IBM® DB2 Universal Database™
DB2 通用数据库

管理指南：性能

版本 8.2

S152-0166-01

IBM® DB2 Universal Database™
DB2 通用数据库

管理指南：性能

版本 8.2

S152-0166-01

在使用本资料及其支持的产品之前，请务必阅读『*声明*』中的一般信息。

本文档包含 IBM 的专利信息。它是根据许可协议提供的，并受版权法保护。本出版物包含的信息不包括任何产品保证，且本手册提供的任何声明不应作如此解释。

可以用在线方式或通过您当地的 IBM 代表订购 IBM 出版物。

- 要以在线方式订购出版物，可访问 IBM 出版物中心（IBM Publications Center），网址为 www.ibm.com/shop/publications/order。
- 要查找您当地的 IBM 代表，可访问 IBM 全球联系人目录（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以任何它认为适当的方式使用或分发，而不必对您负任何责任。

目录

第 6 章 了解 SQL 编译器 121

第 7 章 SQL 说明设施 171

第 3 部分 调整和配置系统 189

第 8 章 操作性能 191

第 9 章 使用控制器 241

关于本书

“管理指南”有三卷提供了使用和管理 DB2 关系数据库管理系统（RDBMS）产品所需要的信息，这些信息包括:

- 关于数据库设计的信息（可在《*管理指南: 计划*》中找到）
- 关于实现和管理数据库的信息（可在《*管理指南: 实现*》中找到）
- 关于配置和调整数据库环境以提高性能的信息（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界面来执行:

- **命令行处理器**，它允许您从图形界面存取和处理数据库。从此界面，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说明的是此界面的用法。有关使用命令行处理器的更多信息，请参阅 *Command Reference*。
- **应用程序编程接口**，允许您在应用程序内执行 DB2 实用程序功能。有关使用应用程序编程接口的更多信息，请参阅 *Administrative API Reference*。
- **控制中心**，允许您使用图形用户界面来执行管理任务，例如，配置系统、管理目录、备份和恢复系统、调度作业和管理介质。“控制中心”还包含“复制管理”，它允许您设置系统间的数据复制。此外，“控制中心”使您能通过图形用户界面执行 DB2 实用程序功能。对于不同的平台，调用“控制中心”的方法也不同。例如，在命令行上使用 db2cc 命令，从 DB2 文件夹中选择“控制中心”图标，而在 Windows 平台上则使用“开始”菜单。要获取介绍性帮助，从“控制中心”窗口的**帮助**下拉菜单中选择**入门**。**Visual Explain** 工具是从控制中心调用的。

  在下面三个视图中提供了“控制中心”:
  - 基本。此视图显示基本对象（例如，数据库、表和存储过程）上的核心 DB2 UDB 功能。
  - 高级。此视图具有所有对象和可用的操作。如果您正在企业环境中工作，并且您想连接至 DB2 z/OS 版或 IMS 版，则使用此视图。
  - 定制。此视图使您能够调整对象树和对象操作。

还可以使用其它工具来执行管理任务。这些工具包括:

- “命令编辑器”，它取代了“命令中心”，它用来生成、编辑、运行和处理 SQL 语句；IMS 和 DB2 命令；使用获得的输出；以及查看说明 SQL 语句的存取方案的图形表示法。
- “开发中心”，它能够支持本机 SQL 持久存储器模块（PSM）存储过程； Java 存储过程 iSeries 版 V5R3 和更新版本；用户定义的函数（UDF）和结构化类型。
- “健康中心”提供了一个工具来帮助 DBA 解决性能和资源分配问题。
- “工具设置”，它用来更改“控制中心”、“健康中心”和“复制中心”的设置。
- “日志”，它用来调度要以无人照管方式运行的作业。
- “数据仓库中心”，它用来管理仓库对象。

本书的读者

本书主要面向需要设计、实现和维护要由本地或远程客户机存取的数据库的数据库管理员、系统管理员、安全性管理员和系统操作员。需要了解 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）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

本书包含有关下列主要主题的信息:

性能介绍

- 第 1 章,『性能介绍』, 介绍管理和提高 DB2 UDB 性能的概念和注意事项。
- 第 2 章,『体系结构和进程』, 介绍基础的“DB2 通用数据库”体系结构和进程。

调整应用程序性能

- 第 3 章,『应用程序注意事项』, 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,『环境注意事项』, 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 编译器』, 描述当使用 SQL 编译器编译 SQL 语句的过程。
- 第 7 章,『SQL 说明设施』, 描述“说明”设施, 它允许您检查 SQL 编译器为存取数据所作的选择。

调整和配置系统

- 第 8 章,『操作性能』, 概述数据库管理器使用内存的方式以及影响运行时性能的其它注意事项。
- 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,『调整配置』, 介绍增大数据库系统大小时要考虑的一些注意事项和要执行的任务。
- 第 11 章,『将数据再分发至数据库分区』, 讨论在一个分区数据库环境中将数据再分发在各分区中所需的任务。
- 第 12 章,『基准程序测试』, 提供有关基准程序测试和如何执行基准程序测试的概述。
- 第 13 章,『配置 DB2』, 讨论数据库管理器和数据库配置文件, 以及数据库管理器、数据库和 DAS 配置参数的值。

附录

- 附录 A,『DB2 注册表变量和环境变量』, 显示概要文件注册表值和环境变量。
- 附录 B,『说明表』, 说明表部分提供关于“DB2 说明”设施使用的表以及如何创建这些表的信息。
- 附录 C,『SQL 说明工具』, 提供有关使用 DB2 说明工具 db2expln 和 dynexpln 的信息。
- 附录 D,『db2exfmt − 说明表格式化』, 格式化 DB2 说明表的内容。

“管理指南”的其它卷的简要概述

管理指南：计划

*《管理指南：计划》*主要讨论数据库设计。它说明了逻辑与物理设计问题和分布式事务问题。下面简要描述该卷中的特定章节和附录:

数据库概念

- “基本关系数据库概念”提供数据库对象（包括恢复对象、存储器对象和系统对象）的概述。
- “并行数据库系统”提供有关 DB2 可使用的并行性类型的介绍。
- “关于数据仓储”提供数据仓储和数据仓储任务的概述。

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的注意事项。
- “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。
- “针对事务管理器进行设计”讨论如何在分布式事务处理环境中使用数据库。

附录

- “发行版之间的不兼容性”说明了“版本 7”和“版本 8”带来的不兼容性，以及将来应注意的不兼容性。
- “本地语言支持（NLS）”描述“DB2 本地语言支持”，包括关于地域、语言和代码页的信息。
- “在 64 位环境中启用大页支持（AIX）”讨论对 16 MB 页大小的支持以及如何启用此支持。

管理指南：实现

*《管理指南：实现》*主要讨论数据库设计的实现。下面简要描述该卷中的特定章节和附录:

实现设计

- “在创建数据库之前”描述在创建数据库和数据库中的对象之前所需的先决条件。
- “创建和使用 DB2 管理服务器（DAS）”讨论什么是 DAS、如何创建它以及如何使用它。
- “创建数据库”描述与创建数据库及相关数据库对象相关联的那些任务。
- “创建表和其它相关表对象”描述在实现数据库设计时如何创建具有特定特征的表。
- “改变数据库”讨论改变数据库之前必须完成的操作，以及与修改或删除数据库或相关数据库对象相关联的任务。
- “改变表和其它相关表对象”描述如何删除表或者如何修改与这些表相关联的特定特征。同时还描述了删除和修改相关的表对象。

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。

- “审计 DB2 活动”描述如何检测和监视不想要或不希望进行的数据存取。

附录

- “符合命名规则”提供在命名数据库和对象时要遵循的规则。
- “使用自动客户机重新路由”讨论自动重新路由客户机应用程序以及如何启用此支持。
- “使用轻量级目录访问协议（LDAP）目录服务”提供关于可以如何使用“LDAP 目录服务”的信息。
- “对多个数据库分区发出命令”讨论如何使用 *db2_all* 和 *rah* shell 脚本来将命令发送至分区数据库环境中的所有分区。
- “Windows 管理规范（WMI）支持”描述 DB2 如何支持此管理基础结构标准将各种硬件和软件管理系统集成在一起。还讨论了 DB2 如何与 WMI 集成。
- “使用 Windows NT 安全性”描述 DB2 通用数据库如何使用 Windows NT 安全性。
- “使用 Windows 性能监视器”提供关于向“Windows NT 性能监视器”注册 DB2 以及使用性能信息的信息。
- “使用 Windows 数据库分区服务器”提供关于可使用 Windows NT 或 Windows 2000 上的数据库分区服务器的实用程序的信息。
- “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。
- “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。

注: 已从此书除去两章。

所有关于移动数据的 DB2 实用程序的信息和来自 *Command Reference* 和 *Administrative API Reference* 的类似主题已合并到 *Data Movement Utilities Guide and Reference* 中。

Data Movement Utilities Guide and Reference 是有关这些主题的主要且单一的信息源。

要了解有关数据复制的更多信息，请参阅 *《IBM DB2 Information Integrator SQL 复制指南与参考》*。

所有关于备份与恢复数据的方法和工具的信息和来自 *Command Reference* 和 *Administrative API Reference* 的类似主题已合并到 *《数据恢复及高可用性指南与参考》* 中。

《数据恢复及高可用性指南与参考》 是有关这些主题的主要且单一的信息源。

第 1 部分 性能介绍

第 1 章 性能介绍

本章中的各节描述性能调整并为执行下列操作提供一些建议:

- 创建性能监视和调整计划
- 使用有关性能问题的用户信息
- 在初始性能调整时快速入门

性能元素

*性能*是计算机系统在特定工作负载下的运行情况。按照系统响应时间、吞吐量和可用性来测量性能。性能还受以下因素影响:

- 系统中可用的资源
- 如何充分利用和共享这些资源。

一般情况下，可调整您的系统来改进其成本和效益比率。具体目标可能包括:

- 处理更大的或更紧迫的工作负载，而不增加处理成本

  例如，增加工作负载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大的吞吐量，而不增加处理成本
- 降低处理成本，而不会降低对用户的服务

将性能从技术指标转换为经济指标比较困难。调整性能在用户时间和处理器时间方面一定会提高成本，因此在对项目进行调整前应衡量其相对于可能效益的成本。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。

其它效益是无形的，例如，由于响应更快而让用户更加满意。应考虑所有这些效益。

相关概念:

- 第 3 页的『性能调整准则』
- 第 6 页的『性能调整的快速启动技巧』

相关任务:

- 第 5 页的『开发性能提高流程』

性能调整准则

以下准则可帮助您制定一个调整性能的总体方案。

记住递减返回定律: 最大的性能收益通常来自于最初的努力。以后的更改通常只能产生越来越小的效益，并且需要更多努力。

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。如果有可能明显提高性能的话，则关键在于对某些作为影响响应时间的主要因素的资源的性能提高。

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它方面。通过一次更改一个，允许您使用基准程序来评估您是否需要进行更改。

按级别测量和重新配置: 和一次只应更改一个参数的理由一样，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
- 应用程序服务器和请求器
- 数据库管理器
- 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视和调整系统。

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。

在开始调整前执行回退过程: 正如前面所讲，某些调整可能产生意外的性能结果。如果此调整使性能降低，应撤销该调整，改试另一种调整。如果保存了以前的设置并可重新调用它，那么撤销不正确的信息将变得非常容易。

相关概念:

- 第 3 页的『性能元素』
- 第 6 页的『性能调整的快速启动技巧』

相关任务:

- 第 5 页的『开发性能提高流程』

性能调整过程

您应制定一个性能监视和调整计划，考虑用户输入，并认识到系统中调整的限制。

开发性能提高流程

性能提高流程是一种可重复的长期方法，用于监视和调整性能的各个方面。您和您的性能小组可以根据监视的结果调整数据库服务器的配置，并可对使用数据库服务器的应用程序进行更改。

性能监视和调整决策必须以您对使用数据的应用程序类型的了解程度和数据存取的模式为依据。不同类型的应用程序具有不同的性能要求。

将下面的性能提高流程概述作为提高性能的准则。

过程:

要开发性能提高流程:

1. 定义性能目标。
2. 为系统中的主要约束建立性能指示器。
3. 开发并执行性能监视方案。
4. 连续分析监视结果以确定哪些资源需要调整。
5. 每次进行一项调整。

   即使您认为需要调整多个资源，或数个调整选项可用于您要调整的资源，也要一次进行一项调整，从而确保所做的调整工作可以产生所需的效果。在某些点，调整数据库服务器和应用程序并不能提高性能。因此，您需要升级硬件。

实际的性能调整要求综合考虑系统资源。例如，为了提高 I/O 性能，您可能会增大缓冲池的容量，但是，缓冲池的容量越大，所需的内存也就越多，这将会降低其它方面的性能。

相关概念:

- 第 3 页的『性能元素』
- 第 3 页的『性能调整准则』
- 第 6 页的『性能调整的快速启动技巧』
- 第 6 页的『性能调整限制』
- 第 5 页的『用户可以提供的性能信息』

用户可以提供的性能信息

需要对系统进行调整的首个征兆可能是用户提出的意见。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视和调整，可听听用户的意见以改善性能。通常可提出几个简单的问题，来确定从哪里开始查找问题。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？
- 其它的用户有相同问题吗？这些用户是一两个人还是整个一组？
- 如果是一组用户遇到相同问题，他们是否与同一个局域网连接？
- 这些问题似乎与特定事务或应用程序相关吗？
- 注意到问题出现时有任何模式吗？例如，问题是否是在一天的特定时间发生，如午餐时间，或它持续的时间是长还是短？

相关概念:

- 第 3 页的『性能调整准则』

相关任务:

- 第 5 页的『开发性能提高流程』

性能调整限制

调整仅能够对系统的效率进行一定程度的更改。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。

例如，如果系统遇到性能瓶颈，通常可以通过调整来提高性能。如果已接近系统的性能限制，而用户数大约增加 10%，则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。

但是，有一个临界点，超过这个临界点再进行调整就没有用了。达到该临界点时，应考虑在环境的限制下修改您的目标和期望值。要显著提高性能，可能需要添加更多磁盘存储器、更快的 CPU、更多 CPU、更多主内存、更快的通信链路，或者它们的组合。

相关概念:

- 第 257 页的『数据库服务器容量的管理』

相关任务:

- 第 5 页的『开发性能提高流程』

性能调整的快速启动技巧

当启动 DB2® 的新实例时，应考虑有关基本配置的下列建议:

- 使用“控制中心”的“配置顾问程序”以获取有关系统的合理初始缺省值的建议。应当为唯一硬件环境调整与 DB2 一起交付的缺省值。

  收集有关站点硬件的信息，以便可以回答向导问题。您可以立即应用建议的配置参数设置，或让该向导基于您的回答创建脚本，并在以后运行该脚本。

  此脚本也提供最一般的调整参数的列表以供以后参考。
- 使用“控制中心”和“客户机配置助手”中的其它向导来执行与性能相关的管理任务。这些任务通常是通过花费较少时间和努力就可获得重要的性能提高的任务。

  其它向导有助于提高个别表和一般数据存取的性能。这些向导包括：“创建数据库”、“创建表、索引”和“配置多站点更新”向导。“健康中心”提供一组监视和调整工具。
- 使用控制中心中的“设计顾问程序”或者使用 `db2advis` 命令来了解哪些索引、具体查询表、多维群集表和数据库分区将提高查询性能。
- 使用 ACTIVATE DATABASE 命令来启动数据库。在分区数据库中，此命令激活所有分区上的数据库，并避免当第一个应用程序进行连接时初始化数据库所必需的启动时间。

  注: 如果使用 ACTIVATE DATABASE 命令，必须用 DEACTIVATE DATABASE 命令关闭数据库。与数据库断开连接的最后一个应用程序不会将它关闭。

- 查阅总结表，该表列示并简要地描述了可用于数据库管理器和每个数据库的每个配置参数。

  这些总结表包含一个列，该列指示调整参数是导致性能高、中、低还是无更改，是更好还是更坏。使用此表来查找您可以调整以获取最大性能提高的参数。

相关概念:

- 第 238 页的『数据库系统监视器信息』

相关参考:

- 第 294 页的『配置参数总结』

第 2 章 体系结构和进程

本章提供有关 DB2 体系结构和进程模式的一般信息。

DB2 体系结构和进程概述

有关 DB2® 体系结构和进程的一般信息可以帮助您了解对特定主题提供的详细信息。

下图提供了对 DB2 UDB 的体系结构和进程的一般概述。

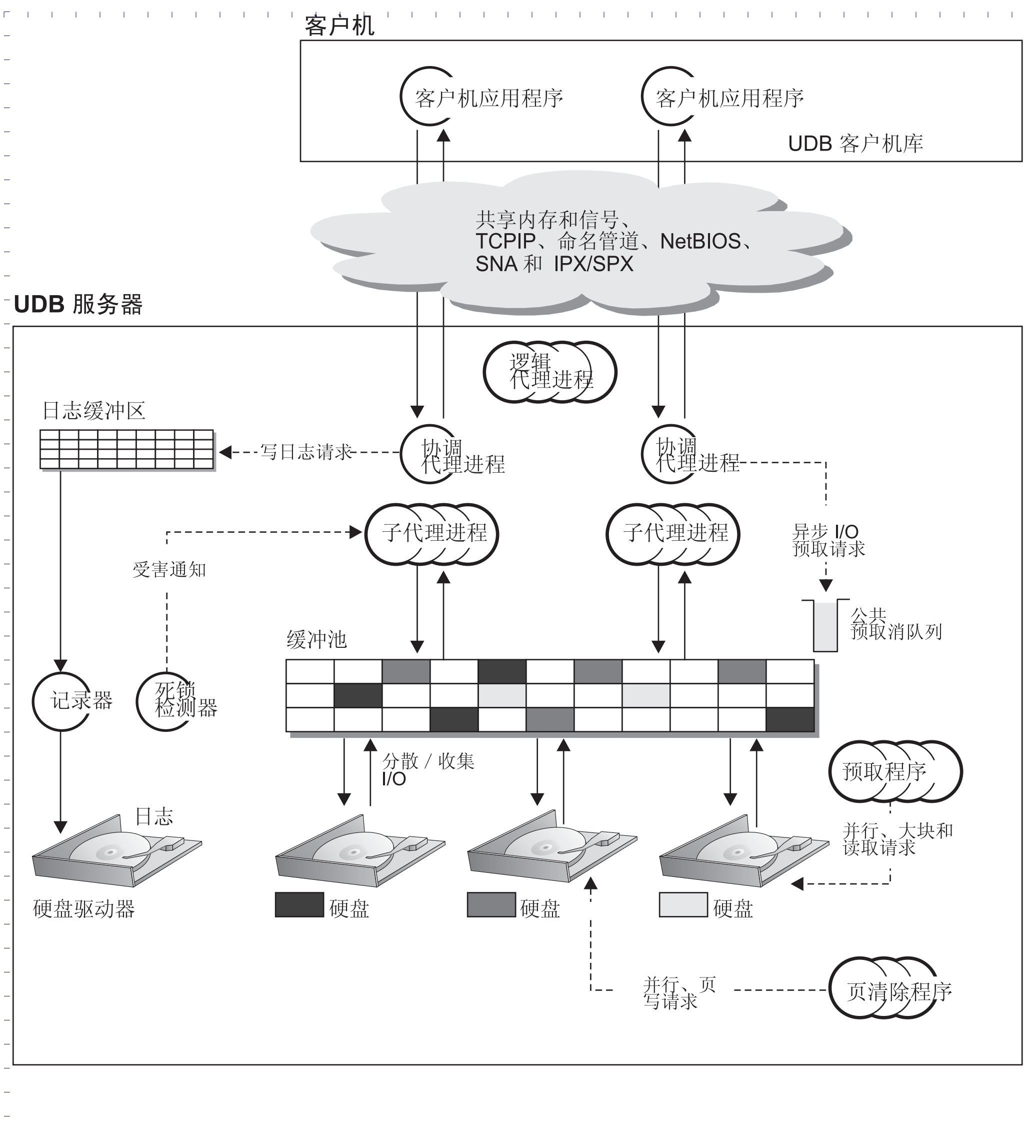


图 1. 体系结构和进程概述

在客户机端，本地或／和远程应用程序都与 DB2 Universal Database™（DB2 通用数据库）客户机库链接。本地客户机使用共享内存和信号进行通信；远程客户机使用一种协议（例如，命名管道（NPIPE）、TCP/IP、NetBIOS 或 SNA）进行通信。

在服务器端，活动由引擎可调度单元（EDU）控制。在本节中的所有图形中，EDU 显示为圆圈或许多组圆圈。在基于 Windows® 的平台上，EDU 是作为单个进程中的许多线程实现的，而在 UNIX® 上则是作为许多进程来实现的。DB2 代理进程是最常见的 EDU 类型。这些代理进程为应用程序执行大量 SQL 处理。其它常见的 EDU 是预取程序和页清除程序。

可以指定一组子代理进程来处理客户机应用程序请求。如果服务器驻留的机器具有多个处理器或是分区数据库的一部分，可以指定多个子代理进程。例如，在对称多处理技术（SMP）环境中，多个 SMP 子代理进程可以利用许多处理器。

所有代理进程和子代理进程都是使用特定入池算法来管理的，该算法将 EDU 的创建和破坏减至最少。

缓冲池是数据库服务器内存的一个区域，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，并可以在该处被修改。因为从内存存取数据可以比从磁盘存取数据快得多，所以缓冲池是数据库性能的重要因素。如果应用程序需要的多个数据存在于缓冲池，则存取数据所需的时间比在磁盘上查找要少。

缓冲池以及预取程序和页清除程序 EDU 的配置控制如何迅速存取数据以及对应用程序是如何容易地可用。

- **预取程序**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，如果没有数据预取程序，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。应用程序的代理进程将异步预读取请求发送至公共预取队列。当预取程序可用时，它们使用大块或散射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。如果具有多个磁盘来存储数据库数据，则可以将数据条带分割到多个磁盘上。条带分割数据让预取程序同时使用多个磁盘来检索数据。
- **页清除程序**将数据从缓冲池移回到磁盘中。页清除程序是独立于应用程序代理进程的后台 EDU。它们从不再使用的缓冲池中查找页并将这些页写入磁盘。页清除程序确保缓冲池中有空间供预取程序正在检索的页使用。

如果没有独立的预取程序和页清除程序 EDU，则应用程序代理进程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。

相关概念:

- 第 207 页的『将数据预取至缓冲池』
- 第 11 页的『应用程序之间的死锁』
- 第 12 页的『数据库目录和文件』
- 第 24 页的『日志处理』
- 第 25 页的『更新处理』
- 第 26 页的『客户机 － 服务器处理模型』
- 第 31 页的『内存管理』
- 第 235 页的『客户机连接的连接集中器改善』

相关参考:

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 346 页的『max_connections – 最大客户机连接数』

应用程序之间的死锁

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”。

当一个应用程序等待另一个应用程序释放数据上的锁定时，创建死锁。每一个等待的应用程序都会锁定另一个应用程序需要的数据。相互等待另一个应用程序释放所挂起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“另一个”应用程序释放挂起数据上的锁定为止。

下图对死锁进行了说明。

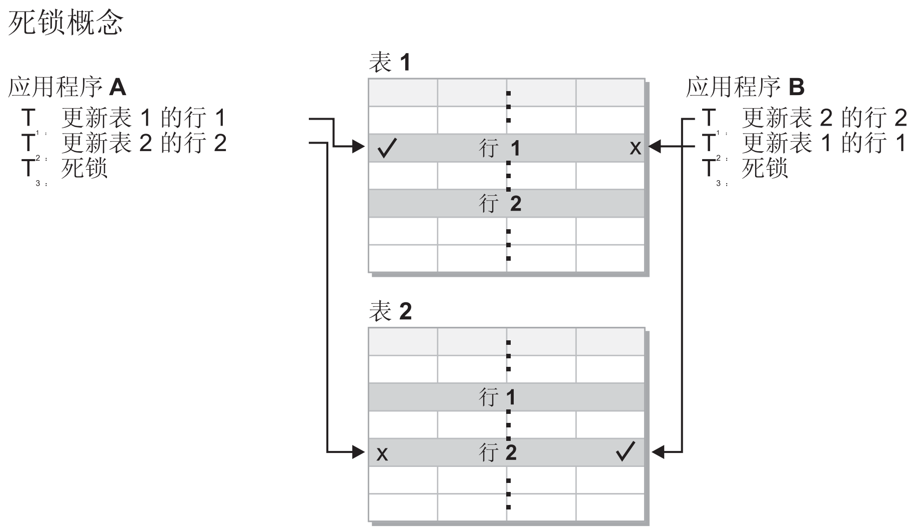


图 2. 死锁检测器

因为应用程序不会自动释放它们所需的数据的锁定，死锁检测器过程是中断死锁并允许应用程序进程继续所必需的。顾名思义，死锁检测器监视关于等待锁定的代理进程的信息。死锁检测器任意选择其中一个处于死锁状态的应用程序，并释放当前由“志愿”应用程序挂起的锁定。通过释放该应用程序的锁定，其它正在等待的应用程序所需的数据便可供使用。然后，等待的应用程序可以存取完成事务必需的数据。

相关概念:

- 第 46 页的『锁定和性能』
- 第 9 页的『DB2 体系结构和进程概述』

磁盘存储器概述

了解将数据如何存储在磁盘上可以帮助您调整 I/O。

磁盘存储器性能因素

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，考虑一些与磁盘存储器相关的蕴含内容。

磁盘存储器管理的四个方面影响性能:

- **存储器的划分**

  如何在索引和数据之间以及表空间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，但浪费的存储器是一种可用于在别处改善性能的资源。

- **磁盘 I/O 的分布**

  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，会影响数据库管理器从磁盘中检索信息的速度。

- **可用存储器的缺乏**

  达到可用存储器限制后，可能会降低整体性能。

相关概念:

- 第 231 页的『DMS 设备注意事项』
- 第 12 页的『数据库目录和文件』
- 第 14 页的『SMS 表空间』
- 第 15 页的『DMS 表空间』
- 第 17 页的『标准表的表和索引管理』
- 第 20 页的『MDC 表的表和索引管理』

数据库目录和文件

当创建一个数据库时，关于该数据库的信息（包括缺省信息）会存储在目录层次结构中。此分层目录结构的创建位置取决于您在 CREATE DATABASE 命令中提供的信息。如果您在创建数据库时未指定目录路径或驱动器的位置，则使用缺省位置。

建议您明确指出希望在何处创建数据库。

在 CREATE DATABASE 命令中指定的目录中，将创建一个使用实例名的子目录。这个子目录确保在同一目录下的不同实例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。这个子目录可以区分逻辑分区数据库环境中的分区。在节点名目录下面，将创建一个名为 SQL00001 的子目录。此子目录的名称使用了数据库标记并表示正在创建的数据库。SQL00001 包含与第一个创建的数据库以及随后创建的具有更高编号（SQL00002等等）的数据库相关联的对象。这些子目录可以区分在 CREATE DATABASE 命令中指定目录下的实例中创建的数据库。

目录结构如下所示:

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

数据库目录中包含下列作为 CREATE DATABASE 命令的一部分进行创建的文件。

- 文件 SQLBP.1 和 SQLBP.2 中包含缓冲池信息。这两个文件中具有相同的副本，从而提供备份。
- 文件 SQLSPCS.1 和 SQLSPCS.2 中包含表空间信息。这两个文件中具有相同的副本，从而提供备份。
- SQLDBCON 文件中包含数据库配置信息。切勿编辑此文件。要更改配置参数，请使用“控制中心”或命令行语句 UPDATE DATABASE CONFIGURATION 和 RESET DATABASE CONFIGURATION。
- DB2RHIST.ASC 历史文件及其备份 DB2RHIST.BAK 中包含关于备份、复原、表装入、表重组、表空间改变和其它数据库更改的历史信息。

  DB2TSCHNG.HIS 文件中包含日志文件级别的表空间更改的历史。对于每个日志文件，DB2TSCHG.HIS 中包含有助于标识日志文件影响哪些表空间的信息。表空间恢复使用此文件中的信息来确定在进行表空间恢复期间要处理哪些日志文件。可以在文本编辑器中检查这两个历史文件中的内容。
- 日志控制文件 SQLOGCTL.LFH 和 SQLOGMIR.LFH 中包含有关活动日志的信息。

  恢复处理过程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: 您应确保不要将日志子目录映射到用于存储数据的磁盘。这样，在磁盘发生问题时，只会影响到数据或日志，而不会同时影响这两者。由于日志文件与数据库容器不会争用同一磁盘磁头的移动，因此这可提供很多性能方面的好处。要更改日志子目录的位置，请更改 *newlogpath* 数据库配置参数。
- SQLINSLK 文件用于确保一个数据库只能由数据库管理器的一个实例使用。

在创建数据库的同时，还创建了详细死锁事件监视器。详细的死锁事件监视器文件存储在目录节点的数据库目录中。当事件监视器达到它要输出的最大文件数时，它将取消激活，并且将把一条消息写入通知日志中。这样可防止事件监视器消耗过多的磁盘空间。除去不再需要的输出文件将允许在下一次数据库激活时再次激活事件监视器。

SMS 数据库目录的附加信息

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

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

每个子目录或容器中都会创建一个名为 SQLTAG.NAM 的文件。这个文件可以标记正在使用中的子目录，因此在以后创建其它表空间时，不会尝试使用这些子目录。

此外，名为 SQL*.DAT 的文件中还存储有关子目录或容器包含的每个表的信息。星号（*）将被唯一的一组数字取代，用来识别每个表。对于每个 SQL*.DAT 文件，可能有一个或多个下列文件，这取决于表类型、表的重组状态或者表是否存在索引、LOB 或 LONG 字段:

- SQL*.BKM（如果是 MDC 表，则包含块分配信息）
- 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 文件的分配和可用空间信息）
- SQL*.INX（包含索引表数据）
- SQL*.IN1（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）

相关概念:

- 『SMS 和 DMS 表空间的比较』（*《管理指南: 计划》*）
- 第 231 页的『DMS 设备注意事项』
- 第 14 页的『SMS 表空间』
- 第 15 页的『DMS 表空间』
- 第 16 页的『DMS 表空间地址映射图表』
- 『了解恢复历史文件』（*《数据恢复及高可用性指南与参考》*）

相关参考:

- 『CREATE DATABASE Command』（*Command Reference*）

表空间概述

以下各节描述表空间并讨论了 DB2 中可用的两种类型的表空间，比较了它们的优点和缺点。

SMS 表空间

“系统管理空间”（SMS）表空间用于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扩展数据块进行条带分割。*扩展数据块*是对数据库定义的一组连续页。文件扩展名表示该文件中存储的数据的类型。为了在表空间中的所有容器上平均分发数据，表的起始扩展数据块以循环方式分布在所有容器上。如果数据库中包含许多容量较小的表，则这种扩展数据块分布特别重要。

在 SMS 表空间中，表的空间大小是按需分配的。分配的空间量取决于 multipage_alloc 数据库配置参数的设置。如果将此配置参数设置为 YES，则当需要空间时将分配完整的扩展数据块。否则，一次分配的空间将为一页。在版本 8.2 之前，该配置参数的缺省设置为 NO，它将导致一次分配一页。可以使用 db2empfa 工具来更改此缺省值。当运行 db2empfa 时，multipage_alloc 数据库配置参数将被设置为 YES。在版本 8.2 中，该配置参数的缺省设置为 YES，这意味着，缺省情况下，一次将分配一个完整的扩展数据块。

多页文件分配将只影响一个表的数据和索引部分。这意味着 .LF、.LB 和 .LBA 文件并不会一次扩展一个扩展数据块。

在 SMS 表空间中，当将单个容器中的所有空间都分配给表时，就认为该表空间“已满”，即使其它容器中还有剩余空间。仅当分区中没有任何容器时，才能向该分区中的 SMS 表空间添加容器。

注: SMS 表空间可以利用文件系统预取和高速缓存功能。

相关概念:

- 『表空间设计』（*《管理指南：计划》*）
- 『SMS 和 DMS 表空间的比较』（*《管理指南：计划》*）

相关任务:

- 『将容器添加至分区上的 SMS 表空间』（*《管理指南：实现》*）

相关参考:

- 第 394 页的『multipage_alloc － 启用的多页文件分配』
- 『db2empfa - Enable Multipage File Allocation Command』（*Command Reference*）

DMS 表空间

数据库管理器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® 数据库管理器管理。对于 SMS 表空间和容器，DMS 表空间和数据库管理器使用“按扩展数据块进行条带分割”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：重要的是扩展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。根据文件系统上其它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。但是，对于 DMS，因为数据库管理器直接与磁盘打交道，所以它可以确保这些页在物理上是连续的。

注: 与 SMS 表空间相似，DMS 文件容器可以利用文件系统预取和高速缓存。然而，DMS 表空间不能。

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：裸设备和文件。当使用文件容器时，数据库管理器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。当使用裸设备容器时，数据库管理器控制整个设备，并总是确保扩展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。但是，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 表空间可以使用其它容器中的任何可用的可用空间。

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。

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。当创建表空间时，扩展数据块大小被定义为许多连续页。扩展数据块是表空间中的空间分配单位。只有一个表或其它对象（例如，索引）能够使用任何一个扩展数据块中的页。将逻辑表空间地址映射中的扩展数据块分配给表空间中创建的所有对象。扩展数据块分配通过“空间映射页”（SMP）进行管理。

逻辑表空间地址映射中的第一个扩展数据块是包含内部控制信息的表空间的头部。第二个扩展数据块是表空间的“空间映射页”（SMP）的第一个扩展数据块。SMP 扩展数据块以固定间隔方式分布在表空间中。每个 SMP 扩展数据块仅仅是当前 SMP 扩展数据块到下一 SMP 扩展数据块的扩展数据块位映射。位映射用来跟踪正在使用哪些中间扩展数据块。

SMP 后面的一个扩展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“扩展数据块映射页”（EMP）扩展数据块的位置。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。

相关概念:

- 『表空间设计』（*《管理指南: 计划》*）
- 『SMS 和 DMS 表空间的比较』（*《管理指南: 计划》*）
- 第 231 页的『DMS 设备注意事项』
- 第 12 页的『数据库目录和文件』
- 第 14 页的『SMS 表空间』
- 第 16 页的『DMS 表空间地址映射图表』

相关任务:

- 『将容器添加至 DMS 表空间』（*《管理指南: 实现》*）

相关参考:

- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

DMS 表空间地址映射图表

下图显示了 DMS 表空间的逻辑地址映射。

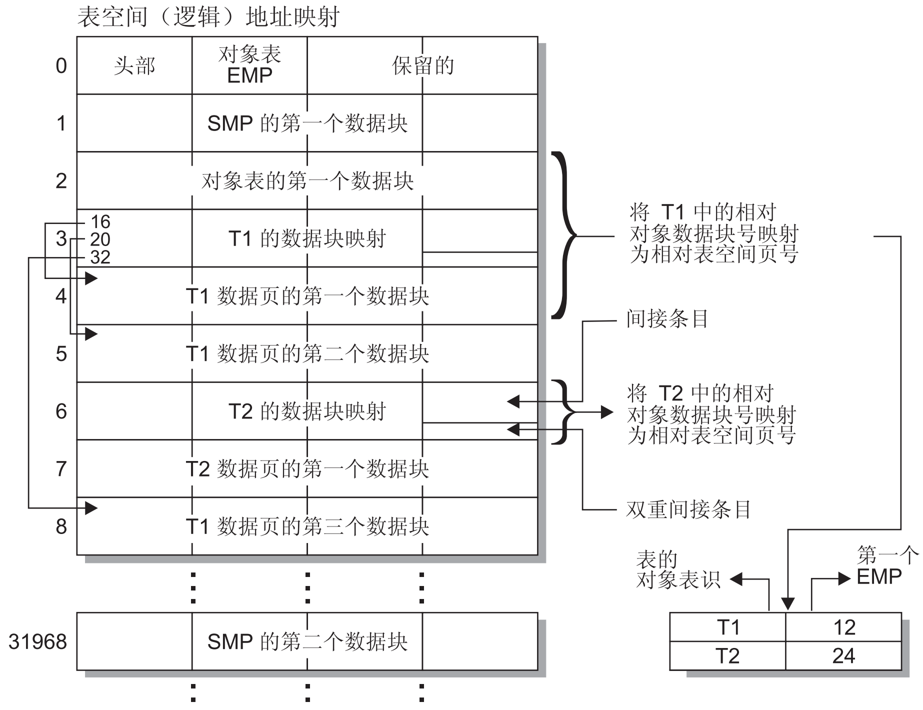


图 3. DMS 表空间

对象表是一个内部关系表，它将对象标识映射至该表的第一个 EMP 扩展数据块的位置。这个 EMP 扩展数据块直接或间接地映射出该对象中的所有扩展数据块。每个 EMP 都包含一连串的条目。每一条目都将对象相对扩展数据块号映射至该对象扩展数据块所在的相对表空间页号。直接 EMP 条目直接将对象相对地址映射至相对表空间地址。第一个 EMP 扩展数据块中的最后一个 EMP 页包含间接条目。间接 EMP 条目映射至 EMP 页，EMP 页然后映射至对象页。第一个 EMP 扩展数据块中最后一个 EMP 页中的最后 16 条目包含双重间接条目。

逻辑表空间地址映射中的扩展数据块以循环顺序在与该表空间相关联的容器上进行条带分割。

相关概念:

- 第 231 页的『DMS 设备注意事项』
- 第 12 页的『磁盘存储器性能因素』
- 第 15 页的『DMS 表空间』

表和索引

以下各节讨论标准和 MDC 表以及这些表上的索引的管理。

标准表的表和索引管理

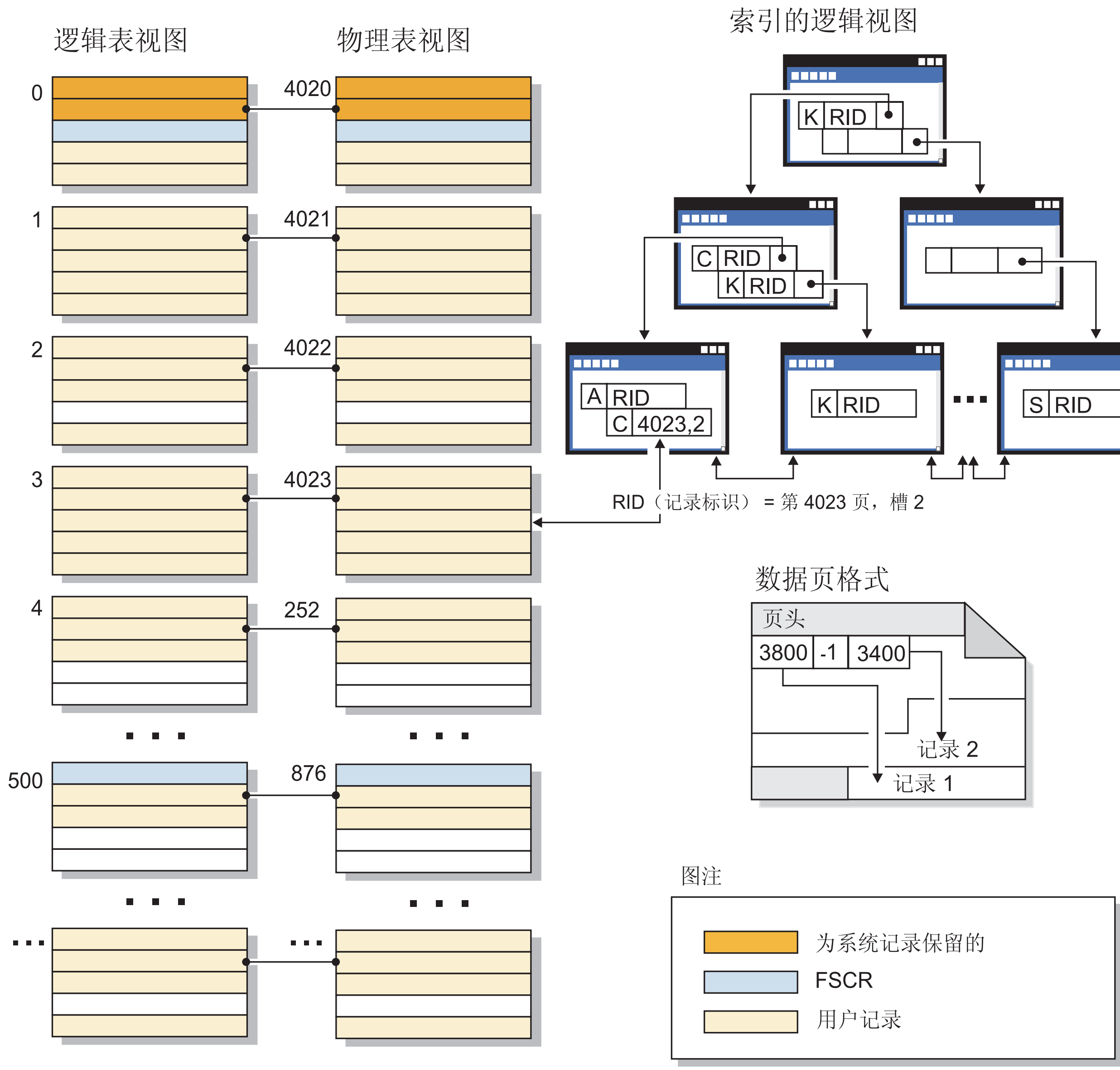


图 4. 标准表的逻辑表、记录和索引结构

在标准表中，数据在逻辑上是按数据页的列表来组织的。这些数据页根据表空间的扩展数据块大小在逻辑上分组在一起。例如，如果扩展数据块大小是 4，第 0 至 3 页是第一个扩展数据块的一部分，则第 4 至 7 页是第二个扩展数据块的一部分，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。但是，少数页包括特殊的内部记录，DB2® 使用这些记录来管理表。例如，在标准表中，每个第 500 个数据页上都有一个“可用空间控制记录”（FSCR）。这些记录映射下面每 500 个数据页（直到下一 FSCR 为止）上的可供新记录使用的可用空间。当将记录插入表时，将使用这部分可用的可用空间。

在逻辑上，索引页组织成 B 树，这可以有效地在表中定位带有给定键值的记录。索引页上的项数不是固定的，但依赖于键的大小。对于 DMS 表空间中的表，索引页中的记录

标识（RID）使用相对表空间页号，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“扩展数据块映射页”（EMP）来进行映射。

每个数据页都具有相同的格式。每个数据页开头都有一个页头。在页头后面，有一个槽目录。槽目录中的每一条目都与该页中的另一个记录相对应。该条目本身是数据页中记录开始位置的字节位移。值为 -1 的条目与已删除的记录相对应。

记录标识和页

记录标识（RID）由三个字节的页号及随后的一个字节的槽号组成。类型 2 索引记录也包含称为 ridFlag 的附加字节。该 ridFlag 存储有关索引中密钥状态的信息，例如，此密钥是否标记为已删除。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。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

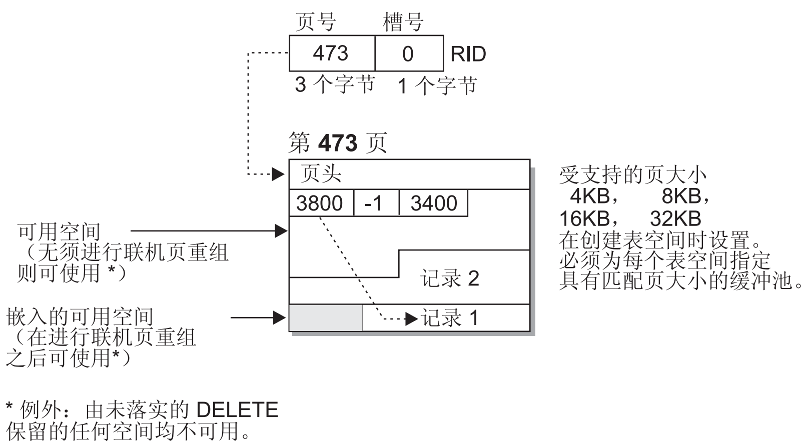


图 5. 数据页和记录标识（RID）格式

重组表时，实际删除记录后在页上留下的嵌入可用空间被转换成可使用的可用空间。根据记录在数据页上的移动重新定义 RID，以利用可使用的可用空间。

DB2 支持不同的页大小。对于有可能连续存取行的工作负载，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载，使用较小的页大小。例如，OLTP 环境中使用的便是随机存取。

标准表中的索引管理

DB2 索引使用最优的 B 树实现，它基于一种高效率且高并发性的索引管理方法，该方法使用预写日志记录。

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。索引页分割通常刚好是对半的，但在高键页上除外，在那些页上，使用 90/10 分割。即，索引键的高 10% 放在新页上。这种类型的索引页分割对于特定工作负载而言非常有用，这些工作负载的 INSERT 请求通常使用新的高键完成。

从版本 8.1 开始，DB2 就使用类型 2 索引。如果从 DB2 的较早版本进行迁移，则将使用类型 1 和类型 2 索引，直到重组索引或执行将类型 1 索引转换为类型 2 的其它操作。索引类型决定如何从索引页除去删除键。

- 对于类型 1 索引，键删除期间从索引页除去键，且当除去一页上的最后一个索引键时，便从索引中释放该页。
- 对于类型 2 索引，仅当对表具有 X 锁定时，才在删除键期间从页除去索引键。如果键不能立即除去，则标记为已删除而稍后除去。有关更多信息，参阅描述类型 2 索引的节。

当创建索引时，如果已通过将 MINPCTUSED 子句设置为大于零的值来启用联机索引整理碎片时，则可以联机合并叶子页。您指定的值是在索引叶子页上使用的空间的最小百分比的阈值。从索引页除去键之后，如果在页上使用的空间的百分比等于或小于所给的值，则数据库管理员尝试将剩余键与相邻页上的键合并。如果有足够的空间，则执行合并，并删除一个索引叶子页。联机索引整理碎片可以改进空间重用，但是如果 MINPCTUSED 值太高，则尝试合并所花的时间会增加，并且成功的可能性更小。对此子句建议的值是百分之五十或更低。

注: 因为仅当从类型 2 索引中的索引页除去键时联机整理碎片才发生，所以如果键仅仅标记为已删除但并未从页除去时，它不发生。

CREATE INDEX 语句的 INCLUDE 子句允许除了包括键列之外，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。但是，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。更新包含列的维护成本小于更新键列，但大于索引中未显示的更新列。只使用键列，而不使用所包括的列来对索引 B 树进行排序。

相关概念:

- 『数据库对象的空间需求』（*《管理指南: 计划》*）
- 『为表选择表空间时的注意事项』（*《管理指南: 计划》*）
- 『设计多维群集（MDC）表』（*《管理指南: 计划》*）
- 第 20 页的『MDC 表的表和索引管理』
- 『多维群集（MDC）表创建、布置和使用』（*《管理指南: 计划》*）
- 第 227 页的『索引清除和维护』
- 第 25 页的『插入处理』

MDC 表的表和索引管理

多维群集（MDC）表的表和索引组织基于与标准表组织相同的逻辑结构。与标准表类似， MDC 表是按页组织的，这些页包含分为许多列的数据行，每页上的行由行标识（RID）标识。但是，MDC 表的各页还分组为按扩展数据块大小计算的块。例如，下图显示扩展数据块大小为 4 的表，前四页（从 0 到 3 编号）是表中的第一个块。下一组页（从 4 到 7 编号）是表中的第二个块。

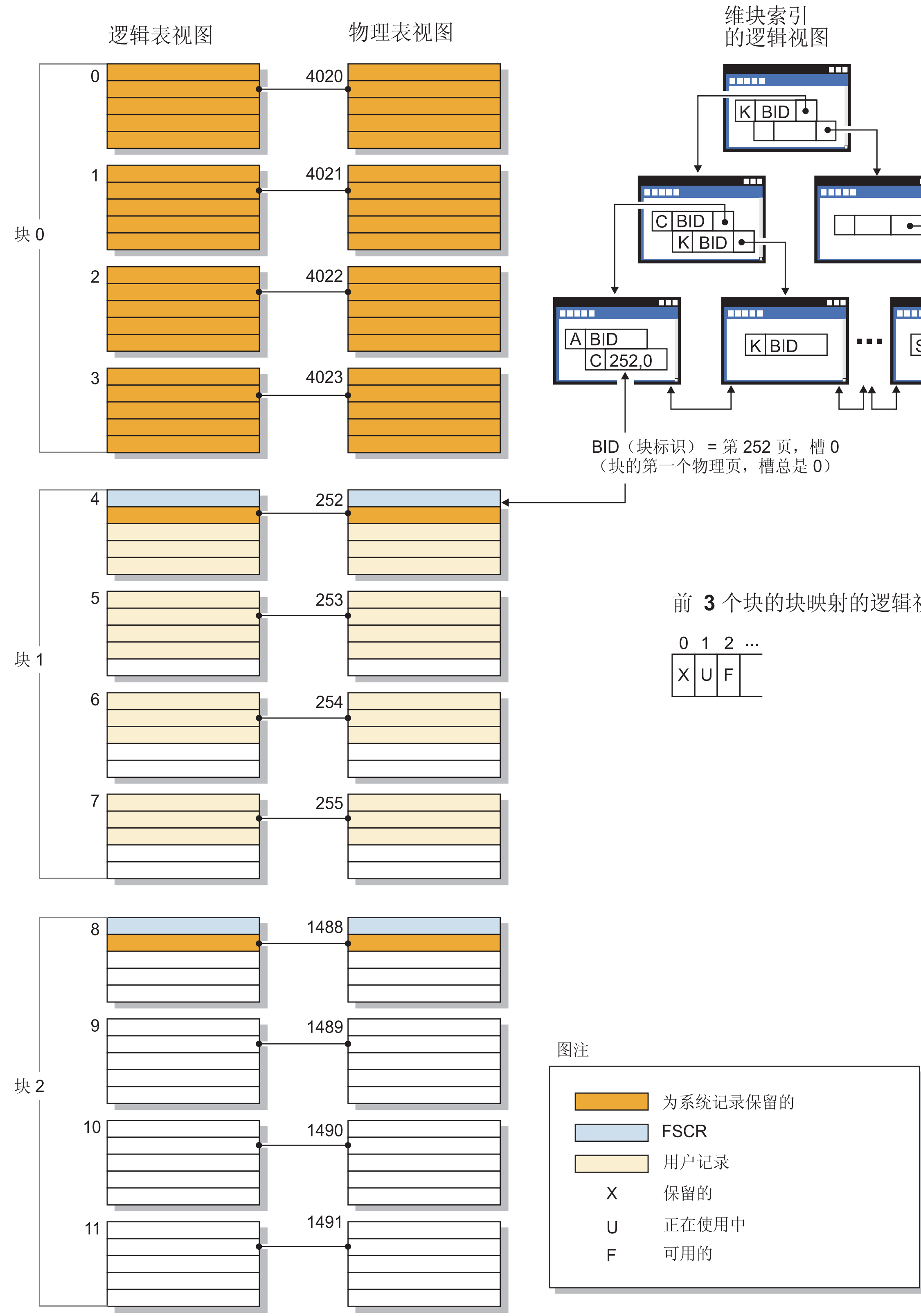


图 6. MDC 表的逻辑表、记录和索引结构。

第一个块包含 DB2® 用来管理表的特殊内部记录，包括可用空间控制记录（FSCR）。在后续块中，第一页包含 FSCR。FSCR 为块中每页上存在的新记录映射可用空间。当将记录插入表时，将使用这部分可用的可用空间。

顾名思义，MDC 表将对多个维上的数据进行群集。每个维都由您在 CREATE TABLE 语句的 ORGANIZE BY DIMENSIONS 子句中指定一列或一组列确定。当创建 MDC 表时，会自动创建以下两种索引:

- 维块索引，它包含指向单个维的每个已占用块的指针。
- 组合块索引，它包含所有维键列。组合块索引用于在插入和更新活动期间维护群集。

当优化器确定特定查询的最有效的存取方案时，它考虑利用维块索引的存取方案。当查询在维值上具有谓词时，优化器可以使用维块索引来标识包含这些值的扩展数据块，并从扩展数据块中进行访存。因为扩展数据块是磁盘中物理上相邻的页，所以这会导致更有效的性能并使 I/O 次数减少到最低。

另外，如果数据存取方案的分析指示特定 RID 索引将提高查询性能，则可以创建这样的索引。

除了维块索引和组合块索引以外，MDC 表还维护包含位图的块映射，该位图指示每个块的可用性状态。在位图列表中编码下列属性:

- X（保留）: 第一个块仅包含表的系统信息。
- U（正在使用）: 此块已被使用并与维块索引相关联
- L（(已装入）: 已通过当前装入操作装入此块
- C（检查约束）: 此块由装入操作设置，用来指定装入期间的增量约束检查。
- T（刷新表）: 此块由装入操作设置，用来指定需要 AST 维护。
- F（空闲）: 如果没有设置其它属性，该块被认为是空闲的。

因为每个块都在块映射文件中具有一个条目，所以文件随表的增长而增长。作为单独对象存储此文件。在 SMS 表空间中，它是一个新的文件类型。在 DMS 表空间中，它在对象表中具有新的对象描述符。

相关概念:

- 『数据库对象的空间需求』（*《管理指南: 计划》*）
- 『设计多维群集（MDC）表』（*《管理指南: 计划》*）
- 『多维群集（MDC）表创建、布置和使用』（*《管理指南: 计划》*）
- 第 25 页的『插入处理』

索引结构

数据库管理器使用 B+ 树结构进行索引存储。一个 B+ 树有一层或多层，如下图中所示，其中，RID 表示行标识:

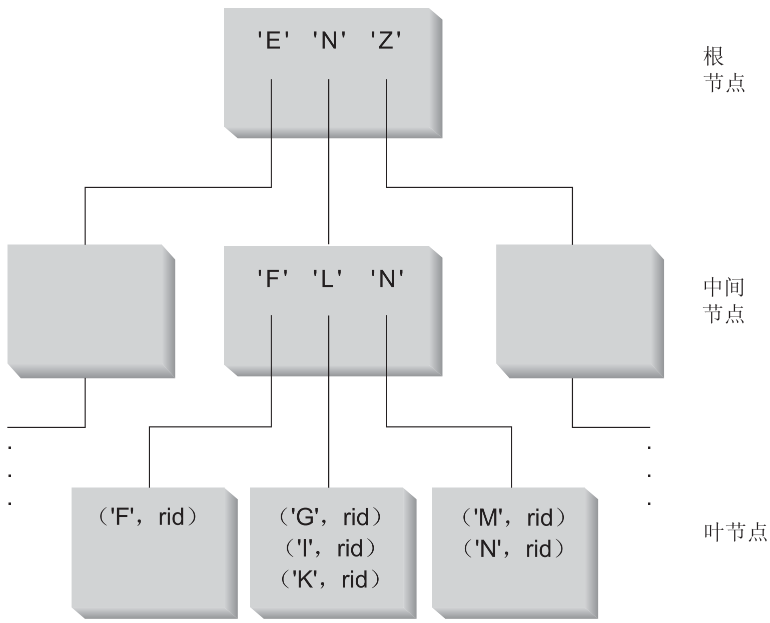


图 7. B+ 树结构

顶层称为*根节点*。底层由*叶节点*组成，底层存储了索引键值，并有一个指针指向包含键值的表中的行。根节点层和叶节点层之间的那些层称为*中间节点*。

当它查找特定的索引键值时，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键。每个键的值是下一层中对应节点的最大现有键值。例如，如果一个索引有三层（如图中所示），那么，要查找一个索引键值，索引管理器搜索根节点，以查找大于或等于要查找的键的第一个键值。根节点键指向特定的中间节点。索引管理器遵循此过程遍历中间节点，直到它找到包含它需要的索引键的叶节点。

该图显示要查找的键是“I”。在根节点中大于或等于“I”的第一个键是“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”的第一个键是“L”。它指向发现有“I”的索引键及其对应的行标识的特定叶节点。行标识标识基本表中的对应行。叶节点层也可以包含指向上一层叶节点的指针。这些指针允许索引管理器在它找到范围内的某个值时按任一方向扫描叶节点以检索某个范围内的值。仅当使用 ALLOW REVERSE SCANS 子句创建了索引时，才可能有按任一方向扫描的能力。

对于多维群集（MDC）表，对给表指定的每个群集维自动创建块索引。也创建附加组合块索引，该索引包含每列中涉及表中任何维的键部分。这些索引包含指向块标识（BID）的指针代替 RID，并提供数据存取改进。

在 DB2® 版本 8.1 和更高版本中，索引可以具有类型 1 或类型 2。*类型 1 索引*是较旧的索引样式。在 DB2 较早版本中创建的索引具有此种类型。

类型 2 索引大于类型 1 索引并提供最小化的下一键锁定的功能部件。为类型 2 索引的叶子页中每个 RID 存储的单字节 *ridFlag* 字节用来将 RID 标记为在逻辑上已删除，以便以后可以物理上除去它。对于包含在索引中的每个可变长度列，一个附加字节存储列值的实际长度。因为某些键可以标记为删除，但尚未在物理上从索引页除去，所以类型 2 索引也可以大于类型 1 索引。在落实 DELETE 或 UPDATE 事务之后，可以清除标记为删除的键。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 229 页的『索引重组』
- 第 230 页的『联机索引整理碎片』

进程

以下各节提供 DB2 进程的一般描述。

日志处理

所有数据库都维护用于记录数据库更改情况的日志文件。有两种日志记录策略可供选择:

- 循环日志记录，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，指的是日志文件由日志记录填满时会被存档。然后使用新的日志文件来存放日志记录。保留日志文件可以启用**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。

无论哪种日志记录策略，对一般数据和索引页所作的所有更改都会被写入日志缓冲区。日志缓冲区中的数据由记录器进程写入磁盘。在下列情况下，查询处理必须等待日志数据写入磁盘后才能进行:

- 运行 COMMIT 语句时
- 在将相应数据页写入磁盘之前，因为 DB2® 需要使用预写日志记录。预写日志记录的好处是当执行 COMMIT 语句完成事务之后，并非所有更改的数据和索引页都需要写入磁盘。
- 在更改元数据（大多数是通过执行 DDL 语句产生的）之前
- 在将日志记录写入日志缓冲区时，如果日志缓冲区已满

DB2 以这种方法管理向磁盘写入日志数据的目的是尽可能地缩短处理延迟时间。在发生许多较小的并发事务的环境中，大多数处理延迟是由 COMMIT 语句造成的，因为此语句必须等待日志数据写入磁盘后才能进行。因此，记录器进程频繁地将少量日志数据写入磁盘会造成大量处理延迟，另外一些延迟是由日志 I/O 开销造成的。为平衡此类日志记录延迟导致的应用程序响应时间，请将 *mincommit* 数据库配置参数设置为大于 1 的值。虽然这样设置可能会对某些应用程序的 COMMIT 命令造成更长的延迟时间，但是这会使系统可在一次操作中写入更多的日志数据。

对大对象（LOB）和 LONG VARCHAR 的更改通过镜像页面调度进行跟踪。除非指定日志保留，并且在使用 CREATE TABLE 语句创建表时没有使用 NOT LOGGED 子句定义 LOB 列，否则不记录 LOB 列更改。如同一般数据页一样，对 LONG 或 LOB 数据类型的分配页的更改也会被记录下来。

相关概念:

- 第 25 页的『更新处理』
- 第 26 页的『客户机 － 服务器处理模型』

相关参考:

- 第 369 页的『mincommit － 对组的落实次数』

插入处理

当 SQL 语句使用 INSERT 来将新信息放置在表中时，INSERT 搜索算法首先搜索“可用空间控制记录”（FSCR）来查找具有足够空间的页。然而，即使 FSCR 指示某页具有足够的可用空间，该空间也可能因为被另一个事务的未落实的 DELETE 语句保留而不可用。要确保未落实的可用空间是可用的，应经常“落实”事务。

DB2MAXFSCRSEARCH 注册表变量的设置确定表中为 INSERT 而搜索的 FSCR 数目。此注册表变量的缺省值是 5。如果在指定数目的 FSCR 中找不到空间，在表的末尾处追加插入的记录。要优化 INSERT 速度，还可能会将后续记录追加至表末尾，直到填满了两个扩展数据块为止。在填满了两个扩展数据块之后，下一个 INSERT 在上一搜索结束处的 FSCR 继续搜索。

注：要优化 INSERT 速度（可能的代价是表增长得更加快），将 DB2MAXFSCRSEARCH 注册表变量设置为较小的数。要优化空间重用（可能的代价是 INSERT 速度减慢），将 DB2MAXFSCRSEARCH 设置为更大的数。

以这种方式搜索整个表中的所有 FSCR 后，追加要插入的记录而不进行附加搜索。另外，在表中某处（例如，在 DELETE 之后）创建空间之前，也不再执行使用 FSCR 的搜索。

有两个其它的 INSERT 算法选项，如下所示:

- 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。使用 ALTER TABLE APPEND ON 语句启用此选项，它可以提高只会增长的表（例如，日志）的性能。
- 在表上定义群集索引。

  在这种情况下，数据库管理器尝试将记录插入到带有类似索引键值的其它记录所在的页上。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，则使用如上所描述的 FSCR 搜索算法，除非使用最差匹配方法，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可用空间的页。此方法为带有此键值的行建立新的群集区。

  当对表定义群集索引时，请在装入或重组表之前使用 ALTER TABLE... PCTFREE。PCTFREE 子句指定在装入并重组之后应该留在表的数据页上的可用空间的百分比。这提高了群集索引操作能在适当的页上找到可用空间的可能性。

相关概念:

- 第 17 页的『标准表的表和索引管理』
- 第 25 页的『更新处理』
- 第 20 页的『MDC 表的表和索引管理』

更新处理

当代理进程更新页时，数据库管理器使用以下协议来将事务必需的 I/O 最小化，并确保可恢复性。

1. 使用互斥锁定将要更新的页锁定住。将描述如何重做和撤销更改的日志记录写入日志缓冲区。作为此操作的一部分，获取日志序列号（LSN），并将其存储在正在更新的页的页头中。
2. 对该页进行更改。
3. 取消锁定该页。

   因为对该页的更改尚未被写入磁盘，所以该页被认为是“脏”的。
4. 日志缓冲区已更新。

   将日志缓冲区中的数据和"脏"数据页强制写入磁盘。

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，或延迟至限制的恢复时间。具体地说，“脏”页在下面这些时候被强制写入磁盘:

- 当另一个代理进程选择它作为干扰页的时候。
- 当由于下列原因造成页清除程序对页执行操作时:
  - 另一个代理进程选择它作为干扰页。
  - 超过 *chngpgs_thresh* 数据库配置参数百分比值。当超过此值时，异步页清除程序便被唤醒，并将更改过的页写入磁盘。

    如果启用了主动清除页，如果启用了主动清除页，则此值不合适，并且不会触发页清除。
  - 超过 *softmax* 数据库配置参数百分比值。一旦超过此值，异步页清除程序便被唤醒，并将更改过的页写入磁盘。

    如果对数据库启用了主动清除页，并且已经为数据库正确配置了页清除程序数，则始终都不会超过该值。
  - 憎恨列表中干净页的数目下降得太低。只有采用主动清除页方法时页清除程序才对此情况起作用。
  - 脏页当前添加至或者计划添加至 LSNGAP 条件。只有采用主动清除页方法时页清除程序才对此情况起作用。
- 当作为调用了 NOT LOGGED INITIALLY 子句并发出了 COMMIT 语句的表的一部分更新该页时。当执行 COMMIT 语句时，所有已更改的页都被清仓至磁盘，以确保可恢复性。

相关概念:

- 第 24 页的『日志处理』
- 第 26 页的『客户机 － 服务器处理模型』

相关参考:

- 第 371 页的『softmax － 恢复范围和软检查点时间间隔』
- 第 338 页的『chngpgs_thresh － 已更改页数的阈值』

客户机 － 服务器处理模型

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。

注: DB2® 管理客户机连接的方式取决于连接集中器是打开还是关闭。当 *max_connections* 数据库管理器配置参数的设置值大于 *max_coordagents* 配置参数的设置值时，连接集中器处于打开状态。

- 如果连接集中器处于“关闭”状态，每个客户机应用程序均会分配一个称为*协调代理进程*的唯一 EDU，它可以协调该应用程序的处理并与之通信。
- 如果连接集中器处于“打开”状态，每个协调代理进程均可一次管理多个客户机连接， 并可协调其它工作程序代理进程以完成这项工作。对于具有许多相关瞬态连接的因特网应用程序或具有许多相关小事务的类似应用程序，连接集中器通过允许进行更多的客户机应用程序连接来提高系统的性能。它还可以减少每个连接对系统资源的使用。

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”，而在 Windows® NT 平台上称为“线程”。

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器之间建立一种通信方法。

在下图中的 A1 处，本地客户机首先通过 db2ipccm 建立通信。在 A2 处，db2ipccm 使用 db2agent EDU，该 EDU 成为来自本地客户机的应用程序请求的协调代理进程。然后该协调代理进程联系 A3 处的客户机应用程序以在客户机应用程序和协调程序之间建立共享内存通信。本地客户机上的应用程序与 A4 处的数据库相连。

在下图中的 B1 处，远程客户机通过 db2tcpcm EDU 建立通信。如果选择其它任何通信协议，则使用相应的通信管理器。db2tcpcm EDU 在客户机应用程序和 db2tcpcm 之间建立 TCP/IP 通信。然后，它与 B2 处的 db2agent（此时 db2agent 成为该应用程序的协调代理进程）一起工作并将连接传送至此代理进程。在 B3 处，协调代理进程与远程客户机应用程序联系并与数据库连接。

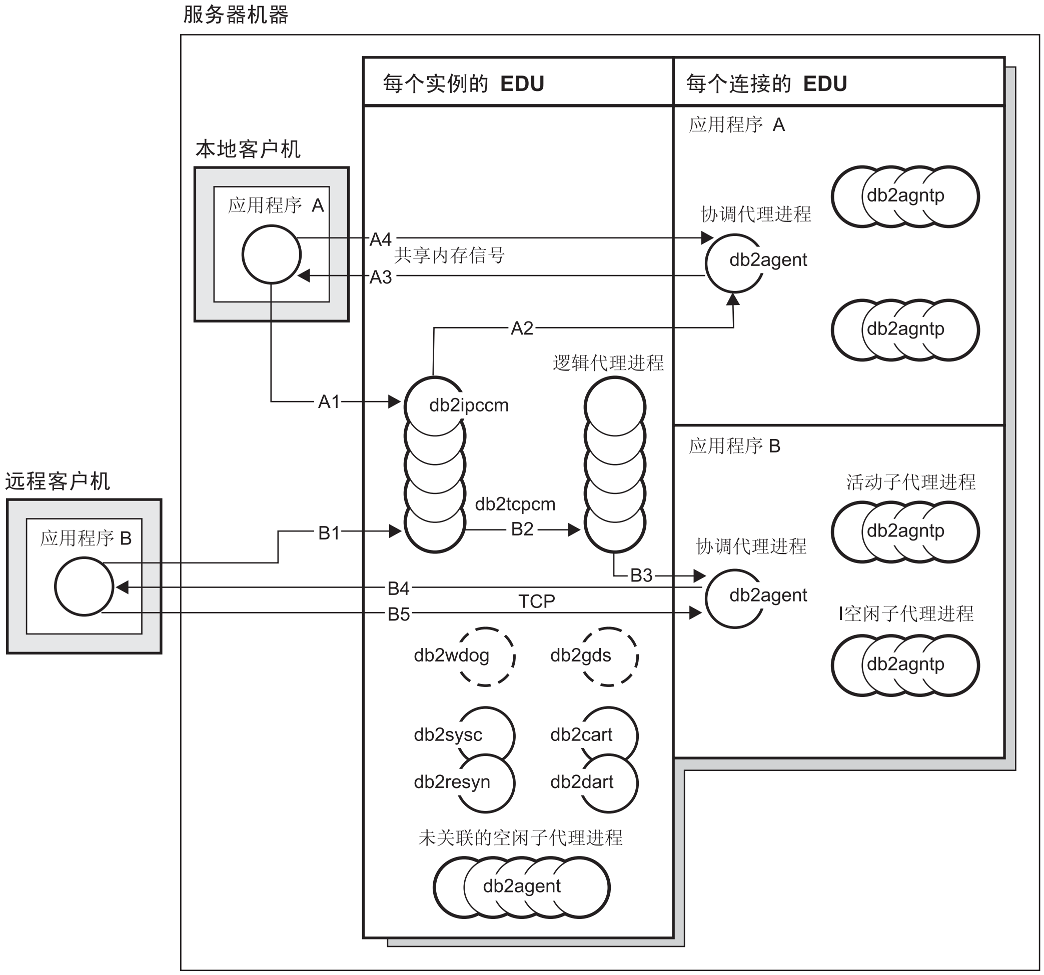


图 8. 进程模型概述

在此图中要注意的其它事项:

- 工作程序代理进程执行应用程序请求。
- 工作程序代理进程共有四种：活动协调代理进程、活动子代理进程、关联子代理进程和空闲代理进程。
- 每个客户机连接均与活动协调代理进程相链接。
- 在分区数据库环境以及启用的分区内并行性环境中，协调代理进程将数据库请求分配给子代理进程（db2agntp）。子代理进程为应用程序执行请求。
- 有一个代理进程池（db2agent），空闲代理进程和池中代理进程在那里等待新工作。
- 其它 EDU 管理客户机连接、日志、两阶段落实（COMMIT）、备份和复原任务以及其它任务。

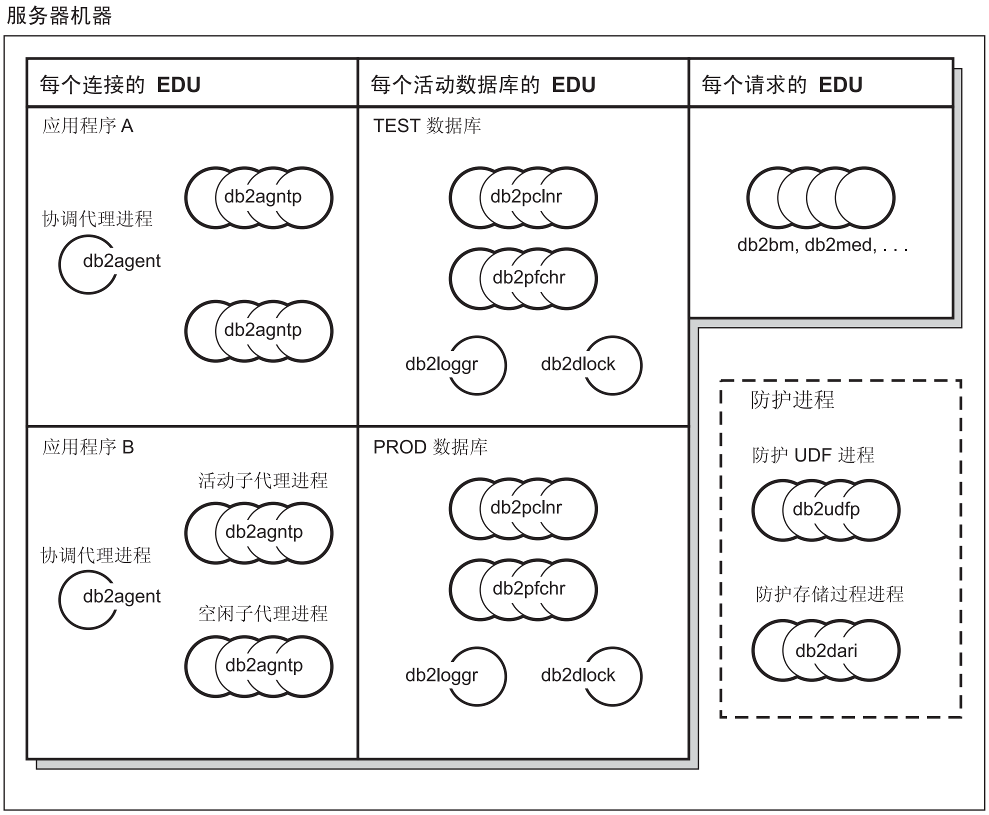


图 9. 进程模型（第二部分）

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）。每个活动数据库均有自身的预取程序（db2pfchr）和页清除程序（db2pclnr）的共享池，并有自身的记录器（db2loggr）和死锁检测器（db2dlock）。

对受防护的用户定义的函数（UDF）和存储过程（图中并未显示）进行管理以尽可能减少与其创建和删除相关联的开销。*keepfenced* 数据库管理器配置参数的缺省值为“YES”，此值使下一次调用的存储过程可以重复使用存储过程进程。

注: 为获得更好的性能，不受防护的 UDF 和存储过程直接在代理进程的地址空间运行。但是，由于它们对代理进程的地址空间的存取权不受限制，因此在使用它们之前，需要对它们进行严格的测试。

多分区处理模型是单一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。下图显示了单一分区处理模型（如前两图中所示）与多分区处理模型之间存在的相似性与差异。

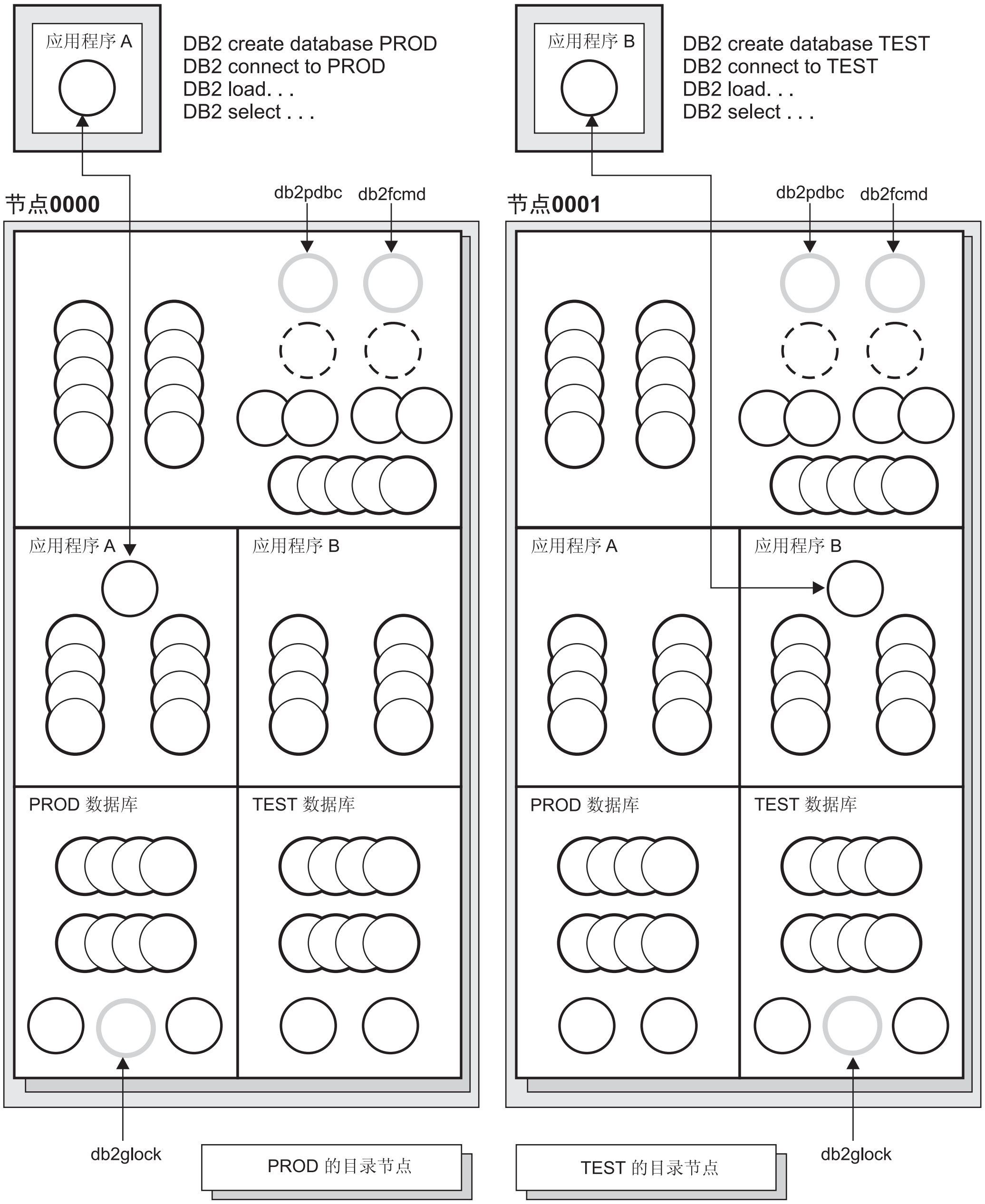


图 10. 进程模型和多分区

在单一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，其中一个分区为目录节点。该目录存放与数据库中的对象相关的所有信息。

如上图所示，由于应用程序 A 在 Node0000 上创建 PROD 数据库，因此在该节点上创建 PROD 数据库的目录。同样，由于应用程序 B 在 Node0001 上创建 TEST 数据库，因此在该节点上创建 TEST 数据库的目录。您可能想在您的系统环境中的不同节点上创建数据库，从而将与每个数据库的目录相关的多余活动平均分布在这些节点上。

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。该 EDU 可以控制活动数据库所在的各个节点上的全局死锁。

每个来自应用程序的 CONNECT 都由与协调代理进程相关联的连接表示，以便对该连接进行处理。*协调代理进程*是指与应用程序相互通信（即接收请求和发送回应）的代理进程。它自身可以处理请求，也可以协调多个子代理进程对请求进行处理。协调代理进程所在的分区称为该应用程序的*协调程序节点*。协调程序节点还可以使用 SET CLIENT CONNECT_NODE 命令进行设置。

协调程序节点将来自应用程序的部分数据库请求发送至其它分区中的子代理进程；然后协调程序节点将其它分区返回的所有结果合并，并发送回给应用程序。

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。

相关概念:

- 第 9 页的『DB2 体系结构和进程概述』
- 第 24 页的『日志处理』
- 第 25 页的『更新处理』
- 第 31 页的『内存管理』
- 第 235 页的『客户机连接的连接集中器改善』

内存管理

主要的性能调整任务是决定如何在数据库的多个区域之间分配可用内存。设置本节介绍的关键配置参数可以调整此类内存分配。

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 都有它自身的专用内存。

“实例共享内存”（也称为“数据库管理器共享内存”）是在启动数据库时分配的。所有其它内存与实例共享内存相连或从其中分配。如果使用“快速通信管理器”（FCM），则会从此内存中分配缓冲区。 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，主要是消息。当第一个应用程序与数据库相连时，将会分配数据库共享内存区、应用程序共享内存区和代理进程专用内存区。实例共享内存可由 *instance_memory* 配置参数控制。缺省情况下，此参数设置为自动，以便 DB2® 可以计算需要为实例分配的内存量。

“数据库共享内存”（也称为“数据库全局内存”）是在首次激活或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”可由 *database_memory* 配置参数控制。缺省情况下，此参数设置为*自动*，以便 DB2 可以计算需要为数据库分配的内存量。

数据库共享内存中包含许多不同的内存区，其中包括:

- 缓冲池
- 锁定列表
- 数据库堆 － 其中包括日志缓冲区.
- 实用程序堆
- 程序包高速缓存
- 目录高速缓存

注: 当数据库运行时，可以分配和释放内存，也可以在不同区域之间交换内存。例如，您可以减少目录高速缓存的内存量，并向任何给定的缓冲池添加同样大小的内存量。但是，在动态地更改配置参数之前，必须与该数据库连接。以上列示的所有内存区都可以进行动态更改，尽管锁定列表内存区只能动态增大而不能减小。

数据库管理器配置参数 *numdb* 指定可以同时活动的本地数据库数。*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”）是在应用程序与下列环境中的数据库连接时分配的：分区数据库环境，或启用内部并行性的非分区数据库环境，或启用连接集中器时的非分区数据库环境。代理进程在执行由连接至数据库的客户机请求的工作时，将使用此内存。

数据库管理器配置参数 *max_connections* 设置可以与数据库连接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涉及到分配一些内存，所以允许大量并发应用程序可能将使用更多内存。

对于特定扩展数据块，最大应用程序数也由数据库管理器配置参数 *maxagents*（或者，用于分区环境的 *max_coordagents*）控制。*maxagents* 参数用于设置分区中数据库管理器代理进程总数的上限。这些数据库管理器代理进程包括活动的协调代理进程、子代理进程、非活动代理进程和空闲代理进程。

当创建一个代理进程时，将会为该代理进程分配“代理进程专用内存”。代理进程专用内存中包含仅供特定代理进程使用的内存分配，如排序堆和应用程序堆。

有几种特殊类型的共享内存:

- 代理进程／本地应用程序共享内存。此类内存用于代理进程与其客户机应用程序之间的 SQL 请求和响应通信。
- UDF／代理进程共享内存。运行受防护的 UDF 或“存储过程”的代理进程与此类内存相连。它用作通信区。
- 扩展存储器。通常是一块非常大的共享内存区（大于 4 GB），用作扩展缓冲池。代理进程／预取程序／页清除程序并不是永久地与其相连，而是根据需要与其中个别段相连。

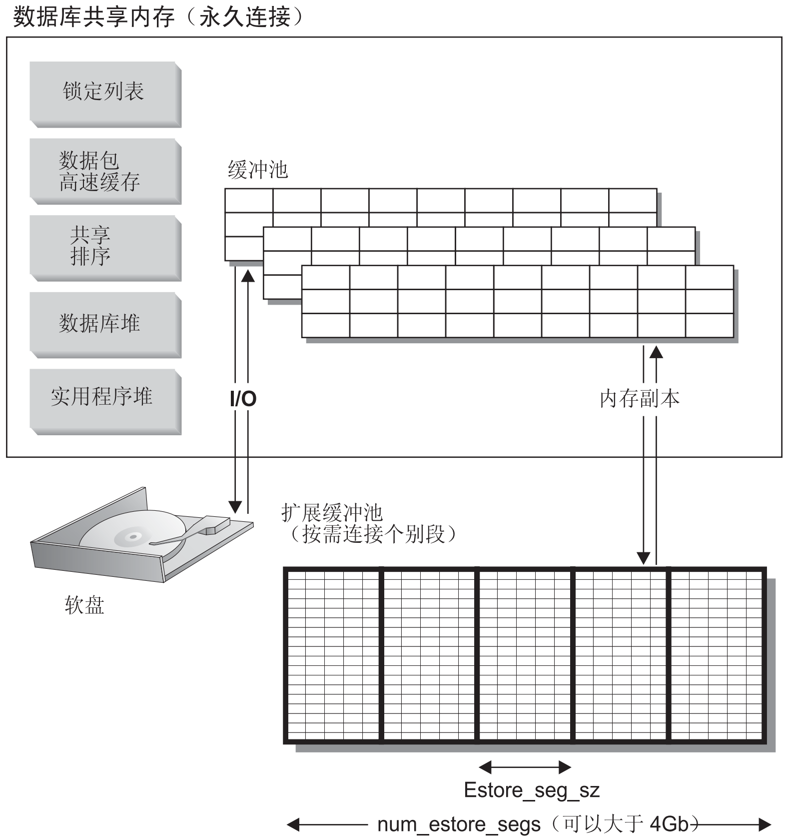


图 11. 缓冲池如何使用扩展存储器

扩展存储器用作主缓冲池的扩展后备缓冲区。它的容量可以大于 4 GB。对于具有大容量主内存的 32 位计算机，可以利用后备缓冲区来提高内存的性能。扩展存储器高速缓存是以内存段的方式进行定义的。对于 64 位计算机，不需要使用此类方法来存取所有可用内存。

但是请注意，如果将某些可寻址的实内存用作扩展存储器高速缓存，则该内存不能再用于此机器上进行的其它目的，例如，日志文件系统高速缓存或某个进程的专用地址空间。向扩展存储器高速缓存分配附加可寻址实内存会导致更高程度的系统页面调度。

下列数据库配置参数影响可用作扩展存储器的内存容量和大小:

- *num_estore_segs* 定义扩展存储器内存段的数目。
- *estore_seg_sz* 定义每个扩展内存段的大小。

每个表空间分配一个缓冲池。扩展存储器高速缓存必须始终与一个或多个特定的缓冲池相关联。扩展存储器高速缓存的页大小必须与其相关联的缓冲池的页大小相匹配。

相关概念:

- 第 191 页的『安排内存的使用』
- 第 193 页的『数据库管理器共享内存』
- 第 196 页的『全局内存和控制它的参数』
- 第 200 页的『缓冲池管理』

- 第 201 页的『在 32 位平台上扩展内存中的辅助缓冲池』
- 第 198 页的『影响内存使用率的参数的调整准则』
- 第 235 页的『客户机连接的连接集中器改善』

相关参考:

- 第 340 页的『estore_seg_sz － 扩充存储器内存段大小』
- 第 346 页的『max_coordagents － 最大协调代理进程数』
- 第 341 页的『num_estore_segs － 扩充存储器内存段数目』
- 第 347 页的『maxagents － 最大代理进程数』
- 第 423 页的『numdb － 同时活动数据库的最大数目（包括主机数据库和 iSeries 数据库管理器）』
- 第 346 页的『max_connections － 最大客户机连接数』
- 第 332 页的『instance_memory － 实例内存』
- 第 308 页的『database_memory － 数据库共享内存大小』

第 2 部分 调整应用程序性能

第 3 章 应用程序注意事项

许多因素可以影响应用程序的运行时性能。本章描述在设计和编码应用程序时以及稍后调整其性能时要考虑的一些因素。

并发性控制和隔离级别

以下各节描述并发性上不同隔离级别的效果。

并发性问题

因为许多用户存取和更改关系数据库中的数据，所以数据库管理器必须能够允许用户进行这些更改并确保保护数据完整性。*并发性*指的是可以同时由多个交互式用户或应用程序共享资源。数据库管理器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，然后继续处理其它 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。
- **幻像读现象**。在下列情况下将出现幻像读现象:
  1. 您的应用程序执行一个查询，该查询根据某些搜索条件读取一组行。
  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）。

     某些附加的（“幻像”）行作为结果集的一部分返回，但在初始执行该查询（步骤 1）时不会返回这些行。

注: 因为已声明临时表只可用于声明它们的应用程序，所以这些临时表没有并发性问题。这种类型的表从应用程序声明它起开始存在，直到应用程序完成或断开连接为止。

联合数据库系统中的并发性控制

*联合数据库系统*支持应用程序和用户落实 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（DBMS）或数据库。要引用数据源（它们由一个 DBMS 和数据组成），DB2® 使用*昵称*。昵称是其它数据库管理器中对象的别名。在联合系统中，DB2 依靠主管请求的数据的数据库管理器的并发性控制协议。

DB2 联合系统为数据库对象提供*位置透明性*。例如，具有位置透明性，如果移动了有关表和视图的信息，则可更新通过昵称对该信息的引用，而无需更改请求该信息的应用用

程序。当应用程序通过昵称存取数据时，DB2 依靠数据源数据库管理器的并发性控制协议来确保隔离级别。尽管 DB2 尝试将数据源的请求隔离级别与等效的逻辑级别匹配，但是，结果可能因数据源的容量而不同。

相关概念:

- 第 38 页的『隔离级别的性能影响』

相关任务:

- 第 41 页的『指定隔离级别』

相关参考:

- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』

隔离级别的性能影响

*隔离级别*确定存取数据时如何锁定数据或使数据不受其它进程影响。该隔离级别将在工作单元运行期间生效。在执行 OPEN CURSOR 的工作单元期间，使用由 WITH HOLD 子句的 DECLARE CURSOR 语句声明的游标的应用程序将保持选定的隔离级别。 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: 某些主机数据库服务器支持*不落实*隔离级别。对于其它数据库，此隔离级别的行为与未落实的读隔离级别一样。

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像行的情况。

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。但是，在该工作单元完成之前其它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”应用程序不能查看其它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，因此其它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）。这将防止出现幻像行。例如，如果您扫描 10 000 行并对它们应用谓词，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*数据之前所有返回的数据保持不变，即使使用了临时表或行分块也是如此。

由于“可重复读”可能获得和挂起大量锁定，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。为了避免锁定升级，优化器在认为很可能会发生锁定升级的时候，可能选择立即获得单个表级别锁定用于索引扫描。这就象数据库管理器代表您发出了一个 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，且不会读取另一个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，使用“读稳定性”时，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*幻像*行（*幻像读现象*）。重新调用扫描 10 000 行的示例时，“读稳定性”只锁定限定的行。这样，使用“读稳定性”时，只检索 10 行，且只对那十行挂起锁定。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 行上挂起锁定。挂起的锁定可以是共享、下次共享、更新或互斥锁定。

注: “读稳定性”隔离级别确保在应用程序*看到*数据之前所有返回的数据保持不变，即使使用了临时表或行分块也是如此。

“读稳定性”隔离级别的其中一个目标是提供较高并发性程度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升级前不获取表级锁定。

“读稳定性”隔离级别最适用于包括下列所有特征的应用程序:
- 在并发环境下运行
- 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。但是，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挂起该锁定。

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，任何其它应用程序都不能更新或删除该行。“游标稳定性”应用程序不能查看其它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像读现象。“游标稳定性”是缺省隔离级别，且应在需要最大并发性，但只看到其它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它事务的未落实的更改。除非其它应用程序尝试删除或改变该表，否则该应用程序也不会锁定正读取的行而使其它应用程序不能存取该行。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它事务的未落实的更改。但是，当该事务正在处理时，正由其它事务创建或删除的表、视图和索引不能使用。其它事务的任何其它更改在落实或回滚前都可被读取。

注: “未落实的读”隔离级别下的可更新操作的游标将按隔离级别是游标稳定性的方式工作。

当它使用隔离级别 UR 运行程序时，应用程序可以使用隔离级别 CS。发生这种情况的原因是因为在应用程序中使用的游标是模糊游标。由于 BLOCKING 选项，可以将模糊游标升级为隔离级别 CS。BLOCKING 选项的缺省值是 UNAMBIG。这意味着将模糊游标当作可更新的，并且隔离级别升级为 CS。要防止此升级，有两种选择:

- 修改应用程序中的游标。以便这些游标是非模糊游标。将 SELECT 语句更改为包括 FOR READ ONLY 子句。
- 将模糊游标保留在应用程序中，但是预编译程序或用 BLOCKING ALL 选项绑定它以允许当运行该程序时将任何模糊游标作为只读的。

如对扫描 10 000 行的“可重复读”给出的示例一样，如果使用“未落实的读”，则不需要任何行锁定。

使用“未落实的读”，可能出现不可重复读行为和幻像读现象。“未落实的读”隔离级别最常用于只读表上的查询，或者若仅执行选择语句且不关心是否可从其它应用程序中看到未落实的数据时也最常用。

隔离级别的总结

下表按不期望的结果总结了几个不同的隔离级别。

表 1. 隔离级别总结

隔离级别	存取未落实的数据	不可重复读	幻像读现象
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

下表提供了简单的试探方法，以帮助您为应用程序选择初始隔离级别。首先考虑下表列示的方法，并参阅先前对各级因素的讨论，可能会找到另一个更适合的隔离级别。

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对于该应用程序避免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。

相关概念:

- 第 37 页的『并发性问题』

相关任务:

- 第 41 页的『指定隔离级别』

指定隔离级别

因为隔离级别确定存取数据时如何锁定数据并使数据不受其它进程影响，所以您应该选择平衡并发性和数据完整性需求的隔离级别。您指定的隔离级别在工作单元运行期间生效。

可以采用几种不同的方式来指定隔离级别。使用下列试探来确定在编译 SQL 语句时将使用哪种隔离级别:

静态 SQL:

- 如果在语句中指定了隔离子句，则使用该子句的值。
- 如果在语句中没有指定隔离子句，则使用的隔离级别是在将程序包绑定至数据库时为该程序包指定的隔离级别。

动态 SQL:

- 如果在语句中指定了隔离子句，则使用该子句的值。
- 如果在语句中没有指定隔离子句，并且在当前会话中已经发出了 SET CURRENT ISOLATION 语句，则使用 CURRENT ISOLATION 专用寄存器的值。
- 如果在语句中没有指定隔离子句，并且在当前会话中没有发出 SET CURRENT ISOLATION 语句，则使用的隔离级别是在将程序包绑定至数据库时为该程序包指定的隔离级别。

注: 许多商业上编写的应用程序提供用于选择隔离级别的方法。有关信息，参阅应用程序文档。

过程:

要指定隔离级别:

1. **在预编译或绑定时:**

   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。也可以使用 PREP 或 BIND API 来指定隔离级别。

   - 如果在预编译时创建绑定文件，则隔离级别存储在绑定文件中。如果在绑定时不指定隔离级别，则缺省值是预编译期间使用的隔离级别。
   - 如果不指定隔离级别，则使用游标稳定性的缺省值。

   注: 要确定程序包的隔离级别，执行下列查询:

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

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

2. **在支持 REXX 的数据库服务器上**:

   当创建一个数据库时，会有多个绑定文件与该数据库绑定，这些文件支持 REXX 中的 SQL 的不同隔离级别。当创建一个数据库时，其它命令行处理器程序包也会与该数据库绑定。

   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 状态下执行更改。

   要验证 REXX 应用程序正在使用的隔离级别，检查 SQLISL REXX 变量的值。每次执行 CHANGE SQLISL 命令时更新该值。

3. **在语句级别**:

   使用 WITH 子句。语句级别的隔离级别覆盖为出现语句所在的程序包指定的隔离级别。

   可以指定下列 SQL 语句的隔离级别:

   - SELECT
   - SELECT INTO
   - 搜索的 DELETE
   - INSERT
   - 搜索的 UPDATE
   - DECLARE CURSOR

   下列情况适用于为语句指定的隔离级别:

   - 在子查询上不能使用 WITH 子句
   - WITH UR 选项只适用于只读操作。在其它情况下，语句自动从 UR 更改为 CS。

4. **运行时从 CLI 或 ODBC**:

   使用 CHANGE ISOLATION LEVEL 命令。对于 DB2 调用级接口（DB2 CLI），可以将隔离级别更改为 DB2 CLI 配置的一部分。在运行时，将 *SQLSetConnectAttr* 函数与 SQL_ATTR_TXN_ISOLATION 属性配合使用，来设置由 *ConnectionHandle* 引用的当前连接的事务隔离级别。也可以在 db2cli.ini 文件中使用 TXNISOLATION 关键字。

5. **当在运行时使用 JDBC 或 SQLJ 时**:

   注: JDBC 和 SQLJ 是通过 DB2 上的 CLI 实现的，这意味着 db2cli.ini 设置可能影响使用 JDBC 和 SQLJ 编写和运行的内容。

   在 java.sql 接口连接中使用 setTransactionIsolation 方法。

在 SQLJ 中，应运行 db2profc SQLJ 优化器来创建程序包。可为此程序包指定的选项包括其隔离级别。

6. **对于当前会话中的动态 SQL**:

   使用 SET CURRENT ISOLATION 语句来设置在会话中发出的动态 SQL 语句的隔离级别。发出此语句将把 CURRENT ISOLATION 专用寄存器设置一个值，该值用来指定在当前会话中发出的任何动态 SQL 语句的隔离级别。一旦设置了该值，CURRENT ISOLATION 专用寄存器就会为在会话中编译的任何后续动态 SQL 语句的隔离级别，无论程序包是否发出了该语句。此隔离级别将适用，直到会话结束或者使用 RESET 选项发出了 SET CURRENT ISOLATION 语句为止。

相关概念:

- 第 37 页的『并发性问题』

相关参考:

- 『SQLSetConnectAttr function (CLI) - Set connection attributes』（*CLI Guide and Reference, Volume 2*）
- 『CONNECT (Type 1) statement』（*SQL Reference, Volume 2*）
- 『Statement attributes (CLI) list』（*CLI Guide and Reference, Volume 2*）

并发性控制和锁定

以下各节描述锁定的不同类别和级别、如何确定这些锁定以及它们对数据存取性能有何影响。

锁定与并发性控制

要提供并发性控制并防止无控制的数据存取，数据库管理器将锁定置于缓冲池、表、表块或表行上。*锁定*将数据库管理器资源与应用程序关联（称为*锁定所有者*），以控制其它应用程序可以存取同一资源的方式。

虽然大多数锁定是对表发生的，但在创建、改变或删除缓冲池时，将设置缓冲池锁定。与此锁定配合使用的方式是 EXCLUSIVE (X)。当使用“命令行处理器”（CLP）获取快照时，您可能会遇到这个锁定。当查看快照时，您将看到使用的锁定名就是缓冲池本身的标识（标识）。

数据库管理器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绑定时指定的。隔离级别可以是下列其中一项:
  - 未落实的读（UR）
  - 游标稳定性（CS）
  - 读稳定性（RS）
  - 可重复读（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像读取。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、索引扫描和其它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升级以及数据库中共享对象上的并发性降低而下降。如果锁定升级频繁发生，增大 `locklist` 或 `maxlocks` 的值，或同时增大它们的值。

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（UR）。
- 选择的隔离级别是可重复的读（RR），存取方案需要不带谓词的扫描。
- 表 LOCKSIZE 属性是"TABLE"。
- 锁定列表已填充，引起升级。
- 通过 LOCK TABLE 语句获取了显式的表锁定。LOCK TABLE 语句使并发应用程序进程不能更改表或使用表。

当在数据库中的行和表中挂起的锁定数等于 `maxlocks` 数据库配置参数指定的锁定列表的百分比时，发生*锁定升级*。锁定升级可能不影响获取触发升级的锁定的表。要将锁定数减小为大约是锁定升级开始时挂起的数目的一半，数据库管理器开始将许多小的行锁定转换为所有活动表的表锁定，并从大对象（LOB）或长 VARCHAR 元素上的任何锁定开始。*互斥锁定升级*是这样的锁定升级：获取的表锁定是互斥锁定。锁定升级降低了并发性。应该避免可能引起锁定升级的情况。

行锁定的持续时间随使用的隔离级别的不同而有所变化:
- UR 扫描：除非行数据正在进行更改，否则不挂起行锁定。
- CS 扫描：仅当游标定位在行上时，才挂起行锁定。
- RS 扫描：完成事务期间只挂起合格的行锁定。
- RR 扫描：完成事务期间挂起所有行锁定。

相关概念:

- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』
- 第 49 页的『锁定的准则』

相关参考:

- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 335 页的『dlchktime － 检查死锁的时间间隔』
- 第 415 页的『diaglevel － 诊断错误捕获级别』
- 第 335 页的『locktimeout － 锁定超时』
- 第 54 页的『锁定类型兼容性』
- 第 55 页的『标准表的锁定方式和存取路径』
- 第 58 页的『表的锁定方式和 MDC 表的 RID 索引扫描』
- 第 60 页的『MDC 表的块索引扫描的锁定』

锁定属性

数据库管理器锁定具有下列基本属性:

方式 允许锁定所有者使用的存取类型以及允许锁定对象的并发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。您可以显式锁定的唯一对象类型是表。数据库管理器也可强制锁定其它类型的资源，例如行、表和表空间。对于多维群集（MDC）表，也可以强制进行块锁定。正被锁定的对象确定锁定的粒度。

持续时间
挂起锁定的时间长度。该查询运行的隔离级别影响锁定持续时间。

下表按照对资源的控制性递增的顺序显示锁定方式及其效果。有关各种级别的锁定的详细信息，参阅锁定方式引用表。

表 3. 锁定方式总结

锁定方式	适用的对象类型	描述
IN（无意向）	表空间、块和表	锁定所有者可以读取对象中的任何数据，包括未落实的数据，但不能更新它的任何数据。其它并发应用程序可以读取或更新表。
IS（意向共享）	表空间、块和表	锁定所有者可读取已锁定的表中的数据，但不能更新此数据。其它应用程序可以读取或更新表。
NS（下一键共享）	行	锁定所有者和所有并发应用程序可读取但不能更新锁定的行。在应用程序的隔离级别为 RS 或 CS 的情况下，对一个表的行获取此锁定，以代替 S 锁定。NS 锁定方式不用于下一键锁定。它在 CS 和 RS 扫描期间用来代替 S 方式以最小化下一键锁定对这些扫描的影响。
S（共享）	行、块和表	锁定所有者和所有并发应用程序可读取（但不能更新）锁定的数据。
IX（意向互斥）	表空间、块和表	锁定所有者和并发应用程序可以读取和更新数据。其它并发应用程序既可以读取也可以更新表。
SIX（在意向互斥下共享）	表和块	锁定所有者可以读取和更新数据。其它并发应用程序可以读取该表。
U（更新）	行、块和表	锁定所有者可以更新数据。其它工作单元可以读取锁定的对象中的数据，但不能尝试更新它。
NW（下一键弱互斥）	行	当将行插入索引时，获取对下一行的 NW 锁定。对于类型 2 索引，仅当下一行当前由 RR 扫描锁定时才发生此情况。锁定所有者可以读取但不可更新已锁定的行。除了与 W 和 NS 锁定兼容之外，此锁定方式类似于 X 锁定。
X（互斥）	行、块、表和缓冲池	锁定所有者既可以读取也可以更新锁定的对象中的数据。只有未落实的读应用程序可以存取已锁定的对象。
W（弱互斥）	行	当将行插入未定义类型 2 索引的表时，即获取对该行的此种锁定。锁定所有者可更改已锁定的行。要确定重复值在其被找到时是否已落实，此锁定也用于插入唯一索引期间。除了与 NW 锁定兼容之外，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（超级互斥）	表空间或表	对表的此种锁定是在一定条件下获取的，例如，当改变或删除表、创建或删除表的索引或对某些类型的表重组时。没有其它的并发应用程序可以读取或更新该表。

相关概念:

- 第 43 页的『锁定与并发性控制』

- 第 46 页的『锁定和性能』

相关参考:

- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
- 第 54 页的『锁定类型兼容性』
- 第 55 页的『标准表的锁定方式和存取路径』

锁定和性能

几个有关因素影响锁定的使用以及对应用程序性能的效果。此处讨论下列因素:

- 并发性与粒度
- 锁定兼容性
- 锁定转换
- 锁定升级
- 锁定等待和超时
- 死锁

并发性与粒度

如果某个应用程序挂起某个数据库对象上的锁定，则另一个应用程序可能不能存取该对象。因此，行级别锁定比表级别锁定对于最大化并发性更好。但是，锁定需要存储器和处理时间，因此单个表锁定最小化锁定开销。

ALTER TABLE 语句的 LOCKSIZE 子句指定行或表级别的锁定的作用域（粒度）。缺省情况下，使用行锁定。由这些已定义的表锁定仅请求 S（共享）和 X（互斥）锁定。ALTER TABLE 语句的 LOCKSIZE ROW 子句不会阻止正常的锁定升级发生。

在下列情况下，由 ALTER TABLE 语句定义的永久表锁定可能比使用 LOCK TABLE 语句来获得单个事务表锁定更可取:

- 表是只读的，且将始终只需要 S 锁定。其它用户也可以获取表的 S 锁定。
- 表通常由只读应用程序存取，但有时由单个用户存取以进行简要维护，而该用户需要 X 锁定。当维护程序运行时，将只读应用程序锁定在外，但在其它情况下，只读应用程序可以使用最小的锁定开销同时存取表。

ALTER TABLE 语句全局指定锁定，它影响存取该表的所有应用程序和用户。单个应用程序可以使用 LOCK TABLE 语句来指定应用程序级别的表锁定。

锁定兼容性

锁定兼容性是表的并发存取中的另一个重要因素。锁定兼容性引用对象的当前锁定和对该对象请求的锁定类型，并确定是否可以授权该请求。

假定应用程序 A 挂起对某个表的锁定，而应用程序 B 也想要存取该表。数据库管理器代表应用程序 B，请求某特定方式的锁定。如果由 A 挂起的锁定的方式许可 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 所挂起的锁定不兼容，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，并且释放所有其它现有不兼容锁定为止。

锁定转换

更改已挂起的锁定方式称为*转换*。当一个进程存取它已挂起锁定的数据对象，且存取方式需要比已挂起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挂起一个锁定，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。

某些锁定方式仅适用于表，而有些锁定方式仅适用于行或块。对于行或块，如果需要 X 且挂起 S 或 U（更新）锁定，则进行转换。

但是，在锁定转换这一点上，IX（意向互斥）和 S（共享）锁定是特殊情况。S 和 IX 锁定的严格程度相当，所以如果挂起了一个而请求另一个，则结果转换为 SIX（带意向互斥的共享）锁定。如果所请求的方式更严格的话，所有其它的转换都导致所请求的锁定方式变成所挂起的锁定方式。

当查询更新行时，也可以发生双重转换。如果行是通过索引存取读取的且锁定为 S，则包含该行的表具有覆盖意向锁定。但是，如果锁定类型为 IS 而不是 IX，则如果随后更改了该行，则表锁定转换为 IX，而行锁定转换为 X。

在执行查询时，锁定转换通常隐式进行。了解不同的查询以及表和索引的组合获得的锁定种类，有助于您设计和调整您的应用程序。

锁定升级

锁定升级是减少挂起的锁定数的一种内部机制。在单个表中，锁定从多个行锁定升级为表锁定，或对于多维群集（MDC）表，从多个行或块锁定升级为表锁定。当应用程序挂起任何类型的太多锁定时进行锁定升级。如果一个特定的数据库代理进程超过为它分配的锁定列表，可对该代理进程进行锁定升级。这类升级在内部处理；唯一在外部可检测的结果可能是对一个或多个表并发存取的减少。

在一个适当配置的数据库中，不会频繁进行锁定升级。例如，当应用程序设计者对大型表创建索引以提高性能与并发性，但是某个事务却存取表中的大部分记录时，可能会发生锁定升级。在此情况下，因为数据库管理器无法预计将要锁定该表的多少部分，所以它分别锁定每个记录而不是只以 S 或 X 方式锁定该表。在此情况下，数据库设计者可以与应用程序设计者协商，并建议对此事务执行 LOCK TABLE 语句。

有时候，接收到内部升级请求的进程几乎不挂起任何表的行锁定，但因为一个或多个进程挂起许多锁定而升级锁定。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升级请求的一个或多个锁定。

注: 锁定升级也可以引起死锁。例如，假定一个只读应用程序和一个更新应用程序正存取同一个表。如果更新应用程序对表中多个行具有互斥锁定，则数据库管理器可能尝试将此表上的锁定升级为互斥表锁定。但是，由只读应用程序挂起的表锁定将导致互斥锁定升级请求等待。如果只读应用程序请求对已由更新应用程序锁定的行的行锁定，则这会创建一个死锁。要避免此类问题，对更新应用程序进行编码以便启动该应用程序时互斥锁定表或增大锁定列表的大小。

锁定等待和超时

锁定超时检测是一个数据库管理器功能，它防止应用程序在异常情况下无限地等待释放锁定。例如，一个事务可能正等待另一个用户的应用程序挂起的锁定，但该用户已离开工作站，而没有允许应用程序落实释放该锁定的事务。要避免在这种情况下停止应用程序，将 *locktimeout* 配置参数设置为任何应用程序应等到获取锁定的最长时间。

设置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。如果锁定请求处于暂挂的时间大于 *locktimeout* 值，则请求应用程序将接收到一个错误并将其事务回滚。例如，如果*程序 1* 试图获取一个已由*程序 2* 挂起的锁定，那么*程序 1* 在超时周期到期时返回 SQLCODE -911（SQLSTATE 40001），原因码为 68。*locktimeout* 的缺省值为 -1，它关闭锁定超时检测。

注: 对于表、行和 MDC 块锁定，应用程序可以使用 SET CURRENT LOCK TIMEOUT 来覆盖数据库级别 *locktimeout* 设置。

要将有关锁定请求超时的更多信息记入管理通知日志，将数据库管理器配置参数 *notifylevel* 设置为 4。记录的信息包括已锁定的对象、锁定方式以及挂起有锁定的应用程序。还可能会记录当前的动态 SQL 语句或静态程序包名。仅在 *notifylevel* 4 记录动态 SQL 语句。

死锁

锁定争用可以导致死锁。例如，假定“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。如果“进程 1”接着试图以 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，两个进程均被暂挂，直到它们的第二个锁定请求被授权为止，但在其中一个进程执行落实或回滚之前不会授权这两个请求。此状态一直继续下去，直到一个外部代理进程激活了一个进程，并强制它执行回滚为止。

要处理死锁，数据库管理器使用一个称为死锁检测器的异步系统后台进程。死锁检测器按 *dlchktime* 配置参数指定的时间间隔成为活动的。当激活时，死锁检测器检查死锁的锁定系统。在分区数据库中，每个分区将*锁定图*发送到包含系统目录视图的数据库分区。在此分区上进行全局死锁检测。

如果它发现死锁，则死锁检测器选择一个已死锁的进程作为*牺牲进程*来回滚。牺牲进程被唤醒，并将 SQLCODE -911（SQLSTATE 40001）返回到调用应用程序，原因码为 2。数据库管理器自动回滚所选择的进程。当回滚完成时，释放属于牺牲进程的锁定，而该死锁涉及的其它进程可以继续进行。

要确保良好的性能，为死锁检测器选择适当的时间间隔。过短的时间间隔导致不必要的开销，而太长的时间间隔使死锁将进程延迟一段不可接受的时间。例如，5 分钟的唤醒时间间隔使死锁几乎存在 5 分钟，这对于短事务处理来说看起来可是长时间。在解决死锁时的可能延迟与检测它们的开销两者之间进行平衡®。

在分区数据库中，*dlchktime* 配置参数时间间隔仅在目录节点上适用。如果在分区数据库中检测到大量的死锁，增大 *dlchktime* 参数的值以解决锁定等待和通信等待。

当构造带有多个存取数据库的独立进程的应用程序以致很可能产生死锁时，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，然后再对同一表执行 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的机会增大。例如，如果两个进程读该表，然后更新该表，那么进程 A 试图获得对行的 X 锁定，而进程 B 对该行具有 S 锁定，反之亦然。为了避免发生这种死锁，存取具有修改意向的数据的应用程序应该执行下列其中一项操作:

- 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。但是将禁用行分块。

- 执行查询时使用 WITH RR USE AND KEEP UPDATE LOCKS 或 WITH RS USE AND KEEP UPDATE LOCKS 子句。任一子句都确保当进程 A 试图读取该数据时进行 U 锁定，并且允许行分块。

