，可以读取该页上的所有行。如果此列的值为 -1，则优化器将试图使用 PAGE_FETCH_PAIRS 和 CLUSTERFACTOR。

  或

- PAGE_FETCH_PAIRS，包含几对数字，它们与 CLUSTERFACTOR 一起模拟将数据页读入各种大小的缓冲池时所需的 I/O 数。当收集一个索引的统计信息时，此信息被视为详细的统计信息。

如果统计信息不可用，则优化器将使用该统计信息的缺省值，在这种情况下假定数据与该索引的群集关系不强。另见第93页的『第5章 系统目录统计信息』和第94页的『使用 RUNSTATS 实用程序收集统计信息』。

您可指定一个群集索引，它将用于在表重组期间群集这些行，并在插入处理期间保留此特性。（有关表重组的信息，参见第223页的『重组目录和用户表』。）后续更新和插入可能降低索引的群集率（这由 RUNSTATS 收集的统计信息测量），因此您可能需要定期重组该表。要降低对易变数据库重组的频率，当改变表时使用 PCTFREE 参数。这将允许把附加的插入与现存的数据群集在一起。

数据相对于索引的群集程度可大大影响性能，您应尝试保持该表上的一个索引的群集率接近 100%。

一般情况下，只能有一个索引可达到百分之百的群集，但下列情况除外：其关键字是群集索引关键字的超集的那些索引；或者，在两个索引的关键字列之间存在实际关系的那些索引。

有关使用群集索引的性能原因详情，参见第86页的『管理索引的性能提示』。有关如何创建群集索引的详情，参考 *SQL Reference* 中的 CREATE INDEX。

使用列表预读取群集页读取：如果优化器使用一个索引来存取行，它可将读取数据页延迟到从该索引获得所有 RID（行标识符）之后进行。例如，假定先前定义的索引 IX1:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

和下列搜索标准:

```
WHERE NAME BETWEEN 'A' and 'I'
```

优化器可使用 IX1 执行索引扫描，以确定要检索的行（和数据页）。如果数据未根据此索引群集，则列表预读取将包括一个步骤来把从索引扫描获得的 RID 的列表排序。有关详情，参见第216页的『了解列表预读取』。

索引页预读取

在适当的时候，数据库管理程序要检测对索引页的顺序存取，并生成预读取请求。这将显著缩短不可选择的索引扫描、以及存取该索引大部分内容的可选择索引扫描的经过时间。

优化器使用象 DENSITY 和 SEQUENTIAL_PAGES 这样的索引统计信息、索引驻留的表空间的特征、以及任何范围定界谓词的影响，来估计将发生的索引页预读取的量。这些估计是确定使用特定索引产生的总成本估计的一个因素。

有关详情，参见第214页的『了解顺序预读取』。

关系扫描与索引扫描

当不能使用索引来进行查询时，或者，如果优化器确定索引扫描的成本太高，优化器将选择关系扫描。索引扫描在下列情况下可能需要更高的成本:

- 表很小
- 索引群集率很低
- 表的大部分已存取过。

您可使用“SQL 解释”设施来确定您的存取方案是使用关系扫描还是索引扫描。参见第177页的『第7章 SQL 解释设施』。

谓词术语

用户应用程序使用 SQL 语句从数据库请求一组行，并通过使用谓词来限定期望的特定行。当优化器决定如何对一条 SQL 语句求值时，谓词分成四个类别。该类别由求值过程中使用该谓词的方式和时间来确定。这些类别如下所示，按最佳性能到最差性能的次序排列:

1. 范围定界谓词

2. 索引 SARGable 谓词
3. 数据 SARGable 谓词
4. 残留谓词。

SARGable 是指可用作*查找变元*的谓词。

第144页的『谓词用法摘要』对影响各种谓词类别性能的特征进行了比较。

范围定界谓词和索引 SARGable 谓词

范围定界谓词是用于限定索引扫描范围的那些谓词。它们提供索引搜索的开始和/或停止关键字值。索引 SARGable 谓词不用于限定搜索范围，但可根据该索引来求值，因为在该谓词中涉及到的列是索引关键字的一部分。例如，假定先前定义的索引 IX1（在第134页的『索引扫描概念』一节中）和如下的 WHERE 子句:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

前两个谓词 (NAME = :hv1, DEPT = :hv2) 是范围定界谓词，而 YEARS > :hv5 是索引 SARGable谓词。

数据库管理程序将使用索引数据来对这些谓词求值，而不必读取基表。这些*索引 SARGable* 谓词通过减小需要从表中读取的行集，来减少存取的数据页数。这些类型的谓词不影响存取的索引页数。

数据 SARGable 谓词

“索引管理器”不能求值但数据管理服务可以求值的谓词称为*数据 SARGable* 谓词。通常，这些谓词需要从基表存取个别行。如果需要，“数据管理服务”将检索对该谓词求值需要的列，以及任何其他列以满足 SELECT 列表中不能从该索引获取的列。

例如，假定在 PROJECT 表上定义了单个索引:

```
INDEX IX0:  PROJNO ASC
```

并假定下列查询，并将 DEPTNO = 'D11' 谓词视为数据 SARGable。

```
SELECT PROJNO, PROJNAME, RESPEMP
  FROM PROJECT
 WHERE DEPTNO = 'D11'
 ORDER BY PROJNO
```

残留谓词

残留谓词是超出简单存取基表范围的、需要 I/O 的那些谓词。残留谓词的示例包括使用关联子查询、使用量化子查询（带 ANY、ALL、SOME 或 IN 的子查询）或

读取 LONG VARCHAR 或 LOB 数据（存储在不含该表的一个文件中）的那些谓词。这些谓词由“关系数据服务”来求值。

有时，在存取数据页时，必须重新应用仅适用于索引的谓词。例如，当存取数据页时，使用索引 OR 运算或索引 AND 运算（参见第139页的『多索引存取』）的存取方案，始终要将这些谓词作为残留谓词重新应用。

谓词用法摘要

在一个查询中使用谓词可有助于减少满足该查询要读取的数据量。不同类别的谓词对一个查询的性能有不同的影响，优化器会考虑到这些影响。下表显示不同类型的谓词的等级，以及每一类谓词可以影响性能的程度。

表 14. 谓词类型特征的摘要

特征	谓词类型			
	范围定界	索引 SARGable	数据 SARGable	残留
减少索引 I/O	是	否	否	否
减少数据页 I/O	是	是	否	否
减少内部传送的行数	是	是	是	否
减少合格行的数目	是	是	是	是

连接概念

*连接*是将一个表中的行与另外一个或多个表中的行并置。例如，假定下列两个表:

```
      TABLE1                    TABLE2
-----------------         -----------------
 PROJ     PROJ_ID         PROJ_ID     NAME
------    -------         -------    ------
  A          1              1        Sam
  B          2              3        Joe
  C          3              4        Mary
  D          4              1        Sue
                            2        Mike
```

将 ID 列相同的 Table1 和 Table2 连接将由下列 SQL 语句表示:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

并将得到下列一组结果行:

```
 PROJ     PROJ_ID     NAME
------    -------    ------
  A          1        Sam
  A          1        Sue
  B          2        Mike
  C          3        Joe
  D          4        Mary
```

当连接两个表时，将一个表选为外部表，而将另一个表选为内部表。首先存取外部表，并只存取它一次。是否扫描内部表多次取决于连接的类型和存在哪些索引。无论您的查询是否连接两个表或不只两个表，优化器一次将只连接两个表。如果需要，将创建临时的、中间结果表。

优化器将根据连接谓词（在第146页的『合并连接』中定义的）是否存在以及所涉及到的、由表和索引统计信息来确定的各种成本，选择两种连接方法（嵌套循环连接或合并连接）中的一种。

嵌套循环连接

嵌套循环连接是用以下两种方式之一执行的:

1. 对于外部表中每个被存取的行，扫描整个内部表

   例如，如果表 T1 和 T2 中的列 A 有下列值:

```
   外部表 T1: 列 A                内部表 T2: 列 A
------------------------      ------------------------
          2                             3
          3                             2
          3                             2
                                        3
                                        1
```

   执行嵌套循环的步骤包括:

   - 从 T1 中读取第一行。A 的值是 『2』
   - 扫描 T2，直到发现一个匹配项 (『2』)，然后连接这两行
   - 扫描 T2，直到发现下一个匹配项 (『2』)，然后连接这两行
   - 扫描 T2，直至该表结尾
   - 返回至 T1，并读取下一行 (『3』)
   - 从第一行开始，扫描 T2，直到发现匹配项 (『3』)，然后连接这两行
   - 扫描 T2，直到发现下一个匹配项 (『3』)，然后连接这两行
   - 扫描 T2，直至该表结尾
   - 返回至 T1，并读取下一行 (『3』)
   - 象以前一样扫描 T2，连接匹配 (『3』) 的所有行。

2. 对于外部表中每个被存取的行，在内部表上执行索引查找。

如果存在下列形式的谓词，则此方法可用于指定的谓词:

```
expr(outer_table.column)  relop  inner_table.column
```

其中，relop 是比较运算符（例如，=、>、>=、< 或 <=），而 expr 是基于外部表的有效表达式。以下是示例:

```
OUTER.C1 + OUTER.C2 <= INNER.C1
```

和

```
OUTER.C4 < INNER.C3
```

此方法可以显著减少在每次存取外部表时要在内部表中存取的行数（虽然这取决于许多因素，包括连接谓词的选择性）。

当对嵌套循环连接求值时，优化器在执行连接前也要确定是否对外部表排序。根据连接列对外部表定序，可减少从磁盘存取内部表的页的读取操作次数，因为很可能这些页已在缓冲池中。如果该连接使用高度群集的索引来存取内部表，且如果外部表已排序，则可将存取的索引页的数目减至最小。

另外，如果优化器期望该连接执行一个成本更高的、更新的排序，则优化器也可选择在连接前执行排序。要支持 GROUP BY、 DISTINCT、ORDER BY 或合并连接，可能必需更新的排序。

合并连接

合并连接（有时称为合并扫描连接或排序合并连接）需要如下形式的一个谓词 table1.column = table2.column。这称为*等式连接谓词*。合并连接需要连接列的定序输入，这个输入可通过索引存取或通过排序来获得。要使用合并连接，该连接列不能是 LONG 字段列或大对象 (LOB) 列。

同时扫描连接的表。合并连接的外部表只扫描一次。除非外部表中有重复的值，否则，内部表也只扫描一次。如果外部表中有重复的值，则可再次扫描内部表中的一组行。例如，如果表 T1 和 T2 中的列 A 有下列值:

```
    外部表 T1: 列 A              内部表 T2: 列 A
------------------------     ------------------------
          2                            1
          3                            2
          3                            2
                                       3
                                       3
```

执行合并连接的步骤包括:

- 从 T1 中读取第一行。A 的值是 『2』
- 扫描 T2，直到发现匹配项，然后连接这两行

- 连续扫描 T2，当列匹配时，连接那些行。
- 当读取 T2 中的 『3』 时，返回至 T1 并读取下一行
- T1 中的下一个值是 『3』，它与 T2 匹配，因此连接那些行
- 连续扫描 T2，当列匹配时，连接那些行
- 到达 T2 结尾
- 返回至 T1 以读取下一行 — 注意 T1 中的下一个值与 T1 中的上一个值相同，因此从 T2 中的第一个 『3』 开始再次扫描 T2（数据库管理程序会记住此位置）。

散列连接

散列连接需要如下形式的一个或多个谓词: table1.columnX = table2.columnY，在该形式中，列类型是**相同的**。对于类型为 CHAR 的列，长度必须相同。对于类型为 DECIMAL 的列，精度和标准尺必须相同。列类型不能是 LONG 字段列或大对象 (LOB) 列。

首先，扫描一个表（称为 INNER 表），然后将行复制到从排序堆分配的内存缓冲区中（参见第303页的『排序堆大小 (sortheap)』数据库配置参数）。根据由连接谓词的列计算而得的“散列码”，将内存缓冲区分为若干分区。如果第一个表的大小超过可用的排序堆空间，则将所选分区中的缓冲区写入临时表。当处理完 INNER 表后，扫描第二个表（称为 OUTER 表）。首先比较由连接谓词的列生成的“散列码”，将 OUTER 表的行与 INNER 表的行匹配。然后，如果 OUTER 行的“散列码”与 INNER 行的“散列码”匹配，则比较实际连接谓词列。

紧接着将与未写入临时表的分区对应的 OUTER 表行与内存中的 INNER 表行匹配。否则，如果对应的 INNER 表分区已写入临时表，要将 OUTER 行写入临时表。最后，从临时表中读取匹配的分区对，并比较它们的行的“散列码”，检查连接谓词。

要实现散列连接的性能效益，可能需要更改 *sortheap* 数据库配置参数和 *sheapthres* 数据库管理程序配置参数的值。

对于决策支持查询，散列连接存取方案使用比非散列连接方案更多的排序堆空间。当将 *sheapthres* 设置为相对接近 *sortheap* 时（也就是，每个并行查询占用的空间减少两三倍），则散列连接将以比优化器所预料的少得多的内存来运行。在内存不足时执行，散列连接可能会非常慢。具有多个排序和散列连接的查询中会发生此问题，在这种情况下，排序或散列连接会获取可用内存中的绝大部分。

解决方案是将 *sheapthres* 配置为足够大（相对于 *sortheap* 而言）。

外部与内部的确定

当连接时，如何确定内部表和外部表？下面给出了优化器如何决定哪个表是内部表、哪个表是外部表的一般准则。

对于**散列连接**，内部表保存在内存缓冲区中。如果内存缓冲区太少，则散列连接不得不溢出。优化器将试图避免这种情况，因此将选取两个表中较小的一个作为内部表，较大的一个作为外部表。

存取表所用的次序对于**嵌套循环连接**特别重要，因为外部表只被存取一次，而对于外部表的每一行内部表都要被存取一次。优化器根据成本估计来选择外部表和内部表。这些成本估计受下列因素影响:

- 大小

  常常选择较小的表作为外部表，以减少必须重新存取内部表的次数。而预读取刚好带来相反的结果。预读取可显著减少存取大表的成本。但是，通常预读取只对连接的外部表有效。因此，较大的表可能被首先存取。有关详情，参见第214页的『将数据预读取至缓冲池』。

- 谓词

  如果可对一个表应用选择性谓词，则很可能将该表选择作为外部表，因为存取内部表只是为了读取满足应用于外部表的谓词的那些行。

- 缓冲

  如果必须对外部表的每一行扫描整个内部表（即，不能对内部表执行索引查找），可以选择两个表中的较小者作为内部表来利用缓冲。这将受表大小和缓冲池大小的影响。注意因为连接判断受缓冲池大小影响，因此如果您在更改缓冲池大小后将您的应用程序与数据库重新联编，则您的应用程序的存取方案可能改变。

  您在下列方面的能力: 创建多个缓冲池、更改该缓冲池的大小和控制使用该缓冲池的表空间，可以影响在内部表和外部表中使用缓冲的时间。

- 索引

  如果在其中一个表上执行索引查找可能的话，则该表就是内部表的一个良好候选者。那么，就可以将外部表的连接关键字谓词用作其中一个关键字值，使用索引关键字查找来存取它。如果一个表没有索引，它不会是内部表的良好候选者，因为在那种情况下，将不得不对外部表的每一行扫描整个内部表。

- 次序要求

  可能首先存取与必需的次序相关的表。例如，如果要根据 t1.c 将 t1 和 t2连接的输出定序，那么较好的选择是使用 t1.c 上的索引将 t1 作为外部表来存取。将对连接的输出定序，而不需要排序。

```
SELECT * FROM t1, t2
  WHERE t1.a = t2.b
  ORDER BY t1.c
```

存取表所用次序的重要性略低于**合并连接**的重要性，因为内部表和外部表只被读取一次。但是，与外部表中重复的连接值对应的内部表部分保留在内存中的缓冲区中。如果外部表中的下一行与上一行相同，则重新读取该缓冲区，否则将该缓冲区复位。如果重复连接值的数目超过内存中缓冲区的容量，则不是全部重复值都保存。仅当任何一个值的重复次数很多，而该值在外部表中有匹配的值时，才会发生这种情况。

将重复值这些因素全部考虑在内，在大多数情况下，将选择具有更少重复值的表作为一个连接中的外部表。但是，最终，优化器会根据详细的成本估计来选择外部表和内部表。

选择最优连接的搜索策略

优化器可以使用不同的搜索策略来确定最优连接方法。将使用的搜索策略由所用的优化级别（参见第57页的『调整优化级别』）确定。搜索策略及其特性是:

- 贪婪连接枚举
  - 可节省空间和时间
  - 单向枚举；即，一旦为两个表选择了一个连接方法，在进一步优化期间将不更改它
  - 当连接多个表时，可能丢失最佳存取方案。如果您的查询只连接两个或三个表，则贪婪连接枚举选择的存取方案将与动态规划连接枚举选择的存取方案相同。如果该查询在相同的列上有多个连接谓词（显式指定的，或通过谓词传递闭包隐式生成的），则更是如此。
- 动态规划连接枚举
  - 空间和时间需求随着连接的表的数目增多呈指数增长
  - 可有效地、全面地搜索以获得最佳存取方案
  - 类似于 DB2 OS/390 版使用的策略。

连接枚举算法是决定优化器使用多少种方案组合的一个重要因子。

星形连接的搜索策略

通常，在一个查询中引用的表应由连接谓词连接起来。如果两个表已连接但不存在连接谓词，则形成了两个表的笛卡儿乘积。即，第一个表的每个合格行与第二个表的每个合格行连接，生成一个结果表，它由两个表大小的交叉乘积组成，该值一般很大。因为这类方案不可能执行得很好，因此优化器甚至会避免确定这类

存取方案的成本。当将优化级别设置为 9 时，或对于“星形模式”的下列特殊情况，会出现唯一的例外。有关详情，参见第57页的『调整优化级别』。

涉及笛卡儿乘积的存取方案执行良好的情况通常是使用“星形模式”技术设计的大型决策支持数据库。星形模式是一种数据库设计，在该设计中，大量未处理的数据保留在含很多列的单个大表中，该表通常称为“事实”表。许多列包含已编码的值，这些值描述存储在事实表中的特定数据的维。为了能够方便地分析这些事实的某个子集，使用维表将已编码的值解码。典型的查询由多个本地谓词组成（这些谓词引用维表中已解码的值），并包含将维表与事实表连接的连接谓词。对于这些种类的查询，在存取大的事实表之前，对多个小维表执行笛卡儿乘积可能有好处。此技术在多个连接谓词与一个多列索引匹配时有用。

DB2 能够认可对使用星形模式设计的、含至少两个维表的数据库的查询，并可增加搜索空间来包括涉及到形成维表的笛卡儿乘积的潜在方案。如果涉及笛卡儿乘积的方案具有最低的估计成本，优化器将选择它。

以上讨论的“星形模式”技术假设连接中使用了主关键字索引。另一个方案可能涉及到外部关键字索引。假定事实表中的外部关键字列是单列索引，且在所有维表中有相当高的选择性，可使用以下“星形连接”技术:

1. 每个维表由下列操作处理:
   - 在维表和事实表上的外部关键字索引之间执行半连接
   - 散列行 ID (RID) 值，以动态创建一个位图。
2. 使用 『and』 谓词将每个位图与上一个位图连接（参见第139页的『多索引存取』）。
3. 在处理最后一个位图后，确定幸存的 RID。
4. 选择是否对这些 RID 排序。
5. 读取一个基表行。
6. 将事实表与其每个维表重新连接，存取 SELECT 子句所需要的维表的列
7. 重新应用谓词（残留谓词）

使用此技术，不需要有多列索引。选择此技术并不要求事实表与维表之间存在显式参考完整性约束，尽管事实表与维表之间的关系应具有此特性。

组合表

另一个重要参数确定一个查询中连接序列的形状。将一对表连接的结果是一个新表，称为组合表。通常，生成的这个组合表成为与另一个内部表相连的连接中的外部表。这称为“组合外部”。在某些情况下，特别是在使用贪婪连接枚举技术

时，提取连接两个表的结果并使该结果成为后续连接的内部表将很有用。当一个连接本身的内部表由两个或更多个表连接的结果组成时，可以说该方案包含“组合内部”。例如，在如下查询中:

```
SELECT COUNT(*)
FROM T1, T2, T3, T4
WHERE T1.A = T2.A AND
      T3.A = T4.A AND
      T2.Z = T3.Z
```

以下做法可能有用: 将表 T1 和 T2 连接 (T1xT2)，然后将 T3 与 T4 连接 (T3xT4)，最后选择第一个连接结果作为外部，选择第二个连接结果作为内部。在最后的方案 ((T1xT2) x (T3xT4)) 中，连接结果 (T3xT4) 称为组合内部。根据查询优化级别，优化器对可以作为一个连接内部表的若干个表的最大数目设置不同的约束。优化级别 5、7 和 9 允许组合内部。

复制的摘要表

可通过在一个分区数据库环境中使用复制的摘要表，并让数据库管理预先计算的基表数据的值，来提高性能。例如，下面的查询将从创建下面的复制摘要表受益。作下列假定:

- SALES 表在多分区表空间 REGIONTABLESPACE 中，且根据 REGION 列分区。
- EMPLOYEE 和 DEPARTMENT 表在单分区节点组中。

然后，根据 EMPLOYEE 表中的信息创建复制的摘要表。

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
      FROM EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

在创建之后，通过运行此语句更新复制的摘要表的内容:

```
REFRESH TABLE R_EMPLOYEE;
```

以下示例计算雇员的销售额、部门总销售额和总计:

```
SELECT d.mgrno, e.empno, SUM(s.sales)
FROM   department AS d, employee AS e, sales AS s
WHERE  s.sales_person = e.lastname
   AND e.workdept = d.deptno
GROUP BY ROLLUP(d.mgrno, e.empno)
ORDER BY d.mgrno, e.empno;
```

如果不使用在唯一一个数据库分区上的 EMMPLOYEE 表，数据库管理程序将使用 R_EMPLOYEE 表，将在 SALES 表所在的每个数据库分区上复制它。这样性能将会增强，因为雇员信息不必通过网络传送到每个数据库分区来计算该连接。

分区数据库中的连接策略

下列几节描述在一个分区数据库环境中可采用的连接策略。DB2 优化器根据每个应用程序的需求自动选择最佳连接策略。下面将提供连接策略来帮助您了解每个策略的内容。“表队列”是在数据库分区之间或单一分区数据库中的各处理器之间传送行的一种机制。

在随后的说明中，*定向*表队列是将它们的行散列至其中一个接收数据库分区的表队列。*广播*表队列是将它们的行发送至所有接收数据库分区（即，不散列它）的一个表队列。在本节的图中，q1、q2 和 q3 是指示例中的表队列。引用的表也分布在两个数据库分区中，以实施这些方案。箭头指示发送表队列的方向。协调程序节点是分区 0。

对于在一个分区数据库中进行的频繁连接所涉及的那些表，要考虑的一点是表并置。表并置提供了一种方法，即在一个分区数据库中，根据相同的分区关键字，可在同一个分区中使用一个表中的数据查找另一个表中的数据。一旦被配置，要连接的数据可参与查询，而不必在执行查询活动时移动至另一个数据库分区。只将连接的回答集移动至协调程序节点。有关此主题的详情，*参考管理指南: 计划*中的“表并置”。

有关连接从属性的信息，参考 *SQL Reference* 手册。

并置连接

要使优化器考虑并置连接，必须并置连接的表，且所有对应的分区关键字对必须参与等值连接谓词。第153页的图14中显示了一个示例。

注: 复制的摘要表提高了并置连接的可能性。有关详情，参见第151页的『复制的摘要表』。

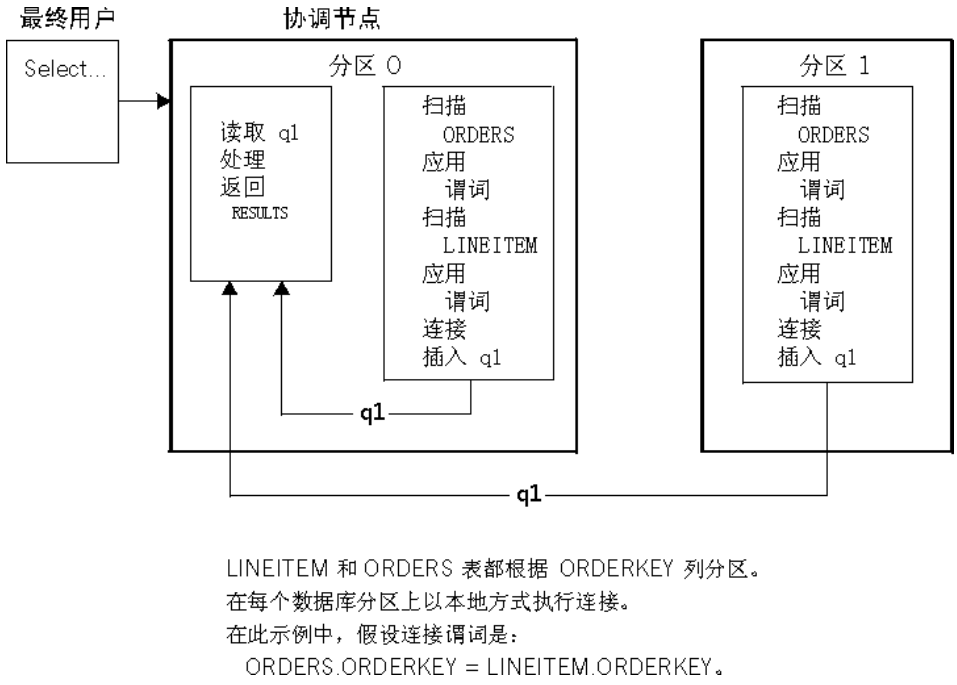
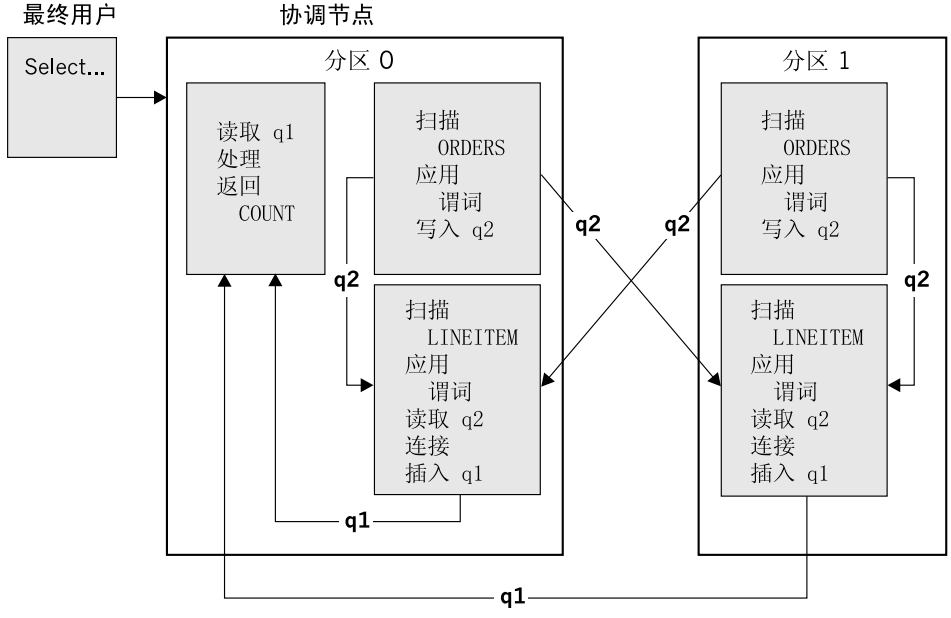


图 14. 并置连接示例

广播外部表连接

如果在连接的表之间没有等值连接谓词，可使用此并行连接策略。在可以将它用作成本最低的连接方法的其他情况中，也可使用它。通常，当有一个很大的表和一个很小的表且任何一个表都没有依据连接谓词列分区时，会发生这种情况。不用将两个表分区，将更小的表广播至更大的表可能“成本更低”。第154页的图15中显示了一个示例。

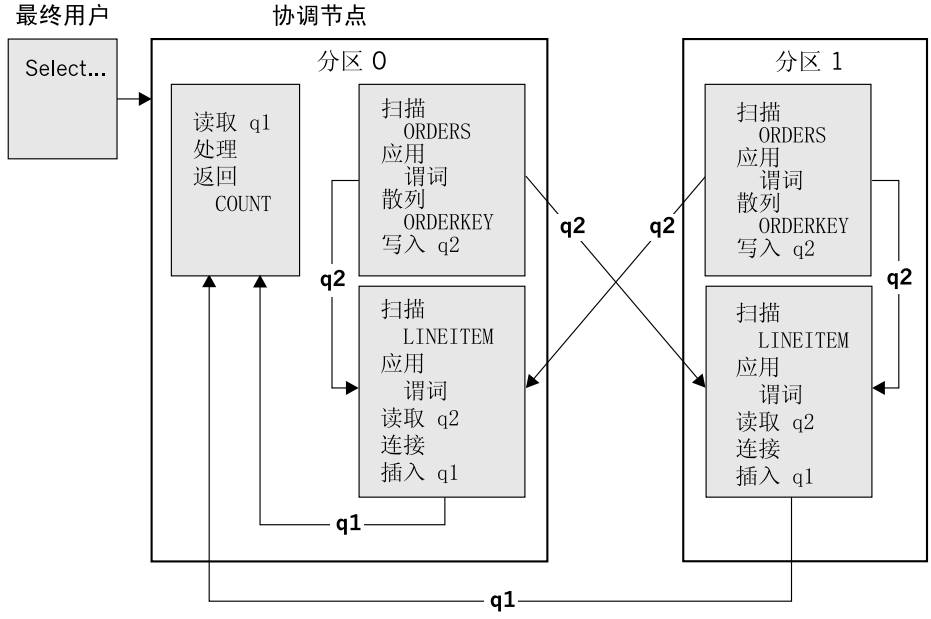


ORDERS 表被发送至所有带有 LINEITEM 表的数据库分区。
表队列 q2 被广播至内部表的所有数据库分区。

图 15. 广播外部表连接示例

定向外部表连接

在这个连接策略中，将外部表的每一行发送至内部表的一个数据库分区（根据内部表的分区属性）。该连接在此数据库分区上进行。第155页的图16中显示了一个示例。

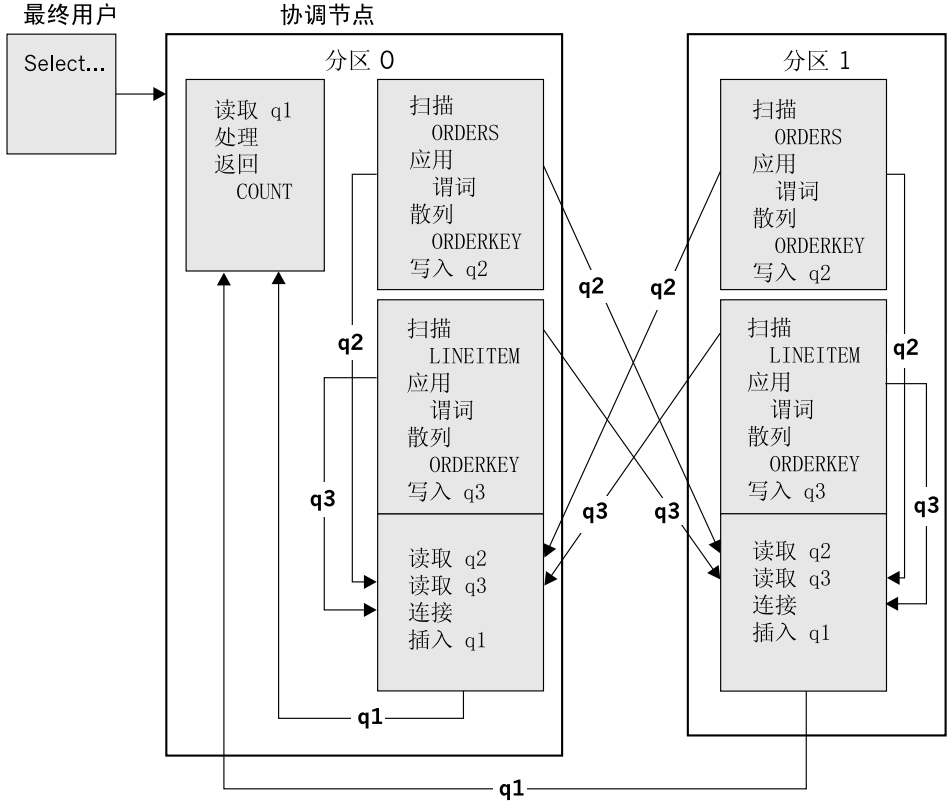


LINEITEM 表根据 ORDERKEY 列进行分区。
ORDERS 表根据另一个列进行分区。
ORDERS 表被散列并发送至正确的 LINEITEM 表数据库分区。
在此示例中，假设连接谓词是：
ORDERS.ORDERKEY = LINEITEM.ORDERKEY。

图 16. 定向外部表连接示例

定向内部表和外部表连接

在此策略中，根据连接列的值，将外部表和内部表的行定向到一组数据库分区。该连接在这些数据库分区上进行。第156页的图17中显示了一个示例。

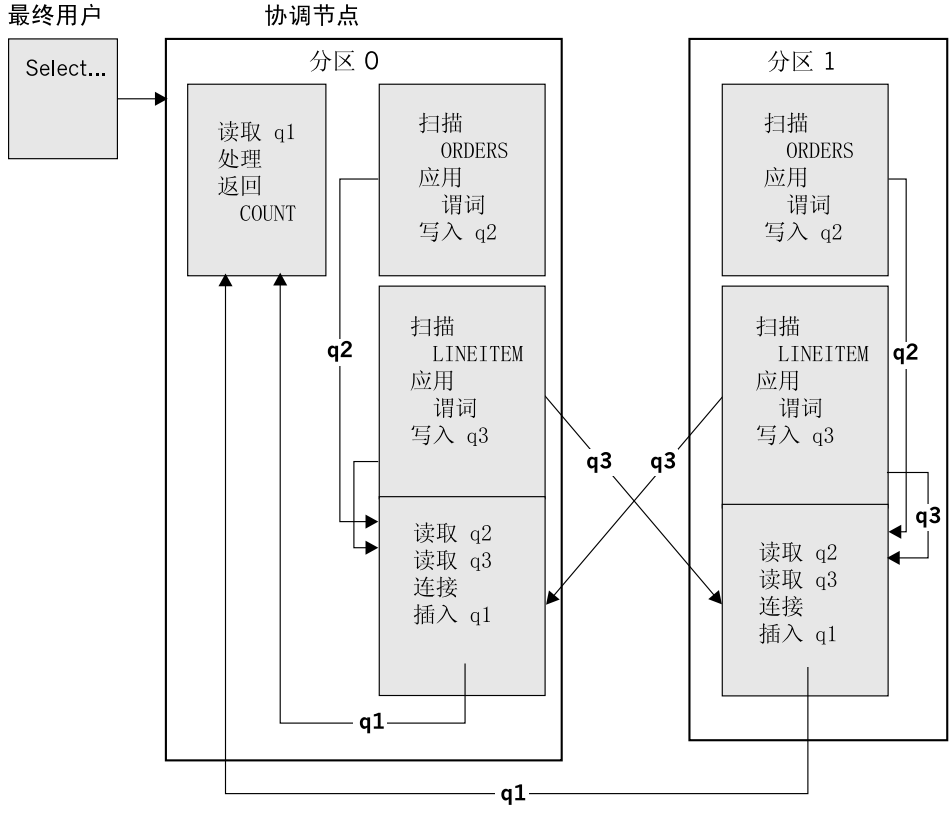


两个表都不根据 ORDERKEY 列分区。
两个表都被散列并发送至新数据库分区，它们在那里连接。
表队列 q2 和 q3 都被定向。
在此示例中，假设连接谓词是：
ORDERS.ORDERKEY = LINEITEM.ORDERKEY

图 17. 定向内部表和外部表连接示例

广播内部表连接

在此策略中，将内部表广播至外部连接表的所有数据库分区。第157页的图18中显示了一个示例。

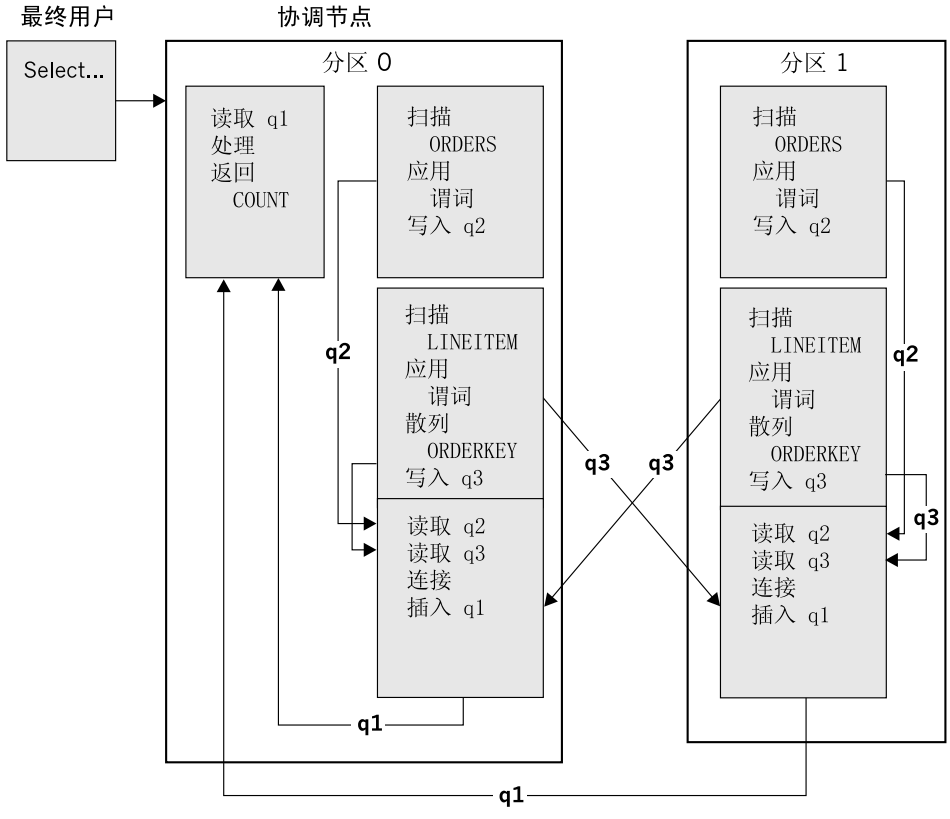


LINEITEM 表被发送至所有带有 ORDERS 表的数据库分区。
表队列 q3 被广播至外部表的所有数据库分区。

图 18. 广播内部表连接示例

定向内部表连接

在此策略中，将内部表的每一行发送至外部连接表的一个数据库分区（根据外部表的分区属性）。该连接在此数据库分区上进行。第158页的图19中显示了一个示例。



ORDERS 表根据 ORDERKEY 列分区。
LINEITEM 表根据另一个列分区。
LINEITEM 表被散列并发送至正确的 ORDERS 表数据库分区。
在此示例中，假设连接谓词是：
ORDERS.ORDERKEY = LINEITEM.ORDERKEY。

图 19. 定向内部表连接示例

表队列

表队列用于:

- 在使用分区内并行性时，将表数据从一个数据库分区传送至另一个
- 在使用分区内并行性时，传送一个数据库分区中的表数据
- 在使用单个分区数据库时，传送一个数据库分区中的表数据。

每个表队列用于按单方向传送数据。

编译程序决定需要哪些表队列，并在方案中包括它们。当执行方案时，数据库分区之间的连接会启动表队列。一旦进程结束，则表队列关闭。

有几种类型的表队列:

- *异步表队列*。这些表队列称为异步，是因为它们在应用程序发出的任何 FETCH 之前读取行。当发出 FETCH 时，从表队列检索该行。

  当您在 SELECT 语句上指定 FOR FETCH ONLY 子句时，使用异步表队列。如果您只读取行，则异步表队列会更快。
- *同步表队列*。这些表队列称为同步，是因为它们对应用程序发出的每个 FETCH 读取一行。在每个数据库分区中，将游标定位在要从该数据库分区读取的下一行上。

  当您未在 SELECT 语句上指定 FOR FETCH ONLY 子句时，使用同步表队列。在一个分区数据库环境中，如果您要更新行，则数据库管理程序将使用同步表队列。
- *合并表队列*。这些表队列维持次序。
- *非合并表队列*。这些表队列也称为“正规”表队列。它们不维持次序。
- *监听程序表队列*。这些表队列在关联子查询中使用。使用这种类型的表队列，将关联值向下传送至子查询，然后将结果向上传送回父查询块。

使用优化器排序的影响

当优化器选择一个存取方案时，它会考虑对数据排序给性能带来的影响。当没有索引满足被读取的行请求的定序时，执行排序。当优化器确定排序的成本低于索引扫描的成本时，也可执行排序。当对数据排序时，优化器可执行下列其中一个操作:

- 当执行查询时，通过管道排序的结果。参见『管道排序与非管道排序』和第77页的『影响查询优化的配置参数』。
- 在数据库管理程序内对排序进行内部处理。参见第160页的『聚合和排序下推运算符』。

管道排序与非管道排序

当排序完成时，如果可按单一顺序读取数据已排序的最终列表，则可*管道传送*这些结果。管道式比使用其他方法（非管道式）传送排序结果快。优化器会尽可能地选择用管道传送排序的结果。

与是否用管道传送排序无关，排序时间将取决于多种因素，包括要排序的行数、关键字大小和行宽。如果要排序的行占用的空间超过排序堆中可用的空间，则执行几遍排序，每一遍都要对整个行集的某个子集排序。每遍排序的结果存储在缓冲池中的一个临时表内。(作为缓冲池管理的一部分，此临时表中的页可能可以写

入磁盘。）一旦所有排序都完成，必须将这些已排序的子集合并为单个已排序的行集。如果用管道传送排序，当合并那些行时，直接将它们递交给“关系数据服务”。

有关详情，参见第221页的『查找排序性能问题的指示符』，或第77页的『影响查询优化的配置参数』中对 *sortheap* 配置参数的讨论。

聚合和排序下推运算符

在有些情况下，优化器可以选择将排序或聚合操作从“关系数据服务”部件下推到数据管理服务部件。下推这些操作将改善性能，因为这样“数据管理服务”部件可直接将数据传送至一个排序或聚合例程。如果不使用此下推，“数据管理服务”将首先把此数据传送至“关系数据服务”，然后“关系数据服务”再与排序或聚合例程连接。例如，以下查询可从此优化受益:

```
SELECT WORKDEPT, AVG(SALARY) AS AVG_DEPT_SALARY
  FROM EMPLOYEE
  GROUP BY WORKDEPT
```

排序中的聚合

当使用排序来产生 GROUP BY 操作所需的次序时，优化器在排序时可选择执行 GROUP BY 的部分或全部聚合。如果每组中的行数较大，这种做法就有利。如果在排序期间执行一些分组以减少或消去排序溢出到磁盘的情况，就会更加有利。

当使用排序中的聚合时，有最多三个必需的聚合阶段来确保计算正确的结果。聚合的第一个阶段“部分聚合”计算在排序堆填满之前的聚合值。部分聚合是接受未聚合数据并产生部分聚合的过程。如果排序堆已填满，其余数据会溢出到磁盘，包括在排序堆的当前填充中已计算的所有部分聚合。在排序堆复位之后，开始新的聚合。

聚合的第二阶段“中间聚合”提取所有溢出的排序运行结果，并根据分组关键字进一步聚合。由于分组关键字列是分区关键字列的一个子集，故不能完成聚合。中间聚合接收现存的部分聚合，然后产生新的部分聚合。此阶段是可选的，且用于分区内并行性和分区间并行性。在最后一种情况下，当全局分组关键字可用时将完成分组。在分区间并行性的情况下，如果分组关键字是用来在分区间分组的分区关键字的子集，从而要求重新分区来完成聚合时，将会发生这种情况。当所有代理程序合并为单个代理程序以完成聚合之前每个代理程序合并完其溢出的排序运行结果时，在分区内并行性中将存在相似的情况。

聚合的最后阶段“最终聚合”提取所有部分聚合，然后完成该聚合。最终聚合接收部分聚合，并产生最终的聚合。此步骤总是通过 GROUP BY 运算符执行。排序不能执行完整的聚合，因为无法保证排序不分裂。完整的聚合接收未聚合的数据，然后产生最终聚合。这种聚合方法通常在对已按正确次序排列的数据进行分区且分区不禁止使用聚合时使用。

分区内并行性的优化策略

优化器可选择一个存取方案，这样，如果在编译 SQL 语句时指定了并行度，就可在一个数据库分区内并行执行查询。

在执行时，创建称为“子代理程序”的多个数据库代理程序来执行该查询。子代理程序的数目小于或等于编译该 SQL 语句时确定的并行度。有关设置 SQL 语句并行度的详情，参考第74页的『应用程序的并行处理』。有关代理程序和子代理程序的详情，参考第227页的『数据库代理程序』。

在一个分区数据库中，并行度应用于每个分区。例如，根据在给定数据库分区中为该 SQL 语句确定的并行度，对在该数据库分区中执行的查询的某部分进一步并行化。

存取方案的并行化是这样进行的：将存取方案划分为每个子代理程序运行的一部分，和协调代理程序运行的一部分。子代理程序通过表队列将数据传送至协调代理程序或其他子代理程序。在一个分区数据库中，子代理程序可以通过表队列从其他数据库分区中的子代理程序发送或接收数据。

本节描述单个数据库分区中的并行化策略。

并行扫描策略

关系扫描和索引扫描可在同一个表或索引上并行执行。要进行并行关系扫描，须将表划分为由页或行组成的范围。将一个范围内的页或行分配给一个子代理程序。一个子代理程序扫描分配给它的范围，当它处理完当前范围时，会给它分配另一个范围。

要进行并行索引扫描，须根据索引关键字值和一个关键字值的索引项数，将索引划分为由记录组成的范围。并行索引扫描的执行方式类似于并行表扫描，将给子代理程序分配一个范围内的记录。当子代理程序处理完当前范围时，会给它分配一个新的范围。

扫描单位（页或行）和扫描粒度由优化器确定。

并行扫描可将工作均匀地分布在各子代理程序中。并行扫描的目标是平衡所有子代理程序的负荷，并使它们保持相同的繁忙程度。如果繁忙子代理程序数等于可用处理器数，且磁盘没有过度处理 I/O请求，则表明机器资源正在得到有效的利用。

当执行查询时，其他存取方案操作可能导致数据分布不均匀。优化器选择并行策略，以便维护数据平衡。

并行排序策略

优化器可选择下列其中一个并行排序策略:

循环排序

它也称为“重新分布排序”。它是一种有效的共享内存排序，尝试将数据尽可能均匀地重新分布到所有子代理程序中。它使用循环时钟类型算法来提供均匀分布。它首先为每个子代理程序创建个别排序。在插入阶段期间，子代理程序以循环方式插入每个个别的排序中。这样就能使数据更均匀地分布。

分区排序

这类似于循环排序，在循环排序中，要为每个子代理程序创建一个排序。子代理程序将一个散列函数应用于排序列，以确定应将行插入哪个排序中。例如，如果一个合并连接的内部和外部是一个分区排序，则子代理程序可使用合并连接来连接对应的分区。这允许合并连接并行执行。

复制型排序

在所有子代理程序都需要所有排序输出的情况下，使用此排序。创建一个排序，并在插入至排序期间使子代理程序同步。当完成排序时，每个子代理程序都读取整个排序。此排序可用于在行数较小时，重新平衡数据流。

共享排序

此排序与复制型排序相同，不同的是子代理程序要对已排序的结果打开一个并行扫描。它将以类似于循环排序的方式把数据分布到子代理程序中。

并行临时表

子代理程序可以协同工作，将行插入同一个表中，来产生临时表。这称为共享临时表。子代理程序可以根据数据流是要复制还是要分区，在共享临时表上打开专用扫描或并行扫描。

并行聚合策略

子代理程序可以并行执行聚合操作。聚合操作要求根据分组的列将数据排序。如果可以保证一个子代理程序接收一组分组列值的所有行，则该程序可以执行完整的聚合。如果由于先前的分区排序，已根据分组列将数据流分区，则这种情况可能会发生。

否则，子代理程序可以执行部分聚合，并使用另一种策略来完成该聚合。其中部分策略如下:

- 通过一个合并表队列将部分聚合的数据发送至协调代理程序。协调程序完成聚合。

- 将部分聚合的数据插入一个分区排序中。根据分组列将该排序分区。这保证一组分组列的所有行都包含在一个排序分区中。
- 如果由于平衡的原因需要复制数据流，可将部分聚合的数据插入一个复制型排序中。每个子代理程序使用复制型排序完成聚合，并接收完全相同的聚合结果副本。

并行连接策略

子代理程序可以并行执行连接操作。并行连接策略由数据流的特征确定。

通过在连接的内部和外部上分区和／或复制数据流，可将连接并行化。例如，如果由于并行扫描已将一个嵌套循环连接的外部流并行化，而且内部流由每个子代理程序单独重新求值，则可将该连接并行化。如果一个合并连接的内部和外部流由于分区排序都已按值分区，则可将该连接并行化。

自动摘要表

摘要表是一种功能强大的改进查询响应时间的方法。在许多可以预测一些基本查询结构的环境中，可使用摘要表来:

- 在一个或多个维上聚合数据
- 在一组表上连接和聚合数据
- 标识通常存取的数据子集（即“热”水平或垂直分区）
- 在分区数据库环境中对表或表的一部分进行重新分区

摘要表的信息已经集成到了“SQL 编译程序”中。在“SQL 编译程序”中，使用了“查询重写”（参见第125页的『由 SQL 编译程序重写查询』）和“优化器”（参见第134页的『数据存取概念和优化』）来将查询与摘要表相匹配以及确定是否为基于基本表的查询替换摘要表。每当使用摘要表来回答查询时，可使用 EXPLAIN 设施（参见第177页的『第7章 SQL 解释设施』）来确定选择了哪一个摘要表。因为摘要表的行为在许多方面类似于常规表，所以有关使用表空间定义来优化存取路径、创建索引以及发出 RUNSTATS 的考虑事项也适用于摘要表。

为了帮助您理解摘要表的功能，我们提供了下面这个多维分析查询的实例，并显示了它如何利用摘要表。

在此示例中，我们假设了这样一个方案: 仓库包含一组客户和一组信用卡帐户。仓库记录使用信用卡进行的交易集。每项交易都包含一批一起购买的物品。我们可将此环境分类成多星型结构，因为两个表（一个包含交易物品，另一个标识购买交易）都很大，并且都是星型的中心。

共有三个分层维描述一项交易：产品、位置和时间。产品层次结构记录在两个分别表示产品组和产品行的标准表中。位置层次结构包含城市、州和国家信息，包含在单个非标准表中。时间层次结构包含日、月和年信息，编码在单一日期字段中。日期维是使用内部函数从交易的日期字段中抽取的。在此方案中，还有其他表示客户帐户信息和客户信息的表。

使用以下每一级的销售总额和销售计数来创建摘要表:

- 产品分级结构
- 位置分级结构
- 时间分级结构，由年、月、日组成。

大的查询范围可从存储的聚合数据中找到答案。下例沿产品组和产品行维；沿城市、州和国家维；以及沿时间维计算销售金额和销售件数。在它的 GROUP BY 子句中，还包括几个其他的列。

```
CREATE TABLE dba.PG_SALESSUM
  AS (
      SELECT l.id AS prodline, pg.id AS pgroup,
             loc.country, loc.state, loc.city,
             l.name AS linename, pg.name AS pgname,
             YEAR(pdate) AS year, MONTH(pdate) AS month,
             t.status,
             SUM(ti.amount) AS amount,
             COUNT(*) AS count
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
         AND YEAR(pdate) > 1990
      GROUP BY l.id, pg.id, loc.country, loc.state, loc.city,
               year(pdate), month(pdate), t.status, l.name, pg.name
    )
DATA INITIALLY DEFERRED REFRESH DEFERRED;

REFRESH TABLE dba.SALESCUBE;
```

摘要表通常比基本事实表小得多。您可以指定 DEFERRED 选择来控制刷新摘要表的时间（如示例所示）。

可利用这类预先计算的总额的查询包括:

- 按月和产品组的销售额
- 1990 年以来的总销售额
- 1995 年或 1996 年的销售额
- 产品组或产品线的销售总额
- 特定产品组或产品线在 1995、1996 年的销售总额

- 特定国家的销售总额。

当摘要表未包括以上任何一个查询的准确答案时，使用摘要表计算答案的成本可能明显地小于使用大型基表的成本，因为此答案的一部分已计算出来。通过摘要表避免或减少了成本高昂的基本数据连接、排序和聚合。

以下是性能明显改进的样本查询，因为它们能使用摘要表中已计算出的结果。第一个示例返回 1995 和 1996 年的销售总额:

```
SET CURRENT REFRESH AGE=ANY

SELECT YEAR(pdate) AS year, SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY year(pdate);
```

第二个示例返回 1995 和 1996 年产品组的销售总额:

```
SET CURRENT REFRESH AGE=ANY

SELECT pg.id AS "PRODUCT GROUP",
       SUM(ti.amount) AS amount
FROM   cube.transitem AS ti, cube.trans AS t,
       cube.loc AS loc, cube.pgroup AS pg,
       cube.prodline AS l
WHERE  ti.transid = t.id
   AND ti.pgid = pg.id
   AND pg.lineid = l.id
   AND t.locid = loc.id
   AND YEAR(pdate) IN (1995, 1996)
GROUP BY pg.id;
```

数据库越大，这样的查询所能获得的响应时间改进也越大。发生这种情况的原因是摘要表的增长速度比基本表的增长速度慢。摘要表的其中一个好处在于，“DB2 通用数据库”使用它们可以有效地消去查询之间的重叠工作，其方法是，在构建摘要表时执行一次计算，并对非常大量的查询重复使用摘要表的内容。

联合体数据库查询编译程序阶段

本节描述联合体数据库系统中附加的查询处理阶段。它还提供可改善联合体数据库查询性能的建议。主要主题包括:

- 『下推分析』
- 第172页的『远程 SQL 生成与全局优化』。

下推分析

下推分析告知 DB2 优化器是否可在远程数据源处执行操作。操作可以是一个函数，如关系运算符、系统函数或用户函数，或 SQL 运算符（GROUP BY 、ORDER BY 等）。

不能下推的函数会显著影响查询性能。考虑强制在本地而不在数据源对选择谓词求值的效果。此方法可能要求 DB2 从远程数据源检索整个表，然后在本地对该谓词进行过滤。如果网络受到限制 — 而该表又很大 — 则会降低查询性能。

未下推的运算符也可能显著地影响查询性能。例如，让 GROUP BY 运算符在本地聚合远程数据，会再次要求 DB2 从远程数据源检索整个表。

例如，假设别名 N1 引用 DB2 OS/390 版数据源中的数据源表 EMPLOYEE。再假设该表有 10,000 行，其中一列包含雇员的姓，一列包含工资。假设以下语句:

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

考虑几种可能的情况:

- 如果 DB2 和 DB2 OS/390 版的整理顺序相同，很可能该查询谓词被下推到 DB2 OS/390 版中。在数据源对结果进行过滤和分组通常更有效，而无需将整个表复制到 DB2 然后在本地执行那些操作。联合体系统中的下推分析确定是否可在数据源执行操作。在这种情况下，该谓词和 GROUP BY 操作可在数据源中执行。
- 如果整理顺序不同，下推分析将确定不能在数据源对整个谓词求值；但优化器可能会决定下推该谓词的 SALARY > 1000 部分。仍然必须在 DB2 进行范围比较。
- 如果整理顺序相同，且优化器知道本地 DB2 服务器非常快，则优化器将决定在 DB2 中本地执行 GROUP BY 操作是最好的（成本最少）方法。将会在数据源中对该谓词求值。这是下推分析结合全局优化的示例。DB2 将考虑可用的路径，然后选择最有效的方案。

通常，目标是确保可考虑函数和运算符由优化器在数据源处求值。许多因素会影响是否可在远程数据源对函数或 SQL 运算符求值。关键因素分三组讨论：服务器特性、别名特性和查询特性。

影响下推机会的服务器特性

下面几节介绍可影响下推机会的数据源因素。一般情况下这些因素都存在，因为 DB2 允许您使用丰富的 SQL 语言来提交查询。与在 DB2 查询期间存取的服务器所支持的 SQL 语言相比，此语言可提供更多的功能。DB2 可弥补数据服务器中函数的不足，但这样做可能要求在 DB2 中执行操作。

SQL 性能: 每个数据源都支持不同的 SQL 语言和不同级别的功能。例如，考虑 GROUP BY 列表。大多数数据源都支持 GROUP BY 运算符；但某些数据源对 GROUP BY 列表中的项数有限制，或者对 GROUP BY 列表上是否允许表达式有限制。如果对远程数据源有限制，DB2 可能只好在本地执行 GROUP BY 操作。

SQL 限制: 每个数据源可能有不同的 SQL 限制。例如，某些数据源要求参数标记将值嵌入远程 SQL 语句中。因此，必须检查参数标记限制以确保每个数据源可支持这种嵌入技术。如果 DB2 确定不了一个好方法来嵌入函数的值，必须在本地对此函数求值。

SQL 限制: DB2 可能允许使用比它的远程数据源所允许的更大的整数；但不能将超出限制的值嵌入发送给数据源的语句。因此，对此常数进行运算的函数或运算符必须在本地求值。

服务器特性: 有几个因素都归入此类别。一个示例是对 NULL 值排序（最高、最低或由定序决定）。例如，如果在某个与 DB2 不同的数据源对 NULL 值排序，则对可空表达式执行的 ORDER BY 操作无法远程求值。

整理顺序: 配置联合体数据库以使用数据源所用的整理顺序，然后将 collating_sequence 服务器选项设置为 'Y'，以允许优化器考虑下推字符型范围比较谓词。

当来自联合体服务器的查询需要排序时，处理排序的地点取决于几个因素。如果联合体数据库的整理顺序与存储被查询数据的数据源的整理顺序相同，可以在数据源进行排序。如果整理顺序相同，优化器可决定完成该查询的最有效方法是本地排序还是在数据源排序。同样，如果查询需要比较字符数据，也可在数据源执行此比较。

通常，即使整理顺序不同，也可在任意一个位置进行数字比较。但是，如果联合体数据库和数据源的空字符加权不同，可能得到不正常的结果。同样，对于比较

语句，要注意是否将语句提交到不区分大小写的数据源。在不区分大小写的数据源中指定给字符 ″I″ 和 ″i″ 的加权是一样的。缺省情况下，DB2 是区分大小写的，并且会给这些字符指定不同的加权。

如果联合体数据库和数据源的整理顺序不同，DB2 将对联合体数据库检索数据，以便可在本地对该数据排序和比较。因为用户希望看到根据对联合体服务器定义的整理顺序排序的查询结果；所以在本地对数据排序，联合体服务器可确保满足这个期望。

检索数据以进行本地排序和比较通常会降低性能。因此，可考虑将联合体数据库配置为使用数据源所用的整理顺序。这样，可能会增加性能，因为联合体服务器可允许在数据源进行排序和比较。例如，在 DB2 UDB OS/390 版中，由 ORDER BY 子句定义的排序用基于 EBCDIC 代码页的整理顺序来实现。如果希望使用联合体服务器来检索根据 ORDER BY 子句排序的 DB2 OS/390 版数据，建议配置联合体数据库以让它使用基于 EBCDIC 代码页的预定义整理顺序。

如果联合体数据库的整理顺序与数据源的不同，并且需要查看以数据源的顺序排序的数据，可用通过方式提交您的查询，或者在数据源视图中定义该查询。

有关整理顺序以及如何设置它们的详情，参见*管理指南: 计划*；有关 collating_sequence 服务器选项的详情，参见第89页的表8。

服务器选项: 几个服务器选项可影响下推机会。尤其要复查 *collating_sequence*、*varchar_no_trailing_blanks* 和 *pushdown* 的设置。有关设置这些选项的信息，参见第88页的『影响联合体数据库查询的服务器选项』。

DB2 类型映射和函数映射因素: 设计 DB2 提供的缺省本地数据类型映射（有关数据类型表，参见 *Application Development Guide*），是为了给每个数据源数据类型提供足够的缓冲区空间（以避免数据丢失）。用户可选择为特定数据源定制类型映射以适合特定的应用程序。例如，如果要存取具有 DATE 数据类型（缺省情况下它被映射为 DB2 TIMESTAMP 数据类型）的 Oracle 数据源列，可将本地数据类型更改为 DB2 DATE 数据类型。

DB2 可补偿数据源不支持的函数。有三种情况会发生函数补偿:

- 远程数据源中不存在此函数。
- 该函数确实存在；但操作数特性违反了函数限制。这种情况的一个例子是 IS NULL 关系运算符。大多数数据源支持它，但某些数据源可能有限制，如只允许 IS NULL 运算符左边存在列名。
- 如果对该函数远程求值，可能返回一个不同的结果。这种情况的一个例子是 '>'（大于）运算符。对于那些具有不同整理顺序的数据源，如果由 DB2 在本地对大于运算符求值，它可能会返回不同的结果。

影响下推机会的别名特性

下面几节介绍可影响下推机会的别名因素。

别名列的本地数据类型: 确保列的本地数据类型没有妨碍在数据源对谓词求值。如先前提到的，提供缺省数据类型映射以避免任何可能的溢出。但是，可能不会考虑在连接列较短的数据源处用谓词连接两个不同长度的列，这取决于 DB2 如何在较长的列中进行连接。这种情况会影响由 DB2 优化器确定的连接顺序中存在的可能性数目。例如，将类型 NUMBER(38) 赋予使用 INTEGER 或 INT 数据类型创建的 Oracle 数据源列。将本地数据类型 FLOAT 赋予此 Oracle 数据类型的别名列，因为 DB2 整数的范围是从 2**31 到 (-2**31)-1，它近似等于 NUMBER(9)。在这种情况下，不能在 DB2 数据源（更短的连接列）将 DB2整数列和 Oracle 整数列连接；但是，如果 DB2 INTEGER 数据类型可容纳此 Oracle 整数列的域，则用 ALTER NICKNAME 语句更改其本地数据类型，以便该连接可在 DB2 数据源进行。

列选项: ALTER NICKNAME SQL 语句可用于添加或更改别名的列选项。

其中一个选项是 ″varchar_no_trailing_blanks″。它可用于标识不包含尾部空格的列。当检查对这样标明的列执行的所有操作时，编译程序下推分析步骤将会考虑此信息。根据该指示，DB2 可能生成另一个形式等效的谓词以用于发送到数据源的远程 SQL 语句。用户可能会看到依靠数据源求值的另一个谓词，但最终结果应是等效的。

另一个列选项是 numeric_string。使用此选项来指示该列中的值是否总是不带尾部空格的数值。

有关列选项值和缺省值，参见表15。

表 15. 列选项及其设置

选项	有效设置		缺省设置
numeric_string	‘Y’	是，此列只包含数字数据串。要点: 如果此列只包含数字串且带尾部空格，建议不要指定 ‘Y’。	‘N’
	‘N’	否，此列不限于数字数据串。	
	将一列的 numeric_string 设置为 ‘Y’，以告知优化器此列不包含可能干扰列数据排序的空格。当数据源的整理顺序与 DB2 的不同时，此选项可发挥作用。由于整理顺序不同，将从本地（数据源）求值中排除用此选项标记的那些列。		

表 15. 列选项及其设置 (续)

选项	有效设置	缺省设置
varchar_no_trailing_blanks	指示特定的 VARCHAR 列是否缺少尾部空格: 'Y' 是，此 VARCHAR 列缺少尾部空格。 'N' 否，此 VARCHAR 列不缺少尾部空格。 如果数据源的 VARCHAR 列不包含填充的空格，那么优化器用于存取这些列的策略在一定程度上取决于这些列是否包含尾部空格。缺省情况下，优化器“假定”这些列确实包含尾部空格。基于这种假设，它制定了一个存取策略，包括修改查询，以使从这些列返回的值是用户期望的值。但是，如果 VARCHAR 列没有尾部空格，而您又让优化器知道这点，它可能会制定一个更有效的存取策略。要告诉优化器特定的列没有尾部空格，在 ALTER NICKNAME 语句中指定该列（有关语法，参见 *SQL Reference*）。	'N'

影响下推机会的查询特性

查询可引用一个 SQL 运算符，它可能涉及多个数据源中的别名。当 DB2 必须使用一个运算符如集合运算符（例如，UNION）来组合来自两个被引用的数据源的结果时，该操作必须在 DB2 中执行。不能在远程数据源直接对该运算符求值。

分析和了解下推分析决策

重新编写 SQL 语句可为 DB2 查询处理提供其他下推机会。本节介绍用于确定在何处对查询求值的工具，列出与查询分析有关的常见问题（和建议的审查范围），最后简要介绍数据源升级问题。

分析在何处对查询求值: DB2 提供了两个实用程序，它们显示在哪里对查询求值:

- Visual explain。用 **db2cc** 或 **db2vexp** 命令启动它。可用它来查看查询存取方案图。每个运算符的执行位置都包括在该运算符详细显示的内容中。

  如果完全下推查询，应在 RQUERY 运算符的顶部看到 RETURN 运算符。RETURN 运算符是标准的 DB2 运算符；RQUERY 运算符是联合体数据库操作所独有的运算符。RQUERY 将 SQL SELECT 语句发送到数据源以检索查询结果。该 SELECT 语句是使用数据源支持的 SQL 语言生成的。它可包含数据源的任何有效的查询。
- SQL 解释。用 **db2expln** 或 **dynexpln** 命令启动它。可用它来查看文本格式的存取方案策略。

了解为什么在数据源或 DB2 中对查询求值: 本节列出典型的方案分析问题和要审查的范围以增加下推机会。主要问题包括:

- 为什么不对此谓词远程求值？

当谓词选择性很强并因此可用于过滤行和减少网络通信量时，会出现此问题。远程谓词求值还影响是否可对同一数据源的两个表之间的连接进行远程求值。

要检查的范围包括:

- 子查询谓词。此谓词包含与另一个数据源相关的子查询吗？此谓词包含涉及此数据源不支持的 SQL 运算符的子查询吗？并非所有数据源都支持谓词中的集合运算符。
- 谓词函数。此谓词包含此远程数据源不能求值的函数吗？关系运算符也被归类为函数。
- 谓词连接需求。如果进行远程求值，此谓词要求嵌入某些值吗？如果是这样，它会违反此数据源的 SQL 限制吗？
- 全局优化。优化器可能已经决定本地处理更节省成本。有关详情，参见第172页的『远程 SQL 生成与全局优化』。

• 为什么不对 GROUP BY 运算符进行远程求值？

有几个方面可检查:

- 对 GROUP BY 运算符的输入进行远程求值吗？如果回答是否定的，则检查输入。
- 数据源对此运算符有任何限制吗？例如:
  - GROUP BY 的有限项数
  - 组合的 GROUP BY 项的有限字节计数
  - 列说明仅存在于 GROUP BY 列表上
- 数据源支持此 SQL 运算符吗？
- 全局优化。优化器可能已经决定本地处理更节省成本。有关详情，参见第172页的『远程 SQL 生成与全局优化』。

• 为什么不对集合运算符进行远程求值？

有几个方面可检查:

- 它的两个操作数全在同一远程数据源中求值吗？答案应该是肯定的，如果不是，检查每个操作数。
- 数据源对此集合运算符有任何限制吗？例如，大对象或长整数字段是否是此特定集合运算符的有效输入？

• 为什么不对 ORDER BY 操作进行远程求值？

考虑:

- 对 ORDER BY 操作的输入进行远程求值吗？如果回答是否定的，则检查输入。
- ORDER BY 子句包含字符表达式吗？如果是，那么远程数据源没有与 DB2 相同的整理顺序吗？

– 数据源对此运算符有任何限制吗？例如，ORDER BY 的项数受限制吗？数据源限制只有 ORDER BY列表才能有列说明吗？

数据源升级与定制: 虽然 DB2 SQL 编译程序具有大量有关数据源 SQL 支持的信息，但是此数据在一段时间之后可能需要调整，因为数据源可能已升级和／或定制。在这种情况下，通过更改本地目录信息来向 DB2 提供增强信息。使用 DB2 DDL 语句（如 CREATE FUNCTION MAPPING 和 ALTER SERVER）来更新目录。有关详情，参见 *SQL Reference*。

远程 SQL 生成与全局优化

此阶段产生对查询求值所用的全局优化存取策略。对于联合体数据库查询，该存取策略可能会将原来的查询分为一组远程查询单元，然后组合结果。

将下推分析的输出用作建议，优化器可决定是要在 DB2 本地还是在数据源远程对每个操作进行求值。此决定基于它的成本模型的输出，此模型不但包括对操作求值的成本，还包括在 DB2 和数据源之间传送数据或信息的成本。

目标是产生优化的查询；但许多因素可能影响全局优化的输出，因而也影响查询性能。将关键因素分成两组讨论：服务器特性和别名特性。

影响全局优化的服务器特性／选项

可影响全局优化的数据源服务器因素包括:

- CPU 速度的相对比率

  使用 *cpu_ratio* 服务器选项来指示与 DB2 CPU 相比，数据源 CPU 速度快多少或慢多少。低比率指示数据源工作站 CPU 比 DB2 工作站 CPU 快。如果是低比率，DB2 优化器更有可能考虑将需要占用大量 CPU 资源的操作下推到数据源。有关此比率的详情，参见第88页的『影响联合体数据库查询的服务器选项』。

- I/O 速度的相对比率

  使用 *io_ratio* 服务器选项来指示与 DB2 系统相比，数据源系统 I/O 速度快多少或慢多少。低比率指示数据源工作站 I/O 速度比 DB2 工作站 I/O 速度快。对于低比率，DB2 优化器会考虑将需要占用大量 I/O 资源的操作下推到数据源。有关此比率的详情，参见第88页的『影响联合体数据库查询的服务器选项』。

- DB2 和数据源之间的通信速率

  使用 *comm_rate* 服务器选项来指示网络容量。低速率（指示 DB2 和数据源之间的网络通信较慢）建议 DB2 优化器减少与此数据源之间传送的信息量。如果将该速率设置为 0，优化器将产生需要最小网络通信量的查询。有关此比率的详情，参见第88页的『影响联合体数据库查询的服务器选项』。

- 数据源整理顺序

  使用 *collating_sequence* 服务器选项来指示数据源整理顺序是否与本地 DB2 数据库整理顺序匹配。如果未将此选项设置为 'Y'，则优化器将认为从此数据源检索的数据是未排序的。有关整理顺序性能问题的详情，参见第167页的『整理顺序』。

- 远程方案提示

  使用 *plan_hints* 服务器选项来指示数据源是否支持方案提示。方案提示是为数据源优化器提供额外信息的语句段。对于特定查询类型，此信息可改进查询性能。方案提示可帮助数据源优化器决定是否使用索引、使用哪个索引或使用哪种表连接顺序。

  如果启用方案提示，则发送到数据源的查询将包含附加的信息。例如，带方案提示发送到 Oracle 优化器的语句可能类似于如下所示:

  ```
  SELECT /*+ INDEX (table1, t1index)*/
     col1
     FROM table1
  ```

  该方案提示是字符串 `/*+ INDEX (table1, t1index)*/`。

- DB2 优化器知识库中的信息

  DB2 有一个包含本地数据源数据的优化器知识库。DB2 优化器不会生成特定 DBMS 无法生成的远程存取方案。换句话说，DB2 避免生成在远程数据源的优化器不能理解或接受的方案。

影响全局优化的别名特性

下面几节包含可影响全局优化的别名因素。

索引考虑事项: DB2 可使用数据源处的索引信息来优化查询。由于此原因，DB2 可用的索引信息应是最新的，这点很重要。在创建别名时最初要获取别名的索引信息。不收集视图别名的索引信息。

创建别名的索引规范: 可为别名创建索引规范。索引规范在目录中构建索引定义（而不是实际的索引），以供 DB2 优化器使用。使用 CREATE INDEX SPECIFICATION ONLY 语句来创建索引规范。基于别名创建索引规范的语法与在本地表上创建索引的语法相似。有关详情，参见*管理指南: 计划*。

出现以下情况时，可考虑创建索引规范:

- 在创建别名期间，DB2 无法检索数据源的任何索引信息。
- 需要视图别名的索引。
- 想鼓励 DB2 优化器将特定的别名用作嵌套循环连接的内部表。如果连接列不存在索引，用户可创建一个。

在基于视图的别名发出 CREATE INDEX 语句之前考虑您的需要。一种情况是，如果视图是对一个有索引的表执行的简单 SELECT，那么在本地基于别名创建与数据源表上的索引匹配的索引可显著改善查询性能。但是，如果是在本地对由并非简单的选择语句组成的视图（例如，通过连接两个表创建的视图）创建索引，可能降低查询性能。例如，如果对通过两个表的连接构成的视图创建索引，优化器可能会选择该视图作为嵌套循环连接中的内部元素。该查询的性能会很差，因为要对连接求值几次。替代方法是为数据源视图中引用的每个表创建别名，并在 DB2 中创建一个本地视图来引用这两个别名。

目录统计信息考虑事项: 目录统计信息描述相关列中别名的总大小和值的范围。当优化器计算最低成本路径以处理包含别名的查询时会使用这些信息。别名统计信息与表统计信息存储在同一目录视图中。有关统计信息类型及如何在本地更新它们的详情，参见第93页的『第5章 系统目录统计信息』和第113页的『更新表和别名统计信息的规则』。

当 DB2 可检索数据源中保存的统计数据时，它不能自动检测对数据源的现存统计数据的更新。另外，DB2 没有办法处理数据源对象的对象定义或结构更改（添加列）。如果更改了对象的统计数据或结构数据，则有两个选择:

- 在数据源处运行 RUNSTATS 的等效命令。然后，卸下当前别名。重新创建别名。如果结构信息已更改，则使用此方法。
- 人工更新 SYSSTAT.TABLES 视图中的统计信息。此方法需要的步骤更少，但如果结构信息更改，它将不起作用。

分析和了解全局优化决策

本节介绍用于分析查询优化的工具，并讨论与查询优化相关的常见问题（及建议的审查范围）。

分析查询优化: DB2 提供了两个实用程序，它们显示全局存取方案:

- Visual explain。用 **db2cc** 或 **db2vexp** 命令启动它。可用它来查看查询存取方案图。每个运算符的执行位置都包括在该运算符详细显示的内容中。还可在 RQUERY（选择操作）运算符中找到为每个数据源生成的远程 SQL 语句。通过检查每个运算符的详细资料，可了解 DB2 优化器估计的作为每个运算符输入输出的行数。还可了解执行每个运算符的估计成本，包括通信成本在内。有关详情，参见第475页的『附录C. SQL 解释工具』。
- SQL 解释。用 db2expln 或 dynexpln 命令启动它。可用它来查看文本格式的存取方案策略。SQL 解释没有提供成本信息；但可获得远程优化器为远程解释函数支持的那些数据源生成的存取方案。有关详情，参见第475页的『附录C. SQL 解释工具』。

了解 DB2 优化决策: 本节列出要审查的优化问题和关键区域以改善性能。主要问题包括:

- 为什么不对同一数据源的两个别名之间的连接进行远程求值?

  要检查的范围包括:

  – 连接操作。数据源能支持它们吗?

  – 连接谓词。能在远程数据源对连接谓词求值吗?如果回答是否定的,则检查连接谓词。有关详情,参见第170页的『了解为什么在数据源或 DB2 中对查询求值』。

  – 连接结果(visual explain 生成的)中的行数。该连接产生比组合的两个别名更大的行集合吗?这些数值有意义吗?如果回答是否定的,则考虑人工更新别名统计信息 (SYSSTAT.TABLES)。

- 为什么不对 GROUP BY 运算符进行远程求值?

  要检查的范围包括:

  – 运算符语法。验证可在远程数据源对该运算符求值。有关详情,参见第170页的『了解为什么在数据源或 DB2 中对查询求值』。

  – 行数。使用 Visual Explain 检查 GROUP BY 运算符输入和输出中的估计行数。这两个数非常接近吗?如果是,则 DB2 优化器认为在本地对此 GROUP BY 求值更有效。而且,这两个数值有意义吗?如果回答是否定的,则考虑人工更新别名统计信息 (SYSSTAT.TABLES)。

- 为什么远程数据源没有完全对该语句求值?

  DB2 优化器基于成本进行优化。即使下推分析指示每个运算符都可在远程数据源求值,优化器仍根据其成本估计来生成全局优化方案。有很多因素可对该方案产生影响。例如,即使远程数据源可处理原始查询中的每个操作,但其 CPU 速度远低于 DB2 的速度,因此在 DB2 执行这些操作可能会更有利。如果结果不理想,验证 SYSCAT.SERVEROPTIONS 中的服务器统计信息。

- 为什么由优化器生成的,且完全在远程数据源求值的方案,其性能比直接在远程数据源中执行的原始查询的性能要差很多?

  要检查的范围包括:

  – DB2 优化器生成的远程 SQL 语句。确保它等于原来的查询。检查谓词定序更改。一个好的查询优化器不应对查询的谓词顺序敏感;但不幸的是,并非所有的 DBMS 优化器都一样,因此很可能远程数据源的优化器会根据输入谓词定序生成不同的方案。如果是这样,这是远程优化器内在的问题。考虑修改 DB2 输入的谓词定序,或与远程数据源的服务机构联系以寻求帮助。

还要检查是否替换过谓词。一个好的查询优化器不应对等效的谓词替换敏感；但不幸的是，并非所有的 DBMS 优化器都一样，因此远程数据源的优化器可能会根据输入谓词生成不同的方案。例如，某些优化器不能为谓词生成传递闭合语句。

– 返回的行数。可从 Visual Explain 获得此数值。如果查询返回大量的行，网络通信量是一个潜在的瓶颈。

– 附加函数。与原始查询相比，远程 SQL 语句包含附加函数吗？可能生成一些附加函数以转换数据类型。确保它们是必需的。

第7章 SQL 解释设施

SQL 解释设施是 SQL 编译程序的一部分，可用来捕捉静态或动态 SQL 语句编译环境的信息。捕捉的信息使您可了解 SQL 语句的结构和潜在的执行性能，包括:

- 处理查询的操作顺序
- 成本信息
- 谓词和可选择性估计
- 对在解释时 SQL 语句中引用的所有对象的统计信息。

此信息可帮助您:

- 了解为查询选择的执行方案
- 辅助设计应用程序
- 确定应用程序应何时重新联编
- 辅助数据库设计。

提供下列主题:

- 『选择解释工具』
- 第179页的『使用 SQL 解释设施』
- 第181页的『解释的基本概念』
- 第183页的『解释信息的组织』
- 第188页的『获取解释数据』
- 第191页的『使用解释输出的准则』
- 第192页的『Visual Explain』
- 第193页的『SQL 建议设施』.