注: 可以考虑定义一个记录何时发生死锁的监视器。使用 SQL 语句 CREATE EVENT 来创建监视器。

在创建数据库的同时，还创建了详细死锁事件监视器。对于任何监视器，都有一些开销与此事件监视器相关联。如果不需要详细死锁事件监视器，则可使用以下命令删除事件监视器:

```
DROP EVENT MONITOR db2detaildeadlock
```

要限制此事件监视器消耗的磁盘空间量，一旦它达到其输出文件的最大数目，该事件监视器就取消激活，并且会有一条消息写入管理通知日志。将不再需要的输出文件除去即可在下次激活数据库时再次激活事件监视器。

在应用程序存取昵称的联合系统环境中，由于数据源处发生死锁而可能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。如果在多个数据源之间发生死锁，则 DB2 依靠数据源超时机制解开死锁。

要记录有关死锁的更多信息，将数据库管理器配置参数 *notifylevel* 设置为 4。管理通知日志存储信息，包括该对象、锁定方式以及挂起该对象上的锁定的应用程序。还可能会记录当前的动态 SQL 语句或静态程序包名。仅当 *notifylevel* 为 4 时才记录动态 SQL 语句。

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 11 页的『应用程序之间的死锁』

相关任务:

- 第 51 页的『更正锁定升级问题』

相关参考:

- 第 415 页的『diaglevel － 诊断错误捕获级别』
- 第 335 页的『locktimeout － 锁定超时』

锁定的准则

当为并发性和数据完整性调整锁定时，考虑下列准则:

- 使用频繁的 COMMIT 语句创建小的工作单元以使许多用户并发存取数据。

  当应用程序在逻辑上一致时，即当已更改的数据一致时，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。
- 指定适当的隔离级别。

  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，可挂起所有锁定，直到发出 COMMIT，这可阻止其它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。

数据库管理器确保应用程序不检索未落实的数据（其它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。

- 适当地使用 LOCK TABLE 语句。

  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它可存取的表以便在并发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。

  以共享方式锁定表

  想要存取*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，其它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，都不能更改它们。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，其它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

  想要更新表的大部分。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，阻止所有其它用户存取该表的开销少并且更有效。

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

- 在应用程序中使用 ALTER TABLE 语句。

  带 LOCKSIZE 参数的 ALTER TABLE 语句是 LOCK TABLE 语句的备用语句。LOCKSIZE 参数允许您指定下一次表存取的 ROW 锁定或 TABLE 锁定的锁定粒度。

  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。选择 TABLE 锁定可提高查询性能，方法是限制需要获取的锁定的数目。但是，并发性可能由于所有锁定位于完整的表上而降低。没有选项会阻止正常的锁定升级。

- 关闭游标以释放它们挂起的锁定。

  当用包括 WITH RELEASE 子句的 CLOSE CURSOR 语句关闭游标时，数据库管理器尝试释放所有为游标挂起的读锁定。表读锁定是 IS、S 和 U 表锁定。行读锁定是 S、NS 和 U 行锁定。块读锁定是 IS、S 和 U 块锁定。

  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*幻像读*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，那么获得新的读锁定。

  在 CLI 应用程序中，DB2® CLI 连接属性 SQL_ATTR_CLOSE_BEHAVIOR 可用来获得与 CLOSE CURSOR WITH RELEASE 相同的结果。

- 在分区数据库中，当更改影响锁定的配置参数时，确保对所有分区都作了更改。

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』
- 第 63 页的『影响锁定的因素』

更正锁定升级问题

数据库管理器可以自动将锁定从行或块级别升级为表级别。*maxlocks* 数据库配置参数指定何时触发锁定升级。获取触发锁定升级的锁定的表可能不受影响。首先对具有大多数锁定的表升级锁定，并从锁定其长对象（LOB）和长 VARCHAR 描述符的表开始，然后是具有次高数锁定的表，依此类推，直到挂起的锁定数减少至大约为 *maxlocks* 指定的值的一半。

在设计良好的数据库中，很少发生锁定升级。但是，如果锁定升级将并发性降低到不可接受的程度，则需要分析问题并决定如何解决它。

先决条件:

确保记录锁定升级信息。将数据库管理器配置参数 *notifylevel* 设置为 3（这是缺省值）或设置为 4。在 *notifylevel* 为 2 时，仅报告错误 SQLCODE。在 *notifylevel* 为 3 或 4 时，当锁定升级失败时，记录错误 SQLCODE 的信息和对其升级失败的表。仅当当前 SQL 语句是当前执行的动态 SQL 语句且将 *notifylevel* 设置为 4 时才记录该语句。

过程:

遵循这些一般步骤来诊断不可接受的锁定升级的原因并应用一个修正措施:

1. 在管理通知日志中对其升级锁定的所有表进行分析。此日志文件包括下列信息:
   - 当前挂起的锁定数目。
   - 在锁定升级完成之前所需的锁定数目。
   - 升级的每个表的表标识信息和表名。
   - 当前挂起的非表锁定的数目。
   - 作为升级的一部分，获取新表级别的锁定。通常，获取“S”（即共享锁定）或“X”（即互斥锁定）。
   - 获取新表锁定级别结果的内部返回码。
2. 使用管理通知日志中的信息来决定如何解决升级问题。考虑下列可能性:
   - 通过增大数据库配置文件中 *maxlocks* 和／或 *locklist* 参数的值来增加全局允许的锁定数。在分区数据库中，对所有分区进行此更改。

     如果其它进程对该表的并发存取很重要，可以选择此方法。但是，获得记录级别锁定的开销可能使其它进程延迟，这些延迟比并发存取表节省下来的延迟更多。
   - 调整引起升级的一个或几个进程。对于这些进程，您可以显式发出 LOCK TABLE 语句。
   - 更改隔离度。但是，注意这可能导致降低并发性。
   - 增大落实的频率以减少在给定时间挂起的锁定数。

- 为需要长 VARCHAR 或各种长对象（LOB）数据的事务考虑频繁的 COMMIT 语句。尽管在结果集具体化之前不从磁盘检索这种数据，但在首次引用数据时锁定描述符。因此，可以挂起比对包含更普通的数据的行多的锁定。

相关参考:

- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
- 第 415 页的『diaglevel － 诊断错误捕获级别』

通过锁定延迟来对未落实的数据求值

为了提高并发性，DB2® 现在允许在某些情况下对 CS 或 RS 隔离扫描延迟行锁定，直到知道一条记录满足查询的谓词为止。缺省情况下，当在表扫描或索引扫描期间执行行锁定时， DB2 会先锁定已扫描的每一行然后再确定该行是否符合查询要求。为了提高扫描的并发性，可以延迟行锁定，直到确定某行符合查询要求为止。

要利用此功能，应启用 DB2_EVALUNCOMMITTED 注册表变量。

当启用了此变量时，可对未落实的数据进行谓词求值。这意味着包含未落实的更新的行可能不符合查询要求，但是，如果等到完成更新的事务之后进行谓词求值，则该行可能会符合查询要求。另外，在表扫描期间会跳过未落实的已删除行。如果启用了 DB2_SKIPDELETED 注册表变量，则 DB2 在类型 2 索引扫描中将跳过已删除的键。

这些注册表变量设置在编译时适用于动态 SQL，而在绑定时适用于静态 SQL。这意味着，即使在运行时启用了该注册表变量也不会使用避免锁定策略（除非在绑定时启用了 DB2_EVALUNCOMMITTED）。如果在绑定时启用了该注册表变量，但是在运行时未启用它，则避免锁定策略仍然有效。对于静态 SQL，如果重新绑定了程序包，则绑定时的注册表变量设置是适用的设置。静态 SQL 的隐式重新绑定将使用 DB2_EVALUNCOMMITTED 的当前设置。

对不同存取方案的未落实数据求值的适用性

表 4. “仅 RID 索引”存取

谓词	对未落实的数据求值
无	否
SARGable	是

表 5. “仅数据”存取（关系 RID 列表或延迟 RID 列表）

谓词	对未落实的数据求值
无	否
SARGable	是

表 6. “RID 索引和数据”存取

谓词		对未落实的数据求值	
索引	数据	索引存取	数据存取
无	无	否	否
无	SARGable	否	否
SARGable	无	是	否

表 6. “RID 索引和数据”存取（续）

谓词		对未落实的数据求值	
索引	数据	索引存取	数据存取
SARGable	SARGable	是	否

表 7. 块索引和数据存取

谓词		对未落实的数据求值	
索引	数据	索引存取	数据存取
无	无	否	否
无	SARGable	否	是
SARGable	无	是	否
SARGable	SARGable	是	是

示例

以下示例对缺省锁定行为与新的对未落实的数据进行求值的行为作了比较。

下表是 SAMPLE 数据库中的 ORG 表。

```
DEPTNUMB DEPTNAME       MANAGER DIVISION   LOCATION
-------- -------------- ------- ---------- -------------
      10 Head Office        160 Corporate  New York
      15 New England         50 Eastern    Boston
      20 Mid Atlantic        10 Eastern    Washington
      38 South Atlantic      30 Eastern    Atlanta
      42 Great Lakes        100 Midwest    Chicago
      51 Plains             140 Midwest    Dallas
      66 Pacific            270 Western    San Francisco
      84 Mountain           290 Western    Denver
```

正在对此表执行下列事务，它具有缺省“游标稳定性”（CS）隔离级别。

表 8. 对具有 CS 隔离级别的 ORG 表执行的事务

会话 1	会话 2
connect to SAMPLE	connect to SAMPLE
+c update org set deptnumb=5 where manager=160	
	select * from org where deptnumb >= 10

会话 1 中未落实的 UPDATE 对表中第一行保持互斥记录锁定，禁止会话 2 中的 SELECT 查询返回，即使会话 1 中正在更新的行当前不符合会话 2 中的查询要求也是如此。这是因为 CS 隔离级别要求当游标位于查询访问的任何一行时必须锁定该行。只有在会话 1 释放对第一行的锁定之后，会话 2 才能获得对该行的锁定。

当对表进行扫描时，可以使用对未落实的数据进行求值这项功能来避免会话 2 中的锁定等待，该功能首先对谓词求值，然后锁定谓词求值结果为 true 的行。这样，会话 2 中的查询将不尝试锁定表中的第一行，从而提高了应用程序并发性。注意，这也意味着对于会话 1 中 deptnumb=5 的未落实的值在会话 2 中将进行谓词求值。尽管会话 1 中对更新的回滚将符合会话 2 中的查询要求，但是会话 2 中的查询仍将忽略它的结果集中的第一行。

如果保留了操作顺序，则通过对未落实的数据进行求值仍然可提高并发性。通过使用缺省锁定行为，会话 2 将首先获取行锁定，禁止执行会话 1 中搜索到的 UPDATE，即使会话 1 中的 UPDATE 将不更改会话 2 的查询锁定的行也是如此。如果会话 1 中搜索到的 UPDATE 首先尝试了检查一些行，然后仅在这些行符合条件时才锁定它们，则会话 1 中的查询将不会分块。

限制

下列外部限制适用于此新功能:

- 必须启用注册表变量 DB2_EVALUNCOMMITTED。
- 隔离级别必须是 CS 或 RS。
- 要发生行锁定。
- 存在 SARGable 求值谓词。
- 未落实的数据求值不适用于对目录表的扫描。
- 对于 MDC 表，可以对索引扫描延迟块锁定；但是，将不对表扫描延迟块锁定。
- 对于正在执行适当的表重组的表，将不会发生延迟锁定。
- 对于索引为类型 1 的索引扫描，将不会发生延迟锁定。
- 对于扫描预取方案，行锁定不会延迟到进行数据存取，而是在移至表中的某行之前在索引存取期间就锁定该行。
- 对于表扫描，会无条件地跳过已删除的行，但是，仅当启用了注册表变量 DB2_SKIPDELETED 时，才会跳过已删除的类型 2 的索引键。

锁定类型兼容性

下表显示有关某些情况的信息，在这些情况下，在给定状态下，当另一个进程挂起或正在请求对同一个资源的锁定时，可以授权锁定请求。**否**指示请求者必须等待，直到所有不兼容的锁定被其它进程释放为止。注意，当请求者正等待锁定时，可能出现超时。**是**指示授予该锁定，除非更早的请求者正在等待该资源。

表 9. 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**	**X**	**Z**	**NW**	**W**
无	是	是	是	是	是	是	是	是	是	是	是	是
IN	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
NS	是	是	是	是	是	否	否	是	否	否	是	否
S	是	是	是	是	是	否	否	是	否	否	否	否
IX	是	是	是	否	否	是	否	否	否	否	否	否
SIX	是	是	是	否	否	否	否	否	否	否	否	否
U	是	是	是	是	是	否	否	否	否	否	否	否
X	是	是	否	否	否	否	否	否	否	否	否	否
Z	是	否	否	否	否	否	否	否	否	否	否	否
NW	是	是	否	是	否	否	否	否	否	否	否	是
W	是	是	否	否	否	否	否	否	否	否	是	否

注:

I 意向

N 无

NS 下一键共享

S 共享

X 互斥

U 更新

Z 超互斥

NW 下一键弱互斥

W 弱互斥

注:

- 是 － 立即**准许**所请求的锁定
- 否 － **等待**释放挂起的锁定或出现超时

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』

相关参考:

- 第 55 页的『标准表的锁定方式和存取路径』
- 第 60 页的『MDC 表的块索引扫描的锁定』

标准表的锁定方式和存取路径

本主题包含有关不同数据存取方案的标准表锁定方法的参考信息。

下表列示了不同存取方案的每个级别的标准表获得的锁定类型。每个条目均由两部分组成：表锁定和行锁定。虚线表示未进行某个特定级别的锁定。

注:

1. 在多维群集（MDC）环境中，还将使用其它锁定级别 BLOCK。
2. 可以使用 SELECT 语句的锁定请求子句来显式更改锁定方式。

表 10. 表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描	更新或删除
存取方法: *无谓词的表扫描*					
RR	S/-	U/-	SIX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
存取方法: *带谓词的表扫描*					
RR	S/-	U/-	SIX/X	U/-	SIX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

注: 在 UR 隔离级别，如果具有类型 1 索引的 IN 锁定，或者如果索引的包含列中具有谓词，则隔离级别将升级为 CS 并且锁定将升级为 IS 表锁定和 NS 行锁定。

表 11. RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描	更新或删除
存取方法: *无谓词的* ***RID*** *索引扫描*					
RR	S/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
存取方法: *带单个限定行的* ***RID*** *索引扫描*					
RR	IS/S	IX/U	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
存取方法: *只带启动和停止谓词的索引扫描*					
RR	IS/S	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
存取方法: *只带索引和其它谓词（**sargs** 或 **resids**）的索引扫描*					
RR	IS/S	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X

表 11. RID 索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描	更新或删除
UR	IN/-	IX/U	IX/X	IX/U	IX/X

下表显示了在某些情况下的锁定方式，在这些情况下，系统将延迟读取数据页以允许将行的列表:

- 使用多个索引进一步限定
- 排序以获得高效的预取

表 12. 延迟的数据页存取所用的索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描	更新或删除
存取方法: *无谓词的 **RID** 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 **RID** 索引扫描之后）*					
RR	IN/-	IX/S	IX/X	X/-	X/-
RS	IS/NS	IX/U	IX/X	IX/X	IX/X
CS	IS/NS	IX/U	IX/X	IX/X	IX/X
UR	IN/-	IX/U	IX/X	IX/X	IX/X
存取方法: *带谓词（**sargs** 或 **resids**）的 **RID** 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: *只带启动和停止谓词的 **RID** 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（在只带启动和停止谓词的 **RID** 索引扫描之后）*					
RR	IN/-	IX/S	IX/X	IX/X	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IS/-	IX/U	IX/X	IX/U	IX/X
存取方法: *延迟的数据页存取（在带谓词的 **RID** 索引扫描之后）*					
RR	IN/-	IX/S	IX/X	IX/S	IX/X
RS	IS/NS	IX/U	IX/X	IX/U	IX/X
CS	IS/NS	IX/U	IX/X	IX/U	IX/X
UR	IN/-	IX/U	IX/X	IX/U	IX/X

相关概念:

- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』

相关参考:

- 第 54 页的『锁定类型兼容性』
- 第 58 页的『表的锁定方式和 MDC 表的 RID 索引扫描』
- 第 60 页的『MDC 表的块索引扫描的锁定』

表的锁定方式和 MDC 表的 RID 索引扫描

在多维群集（MDC）环境中，还将使用附加的锁定级别 BLOCK。下表列示了不同存取方案在各个级别获得的锁定类型。每个条目由三部分组成：表锁定、块锁定和行锁定。虚线表示未使用某个特定级别的锁定。

| **注:** 可以使用 SELECT 语句的锁定请求子句来显式更改锁定方式。

表 13. 表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描或删除	更新
存取方法: *无谓词的表扫描*					
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
存取方法: *只在维列上带谓词的表扫描*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
存取方法: *带其它谓词（**sargs** 或 **resids**）的表扫描*					
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

以下两个表显示了 MDC 表的 RID 索引的锁定方式。

表 14. RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描删除	更新
存取方法: *无谓词的 **RID** 索引扫描*					
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-

表 14. RID 索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描删除	更新
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
存取方法: *带单个限定行的 **RID** 索引扫描*					
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X
存取方法: *只带启动和停止谓词的 **RID** 索引扫描*					
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
存取方法: *只带索引谓词的索引扫描*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
存取方法: *带其它谓词（**sargs** 或 **resids**）的索引扫描*					
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

注: 在下表中，显示了用于延迟的数据页存取的 RID 索引扫描的锁定方式，在 UR 隔离级别，如果具有类型 1 索引的 IN 锁定，或者如果在索引的包含列中具有谓词，则隔离级别将升级至 CS 并且锁定将升级至 IS 表锁定、IS 块锁定和 NS 行锁定。

表 15. 延迟的数据页存取所用的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描删除	更新
存取方法: *无谓词的 **RID** 索引扫描*					
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
存取方法: *延迟的数据页存取（在无谓词的 **RID** 索引扫描之后）*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 15. 延迟的数据页存取所用的 RID 索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索更新或删除	
		扫描	当前	扫描删除	更新
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
存取方法: *带谓词（**sargs** 或 **resids**）的 **RID** 索引扫描*					
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
存取方法: *延迟的数据页存取（在带谓词（**sargs** 或 **resids**）的 **RID** 索引扫描之后）*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
存取方法: *只带启动和停止谓词的 **RID** 索引扫描*					
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	
存取方法: *延迟的数据页存取（在只带启动和停止谓词的 **RID** 索引扫描之后）*					
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』

相关参考:

- 第 54 页的『锁定类型兼容性』
- 第 55 页的『标准表的锁定方式和存取路径』
- 第 60 页的『MDC 表的块索引扫描的锁定』

MDC 表的块索引扫描的锁定

以下表列示在不同存取方案的各个级别获取的锁定类型。每个条目由三部分组成: 表锁定、块锁定和行锁定。虚线指示不执行特定级别的锁定。

注: 可以使用 SELECT 语句的锁定请求子句来显式更改锁定方式。

表 16. 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标的操作		搜索的更新或删除	
		扫描	当前位置	扫描删除	更新
存取方法: *不带谓词*					
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--
存取方法: *仅带维谓词*					
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
存取方法: *只带维开始和停止谓词*					
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
存取方法: *无谓词的索引扫描*					
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

下表列示延迟的数据页存取所使用的块索引扫描的锁定方式:

表 17. 延迟的数据页存取所用的块索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标的操作		搜索的更新或删除	
		扫描	当前位置	扫描删除	更新
存取方法: *无谓词的块索引扫描*					
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
存取方法: *无谓词的块索引扫描之后的延迟数据页存取*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--
存取方法: *只带维谓词的块索引扫描*					
RR	IS/S/--	IX/IX/--		IX/S/--	

表 17. 延迟的数据页存取所用的块索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标的操作		搜索的更新或删除	
		扫描	当前位置	扫描删除	更新
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	
存取方法: *只带维谓词的块索引扫描之后的延迟数据页存取*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
存取方法: *只带开始和停止谓词的块索引扫描*					
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
存取方法: *只带开始和停止谓词的块索引扫描之后的延迟数据页存取*					
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	
存取方法: *带其它谓词(**sargs** 或 **resids**)的块索引扫描*					
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	
存取方法: *带其它谓词(**sargs** 或 **resids**)的块索引扫描之后的延迟数据页存取*					
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

相关概念:

- 第 46 页的『锁定和性能』

相关参考:

- 第 54 页的『锁定类型兼容性』
- 第 55 页的『标准表的锁定方式和存取路径』
- 第 58 页的『表的锁定方式和 MDC 表的 RID 索引扫描』

影响锁定的因素

下列因素影响数据库管理器锁定的方式和粒度:

- 应用程序执行的处理类型
- 数据存取方法
- 索引是类型 2 还是类型 1
- 各种配置参数

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』
- 第 49 页的『锁定的准则』
- 第 227 页的『索引清除和维护』
- 第 63 页的『应用程序处理的锁定和类型』
- 第 64 页的『锁定和数据存取方法』
- 第 65 页的『索引类型和下一键锁定』

影响锁定的因素

应用程序处理的锁定和类型

为了确定锁定属性，可以将应用程序处理划分为下列类型的其中一种:

- 只读

  此类型包括所有这样的选择语句，它们本身是只读的，并具有一个显式的 FOR READ ONLY 子句，或者是模糊的（但是 SQL 编译器因为 PREP 或 BIND 命令指定了 BLOCKING 选项的值而假定是只读的）。此处理类型只要求“共享”锁定（S、NS 或 IS）。
- 更改意向

  此类型包括具有 FOR UPDATE 子句、具有 USE AND KEEP UPDATE LOCKS 子句、具有 USE AND KEEP EXCLUSIVE LOCKS 子句的所有 SELECT 语句，或者 SQL 编译器解释有歧义的语句以表示想进行更改。此类型使用“共享”和“更新”锁定（对于行，为 S、U 及 X；对于块，为 IX、U、X 和 S；对于表，为 IX、U 和 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。此类型要求“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在目标表中进行插入、更新或删除数据的语句执行两种类型的处理。只读处理规则确定对在子选择语句中返回的表的锁定。更改处理规则确定对目标表的锁定。

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』
- 第 49 页的『锁定的准则』
- 第 11 页的『应用程序之间的死锁』
- 第 64 页的『锁定和数据存取方法』
- 第 65 页的『索引类型和下一键锁定』

相关任务:

- 第 51 页的『更正锁定升级问题』

相关参考:

- 第 54 页的『锁定类型兼容性』

锁定和数据存取方法

*存取方案*是优化器选择以从特定表中检索数据的方法。存取方案可对锁定方式具有很大影响。例如，当使用索引扫描来查找特定行时，优化器将可能为该表选择行级别锁定（IS）。例如，如果 EMPLOYEE 表有一个关于职员编号（EMPNO）的索引，通过索引访问可以用来选择单个职员的信息，使用包含下列 SELECT 子句的语句:

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

如果不使用索引，则必须按顺序扫描整个表来找到所选的行，并且可因此获取单个表级锁定（S）。例如，如果 SEX 列上没有索引，则表扫描可以用来选择所有的男性职员，使用包含下列 SELECT 子句的语句:

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

注: 受控游标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，总会获取互斥锁定以执行更新或删除。

范围群集表中的锁定与标准键或下一个键锁定的工作方式稍微有所不同。当存取范围群集表中的一定范围的行时，该范围内的所有行都会被锁定，即使某些行是空的也会被锁定。在标准键或下一个键锁定中，将只锁定具有现有记录的行。

引用表提供了有关哪一类存取方案获取哪种锁定的详细信息。

被延迟的数据页存取意味着对行的存取分两步进行，这导致更复杂的锁定情况。锁定获得的计时和锁定的持久性取决于隔离级别。因为“可重复读”隔离级别保留所有锁定直到事务结束，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。要使并发性最大化，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。

相关概念:

- 第 43 页的『锁定与并发性控制』
- 第 45 页的『锁定属性』
- 第 46 页的『锁定和性能』
- 第 49 页的『锁定的准则』
- 第 63 页的『应用程序处理的锁定和类型』
- 第 65 页的『索引类型和下一键锁定』

相关任务:

- 第 51 页的『更正锁定升级问题』

相关参考:

- 第 54 页的『锁定类型兼容性』
- 第 55 页的『标准表的锁定方式和存取路径』
- 第 58 页的『表的锁定方式和 MDC 表的 RID 索引扫描』
- 第 60 页的『MDC 表的块索引扫描的锁定』

索引类型和下一键锁定

当事务引起类型 1 索引的更改时，会发生某些下一键锁定。对于类型 2 索引，发生最小的下一键锁定。

- 对于类型 2 索引的下一键锁定

  当将某个键插入索引时发生下一键锁定。

  在某个键插入索引期间，仅当与将跟在该索引中新键后面的键对应的行当前由 RR 索引扫描锁定时，才锁定该行。用于下一键锁定的锁定方式是 NW。在实际执行键插入之前，释放该下一键锁定。当将行插入表时，发生键插入。

  当更新某一行导致对该行的索引键的值的更改时，键插入也因为将原始键值标记为已删除而发生且将新的键值插入索引中。对于仅影响索引的包含列的更新，可适当更新键而不发生下一键锁定。

  在 RR 扫描期间，以 S 方式锁定与跟在扫描范围末尾后面的键对应的行。如果没有键跟在扫描范围末尾的后面，则获取表结束锁定以锁定索引的末尾。如果将跟在扫描范围末尾后面的键标记为已删除，则扫描继续锁定对应的行，直到它找到不标记为已删除的键，当它锁定该键的对应行时，或直到锁定索引的末尾为止。

- 对于类型 1 索引的下一键锁定:

  在对索引的删除和插入期间和在索引扫描期间，发生下一键锁定。当更新表中的行、从表中删除行或将行插入表中时，获取对该行的 X 锁定。对于插入，这可能降级为 W 锁定。

当从表索引删除键或将键插入其中时，锁定与索引中跟在已删除或插入的键后面的键对应的表行。对于影响键值的更新，首先删除原始的键值并插入新的值，以便获取两个下一键锁定。如下所示确定这些锁定的持续时间:

– 在索引键删除期间，对下一键的锁定方式是 X 且挂起该锁定，直到落实时为止。
– 在索引键插入期间，对下一键的锁定方式是 NW。仅当存在锁定的争用时，才获取此锁定，在此情况下，在将键实际插入索引之前释放该锁定。
– 在 RR 扫描期间，以 S 方式锁定与正好超出索引扫描范围末尾的键对应的表行并在落实时间之前挂起。
– 在 CS/RS 扫描期间，如果存在锁定的争用，则以 NS 方式锁定与正好超出索引扫描范围末尾的键对应的行。一旦验证了扫描范围的结束，则释放此锁定。

键插入和键删除期间对于类型 1 索引的下一键锁定可以导致死锁。以下示例显示两个事务是如何死锁的。对于类型 2 索引，这种死锁不会发生。

考虑索引的下列示例，该索引包含 6 行，并分别具有下列值: `1 5 6 7 8 12`。

1. 事务 1 删除具有键值 8 的行。以 X 方式锁定具有值 8 的行。当删除索引中对应的键时，以 X 方式锁定具有值 12 的行。
2. 事务 2 删除具有键值 5 的行。以 X 方式锁定具有值 5 的行。当删除索引中对应的键时，以 X 方式锁定具有值 6 的行。
3. 事务 1 插入具有键值 4 的行。以 W 方式锁定此行。当尝试将新的键插入索引时，以 NW 方式锁定具有值 6 的行。此锁定尝试将在事务 2 在此行具有的 X 锁定上等待。
4. 事务 2 插入具有键值 9 的行。以 W 方式锁定此行。当尝试将新的键插入索引时，以 NW 方式锁定具有键值 12 的行。此锁定尝试将在事务 1 在此行具有的 X 锁定上等待。

当使用类型 1 索引时，此方案将导致死锁且将这些事务的其中一项事务回滚。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 225 页的『索引性能技巧』
- 第 22 页的『索引结构』
- 第 229 页的『索引重组』
- 第 230 页的『联机索引整理碎片』
- 第 227 页的『索引清除和维护』

优化因素

本节描述在指定查询的优化级别时要考虑的因素。

优化级别准则

当编译 SQL 查询时，可以指定优化级别以确定优化器如何选择该查询的最有效存取方案。尽管可以单独指定优化技术以提高查询的运行时性能，但是，指定的优化技术越多，查询编译将需要越多的时间和系统资源。

注: 在联合数据库查询中，优化级别不适用于远程优化器。

设置优化级别可以提供有关显式指定优化技术的某些优点，特别是由于下列原因:

- 管理非常小型的数据库或非常简单的动态查询
- 适应数据库服务器上编译时的内存限制
- 减少查询编译时间，如 PREPARE。

通过使用优化级别 5（缺省查询优化级别），大多数语句都可以使用合理数量的资源充分优化。以给定的优化级别，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所执行的优化数量影响。

查询优化级别 1、2、3、5 和 7 都比较通用。仅当您在查询编译时间方面需要进一步减少且您了解 SQL 语句相当简单时，才考虑级别 0。

技巧: 要分析长时间运行的查询，使用 db2batch 运行该查询以确定编译和执行各花费了多长时间。如果编译需要更长时间，则降低优化级别。如果执行需要更长时间，则考虑更高的优化级别。

当选择优化级别时，考虑下列一般准则:

- 通过使用缺省查询优化级别（级别 5）开始。
- 要使用不是缺省值的级别，首先尝试级别 1、2 或 3。级别 0、1 和 2 使用“贪婪”连接枚举算法。
- 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。
- 对运行时间短（不到一秒）的查询，使用低优化级别（0 或 1）。这些查询趋向于具有下列特征:
  - 存取单个表或仅存取少量表
  - 访存一行或仅访存少量行
  - 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。
- 对于运行时间较长（大于 30 秒）的查询，使用高优化级别（3、5 或 7）。
- 级别 3 和更高的级别使用“动态编程”连接枚举算法。与级别 0、1 和 2 相比，此算法考虑更多替代方案，并且可能使编译时间也显著增加，特别是随着表的数量增加，更是如此。
- 仅当具有查询的特定异常的优化需求时，使用优化级别 9。

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。对具有下列特征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，对于这些复杂查询至少应该使用缺省查询优化级别。

对于由查询生成器产生的 SQL 使用较高的查询优化级别。许多查询生成器创建运行效率不高的 SQL。编写得很差的查询，包括那些由查询生成器产生的查询，需要另外优化以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这样的 SQL 查询。

相关概念:

- 第 124 页的『影响查询优化的配置参数』
- 第 277 页的『基准程序测试』
- 第 158 页的『分区内并行性的优化策略』
- 第 160 页的『MDC 表的优化策略』

相关任务:

- 第 70 页的『设置优化级别』

相关参考:

- 第 68 页的『优化级别』

优化级别

可以在编译 SQL 查询时指定下列其中一个优化器级别:

0 - 此级别指导优化器使用最少的优化来生成存取方案。此优化级别具有下列特征:
- 优化器不考虑任何非均匀分布统计信息。
- 仅应用基本的查询重写规则。
- 发生贪婪联合枚举。
- 仅允许使用嵌套循环连接及索引扫描存取方法。
- 在生成的存取方法中不使用列表预取和索引 AND 运算。
- 不考虑星型连接策略。

此级别应该只用于需要最低的查询编译开销的情况。查询优化级别 0 适用于以下应用程序：完全由存取经过良好索引表的很简单的动态 SQL 语句组成。

1 - 此优化级别具有下列特征:
- 优化器不考虑任何非均匀分布统计信息。
- 只应用查询重写规则的一个子集。
- 发生贪婪联合枚举。
- 在生成的存取方法中不使用列表预取和索引 AND 运算，尽管在处理星型连接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 类似于级别 0。

2 - 此级别指导优化器使用比级别 1 显著高的优化程度，而使复杂查询的编译成本显著低于级别 3 及更高级别。此优化级别具有下列特征:
- 利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。
- 除只在极少情况下才适用的计算密集型规则外，将应用所有其它查询重写规则，包括路由对具体查询表的查询。
- 使用了贪婪联合枚举。
- 考虑各种存取方法，包括列表预取和具体查询表路由。

- 如果适用的话，考虑星型连接策略。

优化级别 2 除了使用“贪婪”联合枚举而不是“动态规划”以外，类似于级别 5。在所有使用“贪婪”联合枚举算法的级别中，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。建议将级别 2 用于决策支持或联机分析处理（OLAP）环境中非常复杂的查询。在这种环境下，特定查询很少完全重复，因此查询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 - 此级别请求中等优化。此级别与 DB2 MVS/ESA 版、OS/390 或 z/OS 版的查询优化特征基本匹配。此优化级别具有下列特征:

- 使用非均匀分布统计信息（如果可用的话），该统计信息跟踪频繁出现的值。
- 应用大部分查询重写规则，包括子查询至连接的变换。
- 动态规划连接枚举，如下所示:
  - 组合内部表的有限使用
  - 涉及查找表的星型模式的笛卡尔乘积的有限使用
- 考虑各种存取方法，包括列表预取、索引 AND 运算和星型连接。

此级别适用于大量应用程序。此级别改进具有 4 个或更多连接的查询的存取方案。但是，优化器可能无法考虑使用缺省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。此优化级别具有下列特征:

- 使用所有可用的统计信息，包括频率和分位数分布统计信息。
- 除只在极少情况下才适用的那些计算密集型规则外，将应用所有其它查询重写规则，包括路由对具体查询表的查询。
- 动态规划连接枚举，如下所示:
  - 组合内部表的有限使用
  - 涉及查找表的星型模式的笛卡尔乘积的有限使用
- 考虑各种存取方法，包括列表预取、索引 AND 运算和具体查询表路由。

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。此优化级别设计成可以用高效的方式应用最有价值的查询变换和其它查询优化技术。

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。对于使用大表的很复杂且运行时间很长的查询，可以使用此级别来确定更全面优化是否将生成更好的存取方案。使用“说明”和性能测量来验证是否实际上已找到更好的方案。

相关概念:

- 第 66 页的『优化级别准则』
- 第 158 页的『分区内并行性的优化策略』
- 第 167 页的『在联合数据库中的远程 SQL 生成和全局优化』
- 第 160 页的『MDC 表的优化策略』

相关任务:

- 第 70 页的『设置优化级别』

设置优化级别

当指定优化级别时，考虑查询是使用静态 SQL 还是动态 SQL，以及是否重复执行同一个动态 SQL。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。因为动态语句是在运行时绑定和执行的，所以，考虑对动态语句进行另外的优化开销是否改善总体性能。但是，如果重复执行相同的动态 SQL 语句，则高速缓存所选存取方案。这种语句可以使用与静态 SQL 语句相同的优化级别。

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使用情况，那么您可以执行一些基准程序测试。

过程:

要指定查询优化级别，遵循下列步骤:

1. 按以下方式非正式地或用正式测试分析性能因素:
   - 对于**动态** SQL 语句，测试应比较该语句的平均运行时间。使用下列公式来估计平均运行时间:

     ```
                 编译时间 + 所有重复的执行时间的总和
     --------------------------------------------------------
                         重复次数
     ```

     在此公式中，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: 在初始编译后，当环境更改要求动态 SQL 语句时，将再次编译这些 SQL 语句。如果将 SQL 语句高速缓存之后环境不更改，则不需要再次编译，因为后来的 PREPARE 语句将再使用高速缓存的语句。

   - 对于**静态** SQL 语句，比较语句运行时间。

     尽管您可能对静态 SQL 的编译时间也感兴趣，但该语句的总计编译与运行时间在任何有意义的上下文中都很难估计。比较总时间不能识别这样一个事实，即每次将静态 SQL 语句绑定时，它可以运行多次，而在运行时，一般不将它绑定。
2. 如下所示指定优化级别:

- **动态** SQL 语句使用 CURRENT QUERY OPTIMIZATION 专用寄存器指定的优化级别，该寄存器是用 SQL 语句 SET 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，可以将 SET 语句加入应用程序中。

  如果尚未设置 CURRENT QUERY OPTIMIZATION 寄存器，将使用缺省查询优化级别绑定动态语句。动态和静态 SQL 的缺省值由数据库配置参数 *dft_queryopt* 的值确定。级别 5 是此参数的缺省值。绑定选项和专用寄存器的缺省值也是从 *dft_queryopt* 数据库配置参数读取的。
- **静态** SQL 语句使用 PREP 和 BIND 命令上指定的优化级别。SYSCAT.PACKAGES 目录表中的 QUERYOPT 列记录用于绑定程序包的优化级别。如果以隐式方式或使用 REBIND PACKAGE 命令重新绑定程序包，将把这同一个优化级别用于静态 SQL 语句。要更改用于这些静态 SQL 语句的优化级别，使用 BIND 命令。如果未指定优化级别，则 DB2 使用 *dft_queryopt* 数据库配置参数指定的缺省优化级别。

相关概念:

- 第 66 页的『优化级别准则』

相关参考:

- 第 68 页的『优化级别』

调整应用程序

本节提供调整应用程序执行的查询的准则。

限制选择语句的准则

优化器假定应用程序必须检索由 SELECT 语句指定的所有行。此假设最适合于 OLTP 和批处理环境。但是，在“浏览”应用程序中，查询经常定义一个可能大的答案集，但它们只检索前几行，通常只检索填满该屏幕所需的那么多行。

要提高这种应用程序的性能，可以按下列方式修改 SELECT 语句:

- 使用 FOR UPDATE 子句来指定可由后续定位的 UPDATE 语句更新的列。
- 使用 FOR READ/FETCH ONLY 子句来使返回的列为只读的。
- 使用 OPTIMIZE FOR *n* ROWS 子句来给予检索整个结果集中前 *n* 行的优先级。
- 使用 FETCH FIRST *n* ROWS ONLY 子句来仅检索指定的几行。
- 使用 DECLARE CURSOR WITH HOLD 语句来每次检索一行。

| **注:** 如果使用 FOR UPDATE、FETCH FIRST *n* ROWS ONLY 或 OPTIMIZE FOR *n*
| ROWS 子句，或者如果您声明游标为“滚动”，则会影响行分块。

以下节描述每个方法的性能优点。

FOR UPDATE 子句

FOR UPDATE 子句通过仅包含可由后续定位的 UPDATE 语句更新的列来限制结果集。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。

在下列情况下，不能使用 FOR UPDATE 子句:

- 如果不能删除与 SELECT 语句关联的游标。
- 如果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。

由于同样的目的，在 CLI 应用程序中使用 DB2® CLI 连接属性 SQL_ATTR_ACCESS_MODE。

FOR READ 或 FETCH ONLY 子句

FOR READ ONLY 子句或 FOR FETCH ONLY 子句确保返回只读结果。因为被定义为只读的视图上 SELECT 的结果表也是只读的，所以允许此子句但没有作用。

对于允许更新和删除的结果表，如果数据库管理器可以检索数据块而不是互斥锁定，则指定 FOR READ ONLY 可能会提高 FETCH 操作的性能。不要对用于定位的 UPDATE 或 DELETE 语句的查询使用 FOR READ ONLY 子句。

由于相同的目的，可将 DB2 CLI 连接属性 SQL_ATTR_ACCESS_MODE 用于 CLI 应用程序。

OPTIMIZE FOR n ROWS 子句

OPTIMIZE FOR 子句声明它只想检索结果的一个子集或优先检索前几行。优化器然后可以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，假设您定期查询职员表，来查找具有最高工资的职员。

```
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，由于职员是按职员号排序的，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，优化器将可能选择使用列表预取存取方法，此方法需要对合格的所有行的行标识排序。在将前几个合格行返回至应用程序前，此排序可能导致一个延迟。要防止此延迟，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
```

在此情况下，优化器可能选择直接使用 SALARY 索引，因为只检索具有最高工资的二十个职员。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，优化器优先选择可以避免大量操作或中断行流动（如排序）的存取方案。使用 OPTIMIZE FOR 1 ROW 最有可能影响存取路径。使用此子句可以有下列作用:

- 与组合内部表的连接顺序不大可能，因为它们需要一个临时表。
- 连接方法可以更改。一个嵌套循环连接是非常可能的选择，因为它具有低开销成本，并且通常在检索少量行时更有效率。
- 一个与 ORDER BY 子句匹配的索引更可能，因为对于 ORDER BY 不需要排序。
- 列表预取不大可能，因为此存取方法需要排序。
- 顺序预取不大可能，因为知道只需要少量的几行。
- 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，前提是该外部表上的一个索引提供 ORDER BY 子句所需的排序。

虽然 OPTIMIZE FOR 子句适用于所有优化级别，但它在优化级别 3 和更高级别下工作得最好，因为级别 3 以下的级别使用“贪婪”连接枚举方法。此方法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。如果确实要检索全部合格行，则总经过时间可能大大高于如果优化器为整个答案集进行优化所需的时间。

如果已打包的应用程序使用调用级接口（DB2 CLI 或 ODBC），可在 `db2cli.ini` 配置文件中使用 OPTIMIZEFORNROWS 关键字，让 DB2 CLI 自动将一个 OPTIMIZE FOR 子句追加至每个查询语句的末尾。

当从昵称选择数据时，结果可能随数据源支持的不同而变化。如果该昵称引用的数据源支持 OPTIMIZE FOR 子句且 DB2 优化器将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最低成本方案，则在本地应用 OPTIMIZE FOR 子句。在这种情况下，DB2 优化器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，则这两个值中较小的一个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。

FETCH FIRST n ROWS ONLY 子句

FETCH FIRST *n* ROWS ONLY 子句设置可检索的最大行数。将结果表限制为只包含前几行可提高性能。无论结果集可能另外包含多少行，只检索 *n* 行。

如果同时指定了 FETCH FIRST 子句和 OPTIMIZE FOR 子句，则这两个值中较小的一个影响通信缓冲区大小。为了达到优化目的，这两个值是互相独立的。

DECLARE CURSOR WITH HOLD 语句

当用包括 WITH HOLD 子句的 DECLARE CURSOR 语句声明游标时，在落实该事务时任何打开的游标仍然打开；并且释放所有锁定，除保护打开的 WITH HOLD 游标的当前游标位置的锁定外。

如果回滚事务，则关闭所有打开的游标且释放所有锁定以及释放 LOB 定位器。

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。如果有使用调用级接口（DB2 CLI 或 ODBC）的已打包的应用程序，则在 `db2cli.ini` 配置文件中使用 OPTIMIZEFORNROWS 关键字，让 DB2 CLI 自动为每个已声明游标假设 WITH HOLD 子句。

相关概念:

- 第 75 页的『查询调整准则』
- 第 76 页的『有效的 SELECT 语句』

指定行分块来减少开销

行分块通过在单个操作中检索行*块*来减少游标的数据库管理器开销。

注: 指定的行块是内存中的许多页。它不是一个多维（MDC）表块，该块物理上映射为磁盘上的一个扩展数据块。

行分块级别由 BIND 或 PREP 命令的下列自变量指定:

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。

先决条件:

必须适当设置两个数据库管理器配置参数。两个值设置为内存的许多页。注意这些参数在块大小计算中使用的值。

- 数据库管理器配置参数 *aslheapsz* 指定本地应用程序的应用程序支持层堆大小。
- 数据库管理器配置参数 *rqrioblk* 指定数据库服务器上远程应用程序及其数据库代理进程之间的通信缓冲区大小。

过程:

要指定行分块:

1. 使用 *aslheapsz* 和 *rqrioblk* 配置参数的值来估计为每个块返回多少行。在两个公式中，*orl* 是以字节为单位的输出行长度。
   - 对于本地应用程序，使用下列公式:
     ```
     每块行数 = aslheapsz * 4096 / orl
     ```
     每页字节数为 4 096。
   - 对于远程应用程序，使用下列公式:
     ```
     每块行数 = rqrioblk / orl
     ```
2. 要启用行分块，对 PREP 或 BIND 命令中的 BLOCKING 选项指定适当的自变量。

   如果不指定 BLOCKING 选项，则缺省行分块类型为 UNAMBIG。对于命令行处理器和调用级接口，缺省行分块类型为 ALL。

注: 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

相关参考:

- 第 326 页的『aslheapsz － 应用程序支持层堆大小』
- 第 329 页的『rqrioblk － 客户机 I/O 块大小』

查询调整准则

遵循查询调整准则来细调应用程序中的 SQL 语句。准则将帮助您最小化系统资源的使用以及从大表和复杂查询返回结果所需的时间。

注: 优化器使用的优化级别可以消除某些细调整的要求，因为 SQL 编译器可以将 SQL 代码重写为更有效的格式。

注意，优化器选择的存取方案也受其它因素的影响，这些因素包括环境注意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程序测试，可发现哪些调整可以改善存取方案。

相关概念:

- 第 71 页的『限制选择语句的准则』
- 第 76 页的『有效的 SELECT 语句』
- 第 78 页的『复合 SQL 准则』
- 第 79 页的『字符转换准则』
- 第 79 页的『存储过程的准则』
- 第 80 页的『应用程序的并行处理』
- 第 214 页的『排序性能的准则』

相关任务:

- 第 74 页的『指定行分块来减少开销』

SQL 查询中的数据采样

数据库增长得非常庞大，并且对这些数据库的查询非常复杂，使得存取与查询相关的所有数据变得很不实际，有时也没有必要。在某些情况下，用户对查找总体趋势或模式感兴趣，在这种情况下，允许得到错误在一定范围内的大致答案。加速这种查询的一种方法对数据库的随机样本执行查询。DB2® 允许您在 SQL 查询中执行有效采样，从而在保留高精确度的同时潜在提高根据量值顺序进行大型查询的性能。

最常见的采样应用程序用于聚集查询（例如，AVG、SUM 和 COUNT），其中可从数据样本中获取聚集的合理精确结果。还可以使用采样来获取表中实际行的随机子集以用于审计，或者用于提高数据挖掘和分析任务的速度。

DB2 提供两种采样方法：行级别采样和块级别采样。

行级别 Bernoulli 采样:

行级别 Bernoulli 采样通过使用控制谓词（它包含样本中的每一行的概率为 P/100，排除样本中的每一行的概率为 1 - P/100）来获取表行的 P% 的样本。

行级别 Bernoulli 采样总是获取有效的随机样本，不管数据群集如何。但是，如果没有提供索引，这种采样的性能会很低，这是因为必须检索每行且对其应用采样谓词。如果没有任何索引，则在执行不采样的查询时 I/O 消耗会很大。如果有索引可用，则会提

高使用这种采样的性能，这是因为对索引叶子页内的 RIDS 应用了采样谓词。一般情况下，这要求每个所选 RID 一个 I/O，每个索引叶子页一个 I/O。

系统页级别采样:

系统页级别采样与行级别采样相似，但是它是对页采样而不是对行采样。将一页包含在样本中的概率为 P/100。如果包含了某页，就包含了该页中的所有行。

系统页级别采样的性能很好，因为对于包含在样本中的每一页只需要执行一次 I/O。与不采样相比，页级别采样根据量值顺序提高性能。但是，页级别采样与行级别采样比起来，聚集估计的准确性可能更差。当每个块有许多行时，或者当查询中引用的列在页中显示较高的群集度时，就很可能产生这种准确性的差异。

特定任务的最佳采样方法将由用户的时间约束和期望的准确度来确定。

指定采样方法:

要对表中的数据的随机样本执行查询，可以在 SQL 语句中使用 table-reference 子句的 TABLESAMPLE 子句。要指定采样的方法，使用关键字 BERNOULLI 或 SYSTEM。

BERNOULLI 关键字指定执行行级别 Bernoulli 采样。

SYSTEM 关键字指定执行系统页级别采样，除非优化器确定执行行级别 Bernoulli 采样反而更有效。

相关参考:

- 『Subselect』（*SQL Reference, Volume 1*）

有效的 SELECT 语句

因为 SQL 是一种灵活的高级语言，所以您可以编写几种不同的 SELECT 语句来检索同一数据。但是，对于不同的语句形式和不同的优化级别，性能可能相差很大。

考虑 SELECT 语句的下列准则:

- 仅指定需要的列。尽管用星号（*）指定所有列可能更简单，但会导致不必要的处理和返回不需要的列。
- 使用将回答集仅限制为您需要的那些行的谓词
- 当您需要的行数大大小于可能返回的总行数时，指定 OPTIMIZE FOR 子句。此子句影响存取方案的选择以及在通信缓冲区中分块的行数。
- 当要检索的行数很小时，仅指定 OPTIMIZE FOR *k* ROWS 子句。不需要 FETCH FIRST n ROWS ONLY 子句。但是，如果 *n* 值很大，并且您想快速获取前 *k* 行而对后续 *k* 行进行可能的延迟，则同时指定两个子句。通信缓冲区的大小基于 *n* 和 *k* 中的较小者。下列示例显示两个子句:

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

- 要利用行分块，指定 FOR READ ONLY 或 FOR FETCH ONLY 子句来改善性能。另外，因为在检索的行上从不挂起互斥锁定，所以并发性也会提高。也可发生附加

查询重写。指定 FOR READ ONLY 或 FOR FETCH ONLY 子句以及 BLOCKING ALL BIND 选项可以类似的方式改善联合系统中对昵称查询的性能。

- 对于将使用定位更新来更新的游标，指定 FOR UPDATE OF 子句以允许数据库管理器初始选择更适当的锁定级别并避免发生潜在的死锁。注意，FOR UPDATE 游标不能利用行分块。
- 对于将使用搜索更新来更新的游标，可以通过使用 FOR READ ONLY 和 USE AND KEEP UPDATE LOCKS 子句来对受影响的行强制执行 U 死锁来避免发生死锁并且仍然能够进行行分块。
- 尽可能避免数字数据类型转换。当比较值时，使用有相同数据类型的项可能更有效。如果需要转换，则可能由于精度受限制使结果不准确，并且由于运行时转换而使性能降低。

  如果可能的话，使用下列数据类型:
  - 对于较短的列，尽量使用字符而不是可变字符
  - 尽量使用整数，而不是浮点数或小数
  - 尽量使用日期时间，而不是字符
  - 尽量使用 Numeric™（数字），而不是字符
- 要减小排序操作发生的可能性，省略诸如 DISTINCT 或 ORDER BY 的子句或操作（如果这种操作不是必需的话）。
- 要检查一个表中是否存在某些行，选择某一单行。打开游标并访存一行，或执行单行（SELECT INTO）选择。记住，如果发现多行，则要检查是否有 SQLCODE -811 错误。

  除非您知道表很小，否则，不要使用下列语句来检查非零值:

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

  对于大表，对所有行计数会影响性能。
- 如果更新活动较少且表较大，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。对于配合主机变量使用的大型 IN 列表，循环主机变量的子集可能会提高性能。

下列建议只适用于存取几个表的 SELECT 语句。

- 使用连接谓词来连接表。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。索引也会改善 UPDATE 和 DELETE 语句的性能，这些语句包含存取几个表的 SELECT 语句。
- 如果可能的话，避免将表达式或 OR 子句与连接谓词一起使用，因为数据库管理器不能使用某些连接技术。因此，可能未选择最有效的连接方法。
- 在一个分区数据库环境中，如果可能的话，确保在连接列上对已连接的两个表进行了分区。

相关概念:

- 第 71 页的『限制选择语句的准则』
- 第 75 页的『查询调整准则』

复合 SQL 准则

要减小数据库管理器开销，可以将几条 SQL 语句组合为单个的可执行块。因为块中的 SQL 语句是可以单独执行的子语句，所以这种代码称为*复合 SQL*。除减小数据库管理器开销外，复合 SQL 也会减少必须通过网络为远程客户机传送的请求数。

有两种类型的复合 SQL:

- **基本**

  当所有子语句都成功完成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器接收到一个响应。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。将回滚在此块中对数据库所做的任何更改。

  DB2 Connect 不支持基本的复合 SQL

- **非基本**

  当所有子语句都完成时，应用程序将从数据库管理器接收到一个响应。不管前面的子语句是否成功完成，执行块中的所有子语句。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。

复合 SQL 在存储过程中（也称为 DARI 例程）和下列应用程序开发进程中受支持:

- 嵌入式静态 SQL
- DB2 调用级接口
- JDBC

动态复合 SQL 语句

动态复合语句由 DB2® 作为单个语句编译。此语句可以有效地用于短脚本，它需要很少的控制流逻辑，但需要重要的数据流。对于具有嵌套复杂控制流的较大构造，考虑使用 SQL 过程。

在动态复合语句中，可以在声明中使用下列元素:

- 子语句的变量声明中的 SQL 变量
- 基于条件声明的 SQLSTATE 值的子语句中的条件
- 一个或多个 SQL 过程语句

动态复合语句也可以使用几个流逻辑语句，如 FOR 语句、IF 语句、ITERATE 语句和 WHILE 语句。

如果在动态复合语句中发生错误，则回滚前面的所有 SQL 语句，并且不处理动态复合语句中剩余的 SQL 语句。

可以将动态复合语句嵌入到触发器、SQL 函数或 SQL 方法中，也可以通过动态 SQL 语句发出动态复合语句。可以动态准备此可执行语句。调用该语句不需要特权，但与此语句关联的授权标识必须具有在复合语句中调用 SQL 语句所必需的特权。

相关概念:

- 第 75 页的『查询调整准则』

字符转换准则

当应用程序和数据库不使用相同的代码页时，可能需要数据转换来在应用程序和数据库代码页之间映射数据。因为映射和数据转换需要附加开销，所以，如果应用程序和数据库使用相同的代码页或标识整理顺序，则可提高应用程序性能。

在下列情况下发生字符转换:

- 当客户机或应用程序在与它存取的数据库的代码页不同的代码页中运行时。

  在接收数据的数据库服务器上进行转换。如果数据库服务器接收数据，则字符转换是从应用程序代码页到数据库代码页。如果应用程序机器接收数据，则字符转换是从数据库代码页到应用程序代码页。
- 当导入（或装入）一个文件的客户机或应用程序在与导入（或装入）的文件不同的代码页中运行时。

字符转换不对下列对象发生:

- 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。
- DB2® 产品或平台，没有安装受支持的至／从 EUC（或 UCS-2）的转换函数。在此情况下，应用程序接收到 SQLCODE -332（SQLSTATE 57017）错误。

数据库管理器在转换多字节代码页时使用的转换函数和转换表或 DBCS 转换 API 取决于操作系统环境。

注: 在多字节代码页之间的字符串转换，如 DBCS 与 EUC 之间，可能会增大或减小字符串的长度。另外，分配给 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可能产生不同的结果。

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机变量需要特殊考虑，包括特殊的预编译器、应用程序性能和应用程序设计问题。

在日语和繁体中文的 EUC 代码页中的很多字符需要特殊方法，用于管理数据库和客户机应用程序对图形数据（需要双字节字符）的支持。使用 UCS-2 代码集存储和处理来自这些 EUC 代码页的图形数据。

相关概念:

- 第 165 页的『分析在何处对联合查询求值的准则』

相关参考:

- 『代码页 923 和 924 的转换表』（*《管理指南：计划》*）
- 『支持欧元的代码页的转换表文件』（*《管理指南：计划》*）

存储过程的准则

在许多情况重复的数据库应用程序环境中，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。例如，接收一组固定的数据，对一个数据库执行一组相同的多个请求或者返回一组固定的数据可能代表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。因为一个应用程序包含许多 SQL 语句，它需要许多传输来完成其工作。

但是，当一个数据库客户机使用一个封装许多 SQL 语句的存储过程时，对于整个过程它只需要两个传输。

存储过程通常在与数据库代理进程不同的进程中运行。这种独立性要求存储过程和代理进程通过路由器通信。但是，在代理进程中运行的一种特殊存储过程可以提高性能，尽管该存储过程具有毁坏数据和数据库的巨大风险。

这些有风险的存储过程是创建为*不受防护的*那些过程。对于不受防护的存储过程，未将该存储过程与数据库代理进程使用的数据库控制结构分开。如果某个 DBA 要确保该存储过程操作将不会意外地或有意地破坏数据库控制结构，则省略*不受防护的*选项。

由于有破坏数据库的风险，所以**仅**当您需要最大可能的性能效益时，才使用*不受防护的*存储过程。另外，绝对确保该过程编码得很严密，并且，在允许该过程作为不受防护的存储过程运行前已彻底测试过。如果在运行不受防护的存储过程时发生致命错误，则数据库管理器确定此错误是发生在应用程序中还是在数据库管理器代码中，并执行适当的恢复。

不受防护的存储过程可超过恢复的范围而毁坏数据库管理器，这可能会导致丢失数据和毁坏数据库的可能性。当运行不受防护的可信存储过程时，练习极端措施。在几乎所有情况下，应用程序的正确性能分析都将得到良好的性能，而不使用不受防护的存储过程。例如，触发器可以提高性能。

相关概念:

- 第 75 页的『查询调整准则』

应用程序的并行处理

DB2® 主要在对称多处理器（SMP）机器上支持并行环境，但在单处理器机器上也有限程度地支持它。在 SMP 机器中，多个处理器可以存取数据库，这使得将复杂 SQL 请求的并行执行分到各个处理器上。

要指定编译应用程序时要实现的并行度，使用 CURRENT DEGREE 专用寄存器或 DEGREE 绑定选项。*度*指同时执行的某个查询的部分数。在处理器数目和为并行度选择的值之间没有严格的关系。可以指定比机器上处理器数目多或少的数。甚至对于单处理器机器，您可以设置比以某些方式提高性能的并行度高的并行度。但是，注意，每个并行度都增加系统内存和 CPU 开销。

当使用查询的并行执行时，必须修改某些配置参数来优化性能。特别对于具有高并行度的环境，应该复查和修改控制共享内存量和预取的配置参数。

下列三个配置参数控制和管理分区内并行性。

- *intra_parallel* 数据库管理器配置参数启用或禁用并行性支持。
- *max_querydegree* 数据库配置参数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 绑定选项。
- *dft_degree* 数据库配置参数设置 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项的缺省值。

如果使用 DEGREE = ANY 来编译一个查询，则数据库管理器根据许多因素，包括处理器数目和此查询的特征，来选择分区内并行度。取决于这些因素和系统上的活动量，运行时使用的实际并行度可能低于处理器数。如果大量使用了系统，则在查询执行之前可能降低并行性。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，而这可能相反地影响其它数据库用户的性能。

要显示关于优化器所选择的并行度的信息，使用“SQL 说明设施”来显示存取方案。使用数据库系统监视器来显示关于运行时实际使用的并行度的信息。

非 SMP 环境中的并行性

您可以指定并行度而不必具有 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声明为 2 或更高值而受益。在这种情况下，处理器可能不必等待输入或输出任务完成，就可开始处理新的查询。但是，声明并行度为 2 或更高值并不直接控制一个单处理器机器上的 I/O 并行性。诸如 Load 实用程序这样的实用程序可以不依赖这种声明来控制 I/O 并行性。关键字 ANY 也可用来设置 *dft_degree* 数据库管理器配置参数。ANY 关键字允许优化器确定分区内并行度。

相关概念:

- 第 172 页的『说明工具』
- 第 158 页的『分区内并行性的优化策略』

相关参考:

- 第 414 页的『max_querydegree － 最大查询并行度』
- 第 413 页的『intra_parallel － 启用分区内并行性』
- 第 396 页的『dft_degree － 缺省并行度』

通过与 REOPT 绑定来提高性能

如果用于输入变量（例如，参数标记、主机变量和专用寄存器）的值超出了缺省过滤因子估计的预期范围，则在执行期间 SQL 查询的执行情况可能很差。用于不知道实际数据值的一些方案的缺省过滤因子会估计以下情况：使用实际数据值时，在运行时实际上将返回多少行。

REOPT 绑定选项指定是否让 DB2® 通过使用主机变量、参数标记和专用寄存器的值来优化存取路径。 REOPT 值分别由 BIND、PREP 和 REBIND 命令的下列自变量指定:

REOPT NONE
: 将不使用主机变量、参数标记或专用寄存器的实际值来对包含这些变量的给定 SQL 语句的存取路径进行优化；而是使用这些变量的缺省估计值。对此方案进行了高速缓存，并且将来会使用它。这是缺省行为。

REOPT ONCE
: 在第一次执行查询时，将使用主机变量、参数标记或专用寄存器的实际值来优化给定 SQL 语句的存取路径。对此方案进行了高速缓存，并且将来会使用它。

REOPT ALWAYS
: 将始终使用每次执行时已知道的主机变量、参数标记或专用寄存器的值来编译和重新优化给定 SQL 语句的存取路径。

相关概念:

- 『Effects of REOPT on static SQL』（*Application Development Guide: Programming Client Applications*）
- 『Effects of REOPT on dynamic SQL』（*Application Development Guide: Programming Client Applications*）

第 4 章 环境注意事项

除了在设计和编码应用程序时考虑的因素以外（在第 37 页的第 3 章，『应用程序注意事项』中描述了这些因素），某些环境因素可以影响为应用程序选择的存取方案。

有关直接影响 SQL 优化器的因素的更多信息，请参阅第 87 页的第 5 章，『系统目录统计信息』。

当调整应用程序和对环境进行更改时，重新绑定应用程序以确保使用最佳存取方案。

数据库分区组对查询优化的影响

在分区数据库中，优化器识别表的并置，并在确定查询的最佳存取方案时使用此并置。如果在连接查询中频繁涉及某些表，则应该将那些表划分到分区数据库中的各分区中，以便要连接的每个表中的行位于同一数据库分区中。在执行连接操作期间，两个已连接的表中数据的并置会防止将数据从一个分区移动至另一个分区。将两个表置于同一数据库分区组中以确保并置两个表中的数据。

在分区数据库中，根据表的大小，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、表的大小、那些表中数据的位置以及查询类型（例如，是否需要连接）都会影响查询的成本。

相关概念:

- 第 149 页的『分区数据库中的连接策略』
- 第 150 页的『分区数据库中的连接方法』
- 第 258 页的『分区数据库中的分区』

表空间对查询优化的影响

表空间的某些特征可影响 SQL 编译器选择的存取方案:

- 容器特征

  容器特征可显著影响查询执行期间的关联 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，它提供容器所需时间的估计（以毫秒计），在该时间后才将数据读入内存。此开销活动包括容器的 I/O 控制器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开销
- 对每个完整的旋转计算以毫秒计的旋转等待时间，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，结果如下:

```
(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，它提供将一页数据读入内存所需的时间估计（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 个字节／MB
- 乘以页的大小（以字节计）（例如，对于 4 KB 页为 4 096 个字节 ）

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 TRANSFERRATE 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 TRANSFERRATE 时必须考虑其它注意事项。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。

但是，如果组成容器的阵列中磁盘数目较大，则瓶颈可能不是在磁盘级别，而是在其它某个 I/O 子系统组件级别，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。而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 TRANSFERRATE 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。

您可以从硬件规格或通过实验来获取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，与特定于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。

通过这两个值提供的 I/O 成本信息会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于连接中的内部表和外部表。

- 预取

  当考虑从表空间存取数据的 I/O 成本时，优化器也考虑从磁盘预取数据页和索引页可能给查询性能带来的潜在影响。预取数据页和索引页可减少将数据读入缓冲池所需的开销和等待时间。

  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。缺省扩展数据块大小为 32 页（每页 4 KB），通常足够了。
  – 当创建表空间和／或使用 ALTER TABLESPACE 语句时，可设置 PREFETCHSIZE。缺省预取大小由 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。复查关于估算此参数的建议并按需要进行更改，以改进数据移动。

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，考虑重新绑定应用程序，并执行 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。

相关概念:

- 第 97 页的『目录统计信息表』
- 第 121 页的『SQL 编译器进程』
- 第 16 页的『DMS 表空间地址映射图表』

影响联合数据库的服务器选项

联合系统由 DB2® DBMS（联合数据库）和一个或多个数据源组成。应在发出 CREATE SERVER 语句时，向联合数据库标识数据源。当发出这些语句时，可以包括一些服务器选项，这些选项优化并控制涉及 DB2 和指定数据源的联合系统操作的各方面。要在以后更改服务器选项，使用 ALTER SERVER 语句。

注: 必须安装分布式连接安装选项，并将数据库管理器参数 *federated* 设置为 YES，然后才能创建服务器并指定服务器选项。

指定的服务器选项值影响查询下推分析、全局优化和联合数据库操作的其它方面。例如，在 CREATE SERVER 语句中，可以指定性能统计信息作为服务器选项值，如 *cpu_ratio* 选项，它指定数据源和联合服务器上 CPU 的相对速度。也可将 *cpu_ratio* 选项设置成指示源和联合服务器上数据 I/O 划分的相对速率的值。执行 CREATE SERVER 语句时，将此数据添加到目录视图 SYSCAT.SERVEROPTIONS 中，优化器使用它来制订数据源的存取方案。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），则使用 ALTER SERVER 语句以用此更改来更新 SYSCAT.SERVEROPTIONS。然后，优化器在下次选择数据源的存取方案时，使用新的信息。

相关参考:

- 『ALTER SERVER statement』（*SQL Reference, Volume 2*）
- 『SYSCAT.SERVEROPTIONS catalog view』（*SQL Reference, Volume 1*）
- 第 422 页的『federated － 联合数据库系统支持』
- 『CREATE SEQUENCE statement』（*SQL Reference, Volume 2*）

第 5 章 系统目录统计信息

存储在系统目录中的统计数据帮助优化器选择查询的最佳存取方案。确保执行 RUNSTATS 来更新此统计数据:

- 对于其内容持续更改的表以频繁的常规时间间隔。
- 在大量表行中添加或更改数据的每个操作之后。这样的操作包括批处理更新和添加行的数据装入。

目录统计信息

当 SQL 编译器优化 SQL 查询方案时，其决定主要受有关数据库表和索引大小的统计信息的影响。如果使用表和索引的特定列来选择行或连接表，则优化器也使用有关这些列中的数据分布的信息。优化器使用此信息来估计每个查询的备用存取方案的成本。

除表大小和数据分布信息外，还可以收集下列信息：索引的群集比率、索引中叶子页数目、溢出其原始页的表行数以及表中已填充的页数和空页数。使用此信息来决定何时重组表和索引。

当执行 RUNSTATS 实用程序时，收集本地数据库中特定表和索引的统计信息。收集的统计信息存储在系统目录表中。

仅对驻留在您执行该实用程序的分区上或包含表的数据库分区组中的第一个分区上的表分区收集统计信息。

注: 因为 RUNSTATS 实用程序不支持使用昵称，所以应以不同方式来更新联合数据库查询的统计信息。如果查询存取联合数据库，则对所有数据库中的表执行 RUNSTATS，然后删除并重新创建存取远程表的昵称，以使新的统计信息可用于优化器。

考虑以下技巧来提高 RUNSTATS 的效率和已收集的统计信息的有效性:

- 仅对用来连接表的列或 WHERE、GROUP BY 以及查询的类似子句中的列收集统计信息。如果对这些列建立了索引，则可以用 RUNSTATS 命令的 ONLY ON KEY COLUMNS 子句指定列。
- 为特定表和表中特定列定制 *num_freqvalues* 和 *num_quantiles* 的值。
- 使用 SAMPLE DETAILED 子句收集 DETAILED 索引统计信息，以减少对详细的索引统计信息执行的后台计算量。SAMPLE DETAILED 子句减少收集统计信息所需要的时间，并在大多数情况下产生足够的精度。
- 当创建已填写的表的索引时，添加 COLLECT STATISTICS 子句来在创建索引时创建统计信息。
- 当添加或除去了大量表行时，或如果更新了收集其统计信息的列中的数据，则再次执行 RUNSTATS 来更新统计信息。

- 因为 RUNSTATS 只收集有关单一分区的统计信息，所以，如果数据不是在所有分区中一致分发的，则统计信息将不太准确。如果您怀疑存在变形数据分发，则您可能想在执行 RUNSTATS 之前使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令在各分区之间再分发数据。

不收集分布统计信息:
- 当将 *num_freqvalues* 和 *num_quantiles* 配置参数设置为零（0）时
- 当了解数据分布时，例如，当每个数据值是唯一的时候。
- 当列是从未收集其统计信息的数据类型时。这些数据类型为 LONG、大对象（LOB）或结构化列。
- 对于子表中的行类型，不收集表级统计信息 NPAGES、FPAGES 和 OVERFLOW。
- 如果请求分位数分布，但该列中只有一个非空的值
- 对于已扩展的索引或已声明临时表

注: 可以对已声明临时表执行 RUNSTATS，但结果统计信息不会存储在系统目录中，因为已声明临时表不具有目录条目。但是，统计信息存储在为已声明临时表提供产品目录信息的内存结构中。因此，在某些情况下，对这些表执行 RUNSTATS 可能会有用。

相关概念:
- 第 97 页的『目录统计信息表』
- 第 101 页的『收集的统计信息』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 114 页的『用于对生产数据库建模的统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

相关任务:
- 第 90 页的『收集目录统计信息』

相关参考:
- 第 399 页的『num_freqvalues － 保留的高频值数目』
- 第 400 页的『num_quantiles － 列的分位数的数目』
- 『RUNSTATS Command』（*Command Reference*）

收集和分析目录统计信息

本节提供关于收集目录统计信息的准则和指示信息，以及关于分析收集的数据的一些提示以更好的了解数据分发、群集等。

收集和更新统计信息的准则

RUNSTATS 命令收集表和索引数据的统计信息，以给优化器提供准确信息进行存取方案选择。

注: RUNSTATS 仅收集执行该命令的分区上的表的统计信息。将此分区的 RUNSTATS 结果推广到其它分区。如果执行 RUNSTATS 的数据库分区不包含表分区，则将请求发送到数据库分区组中保留该表的分区的第一个数据库分区。

在下列情况下，使用 RUNSTATS 实用程序来收集统计信息:

- 当数据已装入表中且已创建适当的索引时。
- 当在表中创建新的索引时。如果自从上次在表中运行 RUNSTATS 以来尚未修改表，则只需要对新的索引执行 RUNSTATS。
- 当一个表已用 REORG 实用程序重组时。
- 当通过数据修改、删除和插入已大量更新表及其索引时。（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。）
- 在绑定性能非常重要的应用程序之前
- 当您想要比较当前和先前统计信息时。如果定期更新统计信息，则可以及早发现性能问题。
- 当预取量更改时。
- 当使用了 REDISTRIBUTE DATABASE PARTITION GROUP 实用程序时。

  注: 在先前版本的 DB2® 中，此命令使用了 NODEGROUP 关键字，而不是 DATABASE PARTITION GROUP 关键字。

要提高 RUNSTATS 性能并保存用来存储统计信息的磁盘空间，考虑仅指定应该收集其数据分布统计信息的列。

理论上，您应在运行统计信息之后重新绑定应用程序。如果查询优化器具有新的统计信息，则它可以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，可以运行 RUNSTATS 来每次只更新几个表和索引的统计信息，并轮流完成该组表。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，则在查询优化期间将发出警告消息（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。如果由于表上的活动产生了不一致性并在查询优化期间已检测到，则发出该警告消息。当发生此情况时，应该再次运行 RUNSTATS 来更新分布统计信息。

要确保索引统计信息和表同步，执行 RUNSTATS 来同时收集表和索引统计信息。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组统计信息不能在所有节点上都同步。

对生产系统调用 RUNSTATS 可能会对生产工作负载的性能产生负面影响。 RUNSTATS 实用程序现在支持调速选项，在执行较高级别的数据库活动期间，可以使用调速选项来限制执行 RUNSTATS 的性能影响。

相关概念:

- 第 87 页的『目录统计信息』
- 第 105 页的『优化器如何使用分布统计信息』
- 第 95 页的『自动进行统计信息收集』

相关任务:

- 第 90 页的『收集目录统计信息』
- 第 90 页的『收集特定列的分布统计信息』
- 第 91 页的『收集索引统计信息』

收集目录统计信息

收集表和索引的目录统计信息，以为优化器选择查询的最佳存取方案提供信息。

先决条件:

必须连接至包含表和索引的数据库并具有下列其中一个权限级别:

- sysadm
- sysctrl
- sysmaint
- dbadm
- 对表的 CONTROL 特权

过程:

要收集目录统计信息:

1. 连接至包含要收集其统计信息的表和索引的数据库。
2. 从 DB2 命令行，执行带适当选项 RUNSTATS 命令。这些选项允许您为对表和索引运行的查询定制要收集的统计信息。

   注: RUNSTATS 仅收集执行该命令的分区上的表的统计信息。将此分区的 RUNSTATS 结果推广到其它分区。如果执行 RUNSTATS 的数据库分区不包含表分区，则将请求发送到数据库分区组中保留该表的分区的第一个数据库分区。
3. 当 RUNSTATS 完成时，发出 COMMIT 语句以释放锁定。
4. 重新绑定存取表和索引的程序包，您已重新生成这些表和索引的统计信息。

要使用图形用户界面来指定选项并收集统计信息，使用"控制中心"。

相关概念:

- 第 87 页的『目录统计信息』
- 第 88 页的『收集和更新统计信息的准则』

相关任务:

- 第 90 页的『收集特定列的分布统计信息』
- 第 91 页的『收集索引统计信息』
- 第 217 页的『确定何时重组表』

相关参考:

- 『RUNSTATS Command』（*Command Reference*）

收集特定列的分布统计信息

为了 RUNSTATS 和后续查询方案分析的效率，可以仅收集查询在 WHERE、GROUP BY 和类似子句中使用的表列的分布统计信息。还可以收集关于列的组合组的基数统计信息。优化器使用这种信息来在它为引用组中列的查询估计选择性时检测列相关。

在下列步骤中，假定数据库为 **sales** 并包含表 **customers**，带有索引 **custidx1** 和 **custidx2**。

先决条件:

必须连接至包含表和索引的数据库并具有下列其中一个权限级别:

- sysadm
- sysctrl
- sysmaint
- dbadm
- 对表的 CONTROL 特权

注: RUNSTATS 仅收集执行该命令的分区上的表的统计信息。将此分区的 RUNSTATS 结果推广到其它分区。如果执行 RUNSTATS 的数据库分区不包含表分区，则将请求发送到数据库分区组中保留该表的分区的第一个数据库分区。

过程:

要收集特定列的统计信息:

1. 连接至 **sales** 数据库。
2. 取决于您的需求，在 DB2 命令行执行下列其中一个命令:
   - 要收集列 **zip** 和 **ytdtotal** 的分布统计信息:
     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON COLUMNS (zip, ytdtotal)
     ```
   - 要收集相同列的分布统计信息，但调整分布缺省值:
     ```
     RUNSTATS ON TABLE sales.customers
          WITH DISTRIBUTION ON
          COLUMNS (zip, ytdtotal NUM_FREQVALUES 50 NUM_QUANTILES 75)
     ```
   - 要收集建立了 **custidx1** 和 **custidx2** 索引的列的分布统计信息:
     ```
     RUNSTATS ON TABLE sales.customer
              ON KEY COLUMNS
     ```
   - 要仅收集特定列 **zip** 和 **ytdtotal** 以及包含 **region** 和 **territory** 的列组的表的列统计信息:
     ```
     RUNSTATS ON TABLE sales.customers
               ON COLUMNS (zip, (region, territory), ytdtotal)
     ```

还可以使用“控制中心”来收集分布统计信息。

相关概念:

- 第 97 页的『目录统计信息表』

相关任务:

- 第 90 页的『收集目录统计信息』
- 第 91 页的『收集索引统计信息』

收集索引统计信息

收集索引统计信息以允许优化器评估是否应该使用索引来解析查询。

在下列步骤中，假定数据库为 **sales** 并包含表 **customers**，带有索引 **custidx1** 和 **custidx2**。

先决条件:

必须连接至包含表和索引的数据库并具有下列其中一个权限级别:

- sysadm
- sysctrl
- sysmaint
- dbadm
- 对表的 CONTROL 特权

带 SAMPLED DETAILED 选项执行 RUNSTATS 需要 2MB 统计信息堆。将附加的 488 个 4K 页分配给为此附加内存需求设置的 *stat_heap_sz* 数据库配置参数。如果该堆看起来太小，则 RUNSTATS 在尝试收集统计信息之前返回一条错误。

过程:

要收集索引的详细统计信息:

1. 连接至 **sales** 数据库。
2. 取决于您的需求，在 DB2 命令行执行下列其中一个命令:
   - 要创建 **custidx1** 和 **custidx2** 的详细统计信息:
     ```
     RUNSTATS ON TABLE sales.customers AND DETAILED INDEXES ALL
     ```
   - 要创建两个索引的详细统计信息，但使用采样而不对每个索引条目执行详细计算:
     ```
     RUNSTATS ON TABLE sales.customers AND SAMPLED DETAILED INDEXES ALL
     ```
   - 要创建索引上的详细采样统计信息以及表的分布统计信息，以便索引和表统计信息一致:
     ```
     RUNSTATS ON TABLE sales.customers
              WITH DISTRIBUTION ON KEY COLUMNS
              AND SAMPLED DETAILED INDEXES ALL
     ```

还可以使用“控制中心”来收集索引和表统计信息。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 101 页的『收集的统计信息』
- 第 109 页的『详细的索引统计信息』

相关任务:

- 第 90 页的『收集目录统计信息』

收集有关表数据的样本的统计信息

查询优化器使用表统计信息来为给定的任何查询选择最佳存取方案，因此，使统计信息始终保持为最新的是很重要的，以准确反映表在任何给定时间的状态。对表执行的活动越多，收集统计信息的频率就应该越高。随着数据库的大小增大，找到用来收集统计信息的有效方法就显得更重要。对要收集统计信息的表数据进行随机采样可提高

RUNSTATS 性能。对于 I/O 受限制或 CPU 受限制的系统，性能可以得到极大的优化。样本越小，完成 RUNSTATS 的速度就越快。

从版本 8.2 开始，RUNSTATS 命令提供了通过使用 TABLESAMPLE 选项来收集表中的数据样本的统计信息的选项。此功能可提高收集统计信息的效率，原因是采样只使用一部分数据。并且，这些采样方法可确保准确性更高。

可以采用两种方法来指定要如何收集样本。BERNOULLI 方法对行级别的数据进行采样。在数据页的整个表扫描期间，会依次考虑每一行并根据数字参数指定的概率 P 来选择行。将只收集选择的这些行的统计信息。类似地，SYSTEM 方法对页级别的数据进行采样。因此，选择每一页的概率为 P，不选择每一页的概率为 1 - P/100。

页级别采样的性能很好，原因是对选择的每一页只需要执行一次 I/O。对于行级别采样，I/O 成本没有降低，原因是在整个表扫描过程中将扫描每个表页。但是，行级别采样还是得到了很大改进（尽管 I/O 次数没有减少），这是因为收集统计信息是很消耗 CPU 的。

在数据值的群集程度很高的情况下，行级别采样将比页级别采样提供更好的样本。与页采样相比，行级别样本集可能将更好地反映数据，原因是它将包括每个数据页中 P% 的行数。在页级别采样中，P% 的页数的所有行都将在样本集中。如果这些行是随机分布在表中的，则对行采样获得的统计信息的准确性与对页采样获得的统计信息的准确性差不多。

除非使用了 REPEATABLE 选项，否则每个样本都是在运行 RUNSTATS 命令时随机生成的。在使用 REPEATABLE 子句的情况下，将生成一个与最后一次执行带有 TABLESAMPLE 选项的 RUNSTATS 命令时生成的样本相同的样本。在想要为具有常量数据的表生成完全相同的统计信息的情况下，用户会发现此功能是很有用的。

相关概念:

- 第 75 页的『SQL 查询中的数据采样』

相关参考:

- 『RUNSTATS Command』（*Command Reference*）

使用统计信息概要文件收集统计信息

RUNSTATS 实用程序提供了一个选项来注册并使用统计信息概要文件，该文件是一组选项，这些选项指定要对特定表收集哪些统计信息，例如，表统计信息、索引统计信息或分布统计信息。

此功能部件简化了统计信息收集，它允许您存储在发出 RUNSTATS 命令时指定的一些选项，从而，可以对表反复收集相同的统计信息而不必重新输入命令选项。

无论实际上是否正在收集统计信息，都可以注册或更新统计信息概要文件。例如，要同时注册概要文件和收集统计信息，可发出带有 SET PROFILE 选项的 RUNSTATS 命令。要在实际上并没有收集统计信息的情况下只注册概要文件，可发出带有 SET PROFILE ONLY 选项的 RUNSTATS 命令。

要使用已经注册的统计信息概要文件来收集统计信息，发出 RUNSTATS 命令，并且只指定表名和 USE PROFILE 选项。

要了解在特定表的统计信息概要文件中当前指定了哪些选项，可以使用以下 SELECT 语句来查询目录表，其中 `tablename` 是想要获得其概要文件的表的名称：

```
SELECT STATISTICS_PROFILE FROM SYSIBM.SYSTABLES WHERE NAME = tablename
```

当注册了用户统计信息概要文件时，会同时构建与该概要文件相对应的 RUNSTATS 命令字符串，并将该命令字符串存储在 SYSIBM.SYSTABLES 目录表的 STATISTICS_PROFILE 列中。如果已注册的统计信息概要文件的命令字符串大小大于 STATISTICS_PROFILE 列，则 RUNSTATS 实用程序将构建最长为该列的大小的命令字符串，然后将被截断的字符串存储在 SYSTABLES.STATISTICS_PROFILE 列中。在系统目录中还维护了概要文件的内部版本，它不会被截断。

自动进行统计信息概要分析

统计信息概要文件还可以由 DB2® 自动进行统计信息概要分析的功能自动生成。当启用了此功能时，会收集有关数据库活动的信息，并将这些信息存储在查询反馈仓库中。可根据此数据来生成统计信息概要文件。启用此功能可减少有关哪些统计信息与特定工作负载相关的不确定问题，并且允许收集最少量的统计信息以提供最佳数据库工作负载性能。

可以将此功能与自动收集统计信息功能配合使用，后一项功能会根据自动生成的统计信息概要文件中包含的信息来自动安排统计信息维护。

要启用此功能，需要已经通过设置适当的配置参数启用了表的自动维护。AUTO_STATS_PROF 配置参数将激活查询反馈数据的收集，而 AUTO_PROF_UPD 配置参数将激活生成统计信息概要文件以供自动进行统计信息收集使用。

注： 自动生成统计信息概要文件只能以 DB2 串行方式激活，而对于联合环境、SMP 环境或 MPP 环境中的查询将禁用此功能。

生成统计信息概要文件最适用于具有下列特征的环境：运行使用许多谓词的大而复杂的查询、谓词列的数据中经常具有关联以及对几个表进行连接和分组。它不太适用于主要是具有事务性工作负载的环境。

还可以通过其它一些方法来使用此功能：

- 在测试环境中。在测试系统中，将 AUTO_STATS_PROF 和 AUTO_PROF_UPD 设置为 ON，在该环境中可以很容易地接受运行时监视的性能开销。当测试系统使用真实的数据和查询时，这将允许了解 RUNSTATS 的统计信息参数的正确关联和设置，然后将把这些关联和设置存储在统计信息概要文件中。可以再将这些概要文件传送至生产系统，生产系统对查询有利而不会导致任何监视开销。
- 解决生产环境中的特定查询的性能问题。如果检测到一组特定查询的性能问题，并且这些问题可能是由于错误的统计信息或关联造成的，则可以打开 AUTO_STATS_PROF 并执行某个时间段的目标工作负载。自动进行的统计信息概要分析将分析查询反馈并在 SYSTOOLS.OPT_FEEDBACK_RANKING* 表中创建建议。可以检查这些建议并按照建议来手工优化统计信息概要文件。要让 DB2 按照这些建议自动更新统计信息概要文件，应在打开 AUTO_STATS_PROF 时打开 AUTO_PROF_UPD。

注： 监视查询和将查询反馈数据存储在反馈仓库中将产生一定的性能开销。

创建查询反馈仓库： 反馈仓库由采用 SYSTOOLS 模式的三个表组成，它们存储有关在执行查询期间遇到的谓词的信息。这三个表是

OPT_FEEDBACK_QUERY、OPT_FEEDBACK_PREDICATE 和 OPT_FEEDBACK_PREDICATE_COLUMN。

要使用自动进行的统计信息概要分析，需要首先使用 SYSINSTALLOBJECTS 存储过程来创建查询反馈仓库。此存储过程是以 SYSTOOLS 模式创建和删除对象的常用存储过程。

调用 SYSINSTALLOBJECTS 存储过程，如下所示：

```
call SYSINSTALLOBJECTS ( toolname, action, tablespacename, schemaname)
```

其中：

toolname
: 指定要创建或删除其对象的工具的名称。在此例中为“ASP”或“AUTO STATS PROFILING”。

action
: 指定要执行的操作： “C”表示创建，而“D”表示删除。

tablespacename
: 将创建反馈仓库表的表空间的名称。此输入参数是可选的。如果未指定此参数，则将使用缺省用户空间。

schemaname
: 创建或删除对象时所使用的模式的名称。当前未使用此参数。

例如，要在表空间“A”中创建反馈仓库，输入：`call SYSINSTALLOBJECTS ("ASP", 'C', "A", "")`

相关概念:

- 第 95 页的『自动进行统计信息收集』

相关任务:

- 第 96 页的『使用自动进行统计信息收集』

自动进行统计信息收集

DB2® 优化器使用目录统计信息来确定给定的任何查询的最佳存取方案。如果有关表或索引的统计信息已过时或者不完整，则会导致优化器选择不是最佳的方案，并且会降低执行查询的速度。但是，决定要为给定的工作负载收集哪些统计信息是很复杂的事情，并且使这些统计信息保持最新是一项很花费时间的任务。

通过使用自动进行统计信息收集（它是 DB2 的“自动维护表”功能的一部分），可以让 DB2 确定工作负载需要哪些统计信息以及需要更新哪些统计信息。在启用了自动进行统计信息收集的情况下， DB2 将在后台自动运行 RUNSTATS 实用程序以确保收集并维护正确的统计信息。

通过使用下面几种方法，可以使自动进行统计信息收集对性能的影响降到最低：

- 统计信息收集是通过使用调速的 RUNSTATS 执行的。调速功能根据当前数据库活动来控制 RUNSTATS 实用程序消耗的资源数量：随着数据库活动的增加，RUNSTATS 实用程序的运行速度将变慢，从而减少了它的资源需求。

- 只收集最少量的统计信息以优化性能。这是通过使用统计信息概要分析来实现的，统计信息概要分析使用有关先前的数据库活动的信息来确定数据库工作负载需要哪些统计信息，以及这些统计信息对于数据库中的给定活动类型来说将变得过时的快慢程度。
- 只对具有高级别活动（根据更新、删除和插入的次数来衡量）的表进行统计信息收集。还对大型表（包含的页数超过 4000 页）进行采样以确定高级表活动是否真正更改了统计信息。只有确实更改了，才会收集这些大型表的统计信息。
- 自动将 RUNSTATS 实用程序安排在维护策略定义指定的最佳维护窗口中执行。此策略还指定在自动执行统计信息收集作用域内的一组表，这进一步减少了不必要的资源消耗。
- 当正在执行自动进行的统计信息收集时，受影响的表仍然可用于常规数据库活动（更新、插入和删除），就好象没有对表运行 RUNSTATS 一样。

相关概念:

- 第 93 页的『使用统计信息概要文件收集统计信息』

相关任务:

- 第 96 页的『使用自动进行统计信息收集』

相关参考:

- 第 426 页的『util_impact_lim – 实例影响策略』
- 第 401 页的『autonomic_switches – 自动维护开关』

使用自动进行统计信息收集

要进行高效率的数据存取和获得最佳工作负载性能，具有准确而完整的数据库统计信息是很关键的。使用“自动进行表维护”功能的自动进行统计信息收集功能来更新和维护相关的数据库统计信息。可以选择通过使用以下方法来在串行环境中增强此功能：收集查询数据并生成统计信息概要文件，这可以帮助 DB2 自动收集工作负载需要的一组准确的统计信息。此选项在 SMP、MPP 或联合环境中不可用。

过程:

可以使用图形用户界面工具或命令行界面来打开此功能。

- 要使用图形用户界面工具来将数据库设置为自动进行统计信息收集:
  1. 通过下面两种方式来打开“配置自动维护”向导：一种方法是在控制中心中右键单击数据库对象；另一种方法是在健康中心中右键单击想要配置自动进行统计信息收集的数据库实例。从弹出窗口中选择**配置自动维护**。
  2. 在此向导中，可以启用自动进行统计信息收集、指定想要自动收集其统计信息的表以及指定用于执行 RUNSTATS 实用程序的维护窗口。
  3. 可选：要启用自动生成统计信息概要文件，应使用命令行界面将下面两个配置参数设置为“ON”：
     - AUTO_STATS_PROF
     - AUTO_PROF_UPD
- 要使用命令行界面来将数据库设置为自动进行统计信息收集:
  1. 将下面的每个配置参数都设置为“ON”：
     - AUTO_MAINT

– AUTO_TBL_MAINT

– AUTO_RUNSTATS

2. 可选：要启用自动生成统计信息概要文件，应将下面两个配置参数都设置为“ON”：

– AUTO_STATS_PROF

– AUTO_PROF_UPD

相关概念:

- 第 93 页的『使用统计信息概要文件收集统计信息』
- 第 95 页的『自动进行统计信息收集』

相关参考:

- 第 401 页的『autonomic_switches － 自动维护开关』

收集的统计信息

本节列示目录统计信息表并描述这些表中字段的使用。统计信息表描述后面的节说明可以收集并存储在表中的数据的种类。

目录统计信息表

下列表提供关于包含目录统计信息的系统目录表和收集特定统计信息的 RUNSTATS 选项的信息。

表 18. 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES）

统计信息	描述	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 1）
ACTIVE_BLOCKS	对于 MDC 表，已占用块的总数	是	否

注:

1. 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 19. 列统计信息（SYSCAT.COLUMNS 和 SYSSTAT.COLUMNS）

统计信息	描述	RUNSTATS 选项	
		表	索引
COLCARD	列基数	是	是（注 1）
AVGCOLLEN	列的平均长度	是	是（注 1）
HIGH2KEY	列中的第二个最大值	是	是（注 1）
LOW2KEY	列中的第二个最小值	是	是（注 1）
NUMNULLS	列中的空值数	是	是（注 1）
SUB_COUNT	子元素的平均数	是	否（注 2）

表 19. 列统计信息（SYSCAT.COLUMNS 和 SYSSTAT.COLUMNS） (续)

统计信息	描述	RUNSTATS 选项	
		表	索引
SUB_DELIM_LENGTH	分隔每个子元素的每个定界符的平均值长度	是	否（注 2）

注:

1. 为索引键中的第一列收集列统计信息。
2. 这些统计信息提供关于列中数据的信息，这些列包含一系列用空格定界的子字段或子元素。只对类型为 CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 的单字节字符集字符串列收集 SUB_COUNT 和 SUB_DELIM_LENGTH 统计信息。

表 20. 多列统计信息（SYSCAT.COLGROUPS 和 SYSSTAT.COLGROUPS）

统计信息	描述	RUNSTATS 选项	
		表	索引
COLGROUPCARD	列组的基数	是	否

注: RUNSTATS 不收集以下两个表中列示的多列分布统计信息。但是，可以手工更新它们。

表 21. 多列分布统计信息（SYSCAT.COLGROUPDIST 和 SYSSTAT.COLGROUPDIST）

统计信息	描述	RUNSTATS 选项	
		表	索引
TYPE	F = 频率值 Q = 分位数值	是	否
ORDINAL	组中列的序数	是	否
SEQNO	序列号 *n*，它表示第 *n* 个 TYPE 值	是	否
COLVALUE	作为字符文字的数据值或空值	是	否

表 22. 多列分布统计信息 2（SYSCAT.COLGROUPDISTCOUNTS 和 SYSSTAT.COLGROUPDISTCOUNTS）

统计信息	描述	RUNSTATS 选项	
		表	索引
TYPE	F = 频率值 Q = 分位数值	是	否
SEQNO	序列号 *n*，它表示第 *n* 个 TYPE 值	是	否

表 22. 多列分布统计信息 2（SYSCAT.COLGROUPDISTCOUNTS 和 SYSSTAT.COLGROUPDISTCOUNTS） (续)

统计信息	描述	RUNSTATS 选项	
		表	索引
VALCOUNT	如果 TYPE = F，则 VALCOUNT 是由 SEQNO 标识的列组的 COLVALUE 的发生次数。 如果 TYPE = Q，则 VALCOUNT 是其值小于或等于具有此 SEQNO 的列组的 COLVALUE 的行数。	是	否
DISTCOUNT	如果 TYPE = Q，则此列包含小于或等于具有此 SEQNO 列组的 COLVALUE 的特异值数目。如果不可用，则为空。	是	否

表 23. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES）

统计信息	描述	RUNSTATS 选项	
		表	索引
NLEAF	索引叶子页数	否	是
NLEVELS	索引层数	否	是
CLUSTERRATIO	表数据的群集度	否	是（注 2）
CLUSTERFACTOR	更精确的群集度	否	详细的（注 1 和 2）
DENSITY	SEQUENTIAL_ PAGES 与该索引占用的页范围中页数的比率（百分比）（注 3）	否	是
FIRSTKEYCARD	在索引的第一列中的特异值数	否	是
FIRST2KEYCARD	在索引的前两列中的特异值数	否	是
FIRST3KEYCARD	在索引的前三列中的特异值数	否	是
FIRST4KEYCARD	在索引的前四列中的特异值数	否	是
FULLKEYCARD	在索引的所有列中的特异值数，不包括其所有 RID 标记为删除的类型 2 索引中的任何键值	否	是
PAGE_FETCH_PAIRS	不同缓冲区大小的页访存估计值	否	详细的（注 1 和 2）

表 23. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES） (续)

统计信息	描述	RUNSTATS 选项	
		表	索引
SEQUENTIAL_PAGES	按索引键顺序位于磁盘上的叶子页的数目，在这些叶子页之间很少有或没有大的间隔	否	是
AVERAGE_SEQUENCE_PAGES	可按顺序存取的平均索引页数。这是预取程序可以象按顺序的那样检测的索引页数	否	是
AVERAGE_RANDOM_PAGES	在顺序页存取中的平均随机索引页数	否	是
AVERAGE_SEQUENCE_GAP	序列之间的间隔	否	是
AVERAGE_SEQUENCE_FETCH_PAGES	可按顺序存取的平均表页数。这是预取程序在使用索引访存表行时可以象按顺序的那样检测的表页数	否	是（注 4）
AVERAGE_RANDOM_FETCH_PAGES	当使用索引访存表行时在顺序页存取之间的平均随机表页数	否	是（注 4）
AVERAGE_SEQUENCE_FETCH_GAP	当使用索引访存表行时序列之间的间隔	否	是（注 4）
NUMRIDS	索引中的记录标识（RID）数，包括类型 2 索引中已删除的 RID。	否	是
NUMRIDS_DELETED	索引中标记为删除的 RID 的总数，其上所有记录标识标记为删除的叶子页上的 RID 除外	否	是
NUM_EMPTY_LEAFS	索引中其上所有记录标识标记为删除的叶子页的总数	否	是

注:

1. 通过在 RUNSTATS 命令上指定 DETAILED 子句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 CLUSTERFACTOR 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. 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 来估计如果预取了索引页，那么平均可能从其它索引读取多少个不相关的页。
4. 当此表位于 DMS 表空间时，不能计算这些统计信息。
5. 即使调用命令时指定要收集统计信息，在 LOAD 或 CREATE INDEX 期间也不会收集预取统计信息。如果关闭了“顺序检测标志”配置参数（seqdetect），则也不收集预取统计信息。

表 24. 列分布统计信息（SYSCAT.COLDIST 和 SYSSTAT.COLDIST）

统计信息	描述	RUNSTATS 选项	
		表	索引
DISTCOUNT	如果 TYPE 是 Q，则是小于或等于 COLVALUE 统计信息的特异值数目	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	序列号的频率排序，它帮助唯一标识该表中的行	分布	否
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键列收集 DISTCOUNT。

相关概念:

- 第 87 页的『目录统计信息』
- 第 101 页的『收集的统计信息』
- 第 112 页的『用户定义的函数的统计信息』
- 第 114 页的『用于对生产数据库建模的统计信息』

收集的统计信息

当对表或表及其关联索引执行 RUNSTATS 实用程序时，下列几种统计信息始终存储在系统目录表中:

对于表和索引:
- 正在使用的页数
- 包含行的页数
- 溢出的行数
- 表中的行数（基数）
- 对于 MDC 表，包含数据的块数

对于表中的每列和索引键中的第一列:
- 列的基数
- 列的平均长度
- 列中次高值
- 列中次低值
- 列中的空值数

对于指定的列组:

- 列组的基于时间戳记的名称
- 列组的基数

仅对于索引:
- 叶子页数
- 索引层数
- 此索引的表数据的群集度
- 磁盘上按索引键顺序的叶子页与索引占用的页范围中页数的比率
- 在索引的第一列中的特异值数
- 在索引的前二、三和四列中的特异值数
- 在索引的所有列中的特异值数
- 按索引键顺序位于磁盘上的叶子页的数目，在这些叶子页之间很少有或没有大的间隔
- 页中所有 RID 标记为删除的页数
- 页中标记为删除的 RID 数，并非页中所有 RID 都标记为删除

如果请求索引的详细统计信息，则还可以存储关于表对索引的群集度的更详细信息以及不同缓冲区大小的页访存估计。

还可以收集关于表和索引的下列几种统计信息:
- 数据分布统计信息

  优化器使用数据分布统计信息来估计表的有效存取方案，在这些表中数据不是均匀分布的，且列具有大量重复值。
- 详细索引统计信息

  优化器使用详细索引统计信息来确定通过索引存取表的有效程度。
- 子元素统计信息

  优化器对 LIKE 谓词使用子元素统计信息，特别是对于搜索嵌入字符串中的模式的那些谓词，如 `LIKE %disk%`。

相关概念:
- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』

相关任务:
- 第 90 页的『收集目录统计信息』
- 第 90 页的『收集特定列的分布统计信息』
- 第 91 页的『收集索引统计信息』

分布统计信息

可以收集两种数据分布统计信息:
- 频率统计信息

  这些统计信息提供关于 *num_freqvalues* 数据库配置参数的值所指定级别的具有最高重复的数目和次高重复值的数目等的列和数据值的信息。要禁用收集高频值统计信息，将 *num_freqvalues* 设置为 0。

  还可以将 *num_freqvalues* 设置为每个表和特定列的 RUNSTATS 选项。

- 分位数统计信息

  这些统计信息提供关于与其它值相比如何分布数据值的信息。称为 K 分位数，这些统计信息表示值 V，至少 K 个值位于该值或该值以下。可以通过按升序排序值来计算 K 分位数。K 分位数值是从范围的低端起第 K 个位置中的值。

  要指定应该将列数据值分组成的部分数，将 *num_quantiles* 数据库配置参数设置为 2 和 32,767 之间的某个值。缺省值为 20，它确保对任何相等或小于或大于谓词的优化器估计误差最大为正或负 2.5%，而对任何 BETWEEN 谓词的最大误差为正或负 5%。要禁用分位数统计信息收集，将 num_quantiles 设置为 0 或 1。

  可以设置每个表和特定列的 *num_quantiles*。

注: 如果指定较大的 *num_freqvalues* 和 *num_quantiles* 值，则当执行 RUNSTATS 时，需要更多 CPU 资源和内存，在 *stat_heap_sz* 数据库配置参数中指定它们。

何时收集分布统计信息

要决定是否应创建和更新给定表的分布统计信息，考虑以下两个因素:

- 应用程序是否使用静态或动态 SQL。

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

- 列中的数据是否是均匀分布的。

  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，则创建分布统计信息:

```
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;
```

可能发生两种类型的不均匀数据分布（可能一起发生）:

- 可能按一个或多个子间隔群集数据，而不是均匀地在最高和最低数据值之间分布。考虑以下列，其中，数据在范围（5,10）内群集:

C1
0.0
5.1
6.3
7.1
8.2
8.4
8.5
9.1
93.6
100.0

  分位数统计信息帮助优化器处理这种数据分布。

  要帮助确定是否不是均匀分布的列数据，执行诸如以下示例的查询:

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

- 重复数据值可能经常出现。考虑使用下列频率分布数据的列:

数据值	频率
20	5
30	10
40	10
50	25
60	25
70	20
80	5

要帮助优化器处理重复值，创建分位数和高频值统计信息。

何时仅收集索引统计信息

在下列情况下，您可以只根据索引数据来收集统计信息:
- 自运行 RUNSTATS 实用程序以来已创建了新索引，且您不想再次收集有关该表数据的统计信息。
- 存在影响索引的第一列的许多数据更改。

要指定哪个级别的统计精度

要确定存储分布统计信息使用的精度，指定数据库配置参数 *num_quantiles* 和 *num_freqvalues*。还可以指定这些参数作为收集表或列的统计信息时的 RUNSTATS 选项。这些值设置得越高，RUNSTATS 创建和更新分布统计信息时使用的精度越大。但是，精度越大，在 RUNSTATS 执行期间和在目录表中所需要的存储器中需要使用的资源就越多。

对于大多数数据库，为 *num_freqvalues* 数据库配置参数指定 10 到 100 之间的数。理论下，应创建高频值统计信息，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。数据库管理器收集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。如果需要只收集分位数统计信息，则将 *num_freqvalues* 设置为 0。

要设置分位数的数目，指定 20 到 50 之间作为 *num_quantiles* 数据库配置参数的设置。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，在谓词是任何其它类型的范围谓词（<、<=、> 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，至少指定 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。如果只需要高频值统计信息，则将 *num_quantiles* 设置为 0。如果将此参数设置为“1”，则因为值的整个范围适合一个分位数，因此不收集分位数统计信息。

相关概念:
- 第 87 页的『目录统计信息』

- 第 97 页的『目录统计信息表』
- 第 105 页的『优化器如何使用分布统计信息』
- 第 106 页的『分布统计信息使用的扩展示例』

相关任务:

- 第 90 页的『收集目录统计信息』
- 第 90 页的『收集特定列的分布统计信息』

相关参考:

- 第 399 页的『num_freqvalues － 保留的高频值数目』
- 第 400 页的『num_quantiles － 列的分位数的数目』

优化器如何使用分布统计信息

优化器使用分布统计信息，以便更好地估计各种可能的存取方案的成本来满足查询。

如果不使用 WITH DISTRIBUTION 子句执行 RUNSTATS，则目录统计信息表仅包含关于表大小和表中最高和最低值的信息、表对于任何索引的群集度以及索引列中的特异值的数目。

除非具有关于低和高值之间的值分布的更多信息，否则，优化器假定数据值是均匀分布的。如果数据值彼此差异较大，在范围中的某些部分中群集，或包含许多重复值，则优化器将选择次于最优的存取方案。

考虑下列示例:

优化器需要估计包含满足等式或范围谓词的列值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
```

假定 C1 和 C2 上都有一个索引。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'`。因为执行查询的主要成本通常是检索行的成本，所以最好的方案是需要最少检索的方案。选择此方案需要估计满足每个谓词的行数。

当分布统计信息不可用但已对表执行 RUNSTATS 时，对优化器可用的仅有的信息为列的第二个最高的数据值（HIGH2KEY）、第二个最低的数据值（LOW2KEY）、特异值数目（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）中的几个子间隔内，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。

相关概念:

- 第 87 页的『目录统计信息』
- 第 102 页的『分布统计信息』
- 第 106 页的『分布统计信息使用的扩展示例』

相关任务:

- 第 90 页的『收集特定列的分布统计信息』

分布统计信息使用的扩展示例

要了解优化器可以如何使用分布统计信息，首先考虑包含格式 `C1 = KEY` 的等式谓词的查询。

高频值统计信息示例

如果高频值统计信息是可用的，优化器可以使用这些统计信息来选择适当的存取方案，如下所示:

- 如果 KEY 是 N 个最高频值的其中一个，则优化器使用存储在该目录中的 KEY 的频率。
- 如果 KEY 不是 N 个最高频值的其中一个，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，CARD 是表中的行数，COLCARD 是列的基数，NUM_FREQ_ROWS 是具有 N 个最高频值的其中一个最高频值的总行数。

例如，考虑具有如下数据值和频率的一列（C1）:

数据值	频率
1	2
2	3
3	40
4	4
5	1

如果仅基于最高频值（即，N = 1）的高频值统计信息可用，则对于此列，表中的行数为 50，列基数为 5。对于谓词 `C1 = 3`，刚好有 40 行满足它。如果优化器假定数据均匀分布，则它估计满足该谓词的行数为 50/5 = 10，错误率为 -75%。如果优化器可以使用高频值统计信息，则该行数估计为 40，且没有错误。

考虑另一个示例，其中，有 2 行满足谓词 C1 = 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)
```

并且错误降低了一个数量级，如下所示:

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

分位数统计信息示例

分位数统计信息的下列说明使用术语“K 分位数”。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。要计算 K 分位数，将该列中的所有行按数据值递增的顺序来排序；K 分位数是已排序的列中第 K 行中的数据值。

如果分位数统计信息可用，则优化器可以更好地估计满足一个范围谓词的行数，如以下示例所示。考虑包含下列值的一列（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，如下所示:

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，刚好有 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 替换 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%。

如果分位数统计信息可用，则优化器通过如下方法来估计满足此同一谓词（C <= 8.5）的行数：查找作为其中一个分位数的最高值的 8.5，并通过使用 K 的对应值 7 来作为估计行数。在此情况下，错误缩小为 0。

现在考虑谓词 C <= 10。刚好有 8 行满足此谓词。如果优化器必须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 -87.5%。

与上一个示例不同，值 10 不是存储的其中一个 K 分位数。但是，优化器可以使用分位数来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 的行数。在以上示例中，r_1 = 7。要估计 r_2，优化器使用线性插值：

```
       10 - 8.5
r_2 *= ---------- x（具有值 > 8.5 和 <= 100.0 的行数）
       100 - 8.5
       10 - 8.5
r_2 *= ---------- x (10 - 7)
       100 - 8.5
       1.5
r_2 *= ---- x (3)
       91.5
r_2 *= 0
```

最终估计是 r_1 + r_2 *= 7，而错误率仅为 -12.5%。

在以上示例中分位数提高了估计的准确性，因为实数据值“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和频率的一列：

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 K 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，您可看到刚好有 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
```

使用线性插值，r_2 估计如下:

```
       30 - 20
  ------- x (# 值 > 20 且 <= 40 的行)
  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。

相关概念:

- 第 87 页的『目录统计信息』
- 第 102 页的『分布统计信息』
- 第 105 页的『优化器如何使用分布统计信息』
- 第 118 页的『用于手工更新分布统计信息的规则』

相关任务:

- 第 90 页的『收集特定列的分布统计信息』

详细的索引统计信息

如果使用 DETAILED 子句对索引执行 RUNSTATS，则收集关于索引的统计信息，这允许优化器根据各种缓冲池大小估计将需要多少数据页访存。此附加信息帮助优化器作出通过索引存取表的成本的更好估计。

注: 当收集详细索引统计信息时，RUNSTATS 花费较长时间，并需要更多内存和 CPU 处理。SAMPLED DETAILED 选项（对于此信息仅计算统计上有意义的条目数）需要 2MB 统计信息堆。将附加的 488 个 4K 页分配给为此附加内存需求设置的 *stat_heap_sz* 数据库配置参数。如果堆看起来太小，则 RUNSTATS 在尝试收集统计信息之前返回一条错误。

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。

DETAILED 统计信息提供关于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所需要的物理 I/O 的数目的简明信息。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，并收集缺页故障发生频率的估计值。例如，如果只有一个缓冲区页可用，则索引引用的每个新页导致缺页故障。在更坏的情况下，每行可能引用不同的页，导致最多与已索引的表中的行相同的 I/O 数。另一种极端情况是，当该缓冲区大得足以容纳整个表（但不超过最大缓冲区大小）时，所有表的每页都读取一次。结果，物理 I/O 的数目是缓冲区大小的单调非递增函数。

统计信息还提供更精确估计表行对于索引顺序的群集度。涉及索引的群集的表行越少，通过索引存取表行所需要的 I/O 就越多。优化器在估计通过索引存取表的成本的同时考虑缓冲区大小和群集度。

当查询引用不包括在索引中的列时，应收集 DETAILED 索引统计信息。此外，在下列情况时，也应使用 DETAILED 索引统计信息:

- 表有多个具有不同群集度的非群集索引
- 各键值的群集度不均匀
- 索引中的值被不均匀地更新

如果没有先前的了解或不强制不同缓冲区大小下的索引扫描并接着监视引起的物理 I/O，则很难估计这些情况。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 统计信息并检查这些统计信息。如果得到的 PAGE_FETCH_PAIRS 是非线性的，则保留它们。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』

相关任务:

- 第 90 页的『收集目录统计信息』
- 第 91 页的『收集索引统计信息』

子元素统计信息

如果包含由空格分开的子字段或子元素的列包含在表中，并且查询在 WHERE 子句中引用这些列，则应该收集子元素统计信息以确保最好的存取方案。

例如，假设数据库包含表 DOCUMENTS，表中每行描述一个文档，并假设在 DOCUMENTS 中有一个称为 KEYWORDS 的列，该列中包含为了文本检索目的而与该文档有关的关键字的列表。在 KEYWORDS 中的值可能如下所示:

```
'database simulation analytical business intelligence'
'simulation model fruit fly reproduction temperature'
'forestry spruce soil erosion rainfall'
'forest temperature soil precipitation fire'
```

在本示例中，每列值由 5 个子元素构成，其中每个都是一个单词（关键字），且由一个空格与其它的词分开。

用于查询时，使用 %match_all character 在这些列上指定 LIKE 谓词:

```
SELECT .... FROM DOCUMENTS WHERE KEYWORDS LIKE '%simulation%'
```

它经常有利于优化器了解某些关于列的子元素结构的基本统计信息。

当执行带有 LIKE STATISTICS 子句的 RUNSTATS 时，将收集下列统计信息:

SUB_COUNT
子元素的平均数。

SUB_DELIM_LENGTH
分隔每个子元素的每个定界符的平均长度，其中，在本上下文中，定界符是一个或多个连续的空格字符。

在 KEYWORDS 列示例中，SUB_COUNT 是 5，而 SUB_DELIM_LENGTH 是 1，因为每个定界符是一个单一的空格字符。

DB2®_LIKE_VARCHAR 注册表变量影响优化器处理以下格式谓词的方式:

```
COLUMN LIKE '%xxxxxx'
```

其中 *xxxxxx* 代表任何字符串；即，搜索值从 % 字符开始的任何 LIKE 谓词。(它可能以 % 字符结束或可能不以该字符结束)。将这些称为 "通配符 LIKE 谓词"。对于所有谓词，优化器必须估计有多少行与该谓词匹配。对于通配符 LIKE 谓词，优化器假设要匹配的 COLUMN 包含并置在一起的一系列元素，并且它基于字符串长度估计每个元素的长度，不包括前导和结尾 % 字符。

要检查子元素统计信息的值，查询 SYSIBM.SYSCOLUMNS。例如:

```
select substr(NAME,1,16), SUB_COUNT, SUB_DELIM_LENGTH
   from sysibm.syscolumns where tbname = 'DOCUMENTS'
```

注: 如果使用 LIKE STATISTICS 子句，RUNSTATS 可能花较长时间。例如，如果未使用 DETAILED 和 DISTRIBUTION 选项，则 RUNSTATS 在具有 5 个字符列的表上可能运行 15% 到 40% 以及更长时间。如果指定 DETAILED 或 DISTRIBUTION 选项，则即使开销的绝对数量相同，但开销百分比较少。如果您正在考虑使用此选项，则应该估计对提高查询性能的开销。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』

相关任务:

- 第 90 页的『收集目录统计信息』
- 第 90 页的『收集特定列的分布统计信息』

用户可以更新的目录统计信息

本节描述用户可以手工更新的目录统计数据并提供这样的手工更改的准则。在某些情况下，RUNSTATS 不收集此统计数据，而必须手工添加它们。在其它情况下，可以将收集的统计数据导入测试数据库并为特定目的更改收集的数据，如为实验目的模拟生产数据库。

警告: 在生产数据库中，不要手工更新由 RUNSTATS 收集的数据。可能会导致严重的性能问题。

用户定义的函数的统计信息

要创建用户定义的函数（UDF）的统计信息，编辑 SYSSTAT.FUNCTIONS 目录视图。如果 UDF 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。RUNSTATS 实用程序不收集 UDF 的统计信息。如果统计信息不可用，则统计信息列的值是 -1，且优化器使用缺省值，这些缺省值假定一个简单的 UDF。

下表提供有关统计列的信息，您可以为该列提供估计以提高性能:

表 25. 函数统计信息（SYSCAT.FUNCTIONS 和 SYSSTAT.FUNCTIONS）

统计信息	描述
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。
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，可以按如下所示设置统计信息列:

- 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
- 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。
- PERCENT_ARGBYTES: 设置为 100，指示要转换整个输入字符串
- INITIAL_INSTS、IOS_PER_INVOC、IOS_PER_ARGBYTE 和 INITIAL_IOS: 每个都设置为 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，这导致当找到第一个自变量时，搜索第二个自变量的一半（平均值）。

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，如，记录设置暂存区区域的成本。

要获得有关一个用户定义的函数使用的 I/O 和指令的信息，您可以使用您的编程语言编译器或可用于您的操作系统的监视工具所提供的输出。

相关概念:

- 第 97 页的『目录统计信息表』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

用于建模和 what-if 计划的目录统计信息

可以更改系统目录中的统计信息，以便它不反映表和索引的实际状态，但允许您检查对数据库的各种可能的更改以用于规划目的。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

不要手工更新生产系统的统计信息。如果这样做，则优化器可能不选择包含动态 SQL 的生产查询的最佳存取方案。

需求

您必须具有对数据库的显式 DBADM 权限，才能修改表和索引及其组件的统计信息。即，您的用户标识被记录为在 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。DBADM 可以查看所有用户的统计信息，并可以对在 SYSSTAT 模式中定义的视图执行 SQL UPDATE 语句，来更新这些统计列的值。

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。如果您不具有 DBADM 权限，当您具有每个对象的下列特权时，可以更改单个数据库对象的统计信息:

- 对表的显式 CONTROL 特权。也可更新这些表的列和索引的统计信息。
- 对联合数据库系统中的昵称的显式 CONTROL 特权。也可更新这些昵称的列和索引的统计信息。注意，更新仅影响本地元数据（不会更改数据源表统计信息）。这些更新只影响由 DB2® 优化器生成的全局存取策略。
- 用户定义的函数（UDF）的所有权。

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'
```

当手工更新目录统计信息时，您必须小心。任意更改可以严重改变后续查询的性能。即使在正用于测试或建模的非生产数据库中，您可以使用下列任何一种方法来刷新应用于这些表的更新并使统计信息处于一致状态:

- 对已更改的工作单元执行 ROLLBACK（假定尚未落实该工作单元）。
- 使用 RUNSTATS 实用程序来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。）
- 将目录统计信息替换为更改前它们所包含的数据。仅当您在进行任何更改之前使用了 *db2look* 工具来捕获统计信息时，此方法才成为可能 。

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。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。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 112 页的『用户定义的函数的统计信息』
- 第 114 页的『用于对生产数据库建模的统计信息』

用于对生产数据库建模的统计信息

有时，您可能想要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。

可对生产数据库运行生产工具 *db2look*，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它对象。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。可以在重新创建数据库和设置统计信息的脚本中一起使用 -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更多信息，在命令行上输入如下命令:

```
db2look -h
```

“控制中心”还为 *db2look* 实用程序提供了一个称为“生成 SQL － 对象名”的界面。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。使用控制中心与使用 *db2look* 命令的一个不同之处是，在单个调用中前者只能完成一个表分析而后者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 或 z/OS 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 或 z/OS 对象并在 “DB2® 通用数据库”（UDB）数据库中重新创建它们，则此实用程序非常有用。

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 或 z/OS 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 或 z/OS 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 或 z/OS 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 或 z/OS 版有另一项名为 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

   其它没有 OS/390 或 z/OS 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 或 z/OS 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”的条目才适用于 DB2 UDB 版，这些就是将被考虑到的条目。

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

此外，DB2 OS/390 版 SYSIBM.SYSCOLUMNS 中没有 SEQNO 列。因为这是 DB2 UDB 版所必需的，所以 db2look 自动生成一个。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

相关参考:

- 『db2look - DB2 Statistics and DDL Extraction Tool Command』(*Command Reference*)

用于手工更新目录统计信息的一般规则

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（一列中的特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，则会接收到一条错误（因为 CARD 应小于 COLCARD）。正确的顺序是首先更新 SYSCAT.COLUMNS 中的 COLCARD，然后更新 SYSSTAT.TABLES 中的 CARD。如果您想把 COLCARD 从 100 增大到 250，把 CARD 从 200 增大到 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项统计信息之间存在冲突时，则发出错误消息。但是，并不是在发生冲突时始终都会发出错误消息。在某些情况下，该冲突难于检测及在错误消息中报告，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. Numeric™（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，子表的表级别统计信息 NPAGES、FPAGES 和 OVERFLOW 是不可更新的。

相关概念:

- 第 97 页的『目录统计信息表』
- 第 112 页的『用户定义的函数的统计信息』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』

- 第 114 页的『用于对生产数据库建模的统计信息』
- 第 117 页的『用于手工更新列统计信息的规则』
- 第 118 页的『用于手工更新分布统计信息的规则』
- 第 118 页的『用于手工更新表和昵称统计信息的规则』
- 第 119 页的『用于手工更新索引统计信息的规则』

用于手工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。

- 当手工更新 SYSSTAT.COLUMNS 中的 HIGH2KEY 和 LOW2KEY 时，遵循生成的值的行为:
  - HIGH2KEY 和 LOW2KEY 的值必须是相应的用户列的数据类型的有效值。
  - HIGH2KEY 和 LOW2KEY 值的长度必须是 33 或目标列数据类型的最大长度这两者中的较小者，后者不包括目标列数据类型的附加引号，引号可使字符串的长度达到 68。这意味着在确定 HIGH2KEY 和 LOW2KEY 值时将只考虑相应用户列中的值的前 33 个字符。
  - 存储 HIGH2KEY/LOW2KEY 值之后，可以在 UPDATE 语句的 SET 子句中使用它们，但是进行成本计算时无须考虑它们。对于字符串，这意味着在字符串的开头和结尾要添加单引号，并且对字符串中已经存在的每个引号另外添加了引号。在下表中，提供了用户列值以及它们在 HIGH2KEY 和 LOW2KEY 中的相应值的示例。

表 26. 数据类型的 HIGH2KEY 和 LOW2KEY 值

用户列中的数据类型	用户数据	相应的 HIGH2KEY 和 LOW2KEY 值
INTEGER	−12	−12
CHAR	abc	'abc'
CHAR	ab'c	'ab''c'

  - 只要相应列中具有 3 个以上的特异值，HIGH2KEY 就应该大于 LOW2KEY。
- 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）。
- 一列中的空值数（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）。
- 具有如下数据类型的列不支持统计信息：LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 和 DBCLOB。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

用于手工更新分布统计信息的规则

手工更新分布统计信息只用于对生产数据库建模或对手工构造的数据库执行 what-if 测试。不更新生产数据库中的分布统计信息。

确保目录中的所有统计信息是一致的。尤其是对于每一列，常用数据统计信息和分位数的目录条目必须满足下列约束:

- 高频值统计信息（在 SYSSTAT.COLDIST 目录中）。这些约束包括:
  - 列 VALCOUNT 中的值必须随 SEQNO 值的递增而不变或降低。
  - 列 COLVALUE 中的值的数目必须小于或等于该列中的特异值数目，特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。
- 分位数（在 SYSSTAT.COLDIST 目录中）。这些约束包括:
  - COLVALUE 列中的值必须在 SEQNO 的值增大时不变或降低
  - VALCOUNT 列中的值必须在 SEQNO 的值增大时增大
  - COLVALUE 列中的最大值必须在列 VALCOUNT 中有对应的一个条目，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 中的每一条目必须乘以 F。类似地，要将分位数上调 F 倍，列 VALCOUNT 中的每一条目必须乘以 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并导致不可预测的性能。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

用于手工更新表和昵称统计信息的规则

在 SYSTAT.TABLES 中您可以更新的仅有的统计值为: CARD、FPAGES、NPAGES 和 OVERFLOW，而对于 MDC 表，该值为 ACTIVE_BLOCKS。注意:

1. CARD 必须大于或等于对应于该表的 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“访存”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“访存”值（假定此统计信息与该索引相关）。

当在联合数据库系统中工作时，在手工提供或更新有关远程视图的昵称统计信息时务必小心。统计信息（例如，此昵称将返回的行数）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本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。考虑创建基于昵称的局部视图，以便 DB2 优化器知道如何更准确地推导出该视图的成本。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

用于手工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 统计信息中必须刚好有 11 对数。
   - PAGE_FETCH_PAIRS 的缓冲区大小项的值必须呈升序。
   - | 在 PAGE_FETCH_PAIRS 条目中的任何缓冲区大小值不能大于 MIN(NPAGES, 524287)（对于 32 位操作系统）或 MIN(NPAGES, 2147483647)（对于 64 位操作系统），其中 NPAGES 是相应表（在 SYSSTAT.TABLES 中）中的页数。
   - PAGE_FETCH_PAIRS 的“访存”条目的值必须呈降序，且任何个别的“访存”条目不能小于 NPAGES。PAGE_FETCH_PAIRS 条目中的“访存”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（在 SYSSTAT.COLUMNS 中）。
   - 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。

相关概念:

- 第 87 页的『目录统计信息』
- 第 97 页的『目录统计信息表』
- 第 113 页的『用于建模和 what-if 计划的目录统计信息』
- 第 116 页的『用于手工更新目录统计信息的一般规则』

第 6 章 了解 SQL 编译器

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

在分区数据库环境中，SQL 编译器对 SQL 查询执行的所有工作是在连接的数据库分区上进行的。在运行已编译的查询之前，将该查询发送至数据库中的所有数据库分区上。

本章中的主题提供有关 SQL 编译器如何编译和优化 SQL 语句的更多信息。

SQL 编译器进程

SQL 编译器执行几个步骤来产生可以执行的存取方案。这些步骤显示在下图中并在图下面的节中所描述。注意，某些步骤仅对联合数据库中的查询发生。

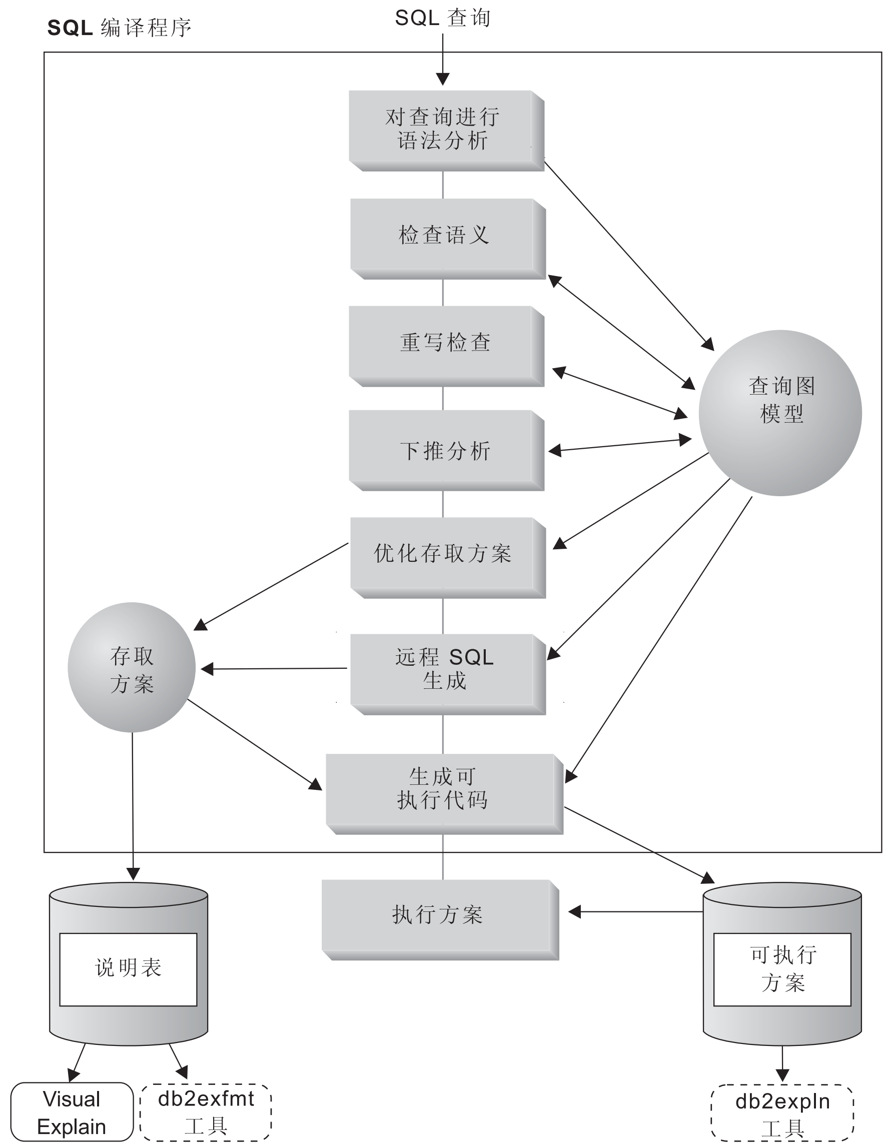


图 12. SQL 编译器执行的步骤

查询图模型

*查询图模型*是一个驻留在内存中的内部数据库，它代表如下所述步骤中处理的查询:

1. **对查询进行语法分析**

   SQL 编译器分析 SQL 查询以验证其语法。如果检测到任何语法错误，则 SQL 编译器停止处理，并将适当的 SQL 错误返回至递交该查询的应用程序。当语法分析完成时，创建该查询的内部表示并将它存储在查询图模型中。

2. **检查语义**

   编译器确保在语句的各个部分中没有不一致。作为语义检查的一个简单示例，编译器验证为 YEAR 标量函数指定的列的数据类型是日期时间数据类型。

编译器也将行为语义添加至该查询图模型，包括引用约束、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、相关、派生的表、表达式、数据类型、数据类型转换、代码页转换和分区键。

3. **重写查询**

   编译器使用存储在查询图模型中的全局语义来将该查询变换为更易于优化的一种形式并将结果存储在查询图模型中。

   例如，编译器可能会移动谓词，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*。在一个分区数据库环境中，下列查询操作的计算更为集中:
   - 聚集
   - 行的再分发
   - 相关子查询，它是包含对该子查询之外表列的引用的子查询。

   对于分区环境中的某些查询，解除相关可能作为重写查询的一部分而发生。

4. **下推分析（联合数据库）**

   此步骤中的主要任务是向优化器建议是否可在数据源处对一个操作以远程方式求值（即*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。

   除非执行联合数据库查询，否则绕过此步骤。

5. **优化存取方案**

   通过使用查询图模型作为输入，编译器的优化器部分生成许许多备用执行方案以满足查询。要估计每个备用方案的执行成本，优化器使用有关表、索引、列和函数的统计信息。然后，它选择具有最小估计执行成本的方案。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、基本表、派生的表、子查询、关联和递归。

   优化器还可考虑另一类下推操作，*聚集与排序*，这种操作可将对这些操作求得的值下推到“数据管理服务”组件中来提高性能。

   在确定页大小选择时，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）环境中实现查询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。

   编译器此步骤的输出是存取方案。此存取方案提供“说明”表中捕获的信息。可以用说明快照捕获用于生成该存取方案的信息。

6. **远程 SQL 生成（联合数据库）**

   优化器选择的最终方案可由一组对远程数据源执行操作的步骤组成。对于每个数据源执行的那些操作，远程 SQL 生成步骤将根据数据源 SQL 方言创建一个高效的 SQL 语句。

7. **生成“可执行”代码**

在最后一个步骤中，编译器使用存取方案和查询图模型来创建查询的可执行存取方案或节。此代码生成步骤使用查询图模型中的信息，以避免重复执行需要对一个查询只计算一次的表达式。可以进行此优化的示例包括代码页转换和主机变量的使用。

要对具有主机变量、专用寄存器或参数标记的静态和动态 SQL 语句启用查询（重新）优化，应将程序包与 REOPT 绑定选项进行绑定。如果使用的话，就将使用这些变量的值而不是由编译器选择的缺省估计值来优化 SQL 语句（它属于该程序包，并且包含主机变量、参数标记或专用寄存器）的存取路径。当值可用时，执行查询时就会进行此优化。

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，数据库管理器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。此信息由 *db2expln* 工具使用。

注: 在经常更改的表上以适当的时间间隔执行 RUNSTATS。优化器需要有关表及其数据的最新统计信息以创建最有效的存取方案。重新绑定应用程序以利用更新的统计信息。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，则优化器可能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。已占用块的总数存储在 ACTIVE_BLOCKS 列中。

相关概念:

- 第 127 页的『查询重写方法和示例』
- 第 135 页的『数据存取方法』
- 第 140 页的『谓词术语』
- 第 142 页的『连接』
- 第 156 页的『排序和分组的效果』
- 第 158 页的『分区内并行性的优化策略』
- 第 160 页的『具体查询表』
- 第 165 页的『分析在何处对联合查询求值的准则』
- 『Advantages of Deferred Binding』（*Application Development Guide: Programming Client Applications*）
- 第 160 页的『MDC 表的优化策略』

影响查询优化的配置参数

有一些配置参数影响 SQL 编译器选择的存取方案。其中的许多参数都适合单一分区数据库，而某些参数仅适合分区数据库。在分区数据库中，用于每个参数的值应该在所有分区上是相同的。

注: 当动态更改配置参数时，优化器可能会由于程序包高速缓存中的旧存取方案而不立即读取已更改的参数值。要复位程序包高速缓存，执行 FLUSH PACKAGE CACHE 命令。

在联合系统中，如果大多数查询都将存取昵称，则在更改环境之前评估您发送的查询类型。例如，在联合数据库中，缓冲池不高速缓存数据源中的页，这些数据源是 DBMS

和联合系统中的数据。因此，增大缓冲区的大小并不保证当优化器为包含昵称的查询选择存取方案时将考虑附加存取方案备用。但是，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。在这种情况下，增加“DB2® 通用数据库”可用的资源可能会提高性能。

下列配置参数或因子影响 SQL 编译器所选的存取方案:

- 当创建或改变缓冲池时指定的缓冲池大小

  当优化器选择存取方案时，优化器要考虑将页从磁盘访存至缓冲池的 I/O 成本并估计满足查询所需要的 I/O 次数。估计包括预测缓冲池使用情况，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。

  优化器考虑 BUFFERPOOLS 系统目录表以及在分区数据库中的 BUFFERPOOLDBPARTITION 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
  - 是否将使用非群集索引来读取数据
- 缺省度（dft_degree）

  `dft_degree` 配置参数通过为 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项提供缺省值来指定并行性。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。
- 缺省查询优化级别（dft_queryopt）

  虽然编译 SQL 查询时可以指定查询优化级别，但可以设置缺省优化度。

  注: 除非通过设置 *intra_parallel* 数据库配置参数来启用内部并行处理，否则不会发生内部并行处理。
- 活动应用程序平均数（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，在运行时可能有多少缓冲池可用。较高的此参数值会影响优化器，使它选择在缓冲池的使用情况方面更节省的存取方案。如果指定值 1，则优化器认为整个缓冲池将可用于应用程序。
- 排序堆大小（sortheap）

  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，可以将该表写入磁盘。当所有排序都完成时，将这些已排序的子集合并到单个排序的行集。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。

  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。
- 锁定列表的最大存储器（locklist）和升级前锁定列表的最大百分比（maxlocks）

当隔离级别是**可重复读（RR）**时，SQL 优化器考虑 *locklist* 和 *maxlocks* 参数的值，来确定是否可以将行级别锁定升级到表级别锁定。如果优化器估计将对表存取发生锁定升级，则它为存取方案选择一个表级别锁定，这样就不会在查询执行期间产生锁定升级的开销。

- 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。CPU 成本估计和各种 I/O 成本估计有助于选择查询的最佳存取方案。

  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。除非您要在测试系统上为生产环境建立模型或评估硬件更改的影响，否则不应调整此参数。使用此参数为不同的硬件环境建立模型可以使您找出可为该环境选择的存取方案。要让 DB2 重新计算此自动配置参数的值，将它设置为 -1。

- 语句堆大小（stmtheap）

  尽管语句堆的大小不影响优化器选择不同的存取路径，但是，它会影响对复杂的 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，指示无足够可用内存来处理语句。例如，SQLCODE +437（SQLSTATE 01602）可能指示用于编译语句的优化量小于您请求的量。

- 最大查询并行度（max_querydegree）

  当 *max_querydegree* 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，则用户指定的值确定应用程序的并行度。

- 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。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 135 页的『数据存取方法』
- 第 158 页的『分区内并行性的优化策略』
- 第 160 页的『MDC 表的优化策略』

相关参考:

- 第 414 页的『max_querydegree － 最大查询并行度』
- 第 419 页的『comm_bandwidth － 通信带宽』
- 第 324 页的『sortheap － 排序堆大小』
- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
- 第 326 页的『stmtheap － 语句堆大小』
- 第 420 页的『cpuspeed － CPU 速度』
- 第 345 页的『avg_appls － 活动应用程序的平均数目』
- 第 396 页的『dft_degree － 缺省并行度』

查询重写

本节描述了优化器可以重写查询以提高性能的方式。

查询重写方法和示例

在重写查询阶段期间，SQL 编译器将 SQL 语句变换为可以更易于优化的格式，并因此可以改善可能的存取路径。重写查询对于非常复杂的查询（包括具有许多子查询或许多连接的查询）特别重要。查询生成器工具通常会创建这些很复杂的查询。

要影响应用于 SQL 语句的查询重写规则的数目，更改优化级别。要查看查询重写的一些结果，使用“说明”设施或 Visual Explain。

可以下列三种主要方式中的任何一种重写查询：

- **操作合并**

  要构造查询以便它具有最少的操作数，尤其是 SELECT 操作，SQL 编译器重写查询来合并查询操作。以下示例举例说明可合并的一些操作：

  – 示例 － 视图合并

    使用视图的 SELECT 语句可限制表的连接顺序并可引入表的冗余连接。如果在查询重写期间合并视图，则可撤销这些限制。

  – 示例 － 子查询至连接的变换

    如果某个 SELECT 语句包含子查询，则可以限制表的顺序处理的选择。

  – 示例 － 消除冗余连接

    在查询重写期间，可除去冗余连接来简化 SELECT 语句。

  – 示例 － 共享聚集

    当查询使用不同的函数时，重写可减少需要执行的计算数。

- **操作移动**

  要构造具有最少数目的操作和谓词的查询，编译器重写查询以移动查询操作。以下示例举例说明可移动的一些操作：

  – 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。在提供的扩展示例中，全部除去了 DISTINCT 操作。

  – 示例 － 一般谓词下推

    在查询重写期间，优化器可更改应用谓词的顺序，以便将选择性更强的谓词尽早地应用于该查询。

  – 示例 － 解除相关

    在分区数据库环境中，在数据库分区之间移动结果集的成本很高。减小必须广播至其它数据库分区的信息的大小和／或减少广播次数，是查询重写的目标。

- **谓词转换**

SQL 编译器重写查询，以便把特定查询的现有谓词转换为更优的谓词。以下示例举例说明可转换的一些谓词:

– 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 – ON 至 IN 的变换

  在查询重写期间，可以将一个 OR 谓词转换为一个 IN 谓词，以获取更有效的存取方案。SQL 编译器也可将一个 IN 谓词转换为一个 OR 谓词，但前提是此变换将创建一种更有效的存取方案。

相关概念:

- 第 128 页的『编译器重写示例: 视图合并』
- 第 130 页的『编译器重写示例: DISTINCT 消除』
- 第 132 页的『编译器重写示例: 隐含谓词』
- 第 133 页的『多个谓词的列相关』

编译器重写示例: 视图合并

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

```
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，来列示既受过高等教育而且工资又超过 $35,000 的职员:

```
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本表，还可执行附加重写。

示例 – 子查询至连接的变换

SQL 编译器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含有不需要的连接。在查询重写阶段期间，也可能产生类似以下的查询。

```
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
```

在此查询中，SQL 编译器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 样本表之间存在引用约束。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，因为他们对基础表没有存取权。他们可能只对以上显示的视图具有存取权。因此，必须在数据库管理器内执行这种类型的优化。

只要符合以下条件，引用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 － 共享聚集

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 编译器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;
```

用下列方式变换该查询:

```
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。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 127 页的『查询重写方法和示例』

编译器重写示例: DISTINCT 消除

如果 EMPNO 列被定义为 EMPLOYEE 表的主键，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，因为选择了主键，因此 SQL 编译器知道返回的每一行已经是唯一的。在这种情况下，DISTINCT 关键字是多余的。如果未重写该查询，则优化器将需要通过必要的处理（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”中所有职员的列表:

```
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'
```

该编译器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。可以下推谓词的其它情况包括 UNION、GROUP BY 和派生的表（嵌套表表达式或公共表表达式）。

示例 – 解除相关

在一个分区数据库环境中，SQL 编译器可重写以下查询:

查找正在从事程序设计项目并被少付工资的所有职员。

```
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 编译器可以重写该查询，如下所示:

- 确定从事程序设计项目的职员的特异值列表，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集:

```
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职员的特异值列表，将该列表与职员表连接，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。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 127 页的『查询重写方法和示例』

编译器重写示例: 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，以下查询将列示其部门号大于“E00”的部门和在这些部门工作的职员的名称。

```
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，该重写阶段添加以下隐含的谓词:

```
EMP.WORKDEPT > 'E00'
```

作为此重写的结果，优化器会减少要连接的行数。

示例 – OR 至 IN 的变换

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 IN 谓词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，数据库管理器可以将一个 IN 谓词转换为一组 OR 子句，以便可执行索引 OR 运算。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 127 页的『查询重写方法和示例』

多个谓词的列相关

应用程序可以包含使用连接构造的查询，以使多个连接谓词连接两个表。当查询需要确定不同表中相似的相关列之间的关系时，这是很平常的。

例如，考虑某个制造商，它利用各种颜色、弹性和质量的原料来生产产品。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。它假定它们是独立的，即，对于每种颜色存在各种弹性，相反，对于每一级别的弹性，存在每种颜色的原料。然后它通过使用每个表的有关弹性级别数和不同颜色数的目录统计信息来估计谓词对的总体选择性。例如，根据此估计，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，且只有其它几种颜色（与弹性材料的颜色不同）的弹性很低的材料可用。于是，谓词的组合选择性排除较少的行，因此查询将返回更多行。考虑极端的情况，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。优化器可能不再选择最佳方案。例如，当合并连接将更快时，它可能选择嵌套循环连接方案。

对于其它数据库产品，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它查询产生不期望的副作用。

如果您定义那些列的索引，或者如果您收集并维护适当列的组列统计信息，则 DB2® UDB 优化器将尝试检测并补偿连接谓词的相关性:

例如，在以上弹性示例中，您可以定义涉及到:

```
PRODUCT.COLOR, PRODUCT.ELASTICITY
```

和／或

```
RAWMATERIAL.COLOR, RAWMATERIAL.ELASTICITY
```

的唯一索引。

对于检测相关的优化器，此索引的非包含列必须是唯一的相关列。索引还可以包括包含列以允许只扫描索引。如果在连接谓词中有两个以上的相关列，则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，一个表中的相关列是其主键。因为主键总是唯一的，所以不需要定义另一个唯一索引。

创建适当的索引之后，确保表的统计信息是最新的，且未因任何原因（如试图影响优化器）以手工方式改变它们的真实值。

优化器使用唯一索引统计信息表的 FIRSTnKEYCARD 和 FULLKEYCARD 列中的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。

作为另一种方法，可以收集一组列的列组统计信息。在以上的灵活性示例中，您可以收集 PRODUCT.COLOR、PRODUCT.ELASTICITY 和／或 RAWMATERIAL.COLOR、RAWMATERIAL.ELASTCITY 这些列的统计信息。

列组统计信息是使用 RUNSTATS 的 “ON COLUMNS” 选项收集的。例如，要收集 PRODUCT.COLOR 和 PRODUCT.ELASTICITY 的列组统计信息，可发出以下 RUNSTATS 命令:

```
RUNSTATS ON TABLE product ON COLUMNS ((color, elasticity))
```

简单相等谓词的相关

除了 JOIN 谓词相关之外，优化器还使用类型为 COL = *constant* 的简单相等谓词来管理相关。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、YEAR、COLOR 和 STYLE（如私家车、旅行车、赛车）。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。但是，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。同一个型号名由两个或两个以上的轿车制造商使用几乎是不可能的，当然轿车制造商也不想这样做。

如果 MAKE 和 MODEL 这两列上存在索引或者收集了列组统计信息，则优化器使用关于该索引或这两列的统计信息来确定组合的特异值数，并调整两列之间的相关的选择性或基数估计。如果这种谓词不是连接谓词，则优化器不需要唯一索引来进行调整。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 127 页的『查询重写方法和示例』

使用 REOPT 绑定选项进行查询优化

要对具有主机变量、专用寄存器或参数标记的静态和动态 SQL 语句启用查询优化（或重新优化），应将程序包与 REOPT 绑定选项进行绑定。当使用此选项时，将使用主机变量、参数标记或专用寄存器的值而不是编译器选择的缺省估计值来优化属于该程序包并且包含这些变量的 SQL 语句的存取路径。当值可用时，执行查询时就会进行此优化。

相关概念:

- 『Effects of REOPT on static SQL』（*Application Development Guide: Programming Client Applications*）
- 『Effects of REOPT on dynamic SQL』（*Application Development Guide: Programming Client Applications*）

数据存取方法

本节描述优化器可以选择以存取查询所需要的数据的方法。

数据存取方法

SQL 优化器在编译一条 SQL 语句时估计满足查询的不同方法的执行成本。根据它的估计，优化器选择优化存取方案。*存取方案*指定指定解析一条 SQL 语句所需的操作顺序。当绑定一个应用程序时，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。

有两种方法来存取表中的数据:

- 顺序扫描整个表
- 通过首先存取表上的索引来定位特定表行

要产生查询请求的结果，根据谓词的条款选择行，通常在 WHERE 子句中说明这些条款。连接存取表中选择的行来产生结果集，并可以通过分组或排序输出来进一步处理结果集。

相关概念:

- 第 121 页的『SQL 编译器进程』
- 第 135 页的『通过索引扫描的数据存取』
- 第 138 页的『索引存取的类型』
- 第 140 页的『索引存取和群集比率』

通过索引扫描的数据存取

在数据库管理器因下列任何一个原因存取一个索引时，执行*索引扫描*:

- 在存取基本表前，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。
- 排序输出。
- 直接检索请求的列数据。如果所有请求的数据位于该索引中，则不需要存取索引表。这就称为*纯索引存取*。

如果使用 ALLOW REVERSE SCANS 选项创建索引，则也可以按与定义的方向相反的方向来执行扫描。

注: 如果未创建适当的索引或如果索引扫描的成本将更高，则优化器选择表索引。当表不大、索引群集比率低或查询请求大多数表行时，索引扫描的成本可能更高。要查明存取方案使用表扫描还是索引扫描，使用“SQL 说明”设施。

定界范围的索引扫描

要确定索引是否可以用于特定查询，优化器从索引第一列开始对每列求值，以了解它是否可用来满足等式和 WHERE 子句中的其它谓词。*谓词*是 WHERE 子句中搜索条件的元素，它表示或隐含比较运算。在下列情况下，可使用谓词来定界索引扫描范围的:

- 测试是否等于一个常数、主机变量、值为常数的表达式或关键字

- 测试“为空”还是“不为空”
- 测试是否等于基本子查询（它是不包含 ANY、ALL 或 SOME 的子查询），且该子查询没有相关列引用其直接父查询块（即，此子查询是其子选择的 SELECT）。
- 测试严格和相容不等式。

以下示例举例说明索引何时可以用来限制某个范围:

- 考虑具有下列定义的索引:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

  在此情况下，下列谓词可以用来限制索引 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 WHERE 子句中，谓词不必按键列在索引中出现的相同顺序来指定。尽管示例使用了主机变量，但其它变量（例如，参数标记、表达式或常数）将会有相同的效果。

- 考虑使用 ALLOW REVERSE SCANS 参数创建的单个索引。这类索引支持以创建索引时定义的方向扫描以及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
```

这是因为有一个键列（MGR）将这些列与最前面两个索引键列分开，因此排序将关闭。但是，一旦该范围由 NAME = :hv1 和 DEPT = :hv2 谓词确定，则可根据其余索引键列来对其余谓词求值。

测试不等式的索引扫描

某些不等式谓词可以对索引扫描的范围定界。有两种类型的不等式谓词:

- 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大于（>）和小于（<）。

只考虑一个具有严格不等式谓词的列用于定界索引扫描的范围。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`。数据库管理器将使用该索引扫描部门 404 和 405 的全部，但当它扫描到职员号（MGR 列）大于 12345 的第一个经理时它将停止扫描部门 406。

排序数据索引扫描

如果查询要求按排序顺序输出，而且，如果排序列从第一个索引键列开始连续出现在该索引中，则可使用索引来对数据排序。确定顺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键列是否等于“常数值”时，该常数值是值为常数的任何表达式。在这种情况下，排序列不能是第一个索引键列。

考虑以下查询:

```
WHERE NAME = 'JONES'
  AND DEPT = 'D93'
ORDER BY MGR
```

对于此查询，可使用索引来对行排序，因为 NAME 和 DEPT 将始终是相同的值，因此将被排序。也就是说，前导 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排序顺序需求。考虑下列索引定义和 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 列再进行排序，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。

相关概念:

- 第 135 页的『数据存取方法』
- 第 22 页的『索引结构』
- 第 138 页的『索引存取的类型』
- 第 140 页的『索引存取和群集比率』

索引存取的类型

在某些情况下，优化器可能发现可以从表上的索引检索来自表中查询需要的所有数据。在其它情况下，优化器可以使用多个索引来存取表。对于范围群集表，可以通过“虚拟”索引来存取数据，“虚拟”索引将计算数据记录的位置。

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本表，就可满足以下查询:

```
SELECT NAME, DEPT, MGR, SALARY
  FROM EMPLOYEE
  WHERE NAME = 'SMITH'
```

但是，经常需要未出现在该索引中的列。要获取这些列的数据，必须读取表行。要允许优化器选择纯索引存取，创建具有包含列的唯一索引。例如，考虑下列索引定义:

```
CREATE UNIQUE INDEX IX1 ON EMPLOYEE
   (NAME ASC)
    INCLUDE (DEPT, MGR, SALARY, YEARS)
```

此索引实现 NAME 列的唯一性，并且也存储和维护 DEPT、MGR、SALARY 和 YEARS 列的数据，它通过仅存取索引来使得满足下列查询:

```
SELECT NAME, DEPT, MGR, SALARY
  FROM EMPLOYEE
   WHERE NAME='SMITH'
```

但是，当考虑将 INCLUDE 列添加至索引时，考虑附加存储空间和维护成本是否合理。如果很少执行通过仅读取这种索引就可以满足的查询，则成本可能不合理。

多个索引存取

优化器可以选择扫描同一表上的多个索引来满足 WHERE 子句的谓词。例如，考虑下列两个索引定义:

```
INDEX IX2:  DEPT    ASC
INDEX IX3:  JOB     ASC,
            YEARS   ASC
```

下列谓词可以通过使用两个索引来满足:

```
WHERE DEPT = :hv1
     OR (JOB   = :hv2
     AND YEARS >= :hv3)
```

扫描索引 IX2 产生满足 DEPT = :hv1 谓词的行标识（RID）的列表。扫描索引 IX3 产生满足 JOB = :hv2 AND YEARS >= :hv3 谓词的 RID 的列表。在存取该表之前，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 IN 子句中指定的谓词，如下示例中所示:

```
WHERE DEPT IN (:hv1, :hv2, :hv3)
```

尽管索引 OR 运算的目的是消去重复的 RID，但是，*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创建多个索引，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 运算"。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。

要在扫描多个索引时实现动态位图的性能效益，可能需要更改排序堆大小（*sortheap*）数据库配置参数和排序堆阈值（*sheapthres*）数据库管理器配置参数的值。

当在存取方案中使用动态位图时，需要附加的排序堆空间。当将 *sheapthres* 设置为相对接近 *sortheap* 时（也就是，每个并发查询占用的空间减少两三倍），则具有多个索引存取的动态位图必须以比优化器所预料的少得多的内存来运行。解决方案是增大相对于 *sortheap* 的 *sheapthres* 的值。

注: 优化器在存取单个表时不将索引 AND 运算和索引 OR 运算组合在一起。

范围群集表中的索引存取

与标准表不同，范围群集表不需要象传统的 B 树索引那样将键值映射至某行的物理索引。但是，它利用列域的顺序特性并使用运作映射来在表中生成给定行的位置。在此映射的最简单的示例中，范围内的第一个键值是表中的第一行，范围内的第二个值是表中的第二行，依此类推。

优化器使用表的范围群集属性来根据最佳群集索引（它的唯一成本就是计算范围群集函数）来生成存取方案。保证了表中的行的群集，这是因为范围群集表保留了它们的原始键值排序。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 135 页的『数据存取方法』
- 第 135 页的『通过索引扫描的数据存取』
- 第 140 页的『索引存取和群集比率』

索引存取和群集比率

当它选择存取方案时，优化器估计将所需要的页从磁盘访存至缓冲池所需要的 I/O 次数。此估计包括预测缓冲池的使用情况，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，系统目录表（SYSCAT.INDEXES）中的信息帮助优化器估计将数据页读入缓冲池的 I/O 成本。它使用 SYSCAT.INDEXES 表中以下列中的信息:

- CLUSTERRATIO 信息指示表数据关于此索引的群集度。该数越高，按索引键顺序排序的行越好。如果表行接近索引键顺序，则在数据页位于缓冲区时可以从该页中读取行。如果此列的值为 -1，则优化器使用 PAGE_FETCH_PAIRS 和 CLUSTERFACTOR 信息（如果它可用的话）。
- PAGE_FETCH_PAIRS 包含几对数字，它们与 CLUSTERFACTOR 信息一起模拟将数据页读入各种大小的缓冲池时所需的 I/O 次数。仅当您使用 DETAILED 子句对索引执行 RUNSTATS 时，才对这些列收集数据。

如果索引群集统计信息不可用，则优化器使用缺省值，在这种情况下假定数据与该索引的群集关系不强。

数据相对于索引的群集度可大大影响性能，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：其键是群集索引键的超集的那些索引，或者在两个索引的键列之间存在实际相关的那些索引。

当您重组表时，您可指定一个索引，它将用于群集这些行并试图在插入处理期间保留此特征。因为更新和插入可能降低表相对于索引的群集率，所以您可能需要定期重组该表。要降低对由于 INSERT、UPDATE 和 DELETES 而频繁更改的表重组的频率，当改变表时使用 PCTFREE 参数。这允许把附加的插入与现有的数据群集在一起。

相关概念:

- 第 225 页的『索引性能技巧』
- 第 138 页的『索引存取的类型』

谓词术语

用户应用程序使用 SQL 语句从数据库请求一组行，该 SQL 语句指定返回为结果集的特定行的限定符。这些限定符通常出现在查询的 WHERE 子句。这种限定符称为*谓词*。可以将谓词分组为四种类别，按如何在求值过程中使用该谓词以及何时使用它来确定这些类别。这些类别列示如下，按最佳性能到最差性能的顺序排列:

1. 范围定界谓词
2. 索引控制谓词

3. 数据控制谓词
4. 残留谓词。

注: *控制*是指可用作*搜索自变量*的术语。

下表总结谓词类别。后续各节更详细地描述每个类别。

表 27. 谓词类型特征的总结

特征	谓词类型			
	范围定界	索引控制	数据控制	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

范围定界和索引控制谓词

范围定界谓词限制索引扫描的范围。它们提供索引搜索的开始和停止键值。索引控制谓词不能限制搜索范围，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键的一部分。例如，考虑下列索引:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

也考虑包含下列 WHERE 子句的查询:

```
WHERE  NAME  = :hv1
 AND   DEPT  = :hv2
 AND   YEARS > :hv5
```

前两个谓词（NAME = :hv1 和 DEPT = :hv2）是范围定界谓词，而 YEARS > :hv5 是索引控制谓词。

优化器在对这些谓词求值时使用索引数据而不是读取基本表。这些*索引控制*谓词减小需要从表中读取的行集，但它们不影响存取的索引页数。

数据控制谓词

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*数据控制*谓词。这些谓词通常需要存取表中的单独行。如果需要，“数据管理服务”检索对该谓词求值需要的列以及任何其它列以满足 SELECT 列表中不能从该索引获取的列。

例如，考虑在 PROJECT 表上定义的单个索引:

```
INDEX IX0:  PROJNO ASC
```

对于下列查询，然后，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

残留谓词比存取表需要更多的 I/O 成本。它们可具有下列特征:

- 使用相关子查询
- 使用量化子查询，这些子查询包含 ANY、ALL、SOME 或 IN 子句
- 读取 LONG VARCHAR 或 LOB 数据，该数据存储在与表分开的文件中

这种谓词由“关系数据服务”来求值。

有时，当存取数据页时，必须重新应用仅应用于索引的谓词。例如，当存取数据页时，使用索引 OR 运算或索引 AND 运算的存取方案始终要将这些谓词作为残留谓词重新应用。

相关概念:

- 第 121 页的『SQL 编译器进程』

连接方法和策略

本节描述优化器如何按需要连接表以将结果返回到查询并说明优化器使用的方法和策略。

连接

*连接*是根据信息的某些公共域从两个或多个表组合信息的过程。当对应行中的信息与连接准则匹配时，一个表中的行与另一个表中的行配对。

例如，考虑下列两个表:

Table1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

Table2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

要将标识列具有相同值的 Table1 和 Table2 连接，使用下列 SQL 语句:

```
SELECT PROJ, x.PROJ_ID, NAME
  FROM TABLE1 x, TABLE2 y
  WHERE x.PROJ_ID = y.PROJ_ID
```

此查询得到下列一组结果行:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike

PROJ	PROJ_ID	NAME
C	3	Joe
D	4	Mary

根据连接谓词是否存在，以及所涉及到的、由表和索引统计信息来确定的各种成本，优化器选择下列其中一种连接方法:

- 嵌套循环连接
- 合并连接
- 散列连接

当连接两个表时，将一个表选择为外部表，而将另一个表选为内部表。首先存取外部表，并只扫描它一次。是否扫描内部表多次取决于连接的类型和存在的索引。即使一个查询连接两个以上表，优化器一次也只连接两个表。如果有必要，创建临时表来保存中间结果。

可以提供显式连接运算符（如 INNER 或 LEFT OUTER JOIN）来确定如何在连接中使用表。然而，以这种方式改变查询之前，应允许优化器确定如何连接表。然后分析查询性能来决定是否要添加连接操作程序。

相关概念:

- 第 143 页的『连接方法』
- 第 149 页的『分区数据库中的连接策略』
- 第 150 页的『分区数据库中的连接方法』
- 第 525 页的『连接信息』

连接方法

当查询需要连接表时，优化器可以选择三种基本连接策略的其中一种。

- 嵌套循环连接
- 合并连接
- 散列连接

这些方法在以下节中描述。

嵌套循环连接

嵌套循环连接是用下面两种方式的其中一种方式执行的:

- 对于外部表中每个被存取的行，扫描内部表

  例如，考虑表 T1 和 T2 中的列 A 具有下列值:

外部表 **T1**: 列 **A**	内部表 **T2**: 列 **A**
2	3
3	2
3	2
	3
	1

要执行嵌套循环连接，数据库管理器执行下列步骤:

1. 从 T1 中读取第一行。A 的值是“2”
2. 扫描 T2，直到发现一个匹配项（“2”），然后连接这两行
3. 扫描 T2，直到发现下一个匹配项（“2”），然后连接这两行
4. 扫描 T2，直至该表结尾
5. 返回至 T1，并读取下一行（“3”）
6. 从第一行开始，扫描 T2，直到发现匹配项（“3”），然后连接这两行
7. 扫描 T2，直到发现下一个匹配项（“3”），然后连接这两行
8. 扫描 T2，直至该表结尾
9. 返回至 T1，并读取下一行（“3”）
10. 象以前一样扫描 T2，连接匹配（“3”）的所有行。

- 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。考虑下列示例:

  ```
  OUTER.C1 + OUTER.C2 <= INNER.C1
  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。

当它对嵌套循环连接求值时，优化器在执行连接前也决定是否对外部表排序。如果它根据连接列对外部表排序，则可以减少从磁盘存取内部表的页的读操作次数，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将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*）需要如下格式的一个谓词 table1.column = table2.column。这称为*等式连接谓词*。合并连接需要连接列的已排序输入，这个输入可通过索引存取或通过排序来获得。如果连接列是 LONG 字段列或大对象（LOB）列，则不能使用合并连接。

在合并连接中，同时扫描已连接的表。合并连接的外部表只扫描一次。除非外部表中出现重复的值，否则，内部表也只扫描一次。如果有重复的值出现，则可能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

外部表 **T1**: 列 **A**	内部表 **T2**: 列 **A**
	3
	3

要执行合并连接，数据库管理器执行下列步骤:

1. 从 T1 中读取第一行。A 的值是“2”。
2. 扫描 T2，直到发现匹配项，然后连接这两行。
3. 连续扫描 T2，当列匹配时，连接那些行。
4. 当读取 T2 中的“3”时，返回至 T1 并读取下一行。
5. T1 中的下一个值是“3”，它与 T2 匹配，因此连接那些行。
6. 连续扫描 T2，当列匹配时，连接那些行。
7. 到达 T2 结尾。
8. 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。数据库管理器记住此位置。

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，列类型相同。对于类型为 CHAR 的列，长度必须相同。对于类型为 DECIMAL 的列，精度和小数位必须相同。列类型不能是 LONG 字段列或大对象（LOB）列。

首先，扫描指定的 INNER 表，然后将行复制到从排序堆划出的内存缓冲区中，该排序堆由 *sortheap* 数据库配置参数指定。根据在连接谓词的列上计算的散列值，将内存缓冲区分为若干分区。如果 INNER 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。

当已处理内部表时，通过首先比较对连接谓词的列计算的散列值，扫描第二个（即 OUTER）表并将该表的行与 INNER 表的行匹配。如果 OUTER 行列的散列值与 INNER 行列的散列值匹配，则比较实际连接谓词列值。

紧接着将对应于未写入临时表的分区的 OUTER 表行与内存中的 INNER 表行匹配。如果对应的 INNER 表分区已写入临时表，则也将 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值，检查连接谓词。

为了获取散列连接的所有性能效益，可能需要更改 *sortheap* 数据库配置参数和 *sheapthres* 数据库管理器配置参数的值。

为了获取散列连接的所有性能效益，可能需要更改 *sortheap* 数据库配置参数和 *sheapthres* 数据库管理器配置参数的值。

如果可以避免散列循环和溢出到磁盘，散列连接性能最佳。要调整散列连接性能，估计对 *sheapthres* 可用的最大内存量，然后调整 **sortheap** 参数。增大它的设置，直到尽可能避免散列循环和磁盘溢出，但不要达到由 *sheapthres* 参数指定的限制。

增大 *sortheap* 值也应提高具有多个类别的查询的性能。

相关概念:

- 第 142 页的『连接』

- 第 149 页的『分区数据库中的连接策略』
- 第 150 页的『分区数据库中的连接方法』
- 第 525 页的『连接信息』

相关参考:

- 第 324 页的『sortheap － 排序堆大小』
- 第 323 页的『sheapthres － 排序堆阈值』

选择最佳连接的策略

优化器使用各种方法来选择查询的最佳连接策略。这些方法包括下列搜索策略，它们由查询的优化级别来确定:

- 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，在进一步优化期间就不更改它
  - 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
  - 随着已连接表数的增加，空间和时间需求按指数规律增加
  - 可有效地、全面地搜索以获得最佳存取方案
  - 类似于 DB2® OS/390 或 z/OS 版使用的策略。

连接枚举算法是对优化器所使用的方案组合数的一个重要决定因子。

星型模式连接

在一个查询中引用的表几乎总是由连接谓词联系起来。如果两个表已连接而没有连接谓词，则形成了两个表的笛卡儿乘积。在笛卡儿乘积中，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。

当将优化器级别设置为 9 时或在星型模式的特殊情况下，发生仅有的例外情况。*星型模式*包含一个称为事实表的中央表，而其它表称为维表。全部维表仅有一个单一连接，该连接将这些维表与事实表连接，而不管查询是什么。每个维表包含展开有关事实表中特定列的信息的附加值。典型的查询由多个引用维表中的值的本地谓词组成，并包含将维表与事实表连接的连接谓词。对于这些查询，在存取大的事实表之前，对多个小维表计算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 可以识别对使用星型模式设计的、含至少两个维表的数据库的查询，并可以增加搜索空间来包括计算维表的笛卡儿乘积的可能方案。如果计算笛卡儿乘积的方案具有最低的估计成本，则优化器选择它。

以上讨论的星型模式连接策略假设连接中使用了主键索引。另一个方案涉及到外键索引。如果事实表中的外键列是单列索引，且在所有维表中有相当高的选择性，可使用以下星型连接技术:

1. 通过下列操作处理每个维表:
   - 在维表和事实表上的外键索引之间执行半连接
   - 散列行标识（RID）值，以动态创建一个位图。
2. 对每个位图的先前位图使用 AND 谓词。
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. 访存一个基本表行。
6. 将事实表与其每个维表重新连接，存取 SELECT 子句所需要的维表中的列。
7. 重新应用残留谓词。

此技术不需要多列索引。不要求事实表与维表之间存在显式引用完整性约束，尽管事实表与维表之间的关系应实际上按此方式联系。

星型连接技术创建和使用的动态位图需要排序堆内存，该内存的大小由“排序堆大小”（*sortheap*）数据库配置参数指定。

组合表

当一对表的连接结果是一个称为*组合*表的新表时，此表通常成为具有另一个内部表的另一个连接的外部表。这称为“组合外连接”。在某些情况下，特别是在使用贪婪连接枚举技术时，使连接两个表的结果成为后续连接的内部表将很有用。当一个连接的内部表由连接两个或更多个表的结果组成时，此方案是一个“组合内连接”。例如，考虑如下查询:

```
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表，选择第二个连接结果作为内部表。在最后的方案（（T1xT2）x（T3xT4））中，连接结果（T3xT4）称为组合内连接。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 允许组合内连接。

相关概念:

- 第 142 页的『连接』
- 第 143 页的『连接方法』
- 第 149 页的『分区数据库中的连接策略』
- 第 150 页的『分区数据库中的连接方法』

分区数据库中的重复具体查询表

重复具体查询表通过允许数据库管理表数据的预先计算的值来提高分区数据库环境中频繁执行连接的性能。

考虑一个查询和重复具体表示例。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 表在单一分区数据库分区组中。

根据 EMPLOYEE 表中的信息创建复制的具体查询表。

```
CREATE TABLE R_EMPLOYEE
  AS (
      SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT
   FROM EMPLOYEE
     )
DATA INITIALLY DEFERRED REFRESH IMMEDIATE
IN REGIONTABLESPACE
REPLICATED;
```

要更新重复的具体查询表的内容，运行下列语句:

```
REFRESH TABLE R_EMPLOYEE;
```

注: 使用 REFRESH 语句之后，您应该在复制表上运行 RUNSTATS，正如您在任何其它表上要运行的一样。

以下示例计算职员的销售额、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;
```

如果不使用仅在一个数据库分区上的 EMPLOYEE 表，则数据库管理器使用 R_EMPLOYEE 表，该表在存储 SALES 表的每个数据库分区上均有复制。这样性能将会增强，因为职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。

并置连接中的重复具体查询表

重复的具体查询表也可以有助于并置连接。例如，如果一个星型模式包含一个在 20 个节点上分布的大的事实表，则在事实表和维表之间的连接是最有效的（如果并置了这些表）。如果所有表位于同一个数据库分区组中，则对于一个并置连接，最多正确分区一个维表。其它的维表不能够在并置连接中使用，因为在事实表上的连接列与事实表的分区键不对应。

考虑在 C1 上分区称为 FACT (C1, C2, C3, ...) 的表；在 C1 上分区称为 DIM1 (C1, dim1a, dim1b, ...) 的表；在 C2 上分区称为 DIM2 (C2, dim2a, dim2b, ...) 的表；等等。

在此情况下，您会发现在 FACT 和 DIM1 之间的连接是最好的，因为并置了谓词 DIM1.C1 = FACT.C1。在列 CI 上分区这两个表。

但是，不能并置 DIM2 和谓词 WHERE DIM2.C2 = FACT.C2 之间的连接，因为在列 C1 上分区 FACT，而不在列 C2 上分区 FACT。在此情况下，可以在事实表的数据库分区组中复制 DIM2，以便连接按本地方式在每个分区上发生。

注: 此处复制的具体查询表讨论与分区内数据库复制有关。交互数据库复制与预订、控制表以及位于不同数据库中和不同操作系统上的数据有关。

当创建复制的具体查询表时，源表可以是数据库分区组中的单节点表或多节点表。在大多数情况下，复制的表不大，并且可以放置在单节点数据库分区组中。可以通过指定只有此表中列的子集，或通过指定使用的谓词行数，或通过使用两种方法来限制要复制的数据。为了复制具体查询表有效，并不需要数据捕获选项。

也可以在多节点数据库分区组中创建复制的具体查询表，以便在所有分区上创建源表的副本。如果向所有分区广播此源表，在大事实表和维表之间的连接在此环境中更可能在本地执行。

未自动创建复制表上的索引。可以创建与源表上的索引不同的索引。但是，为防止未在源表上出现的约束违规，不能创建唯一索引或对复制的表使用约束。即使在源表上发生相同的约束，也禁止这些约束。

可以在查询中直接引用复制的表，但您不可以将 NODENUMBER() 谓词与复制表一起使用，来查看特定分区上的表数据。

使用 EXPLAIN 设施来查看查询的存取方案是否使用了复制的具体查询表。优化器所选择的存取方案是否使用复制的具体查询表取决于连接所需要的信息。如果优化器确定向数据库分区组中的其它分区广播原始源表比较便宜，则优化器可能不使用复制的具体查询表。

相关概念:

- 第 142 页的『连接』

分区数据库中的连接策略

在某些方面，连接策略在分区数据库中与非分区数据库中不同。可以将附加技术应用于标准连接方法以提高性能。

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，根据相同的分区键，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。一旦被并置，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。

表队列

分区数据库中连接技术的描述使用下列术语:

- 表队列

  在数据库分区之间或单一分区数据库中的处理器之间传送行的一种机制。
- 定向表队列

  一种表队列，其中，将行散列至其中一个接收数据库分区。
- 广播表队列

  一种表队列，其中，将行发送至所有接收数据库分区，但不散列它们。

表队列用于下列情况下:

- 在使用分区间并行性时，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
- 在使用单一分区数据库时，传送一个数据库分区中的表数据。

每个表队列按单方向传送数据。编译器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，则数据库管理器将使用同步表队列。
- *合并表队列*。

  这些表队列维持顺序。
- *非合并表队列*。

  这些表队列也称为“正规”表队列。它们不保持顺序。
- *侦听器表队列*。

  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。

相关概念:

- 第 142 页的『连接』
- 第 143 页的『连接方法』
- 第 150 页的『分区数据库中的连接方法』

分区数据库中的连接方法

下列图形举例说明了分区数据库中的连接方法。

注: 在这些图表中，q1、q2、和 q3 表示示例中的表队列。引用的表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。

并置连接

并置连接以本地方式在数据所驻留的分区上发生。该分区在完成连接以后将数据发送到其它分区。要使优化器考虑并置连接，必须并置连接的表，且所有相应的分区键对必须参与等式连接谓词。

以下图形提供了一个示例。

注: 复制的具体查询表提高了并置连接的可能性。

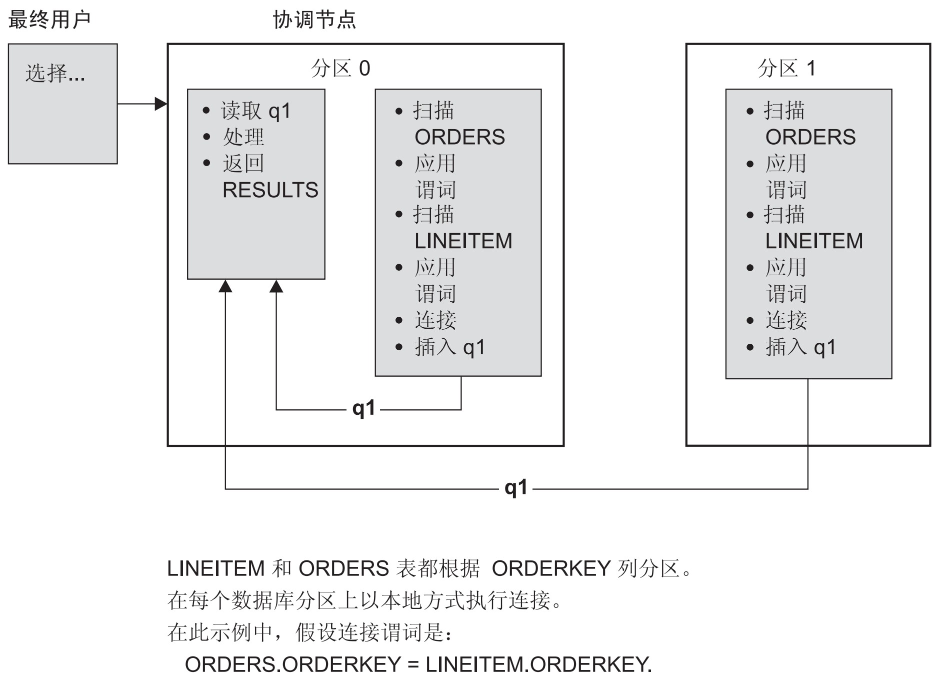


图 13. 并置连接示例

广播外部表连接

广播外部表连接是一个并行连接策略，如果在连接的表之间没有等式连接谓词，可使用它。在可以将它用作成本最低的连接方法的其它情况中，也可使用它。例如，当有一个很大的表和一个很小的表且没有在连接谓词列上对任何一个表进行分区时，可能会发生广播外部表连接。代替将两个表进行分区，将更小的表广播至更大的表可能成本更低。以下图形提供了一个示例。

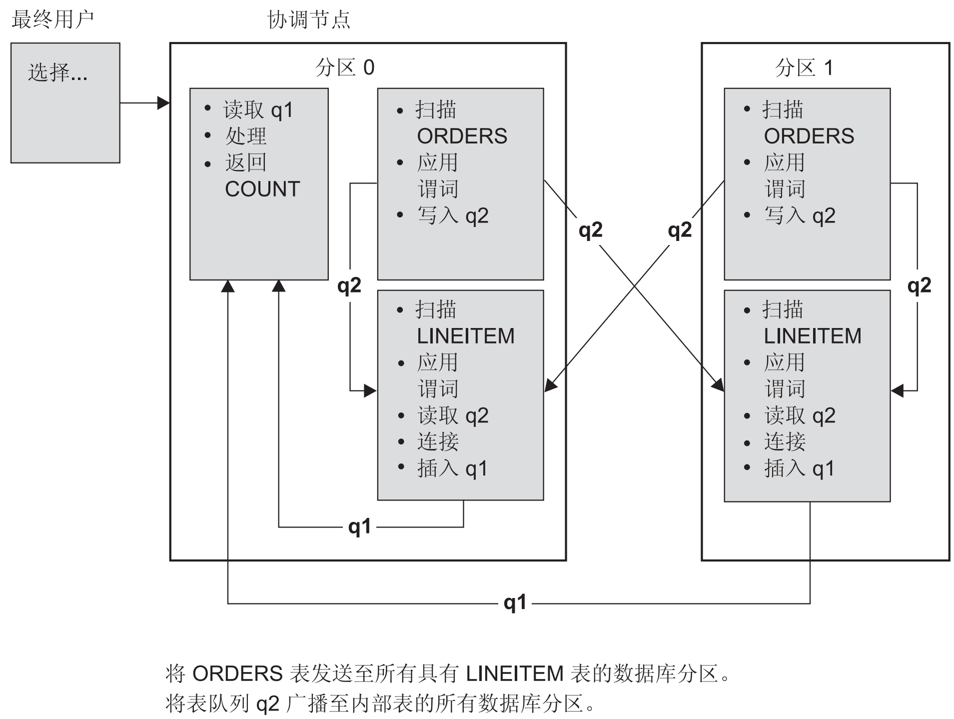
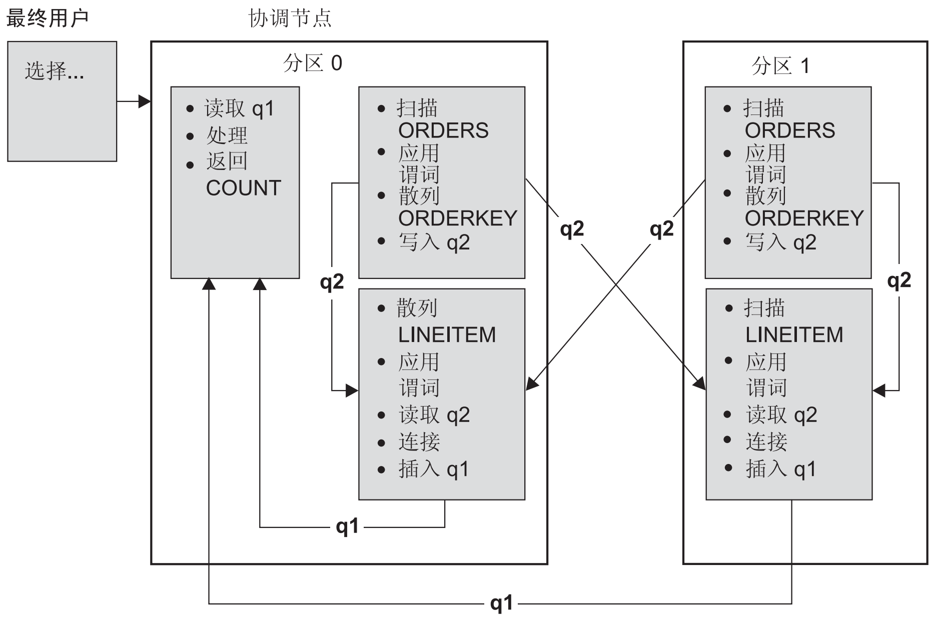


图 14. 广播外部表连接示例

定向外部表连接

在定向外部表连接策略中，根据内部表的分区属性将外部表的每一行发送至内部表的一个分区。该连接在此数据库分区上进行。以下图形提供了一个示例。



LINEITEM 表根据 ORDERKEY 列分区。
ORDERS 表根据另一个列分区。
将 ORDERS 表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 15. 定向外部表连接示例

定向内部表和外部表连接

在定向内部表和外部表策略中，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。以下图形提供了一个示例。以下图形中显示了一个示例。

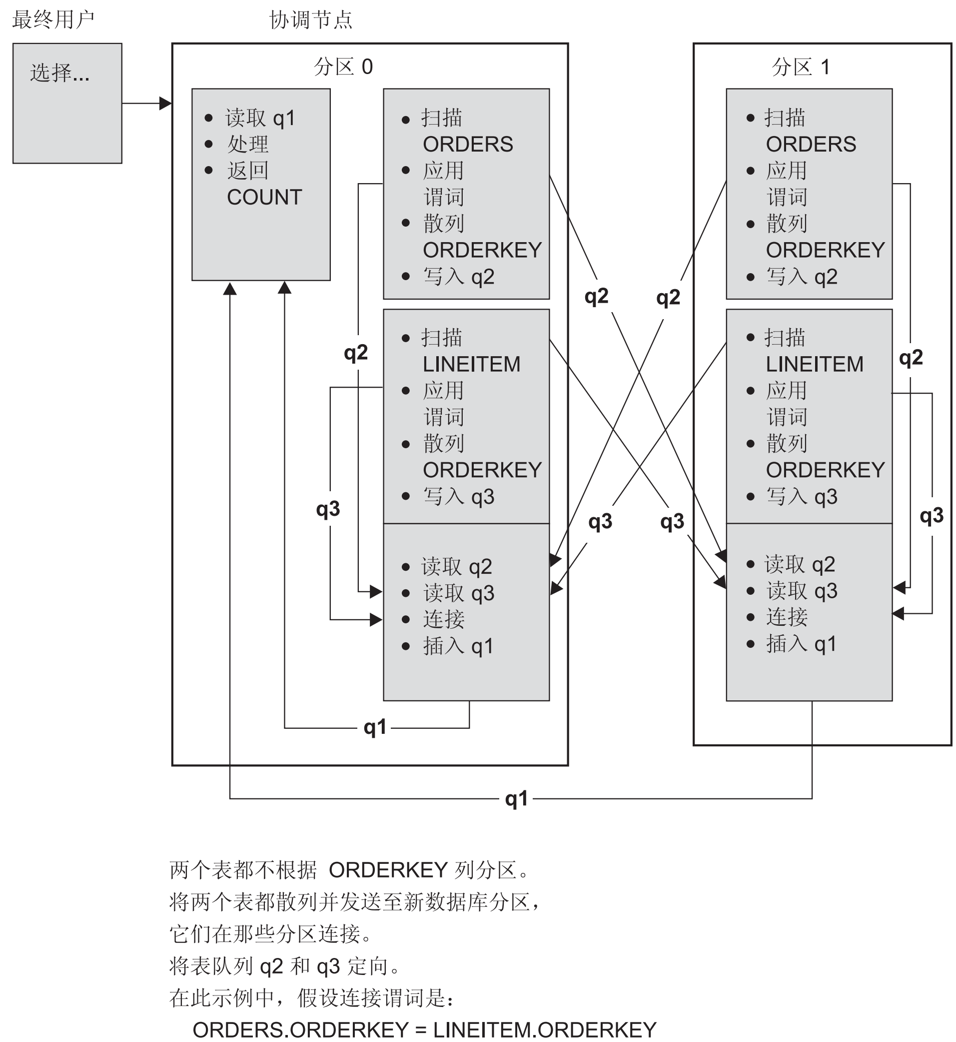
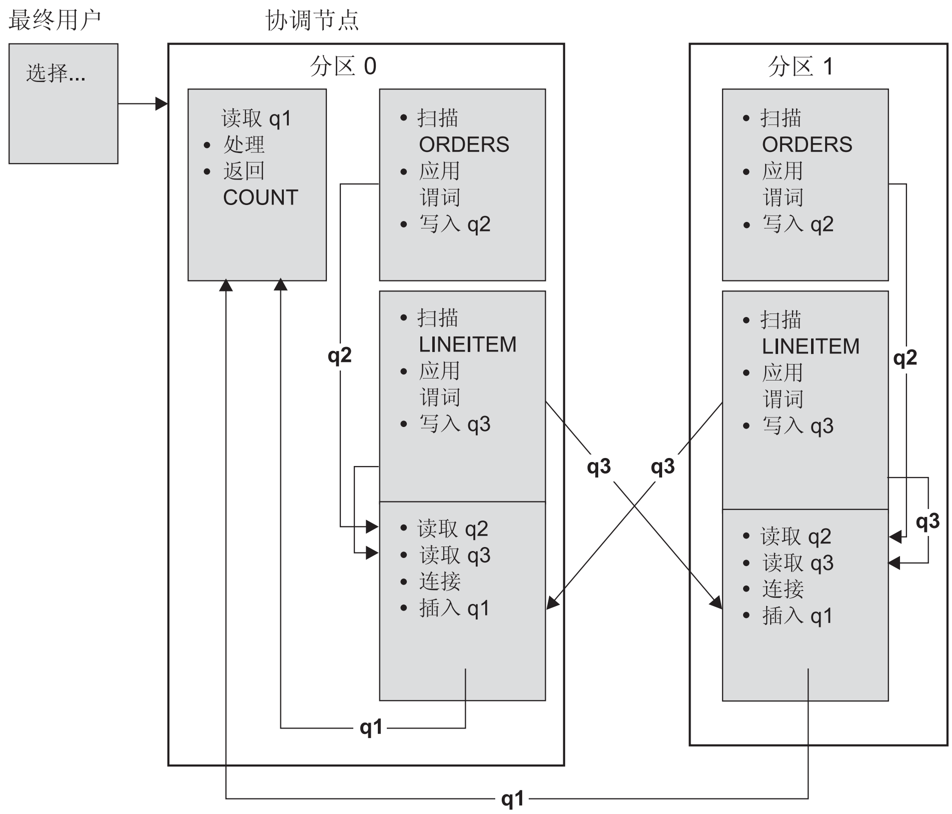


图 16. 定向内部表和外部表连接示例

广播内部表连接

在广播内表部连接策略中，将内部表广播至外连接表的所有数据库分区。以下图形提供了一个示例。

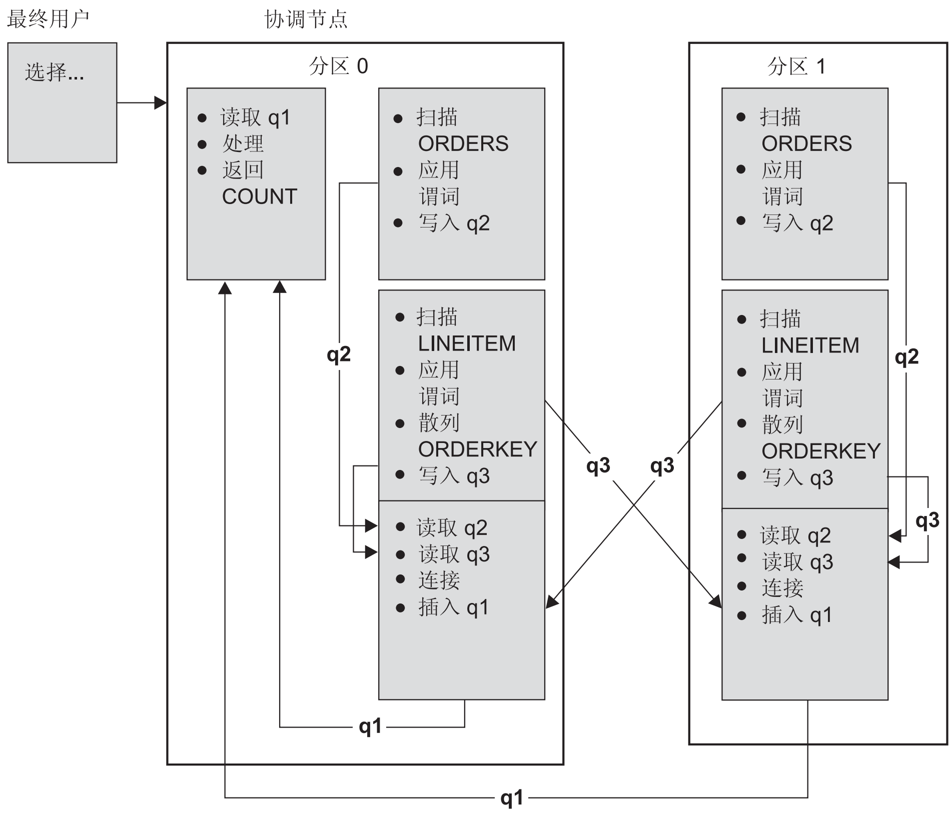


将 LINEITEM 表发送至所有具有 ORDERS 表的数据库分区。
将表队列 q3 广播至外部表的所有数据库分区。

图 17. 广播内部表连接示例

定向内部表连接

使用定向内部表连接策略，根据外部表的分区属性，将内部表的每一行发送至外连接表的一个数据库分区。该连接在此数据库分区上进行。以下图形提供了一个示例。



ORDERS 表根据 ORDERKEY 列分区。
LINEITEM 表根据另一个列分区。
将 LINEITEM 表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 18. 定向内部表连接示例

相关概念:

- 第 142 页的『连接』
- 第 143 页的『连接方法』
- 第 149 页的『分区数据库中的连接策略』

排序和分组的效果

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。当没有索引满足请求的对已访存的行的排序时，执行排序。当优化器确定排序比索引扫描开销较少时，也会进行排序。优化器采用下列其中一种方式来对数据进行排序:

- 当执行查询时，通过管道传送排序结果。
- 在数据库管理器内对排序进行内部处理。

管道传送排序与非管道传送排序

如果可按单一顺序读取数据的最终排序列表，则可以用*管道传送*这些结果。管道传送是一种比非管道传送更快的传送排序结果的方式。优化器会尽可能地选择用管道传送排序的结果。

无论是否使用管道传送排序，排序时间取决于多种因素，包括要排序的行数、键大小和行宽。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。如果缓冲池里没有足够的空间，可以将此临时表中的页写到磁盘。当所有排序都完成，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”。

将下推运算符分组并排序

在有些情况下，优化器可以选择将排序或聚集操作从“关系数据服务”组件下推到“数据管理服务”。下推这些操作将改善性能，因为这样“数据管理服务”组件可直接将数据传送至一个排序或聚集例程。如果不使用此下推，“数据管理服务”首先把此数据传送至“关系数据服务”，然后“关系数据服务”再与排序或聚集例程交互。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
```

排序中的分组操作

当排序产生 GROUP BY 操作所需的顺序时，优化器在排序时可执行 GROUP BY 聚集的部分或全部。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。

排序中的聚集需要以下三种聚集阶段以确保返回正确结果。

1. 聚集的第一个阶段“部分聚集”计算聚集值，直到填满排序堆为止。在部分聚集中，接受未聚集数据并产生部分聚集。如果已填满排序堆，其余数据会溢出到磁盘，包括在当前排序堆中已计算的所有部分聚集。在复位排序堆之后，开始新的聚集。
2. 聚集的第二个阶段“中间聚集”提取所有溢出的排序运行结果，并根据分组键进一步聚集。由于分组键列是分区键列的一个子集，故不能完成聚集。中间聚集使用现有的部分聚集来产生新的部分聚集。此阶段并不总是会发生。它用于分区内和分区间并行性。在分区内并行性情况下，当全局分组键可用时完成分组。在分区间并行性的情况下，如果分组键是用来在分区间分组的分区键的子集，从而要求重新分区来完成聚集时，会发生这种情况。当每个代理进程在缩减为单个代理进程以完成聚集之前完成对其溢出的排序运行结果的合并时，在分区内并行性中将存在相似的情况。
3. 聚集的最后阶段“最终聚集”使用所有部分聚集并产生最终聚集。此步骤总是通过 GROUP BY 运算符执行。排序不能执行完整的聚集，因为它们不能保证排序不会被分割。完整的聚集接收未聚集的数据，然后产生最终聚集。如果分区不禁止它的使用，这种聚集方法通常用于将已处于正确顺序的数据进行分组。

相关概念:

- 第 214 页的『排序性能的准则』

相关参考:

• 第 324 页的『sortheap － 排序堆大小』
• 第 323 页的『sheapthres － 排序堆阈值』

优化策略

本节描述优化器可能对分区内并行性和多维群集（MDC）表使用的特殊策略。

分区内并行性的优化策略

如果在编译 SQL 语句时指定了并行度，优化器能选择一个存取方案来在单个数据库执行查询。

在执行时，创建称为“子代理进程”的多个数据库代理进程来执行该查询。子代理进程的数目小于或等于编译该 SQL 语句时指定的并行度。

要并行化存取方案，优化器将它划分为两个部分，每个子代理进程运行一部分，协调代理进程运行另一部分。子代理进程通过表队列将数据传送至协调代理进程或其它子代理进程。在分区数据库中，子代理进程能通过表队列从其它数据库分区中的子代理进程发送或接收数据。

分区内并行扫描策略

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，需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进程。一个子代理进程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，需根据索引键值和一个键值的索引条目数，将索引划分为许多记录范围。并行索引扫描的执行方式类似于并行表扫描，将给子代理进程分配一个范围内的记录。当子代理进程处理完当前范围时，会给它分配一个新的范围。

优化器确定扫描单位（页或行）和扫描粒度。

并行扫描将工作均匀地分布在各子代理进程中。并行扫描的目标是平衡所有子代理进程的负载，并使它们保持相同的繁忙程度。如果繁忙子代理进程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，其它存取方案策略可能导致数据不平衡。优化器选择并行策略，以便维护子代理进程间的数据平衡。

分区内并行排序策略

优化器能选择下列其中一个并行排序策略:

- **循环排序**

  这也称为*再分布排序*。这种方法有效地使用共享内存，将数据尽可能均匀地再分布到所有子代理进程中。它使用循环算法来提供均匀分布。它首先为每个子代理进程创建个别排序。在插入阶段期间，子代理进程以循环方式插入每个个别的排序中以使数据分布更加均匀。

- **分区排序**

这类似于循环排序，在循环排序中，要为每个子代理进程创建一个排序。子代理进程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表和外部表是一个分区排序，则子代理进程可使用合并连接来连接对应的分区并且并行执行。

- **复制排序**

  如果每个子代理进程需要全部排序输出，则使用这种排序。创建一个排序，当将行插入排序时使子代理进程同步。当完成排序时，每个子代理进程都读取整个排序。如果行数较小，可使用此排序来重新平衡数据流。

- **共享排序**

  此排序与复制排序相同，只是子代理进程要对已排序的结果打开一个并行扫描以便用一种与循环排序相似的方式在子代理进程中分布数据。

分区内并行临时表

子代理进程可以协同工作，将行插入同一个表中来产生临时表。这称为共享临时表。子代理进程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。

分区内并行聚集策略

子代理进程可以并行执行聚集操作。聚集操作要求根据分组列将数据排序。如果可以保证一个子代理进程接收一组分组列值的所有行，则该程序可以执行完整的聚集。如果由于先前的分区排序，已根据分组列将数据流分区，则这种情况可发生。

否则，子代理进程可以执行部分聚集，并使用另一种策略来完成该聚集。其中部分策略如下:

- 通过一个合并表队列将部分聚集的数据发送至协调代理进程。协调程序完成聚集。
- 将部分聚集的数据插入一个分区排序中。在分组列上将该排序分区，并保证一组分组列的所有行都包含在一个排序分区中。
- 如果需要复制数据流以便平衡处理，可将部分聚集的数据插入一个复制排序中。每个子代理进程使用复制排序完成聚集，并接收完全相同的聚集结果副本。

分区内并行连接策略

子代理进程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表上分区和／或复制数据流，可将连接并行化。例如，如果因为并行扫描已将一个嵌套循环连接的外部流分区，而且内部流由每个子代理进程单独重新求值，则可将该连接并行化。如果一个合并连接的内部和外部流因为分区排序都已按值分区，则可将该连接并行化。

相关概念:

- 第 80 页的『应用程序的并行处理』
- 第 160 页的『MDC 表的优化策略』

MDC 表的优化策略

如果创建多维群集（MDC）表，因为优化器可以应用附加的优化策略，可以提高许多查询的性能。这些策略主要基于改进的块索引效率，但在多维上进行群集的优点也允许更快速的数据检索。

注: MDC 表优化策略也可以实现分区内并行性和分区间并行性的性能优点。

考虑 MDC 表的下列特定优点:

- 维块索引查找可以标识表的所需部分，并仅快速扫描所需的块。
- 因为块索引小于 RID 索引，所以查找更快。
- 可以在块级别执行索引 AND 和 OR 运算，并可以将这些运算与 RID 组合在一起。
- 保证在扩展数据块内群集数据，这使得检索更快。

考虑在 **region** 和 **month** 列上定义了维且命名为 **sales** 的 MDC 表的以下简单示例:

```
SELECT * FROM SALES
    WHERE MONTH='March' AND REGION='SE'
```

对于此查询，优化器可以执行维块索引查找来寻找出现在月份为三月且地区为 SE 的块。然后该查询可以仅快速扫描表的结果块来访存结果集。

相关概念:

- 第 20 页的『MDC 表的表和索引管理』

具体查询表

对于复杂查询，尤其是可能需要执行下列某些操作的查询，具体查询表（MQT）能够极大地改进响应时间:

- 在一个或多个维上聚集的数据
- 在一组表上连接和聚集数据
- 通常存取的数据子集（即“热”水平或垂直分区）中的数据
- 在分区数据库环境中来自表或表的一部分的已重新分区的数据

MQT 的知识已经集成到了 SQL 编译器中。在 SQL 编译器中，查询重写阶段和优化器将查询与 MQT 匹配，并确定是否要用 MQT 取代存取基本表的查询。如果使用 MQT，则 EXPLAIN 设施可以提供关于选择了哪个 MQT 的信息。

因为 MQT 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数据存取、创建索引以及发出 RUNSTATS 的准则也适用于 MQT。

为了帮助您理解 MQT 的功能，以下示例说明了多维分析查询以及它如何利用 MQT。

在此示例中，假设这样一个数据库方案：数据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。此模式分类为多星型模式，因为两个大表（一个包含交易商品，另一个标识购买交易）一起成为星型的中心。

描述一项交易的三个层次结构维：产品、位置和时间。产品层次结构存储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、州和国家或地区信息，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。日期维是使用内置函数从交易的日期字段中抽取的。此模式中的其它表表示客户的帐户信息和客户信息。

使用以下层次结构的每一级别的销售总额和销售件数来创建 MQT:

- 产品
- 位置
- 时间，由年、月和日组成。

从此存储的聚集数据可以满足许多查询。以下示例显示如何创建一个 MQT，该 MQT 按照 product group 和 product line 维、按照 city、state 和 country 维以及按照 Time 维来计算销售金额和销售件数。在它的 GROUP BY 子句中，还包括几个其它的列。

```
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;
```

可利用这类预先计算的总额的查询包括以下几种:

- 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
- 特定国家或地区的销售总额。

当 MQT 未包括以上任何一个查询的准确答案时，使用 MQT 计算答案的成本可能明显地小于使用大型基本表的成本，因为此答案的一部分已计算出来。MQT 可以减少基本数据的成本高昂的连接、排序和聚集。

下列样本查询将获得明显的性能提高，因为它们可以使用示例 MQT 中已经计算出的结果。

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;
```

基本表越大，在响应时间中的改进可以越大，因为 MQT 的增长比基本表的增长慢得多。 MQT 可以通过在构建和刷新 MQT 时执行一次计算，并对许多查询重复使用 MQT 的内容可以有效地消去查询之间的重叠工作。

相关概念:

- 第 182 页的『设计顾问程序』
- 第 147 页的『分区数据库中的重复具体查询表』

联合数据库查询编译器阶段

本节描述联合数据库系统中附加的查询处理阶段。它还提供可改善联合数据库查询性能的建议。

联合数据库下推分析

对于联合数据库中的查询，优化器执行下推分析来查明是否可以在远程数据源上执行某个操作。操作可以是一个函数，例如，关系运算符、系统函数或用户函数或 SQL 运算符，例如，GROUP BY 或 ORDER BY 等。

注: 虽然 DB2® SQL 编译器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。

如果不能将函数下推到远程数据源，则它们可显著影响查询性能。考虑强制在本地而不在数据源对选择谓词求值的效果。这种求值可能要求 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。网络约束和大表大小可使性能受到损害。

未下推的运算符还可能显著地影响查询性能。例如，让 GROUP BY 运算符在本地聚集远程数据也会要求 DB2 从远程数据源检索整个表。

例如，假设昵称 N1 引用 DB2 OS/390® 或 z/OS 版数据源中的数据源表 EMPLOYEE。同时假设该表有 10,000 行，其中一列包含职员的姓，一列包含工资。考虑以下语句:

```
SELECT LASTNAME, COUNT(*)  FROM N1
   WHERE LASTNAME  > 'B' AND SALARY > 50000
   GROUP BY LASTNAME;
```

取决于 DB2 和 DB2 OS/390 或 z/OS 版上的整理顺序是否相同，考虑几种可能性:

- 如果整理顺序相同，可能将查询谓词下推到 DB2 OS/390 或 z/OS 版中。在数据源对结果进行过滤和分组通常比将整个表复制到 DB2 然后在本地执行那些操作更有效。对于以上查询，该谓词和 GROUP BY 操作可在数据源中执行。
- 如果整理顺序不同，则不能在数据源对整个谓词求值。但是，优化器可能决定下推谓词的 `SALARY > 50000` 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，则优化器可能决定在 DB2 中本地执行 `GROUP BY` 操作是最好（成本最低）的方法。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。

通常，目标是确保优化器在数据源处对函数和运算符求值。许多因素影响是否在远程数据源对函数或 SQL 运算符求值。将要评估的因素分类为下列三组:

- 服务器特征
- 昵称特征
- 查询特征

影响下推机会的服务器特征

某些特定于数据源的因素会影响下推机会。一般情况下，这些因素由于 DB2 支持的丰富的 SQL 语言而都存在。与查询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 能力: 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，但某些数据源限制 GROUP BY 列表中的项数，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 约束: 每个数据源可能有不同的 SQL 限制。例如，某些数据源需要参数标记才能将值绑定至远程 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绑定机制。如果 DB2 不能确定一个好方法来绑定函数的值，必须在本地对此函数求值。

SQL 限制: 尽管 DB2 可能允许使用比它的远程数据源大的整数，但不能将超出远程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。

服务器细节: 有几个因素都归入此类别。一个示例为是将空值排序为最高或最低值，还是取决于排序。如果在某个与 DB2 不同的数据源对空值排序，则对可空表达式执行的 ORDER BY 操作无法以远程方式求值。

整理顺序: 检索数据以进行本地排序和比较通常会降低性能。因此，可考虑将联合数据库配置为使用数据源所用的整理顺序。如果配置联合数据库以使用与数据源所用的相同的整理顺序，然后将 *collating_sequence* 服务器选项设置为“Y”，则优化器可考虑下推许多查询操作（如果改进性能结果）。

如果整理顺序相同，则可以下推下列操作:

- 字符或数字数据的比较
- 字符范围比较谓词
- 排序

但是，如果联合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。如果将此语句落实给不区分大小写的数据源，则比较语句可能返回意外的结果。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，并且给这些字符指定不同的加权。

为了提高性能，联合服务器允许在数据源处进行排序和比较。例如，在 DB2 UDB OS/390 或 z/OS 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。要使用联合服务器来检索根据 ORDER BY 子句排序的 DB2 OS/390 或 z/OS 版数据，配置联合数据库以让它使用基于 EBCDIC 代码页的预定义整理顺序。

如果联合数据库和数据源的整理顺序不同，DB2 将数据检索到联合数据库中。因为用户希望看到根据对联合服务器定义的整理顺序排序的查询结果，所以通过在本地对数据排序，联合服务器确保实现这个期望。如果需要查看以数据源的整理顺序排序的数据，则用通过方式落实您的查询，或者在数据源视图中定义该查询。

服务器选项: 有几个服务器选项会影响下推机会。尤其要复查 *collating_sequence*、*varchar_no_trailing_blanks* 和 *pushdown* 的设置。

DB2 类型映射和函数映射因素: DB2 提供的缺省本地数据类型映射设计成为每个数据源数据类型提供足够的缓冲区空间，这就可避免数据丢失。用户可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。

在下列三种情况下，DB2 可以补偿数据源不支持的函数:

- 远程数据源中不存在此函数。
- 该函数存在，但操作数特征违反了函数限制。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。
- 如果以远程方式对该函数求值，则它可能返回不同的结果。这种情况的一个例子是“>”（大于）运算符。对于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。

影响下推机会的昵称特征

以下特定于昵称的因素会影响下推机会。

昵称列的本地数据类型: 确保列的本地数据类型不会妨碍在数据源对谓词求值。使用缺省数据类型映射来避免可能的溢出。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 如何绑定较长的列。这种情况可影响由

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 数据类型的昵称列，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: 使用 SQL 语句 ALTER NICKNAME 来添加或更改昵称的列选项。

使用 *varchar_no_trailing_blanks* 选项来标识不包含结尾空白的列。当检查对这样标明的列执行的所有操作时，编译器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。

使用 *numeric_string* 选项来指示该列中的值是否总是不带结尾空白的数值。

下表描述这些选项。

表 28. 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：如果列只包含数字串且后面带结尾空白，不要指定“Y”。 “N” 否，此列不限于数字数据串。 如果将列的 numeric_string 设置为“Y”，则将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，此选项很有用。由于整理顺序不同，从本地（数据源）求值中排除用此选项标记的那些列。	“N”
varchar_no_trailing_blanks	指定此数据源是否使用非空格填充的 VARCHAR 比较语义。对于不包含结尾空白的变长字符串，某些 DBMS 的非空格填充比较语义返回与 DB2 的比较语义相同的结果。如果确认在一个数据源中的所有 VARCHAR 表／视图列都不包含结尾空白，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。确保考虑可能有昵称的所有对象（包括视图）。 “Y” 此数据源的非空格填充比较语义与 DB2 的类似。 “N” 此数据源的非空格填充比较语义与 DB2 的不同。	“N”

影响下推机会的查询特征

查询可引用一个 SQL 运算符，它可能涉及多个数据源中的昵称。该操作必须在 DB2 中执行以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。不能在远程数据源直接对该运算符求值。

相关概念:

- 第 165 页的『分析在何处对联合查询求值的准则』

分析在何处对联合查询求值的准则

DB2® 提供了两个实用程序来说明在何处对查询进行求值:

- Visual Explain。用 **db2cc** 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。

  如果下推查询，应会看到 RETURN 运算符。RETURN 运算符是标准的 DB2 运算符。对于从昵称选择数据的 SELECT 语句，还会看到 SHIP 运算符。SHIP 运算符是联合数据库操作所独有的。它可以更改数据流的服务器属性，并可将本地运算符与远程运算符区分开来。该 SELECT 语句是使用数据源支持的 SQL 语言生成的。它可包含该数据源的任何有效的查询。

  如果可以完全将 INSERT、DELETE 或 UPDATE 查询下推至远程数据库，则 SHIP 语句可能不会出现在存取方案中。会对 RETURN 运算符显示所有以远程方式执行的 INSERT、UPDATE 和 DELETE 语句。但是，如果不能完全下推查询，SHIP 运算符会显示哪些操作是以远程方式执行的。
- SQL 说明。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

在寻求增加下推机会的方法时，请考虑以下关键问题:

- 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流量时，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子查询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。
  - 谓词绑定需求。如果进行远程求值，此谓词要求某些值的绑入吗？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。
- 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
    - 列规格仅存在于 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。
  - GROUP BY 运算符子句包含字符表达式吗？如果包含，验证远程数据源是否与 DB2 一样区分大小写。
- 为什么不对集合运算符进行远程求值？

有几个方面可检查:

– 它的两个操作数完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？

  考虑以下问题:

  – 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，那么远程数据源与 DB2 没有相同的整理顺序或大小写区分方式吗？
  – 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 ORDER BY 列表的列规格吗？

相关概念:

- 第 79 页的『字符转换准则』
- 第 169 页的『联合数据库查询的全局分析』
- 第 167 页的『在联合数据库中的远程 SQL 生成和全局优化』
- 第 531 页的『联合查询信息』

在联合数据库中的远程 SQL 生成和全局优化

对于使用关系昵称的联合数据库查询，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。此远程 SQL 生成帮助产生查询的全局优化存取策略。

优化器使用下推分析的输出来决定是要在 DB2® 以本地方式还是在数据源以远程方式对每个操作进行求值。它根据其成本模型的输出作出决定，该输出不但包括对操作进行求值的成本，还包括在 DB2 和数据源之间传送数据或消息的成本。

虽然目标是产生优化查询，但以下主要因素影响全局优化的输出，因而也会影响查询性能。

- 服务器特征
- 昵称特征

影响全局优化的服务器特征和选项

以下数据源服务器因素可能影响全局优化:

- CPU 速度的相对比率

  使用 *cpu_ratio* 服务器选项来指定与 DB2 CPU 相比，数据源 CPU 速度的快慢。低比率指示数据源计算机 CPU 比 DB2 计算机 CPU 快。如果比率较低，则 DB2 优化器很可能会考虑将需要占用大量 CPU 资源的操作下推到数据源。

- 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。如果是低比率，DB2 优化器考虑将需要占用大量 I/O 资源的操作下推到数据源。

- 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（它指示 DB2 和数据源之间的网络通信较慢）促使 DB2 优化器减少向此数据源（或从中）传送的消息数。如果将该速率设置为 0，优化器将创建需要最小网络流量的存取方案。

- 数据源整理顺序

  使用 *collating_sequence* 服务器选项来指定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。

- 远程方案提示

  使用 *plan_hints* 服务器选项来指定应在数据源生成或使用方案提示。在缺省情况下，DB2 不能将任何方案提示发送到数据源。

  方案提示是为数据源优化器提供额外信息的语句分段。对于某些查询，此信息可以提高性能。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。

影响全局优化的昵称特征

以下特定于昵称的因素会影响全局优化。

索引注意事项： 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。昵称的索引信息最初是在创建昵称时获取的。不收集视图昵称的索引信息。

创建昵称的索引规范： 可为昵称创建索引规范。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。基于昵称创建索引规范的语法与在本地表上创建索引的语法相似。

在下列情况考虑创建索引规范：

- 在创建昵称期间，DB2 无法检索数据源的任何索引信息。
- 需要视图昵称的索引。
- 想鼓励 DB2 优化器将特定的昵称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。

在基于视图的昵称发出 CREATE INDEX 语句之前，考虑您是否需要。如果视图是对一个有索引的表执行的简单 SELECT，那么在本地基于昵称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，如果是在本地对并非由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，可能会损害查询性能。例如，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昵称，并在 DB2 中创建一个局部视图来引用这两个昵称。

目录统计注意事项: 系统目录统计信息描述相关列中昵称的总大小和值的范围。当优化器计算出为处理包含昵称查询的最低费用路径时，使用这些统计信息。昵称统计信息与表统计信息存储在同一目录视图中。

虽然 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有统计数据的更新。而且，DB2 不能对数据源中的对象处理对象定义或结构中的更改，例如，添加一列到对象。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。然后删除当前昵称并重新创建。如果结构信息已更改，则使用此方法。
- 手工更新 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。

相关概念:

- 第 86 页的『影响联合数据库的服务器选项』
- 第 165 页的『分析在何处对联合查询求值的准则』
- 第 169 页的『联合数据库查询的全局分析』

联合数据库查询的全局分析

提供的以下两项实用程序显示全局存取方案:

- Visual Explain。从“控制中心”启动它，或执行 *db2cc* 命令，该命令启动“控制中心”。使用 Visual Explain 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。根据查询类型，还可在 SHIP 或 RETURN 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信息，可以查看 DB2® 优化器作为每个运算符的输入和输出所估计的行数。还可了解执行每个运算符的估计成本，包括通信成本在内。
- SQL 说明。用 *db2expln* 或 *dynexpln* 命令启动它。可用 SQL 说明来查看文本格式的存取方案策略。虽然 SQL 说明没有提供成本信息，您仍然可以获得远程优化器为远程说明功能所支持的那些数据源生成的存取方案。

了解 DB2 优化判定

考虑下列要研究的优化问题和关键范围以改善性能:

- 为什么不对同一数据源的两个昵称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。

  – 连接结果（Visual Explain 生成的）中的行数。该连接产生比组合的两个昵称更大的行集合吗？这些数值有意义吗？如果回答是否定的，则考虑手工更新昵称统计信息（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。
  – 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，则考虑手工更新昵称统计信息（SYSSTAT.TABLES）。
- 为什么远程数据源没有完全对该语句求值？

  DB2 优化器执行基于成本的优化。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 CPU 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。检查谓词排序更改。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排序生成不同的方案。如果是这样，这是远程优化器内在的问题。考虑修改 DB2 输入的谓词排序，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，则网络流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。

相关概念:

- 第 86 页的『影响联合数据库的服务器选项』
- 第 162 页的『联合数据库下推分析』
- 第 165 页的『分析在何处对联合查询求值的准则』
- 第 167 页的『在联合数据库中的远程 SQL 生成和全局优化』
- 第 531 页的『联合查询信息』
- 第 542 页的『示例五：联合数据库方案』

第 7 章 SQL 说明设施

“说明”设施允许您捕获有关由优化器选择的存取方案的信息以及帮助您调整查询的性能信息。

SQL 说明设施

SQL 编译器可以捕获关于存取方案和静态或动态 SQL 语句的环境的信息。已捕获的信息帮助您了解如何执行单个 SQL 语句，以便您可以调整语句和数据库管理器配置来提高性能。

因为下列原因，收集和使用说明数据:

- 了解数据库管理器如何存取表和索引来满足查询
- 评估性能调整操作

  当您更改数据库管理器、SQL 语句或数据库的某个方面时，应该检查说明数据来了解您的操作如何更改了性能。

捕获的信息包括:

- 处理查询的操作顺序
- 成本信息
- 谓词和每个谓词的选择性估计
- 对在捕获说明信息时 SQL 语句中引用的所有对象的统计信息
- 使用主机变量、参数标记或专用寄存器的值来重新优化 SQL 语句。

在可以捕获说明信息之前，应创建优化器将说明信息存储在其中的关系表，并设置确定捕获哪种类型的说明信息的专用寄存器。

要显示说明信息，可以使用命令行工具或 Visual Explain。使用的工具确定您如何设置注册表变量，这些注册表变量确定收集哪些说明数据。例如，如果您期望仅使用 Visual Explain，则需要仅捕获快照信息。如果期望使用其中一个命令行实用程序或使用定制 SQL 语句对说明表执行详细分析，则应该捕获所有说明信息。

相关概念:

- 第 172 页的『说明工具』
- 第 173 页的『使用说明信息的准则』
- 第 174 页的『说明信息的说明表和组织』
- 第 179 页的『捕获说明信息的准则』
- 第 181 页的『分析说明信息的准则』
- 第 511 页的『SQL 说明工具』

用于收集和分析说明信息的工具

可以用几种方式执行“说明”设施，取决于您要执行的性能分析的种类。本节描述实现“说明”设施的工具和可以使用收集的信息的方式。

说明工具

DB2® 提供一种全面的说明设施，该设施提供关于优化器为 SQL 语句选择的存取方案的详细信息。存储说明数据的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。几种工具或方法使您可灵活地捕获、显示和分析说明信息。

详细的优化器信息允许对存取方案进行深入的分析，该信息存储在说明表中，独立于实际的存取方案本身。使用下列其中一个或多个方法，从说明表获取信息:

- 使用 Visual Explain 来查看说明快照信息

  从“控制中心”调用 Visual Explain 来查看查询存取方案的图形显示。可以分析静态和动态 SQL 语句。

  Visual Explain 允许您查看在另一个平台上捕获或获取的快照。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。
- 使用 *db2exfmt* 工具来显示预格式化输出中的说明信息
- 使用 *db2expln* 和 *dynexpln* 工具

  要查看静态 SQL 语句的一个或多个程序包可用的存取方案信息，从命令行使用 *db2expln* 工具。*db2expln* 显示所选存取方案的实际实现。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说明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 语句变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，则使用说明设施。

  db2expln 工具通过检查生成的实际存取方案来对在运行时将发生的操作提供一个相对简洁的英语风格的概述。
- 对照说明表编写自己的查询

  编写自己的查询便于处理输出和比较不同的查询或比较一段时间内的同一查询。

注: 命令行说明工具和其它工具（如 *db2batch*、*dynexpln*、*db2vexp* 和 *db2_all*）的位置在 *sqllib* 目录的 *misc* 子目录中。如果将这些工具移出此路径，则命令行方法可能不起作用。

下表汇总了 DB2 说明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 29. 说明设施工具

期望的特征	Visual Explain	说明表	db2exfmt	db2expln	dynexpln
GUI 界面	是				
文本输出			是	是	是

表 29. 说明设施工具 (续)

期望的特征	Visual Explain	说明表	db2exfmt	db2expln	dynexpln
“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® 应用程序请求器		是			
详细的优化器信息	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。

相关概念:

- 第 517 页的『dynexpln』
- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

相关参考:

- 第 545 页的附录 D, 『db2exfmt － 说明表格式化』
- 第 512 页的『db2expln － SQL 说明』

使用说明信息的准则

因为下列两个主要目的，使用说明信息:

- 了解应用程序性能更改的原因
- 评估性能调整工作

性能更改分析

为了帮助您了解查询性能发生变化的原因，您需要更改前后的说明信息，可以通过执行下列步骤来获得这些信息:

- 捕获作出任何更改前查询的说明信息，并保存生成的说明表，或者，可以保存 `db2exfmt` 说明工具的输出。
- 如果您不想或不能访问 Visual Explain 来查看此信息，则保存或打印当前的目录统计信息。还可以使用 `db2look` 生产工具来帮助执行此任务。
- 保存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。

按此方式收集的信息为将来分析提供参考点。对于动态 SQL 语句，您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，您也可在绑定时收集此信息。要分析性能更改，将已收集的信息与关于查询收集的信息和启动分析时的环境比较。

举一个简单的例子，您的分析可以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绑定时的索引层数。于是，您可以选择执行下列其中一项操作:

- 重组该索引
- 收集您的表和索引的新统计信息
- 当重新绑定您的查询时收集说明信息

执行其中一项操作之后，再次检查存取方案。如果再次使用索引，查询的性能可能不再是一个问题。如果仍未使用索引或如果性能仍是一个问题，则执行第二操作并检查结果。重复这些步骤，直到解决问题为止。

性能调整工作的评估

可以采取几个操作来帮助提高查询性能，如调整配置参数、添加容器以及收集新的目录统计信息等。

当在其中任何一个区域中进行更改后，您可使用 SQL 说明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，如果您根据索引准则来添加索引或具体查询表（MQT），则说明数据可以帮助您确定是否按预期的那样实际使用该索引或具体查询表。

尽管说明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，但准确测量一个查询的性能提高的唯一方法是使用基准程序测试技术。

相关概念:

- 第 171 页的『SQL 说明设施』
- 第 174 页的『说明信息的说明表和组织』
- 第 179 页的『捕获说明信息的准则』
- 第 182 页的『设计顾问程序』
- 第 160 页的『具体查询表』

相关参考:

- 第 545 页的附录 D, 『db2exfmt － 说明表格式化』

搜集的说明信息

本节列示并描述存储说明数据的每个表，描述可以从收集的数据中检索的信息种类，并提供捕获对性能分析有用的信息的准则。

说明信息的说明表和组织

所有说明信息都是按说明实例的概念组织的。一个说明实例代表说明设施对一条或多条 SQL 语句的一次调用。在一个说明实例中捕获的说明信息包括 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。例如，一个说明实例可以由以下其中任何一项组成:

- 静态 SQL 语句的一个程序包中所有合格的 SQL 语句。可以捕获 SELECT、SELECT INTO、UPDATE、INSERT、VALUES、VALUES INTO 和 DELETE 语句的说明信息
- 增量绑定 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）

说明表信息反映存取方案中运算符和数据对象之间的关系。下图显示这些表之间的关系。

说明信息存储在下列表中:

表 30. 存储说明数据的关系表

表名	描述
EXPLAIN_ARGUMENT	包含有关每个个别运算符的唯一特征的信息（如果存在的话）。
EXPLAIN_INSTANCE	所有“说明”信息的主控制表。说明表中的每一行数据都显式地链接至此表中唯一的一行。有关要说明的 SQL 语句的源和环境信息的基本信息保存在此表中。
EXPLAIN_OBJECT	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。
EXPLAIN_OPERATOR	包含 SQL 编译器为满足 SQL 语句所需的所有运算符。
EXPLAIN_PREDICATE	标识特定的运算符所应用的谓词。
EXPLAIN_STATEMENT	包含对于不同级别的说明信息而存在的 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。 当请求说明快照时，要记录附加说明信息，以便描述 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。其它应用程序不能使用此格式。
EXPLAIN_STREAM	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。在 EXPLAIN_OPERATOR 表中表示数据流中涉及的运算符。
ADVISE_WORKLOAD	允许用户向数据库描述其工作负载。表中的每一行代表工作负载中的一条 SQL 语句，并由相关的频率描述。db2advis 工具使用此表来收集并存储工作负载信息。
ADVISE_INSTANCE	包含有关执行 db2advis 的信息，包括起始时间。每次执行 db2advis 都会产生相应的一行。
ADVISE_INDEX	存储有关建议的索引的信息。该表可由 SQL 编译器、db2advis 实用程序或用户填充。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。
ADVISE_MQT	包含 CREATE DDL、用来定义建议的每个 MQT 的查询、每个 MQT 的统计信息（例如，COLSTATS（每一列））（采用 XML 格式）以及 NUMROWS 等等，还包含采样查询以获得每个 MQT 的采样统计信息。
ADVISE_TABLE	存储使用为建议的 MQT、MDC 和分区而最终提供的“设计顾问程序”建议来创建表时所使用的 DDL，这取决于指定的选项和生成的建议。
ADVISE_PARTITION	存储由 db2advis 生成并评估的虚拟分区。

注: 在缺省情况下，并不创建上面所有的表。要创建它们，运行 *sqllib* 子目录的 *misc* 子目录中的 EXPLAIN.DDL 脚本。

说明表可能对多个用户是公共的。但是，可以为一个用户定义说明表，然后可以使用相同名称来指向已定义的表来为每个附加用户定义别名。共享公用说明表的每个用户必须对那些表有插入许可权。

相关概念:

- 第 171 页的『SQL 说明设施』
- 第 176 页的『数据对象的说明信息』
- 第 177 页的『实例的说明信息』
- 第 176 页的『数据运算符的说明信息』
- 第 511 页的『SQL 说明工具』

数据对象的说明信息

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: 说明设施记录有关该对象的信息，如下列信息:

- 创建时间
- 上次收集该对象的统计信息的时间
- 有关是否将对象（仅表或索引对象）中的数据排序的指示
- 对象（仅表或索引对象）中的列数
- 对象（仅表或索引对象）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）
- 预取大小和扩展数据块大小（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子页数和树中的级别数
- 此对象的索引中相异完整键值的数目
- 表中溢出记录的总数

相关概念:

- 第 174 页的『说明信息的说明表和组织』
- 第 177 页的『实例的说明信息』
- 第 176 页的『数据运算符的说明信息』
- 第 181 页的『分析说明信息的准则』

数据运算符的说明信息

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 编译器确定必需的操作，如表扫描、索引扫描、嵌套的循环连接或 group-by 运算符。

除显示存取方案中使用的运算符和有关每个运算符的信息外，说明信息还显示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，包括捕获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开销
- 通信成本（以帧计）

Timerons 是发明的相对计量单位。Timerons 由优化器基于在使用数据库时会更改的内部值（如统计信息）确定。因此，不能保证对 SQL 语句的 timerons 评测在每次确定以 timerons 计的估计成本时都是相同的。

运算符属性: 说明设施记录下列信息来描述每个运算符的属性:

- 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）

相关概念:

- 第 174 页的『说明信息的说明表和组织』
- 第 176 页的『数据对象的说明信息』
- 第 177 页的『实例的说明信息』
- 第 181 页的『分析说明信息的准则』

实例的说明信息

说明实例信息存储在 EXPLAIN_INSTANCE 表中。有关在一个实例中的每个 SQL 语句的其它特定信息存储在 EXPLAIN_STATEMENT 表中。

说明实例标识: 使用此信息来唯一地标识每个说明实例，并将 SQL 语句的信息与该设施的给定调用相关:

- 请求说明信息的用户
- 说明请求开始的时间
- 包含说明的 SQL 语句的程序包的名称
- 包含说明的 SQL 语句的程序包的模式
- 包含语句的程序包的版本
- 是否收集了快照信息

环境设置: 捕获有关 SQL 编译器优化您的查询的数据库管理器环境的信息。环境信息包括下列各项:

- DB2® 级别的版本号和发行版号

- 编译该查询的并行度

  CURRENT DEGREE 专用寄存器、DEGREE 绑定选项、SET RUNTIME DEGREE API 和 *dft_degree* 配置参数确定编译特定查询的并行度。
- 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
- 当编译查询时指定的游标行分块的类型
- 该查询运行的隔离级别
- 编译该查询时各种配置参数的值。当获取说明快照时，记录下列参数:
  - 排序堆大小（*sortheap*）
  - 活动应用程序平均数（*avg_appls*）
  - 数据库堆（*dbheap*）
  - 锁定列表的最大存储器（*locklist*）
  - 升级前锁定列表的最大百分比（*maxlocks*）
  - CPU 速度（*cpuspeed*）
  - 通信带宽（*comm_bandwidth*）

SQL 语句标识: 可能为每个说明实例已说明多个 SQL 语句。除了唯一标识说明实例的信息外，下列信息还帮助标识个别的 SQL 语句:
- 语句的类型: SELECT、DELETE、INSERT、UPDATE、定位的 DELETE 和定位的 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

EXPLAIN_STATEMENT 表中的 QUERYTAG 和 QUERYNO 字段包含设置为说明过程的一部分的标识。对于在一个 CLP 或 CLI 会话期间落实的动态说明 SQL 语句，当 EXPLAIN MODE 或 EXPLAIN SNAPSHOT 是活动时，将 QUERYTAG 设置为 “CLP” 或 “CLI”。在此情况下，对于每个语句，QUERYNO 缺省为按 1 或更大的值递增的一个数。对于所有其它动态说明 SQL 语句（不是来自 CLP 、CLI 或不使用 EXPLAIN SQL 语句），将 QUERYTAG 设置为空白，且 QUERYNO 始终是 “1”。

成本估计: 对于每个说明语句，优化器记录执行所选存取方案的相关成本估计值。此成本按发明的称为 *timeron* 的相对计量单位来陈述。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，而只估计资源消耗。
- 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。几个运行时因素影响经过时间，包括: 系统工作负载、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: 对于每个已说明的语句，记录 SQL 语句文本的两个版本。一种版本是 SQL 编译器从应用程序接收的代码。另一种版本是从查询的内部编译器表示反向转换的。尽管此转换看上去类似于其它 SQL 语句，但它既不必遵循正确的 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在的 SQL 上下文。要了解 SQL 编译器如何重新编写您的查询来获得更

好的优化，将用户编写的语句文本与 SQL 语句的内部表示比较。重新编写的语句也显示该环境中影响您的语句的其它元素，如触发器和约束。此“优化的”文本使用的一些关键字包括:

$C*n*
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.T*n*
派生的表的名称，其中，*n* 表示整数值。

$INTERNAL_FUNC$
此标记指示 SQL 编译器使用的一个函数是否存在，该函数用于说明的查询，但不可通用。

$INTERNAL_PRED$
此标记指示 SQL 编译器在已说明的查询的编译期间添加的一个谓词是否存在，但不可通用。该编译器使用一个内部谓词来满足由于触发器和约束而添加至原始 SQL 语句的附加上下文。

RID
一个标记，用于标识特定行的“行标识”（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，此前缀出现在文本开始处。此前缀之后显示影响 SQL 语句的编译和解析的任何触发器或约束的名称列表。

相关概念:

- 第 174 页的『说明信息的说明表和组织』
- 第 176 页的『数据对象的说明信息』
- 第 176 页的『数据运算符的说明信息』
- 第 181 页的『分析说明信息的准则』

相关参考:

- 第 419 页的『comm_bandwidth － 通信带宽』
- 第 324 页的『sortheap － 排序堆大小』
- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
- 第 309 页的『dbheap － 数据库堆』
- 第 420 页的『cpuspeed － CPU 速度』
- 第 345 页的『avg_appls － 活动应用程序的平均数目』
- 第 396 页的『dft_degree － 缺省并行度』

捕获说明信息的准则

如果在编译 SQL 语句时请求说明数据，则捕获该数据。考虑您期望在请求说明数据时如何使用捕获的信息。

注:

1. 如果在运行时编译增量绑定 SQL 语句，则在运行时而不是绑定时将数据放入说明表中。对于这些语句，插入的说明表限定符和授权标识是程序包所有者（而不是运行该程序包的用户）的限定符和授权标识。
2. 仅当编译 SQL 语句时才捕获说明信息。在初始编译之后，当环境更改要求动态 SQL 语句时或当说明设施活动时，将再次编译这些动态 SQL 语句。如果对同一 SQL 语句发出相同的 PREPARE 语句，则每当准备或执行 PREPARE 语句时，都将编译该 SQL 语句并捕获说明数据。
3. 如果程序包是使用绑定选项 REOPT ONCE/ALWAYS 绑定的，则将编译包含主机变量、参数标记或专用寄存器的 SQL 语句，并且将使用这些变量的真实值（如果它们已知的话）和使用缺省估计（如果编译时不知道这些值的话）来创建存取路径。
4. 如果使用了 FOR REOPT ONCE 子句，则尝试使指定的 SQL 语句与程序包高速缓存中的同一语句相匹配。将使用此已经重新优化的高速缓存的 SQL 语句的值来重新优化指定的 SQL 语句。如果用户具有必需的存取特权，则说明表将包含新生成的经过重新优化的存取方案和用于此重新优化的值。
5. 在多分区系统中，应该在最初编译和使用 REOPT ONCE 来重新优化语句的分区上说明该语句，否则，将返回错误。

捕获说明表中的信息

- **静态或增量绑定 SQL 语句:**

  在 BIND 或 PREP 命令上指定 EXPLAIN ALL 或 EXPLAIN YES 选项或在源程序中包含静态 EXPLAIN SQL 语句。

- **动态 SQL 语句:**

  在下列任何一种情况下捕获说明表信息:

  – 将 CURRENT EXPLAIN MODE 专用寄存器设置为:
    - YES: SQL 编译器捕获说明数据并执行该 SQL 语句。
    - EXPLAIN: SQL 编译器捕获说明数据，但不执行该 SQL 语句。
    - RECOMMEND INDEXES: SQL 编译器捕获说明数据，并将建议的索引放进 ADVISE_INDEX 表中，但不执行该 SQL 语句。
    - EVALUATE INDEXES: SQL 编译器使用用户放进 ADVISE_INDEX 表中的索引进行求值。在 EVALUATE INDEXES 方式中，说明所有动态语句，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 编译器将选择使用虚拟索引。否则，忽略这些索引。要了解已提出的索引是否有用，复查 EXPLAIN 结果。
- 已在 BIND 或 PREP 命令上指定了 EXPLAIN ALL 选项。SQL 编译器在运行时捕获动态 SQL 的说明数据，即使将 CURRENT EXPLAIN MODE 专用寄存器设置为 NO 亦如此。该 SQL 语句也将执行，并返回该查询的结果。

捕获说明快照信息

当请求说明快照时，将说明信息按 Visual Explain 需要的格式存储在 EXPLAIN_STATEMENT 表的 SNAPSHOT 列中。其它应用程序不能使用此格式。可从 Visual Explain 本身获取有关说明快照信息内容的附加信息。此信息包括关于数据对象和数据运算符的信息。

当编译 SQL 语句并已请求说明数据时，会捕获说明快照数据，如下所示:

- **静态或增量绑定 SQL 语句**:

  当在 BIND 或 PREP 命令上指定了 EXPLSNAP ALL 或 EXPLSNAP YES 子句时或当源程序包含使用 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的静态 EXPLAIN SQL 语句时，将捕获说明快照。

- **动态 SQL 语句**:

  在下列任何一种情况下捕获说明快照:

  – 发出带 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的 EXPLAIN SQL 语句。对于 FOR SNAPSHOT 子句，仅捕获说明快照信息。对于 WITH SNAPSHOT 子句，除快照信息外，也捕获所有说明信息。

  – 将 CURRENT EXPLAIN SNAPSHOT 专用寄存器设置为:

    - YES: SQL 编译器捕获快照说明数据并执行该 SQL 语句。

    - EXPLAIN: SQL 编译器捕获快照说明数据，但不执行该 SQL 语句。

  – 在 BIND 或 PREP 命令上指定 EXPLAIN ALL 选项。SQL 编译器在运行时捕获快照说明数据，即使 CURRENT EXPLAIN SNAPSHOT 专用寄存器的设置是 NO。它也执行该 SQL 语句。

相关概念:

- 第 171 页的『SQL 说明设施』
- 第 173 页的『使用说明信息的准则』
- 第 174 页的『说明信息的说明表和组织』
- 第 182 页的『设计顾问程序』
- 第 181 页的『分析说明信息的准则』
- 第 511 页的『SQL 说明工具』

分析说明信息的准则

尽管说明信息的主要使用是分析 SELECT 语句的存取路径，但有许多方法用来分析说明数据，以帮助您调整查询和环境。考虑下列几种分析:

- **索引使用**

  适当的索引可明显地有益于性能。使用说明输出，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。在说明输出中，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

  也可使用说明设施来评估是否可使用另一个索引来代替现有的索引，或根本不使用索引。当创建新索引后，使用 RUNSTATS 命令来收集该索引的统计信息并重新编译该查询。经过一段时间，您可能通过说明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具有低的群集比率，则您可能要根据该索引重组表以群集其数据，使用 RUNSTATS 命令来收集索引和表的统计信息，然后重新编译该查询。要确定重组表是否已改进存取方案，对重新编译查询重新检查说明输出。

- **存取类型**

  分析说明输出，并查找存取数据的类型，这种存取对于您正在运行的应用程序的类型而言通常不是最优的。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，因为它们倾向于对键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，应分析说明数据以确定不使用索引扫描的原因。

  - **仅浏览查询**

    “浏览”类型查询的搜索条件可能很含糊，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，您可能指定在返回一些结果前，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，如果说明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。如果您指定此选项，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。

- **连接方法**

  如果一个查询连接了两个表，检查所用的连接的类型。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，可以更改这些典型连接工作的方式。

相关概念:

- 第 171 页的『SQL 说明设施』
- 第 511 页的『SQL 说明工具』
- 第 517 页的『db2expln 和 dynexpln 输出的描述』

设计顾问程序

“DB2® 设计顾问程序”是一个工具，可以帮助您显著提高工作负载性能。选择要为复杂工作负载创建哪些索引、MQT、群集维或分区的任务的难度可能很大。“设计顾问程序”标识了提高工作负载性能所需要的所有对象。如果工作负载中存在一组 SQL 语句，则“设计顾问程序”将为下列各项提供一些建议:

- 新的索引
- 新的具体查询表（MQT）
- 对多维群集（MDC）表的转换

- 表的重新分区
- （通过 GUI 工具）删除指定的工作负载未使用的索引和 MQT

可以使用“设计顾问程序”立即实现这些建议中的一部分建议或所有建议，也可以安排稍后实现这些建议。

“设计顾问程序”可以使用“设计顾问程序”GUI 或命令行工具来帮助简化下列任务:

规划或建立新的数据库

当设计数据库时，使用“设计顾问程序”来执行下列任务:

- 在分区数据库环境或索引、MQT 和 MDC 表的测试环境中生成设计备用项。
- 对于分区数据库环境，可以使用“设计顾问程序”来完成下列任务:
  - 在将数据装入数据库之前确定分区策略。
  - 帮助从单一分区 DB2 数据库迁移到多分区 DB2 数据库。
  - 帮助从另一个数据库产品迁移到多分区 DB2 数据库。
- 对已经手工生成的索引、MQT、MDC 表或分区策略进行评估。

工作负载性能调整

在建立数据库之后，可以使用“设计顾问程序”来:

- 提高特定语句或工作负载的性能。
- 提高总体数据库性能，使用样本工作负载的性能作为衡量标准。
- 提高最频繁执行的查询（例如，由“活动监视器”标识的查询）的性能。
- 确定如何优化新键查询的性能。
- 对“健康中心”提供的一些建议作出响应，这些建议是针对共享内存实用程序或具有大量排序的工作负载中的排序堆问题提出的。
- 查找工作负载中未使用的对象。

设计顾问程序输出

如果使用“设计顾问程序”GUI，则可以从“设计顾问程序”中查看、保存或实现建议。如果从命令行运行“设计顾问程序”，则缺省情况下将把输出打印至标准输出并保存在 ADVISE_TABLE 和 ADVISE_INDEX 表中:

- 对于 MQT、MDC 表和分区策略建议，ADVISE_TABLE 包含 USE_TABLE='Y'。

  也可以在 ADVISE_MQT 表中找到 MQT 建议；也可以在 ADVISE_TABLE 表中找到 MDC 建议；也可以在 ADVISE_PARTITION 表中找到分区策略建议。每次执行“设计顾问程序”时，这些表中的 RUN_ID 值都对应于 ADVISE_INSTANCE 表中的 START_TIME 值。
- 对于索引建议，ADVISE_INDEX 表包含 USE_INDEX='Y' 或 'R'。

每次运行“设计顾问程序”时，都还要对 ADVISE_INSTANCE 表更新一行:

- START_TIME 字段和 END_TIME 字段分别显示实用程序的启动时间和停止时间。
- 如果实用程序成功结束，则 STATUS 字段将包含“COMPLETED”。
- MODE 字段指示是否使用了 -m 选项。
- COMPRESSION 字段指示使用的压缩类型。

可以使用 -o 选项将“设计顾问程序”建议保存至某个文件。保存的“设计顾问程序”输出包含下列元素:

- 为新的索引、MQT、分区策略和 MDC 表提供的 CREATE STATEMENTS。
- 为 MQT 提供的 REFRESH 语句。
- 为新对象提供的 RUNSTATS 命令。
- 如果使用了现有 MQT 和索引并且使用它们来执行工作负载，则它们将出现在建议的脚本中。

注: ADVISE_MQT 表的 COLSTATS 列包含 MQT 的列统计信息。统计信息采用 XML 结构，如下所示:

```
<?xml version=\"1.0\" encoding=\"USASCII\"?>
<colstats>
    <column>
         <name>COLNAME1</name>
         <colcard>1000</colcard>
         <high2key>999</high2key>
         <low2key>2</low2key>
    </column>
    ....