解释输出存储在关系表中，也可以选择使用某种格式存储以便可用 Visual Explain 工具以图形方式显示。您应考虑使用解释表来查找那些对您感兴趣的解释表运行的查询。有关解释设施使用的表和如何创建那些表的详情，参见第443页的『附录 B. 解释表和定义』。

选择解释工具

DB2 提供了业界最完备的解释设施，并对解释 SQL 语句选择的存取方案提供了详细的优化器信息。提供了若干种方法，使您可灵活地捕捉和存取解释信息。

详细的优化器信息允许对存取方案进行深入的分析，该信息保存在解释表中，它是与实际的存取方案分开保存的。有三种方法来从解释表获取信息:

1. 编写自己的查询（根据第443页的『附录B. 解释表和定义』中显示的解释表说明）
2. 使用 *db2exfmt* 工具
3. 使用 Visual Explain（以查看解释快照信息）

解释表在所有受支持的平台上都是可存取的，它包含静态和动态 SQL 语句的信息。您可使用 SQL 语句存取解释表，这种方式允许方便地处理输出，并可比较不同的查询，或比较一段时间内的同一查询。如果希望用预定义格式显示解释表中的信息，可使用 *db2exfmt* 工具。有关此工具的详情，参见第517页的『附录D. db2exfmt - 解释表格式的工具』。另外，您需要创建自己的语句来存取这些表。

注: 此工具（以及其他工具如 *db2batch*、*dynexpln*、*db2vexp* 和 *db2_all*）的位置在 *sqllib* 目录的 *misc* 子目录中。如果此工具已从此路径转移至别处，则上面提到的命令行项可能不起作用。

Visual Explain 允许通过一个图形界面来分析存取方案和解释表中的优化器信息。可以使用此工具来分析静态和动态 SQL 语句。Visual Explain 通常是从控制中心内调用的。控制中心可通过从命令行输入 *db2cc* 来启动。对于单个 SQL 语句，也可使用 *db2vexp* 命令直接从命令行调用 Visual Explain。在某些平台上，可从 “DB2 通用数据库” 文件夹内使用文件夹调用 Visual Explain。Visual Explain 不是在所有支持的平台上都可用。您应检查适合您平台的*快速入门*手册，来查看它是否支持 Visual Explain。Visual Explain 允许您查看在另一个平台上捕捉的快照。例如，Windows NT 客户机可以将在 DB2 HP-UX 版服务器上生成的快照绘成图形。为此，两个平台都必须处于版本 5 级别或更高级别。不易于对 Visual Explain 的输出进行处理来进行进一步的分析，其他应用程序也无法存取该信息。有关 *db2vexp* 命令的详情，在命令行上输入 *db2vexp -h*，或参阅 *Command Reference* 手册。有关 Visual Explain 的其他信息，应通过输入 *db2cc* 来参考控制中心中的联机帮助。

作为程序包的一部分，有关静态 SQL 语句存取方案的信息是在系统目录中生成和存储的。要查看一个或多个程序包可用的存取方案信息，可从命令行使用 *db2expln* 工具。*db2expln* 显示所选存取方案的实际实施。它不显示优化器信息。

dynexpln 工具在自己内部使用 *db2expln*，该工具提供一种快速方法来解释不包含参数标记的动态 SQL 语句。要从 *dynexpln* 内使用 *db2expln*，可将输入 SQL 语句转换为一个伪程序包内的静态语句。当执行此操作时，该信息可能不是始终都是完全准确的。如果希望完全准确，应使用解释设施，如第179页的『使用 SQL 解释设施』所述。

db2expln 工具通过检查生成的实际存取方案（有关如何生成该代码的详情，参见124），来对在运行期将发生的操作提供一个相对简洁的、英国风格的概述。有关使用 *db2expln* 和解释该输出的其他详情可在第475页的『附录C. SQL 解释工具』中找到。

表16汇总了 DB2 解释设施可使用的不同工具以及它们的个别特征。参考此表来选择最适合您的环境和需要的工具。

表 16. 解释设施工具

期望的特征	Visual Explain	db2vexp	解释表	db2exfmt	db2expln	dynexpln
GUI 界面	是	是				
文本输出				是	是	是
“快速和脏的”静态 SQL 分析					是	
支持的静态 SQL	是		是	是	是	
支持的动态 SQL	是	是	是	是		是*
支持的 CLI 应用程序	是		是	是		
可用于 DRDA 应用请求器			是			
详细的优化器信息	是	是	是	是		
适合对多个语句的分析			是	是	是	是
可从应用程序内存取的信息			是			
注:						
* 间接使用 db2expln; 存在一些限制。						

使用 SQL 解释设施

捕捉解释信息的不同方法包括使用:

1. EXPLAIN 和 EXPLSNAP BIND/PREP 选项
2. CURRENT EXPLAIN MODE 和 CURRENT EXPLAIN SNAPSHOT 专用寄存器
3. EXPLAIN SQL 语句
4. *db2vexp* 工具（也直接调用 Visual Explain 来显示该信息）

您可能要收集和使用解释数据的原因有以下三种:

1. 了解数据库管理程序为满足您的查询而必须执行的步骤（存取方案）。 第134页的『数据存取概念和优化』提供了当您希望了解解释输出时可能需要参考的信息。

2. 帮助评估您的性能调整动机。您可执行许多操作来帮助改善查询性能。在下列主题的子标题中描述了多个这种可能的操作:
   - 第37页的『第3章 应用程序考虑事项』
   - 第77页的『第4章 环境考虑事项』
   - 第93页的『第5章 系统目录统计信息』。

   当在其中任何一个区域中进行更改后，您可使用 SQL 解释设施来确定该更改对所选存取方案的影响（如果有的话）。例如，如果您根据第82页的『索引对查询优化的影响』中提供的建议添加了一个索引，则解释数据可帮助您确定是否正按预期的那样使用该索引。

   当解释输出提供信息以允许您确定选择的存取方案及其相关成本时，准确测量一个查询的性能改进的唯一方法是使用基准测试技术，如第265页的『第12章 基准测试』中所述。
3. 为了帮助您了解查询性能更改的原因，您需要拥有更改前后的解释信息以分析该影响。因此，当编译对数据库执行的一条 SQL 语句时，您应:
   - 使用解释设施来捕捉更改前的方案信息，并保存生成的解释表；或者，保存 `db2exfmt` 解释工具的输出。
   - 如果您不想或不能存取 Visual Explain 来查看此信息，则保存和／或打印当前的目录统计信息。（可使用第118页的『建立生产数据库的模型』中描述的 `db2look` 生产工具来帮助执行此任务。）
   - 保存和／或打印数据定义语言 (DDL) 语句，包括用于 CREATE TABLE、CREATE VIEW、CREATE INDEX 和 CREATE TABLESPACE 的那些语句。

   以上信息为您提供了更改*前*的映象，您可将它用作以后分析的参考点。对于动态 SQL 语句，您也可在首次运行您的应用程序时收集此信息。对于静态 SQL 语句，您也可在联编时收集此信息。

   当您要分析性能更改的原因时，可将*前*数据与您启动分析时收集的有关查询和环境的信息（*后*数据）比较。

   举一个简单的例子，您的分析可能显示不再将一个索引用作存取路径的一部分。使用 Visual Explain 中的目录统计信息，您可能注意到索引层数（NLEVELS 列）现在大大高于当该查询首次与数据库联编时的索引层数。于是，您可能选择:
   - 重组该索引
   - 收集您的表和索引的新统计信息
   - 当重新联编您的查询时收集解释信息。

执行这些操作后，您可能注意到在存取方案中再次使用了该索引，且该查询的性能不再是问题。

解释的基本概念

您可以使用解释信息来分析优化器根据第134页的『数据存取概念和优化』中描述的选项选择的存取方案。例如，解释信息可能指示优化器选择了索引扫描（参见第134页的『索引扫描概念』）。此外，它还可允许您确定下列各项:

- 有多少个索引列用作搜索标准，如第143页的『范围定界谓词和索引 SARGable 谓词』中所述
- 是否使用纯索引存取，如第139页的『纯索引存取』中所述
- 是否使用列表预读取来读取页，如第216页的『了解列表预读取』中所述。

另一个示例是，解释信息也可帮助您了解两个表是如何连接的:

- 连接方法
- 连接表的次序
- 排序的实例和类型。

虽然您可对 SELECT、SELECT INTO、UPDATE、INSERT、VALUES、VALUES INTO 和 DELETE SQL 语句使用解释，但解释的主要用途是观察语句的 SELECT 部分的存取路径。

要满足 SQL 查询，数据库管理程序通常:

- 使用一个或多个数据对象（表和／或索引）
- 执行一个或多个操作（例如，表扫描、索引扫描和连接）
- 将结果集返回到调用应用程序。

对于简单的 SQL 查询，如:

```
SELECT DEPTNO, DEPTNAME
  FROM DEPARTMENT
```

所执行的步骤的图形表示可由 Visual Explain 显示:

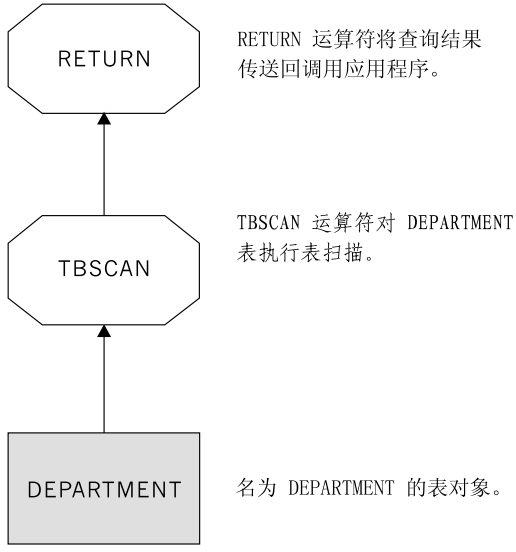


图 20. 解释输出的图形显示

下列主题讨论您可查看的对象和运算符细节的类型:

- 『数据对象的解释信息』
- 第183页的『数据运算符的解释信息』

数据对象的解释信息

单个存取方案可使用一个或多个数据对象来满足 SQL 语句。

对象统计信息: 解释设施记录有关该对象的事实, 如:

- 创建时间
- 上次收集该对象的统计信息的时间（参见第93页的『第5章 系统目录统计信息』）
- 有关是否将对象中的数据排序的指示
- 对象中的列数
- 对象中的估计行数
- 对象占用缓冲池的页数
- 对于此对象存储所在的指定表空间, 所发生的单个随机 I/O 的估计总额外开销（以毫秒计）
- 从指定的表空间读取 4K 页的估计传送速率（以毫秒计）
- 预读取大小和数据块大小（以 4K 页计）
- 使用索引群集数据的程度

- 此对象的索引使用的叶页数和树中的级别数
- 此对象的索引中相异完整关键字值的数目
- 表中溢出记录的总数。

数据运算符的解释信息

单个存取方案可对数据执行几个运算，来满足 SQL 语句并将结果返回给您。SQL 编译程序确定必需的操作；例如，表扫描、索引扫描、嵌套的循环连接或 group-by。第134页的『数据存取概念和优化』中提供了多个这种运算符的详情。

除显示存取方案中使用的各种运算符外，还可获得每个运算符的解释信息以及存取方案的累计效果。

估计成本信息：可以显示以下估计的运算符累计成本。这些成本是所选存取方案的成本，包括捕捉其信息的运算符的成本。

- 总计成本（以 timerons 计）
- 页 I/O 数
- CPU 指令数
- 获取第一行的成本（以 timerons 计），包括必需的任何初始额外开销
- 通信成本（以帧计）。

Timerons 是编制的相对计量单位。

运算符特性：解释设施记录下列信息来描述每个运算符的特性:

- 已存取的表的集合
- 已存取的列的集合
- 对数据排序所依据的列，如果优化器确定后续运算符可使用此排序
- 已应用的谓词的集合
- 将返回的估计行数（基数）。

解释信息的组织

所有解释信息都是按解释实例的概念组织的。一个解释实例代表解释设施对一条或多条 SQL 语句的一次调用。一个解释实例代表下列各项的解释信息:

- **静态** SQL 语句的一个程序包中所有入选的 SQL 语句
- 增量联编 SQL 语句的一个特定的 SQL 语句
- **动态** SQL 语句的一个特定的 SQL 语句
- 每个 EXPLAIN SQL 语句（无论是动态还是静态）。

在一个解释实例中捕捉的解释信息包括 SQL 编译环境以及为满足要编译的 SQL 语句所选择的存取方案。解释信息分成 3 个子集:

解释实例信息 为每个解释实例捕捉的编译环境信息。

解释快照信息 Visual Explain 使用的信息。

解释表信息 当请求解释表信息时收集的信息。

解释实例信息

解释实例信息存储在 EXPLAIN_INSTANCE 表中。有关在一个解释实例中解释的每条 SQL 语句的其他特定信息存储在 EXPLAIN_STATEMENT 表中。

解释实例标识: 您可以唯一地标识每个解释实例，并将 SQL 语句的信息与该工具使用此信息进行的指定调用相关:

- 请求解释信息的用户
- 解释请求开始的时间
- 解释的 SQL 语句来自的程序包的名称
- 解释的 SQL 语句来自的程序包的模式
- 有关一个快照是否是解释请求的一部分的指示。

环境设置: 捕捉有关 SQL 编译程序是如何优化您的查询的环境信息。环境信息包括下列各项:

- 正使用的 DB2 级别的版本号和发行版号。
- 用于编译该查询的并行度。可使用 CURRENT DEGREE 专用寄存器、DEGREE 联编选项、SET RUNTIME DEGREE API 和 *dft_degree* 配置参数，来确定当编译特定查询时要使用的并行度。
- 该 SQL 语句是动态的还是静态的。
- 用于编译该查询的查询优化级别。有关详情，参见第57页的『调整优化级别』。
- 当编译查询时指定的游标分块的类型。有关游标的详情，参考 *SQL Reference* 手册。有关游标分块的详情，参见第67页的『行分块』。
- 当编译该查询时使用的隔离级别。有关详情，参见第37页的『并行性』。
- 编译该查询时各种配置参数的值。有关可影响查询优化的配置参数的详情，参见第77页的『影响查询优化的配置参数』，包括当执行解释快照时记录的下列参数:
  - 第290页的『缓冲池大小 (buffpage)』
  - 第303页的『排序堆大小 (sortheap)』
  - 第332页的『活动应用程序的平均数 (avg_appls)』

– 第292页的『数据库堆 (dbheap)』
– 第297页的『锁定列表的最大存储器 (locklist)』
– 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』
– 第400页的『CPU 速度 (cpuspeed)』
– 第400页的『通信带宽 (comm_bandwidth)』。

SQL 语句标识: 对于每个解释实例，可能已解释了多条 SQL 语句。下列信息与唯一标识解释实例的信息一起帮助标识每个个别的 SQL 语句。

- 语句的类型: SELECT、DELETE、INSERT、UPDATE、定位的 DELETE、定位的 UPDATE。
- 该程序包中发出该 SQL 语句的语句和节号，它们记录在 SYSCAT.STATEMENTS 目录视图中。

在 EXPLAIN_STATEMENT 表内，QUERYTAG 和 QUERYNO 字段包含标识符，且是作为解释过程的一部分为您设置的。

对于在一个 CLP 或 CLI 对话期间提交的动态解释 SQL 语句，当 EXPLAIN MODE 或 EXPLAIN SNAPSHOT 是活动时，将 QUERYTAG 设置为 ″CLP″ 或 ″CLI″。当进行此操作时，对于每个语句，QUERYNO 缺省为按 1 或更大的值递增的一个数。

对于所有其他动态解释 SQL 语句（不是来自 CLP、CLI 或使用 EXPLAIN SQL 语句），则将 QUERYTAG 设置为空白，且 QUERYNO 将始终是 ″1″。

成本估计: 对于解释的每个语句，记录执行所选存取方案的相关成本估计值。此成本是使用一种称为 *timerons* 的编制的相对计量单位来给出的。由于下列原因，**不**提供经过时间估计值:

- SQL 优化器不估计经过时间，而估计资源消耗。
- 优化器并非将可影响经过时间的所有因素都考虑在模型内；它忽略不影响存取方案效率的那些因素。经过时间**确实**受到运行期因素数目的影响，包括: 系统工作负荷；资源争用量；并行处理和 I/O 的量；将行返回至用户的成本；以及客户机和服务器之间的通信时间。

语句文本: 对于解释的每个语句，记录 SQL 语句文本的两个版本。一个版本是 SQL 编译程序接收到的文本。另一个版本是从查询的内部编译程序表示反向转换的语句文本的版本。此转换看上去类似于其他 SQL语句，但它**不**必遵循正确的 SQL 语法，也不必从整体上反映内部表示的实际内容。提供此转换，只是为了让您了解 SQL 优化器选择该存取方案所依据的 SQL 内容。将用户编写的语句文本与 SQL 语句的内部表示比较，可帮助您了解 SQL 编译程序如何重新编写您的查询来

获得更好的优化。（参见第125页的『由 SQL 编译程序重写查询』。）它也显示该环境中影响您的语句的其他元素，如触发器和约束。此“优化的”文本使用的一些关键字包括:

$Cn 派生的列的名称，其中，n 表示整数值。

$CONSTRAINT$ 一个标记，用于指示在编译期间添加至原始 SQL 语句的约束的名称。它与 $WITH_CONTEXT$ 前缀一起使用。

$DERIVED.Tn 派生表的名称，其中，n 表示整数值。

$INTERNAL_FUNC$ 一个标记，用于指示 SQL 编译程序使用的一个函数存在，该函数用于解释的查询，但不可通用。

$INTERNAL_PRED$ 一个标记，用于指示在编译解释的查询期间 SQL 编译程序添加的一个谓词存在。这类谓词也不可通用。该编译程序使用一个内部谓词来满足作为触发器和约束的结果添加至原始 SQL 语句的附加内容。

RID 一个标记，用于标识特定行的“行标识符”(RID) 列。

$TRIGGER$ 一个标记，用于指示在编译期间添加至原始 SQL 语句的触发器的名称。它与 $WITH_CONTEXT$ 前缀一起使用。

$WITH_CONTEXT$(...) 当已将附加触发器或约束添加至原始 SQL 语句中后，此前缀将出现在文本开始处。此前缀之后将出现影响 SQL 语句的编译和解析的任何触发器或约束的名称列表。

解释快照信息

当请求解释快照时，要记录附加解释信息，它们描述 SQL 优化器选择的存取方案。此信息使用 Visual Explain 需要的格式存储在 EXPLAIN_STATEMENT 表的 SNAPSHOT 列中。其他应用程序不能使用此格式。

可从 Visual Explain 本身和如下位置获取有关解释快照信息内容的附加信息:

- 第182页的『数据对象的解释信息』
- 第183页的『数据运算符的解释信息』。

解释表信息

当请求解释表信息时，要记录附加信息，它们描述 SQL 优化器选择的存取方案。此信息存储在下列解释表中:

- EXPLAIN_ARGUMENT。此表表示每个个别运算符的唯一特征（如果存在的话）。
- EXPLAIN_INSTANCE。此表是所有解释信息的主控制表。解释表中的每一行数据都显式地链接至此表中唯一的一行。有关要解释的 SQL 语句的源和环境信息的基本资料保存在此表中。
- EXPLAIN_OBJECT。此表标识为满足 SQL 语句所生成的存取方案必需的那些数据对象。
- EXPLAIN_OPERATOR。此表包含 SQL 编译程序为满足 SQL 语句所需的所有运算符。
- EXPLAIN_PREDICATE。此表标识哪些谓词适用于特定的运算符。
- EXPLAIN_STATEMENT。此表包含对于不同级别的解释信息而存在的 SQL 语句的文本。用户输入的原始 SQL 语句与（由优化器）使用的版本一起存储在此表中以选择存取方案以满足该 SQL 语句。
- EXPLAIN_STREAM。此表表示在个别运算符和数据对象之间的输入和输出数据流。在 EXPLAIN_OBJECT 表中表示数据对象本身。在 EXPLAIN_OPERATOR 表中表示数据流中涉及的运算符。
- ADVISE_WORKLOAD。此表允许用户向数据库描述其工作负荷。工作负荷中的每一行代表一条 SQL 语句，并由一个相关的频率来描述。db2advis 工具和“索引”向导使用此表来挑选和存储工作及信息。
- ADVISE_INDEX。此表存储有关建议索引的信息。此表由 SQL 编译程序、db2advis 实用程序、“索引”向导或用户填充。可以两种方式使用此表:
  - 获取建议的索引。
  - 根据有关建议的索引的输入对索引求值。

在缺省情况下不会创建上面所有的表。可运行在 sqllib 子目录的 misc 子目录中找到的 EXPLAIN.DDL 脚本来创建它们。与需要“解释”和“建议”表的数据库连接。然后发出命令: db2 -tf EXPLAIN.DDL，这样就创建了表。如果有必要，这些表还可由“索引”向导自动创建。

Visual Explain 的每个矩形*对象*节点对应于 EXPLAIN_OBJECT 表中的一行。Visual Explain 的每个八角形“运算符”节点对应于 EXPLAIN_OPERATOR 表中的一行。在运算符之间或运算符的对象之间的每个链路对应于 EXPLAIN_STREAM 表的一行。

解释表信息在内容方面类似于为解释快照记录的信息，但是，此信息存储在使用标准 SQL 语句可存取的普通关系表中。

解释表象 Visual Explain 存取方案图一样，是为反映该存取方案内运算符和数据对象之间的关系而设计的。下图显示这些表之间的关系。

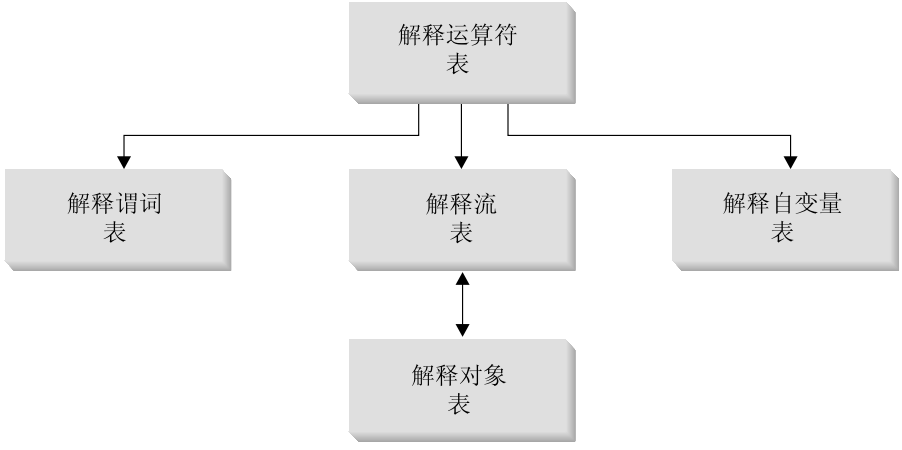


图 21. 解释表关系概述（未显示所有表）。

拥有多个用户公用的解释表是可能的。可为一个用户定义解释表。然后可对每个附加用户使用相同的名称来定义别名，以指向定义的表。共享公用解释表的每个用户必须对那些表有插入许可权。

有关解释表和如何创建这些表的详情，参见第475页的『附录C. SQL 解释工具』。有关解释表信息内容的附加信息可从下面获得:

- 第182页的『数据对象的解释信息』
- 第183页的『数据运算符的解释信息』。

在 `sqllib` 目录下的 `misc` 子目录中提供的 `db2exfmt` 工具可用于将解释表的内容格式化为一个易读的有序输出。

获取解释数据

在可以获取一条 SQL 语句的解释数据之前，必须定义一组解释表，所用模式与调用解释设施的授权 ID 所用的相同。有关如何创建这些表的信息，参见第462页的『解释表的表定义』。

捕捉解释表信息

一旦定义了这些表，则当编译 SQL 语句并已请求解释数据时，会捕捉解释数据:

- 对于**静态**或**增量联编** SQL 语句，当在 BIND 或 PREP 命令上指定 EXPLAIN ALL 或 EXPLAIN YES 选项时；或在源程序中使用静态 EXPLAIN SQL 语句时，将捕捉解释表信息。

  注: 当在运行期编译增量联编 SQL 语句时，在运行期而不是联编期将它们放入解释表。并且，用于对解释表的插入操作的解释表限定符和权限 ID 是程序包拥有者（而不是运行该程序包的用户）的限定符和权限 ID。

- 对于**动态** SQL 语句，将对下列任何一种情况捕捉解释表信息:
  - EXPLAIN SQL 语句。除非使用 FOR SNAPSHOT 子句，否则，捕捉所有解释信息并将这些信息放在解释表中。

    EXPLAIN SQL 语句的一个示例是:

    ```
    EXPLAIN PLAN FOR < 任何有效的 DELETE、INSERT、SELECT、SELECT INTO、
       UPDATE、VALUES 或 VALUES INTO SQL 语句 >
    ```

  - CURRENT EXPLAIN MODE 专用寄存器被设置为 YES。此设置导致 SQL 编译程序捕捉解释数据，并允许该 SQL 语句执行，然后返回该查询的结果。
  - CURRENT EXPLAIN MODE 专用寄存器被设置为 EXPLAIN。此设置导致 SQL 编译程序捕捉解释数据，但不执行该 SQL 语句。
  - CURRENT EXPLAIN MODE 专用寄存器被设置为 RECOMMEND INDEXES。此设置导致 SQL 编译程序捕捉解释数据，并将推荐的索引放进 ADVISE_INDEX 表中；但不执行该 SQL 语句。
  - CURRENT EXPLAIN MODE 专用寄存器被设置为 EVALUATE INDEXES。此设置导致 SQL 编译程序使用用户放进 ADVISE_INDEX 表中的索引。用户为应该求值的每个索引插入一个新行。每个索引的必需信息为: 索引名、表名和列名，它们组成正在求值的索引。在输入之后，应将专用寄存器 CURRENT EXPLAIN MODE 设置为 EVALUATE INDEXES。然后 SQL 编译程序扫描 ADVISE_INDEX 表，其中字段 USE_INDEX 设置为 『Y』（这称为虚拟索引）。解释所有以 EVALUATE INDEXES 方式执行的动态语句，就象这些虚拟索引是可用的。如果虚拟索引可提高语句性能的话，SQL 编译程序将选择使用虚拟索引。否则，忽略这些索引。通过复查 EXPLAIN 结果，可看出 SQL 编译程序是否使用了用户建议的索引。应考虑实现那些已使用的索引，以改进存取。
  - 已在 BIND 或 PREP 命令上指定了 EXPLAIN ALL 选项。此设置导致 SQL 编译程序在运行期捕捉动态 SQL 的解释数据，即使 CURRENT EXPLAIN MODE 专用寄存器的设置是 NO 亦如此。该 SQL 语句也将执行，并返回该查询的结果。

  注: 当编译 SQL 语句时只捕捉解释信息。在初始编译之后，当对环境的更改要求重新编译该语句时只重新编译动态 SQL 语句。如果对相同的 SQL 语句

连续发出了相同的 PREPARE 语句，则将只在第一次发出 PREPARE 语句时，编译 SQL 语句并捕捉解释数据，并假定环境不变。

有关使用 EXPLAIN SQL 语句或使用 CURRENT EXPLAIN MODE 寄存器的详情，参考 *SQL Reference* 手册。有关 BIND 和 PREP 命令的详情，参考 *Command Reference* 手册。

捕捉解释快照信息

当编译 SQL 语句并已请求解释数据时，会捕捉解释快照数据:

- 对于**静态**或**增量联编** SQL 语句，当在 BIND 或 PREP 命令上指定了 EXPLSNAP ALL 或 EXPLSNAP YES 子句时；或在源程序中使用包含 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的静态 EXPLAIN SQL 语句时，将捕捉解释快照。

  注: 当在运行期编译增量联编 SQL 语句时，在运行期而不是联编期将它们放入解释表。并且，用于对解释表的插入操作的解释表限定符和权限 ID 是程序包拥有者（而不是运行该程序包的用户）的限定符和权限 ID。

- 对于**动态** SQL 语句，将在下列任何一种情况下捕捉解释快照:
  - 使用 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的 EXPLAIN SQL 语句。除与解释快照相关的信息外， FOR SNAPSHOT 子句不捕捉其他解释表信息。除与解释快照相关的信息外，WITH SNAPSHOT 子句捕捉所有解释表信息。

    使用 EXPLAIN SQL 语句的一个解释快照的示例是:

    ```
    EXPLAIN PLAN FOR SNAPSHOT FOR < 任何有效的 DELETE、INSERT、SELECT、
       SELECT INTO、UPDATE、VALUES 或 VALUES INTO SQL 语句 >
    ```

    只捕捉解释快照，并将捕捉的信息放在 EXPLAIN_INSTANCE 和 EXPLAIN_STATEMENT 表中。
  - CURRENT EXPLAIN SNAPSHOT 专用寄存器被设置为 YES。 此设置导致 SQL 编译程序捕捉解释数据的快照，并允许该 SQL 语句执行，然后返回该查询的结果。
  - CURRENT EXPLAIN SNAPSHOT 专用寄存器被设置为 EXPLAIN。此设置导致 SQL 编译程序捕捉解释数据的快照，但不执行该 SQL 语句。
  - 已在 BIND 或 PREP 命令上指定了 EXPLAIN ALL 选项。此设置导致 SQL 编译程序在运行期捕捉解释数据的快照，即使 CURRENT EXPLAIN SNAPSHOT 专用寄存器的设置是 NO。该 SQL 语句也将执行，并返回该查询的结果。

注: 当编译 SQL 语句时只捕捉解释信息。在初始编译之后，当对环境的更改要求重新编译该语句时只重新编译动态 SQL 语句。如果对相同的 SQL 语句连续发出了相同的 PREPARE 语句，则将只在第一次发出 PREPARE 语句时，编译 SQL 语句并捕捉解释数据，并假定环境不变。

有关使用 EXPLAIN SQL 语句和 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的详情，或者有关使用 CURRENT EXPLAIN SNAPSHOT 寄存器的详情，参考 *SQL Reference* 手册。有关 BIND 和 PREP 命令的详情，参考 *Command Reference* 手册。

使用解释输出的准则

有许多方法用来分析解释数据，以帮助您调整查询和环境。例如:

- **正在使用索引吗?**

  正如在第82页的『索引对查询优化的影响』中所讨论的，适当的索引可大大改善性能。使用解释输出，您可以确定是否正在使用您创建的索引来帮助进行一组特定的查询。在解释输出中，应在下列几方面查找索引的使用:

  – 连接谓词
  – 本地谓词
  – GROUP BY 子句
  – ORDER BY 子句
  – 选择列表。

  也可使用解释设施来评估是否可使用另一个索引来代替现存的索引，或根本不使用索引。当创建一个新索引后，收集该索引的统计信息（使用 RUNSTATS 命令），并重新编译您的查询。经过一段时间，您可能通过解释数据会注意到现在正在使用表扫描，而不是索引扫描。这可能是表数据的群集发生改变所导致的。如果先前使用的索引现在有较低的群集比率，您可能要:

  – 重组您的表，以便根据该索引将数据群集
  – 使用 RUNSTATS 命令以更新该表和索引的目录统计信息
  – 重新编译查询
  – 重新检查解释输出，以确定重组表是否影响存取方案。

- **存取类型适合您的应用程序吗?**

  您可以分析解释输出，并查找存取数据的类型，通常，这种存取对于您正在运行的应用程序的类型而言不是最优的。例如:

  – **联机事务处理 (OLTP) 查询**

OLTP 应用程序是用范围定界的谓词来使用索引扫描的主要候选者，因为它们倾向于对关键字列使用等式谓词以只返回少数合格的行。如果您的 OLTP 查询使用表扫描，应分析解释数据以确定不使用索引扫描的原因。

– **“仅浏览”查询**

“浏览”类型查询的搜索标准可能很含糊，这样会产生大量合格的行。如果用户通常只查看少数输出数据屏幕，您可能想尝试确保在返回一些结果前，不需要计算整个回答集。对于这种情况，用户的目标与优化器的基本操作原则不同，优化器试图将整个查询的资源消耗减小到最低程度，而不只是前几个数据屏幕。

例如，如果解释输出表明在存取方案中使用了合并扫描连接和排序运算符，则在将任何行返回至应用程序之前将在一个临时表中实现整个回答集。在这种情况下，您可以尝试在 SELECT 语句上使用 OPTIMIZE FOR 子句来更改存取方案。（有关 OPTIMIZE FOR 子句的详情，参见 第64页的『OPTIMIZE FOR n ROWS 子句』 手册。）使用这种方式，优化器可尝试选择一个存取方案，该方案在将前面几行返回至应用程序前，不在临时表中产生整个回答集。

- **正在使用什么类型的连接方法?**

  如果一个查询连接了两个表，您可检查所用的连接处理的类型。涉及多个行的连接，如在决策支持查询中的那些连接，使用合并连接通常会运行得更快。只涉及少数行的连接，如 OLTP 查询，通常使用嵌套的循环连接会运行得更快。但是，在任何一种情况下都可能存在运行减慢的情况，如使用本地谓词或索引，将更改这些典型连接工作的方式。（有关这两个连接方法的工作原理，参见第145页的『嵌套循环连接』和第146页的『合并连接』。）

Visual Explain

与其他方法比较时，尤其是包含更复杂的操作序列的那些方法，可使用 Visual Explain 更详细地研究查询。Visual Explain 不是在所有支持的平台上都可用。您应检查适合您平台的*快速入门*手册，以查看是否支持 Visual Explain。

Visual Explain 允许将解释的 SQL 语句的存取方案看作一个图。您可以使用可从图中获得的信息来调整 SQL 查询，以获取更好的性能。Visual Explain 还允许您动态解释 SQL 语句，并查看生成的存取方案图。

优化器选择一个存取方案，而 Visual Explain 将该信息显示为一个存取方案图，在该图中表和索引以及对它们执行的操作表示为节点，数据流由这些节点之间的链路表示。

要显示存取方案图，必须事先创建解释快照。从存取方案图，可以查看下列各项的详情:

- 表和索引（及其相关的列）
- 运算符（如表扫描、排序以及连接）
- 表空间和函数。

也可使用 Visual Explain 来:

- 查看优化时使用的统计信息。然后可以将这些统计信息与当前目录统计信息比较，以帮助您确定重新联编程序包是否可以改进性能。
- 确定是否使用了索引来存取表。如果未使用索引，Visual Explain 可帮助您确定对哪些列建立索引可能受益。
- 通过比较一个查询的存取方案图调整之前和之后的版本，查看执行各种调整技术的效果。
- 获取存取方案中关于每个操作的信息，包括总估计成本和检索行数（基数）。

有关 Visual Explain 的其他详情，您应参考通过控制中心可获得的联机信息。在命令行上输入 `db2cc` 可存取控制中心。

SQL 建议设施

“索引顾问”是一个管理工具，它降低了对数据设计和定义适当的索引的需要。

“索引顾问”擅长于:

- 查找问题查询的最佳索引。
- 为查询集（工作负荷）查找最佳索引，但受可选择应用的资源限制的支配。
- 测试工作负荷的索引而不必创建索引。

以下是与“SQL 建议设施”相关的概念。首先是*工作负荷*。工作负荷是数据库管理程序必须在给定时间内处理的一组 SQL 语句。这些 SQL 语句可包括: SELECT、INSERT、UPDATE 和 DELETE 语句。例如，在一个月的时间内，数据库管理程序可能必须处理 1 000 个 INSERT、10 000 个 UPDATE、10 000 个 SELECT 和 1 000 个 DELETE。工作负荷中的信息与给定时间内 SQL 语句的类型和频率有关。建议引擎使用此工作负荷信息和数据库信息来推荐索引。建议引擎的目标是将总工作负荷成本减至最低。

其次是*虚拟索引*的概念。虚拟索引是当前数据库模式中不存在的索引。这些索引可能是“建议设施”对您提出的建议，或者是希望“建议设施”求值的索引。这些索引也可能是“建议设施”视为该过程的一部分，后来由于不推荐它们而废弃的那些索引。可使用 ADVISE_INDEX 表将虚拟索引在您和“建议设施”之间来回传送。

“建议设施”使用数据库的工作负荷和统计信息来生成建议的索引。

“建议设施”使用两个 EXPLAIN 表:

- ADVISE_WORKLOAD

  此表用来描述要考虑的工作负荷。表中的每一行代表一条 SQL 语句，并由相关的频率描述。每个工作负荷都有一个标识符，它是表中称为『WORKLOAD_NAME』的字段。分担同一工作负荷的所有 SQL 语句应具有相同的 WORKLOAD_NAME。

  索引向导和 db2advis 工具使用该表来挑选和存储工作负荷信息。

- ADVISE_INDEX

  此表存储有关建议索引的信息。将来自 SQL 编译程序、“索引”向导、db2advis 工具或您的信息放入此表。

  可以两种方式使用此表:

  – 从“建议设施”获取建议索引

  – 对索引求值。

注: 要创建这些表，可运行在 sqllib 子目录的 misc 子目录中的 EXPLAIN.DDL 脚本。如果尚未创建这些表，也可用“索引”向导创建。

使用“索引顾问”的过程包括输入、调用顾问、输出和某些应考虑的特殊情况。

为“索引顾问”创建输入有三种方法:

- 捕捉工作负荷。

  即，使用下列其中一种方法来创建将求值的 SQL:

  – 使用监控程序获取动态 SQL。

  – 使用 SYSSTMT 目录视图获取静态 SQL。

  – 通过剪切那些值并将其粘贴进 ADVISE_INDEX 表来添加语句和频率。

- 修改工作负荷频率以增加或减小查询的重要性。
- 确定对数据的约束（如果有的话）。

调用“索引顾问”有四种方法:

- 使用控制中心。

  这是使用“索引顾问”的建议方法。从控制中心，展开对象树，直到找到**索引**文件夹为止。用鼠标右按钮单击**索引**文件夹并从弹出菜单中选择**创建 -> 使用向导创建索引**。“索引”向导打开。“索引”向导有详尽的帮助，使用它很容易。此向导还包含一些功能部件，它们通过查找最近执行的 SQL、查找最近使用的程序包或人工添加 SQL 语句来构建工作负荷。

- 使用命令行处理器。

  在命令行中输入 db2advis。从以下三个位置之一读入工作负荷来启动 db2advis:

  – 从命令行
  – 从文本文件中的语句
  – 从 ADVISE_WORKLOAD 表（当您插入具有建议的工作负荷（SQL 和频率）的行之后）。

  然后该工具使用 CURRENT EXPLAIN MODE 寄存器来获得建议的索引，并与内部优化算法组合在一起以选出最佳索引。输出将按需要传送到终端屏幕、ADVISE_INDEX 表和输出文件。

  例如，您可能希望该工具对一个简单的查询 『select count(*) from sales where region = 'Quebec'』 推荐索引。

```
$ db2advis -d sample \
 -s "select count(*) from sales where region = 'Quebec'" \
 -t 1
performing auto-bind

Bind is successful. Used bindfile: /home3/valentin/sqllib/bnd/db2advis.bnd

Calculating initial cost (without recommended indexes) [31.198040] timerons
Initial set of proposed indexes is ready.
Found maximum set of [1] recommended indexes
Cost of workload with all indexes included [2.177133] timerons
cost without index [0] is [31.198040] timerons.  Derived benefit is
[29.020907]
total disk space needed for initial set [1] MB
total disk space constrained to         [-1] MB
  1  indexes in current solution
 [31.198040] timerons  (without indexes)
 [2.177133] timerons  (with current solution)
 [%93.02] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ689 ON VALENTIN.SALES (REGION DESC)
===========================
"索引顾问"工具完成。
```

  db2advis 工具也可用于对工作负荷建议索引。可创建称为 『sample.sql』 的输入文件:

```
--#SET FREQUENCY 100
select count(*) from sales where region = ?;
--#SET FREQUENCY 3
select projno, sum(comm) tot_comm  from employee, emp_act
```

```
where employee.empno = emp_act.empno and
     employee.job='DESIGNER'
group by projno
order by tot_comm desc;
--#SET FREQUENCY 50
select * from sales where sales_date = ?;
```

然后执行以下命令:

```
$ db2advis -d sample -i sample.sql -t 0
  found [3] SQL statements from the input file

Calculating initial cost (without recommmended indexes) [62.331280] timerons
Initial set of proposed indexes is ready.
Found maximum set of [2] recommended indexes
Cost of workload with all indexes included [29.795755] timerons
cost without index [0] is [58.816662] timerons.  Derived benefit is
[29.020907]
cost without index [1] is [33.310373] timerons.  Derived benefit is
[3.514618]
total disk space needed for initial set [2] MB
total disk space constrained to         [-1] MB
  2  indexes in current solution
 [62.331280] timerons  (without indexes)
 [29.795755] timerons  (with current solution)
 [%52.20] improvement

Trying variations of the solution set.
Time elapsed.
LIST OF RECOMMENDED INDEXES
===========================
index[1], 1MB CREATE INDEX WIZ119 ON VALENTIN.SALES (SALES_DATE DESC,
SALES_PERSON DESC)
index[2], 1MB CREATE INDEX WIZ63 ON VALENTIN.SALES (REGION DESC)
===========================
 “索引顾问”工具完成。
```

- 使用自定向方法，包括 EXPLAIN 方式和 PREP 选项。

  例如，将 CURRENT EXPLAIN MODE 专用寄存器设置为 RECOMMEND INDEXES。此设置将导致 SQL 编译程序捕捉解释数据，并将推荐的索引放进 ADVISE_INDEX 表中；但不执行该 SQL 语句。

  或者将 CURRENT EXPLAIN MODE 专用寄存器设置为 EVALUATE INDEXES。此设置将导致 SQL 编译程序使用用户放进 ADVISE_INDEX 表中的索引。用户为应该求值的每个索引插入一个新行。每个索引的必需信息为: 索引名、表名和列名，它们组成正在求值的索引。在输入之后，应将专用寄存器 CURRENT EXPLAIN MODE 设置为 EVALUATE INDEXES。然后，SQL 编译程序扫描 ADVISE_INDEX 表，查找字段 USE_INDEX=『Y』 的索引（这些索引称为虚拟索引）。解释所有以 EVALUATE INDEXES 方式执行的动态语句，就象这些虚拟索引是可用的。如果虚拟索引可提高语句性能的话，SQL 编译

程序将选择使用虚拟索引。否则，忽略这些索引。通过复查 EXPLAIN 结果，可看出 SQL 编译程序是否使用了用户建议的索引。应考虑实现那些已使用的索引，以改进存取。

- 使用“调用层接口”(CLI)。

  如果要使用此接口来编写应用程序，也可使用顾问。

有不同的方法来使用顾问产生的结果:

- 解释“索引顾问”的输出。

  要了解“建议设施”建议了什么索引，可使用以下查询:

  ```
  SELECT CAST(CREATION_TEXT as CHAR(200))
     FROM ADVISE_INDEX
  ```

- 应用“索引顾问”的建议。
- 了解何时卸下索引。

要获得特定查询的更好建议，建议该查询参照自身。可使用“索引”向导构建只包含单个查询的工作负荷，来对单个查询建议索引。

可从“事件监控程序”的输出收集样本工作负荷。“事件监控程序”可用于收集动态 SQL 语句的执行结果。然后可将这些语句反馈回“建议设施”。

“索引”向导是一个简单、直观、易用、形象化的界面，它提供了一种很好的方法来存取“建议设施”。

第3部分 调整和配置系统

第8章 操作性能

以下主题提供在运行期期间如何影响 SQL 查询性能的信息:

- DB2 如何使用内存
- 管理数据库缓冲池
- 管理多数据库缓冲池
- 将数据预读取至缓冲池
- 配置 I/O 服务器启用预读取和并行 I/O
- 排序
- 重组目录和用户表
- DMS 设备的性能考虑事项
- 管理初始化额外开销
- 数据库代理程序
- 使用数据库系统监控程序
- 扩充内存。

下面章节也提供如何影响性能的信息:

- 第37页的『第3章 应用程序考虑事项』
- 第77页的『第4章 环境考虑事项』
- 第93页的『第5章 系统目录统计信息』。

也可参考*管理指南: 计划*中的物理数据库设计考虑事项。

DB2 如何使用内存

在 DB2 中许多可用的配置参数影响系统中内存的使用。一些参数可能影响服务器上的内存，而另一些可能影响客户机上的内存，还有一些会影响上述两者。而且，内存是在不同的时间从系统中的不同区域分配和释放的。

系统管理员也应当考虑如何平衡系统中整体内存的使用。在操作系统上运行的不同应用程序可能以不同方式使用内存。例如，一些应用程序可能使用文件系统高速缓存，而数据库管理程序用自己的缓冲池来代替操作系统设施进行数据高速缓存。有关其他考虑事项，参见第207页的『设置影响内存使用的参数』。

第202页的图22显示数据库管理程序使用不同类型的内存。有一个假定，即此图说明的既不是“扩充企业版”环境也不是多逻辑节点环境中的内存使用情况。在

“扩充企业版”或多逻辑节点环境中，有多个“数据库管理程序共享内存”集（每个节点一个）。

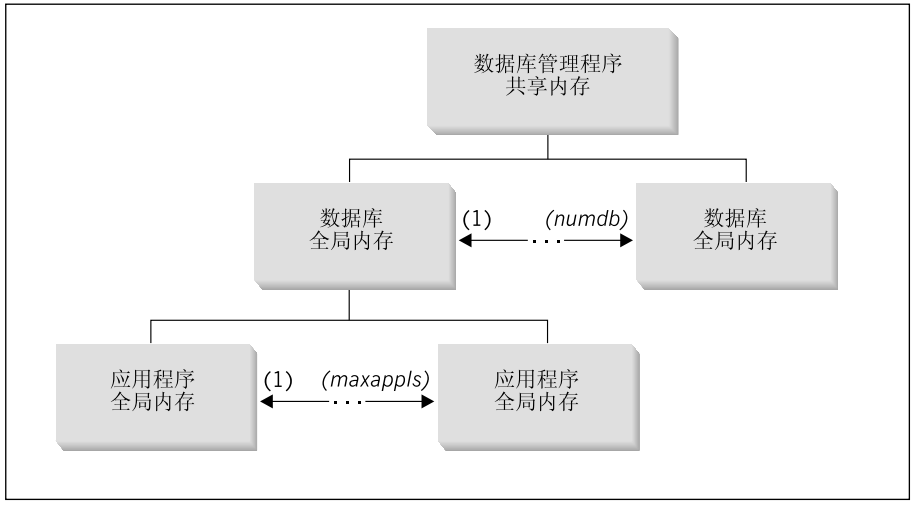


图 22. 数据库管理程序所用的内存类型

在以下时间，为数据库管理程序的每个实例分配内存:

- 当启动数据库管理程序 (db2start) 时，分配标记为“数据库管理程序共享内存”的区域，并且此区域保持已分配状态，直到停止数据库管理程序 (db2stop)。此区域包含数据库管理程序为管理所有数据库连接上的活动所需的信息。当第一个应用程序与一个数据库连接时，要分配全局的和专用的内存区。
- 当第一次激活或连接数据库时，会分配“数据库全局内存”。数据库全局内存在与该数据库连接的所有应用程序中使用，它包含诸如缓冲池、锁定列表、数据库堆和实用程序堆等内存区。
- 当应用程序与数据库连接时，会分配“应用程序全局内存”（仅在分区数据库环境中，或者仅当启用 *intra_parallel* 配置参数时，才会发生此情况）。此内存由为该应用程序工作的代理程序使用，以便在它们之间共享数据和协调活动。
- （前一框图中未显示: ）当指定一个代理程序来为特定的应用程序工作时（作为连接请求的结果，或并行环境中的新 SQL 请求的结果），就会为该代理程序分配“代理程序专用内存”。代理程序专用内存区是为代理程序分配的，它包含仅供此特定代理程序使用的内存分配，如排序堆和应用程序堆。

  一旦一个数据库已由某个应用程序使用，任何随后连接的应用程序将只拥有以它们的名义分配的代理程序专用内存和应用程序全局共享内存。

第202页的图22显示配置参数设置可以如何影响内存。特别是，以下列表中的参数可以限制为特定目的分配的内存容量。（在分区数据库环境中的每个数据库分区上，此内存都是必需的。）

- *numdb* 定义（由不同应用程序使用的）并行活动数据库的最大数目。因为每个数据库有自己的全局内存区，所以如果此参数的值增加，则有可能分配的内存容量也会增加。
- *maxappls* 定义可同时与单个数据库连接的应用程序的最大数目。它影响可能为该数据库分配的“代理程序专用内存”和“应用程序全局内存”的内存容量。（注意对每个数据库，可用不同方式设置此参数。）
- （前一框图中未显示: ）*maxagents*（和用于并行环境的 *max_coordagents*）对可在实例内所有活动数据库中同时存在的数据库管理程序代理程序的数目进行限制。与 *maxappls* 一起，这两个参数限制为“代理程序专用内存”和“应用程序全局内存”分配的内存容量。（有关代理程序的信息，参见第227页的『数据库代理程序』。）

第204页的图23总结要使用多大内存来支持应用程序。下列配置参数允许您通过限制“内存段”（逻辑内存的一部分）的数目和它们的大小，来控制此内存的大小。

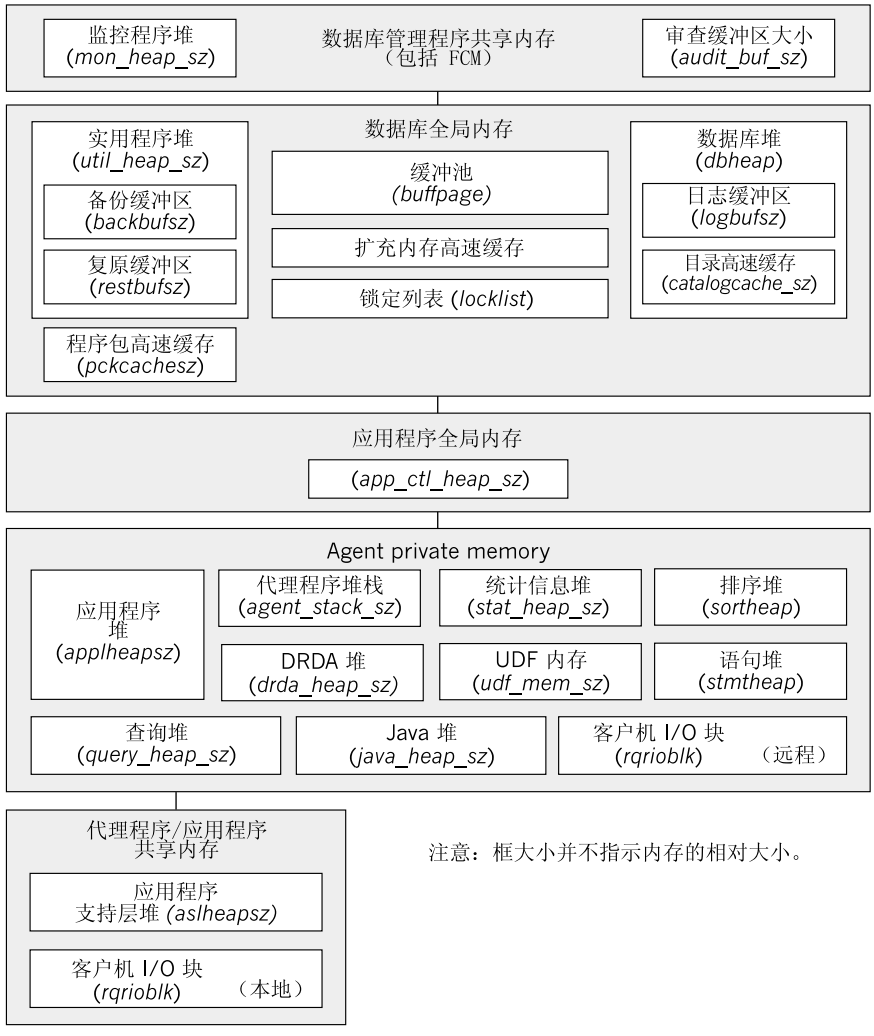


图 23. 数据库管理程序如何使用内存

数据库管理程序共享内存

内存空间是数据库管理程序运行所必需的。此空间可以很大，尤其是在分区内和分区间并行环境中。通过复查下列几节，您可以预测和控制此空间的大小:

- 第227页的『数据库代理程序』. 代表应用程序运行的代理程序需要足够多的内存空间，尤其在 *maxagents* 的值不合适时。
- 第208页的『FCM 需求』。对于分区数据库系统，快速通信管理器 (FCM) 需要足够多的内存空间，尤其在 *fcm_num_buffers* 的值不合适时。

FCM 内存是从 FCM 缓冲池分配的，或是从数据库管理程序共享内存和 FCM 缓冲池分配的，这取决于分区数据库系统是否使用多个逻辑节点。有关详情，参见 FCM 缓冲池的以下说明。

FCM 缓冲池

如果您的分区数据库系统不具有多个逻辑节点，则数据库管理程序共享内存和 FCM 缓冲池如图24中所示。

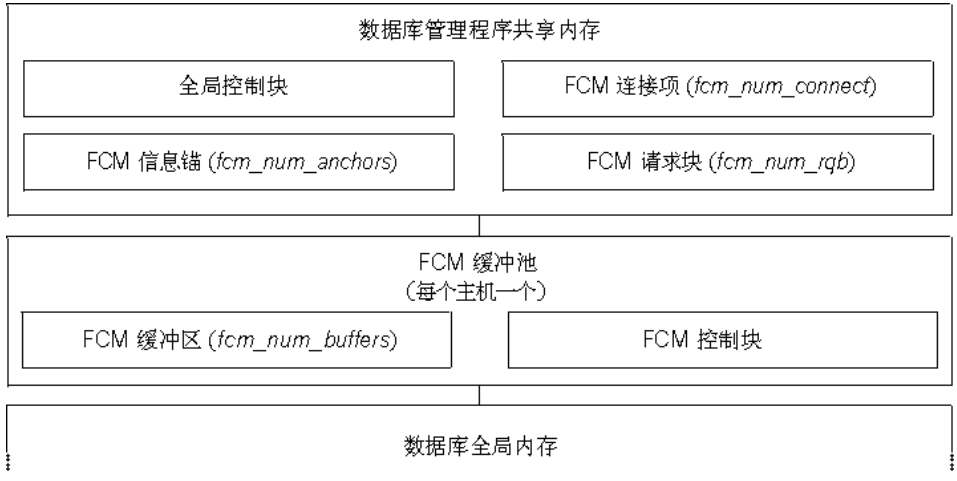


图 24. 不使用多个逻辑节点时的 FCM 缓冲池

如果您的分区数据库系统使用多个逻辑节点，则数据库管理程序共享内存和 FCM 缓冲池如第206页的图25中所示。

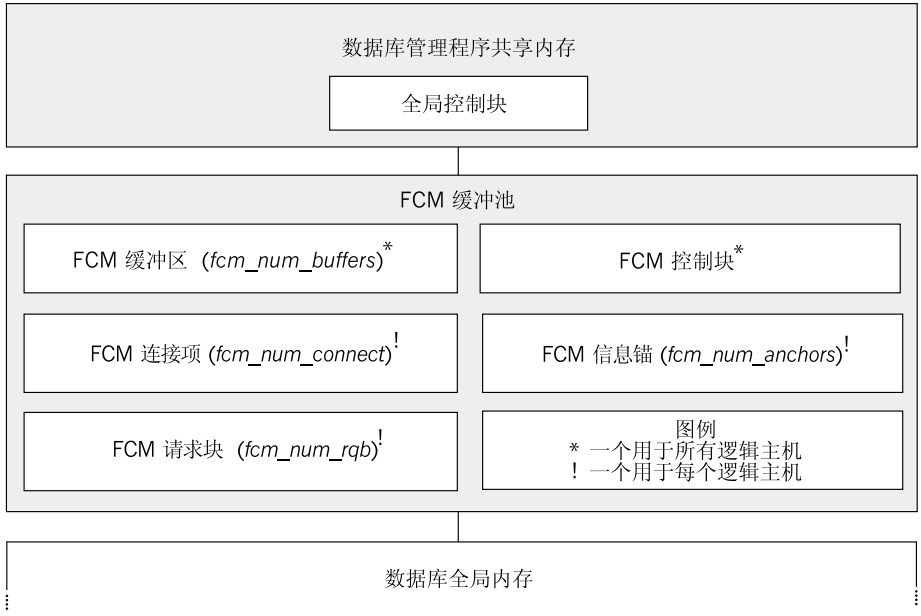


图 25. 使用多个逻辑节点时的 FCM 缓冲池

数据库全局内存

"数据库全局内存" 受下列配置参数影响:

- 内存段数受 *numdb* 限制（参见第401页的『并行活动数据库的最大数目 (numdb)』）。
- 内存段的最大大小由下列参数的值确定:
  - 第290页的『缓冲池大小 (buffpage)』（如果缓冲池大小是 -1），或当创建或改变缓冲池时指定的显式大小
  - 第297页的『锁定列表的最大存储器 (locklist)』
  - 第292页的『数据库堆 (dbheap)』
  - 第295页的『实用程序堆大小 (util_heap_sz)』
  - 第329页的『扩充内存段大小 (estore_seg_sz)』
  - 第330页的『扩充内存段数 (num_estore_segs)』。
  - 第299页的『程序包高速缓存大小 (pckcachesz)』。

应用程序全局内存

"应用程序全局内存" 受下列配置参数影响:

- 第301页的『应用程序控制堆大小 (app_ctl_heap_sz)』。

对于并行系统，应用程序控制堆也需要空间，这种堆为在一个数据库分区中代表同一个应用程序工作的代理程序所共享。当从应用程序收到请求的

第一个代理程序请求连接时，分配这个堆。该代理程序可以是协调代理程序，也可以是子代理程序（参见第227页的『数据库代理程序』）。

代理程序专用内存

- 内存段的数目受下列值中的较小者限制:
  - 所有活动数据库的 *maxappls* 的总数（参见第331页的『活动应用程序的最大数目 (maxappls)』）
  - *maxagents*（参见第336页的『代理程序的最大数目 (maxagents)』）。
- 内存段的最大大小由下列参数的值确定:
  - 第305页的『应用程序堆大小 (applheapsz)』
  - 第303页的『排序堆大小 (sortheap)』
  - 第305页的『语句堆大小 (stmtheap)』
  - 第306页的『统计堆大小 (stat_heap_sz)』
  - 第307页的『查询堆大小 (query_heap_sz)』
  - 第308页的『DRDA 堆大小 (drda_heap_sz)』
  - 第308页的『UDF 共享内存集大小 (udf_mem_sz)』
  - 第309页的『代理程序栈大小 (agent_stack_sz)』。

代理程序 / 应用程序共享内存

- 代理程序 / 应用程序共享内存段（用于本地客户机）的总数受下列值中的较低者限制:
  - 所有活动数据库的 *maxappls* 的总数（参见第331页的『活动应用程序的最大数目 (maxappls)』）
  - *#maxagents*（参见第336页的『代理程序的最大数目 (maxagents)』）或（对于并行系统） *max_coordagents* 的值（参见第338页的『最大协调代理程序数 (max_coordagents)』）。
- “代理程序 / 应用程序共享内存”还受下列配置参数影响:
  - 第313页的『应用程序支持层堆大小 (aslheapsz)』。
  - 第314页的『客户机 I/O 块大小 (rqrioblk)』。

设置影响内存使用的参数

除非已仔细调整好分配内存的参数值，否则，即使在安装了最大内存量的系统上，也*永远不*应将该参数设为最高值。许多参数可使数据库管理程序快捷地占用一台机器上所有可用的内存。此外，对大量内存的管理可能给数据库管理程序带来大量的额外工作，因而产生更多的额外开销。

某些基于 UNIX 的操作系统会在一个进程分配内存时（而不是在它调出页至交换空间时）分配交换空间。在这种情况下，应当确保总的共享内存大小受相同大小的分页空间支持。

对于大多数配置参数，内存是在需要时唯一要落实的。这些参数反映了特定内存堆的最大大小。此规则的显著例外是下列参数，它们的内存是根据参数值完全落实的:

- 第290页的『缓冲池大小 (buffpage)』（如果缓冲池大小是 -1），或当创建或改变缓冲池时指定的显式大小
- 第303页的『排序堆阈值 (sheapthres)』
- 第297页的『锁定列表的最大存储器 (locklist)』
- 第313页的『应用程序支持层堆大小 (aslheapsz)』
- 第389页的『FCM 信息锚数 (fcm_num_anchors)』
- 第390页的『FCM 缓冲区数 (fcm_num_buffers)』
- 第391页的『FCM 连接项数 (fcm_num_connect)』
- 第391页的『FCM 请求块数 (fcm_num_rqb)』。

可以通过基准测试来最准确地确定这几类参数的适当值: 对服务器运行典型的和最坏情况下的 SQL 语句，同时修改这些参数的值，直至找到性能开始下降的那一点。如果将性能与参数值的对比用图形表示，则曲线开始上升或下降的那一点将指示在此点上附加分配并不再增加应用程序性能的值，因此只是浪费内存而已。（参见第265页的『第12章 基准测试』。）

几个参数的内存分配的上限也许超过了现有的硬件和操作系统的内存能力。选择这些限制，以允许将来的增长。

有关有效参数的范围，参见第277页的『第13章 配置 DB2』中的参数说明。

FCM 需求

当配置下列“快速通信管理器”(FCM) 配置参数时，从缺省值开始:

- 第390页的『FCM 缓冲区数 (fcm_num_buffers)』
- 第391页的『FCM 请求块数 (fcm_num_rqb)』
- 第391页的『FCM 连接项数 (fcm_num_connect)』
- 第389页的『FCM 信息锚数 (fcm_num_anchors)』。

要调整这些参数，使用数据库系统监控程序来监控空闲缓冲区、空闲信息锚、空闲连接项和空闲请求块的低水位标志。如果低水位标志小于对应空闲数据项目数的百分之十，则增加对应参数的值。有关数据库系统监控程序的信息，参见第232页的『使用数据库系统监控程序』。

有关启用 FCM 通信的信息，*参考管理指南: 计划*。

管理数据库缓冲池

缓冲池是存储器的一个区域，在该区域中临时读入和更改数据库页（包含表行或索引项）。缓冲池的目的是改善数据库系统性能。从内存存取数据可以比从磁盘存取数据快得多。因此，数据库管理程序需要读写磁盘的次数越少，性能就越好。

一个或多个缓冲池的配置是最重要的调整区域，因为正是在这里与数据库连接的应用程序进行了大多数数据处理（不包括大对象和长整数字段数据）。

当一个应用程序第一次存取某个表的某行时，数据库管理程序将包含该行的页放入缓冲池内。下一次任何应用程序请求数据时，则先检查该缓冲池。如果发现请求的数据位于缓冲池保存的页中，则数据库管理程序就不需要到磁盘存储器检索所请求的数据。避免从磁盘存储器检索数据，可获得更快的性能。

当激活一个数据库时或第一个应用程序与该数据库连接时，就会分配与缓冲池相关的存储器。应用程序是缓冲池的主要受益者；一旦断开所有应用程序，将释放与该缓冲池相关的存储器。

这些页一直保存在缓冲池中，直到数据库关闭，或直到其它页需要使用某一页所占用的空间。使用如下标准来选择为引入另一页而在缓冲池中选择的空间:

- 对某一页的上次引用
- 查看过该页的上一个代理程序再次引用该页的可能性
- 页的类型
- 某一页是否只在内存中更改而不写到磁盘中去。（更改的页通常在被覆盖之前写入到磁盘中。）

注: 在将更改的页写入磁盘后，除非其他页需要它们所占用的空间，否则不从缓冲池中除去它们。如果需要这些页中的数据，则在它们被覆盖之前可以再次存取它们。

创建缓冲池时，缺省情况下页大小为 4 KB。创建缓冲池时可以选择将页大小设置为 4 KB、8 KB、16 KB 或 32 KB。如果缓冲池是使用某一种页大小创建的，则只有使用相同页大小创建的表空间可以与它们相关。不能在创建缓冲池之后改变它的页大小。

缓冲池中的页可以有不同的属性:

- 当前正在读取或更新使用中的页。其他代理程序可以读取而不能更新它们。
- “脏”页是数据已被更改而尚未写入磁盘的页。在一个页写入磁盘后，可认为它是“干净的”并且它仍然留在缓冲池中。被干净的页占用的空间可给新页使用，并可用于至一个相关扩充高速缓存（如果定义了）的迁移。

当缓冲池中更改的页所占空间的百分比已超过由 *chngpgs_thresh* 配置参数指定的值时，可以将这些页从缓冲池写至磁盘。也许您还需要配置此数据库，以包括多个页清除器的代理程序。这些代理程序将更改的页写至磁盘，以便数据库代理程序可在缓冲池中找到可用的空间。

页清除器的代理程序执行原来必须由数据库代理程序执行的 I/O。这样，您的应用程序可以运行得更快，因为当事务的数据库代理程序将页写至磁盘时，不会强制事务等待。（页清除器的代理程序有时称为*异步页清除器*或*异步缓冲区写程序*，因为它们可以与数据库代理程序一起并行运行。）

要更改页清除器的代理程序数目，使用 *num_iocleaners* 配置参数（缺省情况是创建一个页清除器的代理程序）。有关详情，参见第325页的『异步页清除器数 (num_iocleaners)』。将此参数的值设置为介于 1 与数据库中的物理磁盘数之间。此数越大，执行更新量很大的工作负荷时的性能也就越好。当有大量涉及异步数据页或索引页写操作的数据页或索引页写操作时，情况亦如此。

将页写至磁盘也允许在系统崩溃时更快地恢复数据库，因为数据库管理程序可以从磁盘重建更多的缓冲池，而不必使用数据库日志文件。因此，如果在恢复期间需要读取的日志的大小超过如下最大值，则要请求执行页清除:

```
logfilsiz * softmax
```

其中:

- *logfilsiz* 表示日志文件的大小（参见第345页的『日志文件的大小 (logfilsiz)』）
- *softmax* 表示在数据库崩溃后要恢复的日志文件的百分比（参见第351页的『恢复范围和软检查点间隔 (softmax)』）。

  例如，如果 softmax 的值为 250，则如果发生崩溃，2.5 个日志文件将包含需要恢复的更改。

可使用数据库系统监控程序来帮助您跟踪在恢复期间，请求执行页清除以将日志读取时间减至最小的次数。有关详情，参见 *System Monitor Guide and Reference* 手册中的 *pool_lsn_gap_clns（触发的缓冲池日志空间清除器）*监控程序元素说明。

恢复期间需要读取的日志的大小是日志中下列项的位置之间的差值:

- 最近写入的日志记录

- 描述对缓冲池中的数据所做的最早更改的日志记录。

下图举例说明了与执行所有 I/O 的数据库代理程序比较，页清除器代理程序和数据库代理程序如何共享管理缓冲池的工作。

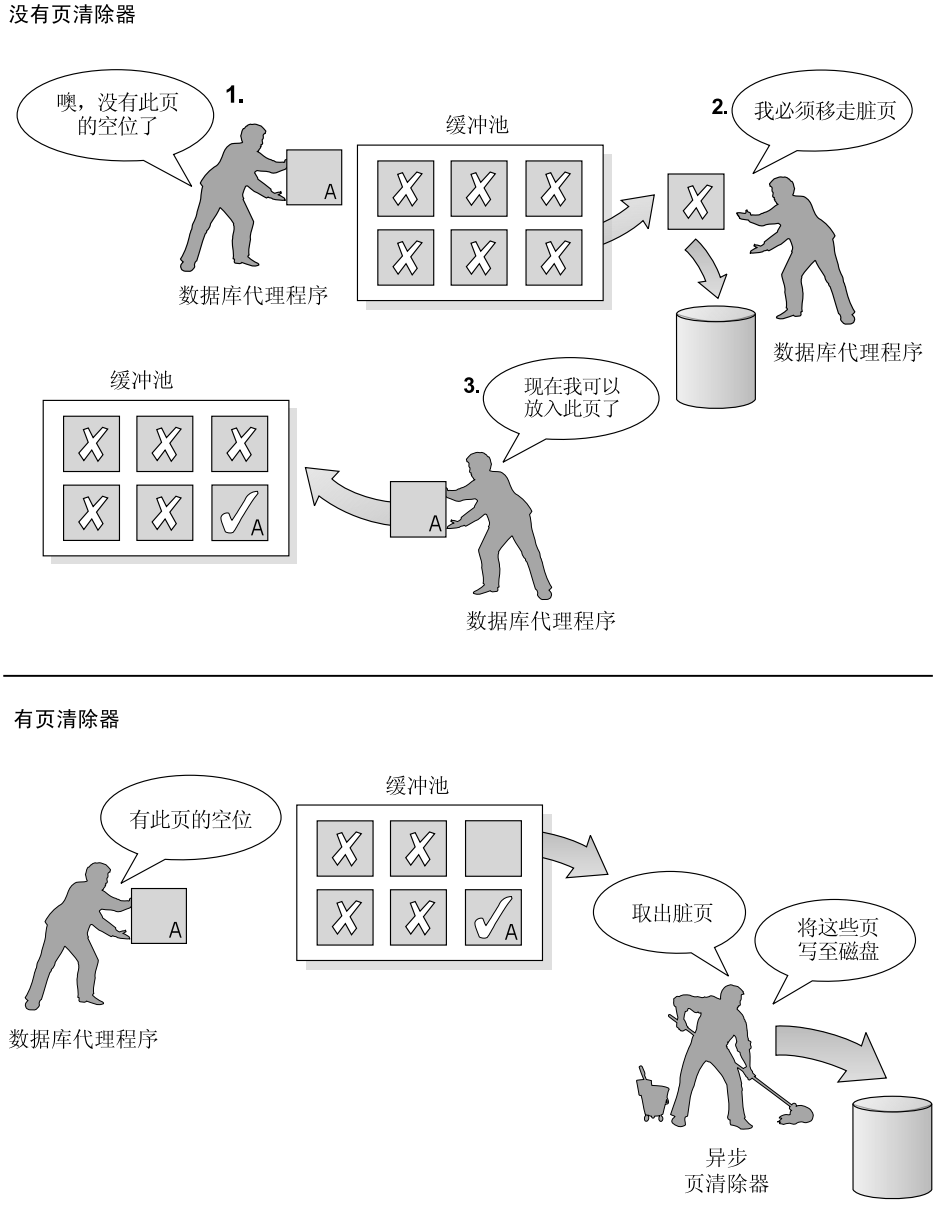


图 26. 异步页清除器. 把“脏”页写出至磁盘。

管理多数据库缓冲池

每个数据库至少需要一个缓冲池。然而，根据您的需要，可以为单个数据库创建几个缓冲池，且每个缓冲池有不同的大小。CREATE、ALTER 和 DROP BUFFERPOOL 语句允许您创建、更改或除去缓冲池。您可使用 CREATE TABLESPACE 和 ALTER TABLESPACE 语句来指定将哪些数据高速缓存到一个缓冲池中。

如果在 SYSIBM.SYSBUFFERPOOLS 目录视图中指定缓冲池的大小为 -1，则可用 *buffpage* 配置参数指定任何缓冲池的大小。（否则，忽略此参数。）可用 DDL 语句 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 来设置缓冲池的大小。

新数据库具有缺省缓冲池 IBMDEFAULTBP，它的大小是由平台确定的。创建或迁移一个数据库后，就可为它创建其他缓冲池。

当进行数据库设计时，您可能已确定具有 8 KB 页大小的表是最佳选择。因此，您应该创建含有 8 KB 页大小的缓冲池（以及一个或多个含有相同页大小的表空间）。

在一个分区数据库环境中，一个数据库的每个缓冲池在所有数据库分区上有相同的缺省定义（除非在 CREATE BUFFERPOOL 语句中另外指定了它，或使用 ALTER BUFFERPOOL 语句为特定数据库分区更改了该缓冲池的大小）。

当创建页大小为 4KB 的表空间且不给它分配特定缓冲池时，将给该表空间分配缺省缓冲池。如果创建页大小大于 4 KB（8 KB、16 KB、32 KB）的表空间，应该给它分配使用相同页大小的缓冲池。如果此缓冲池当前不是活动的，DB2 将尝试把该表空间分配给使用相同页大小的活动缓冲池（如果存在的话）。此分配（如果完成）是临时性的。当再次激活该数据库且原来指定的缓冲池活动时，DB2 会将该表空间分配给该缓冲池。

不能使用 ALTER TABLESPACE 语句将该表空间添加至使用另一种页大小的缓冲池。

创建或改变缓冲池时，所有缓冲池必需的全部内存必须可供数据库管理程序使用，以便在启动该数据库时可分配所有缓冲池。如果启动数据库时此内存不可用，数据库管理程序将试图启动缺省缓冲池 (IBMDEFAULTBP) 以及用另一种页大小定义的一个缓冲池，但是只能尝试启动最小大小 16 页的缓冲池。此最小缓冲池的大小可用注册表变量 *DB2_OVERRIDE_BPF* 替换有关这个和其他注册表变量和环境变量的详情，参见第417页的『附录A. DB2 注册表变量和环境变量』。每次启动缓冲池的尝试失败时都返回一个警告信息；数据库会继续保持在此操作状态，直到其配置被更改且数据库可以完全重新启动为止。

允许数据库管理程序用最小大小的值启动的原因是允许您连接该数据库。然后您可立即重新配置缓冲池大小；或者执行其他关键任务。不要考虑在这种状态下长时间运行数据库。

注： 虽然可以更改与缺省缓冲池相关的大小和属性，但是不能卸下它。另外，对于每个缓冲池，都存在一个基于所用平台的最小大小。

给缓冲池分配大量内存是有好处的。例如，更大的缓冲池大小:

- 可以将经常请求的数据页保存在缓冲池中，这样可更快地存取。更少的 I/O 操作可以减少 I/O 争用，从而可提供更短的响应时间并减少 I/O 操作所需的处理器资源。
- 提供在相同的响应时间内到达更高的事务处理速率的机会。
- 防止频繁使用的磁盘存储设备如目录表和频繁引用的用户表和索引的 I/O 争用。由于包含临时表空间的磁盘存储设备上的 I/O 争用减少，查询所需的排序也从中受益。

选择一个或多个缓冲池

如果下列其中任何一个条件适用于您的系统，应当只使用单个缓冲池:

- 总计缓冲空间小于 10 000 个 4 KB 页。
- 找不到具备该应用程序知识来执行专门调试的人。
- 您在一个测试系统上工作。

如果您的系统不受上述条件限制，可考虑使用多个缓冲池以实现以下潜在的性能改进:

- 您可以把临时表空间放入单个缓冲池中，以便为需要临时存储器的查询尤其是执行大量排序的查询提供更佳性能。
- 如果您的数据必须由很多小的更新事务应用程序反复地快速存取，应当考虑将包含该数据的表空间移入单独的缓冲池内。如果此缓冲池的大小定得合适，将有更多的机会找到它的页，以利于缩短响应时间和降低事务成本。
- 您可以将数据隔离到单独的缓冲池中，以利于特定的应用程序、数据和索引。例如，您可能要将频繁更新的表和索引置于一个单独的缓冲池中，与那些虽频繁查询但不频繁更新的表和索引分开。此更改将减少频繁更新（对第一组表）对频繁查询（对第二组表）的影响。
- 对于很少使用的应用程序所存取的数据，尤其是在一个应用程序需要对一个很大的表非常随机地进行存取时，您可使用较小的缓冲池。在这种情况下，没有

必要使数据保存在缓冲池内存中的时间超过单个查询的保存时间。最好保留此数据所用的小缓冲池，释放多余的内存用于其他用途（例如，用于其他缓冲池）。

- 当将不同的活动和数据隔离到单独的缓冲池中后，可从统计信息和记帐跟踪获得一个良好的、成本较低的性能诊断数据。

将数据预读取至缓冲池

将索引和数据页预读取至缓冲池，可以减少等待 I/O 完成所用的时间，以有助于改进性能。*预读取*页就是在将要使用那些页之前，提前从磁盘中检索一个或多个页。有两种类别的预读取:

- *顺序预读取*是应用程序在需要这些页之前，将连续的页读至缓冲池中的一种方法。（参见『了解顺序预读取』。）
- *列表预读取*或列表顺序预读取，是甚至在所需数据页并不连续时，也能有效存取数据页的一种方法。（参见第216页的『了解列表预读取』。）

这两种读取数据页的方法是对常规读取方法的补充。当仅有一页或少量连续页要检索时，使用常规读取方法。在常规读取期间，只传送一页数据。

有关启用预读取的详情，参见第216页的『配置 I/O 服务器启用预读取和并行 I/O』。

了解顺序预读取

使用单一 I/O 操作将几个连续的页读到缓冲池中可以大大减少运行应用程序所需的额外开销。此外，以并行方式执行多个 I/O 操作以同时读入连续几页，可帮助减少应用程序等待 I/O 操作完成所需的时间。

当数据库管理程序确定顺序 I/O 是合适的且预读取可有助于改善性能时，启动预读取。对于象表扫描和表排序这类情况，数据库管理程序可以轻易地确定顺序预读取将有利于改进 I/O 性能。在这些情况下，数据库管理程序会自动启动顺序预读取。以下示例可能需要表扫描，且较适合顺序预读取:

```
SELECT NAME FROM EMPLOYEE
```

可在 CREATE TABLESPACE 或 ALTER TABLESPACE 语句中使用 PREFETCHSIZE 子句，来为每个表空间定义数据库管理程序将预读取的页数。在 SYSCAT.TABLESPACES 系统目录表的 PREFETCHSIZE 列中维护指定的值。

一个好的方法是将 PREFETCHSIZE 值显式地设置为您表空间的 EXTENTSIZE 值与表空间容器数相乘的一个倍数。（数据块大小是在使用另一个容器前数据库管理

程序可写到一个容器中的页数；*参考管理指南: 计划*中的“设计和选择表空间”。）例如，如果数据块大小为 16 页，且该表空间有两个容器，则您可以选择将预读取量设置为 32 页。

数据库管理程序监控缓冲池的使用，以确保在另一工作单元需要这些页的情况下，数据预读取不会从缓冲池中除去这些页。为避免引起问题，可选择将预读取的页数限制到小于您为表空间所指定的数的一个值。

预读取大小的设置可以显著影响性能，特别是对大表扫描更是如此。可使用数据库系统监控程序和其他系统监控工具来帮助您调整表空间的 PREFETCHSIZE。例如，您可以收集有关如下情况的信息:

- 有 I/O 等待您的查询，这可使用您的操作系统可用的监控工具来获知。
- 正在进行预读取，可查看数据库系统监控程序提供的 *pool_async_data_reads*（*缓冲池异步数据读取*）数据元素来获知。有关详情，参考 *System Monitor Guide and Reference*。

如果有 I/O 等待并且该查询是预读取数据，您可尝试增加 PREFETCHSIZE 的值。也有可能预读取程序不是 I/O 等待的原因，这种情况下增加 PREFETCHSIZE 值将不会改进您查询的性能。

在所有预读取类型中，当预读取大小是表空间的数据块大小的倍数，并且该表空间的数据块是在单独的容器中时，可以并行执行多个 I/O 操作。为获得更好的性能，应当将容器配置为使用单独的物理设备。有关并行预读取的详情，参见第216页的『配置 I/O 服务器启用预读取和并行 I/O』。

了解顺序检测

有的情况并不能立即确定顺序预读取是否将改进性能。在这类情况下，数据库管理程序可以监控 I/O，且如果正在读取顺序页，则数据库管理程序可以激活预读取。这种情况下，在认为合适时，数据库管理程序可以激活和禁用预读取。这种类型的顺序预读取称为*顺序检测*，并且同时适用于索引页和数据页。可以使用 *seqdetect* 配置参数（参见第327页的『顺序检测标志 (seqdetect)』）来控制数据库管理程序是否应执行顺序检测。如果启用顺序检测，则它可以确定下列 SQL 语句是否从顺序预读取受益:

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

在这个例子中，优化器可能已经选择使用 EMPNO 列上的索引来扫描该表。如果该表与此索引密切相关，则数据页读取将几乎是顺序的，并且预读取也可改进性能。在这种情况下，将执行数据页预读取。

索引页预读取也可能发生在此示例中。如果不得不检查大量索引页，并且数据库管理程序检测到正在进行索引页的顺序页读取，则将进行索引页预读取。

了解列表预读取

*列表预读取*或列表顺序预读取，是甚至在所需的数据页并不连续时，也能有效存取数据页的一种方法。列表预读取可以与单个或多个索引存取结合使用。

预读取和分区内并行性

预读取对于分区内并行性的性能很重要，此分区内并行性在扫描一个索引或表时使用多个子代理程序。这些并行扫描产生更大的数据耗用速率，并需要更高的预读取速率。

对于少量预读取的成本，并行扫描比串行扫描更高。如果在执行一个串行扫描时未进行预读取，则由于代理程序始终需要等待 I/O，所以该查询运行起来更慢。如果在执行一个并行扫描时未进行预读取，则所有子代理程序可能需要等待一个在等待 I/O 的子代理程序。

由于其重要性，使用分区内并行性可以更主动地执行预读取。顺序检测方法容许相邻页之间有更大的间隙，以便可以将这些页视为顺序页。这些间隙的宽度随着扫描中涉及的子代理程序数目的增加而增加。

配置 I/O 服务器启用预读取和并行 I/O

要启用预读取，数据库管理程序要启动不同的控制线程（称为*I/O 服务器*）来执行页读取。因此，查询处理分为两个并行的活动：数据处理 (CPU) 和数据页 I/O。I/O 服务器等待从 CPU 处理活动发来的预读取请求。这些预读取请求包含满足期望的数据需要所需的 I/O 的说明。预读取的原因确定数据库管理程序何时及如何生成预读取请求。（有关详情，参见第214页的『了解顺序预读取』和『了解列表预读取』）。

下图举例说明如何使用 I/O 服务器来将数据预读取到缓冲池中。

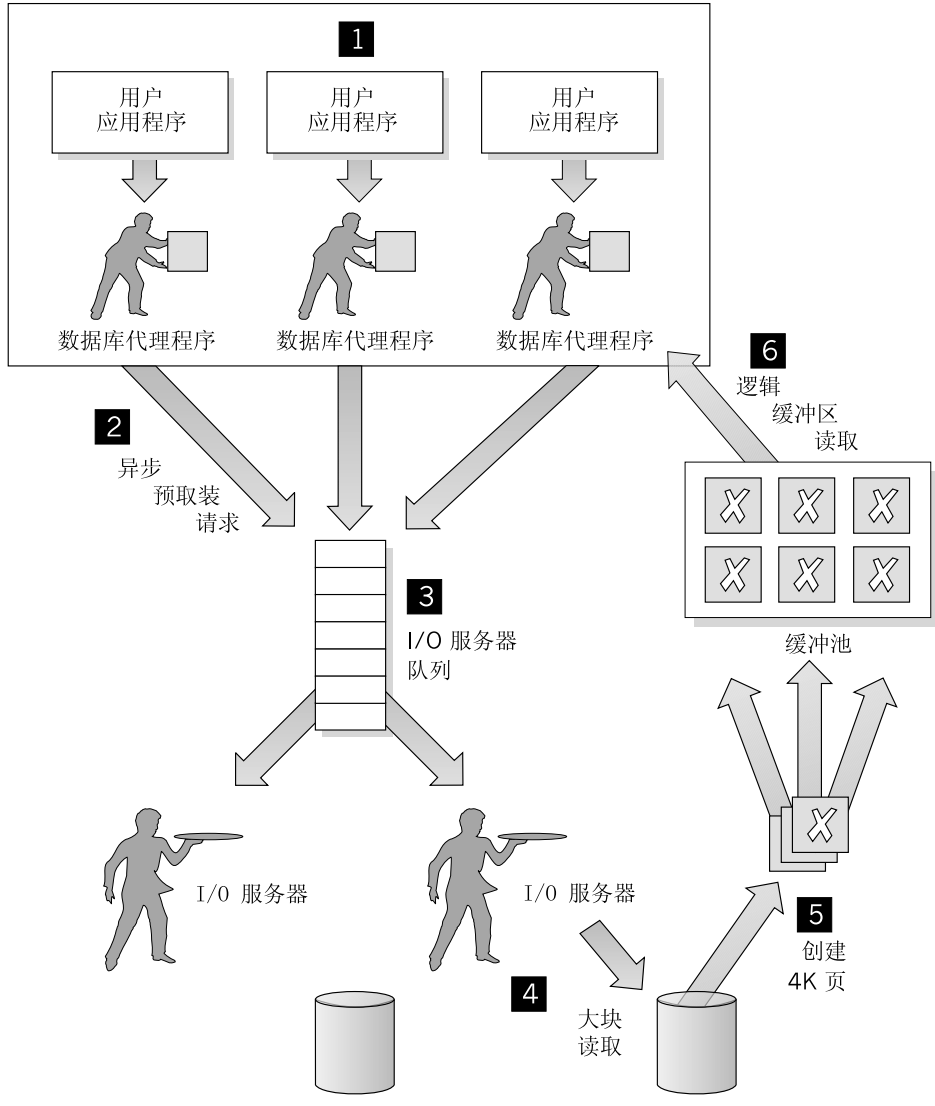


图 27. 使用 I/O 服务器预读取数据

在图27中举例说明了下列步骤:

1 用户应用程序将 SQL 请求传送到数据库代理程序。

2, **3**

数据库代理程序确定应使用预读取来获取满足该 SQL 请求所需的数据，并向 I/O 服务器队列写入一个预读取请求。

4, **5**

第一个可用的 I/O 服务器将从这个队列读取预读取请求，并从表空间将数据读入缓冲池。根据该队列中预读取请求的数目以及 *num_ioservers* 配置参数配置的 I/O 服务器的数目，多个 I/O 服务器可以同时从表空间读取数据。

6 数据库代理程序对缓冲池中的数据页执行需要的操作，以便将 SQL 请求的结果返回到用户应用程序。

使用配置参数 *num_ioservers* 配置足够的 I/O 服务器，可极大地提高使用数据预读取的查询的性能。配置几个额外的 I/O 服务器将不会降低性能，因为额外的 I/O 服务器不会用上而且它们的内存页将被调出。每个 I/O 服务器进程都被编号，而数据库管理程序将始终使用可用的号码最小的进程，因而某些号码较大的进程可能永远用不到。

要确定应配置的 I/O 服务器的数目，考虑下列因素:

- 对此数据库执行的并行活动的数量。即，在任何给定时间可将预读取请求写入 I/O 服务器队列的数据库代理程序的数目。
- I/O 服务器可并行工作的最大程度。有关详情，参见『启用并行 I/O』。

要使并行 I/O 的机会最大化，请将 *num_ioservers* 设置为至少等于数据库中的物理磁盘数。

启用并行 I/O

表空间存在多个容器的情况下，数据库管理程序启动*并行 I/O*。并行 I/O 指数据库管理程序使用多个 I/O 服务器处理单个查询的 I/O 要求的能力。每个 I/O 服务器被分配单个容器的 I/O 工作负荷，这样可以并行读取几个容器。以并行方式执行 I/O 可以显著增加 I/O 吞吐量。

当单个的 I/O 服务器将处理每个容器的工作负荷时，执行并行 I/O 的 I/O 服务器的实际数目将限制为请求的数据所分布的物理设备的数目。这意味着您需要的 I/O 服务器的数目等于物理设备的数目。

并行 I/O 的启动、使用方式取决于执行 I/O 的原因:

- **顺序预读取**

  对于顺序预读取，当预读取大小是一个表空间的数据块大小的倍数时，启动并行 I/O。然后沿块边界将每个预读取请求分为多个更小的请求。这些较小的请求再分配给不同的 I/O 服务器。

- **列表预读取**

对于列表预读取，根据存储数据页的容器，将每个页列表分成更小的列表。然后把这些更小的列表分配到不同的 I/O 服务器。

- **数据库或表空间的备份和复原**

  为了备份或复原数据，并行 I/O请求的数目等于备份缓冲区大小除以数据块大小所得的商，最大值可大到达容器数。

- **数据库或表空间的复原**

  为了复原数据，采取用于顺序预读取一样的方式来启动和分割并行 I/O 请求。直接将数据从复原缓冲区移动到磁盘中，而不是将数据复原到缓冲池中。

- **装入**

  在装入数据时，您可以使用 LOAD 命令的 DISK_PARALLELISM 选项，来指定 I/O 并行性的级别。（如果未指定它，则使用缺省值，它基于与该表相关的所有表空间的表空间容器的累计数目。）

为保证并行 I/O 的优化性能，确保:

- 有足够的 I/O 服务器。I/O 服务器的数目应配置为稍大于用于数据库内所有表空间的容器的数目。
- 表空间的数据块大小和预读取大小是合理的。预读取大小不应太大，以防止过分使用缓冲池。（理想的大小是数据块大小和表空间容器数之积。）数据块大小应该稍小些，合适的值在 8 至 32 页之间。
- 配置容器驻留在不同的物理驱动器上。
- 为了确保一致的并行度，所有容器要有相同大小。

  如果一个或多个容器比其他容器小，将使优化并行预读取的可能性降低。例如:

  – 在一个较小的容器充满后，附加的数据会储存在其余的一些容器中，导致容器变得不平衡。不平衡的容器会降低并行预读取的性能，因为可从中预读取数据的容器数目也许小于容器的总数。

  – 如果以后添加了一个较小的容器并且对数据进行了重新平衡，则此较小容器包含的数据要少于其他容器。相对于其他容器，此容器的少量数据将不会优化并行预读取。

  – 如果某个容器较大且所有其他的容器都已填充，该容器将是存储附加数据的唯一容器。数据库管理程序将不能使用并行预读取存取附加的数据。

- 当使用分区内并行性时，存在足够的 I/O 容量。可在 SMP 机器上使用分区内并行性，以便在多处理器上运行查询来减少一个查询的经过时间。必需足够的 I/O 容量来使每个处理器都忙于工作，一般需要附加的物理驱动器来提供该 I/O 容量。

必须较频繁地进行预读取，以便有效地利用 I/O性能。预读取大小应较大，这样可以较频繁地进行预读取。预读取大小必须是数据块大小与表空间容器数的乘积。理想情况下，应当把容器配置成驻留在单独的物理驱动器上。

要求的物理驱动器的数目取决于驱动器和 I/O 总线的速度和性能，以及处理器的速度。

一次分配多页

按需求扩充 SMS 表空间。缺省情况下，一次只扩充一页。但是，在某些工作负荷中（如执行块插入），可使用 *db2empfa* 工具告知 DB2 以页组或块组大小扩展表空间，从而提高性能。*db2empfa* 工具位于 *sqllib* 目录的 *bin* 子目录中。运行它会使数据库配置参数 *multipage_alloc* 设置为 YES。有关此工具的详情，参考 *Command Reference*。

在第234页的『扩充内存』中讨论了另一种使用可用内存的最佳方法

排序

常常需要对一个查询排序，排序堆区域的适当配置对查询的性能很重要。在以下情况下需要排序:

- 没有索引满足请求的定序（例如使用 ORDER BY 子句的 SELECT 语句）。
- 索引存在，但排序会比用索引更有效
- 创建一个索引（如果将 *indexsort* 配置参数设置为 YES）。

排序的不同类型

排序分两个步骤:

1. 排序阶段
2. 返回排序结果的阶段。

在这两个步骤中排序过程的不同，将产生不同类别或类型的排序。当考虑排序阶段时，可将排序分类为“溢出的”或“非溢出的”。当考虑返回排序阶段的结果时，可将排序分类为“管道式”或“非管道式”。

溢出的和非溢出的

如果正排序的信息不能完全放入排序堆（每次执行排序时分配的一块内存）中，则溢出至临时数据库表。未溢出的排序总是比那些溢出的排序执行得好。

管道式和非管道式

如果已排序的信息可直接返回，而不需要一个临时表来存储数据的最终排

序列表，则称为“管道式排序”。如果已排序的信息需要返回一个临时表，则称为“非管道排序”。一个管道排序始终比一个非管道排序执行得好。

调整影响排序的参数

下列情况影响排序的性能:

- 下列配置参数的设置:

  第303页的『排序堆大小 (sortheap)』
  指定要用于每个排序的内存量

  第303页的『排序堆阈值 (sheapthres)』
  控制可用于整个实例中所有排序的内存总量。
- 涉及到大量排序的语句
- 丢失可帮助避免不需要的排序的索引
- 不将排序最小化的应用逻辑
- 并行排序，它改善排序性能，但只有在语句使用分区内并行性时才可进行（参见第218页的『启用并行 I/O』）。

查找排序性能问题的指示符

要识别您是否有排序上的整体问题，可将排序所花的总计 CPU 时间与整个应用程序所花的该时间比较。数据库系统监控程序可提供帮助（参见第232页的『使用数据库系统监控程序』）。尤其是，“性能监控程序”（由“快照监控程序”和“事件监控程序”组成，且可从控制中心获取），在缺省情况下显示*总计排序时间*以及其他时间，如 *I/O* 和*锁定等待*。

如果总计排序时间占了其他时间的一大部分，可查看下列值，缺省情况下也显示它们。

溢出排序的百分比
此变量（有关“快照监控程序”的性能详细资料视图）显示溢出的排序的百分比。如果溢出排序的百分比高，且如果有任何后阈值排序，则增加 *sortheap* 和／或 *sheapthres* 配置参数。（要确定是否有任何后阈值排序，使用“快照监控程序”。）

后阈值排序
如果后阈值排序多，则增加 *sheapthres* 和／或减少 *sortheap*。

一般情况下，使整个实例可用的总排序内存 (*sheapthres*) 尽可能大，而不会导致过度分页。完全在排序内存中执行一个排序是可能的，但是，如果这导致操作系统

执行过多的页交换以容纳该排序内存，您可能失去大排序堆的优势。所以，无论何时您调整排序配置参数，要使用一个操作系统监控程序来跟踪系统分页中的任何变化。

注: 随着 DB2 部分关键字二进制排序技术的提高，已包含了非整数数据类型关键字，因此排序长关键字时需要附加的内存。如果您认为正在使用长关键字，则须增加 *sortheap* 配置参数

还要注意在管道式排序中，在应用程序关闭与排序关联的游标前，不会释放排序堆。因此，一个管道排序可用尽内存，直到游标被关闭。

管理排序性能的技术

您可以使用数据库系统监控程序和基准测试技术，来帮助设置 *sortheap* 和 *sheapthres* 配置参数。对每个数据库管理程序和它的数据库执行下列操作:

- 建立并运行一个代表性工作负荷。
- 对于每个适用的数据库，在基准测试工作负荷周期内对以下性能变量收集其平均值:
  - 正在使用的总计排序堆
  - 活动的排序

  在“快照监控程序”的性能详细资料视图上显示了这些性能变量。
- 将 *sortheap* 设置为每个数据库平均*使用的排序堆总和*。
- 通过执行下列各项来设置 *sheapthres*:
  1. 确定实例中哪个数据库具有最大的 *sortheap* 值。
  2. 确定此数据库的排序堆平均大小。

     如果太难于确定的话，则使用最大排序堆的 80%
  3. 将 *sheapthres* 设置为平均活动排序次数乘以上面计算的平均排序堆大小。

     这是建议的初始设置。以后可以使用基准测试技术来细化此值。

也可标识排序对性能有主要影响的特定应用程序和语句:

- 将事件监控程序设置在应用程序级别和语句级别，以帮助您标识总计排序时间最长的应用程序。
- 在其中每个应用程序中，查找具有最长*总计排序时间*的语句。
- 使用如 Visual Explain 的工具调整这些语句。
- 确保存在适当的索引。使用 Visual Explain 标识给定语句的所有排序操作。然后审查对于此语句存取的每个表，是否存在一个适当的索引。

注: 您可以搜索解释表，以标识哪些查询有排序操作。（参见第475页的『附录C. SQL 解释工具』。）

重组目录和用户表

使用索引的 SQL 语句的性能，在执行了许多更新、删除或插入后，可能降低。一般情况下，不能把新插入的行放在与该索引定义的逻辑序列相同的物理序列中（除非您使用群集索引）。这表示数据库管理程序必须执行附加的读取操作来存取数据，因为逻辑顺序数据可能在不同的、非顺序的物理数据页上。

一般情况下，重组一个表所花时间比运行统计所花时间多。通过获得当前的数据统计信息并重新联编应用程序可显著改进性能，因此先试此操作。如果这样不能改进性能，也许是没有有效地安排好这些表和索引中的数据，因此重组可能有帮助。本小节中的信息不仅适用于重组您自己的表，也适用于可能也需要重组的系统目录表。

对于类型表，指定的表名必须是层次结构的根表的名称。

REORGCHK 命令返回有关一个表的物理特征的信息，以及重组该表是否有好处的信息。可通过命令行处理器使用该命令。参考 *Command Reference* 以获取详情，包括如何解释命令输出。

注: REORGCHK 命令既不显示扩充索引的任何数据，也不显示已说明临时表的数据。

REORG 实用程序可以选择根据指定的索引将数据重新排列成一个物理序列。REORG 有一选项，可用来指定具有索引的表中各行的顺序，因此可根据索引群集表数据，改进 RUNSTATS 实用程序搜集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。这样，可以更有效地处理要求行按索引次序排序的 SQL 语句。通过除去无用的、空的空间，REORG 也可更紧凑地存储这些表（即使您在使用 ALTER TABLE 时指定了 PCTFREE，该空间仍会保持为无用状态）。

不要使用具有别名的 REORG 或 REORGCHK 命令。

REORG 实用程序要求通常基于受影响的表数据和索引工作的所有其他应用程序是脱机的。不要在 REORG 或 REORGCHK 命令中使用别名。在此环境中，可考虑使用联机索引重组实用程序。

您可能应当考虑如下因素，以确定何时重组您的表数据:

- 插入、更新和删除的活动量
- 对于使用具有高群集比率的索引的查询，对其性能的任何显著更改

- 运行统计 (RUNSTATS) 不能改进查询的性能
- REORGCHK 命令指示需要重组表
- 重组表的成本，包括 CPU 时间、经过时间，以及在重组完成以前，因 REORG 实用程序锁定该表而引起的降低的并行性。

要执行 REORG 实用程序，您必须对该表有 SYSADM、 SYSMAINT、SYSCTRL 或 DBADM 权限，或 CONTROL 特权。

如果向一个表中添加了列，或一个表有 LOB 列，则 REORG 实用程序将使用可以比原始表大很多的临时表。如果这些临时表较大，则由 REORG 实用程序创建的结果永久表也会较大。

对于类型表，指定的表名必须是层次结构的根表的名称。

注: 不能使用 REORG 实用程序来重组已说明临时表。

REORG 实用程序允许您指定一个临时表空间，用于创建临时 REORG 表。如果未指定一个临时表空间，则 REORG 实用程序将在包含要重组的表的表空间中创建临时 REORG 表。下列准则可辅助您确定是否使用一个临时表空间:

- 如果指定临时表空间，则一般建议您指定 SMS 临时表空间。不建议使用 DMS 临时表空间，因为您只能有一个正在进行中的 REORG 在使用这种类型的表空间。
- 如果未指定一个临时表空间，则 REORG 实用程序将在包含要重组的表的表空间中创建临时 REORG 表。一般不建议使用临时 DMS 表空间，因为您只能有一个正在进行中的 REORG 在使用这种类型的表空间。

REORG 实用程序隐式关闭所有打开的游标。

记住您可能正在一个页大小超过 4 KB（8 KB、16 KB 或 32 KB）的表空间中重组表。重组期间使用的临时表空间必须与基本表空间有相同大小的页。

如果 REORG 实用程序未成功完成，则**不要**删除任何临时的文件、表或表空间。数据库管理程序使用这些文件和表来回滚由 REORG 实用程序所做的更改，或完成重组，这取决于在失败前重组进行到什么程度。

在一个分区数据库中，REORG 实用程序在每个分区上重组数据。如果实用程序在任何分区失败，则只要回滚这个失败的分区。如果指定一个存储临时表的目录路径，数据库管理程序会在每个数据库分区扩展此路径。因此，如果您指定一个由其他数据库分区共享的路径，则可在此路径下的不同子目录（由节点名标识）中存储临时文件。

联机索引重组

对索引叶页上的最大空闲空间容量提供用户可定义的阈值，可进行联机重组。当从叶页删除了索引关键字且超过阈值时，可检查相邻的索引叶页以确定是否可合并两个叶页。如果某一页上有足够的空间用于合并两个相邻的页，则可以进行合并而不必使该数据库脱机。

索引的这种联机重组只能对此发行版和后续发行版中创建的索引使用。要求能够以这种方式联机重组的现存索引必须删除，然后重建，以便对索引叶页进行必要的内部更改。要对特定索引启用联机索引重组，当创建索引时指定 MINPCTUSED 值。应将 MINPCTUSED 值设置为小于 100。此值成为重组阈值，它是在尝试将一个索引叶页与其相邻的叶页合并之前在该索引页上已用空间的百分比，这是可接受的最小值。MINPCTUSED 的建议值是小于 50% 的一个值，因为目标是将两个相邻的索引叶页合并。MINPCTUSED 的零值（也是缺省值）禁用联机重组。

在联机索引重组后释放的索引叶页可以重新使用。但释放的页只能用于同一个表中的其他索引。当使用 DMS 存储器模型时，全面地重组表将释放其他对象所用的页；当使用 SMS 存储器模型时，将释放磁盘空间。

在联机索引重组后不释放索引非叶页来使用。但表的全面重组可以使索引尽可能地小。会减少叶页和非叶页的数目以及索引的级别。

降低重组表的需要

为降低重组表的需要，在创建该表后执行下列操作:

- 改变表以添加 PCTFREE
- 在索引上使用 PCTFREE 来创建群集索引
- 将数据排序
- 装入数据。

现在就有了带群集索引的表。群集索引，与表上的 PCTFREE 一起，将保留原始排序次序。当页上有足够的空间时，可在正确的页上插入新数据，因此维护了群集索引的群集特征。随着插入更多的数据，如果表的页变满，则将记录追加至表的末尾，而该表逐渐变成非群集的。

建议在创建一个群集索引后，执行 REORG 或排序，然后执行 LOAD。群集索引试图维护数据的特定次序，以改进 RUNSTATS 实用程序收集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。

在 REORG 期间在每一页上要留下的空闲空间量由该表的 PCTFREE 值确定。如果未设置此值，则重组数据时 REORG 将把页填满。

DMS 设备的性能考虑事项

如果对表空间使用“数据库管理的存储器”(DMS) 设备容器，需要了解如下事项，以便可以有效地管理环境:

- **文件系统高速缓存**

  执行文件系统高速缓存，如下所示:

  – 对于 DMS 文件容器（和所有 SMS 容器），操作系统可能会将页高速缓存在文件系统高速缓存中

  – 对于 DMS 设备容器表空间，操作系统不会将页高速缓存在该文件系统高速缓存中。

  注: 当在 Windows NT 上工作时，注册表变量 DB2NTNOCACHE 指定 DB2 是否用 NOCACHE 选项打开数据库文件。如果 DB2NTNOCACHE=ON，则禁用文件系统高速缓存。如果 DB2NTNOCACHE=OFF，则操作系统高速缓存 DB2 文件。这适用于除包含 LONG FIELDS 或 LOBS 的文件以外的所有数据。禁用系统高速缓存使数据库有更多内存可用，这样可以增加缓冲池或排序堆。

- **数据的缓冲**

  从磁盘读取的表数据通常可在数据库的缓冲池中找到（参见第209页的『管理数据库缓冲池』）。在某些情况下，在应用程序实际使用该页之前，可从缓冲池释放一个数据页。（如果其他数据页需要缓冲池空间，则可能发生此情况。）对于使用系统管理的存储器 (SMS) 或数据库管理的存储器 (DMS) 文件容器的表空间，可参见上面的文件系统高速缓存的说明。这可以消除在其它方面应是必需的 I/0。

  使用数据库管理的存储器 (DMS) 设备容器的表空间**不**使用文件系统或它的高速缓存。因此，您可能想增加数据库缓冲池的大小，减少文件系统高速缓存的大小，以修正这样一个事实，即使用设备容器的 DMS 表空间并未执行双重缓冲。

  如果您通过使用系统级别的监控工具注意到与等效的 SMS 表空间相比，使用设备容器的 DMS 表空间的 I/O 更高，这种差别可能是由于上面讨论的双重缓冲所导致的。

- **使用 LOB 或 LONG 数据**

  当一个应用程序检索 LOB 或 LONG 数据时，数据库管理程序不使用它的缓冲区来高速缓存该数据。每次应用程序需要其中一页时，数据库管理程序必须从磁盘中检索它。

  但是，如果 LOB 或 LONG 数据存储在 SMS 或 DMS 文件容器中，文件系统高速缓存就可提供缓冲，因此也就改善了性能。

因为系统目录包含一些 LOB 列，建议将它们保存在 SMS（或者在 DMS 文件）表空间中。

管理初始化额外开销

ACTIVATE DATABASE 命令启动选择的数据库。在一个分区数据库中使用此命令，将导致尝试在所有数据库分区上激活选择的分区数据库。通过使用此命令，应用程序不用在数据库初始化或启动时花时间。

使用 ACTIVATE DATABASE 命令初始化的数据库必须用 DEACTIVATE DATABASE 命令来停止；与该数据库断开的最后一个应用程序将不停止。有关 ACTIVATE 和 DEACTIVATE 命令的详情，参考 *Command Reference* 手册。

如果一个数据库尚未启动，并且在一个应用程序中遇到 CONNECT TO （或一个隐式连接），则在数据库管理程序启动所需的数据库时应用程序必须等待，启动完毕后该应用程序才可对该数据库做任何处理。这是存取特定数据库的第一个应用程序所花费的启动成本。在一个分区数据库中，在每个数据库分区上都会产生此启动成本。一旦启动了该数据库，所有其他应用程序都可以连接并使用这个数据库，而不用花时间来单独启动该数据库。

数据库代理程序

DB2 服务器必须为在数据库管理程序和客户机及本地应用程序之间的通信提供方便。基于 UNIX 的环境使用基于*进程*的体系结构。例如， DB2 通信监听程序是作为进程创建的。Intel 操作系统如 OS/2 和 Windows NT 使用基于*线程*的体系结构以最大地提高性能。例如， DB2 通信监听程序是作为 DB2 服务器的系统控制器进程内的线程创建的。对于要存取的每个数据库，可启动不同的进程／线程来处理不同的数据库任务（例如，预读取、通信和记录）。

最关键的进程／线程之一是那些数据库代理程序，它们简化应用程序对数据库执行操作。

*逻辑代理程序*表示与数据库管理程序相连的应用程序。逻辑代理程序有应用程序所需的所有信息和控制块。最大逻辑代理程序数由 *max_logicagents* 数据库管理程序配置参数控制。因为每个应用程序都将有一个逻辑代理程序，所以此参数控制的是可以与实例相连的最大应用程序数。

*工作代理程序*执行应用程序请求，但并不与任何特定应用程序永久相连。工作代理程序有在数据库管理程序中完成应用程序所请求的操作所需的所有信息和控制块。

工作代理程序共有四种: *活动协调代理程序*、*子代理程序*、*不活动代理程序*和空*闲代理程序*。

空闲代理程序是最简单的工作代理程序形式: 它不与逻辑代理程序相连，它不带出站连接，它不带本地数据库连接或实例连接。

不活动代理程序是这样的工作代理程序: 它不处于活动事务中，不与逻辑代理程序相连，不带出站连接，不带本地数据库连接或实例连接。不活动代理程序可以与另一逻辑代理程序相连，以开始为该逻辑代理程序所表示的应用程序提供服务。

一个客户机应用程序的每个进程 / 线程都有一个在数据库上运行的*活动协调代理程序*。一旦创建了协调代理程序，它就代表它的应用程序执行所有数据库请求并使用进程间通信 (IPC) 或远程通信协议与其他代理程序进行通信。每个代理程序使用它自己的专用内存来运行，并与其他代理程序共享数据库管理程序和数据库全局资源，如缓冲池。当事务完成时，活动协调代理程序可以与逻辑代理程序拆离，因而成为不活动代理程序。

在分区数据库环境和启用了分区内并行性的环境中，协调代理程序会将数据库请求分发至*子代理程序*，然后由这些代理程序执行对该应用程序的请求。一旦创建了该协调代理程序，它将通过协调执行数据库请求的子代理程序，来代表它的应用程序处理所有数据库请求。

当客户机断开同数据库的连接或拆离了同实例的连接时，协调代理程序将为:

- 活动代理程序。如果其他逻辑代理程序正在等待，则工作代理程序将成为活动协调代理程序。
- 被释放并标记为空闲（如果没有其他逻辑代理程序正在等待且尚未达到缓冲池代理程序的最大数目的话）。
- 被终止并释放其存储器（如果没有其他逻辑代理程序正在等待且已达到缓冲池代理程序的最大数目的话)。

那些不代表任何应用程序执行工作且正在等待分配的代理程序被认为是空闲代理程序，并驻留在一个*代理程序缓冲池*中。这些代理程序可用于处理代表客户机程序运行的协调代理程序发出的请求，或可用作代表现存协调代理程序运行的子代理程序。可用的代理程序数目取决于数据库管理程序配置参数 *maxagents* 和 *num_poolagents*。

重用代理程序池 (*num_poolagents*) 中的代理程序作为协调代理程序:

- 对于基于 TCP/IP 的远程应用程序; 或
- 对于在基于 UNIX 的操作系统上的本地应用程序; 或

- 对于 Windows NT 和 OS/2 操作系统上的本地和远程应用程序。

否则，远程应用程序总是创建新代理程序。

当请求一个代理程序时，如果没有空闲的代理程序存在，必须动态创建一个新的代理程序。创建一个新代理程序涉及到一定量的额外开销，因此，如果有可为客户机激活的空闲代理程序，可明显改进 CONNECT 和 ATTACH 的性能。

当一个子代理程序代表一个应用程序工作时，可认为它是与那个应用程序*相关的*。当完成指定的工作后，可能会将它放在代理程序缓冲池中，但它仍然与原始应用程序相关。当该应用程序请求附加的工作时，数据库管理程序在查找为这个应用程序工作的代理程序时，首先检查相关代理程序的空闲缓冲池。

单独控制连接的应用程序数（通过使用由 *max_logicagents* 定义的逻辑代理程序数）和可以处理的应用程序请求数（通过使用由 *max_coordagents* 定义的活动协调代理程序数）的能力使得在数据库上处理的工作负荷具有很大的灵活性。连接的应用程序数与可以处理的应用程序请求数之间的一对一关系是应用程序使用数据库的典型方法。然而，您的工作环境可能要求在连接的应用程序数与可以处理的应用程序请求数之间存在多对一关系。

因为数据库全局资源额外开销与活动协调代理程序相关联，所以这些代理程序数越大就意味着达到可用数据库全局资源上限的机会也就越大。您可能想允许连接的应用程序数多于活动协调代理程序数，以便不会达到可用数据库全局资源上限。通过将 *max_logicagents* 的值设置为大于 *max_coordagents* 的值，您可以专注于数据库工作。

有关如何使用 DB2 Connect 作为 XA 事务支持集中器的详情和示例，请参考 *DB2 Connect 用户指南*。

当工作环境要求使用 DB2 Connect 同远程系统连接时，则有一个*出站连接池*。此连接池可缩短同主机连接的时间（在第一次连接后）。当请求与主机断开时，DB2 Connect 会卸下入站连接而在存储池中保留与主机的出站连接。当一个新的请求要与主机连接时，DB2 Connect 重新使用该存储池中现有的出站连接（如果有的话）。

注: 当使用连接存储时，DB2 Connect 只能进行入站 TCP/IP、出站 TCP/IP 和 SNA 连接。当使用 SNA 时，安全性类型必须是 NONE，才可将连接置于存储池中。

有了连接存储，活动的代理程序在断开后不关闭其出站连接，而是保持与远程主机的活动连接并进入代理程序存储池。这种类型的代理程序称为*待用 DRDA 代理程序*。待用 DRDA 代理程序的存储池与出站连接存储池同义。

考虑以下示例，它们基于四种不同的用法和工作负荷要求:

1. 在第一个示例中，40 个并行用户（平均值）通过 DB2 Connect 与远程主机数据库连接。有时并行连接数最高可达到大约 50，但不会超过 55。事务处理时间短，用户频繁连接和断开。

   在这些情况下，系统管理员应将 MAX_COORDAGENTS 配置为 55，因为他知道无论尝试何种方法通过 DB2 Connect 可同时连接的最大用户数是 55。应将代理程序存储池大小 NUM_POOLAGENTS 设置为 40，因为在任何时候它都是已连接或尝试连接的平均用户数。此存储池大小保证可容纳足够多的现存远程数据库连接，以满足所有入站客户机，而不必建立任何新的连接，除非是在工作负荷达到峰值时。
2. 在第二个示例中，工作负荷高得多，它有大约 1 000 个入站客户机。用户连接的时间也短。系统管理员不想允许更多的并行连接。因此，系统管理员将 MAX_COORDAGENTS 和 NUM_POOLAGENTS 设置为 1 000。这表示可以同时与远程数据库连接的最大入站客户机数是 1 000。当所有客户机断开时，存储池将只包含 1 000 个已连接的代理程序，它们全在等待为新的入站客户机服务。
3. 第三个示例说明单个应用程序通过 DB2 Connect 与一个远程数据库连接。该应用程序长时间保持连接。在此方案中，代理程序和连接存储池的最佳配置是将 MAX_COORDAGENTS 设置为 1，因为我们知道最多只有一个客户机要连接。在此例中可将 NUM_POOLAGENTS 设置为零，因为没有与远程主机的频繁连接和断开。将 NUM_POOLAGENTS 设置为零，可有效禁止连接存储，因为存储池中没有保留与远程数据库有活动连接的代理程序。对于连接的每个新的入站客户机，会创建一个新代理程序，并建立一个新的远程连接来为它服务。
4. 第四个示例以前面三个方案为基础，并作了一些改变。在此示例中，系统管理员想把对远程数据库的并行存取限制为刚好 100。因此，将 MAX_COORDAGENTS 设置为 100，而且为了最大限度地提高连接性能，将 NUM_POOLAGENTS 设置为 100。但后来，还可能需要进行本地连接以监控装有 DB2 Connect 的系统上的工作负荷。期望在任何时候并行的监控程序快照不超过 5 个，因此将 MAX_COORDAGENTS 设置为 105。这个新配置值允许最大并行应用程序数超过前面所讲的上限 100，以容纳偶发的监控程序快照和／或实例连接。

对于分区数据库环境和启用分区内并行性的环境，每个分区（即每个数据库服务器或节点）有它自己的代理程序存储池，可从中抽出子代理程序。因为有此缓冲池，子代理程序就不必在每次需要时创建或完成其工作时破坏掉。这些子代理程序可以仍然作为此缓冲池中的相关代理程序，并由数据库管理程序使用，以执行与它们相关的应用程序发出的新请求。

下列数据库管理程序配置参数影响数据库代理程序的数目:

- 第336页的『代理程序的最大数目 (maxagents)』。一旦工作代理程序数达到此值，所有需要新代理程序的后续请求都会被拒绝，直到代理程序数降低到该值之下。此值适用于执行全部应用程序的代理程序总数，包括在所有应用程序上工作的协调代理程序、子代理程序、不活动代理程序和空闲代理程序。
- 第339页的『代理程序池大小 (num_poolagents)』。在代理程序缓冲池中的不活动代理程序、空闲代理程序以及关联子代理程序数不能超过此值。
- 第340页的『池中初始代理程序数 (num_initagents)』。当启动数据库管理程序时，根据此值创建一个工作代理程序的缓冲池。这提高了初始查询的性能。工作代理程序全都以空闲代理程序开始。
- 第339页的『逻辑代理程序的最大数目 (max_logicagents)』。最大逻辑代理程序数。因为每个应用程序都将有一个逻辑代理程序，所以此参数控制的是可以与实例相连的最大应用程序数。
- 第338页的『最大协调代理程序数 (max_coordagents)』。对于分区数据库环境和启用了分区内并行性的环境，此值限制协调代理程序的数目。
- 第337页的『并行代理程序的最大数目 (maxcagents)』。此值控制数据库管理程序允许的标记数。对于一个客户机与数据库连接时所进行的每个数据库事务（工作单元），协调代理程序必须从数据库管理程序获得许可权来处理该事务（称为处理标记）。数据库管理程序只允许带处理标记的代理程序对数据库执行工作单元。如果没有可用的标记，代理程序将一直等到有可用的标记为止，此时才处理请求的工作单元。

  在高峰使用需求超过系统资源（内存、CPU 和磁盘）的环境中此参数可能很有用。在这样的环境中，峰值负荷可能会由于如调页等原因而导致性能过度降低。您可以使用此参数来控制负荷，并避免性能降低。

对于分区数据库环境和启用分区内并行性的环境，对系统中性能和内存成本的影响与代理程序存储池是如何调整的有很大关系:

- 代理程序缓冲池大小的数据库管理程序配置参数 (*num_poolagents*) 影响可与一个分区（即节点）上的应用程序保持相关的子代理程序的总数。如果缓冲池大小太小（且该缓冲池是满的），则一个子代理程序将解除它自己与它代表的应用程序的关联，并终止。这种情况将导致很差的性能，因为必须常常创建子代理程序，并重新使它们与应用程序相关。

  另外，如果 *num_poolagents* 的值太小，则一个应用程序就可能填满有相关子代理程序的缓冲池。因此，当另一个应用程序需要新的子代理程序，并且在其相关的代理程序缓冲池中没有子代理程序时，它将从其他应用程序的代理程序缓冲池中“窃用”子代理程序。这种情况会增加成本，且导致很差的性能。
- 必须根据在任何给定时间允许过多代理程序活动的资源成本权衡上述情况。

例如，如果 *num_poolagents* 的值太大，则相关的子代理程序也许很长时间都留在缓冲池中未使用。这些子代理程序使用其他任务不可用的数据库管理程序资源。

除数据库代理程序外，还有数据库管理程序执行的其他异步活动，它们在自己的进程（或线程）中运行，包括:

- 数据库 I/O 服务器（或 I/O 预读取程序）（参见第214页的『将数据预读取至缓冲池』）
- 数据库异步页清除器（参见第209页的『管理数据库缓冲池』）
- 数据库记录器
- 数据库死锁检测器
- 事件监控程序
- 通信和 IPC 监听程序
- 表空间容器重平衡程序。

有关标识不同 DB2 进程的详情，参见 *Troubleshooting Guide*。

使用数据库系统监控程序

DB2 数据库管理程序维护有关它的操作、性能和使用它的应用程序的数据。当数据库管理程序运行时维护此数据，并且此数据可提供重要的性能和排除故障的信息。例如，您可以了解:

- 与一个数据库连接的应用程序数、它们的状态和每个应用程序正执行哪些 SQL 语句（如果存在的话）。
- 显示数据库管理程序和数据库是如何配置的以及帮助您调整它们的信息。
- 当特定数据库发生死锁时，涉及到哪些应用程序，又有哪些锁定处于争用中。
- 一个应用程序或一个数据库持有的锁定的列表。如果由于应用程序在等待锁定而不能继续执行，则存在有关该锁定的附加信息，包括哪个应用程序持有该锁定。

因为要收集一些这种数据会给 DB2 的操作带来额外开销，所以可用**监控开关**来控制要收集哪些信息。 要明确设置监控开关，使用 UPDATE MONITOR SWITCHES 命令或 sqlmon() API。（您必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限。）

有两种方法来存取数据库管理程序维护的数据:

- **捕捉快照**

从命令行使用 GET SNAPSHOT 命令；通过 OS/2、Windows 95 或 Windows NT 操作系统上的控制中心来使用图形界面，或使用 sqlmonss() API 调用编写自己的应用程序。

可从 DB2 文件夹或使用 db2cc 命令打开控制中心，它提供一个性能监控工具，通过提取快照定期对监控数据采样。此图形界面以细节和摘要的形式，提供快照数据的图形或文本视图。您也可以使用数据库监控程序返回的数据元素来定义性能变量。

控制中心的“快照监控程序”工具通过在性能变量上指定阈值，来允许您定义异常情况。当达到阈值时，可预先定义要发生的以下任何一个操作：通过窗口显示的通知或听得到的警报，和／或脚本或程序的执行。

如果从控制中心提取快照，当您在一个数据库对象（如实例或数据库）上或它的任何一个子对象上执行快照监控时不能执行改变、更改或删除该对象的操作。（另外，如果您正在监控一个分区数据库系统，则不能刷新分区数据库对象的视图。）例如，如果想要除去数据库 A 的实例，则不能监控数据库 A。然而，如果您只监控这个实例，则可以改变数据库 A。

要停止对一个实例（包括它的任何子对象）的所有监控，可从该实例的弹出菜单中选择**停止所有监控**。应总是停止对实例的监控，因为这确保释放性能监控程序所持有的所有锁定。

- **使用事件监控程序**

  当发生象事务结束、语句结束或检测到死锁这样的特定事件后，一个事件监控程序将捕捉系统监控程序信息。可以将此信息写入文件或命名管道中。

  要使用事件监控程序：

  1. 用控制中心或 SQL 语句 CREATE EVENT MONITOR 创建其定义。此语句在数据库系统目录中存储该定义。
  2. 通过控制中心或用以下的 SQL 语句激活事件监控程序：

     ```
     SET EVENT MONITOR evname STATE 1
     ```

     如果写入一个命名管道，则在激活事件监控程序之前，启动从这个命名管道中读取的应用程序。可编写自己的应用程序来执行此操作，或使用 **db2evmon**。一旦事件监控程序被激活，并且开始向管道中写入事件，则 **db2evmon** 将在生成这些事件时读取它们，并将它们写到标准输出中。
  3. 读取该跟踪。如果使用文件事件监控程序，则您可以查看它用下列任何一种方式创建的二进制跟踪：
     - 使用 **db2evmon** 工具来将跟踪格式化为标准输出。
     - 单击控制中心（在 Windows 95、Windows NT 或 OS/2 系统上）中的**事件分析程序**图符，以使用图形界面来查看跟踪、搜索关键字并过滤掉不想要的数据。

注: 如果监控的数据库系统不在控制中心所在的机器上运行，在可以查看跟踪前必须将事件监控文件复制到控制中心所在的机器上。另一种方法是将该文件置于两台机器都可存取的共享文件系统中。

有关系统数据库监控程序和事件监控程序的信息，参见 *System Monitor Guide and Reference*。

扩充内存

您的机器的可寻址实内存可能多于最大可寻址虚拟内存量（例如，在大多数平台上，可寻址虚拟内存通常在 2 GB 和 4 GB 之间）。您可以将可寻址虚拟内存外的任何附加可寻址实内存配置为*扩充高速缓存*。 这种扩充高速缓存可以供任何定义的缓冲池使用，并应当能改进数据库管理程序的性能。扩充高速缓存是用内存段来定义的。

当决定将一些可寻址实内存用作扩充高速缓存时，您应该知道，此内存不再能够用于机器上的其他用途，如 JFS 高速缓存或进程专用地址空间。向扩充高速缓存分配附加可寻址实内存会导致更高程度的系统分页。

DB2 在具有缓冲池的机器中使用可寻址内存（参见第209页的『管理数据库缓冲池』）。缓冲池将扩充高速缓存用作次级高速缓存（而缓冲池执行第一级高速缓存）。理想情况下，缓冲池可以保存那些最频繁存取的数据，而扩充高速缓存可以保存那些较不频繁存取的数据。

下列数据库配置参数影响可用作扩充存储器的内存容量和大小:

- *num_estore_segs* 定义扩充内存段的数目。此配置参数的缺省值为零，它指定不存在扩充高速缓存。（参见第330页的『扩充内存段数 (num_estore_segs)』。）
- *estore_seg_sz* 定义每个扩充内存段的大小。此大小受使用扩充高速缓存的平台限制。（参见第329页的『扩充内存段大小 (estore_seg_sz)』。）

因为扩充高速缓存是对缓冲池的扩充，所以它必须始终与一个或多个特定的缓冲池相关。因此，您必须在创建一个高速缓存后马上说明哪些缓冲池可以利用此高速缓存。CREATE 和 ALTER BUFFERPOOL 语句具有 NOT EXTENDED STORAGE 和 EXTENDED STORAGE 属性，它们控制高速缓存的使用。缺省情况下，IBMDEFAULTBP 和任何新创建的缓冲池都不使用扩充存储。

注: 如果使用用不同页大小定义的缓冲池，则禁用缓冲池的扩充存储器支持。

数据库管理程序不能直接操纵驻留在扩充高速缓存中的数据。然而，它能够以比从磁盘存储器传送高得多的速度将数据从扩充高速缓存传送至缓冲池。

当需要从一个扩充高速缓存中的一页得到一行数据时，会将这整页读到一个相应的缓冲池中。

当激活或第一次连接一个数据库时，会创建一个缓冲池及其相关的扩充高速缓存（如果定义了）。

第9章 使用控制器

使用控制器来监控和更改对数据库运行的应用程序的行为。

控制器由两部分组成:
- 前端实用程序
- 精灵程序

当启动控制器时，从控制器前端实用程序发出启动命令，然后它启动控制器精灵程序。缺省情况下，在一个分区数据库中的每个分区上都启动一个精灵程序，但您也可使用前端实用程序在特定分区上启动单个精灵程序，以监控在该处找到的数据库分区的活动。或者，一个精灵程序可监控单分区数据库上的活动。有关详情，参见『启动和停止控制器』。

每个控制器精灵程序收集关于对数据库运行的应用程序的统计信息。然后它根据您在控制器配置文件（适用于该特定数据库）中所指定的规则检查这些统计信息。（有关详情，参见第240页的『创建控制器配置文件』。）然后控制器根据这些规则进行操作。例如，某个规则可能指示一个应用程序使用的资源过多。在此情况下，控制器可根据您在控制器配置文件中指定的说明，更改该应用程序的优先级或强制它脱离该数据库。

如果与某个规则相关的操作是更改该应用程序的优先级，则控制器在检测到资源违规的数据库分区上更改该代理程序的优先级。如果与某个规则相关的操作是强制一个应用程序，则即使检测到该资源违规的控制器正在该应用程序的协调程序节点上或分区数据库环境中运行，也强制该应用程序。

控制器还记录它所执行的任何操作。您可查询日志文件，以复查控制器执行的操作。有关详情，参见第247页的『控制器日志文件』和第248页的『查询控制器日志文件』。

启动和停止控制器

使用 db2gov 控制器前端实用程序来启动或停止控制器（在全部数据库分区上或在单个数据库分区上）。您需要 SYSADM 或 SYSCTRL 权限才可使用该实用程序。

db2gov 的语法如下所示:

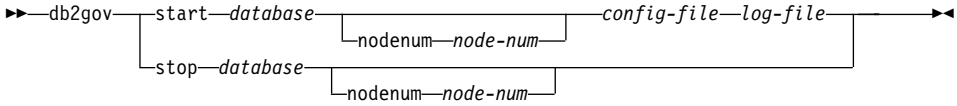


图 28. db2gov 的语法

这些参数如下所示:

start **database**

启动控制器精灵程序以监控指定的数据库。对于 *database*，您可指定数据库名或数据库别名。

您指定的数据库名必须与在控制器配置文件中指定的名称相同。控制器检查这两个名称以确保您正使用正确的配置文件。如果用一个别名启动前端实用程序，而控制器配置文件包含不同的别名，则会报告一个错误，因为控制器不能确定这些名称是否是同一数据库的别名。

如果您在分区数据库环境中，当您在所有分区上启动控制器时，前端实用程序首先检查该配置文件是否未包含错误。然后它读取节点配置文件，并将一个带启动选项的命令发送至每个数据库分区，以便在每个数据库分区上启动控制器前端实用程序（从而在每个数据库分区上启动该精灵程序）。

注: 因为控制器是在数据库级进行监控，所以对于正在监控的每个数据库，都有一个精灵程序在运行。（在分区数据库环境中，对于每个数据库分区，都有一个精灵程序在运行。）如果控制器是对多个数据库运行的，则在该数据库服务器中将运行多个精灵程序。

nodenum **node-num**

指定要启动控制器精灵程序的数据库分区。该编号与节点配置文件中指定的编号相同。

当您在单个数据库分区上启动控制器时，前端实用程序创建一个精灵程序来验证控制器配置文件。控制器精灵程序确保没有其他精灵程序在该分区上运行。

config-file

指定当监控数据库时使用的配置文件。

该配置文件的缺省位置是 `sqllib` 目录。如果指定的文件不在该处，则前端假定指定的名称是该文件的全名。

log-file

指定控制器将向其中写入日志记录的文件的基本名称。该日志文件存储在 `sqllib` 目录的 `log` 子目录中。（在 Windows NT 上，`log` 子目录位于实

例目录下面。）自动将控制器运行所在的数据库分区的编号追加至日志文件名（例如，mylog.0、mylog.1、mylog.2）。

stop **database**

停止正监控指定数据库的控制器精灵程序。

如果您处于分区数据库环境中，则前端实用程序通过以下方法在所有数据库分区上停止控制器：读取节点配置文件，然后将一个带停止参数的命令发送至每个数据库分区，以调用控制器前端实用程序。这将在每个数据库分区上停止该精灵程序。

nodenum **node-num**

指定要停止控制器精灵程序的数据库分区。该编号与节点配置文件中指定的编号相同。

当前端实用程序在单个数据库分区上停止控制器精灵程序时，它通过在 sqllib 目录的 tmp 子目录中创建、移动或删除文件，来与数据库分区上的精灵程序通信。不要尝试删除或修改这些文件。

控制器精灵程序

当通过 db2gov 前端实用程序或通过唤醒来启动控制器精灵程序时，它将在一个循环中运行。它执行的第一个任务是检查其控制器配置文件是否已更改或是否还未读取。如果两个条件都是真，则该精灵程序读取该文件中的规则。这允许您在控制器精灵程序运行时更改其行为。

然后，控制器精灵程序发出一个快照请求，以获取在该数据库上运行的每个应用程序和代理程序的统计信息。

注： 在某些平台上，无法从 “DB2 监控程序” 获取 CPU 统计信息。在此情况下，帐户规则和 CPU 限制将不可用。

控制器然后根据控制器配置文件中的规则检查每个应用程序的统计信息。如果一条规则适用于某个应用程序，则根据该规则指定的操作控制器可以：强制该应用程序；更改该应用程序的优先级（这会间接更改在该节点上为该应用程序工作的代理程序和子代理程序的所有代理程序优先级）；或者，更改该应用程序的调度（这会间接更改在该应用程序上生效的代理程序优先级）。控制器将有关它执行的任何操作的记录写入一个日志文件。

注： 如果 *agentpri* 数据库管理程序配置参数不是系统缺省值，则不能将控制器用作调整代理程序优先级的替代方法。（此注释不适用于 OS/2 或 Windows NT 平台。）

当控制器检查完所有应用程序后，它会在配置文件中指定的时间间隔内休眠。一旦经过此时间，控制器醒来并再次开始执行循环。

当控制器遇到错误或停止信号时，它在结束之前会先执行清除处理。清除处理复位所有应用程序的代理程序优先级（使用已设置优先级的应用程序的列表）。然后它复位不再在应用程序上运行的任何代理程序的优先级。这确保在控制器结束后，代理程序不会一直以非缺省优先级运行。如果发生错误，将信息写入 db2diag.log 文件以指示控制器异常结束。

注: 控制器精灵程序不是数据库应用程序，因此，它不维护与数据库的连接。（然而，它确实有一个瞬时连接。） 控制器精灵程序可检测数据库管理程序何时结束，因为它可发出快照请求。

创建控制器配置文件

当启动控制器时，您要指定配置文件的名称，配置文件包含一些规则，可控制对数据库运行的应用程序。控制器根据这些规则进行操作。

如果控制数据库的要求改变，可编辑该配置文件而不用停止控制器。每个控制器精灵程序将检测该文件是否已更改，并重新读取该文件。

您必须在一个目录中创建该配置文件以安装在所有数据库节点上，因为每个节点上的控制器精灵程序只能读取同一配置文件。

该配置文件由规则和注解组成。可用大写、小写或大小写混合字符来指定大多数项。异常情况是 applname，它是区分大小写的。

将注解限制在 { } 大括号内。这些规则包括:

- 这些规则适用的数据库。
- 在唤醒以检查应用程序之前，控制器休眠的时间。
- 指定如何控制应用程序的那些规则。这些规则由称为规则子句的更小的部件组成。

该文件中的每个规则必须后接分号 (;)。

下列规则指定要监控的数据库，以及该精灵程序在完成一个活动周期的工作后唤醒的时间间隔（这在 第239页的『控制器精灵程序』 中有描述）。每一条规则仅在该文件中指定一次。

dbname

要监控的数据库的名称或别名。

account *nnn*

每隔指定的分钟数，对每个连接编写帐户记录，包含 CPU 使用统计信息。

注: 此选项在 Windows NT 或 OS/2 环境中不可用。

如果一个短期连接对话全部发生在该帐户时间间隔内，则不会编写任何日志记录。当编写日志记录时，它们包含 CPU 统计信息，这些信息反映自该连接的上一个日志记录以来 CPU 的使用情况。如果将控制器停止，然后重新启动，则在两个日志记录中都可能反映 CPU 的使用情况；这些信息可通过日志记录中的应用程序 ID 来标识。有关控制器日志文件的详情，参见第247页的『控制器日志文件』。

interval

精灵程序唤醒的时间间隔，以秒计。如果不指定时间间隔，则使用时间间隔 120 秒。

组合下列规则子句以构成一个规则（即，一条完整的规则后跟分号，而每个单独的子句不跟分号）。这些子句指定该规则适用的时间范围、对可使用的资源的限制，以及选项：特定的用户和应用程序，或者如果超过该规则中指定的限制，控制器要执行的任何操作。这些子句只能在一条规则中指定一次，但可在多条规则中指定。这些子句必须按显示的次序指定。在后面的说明中，[ ] 指示可选子句。

[desc] 指定该规则的文本说明。必须用单引号或双引号将该说明引起来。

[time] 指定对该规则求值的时间段。

必须以下列格式指定该时间段：`time hh:mm hh:mm`，例如，`time 8:00 18:00`。如果不指定此子句，该规则一天 24 小时都有效。

[authid]

指定该应用程序在执行时所使用的一个或多个授权 ID (authid)。必须用逗号 (,) 将多个 authid 分隔，例如，`authid gene,michael,james`。如果在规则中不出现此子句，则该规则适用于所有 authid。

[applname]

指定建立与数据库的连接的可执行程序（或对象文件）的名称。

必须用逗号 (,) 将多个应用程序名分隔，例如，`applname db2bp,betha,geneprog`。如果在规则中不出现此子句，则该规则适用于所有应用程序名。

注:

1. 应用程序名区分大小写。

2. 数据库管理程序将所有应用程序名截断为 20 个字符。应确保想控制的应用程序由其应用程序名的前 20 个字符唯一地标识；否则可能控制的不是要控制的应用程序。

   将在控制器配置文件中指定的应用程序名截断为 20 个字符，以匹配其内部表示。

setlimit

指定控制器要检查的一个或多个限制。这些限制只能是 -1 或大于 0 （例如，`cpu -1 locks 1000 rowssel 10000`）。必须至少指定这些限制 (cpu, locks, rowsread, uowtime) 中的一个，且该规则未指定的任何限制都不受该特定规则的限制。控制器可检查下列限制:

cpu *nnn*

指定一个应用程序可消耗的 CPU 秒数。如果您指定 -1，则控制器不限制应用程序的 CPU 使用。

注: 此选项在 Windows NT 或 OS/2 环境中不可用。

locks *nnn*

指定一个应用程序可持有的锁定数。如果您指定 -1，则控制器不限制应用程序持有的锁定数。

rowssel *nnn*

指定返回至应用程序的行数。此值将仅在协调程序节点上为非零值。如果您指定 -1，则控制器不限制可选择的行数。

uowtime *nnn*

指定从工作单元 (UOW) 第一次成为活动的时间开始可经过的秒数。如果指定 -1，则不限制经过时间。

注: 如果使用 sqlmon（数据库系统监控程序开关）API 来停用工作单元开关，这将影响控制器根据工作单元经过时间来控制应用程序的能力。控制器使用监控程序来收集有关该系统的信息。如果您在数据库管理程序配置文件中关闭了这些开关，则对于整个实例，它也会关闭，且控制器将不再接收此信息。

idle *nnn*

指定在执行指定的操作之前对一个连接允许的空闲秒数。如果指定 -1，则不限制连接的空闲时间。

rowsread *nnn*

指定一个应用程序可选择的行数。如果指定 -1，则不限制该应用程序可选择的行数。

注: 此限制与 rowssel 不同。不同点是，rowsread 是为了返回结果集而必须读取的行数的计数。读取的行数包括引擎对目录表的读取，当使用索引时这种读取可能减少。

[操作] 指定当超过一个或多个指定的限制时要采取的操作。您可以指定下列操作。

注: 如果超过限制且未指定操作子句，则控制器将把为该应用程序工作的代理程序的优先级降低 10。

priority *nnn*

指定为应用程序工作的代理程序的优先级更改。有效值在 −20 至 +20 之间。

要使此参数有效:

- 在基于 UNIX 的平台上，必须将 *agentpri* 数据库管理程序参数设置为缺省值；否则，它会替换该 priority 子句。
- 在 OS/2 和 Windows NT 上，*agentpri* 数据库管理程序参数可能会与 *priority* 操作一起使用。

force 指定强制为该应用程序服务的代理程序。（发出 FORCE APPLICATION 以终止该协调代理程序。）

schedule [class]

调度可改善执行应用程序的代理程序的优先级，目标是将平均应答时间减至最低，同时维护所有应用程序之间的公平性。

控制器通过为执行应用程序的代理程序设置优先级，并使用 DB2 内部查询编译程序产生的查询成本估计，来实施其调度。如果指定了类别选项，则仅在根据该规则选择的所有应用程序中进行调度。如果不指定此选项，则控制器使用一个或多个类别，并在每个类别中执行调度。

在每个类别中，如何设置应用程序的优先级基于:

- 该类别中的应用程序所持有的锁定数。（由于锁定而挂起许多其他应用程序的一个应用程序被给予高优先级。）
- 该应用程序的寿命。（长时间在系统中的应用程序被给予高优先级。）
- 该应用程序的估计剩余运行时间。（接近完成的应用程序被给予高优先级。）

没有被任何调度约束的应用程序以最高权限运行。

注: 如果使用了 sqlmon（数据库系统监控程序开关）API 来停用该语句开关，这将影响控制器根据该语句经过时间来控制应

用程序的能力。控制器使用监控程序来收集有关该系统的信息。如果您在数据库管理程序配置文件中关闭了这些开关，则对于整个实例，它也会关闭，且控制器将不再接收此信息。

调度操作可以:

- 确保不同组中的每个应用程序获取时间，而不用在所有应用程序之间均匀地划分时间。

  例如，如果 12 个应用程序（三个短的、五个中等长度和六个长的）同时运行，则它们的响应时间可能都很长，因为它们同时使用 CPU。数据库管理员可以设置两个组：中等长度的应用程序和长应用程序。使用优先级，控制器允许所有短的应用程序运行，并确保最多三个中等长度的应用程序和三个长应用程序同时运行。为此，控制器配置文件包含一条针对中等长度应用程序的规则，和一条针对长应用程序的规则。以下示例显示一个控制器配置文件的一部分，以举例说明这一点:

  ```
  desc "Group together medium applications in 1 schedule class"
  applname medq1, medq2, medq3, medq4, medq5
  setlimit cpu -1
  action schedule class;

  desc "Group together long applications in 1 schedule class"
  applname longq1, longq2, longq3, longq4, longq5, longq6
  setlimit cpu -1
  action schedule class;
  ```

- 确保多个用户组（例如，机构性部门）中的每一个都获取相等的优先级。

  如果一个组正在运行大量的应用程序，则管理员可以确保其他组仍能够获得合理的响应时间，以运行他们的应用程序。例如，对于涉及到三个部门（财务、库存和计划）的情况，所有“财务”用户可能会被置于一个组，所有“库存”用户可能会被置于第二个组，而所有“计划”用户可能会被置于第三个组。应将处理能力在这三个部门中较为均匀地划分。以下示例显示一个控制器配置文件的一部分，以举例说明这一点:

  ```
  desc "Group together Finance department users"
  authid tom, dick, harry, mo, larry, curly
  setlimit cpu -1
  action schedule class;

  desc "Group together Inventory department users"
  authid pat, chris, jack, jill
  setlimit cpu -1
  action schedule class;
  ```

```
desc "Group together Planning department users"
authid tara, dianne, henrietta, maureen, linda, candy
setlimit cpu -1
action schedule class;
```

- 允许控制器调度所有的应用程序。

  如果该操作中未包括 class 选项，则控制器会根据该调度操作下有多少个应用程序来创建它自己的类别，并根据 DB2 查询编译程序为该应用程序运行的查询所做的成本估计，来将这些应用程序置于不同的类中。管理员可以不限定选择哪些应用程序，以调度所有的应用程序。即，不提供 *applname* 或 *authid* 子句，*setlimit* 子句就不会产生任何限制。

注： 如果超过限制且未指定操作子句，则控制器降低执行该应用程序的代理程序的优先级。

如果有多个规则适用于某个应用程序，将最靠近该配置文件末尾的那个规则应用于该应用程序。如果为一条规则中的子句指定 -1，则发生异常。在这种情况下，为后续规则中的子句指定的值只能替换先前为*相同*子句指定的值：先前规则中的其他子句仍有效。例如，一个规则指示，如果一个应用程序的经过时间大于 1 小时，或者如果它选择多于 100 000 行（即 `rowssel 100000 uowtime 3600`），将降低该应用程序的优先级。后续规则指示同一应用程序可有不受限制的经过时间（即，`uowtime -1`）。在此情况下，如果该应用程序运行时间超过 1 小时，则不会更改其优先级（即 `uowtime -1` 替换 `uowtime 3600`），但是，如果它选择多于 100 000 行，则将降低其优先级（而 `rowssel 100000` 仍有效）。

第246页的图29显示一个配置文件示例。

```
{ Wake up once a second, the database name is ibmsampl
 do accounting every 30 minutes.  }
interval 1; dbname ibmsampl; account 30;

desc "CPU restrictions apply 24 hours a day to everyone"
setlimit cpu 600 rowssel 1000000 rowsread 5000000;

desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;

desc 'Slow down a subset of applications'
applname jointA, jointB, jointC, quryA
setlimit cpu 3 locks 1000 rowssel 500 rowsread 5000;

desc "Have governor prioritize these 6 long apps in 1 class"
applname longq1, longq2, longq3, longq4, longq5, longq6
setlimit cpu -1
action schedule class;

desc "Schedule all applications run by the planning dept"
authid planid1, planid2, planid3, planid4, planid5
setlimit cpu -1
action schedule;

desc "Schedule all CPU hogs in one class which will control consumption"
setlimit cpu 3600
action schedule class;

desc "Slow down the use of db2 CLP by the novice user"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "During day hours do not let anyone run for more than 10 seconds"
time 8:30 17:00 setlimit cpu 10 action force;

desc "Allow users doing performance tuning to run some of
     their applications during lunch hour"
time 12:00 13:00 authid ming, geoffrey, john, bill
applname tpcc1, tpcc2, tpcA, tpvG setlimit cpu 600 rowssel 120000 action force;

desc "Some people should not be limited -- database administrator
  and a few others.  As this is the last specification in the
  file, it will override what came before."
authid gene, hershel, janet setlimit cpu -1 locks -1 rowssel -1 uowtime -1;

desc "Increase the priority of an important application so it always
      completes quickly"
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;
```

图 29. 控制器配置文件示例

控制器日志文件

当控制器精灵程序强制应用程序、读取控制器配置文件、更改应用程序的优先级、遇到错误或警告、启动或结束时，它会将一条记录写入日志文件。对于每个控制器精灵程序，都存在一个单独的日志文件。这防止文件锁定瓶颈，此现象是由许多控制器精灵程序同时写入同一文件产生的。可使用 db2govlg 实用程序来将日志文件合并在一起，并查询它们。第248页的『查询控制器日志文件』中描述了此实用程序。

日志文件存储在 sqllib 目录的 log 子目录中。（在 Windows NT 上，log 子目录位于实例目录下面。）当发出 db2gov 命令时，提供该日志文件的基本名。应确保该日志文件名包含数据库名，因为对于受控制的每个数据库的每个节点，将都存在一个日志文件。在分区数据库环境中，控制器运行所在的数据库分区的节点号被自动追加至该日志文件名，以确保对于每个控制器，该文件名是唯一的。

日志文件中的每个记录具有下列格式:

```
Date Time NodeNum RecType Message
```

Date 和 *Time* 字段使用 志 yyyy-mm-dd-hh.mm.ss 格式，以便您可通过对此字段排序来合并每个数据库分区的日志文件。

NodeNum 字段指示控制器运行所在的数据库分区的编号。

RecType 字段包含不同的值，这取决于写入该日志的日志记录的类型。可记录的值有:

- START 指示启动了控制器
- FORCE 指示强制了一个应用程序
- PRIORITY 指示更改了一个应用程序的优先级
- ERROR 指示一个错误
- WARNING 指示一个警告
- READCFG 指示控制器读取了配置文件
- STOP 指示停止了控制器
- ACCOUNT 指示该应用程序的记帐统计信息。

  这些字段有:

  - authid
  - appl_id
  - written_usr_cpu
  - written_sys_cpu
  - appl_con_time

- SCHEDULE 指示代理优先级发生更改。

因为写入的是标准值，您可查询不同类型的操作的日志文件。*Message* 字段提供其他非标准信息，这些信息根据 *Rectype* 字段下的值的不同而不同。例如，FORCE 或 NICE 记录指示 *Message* 字段中的应用程序信息，而 ERROR 记录包括出错信息。

如下所示是一个日志文件示例:

```
1995-12-11 14.54.52    0 START      Database = TQTEST
1995-12-11 14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
1995-12-11 14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
1995-12-11 14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

查询控制器日志文件

每个控制器精灵程序都写入自己的日志文件中。可使用 db2govlg 实用程序来查询该日志文件。可按日期和时间排序，来列出单个分区或所有数据库分区的日志文件。也可根据 *RecType* 日志字段进行查询。db2govlg 的语法如下所示:

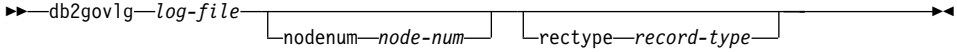


图 30. db2govlg 的语法

这些参数如下所示:

log-file
: 您想查询的一个或多个日志文件的基本名。

nodenum **node-num**
: 控制器运行所在的数据库分区的节点号。

rectype **record-type**
: 您想查询的记录的类型。这些记录类型有:
  - START
  - READCFG
  - STOP
  - FORCE
  - NICE
  - ERROR
  - WARNING
  - ACCOUNT

使用此实用程序不存在授权限制。这允许所有用户查询控制器是否影响了他们的应用程序。如果您想限制对此实用程序的存取，可更改 db2govlg 文件的组许可权。

运行控制器和数据库管理程序性能

因为控制器请求数据库管理程序的快照，因此它可影响数据库管理程序的性能。如果控制器使用太多的 CPU 资源，增加唤醒时间间隔以减少它对 CPU 的使用。

第10章 通过添加处理器调整配置

您可能发现配置的特性无法满足您的当前和计划的需要。结果是，您应考虑执行能够提高配置的容量和／或性能的操作。例如，对配置添加容器不仅能够提高存储数据的能力，还可以改进实用程序使用（如装入数据）期间的性能。提高容量或性能的其他方法包括：在对称多处理器或分区数据库环境中添加内存和添加处理器。

本章把注意力放在通过增大配置中的处理器数来改进性能方面。

如果出现以下情况，应按本章其余部分所述考虑调整配置:

- 您拥有具有单个处理器的单分区配置，该处理器已被最大限度地使用。因此，您已决定更改配置，并且:
  - 已确定“对称多处理器”(SMP) 配置是新环境下的最佳选择。您可能选择了此项，因为您想利用多个处理器可提供的强大处理能力。每个处理器共享内存和存储器系统资源。所有处理器在一个系统内，因此不存在附加的考虑事项（如系统间的通信线路），也可能没有附加的管理人员来支持任何新系统，并且系统间的任务协调不是一个简单的问题。“DB2 通用数据库”支持此环境。
  - 确定分区数据库配置是新环境下的最佳选择。您可能选择了此项，因为您想利用多个处理器（与第一个处理器实际分开）提供的强大处理能力。每个处理器具有自己的内存和存储器系统资源，而不必与其他处理器共享。当您可能有以上提到的附加考虑事项（通信、人员和任务协调）时，此选项将体现出其优点，如跨多个系统平衡数据及用户存取的能力。“DB2 通用数据库”支持此环境。
- 当前具有一个 SMP 配置，正计划添加一个或多个附加处理器。在此情况下，您已熟悉与此类型环境相关的那些考虑事项。添加一个或多个附加处理器后，您只是对环境添加了计算能力，但并没有增加新的考虑事项。“DB2 通用数据库”支持此环境。
- 具有一个分区数据库配置，正计划添加一个或多个附加数据库分区。在此情况下，您已熟悉与此类型环境相关的那些考虑事项。添加一个或多个附加数据库分区后，您只是对环境添加了计算能力，除了要转换为更大的分区数之外，并没有增加新的考虑事项。“DB2 通用数据库”支持此环境。

  分区数据库配置的一种变化是数据库分区所在位置变为 SMP 机器。“DB2 通用数据库”支持此环境。

当通过更改环境来调整系统时，应意识到这类更改可给数据库过程（如装入数据、备份数据库和复原数据库）带来的影响。

当添加一个新的数据库分区时，在该过程完成并将新服务器成功集成到系统中之前，不能卸下数据库或创建使用新分区的数据库。

向机器添加处理器

如果现存处理器在许多时候都能得到充分利用，则应考虑在机器中安装一个或多个附加处理器。要允许 DB2 数据库管理程序利用新的处理器，应复查、可能还要更新配置参数。（一些操作系统，如 Solaris，可动态地将处理器联机和脱机。）用来确定所用的处理器数目以及可能需要更新的参数包括:

- 第373页的『缺省度 (dft_degree)』
- 第394页的『最大查询并行度 (max_querydegree)』
- 第395页的『启用分区内并行性 (intra_parallel)』

还应考虑与可能需要更新的应用程序相关的参数。有关详情，参见第74页的『应用程序的并行处理』。

当在使用 TCP/IP 进行通信的环境中工作时，您应考虑 DB2TCPCONNMGRS 注册表变量的值。参见第417页的『附录A. DB2 注册表变量和环境变量』以了解有关此变量的详情。

向分区数据库系统添加数据库分区

可在分区数据库系统运行时或停止时，向其添加数据库分区。下列几节描述如何执行此任务。因为添加新服务器可能要花费时间，因此应在该数据库管理程序已经运行时来添加。在第253页的『向运行中的系统添加数据库分区』中描述了此过程。

ADD NODE 命令用来将数据库分区添加至系统。可调用此命令:

- 作为 `db2start` 上的一个选项
- 使用:
  - 命令行处理器 ADD NODE 命令
  - `sqleaddn`
  - `sqlepstart`。

调用该命令所用的方法取决于系统是停止的（使用 `db2start`）还是正在运行（使用任何其他选项）。

当使用 ADD NODE 命令将一个新数据库分区添加至系统时，在新数据库分区上会创建该实例中所有现存的数据库。也可指定临时表空间的哪些容器将被用于创建的数据库。这些容器可能有下列特征:

- 与为每个数据库的目录节点定义的那些容器相同。（这是缺省情况。）
- 与为另一个数据库分区定义的那些容器相同。
- 根本没有创建。在可使用数据库之前，必须使用 ALTER TABLESPACE 语句来将临时表空间容器添加至每个数据库。

在改变一个或多个节点组以包括新数据库分区之前，不能使用新分区上的数据库来包含数据。有关如何改变节点组的详情，参见第260页的『添加和卸下数据库分区』。

注: 如果系统中没有定义任何数据库且您正在基于 UNIX 的系统上运行 DB2 扩充企业版，则编辑 db2nodes.cfg 文件，以添加新的数据库分区定义；不要使用下列任何一个过程，因为它们只有在存在数据库时才适用。有关如何更新节点配置文件的详情，*参考管理指南: 计划*中的 “改变节点组”。

Windows NT 考虑事项: 如果在 Windows NT 上使用 DB2 扩充企业版，并且该实例没有数据库，则应使用 DB2NCRT 命令来调整该数据库系统。有关此命令的信息，参考 *Command Reference*。另一方面，如果您已有数据库，则应使用 DB2START ADDNODE 命令，因为这确保在调整系统时为每个现存的数据库创建一个数据库分区。有关 DB2START 命令和必须在 Windows NT 上使用的参数的信息，参考 *Command Reference*。在 Windows NT 上，始终都不应人工编辑节点配置文件 (db2nodes.cfg)，因为这会给该文件带来不一致性。

向运行中的系统添加数据库分区

当系统正在运行时且应用程序与数据库连接时，可将新数据库分区添加至分区数据库系统。然而，只有将数据库管理程序关闭并重新启动之后，新添加的服务器才能供所有数据库使用。

要将一个数据库分区添加至多服务器系统:

1. 如果要在系统中已存在的一个服务器上创建数据库分区，则转至下一个步骤。否则，执行下列操作:
   - 在 UNIX 平台上，
     a. 安装新服务器。这包括使可执行程序可供存取（使用共享文件系统安装或本地副本），使操作系统文件与现存处理器上的那些文件同步，确保 sqllib 目录作为共享文件系统是可存取的，确保将相关的操作系统参数（如最大进程数）设置为适当的值。

b. 向名称服务器注册该主机名，或者在所有数据库分区的 etc 目录中的 hosts 文件中注册该主机名。

- 在 Windows NT 平台上，
  a. 安装新服务器。
  b. 在新服务器上运行 ADD NODE 命令。此命令导致为已存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。
  c. 转至第三 (3) 点。

2. 在任何数据库分区上运行 DB2START 命令，并指定 NODENUM、ADDNODE、HOSTNAME、PORT 和 NETNAME 参数。在 Windows NT 平台上，还必须指定 COMPUTER、USER 和 PASSWORD 参数。有关 DB2START 命令的详情，参考 *Command Reference*。

   您还可选择为需要在该数据库中创建的任何“临时表空间容器定义”指定源。如果未提供表空间信息，则从每个数据库的目录节点检索该临时表空间容器的定义。

   当该命令完成时，停止新服务器。在执行 DB2STOP 之前，不使用新服务器信息更新节点配置文件。这确保 ADD NODE 命令（在指定 ADDNODE 参数时调用的命令）在正确的数据库分区上运行。当该实用程序结束时，停止该新服务器。

3. 通过运行 DB2STOP 命令来停止数据库管理程序。

   当停止系统中的所有数据库分区时，会更新节点配置文件以包括新的数据库分区。

4. 通过运行 DB2START 命令来启动数据库管理程序。

   现在，新添加的数据库分区与系统的其余部分一起启动。

   当系统中的所有数据库分区正运行时，可执行系统范围内的活动，如创建或卸下数据库。

   注: 可能必须对所有数据库分区服务器发出 DB2START 命令两次，以存取新的 db2nodes.cfg 文件。

5. 可选择对新数据库分区上的所有数据库进行备份。
6. 可选择将数据重新分布至该新数据库分区。有关详情，参见第259页的『第11章 将数据重新分布至数据库分区』。

向停止的系统添加数据库分区

当分区数据库系统停止时，就可以将新数据库分区添加至其中。当再次启动该数据库管理程序时，新添加的服务器可用于所有数据库。有两种选择。可让数据库管理程序更新节点配置文件，或人工更新它。两个过程的准备步骤是相同的。

注: 当在 Windows NT 上工作时，不应人工更新该节点配置文件。而应该使用数据库管理程序来更新此文件（如下所述）。

要将新数据库分区添加至多服务器系统:

1. 发出 DB2STOP 停止所有数据库分区。
2. 如果在系统中已存在的一个处理器上创建该服务器，则转至下一个步骤。否则，执行下列操作:
   a. 在 UNIX 平台上,
      1) 安装新服务器。这包括使可执行程序可供存取（使用共享文件系统安装或本地副本），使操作系统文件与现存处理器上的那些文件同步，确保 `sqllib` 目录作为共享文件系统是可存取的，确保将相关的操作系统参数（如最大进程数）设置为适当的值。
      2) 向名称服务器注册该主机名，或者在所有数据库分区的 `etc` 目录中的 `hosts` 文件中注册该主机名。
   b. 在 Windows NT 平台上,
      1) 安装新服务器。
      2) 在新服务器上运行 ADD NODE 命令。此命令导致为已存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。
      3) 运行 DB2START 命令启动数据库系统。注意，在新服务器的安装期间已更新了节点配置文件 (`db2nodes.cfg`)，以包括该新的服务器。
      4) 可选择将数据重新分布到新的服务器。有关如何完成此操作的详情，参见第259页的『第11章 将数据重新分布至数据库分区』。
   c. 如果希望该数据库管理程序更新 `db2nodes.cfg` 文件，则继续遵循第256页的『让数据库管理程序更新节点配置文件』中的说明。

      注: 在 Windows NT 上，不应人工编辑 `db2nodes.cfg` 文件，因为这会给该文件带来不一致性。而应让数据库管理程序更新此文件。

      如果想自己更新 `db2nodes.cfg` 文件，则继续遵循第256页的『人工更新节点配置文件』中的说明。

让数据库管理程序更新节点配置文件

在向分区数据库系统添加一个或多个新数据库分区之后，要使新分区可用，您必须更新 db2nodes.cfg 文件。在上面的 『c』 点中，您决定让数据库管理程序更新节点配置文件。本节介绍了这是如何完成的。

注: 在上面的 『c』 点中，如果您决定人工更新节点配置文件，则您应跳过本节并转至下一节。

继续该过程，如下所示:

1. 在新数据库分区上运行 DB2START 命令，并指定 NODENUM、 ADDNODE、HOSTNAME、PORT 和 NETNAME 参数。在 Windows NT 平台上，还必须指定 COMPUTER、USER 和 PASSWORD 参数。有关 DB2START 命令的详情，参考 *Command Reference*。为这些参数指定的值被用于更新节点配置文件。

   当该命令完成时，停止新服务器。在执行 DB2STOP 之前，不使用新服务器信息更新节点配置文件。这确保 ADD NODE 命令（在指定 ADDNODE 参数时调用的命令）在正确的数据库分区上运行。当该实用程序结束时，停止该新服务器。
2. 发出 DB2STOP 命令。

   当发出 DB2STOP 命令时，更新节点配置文件以包括该新服务器。
3. 发出 DB2START 命令以启动该数据库系统。

   注: 可能必须对所有数据库分区服务器发出 DB2START 命令两次，以存取新的节点配置文件。
4. 可选择对新数据库分区上的所有数据库进行备份。
5. 可选择将数据重新分布至该新服务器。有关详情，参见第259页的『第11章 将数据重新分布至数据库分区』。

人工更新节点配置文件

在向分区数据库系统添加一个或多个新数据库分区之后，要使新分区可用，您必须更新 db2nodes.cfg 文件。在前一节之前的 『c』 点中，您决定人工更新节点配置文件。（记住，当在 Windows NT 上工作时，您不应人工更新节点配置文件。）本节介绍了如何人工更新节点配置文件。

注: 在前一节之前的 『c』 点中，如果您决定让数据库管理程序更新节点配置文件，则您应返回到前一节。

继续该过程，如下所示:

1. 编辑 db2nodes.cfg 文件，并将新数据库分区添加至该文件。
2. 发出下列命令以启动新节点:

```
DB2START NODENUM nodenum
```

将您赋予该新数据库分区服务器的号码指定为 *nodenum* 的值。

3. 如果该新服务器是一个逻辑数据库分区（即，它不是节点 0），则使用 **db2set** 命令来更新 DB2NODE 注册表值，并指定您将添加的服务器号。
4. 在新服务器上运行 ADD NODE 命令。

   此命令还导致为已存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。
5. 当 ADD NODE 命令完成时，发出 DB2START 命令启动系统中的其他数据库分区。

   在成功启动所有数据库分区之前，不应尝试执行任何系统范围内的活动，如创建或卸下数据库。
6. 可选择对新服务器上的所有数据库分区进行备份。
7. 可选择将数据重新分布至该新数据库分区。有关详情，参见第259页的『第11章 将数据重新分布至数据库分区』。

从系统卸下数据库分区

可使用带 DROP NODENUM 参数的 DB2STOP 命令或 sqlepstp API 来卸下数据库分区。在进行此工作之前，必须首先确保要卸下的数据库分区未被任何数据库使用。要检查是否在使用，发出 DROP NODE VERIFY 命令。

应确保所有事务（此数据库分区是其协调程序）已成功确认或回滚。这可能需要在其他服务器上执行应急恢复。

例如，如果卸下协调程序数据库分区（即，协调程序节点），且参与事务的另一个数据库分区在卸下协调程序节点之前崩溃，则该崩溃的数据库分区将不能查询协调程序节点来获取任何不确定事务的输出。

要从分区数据库系统中卸下数据库分区:

1. 重新分布驻留在此节点上的每个数据库的数据。这能满足一个需求，即要卸下的数据库分区未被任何数据库使用。有关详情，参见第259页的『第11章 将数据重新分布至数据库分区』。
2. 发出 DROP NODE VERIFY 命令或 sqledrpn API 以验证该服务器未在使用中。

根据接收到的信息，继续步骤 3 或步骤 4。

3. 如果接收到信息 SQL6034W（没有数据库在使用该节点），可执行下列操作:

a. 发出带 DROP NODENUM 参数的 DB2STOP 命令卸下该数据库分区。在命令成功完成后，停止系统。

b. 可根据需要使用 DB2START 命令启动数据库管理程序。

4. 如果接收到信息 SQL6035W（数据库在使用该节点），则执行下列操作:

a. 使用 REDISTRIBUTE NODEGROUP 命令，将正卸下的数据库分区中的数据重新分布至使用该数据库别名的其他数据库分区，如信息 SQL6035W 中所指示的。在完成此操作之前，不能卸下该数据库分区。

b. 卸下在该数据库分区上定义的任何事件监控程序。

c. 返回至步骤第257页的2，然后继续。

第11章 将数据重新分布至数据库分区

仅当在一个分区数据库环境中工作时，才需要考虑数据重新分布。如果在单分区数据库环境中，则不需要参考本章中的信息。

使用“数据重新分布”实用程序在一个现存的节点组中的各数据库分区之间移动数据。可使用它来执行下列操作:

- 平衡数据库分区之间的数据量和处理负荷。

  如果您要定期存取某个数据库表中的所有数据，则该功能很有用。
- 将数据不均匀地分布在各数据库分区中。

  如果您仅需定期存取某个数据库表中的一部分数据，则该功能很有用。在这种情况下，可重新分布该表，以使不经常存取的数据位于节点组中的少量数据库分区中，而频繁存取的数据分布在更多的分区中。这将改进最频繁运行的应用程序的存取性能和处理能力。

使用 REDISTRIBUTE NODEGROUP 命令来调用“数据重新分布”实用程序。参考 *Command Reference* 以了解有关此命令的语法的详情。

要保留表并置，将此操作应用于节点组中的所有表，并在节点组级而不是在表级执行重新分布。

要实现您期望的数据分布，该实用程序使用分区映象在节点组的各数据库分区之间移动表的行。根据您指定的选项，该实用程序可生成目标分区映象，也可以将一个现存的分区映象用作输入。

注:

1. 应根据您认为的“数据重新分布”操作将需要的日志空间要求来指定日志文件大小。还应确保该日志足够大，以适应在重新分布数据所处的每个数据库分区上执行的 INSERT 和 DELETE 操作。
2. 如果要在包含复制的摘要表的节点组中重新分布该数据，必须首先卸下这些表、重新分布该节点组、然后重新创建这些表。不能重新分布包含复制的摘要表的节点组。

如何将数据分区

缺省情况下，“数据重新分布”实用程序假定相同数目的行散列至每个散列分区，因此它均匀地将散列分区分布至节点组中的所有数据库分区上。如果不是相同数目的行散列至每个散列分区，可使用*分布文件*来指定当前分布。此文件对每个 4 096 散列分区都包含一个值。每个值用作相应散列分区的*加权因子*。“数据重新分布”实用程序生成一个目标分区映象，其中的所有数据库分区都有大约相同的加权因子。因此，即使数据分布是不对称的，也可使用该分布文件来实现均匀的数据分布。

可使用自动装入实用程序和 ANALYZE 选项来创建数据分布文件。可将此文件用作“数据重新分布”实用程序的输入。参考 *Data Movement Utilities Guide and Reference* 以了解有关自动装入 实用程序的详情。

或者，可使用 PARTITION 和 NODENUMBER SQL 函数来确定散列分区或数据库分区中的当前数据分布。（使用 PARTITION 函数来确定在散列分区上的分布。）可使用此信息来派生一个分布文件和一个目标分区映象。

例如，要了解哪些数据库分区（如果有的话）因为数据分布不统一而通常带有相当大量的行:

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
   GROUP BY PARTITION(column_name)
   ORDER BY PARTITION(column_name) DESC
   FETCH FIRST 100 ROWS ONLY
```

应确保 table_name 是最大的表，而 column_name 是该表中适当的列。

添加和卸下数据库分区

可使用 ALTER NODEGROUP 语句在节点组中添加或卸下数据库分区。当添加数据库分区时，该分区必须已在节点配置文件中定义。

在使用 ALTER NODEGROUP 语句之后，将创建一个新的分区映象。当使用“数据重新分布”实用程序时，这个新的分区映象可成为目标分区映象。（创建目标分区映象的另一个方法是自己创建它。）

如果使用带 WITHOUT TABLESPACES 子句的 ALTER NODEGROUP 语句，则必须在重新分布数据前将表空间容器添加到一个或多个新的数据库分区上。有关 ALTER NODEGROUP 语句的其他信息，参考 *SQL Reference*。

指定目标分区映象

“数据重新分布”实用程序使用分区映象来执行数据重新分布。它可以创建自己的目标分区映象，或者您可提供一个给该实用程序使用。如果创建了一个，则项目数将确定由于数据重新分布而产生的节点组的类型:

- 1 项表示单分区节点组
- 4 096 项表示多分区节点组

如果目标分区映象有多个数据库分区，则必须在节点组的所有表中定义同一个分区关键字。

目标分区映象只能包含在 SYSCAT.NODEGROUPDEF 目录表中定义的数据库分区号，不包含 IN_USE 值为 'T' 的那些分区。（'T' 表示该分区不在目标分区映象中。）对于 IN_USE 值为 'D'（表示卸下）且未出现在目标分区映象中的所有数据库分区，当重新分布操作成功完成时要卸下它们。

如何将数据重新分布到数据库分区中

“数据重新分布”操作在数据库的指定节点组中的一组表上执行。（在执行该操作前，该应用程序必须在目录数据库分区与数据库连接。）该实用程序使用源分区映象和目标分区映象来标识已将哪些散列分区分配到了新的位置（即，新的数据库分区号）。与具有新位置的一个分区对应的所有行从源分区映象中指定的数据库分区移动到目标分区映象中指定的数据库分区。

“数据重新分布”实用程序执行下列操作:

1. 获取目标分区映象的新分区映象 ID，并将它插入 SYSCAT.PARTITIONMAPS 目录视图。
2. 使用新的分区映象 ID 更新该节点组的 SYSCAT.NODEGROUPS 目录视图中的 REBALANCE_PMAP_ID 列。
3. 将任何新的数据库分区添加至 SYSCAT.NODEGROUPS 目录视图。
4. 对要卸下的任何数据库分区，将 SYSCAT.NODEGROUPS 目录视图中的 IN_USE 列设置为 'D'。
5. 执行 COMMIT 以更新目录。
6. 为所有新的数据库分区创建数据库文件。
7. 对于节点组中的每个表，重新分布每个表中的数据。这在第262页的『如何重新分布表中的数据』中有描述。
8. 对于先前标记为要卸下的数据库分区，删除数据库文件，并删除 SYSCAT.NODEGROUPS 目录视图中的对应项。

9. 更新 SYSCAT.NODEGROUPS 目录视图中的节点组记录，以将 PMAP_ID 设置为 REBALANCE_PMAP_ID 的值，并将 REBALANCE_PMAP_ID 设置为空值。
10. 从 SYSCAT.NODEGROUPS 目录视图中删除旧的分区映象。
11. 对所有更改执行 COMMIT。

如何重新分布表中的数据

当对一个表执行数据重新分布时，该实用程序执行下列操作:

1. 在 SYSTABLES 目录表中锁定关于该表的那一行。
2. 使涉及此表的所有程序包失效。因为要重新分布该表，因此与该表相关的分区映象 ID 将更改。因为程序包失效，编译程序必须获得该表的新的分区信息，并相应地生成程序包。
3. 以独占方式锁定该表。
4. 通过 DELETE 和 INSERT，重新分布表中的数据。
5. 如果重新分布操作成功，则该实用程序:
   a. 对该表发出 COMMIT。
   b. 继续处理节点组中的下一个表。

   如果在表被全部重新分布之前该操作失败，则该实用程序:
   a. 对该表的更新发出 ROLLBACK。
   b. 结束整个重新分布操作，并返回错误。

估计分布数据时的日志空间要求非常重要。日志的大小必须足以容纳重新分布数据的每个数据库分区上的 INSERT 和 DELETE 操作。最高的记录需求将是将丢失最多数据的数据库分区，或者是将获得最多数据的数据库分区。如果正在移至大量数据库分区，则当前数据库分区数与新数据库分区数的比率将有助于您确定 INSERT 和 DELETE 操作的数目。

例如，如果是从 4 个数据库分区移至 5 个数据库分区，则将把四个原始数据库分区的大约 20% 的数据移至新数据库分区。这意味着四个原始数据库分区中的每一个都将执行该数据库分区上数据总量的百分之二十的 DELETE 操作。新数据库分区将执行所有 INSERT 操作（即，与全部四个原始数据库分区的 DELETE 操作数相等）。

上例假设的是统一分布数据。也会有不统一分布数据的情况，比如分区关键字中有大量 NULL 值存在时便是如此。在这种情况下，所有这些行在旧分区方案下将

前往一个数据库分区，在新分区方案下，将前往另一数据库分区。结果是，这可能会增加那两个数据库分区上需要的日志空间量，并且很可能会大大超过假设统一分布时计算得出的数量。

执行实际的计算时，必须将最大的表的大小乘以更改百分比（如 20%）。这样做的原因是每个表的重新分布都是作为单一事务完成的。

注: 然而，最大的表可能是统一分布的，但次大的表（只是举个例子）可能带有一个或多个膨胀的数据库分区。在这样的情况下，您应考虑使用第二个表，而不是最大的表。

在计算得出要在一个数据库分区上插入或删除的最大数据量之后，将该数字加倍，以确定活动日志大小的峰值。如果它超过活动日志限制（即 32 GB），则数据重新分布必须分步骤完成。可使用一个名为 『makepmap』 的实用程序来生成一系列目标分区映射，即为每个步骤生成一个。

校正重新分布错误

当重新分布操作开始执行后，要将一个文件写入 sqllib 目录的 redist 子目录中。此状态文件列出在数据库分区上执行的任何操作、重新分布的表的名称以及该操作的完成状态。如果不能重新分布一个表，则在该文件中列出其名称和适当的 SQLCODE。如果由于不正确的输入参数而使重新分布操作不能开始，则不写入该文件，而返回一个 SQLCODE。

此文件有下列命名约定:

databasename.nodegroupname.timestamp （对于 UNIX 平台）
databasename\nodegroupname\date\time （对于非 UNIX 平台）

注: 在非 UNIX 平台上，只使用 nodegroupname 的前八 (8) 个字节。

如果数据重新分布操作失败，则某些表可能要重新分布，而其他表不用。这是因为数据重新分布是以一次一个表的形式执行的。有两个恢复选项:

- 使用 CONTINUE 选项继续该操作，重新分布其余的表。
- 使用 ROLLBACK 选项来撤消该重新分布，并将已重新分布的表复位为它们的原始状态。回滚操作所花的时间大致与原始重新分布操作所花的时间相同。

在可以使用任何一个选项前，先前的数据重新分布操作必须已经失败，这样才能把 SYSNODEGROUPS 目录表中的 REBALANCE_PMID 列设置为空值以外的值。

如果碰巧错误地删除了此状态文件，仍可尝试 CONTINUE 操作。

数据重新分布和其他操作

当该实用程序正在运行时，您可对节点组的对象执行下列操作。但是，不能在要重新分布的表上执行。您可以:

- 在其他表上创建索引。CREATE INDEX 语句使用受影响的表的分区映象。
- 卸下其他表。DROP TABLE 语句使用受影响的表的分区映象。
- 卸下其他表上的索引。DROP INDEX 语句使用受影响的表的分区映象。
- 查询其他表。
- 更新其他表。
- 在节点组中定义的表空间中创建新表。CREATE TABLE 语句使用目标分区映象。
- 在节点组中创建表空间。

当该实用程序正在运行时，不能执行下列操作:

- 节点组上的另一个重新分布操作
- 对节点组中任何一个表的 ALTER TABLE 操作
- 卸下节点组
- 改变节点组。

在数据重新分布之后

当在一个节点组中重新分布数据之后，强烈建议您执行 RUNSTATS 来更新与可能已重新分布的表相关的统计信息。

有关 RUNSTATS 命令的详情，参考 *Command Reference* 手册。

第12章 基准测试

基准测试是应用程序开发有效期的一个常规部分。它是由应用程序开发人员和数据库管理员 (DBA) 等小组成员参与的工作，并应对您的应用程序实施，以确定和提高性能。假定应用程序代码编写得很有效率，那么要想再改善性能，可调整数据库和数据库管理程序配置参数甚至应用程序参数以满足应用程序的要求。

有几种不同类型的基准测试。*每秒事务数*基准将在特定的有限的实验室条件下确定数据库管理程序的吞吐量能力。*应用程序*基准将测试相同的吞吐量能力，但它的测试条件更接近于您的应用程序实施时所处的条件。基准测试的目的是调整配置参数，它基于这些“实际环境”条件，并使用各种参数值反复运行从您的应用程序中提取的 SQL，直到您的应用程序运行得尽可能有效率为止。

本节中描述的基准测试方法是针对配置参数的。但是，该基本技术同样可以用于调整影响性能的其他因素，如:

- SQL 语句
- 索引
- 表空间配置
- 应用程序代码
- 硬件配置。

基准测试有助于了解数据库管理程序在各种条件下是如何应答的。可创建多个方案，以测试死锁处理、实用程序性能、装入数据的不同方法、当添加更多的用户时事务执行速率的特征，甚至还可测试使用该产品的新发行版对应用程序产生的影响。

提供下列主题:

- 『基准测试方法』
- 第266页的『准备基准测试』
- 第268页的『创建基准测试程序』
- 第273页的『执行基准测试』。

基准测试方法

此基准测试技术以科学的方法为基础。将创建一个可重复的环境，在这个环境中，在相同的条件下运行相同的测试，以得到可比较的结果。

也可在一个正常的环境中运行测试应用程序，来开始基准测试。随着性能问题的减少，可以编写专用的测试实例，以限制要测试和观察的功能的范围。该专用测试实例不需要仿真整个应用程序，这样可获取有价值的信息。从简单的测量开始，仅当有保证时，才增加复杂程度。

优秀的基准测试（或测量）特征包括:
- 每个测试都是可重复的。
- 测试的每次重复都是在相同系统状态下开始的。
- 系统中除了正在测量的那些功能或应用程序外，没有其他功能或应用程序是活动的（除非该方案包括在系统中执行一定量的其他活动)。

  注: 启动的应用程序即使是在最小化或闲置时也会占用内存。这样分页就更有可能使基准测试程序的运行结果产生偏差和违反可重复性规则。
- 用于基准测试的硬件和软件要与您的生产环境匹配。

正如任何基准测试一样，必须设计一个方案，然后执行它。下列信息将这些概念应用于 DB2 环境。
- 『准备基准测试』
- 第268页的『创建基准测试程序』
- 第273页的『执行基准测试』。

准备基准测试

在开始性能基准测试之前，应该完成您的应用程序数据库的逻辑设计。需要设置和填写表、视图和索引。表应该是规范化的，而应用程序程序包应已联编，而且表应用真实的数据填写。

您应该已经确定了数据库的最终物理设计。应将数据库管理程序对象置于它们最终的磁盘位置，调整日志文件的大小，确定工作文件和备份的位置，并测试备份过程。此外，应该检查程序包，以确保在可能时启用性能选项，如行分块。

应在应用程序的程序设计和测试阶段达到一个标准，以允许创建基准测试程序（参见下一节）。在基准测试期间，一个应用程序的实际限制可能会暴露出来；但是，此处描述的基准测试程序的用途是测量性能，而不是检测故障或异常终止。

基准测试程序需要在尽可能接近于最终生产环境的条件下运行；理想情况是，在有相同内存和磁盘配置的同一种型号的服务器上运行。当该应用程序最终将涉及大量用户和大量数据时，这一点尤其重要。操作系统本身和基准测试程序直接使用的任何通信或文件服务设施也应已调整好。

使用具有产品大小的数据库进行基准测试也很重要。个别 SQL 语句返回的数据相当于它在生产环境中实施时将返回的那么多数据，并参与在该生产环境中实施时将执行的那么多排序。坚持此规则将确保应用程序产生典型的内存需求。

要进行基准测试的 SQL 语句的类型应该是*典型的*或*恶劣的*，如下所述:

典型的 SQL

典型的 SQL 包括在对应用程序的典型操作进行基准测试期间执行的那些语句。选择的语句将取决于应用程序的性质。例如，数据输入应用程序可能要测试 INSERT 语句，而银行业务事务可能会测试 FETCH、UPDATE 和多个 INSERT。选择的语句执行的频率和处理的数据量应被视为平均值。如果该数据量过大，则即使是典型的 SQL 语句，也应将这些语句归为*恶劣*类别考虑。

恶劣的 SQL

归入此类别的语句包括:

- 频繁执行的语句。
- 处理高容量数据的语句。
- 与时间密切相关的语句。

  例如，当接收到客户的电话时运行的应用程序，而当客户等待时，必须运行语句以检索和更新客户信息。
- 要连接最大数量的表的语句，或含有应用程序中最复杂的 SQL 的语句。

  例如，银行业务应用程序，它针对所有不同类型的帐户生成每月活动的综合客户报表。一个公用表可能会列出客户地址和帐号；但是，还必须合并其他几个表，以处理和集成所有需要的帐户事务信息。处理一个帐户所需的工作量乘以同一个周期内必须处理的几千个帐户，对于这种情况，可能节省的时间将迫使提高性能要求。
- 有不良存取路径的语句，如不经常执行的且为涉及的表创建的索引不支持的语句。
- 耗时长的语句。
- 只在应用程序初始化时才执行但有不适当的资源需求的语句。

  例如，生成当天必须处理的帐户工作列表的应用程序。当启动该应用程序时，第一个主 SQL 语句产生 7 路合并，这为此应用程序用户负责的所有帐户创建了一个非常大的列表。该语句可能每天只运行几次，但是如果未将它调整好就会耗费数分钟来运行。

创建基准测试程序

当设计和实施一个基准测试程序时，有多种因素要考虑。由于该程序的主要目的是模拟用户应用程序，所以该程序的总体结构可以改变。可以将整个应用程序用作基准测试程序，然后只需使用某种方法来计算要分析的 SQL 语句的时间。对于大的或复杂的应用程序，它可能更实用，可正好包括包含重要语句的块。

要测试特定 SQL 语句的性能，另一种方法是将这些语句以及需要的 CONNECT、PREPARE、OPEN 和其他语句一起单独包括在该基准测试程序中，并引入最佳值机制。

另一个要考虑的因素是要使用的基准测试程序的类型。一个选项是在一个时间间隔内重复运行一组 SQL 语句。执行的语句数量与此时间间隔的比率就是该应用程序的吞吐量。另一个选项是，只需确定执行个别 SQL 语句所需的时间。

无论是什么类型的基准测试程序，都需要一个高效率的最佳值系统，来计算个别 SQL 语句或整个应用程序的运行经过时间。要模拟个别 SQL 语句单独执行所在的应用程序，重要的一点是要考虑 CONNECT、PREPARE 和 COMMIT 语句的时间。但是，对于处理多个不同语句的程序，或许只有单个 CONNECT 或 COMMIT 是需要的，所以可能会优先测试个别语句的执行时间。

当每个查询的经过时间是性能分析中的一个重要因素时，可能就不必暴露瓶颈。例如，有关 CPU 的使用情况、锁定和缓冲池 I/O 的信息可能显示该应用程序达到 I/O 极限，而不是 CPU 的使用达到满负荷。基准测试程序应该允许您获取此类数据，以便在需要时进行更详细的分析。

并非所有的应用程序都需要将从查询检索到的整组行发送至某个输出设备。例如，某些应用程序可能使用整个回答集作为另一个程序的输入（即，不将任何行发送至输出）。格式化屏幕输出的数据常常产生很高的 CPU 成本，且可能无法反映用户需要。为提供准确的模拟，基准测试程序应该反映特定应用程序的行处理。如果确实将行发送至输出设备，则效率不高的格式化可能消耗大量的 CPU 处理时间，并会误报 SQL 语句本身的实际性能。

db2batch 基准测试工具：在您的实例 sqllib 目录的 bin 子目录中提供了一个基准测试工具 (db2batch)。此工具将上面提到的关于创建基准测试程序的许多要点都纳入考虑事项。此工具将从文本文件或标准输入读取 SQL 语句，动态地描述和准备这些语句，并返回回答集。它还提供附加的灵活性，允许控制回答集的大小，以及应从此回答集发送至输出设备的行数。

还可以指定提供的与性能相关的信息的级别，包括经过时间、CPU 和缓冲池的使用情况、锁定和从数据库监控程序收集的其他统计信息。如果正在计算一组 SQL 语

句的运行时间，则 db2batch 也将汇总性能结果，并提供算术和几何平均数。有关调用语法和选项的详情，可在命令行输入 db2batch -h。

有关 db2batch 的详情，也可以参考 *Command Reference* 手册。

以下是如何对一个输入文件 db2batch.sql 使用 db2batch 的示例:

```
-- db2batch.sql
-- ------------
--#SET PERF_DETAIL 3 ROWS_OUT 5

-- This query lists employees, the name of their department
-- and the number of activities to which they are assigned for
-- employees who are assigned to more than one activity less than
-- full-time.
--#COMMENT Query 1
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2;
--#SET PERF_DETAIL 1 ROWS_OUT 5
--#COMMENT Query 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2;
```

图 31. 基准测试程序输入文件样本: db2batch.sql

使用基准测试程序工具进行以下调用:

```
db2batch -d sample -f db2batch.sql
```

产生下面的输出:

```
--#SET PERF_DETAIL 3 ROWS_OUT 5
Query 1

Statement number: 1

select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) > 2
```

图 32. db2batch 的样本输出（第一部分）

```
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
-------------------------------------------------------------------------
JEFFERSON        JAMES         ADMINISTRATION SYSTEMS                3
JOHNSON          SYBIL         ADMINISTRATION SYSTEMS                4
NICHOLLS         HEATHER       INFORMATION CENTER                    4
PEREZ            MARIA         ADMINISTRATION SYSTEMS                4
SMITH            DANIEL        ADMINISTRATION SYSTEMS                7
Number of rows retrieved is:        5
Number of rows sent to output is:   5
Elapsed Time is:           0.074      seconds
Locks held currently                        = 0
Lock escalations                            = 0
Total sorts                                 = 5
Total sort time (ms)                        = 0
Sort overflows                              = 0
Buffer pool data logical reads              = 13
Buffer pool data physical reads             = 5
Buffer pool data writes                     = 0
Buffer pool index logical reads             = 3
Buffer pool index physical reads            = 0
Buffer pool index writes                    = 0
Total buffer pool read time (ms)            = 23
Total buffer pool write time (ms)           = 0
Asynchronous pool data page reads           = 0
Asynchronous pool data page writes          = 0
Asynchronous pool index page reads          = 0
Asynchronous pool index page writes         = 0
Total elapsed asynchronous read time        = 0
Total elapsed asynchronous write time       = 0
Asynchronous read requests                  = 0
LSN Gap cleaner triggers                    = 0
Dirty page steal cleaner triggers           = 0
Dirty page threshold cleaner triggers       = 0
Direct reads                                = 8
Direct writes                               = 0
Direct read requests                        = 4
Direct write requests                       = 0
Direct read elapsed time (ms)               = 0
Direct write elapsed time (ms)              = 0
Rows selected                               = 5
Log pages read                              = 0
Log pages written                           = 0
Catalog cache lookups                       = 3
Catalog cache inserts                       = 3
Buffer pool data pages copied to ext storage    = 0
Buffer pool index pages copied to ext storage   = 0
Buffer pool data pages copied from ext storage  = 0
Buffer pool index pages copied from ext storage = 0
Total Agent CPU Time (seconds)              = 0.02
Post threshold sorts                        = 0
Piped sorts requested                       = 5
Piped sorts accepted                        = 5
```

图 33. db2batch 的样本输出（第一部分）

```
--#SET PERF_DETAIL 1 ROWS_OUT 5
Query 2
Statement number: 2
select lastname, firstnme,
deptname, count(*) as num_act
from employee, department, emp_act
where employee.workdept = department.deptno and
employee.empno = emp_act.empno and
emp_act.emptime < 1
group by lastname, firstnme, deptname
having count(*) <= 2
LASTNAME         FIRSTNME      DEPTNAME                        NUM_ACT
------------------------------------------------------------------------
GEYER            JOHN          SUPPORT SERVICES                      2
GOUNOT           JASON         SOFTWARE SUPPORT                      2
HAAS             CHRISTINE     SPIFFY COMPUTER SERVICE DIV.          2
JONES            WILLIAM       MANUFACTURING SYSTEMS                 2
KWAN             SALLY         INFORMATION CENTER                    2
Number of rows retrieved is:        8
Number of rows sent to output is:   5
Elapsed Time is:           0.037      seconds
Summary of Results
==================
                Elapsed             Agent CPU         Rows      Rows
Statement #     Time (s)            Time (s)          Fetched   Printed
1                     0.074               0.020         5         5
2                     0.037       Not Collected         8         5
Arith. mean     0.055
Geom.  mean     0.052
```

图 34. db2batch 的样本输出（第二部分）

以上样本输出包括由数据库系统监控程序返回的特定数据元素。有关这些和其他监控程序元素的详情，参见 *System Monitor Guide and Reference* 手册。

在下一个示例中（在 UNIX 上），仅生成摘要表。

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

仅生成摘要表。如果使用 -r 选项，则 outfile1 被 /dev/null 置换，而 outfile2（它只包含摘要表）是空的，所以 db2batch 将输出发送至屏幕:

```
Summary of Results
==================
                Elapsed              Agent CPU         Rows      Rows
Statement #     Time (s)             Time (s)          Fetched   Printed
1                    0.074                0.020           5         5
2                    0.037       Not Collected            8         5
Arith. mean     0.055
Geom.  mean     0.052
```

图 35. db2batch 的样本输出 -- 仅摘要表

此基准测试工具也有一个 CLI 选项。使用此选项，可以指定高速缓存的大小。在下列示例中，db2batch 以 CLI 方式运行，其高速缓存大小为 30 个语句:

```
db2batch -d sample -f db2batch.sql -cli 30
```

执行基准测试

一种数据库基准测试程序涉及选择一个配置参数，并使用该参数的不同值运行该测试，直到取得最大好处为止。单个测试应该包括通过相同参数值的几次迭代（例如，10 次）来获取平均最佳值以执行该应用程序，这可以更好地显示参数更改所产生的影响。

当运行您的基准测试程序时，应该将第一次迭代（称为热身运行）视为独立于后续迭代（称为正常运行）的一个单独实例。这是必需的，原因是热身运行产生的结果将包括某些启动活动（如初始化缓冲池）。因此，热身运行与正常运行相比会稍长一些。虽然来自热身运行的信息*可能*实际是有效的，但从统计角度来说是无效的。所以，当计算一组特定参数值的平均最佳值或 CPU 时，要使用来自正常运行的结果。

您可能要考虑使用“性能配置”向导来创建基准测试的热身运行。“性能配置”向导中问到的问题，将涉及到在进行基准测试活动期间为正常运行调整环境配置时要考虑的若干事宜。要使用“性能配置”向导，输入 db2cc 以进入控制中心并从此处继续。

如果使用个别查询来进行基准测试，需要确保将先前查询的可能影响降至最小。这可以通过清除缓冲池来实现，即可读取大量的页（与您的查询无关的）来填充该缓冲池。

在完成单组参数值的迭代之后，就可以更改单个参数。但是，在每个迭代之间，应执行下列任务，以便将基准测试程序的环境复原至它的初始状态:

- 将应用程序数据和数据库管理程序的统计信息返回至它们的初始状态。如果由于测试的需要更新了目录统计信息，则要确保每个迭代都使用相同的统计值。如果使用的数据在测试过程中已更新，则它在测试中必须保持一致。为此:

  - 使用 RESTORE 实用程序来复原整个数据库。该数据库的副本应为它的先前状态，并准备好做下次测试。
  - 使用 IMPORT 或 LOAD 实用程序来复原该数据的调出副本。此方法只允许复原受影响的数据。应对包含此数据的表和索引运行 REORG 和 RUNSTATS 实用程序。
- 通过将应用程序与数据库重新连接，以将该应用程序返回至它的原始状态。

以下是在 OS/2 上进行基准测试时的其他考虑事项:

- 如果在该方案期间发生分页，则要确保 SWAPPER.DAT 已返回至原始大小。
- 要重复操作，重新引导系统（如果有必要的话）。

基准测试程序的输出应该包括每个测试的标识符、程序执行的迭代、语句号和执行的最佳值。在经过一系列测量之后，基准测试结果的摘要可能类似于如下所示:

```
Test      Iter.    Stmt     Timing          SQL Statement
Numbr     Numbr    Numbr    (hh:mm:ss.ss)
 002       05       01      00:00:01.34     CONNECT TO SAMPLE
 002       05       10      00:02:08.15     OPEN cursor_01
 002       05       15      00:00:00.24     FETCH cursor_01
 002       05       15      00:00:00.23     FETCH cursor_01
 002       05       15      00:00:00.28     FETCH cursor_01
 002       05       15      00:00:00.21     FETCH cursor_01
 002       05       15      00:00:00.20     FETCH cursor_01
 002       05       15      00:00:00.22     FETCH cursor_01
 002       05       15      00:00:00.22     FETCH cursor_01
 002       05       20      00:00:00.84     CLOSE cursor_01
 002       05       99      00:00:00.03     CONNECT RESET
```

图 36. 基准测试程序的样本结果

注: 上面报告中的数据仅供说明之用。它**不**表示测量的结果。

检查此报告会看到: CONNECT（语句 01）用去 1.34 秒， OPEN CURSOR（语句 10）用去 2 分钟 8.15 秒，FETCHES（语句 15）返回七行且延迟时间最长，为 .28 秒，CLOSE CURSOR（语句 20）用去 .84 秒，而 CONNECT RESET（语句 99）用去 .03 秒。

您的程序以定界的 ASCII 格式输出数据可能会更好，这样以后可将该数据调入数据库表或电子表格，以便进一步统计分析。

基准测试程序报告的样本输出可能是:

```
      PARAMETER        VALUES FOR EACH BENCHMARK TEST
      TEST NUMBER      001     002     003     004     005
      locklist         63      63      63      63      63
>>    buffpage         1000    1175    1250    1325    1400     <<
      maxappls         8       8       8       8       8
      applheapsz       48      48      48      48      48
      dbheap           128     128     128     128     128
      sortheap         256     256     256     256     256
      maxlocks         22      22      22      22      22
      stmtheap         1024    1024    1024    1024    1024
      SQL STMT         AVERAGE TIMINGS (seconds)
        01             01.34   01.34   01.35   01.35   01.36
        10             02.15   02.00   01.55   01.24   01.00
        15             00.22   00.22   00.22   00.22   00.22
        20             00.84   00.84   00.84   00.84   00.84
        99             00.03   00.03   00.03   00.03   00.03
```

图 37. 基准测试程序的样本最佳值报告

注: 上面报告中的数据仅供说明之用。它**不**表示任何测量的结果。

检查此示例中的数据，表明连续更改 *buffpage* 参数将 OPEN CURSOR 时间从 2.15 秒降低至 1.00 秒。(假定只有一个缓冲池 (1)，且将其大小 (NPAGES) 设置为 -1。这表示该缓冲池的大小由 *buffpage* 参数控制。)

总之，可执行下列步骤 / 迭代来对一个数据库应用程序进行基准测试:

第一步 除下列参数外，将数据库和数据库管理程序的其他调整参数保持为它们的**缺省**值:

- 对于测试的工作负荷和目标很重要的那些参数。(您很少有足够的时间执行基准测试以调整所有参数，所以可能需要使用某些参数的最佳推测值并从该点上开始调整。)
- 日志大小，它应在应用程序的单元测试和系统测试期间确定。(有关详情，参见第345页的『日志文件的大小 (logfilsiz)』。)
- 为了使应用程序能够运行任何必须更改的参数 (即，为防止出现语句堆内存用完这类事件而导致产生负的 SQL 返回码所需的更改)。

对此初始情况运行一组迭代，然后计算平均最佳值或 CPU。

第二步 选择一个且唯一一个调整参数来测试，并更改它的值。

第三步 运行另一组迭代，然后计算平均最佳值或 CPU。

第四步 根据基准测试的结果，执行下列其中一项操作:

- 如果性能提高，则更改同一个参数的值并返回至第三步。继续更改此参数，直到产生最大效益为止。

- 如果性能降低或保持不变，则将该参数返回至其原来的值，返回至第二步，并选择新的参数。重复此过程，直到所有的参数都已被测试为止。

  注： 如果您想将该性能结果绘制成图表，则要查找曲线开始上升或下降的点。

可以编写一个驱动程序，以帮助您进行基准测试。可使用象 REXX 这类语言来编写此驱动程序，或者对于基于 UNIX 的平台，使用 shell 脚本。

此驱动程序将执行基准测试程序，将适当的参数传送给它，通过多次迭代驱动该测试，将环境复原至一致的状态，使用新的参数值设置下一个测试，以及收集／合并测试结果。这些驱动程序可以很灵活，它们可用于运行一整套基准测试，分析结果，并为给定测试提供一个最终和最优参数值报告。

第13章 配置 DB2

配置参数是影响数据库或数据库管理的系统操作特性的值。

数据库管理程序配置参数存在于服务器和客户机上；但是，在客户机中只能设置特定的数据库管理程序配置参数。这些参数是可在服务器上设置的数据库管理配置参数的一个子集。根据所用的“DB2 通用数据库”产品的类型，有一些与配置参数相关的特定问题。例如，在“DB2 扩充企业版”中，一个数据库管理程序配置文件在实例中的所有数据库分区服务器之间共享。

数据库配置参数只驻留在服务器上。

DB2 中设计了一组应用范围较广的调整和配置参数。这些参数分为两个一般类别:

- 第278页的『数据库管理程序参数』
- 第283页的『数据库参数』。

除个别参数的说明外，还提供了下列主题，它们会受配置参数的显著影响:

- 『调整配置参数』。
- 第288页的『按功能分类的参数细节』（每个功能区都有它自己的配置参数列表。）
- 第417页的『附录A. DB2 注册表变量和环境变量』。

  除了与性能相关的配置参数外，对于特定平台，还有应考虑使用的与性能相关的环境变量。
- 第201页的『第8章 操作性能』。
- 第265页的『第12章 基准测试』。

您应该复查第280页的表17和第285页的表19中的所有参数摘要，然后着重复查将给您的工作环境带来最大效益的那些参数的说明和调整。

调整配置参数

数据库管理程序根据这些参数的缺省值分配的磁盘空间和内存可能足以满足您的需要。但在某些情况下，使用这些缺省值可能无法实现最佳性能。

因为缺省值适用于内存相对较小且专门用作数据库服务器的机器，所以如果您的环境存在以下情况，可能需要修改这些参数:

- 大型数据库

- 大量连接
- 对特定应用程序有高性能需求
- 唯一的查询或事务负荷或类型
- 不同的机器配置或用途。

在一个或多个方面，每个事务处理环境都是唯一的。当使用缺省配置时，这些差异可能对数据库管理程序的性能产生深远的影响。因此，强烈建议您调整您的环境配置。

不同类型的应用程序和用户有不同的响应时间需求和期望。应用程序的范围可以从简单的数据输入屏幕到包含几十个复杂的 SQL 语句的策略性应用程序，这些语句在每个工作单元内可存取几十个表。例如，电话客户服务应用程序与批处理报告生成应用程序比较，其响应时间需求可能有相当大的差异。

其他相关的主题可用来帮助您对应用程序进行基准测试，以调整配置参数:

- 『数据库管理程序参数』
- 第283页的『数据库参数』
- 第288页的『按功能分类的参数细节』（每个功能区有它自己的配置参数列表）
- 第201页的『第8章 操作性能』
- 第265页的『第12章 基准测试』
- 在 *System Monitor Guide and Reference* 中的数据库系统监控程序元素说明。

数据库管理程序参数

数据库管理程序参数存储在名为 db2systm 的文件中。此文件是在创建数据库管理程序的实例时创建的。在基于 UNIX 的环境中，可在数据库管理程序的实例的 sqllib 子目录中找到此文件。在所有其他环境中，此文件的缺省位置是 sqllib 目录的 instance 子目录。如果设置了 DB2INSTPROF 变量，则该文件在 DB2INSTPROF 变量指定的目录的 instance 子目录中。

在分区数据库环境中，此文件驻留在共享文件系统上，以便所有数据库分区服务器都具有对同一个文件的存取权。数据库管理程序的配置在所有数据库分区服务器上是相同的。

大多数参数会影响将分配给数据库管理程序的单个实例的系统资源的数量，或者它们会根据对环境的考虑来配置数据库管理程序和不同的通信子系统的设置。此外，还有一些参数，它们仅提供信息而不能更改。所有这些参数都具有全局适用性，与存储在数据库管理程序的该实例下的任何单个数据库无关。

不能直接编辑 db2systm 文件。只能使用提供的 API 或通过调用该 API 的工具来更改或查看它。

注意: 如果您使用非该产品提供的方法编辑该文件，则可能会使您的系统不可用。**我们强烈建议**您不要使用非 DB2 记载和支持的方法来更改此文件。

可使用下列任何一种方法来复位、更新和查看数据库管理程序配置参数:

- 使用 DB2 控制中心。DB2 控制中心提供了“配置实例”笔记本，可用来在客户机或服务器上设置数据库管理程序配置参数。DB2 控制中心还提供了“性能配置”向导，可用来改变服务器上配置参数的值。此向导根据您对一组问题（如对数据库运行的事务的工作负荷和类型）提供的回答，来为参数生成值。有关使用这些界面的信息，参见控制中心提供的联机帮助。
- 使用命令行处理器。可以快速方便地输入更改这些设置的命令。参考 *Command Reference* 以了解有关下列命令的详情:
  - GET DATABASE MANAGER CONFIGURATION（或 GET DBM CFG）
  - UPDATE DATABASE MANAGER CONFIGURATION（或 UPDATE DBM CFG）
  - RESET DATABASE MANAGER CONFIGURATION（或 RESET DBM CFG）。
- 使用应用程序设计接口 (API)。这些 API 可以很容易地从应用程序中调用。有关详情，参考 *Administrative API Reference*。
- 使用“客户机配置辅助程序”。只能使用“客户机配置辅助程序”在客户机上设置数据库管理程序配置参数。

在更改了这些参数之后，必须停止数据库管理程序 (db2stop)，然后重新启动它 (db2start)，以使新参数值生效。对于客户机，数据库管理程序配置参数中的更改将在下次该客户机与服务器连接时生效。当新参数值未立即生效时，查看参数设置，它们将始终显示最新的更新。

注: 如果您更新了 *dft_monswitches* 参数的值，则不必重新启动数据库管理程序；因为当您更改它的值时，会自动更新此参数。

数据库管理程序配置参数摘要

下表列出数据库服务器的数据库管理程序配置文件中的参数。当更改数据库管理程序配置参数时，要考虑每个参数的详细信息。包括缺省值的特定操作环境信息是每个参数说明的一部分。

下表中的“性能影响”列指示每个参数影响系统性能的相对程度。不可能将此列准确地应用于所有环境；您应该将此信息视为一般情况。

- **高** – 指示该参数可以对性能有重要影响。应有意识地决定这些参数的值；在某些情况下，将意味着接受提供的缺省值。
- **中** – 指示该参数可以对性能有某些影响。您的特定环境和需要将确定应对这些参数进行多大程度的调整。
- **低** – 指示该参数对性能没有那么普遍或没有那么重要的影响。
- **无** – 指示该参数对性能没有直接的影响。当您不必因性能的原因调整这些参数时，它们对于您系统配置的其他方面（如启用通信支持）可能很重要。

表 17. 可配置的数据库管理程序配置参数

参数	性能影响	其他信息
agentpri	高	第335页的『代理程序优先级 (agentpri)』
agent_stack_sz	低	第309页的『代理程序栈大小 (agent_stack_sz)』
aslheapsz	高	第313页的『应用程序支持层堆大小 (aslheapsz)』
audit_buf_sz	高	第319页的『审查缓冲区大小 (audit_buf_sz)』
authentication	低	第409页的『认证类型 (authentication)』
backbufsz	中	第296页的『缺省备份缓冲区大小 (backbufsz)』
catalog_noauth	无	第411页的『不需权限就允许编目 (catalog_noauth)』
comm_bandwidth	中	第400页的『通信带宽 (comm_bandwidth)』
conn_elapse	中	第389页的『连接经过时间 (conn_elapse)』
cpuspeed	低（参见注释）	第400页的『CPU 速度 (cpuspeed)』
datalinks	低	第369页的『启用数据链路支持 (datalinks)』
dft_account_str	无	第405页的『缺省交费帐户 (dft_account_str)』
dft_client_adpt	无	第385页的『缺省客户机适配器号 (dft_client_adpt)』
dft_client_comm	无	第384页的『缺省客户机通信协议 (dft_client_comm)』
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_uow*	中	第398页的『缺省数据库系统监控程序开关 (dft_monswitches)』
dftdbpath	无	第411页的『缺省数据库路径 (dftdbpath)』
diaglevel	低	第396页的『诊断错误捕捉级别 (diaglevel)』
diagpath	无	第396页的『诊断数据目录路径 (diagpath)』

表 17. 可配置的数据库管理程序配置参数 (续)

参数	性能影响	其他信息
dir_cache	中	第318页的『目录高速缓存支持 (dir_cache)』
dir_obj_name	无	第383页的『DCE 名称空间中的对象名 (dir_obj_name)』
dir_path_name	无	第382页的『DCE 名称空间中的目录路径名 (dir_path_name)』
dir_type	无	第381页的『目录服务类型 (dir_type)』
discover	中	第386页的『发现方式 (discover)』
discover_comm	低	第387页的『搜索发现通信协议 (discover_comm)』
discover_inst	低	第388页的『Discover 服务器实例 (discover_inst)』
dos_rqrioblk	高	第315页的『DOS 请求器 I/O 块大小 (dos_rqrioblk)』
drda_heap_sz	低	第308页的『DRDA 堆大小 (drda_heap_sz)』
fcm_num_anchors	高	第389页的『FCM 信息锚数 (fcm_num_anchors)』
fcm_num_buffers	高	第390页的『FCM 缓冲区数 (fcm_num_buffers)』
fcm_num_connect	高	第391页的『FCM 连接项数 (fcm_num_connect)』
fcm_num_rqb	高	第391页的『FCM 请求块数 (fcm_num_rqb)』
federated	中	第406页的『联合体数据库系统支持 (federated)』
fileserver	无	第379页的『IPX/SPX 文件服务器名 (fileserver)』
indexrec	中	第355页的『索引重建时间 (indexrec)』
initdari_jvm	中	第343页的『用 JVM 初始化 DARI 进程 (initdari_jvm)』
intra_parallel	高	第395页的『启用分区内并行性 (intra_parallel)』
ipx_socket	无	第380页的『IPX/SPX 套接字号 (ipx_socket)』
java_heap_sz	高	第320页的『最大 Java 解释程序堆大小 (java_heap_sz)』
jdk11_path	无	第405页的『Java Development Kit 1.1 安装路径 (jdk11_path)』
keepdari	中	第341页的『保存 DARI 进程指示符 (keepdari)』
maxagents	中	第336页的『代理程序的最大数目 (maxagents)』
maxcagents	中	第337页的『并行代理程序的最大数目 (maxcagents)』
max_connretries	中	第392页的『节点连接重试次数 (max_connretries)』
max_coordagents	中	第338页的『最大协调代理程序数 (max_coordagents)』

表 17. 可配置的数据库管理程序配置参数 (续)

参数	性能影响	其他信息
maxdari	中	第342页的『DARI 进程的最大数目 (maxdari)』
max_logicagents	中	第339页的『逻辑代理程序的最大数目 (max_logicagents)』
max_querydegree	高	第394页的『最大查询并行度 (max_querydegree)』
max_time_diff	中	第393页的『节点之间的最大时间差 (max_time_diff)』
maxtotfilop	中	第334页的『每个应用程序可打开的最大总文件数 (maxtotfilop)』
min_priv_mem	中	第311页的『落实的最小专用内存 (min_priv_mem)』
mon_heap_sz	低	第317页的『数据库系统监控程序堆大小 (mon_heap_sz)』
nname	无	第377页的『NetBIOS 工作站名 (nname)』
notifylevel	低	第397页的『通知级别 (notifylevel)』
numdb	低	第401页的『并行活动数据库的最大数目 (numdb)』
num_initagents	中	第340页的『池中初始代理程序数 (num_initagents)』
num_initdaris	中	第344页的『存储池中防护 DARI 进程的初始数目 (num_initdaris)』
num_poolagents	高	第339页的『代理程序池大小 (num_poolagents)』
objectname	无	第380页的『IPX/SPX DB2 服务器对象名 (objectname)』
priv_mem_thresh	中	第311页的『专用内存阈值 (priv_mem_thresh)』
query_heap_sz	中	第307页的『查询堆大小 (query_heap_sz)』
restbufsz	中	第296页的『缺省复原缓冲区大小 (restbufsz)』
resync_interval	无	第361页的『事务再同步间隔 (resync_interval)』
route_obj_name	无	第383页的『路径选择信息对象名 (route_obj_name)』
rqrioblk	高	第314页的『客户机 I/O 块大小 (rqrioblk)』
sheapthres	高	第303页的『排序堆阈值 (sheapthres)』
spm_log_file_sz	低	第362页的『同步点管理程序日志文件大小 (spm_log_file_sz)』
spm_log_path	中	第361页的『同步点管理程序日志文件路径 (spm_log_path)』

表 17. 可配置的数据库管理程序配置参数 (续)

参数	性能影响	其他信息
spm_max_resync	低	第363页的『同步点管理程序再同步代理程序限制 (spm_max_resync)』
spm_name	无	第362页的『同步点管理程序名 (spm_name)』
ss_logon	无	第412页的『DB2START/DB2STOP 必需的注册 (ss_logon)』
start_stop_time	低	第393页的『启动和停止超时 (start_stop_time)』
svcename	无	第378页的『TCP/IP 服务名 (svcename)』
sysadm_group	无	第407页的『系统管理权限组名 (sysadm_group)』
sysctrl_group	无	第408页的『系统控制权限组名 (sysctrl_group)』
sysmaint_group	无	第409页的『系统维护权限组名 (sysmaint_group)』
tm_database	无	第360页的『事务管理程序数据库名 (tm_database)』
tp_mon_name	无	第402页的『事务处理器监控程序名 (tp_mon_name)』
tpname	无	第379页的『APPC 事务程序名 (tpname)』
trust_allclnts	无	第413页的『信任所有客户机 (trust_allclnts)』
trust_clntauth	无	第414页的『可信赖客户机认证 (trust_clntauth)』
udf_mem_sz	低	第308页的『UDF 共享内存集大小 (udf_mem_sz)』
注: *cpuspeed* 参数可以对性能产生显著影响，除非在非常特定的情况下，否则应使用缺省值，如参数说明中所述。		

表 18. 参考性数据库管理程序配置参数

参数	其他信息
nodetype	第404页的『机器节点类型 (nodetype)』
release	第365页的『配置文件发行版级别 (release)』

数据库参数

个别数据库的参数存储在名为 SQLDBCON 的配置文件中。此文件与该数据库的其他控制文件一起存储在 SQLnnnnn 目录中，其中 nnnnn 是创建该数据库时指定的编号。（有关此目录位置的详情，*参考管理指南: 计划*中的“数据库物理目录”。）每个数据库都有它自己的配置文件，而该文件中的大多数参数指定分配给该数据库的资源量。该文件还包含说明性信息，以及指示数据库状态的标志。

不能直接编辑 SQLDBCON 文件，而只能通过提供的 API 或调用该 API 的工具来更改或查看。

注意: 如果您使用非 DB2 提供的方法编辑该文件，则可能会使该数据库不可用。**我们强烈建议**您不要使用非 DB2 记载和支持的方法来更改此文件。

可用下列三种方法之一来复位、更新和查看数据库配置参数:

- 使用控制中心。DB2 控制中心同时提供了“配置数据库”笔记本和“性能配置”向导来改变配置参数的值。此向导根据您对一组问题（如对数据库运行的事务的工作负荷和类型）提供的回答，来为参数生成值。有关使用这些界面的信息，参见控制中心提供的联机帮助。

  在分区数据库环境中，每个数据库分区都存在 SQLDBCON 文件。如果从“控制中心”的树视图中的数据库对象启动“控制中心配置数据库”笔记本，则该笔记本将更改所有分区上的值。如果从数据库分区对象启动该笔记本，则它将只更改该分区的值。

  注: “性能配置”向导在分区数据库环境中不可用。
- 使用命令行处理器。可以快速方便地输入更改这些设置的命令。有关以下命令的详情，参考 *Command Reference*:
  - GET DATABASE CONFIGURATION（或 GET DB CFG）
  - UPDATE DATABASE CONFIGURATION（或 UPDATE DB CFG）
  - RESET DATABASE CONFIGURATION（或 RESET DB CFG）
- 使用应用程序设计接口 (API)。这些 API 可以很容易地从主机语言程序中调用。有关详情，参考 *Administrative API Reference*。

当应用程序与数据库连接时，大多数可更改参数的更新不会生效。所有应用程序必须先与该数据库断开。（如果已激活该数据库，则必须将它停用，然后重新激活它。）然后，当与该数据库建立第一个新的连接时，这些更改才生效。您应该注意，某些参数更改，如 *newlogpath*、 *logfilsiz* 和 *logprimary*，由于与分配空间相关的额外开销而可能花相当长的时间才会生效。您可能希望与该数据库建立一个测试连接，以便在存在测试连接时进行更改，这样任何额外开销都不会影响其他用户。如果关心此处讨论的额外开销，应考虑使用 *Command Reference* 中描述的 ACTIVATE DATABASE 命令。

注: 如果您更新了 *mincommit* 参数的值，则不必与该数据库断开；因为当您更改它的值时，会自动更新此参数。

更改某些数据库配置参数可能影响 SQL 优化器选择的存取方案。这些数据库参数在第77页的『影响查询优化的配置参数』中有讨论。当更改该处讨论的任何一个参数后，您应该考虑重新联编您的应用程序，以确保对您的 SQL 语句使用最佳的存取方案。

当新参数值可能未立即生效时，查看参数设置，它们将始终显示最新的更新。

注: 在帮助和其他 DB2 书籍中，大量数据库配置参数（例如，*userexit*）被描述为具有可接受的值 『Yes』 或 『No』，或者 『On』 或 『Off』。为澄清可能混淆的概念，应认为 『Yes』 等效于 『On』，而 『No』 等效于 『Off』。

数据库配置参数摘要

下表列出数据库配置文件中的参数。当更改该数据库配置参数时，要参考该参数的详细信息。

下表中的“性能影响”列指示每个参数影响系统性能的相对程度。不可能将此列准确地应用于所有环境；您应该将此信息视为一般情况。

- **高** - 指示该参数可以对性能有重要影响。应有意识地决定这些参数的值；在某些情况下，将意味着接受提供的缺省值。
- **中** - 指示该参数可以对性能有某些影响。您的特定环境和需要将确定应对这些参数进行多大程度的调整。
- **低** - 指示该参数对性能没有那么普遍或没有那么重要的影响。
- **无** - 指示该参数对性能没有直接的影响。当您不必因性能的原因调整这些参数时，它们对于您系统配置的其他方面（如启用通信支持）可能很重要。

表 19. 可配置的数据库配置参数

参数	性能影响	其他信息
app_ctl_heap_sz	中	第301页的『应用程序控制堆大小 (app_ctl_heap_sz)』
applheapsz	中	第305页的『应用程序堆大小 (applheapsz)』
audit_buf_sz	中	第319页的『审查缓冲区大小 (audit_buf_sz)』
autorestart	低	第355页的『启用自动重新启动 (autorestart)』
avg_appls	高	第332页的『活动应用程序的平均数 (avg_appls)』
buffpage	高（活动时）	第290页的『缓冲池大小 (buffpage)』
catalogcache_sz	中	第293页的『目录高速缓存大小 (catalogcache_sz)』
chngpgs_thresh	高	第324页的『更改页阈值 (chngpgs_thresh)』
copyprotect	无	第367页的『启用副本保护 (copyprotect)』
dbheap	中	第292页的『数据库堆 (dbheap)』
dft_degree	高	第373页的『缺省度 (dft_degree)』
dft_extent_sz	中	第329页的『表空间的缺省数据块大小 (dft_extent_sz)』
dft_loadrec_ses	中	第357页的『缺省的装入恢复对话数 (dft_loadrec_ses)』

表 19. 可配置的数据库配置参数 (续)

参数	性能影响	其他信息
dft_prefetch_sz	中	第328页的『缺省预读取大小 (dft_prefetch_sz)』
dft_queryopt	中	第374页的『缺省查询优化级别 (dft_queryopt)』
dft_refresh_age	中	第374页的『缺省刷新时限 (dft_refresh_age)』
dft_sqlmathwarn	无	第372页的『遇到算术异常继续 (dft_sqlmathwarn)』
dir_obj_name	无	第383页的『DCE 名称空间中的对象名 (dir_obj_name)』
discover_db	中	第386页的『发现数据库 (discover_db)』
dlchktime	中	第321页的『检查死锁的时间间隔 (dlchktime)』
dl_expint	无	第367页的『数据链路存取令牌到期时间间隔 (dl_expint)』
dl_num_copies	无	第368页的『数据链路副本数 (dl_num_copies)』
dl_time_drop	无	第368页的『卸下后的数据链路时间 (dl_time_drop)』
dl_token	低	第368页的『数据链路令牌算法 (dl_token)』
dl_upper	无	第369页的『大写的数据链路令牌 (dl_upper)』
dyn_query_mgmt	低	第364页的『动态 SQL 查询管理 (dyn_query_mgmt)』
estore_seg_sz	中	第329页的『扩充内存段大小 (estore_seg_sz)』
indexrec	中	第355页的『索引重建时间 (indexrec)』
indexsort	低（参见287）	第327页的『索引排序标志 (indexsort)』
locklist	高（当它影响逐步升级时）	第297页的『锁定列表的最大存储器 (locklist)』
locktimeout	中	第323页的『锁定超时 (locktimeout)』
logbufsz	高	第294页的『日志缓冲区大小 (logbufsz)』
logfilsiz	中	第345页的『日志文件的大小 (logfilsiz)』
logprimary	中	第346页的『主日志文件数 (logprimary)』
logretain	低	第353页的『启用日志保留 (logretain)』
logsecond	中	第348页的『辅助日志文件数 (logsecond)』
maxappls	中	第331页的『活动应用程序的最大数目 (maxappls)』
maxfilop	中	第333页的『每个应用程序可打开的数据库文件的最大数目 (maxfilop)』
maxlocks	高（当它影响逐步升级时）	第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』
mincommit	高	第350页的『对组的落实次数 (mincommit)』

表 19. 可配置的数据库配置参数 (续)

参数	性能影响	其他信息
newlogpath	低	第348页的『更改数据库日志路径 (newlogpath)』
num_db_backups	无	第357页的『数据库备份数 (num_db_backups)』
num_estore_segs	中	第330页的『扩充内存段数 (num_estore_segs)』
num_freqvalues	低	第375页的『保存的高频值数目 (num_freqvalues)』
num_iocleaners	高	第325页的『异步页清除器数 (num_iocleaners)』
num_ioservers	高	第326页的『I/O 服务器数目 (num_ioservers)』
num_quantiles	低	第376页的『列的分位数数目 (num_quantiles)』
pckcachesz	高	第299页的『程序包高速缓存大小 (pckcachesz)』
rec_his_retentn	无	第357页的『恢复历史保留期 (rec_his_retentn)』
seqdetect	高	第327页的『顺序检测标志 (seqdetect)』
softmax	中	第351页的『恢复范围和软检查点间隔 (softmax)』
sortheap	高	第303页的『排序堆大小 (sortheap)』
stat_heap_sz	低	第306页的『统计堆大小 (stat_heap_sz)』
stmtheap	中	第305页的『语句堆大小 (stmtheap)』
tsm_mgmtclass	无	第358页的『Tivoli 存储管理器管理类 (tsm_mgmtclass)』
tsm_nodename	无	第359页的『Tivoli 存储管理器节点名 (tsm_nodename)』
tsm_owner	无	第359页的『Tivoli 存储管理器拥有者名 (tsm_owner)』
tsm_password	无	第358页的『Tivoli 存储管理器 口令 (tsm_password)』
userexit	低	第354页的『启用用户出口 (userexit)』
util_heap_sz	低	第295页的『实用程序堆大小 (util_heap_sz)』
注: 将 *indexsort* 参数更改为非缺省值的值，可能对创建索引的性能带来负面影响。您应该始终尝试使用此参数的缺省值。		

表 20. 参考性数据库配置参数

参数	其他信息
backup_pending	第370页的『备份暂挂指示符 (backup_pending)』
codepage	第366页的『数据库的代码页 (codepage)』
codeset	第366页的『数据库的代码集 (codeset)』
collate_info	第366页的『整理信息 (collate_info)』
country	第365页的『数据库的国家代码 (country)』

表 20. 参考性数据库配置参数 (续)

参数	其他信息
database_consistent	第370页的『数据库一致 (database_consistent)』
database_level	第365页的『数据库发行版级别 (database_level)』
log_retain_status	第371页的『日志保留状态指示符 (log_retain_status)』
loghead	第350页的『第一个活动的日志文件 (loghead)』
logpath	第350页的『日志文件的位置 (logpath)』
multipage_alloc	第371页的『启用多页文件分配 (multipage_alloc)』
numsegs	第329页的『缺省 SMS 容器数 (numsegs)』
release	第365页的『配置文件发行版级别 (release)』
restore_pending	第371页的『复原暂挂 (restore_pending)』
rollfwd_pending	第370页的『前滚暂挂指示符 (rollfwd_pending)』
territory	第365页的『数据库的地区 (territory)』
user_exit_status	第371页的『用户出口状态指示符 (user_exit_status)』

按功能分类的参数细节

下面几节提供辅助您理解和调整不同配置参数的附加详情。对个别参数的讨论是根据它们的功能或用途来组织的:

- 第289页的『能力管理』
- 第344页的『记录和恢复』
- 第364页的『数据库管理』
- 第377页的『通信』
- 第388页的『并行』
- 第395页的『实例管理』。

每个参数的讨论包括以下信息:

配置类型

指示哪个配置文件包含该参数的设置:

- 数据库管理程序（它影响数据库管理程序的实例和在该实例内定义的所有数据库）
- 数据库（它影响特定的数据库）

参数类型

指示您是否可以更改该参数值:

- *可配置的*

可能是一个范围内的值，可能需要根据数据库管理员对该应用程序的了解和／或从基准测试获得的经验来调整该参数。

- *参考性*

  这些参数只能由数据库管理程序自己更改，并将包含类似如下的信息：创建的数据库所属的 DB2 的发行版，或必需的备份暂挂的指示。

能力管理

在数据库和数据库管理程序级别有大量配置参数可影响您系统上的吞吐量。这些参数被分为下列几组:

- 『数据库共享内存』
- 第301页的『应用程序共享内存』
- 第302页的『代理程序专用内存』
- 第313页的『代理程序／应用程序通信内存』
- 第317页的『数据库管理程序实例内存』
- 第321页的『锁定』
- 第324页的『I/O 和存储器』
- 第330页的『代理程序』
- 第341页的『数据库应用程序远程接口 (DARI)』。

有关 DB2 内存管理的介绍，参见 第201页的『DB2 如何使用内存』。

数据库共享内存

下列参数影响在系统上分配的数据库全局内存:

- 第290页的『缓冲池大小 (buffpage)』。
- 第292页的『数据库堆 (dbheap)』。
- 第293页的『目录高速缓存大小 (catalogcache_sz)』。
- 第294页的『日志缓冲区大小 (logbufsz)』。
- 第295页的『实用程序堆大小 (util_heap_sz)』。
- 第296页的『缺省备份缓冲区大小 (backbufsz)』。
- 第296页的『缺省复原缓冲区大小 (restbufsz)』。
- 第297页的『锁定列表的最大存储器 (locklist)』。
- 第299页的『程序包高速缓存大小 (pckcachesz)』。

有关数据库全局内存如何与数据库管理程序分配的内存的剩余部分相关的信息，参见第201页的『DB2 如何使用内存』。

缓冲池大小 (buffpage)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 32 位平台

1 000 [2 — 524 288]

UNIX 64 位平台

1 000 [2 — 2 147 483 647]

OS/2 和 Windows NT

250 [2 — 524 288]

计量单位 页

分配时 当第一个应用程序连接至数据库时

释放时 当最后一个应用程序与数据库断开时

相关参数

- 第324页的『更改页阈值 (chngpgs_thresh)』
- 第292页的『数据库堆 (dbheap)』
- 第325页的『异步页清除器数 (num_iocleaners)』

每个数据库至少有一个缓冲池（在创建该数据库时创建的 IBMDEFAULTBP），并且可以有多个缓冲池。所有缓冲池都驻留在全局内存中，全局内存对于使用该数据库的所有应用程序都是可用的。该内存是在该数据库所在的机器上分配的。如果缓冲池足够大，可以将需要的数据保留在内存中，则将进行的磁盘活动就较少。相反，如果缓冲池不够大，由于需要进行大量磁盘活动 (I/O) 来处理应用程序所需要的数据，数据库的总体性能可能会大大下降，并且数据库管理程序可能会受到 I/O 限制。

当用 NPAGES -1 运行 CREATE BUFFERPOOL 或 ALTER BUFFERPOOL 语句时，*buffpage* 参数控制缓冲池的大小；否则，忽略 *buffpage* 参数，并且将用由 NPAGES 参数指定的页数来创建缓冲池。

要确定 *buffpage* 参数是否对于缓冲池是活动的，执行:

```
SELECT * from SYSCAT.BUFFERPOOLS。
```

NPAGES 值为 -1 的每个缓冲池都使用 *buffpage*。

在缓冲池大小和其他系统用户的内存分配之间有一个折衷选择。在多用户高事务率服务器上，数据库服务器的内存需求十分重要，以致数据库服务器和文件或通信服务器通常是分开的，并且驻留在不同的机器上。

如果您的查询存取别名，当出现下列情况时，则考虑增加缓冲池的大小:

- 优化器决定大多数或所有的操作在本地完成。处理查询时，优化器通常会尽可能将这些操作下推到数据源。例如，通常在数据源对 GROUP BY 运算符求值。然而，在 DB2 中具体化表和在本地执行操作可能是花费最少的方法。如果 DB2 服务器工作站比数据源工作站功能更强大，则会出现这种情况。
- 排序操作必须在本地完成。按照 DB2 整理顺序对包含别名的查询排序。如果数据源的整理顺序不同，则在本地执行所有的排序操作。

当第一个应用程序连接至该数据库时，或显式激活该数据库时，分配所有缓冲池。当应用程序请求该数据库外部的数据时，将包含该数据的页从磁盘传送至其中一个缓冲池中。（注意: 该数据库数据存储在磁盘上表内的页中。） 在该页被更改且发生下列一种情况之前，不会将页写回至磁盘:

- 所有应用程序与该数据库断开
- 显式释放该数据库
- 该数据库停顿（即所有连接的应用程序已落实）
- 对于需要读入缓冲池的另一页，其空间是必需的
- 页清除器可用 (*num_iocleaners*) 并且由数据库管理程序激活。

建议:

- 如果不使用 *buffpage* 配置参数，您可使用 CREATE BUFFERPOOL 和 ALTER BUFFERPOOL SQL 语句来创建和更改缓冲池及其大小
- 缓冲池大小由优化器用于确定存取方案。在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。
- 因为所有缓冲池的大小都可能对性能有重大影响，所以您应该考虑下列因素，以确保不发生过度的页交换:
  - 您的机器上已安装的内存量。
  - 与数据库管理程序在同一机器上并行运行的其他应用程序所需要的内存。

  当没有足够内存保存正存取的页时，发生*页交换*。结果是将该页写入（“交换到”）临时磁盘存储器中以给其他页留出空间。当需要临时磁盘存储器上的页时，将该页“交换回”至内存中。
- 在下列情况下，可能希望将机器内存的 75% 分配给数据库缓冲池:
  - 多个用户
  - 只将机器用作数据库服务器
  - 大量对相同数据和索引页的重复存取
  - 机器上有一个数据库。
- 对于已分配的每个缓冲池页，有些空间用于内部控制结构的数据库堆。

如果缓冲池的总大小增加，则也可能需要增加 *dbheap*。

- 如果数据源整理顺序与 DB2 整理顺序匹配，则确保将服务器选项 collating_sequence 设置为指示这种情况。

可以使用数据库系统监控程序来计算缓冲池命中率，它可以帮助您调整缓冲池。参考 *System Monitor Guide and Reference*。

数据库堆 (dbheap)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 1200 [32 – 524 288]

OS/2 和 Windows NT 带本地客户机和远程客户机的数据库服务器
600 [32 – 524 288]

OS/2 和 Windows NT 带本地客户机的数据库服务器 300 [32 – 524 288]

计量单位 页 (4 KB)

分配时 与数据库的首个连接

释放时 当最后一个应用程序与数据库断开时

相关参数

- 第293页的『目录高速缓存大小 (catalogcache_sz)』
- 第294页的『日志缓冲区大小 (logbufsz)』

每个数据库有一个数据库堆，并且数据库管理程序代表所有连接至数据库的应用程序使用数据库堆。它包含表、索引、表空间和缓冲池的控制块信息。它还包含日志缓冲区 (*logbufsz*) 和目录高速缓存 (*catalogcache_sz*) 的空间。因此，堆大小将取决于在某个给定时间堆中存储的控制块数。控制块信息保存在堆中，直到所有应用程序与数据库断开为止。

在第一个连接上分配数据库管理程序启动所需的最小量。按需要将该数据区扩充至由 *dbheap* 指定的最大大小。

建议: 当应用程序接收到一个错误，指示数据库堆中没有足够的存储器可用于处理该语句时，需要增加此值。

可以使用数据库系统监控程序跟踪用于数据库堆的最大内存量。有关详情，参见 *System Monitor Guide and Reference* 中的 *db_heap_top（分配的最大数据库堆）*监控程序元素说明。

当设置此参数时，您应该考虑:

- *logbufsz* 的值，因为该日志缓冲区是从数据库堆中分配的。
- *catalogcache_sz* 的值，因为该目录高速缓存是从数据库堆中分配的。

目录高速缓存大小 (catalogcache_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 64 [1 – 60 000]

OS/2 和 Windows NT 带本地客户机和远程客户机的数据库服务器 32 [1 – 60 000]

OS/2 和 Windows NT 带本地客户机的数据库服务器 16 [1 – 60 000]

计量单位 页 (4 KB)

相关参数

- 第292页的『数据库堆 (dbheap)』
- 第294页的『日志缓冲区大小 (logbufsz)』
- 第301页的『应用程序控制堆大小 (app_ctl_heap_sz)』

此参数指示目录高速缓存可以从数据库堆 (*dbheap*) 中使用的最大空间容量。目录高速缓存用于存储在 SQL 语句编译期间，引用表、视图或别名时所使用的表描述符信息。

如果先前语句中已引用了相同的表、视图或别名，则使用此高速缓存有助于改进联编 SQL 语句（包括动态 SQL）的性能。已说明临时表的描述符信息不存储在目录高速缓存中；而是使用应用程序控制堆。

对一个表运行任何 DDL 语句都将清除该表在目录高速缓存中的项目。否则表的项目会一直保存到高速缓存中另一个不同的表需要空间为止，但是要等到与引用该表的任何工作单元完成后，才从高速缓存中清除该表的项目。

建议: 从缺省值开始，并使用数据库系统监控程序来调整它。

有关下列监控程序元素的信息，参见 *System Monitor Guide and Reference*:

- *cat_cache_lookups*（目录高速缓存查找）
- *cat_cache_inserts*（目录高速缓存插入）
- *cat_cache_overflows*（目录高速缓存溢出）
- *cat_cache_heap_full*（目录高速缓存堆满载）

这些数据库系统监控程序元素能帮助您确定是否应调整此配置参数。在调整此参数时，增加量要小，例如，一次两页。

注: 目录高速缓存只存在于多节点环境的目录节点中。

一般来说，如果一个工作单元包含几个动态 SQL 语句或者您正在联编包含许多静态 SQL 语句的程序包，则需要更多的高速缓存空间。

在设置目录高速缓存的大小时，还应考虑日志文件 (*logbufsz*) 的大小，因为 *catalogcache_sz* 和 *logbufsz* 都是从数据库堆 (*dbheap*) 中分配的。

日志缓冲区大小 (logbufsz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 32 位平台

8 [4 — 4 096]

UNIX 64 位平台

8 [4 — 65 535]

OS/2 和 Windows NT

8 [4 — 4 096]

计量单位 页 (4 KB)

相关参数

- 第293页的『目录高速缓存大小 (catalogcache_sz)』
- 第292页的『数据库堆 (dbheap)』
- 第350页的『对组的落实次数 (mincommit)』

在将这些记录写入磁盘之前，此参数允许您指定用作日志记录的缓冲区的数据库堆阵的容量（由 *dbheap* 参数定义）。当发生下列一种情况时会将日志记录写入磁盘:

- 一个事务落实或一组事务落实，由 *mincommit* 配置参数定义

- 日志缓冲区已满
- 发生其他内部数据库管理程序事件。

此参数也必须小于或等于 *dbheap* 参数。缓冲日志记录将使得日志文件 I/O 的效率更高，因为将日志记录写入磁盘的频率减小，每次可写入更多的日志记录得。

建议: 如果在专用的日志磁盘上有大量的读取活动，或者频繁使用磁盘，则要增加此缓冲区的大小。当增加此参数的值时，您也应考虑 *dbheap* 参数，因为该日志缓冲区使用由 *dbheap* 参数控制的空间。

可以使用数据库系统监控程序确定多少日志缓冲空间用于特定事务（或工作单元）。

有关详情，参考 *System Monitor Guide and Reference* 中的 *log_space_used*（使用的工作单元日志空间）监控程序元素说明。

当设置日志缓冲区大小时，还要考虑目录高速缓存的大小 (*catalogcache_sz*)，因为 *logbufsz_sz* 和 *catalogcache_sz* 都是从数据库堆 (*dbheap*) 分配的。

实用程序堆大小 (util_heap_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 5000 [16 – 524 288]

计量单位 页 (4 KB)

分配时 当数据库管理程序实用程序需要时

释放时 当实用程序不再需要内存时

相关参数

- 第296页的『缺省备份缓冲区大小 (backbufsz)』
- 第296页的『缺省复原缓冲区大小 (restbufsz)』

此参数指示可由 BACKUP、RESTORE 和 LOAD 以及装入恢复实用程序同时使用的最大内存量。

建议: 使用缺省值，除非实用程序耗尽空间，在这种情况下应增加此值。如果系统上的内存受约束，则可能期望降低此参数的值以限制数据库实用程序使用的内存。如果此参数设置得太低，则可能不能并行地运行实用程序。必须将此参数设置得足够大容纳想要分配用于并行实用程序的所有缓冲区。

缺省备份缓冲区大小 (backbufsz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 1024 [8 — 524 288]

计量单位 页 (4 KB)

分配时 当调用备份实用程序时

释放时 当备份实用程序完成其处理时

相关参数

- 『缺省复原缓冲区大小 (restbufsz)』
- 第295页的『实用程序堆大小 (util_heap_sz)』

如果调用备份实用程序时未显式指定缓冲区大小的话，此参数指定备份数据库时使用的缓冲区的大小。有关备份实用程序的详情，参考 *Command Reference*。

当备份数据库时，首先将数据复制到内部缓冲区中。然后在缓冲区已满时将数据从此缓冲区写入备份媒体。

调整此缓冲区大小可帮助改进备份实用程序的性能，并将对其他并行数据库操作性能的影响减至最小。

缺省复原缓冲区大小 (restbufsz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 1024 [16 — 524 288]

计量单位 页 (4 KB)

分配时 当调用复原实用程序时

释放时 当复原实用程序完成其处理时

相关参数

- 第296页的『缺省备份缓冲区大小 (backbufsz)』
- 第295页的『实用程序堆大小 (util_heap_sz)』

如果调用复原实用程序时未显式指定缓冲区大小，则此参数指定复原数据库时所使用的缓冲区的大小。有关复原实用程序的详情，参考 *Command Reference*。

当复原数据库时，首先将数据从备份媒体复制到内部缓冲区中。然后在缓冲区已满时将数据从此缓冲区写入目标数据库媒体。

调整此缓冲区大小可帮助改进复原数据库实用程序的性能，并将对其他并行数据库操作性能的影响减至最小。

锁定列表的最大存储器 (locklist)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 100 [4 – 60 000]

OS/2 和 NT 带本地客户机和远程客户机的数据库服务器 50 [4 – 60 000]

OS/2 和 NT 带本地客户机的数据库服务器 25 [4 – 60 000]

计量单位 页 (4 KB)

分配时 当第一个应用程序连接至数据库时

释放时 当最后一个应用程序与数据库断开时

相关参数

- 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』
- 第331页的『活动应用程序的最大数目 (maxappls)』

此参数指示分配给锁定列表的内存量。每个数据库有一个锁定列表，并且锁定列表包含由并行连接至数据库的所有应用程序持有的锁定。锁定是数据库管理程序

用来控制多个应用程序并行存取数据库中的数据的机制。行和表都可以锁定。数据库管理程序也可获取锁定来供内部使用。

每个锁定需要 36 字节或 72 字节的锁定列表，这取决于是否持有对该对象的其他锁定:

- 在一个没有持有其他锁定的对象上持有一个锁定需要 72 字节
- 在一个持有了现存锁定的对象上记录一个锁定需要 36 字节。

当一个应用程序使用的锁定列表的百分比达到 *maxlocks* 时，数据库管理程序对应用程序持有的锁定执行从行到表的锁定逐步升级（以下将说明）。虽然逐步升级过程本身花不了多少时间，但锁定整个表（与个别行比较）降低了并行性，并可能因对受影响的表进行后续存取而降低整个数据库性能。关于如何控制锁定列表大小的建议是:

- 经常执行 COMMIT 以释放锁定。
- 当执行很多更新时，在更新前锁定整个表（使用 SQL LOCK TABLE 语句）。这样将只使用一个锁定，防止其他锁定干扰更新，但却降低了数据的并行性。

  您还可以在 ALTER TABLE 语句中使用 LOCKSIZE 参数，来控制对特定表执行锁定的方式。有关详情，参考 *SQL Reference*。

  使用“可重复读”隔离级别可能导致自动表锁定。 有关隔离级别的详情，参见第37页的『第3章 应用程序考虑事项』。
- 只要有可能，使用“游标稳定性”隔离级别以减少所持有的共享锁定数。如果未解决应用程序的完整性需求，则使用“未落实的读”代替“游标稳定性”以进一步减少锁定量。

一旦锁定列表已满，性能就可能会降低，因为锁定逐步升级将生成更多的表锁定和更少的行锁定，从而降低数据库中共享对象上的并行性。另外，应用程序间可能有更多的死锁（因为这些应用程序都在等待有限数目的表锁定），这样将导致事务回滚。当数据库的锁定请求数达到最大值时，应用程序将接收到值为 -912 的 SQLCODE。

建议: 如果锁定逐步升级导致性能问题，可能需要增加此参数或 *maxlocks* 参数的值。可以使用数据库系统监控程序确定是否正在发生锁定逐步升级。

有关详情，参见 *System Monitor Guide and Reference* 中的 *lock_escals（锁定逐步升级）*监控程序元素说明。

下列步骤可帮助确定锁定列表所需的页数:

1. 计算锁定列表大小的下限:

   ```
   (512 * 36 * maxappls) / 4096
   ```

其中 512 是估计的每个应用程序的平均锁定数，36 是对具有现存锁定的对象的每个锁定所需要的字节数。

2. 计算锁定列表大小的上限:

```
(512 * 72 * maxappls) / 4096
```

其中 72 是对一个对象的第一个锁定所需要的字节数。

3. 估计数据将发生的并行量，并根据您的期望为 *locklist* 选择一个在您计算的上限和下限之间的初始值。
4. 按如下所述，使用数据库系统监控程序调整此参数的值。

可以使用数据库系统监控程序确定一个给定事务持有的锁定的最大数目。

有关详情，参见 *System Monitor Guide and Reference* 中的 *locks_held_top（保持的最大锁定数）*监控程序元素说明。

此信息可帮助您验证或调整估计的每个应用程序的平均锁定数。为了执行此验证，将必须对几个应用程序进行采样，记下在事务级别而不是应用程序级别上提供的监控信息。

如果增加了 *maxappls*，或者如果正运行的应用程序执行的落实不频繁，可能也要增加 *locklist*。

在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

有关应用程序性能和影响查询优化的详情，参见第35页的『第2部分 调整应用程序性能』。

程序包高速缓存大小 (pckcachesz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 32 位平台

-1 [-1, 32 — 64 000]

UNIX 64 位平台

-1 [-1, 32 — 524 288]

OS/2 和 Windows NT

-1 [-1, 32 — 64 000]

计量单位 页 (4 KB)

分配时	当初始化数据库时
释放时	当关闭数据库时

此参数是在数据库全局内存之外分配的，用于数据库上的高速缓存静态和动态 SQL 语句。在一个分区数据库系统中，每个数据库分区都有一个程序包高速缓存。

高速缓存程序包使数据库管理程序在重新装入程序包时可以不存取系统目录；或者对于动态 SQL，可以免去编译这一步，从而减少其内部额外开销。将这些段保存在程序包高速缓存中，直到发生下列其中一种情况:
- 数据库关闭
- 程序包或动态 SQL 语句无效
- 高速缓存空间用完。

静态或动态 SQL 语句程序段的高速缓存可提高性能，尤其是在与数据库连接的应用程序多次使用同一个语句时。这在事务处理应用程序中特别重要。

通过在服务器或分区数据库环境中使用缺省值 (-1)，用来计算该页分配的值是为 *maxappls* 配置参数指定的值的 8 倍。例外情况是 *maxappls* 的 8 倍小于 32。在这种情况下，缺省值 -1 将 *pckcachesz* 设置为 32。

建议: 当调整此参数时，应考虑如果将为该程序包高速缓存保留的额外内存用于其他目的（如缓冲池），该内存是否可能更有效。因此，应在调整此参数时使用基准技术。

当最初使用好几个程序段，而后只有少数几个程序段重复运行时，调整此参数就特别重要。如果高速缓存太大，则因保存初始程序段的副本而浪费内存。

有关下列监控程序元素的信息，参见 *System Monitor Guide and Reference*:
- *pkg_cache_lookups*（程序包高速缓存查找数）
- *pkg_cache_inserts*（程序包高速缓存插入数）
- *pkg_cache_size_top*（最大程序包高速缓存大小）
- *pkg_cache_num_overflows*（程序包高速缓存溢出数）

这些数据库系统监控程序元素能帮助您确定是否应调整此配置参数。

注: 程序包高速缓存是工作高速缓存，所以不能将此参数设置为零。此高速缓存中必须分配有足够的内存以保存当前执行的 SQL 语句的所有程序段。如果分配的空间比当前需要的空间多，则程序段被高速缓存。下次需要这些程序段时只需简单地执行它们，而不必装入或编译。

由 *pckcachese* 参数指定的限制是软限制。如果数据库共享集合中还有可用的内存，如果有必要的话，可以超过该限制。可使用 *pkg_cache_size_top* 监控程序

元素确定程序包高速缓存达到的最大值，并用 *pkg_cache_num_overflows* 监控程序元素确定超过了由 *pckcachesz* 参数指定的限制的多少倍。

应用程序共享内存

下列参数指定为一个应用程序工作的所有代理程序（协调代理程序和子代理程序）使用的工作区:

- 『应用程序控制堆大小 (app_ctl_heap_sz)』

应用程序控制堆大小 (app_ctl_heap_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

带本地客户机和远程客户机的数据库服务器
128 [1–64 000]

带本地客户机的数据库服务器
64 [1–64 000]（仅适用于非 UNIX 平台）

128 [1–64 000]（仅适用于基于 UNIX 的平台）

带本地客户机和远程客户机的分区数据库服务器
256 [1–64 000]

计量单位 页 (4 KB)

分配时 当应用程序启动时

释放时 当应用程序完成时

相关参数

- 第293页的『目录高速缓存大小 (catalogcache_sz)』
- 第305页的『应用程序堆大小 (applheapsz)』
- 第395页的『启用分区内并行性 (intra_parallel)』

此参数确定应用程序控制共享内存的最大大小，以 4 KB 页为单位。应用程序控制堆是从此共享内存中分配的。

在数据库中（或者，如果是分区数据库系统，则在每个数据库分区中），每个活动的应用程序都会分配一个应用程序控制堆。第一个代理程序在处理连接期间分配该堆，以接收对数据库（或数据库分区）中应用程序的请求。该堆是必需的，

这样才可以在为同一个应用程序工作的代理程序之间共享信息（在分区数据库环境中，共享发生在数据库分区级别：共享不会跨数据库分区发生）。

这个堆还用来存储已说明临时表的描述符信息。所有尚未显式卸下的已说明临时表的描述符信息存放在这个堆的内存中，在卸下已说明临时表之前，不能卸下这些信息。

注:

1. 在分区数据库环境中，这个堆用于存储代理程序和子代理程序的 SQL 语句执行段的副本。然而，正如在所有其他环境中的代理程序所做的那样，对称多处理器代理程序 (SMP) 的子代理程序使用 *applheapsz*。
2. 仅对将 *intra_parallel* 参数设置为 on 并将 CURRENT DEGREE 专用寄存器设置为大于 1 的值的其他数据库进行分配。有关 CURRENT DEGREE 专用寄存器的详情，参考 *SQL Reference*。

建议：最初，从缺省值开始。如果您在运行复杂的应用程序，如果您的系统包含大量的数据库分区，或者如果您使用已说明临时表，则可能需要将该值设置得高一些。所需的内存量随当前活动的已说明临时表数的增加而增加。与带有较少列的表相比，带有许多列的已说明临时表的表说明符大小要大得多，因此，如果应用程序的已说明临时表中有相当多的列，则也会增加对应用程序控制堆的需求。

代理程序专用内存

下列参数影响每个数据库代理程序使用的内存容量:

- 第303页的『排序堆大小 (sortheap)』。
- 第303页的『排序堆阈值 (sheapthres)』。
- 第305页的『语句堆大小 (stmtheap)』。
- 第305页的『应用程序堆大小 (applheapsz)』。
- 第306页的『统计堆大小 (stat_heap_sz)』。
- 第307页的『查询堆大小 (query_heap_sz)』。
- 第308页的『DRDA 堆大小 (drda_heap_sz)』。
- 第308页的『UDF 共享内存集大小 (udf_mem_sz)』。
- 第309页的『代理程序栈大小 (agent_stack_sz)』。
- 第311页的『落实的最小专用内存 (min_priv_mem)』。
- 第311页的『专用内存阈值 (priv_mem_thresh)』。
- 第320页的『最大 Java 解释程序堆大小 (java_heap_sz)』。在基于 UNIX 的平台上，为每个代理程序都分配了 *java_heap_sz*。

有关专用代理程序内存如何与数据库管理程序分配的内存的剩余部分相关的信息，参见 第201页的『DB2 如何使用内存』。

排序堆大小 (sortheap)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 256 [16 – 524 288]

计量单位 页 (4 KB)

分配时 需要执行排序时

释放时 当排序完成时

相关参数 『排序堆阈值 (sheapthres)』

此参数定义要用于专用排序的专用内存页的最大数目或要用于共享排序的共享内存页的最大数目。如果排序为专用排序，则此参数将影响代理程序专用内存。如果排序为共享排序，则此参数将影响数据库共享内存。每个排序有一个独立排序堆，该排序堆是由数据库管理程序按照需要分配的。此排序堆是将数据排序的区域。如果由优化器定向，则将使用优化器提供的信息分配一个比此参数指定的排序堆小的排序堆。

建议:

- 适当的索引可使排序堆的使用减至最小程度。
- 当需要进行频繁的大排序时，增大此参数的大小。
- 当增加此参数的值时，您应检查在数据库管理程序配置文件中的 *sheapthres* 参数是否也需要调整。
- 优化器在确定存取路径时使用排序堆大小。在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

排序堆阈值 (sheapthres)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

UNIX 32 位平台

20 000 [250 — 2 097 152]

UNIX 64 位平台

20 000 [250 — 2 147 483 647]

OS/2 和 Windows NT

10 000 [250 — 2 097 152]

计量单位 页 (4 KB)

相关参数 第303页的『排序堆大小 (sortheap)』

专用排序和共享排序使用两个不同内存资源中的内存。共享排序内存区的大小是根据 *sheapthres* 的值，在第一次与数据库连接时静态地预先确定的。专用排序内存区的大小不受限制。

sheapthres 参数对于专用和共享的排序用法是不同的:

- 对于专用排序，此参数是在一个实例内对任何给定时间专用排序可占用的总内存量的软限制。当一个实例的总专用排序内存占用达到了此限制时，为附加进入的专用排序请求分配的内存将显著减少。
- 对于共享排序，此参数是在一个数据库内对任何给定时间共享排序占用的总内存量的硬限制。当达到此限制时，将不允许其他共享排序内存请求（直到总共享内存的占用降低至 *sheapthres* 指定的限制以下为止）。

使用该排序堆的那些操作的示例包括: 散列连接和表位于内存中的操作。

显式定义阈值可防止数据库管理程序对大量排序使用过量内存。

建议: 理想的情况是，应将此参数设置为您在数据库管理程序实例中拥有的最大 *sortheap* 参数的一个合理倍数。此参数**至少**应是该实例中为任何数据库定义的最大 *sortheap* 的两倍。

如果您正执行专用排序并且您的系统不受内存约束，则可使用下列步骤来计算此参数的理想值:

1. 计算每个数据库的典型排序堆的使用:

   （对该数据库运行的并行代理程序的典型数目）
   * （sortheap，为该数据库定义的）

2. 对以上结果求和，该值提供可在实例中所有数据库的典型环境中使用的总排序堆。

有关在 SMP 环境中执行排序的信息，参阅第162页的『并行排序策略』。

应使用基准技术来调整此参数以寻求排序性能和内存用法之间的恰当平衡。有关详情，参见第265页的『第12章 基准测试』。有关排序的详情，另见第220页的『排序』。

可使用数据库系统监控程序追踪排序活动。

有关详情，参考 *System Monitor Guide and Reference* 中的下列监控程序元素说明:

- *post_threshold_sorts（后阈值排序）*

语句堆大小 (stmtheap)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	2048 [128 – 60 000]
计量单位	页 (4 KB)
分配时	对于预编译或联编期间的每个语句
释放时	当每个语句的预编译或联编完成时

语句堆在 SQL 语句的编译期间用作 SQL 编译程序的工作空间。此参数指定此工作空间的大小。

此区域并不总是处于分配状态，对于每个处理的 SQL 语句要进行分配和释放。注意: 对于动态 SQL 语句，将在程序执行期间使用此工作区; 而对于静态 SQL 语句，在联编进程而不是在程序执行期间使用此工作区。

建议: 在大多数情况下，此参数的缺省值都是可接受的。如果有很大的 SQL 语句，并且数据库管理程序在尝试优化语句时发出一个错误信息（语句太复杂），则应按固定增量（如 256 或 1024）增大此参数的值，直至消除此错误情况。

应用程序堆大小 (applheapsz)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	128 [16 – 60 000] 64 [16 – 60 000]（多分区）
计量单位	页 (4 KB)
分配时	当初始化代理程序为应用程序工作时
释放时	当代理程序完成应用程序要做的工作时

相关参数 第301页的『应用程序控制堆大小 (app_ctl_heap_sz)』

此参数定义要由数据库管理程序代表特定代理程序或子代理程序使用的可用的专用内存页的数目。

该堆是在为应用程序初始化代理程序或子代理程序时分配的。分配的量将是处理已经给予代理程序或子代理程序的请求所需要的最小量。由于代理程序或子代理程序需要更多的堆空间来处理较大的 SQL 语句，数据库管理程序将按需要分配内存，最大可为由此参数指定的最大值。

注: 在分区数据库环境中，使用应用程序控制堆 (*app_ctl_heap_sz*) 来存储代理程序和子代理程序的 SQL 语句的执行程序段副本。然而，SMP 子代理程序象其他所有环境中的代理程序一样使用 *applheapsz*。

建议: 如果您的应用程序接收到一个错误，指示在该应用程序堆中没有足够的存储器，则增加此参数的值。

在代理程序专用内存之外分配应用程序堆 (*applheapsz*)。

统计堆大小 (stat_heap_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 4384 [1096 – 524 288]

计量单位 页 (4 KB)

分配时 当启动 RUNSTATS 实用程序时

释放时 当 RUNSTATS 实用程序完成时

相关参数

- 第375页的『保存的高频值数目 (num_freqvalues)』
- 第376页的『列的分位数数目 (num_quantiles)』

此参数指示使用 RUNSTATS 命令收集统计信息时所用的堆的**最大**大小。

建议: 当不收集任何分布统计信息时，或当只收集相对较窄的表的分布统计信息时，缺省值较为合适。当收集分布统计信息时，**不**建议使用最小值，因为只有包含一列或两列的表才适合该堆。

应该根据正在收集的统计的列数调整此参数值。具有相对较少列数的窄表收集分布统计信息所需要的内存较少。具有很多列数的宽表则需要非常多的内存。如果

您正在收集非常宽而且需要大型统计堆的表的分布统计信息，可能希望在系统活动较少期间收集统计，这样就不会影响其他用户的内存需求。

查询堆大小 (query_heap_sz)

配置类型	数据库管理程序
适用于	• 带本地客户机和远程客户机的数据库服务器 • 带本地客户机的数据库服务器 • 带本地客户机和远程客户机的分区数据库服务器 • 带本地客户机的卫星数据库服务器
参数类型	可配置的
缺省值（范围）	1000 [2 – 524 288]
计量单位	页 (4 KB)
分配时	当应用程序（以本地方式或远程方式）与该数据库连接时
释放时	当应用程序与数据库断开连接时，或与实例断开连接时
相关参数	第313页的『应用程序支持层堆大小 (aslheapsz)』

此参数指定可为查询堆分配的**最大**内存容量。查询堆用来将每个查询存储在代理程序专用内存中。每个查询的信息由输入和输出 SQLDA、语句文本、SQLCA、程序包名、创建者、节号以及一致性令牌组成。提供此参数以防止应用程序不必要地消耗代理程序内部大量的虚拟内存。

查询堆也用作分配给分块游标的内存的内存源。此内存由游标控制块和全分辨输出 SQLDA 组成。

所分配的初始查询堆将与该应用程序支持层堆的大小相同，后者由 *aslheapsz* 参数指定。该查询堆大小必须大于或等于二 (2)，且必须大于或等于 *aslheapsz* 参数。如果此查询堆不够大，无法处理给定的请求，将重新分配它，使其达到该请求所需的大小（不超过 *query_heap_sz*）。如果此新查询堆超过 *aslheapsz* 的 1.5 倍，则在该查询结束时将重新分配该查询堆，使其大小为 *aslheapsz*。

建议：在大多数情况下，缺省值已足够使用。作为最小值，您应该将 *query_heap_sz* 设置为至少大于 *aslheapsz* 五倍的值。这将允许进行超过 *aslheapsz* 的查询，并为要在同一个给定时间打开的三个或四个分块游标提供附加内存。

如果您拥有很大的 LOB，您可能需要增加此参数的值，以便查询堆的大小足够容纳那些 LOB。

DRDA 堆大小 (drda_heap_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 128 [16 – 60 000]

计量单位 页 (4 KB)

分配时

- 每次 “DRDA 应用请求器” (AR) 与 DB2 数据库连接时， “DRDA 应用服务器” (AS) 分配一个 DRDA 堆。
- DB2 Connect 在每次与 DRDA AS 连接时分配一个 DA 堆。

释放时 当 DRDA AR 与数据库断开连接时

此参数指示要分配给 DB2 Connect 和 “DRDA 应用服务器支持功能” 使用的内存的页数。下列项目影响在此堆之外分配的内存容量:

- 应用程序打开的游标数
- 输入主变量数
- 选择列表中的项目数
- 输入数据和输出数据的大小
- 正在联编或准备的 SQL 语句的长度。

建议: 除非接收到一个指示 DRDA 堆不够的错误码，否则使用缺省值。

UDF 共享内存集大小 (udf_mem_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器

- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 256 [128 – 60 000]

计量单位 页 (4 KB)

分配时 当 UDF 启动时

释放时 当 UDF 完成时

此参数是防扰和非防扰“用户定义函数”(UDF) 所共有的。对于防扰 UDF，该参数指定数据库进程和 UDF 将要共享的缺省内存分配。在一个单分区数据库环境中，只有一个共享内存集。在分区数据库环境中，每个数据库分区服务器都有一个共享内存集，并且所有运行在该服务器上的应用程序代理程序和子代理程序都使用同一共享内存集。

对于非防扰 UDF，该参数指定专用内存集的大小。在一个单分区数据库环境中，堆是从专用内存中分配的。在一个分区数据库环境中，堆是从每个数据库分区服务器的“应用程序全局”内存中分配的，而代表该应用程序在该数据库分区服务器上运行的所有代理程序和子代理程序使用同一个共享内存集。

对于防扰 UDF 和非防扰 UDF，此内存用于将数据传送至 UDF 并传回至数据库。

如果应用程序中未使用 UDF，则不分配该内存。如果防扰和非防扰 UDF 都在同一个应用程序中运行，则分配两个内存：一个用于防扰 UDF，一个用于非防扰 UDF。

有关用户定义函数的详情，参见 *Application Development Guide* 和 *SQL Reference*。

建议：缺省设置应满足所有情况，向 UDF 传送 LOB 数据的情况除外。对于将 LOB 数据传送至 UDF 的情况，可能需要增大分配的内存量。应将此参数的值设置为至少比输入自变量和外部函数的结果的大小大两页。

注： UDF 的内存需求往往需要添加，所以应用程序中引用的 UDF 的数目将影响此参数的优化设置。

代理程序栈大小 (agent_stack_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

OS/2 64 [8 – 1000]

Windows NT 16 [8 – 1000]

计量单位 页 (4 KB)

分配时 当初始化代理程序为应用程序工作时

释放时 当代理程序完成应用程序要做的工作时

代理程序栈是 DB2 为每个代理程序分配的虚拟内存。当需要此内存以处理 SQL 语句时，才落实此内存。对于给定的一组应用程序，可使用此参数优化服务器的内存利用。与用于简单查询的空间相比，越复杂的查询将使用越多的栈空间。

此参数不适用于基于 UNIX 的平台。

建议：在大多数情况下，应可使用缺省堆栈大小。仅当您的环境包括很多非常复杂的查询时，才需要增加此参数的值。如果栈大小不够大，不能处理 SQL 语句，则将把一个错误信息记录至 db2diag.log 文件，并发出一个 SQL 代码。需要增加 *agent_stack_sz*，然后重新启动该数据库实例。

如果您的环境符合下列条件，也许能够减小堆栈大小以使更多地址空间可供其他客户机使用:

- 仅包含简单应用程序（例如灵巧的 OLTP ），其中从不会有复杂查询
- 需要相当大量的并行客户机（例如，超过 100 个）。

代理程序堆栈大小和并行客户机数是成反比的：堆栈大小越大，则可运行的并行客户机的可能数量就越少。这种情况是由于在 OS/2 和 Windows NT 平台上对地址空间的限制而造成的。例如，在 OS/2 上，假定有 400MB 的地址空间（虽然该数量取决于 `config.sys` 文件）。如果您将 *agent_stack_sz* 的值设置为 1MB，则您将无法获取多于 400 个的代理程序。（实际上，由于对地址空间的其他需求，如缓冲池，您获得的代理程序可能少得多。） 这就意味着如果您将 *maxagents* 设置为更大的值（例如，5000），您将永远达不到此限制。

落实的最小专用内存 (min_priv_mem)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 32 [32 – 112 000]

计量单位 页 (4 KB)

分配时 当启动数据库管理程序时

释放时 当停止数据库管理程序时

相关参数 『专用内存阈值 (priv_mem_thresh)』

此参数指定当启动 (db2start) 数据库管理程序实例时，数据库服务器将保留用作专用虚拟内存的页数。如果服务器需要更多专用内存，则需要时将设法从操作系统中获取更多专用内存。

此参数不适用于基于 UNIX 的系统。

建议: 使用缺省值。

如果想要将更多内存落实到数据库服务器，应只更改此参数的值。此操作将节省分配时间。然而应注意不要将该值设置得太大，因为这样会影响非 DB2 应用程序的性能。

专用内存阈值 (priv_mem_thresh)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 1296 [-1; 32 – 112 000]

32 [-1; 32 – 112 000] 在带本地客户机的卫星数据库服务器上

计量单位 页 (4 KB)

相关参数 第311页的『落实的最小专用内存 (min_priv_mem)』

此参数用于确定将处于分配状况、准备供启动的新代理程序使用的未用过的代理程序专用内存量。此参数不适用于基于 UNIX 的平台。

当终止一个代理程序而不是自动释放该代理程序使用的所有内存时，数据库管理程序将只释放*多余的内存分配*，这由下列公式确定:

```
分配的专用内存 -
(使用的专用内存 + priv_mem_thresh)
```

如果此公式产生负结果，将不采取任何操作。

下表是一个示例，举例说明何时分配和解除分配内存。此示例使用 100 作为 *priv_mem_thresh* 的任意设置。

操作说明	分配的内存	使用的内存
一些代理程序正在运行并已分配内存。	1000	1000
新代理程序已启动并使用 100 页内存。	1100	1100
一个使用 200 页内存的代理程序终止。(注意: 已释放 100 页内存，而保持有 100 页内存可分配以备将来使用。)	1000	900
一个使用 50 页的代理程序终止。(注意: 已释放 50 页内存，与现存代理程序使用的内存比较，仍有 100 页可分配的额外内存。)	950	850
启动一个新代理程序，并需要 150 页内存。(150 页中的 100 页已分配，数据库管理程序只需要为此代理程序分配 50 页附加内存。)	1000	1000

值 『-1』 将使此参数使用 *min_priv_mem* 参数的值。

建议: 当设置此参数时，您应考虑客户机连接／断开模式，以及同一台机器上其他进程的内存需求。

如果仅在一个短暂的时期内有很多客户机都与数据库并行连接，则高阈值将防止未使用的内存被解除落实，从而被其他进程使用。这种情况将导致内存管理混乱，从而会影响其他需要内存的进程。

如果并行客户机数围绕一个固定值频繁变动，则高阈值将帮助确保内存可用于客户机进程并减少分配内存和解除对内存的分配的额外开销。

代理程序 / 应用程序通信内存

下列参数影响为了在应用程序和代理程序进程之间传送数据而分配的内存容量:

- 『应用程序支持层堆大小 (aslheapsz)』
- 第314页的『客户机 I/O 块大小 (rqrioblk)』
- 第315页的『DOS 请求器 I/O 块大小 (dos_rqrioblk)』

有关此代理程序 / 应用程序共享内存如何与数据库管理程序分配的内存的剩余部分相关的信息，参见 第201页的『DB2 如何使用内存』。

应用程序支持层堆大小 (aslheapsz)

配置类型	数据库管理程序
适用于	• 带本地客户机和远程客户机的数据库服务器 • 带本地客户机的数据库服务器 • 带本地客户机和远程客户机的分区数据库服务器 • 带本地客户机的卫星数据库服务器
参数类型	可配置的
缺省值（范围）	15 [1 – 524 288]
计量单位	页 (4 KB)
分配时	当为本地应用程序启动数据库管理程序代理程序进程时
释放时	当终止数据库管理程序代理程序进程时
相关参数	第307页的『查询堆大小 (query_heap_sz)』

应用程序支持层堆表示本地应用程序和其关联的代理程序之间的通信缓冲区。此缓冲区被分配为每个已启动的数据库管理程序代理程序所共享的内存。

如果对数据库管理程序的请求或其关联的答复不适合该缓冲区，则该请求和答复将分成两个或更多的发送 - 接收对。应将此缓冲区的大小设置为可使用单个发送 - 接收对来处理大多数请求。请求的大小基于保存下列各项所需的存储器:

- 输入 SQLDA
- SQLVAR 中的所有相关数据
- 输出 SQLDA

- 一般不超过 250 个字节的其他字段。

除此通信缓冲区之外，此参数还用来确定打开分块游标时的 I/O 块大小。这个用于分块游标的内存是在应用程序专用地址空间之外分配的，所以应确定要分配给每个应用程序的最佳专用内存量。如果数据库客户机不能在应用程序专用内存之外为分块游标分配空间，则将打开非分块游标。

数据库管理程序将从本地应用程序发送的数据接收到从查询堆中分配的一组相邻内存中。*aslheapsz* 参数用于确定查询堆（用于本地和远程客户机）的初始大小。查询堆的最大大小由 *query_heap_sz* 参数定义。

建议: 如果您的应用程序的请求通常较小，并且该应用程序在内存受约束的系统上运行，则您可能希望减小此参数的值。如果您的查询一般都很大，需要多个发送和接收请求，并且您的系统不受内存约束，则您可能希望增加此参数的值。

使用如下公式计算 *aslheapsz* 的页数:

```
aslheapsz >= ( sizeof(input SQLDA)
             + sizeof(each input SQLVAR)
             + sizeof(output SQLDA)
             + 250 ) / 4096
```

还应考虑此参数对分游标的数目和潜在大小的影响。如果所传送的行的数目或大小较大（例如，如果数据量大于 4096 字节），则大行块可能获得更佳性能。然而，由于较大的记录块会增加每个连接的工作集内存大小，所以要加以折衷。

大记录块也可能导致比应用程序实际需要的更多的读取请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制读取请求数。有关 OPTIMIZE FOR 子句的详情，参见 第64页的『OPTIMIZE FOR n ROWS 子句』。

客户机 I/O 块大小 (rqrioblk)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 32 767 [4096 – 65 535]

计量单位 字节

分配时

- 当远程客户机应用程序对服务器数据库发出连接请求时
- 当打开分块游标时，在该客户机上打开附加块

释放时

- 当断开远程应用程序与服务器数据库的连接时
- 当关闭分块游标时

相关参数 『DOS 请求器 I/O 块大小 (dos_rqrioblk)』

此参数指定数据库服务器上远程应用程序及其数据库代理程序之间的通信缓冲区大小。当数据库客户机请求与远程数据库连接时，在客户机上分配此通信缓冲区。在数据库服务器上，起初分配一个大小为 32767 字节的通信缓冲区，直到建立连接从而服务器可确定客户机上 *rqrioblk* 的值为止。一旦服务器知道此值，如果客户机缓冲区不为 32767 个字节，则服务器将重新分配其通信缓冲区。

除了此通信缓冲区之外，此参数适用于确定打开分块游标时数据库客户机上 I/O 块大小。这个用于分块游标的内存是在应用程序专用地址空间之外分配的，所以应确定要分配给每个应用程序的最佳专用内存量。如果数据库客户机不能在应用程序专用内存之外为分块游标分配空间，则将打开非分块游标。

建议: 对于非分块游标，增大此参数的值的一个原因是：要由单个 SQL 语句发送的数据（例如，大对象数据）很大，以致缺省值不够。

还应考虑此参数对分游标的数目和潜在大小的影响。如果所传送的行的数目或大小较大（例如，如果数据量大于 4096 字节），则大行块可能获得更佳性能。然而，由于较大的记录块会增加每个连接的工作集内存大小，所以要加以折衷。

大记录块也可能导致比应用程序实际需要的更多的读取请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制读取请求数。有关 OPTIMIZE FOR 子句的详情，参见 第64页的『OPTIMIZE FOR n ROWS 子句』。

DOS 请求器 I/O 块大小 (dos_rqrioblk)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器

- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 4096 [4096 – 65 535]

计量单位 字节

分配时

- 当远程 DOS 或 Windows 3.1 客户机向服务器数据库发出连接请求时
- 当打开分块游标时，在该客户机上打开附加块

释放时

- 当断开远程应用程序与数据库的连接时
- 当关闭分块游标时

相关参数 第314页的『客户机 I/O 块大小 (rqrioblk)』

此参数指定数据库服务器上 DOS/Windows 3.1 应用程序与其数据库代理程序之间的通信缓冲区大小。此参数类似于 *rqrioblk* 参数，唯一不同的是它允许您为在 DOS/Windows 3.1 客户机中使用的块设置不同的值。在 DB2 配置文件中，您可以同时设置 *rqrioblk* 参数（用于 Windows 32 位、OS/2 和 UNIX 客户机）和 *dos_rqrioblk* 参数（用于 DOS 和 Windows 3.1 客户机）。

除了此通信缓冲区之外，此参数适用于确定打开分块游标时数据库客户机上 I/O 块大小。这个用于分块游标的内存是在应用程序专用地址空间之外分配的，所以应确定要分配给每个应用程序的最佳专用内存量。如果数据库客户机不能在应用程序专用内存之外为分块游标分配空间，则将打开非分块游标。

建议: 对于非分块游标，增大此参数的值的一个原因是: 要由单个 SQL 语句发送的数据（例如，大对象数据）很大，以致缺省值不够。

还应考虑此参数对分游标的数目和潜在大小的影响。如果所传送的行的数目或大小较大（例如，如果数据量大于 4096 字节），则大行块可能获得更佳性能。然而，由于较大的记录块会增加每个连接的工作集内存大小，所以要加以折衷。

大记录块也可能导致比应用程序实际需要的更多的读取请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制读取请求数。有关 OPTIMIZE FOR 子句的详情，参见 第64页的『OPTIMIZE FOR n ROWS 子句』。

数据库管理程序实例内存

下列参数影响在实例级分配和使用的内存:

- 『数据库系统监控程序堆大小 (mon_heap_sz)』
- 第318页的『目录高速缓存支持 (dir_cache)』
- 第319页的『审查缓冲区大小 (audit_buf_sz)』
- 第320页的『最大 Java 解释程序堆大小 (java_heap_sz)』

数据库系统监控程序堆大小 (mon_heap_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值(范围)

UNIX 56 [0 – 60 000]

OS/2 和 Windows NT 带本地客户机和远程客户机的数据库服务器和带本地客户机的卫星数据库服务器 24 [0 – 60 000]

OS/2 和 Windows NT 带本地客户机的数据库服务器 12 [0 – 60 000]

计量单位 页 (4 KB)

分配时 当用 *db2start* 命令启动数据库管理程序时

释放时 当用 *db2stop* 命令停止数据库管理程序时

相关参数 第398页的『缺省数据库系统监控程序开关 (dft_monswitches)』

此参数确定分配给数据库系统监控程序数据的内存量(以页计)。当进行数据库监控活动(如记录快照、调整监控程序开关、重新设置监控程序或激活事件监控程序)时,从监控程序堆分配内存。

值 0 阻止数据库管理程序收集数据库系统监控程序数据。

建议: 监控活动所需的内存量取决于监控应用程序（捕捉快照的应用程序或事件监控程序）的数目、设置了哪些开关以及数据库活动的级别。

下列公式提供监控程序堆必需的近似页数:

```
( 监控应用程序数 + 1 ) * ( 数据库数 * (800 + ( 存取的表数 * 20 ) + ( ( 连接的应用程序数
```

如果此堆中的可用内存都用尽，将发生下列其中一种情况:

- 当第一个应用程序与定义了此事件监控程序的数据库连接时，就会向 db2alert.log 和 db2diag.log 文件写入第 2 级的出错信息。
- 如果使用 SET EVENT MONITOR 语句动态启动的事件监控程序失败，则向您的应用程序返回一个错误码。
- 如果一个监控程序命令或 API 子例程失败，则向您的应用程序返回一个错误码。

目录高速缓存支持 (dir_cache)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） Yes [Yes; No]

分配时

- 当应用程序发出第一个连接请求时，分配专用高速缓存
- 当启动数据库管理程序实例时 (db2start)，分配共享高速缓存。

释放时

- 当应用程序进程终止时，释放专用高速缓存
- 当停止数据库管理程序实例时 (db2stop)，释放共享高速缓存。

通过将 *dir_cache* 设置为 『Yes』，将数据库、节点和 DCS 目录文件高速缓存到内存。使用目录高速缓存可减小连接成本，因为这样消除了目录文件 I/O 并将检索目录信息所需的目录搜索减至最少。有两种类型的目录高速缓存:

- 分配并用于每个应用程序进程的专用高速缓存，该高速缓存在应用程序正运行的机器上。
- 分配并用于一些内部数据库管理程序进程的共享高速缓存。

注： 只有专用高速缓存适用于受支持的 Windows 环境。

对于专用高速缓存，当应用程序发出第一个连接请求时，读取每个目录文件并将信息高速缓存在此应用程序的专用内存中。此高速缓存由应用程序进程用在后续连接请求上，并在应用程序进程的生命期内保持。如果专用高速缓存中找不到某个数据库，则搜索目录文件以获取信息，但未更新高速缓存。如果应用程序修改目录项，则该应用程序中的下一个连接将导致刷新此应用程序。将不刷新其他应用程序的专用高速缓存。当应用程序进程终止时，释放高速缓存。（要刷新命令行处理器对话所使用的目录高速缓存，发出 db2 terminate 命令。）

对于共享高速缓存，当启动 (db2start) 数据库管理程序实例时，读取每个目录文件并将信息高速缓存在共享内存中。此高速缓存由一些数据库管理程序进程使用，并在停止 (db2stop) 实例之前一直保持。如果此高速缓存中找不到某个目录项，则搜索目录文件以获取信息。在实例运行期内永不刷新此共享高速缓存。

建议：如果目录文件不经常更改且性能很重要，则使用目录高速缓存。

另外在远程客户机上，如果应用程序发出几个不同的连接请求，则目录高速缓存会很有益。在这种情况下，高速缓存减少单个应用程序必须读取目录文件的次数。

目录高速缓存也可以改进记录数据库系统监控程序快照的性能。另外，应显式引用快照调用中的数据库名，而不使用数据库别名。

注： 如果目录高速缓存打开以及在启动数据库管理程序后编目、取消编目、创建或卸放数据库，则在执行快照调用时可能发生错误。

审查缓冲区大小 (audit_buf_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 0 [0 – 65 000]

计量单位 页 (4 KB)

分配时 当启动 DB2 时

释放时 当停止 DB2 时

此参数指定审查该数据库时所用缓冲区的大小。有关审查设施的详情，参考*管理指南: 计划*中的 “审查 DB2 活动”。

此参数的缺省值是零 (0)。如果该值为零 (0)，则不使用该审查缓冲区。如果该值大于零 (0)，就会为审查缓冲区分配空间，以用于放置审查设施所生成的审查记录。该值乘以 4KB 页就是为审查缓冲区分配的空间容量。不能动态分配该审查缓冲区；即必须停止 DB2，然后重新启动它，此参数的新值才会生效。

通过将此参数从缺省值更改为大于零 (0) 的某个值，该审查设施将记录写入磁盘与执行生成审查记录的语句将异步进行。这使 DB2 性能比保留该参数值为零 (0) 时有所提高。该值为零 (0) 意味着，审查设施将记录写入磁盘与执行生成审查记录的语句将同步进行（同时进行）。审查期间进行同步操作会降低在 DB2 中运行的应用程序的性能。

最大 Java 解释程序堆大小 (java_heap_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 512 [0 - 4 096]

计量单位 页 (4 KB)

分配时 当 Java 应用程序启动时

释放时 当 Java 应用程序完成时

相关参数 第405页的『Java Development Kit 1.1 安装路径 (jdk11_path)』

此参数确定 Java 解释程序所使用的最大堆大小。

每个 DB2 进程有一个堆（在基于 UNIX 的平台上，每个代理程序或子代理程序一个堆；而在其他平台上，则每个实例一个堆），并且每个防护 UDF 和防护存储过程进程也有一个堆。在所有情况下，只有运行 Java UDF 或存储过程的代理程序或进程经常分配此内存。在分区数据库系统上，堆按数据库分区服务器的数目倍数增大。

锁定

下列参数影响在环境中如何管理锁定:

- 『检查死锁的时间间隔 (dlchktime)』
- 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』
- 第323页的『锁定超时 (locktimeout)』

另见第297页的『锁定列表的最大存储器 (locklist)』。

第43页的『锁定』对数据库管理程序如何使用锁定来维护数据完整性提供了一个整体概述。

检查死锁的时间间隔 (dlchktime)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 10 000（10 秒） [1000 – 600 000]

计量单位 毫秒

相关参数

- 第297页的『锁定列表的最大存储器 (locklist)』
- 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』

当连接至同一数据库的两个或多个应用程序无限制地等待一个资源时发生死锁。因为每个应用程序都持有对方执行所需要的资源，所以该等待永远得不到解决。

死锁检查间隔定义数据库管理程序检查所有与数据库连接的应用程序中间的死锁的频率。

注:

1. 在分区数据库环境中，此参数只适用于目录节点。
2. 在分区数据库环境中，死锁直到第二次重复出现之后才被标记。

建议: 增加此参数以降低检查死锁的频率，因此增加应用程序必须等待消除死锁的时间。

减小此参数会增大检查死锁的频率，从而减少应用程序必须等待死锁解决的时间，但会增加数据库管理程序检查死锁所花的时间。如果死锁间隔太小，可能会降低运行期性能，因为数据库管理程序频繁执行死锁检测。如果将此参数设置得较低以改善并行性，则应确保适当地设置 *maxlocks* 和 *locklist*，以避免不必要的锁定逐步升级，这种升级可能导致更多的锁定争用，由此产生更多死锁的情况。

逐步升级前锁定列表的最大百分比 (maxlocks)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 10 [1 – 100]

OS/2 和 Windows NT 22 [1 – 100]

计量单位 百分率

相关参数

- 第297页的『锁定列表的最大存储器 (locklist)』
- 第331页的『活动应用程序的最大数目 (maxappls)』

锁定逐步升级是用表锁定替换行锁定的过程，以减少列表中的锁定数。此参数定义应用程序持有的锁定列表的百分率，必须在数据库管理程序执行逐步升级之前填写该列表。当任何一个应用程序持有的锁定数达到总锁定列表大小的此百分比时，将发生该应用程序所持有锁定的锁定逐步升级。当锁定列表空间用尽时也会发生锁定逐步升级。

数据库管理程序通过浏览锁定列表以找到应用程序并查找有最多行锁定的表，确定要逐步升级哪些锁定。如果用单个表锁定替换这些锁定之后，不再超过 *maxlocks* 值，锁定逐步升级将停止。否则，将继续逐步升级，直到保持的锁定列表的百分率低于 *maxlocks* 的值。*maxlocks* 参数乘以 *maxappls* 参数不能小于 100。

建议: 设置 *maxlocks*时，应考虑锁定列表 (*locklist*) 的大小:

```
maxlocks = 100 *
           (每个应用程序 512 锁定
           * 每个锁定 32 字节
           * 2) / (locklist * 4096 字节)
```

此样本公式允许任何应用程序保持两倍于平均锁定数的锁定。

如果很少有应用程序并行运行，可以增加 *maxlocks*，因为在这种情况下，没有太多对锁定列表空间的争用。

可使用数据库系统监控程序帮助您跟踪和调整此配置参数。

有关详情，参见 *System Monitor Guide and Reference* 中的 *locks_held_top（保持的最大锁定数）*监控程序元素说明。

通过此参数控制锁定逐步升级对优化器是很重要的，因为优化器使用此参数确定存取路径。在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

锁定超时 (locktimeout)

配置类型 数据库

参数类型 可配置的

缺省值（范围） -1 [-1; 0 – 30 000]

计量单位 秒

相关参数

- 第297页的『锁定列表的最大存储器 (locklist)』
- 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』

此参数指定应用程序为获取一个锁定将等待的秒数。这可帮助避免应用程序全局死锁。

如果将此参数设置为 0，则不会等待锁定。在此情况下，如果请求时未提供任何锁定，则应用程序就会立即接收到 -911。

如果将此参数设置为 -1，则锁定超时检测关闭。在此情况下，将等待锁定（如果请求时无可用锁定），直到出现下列其中一种情况:

- 授予锁定
- 发生死锁。

建议: 在事务处理 (OLTP) 环境中，可以使用 30 秒的初始启动值。在一个只查询的环境中，您可以以一个较高的值开始。在这两种情况下，您都应该使用基准技术来调整此参数。

当使用 DataLinks Manager 时，如果您发现 DataLinks Manager (dlfm) 实例的 db2diag.log 中发生锁定超时，则应增大 *locktimeout* 的值。您还应当考虑增大 *locklist* 的值。

此值应该设置为能够快速检测由于异常情况如事务停止（可能是因为用户离开工作站）而发生的等待。您应该将此值设置得足够大，以便有效锁定请求不会因为峰值工作负荷（在该时期内需要更多的锁定等待）而超时。

可以使用数据库系统监控程序来帮助您跟踪一个应用程序（连接）经历的锁定超时的次数，或跟踪数据库检测到连接的所有应用程序超时情况的次数。有关详情，参见 *System Monitor Guide and Reference* 中描述的 *locks_timeouts*（锁定超时的数量）。

lock_timeout 监控程序元素的高值可能由下列情况导致:

- 此配置参数的值太低。
- 延长持有锁定的时间的应用程序（事务）。您可以使用数据库系统监控程序更进一步调查这些应用程序。
- 并行问题，它可能是由锁定升级（从行级别至表级别）引起的。有关详情，参见第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』和第297页的『锁定列表的最大存储器 (locklist)』。

有关使用此参数的详情，参见第50页的『锁定等待和超时』。

I/O 和存储器

下列参数可以影响与数据库操作相关的 I/O 和存储器成本:

- 『更改页阈值 (chngpgs_thresh)』
- 第325页的『异步页清除器数 (num_iocleaners)』
- 第326页的『I/O 服务器数目 (num_ioservers)』
- 第327页的『索引排序标志 (indexsort)』
- 第327页的『顺序检测标志 (seqdetect)』
- 第328页的『缺省预读取大小 (dft_prefetch_sz)』
- 第329页的『缺省 SMS 容器数 (numsegs)』
- 第329页的『表空间的缺省数据块大小 (dft_extent_sz)』
- 第329页的『扩充内存段大小 (estore_seg_sz)』
- 第330页的『扩充内存段数 (num_estore_segs)』

更改页阈值 (chngpgs_thresh)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 60 [5 – 99]

计量单位 百分率

相关参数 『异步页清除器数 (num_iocleaners)』

异步页清除器在数据库代理程序需要缓冲池中的空间之前将更改的页从缓冲池中写入磁盘。这意味着代理程序将不必等待更改的页写出就能读取页，这样应用程序的事务应运行得更快。

可使用此参数指定更改的页的级别（百分率），如果异步页清除器当前不活动，则将从该级别启动。当启动页清除器时，页清除器将构建要写入磁盘的页的列表。一旦页清除器已将这些页写入磁盘，则将再次变为不活动的，并等待下一个触发器来启动。

在只读（例如查询）环境中，不使用这些页清除器。

建议: 对于具有繁重的更新事务工作负荷的数据库，您通常可以将该参数值设置为等于或小于缺省值，以确保在缓冲池中有足够的清洁页面。如果数据库有少量很大的表，则大于缺省值的百分率有助于提高性能。

异步页清除器数 (num_iocleaners)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 1 [0 – 255]

计量单位 计数器

相关参数

- 第290页的『缓冲池大小 (buffpage)』
- 第324页的『更改页阈值 (chngpgs_thresh)』

此参数允许您指定数据库的异步页清除器数。在数据库代理程序需要缓冲池的空间之前，这些页清除器将更改的页从缓冲池写至磁盘。这意味着这些代理程序将不必等待更改的页写出就能读取一个页。因此，您的应用程序的事务应运行得更快。

如果您将该参数设置为零 (0)，则未启动页清除器，因此，数据库代理程序将执行将所有页从缓冲池写入磁盘的操作。此参数对存储在许多物理存储设备上的数据库的性能有重大影响，因为在这种情况下很可能其中一个设备将是空闲的。如果未配置页清除器，应用程序可能会遇到周期性日志已满的情况。

如果一个数据库的应用程序主要由更新数据的事务组成，则增加清除器的数目将会提高运行速度。增加页清除器也将减少从软故障（如断电）恢复的时间，因为磁盘上数据库的内容在任何给定的时间都是较新的。

建议：当设置此参数的值时，考虑下列因素：

- 应用程序类型
  - 如果它是一个不会有更新的只查询数据库，则将此参数设置为零 (0)。例外情况是查询工作负荷是否导致创建许多 TEMP 表（您可以使用解释实用程序来确定这点）。
  - 如果事务是对数据库运行的，则将此参数的值设置在 1 与用于数据库的物理存储设备数之间。
- 工作负荷

  具有较高更新事务比率的环境可能需要配置更多页清除器。
- 缓冲池大小 (*buffpage*)

  含有大缓冲池的环境可能也需要配置更多的页清除器。

可使用数据库系统监控程序来帮助调整此配置参数，该数据库系统监控程序使用事件监控程序收集的有关缓冲池写入活动的信息：

- 如果下列两种情况都为真，则可以减小该参数：
  - *pool_data_writes* 大约等于 *pool_async_data_writes*
  - *pool_index_writes* 大约等于 *pool_async_index_writes*。
- 如果下列任何一种情况为真，则应增加该参数：
  - *pool_data_writes* 比 *pool_async_data_writes* 大很多
  - *pool_index_writes* 比 *pool_async_index_writes* 大很多

有关详情，参见 *System Monitor Guide and Reference* 中下列监控程序元素的说明：

- *pool_data_writes* （缓冲池数据写入）
- *pool_index_writes* （缓冲池索引写入）
- *pool_async_data_writes* （缓冲池异步数据写入）
- *pool_async_index_writes* （缓冲池异步索引写入）。

I/O 服务器数目 (num_ioservers)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	3 [1 – 255] 1 [1 – 255] on 带本地客户机的卫星数据库服务器
计量单位	计数器

分配时 当应用程序连接至数据库时

释放时 当应用程序从数据库断开时

相关参数

- 第328页的『缺省预读取大小 (dft_prefetch_sz)』
- 『顺序检测标志 (seqdetect)』

用 I/O 服务器代表数据库代理程序从而通过实用程序（如备份和复原）执行预读取 I/O 和异步 I/O。此参数指定用于数据库的 I/O 服务器的数目。任何时候，对一个数据库执行预读取和实用程序的 I/O 数不能超过此数目。I/O 服务器在它所启动的 I/O 操作处理期间等待。非预读取 I/O 直接由数据库代理程序调度，所以不受 *num_ioservers* 约束

建议: 为了最大限度地利用系统中的所有 I/O 设备，一个优选值通常是该数据库驻留的物理设备的数目加 1 或 2。最好配置额外的 I/O 服务器，因为每个 I/O 服务器都有相关的最小额外开销，而且任何未使用的 I/O 服务器都将保持空闲。

有关详情，参见第214页的『将数据预读取至缓冲池』和第216页的『配置 I/O 服务器启用预读取和并行 I/O』。

索引排序标志 (indexsort)

配置类型 数据库

参数类型 可配置的

缺省值（范围） Yes [Yes; No]

此参数指示在创建索引期间是否将对索引关键字进行排序。通过首先进行排序，增强创建索引的性能，特别对于群集比率或群集因子较低的索引来说尤其如此。如果通过排序创建索引，查询的性能也可能更优。这一性能增强的成本是排序需要更多的磁盘空间，所需要的空间可能为创建索引而不执行初始排序的空间的两倍。

建议: 除非没有足够的磁盘空间，否则使用缺省设置（『Yes』）。注意: 此排序所需要的磁盘空间大致与从表中 SELECT 有 ORDER BY 子句的列作为索引的列所需要的空间相等。

如果您有对称多处理器 (SMP) 环境，并指定此参数的值为 『No』，则在创建索引期间不使用在 SMP 环境中可能进行的多重处理。

顺序检测标志 (seqdetect)

配置类型 数据库

参数类型 可配置的

缺省值（范围） Yes [Yes; No]

相关参数 『缺省预读取大小 (dft_prefetch_sz)』

数据库管理程序可以监控 I/O，如果正在进行顺序页读取，则数据库管理程序还可激活 I/O 预读取。这种类型的顺序预读取称为*顺序检测*。可以使用 *seqdetect* 配置参数来控制数据库管理程序是否应执行顺序检测。

如果此参数设置为 『no』，则仅当数据库管理程序知道预读取将有用时才会发生预读取。例如，表排序、表扫描或列表预读取。

建议: 在大多数情况下，应使用此参数的缺省值。仅当其他调整都不能解决严重的查询性能问题时，才尝试关闭顺序检测。

缺省预读取大小 (dft_prefetch_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 32 [0 — 32 767]

OS/2 和 Windows NT
16 [0 — 32 767]

计量单位 页

相关参数

- 第329页的『表空间的缺省数据块大小 (dft_extent_sz)』
- 第326页的『I/O 服务器数目 (num_ioservers)』

当创建表空间时，可任意地指定 `PREFETCHSIZE n`，其中 `n` 为数据库管理程序在执行预读取时将读取的页数。如果不在 CREATE TABLESPACE 语句上指定预读取大小，则数据库管理程序使用由此参数给定的值。