    <column>
         <name>COLNAME100</name>
         <colcard>55000</colcard>
         <high2key>49999</high2key>
         <low2key>100</low2key>
    </column>
</colstats>
```

注意，XML 结构可以包含多个列。对于每一列，会显示列基数（即，该列中的值的个数）以及 high2 和 low2 键（可选）。

在进行一些细微的修改之后，可以将此输出文件作为 CLP 脚本来运行以创建建议的对象。您可能想执行的修改包括:

- 将所有 RUNSTATS 命令语句组合为对新对象或已修改的对象的单个 RUNSTATS 调用。
- 提供比系统生成的标识更适用的对象名。
- 除去或注释掉不想立即实现的对象的任何 DDL。

相关概念:

- 『数据分区』（*《管理指南: 计划》*）
- 第 221 页的『索引的优点和缺点』
- 第 223 页的『索引规划技巧』
- 第 160 页的『具体查询表』
- 『多维群集表』（*《管理指南: 计划》*）

相关参考:

- 『db2advis - DB2 Design Advisor Command』（*Command Reference*）

定义“设计顾问程序”的工作负载

*工作负载*是数据库管理器必须在给定时间段内处理的一组 SQL 语句。例如，在一个月的时间内，数据库管理器可能必须处理 1000 个 INSERT、10000 个 UPDATE、10000 个 SELECT 和 1000 个 DELETE。“设计顾问程序”将分析指定的工作负载并且考虑这样一些因素：例如，工作负载语句的类型、执行特定语句的频率和要生成使运行工作负载的总成本最小的一些建议的数据库的特征。

过程:

设计顾问程序 GUI 方法:

从“设计顾问程序”GUI 工作负载页中，可以创建新的工作负载文件，或者修改先前已有的工作负载文件。可以从以下几种源将一些语句导入文件中:

- 定界文本文件
- 事件监视器表
- Query Patroller 历史数据表（通过从命令行使用 -qp 选项）
- EXPLAINED_STATEMENT 表中的说明语句
- 使用 DB2 快照捕获到的最新 SQL 语句。

在导入 SQL 语句之后，可以添加、更改、修改或除去语句并修改它们的频率。

设计顾问程序命令行方法:

从命令行，使用下列语句来运行“设计顾问程序”：

- 通过直接插入在命令中而输入的单个 SQL 语句
- 在 DB2 快照中捕获到的一组动态 SQL 语句
- 工作负载文件中包含的一组 SQL 语句。

要使用动态 SQL 语句运行“设计顾问程序”：

1. 使用以下命令复位数据库监视器:

   ```
   db2 reset monitor for database database-name
   ```
2. 等待适当的时间间隔以便对数据库执行动态 SQL 语句。
3. 发出带有 -g 选项的 **db2advis** 命令。如果想要将动态 SQL 语句保存在 ADVISE_WORKLOAD 表中以便将来引用，则还应该使用 -p 选项。

要使用工作负载文件中包含的一组 SQL 语句来运行“设计顾问程序”：

1. 手工创建工作负载文件，并将每个 SQL 语句用分号隔开，或者从上面列示的一个或多个源中导入 SQL 语句。
2. 设置工作负载中的语句的频率。缺省情况下，为工作负载文件中的每个语句指定的频率为 1。 SQL 语句的频率表示该语句相对其它语句在工作负载中出现的次数。例如，一个特定的 SELECT 语句在工作负载中可能出现 100 次，而另一个 SELECT 语句则出现 10 次。为了表示这两个语句的相对频率，可指定第一个 SELECT 语句的频率为 10，第二个 SELECT 语句的频率为 1。可以通过在特定语句后面插入下面这一行来手工更改该语句在工作负载中具有的频率或权重 – # SET FREQUENCY *n*，其中 *n* 是想要为该语句指定的频率值。
3. 通过使用 -i 选项并且后跟工作负载文件的名称来运行 **db2advis** 命令。

要对 ADVISE_WORKLOAD 表中包含的工作负载运行“设计顾问程序”，运行带有 -w 选项并且后跟工作负载名称的 **db2advis**。

相关概念:

- 第 182 页的『设计顾问程序』

使用设计顾问程序从单一分区迁移到多分区数据库

可以使用“设计顾问程序”来帮助您从单一分区迁移到多分区数据库。“设计顾问程序”可提供有关对数据进行分区的建议，同时提供有关新索引、具体查询表（MQT）和多维群集（MDC）表的建议。

过程:

1. 更新 DB2 UDB ESE 的产品许可证密钥。
2. 在多分区数据库分区组中至少创建一个表空间。

   注: 因为“设计顾问程序”只能建议对现有表空间进行重新分区，所以在运行“设计顾问程序”之前，分区数据库中必须存在您想要“设计顾问程序”考虑的表空间。
3. 在使用在“设计顾问程序”GUI 中选择的分区功能部件或者使用为 **db2advis** 命令指定的分区选项的情况下运行“设计顾问程序”。
4. 如果正在“控制中心”中使用“设计顾问程序”，则可自动实现分区建议。如果正在使用 **db2advis** 命令，则在运行由“设计顾问程序”生成的 DDL 语句之前需要稍微修改 **db2advis** 输出文件。

相关概念:

- 第 182 页的『设计顾问程序』

相关任务:

- 『使用 db2licm 命令注册 DB2 产品许可证密钥』（*《安装与配置补充手册》*）

设计顾问程序的局限性和限制

1. **对索引建议的限制**
   - 对具体查询表（MQT）建议的索引会提高工作负载性能（这与 REFRESH TABLE 性能相反）。而且，如果工作负载中不包括更新、插入或删除，则对于 IMMEDIATE MQT，将不包括更改（例如，更新）MQT 的性能。因此，建议这些 MQT 对 IMMEDIATE MQT 隐含的唯一键创建唯一索引。隐含的唯一键将基于 MQT 查询定义的 GROUP BY 子句中的那些列。
   - 仅当要选择多维群集时，才建议创建群集 RID 索引。顾问程序将把 RID 群集索引作为一个选项来而不是为表创建 MDC 结构。
2. **对 MQT 建议的限制**
   - “设计顾问程序”建议不要使用增量 MQT。如果想要创建增量 MQT，可使用 REFRESH IMMEDIATE MQT 并使用您选择的分级表来将它们转换为增量 MQT。
   - 对 MQT 建议的索引用来提高工作负载性能而不是 MQT 刷新性能。

- 如果指定的工作负载中不包括更新、插入或删除，则不考虑更新建议的 REFRESH IMMEDIATE MQT 对性能产生的影响。建议 REFRESH IMMEDIATE MQT 对隐含的唯一键创建唯一索引，隐含的唯一键是由 MQT 查询定义的 GROUP BY 子句中的列组成的。

3. **对 MDC 建议的限制**
   - 现有表中必须具有数据；否则，MDC 将不会考虑该表。
   - 除非对命令使用了采样选项 -r 或者在 GUI 工具中选择了 MQT 采样，否则将不考虑 MDC 对新 MQT 的建议。
   - “设计顾问程序”不会为类型表或临时表生成 MDC 建议。
   - “设计顾问程序”不会为联合表生成 MDC 建议。
   - 对于在执行“设计顾问程序”期间所使用的采样数据必须具有存储空间，否则在不相关假设下将只检查已采样的表以找到基本列。在此情况下，将生成警告消息。
   - 将跳过（即，不考虑）没有收集统计信息的表。
   - “设计顾问程序”不会为多列维生成建议。
   - 现有表中必须存在数据以便在进行 MDC 选择时对数据采样。
4. **对分区建议的限制**

   “设计顾问程序”只能建议对“DB2® 企业服务器版”进行分区。如果对 **db2advis** 命令指定了分区选项，则会返回错误。在“设计顾问程序”GUI 中，不能对单一分区的数据库环境选择分区功能。
5. **其它限制**

   在执行“设计顾问程序”期间，会创建模拟目录表。当执行完“设计顾问程序”时，会删除这些表。如果未执行完“设计顾问程序”，可能会导致不删除这些表中的某些表。在此情况下，可以通过从命令行重新启动实用程序来使用“设计顾问程序”删除这些表。要除去模拟目录表，应同时指定 -f 选项和 -n 选项（对于 -n，指定不完整执行时所使用的用户名）。如果不指定 -f 选项，则“设计顾问程序”将生成除去这些表所需要的 DROP 语句。

   应该另外创建一个表空间来存储这些模拟目录表，并将 DROP TABLE RECOVERY 设置为“OFF”。这样就会更容易清除，并且执行“设计顾问程序”的速度也会更快。

相关概念:

- 第 182 页的『设计顾问程序』

第 3 部分 调整和配置系统

第 8 章 操作性能

本章提供关于在运行时影响 SQL 查询的性能的因素的信息。您也可以参阅有关物理数据库设计注意事项的计划信息，特别是为便利分区、多维群集（MDC）表和类似功能部件起见。

内存使用情况

本节描述数据库管理器如何使用内存并列示控制数据库管理器和数据库内存使用的参数。

安排内存的使用

了解 DB2® 如何通过安排内存来帮助您调整内存的使用以提高系统性能。许多配置参数会影响内存的使用。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的内存。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。当数据库服务器运行时，可以增大或减小数据库共享内存中的内存区域大小。

系统管理员需要综合考虑系统内存的均衡使用。操作系统中运行的不同应用程序可能以不同的方式使用内存。例如，一些应用程序可能使用操作系统的高速缓存，而数据库管理器使用其自身的缓冲池（而不是操作系统的高速缓存）进行数据高速缓存。

下图显示了数据库管理器为不同用途分配的各个内存部分。

注: 本图没有显示如何在包含多个逻辑节点的“企业服务器版”环境中使用内存。在此类环境中，每个节点均包含一个“数据库管理器共享内存”集。

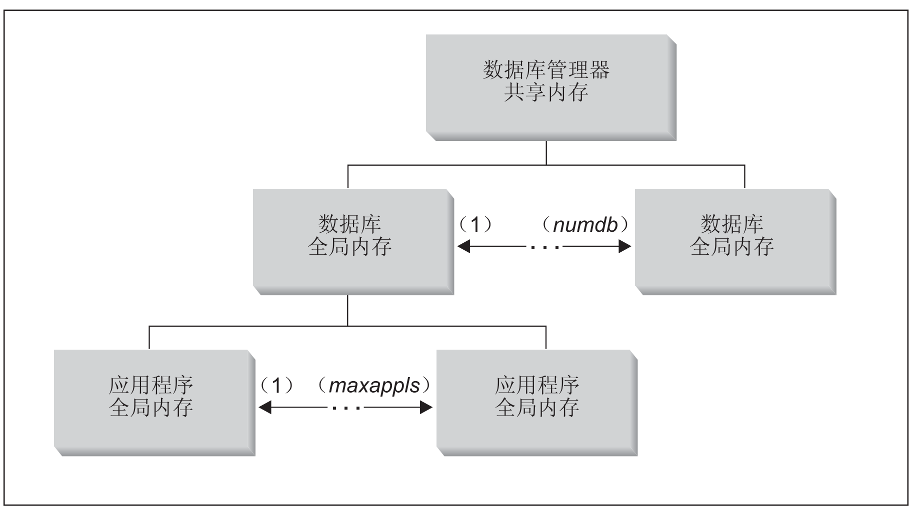


图 19. 数据库管理器所用的内存类型

在发生下列事件时，将为数据库管理器的每个实例分配内存:

- **当启动数据库管理器（db2start）时**: 分配数据库管理器全局共享内存，并且一直会保持此分配状态，直到数据库管理器停止运行（db2stop）为止。此区域中包含数据库管理器在管理所有数据库连接上的活动时所需的信息。当第一个应用程序与数据库连接时，将分配全局内存区和专用内存区。
- **当第一次激活数据库或与数据库连接时**: 将分配数据库全局内存。所有可能与该数据库连接的应用程序均使用数据库全局内存。数据库全局内存的大小由 *database_memory* 配置参数指定。指定的内存可以大于最初所需的内存，以便以后可以动态地分配附加内存。虽然在数据库活动时不能增加或减少数据库全局内存的总容量，但是可以调整数据库全局内存中的各个内存区的大小。此类内存区包括缓冲池、锁定表、数据库堆、实用程序堆、程序包高速缓存和目录高速缓存。在已启用数据库管理器分区内并行性配置参数（intra_parallel）的环境中，或者在已启用连接集中器的环境中，还可以分配共享排序堆作为数据库全局内存的一部分。
- **当应用程序与数据库连接时**: 在分区数据库环境中、在已启用数据库管理器分区内并行性配置参数（*intra_parallel*）的非分区数据库环境中或在已启用连接集中器的环境中，可将多个应用程序分配至*应用程序组*以共享内存。每个应用程序组均有其自身分配的共享内存。在应用程序组共享内存中，每个应用程序均有其自身的应用程序控制堆，但都使用应用程序组的共享堆。

  以下三个数据库配置参数决定了应用程序组内存的大小:
  - *appgroup_mem_sz* 参数，用于指定应用程序组的共享内存的大小
  - *groupheap_ratio* 参数，用于指定允许共享堆占用应用程序组共享内存的百分比
  - *app_ctl_heap_sz* 参数，用于指定应用程序组中每个应用程序的控制堆大小。

  在性能方面，对应用程序内存进行分组的好处是提高了高速缓存和内存使用的效率。

  此外，应用程序全局内存的某些元素还可以动态地调整大小。
- **当创建代理进程时**: 图中未显示此事件。当出现连接请求或并行环境中出现新的 SQL 请求时，系统会指定一个代理进程并为其分配代理进程专用内存，其中包含仅用于该代理进程的内存分配，例如排序堆和应用程序堆。

  当数据库已被一个应用程序使用时，任何随后连接的应用程序只能分配到代理进程专用内存和应用程序全局共享内存。

此外，图中还列示了下列配置参数设置，它们用于限制为每个特定用途分配的内存大小。请注意，在分区数据库环境中，将在每个数据库分区中分配此类内存。

- *numdb*

  此参数指定不同应用程序可以使用的并发活动数据库的最大数目。由于每个数据库都有其自身的全局内存区，因此在增大此参数的值时，所分配的内存容量也可能会增加。
- *maxappls*

  此参数指定一个数据库可以同时连接的应用程序的最大数目。它将影响可能为该数据库分配的代理进程专用内存容量和应用程序全局内存容量。请注意，对于每个数据库，可将此参数设置为不同的值。
- 用于并行处理的 *maxagents* 和 *max_coordagents* 参数

图中没有显示这些参数。它们用于限制一个实例中的所有活动数据库上可以同时存在的数据库管理器代理进程的数目。与 *maxappls* 一起，这些参数可以限制为代理进程专用内存和应用程序全局内存分配的内存容量。

相关概念:

- 第 193 页的『数据库管理器共享内存』
- 第 195 页的『FCM 缓冲池和内存需求』
- 第 196 页的『全局内存和控制它的参数』
- 第 198 页的『影响内存使用率的参数的调整准则』
- 第 31 页的『内存管理』

数据库管理器共享内存

内存空间是数据库管理器运行所必需的。此空间可以很大，尤其是在分区内和分区间并行性环境中。

下图显示要使用多大内存来支持应用程序。显示的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 20. 数据库管理器如何使用内存

通过复查关于数据库代理进程的信息，可以预测和控制此空间的大小。代表应用程序运行的代理进程需要足够多的内存空间，尤其在 *maxagents* 的值设置得太大时。这是因

为根据 *maxagents* 的值为每个代理进程预先分配了一些内存。如果 *maxagents* 设置得高于需要的数目，则将预先分配一些内存，但是实际上会被数据库管理器使用，从而导致不必要的内存消耗。

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，尤其是在 *fcm_num_buffers* 的值较大时。另外，FCM 内存需求是从"FCM 缓冲池"分配的，或是从"数据库管理器共享内存"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。

相关概念:

- 第 191 页的『安排内存的使用』
- 第 196 页的『全局内存和控制它的参数』
- 第 200 页的『缓冲池管理』
- 第 198 页的『影响内存使用率的参数的调整准则』
- 第 31 页的『内存管理』

FCM 缓冲池和内存需求

如果您的分区数据库系统不具有多个逻辑节点，则"数据库管理器共享内存"和"FCM 缓冲池"如下所示。

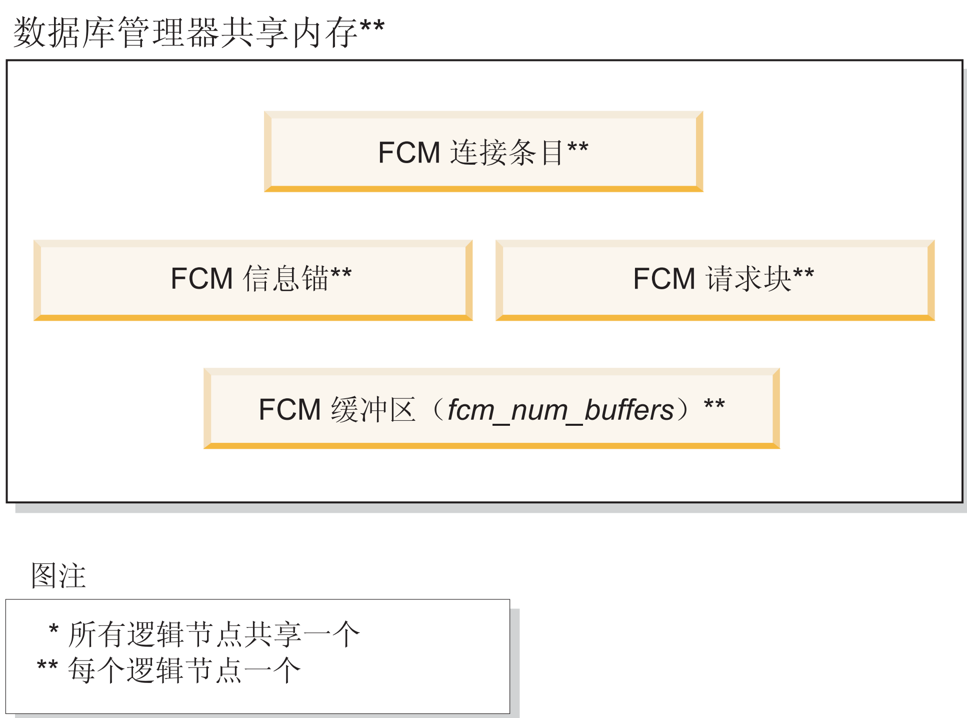


图 21. 当没有使用多个逻辑节点时的 FCM 缓冲池

如果您的分区数据库系统使用多个逻辑节点，则"数据库管理器共享内存"和"FCM 缓冲池"如下所示。

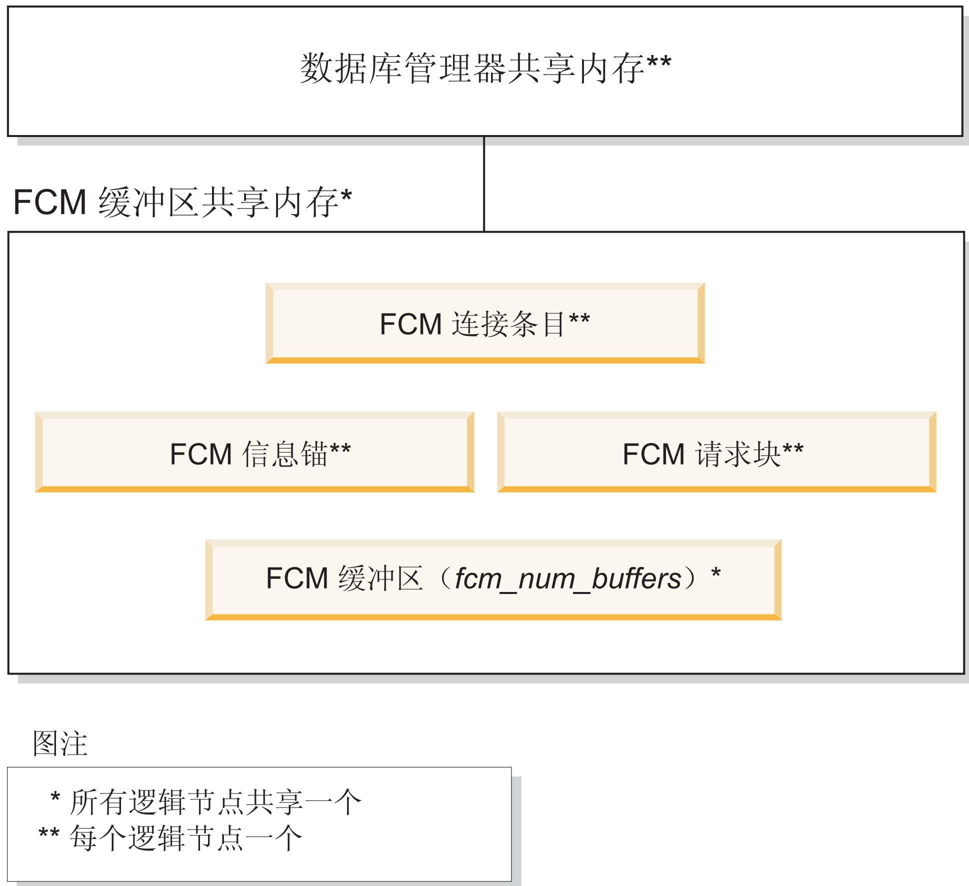


图 22. 当使用多个逻辑节点时的 FCM 缓冲池

当配置快速通信管理器（FCM）时，从“FCM 缓冲区数目”（*fcm_num_buffers*）的缺省值开始。有关 AIX® 平台上的 FCM 的更多信息，参阅 DB2_FORCE_FCP_BP 注册表变量的描述。

要调整此参数，使用数据库系统监视器来监视空闲缓冲区的低水位标记。

相关概念:

- 第 193 页的『数据库管理器共享内存』

全局内存和控制它的参数

数据库管理器共享内存由以下组件构成:

数据库全局内存

“数据库全局内存”受下列配置参数影响:

- *database_memory* 参数为数据库全局内存的大小提供了一个下限。
- 下列参数或因子指定内存段的最大大小:
  - 缓冲池的大小。
  - 锁定列表的最大存储器（*locklist*）
  - 数据库堆（*dbheap*）
  - 实用程序堆大小（*util_heap_sz*）

- 扩充存储器内存段大小（*estore_seg_sz*）
- 扩充存储器内存段数（*num_estore_segs*）
- 程序包高速缓存大小（*pckcachesz*）
- 共享排序堆（*sheapthres_shr*）

应用程序全局内存

"应用程序全局内存"受"应用程序控制堆大小"（*app_ctl_heap_sz*）配置参数的影响。

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为同一个应用程序工作的代理进程所共享。当从应用程序接收到请求的第一个代理进程请求连接时，分配这个堆。该代理进程可以是协调代理进程，也可以是子代理进程。

代理进程专用内存

- 内存段的数目受下列值中的较小者限制:
  - 所有活动的数据库的 *maxappls* 配置参数的总数，它指定活动应用程序的最大允许数目。
  - *maxagents* 配置参数的值，它指定代理进程的最大允许数目。
- 内存段的最大大小由下列参数的值确定:
  - 应用程序堆大小（*applheaps*）
  - 排序堆大小（*sortheap*）
  - 语句堆大小（*stmtheap*）
  - 统计信息堆大小（*stat_heap_sz*）
  - 查询堆大小（*query_heap_sz*）
  - 代理进程堆栈大小（*agent_stack_sz*）

代理进程／应用程序共享内存

- 本地客户机的代理进程／应用程序共享内存段的总数受下列数据库配置参数中的较低者限制:
  - 所有活动的数据库的 *maxappls* 的总数
  - *maxagents* 的值，或并行系统的 *max_coordagents* 的值。
- "代理进程／应用程序共享内存"还受下列数据库配置参数影响:
  - 应用程序支持层堆大小（*aslheapsz*）参数
  - 客户机 I/O 块大小（*rqrioblk*）参数

相关参考:

- 第 340 页的『estore_seg_sz － 扩充存储器内存段大小』
- 第 346 页的『max_coordagents － 最大协调代理进程数』
- 第 341 页的『num_estore_segs － 扩充存储器内存段数目』
- 第 324 页的『sortheap － 排序堆大小』
- 第 347 页的『maxagents － 最大代理进程数』
- 第 326 页的『aslheapsz － 应用程序支持层堆大小』
- 第 319 页的『applheapsz － 应用程序堆大小』
- 第 313 页的『pckcachesz － 程序包高速缓存大小』

- 第 310 页的『locklist – 锁定列表的最大存储器』
- 第 329 页的『rqrioblk – 客户机 I/O 块大小』
- 第 309 页的『dbheap – 数据库堆』
- 第 326 页的『stmtheap – 语句堆大小』
- 第 348 页的『maxappls – 最大活动应用程序数』
- 第 318 页的『agent_stack_sz – 代理进程堆栈大小』
- 第 322 页的『query_heap_sz – 查询堆大小』
- 第 315 页的『util_heap_sz – 实用程序堆大小』
- 第 325 页的『stat_heap_sz – 统计信息堆大小』
- 第 308 页的『database_memory – 数据库共享内存大小』

影响内存使用率的参数的调整准则

设置内存分配参数的首要原则是：除非经过慎重分析，否则切勿将这些参数设置为最大值。此原则甚至还适用于具有最大内存的系统。许多影响内存的参数可使数据库管理器能够方便快捷地占用计算机上的所有可用内存。此外，在管理大量内存时，可能给数据库管理器部件带来额外的工作，因而产生更多开销。

某些 UNIX® 操作系统在一个进程分配内存时（而不是在进程导出页至交换空间时）分配交换空间。对于这些系统，请确保为其提供与总共享内存空间相同的调页空间。

对于大多数配置参数，仅在需要时才会落实内存。这些参数决定特定内存堆的最大空间。但是，在下列情况下，将分配参数指定的全部内存容量:

- 锁定列表的最大存储器（*locklist*）
- 应用程序支持层堆大小（*aslheapsz*）
- FCM 缓冲器的数量（*fcm_num_buffers*）

注: 要更改缓冲池的大小，使用 DDL 语句 ALTER BUFFERPOOL。

影响应用程序组使用内存的参数

影响应用程序组使用内存的参数仅应用于分区数据库、已为其启用内部并行处理的数据库以及已为其启用连接集中器的数据库。下列参数决定应用程序组中的应用程序如何使用它们的共享内存;

- *appgroup_mem_sz* 参数指定应用程序组的共享内存大小。

  将 *appgroup_mem_sz* 配置参数的值设置太高会导致相反的效果。由于应用程序组中的所有应用程序共享应用程序组堆中的高速缓存，因此设置太多的应用程序会增大高速缓存的争用程度。另一方面，如果每个应用程序组包含太少的应用程序，高速缓存的效果也将有限。
- *groupheap_ratio* 参数指定允许共享堆使用的内存百分比。

  将 *groupheap_ratio* 的值设置太低会限制高速缓存的大小。将 *groupheap_ratio* 的值设置太高会造成应用程序控制堆太小而可能导致 SQL 错误 SQL0973，此错误会警告您在运行时用尽了应用程序控制堆内存。
- *app_ctl_heap_sz* 参数指定应用程序组中每个应用程序的控制堆大小。

在配置数据库服务器时，请接受这些参数的缺省设置。只有在性能受到损害时才有必要调整缺省设置。例如，设置 *appgroup_mem_sz* 以控制每个应用程序组中的应用程序数量。一般而言，数量为 10 时太少而为 100 时太多。缺省值可能是一个适当的值。运行平均的工作负载，然后使用控制中心中的健康中心实用程序或系统监视程序来收集有关目录高速缓存、程序包高速缓存和共享工作空间的命中率信息。

- 如果发生许多 `sql0973` 错误，则说明 *groupheap_ratio* 的设置值太高。
- 如果内存跟踪程序显示应用程序控制堆的平均和最大使用量远远低于 `app_ctl_heap_sz * (100 - groupheap_ratio)/ 100`，则应减小 *app_ctl_heap_sz* 配置参数的值。
- 如果高速缓存使用量指示高速缓存达到其限制值，则应增大 *group_heap_ratio* 配置参数的值或减少应用程序组中的应用程序数量。

注:

- 基准程序测试可以提供有关为内存参数设置适当值的最佳信息。在基准程序测试中，将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。如果用图形表示性能与参数值的比率，则曲线开始到达水平稳定期或开始下降的那一点将指示在此点上分配附加内存并不会增加应用程序性能的值，因此只是浪费内存而已。
- 几个参数的内存分配上限可能超出了现有硬件和操作系统的内存容量。这些限制允许用于将来的内存增长。
- 要了解参数的有效范围，请参阅每个参数的详细信息。

相关概念:

- 第 191 页的『安排内存的使用』
- 第 193 页的『数据库管理器共享内存』
- 第 196 页的『全局内存和控制它的参数』

相关参考:

- 第 315 页的『app_ctl_heap_sz – 应用程序控制堆大小』
- 第 408 页的『fcm_num_buffers – FCM 缓冲区数目』
- 第 323 页的『sheapthres – 排序堆阈值』
- 第 326 页的『aslheapsz – 应用程序支持层堆大小』
- 第 313 页的『pckcachesz – 程序包高速缓存大小』
- 第 310 页的『locklist – 锁定列表的最大存储器』
- 第 309 页的『dbheap – 数据库堆』
- 第 315 页的『util_heap_sz – 实用程序堆大小』
- 第 307 页的『catalogcache_sz – 目录高速缓存大小』
- 第 316 页的『appgroup_mem_sz – 应用程序组内存集的最大大小』
- 第 317 页的『groupheap_ratio – 应用程序组堆的内存百分比』

缓冲池

缓冲池是非常重要的内存组件。本节描述缓冲池并提供关于管理它们以获取良好性能的信息。

缓冲池管理

缓冲池是指从磁盘中读取表或索引数据页或对它们进行修改时，用于对这些表或索引数据页进行高速缓存的内存。缓冲池通过允许从内存（而不是磁盘）中读取数据，来改善数据库系统的性能。由于内存的存取速度比磁盘的存取速度快得多，因此，数据库管理器需要读写磁盘的次数越少，性能就越好。由于大多数数据处理发生在缓冲池内，因此配置缓冲池是唯一的最为重要的调整环节。只有大对象和长字段数据不在缓冲池中处理。

当一个应用程序第一次存取某个表的某行时，数据库管理器会将包含该行的页放入缓冲池中。下一次任何应用程序请求数据时，数据库管理器会先在缓冲池中查找此数据。如果请求的数据在缓冲池中，则无需存取磁盘即可检索到此数据，因而提高了系统性能。

当激活数据库时或第一个应用程序与该数据库连接时，就会为缓冲池分配内存。此外，在运行数据库管理器时，还可以创建和删除缓冲池以及调整缓冲池的大小。如果使用 ALTER BUFFERPOOL 来增大缓冲池的大小时使用 IMMEDIATE 关键字，则一输入命令就分配内存（如果内存可用）。如果内存不可用，则当所有应用程序都断开连接并且重新激活数据库时发生更改。如果要减少缓冲池的大小，则在落实时释放内存。当所有应用程序都断开连接时，会释放缓冲池内存。

注: 为了避免因增大缓冲池的大小而必须增大 *dbheap* 数据库配置参数的大小，几乎所有缓冲池内存（包括页描述符、缓冲池描述符和散列表）均由数据库共享内存集产生，并且可以自动调整大小。

为了确保适当的缓冲池在所有情况下均可用，DB2® 创建了许多小型缓冲池，它们的每个页的大小为：4K、8K、16K 和 32K。每个缓冲池的大小为 16 页。用户看不到这些缓冲池。它们不会出现在系统目录或缓冲池系统文件中。您不能直接使用或改变这些缓冲池，但 DB2 在下列情况下使用它们:

- 所需页大小的缓冲池由于以下原因变得不活动时：在用 IMMEDIATE 关键字执行 CREATE BUFFERPOOL 语句后因内存不足而无法创建缓冲池。

  将一条消息写入到管理通知日志。如果必要，将表空间重新映射到隐藏的缓冲池。性能可能明显下降。
- 在数据库连接期间无法启动普通的缓冲池

  此问题通常由严重的原因（如内存不够条件）引起。虽然 DB2 将因为隐藏的缓冲池充分发挥其功能，但性能会明显下降。您应立即解决此问题。当发生此问题时，您会接收到一条警告，并且系统将一条消息写入管理通知日志。

这些页一直保存在缓冲池中，直到关闭数据库，或直到其它页需要使用某一页所占用的空间。以下标准用于确定要除去哪一页以引入其它页:

- 是多久前引用的页
- 查看过该页的最后一个代理进程再次引用该页的可能性
- 页中数据的类型
- 该页是否已在内存中更改但尚未写入磁盘（更改的页始终在被覆盖之前写入磁盘）。

为了可以再次从内存中存取某些页，除非其它页需要这些页所占用的空间，否则不会从缓冲池中除去它们。

当创建缓冲池时，缺省页大小为 4 KB，但您可以将页大小指定为 4 KB、8 KB、16 KB 或 32 KB。由于仅在表空间页大小与缓冲池页大小相同时才可以将页读入缓冲池，因此您为缓冲池指定的页大小应由表空间的页大小确定。您不能在创建缓冲池之后改变它的页大小。必须用不同的页大小创建新的缓冲池。

注: 在运行 Windows® NT 的 32 位平台上，如果已启用 Windows 2000 上的地址窗口扩展（AWE）或“高级服务器”和“数据中心服务器”，则可以创建大型缓冲池。

相关概念:

- 第 191 页的『安排内存的使用』
- 第 201 页的『在 32 位平台上扩展内存中的辅助缓冲池』
- 第 202 页的『数据页的缓冲池管理』
- 第 204 页的『缓冲池数据页管理的示例』
- 第 205 页的『多个数据库缓冲池的管理』

在 32 位平台上扩展内存中的辅助缓冲池

在 64 位平台上，可以正常方式存取大的可寻址虚拟内存，而不必使用特殊技术。但是，在 32 位平台上，可寻址虚拟内存通常限制在 2 GB 和 4 GB 之间。如果您的 32 位机器的可寻址实内存超过最大数量时，您可将可寻址虚拟内存外的任何附加可寻址实内存配置为*扩充存储器高速缓存*。任何已定义的缓冲池能使用扩充存储器高速缓存来提高性能。应将扩充存储器高速缓存定义为许多内存段。

如果将一些可寻址实内存定义为扩充存储器高速缓存，此内存不再能够用于其它用途，如 JFS 高速缓存或进程专用地址空间。如果您将可寻址实内存分配到扩充存储器高速缓存，可能会发生更多的系统页面调度。

缓冲池执行第一级高速缓存，并且缓冲池将任何扩充存储器高速缓存用作次级高速缓存。理论上，缓冲池保存那些最频繁存取的数据，而扩充存储器高速缓存保存那些较不频繁存取的数据。

注: 可以使用 DB2_AWE 注册表变量分配 Windows® 2000“地址窗口扩展”（AWE）缓冲池。Windows AWE 是一组内存管理扩展，它允许应用程序在某些限制以上处理内存，该限制取决于应用程序的进程模型。有关信息，参阅 Windows 系统文档。但是，请注意，如果将内存用于此目的，您也不能使用扩充存储器高速缓存。

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* 定义扩充存储器内存段的数目。此配置参数的缺省值为零，它指定不存在扩充存储器高速缓存。
- *estore_seg_sz* 定义每个扩展内存段的大小。此大小由使用扩充存储器高速缓存的平台确定。

因为扩充存储器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声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器。

注: 如果使用用不同页大小定义的缓冲池，可以定义其中的任何缓冲池来使用扩充存储器。使用扩充存储器支持的页大小是已定义的最大页大小。

虽然数据库管理器不能直接处理驻留在扩充存储器高速缓存中的数据，但是它可以采用比从磁盘存储器传送高得多的速度将数据从扩充存储器高速缓存传送至缓冲池。

当需要从一个扩充存储器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。

当激活数据库或第一次进行连接时，会分配一个缓冲池及其定义的关联扩充存储器高速缓存。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 31 页的『内存管理』

相关参考:

- 第 340 页的『estore_seg_sz － 扩充存储器内存段大小』
- 第 341 页的『num_estore_segs － 扩充存储器内存段数目』

数据页的缓冲池管理

缓冲池中的页可能*正在使用*，也可能没有使用，它们可能是*脏*页，也可能是*干净*页:

- 正在使用的页就是当前正在读取或更新的页。当代理进程正在使用一个页时，数据库中的其它代理进程或预取程序可以读取该页，但是不能更新该页。
- “脏”页包含已更改但尚未写入磁盘的数据。
- 在将一个已更改的页写入磁盘后，它是“干净的”，但它仍然留在缓冲池中，直到新的页需要它的空间为止。也可以将干净页迁移到关联的扩充存储器高速缓存（如果定义了一个的话）。

页清除程序代理进程

在一个经过认真调整的系统中，通常是由页清除程序代理进程将已更改的页或者“脏”页写入磁盘中。页清除程序代理进程作为后台进程执行 I/O 并允许应用程序快速运行，因为其代理进程可以执行实际事务工作。页清除程序的代理进程有时称为*异步页清除程序*或*异步缓冲区写程序*，因为它们不与其它代理进程的工作协调进行，而仅在需要时工作。

为了提高具有大量更新的工作负载的性能，您可能想配置更多页清除程序代理进程。如果提供了更多页清除程序代理进程来将脏页写入磁盘，则性能会提高。当快照揭示有许多同步数据页或索引页写操作与异步数据页或索引页写操作相关时尤其是如此。

页清除和快速恢复

如果已将多页写入磁盘，则在系统崩溃后数据库的恢复更快，因为数据库管理器可以从磁盘重新构建更多的缓冲池，而不必从数据库日志文件回放事务。

恢复期间必须读取的日志的大小是日志中下列记录的位置之间的差值:

- 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。

页清除程序的缺省行为是：如果在恢复期间需要重放的日志的大小超过了下面的最大值，则执行页清除：

```
logfilsiz * softmax
```

其中：

- *logfilsiz* 表示日志文件的大小
- *softmax* 表示在数据库崩溃后要恢复的日志文件的百分比。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。

要将恢复期间日志读取时间减至最小，使用数据库系统监视器来跟踪执行页清除的次数。如果尚未对数据库启用主动清除页，则系统监视器 *pool_lsn_gap_clns*（*触发的缓冲池日志空间清除程序*）监视器元素将提供此信息。如果已经启用了此备用页清除，则不应该发生这种情况，并且 *pool_lsn_gap_clns* 监视器元素始终为 0。

可以使用 *log_held_by_dirty_pages* 监视器元素来确定是否页清除程序未清除足够多的页数以满足用户设置的恢复条件。如果 *log_held_by_dirty_pages* 始终明显大于 *logfilsiz* * *softmax*，则要么需要提供更多页清除程序，要么需要调整 *softmax*。

相关概念:

- 第 204 页的『缓冲池数据页管理的示例』

相关参考:

- 第 469 页的『性能变量』

主动的页清除

从版本 8.1.4 开始，可以使用另一种方法来配置系统中的页清除。这种方法与缺省行为的差别在于，在选择任何给定时刻要写出哪些脏页方面，页清除程序表现得更为主动。这种新的页清除方法与缺省页清除方法主要在以下两方面不同:

1. 页清除程序不考虑 chngpgs_thresh 配置参数。

   当使用这种页清除方法时，页清除程序将不再对 chngpgs_thresh 配置参数的值作出反应。这种页清除方法不会尝试使一定百分比的缓冲池保持为干净的，这种页清除方法提供了一种机制，在这种机制中，将向代理进程通知刚写出的良好干扰页的位置，因此，代理进程不必通过搜索缓冲池来查找干扰页。当良好干扰页的数目减少到低于可接受的值时，就会触发页清除程序，它会继续搜索整个缓冲池，写出潜在的干扰页，并通知代理进程这些页的位置。

2. 页清除程序不再响应记录器发出的 LSN 间隔触发器。

   如果日志空间的数量中包含更新了缓冲池中的最旧页的日志记录，并且当前日志位置超过了 softmax 参数所允许的值，就认为数据库处于“LSN 间隔”状态。当使用缺省的页清除方法时，如果记录器检测到产生 LSN 间隔，则它将触发页清除程序来写出导致 LSN 间隔状态的所有页。也就是说，它将写出比 softmax 参数允许的页更早的页。在没有产生 LSN 间隔的时间段内，页清除程序将处于空闲状态。但是，一旦产生 LSN 间隔，就会激活页清除程序来写出大量页，然后它又会返回到休眠状态。这将使 I/O 子系统达到饱和，这又会影响正在对页进行读写操作的其它代理进程。而且，当触发 LSN 间隔时，页清除程序可能不能够快速进行清除，并且 DB2® 可能用完了日志空间。

另一种页清除方法通过在更长的时间段内进行相同次数的写操作来调整此行为。清除程序通过根据活动的当前级别不但主动清除当前处于 LSN 间隔状态的页，而且清除不久将进入 LSN 间隔状态的页来完成此调整。

要使用新的页清除方法，应将 DB2_USE_ALTERNATE_PAGE_CLEANING 注册表变量设置为 “ON”。

相关概念:

- 第 202 页的『数据页的缓冲池管理』

缓冲池数据页管理的示例

下图举例说明了与执行所有 I/O 的数据库代理进程比较，页清除程序代理进程和数据库代理进程如何共享管理缓冲池的工作。

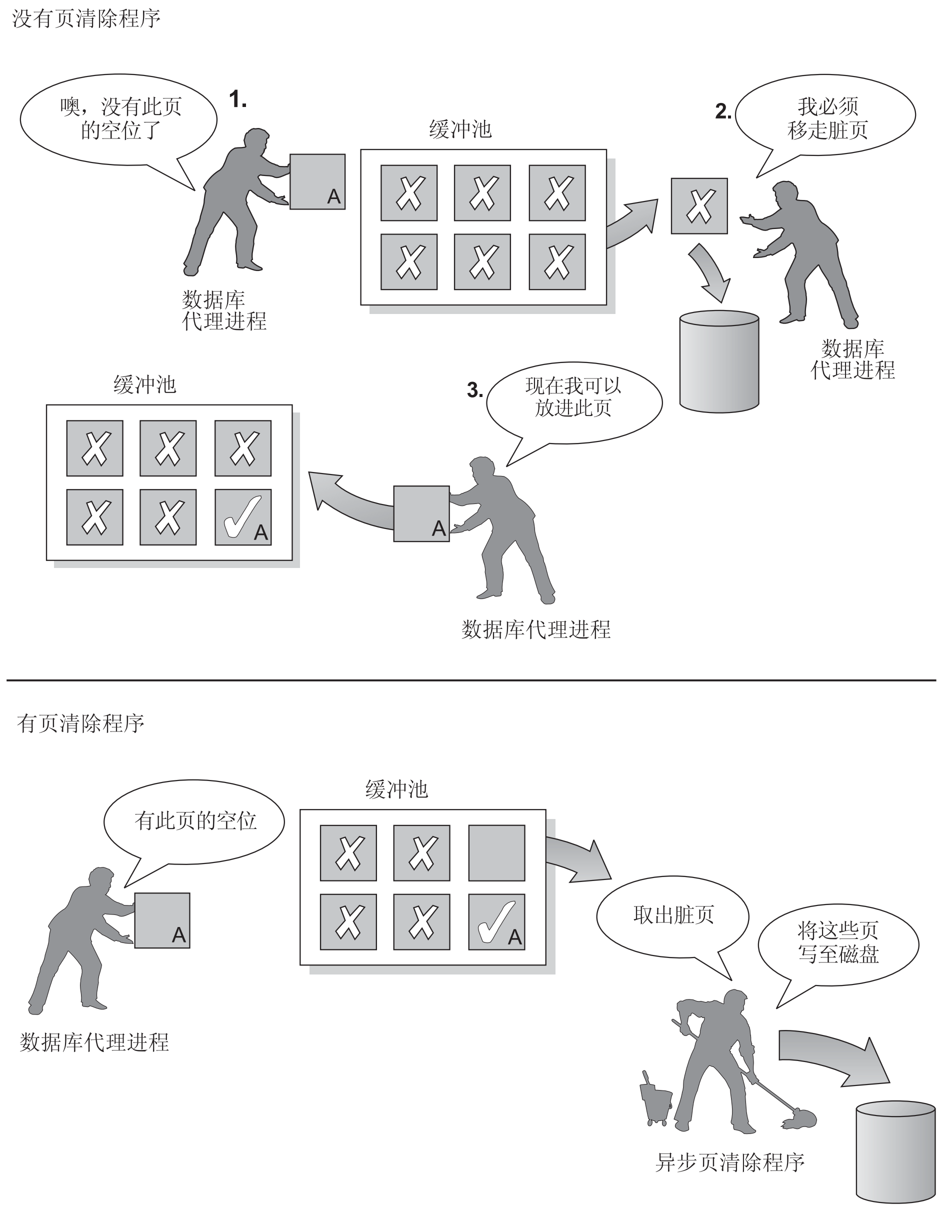


图 23. 异步页清除程序. 把“脏”页写出至磁盘。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 202 页的『数据页的缓冲池管理』

多个数据库缓冲池的管理

尽管每个数据库至少需要一个缓冲池，但可以为具有多个页大小的表空间的单个数据库创建几个缓冲池，每个缓冲池具有不同的大小或具有不同的页大小。每个缓冲池具有最小大小，这取决于平台。

新数据库具有缺省缓冲池 IBMDEFAULTBP，它的大小是由平台确定的且缺省页大小为 4 KB。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。可以对缺省缓冲池调整大小以及更改其属性，但不能删除它。

注: 在正常的数据库管理器操作期间，可以使用 ALTER BUFFERPOOL 命令来对缓冲池调整大小。

缓冲池的页大小

创建或迁移数据库之后，可以创建其它缓冲池。例如，当规划数据库时，可能已确定 8 KB 页大小对于表是最佳的。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具有相同页大小的表空间。不能使用 ALTER TABLESPACE 语句将表空间指定给使用另一种页大小的缓冲池。

注: 如果创建页大小大于 4 KB（如 8 KB、16 KB 或 32 KB）的表空间，需要给它分配使用相同页大小的缓冲池。如果此缓冲池当前不活动，DB2® 尝试临时将该表空间指定给另一个使用相同页大小的活动缓冲池（如果存在第一次将客户机连接至数据库时 DB2 创建的其中一个或至其中一个缺省“隐藏”缓冲池的话）。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。

当创建缓冲池时，应指定缓冲池的大小作为 DDL 语句 CREATE BUFFERPOOL 的必需参数。稍后要增大或减小缓冲池的大小，使用 DDL 语句 ALTER BUFFERPOOL。

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 ALTER BUFFERPOOL 语句为特定数据库分区更改了缓冲池大小。

大缓冲池的优点

大缓冲池提供下列优点:

- 使频繁请求的数据页能够保存在缓冲池中，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。

许多缓冲池的优点

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。

在所有其它情况下，由于下列原因而考虑使用多个缓冲池:

- 可以把临时表空间分配给单个缓冲池中，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。
- 如果数据必须由很多小的更新事务应用程序反复地快速存取，考虑将包含该数据的表空间指定给单独的缓冲池。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不同的缓冲池中，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。此更改将减少对第一组表的频繁更新对第二组表的频繁查询具有的影响。
- 对于很少使用的应用程序所存取的数据，可使用较小的缓冲池，尤其对于需要对一个很大的表非常随机地进行存取的应用程序。在这种情况下，不需要将数据保存在缓冲池中的时间超过单个查询的保存时间。最好为此数据保留一个小缓冲池，而释放多余的内存用于其它用途，如用于其它缓冲池。
- 当将不同的活动和数据隔离到不同的缓冲池之后，可从统计信息和记帐跟踪获得一个良好的、相对成本较低的性能诊断数据。

启动时缓冲池内存分配

当使用 CREATE BUFFERPOOL 命令来创建缓冲池或使用 ALTER BUFFERPOOL 命令来变更缓冲池时，所有缓冲池需要的全部内存必须可供数据库管理器使用，以便在启动该数据库时可分配所有缓冲池。如果在数据库管理器联机时创建或修改缓冲池，则附加内存应在数据库全局内存中可用。如果在创建新缓冲池或增大现有缓冲池的大小时指定 IMMEDIATE 关键字，但所需内存不可用，则数据库管理器在下次激活数据库时进行更改。在 32 位平台上，内存必须可用且可以保留在全局数据库内存中，如 *database_memory* 数据库配置参数中的详细信息中所述。

如果启动数据库时此内存不可用，数据库管理器将尝试启动用另一种页大小定义的一个缓冲池。但是，只能启动每个为最小大小 16 页的缓冲池。要指定另一个最小缓冲池大小，使用 DB2_OVERRIDE_BPF 注册表变量。当不能在启动时分配缓冲池时，返回 SQL1478W（SQLSTATE 01626）警告。数据库会继续保持在此运作状态，直到更改其配置且数据库可以完全重新启动为止。

数据库管理器仅使用最小大小值启动来允许您连接至数据库，以便可以重新配置缓冲池大小或执行其它关键任务。只要执行这些任务，就重新启动数据库。不要在这种状态下超长时间运行数据库。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 201 页的『在 32 位平台上扩展内存中的辅助缓冲池』

预取概念

将数据预取到缓冲池通常会提高性能，因为它减少磁盘存取次数并将经常存取的数据保留在内存中。

将数据预取至缓冲池

*预取*页就是在应用程序将需要一个或多个页之前，提前从磁盘中检索那些页。将索引和数据页预取至缓冲池，可以通过减少 I/O 等待时间帮助提高性能。此外，并行 I/O 增强预取效率。

有两种类别的预取:

- **顺序预取:** 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机制。
- **列表预取:** 有时称为*列表顺序预取*。有效预取一组非连续的数据页。

这两种读取数据页的方法是对常规读取方法的补充。当仅检索一页或少量连续页时，使用常规读取方法。在常规读取期间，只传送一页数据。

预取和分区内并行性

预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进程。这些并行扫描产生更大的数据耗用速率，并需要更高的预取速率。

对于少量预取的成本，并行扫描比串行扫描更高。如果对于一个串行扫描未进行预取，则由于代理进程始终需要等待 I/O，所以该查询运行起来更慢。如果对于一个并行扫描未进行预取，则所有子代理进程可能需要等待，因为一个子代理进程在等待 I/O。

由于其重要性，使用分区内并行性可以更主动地执行预取。顺序检测机制容许相邻页之间有更大的间隔，以便可以将这些页视为顺序页。这些间隔的宽度随着扫描中涉及的子代理进程数目的增加而增加。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 208 页的『顺序预取』
- 第 210 页的『列表预取』
- 第 211 页的『预取和并行性的 I/O 服务器配置』
- 第 212 页的『使用并行 I/O 预取示例』

顺序预取

使用单一 I/O 操作将几个连续的页读到缓冲池中，可以大大减少应用程序开销。此外，将几个范围内的页读到缓冲池中的多个并行 I/O 操作可以有助于减少 I/O 等待时间。

当数据库管理器确定顺序 I/O 是合适的且预取可改善性能时，预取启动。对于象表扫描和表排序这类情况，数据库管理器可以轻易地确定顺序预取将有利于提高 I/O 性能。在这些情况下，数据库管理器会自动启动顺序预取。以下示例（可能需要表扫描）将适合于顺序预取:

```
SELECT NAME FROM EMPLOYEE
```

表空间的 PREFETCHSIZE 的含义

要为每个表空间定义预取的页数，在 CREATE TABLESPACE 或 ALTER TABLESPACE 语句中使用 PREFETCHSIZE 子句。在 SYSCAT.TABLESPACES 系统目录表的 PREFETCHSIZE 列中维护指定的值。

一个好方法是将 PREFETCHSIZE 值显式设置为表空间容器数、每个容器下的物理磁盘数（如果使用了 RAID 设备的话）与您的表空间的 EXTENTSIZE 值的乘积，该值是在数据库管理器使用另一个容器之前写入容器的页数。例如，如果扩展数据块大小为 16 页，且该表空间有两个容器，则您可以将预取量设置为 32 页。如果每个容器具有 5 个物理磁盘，则可以将预取数量设置为 160 页。

数据库管理器监视缓冲池的使用情况，以确保在另一个工作单元需要这些页的情况下，预取不会从缓冲池中除去这些页。为避免引起问题，数据库管理器可将预取的页数限制到小于您为表空间所指定的数。

预取大小可以显著影响性能，特别是对大表扫描更是如此。使用数据库系统监视器和其它系统监视工具来帮助您调整表空间的 PREFETCHSIZE。您可以收集有关是否为如下情况的信息:

- 有 I/O 等待您的查询，这可使用您的操作系统可用的监视工具来获知。
- 正在进行预取，可查看数据库系统监视器提供的 *pool_async_data_reads*（*缓冲池异步数据读取*）数据元素来获知。

如果有 I/O 等待并且该查询是预取数据，您可增大 PREFETCHSIZE 的值。如果预取程序不是 I/O 等待的原因，增大 PREFETCHSIZE 值将不会提高查询的性能。

在所有预取类型中，当预取大小是表空间的扩展数据块大小的倍数，并且该表空间的扩展数据块是在不同的容器中时，可以并行执行多个 I/O 操作。为获取更好性能，配置容器以使用独立的物理设备。

顺序检测

在某些情况下，并不能立即确定顺序预取是否将提高性能。在这类情况下，数据库管理器可以监视 I/O，且如果正在读取顺序页，则数据库管理器可以激活预取。在这种情况下，数据库管理器在适当时激活和取消激活预取。这种类型的顺序预取称为*顺序检测*，并且同时适用于索引页和数据页。使用 *seqdetect* 配置参数来控制数据库管理器是否执行顺序检测。

例如，如果打开顺序检测，则下列 SQL 语句可能从顺序预取受益:

```
SELECT NAME FROM EMPLOYEE   WHERE EMPNO BETWEEN 100 AND 3000
```

在此示例中，优化器可能已经启动使用 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，并且预取也可提高性能，因此将会进行数据页预取。

索引页预取也可能发生在此示例中。如果必须检查许多索引页，并且数据库管理器检测到正在进行索引页的顺序页读取，则将进行索引页预取。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 207 页的『将数据预取至缓冲池』
- 第 210 页的『列表预取』
- 第 209 页的『改进的顺序预取的基于块的缓冲池』

改进的顺序预取的基于块的缓冲池

从磁盘预取页由于 I/O 开销而使成本很高。如果处理与 I/O 交迭，则可以显著提高吞吐量。大多数平台提供高性能的将连续页从磁盘读到内存的非连续部分的原语。这些原语通常称为*散射读取*或*向量化 I/O*。在某些平台上，这些原语的性能不能与在大块大小中执行 I/O 相比。

缺省情况下，缓冲池基于页，这指的是将磁盘上的连续页预取到内存非连续的页。如果可以将连续页从磁盘读到缓冲池中的连续页中，则可以增强顺序预取。

可以为此目的创建基于块的缓冲池。基于块的缓冲池由页区域和块区域组成。页区域对于非顺序预取工作负载是必需的。块区域由块组成，其中每个块包含指定数量的连续页，它称为*块大小*。

基于块的缓冲池的最佳使用情况取决于指定的块大小。块大小是执行顺序预取的 I/O 服务器考虑按其执行基于块的 I/O 的粒度。扩展数据块是在容器中对表空间进行条带分割的粒度。因为可以将具有不同扩展数据块大小的多个表空间绑定到用同一块大小定义的缓冲池，所以考虑扩展数据块大小如何与块大小交互以便有效使用缓冲池内存。在下列情况下可能浪费缓冲池内存:

- 如果扩展数据块大小（它确定预取请求大小）小于为缓冲池指定的 BLOCK_SIZE。
- 如果在预取请求中请求的某些页已存在于缓冲池的页区域中。

I/O 服务器允许每个缓冲池块中的某些浪费的页，但是，如果将浪费过多的块，则 I/O 服务器执行非基于块的预取到缓冲池的页区域中。这不是最佳性能。

为了获取最佳性能，将相同扩展数据块大小的表空间绑定到块大小等于表空间扩展数据块大小的缓冲池。如果扩展数据块大小大于块大小，但不是在扩展数据块大小小于块大小时，可以达到良好性能。

要创建基于块的缓冲池，使用 CREATE 和 ALTER BUFFERPOOL 语句。基于块的缓冲池具有下列限制:

- 不能同时基于块和使用扩充存储器来建立缓冲池。
- 基于块的 I/O 和 AWE 支持不能由缓冲池同时使用。当对于给定的缓冲池同时启用了 AWE 支持和基于块的 I/O 支持时，AWE 支持优先于基于块的 I/O 支持。在此情况下，对该缓冲池，禁用基于块的 I/O 支持。当禁用 AWE 支持，不重新启用基于块的 I/O。

注: 基于块的缓冲池将用于顺序预取。如果应用程序不使用顺序预取，则会浪费缓冲池的块区域。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 207 页的『将数据预取至缓冲池』
- 第 208 页的『顺序预取』

列表预取

*列表预取*或*列表顺序预取*是一种有效存取数据页的方法，甚至在所需的数据页并不连续时，也是如此。列表预取可以与单个或多个索引存取结合使用。

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行标识（RID）之后进行。例如，优化器可执行索引扫描来确定要检索的行和数据页，并给予先前定义的索引 IX1:

```
INDEX IX1:  NAME    ASC,
            DEPT    ASC,
            MGR     DESC,
            SALARY  DESC,
            YEARS   ASC
```

和下列搜索条件:

```
WHERE NAME BETWEEN 'A' and 'I'
```

如果数据未根据此索引群集，则列表预取包括一个把从索引扫描获得的 RID 的列表排序的步骤。

相关概念:

- 第 200 页的『缓冲池管理』
- 第 207 页的『将数据预取至缓冲池』
- 第 208 页的『顺序预取』

I/O 管理

本节描述如何调整 I/O 服务器以获取最佳性能。

预取和并行性的 I/O 服务器配置

要启用预取，数据库管理器需要启动不同的控制线程（称为 *I/O 服务器*）来读取数据页。因此，查询处理分为两个并行活动：数据处理（CPU）和数据页 I/O。I/O 服务器等待从 CPU 处理活动发来的预取请求。这些预取请求中包含有关满足查询所需的 I/O 的描述。可能的预取方法决定数据库管理器何时及如何生成预取请求。

使用配置参数 *num_ioservers* 配置足够的 I/O 服务器，可以大大提高可使用数据预取的查询的性能。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。

高估 I/O 服务器的数目要好于低估 I/O 服务器的数目。如果指定了多余的 I/O 服务器，虽然这些服务器得不到使用，但其内存页会被调出。因此，性能不会受到影响。每个 I/O 服务器进程均有编号。数据库管理器始终使用编号最低的进程，因此某些编号较高的进程可能永远得不到使用。

要确定所需的 I/O 服务器数目，请考虑下列因素:

- 可以同时将预取请求写入 I/O 服务器队列的数据库代理进程的数目。
- I/O 服务器可并行工作的最大程度。

异步 I/O 的配置

在某些平台上，DB2® 使用异步 I/O（AIO）来提高活动（例如页清除和页预取）的性能。如果将容器中的数据分配到多个磁盘上，则 AIO 是最有效的方法。此外，通过调整基本操作系统的 AIO 基础结构，也可以提高性能。

例如，在 AIX® 上，可能需要调整操作系统中的 AIO。当 AIO 对 SMS 或 DMS 文件容器工作时，称为 AIO 服务器的操作系统进程将会管理 I/O。如果此类服务器的数量太少，则 AIO 的优点可能会因 AIO 请求数太少而得不到充分的体现。要在 AIX 上配置 AIO 服务器的数目，请使用 *smit* AIO *minservers* 和 *maxservers* 参数。

相关概念:

- 第 80 页的『应用程序的并行处理』
- 第 212 页的『使用并行 I/O 预取示例』
- 第 213 页的『并行 I/O 管理』

相关参考:

- 第 342 页的『num_ioservers – I/O 服务器数目』

使用并行 I/O 预取示例

下图举例说明如何使用 I/O 服务器来将数据预取到缓冲池中。

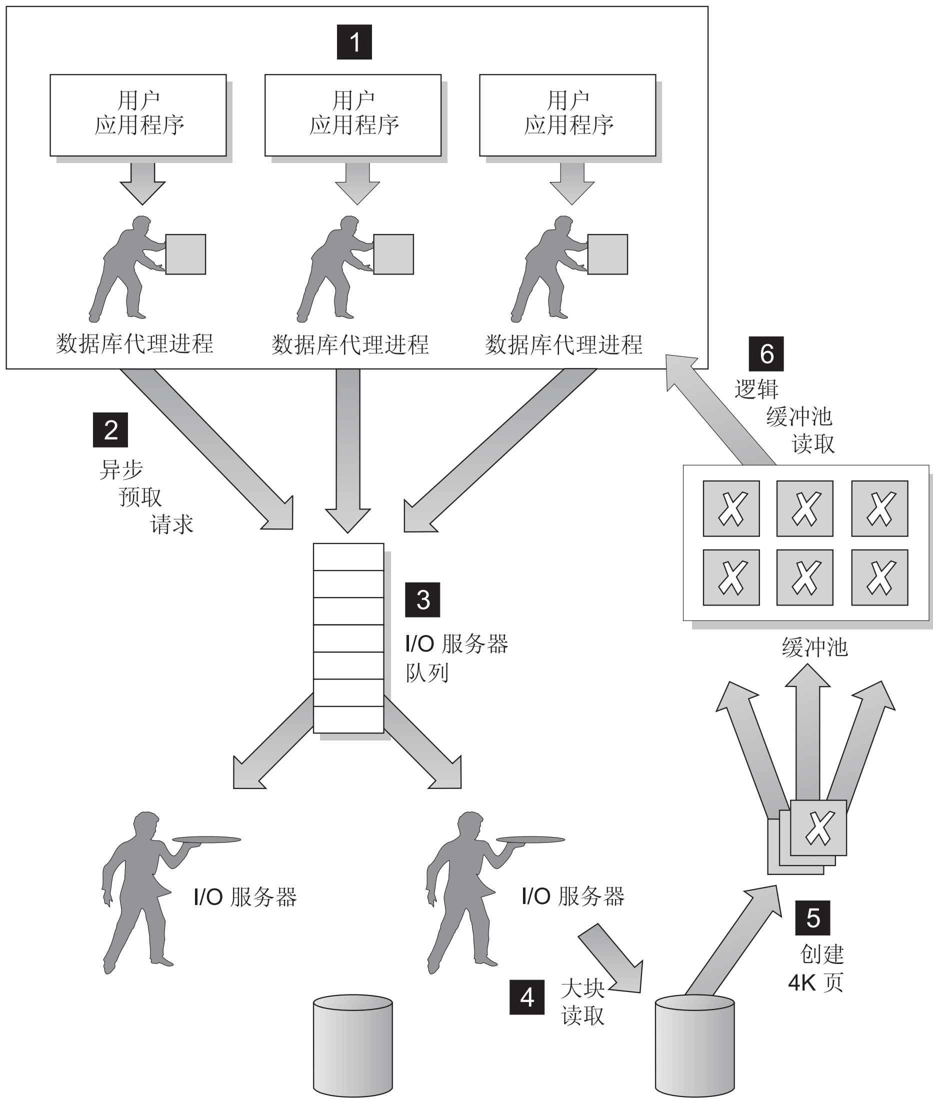


图 24. 使用 I/O 服务器预取数据

1 　用户应用程序将 SQL 请求传送到数据库代理进程，该代理进程已由数据库管理器指定给用户应用程序。

2，**3**

数据库代理进程确定应使用预取来获取满足该 SQL 请求所需的数据，并向 I/O 服务器队列写入一个预取请求。

4，**5**

第一个可用的 I/O 服务器从这个队列读取预取请求，然后从表空间将数据读入

缓冲池。可以同时从表空间访存数据的 I/O 服务器的数目取决于该队列中预取请求的数目以及 *num_ioservers* 数据库配置参数配置的 I/O 服务器的数目。

6 数据库代理进程对缓冲池中的数据页执行必要的操作并将结果返回到用户应用程序。

相关概念:

- 第 207 页的『将数据预取至缓冲池』
- 第 208 页的『顺序预取』
- 第 210 页的『列表预取』
- 第 211 页的『预取和并行性的 I/O 服务器配置』
- 第 213 页的『并行 I/O 管理』
- 第 237 页的『分区数据库中的代理进程』

并行 I/O 管理

如果在表空间存在多个容器，数据库管理器可以启动*并行 I/O*，其中，数据库管理器使用多个 I/O 服务器处理单个查询的 I/O 需求。每个 I/O 服务器处理单个容器的 I/O 工作负载，因此可以并行方式读取几个容器。以并行方式执行 I/O 可以显著提高 I/O 吞吐量。

尽管单个 I/O 服务器可以处理每个容器的工作负载，但是，可以并行方式执行 I/O 的 I/O 服务器的实际数目限制为请求的数据所分布的物理设备的数目。因此，您需要与物理设备同样多的 I/O 服务器。

在下列情况下，以不同方式启动并行 I/O:

- **顺序预取**

  对于顺序预取，当预取大小是一个表空间的扩展数据块大小的倍数时，启动并行 I/O。然后，沿扩展数据块边界将每个预取请求分为许多小的请求。然后将这些小的请求再分配给不同的 I/O 服务器。

- **列表预取**

  对于列表预取，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扩展数据块大小所得的商，最大值可达到容器数。

- **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。如果未指定此选项，则数据库管理器使用缺省值，该值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。将 I/O 服务器的数目指定为稍大于用于数据库内所有表空间的容器的数目。
- 表空间的扩展数据块大小和预取大小是合理的。要防止过分使用缓冲池，预取大小不应太大。理想大小是扩展数据块大小、每个容器下的物理磁盘数（如果使用了 RAID 设备的话）与表空间容器数的乘积。扩展数据块大小应该稍小些，合适的值在 8 至 32 页之间。
- 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它容器小，降低优化并行预取的可能性。考虑下列示例:

  – 在填充一个较小的容器后，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取的性能，因为可从中预取数据的容器数目可能小于容器的总数。
  – 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它容器。相对于其它容器，此容器的少量数据将不会优化并行预取。
  – 如果某个容器较大且所有其它的容器都已填充，该容器是存储附加数据的唯一容器。数据库管理器不能使用并行预取存取附加的数据。
- 当使用分区内并行性时，存在足够的 I/O 容量。在 SMP 机器上，分区内并行性可以通过在多个处理器上运行该查询来减少查询所用时间。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。

  预取大小必须较大，以用于以较高速率进行预取并有效使用 I/O 容量。

  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。

相关概念:

- 第 211 页的『预取和并行性的 I/O 服务器配置』
- 第 212 页的『使用并行 I/O 预取示例』
- 第 214 页的『排序性能的准则』

排序性能的准则

因为查询经常需要已排序或已分组的结果，所以排序经常是必需的，且排序堆区域的适当配置对良好的查询性能很重要。在以下情况下需要排序:

- 没有索引满足请求的排序（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。
- 创建索引。
- 删除索引，这导致对索引页号进行排序。

排序分两个步骤:

1. 排序阶段

   排序可以是*溢出的*或*非溢出的*。如果已排序的数据不能完全放入排序堆（该排序堆是每次执行排序时分配的一块内存）中，则它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。

2. 返回排序结果的阶段。

   返回可以是*管道的*或*非管道的*。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，则它是一个管道式排序。如果已排序的信息需要返回一个临时表，则它是一个非管道排序。一个管道排序始终比一个非管道排序执行得好。

影响排序的元素

下列元素影响排序性能:

- 下列数据库配置参数的设置:
  - 排序堆大小（*sortheap*），它指定要用于每个排序的内存量
  - 排序堆阈值（*sheapthres*）和共享排序的排序堆阈值（*sheapthres_shr*），它们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程序逻辑
- 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。

一般情况下，整个实例可用的总排序内存（*sheapthres*）应尽可能大，而不会导致过度页面调度。尽管可以在排序内存中完全执行排序，但这可以导致过度页交换。在此情况下，您失去大的排序堆的优势。因此，无论何时您调整排序配置参数，应该使用一个操作系统监视器来跟踪系统页面调度中的任何变化。

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。一个管道排序可继续用尽内存，直到关闭游标。

注: 随着 DB2® 部分键二进制排序技术的提高，已包含了非整数数据类型键，因此排序长键时需要附加的内存。如果长键用于排序，则需增大 *sortheap* 配置参数。

管理排序性能的技术

标识排序对性能有主要影响的特定应用程序和语句:

- 将事件监视器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。然后审计对于此语句存取的每个表，是否存在一个适当的索引。

注: 您可以搜索说明表，以标识有排序操作的查询。

您可以使用数据库系统监视器和基准程序测试技术，来帮助设置 *sortheap* 和 *sheapthres* 配置参数。对每个数据库管理器及其数据库:

- 建立并运行一个代表性工作负载。
- 对于每个适用的数据库，在基准程序工作负载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

- 将 *sortheap* 设置为每个数据库平均*使用的排序堆总和*。
- 设置 *sheapthres*。要估计适当的大小:
  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优化此值。

相关参考:

- 第 324 页的『sortheap － 排序堆大小』
- 第 323 页的『sheapthres － 排序堆阈值』
- 第 314 页的『sheapthres_shr － 共享排序的排序堆阈值』

表管理

本节描述用于提高性能的管理表的方法。

表重组

对表数据进行大量更改之后，逻辑顺序数据可能在非顺序物理数据页，因此数据库管理器必须执行附加读操作来存取数据。另外，在删除大量行后，也需要执行附加的读操作。在这样的情况下，您可以考虑重组表以匹配索引和回收空间。此时，既可重组系统目录表，也可以重组数据库表。

注: 由于重组表的时间通常要比运行统计信息的时间长，因此您可以执行 RUNSTATS 以刷新当前的数据统计信息并重新绑定应用程序。如果刷新的统计信息并没有改善性能，则重组可能会有所帮助。有关 REORG TABLE 实用程序的选项和行为的详细信息，参阅其命令参考。

考虑以下因素，以确定是否应重组表:

- 对查询所存取的表进行了大量的插入、更新和删除活动
- 对于使用具有高群集率的索引的查询，其性能发生了明显变化
- 在执行 RUNSTATS 以刷新统计信息后，性能没有得到改善
- REORGCHK 命令指示需要重组表
- 综合考虑查询性能不断降低所浪费的成本和重组表所需的成本（包括 CPU 时间、经过的时间和 REORG 实用程序在完成重组操作之前锁定表造成的并发性降低），以确定是否进行表重组。

减少重组表的需要

要减少重组表的需要，在创建表后执行下列任务:

- 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

在执行这些任务后，具有群集索引和 PCTFREE 设置的表有助于保持原来的排序顺序。当表页上有足够的空间时，可以将新数据插入正确的页，以维护索引的群集特征。随着更多的数据插入，表页会因此而变满，记录会被追加至表的末尾，因而表将逐渐失去群集特性。

如果在创建群集索引后执行 REORG TABLE 命令，或者执行排序和 LOAD 命令，则索引会尝试维护数据的特定顺序，这将改善 RUNSTATS 实用程序所搜集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。

注: 创建多维群集（MDC）表可以减少重组表的必要性。对于 MDC 表，将在 CREATE TABLE 语句的 ORGANIZE BY DIMENSIONS 子句中指定为参数的列上维护群集。但是，如果 REORGCHK 认为存在太多未使用的块或需要对块进行压缩，则可能会建议重组 MDC 表。

相关概念:

- 第 229 页的『索引重组』
- 第 231 页的『DMS 设备注意事项』
- 第 14 页的『SMS 表空间』
- 第 15 页的『DMS 表空间』
- 第 17 页的『标准表的表和索引管理』
- 『Snapshot monitor』（*System Monitor Guide and Reference*）
- 第 20 页的『MDC 表的表和索引管理』

相关任务:

- 第 217 页的『确定何时重组表』
- 第 219 页的『选择重组表的方法』

确定何时重组表

正常的数据库活动（例如，反复插入、删除和更新）会影响经过一段时间之后表和索引中的数据的组织，并且会对数据库性能产生不利影响。 RUNSTATS 实用程序收集有关数据库表中数据的组织的统计信息，并允许您收集有关下列各项的统计信息：表大小和数据分发、索引的群集比率、索引中叶子页数目、溢出其原始页的表行数以及表中已填充的页数和空页数。还可以使用 RUNSTATS 来收集有关预取效率的信息。

RUNSTATS 收集的信息存储在系统目录表中。此信息可以帮助您决定何时重组表和索引以及需要进行哪种类型的重组。

除了使用 RUNSTATS 来捕获单个时间点的有关数据库组织的信息之外，最好是定期运行 RUNSTATS 以标识与数据库性能更改相关的整体趋势。

注: REORGCHK 命令还返回有关数据组织的统计信息，并且可以建议您是否需要重组特定表。然而，定期或在特定时间对目录统计信息表运行特定查询可以提供性能历史，该记录使您可以发现可能具有更广性能隐含的趋势。

过程:

要确定是否需要重组表，查询目录统计信息表并监视下列统计信息:

1. 行的溢出

查询 SYSSTAT.TABLES 表中的 OVERFLOW 列以监视溢出值。此列中的值表示不适合其原始页的行数。当使用比初始值更长的值来更新 VARCHAR 列时，行数据可能会溢出。在这些情况下，在行中的原始位置将保留一个指针，而实际值存储在由指针指示的另一个位置。这可能会影响性能，因为数据库管理器必须根据指针来查找行的内容。这个包括两个步骤的过程增加了处理时间，并且可能还会增加需要的 I/O 数目。

重组表数据将消除行溢出；因此，随着溢出行数的增加，重组表数据就可能会具有更多好处。

2. 访存统计信息

   当以索引顺序访问表时，查询 SYSCAT.INDEXES 和 SYSSTAT.INDEXES 目录统计信息表中的以下三个列来确定预取程序的效率。这些统计信息对照基础表来体现预取程序的平均性能的特征。

   - AVERAGE_SEQUENCE_FETCH_PAGES 列存储可以按表中的顺序存取的平均页数。可以按顺序存取的页适合于预取。较小的数目指示预取程序没有充分发挥作用，原因是它们不能读入由表空间的 PREFETCHSIZE 设置指定的所有页数。较大的数指示预取程序将有效地执行。对于群集索引和表，此数目应该接近 NPAGES（包含一些行的页数）的值。
   - AVERAGE_RANDOM_FETCH_PAGES 列存储当使用索引来访存表行时在顺序页存取之间的平均随机表页数。当大多数页按顺序时，预取程序忽略少量的随机页，并按已配置的预取大小继续预取。当表变得更加混乱时，随机访存页数就会增加。通常由于在表的末尾或在溢出页中发生了无序的插入导致这样的无组织。当使用索引来存取某一范围内的值时，这会导致降低查询性能的访存。
   - AVERAGE_SEQUENCE_FETCH_GAP 列存储当使用索引进行访存时表页序列之间的平均间隔。通过扫描索引叶子页来进行检测，每个间隔表示在表页的各个序列之间必须随机访存的平均表页数。当随机存取许多页时发生这些情况，这会中断预取程序。较大的数指示无组织的表或较差群集的索引。

3. 包含标记为已删除但未除去的 RID 的索引叶子页数

   在类型 2 索引中，当 RID 标记为删除时，通常未在物理上删除 RID。这意味着可用空间可能会被这些逻辑上已删除的 RID 占用。要检索每个 RID 都被标记为已删除的叶子页的数目，查询 SYSCAT.INDEXES 和 SYSSTAT.INDEXES 统计信息表的 NUM_EMPTY_LEAFS 列。对于并非所有 RID 都标记为删除的叶子页，逻辑上已删除的 RID 的总数存储在 NUMRIDS_DELETED 列中。

   使用此信息来通过执行带 CLEANUP ALL 选项的 REORG INDEXES 估计可以回收多少空间。要想只回收所有 RID 都被标记为已删除的页中的空间，执行带有 CLEANUP ONLY PAGES 选项的 REORG INDEXES。

4. 索引的群集比率和群集因子统计信息

   群集比率统计信息存储在 SYSTCAT.INDEXES 目录表的 CLUSTERRATIO 列中。此值（在 0 到 100 之间）表示使用索引的数据群集的程度。如果收集 DETAILED 索引统计信息，0 和 1 之间的好的群集因子统计信息存储在 CLUSTERFACTOR 列中，并且 CLUSTERRATIO 的值为 −1。这两个群集统计信息中只有一个可以记录在 SYSCAT.INDEXES 目录表中。要将 CLUSTERFACTOR 值与 CLUSTERRATIO 值进行比较，可以将 CLUSTERFACTOR 乘以 100 以获得一个百分比。

注: 通常，表中只有一个索引可以具有较高的群集度。

不是只索引存取的索引扫描在具有较高群集比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。增大缓冲区大小也可以提高非群集索引的性能。

如果表数据最初是对某个索引群集的，而群集统计信息指示现在很少为同一索引群集数据，则您可能想重组该表以再次群集数据。

5. 叶子页数

   查询 SYSCAT.INDEXES 表中的 NLEAF 列以便了解索引占用的叶子页数目。该数目告诉您对索引进行完整的扫描需要多少索引页 I/O。

   理想情况下，索引应该尽可能占用最小的空间量，以便减少进行索引扫描所需要的 I/O 次数。随机的更新活动可导致页分割，这就增大了索引的大小。当在重组表期间重新构建索引时，可以使用最小空间量来构建每个索引。

   注: 缺省情况下，当构建索引时，会在每个索引页上保留百分之十的可用空间。要增加可用空间量，当创建索引时指定 PCTFREE 参数。无论何时重组索引，都使用 PCTFREE 值。因为附加空间可以容纳附加索引插入，所以大于百分之十的可用空间可能缩小索引重组的频率。

6. 文件页的比较

   要计算表中的空页数，查询 SYSCAT.TABLES 中的 FPAGES 和 NPAGES 列并从 FPAGES 数减去 NPAGES 数。FPAGES 列存储正在使用的总页数；NPAGES 列存储包含一些行的页数。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，就更需要进行表重组。重组表时将回收空页并减少表所使用的空间量。另外，因为会将空页读入缓冲池以进行表扫描，所以回收未使用的页可以提高表扫描的性能。

相关概念:

- 第 97 页的『目录统计信息表』
- 第 216 页的『表重组』
- 第 229 页的『索引重组』

相关任务:

- 第 90 页的『收集目录统计信息』

选择重组表的方法

DB2® 提供了两种重组表的方法：传统和 in-place。一般情况下，传统表重组更快，但仅当在重组期间应用程序运行而没有对表的写访问权时才应使用它。如果环境不允许此限制，尽管 in-place 重组更慢，它可以在后台发生，而正常数据存取继续。考虑每种方法的特点，然后决定哪种方法更适用于您的环境。

过程:

要选择表重组方法，考虑下列方法的特征：

- **传统表重组方法**

  这种方法可以用最快的速度完成表重组，特别适用于不需要重组 LOB 或 LONG 数据的场合。此外，这种方法在重组表后还会以最佳的顺序重新构建索引。除重组的最后阶段之外，只读应用程序可以在其它任何阶段存取表的原始副本，因为永久性表将在此阶段取代表的镜像副本并重新构建索引。

  另一方面，考虑以下可能存在的缺点:

  – 需要大量的空间

    由于传统表重组方法需要创建表的镜像副本，因此它需要占用两倍于原始表的空间。如果重组后的表大于原始表，则重组可能需要占用两倍于原始表以上的空间。

    如果表空间不足够大，可以在临时表空间中构建镜像副本，但是取代阶段最好在相同的 DMS 表空间中进行。SMS 表空间中的表应始终将镜像副本存储在临时表空间中。

  – 受限制的表存取

    只能在重组过程的第一阶段对表进行一般性只读存取。

  – 完全成功或失败过程

    如果重组在任何一点失败，则必须从失败的节点上开始重新启动整个重组

  – 重组在调用它的应用程序的控制器中执行

    只有应用程序或那些知道如何停止进程且有权对此应用程序执行 FORCE 命令的用户才能终止重组。

  建议：如果可在维护窗口中重组表，则选择此方法。

- **In-place 表重组**

  In-place 方法的速度较慢，并且不能确保以最佳的顺序排列数据，但它允许应用程序在重组期间对表进行存取。此外，任何具有相应权限的人员均可使用模式和表名暂停和恢复 In-place 表重组。

  注：In-place 表重组方法只适用于具有类型 2 索引和无扩展索引的表。

  考虑以下缺点:

  – 非最佳索引重组

    可能在以后需要重组索引以减少索引碎片和回收索引对象空间。

  – 需要较长的时间才能完成

    在需要时，In-place 重组会延迟并发应用程序。这意味着耗时的语句或耗时应用程序中的 RR 和 RS 阅读器会使重组进度放慢。In-place 重组方法可能在产生许多小事务的 OLTP 环境中运行更快

  – 需要更大的日志空间

    由于 In-place 表重组记录其活动以便在出现意外故障后有可能恢复，因此它需要的日志空间要大于一般重组的日志空间。

In-place 重组方法所需的日志空间可能是所重组的表大小的数倍。所需的空间数量取决于要移动的行数和表中索引的数量和大小。

推荐: 如果使用最少维护窗口执行 24x7 操作，请选择 In-place 表重组方法。

有关执行这些表重组方法的详情，请参阅 REORG TABLE 语法描述。

监视表重组的进度

有关表重组的当前进度的信息将写入数据库活动的历史文件中。历史文件中包含每个重组事件的记录。要查看此文件，请执行 `db2 list history` 命令以打开包含所重组表的数据库。

此外，也可使用表快照来监视表重组的进度。 不管如何设置“数据库监视程序表”开关，系统均会记录表重组监视数据。

如果出现错误，SQLCA 转储将写入到历史文件中。对于 In-place 表重组方法，会将其状态记录为 PAUSED。

相关概念:

- 第 216 页的『表重组』
- 第 229 页的『索引重组』

相关任务:

- 第 217 页的『确定何时重组表』

索引管理

以下各节描述用于提高性能的索引重组。

索引的优点和缺点

尽管优化器决定是否使用索引来存取表数据，但除了下列情况以外，您必须决定哪些索引可以改善性能并创建这些索引。例外情况有维块索引和组合块索引，当创建多维群集（MDC）表时，为指定的每一维自动创建这些索引。

也必须在下列情况下执行 RUNSTATS 实用程序来收集关于索引的新的统计信息:

- 在创建索引之后
- 在更改预取大小之后

还应该定期执行 RUNSTATS 实用程序来保持统计信息为当前的。如果没有有关索引的最新统计信息，则优化器不能确定查询的最佳数据存取方案。

注: 要确定是否在特定的程序包中使用索引，使用 SQL“说明”设施。要计划索引，使用“控制中心”中的“设计顾问程序”或者 `db2advis` 工具来获取有关可以由一个或多个 SQL 语句使用的索引的建议。

索引比无索引的优点

如果表上不存在索引，则必须对数据库查询中引用的每个表执行表扫描。表越大，表扫描所花的时间越长，因为表扫描需要顺序存取每个表行。尽管表扫描对于需要表中大多数行的复杂查询更有效，但对于仅返回某些表行的查询，索引扫描可以更有效地存取表行。

如果在 SELECT 语句中引用了索引列且优化器估计索引扫描比表扫描快，则优化器选择索引扫描。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。应用于索引的谓词减少了要从数据页读取的行数。

| 如果对输出的排序需求可以与索引列相匹配，则按列顺序扫描索引将允许按正确顺序
| 检索行而不需要排序。

每个索引条目包含一个搜索键值和一个指向包含该值的行的指针。如果在 CREATE INDEX 语句中指定了 ALLOW REVERSE SCANS 参数，则可以按升序和降序搜索这些值。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，使数据库管理器在从表中读取这些行之后不必对它们排序。

除搜索键值和行指针外，索引还可以包括包含列，这些列是索引行中的非索引列。这样的列有可能使优化器仅从索引获取所需要的信息，而不必存取表本身。

注: 要查询的表上索引的存在并不保证已排序的结果集。只有 ORDER BY 子句才确保结果集的排序。

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

- 每个索引都需要存储器或磁盘空间。准确的容量取决于表的大小以及索引中的列的大小和数目。
- 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。对于更改索引键值的每个 UPDATE 操作，也是如此。
- LOAD 实用程序重新构建任何现有的索引或追加至现有的索引之后。

  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备用存取路径以供优化器考虑，这会增加编译时间。

谨慎选择索引来满足应用程序的需要。

相关概念:

- 『索引的空间需求』（*《管理指南：计划》*）
- 第 223 页的『索引规划技巧』
- 第 225 页的『索引性能技巧』
- 第 182 页的『设计顾问程序』
- 第 216 页的『表重组』
- 第 229 页的『索引重组』
- 第 17 页的『标准表的表和索引管理』
- 第 20 页的『MDC 表的表和索引管理』
- 第 227 页的『索引清除和维护』

相关任务:

- 『创建索引』（*《管理指南：实现》*）
- 第 90 页的『收集目录统计信息』
- 第 91 页的『收集索引统计信息』

索引规划技巧

创建的索引应该取决于数据和存取该数据的查询。

使用“控制中心”中的“设计顾问程序”或者 db2advis 工具来查找特定查询或定义工作负载的一组查询的最佳索引。此工具为索引建议了这样一些性能提高功能部件：INCLUDE 列、继承的唯一索引和 ALLOW REVERSE SCANS 索引。

以下准则可帮助您确定如何创建可用于各种目的索引：
- 要避免某些排序，只要有可能，就通过使用 CREATE UNIQUE INDEX 语句定义主键和唯一键。
- 要改善数据检索，将 INCLUDE 列添加至唯一索引。合适的列为：
  - 被频繁存取，因此可从纯索引存取中受益的列。
  - 不需要用来限制索引扫描的范围的列
  - 不影响索引键的排序或唯一性的列。
- 要有效存取小表，使用索引来优化对含有较多数据页的表的频繁查询，数据页数记录在 SYSCAT.TABLES 目录视图的 NPAGES 列中。您应该：
  - 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。
- 要有效地搜索，决定对键使用升序还是降序，这取决于将最常使用的顺序。尽管当在 CREATE INDEX 语句中指定了 ALLOW REVERSE SCANS 参数时可以按逆向方向搜索值，但是，执行按指定索引顺序的扫描比执行逆向扫描稍微更快一些。
- 要保存索引维护成本和空间：
  - 避免创建的索引是这些列上其它索引键的部分键。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。
  - 不在所有列上任意创建索引。不必要的索引不仅使用空间，而且导致大量准备时间。当使用具有动态编程连接枚举的优化级别时，这对于复杂的查询特别重要。

    使用下列一般规则来确定将为表定义的索引的典型数目。此数目根据数据库的主要使用来确定：
    - 对于联机事务处理（OLTP）环境，创建一个或两个索引
    - 对于只读查询环境，可以创建 5 个以上索引
    - 对于混合查询和 OLTP 环境，可以创建 2 到 5 个索引。
- 要提高对父表执行的删除和更新操作的性能，在外键上创建索引。
- 要提高涉及到 IMMEDIATE 和 INCREMENTAL MQT 的 DELETE 和 UPDATE 操作的性能，对 MQT 的隐含唯一键创建唯一索引，它是 MQT 定义的 GROUP BY 子句中的列。
- 对于快速排序操作，在频繁用于排序数据的列上创建索引。
- 要提高多列索引的连接性能，如果第一个键列有多项选择，则使用最常用“=”（等值连接）谓词指定的那一列，或使用如第一个键那样具有最多特异值的那些列。
- 要帮助新插入的行根据索引进行群集并避免页分割，定义一个群集索引。群集索引应显著减少重组表的需要。

当定义表时使用 PCTFREE 关键字来指定页上应该留下多少可用空间，才能允许将插入行适当地放在页上。也可以指定 LOAD 命令的 pagefreespace MODIFIED BY 子句。

- 要启用联机索引整理碎片，创建索引时使用 MINPCTUSED 选项。MINPCTUSED 指定索引叶子页中最小使用空间量的阈值并启用联机索引整理碎片。如果这些删除实际上从索引页除去键，则这可以在键删除期间以性能损失为代价而减少重组的需要。

在下列情况下，考虑创建索引:

- 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'
  ```

  通常将会从 WORKDEPT 上的索引获益，除非 WORKDEPT 列包含许多重复值。
- 在将按查询所需要的顺序对行排序的一列或多列上创建索引。不仅在 ORDER BY 子句中，而且其它功能，如 DISTINCT 和 GROUP BY 子句都需要排序。