有关详情，参见第214页的『将数据预读取至缓冲池』。

建议: 使用系统监控工具，可以确定当系统等待 I/O 时您的 CPU 是否空闲。如果未定义正使用的表空间的预读取大小，则增大此参数的值可能有帮助。

此参数提供整个数据库的缺省值，可能并不适合数据库中的所有表空间。例如，值 32 可能适合数据块大小为 32 页的表空间，但不适合数据块大小为 25 表空间。理想情况下，应显式设置每个表空间的预读取大小。

为了帮助将用缺省数据块大小 (*dft_extent_sz*) 定义的表空间的 I/O 降至最低，应该将此参数设置为 *dft_extent_sz* 参数值的一个因子或整数倍。例如，如果 *dft_extent_sz* 参数是 32，可将 *dft_prefetch_sz* 设置为 16（32 的因子）或 64（32 的整数倍）。如果预读取大小是该数据块大小的一个倍数，则数据库管理程序可以并行方式执行 I/O，其前提为下列情况为真:

- 正在预读取的范围在不同的物理设备上
- 配置了多个 I/O 服务器 (*num_ioservers*)。

缺省 SMS 容器数 (numsegs)

配置类型 数据库

参数类型 资料性

计量单位 计数器

此参数指示将在缺省表空间中创建的容器的数目，它仅适用于 SMS 表空间。此参数将显示在创建数据库时使用的信息，不管此参数在 CREATE DATABASE 命令上是显式还是隐式指定的。CREATE TABLESPACE 语句**不**以任何方式来使用此参数。

有关详情，*参考管理指南: 计划*中的 “数据库物理目录”。

表空间的缺省数据块大小 (dft_extent_sz)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 32 [2 – 256]

计量单位 页

相关参数 第328页的『缺省预读取大小 (dft_prefetch_sz)』

在创建表空间时，可任意地指定 `EXTENTSIZE n`，其中 `n` 为数据块大小。如果不在 CREATE TABLESPACE 语句上指定数据块大小，则数据库管理程序使用由此参数给定的值。

有关详情，*参考管理指南: 计划*中的 “设计和选择表空间”。

建议: 在多数情况中，应在创建表空间时明确指定数据块大小。在选择此参数的值之前，应了解如何显式选择 CREATE TABLESPACE 语句的。有关详情，参见第80页的『表空间对查询优化的影响』。

扩充内存段大小 (estore_seg_sz)

配置类型 数据库

参数类型	可配置的
缺省值（范围）	16 000 [0 – 1 048 575]
计量单位	页
相关参数	『扩充内存段数 (num_estore_segs)』

此参数指定数据库中每个扩充内存段中的页数。设置此配置参数时要考虑从属于平台方面的因素。

建议: 此参数仅当扩充内存可用并且按 *num_estore_segs* 参数所示使用时才有效。当指定每个扩充内存段中要使用的页数时，也应通过复查并修改 *num_estore_segs* 参数来考虑扩充内存段数。有关扩充内存的详情，参见第234页的『扩充内存』。

扩充内存段数 (num_estore_segs)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	0 [0 – 214 7483 647]
相关参数	第329页的『扩充内存段大小 (estore_seg_sz)』

此参数指定可供数据库使用的扩充内存段的数目。

缺省值为无扩充内存段。

建议: 仅当您的平台环境所具有的内存超过最大地址空间并且您希望使用此内存时，才使用此参数来利用扩充内存段。当指定段数时，也应通过复查与修改 *estore_seg_sz* 参数来考虑每个段的大小。

当同时设置 *num_estore_segs* 和 *estore_seg_sz* 两个配置参数时，应通过 CREATE/ALTER BUFFERPOOL 语句指定哪些缓冲池将使用扩充内存 有关扩充内存的详情，参见第234页的『扩充内存』。

代理程序

下列参数会影响可并行运行并达到最优性能的应用程序的数目:

- 第331页的『活动应用程序的最大数目 (maxappls)』
- 第332页的『活动应用程序的平均数 (avg_appls)』
- 第333页的『每个应用程序可打开的数据库文件的最大数目 (maxfilop)』
- 第334页的『每个应用程序可打开的最大总文件数 (maxtotfilop)』
- 第335页的『代理程序优先级 (agentpri)』

- 第336页的『代理程序的最大数目 (maxagents)』
- 第337页的『并行代理程序的最大数目 (maxcagents)』
- 第338页的『最大协调代理程序数 (max_coordagents)』
- 第339页的『逻辑代理程序的最大数目 (max_logicagents)』
- 第339页的『代理程序池大小 (num_poolagents)』
- 第340页的『池中初始代理程序数 (num_initagents)』

活动应用程序的最大数目 (maxappls)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 40 [1 – 64 000]

OS/2 和 Windows NT 带本地客户机和远程客户机的数据库服务器 20 [1 – 64 000]

OS/2 和 Windows NT 带本地客户机的数据库服务器 10 [1 – 64 000]

计量单位 计数器

相关参数

- 第336页的『代理程序的最大数目 (maxagents)』
- 第338页的『最大协调代理程序数 (max_coordagents)』
- 第322页的『逐步升级前锁定列表的最大百分比 (maxlocks)』
- 第297页的『锁定列表的最大存储器 (locklist)』
- 第332页的『活动应用程序的平均数 (avg_appls)』

此参数指定可与一个数据库连接（本地和远程）的并行应用程序的最大数目。因为每个与数据库连接的应用程序都导致分配一些专用内存，所以允许大量并行应用程序可能将使用更多内存。

此参数的值必须等于或大于连接的应用程序数，加上可能正在同时完成一个两阶段落实或回滚的相同应用程序的数目之和。然后将此总和与在任何时刻可能存在的预期不确定事务数相加。有关不确定事务的详情，*参考管理指南: 计划*中的“在两阶段落实期间校正问题”。

当应用程序试图与数据库连接，但是已经达到了 *maxappls* 时，会向该应用程序返回一个错误，指示已有最大数量的应用程序与数据库连接。

当有更多应用程序使用 DataLinks Manager 时，应增大 *maxappls* 的值。使用如下公式来计算您需要的值:

```
<maxappls> = 5 * (节点数) + (使用 DataLinks Manager 的活动应用程序数
                  的峰值)
```

DataLinks Manager 支持的最大值是 2 000。

在分区数据库环境中，这是在一个数据库分区并行活动的应用程序的最大数目。此参数限制对于数据库分区服务器上的数据库分区活动的应用程序数，不管该服务器是不是应用程序的协调程序节点。分区数据库环境中的目录节点要求 *maxappls* 的值应比该参数在其他类型的环境中的值高，因为在分区数据库环境中，每个应用程序都需要与该目录节点连接。

建议: 增加此参数的值而不降低 *maxlocks* 参数或增加 *locklist* 参数的值，可能导致您达到锁定的数据库限制 (*locklist*) ，而不是应用程序限制，从而产生扩散性锁定逐步升级问题。

在一定程度上，*maxagents* 也控制应用程序的最大数量。如果有可用的连接 (*maxappls*) 以及可用的代理程序 (*maxagents*)，则应用程序只能与该数据库连接。此外，应用程序的最大数目也受 *max_coordagents* 配置参数控制，因为如果达到 *max_coordagents*，则不能启动任何新的应用程序（即，协调代理程序）。

活动应用程序的平均数 (avg_appls)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 1 [1 – maxappls]

计量单位 计数器

相关参数

- 第 331 页的『活动应用程序的最大数目 (maxappls)』

SQL 优化器使用此参数来帮助估计在运行期内将有多少缓冲池可用于选择的存取方案。增大此参数的值可以影响优化器为查询选择的存取方案，从而在使用缓冲池时更为谨慎。

建议: 当在多用户环境特别是使用复杂查询和大缓冲池的环境中运行 DB2 时，您可能希望 SQL 优化器知道多个查询用户正在使用您的系统，以便该优化器应该在判断缓冲池的可用性时更保守。

在设置此参数时，应估计通常使用数据库的大型查询应用程序的数目。此估计应排除所有小型的 OLTP 应用程序。如果估算此数有困难，可以乘以下列数:

- 对数据库运行的所有应用程序的平均数。数据库系统监控程序可提供关于任何给定时间的应用程序数的信息，从而可以使用采样技术来计算一段时间内的平均值。来自数据库系统监控程序的信息包括 OLTP 和非 OLTP 应用程序。
- 对于大型查询应用程序的百分率的估计。

正如调整影响优化器的其他配置参数一样，应以较小的增量调整此参数。这样使您能够将路径选择差别减至最小。

在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

每个应用程序可打开的数据库文件的最大数目 (maxfilop)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX 64 [2 – 1950]

OS/2 和 Windows NT 64 [2 – 32 768]

计量单位 计数器

相关参数

- 第334页的『每个应用程序可打开的最大总文件数 (maxtotfilop)』
- 第331页的『活动应用程序的最大数目 (maxappls)』

此参数指定对于每个数据库代理程序可打开的文件句柄的最大数目。如果打开一个文件导致超过此值，则关闭此代理程序正在使用的一些文件。如果 *maxfilop* 太小，为了不超过此限制，打开和关闭文件的额外开销将变得极大，且可能降低性能。

在数据库管理程序与操作系统交互作用时，将 SMS 表空间和 DMS 表空间文件容器作为文件来处理，且需要文件句柄。与一个 DMS 文件表空间所用的容器数相比，SMS 表空间通常要使用更多的文件。因此，如果您使用的是 SMS 表空间，则您将为此参数设置的值应大于为 DMS 文件表空间设置的值。

也可以通过将平均每个代理程序的句柄数限制为一个特定的数，使用此参数来确保数据库管理程序使用的文件句柄总数不超过操作系统限制；实际数将根据并行运行的代理程序数而变化。

每个应用程序可打开的最大总文件数 (maxtotfilop)

配置类型	数据库管理程序
适用于	• 带本地客户机和远程客户机的数据库服务器 • 带本地客户机的数据库服务器 • 带本地客户机和远程客户机的分区数据库服务器 • 带本地客户机的卫星数据库服务器
参数类型	可配置的
缺省值（范围）	16 000 [100 – 32 768]
计量单位	计数器
相关参数	第333页的『每个应用程序可打开的数据库文件的最大数目 (maxfilop)』

此参数定义所有代理程序和其他在单个数据库管理程序实例中执行的线程可打开的文件的最大数目。如果打开一个文件导致超过此值，则将有一个错误信息返回至您的应用程序。

注: 此参数不适用于基于 UNIX 的平台。

建议: 当设置此参数时，应考虑可用于数据库管理程序实例中每个数据库的文件句柄数。要估计此参数的上限:

1. 使用如下公式，计算可为该实例中的每个数据库打开的文件句柄最大数目:

   ```
   maxappls * maxfilop
   ```

2. 计算以上结果的总和并验证该值不超过该参数最大值。

如果创建了一个新数据库，应重新估计此参数的值。

您还应该使用如下公式，来验证在您的系统上可能使用的文件句柄总数未超过系统最大值:

```
  (机器上所有实例的 maxtotfilop 之和)
+ (其他应用程序必需的文件句柄估计数)
<= 65535
```

代理程序优先级 (agentpri)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

AIX -1 [41 - 125]

其他 UNIX
-1 [41 - 128]

Windows NT
-1 [0 - 6]

OS/2 -1 [200 - 231; 300 - 331; 400 - 431]

此参数通过操作系统调度程序来控制给予所有代理程序和其他数据库管理程序实例进程和线程的优先级。在分区数据库环境中，这也包括协调和并行代理程序、并行系统控制器以及 FCM 精灵程序。此优先级确定如何将 CPU 时间给予与在该机器上运行的其他进程和线程有关的 DB2 进程、代理程序和线程。当将该参数设置为 -1 时，不采取任何特殊操作，并且以操作系统调度所有进程和线程的正常方式调度数据库管理程序。当将该参数设置为一个不为 -1 的值时，数据库管理程序将在该参数值设置为静态优先级的情况下创建其进程和线程。因此，此参数允许您控制数据库管理程序进程和线程在机器上执行时所用的优先级。

可使用此参数来增加数据库管理程序吞吐量。用于设置此参数的值取决于数据库管理程序正在哪个操作系统上运行。例如，在基于 UNIX 的环境中，低数值代表高优先级。当将该参数设置为在 41 和 125 之间的一个值时，数据库管理程序在该参数的值设置为 UNIX静态优先级的情况下创建其代理程序。这在基于 UNIX 的环境中很重要，因为低数值代表数据库管理程序的高优先级，但其他进程（包括应用程序和用户）可能因为不能获取足够的 CPU 时间而遇到延迟。应该使此参数的设置与机器上其他预期的活动相平衡。

在 OS/2 环境中，高数值代表高优先级。

建议: 最初应使用缺省值。此值提供了对其他用户／应用程序的响应时间与数据库管理程序吞吐量之间关系的很好的折衷办法。

如果关心的是数据库性能，可以使用基准技术来确定此参数的最佳设置。提高数据库管理程序的优先级时应仔细，因为可能会大大降低其他用户进程的性能，特别是当 CPU 利用率很高的时候更是如此。提高数据库管理程序进程和线程的优先级可显著地提高性能。

注: 如果在基于 UNIX 的平台上您将此参数设置为非缺省值，则不能使用控制器来改变代理程序的优先级。

代理程序的最大数目 (maxagents)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 200 [1 – 64 000]

400 [1 – 64 000] 在带本地客户机和远程客户机的分区数据库服务器上

10 [1 – 64 000] 在带本地客户机的卫星数据库服务器上

计量单位 计数器

相关参数

- 第331页的『活动应用程序的最大数目 (maxappls)』
- 第337页的『并行代理程序的最大数目 (maxcagents)』
- 第338页的『最大协调代理程序数 (max_coordagents)』
- 第342页的『DARI 进程的最大数目 (maxdari)』
- 第311页的『落实的最小专用内存 (min_priv_mem)』
- 第339页的『代理程序池大小 (num_poolagents)』

此参数指示可在任何给定时间接受应用程序请求的数据库管理程序代理程序（无论是协调代理程序还是子代理程序）的最大数目。如果您想限制协调代理程序数，使用 *max_coordagents* 参数。

此参数可在内存受约束的环境中用来限制数据库管理程序使用的内存总量，因为每个附加代理程序都需要附加内存。

建议: *maxagents* 的值至少应为每个数据库中允许同时存取的 *maxappls* 的值之和。如果数据库数大于 *numdb* 参数，则最安全的过程是使用具有 *maxappls* 的最大值的 *numdb* 的产品。

每个附加代理程序都需要一些在数据库管理程序启动时分配的资源额外开销。

并行代理程序的最大数目 (maxcagents)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 (*max_coordagents*) [-1; 1 – *max_coordagents*]

计量单位 计数器

相关参数

- 第331页的『活动应用程序的最大数目 (maxappls)』
- 第336页的『代理程序的最大数目 (maxagents)』
- 第338页的『最大协调代理程序数 (max_coordagents)』

可并行执行一个数据库管理程序事务的数据库管理程序协调程序代理的最大数目。此参数用来控制并行应用程序活动高峰期内系统上的负荷。例如，您可以让一个系统需要大量连接但只将有限内存量用于这些连接。在并行活动高峰期可能会导致过度的操作系统调页的环境中调整此参数可能会很有用。

此参数不限制可与数据库连接的应用程序的数目。此参数只限制在任何时间数据库管理程序可并行处理的数据库管理程序代理程序的数目，从而限制在处理高峰期内系统资源的使用。

值 −1 指示该限制是 *max_coordagents*。

建议: 在大多数情况下，此参数的缺省值将是可接受的。在应用程序的高并行性导致问题的情况下，可以使用基准测试来调整此参数以优化性能。

最大协调代理程序数 (max_coordagents)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 (*maxagents* — *num_initagents*)

[-1, 0–*maxagents*]

对于分区数据库环境和将 *intra_parallel* 设置为『Yes』的环境，缺省值为 *maxagents* - *num_initagents*; 否则，缺省值为 *maxagents*。这确保了在非分区数据库环境中，*max_coordagents* 总是等于 *maxagents*，除非将系统配置为分区内并行性。

如果没有分区数据库环境，且未启用 *intra_parallel* 参数，则 *max_coordagents* 必须等于 *maxagents*。

相关参数

- 第340页的『池中初始代理程序数 (num_initagents)』
- 第339页的『代理程序池大小 (num_poolagents)』
- 第336页的『代理程序的最大数目 (maxagents)』
- 第395页的『启用分区内并行性 (intra_parallel)』

此参数确定在分区或非分区数据库环境中，可同时在一个服务器上存在的最大协调代理程序数。

连接至数据库或连接至实例的每个本地或远程应用程序各获得一个协调代理程序。需要连接实例的请求包括 CREATE DATABASE、DROP DATABASE 以及数据库系统监控程序命令。

逻辑代理程序的最大数目 (max_logicagents)

配置类型 数据库管理程序

参数类型 可配置的

缺省值（范围） -1 (*max_coordagents*) [-1; *max_coordagents* — 64 000]

此参数控制可以与实例相连的最大应用程序数。通常，每个应用程序都被指定了一个协调代理程序。代理程序简化了应用程序与数据库之间的操作。当使用此参数的缺省值时，不激活集中器功能部件。因此，每个代理程序使用它自己的专用内存来运行，并与其他代理程序共享数据库管理程序和数据库全局资源，如缓冲池。当将此参数设置为大于缺省值的值时，激活集中器功能部件。集中器的目的是将每个客户机应用程序所使用的服务器资源减少到 DB2 Connect 网关可以处理超过 10 000 个客户机连接这样的程度。

值 -1 指示限制为 *max_coordagents*。

代理程序池大小 (num_poolagents)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 [-1, 0 — *maxagents*]

使用缺省值时，对于具有非分区数据库和本地客户机的服务器来说，其值为 *maxagents*/50 或 *max_querydegree* 之中较大的一个。

使用缺省值时，对于具有非分区数据库及本地和远程客户机的服务器来说，其值为 *maxagents*/50 x *max_querydegree* 或 *maxagents* - *max_coordagents* 中较大的一个。

使用缺省值时，对于数据库分区服务器来说，其值为 *maxagents*/10 x *max_querydegree* 或 *maxagents* - *max_coordagents* 之中较大的一个。

相关参数

- 『池中初始代理程序数 (num_initagents)』
- 第336页的『代理程序的最大数目 (maxagents)』
- 第394页的『最大查询并行度 (max_querydegree)』
- 第338页的『最大协调代理程序数 (max_coordagents)』

此参数是关于您想要代理程序池增长至多大的准则（并替换在 DB2 版本 2 中使用的 *max_idleagents* 参数）。

代理程序池包含子代理程序和空闲代理程序。空闲代理程序可用作并行子代理程序或协调代理程序。如果创建的代理程序超过此参数值指示的数目，则当这些代理程序执行完其当前请求时，将被终止，而不是返回至池中。

如果此参数的值为 0，则按需要创建代理程序，并且当这些代理程序执行完其当前请求时，可能被终止。如果该值为 *maxagents*，并且该池充满关联的子代理程序，则服务器不能用作协调程序节点，因为不能创建新的协调代理程序。

建议： 如果运行其中有很少应用程序并行连接的决策支持环境，则将 *num_poolagents* 设置为较小的值以避免有充满空闲代理程序的代理程序池。

如果运行有很多应用程序并行连接的事务处理环境，则增加 *num_poolagents* 的值以节省与频繁创建和终止代理程序关联的成本。

池中初始代理程序数 (num_initagents)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 0 [0 — *num_poolagents*]

相关参数

- 第336页的『代理程序的最大数目 (maxagents)』

- 第339页的『代理程序池大小 (num_poolagents)』
- 第338页的『最大协调代理程序数 (max_coordagents)』

此参数确定在 DB2START 时在代理程序池中创建的初始空闲代理程序数。

数据库应用程序远程接口 (DARI)

下列参数可以影响数据库应用程序远程接口 (DARI) 应用程序:

- 『保存 DARI 进程指示符 (keepdari)』
- 第342页的『DARI 进程的最大数目 (maxdari)』
- 第343页的『用 JVM 初始化 DARI 进程 (initdari_jvm)』
- 第344页的『存储池中防护 DARI 进程的初始数目 (num_initdaris)』

注: 术语 DARI 指存储过程。

保存 DARI 进程指示符 (keepdari)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 (范围) Yes [Yes; No]

相关参数 第342页的『DARI 进程的最大数目 (maxdari)』

此参数指示在 DARI 调用完成后是否保存 DARI 进程。DARI 进程是作为独立系统实体创建的, 以便将用户编写的 DARI 代码与数据库管理程序的代理程序进程分开。此参数仅在数据库服务器上可用。

如果将 *keepdari* 设置为 *no*, 则会为每次 DARI 调用创建一个新的 DARI 进程, 然后将它破坏。如果将 *keepdari* 设置为 *yes*, 则后续的 DARI 调用都重复使用一个 DARI 进程。当停止数据库管理程序时, 将终止所有未完成的 DARI 进程。

将此参数设置为 *yes* 将会导致数据库管理程序消耗其他系统资源以用于激活的每个 DARI 进程，直到使用资源的进程数达到 *maxdari* 参数中包含的值为止。这仅当没有可用来处理后续 DARI 调用的现存 DARI 进程时才为真。如果将 *maxdari* 设置为 0，则忽略此参数。

建议: 在 DARI 请求的数量相对于非 DARI 请求的数量很大的环境中，且系统资源不受约束，就可以将此参数设置为 *yes*。这将通过避免创建初始 DARI 进程这一额外开销而改进 DARI 性能，因为将使用现存的 DARI 进程来处理调用。

例如，在 OLTP 借贷银行事务应用程序中，用于执行每个事务的代码可在一个存储过程中执行，该存储过程是 DARI 进程的一部分。在此应用程序中，主工作负荷是在 DARI 进程外部执行的。如果将此参数设置为 *no*，则每个事务都会因创建新的 DARI 进程而带来额外开销，从而导致性能显著降低。但是，如果将此参数设置为 *yes*，则每个事务将尝试使用现存的 DARI 进程，这就避免了此额外开销。

DARI 进程的最大数目 (maxdari)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 (*max_coordagents*) [-1; 0 – *max_coordagents*]

计量单位 计数器

相关参数

- 第336页的『代理程序的最大数目 (maxagents)』
- 第341页的『保存 DARI 进程指示符 (keepdari)』
- 第344页的『存储池中防护 DARI 进程的初始数目 (num_initdaris)』
- 第338页的『最大协调代理程序数 (max_coordagents)』

此参数指示可驻留在数据库服务器上的 DARI 进程的最大数目。一旦达到此限制，就不可能再调用新的 DARI 请求。此参数仅在数据库服务器上可用。

每个协调代理程序只能有一个 DARI 进程活动，以便 DARI 进程的最大数目也受协调代理程序的最大数目 (*max_coordagents*) 限定。

建议：如果您的环境具有在数据库管理程序内使用 DARI 设施的能力，则此参数可以用于确保有适当数量的 DARI 进程可用，以处理在数据库管理程序内任何时间进行的 DARI 调用。

如果将该参数设置为 −1，则 DARI 进程的最大数目将与在 *max_coordagents* 参数中设置的值相同。

如果您发现缺省值不适合于您的环境（因为将数量不恰当的系统资源分配给了 DARI 进程，这影响了数据库管理程序的性能），则下列操作可能有助于提供调整此参数的起点:

```
axdari = 允许在一个时间执行 DARI 调用的应用程序数
```

如果将 *keepdari* 设置为 *yes*，则即使是已经处理了 DARI 调用并返回至代理程序之后，而创建的每个 DARI 进程仍将继续存在并使用系统资源。

如果您的环境受到严格的约束，不能提供与 DARI 相关的进程资源，则可以通过将此参数设置为零 (0) 来禁用 DARI。

用 JVM 初始化 DARI 进程 (initdari_jvm)

配置类型　数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型　可配置的

缺省值（范围）　No [Yes; No]

相关参数

- 第342页的『DARI 进程的最大数目 (maxdari)』
- 第344页的『存储池中防护 DARI 进程的初始数目 (num_initdaris)』
- 第341页的『保存 DARI 进程指示符 (keepdari)』

此参数指示每个防护 DARI 进程启动时是否装入 “Java 虚拟机” (JVM)。此参数将缩短防护 Java 存储过程的初始启动时间，尤其是与 *num_initdaris* 参数一起使用时。如果非 Java 防护存储过程不需要 JVM，此参数可能会增加这类过程的初始装入时间。

存储池中防护 DARI 进程的初始数目 (num_initdaris)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 0 [0 — *maxdari*]

相关参数

- 第342页的『DARI 进程的最大数目 (maxdari)』
- 第343页的『用 JVM 初始化 DARI 进程 (initdari_jvm)』
- 第341页的『保存 DARI 进程指示符 (keepdari)』

此参数指示在 DB2START 时在 DARI 存储池中创建的空闲防护 DARI 的初始数目。设置此参数将缩短防护存储过程的初始启动时间。如果未指定 *keepdari*，则忽略此参数。

记录和恢复

恢复您的环境对防止重要数据的丢失可以说是非常重要。提供了大量参数来帮助您管理您的环境，并确保您能够对您的数据或事务执行充分的恢复。这些参数分为下列几类:

- 第345页的『数据库日志文件』
- 第350页的『数据库日志活动』
- 第354页的『恢复』
- 第360页的『分布式工作单元恢复』。

数据库日志文件

下列参数提供有关用于数据库记录的文件的数量、大小和状态的信息:

- 『日志文件的大小 (logfilsiz)』
- 第346页的『主日志文件数 (logprimary)』
- 第348页的『辅助日志文件数 (logsecond)』
- 第348页的『更改数据库日志路径 (newlogpath)』
- 第350页的『日志文件的位置 (logpath)』
- 第350页的『第一个活动的日志文件 (loghead)』

日志文件的大小 (logfilsiz)

配置类型 数据库

参数类型 可配置的

缺省值（范围）

UNIX	1000 [4 — 65 535]
Windows NT	250 [4 — 65 535]
OS/2	250 [4 — 65 535]

计量单位 页 (4 KB)

相关参数

- 第346页的『主日志文件数 (logprimary)』
- 第348页的『辅助日志文件数 (logsecond)』
- 第351页的『恢复范围和软检查点间隔 (softmax)』

此参数定义每个主日志文件和辅助日志文件的大小。在这些日志文件已满且需要新日志文件之前，这些日志文件的大小限制可写入这些日志文件的日志记录数。

主日志文件和辅助日志文件的使用以及在日志文件已满时进行的操作取决于正在执行的记录的类型志:

- 循环记录

  在已落实记录在主日志文件中的更改时可重新使用该主日志文件。如果日志文件大小较小，并且应用程序已处理大量对数据库的更改而没有落实这些更改，则主日志文件可能很快变满。如果所有主日志文件变满，则数据库管理程序将分配辅助日志文件来保存新的日志记录。
- 日志保留记录

  当主日志文件已满时，将该日志归档并分配新的主日志文件。

建议: 必须使日志文件的大小与主日志文件数平衡:

- 如果数据库要运行大量更新、删除和 / 或插入事务，而这将导致日志文件很快变满，则应增大 *logfilsiz* 的值。

  注: 整个日志文件大小限制为 32 GB。即日志文件数 (*logprimary* + *logsecond*) 乘以每个日志文件的大小（以字节计）(*logfilsiz* * 4096) 必须小于 32 GB。

  日志文件太小则会因归档旧日志文件、分配新日志文件以及等待可用的日志文件的额外开销而影响系统性能。
- 如果磁盘空间不足，则应减小 *logfilsiz* 的值，因为主日志是按此大小预分配的。

  太大的日志文件会减小管理归档日志文件和日志文件副本时的灵活性，因为某些媒体可能无法保存整个日志文件。

如果正在使用日志保留，则当最后一个应用程序与数据库断开时，关闭并截断当前的活动日志文件。下次与数据库连接时使用下一个日志文件。所以，如果了解您的并行应用程序的记录需求，也许能确定一个将不会分配过量浪费空间的日志文件大小。

有关此参数的详情，*参考管理指南: 实现*中的 “配置数据库记录的参数”。

主日志文件数 (logprimary)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	3 [2 – 128]
计量单位	计数器
分配时	• 创建数据库 • 将日志移至一个不同位置（当更新 *logpath* 参数时会发生这种情况） • 随后，在所有用户已断开之后的下一个数据库连接期间，此参数 (*logprimary*) 的值增加 • 已将一个日志文件归档，并分配了一个新日志文件（必须启用 *logretain* 或 *userexit* 参数） • 如果已更改 *logfilsiz* 参数，则在所有用户已断开之后的下一个连接数据库期间，重新确定活动日志文件的大小。

释放时

除非此参数减小，否则不释放。如果此参数减小，则在下一个数据库连接期间删除不需要的日志文件。

相关参数

- 第345页的『日志文件的大小 (logfilsiz)』
- 第348页的『辅助日志文件数 (logsecond)』
- 第353页的『启用日志保留 (logretain)』
- 第354页的『启用用户出口 (userexit)』

主日志文件建立分配给恢复日志文件的固定内存量。此参数允许您指定要预分配的主日志文件数。

在循环记录类型下，按顺序重复使用主日志。即，当日志已满时，使用序列中的下一个主日志（如果该主日志可用）。如果一个日志中所有具有日志记录的工作单元都已落实或回滚，则认为该日志可用。如果序列中下一个主日志不可用，则分配并使用一个辅助日志。分配并使用附加的辅助日志，直到序列中下一个主日志变为可用或达到 *logsecond* 参数所设置的限制为止。当数据库管理程序不再需要这些辅助日志文件时，动态释放这些辅助日志文件。

主日志文件和辅助日志文件的数目必须符合下列方程式:

- (*logprimary* + *logsecond*) <= 128

建议: 此参数值的选择取决于许多因素，包括正使用的记录类型、日志文件的大小以及处理环境的类型（例如，事务的长度和落实的频率）。

增加此值将增加日志的磁盘需求，因为主日志文件就是在第一个与数据库的连接期间预分配的。

如果发现正在频繁分配辅助日志文件，也许能够通过增大日志文件大小 (*logfilsiz*) 或增大主日志文件数来改进系统性能。

对于那些不频繁存取的数据库，为了节约磁盘存储量，将该参数设置为 2。对于为前滚恢复而启用的数据库，将此参数设置得大一些，以避免几乎立即就分配新日志的额外开销。

可以使用数据库系统监控程序来帮助您确定主日志文件的大小。

有关详情，参见 *System Monitor Guide and Reference* 中下列监控程序元素的说明:

- *sec_log_used_top（使用的最大辅助日志空间）*
- *tot_log_used_top（使用的总日志空间）*
- *sec_logs_allocated（当前分配的辅助日志）*

在一个时间周期内对这些监控程序值进行观察将帮助您作出更好的调整决定，因为平均值也许更能代表您现在的需求。

辅助日志文件数 (logsecond)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 2 [0 – 126]

计量单位 计数器

分配时 当 *logprimary* 不够时需要（查看以下详情）

释放时 数据库管理程序确定将不再需要它们的结束时间。

相关参数

- 第345页的『日志文件的大小 (logfilsiz)』
- 第346页的『主日志文件数 (logprimary)』
- 第353页的『启用日志保留 (logretain)』
- 第354页的『启用用户出口 (userexit)』

此参数指定（仅需要时）创建并用于恢复日志文件的辅助日志文件的数目。当主日志文件已满时，可按需要一次分配一个辅助日志文件（大小为 *logfilsiz*），最多可分配此参数控制的最大数目。如果需要的辅助日志文件数超过此参数所允许的数，则将返回一个错误码至该应用程序，并将关闭该数据库。

有关如何使用辅助日志的详情，参阅第346页的『主日志文件数 (logprimary)』。

建议：对于定期需要大量日志空间的数据库，使用辅助日志文件。例如，每月运行一次的应用程序需要的日志空间可能超过由主日志文件提供的日志空间。由于辅助日志文件不需要永久的文件空间，所以辅助日志文件在这种情况下有优势。

更改数据库日志路径 (newlogpath)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 空值（Null（任何有效的路径或设备）

相关参数

- 第350页的『日志文件的位置 (logpath)』
- 第370页的『数据库一致 (database_consistent)』

此参数允许您指定最多 242 个字节的字符串，以更改存储日志文件的位置。该字符串可指向一个路径名或一个原始设备。如果字符串指向路径名，则它必须是全限定路径名，而不能是相对路径名。

注: 在分区数据库环境中，节点号自动附加至路径。这样做是为了保持多逻辑节点配置中路径的唯一性。

要指定设备，指定操作系统标识成设备的字符串。例如:

- 在 Windows NT 上，为 `\\.\d:` 或 `\\.\PhysicalDisk5`

  注: 您必须拥有安装了服务包 3 的 Windows NT 版本 4.0，才能够将日志写入设备。
- 在基于 UNIX 的平台上，为 `/dev/rdblog8`

注: 您只能在 AIX、Windows NT 和 Solaris 平台上指定设备。

在下列两种情况发生之前，新设置值不会成为 *logpath* 的值:

- 该数据库处于一致的状态，如 *database_consistent* 参数所指。
- 所有用户与数据库断开

当建立与该数据库的第一个新连接时，数据库管理程序将这些日志移至由 *logpath* 指定的新位置。

旧日志路径中可能会有日志文件。这些日志文件可能尚未归档。您可能需要人工归档这些日志文件。并且，如果正对此数据库运行复制，则复制可能仍需要日志路径更改之前的日志文件。如果配置数据库时将“用户出口启用”(*userexit*) 数据库配置参数设置为 『Yes』，且 DB2 已自动地或您已人工地归档了所有日志文件，则 DB2 将能够检索日志文件来完成复制过程。否则，可以将这些文件从旧日志路径复制至新日志路径。

建议: 理想情况是，这些日志文件将位于**没有**大量 I/O 的物理磁盘上。例如，避免将日志与操作系统或大容量数据库放在同一磁盘上。这将提高记录活动的效率并使其额外开销（如等待 I/O）最小。

可使用数据库系统监控程序跟踪与数据库记录相关的 I/O 数。

有关详情，参考 *System Monitor Guide and Reference* 中下列监控程序元素的说明:

- *log_reads*（读取的日志页数）
- *log_writes*（写入的日志页数）。

前导数据元素返回与数据库记录相关的 I/O 活动量。可使用操作系统监控工具收集关于其他磁盘 I/O 活动的信息，然后比较这两种类型的 I/O 活动。

日志文件的位置 (logpath)

配置类型 数据库

参数类型 资料性

相关参数 第348页的『更改数据库日志路径 (newlogpath)』

此参数包含用于进行记录的当前路径。当对 *newlogpath* 参数的更改生效后，您不能直接更改此参数，因为它是由数据库管理程序设置的。

当创建一个数据库时，在包含该数据库的目录的一个子目录中创建该数据库的恢复日志文件。缺省值是为该数据库创建的目录之下的子目录 SQLOGDIR。

第一个活动的日志文件 (loghead)

配置类型 数据库

参数类型 资料性

此参数包含当前活动的日志文件的名称。

数据库日志活动

下列参数会影响数据库记录的类型和性能:

- 『对组的落实次数 (mincommit)』
- 第351页的『恢复范围和软检查点间隔 (softmax)』
- 第353页的『启用日志保留 (logretain)』
- 第354页的『启用用户出口 (userexit)』

对组的落实次数 (mincommit)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 1 [1 – 25]

计量单位 计数器

此参数允许您延迟将日志记录写入磁盘，直到执行了最小数目的落实为止。此延迟有助于减少与写入日志记录相关的数据库管理程序额外开销，因而在对一个数据库运行多个应用程序以及应用程序在一个很短的时间帧内请求多个落实时能够改进性能。

仅当此参数的值大于 1 且连接至该数据库的应用程序的数目大于或等于此参数的值时，才会发生这种落实组合。执行落实组合时，保持应用程序落实请求，直到经过 1秒钟或落实请求数等于此参数的值为止。

对为此参数指定的值的更改立即生效；您不必等到所有应用程序与该数据库断开。

建议: 如果多个读／写应用程序在一般情况下请求并行数据库落实，增加此参数的缺省值。这将导致更有效的日志文件 I/O，因为日志文件 I/O 进行的频率减小，每次进行日志文件 I/O 时能够写入更多的日志记录。

也可以对每秒事务数采样并调整此参数以调整每秒事务数的峰值（或该峰值的较大百分率）。调整峰值活动会使高负荷期写入日志记录的额外开销减至最小。

如果增加 *mincommit*，也可能需要增加 *logbufsz* 参数，以避免在这些重负荷期间让已满的日志缓冲区强制写入。在此情况下， *logbufsz* 应等于:

```
mincommit * (事务使用的日志空间的平均值)
```

可使用数据库系统监控程序来帮助您以下列方式调整此参数:

- 计算每秒的事务峰值数:

  通过抽取典型的一天的监控程序样本，您可确定您的重负荷周期。您可通过添加下列监控元素来计算总事务数:
  - *commit_sql_stmts （尝试的落实语句数）*
  - *rollback_sql_stmts （尝试的回滚语句数）*

  使用此信息和可用的时间戳记，就可以计算出每秒事务数。
- 计算每个事务使用的日志空间:

  通过对一段时间和大量事件使用样本技术，您可计算与下列监控元素一起使用的平均日志空间:
  - *log_space_used（使用的工作单元日志空间）*

  有关数据库系统监控程序的详情，参见 *System Monitor Guide and Reference*。

恢复范围和软检查点间隔 (softmax)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	100 [1 – 100 * *logprimary*]
计量单位	主日志文件总数的百分率
相关参数	• 第345页的『日志文件的大小 (logfilsiz)』

• 第346页的『主日志文件数 (logprimary)』

此参数用于:

- 影响在一次崩溃（如电源故障）之后需要恢复的日志数。例如，如果使用缺省值，数据库管理程序将尝试把需要恢复的日志数保持为 1。如果您指定 300 作为此参数的值，则数据库管理程序将尝试把需要恢复的日志数保持为 3。

  要影响进行应急恢复所需要的日志数，数据库管理程序使用此参数来触发页清除器，以确保比指定的恢复窗口旧的页都已写入磁盘。
- 确定软检查点的频率。

当由于事件（如电源故障）造成数据库失效时，可能会对该数据库作下列更改:

- 尚未落实，但更新了缓冲池中的数据
- 已落实，但尚未从缓冲池写入磁盘
- 已落实并从缓冲池写入磁盘。

当重新启动某个数据库时，将使用日志文件来执行该数据库的应急恢复，这确保该数据库处于一致状态（即，所有已落实的事务都应用于该数据库而所有未落实的事务都不应用于该数据库）。

要确定需要将日志文件中的哪些记录应用于该数据库，数据库管理程序使用日志控制文件。定期将此日志控制文件写入磁盘，并且根据此事件的频率，数据库管理程序可能正在应用已落实事务的日志记录，或正在应用那些描述已从缓冲池写入磁盘的更改的日志记录。这些日志记录对数据库没有影响，但应用这些日志记录会将一些额外开销引入到数据库重新启动进程中。

当日志文件已满时以及在软检查点期间，总是将日志控制文件写入磁盘。可使用此配置参数来触发附加软检查点。

软检查点的定时基于“当前状态”和“记录的状态”之间的差别，该差别以 *logfilsiz* 的百分比给出。“记录的状态”由磁盘上日志控制文件中指示的最旧的有效日志记录确定，而“当前状态”由内存中的日志控制信息确定。（最旧的有效日志记录是恢复进程将读取的第一个日志记录。）如果下列公式计算的值大于或等于此参数的值，将设置该软检查点:

```
( (记录的状态和当前状态之间的空间) / logfilsiz ) *  100 * logprimary
```

建议: 您可能想增加或减少此参数的值，这取决于您的可接受的恢复窗口是大于还是小于一个日志文件。降低此参数的值将导致数据库管理程序不但更经常地触发页清除器而且还更频繁地采用软检查点。这些操作可减少需要处理的日志记录数和在应急恢复期间处理的冗余日志记录数。

但是注意: 更多的页清除器触发器和更频繁的软检查点增加了与数据库记录关联的额外开销，这可能会影响数据库管理程序的性能。而且，如果您遇到下列情况，更频繁的软检查点可能不会缩短重新启动一个数据库所需的时间:

- 落实点很少的很长的事务。
- 很大的缓冲池并且未将包含落实事务的页很频繁地写回至磁盘。(注意: 使用异步页清除器可避免这种情况。查看第325页的『异步页清除器数 (num_iocleaners)』。)

在这两种情况中，保持在内存中的日志控制信息更改不频繁，因而在将日志控制信息写入磁盘时没有意义，除非日志控制信息已更改。

启用日志保留 (logretain)

配置类型 数据库

参数类型 可配置的

缺省值 (范围) No [Recovery; Capture; No]

相关参数

- 第354页的『启用用户出口 (userexit)』
- 第371页的『日志保留状态指示符 (log_retain_status)』
- 第370页的『备份暂挂指示符 (backup_pending)』

这些值如下所示:

- No，表示不保留这些日志。
- Recovery，表示要保留这些日志，并可用于正向恢复。而且，如果使用数据复本，Capture 程序将记录在这些日志中的更新写到更改表中。
- Capture，表示仅保留这些日志，以便 Capture 程序可将更新写入更改表中。这些日志可用于正向恢复 (如果它们未被修剪的话)。

如果 *logretain* 设置为 『Recovery』 或 *userexit* 设置为 『Yes』，则将保留活动日志文件，并且这些文件将成为联机归档日志文件以用于前滚恢复。这称为日志保留记录。

在 *logretain* 设置为 『Recovery』 或 *userexit* 设置为 『Yes』 (或两者都这样设置) 之后，则必须为该数据库建立一个完整的备份。此状态由 *backup_pending* 标志参数指示。

如果 *logretain* 设置为 『No』 且 *userexit* 设置为 『No』，则不能对数据库进行前滚恢复。

将 *logretain* 设置为 『Capture』 时，则在 Capture 程序完成时它将调用 PRUNE LOGFILE 命令来删除日志文件。如果希望对数据库执行前滚恢复，则不应将 *logretain* 设置为 『Capture』。

如果 *logretain* 设置为 『No』 且 *userexit* 设置为 『No』，则不保留这些日志。在这种情况下，数据库管理程序将删除 *logpath* 目录中的所有日志文件（包括联机归档日志文件），分配新的活动日志文件，并还原至循环记录。

启用用户出口 (userexit)

配置类型 数据库

参数类型 可配置的

缺省值（范围） No [Yes; No]

相关参数

- 第353页的『启用日志保留 (logretain)』
- 第371页的『用户出口状态指示符 (user_exit_status)』
- 第370页的『备份暂挂指示符 (backup_pending)』

如果启用此参数，则不管 *logretain* 参数是如何设置的都执行日志保留记录。此参数也指示应使用用户出口程序以归档和检索日志文件。当数据库管理程序关闭日志文件时，将日志文件归档。当 ROLLFORWARD 实用程序需要使用这些日志文件复原数据库时，将检索这些日志文件。

在启用 *logretain* 和／或 *userexit* 参数后，您必须为该数据库建立一个完整的备份。此状态由 *backup_pending* 标志参数指示。

如果取消对这两个参数的选择，则前滚恢复对该数据库不可用，因为将不再保留日志。在此情况下，数据库管理程序删除 *logpath* 目录中的所有日志文件（包括联机归档日志文件），分配新的活动日志文件，并还原至循环记录。

有关用户出口程序的详情，*参考管理指南: 实现*中的 “数据库恢复的用户出口”。

恢复

下列参数影响数据库恢复的各个方面:

- 第355页的『启用自动重新启动 (autorestart)』
- 第355页的『索引重建时间 (indexrec)』
- 第357页的『缺省的装入恢复对话数 (dft_loadrec_ses)』
- 第357页的『恢复历史保留期 (rec_his_retentn)』

- 第357页的『数据库备份数 (num_db_backups)』

另见第360页的『分布式工作单元恢复』。

当使用 Tivoli 存储管理器 (TSM) 时，使用下列参数:

- 第358页的『Tivoli 存储管理器管理类 (tsm_mgmtclass)』
- 第358页的『Tivoli 存储管理器 口令 (tsm_password)』
- 第359页的『Tivoli 存储管理器节点名 (tsm_nodename)』
- 第359页的『Tivoli 存储管理器拥有者名 (tsm_owner)』

启用自动重新启动 (autorestart)

配置类型 数据库

参数类型 可配置的

缺省值（范围） On [On; Off]

当此参数设置为 on 时，如果需要，数据库管理程序在应用程序连接至数据库时自动调用重新启动数据库实用程序。应急恢复是由重新启动数据库实用程序所执行的操作。如果数据库在与应用程序连接时异常终止，则执行应急恢复。电源故障或系统软件故障可能会导致数据库异常终止。应急恢复应用发生故障时在数据库缓冲池中但未写入磁盘的任何已落实的事务。应急恢复也取消已写入磁盘的任何未落实的事务。

如果未启用 *autorestart*，则试图与需要执行应急恢复（需要重新启动）的数据库连接的应用程序将接收到 SQL1015N 错误。在这种情况下，该应用程序可调用重新启动数据库实用程序，或者您可以通过选择该恢复工具的重新启动操作来重新启动数据库。

索引重建时间 (indexrec)

配置类型 数据库和数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

UNIX 数据库管理程序
restart [restart; access]

OS/2 和 Windows NT 数据库管理程序
access [restart; access]

数据库 使用系统设置 [system; restart; access]

相关参数 第355页的『启用自动重新启动 (autorestart)』

此参数指示数据库管理程序将何时尝试重新构建无效索引。此参数有三个可能的设置:

SYSTEM *使用系统设置*，这将导致在数据库管理程序配置文件中指定的时间重建无效的索引。(注意: 此设置仅对数据库配置有效。)

ACCESS *在索引存取期间*，这将导致第一次存取该索引时重建无效的索引。

RESTART *在数据库重新启动期间*，这将导致在显式或隐式发出 RESTART DATABASE 命令时重建无效的索引。注意，如果启用 *autorestart* 参数，则隐式发出 RESTART DATABASE 命令。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

当出现严重的磁盘问题时，索引可能会变为无效。如果数据本身出现这个问题，则数据可能丢失。然而，如果索引出现这个问题，则可通过重新构建该索引来恢复索引。如果在用户连接至数据库时重新构建索引，则可能出现两个问题:

- 重建索引文件时可能出现响应时间意外降低现象。存取表和使用此特定索引的用户将在重新构建索引时等待。
- 在重建索引之后可能持有不期望的锁定，尤其是导致索引重建的用户事务从未执行过 COMMIT 或 ROLLBACK 的话，更是如此。

建议: 在高端用户服务器上且如果重新启动时间不重要，则此选项的最佳选择将是在 DATABASE RESTART 时重建该索引，以作为在崩溃后重新将该数据库联机的过程的一部分。

将此参数设置为 『ACCESS』 将导致重建索引时数据库管理程序性能的降低。任何存取该特定索引或表的用户将必须等待，直到重建过程完成为止。

如果将此参数设置为 『RESTART』，则重新启动数据库所花的时间将因索引重建而较长，但是一旦数据库恢复联机，正常处理将不受影响。

缺省的装入恢复对话数 (dft_loadrec_ses)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	1 [1 – 30 000]
计量单位	计数器

此参数指定将在表装入的恢复期间使用的缺省对话数。应将此值设置为要用来检索装入副本的最佳 I/O 对话数。对装入副本进行检索是与复原相似的操作。也可以通过在环境变量 DB2LOADREC 指定的副本位置文件中的项来覆盖此参数。

用于装入检索的缺省缓冲区数比此参数的值多两个。也可以覆盖副本位置文件中的缓冲区数。

此参数仅当启用了前滚恢复时才适用。

有关装入恢复的详情，参考 *Data Movement Utilities Guide and Reference*。

数据库备份数 (num_db_backups)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	12 [1 — 32 768]
相关参数	『恢复历史保留期 (rec_his_retentn)』

此参数指定为一个数据库保存的数据库备份的数目。当达到指定的备份数时，会在恢复历史文件中将旧备份标记为到期。与到期的数据库备份相关的表空间备份和装入副本备份的恢复历史文件项也标记为到期。当将一个备份标记为到期时，可从存储的地方（例如，磁盘、磁带、ADSM）除去物理备份。下一个数据库备份将从恢复历史文件中删除到期的项。

当在历史文件中将一个数据库备份标记为到期时，将从其归档服务器中除去通过 DB2 DataLinks Manager链接的任何对应的文件备份。

应将 *rec_his_retentn* 配置参数设置为与 *num_db_backups* 的值兼容的值。例如，如果将 *num_db_backup* 设置为一个大的值，则 *rec_his_retentn* 的值应足够大以支持该备份数。

恢复历史保留期 (rec_his_retentn)

配置类型	数据库

参数类型	可配置的
缺省值（范围）	366 [-1; 0 — 30 000]
计量单位	天数
相关参数	第357页的『数据库备份数 (num_db_backups)』

此参数用以指定应保留关于备份的历史信息的天数。如果不需要恢复历史文件来跟踪备份、复原以及装入，则可将此参数值设置为一个较小的数。

如果此参数值是 -1，则只能使用可用的命令或 API 显式修剪恢复历史文件。如果此参数值不是 -1，则在每一个全数据库备份后都修剪恢复历史文件。

此参数的值将取代 *num_db_backups* 参数的值，但是 *rec_his_retentn* 和 *num_db_backups* 必须一起工作。如果 *num_db_backups* 的值很大，则 *rec_his_retentn* 的值应足够大以便支持那个备份数目。

除非使用带 FORCE 选项的 PRUNE 工具，否则不管保留期多短，最新的整个数据库备份加上它的复原集都将始终保留。有关此实用程序的详情，参见 *Command Reference*。

Tivoli 存储管理器管理类 (tsm_mgmtclass)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	空值（任何字符串）

Tivoli 存储管理器管理类决定 TSM 服务器应如何管理正在备份的对象的备份版本。

缺省情况是无 TSM 管理类。

管理类是由 Tivoli 存储管理器管理员指定的。一旦指定了管理类，就应该将此参数设置为管理类名。当执行任何 TSM 备份时，数据库管理程序使用此参数来将管理类传送至 TSM。

有关 Tivoli 存储管理器的详情，*参考管理指南: 实现*中的 “Tivoli 存储管理程序”。

Tivoli 存储管理器 口令 (tsm_password)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	空值（任何字符串）

此参数用于替换与 Tivoli 存储管理器 (TSM) 产品相关的口令的缺省设置。需要口令以允许您复原从另一节点备份至 TSM 的数据库。

注: 如果在用 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间替换了 *tsm_nodename*，则可能还必须设置 *tsm_password*。

缺省情况为只能从执行了备份的同一节点上的 TSM 中复原一个数据库。有可能在用 DB2 执行备份期间替换 *tsm_nodename*。

有关 Tivoli 存储管理器的详情，参考*管理指南: 实现*中的 “Tivoli 存储管理程序”。

Tivoli 存储管理器节点名 (tsm_nodename)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 空值（任何字符串）

此参数用于覆盖与 Tivoli 存储管理器 (TSM) 产品相关的节点名的缺省设置。需要节点名以允许您复原从另一节点备份至 TSM 的数据库。

缺省情况为只能从执行了备份的同一节点上的 TSM 中复原一个数据库。有可能在通过 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间替换 *tsm_nodename*。

有关 Tivoli 存储管理器的详情，参考*管理指南: 实现*中的 “Tivoli 存储管理程序”。

Tivoli 存储管理器拥有者名 (tsm_owner)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 空值（任何字符串）

此参数用于覆盖与 Tivoli 存储管理器(TSM) 产品相关的拥有者的缺省设置。需要拥有者名称以允许您复原从另一节点备份至 ADSM 的数据库。有可能在通过 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间替换 *tsm_owner*。

注: 拥有者名称是区分大小写的。

缺省情况为只能从执行了备份的同一节点上的 TSM 中复原一个数据库。

有关 Tivoli 存储管理器的详情，参考*管理指南: 实现*中的 “Tivoli 存储管理程序”。

分布式工作单元恢复

下列参数影响分布式工作单元 (DUOW) 事务的恢复:

- 『事务管理程序数据库名 (tm_database)』
- 第361页的『事务再同步间隔 (resync_interval)』
- 第361页的『同步点管理程序日志文件路径(spm_log_path)』
- 第362页的『同步点管理程序名 (spm_name)』
- 第362页的『同步点管理程序日志文件大小 (spm_log_file_sz)』
- 第363页的『同步点管理程序再同步代理程序限制 (spm_max_resync)』

事务管理程序数据库名 (tm_database)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 1ST_CONN（任何有效的数据库名）

此参数标识每个 DB2 实例的“事务管理程序”(TM) 数据库的名称。TM 数据库可以是:

- 本地“DB2 通用数据库”数据库
- 不驻留在主机或 AS/400 系统上的远程“DB2 通用数据库”数据库
- DB2 OS/390 版的版本 5 数据库（如果通过 TCP/IP 存取且未使用“同步点管理程序”的话）。

TM 数据库是用作记录器和协调程序的数据库，并用来对不确定事务执行恢复。

可以将此参数设置为 **1ST_CONN**，这样将把 TM 数据库设置为用户所连接的第一个数据库。

有关分布式工作单元的详情，*参考管理指南: 计划*中的“分布式数据库”。

建议: 为了简化管理和操作，您可能希望对许多实例创建几个数据库，并将这些数据库专门用作 TM 数据库。

事务再同步间隔 (resync_interval)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 180 [1 – 60 000]

计量单位 秒

此参数指定“事务管理程序”(TM)、“资源管理器”(RM) 或“同步点管理程序”(SPM) 再试恢复 TM、RM 或 SPM 中任何未完成的不确定事务的时间间隔（以秒计）。当有事务运行于分布式工作单元 (DUOW) 环境中时此参数适用。

有关分布式工作单元的详情，*参考管理指南: 计划*中的“分布式数据库”。

建议: 如果在您的环境中不确定事务将不干扰应用于您的数据库的其他事务，则您可能希望增加此参数的值。如果正使用 DB2 Connect 网关存取“DRDA2 应用服务器”，则即使将不干扰本地数据存取，也应考虑不确定事务对“应用服务器”产生的可能影响。如果没有不确定事务，则性能影响将最小。

同步点管理程序日志文件路径(spm_log_path)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 sqllib/spmlog [任何有效的路径或设备]

此参数指定将“同步点管理程序”(SPM) 日志写入的目录。缺省情况下，将这些日志写入 `sqllib/spmlog` 目录，这在有大量事务的环境中可导致 I/O 瓶颈。使用此参数来将 SPM 日志文件放在比当前 `sqllib/spmlog` 目录更快的磁盘上。它允许在 SPM 代理程序中有更好的并行性。

有关“同步点管理程序”的详情，*参考安装和配置补遗*。

有关恢复未确定的 DRDA 事务的详情，*参考管理指南: 计划*中的“恢复主机上的不确定事务”。

同步点管理程序名 (spm_name)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 从 TCP/IP 主机名派生

此参数向数据库管理程序标识“同步点管理程序”(SPM) 实例的名称。

有关“同步点管理程序”的详情，*参考安装和配置补遗*。

有关恢复未确定的 DRDA 事务的详情，*参考管理指南: 计划*中的“恢复主机上的不确定事务”。

同步点管理程序日志文件大小 (spm_log_file_sz)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 256 [4 — 1 000]

计量单位 页

此参数以 4 KB 页为单位标识“同步点管理程序”(SPM) 日志文件的大小。日志文件包含在 sqllib 下的 spmlog 子目录中，并且在首次启动 SPM 时创建。

有关“同步点管理程序”的详情，*参考安装和配置补遗*。

有关恢复未确定的 DRDA 事务的详情，*参考管理指南: 计划*中的“恢复主机上的不确定事务”。

建议: “同步点管理程序”日志文件大小应适中，以便既可维护性能又防止浪费空间。所需要的大小取决于使用保护会话的事务数目和发出 COMMIT 或 ROLLBACK 的频率。

要更改 SPM 日志文件的大小:

1. 通过使用 LIST DRDA INDOUBT TRANSACTIONS 命令确定没有任何不确定事务。
2. 如果没有不确定事务，则停止数据库管理程序。
3. 用新的 SPM 日志文件大小来更新“数据库管理程序配置”。
4. 转至 `$HOME/sqllib` 目录，并发出 `rm -fr spmlog` 命令以删除当前 SPM 日志。（注意: 显示的是 AIX 命令。其他系统可能会需要不同的除去或删除命令。）
5. 启动数据库管理程序。（在启动数据库管理程序时创建一个指定大小的新的 SPM 日志。）

同步点管理程序再同步代理程序限制 (spm_max_resync)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 20 [10 — 256]

此参数标识可同时执行再同步操作的代理程序数。

有关恢复未确定的 DRDA 事务的详情，*参考管理指南: 计划*中的“恢复主机上的不确定事务”。

有关“同步点管理程序”的详情，*参考安装和配置补遗*。

提供了大量参数，它们提供有关您数据库的信息或影响数据库的管理。这些参数分为以下几组:

- 『Query Enabler』
- 『属性』
- 第367页的『DB2 DataLinks Manager』
- 第369页的『状态』
- 第371页的『编译程序设置』。

Query Enabler

下列参数提供用于控制 Query Enabler 的信息:

- 『动态 SQL 查询管理 (dyn_query_mgmt)』

动态 SQL 查询管理 (dyn_query_mgmt)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	0 (DISABLE) [1(ENABLE), 0 (DISABLE)]

此参数与 DB2 Query Patroller 的安装位置相关。如果数据库配置参数 *dyn_query_mgmt* 设置为 『ENABLE』，且动态查询的成本超过用户或组的 trap_threshold（在 DB2 Query Patroller 用户简要表中指定），则 DB2 Query Patroller 将捕捉此查询。trap_threshold 是一个基于成本的触发器，用于用户在 DB2 Query Patroller 中建立的查询捕捉。当捕捉到动态查询时，将显示一个对话，供用户指定运行期参数。

如果 *dyn_query_mgmt* 设置为 『DISABLE』，则不捕捉查询。

属性

下列参数提供有关数据库的一般信息:

- 第365页的『配置文件发行版级别 (release)』
- 第365页的『数据库发行版级别 (database_level)』
- 第365页的『数据库的地区 (territory)』
- 第365页的『数据库的国家代码 (country)』
- 第366页的『数据库的代码集 (codeset)』
- 第366页的『数据库的代码页 (codepage)』
- 第366页的『整理信息 (collate_info)』

- 第367页的『启用副本保护 (copyprotect)』

除 *copyprotect* 外，提供的其他参数仅供参考。

配置文件发行版级别 (release)

配置类型 数据库管理程序，数据库

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 资料性

相关参数 『数据库发行版级别 (database_level)』

此参数指定配置文件的发行版级别。

数据库发行版级别 (database_level)

配置类型 数据库

参数类型 资料性

相关参数 『配置文件发行版级别 (release)』

此参数指示可使用数据库的数据库管理程序的发行版级别。在未完成或失败的迁移情况中，此参数将反映未迁移数据库的发行版级别，而且它可能与 *release* 参数（数据库配置文件的发行版级别）不同。否则，*database_level* 的值将等于 *release* 参数的值。

数据库的地区 (territory)

配置类型 数据库

参数类型 资料性

相关参数 『数据库的国家代码 (country)』

此参数显示用于创建数据库的地区。数据库管理程序使用地区来确定 *country* 参数值。有关数据库管理程序如何使用地区的详情，参见*快速入门*。

数据库的国家代码 (country)

配置类型 数据库

参数类型 资料性

相关参数 第365页的『数据库的地区 (territory)』

此参数显示用于创建数据库的国家代码。*country* 参数是根据 *territory* 参数派生的。有关详情，参见*快速入门*。

数据库的代码集 (codeset)

配置类型 数据库

参数类型 资料性

相关参数 『数据库的代码页 (codepage)』

此参数显示创建数据库所使用的代码集。数据库管理程序使用代码集来确定 *codepage* 参数值。有关数据库管理程序如何使用代码集的详情，参见*快速入门*。

数据库的代码页 (codepage)

配置类型 数据库

参数类型 资料性

相关参数 『数据库的代码集 (codeset)』

此参数显示创建数据库所使用的代码页。*codepage* 参数是根据 *codeset* 参数派生的。有关详情，参见*快速入门*。

整理信息 (collate_info)

只能使用 GET DATABASE CONFIGURATION API 来显示此参数。它**不能**通过命令行处理器或控制中心来显示。

配置类型 数据库

参数类型 资料性

此参数提供 260 字节的数据库整理信息。前面 256 个字节指定数据库整理顺序，其中字节 ″n″ 包含代码点的排序加权，该代码点在数据库代码页中的基本十进制表示为 ″n″。

最后 4 个字节包含关于整理顺序类型的内部信息。可将它看作一个可应用于数据库平台的整数。有三个值:

- **0** – 该顺序包含非唯一的加权
- **1** – 该顺序包含所有唯一的加权
- **2** – 该顺序是等同顺序，对于这种顺序将逐个字节地比较字符串。

如果使用此内部类型信息，在不同平台上检索数据库的信息时需要考虑字节转换。

可在创建数据库时指定整理顺序。

启用副本保护 (copyprotect)

配置类型 数据库

参数类型 可配置的

缺省值（范围） No [Yes; No]

此参数启用副本保护属性，缺省情况下禁用此参数。在数据库管理程序的版本 2 以前的版本中，缺省值为启用副本保护属性。

此参数不适用于基于 UNIX 的环境。

备份数据库和复原数据库实用程序不受 *copyprotect* 参数的影响。可以备份一个副本保护数据库，将它复原到另一个工作站，然后编目并存取该数据库。

注意: 在重新安装数据库管理程序或操作系统之前，从所有数据库中除去副本保护。如果不除去副本保护，将在尝试存取数据库时收到一个错误信息。重新安装之后，可以启用副本保护。

DB2 DataLinks Manager

下列参数和 DB2 DataLinks Manager 有关:

- 『数据链路存取令牌到期时间间隔 (dl_expint)』
- 第368页的『数据链路副本数 (dl_num_copies)』
- 第368页的『卸下后的数据链路时间 (dl_time_drop)』
- 第368页的『数据链路令牌算法 (dl_token)』
- 第369页的『大写的数据链路令牌 (dl_upper)』
- 第369页的『启用数据链路支持 (datalinks)』

数据链路存取令牌到期时间间隔 (dl_expint)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 60 [-1, 1 — 31 536 000]

计量单位 秒

此参数指定生成的文件存取控制令牌持续有效的时间间隔（以秒计）。该令牌的有效秒数从生成它时开始计算。该 DataLinks Filesystem Filter 依据此到期时间检查令牌的有效性。

有关文件存取控制令牌的信息，参考 *DB2 Data Links Manager Quick Beginnings* 一书。

此参数的缺省值是六十 (60) 秒。-1 表示该令牌实际将不会到期。

此参数适用于指定 『READ PERMISSION DB』 的 DATALINK 列。

数据链路副本数 (dl_num_copies)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 0 [0 – 15]

此参数指定当一个文件与该数据库链接时，在归档服务器（如 ADSM 服务器）中要为该文件生成的附加副本数。

此参数的缺省值是零 (0)。

此参数适用于指定 『Recovery=Yes』 的 DATALINK 列。

卸下后的数据链路时间 (dl_time_drop)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 1 [0 — 365]

计量单位 天数

此参数指定在发出 DROP DATABASE 之后在归档服务器（如 TSM 服务器）上保留文件的时间间隔（以天计）。

此参数的缺省值是 1 天。零值 (0) 表示当发出 DROP 命令或语句时立即从归档服务器中删除文件。（除非为 DATALINK 列指定了 ON UNLINK DELETE 参数，否则不删除实际文件。）

此参数适用于指定 『Recovery=Yes』 的 DATALINK 列。

数据链路令牌算法 (dl_token)

配置类型 数据库

参数类型 可配置的

缺省值（范围） MAC0 [MAC0; MAC1]

此参数指定在生成 DATALINK 文件存取控制令牌时使用的算法。值 MAC1（信息认证代码）生成比 MAC0 更安全的信息认证代码，但也增加了性能的额外开销。

有关文件存取控制令牌的信息，参考 *DB2 Data Links Manager Quick Beginnings* 一书。

此参数适用于指定 『READ PERMISSION DB』 的 DATALINK 列。

大写的数据链路令牌 (dl_upper)

配置类型 数据库

参数类型 可配置的

缺省值（范围） NO [YES; NO]

此参数指示文件存取控制令牌是否使用大写字母。值 『YES』 指定存取控制令牌中的所有字母都是大写。值 『NO』 指定该令牌可包含大写和小写字母。

有关文件存取控制令牌的信息，参考 *DB2 Data Links Manager Quick Beginnings* 一书。

此参数适用于指定 『READ PERMISSION DB』 的 DATALINK 列。

启用数据链路支持 (datalinks)

配置类型 数据库管理程序

参数类型 可配置的

缺省值（范围） NO [YES; NO]

此参数指定是否启用数据链路支持。值 『YES』 指定是否对本机文件系统（例如，AIX 上的 JFS）中存储的 Data Links Manger 链接文件启用数据链路支持。值 『NO』 指定不启用数据链路支持。

状态

下列参数提供有关数据库状态的信息:

- 第370页的『备份暂挂指示符 (backup_pending)』
- 第370页的『数据库一致 (database_consistent)』
- 第370页的『前滚暂挂指示符 (rollfwd_pending)』
- 第371页的『日志保留状态指示符 (log_retain_status)』

- 第371页的『用户出口状态指示符 (user_exit_status)』
- 第371页的『复原暂挂 (restore_pending)』
- 第371页的『启用多页文件分配 (multipage_alloc)』

备份暂挂指示符 (backup_pending)

配置类型 数据库

参数类型 资料性

如果设置为 on，此参数指示您必须在存取该数据库之前，为其建立一个完整的备份。此参数仅在下列情况下才为 on: 该数据库配置更改，以便该数据从不可恢复转变为可恢复（即，最初将 *logretain* 和 *userexit* 参数设置为 NO，然后将这两个参数中的一个或两个设置为 YES，并接受对该数据库配置的更新。）

数据库一致 (database_consistent)

配置类型 数据库

参数类型 资料性

此参数指示数据库是否处于一致状态。

是指示所有事务都已落实或回滚，以保证数据是一致的。如果系统在数据库一致时“崩溃”，则不必进行任何特别操作以使该数据库可用。

否指示一个事务暂挂，或对该数据库执行的某个其他任务暂挂，且此时数据不一致。如果系统在数据库不一致时“崩溃”，则需要使用 RESTART DATABASE 命令重新启动数据库以使该数据库可用。有关 RESTART DATABASE 命令的详情，参见 *Command Reference*。

前滚暂挂指示符 (rollfwd_pending)

配置类型 数据库

参数类型 资料性

此参数可指示下列其中一种状态:
- **DATABASE**，表示此数据库必需前滚恢复过程。
- **TABLESPACE**，表示一个或多个表空间需要前滚
- **NO**，表示该数据库是可用的，且不需要前滚恢复。

在可以存取数据库或表空间之前，必须完成恢复（使用 ROLLFORWARD DATABASE）。有关 ROLLFORWARD DATABASE 的详情，参阅 *Command Reference*。

日志保留状态指示符 (log_retain_status)

配置类型 数据库

参数类型 资料性

相关参数 第353页的『启用日志保留 (logretain)』

如果设置了此参数，则此参数指示日志文件被保留以用于前滚恢复。

当 *logretain* 参数设置成为活动时，设置此参数。

用户出口状态指示符 (user_exit_status)

配置类型 数据库

参数类型 资料性

相关参数 第354页的『启用用户出口 (userexit)』

如果设置为 ON，则指示启用数据库管理程序用于前滚恢复，并且将使用用户出口程序在数据库管理程序调用日志文件时归档和检索日志文件。

复原暂挂 (restore_pending)

配置类型 数据库

参数类型 资料性

此参数说明数据库中是否有 RESTORE PENDING 状态。

启用多页文件分配 (multipage_alloc)

配置类型 数据库

参数类型 资料性

多页文件分配用来提高插入性能。它只适用于 SMS 表空间。如果启用多页文件分配，则影响所有 SMS 表空间：不可能对各个 SMS 表空间进行选择。

该参数的缺省值为 NO：未启用多页文件分配。

创建数据库之后，该参数可能设置为 YES，指示启用了多页文件分配。这使用 *db2empfa* 工具来完成。一旦设置为 YES，该参数不能再更改为 NO。

编译程序设置

下列参数提供影响编译程序的信息：

- 第372页的『遇到算术异常继续 (dft_sqlmathwarn)』

- 第373页的『缺省度 (dft_degree)』
- 第374页的『缺省查询优化级别 (dft_queryopt)』
- 第374页的『缺省刷新时限 (dft_refresh_age)』
- 第375页的『保存的高频值数目 (num_freqvalues)』
- 第376页的『列的分位数数目 (num_quantiles)』

遇到算术异常继续 (dft_sqlmathwarn)

配置类型 数据库

参数类型 可配置的

缺省值（范围） No [No, Yes]

此参数设置缺省值，在 SQL 语句编译期间出现错误或警告时，该缺省值确定处理算术错误和检索转换错误的方法。对于静态 SQL 语句，此参数值与联编时的程序包相关。对于动态 SQL DML 语句，在准备语句时使用此参数的值。

注意: 如果您更改一个数据库的 *dft_sqlmathwarn* 值，则包括算术表达式的检查约束、触发器和视图的行为可能改变。反过来这也可能影响数据库的数据完整性。您只应在仔细评估新的算术异常处理行为可以如何影响检查约束、触发器和视图之后，才能为一个数据库更改 *dft_sqlmathwarn* 的设置。一旦更改，接下来的更改同样也需要仔细评估。

例如，考虑下列检查约束，它包括一个除法算术运算:

```
A/B > 0
```

当 *dft_sqlmathwarn* 为 『No』 且试图执行 B=0 的 INSERT 时，将被零除作为算术错误来处理。由于 DB2 不能检查约束，该插入操作失败。如果将 *dft_sqlmathwarn* 更改为 『Yes』，则将被零除作为算术警告来处理，结果为 NULL。结果为 NULL 会导致 『>』 谓词结果为 UNKNOWN，从而插入操作能够成功运行。如果将 *dft_sqlmathwarn* 更改回 『No』，则插入同一行的尝试将失败，因为被零除的错误将阻止 DB2 评估该约束。当 *dft_sqlmathwarn* 为 『Yes』 时用 B=0 插入的行将保留在该表中，且可选择。对于导致对约束进行估算的那一行的更新将失败，而对于不需要约束重算的那一行的更新则将成功。

在把 *dft_sqlmathwarn* 从 『No』 更改为 『Yes』 之前，您应考虑重新编写该约束，以显式处理算术表达式产生的空值 例如:

```
( A/B > 0 ) AND ( CASE
                  WHEN A IS NULL THEN 1
                  WHEN B IS NULL THEN 1
```

```
    WHEN A/B IS NULL THEN 0
    ELSE 1
    END
  = 1 )
```

能在 A 和 B 为可空时使用。如果 A 或 B 为不可空，则可除去相应的 IS NULL WHEN 子句。

在把 *dft_sqlmathwarn* 从 『Yes』 更改为 『No』 之前，您应首先检查是否有可能变得不一致的数据，例如，通过使用如下所示的谓词:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

当隔离不一致的行时，在更改 *dft_sqlmathwarn* 之前，您应执行适当的操作以校正不一致性 也可以在更改之后用算术表达式人工复查约束。为此，首先使受到影响的表处于检查暂挂状态（通过 SET CONSTRAINTS 语句的 OFF 子句），然后请求检查这些表（通过 SET CONSTRAINTS 语句的 IMMEDIATE CHECKED 子句）。算术错误将指示不一致的数据，这可避免对约束进行估算。

建议: 除非您明确地需要处理包括算术异常的查询，否则，使用缺省设置 no。然后将该值指定为 yes。如果在其他数据库管理程序中处理提供结果的 SQL 语句，而不考虑发生的算术异常，则可能发生此情况。

缺省度 (dft_degree)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	1 [-1, 1 – 32 767]
相关参数	第394页的『最大查询并行度 (max_querydegree)』

此参数指定 CURRENT DEGREE 专用寄存器和 DEGREE 联编选项的缺省值。

缺省值为 1。

1 意味着无分区内并行性。-1 意味着优化器根据处理器数目和查询类型确定分区内并行度。

编译语句时使用 CURRENT DEGREE 专用寄存器或 DEGREE 联编选项指定 SQL 语句的分区内并行度。使用 SET RUNTIME DEGREE 命令指定活动应用程序的分区内的最大运行期并行度。 最大查询并行度 (max_querydegree) 配置参数指定对于所有 SQL 查询的最大查询分区内并行度。

所用的实际运行时间度是下列参数中最低的:

- *max_querydegree* 配置参数

- 应用程序运行期并行度
- SQL 语句编译并行度

缺省查询优化级别 (dft_queryopt)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	5 [0 — 9]
计量单位	查询优化级别（参见以下内容）

在编译 SQL 查询时，查询优化级别用于指导优化器使用不同程度的优化。此参数通过设置所使用的缺省查询优化级别（在既未使用 SET CURRENT QUERY OPTIMIZATION 语句又未使用 QUERYOPT 联编命令时采用的），从而提供了更大的灵活性。

当前定义的查询优化级别是:
- 0 - 最小查询优化。
- 1 - 大致与 DB2 版本 1 相当。
- 2 - 轻微优化。
- 3 - 适度查询优化。
- 5 - 有效的试探查询优化，可限制选择存取方案所消耗的成本。这是缺省情况。
- 7 - 有效的查询优化。
- 9 - 最大查询优化

建议: 有关选择合适的查询优化级别的详情和准则，参见第57页的『调整优化级别』。

有关程序如何检索和修改数据库配置参数的详情，参见 *Administrative API Reference*。

缺省刷新时限 (dft_refresh_age)

配置类型	数据库
参数类型	可配置的
缺省值（范围）	0 [0, 99999999999999 (ANY)]

此参数带有未指定 CURRENT REFRESH AGE 专用寄存器时用于 REFRESH AGE 的缺省值。此参数指定一个时间戳记宽度值，其数据类型为 DECIMAL(20,6)。此时间宽度表示自从在特定 REFRESH DEFERRED 摘要表上处理 REFRESH TABLE 语句之后的最大持续时间，在此期间，可以使用该摘要表来优化查询处理。如果

CURRENT REFRESH AGE 的值为 99999999999999 (ANY)，且 QUERY OPTIMIZATION 类大于或等于 5，则考虑使用 REFRESH DEFERRED 摘要表来优化动态 SQL 查询的处理。

保存的高频值数目 (num_freqvalues)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 10 [0 — 32 767]

计量单位 计数器

相关参数

- 第376页的『列的分位数数目 (num_quantiles)』
- 第306页的『统计堆大小 (stat_heap_sz)』

此参数允许您指定在 RUNSTATS 命令中指定 WITH DISTRIBUTION 选项时收集的“最高频值”的数目。增大此参数的值将会增加在收集统计信息时所使用的统计信息堆 (*stat_heap_sz*) 的数量

“最高频值”统计信息帮助优化器了解列中数据值的分布。较高的值可使 SQL 优化器得到更多信息，但是需要额外的目录空间。当指定为 0 时，即使您请求收集分布统计信息，也不保留任何高频值统计信息。

对于非均匀分布数据上的一些谓词 (=、<、>、IS NULL、IS NOT NULL)，更新此参数能帮助优化器获得更好的选择性估计。选择性计算越精确，选择的存取方案就越有效。

在更改此参数的值之后，您需要:
- 在所有用户已与数据库断开并且您已重新连接至数据库之后，运行 RUNSTATS 命令。
- 重新联编任何包含静态 SQL 的程序包。

有关详情，参见第100页的『收集和使用分布统计信息』。

建议: 为了更新此参数，您应确定通常具有选择谓词的最重要列（在最重要的表中）中的非均匀度。使用提供每个值在一列中出现次数的有序顺序的 SQL SELECT 语句就可实现这一点。不应考虑均匀分布的、唯一的、长的或 LOB 列。此参数合理的实际值范围是从 10 至 100。

注意: 收集高频值统计信息的过程需要大量的 CPU 和内存 (*stat_heap_sz*) 资源

列的分位数数目 (num_quantiles)

配置类型 数据库

参数类型 可配置的

缺省值（范围） 20 [0 – 32 767]

计量单位 计数器

相关参数

- 第375页的『保存的高频值数目 (num_freqvalues)』
- 第306页的『统计堆大小 (stat_heap_sz)』

此参数控制在 RUNSTATS 命令中指定 WITH DISTRIBUTION 选项时将收集的分位数的数目。增大此参数的值将会增加在收集统计信息时所使用的统计信息堆 (*stat_heap_sz*) 的数量。

“分位数”统计信息帮助优化器了解列中数据值的分布。较高的值可使 SQL 优化器得到更多信息，但是需要额外的目录空间。指定 0 或 1 时，不保留分位数统计信息，即使您请求收集分布统计信息。

对于非均匀分布数据上的范围谓词，更新此参数能帮助获得更好的选择性估计。在其他优化器决策中，此信息对于选择索引扫描还是选择表扫描具有很大的影响。（存取频繁出现的值的范围使用表扫描是更为有效的，而对不频繁出现的值的范围则使用索引扫描更为有效。）

在更改此参数的值之后，您需要:

- 在所有用户已与数据库断开并且您已重新连接至数据库之后，运行 RUNSTATS 命令。
- 重新联编任何包含静态 SQL 的程序包。

有关详情，参见第100页的『收集和使用分布统计信息』。

建议: 此参数的缺省值保证，对于任何单边范围谓词 （>、>=、< 或 <=）的最大估计错误大约为 2.5%，而对于任何 BETWEEN 谓词的最大错误则为 5%。确定分位数数目的经验方法为:

- 确定在估计任何范围查询的行数时可允许的最大错误百分比，即 P
- 如果大多数谓词为 BETWEEN 谓词，则分位数的数目应大约为 100/P，如果大多数的谓词是其他类型的范围谓词（<、<=、> 或 >=），则分位数的数目大约为 50/P。

例如，25 个分位数导致的最大估计错误对于 BETWEEN 谓词应为 4%，而对于 ″>″ 谓词则应为 2%。此参数合理的实际值范围为 10 至 50。

通信

下列几组参数提供有关在客户机／服务器环境中使用 DB2 的信息:

- 『通信协议设置』
- 第381页的『分布式服务』
- 第385页的『DB2 Discovery』

通信协议设置

可使用下列参数来配置数据库客户机和数据库服务器:

- 『NetBIOS 工作站名 (nname)』
- 第378页的『TCP/IP 服务名 (svcename)』
- 第379页的『APPC 事务程序名 (tpname)』
- 第379页的『IPX/SPX 文件服务器名 (fileserver)』
- 第380页的『IPX/SPX DB2 服务器对象名 (objectname)』
- 第380页的『IPX/SPX 套接字号 (ipx_socket)』

NetBIOS 工作站名 (nname)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

此参数允许为 NetBIOS LAN 环境中工作站上的数据库实例指定唯一的名称。此 *nname* 是将对工作站的 NetBIOS 注册的实际 NetBIOS 名称的基础。

因为 NetBIOS 协议使用这些 NetBIOS 名建立连接，所以必须为客户机和服务器设置 *nname* 参数。

客户机应用程序必须获知包含要存取的数据库的服务器的 *nname*。必须使用 CATALOG NETBIOS NODE 命令将服务器的 *nname* 作为 『server-nname』 参数编目在客户机的节点目录中。

如果服务器节点上的 *nname* 更改为新名称，则所有存取该服务器上的数据库的客户机必须为该服务器编目此新名称。

TCP/IP 服务名 (svcename)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 空

此参数包含数据库服务器将用于等待来自远程客户机节点的通信的 TCP/IP 端口的名称。此名称必须是保留给数据库管理程序使用的两个连续端口的第一个；第二个端口用于处理先前版本的客户机的中断请求。

为了使用 TCP/IP 接受来自数据库客户机的连接请求， 数据库服务器必须在指定给该服务器的端口上进行监听。数据库服务器的系统管理员必须保留一个端口（编号为 *n*）并在服务器的 services 文件中定义其相关 TCP/IP 服务名。如果数据库服务器需要支持来自先前版本的客户机的请求，则需要在该服务器的 services 文件中定义第二个端口（编号为 *n+1*，用于中断请求）。

需要在数据库客户机上的 services 文件中定义数据库服务器端口（编号为 *n*）和它的 TCP/IP 服务名。先前版本的客户机还需要在客户机的 services 文件中定义中断端口（编号为 *n+1*）。

services 文件的位置取决于操作环境。例如:
- 在 UNIX 中 — /etc/services
- 在 OS/2 中 — \tcpip\etc\services
- 在 OS/2 Warp 中 — \mptn\etc\services。

应将 *svcename* 参数设置为与主连接端口相关的服务名，以便当启动数据库服务器时，它可以确定在哪个端口上监听入局连接请求。如果支持或使用先前版本的客户机，则配置文件中不保存中断端口的服务名。根据主连接端口号可得到中断端口号 （*中断端口号 = 主连接端口 + 1*）。

有关为数据库服务器设置 TCP/IP 的详情，*参考安装和配置补遗*。

APPC 事务程序名 (tpname)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

此参数定义数据库客户机在使用 APPC 通信协议向数据库服务器发出分配请求时必须使用的远程事务程序的名称。必须在数据库服务器上的配置文件中设置此参数。

此参数必须与 SNA 事务程序定义中配置的事务程序名相同。有关为 DB2 产品设置 APPC 的详情，*参考安装和配置补遗*。

建议: 只能接受用于此名称中的字符为:
- 字母（A 至 Z; 或 a 至 z）
- 数字（0 至 9）
- 美元符号 ($)、数值符号 (#)、at 符号 (@) 和句点 (.)

IPX/SPX 文件服务器名 (fileserver)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第380页的『IPX/SPX DB2 服务器对象名 (objectname)』
- 第380页的『IPX/SPX 套接字号 (ipx_socket)』

此参数指定数据库管理程序的网际地址注册处的 NetWare** 文件服务器的名称。数据库管理程序的网际地址存储在 NetWare 文件服务器上的装订库中。如果注册的文件服务器名改变，则所有存取该服务器实例的客户机必须:

- 取消编目服务器节点
- 编目服务器节点，指定新文件服务器名。

有关详情，*参考安装和配置补遗*。

IPX/SPX DB2 服务器对象名 (objectname)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第379页的『IPX/SPX 文件服务器名 (fileserver)』
- 『IPX/SPX 套接字号 (ipx_socket)』

此参数提供 IPX/SPX 网络中数据库管理程序实例的名称。每个注册至 NetWare 文件服务器的服务器实例必须有唯一名称。如果此名称在数据库服务器上更改，则所有存取该服务器的客户机必须取消对该服务器节点的编目，并再重新编目该节点，指定新对象名。

IPX/SPX 套接字号 (ipx_socket)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） 879E [879E – 87A2] 为了确保不存在冲突，向

Novell 注册了五个唯一的套接字号 （879E 至 87A2），以供 DB2 使用。

相关参数

- 第379页的『IPX/SPX 文件服务器名 (fileserver)』
- 第380页的『IPX/SPX DB2 服务器对象名 (objectname)』

此参数指定“公认”套接字号并表示 DB2 服务器网际地址中的连接端点。对于给定机器上每个 DB2 服务器实例，套接字号必须是唯一的，并在所有在此机器上运行的 Novell** IPX/SPX 应用程序中是唯一的。这将保证 DB2 服务器能够使用此套接字号监听入局的 IPX/SPX 连接请求。

分布式服务

可使用下列参数配置数据库客户机和数据库服务器使用 DCE 目录服务:

- 『目录服务类型 (dir_type)』
- 第382页的『DCE 名称空间中的目录路径名 (dir_path_name)』
- 第383页的『DCE 名称空间中的对象名 (dir_obj_name)』
- 第383页的『路径选择信息对象名 (route_obj_name)』
- 第384页的『缺省客户机通信协议 (dft_client_comm)』
- 第385页的『缺省客户机适配器号 (dft_client_adpt)』

有关 DB2 如何使用 DCE 目录的信息，*参考管理指南: 计划*中的“使用分布式计算环境 (DCE) 目录服务”。

目录服务类型 (dir_type)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- UNIX 和 OS/2 客户机
- UNIX 和 OS/2 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） NONE [NONE; DCE]

相关参数

- 第383页的『DCE 名称空间中的对象名 (dir_obj_name)』
- 『DCE 名称空间中的目录路径名 (dir_path_name)』
- 第383页的『路径选择信息对象名 (route_obj_name)』
- 第384页的『缺省客户机通信协议 (dft_client_comm)』
- 第385页的『缺省客户机适配器号 (dft_client_adpt)』

此参数指示是否使用 DCE 目录服务。

如果将此参数设置为 **NONE**，则只搜索本地目录文件，以查找 CONNECT 或 ATTACH 请求的目标。但是，您仍可以使用 *dir_path_name* 和 *dir_obj_name* 参数来记录 DCE 名称空间的数据库实例和数据库的名称。

如果将此参数设置为 **DCE**，则当在此数据库管理程序实例内运行的应用程序找不到它的 CONNECT 或 ATTACH 请求的目标时，就会搜索 DCE 目录。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

DCE 名称空间中的目录路径名 (dir_path_name)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- UNIX 和 OS/2 客户机
- UNIX 和 OS/2 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 /.:/subsys/database/

相关参数

- 第383页的『DCE 名称空间中的对象名 (dir_obj_name)』
- 第381页的『目录服务类型 (dir_type)』
- 第383页的『路径选择信息对象名 (route_obj_name)』

全局名称空间中的数据库管理程序实例的唯一名称是由此值和 *dir_obj_name* 参数中的值组成。

所有在此实例内运行的客户机应用程序还使用它作为其 CONNECT 或 ATTACH 请求的缺省路径名，除非它被 DB2DIRPATHNAME 环境变量的值替换。

建议： 使用您的 DCE 管理员提供的名称。

DCE 名称空间中的对象名 (dir_obj_name)

配置类型 数据库管理程序，数据库

适用于

- 带本地客户机和远程客户机的数据库服务器
- UNIX 和 OS/2 客户机
- UNIX 和 OS/2 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第381页的『目录服务类型 (dir_type)』
- 第382页的『DCE 名称空间中的目录路径名 (dir_path_name)』

目录中表示数据库管理程序实例（或数据库）的对象名。此值与 *dir_path_name* 值合并构成一个全局名称，它唯一地标识由 *dir_type* 参数指定的目录服务所管理的名称空间中的数据库管理程序实例或数据库。

仅当指定了 *dir_path_name* 参数时，此参数才有意义。

配置参数 *dir_path_name* 和 *dir_obj_name* 的总长度必须小于 255 个字符。

建议：有关详情，参考*管理指南: 实现*中的“使用分布式计算环境 (DCE) 目录服务”。

路径选择信息对象名 (route_obj_name)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第382页的『DCE 名称空间中的目录路径名 (dir_path_name)』
- 第381页的『目录服务类型 (dir_type)』

此参数指定缺省路径选择信息对象项的名称，所有尝试存取 DRDA 服务器的客户机应用程序将使用该路径选择信息对象项。此参数仅适用于 OS/2 和基于 UNIX 的环境。

如果此参数的值以 */.:/* 或 */.../* 开头，则按原样使用该值。否则，将它追加至 *dir_path_name* 参数（或 DB2DIRPATHNAME 变量）值之后，以构成路径选择信息对象的全名。

可使用环境变量 DB2ROUTE 覆盖此缺省值。

仅当将 *dir_type* 参数设置为 DCE 时，此参数才有意义。

建议：有关详情，*参考管理指南: 实现*中的“使用分布式计算环境 (DCE) 目录服务”。

缺省客户机通信协议 (dft_client_comm)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） NULL[NULL; TCPIP; APPC; IPXSPX（仅用于 OS/2）; NETBIOS（仅用于 OS/2）]

相关参数 第381页的『目录服务类型 (dir_type)』

此参数指示此实例上的客户机应用程序可用于远程连接的通信协议。该参数的内容是一个字符串，由一个或多个令牌组成。如果要指定多个令牌，则用逗号将它们分开。就优先权来说令牌的次序是有意义的。

此参数只能和 DCE 一起使用，并只适用于 OS/2 和基于 UNIX 的环境。

可通过设置 DB2CLIENTCOMM 环境变量来临时覆盖此参数的值。

如果此参数的值为 NULL，并且尚未设置环境变量，则使用服务器的全局目录对象中指定的第一个协议。

如果将 *dir_type* 设置为 NONE，则忽略此参数。

建议: 应该首先指定最常使用的协议。

缺省客户机适配器号 (dft_client_adpt)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器

参数类型 可配置的

缺省值（范围） 0 [0–15]

相关参数

- 第384页的『缺省客户机通信协议 (dft_client_comm)』.
- 第381页的『目录服务类型 (dir_type)』.（将 dir_type 设置为 DCE 时。）

此参数定义 NETBIOS 协议的缺省客户机适配器号，该客户机的服务器 nname 是从 “DCE 单元目录服务” (CDS) 中抽取的。此参数只适用于 OS/2 环境。

此参数只能在 DCE 中使用。

您可以通过设置 DB2CLIENTADPT 环境变量来临时替换此参数的值。如果此环境变量包含非数值的编号或超出范围的编号，则使用 0（零）作为适配器号。

DB2 Discovery

可使用下列参数来建立 DB2 Discovery:

- 第386页的『发现数据库 (discover_db)』
- 第386页的『发现方式 (discover)』
- 第387页的『搜索发现通信协议 (discover_comm)』
- 第388页的『Discover 服务器实例 (discover_inst)』

发现数据库 (discover_db)

配置类型 数据库

参数类型 可配置的

缺省值（范围） Enable [Disable, Enable]

在服务器上接收到发现请求时，此参数用来防止将关于数据库的信息返回至客户机。

此参数的缺省值为对此数据库启用发现。

通过将此参数值更改为 『Disable』，可以隐藏含有敏感数据的数据库，以防止被发现过程发现。此操作可与数据库上的其他数据库安全性控制一起完成。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

发现方式 (discover)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） search [disable, known, search]

相关参数 第387页的『搜索发现通信协议 (discover_comm)』

从管理服务器角度来看，此配置参数确定当 DB2ADMIN 启动时启动的发现方式的类型。

- 当 DB2ADMIN 启动时，如果 *discover* = SEARCH，则启动 *discover_comm* 中指定的每种协议的搜索发现连接管理程序。另外，还启动 DB2COMM 注册表变量中指定的每种协议的连接管理程序。这使管理服务器能够处理来自客户机的搜索发现请求。SEARCH 提供了 KNOWN 发现所提供的功能的超集。当 *discover* = SEARCH 时，管理服务器将同时处理来自客户机的搜索和已知发现请求。

- 当 DB2ADMIN 启动时，如果 *discover* = KNOWN，则只启动 DB2COMM 注册表变量中指定的连接管理程序。这些连接管理程序处理 KNOWN 发现请求。
- 当 DB2ADMIN 启动时，如果 *discover* = DISABLE，则管理服务器将不会处理任何类型的发现请求。

从服务器实例角度来看，如果 *discover* = DISABLE，则此服务器实例的信息基本上对客户机隐藏了起来。当任何客户机对此系统发出已知发现请求时，管理服务器不封装关于此实例的信息。

从客户机角度来看，将发生下列其中一项:

- 如果 *discover* = SEARCH，则客户机可以发出搜索发现请求来查找网络上的 DB2 服务器系统。搜索发现提供了 KNOWN 发现所提供的功能的超集。如果 *discover* = SEARCH，则客户机既可发出搜索发现请求也可发出已知发现请求。
- 如果 *discover* = KNOWN，则客户机只能发出已知发现请求。通过对特定系统上的管理服务器指定一些连接信息，可以把 DB2 系统上的所有实例和数据库信息返回给客户机。
- 如果 *discover* = DISABLE，则在客户机上禁用发现。

缺省发现方式是 SEARCH。

有关这些值的等效数字和 API 约束，参考 *Administrative API Reference*。

有关 DB2 Discovery 的详情，参考适合您平台的*快速入门*手册。

搜索发现通信协议 (discover_comm)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 无（NETBIOS 与 TCPIP 的任何组合）

相关参数 第386页的『发现方式 (discover)』

从管理服务器角度来看，这个参数定义在 DB2ADMIN 启动时启动的搜索发现管理程序。这些管理程序服务搜索来自客户机的发现请求。

注: *discover_comm* 中定义的协议还必须在 DB2COMM 注册表变量中指定。

从客户机角度来看，此参数定义客户机用来发出搜索发现请求的协议。

可指定多个协议，协议之间用逗号分隔；也可让该参数保留为空白。

此参数的缺省值为“无”，意味着没有任何搜索发现通信协议。

Discover 服务器实例 (discover_inst)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） enable [enable, disable]

此参数指定是否可用 DB2 Discovery 检测此实例。缺省值 『enable』 指定可检测该实例，而 『disable』 则防止该实例被发现。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

有关 DB2 Discovery 的详情，参考适合您平台的*快速入门*手册。

并行

下列几组参数提供有关并行操作和分区数据库环境的信息:

- 『通信』
- 第394页的『并行处理』.