  以下示例使用 DISTINCT 子句:

  ```
  SELECT DISTINCT WORKDEPT
    FROM EMPLOYEE
  ```

  数据库管理器可使用 WORKDEPT 上定义为升序或降序的索引来消除重复值。此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
  ```
- 使用复合键创建索引，该键命名语句中引用的每个列。当用此方式指定索引时，可以从纯索引检索数据，这比存取表可更有效。

  例如，考虑下列 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。
- 使用 INCLUDE 列创建索引可改善表上索引的使用。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 列而不是索引键的一部分，意味着 lastname 只存储在索引的叶子页上。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 225 页的『索引性能技巧』
- 第 182 页的『设计顾问程序』
- 第 229 页的『索引重组』

- 第 230 页的『联机索引整理碎片』
- 『多维群集（MDC）表创建、布置和使用』（*《管理指南：计划》*）

索引性能技巧

考虑关于使用和管理索引的下列建议:

- **在创建和重组索引时指定并行性**

  当对 SMP 机器主管的大型表创建和重组索引时，可考虑将 *intra_parallel* 设置为 YES（1）或 SYSTEM（-1），以利用并行性能提高。

  可使用多个处理器来扫描数据并对数据排序。

- **指定大型实用程序堆**

  CREATE INDEX 和 REORG INDEXES 都支持其他用户或其它应用程序对基础表进行写存取。当您期望为正在创建或重组的索引而对基础表执行大量更新活动时，请考虑配置大型实用程序堆。在同步更新阶段，大型实用程序堆将提高索引创建或索引重组的速度。对正在创建或重组的索引执行的所有写活动都将记录在 DB2® 日志和内部内存缓冲空间中。内部内存缓冲空间是根据需要从实用程序堆中分配的指定内存区域，它用来存储对正在创建或重组的索引所作的更改。使用该内存区域可以使同步更新阶段能够更快地工作。在创建或重组操作完成后，将释放所分配的内存。如果确保有足够的实用程序堆来容纳对正在创建或重组的索引进行的全部或大部分更改，则会对同步更新阶段的性能产生非常积极的影响。

- **为索引指定不同的表空间**

  可将索引存储在与表数据不同的表空间中。这样，通过减少索引存取期间读／写磁头的移动，可以更有效地使用磁盘存储器。也可以在更快的物理设备上创建索引表空间。而且，可以将索引表空间指定给不同的缓冲池，由于它们不与表数据页竞争，这可以将索引页较长时间保存在缓冲区内。

  当不将索引放置在不同的表空间中时，数据页和索引页使用相同的扩展数据块大小和预取量。如果对索引使用不同的表空间，则可为表空间的所有特征选择不同的值。因为索引通常比表小，且分布在更少的容器中，因此索引经常只具有较小的扩展数据块大小，如 8 和 16 页。当 SQL 优化器选择存取方案时，它考虑用于表空间的设备的速度。

- **确保群集度**

  如果 SQL 语句需要排序（例如，ORDER BY、GROUP BY 和 DISTINCT），则即使索引可以满足排序，但是在下列情况下，优化器可能不会选择索引:
  - 索引群集程度较低。有关信息，检查 SYSCAT.INDEXES 的 CLUSTERRATIO 和 CLUSTERFACTOR 列。
  - 由于表很小，因此扫描该表并在内存中对回答集进行排序所消耗的成本更低。
  - 对存取该表有多个索引可供选择。

  创建群集索引之后，以传统方式执行 REORG TABLE，这可创建组织完好的索引。要重新群集表，可以执行排序和 LOAD 来代替。通常，只能在一个索引上群集表。在构建群集索引之后构建附加索引。群集索引试图维护数据的特定顺序，以改进 RUNSTATS 实用程序收集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。

要帮助维护群集比率，在装入或重组表之前，在改变表时指定适当的 PCTFREE。PCTFREE 指定的每页上的可用空间提供用于插入的空间，以便可以适当地群集这些插入。如果不为表指定 PCTFREE，则重组会消除所有额外空间。

注: 目前，在更新期间不维护群集，除非您正在使用范围群集表。即，如果更新了一条记录，以便其键值在群集索引中更改，则不必将该记录移动至新页来维护群集顺序。要维护群集，使用 DELETE，然后使用 INSERT，而不是使用 UPDATE。

- **保持表和索引统计信息为最新的**

  当创建新索引后，运行 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提高存取性能。

- **启用联机索引整理碎片**

  如果将 MINPCTUSED 子句设置为大于零，则对索引启用联机索引整理碎片。当页上的可用空间为指定级别或低于指定级别而索引仍可用时，联机索引整理碎片允许通过合并叶子页来压缩索引。

- **必要时重组索引**

  要从索引获取最佳性能，考虑定期重组索引，这是因为对表进行更新会使得索引页预取效率降低。

  要重组索引，删除它并重新创建它，或使用 REORG 实用程序。

  要减小对频繁重组的需要，当创建索引时，指定适当的 PCTFREE 来在创建的每个索引叶子页上保留一定百分比的可用空间。在将来的活动中，可将记录插入索引，从而使索引页分割的可能性较小。页分割导致索引页既不是连续的也不是按顺序的，从而导致降低索引页预取的效率。

  注: 在重组索引时，保留创建该索引时指定的 PCTFREE。

  若删除并重新创建或重组索引，也会创建大致连续且按顺序排列的一组新页，并提高索引页预取。尽管时间和资源方面的成本更高，但 REORG TABLE 实用程序也确保数据页的群集。群集为存取大量数据页的索引扫描带来很大的好处。

  在对称多处理器（SMP）环境中，如果 *intra_parallel* 数据库管理器配置参数为 YES 或 ANY，则“传统的”REORG TABLE 方式（该方式使用镜像表进行快速表重组）可以使用多个处理器来重新构建索引。

- **分析关于索引使用情况的 EXPLAIN 信息**

  定期在您最频繁使用的查询上运行 EXPLAIN，并验证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，考虑删除该索引。

  EXPLAIN 信息也允许您查看在大型表上的表扫描是否是作为嵌套循环连接的内部表来处理的。如果是这样，则连接谓词列上的索引或者丢失，或者被认为应用于该连接谓词的效率不高。

- **对大小变化范围大的表使用易失表。**

  *易失表*是其大小在运行时可以从空变为很大的一种表。对于这种表，其中的基数变化范围很大，优化器可能生成适合表扫描而不是索引扫描的存取方案。

使用 ALTER TABLE...VOLATILE 语句声明表是“易失的”，会允许优化器对易失表使用索引扫描。在下列情况下，无论统计信息如何，优化器将使用索引扫描而不是使用表扫描:

– 引用的所有列都在索引中
– 索引可以在索引扫描中应用谓词。

如果该表是一个类型表，则在该类型表层次结构的根表中仅支持使用 ALTER TABLE...VOLATILE 语句。

相关概念:

• 第 221 页的『索引的优点和缺点』
• 第 223 页的『索引规划技巧』
• 第 22 页的『索引结构』
• 第 140 页的『索引存取和群集比率』
• 第 216 页的『表重组』
• 第 229 页的『索引重组』
• 第 17 页的『标准表的表和索引管理』
• 第 230 页的『联机索引整理碎片』
• 第 20 页的『MDC 表的表和索引管理』
• 第 227 页的『索引清除和维护』

相关参考:

• 第 413 页的『intra_parallel － 启用分区内并行性』

索引清除和维护

在创建索引之后，除非使索引保持压缩和有组织，否则性能会下降。考虑以下建议使索引尽可能小并效率高:

• 启用联机索引整理碎片

  使用 MINPCTUSED 子句创建索引。如果必要，删除并重新创建现有的索引。

• 如果频繁的 COMMITS 不可能，则显式或通过锁定升级来执行频繁的 COMMIT 或获取对表的 X 锁定。

  可以在 COMMIT 后从表中物理除去标记为删除的索引键。当删除的键标记为删除时，表的 X 锁定允许在物理上除去删除的键，如下所示。

• 使用 REORGCHK 来帮助确定何时重组索引或／和表以及何时使用带 CLEANUP ONLY 选项的 REORG INDEXES。

  要在重组期间允许对索引的读和写存取，运行带有 ALLOW WRITE ACCESS 选项的 REORG INDEXES。

注: 在 DB2® 版本 8.1 和更高版本中，所有新索引都创建为类型 2 索引。一个例外是当您在已具有类型 1 索引的表上添加索引时。仅在这种情况下，新索引也将是类型 1 索引。要了解一个表存在什么类型的索引，执行 INSPECT 命令。要将类型 1 索引转换为类型 2 索引，执行 REORG INDEXES 命令。

类型 2 索引的主要优点如下所示:

- 可以在其长度大于 255 个字节的列上创建索引。
- 使下一键锁定的使用减小到最小，这会改善并发性。因为将键标记为删除而不是在物理上从索引页除去，所以消除了大多数下一键锁定。有关键锁定的信息，参阅讨论锁定性能含义的主题。

在下列情况下，清除标记为删除的索引键:

- 在后续插入、更新或删除活动期间

  在键插入期间，清除标记为删除和已知要落实的键（如果这样的清除可以避免执行页分割的需要并防止增大索引的大小）。

  在键删除期间，当某个页上的所有键标记为删除时，尝试查找其中所有键标记为删除并且所有删除已落实的另一个索引页。如果找到这样的页，将从索引树中删除它。

  如果删除键时存在对表的 X 锁定，则会在物理上删除该键，而不只是标记为删除。在此物理删除期间，如果相同页上的任何已删除的键标记为删除且已知要落实，则也会除去这些键。
- 当用 CLEANUP 选项执行 REORG INDEXES 命令时

  CLEANUP ONLY PAGES 选项搜索并释放索引页，该页上的所有键都标记为删除并已知要落实。

  CLEANUP ONLY ALL 选项不仅释放其上所有键都标记为删除并已知要落实的索引页，它还除去一些页上的标记为删除并已知要落实的 RID，这些页包含某些未删除的 RID。

  此选项也尝试合并相邻叶子页（如果这样做会导致合并的叶子页上具有至少 PCTFREE 可用空间）。PCTFREE 值是当创建索引时为索引定义的可用空间的百分比。缺省 PCTFREE 是百分之十。如果可以合并两页，则将释放其中一页。
- 索引的任何重新构建

  重新构建索引的实用程序包括下列各项:
  - 当未使用其中一个 CLEANUP 选项时的 REORG INDEXES
  - 当未使用 INPLACE 选项时的 REORG TABLE
  - 带 REPLACE 选项的 IMPORT
  - 带 INDEXING MODE REBUILD 选项的 LOAD

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 223 页的『索引规划技巧』
- 第 22 页的『索引结构』
- 第 216 页的『表重组』
- 第 229 页的『索引重组』

索引重组

通过删除和插入操作对表进行更新后，索引的性能会降低，其表现方式如下:

- 叶子页分段

  叶子页被分段之后，由于必须读取更多的叶子页才能访存表页，因此 I/O 操作成本会增加。
- 物理索引页的顺序不再与这些页上的键顺序相匹配（此称为*不良群集*索引）。

  叶子页出现不良群集情况后，随之将造成预取操作效率降低，因而会导致更多的 I/O 等待。
- 形成的索引大于其最有效的级别数。

  在此情况下应重组索引。

如果在创建索引时设置了 MINPCTUSED 参数，则在删除某个键且可用空间小于指定的百分比时，数据库服务器会自动合并索引叶子页。此过程称为*联机索引整理碎片*。但是，要复原索引群集和可用空间以及降低叶级别，请使用下列其中一种方法:

- 删除并重新创建索引。
- 使用 REORG INDEXES 命令联机重组索引。

  因为此方法允许用户在重新构建表索引期间对表进行读写操作，所以在生产环境中可能需要选择它。
- 使用允许脱机重组表及其索引的选项运行 REORG TABLE 命令。

联机索引重组

在使用 ALLOW WRITE ACCESS 选项运行 REORG INDEXES 命令时，如果同时允许对指定的表进行读写存取，则会重新构建该表的所有索引。进行重组时，对基础表所作的任何将会影响到索引的更改都将记录在 DB2® 日志中。另外，如果有任何内部内存缓冲空间可供使用，则还将这些更改放在这样的内存空间中。重组将处理所记录的更改以便在重新构建索引时与当前写活动保持同步更新。内部内存缓冲空间是根据需要从实用程序堆中分配的指定内存区域，它用来存储对正在创建或重组的索引所作的更改。使用内存缓冲空间使索引重组操作能够通过这样的方式来处理更改，即先直接从内存读取，然后读取日志（如有必要），但读取日志的时间要晚得多。在重组操作完成后，将释放所分配的内存。重组完成后，重新构建的索引可能不是最佳群集的索引。如果为索引指定 PCTFREE，则在重组期间，每页上均会保留相应百分比的空间。

注: REORG INDEXES 命令的 CLEANUP ONLY 选项不能完全重组索引。CLEANUP ONLY ALL 选项将除去那些标记为“删除”且被认为要落实的键。此外，它还将释放所有标记为“删除”且被认为要落实的键所在的页。在释放页后，相邻的叶子页将会合并，前提是这样做可以在合并页上至少留出 PCTFREE 可用空间。PCTFREE 是指在创建索引时为其定义的可用空间百分比。CLEANUP ONLY PAGES 选项仅删除那些标记为“删除”且被认为要落实的所有键所在的页。

REORG INDEXES 具有下列要求:

- 对索引和表具有 SYSADM、SYSMAINT、SYSCTRL 或 DBADM 权限，或 CONTROL 特权。
- 用于存储索引的表空间的可用空间数量等于索引的当前大小

在发出 CREATE TABLE 语句时，考虑在大型表空间中重组索引。

- 附加日志空间

  REORG INDEXES 需要记录其活动。因此，重组可能会失败，尤其是在系统繁忙和记录其它并发活动时。

  注: 如果具有 ALLOW NO ACCESS 选项的 REORG INDEXES ALL 命令运行失败，则会标记此索引不良并且此项操作不可撤销。但是，如果具有 ALLOW READ ACCESS 选项的 REORG 命令或具有 ALLOW WRITE ACCESS 选项的 REORG 命令运行失败，则可以复原原来的索引对象。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 223 页的『索引规划技巧』
- 第 225 页的『索引性能技巧』
- 第 230 页的『联机索引整理碎片』
- 第 227 页的『索引清除和维护』

相关任务:

- 第 219 页的『选择重组表的方法』

联机索引整理碎片

用户定义的索引叶子页上的最小已使用空间量的阈值将启用联机索引整理碎片。当从叶子页删除了索引键且超过阈值时，可检查相邻的索引叶子页以确定是否可合并两个叶子页。如果某一页上有足够的空间用于合并两个相邻的页，则立即在背景中合并。

索引的联机整理碎片只能对在版本 6 和更高版本中创建的索引使用。如果现有的索引需要合并联机的能力，则必须删除它们然后用 MINPCTUSED 子句重新创建。将 MINPCTUSED 值设置为小于 100。MINPCTUSED 的建议值小于 50，因为目标是将两个相邻的索引叶子页合并。MINPCTUSED 的零值（也是缺省值）禁用联机整理碎片。

当除去一页上的最后一个索引键时，便从索引中释放该页。当指定了 CREATE INDEX 语句中的 MINPCTUSED 子句时，便会发生这一规则的例外情况。MINPCTUSED 子句指定索引叶子页上空间的百分比。当删除索引键时，如果页上填充空间的百分比等于或低于指定值，数据库管理器尝试合并剩余键和相邻页上的键。如果相邻页上有足够的空间，则执行合并，并删除一个索引叶子页。

联机索引整理碎片期间，不合并索引非叶子页。但是，删除空非叶子页以供相同表上的其它索引重新使用。要对 DMS 存储模型中的其它对象释放这些非叶子页，或者要释放 SMS 存储模型中的磁盘空间，则对表或索引进行全面重组。表和索引的全面重组可以使索引尽可能地小。联机索引整理碎片期间，不合并索引非叶子页。但如果它们成为空的，删除并释放它们，以便重新使用。可能会减少索引中的层数和叶和非叶子页的数量。

对于类型 2 索引，仅当对表具有 X 锁定时，才在删除键期间从页除去键。这种操作期间，联机索引整理碎片将有效。但是，如果在键删除期间对表没有 X 锁定，键标记为已删除，但实际上并未从索引页上除去。结果整理碎片未成功。

要整理键标记为已删除但实际上仍在索引页中的类型 2 索引的碎片，用 CLEANUPONLYALL 选项执行 REORG INDEXES 命令。CLEANUP ONLY ALL 选项整理索引碎片，与 MINPCTUSED 的值无关。如果用 CLEANUP ONLY ALL 执行 REORG INDEXES，则如果两个相邻叶子页的合并至少可以在合并页上留下 PCTFREE 可用空间，则合并这两个相邻叶子页。在索引创建时间指定 PCTFREE，且缺省为百分之十。

相关概念:

- 第 221 页的『索引的优点和缺点』
- 第 225 页的『索引性能技巧』
- 第 22 页的『索引结构』
- 第 229 页的『索引重组』

DMS 设备注意事项

如果对表空间使用“数据库管理存储器”（DMS）设备容器，考虑下列因素以便有效管理:

- **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: 在 Windows® NT 上，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，则消除文件系统高速缓存。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。消除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。在某些情况下，在应用程序实际使用一个数据页之前，可能从缓冲池释放该页，特别在其它数据页需要缓冲池空间时，更是如此。对于使用系统管理存储器（SMS）或数据库管理存储器（DMS）文件容器的表空间，以上文件系统高速缓存可以消除另外将需要的 I/O。

  使用数据库管理存储器（DMS）设备容器的表空间不使用文件系统或其高速缓存。因此，您可以增大数据库缓冲池的大小，减小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 表空间并未执行双缓冲区。

  如果系统级别的监视工具显示：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双缓冲区所导致的。

- **使用 LOB 或 LONG 数据**

当一个应用程序检索 LOB 或 LONG 数据时，数据库管理器不在它的缓冲区中高速缓存该数据，每次应用程序需要其中一个页时，数据库管理器必须从磁盘检索它。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，所以应该将它们保存在 SMS 表空间或 DMS 文件表空间中。

相关概念:

- 第 12 页的『数据库目录和文件』
- 第 14 页的『SMS 表空间』
- 第 15 页的『DMS 表空间』

代理进程管理

本节描述数据库管理器如何使用代理进程和如何管理代理进程以获取良好性能。

数据库代理进程

对于应用程序存取的每个数据库，各种进程或线程启动来执行各种应用程序任务。这些任务包括日志记录、通信和预取。

数据库代理进程是引擎可调度单元（EDU）进程或线程。数据库代理进程在数据库管理器中执行应用程序请求的工作。在 UNIX® 环境中，这些代理进程作为进程运行。在基于 Intel 的操作系统（例如， Windows®）中，代理进程作为线程运行。

最大应用程序连接数由 *max_connections* 数据库管理器配置参数控制。每个应用程序连接的工作由单个工作程序代理进程协调。

*工作程序代理进程*执行应用程序请求，但并不与任何特定应用程序永久相连。协调程序工作程序代理进程具有在数据库管理器中完成应用程序所请求的操作所需的所有信息和控制块。

有四种类型的工作程序代理进程:

- 空闲代理进程
- 不活动代理进程
- 活动协调代理进程
- 子代理进程

空闲代理进程
: 这是最简单的工作程序代理进程形式。它不带出站连接，且不带本地数据库连接或实例连接。

不活动代理进程
: 不活动代理进程是这样的工作程序代理进程：它不处于活动事务中，不带出站连接，不带本地数据库连接或实例连接。不活动代理进程自由开始对应用程序连接执行工作。

活动协调代理进程
: 一个客户机应用程序的每个进程或线程都有一个在数据库上协调其工作的活动代理进程。在创建了协调代理进程之后，它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它代理进程进行通信。每个代理进程使用它自己的专用内存来运行，并与其它代理进程共享数据库管理器和数据库全局资源，如缓冲池。当事务完成时，活动协调代理进程可以成为不活动代理进程。

当客户机与数据库断开连接或者与实例拆离时，其协调代理进程将为:

- 活动代理进程。如果其它连接正在等待，则工作程序代理进程成为活动协调代理进程。
- 被释放并标记为空闲（如果没有连接正在等待且尚未达到池代理进程的最大数目的话）。
- 被终止并释放其存储器（如果没有连接正在等待且已达到缓冲池代理进程的最大数目的话)。

子代理进程

在分区数据库环境和启用了分区内并行性的环境中，协调代理进程会将数据库请求分发至子代理进程，然后由这些代理进程执行对该应用程序的请求。在创建了该协调代理进程之后，它将通过协调执行数据库请求的子代理进程，来代表它的应用程序处理所有数据库请求。

不为任何应用程序执行工作且正在等待分配的代理进程被认为是空闲代理进程，并驻留在一个*代理进程缓冲池*中。这些代理进程可用于处理为客户机程序运行的协调代理进程发出的请求，或可用于现有协调代理进程运行的子代理进程。可用的代理进程数目取决于数据库管理器配置参数 *maxagents* 和 *num_poolagents*。

当一个代理进程完成其工作但仍具有与数据库的连接时，将它放在代理进程缓冲池中。不管是否为数据库启用了连接集中器，如果在一定的时间周期中未唤醒代理进程来为新的请求服务，且活动和合用的代理进程的当前数量大于 *num_poolagents*，则终止该进程。

重用代理进程池（*num_poolagents*）中的代理进程作为下列几种应用程序的协调代理进程:

- 远程基于 TCP/IP 的应用程序
- 基于 UNIX 的操作系统上的本地应用程序
- Windows 操作系统上的本地和远程应用程序。

其它类型的远程应用程序总是创建新代理进程。当需要一个代理进程时，如果没有空闲的代理进程存在，动态创建一个新的代理进程。因为创建一个新代理进程需要一定量的开销，因此，如果可为客户机激活空闲代理进程，可提高 CONNECT 和 ATTACH 性能。

当一个子代理进程代表一个应用程序工作时，它是与那个应用程序*相关的*。在它完成指定的工作后，可以将它放在代理进程缓冲池中，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，数据库管理器创建新的代理进程之前，首先检查相关代理进程的空闲缓冲池。

相关概念:

- 第 234 页的『数据库代理进程管理』
- 第 237 页的『分区数据库中的代理进程』
- 第 235 页的『客户机连接的连接集中器改善』

- 第 234 页的『影响代理进程数的配置参数』

数据库代理进程管理

大多数应用程序在连接的应用程序数与数据库可以处理的应用程序请求数之间建立一对一关系。但是，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。

两个数据库管理器配置参数提供了各自控制这些因子的能力:

- *max_connections* 参数，它指定已连接的应用程序数
- *max_coordagents* 参数，它指定可以处理的应用程序请求数

当 *max_connections* 的值大于 *max_coordagents* 的值时，启用连接集中器。

因为每个活动协调代理进程需要全局资源开销，所以这些代理进程数越大，达到可用数据库全局资源上限的机会也就越大。要防止达到可用数据库全局资源的上限，可将 *max_connections* 的值设置为高于 *max_coordagents* 的值。

相关概念:

- 第 237 页的『分区数据库中的代理进程』
- 第 235 页的『客户机连接的连接集中器改善』
- 第 234 页的『影响代理进程数的配置参数』

影响代理进程数的配置参数

下列数据库管理器配置参数确定创建多少数据库代理进程和如何管理它们:

- 最大代理进程数（*maxagents*）： 可在任何时间工作的代理进程数。此值适用于在所有应用程序上工作的代理进程总数，包括协调代理进程、子代理进程、不活动代理进程和空闲代理进程。
- 代理进程池大小（*num_poolagents*）： 系统中可用的代理进程的总数，包括活动代理进程和代理进程池中的代理进程。此参数的缺省值是对 *maxagents* 指定的数目的一半。
- “池中初始代理进程数”（*num_initagents*）： 当启动数据库管理器时，根据此值创建一个工作程序代理进程池。这提高了初始查询的性能。工作程序代理进程全都以空闲代理进程开始。
- “最大连接数”（*max_connections*）： 指定每个分区上允许的与数据库管理器系统的最大连接数。
- “最大协调代理进程数”（*max_coordagents*）： 当启用了连接协调程序时，对于分区数据库环境和启用了分区内并行性的环境，此值限制协调代理进程的数目。
- “最大并发代理进程数”（*maxcagents*）： 此值控制数据库管理器允许的*标记*数。对于将客户机连接至数据库时所进行的每个数据库事务（工作单元），协调代理进程必须从数据库管理器获得许可权来处理该事务。此许可权称为*处理标记*。数据库管理器只允许具有处理标记的代理进程对数据库执行工作单元。如果标记不可用，代理进程必须一直等到有可用的标记来处理该事务。

在高峰使用需求超过内存、CPU 和磁盘的系统资源的环境中此参数可能很有用。例如，在这种环境中，页面调度可导致高峰负载期间性能降低。尽管会影响并发性和／或等待时间，您仍然可以使用此参数来控制负载，并避免性能降低。

相关概念:

- 第 232 页的『数据库代理进程』
- 第 234 页的『数据库代理进程管理』
- 第 237 页的『分区数据库中的代理进程』

客户机连接的连接集中器改善

对于具有许多相对瞬时的连接的因特网应用程序或相似类型的应用程序，连接集中器通过允许有效地处理更多客户机连接来改善性能。它也减少每个连接的内存使用，并减少上下文转接的次数。

注: 当 *max_connections* 的值大于 *max_coordagents* 的值时，启用连接集中器。

在要求同时发生许多用户连接的环境内，可以启用连接集中器以更有效地使用系统资源。此功能部件结合了以前仅在 DB2 Connect 连接池中的优点。《DB2 Connect 用户指南》中描述了连接池和连接集中器。首次连接后，连接集中器缩小对主机的连接时间。当请求与主机断开连接时，会断开入站连接，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® 尝试重新使用该存储池中现有的出站连接。

注: 当应用程序使用连接池或连接集中器时，调整控制高速缓存数据块大小的参数，以获取最佳性能。有关更多信息，参考《DB2 Connect 用户指南》。

使用 DB2Connect 连接池和连接集中器，在客户机断开连接后，活动代理进程不会关闭它的出站连接，但放置在应用程序的代理进程池中，并在该池中成为*逻辑子代理进程*，该子代理进程受*逻辑协调代理进程*控制。具有远程主机的活动连接。

当使用连接池时，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。在连接池中，这些空闲的代理进程称为*不活动的代理进程*。不活动的代理进程池与出站连接池同义。连接集中器实现了一个类似的办法，即将不活动的代理进程保留在应用程序特定池中以供以后使用。

用法示例:

1. 考虑具有单个数据库分区的 ESE 环境其中，平均有 1000 个用户连接到该数据库。有时，并发事务的数目高达 200，但绝不会高于 250。事务较短。

   对于此工作负载，管理员设置下列数据库管理器配置参数:

   - *max_connections* 设置为 1000 以确保支持平均的连接数。
   - *max_coordagents* 设置为 250 以支持最大的并发事务数。
   - *maxagents* 设置为足够高以支持在节点上执行事务所需要的所有协调代理进程和子代理进程（如果适用）。

     如果 *intra_parallel* 是 OFF，则 *maxagents* 设置为 250，因为这种环境下没有子代理进程。如果 *intra_parallel* 是 ON，则 *maxagents* 应该设置为足够大，以满足在节点上存取数据的每个事务所需要的协调代理进程和子代理进程的需要。例如，如果每个事务要求 4 个子代理进程，则 *maxagents* 应该设置为 (4+1) * 250,

它是 1250。要进一步调整 *maxagents*，对数据库管理器进行监视器快照。代理进程的高水位标记将指示 *maxagents* 的正确设置。

- *num_poolagents* 至少设置为 250 或高达 1250，这取决于 *maxagents* 的值，以确保有足够的数据库代理进程可用于服务入局客户机请求，而避免创建新的代理进程的开销。

  但是，可以降低此数以在低使用周期期间减少资源使用。将此值设置过低会导致释放代理进程而不是将保存到代理进程池，这会要求创建新的代理进程，服务器才可以处理平均工作负载。

- *num_init_agents* 设置为与 *num_poolagents* 相同，因为您知道应该是活动的代理进程的数目。这导致数据库在启动时创建正确数目的代理进程而不是在可以处理给定请求之前创建它们。

此处不讨论处理给定工作负载的基础硬件的能力。如果基础硬件不能处理同时工作的 X 个代理进程，那么，您需要将此数目减少到硬件可以支持的最大数目。例如，如果最大数目仅为 1500 个代理进程，那么，这限制可以处理的并发事务的最大数目。您应该监视这种与性能相关的设置，因为并非总是能够确定在所给定的时间点发送给其它节点的确切请求。

2. 在工作负载需要限制为最大 100 个并发事务，且连接的用户的数目与示例 1 一样的系统内，可以将数据库管理器配置参数设置为如下:
   - *max_coordagents* 设置为 100
   - *num_poolagents* 设置为 100

   使用这些设置，可以同时执行事务的最大客户机数为 100。当所有客户机断开连接时，100 个代理进程等待服务新的客户机连接。但是，您应该基于工作负载的类型、查询内并行性设置、数据库分区数目和基础硬件，来设置 *maxagents*。
3. 下一步考虑具有五个数据库分区的 ESE 安装，其中每个分区平均有 1000 个用户连接，且并发事务高达 200，但从不高于 250，将数据库配置参数设置为如下:
   - *max_coordagents* 设置为 250，因为与示例 1 一样，最多 250 台客户机可以同时执行事务。
   - *maxagents* 设置为 1500，假定数据分布在五个分区，每个事务至少可以在系统中的每个节点上执行一小节。（（1 个协调代理进程 + 5 个子代理进程）* 250 = 1500。）
   - *num_poolagents* 设置为 1200（假定最大值为 250 的 200 个并发事务的平均值。因此，平均所需的代理进程的数目将为 (1+5)*200 = 1200。）
   - *num_init_agents* 设置为与 *num_poolagents* 相同，与示例 1 中一样。
4. 在不想启用连接集中器，但要一次允许 250 个连接的用户的系统中，将数据库管理器配置参数设置为如下:
   - *max_connections* 设置为 250。
   - *max_coordagents* 设置为 250。

相关概念:

- 第 232 页的『数据库代理进程』
- 第 234 页的『数据库代理进程管理』
- 第 9 页的『DB2 体系结构和进程概述』
- 第 31 页的『内存管理』

分区数据库中的代理进程

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）有它自己的代理进程存储池，可从中抽出子代理进程。因为有此缓冲池，子代理进程就不必在每次需要时创建或完成其工作时破坏掉。这些子代理进程可以仍然作为此缓冲池中的相关代理进程，并由数据库管理器使用，以执行与它们相关的应用程序发出的新请求。

注: 如果启用了连接集中器，则子代理进程不需要和应用程序关联。

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进程存储池是如何调整的有很大关系:

- 代理进程缓冲池大小的数据库管理器配置参数（*num_poolagents*）影响可与一个分区（也称为节点）上的应用程序保持相关的子代理进程的总数。如果缓冲池大小太小且该缓冲池已满，则子代理进程解除它自己与它正在其上工作的应用程序的关联并终止。因为必须常常创建子代理进程，并重新使它们与应用程序相关，所以性能会受损。

  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进程的缓冲池。因此，当另一个应用程序需要新的子代理进程，并且在其相关的代理进程缓冲池中没有子代理进程时，它将从其它应用程序的代理进程缓冲池中“窃用”子代理进程。这种情况成本高，且导致较差的性能。
- 必须根据在任何给定时间允许过多代理进程活动的资源成本权衡这些情况。

  例如，如果 *num_poolagents* 的值太大，使用不能用于其它任务的数据库管理器资源，则相关的子代理进程也许很长时间都留在缓冲池中未使用。

  注: 当启用连接集中器时，*num_poolagents* 指定的代理进程数仅仅是建议性的。在任何给定的时间内，代理进程池中可能有更多的代理进程。

其它异步进程和线程

除数据库代理进程外，其它异步数据库管理器活动还作为它们自己的进程或线程运行，包括:

- 数据库 I/O 服务器或 I/O 预取程序
- 数据库异步页清除程序
- 数据库记录器
- 数据库死锁检测器
- 事件监视器
- 通信和 IPC 侦听器
- 表空间容器重平衡程序。

相关概念:

- 第 211 页的『预取和并行性的 I/O 服务器配置』
- 第 212 页的『使用并行 I/O 预取示例』
- 第 232 页的『数据库代理进程』
- 第 234 页的『数据库代理进程管理』
- 第 234 页的『影响代理进程数的配置参数』

数据库系统监视器信息

DB2® 数据库管理器维护有关它的操作、性能和使用它的应用程序的数据。当数据库管理器运行时维护此数据，并且此数据可提供重要的性能和故障诊断信息。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）。
- 显示数据库管理器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挂起的锁定的列表。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，包括哪个应用程序挂起该锁定。

因为要收集一些这种数据会给 DB2 的操作带来开销，所以可用**监视开关**来控制要收集哪些信息。要明确设置监视开关，使用 UPDATE MONITOR SWITCHES 命令或 sqlmon() API。（您必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限。）

可以通过获取快照或通过使用事件监视器来存取数据库管理器维护的数据。

获取快照

可以采用下面两种方式的其中一种来获取快照:

- 从命令行使用 GET SNAPSHOT 命令。
- 使用 sqlmonss() API 调用编写自己的应用程序。

使用事件监视器

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，一个事件监视器将捕获系统监视器信息。可以将此信息写入文件或命名管道中。

要使用事件监视器:

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 语句激活事件监视器:

   ```
   SET EVENT MONITOR evname STATE 1
   ```

   如果写入一个命名管道，则在激活事件监视器之前，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**。一旦事件监视器被激活，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视器，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。
   - 单击基于 Windows® 的操作系统上的控制中心中的**事件分析器**图标，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: 如果监视的数据库系统不在控制中心所在的机器上运行，在可以查看跟踪之前必须将事件监视器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。

相关概念:

- 第 6 页的『性能调整的快速启动技巧』

第 9 章 使用控制器

本章描述如何设置和运行控制器工具来监视和控制数据库活动。

控制器实用程序

控制器可以监视对数据库运行的应用程序的行为并可以更改某些行为，这取决于在控制器配置文件中指定的规则。

控制器实例由一个前端实用程序和一个或多个守护进程组成。您启动的控制器的每个实例是特定于数据库管理器的一个实例。缺省情况下，当启动控制器时，控制器守护进程在分区数据库的每个分区上启动。但是，可以指定在要监视的单一分区上启动守护进程。

注： 当控制器活动时，其快照请求可能会影响数据库管理器性能。要提高性能，增大控制器唤醒时间间隔以降低其 CPU 使用情况。

每个控制器守护进程收集关于对数据库运行的应用程序的信息。然后它对照您在控制器配置文件中对此数据库指定的规则检查此信息。

控制器按在配置文件中规则所指定的那样管理应用程序事务。例如，应用某个规则可能指示应用程序正使用过多的特定资源。规则将指定要执行的操作，如更改应用程序的优先级或强制应用程序与数据库断开连接。

如果与某个规则相关的操作更改应用程序的优先级，则控制器在发生资源违规的数据库分区上更改代理进程的优先级。在分区数据库中，如果强制将应用程序与数据库断开连接，则该操作发生，即使检测到违规的守护进程正在应用程序的协调程序节点上运行也是如此。

控制器记录它所执行的任何操作。要复查操作，查询日志文件。

相关概念:

- 第 243 页的『控制器守护进程』
- 第 244 页的『控制器配置文件』
- 第 251 页的『控制器日志文件』

相关任务:

- 第 242 页的『启动和停止控制器』
- 第 244 页的『配置控制器』

相关参考:

- 『db2gov - DB2 Governor Command』（*Command Reference*）

控制器启动和关闭

本节说明如何启动和停止控制器工具，并描述控制器守护进程的活动。

启动和停止控制器

控制器实用程序监视连接至数据库的应用程序，并根据在控制器配置文件中对该数据库指定的规则更改应用程序的行为。

先决条件:

启动控制器之前，必须创建配置文件。

限制:

要启动或停止控制器，必须具有 *sysadm* 或 *sysctrl* 权限。

过程:

要启动或停止控制器:

1. 要启动控制器，在 DB2 命令行执行 *db2gov* 命令。输入下列必需的参数:
   - START *database_name*

     指定的数据库名称必须与指定的配置文件中的数据库名称匹配。如果名称不相同，则返回一条错误。注意，如果控制器是对多个数据库运行的，则将为每个数据库启动一个守护进程。
   - *config_file_name*

     此数据库上控制器的配置文件的名称。如果该文件不在缺省位置（该位置为 sqllib 目录），则必须包括路径和文件名。
   - *log_file_name*

     此控制器的日志文件的基本名称。在分区数据库上，在守护进程为此控制器实例运行的每个分区上添加一个分区号。

   要在分区数据库的单一分区上启动控制器，添加 *nodenum* 选项。例如，要使用配置文件（称为 *salescfg*）和日志文件（称为 *saleslog*）仅在分区数据库的节点 3 上为数据库（称为 *sales*）启动控制器，输入下列命令:

   ```
   db2gov START sales nodenum 3 salescfg saleslog
   ```

   要在 *sales* 数据库的所有分区上启动控制器，输入下列命令:

   ```
   db2gov START sales  salescfg saleslog
   ```
2. 要停止控制器，输入带 STOP 选项的 *db2gov* 命令。

   例如，要在 *sales* 数据库的所有分区上停止控制器，输入下列命令:

   ```
   db2gov STOP sales
   ```

   要仅在分区 3 上停止控制器，输入下列命令:

   ```
   db2gov START sales nodenum 3
   ```

相关概念:

- 第 241 页的『控制器实用程序』
- 第 243 页的『控制器守护进程』

相关参考:

- 『db2gov - DB2 Governor Command』（*Command Reference*）

控制器守护进程

当控制器守护进程启动时（当您通过 `db2gov` 实用程序执行它或当它唤醒时），它运行下列任务循环。

1. 它检查其控制器配置文件是否已更改或是否尚未读取。如果满足任何一个条件，则该守护进程将读取该文件中的规则。这允许您在控制器守护进程运行时更改其行为。
2. 它请求关于正在数据库工作的每个应用程序和代理进程的资源使用统计信息的快照信息。

   注: 在某些平台上，不能从“DB2® 监视器”获取 CPU 统计信息。在这种情况下，帐户规则和 CPU 限制不可用。
3. 它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。如果规则适用于应用程序，则控制器执行指定的操作。

   注: 控制器将累积的信息与配置文件中定义的值作比较。这表示如果使用应用程序可能已违反的新值来更新配置文件，则与该违规相关的控制器规则在下一个控制器时间间隔将立即应用于该应用程序。
4. 它对执行的任何操作在控制器日志文件中写入一条记录。

注: 如果 *agentpri* 数据库管理器配置参数不是系统缺省值，则不能使用控制器来调整代理进程优先级。（此注意事项不适用于 Windows® NT 平台。）

当控制器完成其任务后，它会在配置文件中指定的时间间隔内休眠。当经过此时间间隔之后，控制器唤醒并再次开始任务循环。

当控制器遇到错误或停止信号时，它会在结束之前先执行清除处理。通过使用已设置其优先级的应用程序的列表，清除处理复位所有应用程序代理进程的优先级。然后它复位不再在应用程序上运行的任何代理进程的优先级。这确保在控制器结束后，代理进程不会一直以非缺省优先级运行。如果发生错误，则控制器将一条消息写入管理通知日志，以指示它异常结束。

注: 尽管控制器守护进程不是数据库应用程序，并因此不维护与数据库的连接，但它确实具有实例连接。因为它可以发出快照请求，所以控制器守护进程可以检测数据库管理器何时结束。

相关概念:

- 第 241 页的『控制器实用程序』

相关任务:

- 第 242 页的『启动和停止控制器』

控制器配置

本节说明如何配置控制器来监视和控制数据库活动。

配置控制器

要配置“控制器”，创建配置文件，以确定“控制器”的实例监视的数据库以及它如何管理查询。

配置文件由一组规则组成。前三条规则指定要监视的数据库、写日志记录的时间间隔以及唤醒以进行监视的时间间隔。其余规则指定如何监视数据库服务器以及在特定情况下要采取的操作。

过程:

要创建“控制器”配置文件:

1. 在所有数据库管理器分区中的已安装或可用的目录中，用描述性名称创建 ASCII 文件。例如，监视 **sales** 数据库的控制器实例的配置文件可以称为 *govcfgsales*。
2. 在任何文本编辑器中打开该文件并输入配置信息和操作条件。

   用分号（;）结束每条规则。建议下列配置信息:
   - dbname: 要监视的数据库的名称或别名。
   - account: 分钟数，在该分钟数之后控制器实例将 CPU 使用情况统计信息写入其日志文件。此选项在 Windows NT 上不可用。
   - interval: 秒数，经过该秒数之后将唤醒控制器守护进程来监视活动。如果不指定时间间隔，则使用缺省值 120 秒。

   例如，配置文件中的头三条规则可能看起来如下:

   ```
   { Wake up once a second, the database name is sales,
    do accounting every 30 minutes.  }
   interval 1; dbname sales; account 30;
   ```

   添加一些规则，这些规则指定要监视的条件和当规则评估为真时要采取的操作。例如，可以添加这样一条规则，该规则将工作单元（UOW）可运行的时间量限制为 1 小时，之后将被强制与数据库断开连接，如下所示:

   ```
   setlimit uowtime 3600 action force;
   ```
3. 保存该文件。

相关概念:

- 第 244 页的『控制器配置文件』
- 第 247 页的『控制器规则元素』
- 第 250 页的『控制器配置文件示例』
- 第 251 页的『控制器日志文件』

相关参考:

- 『db2gov - DB2 Governor Command』（*Command Reference*）

控制器配置文件

当启动控制器时，您要指定包含规则的配置文件，这些规则控制对数据库运行的应用程序。控制器评估每条规则并当规则评估为真时执行指定的操作。

如果规则需求更改，则编辑配置文件而不必停止控制器。每个控制器守护进程检测到文件已更改，并重新读取文件。

必须在一个跨所有数据库分区安装的目录中创建配置文件，以便每个分区上的控制器守护进程可以读取同一配置文件。

配置文件由三个必需的规则组成，这些规则标识要监视的数据库、写日志记录的时间间隔以及控制器守护进程的休眠时间间隔。遵循这些参数，配置文件包含一组可选的应用程序监视规则和操作。下列注释适用于所有规则:

- 将注释定界在 { } 花括号内。
- 可用大写、小写或大小写混合字符来指定大多数项。应用程序名称是一个例外，指定为 `applname` 规则的一个自变量，它区分大小写。
- 每条规则以一个分号（;）结束。

必需的规则

下列规则指定要监视的数据库，以及守护进程在每个活动循环后唤醒的时间间隔。这些规则中的每一条在文件中仅指定一次。

dbname
要监视的数据库的名称或别名。

account *nnn*
编写帐户记录，该记录包含每个连接在指定的分钟数内的 CPU 使用统计信息。

注: 此选项在 Windows NT 环境中不可用。

如果一个短期连接会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标识来标识。

interval
守护进程唤醒的时间间隔，以秒计。如果不指定时间间隔，则使用缺省值 120 秒。

控制操作的规则

遵循所需要的规则，可以添加指定如何控制应用程序的规则。这些规则由称为规则子句的更小的组件组成。如果使用的话，必须以特定的顺序在规则语句中输入子句，如下所示:

1. **desc**（可选）：关于规则的注释，用引号括起
2. **time**（可选）：在评估规则时的当天时间
3. **authid**（可选）：应用程序在其下执行语句的一个或多个授权标识
4. **applname**（可选）：连接到数据库的可执行文件或对象文件的名称。此名称区分大小写。如果应用程序名中包含空格，则必须用双引号将应用程序名引起来。
5. **setlimit**: 控制器检查的限制。这些限制可以是几项限制（例如，CPU 时间、返回的行数或者空闲时间）的其中一项。

6. **action**（可选）：当达到限制时要执行的操作。如果未指定操作，则当达到限制时，控制器会将为应用程序工作的代理进程的优先级降低 10。对应用程序的操作可以包括降低其代理进程的优先级，强制它与数据库断开连接或为其操作设置调度选项。

通过在每条规则中仅使用每个子句一次，可以组合多个规则子句来形成一条规则，并用分号结束规则，如下列示例中所示:

```
desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;

desc "Slow down the use of db2 CLP by the novice user"
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;
```

如果有多条规则适用于某个应用程序，则应用所有这些规则。通常，与首先遇到的规则限制相关的操作是第一个要应用的操作。如果规则中的某个子句为 -1，则发生您指定的异常。在这种情况下，为后续规则中的子句指定的值只能覆盖先前为*相同*子句指定的值：先前规则中的其它子句仍有效。例如，一条规则使用 `rowssel 100000 uowtime 3600` 子句来指定如果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，则降低该应用程序的优先级。后续规则使用 `uowtime -1` 子句来指定同一应用程序可具有不受限制的经过时间。在这种情况下，如果应用程序运行 1 小时以上，不会更改其优先级。也就是说，`uowtime -1` 覆盖 `uowtime 3600`。但是，如果它选择多于 100 000 行，则由于 `rowssel 100000` 仍有效而会降低其优先级。

规则应用程序的顺序

控制器在配置文件中从文件顶部到文件底部来处理规则。但是，如果后面规则的 **setlimit** 子句比前面的规则松散，则仍会应用有更多限制的规则。例如，在下面的配置文件中，admin 将被限制为 5000 行而不考虑后面的规则，原因是第一条规则的限制性更强。

```
desc "Force anyone selecting 5000 or more rows"
setlimit rowssel 5000 action force;

desc "Allow user admin to select more rows"
authid admin
setlimit rowssel 10000 action force;
```

为了确保限制较少的规则能够覆盖文件中早些时间发生的限制较多的规则，可以指定 -1 选项以在应用新规则之前清除先前的规则。例如，在下面的配置文件中，初始规则将所有用户限制为使用 5000 行。第二条规则清除对 admin 的这一限制，第三条规则将对 admin 的限制重设为 10000 行。

```
desc "Force anyone selecting 5000 or more rows"
setlimit rowssel 5000 action force;

desc "Clear the rowssel limit for admin"
authid admin
setlimit rowssel -1;

desc "Now set the higher rowssel limit for admin"
authid admin
setlimit rowssel 10000 action force;
```

相关概念:

- 第 247 页的『控制器规则元素』
- 第 250 页的『控制器配置文件示例』

控制器规则元素

控制器配置文件中的每条规则都由一些子句组成，这些子句指定应用规则的条件和如果规则评估为真而导致的操作。这些子句必须按显示的顺序指定。在子句描述中，[ ] 指示可选子句。

可选的开始元素

[desc] 指定该规则的文本描述。必须用单引号或双引号将该描述引起来。

[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标识 (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 (,) 将多个应用程序名隔开，例如，`applname db2bp, batch, 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. 数据库管理器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。

限制子句

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 和 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 环境中不可用。

locks *nnn*

指定一个应用程序可挂起的锁定数。如果您指定 -1，则控制器不限制应用程序挂起的锁定数。

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（数据库系统监视器开关）API 来取消激活工作单元开关，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。控制器使用监视器来收集有关该系统的信息。如果您在数据库管理器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。

操作子句

[action]

指定当超过一个或多个指定的限制时要采取的操作。您可以指定下面的操作。

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进程的优先级降低 10。

nice *nnn*

指定为应用程序工作的代理进程的优先级更改。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* 数据库管理器参数设置为缺省值；否则，它将覆盖 priority 子句。
- 在 Windows 平台上，可以将 *agentpri* 数据库管理器参数与*优先级*操作一起使用。

force 指定强制为该应用程序服务的代理进程。（发出 FORCE APPLICATION 以终止该协调代理进程。）

schedule [class]

调度可改善执行应用程序的代理进程的优先级，目标是将平均响应时间减至最低，同时维护所有应用程序之间的公平性。

控制器根据下列三个标准选择用于调度的顶部几个应用程序:

- 挂起大多数锁定的应用程序

  此选项尝试减小锁定等待数。
- 最旧的应用程序
- 具有最短估计剩余运行时间的应用程序。

此选项尝试允许在时间间隔内完成尽可能多的短时间活动的语句。

给予每个标准中的顶部三个应用程序比所有其它应用程序更高的优先级，也就是说，给予每个条件组中的顶部应用程序最高的优先级，给予次高应用程序第二高的优先级，以及给予排第三位的应用程序第三高的优先级。如果将单个应用程序排在多个标准的前三位，则给予它对排在最高位的条件的适当优先级，而给予次高应用程序对其它标准的次高优先级。 例如，如果应用程序 A 挂起最多的锁定，但具有第三少的估计剩余运行时间，则给予它第一个条件的最高优先级，而给予具有最短估计剩余运行时间的排第四位的应用程序对该条件的第三高的优先级。

此控制器规则选择的应用程序最多分为三类。对于每一类，控制器根据以上列示的标准选择 9 个应用程序，它们是每个类中的顶部三个应用程序。如果指定类选项，此规则所选择的所有应用程序被看作单个类，并且选择 9 个应用程序并给予它们较高的优先级，如上所述。

如果在多个控制器规则中选择了某个应用程序，则由选择了该应用程序的最后一个规则控制该应用程序。

注: 如果使用了 sqlmon（数据库系统监视器开关）API 来取消激活该语句开关，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。控制器使用监视器来收集有关该系统的信息。如果您在数据库管理器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 14 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它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。

  如果操作中未包括类选项，则控制器会根据调度操作下有多少个活动应用程序来创建它自己的类，并根据 DB2 查询编译器为该应用程序运行的查询所作的成本估计，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进程的优先级。

相关概念:

- 第 244 页的『控制器配置文件』
- 第 250 页的『控制器配置文件示例』

相关任务:

- 第 244 页的『配置控制器』

控制器配置文件示例

以下示例显示一个控制器配置文件，它设置几条带操作的规则:

```
{ 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;
```

相关概念:

- 第 244 页的『控制器配置文件』
- 第 247 页的『控制器规则元素』

相关任务:

- 第 244 页的『配置控制器』

控制器日志文件使用

本节描述控制器日志文件并说明如何查询它们来检索信息。

控制器日志文件

每当控制器守护进程执行操作时，它将一条记录写入它的日志文件。操作包括以下各项:

- 强制应用程序
- 读取控制器配置文件
- 更改应用程序优先级
- 遇到错误或警告
- 启动或结束

每个控制器守护进程都有一个单独的日志文件中。独立的日志文件防止文件锁定瓶颈，此瓶颈可能会在许多控制器守护进程同时写入同一个文件时产生。要将日志文件合并在一起并查询它们，使用 db2govlg 实用程序。

日志文件存储在 sqllib 目录的 log 子目录中，除了在 Windows® NT 上以外，log 子目录位于实例目录下面。当通过 db2gov 命令启动控制器时，应提供该日志文件的基本名称。确保日志文件名称包含数据库名称，以区分受控制的每个数据库的每个分区上日志文件。为了确保文件名对于分区数据库环境中的每个控制器是唯一的，自动将控制器守护进程运行所在的分区号追加至日志文件名。

日志文件记录格式

日志文件中的每个记录具有下列格式:

```
Date Time NodeNum RecType Message
```

注: *Date* 和 *Time* 字段的格式为 yyyy-mm-dd hh.mm.ss。可以通过对此字段进行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: 启动了控制器
- STOP: 停止了控制器
- FORCE: 强制了应用程序
- NICE: 更改了应用程序的优先级
- ERROR: 发生错误
- WARNING: 发生警告
- READCFG: 控制器读取了配置文件
- ACCOUNT: 应用程序记帐统计信息。
- SCHEDGRP: 发生了代理进程优先级更改。

下面更详细地描述了这些值中的某一些。

START
START 记录是在控制器启动时编写的。它具有以下格式:

```
Database = <database_name>
```

STOP STOP 记录是在控制器停止时编写的。它具有以下格式:

```
Database = <database_name>
```

FORCE
每当控制器确定根据控制器配置文件中的规则需要强制使用应用程序时，就会编写 FORCE 记录。FORCE 记录具有以下格式:

```
<appl_name> <auth_id> <appl_id> <coord_partition> <cfg_line>
<restriction_exceeded>
```

其中:

<coord_partition>
指定应用程序的协调分区的号码。

<cfg_line>
指定控制器配置文件中导致强制使用应用程序的规则所在的行号。

<restriction_exceeded>
提供有关超出规则的情况的详细信息。这可包含下列值:
- CPU: 总应用程序 USR cpu 加 SYS cpu 时间（以秒计）
- Locks: 应用程序持有的总锁定数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序空闲的时间
- ET（经过时间）: 自从启动应用程序的当前工作单元以来的经过时间（超过了 uowtime setlimit）

NICE NICE 记录是在通过控制器配置文件中的指定的优先级操作更改应用程序的优先级时编写的。NICE 记录具有以下格式:

```
<appl_name> <auth_id> <appl_id> <nice value> (<cfg_line>)
<restriction_exceeded>
```

其中:

<nice value>
指定将对应用程序的代理进程的优先级值所作的增大（或减小）。

<cfg_line>
指定控制器配置文件中导致更改应用程序优先级的规则所在的行号。

<restriction_exceeded>
提供有关超出规则的情况的详细信息。这可包含下列值:
- CPU: 总应用程序 USR cpu 加 SYS cpu 时间（以秒计）
- Locks: 应用程序持有的总锁定数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序空闲的时间
- ET（经过时间）: 自从启动应用程序的当前工作单元以来的经过时间（超过了 uowtime setlimit）

ERROR
ERROR 记录是在控制器守护进程需要关闭时编写的。

WARNING
WARNING 记录在下列情况下会写入控制器日志:
- 调用 sqlefrce API 以强制使用应用程序，但该 API 返回正的 SQLCODE。
- 快照调用返回并非 1611（“SQL1661 不返回任何数据”）的正的 SQLCODE。
- 快照调用返回并非 −1224（“SQL 1224N 不能启动数据库代理进程来处理请求，或者数据库代理进程因数据库系统关闭或强制命令而终止”）或 −1032（“SQL1032N 没有发出任何启动应用程序管理器命令”）的负的 SQLCODE。这些返回码在关闭先前活动的示例后发生。
- 在 UNIX® 环境中，尝试安装信号处理程序，但尝试失败。

ACCOUNT
ACCOUNT 记录在下列情况下写入控制器日志:

- 自上次为此应用程序编写 ACCOUNT 记录以来，此应用程序的 agent_usr_cpu 或 agent_sys_cpu 的值已更改。
- 应用程序不再活动

ACCOUNT 记录具有以下格式:

<auth_id> <appl_id> <applname> <connect time> <agent_usr_cpu delta> <agent_sys_cpu delta>

SCHEDGRP

SCHEDGRP 记录是在下列情况下编写的:

- 将应用程序添加至调度组
- 将应用程序从一个调度组移至另一个调度组。

SCHEDGRP 记录具有以下格式:

`<appl_name> <auth_id> <appl_id> <cfg_line> <restriction_exceeded>`

其中:

<cfg_line>

指定控制器配置文件中导致调度应用程序的规则所在的行号。

<restriction_exceeded>

提供有关超出规则的情况的详细信息。这可包含下列值:

- CPU: 总应用程序 USR cpu 加 SYS cpu 时间（以秒计）
- Locks: 应用程序持有的总锁定数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序空闲的时间
- ET（经过时间）: 自从启动应用程序的当前工作单元以来的经过时间（超过了 uowtime setlimit）

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* 字段提供其它非标准信息，这些信息根据 *RecType* 字段中的值的不同而有所变化。例如，FORCE 或 NICE 记录指示 *Message* 字段中的应用程序信息，而 ERROR 记录包括错误消息。

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
```

相关概念:

- 第 241 页的『控制器实用程序』
- 第 255 页的『控制器日志文件查询』

控制器日志文件查询

每个控制器守护进程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，来列示单一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:

```
>>—db2govlg—log-file——┬────────────────────┬──┬──────────────────────────┬————————><
                      └─nodenum—node-num─┘    └─rectype—record-type─┘
```

图 25. 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。

相关概念:

- 第 241 页的『控制器实用程序』
- 第 251 页的『控制器日志文件』

第 10 章 调整配置

本章描述如何才能管理数据库容量（主要通过添加和删除数据库分区进行）。增大容量的其它方法包括添加 CPU 和添加内存。

数据库服务器容量的管理

如果数据库管理器的容量不能满足目前或将来的需求，则可以采用下列方法扩充其容量:

- 添加磁盘空间并创建附加容器。
- 添加内存。

如果这些简单措施不能添加所需的容量，请考虑以下方法:

- 添加处理器。

  如果具有单个处理器的单一分区配置已被最大限度地使用，则可能需要添加处理器或分区。添加处理器可以获得更大的处理能力。在 SMP 系统中，多个处理器共享内存和存储系统资源。由于所有处理器均在一个系统内，因此不存在附加的开销费用（如铺设系统间的通信线路），也不会增加系统间的协调任务。DB2® 中的实用程序（例如，装入、备份和复原）可利用其它处理器。DB2 Universal Database™（DB2 通用数据库）支持此环境。

  注: 某些操作系统（例如，Solaris Operating Environment）可以动态地使处理器联机和脱机。

  如果添加了处理器，请检查并修改那些决定处理器使用数目的数据库配置参数。下列数据库配置参数决定处理器的使用数目，可能需要进行更新:

  – 缺省级别（dft_degree）

  – 最大并行度（max_querydegree）

  – 启用分区内并行性（intra_parallel）

  此外，还应评估那些决定应用程序如何执行并行处理的参数。

  在使用 TCP/IP 进行通信的环境中，应检查 DB2TCPCONNMGRS 注册表变量的值。

- 添加物理节点。

  如果数据库管理器当前已被分区，则可以通过添加不同的单处理器物理节点或多处理器物理节点来增加数据存储空间和处理能力。每个节点上的内存和存储系统资源不与其它节点共享。虽然添加节点可能会导致通信和任务协调问题，但这一选择所带来的好处是可在多个系统上平衡数据和用户存取。“DB2 通用数据库”支持此环境。

  无论数据库管理器系统处于运行状态还是停止状态，均可添加节点。但是，如果在系统运行时添加节点，则必须在停止并重新启动系统后，才能将数据库移植到新添的节点。

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，不能删除数据库或创建使用新分区的数据库。

相关概念:

- 第 258 页的『分区数据库中的分区』

分区数据库中的分区

可在分区数据库系统运行或停止时，向其添加数据库分区。因为添加新服务器可能要花费较长时间，因此应在数据库管理器已经运行时来添加。

使用 ADD DBPARTITIONNUM 命令来将数据库分区添加到系统。可以按下列方式调用此命令:

- 作为 db2start 上的一个选项
- 用命令行处理器 ADD DBPARTITIONNUM 命令
- 用 API 功能 sqleaddn
- 用 API 功能 sqlepstart

如果您的系统已停止，使用 db2start。如果系统在运行，可以使用任何其它选项。

当使用 ADD DBPARTITIONNUM 命令将一个新数据库分区添加至系统时，会将实例中的所有现有的数据库扩充到新数据库分区。还可指定要将临时表空间的哪些容器用于数据库。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。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之后，才能使用数据库。

在改变一个或多个数据库分区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。

不能简单地通过对系统添加分区来从单一分区系统更改为多分区系统。这是因为跨分区再分发数据需要每个受影响的表上的分区键。分区键是在多分区环境中创建表时自动生成的。在单一分区环境中，可使用 CREATE TABLE 或 ALTER TABLE SQL 语句显式地创建分区键。

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“企业服务器版”，则编辑 db2nodes.cfg 文件以添加新的数据库分区定义；不要使用所述的任何一个过程，因为它们只有在存在数据库时才适用。

Windows NT 注意事项: 如果在 Windows NT 上使用“企业服务器版”，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。 但是，如果您已有数据库，使用 DB2START ADDNODE 命令，以确保在调整系统时为每个现有的数据库创建一个数据库分区。 在 Windows NT 上，永远都不应手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。

相关任务:

- 第 259 页的『将分区添加到运行的数据库系统』
- 第 260 页的『将分区添加到 Windows NT 上已停止的数据库系统』
- 第 264 页的『删除数据库分区』

将分区添加到运行的数据库系统

当分区数据库系统正在运行且应用程序与数据库已连接时，可将新数据库分区添加至该系统。但是，只有将数据库管理器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。

过程:

要将数据库分区添加到运行的数据库管理器:

1. 在任何现有的数据库分区上，运行 DB2START 命令。

   在所有平台上，为 DBPARTITIONNUM、ADD DBPARTITIONNUM、HOSTNAME、PORT 和 NETNAME 参数指定新分区值。在 Windows NT 平台上，也指定 COMPUTER、USER 和 PASSWORD 参数。

   您还可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临时表空间容器的定义。

   当完成 DB2START 命令时，停止新服务器。
2. 通过运行 DB2STOP 命令来停止所有分区上的数据库管理器。

   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。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 DBPARTITIONNUM 命令（当对 DB2START 命令指定 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，停止新服务器分区。
3. 通过运行 DB2START 命令来启动数据库管理器。

   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，如创建或删除数据库。

   注: 可能必须对所有数据库分区服务器发出 DB2START 命令两次，才能存取新的 `db2nodes.cfg` 文件。
4. 在新数据库分区上备份所有数据库。（可选）
5. 将数据再分发至该新数据库分区。（可选）

相关概念:

- 第 258 页的『分区数据库中的分区』

相关任务:

- 第 260 页的『将分区添加到 Windows NT 上已停止的数据库系统』
- 第 261 页的『将分区添加到 UNIX 上已停止的数据库系统』

将分区添加到 Windows NT 上已停止的数据库系统

当分区数据库系统停止时，可以将新数据库分区添加至其中。当再次启动数据库管理器时，新添加的数据库分区可用于所有数据库。

先决条件:

必须安装新服务器，才可以在该服务器上创建分区。

过程:

要将分区添加到已停止的分区数据库服务器:

1. 发出 DB2STOP 来停止所有的数据库分区。
2. 在新服务器上运行 ADD DBPARTITIONNUM 命令。

   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。更新数据库配置参数值，以便与其它数据库分区上的值相匹配。
3. 运行 DB2START 命令启动数据库系统。注意，新服务器安装期间，节点配置文件（cfg）已更新为包括新服务器。
4. 将新分区的配置文件更新为如下所示:
   a. 在任何现有的数据库分区上，运行 DB2START 命令。

      为 DBPARTITIONNUM、ADDDB2PARTITIONNUM、HOSTNAME、PORT 和 NETNAME 参数以及 COMPUTER、USER 和 PASSWORD 参数指定新分区值。

      您还可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临时表空间容器的定义。

      当完成 DB2START 命令时，停止新服务器。
   b. 通过运行 DB2STOP 命令来停止整个数据库管理器。

      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。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 DB2PARTITIONNUM 命令（当对 DB2START 命令指定 ADDDB2PARTITIONNUM 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，停止新服务器分区。
5. 通过运行 DB2START 命令来启动数据库管理器。

   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，如创建或删除数据库。

   注: 可能必须对所有数据库分区服务器发出 DB2START 命令两次，才能存取新的 `db2nodes.cfg` 文件。
6. 在新数据库分区上备份所有数据库。（可选）
7. 将数据再分发至该新数据库分区。（可选）

相关概念:

- 第 258 页的『分区数据库中的分区』
- 第 262 页的『添加节点错误恢复』

相关任务:

- 第 259 页的『将分区添加到运行的数据库系统』
- 第 261 页的『将分区添加到 UNIX 上已停止的数据库系统』

将分区添加到 UNIX 上已停止的数据库系统

当分区数据库系统停止时，可以将新数据库分区添加至其中。当再次启动数据库管理器时，新添加的数据库分区可用于所有数据库。

先决条件:

如果新服务器不存在，必须安装该服务器，包括下列任务:

- 使可执行文件可存取（使用共享文件系统安装或本地副本）
- 使操作系统文件与现有处理器上的操作系统文件同步。
- 确保 `sqllib` 目录可作为共享文件系统存取。
- 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。

向名称服务器注册该主机名，或者在所有数据库分区的 etc 目录中的主机文件中注册该主机名。

过程:

要将分区添加到已停止的分区数据库服务器:

1. 发出 DB2STOP 来停止所有的数据库分区。
2. 在新服务器上运行 ADD DB2PARTITIONNUM 命令。

   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。更新数据库配置参数值，以便与其它数据库分区上的值相匹配。
3. 运行 DB2START 命令启动数据库系统。注意，新服务器安装期间，节点配置文件（cfg）已更新为包括新服务器。
4. 将新分区的配置文件更新为如下所示:
   a. 在任何现有的数据库分区上，运行 DB2START 命令。

      为 DB2PARTITIONNUM、ADDDB2PARTITIONNUM、HOSTNAME、PORT 和 NETNAME 参数以及 COMPUTER、USER 和 PASSWORD 参数指定新分区值。

      您还可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临时表空间容器的定义。

      当完成 DB2START 命令时，停止新服务器。
   b. 通过运行 DB2STOP 命令来停止整个数据库管理器。

      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。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 DB2PARTITIONNUM 命令（当对 DB2START 命令指定 ADDDB2PARTITIONNUM 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，停止新服务器分区。

5. 通过运行 DB2START 命令来启动数据库管理器。

   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，如创建或删除数据库。

   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。
6. 在新数据库分区上备份所有数据库。（可选）
7. 将数据再分发至该新数据库分区。（可选）

也可以手工更新配置文件，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. 发出下列命令来启动新节点: `DB2START DB2PARTITIONNUM partitionnum`

   将您分配给该新数据库分区的编号指定为 nodenum 的值。
3. 如果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 DB2PARTITIONNUM 注册表变量。指定要添加的数据库的号码。
4. 在新数据库分区上运行 ADD NODE 命令。

   此命令还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。更新数据库配置参数值，以便与其它数据库分区上的值相匹配。
5. 当 ADD DB2PARTITIONNUM 命令完成时，发出 DB2START 命令启动系统中的其它数据库分区。

   在成功启动所有数据库分区之前，不要执行任何系统范围内的活动，如创建或删除数据库。

相关概念:

• 第 262 页的『添加节点错误恢复』

相关任务:

• 第 259 页的『将分区添加到运行的数据库系统』
• 第 260 页的『将分区添加到 Windows NT 上已停止的数据库系统』
• 第 264 页的『删除数据库分区』

添加节点错误恢复

因为在版本 8.1 和更高级别版本中，DB2® 创建“隐藏”缓冲池以对所有缓冲池页大小提供缺省自动支持，由于不存在的缓冲池，节点添加不会失败。但是，因为隐藏的缓冲池非常小，所以如果使用其中一个“隐藏”缓冲池，可能会严重影响性能。如果使用隐藏的缓冲池，管理通知记录中写入一条消息。

在下列环境中的节点添加情景说明中使用隐藏的缓冲池:

- 将节点添加到分区数据库，该数据库具与缺省值 4KB 不同的页大小的一个或多个系统临时表空间。当创建节点时，仅 IBMDEFAULTDP 缓冲池存在，此缓冲池页大小为 4 KB。

  考虑下列示例:

  1. 使用 db2start 命令来将节点添加到当前分区数据库:

     ```
     DB2START DB2PARTITIONNUM 2 ADD DB2PARTITIONNUM HOSTNAME newhost PORT 2
     ```

  2. 在使用新的节点描述手工更新 db2nodes.cfg 文件之后，使用 ADD DB2PARTITIONNUM 命令。

  防止这些问题发生的一种方法是在 ADD NODE 或 **db2start** 命令上指定 WITHOUT TABLESPACES 子句。此后，需要使用 CREATE BUFFERPOOL 语句来创建要使用的缓冲池，并使用 ALTER TABLESPACE 语句将系统临时表空间与缓冲池相关联。

- 将节点添加到现有的数据库分区组，该数据库分区组具与缺省页大小 4KB 不同的页大小的一个或多个表空间。发生此情况的原因是尚未对表空间激活在新节点上创建的非缺省页大小缓冲池。

  注: 在先前版本的 DB2 中，此命令使用了 NODEGROUP 关键字，而不是使用 DATABASE PARTITION GROUP 关键字。

  考虑下列示例:

  – 可以使用 ALTER DATABASE PARTITION GROUP 语句来向数据库分区组添加节点，如下所示:

    ```
    DB2START
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

    防止此问题发生的一种方法是在发出 ALTER DATABASE PARTITION GROUP 语句之前，为每个页大小创建缓冲池，然后重新连接至数据库:

    ```
    DB2START
    CONNECT TO mpp1
    CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
    CONNECT RESET
    CONNECT TO mpp1
    ALTER DATABASE PARTITION GROUP ng1 ADD NODE (2)
    ```

注: 如果数据库分区组具有页大小为缺省值的表空间，则返回下列消息:

```
SQL1759W 需要再分发节点组来更改
         节点组“ng1”中的对象的数据定位，以包括某些
         添加的节点或排除某些删除节点。
```

相关任务:

- 第 259 页的『将分区添加到运行的数据库系统』
- 第 260 页的『将分区添加到 Windows NT 上已停止的数据库系统』

删除数据库分区

您可以删除任何数据库都未在使用的数据库分区，并为其它的用户空出计算机。

先决条件:

发出 DROP NODE VERIFY 命令或 *sqledrpn* API 来验证此分区是否未在使用中。

- 如果接收到消息 SQL6034W（没有数据库在使用该节点），可以删除该分区。
- 如果接收到消息 SQL6035W（节点正在使用），使用 REDISTRIBUTE NODEGROUP 命令，将正在删除的数据库分区的数据再分发到数据库别名的其它数据库分区。

还应确保所有事务（此数据库分区是其协调程序）已成功落实或回滚。这可能需要在其它服务器上执行崩溃恢复。例如，如果删除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删除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。

过程:

要删除数据库分区:

1. 发出带 DROP NODENUM 参数的 DB2STOP 命令来删除该数据库分区。在命令成功完成后，停止系统。
2. 用 DB2START 命令启动数据库管理器。

相关概念:

- 第 257 页的『数据库服务器容量的管理』
- 第 258 页的『分区数据库中的分区』

第 11 章 将数据再分发至数据库分区

本章提供有关确定何时将数据再分发至分区、如何执行再分发和如何从再分发错误恢复的信息。

数据再分发

要在分区数据库中的各分区中再分发表数据，使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令。

注： 在先前版本的 DB2® 中，此命令使用了 NODEGROUP 关键字，而不是 DATABASE PARTITION GROUP 关键字。

在分区数据库中，可能由于下列原因而再分发数据:

- 为了在数据库分区之间平衡数据量和处理负载。

  如果可以将数据存取分布在多个分区上，则性能会提高。

- 为了在跨数据库分区的数据分发中引入数据变形。

  如果再分发频繁存取的表中的数据，以便不经常存取的数据位于数据库分区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，则存取和吞吐量性能可能会提高。这将提高最频繁运行的应用程序的存取性能和吞吐量。

要保留表并置，使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令来在数据库分区组级别上再分发数据。在单个操作中再分发所有表。要完成指定的数据分发，REDISTRIBUTE DATABASE PARTITION GROUP 命令当它移动行时将表分到各个数据库分区上。根据您指定的选项，实用程序可生成目标分区映射或将一个现有的分区映射用作输入。

如何在数据库分区之间再分发数据

在数据库的指定数据库分区组中的一组表上执行数据再分发。在执行 REDISTRIBUTE DATABASE PARTITION GROUP 命令来调用“数据再分发”实用程序之前，必须连接到目录数据库分区的数据库。该实用程序使用源分区映射和目标分区映射来标识已将哪些散列分区分配到新位置，它是新的数据库分区号。与具有新位置的一个分区对应的所有行从源分区映射中指定的数据库分区移动到目标分区映射中指定的数据库分区。

“数据再分发”实用程序执行下列步骤:

1. 获取目标分区映射的新分区映射标识，并将它插入 SYSCAT.PARTITIONMAPS 目录视图。
2. 使用新的分区映射标识更新数据库分区组的 SYSCAT.DBPARTITIONGROUPS 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.DBPARTITIONGROUPDEF 目录视图。
4. 对要删除的任何数据库分区，将 SYSCAT.DBPARTITIONGROUPDEF 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. 对于数据库分区组中的每个表，按下列步骤再分发每个表中的数据:
   a. 在 SYSTABLES 目录表中锁定关于该表的那一行。
   b. 使涉及此表的所有程序包失效。因为再分发表行，所以与该表关联的分区映射标识更改。因为程序包无效，编译器必须获得该表的新的分区信息，并相应地生成程序包。
   c. 以互斥方式锁定该表。
   d. 使用 DELETE 和 INSERT 来再分发表中的数据。
   e. 如果再分发操作成功，则它对表发出 COMMIT 并对数据库分区组中的下一个表继续。如果在完全再分发表之前该操作失败，则实用程序对表的更新发出 ROLLBACK，结束整个再分发操作并返回一个错误。
8. 对于先前标记为要删除的数据库分区，删除数据库文件并删除 SYSCAT.NODEGROUPS 目录视图中的条目。
9. 更新 SYSCAT.NODEGROUPS 目录视图中的数据库分区组记录，以将 PMAP_ID 设置为 REBALANCE_PMAP_ID 的值，并将 REBALANCE_PMAP_ID 设置为空。
10. 从 SYSCAT.NODEGROUPS 目录视图中删除旧的分区映射。
11. 对所有更改执行 COMMIT。

相关概念:

- 第 268 页的『数据再分发的日志空间需求』
- 第 269 页的『再分发错误恢复』

相关任务:

- 第 267 页的『在分区之间再分发数据』
- 第 266 页的『确定是否要再分发数据』

确定是否要再分发数据

在决定要重新分配数据之前，查明数据在分区之间是否分布不均匀。在有了当前分布信息之后，可以使用此信息来创建定制再分发文件或分区映射。

过程:

要获取关于数据库分区组中分区的当前数据分布的信息:

1. 确定是否有任何数据库分区具有不相等的行分布。

   对于最大的表，使用适当的分区列并输入如下的查询:

```
SELECT PARTITION(column_name), COUNT(*) FROM table_name
   GROUP BY PARTITION(column_name)
   ORDER BY PARTITION(column_name) DESC
FETCH FIRST 100 ROWS ONLY
```

   PARTITION 和 DBPARTITIONNUM SQL 函数确定散列分区或数据库分区上的当前数据分布。PARTITION 函数返回表中每行的分区映射索引。DBPARTITIONNUM 函数返回行的分区号。

2. 对在数据库分区组上分区的其它大表执行此查询。

3. 使用此信息来创建一个分布文件和一个目标分区映射。

注: 也可以使用带 ANALYZE 选项的 AutoLoader 实用程序来创建数据分布文件。可将此文件用作“数据再分发”实用程序的输入。

相关概念:

- 第 265 页的『数据再分发』
- 第 268 页的『数据再分发的日志空间需求』

相关任务:

- 第 267 页的『在分区之间再分发数据』

在分区之间再分发数据

在分区数据库中，在下列情况下可以在分区之间再分发数据以平衡数据存取:

- 当某些分区包含的数据比其它分区多时
- 当存取某些分区比存取其它分区频繁时

先决条件:

日志文件大小: 确保日志文件对于数据再分发操作足够大。每个受影响的分区上的日志文件必须足够大以满足在那里执行的 INSERT 和 DELETE 操作的需要。

重复的具体查询表: 如果数据库分区组中的数据包含重复的具体查询表，则在再分发数据之前必须删除这些表。再分发数据之后，可以重新创建具体查询表。

限制:

当该实用程序正在运行时，您可对数据库分区组的对象执行下列操作。但是，不能在要再分发的表上执行。您可以:

- 在其它表上创建索引。CREATE INDEX 语句使用受影响的表的分区映射。
- 删除其它表。DROP TABLE 语句使用受影响的表的分区映射。
- 删除其它表上的索引。DROP INDEX 语句使用受影响的表的分区映射。
- 查询其它表。
- 更新其它表。
- 在数据库分区组中定义的表空间中创建新表。CREATE TABLE 语句使用目标分区映射。
- 在数据库分区组中创建表空间。

当该实用程序正在运行时，不能执行下列操作:

- 对数据库分区组启动另一个再分发操作
- 对数据库分区组中任何一个表执行 ALTER TABLE 语句
- 删除数据库分区组
- 改变数据库分区组。

除非所有受影响的表都有分区键，否则不能使用此过程来在将分区添加到单一分区系统后再分发数据。REDISTRIBUTE DATABASE PARTITION GROUP 命令依赖于分区键来再分发数据。分区键是在多分区数据库分区组中创建表时自动生成的，也可使用

CREATE TABLE 或 ALTER TABLE SQL 语句显式地定义它。如果表是在单一分区分区组中创建的，并且没有在 CREATE TABLE SQL 语句中定义分区键，则不会定义分区键。在再分发数据之前，必须使用 ALTER TABLE SQL 语句来为每个受影响的表创建分区键。

过程:

要在数据库分区组中的分区之间再分发数据:

1. 连接至包含系统目录表的数据库分区。
2. 执行先决条件任务（如果有必要的话）。
3. 发出 REDISTRIBUTE DATABASE PARTITION GROUP 命令。

   注: 在先前版本的 DB2 中，此命令使用了 NODEGROUP 关键字，而不是使用 DATABASE PARTITION GROUP 关键字。

   指定下列自变量:

   数据库分区组名
   必须指定要在其中再分发数据的数据库分区组。

   UNIFORM
   如果数据已均匀分布且要保持均匀分布，则指定 UNIFORM 或省略任何分布类型的自变量。UNIFORM 是缺省值。

   USING DISTFILE distfile-name
   要指定改正或创建数据变形的定制分布，包括分布文件名。“再分发数据”实用程序使用此文件来构造目标分区映射。

   USING TARGETMAP targetmap-name
   “再分发数据”实用程序直接使用指定的目标映射。

   有关详细信息，参阅 REDISTRIBUTE DATABASE PARTITION GROUP 命令行实用程序信息。
4. 再分发完成之后:
   - 重新创建在再分发之前删除的任何重复的具体查询表。
   - 执行 RUNSTATS 命令来收集数据分布统计信息，以便 SQL 编译器和优化器在为查询选择数据存取方案时使用。

注: “说明”表包含关于用来再分发数据的分区映射的信息。

相关概念:

- 第 265 页的『数据再分发』
- 第 268 页的『数据再分发的日志空间需求』
- 第 269 页的『再分发错误恢复』

相关任务:

- 第 266 页的『确定是否要再分发数据』

数据再分发的日志空间需求

在分区上再分发数据之前，考虑日志空间需求。

日志的大小必须足以满足再分发数据的每个数据库分区上的 INSERT 和 DELETE 操作的需要。最高的日志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。

如果要移动到大量数据库分区，则使用当前数据库分区数与新数据库分区数的比率来估计 INSERT 和 DELETE 操作的数目。例如，考虑再分发在再分发之前均匀分布的数据。如果从 4 个数据库分区移动至 5 个数据库分区，则 4 个原始数据库分区的大约 20% 的数据将移动至新数据库分区。这表示 DELETE 操作的 20% 将在 4 个原始数据库分区中的每个分区执行，所有 INSERT 操作将在新数据库分区上执行。

考虑非均匀分布的数据，如分区键包含许多空值的情况。在这种情况下，分区键中包含空值的所有行从旧分区方案下的一个数据库分区移动至新分区方案下的另一个数据库分区。结果，那两个数据库分区上所需要的日志空间量会增加，并且很可能会大大超过假设均匀分布时所计算出的数量。

每个表的再分发是单一事务。因此，当估计日志空间时将最大表的大小乘以更改百分比（如 20%）。但是，考虑最大的表可能是均匀分布的，但次大的表（只是举个例子）可能带有一个或多个扩大的数据库分区。在这种情况下，考虑使用非均匀分布的表，而不使用最大的那个表。

注: 在估计出要在一个数据库分区上插入或删除的最大数据量之后，将该估计数加倍来确定活动日志的峰值大小。如果此估计大于活动日志限制（即 256 GB），则必须分步骤完成数据再分发。使用“makepmap”实用程序来生成一系列目标分区映射，每个步骤生成一个映射。也可以将 *logsecond* 数据库配置参数设置为 `-1`，以避免大多数日志空间问题。

相关概念:

- 第 265 页的『数据再分发』

再分发错误恢复

当再分发操作开始执行后，要将一个文件写入 `sqllib` 目录的 `redist` 子目录中。此状态文件列示在数据库分区上执行的任何操作、再分发的表的名称以及该操作的完成状态。如果不能再分发一个表，则在该文件中将列示其名称和适当的 SQLCODE。如果由于不正确的输入参数而使再分发操作不能开始，则不写入该文件，而返回一个 SQLCODE。

此文件有下列命名约定:

对于 UNIX® 平台:
  databasename.database partition groupname.timestamp
对于非 UNIX 平台:
  databasename\database partition groupname\date\time

注: 在非 UNIX 平台上，仅使用数据库分区组名的前八（8）个字节。

如果数据再分发操作失败，则某些表可能要再分发，而其它表不用。这是因为数据再分发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，再分发其余的表。
- 使用 ROLLBACK 选项来撤销该再分发，并将已再分发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再分发操作所花的时间相同。

在可以使用任何一个选项前，先前的数据再分发操作必须已经失败，这样才能把 SYSCAT.DBPARTITIONGROUPS 表中的 REBALANCE_PMID 列设置为非空值。

如果碰巧错误地删除了此状态文件，仍可尝试 CONTINUE 操作。

相关概念:

- 第 265 页的『数据再分发』

相关任务:

- 第 267 页的『在分区之间再分发数据』

再分发存储过程和函数

按步骤的再分发存储过程可用来安全地在许多步骤中再分发数据库分区组:

1. 分析与日志空间可用性和数据变形有关的数据库分区组。
2. 为给定的表创建数据分发文件。
3. 为数据库分区组创建并报告按步骤的再分发计划的内容。
4. 按照计划再分发数据库分区组。

当在以下过程中使用单个参数时，当不能获得参数值时将“-1”用作参数的输出值。

注: 再分发存储过程和函数仅在分区数据库中运行，在该数据库中为每个表定义了分区键。

get_swrd_settings 存储过程

get_swrd_settings 函数读取给定数据库分区组的现有的再分发注册表记录。

表 31. get_swrd_settings 输入参数

名称	数据类型	描述
dbpgName	VARCHAR(128)	数据库分区组名，针对它来运行再分发进程。
matchingSpec	SMALLINT	表 32 中的一位宽度的字段标识，指示由第 271 页的表 33 中的输出参数返回的目标字段。可以将那些不需要的输出参数设置为空。 例如，如果 matchingSpec 设置为 96，这是 (REDIST_STAGE_SIZE | REDIST_NEXT_STEP) 的整数值，则此函数的调用程序仅需要提供 stageSize 和 nextStep 就可以接收值，余下的输出参数可为空。

表 32. 一位宽度的字段标识

字段名	十六进制值	十进制值
REDIST_METHOD	0x0001<<0	1
REDIST_PMAP_FILE	0x0001<<1	2
REDIST_DIST_FILE	0x0001<<2	4
REDIST_STEP_SIZE	0x0001<<3	8
REDIST_NUM_STEPS	0x0001<<4	16
REDIST_STAGE_SIZE	0x0001<<5	32
REDIST_NEXT_STEP	0x0001<<6	64
REDIST_PROCESS_STATE	0x0001<<7	128

表 32. 一位宽度的字段标识 (续)

字段名	十六进制值	十进制值
REDIST_PWEIGHT_START_NODE	0x0001<<8	256
REDIST_PWEIGHT	0x0001<<9	512

表 33. get_swrd_settings 输出参数

名称	数据类型	描述
redistMethod	SMALLINT	数字，指示要使用分发文件或目标分区映射来运行再分发。
pMapFile	VARCHAR(255)	目标分区映射的全路径文件名。
distFile	VARCHAR(255)	数据分发文件的全路径文件名。
stepSize	BIGINT	在必须调用落实来防止日志充满的情况之前可以移动的最大行数。在每个再分发步骤中可以移动的数目。
totalSteps	SMALLINT	用来完全再分发给定的数据库分区组的步骤数。
stageSize	SMALLINT	要连续运行的步骤数。
nextStep	SMALLINT	区分哪些步骤已完成、哪些仍需要运行的索引。
processState	SMALLINT	可由用户设置以在 nextStep 上停止再分发阶段的标志。
pNumber	VARCHAR(6000)	预先分配的字符串，将对应分区权重由数据库分区组中的所有分区号来填充。字符串中的分区号由 “,” 分隔。
pWeight	VARCHAR(6000)	预先分配的字符串，将使用已通过 SET_SWRD_SETTINGS 存储过程指定的每个分区上所有列的相对权重来填充。字符串中的分区权重由 “,” 分隔。

set_swrd_settings 存储过程

set_swrd_settings 函数创建或更改再分发注册表。如果该注册表不存在，则将创建它并向其中添加记录。如果注册表已存在，则使用 overwriteSpec 来标识要覆盖哪些字段值。overwriteSpec 字段使此函数能够将 NULL 输入用于不需要更新的字段。

值 “-2” 可用于此过程中的 stepSize 和 totalSteps，以指示该数字无限制。

表 34. set_swrd_settings 输入参数

名称	数据类型	描述
dbpgName	VARCHAR(128)	数据库分区组名，针对它来运行再分发进程。
overwriteSpec	SMALLINT	第 270 页的表 32 中的一位宽度的字段标识，指示要写入或覆盖至再分发设置注册表中的目标字段。
redistMethod	SMALLINT	数字，指示要使用分发文件或目标分区映射来运行再分发。
pMapFile	VARCHAR(255)	目标分区映射的全路径文件名。
distFile	VARCHAR(255)	数据分发文件的全路径文件名。
stepSize	BIGINT	在必须调用落实来防止日志充满的情况之前可以移动的最大行数。在每个再分发步骤中可以移动的数目。
totalSteps	SMALLINT	用来完全再分发给定的数据库分区组的步骤数。
stageSize	SMALLINT	要连续运行的步骤数。
nextStep	SMALLINT	区分哪些步骤已完成、哪些仍需要运行的索引。
processState	SMALLINT	可由用户设置以在 nextStep 上停止再分发阶段的标志。

表 34. set_swrd_settings 输入参数 (续)

名称	数据类型	描述
pNumber	VARCHAR(6000)	包含对应于分区权重的所有分区号的字符串。每个分区号都在 0 至 999 之间，数字由“,”分隔。该字符串中不允许空格。
pWeight	VARCHAR(6000)	字符串，包含用户根据 pNumber 字符串中的分区号指定的所有分区权重。每个分区权重都是 0 至 SQL_MAXSMALLVAL 之间的数字，数字由“,”分隔。该字符串中不允许空格。

analyze_log_space 存储过程

analyze_log_space 函数返回日志空间分析结果的结果集（开放式游标），对于给定的数据库分区组的每个数据库分区，包含以下字段。

表 35. analyze_log_space 字段

列名	列类型	描述
PARTITION_NUM	SMALLINT	日志空间分析的分区号
TOTAL_LOG_SIZE	BIGINT	已分配的总日志空间（以字节计），-1 指示大小无限制。
AVAIL_LOG_SPACE	BIGINT	空闲并可由再分发进程使用的日志空间量（以字节计）
DATA_SKEW	BIGINT	与目标级别不符的数据大小的绝对值（以字节计）
REQ_LOG_SPACE	BIGINT	达到期望的数据分发所需的空间量（以字节计）
NUM_OF_STEPS	SMALLINT	将数据变形减少到零所需的步骤数
MAX_STEP_SIZE	BIGINT	在不导致日志充满错误的情况下，一次可移动的最大数据量（以字节计）

表 36. analyze_log_space 输入参数

名称	数据类型	描述
inDBPGroup	VARCHAR(128)	数据库分区组名
inMainTbSchema	VARCHAR(128)	主表的模式
inMainTable	VARCHAR(128)	数据库分区组中的主表（通常，是数据库分区组中最大的表）
useTbType	SMALLINT	分析类型的指示符: SWRD_USE_STMG_TABLE 1: 指示使用存储管理表中的信息来查找每个分区的表的行计数 SWRD_USE_REALTIME_ANALYSIS 2: 指示使用 select 查询来查找每个分区的表的行计数
inStmgTime	VARCHAR(26)	存储管理记录的时间戳记。当 analysisType 设置为 SWRD_USE_REALTIME_ANALYSIS 时，忽略此参数。
addDropOption	CHAR(1)	添加或删除分区的指示符 “A”表示添加分区 “D”表示删除分区 “N”表示不添加或删除
addDropList	VARCHAR(6000)	要添加或删除的分区列示在一个字符串中，由逗号分隔（,）
pNumber	VARCHAR(6000)	包含对应于分区权重的所有分区号的字符串。每个分区号都在 0 至 999 之间，数字由“,”分隔。该字符串中不允许空格。

表 36. analyze_log_space 输入参数 (续)

名称	数据类型	描述
pWeight	VARCHAR(6000)	字符串，包含用户根据 pNumber 字符串中的分区号指定的所有分区权重。每个分区权重都是 0 至 SQL_MAXSMALLVAL 之间的数字，数字由 “,” 分隔。该字符串中不允许空格。

generate_Distfile 存储过程

generate_Distfile 函数针对给定的表生成数据分发文件，并将它以给定的 fileName 保存。

表 37. generate_Distfile 输入参数

名称	数据类型	描述
inTbSchema	VARCHAR(128)	表模式名
inTbName	VARCHAR(128)	表名

表 38. generate_Distfile 输入和输出参数

名称	数据类型	描述
fileName	VARCHAR(255)	数据分发文件名。如果给定的文件名只是文件名，则该文件被保存至 instance/tmp 目录，该参数返回完整文件路径名。

stepwise_redistribute_dbpg 存储过程

stepwise_redistribute_dbpg 函数根据输入和设置文件来再分发数据库分区组的部分。

步骤 1. 使用 inDbPGroup 名称来搜索设置注册表:

- 如果找不到注册表，则返回错误
- 如果找到了注册表，则读取以下值:
  - 当前步骤
  - 步骤数目
  - 分区映射名
  - 最大数据大小（在每个步骤中可移动的）
  - 进程状态
  - 分区权重

  您可以通过对 inNumSteps 输入 -1（SWRD_UNLIMITED_STEPS）来同时再分发所有数据库分区组

步骤 2. 如果注册表进程状态为 SWRD_STOP，则将停止该进程并返回警告消息。如果进程状态为 SWRD_CONTINUE，则进程继续。

步骤 3. 如果 inStartingPoint 不是 NULL 并且有效，则覆盖相应的注册表值，否则将读取当前步骤的注册表值并用作此步骤的开始点。

步骤 4. 将生成当前步骤的分区映射并从设置注册表中检索分区权重。如果已提供现有的完整分区映射或分发文件，则将相应地生成分区映射。

步骤 5. 使用分区映射选项调用再分发 API。当进程完成后，下一步的注册表值增加。

对注册表中指示的步骤数重复步骤 1 至 5。

值 “-2” 可用于此过程中以指示该数字无限制。

表 39. *stepwise_redistribute_dbpg* 输入参数

名称	数据类型	描述
inDBPGroup	VARCHAR(128)	目标数据库分区组的名称
inStartingPoint	SMALLINT	此参数可以为 NULL。如果它不为空并指向一个正数，则使用它来覆盖由 swrd 设置注册表提供的 “nextStep” 值。当您要从特定步骤重新运行 SWRD 时，这是一个很有用的选项。
inNumSteps	SMALLINT	要运行的步骤数。如果它不为空并指向一个正数，则使用它来覆盖由 swrd 设置注册表提供的 “numSteps” 值。当您要对不同于设置中指定数目的步骤重新运行 SWRD 时，这是一个很有用的选项。例如，如果在已调度的阶段有五个步骤，而 SWRD 进程在步骤 3 失败，则在更正错误条件后，可调用 SWRD 来运行余下的三个步骤。

db_partitions UDF

db_partitions 用户定义的函数对 `db2nodes.cfg` 文件进行语法分析，并为找到的每个分区返回一行。

输入参数: 无

表 40. *db_partitions* 输出参数

名称	数据类型	描述
PARTITION_NUMBER	SMALLINT	分区号
HOST_NAME	VARCHAR(128)	主机名（对于 Intel 平台，这是机器名）
PORT_NUMBER	SMALLINT	逻辑端口号
SWITCH_NAME	VARCHAR(128)	网络开关名称（可以为 NULL）

用法示例

以下是 AIX 上的 CLP 脚本的一个示例:

```
# -----------------------------------------------------------------------------------
# Set the database you wish to connect to
# -----------------------------------------------------------------------------------
dbName="SAMPLE"

# -----------------------------------------------------------------------------------
# Set the target database partition group name
# -----------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# -----------------------------------------------------------------------------------
# Specify the table name and schema
# -----------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# -----------------------------------------------------------------------------------
# Specify the name of the data distribution file
# -----------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# -----------------------------------------------------------------------------------
# Invoke call statements in clp
```

```
# ---------------------------------------------------------------------------------------
    db2startdb2 -v "connect to $dbName"

# ---------------------------------------------------------------------------------------
# Analysing the effect of adding a partition without applying the changes - a 'what if'
# hypothetical analysis
#
# - In the following case, the hypothesis is adding partition 40, 50 and 60 to the
#   database partition group, and for partitions 10,20,30,40,50,60, using a respective
#   target ratio of 1:2:1:2:1:2.
#
# NOTE: in this example only partitions 10, 20 and 30 actually exist in the database
#       partition group
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ---------------------------------------------------------------------------------------
# Analysing the effect of droping a partition without applying the changes
#
# - In the following case, the hypothesis is dropping partition 30 from the database
#   partition group, and redistributing the data in partitions 10 and 20 using a
#   respective target ratio of 1 : 1
#
# NOTE: In this example all partitions 10, 20 and 30 should exist in the database
#       partition group
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ---------------------------------------------------------------------------------------
# Generate a data distribution file to be used by the redistribute process
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ---------------------------------------------------------------------------------------
# Write a step wise redistribution plan into a registry
#
# Setting the 10th parameter to 1, may cause a currently running step wise redistribute
# stored procedure to complete the current step and stop, until this parameter is reset
# to 0, and the redistribute stored procedure is called again.
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ---------------------------------------------------------------------------------------
# Report the content of the step wise redistribution plan for the given database
# partition group.
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ---------------------------------------------------------------------------------------
# Redistribute the database partition group "dbpgName" according to the redistribution
# plan stored in the registry by set_swrd_settings. It starting with step 3 and
# redistributes the data until 2 steps in the redistribution plan are completed.
# ---------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

相关概念:

- 第 265 页的『数据再分发』

第 12 章 基准程序测试

本章说明基准程序进程以及如何使用 *db2batch* 实用程序来执行数据库工作负载的基准程序测试。

基准程序测试

基准程序测试是应用程序开发生命周期的一个常规部分。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，应对您的应用程序执行它，以确定当前性能并提高性能。如果已将应用程序代码编写得尽可能效率高，那么要想再改善性能，可调整数据库和数据库管理器配置参数。甚至可以调整应用程序参数来更好地满足应用程序的需求。

运行不同类型的基准程序测试来发现特定种类的信息:

- *每秒事务数*基准程序确定在某些有限的实验室条件下数据库管理器的吞吐量能力。
- *应用程序*基准程序测试与生产条件接近的条件下的相同吞吐量能力。

调整配置参数的基准程序测试基于这些“现实世界”条件，并需要使用各种参数值反复运行从您的应用程序中获取的 SQL，直到应用程序运行得尽可能有效率为止。

此处所描述的基准程序测试方法基于调整配置参数。但是，该基本技术同样可以用于调整影响性能的其它因素，如:

- SQL 语句
- 索引
- 表空间配置
- 应用程序代码
- 硬件配置

基准程序测试有助于了解数据库管理器在各种条件下是如何响应的。可以创建多个方案来测试死锁处理、实用程序性能、装入数据的不同方法以及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。

基准程序测试方法

基准程序测试基于可重复的环境，因此在相同条件下运行的相同测试将产生可以合理比较的结果。

可通过在一个正常的环境中运行测试应用程序来开始基准程序测试。随着您缩小性能问题的范围，可以开发专用的测试用例，以限制正测试的功能的作用域。这些专用测试用例不需要仿真整个应用程序来获取有价值的信息。从简单的评估开始，仅在必要时才增加复杂程度。

好的基准程序测试或评估应包括以下特征:

- 测试是可重复的。
- 测试的每次迭代在相同系统状态下开始。
- 除非方案包括系统中执行的一定量的其它活动，否则，系统中的其它功能或应用程序是不活动的。

注: 已启动的应用程序即使是在最小化或空闲时也会占用内存。这样增大了页面调度将使基准程序的运行结果产生偏差和违反可重复性规则的概率。

- 用于基准程序测试的硬件和软件与您的生产环境匹配。

对于基准程序测试，创建一个方案，然后在此方案中创建几次应用程序，以在每次运行期间捕获关键信息。在每次运行之后捕获关键信息在确定可以提高应用程序和数据库的性能的更改方面具有主要的重要性。

相关概念:

- 第 278 页的『基准程序准备』
- 第 279 页的『基准程序测试创建』
- 第 284 页的『基准程序测试执行』
- 第 285 页的『基准程序测试分析示例』

基准程序准备

在开始性能基准程序测试之前，完成应用程序对其运行的数据库的逻辑设计。设置并填充表、视图和索引。规范化表，绑定应用程序程序包并用实际数据填写表。

您还应该已确定了数据库的最终物理设计。将数据库管理器对象放在其最终磁盘位置，接大小排列日志文件，以确定工作文件和备份的位置，并测试备份过程。此外，检查程序包，以确保在可能时启用性能选项，如行分块。

您应该已在应用程序编程和测试阶段达到这样一个时候，即，将允许您创建基准程序。尽管在基准程序测试期间，可能会暴露应用程序的实际限制，但是，此处描述的基准程序的目的是测量性能，而不是检测故障或异常结束。

基准程序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。理论上，它应该在具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程序直接使用的任何通信或文件服务设施也应已调整好。

确保使用生产大小的数据库运行基准程序测试。单个 SQL 语句返回的数据量以及需要的排序量应与在生产环境中相同。此规则确保应用程序将测试典型的内存需求。

要进行基准程序测试的 SQL 语句应该是*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程序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。考虑选择平均值的语句的执行频率和处理的数据量。如果该数据量过大，则即使这些语句是典型的 SQL 语句，也将它们归为*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。
  - 处理大量数据的语句。
  - 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示客户地址和帐户；但是，还必须合并其它几个表，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，在这种情况下，可能节省的时间迫使提高性能要求。
- 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。

相关概念:

- 第 277 页的『基准程序测试』
- 第 279 页的『基准程序测试创建』

基准程序测试创建

当设计和实现基准程序时，考虑各种因素。由于该程序的主要目的是模拟用户应用程序，所以程序的总体结构是变化的。可以将整个应用程序用作基准程序，只需引入某种方法来对要分析的 SQL 语句进行计时。对于大的或复杂的应用程序，只包括包含重要语句的块可能更实用。

要测试特定 SQL 语句的性能，可以将这些语句以及必需的 CONNECT、PREPARE、OPEN 和其它语句以及计时机制一起单独包括在该基准程序中。

另一个要考虑的因素是要使用的基准程序的类型。一个选择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。另一个选择是只确定执行个别 SQL 语句所需的时间。

对于所有基准程序测试，需要一个高效率的计时系统来计算个别 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，重点是跟踪 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，而可能会优先测试个别语句的执行时间。

尽管每个查询的经过时间是性能分析中的一个重要因素，但可能不必暴露瓶颈。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 限制，而不是 CPU 的使用达到满负荷。基准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将从查询检索到的整组行发送至某个输出设备。例如，整个回答集可能是另一个程序的输入，因此，不发送第一个应用程序的任何行作为输出。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。要提供准确的模拟，基准程序应该反映特定应用程序的行处理。如果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 基准程序工具：在您的实例 `sqllib` 目录的 `bin` 子目录中提供了一个基准程序工具（`db2batch`）。此工具使用许多准则，以用于创建基准程序。此工具可以从平面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。它还允许控制回答集的大小以及从此回答集发送至输出设备的行数。

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、锁定和从数据库监视器收集的其它统计信息。如果正在对一组 SQL 语句进行计时，则 `db2batch` 也汇总性能结果，并提供算术和几何平均数。 对于语法和选项，在命令行上输入 `db2batch -h`。

此基准程序测试工具也有一个 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
```

可以在远程运行 db2batch。若使用基准程序工具的

`-f <filename>`

或

`-o <options>`

命令参数，则:

- 控制选项

  `perf_detail`

  和

  `-p <perf_detail>`

  （指定要返回的性能信息级别）当设置为大于 1 时不受支持（在远程运行时）。

除了这两项以外，

`perf_detail`

和

`-p <perf_detail>`

在所有“DB2® 通用数据库”平台上都受支持且有效。

db2batch 测试示例

以下示例说明可以如何对输入文件 `db2batch.sql` 使用 `db2batch`:

```
-- 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 26. 样本基准程序输入文件: db2batch.sql

使用对基准程序工具的以下调用:

```
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 27. 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 28. 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 29. 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视器返回的特定数据元素。

在下一个示例中（在 UNIX® 上），仅生成具体查询表。

```
db2batch -d sample -f db2batch.sql -r /dev/null,
```

仅生成具体查询表。如果使用 -r 选项，则 outfile1 被替换为 /dev/null，而 outfile2（它只包含具体查询表）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 30. db2batch 的样本输出 – 只包含具体查询表

相关概念:

- 第 279 页的『基准程序测试创建』
- 第 284 页的『基准程序测试执行』
- 第 285 页的『基准程序测试分析示例』

基准程序测试执行

对于一种类型的数据库基准程序，选择一个配置参数并使用该参数的不同值运行该测试，直至达到最佳效果。单个测试应该包括通过相同参数值的多次迭代（例如，20 或 30 次）来执行该应用程序以获取平均计时，这可以更清楚地显示参数更改所产生的影响。

当运行基准程序时，应该将第一次迭代（称为热身运行）视为不同于后续迭代（称为正常运行）的一种特殊情况。因为热身运行包括某些启动活动（例如，初始化缓冲池），因此，花费的时间比正常运行要稍微长一些。虽然来自热身运行的信息可能实际上是有效的，但从统计角度来说是无效的。当计算一组特定参数值的平均计时或 CPU 时，仅使用来自正常运行的结果。

可以考虑使用“配置顾问程序”来创建基准程序的热身运行。“配置顾问程序”问到的问题可以涉及到在进行基准程序活动期间为正常运行调整环境配置时要考虑的若干事宜。可以从“控制中心”或通过带适当选项执行 db2 自动配置命令来启动“配置顾问程序”。

如果基准程序测试使用单个查询，则通过刷新缓冲池来确保最小化先前查询的潜在影响。要刷新缓冲池，读取与查询无关的大量的页来填充该缓冲池。

在完成单组参数值的迭代之后，可以更改单个参数。但是，在每个迭代之间，执行下列任务，以便将基准程序的环境复原至它的初始状态:
- . 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。
- 如果测试更新了测试中使用的数据，则此数据必须保持一致。为此:
  - 使用 RESTORE 实用程序来复原整个数据库。数据库的备份副本包含它的先前状态，即已准备好进行下次测试。
  - 使用 IMPORT 或 LOAD 实用程序来复原该数据的导出副本。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。
- 要将应用程序返回至它的原始状态，将应用程序重新绑定至数据库。

概括起来，遵循以下步骤或迭代来对数据库应用程序执行基准测试:

第一步 除下列参数外，将数据库和数据库管理器的其它调整参数保持为它们的**缺省**值:
- 对于测试的工作负载和目标很重要的那些参数。（您很少有足够的时间执行基准程序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。
- 为了使应用程序能够运行而必须更改的参数（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，然后计算平均计时或 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，然后计算平均计时或 CPU。

第四步 根据基准程序测试的结果，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，以帮助您进行基准程序测试。可使用 REXX 之类的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程序测试，分析结果，并为给定测试提供一个最终和最优参数值报告。

相关概念:

- 第 277 页的『基准程序测试』
- 第 278 页的『基准程序准备』
- 第 280 页的『db2batch 测试示例』

基准程序测试分析示例

基准程序的输出应该包括每个测试的标识、程序执行的迭代、语句号和执行的计时。

在经过一系列测量之后，基准程序测试结果的总结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 31. 基准程序的样本结果

注： 上面报告中的数据仅供说明之用。它**不**表示测量的结果。

分析显示：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。

如果您的程序可以定界的 ASCII 格式输出数据，则可以在以后将该数据导入数据库表或电子表格进行进一步统计分析。

基准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
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 32. 基准程序的样本计时报告

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。

相关概念:

- 第 277 页的『基准程序测试』
- 第 279 页的『基准程序测试创建』
- 第 284 页的『基准程序测试执行』

第 13 章 配置 DB2

配置参数

当创建了 DB2 Universal Database™（DB2 通用数据库）实例或数据库时，就会使用缺省参数值创建相应的配置文件。可以修改这些参数值以提高性能以及实例或数据库的其它特征。

*配置文件*包含一些参数，这些参数定义诸如分配给 DB2 UDB 产品和各个数据库的资源以及诊断级别之类的值。有两种类型的配置文件:

- 每个 DB2 UDB 实例的数据库管理器配置文件
- 每个独立的数据库的数据库配置文件。

*数据库管理器配置文件*是在创建 DB2 UDB 实例时创建的。它包含的参数在实例级影响系统资源，不受任何一个作为该实例一部分的数据库的影响。根据系统的配置，可将这些参数中的许多参数的值更改为非系统缺省值，以提高性能或增大容量。

每个客户机安装也有一个数据库管理器配置文件。此文件包含关于特定工作站的客户机启用程序的信息。在可用于服务器的参数中，有一个子集可用于客户机。

数据库管理器配置参数存储在名为 db2systm 的文件中。此文件是在创建数据库管理器的实例时创建的。在基于 UNIX 的环境中，可以在数据库管理器的实例的 sqllib 子目录中找到此文件。在 Windows 中，此文件的缺省位置是 sqllib 目录的实例子目录。如果设置 DB2INSTPROF 变量，则文件在 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对同一文件都具有存取权。数据库管理器的配置在所有数据库分区服务器上都相同。

大多数参数会影响将分配给数据库管理器的单个实例的系统资源量，或者这些参数会配置数据库管理器和基于环境考虑的不同通信子系统的设置。另外，存在仅供参考的其它参数，不能更改它们。所有这些参数都具有全局适用性，独立于存储在数据库管理器的该实例下的任何单个数据库。

*数据库配置文件*是在创建数据库时创建的，它驻留在数据库所驻留的地方。每个数据库都有一个配置文件。其参数指定要分配给该数据库的资源量以及其它事项。可更改许多参数的值，以提高性能或增大容量。根据特定数据库中活动类型的不同，可能需要进行不同的更改。

个别数据库的参数存储在名为 SQLDBCON 的配置文件中。此文件与 SQLnnnnn 目录中的数据库的其它控制文件存储在一起，其中 nnnnn 是创建数据库时指定的数字。每个数据库都有它自己的配置文件，并且文件中的大多数参数指定分配给该数据库的资源量。该文件还包含描述信息以及指示数据库的状态的标志。

在分区数据库环境中，对于每个数据库分区都存在一个单独的 SQLDBCON 文件。在每个数据库分区中，SQLDBCON 文件中的值可能相同或不同，但是建议数据库配置参数值在所有分区上相同。

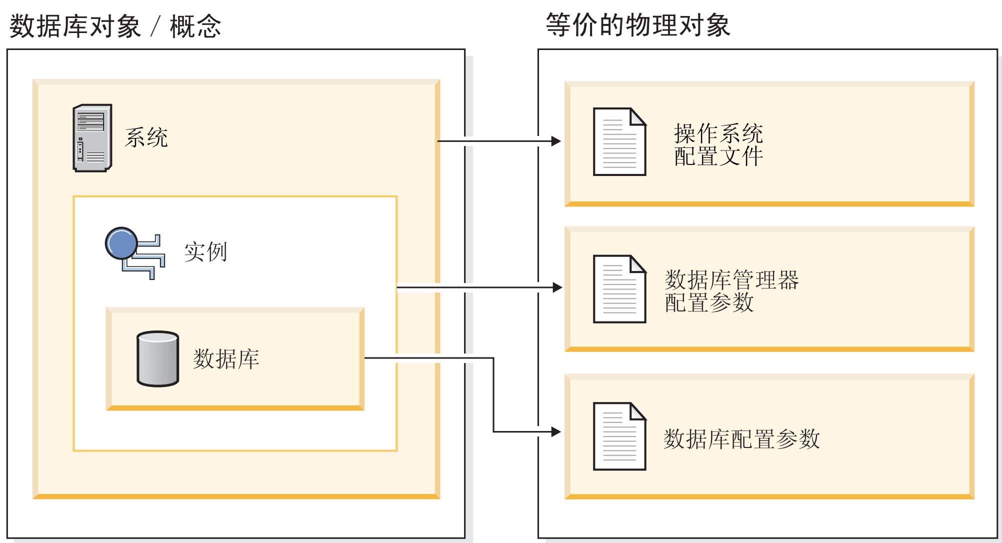


图 33. 数据库对象与配置文件之间的关系

相关概念:

- 第 288 页的『配置参数调整』

相关任务:

- 第 289 页的『使用配置参数配置 DB2』

配置参数调整

数据库管理器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载或类型
- 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器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。

调整配置的好的起始点是“性能配置”向导或 AUTOCONFIGURE 命令。

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。

可以将某些配置参数设置为*自动*。然后，DB2® 将自动调整这些参数以反映当前资源需求。

相关概念:

- 第 287 页的『配置参数』

相关任务:

- 第 289 页的『使用配置参数配置 DB2』

相关参考:

- 第 294 页的『配置参数总结』

使用配置参数配置 DB2

数据库管理器配置参数存储在名为 db2systm 的文件中。数据库配置参数存储在名为 SQLDBCON 的文件中。不能直接编辑这些文件，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 db2systm 或 SQLDBCON，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这些文件。

可以使用下列其中一种方法来复位、更新和查看配置参数:

- 使用控制中心。“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器配置参数。“配置数据库”笔记本可用来改变数据库配置参数的值。DB2 控制中心还提供了“配置顾问程序”，可用来改变配置参数的值。此顾问程序根据您对一组问题（例如，对数据库运行的事务的工作负载和类型）提供的回答来为参数生成值。

  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”的树形视图中的数据库对象启动“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。（但是，建议配置参数值应在所有分区上相同。）

  注: “配置顾问程序”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:

  对于数据库管理器配置参数:

  - GET DATABASE MANAGER CONFIGURATION（或 GET DBM CFG）
  - UPDATE DATABASE MANAGER CONFIGURATION（或 UPDATE DBM CFG）
  - RESET DATABASE MANAGER CONFIGURATION（或 RESET DBM CFG），来将*所有*数据库管理器参数复位到其缺省值
  - AUTOCONFIGURE。

  对于数据库配置参数:

  - GET DATABASE CONFIGURATION（或 GET DB CFG）
  - UPDATE DATABASE CONFIGURATION（或 UPDATE DB CFG）

- RESET DATABASE CONFIGURATION（或 RESET DB CFG），来将*所有*数据库参数复位到其缺省值
- AUTOCONFIGURE。

- 使用应用程序编程接口（API）。这些 API 可以很容易地从应用程序或主机语言程序中调用。
- 使用“配置助手”（对于数据库管理器配置参数）。只能使用“配置助手”在客户机上设置数据库管理器配置参数。

对于某些数据库管理器配置参数，必须停止数据库管理器（db2stop），然后重新启动它（db2start），以使新参数值生效。

对于某些数据库参数，仅当重新激活数据库时，更改才将生效。在这些情况下，所有应用程序必须先与该数据库断开连接。（如果已激活该数据库，则必须先取消激活它，然后再重新激活它。）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。

可以联机更改其它参数；这些参数称为*可配置联机配置参数*。

如果在连接至某个实例时更改可配置联机数据库管理器配置参数的设置，则 UPDATE DBM CFG 命令的缺省行为将是立即应用更改。如果您不想立即应用更改，在 UPDATE DBM CFG 命令上使用 DEFERRED 选项。

要联机更改数据库管理器配置参数:

```
db2 attach to <instance-name>
   db2 update dbm cfg using <parameter-name> <value>
   db2 detach
```

对于客户机，数据库管理器配置参数的更改在下次该客户机与服务器连接时生效。

如果在连接时更改可配置联机数据库配置参数，则缺省行为是联机应用更改（只要有可能）。您应该注意，某些参数更改由于与分配空间相关的开销而可能花相当长的时间才会生效。要从命令行处理器联机更改配置参数，需要与数据库的连接。要联机更改数据库配置参数:

```
db2 connect to <dbname>
   db2 update db cfg using <parameter-name> <parameter-value>
   db2 connect reset
```

每个可配置联机配置参数都有一个与之关联的*传播类*。传播类指示您可以期望配置参数的更改何时生效。有三种传播类:

- **立即**: 在命令或 API 调用中立即更改的参数。例如，*diaglevel* 具有立即传播类。
- **语句边界**: 在语句或类似语句的边界上更改的参数。例如，如果您更改 *sortheap* 的值，则所有新的 SQL 请求将使用该新值启动。
- **事务边界**: 在事务边界上更改的参数。例如，在 COMMIT 语句之后更新 *dl_expint* 的新值。

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。当更改其中任何一个参数后，您应该考虑重新绑定您的应用程序，以确保对您的 SQL 语句使用最佳的存取方案。联机修改（例如，通过使用 UPDATE DATABASE CONFIGURATION IMMEDIATE 命令）的任何参数将导致 SQL 优化器为新的 SQL 语句选择新的存取方案。但是，SQL 语句高速缓存将不清除现有条目。要清除 SQL 高速缓存的内容，使用 FLUSH PACKAGE CACHE 语句。

当新参数值可能未立即生效时，查看参数设置（使用 GET DATABASE MANAGER CONFIGURATION 或 GET DATABASE CONFIGURATION 命令），它们将始终显示最新的更新。使用这些命令上的 SHOW DETAIL 子句查看参数设置将显示内存中的最新更新和值。

注: 在帮助和其它 DB2 文档中，大量配置参数（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”。

相关概念:

- 第 287 页的『配置参数』
- 第 288 页的『配置参数调整』

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 第 294 页的『配置参数总结』
- 『FLUSH PACKAGE CACHE statement』（*SQL Reference, Volume 2*）

动态地配置参数

DB2 允许您利用动态配置。在 DB2 数据库正在运行、接受连接或处理事务时，可以更改该数据库或实例中的某些配置参数。下面是一个说明可以如何使用 DB2 中的动态配置功能的示例。

在此方案中，为数据库服务器配置了 4 GB 的内存，其中 3.5 GB 内存可用于数据库管理器。有 8 个处理器。数据库服务器专用于正在 DB2INST 实例上运行的名为 DB2DYN 的单个数据库。数据库工作负载在一天和一周中是有所变化的，如下所示:

- 白天的工作负载（05:00-20:00）包括许多连接和并发事务。
- 一天结束时的工作负载（20:00-24:00）包括总结报告和决策支持查询，具有很少的连接和事务。
- 日常维护工作负载（24:00-05:00）包括联机装入操作、增量备份和索引创建等等。
- 每周维护工作负载包括大型表重组操作、运行统计信息操作和创建较大索引。

如果具有这些工作负载特征，则可如下表所示来配置系统。

	白天（05:00 - 20:00）	一天的结束（20:00 - 24:00）	日常维护（24:00 - 05:00）	每周维护（星期天）
数据库活动	较重的事务工作负载	决策支持查询	装入、备份和索引创建	在缓冲池 1 中执行重组和运行统计信息

	白天（05:00 - 20:00）	一天的结束（20:00 - 24:00）	日常维护（24:00 - 05:00）	每周维护（星期天）
缓冲池 1（MB）	1000	500	500	2000
缓冲池 2（MB）	1000	500	500	200
排序堆（MB）	0.1	20	200	200
目录高速缓存（MB）	200	200	50	50
程序包高速缓存（MB）	800	200	200	200
实用程序堆（MB）	0	0	1000	0
对话级别	1	3	4	4

可以从将数据库配置参数 *database_memory* 设置为 3.5GB 开始，从而为数据库保留此指定数量的可用内存。这是一次性操作，一旦完成此操作，它就将为数据库提供 3.5 GB（或 917 504 个 4 KB 页）的保留内存，用于创建缓冲池或调整配置。

```
db2start
db2 update db cfg for db2dyn using database_memory 917504
 db2stop
```

然后，可以使用下列脚本来将数据库从一种配置转换为另一种配置。（可以安排在适当的时间运行这些脚本。）

```
MorningConfiguration.sh

# This script is used to prepare
# the database server for the
# morning configuration,
# for a workload consisting of
# a large number of OLTP connections and
# concurent transactions.

db2 connect to db2dyn

db2 update db cfg using sortheap 25
db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool1 size 262144
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 204800
db2 alter bufferpool bufferpool2 size 262144
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
   db2 connect reset
db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 1
db2 get dbm cfg show detail
   db2 detach


EveningConfiguration.sh

# This script is used to prepare
# the database server for the
# evening configuration,
# for a workload consisting of
# decision-support queries.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
```

```
db2 update db cfg using pkcachesz 51200
db2 update db cfg using catcachesz 51200
db2 update db cfg using util_heap_sz 32
db2 update db cfg using sortheap 5120
db2 flush package cache dynamic
db2 get db cfg show detail
   db2 connect reset
db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 3
db2 get dbm cfg show detail
   db2 detach

NightTimeConfiguration.sh

# This script is used to prepare
# the database server for the
# daily maintenance configuration,
# for a workload consisting of
# index creation and load and backup
# operations.

db2 connect to db2dyn

db2 alter bufferpool bufferpool1 size 131072
db2 commit
db2 alter bufferpool bufferpool2 size 131072
db2 commit
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 update db cfg using sortheap 51200
db2 update db cfg using util_heap_sz 262144
db2 flush package cache dynamic
db2 get db cfg show detail
   db2 connect reset
db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
   db2 detach

MaintenanceConfiguration.sh

# This script is used to prepare
# the database server for the
# weekly maintenance configuration,
# for a workload consisting of
# reorg and runstats operations executed
# in buffer pool 1.

db2 connect to db2dyn

db2 update db cfg using util_heap_sz 32
db2 alter bufferpool bufferpool2 size 51200
db2 commit
db2 update db cfg using sortheap 5120
db2 update db cfg using catcachesz 51200
db2 update db cfg using pkcachesz 51200
db2 alter bufferpool bufferpool1 size 524288
db2 commit
db2 flush package cache dynamic
db2 get db cfg show detail
   db2 connect reset
db2 attach to instance db2inst
db2 update dbm cfg using diaglevel 4
db2 get dbm cfg show detail
   db2 detach
```