通信

下列参数提供有关在分区数据库环境中通信的信息:

- 第389页的『连接经过时间 (conn_elapse)』
- 第389页的『FCM 信息锚数 (fcm_num_anchors)』

- 第390页的『FCM 缓冲区数 (fcm_num_buffers)』
- 第391页的『FCM 连接项数 (fcm_num_connect)』
- 第391页的『FCM 请求块数 (fcm_num_rqb)』
- 第392页的『节点连接重试次数 (max_connretries)』
- 第393页的『节点之间的最大时间差 (max_time_diff)』
- 第393页的『启动和停止超时 (start_stop_time)』。

连接经过时间 (conn_elapse)

配置类型	数据库管理程序
适用于	带本地客户机和远程客户机的分区数据库服务器
参数类型	可配置的
缺省值（范围）	10 [0–100]
计量单位	秒
相关参数	第392页的『节点连接重试次数 (max_connretries)』

此参数指定在两个数据库分区服务器之间建立 TCP/IP 连接所用的秒数。如果该尝试在此参数指定的时间内完成，则建立了通信。如果失败，则进行另一次尝试来建立通信。如果连接尝试的次数达到 *max_connretries* 参数指定的次数且始终超时，则发出错误。

FCM 信息锚数 (fcm_num_anchors)

配置类型	数据库管理程序
适用于	• 带本地客户机和远程客户机的数据库服务器 • 带本地客户机的数据库服务器 • 带本地客户机和远程客户机的分区数据库服务器 • 带本地客户机的卫星数据库服务器
参数类型	可配置的
缺省值（范围）	-1 [-1, 128–*fcm_num_rqb*] 在非分区数据库系统中，*intra_parallel* 参数必须是活动的才能使用此参数。
相关参数	

- 第391页的『FCM 请求块数 (fcm_num_rqb)』
- 第395页的『启用分区内并行性 (intra_parallel)』

此参数指定 FCM *信息锚*数。代理程序使用信息锚在相互之间发送信息。缺省值 (-1) 指示为 *fcm_num_rqb* 指定的值的 75%。

FCM 缓冲区数 (fcm_num_buffers)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 512、1 024 或 4 096 [128 — *fcm_num_rqb*]

- 带本地客户机和远程客户机的数据库服务器:缺省值是 1 024
- 带本地客户机的数据库服务器:缺省值是 512
- 带本地客户机和远程客户机的分区数据库服务器:缺省值是 4 096

在单分区数据库系统中，*intra_parallel* 参数必须是活动的才能使用此参数。

此参数指定分区数据库环境中数据库服务器之间及内部用于内部通信（信息）的 4 KB 缓冲区数。

有关 FCM 的详情，*参考管理指南: 实现*中的“启用 FCM 通信”。

如果您在一个处理机上有多个逻辑节点，则可能发现有必要增加此参数的值。如果由于系统上的用户数、系统上的数据库分区服务器数或这些应用程序的复杂程度，而使信息缓冲区用尽，您可能也会发现增加此参数的值是必要的。

如果您正在使用多逻辑节点，则在非 AIX 系统上，同一机器上的所有多逻辑节点共享一个缓冲区数为 *fcm_num_buffers* 的池，而在 AIX 上:

- 如果在数据库管理程序所使用的一般内存中有足够的空间，则将从一般内存中分配 FCM 缓冲堆。在这种情况下，每个数据库分区服务器将会有它自己的 *fcm_num_buffers* 缓冲区；数据库分区服务器将不会共享 FCM 缓冲池（这曾是 DB2 版本 5 的新增内容）。
- 如果在数据库管理程序所使用的一般内存中没有足够的空间，则将从一般内存（AIX 共享内存集）中分配 FCM 缓冲堆，即由相同机器上所有多个本地节点共享的。同一机器上所有多逻辑节点将共享一个缓冲区数为 *fcm_num_buffers* 的池。这与非 AIX 系统相同，也与用于 AIX 的 “DB2 并行版的版本 1.2” 相同。

给 AIX 上现有的并行版客户的建议：如果正在使用多逻辑节点，则在并行版的版本 1.2 中使用的 *fcm_num_buffers* 的值现在可能会使每台机器使用的存储器显著增加。例如，一个四个节点的多逻辑节点配置最后得出的 FCM 缓冲区数可能为以前的四倍。

复查正在使用的值；考虑在多逻辑节点所在的机器上总共将分配多少 FCM 缓冲区。您可能想更改 *fcm_num_buffers* 以说明上述行为。

FCM 连接项数 (fcm_num_connect)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 [-1, 128— *fcm_num_rqb*]

在非分区数据库系统中，*intra_parallel* 参数必须是活动的才能使用此参数。

相关参数 『FCM 请求块数 (fcm_num_rqb)』

此参数指定 FCM *连接项*数。代理程序使用连接项在相互之间传送数据。缺省值 (-1) 指示为 *fcm_num_rqb* 指定的值的 75%。

FCM 请求块数 (fcm_num_rqb)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

UNIX 32 位平台

256、512、2 048 [128 — 120 000]

UNIX 64 位平台

256、512、2 048 [128 — 524 288]

OS/2 和 Windows NT

10 000 [250 — 2 097 152]

- 带本地客户机和远程客户机的数据库服务器: 缺省值是 512
- 带本地客户机的数据库服务器:缺省值是 256
- 带本地客户机和远程客户机的分区数据库服务器: 缺省值是 2 048

在非分区数据库系统中，*intra_parallel* 参数必须是活动的才能使用此参数。

此参数指定 FCM *请求块*数。请求块是在 FCM 精灵程序和代理程序之间或代理程序和代理程序之间传送信息的媒体。

需要的请求块数将根据系统上的用户数、系统中的数据库分区服务器数以及所运行的查询的复杂性而变化。最初，以缺省数开始，并在微调此参数时使用来自数据库系统监控程序的结果。

节点连接重试次数 (max_connretries)

配置类型 数据库管理程序

适用于 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） 5 [0–100]

相关参数 第389页的『连接经过时间 (conn_elapse)』

如果试图在两个数据库分区服务器之间建立通信失败（例如，达到 *conn_elapse* 参数指定的值），则 *max_connretries* 指定可对数据库分区服务器进行连接重试的次数。如果超过为此参数指定的值，将返回一个错误。

节点之间的最大时间差 (max_time_diff)

配置类型 数据库管理程序

适用于 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） 60 [1–1 440]

计量单位 分钟

每个数据库分区服务器都有自己的系统时钟。此参数以分钟为单位指定列出在节点配置文件文件中的数据库分区服务器所允许的最大时间差。

如果两个或更多的数据库分区服务器与某一事务关联，并且它们的时钟在由此参数指定的时间内不同步，则拒绝该事务，并将一个警告或者错误信息注册在 db2diag.log 文件中。（仅当数据修改与事务相关时，才拒绝事务。）

DB2 通用数据库扩充企业版使用*标准世界时* (UTC)，因此当您设置此参数时，不用考虑不同的时区。“世界时”与“格林威治时”相同。

启动和停止超时 (start_stop_time)

配置类型 数据库管理程序

适用于 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） 10 [1 — 1 440]

计量单位 分钟

此参数只适用于分区数据库环境。此参数以分钟为单位指定时间，在该段时间内所有数据库分区服务器必须响应 DB2START 或 DB2STOP 命令。它也用作在 ADDNODE 操作期间的超时值。

在指定的时间内未响应 DB2START 命令的数据库分区服务器向此实例的 HOME 目录下 sqllib 子目录的 log 子目录中的 db2start 错误日志发送一条信息。应在重新启动这些节点之前对这些节点发出 DB2STOP 命令。

在指定的时间内未响应 DB2STOP 命令的数据库分区服务器向此实例的 HOME 目录下 sqllib 子目录的 log 子目录中的 db2stop 错误日志发送一条信息。您可以对每个未响应的数据库分区服务器或者对所有服务器发出 DB2STOP 命令。（那些已停止的服务器将返回一条信息以表明他们是停止的。）

并行处理

下列参数提供有关并行处理的信息:

- 『最大查询并行度 (max_querydegree)』
- 第395页的『启用分区内并行性 (intra_parallel)』。

最大查询并行度 (max_querydegree)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 (ANY) [ANY, 1—32 767]（ANY 意指由系统确定）

相关参数

- 第373页的『缺省度 (dft_degree)』
- 第395页的『启用分区内并行性 (intra_parallel)』

此参数指定用于在数据库管理程序的此实例上执行的任何 SQL 语句的最大分区内并行度。当执行某条 SQL 语句时，该语句在一个分区内使用的并行操作的数目将不大于此数目。必须将 *intra_parallel* 配置参数设置为 『YES』，以允许数据库分区使用分区内并行性。

此配置参数的缺省值为 -1。此值意味着系统使用由优化器确定的并行度；否则，使用用户指定的值。

注: 可使用 CURRENT DEGREE 专用寄存器或 DEGREE 联编选项在编译语句时指定 SQL 语句的并行度。

活动应用程序的最大查询并行度可以使用 SET RUNTIME DEGREE 命令来修改。实际使用的运行期并行度是下列值中较小的那一个:

- *max_querydegree* 配置参数
- 应用程序运行期并行度
- SQL 语句编译并行度

有关确定实际查询并行度的例外情况在创建索引时出现。在这种情况下，如果 *intra_parallel* 为 『YES』，且该表足够大以致于可利用使用多处理器所带来的好处，则创建索引会使用联机处理器的数量（最多为 6）加 1。不受以上提到的其他参数、联编选项或专用寄存器的影响。

启用分区内并行性 (intra_parallel)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） NO (0) [SYSTEM (-1), NO (0), YES (1)]

值 -1 导致该参数值设置为 『YES』 或 『NO』，这取决于正在运行数据库管理程序的硬件。

相关参数 第394页的『最大查询并行度 (max_querydegree)』

此参数指定数据库管理程序是否可以使用分区内并行性。

当此参数设置为 ″YES″ 时，一些参数可利用并行性能改进，这些操作包括数据库查询和索引创建。

注: 如果更改此参数值，则可能将程序包重新联编至数据库。如果发生这种情况，则重新联编期间性能可能降低。

实例管理

有大量参数可帮助您管理您的数据库管理程序实例。这些参数分为以下几类:

- 第396页的『诊断』
- 第398页的『数据库系统监控程序参数』
- 第399页的『系统管理』

- 第406页的『实例管理』

诊断

下列参数允许您控制数据库管理程序提供的诊断信息:

- 『诊断错误捕捉级别 (diaglevel)』
- 『诊断数据目录路径 (diagpath)』
- 第397页的『通知级别 (notifylevel)』.

诊断错误捕捉级别 (diaglevel)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 3 [0 — 4]

相关参数 『诊断数据目录路径 (diagpath)』

记录在 db2diag.log 文件中的诊断错误的类型由此参数确定。有效的值为:

0 – 未捕捉到诊断数据

1 – 仅严重错误

2 – 所有错误

3 – 所有错误和警告

4 – 所有错误、警告以及资料性信息

正是使用 *diagpath* 配置参数来指定一个目录，它将包含根据 *diaglevel* 参数的值可能生成的错误文件、事件日志文件（仅在 Windows NT 上）、警报日志文件以及任何转储文件。

建议: 可增大此参数的值来收集确定问题的其他数据，以帮助解决问题。

诊断数据目录路径 (diagpath)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 空值（任何有效路径名）

相关参数 第396页的『诊断错误捕捉级别 (diaglevel)』

此参数允许指定 DB2 诊断信息的全限定路径。根据您所用的平台，此目录可能包含转储文件、软中断文件、错误日志和警报日志文件。

如果此参数为空值，则诊断信息将写入下列其中一个目录或文件夹中的文件:

- 对于 OS/2 和受支持的 Windows 环境:
  - 如果**未**设置 DB2INSTPROF 环境变量或关键字，则信息将写入 x:\SQLLIB\DB2INSTANCE，其中，x:\SQLLIB 为 DB2PATH 注册表变量或环境变量中指定的驱动器引用和目录，而 DB2INSTANCE 是实例拥有者的名称。

    注: 该目录不一定要命名为 SQLLIB。
  - 如果设置了 DB2INSTPROF 环境变量或关键字，则信息写入 x:\DB2INSTPROF\DB2INSTANCE，其中 DB2INSTPROF 为实例简要表目录的名称，而 DB2INSTANCE 为实例拥有者的名称。
- 对于基于 UNIX 的环境: 信息写入 INSTHOME/sqllib/db2dump，其中 INSTHOME 为实例拥有者的主目录。
- 对于 Macintosh 环境: 信息写入 DB2 文件夹。

建议: 使用缺省值或获得多个实例的诊断路径的中央位置。

在多节点环境中，指定的路径必须驻留在共享文件系统中。

通知级别 (notifylevel)

配置类型 数据库管理程序

适用于

- Windows NT 上的带本地客户机和远程客户机的数据库服务器
- Windows NT 上的客户机

- Windows NT 上的带本地客户机的数据库服务器
- Windows NT 上的带本地客户机和远程客户机的分区数据库服务器
- Windows 95、Windows 98 和 Windows NT 上的带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 2 [0 — 4]

此参数指定写入文件的管理通知错误信息的类型。对于卫星节点类型的服务器，将错误写入通知文件 *instance*.nfy。对于所有其他节点类型，此参数仅在 Windows NT 平台上可用，并将错误写入 Windows NT 事件日志。这些错误可能是由 DB2、Capture 和 Apply 程序以及用户应用程序写入的。

此参数的有效值是:

0 — 未捕捉到诊断数据

1 — 仅严重错误

2 — 所有错误

3 — 所有错误和警告

4 — 所有错误、警告和资料性信息

用户应用程序要想写入通知文件或 Windows NT 事件日志，它必须调用 db2AdminMsgWrite API。有关此 API 的详情，参考 *Administrative API Reference*。

建议: 可增大此参数的值来收集确定问题的其他数据，以帮助解决问题。

数据库系统监控程序参数

下列参数允许您控制数据库系统监控程序的各个方面:

- 『缺省数据库系统监控程序开关 (dft_monswitches)』

缺省数据库系统监控程序开关 (dft_monswitches)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 所有开关关闭

此参数是唯一的，允许在该参数中设置许多开关，这些开关在内部分别由该参数的一位表示。根据正用来更新数据库管理程序配置的接口，也许可直接更新此参数。您也可设置下列参数来分别更新其中每个开关:

dft_mon_uow 快照监控程序的工作单元 (UOW) 开关的缺省值

dft_mon_stmt 快照监控程序的语句开关的缺省值

dft_mon_table 快照监控程序的表开关的缺省值

dft_mon_bufpool 快照监控程序的缓冲池开关的缺省值

dft_mon_lock 快照监控程序的锁定开关的缺省值

dft_mon_sort 快照监控程序的排序开关的缺省值

对这些数据库系统监控程序开关中任何一个的更改会立即生效；即，您不必停止并重新启动数据库管理程序。

注: 一个现存的监控应用程序将不会自动使用一个开关的新缺省值。要使用新的一个或多个值，该应用程序必须终止该实例，然后重新连接它。

有关快照监控程序和它如何使用监控程序开关的详情，参见 *System Monitor Guide and Reference*。

建议: 任何“打开”的开关都指导数据库管理程序收集与该开关相关的监控程序数据。收集附加监控程序数据增加数据库管理程序的额外开销，这会影响系统性能。

当应用程序发出其第一个监控请求（例如设置开关、激活事件监控程序、记录快照）时，所有监控应用程序继承这些缺省开关设置。仅当想自启动数据库管理程序起开始收集数据时，才应打开配置文件中的开关。（否则，每个监控应用程序可设置其自己的开关，并且该程序收集的数据与设置开关的时间有关。）

系统管理

下列参数与系统管理有关:

- 第400页的『通信带宽 (comm_bandwidth)』
- 第400页的『CPU 速度 (cpuspeed)』
- 第401页的『并行活动数据库的最大数目 (numdb)』
- 第402页的『事务处理器监控程序名 (tp_mon_name)』

- 第404页的『机器节点类型 (nodetype)』
- 第405页的『缺省交费帐户 (dft_account_str)』
- 第405页的『Java Development Kit 1.1 安装路径 (jdk11_path)』
- 第406页的『联合体数据库系统支持 (federated)』.

通信带宽 (comm_bandwidth)

配置类型 数据库管理程序

适用于 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） -1 [.1 – 100 000]

值 -1 导致将该参数值复位为缺省值。缺省值是根据是否在使用高速开关来计算的。

计量单位 每秒兆字节

为通信带宽计算的值以每秒兆字节计，SQL 优化器使用它来估算在一个分区数据库系统的数据库分区服务器之间执行特定操作的成本。该优化器不为客户机和服务器之间的通信成本建立模型，所以此参数应该只反映数据库分区服务器之间的标称带宽（如果有的话）。

您可以显式地设置此值，以便在您的测试系统上建立一个产品环境模型，或者评估硬件升级的效果。

建议: 如果您要建立不同的环境模型，则只应调整此参数。

优化器使用通信带宽来确定存取路径。在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

CPU 速度 (cpuspeed)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） -1 [1e-10 — 1] 根据度量程序的运行情况，值 -1 将导致该参数值被复位。

SQL 优化器使用 CPU 速度（以每条指令的毫秒数计）来估计执行某个操作的成本。此参数的值是在您安装数据库管理程序时根据一个特定程序的输出自动设置的，该程序是为测量 CPU 速度而设计的。如果由于下列任何一个原因，基准测试程序的运行结果不可用，则执行此程序:

- 平台没有 db2spec.dat 文件的支持
- db2spec.dat 文件找不到
- 该文件中找不到 IBM RISC System/6000 型号 530H 的数据
- 该文件中找不到您的机器的数据。

您可以显式地设置此值，以便在您的测试系统上建立一个产品环境模型，或者评估硬件升级的效果。通过将它设置为 -1，以重新计算 *cpuspeed*。

建议: 如果您要建立不同的环境模型，则只应调整此参数。

优化器在确定存取路径时使用 CPU 速度。在更改此参数之后应考虑重新联编应用程序（使用 REBIND PACKAGE 命令）。

并行活动数据库的最大数目 (numdb)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

UNIX 8 [1 — 256]

OS/2 和 Windows NT 带本地客户机和远程客户机的数据库服务器
8 [1 — 256]

OS/2 和 Windows NT 带本地客户机的数据库服务器和带本地客户机的卫星数据库服务器
3 [1 — 256]

计量单位 计数器

此参数指定可并行活动（即有与之连接的应用程序）的本地数据库数。在分区数据库环境中，此参数限制数据库分区服务器上的活动数据库分区数，不管该服务器是否是应用程序的协调程序节点。

由于每个数据库都占用存储器，并且活动数据库使用新的共享内存段，可以通过限制机器上的独立数据库数来减少系统资源的使用。然而，任意减少数据库数不是办法。也就是说，将所有数据（不管多么不相关）放在一个数据库中将减少磁盘空间，但可能不是好办法。通常一个好的作法是只将功能上相关的信息保存在同一数据库中。

建议：一般情况下，最好将此值设置为已经定义给数据库管理程序的数据库的实际数量，并添加一个合理的增量，以考虑短期内（例如，6 个月至 1 年）数据库数量的未来增长。实际增量不应过份大，但应允许添加新数据库而不必频繁地更新此参数。

更改 *numdb* 参数可能会影响已分配的总内存量。因此，建议不要频繁更改此参数。更新此参数时，应考虑可以为数据库或与该数据库连接的应用程序分配内存的其他配置参数，包括:

- 第290页的『缓冲池大小 (buffpage)』
- 第297页的『锁定列表的最大存储器 (locklist)』
- 第305页的『应用程序堆大小 (applheapsz)』
- 第301页的『应用程序控制堆大小 (app_ctl_heap_sz)』
- 第303页的『排序堆大小 (sortheap)』
- 第305页的『语句堆大小 (stmtheap)』
- 第313页的『应用程序支持层堆大小 (aslheapsz)』
- 第292页的『数据库堆 (dbheap)』
- 第317页的『数据库系统监控程序堆大小 (mon_heap_sz)』
- 第306页的『统计堆大小 (stat_heap_sz)』

事务处理器监控程序名 (tp_mon_name)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 无缺省值

有效值

- CICS
- MQ
- ENCINA
- CB
- SF
- TUXEDO
- TOPEND
- 空白或一些其他值（对于 UNIX、OS/2 和 Windows NT；对于 Solaris 或 SINIX，没有其他可能的值）

此参数标识正在使用的事务处理 (TP) 监控程序产品的名称。

- 如果应用程序在 “WebSphere 企业版 CICS” 环境中运行，则此参数应设置为 『CICS』
- 如果应用程序在 “WebSphere 企业版 Encina” 环境中运行，则此参数应设置为 『ENCINA』
- 如果应用程序在 “WebSphere 企业版 Component Broker” 环境中运行，则此参数应设置为 『CB』
- 如果应用程序在 IBM MQSeries 环境中运行，则此参数应设置为 『MQ』
- 如果应用程序在 BEA Tuxedo 环境中运行，则此参数应设置为 『TUXEDO』
- 如果应用程序在 IBM San Francisco 环境中运行，则此参数应设置为 『SF』。

IBM WebSphere EJB 和 **Microsoft 事务服务器** 用户无需对此参数配置任何值。

如果不是使用上述产品，则不应配置此参数，而应保留它为空白。

在 OS/2 和 Windows NT 环境中的先前版本的 “DB2 通用数据库” 中，此参数包含特定 DLL 的路径和名称，该 DLL 包含 “XA 事务管理程序” 的函数 *ax_reg* 和 *ax_unreg*。此格式仍然受支持。如果此参数的值与上述任何 “TP 监控程序” 名不匹配，则将假设它的值是包含 *ax_reg* 和 *ax_unreg* 函数的库的名称。对于 UNIX、OS/2 和 Windows NT 环境，情况均如此。

TXSeries CICS 和 Encina 用户：在 OS/2 和 Windows NT 上的此产品的先前版本中，必须将此参数配置成 『libEncServer:C』 或 『libEncServer:E』。虽然仍受支持，但已不再是一定要这样做。将此参数配置成 『CICS』 或 『ENCINA』就足够了。

MQSeries 用户: 在 OS/2 和 Windows NT 上的此产品的先前版本中，必须将此参数配置成 『mqmax』。虽然仍受支持，但已不再是一定要这样做。将此参数配置成 『MQ』 就足够了。

Component Broker 用户: 在 OS/2 和 Windows NT 上的此产品的先前版本中，必须将此参数配置成 『somtrx1i』。虽然仍受支持，但已不再是一定要这样做。将此参数配置成 『CB』 就足够了。

San Francisco 用户: 在 OS/2 和 Windows NT 上的此产品的先前版本中，必须将此参数配置成 『ibmsfDB2』。虽然仍受支持，但已不再是一定要这样做。将此参数配置成 『SF』 就足够了。

可为此参数指定的字符串的最大长度为 19 个字符。

也有可能在“DB2 通用数据库”的 XA OPEN 字符串中配置此信息。如果多个“事务处理监控程序”正在使用单一 DB2 实例，则必须使用此功能。有关使用 XA OPEN 字符串的其他信息，*参考管理指南: 计划*。

机器节点类型 (nodetype)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 资料性

此参数提供关于已安装在机器上的 DB2 产品的信息，从而提供关于数据库管理程序配置类型的信息。以下是此参数返回的可能值以及与该节点类型相关的产品:

- **带本地客户机和远程客户机的数据库服务器** – 一个 DB2 服务器产品，支持本地和远程数据库客户机，并能存取其他远程数据库服务器。
- **客户机** – 一个能存取远程数据库服务器的数据库客户机。
- **带本地客户机的数据库服务器** – 一个 DB2 关系数据库管理系统，支持本地数据库客户机并能存取其他远程数据库服务器。
- **带本地客户机和远程客户机的分区数据库服务器** – 一个 DB2 服务器产品，支持本地和远程数据库客户机，能存取其他远程数据库服务器，且具有分区并行操作的能力。

- **带本地客户机的卫星数据库服务器** DB2 关系数据库管理系统，支持本地数据库客户机并能存取其他远程数据库服务器。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

缺省交费帐户 (dft_account_str)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 空值（任何有效字符串）

对于每个应用程序连接请求，将一个由 DB2 Connect 生成的前缀和用户提供的后缀组成的记帐标识符从应用请求器发送至 DRDA 应用服务器。此记帐信息给系统管理员提供一种使资源使用与每个用户存取关联的机制。

注: 此参数仅适用于 DB2 Connect。

该后缀是通过应用程序调用 `sqlesact()` API 或用户设置环境变量 DB2ACCOUNT 来提供的。如果 API 或环境变量未提供后缀，则 DB2 Connect 将此参数的值用作缺省后缀值。对于没有能力向 DB2 Connect 转送记帐字符串的低级别数据库客户机（版本 2 以前的任何版本），此参数尤其有用。

建议: 使用下列项设置此记帐字符串:
- 字母（A 至 Z）
- 数字（0 至 9）
- 下划线 (_)。

Java Development Kit 1.1 安装路径 (jdk11_path)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机

- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） 空值（有效路径）

相关参数

- 第320页的『最大 Java 解释程序堆大小 (java_heap_sz)』

此参数指定在哪个目录安装 Java Development Kit 1.1。Java 解释程序使用的 CLASSPATH 和其他环境变量是通过此参数的值计算的。

因为此参数没有缺省值，所以安装 Java Development Kit 时应指定此参数的值。

联合体数据库系统支持 (federated)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） NO [YES; NO]

此参数启用或禁用对提交分布式请求的应用程序的支持，那些请求是针对数据源（如 DB2 系列和 Oracle）管理的数据。

实例管理

下列参数与数据库管理程序实例的安全性和管理有关:

- 第407页的『系统管理权限组名 (sysadm_group)』
- 第408页的『系统控制权限组名 (sysctrl_group)』
- 第409页的『系统维护权限组名 (sysmaint_group)』
- 第409页的『认证类型 (authentication)』
- 第411页的『不需权限就允许编目 (catalog_noauth)』

- 第411页的『缺省数据库路径 (dftdbpath)』
- 第412页的『DB2START/DB2STOP 必需的注册 (ss_logon)』
- 第413页的『信任所有客户机 (trust_allclnts)』
- 第414页的『可信赖客户机认证 (trust_clntauth)』

系统管理权限组名 (sysadm_group)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第408页的『系统控制权限组名 (sysctrl_group)』
- 第409页的『系统维护权限组名 (sysmaint_group)』

系统管理 (SYSADM) 权限是数据库管理程序中最高级别的权限，它控制所有数据库对象。此参数定义具有数据库管理程序实例的 SYSADM 权限的组名。

SYSADM 权限是由用于特定的操作环境中的安全性设施确定的。

- 在 Windows 95 或 Windows 98 操作系统中，SYSADM 组必须为 NULL。

  当使用系统安全性时，对于 Windows 95 或 Windows 98 客户机，此参数必须是『NULL』，因为 Windows 95 或 Windows 98 操作系统不存储组信息，因此不提供任何方法来确定一个用户是否是指定的 SYSADM 组的成员。指定组名时，所有用户都不能是它的成员。
- 对于 Windows NT 和 Windows 2000 操作系统，可将此参数设置为任何一个本地组，该组的名称不超过 8 个字符且是在 Windows NT 和 Windows 2000 安全性数据库中定义的。如果为此参数指定了『NULL』，则“管理员”组的所有成员都有 SYSADM 权限。
- 对于基于 UNIX 的系统，如果指定『NULL』作为此参数的值，则 SYSADM 组缺省为实例拥有者的主组。

  如果该值不为『NULL』，则 SYSADM 组可为任何有效的 UNIX 组名。

- 在 OS/2 中，如果为此参数指定的值是 『NULL』，则在用户简要表管理中定义为管理员的用户就具有 SYSADM 权限。

  如果为此参数指定了组名，则只有属于该组的用户具有 SYSADM 权限。指定的组可以是任何一个“用户简要表管理”(UPM) 组。有关“用户简要表管理”(UPM) 组的详情，参阅 *DB2（OS/2 版）Quick Beginnings* 一书。

如果使用 DCE 安全性且 *sysadm_group* 为 『NULL』，则使用缺省的 DCE 组名 DB2ADMIN。权限 ID 映射为 DB2ADMIN 的有效 DCE 负责人必须已存在。可指定一个不同的组名。

要将该参数复原为其缺省值 (NULL)，使用 UPDATE DBM CFG USING SYSADM_GROUP NULL。必须用大写字母指定关键字 『NULL』。也可以使用 DB2 控制中心中的“配置实例”笔记本。

系统控制权限组名 (sysctrl_group)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第407页的『系统管理权限组名 (sysadm_group)』
- 第409页的『系统维护权限组名 (sysmaint_group)』

此参数定义具有系统控制 (SYSCTRL) 权限的组名。SYSCTRL 具有一些特权，这些特权允许执行影响系统资源的操作但不允许直接存取数据。

注意: 对于 Windows 95 和 Windows 98 客户机，在使用系统安全性时（即认证是 CLIENT、SERVER、DCS 或任何其他有效认证），此参数必须为 NULL。这是因为 Windows 95 和 Windows 98 操作系统不存储组信息，因此不提供任何方法来确定一个用户是否是指定的 SYSCTRL 组的成员。指定组名时，所有用户都不能是它的成员。使用 DCE 认证时，情况并非如此。在此情况下，可指定组名。

要将该参数复原至它的缺省值 (NULL)，使用 UPDATE DBM CFG USING SYSCTRL_GROUP NULL。必须用大写字母指定关键字 『NULL』。也可以使用 DB2 控制中心中的“配置实例”笔记本。

系统维护权限组名 (sysmaint_group)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值 空

相关参数

- 第407页的『系统管理权限组名 (sysadm_group)』
- 第408页的『系统控制权限组名 (sysctrl_group)』

此参数定义具有系统维护 (SYSMAINT) 权限的组名。SYSMAINT 具有对所有与实例相关的数据库执行维护操作的特权，但没有对数据的直接存取权。

注意: 对于 Windows 95 和 Windows 98 客户机，在使用系统安全性时（即认证是 CLIENT、SERVER、DCS 或任何其他有效认证），此参数必须为 NULL。这是因为 Windows 95 和 Windows 98 操作系统不存储组信息，因此不提供任何方法来确定一个用户是否是指定的 SYSMAINT 组的成员。指定组名时，所有用户都不能是它的成员。使用 DCE 认证时，情况并非如此。在此情况下，可指定组名。

要将该参数复原为其缺省值 (NULL)，使用 UPDATE DBM CFG USING SYSMAINT_GROUP NULL。必须用大写字母指定关键字 『NULL』。也可以使用 DB2 控制中心中的“配置实例”笔记本。

认证类型 (authentication)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机

- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围） SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DCS; DCS_ENCRYPT; DCE; DCE_SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT]

此参数确定用户的认证如何进行以及在何处进行。

如果认证是 SERVER，则将用户 ID 和口令从客户机发送至服务器，以便可在服务器上发生认证。值 SERVER_ENCRYPT 提供与 SERVER 相同的行为，只是要加密通过网络发送的任何口令。

值 CLIENT 指示所有的认证在客户机上进行，所以不需要在服务器上执行任何认证。

值 DCS 指示认证在主机或 AS/400 系统上进行。值 DCS_ENCRYPT 提供与 DCS 相同的行为，只是要加密通过网络发送的任何口令。如果您使用 APPC 和不向 DB2 服务器暴露该客户机口令的通信产品，可指定 DCS 以获取:
- 非 DRDA 客户机的 SERVER 类型认证
- DRDA 客户机的 CLIENT 类型认证

值 DCE 意味着在 DCE 服务器上使用“DCE 安全性服务”执行认证。值 DCE_SERVER_ENCRYPT 提供与 DCE 相同的行为，只是要加密通过网络发送的任何口令。DCE_SERVER_ENCRYPT 值只用于服务器。此值表明服务器可接受 DCE 认证或 SERVER_ENCRYPT 认证。

值 KERBEROS 意味着在 Kerberos 服务器上使用 Kerberos 认证安全性协议来执行认证。如果支持 Kerberos 安全性系统的服务器和客户机上的认证类型为 KRB_SERVER_ENCRYPT，则有效系统认证类型是 KERBEROS。如果客户机不支持 Kerberos 安全性系统，则有效的系统认证类型等价于 SERVER_ENCRYPT。

注: Kerberos 认证类型只有在运行 Windows 2000 的服务器上才受支持。

支持口令加密的认证值包括:
SERVER_ENCRYPT、DCS_ENCRYPT、DCE_SERVER_ENCRYPT 和 KRB_SERVER_ENCRYPT。就认证位置而言，这些值分别提供与 SERVER、DCS、DCE 和 KERBEROS 相同的功能，不同的是要在源加密流经的

任何口令，并要求在目标解密，这由在源编目的认证类型来指定。然后，可使用与认证位置匹配的加密和未加密值在客户机和网关之间、或网关和服务器之间选择不同的加密组合，而不会影响发生认证的位置。

有关这些值的等效数字和 API 常量，参考 *Administrative API Reference*。

有关何时及为何使用 DCE 或 DCS 以及与联合体数据库相关的认证问题的详情，参考*管理指南: 实现*中的“控制数据库存取”一章。

建议: 通常，缺省值 (SERVER) 足够使用。如果您有由 Kerberos、DB2 Connect 或 DCE 处理的入局请求，则参考*管理指南: 实现*中的“控制数据库存取”一章。

不需权限就允许编目 (catalog_noauth)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 客户机
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

带本地客户机和远程客户机的数据库服务器; 带本地客户机和远程客户机的数据库服务器

NO [NO (0) — YES (1)]

客户机; 带本地客户机的数据库服务器; 带本地客户机的卫星数据库服务器

YES [NO (0) — YES (1)]

此参数指定用户是否可以编目和取消编目数据库和节点，或者 DCS 和 ODBC 目录，而不需要 SYSADM 权限。此参数的缺省值 (0) 指示 SYSADM 权限是必需的。当将此参数设置为 1（是）时，就不需要 SYSADM 权限。

缺省数据库路径 (dftdbpath)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器

- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器
- 带本地客户机的卫星数据库服务器

参数类型 可配置的

缺省值（范围）

UNIX 实例拥有者的主目录 [任何现存的路径]

OS/2 和 Windows NT 安装了 DB2 的驱动器 [任何现存的路径]

此参数包含用来在数据库管理程序下创建数据库的缺省文件路径。如果创建数据库时未指定路径，则在 *dftdbpath* 参数指定的路径下创建该数据库。

在一个分区数据库环境中，您应该确保创建数据库所在的路径不是安装 NFS 的路径（在基于 UNIX 的平台上）或网络驱动器（在 Windows NT 环境中）。指定的路径必须实际存在于每个数据库分区服务器上。为避免混淆，最好指定以本地方式安装在每个数据库分区服务器上的路径。该路径的最大长度为 205 个字符。系统将节点名附加至该路径的尾部。

假如数据库可扩充至更大，并且很多用户可能要创建数据库（根据环境和意向），如果能在指定的位置上创建和存储所有数据库，通常会很方便。这样也能将数据库与其他应用程序和数据分离，保持了数据库完整性并便于备份和恢复。

对于基于 UNIX 的环境，*dftdbpath* 名称长度不能超过 215 个字符且必须为有效的绝对路径名。对于 OS/2 和 Windows NT，*dftdbpath* 可以是驱动器名，后面可以跟一个冒号。

建议: 如果有可能，将高容量数据库放在不同于其他被频繁存取的数据（如操作系统文件和数据库日志）所在的磁盘上。

DB2START/DB2STOP 必需的注册 (ss_logon)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器

参数类型 可配置的

缺省值（范围） YES [NO (0), YES (1)]

此参数只适用于 OS/2 环境。通过接收此参数的缺省值，在发出 DB2START 或 DB2STOP 之前，需要 LOGON 用户 ID 和口令。

信任所有客户机 (trust_allclnts)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） YES [NO, YES, DRDAONLY]

相关参数

- 第409页的『认证类型 (authentication)』
- 第414页的『可信赖客户机认证 (trust_clntauth)』

仅当将 *authentication* 参数设置为 CLIENT 时，此参数才是活动的。

此参数和 *trust_clntauth* 用于确定在何处向数据库环境验证用户。

通过接受此参数的缺省值 『YES』，所有的客户机都作为可信赖客户机对待。这意味着，服务器假定在客户机中安全性级别可用并且可在客户机上验证用户。

仅当将*认证*参数设置为 CLIENT 时，才可将此参数更改为 『NO』。如果此参数设置为 『NO』，则不可信赖客户机在连接到服务器时必须提供用户 ID 和口令组合。不可信赖客户机为没有用于认证用户的安全子系统的操作系统平台。

将此参数设置为 『DRDAONLY』，这将防止认证除 DRDA 客户机之外的 DB2 MVS 和 OS/390 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版的所有客户机。只有这些客户机受信赖，因此才可执行客户机端的认证。所有其他客户机必须提供用户 ID 和口令，以供服务器认证。

将 *trust_allclnts* 设置为 『DRDAONLY』 时，使用 *trust_clntauth* 参数来确定在何处认证客户机。如果将 *trust_clntauth* 设置为 『CLIENT』，则在客户机进行认证。如果将 *trust_clntauth* 设置为 『SERVER』，如果未提供口令则在客户机认证，如果提供了口令则在服务器认证。

有关可信赖客户机的详情，*参考管理指南: 实现*中的“为服务器选择认证方法”。

可信赖客户机认证 (trust_clntauth)

配置类型 数据库管理程序

适用于

- 带本地客户机和远程客户机的数据库服务器
- 带本地客户机的数据库服务器
- 带本地客户机和远程客户机的分区数据库服务器

参数类型 可配置的

缺省值（范围） CLIENT [CLIENT, SERVER]

相关参数

- 第409页的『认证类型 (authentication)』
- 第413页的『信任所有客户机 (trust_allclnts)』

此参数指定，当客户机提供用于连接的用户 ID 和口令组合时，是在服务器还是在客户机中认证可信赖的客户机。仅当将 *authentication* 参数设置为 CLIENT 时，此参数（和 *trust_allclnts*）才是活动的。如果不提供用户 ID 和口令，则假定客户机已验证该用户，因而在服务器上不需要进一步验证。

如果此参数设置为『CLIENT』（缺省值），则无须提供用户 ID 和口令组合，可信赖客户机就能连接，并假定操作系统已认证该用户。如果此参数设置为『SERVER』，则将在服务器上验证用户 ID 和口令。

CLIENT 的数值是 0。SERVER 的数值是 1。

有关可信赖客户机的详情，*参考管理指南: 实现*中的 “为服务器选择认证方法”。

第4部分 附录及附属资料

附录A. DB2 注册表变量和环境变量

以下是 DB2 注册表变量和环境变量的一个列表，这可能是在启动和运行时所要了解的内容。每个变量都有一个简短说明；其中有一些可能不适用于您的环境。

可使用下面的命令来查看所有受支持的注册表变量的列表:

```
db2set -lr
```

可使用下面的命令来更改当前会话的变量值:

```
db2set registry_variable_name=new_value
```

要更新环境变量，必须使用 set 命令，然后重新引导系统。

必须在发出 DB2START 命令之前设置已更改注册表变量的值。有关更改和使用注册表变量的详情，*参考管理指南: 计划*。

表 21. 一般注册表变量

变量名 说明	操作系统	值
DB2ACCOUNT	所有	缺省值=null
发送至远程主机的记帐字符串。有关详情，*参考 DB2 Connect 用户指南*。		
DB2BIDI	所有	缺省值=NO 值: YES 或 NO
此变量启用双向支持，*db2codepage* 变量用于声明要使用的代码页。有关双向支持的其他信息，*参考管理指南: 计划*中的“国家语言支持”附录。		
DB2CODEPAGE	所有	缺省值: 从语言 ID 中派生，由操作系统指定。
为数据库客户机应用程序指定呈示给 DB2 的数据的代码页。除非在 DB2 文档中明确声明或 DB2 服务要求这样做，否则，用户不应设置 *db2codepage*。将 *db2codepage* 设置为不受操作系统支持的值可产生非预期的结果。通常，不需要设置 *db2codepage*，因为 DB2 会自动从操作系统派生该代码页信息。		
DB2COUNTRY	所有	缺省值: 从语言 ID 中派生，由操作系统指定。
指定客户机应用程序的国家代码，它影响日期和时间格式。		
DB2DBDFT	所有	缺省值=null
指定用于隐式连接的数据库的数据库别名。如果应用程序没有数据库连接但发出了 SQL 语句，则只要在缺省数据库中定义了 DB2DBDFT 环境变量，将建立隐式连接。		

表 21. 一般注册表变量 (续)

变量名 说明	操作系统	值
DB2DBMSADDR	Windows 32 位操作系统	对于 Windows NT，缺省值=0x20000000，对于 Windows 95 则为 0x90000000 值：0x20000000 到 0xB0000000，增量为 0x10000
以十六进制格式指定缺省数据库管理程序的共享内存地址。如果 *db2start* 因共享内存地址冲突而失败，则可修改此注册表变量，以强制数据库管理程序实例在另一个地址分配其共享内存。		
DB2_DISABLE_FLUSH_LOG	所有	缺省值=OFF 值：ON 或 OFF
指定是否要禁止在任何联机备份中包括上一活动日志文件。 当联机备份完成时，作为备份的一部分截断、关闭和归档上一活动日志文件。这执行完整的备份，它包括复原该备份所需的所有日志。 如果您不想浪费“日志序号”(LSN) 空间的任何部分，则您可能希望禁止包括上一活动日志文件。每次截断活动日志文件时，LSN 都会增大，幅度为截断的空间的那一部分。如果每天都执行非常多次的联机备份，则您可能希望禁止包括上一活动日志文件。 如果您发现联机备份完成后不久您就接收到“日志满”信息，您也可能希望禁止包括上一活动日志文件。当截断日志文件时，保留的活动日志空间会增大，幅度为被截断日志的大小部分。在收回截断的日志文件之后，便释放活动日志空间。此回收操作在日志文件变为不活动片刻之后便会发生。在其间的这一小段时间内，您可能会接收到“日志满”信息。		
DB2DISCOVERYTIME	OS/2 和 Windows 32 位操作系统	缺省值=40 秒， 最小值=20 秒
指定 SEARCH discovery 将在 DB2 系统中搜索的时间。		
DB2INCLUDE	所有	缺省值=当前目录
指定在 DB2 PREP 处理期间处理 SQL INCLUDE 文本文件语句时要使用的路径。它提供一个有可能在其中找到包含文件的目录列表。参考 *Application Development Guide*，以获得如何在不同的预编译语言中使用 *db2include* 的有关说明。		
DB2INSTDEF	OS/2 和 Windows 32 位操作系统	缺省值=DB2
如果未定义 *DB2INSTANCE*，设置要使用的值。		
DB2INSTOWNER	Windows NT	缺省值=null
第一次创建实例时在 DB2 简要表注册表中创建的注册表变量。将此变量设置为该拥有实例的机器的名称。		

表 21. 一般注册表变量 (续)

变量名 说明	操作系统	值
DB2_LIC_STAT_SIZE	所有	缺省值=null 范围: 0 到 32767
该注册表变量用于确定包含系统许可证统计信息的文件的最大大小（以 MB 计）。零值关闭许可证统计信息收集。如果无法识别或未定义该变量，则该变量缺省为不受限制的大小。使用许可证中心显示该统计信息。		
DB2NBDISCOVERRCVBUFS	所有	缺省值=16 个缓冲区, 最小值=16 个缓冲区
此变量用于 NetBIOS 搜索查找。此变量指定一个客户机可接收的并行查找应答的数目。如果客户机接收到的并行应答多于此变量指定的数目，则多出的应答将被 NetBIOS 层废弃。缺省值为十六 (16) 个 NetBIOS 接收缓冲区。如果选择了比缺省值小的数，则使用缺省值。		
DB2OPTIONS	除 Windows 3.1 和 Macintosh 外的所有系统	缺省值=null
设置命令行处理器选项。		
DB2SLOGON	Windows 3.x	缺省值=null, 值: YES 或 NO
在 DB2 Windows 3.x 版中启用安全注册。如果 *db2slogon*=YES，DB2 不将用户 ID 和口令写入文件，而是使用一段内存来保存它们。当启用了 *db2slogon* 时，每次启动 Windows 3.x 时用户都必须注册。		
DB2TIMEOUT	Windows 3.x 和 Macintosh	缺省值=（未设置）
用于控制在长时间的 SQL 查询期间 Windows 3.x 和 Macintosh 客户机的超时期。当超时期到期后，将弹出一个对话框，询问是中断查询还是允许它继续执行。此变量的最小值是 30 秒。如果将 *db2timeout* 设置为 1 和 30 之间的一个值，将使用缺省最小值。如果将 *db2timeout* 设置为 0 或负值，则禁用超时功能部件。在缺省情况下，禁用此功能部件。		
DB2TRACENAME	Windows 3.x 和 Macintosh	缺省值=DB2WIN.TRC（在 Windows 3.x 上）， DB2MAC.TRC（在 Macintosh 上）
在 Windows 3.x 和 Macintosh 上，指定存储跟踪信息的文件名。每个系统的缺省值保存在当前实例目录中（例如，\sqllib\db2）。强烈建议您在命名跟踪文件时指定完整的路径名。		
DB2TRACEON	Windows 3.x 和 Macintosh	缺省值=NO 值: YES 或 NO

表 21. 一般注册表变量 (续)

变量名 说明	操作系统	值
在 Windows 3.x 和 Macintosh 上，打开跟踪以便将发生问题时的有关信息提供给 IBM。（除非您遇到不能解决的问题，否则，不建议您打开跟踪）。有关对客户机使用跟踪设施的信息，参考 *Troubleshooting Guide*。		
DB2TRCFLUSH	Windows 3.x 和 Macintosh	缺省值=NO 值: YES 或 NO
在 Windows 3.x 和 Macintosh 上，*db2trcflush* 和 *db2traceon*=YES 可一起使用。设置 *db2trcflush*=YES 将导致立即将每个跟踪记录写入跟踪文件中。这将大大减慢 DB2 系统的速度，因此缺省设置是 *db2trcflush* = NO。在应用程序挂起系统并需要重新引导系统的情况下，此设置将会很有用。设置此关键字将保证不会因重新引导而丢失跟踪文件和跟踪项。		
DB2TRCSYSERR	Windows 3.x 和 Macintosh	缺省值=1 值: 1-32767
指定在客户机关闭跟踪前要跟踪的系统错误数。缺省情况下，跟踪一个系统错误，然后关闭跟踪。		
DB2YIELD	Windows 3.x	缺省值=NO 值: YES 或 NO
指定在与远程服务器通信时 Windows 3.x 客户机的行为。当设置为 NO 时，则在客户机应用程序与远程服务器通信时，该客户机不会将 CPU 提供给其他 Windows 3.x 应用程序使用，且 Windows 环境暂停。必须等待该通信操作完成，才可以继续任何其他任务。当设置为 YES 时，系统正常工作。建议您在尝试运行应用程序前设置*db2yield*=YES。如果系统发生灾难性故障，将需要设置 *db2yield*=NO。对于应用程序开发，确保将您的应用程序编写为在等待通信操作完成的同时还可接受和处理 Windows 信息。		

表 22. 系统环境变量

变量名 说明	操作系统	值
DB2CONNECT_IN_APP_PROCESS	所有	缺省值=YES 值: YES 或 NO
将此变量设置为 NO 时，则强制 DB2 Connect 企业版上的 DB2 Connect 客户机在代理程序内运行。在代理程序内运行的一些优点是可监控本地客户机，且本地客户机可使用 SYSPLEX 支持。		
DB2DOMAINLIST	限于 Windows NT 服务器	缺省值=null 值: Windows NT 域名列表，域名由逗号(『,』)分隔。

表 22. 系统环境变量 (续)

变量名	操作系统	值
说明		
仅当“数据库管理程序”配置中设置了 CLIENT 认证时，此变量才有效；如果 Windows NT 域环境中需要来自 Windows NT 桌面的单一注册，则需要此变量。列表定义的是将要对其认证用户 ID 的域。 **注:** 设置此变量时，如果还指定了 CLIENT 认证，则只允许来自“DB2 通用数据库 Windows NT 版”客户机版本 7 （或更新版本）的连接请求。		
DB2ENVLIST	UNIX	缺省值: null
列出存储过程或用户定义函数的特定变量名。缺省情况下，**db2start** 命令过滤掉除带 **DB2** 或 **db2** 前缀之外的所有用户环境变量。如果必须将特定的注册表变量传送至存储过程或用户定义函数，您可以在 *db2envlist* 注册表变量中列出这些变量名。用一个或多个空格将每个变量名隔开。DB2 构造它自己的 PATH 和 LIBPATH，所以如果在 *db2envlist* 中指定了 PATH 或 LIBPATH，则将该变量名的实际值追加至 DB2 构造值的末尾。		
DB2INSTANCE	所有	在 OS/2 和 Windows 32 位操作系统上，缺省值=*db2instdef*。
用于指定在缺省情况下为活动的实例的环境变量。在 UNIX 平台，用户必须指定 *DB2INSTANCE* 的值。		
DB2INSTPROF	OS/2, Windows 3.x 和 Windows 32 位操作系统	缺省值: null
用于指定在 OS/2, Windows 3.x 和 Windows 32 位操作系统上实例目录的位置（如果与 *DB2PATH* 不同）的环境变量。		
DB2LIBPATH	UNIX	缺省值: null
在 *db2libpath* 注册表变量中指定值或 LIBPATH。如果用户 ID 已更改，则不能在父进程与子进程之间继承 LIBPATH 的值。因为 **db2start** 可执行程序由超级用户拥有，故 DB2 不能继承最终用户的 LIBPATH 设置。如果您在 *db2envlist* 注册表变量中列出变量名 LIBPATH，则还必须在 *db2libpath* 注册表值中指定 LIBPATH 的值。然后 **db2start** 可执行程序读取 *db2libpath* 的值，并将此值追加至 DB2 构造的 LIBPATH 的末尾。		
DB2NODE	所有	缺省值: null 值: 1 至 999
用于指定希望连接的 DB2 扩展企业版数据库分区服务器的目标逻辑节点。如果未设置此变量，目标逻辑节点将缺省为用该机器上的端口 `0` 定义的逻辑节点。		
DB2_PARALLEL_IO	所有	缺省值: null 值: *（表示每个表空间）或多个已定义表空间的逗号分隔列表
当对表空间容器读写数据时，如果数据库中的容器数大于 1，则 DB2 可使用并行 I/O。要对单一容器强制并行 I/O，请使用此注册表变量。在设置此注册表变量之后，发出 DB2STOP，然后输入 DB2START，以使更改生效。		

表 22. 系统环境变量 (续)

变量名 说明	操作系统	值
DB2PATH	OS/2, Windows 3.x 和 Windows 32 位操作系统	缺省值: （随操作系统变化）
用于指定在 OS/2, Windows 3.x 和 Windows 32 位操作系统上装有该产品的目录的环境变量。		
DB2_STRIPED_CONTAINERS	所有	缺省值: null 值: ON, null
当使用 RAID 设备作为表空间容器时，建议用等于或数倍于 RAID 条大小的数据块大小创建表空间。但是，由于存在单页容器标记，因此块不会按 RAID 条排列，并且在执行 I/O 请求时可能需要存取比最优情况更多的物理磁盘。 当使用 DMS 表空间容器时，此问题是通过将此标记自己的（全部）范围分配给它来避免的。这可避免上述问题，但它需要在该容器中占用额外的块。 在设置此注册表变量之后，发出 DB2STOP，然后输入 DB2START，以使更改生效。		

表 23. 通信变量

变量名 说明	操作系统	值
DB2CHECKCLIENTINTERVAL	AIX，仅适用于服务器	缺省值=0 值: 一个大于零的数值。
用于验证 APPC 客户机连接的状态。允许提前检测客户机的终止，而不是等到查询完成之后。当设置为零时，将不做任何检查。当设置为大于零的数值时，该值表示 DB2 内部工作单元。为了给予指导，提供了下列检查频率值: 低频使用 300; 中频使用 100; 高频使用 50。当执行数据库请求时更频繁地检查客户机状态会加长完成查询所用的时间。如果 DB2 工作负荷很重（即，它参与许多内部请求），则将 DB2CHECKCLIENTINTERVAL 设置为较低的值，较之于在工作负荷轻且大多数时间 DB2 都在等待的情况对性能的影响会更大。		
DB2COMM	所有，仅适用于服务器	缺省值=null 值: APPC, IPXSPX, NETBIOS, NPIPE, TCPIP 的任何组合
指定当启动数据库管理程序时所启动的通信管理程序。如果未设置，则不在服务器上启动任何 DB2 通信管理程序。		
DB2_FORCE_NLS_CACHE	AIX, HP_UX, Solaris	缺省值=FALSE 值: TRUE 或 FALSE

表 23. 通信变量 (续)

变量名	操作系统	值
说明		
用于消去多线程应用程序中锁定争用的可能。当此注册表变量为 『TRUE』 时，在线程第一次存取它时将保存代码页和国家代码信息。由此，高速缓存的信息就可用于请求此信息的任何其他线程。在某些情况中，这样可消去锁定竞用并导致有利于性能的结果。如果该应用程序更改了连接之间的本国语言环境设置，则不应该使用此设置。在此情况中，无论如何都不可能需要这样，原因是多线程应用程序一般不会更改它们的本国语言环境设置，因为这样做不是“线程安全”。		
DB2NBADAPTERS	OS/2 和 Windows NT	缺省值=0 范围: 0-15, 多个值应用逗号隔开
用于指定哪些本地适配器用于 DB2 NetBIOS LAN 通信。使用各自的逻辑适配器号来指定每一个本地适配器。		
DB2NBCHECKUPTIME	OS/2 和 Windows NT，仅适用于服务器	缺省值=1 分钟 值: 1-720
指定 NetBIOS 协议检查过程的每次调用之间的时间间隔。检查时间以分钟指定。 较低的值将确保 NetBIOS 协议检查更频繁地运行，并在发生意外的代理程序／对话终止时，释放内存和剩余的其他系统资源。		
DB2NBINTRLISTENS	OS/2 和 Windows NT，仅适用于服务器	缺省值=1 值: 1-10 多个值应用逗号隔开
指定将以异步方式发出以便于远程客户机中断的 NetBIOS 监听发送命令 (NCB) 的数目。为“中断活动”环境提供这种灵活性，以确保当服务器忙于为其他远程中断提供服务时，来自远程客户机的中断调用将能够建立连接。 将 *db2nbintrlistens* 设置为较低的值将保存服务器上的 NetBIOS 对话和 NCB。但是，在客户机中断较为常见的环境中，您可能需要将 *db2nbintrlistens* 设置为较高的值，以便对中断客户机更敏感。 **注:** 指定的值对位置敏感；它们与 *db2nbadapters* 的对应值的位置相关。		
DB2NBRECVBUFFSIZE	OS/2 和 Windows NT，仅适用于服务器	缺省值=4096 字节 范围: 4096-65536
指定 DB2 NetBIOS 协议接收缓冲区的大小。这些缓冲区是分配给 NetBIOS 接收 NCB 的。较低的值保存服务器内存，当客户机数据传送量较大时，可能需要更高的值。		

表 23. 通信变量 (续)

变量名 说明	操作系统	值
DB2NBBRECVNCBS	OS/2 和 Windows NT，仅适用于服务器	缺省值=10 范围: 1-99
指定服务器在操作期间将发出和维护的 NetBIOS ″receive_any″ 命令 (NCB) 的数目。可根据您的服务器连接的远程客户机数来调整此值。较低的值将保存服务器资源。 **注:** 使用的每个适配器可以有自己的唯一的接收 NCB 值，该值由 *db2nbbrecvncbs* 指定。指定的值对位置敏感；它们与 *db2nbadapters* 的对应值的位置相关。		
DB2NBRESOURCES	仅 OS/2 和 Windows NT 服务器	缺省值=null
指定在多场境环境中为 DB2 使用分配的 NetBIOS 资源量。此变量限于用于多场境客户机操作。		
DB2NBSENDNCBS	OS/2 和 Windows NT，仅适用于服务器	缺省值=6 范围: 1-720
指定服务器将保留使用的发送 NetBIOS 命令 (NCB) 的数目。可根据您的服务器连接的远程客户机数来调整此值。将 *db2nbsendncbs* 设置为较低的值将保存服务器资源。但是，您可能需要将它设置为较高的值，以防止当所有其他发送命令在使用时，服务器等待向远程客户机发送。		
DB2NBSESSIONS	OS/2 和 Windows NT，仅适用于服务器	缺省值=null 范围: 5-254
指定 DB2 应请求的、要保留给 DB2 使用的对话数。可以设置 *db2nbsessions* 的值，以便为使用 *db2nbadapters* 指定的每个适配器请求特定的对话。 **注:** 指定的值对位置敏感；它们与 *db2nbadapters* 的对应值的位置相关。		
DB2NBXTRANCBS	OS/2 和 Windows NT，仅适用于服务器	缺省值=每个适配器为 5 范围: 5-254
指定当发出 **db2start** 命令时服务器需要保留的 “额外” NetBIOS 命令 (NCB) 的数目。可以设置 *db2nbxtrancbs* 的值，以便为使用 *db2nbadapters* 指定的每个适配器请求特定的对话。		
DB2NETREQ	Windows 3.x	缺省值=3 范围: 0-25

表 23. 通信变量 (续)

变量名	操作系统	值
说明		
指定可在 Windows 3.x 客户机上并行运行的 NetBIOS 请求数。将此值设置得越大，将使用的 1MB 以下的内存就越多。当使用 NetBIOS 服务的并行请求数达到所设置的数值时，对 NetBIOS 服务的后续入局请求将放在一个队列中，在当前请求完成后这些请求将成为活动的。如果对 *db2netreq* 输入 0（零），则 Windows 数据库客户机使用 NetBIOS 等待选项以同步方式发出 NetBIOS 调用。在此方式下，数据库客户机只允许当前 NetBIOS 请求成为活动的，而且在当前请求完成前不处理另一个请求。这将会影响其他应用程序。仅仅为了反向兼容才提供值 0。极力建议您不使用 0。		
DB2RETRY	OS/2 和 Windows NT	缺省值=0 范围: 0-20 000
DB2 尝试重新启动 APPC 监听程序的次数。如果位于服务器／网关的 SNA 子系统停机，则可使用此简要表变量与 *db2retrytime* 来自动重新启动 APPC 监听程序而不中断使用其他协议的客户机通信。在这种方案中，不再需要停止然后重新启动 DB2 来重新启动 APPC 客户机通信。		
DB2RETRYTIME	OS/2 和 Windows NT	缺省值=1 分钟 范围: 0-7 200 分钟
以一分钟为增量， DB2 允许对启动 APPC 监听程序进行连续重试的间隔分钟数。如果位于服务器／网关的 SNA 子系统停机，则可使用此简要表变量与 *db2retry* 来自动重新启动 APPC 监听程序而不中断使用其他协议的客户机通信。在这种方案中，不再需要停止然后重新启动 DB2 来重新启动 APPC 客户机通信。		
DB2SERVICETPINSTANCE	OS/2 和 Windows NT	缺省值=null
用于支持来自 DB2 工作站 V.1 客户机或 DB2 MVS 数据库的入局 APPC 连接。当调用 **db2start** 命令时，指定的实例将对下列 TP 名称启动 APPC 监听程序: • DB2INTERRUPT • x'07'68 • x'07'6SN		
DB2SOSNDBUF	Windows 95 和 Windows NT	缺省值=32767
指定在 Windows 95 和 Windows NT 操作系统上的 TCP/IP 发送缓冲区的值。		
DB2SYSPLEX_SERVER	OS/2, Windows NT 和 UNIX	缺省值=null
指定在与 DB2 OS/390 版连接时是否启用 SYSPLEX。如果此注册表变量未设置（为缺省情况），或被设置为一个非零值，则允许使用。如果此注册表变量被设置为零 (0)，则禁止使用。当设置为零时，则对网关禁用 SYSPLEX，而不管 DCS 数据库目录项是如何指定的。有关详情，参见 *Command Reference* 和 **CATALOG DCS DATABASE** 命令。		

表 23. 通信变量 (续)

变量名 说明	操作系统	值
DB2TCPCONNMGRS	所有	在串行机器上，缺省值=1；在对称多处理器机器上，缺省值为处理器数平方根进位舍入到最大连接管理程序数 (8)。 值：1 至 8
如果未设置此注册表变量，则创建缺省数目个连接管理程序。如果设置了此注册表变量，则此处指定的值覆盖缺省值。将创建指定数目个 TCP/IP 连接管理程序，最多创建 8 个。如果指定小于 1 的值，则 DB2TCPCONNMGRS 设置为值 1，并记录一个警告，指示值超出范围。如果指定大于 8 的值，则 DB2TCPCONNMGRS 设置为值 8，并记录一个警告，指示值超出范围。1 与 8 之间的值按给定的那样使用。当创建多于 1 个连接管理程序时，如果同时接收多个客户机连接，则连接处理能力应能得到改进。如果用户在 SMP 机器上工作，或修改了 DB2TCPCONNMGRS 注册表变量，则可能有附加的 TCP/IP 连接管理程序进程（在 UNIX 上）或线程（在 OS/2 和 Windows 操作系统上）。附加的进程或线程需要附加的存储器。		
DB2_VI_ENABLE	Windows NT	缺省值=OFF 值：ON 或 OFF
指定是否使用“虚拟接口体系结构”(VIA) 通信协议。如果此注册表变量为 『ON』，则 FCM 将使用 VI 来进行节点间通信。如果此注册表变量为 『OFF』，则 FCM 将使用 TCP/IP 来进行节点间通信。 **注：** 此注册表变量的值必须在该实例的所有数据库分区中相同。		
DB2_VI_VIPL	Windows NT	缺省值=`vipl.dll`
指定 DB2 将使用的“虚拟接口供应商库”(VIPL) 的名称。为成功地装入库，在此注册表变量中使用的库名必须在 *PATH* 用户环境变量中。当前受支持的实施全部使用相同的库名。		
DB2_VI_DEVICE	Windows NT	缺省值=null 值：`nic0`
指定与“网络接口卡”(NIC) 相关的设备或“虚拟接口供应商实例”的符号名。“独立硬件供应商”(IHV) 都生产它们自己的 NIC。每个 Windows NT 机器上只允许一个 (1) NIC；同一个物理机器上的多个逻辑节点将共享同一个 NIC。当前受支持的实施全部使用相同的符号名。		

表 24. DCE 目录变量

变量名 说明	操作系统	值
DB2DIRPATHNAME	OS/2，UNIX 和 Windows 32 位操作系统	缺省值=null

表 24. DCE 目录变量 (续)

变量名	操作系统	值
说明		
指定数据库管理程序配置文件中 DIR_PATH_NAME 参数值的临时替换项。如果使用目录服务器且未显式编目 CONNECT 语句或 ATTACH 命令的目标，则该目标与 DB2DIRPATHNAME（如果指定）合并，以构成全限定 DCE 名。 **注:** *db2dirpathname* 变量对实例的全局名没有影响，该全局名始终由数据库管理程序配置参数 DIR_PATH_NAME 和 DIR_OBJ_NAME 标识。		
DB2CLIENTADPT	OS/2 和 Windows 32 位操作系统	缺省值=null 范围: 0-15
指定 OS/2 和 Windows 32 位操作系统上 NETBIOS 协议的客户机适配器号。*db2clientadpt* 值替换数据库管理程序配置文件中的 DFT_CLIENT_ADPT 参数值。		
DB2CLIENTCOMM	OS/2, UNIX 和 Windows 32 位操作系统	缺省值=null
指定数据库管理程序配置文件中 DFT_CLIENT_COMM 参数值的临时替换项。如果未指定 DFT_CLIENT_COMM 和 *db2clientcomm*，则使用在该对象中找到的第一个协议。如果指定了其中一个参数或同时指定了两个参数，将只使用第一个匹配的协议。在任何一种情况下，如果第一次连接失败，都不再重试。		
DB2ROUTE	OS/2, UNIX 和 Windows 32 位操作系统	缺省值=null
指定当客户机用另一种数据库协议连接数据库时使用的“路由选择信息对象”的名称。*db2route* 值替换数据库管理程序配置文件中的 ROUTE_OBJ_NAME 参数值。		

表 25. 命令行变量

变量名	操作系统	值
说明		
DB2BQTIME	所有	缺省值=1 秒 最大值: 1 秒
指定命令行处理器前端在检查后端进程是否是活动的并建立与它的连接之前，将休眠的时间。		
DB2BQTRY	所有	缺省值=60 次重试 最小值: 0 次重试
指定命令行处理器前端进程尝试确定后端进程是否已活动的次数。它与 *db2bqtime* 一起使用。		
DB2IQTIME	所有	缺省值=5 秒 最小值: 1 秒

表 25. 命令行变量 (续)

变量名	操作系统	值
说明		
指定命令行处理器后端进程在输入队列上等待前端进程传送命令的时间。		
DB2RQTIME	所有	缺省值=5 秒 最小值: 1 秒
指定命令行处理器后端进程等待前端进程提出请求的时间。		

表 26. MPP 配置变量

变量名	操作系统	值
DB2ATLD_PORTS	AIX, Solaris 和 Windows NT 上的 DB2 UDB EEE	缺省值=6000:6063 值: num1:num2, 这两个值都介于 1 和 65535 之间, 且 num1<=num2
指定用于“自动装入”实用程序的内部 TCPIP 通信的端口号的范围。如果未设置, “自动装入”则使用内部缺省端口范围 6000:6063。当有其他应用程序使用“自动装入”缺省端口范围时, 此变量可用于选择替代端口范围。		
DB2ATLD_PWFILE	AIX, Solaris 和 Windows NT 上的 DB2 UDB EEE	缺省值=null 值: 文件路径表达式
指定一个文件的路径, 该文件包含在“自动装入”认证期间使用的口令。如果未设置, 则“自动装入”从其配置文件中抽取口令, 或以交互式方式提示您输入口令。使用此变量将解决口令安全性问题, 并允许“自动装入”配置信息和认证信息分开。		
DB2CHGPWD_EEE	AIX 和 Windows NT 上的 DB2 UDB EEE	缺省值=null 值: YES 或 NO
指定是否允许其他用户更改 AIX 或 Windows NT EEE 系统上的口令。必须确保在 Windows NT 上使用 Windows NT 域控制器或在 AIX 上使用 NIS 来集中维护所有分区或节点的口令。如果没有集中维护, 在所有分区或节点之间口令可能会不一致。这可能会导致只在用户为了更改而连接的数据库分区中更改口令。为了修改这个全局注册表变量, 您必须在根目录和 DAS 实例上。		
DB2_FORCE_FCM_BP	AIX	缺省值=NO 值: YES 或 NO

表 26. MPP 配置变量 (续)

变量名	操作系统	值
此注册表变量适用于使用多逻辑分区时的 DB2 UDB EEE AIX 版。当发出 DB2START 时，DB2 从数据库全局内存中分配 FCM 缓冲区，或者，如果该处没有足够的空间，则从同一个物理机器上（该实例的）所有 FCM 精灵程序所用的一个单独的共享内存段中分配。选择哪种情况很大程度上取决于要创建的 FCM 缓冲区的数目（也就是由 FCM_NUM_BUFFERS 数据库管理程序配置参数来确定）。如果将此注册表变量设置为 YES，则总是在单独的内存段中创建 FCM 缓冲区。当在单独的内存段中创建 FCM 缓冲区时，同一个物理节点上不同逻辑分区的 FCM 精灵程序之间的通信通过共享内存来进行。否则，同一个节点上的 FCM 精灵程序将通过 UNIX 套接字进行通信。使用此方式通过共享内存通信的优点是更快。其缺点是可用于其他用途（最明显的是用于数据库缓冲池）的共享内存段更少。它减少了数据库缓冲池的最大大小。		
DB2_NUM_FAILOVER_NODES	AIX、Solaris 和 Windows NT 上的 DB2 UDB	缺省值：2 值：0 至逻辑节点数
指定可以用作高可用性环境中的故障恢复节点的节点数。借助高可用性，当某个节点失效时，故障恢复节点便可作为另一主机上的第二个逻辑节点重新启动。配合此变量使用的数字确定为用于故障恢复节点的 FCM 资源保留多少内存。 例如，主机 A 有两个逻辑节点：1 和 2；主机 B 有两个逻辑节点：3 和 4。假定 DB2_NUM_FAILOVER_NODES 设置为 2。DB2START 期间，主机 A 和主机 B 都将为 FCM 保留足够的内存，以便最多可以管理 4 个逻辑节点。于是，即使一个主机失效，失效主机的逻辑节点也可以在另一主机上重新启动。		
DB2PORTRANGE	Windows NT	值：nnnn:nnnn
将此值设置为 FCM 使用的 TCP/IP 端口范围，以便在另一台机器上创建的任何附加分区也有同样的端口范围。		

表 27. SQL 编译程序变量

变量名 说明	操作系统	值
DB2_ANTIJOIN	所有	缺省值=NO 值：YES 或 NO
此注册表变量适用于“DB2 通用数据库 EEE”环境。它指定优化器是否找寻机会来将 『NOT EXISTS』子查询转换成 DB2 能够更有效地进行处理的反连接。		
DB2_CORRELATED_PREDICATES	所有	缺省值=OFF 值：ON 或 OFF
当在一个连接中的相关列上存在唯一的索引且此注册表变量为 ON 时，优化器将试图检测并补偿连接谓词的相关性。当此注册表变量为 ON 时，优化器使用唯一索引统计信息的 KEYCARD 信息来检测相关性情况，动态调整相关谓词的组合选择性，从而获得该连接大小和成本的更准确估计。		

表 27. SQL 编译程序变量 (续)

变量名 说明	操作系统	值
DB2_HASH_JOIN 将散列连接指定为当编译存取方案时可能的连接方法。	所有	缺省值=NO 值: YES 或 NO
DB2_LIKE_VARCHAR 指定优化器如何使用如下形式的谓词 `COLUMN LIKE '%XXXXXX%'` 其中 xxxxxx 是任意一个字符串。 对于所有谓词，优化器必须估计有多少行与该谓词匹配。对于以 % 字符为前导和尾部字符的 LIKE 谓词，优化器假设要匹配的 COLUMN 具有这样一个结构，即它由一系列元素并置在一起构成整个列。然后优化器根据包围在 % 字符之间的字符串的长度，估计每个元素的长度。	所有	缺省值=NO 值: YES, NO 或 0 与 6.001 之间的浮点常数

DB2_NEW_CORR_SQ_FF
将它设置为 『ON』 时，影响 SQL 优化器为某些子查询谓词计算的选择值。它可用于改善一种等式子查询谓词的选择值的精确性，这种等式子查询谓词使用该子查询的 SELECT 列表中的 MIN 或 MAX 聚合函数。例如:
`SELECT * FROM T WHERE`
`T.COL = (SELECT MIN(T.COL)`
`FROM T WHERE ...)`
	所有	缺省值=OFF
		值: ON 或 OFF

DB2_PRED_FACTORIZE	所有	缺省值=NO 值: YES 或 NO

表 27. SQL 编译程序变量 (续)

变量名	操作系统	值
说明		

指定优化器是否找寻机会来从析取项抽取附加的谓词。在某些情况下，附加的谓词可改变中间的和最终的结果集的估算基数。使用以下查询:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`
优化器可生成下列附加谓词:
`SELECT n1.empno,`
`n1.lastname`
`FROM employee n1,`
`employee n2`
`WHERE n1.lastname IN`
`('SMITH', 'JONES')`
`AND n2.lastname IN`
`('SMITH', 'JONES')`
`AND`
`((n1.lastname='SMITH'`
`AND n2.lastname='JONES')`
`OR (n1.lastname='JONES'`
`AND n2.lastname='SMITH'))`		

表 28. 性能变量

变量名	操作系统	值
说明		
DB2_AVOID_PREFETCH	所有	缺省值=OFF, 值: ON 或 OFF
指定在崩溃恢复期间是否应使用预读取。如果 *db2_avoid_prefetch*=ON，则不使用预读取。		
DB2_BINSORT	AIX	缺省值=YES 值: YES 或 NO

表 28. 性能变量 (续)

变量名	操作系统	值
说明		
启用新的排序算法，以减少排序的 CPU 时间和经过时间。此新的算法将 DB2 UDB 的非常有效的整数排序技术扩展到所有排序数据类型，如 BIGINT, CHAR, VARCHAR, FLOAT 和 DECIMAL 以及这些数据类型的组合。要启用此新算法，使用以下命令: `db2set DB2_BINSORT = yes`		
DB2BPVARS	Windows NT	缺省值=路径

对包含调节缓冲池时使用的参数的文件指定路径。当前受支持的参数为: NT_SCATTER_DMSFILE, NT_SCATTER_DMSDEVICE 和 NT_SCATTER_SMS。
 	
对于这些参数中的每一个，缺省值为零（或 OFF）；且可能的值包括: 零（或 OFF）和 1（或 ON）。每个参数用于对相应类型的容器打开分散读取。仅当在注册表中将 DB2NTNOCACHE 设置为 ON 时才能启用（打开）每个参数。如果将 DB2NTNOCACHE 设置为 OFF（或未设置），会向 db2diag.log 中写入一条警告信息，且仍然禁止分散读取。对于对相应类型的容器有大量顺序预读取的系统，并且您已经决定将 DB2NTNOCACHE 设置为 OFF，则建议使用这些参数。
 	
下面是说明如何设置该文件路径的一个示例:
`db2set DB2BPVARS =`
`f:\BPVARSFILE`
 	
该文件的内容是使用以下格式的一个参数:
`parameter=value`		

DB2CHKPTR	所有	缺省值=OFF, 值: ON 或 OFF
指定是否需要对输入进行指针检查。		
DB2_DARI_LOOKUP_ALL	所有	缺省值=OFF 值: ON 或 OFF
指定在 *sqllib* 子目录下的 *function* 子目录中查找以及在 *sqllib* 子目录下 *function* 子目录的 *unfenced* 子目录中查找之前，UDB 服务器是否要为 ALL DARI 和存储过程执行目录查找。 **注:** 对于位于上面提到的目录中的 PARAMETER TYPE DB2DARI 存储过程，将此值设置为 『ON』 将会降低性能，因为在搜索 function 目录之前可能会在 EEE 配置中的另一个节点上执行目录查找。		
DB2_EXTENDED_OPTIMIZATION	所有	缺省值=OFF 值: ON 或 OFF

表 28. 性能变量 (续)

变量名	操作系统	值
说明		
指定查询优化器是否使用优化扩充功能来改进查询性能。此扩充功能并不能在所有环境中改进查询性能。应执行测试来确定个别的查询性能改进。		
DB2MEMDISCLAIM	AIX	缺省值=null 值: YES 或 NO
根据执行的工作负荷和存储池代理程序配置，可能会遇到这样一种情况: 甚至在代理程序空闲时，每个 DB2 代理程序开销的内存会保持在 32 MB 以上。这种行为是期望的行为，并且通常会带来良好的性能，因为它将内存保留起来以进行快速重新使用。但是，在内存不足的系统上，这可能是一个不期望的副作用。要避免这种情况，发出以下命令: `db2set DB2MEMDISCLAIM = yes` 放弃内存将告知 AIX 操作系统停止该区域的调页操作，以便它不再占用任何实存。将 DB2MEMDISCLAIM 设置为 『YES』 将告知 DB2 UDB 在内存释放时放弃某些或全部内存，这由 DB2MEMMAXFREE 给出的值决定。如果 DB2MEMMAXFREE 为空，则在释放时收回所有内存。如果 DB2MEMMAXFREE 给出了一个值，则释放时仅收回某些内存（以 DB2MEMMAXFREE 中给出的值为限）。这确保内存只要一被释放，便可供其他进程使用。另见 DB2MEMMAXFREE。这两个注册表变量能一起工作。		
DB2MEMMAXFREE	AIX	缺省值=null 值: 4000000 到 256000000

指定每个 DB2 代理程序保留的空闲内存容量。可将此变量设置为 4 MB 至 256 MB 之间的值。如果您使用此功能部件，建议您指定 8 MB 的值:
`db2set DB2MEMMAXFREE`
`  = 8000000`
 	
另见 DB2MEMDISCLAIM。这两个注册表变量能一起工作。		

DB2_MMAP_READ	AIX	缺省值=ON, 值: ON 或 OFF
与 *db2_mmap_write* 一起使用，以允许 DB2 使用 mmap 作为 I/O 的一个替代方法。在大多数环境中，当多个进程写入同一文件的不同部分时，应使用 mmap 以避免操作系统锁定。但是，也许您已从 “并行版 V1.2” 迁移，在 V1.2 中缺省值是 OFF，它允许 AIX 把从 JFS 文件系统读取的 DB2 数据高速缓存到内存（在缓冲池外）中。如果您希望获得相当于 DB2 UDB 的性能，您可以增加缓冲池的大小，或将 *db2_mmap_read* 和 *db2_mmap_write* 更改为 `OFF`。		
DB2_MMAP_WRITE	AIX	缺省值=ON 值: ON 或 OFF

表 28. 性能变量 (续)

变量名	操作系统	值
说明		
与 *db2_mmap_read* 一起使用，以允许 DB2 使用 mmap 作为 I/O 的替代方法。在大多数环境中，当多个进程写入同一文件的不同部分时，应使用 mmap 以避免操作系统锁定。但是，也许您已从“并行版 V1.2”迁移了，其中缺省值是 OFF，它允许 AIX 把从 JFS 文件系统读取的 DB2 数据高速缓存到内存（在缓冲池外）。如果您希望获得相当于 DB2 UDB 的性能，您可以增加缓冲池的大小，或将 *db2_mmap_read* 和 *db2_mmap_write* 更改为 OFF。		
DB2_NO_PKG_LOCK	所有	缺省值=OFF 值：ON 或 OFF
允许“全局 SQL 高速缓存”运行，而不必使用程序包锁定来保护高速缓存的程序包项目。（程序包锁定是内部系统锁定。） 要提高性能（因为获取和释放锁定需要时间），您现在可选择用“无程序包锁定”方式工作。在此方式下，不允许某些数据库操作。这些操作可能包括：作废程序包的操作、使程序包不起作用的操作和直接更改程序包的操作。		
DB2NTMEMSIZE	Windows NT	缺省值=（随内存段变化）

Windows NT 要求在 DLL 初始化时保留所有共享内存段，以保证在不同的进程之间的地址匹配。引入了 *DB2NTMEMSIZE*，以允许用户在必要时替换 Windows NT 上的 DB2 缺省值。在大多数情况下，缺省值应足够使用。内存段、缺省大小和替换选项为：1) 数据库内核：缺省大小为 16777216 (16 MB)；替换选项为 DBMS: <字节数>。2) 并行 FCM 缓冲区：缺省大小为 22020096 (21 MB)；替换选项为 FCM: <字节数>。3) 数据库管理 GUI：缺省大小为 33554432 (32 MB)；替换选项为 DBAT: <字节数 >。4) 防护存储过程：缺省大小为 16777216 (16 MB)；替换选项为 APLD: <字节数>。通过用分号 (;) 来隔开替换选项，可替换多个段。例如，要将数据库内核限制在大约 256K，将 FCM 缓冲区限制在大约 64 MB，使用:
`db2set DB2NTMEMSIZE=`
`DBMS:256000;FCM:64000000`		

DB2NTNOCACHE	Windows NT	缺省值=OFF 值：ON 或 OFF
指定 DB2 是否用 NOCACHE 选项打开数据库文件。如果 *db2ntnocache*=ON，则禁用文件系统高速缓存。如果 *db2ntnocache*=OFF，则操作系统高速缓存 DB2 文件。这适用于除包含 LONG FIELDS 或 LOBS 的文件以外的所有数据。禁用系统高速缓存使数据库有更多内存可用，这样可以增加缓冲池或排序堆。		
DB2NTPRICLASS	Windows NT	缺省值=null 值：R、H、任何其他值

表 28. 性能变量 (续)

变量名	操作系统	值
说明		
设置 DB2 实例（程序 DB2SYSCS.EXE）的优先级类别。有三种优先级类别: • NORMAL_PRIORITY_CLASS （缺省值优先级类别） • REALTIME_PRIORITY_CLASS （使用 『R』 设置） • HIGH_PRIORITY_CLASS （使用 『H』 设置） 此变量与个别线程优先级（使用 *DB2PRIORITIES* 设置）一起使用，以确定此系统中 DB2 线程相对于其他线程的绝对优先级。 **注:** 使用此变量时应当小心。误用可能会对整个系统的性能有负面影响。 有关详情，请参考 Win32 文档中的 *SetPriorityClass()* API。		
DB2NTWORKSET	Windows NT	缺省值=1,1
用于修改 DB2 可用的最小和最大工作集大小。缺省值情况下，当 Windows NT 未处于调页状态时，进程的工作集可按需要尽量增大。但是，当发生调页时，进程可拥有的最大工作集大约是 1 MB。DB2NTWORKSET 允许重设此缺省行为。 使用语法 *db2ntworkset*=min,max 为 DB2 指定 DB2NTWORKSET，其中 min 和 max 以兆字节表示。		
DB2_OVERRIDE_BPF	所有	缺省值=未设置 值: 正的页数
以页为单位指定在数据库激活时或第一次连接时要创建的缓冲池的大小。在数据库激活或第一次连接期间因内存受限制而失败时，此选项就很有用。如果数据库管理程序连 16 页的最小缓冲池都无法建立，那么用户可在使用此环境变量指定一个更小的页数之后再试。出现内存不足可能是因为实内存短缺（这种情况很少发生）；或者因为数据库管理程序试图分配大而没有精确配置的缓冲池。如果设置了此值，它将替换当前的缓冲池大小。		
DB2PRIORITIES	所有	值的设置是与平台相关。
控制 DB2 进程和线程的优先级。		
DB2_RR_TO_RS	所有	缺省值=NO 值: YES 或 NO
当设置为 YES 时，RR 隔离级将很明显地降级为用户表的 RS。在数据库管理程序实例中不再提供 RR 语义学。如果应用程序不需要 RR 语义学，则此注册表变量可用于减少在 RR 下有时可能发生的下一关键字锁定争用问题。		
DB2_SORT_AFTER_TQ	所有	缺省值=NO 值: YES 或 NO

表 28. 性能变量 (续)

变量名	操作系统	值
说明		
指定当接收端要求对数据排序且接收节点数与发送节点数相等时，优化器如何在分区数据库中使用定向的表队列。 当 *DB2_SORT_AFTER_TQ*= NO 时，优化器往往会在发送端排序，而在接收端合并行。 当 *DB2_SORT_AFTER_TQ*= YES 时，优化器往往会发送未排序的行，在接收端不合并，而在接收完所有的行之后才在接收端对这些行排序。		

表 29. 数据链路变量

变量名	操作系统	值
说明		
DLFM_BACKUP_DIR_NAME	AIX, Windows NT	缺省值: null 值: TSM 或任何有效路径
指定要使用的备份设备。在运行期，即使在 TSM 与路径之间更改此注册表变量的设置，也不会移动归档的文件。只会将新备份放在新位置中。而不会移动先前归档的文件。		
DLFM_BACKUP_LOCAL_MP	AIX, Windows NT	缺省值: null 值: DFS 系统中的本地安装点的任何有效路径
指定 DFS 系统中的安装点的全限定路径。当给出路径时，将使用此路径，而不是使用 DLFM_BACKUP_DIR_NAME 给出的路径。		
DLFM_BACKUP_TARGET	AIX, Windows NT	缺省值: null 值: LOCAL, TSM, XBSA
指定使用的备份的类型。		
DLFM_BACKUP_TARGET_LIBRARY	AIX, Windows NT	缺省值: null 值: DLL 或共享库名的任何有效路径
指定 DLL 或共享库的全限定路径。这个库使用 *libdfmxbsa.a* 库来装入。		
DLFM_ENABLE_STPROC	AIX, Windows NT	缺省值: NO 值: YES 或 NO
指定是否使用存储过程来链接文件的组。		
DLFM_FS_ENVIRONMENT	AIX, Windows NT	缺省值: NATIVE 值: NATIVE 或 DFS

表 29. 数据链路变量 (续)

变量名	操作系统	值
说明		
指定“数据链路”服务器的操作环境。NATIVE 指示“数据链路”服务器处于单机环境中，服务器可以接管它自己的机器上的文件。DFS 指示“数据链路”服务器处于分布式文件系统 (DFS) 环境中，服务器可以接管整个文件系统中的文件。不允许混合 DFS 文件集和本机文件系统。		
DLFM_GC_MODE	AIX, Windows NT	缺省值: PASSIVE 值: SLEEP, PASSIVE 或 ACTIVE
指定在数据链路服务器上对无用信息文件收集的控制。当设置为 SLEEP 时，不进行无用信息收集。当设置为 PASSIVE 时，仅当没有其他事务运行时才运行无用信息收集。当设置为 ACTIVE 时，即使其他事务在运行也运行无用信息收集。		
DLFM_INSTALL_PATH	AIX, Windows NT	缺省值 在 AIX 上: /usr/lpp/db2_06_00 /adm 在 NT 上: DB2PATH /bin 范围: 任何有效路径
指定数据链路可执行文件的安装路径。		
DLFM_LOG_LEVEL	AIX, Windows NT	缺省值: LOG_INFO 值: LOG_CRIT, LOG_DEBUG, LOG_ERR, LOG_INFO, LOG_NOTICE, LOG_WARNING
指定要记录的诊断信息级。		
DLFM_PORT	除 Windows 3.n 以外的所有平台	缺省值: 50100 值: 任何有效的端口号
指定用于与运行 DB2 Data Links Manager 的数据链路服务器通信的端口号。只在表包含『DATALINKS』 列时才使用此环境变量。		

表 30. 其他变量

变量名	操作系统	值
说明		
DB2ADMINSERVER	OS/2, Windows 95, Windows NT 和 UNIX	缺省值=null
指定将哪个 DB2 实例设置为“DB2 管理服务器”。		
DB2CLIINIPATH	所有	缺省值=null

表 30. 其他变量 (续)

变量名	操作系统	值
说明		
用于替换 DB2 CLI/ODBC 配置文件 (db2cli.ini) 的缺省路径并指定客户机上另一个位置。指定的值必须是客户机系统上的有效路径。		
DB2DEFPREP	所有	缺省值=NO 值: ALL, YES 或 NO
对在 DEFERRED_PREPARE 预编译选项可用之前预编译的应用程序，模拟此选项的运行期行为。例如，如果 DB2 v2.1.1 或更早的应用程序在 DB2 v2.1.2 或更高版本的环境中运行，则可能使用 *db2defprep* 来指示期望的“延迟准备”行为。		
DB2_DJ_COMM	所有	缺省值=null 值包括: libdrda.a, libsqlnet.a, libnet8.a, libdrda.dll, libsqlnet.dll, libnet8.dll 等。
指定当启动数据库管理程序时装入的封装器库。指定此变量将减少装入频繁使用的封装器的运行期成本。支持其他操作系统的其他值（.dll 扩展名用于 Windows NT 操作系统；.a 扩展名用于 AIX 操作系统）。库名随协议和操作系统变化。此变量不可用，除非将数据库管理程序参数 *federated* 设置为 YES。		
DB2DMNBCKCTLR	Windows NT	缺省值=null 值: ? 或域名
如果您知道 DB2 服务器充当备份域控制器的域的名称，则设置 *db2dmnbckctlr*=DOMAIN_NAME。DOMAIN_NAME 必须是大写的。要让 DB2 确定本地机器是其备份域控制器的域，设置 *db2dmnbckctlr*=?。如果 *db2dmnbckctlr* 简要表变量未设置或被设置为空白，则 DB2 在主域控制器上执行认证。 **注:** 缺省情况下，DB2 不使用现存的备份域控制器，因为备份域控制器可以脱离与主域控制器的同步，从而导致安全性漏洞。当更新了主域控制器的安全性数据库但未将该更改传送至备份域控制器时，则可能引起脱离同步。如果存在网络等待时间或计算机浏览器服务未运行，也可能发生此情况。		
DB2_ENABLE_LDAP	所有	缺省值=NO 值: YES 或 NO
指定是否使用“轻量目录存取协议”(LDAP)。LDAP 是一种目录服务存取方法。		
DB2_FALLBACK	Windows NT	缺省值=OFF 值: ON 或 OFF
此变量允许在待缓处理期间强制所有数据库连接断开。它与故障恢复支持一起用于带有“Microsoft 群集服务器” (MSCS) 的 Windows NT 环境中。如果 *DB2_FALLBACK* 没有设置或设置为 OFF，且在待缓期间数据库连接一直存在，则不会导致 DB2 资源脱机。这意味着待缓处理将失败。		
DB2_FORCE_TRUNCATION	所有	缺省值=NO 值: YES 或 NO

表 30. 其他变量 (续)

变量名	操作系统	值
说明		
在重新启动恢复期间使用。当设置为 『NO』 时，如果确定一个坏页正在过早地停止重新启动恢复，则它会暂停重新启动恢复（亦即，尚未读取所有活动日志）。这通常是由一个日志中的坏页引起的。用户可将此变量设置为 『YES』，来通知重新启动恢复就象到达了日志末尾一样，它应继续处理。在把此变量设置为 『YES』 之后，当数据库再次变为活动时，那些在重新启动恢复期间未被读取的日志将被覆盖。缺省值为 『NO』，如果未发现坏页，它**不**会继续执行。此变量只应在 IBM 服务人员的指导下使用。		
DB2_GRP_LOOKUP	Windows NT	缺省值=null 值: LOCAL, DOMAIN
此变量用于告知 DB2 到何处验证用户帐号并执行组成员查找。将该变量设置为 LOCAL 以强制 DB2 总是在 DB2 服务器上枚举组和验证用户帐号。将此变量设置为 DOMAIN 以强制 DB2 总是在用户帐号所属的 Windows NT 域上枚举组和验证用户帐号。		
DB2LDAP_BASEDN	所有	缺省值=null 值: 任何有效基本域名。
指定 LDAP 目录的基本域名。		
DB2LDAPCACHE	所有	缺省值=YES 值: YES 或 NO

指定要启用 LDAP 高速缓存。此高速缓存用来编目本地机器上的数据库、节点和 DCS 目录。
 	
要确保高速缓存中有最新的项目，请执行下列各项:
`REFRESH LDAP DB DIR`
`REFRESH LDAP NODE DIR`
 	
这些命令从数据库目录和节点目录中更新和除去不正确的项目。		

DB2LDAP_CLIENT_PROVIDER	限于 Windows 95/98/NT/2000	缺省值=null（如果可用的话，使用 Microsoft；否则使用 IBM。） 值: IBM 或 Microsoft
当在 Windows 环境中运行时，DB2 支持使用 Microsoft LDAP 客户机或 IBM LDAP 客户机来存取 LDAP 目录。这个注册表变量用来显式地选择 DB2 要使用的 LDAP 客户机。 **注:** 要显示此注册表变量的当前值，请使用 `db2set` 命令: `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	所有	缺省值=null 值: 任何有效主机名。

表 30. 其他变量 (续)

变量名 说明	操作系统	值
指定 LDAP 目录的位置的主机名。		
DB2LDAP_SEARCH_SCOPE	所有	缺省值= DOMAIN 值: LOCAL, DOMAIN, GLOBAL
指定以 “轻便目录存取协议” (LDAP) 在分区或域中查找信息的搜索范围。『LOCAL』 禁止在 LDAP 目录中搜索。『DOMAIN』 只在当前目录分区中以 LDAP 搜索。『GLOBAL』 在所有目录分区中以 LDAP 搜索，直到找到对象为止。		
DB2LOADREC	所有	缺省值=null
用于在前滚期间替换装入副本的位置。如果用户更改了装入副本的物理位置，则在发出前滚之前必须设置 *db2loadrec*。		
DB2LOCK_TO_RB	所有	缺省值=null 值: 语句
指定锁定超时是导致回滚整个事务还是仅回滚当前语句。如果将 *db2lock_to_rb* 设置为 STATEMENT，则锁定超时只导致回滚当前语句。任何其他设置都导致事务回滚。		
DB2NOEXITLIST	所有	缺省值=OFF 值: ON 或 OFF
如果已定义此变量，它将指示 DB2 不要在应用程序中安装出口列表处理程序且不要执行 COMMIT。通常，DB2 在应用程序中安装进程出口列表处理程序，如果应用程序正常结束，则出口列表处理程序执行 COMMIT 操作。 对于动态装入 DB2 库并在应用程序终止前卸载它的应用程序，出口列表处理程序的调用将失败，因为该处理程序例程已不再装入到应用程序中。如果您的应用程序以这种方式操作，则应设置 *DB2NOEXITLIST* 变量，并确保应用程序显式地调用所有必需的 COMMIT。		
DB2REMOTEPREG	Windows 95 和 Windows NT	缺省值=null 值: 任何有效的 Windows 95 和 Windows NT 机器名
指定包含 DB2 实例简要表和 DB2 实例的 Win32 注册列表的远程机器名。DB2REMOTEPREG 的值只应在安装 DB2 之后设置一次，且不应该修改。使用此变量要格外小心。		
DB2ROUTINE_DEBUG	AIX 和 Windows NT	缺省值=OFF 值: ON, OFF
指定是否对 Java 存储过程启用调试能力。如果不调试 Java 存储过程，则使用缺省值 OFF。启用调试对性能有影响。有关调试 Java 存储过程的详情，参考 *Application Development Guide*。		

表 30. 其他变量 (续)

变量名 说明	操作系统	值
DB2SORCVBUF 指定在 Windows 95 和 Windows NT 操作系统上的 TCP/IP 接收缓冲区的值。	Windows 95 和 Windows NT	缺省值=32767
DB2SORT 指定 LOAD 实用程序在运行期要装入的库的位置。该库包含在对索引数据排序时使用的函数的入口点。使用 *db2sort* 来利用供应商提供的排序产品，以便在生成表索引时与 LOAD 实用程序一起使用。提供的路径必须是相对于数据库服务器的路径。	所有，仅适用于服务器	缺省值=null
DB2SYSTEM 指定您的用户和数据库管理员用于标识 DB2 服务器系统的名称。如果可能的话，此名称应在您的网络内是唯一的。 此名称显示在控制中心的对象树的系统层，以辅助管理员标识可以从控制中心管理的服务器系统。 当使用“客户机配置辅助”的“搜索网络”功能时，DB2 discovery 返回此名称，并在结果对象树中的系统层显示它。此名称辅助用户标识包含他们希望存取的数据库的系统。在安装时，将 *db2system* 的值按下列情况设置: • 在 Windows NT 或 Windows 95 上，安装程序将它设置成为 Windows 系统指定的计算机名。 • 在 OS/2 上，在安装过程期间提示用户输入 DB2SYSTEM 名。 • 在 UNIX 系统上，将它设置成 UNIX 系统的 TCP/IP 主机名。	Windows NT, Windows 95, OS/2 和 UNIX	缺省值=null
DB2UPMPR 指定当用户在 OS/2 上输入错误的用户 ID 或口令时， UPM 注册屏幕是否会在屏幕上显示。	OS/2	缺省值=ON 值: ON 或 OFF

附录B. 解释表和定义

当激活解释设施时，解释表将捕捉存取方案。本节描述下列解释表和定义:

在可以调用解释设施之前必须创建解释表。要创建它们，使用在 EXPLAIN.DDL 文件中提供的样本命令行处理器输入脚本， EXPLAIN.DLL 文件位于 'sqllib' 目录的 'misc' 子目录中。与需要解释表的数据库连接。然后发出命令: `db2 -tf EXPLAIN.DDL`，这样就创建了表。有关详情，参见第462页的『解释表的表定义』。

解释设施对解释表的填充既不会激活任何触发器，也不会激活任何参考或检查约束。例如，如果在 EXPLAIN_INSTANCE 表上定义了插入触发器，并解释了适用的语句，则不会激活触发器。

查看第177页的『第7章 SQL 解释设施』以获取关于解释设施的更多细节。

解释表的图例:

标题	说明
列名	列的名称
数据类型	列的数据类型
可空?	是: 允许空值
	否: 不允许空值
关键字?	PK: 列是主关键字的一部分
	FK: 列是外部关键字的一部分
说明	列的说明

EXPLAIN_ARGUMENT 表

EXPLAIN_ARGUMENT 表表示每个单独运算符的唯一特征（如果有的话）。

表 31. EXPLAIN_ARGUMENT 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	解释动态语句时运行的程序包名或解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的解释级的信息。
STMTNO	INTEGER	否	FK	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此解释信息相关的段编号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一 ID。
ARGUMENT_TYPE	CHAR(8)	否	否	此运算符的自变量类型。
ARGUMENT_VALUE	VARCHAR(1024)	是	否	此运算符的自变量值。如果此值用 LONG_ARGUMENT_VALUE 表示，则为空值。
LONG_ARGUMENT_VALUE	CLOB(1M)	是	否	当文本超出 ARGUMENT_VALUE 的长度时此运算符的自变量值。如果此值用 ARGUMENT_VALUE 表示，则为空值。

表 32. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	说明
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	部份聚合指示符。
BITFLTR	TRUE FALSE	“散列连接”将使用位过滤器来增强性能。
CSETEMP	TRUE FALSE	常用子表达式标志上的临时表
DIRECT	TRUE	直接读取指示符。
DUPLWARN	TRUE FALSE	重复的警告标志。
EARLYOUT	TRUE FALSE	早出指示符。
ENVVAR	此类型的每行将包含: • 环境变量名 • 环境变量值	影响优化器的环境变量

表 32. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	说明
FETCHMAX	IGNORE INTEGER	FETCH 运算符上的 MAXPAGES 自变量的替换值。
GROUPBYC	TRUE FALSE	是否提供 Group By 列。
GROUPBYN	整数	比较列的数目。
GROUPBYR	此类型的每行将包含: • Group By 子句中的列的顺序值（后跟冒号和空格） • 列名	Group By 需求。
INNERCOL	此类型的每行将包含: • 有序列的顺序值（后跟冒号和空格） • 列名 • 次序值 **(A)** 升序 **(D)** 降序	内部顺序列。
ISCANMAX	IGNORE INTEGER	ISCAN 运算符上的 MAXPAGES 自变量的替换值。
JN_INPUT	INNER OUTER	指示运算符是否是提供内部或外部连接的指示符。
LISTENER	TRUE FALSE	“监听程序表队列”指示符。
MAXPAGES	ALL 无 INTEGER	期望“预读取”的最大页数。
MAXRIDS	无 INTEGER	要包括在每个列表预读取请求中的“最大行标识符”。
NUMROWS	INTEGER	期望排序的行数。
ONEFETCH	TRUE FALSE	一个“读取”指示符。
OUTERCOL	此类型的每行将包含: • 有序列的顺序值（后跟冒号和空格） • 列名 • 次序值 **(A)** 升序 **(D)** 降序	外部顺序列。

表 32. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	说明
OUTERJN	LEFT RIGHT	外部连接指示符。
PARTCOLS	列名	运算符的分区列。
PREFETCH	LIST 无 SEQUENTIAL	合格的预读取类型。
RMTQTEXT	查询文本	远程查询文本
ROWLOCK	EXCLUSIVE 无 REUSE SHARE SHORT (INSTANT) SHARE UPDATE	行锁定意图。
ROWWIDTH	INTEGER	要排序的行宽。***
SCANDIR	FORWARD REVERSE	扫描方向。
SCANGRAN	INTEGER	分区内并行性，分区内并行扫描的粒度，以 SCANUNIT 表示。
SCANTYPE	LOCAL PARALLEL	分区内并行性，索引扫描或表扫描。
SCANUNIT	ROW PAGE	分区内并行性，扫描粒度单位。
SERVER	远程服务器	远程服务器
SHARED	TRUE	分区内并行性，共享的 TEMP 指示符。
SLOWMAT	TRUE FALSE	“慢速具体化”标志。
SNGLPROD	TRUE FALSE	单个代理程序产生的分区内并行性排序或 temp。
SORTKEY	此类型的每行将包含: • 关键字中列的顺序值（后跟冒号和空格） • 列名 • 次序值 **(A)** 升序 **(D)** 降序	排序关键字列。
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	分区内并行性，排序类型。

表 32. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	说明
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	表锁定意图。
TQDEGREE	INTEGER	分区内并行性，存取“表队列”的子代理程序数。
TQMERGE	TRUE FALSE	合并（排序的）“表队列”指示符。
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	“表队列”读取特性。
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	“表队列”发送特性。
TQTYPE	LOCAL	分区内并行性，表队列。
TRUNCSRT	TRUE	被截断的排序（限制产生的行数）。
UNIQUE	TRUE FALSE	唯一性指示符。
UNIQKEY	此类型的每行将包含: • 关键字中列的顺序值（后跟冒号和空格） • 列名	唯一关键字列。
VOLATILE	TRUE	易变的表

EXPLAIN_INSTANCE 表

EXPLAIN_INSTANCE 表是所有解释信息的主控制表。解释表中的每一行数据都显式地链接至此表中唯一的一行。EXPLAIN_INSTANCE 表提供关于解释的 SQL 语句的源基本信息，以及进行解释所在的环境的信息。

有关此表的定义，参见 第465页的『EXPLAIN_INSTANCE 表定义』。

表 33. EXPLAIN_INSTANCE 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	PK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	PK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	PK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	PK	解释请求的源的模式或限定符。
EXPLAIN_OPTION	CHAR(1)	否	否	指示为此请求所请求的解释信息。 可能的值是: **P** PLAN SELECTION
SNAPSHOT_TAKEN	CHAR(1)	否	否	指示是否为此请求抽取了解释快照。 可能的值是: **Y** 是，抽取了解释快照并存储在 EXPLAIN_STATEMENT 表中。同时捕捉了常规的解释信息。 **N** 未抽取解释快照。捕捉常规的解释信息。 **O** 只抽取了一个解释快照。未捕捉常规的解释信息。
DB2_VERSION	CHAR(7)	否	否	处理此解释请求的 DB2 通用数据库的产品发行版号。格式是 vv.rr.m，其中: **vv** 版本号 **rr** 发行版号 **m** 维护发行版号
SQL_TYPE	CHAR(1)	否	否	指示解释实例是静态 SQL 还是动态 SQL 的实例。 可能的值是: **S** 静态 SQL **D** 动态 SQL
QUERYOPT	INTEGER	否	否	指示 SQL 编译程序在调用解释时使用的查询优化级别。该值指示 SQL 编译程序对正解释的 SQL 语句执行的查询优化级别。

表 33. EXPLAIN_INSTANCE 表 (续)

列名	数据类型	可空?	关键字?	说明
BLOCK	CHAR(1)	否	否	指示编译 SQL 语句时所使用的游标分块类型。有关详情，可查看 SYSCAT.PACKAGES 中的 BLOCK 列。 可能的值是: **N** 无分块 **U** 分块相异值游标 **B** 分块所有游标
ISOLATION	CHAR(2)	否	否	指示编译 SQL 语句时所使用的隔离类型。有关详情，可查看 SYSCAT.PACKAGES 中的 ISOLATION 列。 可能的值是: **RR** 可重复读 **RS** 读稳定性 **CS** 游标稳定性 **UR** 未落实的读
BUFFPAGE	INTEGER	否	否	包含调用解释设施时 BUFFPAGE 数据库配置设置的值。
AVG_APPLS	INTEGER	否	否	包含调用解释设施时 AVG_APPLS 配置参数的值。
SORTHEAP	INTEGER	否	否	包含调用解释设施时 SORTHEAP 数据库配置设置的值。
LOCKLIST	INTEGER	否	否	包含调用解释设施时 LOCKLIST 数据库配置设置的值。
MAXLOCKS	SMALLINT	否	否	包含调用解释设施时 MAXLOCKS 数据库配置设置的值。
LOCKS_AVAIL	INTEGER	否	否	包含由每个用户的优化器假定为可用的锁定数目。（从 LOCKLIST 和 MAXLOCKS 中派生。）
CPU_SPEED	DOUBLE	否	否	包含调用解释设施时 CPUSPEED 数据库管理程序配置设置的值。
REMARKS	VARCHAR(254)	是	否	用户提供的注解。
DBHEAP	INTEGER	否	否	包含调用解释设施时 DBHEAP 数据库配置设置的值。
COMM_SPEED	DOUBLE	否	否	包含调用解释设施时 COMM_BANDWIDTH 数据库配置设置的值。

表 33. EXPLAIN_INSTANCE 表 (续)

列名	数据类型	可空?	关键字?	说明
PARALLELISM	CHAR(2)	否	否	可能的值是: • N = 没有并行性 • P = 分区内并行性 • IP = 分区间并行性 • BP = 分区内并行性和分区间并行性
DATAJOINER	CHAR(1)	否	否	可能的值是: • N = 非联合体系统方案 • Y = 联合体系统方案

EXPLAIN_OBJECT 表

EXPLAIN_OBJECT 表标识那些生成用来满足 SQL 语句的存取方案所要求的数据对象。

表 34. EXPLAIN_OBJECT 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的解释级的信息。
STMTNO	INTEGER	否	FK	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此解释信息相关的段编号。
OBJECT_SCHEMA	VARCHAR(128)	否	否	此对象所属的模式。
OBJECT_NAME	VARCHAR(128)	否	否	对象名称。
OBJECT_TYPE	CHAR(2)	否	否	对象类型的描述性标签。
CREATE_TIME	TIMESTAMP	是	否	“对象时间”的创建；如果是表函数则为空。
STATISTICS_TIME	TIMESTAMP	是	否	对此对象的统计信息的上次更新时间；如果此对象的统计信息不存在，则为空。
COLUMN_COUNT	SMALLINT	否	否	此对象中的列数。
ROW_COUNT	INTEGER	否	否	此对象中行的估计数目。
WIDTH	INTEGER	否	否	对象的平均宽度，以字节计。对索引，设置为 -1。
PAGES	INTEGER	否	否	对象占用的缓冲池的估计页数。对表函数设置为 -1。

表 34. EXPLAIN_OBJECT 表 (续)

列名	数据类型	可空?	关键字?	说明
DISTINCT	CHAR(1)	否	否	指示对象中的行是否不同（即，没有重复） 可能的值是: **Y** 是 **N** 否
TABLESPACE_NAME	VARCHAR(128)	是	否	在其中存储此对象的表空间的名称；如果未涉及任何表空间，则设置为空。
OVERHEAD	DOUBLE	否	否	以毫秒为单位，对指定的表空间的单个随机 I/O 操作估计的总额外开销。包括控制器额外开销、磁盘查找和等待时间。如果不涉及表空间，则设置为 -1。
TRANSFER_RATE	DOUBLE	否	否	以毫秒计的从指定的表空间读取一页数据的估计时间。如果不涉及表空间，则设置为 -1。
PREFETCHSIZE	INTEGER	否	否	当执行预读取时要读取的数据页数。对表函数设置为 -1。
EXTENTSIZE	INTEGER	否	否	范围的大小，以数据页计。在转换至下一个容器之前，这些页被写入表空间中的一个容器。对表函数设置为 -1。
CLUSTER	DOUBLE	否	否	带索引的数据分组级别。如果>= 1，则这是 CLUSTERRATIO。如果>= 0且< 1，则这是 CLUSTERFACTOR。对表、表函数或者如果此统计信息不可用，设置为 -1。
NLEAF	INTEGER	否	否	此索引对象的值占用的叶页数。对表、表函数或者如果此统计信息不可用，设置为 -1。
NLEVELS	INTEGER	否	否	此索引对象的树中的索引层数。对表、表函数或者如果此统计信息不可用，设置为 -1。
FULLKEYCARD	BIGINT	否	否	此索引对象中包含的相异完整关键字值的数目。对表、表函数或者如果此统计信息不可用，设置为 -1。
OVERFLOW	INTEGER	否	否	表中溢出记录的总数。对索引、表函数或者如果此统计信息不可用，设置为 -1。
FIRSTKEYCARD	BIGINT	否	否	相异的第一个关键字值的数目。对表、表函数或者如果此统计信息不可用，设置为 −1。
FIRST2KEYCARD	BIGINT	否	否	使用索引前 {2,3,4} 列的前面几个相异值关键字数。对表、表函数或者如果此统计信息不可用，设置为 −1。
FIRST3KEYCARD	BIGINT	否	否	
FIRST4KEYCARD	BIGINT	否	否	
SEQUENTIAL_PAGES	INTEGER	否	否	磁盘上的叶页数；叶页之间的间隔很小，按索引关键字次序排列。对表、表函数或者如果此统计信息不可用，设置为 −1。

表 34. EXPLAIN_OBJECT 表 (续)

列名	数据类型	可空?	关键字?	说明
DENSITY	INTEGER	否	否	SEQUENTIAL_PAGES 与该索引占用的页范围中页数的比率，表示为一个百分比（在 0 和 100 之间的整数）。对表、表函数或者如果此统计信息不可用，设置为 −1。

表 35. 可能的 OBJECT_TYPE 值

值	说明
IX	索引
TA	表
TF	表函数

EXPLAIN_OPERATOR 表

EXPLAIN_OPERATOR 表包含 SQL 编译程序为满足 SQL 语句所需的所有运算符。

表 36. EXPLAIN_OPERATOR 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的解释级的信息。
STMTNO	INTEGER	否	FK	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此解释信息相关的段编号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一 ID。
OPERATOR_TYPE	CHAR(6)	否	否	运算符类型的描述性标签。
TOTAL_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），执行选择的存取方案的估计累积总成本（以 timerons 为单位）。
IO_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），执行选择的存取方案的估计累积 I/O 成本（以 I/O 的数据页数计）。
CPU_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），执行选择的存取方案的估计累积 CPU 成本（以指令数计）。
FIRST_ROW_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），读取存取方案的第一行的估计累积成本（以 timerons 为单位）。此值包括任何必需的初始额外开销。

表 36. EXPLAIN_OPERATOR 表 (续)

列名	数据类型	可空?	关键字?	说明
RE_TOTAL_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），读取所选择的存取方案的下一行的估计累积成本（以 timerons为单位）。
RE_IO_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），读取选择的存取方案的下一行的估计累积 I/O 成本（以 I/O 的数据页数计）。
RE_CPU_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），读取所选择的存取方案的下一行的估计累积 CPU 成本（以 timerons 为单位）。
COMM_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），执行选择的存取方案的估计累积通信成本（以 TCP/IP 帧数计）。
FIRST_COMM_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），读取所选择的存取方案的第一行的估计累积通信成本（以 TCP/IP 帧数计）。此值包括任何必需的初始额外开销。
BUFFERS	DOUBLE	否	否	此运算符及其输入的估计缓冲区需求。
REMOTE_TOTAL_COST	DOUBLE	否	否	对远程数据库执行操作的估计累积总成本（以 timerons 为单位）。
REMOTE_COMM_COST	DOUBLE	否	否	到此运算符为止（包括此运算符），执行选择的远程存取方案的估计累积通信成本。

表 37. OPERATOR_TYPE 值

值	说明
DELETE	删除
FETCH	读取
FILTER	过滤器行
GENROW	生成行
GRPBY	分组方式
HSJOIN	散列连接
INSERT	插入
IXAND	动态位图索引 AND 运算
IXSCAN	索引扫描
MSJOIN	合并扫描连接
NLJOIN	嵌套循环连接
RETURN	结果
RIDSCN	行标识符 (RID) 扫描
RQUERY	远程查询
SORT	排序
TBSCAN	表扫描

表 37. OPERATOR_TYPE 值 (续)

值	说明
TEMP	临时表构造
TQ	表队列
UNION	联合
UNIQUE	消去重复
UPDATE	更新

EXPLAIN_PREDICATE 表

EXPLAIN_PREDICATE 表标识特定的运算符应用哪些谓词。

表 38. EXPLAIN_PREDICATE 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的解释级的信息。
STMTNO	INTEGER	否	FK	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此解释信息相关的段编号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一 ID。
PREDICATE_ID	INTEGER	否	否	指定运算符的此谓词的唯一 ID。
HOW_APPLIED	CHAR(5)	否	否	指定的运算符正在如何使用谓词。
WHEN_EVALUATED	CHAR(3)	否	否	指示何时对用于此谓词的子查询求值。 可能的值是: **空白** 此谓词不包含子查询。 **EAA** 在应用程序 (EAA) 中对用于此谓词的子查询求值。即，当正在应用该谓词时，就对指定的运算符所处理的每一行重新求值。 **EAO** 在开放式 (EAO) 上对此谓词中使用的子查询求值。即，对指定的运算符只将它重新求值一次，并且对每一行应用该谓词时重新使用其结果。 **MUL** 在此谓词中有多种的子查询类型。
RELOP_TYPE	CHAR(2)	否	否	在此谓词中使用的关系运算符的类型。

表 38. EXPLAIN_PREDICATE 表 (续)

列名	数据类型	可空?	关键字?	说明
SUBQUERY	CHAR(1)	否	否	此谓词是否需要来自子查询的数据流。可能需要多个子查询流。 可能的值是: **N** 不需要子查询流 **Y** 需要一个或多个子查询流
FILTER_FACTOR	DOUBLE	否	否	将由此谓词限定的行的估计比例。
PREDICATE_TEXT	CLOB(1M)	是	否	如同根据 SQL 语句的内部表示再创建的谓词文本。 如果不可用，则为空。

表 39. 可能的 HOW_APPLIED 值

值	说明
JOIN	用于连接表
RESID	作为剩余的谓词求值
SARG	作为索引或数据页的 sargable 谓词求值
START	用作起始条件
STOP	用作停止条件

表 40. 可能的 RELOP_TYPE 值

值	说明
空白	不适用
EQ	等于
GE	大于或等于
GT	大于
IN	在列表中
LE	小于或等于
LK	相似
LT	小于
NE	不等于
NL	为空
NN	不为空

EXPLAIN_STATEMENT 表

当存在不同级别的解释信息的 EXPLAIN_STATEMENT 表时，它包含 SQL 语句的文本。用户输入的原始 SQL 语句与（由优化器）使用的版本一起存储在此表中以选择存取方案以满足该 SQL 语句。此后的版本可能很少有与初始版本相似的地方，因为它可能已被重写，并且／或者已用由 SQL 编译程序确定的附加的谓词增强。

有关此表的定义，可查看第469页的『EXPLAIN_STATEMENT 表定义』。

表 41. EXPLAIN_STATEMENT 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	PK, FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	PK, FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	PK, FK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	PK, FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	PK	与此行相关的解释级的信息。 有效的值为: **O** 初始文本（用户输入的） **P** PLAN SELECTION
STMTNO	INTEGER	否	PK	程序包中与此解释信息相关的语句号。对于动态解释 SQL 语句，设置为 1。对于静态 SQL 语句，此值与用于 SYSCAT.STATEMENTS 目录视图的值相同。
SECTNO	INTEGER	否	PK	程序包中包含此 SQL 语句的段编号。对于动态解释 SQL 语句，这是在运行期保存此语句的段的段号。对于静态 SQL 语句，此值与用于 SYSCAT.STATEMENTS 目录视图的值相同。
QUERYNO	INTEGER	否	否	解释的 SQL 语句的数值标识符。对于通过 CLP 或 CLI 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值是顺序增加的值。否则，对于静态 SQL 语句，缺省值为 STMTNO 的值，而对于动态 SQL 语句，则为 1。
QUERYTAG	CHAR(20)	否	否	每个解释的 SQL 语句的标识符标记。对于通过 CLP 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值为 'CLP'。对于通过 CLI 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值为 'CLI'。否则，所用的缺省值为空白。

表 41. EXPLAIN_STATEMENT 表 (续)

列名	数据类型	可空?	关键字?	说明
STATEMENT_TYPE	CHAR(2)	否	否	解释的查询类型的描述性标记。 可能的值是: **S** 选择 **D** 删除 **DC** 自游标当前所在位置删除 **I** 插入 **U** 更新 **UC** 自游标当前所在位置更新
UPDATABLE	CHAR(1)	否	否	指示此语句是否被视作可更新的。这与 SELECT 语句特别有关，这些语句可以被确定为潜在可更新的。 可能的值是: **' '** 不适用的（空白） **N** 否 **Y** 是
DELETABLE	CHAR(1)	否	否	指示此语句是否被视作可删除的。这与 SELECT 语句特别有关，这些语句可能被确定为潜在可删除的。 可能的值是: **' '** 不适用的（空白） **N** 否 **Y** 是
TOTAL_COST	DOUBLE	否	否	执行为此语句所选的存取方案的估计成本总量（以 timerons 为单位）；如果 EXPLAIN_LEVEL 是 *O*（原始文本）则设置为 0（零），因为此时未选择存取方案。
STATEMENT_TEXT	CLOB(1M)	否	否	解释的 SQL 语句的文本或部分文本。为解释的“方案选择”级别显示的文本已根据内部表示法重新构造，并且本质上与 SQL 相似；即，重新构造的语句并不保证符合正确的 SQL 语法。
SNAPSHOT	BLOB(10M)	是	否	在显示的 Explain_Level 此 SQL 语句的内部表示法快照。 此列将用于DB2 Visual Explain。如果 EXPLAIN_LEVEL 是 *0*（原始语句），则列被设置为空值，因为在捕捉语句的此特定版本时未选择存取方案。

表 41. EXPLAIN_STATEMENT 表 (续)

列名	数据类型	可空?	关键字?	说明
QUERY_DEGREE	INTEGER	否	否	指示调用解释设施时的内部分区并行度。对于原始语句，这包含定向的内部分区并行度。对于 PLAN SELECTION，这包含为要使用的方案生成的内部分区并行度。

EXPLAIN_STREAM 表

EXPLAIN_STREAM 表表示单个运算符与数据对象之间的输入和输出数据流。在 EXPLAIN_OBJECT 表中表示数据对象本身。涉及数据流的运算符可在 EXPLAIN_OPERATOR 表中找到。

表 42. EXPLAIN_STREAM 表

列名称	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	FK	解释请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当解释动态语句时运行的程序包名或当解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的解释级的信息。
STMTNO	INTEGER	否	FK	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此解释信息相关的段编号。
STREAM_ID	INTEGER	否	否	指定的运算符中此数据流的唯一 ID。
SOURCE_TYPE	CHAR(1)	否	否	指示此数据流的源: **O** 运算符 **D** 数据对象
SOURCE_ID	SMALLINT	否	否	此查询内的运算符的唯一 ID，此查询是此数据流的源。如果 SOURCE_TYPE 是 'D'，则设置为 -1。
TARGET_TYPE	CHAR(1)	否	否	指示此数据流的目标: **O** 运算符 **D** 数据对象
TARGET_ID	SMALLINT	否	否	此查询内的运算符的唯一 ID，此查询是此数据流的目标。如果 TARGET_TYPE 是 'D'，则设置为 -1。
OBJECT_SCHEMA	VARCHAR(128)	是	否	受影响的数据对象所属的模式。如果 SOURCE_TYPE 与 TARGET_TYPE 都是 'O'，则设置为空。

表 42. EXPLAIN_STREAM 表 (续)

列名称	数据类型	可空?	关键字?	说明
OBJECT_NAME	VARCHAR(128)	是	否	作为数据流主题的对象名。如果 SOURCE_TYPE 与 TARGET_TYPE 都是 'O'，则设置为空。
STREAM_COUNT	DOUBLE	否	否	估计的数据流基数。
COLUMN_COUNT	SMALLINT	否	否	数据流中的列数。
PREDICATE_ID	INTEGER	否	否	如果此数据流是谓词的子查询的一部分，则该谓词 ID 将在此反映出来，否则将该列设置为 -1。
COLUMN_NAMES	CLOB(1M)	是	否	此列包含涉及此数据流的列的名称和排序信息。 这些名称将以下列格式出现: `NAME1(A)+NAME2(D)+NAME3+NAME4` 其中 *(A)* 指示以升序排列的列， *(D)* 指示以降序排列的列，没有排序信息指示列未排序，或者排序是不相关的。
PMID	SMALLINT	否	否	分区映象 ID。
SINGLE_NODE	CHAR(5)	是	否	指示此数据流是在单个分区还是在多个分区上: **MULT** 在多个分区上 **COOR** 在协调程序节点上 **HASH** 使用散列定向 **RID** 使用行 ID 定向 **FUNC** 使用函数定向（PARTITION() 或 NODENUMBER()） **CORR** 使用相关值定向 **Numberic** 定向至预先确定的单一节点
PARTITION_COLUMNS	CLOB(64K)	是	否	将此数据流分区时所依据列的列表。

ADVISE_INDEX 表

ADVISE_INDEX 表提供了建议的索引。

表 43. ADVISE_INDEX 表

列名	数据类型	可空?	关键字?	说明
EXPLAIN_REQUESTER	VARCHAR(128)	否	否	发出此“解释”请求的授权 ID。
EXPLAIN_TIME	TIMESTAMP	否	否	解释请求的启动时间。

表 43. ADVISE_INDEX 表 (续)

列名	数据类型	可空?	关键字?	说明
SOURCE_NAME	VARCHAR(128)	否	否	解释动态语句时运行的程序包名或解释静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	否	解释请求的源的模式或限定符。
EXPLAIN_LEVEL	CHAR(1)	否	否	与此行相关的解释级的信息。
STMTNO	INTEGER	否	否	程序包中与此解释信息相关的语句号。
SECTNO	INTEGER	否	否	程序包中与此解释信息相关的段编号。
QUERYNO	INTEGER	否	否	解释的 SQL 语句的数值标识符。对于通过 CLP 或 CLI 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值是顺序增加的值。否则，对于静态 SQL 语句，缺省值为 STMTNO 的值，而对于动态 SQL 语句，则为 1。
QUERYTAG	CHAR(20)	否	否	每个解释的 SQL 语句的标识符标记。对于通过 CLP 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值为 'CLP'。对于通过 CLI 发出的动态 SQL 语句（除了 EXPLAIN SQL 语句），缺省值为 'CLI'。否则，所用的缺省值为空白。
NAME	VARCHAR(128)	否	否	索引名。
CREATOR	VARCHAR(128)	否	否	索引名的限定符。
TBNAME	VARCHAR(128)	否	否	定义索引所依据的表的名称或别名。
TBCREATOR	VARCHAR(128)	否	否	表名的限定符。
COLNAMES	CLOB(64K)	否	否	列名列表。
UNIQUERULE	CHAR(1)	否	否	唯一的规则: D = 允许重复值 P = 主索引 U = 只允许唯一的项
COLCOUNT	SMALLINT	否	否	关键字中的列数以及包括列（如果有）的列数。
IID	SMALLINT	否	否	内部索引 ID。
NLEAF	INTEGER	否	否	叶页数；如果未收集统计信息，则为 −1。
NLEVELS	SMALLINT	否	否	索引层号；如果未收集统计信息，则为 −1。
FULLKEYCARD	BIGINT	否	否	全相异值关键字数；如果未收集统计信息，则为 −1。
FIRSTKEYCARD	BIGINT	否	否	第一个相异值关键字数；如果未收集统计信息，则为 −1。
CLUSTERRATIO	SMALLINT	否	否	该索引的数据分组级别；如果未收集统计信息，或者如果收集了详细的索引统计信息（在这种情况下，将改用 CLUSTERFACTOR），则为 −1。
CLUSTERFACTOR	DOUBLE	否	否	分组级别更准确的测量。如果没有收集详细的索引统计信息或者根据别名定义了索引，则为 −1。
USERDEFINED	SMALLINT	否	否	由用户定义。

表 43. ADVISE_INDEX 表 (续)

列名	数据类型	可空?	关键字?	说明
SYSTEM_REQUIRED	SMALLINT	否	否	如果此索引是主关键字或唯一关键字约束所必需的，或者，如果它是具有类型的表的对象标识符 (OID) 列上的索引，则为 1。 如果此索引是主关键字或唯一关键字约束所必需的，而且，它是具有类型的表的对象标识符 (OID) 列上的索引，则为 2。 否则，为 0。
CREATE_TIME	TIMESTAMP	否	否	创建索引的时间。
STATS_TIME	TIMESTAMP	是	否	上次更改为此索引记录的统计信息的时间。如果没有统计信息可用，则为空。
PAGE_FETCH_PAIRS	VARCHAR(254)	否	否	成对整数列表，以字符形式表示。每一对表示一个假设缓冲区中的页数，以及使用该假设缓冲区时使用此索引来扫描该表所需的页读取数。(如果没有可用的数据，则字符串长度为零。)
REMARKS	VARCHAR(254)	是	否	用户提供的注解，或为空。
DEFINER	VARCHAR(128)	否	否	创建索引的用户。
CONVERTED	CHAR(1)	否	否	保留以备将来使用。
SEQUENTIAL_PAGES	INTEGER	否	否	磁盘上的叶页数；叶页之间的间隔很小，按索引关键字次序排列。(如果没有统计信息可用，则为 −1)。
DENSITY	INTEGER	否	否	SEQUENTIAL_PAGES 和索引所占页数的比值，用百分比表示 （0 和 100 之间的整数），如果统计信息不可用，则为 −1。
FIRST2KEYCARD	BIGINT	否	否	使用索引前两列的相异值关键字数 (如果没有统计信息或不适用，则为 −1)
FIRST3KEYCARD	BIGINT	否	否	使用索引前三列的相异值关键字数 (如果没有统计信息或不适用，则为 −1)
FIRST4KEYCARD	BIGINT	否	否	使用索引前四列的相异值关键字数 (如果没有统计信息或不适用，则为 −1)
PCTFREE	SMALLINT	否	否	在索引的初始构建期间，每个索引叶页上要保留的百分比。当构建索引后，此空间可用于以后的插入操作。
UNIQUE_COLCOUNT	SMALLINT	否	否	唯一关键字必需的列数。总是 <=COLCOUNT。仅当有包括列时，才 < COLCOUNT。如果索引没有唯一关键字 (允许重复值)，则为 −1
MINPCTUSED	SMALLINT	否	否	如果不为零，则启用联机索引重组，且该值为在合并页之前已用最小空间的阈值。
REVERSE_SCANS	CHAR(1)	否	否	Y = 索引支持反向扫描 N = 索引不支持反向扫描

表 43. ADVISE_INDEX 表 (续)

列名	数据类型	可空?	关键字?	说明
USE_INDEX	CHAR(1)	是	否	Y = 建议或评估的索引 N = 不建议的索引
CREATION_TEXT	CLOB(1M)	否	否	用来创建索引的 SQL 语句。
PACKED_DESC	BLOB(20M)	是	否	表的内部说明。

ADVISE_WORKLOAD 表

ADVISE_WORKLOAD 表提供构成工作负荷的语句。有关工作负荷的详情，参考管*理指南: 性能*。

表 44. ADVISE_WORKLOAD 表

列名	数据类型	可空?	关键字?	说明
WORKLOAD_NAME	CHAR(128)	否	否	此语句所属的 SQL 语句集（工作负荷）的名称。
STATEMENT_NO	INTEGER	否	否	工作负荷中与此解释信息相关的语句号。
STATEMENT_TEXT	CLOB(1M)	否	否	SQL 语句的内容。
STATEMENT_TAG	VARCHAR(256)	否	否	每个解释的 SQL 语句的标识符标记。
FREQUENCY	INTEGER	否	否	此语句在工作负荷中出现的次数。
IMPORTANCE	DOUBLE	否	否	此语句的重要性。
COST_BEFORE	DOUBLE	是	否	在不创建建议的索引的情况下，此查询的成本（以 timerons 计）。
COST_AFTER	DOUBLE	是	否	在创建建议的索引的情况下，此查询的成本（以 timerons 计）。

解释表的表定义

在可以调用解释设施之前必须创建解释表。以下定义指定如何创建需要的解释表:

- 第464页的『EXPLAIN_ARGUMENT 表定义』
- 第465页的『EXPLAIN_INSTANCE 表定义』
- 第466页的『EXPLAIN_OBJECT 表定义』
- 第467页的『EXPLAIN_OPERATOR 表定义』
- 第468页的『EXPLAIN_PREDICATE 表定义』
- 第469页的『EXPLAIN_STATEMENT 表定义』
- 第470页的『EXPLAIN_STREAM 表定义』
- 第471页的『ADVISE_INDEX 表定义』
- 第473页的『ADVISE_WORKLOAD 表定义』

或者，可使用在 EXPLAIN.DDL 文件中提供的样本命令行处理器输入脚本来创建它们，该 EXPLAIN.DLL 文件位于 'sqllib' 目录的 'misc' 子目录中。与需要解释表的数据库连接。然后发出命令：`db2 -tf EXPLAIN.DDL`，这样就创建了表。

EXPLAIN_ARGUMENT 表定义

```
CREATE TABLE EXPLAIN_ARGUMENT ( EXPLAIN_REQUESTER   VARCHAR(128)  NOT NULL,
                                EXPLAIN_TIME        TIMESTAMP     NOT NULL,
                                SOURCE_NAME         VARCHAR(128)  NOT NULL,
                                SOURCE_SCHEMA       VARCHAR(128)  NOT NULL,
                                EXPLAIN_LEVEL       CHAR(1)       NOT NULL,
                                STMTNO              INTEGER       NOT NULL,
                                SECTNO              INTEGER       NOT NULL,
                                OPERATOR_ID         INTEGER       NOT NULL,
                                ARGUMENT_TYPE       CHAR(8)       NOT NULL,
                                ARGUMENT_VALUE      VARCHAR(1024) NOT NULL,
                                LONG_ARGUMENT_VALUE CLOB(1M)      NOT LOGGED,
                                   FOREIGN KEY (EXPLAIN_REQUESTER,
                                                EXPLAIN_TIME,
                                                SOURCE_NAME,
                                                SOURCE_SCHEMA,
                                                EXPLAIN_LEVEL,
                                                STMTNO,
                                                SECTNO)
                                   REFERENCES EXPLAIN_STATEMENT
                                   ON DELETE CASCADE )
```

EXPLAIN_INSTANCE 表定义

```
CREATE TABLE EXPLAIN_INSTANCE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_OPTION    CHAR(1)   NOT NULL,
                                SNAPSHOT_TAKEN    CHAR(1)   NOT NULL,
                                DB2_VERSION       CHAR(7)   NOT NULL,
                                SQL_TYPE          CHAR(1)   NOT NULL,
                                QUERYOPT          INTEGER   NOT NULL,
                                BLOCK             CHAR(1)   NOT NULL,
                                ISOLATION         CHAR(2)   NOT NULL,
                                BUFFPAGE          INTEGER   NOT NULL,
                                AVG_APPLS         INTEGER   NOT NULL,
                                SORTHEAP          INTEGER   NOT NULL,
                                LOCKLIST          INTEGER   NOT NULL,
                                MAXLOCKS          SMALLINT  NOT NULL,
                                LOCKS_AVAIL       INTEGER   NOT NULL,
                                CPU_SPEED         DOUBLE    NOT NULL,
                                REMARKS           VARCHAR(254),
                                DBHEAP            INTEGER   NOT NULL,
                                COMM_SPEED        DOUBLE    NOT NULL,
                                PARALLELISM       CHAR(2)   NOT NULL,
                                DATAJOINER        CHAR(1)      NOT NULL,
                                    PRIMARY KEY (EXPLAIN_REQUESTER,
                                                 EXPLAIN_TIME,
                                                 SOURCE_NAME,
                                                 SOURCE_SCHEMA))
```

EXPLAIN_OBJECT 表定义

```
CREATE TABLE EXPLAIN_OBJECT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128) NOT NULL,
                              OBJECT_NAME       VARCHAR(128) NOT NULL,
                              OBJECT_TYPE       CHAR(2)      NOT NULL,
                              CREATE_TIME       TIMESTAMP,
                              STATISTICS_TIME   TIMESTAMP,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              ROW_COUNT         INTEGER      NOT NULL,
                              WIDTH             INTEGER      NOT NULL,
                              PAGES             INTEGER      NOT NULL,
                              DISTINCT          CHAR(1)      NOT NULL,
                              TABLESPACE_NAME   VARCHAR(128),
                              OVERHEAD          DOUBLE       NOT NULL,
                              TRANSFER_RATE     DOUBLE       NOT NULL,
                              PREFETCHSIZE      INTEGER      NOT NULL,
                              EXTENTSIZE        INTEGER      NOT NULL,
                              CLUSTER           DOUBLE       NOT NULL,
                              NLEAF             INTEGER      NOT NULL,
                              NLEVELS           INTEGER      NOT NULL,
                              FULLKEYCARD       BIGINT       NOT NULL,
                              OVERFLOW          INTEGER      NOT NULL,
                              FIRSTKEYCARD      BIGINT       NOT NULL,
                              FIRST2KEYCARD     BIGINT       NOT NULL,
                              FIRST3KEYCARD     BIGINT       NOT NULL,
                              FIRST4KEYCARD     BIGINT       NOT NULL,
                              SEQUENTIAL_PAGES  INTEGER      NOT NULL,
                              DENSITY           INTEGER      NOT NULL,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                   REFERENCES EXPLAIN_STATEMENT
                                   ON DELETE CASCADE )
```

EXPLAIN_OPERATOR 表定义

```
CREATE TABLE EXPLAIN_OPERATOR ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                SOURCE_NAME       VARCHAR(128) NOT NULL,
                                SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                STMTNO            INTEGER      NOT NULL,
                                SECTNO            INTEGER      NOT NULL,
                                OPERATOR_ID       INTEGER      NOT NULL,
                                OPERATOR_TYPE     CHAR(6)      NOT NULL,
                                TOTAL_COST        DOUBLE       NOT NULL,
                                IO_COST           DOUBLE       NOT NULL,
                                CPU_COST          DOUBLE       NOT NULL,
                                FIRST_ROW_COST    DOUBLE       NOT NULL,
                                RE_TOTAL_COST     DOUBLE       NOT NULL,
                                RE_IO_COST        DOUBLE       NOT NULL,
                                RE_CPU_COST       DOUBLE       NOT NULL,
                                COMM_COST         DOUBLE       NOT NULL,
                                FIRST_COMM_COST   DOUBLE       NOT NULL,
                                REMOTE_TOTAL_COST DOUBLE       NOT NULL,
                                REMOTE_COMM_COST  DOUBLE       NOT NULL,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

EXPLAIN_PREDICATE 表定义

```
CREATE TABLE EXPLAIN_PREDICATE ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 OPERATOR_ID       INTEGER      NOT NULL,
                                 PREDICATE_ID      INTEGER      NOT NULL,
                                 HOW_APPLIED       CHAR(5)      NOT NULL,
                                 WHEN_EVALUATED    CHAR(3)      NOT NULL,
                                 RELOP_TYPE        CHAR(2)      NOT NULL,
                                 SUBQUERY          CHAR(1)      NOT NULL,
                                 FILTER_FACTOR     DOUBLE       NOT NULL,
                                 PREDICATE_TEXT    CLOB(1M)     NOT LOGGED,
                                      FOREIGN KEY (EXPLAIN_REQUESTER,
                                                   EXPLAIN_TIME,
                                                   SOURCE_NAME,
                                                   SOURCE_SCHEMA,
                                                   EXPLAIN_LEVEL,
                                                   STMTNO,
                                                   SECTNO)
                                      REFERENCES EXPLAIN_STATEMENT
                                      ON DELETE CASCADE )
```

EXPLAIN_STATEMENT 表定义

```
CREATE TABLE EXPLAIN_STATEMENT ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                                 EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                                 SOURCE_NAME       VARCHAR(128) NOT NULL,
                                 SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                                 EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                                 STMTNO            INTEGER      NOT NULL,
                                 SECTNO            INTEGER      NOT NULL,
                                 QUERYNO           INTEGER      NOT NULL,
                                 QUERYTAG          CHAR(20)     NOT NULL,
                                 STATEMENT_TYPE    CHAR(2)      NOT NULL,
                                 UPDATABLE         CHAR(1)      NOT NULL,
                                 DELETABLE         CHAR(1)      NOT NULL
                                 TOTAL_COST        DOUBLE       NOT NULL,
                                 STATEMENT_TEXT    CLOB(1M)     NOT NULL
                                                                NOT LOGGED,
                                 SNAPSHOT          BLOB(10M)    NOT LOGGED,
                                 QUERY_DEGREE      INTEGER      NOT NULL,
                                     PRIMARY KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA,
                                                  EXPLAIN_LEVEL,
                                                  STMTNO,
                                                  SECTNO),
                                     FOREIGN KEY (EXPLAIN_REQUESTER,
                                                  EXPLAIN_TIME,
                                                  SOURCE_NAME,
                                                  SOURCE_SCHEMA)
                                     REFERENCES EXPLAIN_INSTANCE
                                     ON DELETE CASCADE )
```

EXPLAIN_STREAM 表定义

```
CREATE TABLE EXPLAIN_STREAM ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
                              EXPLAIN_TIME      TIMESTAMP    NOT NULL,
                              SOURCE_NAME       VARCHAR(128) NOT NULL,
                              SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
                              EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
                              STMTNO            INTEGER      NOT NULL,
                              SECTNO            INTEGER      NOT NULL,
                              STREAM_ID         INTEGER      NOT NULL,
                              SOURCE_TYPE       CHAR(1)      NOT NULL,
                              SOURCE_ID         SMALLINT     NOT NULL,
                              TARGET_TYPE       CHAR(1)      NOT NULL,
                              TARGET_ID         SMALLINT     NOT NULL,
                              OBJECT_SCHEMA     VARCHAR(128),
                              OBJECT_NAME       VARCHAR(128),
                              STREAM_COUNT      DOUBLE       NOT NULL,
                              COLUMN_COUNT      SMALLINT     NOT NULL,
                              PREDICATE_ID      INTEGER      NOT NULL,
                              COLUMN_NAMES      CLOB(1M)     NOT LOGGED,
                              PMID              SMALLINT     NOT NULL,
                              SINGLE_NODE       CHAR(5),
                              PARTITION_COLUMNS CLOB(64K)    NOT LOGGED,
                                  FOREIGN KEY (EXPLAIN_REQUESTER,
                                               EXPLAIN_TIME,
                                               SOURCE_NAME,
                                               SOURCE_SCHEMA,
                                               EXPLAIN_LEVEL,
                                               STMTNO,
                                               SECTNO)
                                  REFERENCES EXPLAIN_STATEMENT
                                  ON DELETE CASCADE )
```

ADVISE_INDEX 表定义

```
CREATE TABLE ADVISE_INDEX (EXPLAIN_REQUESTER VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_TIME      TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           SOURCE_NAME       VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           SOURCE_SCHEMA     VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           EXPLAIN_LEVEL     CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           STMTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           SECTNO            INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYNO           INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           QUERYTAG          CHAR(20)     NOT NULL
                              WITH DEFAULT '',
                           NAME              VARCHAR(128) NOT NULL,
                           CREATOR           VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           TBNAME            VARCHAR(128) NOT NULL,
                           TBCREATOR         VARCHAR(128) NOT NULL
                              WITH DEFAULT '',
                           COLNAMES          CLOB(64K)    NOT NULL,
                           UNIQUERULE        CHAR(1)      NOT NULL
                              WITH DEFAULT '',
                           COLCOUNT          SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           IID               SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           NLEAF             INTEGER      NOT NULL
                              WITH DEFAULT 0,
                           NLEVELS           SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           FIRSTKEYCARD      BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           FULLKEYCARD       BIGINT       NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERRATIO      SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CLUSTERFACTOR     DOUBLE       NOT NULL
                              WITH DEFAULT 0,
                           USERDEFINED       SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           SYSTEM_REQUIRED   SMALLINT     NOT NULL
                              WITH DEFAULT 0,
                           CREATE_TIME       TIMESTAMP    NOT NULL
                              WITH DEFAULT CURRENT TIMESTAMP,
                           STATS_TIME        TIMESTAMP
                              WITH DEFAULT CURRENT TIMESTAMP,
                           PAGE_FETCH_PAIRS  VARCHAR(254) NOT NULL
                              WITH DEFAULT '',
```

```
REMARKS           VARCHAR(254)
                     WITH DEFAULT '',
DEFINER           VARCHAR(128) NOT NULL
                     WITH DEFAULT '',
CONVERTED          CHAR(1) NOT NULL
                     WITH DEFAULT '',
SEQUENTIAL_PAGES  INTEGER      NOT NULL
                     WITH DEFAULT 0,
DENSITY           INTEGER      NOT NULL
                     WITH DEFAULT 0,
FIRST2KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST3KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
FIRST4KEYCARD     BIGINT       NOT NULL
                     WITH DEFAULT 0,
PCTFREE           SMALLINT     NOT NULL
                     WITH DEFAULT -1,
UNIQUE_COLCOUNT   SMALLINT     NOT NULL
                     WITH DEFAULT -1,
MINPCTUSED        SMALLINT     NOT NULL
                     WITH DEFAULT 0,
REVERSE_SCANS     CHAR(1)      NOT NULL
                     WITH DEFAULT 'N',
USE_INDEX         CHAR(1),
CREATION_TEXT     CLOB(1M)     NOT NULL
                     NOT LOGGED WITH DEFAULT '',
PACKED_DESC       BLOB(1M)     NOT LOGGED)
```

ADVISE_WORKLOAD 表定义

```
CREATE TABLE ADVISE_WORKLOAD (WORKLOAD_NAME  CHAR(128)    NOT NULL
                                WITH DEFAULT 'WK0',
                              STATEMENT_NO   INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              STATEMENT_TEXT CLOB(1M)     NOT NULL NOT LOGGED,
                              STATEMENT_TAG  VARCHAR(256) NOT NULL
                                WITH DEFAULT '',
                              FREQUENCY      INTEGER      NOT NULL
                                WITH DEFAULT 1,
                              IMPORTANCE     DOUBLE       NOT NULL
                                WITH DEFAULT 1,
                              COST_BEFORE    DOUBLE,
                              COST_AFTER     DOUBLE)
```

解释表

附录C. SQL 解释工具

db2expln 工具描述为存储在系统目录表中的程序包内的静态 SQL 语句选择的存取方案。对于在联编时未捕捉到解释数据的程序包，可使用此工具来获得该程序包的所选存取方案的快捷解释。

dynexpln 工具描述为动态语句选择的存取方案。它为语句创建静态程序包，然后使用 **db2expln** 工具描述它们。

可以使用这些解释工具来了解为特定 SQL 语句选择的存取方案。或者可使用集成解释设施(第177页的『第7章 SQL 解释设施』) 以及 Visual Explain 来了解为特定 SQL 语句选择的存取方案。动态和静态 SQL 语句都可以使用解释设施来解释。与解释工具的一个不同之处是，使用 Visual Explain 时，解释信息是以图形方式显示的。而其他方面，这两种方法所提供的细节级别是等效的。

要充分使用 db2expln 和 dynexpln 的输出，您必须了解:

- 支持的不同 SQL 语句和与那些语句相关的术语（如 SELECT 语句中的谓词）。
- 程序包（存取方案）的用途。（有关此信息，参见第134页的『数据存取概念和优化』。）
- 系统目录表的用途和内容。（有关详情，参考 *SQL Reference*。）
- 第35页的『第2部分 调整应用程序性能』中描述的其他概念。

下列主题提供有关 db2expln 和 dynexpln 的信息:

- 运行 db2expln 和 dynexpln
- db2expln 语法和参数
- db2expln 用法注释
- dynexpln 语法和参数
- dynexpln 用法注释
- db2expln 和 dynexpln 输出说明
- db2expln 和 dynexpln 输出示例。

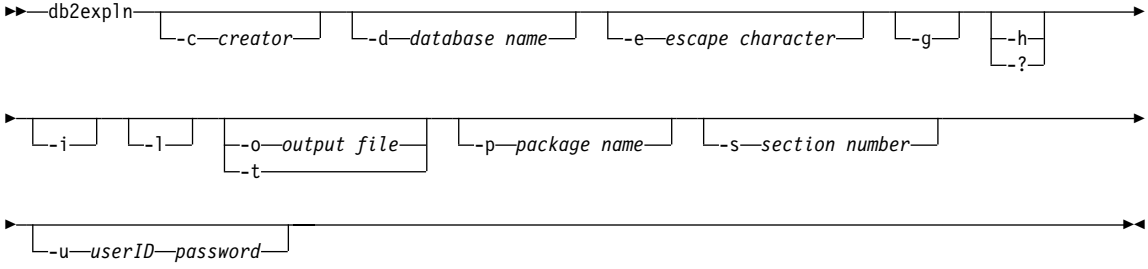
运行 db2expln 和 dynexpln

这些解释工具（db2expln 和 dynexpln）位于您的实例 sqllib 目录的 misc 子目录中。如果 db2expln 和 dynexpln 不在您当前的目录中，它们一定位于在 PATH 环境变量中出现的某个目录中。

db2expln 程序在第一次存取一个数据库时，使用 db2expln.bnd 文件将它自己与该数据库连接和联编。db2expln.bnd 文件位于 sqllib 目录的 bnd 子目录中。

要运行 db2expln，您必须对系统目录视图具有 SELECT 特权，对 db2expln 程序包具有 EXECUTE 权限。要运行 dynexpln，您必须对数据库具有 BINDADD 权限，对要解释的 SQL 语句具有任何需要的特权。（注意，如果您具有 SYSADM 或 DBADM 权限，您将自动具有所有这些授权级别。）

db2expln 语法和参数



其中:

-c creator

程序包创建者的用户 ID。

如果未指定此选项，将提示您输入它。

您可使用模式匹配字符即在 LIKE 谓词中可用的百分比符号 (%) 和下划线字符 (_) 来指定创建者名。

-d database name

包含要解释的程序包的数据库的名称。

如果未指定此选项，将提示您输入它。

-e escape character

用于指定要解释为转义字符而非模式匹配字符的字符。

例如，解释程序包 TESTID.CALC% 的 db2expln 命令为 db2expln -c TESTID -p CALC%。 但是，此命令还可以解释以 CALC 开头的任何其他方案。要只解释 TESTID.CALC% 程序包，您必须使用转义字符。通过将该命令更改为: db2expln -c TESTID -e ! -p CALC!%，您指定将 ! 字符用作转义字符，并将 !% 解释为 % 字符。

-g 显示优化器方案图。检查每一程序段，并构造原始优化器方案图（如 Visual Explain 所显示的）。注意，生成的图可能与原始方案不匹配。

-h 或 -?

获取有关输入参数的帮助信息。指定此选项会替换所有其他选项。

-i 在解释的方案中显示运算符 ID。运算符 ID 允许将 db2expln 的输出与解释设施的输出匹配。

-l 如果指定了此选项，则程序包名可以是小写或大小写混合。如果未指定 **-l** 选项，则会将程序包名转换为大写的

-o output file

db2expln 将结果写入其中的文件的名称。

如果指定 **-o** 而未带文件名，会提示您输入文件名。缺省文件名是 db2expln.out。

-p package name

要解释的程序包的名称。

如果未指定此选项，会提示您提供它。

您可使用模式匹配字符即在 LIKE 谓词中可使用的百分比符号 (%) 和下划线字符 (_) 来指定程序包名。

-s section number

程序包内要解释的程序段号。如果您希望解释程序包中所有程序段，则指定程序段号为零 (0)。如果程序包创建者 (**-c**) 或程序包名 (**-p**) 自变量暗示将解释多个程序包，因此将解释多个程序段，则提供的程序段值将被零 (0) 替换。

如果未指定此选项，会提示您提供它。

可以查询系统目录 SYSCAT.STATEMENTS 来查找程序段号（有关系统目录表的说明，参考 *SQL Reference*。）

-t 将输出定向到终端。

如果未指定 **-o** 或 **-t**，将提示您输入文件名，缺省情况是在终端显示输出。

-u userID password

当与一个数据库连接时，使用提供的用户 ID 和口令。

用户 ID 和口令都必须符合命名约定，且是数据库可识别的。

上述某些选项标志可能对于您的操作系统具有特殊含义，因此，在 db2expln 命令行中可能无法正确地解释它们。但是，可能可以在这些字符之前加转义字符来输入它们。有关详情，参见您的操作系统的用户手册。

db2expln 生成的帮助和初始状态信息被写入标准输出。所有由该解释工具生成的提示和其他状态信息被写入标准错误。根据选择的输出选项，解释文本被写入标准输出或文件。

使用 **-p** 和 **-c** 选项，并使用 LIKE 模式为程序包和创建者指定字符串常量，可对解释工具调用一次来解释多个方案。即，可以使用下划线字符 (_) 来表示单个字符，使用百分比符号 (%) 来表示出现的零个或多个字符。

例如，要解释 SAMPLE 数据库中所有程序包的所有程序段，并将结果写入文件 **my.exp** 中，输入

```
db2expln -d SAMPLE -p % -c % -s 0 -o my.exp
```

db2expln 用法注释

以下是 db2expln 显示的常见信息:

- 找不到数据库 <database> 内程序包模式为: <creator>.<package> 的程序包。

  如果在数据库中找不到与指定模式匹配的程序包，则会输出此信息。

- 可在 db2expln.msg 中找到联编信息

  如果 db2expln.bnd 的联编未成功，则会输出此信息。有关所遇到的问题的更详细信息可在当前目录中的 db2expln.msg 文件中找到。

- 以 0 替换程序段号，以查找潜在的多个程序包。

  如果 db2expln 可能遇到多个程序包，则会输出此信息。如果在程序包或创建者输入自变量中使用这些模式匹配字符之一，将会执行此操作。

- 程序包没有合格的静态程序段。

  如果指定的程序包只包含动态 SQL 语句，这表示没有静态程序段，则会输出此信息。

- 数据库 <database> 内的程序包 <creator>.<package> 无效。重新联编，然后重新运行 db2expln。

  如果指定的程序包当前无效，则会输出此信息。根据指示，重新发出 BIND 或 REBIND 命令，以便该方案在数据库中重建一个有效的程序包，然后重新运行 db2expln。

- 程序段未被处理: 这是由不受支持的发行版生成的。

  如果当前正在处理的程序段不是由对其提供了此 db2expln 可执行程序的 DB2 发行版生成的，也将输出此信息。在这种情况下，使用生成该程序段的 DB2 发行版中的 db2expln 副本。

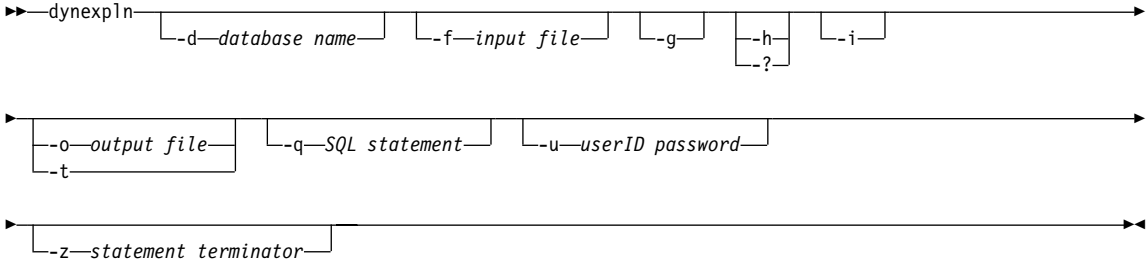
排除的 SQL 语句: 将不解释下列语句:

- BEGIN/END DECLARE SECTION

- BEGIN/END COMPOUND
- INCLUDE
- WHENEVER
- COMMIT and ROLLBACK
- CONNECT
- OPEN cursor
- FETCH
- CLOSE cursor
- PREPARE
- EXECUTE
- EXECUTE IMMEDIATE
- DESCRIBE
- Dynamic DECLARE CURSOR
- SQL 控制语句

一个复合 SQL 语句内的每个子语句可有它自己的程序段，这些程序段可通过 **db2expln** 来解释。

dynexpln 语法和参数



其中:

-d database name

包含要解释的程序包的数据库的名称。

如果未指定此选项，将提示您输入它。

-f input file

包含要解释的 SQL 语句的文件的名称。

除非您使用语句结束符 (**-e**) 选项，否则，该文件的每一行只应出现一条 SQL 语句。可在该文件中输入 SQL 注解。SQL 注解以 -- 开头，直到该行结尾。

-g 显示优化器方案图。检查每一程序段，并构造原始优化器方案图（如 Visual Explain 所显示的）。注意，生成的图可能与原始方案不匹配。

-h 或 -?
获取有关输入参数的帮助信息。指定此选项会替换所有其他选项。

-i 在解释的方案中显示运算符 ID。运算符 ID 允许将 db2expln 的输出与解释设施的输出匹配。

-o output file
db2expln 将结果写入其中的文件的名称。

-q SQL statement
要解释的 SQL 语句。

如果未指定此选项且未指定输入文件 (**-f**) 可选参数，则会提示您提供要解释的 SQL 语句。

如果同时指定了此选项和输入文件 (**-f**) 可选参数，则 dynexpln 将首先描述由 SQL 语句 (**-s**) 选项提供的语句，然后描述输入文件 (**-f**) 中的语句。

-t 将输出定向到终端。

如果输出 (**-o**) 和 **-t** 选项都指定了，则将输出定向到终端。

如果未指定输出文件 (**-o**) 或 **-t** 选项，将提示您输入文件名，缺省情况是将输出显示在终端上。

-u userID password
当与一个数据库连接时，使用提供的用户 ID 和口令。

用户 ID 和口令都必须符合命名约定，且是数据库可识别的。

-z statement terminator
用于指示已达到一条 SQL 语句的结尾的字符。

缺省情况是没有语句结束符。如果不使用此选项，则认为文件的每一行是单独的一条 SQL 语句。如果使用此选项，dynexpln 将使用指定的终止字符来分隔这些语句。

上述某些选项标志可能对于您的操作系统具有特殊含义，因此，在 dynexpln 命令行中可能无法正确地解释它们。但是，可能可以在这些字符之前加转义字符来输入它们。有关详情，参见您的操作系统的用户手册。

如果使用语句结束符 (**-e**) 选项，则可使用 SQL 语句 (**-s**) 选项来输入多个语句。为此，您应该使用终止字符将这些语句隔开。

dynexpln 生成的帮助和初始状态信息被写入标准输出。所有由该解释工具生成的提示和其他状态信息被写入标准错误。根据选择的输出选项，解释文本被写入标准输出或文件。

例如，要与名为 SAMPLE 的数据库连接并解释文件 TRYIT 中的所有语句，并将结果写入文件 my.exp 中，须输入

```
dynexpln -d SAMPLE -f TRYIT -o my.exp
```

dynexpln 用法注释

要解释动态语句，dynexpln 会为这些语句创建一个静态应用程序，然后调用 db2expln。要创建静态语句， dynexpln 生成一个含有这些语句的小 C 程序，然后调用 DB2 预编译程序来创建该程序包。(生成的 C 程序不完整，且不能被编译；它只包含足够预编译程序构建该程序包用的信息。)

以下是 dynexpln 显示的常见信息:

- 来自 db2expln 的所有错误信息。

  由于 dynexpln 调用 db2expln，所以查看 db2expln 的大多数错误信息是可能的。

- 与数据库连接时出错。

  如果在与数据库连接时发生错误，则会输出此信息。也将显示一个 CLI 错误信息，指示该连接未能完成的原因。校正错误的起因，再次运行 dynexpln。

- 在 dynexpln 运行之前，必须除去文件 "<filename>"。

  如果运行 dynexpln 时，给定的文件存在，则会显示此信息。除去该文件，或更改 DYNEXPLN_PACKAGE 环境变量的值，以更改将创建的文件名，然后再次运行 dynexpln。

- 在 dynexpln 运行之前，必须卸下程序包 "<creator>.<package>"。

  如果在运行 dynexpln 时，给定的文件存在，则会显示此信息。卸下该程序包，再运行；或更改 **DYNEXPLN_PACKAGE** 环境变量的值，以更改将创建的程序包的名称，然后再次运行 dynexpln。

- 写入文件 "<filename>" 时出错。

  如果不能写入给定的文件，则会显示此信息。确保 dynexpln 可以在当前目录中写入文件，并再次运行它。

- 读取输入文件 "<filename>" 时出错。

如果不能读取用 **-f** 选项给出的文件，则会显示此信息。确保该文件存在且 dynexpln 可以读取它。然后再次运行 dynexpln。

环境变量: dynexpln 可以使用两个不同的环境变量:

- **DYNEXPLN_OPTIONS** 是为语句构建程序包时使用的 SQL 预编译程序选项。使用与通过 CLP 发出 PREP 命令时将要用的相同的语法变量。

  例如: DYNEXPLN_OPTIONS="OPTLEVEL 5 BLOCKING ALL"
- **DYNEXPLN_PACKAGE** 是在数据库中创建的程序包的名称。要描述的语句位于此程序包中。如果未定义此变量，则赋予该程序包缺省值 **DYNEXPLN**。（只使用此环境变量中该名称的前八个字符。）

  该名称也用于创建 dynexpln 使用的中间文件的名称。

db2expln 和 dynexpln 输出说明

在该输出中，每个程序包的解释信息分成两个部分:

- 程序包信息，如联编日期和相关的联编选项
- 程序段信息，如程序段号，后跟要解释的 SQL 语句。在程序段信息之下将是为显示的 SQL 语句选择的存取方案的解释输出。

一个存取方案或程序段的步骤将以数据库管理程序执行它们的次序显示。每个主要步骤将显示为一个左对齐标题，有关该步骤的信息以缩进形式显示在该标题下。该存取方案的解释输出有一个缩进条，它位于该输出的左边。这些缩进条还提供操作“范围”；在返回至缩进的上一级之前，先处理处于同一个操作内较低缩进级别（即，右边的那些）的操作。

重要的一点是，要记住选择的存取方案是基于原始 SQL 语句（输出中显示的语句）的增强版本。 例如，原始语句可能导致任意多个触发器和约束被激活。 而且，可以通过 SQL 编译程序的“查询重写”部件，将该 SQL 语句重新编写为一种等效但更有效的格式。当优化器确定满足该语句的最有效的方案时，会在提供给优化器的信息中包括所有这些因素。因此，在解释输出中显示的存取方案可能与对原始 SQL 语句期望的存取方案完全不同。集成解释设施（参见第177页的『第7章 SQL 解释设施』）显示用于优化的实际 SQL 语句，其格式为类似于 SQL 的语句，它是通过反向转换查询的内部表示而创建的。

当将 db2expln 或 dynexpln 的输出与解释设施的输出比较时，运算符 ID 选项 **(-i)** 可能非常有用。每次 db2expln 或 dynexpln 开始处理解释设施中的新运算符时，就会将该新运算符 ID 号打印在解释方案的左边。运算符 ID 可以用于与各种格式的存取方案中的步骤匹配。注意，在解释设施输出中的运算符与 db2expln 和 dynexpln 显示的操作之间不会始终都存在一一对应的关系。

下列主题描述 db2expln 和 dynexpln 可能生成的解释文本:

- 表存取
- 临时表
- 连接
- 数据流
- 插入、更新和删除
- 行标识符 (RID) 准备
- 聚合
- 并行处理
- 联合体语句处理
- 其他语句。

表存取

此语句告知正在存取的表的名称和类型。它有两种使用格式:

1. 三种类型的常规表:
   - 存取表名:

     ```
     Access Table Name = schema.name  ID = ts,n
     ```

     其中:
     - *schema.name* 是正在存取的表的全限定名
     - *ID* 是该表的 SYSCAT.TABLES 目录中对应的 TABLESPACEID 和 TABLEID
   - 存取层次结构表名:

     ```
     Access Hierarchy Table Name = schema.name  ID = ts,n
     ```

     其中:
     - *schema.name* 是正在存取的表的全限定名
     - *ID* 是该表的 SYSCAT.TABLES 目录中对应的 TABLESPACEID 和 TABLEID
   - 存取摘要表名:

     ```
     Access Summary Table Name = schema.name  ID = ts,n
     ```

     其中:
     - *schema.name* 是正在存取的表的全限定名

– *ID* 是该表的 SYSCAT.TABLES 目录中对应的 TABLESPACEID 和 TABLEID

2. 两种类型的临时表:
   - 存取临时表 ID:

     ```
     Access Temp Table  ID = tn
     ```

     其中:
     – *ID* 是 db2expln 指定的对应标识符
   - 存取已说明全局临时表 ID:

     ```
     Access Global Temp Table  ID = ts,tn
     ```

     其中:
     – *ID* 是该表 (ts) 的 SYSCAT.TABLES 目录中对应的 TABLESPACEID; 以及由 db2expln (tn) 指定的对应标识符

在表存取语句之后，将提供附加语句以进一步描述该存取。这些语句以缩进形式显示在表存取语句之下。可能的语句有:

- 列数
- 并行扫描
- 扫描方向
- 行存取方法
- 锁定意图
- 谓词
- 其他表语句。

列数

以下语句指示该表的每一行中所用的列数:

```
#Columns = n
```

并行扫描

以下语句指示数据库管理程序将使用几个子代理程序来并行读取该表:

```
Parallel Scan
```

如果未显示此文本，则该表只能由一个代理程序（或子代理程序）读取。

扫描方向

以下语句指示数据库管理程序将按倒序读取行:

```
扫描方向 = 反向
```

如果未显示此文本，则扫描方向为正向，这是缺省值。

行存取方法

将显示下列语句之一，指示如何存取表中的限定行:

- 关系扫描语句指示按顺序扫描该表，以查找限定行。
  - 以下语句指示将不执行数据的预读取:

    ```
        Relation Scan
    |   Prefetch: None
    ```

  - 以下语句指示优化器已预先确定了将预读取的页数:

    ```
        Relation Scan
    |   Prefetch: n Pages
    ```

  - 以下语句指示应该预读取数据:

    ```
    Relation Scan
    |   Prefetch: Eligible
    ```

  - 以下语句指示正在通过索引标识和存取限定行:

    ```
    Index Scan:  Name = schema.name  ID = xx
    |   Index Columns:
    ```

    其中:

    - *schema.name* 是正在扫描的索引的全限定名
    - *ID* 是 SYSCAT.INDEXES 目录视图中的对应 IID 列。

    对于索引中的每一列，都有一行跟在后面。将以下列一种形式列出索引中的每一列:

    ```
    n: column_name (Ascending)
    n: column_name (Descending)
    n: column_name (Include Column)
    ```

    提供以下语句，以说明索引扫描的类型:

    - 通过以下项显示索引的范围定界谓词:

      ```
      #Key Columns = n
          |   Start Key: xxxxx
          |   Stop Key: xxxxx
      ```

      其中 xxxxx 是下列其中一个:

      - 索引开始
      - 索引结束
      - 包括的值: 或排除的值:

将会在该索引扫描中包括内含的关键字值。不会在扫描中包括排除的关键字值。下列其中一行将对关键字的每一部分给出该关键字的值:

```
n: 'string'
n: nnn
n: yyyy-mm-dd
n: hh:mm:ss
n: yyyy-mm-dd hh:mm:ss.uuuuuu
n: NULL
n: ?
```

如果出现文字串，则显示前面 20 个字符。如果该字符串长于 20 个字符，则在该字符串的最后显示 ...。一些关键字只有在执行该部分时才能确定。因此会显示 ? 作为其值。

- 纯索引存取

  如果可以从索引关键字获得所有需要的列，将出现此语句，且不会存取任何表数据。

- 以下语句指示将不执行索引页的预读取:

  ```
  Index Prefetch: None
  ```

- 以下语句指示应该预读取索引页:

  ```
  Index Prefetch: Eligible
  ```

- 以下语句指示将不执行数据页的预读取:

  ```
  Data Prefetch: None
  ```

- 以下语句指示应该预读取数据页:

  ```
  Data Prefetch: Eligible
  ```

- 如果存在可以传送至“索引管理程序”以帮助限定索引项的谓词，则以下语句用于显示谓词数:

  ```
  Sargable Index Predicate(s)
  | #Predicates = n
  ```

– 直接读取语句指示正在通过使用在存取方案中先前准备的行 ID (RID) 来存取限定行。

锁定意图

对于每个表存取，都会用以下语句显示将在表和行级别获取的锁定类型:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

表锁定的可能值有:

- 独占
- 有意独占

- 无任何意图
- 有意共享
- 共享
- 共享有意独占
- 超互斥
- 更新

行锁定的可能的值有:

- 独占
- 下一个关键字独占（不在 db2expln 输出中显示）
- 无
- 共享
- 下一个关键字共享
- 更新
- 下一个关键字弱互斥
- 弱互斥

这些锁定类型的说明可在第44页的『锁定的属性』中查找。

谓词

有两个语句，它们提供有关在存取方案中使用的谓词的信息:

1. 以下语句指示一旦返回数据，将求出的谓词数:

```
Residual Predicate(s)
#Predicates = n
```

2. 以下语句指示当正在存取数据时将求出的谓词数。谓词的计数不包括压入操作，如聚合或排序。

```
Sargable Predicate(s)
#Predicates = n
```

在上述语句中显示的谓词数也许未能反映在 SQL 语句中提供的谓词数，因为谓词可以:

- 在同一个查询中应用多次
- 在查询优化过程期间，通过添加隐式谓词来转换和扩展
- 在查询优化过程期间，被转换和压缩成更少的谓词。

其他表语句

- 以下语句指示将只存取一行:

  ```
  Single Record
  ```

- 当此表存取所使用的隔离级别与该程序包所用的不同时，就会出现以下语句:

  ```
  Isolation Level: xxxx
  ```

  由于许多原因，可能使用不同的隔离级别:

  – 一个程序包是用“可重复读取”联编的，这影响了参考完整性约束；为检查参考完整性约束而对父表进行的存取会被降级为“游标稳定性”隔离级别，以避免对此表持有不必要的锁定。

  – 用“未落实读取”联编的程序包发出 DELETE 或 UPDATE 语句；用于实际删除的表存取会被升级为“游标稳定性”。

- 如下语句指示如果有足够的排堆内存可用，从临时表读取的部分或全部行将高速缓存到缓冲池外:

  ```
  Keep Rows In Private Memory
  ```

- 如果该表具有易变的基数属性集，则会通过以下信息指出:

  ```
  Volatile Cardinality
  ```

临时表

存取方案在瞬态或临时工作表中执行操作期间，它使用临时表来存储数据。仅当执行存取方案时，此表才存在。通常，当在存取方案中需要提前对子查询求值时，或当中间结果不适合可用内存时，会使用临时表。

如果需要创建临时表，则会出现两个可能的语句中的一个。这些语句指示将创建一个临时表，并将行插入其中。该 ID 是引用临时表时为方便起见而由 db2expln 指定的标识符。此 ID 以字母 ’t’ 为前缀，以指示该表是临时表。

- 以下语句指示将创建一个普通的临时表:

  ```
  Insert Into Temp Table  ID = tn
  ```

- 以下语句指示多个子代理程序将并行创建一个普通的临时表:

  ```
  Insert Into Shared Temp Table  ID = tn
  ```

- 以下语句指示将创建一个已排序的临时表:

  ```
  Insert Into Sorted Temp Table  ID = tn
  ```

- 以下语句指示多个子代理程序将并行创建一个已排序的临时表:

  ```
  Insert Into Sorted Shared Temp Table  ID = tn
  ```

- 以下语句指示将创建一个已说明临时表:

  ```
  Insert Into Global Temp Table  ID = ts,tn
  ```

- 以下语句指示多个子代理程序将并行创建一个已说明临时表:

```
Insert Into Shared Global Temp Table  ID = ts,tn
```

- 以下语句指示将创建一个已排序已说明临时表:

```
Insert Into Sorted Global Temp Table  ID = ts,tn
```

- 以下语句指示多个子代理程序将并行创建一个已排序已说明临时表:

```
Insert Into Sorted Shared Global Temp Table  ID = ts,tn
```

上述每个语句都将后跟下列内容:

```
#Columns = n
```

它指示要插入临时表中的每一行有多少列。

已排序的临时表

已排序的临时表可以由类似如下的操作产生:

- ORDER BY
- DISTINCT
- GROUP BY
- Merge Join
- '= ANY' 子查询
- '<> ALL' 子查询
- INTERSECT 或 EXCEPT
- UNION（不带 ALL 关键字）

在已排序的临时表的原始创建语句后，可跟许多附加语句:

- 以下语句指示在排序中使用的关键字列数:

```
#Sort Key Columns = n
```

对于排序关键字中的每一列，将会显示下列其中一行:

```
Key n: column_name (Ascending)
Key n: column_name (Descending)
Key n: (Ascending)
Key n: (Descending)
```

- 以下语句提供对行数和行大小的估计，以便在运行时可以分配最优的排序堆。

```
Sortheap Allocation Parameters:
  #Rows      = n
  Row Width = n
```

- 如果只需要排序结果的前几行，则会显示下列内容:

```
Sort Limited To Estimated Row Count
```

- 对于在“对称多处理器”(SMP) 环境中的排序，要执行的排序的类型由下列语句之一指示:

  ```
  Use Partitioned Sort
  Use Shared Sort
  Use Replicated Sort
  Use Round-Robin Sort
  ```

  有关不同排序技术的说明，参见第162页的『并行排序策略』。

- 以下语句指示排序产生的结果是否留在排序堆中:

  ```
  Piped
  ```

  和

  ```
  Not Piped
  ```

  如果指示管道排序，则数据库管理程序会将排序的输出保留在内存中，而不是将排序结果置于另一个临时表中。（有关管道排序与非管道排序的说明，参见第159页的『使用优化器排序的影响』。）

- 以下语句指示在排序期间将除去重复的值:

  ```
  消去重复
  ```

- 如果正在排序中执行聚合，则以下语句之一会指示它:

  ```
  Partial Aggregation
  Intermediate Aggregation
  Buffered Partial Aggregation
  Buffered Intermediate Aggregation
  ```

临时表完成

在包含用于创建临时表的压入操作的表存取之后（即，创建临时表操作是在表存取范围内发生的），有一个“完成”语句，它通过让临时表准备向后续的临时表存取提供行，来处理文件结尾。将显示下列其中一行:

```
Temp Table Completion  ID = tn
Shared Temp Table Completion  ID = tn
Sorted Temp Table Completion  ID = tn
Sorted Shared Temp Table Completion  ID = tn
```

表函数

表函数是用户定义函数 (UDF)，它将数据以表的形式返回至该语句。有关表函数的详情，参考 *SQL Reference*。表函数由以下语句指示:

```
Access User Defined Table Function
|   Name = schema.funcname
|   Language = xxxx
|   Fenced   Deterministic   NULL Call   Disallow Parallel
```

编写表函数的语言（C、OLE 或 Java）是与表函数的属性一起提供的。

连接

连接有三种类型（有关这些连接的说明，参见第144页的『连接概念』）：

- 散列连接
- 合并连接
- 嵌套循环连接。

当在一个程序段中执行到要执行连接时，会显示下列其中一个语句:

```
散列连接
```

和／或

```
Merge Join
```

和／或

```
Nested Loop Join
```

执行左外连接是可能的。左外连接由下列其中一个语句指示:

```
Left Outer Hash Join
```

和／或

```
Left Outer Merge Join
```

和／或

```
Left Outer Nested Loop Join
```

对于合并和嵌套循环连接，连接的外部表将是在输出中显示的先前存取语句所引用的表。连接的内部表将是在连接语句范围内包含的存取语句所引用的表。对于散列连接，将存取语句反向，将外部表包含在连接范围之内，而内部表出现在连接之前。

对于散列或合并连接，可能出现下列附加语句:

- 在某些环境中，连接只需要确定内部表中的任何行是否与外部表中的当前行匹配。它是通过下列语句来指示的:

  ```
  Early Out: Single Match Per Outer Row
  ```

- 在连接完成之后，应用谓词是可能的。将指示应用的谓词数，如下所示:

  ```
  Residual Predicate(s)
  |  #Predicates = n
  ```

对于散列连接，可能出现下列附加语句:

- 已根据内部表构建了散列表。如果在进行内部表存取时将散列表的构建压入一个谓词中，则在存取内部表时用以下语句指示该构建:

  ```
  Process Hash Table For Join
  ```

- 当存取外部表时，可以构建一个试探表来提高连接的性能。在存取外部表时用以下语句指示试探表的构建:

  ```
  Process Probe Table For Hash Join
  ```

- 构建散列表所需的估计字节数由以下项表示:

  ```
  Estimated Build Size: n
  ```

- 构建试探表所需的估计字节数由以下项表示:

  ```
  Estimated Probe Size: n
  ```

对于嵌套循环连接，紧接在连接语句之后可能会出现以下附加语句:

```
Piped Inner
```

此语句指示连接的内部表是另一系列的操作的结果。这也称为*复合内部*。

如果一个连接涉及两个以上的表，则应从头到尾读取解释步骤。例如，假设解释输出具有下列数据流:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

执行的步骤将是:

1. 提取 W 中合格的行。
2. 将 W 中的行与 X 中的（下一）行连接，并调用结果 P1（表示编号 1 的部分连接结果）。
3. 将 P1 与 Y 中的（下一）行连接，以创建 P2。
4. 将 P2 与 Z 中的（下一）行连接，以获取一个完整的结果行。
5. 如果 Z 中存在其他行，则转至步骤 4。
6. 如果 Y 中存在其他行，则转至步骤 3。
7. 如果 X 中存在其他行，则转至步骤 2。
8. 如果 W 中存在其他行，则转至步骤 1。

数据流

在一个存取方案内，经常需要控制数据的创建以及数据从一系列操作到另一系列操作的流动。数据流的概念允许将一个存取方案内的一组操作当作一个单元来控制。数据流的开始由以下语句指示:

```
Data Stream n
```

其中，n 是由 db2expln 为方便引用而指定的唯一的标识符。数据流的结束由以下语句指示:

```
End of Data Stream n
```

这些语句之间的所有操作被认为是同一个数据流的一部分。

一个数据流有许多特征，在初始数据流语句之后可跟一个或多个语句，来描述这些特征:

- 如果数据流的操作依赖于存取方案中早先生成的值，则数据流标记为:

  ```
  Correlated
  ```

- 类似于已排序的临时表，下列语句指示数据流的结果是否将保留在内存中:

  ```
  Piped
  ```

  和

  ```
  Not Piped
  ```

  同临时表的情况一样，如果执行时没有足够的内存，则可能将管道数据流写入磁盘。该存取方案将提供这两种可能性。

- 以下语句指示此数据流只需要单个记录:

  ```
  Single Record
  ```

当存取一个数据流时，将会输出以下语句:

```
Access Data Stream n
```

插入、更新和删除

这些 SQL 语句的解释文本是自我解释的。这些 SQL 操作的语句文本可能为:

- `Insert: Table Name = schema.name ID = ts,n`
- `Update: Table Name = schema.name ID = ts,n`
- `Delete: Table Name = schema.name ID = ts,n`
- `Insert: Hierarchy Table Name = schema.name ID = ts,n`
- `Update: Hierarchy Table Name = schema.name ID = ts,n`

- Delete: Hierarchy Table Name = schema.name ID = ts,n
- Insert: Summary Table Name = schema.name ID = ts,n
- Update: Summary Table Name = schema.name ID = ts,n
- Delete: Summary Table Name = schema.name ID = ts,n
- Insert: Global Temporary Table ID = ts, tn
- Update: Global Temporary Table ID = ts, tn
- Delete: Global Temporary Table ID = ts, tn

行标识符 (RID) 准备

对于某些存取方案，如果对限定行标识符 (RID) 排序并除去了重复的标识符（如果是索引 OR 运算），或者在执行实际的表存取之前，使用一种技术来标识在所有正在存取的索引中出现的 RID（如果是索引 AND 运算），则会更有效。正如解释语句所指示，RID 准备有下列三种主要用途:

- 以下语句指示“索引 OR 运算”用于准备限定 RID 的列表:

  ```
  Index ORing RID Preparation
  ```

  *索引 OR 运算*是指建立多个索引存取，并组合结果以包括在存取的任何索引中出现的不同 RID 的一种技术。当通过 OR 关键字将谓词连接或存在 IN 谓词时，优化器将考虑索引 OR 运算。索引存取可以在相同索引或不同索引上进行。
- RID 准备的另一个用途是准备在列表预读取期间要使用的输入数据，如下所示:

  ```
  List Prefetch RID Preparation
  ```
- *索引 AND 运算*是指建立多个索引存取，并组合结果以包括在存取的所有索引中出现的 RID 的一种技术。索引 AND 运算的处理以下列语句开始:

  ```
  Index ANDing
  ```

  如果优化器已估计了结果集的大小，则用以下语句显示该估计值:

  ```
  Optimizer Estimate of Set Size: n
  ```

  索引 AND 运算过滤器操作处理 RID，并使用位过滤器技术来确定在存取的每个索引中出现的 RID。以下语句指示正在为索引 AND 运算处理 RID:

  ```
  Index ANDing Bitmap Build
  Index ANDing Bitmap Probe
  Index ANDing Bitmap Build and Probe
  ```

  如果优化器已估计了一个位图的结果集的大小，则会用以下语句显示该估计值:

  ```
  Optimizer Estimate of Set Size: n
  ```

对于任何类型的 RID 准备，如果可以执行列表预读取，则将用以下语句指示它:

```
Prefetch: Enabled
```

聚合

对满足指定标准的那些行执行聚合，如果存在这样的行，则它们是由 SQL 语句谓词提供的。如果要执行某种聚合函数，会出现下列其中一个语句:

```
Aggregation
Predicate Aggregation
Partial Aggregation
Partial Predicate Aggregation
Intermediate Aggregation
Intermediate Predicate Aggregation
Final Aggregation
Final Predicate Aggregation
```

谓词聚合指示在实际存取数据时，将聚合操作压入为一个谓词来处理。

在上述任何一种聚合语句之下将是要执行的聚合函数的类型指示:

- Group By
- 列函数
- 单个记录。

可从初始的 SQL 语句派生出特定的列函数。从索引读取单个记录，以满足 MIN 或 MAX 操作。

如果使用谓词聚合，则在出现了聚合的表存取语句之后，将是聚合“完成”，它在每个组完成或文件结束时执行任何需要的处理。显示下列其中一行:

```
Aggregation Completion
Partial Aggregation Completion
Intermediate Aggregation Completion
Final Aggregation Completion
```

并行处理

并行执行 SQL 语句（使用分区内或分区间并行性）需要一些特殊的操作。以下描述用于并行方案的操作。

- 当运行分区内并行方案时，将使用几个子代理程序同时执行该方案的各个部分。这些子代理程序的创建由下列语句指示:

  ```
  Process Using n Subagents
  ```

- 当运行分区间并行方案时，该程序段被分为几个子段。每个子段被发送至一个或多个节点，以便运行。一个重要的子段是*协调程序子段*。协调程序子段是每个方案中的第一个子段。它首先获得控制权，并负责分发其他子段，然后将结果返回至调用应用程序。

  子段的分发由以下语句指示:

  ```
  Distribute Subsection #n
  ```

  可用八种方式之一来确定接收子段的节点:

  – 以下示例指示将根据列值把子段发送至节点组内的一个节点。

    ```
    Directed by Hash
    |  #Columns = n
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – 以下示例指示将把子段发送至一个预定的节点。（当该语句使用 NODENUMBER() 函数时，会常常看到这种情况。）

    ```
    Directed by Node Number
    ```

  – 以下示例指示将把子段发送至与给定节点组中预定分区号对应的节点。（当该语句使用 PARTITION() 函数时，会常常看到这种情况。）

    ```
    Directed by Partition Number
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  – 以下示例指示将把子段发送至为应用程序的游标提供当前行的节点。

    ```
    Directed by Position
    ```

  – 以下示例指示只有编译该语句时确定的一个节点会接收该子段。

    ```
    Directed to Single Node
    |  Node Number = n
    ```

  – 以下示例指示将在协调程序节点上执行子段。

    ```
       Directed to Coordinator Node
    ```

  – 以下示例指示将把子段发送到列出的所有节点。

    ```
    Broadcast to Node List
    |  Nodes = n1, n2, n3, ...
    ```

  – 以下示例指示只有执行该语句时确定的一个节点会接收该子段。

    ```
    Directed to Any Node
    ```

- 表队列用于在一个分区数据库环境中的子段之间或一个对称多处理器 (SMP) 环境中的子代理程序之间移动数据。表队列的描述如下:

  – 以下语句指示正在将数据插入一个表队列:

    ```
    Insert Into Synchronous Table Queue  ID = qn
    Insert Into Asynchronous Table Queue  ID = qn
    Insert Into Synchronous Local Table Queue  ID = qn
    Insert Into Asynchronous Local Table Queue  ID = qn
    ```

– 对于数据库分区表队列，插入表队列中的行的目的地由下列一个语句描述:

```
Broadcast to Coordinator Node
```

将所有的行发送至协调程序节点。

```
Broadcast to All Nodes of Subsection n
```

将所有的行发送至给定子段运行所在的每个数据库分区。

```
Hash to Specific Node
```

根据行中的值，将每一行发送至一个数据库分区。

```
Send to Specific Node
```

将每一行发送至执行该语句时所确定的数据库分区。

```
Send to Random Node
```

将每一行发送至随机数据库分区。

– 在某些情况中，一个数据库分区表队列不得不临时将某些行溢出至临时表。这种可能性由下列语句标识:

```
Rows Can Overflow to Temporary Table
```

– 在包含将行插入一个表队列的压入操作的表存取之后，将有一个“完成”语句，它处理未能立即发送的行。显示下列其中一行:

```
Insert Into Synchronous Table Queue Completion  ID = qn
Insert Into Asynchronous Table Queue Completion  ID = qn
Insert Into Synchronous Local Table Queue Completion  ID = qn
Insert Into Asynchronous Local Table Queue Completion  ID = qn
```

– 以下语句指示正在从一个表队列中检索数据:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

这些信息之后始终是有关检索列数的指示。

```
#Columns = n
```

– 如果表队列在接收端对行排序，则表队列存取也将会有下列一条信息:

```
Output Sorted
Output Sorted and Unique
```

这些信息之后是有关用于排序操作的关键字数的指示。

```
#Key Columns = n
```

对于排序关键字中的每一列，显示下列其中一项:

```
Key n: (Ascending)
Key n: (Descending)
```

– 如果通过表队列的接收端将谓词应用于行，则会显示下列信息:

```
Residual Predicate(s)
|  #Predicates = n
```

- 一个分区数据库环境中的某些子段明确地循环回至该子段的开始处，这由以下语句指示:

```
Jump Back to Start of Subsection
```

联合体语句处理

在联合体数据库中执行 SQL 语句需要有对其他数据源执行部分语句的能力。

以下语句指示将要存取数据源:

```
Distributed Subquery #n
|  #Columns = n
```

有可能对分布式子查询返回的数据应用谓词。将指示应用的谓词数，如下所示:

```
Residual Predicate(s)
|  #Predicates = n
```

每个分布式子查询的详细资料都单独提供。分布式子查询的选项描述如下:

- 子查询的数据源通过下列其中一项显示:

```
Server: server_name (type, version)
Server: server_name (type)
Server: server_name
```

- 子查询的 SQL 语句显示成:

```
Subquery SQL Statement:
statement
```

- 子查询中引用的别名列示如下:

```
Nickname Referenced:
Schema.nickname Base = baseschema.basetable
```

- 在执行子查询之前，如果将值从联合体服务器传送至数据源，则值的数目将由下面这一项显示:

```
#Input Columns: n
```

- 在执行子查询之后，如果将值从数据源传送至联合体服务器，则值的数目将由下面这一项显示:

```
#Output Columns: n
```

其他语句

- 数据定义语言语句的程序段将在输出中用下列语句指示:

```
DDL Statement
```

不对 DDL 语句提供其他解释输出。

- 用于可更新的专用寄存器的 SET 语句（如 **CURRENT EXPLAIN SNAPSHOT**）的程序段将在输出中用以下语句指示:

  ```
  SET Statement
  ```

  不对 SET 语句提供其他解释输出。

- 如果 SQL 语句 包含 DISTINCT 子句，则输出中可能出现下列文本:

  ```
  Distinct Filter  #Columns = n
  ```

  其中，n 是参与获取不同行的列数。要检索不同行的值，必须对行排序，以便可以跳过重复的行。如果数据库管理程序不必明确消去重复行，如下列情况，则不显示此语句:

  – 存在唯一的索引，且索引关键字中的所有列都是 DISTINCT 操作的一部分
  – 在排序期间可以消去的重复项。

- 如果下一个操作取决于特定的记录标识符，将出现以下语句:

  ```
  Positioned Operation
  ```

  对于任何使用 WHERE CURRENT OF 语法的 SQL 语句，将会出现此语句。

- 如果存在必须应用于该结果但不能作为另一个操作的一部分来应用的谓词，将出现以下语句:

  ```
  Residual Predicate Application
  |  #Predicates = n
  ```

- 如果该 SQL 语句中有 UNION 运算符，将出现以下语句:

  ```
     UNION
  ```

- 如果存取方案中有一个操作，它的唯一目的是产生供后续操作使用的行值，将出现以下语句:

  ```
  Table Constructor
  |  n-Row(s)
  ```

  表构造器可以用于将一个集合中的值转换成可以传送至后续操作的一系列行。当提示表构造器输入下一行时，将出现以下语句:

  ```
  Access Table Constructor
  ```

- 如果存在只在特定情况下才处理的操作，将出现下列语句:

  ```
  Conditional Evaluation
  |  Condition #n:
  |  |  #Predicates = n
  |  Action #n:
  ```

条件判定用于实施象 SQL CASE 语句这样的活动，或象参考完整性约束或触发器这样的内部机制。如果未显示任何操作，则当该条件为真时，只处理数据处理操作。

- 如果在该存取方案中正在处理 ALL、ANY 或 EXISTS 子查询，则将显示下列其中一个语句:
  - ANY/ALL 子查询
  - EXISTS 子查询
  - EXISTS SINGLE 子查询
- 在特定的 UPDATE 和 DELETE 操作之前，需要建立表内特定行的位置。这由下列语句指示:

  ```
  Establish Row Position
  ```

- 如果有要返回至该应用程序的行，将会显示以下语句:

  ```
  Return Data to Application
  #Columns = n
  ```

  如果该操作被下推至表存取中，它将需要一个完成程序段。此程序段显示为:

  ```
  Return Data Completion
  ```

db2expln 和 dynexpln 输出示例

以下显示了五个示例，以帮助了解 db2expln 和 dynexpln 的输出的布局和格式。这些示例是对 DB2 中提供的 SAMPLE 数据库运行的。对每个示例都提供了简短的讨论。示例之间的主要区别用**黑体**显示。

示例一: 无并行性方案

本示例只请求一个列表，它包含所有雇员的姓名、职务、部门名称和位置，以及他们所从事的项目的名称。此存取方案的实质是使用合并连接将每个指定的表中的相关数据连接在一起。因为没有索引可用，故该存取方案执行每个表的关系扫描，且在连接之前必须对每个表排序。

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:47:58

Bind Timestamp = 2000-01-03-15.47.58.607455

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5
```

```
Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 126
Estimated Cardinality = 153

Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Insert Into Sorted Temp Table  ID = t1
|  |  #Columns = 3
|  |  #Sort Key Columns = 1
|  |  |  Key 1: DEPTNO (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 40
|  |  |  Row Width = 48
|  |  Piped
Sorted Temp Table Completion  ID = t1
Access Temp Table  ID = t1
|  #Columns = 3
|  Relation Scan
|  |  Prefetch: Eligible
   Merge Join
|  Access Table Name = DOOLE.PROJECT  ID = 2,7
|  |  #Columns = 2
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
|  |  Insert Into Sorted Temp Table  ID = t2
|  |  |  #Columns = 2
|  |  |  #Sort Key Columns = 1
|  |  |  |  Key 1: DEPTNO (Ascending)
|  |  |  Sortheap Allocation Parameters:
|  |  |  |  #Rows     = 38
|  |  |  |  Row Width = 28
```

```
|                  |  |  Piped
|                  |  Sorted Temp Table Completion  ID = t2
|                  |  Access Temp Table  ID = t2
|                  |  |  #Columns = 2
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                     Merge Join
|                  |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|                  |  |  #Columns = 3
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                  |  |  Lock Intents
|                  |  |  |  Table: Intent Share
|                  |  |  |  Row  : Next Key Share
|                  |  |  Insert Into Sorted Temp Table  ID = t3
|                  |  |  |  #Columns = 3
|                  |  |  |  #Sort Key Columns = 1
|                  |  |  |  |  Key 1: WORKDEPT (Ascending)
|                  |  |  |  Sortheap Allocation Parameters:
|                  |  |  |  |  #Rows     = 63
|                  |  |  |  |  Row Width = 32
|                  |  |  |  Piped
|                  |  Sorted Temp Table Completion  ID = t3
|                  |  |  Access Temp Table  ID = t3
|                  |  |  #Columns = 3
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                     Return Data to Application
|                  |  #Columns = 5
|
|                  End of section
|
|
|                  Optimizer Plan:
|
|                                 RETURN
|                               (   1)
|                                  |
|                  MSJOIN
|                               (   2)
|                               /      \
|                         MSJOIN        TBSCAN
|                         (   3)         (  12)
|                        /      \          |
|                     TBSCAN    TBSCAN     SORT
|                     (   4)    (   8)    (  13)
|                       |         |         |
|                     SORT      SORT      TBSCAN
|                     (   5)    (   9)    (  14)
|                       |         |         |
|                     TBSCAN    TBSCAN  Table:
|                     (   6)    (  10)   DOOLE
```

```
    |         |     EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
    DEPARTMENT  PROJECT
```

该方案的第一部分存取 DEPARTMENT 和 PROJECT 表，并用合并连接来连接它们。此连接的结果被连接到 EMPLOYEE 表。结果行返回至应用程序。

示例二: 具有分区内并行性的单分区数据库方案

本示例显示与第500页的『示例一: 无并行性方案』相同的 SQL 语句，但是此查询是为一个 4 路 SMP 机器编译的。

```
******************** PACKAGE ************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:48:51

Bind Timestamp = 2000-01-03-15.48.51.402403

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = Yes (Bind Degree = 4)

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 142
Estimated Cardinality = 153

Process Using 4 Subagents
|  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|  |  #Columns = 3
|  |  Parallel Scan
|  |  Relation Scan
|  |  |  Prefetch: Eligible
|  |  Lock Intents
|  |  |  Table: Intent Share
```

```
|                  |  |  |  Row  : Next Key Share
|                  |  |  Insert Into Sorted Shared Temp Table  ID = t1
|                  |  |  |  #Columns = 3
|                  |  |  |  #Sort Key Columns = 1
|                  |  |  |  |  Key 1: DEPTNO (Ascending)
|                  |  |  |  |  Use Round-Robin Sort
|                  |  |  |  Sortheap Allocation Parameters:
|                  |  |  |  |  #Rows     = 40
|                  |  |  |  |  Row Width = 48
|                  |  |  |  Piped
|                  |  Sorted Shared Temp Table Completion  ID = t1
|                  |  |  Access Temp Table  ID = t1
|                  |  |  #Columns = 3
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                  |  |  Merge Join
|                  |  |  Access Table Name = DOOLE.PROJECT  ID = 2,7
|                  |  |  |  #Columns = 2
|                  |  |  |  Parallel Scan
|                  |  |  |  |  Relation Scan
|                  |  |  |  |  |  Prefetch: Eligible
|                  |  |  |  Lock Intents
|                  |  |  |  |  |  Table: Intent Share
|                  |  |  |  |  |  Row  : Next Key Share
|                  |  |  |  |  Insert Into Sorted Shared Temp Table  ID = t2
|                  |  |  |  |  #Columns = 2
|                  |  |  |  |  |  #Sort Key Columns = 1
|                  |  |  |  |  |  |  Key 1: DEPTNO (Ascending)
|                  |  |  |  |  Use Replicated Sort
|                  |  |  |  |  |  Sortheap Allocation Parameters:
|                  |  |  |  |  |  #Rows     = 38
|                  |  |  |  |  |  Row Width = 28
|                  |  |  |  |  |  Piped
|                  |  |  |  |  Sorted Shared Temp Table Completion  ID = t2
|                  |  |  |  |  Access Temp Table  ID = t2
|                  |  |  |  |  #Columns = 2
|                  |  |  |  |  |  Relation Scan
|                  |  |  |  |  |  |  Prefetch: Eligible
|                  |  |  Insert Into Sorted Shared Temp Table  ID = t3
|                  |  |  |  #Columns = 5
|                  |  |  |  #Sort Key Columns = 1
|                  |  |  |  |  Key 1: (Ascending)
|                  |  |  |  Use Partitioned Sort
|                  |  |  |  Sortheap Allocation Parameters:
|                  |  |  |  |  #Rows     = 61
|                  |  |  |  |  Row Width = 72
|                  |  |  |  Piped
|                  |  |  |  Access Temp Table  ID = t3
|                  |  |  |  #Columns = 5
|                  |  |  |  Relation Scan
|                  |  |  |  |  Prefetch: Eligible
|                  |  |  |  Merge Join
|                  |  |  |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|                  |  |  |  |  #Columns = 3
|                  |  |  |  |  Parallel Scan
```

```
    |  |  |  Relation Scan
    |  |  |  |  Prefetch: Eligible
    |  |  Lock Intents
    |  |  |  |  Table: Intent Share
    |  |  |  |  Row  : Next Key Share
    |  |  Insert Into Sorted Shared Temp Table  ID = t4
    |  |  |  |  #Columns = 3
    |  |  |  |  #Sort Key Columns = 1
    |  |  |  |  Key 1: WORKDEPT (Ascending)
    |  |  |  |  Use Partitioned Sort
    |  |  |  |  Sortheap Allocation Parameters:
    |  |  |  |  #Rows     = 63
    |  |  |  |  Row Width = 32
    |  |  |  |  Piped
    |  Sorted Shared Temp Table Completion  ID = t4
    |  |  Access Temp Table  ID = t4
    |  |  #Columns = 3
    |  |  |  Relation Scan
    |  |  |  |  Prefetch: Eligible
    |  Insert Into Asynchronous Local Table Queue  ID = q1
Access Local Table Queue  ID = q1  #Columns = 5
   Return Data to Application
    |  #Columns = 5
    |
    End of section


    Optimizer Plan:

                    RETURN
                  (   1)
                    |
                   LTQ
                  (   2)
                    |
    MSJOIN
                  (   3)
                 /      \
          TBSCAN          TBSCAN
          (   4)          (  15)
            |               |
           SORT            SORT
          (   5)          (  16)
            |               |
          MSJOIN          TBSCAN
          (   6)          (  17)
         /       \          |
    TBSCAN     TBSCAN   Table:
    (   7)     (  11)   DOOLE
      |          |      EMPLOYEE
     SORT             SORT
    (   8)     (  12)
      |          |
     TBSCAN           TBSCAN
    (   9)     (  13)
```

```
     |         |
 Table:      Table:
 DOOLE       DOOLE
    DEPARTMENT  PROJECT
```

此方案几乎与第一个示例中的方案完全一样。主要区别是当该方案第一次启动时要创建四个子代理程序，而在该方案结束时表队列要收集每个子代理程序的工作结果，然后将这些结果返回至应用程序。

还要注意在与 EMPLOYEE 连接之前需要进行额外的排序。需要执行此操作，因为处理 DEPARTMENT 和 PROJECT 之间的合并连接的子代理程序可能会产生顺序紊乱的连接行。

示例三: 具有分区间并行性的多分区数据库方案

本示例显示与第500页的『示例一: 无并行性方案』相同的 SQL 语句，但此查询已在由三个数据库分区组成的分区数据库上进行了编译。

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:21:29

Bind Timestamp = 2000-01-03-15.21.29.990983

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno


Estimated Cost        = 118
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
```

```
|                  |  |  Nodes = 13, 82, 193
|                  Distribute Subsection #3
|                  |  Broadcast to Node List
|                  |  |  Nodes = 13, 82, 193
|                  Distribute Subsection #1
|                  |  Broadcast to Node List
|                  |  |  Nodes = 13, 82, 193
|                  Access Table Queue  ID = q1  #Columns = 5
|                  Return Data to Application
|                  |  #Columns = 5
|
|               Subsection #1:
|                  Access Table Queue  ID = q2  #Columns = 3
|                  |  Output Sorted
|                  |  |  #Key Columns = 1
|                  |  |  |  Key 1: (Ascending)
|                  Merge Join
|                  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
|                  |  |  #Columns = 3
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                  |  |  Lock Intents
|                  |  |  |  Table: Intent Share
|                  |  |  |  Row  : Next Key Share
|                  |  |  Insert Into Sorted Temp Table  ID = t1
|                  |  |  |  #Columns = 3
|                  |  |  |  #Sort Key Columns = 1
|                  |  |  |  |  Key 1: DEPTNO (Ascending)
|                  |  |  |  Sortheap Allocation Parameters:
|                  |  |  |  |  #Rows     = 40
|                  |  |  |  |  Row Width = 48
|                  |  |  |  Piped
|                  |  Sorted Temp Table Completion  ID = t1
|                  |  Access Temp Table  ID = t1
|                  |  |  #Columns = 3
|                  |  |  Relation Scan
|                  |  |  |  Prefetch: Eligible
|                  Merge Join
|                  |  Access Table Queue  ID = q3  #Columns = 2
|                  |  |  Output Sorted
|                  |  |  |  #Key Columns = 1
|                  |  |  |  |  Key 1: (Ascending)
|                  Insert Into Asynchronous Table Queue  ID = q1
|                  |  Broadcast to Coordinator Node
|                  |  Rows Can Overflow to Temporary Table
|
|               Subsection #2:
|                  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
|                  |  #Columns = 3
|                  |  Relation Scan
|                  |  |  Prefetch: Eligible
|                  |  Lock Intents
|                  |  |  Table: Intent Share
|                  |  |  Row  : Next Key Share
|                  |  Insert Into Sorted Temp Table  ID = t2
```

```
|                    |  |  #Columns = 3
|                    |  |  #Sort Key Columns = 1
|                    |  |  |  Key 1: WORKDEPT (Ascending)
|                    |  |  Sortheap Allocation Parameters:
|                    |  |  |  #Rows     = 27
|                    |  |  |  Row Width = 32
|                    |  |  Piped
|                    Sorted Temp Table Completion  ID = t2
|                    Access Temp Table  ID = t2
|                    |  #Columns = 3
|                    |  Relation Scan
|                    |  |  Prefetch: Eligible
|                    |  Insert Into Asynchronous Table Queue  ID = q2
|                    |  |  Hash to Specific Node
|                    |  |  Rows Can Overflow to Temporary Tables
|                    Insert Into Asynchronous Table Queue Completion  ID = q2
|
|                 Subsection #3:
|                    Access Table Name = DOOLE.PROJECT  ID = 2,7
|                    |  #Columns = 2
|                    |  Relation Scan
|                    |  |  Prefetch: Eligible
|                    |  Lock Intents
|                    |  |  Table: Intent Share
|                    |  |  Row  : Next Key Share
|                    |  Insert Into Sorted Temp Table  ID = t3
|                    |  |  #Columns = 2
|                    |  |  #Sort Key Columns = 1
|                    |  |  |  Key 1: DEPTNO (Ascending)
|                    |  |  Sortheap Allocation Parameters:
|                    |  |  |  #Rows     = 38
|                    |  |  |  Row Width = 28
|                    |  |  Piped
|                    Sorted Temp Table Completion  ID = t3
|                    Access Temp Table  ID = t3
|                    |  #Columns = 2
|                    |  Relation Scan
|                    |  |  Prefetch: Eligible
|                    |  Insert Into Asynchronous Table Queue  ID = q3
|                    |  |  Hash to Specific Node
|                    |  |  Rows Can Overflow to Temporary Tables
|                    Insert Into Asynchronous Table Queue Completion  ID = q3
|
|                 End of section
|
|
|                 Optimizer Plan:
|
|                                RETURN
|                              (   1)
|                                |
|                               BTQ
|                               (   2)
|                                |
|                 MSJOIN
```

```
                ( 3)
               /    \
          MSJOIN     MDTQ
          ( 4)       ( 14)
         /    \        |
      MDTQ    TBSCAN   TBSCAN
      ( 5)    ( 10)    ( 15)
        |       |        |
      TBSCAN    SORT     SORT
      ( 6)    ( 11)    ( 16)
        |       |        |
       SORT    TBSCAN   TBSCAN
      ( 7)    ( 12)    ( 17)
        |       |        |
      TBSCAN  Table:   Table:
      ( 8)    DOOLE    DOOLE
        |     DEPARTMENT PROJECT
     Table:
     DOOLE
     EMPLOYEE
```

此方案与第一个示例中的方案有完全相同的细节，但是该程序段分成四个子段。这些子段具有下列任务:

- **协调程序子段**。此子段协调其他子段。在此方案中，它导致分发其他子段，然后使用表队列来收集要返回至应用程序的结果。
- **子段 #1**。此子段扫描表队列 q2，并用合并连接将它与 DEPARTMENT 表连接。接着第二个合并连接将表队列 q3 中的数据添加进来。然后使用表队列 q1 将连接的行发送到协调程序子段。
- **子段 #2**。此子段扫描 EMPLOYEE 表，对它进行排序，然后将结果散列到特定节点。这些结果由子段 1 读取。
- **子段 #3**。此子段扫描 PROJECT 表，对它进行排序，然后将结果散列到特定节点。这些结果由子段 1 读取。

示例四: 具有分区间和分区内并行性的多分区数据库方案

本示例显示与第500页的『示例一: 无并行性方案』相同的 SQL 语句，但此查询已在由三个数据库分区组成的分区数据库上进行了编译，这些数据库分区的每一个都在四路 SMP 机器上。

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 15:22:14

Bind Timestamp = 2000-01-03-15.22.14.659970

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
```

```
Query Optimization Class = 5

Partition Parallel       = Yes
Intra-Partition Parallel = Yes (Bind Degree = 4)

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION -------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 140
Estimated Cardinality = 263

Coordinator Subsection:
   Distribute Subsection #2
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #3
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Distribute Subsection #1
   |  Broadcast to Node List
   |  |  Nodes = 13, 82, 193
   Access Table Queue  ID = q1  #Columns = 5
   Return Data to Application
   |  #Columns = 5

Subsection #1:
   Process Using 4 Subagents
   |  Access Table Queue  ID = q3  #Columns = 3
   |  Insert Into Sorted Shared Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  #Sort Key Columns = 1
   |  |  |  Key 1: (Ascending)
   |  |  Use Partitioned Sort
   |  |  Sortheap Allocation Parameters:
   |  |  |  #Rows     = 27
   |  |  |  Row Width = 32
   |  |  Piped
   |  Access Temp Table  ID = t1
   |  |  #Columns = 3
   |  |  Relation Scan
   |  |  |  Prefetch: Eligible
   |  Merge Join
   |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
```

```
|                    |  |  |  #Columns = 3
|                    |  |  |  Parallel Scan
|                    |  |  |  Relation Scan
|                    |  |  |  |  Prefetch: Eligible
|                    |  |  |  Lock Intents
|                    |  |  |  |  Table: Intent Share
|                    |  |  |  |  Row  : Next Key Share
|                    |  |  |  Insert Into Sorted Shared Temp Table  ID = t2
|                    |  |  |  |  #Columns = 3
|                    |  |  |  |  #Sort Key Columns = 1
|                    |  |  |  |  |  Key 1: DEPTNO (Ascending)
|                    |  |  |  |  Use Partitioned Sort
|                    |  |  |  |  Sortheap Allocation Parameters:
|                    |  |  |  |  |  #Rows     = 40
|                    |  |  |  |  |  Row Width = 48
|                    |  |  |  |  Piped
|                    |  |  |  Sorted Shared Temp Table Completion  ID = t2
|                    |  |  |  Access Temp Table  ID = t2
|                    |  |  |  |  #Columns = 3
|                    |  |  |  |  Relation Scan
|                    |  |  |  |  |  Prefetch: Eligible
|                    |  |  Insert Into Sorted Shared Temp Table  ID = t3
|                    |  |  |  #Columns = 6
|                    |  |  |  #Sort Key Columns = 1
|                    |  |  |  |  Key 1: (Ascending)
|                    |  |  |  Use Partitioned Sort
|                    |  |  |  Sortheap Allocation Parameters:
|                    |  |  |  |  #Rows     = 44
|                    |  |  |  |  Row Width = 76
|                    |  |  |  Piped
|                    |  |  Access Temp Table  ID = t3
|                    |  |  |  #Columns = 6
|                    |  |  |  Relation Scan
|                    |  |  |  |  Prefetch: Eligible
|                    |  |  Merge Join
|                    |  |  |  Access Table Queue  ID = q5  #Columns = 2
|                    |  |  |  Insert Into Sorted Shared Temp Table  ID = t4
|                    |  |  |  |  #Columns = 2
|                    |  |  |  |  #Sort Key Columns = 1
|                    |  |  |  |  |  Key 1: (Ascending)
|                    |  |  |  |  Use Partitioned Sort
|                    |  |  |  |  Sortheap Allocation Parameters:
|                    |  |  |  |  |  #Rows     = 38
|                    |  |  |  |  |  Row Width = 28
|                    |  |  |  |  Piped
|                    |  |  |  Access Temp Table  ID = t4
|                    |  |  |  |  #Columns = 2
|                    |  |  |  |  Relation Scan
|                    |  |  |  |  |  Prefetch: Eligible
|                    |  |  Insert Into Asynchronous Local Table Queue  ID = q2
|                    Access Local Table Queue  ID = q2  #Columns = 5
|                    Insert Into Asynchronous Table Queue  ID = q1
|                    |  Broadcast to Coordinator Node
|                    |  Rows Can Overflow to Temporary Table
|
```

```
Subsection #2:
   Process Using 4 Subagents
      Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
      |  #Columns = 3
      |  Parallel Scan
      |  Relation Scan
      |  |  Prefetch: Eligible
      |  Lock Intents
      |  |  Table: Intent Share
      |  |  Row  : Next Key Share
      |  Insert Into Sorted Shared Temp Table  ID = t5
      |  |  #Columns = 3
      |  |  #Sort Key Columns = 1
      |  |  |  Key 1: WORKDEPT (Ascending)
      |  |  Use Round-Robin Sort
      |  |  Sortheap Allocation Parameters:
      |  |  |  #Rows     = 27
      |  |  |  Row Width = 32
      |  |  Piped
      Sorted Shared Temp Table Completion  ID = t5
      Access Temp Table  ID = t5
      |  #Columns = 3
      |  Relation Scan
      |  |  Prefetch: Eligible
      |  Insert Into Asynchronous Local Table Queue  ID = q4
      Access Local Table Queue  ID = q4  #Columns = 3
      Insert Into Asynchronous Table Queue  ID = q3
      |  Hash to Specific Node
      |  Rows Can Overflow to Temporary Tables
      |
Subsection #3:
   Process Using 4 Subagents
      Access Table Name = DOOLE.PROJECT  ID = 2,7
      |  #Columns = 2
      |  Parallel Scan
      |  Relation Scan
      |  |  Prefetch: Eligible
      |  Lock Intents
      |  |  Table: Intent Share
      |  |  Row  : Next Key Share
      |  Insert Into Sorted Shared Temp Table  ID = t6
      |  |  #Columns = 2
      |  |  #Sort Key Columns = 1
      |  |  |  Key 1: DEPTNO (Ascending)
      |  |  Use Round-Robin Sort
      |  |  Sortheap Allocation Parameters:
      |  |  |  #Rows     = 38
      |  |  |  Row Width = 28
      |  |  Piped
      Sorted Shared Temp Table Completion  ID = t6
      Access Temp Table  ID = t6
      |  #Columns = 2
      |  Relation Scan
      |  |  Prefetch: Eligible
      |  Insert Into Asynchronous Local Table Queue  ID = q6
```

```
   Access Local Table Queue  ID = q6  #Columns = 2
   Insert Into Asynchronous Table Queue  ID = q5
   |  Hash to Specific Node
   |  Rows Can Overflow to Temporary Tables

End of section


Optimizer Plan:

                RETURN
              (   1)
                |
               BTQ
              (   2)
                |
               LTQ
              (   3)
                |
MSJOIN
              (   4)
             /      \
       TBSCAN         TBSCAN
       (   5)         (  20)
         |              |
        SORT           SORT
       (   6)         (  21)
         |              |
       MSJOIN          DTQ
       (   7)         (  22)
      /      \          |
  TBSCAN    TBSCAN     LTQ
  (   8)    (  16)    (  23)
    |         |         |
   SORT       SORT     TBSCAN
  (   9)    (  17)    (  24)
    |         |         |
   DTQ       TBSCAN     SORT
  (  10)    (  18)    (  25)
    |         |         |
   LTQ     Table:      TBSCAN
  (  11)   DOOLE       (  26)
    |      DEPARTMENT    |
  TBSCAN                Table:
  (  12)                DOOLE
    |                   PROJECT
SORT
  (  13)
    |
TBSCAN
  (  14)
    |
 Table:
 DOOLE
 EMPLOYEE
```

此方案与第506页的『示例三：具有分区间并行性的多分区数据库方案』中的方案相似，不同的是多个子代理程序执行每个子段。另外，在每个子段结束时，在将合格行插入要散列到特定节点的第二个表队列之前，本地表队列将收集所有子代理程序的结果。

示例五：联合体数据库方案

本示例显示与第500页的『示例一：无并行性方案』相同的 SQL 语句，但此查询已在一个联合体数据库上编译，在这个联合体数据库中，表 DEPARTMENT 和 PROJECT 在数据源上，而表 EMPLOYEE 在联合体服务器上。

```
******************** PACKAGE ****************************************

Package Name = DOOLE.DYNEXPLN
Prep Date = 2000/01/03
Prep Time = 16:29:01

Bind Timestamp = 2000-01-03-16.29.01.479230

Isolation Level          = Cursor Stability
Blocking                 = Block Unambiguous Cursors
Query Optimization Class = 5

Partition Parallel       = No
Intra-Partition Parallel = No

Function Path            = "SYSIBM", "SYSFUN", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:

  SELECT x.lastname, x.job, y.deptname, y.location, z.projname
  FROM employee AS x, department AS y, project AS z
  WHERE x.workdept = y.deptno AND x.workdept = z.deptno AND y.deptno
          = z.deptno

Estimated Cost        = 1954
Estimated Cardinality = 100800

Distribute Subquery #2
   |  #Columns = 3
   Insert Into Sorted Shared Temp Table  ID = t1
   |  #Columns = 3
   |  #Sort Key Columns = 1
   |  |  Key 1: Remote Query #2, Output Column 1 (Ascending)
   |  Sortheap Allocation Parameters:
   |  |  #Rows     = 1000
   |  |  Row Width = 56
   |  Piped
Access Temp Table  ID = t1
```

```
                    |  #Columns = 3
                    |  Relation Scan
                    |  |  Prefetch: Eligible
                    Merge Join
                    |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,5
                    |  #Columns = 3
                    |  Relation Scan
                    |  |  Prefetch: Eligible
                    |  |  Lock Intents
                    |  |  |  Table: Intent Share
                    |  |  |  Row  : Next Key Share
                |   |  Insert Into Sorted Temp Table  ID = t2
                    |  |  |  #Columns = 3
                    |  |  |  #Sort Key Columns = 1
                    |  |  |  |  Key 1: WORKDEPT (Ascending)
                    |  |  |  Sortheap Allocation Parameters:
                |   |  |  |  #Rows     = 63
                    |  |  |  |  Row Width = 32
                    |  |  |  Piped
                |   Sorted Temp Table Completion  ID = t2
                |   Access Temp Table  ID = t2
                    |  |  #Columns = 3
                    |  |  Relation Scan
                    |  |  |  Prefetch: Eligible
                    Merge Join
                    |  Distribute Subquery #1
                    |  |  #Columns = 2
                    |  Insert Into Sorted Temp Table  ID = t3
                    |  |  #Columns = 2
                    |  |  |  Key 1: Remote Query #1, Output Column 1 (Ascending)
                    |  |  Sortheap Allocation Parameters:
                    |  |  |  #Rows     = 1000
                    |  |  |  Row Width = 36
                    |  |  Piped
                    |  Access Temp Table  ID = t3
                    |  |  #Columns = 2
                    |  |  Relation Scan
                    |  |  |  Prefetch: Eligible
                    Return Data to Application
                    |  #Columns = 5
                    |
                    Distributed Subquery #1:
                    Server: REMOTE_SAMPLE  (DB2/CS 7.1)
                    Subquery SQL Statement:
                    |
                    |    SELECT A0."DEPTNO", A0."PROJNAME"
                    |    FROM "DOOLE"."PROJECT" A0
                    |
                    Nicknames Referenced:
                    |  REMOTE.PROJECT  ID = 7  Base = DOOLE.PROJECT
                    #Output Columns = 2
                    |
                    Distributed Subquery #2:
                    Server: REMOTE_SAMPLE  (DB2/CS 7.1)
                    Subquery SQL Statement:
```

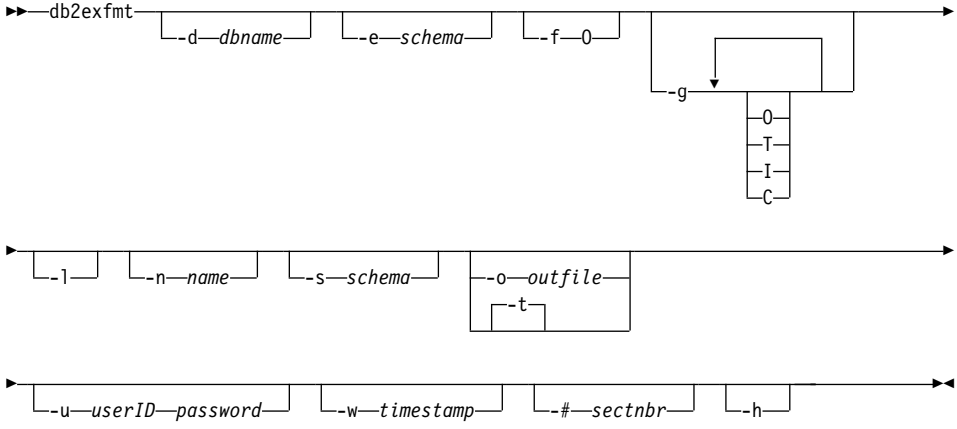
```
        SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
        FROM "DOOLE"."DEPARTMENT" A0

   Nicknames Referenced:
      REMOTE.DEPARTMENT  ID = 4  Base = DOOLE.DEPARTMENT
   #Output Columns = 3

End of section


Optimizer Plan:

                RETURN
              (   1)
                 |
MSJOIN
              (   2)
              /       \
        MSJOIN        TBSCAN
       (   3)           (  13)
      /      \            |
   TBSCAN    TBSCAN     SORT
  (   4)      (   9)    (  14)
    |           |         |
   SORT        SORT     DSBQRY
  (   5)      (  10)    (  15)
    |           |         |
  DSBQRY      TBSCAN   Nickname:
  (   6)      (  11)   REMOTE
    |           |      PROJECT
 Nickname:    Table:
 REMOTE       DOOLE
 DEPARTMENT   EMPLOYEE
```

此方案与第一个示例中的方案有完全相同的细节，但有两个表的数据来自数据源。这两个表是通过分布式子查询存取的，在本案例中，这些分布式子查询只是从那些表中选择所有行。在将数据返回至联合体服务器之后，它便与来自本地表的数据连接到一起。

附录D. db2exfmt - 解释表格式的工具

可以使用 db2exfmt 工具来格式化解释表的内容。此工具位于实例 sqllib 目录的 misc 子目录中。

要使用该工具，需要有要格式化的解释表的读存取权。



-d dbname

包含程序包的数据库的名称。

-e schema

解释表的模式。

-f 格式化标志。在此发行版中，唯一支持的值是0（运算符摘要）。

-g 图形方案。如果仅指定 -g，则生成一个图，后跟所有表的格式化信息。另外，可以指定下列有效值的任意组合:

O 只生成图形。不格式化表的内容。

T 在图形中每个运算符下包括总成本。

I 在图形中每个运算符下包括 I/O 成本。

C 在图形中包括每个运算符的期望输出基数（元组数）。

-l 当处理程序包名时区分大小写。

-n name

解释请求的源名 (SOURCE_NAME)。

-s schema

解释请求的源的模式或限定符 (SOURCE_SCHEMA)。

-o outfile
输出文件名。

-t 将输出定向到终端。

-u user ID password
当与一个数据库连接时，使用提供的用户 ID 和口令。

用户 ID 和口令都必须符合命名约定，且是数据库可识别的。

-w timestamp
解释时间戳记。指定 -1 来获取最新的解释请求。

-# sectnbr
源中的段号。要请求所有的段，指定零。

-h 显示帮助信息。当指定了此选项时，其他所有的选项都会被忽略，且只显示帮助信息。

除 -h 和 -l 选项外，将提示您输入任何未提供或未完整指定的参数值。

如果未提供解释表模式，则环境变量 **USER** 的值就被用作缺省值。如果找不到此变量，则会提示用户提供解释表的模式。

源名、源模式和解释时间戳记可用 LIKE 谓词格式提供，它允许百分比符号 (%) 和下划线字符 (_) 用作模式匹配字符，以通过一个调用选择多个源。对于最新的解释语句，可将解释时间指定为 -1。

如果指定 -o 而不带文件名，且未指定 -t，则会提示用户提供文件名（缺省文件名为 db2exfmt.out）。如果既未指定 -o 也未指定 -t，则会提示用户提供文件名（缺省选项是终端输出）。如果同时指定 -o 和 -t，则将输出定向到终端。

附录E. 配置 XA 事务管理程序以使用 DB2 UDB

下面几节描述如何配置特定的产品，以将 DB2 用作资源管理程序。可使用下列任何一项:

- 『配置 IBM TXSeries CICS』
- 『配置 IBM TXSeries Encina』
- 第522页的『配置 BEA Tuxedo』
- 第523页的『配置 Microsoft 事务服务器』。

配置 IBM TXSeries CICS

有关如何配置 IBM TXSeries CICS 以将 DB2 用作资源管理程序的详情，参考 *IBM TXSeries CICS Administration Guide*。TXSeries 文档可以通过访问如下站点来联机查看:
http://www.transarc.com/dfs/public/www/htdocs/.hosts/external/Library/index.html

主机和 AS/400 数据库服务器可以参与 CICS 协调的事务。

配置 IBM TXSeries Encina

以下是当通过 DB2 Connect 进行存取时，将 “Encina 监控程序” 与 “DB2 通用数据库” 服务器或 DB2 MVS 版、DB2 OS/390 版、DB2 AS/400 版或 DB2 VSE 版和 VM 版集成，所需的各种 API 和配置参数。TXSeries 文档可以通过访问如下站点来联机查看:
http://www.transarc.com/dfs/public/www/htdocs/.hosts/external/Library/index.html

主机和 AS/400 数据库服务器可以参与 Encina 协调的事务。

配置 DB2

要配置 DB2:

1. 必须在 DB2 数据库目录中定义每个数据库的名称。若该数据库是一个远程数据库，则还必须定义 “节点目录” 项。您可以使用 GUI “客户机配置辅助程序” (CCA) 或 DB2 “命令行处理器” (CLP) 来执行该配置。例如:

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER

   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. 若 DB2 客户机知道它处理的是 Encina，则它可以针对 Encina 优化它的内部处理。可以将 *tp_mon_name* 数据库管理程序配置参数设置为 ENCINA 来指定此一点。缺省值表示不进行特殊的优化。若设置了 *tp_mon_name*，则应用程序必须确保执行工作单元的线程在结束工作之后也立即落实该工作。不能启动任何其他工作单元。若您的环境不是这样的，则要确保 *tp_mon_name* 的值是 NONE（或者，通过 CLP，将该值设置为 NULL）。*tp_mon_name* 可通过调用 CCA 进行更新，也可以由 CLP 更新:
   - 在 AIX 上，使用: `UPDATE DATABASE MANAGER CONFIGURATION USING TP_MON_NAME ENCINA`
   - 在 Windows NT 上，使用: `UPDATE DATABASE MANAGER CONFIGURATION USING TP_MON_NAME libEncServer:E`

     在 Intel 环境中，此参数包含在一个外部事务管理程序产品中具有函数 ax_reg 和 ax_unreg 的 DLL 的路径和名称，并通知 DB2 正在使用哪个 “TP 监控程序”。

为每个资源管理程序配置 Encina

要为每个资源管理程序配置 Encina，管理员必须为作为资源管理程序的每个 DB2 数据库定义 “打开字符串”、“关闭字符串” 和 “控制线程协议”，然后才能为应用程序中的事务注册该资源管理程序。可以使用 Enconcole 全屏幕界面或 Encina 命令行界面来执行该配置。例如:

```
monadmin create rm inventdb  -open "inventdb,user1,password1"
```

每个 DB2 数据库都有一个资源管理程序配置，且每个资源管理程序 (RM) 配置必须具有 rm 名称（“逻辑 RM 名称”）。为了简化情况，您应该使它与数据库名相同。

“XA 打开字符串” 包含建立与数据库的连接所需的信息。该字符串的内容是 RM 所独有的。 DB2 UDB 的 “XA 打开字符串” 包含要打开的数据库的别名，以及与该连接相关的可选的用户 ID 和口令。注意，此处定义的数据库名也必须在所有数据库存取都必需的正规数据库目录中编目。该名称的长度可以多达 8 个字节。

DB2 不使用 “XA 关闭字符串”。

“控制线程协议” 确定一个应用程序的代理程序线程是否一次可以处理多个事务。DB2 V5.0 和后续版本支持 TMXA_SERIALIZE_ALL_OPERATIONS 的缺省值，并且只有当一个事务完成之后才能再次使用一个线程。

若您正存取 DB2 OS/390 版、DB2 MVS 版、DB2 AS/400 版或 DB2 VSE 版和 VM 版，则必须使用 DB2 同步点管理程序。有关配置指示，请参考 *DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings* 手册。

从 Encina 应用程序中引用 DB2 数据库

要从 Encina 应用程序中引用 DB2 数据库:

1. 使用 “Encina 调度策略 API”，来指定可以从单个 “TP 监控程序” 的应用程序进程中运行多少个应用程序的代理程序。例如:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

   对于 DB2 （DB2 通用数据库、主机或 AS/400 数据库服务器），您应该使用 MON_EXCLUSIVE 的缺省设置。这确保:

   - 应用程序进程在事务的寿命期间被锁定。
   - 应用程序使用单线程。

   注: 若您使用的是 ODBC 或 DB2 调用层接口，则必须禁用多线程支持。为此，您可以设置 CLI 配置参数 DISABLEMULTITHREAD = 1（禁用多线程）。DB2 通用数据库的缺省值为 DISABLEMULTITHREAD = 0（启用多线程）。有关详情，参考 *CLI Guide and Reference*。

2. 使用 “Encina RM 注册 API” 来提供 XA 开关和逻辑 RM 名称，以供 Encina 在应用程序进程中引用 RM 时使用。例如:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch, /* xa switch */
                          "inventdb", /* logical RM name */
                         &rmiId  ); /* internal RM id */
   ```

   XA 开关包含 RM 中 TM 可以调用的 XA 例程的地址，它还指定由 RM 提供的功能。DB2 通用数据库的 XA 开关是 db2xa_switch，且它驻留在 DB2 客户机库（在 INTEL 平台上为 db2app.dll，在基于 UNIX 的平台上为 libdb2）中。

   逻辑 RM 名是 Encina 使用的那个名称，而不是在 Encina 下运行的 SQL 应用程序使用的实际数据库名。实际数据库名是在 “Encina RM 注册表” 的 “XA 打开字符串” 中指定的。为了简化情况，将逻辑 RM 名设置为与本示例中的数据库名相同。

   第三个参数返回 TM 引用此连接所用的内部标识符或句柄。

注: 当通过 TM-XA 接口使用 Encina 来进行 DB2 的事务处理时，注意 Encina 嵌套的事务当前不受 DB2 XA 接口的支持。可能的话，应避免使用这些事务。若无法避免，则确保仅在 Encina 事务系列的一个成员中执行 SQL 工作。

配置 BEA Tuxedo

注: 在 Tuxedo 环境中存取主机或 AS/400 数据库服务器的应用程序只能对这些服务器进行只读存取。

要配置 Tuxedo 以将 DB2 用作资源管理程序，须执行下列步骤:

1. 按该产品的文档中指定的步骤，安装 Tuxedo。确保执行所有基本的 Tuxedo 配置，包括日志文件和环境变量。

   您还需要一个编译程序和 DB2 应用程序开发客户机。需要时安装它们。

2. 在 Tuxedo 服务器 ID 中，设置 DB2INSTANCE 环境变量，以引用包含您希望 Tuxedo 使用的数据库的实例。再设置 PATH 变量以包括 DB2 程序目录。然后确认 Tuxedo 服务器 ID 可以与 DB2 数据库连接。

3. 仅适用于 Windows NT。使用包含 `ax_reg` 和 `ax_unreg` 例程的 DLL 的名称来更新 *tp_mon_name* 数据库管理程序配置参数。在 Tuxedo 中，此 DLL 称为 `libtux`。

4. 将 DB2 的定义添加至 Tuxedo 资源管理程序的定义文件。在下面的示例中，`UDB_XA` 是为 DB2 在本地定义的 Tuxedo 资源管理程序名，而 `db2xa_switch` 是类型为 `xa_switch_t` 的一个结构的 DB2 定义的名称。

   - 对于 AIX。在文件 `${TUXDIR}/udataobj/RM` 中，添加定义:

     ```
     # DB2 UDB
     UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
     ```

     其中 `{TUXDIR}` 是安装了 Tuxedo 的目录，而 `{DB2DIR}` 是 DB2 实例的目录。

   - 对于 Windows NT。在文件 `%TUXDIR%\udataobj\rm` 中，添加定义:

     ```
     # DB2 UDB
     UDB_XA;db2xa_switch;%DB2DIR%\lib\db2api.lib
     ```

     其中 `%TUXDIR%` 是安装了 Tuxedo 的目录，而 `%DB2DIR%` 是 DB2 实例的目录。

5. 为 DB2 构建 Tuxedo 事务监控程序服务器程序:

   - 对于 AIX:

     ```
     ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
     ```

     其中 `{TUXDIR}` 是安装了 Tuxedo 的目录。

   - 对于 Windows NT:

     ```
     %TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
     ```

6. 构建应用服务器。在下面的示例中，-r 选项指定资源管理程序名，-f 选项（使用了一次或多次）指定包含应用程序服务的文件，-s 选项指定此服务器的应用程序服务名，而 -o 选项指定输出服务器文件名:

   - 对于 AIX:

     ```
     ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

     其中 {TUXDIR} 是安装了 Tuxedo 的目录。

   - 对于 Windows NT:

     ```
     %TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

     其中 %TUXDIR% 是安装了 Tuxedo 的目录。

7. 设置 Tuxedo 配置文件以引用 DB2 服务器。在 UDBCONFIG 文件的 *GROUPS 小节中，添加类似如下的项:

   ```
   UDB_GRP   LMID=simp GRPNO=3
     TMSNAME=TMS_UDB TMSCOUNT=2
     OPENINFO="UDB_XA:SAMPLE,db2_user,,db2_user_pwd"
   ```

   其中 TMSNAME 参数指定您先前构建的事务监控程序的服务器程序，而 OPENINFO 参数指定资源管理程序名。其后是数据库名和 DB2 用户和口令（用于认证）。

   您先前构建的应用服务器在 Tuxedo 配置文件的 *SERVERS 节中被引用。

8. 启动 Tuxedo:

   ```
   tmboot -y
   ```

   在该命令完成之后， Tuxedo 信息应指示服务器已启动。另外，若您发出 DB2 命令 LIST APPLICATIONS ALL，您应该看到两个连接（在这种情况下，由 Tuxedo 配置文件 UDBCONFIG 中的 UDB 组的 TMSCOUNT 参数指定）。

配置 Microsoft 事务服务器

DB2 UDB V5.2 和及后续版本可以完全与 Microsoft 事务服务器 (MTS) 版本 2.0 集成在一起。在 Windows 32 位操作系统上的 MTS 下运行的应用程序，可以使用 MTS 来协调多个 DB2 UDB、主机和 AS/400 数据库服务器以及其他 MTS 兼容的资源管理程序的两阶段落实。

在 DB2 中启用 MTS 支持

自动启用 “Microsoft 事务服务器” 支持。可将 *tp_mon_name* 数据库管理程序配置参数设置为 『MTS』，但这没有必要，而且将忽略它。

注: IBM web 站点上可能提供了其他技术信息，以帮助您安装和配置 DB2 MTS 支持。将 URL 设置为 http://www.ibm.com/software/data/db2/library/，并使用关键字 ″MTS″ 来搜索 DB2 通用数据库 “技术注释”。

MTS 软件的先决条件

MTS 支持需要 DB2 客户机应用程序使能器 (CAE) 版本 5.2 或更高版本，且 MTS 必须是带有最新修订 0772 的版本 2.0 或更高版本。

在 Windows 32 位操作系统上安装 DB2 ODBC 驱动程序将自动把一个新关键字添加至该注册表:

```
HKEY_LOCAL_MACHINE\software\ODBC\odbcinit.ini\IBM DB2 ODBC Driver:
关键字值名: CPTimeout
数据类型: REG_SZ
值: 60
```

安装和配置

以下是 MTS 的安装和配置考虑事项的摘要。要使用 DB2 的 MTS 支持，用户必须:

1. 在运行 MTS 应用程序的机器上安装 MTS 和 DB2 客户机。
2. 若主机或 AS/400 数据库服务器要参与多站点更新:
   a. 在本地机器或远程机器上安装 “DB2 Connect 企业版”。 “DB2 Connect 企业版” 允许主机或 AS/400 数据库服务器参与多站点更新事务。
   b. 确保允许 “DB2 Connect 企业版” 参与多站点更新。有关允许 DB2 Connect 参与多站点更新的信息，请参阅该平台的 “DB2 Connect 企业版快速入门” 手册。

当运行 DB2 CLI/ODBC 应用程序时，下列配置关键字（与 `db2cli.ini` 文件中设置的相同）必须为缺省值，不得更改:

- CONNECTYPE 关键字（缺省值为 1）
- MULTICONNECT 关键字（缺省值为 1）
- DISABLEMULTITHREAD 关键字（缺省值为 1）
- CONNECTIONPOOLING 关键字（缺省值为 0）
- KEEPCONNECTION 关键字（缺省值为 0）

为利用 MTS 支持而编写的 DB2 CLI 应用程序不能更改与上述关键字对应的属性值。而且，应用程序不能更改下列属性的缺省值:

- SQL_ATTR_CONNECT_TYPE 属性（缺省值为 SQL_CONCURRENT_TRANS）
- SQL_ATTR_CONNECTON_POOLING 属性（缺省值为 SQL_CP_OFF）

注: IBM web 站点上可能提供了其他技术信息，以帮助您安装和配置 DB2 MTS 支持。将 URL 设置为 http://www.ibm.com/software/data/db2/library/，并使用关键字 ″MTS″ 来搜索 DB2 通用数据库 “技术注释”。

验证安装

1. 配置 DB2 客户机和 DB2 Connect EE 以存取您的 DB2 UDB、主机或 AS/400 服务器。
2. 验证从 DB2 CAE 机器至 DB2 UDB 数据库服务器的连接。
3. 使用 DB2 CLP 来验证从 DB2 Connect 机器至您的主机或 AS/400 数据库服务器的连接，并发出一些查询。
4. 验证从 DB2 CAE 机器经由 DB2 Connect 网关至您的主机或 AS/400 数据库服务器的连接，并发出一些查询。

受支持的 DB2 数据库服务器

支持下列服务器使用 MTS 协调的事务来进行多站点更新:

- DB2 通用数据库企业版的版本 5.2
- DB2 扩充企业版版本 5.2
- DB2 OS/390 版
- DB2 MVS 版
- DB2 AS/400 版
- DB2 VM 版和 VSE 版
- DB2 公共服务器 SCO 版的版本 2
- 带 PTF U453782 的 DB2 通用数据库 AIX 版
- 带 PTF U453784 的 DB2 通用数据库 HP-UX 版
- 带 PTF WR09033 的 DB2 通用数据库企业版 OS/2 版
- 带 PTF U453783 的 DB2 通用数据库 SOLARIS 版
- 带 PTF WR09034 的 DB2 通用数据库企业版 Windows NT 版
- DB2 通用数据库扩充企业版 UNIX 版或 Windows NT 版。

MTS 事务超时和 DB2 连接特性

您可以在 MTS Explorer 工具中设置事务超时值。有关详情，请参考联机的 *MTS Administrator Guide*。

若一个事务占用的时间超过该事务超时值（缺省值为 60 秒），则 MTS 将以异步方式对所有参与的“资源管理程序”发出异常终止，然后将整个事务异常终止。

对于与 DB2 服务器的连接，该异常终止会被转换成 DB2 回滚请求。象任何其他数据库请求一样，将在该连接上连续执行回滚请求，以保证数据库服务器上的数据的完整性。

其结果为:

- 若该连接是空闲的，则立即执行回滚。
- 若正在执行一个运行时间较长的 SQL 语句，则会等到该 SQL 语句完成后，再执行回滚请求。

连接缓冲

连接缓冲允许一个应用程序使用一个连接缓冲池中的连接，而不必在每次使用时重新建立连接。一旦创建了一个连接并将其置于缓冲池中，则应用程序可以再次使用该连接，而不用执行完整的连接过程。当应用程序与 ODBC 数据源断开时将缓冲该连接，并将它给予一个属性相同的新连接。

连接缓冲一直是“ODBC 驱动程序管理器 2.x”的一个功能部件。在与 MTS 一起交付的最新 ODBC 驱动程序管理器（版本 3.5）中，对于事务性 MTS COM 对象的 ODBC 连接，连接缓冲有一些配置变化和新的特性（参见第527页的『在参与同一个事务的 COM 对象之间再次使用 ODBC 连接』）。

“ODBC 驱动程序管理器 3.5”要求 ODBC 驱动程序在注册表中注册一个新关键字，才允许将连接缓冲激活。该关键字为:

```
关键字名: SOFTWARE\ODBC\ODBCINST.INI\IBM DB2 ODBC DRIVER
名称: CPTimeout
类型: REG_SZ
数据: 60
```

32 位操作系统的 DB2 ODBC 驱动程序版本 6 和更高版本完全支持连接缓冲，因此注册了此关键字。版本 5.2 客户机必须安装修订包 3 (WR09024) 或更高版本。

缺省值 (60) 意味着在连接被实际断开之前将缓冲 60 秒。

在一个繁忙的环境中，最好是将 CPTimeout 的值该为一个较大的数 （Microsoft 有时建议对于某些环境使用 10 秒），以阻止过多的物理连接和断开，因为这些操作会占用大量系统资源，包括系统内存和通信堆栈资源。

在参与同一个事务的 COM 对象之间再次使用 ODBC 连接

MTS COM 对象中的 ODBC 连接会自动打开连接缓冲（不管该 COM 对象是否是事务性的）。

对于参与同一个事务的多个 MTS COM 对象，可以使用如下方式在两个或多个 COM 对象之间再次使用该连接。

假定有两个 COM 对象 COM1 和 COM2，它们与同一个 ODBC 数据源连接并参与同一个事务。

在 COM1 连接并完成它的工作之后，它就会断开，并缓冲该连接。但是，此连接将被保留，以备同一个事务的其他 COM 对象使用。只有在当前事务结束之后，它才可供其他事务使用。

当在同一个事务中调用 COM2 时，会给予它缓冲的连接。MTS 将确保只将该连接分配给参与同一个事务的 COM 对象。

另一方面，若 COM1 未显式断开，则在该事务结束之后它仍维持着该连接。当在同一个事务中调用 COM2 时，会请求另一个连接。这样，此事务同时维持着两个连接，而不是一个。

由于下列原因，最好对参与同一个事务的 COM 对象使用再次使用连接的功能:

- 它在客户机和服务器中都使用较少的资源。只需要一个连接。
- 它消除了参与同一个事务的两个连接（存取相同的数据库服务器和存取相同的数据）相互锁定的可能性，因为 DB2 服务器将来自 MTS COM 对象的不同连接当作单独的事务来处理。

调整 TCP/IP 通信

若在同时进行太多的物理连接和断开的一个高工作负荷环境中，使用很小的 CPTimeout 值，则 TCP/IP 堆栈可能会遇到资源问题。

要消除此问题，您应该使用 “TCP/IP 注册表项”。有关信息在 *Windows NT Resource Guide* 的第一卷中有描述。该注册表主键值位于 “HKEY_LOCAL_MACHINE-> SYSTEM-> CurrentControlSet-> Services-> TCPIP-> Parameters” 中。

其缺省值和建议的设置如下所示:

名称	缺省值	建议的值
KeepAlive 时间	7200000（2 小时）	相同
KeepAlive 时间间隔	1000（1 秒）	10000（10 秒）

名称	缺省值	建议的值
TcpKeepCnt	120（2 分钟）	240（4 分钟）
TcpKeepTries	20（20 次重试）	相同
TcpMaxConnectAttempts	3	6
TcpMaxConnectRetransmission	3	6
TcpMaxDataRetransmission	5	8
TcpMaxRetransmissionAttempts	7	10
若未定义注册表的值，则创建它。		

使用 MTS ″BANK″ 样本应用程序测试 DB2

可以使用 MTS 所附带的 ″BANK″ 样本程序来测试客户机产品和 MTS 的设置。

执行下列步骤:

1. 更改文件 \Program Files\Common Files\ODBC\Data Sources\ MTSSamples.dsn, 以使它类似于如下所示:

   ```
   [ODBC]
   DRIVER=IBM DB2 ODBC DRIVER
   UID=your_user_id
   PWD=your_password
   DSN=your_database_alias
   Description=MTS Samples
   ```

   其中:

   - your_user_id 和 your_password 是用于连接主机的用户 ID 和口令。
   - your_database_alias 是用于连接数据库服务器的数据库别名。

2. 转至 “控制面板” 中的 “ODBC 管理”，单击 “系统 DSN” 标签并添加数据源:
   a. 选择 “IBM ODBC 驱动程序” 并单击 “完成”。
   b. 当呈示数据库别名的列表时，选择先前指定的那一个。
   c. 单击 “确认”
3. 以 ID your_user_id 的名义，使用 DB2 CLP 与 DB2 数据库连接，如上所述。
   a. 联编 db2cli.lst:

      ```
      db2 bind @C:\sqllib\bnd\db2cli.lst blocking all grant public
      ```

   b. 联编这些实用程序。

      若服务器是 DRDA 主机服务器，则根据您连接的主机（OS/390、AS/400、VSE 或 VM），联编 ddcsmvs.lst、ddcs400.lst 或 ddcsvm.lst。例如:

```
db2 bind @C:\sqllib\bnd\@ddcsmvs.lst blocking all grant public
```

否则，联编 db2ubind.lst:

```
db2 bind @C:\sqllib\bnd\@db2ubind.lst blocking all grant public
```

c. 然后为 MTS 样本应用程序创建样本表和数据，如下所示:

```
DB2 CREATE TABLE ACCOUNT (ACCOUNTNO INT, BALANCE INT)
DB2 INSERT INTO ACCOUNT VALUES(1, 1)
```

4. 在 DB2 客户机上，确保将数据库管理程序配置参数 *tp_mon_name* 设置为 ″MTS″

5. 运行 ″BANK″ 应用程序。选择**帐户**按钮和 **Visual C++** 选项，然后提交该请求。其他选项可使用专用于 SQL 服务器的 SQL，但可能不起作用。

附录F. 使用 DB2 资料库

DB2 通用数据库由联机帮助、书籍（PDF 和 HTML）和 HTML 格式的样本程序组成。本节描述所提供的信息以及如何访问这些信息。

要存取联机产品信息，可以使用“信息中心”。有关详情，参见第544页的『用“信息中心”存取“信息”』。可以查看任务信息、DB2 书籍、疑难解答信息、样本程序和 Web 上的 DB2 信息。

DB2 PDF 文件和打印的书籍

DB2 信息

下表将 DB2 书籍分为四个类别:

DB2 指南和参考信息
这些书籍包含所有平台的公共 DB2 信息。

DB2 安装和配置信息
这些书籍是针对特定平台上的 DB2 的。例如，有分别针对 OS/2 平台、Windows 平台和基于 UNIX 的平台上 DB2 的*快速入门*书籍。

HTML 格式的跨平台样本程序
这些样本是与“应用程序开发客户机”一起安装的样本程序的 HTML 版本。样本仅供参考，并不替代实际程序。

发行说明
这些文件包含 DB2 书籍中未能包括的最新信息。

HTML 格式的安装手册、发行说明和教程可直接在产品 CD-ROM 上看到。大部分书籍在产品 CD-ROM 上都有 HTML 格式以便查看，而在 DB2 出版物 CD-ROM 上则有 Adobe Acrobat (PDF) 格式以便查看和打印。还可从 IBM 订购打印的副本；参见第540页的『订购打印书籍』。下表列示了可订购的书籍。

在 OS/2 和 Windows 平台上，可在 `sqllib\doc\html` 目录下安装 HTML 文件。DB2 信息被翻译成各种语言；但是，并非所有的信息都有每一种语言的翻译版本。每当信息不能以某种特定语言表示出来时，就会提供英语信息

在 UNIX 平台上，可在 `doc/%L/html` （其中 *%L* 表示本国语言环境）目录下安装多种语言版本的 HTML 文件。有关详情，参考适当的*快速入门*书籍。

您可以各种方法来获取 DB2 书籍并存取信息:

- 第543页的『查看联机信息』
- 第547页的『搜索联机信息』
- 第540页的『订购打印书籍』
- 第539页的『打印 PDF 书籍』

表 45. DB2 信息

名称	说明	书号 **PDF 文件名**	**HTML 目录**
	DB2 指南和参考信息		
管理指南	*管理指南: 计划*提供数据库概念的概述、有关设计问题（如逻辑和物理数据库设计）的信息，以及高可用性的讨论。 *管理指南: 实现*提供有关实现问题（如实现设计、存取数据库、审核、备份和恢复）的信息。 *管理指南: 性能*提供有关数据库环境以及应用程序性能评估和调整的信息。 在北美，可使用书号 SBOF-8934 来订购三卷英文版的*管理指南*。	SB84-0219 db2d1x70 SB84-0218 db2d2x70 SB84-0243 db2d3x70	db2d0
Administrative API Reference	描述 DB2 应用程序设计接口 (API) 以及您可以用来管理数据库的数据结构。此书还说明如何在应用程序中调用 API。	SC09-2947 db2b0x70	db2b0
应用程序构建指南	提供环境设置信息和关于如何在 Windows、 OS/2 和基于 UNIX 的平台上编译、链接和运行 DB2 应用程序的逐步指导。	SB84-0220 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	提供关于使用 DB2 通用数据库产品时可能遇到的 APPC、 CPI-C 和 SNA 检测码的一般信息。 仅有 HTML 格式的版本。	无书号 db2apx70	db2ap

表 45. DB2 信息 (续)

名称	说明	书号 PDF 文件名	HTML 目录
Application Development Guide	说明如何开发使用嵌入式 SQL 或 Java（JDBC 和 SQLJ）来存取 DB2 数据库的应用程序。讨论主题包括在分区环境或联合体系统中编写存储过程、编写用户定义函数、创建用户定义类型、使用触发器和开发应用程序。	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	说明如何开发使用 “DB2 调用层接口”（一个与 Microsoft ODBC 规范兼容的可调用 SQL 接口）来存取 DB2 数据库的应用程序。	SC09-2950 db2l0x70	db2l0
Command Reference	说明如何使用“命令行处理器”，并描述可用来管理数据库的 DB2 命令。	SC09-2951 db2n0x70	db2n0
Connectivity Supplement	提供有关以下各项的设置和参考信息：如何将作为 DRDA 应用程序请求器的 DB2 AS/400 版、DB2 OS/390 版、 DB2 MVS 版、DB2 VM 版与 DB2 通用数据库服务器配合使用。此书还详述了如何将 DRDA 应用服务器与 DB2 Connect 应用程序请求器配合使用。 仅有 HTML 和 PDF 格式。	无书号 db2h1x70	db2h1
Data Movement Utilities Guide and Reference	说明如何使用 DB2 实用程序（如调入、调出、装入、自动装入程序和 DPROP）来使数据移动易于进行。	SC09-2955 db2dmx70	db2dm
数据仓库中心管理指南	提供有关如何使用“数据仓库中心”构建和维护数据仓库的信息。	SB84-0226 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	提供帮助程序员将应用程序与“数据仓库中心”和“信息目录管理程序”集成的信息。	SC26-9994 db2adx70	db2ad
DB2 Connect 用户指南	提供 DB2 Connect 产品的概念、程序设计以及一般用法信息。	SB84-0221 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	提供 DB2 Query Patroller 系统的操作概述、特定操作和管理信息以及管理图形用户界面实用程序的任务信息。	SC09-2958 db2dwx70	db2dw

表 45. DB2 信息 (续)

名称	说明	书号 **PDF 文件名**	**HTML 目录**
DB2 Query Patroller 用户指南	描述如何使用 DB2 Query Patroller 的工具和功能。	SB84-0222 db2wwx70	db2ww
词汇表	提供 DB2 及其部件中使用的术语的定义。 有 HTML 格式可用且在 *SQL Reference* 中。	无书号 db2t0x70	db2t0
Image, Audio, and Video Extenders 管理和程序设计	提供有关 DB2 Extender 的一般信息，有关 Image, Audio and Video (IAV) Extender 的管理和配置的信息，以及有关使用 IAV Extender 进行程序设计的信息。它包括参考信息、诊断资料（带有信息）和样本。	SB84-0247 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	提供有关管理信息目录的指南。	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	提供“信息目录管理程序”的体系结构接口的定义。	SC26-9997 db2bix70	db2bi
信息目录管理程序用户指南	提供有关使用“信息目录管理程序”用户界面的信息。	SB84-0227 db2aix70	db2ai
安装和配置补遗	指导您了解计划、安装和设置特定于平台的 DB2 客户机。此补遗还包含关于联编、设置客户机和服务器通信、DB2 GUI 工具、 DRDA AS、分布式安装、配置分布式请求和存取多机种数据源的信息。	GB84-0127 db2iyx70	db2iy
信息参考	列出由 DB2、信息目录管理程序和数据仓库中心发出的信息和代码，并描述应执行的操作。 在北美，您可订购两卷英文版的信息参考（使用书号 SBOF-8932）。	第 1 卷 GB84-0216 db2m1x70 第 2 卷 GB84-0217 db2m2x70	db2m0
OLAP Integration Server Administration Guide	说明如何使用“OLAP 集成服务器”的“管理程序”部件。	SC27-0787 db2dpx70	n/a

表 45. DB2 信息 (续)

名称	说明	书号 PDF 文件名	HTML 目录
OLAP Integration Server Metaoutline User's Guide	说明如何使用标准 "OLAP 元轮廓" 接口（而非通过使用 "元轮廓辅助程序"）创建和填充 OLAP 元轮廓。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	说明如何使用标准 "OLAP 模型接口"（而非使用 "模型辅助程序"）来创建 OLAP模型。	SC27-0783 db2lpx70	n/a
OLAP Setup and User's Guide	提供 OLAP Starter Kit 的配置和设置信息。	SC27-0702 db2ipx70	db2ip
OLAP Spreadsheet Add-in User's Guide for Excel	描述如何使用 Excel 电子表格程序来分析 OLAP 数据。	SA40-1756 db2epx70	db2ep
OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3	描述如何使用 Lotus 1-2-3 电子表格程序来分析 OLAP 数据。	SA40-1757 db2tpx70	db2tp
Replication Guide and Reference	提供随 DB2 提供的 "IBM 复制" 工具的计划、配置、管理和用法信息。	SC26-9920 db2e0x70	db2e0
Spatial Extender 用户指南和参考	提供关于 Spatial Extender 的安装、配置、管理、程序设计和疑难解答的信息。还提供空间数据概念的重要说明，并提供 Spatial Extender 特定的参考资料（信息和 SQL）。	SB84-0249 db2sbx70	db2sb
SQL 入门	介绍 SQL 概念，并提供许多构造和任务的示例。	SB84-0223 db2y0x70	db2y0
SQL Reference，第 1 卷和第 2 卷	描述 SQL 语法、语义和语言规则。此书还包括关于发行版间的不兼容性、产品限制和目录视图的信息。 在北美，可使用书号 SBOF-8933 来订购两卷英文版的 *SQL Reference*。	第 1 卷 SC09-2974 db2s1x70 第 2 卷SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	描述如何收集关于数据库和数据库管理程序的各种信息。此书说明如何利用信息来了解数据库活动、提高性能和确定问题的原因。	SC09-2956 db2f0x70	db2f0

表 45. DB2 信息 (续)

名称	说明	书号 PDF 文件名	HTML 目录
Text Extender 管理和程序设计	提供有关 DB2 Extender 的一般信息，有关 Text Extender 的管理和配置的信息，以及有关使用 Text Extender 进行程序设计的信息。它包括参考信息、诊断资料（带有信息）和样本。	SB84-0248 desu9x70	desu9
Troubleshooting Guide	帮助您确定错误源、从问题中恢复并向“DB2 客户服务”咨询以使用诊断工具。	GC09-2850 db2p0x70	db2p0
新增内容	描述 DB2 通用数据库（版本 7）中的新特性、函数和增强功能。	SB84-0224 db2q0x70	db2q0
	DB2 安装和配置信息		
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	提供 OS/2 和 Windows 32 位操作系统上的 DB2 Connect 企业版的计划、迁移、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2953 db2c6x70	db2c6
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	提供基于 UNIX 的平台上的 DB2 Connect 企业版的计划、迁移、安装、配置和任务信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2952 db2cyx70	db2cy
DB2 Connect 个人版快速入门	提供 OS/2 和 Windows 32 位操作系统上的 DB2 Connect 个人版的计划、迁移、安装、配置和任务信息。此书还包含所有受支持的客户机的安装和设置信息。	GB84-0212 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings Linux 版	在进行所有受支持的 Linux 分布式系统时，提供“DB2 Connect 个人版”的计划、安装、迁移和配置信息。	GC09-2962 db2c4x70	db2c4
DB2 DataLinks Manager 快速入门	提供“DB2 DataLinks Manager AIX 版”和 Windows 32 位操作系统的计划、安装、配置和任务信息。	GB84-0211 db2z6x70	db2z6
DB2 扩充企业版 UNIX 版快速入门	提供在基于 UNIX 的平台上的 DB2 扩充企业版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0209 db2v3x70	db2v3

表 45. DB2 信息 (续)

名称	说明	书号 PDF 文件名	HTML 目录
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	提供 DB2 扩充企业版 Windows 32 位操作系统版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2963 db2v6x70	db2v6
DB2（OS/2 版）Quick Beginnings	提供 OS/2 操作系统上的 DB2 通用数据库的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2968 db2i2x70	db2i2
DB2（UNIX 版）快速入门	提供在基于 UNIX 的平台上的 DB2 通用数据库的计划、安装、迁移和配置信息。 此书还包含许多受支持的客户机的安装和设置信息。	GB84-0214 db2ixx70	db2ix
DB2 Windows 版快速入门	提供 Windows 32 位操作系统上的 DB2 通用数据库的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0215 db2i6x70	db2i6
DB2 个人版快速入门	提供 OS/2 和 Windows 32 位操作系统上的 “DB2 通用数据库个人版” 的计划、安装、迁移和配置信息。	GB84-0213 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings Linux 版	在进行所有受支持的 Linux 分布式系统时，提供 “DB2 通用数据库个人版” 的计划、安装、迁移和配置信息。	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller 安装指南	提供有关 DB2 Query Patroller 的安装信息。	GB84-0208 db2iwx70	db2iw
DB2 数据仓库管理程序安装指南	提供仓库代理程序、仓库变换器和 “信息目录管理程序” 的安装信息。	GB84-0122 db2idx70	db2id
	HTML 格式的跨平台样本程序		

表 45. DB2 信息 (续)

名称	说明	书号 PDF 文件名	HTML 目录
HTML 格式的样本程序	为所有受 DB2 支持的平台上的程序设计语言提供 HTML 格式的样本程序。提供的样本程序仅供参考。并非所有样本都有所有程序设计语言的版本。HTML 样本仅当安装了 “DB2 应用程序开发客户机” 时才可用。 有关这些程序的详情，*参考应用程序构建指南*。	无书号	db2hs
	发行说明		
DB2 Connect 发行说明	提供 DB2 书籍中未能包括的最新信息。	参见注释 2。	db2cr
DB2 安装注释	提供 DB2 书籍中未能包括的最新安装特定信息。	仅在产品 CD-ROM 上提供。	
DB2 发行说明	提供 DB2 书籍中未能包括的、有关所有 DB2 产品和功能部件的最新信息。	参见注释 2。	db2ir

注:

1. 文件名第六个位置的字符 *x* 指示书籍的语言版本。例如，文件名 db2d0e70 标识英语版本的*管理指南*，而文件名 db2d0f70 标识同一本书的法语版本。下列字母用在文件名的第六个位置以指示语言版本:

语言	标识符
巴西葡萄牙语	b
保加利亚语	u
捷克语	x
丹麦语	d
荷兰语	q
英语	e
芬兰语	y
法语	f
德语	g
希腊语	a
匈牙利语	h
意大利语	i
日语	j
韩国语	k
挪威语	n
波兰语	p

葡萄牙语	v
俄语	r
简体中文	c
斯洛文尼亚语	l
西班牙语	z
瑞典语	s
繁体中文	t
土耳其语	m

2. DB2 书籍中未能包括的最新信息以 HTML 格式在“发行说明”中提供，或作为 ASCII 文件提供。在“信息中心”中和产品 CD-ROM 上都提供了 HTML 版本。要查看 ASCII 文件:
   - 在基于 UNIX 的平台上，参见 `Release.Notes` 文件。此文件位于 `DB2DIR/Readme/%L` 目录中，其中 *%L* 表示本国语言环境名，而 `DB2DIR` 表示:
     - 在 AIX 上，是 `/usr/lpp/db2_07_01`
     - 在 HP-UX、PTX、Solaris 和 Silicon Graphics IRIX 上，是 `/opt/IBMdb2/V7.1`
     - 在 Linux 上，是 `/usr/IBMdb2/V7.1`。
   - 在其它平台上，参见 `RELEASE.TXT` 文件。此文件在安装了产品的目录中。在 OS/2 平台上，还可双击 **IBM DB2** 文件夹，然后双击**发行说明**图符。

打印 PDF 书籍

如果想要书籍的打印副本，则可打印 DB2 出版物 CD-ROM 上的 PDF 文件。使用 Adobe Acrobat 读入程序，可打印整本书籍或特定范围内的页。有关库中每本书的文件名，参见第532页的表45。

可从 Adobe Web 站点（网址 http://www.adobe.com）获取 Adobe Acrobat 读入程序的最新版本。

这些 PDF 文件包括在 DB2 出版物 CD-ROM 上，文件扩展名为 PDF。要存取这些 PDF 文件:

1. 插入 DB2 出版物 CD-ROM。在基于 UNIX 的平台上，安装 DB2 出版物 CD-ROM。参考*快速入门*一书以了解安装过程。
2. 启动 Acrobat 读入程序。
3. 从下列位置之一打开期望的 PDF 文件:
   - 在 OS/2 和 Windows 平台上:

     x`:\doc\`*language* 目录，其中 *x* 表示 CD-ROM 驱动器而 *language* 表示两个字符的国家代码，它表示您所用的语言（例如，EN 表示英语）。
   - 在基于 UNIX 的平台上:

CD-ROM 上的 */cdrom*/doc/*%L* 目录，其中 */cdrom* 表示 CD-ROM 的安装点而 *%L* 表示期望的本国语言环境的名称。

还可从 CD-ROM 将 PDF 文件复制至本地或网络驱动器并从该处读取它们。

订购打印书籍

可通过使用销售单 (SBOF) 书号单本地或成套地订购打印的 DB2 书籍（仅限北美）。要订购书籍，与 IBM 授权经销商或市场代表联系，或致电 1-800-879-2755（美国）或 1-800-IBM-4YOU（加拿大）。还可从 Publications Web 页（网址为 http://www.elink.ibmlink.ibm.com/pbl/pbl）订购这些书籍。

有两套书籍。SBOF-8935 提供了“DB2 仓库管理程序”的参考和用法信息。SBOF-8931 提供了所有其他“DB2 通用数据库”产品和功能部件的参考和用法信息。每个 SBOF 的内容列示在下表中:

表 46. 订购打印书籍

SBOF 号	包括的书籍	
SBOF-8931	• Administration Guide: Planning • Administration Guide: Implementation • Administration Guide: Performance • Administrative API Reference • Application Building Guide • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • Data Warehouse Center Administration Guide • Data Warehouse Center Application Integration Guide • DB2 Connect User's Guide • Installation and Configuration Supplement • Image, Audio, and Video Extenders Administration and Programming • Message Reference, Volumes 1 and 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP Setup and User's Guide • OLAP Spreadsheet Add-in User's Guide for Excel • OLAP Spreadsheet Add-in User's Guide for Lotus 1-2-3 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL Getting Started • SQL Reference, Volumes 1 and 2 • System Monitor Guide and Reference • Text Extender Administration and Programming • Troubleshooting Guide • What's New
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager User's Guide • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2 联机文档

存取联机帮助

随所有 DB2 部件都附带提供了联机帮助。下表描述了各种类型的联机帮助。

帮助类型	内容	如何存取...
命令帮助	说明命令行处理器中命令的语法。	从命令行处理器，以交互方式输入: `? command` 其中 *command* 表示一个关键字或整个命令。 例如，`? catalog` 显示所有 CATALOG 命令的帮助，而 `? catalog database` 显示 CATALOG DATABASE 命令的帮助。
客户机配置辅助程序帮助 *命令中心帮助* *控制中心帮助* *数据仓库中心帮助* *事件分析程序帮助* *信息目录管理程序帮助* *卫星管理中心帮助* *脚本中心帮助*	说明您可在窗口或笔记本中执行的任务。此帮助包括您需要知道的概述和前提条件信息，并描述如何使用窗口或笔记本控件。	从窗口或笔记本，单击**帮助**按钮或按 **F1** 键。

信息帮助	描述信息的起因以及您应该执行的任何操作。	从命令行处理器，以交互方式输入:
		`? XXXnnnnn`
		其中 *XXXnnnnn* 表示有效的信息标识符。
		例如，`? SQL30081` 显示关于 SQL30081 信息的帮助。
		要每次查看一屏信息帮助，可输入:
		`? XXXnnnnn | 尚有`
		要在文件中保存信息帮助，可输入:
		`? XXXnnnnn > filename.ext`
		其中 *filename.ext* 表示想要保存信息帮助的文件。

帮助类型	内容	如何存取...
SQL 帮助	说明 SQL 语句的语法。	从命令行处理器，以交互方式输入: `help statement` 其中，*statement* 表示 SQL 语句。 例如， `help SELECT` 显示有关 SELECT 语句的帮助。 **注:** 在基于 UNIX 的平台上，SQL 帮助不可用。
SQLSTATE 帮助	说明 SQL 状态及类代码。	从命令行处理器，以交互方式输入: `? sqlstate` 或 `? class code` 其中， *sqlstate* 表示有效的五位 SQL 状态，而 *class code* 表示该 SQL 状态的头两位。 例如，`? 08003` 显示 08003 SQL状态的帮助，而 `? 08` 显示 08 类代码的帮助。

查看联机信息

此产品中的书籍为超文本标记语言 (HTML) 软拷贝格式。软拷贝格式使您可搜索或浏览信息，并提供访问相关信息的超文本链接。 它还使得在站点间共享库更容易。

可使用遵循 HTML 版本 3.2 规范的任何浏览器来查看联机书籍或样本程序。

要查看联机书籍或样本程序:

- 如果正在运行 DB2 管理工具，则使用 “信息中心”。
- 从浏览器，单击**文件 —> 打开页**。打开的页中包含 DB2 信息的描述和至 DB2 信息的链接:
  - 在基于 UNIX 的平台上，打开以下页:

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    其中 *%L*表示本国语言环境名称
  - 在其它平台上，打开以下页:

    sqllib\doc\html\index.htm

    该路径位于安装了 DB2 的驱动器上。

    如果尚未安装 “信息中心”，则可通过双击 **DB2 信息**图符来打开该页。视您正在使用的系统不同，图符在主产品文件夹中或在 “Windows 开始” 菜单中。

安装 Netscape 浏览器

如果还未安装 Web 浏览器，则可从产品包装箱中的 Netscape CD-ROM 安装 Netscape。要获取如何安装它的详细指导，执行:

1. 插入 Netscape CD-ROM。
2. 安装 CD-ROM（仅限于在基于 UNIX 的平台上）。参考*快速入门*一书以了解安装过程。
3. 有关安装指导，参考 CDNAV *nn*.txt 文件，其中 *nn* 表示两字符语言标识符。该文件位于 CD-ROM 的根目录下。

用“信息中心”存取“信息”

“信息中心”提供对 DB2 产品信息的快速存取。在所有装有 DB2 管理工具的平台上，都提供了“信息中心”。

可通过双击“信息中心”图符来打开“信息中心”。视正在使用的系统的不同，该图符在主产品文件夹的“信息”文件夹中，或在 Windows 的**开始**菜单中。

还可通过使用工具栏和 DB2 Windows 平台上的**帮助**菜单来存取“信息中心”。

“信息中心”提供了六种类型的信息。单击适当的标签来查看提供给该类型的主题。

任务 可使用 DB2 执行的关键任务。

参考 DB2 参考信息，如关键字、命令以及 API。

书籍 DB2 书籍。

疑难解答 错误信息类别及其恢复操作。

样本程序 随“DB2 应用程序开发客户机”一起提供的样本程序。如果未安装“DB2 应用程序开发客户机”，则不显示此标签。

Web 万维网（WWW）上的 DB2 信息。要存取此信息，必须从系统连接至 Web。

当选择其中一个列表中的项时，“信息中心”启动一个查看器来显示信息。视所选择的信息种类的不同，查看器可能是系统帮助查看器、编辑器或 Web 浏览器。

“信息中心”提供了查找功能部件，因此您不用浏览这些列表就能查找特定主题。

对于全文本搜索，请遵循“信息中心”中指向**搜索 DB2 联机信息**搜索表格的超文本链接。

HTML 搜索服务器通常是自动启动的。如果 HTML 信息中的搜索不起作用，则可能必须使用下列其中一个方法来启动搜索服务器:

在 Windows 上

单击**开始**并选择**程序 —> IBM DB2 —> 信息 —> 启动 HTML 搜索服务器**。

在 OS/2 上

双击 **DB2 OS/2 版**文件夹，然后双击**启动 HTML 搜索服务器**图符。

如果在搜索 HTML 信息时遇到任何其它问题，可参考发行说明。

注: 搜索功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。

使用 DB2 向导

向导通过让您一次一步地完成每一个任务来协助您完成特定管理任务。可通过控制中心和客户机配置辅助程序来获取向导。下表列出了这些向导并描述了它们的用途。

注: “创建数据库”、“创建索引”、“配置多站点更新”和“性能配置”向导对分区数据库环境可用。

向导	帮助您...	如何存取...
添加数据库	在客户机工作站上编目数据库。	从“客户机配置辅助程序”单击**添加**。
备份数据库	确定、创建并调度应急计划。	从“控制中心”，用鼠标右键单击想要备份的数据库并选择**备份 —> 数据库（使用向导）**。
配置多站点更新	配置多站点更新、分布式事务或两阶段落实。	从“控制中心”，用鼠标右键单击**数据库**文件夹并选择**多站点更新**。
创建数据库	创建数据库并执行一些基本配置任务。	从“控制中心”，用鼠标右键单击**数据库**文件夹，并选择**创建 —> 数据库（使用向导）**。
创建表	选择基本数据类型并创建表的主关键字。	从“控制中心”，用鼠标右键单击**表**图符，并选择**创建 —> 表（使用向导）**。
创建表空间	创建新的表空间。	从“控制中心”，用鼠标右键单击**表空间**图符，并选择**创建 —> 表空间（使用向导）**。

向导	帮助您...	如何存取...
创建索引	建议对于所有查询要创建和卸下哪些索引。	从“控制中心”，用鼠标右键单击**索引**图符，并选择**创建 —> 索引（使用向导）**。
性能配置	通过更新配置参数来调整数据库性能以满足您的业务需求。	从“控制中心”，用鼠标右键单击想要调整的数据库并选择**使用向导配置性能**。 对于分区数据库环境，从“数据库分区”视图，用鼠标右键单击想要调整的首个数据库分区并选择**使用向导配置性能**。
复原数据库	在故障之后恢复数据库。它帮助您了解要使用的备份及要重放的纪录。	从“控制中心”，用鼠标右键单击想要复原的数据库并选择**复原 —> 数据库（使用向导）**。

设置文档服务器

在缺省情况下，DB2 信息安装在本地系统上。这表示需要存取 DB2 信息的每个人都必须安装相同的文件。要将 DB2 信息存储在单个位置中，执行下列步骤:

1. 将所有文件和子目录从本地系统上的 `\sqllib\doc\html` 复制至 Web 服务器。每一本书都有其自己的子目录，该子目录包含构成该书的所有必需的 HTML 和 GIF 文件。确保目录结构仍相同。
2. 配置 Web 服务器以查找新位置中的文件。有关信息，可参考*安装和配置补遗*中的 NetQuestion 附录。
3. 如果正在使用“信息中心”的 Java 版本，可为所有 HTML 文件指定基本的 URL。您应将该 URL 用于书籍列表。
4. 当能够查看书籍文件时，可将经常查看的主题做成书签。您可能想把下列各页做成书签:
   - 书籍列表
   - 经常使用的书籍的目录
   - 经常引用的文章，如 ALTER TABLE 主题
   - 搜索格式

有关如何从中央机器处理 DB2 通用数据库联机文档文件的信息，*参考安装和配置补遗*中的 NetQuestion 附录。

搜索联机信息

要查找 HTML 文件中的信息，使用下列方法之一:

- 在顶部框中单击**搜索**。使用搜索格式来查找特定的主题。 此功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。
- 在顶部框中单击**索引**。使用索引来查找书中的特定主题。
- 显示帮助或 HTML 书籍的目录或索引，然后使用 Web 浏览器的查找功能查找书中的特定主题。
- 使用 Web 浏览器的书签功能来快速返回至特定的主题。
- 使用“信息中心”的搜索功能来查找特定的主题。参见第544页的『用“信息中心”存取“信息”』以获取详情。

附录G. 注意事项

IBM 可能未在所有国家中提供本文档中讨论的产品、服务或功能部件。关于您所在区域目前可用的产品及服务的信息，请向当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来替代 IBM 产品、程序或服务。当然，评估和验证非 IBM 产品、程序或服务均由用户自行负责。

本文档的议题可能涉及 IBM 的某些专利或正在申请中的专利的应用。提供本文档，并不表示允许您使用这些专利。您可以将许可证查询以书面形式寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

关于双字节 (DBCS) 许可证查询的信息，请与您所在国家的 IBM 知识产权部门联系，将查询以书面形式寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于英国与其它当地法律不允许这种供应方式的国家: 国际商用机器公司『按原样』出版此书，不做任何明确或暗示的担保，包括但不限于有关非伪造、商业性或符合特殊目的的隐含保证。一些地区在某些事务中不允许否认拒绝明确或暗示的担保，因此本条款可能不适合您。

本信息中可能有技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些信息将包含在本书新的版本中。IBM 可以随时对本书中说明的产品和/或程序进行改进和/或改动，而不必通知您。

此信息中对非 IBM Web 站点的任何引用仅是为了方便起见，而不以任何方式为那些 Web 站点作保证。那些 Web 站点的资料并非此 IBM 产品资料的一部分，使用那些 Web 站点的风险由您自己承担。

对于您所提供的任何信息，IBM 有权利以任何她认为适当的方式使用或散发，而不必对您负任何责任。

为了以下目的：(1) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换 (2) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付款，都可获取这方面的信息。

这些信息中描述的特许程序及其所有可用的特许资料，按 IBM 客户协议、IBM 国际程序许可证协议或任何等价的协议中的条款，由 IBM 提供。

此处包含的所有性能数据都是在受控环境中确定的。因此，在其他操作环境中获得的结果可能与之相差很大。某些测量可能是在开发级的系统上进行的，不能保证这些测量方法在通用系统上同样可用。此外，某些测量方法可能是通过外推法归纳来估计的。实际结果可能会有所不同。此文档的用户应针对他们的特定环境验证数据是否适用。

涉及非 IBM 产品的信息可从这些产品的供应商、其发行公告或其它公众可用源得到。IBM 未测试这些产品，因此不能确认性能的精确度、兼容性或其它对非 IBM 产品的索赔赔偿要求等。有关非 IBM 产品功能方面的问题可向它们的供应商提出。

所有关于 IBM 未来方向或意向的声明都可能随时更改或撤消，而不作任何通知，并且仅代表发展目标。

此信息包含了用于日常商业处理的数据和报表的示例。为了尽可能完整地说明问题，这些示例中包含了个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址相似，纯属巧合。

版权许可证:

本信息中可能包含用源语言编写的示例应用程序，它们说明了各种不同的操作平台上的程序设计技术。您可以为了开发、使用、市场营销或分发应用程序(这些应用程序遵守编写这些示例程序的操作平台的应用程序接口)的目的，以任何形式复

制、修改和分发这些示例程序，不用向 IBM 付费。这些例子未经所有条件下的完整测试。因此，IBM 不能保证或暗示其可靠性、可用性或这些程序的功能。

这些样本程序或任何派生产品的每个副本或任何部分必须包含如下的版权公告:

注册商标

以星号 (*) 标出的下列术语是 IBM 公司在美国和／或其他国家的商标。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extender
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

下列各项是其他公司的商标或注册商标:

Microsoft、Windows、和 Windows NT 是 Microsoft 公司的商标或注册商标。

Java 或所有基于 Java 的商标和标志以及 Solaris 是 Sun Microsystems 公司在美国和／或其他国家的商标。

Tivoli 和 NetView 是 Tivoli Systems 公司在美国和／或其他国家的商标。

UNIX 是经 X/Open 有限公司唯一许可的在美国和／或其它国家的注册商标。

以双星号 (**) 标出的其他公司、产品或服务名，可能是其他公司的商标或服务标志。

索引

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[N]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

E

F

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

与 IBM 联系

如果有技术问题，请在与“DB2 客户支持中心”联系之前复查并执行 *Troubleshooting Guide* 所建议的操作。本指南对您可以收集哪些信息以使“DB2 客户支持中心”更好地为您服务提出了建议。

要获取信息或订购任何“DB2 通用数据库”产品，与当地分支机构的 IBM 代表联系，或与任何特许 IBM 软件经销商联系。

您如果住在美国，请致电下列其中一个号码:

- 1-800-237-5511，可获得客户支持
- 1-888-426-4343，可了解所提供的服务项目

产品信息

您如果住在美国，请致电下列其中一个号码:

- 1-800-IBM-CALL (1-800-426-2255) 或 1-800-3IBM-OS2 (1-800-342-6672)，可订购产品或获取一般信息。
- 1-800-879-2755，可订购出版物。

http://www.ibm.com/software/data/

DB2 万维网网页提供关于新闻、产品说明、培训计划等等的当前 DB2 信息。

http://www.ibm.com/software/data/db2/library/

“DB2 产品和服务技术库”可供您访问常见问题、修订、书籍以及最新的 DB2 技术资料。

注: 此资料可能只有英文版。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

International Publications Ordering Web 站点提供关于如何订购书籍的信息。

http://www.ibm.com/education/certify/

IBM Web 站点中的“专业认证程序”提供各种 IBM 产品（包括 DB2）的认证测试信息。

ftp.software.ibm.com

以匿名形式注册。可在目录 `/ps/products/db2` 中找到有关 DB2 和许多其他产品的演示程序、修订、信息和工具。

comp.databases.ibm-db2, bit.listserv.db2-l

这些 Internet 新闻组可供用户来讨论使用 DB2 产品的经验。

On Compuserve: GO IBMDB2

输入此命令来访问 IBM DB2 系列论坛。这些论坛支持所有的 DB2 产品。

有关如何在美国以外的地区与 IBM 联系的信息，参见 *IBM Software Support Handbook* 的附录 A。要存取此文档，访问以下 Web 页面：http://www.ibm.com/support/，然后选择该页面底部附近的 IBM Software Support Handbook 链接。

注： 在某些国家，IBM 特许经销商应与他们的经销商支持机构联系，而不是与“IBM 支持中心”联系。

Printed in China

SB84-0243-00