注意，一个脚本与下一个脚本的操作顺序是不同的。应该在执行会增大内存需求的操作之前完成将减少内存需求的操作；否则，增大内存需求可能会失败。

所有缓冲池大小调整都是在落实操作之前进行的，原因是缓冲池更改是 SQL 操作，并且是事务的一部分。

每次重新配置之后都会刷新程序包高速缓存，以发出信号通知优化器已经对配置进行了重大更改，现有的所有动态 SQL 存取方案现在都变得无效。

动态数据库管理器重新配置操作是在应用程序连接至实例时执行的，而动态数据库操作则是在应用程序连接至数据库时执行的。

GET DATABASE MANAGER CONFIGURATION 命令或 GET DATABASE CONFIGURATION 命令的 SHOW DETAIL 子句可用来验证动态重新配置操作是否已生效。

此方案中可能已动态更改的其它数据库配置参数包括:

- *locklist* 参数，如果数据库经常进行锁定升级，则可动态增大该参数。但是，不能动态减小此参数的值。
- *dft_queryopt* 参数，它只适用于新的连接，在启动决策支持工作负载之前可以用来提高优化级别。
- *dft_degree* 参数，它也是只适用于新的连接，可用来为决策支持查询提供增强的查询内并行性。

配置参数总结

数据库管理器配置参数总结

下表列示了数据库服务器的数据库管理器配置文件中的参数。当更改数据库管理器配置参数时，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描述的一部分。

对于某些数据库管理器配置参数，必须停止数据库管理器（db2stop），然后重新启动它（db2start）才能使新的参数值生效。可以联机更改其它参数；这些参数称为*可联机配置的配置参数*。如果在连接至某个实例时更改可联机配置的数据库管理器配置参数的设置，则 UPDATE DBM CFG 命令的缺省行为是立即应用更改。如果您不想立即应用更改，在 UPDATE DBM CFG 命令上使用 DEFERRED 选项。

下表中的“自动”列指示参数是否支持 UPDATE DATABASE MANAGER CONFIGURATION 命令中的 AUTOMATIC 关键字。如果将某个参数设置为自动，则 DB2 将自动调整该参数以反映当前的资源需求。

“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，这意味着将接受所提供的缺省值。
- **中** – 指示该参数对性能有某些影响。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影响。尽管不必调整这些参数来增强性能，但是它们对于系统配置的其它方面（例如，通信支持）可能很重要。

“标记”、“标记值”和“数据类型”列提供调用 **db2CfgGet** 或 **db2CfgSet** API 时将需要的信息。此信息包括配置参数标识、*db2CfgParam* 数据结构中的 *token* 元素的条目和传递至该结构的值的数据类型。

表 41. 可配置的数据库管理器配置参数

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
agent_stack_sz	否	否	低	SQLF_KTN_AGENT_STACK_SZ	61	Uint16	第 318 页的『agent_stack_sz － 代理进程堆栈大小』
agentpri	否	否	高	SQLF_KTN_AGENTPRI	26	Sint16	第 344 页的『agentpri － 代理进程的优先级』
aslheapsz	否	否	高	SQLF_KTN_ASLHEAPSZ	15	Uint32	第 326 页的『aslheapsz － 应用程序支持层堆大小』
audit_buf_sz	否	否	高	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32	第 330 页的『audit_buf_sz － 审计缓冲区大小』
authentication[1]	否	否	低	SQLF_KTN_AUTHENTICATION	78	Uint16	第 427 页的『authentication － 认证类型』
catalog_noauth	是	否	无	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	第 428 页的『catalog_noauth － 在没有权限的情况下允许编目』
clnt_krb_plugin	否	否	无	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)	第 429 页的『clnt_krb_plugin － 客户机 Kerberos 插件』
clnt_pw_plugin	否	否	无	SQLF_KTN_CLNT_PW_PLUGIN	811	char(33)	第 429 页的『clnt_pw_plugin － 客户机用户标识密码插件』
comm_bandwidth	是	否	中	SQLF_KTN_COMM_BANDWIDTH	307	float	第 419 页的『comm_bandwidth － 通信带宽』
conn_elapse	是	否	中	SQLF_KTN_CONN_ELAPSE	508	Uint16	第 407 页的『conn_elapse － 连接经过时间』
cpuspeed	是	否	Low[2]	SQLF_KTN_CPUSPEED	42	float	第 420 页的『cpuspeed － CPU 速度』
datalinks	否	否	低	SQLF_KTN_DATALINKS	603	Sint16	第 391 页的『datalinks － 启用 Data Links 支持』
dft_account_str	是	否	无	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)	第 421 页的『dft_account_str － 缺省对方付费帐户』
dft_monswitches • *dft_mon_bufpool* • *dft_mon_lock* • *dft_mon_sort* • *dft_mon_stmt* • *dft_mon_table* • *dft_mon_timestamp* • *dft_mon_uow*	是	否	中	SQLF_KTN_DFT_MONSWITCHES[3] • SQLF_KTN_DFT_MON_BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_TABLE • SQLF_KTN_DFT_MON_TIMESTAMP • SQLF_KTN_DFT_MON_UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	第 418 页的『dft_monswitches － 缺省数据库系统监视器开关』
dftdbpath	是	否	无	SQLF_KTN_DFTDBPATH	27	char(215)	第 430 页的『dftdbpath － 缺省值数据库路径』
diaglevel	是	否	低	SQLF_KTN_DIAGLEVEL	64	Uint16	第 415 页的『diaglevel － 诊断错误捕获级别』
diagpath	是	否	无	SQLF_KTN_DIAGPATH	65	char(215)	第 416 页的『diagpath － 诊断数据目录路径』
dir_cache	否	否	中	SQLF_KTN_DIR_CACHE	40	Uint16	第 331 页的『dir_cache － 目录高速缓存支持』
discover[4]	否	否	中	SQLF_KTN_DISCOVER	304	Uint16	第 405 页的『discover － 发现方式』
discover_inst	是	否	低	SQLF_KTN_DISCOVER_INST	308	Uint16	第 406 页的『discover_inst － 发现服务器实例』

表 41. 可配置的数据库管理器配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
fcm_num_anchors	是	是	中	SQLF_KTN_FCM_NUM_ANCHORS	506	Sint32	第 408 页的『fcm_num_anchors － FCM 消息锚点数』
fcm_num_buffers	是	是	中	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32	第 408 页的『fcm_num_buffers － FCM 缓冲区数目』
fcm_num_connect	是	是	中	SQLF_KTN_FCM_NUM_CONNECT	505	Sint32	第 409 页的『fcm_num_connect － FCM 连接条目数』
fcm_num_rqb	是	是	中	SQLF_KTN_FCM_NUM_RQB	504	Uint32	第 410 页的『fcm_num_rqb － FCM 请求块数』
fed_noauth	是	否	无	SQLF_KTN_FED_NOAUTH	806	Uint16	第 431 页的『fed_noauth － 绕过联合认证』
federated	否	否	中	SQLF_KTN_FEDERATED	604	Sint16	第 422 页的『federated － 联合数据库系统支持』
fenced_pool	否	否	中	SQLF_KTN_FENCED_POOL	80	Sint32	第 353 页的『fenced_pool － 受防护的进程的最大数目』
group_plugin	否	否	无	SQLF_KTN_GROUP_PLUGIN	810	char(33)	第 431 页的『group_plugin － 组插件』
health_mon	是	否	低	SQLF_KTN_HEALTH_MON	804	Uint16	第 417 页的『health_mon － 运行状况监视』
indexrec[5]	是	否	中	SQLF_KTN_INDEXREC	20	Uint16	第 378 页的『indexrec － 索引重新创建时间』
instance_memory	否	是	中	SQLF_KTN_INSTANCE_MEMORY	803	Uint64	第 332 页的『instance_memory － 实例内存』
intra_parallel	否	否	高	SQLF_KTN_INTRA_PARALLEL	306	Sint16	第 413 页的『intra_parallel － 启用分区内并行性』
java_heap_sz	否	否	高	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32	第 333 页的『java_heap_sz － 最大 Java 解释器堆大小』
jdk_path	否	否	无	SQLF_KTN_JDK_PATH	311	char(255)	第 422 页的『jdk_path － Software Developer's Kit for Java 安装路径』
keepfenced	否	否	中	SQLF_KTN_KEEPFENCED	81	Uint16	第 354 页的『keepfenced － 保持受防护的进程』
local_gssplugin	否	否	无	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)	第 432 页的『local_gssplugin － 用于本地实例级别授权的 GSS API 插件』
max_connections	否	否	中	SQLF_DBTN_MAX_CONNECTIONS	802	Sint32	第 346 页的『max_connections － 最大客户机连接数』
max_connretries	是	否	中	SQLF_KTN_MAX_CONNRETRIES	509	Uint16	第 411 页的『max_connretries － 节点连接重试次数』
max_coordagents	否	否	中	SQLF_KTN_MAX_COORDAGENTS	501	Sint32	第 346 页的『max_coordagents － 最大协调代理进程数』
max_querydegree	是	否	高	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32	第 414 页的『max_querydegree － 最大查询并行度』
max_time_diff	否	否	中	SQLF_KTN_MAX_TIME_DIFF	510	Uint16	第 412 页的『max_time_diff － 节点之间的最大时间差』
maxagents	否	否	中	SQLF_KTN_MAXAGENTS	12	Uint32	第 347 页的『maxagents － 最大代理进程数』
maxcagents	否	否	中	SQLF_KTN_MAXCAGENTS	13	Sint32	第 349 页的『maxcagents － 最大并发代理进程数』
maxtotfilop	否	否	中	SQLF_KTN_MAXTOTFILOP	45	Uint16	第 351 页的『maxtotfilop － 打开的最大总文件数』
min_priv_mem	否	否	中	SQLF_KTN_MIN_PRIV_MEM	43	Uint32	第 320 页的『min_priv_mem － 最小已落实专用内存』
mon_heap_sz	否	否	低	SQLF_KTN_MON_HEAP_SZ	79	Uint16	第 334 页的『mon_heap_sz － 数据库系统监视器堆大小』
nname	否	否	无	SQLF_KTN_NNAME	7	char(8)	第 403 页的『nname － NetBIOS 工作站名称』

表 41. 可配置的数据库管理器配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
notifylevel	是	否	低	SQLF_KTN_NOTIFYLEVEL	605	Sint16	第 417 页的『notifylevel － 通知级别』
num_initagents	否	否	中	SQLF_KTN_NUM_INITAGENTS	500	Uint32	第 352 页的『num_initagents － 池中的代理进程的初始数目』
num_initfenced	否	否	中	SQLF_KTN_NUM_INITFENCED	601	Sint32	第 355 页的『num_initfenced － 受防护的进程的初始数目』
num_poolagents	否	否	高	SQLF_KTN_NUM_POOLAGENTS	502	Sint32	第 352 页的『num_poolagents － 代理进程池大小』
numdb	否	否	低	SQLF_KTN_NUMDB	6	Uint16	第 423 页的『numdb － 同时活动数据库的最大数目（包括主机数据库和 iSeries 数据库管理器）』
priv_mem_thresh	否	否	中	SQLF_KTN_PRIV_MEM_THRESH	44	Sint32	第 321 页的『priv_mem_thresh － 专用内存阈值』
query_heap_sz	否	否	中	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32	第 322 页的『query_heap_sz － 查询堆大小』
resync_interval	否	否	无	SQLF_KTN_RESYNC_INTERVAL	68	Uint16	第 383 页的『resync_interval － 事务再同步时间间隔』
rqrioblk	否	否	高	SQLF_KTN_RQRIOBLK	1	Uint16	第 329 页的『rqrioblk － 客户机 I/O 块大小』
sheapthres	否	否	高	SQLF_KTN_SHEAPTHRES	21	Uint32	第 323 页的『sheapthres － 排序堆阈值』
spm_log_file_sz	否	否	低	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32	第 384 页的『spm_log_file_sz － 同步点管理器日志文件大小』
spm_log_path	否	否	中	SQLF_KTN_SPM_LOG_PATH	313	char(226)	第 385 页的『spm_log_path － 同步点管理器日志文件路径』
spm_max_resync	否	否	低	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32	第 385 页的『spm_max_resync － 同步点管理器再同步代理进程限制』
spm_name	否	否	无	SQLF_KTN_SPM_NAME	92	char(8)	第 386 页的『spm_name － 同步点管理器名称』
srvcon_auth	否	否	无	SQLF_KTN_SRVCON_AUTH	815	Uint16	第 432 页的『srvcon_auth － 服务器上的入局连接的认证类型』
srvcon_gssplugin_list	否	否	无	SQLF_KTN_SRVCON_GSSPLUGIN_LIST	814	char(256)	第 433 页的『srvcon_gssplugin_list － 用于服务器上的入局连接的 GSS API 插件的列表』
srv_plugin_mode	否	否	无	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16	第 434 页的『srv_plugin_mode － 服务器插件方式』
srvcon_pw_plugin	否	否	无	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)	第 434 页的『srvcon_pw_plugin － 服务器上的入局连接的用户标识密码插件』
start_stop_time	是	否	低	SQLF_KTN_START_STOP_TIME	511	Uint16	第 412 页的『start_stop_time － 启动和停止超时』
svcename	否	否	无	SQLF_KTN_SVCENAME	24	char(14)	第 404 页的『svcename － TCP/IP 服务名称』
sysadm_group	否	否	无	SQLF_KTN_SYSADM_GROUP	39	char(16)	第 435 页的『sysadm_group － 系统管理权限组名』
sysctrl_group	否	否	无	SQLF_KTN_SYSCTRL_GROUP	63	char(16)	第 436 页的『sysctrl_group － 系统控制权限组名』
sysmaint_group	否	否	无	SQLF_KTN_SYSMAINT_GROUP	62	char(16)	第 436 页的『sysmaint_group － 系统维护权限组名』
sysmon_group	否	否	无	SQLF_KTN_SYSMON	808	char(9)	第 437 页的『sysmon_group － 系统监视器权限组名』
tm_database	否	否	无	SQLF_KTN_TM_DATABASE	67	char(8)	第 386 页的『tm_database － 事务管理器数据库名称』
tp_mon_name	否	否	无	SQLF_KTN_TP_MON_NAME	66	char(19)	第 424 页的『tp_mon_name － 事务处理器监视器名称』

表 41. 可配置的数据库管理器配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
tpname	否	否	无	SQLF_KTN_TPNAME	25	char(64)	第 404 页的『tpname － APPC 事务程序名』
trust_allclnts[6]	否	否	无	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16	第 438 页的『trust_allclnts － 信赖所有客户机』
trust_clntauth	否	否	无	SQLF_KTN_TRUST_CLNTAUTH	302	Uint16	第 439 页的『trust_clntauth － 可信的客户机认证』
use_sna_auth	是	否	无	SQLF_KTN_USE_SNA_AUTH	805	Uint16	第 439 页的『use_sna_auth － 使用 SNA 认证』
util_impact_lim	是	否	高	SQLF_KTN_UTIL_IMPACT_LIM	807	Uint32	第 426 页的『util_impact_lim － 实例影响策略』

注:

1. 有效值（在 sqlenv.h 中定义）为:

```
SQL_AUTHENTICATION_SERVER (0)
SQL_AUTHENTICATION_CLIENT (1)
SQL_AUTHENTICATION_DCS (2)
SQL_AUTHENTICATION_DCE (3)
SQL_AUTHENTICATION_SVR_ENCRYPT (4)
SQL_AUTHENTICATION_DCS_ENCRYPT (5)
SQL_AUTHENTICATION_DCE_SVR_ENC (6)
SQL_AUTHENTICATION_KERBEROS (7)
SQL_AUTHENTICATION_KRB_SVR_ENC (8)
SQL_AUTHENTICATION_GSSPLUGIN (9)
SQL_AUTHENTICATION_GSS_SVR_ENC (10)
SQL_AUTHENTICATION_DATAENC (11)
SQL_AUTHENTICATION_DATAENC_CMP (12)
SQL_AUTHENTICATION_NOT_SPEC (255)
```

2. *cpuspeed* 参数可以对性能产生显著影响，但是除非在非常特定的情况下，否则应使用缺省值，如参数描述中所述。
3.
```
Bit 1 (xxxx xxx1): dft_mon_uow
Bit 2 (xxxx xx1x): dft_mon_stmt
Bit 3 (xxxx x1xx): dft_mon_table
Bit 4 (xxxx 1xxx): dft_mon_buffpool
Bit 5 (xxx1 xxxx): dft_mon_lock
Bit 6 (xx1x xxxx): dft_mon_sort
Bit 7 (x1xx xxxx): dft_mon_timestamp
```

4. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

5. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
```

6. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

表 42. 参考数据库管理器配置参数

参数	标记	标记值	数据类型	其它信息
nodetype[1]	SQLF_KTN_NODETYPE	100	Uint16	第 423 页的『nodetype － 机器节点类型』
release	SQLF_KTN_RELEASE	101	Uint16	第 390 页的『release － 配置文件发行版级别』

表 42. 参考数据库管理器配置参数 (续)

参数	标记	标记值	数据类型	其它信息

注:

1. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_NT_STANDALONE (0)
SQLF_NT_SERVER (1)
SQLF_NT_REQUESTOR (2)
SQLF_NT_STAND_REQ (3)
SQLF_NT_MPP (4)
SQLF_NT_SATELLITE (5)
```

数据库配置参数总结

下表列示了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。

对于某些数据库配置参数，仅当重新激活数据库时，更改才将生效。在这些情况下，所有应用程序必须首先与该数据库断开连接。（如果已激活该数据库，则必须先取消激活它，然后再重新激活它。）这些更改将在下次连接数据库时生效。可以联机更改其它参数；这些参数称为*可联机配置的配置参数*。

下表中的“自动”列指示参数是否支持 UPDATE DATABASE MANAGER CONFIGURATION 命令中的 AUTOMATIC 关键字。如果将某个参数设置为自动，则 DB2 将自动调整该参数以反映当前的资源需求。

“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，这意味着将接受所提供的缺省值。
- **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。尽管不必调整这些参数来增强性能，但是它们对于系统配置的其它方面（例如，通信支持）可能很重要。

“标记”、“标记值”和“数据类型”列提供调用 **db2CfgGet** 或 **db2CfgSet** API 时将需要的信息。此信息包括配置参数标识、*db2CfgParam* 数据结构中的 *token* 元素的条目和传递至该结构的值的数据类型。

表 43. 可配置的数据库配置参数

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
alt_collate	否	否	无	SQLF_DBTN_ALT_COLLATE	809	Uint32	第 388 页的『alt_collate – 备用整理顺序』
app_ctl_heap_sz	否	否	中	SQLF_DBTN_APP_CTL_HEAP_SZ	500	Uint16	第 315 页的『app_ctl_heap_sz – 应用程序控制堆大小』

表 43. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
appgroup_mem_sz	否	否	中	SQLF_DBTN_APPGROUP_MEM_SZ	800	Uint32	第 316 页的『appgroup_mem_sz － 应用程序组内存集的最大大小』
applheapsz	否	否	中	SQLF_DBTN_APPLHEAPSZ	51	Uint16	第 319 页的『applheapsz － 应用程序堆大小』
archretrydelay	是	否	无	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16	第 365 页的『archretrydelay － 出错时的归档重试延迟』
autonomic_switches • *auto_maint* • *auto_db_backup* • *auto_tbl_maint* • *auto_runstats* • *auto_stats_prof* • *auto_prof_upd* • *auto_reorg*	是	否	中	SQLF_DBTN_AUTONOMIC_SWITCHES[1] • SQLF_ENABLE_AUTO_MAINT • SQLF_ENABLE_AUTO_DB_BACKUP • SQLF_ENABLE_AUTO_TBL_MAINT • SQLF_ENABLE_AUTO_RUNSTATS • SQLF_ENABLE_AUTO_STATS_PROF • SQLF_ENABLE_AUTO_PROF_UPD • SQLF_ENABLE_AUTO_REORG	830 • 831 • 833 • 835 • 837 • 839 • 844 • 841	Uint32	第 401 页的『autonomic_switches － 自动维护开关』
autorestart	是	否	低	SQLF_DBTN_AUTO_RESTART	25	Uint16	第 374 页的『autorestart － 启用自动重新启动』
avg_appls	是	否	高	SQLF_DBTN_AVG_APPLS	47	Uint16	第 345 页的『avg_appls － 活动应用程序的平均数目』
blk_log_dsk_ful	是	否	无	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16	第 365 页的『blk_log_dsk_ful － 磁盘已满时阻止进行日志记录』
catalogcache_sz	是	否	高	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32	第 307 页的『catalogcache_sz － 目录高速缓存大小』
chngpgs_thresh	否	否	高	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16	第 338 页的『chngpgs_thresh － 已更改页数的阈值』
database_memory	否	是	中	SQLF_DBTN_DATABASE_MEMORY	803	Uint64	第 308 页的『database_memory － 数据库共享内存大小』
dbheap	是	否	中	SQLF_DBTN_DB_HEAP	58	Uint64	第 309 页的『dbheap － 数据库堆』
dft_degree	是	否	高	SQLF_DBTN_DFT_DEGREE	301	Sint32	第 396 页的『dft_degree － 缺省并行度』
dft_extent_sz	是	否	中	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	第 339 页的『dft_extent_sz － 表空间的缺省扩展数据块大小』
dft_loadrec_ses	是	否	中	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16	第 374 页的『dft_loadrec_ses － 装入恢复会话的缺省数目』
dft_mttb_types	否	否	无	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32	第 396 页的『dft_mttb_types － 缺省维护的用于优化的表类型』
dft_prefetch_sz	是	是	中	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16	第 339 页的『dft_prefetch_sz － 缺省预取大小』
dft_queryopt	是	否	中	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	第 397 页的『dft_queryopt － 缺省查询优化类』
dft_refresh_age	否	否	中	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)	第 397 页的『dft_refresh_age － 缺省刷新寿命』
dft_sqlmathwarn	否	否	无	SQLF_DBTN_DFT_SQLMATHWARN	309	Sint16	第 398 页的『dft_sqlmathwarn － 算术异常时继续进行』
discover_db	是	否	中	SQLF_DBTN_DISCOVER	308	Uint16	第 406 页的『discover_db － 发现数据库』
dl_expint	是	否	无	SQLF_DBTN_DL_EXPINT	350	Sint32	第 391 页的『dl_expint － Data Links 访问标记到期时间间隔』

表 43. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
dl_num_copies	是	否	无	SQLF_DBTN_DL_NUM_COPIES	351	Uint16	第 391 页的『dl_num_copies － Data Links 副本数』
dl_time_drop	是	否	无	SQLF_DBTN_DL_TIME_DROP	353	Uint16	第 392 页的『dl_time_drop － 删除后的 Data Links 时间』
dl_token	是	否	低	SQLF_DBTN_DL_TOKEN	602	char(10)	第 392 页的『dl_token － Data Links 标记算法』
dl_upper	是	否	无	SQLF_DBTN_DL_UPPER	603	Sint16	第 393 页的『dl_upper － 采用大写的 Data Links 标记』
dl_wt_iexpint	是	否	无	SQLF_DBTN_DL_WT_IEXPINT	354	Sint32	第 393 页的『dl_wt_iexpint － Data Links 写令牌初始到期时间间隔』
dlchktime	是	否	中	SQLF_DBTN_DLCHKTIME	9	Uint32	第 335 页的『dlchktime － 检查死锁的时间间隔』
dyn_query_mgmt	否	否	低	SQLF_DBTN_DYN_QUERY_MGMT	604	Uint16	第 387 页的『dyn_query_mgmt － 动态 SQL 查询管理』
estore_seg_sz	否	否	中	SQLF_DBTN_ESTORE_SEG_SZ	303	Sint32	第 340 页的『estore_seg_sz － 扩充存储器内存段大小』
failarchpath	是	否	无	SQLF_DBTN_FAILARCHPATH	826	char(243)	第 366 页的『failarchpath － 故障转移日志归档路径』
groupheap_ratio	否	否	中	SQLF_DBTN_GROUPHEAP_RATIO	801	Uint16	第 317 页的『groupheap_ratio － 应用程序组堆的内存百分比』
hadr_local_host	否	否	无	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)	第 375 页的『hadr_local_host － HADR 本地主机名』
hadr_local_svc	否	否	无	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)	第 376 页的『hadr_local_svc － HADR 本地服务名称』
hadr_remote_host	否	否	无	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	第 376 页的『hadr_remote_host － HADR 远程主机名』
hadr_remote_inst	否	否	无	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	第 376 页的『hadr_remote_inst － 远程服务器的 HADR 实例名』
hadr_remote_svc	否	否	无	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	第 377 页的『hadr_remote_svc - HADR 远程服务名称』
hadr_syncmode	否	否	无	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	第 377 页的『hadr_syncmode － 处于对等状态时日志写操作的 HADR 同步方式』
hadr_timeout	否	否	无	SQLF_DBTN_HADR_TIMEOUT	816	Sint32	第 378 页的『hadr_timeout － HADR 超时值』
indexrec[2]	是	否	中	SQLF_DBTN_INDEXREC	30	Uint16	第 378 页的『indexrec － 索引重新创建时间』
locklist	是	否	高（当它影响升级时）	SQLF_DBTN_LOCK_LIST	704	Uint64	第 310 页的『locklist － 锁定列表的最大存储器』
locktimeout	否	否	中	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	第 335 页的『locktimeout － 锁定超时』
logarchmeth1	是	否	无	SQLF_DBTN_LOGARCHMETH1	822	Uint16	第 366 页的『logarchmeth1 － 主日志归档方法』
logarchmeth2	是	否	无	SQLF_DBTN_LOGARCHMETH2	823	Uint16	第 367 页的『logarchmeth2 － 辅助日志归档方法』
logarchopt1	是	否	无	SQLF_DBTN_LOGARCHOPT1	824	char(243)	第 367 页的『logarchopt1 － 主日志归档选项』
logarchopt2	是	否	无	SQLF_DBTN_LOGARCHOPT2	825	char(243)	第 368 页的『logarchopt2 － 辅助日志归档选项』

表 43. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
logbufsz	否	否	高	SQLF_DBTN_LOGBUFSZ	33	Uint16	第 312 页的『logbufsz － 日志缓冲区大小』
logfilsiz	否	否	中	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	第 356 页的『logfilsiz － 日志文件大小』
logindexbuild	是	是	无	SQLF_DBTN_LOGINDEXBUILD	818	Uint32	第 368 页的『logindexbuild － 创建的日志索引页』
logprimary	否	否	中	SQLF_DBTN_LOGPRIMARY	16	Uint16	第 358 页的『logprimary － 主日志文件数』
logretain[3]	否	否	低	SQLF_DBTN_LOG_RETAIN	23	Uint16	第 368 页的『logretain － 启用日志保留』
logsecond	是	否	中	SQLF_DBTN_LOGSECOND	17	Uint16	第 359 页的『logsecond － 辅助日志文件数』
max_log	是	是		SQLF_DBTN_MAX_LOG	807	Uint16	第 360 页的『max_log － 每个事务的最大日志』
maxappls	是	是	中	SQLF_DBTN_MAXAPPLS	6	Uint16	第 348 页的『maxappls － 最大活动应用程序数』
maxfilop	是	否	中	SQLF_DBTN_MAXFILOP	3	Uint16	第 350 页的『maxfilop － 每个应用程序打开的最大数据库文件数』
maxlocks	是	否	高（当它影响升级时）	SQLF_DBTN_MAXLOCKS	15	Uint16	第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
min_dec_div_3	否	否	高	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	第 328 页的『min_dec_div_3 － 十进制分隔小数位为 3』
mincommit	是	否	高	SQLF_DBTN_MINCOMMIT	32	Uint16	第 369 页的『mincommit － 对组的落实次数』
mirrorlogpath	否	否	低	SQLF_DBTN_MIRRORLOGPATH	806	char(242)	第 361 页的『mirrorlogpath － 镜像日志路径』
newlogpath	否	否	低	SQLF_DBTN_NEWLOGPATH	20	char(242)	第 362 页的『newlogpath － 更改数据库日志路径』
num_db_backups	是	否	无	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16	第 380 页的『num_db_backups － 数据库备份数』
num_estore_segs	否	否	中	SQLF_DBTN_NUM_ESTORE_SEGS	304	Sint32	第 341 页的『num_estore_segs － 扩充存储器内存段数目』
num_freqvalues	是	否	低	SQLF_DBTN_NUM_FREQVALUES	36	Uint16	第 399 页的『num_freqvalues － 保留的高频值数目』
num_iocleaners	否	否	高	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16	第 341 页的『num_iocleaners － 异步页清除程序的数目』
num_ioservers	否	否	高	SQLF_DBTN_NUM_IOSERVERS	39	Uint16	第 342 页的『num_ioservers － I/O 服务器数目』
num_log_span	是	是		SQLF_DBTN_NUM_LOG_SPAN	808	Uint16	第 363 页的『num_log_span － 跨越的日志数』
num_quantiles	是	否	低	SQLF_DBTN_NUM_QUANTILES	48	Uint16	第 400 页的『num_quantiles － 列的分位数的数目』
numarchretry	是	否	无	SQLF_DBTN_NUMARCHRETRY	827	Uint16	第 370 页的『numarchretry － 出错时的重试次数』
overflowlogpath	否	否	中	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)	第 363 页的『overflowlogpath － 溢出日志路径』
pckcachesz	是	否	高	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	第 313 页的『pckcachesz － 程序包高速缓存大小』
rec_his_retentn	否	否	无	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16	第 380 页的『rec_his_retentn － 恢复历史保留时间段』
seqdetect	是	否	高	SQLF_DBTN_SEQDETECT	41	Uint16	第 343 页的『seqdetect － 顺序检测标志』

表 43. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	性能影响	标记	标记值	数据类型	其它信息
sheapthres_shr	否	否	高	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32	第 314 页的『sheapthres_shr – 共享排序的排序堆阈值』
softmax	否	否	中	SQLF_DBTN_SOFTMAX	5	Uint16	第 371 页的『softmax – 恢复范围和软检查点时间间隔』
sortheap	是	否	高	SQLF_DBTN_SORT_HEAP	52	Uint32	第 324 页的『sortheap – 排序堆大小』
stat_heap_sz	否	否	低	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32	第 325 页的『stat_heap_sz – 统计信息堆大小』
stmtheap	是	否	中	SQLF_DBTN_STMT_HEAP	821	Uint32	第 326 页的『stmtheap – 语句堆大小』
trackmod	否	否	低	SQLF_DBTN_TRACKMOD	703	Uint16	第 381 页的『trackmod – 启用跟踪已修改的页』
tsm_mgmtclass	是	否	无	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)	第 381 页的『tsm_mgmtclass – Tivoli Storage Manager 管理类』
tsm_nodename	是	否	无	SQLF_DBTN_TSM_NODENAME	306	char(64)	第 382 页的『tsm_nodename – Tivoli Storage Manager 节点名』
tsm_owner	是	否	无	SQLF_DBTN_TSM_OWNER	305	char(64)	第 382 页的『tsm_owner – Tivoli Storage Manager 所有者名称』
tsm_password	是	否	无	SQLF_DBTN_TSM_PASSWORD	501	char(64)	第 383 页的『tsm_password – Tivoli Storage Manager 密码』
userexit	否	否	低	SQLF_DBTN_USER_EXIT	24	Uint16	第 372 页的『userexit – 启用用户出口』
util_heap_sz	是	否	低	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32	第 315 页的『util_heap_sz – 实用程序堆大小』
vendoropt	是	否	无	SQLF_DBTN_VENDOROPT	829	char(242)	第 373 页的『vendoropt – 供应商选项』

注:

1.
```
Default => Bit 1 on  (xxxx xxxx xxxx xxx1): auto_maint
           Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
           Bit 3 on  (xxxx xxxx xxxx x0xx): auto_tbl_maint
           Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
           Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
           Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
           Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
                        0    0    1    9

Maximum => Bit 1 on  (xxxx xxxx xxxx xxx1): auto_maint
           Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
           Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
           Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
           Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
           Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
           Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
                        0    0    7    F
```

2. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)
```

3. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_LOGRETAIN_NO (0)
SQLF_LOGRETAIN_RECOVERY (1)
SQLF_LOGRETAIN_CAPTURE (2)
```

表 44. 参考数据库配置参数

参数	标记	标记值	数据类型	其它信息
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16	第 394 页的『backup_pending – 备份暂挂指示符』
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	第 388 页的『codepage – 数据库的代码页』
codeset	SQLF_DBTN_CODESET	120	char(9)[1]	第 389 页的『codeset – 数据库的代码集』
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	第 389 页的『collate_info – 整理信息』
country	SQLF_DBTN_COUNTRY	100	Uint16	第 389 页的『country – 数据库地域代码』
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	第 394 页的『database_consistent – 数据库是一致的』
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	第 389 页的『database_level – 数据库发行版级别』
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	第 375 页的『hadr_db_role – HADR 数据库角色』
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	第 394 页的『log_retain_status – 日志保留状态指示符』
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	第 357 页的『loghead – 首个活动日志文件』
logpath	SQLF_DBTN_LOGPATH	103	char(242)	第 358 页的『logpath – 日志文件的位置』
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	第 394 页的『multipage_alloc – 启用的多页文件分配』
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16	第 343 页的『numsegs – SMS 容器的缺省数目』
release	SQLF_DBTN_RELEASE	102	Uint16	第 390 页的『release – 配置文件发行版级别』
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	第 395 页的『restore_pending – 复原暂挂』
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16	第 395 页的『rollfwd_pending – 前滚暂挂指示符』
territory	SQLF_DBTN_TERRITORY	121	char(5)[2]	第 390 页的『territory – 数据库地域』
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	第 395 页的『user_exit_status – 用户出口状态指示符』

表 44. 参考数据库配置参数 (续)

参数	标记	标记值	数据类型	其它信息
注: 1. char(17)，在 HP-UX 和 Solaris Operating Environment 上。 2. char(33)，在 HP-UX 和 Solaris Operating Environment 上。				

DB2 管理服务器（DAS）配置参数总结

表 45. DAS 配置参数

参数	参数类型	其它信息
authentication	可配置	第 440 页的『authentication － 认证类型 DAS』
contact_host	可联机配置	第 441 页的『contact_host － 联系人列表的位置』
das_codepage	可联机配置	第 441 页的『das_codepage － DAS 代码页』
das_territory	可联机配置	第 442 页的『das_territory － DAS 地域』
dasadm_group	可配置	第 442 页的『dasadm_group － DAS 管理权限组名』
db2system	可联机配置	第 443 页的『db2system － DB2 服务器系统的名称』
discover	可联机配置	第 443 页的『discover － DAS 发现方式』
exec_exp_task	可配置	第 444 页的『exec_exp_task － 执行已到期的任务』
jdk_64_path	可联机配置	第 445 页的『jdk_64_path － 64 位 Software Developer's Kit for Java 安装路径 DAS』
jdk_path	可联机配置	第 445 页的『jdk_path － Software Developer's Kit for Java 安装路径 DAS』
sched_enable	可配置	第 446 页的『sched_enable － 调度程序方式』
sched_userid	参考	第 446 页的『sched_userid － 调度程序用户标识』
smtp_server	可联机配置	第 447 页的『smtp_server － SMTP 服务器』
toolscat_db	可配置	第 447 页的『toolscat_db － 工具目录数据库』
toolscat_inst	可配置	第 448 页的『toolscat_inst － 工具目录数据库实例』
toolscat_schema	可配置	第 448 页的『toolscat_schema － 工具目录数据库模式』

按功能分类的参数详细信息

下面几节提供帮助您理解和调整不同配置参数的附加详细信息。对个别参数的讨论是根据它们的功能或用途来组织的:

- 第 306 页的『容量管理』
- 第 356 页的『日志记录和恢复』
- 第 387 页的『数据库管理』
- 第 403 页的『通信』
- 第 407 页的『分区数据库环境』
- 第 415 页的『实例管理』
- 第 440 页的『DB2 管理服务器』

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:
- *数据库管理器*影响数据库管理器的实例和在该实例内定义的所有数据库
- *数据库*影响特定数据库

参数类型
指示您是否可以更改参数值，以及更改是否会联机生效:
- *可配置*

  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或者从基准程序测试获得的经验来调整该参数。
- *可联机配置的*

  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／或从基准程序测试获得的经验来调整该参数。可以在数据库联机时应用更改，而不必停止并重新启动数据库管理器，或重新激活数据库。
- *参考*

  这些参数只能由数据库管理器自己更改，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信息。

容量管理

在数据库和数据库管理器级别有大量配置参数可影响您系统上的吞吐量。将这些参数按下列组分类:

- 『数据库共享内存』
- 第 315 页的『应用程序共享内存』
- 第 318 页的『代理进程专用内存』
- 第 326 页的『代理进程／应用程序通信内存』
- 第 330 页的『数据库管理器实例内存』
- 第 335 页的『锁定』
- 第 338 页的『I/O 和存储器』
- 第 344 页的『代理进程』
- 第 353 页的『存储过程和用户定义的函数』

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 307 页的『catalogcache_sz － 目录高速缓存大小』
- 第 308 页的『database_memory － 数据库共享内存大小』
- 第 309 页的『dbheap － 数据库堆』

- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 312 页的『logbufsz － 日志缓冲区大小』
- 第 313 页的『pckcachesz － 程序包高速缓存大小』
- 第 314 页的『sheapthres_shr － 共享排序的排序堆阈值』
- 第 315 页的『util_heap_sz － 实用程序堆大小』

catalogcache_sz － 目录高速缓存大小

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	-1 [8 – 524 288]
计量单位	页（4 KB）
分配时	当初始化数据库时
释放时	当数据库关闭时

此参数是在数据库共享内存外分配的，并且用于高速缓存系统目录信息。在分区数据库系统中，每个数据库分区都有一个目录高速缓存。

在各个分区中高速缓存目录信息允许数据库管理器降低其内部开销，方法是消除对存取系统目录（或分区数据库环境中的目录节点）以获取先前检索的信息的需要。目录高速缓存用来存储:

- SYSTABLES 信息（包括压缩的描述符）
- 授权信息，包括 SYSDBAUTH 信息和例程的执行特权
- SYSROUTINES 信息

使用目录高速缓存可以帮助提高下列各项的整体性能:

- 绑定程序包和编译 SQL 语句
- 涉及检查数据库级别特权的操作
- 涉及检查例程的执行特权的操作
- 连接至分区数据库环境中的非目录节点的应用程序

通过采用服务器或分区数据库环境中的缺省值（-1），用来计算页分配的值是为 *maxappls* 配置参数指定的值的四倍。如果 *maxappls* 的四倍小于 8 则出现异常。在此情况下，缺省值 -1 会将 *catalogcache_sz* 设置为 8。

建议: 开始使用缺省值，并使用数据库系统监视器来调整它。当调整此参数时，应考虑如果为目录高速缓存保留的额外内存是为另一目的分配的（如缓冲池或程序包高速缓存），它是否会更有效。

如果工作负载在短时间内涉及许多 SQL 编译，且其后 SQL 编译很少或没有，则调整此参数尤其重要。如果高速缓存太大，则内存可能会因保留不再使用的信息的副本而造成浪费。

在分区数据库环境中，考虑是否需要将目录节点上的 *catalogcache_sz* 设置为更大的值，因为非目录节点上要求的目录信息将总是首先在目录节点上高速缓存。

cat_cache_lookups（目录高速缓存查询）、*cat_cache_inserts*（目录高速缓存插入）、*cat_cache_overflows*（目录高速缓存溢出）和 *cat_cache_size_top*（目录高速缓存高水位标记）监视元素将帮助您确定是否应调整此配置参数。

注: 目录高速缓存在分区数据库环境中的所有节点上存在。因为每个节点都有本地数据库配置文件，所以每个节点的 *catalogcache_sz* 值定义本地目录高速缓存的大小。为了提供足够的高速缓存并避免溢出情况的发生，需要在每个节点上显式设置 *catalogcache_sz* 值，并考虑将非目录节点上的 *catalogcache_sz* 设置为比在目录节点上的该值小的可行性；记住将从目录节点的高速缓存中检索需要在非目录节点高速缓存的信息。因此，在非目录节点的目录高速缓存就象目录节点上的目录高速缓存中的信息的子集。

通常情况下，如果工作单元包含几个动态 SQL 语句或如果正在绑定包含许多静态 SQL 语句的程序包，则需要较多的高速缓存空间。

相关参考:

- 第 348 页的『maxappls － 最大活动应用程序数』
- 『cat_cache_lookups - Catalog Cache Lookups monitor element』（*System Monitor Guide and Reference*）
- 『cat_cache_inserts - Catalog Cache Inserts monitor element』（*System Monitor Guide and Reference*）
- 『cat_cache_overflows - Catalog Cache Overflows monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『cat_cache_size_top - Catalog Cache High Water Mark monitor element』（*System Monitor Guide and Reference*）

database_memory － 数据库共享内存大小

配置类型	数据库
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	Automatic [0 — 4 294 967 295]
计量单位	页（4 KB）
分配时	当数据库激活时
释放时	当取消激活数据库时

此参数指定为数据库共享内存区域保留的共享内存量。如果此数量小于根据个别参数（例如，锁定列表、实用程序堆和缓冲池等）计算而得的数量，则将使用较大的数量。

| 为了简化此参数的管理，AUTOMATIC 设置指示 DB2 计算需要的内存量，并在数据库激活时间分配它。DB2 还将为溢出缓冲区分配一些附加内存。每当堆超过为它配置的大小时，就使用溢出缓冲区来满足数据库共享内存区域中任何堆的峰值内存需求。其它操作（例如，动态配置更新）也对此溢出缓冲区具有存取权。可以使用带有 -memsets 选项的 **db2pd** 命令来监视溢出缓冲区中剩余的未使用的内存量。在 64 位的“DB2 AIX 版”上，数据库实际共享内存的使用量将动态地增长以适应数据库的需要（直至 64GB 的大小），因此不需要显式控制 DATABASE_MEMORY 参数。但是，设置 DB2_PINNED_BP 或 DB2_LGPAGE_BP 注册表变量将限制增大数据库共享内存。请参阅“性能变量”中有关这些注册表变量的描述。

建议: 此值通常保持为 AUTOMATIC。但是，可使用它来为将来的扩展保留附加内存。例如，附加内存可用于创建新的缓冲池或增大现有缓冲池的大小。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 469 页的『性能变量』
- 『db2pd - Monitor and Troubleshoot DB2 Command』（*Command Reference*）

dbheap － 数据库堆

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围]

UNIX 1200 [32 – 524 288]

带本地客户机和远程客户机的 Windows 数据库服务器 600 [32 – 524 288]

带本地客户机的 Windows 64 位数据库服务器 600 [32 – 524 288]

带本地客户机的 Windows 32 位数据库服务器 300 [32 – 524 288]

计量单位 页（4 KB）

分配时 当数据库激活时

释放时 当取消激活数据库时

每个数据库有一个数据库堆，并且数据库管理器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的空间（*logbufsz*）和实用程序使用的临时内存。因此堆大小将取决于许多变量。控制块信息保存在堆中，直到所有应用程序与数据库断开连接为止。

| 启动数据库管理器时需要的最小量是在第一次连接时分配的。根据需要扩充数据区，直到使用了数据库共享内存中的所有溢出内存区为止。

可以使用数据库系统监视器来跟踪用于使用 db_heap_top（分配的最大数据库堆）元素的数据库堆的最高内存量。

相关参考:

- 第 312 页的『logbufsz － 日志缓冲区大小』
- 『db_heap_top - Maximum Database Heap Allocated monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

locklist － 锁定列表的最大存储器

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围]

UNIX 100 [4 – 524 288]

带本地客户机和远程客户机的 Windows 数据库服务器 50 [4 – 524 288]

带本地客户机的 Windows 64 位数据库服务器 50 [4 – 60 000]

带本地客户机的 Windows 32 位数据库服务器 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连接时

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同时连接至数据库的所有应用程序所挂起的锁定。锁定是数据库管理器用来控制多个应用程序并发存取数据库中的数据的机制。行和表都可以锁定。数据库管理器还可以获取锁定来供内部使用。

可以联机更改此参数，但是只能联机增大它，而不能降低它。如果想要降低 *locklist* 的值，则必须重新激活数据库。

在 32 位平台上，每个锁定需要 36 或 72 个字节的锁定列表，这取决于该对象否挂起有其它锁定:

- 一个没有挂起其它锁定的对象挂起一个锁定需要 72 个字节
- 一个挂起有现有锁定的对象记录一个锁定需要 36 个字节。

在 64 位平台上，每个锁定需要 56 或 112 个字节的锁定列表，这取决于该对象否挂起有其它锁定:

- 要对没有其它锁定的对象保持锁定需要 112 个字节
- 记录对具有现有锁定的对象的锁定需要 56 个字节。

当一个应用程序使用的锁定列表的百分比达到 *maxlocks* 时，数据库管理器对应用程序所挂起的锁定执行从行到表的锁定升级（以下将描述）。虽然升级过程本身花不了多少时间，但是锁定整个表（与个别行比较）降低了并发性，并且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，防止其它锁定干扰更新，但却降低了数据的并发性。

  还可以使用 ALTER TABLE 语句的 LOCKSIZE 选项来控制如何对特定表进行锁定。

  使用“可重复读”隔离级别可能会导致自动表锁定。
- 只要有可能，使用“游标稳定性”隔离级别以减少所挂起的共享锁定数。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的并发性。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: 如果锁定升级导致性能问题，可能需要增大此参数或 *maxlocks* 参数的值。可以使用数据库系统监视器来确定是否正在发生锁定升级。参阅 *lock_escals*（*锁定升级*）监视元素。

下列步骤可以帮助确定锁定列表所需的页数:

1. 根据您的环境，使用下列计算的*其中一种*计算来为锁定列表的大小计算下限:
   a. `(512 * x * maxappls) / 4096`
   b. 启用“集中器”:

      `(512 * x * max_coordagents) / 4096`
   c. 在启用“集中器”的分区数据库中:

      `(512 * x * max_coordagents * number of database partitions) / 4096`

   其中 512 是每个应用程序的平均锁定数目的估计值，而 `x` 是针对具有现有锁定的对象的每个锁定所需要的字节数（在 32 位平台上需要 36 位，则 64 位平台上需要 56 位）。
2. 计算锁定列表大小的上限:

   `(512 * y * maxappls) / 4096`

   其中 `y` 是针对对象的第一个锁定所需的字节数（在 32 位平台上需要 72 位，在 64 位平台上需要 112 位）。
3. 估计数据将发生的并发性数量，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，使用数据库系统监视器调整此参数的值。

可以使用数据库系统监视器来确定给定的事务挂起的最大锁定数。参阅 *locks_held_top*（*挂起的最大锁定数*）监视元素。

此信息可帮助您验证或调整每个应用程序的估计锁定数。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视信息。

如果增大了 *maxappls*，或者如果正运行的应用程序执行的落实不频繁，则可能也要增大 *locklist*。

在更改此参数之后，应考虑重新绑定应用程序（使用 REBIND 命令）。

相关参考:

- 第 336 页的『maxlocks – 升级前锁定列表的最大百分比』
- 第 348 页的『maxappls – 最大活动应用程序数』
- 『lock_escals - Number of Lock Escalations monitor element』（*System Monitor Guide and Reference*）
- 『locks_held_top - Maximum Number of Locks Held monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『REBIND Command』（*Command Reference*）

logbufsz – 日志缓冲区大小

配置类型	数据库	
参数类型	可配置	
缺省值 [范围]		
	32 位平台	8 [4 — 4 096]
	64 位平台	8 [4 — 65 535]
计量单位	页（4 KB）	

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
- 发生了某些其它内部数据库管理器事件。

此参数也必须小于或等于 *dbheap* 参数。缓冲日志记录将使日志记录文件 I/O 更有效，因为将日志记录写入磁盘的频率越低，则每次可写入的日志记录就越多。

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，则要增大此缓冲区的大小。当增大此参数的值时，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视器来确定将多少日志缓冲区空间用于特定事务（或工作单元）。参阅 *log_space_used*（使用的工作单元日志空间）监视元素。

相关参考:

- 第 369 页的『mincommit – 对组的落实次数』

- 第 309 页的『dbheap － 数据库堆』
- 『uow_log_space_used - Unit of Work Log Space Used monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

pckcachesz － 程序包高速缓存大小

配置类型	数据库	
参数类型	可联机配置	
传播类	立即	
缺省值 [范围]		
	32 位平台	-1 [-1, 32 — 128 000]
	64 位平台	-1 [-1, 32 — 524 288]
计量单位	页（4 KB）	
分配时	当初始化数据库时	
释放时	当数据库关闭时	

此参数是在数据库共享内存之外分配的，并且用于高速缓存数据库上的静态和动态 SQL 语句的部分。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器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 语句节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，应考虑如果为程序包高速缓存保留的额外内存是为另一目的分配的（如缓冲池或目录高速缓存），它是否会更有效。因此，应在调整此参数时使用基准程序技术。

当最初使用几节，而后只有少数几节反复运行时，调整此参数就特别重要。如果高速缓存太大，则因保存初始节的副本而浪费内存。

下列监视元素可以帮助您确定是否应调整此配置参数:

- *pkg_cache_lookups*（程序包高速缓存查询数）
- *pkg_cache_inserts*（程序包高速缓存插入数）

- *pkg_cache_size_top*（程序包高速缓存高水位标记）
- *pkg_cache_num_overflows*（程序包高速缓存溢出）

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 语句的所有节。如果分配的空间比当前需要的空间多，则各节被高速缓存。下一次需要这些部分时，只需执行它们而不必装入或编译它们。

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* 监视元素确定程序包高速缓存达到的最大值，并用 *pkg_cache_num_overflows* 监视元素确定超过了由 *pckcachesz* 参数指定的限制的多少倍。

相关参考:

- 第 348 页的『maxappls － 最大活动应用程序数』
- 『pkg_cache_lookups - Package Cache Lookups monitor element』（*System Monitor Guide and Reference*）
- 『pkg_cache_inserts - Package Cache Inserts monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『pkg_cache_num_overflows - Package Cache Overflows monitor element』（*System Monitor Guide and Reference*）
- 『pkg_cache_size_top - Package Cache High Water Mark monitor element』（*System Monitor Guide and Reference*）

sheapthres_shr － 共享排序的排序堆阈值

配置类型	数据库	
参数类型	可配置	
缺省值 [范围]		
	32 位平台	*sheapthres* [250 — 2 097 152]
	64 位平台	*sheapthres* [250 — 2 147 483 647]
计量单位	页（4 KB）	

此参数表示对可一次用于排序的数据库共享内存总量的硬限制。当活动共享排序的共享内存总量达到此限制时，后续排序将失败（SQL0955C）。如果 *sheapthres_shr* 的值为 0，则共享排序内存的阈值将等于 *sheapthres* 数据库管理器配置参数的值，此参数值也用来表示专用排序的排序内存阈值。如果 *sheapthres_shr* 的值是非零值，则此非零值将用于共享排序内存阈值。

sheapthres_shr 仅在两种情况是有意义的;

- 在 *intra_parallel* 数据库管理器配置参数设置为 *yes* 的情况下，因为当 *intra_parallel* 设置为 *no* 时，将没有任何共享排序。

- 在“集中器”打开（即，当 *max_connections* 大于 *max_coordagents*）的情况下，因为使用以 WITH HOLD 选项声明的游标的排序将从共享内存中分配。

相关参考:

- 第 323 页的『sheapthres – 排序堆阈值』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

util_heap_sz – 实用程序堆大小

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	5000 [16 – 524 288]
计量单位	页（4 KB）
分配时	数据库管理器实用程序需要时
释放时	当实用程序不再需要内存时

此参数指示可由 BACKUP、RESTORE 和 LOAD（包括装入恢复）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，在这种情况下应增大此值。如果系统上的内存受约束，则您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，并且溢出区域没有更多内存可用，则可能不能同时运行多个实用程序。应该根据需要动态更新此参数。如果实用程序较少，则将此参数设置为较小的值。如果实用程序较多，或者实用程序消耗的内存较多，则应该将此参数设置为较大的值。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

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进程（协调代理进程和子代理进程）使用的工作区:

- 『app_ctl_heap_sz – 应用程序控制堆大小』
- 第 316 页的『appgroup_mem_sz – 应用程序组内存集的最大大小』
- 第 317 页的『groupheap_ratio – 应用程序组堆的内存百分比』

app_ctl_heap_sz – 应用程序控制堆大小

配置类型	数据库
参数类型	可配置
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]（适用于非 UNIX 平台）

128 [1–64 000]（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
512 [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

对于分区数据库以及启用了内部并行性（intra_parallel=ON）的非分区数据库，此参数指定分配给应用程序的共享内存区域的平均大小。对于禁用内部并行性的非分区数据库（intra_parallel=OFF），这是将为堆分配的最大专用内存。每个分区的每个连接有一个应用程序控制堆。

主要是在代表同一请求工作的代理进程之间共享信息时需要此应用程序控制堆。对于非分区数据库，当运行带有并行度等于 1 的查询时，此堆所使用的资源最少。

这个堆还用来存储已声明临时表的描述符信息。所有尚未显式删除的已声明临时表的描述符信息存放在这个堆的内存中，在删除已声明临时表之前，不能删除这些信息。

建议： 最初，从缺省值开始。如果在运行复杂的应用程序或您的系统包含大量的数据库分区，又或者您使用已声明临时表，则可能需要将该值设置得高一些。所需的内存量随已声明同时活动的临时表数的增加而增加。与带有较少列的表相比，带有许多列的已声明临时表的表描述符大小要大，因此，如果应用程序的已声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。

相关参考:

- 第 413 页的『intra_parallel － 启用分区内并行性』
- 第 319 页的『applheapsz － 应用程序堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 316 页的『appgroup_mem_sz － 应用程序组内存集的最大大小』
- 第 317 页的『groupheap_ratio － 应用程序组堆的内存百分比』

appgroup_mem_sz － 应用程序组内存集的最大大小

配置类型
数据库

参数类型
可配置

缺省值 [范围]

带有本地客户机的 UNIX 数据库服务器（除了 32 位 HP-UX 以外） 20 000 [1 – 1 000 000]

32 位 HP-UX

- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

10 000 [1 – 1 000 000]

带本地客户机的 Windows 数据库服务器

10 000 [1 – 1 000 000]

带本地客户机和远程客户机的数据库服务器（除了 32 位 HP-UX 以外）

30 000 [1 – 1 000 000]

带本地客户机和远程客户机的分区数据库服务器（除了 32 位 HP-UX 以外）

40 000 [1 – 1 000 000]

计量单位 页（4 KB）

此参数确定应用程序组共享内存段的大小。需要在工作在同一应用程序上的代理进程之间共享的信息存储在应用程序组共享内存段中。

在分区数据库中，或在启用了分区内并行性或集中器的非分区数据库中，多个应用程序共享一个应用程序组。为应用程序组分配了一个应用程序组共享内存段。在应用程序组共享内存段中，每个应用程序将有它自己的应用程序控制堆，并且所有应用程序将共享一个应用程序组共享堆。

一个应用程序组中的应用程序的数目通过以下公式计算:

```
appgroup_mem_sz / app_ctl_heap_sz
```

应用程序组共享堆大小通过以下公式计算:

```
appgroup_mem_sz * groupheap_ratio / 100
```

每个应用程序控制堆的大小通过以下公式计算:

```
app_ctl_heap_sz * (100 - groupheap_ratio) / 100
```

建议: 保留此参数的缺省值，除非遇到性能问题。

相关参考:

- 第 315 页的『app_ctl_heap_sz – 应用程序控制堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 317 页的『groupheap_ratio – 应用程序组堆的内存百分比』

groupheap_ratio – 应用程序组堆的内存百分比

配置类型 数据库

参数类型 可配置

缺省值 [范围] 70 [1 – 99]

计量单位 百分比

此参数指定用于应用程序组共享堆的应用程序控制共享内存集中的内存百分比。

此参数对于集中器关闭且禁用分区内并行性的非分区数据库不起任何作用。

建议: 保留此参数的缺省值，除非遇到性能问题。

相关参考:

- 第 315 页的『app_ctl_heap_sz – 应用程序控制堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 316 页的『appgroup_mem_sz – 应用程序组内存集的最大大小』

代理进程专用内存

下列参数影响每个数据库代理进程使用的内存量:

- 『agent_stack_sz – 代理进程堆栈大小』
- 第 319 页的『applheapsz – 应用程序堆大小』
- 第 320 页的『min_priv_mem – 最小已落实专用内存』
- 第 321 页的『priv_mem_thresh – 专用内存阈值』
- 第 322 页的『query_heap_sz – 查询堆大小』
- 第 323 页的『sheapthres – 排序堆阈值』
- 第 324 页的『sortheap – 排序堆大小』
- 第 325 页的『stat_heap_sz – 统计信息堆大小』
- 第 326 页的『stmtheap – 语句堆大小』

agent_stack_sz – 代理进程堆栈大小

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] 16 [8 – 1000]

计量单位 页（4 KB）

分配时 当初始化代理进程为应用程序工作时

释放时 当代理进程完成应用程序要做的工作时

代理进程堆栈是 DB2 为每个代理进程分配的虚拟内存。当需要此内存以处理 SQL 语句时，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率。与用于简单查询的空间相比，越复杂的查询将使用越多的堆栈空间。

此参数用来设置在 Windows 环境中的每个代理进程的初始落实堆栈大小。缺省情况下，每个代理进程堆栈可增长至缺省保留堆栈大小 256 KB（64 个 4 KB 页）。此限制对于大多数数据库操作已足够了。但是，当准备大型 SQL 语句时，代理进程可能用完堆栈空间，且系统将生成堆栈溢出异常（0xC000000D）。当发生此情况时，服务器将关闭，因为该错误是不可恢复的。

可以通过将 *agent_stack_sz* 的值设置为大于 64 页的缺省保留堆栈大小来增大代理进程堆栈大小。注意，当 *agent_stack_sz* 的值大于缺省保留堆栈大小时，Windows 操作系统会将其四舍五入为最接近的 1 MB 的倍数；将代理进程堆栈大小设置为 128 个 4 KB 的页实际上为每个代理进程保留 1 MB 的堆栈。将 *agent_stack_sz* 的值设置为小于缺省保留堆栈大小不会对最大限制产生影响，因为在必要时，堆栈将增长至缺省保留堆栈大小。在此情况下，*agent_stack_sz* 的值是创建代理进程时堆栈的初始落实内存。

可以通过使用 db2hdr 实用程序更改 `db2syscs.exe` 文件的头信息来更改缺省保留堆栈大小。更改缺省保留堆栈大小将影响所有线程，而更改 *agent_stack_sz* 仅影响代理进程的堆栈大小。使用 db2hdr 实用程序更改缺省堆栈大小的优点是提供更佳的粒度，从而允许以最小的必要堆栈大小来设置堆栈大小。但是，您必须停止并重新启动 DB2 以便使对 `db2syscs.exe` 的更改生效。

建议：在大多数情况下，应可使用缺省堆栈大小。仅当环境包括许多很复杂的查询时，才需要增大此参数的值。

如果您的环境符合下列条件，可能可以减小堆栈大小以使更多地址空间可供其它客户机使用：

- 仅包含简单应用程序（例如，小型的 OLTP），其中从不会有复杂查询
- 需要相当大量的并发客户机（例如，超过 100 个）。

代理进程堆栈大小和并发客户机数是成反比的：堆栈大小越大，则可运行的并发客户机的可能数量就越少。发生这种情况的原因是 Windows 平台上的地址空间有限。

此参数不适用于基于 UNIX 的平台。

相关参考：

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

applheapsz – 应用程序堆大小

配置类型 数据库

参数类型 可配置

缺省值 [范围]

带本地客户机和远程客户机的 32 位数据库服务器
256 [16 – 60 000]

带本地客户机和远程客户机的 64 位数据库服务器
256 [16 – 60 000]

带本地客户机和远程客户机的 32 位分区数据库服务器
64 [16 – 60 000]

带本地客户机和远程客户机的 64 位分区数据库服务器
128 [16 – 60 000]

计量单位
页（4 KB）

分配时
当初始化代理进程为应用程序工作时

释放时
当代理进程完成为应用程序所作的工作时

此参数定义可用的专用内存页数目，它们由数据库管理器作为特定代理进程或子代理进程进行使用。

该堆是在为应用程序初始化代理进程或子代理进程时分配的。分配的量将是处理已经给予代理进程或子代理进程的请求所需要的最小量。由于代理进程或子代理进程需要更多的堆空间来处理较大的 SQL 语句，数据库管理器将按需要分配内存，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*）来存储代理进程和子代理进程的 SQL 语句的执行节副本。然而，SMP 子代理进程象其它所有环境中的代理进程一样使用 *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空间，则增大此参数的值。

在代理进程专用内存之外分配应用程序堆（*applheapsz*）。

相关参考:

- 第 315 页的『app_ctl_heap_sz － 应用程序控制堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

min_priv_mem － 最小已落实专用内存

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
32 [32 – 112 000]

计量单位
页（4 KB）

分配时
当启动数据库管理器时

释放时
当停止数据库管理器时

此参数指定当启动（db2start）数据库管理器实例时，数据库服务器进程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。

相关参考:

- 第 321 页的『priv_mem_thresh － 专用内存阈值』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

priv_mem_thresh － 专用内存阈值

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	20 000 [-1; 32 – 112 000]
计量单位	页（4 KB）

此参数用于确定未用过的代理进程专用内存量，这些内存将处于分配状态，准备供启动的新代理进程使用。此参数不适用于基于 UNIX 的平台。

值 -1 将导致此参数使用 *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／断开连接模式，以及同一台机器上其它进程的内存需求。

如果仅在一个短暂的时期内有很多客户机同时与数据库连接，则高阈值将防止未使用的内存被取消落实，从而被其它进程使用。这种情况将导致内存管理混乱，从而会影响其它需要内存的进程。

如果并发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释放内存的开销。

相关参考:

- 第 320 页的『min_priv_mem － 最小已落实专用内存』

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

query_heap_sz － 查询堆大小

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
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，或与实例拆离时

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进程专用内存中。每个查询的信息由输入和输出 SQLDA、语句文本、SQLCA、程序包名、创建程序、节号以及一致性标记组成。提供此参数以防止应用程序不必要地消耗代理进程内部大量的虚拟内存。

查询堆也用于分配给分块游标的内存。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，则可能需要增大此参数的值，以便查询堆的大小足够容纳那些 LOB。

相关参考:

- 第 326 页的『aslheapsz － 应用程序支持层堆大小』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

sheapthres － 排序堆阈值

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]

UNIX 32 位平台
20 000 [250 — 2 097 152]

Windows 平台
10 000 [250 — 2 097 152]

64 位平台
20 000 [250 — 2 147 483 647]

计量单位
页（4 KB）

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，为附加入站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，将不允许其它共享排序内存请求（直到总共享排序内存的占用降低至 *sheapthres* 指定的限制以下为止）。（在某些情况下，配置共享排序最大值的另一种方法是使用 *sheapthres_shr* 数据库配置参数。）

使用排序堆的操作的示例包括：排序、散列连接、动态位图（用于索引 AND 运算和星型连接）和表位于内存中的操作。

显式定义阈值可防止数据库管理器对大量排序使用过量内存。

当从非分区数据库移至分区数据库环境时，不应增大此参数的值。一旦在单个数据库分区环境中调整了数据库和数据库管理器配置参数，在大多数情况下，相同的值在分区数据库环境将同样合适。

作为数据库管理器配置参数的“排序堆阈值”参数应用于整个 DB2 实例。将该参数设置为不同节点或分区上的不同值的唯一方法是创建多个 DB2 实例。这将需要通过不同数据库分区组管理不同的 DB2 数据库。这样安排将无法发挥分区数据库环境的许多优点。

建议: 理想情况下，应将此参数设置为您在数据库管理器实例中拥有的最大 *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发代理进程的典型数目）
  * （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。

应使用基准程序技术来调整此参数以找到在排序性能和内存使用之间的相应平衡。

可以使用数据库系统监视器并借助后阈值排序（*post_threshold_sorts*）监视元素来跟踪排序活动。

相关参考:

- 第 324 页的『sortheap – 排序堆大小』
- 『post_threshold_sorts - Post Threshold Sorts monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 第 314 页的『sheapthres_shr – 共享排序的排序堆阈值』

sortheap – 排序堆大小

配置类型
数据库

参数类型
可联机配置

传播类
语句边界

缺省值 [范围]

32 位平台
256 [16 – 524 288]

64 位平台
256 [16 – 4 194 303]

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

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进程专用内存。如果排序为共享排序，则此参数将影响数据库共享内存。每个排序都有一个独立的排序堆，该排

序堆是由数据库管理器按照需要分配的。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：当使用排序堆时，应该考虑下列事项：

- 适当的索引可使排序堆的使用减至最小程度。
- 散列连接缓冲区和动态位图（用于索引 AND 计算和星型连接）使用排序堆内存。在使用这些技术时，增大此参数的大小。
- 当需要进行频繁的大型排序时，增大此参数的大小。
- 当增大此参数的值时，应该检查是否还需要调整数据库管理器配置文件中的 *sheapthres* 参数。
- 排序堆大小由优化器在确定存取路径时使用。在更改此参数后，应考虑重新绑定应用程序（使用 REBIND 命令）。

相关参考：

- 第 323 页的『sheapthres － 排序堆阈值』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『REBIND Command』（*Command Reference*）
- 第 314 页的『sheapthres_shr － 共享排序的排序堆阈值』

stat_heap_sz － 统计信息堆大小

配置类型	数据库
参数类型	可配置
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

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信息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，则您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。

相关参考：

- 第 399 页的『num_freqvalues － 保留的高频值数目』
- 第 400 页的『num_quantiles － 列的分位数的数目』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）

- 『RUNSTATS Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

stmtheap － 语句堆大小

配置类型	数据库
参数类型	可联机配置
传播类	语句边界
缺省值 [范围]	2048 [128 – 65 535]
计量单位	页（4 KB）
分配时	对于预编译或绑定期间的每个语句
释放时	当每个语句的预编译或绑定完成时

语句堆在 SQL 语句的编译期间用作 SQL 编译器的工作空间。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，但要对每个处理的 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，在绑定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，并且数据库管理器在尝试优化语句时发出一条错误信息（语句太复杂），则应按固定增量（例如，256 或 1024）增大此参数的值，直至消除此错误情况。

相关参考：
- 第 324 页的『sortheap － 排序堆大小』
- 第 319 页的『applheapsz － 应用程序堆大小』
- 第 325 页的『stat_heap_sz － 统计信息堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

代理进程／应用程序通信内存

下列参数影响为了在应用程序和代理进程之间传送数据而分配的内存量：
- 『aslheapsz － 应用程序支持层堆大小』
- 第 328 页的『min_dec_div_3 － 十进制分隔小数位为 3』
- 第 329 页的『rqrioblk － 客户机 I/O 块大小』

aslheapsz － 应用程序支持层堆大小

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器

参数类型	可配置
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器代理进程时
释放时	当数据库管理器代理进程终止时

应用程序支持层堆表示本地应用程序和其关联的代理进程之间的通信缓冲区。此缓冲区被分配为每个已启动的数据库管理器代理进程所共享的内存。

如果对数据库管理器的请求或其相关联的应答不适合该缓冲区，则该请求和应答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 个字节的其它字段。

除了此通信缓冲区外，此参数也用于两个其它目的:

- 它用来在分块游标打开时确定 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为在应用程序的专用内存之外的分块游标分配空间，则将打开非分块游标。
- 它用来确定代理进程和 db2fmp 进程之间的通信大小。（db2fmp 进程可以是用户定义的函数或受防护的存储过程。）字节的数目是从系统上活动的每个 db2fmp 进程或线程的共享内存分配的。

数据库管理器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，则您可能希望增大此参数的值。

使用如下公式计算 *aslheapsz* 的最小页数:

```
aslheapsz >= ( sizeof(input SQLDA)
             + sizeof(each input SQLVAR)
             + sizeof(output SQLDA)
             + 250 ) / 4096
```

其中 sizeof(x) 是 x 的字节大小，它计算给定输入或输出值的页数。

还应考虑此参数对分块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 4 096 个字节），则大行块可能获得更佳性能。然而，由于较大的记录块会增大每个连接的工作集内存大小，所以要加以折衷。

大记录块还可能导致比应用程序实际需要的更多的访存请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制访存请求数。

相关参考:

- 第 322 页的『query_heap_sz – 查询堆大小』
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

min_dec_div_3 – 十进制分隔小数位为 3

配置类型	数据库
参数类型	可配置
缺省值 [范围]	No [Yes, No]

min_dec_div_3 数据库配置参数用作启动对 SQL 中十进制小数位的计算更改的快速方法。*min_dec_div_3* 可以设置为“Yes”或“No”。 *min_dec_div_3* 的缺省值为“No”。

min_dec_div_3 数据库配置参数会更改涉及小数的十进制算术运算的结果小数位。如果值为“No”，则小数位计算为 31-p+s-s'。如果设置为“Yes”，则小数位计算为 MAX(3, 31-p+s-s')。这会导致十进制小数部分始终至少具有 3 位的小数位。精度始终为 31。

更改此数据库配置参数可能会导致更改现有数据库的应用程序。当更改此数据库配置参数会影响十进制小数部分的结果小数位时，就可能发生此情况。下面列示的是可能会影响应用程序的一些可能的情况。在使用现有数据库的数据库服务器上更改 *min_dec_div_3* 之前应该考虑这些情景说明。

- 如果更改其中一个视图列的结果小数位，则对于在某一环境中用一个设置定义的视图，当更改数据库配置参数之后引用该视图时，该视图可能失败，返回 SQLCODE -344。消息 SQL0344N 指的是递归公共表表达式，然而，如果对象名(第一个标记)是一个视图，则您将需要删除此视图并再次创建它，以避免此错误。
- 在隐式或显式重新绑定静态程序包之后，静态程序包才会更改行为。例如，将值从 NO 更改为 YES 之后，在执行重新绑定之后，才会在结果中包括附加的小数位。对于任何已更改的静态程序包，显式 REBIND 命令可以用来强制重新绑定。
- 涉及十进制小数部分的检查约束可能会限制先前已接受的某些值。这种行现在违犯约束，但在更新涉及检查约束行的其中一列或处理带有 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句之前检测不到。要强制检查这样的约束，执行 ALTER TABLE 语句以删除检查约束，然后执行 ALTER TABLE 语句来再次添加该约束。

注: *min_dec_div_3* 还有下列限制:

1. 命令 `GET DB CFG FOR DBNAME` 将不显示 *min_dec_div_3* 设置。确定当前设置的最好方法是观察十进制小数结果的副作用。例如，考虑如下语句:

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   如果此语句返回 SQL 代码 SQL0419N，则该数据库不具有 *min_dec_div_3* 支持，或者已将它设置为“No”。如果该语句返回 `1.000`，则 *min_dec_div_3* 设置为“Yes”。

2. 当运行下列命令时，在配置关键字列表中不会出现 *min_dec_div_3*: ? `UPDATE DB CFG`

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

rqrioblk – 客户机 I/O 块大小

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
32 767 [4 096 – 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

- 当远程应用程序与服务器数据库断开连接时
- 当关闭分块游标时

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进程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，最初分配一个大小为 32 767 个字节的通信缓冲区，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 32 767 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，此参数还用来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为在应用程序的专用内存之外的分块游标分配空间，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块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 4 096 个字节），则大行块可能获得更佳性能。然而，由于较大的记录块会增大每个连接的工作集内存大小，所以要加以折衷。

大记录块还可能导致比应用程序实际需要的更多的访存请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制访存请求数。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

数据库管理器实例内存

下列参数影响在实例级分配和使用的内存:

- 『audit_buf_sz – 审计缓冲区大小』
- 第 331 页的『dir_cache – 目录高速缓存支持』
- 第 332 页的『instance_memory – 实例内存』
- 第 333 页的『java_heap_sz – 最大 Java 解释器堆大小』
- 第 334 页的『mon_heap_sz – 数据库系统监视器堆大小』

audit_buf_sz – 审计缓冲区大小

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

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

此参数指定当审计数据库时使用的缓冲区的大小。

此参数的缺省值是零(0)。如果该值为零(0),则不使用该审计缓冲区。如果该值大于零(0),就会为审计缓冲区分配空间,以用于放置审计设施所生成的审计记录。该值乘以 4KB 页就是为审计缓冲区分配的空间容量。不能动态分配该审计缓冲区;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,该审计设施将记录写入磁盘与执行生成审计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,审计设施将记录写入磁盘与执行生成审计记录的语句将同步进行(同时进行)。审计期间进行同步操作会降低在 DB2 中运行的应用程序的性能。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

dir_cache － 目录高速缓存支持

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
Yes [Yes; No]

分配时

- 当应用程序发出它的第一个连接时，分配应用程序目录高速缓存
- 当启动（db2start）数据库管理器实例时，分配服务器目录高速缓存。

释放时

- 当应用程序进程终止时，释放应用程序目录高速缓存
- 当停止（db2stop）数据库管理器实例时，释放服务器目录高速缓存。

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:

- 分配并用于应用程序在其上运行的机器上的每个应用程序进程的应用程序目录高速缓存。
- 分配并用于一些内部数据库管理器进程的服务器目录高速缓存。

对于应用程序目录高速缓存，当应用程序发出它的第一个连接时，会读取每个目录文件且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。如果未在应用程序目录高速缓存中发现数据库，则将搜索目录文件以查找该信息，但是不更新高速缓存。如果应用程序修改目录条目，则该应用程序中的下一个连接将导致刷新此应用程序的高速缓存。不会刷新其它应用程序的应用程序目录高速缓存。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会话所使用的目录高速缓存，发出 db2 terminate 命令。）

对于服务器目录高速缓存，当启动（db2start）数据库管理器实例时，会读取每个目录文件并且将信息高速缓存在服务器内存中。将保留此高速缓存，直到该实例停止

（db2stop）。如果此高速缓存中找不到某个目录条目，则搜索目录文件以获取信息。在实例运行期间，从不刷新此服务器目录高速缓存。

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。

目录高速缓存还可以提高获取数据库系统监视器快照的性能。另外，应显式引用快照调用中的数据库名称，而不使用数据库别名。

注： 如果目录高速缓存打开并且在启动数据库管理器之后编目、取消编目、创建或删除数据库，则在执行快照调用时可能会发生错误。

相关参考：

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

instance_memory － 实例内存

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	Automatic [8 - 524 288]
计量单位	页（4 KB）
分配时	当实例启动时
释放时	当实例停止时

此参数指定应该为进行实例管理而保留的内存量。这包括描述实例上的数据库的内存区域。

| 如果将此参数设置为 AUTOMATIC，则 DB2 将计算当前配置所需要的实例内存量。
| DB2 还将为溢出缓冲区分配一些附加内存。每当堆超过为它配置的大小时，就使用溢出
| 缓冲区来满足实例共享内存区域中任何堆的峰值内存需求。其它操作（例如，动态配
| 置更新）也对此溢出缓冲区具有存取权。可以使用带有 -memsets 选项的 **db2pd** 命令
| 来监视溢出缓冲区中剩余的未使用的内存量。

相关参考：

- 第 347 页的『maxagents － 最大代理进程数』

- 第 423 页的『numdb － 同时活动数据库的最大数目（包括主机数据库和 iSeries 数据库管理器）』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『db2pd - Monitor and Troubleshoot DB2 Command』（*Command Reference*）

java_heap_sz － 最大 Java 解释器堆大小

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
512 [0 - 524 288]

计量单位
页（4 KB）

分配时
当 Java 存储过程或 UDF 启动时

释放时
当 db2fmp 进程（受防护的）或 db2agent 进程（可信的）终止时。

此参数确定由已启动以便为 Java DB2 存储过程和 UDF 提供服务的 Java 解释器使用的堆的最大大小。

每个 DB2 进程都有一个堆（一个用于在基于 UNIX 的平台上的代理进程或子代理进程，一个用于在其它平台上的每个实例）。每个受防护的 UDF 和受防护的存储过程进程都有一个堆。可信的例程的每个代理进程（不包括子代理进程）都有一个堆。每个运行 Java 存储过程的 db2fmp 进程都有一个堆。对于多线程 db2fmp 进程，使用线程安全受防护的例程的多个应用程序是由单个堆提供服务的。在所有情况下，只有运行 Java UDF 或存储过程的代理进程或进程才分配此内存。在分区数据库系统上，在每个分区上使用相同值。

相关参考:

- 第 422 页的『jdk_path － Software Developer's Kit for Java 安装路径』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

mon_heap_sz － 数据库系统监视器堆大小

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围]

UNIX 90 [0 – 60 000]

带本地客户机和远程客户机的 Windows 数据库服务器 66 [0 – 60 000]

带本地客户机的 Windows 数据库服务器 46 [0 – 60 000]

计量单位 页（4 KB）

分配时 当使用 *db2start* 命令启动数据库管理器时

释放时 当使用 *db2stop* 命令停止数据库管理器时

此参数确定分配给数据库系统监视器数据的内存量（以页计）。当进行数据库监视活动（例如，生成快照、调整监视器开关、重新设置监视器或激活事件监视器）时，从监视器堆分配内存。

如果值为零，则将阻止数据库管理器收集数据库系统监视器数据。

建议: 监视活动所需的内存量取决于监视应用程序（捕获快照的应用程序或事件监视器）的数目、设置了哪些开关以及数据库活动的级别。

| 如果此堆中的可用内存都用尽，并且溢出缓冲区没有任何未使用的内存，则将发生下列其中一种情况:

- 当第一个应用程序连接至定义了此事件监视器的数据库时，会将一条错误消息写入到管理通知日志。
- 如果使用 SET EVENT MONITOR 语句动态启动的事件监视器失败，则向您的应用程序返回错误代码。
- 如果一个监视器命令或 API 子例程失败，则向您的应用程序返回错误代码。

相关概念:

- 『Database system monitor memory requirements』（*System Monitor Guide and Reference*）

相关参考:

- 第 418 页的『dft_monswitches － 缺省数据库系统监视器开关』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

锁定

下列参数影响在环境中如何管理锁定:

- 『dlchktime － 检查死锁的时间间隔』
- 『locktimeout － 锁定超时』
- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』

另请参阅第 310 页的『locklist － 锁定列表的最大存储器』。

dlchktime － 检查死锁的时间间隔

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	10 000（10 秒）[1 000 – 600 000]
计量单位	毫秒

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挂起对方执行所需要的资源，所以该等待永远得不到解决。

死锁检查时间间隔定义数据库管理器检查所有与数据库连接的应用程序间的死锁的频率。

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: 增大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，但是会增加数据库管理器检查死锁所花的时间。如果死锁时间间隔太小，可能会降低运行时性能，因为数据库管理器频繁执行死锁检测。如果将此参数设置得较低以改善并发性，则应确保适当地设置 *maxlocks* 和 *locklist*，以避免不必要的锁定升级，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。

相关参考:

- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 336 页的『maxlocks － 升级前锁定列表的最大百分比』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

locktimeout － 锁定超时

配置类型	数据库

参数类型 可配置

缺省值 [范围] -1 [-1; 0 – 32 767]

计量单位 秒

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，您可以从一个较高的值开始。在这两种情况下，您都应该使用基准程序技术来调整此参数。

当使用 Data Links Manager 时，如果您发现 Data Links Manager（dlfm）实例的管理通知日志中发生锁定超时，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载（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视器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。

lock_timeout（锁定超时的数目）监视元素的高值可能是下列原因造成的:

- 此配置参数的值太低。
- 挂起锁定较长时间的应用程序（事务）。可以使用数据库系统监视器来进一步调查这些应用程序。
- 并发性问题，它可能是由锁定升级（从行级别至表级别锁定）引起的。

相关参考:

- 第 310 页的『locklist – 锁定列表的最大存储器』
- 第 336 页的『maxlocks – 升级前锁定列表的最大百分比』
- 『lock_timeouts - Number of Lock Timeouts monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

maxlocks – 升级前锁定列表的最大百分比

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围]

UNIX	10 [1 – 100]
Windows	22 [1 – 100]

计量单位 百分比

锁定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所挂起的锁定列表的百分比，必须在数据库管理器执行升级之前填写该列表。当任何一个应用程序所挂起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挂起锁定的锁定升级。当锁定列表空间用尽时也会发生锁定升级。

数据库管理器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，锁定升级将停止。否则，将继续升级，直到保持的锁定列表的百分比低于 *maxlocks* 的值。*maxlocks* 参数乘以 *maxappls* 参数不能小于 100。

建议：下列公式允许您设置 *maxlocks*，以允许应用程序保留两次锁定的平均数：

```
maxlocks = 2 * 100 / maxappls
```

其中 2 用来完成两次平均，而 100 表示允许的最大百分比值。如果您仅有几个同时运行的应用程序，则可以将下列公式当作第一个公式的替代公式：

```
maxlocks = 2 * 100 /（同时运行的应用程序的平均数目）
```

当设置 *maxlocks* 时的其中一项注意事项是将它与锁定列表（*locklist*）的大小连接使用。在锁定升级发生之前，应用程序保留的锁定数的实际限制是：

```
maxlocks * locklist * 4 096 / (100 * 36)（在 32 位系统上）

maxlocks * locklist * 4 096 / (100 * 56)（在 64 位系统上）
```

其中 4 096 是页中的字节数，100 是 *maxlocks* 允许的最大百分比值，36 是 32 位系统上的每个锁定的字节数，56 是 64 位系统上的每个锁定的字节数。如果您知道您的其中一个应用程序需要 1 000 个锁定，而您不想要锁定升级发生，则您应该为该公式中的 *maxlocks* 和 *locklist* 选择值，以便结果大于 1 000。（对 *maxlocks* 使用 10，且对 *locklist* 使用 100，该公式将得到所需的多于 1 000 个的锁定。）

如果将 *maxlocks* 设置的太低，则当对其它并发应用程序仍然有足够的锁定空间时也会发生锁定升级。如果将 *maxlocks* 设置的太高，则几个应用程序可以消耗大多数锁定空间，而其它应用程序将必须执行锁定升级。在此情况下需要锁定升级会导致较差的并发性。

可使用数据库系统监视器帮助您跟踪和调整此配置参数。

相关参考：

- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 348 页的『maxappls － 最大活动应用程序数』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

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:

- 『chngpgs_thresh – 已更改页数的阈值』
- 第 339 页的『dft_extent_sz – 表空间的缺省扩展数据块大小』
- 第 339 页的『dft_prefetch_sz – 缺省预取大小』
- 第 340 页的『estore_seg_sz – 扩充存储器内存段大小』
- 第 341 页的『num_estore_segs – 扩充存储器内存段数目』
- 第 341 页的『num_iocleaners – 异步页清除程序的数目』
- 第 342 页的『num_ioservers – I/O 服务器数目』
- 第 343 页的『numsegs – SMS 容器的缺省数目』
- 第 343 页的『seqdetect – 顺序检测标志』

chngpgs_thresh – 已更改页数的阈值

配置类型	数据库
参数类型	可配置
缺省值 [范围]	60 [5 – 99]
计量单位	百分比

异步页清除程序在数据库代理进程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。因此，数据库代理进程应该不必等待写出已更改的页，它们也能使用缓冲池中的空间。这提高了数据库应用程序的整体性能。

可以使用此参数指定更改的页的级别（百分比），如果异步页清除程序当前不活动，则将从该级别启动。当启动页清除程序时，页清除程序将构建要写入磁盘的页的列表。一旦页清除程序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，不使用这些页清除程序。

当设置了 DB2_USE_ALTERNATE_PAGE_CLEANING 注册表变量时（即，使用了进行页清理的另一种方法）， *chngpgs_thresh* 参数将不起作用，DB2 将自动确定在缓冲池中要维护多少脏页。

建议: 对于具有繁重的更新事务工作负载的数据库，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干净页。如果数据库有少量很大的表，则大于缺省值的百分比有助于提高性能。

相关参考:

- 第 341 页的『num_iocleaners – 异步页清除程序的数目』
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

dft_extent_sz － 表空间的缺省扩展数据块大小

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围] 32 [2 – 256]

计量单位 页

在创建表空间时，可任意地指定 `EXTENTSIZE n`，其中 `n` 为扩展数据块大小。如果不在 CREATE TABLESPACE 语句上指定扩展数据块大小，则数据库管理器使用由此参数给定的值。

建议：在多数情况中，应在创建表空间时明确指定扩展数据块大小。在选择此参数的值之前，应了解如何显式选择 CREATE TABLESPACE 语句的扩展数据块大小。

相关概念:

- 『扩展数据块大小』（*《管理指南：计划》*）

相关参考:

- 第 339 页的『dft_prefetch_sz － 缺省预取大小』
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

dft_prefetch_sz － 缺省预取大小

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围]

UNIX	Automatic [0 — 32 767]
Windows	Automatic [0 — 32 767]

计量单位 页

当创建表空间时，可选择指定 PREFETCHSIZE *n*（也可以不指定），其中 *n* 表示数据库管理器在执行预取时将读取的页数。如果调用 CREATE TABLESPACE 语句时不指定预取大小，则数据库管理器将使用 *dft_prefetch_sz* 参数的当前值。

如果表空间是使用 AUTOMATIC DFT_PREFETCH_SZ 创建的，则表空间的预取大小将变成 AUTOMATIC，这意味着 DB2 将使用以下等式来自动计算和更新表空间的预取大小:

```
prefetch size = (# containers) * (# physical spindles) * extent size
```

其中，物理主轴数目的缺省值为 1，可以通过 DB2 注册表变量 DB2_PARALLEL_IO 来指定此数目。在下列情况下执行此计算:

- 在启动数据库时
- 第一次使用 AUTOMATIC 预取大小来创建表空间时
- 当通过执行 ALTER TABLESPACE 语句来更改表空间的容器数目时
- 当通过执行 ALTER TABLESPACE 语句来将表空间的预取大小更新为 AUTOMATIC 时

一旦通过调用 ALTER TABLESPACE 语句手工更新了预取大小，就可以打开或关闭预取大小的 AUTOMATIC 状态。

建议：使用系统监视工具，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取大小，则增大此参数的值可能有所帮助。

此参数提供整个数据库的缺省值，它可能并不适合数据库中的所有表空间。例如，值 32 可能适合扩展数据块大小为 32 页的表空间，但不适合于扩展数据块大小为 25 页的表空间。理想情况下，应显式设置每个表空间的预取大小。

为了帮助将用缺省扩展数据块大小（*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 的分数）或 64（32 的整数倍）。如果预取大小是该扩展数据块大小的倍数，则在下列情况下数据库管理器可以并行执行 I/O:

- 正在预取的扩展数据块在不同的物理设备上
- 配置了多个 I/O 服务器（*num_ioservers*）。

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 456 页的『系统环境变量』

estore_seg_sz － 扩充存储器内存段大小

配置类型	数据库
参数类型	可配置
缺省值 [范围]	16 000 [0 – 1 048 575]
计量单位	页（4 KB）

此参数指定数据库中每个扩展内存段中的页数。仅当机器的可寻址实内存超过可寻址虚拟内存的最大数量时，才使用此参数。

建议：此参数仅当扩充存储器可用并且按 *num_estore_segs* 参数所示使用时才有效。当指定每个扩展内存段中要使用的页数时，也应通过复查并修改 *num_estore_segs* 参数来考虑扩展内存段数。

相关参考:

- 第 341 页的『num_estore_segs － 扩充存储器内存段数目』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

num_estore_segs － 扩充存储器内存段数目

配置类型	数据库
参数类型	可配置
缺省值 [范围]	0 [0 – 2 147 483 647]

此参数指定可供数据库使用的扩充存储器内存段的数目。

缺省值为无扩充存储器内存段。

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存储器内存段。当指定段数时，也应通过复查并修改 *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展内存。

相关参考:

- 第 340 页的『estore_seg_sz － 扩充存储器内存段大小』
- 『ALTER BUFFERPOOL statement』(*SQL Reference, Volume 2*)
- 『CREATE BUFFERPOOL statement』(*SQL Reference, Volume 2*)
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

num_iocleaners － 异步页清除程序的数目

配置类型	数据库
参数类型	可配置
缺省值 [范围]	1 [0 – 255]
计量单位	计数器

此参数允许您指定数据库的异步页清除程序数。在数据库代理进程需要缓冲池的空间之前，这些页清除程序将更改的页从缓冲池写入磁盘。因此，数据库代理进程应该不必等待写出已更改的页，它们也能使用缓冲池中的空间。这提高了数据库应用程序的整体性能。

如果您将该参数设置为零(0)，则未启动页清除程序，因此，数据库代理进程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。如果未配置页清除程序，则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，则增加清除程序的数目将会提高运行速度。增加页清除程序也将减少从软故障(例如，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最新的。

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载导致创建许多 TEMP 表（您可以使用说明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。
- 工作负载

  具有较高更新事务比率的环境可能需要配置更多页清除程序。
- 缓冲池大小

  含有大缓冲池的环境可能也需要配置更多的页清除程序。

可以使用数据库系统监视器来帮助调整此配置参数，该数据库系统监视器使用事件监视器收集的有关缓冲池写入活动的信息:

- 如果同时满足下面两个条件，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，则应增大该参数:
  - *pool_data_writes* 比 *pool_async_data_writes* 大很多
  - *pool_index_writes* 比 *pool_async_index_writes* 大很多

相关参考:

- 第 338 页的『chngpgs_thresh － 已更改页数的阈值』
- 『pool_data_writes - Buffer Pool Data Writes monitor element』（*System Monitor Guide and Reference*）
- 『pool_index_writes - Buffer Pool Index Writes monitor element』（*System Monitor Guide and Reference*）
- 『pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element』（*System Monitor Guide and Reference*）
- 『pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

num_ioservers － I/O 服务器数目

配置类型	数据库
参数类型	可配置
缺省值 [范围]	3 [1 – 255]
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	当应用程序与数据库断开连接时

用 I/O 服务器代表数据库代理进程从而通过实用程序（例如，备份和复原）执行预取 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。非预取 I/O 直接由数据库代理进程调度，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开销，而且任何未使用的 I/O 服务器都将保持空闲。

相关参考:

- 第 339 页的『dft_prefetch_sz － 缺省预取大小』
- 第 343 页的『seqdetect － 顺序检测标志』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

numsegs － SMS 容器的缺省数目

配置类型	数据库
参数类型	参考
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。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

seqdetect － 顺序检测标志

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Yes [Yes; No]

数据库管理器可以监视 I/O，如果正在进行顺序页读取，则数据库管理器可以激活 I/O 预取。这种类型的顺序预取称为*顺序检测*。可以使用 *seqdetect* 配置参数来控制数据库管理器是否应该执行顺序检测。

如果此参数设置为 No，则仅当数据库管理器知道预取将有用时才会发生预取。例如，表排序、表扫描或列表预取。

建议: 在大多数情况下，应使用此参数的缺省值。仅当其它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。

相关参考:

- 第 339 页的『dft_prefetch_sz － 缺省预取大小』

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

代理进程

下列参数会影响可同时运行并达到最佳性能的应用程序的数目:

- 『agentpri – 代理进程的优先级』
- 第 345 页的『avg_appls – 活动应用程序的平均数目』
- 第 346 页的『max_connections – 最大客户机连接数』
- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 347 页的『maxagents – 最大代理进程数』
- 第 348 页的『maxappls – 最大活动应用程序数』
- 第 349 页的『maxcagents – 最大并发代理进程数』
- 第 350 页的『maxfilop – 每个应用程序打开的最大数据库文件数』
- 第 351 页的『maxtotfilop – 打开的最大总文件数』
- 第 352 页的『num_initagents – 池中的代理进程的初始数目』
- 第 352 页的『num_poolagents – 代理进程池大小』

agentpri – 代理进程的优先级

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]

AIX -1 [41 - 125]

其它 UNIX
-1 [41 - 128]

Windows
-1 [0 - 6]

此参数通过操作系统调度程序来控制给予所有代理进程和其它数据库管理器实例进程和线程的优先级。在分区数据库环境中，这也包括协调和子代理进程、并行系统控制器以及 FCM 守护进程。此优先级确定如何将 CPU 时间给予与在该机器上运行的其它进程和线程有关的 DB2 进程、代理进程和线程。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器。当将该参数设置为一个不为 -1 的值时，数据库管理器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器进程和线程在机器上执行时所用的优先级。

可使用此参数来增大数据库管理器吞吐量。用于设置此参数的值取决于数据库管理器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，数据库管理器在该参数的值设置为 UNIX 静态优先级的情况下创建其代理进程。这在基于 UNIX 的环境中很重要，因为低数值代表数据库管理器的高优先级，但其它进程（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它预期的活动相平衡。

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器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，可以使用基准程序技术来确定此参数的最佳设置。提高数据库管理器的优先级时应仔细，因为可能会极大地降低其它用户进程的性能，特别是当 CPU 利用率很高的时候更是如此。提高数据库管理器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进程的优先级。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

avg_appls － 活动应用程序的平均数目

配置类型	数据库
参数类型	可联机配置
传播类	语句边界
缺省值 [范围]	1 [1 – maxappls]
计量单位	计数器

SQL 优化器使用此参数来帮助估计在运行时内将有多少缓冲池可用于选择的存取方案。

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复杂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。如果对此数求值有困难，可以乘以下列数:

- 对数据库运行的所有应用程序的平均数。数据库系统监视器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。来自数据库系统监视器的信息包括 OLTP 和非 OLTP 应用程序。
- 对于复杂查询应用程序的百分比的估计。

正如调整影响优化器的其它配置参数一样，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绑定应用程序（使用 REBIND PACKAGE 命令）。

相关参考:

- 第 348 页的『maxappls － 最大活动应用程序数』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

max_connections － 最大客户机连接数

配置类型	数据库管理器
参数类型	可配置
缺省值 [范围]	-1 (*max_coordagents*) [-1; *max_coordagents* — 64 000]

当“集中器”关闭时，此参数指示每个分区允许的客户机连接的最大数目。当 *max_connections* 等于 *max_coordagents* 时，“集中器”关闭。当 *max_connections* 大于 *max_coordagents* 时，“集中器”打开。

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进程。代理进程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，每个代理进程使用它自己的专用内存来运行，并与其它代理进程共享数据库管理器和数据库全局资源，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*。

在先前版本的 DB2 中，此参数称为 *max_logicagents*。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

max_coordagents － 最大协调代理进程数

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置

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*。

当集中器关闭时，即，当 *max_connections* 等于 *max_coordagents* 时，此参数确定在分区或非分区数据库环境中的服务器上可同时存在的协调代理进程的最大数目。

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进程。需要连接实例的请求包括 CREATE DATABASE、DROP DATABASE 和“数据库系统监视器”命令。

当“集中器”打开时，即，当 *max_connections* 大于 *max_coordagents* 时，可能有比协调代理进程多的连接来为它们提供服务。仅当有协调代理进程在为应用程序提供服务时，应用程序才处于活动状态。否则，应用程序处于不活动状态。来自活动应用程序的请求将由数据库协调代理进程（以及 SMP 或 MPP 配置中的子代理进程）来提供服务。来自不活动的应用程序的请求将会进行排队，直到指定数据库协调代理进程来为该应用程序提供服务，此时，应用程序变成活动的。因此，此参数可用来控制系统上的装入。

相关参考:

- 第 352 页的『num_initagents – 池中的代理进程的初始数目』
- 第 352 页的『num_poolagents – 代理进程池大小』
- 第 413 页的『intra_parallel – 启用分区内并行性』
- 第 347 页的『maxagents – 最大代理进程数』

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

maxagents – 最大代理进程数

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	200 [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为 400 [1 – 64 000]
计量单位	计数器

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器代理进程（无论是协调代理进程还是子代理进程）的最大数目。如果您想限制协调代理进程数，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器使用的内存总量，因为每个附加代理进程都需要附加内存。

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。

每个附加代理进程都需要一些在数据库管理器启动时分配的资源开销。

相关参考:

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 352 页的『num_poolagents – 代理进程池大小』
- 第 349 页的『maxcagents – 最大并发代理进程数』
- 第 353 页的『fenced_pool – 受防护的进程的最大数目』
- 第 348 页的『maxappls – 最大活动应用程序数』
- 第 320 页的『min_priv_mem – 最小已落实专用内存』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

maxappls – 最大活动应用程序数

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Automatic [Automatic; 1 – 60 000]
计量单位	计数器

此参数指定可与一个数据库连接（本地和远程）的并发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，所以允许大量并发应用程序可能将使用更多内存。

将 *maxappls* 设置为 *automatic* 将允许任何数目的已连接应用程序。DB2 将动态分配它需要用来支持新应用程序的资源。

如果不想将此参数设置为 *automatic*，则此参数的值必须大于或等于已连接应用程序的总数，再加上可能在完成两阶段落实或回滚进程中同时运行的这些相同应用程序的数目。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。

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 Data Links 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 Data Links Manager 的活动应用程序数的峰值)
```

Data Links 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同时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，不管该服务器是否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它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: 增大此参数的值而不降低 *maxlocks* 参数或增大 *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*）以及可用的代理进程（*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，协调代理进程）。

相关任务:

- 『手工解析不确定事务』（*《管理指南: 计划》*）

相关参考:

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 347 页的『maxagents – 最大代理进程数』
- 第 310 页的『locklist – 锁定列表的最大存储器』
- 第 336 页的『maxlocks – 升级前锁定列表的最大百分比』
- 第 345 页的『avg_appls – 活动应用程序的平均数目』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

maxcagents – 最大并发代理进程数

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] -1 (*max_coordagents*) [-1; 1 – *max_coordagents*]

计量单位 计数器

可同时执行一个数据库管理器事务的数据库管理器代理进程的最大数目。此参数用来控制并行应用程序活动高峰期内系统上的负载。例如，可以让一个系统需要大量连接但是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页面调度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器可同时处理的数据库管理器代理进程的数目，从而限制在处理高峰期内系统资源的使用。

值为 −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。在应用程序的高并发性导致问题的情况下，可以使用基准程序测试来调整此参数以优化数据库的性能。

相关参考：

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 347 页的『maxagents – 最大代理进程数』
- 第 348 页的『maxappls – 最大活动应用程序数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

maxfilop – 每个应用程序打开的最大数据库文件数

配置类型 数据库

参数类型 可联机配置

传播类 事务边界

缺省值 [范围]

UNIX	64 [2 – 1950]
Windows	64 [2 – 32 768]

计量单位 计数器

此参数指定对于每个数据库代理进程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，则关闭此代理进程正在使用的一些文件。如果 *maxfilop* 太小，为了不超过此限制，打开和关闭文件的开销将变得极大，并且可能会降低性能。

在数据库管理器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。

也可以使用此参数来确保数据库管理器使用的文件句柄总数不超过操作系统限制，其方法是将每个代理进程的句柄数限制为一个特定的数。实际的限制数将根据同时运行的代理进程数的不同而有所变化。

相关参考:

- 第 348 页的『maxappls － 最大活动应用程序数』
- 第 351 页的『maxtotfilop － 打开的最大总文件数』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

maxtotfilop － 打开的最大总文件数

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
16 000 [100 – 32 768]

计量单位
计数器

此参数定义所有代理进程和其它在单个数据库管理器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器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。

相关参考:

- 第 350 页的『maxfilop － 每个应用程序打开的最大数据库文件数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

num_initagents – 池中的代理进程的初始数目

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] 0 [0 — *num_poolagents*]

此参数确定在 DB2START 时在代理进程池中创建的初始空闲代理进程数。

相关参考:

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 352 页的『num_poolagents – 代理进程池大小』
- 第 347 页的『maxagents – 最大代理进程数』
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

num_poolagents – 代理进程池大小

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] -1 (*maxagents* / 2) [-1; 0 — *maxagents*]

当“集中器”关闭时，即，当 *max_connections* 等于 *max_coordagents* 时，此参数确定空闲代理进程池的最大大小。空闲代理进程可用作并行子代理进程或协调代理进程。如果创建的代理进程超过此参数值指示的数目，则当这些代理进程执行完其当前请求时，将被终止，而不是返回至池中。

当“集中器”打开时，即，当 *max_connections* 大于 *max_coordagents* 时，无论此参数的值是什么，代理进程始终都将返回到池中。根据系统装入和代理进程在池中保持空闲的时间，代理进程可能会根据需要终止它们以将空闲池的大小降低至配置的参数值。

除了打开“集中器”的情况之外，如果此参数的值为 0，则按需要创建代理进程，并且当这些代理进程执行完它们的当前请求时将立即终止。

建议：如果运行几乎没有并发应用程序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进程的代理进程池。

如果运行一个同时连接很多应用程序的事务处理环境，则增大 *num_poolagents* 的值以避免频繁创建和终止代理进程所产生的成本。

相关参考：

- 第 352 页的『num_initagents － 池中的代理进程的初始数目』
- 第 346 页的『max_coordagents － 最大协调代理进程数』
- 第 414 页的『max_querydegree － 最大查询并行度』
- 第 347 页的『maxagents － 最大代理进程数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

存储过程和用户定义的函数

下列参数会影响受防护的存储过程和用户定义的函数性能：

- 『fenced_pool － 受防护的进程的最大数目』
- 第 354 页的『keepfenced － 保持受防护的进程』
- 第 355 页的『num_initfenced － 受防护的进程的初始数目』

fenced_pool － 受防护的进程的最大数目

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	-1 (*max_coordagents*)
计量单位	计数器

对于线程化 db2fmp 进程（为线程安全存储过程和 UDF 提供服务的进程），此参数表示每个 db2fmp 进程中高速缓存的线程数目。对于非线程 db2fmp 进程，此参数表示高速缓存的线程数目。

建议：如果您的环境使用受防护的存储过程或用户定义的函数，则此参数可用来确保提供有适当数目的 db2fmp 进程，以处理在此实例上运行的最大数目的并发存储过程和 UDF，这将确保不需要创建新的设防方式进程作为存储过程和 UDF 执行的一部分。

如果该参数设置为 −1，则高速缓存的 db2fmp 进程的最大数目将与在 *max_coordagents* 参数中设置的值相同。

如果由于给定 db2fmp 进程的系统资源量不合适且影响数据库管理器的性能而导致缺省值对您的环境不适用，则以下内容在作为提供调整该参数的起点方面可能有用:

```
fenced_pool = 允许其中 # 个应用程序同时进行存储过程和 UDF 调用
```

如果 *keepfenced* 设置为 *yes*，则在高速缓存池中创建的每个 db2fmp 进程将继续存在并使用系统资源，即使在受防护的例程调用已处理并且返回至代理进程之后。

如果 *keepfenced* 设置为 *no*，则非线程 db2fmp 进程将在完成执行后终止，且没有任何高速缓存池。多线程 db2fmp 进程将继续存在，但是不会在这些进程中合用线程。这意味着即使将 *keepfenced* 设置为 *no*，您也可以在系统上有一个线程化 C db2fmp 进程和一个线程化 Java db2fmp 进程。

在先前版本的 DB2 中，此参数称为 *maxdari*。

相关参考:

- 第 346 页的『max_coordagents – 最大协调代理进程数』
- 第 347 页的『maxagents – 最大代理进程数』
- 第 354 页的『keepfenced – 保持受防护的进程』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 第 355 页的『num_initfenced – 受防护的进程的初始数目』

keepfenced – 保持受防护的进程

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可配置

缺省值 [范围]　　Yes [Yes; No]

此参数指示在设防方式例程调用完成后，是否保留设防方式进程。将设防方式进程创建为独立的系统实体以便将用户编写的设防方式代码与数据库管理器代理进程分开。此参数仅可应用于数据库服务器。

如果将 *keepfenced* 设置为 *no*，且正执行的例程不是线程安全的，则对于每个设防方式调用，新的设防方式都会被创建并被破坏。如果将 *keepfenced* 设置为 *no*，且正执行的例程是线程安全的，则设防方式进程将会仍然存在，但是为调用创建的线程将会终止。如果将 *keepfenced* 设置为 *yes*，则对于后续设防方式调用，将重复使用设防方式进程或线程。当数据库管理器停止时，将终止所有未完成的设防方式进程和线程。

将此参数设置为 *yes* 将导致每个激活的设防方式进程的数据库管理器消耗更多系统资源，最多为在 *fenced_pool* 参数中包含的值。仅当没有提供现有设防方式进程来处理后续受防护的例程调用时，才会创建新的进程。如果将 *fenced_pool* 设置为 0，则将忽略此参数。

建议： 在设防方式请求数目相对于不设防方式请求数目很大的环境中，且系统资源不受约束，可将此参数设置为 *yes*。这将通过避免初始设防方式进程创建开销来提高设防方式进程性能，因为将使用现有设防方式进程来处理调用。尤其是对于 Java 例程，这将节省启动“Java 虚拟机”（JVM）的开销，这会使性能有很显著的提高。

例如，在 OLTP 信贷银行事务应用程序中，用来执行每个事务的代码可以在设防方式进程中执行的存储过程中执行。在此应用程序中，主要的工作负载在设防方式进程之外执行。如果将此参数设置为 *no*，每个事务都会导致创建新的设防方式进程的开销，这将使性能显著降低。但是，如果此参数设置为 *yes*，则每个事务都将尝试使用现有设防方式进程，这将避免此开销。

在先前版本的 DB2 中，此参数称为 *keepdari*。

相关参考：

- 第 353 页的『fenced_pool － 受防护的进程的最大数目』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

num_initfenced － 受防护的进程的初始数目

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
0 [0 — *max_connections* + (*maxagents* - *max_coordagents*)]

此参数指示在 DB2START 时间 db2fmp 池中创建的非线程的空闲 db2fmp 进程的初始数目。设置此参数将减少运行非线程安全 C 和 Cobol 例程的初始启动时间。如果未指定 *keepfenced*，则将忽略此参数。

为您的系统将 *fenced_pool* 设置为适当的大小比在 DB2START 时间启动一些 db2fmp 进程重要得多。

在先前版本的 DB2 中，此参数称为 *num_initdaris*。

相关参考：

- 第 354 页的『keepfenced － 保持受防护的进程』
- 第 353 页的『fenced_pool － 受防护的进程的最大数目』
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

日志记录和恢复

恢复您的环境对防止重要数据的丢失非常重要。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个类别:

- 『数据库日志文件』
- 第 364 页的『数据库日志活动』
- 第 373 页的『恢复』
- 第 383 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日志记录的文件的数量、大小和状态的信息:

- 『logfilsiz － 日志文件大小』
- 第 357 页的『loghead － 首个活动日志文件』
- 第 358 页的『logpath － 日志文件的位置』
- 第 358 页的『logprimary － 主日志文件数』
- 第 359 页的『logsecond － 辅助日志文件数』
- 第 360 页的『max_log － 每个事务的最大日志』
- 第 361 页的『mirrorlogpath － 镜像日志路径』
- 第 362 页的『newlogpath － 更改数据库日志路径』
- 第 363 页的『num_log_span － 跨越的日志数』
- 第 363 页的『overflowlogpath － 溢出日志路径』

logfilsiz － 日志文件大小

配置类型	数据库	
参数类型	可配置	
缺省值 [范围]		
	UNIX	1000 [4 — 262 144]
	Windows	1000 [4 — 262 144]
计量单位	页（4 KB）	

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日志记录的类型:

- 循环日志记录

  当记录在主日志文件中的更改已落实后，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了大量对数据库的更改但未落实这些更改，主日志文件可能会很快变满。如果所有主日志文件变满，则数据库管理器将分配辅助日志文件来保存新的日志记录。

- 日志保留日志记录

  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: 日志文件大小的上限与日志文件数目的上限一起（*logprimary* + *logsecond*），给定活动日志空间的上限 256 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连接时，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。因此，如果了解您的并发应用程序的日志记录需求，可能可以确定一个将不会分配过量浪费空间的日志文件大小。

相关参考:

- 第 358 页的『logprimary － 主日志文件数』
- 第 359 页的『logsecond － 辅助日志文件数』
- 第 371 页的『softmax － 恢复范围和软检查点时间间隔』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

loghead － 首个活动日志文件

配置类型	数据库
参数类型	参考

此参数包含当前活动的日志文件的名称。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

logpath － 日志文件的位置

配置类型 数据库

参数类型 参考

此参数包含用于进行日志记录的当前路径。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器设置的。

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。

相关参考:

- 第 362 页的『newlogpath － 更改数据库日志路径』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

logprimary － 主日志文件数

配置类型 数据库

参数类型 可配置

缺省值 [范围] 3 [2 – 256]

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）
- 在所有用户断开连接之后，下一次数据库连接期间，会增大此参数（*logprimary*）的值
- 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，则在所有用户都已断开连接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。

主日志文件建立分配给恢复日志文件的固定存储器数量。此参数允许您指定要预分配的主日志文件数。

在循环日志记录下，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）。如果已落实或回滚日志中具有日志记录的所有工作单元，则认为该日志是可用的。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。当数据库管理器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与下列内容一致:

- 如果 *logsecond* 的值为 -1，则 *logprimary* 小于等于 256。

- 如果 *logsecond* 的值不是 -1，则（*logprimary* + *logsecond*） 小于等于 256。

建议: 为此参数选择的值取决于许多因素，包括正在使用的日志记录类型、日志文件的大小和处理环境的类型（例如，事务的长度和落实的频率）。

增大此值将增大日志的磁盘需求，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。

如果您发现经常分配辅助日志文件，则可能可以通过增大日志文件大小（*logfilsiz*）或增大主日志文件的数目来提高系统性能。

对于不经常存取的数据库，为了节省磁盘存储空间，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开销。

可以使用数据库系统监视器来帮助您确定主日志文件的大小。在一段时间内对下列监视器值的观察将有助于更好的调整决定，因为平均值更能代表当前需求。
- *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）*

相关参考:
- 第 356 页的『logfilsiz － 日志文件大小』
- 第 359 页的『logsecond － 辅助日志文件数』
- 第 368 页的『logretain － 启用日志保留』
- 第 372 页的『userexit － 启用用户出口』
- 『sec_log_used_top - Maximum Secondary Log Space Used monitor element』（*System Monitor Guide and Reference*）
- 『tot_log_used_top - Maximum Total Log Space Used monitor element』（*System Monitor Guide and Reference*）
- 『sec_logs_allocated - Secondary Logs Allocated Currently monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

logsecond － 辅助日志文件数

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	2 [-1; 0 – 254]
计量单位	计数器
分配时	当 *logprimary* 不够时需要（查看以下详细信息）
释放时	数据库管理器确定将不再需要它们的结束时间。

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，则将返回错误代码至该应用程序，并将关闭该数据库。

如果将 *logsecond* 设置为 -1，则数据库将配置为具有无限活动日志空间。对在此数据库上运行的未完成事务的大小和数目没有限制。如果将 *logsecond* 设置为 -1，则仍使用 *logprimary* 和 *logfilsiz* 配置参数来指定 DB2 应在活动日志路径中保留多少个日志文件。如果 DB2 需要从日志文件读取日志数据，但该文件不在活动日志文件路径中，则 DB2 将调用用户出口程序来将该日志文件从压缩文档检索至活动日志路径。（如果配置了溢出日志路径，则 DB2 会将这些文件检索至该路径。）一旦检索到日志文件，DB2 将把此文件高速缓存在活动日志路径中，以便他人从同一文件读取日志数据时可提高速度。DB2 将根据需要管理这些日志文件的检索、高速缓存和除去操作。

如果您的日志路径是裸设备，则必须配置 *overflowlogpath* 配置参数以便将 *logsecond* 设置为 -1。

通过将 *logsecond* 设置为 -1，将对工作单元的大小或并发工作单元的数目没有限制。但是，由于需要从压缩文档检索日志文件，回滚（在保存点级别和在工作单元级别）将可能变得很慢。同样的原因，崩溃恢复也可能变得很慢。DB2 会将一条消息写入到管理通知日志，以警告您活动的工作单元的当前集已超出主日志文件。此指示回滚或崩溃恢复可能会非常慢。

要将 *logsecond* 设置为 -1，必须将 *userexit* 配置参数设置为 *yes*。

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会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。

相关参考:

- 第 356 页的『logfilsiz － 日志文件大小』
- 第 358 页的『logprimary － 主日志文件数』
- 第 368 页的『logretain － 启用日志保留』
- 第 372 页的『userexit － 启用用户出口』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 363 页的『overflowlogpath － 溢出日志路径』

max_log － 每个事务的最大日志

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	0 [0 — 100]
计量单位	百分比

如果该值不为 0，则此参数指示一个事务可以消耗的活动日志空间的百分比。

如果该值设置为 0，则对单个事务可以消耗的空间（按总的活动日志空间的百分比计算）没有限制。这是版本 8 之前的事务的行为。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 363 页的『num_log_span － 跨越的日志数』

mirrorlogpath － 镜像日志路径

配置类型	数据库
参数类型	可配置
缺省值 [范围]	Null（任何有效的路径或设备）

此参数允许您为镜像日志路径指定最多 242 个字节的字符串。该字符串必须指向路径名，并且它必须为全限定路径名，而不是相对路径名。

注: 在分区数据库环境中，会将节点号自动追加至路径。这样做是为了维护多个逻辑节点配置中路径的唯一性。

如果配置了 *mirrorlogpath*，则 DB2 将在日志路径和镜像日志路径中创建活动日志文件。会将所有日志数据写入这两个路径。镜像日志路径有一组相同的活动日志文件，因此如果存在破坏其中一条路径上的活动日志文件的磁盘错误或人为错误，数据库仍能起作用。

如果更改镜像日志路径，则旧的镜像日志路径中可能有日志文件。可能还没有归档这些日志文件，所以您可能需要手工归档这些日志文件。同样地，如果在此数据库上运行复制，复制可能仍需要更改日志路径前的日志文件。如果在将“启用用户出口”（*userexit*）数据库配置参数设置为“Yes”的情况下配置数据库，并且如果 DB2 自动对所有日志文件进行归档或由您手工归档，则 DB2 将能够检索日志文件以完成复制过程。否则的话，您可以将文件从旧的镜像日志路径复制到新的镜像日志路径中。

如果 *logpath* 或 *newlogpath* 将裸设备指定为存储日志文件的位置，则就象 *mirrorlogpath* 指示的那样，不允许镜像日志记录。如果 *logpath* 或 *newlogpath* 将文件路径指定为存储日志文件的位置，则允许镜像日志记录，并且 *mirrorlogpath* 还必须指定文件路径。

建议: 就象日志文件一样，镜像日志文件应在不是高 I/O 的物理磁盘上。

强烈建议此路径在与主日志路径不同的设备上。

可以使用数据库系统监视器来跟踪与数据库日志记录相关的 I/O 数。

下列数据元素返回与数据库日志记录相关的 I/O 活动数。您可以使用操作系统监视工具来收集关于其它磁盘 I/O 活动的信息，然后比较两种类型的 I/O 活动。

- *log_reads*（读取的日志页数）
- *log_writes*（写入的日志页数）。

相关参考:

- 第 358 页的『logpath － 日志文件的位置』

- 第 362 页的『newlogpath － 更改数据库日志路径』
- 『log_reads - Number of Log Pages Read monitor element』（*System Monitor Guide and Reference*）
- 『log_writes - Number of Log Pages Written monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 363 页的『overflowlogpath － 溢出日志路径』

newlogpath － 更改数据库日志路径

配置类型	数据库
参数类型	可配置
缺省值 [范围]	Null（任何有效的路径或设备）

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。该字符串可以指向路径名或裸设备。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，会将节点号自动追加至路径。这样做是为了维护多个逻辑节点配置中路径的唯一性。

如果想要使用复制，并且您的日志路径是裸设备，则必须配置 *overflowlogpath* 配置参数。

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 Service Pack 3 的 Windows NT V4.0 或更高版本，才能够将日志写入设备。
- 在基于 UNIX 的平台上，为 `/dev/rdblog8`

注: 只能在 AIX、Windows 2000、Windows NT、Solaris Operating Environment、HP-UX 和 Linux 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:

- 数据库处于一致状态，如 *database_consistent* 参数所指示。
- 所有用户与数据库断开连接

当建立与该数据库的第一个新连接时，数据库管理器将这些日志移至由 *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。您可能需要手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。如果在将“启用用户出口”（*userexit*）数据库配置参数设置为 Yes 的情况下配置数据库，并且如果 DB2 自动对所有日志文件进行归档或由您手工归档，则 DB2 将能够检索日志文件以完成复制过程。否则，可以将这些文件从旧日志路径复制至新日志路径。

如果 *logpath* 或 *newlogpath* 将裸设备指定为存储日志文件的位置，则就象 *mirrorlogpath* 指示的那样，不允许镜像日志记录。如果 *logpath* 或 *newlogpath* 将文件路径指定为存储日志文件的位置，则允许镜像日志记录，并且 *mirrorlogpath* 还必须指定文件路径。

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。这将提高日志记录活动的效率并使其开销（例如，等待 I/O）最小。

可以使用数据库系统监视器来跟踪与数据库日志记录相关的 I/O 数。

监视元素 *log_reads*（读取的日志页的数目）和 *log_writes*（写入的日志页的数目）返回与数据库日志记录相关的 I/O 活动数。您可以使用操作系统监视工具来收集关于其它磁盘 I/O 活动的信息，然后比较两种类型的 I/O 活动。

相关参考:

- 第 358 页的『logpath － 日志文件的位置』
- 第 394 页的『database_consistent － 数据库是一致的』
- 『log_reads - Number of Log Pages Read monitor element』（*System Monitor Guide and Reference*）
- 『log_writes - Number of Log Pages Written monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

num_log_span － 跨越的日志数

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	0 [0 — 65 535]
计量单位	计数器

如果该值不为 0，则此参数指示允许一个活动事务跨越的活动日志文件数。

如果该值设置为 0，则对单个事务可以跨越的日志文件数没有限制。这是版本 8 之前的事务的行为。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 360 页的『max_log － 每个事务的最大日志』

overflowlogpath － 溢出日志路径

配置类型	数据库

参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（任何有效路径）

此参数可以用于几种函数，这取决于日志记录需求。

- 此参数允许您指定位置，以便 DB2 查找前滚操作所需的日志文件。它与 ROLLFORWARD 命令上的 OVERFLOW LOG PATH 选项相似。可以只设置此配置参数一次，而不是在每个 ROLLFORWARD 命令上都指定 OVERFLOW LOG PATH。但是，如果两种方法都使用，则对于该特定前滚操作，OVERFLOW LOG PATH 选项将覆盖 *overflowlogpath* 配置参数。
- 如果将 *logsecond* 设置为 -1，则 *overflowlogpath* 允许您为 DB2 指定目录以存储从压缩文档检索到的活动日志文件。（如果活动日志文件不再存在于活动日志路径中，则必须检索它们以用于回滚操作）。没有 overflowlogpath，DB2 将把日志文件检索到活动日志路径中。使用 *overflowlogpath* 允许您为 DB2 提供附加资源以存储检索的日志文件。好处包括将 I/O 成本分布到不同的磁盘上，以及允许将更多的日志文件存储在活动日志路径中。
- 例如，如果您需要使用 db2ReadLog API（在 DB2 V8 之前，db2ReadLog 称为 sqlurlog）来进行复制，则 *overflowlogpath* 允许您为 DB2 指定位置来搜索此 API 需要的日志文件。如果未找到日志文件（在活动日志路径或溢出日志路径中）且数据库配置为启用 *userexit*，则 DB2 将检索日志文件。*overflowlogpath* 也允许您为 DB2 指定目录以存储检索的日志文件。好处包括降低活动日志路径上的 I/O 成本以及允许将更多的日志文件存储在活动日志路径中。
- 如果为活动日志路径配置了裸设备，则如果想要将 *logsecond* 设置为 -1，或如果想要使用 db2ReadLog API，就必须配置 *overflowlogpath*。

要设置 *overflowlogpath*，指定最多为 242 个字节的字符串。该字符串必须指向路径名，并且它必须为全限定路径名，而不是相对路径名。路径名必须为目录，不能为裸设备。

注: 在分区数据库环境中，会将节点号自动追加至路径。这样做是为了维护多个逻辑节点配置中路径的唯一性。

相关参考:

- 第 359 页的『logsecond － 辅助日志文件数』
- 『db2ReadLog - Asynchronous Read Log』（*Administrative API Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『ROLLFORWARD DATABASE Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

数据库日志活动

下列参数会影响数据库日志记录的类型和性能:

- 第 365 页的『archretrydelay － 出错时的归档重试延迟』
- 第 365 页的『blk_log_dsk_ful － 磁盘已满时阻止进行日志记录』
- 第 366 页的『failarchpath － 故障转移日志归档路径』

- 第 366 页的『logarchmeth1 － 主日志归档方法』
- 第 367 页的『logarchmeth2 － 辅助日志归档方法』
- 第 367 页的『logarchopt1 － 主日志归档选项』
- 第 368 页的『logarchopt2 － 辅助日志归档选项』
- 第 368 页的『logindexbuild － 创建的日志索引页』
- 第 368 页的『logretain － 启用日志保留』
- 第 369 页的『mincommit － 对组的落实次数』
- 第 370 页的『numarchretry － 出错时的重试次数』
- 第 371 页的『softmax － 恢复范围和软检查点时间间隔』
- 第 372 页的『userexit － 启用用户出口』
- 第 373 页的『vendoropt － 供应商选项』

archretrydelay － 出错时的归档重试延迟

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] 20 [0 – 65 535]

此参数指定在对日志文件尝试进行归档失败之后再次尝试归档之前要等待的秒数。仅当 *numarchretry* 数据库配置参数的值至少为 1 时，后续的重试才会生效。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

blk_log_dsk_ful － 磁盘已满时阻止进行日志记录

配置类型 数据库

参数类型 可联机配置

传播类 立即

缺省值 [范围] No [Yes; No]

可以设置此配置参数以防止当 DB2 不能在活动日志路径中创建新的日志文件时发生磁盘已满错误。而且，DB2 将尝试每隔五分钟就创建一次日志文件，直至成功。在每次尝试之后，DB2 会将消息写入“管理通知”日志。监视“管理通知”日志是确认应用程序是否由于日志磁盘已满而挂起的唯一方法。在成功创建日志文件之前，尝试更新表数据的任何用户应用程序都无法落实事务。只读查询可能不会直接受影响；但是，如果查询需要存取被更新请求锁定的数据或者由更新应用程序在缓冲池中修正的数据页时，只读查询也将象挂起一样。

将 *blk_log_dsk_ful* 设置为 *yes* 会使 DB2 遇到日志磁盘已满错误时应用程序挂起，从而使您可解决该错误并允许事务完成。可以通过将旧日志文件移至另一个文件系统或扩充文件系统来解决磁盘已满情况，以便挂起的应用程序可以完成。

如果 *blk_log_dsk_ful* 设置为 *no*，则接收日志磁盘已满错误的事务将会失败并回滚。在某些情况下，如果事务导致日志磁盘已满错误，数据库将关闭。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

failarchpath － 故障转移日志归档路径

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] Null [ ]

此参数指定一条路径，如果由于影响归档目标的介质问题，不能将日志文件归档至主要归档目标或辅助归档目标（如果设置了此项的话），则 DB2 将尝试按此路径对日志文件进行归档。指定的此路径必须引用一个磁盘。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

logarchmeth1 － 主日志归档方法

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] Off [ ]

此参数指定已归档日志的主要目标的介质类型。

相关概念:

- 第 547 页的附录 E,『使用 db2adutl 命令以及 logarchopt1 和 vendoropt 数据库配置参数来进行跨节点恢复』

相关参考:
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

logarchmeth2 – 辅助日志归档方法

配置类型 数据库

适用于
- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] Off [ ]

此参数指定已归档日志的辅助目标的介质类型。如果指定了此路径，则日志文件将被同时归档至此目标和由 *logarchmeth1* 数据库配置参数指定的目标。

相关参考:
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

logarchopt1 – 主日志归档选项

配置类型 数据库

适用于
- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] Null [ ]

此参数为已归档日志的主要目标指定选项字段(如果需要的话)。

相关参考:
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

logarchopt2 – 辅助日志归档选项

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

缺省值 [范围]
Null [ ]

此参数为已归档日志的辅助目标指定选项字段。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

logindexbuild – 创建的日志索引页

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

缺省值 [范围]
Off [On; Off]

此参数指定是否要记录索引创建、重新创建或重组操作，以便可以在 DB2 前滚操作或高可用性灾难恢复(HADR)日志重放过程中重构索引。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

logretain – 启用日志保留

配置类型
数据库

参数类型
可配置

缺省值 [范围]
No [Recovery; No]

这些值如下所示:

- No，表示不保留这些日志。

• 恢复，指示保留这些日志，并可用于正向恢复。

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。这称为日志保留日志记录。

在 *logretain* 设置为 Recovery 或 *userexit* 设置为 Yes（或两者都这样设置）之后，则必须对该数据库进行完全备份。此状态由 *backup_pending* 标志参数指示。

如果将 *logretain* 设置为 No 并将 *userexit* 设置为 No，则因为未保留日志而使得前滚恢复对于数据库不可用。在这种情况下，数据库管理器将删除 *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，并还原至循环日志记录。

相关参考:

- 第 394 页的『log_retain_status － 日志保留状态指示符』
- 第 372 页的『userexit － 启用用户出口』
- 第 394 页的『backup_pending － 备份暂挂指示符』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

mincommit － 对组的落实次数

配置类型	数据库
参数类型	可联机配置
传播类	立即
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。此延迟有助于缩小与写入日志记录相关联的数据库管理器开销。这样，若您有多个应用程序对数据库运行，且在很短的时间范围内这些应用程序请求了许多落实，则可提高性能。

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。当正在执行落实组合，将保留应用程序落实请求直至经过一秒的时间或落实请求的数目等于此参数的值。

| 只应该稍微增大此参数，例如，增大一（1）。还应该使用多用户测试来验证增大此参
| 数的值是否能获得期望的结果。

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连接。

建议: 如果多个读／写应用程序在一般情况下请求并发数据库落实，此参数将从缺省值增大。这将导致更有效的日志记录文件 I/O，因为日志记录文件 I/O 进行的频率减小，每次进行日志记录文件 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比）。调整峰值活动会使事务高峰期写入日志记录的开销减至最小。

如果增大 *mincommit*，也可能需要增大 *logbufsz* 参数，以避免在这些事务高峰期让已满的日志缓冲区强制写入。在此情况下，*logbufsz* 应该等于:

```
mincommit * (事务使用的日志空间的平均值)
```

可以使用数据库系统监视器来帮助您以下列方式调整此参数:

- 计算每秒的峰值事务数:

  通过抽取典型的一天的监视器样本，您可确定您的事务高峰期。可通过添加下列监视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务使用采样技术，您可计算与下列监视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*）

相关参考:

- 『uow_log_space_used - Unit of Work Log Space Used monitor element』（*System Monitor Guide and Reference*）
- 『commit_sql_stmts - Commit Statements Attempted monitor element』（*System Monitor Guide and Reference*）
- 『rollback_sql_stmts - Rollback Statements Attempted monitor element』（*System Monitor Guide and Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

numarchretry － 出错时的重试次数

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

缺省值 [范围]
5 [0 – 65 535]

此参数指定在尝试将日志文件归档至故障转移目录之前， DB2 会尝试将日志文件归档至主归档目录或辅助归档目录的次数。仅当设置了 *failarchpath* 数据库配置参数时才使用此参数。如果未设置 *numarchretry*，则 DB2 将不断重试归档至主要日志路径或辅助日志路径。

相关参考:

|
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)

|
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)

|
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

softmax － 恢复范围和软检查点时间间隔

配置类型	数据库
参数类型	可配置
缺省值 [范围]	100 [1 – 100 * *logprimary*]
计量单位	一个主日志文件的大小百分比

此参数用来:

- 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，数据库管理器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，则数据库管理器将尝试把需要恢复的日志数保持为 3。

  要影响进行崩溃恢复所需要的日志数，数据库管理器使用此参数来触发页清除程序，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(例如，电源故障)造成数据库故障时，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崩溃恢复，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，数据库管理器使用日志控制文件。定期将此日志控制文件写入磁盘，并且根据此事件的频率，数据库管理器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，将采用该软检查点:

```
( (记录的状态和当前状态之间的空间) / logfilsiz ) *  100
```

建议: 您可能想增大或减小此参数的值，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器不但更经常地触发页清除程序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崩溃恢复期间处理的冗余日志记录数。

但是注意：更多的页清除程序触发器和更频繁的软检查点增加了与数据库日志记录相关联的开销，这可能会影响数据库管理器的性能。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，使用异步页清除程序可能有助于避免此情况。）

在这两种情况中，保留在内存中的日志控制信息不频繁更改，因而在将日志控制信息写入磁盘时不存在优势，除非日志控制信息已更改。

相关参考:

- 第 356 页的『logfilsiz － 日志文件大小』
- 第 358 页的『logprimary － 主日志文件数』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

userexit － 启用用户出口

配置类型	数据库
参数类型	可配置
缺省值 [范围]	No [Yes; No]

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日志记录。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。当数据库管理器关闭日志文件时，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* 参数之后，您必须对该数据库进行完全备份。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。在这种情况下，数据库管理器将删除 *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，并还原至循环日志记录。

相关参考:

- 『用于数据库恢复的用户出口』（*《数据恢复及高可用性指南与参考》*）
- 第 368 页的『logretain － 启用日志保留』
- 第 395 页的『user_exit_status － 用户出口状态指示符』
- 第 394 页的『backup_pending － 备份暂挂指示符』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『ROLLFORWARD DATABASE Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

vendoropt － 供应商选项

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

缺省值 [范围] Null [ ]

此参数指定 DB2 在备份、复原或装入副本操作期间可能需要用来与存储系统通信的其它参数。

相关概念:

- 第 547 页的附录 E,『使用 db2adutl 命令以及 logarchopt1 和 vendoropt 数据库配置参数来进行跨节点恢复』

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

恢复

下列参数影响数据库恢复的各个方面:

- 第 374 页的『autorestart － 启用自动重新启动』
- 第 374 页的『dft_loadrec_ses － 装入恢复会话的缺省数目』
- 第 378 页的『indexrec － 索引重新创建时间』
- 第 380 页的『num_db_backups － 数据库备份数』
- 第 380 页的『rec_his_retentn － 恢复历史保留时间段』
- 第 381 页的『trackmod － 启用跟踪已修改的页』

另请参阅第 383 页的『分布式工作单元恢复』。

当使用 Tivoli Storage Manager(TSM)时,使用下列参数:

- 第 381 页的『tsm_mgmtclass － Tivoli Storage Manager 管理类』
- 第 382 页的『tsm_nodename － Tivoli Storage Manager 节点名』
- 第 382 页的『tsm_owner － Tivoli Storage Manager 所有者名称』
- 第 383 页的『tsm_password － Tivoli Storage Manager 密码』

下列参数与高可用性灾难恢复(HADR)有关:

- 第 375 页的『hadr_db_role － HADR 数据库角色』
- 第 375 页的『hadr_local_host － HADR 本地主机名』
- 第 376 页的『hadr_local_svc － HADR 本地服务名称』

- 第 376 页的『hadr_remote_host － HADR 远程主机名』
- 第 376 页的『hadr_remote_inst － 远程服务器的 HADR 实例名』
- 第 377 页的『hadr_remote_svc - HADR 远程服务名称』
- 第 377 页的『hadr_syncmode － 处于对等状态时日志写操作的 HADR 同步方式』
- 第 378 页的『hadr_timeout － HADR 超时值』

autorestart － 启用自动重新启动

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	On [On; Off]

当此参数设置为 on 时，如果需要，数据库管理器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。*崩溃恢复*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，则执行崩溃恢复。电源故障或系统软件故障可能会导致数据库异常终止。崩溃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。崩溃恢复还会取消可能已经写入磁盘的任何未落实的事务。

如果未启用 *autorestart*，则试图与需要执行崩溃恢复（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。

相关概念:
- 『崩溃恢复』（*《数据恢复及高可用性指南与参考》*）

相关参考:
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESTART DATABASE Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

dft_loadrec_ses － 装入恢复会话的缺省数目

配置类型	数据库
参数类型	可联机配置
传播类	立即
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会话数。应将此值设置为要用来检索装入副本的最佳 I/O 会话数。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条目来覆盖此参数。

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。

仅当启用了前滚恢复时，此参数才可应用。

相关概念:

- 『Load Overview』（*Data Movement Utilities Guide and Reference*）

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 481 页的『其它变量』

hadr_db_role － HADR 数据库角色

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 参考

此参数指示数据库的当前角色，不管数据库是处于联机还是脱机状态。有效值为: STANDARD、PRIMARY 或 STANDBY。

注: 虽然 GET SNAPSHOT FOR DATABASE 命令将返回高可用性灾难恢复（HADR）状态，但是仅当数据库处于联机状态时，该命令才会返回此状态。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_local_host － HADR 本地主机名

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 空

此参数指定用于高可用性灾难恢复（HADR）TCP 通信的本地主机。可以使用主机名也可以使用 IP 地址。如果指定了主机名，但是它映射至多个 IP 地址，则会返回错误并且将不启动 HADR。如果主机名映射至多个 IP 地址（即使对主数据库和备用数据库指定同一个主机名，它也映射至多个 IP 地址），则主数据库和备用数据库会结束将此主机名映射至不同的 IP 地址，原因是一些 DNS 服务器将按不确定的顺序返回 IP 地址列表。

主机名的格式为: myserver.ibm.com。IP 地址的格式为: “12.34.56.78”。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_local_svc － HADR 本地服务名称

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值
空

此参数指定本地高可用性灾难恢复（HADR）进程将接受连接的 TCP 服务名称或端口号。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_remote_host － HADR 远程主机名

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值
空

此参数指定远程高可用性灾难恢复（HADR）节点的 TCP/IP 主机名或 IP 地址。与 *hadr_local_host* 相似，此参数必须只映射至一个 IP 地址。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_remote_inst － 远程服务器的 HADR 实例名

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型	可配置
缺省值	空

此参数指定远程服务器的实例名称。一些管理工具（例如，DB2 控制中心）使用此参数来与远程服务器联系。高可用性灾难恢复（HADR）还检查请求连接的远程数据库是否属于声明的远程实例。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_remote_svc - HADR 远程服务名称

配置类型	数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型	可配置
缺省值	空

此参数指定远程高可用性灾难恢复（HADR）节点将使用的 TCP 服务名称或端口号。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_syncmode – 处于对等状态时日志写操作的 HADR 同步方式

配置类型	数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型	可配置

缺省值 [范围] NEARSYNC [ASYNC; SYNC]

此参数指定同步方式，它确定当系统处于对等状态时主日志写操作如何与备用日志写操作同步。有效值为:

SYNC
此方式对防止发生事务丢失提供了最强有力的保护，但是事务响应时间较长。

NEARSYNC
此方式对防止发生事务丢失提供的保护稍微差一点，但是，与 SYNC 方式比较起来，NEARSYNC 方式的事务响应时间较短。

ASYNC
在主数据库发生故障的情况下，此方式发生丢失事务的可能性最大，但是，在这三种方式中，该方式的事务响应时间最短。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

hadr_timeout － HADR 超时值

配置类型 数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] 120 [1 – 655 535]

此参数指定在认为尝试通信失败之前高可用性灾难恢复（HADR）进程要等待的时间（以秒计）。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

indexrec － 索引重新创建时间

配置类型 数据库和数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

传播类 立即

缺省值 [范围]

UNIX 数据库管理器
restart [restart; access]

Windows 数据库管理器
restart [restart; access]

数据库 使用系统设置 [system; restart; access]

此参数指示数据库管理器何时将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器配置文件中指定的时间重新构建无效的索引。（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新构建无效的索引。

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 命令时重新构建无效的索引。注意，如果启用 *autorestart* 参数，则隐式发出 RESTART DATABASE 命令。

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，则可通过重新创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:

- 重新创建索引文件时可能会发生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。
- 在重新创建索引之后可能挂起意外的锁定，尤其是导致索引重新创建的用户事务从未执行过 COMMIT 或 ROLLBACK 的情况下更是如此。

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 时重新构建该索引，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”将导致重新创建索引时数据库管理器性能的降低。任何存取该特定索引或表的用户将必须等待，直到重新创建索引为止。

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重新创建索引而较长，但是一旦数据库恢复联机，正常处理将不受影响。

注：在进行数据库恢复时，将除去可在属于正在恢复的数据库的文件系统上执行的所有 SQL 过程。如果 *indexrec* 设置为 RESTART，则会从数据库目录中抽取所有 SQL 过程可执行文件，并在下一次连接至数据库时放回到该文件系统上。如果未将 *indexrec* 设置为 RESTART，则仅在第一次执行该 SQL 过程时才会将 SQL 可执行文件抽取到该文件系统上。

相关任务:

- 『备份和复原在 DB2 8.2 之前创建的 SQL 过程』（*《应用程序开发指南：构建和运行应用程序》*）

相关参考:

- 第 374 页的『autorestart － 启用自动重新启动』
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESTART DATABASE Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

num_db_backups － 数据库备份数

配置类型	数据库
参数类型	可联机配置
传播类	事务边界
缺省值 [范围]	12 [1 — 32 768]

此参数指定为一个数据库保留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条目也标记为到期。当备份标记为到期时，可从存储物理备份的地方（例如，磁盘、磁带和 TSM）除去它们。下一个数据库备份将从恢复历史文件中修剪到期的条目。

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 Data Links 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。

相关参考:

- 第 380 页的『rec_his_retentn － 恢复历史保留时间段』
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

rec_his_retentn － 恢复历史保留时间段

配置类型	数据库
参数类型	可配置
缺省值 [范围]	366 [-1; 0 — 30 000]
计量单位	天数

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。

此参数的值将覆盖 *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。

不管保留周期多小，将总是保留最近的完整数据库备份及其复原集，除非您使用带有 FORCE 选项的 PRUNE 实用程序。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『PRUNE HISTORY/LOGFILE Command』（*Command Reference*）
- 第 380 页的『num_db_backups － 数据库备份数』

trackmod － 启用跟踪已修改的页

配置类型	数据库
参数类型	可配置
缺省值 [范围]	No [Yes, No]

当此参数设置为“YES”时，数据库管理器跟踪数据库修订，以便备份实用程序可以检测到数据库页的哪些子集必须通过增量备份来检查并可能包括在备份映像中。在将此参数设置为“YES”之后，必须执行全数据库备份，才能获得可以对其执行增量备份的基线。并且，若启用此参数并创建了一个表空间，则必须执行包含该表空间的备份。此备份可以是数据库备份，也可以是表空间备份。在执行备份之后，将允许执行增量备份来包含此表空间。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

tsm_mgmtclass － Tivoli Storage Manager 管理类

配置类型	数据库
参数类型	可配置
缺省值 [范围]	Null（任何字符串）

Tivoli Storage Manager 管理类确定 TSM 服务器应如何管理正在备份的对象的备份版本。

缺省情况是没有 DB2 指定的管理类。

当执行任何 TSM 备份时，在使用数据库配置参数指定的管理类之前，TSM 首先尝试将备份对象绑定到 TSM 客户机选项文件中的 INCLUDE-EXCLUDE 列表中指定的管理

类。如果找不到匹配项，则将使用 TSM 服务器上指定的缺省 TSM 管理类。然后，TSM 将把备份对象重新绑定到数据库配置参数指定的管理类。

因此，缺省管理类以及数据库配置参数指定的管理类必须包含备份副本组，否则备份操作将失败。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『Tivoli Storage Manager』（*《数据恢复及高可用性指南与参考》*）

tsm_nodename － Tivoli Storage Manager 节点名

配置类型	数据库
参数类型	可联机配置
传播类	语句边界
缺省值 [范围]	Null（任何字符串）

此参数用于覆盖与 Tivoli Storage Manager（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个节点备份至 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）期间覆盖 *tsm_nodename*。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

tsm_owner － Tivoli Storage Manager 所有者名称

配置类型	数据库
参数类型	可联机配置
传播类	语句边界
缺省值 [范围]	Null（任何字符串）

此参数用于覆盖与 Tivoli Storage Manager（TSM）产品相关的所有者的缺省设置。所有者名称是允许您复原从另一个节点备份至 TSM 的数据库所需的。有可能在通过 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间覆盖 *tsm_owner*。

注: 所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。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

tsm_password － Tivoli Storage Manager 密码

配置类型 数据库

参数类型 可联机配置

传播类 语句边界

缺省值 [范围] Null(任何字符串)

此参数用于重设与 Tivoli Storage Manager(TSM)产品相关的密码的缺省设置。需要密码以允许您复原从另一个节点备份至 TSM 的数据库。

注: 如果在使用 DB2 执行备份(例如, 使用 BACKUP DATABASE 命令)期间覆盖了 *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。有可能在使用 DB2 执行备份期间覆盖 *tsm_nodename*。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

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:

- 『resync_interval － 事务再同步时间间隔』
- 第 384 页的『spm_log_file_sz － 同步点管理器日志文件大小』
- 第 385 页的『spm_log_path － 同步点管理器日志文件路径』
- 第 385 页的『spm_max_resync － 同步点管理器再同步代理进程限制』
- 第 386 页的『spm_name － 同步点管理器名称』
- 第 386 页的『tm_database － 事务管理器数据库名称』

resync_interval － 事务再同步时间间隔

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] 180 [1 – 60 000]

计量单位 秒

此参数指定事务管理器（TM）、资源管理器（RM）或同步点管理器（SPM）重试恢复 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。此参数也适用于联合数据库系统的恢复。

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它事务，则您可能希望增大此参数的值。如果正在使用 DB2 Connect 网关访问 DRDA2 应用程序服务器，则即使将不干扰本地数据存取，也应考虑不确定事务对该应用程序服务器可能产生的影响。如果没有不确定事务，则性能影响将最小。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

spm_log_file_sz － 同步点管理器日志文件大小

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	256 [4 — 1 000]
计量单位	页（4 KB）

此参数以 4 KB 页为单位标识“同步点管理器”（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。

建议: 同步点管理器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受保护对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，则停止数据库管理器。
3. 用新的 SPM 日志文件大小来更新数据库管理器配置。
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。其它系统可能会需要不同的除去或删除命令。）
5. 启动数据库管理器。在启动数据库管理器时创建一个指定大小的新的 SPM 日志。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

spm_log_path － 同步点管理器日志文件路径

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
sqllib/spmlog [任何有效的路径或设备]

此参数指定将“同步点管理器”（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 代理进程中有更好的并发性。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

spm_max_resync － 同步点管理器再同步代理进程限制

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

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进程数。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

spm_name － 同步点管理器名称

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可配置

缺省值　　从 TCP/IP 主机名派生

此参数向数据库管理器标识“同步点管理器”（SPM）实例的名称。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

tm_database － 事务管理器数据库名称

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可配置

缺省值 [范围]　　1ST_CONN（任何有效的数据库名称）

此参数标识每个 DB2 实例的“事务管理器”（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“同步点管理器”（SPM）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。

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。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

数据库管理

提供了大量参数，它们提供有关数据库的信息或者影响数据库的管理。这些参数分为以下几组:

- 『查询启用程序』
- 第 388 页的『属性』
- 第 390 页的『DB2 Data Links Manager』
- 第 393 页的『状态』
- 第 396 页的『编译器设置』
- 第 401 页的『自动维护』

查询启用程序

以下参数提供用于控制“查询启用程序”的信息:

- 『dyn_query_mgmt － 动态 SQL 查询管理』

dyn_query_mgmt － 动态 SQL 查询管理

配置类型	数据库
参数类型	可联机配置
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。如果此参数设置为“ENABLE”，则 Query Patroller 将捕获有关查询的信息，例如，提交者标识和估计的执行成本（由优化器计算）。这些值用来确定是否应该由 Query Patroller 根据用户级别阈值和系统级别阈值来管理查询。

如果此参数设置为“DISABLE”，则 Query Patroller 不会捕获有关提交的查询的任何信息，也不会进行查询管理。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

属性

下列参数提供有关数据库的一般信息:

- 『alt_collate － 备用整理顺序』
- 『codepage － 数据库的代码页』
- 第 389 页的『codeset － 数据库的代码集』
- 第 389 页的『collate_info － 整理信息』
- 第 389 页的『country － 数据库地域代码』
- 第 389 页的『database_level － 数据库发行版级别』
- 第 390 页的『release － 配置文件发行版级别』
- 第 390 页的『territory － 数据库地域』

除了 *alt_collate* 以外，提供这些参数仅供参考。

alt_collate － 备用整理顺序

配置类型
数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
Null [IDENTITY_16BIT]

此参数指定了要用于非 Unicode 数据库中的 Unicode 表的整理顺序。只有设置了此参数之后，才能在非 Unicode 数据库中创建 Unicode 表和例程。一旦设置了此参数，就不能更改或复位它。

不能对 Unicode 数据库设置此参数。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

codepage － 数据库的代码页

配置类型
数据库

参数类型
参考

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。

相关参考:

- 第 389 页的『codeset － 数据库的代码集』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

codeset – 数据库的代码集

配置类型 数据库

参数类型 参考

此参数显示创建数据库所使用的代码集。数据库管理器使用代码集来确定 *codepage* 参数值。

相关参考:

- 第 388 页的『codepage – 数据库的代码页』
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)

collate_info – 整理信息

只能使用 db2CfgGet 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 参考

此参数提供 260 个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”包含代码点的排序权重，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。它有三个值:

- **0** – 该顺序包含非唯一的权重。
- **1** – 该顺序包含所有唯一的权重。
- **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反向。

可在创建数据库时指定整理顺序。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)

country – 数据库地域代码

配置类型 数据库

参数类型 参考

此参数显示用来创建数据库的地域代码。

相关参考:

- 第 390 页的『territory – 数据库地域』
- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)

database_level – 数据库发行版级别

配置类型 数据库

参数类型 参考

此参数指示可使用数据库的数据库管理器的发行版级别。在迁移未完成或失败的情况下，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。

相关参考:

- 第 390 页的『release － 配置文件发行版级别』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

release － 配置文件发行版级别

配置类型 数据库管理器，数据库

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 参考

此参数指定配置文件的发行版级别。

相关参考:

- 第 389 页的『database_level － 数据库发行版级别』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

territory － 数据库地域

配置类型 数据库

参数类型 参考

此参数显示用于创建数据库的地域。数据库管理器使用 *territory* 来确定地域代码（*territory*）参数值。

相关参考:

- 第 389 页的『country － 数据库地域代码』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

DB2 Data Links Manager

下列参数与 DB2 Data Links Manager 相关:

- 第 391 页的『datalinks － 启用 Data Links 支持』
- 第 391 页的『dl_expint － Data Links 访问标记到期时间间隔』
- 第 391 页的『dl_num_copies － Data Links 副本数』
- 第 392 页的『dl_time_drop － 删除后的 Data Links 时间』
- 第 392 页的『dl_token － Data Links 标记算法』
- 第 393 页的『dl_upper － 采用大写的 Data Links 标记』
- 第 393 页的『dl_wt_iexpint － Data Links 写令牌初始到期时间间隔』

datalinks － 启用 Data Links 支持

配置类型 数据库管理器

参数类型 可配置

缺省值 [范围] NO [YES; NO]

此参数指定是否启用 Data Links 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 Data Links 支持。值“NO”指定不启用 Data Links 支持。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

dl_expint － Data Links 访问标记到期时间间隔

配置类型 数据库

参数类型 可联机配置

传播类 事务边界

缺省值 [范围] 60 [1 — 31 536 000]

计量单位 秒

此参数指定生成的文件存取控制标记持续有效的时间间隔（以秒计）。该标记的有效秒数从生成它时开始计算。Data Links 文件系统过滤器针对此到期时间检查标记的有效性。

此参数适用于指定“READ PERMISSION DB”的 DATALINK 列。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

dl_num_copies － Data Links 副本数

配置类型 数据库

参数类型 可联机配置

传播类 立即

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（例如，TSM 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。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

dl_time_drop – 删除后的 Data Links 时间

配置类型	数据库
参数类型	可联机配置
传播类	立即
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。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

dl_token – Data Links 标记算法

配置类型	数据库
参数类型	可联机配置
传播类	事务边界
缺省值 [范围]	MAC0 [MAC0; MAC1]

此参数指定在生成 DATALINK 文件存取控制标记时使用的算法。值 MAC1(消息认证代码)生成比 MAC0 更安全的消息认证代码,但也增加了性能的开销。

此参数适用于指定“READ PERMISSION DB”的 DATALINK 列。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

dl_upper – 采用大写的 Data Links 标记

配置类型	数据库
参数类型	可联机配置
传播类	事务边界
缺省值 [范围]	NO [YES; NO]

该参数指示文件存取控制标记是否使用大写字母。值“YES”指定存取控制标记中的所有字母都采用大写。值“NO”指定该标记可同时包含大写和小写字母。

此参数适用于指定“READ PERMISSION DB”的 DATALINK 列。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

dl_wt_iexpint – Data Links 写令牌初始到期时间间隔

配置类型	数据库
参数类型	可联机配置
传播类	事务边界
缺省值 [范围]	60 [1 – 31 536 000]
计量单位	秒

此参数指定从 DATALINK 列生成的写令牌的初始到期时间间隔。初始到期时间间隔是在生成令牌的时间和首次使用它打开文件的时间之间的时间段。

此参数仅适用于以 WRITE PERMISSION ADMIN 定义的 DATALINK 列。

建议: 该值应足够大才能覆盖从检索写令牌到使用它来打开文件的时间。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

状态

下列参数提供有关数据库状态的信息:

- 第 394 页的『backup_pending – 备份暂挂指示符』
- 第 394 页的『database_consistent – 数据库是一致的』
- 第 394 页的『log_retain_status – 日志保留状态指示符』
- 第 394 页的『multipage_alloc – 启用的多页文件分配』
- 第 395 页的『restore_pending – 复原暂挂』
- 第 395 页的『rollfwd_pending – 前滚暂挂指示符』
- 第 395 页的『user_exit_status – 用户出口状态指示符』

backup_pending － 备份暂挂指示符

配置类型 数据库

参数类型 参考

如果设置为 on，此参数指示您在存取该数据库之前必须对该数据库进行完全备份。此参数仅在下列情况下才为 on：该数据库配置更改，以便该数据库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。）

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

database_consistent － 数据库是一致的

配置类型 数据库

参数类型 参考

此参数指示数据库是否处于一致状态。

YES 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。

NO 指示一个事务暂挂，或对该数据库执行的某个其它任务暂挂，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。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

log_retain_status － 日志保留状态指示符

配置类型 数据库

参数类型 参考

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* 参数设置等于 Recovery 时，设置此参数。

相关参考:

- 第 368 页的『logretain － 启用日志保留』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

multipage_alloc － 启用的多页文件分配

配置类型 数据库

参数类型 参考

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 Yes：启用了多页文件分配。

| 创建数据库之后，不能将此参数设置为 No。一旦启用了多页文件分配就不能再禁用它。
| 如果不想使用多页文件分配，则在创建数据库之前必须适当地设置 DB2_NO_MPFA_FOR_NEW_DB DB2 注册表变量。可以使用 **db2empfa** 工具来对当
| 前禁用了多页文件分配的数据库启用多页文件分配。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 第 469 页的『性能变量』

restore_pending － 复原暂挂

配置类型	数据库
参数类型	参考

此参数说明数据库中是否有 RESTORE PENDING 状态。

相关参考:

- 第 372 页的『userexit － 启用用户出口』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

rollfwd_pending － 前滚暂挂指示符

配置类型	数据库
参数类型	参考

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）。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

user_exit_status － 用户出口状态指示符

配置类型	数据库
参数类型	参考

如果设置为 Yes，则指示启用数据库管理器用于前滚恢复，并且将使用用户出口程序在数据库管理器调用日志文件时归档和检索日志文件。

相关参考:

- 第 372 页的『userexit － 启用用户出口』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）

编译器设置

下列参数提供影响编译器的信息:

- 『dft_degree – 缺省并行度』
- 『dft_mttb_types – 缺省维护的用于优化的表类型』
- 第 397 页的『dft_queryopt – 缺省查询优化类』
- 第 397 页的『dft_refresh_age – 缺省刷新寿命』
- 第 398 页的『dft_sqlmathwarn – 算术异常时继续进行』
- 第 399 页的『num_freqvalues – 保留的高频值数目』
- 第 400 页的『num_quantiles – 列的分位数的数目』

dft_degree – 缺省并行度

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	1 [-1, 1 – 32 767]

此参数指定 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项的缺省值。

缺省值为 1。

值为 1 则意味着没有分区内并行性。值为 -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 绑定选项指定 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最大运行时分区内并行度。“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。

实际使用的运行时并行度是下列值中最小的一个:

- *max_querydegree* 配置参数
- 应用程序运行时并行度
- SQL 语句编译并行度

相关参考:

- 第 414 页的『max_querydegree – 最大查询并行度』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

dft_mttb_types – 缺省维护的用于优化的表类型

配置类型	数据库
参数类型	可配置
缺省值 [范围]	SYSTEM [ALL, NONE, FEDERATED_TOOL, SYSTEM, USER 或者值的列表]

此参数指定 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 专用寄存器的缺省值。此专用寄存器的值确定在优化查询期间将使用哪些类型的延迟刷新的具体查询表。

可以指定用逗号分隔的值的列表；例如，“USER,FEDERATED_TOOL”。不能将 ALL 或 NONE 与其它值一起列示，并且不能多次指定同一个值。

相关参考:

- 『CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register』（*SQL Reference, Volume 1*）

dft_queryopt － 缺省查询优化类

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	5 [0 — 9]
计量单位	查询优化级别（请参阅以下内容）

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。此参数提供了额外的灵活性，这是通过设置在既没有使用 SET CURRENT QUERY OPTIMIZATION 语句也没有在绑定命令上的 QUERYOPT 选项时使用缺省查询优化级别实现的。

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

相关参考:

- 『SET CURRENT QUERY OPTIMIZATION statement』（*SQL Reference, Volume 2*）
- 『BIND Command』（*Command Reference*）
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）
- 『LIST DRDA INDOUBT TRANSACTIONS Command』（*Command Reference*）

dft_refresh_age － 缺省刷新寿命

配置类型	数据库
参数类型	可配置
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持续时间值，其数据类型为 DECIMAL(20,6)。此持续时间代表自在特定 REFRESH DEFERRED 具体查询表上处理 REFRESH TABLE 语句之后的最长持续时间，在此期间该总结表可用来优化查询的处理。如果 CURRENT REFRESH AGE 的值为 99999999999999（ANY），且 QUERY OPTIMIZATION 级别为二、五或更多，则考虑使用 REFRESH DEFERRED 具体查询表来优化动态 SQL 查询的处理。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

dft_sqlmathwarn － 算术异常时继续进行

配置类型	数据库
参数类型	可配置
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，此参数值与绑定时的程序包相关。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也可能会更改。反过来，这也可能会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求值的那一行的更新将失败，而对于不需要约束重新求值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。

在将 *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手工复查约束。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求值。

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。如果在其它数据库管理器中处理提供结果的 SQL 语句，而不考虑发生的算术异常，则可能发生此情况。

相关参考：

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

num_freqvalues – 保留的高频值数目

配置类型	数据库
参数类型	可联机配置
传播类	立即
缺省值 [范围]	10 [0 — 32 767]
计量单位	计数器

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。

还可在表或列级别指定作为 RUNSTATS 命令的一部分保留的高频值的数目。如果未指定任何值，则将使用 *num_freqvalues* 配置参数值。

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连接并且您已重新连接至数据库之后，运行 RUNSTATS 命令。
- 重新绑定任何包含静态 SQL 的程序包。

RUNSTATS 命令通过使用 NUM_REQVALUES 选项允许保留的高频值数目的规范。通过 RUNSTATS 命令更改保留的高频值数目比通过使用 *num_freqvalues* 数据库配置参数进行更改来得容易。

使用 RUNSTATS 时，可在表级别和列级别限制收集的高频值的数目。这允许您通过减少列的分布统计信息来优化目录中占用的空间，在此空间中不能使用这些目录，而这些目录仍然使用关键列的信息。

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排序的 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。

相关参考：

- 第 400 页的『num_quantiles － 列的分位数的数目』
- 第 325 页的『stat_heap_sz － 统计信息堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

num_quantiles － 列的分位数的数目

配置类型	数据库
参数类型	可联机配置的
传播类	立即
缺省值 [范围]	20 [0 – 32 767]
计量单位	计数器

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。

还可在表或列级别指定作为 RUNSTATS 命令的一部分收集的分位数的数目。如果未指定任何值，则将使用 *num_quantiles* 配置参数值。

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。在其它优化器决策中，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连接并且您已重新连接至数据库之后，运行 RUNSTATS 命令。

- 重新绑定任何包含静态 SQL 的程序包。

RUNSTATS 命令通过使用 NUM_QUANTILES 选项来允许将要收集的分位数的数目的规范。通过 RUNSTATS 命令更改将收集的分位数的数目比通过使用 *num_quantiles* 数据库配置参数进行更改容易。

使用 RUNSTATS 时，可在表级别和列级别限制收集的分位数的数目。这允许您通过减少列的分布统计信息来优化目录中占用的空间，在此空间中不能使用这些目录，而这些目录仍然使用关键列的信息。

建议：此参数的缺省值保证，对于任何单方范围谓词（>、>=、< 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%。计算分位数数目的近似值的简单方法是：

- 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它类型的范围谓词（<、<=、> 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。

相关参考：

- 第 399 页的『num_freqvalues － 保留的高频值数目』
- 第 325 页的『stat_heap_sz － 统计信息堆大小』
- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE CONFIGURATION Command』（*Command Reference*）

自动维护

以下参数允许您控制多个 DB2 实用程序的自动维护活动：

- 『autonomic_switches － 自动维护开关』

autonomic_switches － 自动维护开关

配置类型
数据库

适用于
- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值
关闭所有开关

此参数使您可以设置许多开关，这些开关在内部分别由该参数的一位表示。可以通过设置下列参数来分别更新这些开关中的每一个开关：

auto_maint
此参数是所有其它自动维护数据库配置参数

	（*auto_db_backup*、*auto_tbl_maint*、*auto_runstats*、*auto_stats_prof*、*auto_prof_upd* 和 *auto_reorg*）的父参数。当禁用此参数时，也就禁用了它的所有子参数，但是，这些子参数的设置不会更改，这些设置是记录在数据库配置文件中的。当启用此父参数时，记录的它的子参数的值就会生效。因此，可以全局启用或禁用自动维护。
auto_db_backup	此自动维护参数将启用或禁用数据库的自动备份操作。备份策略（已定义的一组规则或准则）可以用来指定自动行为。备份策略的目标是确保定期备份数据库。当第一次运行 “DB2 健康监视器” 时，会自动创建数据库的备份策略。如果要启用，则必须将此参数设置为 ON，并且还必须启用它的父参数。
auto_tbl_maint	此参数是所有表维护参数（*auto_runstats*、*auto_stats_prof*、*auto_prof_upd* 和 *auto_reorg*）的父参数。当禁用此参数时，也就禁用了它的所有子参数，但是，这些子参数的设置不会更改，这些设置是记录在数据库配置文件中的。当启用此父参数时，记录的它的子参数的值就会生效。因此，可以全局启用或禁用表维护。
auto_runstats	此自动维护表参数将启用或禁用数据库的自动表运行统计操作。运行统计策略（已定义的一组规则或准则）可以用来指定自动行为。优化器使用 runstats 实用程序收集的统计信息来确定存取物理数据的最佳方案。如果要启用，则必须将此参数设置为 ON，并且还必须启用它的父参数。
auto_stats_prof	当启用此参数时，此自动维护表参数将打开统计概要文件的生成，生成统计概要文件可以用来提高这样一些应用程序的性能：它们的工作负载包括对几个表的复杂查询、许多谓词、连接和分组操作。如果要启用，则必须将此参数设置为 ON，并且还必须启用它的父参数。
auto_prof_upd	当启用此参数时，此自动维护表参数（*auto_stats_prof* 的子参数）指定要使用建议来更新 runstats 概要文件。当禁用此参数时，会将建议存储在 opt_feedback_ranking 表中，在手工更新 runstats 概要文件时可检查该表。如果要启用，则必须将此参数设置为 ON，并且还必须启用它的父参数。
auto_reorg	此自动维护表参数将启用或禁用数据库的自动重组表和索引。重组策略（已定义的一组规则或准则）可以用来指定自动行为。如果要启用，则必须将此参数设置为 ON，并且还必须启用它的父参数。

相关参考:

- 『GET DATABASE CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE CONFIGURATION Command』（*Command Reference*）

• 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

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 405 页的『DB2 发现』

通信协议设置

可以使用下列参数来配置数据库客户机和数据库服务器:

- 『nname – NetBIOS 工作站名称』
- 第 404 页的『svcename – TCP/IP 服务名称』
- 第 404 页的『tpname – APPC 事务程序名』

nname – NetBIOS 工作站名称

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

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。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『CATALOG NETBIOS NODE Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

svcename – TCP/IP 服务名称

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

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器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，数据库服务器必须侦听指定给该服务器的端口。数据库服务器的系统管理员必须保留一个端口（端口号为 *n*）并在服务器的 services 文件中定义其相关 TCP/IP 服务名称。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（端口号为 *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（端口号为 *n*）和它的 TCP/IP 服务名称。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（端口号为 *n+1*）。

在基于 UNIX 的系统上，services 文件位于：`/etc/services`

应将 *svcename* 参数设置为与主连接端口相关的服务名称，以便当启动数据库服务器时，它可以确定在哪个端口上侦听入局连接请求。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称。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*）。

相关参考：

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

tpname – APPC 事务程序名

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

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。

建议: 此名称中只接受如下列示的字符:

- 字母（A 至 Z; 或 a 至 z）
- 数字（0 至 9）
- 美元符号（$）、数值符号（#）、位置符号（@）和句点（.）

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

DB2 发现

可使用下列参数来建立“DB2 发现”:

- 『discover － 发现方式』
- 第 406 页的『discover_db － 发现数据库』
- 第 406 页的『discover_inst － 发现服务器实例』

discover － 发现方式

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] SEARCH [DISABLE, KNOWN, SEARCH]

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。

缺省发现方式为 SEARCH。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 第 459 页的『通信变量』

discover_db – 发现数据库

配置类型	数据库
参数类型	可联机配置
传播类	立即
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，以防止被发现进程发现。此操作可与数据库上的其它数据库安全性控制一起完成。

相关参考:

- 『GET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE CONFIGURATION Command』(*Command Reference*)

discover_inst – 发现服务器实例

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 客户机 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	ENABLE [ENABLE, DISABLE]

此参数指定是否可用“DB2 发现”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。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

分区数据库环境

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 413 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:

- 『conn_elapse － 连接经过时间』
- 第 408 页的『fcm_num_anchors － FCM 消息锚点数』
- 第 408 页的『fcm_num_buffers － FCM 缓冲区数目』
- 第 409 页的『fcm_num_connect － FCM 连接条目数』
- 第 410 页的『fcm_num_rqb － FCM 请求块数』
- 第 411 页的『max_connretries － 节点连接重试次数』
- 第 412 页的『max_time_diff － 节点之间的最大时间差』
- 第 412 页的『start_stop_time － 启动和停止超时』

conn_elapse － 连接经过时间

配置类型	数据库管理器
适用于	带有本地和远程客户机的分区数据库服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	10 [0–100]
计量单位	秒

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。如果尝试连接的次数达到 *max_connretries* 参数指定的次数且始终超时，则发出错误。

相关参考:

- 第 411 页的『max_connretries － 节点连接重试次数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

fcm_num_anchors － FCM 消息锚点数

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器
- 带有本地客户机的卫星数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]
-1 [-1，128 — *fcm_num_rqb*]

在非分区数据库系统上，*intra_parallel* 参数必须是活动的然后才能使用 *fcm_num_anchors*。

此参数指定 FCM *消息锚点*数。代理进程使用消息锚点来互相发送消息。缺省值（-1）指定为 *fcm_num_rqb* 指定的值的 75%。

相关概念:

- 『快速通信管理器（FCM）通信』（*《管理指南：实现》*）

相关参考:

- 第 408 页的『fcm_num_buffers － FCM 缓冲区数目』
- 第 409 页的『fcm_num_connect － FCM 连接条目数』
- 第 410 页的『fcm_num_rqb － FCM 请求块数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

fcm_num_buffers － FCM 缓冲区数目

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]

32 位平台

512，1 024 或 4 096 [128 — *65 300*]

64 位平台

512，1 024 或 4 096 [128 — *524 288*]

- 带本地客户机和远程客户机的数据库服务器：缺省值为 1 024
- 带本地客户机的数据库服务器：缺省值为 512
- 带本地客户机和远程客户机的分区数据库服务器：缺省值为 4 096

在单一分区数据库系统上，如果 *intra_parallel* 参数不是活动的，则不使用此参数。

此参数指定数据库服务器之间及内部用于内部通信（消息）的 4 KB 缓冲区数。

如果同一机器上有多个逻辑节点，则您可能会发现需要增大此参数的值。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消息缓冲区用尽，您可能也会发现需要增大此参数的值。

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：

- 如果在数据库管理器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 缓冲区堆。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）。
- 如果在数据库管理器所使用的一般内存中没有足够的空间，则将从另一个内存区（AIX 共享内存集）中分配 FCM 缓冲区堆，即，由相同机器上所有多逻辑节点共享。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。这是所有非 AIX 平台的缺省配置。

复查正在使用的值。考虑在多逻辑节点所驻留的机器上总共将分配多少 FCM 缓冲区。

相关概念:

- 『快速通信管理器（FCM）通信』（*《管理指南：实现》*）

相关参考:

- 第 409 页的『fcm_num_connect － FCM 连接条目数』
- 第 408 页的『fcm_num_anchors － FCM 消息锚点数』
- 第 410 页的『fcm_num_rqb － FCM 请求块数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

fcm_num_connect － FCM 连接条目数

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器
- 带有本地客户机的卫星数据库服务器

参数类型 可联机配置

传播类 立即

缺省值 [范围] -1 [-1, 128 — *fcm_num_rqb*]

在非分区数据库系统上，*intra_parallel* 参数必须是活动的然后才能使用 *fcm_num_connect*。

此参数指定 FCM *连接条目*数。代理进程使用连接条目来互相传递数据。缺省值（-1）指定为 *fcm_num_rqb* 指定的值的 75%。

相关概念:

- 『快速通信管理器（FCM）通信』（*《管理指南: 实现》*）

相关参考:

- 第 408 页的『fcm_num_buffers － FCM 缓冲区数目』
- 第 408 页的『fcm_num_anchors － FCM 消息锚点数』
- 第 410 页的『fcm_num_rqb － FCM 请求块数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

fcm_num_rqb － FCM 请求块数

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器
- 带有本地客户机的卫星数据库服务器

参数类型 可联机配置

传播类 立即

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]

Windows NT 32 位
10 000 [250 — 2 097 152]

Windows NT 64 位
256，512，2 048 [128 — 524 288]

- 对于带本地客户机和远程客户机的数据库服务器，缺省值为 512。
- 对于带本地客户机的数据库服务器，缺省值为 256。
- 对于带本地客户机和远程客户机的分区数据库服务器，缺省值为 2 048。

在非分区数据库系统上，*intra_parallel* 参数必须是活动的然后才能使用 *fcm_num_rqb*。

此参数指定 FCM *请求块*的数目。请求块是一种介质，通过它可以在 FCM 守护进程与代理进程之间传递信息，也可以在代理进程之间传递信息。

对于请求块的需求将随着系统上的用户数、系统上的数据库分区服务器数目和查询的复杂程度不同而有所不同。当精确调整此参数时，首先使用缺省值，然后使用来自数据库系统监视器的结果。

相关概念:

- 『快速通信管理器（FCM）通信』（*《管理指南: 实现》*）

相关参考:

- 第 408 页的『fcm_num_buffers － FCM 缓冲区数目』
- 第 409 页的『fcm_num_connect － FCM 连接条目数』
- 第 408 页的『fcm_num_anchors － FCM 消息锚点数』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

max_connretries － 节点连接重试次数

配置类型	数据库管理器
适用于	带有本地和远程客户机的分区数据库服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	5 [0–100]

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。

相关参考:

- 第 407 页的『conn_elapse － 连接经过时间』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

max_time_diff － 节点之间的最大时间差

配置类型	数据库管理器
适用于	带有本地和远程客户机的分区数据库服务器
参数类型	可配置
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示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。

如果两个或多个数据库分区服务器与某一事务相关联，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，并且返回 SQLCODE。（仅当数据修改与事务相关时，才拒绝事务。）

DB2 使用*全球标准时间*（UTC），所以当设置此参数时，不用考虑不同的时区。“全球标准时间”与“格林威治标准时间”是一样的。

相关参考:
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

start_stop_time － 启动和停止超时

配置类型	数据库管理器
适用于	带有本地和远程客户机的数据库服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	10 [1 — 1 440]
计量单位	分钟

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。在 ADD DBPARTITIONNUM 操作期间，它也用作超时值。

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主目录下 `sqllib` 子目录的 `log` 子目录中的 `db2start` 错误日志发送一条消息。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主目录下 `sqllib` 子目录的 `log` 子目录中的 `db2stop` 错误日志发送一条消息。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它们是停止的。)

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『ADD DBPARTITIONNUM Command』(*Command Reference*)

并行处理

下列参数提供有关并行处理的信息:

- 『intra_parallel – 启用分区内并行性』
- 第 414 页的『max_querydegree – 最大查询并行度』

intra_parallel – 启用分区内并行性

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

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器的硬件。

此参数指定数据库管理器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绑定至数据库，并且可能会使性能有一定下降。

相关参考:

- 第 414 页的『max_querydegree – 最大查询并行度』

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

max_querydegree – 最大查询并行度

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可联机配置

传播类　　语句边界

缺省值 [范围]　　-1 (ANY) [ANY, 1—32 767]（ANY 意味着由系统确定）

此参数指定用于在数据库管理器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 绑定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。实际使用的运行时并行度是下列值中较小的那一个:

- *max_querydegree* 配置参数
- 应用程序运行时并行度
- 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个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。不受以上提到的其它参数、绑定选项或专用寄存器的影响。

相关参考:

- 第 413 页的『intra_parallel – 启用分区内并行性』
- 第 396 页的『dft_degree – 缺省并行度』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

• 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

实例管理

有大量参数可帮助您管理数据库管理器实例。这些参数分为以下几个类别:

- 『诊断』
- 第 418 页的『数据库系统监视器参数』
- 第 419 页的『系统管理』
- 第 427 页的『实例管理』

诊断

下列参数允许您控制数据库管理器提供的诊断信息:

- 『diaglevel － 诊断错误捕获级别』
- 第 416 页的『diagpath － 诊断数据目录路径』
- 第 417 页的『health_mon － 运行状况监视』
- 第 417 页的『notifylevel － 通知级别』

diaglevel － 诊断错误捕获级别

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可联机配置

传播类 立即

缺省值 [范围] 3 [0 — 4]

此参数指定将记录在 db2diag.log 文件中的诊断错误类型。有效值为:

0 － 未捕获到诊断数据

1 － 仅严重错误

2 － 所有错误

3 － 所有错误和警告

4 － 所有错误、警告以及参考消息

diagpath 配置参数用来指定一个目录，它将包含根据 *diaglevel* 参数的值可能会生成的错误文件、事件日志文件（仅在 Windows NT 上）、报警日志文件以及任何转储文件。

建议: 您可能希望增大此参数的值以收集更多问题确定数据以帮助解决问题。

相关参考:

- 第 416 页的『diagpath － 诊断数据目录路径』

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

diagpath － 诊断数据目录路径

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]
Null（任何有效路径名）

此参数允许指定 DB2 诊断信息的全限定路径。根据您的平台，此目录可能包含转储文件、陷阱文件、错误日志、通知文件和报警日志文件。

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` 是 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` 是实例所有者的名称。

    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 是实例概要文件目录的名称，而 DB2INSTANCE 是实例所有者的名称。
- 对于基于 UNIX 的环境: 信息写入 `INSTHOME/sqllib/db2dump`，其中 `INSTHOME` 是实例所有者的主目录。

建议: 使用缺省值或获得多个实例的诊断路径的中央位置。

在分区数据库环境中，指定的路径必须驻留在共享文件系统中。

相关参考:

- 第 415 页的『diaglevel － 诊断错误捕获级别』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

health_mon － 运行状况监视

配置类型
数据库管理器

参数类型
可联机配置

传播类
立即

缺省值 [范围]
Off [On; Off]

相关参数

此参数允许您指定是否想要根据各种健康指示器来监视实例、它的相关数据库和数据库对象。如果打开了 *health_mon*，则代理进程将收集关于所选对象的运行状况的信息。如果认为对象处于不健康的状态，则根据您设置的阈值，可以发送通知，并且可以自动执行操作。如果关闭了 *health_mon*（这是缺省情况），则不会监视对象的运行状况。

可以使用“健康中心”或 CLP 来选择想要监视的实例和数据库对象。还可以根据健康监视器收集的数据来指定应将通知发送至何处，以及执行哪些操作。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

notifylevel － 通知级别

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]
3 [0 — 4]

此参数指定写入管理通知日志的管理通知消息的类型。在 UNIX 平台上，管理通知日志是称为 `instance.nfy` 的文本文件。在 Windows 上，所有管理通知消息都写到“事件日志”中。错误可由 DB2、“健康监视器”、Capture 和 Apply 程序和用户应用程序写入。

此参数的有效值为:

0 － 未捕获任何管理通知消息。（不建议使用此设置。）

1 － 致命或不可恢复错误。仅记录致命和不可恢复错误。要从这些情况中的某些情况进行恢复，可能需要来自 DB2 服务机构的帮助。

2 － 需要立即操作。记录了需要来自系统管理员或数据库管理员立即注意的情况。如果未解决该情况，则它可能会导致致命错误。非常重要并且没有错误的活动（例如，恢复）的通知可能也在此级别记录。此级别将捕获“健康监视器”警报。

3 － 重要信息，不需要立即操作。记录没有威胁也不需要立即操作但是可能表示并非最佳系统的情况。此级别将捕获“健康监视器”警报和“健康监视器”警告和“健康监视器”引起注意信息。

4 － 参考消息。

管理通知日志包括具有最多且包括 *notifylevel* 的值的消息。例如，将 *notifylevel* 设置为 3 将导致管理通知日志包括可应用于级别 1、2 和 3 的消息。

要使用户应用程序能够写入通知文件或“Windows 事件日志”，它必须调用 db2AdminMsgWrite API。

建议: 您可能希望增大此参数的值以收集更多问题确定数据以帮助解决问题。注意，您必须将 *notifylevel* 的值设置为 2 或更高，以便“健康监视器”向在其配置中定义的联系人发送所有通知。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

数据库系统监视器参数

以下参数允许您控制数据库系统监视器的各个方面:

- 『dft_monswitches － 缺省数据库系统监视器开关』

dft_monswitches － 缺省数据库系统监视器开关

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可联机配置
传播类	立即
缺省值	所有开关关闭，但 dft_mon_timestamp 除外，该开关缺省情况下是打开的

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。可以通过设置下列参数来分别更新这些开关中的每一个开关:

dft_mon_uow	快照监视器的工作单元（UOW）开关的缺省值
dft_mon_stmt	快照监视器的语句开关的缺省值
dft_mon_table	快照监视器的表开关的缺省值
dft_mon_bufpool	快照监视器的缓冲池开关的缺省值
dft_mon_lock	快照监视器的锁定开关的缺省值
dft_mon_sort	快照监视器的排序开关的缺省值
dft_mon_timestamp	快照监视器的时间戳记开关的缺省值

建议: 任何打开的开关（dft_mon_timestamp 除外）指示数据库管理器收集与该开关相关的监视器数据。收集更多监视器数据将增加数据库管理器的开销，这会影响系统性能。在 CPU 利用率达到 100% 时应将 dft_mon_timestamp 开关关闭。当此情况发生时，发出时间戳记所需的 CPU 时间急剧增长。因此，如果时间戳记开关已关闭，则在监视器开关控制下的其它数据的总体成本大幅减少。

当应用程序发出其第一个监视请求（例如，设置开关、激活事件监视器、生成快照）时，所有监视应用程序继承这些缺省开关设置。仅当想从启动数据库管理器起开始收集数据时，才应打开配置文件中的开关。（否则，每个监视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。）

相关参考:

- 『GET MONITOR SWITCHES Command』（*Command Reference*）
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

系统管理

下列参数与系统管理有关:

comm_bandwidth － 通信带宽

配置类型	数据库管理器

适用于　带有本地和远程客户机的分区数据库服务器

参数类型　可联机配置

传播类　语句边界

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 优化器使用它来估计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生产环境模型，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绑定应用程序（使用 REBIND PACKAGE 命令）。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

cpuspeed – CPU 速度

配置类型　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　可联机配置

传播类　语句边界

缺省值 [范围]　-1 [1^{-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。

计量单位　秒

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器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，基准程序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生产环境模型，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绑定应用程序（使用 REBIND PACKAGE 命令）。

相关参考:
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

dft_account_str － 缺省对方付费帐户

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 客户机 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（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从应用程序请求器发送至 DRDA 应用程序服务器。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访问关联的机制。

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 转发记帐字符串的低级别数据库客户机（版本 2 之前的任何版本），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（_）。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

federated － 联合数据库系统支持

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	No [Yes; No]

此参数启用或禁用对落实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。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

jdk_path － Software Developer's Kit for Java 安装路径

配置类型	数据库管理器
适用于	• 带有本地和远程客户机的数据库服务器 • 带有本地客户机的数据库服务器 • 带有本地和远程客户机的分区数据库服务器
参数类型	可配置
缺省值 [范围]	Null（有效路径）

此参数指定安装要用于运行 Java 存储过程和用户定义的函数的 Java 软件开发者工具箱（SDK）的目录。 Java 解释器使用的 CLASSPATH 和其它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 SDK 时应指定值。

相关参考:

- 第 333 页的『java_heap_sz － 最大 Java 解释器堆大小』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

nodetype － 机器节点类型

配置类型
: 数据库管理器

适用于
- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
: 参考

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器配置类型的信息。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，并能访问其它远程数据库服务器。
- **客户机** － 能够访问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。并且能够访问其它远程数据库服务器。
- **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，能够访问其它远程数据库服务器，并且能够实现分区并行性。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

numdb － 同时活动数据库的最大数目（包括主机数据库和 iSeries 数据库管理器）

配置类型
: 数据库管理器

适用于
- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
: 可配置

缺省值 [范围]

UNIX 8 [1 — 256]

带本地客户机和远程客户机的 Windows 数据库服务器
8 [1 — 256]

带本地客户机的 Windows 数据库服务器
3 [1 — 256]

计量单位
计数器

此参数指定可以同时活动（即，具有与它们相连接的应用程序）的本地数据库数或者是可以在 DB2 Connect 服务器上编目的不同数据库别名的最大数目。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。

建议：一般情况下，最好将此值设置为已经为数据库管理器定义的数据库的实际数目，并为此值添加大约 10% 的增量以满足将来的增长。

更改 *numdb* 参数可能会影响已分配的总内存量。因此，建议不要频繁更新此参数。当更新此参数时，应考虑可为数据库或连接至该数据库的应用程序分配内存的其它配置参数。

相关参考:
- 第 315 页的『app_ctl_heap_sz － 应用程序控制堆大小』
- 第 324 页的『sortheap － 排序堆大小』
- 第 326 页的『aslheapsz － 应用程序支持层堆大小』
- 第 319 页的『applheapsz － 应用程序堆大小』
- 第 310 页的『locklist － 锁定列表的最大存储器』
- 第 309 页的『dbheap － 数据库堆』
- 第 326 页的『stmtheap － 语句堆大小』
- 第 334 页的『mon_heap_sz － 数据库系统监视器堆大小』
- 第 325 页的『stat_heap_sz － 统计信息堆大小』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 第 308 页的『database_memory － 数据库共享内存大小』

tp_mon_name － 事务处理器监视器名称

配置类型
数据库管理器

适用于
- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值
无缺省值

有效值

- CICS
- MQ
- ENCINA
- CB
- SF
- TUXEDO
- TOPEND
- 空白或其它值（对于 UNIX 和 Windows；对于 Solaris 或 SINIX 则没有其它可能值）

此参数标识正在使用的事务处理（TP）监视器产品的名称。

- 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 Transaction Server 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 Windows NT 上的先前版本的 DB2 通用数据库中，此参数包含 DLL 的路径和名称，该 DLL 包含“XA 事务管理器”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 监视器”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 和 Windows NT 环境就是这样的。

TXSeries CICS 和 Encina 用户: 在 Windows NT 上的此产品的先前版本中，需要将此参数配置为“libEncServer:C”或“libEncServer:E”。虽然仍受支持，但是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 Windows NT 上此产品的先前版本中，需要将此参数配置为“mqmax”。虽然仍受支持，但是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 Windows NT 上此产品的先前版本中，需要将此参数配置为“somtrx1i”。虽然仍受支持，但是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 Windows NT 上此产品的先前版本中，需要将此参数配置为“ibmsfDB2”。虽然仍受支持，但是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“事务处理监视器”正在使用单一 DB2 实例，则必须使用此功能。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『xa_open 字符串格式』（*《管理指南: 计划》*）

util_impact_lim － 实例影响策略

配置类型
数据库管理器

适用于

- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

缺省值 [范围]
100 [1 - 100]

计量单位
可允许的对工作负载的影响的百分比

此参数允许数据库管理员（DBA）限制已调速实用程序对工作负载的性能降低。然后，DBA 可在关键生产时间段运行联机实用程序，并保证对生产工作的性能影响在可接受的限制之内。

例如，指定 *util_impact_lim*（影响策略）值为 10 的 DBA 可能期望已调速备份调用影响的工作负载不会超过 10%。

如果 *util_impact_lim* 为 100（缺省值），则不会对任何实用程序调用进行调速。在这种情况下，实用程序可能对工作负载有任意的（和不期望的）影响。如果将 *util_impact_lim* 设置为小于 100 的值，则可以调速方式调用实用程序。要以调速方式运行实用程序，还必须使用非零优先级调用该实用程序。

建议: 大多数用户将从将 *util_impact_lim* 设置为低值（例如，1 与 10 之间的值）中受益。

完成已调速实用程序所花的时间通常比完成非调速实用程序要长。如果发现实用程序运行的时间过长，则增大 *util_impact_lim* 的值，或通过将 *util_impact_lim* 设置为 100 来整体禁用调速。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

实例管理

下列参数与数据库管理器实例的安全性和管理相关:

- 『authentication － 认证类型』
- 第 428 页的『catalog_noauth － 在没有权限的情况下允许编目』
- 第 429 页的『clnt_krb_plugin － 客户机 Kerberos 插件』
- 第 429 页的『clnt_pw_plugin － 客户机用户标识密码插件』
- 第 430 页的『dftdbpath － 缺省值数据库路径』
- 第 431 页的『fed_noauth － 绕过联合认证』
- 第 431 页的『group_plugin － 组插件』
- 第 432 页的『local_gssplugin － 用于本地实例级别授权的 GSS API 插件』
- 第 432 页的『srvcon_auth － 服务器上的入局连接的认证类型』
- 第 433 页的『srvcon_gssplugin_list － 用于服务器上的入局连接的 GSS API 插件的列表』
- 第 434 页的『srvcon_pw_plugin － 服务器上的入局连接的用户标识密码插件』
- 第 434 页的『srv_plugin_mode － 服务器插件方式』
- 第 435 页的『sysadm_group － 系统管理权限组名』
- 第 436 页的『sysctrl_group － 系统控制权限组名』
- 第 436 页的『sysmaint_group － 系统维护权限组名』
- 第 437 页的『sysmon_group － 系统监视器权限组名』
- 第 438 页的『trust_allclnts － 信赖所有客户机』
- 第 439 页的『trust_clntauth － 可信的客户机认证』
- 第 439 页的『use_sna_auth － 使用 SNA 认证』

authentication － 认证类型

配置类型　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　可配置

缺省值 [范围]　SERVER [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

此参数指定并确定用户的认证如何进行以及在何处进行。

如果认证是 SERVER，则将用户标识和密码从客户机发送至服务器，以便可在服务器上进行认证。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密码。

注： 对于符合公共条件的配置，SERVER 是唯一受支持的值。

值 CLIENT 指示所有认证都在客户机上进行。不需要在服务器上进行认证。

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。

值 GSSPLUGIN 意味着使用外部的基于 GSSAPI 的安全机制来执行认证。如果支持 GSSPLUGIN 安全机制的服务器和客户机的认证类型为 GSS_SERVER_ENCRYPT，则有效系统认证类型是 GSSPLUGIN（即，如果客户机支持服务器的其中一个插件）。如果客户机不支持 GSSPLUGIN 安全机制，则有效的系统认证类型等价于 SERVER_ENCRYPT。

建议： 通常，使用缺省值（SERVER）就可以了。

相关参考：

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

catalog_noauth － 在没有权限的情况下允许编目

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]

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
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。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

clnt_krb_plugin － 客户机 Kerberos 插件

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
Null 或 IBMkrb5 [任何有效字符串]

此参数指定要用于客户机端认证和本地授权的缺省 Kerberos 插件库的名称。缺省情况下，在基于 UNIX 的系统上，该值为空值，而在 Windows 操作系统上为 `IBMkrb5`。当使用 KERBEROS 认证来对客户机进行认证时，使用此插件。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

clnt_pw_plugin － 客户机用户标识密码插件

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
Null（任何有效字符串）

此参数指定要用于客户机端认证和本地授权的用户标识密码插件库的名称。缺省情况下，该值是空值并且使用 DB2 提供的用户标识密码插件库。当使用 CLIENT、SERVER 或 SERVER_ENCRYPT 认证来对客户机进行认证时，使用该插件。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

dftdbpath － 缺省值数据库路径

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可联机配置

传播类
立即

缺省值 [范围]

UNIX 实例所有者的主目录 [任何现有路径]

Windows 安装了 DB2 的驱动器 [任何现有路径]

此参数包含用来在数据库管理器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。

在分区数据库环境中，应确保正在其上创建数据库的路径不是安装了 NFS 的路径（在基于 UNIX 的平台上）或网络驱动器（在 Windows 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。系统将节点名追加至该路径的末尾。

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它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 Windows，*dftdbpath* 可以为驱动器盘符，可选择后跟冒号。

建议: 如果有可能，将高容量数据库放在不同于其它被频繁存取的数据（如操作系统文件和数据库日志）所在的磁盘上。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

fed_noauth － 绕过联合认证

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可联机配置

传播类　　立即

缺省值 [范围]　　No [Yes; No]

当 *fed_noauth* 设置为 *yes*，*authentication* 设置为 *server* 或 *server_encrypt* 时，且 *federated* 设置为 *yes*，则将忽略实例上的认证。假定认证将在数据源上进行。在将 *fed_noauth* 设置为 *yes* 时将出现警告。认证既不是在客户机上也不是在 DB2 上完成的。所有知道 SYSADM 认证名的用户都可对联合服务器采用 SYSADM 权限。

相关参考:

- 第 427 页的『authentication － 认证类型』
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 第 422 页的『federated － 联合数据库系统支持』

| ## group_plugin － 组插件

| **配置类型**　　数据库管理器

| **适用于**

| - 带有本地和远程客户机的数据库服务器
| - 客户机
| - 带有本地客户机的数据库服务器
| - 带有本地和远程客户机的分区数据库服务器

| **参数类型**　　可配置

| **缺省值 [范围]**　　Null（任何有效字符串）

此参数指定组插件库的名称。缺省情况下，此值是 NULL，DB2 使用操作系统组查找。该插件将用于所有组查找。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

local_gssplugin － 用于本地实例级别授权的 GSS API 插件

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] Null（任何有效字符串）

此参数指定当 *authentication* 数据库管理器配置参数的值设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时要用于实例级别本地授权的缺省 GSS API 插件库的名称。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

srvcon_auth － 服务器上的入局连接的认证类型

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 [范围] Null [CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT]

此参数指定当处理服务器上的入局连接时如何进行用户认证以及在何处进行用户认证；它用来重设当前认证类型。如果未指定值，则 DB2 使用 *authentication* 数据库管理器配置参数的值。

有关每种认证类型的描述，请参阅第 427 页的『authentication － 认证类型』。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

srvcon_gssplugin_list － 用于服务器上的入局连接的 GSS API 插件的列表

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可配置

缺省值 [范围]　　Null（任何有效字符串）

此参数指定受数据库服务器支持的 GSS API 插件库。缺省情况下，该值为 NULL。如果认证类型为 GSSPLUGIN 且此参数为 NULL，则会返回错误。如果认证类型为 KERBEROS 且此参数为 NULL，则使用 DB2 提供的 Kerberos 模块或库。如果使用另一种认证类型，则不使用此参数。

当认证类型是 KERBEROS 且此参数的值不为 NULL，则列表必须只包含一个 Kerberos 插件并且将该插件用于认证（忽略列表中的所有其它 GSS 插件）。如果存在多个 Kerberos 插件，则会返回一条错误。

必须用逗号（,）将每个 GSS API 插件名称隔开，并且逗号前后不能有空格。应该按首选项的顺序来列示插件名称。当将 *srvcon_auth* 参数指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时，或者当未指定 *srvcon_auth* 且将认证类型指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时，此参数将处理服务器上的入局连接。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

srvcon_pw_plugin － 服务器上的入局连接的用户标识密码插件

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
Null（任何有效字符串）

此参数指定要用于服务器端认证的缺省用户标识密码插件库的名称。缺省情况下，该值为 NULL 并且使用 DB2 提供的用户标识密码插件库。

当将 *srvcon_auth* 参数指定为 SERVER 或 SERVER_ENCRYPT 时，或者当未指定 *srvcon_auth* 且将 *authentication* 指定为 CLIENT、SERVER 或 SERVER_ENCRYPT 时，该参数将处理入局连接。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

srv_plugin_mode － 服务器插件方式

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
UNFENCED

此参数指定是以设防方式还是以不设防方式运行插件。不设防方式是唯一受支持的方式。

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

| • 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

sysadm_group － 系统管理权限组名

配置类型
数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值
空

系统管理（SYSADM）权限是数据库管理器中最高级别的权限，它控制所有数据库对象。此参数定义具有数据库管理器实例的 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 98 操作系统中，SYSADM 组必须为 NULL。

  当使用系统安全性时，对于 Windows 98 客户机，此参数必须是“NULL”，因为 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 组缺省为实例所有者的主组。

  如果该值不为“NULL”，则 SYSADM 组可为任何有效的 UNIX 组名。

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。

相关参考:

- 第 436 页的『sysctrl_group － 系统控制权限组名』
- 第 436 页的『sysmaint_group － 系统维护权限组名』
- 『GET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『RESET DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』(*Command Reference*)

sysctrl_group － 系统控制权限组名

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 空

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 98 客户机，在使用系统安全性时（即认证是 CLIENT、SERVER、DCS 或任何其它有效认证），此参数必须为 NULL。这是因为 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。

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。

相关参考：

- 第 435 页的『sysadm_group － 系统管理权限组名』
- 第 436 页的『sysmaint_group － 系统维护权限组名』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

sysmaint_group － 系统维护权限组名

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 空

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 98 客户机，在使用系统安全性时（即认证是 CLIENT、SERVER、DCS 或任何其它有效认证），此参数必须为 NULL。这是因为 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。

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。

相关参考:

- 第 435 页的『sysadm_group － 系统管理权限组名』
- 第 436 页的『sysctrl_group － 系统控制权限组名』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

sysmon_group － 系统监视器权限组名

配置类型 数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型 可配置

缺省值 空

此参数定义具有系统监视（SYSMON）权限的组名。在实例级别具有 SYSMON 权限的用户能够为数据库管理器实例或它的数据库生成数据库系统监视器快照。SYSMON 权限能够使用下列命令:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

具有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户自动具有生成数据库系统监视器快照和使用这些命令的能力。

注意：对于 Windows 98 客户机，在使用系统安全性时（即认证是 CLIENT、SERVER、DCS 或任何其它有效认证），此参数必须为 NULL。这是因为 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 SYSMON 组的成员。指定组名时，所有用户都不能是它的成员。

要将该参数复原为它的缺省值（NULL），使用 UPDATE DBM CFG USING SYSMON_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。

相关参考：

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

trust_allclnts – 信赖所有客户机

配置类型　　数据库管理器

适用于

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型　　可配置

缺省值 [范围]　　YES [NO, YES, DRDAONLY]

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* 用来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”，所有客户机都作为可信的客户机对待。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”，则不可信客户机在连接到服务器时必须提供用户标识和密码组合。不可信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”，这将防止认证除来自 DB2 OS/390 和 z/OS 版、DB2 VM 和 VSE 版以及 DB2 OS/400 版的客户机之外的所有客户机。只有这些客户机可信赖，才能执行客户机端的认证。所有其它客户机必须提供用户标识和密码，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”，如果未提供密码则在客户机认证，如果提供了密码则在服务器认证。

相关参考：

- 第 427 页的『authentication － 认证类型』
- 第 439 页的『trust_clntauth － 可信的客户机认证』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

trust_clntauth － 可信的客户机认证

配置类型
数据库管理器

适用于
- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

参数类型
可配置

缺省值 [范围]
CLIENT [CLIENT, SERVER]

此参数指定，当客户机提供用于连接的用户标识和密码组合时，是在服务器还是在客户机中认证可信的客户机。仅当将 *authentication* 参数设置为 CLIENT 时，此参数（和 *trust_allclnts*）才是活动的。如果不提供用户标识和密码，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），则无须提供用户标识和密码组合，可信客户机就能连接，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标识和密码。

CLIENT 的数值是 0。SERVER 的数值是 1。

相关参考:
- 第 427 页的『authentication － 认证类型』
- 第 438 页的『trust_allclnts － 信赖所有客户机』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

use_sna_auth － 使用 SNA 认证

配置类型
数据库管理器

适用于
- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

- 带有本地和远程客户机的分区数据库服务器

参数类型	可联机配置
传播类	立即
缺省值 [范围]	No [Yes; No]

在将 *use_sna_auth* 设置为 *yes* 并将 *authentication* 设置为 *server* 时，与使用带有安全性类型 SAME 或 PROGRAM 的 SNA 协议的服务器的入站连接仅在 SNA 层认证而不是由 DB2 认证。

相关参考:

- 第 427 页的『authentication － 认证类型』
- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『RESET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

DB2 管理服务器

下列参数与 DB2 管理服务器相关:

- 『authentication － 认证类型 DAS』
- 第 441 页的『contact_host － 联系人列表的位置』
- 第 441 页的『das_codepage － DAS 代码页』
- 第 442 页的『das_territory － DAS 地域』
- 第 442 页的『dasadm_group － DAS 管理权限组名』
- 第 443 页的『db2system － DB2 服务器系统的名称』
- 第 443 页的『discover － DAS 发现方式』
- 第 444 页的『exec_exp_task － 执行已到期的任务』
- 第 445 页的『jdk_64_path － 64 位 Software Developer's Kit for Java 安装路径 DAS』
- 第 445 页的『jdk_path － Software Developer's Kit for Java 安装路径 DAS』
- 第 446 页的『sched_enable － 调度程序方式』
- 第 446 页的『sched_userid － 调度程序用户标识』
- 第 447 页的『smtp_server － SMTP 服务器』
- 第 447 页的『toolscat_db － 工具目录数据库』
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 448 页的『toolscat_schema － 工具目录数据库模式』

authentication － 认证类型 DAS

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置

缺省值 [范围]	SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT]

此参数确定如何进行以及在何处进行用户认证。

如果认证是 SERVER_ENCRYPT，则用户标识和密码从客户机发送到服务器，所以认证会在服务器上进行。通过网络发送的密码是加密的。

值 KERBEROS_ENCRYPT 意味着认证在 Kerberos 服务器上通过对认证使用 Kerberos 安全协议来执行。

注： 仅在运行 Windows 2000 的服务器上支持 KERBEROS_ENCRYPT 认证类型。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

contact_host － 联系人列表的位置

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（任何有效版本 8 DB2 管理服务器 TCP/IP 主机名）

此参数指定存储由“调度程序”和“健康监视器”用于通知的联系人信息的位置。该位置定义为 DB2 管理服务器的 TCP/IP 主机名。允许 *contact_host* 位于远程 DAS 上将支持在多个 DB2 管理服务器之间共享联系人列表。如果未指定 *contact_host*，则 DAS 假定联系人信息是本地的。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

das_codepage － DAS 代码页

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即

缺省值 [范围]	Null（任何有效的 DB2 代码页）

此参数指示由 DB2 管理服务器使用的代码页。如果参数为 null，则使用系统的缺省代码页。此参数应与本地 DB2 实例的语言环境兼容。否则，DB2 管理服务器不能与 DB2 实例通信。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 442 页的『das_territory － DAS 地域』

das_territory － DAS 地域

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（任何有效的 DB2 地域）

此参数显示由 DB2 管理服务器使用的地域。如果该参数为 null，则使用系统的缺省地域。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 441 页的『das_codepage － DAS 代码页』

dasadm_group － DAS 管理权限组名

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	Null（任何有效的组名）

“DAS 管理”（DASADM）权限是 DAS 内的最高级别的权限。此参数对 DAS 定义带有 DASADM 权限的组名。

DASADM 权限由在特定操作环境中使用的安全设施确定。

- 对于 Windows NT 和 Windows 2000 操作系统，此参数可以设置为具有 8 个或更少字符的名称的任何本地组，并且是在 Windows NT 和 Windows 2000 安全数据库中定义的。如果为此参数指定了“NULL”，则“管理员”组的所有成员都有 DASADM 权限。
- 对于基于 UNIX 的系统，如果将此参数的值指定为“NULL”，则 DASADM 组缺省为实例所有者的主组。

  如果值不是“NULL”，则 DASADM 组可为任何有效的 UNIX 组名。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

db2system – DB2 服务器系统的名称

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
缺省值 [范围]	TCP/IP 主机名（任何有效的系统名称）

此参数指定由您的用户和数据库管理员使用以标识 DB2 服务器系统的名称。应尽可能使此名称在网络中是唯一的。

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“配置助手”的“搜索网络”功能时，“DB2 发现”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* 的值设置如下:

- 在 Windows 上，设置程序将其设置为对 Windows 系统指定的计算机名称。
- 在 UNIX 系统上，将它设置成 UNIX 系统的 TCP/IP 主机名。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 443 页的『discover – DAS 发现方式』

discover – DAS 发现方式

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即

缺省值 [范围]	SEARCH [DISABLE; KNOWN; SEARCH]

从管理服务器的角度来看，此配置参数确定“DB2 管理服务器”启动时启动的发现方式的类型。

- 如果 discover = SEARCH，则管理服务器处理来自客户机的 SEARCH 发现请求。SEARCH 提供由 KNOWN 发现提供的功能的超集。当 discover = SEARCH，管理服务器将处理来自客户机的 SEARCH 和 KNOWN 发现请求。
- 如果 discover = KNOWN，则管理服务器仅处理来自客户机的 KNOWN 发现请求。
- 如果 discover = DISABLE，则管理服务器不会处理任何类型的发现请求。此服务器系统的信息本质上已从客户机隐藏。

缺省发现方式为 SEARCH。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 443 页的『db2system － DB2 服务器系统的名称』

exec_exp_task － 执行已到期的任务

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	No [Yes; No]

此参数指定“调度程序”是否将执行已在过去调度但是尚未执行的任务。“调度程序”在启动时，仅检测到期任务。

例如，如果有一个作业调度为在每个星期六运行，并且“调度程序”在星期五关闭并在星期一重新启动，则对星期六调度的作业现在是在过去调度的作业。如果将 *exec_exp_task* 设置为 Yes，则当“调度程序”重新启动时，将运行星期六作业。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 446 页的『sched_enable － 调度程序方式』
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 447 页的『toolscat_db － 工具目录数据库』
- 第 448 页的『toolscat_schema － 工具目录数据库模式』
- 第 447 页的『smtp_server － SMTP 服务器』

- 第 446 页的『sched_userid － 调度程序用户标识』

jdk_64_path － 64 位 Software Developer's Kit for Java 安装路径 DAS

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（任何有效路径）

此参数指定要用于运行 DB2 管理服务器函数的 64 位 Java 软件开发者工具箱（SDK）的安装目录。

注: 此参数与 *jdk_path* 配置参数不同，后者指定 32 位的 Java SDK。

Java 解释器使用的环境变量是根据此参数的值计算的。此参数仅在那些既支持 32 位实例又支持 64 位实例的平台上使用。那些平台也称为 64 位混合平台，它们包括 AIX、HP-UX 和 Solaris Operating Environment。在所有其它平台上，只使用 *jdk_path*。

因为此参数没有缺省值，所以安装 Java SDK 时应指定值。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

jdk_path － Software Developer's Kit for Java 安装路径 DAS

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	缺省 Java 安装路径（任何有效路径）

此参数指定要用于运行 DB2 管理服务器函数的 Java 软件开发者工具箱（SDK）的安装目录。 Java 解释器使用的环境变量是根据此参数的值计算的。

在 Windows 操作系统上，在 DB2 安装期间，Java 文件（如果需要的话）放置在 sqllib 目录（在 java\jdk 中）下面。然后将 *jdk_path* 配置参数设置为 sqllib\java\jdk。DB2 永远不会实际在 Windows 平台上安装 Java；只是将文件放置在 sqllib 目录下面，并且不管是否已安装 Java 都会这样做。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

sched_enable － 调度程序方式

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	Off [On; Off]

此参数指示“调度程序”是否由管理服务器启动。“调度程序”允许工具（例如“任务中心”）来在管理服务器上调度和执行任务。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 447 页的『toolscat_db － 工具目录数据库』
- 第 448 页的『toolscat_schema － 工具目录数据库模式』
- 第 447 页的『smtp_server － SMTP 服务器』
- 第 444 页的『exec_exp_task － 执行已到期的任务』
- 第 446 页的『sched_userid － 调度程序用户标识』

sched_userid － 调度程序用户标识

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	参考
缺省值 [范围]	Null（任何有效的用户标识）

此参数指定由“调度程序”使用以与工具目录数据库连接的用户标识。仅当工具目录数据库对于 DB2 管理服务器是远程时，此参数才是切题的。

由“调度程序”使用以与远程工具目录数据库连接的用户标识和密码是使用 **db2admin** 命令指定的。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 446 页的『sched_enable － 调度程序方式』
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 447 页的『toolscat_db － 工具目录数据库』

- 第 448 页的『toolscat_schema － 工具目录数据库模式』
- 第 447 页的『smtp_server － SMTP 服务器』
- 第 444 页的『exec_exp_task － 执行已到期的任务』

smtp_server － SMTP 服务器

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可联机配置
传播类	立即
缺省值 [范围]	Null（任何有效的 SMTP 服务器 TCP/IP 主机名）

此参数由“调度程序”和“运行状况监视器”使用。

当“调度程序”打开时，此参数标识“调度程序”将用来发送电子邮件和寻呼机通知的 SMTP 服务器。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 446 页的『sched_enable － 调度程序方式』
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 447 页的『toolscat_db － 工具目录数据库』
- 第 448 页的『toolscat_schema － 工具目录数据库模式』
- 第 444 页的『exec_exp_task － 执行已到期的任务』
- 第 446 页的『sched_userid － 调度程序用户标识』

toolscat_db － 工具目录数据库

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	Null（任何有效的数据库别名）

此参数指示由“调度程序”使用的工具目录数据库。此数据库必须在由 *toolscat_inst* 指定的实例的数据库目录中。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）

- 『UPDATE ADMIN CONFIGURATION Command』(*Command Reference*)
- 第 446 页的『sched_enable – 调度程序方式』
- 第 448 页的『toolscat_inst – 工具目录数据库实例』
- 第 448 页的『toolscat_schema – 工具目录数据库模式』
- 第 447 页的『smtp_server – SMTP 服务器』
- 第 444 页的『exec_exp_task – 执行已到期的任务』
- 第 446 页的『sched_userid – 调度程序用户标识』

toolscat_inst – 工具目录数据库实例

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	Null(任何有效的实例)

此参数指示“调度程序”以及 toolscat_db 和 toolscat_schema 使用的用于标识工具目录数据库的实例名。工具目录数据库包含“任务中心”和“控制中心”创建的任务信息。工具目录数据库必须列示在此配置参数指定的实例的数据库目录中。数据库可以是本地的或远程的。如果工具目录数据库是本地的，则必须为 TCP/IP 配置实例。如果数据库是远程的，则数据库目录中编目的节点必须是 TCP/IP 节点。

此参数只能从版本 8 命令行处理器(CLP)更新。

相关任务:

- 『工具目录数据库和 DAS 调度程序安装和配置』(*《管理指南: 实现》*)

相关参考:

- 『GET ADMIN CONFIGURATION Command』(*Command Reference*)
- 『RESET ADMIN CONFIGURATION Command』(*Command Reference*)
- 『UPDATE ADMIN CONFIGURATION Command』(*Command Reference*)
- 第 446 页的『sched_enable – 调度程序方式』
- 第 447 页的『toolscat_db – 工具目录数据库』
- 第 448 页的『toolscat_schema – 工具目录数据库模式』
- 第 447 页的『smtp_server – SMTP 服务器』
- 第 444 页的『exec_exp_task – 执行已到期的任务』
- 第 446 页的『sched_userid – 调度程序用户标识』

toolscat_schema – 工具目录数据库模式

配置类型	DB2 管理服务器
适用于	DB2 管理服务器
参数类型	可配置
缺省值 [范围]	Null(任何有效的模式)

此参数指示由“调度程序”使用的工具目录数据库的模式。此模式用来唯一标识在数据库内的一组工具目录表和视图。

此参数只能从版本 8 命令行处理器（CLP）更新。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『RESET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 第 446 页的『sched_enable － 调度程序方式』
- 第 448 页的『toolscat_inst － 工具目录数据库实例』
- 第 447 页的『toolscat_db － 工具目录数据库』
- 第 447 页的『smtp_server － SMTP 服务器』
- 第 444 页的『exec_exp_task － 执行已到期的任务』
- 第 446 页的『sched_userid － 调度程序用户标识』

第 4 部分 附录

附录 A. DB2 注册表变量和环境变量

本章描述如何使用注册表和环境变量，并为每个类别的变量列表提供它们的语法和用法的说明。

DB2 注册表和环境变量

本节列示 DB2® 注册表变量和环境变量，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描述。某些变量不能在环境中应用。

要查看所有受支持的注册表变量的列表，执行下列命令:

```
db2set -lr
```

要更改当前实例或缺省实例中变量的值，执行下列命令:

```
db2set registry_variable_name=new_value
```

DB2 环境变量 DB2INSTANCE、DB2NODE、DB2PATH 和 DB2INSTPROF 是否存储在 DB2 概要文件注册表中取决于操作系统。要更新这些环境变量，使用 `set` 命令。这些更改在下次重新引导系统之前一直有效。在 UNIX® 平台上，可以使用 `export` 命令代替 `set` 命令。

必须在执行 DB2START 命令之前设置已更改注册表变量的值。

注: 如果注册表变量要求“布尔”值作为自变量，则值 YES、1 和 ON 都是等价的，值 NO、0 和 OFF 也是等价的。对于任何变量，可以指定任何适当的等价值。

相关概念:

- 『环境变量和概要文件注册表』（*《管理指南：实现》*）

相关任务:

- 『在 LDAP 环境中的用户级别设置 DB2 注册表变量』（*《管理指南：实现》*）

相关参考:

- 第 454 页的『一般注册表变量』
- 第 456 页的『系统环境变量』
- 第 459 页的『通信变量』
- 第 463 页的『命令行变量』
- 第 464 页的『MPP 配置变量』
- 第 465 页的『SQL 编译器变量』
- 第 469 页的『性能变量』
- 第 479 页的『Data links 变量』
- 第 481 页的『其它变量』

按类别的注册表和环境变量

以下各节根据数据库管理器的特征或它们控制的数据库行为列示注册表和环境变量。

一般注册表变量

表 46. 一般注册表变量

变量名	操作系统	值
描述		
DB2ACCOUNT	所有	缺省值 = null
发送至远程主机的记帐字符串。有关详细信息，参阅*《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，并且 DB2CODEPAGE 变量用于声明要使用的代码页。有关双向支持的其它信息，参阅“本地语言支持”附录。		
DB2CODEPAGE	所有	缺省值：从语言标识中派生，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 文档中显式声明或 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能产生意外的的结果。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。		
DB2_COLLECT_TS_REC_INFO	所有	缺省值 = OFF 值：YES 或 NO
此变量指定在前滚表空间时，DB2 是否将处理所有日志文件，无论日志文件是否包含会影响表空间的日志记录。要跳过已知不包含将影响表空间的任何日志记录的日志文件，应将此变量设置为“ON”。 在创建和使用日志文件之前，必须设置 DB2_COLLECT_TS_REC_INFO，以便收集跳过日志文件的所需要的信息。		
DB2CONSOLECP	Windows	缺省值 = NULL 值：所有有效的代码页值
指定用于显示 DB2 消息文本的代码页。当指定了此值时，它会覆盖操作系统代码页设置。		
DB2COUNTRY	Windows	缺省值 = null 值：所有有效数字的国家或地区、地域或区域代码
指定客户机应用程序的国家或地区、地域或区域代码。当指定了此值时，它会覆盖操作系统设置。		
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。		
DB2DBMSADDR	Windows 32 位操作系统	对于 Windows NT，缺省值 = 0x20000000 值：0x20000000 到 0xB0000000，增量为 0x10000
以十六进制格式指定缺省数据库管理器的共享内存地址。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，以强制数据库管理器实例在另一个地址分配其共享内存。		
DB2DISCOVERYTIME	Windows 操作系统	缺省值 = 40 秒， 最小值 = 20 秒

表 46. 一般注册表变量 (续)

变量名	操作系统	值
描述		
指定 SEARCH 发现将在 DB2 系统中搜索的时间。		
DB2_FORCE_APP_ON_MAX_LOG	所有	缺省值：TRUE 值：TRUE 和 FALSE
指定当超过了 MAX_LOG 配置参数值时将发生的情况。如果设置为 TRUE，则会强制应用程序关闭数据库并回滚工作单元。 如果设置为 FALSE，则当前语句将失败。该应用程序仍然可以落实在工作单元中由先前语句完成的工作，它也可以回滚已完成的工作以撤销该工作单元。		
DB2GRAPHICUNICODESERVER	所有	缺省值 = OFF 值：ON 或 OFF
此注册表变量用来存放现有的为将图形数据插入 Unicode 数据库而编写的应用程序。仅需要对那些特别地发送 Unicode 格式（而不是客户机的代码页）的 sqldbchar（图形）数据的应用程序使用它。（sqldbchar 是 C 和 C++ 中受支持的 SQL 数据类型，可以包含单个双字节字符。）若设置为“ON”，则是通知数据库将接收的图形数据是 Unicode 格式，并且应用程序期望接收到 Unicode 格式的图形数据。		
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。参阅*《应用程序开发指南》*，以获得如何在不同的预编译语言中使用 DB2INCLUDE 的描述。		
DB2INSTDEF	Windows 操作系统	缺省值 = DB2
设置未定义 DB2INCLUDE 时要使用的值。		
DB2INSTOWNER	Windows NT	缺省值 = 空
第一次创建实例时在 DB2 概要文件注册表中创建的注册表变量。将此变量设置为实例拥有的机器的名称。		
DB2_LIC_STAT_SIZE	所有	缺省值 = null 范围：0 到 32 767
该注册表变量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。		
DB2LOCALE	所有	缺省值：NO 值：YES 或 NO
指定在调用 DB2 之后是否将进程的缺省值“C”语言环境复原为缺省“C”语言环境，以及在调用 DB2 函数之后是否将进程语言环境复原为原始“C”。如果原始语言环境不是“C”，则忽略此注册表变量。		
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 搜索发现。此变量指定一个客户机可接收的并发发现响应的数目。如果客户机接收到的并发响应多于此变量指定的数目，则多余的响应将被 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。		
DB2_OBJECT_TABLE_ENTRIES	所有	缺省值 = 0 值：0 至 50000

表 46. 一般注册表变量 (续)

变量名	操作系统	值
描述		
指定表空间中期望的对象数。如果知道将在 DMS 表空间中创建大量对象（例如，1000 或者更多个对象），则在创建该表空间之前，应该将此注册表变量设置为一个接近的数目。在创建表空间期间，这将为对象元数据保留连续的存储器。保留连续的存储器会降低联机备份将阻塞一些将更新元数据中的条目的操作（例如，CREATE INDEX 和 IMPORT REPLACE）的机率。它还将更容易调整表空间大小，原因是元数据将存储在表空间的开始部分。 如果表空间的初始大小不足以保留连续存储器，则将继续创建表空间而不保留附加空间。		
DB2OPTIONS	所有	缺省值 = null
设置命令行处理器选项。		
DB2TERRITORY	所有	缺省值：从语言标识中派生，由操作系统指定。
指定客户机应用程序的区域或地域代码，它影响日期和时间格式。		
DB2_VIEW_REOPT_VALUES	所有	缺省值 = NO 值：YES，NO
当说明重新优化的 SQL 语句时，此变量使所有用户都能够将该语句的经过高速缓存的值存储在 EXPLAIN_PREDICATE 表中。当此变量设置为 NO 时，只允许 DBADM 将这些值保存在 EXPLAIN_PREDICATE 表中。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

系统环境变量

表 47. 系统环境变量

变量名	操作系统	值
描述		
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 企业版机器上的 DB2 Connect 客户机在代理进程内运行。在代理进程内运行的一些优点是可监视本地客户机，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。
定义一个或多个 Windows NT 域。列表定义将对其进行认证的请求用户标识的域。只有属于这些域的用户会才会接受它们的连接或连接请求。 仅当在“数据库管理器”配置中设置了 CLIENT 认证时，此变量才有效，如果 Windows NT 域环境中要求从 Windows NT 桌面进行单次注册，则需要此变量。 该注册表变量应该仅在纯 Windows NT 域环境下与运行 DB2 通用数据库版本 7.1（或更高版本）的 DB2 服务器和客户机一起使用。		
DB2ENVLIST	UNIX	缺省值：null

表 47. 系统环境变量 (续)

变量名	操作系统	值
描述		
列示存储过程或用户定义的函数的特定变量名。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。如果必须将特定的环境变量传送至存储过程或用户定义的函数，则可列示 DB2ENVLIST 环境变量中的变量名。用一个或多个空格将每个变量名隔开。		
DB2INSTANCE	所有	在 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	Windows 操作系统	缺省值：null
用于在 Windows 操作系统上指定实例目录的位置（如果与 DB2PATH 不同）的环境变量。		
DB2LIBPATH	UNIX	缺省值：null
DB2 构造其自身的共享库路径。若要将 PATH 添加至引擎的库路径（例如，在 AIX 上，用户定义的函数在 LIBPATH 中需要特定的条目），则必须设置 DB2LIBPATH。 DB2LIBPATH 的实际值追加到 DB2 构造的共享库路径的末尾。		
DB2NODE	所有	缺省值：null 值：1 至 999
用于指定希望连接的“DB2 企业服务器版”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： • *[:*number-of-disks*] – 意味着每个表空间都将启用并行 I/O。可以在冒号后面提供值或符号来定义所有表空间的每个容器的缺省磁盘数。如果不指定磁盘数，则将使用缺省值 6（RAID-5 设备的值）。 • TablespaceID[:*number-of-disks*],... – 用逗号分隔的已定义的表空间的列表。要指定该表空间的每个容器的磁盘数，在每个表空间标识后面的冒号后面指定一个值。此值可以是数字值或者在下表中描述的其中一个符号。 • 上面两种值类型的组合，并将每个值用逗号隔开

表 47. 系统环境变量 (续)

变量名	操作系统	值
描述		
此注册表变量用来更改 DB2 计算表空间的 I/O 并行性的方式。当启用了 I/O 并行性时（要么通过使用多个容器来隐式启用，要么通过设置 DB2_PARALLEL_IO 来显式启用），通过发出正确数目的预取请求来实现此目标。每个预取请求都是对页的扩展数据块的请求。 如果未设置此注册表变量，则任何表空间的并行度都将是表空间的容器数。例如，如果 DB2_PARALLEL_IO 设置为 NULL，并且表空间具有四个容器，则将发出四个按扩展数据块大小计算的预取请求。 如果设置了此注册表变量，则表空间的并行度将是此表空间的预取大小与扩展数据块大小之间的比率。例如，如果对预取大小为 160 而扩展数据块大小为 32 页的表空间设置了 DB2_PARALLEL_IO，则将发出五个按扩展数据块大小计算的预取请求。通配符“*”可以用来告诉 DB2 以此方式计算所有表空间的 I/O 并行性。 在支持对每个 DB2 容器下面的物理主轴（例如，RAID 设备）进行条带分割的 I/O 子系统中，当选择表空间的预取大小时，应该考虑每个 DB2 容器中的磁盘数。应该根据以下等式来计算预取大小: 预取大小 =（容器数）*（每个容器的磁盘数）* 扩展数据块大小 如果表空间的预取大小为 AUTOMATIC，则 DB2 会自动使用上面的等式计算表空间的预取大小。 可以使用 DB2_PARALLEL_IO 注册表变量来告诉 DB2 每个容器的磁盘数。例如，如果 DB2_PARALLEL_IO=″1:4″ 并且第一个表空间具有 3 个容器，扩展数据块大小为 32，预取大小为 AUTOMATIC，则预取大小将为 3 * 4 * 32 = 384 页。此表空间的 I/O 并行性将为 384 / 32 = 12。如果表空间的预取大小不是 AUTOMATIC，则将不使用有关每个容器的磁盘数的信息。 缺省情况下，假定在 DB2_PARALLEL_IO 下面指定的任何表空间的每个容器中的磁盘数为 6，除非在注册表变量中另有指定。例如，如果 DB2_PARALLEL_IO=*,1:3，则所有表空间都将使用 6 作为每个容器中的磁盘数，但是第一个表空间除外（它将使用 3 作为每个容器中的磁盘数）。		
DB2PATH	Windows 操作系统	缺省值：（随操作系统变化）
用于指定在 Windows 32 位操作系统上安装了该产品的目录的环境变量。		
DB2PROCESSORS	Windows 操作系统	缺省值：null 值：–n-1（其中 n= 处理器的数目）
设置特定 db2syscs 进程的进程亲缘关系掩码。在运行多个逻辑节点的环境中，这个变量用来将逻辑节点与处理器或一组处理器相关联。 当指定了此变量，则 DB2 发出 SetProcessAffinityMask() API。如果未指定此变量，则 db2syscs 进程与机器上的所有处理器相关联。		
DB2_USE_PAGE_CONTAINER_TAG	所有	缺省值：null 值：ON，null
缺省情况下，DB2 将容器标记存储在每个 DMS 容器的第一个扩展数据块中，而无论它是文件还是设备。容器标记是容器的元数据。在 DB2 版本 8.1 之前，容器标记存储在单个页中，因此，它在容器中需要较少的空间。要继续将容器标记存储在单个页中，将 DB2_USE_PAGE_CONTAINER_TAG 设置为 ON。 但是，如果当对容器使用 RAID 设备时，将注册变量设置为 ON，则会降低 I／O 性能。因为对于您使用等于 RAID 分割区大小或其倍数的扩展数据块大小创建表空间的 RAID 设备，将 DB2_USE_PAGE_CONTAINER_TAG 设置为 ON 导致扩展数据块不与 RAID 分割区排列在一起。因此，I/O 请求可能需要存取比将优化的物理磁盘更多的磁盘。强烈建议用户不要启用此注册表变量。 要激活对此注册变量的更改，发出 DB2STOP 命令，并输入 DB2START 命令。		

表 47. 系统环境变量 (续)

变量名	操作系统	值
描述		
DB2_CLPHISTSIZE	所有	缺省值：20 **注**：在安装期间，不会将此注册表变量设置为缺省值。相反，如果未设置该注册表变量或者将它设置为有效范围之外的值，则使用此变量的代码使用缺省值 20。值：1 至 500（包括 1 和 500）
此变量确定 CLP 交互式会话期间存储在命令历史中的命令数。由于命令历史保留在内存中，所以如果此变量的值非常高的话可能会导致性能影响，这取决于在会话中运行的命令的数目和长度。		
DB2_CLP_EDITOR	所有	缺省值： Windows 平台：’Notepad’ UNIX：’vi’ **注**：在安装期间，不会将此注册表变量设置为缺省值。相反，如果不设置该注册表变量，则使用此变量的代码会使用缺省值。值：位于操作系统路径中的任何有效编辑器
此变量确定执行 **EDIT** 命令时要使用的编辑器。从 CLP 交互式会话中，**EDIT** 命令启动使用用户指定命令预先装入的编辑器，于是就可编辑和运行该编辑器。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

通信变量

表 48. 通信变量

变量名	操作系统	值
描述		
DB2CHECKCLIENTINTERVAL	所有，仅适用于服务器	缺省值 = 50 值：一个大于零的数值。
指定 APPC 和 TCP/IP 客户机连接验证的频率。允许提前检测客户机的终止，而不是等到查询完成之后。如果将此变量设置为 0，则不执行验证。 较低的值表示检查频率更高。作为一条准则，对于较低频率，使用 100；对于中等频率，使用 50；对于较高频率，使用 10。当执行数据库请求时更频繁地检查客户机状态会延长完成查询所花的时间。如果 DB2 工作负载很重（即，它参与许多内部请求），则将 DB2CHECKCLIENTINTERVAL 设置为较低的值，较之于在工作负载较轻并且 DB2 大多数时间都处于等待状态的情况对性能的影响会更大。 在 DB2 UDB 版本 8.1.4 中，DB2CHECKCLIENTINTERVAL 的缺省值为 50。在版本 8.1.4 之前，缺省值为 0。		
DB2COMM	所有，仅适用于服务器	缺省值 = null 值：APPC、IPXSPX、NETBIOS、NPIPE 和 TCPIP 的任何组合
指定当启动数据库管理器时所启动的通信管理器。如果未设置，则不在服务器上启动任何 DB2 通信管理器。		

表 48. 通信变量 (续)

变量名	操作系统	值
描述		
DB2_FORCE_NLS_CACHE	AIX、HP-UX 和 Solaris Operating Environment	缺省值 = FALSE 值：TRUE 或 FALSE
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地域代码信息。由此，高速缓存的信息就可用于请求此信息的任何其它线程。在某些情况中，这样可消去锁定争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语言环境设置，则不应该使用此设置。在此情况中，可能不需要这样，原因是多线程应用程序一般不会更改它们的语言环境设置，因为这样做不是“线程安全”的。		
DB2JD_PORT_NUMBER	所有	缺省值 = 6789 值：1 - 65535
可以设置此注册表变量来覆盖 db2jd 的缺省端口号。如果设置了此注册表变量，则当发出不带任何参数的 db2jstrt 时，db2jd 将尝试将它用作 db2jd 的侦听端口。如果未设置此注册表变量并且没有提供端口参数，则 db2jd 将在缺省端口 6789 上启动。		
DB2NBADAPTERS	Windows	缺省值 = 0 范围：0-15， 应将多个值用逗号隔开
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	限于 Windows 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进程／会话终止时，释放内存和剩余的其它系统资源。		
DB2NBINTRLISTENS	限于 Windows 服务器	缺省值 = 1 值：1-10 多个值应用逗号隔开
指定以异步方式发出以便于远程客户机中断的 NetBIOS 侦听发送命令（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它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 设置为较低的值将节约服务器上的 NetBIOS 会话和 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，以便对正在中断的客户机进行响应。 **注**：指定的值对位置敏感；它们与 DB2NBADAPTERS 的对应值位置相关。		
DB2NBRECVBUFFSIZE	限于 Windows 服务器	缺省值 = 4096 个字节 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。		
DB2NBBRECVNCBS	限于 Windows 服务器	缺省值 = 10 范围：1-99

表 48. 通信变量 (续)

变量名	操作系统	值
描述		
指定服务器在操作期间发出和维护的 NetBIOS “receive_any” 命令（NCB）的数目。根据您的服务器连接的远程客户机数来调整此值。较低的值可以节约服务器资源。 **注**：使用中的每个适配器可以有自己的唯一接收 NCB 值，该值由 DB2NBBRECVNCBS 指定。指定的值对位置敏感；它们与 DB2NBADAPTERS 的对应值位置相关。		
DB2NBRESOURCES	限于 Windows 服务器	缺省值 = null
指定在多上下文环境中为 DB2 使用分配的 NetBIOS 资源量。此变量仅限用于多上下文客户机操作。		
DB2NBSENDNCBS	限于 Windows 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 设置为较低的值将节约服务器资源。但是，您可能需要将它设置为较高的值，以防止当所有其它发送命令在使用时，服务器等待向远程客户机发送。		
DB2NBSESSIONS	限于 Windows 服务器	缺省值 = null 范围：5-254
指定 DB2 应请求的、要保留给 DB2 使用的会话数。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会话。 **注**：指定的值对位置敏感；它们与 DB2NBADAPTERS 的对应值位置相关。		
DB2NBXTRANCBS	限于 Windows 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会话。		
DB2RETRY	Windows	缺省值 = 0 范围：0-20 000
DB2 尝试重新启动 APPC 侦听器的次数。如果位于服务器／网关的 SNA 子系统停机，则可使用此概要文件变量与 DB2RETRYTIME 一起来自动重新启动 APPC 侦听器而不中断使用其它协议的客户机通信。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	Windows	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 侦听器进行连续重试的间隔分钟数。如果服务器／网关上的 SNA 子系统停机，则可使用此概要文件变量与 DB2RETRY 一起来自动重新启动 APPC 侦听器而不中断使用其它协议的客户机通信。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	Windows、AIX 和 Solaris Operating Environment	缺省值 = null
用来解决由下列情况产生的问题： • 多个实例在同一机器上运行 • 在同一机器上运行的版本 6 或 版本 7 实例尝试注册同一个 TP 名称。 当调用 **db2start** 命令时，指定的实例将对下列 TP 名称启动 APPC 侦听器： • DB2DRDA • x’07’6DB		

表 48. 通信变量 (续)

变量名	操作系统	值
描述		
DB2SORCVBUF	所有	缺省值 = 65536
指定 Windows 操作系统上的 TCP/IP 接收缓冲区的值。		
DB2SOSNDBUF	所有	缺省值 = 65536
指定在 Windows NT 操作系统上的 TCP/IP 发送缓冲区的值。		
DB2SYSPLEX_SERVER	Windows NT 和 UNIX	缺省值 = null
指定在与 DB2 OS/390 或 z/OS 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 数据库目录条目是如何指定的。有关更多信息，请参阅命令行处理器 **CATALOG DCS DATABASE** 命令。		
DB2TCPCONNMGRS	所有	在串行机器上，缺省值 = 1；在对称多处理器上，缺省值为处理器数平方根进位舍入到最大连接管理器数（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器。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 连接管理器，最多创建 8 个。如果指定小于 1 的值，则 DB2TCPCONNMGRS 设置为值 1，并记录一条警告，指示值超出范围。如果指定大于 8 的值，则 DB2TCPCONNMGRS 设置为值 8，并记录一条警告，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 个连接管理器时，如果同时接收多个客户机连接，则连接处理能力应能得到提高。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 连接管理器进程（在 UNIX 上）或线程（在 Windows 操作系统上）。附加的进程或线程需要附加的存储器。 **注**：将连接管理器数设置为 1 会导致在具有许多用户的系统中的远程连接的性能降低、频繁连接和／或断开连接。		
DB2_VI_ENABLE	Windows NT	缺省值 = OFF 值：ON 或 OFF
指定是否使用“虚拟接口（VI）体系结构”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。为了成功地装入库，在此注册表变量中使用的库名必须在 PATH 用户环境变量中。当前受支持的实现全部使用相同的库名。		
DB2_VI_DEVICE	Windows NT	缺省值 = null 值：`nic0` 或 `VINIC`
指定设备的符号名称或与“网络接口卡”相关的“虚拟接口供应商实例”。“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 NIC；同一个物理机器上的多个逻辑节点将共享同一个 NIC。符号设备名“VINIC”必须是大写的，并且仅可以与“Synfinity 互连”一起使用。所有其它当前受支持的实现都将“nic0”用作符号设备名。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

命令行变量

表 49. 命令行变量

变量名	操作系统	值
描述		
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活动并建立与它的连接之前休眠的时间量。		
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活动的次数。这个变量与 DB2BQTIME 共同起作用。		
DB2_CLPPROMPT	所有	缺省值 = 无 如果未定义此变量，则将把“db2 =>”用作缺省 CLP 交互式提示符。 可能值：长度小于 100 且包含零个或多个下列记号的任何文本字符串：%i、%d、%ia、%da 或 %n。除非用户确实希望更改缺省 CLP 交互式提示符（db2 =>），否则他们不需要设置此变量。
此注册表变量允许用户定义要在“命令行处理器”（CLP）交互方式中使用的提示符。可将此变量设置为任何长度小于 100 个字符并且包含零个或多个可选记号 %i、%d、%ia、%da 或 %n 的文本字符串。当以 CLP 交互方式运行时，要使用的提示符的构造方式如下：通过获取 DB2_CLPPROMPT 注册表变量中指定的文本字符串并将 %i、%d、%ia、%da 或 %n 记号的所有出现分别替换为当前所连接的实例的本地别名、当前数据库连接的本地别名、当前所连接的实例的授权标识、当前数据库连接的授权标识以及换行（即，回车符）。 **注:** 1. 如果在 CLP 交互方式下更改 DB2_CLPPROMPT 注册表变量，则在关闭并重新打开 CLP 交互方式之前，DB2_CLPPROMPT 的新值不会生效。 2. 如果不存在实例连接，则将 %ia 替换为空字符串，并将 %i 替换为 DB2INSTANCE 注册表变量的值。（仅在 Windows 平台上）如果未设置 DB2INSTANCE 变量，则将 %i 替换为 DB2INSTDEF 注册表变量的值。如果这些变量都没有设置，则将 %i 替换为空字符串。 3. 如果不存在数据库连接，则将 %da 替换为空字符串，并将 %d 替换为 DB2DBDFT 注册表变量的值。如果未设置 DB2DBDFT 变量，则将 %d 替换为空字符串。 4. 交互式输入提示符将总是以大写形式显示授权标识、数据库名称和实例名的值。		
DB2IQTIME	所有	缺省值 = 5 秒 最小值：1 秒
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量。		
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量。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

MPP 配置变量

表 50. MPP 配置变量

<table>
<tr><th>变量名</th><th>操作系统</th><th>值</th></tr>
<tr><td>DB2ATLD_PWFILE</td><td>AIX、Solaris Operating Environment 和 Windows NT 上的 DB2 UDB ESE</td><td>缺省值 = 空

值：文件路径表达式</td></tr>
<tr><td colspan="3">指定一个文件的路径，该文件包含在“自动装入”认证期间使用的密码。如果未设置，则“自动装入”从其配置文件中抽取密码，或以交互式方式提示您输入密码。使用此变量将解决密码安全性问题，并允许“自动装入”配置信息和认证信息分开。

不再需要此注册表变量，但为了保证向下兼容而保留它。</td></tr>
<tr><td>DB2CHGPWD_EEE</td><td>AIX 和 Windows NT 上的 DB2 UDB ESE</td><td>缺省值 = 空

值：YES 或 NO</td></tr>
<tr><td colspan="3">指定是否允许其它用户更改 AIX 或 Windows NT ESE 系统上的密码。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密码。如果没有集中维护，在所有分区或节点之间密码可能会不一致。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密码。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。

仅当使用旧 db2atld 实用程序而不是新 LOAD 实用程序时，才需要此变量。</td></tr>
<tr><td>DB2_FORCE_FCM_BP</td><td>AIX</td><td>缺省值 = No

值：Yes 或 No</td></tr>
<tr><td colspan="3">此注册表变量适用于使用多逻辑分区的 DB2 UDB ESE AIX 版。当发出 DB2START 时，DB2 从数据库全局内存中或从单独的共享内存段（如果没有足够的全局内存可用）分配 FCM 缓冲区。所有 FCM 守护进程对相同物理机器上的实例使用这些缓冲区。分配的这种内存主要取决于要创建的 FCM 缓冲区数，该数由 fcm_num_buffers 数据库管理器配置参数指定。

如果 DB2_FORCE_FCM_BP 变量设置为 YES，总是在单独的内存段创建 FCM 缓冲区，以便同一个物理节点上不同逻辑分区的 FCM 守护进程之间的通信通过共享内存来进行。否则，同一个节点上的 FCM 守护进程将通过 UNIX 套接字进行通信。通过共享内存的通信更快，但其它方面可以使用的共享内存段会减少，特别是对于数据库缓冲池来说。因此启用 DB2_FORCE_FCM_BP 注册表变量会减小数据库缓冲池的最大大小。</td></tr>
<tr><td>DB2_NUM_FAILOVER_NODES</td><td>所有</td><td>缺省值：2

值：0 至逻辑节点数</td></tr>
<tr><td colspan="3">指定可以用作高可用性环境中的故障转移节点的节点数。借助高可用性，当某个节点失败时，该节点便可作为另一台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转移节点的 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。在 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，即使一个主机失败，失败主机的逻辑节点也可以在另一台主机上重新启动。</td></tr>
<tr><td>DB2_PARTITIONEDLOAD__DEFAULT</td><td>所有受支持的 ESE 平台</td><td>缺省值：YES;

值范围：YES / NO</td></tr>
<tr><td colspan="3">DB2_PARTITIONEDLOAD_DEFAULT 注册表变量允许用户在未指定 ESE 特定的“装入”选项时更改 ESE 环境中 Load 实用程序的缺省行为。缺省值为 YES，这指定在 ESE 环境中，如果未指定 ESE 特定的“装入”选项，则尝试在其上定义了目标表的所有分区上装入。

当值为 NO 时，仅在 Load 实用程序当前连接至的分区上尝试装入。</td></tr>
</table>

表 50. MPP 配置变量 (续)

变量名	操作系统	值
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。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

SQL 编译器变量

表 51. SQL 编译器变量

变量名	操作系统	值
描述		
DB2_ANTIJOIN	所有	在 ESE 环境中，缺省值 = NO 在非 EEE 环境中，缺省值 = YES 值：YES、NO 或 EXTEND
对于“DB2 通用数据库”ESE 环境：当指定了“YES”时，优化器搜索将“NOT EXISTS”子查询变换为可以由 DB2 更有效处理的反连接的机会。对于非 ESE 环境：当指定了“NO”时，优化器会限制将“NOT EXISTS”子查询变换为反连接的机会。 在 ESE 和非 ESE 环境中，当指定了 EXTEND 时，优化器搜索将“NOT IN”和“NOT EXISTS”子查询都变换为反连接的机会。		
DB2_CORRELATED_PREDICATES	所有	缺省值 = Yes 值：Yes 或 No
此变量的缺省值是“Yes”。当在一个连接中的相关列上存在唯一索引且此注册表变量为“Yes”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“Yes”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。如果在 C1 和 C2 上存在一个索引，还会对简单等式谓词（如 `WHERE C1=5 AND C2=10`）的相关性进行调整。索引不需要是唯一的，但是等式谓词列必须覆盖索引中的所有列。		
DB2_HASH_JOIN	所有	缺省值 = YES 值：YES 或 NO
将散列连接指定为当编译存取方案时可能的连接方法。		
DB2_INLIST_TO_NLJN	所有	缺省值 = NO 值：YES 或 NO

表 51. SQL 编译器变量 (续)

<table>
<tr><th>变量名</th><th>操作系统</th><th>值</th></tr>
<tr><th colspan="3">描述</th></tr>
<tr><td colspan="3">在某些情况下，SQL 编译器可以将 IN 列表谓词重写入连接。例如，在以下查询中:
<pre>
   SELECT *
    FROM EMPLOYEE
    WHERE DEPTNO IN ('D11', 'D21', 'E21')
</pre>可以编写为:<pre>
       SELECT *
    FROM EMPLOYEE, (VALUES 'D11', 'D21', 'E21) AS V(DNO)
    WHERE DEPTNO = V.DNO
</pre>如果 DEPTNO 上有索引，则此修订版可以提供更好的性能。首先存取值列表，然后使用索引应用连接谓词，通过嵌套循环连接将它连接到 EMPLOYEE。

有时，优化器没有准确的信息来确定用于查询的重写版本的最好连接方法。如果 IN 列表包含阻止优化器使用目录统计信息来确定选择性的参数标记或主机变量，则会发生这种情况。此注册表变量导致优化器使用表来支持嵌套循环连接来连接值列表，该表提供 IN 列表作为连接中的内部表。</td></tr>
<tr><td>DB2_LIKE_VARCHAR</td><td>所有</td><td>缺省值 = Y,Y</td></tr>
<tr><td colspan="3">控制子元素统计信息的使用。当数据具有空格定界的一系列子字段或子元素形式的结构时，它们是列中具有关于数据内容的统计信息。子元素统计信息的收集是可选的并由 RUNSTATS 命令或 API 中的选项来控制。

此注册表变量将影响优化器如何处理以下格式的谓词:<pre>
  COLUMN LIKE '%xxxxxx%'
</pre>其中 xxxxxx 是任意一个字符串。

显示如何使用此注册表变量的语法如下:<pre>
  db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]
</pre>其中
• 逗号前的项或者谓词右边唯一的项，具有以下含义（仅当第二项指定为 N 或该列没有正的子元素统计信息）:
– S – 优化器根据括在 % 字符中的字符串的长度，估计并置在一起构成一个列的一系列元素中每个元素的长度。
– Y – 缺省值。对算法参数使用缺省值 1.9。将变长子元素算法与算法参数一起使用。
– N – 使用定长子元素算法。
– num1 – 将 num1 的值与变长度子元素算法一起用作算法参数。
• 逗号后的项具有以下含义（仅适用于有正的子元素统计信息的列）:
– N – 不要使用子元素统计信息。第一项生效
– Y – 缺省值。使用变长子元素算法，该算法在列具有正的子元素统计信息的情况下，将子元素统计信息与算法参数的 1.9 缺省值结合使用。
– num2 – 使用变长子元素算法，该算法在列具有正的子元素统计信息的情况下，将子元素统计信息与算法参数的 num2 的值结合使用。</td></tr>
<tr><td>DB2_MINIMIZE_LISTPREFETCH</td><td>所有</td><td>缺省值 = NO

值: YES 或 NO</td></tr>
</table>

表 51. SQL 编译器变量 (续)

<table>
<tr><th>变量名</th><th>操作系统</th><th>值</th></tr>
<tr><th colspan="3">描述</th></tr>
<tr><td colspan="3">列表预取是特殊的表存取方法，该方法涉及从索引检索限定 RID、按页号排序它们和预取数据页。 有时，优化器没有准确的信息来确定列表预取是否是好的存取方法。当谓词选择性包含阻止优化器使用目录统计信息来确定选择性的参数标记或主机变量时，可能发生这种情况。

此注册表变量阻止优化器在这种情况下考虑列表预取。</td></tr>
<tr><td>DB2_SELECTIVITY</td><td>全部</td><td>缺省值 = No

值: Yes 或 No</td></tr>
<tr><td colspan="3">此注册表变量控制在 SQL 语句的搜索条件中，在哪里可以使用 SELECTIVITY 子句。

当此注册表变量设置为 “Yes” 时，可以为下列谓词指定 SELECTIVITY 子句:
• 最少有一个表达式包含主机变量的基本谓词
• 其中的 MATCH 表达式、谓词表达式或转义表达式包含主机变量的 LIKE 谓词</td></tr>
<tr><td>DB2_NEW_CORR_SQ_FF</td><td>所有</td><td>缺省值 = OFF

值: ON 或 OFF</td></tr>
<tr><td colspan="3">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。它可用于改善等式子查询谓词的选择值的精确性，这些谓词使用该子查询的 SELECT 列表中的 MIN 或 MAX 聚集函数。例如:

SELECT * FROM T WHERE
T.COL = (SELECT MIN(T.COL)
FROM T WHERE ...)</td></tr>
<tr><td>DB2_PRED_FACTORIZE</td><td>所有</td><td>缺省值 = NO

值: YES 或 NO</td></tr>
<tr><td colspan="3">指定优化器是否寻找机会来从析取项抽取其它谓词。在某些情况下，其它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'))</td></tr>
</table>

表 51. SQL 编译器变量 (续)

<table>
<tr><th>变量名</th><th>操作系统</th><th>值</th></tr>
<tr><th colspan="3">描述</th></tr>
<tr><td>DB2_REDUCED_OPTIMIZATION</td><td>所有</td><td>缺省值 = NO

值: NO、YES、任何整数或 DISABLE</td></tr>
<tr><td colspan="3">此注册表变量允许使您请求缩小的优化功能部件或指定的优化级别的优化功能部件的严格使用。如果减少使用优化技术的次数，也会减少优化期间时间和资源的使用。
注: 尽管可以减少优化时间和资源的使用，但会增大产生小于最佳数据存取方案的风险。仅当 IBM 或它的伙伴之一建议时，才使用此注册表变量。
• 如果设置为 NO

优化器将不更改它的优化技术。
• 如果设置为 YES

如果优化级别为 5（缺省值）或更低，优化器会禁用某些优化技术，这些技术可能消耗显著的准备时间和资源，但通常不会产生更好的存取方案。

如果优化级别恰好是 5，优化器按比例逐步减少或禁用某些附加技术，这些技术可能进一步减少优化时间和资源使用，但也进一步增加小于最佳存取方案的风险。对于小于 5 的优化级别，其中某些技术不可能在任何情况下都有效。但是，如果它们有效，则它们仍然有效。
• 如果设置为任何整数

效果与 YES 相同，对于在级别 5 优化的动态准备的查询，具有以下附加行为。如果任何查询块中的连接总数超出设置，则优化器切换至贪婪连接枚举，而不是禁用附加的优化技术，如以上对级别 5 优化级别所述。这意味将以与优化级别 2 类似的级别优化查询。
• 如果设置为 DISABLE

当不受此 DB2_REDUCED_OPTIMIZATION 变量约束时，优化器的行为有时会对优化级别 5 的动态查询动态地减少优化，此设置禁用此行为并要求优化器执行全部的级别 5 优化。

注意: 优化级别 5 的动态优化减少优先于对于当 DB2_REDUCED_OPTIMIZATION 设置为 YES 时恰好是 5 的优化级别所描述的行为以及对整数设置描述的行为。</td></tr>
<tr><td>DB2_SQLROUTINE_PREPOPTS</td><td>所有</td><td>缺省值 = 空字符串

值:
• BLOCKING {UNAMBIG | ALL | NO}
• DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
• DEGREE {1 | degree-of-parallelism | ANY}
• DYNAMICRULES {BIND | RUN}
• EXPLAIN {NO | YES | ALL}
• EXPLSNAP {NO | YES | ALL}
• FEDERATED {NO | YES}
• INSERT {DEF | BUF}
• ISOLATION {CS | RR | UR | RS | NC}
• QUERYOPT optimization-level
• VALIDATE {RUN | BIND}</td></tr>
</table>

表 51. SQL 编译器变量 (续)

变量名	操作系统	值
描述		
DB2_SQLROUTINE_PREPOPTS 注册表变量可以用来定制 SQL 过程的预编译和绑定选项。		

相关概念:

- 第 66 页的『优化级别准则』
- 第 146 页的『选择最佳连接的策略』
- 第 453 页的『DB2 注册表和环境变量』

相关参考:

- 第 68 页的『优化级别』

性能变量

表 52. 性能变量

变量名	操作系统	值
描述		
DB2AFFINITIES	AIX 5 或更高版本，除了 zSeries 之外的所有 Linux（32 位）	缺省值 = 未设置 值：指向配置文件的有效路径
定义可用来限制 DB2 使用哪些操作系统资源的资源策略。例如，在 AIX 或 Linux 上，可使用此注册表变量来限制 DB2 使用的一组处理器。 在启用了 AIX NUMA 的机器上，可以定义指定 DB2 将使用的资源集的策略。当使用资源集绑定时，将把各个 DB2 进程绑定至特定资源集。这对于一些性能调整方案很有用。 可以设置该注册表变量来指示指向这样一个配置文件的路径：该配置文件定义用于将 DB2 进程绑定至操作系统资源的策略。资源策略使您可以指定一组操作系统资源来限制 DB2。每个 DB2 进程都被绑定至该资源集的单个资源。资源分配是以循环方式进行的。 样本配置文件:		

```
示例 1: 将所有 DB2 进程绑定至 CPU 1 或 3。
  <RESOURCE_POLICY>
  <METHOD>CPU</METHOD>
  <RESOURCE>1</RESOURCE>
  <RESOURCE>3</RESOURCE>
  </RESOURCE_POLICY>

示例 2: 将 DB2 进程绑定至下列其中一个资源集:
  sys/node.03.00000、sys/node.03.00001、sys/node.03.00002
和 sys/node.03.00003
  <RESOURCE_POLICY>
  <METHOD>RSET</METHOD>
  <RESOURCE>sys/node.03.00000</RESOURCE>
  <RESOURCE>sys/node.03.00001</RESOURCE>
  <RESOURCE>sys/node.03.00002</RESOURCE>
  <RESOURCE>sys/node.03.00003</RESOURCE>
  </RESOURCE_POLICY>
```

注: 使用 RSET 方法需要具有 CAP_NUMA_ATTACH 能力，并且在 Linux 上不受支持。

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
DB2_ALLOCATION_SIZE	所有	缺省值 = 8 MB 范围：32 KB 至 256 MB
指定缓冲池的内存分配大小。 为此注册表变量设置较高的值可能具有下列优点：达到期望对缓冲池分配的内存量将需要更少的分配。 为此注册表变量设置较高的值可能会增加成本，这是因为如果缓冲池的改变量不是分配大小的倍数，则可能会浪费内存。例如，如果 DB2_ALLOCATION_SIZE 的值为 8 MB，而缓冲池减少了 4 MB，则将浪费 4 MB，原因是不能释放整个 8 MB 段。		
DB2_APM_PERFORMANCE	所有	缺省值 = OFF 值：ON 或 OFF
将此变量设置为 ON 来启用存取方案管理器（APM）中与性能相关的更改，这些更改影响 SQL 高速缓存（包高速缓存）的行为。通常建议不要对生产系统使用这些设置。它们会引入一些限制。如程序包外的高速缓存错误和／或增加的内存使用的可能性。 将 DB2_APM_PERFORMANCE 设置为 ON 也会启用“无程序包锁定”方式。此方式允许“全局 SQL 高速缓存”运行，而不必使用程序包锁定，这些锁定是内部系统锁定，可以保护高速缓存的程序包条目不会被除去。“无程序包锁定”方式可能导致稍微提高性能，但不允许某些数据库操作。这些被禁止的操作可能包括：使程序包无效的操作、使程序包不起作用的操作、PRECOMPILE、BIND 和 REBIND。		
DB2ASSUMEUPDATE	所有	缺省值 = OFF 值：ON 或 OFF
当启用此变量时，允许 DB2 假定真实更改了 UPDATE 语句中提供的所有定长列。这使得 DB2 无需将现有列值与提供的新值进行比较就可以确定实际上是否正在进行更改。当为更新提供了一些列（例如，在 SET 子句中）但是实际上没有进行修改时，使用此注册表变量会导致更多日志记录和索引维护。 在更新时将检查此注册表变量。		
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取。如果 DB2_AVOID_PREFETCH=ON，则不使用预取。		
DB2_AWE	Windows 2000	缺省值 = null 值：<entry>[;<entry>...] 其中 <entry>=<buffer pool ID>,<number of physical pages>, <number of address windows>
允许 32 位 Windows 2000 平台上的 DB2 UDB 分配缓冲池，该缓冲池最多使用 64 GB 内存。必须正确配置 Windows 2000，以支持“地址窗口扩展”（AWE）缓冲池。它包括将“在内存中锁定页”权限与用户关联、分配物理页和地址窗口页和设置此注册表变量。在设置此变量时，您需要了解要用于 AWE 支持的缓冲池的缓冲池标识。可以在 SYSCAT.BUFFERPOOLS 系统目录视图的 BUFFERPOOLID 列中查看缓冲池标识。 **注:** • 如果启用了 AWE 支持，则扩充存储器不能用于数据库中的任何缓冲池。 • 此注册表变量引用的缓冲池必须在 SYSCAT.SYSBUFFERPOOLS 中已存在。 • 为 AWE 启用的缓冲池的级别高于为基于块的 I/O 启用的缓冲池。如果为 AWE 和基于块的 I/O 都配置了缓冲池，则 AWE 的级别高于基于块的 I/O。		

表 52. 性能变量 (续)

变量名	**操作系统**	**值**
描述		
DB2_BINSORT	所有	缺省值 = YES 值: YES 或 NO
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	如对每个参数指定的	缺省值 = 路径

两组参数可用于调整缓冲池。一组参数（仅在 Windows 上可用）指定缓冲池应对特定类型的容器使用散射读。另一组参数（在所有平台上都可用）影响预取行为。
 	
以格式 `parameter=value` 在 ASCII 文件中指定参数，每行一个参数。例如，名为 `bpvars.vars` 的文件可能包含下列行:
 	
`NO_NT_SCATTER = 1`
`NUMPREFETCHQUEUES = 2`
 	
假定 `bpvars.vars` 存储在 F:\vars\ 中，执行下列命令来设置这些变量:
 	
`db2set DB2BPVARS=F:\vars\bpvars.vars`
 	
散射读参数
 	
对于针对相应类型的容器有大量顺序预取的系统以及您已将 DB2NTNOCACHE 设置为 ON 的系统，建议使用这些散射读参数。这些参数仅在 Windows 平台上可用，它们是 NT_SCATTER_DMSFILE、NT_SCATTER_DMSDEVICE 和 NT_SCATTER_SMS。指定 NO_NT_SCATTER 参数，以显式地禁止对任何容器的散射读。特定参数用于对指示的类型的所有容器打开散射读。对于这些参数中的每个参数，缺省值为零（或 OFF）；且可能的值包括：零（或 OFF）和 1（或 ON）。
注: 仅当 DB2NTNOCACHE 设置为 ON 来关闭 Windows 文件高速缓存时，才可以打开散射读。如果 DB2NTNOCACHE 设置为 OFF 或未设置，当您试图对任何容器打开散射读时，会向管理通知记录中写入一条警告消息，且仍然禁用散射读。
 	
预取调整参数
 	
预取调整参数是 NUMPREFETCHQUEUES 和 PREFETCHQUEUESIZE。这些参数在所有平台上都可用，且可用于改进缓冲池数据预取。例如，考虑其中期望的 PREFETCHSIZE 划分为 PREFETCHSIZE/EXTENTSIZE 预取请求的顺序预取。在这种情况下，将请求放置在预取队列上，从该队列分派 I/O 服务器来执行异步 I/O。缺省情况下，DB2 对每个数据库分区都维护一个大小为 100 与 2*NUM_IOSERVERS 之间的最大值的一个队列。在某些环境中，随着队列和／或不同大小的队列的增加，性能会得到提高。预取队列的数目应该最多为 I/O 服务器数目的一半。当设置这些参数时，考虑其它参数（如 PREFETCHSIZE、EXTENTSIZE、NUM_IOSERVERS 和缓冲池大小），以及工作负载特征（例如，当前用户数目）。
 	
如果认为缺省值对于您的环境太小，首先稍稍增大该值。例如，您可以设置 NUMPREFETCHQUEUES=4 和 PREFETCHQUEUESIZE=200。以受控方式对这些参数进行更改，以便可以监视和评估这些更改的效果。
 	
对于 NUMPREFETCHQUEUES，缺省值是 1，该值的范围是 1 至 NUM_IOSERVERS。如果将 NUMPREFETCHQUEUES 设置为小于 1，将它调整为 1，如果将它设置为大于 NUM_IOSERVERS，将它调整为 NUM_IOSERVERS。
 	
对于 PREFETCHQUEUESIZE，缺省值是 max(100,2*NUM_IOSERVERS)。值的范围是从 1 至 32767。如果将 PREFETCHQUEUESIZE 设置为小于 1，将它调整为缺省值。如果设置大于 32767，将它调整为 32767。		

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
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。		
DB2CHKSQLDA	所有	缺省值 = OFF, 值: ON 或 OFF
指定是否需要对输入进行 SQLDA 检查。		
DB2_ENABLE_BUFPD	所有	缺省值 = YES 值: ON 或 OFF
指定 DB2 是否使用中间缓冲来提高查询性能。此缓冲并非能在所有环境中提高查询性能。应执行测试来确定个别的查询性能提高。		
DB2_EVALUNCOMMITTED	所有	缺省值 = OFF 值: ON 或 OFF
当启用此变量时，在可能的情况下，将进行表或索引存取扫描以延迟或避免行锁定，直到知道数据记录满足谓词求值为止。 当启用此变量时，可对未落实的数据进行谓词求值。 它只适用于使用“游标稳定性”或“读稳定性”隔离级别的语句。对于索引扫描，索引必须是类型 2 索引。 而且，除非还设置了注册表变量 DB2_SKIPDELETED，否则，对于表扫描存取将无条件地跳过已删除的行，而对于类型 2 索引扫描则不会跳过已删除的键。 在 **db2start** 时激活此 DB2_EVALUNCOMMITTED 注册表变量是有效的，而有关延迟锁定是否合适的决定则是在语句编译或绑定时作出的。		
DB2_EXTENDED_OPTIMIZATION	所有	缺省值 = OFF 值: ON 或 OFF
指定查询优化器是否使用优化扩充功能来提高查询性能。此扩充功能并不能在所有环境中提高查询性能。应执行测试来确定个别的查询性能提高。		
DB2_KEEPTABLELOCK	所有	缺省值 = OFF 值: ON 或 OFF
当启用此变量时，如果关闭了“未落实的读”或“游标稳定性”隔离级别，则允许 DB2 不释放表锁定。当事务结束时将释放保持的表锁定，就象为“读稳定性”和“可重复读”扫描释放表锁定一样。 在进行语句编译或绑定时将检查此注册表变量。		
DB2_LGPAGE_BP	仅 AIX 5.x 64 位 Linux	缺省值 = OFF 值: ON 或 OFF

表 52. 性能变量 (续)

<table>
<tr><th>变量名</th><th>操作系统</th><th>值</th></tr>
<tr><th colspan="3">描述</th></tr>
<tr><td colspan="3">DB2_LGPAGE_BP 注册表变量用来在 AIX 5.x 或者具有适当内核支持的任何 Linux 体系结构上运行时启用大页支持。只有 DB2 UDB AIX 版的 64 位版和 DB2 UDB Linux 版才支持此注册表变量。使用大页的目的主要是为高性能计算应用程序提供性能提高。使用大量虚拟内存的密集内存存取型应用程序可以通过使用大页来提高性能。要使 DB2 能够使用大页，必须首先将操作系统配置为使用大页。

在 64 位的 DB2 AIX 版上，启用此变量将减小共享内存段的大小，使数据库内存减少到最低需求（缺省情况是创建 64GB 段 – 有关更多详细信息，请参阅 database_memory 配置参数）。这是为了避免使保留在 RAM 中的共享内存多于可能要使用的共享内存。

通过设置此变量，将限制动态增加总的数据库共享内存配置（例如，增大缓冲池大小）的能力。

在 Linux 上，还要求 libcap.so 库可用。必须先安装此库才能使此选项起作用。如果打开了此选项而系统中没有该库，则 DB2 将禁用大型内核页并且继续象以前那样工作。

在 Linux 上，要验证大型内核页是否可用，发出以下命令:

<code>cat /proc/meminfo</code>

如果大型内核页可用，则应该出现下面三行（机器上配置的内存量不同，显示的数目也会不同）

<code>HugePages_Total: 200</code>
<code>HugePages_Free: 200</code>
<code>Hugepagesize: 16384 kB</code>

如果未出现这些行，或者 HugePages_Total 为 0，则需要操作系统或内核的配置。</td></tr>
<tr><td>DB2MAXFSCRSEARCH</td><td>所有</td><td>缺省值 = 5

值: -1、1 至 33 554</td></tr>
<tr><td colspan="3">指定当将记录添加至表中时，要搜索的可用空间控制记录数。缺省值是搜索 5 个可用空间控制记录。修改此值允许您平衡插入速度与空间重用。使用大的值以优化空间重用。使用小的值以优化插入速度。将值设置为 -1 强制数据库管理器搜索所有可用空间控制记录。</td></tr>
<tr><td>DB2_MAX_NON_TABLE_LOCKS</td><td>所有</td><td>缺省值 = YES

值: 请参阅下面的信息。</td></tr>
<tr><td colspan="3">定义在事务释放所有 NON 表锁定之前该事务可具有的 NON 表锁定的最大数目。 NON 表锁定是这样一些表锁定: 即使事务已使用完它们，它们仍保留在散列表和事务链中。因为事务经常多次存取同一个表，所以保留锁定并将它们的状态更改为 NON 可以提高性能，这是因为将不必重新创建锁定。

为了获得最佳结果，此变量的建议值是想要任何连接存取的表的最大数目。如果未指定用户定义的值，则缺省值为如下所示: 如果锁定列表大小大于或等于 SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS（当前为 8000），则缺省值将为 SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE（当前为 150）。否则，缺省值将为 SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL（当前为 0）。</td></tr>
<tr><td>DB2MEMDISCLAIM</td><td>AIX</td><td>缺省值 = YES

值: YES 或 NO</td></tr>
<tr><td colspan="3">在 AIX 上，DB2 处理器使用的内存可能有一些关联的调页空间。甚至当已经释放关联的内存时，该调页空间可能仍为保留的。是否如此，取决于 AIX 系统的（可调的）虚拟内存管理资源分配策略。DB2MEMDISCLAIM 注册表变量控制 DB2 代理进程是否显式请求 AIX 从释放的内存中解除关联保留的调页空间。

DB2MEMDISCLAIM 设置为 YES 会导致较少的调页空间需求，并且可能导致较少的页面调度磁盘活动。DB2MEMDISCLAIM 设置为 NO 将会导致较多的调页空间需求，并且可能导致更多的页面调度磁盘活动。在某些情况下，例如，如果调页空间很大，并且实内存很大从不会发生页面调度，则设置为 NO 会提供较小的性能提高。</td></tr>
</table>

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
DB2MEMMAXFREE	所有	缺省值 = 8 388 608 个字节 值: 0 至 2^{32}-1 个字节
指定未使用的专用内存的最大字节数，在将未使用的内存返回到操作系统之前该字节数由 DB2 进程保留。		
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。 当将这些变量设置为 ON 时，对 DB2 缓冲池读写的数据将绕过 AIX 内存高速缓存。如果 DB2 缓冲池相对较小，则您不能或选择不增大此缓冲池的大小，您应该考虑通过将 DB2_MMAP_READ 和 DB2_MMAP_WRITE 设置为 OFF 来利用 AIX 内存高速缓存。		
DB2_MMAP_WRITE	AIX	缺省值 = ON 值: ON 或 OFF
与 DB2_MMAP_READ 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。 当将这些变量设置为 ON 时，对 DB2 缓冲池读写的数据将绕过 AIX 内存高速缓存。如果 DB2 缓冲池相对较小，则您不能或选择不增大此缓冲池的大小，您应该考虑通过将 DB2_MMAP_READ 和 DB2_MMAP_WRITE 设置为 OFF 来利用 AIX 内存高速缓存。		
DB2_NO_FORK_CHECK	UNIX	缺省值 = OFF 值: ON 或 OFF
当启用此注册表变量时，DB2 运行时客户机将使确定当前过程是否是派生调用的结果而进行的检查次数最少。这样可提高不使用 fork() API 的 DB2 应用程序的性能。		
DB2_NO_MPFA_FOR_NEW_DB	所有	缺省值 = 未设置 值: YES
CREATE DATABASE 命令或等价的 API 创建的数据库启用了多页文件分配（MPFA）。一旦对数据库启用了 MPFA，就不能再禁用它。要创建一个禁用了 MPFA 的数据库，应将此注册表变量设置为 YES 并在创建数据库之前重新启动实例。当设置了此注册表变量时，创建的任何数据库都将禁用 MPFA。 要对一个禁用了 MPFA 的数据库启用 MPFA，使用 **db2empfa** 命令。		
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。DB2NTMEMSIZE 允许用户在必要时覆盖 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) 受防护的存储过程: 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

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
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。这适用于除包含长字段或 LOB 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。 在 Windows NT 中，在打开文件时对文件进行高速缓存，这是缺省行为。为文件中的每 1 GB 从系统池保留 1 MB。使用此注册表变量来覆盖高速缓存的无正式文件的 192 MB 限制。当达到该高速缓存限制时，给出资源不够错误。		
DB2NTPRICLASS	Windows NT	缺省值 = null 值：R、H 或任何其它值
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 线程相对于其它线程的绝对优先级。 **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。 有关更多信息，请参阅 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 未处于页面调度情况时，进程的工作集可按需要增大。但是，当发生页面调度时，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：正数页数 OR <entry>[;<entry>...] 其中 <entry>=<buffer pool ID>,<number of pages>
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。如果数据库管理器连 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器试图分配大而没有精确配置的缓冲池。如果设置了此值，它将覆盖当前的缓冲池大小。 还可以使用 <entry>[;<entry>...]（其中 <entry>=<buffer pool ID>,<number of pages>）来临时更改所有缓冲池或部分缓冲池的大小以使它们可以启动。		
DB2_PINNED_BP	AIX, HP-UX	缺省值 = NO 值：YES 或 NO

表 52. 性能变量 (续)

变量名	操作系统	值
描述		

此变量用于指定在某些 AIX 操作系统上与主存中的数据库关联的数据库全局内存（包括缓冲池）。保持系统主存中的数据库全局内存允许数据库的性能更加一致。
 	
例如，如果将缓冲池交换到系统主存外，数据库性能会恶化。通过在系统内存中具有缓冲池来减少磁盘 I/O 可提高数据库的性能。如果其它应用程序需要更多主存，则取决于系统主存需求，允许在主存外交换数据库全局内存。
 	
在 64 位的 DB2 AIX 版上，启用此变量将减小共享内存段的大小，使数据库内存减少到最低需求（缺省情况是创建 64GB 段 – 有关更多详细信息，请参阅 database_memory 配置参数）。这是为了避免使保留在 RAM 中的共享内存多于可能要使用的共享内存。
 	
通过设置此变量，将限制动态增加总的数据库共享内存配置（例如，增大缓冲池大小）的能力。
 	
对于 64 位环境中的 HP-UX，除了修改此注册表变量外，还必须给予 DB2 实例组 MLOCK 特权。为此，具有 root 用户存取权限的用户要执行下列操作:
1. 将 DB2 实例添加至 `/etc/privgroup` 文件中。例如，如果 DB2 实例组属于 `db2iadm1` 组，则必须将下面这一行添加到 `/etc/privgroup` 文件中:
`db2iadm1 MLOCK`
2. 发出以下命令:
`setprivgrp -f /etc/privgroup`		

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_SCATTERED_IO	Linux	缺省值 = OFF 值: ON 或 OFF
此变量打开分散 I/O，分散 I/O 使用 readv() 来读取磁盘。如果正在包含向量化原始 I/O 性能提高 Linux 内核补丁的系统中运行，则应该打开此变量来提高性能。 当前，在用于 IA-32 的 UnitedLinux 1.0 SP2 或更高版本中提供了此内核补丁，在将来所有的 Linux 2.6 内核中都将提供此内核补丁。		
DB2_SKIPDELETED	所有	缺省值 = OFF 值: ON 或 OFF
当启用了此注册表变量时，允许使用“游标稳定性”或“读稳定性”隔离级别的语句（在索引存取期间）无条件地跳过已删除的键和（在表存取期间）跳过已删除的行。当启用了 DB2_EVALUNCOMMITTED 时，将自动跳过已删除的行，但是，除非还启用了 DB2_SKIPDELETED，否则，将不会跳过类型 2 索引中未落实的伪删除键。 此注册表变量不会影响 DB2 目录表上的游标的行为。 在执行 db2start 时，可激活此注册表变量。		
DB2_SMS_TRUNC_TMPTABLE_THRESH	Windows	-1 和 0-n，其中 n = 每个容器中要维护的扩展数据块的数目

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
指定表示将在 SMS 表空间中维护的临时表的文件的最小文件大小阈值。将此变量设置为大于 0 的值可减少每次使用临时表时删除并重新创建文件所造成的系统开销中的一部分开销。缺省情况下，当不再需要临时表时，该表的文件将被截断为每个容器 1 个扩展数据块。如果该文件的大小已经是一个扩展数据块或更小，则它将保持现状。当此变量的值大于 1 时，将维护更大的文件。 如果此变量设置为 -1，则根本不会截断该文件，并且允许它无限增长，只不过会受到系统资源的限制。 如果此变量设置为 0，则不会执行特殊阈值处理。然而，一旦不再需要临时表，该文件就会被截断为 0。		
DB2_SORT_AFTER_TQ	所有	缺省值 = NO 值: YES 或 NO
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。		
DB2_SELUDI_COMM_BUFFER	所有	缺省值 = OFF 值 = ON 或 OFF
适用于通过 UPDATE/INSERT/DELETE（UDI）查询中的 SELECT 来处理分块游标。当启用此注册表变量时，它会阻止将查询结果存储在临时表中。然而，在对 UDI 查询的 SELECT 的分块游标执行 OPEN 处理期间， DB2 会尝试将查询的整个结果直接缓存到通信缓冲区内存区域中。 **注**: 如果通信缓冲区空间不足以容纳查询的整个结果，则发出 SQLCODE -906 并回滚事务。有关分别调整本地和远程应用程序的通信缓冲区内存区域大小的信息，请参阅 *aslheapsz* 和 *rqrioblk* 数据库管理器配置参数。 在分区数据库环境中或者启用了分区内并行性时，不支持此注册表变量。		
DB2_TRUSTED_BINDIN	所有	缺省值 = OFF 值 = OFF、ON 或 CHECK

表 52. 性能变量 (续)

变量名	操作系统	值
描述		
当启用了 DB2_TRUSTED_BINDIN 时，会加速执行不受防护的嵌入式存储过程中包含主机变量的 SQL 语句。 当启用了此变量时，在绑定不受防护的嵌入式存储过程中包含的 SQL 语句期间，不会将外部 SQLDA 格式转换为内部 DB2 格式。这将加速处理嵌入式 SQL 语句。 当启用了此变量时，不受防护的嵌入式存储过程中不支持下列数据类型： • SQL_TYP_DATE • SQL_TYP_TIME • SQL_TYP_STAMP • SQL_TYP_DATALINK • SQL_TYP_CGSTR • SQL_TYP_BLOB • SQL_TYP_CLOB • SQL_TYP_DBCLOB • SQL_TYP_CSTR • SQL_TYP_LSTR • SQL_TYP_BLOB_LOCATOR • SQL_TYP_CLOB_LOCATOR • SQL_TYP_DCLOB_LOCATOR • SQL_TYP_BLOB_FILE • SQL_TYP_CLOB_FILE • SQL_TYP_DCLOB_FILE • SQL_TYP_BLOB_FILE_OBSOLETE • SQL_TYP_CLOB_FILE_OBSOLETE • SQL_TYP_DCLOB_FILE_OBSOLETE 如果遇到了这些数据类型，则将返回 SQLCODE −804 和 SQLSTATE 07002。 **注**：输入主机变量的数据类型和长度必须与相应元素的内部数据类型和长度精确匹配。对于主机变量，将始终满足此要求。但是，对于参数标记，必须注意确保使用相匹配的数据类型。可以使用 CHECK 选项来确保所有输入主机变量的数据类型和长度都匹配，但是此选项将不利于性能提高。		
DB2_USE_ALTERNATE_PAGE_CLEANING	所有	缺省值 = 未设置 值：ON 或 OFF
指定 DB2 是否使用页清除算法的另一种方法而不使用页清理的缺省方法。当此变量设置为“ON”时，DB2 使用主动方法进行页清除、将更改的页写入磁盘、保持在 LSN_GAP 前面并且主动查找干扰页。这样做允许页清除程序更好地利用可用的磁盘 I/O 带宽。当此变量设置为“ON”时，chngpgs_thresh 数据库配置参数不再相关，原因是它不会控制页清除程序活动。		

相关概念:

• 第 453 页的『DB2 注册表和环境变量』

Data links 变量

表 53. Data Links 变量

变量名	操作系统	值
描述		
DLFM_BACKUP_DIR_NAME	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：null 值：任何有效路径
指定当 DLFM_BACKUP_TARGET 设置为 LOCAL 时要在其中备份归档文件的目录的路径。		
DLFM_BACKUP_TARGET	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：null 值：LOCAL、TSM 或 XBSA
指定要使用的备份介质。 如果将此变量设置为 LOCAL，则必须设置 DLFM_BACKUP_DIR_NAME 变量。 如果将此变量设置为 XBSA，则必须设置 DLFM_BACKUP_TARGET_LIBRARY 变量。 即使将此注册表变量的设置由一个目标更改为另一个目标，也不会移动已归档文件。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_TARGET_LIBRARY	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：null 值：DLL 或共享库名的任何有效路径
指定与 XBSA 相符的归档服务器 DLL 或共享库的全限定路径。这个库使用 *libdfmxbsa.a* 库来装入。 如果将 DLFM_BACKUP_TARGET 设置为 XBSA，则必须设置此变量。如果将 DLFM_BACKUP_TARGET 变量设置为另一个值，则此变量不适用。		
DLFM_GC_MODE	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：PASSIVE 值：SLEEP、PASSIVE 或 ACTIVE
指定在 Data Links 服务器上对垃圾文件收集的控制。当设置为 SLEEP 时，不进行垃圾收集。当设置为 PASSIVE 时，仅当没有其它事务运行时才运行垃圾收集。当设置为 ACTIVE 时，即使其它事务在运行也运行垃圾收集。		
DLFM_INSTALL_PATH	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值 在 AIX 和 Solaris Operating Environment 上：`/home/<instance>/sqllib/bin`，其中 <instance> 是 Data Links Manager 实例标识 在 Windows 上：`%DB2PATH%\bin`（如果设置了 %DB2PATH%）或 `c:\sqllib\bin`（如果未设置 %DB2PATH%） 范围：任何有效路径
指定 Data Links 可执行文件的安装路径。		

表 53. Data Links 变量 (续)

变量名	操作系统	值
描述		
DLFM_PORT	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	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 Data Links 服务器通信的端口号。		
DLFM_TSM_MGMTCLASS	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：缺省 TSM 管理类 值：任何有效的 TSM 管理类
指定要用于归档和检索已链接文件的 TSM 管理类。如果未为此变量设置值，则使用缺省 TSM 管理类。		
DLFM_START_ASNCOPYD	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：NO 值：YES，NO
指定每当启动 DLFM 时是否启动“Data Links Manager 复制守护进程”（DLFM_ASNCOPYD）。当使用 DB2 Replication 来将 DATALINK 文件复制至 Data Links Manager 服务器或从 Data Links Manager 服务器复制 DATALINK 文件时，必须启动“Data Links Manager 复制守护进程”。 当将此变量设置为 YES 时，必须设置 DLFM_ASNCOPYD_PORT 变量。		
DLFM_ASNCOPYD_PORT	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：null 值：任何有效的端口号
指定“Data Links Manager 复制守护进程”（DLFM_ASNCOPYD）用来侦听文件复制请求的 TCP/IP 端口号。 当将 DLFM_START_ASNCOPYD 变量设置为 YES 时，必须指定此变量。		
DLFM_NUM_ARCHIVE_SUBSYSTEMS	AIX、Windows NT、Windows 2000 和 Solaris Operating Environment	缺省值：2 值：任何大于或等于 1 的数字
指定要在给定 DLFM 服务器下运行的“DLFM 复制守护进程”进程的数目。复制进程的数目越大，备份链接文件的吞吐量就越大。然而，这个值应该与可用于将链接文件复制至所指定的归档区域的 I/O 通道的数目相对应。如果值太大，则所消耗的系统资源量可能会减少 I/O 并行性的好处。		
DLFM_AUTOSTART	AIX，Solaris Operating Environment	缺省值：NO 值：YES，NO
指定每当操作系统重新引导时 DLFM 服务器是否自动启动。在引导处理期间，从 `/etc/inittab` 文件中调用的 `dlfsmount` 脚本将检查此变量。		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

其它变量

表 54. 其它变量

变量名	操作系统	值
描述		
DB2ADMINSERVER	Windows 和 UNIX	缺省值 = null
指定“DB2 管理服务器”。		
DB2CLIINIPATH	所有	缺省值 = null
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。		
DB2_COMMIT_ON_EXIT	UNIX	缺省值 = OFF 值：OFF/NO/0 或 ON/YES/1
在 UNIX 平台上，在版本 8 之前，当成功退出应用程序时，DB2 将落实任何仍然未完成的事务。在版本 8 中，该行为更改为退出时将回滚未完成的事务。此注册表变量允许具有依赖于先前行为的应用程序的用户在版本 8 中继续启用它。 注意，在版本 10 中将建议不要使用此注册表变量，并且将不再支持“退出时落实”这种行为。用户应该确定他们在版本 8 之前开发的任何应用程序是否继续依赖于此功能，并根据需要将适当的显式 COMMIT 语句添加至应用程序。如果打开该注册表变量，则应该注意不要实现在退出之前未能显式 COMMIT（落实）的新应用程序。 大多数用户都应该将此注册表变量保留为缺省设置。		
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，模拟此选项的运行时行为。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： libdb2drda.a、libdb2net8.a、libdb2informix.a、db2drda.dll、db2net8.dll 和 db2informix.a 等等。
指定当启动数据库管理器时装入的包装器库。指定此变量将减少装入频繁使用的包装器的运行时成本。支持其它操作系统的其它值（.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。 除非将数据库管理器参数 FEDERATED 设置为 YES，否则将忽略此变量。		
DB2_DJ_INI	所有	缺省值： • UNIX: db2_instance_directory/cfg/db2dj.ini • Windows: db2_install_directory\cfg\db2dj.ini

表 54. 其它变量 (续)

变量名	操作系统	值
描述		
指定联合配置文件的绝对路径名，例如：db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini，此文件包含数据源环境变量的设置。这些环境变量由 Informix 包装器和 DB2 Information Integrator 提供的包装器使用。 下面是一个样本联合配置文件： INFORMIXDIR=/informix/client_sdk INFORMIXSERVER=inf93 ORACLE_HOME=/usr/oracle9i SYBASE=/sybase/V12 SYBASE_OCS=OCS-12_5 下列限制适用于 db2dj.ini 文件： • 各个条目必须遵循以下格式：*evname=value*，其中 evname 是环境变量的名称，而 value 是它的值。 • 环境变量名称的最大长度为 255 个字节。 • 环境变量值的最大长度为 765 个字节。 除非将数据库管理器参数 FEDERATED 设置为 YES，否则将忽略此变量。		
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 概要文件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 不使用现有的备份域控制器，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播至备份域控制器时，则可能导致不同步。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_DOCHOST	所有	缺省值：http://publib.boulder.ibm.com/infocenter/db2help/ http://*hostname*，其中 *hostname*= 有效主机名或 IP 地址
指定安装了 DB2 信息中心的主机名。如果在 DB2 安装向导中选择了自动配置选项，则在安装 DB2 信息中心期间可以自动设置此变量。		
DB2_DOCPORT	所有	缺省值：NULL 值：任何有效的端口号
指定 DB2 帮助系统为 DB2 文档服务时使用的端口号。如果在 DB2 安装向导中选择了自动配置选项，则在安装 DB2 信息中心期间可以自动设置此变量。		
DB2_EXTSECURITY	Windows 平台	缺省值 = ON 值：ON 或 OFF
通过锁定 DB2 系统文件来防止对 DB2 进行未经授权的访问。为避免发生潜在的问题，不应该关闭此注册表变量。		
DB2_ENABLE_LDAP	所有	缺省值 = NO 值：YES 或 NO
指定是否使用“轻量级目录访问协议”（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

表 54. 其它变量 (续)

变量名	操作系统	值
描述		
此变量允许在回退处理期间强制所有数据库连接断开。它与故障转移支持一起用于带有“Microsoft 群集服务器”（MSCS）的 Windows NT 环境中。如果 DB2_FALLBACK 未设置或设置为 OFF，且在回退期间数据库连接一直存在，则不会导致 DB2 资源脱机。这意味着回退处理将失败。		
DB2_FMP_COMM_HEAPSZ	Windows，除 AIX 外的所有 UNIX	20mb 或足够的空间来运行 10 个受防护的例程（以较大者为准）
此变量指定用于受防护的例程调用（例如，存储过程或用户定义的函数调用）的池的大小（以 4 KB 页计）。每个受防护的例程使用的空间是 aslheapsz 配置参数值的两倍。 如果正在系统上运行大量的受防护的例程，则可能需要增大此变量的值。如果只是运行非常少量的受防护的例程，则可以减小这个变量值。 将此值设置为 0 表示不创建集合，因此无法调用任何受防护的例程。它还意味着将禁用健康监视器和数据库自动维护功能（例如，自动备份、收集统计信息和 REORG），原因是此功能依赖于受防护的例程基础结构。		
DB2_GRP_LOOKUP	Windows NT	缺省值 = null 值：LOCAL，DOMAIN
此变量用于告知 DB2 到何处验证用户帐户并执行组成员查找。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户。将此变量设置为 DOMAIN 以强制 DB2 总是在用户帐户所属的 Windows NT 域上枚举组和验证用户帐户。		
DB2_HADR_BUF_SIZE	所有	缺省值 = 2*LOGBUFSZ
此变量指定备用日志接收缓冲区大小（以日志页为单位）。如果未设置此变量，则 DB2 将两次把主要 LOGBUFSZ 配置参数值用作备用接收缓冲区大小。应该在备用实例中设置此变量。主数据库会忽略它。 如果 HADR 同步方式（HADR_SYNCMODE 数据库配置参数）设置为 ASYNC，则在对等状态下，低速的备用数据库可使对主数据库的发送操作停止，因此会阻塞对主数据库的事务处理。可以在备用数据库上配置大于缺省日志接收缓冲区的缓冲区用来保存更多未处理的日志数据。这可使得在很短的时间内主数据库生成日志数据的速度比备用数据库使用这些数据的速度更快，而不会阻塞主数据库上的事务处理。		
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。
 	
要确保高速缓存中具有最新的条目，请执行下列操作:
 	
`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条目。		

DB2LDAP_CLIENT_PROVIDER	Windows	缺省值 = null（如果可用的话，使用 Microsoft；否则使用 IBM。） 值：IBM 或 Microsoft

表 54. 其它变量 (续)

变量名	操作系统	值
描述		
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，可使用 `db2set` 命令： `db2set DB2LDAP_CLIENT_PROVIDER`		
DB2LDAPHOST	所有	缺省值 = null 值：任何有效主机名。
指定 LDAP 目录的位置的主机名。		
DB2LDAP_KEEP_CONNECTION	所有	缺省值 = YES 值：YES，NO
指定 DB2 是否对它的内部 LDAP 连接句柄进行高速缓存。当此变量设置为 NO 时， DB2 将不把它的 LDAP 连接句柄高速缓存到目录服务器中。这可能会降低性能，但是如果需要使与目录服务器建立的同时处于活动状态的 LDAP 客户机连接数最小，则可能需要将 DB2LDAP_KEEP_CONNECTION 设置为 NO。 要获得最佳性能，缺省情况下，应将此变量设置为 YES。 DB2LDAP_KEEP_CONNECTION 注册表变量只是作为 LDAP 中的全局级概要文件注册表变量来实现的，因此必须通过对 **db2set** 命令指定 -gl 选项来设置它，如下所示： `db2set -gl DB2LDAP_KEEP_CONNECTION=NO`		
DB2LDAP_SEARCH_SCOPE	所有	缺省值 = DOMAIN 值：LOCAL、DOMAIN 或 GLOBAL
指定以“轻量级目录访问协议”（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。		
DB2_LOAD_COPY_NO_OVERRIDE	所有	缺省值：NONRECOVERABLE 值：COPY YES 或 NONRECOVERABLE
此变量将把任何 LOAD COPY NO 转换为 LOAD COPY YES 或 NONRECOVERABLE，这取决于该变量的值。此变量适用于 HADR 主数据库以及标准（非 HADR）数据库；HADR 备用数据库将忽略此变量。在 HADR 主数据库上，如果未设置此变量，则 LOAD COPY NO 将转换为 LOAD NONRECOVERABLE。此变量的值指定不可恢复的装入或复制目标，它使用与 COPY YES 子句相同的语法。		
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：STATEMENT
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。任何其它设置都导致事务回滚。		
DB2_NEWLOGPATH2	UNIX	缺省值 = 0 值：0 或 1
该参数允许您指定是否应该使用辅助路径以实现双日志记录。通过对 *logpath* 数据库配置参数的当前值追加一个“2”来生成辅助路径名称。		

表 54. 其它变量 (续)

变量名	操作系统	值
描述		
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之前卸装它的应用程序，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。		
DB2OLDEVMON	所有	值: 用逗号分隔的事件监视器名称 *evmon1*, *evmon2*, ...
指定采用版本 6 之前的格式来编写数据的事件监视器的名称。在 DB2 版本 6 中，自描述数据流成为输出到文件和管道的系统监视器输出的标准形式。在版本 6 之前，系统监视器数据是以固定数据结构返回的。		
DB2REMOTEPREG	Windows NT	缺省值 = null 值: 任何有效的 Windows NT 机器名
指定包含 DB2 实例概要文件和 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 或 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。		
DB2SATELLITEID	所有	缺省值 = null 值: 在卫星控制数据库中声明的有效卫星标识
指定在卫星进行同步时被传递至卫星控制服务器的卫星标识。如果没有为此变量指定值，则将登录标识用作卫星标识。		
DB2SORT	所有，仅适用于服务器	缺省值 = null
指定 LOAD 实用程序在运行时要装入的库的位置。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。		
DB2SYSTEM	Windows 和 UNIX	缺省值 = null
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“客户机配置助手”的“搜索网络”功能时，DB2“发现”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 上，安装程序将它设置成为 Windows 系统指定的计算机名称。 • 在 UNIX 系统上，将它设置成 UNIX 系统的 TCP/IP 主机名。		
DB2_VENDOR_INI	AIX、HP-UX、Solaris Operating Environment 和 Windows	缺省值 = null 值: 任何有效路径和文件。
指向包含所有特定于供应商的环境设置的文件。当数据库管理器启动时读取该值。		
DB2_XBSA_LIBRARY	AIX、HP-UX、Solaris Operating Environment 和 Windows	缺省值 = null 值: 任何有效路径和文件。

表 54. 其它变量 (续)

变量名	操作系统	值
描述		

指向供应商提供的 XBSA 库。在 AIX 上，如果共享对象未命名为 shr.o，设置必须包括该共享对象。HP-UX、Solaris Operating Environment 和 Windows NT 不需要共享对象名。例如，要对 DB2 使用 Legato 的 NetWorker Business Suite Module，则必须按如下所示设置注册表变量:
`db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"`
通过 BACKUP DATABASE 或 RESTORE DATABASE 命令可以调用 XBSA 接口。例如:
`db2 backup db sample use XBSA`
`db2 restore db sample use XBSA`		

相关概念:

- 第 453 页的『DB2 注册表和环境变量』

附录 B. 说明表

说明表

当激活说明设施时，说明表将捕获存取方案。在可以调用说明之前必须创建说明表。可以使用备有证明文件的表定义创建它们，也可以调用在 EXPLAIN.DDL 文件中提供的样本命令行处理器（CLP）脚本来创建它们，该 EXPLAIN.DLL 文件位于“sqllib”目录的“misc”子目录中。要调用该脚本，连接至需要说明表的数据库，然后发出以下命令:

```
db2 -tf EXPLAIN.DDL
```

说明设施对说明表的填充将不会激活触发器、参考约束或检查约束。例如，如果在 EXPLAIN_INSTANCE 表上定义了插入触发器，并说明了适合的语句，则不会激活触发器。

相关参考:

- 第 488 页的『EXPLAIN_ARGUMENT 表』
- 第 494 页的『EXPLAIN_OBJECT 表』
- 第 496 页的『EXPLAIN_OPERATOR 表』
- 第 498 页的『EXPLAIN_PREDICATE 表』
- 第 502 页的『EXPLAIN_STREAM 表』
- 第 504 页的『ADVISE_INDEX 表』
- 第 510 页的『ADVISE_WORKLOAD 表』
- 第 492 页的『EXPLAIN_INSTANCE 表』
- 第 500 页的『EXPLAIN_STATEMENT 表』
- 第 506 页的『ADVISE_INSTANCE 表』
- 第 507 页的『ADVISE_MQT 表』
- 第 508 页的『ADVISE_PARTITION 表』
- 第 509 页的『ADVISE_TABLE 表』

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 55. EXPLAIN_ARGUMENT 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	说明动态语句时运行的程序包名或说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	FK	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此说明信息相关的节号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一标识。
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(2M)	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 56. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	描述
AGGMODE	COMPLETE PARTIAL INTERMEDIATE FINAL	部分聚集指示符。
BITFLTR	TRUE FALSE	“散列连接”将使用位过滤器来增强性能。
BLD_LEVEL	DB2 构建标识	源代码版本的内部标识字符串。
BLKLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT SHARE 无 SHARE UPDATE	块级别锁定意向。
CSERQY	TRUE FALSE	远程查询是一个常用子表达式。
CSETEMP	TRUE FALSE	常用子表达式标志上的临时表。
DIRECT	TRUE	直接访存指示符。
DSTSEVER	服务器名称	目标（装运自）服务器。
DUPLWARN	TRUE FALSE	重复的警告标志。
EARLYOUT	LEFT RIGHT 无	早出指示符。

表 56. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	描述
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
FETCHMAX	IGNORE INTEGER	FETCH 运算符上的 MAXPAGES 自变量的重设值。
GREEDY	TRUE	指示优化器使用“贪婪”算法来计划存取。
GROUPBYC	TRUE FALSE	是否提供了 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 • 顺序值 **（A）** 升序 **（D）** 降序	内部顺序列。
ISCANMAX	IGNORE INTEGER	ISCAN 运算符上的 MAXPAGES 自变量的重设值。
JN_INPUT	INNER OUTER	指示运算符是否是提供内连接或外连接的运算符。
LISTENER	TRUE FALSE	“侦听器表队列”指示符。
MAXPAGES	ALL 无 INTEGER	期望“预取”的最大页数。
MAXRIDS	无 INTEGER	要包括在每个列表预取请求中的“最大行标识”。
NUMROWS	INTEGER	期望排序的行数。
ONEFETCH	TRUE FALSE	一个“访存”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 • 顺序值 **（A）** 升序 **（D）** 降序	外部顺序列。
OUTERJN	LEFT RIGHT FULL LEFT (ANTI) RIGHT (ANTI)	外连接指示符。
PARTCOLS	列名	运算符的分区列。

表 56. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	描述
PREFETCH	LIST 无 SEQUENTIAL	合格的预取类型。
RMTQTEXT	查询文本	远程查询文本
RNG_PROD	函数名	用于扩展索引存取的范围生成函数。
ROWLOCK	EXCLUSIVE NONE REUSE SHARE SHORT (INSTANT) SHARE UPDATE	行锁定意向。
ROWWIDTH	INTEGER	要排序的行宽。
RSUFFIX	查询文本	远程 SQL 后缀。
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。
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	单个代理进程产生的分区内并行性排序或 temp。
SORTKEY	此类型的每行将包含: • 键中列的顺序值（后跟冒号和空格） • 列名 • 顺序值 **(A)** 升序 **(D)** 降序	排序键列。
SORTTYPE	PARTITIONED SHARED ROUND ROBIN REPLICATED	分区内并行性，排序类型。
SRCSEVER	服务器名称	源（装运到）服务器。
STREAM	TRUE FALSE	远程源正在进行流式传输。
TABLOCK	EXCLUSIVE INTENT EXCLUSIVE INTENT NONE INTENT SHARE REUSE SHARE SHARE INTENT EXCLUSIVE SUPER EXCLUSIVE UPDATE	表锁定意向。
TEMPSIZE	INTEGER	临时表页大小。

表 56. ARGUMENT_TYPE 和 ARGUMENT_VALUE 列值 (续)

ARGUMENT_TYPE 值	可能的 ARGUMENT_VALUE 值	描述
TQDEGREE	INTEGER	分区内并行性，访问“表队列”的子代理进程数。
TQMERGE	TRUE FALSE	合并（排序的）“表队列”指示符。
TQREAD	READ AHEAD STEPPING SUBQUERY STEPPING	“表队列”读取属性。
TQSEND	BROADCAST DIRECTED SCATTER SUBQUERY DIRECTED	“表队列”发送属性。
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: • 键中列的顺序值（后跟冒号和空格） • 列名	唯一键列。
VOLATILE	TRUE	易变的表

EXPLAIN_INSTANCE 表

EXPLAIN_INSTANCE 表是所有说明信息的主控制表。说明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 表提供关于说明的 SQL 语句的源基本信息，以及进行说明所在的环境的信息。

表 57. EXPLAIN_INSTANCE 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	PK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	PK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	PK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	PK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	PK	“说明”请求的源的版本。
EXPLAIN_OPTION	CHAR(1)	否	否	指示为此请求所请求的说明信息。 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说明快照。 可能的值是: **Y** 是，抽取了说明快照并存储在 EXPLAIN_STATEMENT 表中。同时捕获了常规的说明信息。 **N** 未抽取说明快照。捕获常规的说明信息。 **O** 只抽取了一个说明快照。未捕获常规的说明信息。
DB2_VERSION	CHAR(7)	否	否	处理此说明请求的 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** 维护版号
SQL_TYPE	CHAR(1)	否	否	指示说明实例是静态 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 编译器在调用说明时使用的查询优化级别。该值指示 SQL 编译器对正说明的 SQL 语句执行的查询优化级别。
BLOCK	CHAR(1)	否	否	指示编译 SQL 语句时所使用的游标分块类型。有关更多信息，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

表 57. EXPLAIN_INSTANCE 表 (续). PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
ISOLATION	CHAR(2)	否	否	指示编译 SQL 语句时所使用的隔离类型。有关更多信息，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	包含调用说明设施时 BUFFPAGE 数据库配置设置的值。
AVG_APPLS	INTEGER	否	否	包含调用说明设施时 AVG_APPLS 配置参数的值。
SORTHEAP	INTEGER	否	否	包含调用说明设施时 SORTHEAP 数据库配置设置的值。
LOCKLIST	INTEGER	否	否	包含调用说明设施时 LOCKLIST 数据库配置设置的值。
MAXLOCKS	SMALLINT	否	否	包含调用说明设施时 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。（从 LOCKLIST 和 MAXLOCKS 中派生。）
CPU_SPEED	DOUBLE	否	否	包含调用说明设施时 CPUSPEED 数据库管理器配置设置的值。
REMARKS	VARCHAR(254)	是	否	用户提供的注释。
DBHEAP	INTEGER	否	否	包含调用说明设施时 DBHEAP 数据库配置设置的值。
COMM_SPEED	DOUBLE	否	否	包含调用说明设施时 COMM_BANDWIDTH 数据库配置设置的值。
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 = 非联合系统方案 • Y = 联合系统方案

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 58. EXPLAIN_OBJECT 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	FK	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此说明信息相关的节号。
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	对象的创建时间；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	此对象中的估计行数。
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。
DISTINCT	CHAR(1)	否	否	指示对象中的行是否各不相同（即，是否有重复的行）。 可能的值为: **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 操作估计的总开销。包括控制器开销、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	当执行预取时要读取的数据页数。对表函数设置为 -1。
EXTENTSIZE	INTEGER	否	否	扩展数据块的大小，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	带索引的数据群集的程度。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子页数。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。

表 58. EXPLAIN_OBJECT 表 (续). PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
FULLKEYCARD	BIGINT	否	否	此索引对象中包含的相异完整键值的数目。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	相异的第一个键值的数目。对于表、表函数或者如果此统计信息不可用，则设置为 −1。
FIRST2KEYCARD	BIGINT	否	否	使用索引前 {2,3,4} 列的前面几个相异键值数。对于表、表函数或者如果此统计信息不可用，则设置为 −1。
FIRST3KEYCARD	BIGINT	否	否	
FIRST4KEYCARD	BIGINT	否	否	
SEQUENTIAL_PAGES	INTEGER	否	否	磁盘上的叶子页数；叶子页之间的间隔很小，按索引键顺序排列。对于表、表函数或者如果此统计信息不可用，则设置为 −1。
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）。对于表、表函数或者如果此统计信息不可用，则设置为 −1。
STATS_SRC	CHAR(1)	否	否	指示统计信息的源。如果是来自于单个节点，则设置为 1。
AVERAGE_SEQUENCE_GAP	DOUBLE	否	否	序列之间的间隔。
AVERAGE_SEQUENCE_FETCH_GAP	DOUBLE	否	否	当使用索引来进行访存时序列之间的间隔。
AVERAGE_SEQUENCE_PAGES	DOUBLE	否	否	可按顺序存取的平均索引页数。
AVERAGE_SEQUENCE_FETCH_PAGES	DOUBLE	否	否	当使用索引来进行访存时可按顺序存取的平均表页数。
AVERAGE_RANDOM_PAGES	DOUBLE	否	否	在各顺序页存取之间的平均随机索引页数。
AVERAGE_RANDOM_FETCH_PAGES	DOUBLE	否	否	当使用索引来进行访存时各顺序页存取之间的平均随机表页数。
NUMRIDS	BIGINT	否	否	索引中行标识的总数。
NUMRIDS_DELETED	BIGINT	否	否	索引中伪删除的行标识的总数。
NUM_EMPTY_LEAFS	BIGINT	否	否	索引中空叶子页的总数。
ACTIVE_BLOCKS	BIGINT	否	否	表中的活动多维群集（MDC）块的总数。

表 59. 可能的 OBJECT_TYPE 值

值	描述
IX	索引
TA	表
TF	表函数

EXPLAIN_OPERATOR 表

EXPLAIN_OPERATOR 表包含 SQL 编译器为满足 SQL 语句所需的所有运算符。

表 60. EXPLAIN_OPERATOR 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空？	键？	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	FK	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此说明信息相关的节号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一标识。
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开销。
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（以指令为单位）。
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 61. OPERATOR_TYPE 值

值	描述
DELETE	删除
FETCH	访存
FILTER	过滤器行
GENROW	生成行
GRPBY	分组方式

表 61. OPERATOR_TYPE 值 (续)

值	描述
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（RID）扫描
SHIP	将查询交付给远程系统
SORT	排序
TBSCAN	表扫描
TEMP	临时表构造
TQ	表队列
UNION	联合
UNIQUE	消除重复
UPDATE	更新

EXPLAIN_PREDICATE 表

EXPLAIN_PREDICATE 表标识特定的运算符应用哪些谓词。

表 62. EXPLAIN_PREDICATE 表. PK 意味着列是主键的一部分; FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	FK	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此说明信息相关的节号。
OPERATOR_ID	INTEGER	否	否	此查询中此运算符的唯一标识。
PREDICATE_ID	INTEGER	否	否	指定运算符的此谓词的唯一标识。
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。
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(2M)	是	否	如同根据 SQL 语句的内部表示再创建的谓词文本。 如果在编译语句期间使用，它还将包含主机变量、专用寄存器或参数标记的值。 仅当由具有 DBADM 权限的用户执行该语句时，才会将值转储至 EXPLAIN_PREDICATE 表。注意，尽管没有 DBADM 权限的其他用户将不能把值转储到此表中，但是，如果值在表中，这些用户仍然能够查看该值。 如果不可用，则为空。

表 63. 可能的 HOW_APPLIED 值

值	描述
BSARG	作为每个块的控制谓词求值
JOIN	用来连接表
RESID	作为剩余的谓词求值
SARG	作为索引或数据页的控制谓词求值
START	用作启动条件
STOP	用作停止条件

表 64. 可能的 RELOP_TYPE 值

值	描述
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

当存在不同级别的说明信息的 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 编译器确定的附加的谓词增强。

表 65. EXPLAIN_STATEMENT 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	PK，FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	PK，FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	PK，FK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	PK，FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	PK	与此行相关的说明信息的级别。 有效的值为: **O** 初始文本（用户输入的） **P** PLAN SELECTION
STMTNO	INTEGER	否	PK	程序包中与此说明信息相关的语句号。对于动态说明 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 语句的节号。对于动态说明 SQL 语句，这是在运行时保存此语句的段的节号。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	说明的 SQL 语句的数值标识。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	每个说明的 SQL 语句的标识标记。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。
STATEMENT_TYPE	CHAR(2)	否	否	说明的查询类型的描述性标注。 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

表 65. EXPLAIN_STATEMENT 表 (续). PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
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(2M)	否	否	说明的 SQL 语句的文本或部分文本。为说明的“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，因为在捕获语句的此特定版本时未选择存取方案。
QUERY_DEGREE	INTEGER	否	否	指示调用说明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。
REOPT	CHAR(1)	否	否	指示是否使用主机变量、参数标记或专用寄存器的实际值来重新优化 SQL 语句。 可能的值为: **N** 使用缺省变量估计来创建存取路径。 **Y** 使用主机变量、参数标记或专用寄存器的可用值来重新优化存取路径。

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 66. EXPLAIN_STREAM 表. PK 意味着列是主键的一部分; FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	FK	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	FK	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	FK	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	FK	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	FK	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	FK	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	FK	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	FK	程序包中与此说明信息相关的节号。
STREAM_ID	INTEGER	否	否	指定的运算符中此数据流的唯一标识。
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标识，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标识，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。
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，则该谓词标识将在此反映出来，否则将该列设置为 -1。
COLUMN_NAMES	CLOB(2M)	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	分区映射标识。

表 66. EXPLAIN_STREAM 表 (续). PK 意味着列是主键的一部分; FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
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** 使用行标识定向 **FUNC** 使用函数（HASHEDVALUE() 或 DBPARTITIONNUM()）定向 **CORR** 使用相关值定向 **Numberic** 定向至预先确定的单一节点
PARTITION_COLUMNS	CLOB(2M)	是	否	将此数据流分区时所依据的列的列表。

ADVISE_INDEX 表

ADVISE_INDEX 表提供了建议的索引。

表 67. ADVISE_INDEX 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	否	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	否	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	否	说明动态语句时运行的程序包名或说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	否	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	否	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	否	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	否	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	否	程序包中与此说明信息相关的节号。
QUERYNO	INTEGER	否	否	说明的 SQL 语句的数值标识。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	每个说明的 SQL 语句的标识标记。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昵称。
TBCREATOR	VARCHAR(128)	否	否	表名的限定符。
COLNAMES	CLOB(2M)	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 = 只允许唯一的条目
COLCOUNT	SMALLINT	否	否	键中的列数以及包括列（如果有）的列数。
IID	SMALLINT	否	否	内部索引标识。
NLEAF	INTEGER	否	否	叶子页数；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。
FIRSTKEYCARD	BIGINT	否	否	第一个相异键值数；如果未收集统计信息，则为 −1。
FULLKEYCARD	BIGINT	否	否	全相异键值数；如果未收集统计信息，则为 −1。
CLUSTERRATIO	SMALLINT	否	否	带索引的数据群集的程度；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	群集度的更准确度量，如果没有收集详细的索引统计信息或者根据昵称定义了索引，则为 −1。
USERDEFINED	SMALLINT	否	否	由用户定义。

表 67. ADVISE_INDEX 表 (续). PK 意味着列是主键的一部分; FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
SYSTEM_REQUIRED	SMALLINT	否	否	如果满足下面两个条件中的任何一个条件，则为 1: – 此索引对于主键或唯一键约束是必需的，或此索引是多维群集（MDC）表的维块索引或组合块索引。 – 这是类型表的（OID）列上的索引。 如果同时满足下面两个条件，则为 2: – 此索引对于主键或唯一键约束是必需的，或此索引是 MDC 表的维块索引或组合块索引。 – 这是类型表的（OID）列上的索引。 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访存数。（如果没有可用的数据，则字符串长度为零。）
REMARKS	VARCHAR(254)	是	否	用户提供的注释，或者为空值。
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	磁盘上的叶子页数；叶子页之间的间隔很小，按索引键顺序排列。（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键值数（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键值数（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键值数（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，每个索引叶子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	唯一键必需的列数。总是 <=COLCOUNT。仅当有包括列时，才 < COLCOUNT。如果索引没有唯一键（允许重复值），则为 −1
MINPCTUSED	SMALLINT	否	否	如果不为零，则启用联机索引碎片整理，且该值为在合并页之前最小已用空间的阈值。
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
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(2M)	否	否	用来创建索引的 SQL 语句。
PACKED_DESC	BLOB(1M)	是	否	表的内部描述。

ADVISE_INSTANCE 表

ADVISE_INSTANCE 表包含有关 **db2advis** 执行情况的信息，包括起始时间。每次执行 **db2advis** 都会产生相应的一行。其它 ADVISE 表具有链接至 ADVISE_INSTANCE 表的 START_TIME 列的外键（RUN_ID），ADVISE_INSTANCE 表是在同一“顾问程序”运行期间为行创建的。

表 68. ADVISE_INSTANCE 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
START_TIME	TIMESTAMP	否	PK	开始执行 **db2advis** 的时间。
END_TIME	TIMESTAMP	否	否	结束执行 **db2advis** 的时间。
MODE	VARCHAR(4)	否	否	“设计顾问程序”上带有 -m 选项指定的值，例如，’MC’ 指定 MQT 和 MDC。
WKLD_COMPRESSION	CHAR(4)	否	否	“设计顾问程序”正在其下运行的工作负载压缩。
STATUS	CHAR(9)	否	否	“设计顾问程序”的运行状态。状态可以为 ’STARTED’、’COMPLETED’（如果成功的话）或带有前缀“EI”或“EX”错误号，“EI”表示内部错误，“EX”表示外部错误，在这种情况下，该错误号表示 SQLCODE。

ADVISE_MQT 表

ADVISE_MQT 表包含有关“设计顾问程序”建议的具体查询表（MQT）的信息。

表 69. ADVISE_MQT 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	否	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	否	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	否	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	否	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	否	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	否	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	否	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	否	程序包中与此说明信息相关的语句号。
NAME	VARCHAR(128)	否	否	MQT 名称
CREATOR	VARCHAR(128)	否	否	MQT 创建程序名称
IID	SMALLINT	否	否	内部标识
CREATE_TIME	TIMESTAMP	否	否	创建 MQT 的时间
STATS_TIME	TIMESTAMP	是	否	进行统计信息的时间
NUMROWS	DOUBLE	否	否	MQT 中估计的行数
NUMCOLS	SMALLINT	否	否	MQT 中定义的列数
ROWSIZE	DOUBLE	否	否	MQT 中行的平均长度（按字节计）
BENEFIT	FLOAT	否	否	保留以备将来使用。
USE_MQT	CHAR(1)	是	否	当推荐 MQT 时，设置为 ’Y’。
MQT_SOURCE	CHAR(1)	是	否	指示在何处生成了 MQT 标识。如果 MQT 候选值为立即刷新 MQT，则设置为 ’I’，或者，如果它只可能创建为完全刷新延迟 MQT，则设置为 ’D’。
QUERY_TEXT	CLOB(2M)	否	否	包括定义 MQT 的查询。
CREATION_TEXT	CLOB(2M)	否	否	包括用于 MQT 的 CREATE TABLE DDL。
SAMPLE_TEXT	CLOB(2M)	否	否	包括用来获取 MQT 详细的统计信息的采样查询。只有当“设计顾问程序”需要详细的统计信息时才使用。所得到的采样统计信息将显示在该表中。如果为空的话，则没有为该 MQT 创建采样查询。
COLSTATS	CLOB(2M)	否	否	包括 MQT 的列统计信息（如果不为空的话）。这些统计信息是 XML 格式的并包括列名、列基数以及（可选）HIGH2KEY 和 LOW2KEY 值。
EXTRA_INFO	BLOB(2M)	否	否	保留以用于其它输出。
TBSPACE	VARCHAR(128)	否	否	为 MQT 推荐的表空间。
RUN_ID	TIMESTAMP	是	FK	与 ADVISE_INSTANCE 表中的行的 START_TIME 相应的值，将它链接至同一“顾问程序”运行。
REFRESH_TYPE	CHAR(1)	否	否	设置为 ’I’ 表示立即或设置为 ’D’ 表示延迟。
EXISTS	CHAR(1)	否	否	如果 MQT 在数据库目录中存在，则设置为 ’Y’。

ADVISE_PARTITION 表

ADVISE_PARTITION 表包含有关“设计顾问程序”建议的数据库分区的信息，并且只能在分区数据库环境中填充它。

表 70. ADVISE_PARTITION 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
EXPLAIN_REQUESTER	VARCHAR(128)	否	否	发出此“说明”请求的启动器的授权标识。
EXPLAIN_TIME	TIMESTAMP	否	否	“说明”请求的启动时间。
SOURCE_NAME	VARCHAR(128)	否	否	当说明动态语句时运行的程序包名或当说明静态 SQL 时的源文件名。
SOURCE_SCHEMA	VARCHAR(128)	否	否	“说明”请求的源的模式或限定符。
SOURCE_VERSION	VARCHAR(64)	否	否	“说明”请求的源的版本。
EXPLAIN_LEVEL	CHAR(1)	否	否	与此行相关的说明信息的级别。
STMTNO	INTEGER	否	否	程序包中与此说明信息相关的语句号。
SECTNO	INTEGER	否	否	程序包中与此说明信息相关的语句号。
QUERYNO	INTEGER	否	否	说明的 SQL 语句的数值标识。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	每个说明的 SQL 语句的标识标记。对于通过 CLP 发出的动态 SQL 语句（EXPLAIN SQL 语句除外），缺省值为“CLP”。对于通过 CLI 发出的动态 SQL 语句（EXPLAIN SQL 语句除外），缺省值为“CLI”。否则，所用的缺省值为空白。
TBNAME	VARCHAR(128)	是	否	指定表名。
TBCREATOR	VARCHAR(128)	是	否	指定表创建程序的名称。
PMID	SMALLINT	是	否	指定分区映射标识。
TBSPACE	VARCHAR(128)	是	否	指定表所驻留的表空间。
COLNAMES	CLOB(2M)	是	否	指定分区列名称，用逗号分开。
COLCOUNT	SMALLINT	是	否	指定分区列的数目。
REPLICATE	CHAR(1)	是	否	指定是否复制分区。
COST	DOUBLE	是	否	指定使用分区的成本。
USEIT	CHAR(1)	是	否	指定是否以 EVALUATE PARTITION 方式使用分区。如果 USEIT 设置为 Y 或 y，则使用分区。
RUN_ID	TIMESTAMP	是	FK	与 ADVISE_INSTANCE 表中的行的 START_TIME 相应的值，将它链接至同一“顾问程序”运行。

ADVISE_TABLE 表

ADVISE_TABLE 表存储数据定义语言（DDL），以便使用“设计顾问程序”对具体查询表（MQT）、多维群集表（MDC）和分区的最终建议来创建表。

表 71. ADVISE_TABLE 表. PK 意味着列是主键的一部分；FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
RUN_ID	TIMESTAMP	是	FK	与 ADVISE_INSTANCE 表中的行的 START_TIME 相应的值，将它链接至同一“顾问程序”运行。
TABLE_NAME	VARCHAR(128)	否	否	表的名称
TABLE_SCHEMA	VARCHAR(128)	否	否	表创建程序的名称
TABLESPACE	VARCHAR(128)	否	否	要在其中创建表的表空间
SELECTION_FLAG	VARCHAR(4)	否	否	指示推荐类型。有效的值是 ’M’ 用于 MQT、’P’ 用于分区以及 ’C’ 用于 MDC。此字段可以包括这些值的任何子集。例如，’MC’ 指示将表推荐为 MQT 和 MDC 表。
TABLE_EXISTS	CHAR(1)	否	否	如果该表在数据库目录中存在，则设置为 ’Y’。
USE_TABLE	CHAR(1)	否	否	如果“设置顾问程序”已推荐该表，则设置为 ’Y’。
GEN_COLUMNS	CLOB(2M)	否	否	如果该行包括 MDC 推荐（该推荐需要创建表 DDL 中的生成列），则包括生成列字符串。
ORGANIZE_BY	CLOB(2M)	否	否	对于 MDC 推荐，包括创建表 DDL 的 ORGANIZE BY 子句。
CREATION_TEXT	CLOB(2M)	否	否	包括创建表 DDL。
ALTER_COMMAND	CLOB(2M)	否	否	包括用于该表的 ALTER TABLE 语句。

ADVISE_WORKLOAD 表

ADVISE_WORKLOAD 表提供构成工作负载的语句。

表 72. ADVISE_WORKLOAD 表. PK 意味着列是主键的一部分; FK 意味着列是外键的一部分。

列名	数据类型	可空?	键?	描述
WORKLOAD_NAME	CHAR(128)	否	否	此语句所属的 SQL 语句集合（工作负载）的名称。
STATEMENT_NO	INTEGER	否	否	工作负载中与此说明信息相关的语句号。
STATEMENT_TEXT	CLOB(1M)	否	否	SQL 语句的内容。
STATEMENT_TAG	VARCHAR(256)	否	否	每个说明的 SQL 语句的标识标记。
FREQUENCY	INTEGER	否	否	此语句在工作负载中出现的次数。
IMPORTANCE	DOUBLE	否	否	此语句的重要性。
WEIGHT	DOUBLE	否	否	语句的优先级。
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）。
COMPILABLE	CHAR(17)	是	否	指示在尝试准备语句时发生的任何查询编译错误。如果此列为 NULL 或者不是以 SQLCA 开头，则可以由 db2advis 来编译 SQL 查询。如果 db2advis 或“设计顾问程序”找到了编译错误，则 COMPILABLE 列值由 8 字符的 SQLCA.sqlcaid 字段，后面接着是一个冒号（:），然后是一个 8 字符的 SQLCA.sqlstate 字段组成，它是 SQL 语句的返回码。

附录 C. SQL 说明工具

可以使用 db2expln 和 dynexpln 工具来了解为特定 SQL 语句选择的存取方案。也可使用“控制中心”中的集成说明设施以及 Visual Explain 来了解为特定 SQL 语句选择的存取方案。动态和静态 SQL 语句都可以使用说明设施来说明。与说明工具的一个不同之处是，使用 Visual Explain 时，说明信息是以图形方式显示的。而其它方面，这两种方法所提供的详细信息级别是等效的。

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（例如，SELECT 语句中的谓词）
- 程序包（存取方案）的用途
- 系统目录表的用途和内容
- 一般应用程序调整概念

本节中的主题提供有关 db2expln 和 dynexpln 的信息。

SQL 说明工具

db2expln 工具描述为 SQL 语句选择的存取方案。当未捕获到说明数据时，它可用来获取所选存取方案的快速说明。对于静态 SQL，db2expln 检查存储在系统目录表中的程序包。对于动态 SQL，db2expln 检查 SQL 高速缓存中的各节。

dynexpln 工具也可以用来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。然而，因为可以通过 db2expln 检查动态 SQL，保留此实用程序仅为了向下兼容。

这些说明工具（db2expln 和 dynexpln）位于您的实例 sqllib 目录的 bin 子目录中。如果 db2expln 和 dynexpln 不在当前目录中，它们必须在 PATH 环境变量中出现的某个目录中。

首次访问数据库时，db2expln 程序连接并使用 db2expln.bnd、db2exsrv.bnd 和 db2exdyn.bnd 文件来将它自已绑定到数据库。

要运行 db2expln，您必须对系统目录视图具有 SELECT 特权并对 db2expln、db2exsrv 和 db2exdyn 程序包具有 EXECUTE 特权。 要运行 dynexpln，您必须对数据库具有 BINDADD 权限，您将用来连接至数据库的模式必须存在或您必须对数据库具有 EXPLICIT_SCHEMA 权限。要使用 db2expln 或 dynexpln 说明动态 SQL，还必须具有正在被说明的 SQL 语句所需要的所有特权。（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。）

相关概念:

- 第 171 页的『SQL 说明设施』
- 第 179 页的『捕获说明信息的准则』
- 第 181 页的『分析说明信息的准则』

db2expln

以下各节描述 *db2expln* 的语法和参数并提供使用情况注意事项。

db2expln – SQL 说明

命令语法:

```
>>--db2expln--+----------------------+--+------------------+------------------->
              '-| connection-options |'  '-| output-options |'

>--+-------------------+--+-------------------+--+-------------------+--------->
   '-| package-options |'  '-| dynamic-options |'  '-| explain-options |'

>--+-------+------------------------------------------------------------------><
   '--help-'
```

connection-options:

```
|---database--database-name--+------------------------------+-----------------|
                             '--user--user-id--password-----'
```

output-options:

```
|--+------------------------+--+-------------+--------------------------------|
   '- -output--output-file--'  '- -terminal--'
```

package-options:

```
|-- -schema--schema-name---package--package-name------------------------------>

>--+-------------------------------+--+------------------------------+-------->
   '--version--version-identifier--'  '--escape--escape-character---'

>--+-----------+--+-----------------------------+-----------------------------|
   '--noupper--'  '--section--section-number----'
```

dynamic-options:

```
|--+--------------------------------+--+----------------------------------+--->
   '--statement--sql-statement-----'  '--stmtfile--sql-statement-file----'

>--+-------------------------------------------+--+---------+-----------------|
   '--terminator--termination-character--------'  '--noenv--'
```

explain-options:

```
|--+----------+--+----------+-------------------------------------------------|
   '--graph---'  '--opids---'
```

命令参数:

可以按任何顺序指定该选项。

connection-options:

这些选项指定要连接至的数据库和进行连接所必需的任何选项。除了在指定了 **-help** 选项时以外，这些连接选项是必需的。

-database *database-name*

包含要说明的程序包的数据库的名称。

为了向下兼容，可以使用 **-d** 代替 **-database**。

-user *user-id password*

建立数据库连接时要使用的授权标识和密码。*user-id* 和 *password* 必须符合 DB2® 命名约定，且必须是数据库可识别的。

为了向下兼容，可以使用 **-u** 代替 **-user**。

output-options:

这些选项指定 `db2expln` 输出应该定向至的位置。除了在指定了 **-help** 选项时以外，必须至少指定一个输出选项。如果指定两个选项，输出发送到文件和终端。

-output *output-file*

`db2expln` 的输出写入到指定的文件。

为了向下兼容，可以使用 **-o** 代替 **-output**。

-terminal

将 `db2expln` 输出定向到终端。

为了向下兼容，可以使用 **-t** 代替 **-terminal**。

package-options:

这些选项指定一个或多个要说明的程序包和节。仅说明程序包和各节中的静态 SQL。

注: 与在 LIKE 谓词中一样，可以使用模式匹配字符，它们是百分比符号（%）和下划线（_），来指定 *schema-name*、*package-name* 和 *version-identifier*。

-schema *schema-name*

要说明的程序包的模式。

为了向下兼容，可以使用 **-c** 代替 **-schema**。

-package *package-name*

要说明的程序包的名称。

为了向下兼容，可以使用 **-p** 而不使用 **-package**。

-version *version-identifier*

要说明的程序包的版本标识。缺省版本是空字符串。

-escape *escape-character*

在 *schema-name*、*package-name* 和 *version-identifier* 要用作模式匹配的转义字符的字符 *escape-character*。

例如，说明程序包 TESTID.CALC% 的 `db2expln` 命令如下所示:

```
db2expln -schema TESTID -package CALC% ....
```

但是，此命令还可以说明以 CALC 开头的任何其它方案。要只说明 TESTID.CALC% 程序包，您必须使用转义字符。如果指定惊叹号（!）作为转义字符，可以将命令更改为: `db2expln -schema TESTID -escape ! -package CALC!% ...`。这样，! 字符用作转义字符，因此，!% 解释为 % 字符，而未解释为“匹配任何项”模式。没有缺省转义字符。

为了向下兼容，可以使用 **-e** 代替 **-escape**。

注: 为了避免问题，不要指定操作系统转义字符作为 `db2expln` 转义字符。

-noupper
指定在搜索匹配程序包之前不应将 *schema-name*、*package-name* 和 *version-identifier* 转换为大写字母。

缺省情况下，这些变量在搜索程序包之前转换为大写字母。此选项表示必须按输入正确使用这些值。

为了向下兼容，可以使用 **-l**（它是小写字母 L，不是数字 1）代替 **-noupper**。

-section *section-number*
所选一个或多个程序包内要说明的节号。

要说明每个程序包中的所有节，使用数字零（0）。这是缺省行为。如果未指定此选项，或如果 *schema-name*、*package-name* 或 *version-identifier* 包含模式匹配字符，显示所有节。

要查找节号，查询系统目录视图SYSCAT.STATEMENTS。参阅 *SQL Reference* 以获取系统目录视图的描述。

为了向下兼容，可以使用 **-s** 代替 **-section**。

dynamic-options:

这些选项指定一个或多个要说明的动态 SQL 语句。

-statement *sql-statement*
要动态准备和说明的 SQL 语句。要说明多个语句，使用 **-stmtfile** 选项来提供包含要说明的 SQL 语句的文件，或使用 **-terminator** 选项来定义可用来隔离 **-statement** 选项中的语句的终止字符。

为了与 `dynexpln` 的兼容性，可以使用 **-q** 代替 **-statement**。

-stmtfile *sql-statement-file*
包含一个或多个要动态准备和说明的 SQL 语句的文件。缺省情况下，假定文件的每行是单值 SQL 语句。如果语句必须跨行，使用 **-terminator** 选项来指定标记 SQL 语句结束的字符。

为了与 `dynexpln` 的兼容性，可以使用 **-q** 代替 **-stmtfile**。

-terminator *termination-character*
表示动态 SQL 语句结束的字符。缺省情况下，**-statement** 选项提供单个 SQL 语句，**-stmtfile** 中文件的每行作为单独的 SQL 语句进行处理。您指定的终止字符可以用来提供多个具有 **-statement** 的 SQL 语句或使语句跨越 **-stmtfile** 文件中的行。

为了与 `dynexpln` 的兼容性，可以使用 **-z** 代替 **-terminator**。

-noenv
指定改变编译环境的动态语句被说明之后，不应执行它们。

缺省情况下，db2expln 将执行以下任何一个语句（在说明它们之后）:

```
SET CURRENT DEFAULT TRANSFORM GROUP
SET CURRENT DEGREE
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
```

```
SET CURRENT QUERY OPTIMIZATION
SET CURRENT REFRESH AGE
SET PATH
SET SCHEMA
```

这些语句可以改变为 db2expln 所处理的后续动态 SQL 语句选择的方案。

如果指定 **-noenv**，则说明这些语句，但不执行它们。

说明动态 SQL 必须指定 **-statement** 或 **-stmtfile**。两个选项都可在 db2expln 的单个调用中指定。

explain-options:

这些选项确定在已说明的方案中提供哪些附加信息。

-graph
显示优化器方案图。检查每一节，并构造原始优化器方案图（如 Visual Explain 所显示的）。注意，生成的图与 Visual Explain 图可能并不完全匹配。

为了向下兼容，可以指定 **-g** 代替 **-graph**。

-opids 在说明的方案中显示运算符标识号。

运算符标识号允许将 db2expln 的输出与说明设施的输出匹配。注意，并非所有运算符都有标识号，出现在说明设施输出中的一些标识号不出现在 db2expln 输出中。

为了向下兼容，可以指定 **-i** 代替 **-opids**.

-help 显示 db2expln 的帮助文本。如果指定此选项，则不说明任何程序包。

大多数命令行在 *db2exsrv* 存储过程中处理。要获取有关所有可用选项的帮助，必须提供 **connection-options** 和 **-help**。例如，使用:

```
db2expln -help -database SAMPLE
```

为了向下兼容，可以指定 **-h** 或 **-?**。

使用注意事项:

除非指定 **-help** 选项，否则必须指定程序包选项或动态选项。可以用 db2expln 的单个调用说明程序包和动态 SQL。

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，也许可以在这些字符之前加操作系统转义字符来输入它们。有关更多信息，请参阅操作系统文档。确保没有无意指定操作系统转义字符作为 db2expln 转义字符。

db2expln 生成的帮助和初始状态消息被写入标准输出。所有由该说明工具生成的提示和其它状态消息被写入标准错误。根据选择的输出选项，说明文本被写入标准输出或文件。

示例:

要说明具有一个 db2expln 调用的多个方案，使用 **-package**、**-schema** 和 **-version** 选项，并对数据包和具有 LIKE 模式的创建程序指定字符串常量。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。

要说明 SAMPLE 数据库中所有程序包的所有节，并将结果写入文件 **my.exp** 中，可输入:

```
db2expln -database SAMPLE -schema % -package %  -output my.exp
```

作为另一个示例，假设某个用户具有称为“statements.db2”的 CLP 脚本文件且要说明该文件中的语句。该文件包含下列语句:

```
SET PATH=SYSIBM, SYSFUN, DEPT01, DEPT93@
SELECT EMPNO, TITLE(JOBID) FROM EMPLOYEE@
```

要说明这些语句，输入下列命令:

```
db2expln -database DEPTDATA -stmtfile statements.db2 -terminator @ -terminal
```

相关概念:

- 第 511 页的『SQL 说明工具』
- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

db2expln 的使用注意事项

以下是 db2expln 显示的常见消息:

- 对于以下数据库程序包模式，未找到程序包: "<creator>".<package>，版本为 "<version>"

  如果在数据库中找不到与指定模式匹配的程序包，则会输出此消息。
- 可在 db2expln.msg 中找到绑定消息

  如果 db2expln.bnd 的绑定未成功，则会输出此消息。有关所遇到的问题的进一步的信息可在当前目录中的 db2expln.msg 文件中找到。
- 对于可能的多个程序包，节号已覆盖为 0（所有段）。

  如果 db2expln 可能遇到多个程序包，则会输出此消息。如果在程序包或创建程序输入自变量中使用这些模式匹配字符的其中一个，则会执行此操作。
- 可以在 <message file> 中找到 <bind file> 的绑定消息。

  如果给定的绑定文件的绑定未成功，则会出现此消息。有关所遇到的问题的进一步的信息可在数据库服务器上的给定消息文件中找到。
- 程序包没有合格的静态节。

  如果指定的程序包只包含动态 SQL 语句，这表示没有静态节，则会输出此消息。
- 程序包 "<creator>"."<package>", "<version>" 无效。重新绑定程序包，然后重新运行 db2expln。

  如果指定的程序包当前无效，则会输出此消息。根据指示，重新发出 BIND 或 REBIND 命令，以便该方案在数据库中重新创建一个有效的程序包，然后重新运行 db2expln。

排除的 SQL 语句: 将不说明下列语句:
- 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节，这些节可通过 **db2expln** 来说明。

dynexpln

`dynexpln` 工具仍可用，以便向下兼容。然而，可以使用 `db2expln` 的**动态选项**来执行 `dynexpln` 的所有功能。

当使用 `db2expln` 的动态选项时，准备该语句作为真实的动态 SQL，且从 SQL 高速缓存说明生成的方案。此说明输出方法提供比 `dynexpln` 更精确的存取方案，它将语句准备为静态 SQL。它也允许使用仅在动态 SQL 中可用的功能部件，例如，参数标记。

相关概念:

- 第 511 页的『SQL 说明工具』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

说明输出信息

以下各节描述 *db2expln* 和 *dynexpln* 可以提供的信息种类。

db2expln 和 dynexpln 输出的描述

在该输出中，每个程序包的说明信息显示在下列两个部分:

- 程序包信息，如绑定日期和相关的绑定选项
- 后跟要说明的 SQL 语句的节信息，例如，节号。节信息之下，显示为 SQL 语句选择的存取方案的说明输出。

一个存取方案或节的步骤以数据库管理器执行它们的顺序显示。每个主要步骤显示为一个向左对齐的标题，有关该步骤的信息以缩进形式显示在该标题下。缩进条出现在

该存取方案的说明输出的左页边距中。这些条也标记操作的作用域。在返回到缩进的先前级别之前，在同一操作中处理更低级别的缩进（更远离右边）的操作。

记住，选择的存取方案是基于输出中显示的原始 SQL 语句的增强版本。例如，原始语句可能导致激活触发器和约束。另外，SQL 编译器的查询重写组件可以将 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，会在优化器使用的信息中包括所有这些因素。因此，在说明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。SQL 说明设施（包括说明表、SET CURRENT EXPLAIN 方式和 Visual Explain）显示用于优化的实际 SQL 语句，其格式为 SQL 风格的语句，它是通过逆向转换查询的内部表示而创建的。

当将 `db2expln` 或 `dynexpln` 的输出与说明设施的输出比较时，运算符标识选项（**-opids**）非常有用。每次 `db2expln` 或 `dynexpln` 开始处理说明设施中的新运算符时，就会将该新运算符标识号打印在说明方案的左边。运算符标识可以用于与各种格式的存取方案中的步骤匹配。注意，在说明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。

相关概念:

- 第 517 页的『dynexpln』
- 第 518 页的『表存取信息』
- 第 523 页的『临时表信息』
- 第 525 页的『连接信息』
- 第 527 页的『数据流信息』
- 第 527 页的『插入、更新和删除信息』
- 第 528 页的『块和行标识准备信息』
- 第 529 页的『聚集信息』
- 第 529 页的『并行处理信息』
- 第 531 页的『联合查询信息』
- 第 532 页的『其它信息』

相关参考:

- 第 512 页的『db2expln – SQL 说明』

表存取信息

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

- 存取具体查询表名称:

```
Access Materialized Query 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:

- 存取临时表标识:

```
Access Temp Table  ID = tn
```

其中:

– *ID* 是 db2expln 指定的对应标识

- 存取已声明全局临时表标识:

```
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

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
- 块存取
- 并行扫描
- 扫描伪指令
- 行存取方法
- 锁定意向
- 谓词
- 其它语句

列数

以下语句指示该表的每一行中所用的列数:

```
#Columns = n
```

块存取

以下语句指示表具有一个或多个在其上定义的维块索引:

```
Clustered by Dimension for Block Index Access
```

如果未显示此文本，则未使用 DIMENSION 子句创建该表。

并行扫描

以下语句指示数据库管理器将使用几个子代理进程来并行读取该表:

```
Parallel Scan
```

如果未显示此文本，则该表只能由一个代理进程（或子代理进程）读取。

扫描方向

以下语句指示数据库管理器将按倒序读取行:

```
Scan Direction = Reverse
```

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取:

    ```
    Relation Scan
    |  Prefetch: None
    ```

  – 以下语句指示优化器已预先确定了将预取的页数:

    ```
    Relation Scan
    |  Prefetch: n Pages
    ```

  – 以下语句指示应该预取数据:

    ```
    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 type
    | 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。
    - 索引类型是下列其中一项:

      ```
      Regular Index (Not Clustered)
      Regular Index (Clustered)
      Dimension Block Index
      Composite Dimension Block Index
      ```

      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示索引中的每一列:

      ```
      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键值。不会在扫描中包括排除的键值。下列其中一行将对键的每一部分给出该键的值:

  ```
  n: 'string'
  n: nnn
  n: yyyy-mm-dd
  n: hh:mm:ss
  n: yyyy-mm-dd hh:mm:ss.uuuuuu
  n: NULL
  n: ?
  ```

  如果出现文字字符串，则仅显示前面 20 个字符。如果该字符串长度大于 20 个字符，则在该字符串的最后显示 ...。一些键只有在执行该部分之后才能确定。因此会显示 ? 作为其值。

- 纯索引存取

  如果可以从索引键获得所有需要的列，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取:

  ```
  Index Prefetch: None
  ```

- 以下语句指示应该预取索引页:

  ```
  Index Prefetch: Eligible
  ```

- 以下语句指示将不执行数据页的预取:

  ```
  Data Prefetch: None
  ```

- 以下语句指示应该预取数据页:

  ```
  Data Prefetch: Eligible
  ```

- 如果存在可以传送至“索引管理器”以帮助限定索引条目的谓词，则以下语句用于显示谓词数:

  ```
  Sargable Index Predicate(s)
  |  #Predicates = n
  ```

– 如果正在通过使用在存取方案中先前准备的行标识（RID）来存取限定行，将用下列语句指示它:

  ```
  Fetch Direct Using Row IDs
  ```

  如果表具有为它定义的一个或多个块索引，则块或行标识都可以访问行。这由以下指示:

  ```
  Fetch Direct Using Block or Row IOs
  ```

锁定意向

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:

- 互斥
- 意向互斥
- 无任何意向
- 意向共享
- 共享
- 共享意向互斥
- 超互斥
- 更新

行锁定的可能的值有:

- 互斥
- 下一键互斥（不在 db2expln 输出中显示）
- 无
- 共享
- 下一键共享
- 更新
- 下一键弱互斥
- 弱互斥

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. 以下语句指示将对从分块索引中检索的每个数据块求出的谓词数。
```
Block Predicates(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。谓词的计数不包括下推操作，例如，聚集或排序。
```
Sargable Predicate(s)
|  #Predicates = n
```
3. 以下语句指示一旦返回数据，将求出的谓词数:
```
Residual 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
- 在查询优化过程中，通过添加隐式谓词来变换和扩展
- 在查询优化过程中，被变换和压缩成更少的谓词。

其它表语句

- 以下语句指示将只存取一行:
```
Single Record
```
- 当此表访问所使用的隔离级别与该语句所用的不同时，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，包括:

– 一个程序包是用“可重复读取”绑定的，这影响了引用完整性约束；为检查引用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，以避免对此表挂起不必要的锁定。
– 用“未落实的读”绑定的程序包发出 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序堆内存可用，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
```

相关概念:

- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

临时表信息

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。该标识是引用临时表时为方便起见而由 db2expln 指定的标识。此标识以字母“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进程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进程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声明全局临时表:

```
Insert Into Global Temp Table  ID = ts,tn
```

- 以下语句指示多个子代理进程将并行创建一个已声明全局临时表:

```
Insert Into Shared Global Temp Table  ID = ts,tn
```

- 以下语句指示将创建一个已排序已声明全局临时表:

```
Insert Into Sorted Global Temp Table  ID = ts,tn
```

- 以下语句指示多个子代理进程将并行创建一个已排序已声明全局临时表:

```
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键列数:

  ```
  #Sort Key Columns = n
  ```

  对于排序键中的每一列，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
  ```

- 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，则数据库管理器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。

- 以下语句指示在排序期间将除去重复的值:

  ```
  Duplicate Elimination
  ```

- 如果正在排序中执行聚集，则下列其中一个语句会指示它:

```
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下推操作的表存取之后（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

表函数是用户定义的函数（UDF），它将数据以表的形式返回至语句。表函数由以下语句指示:

```
Access User Defined Table Function
      |  Name = schema.funcname
      |  Specific Name = specificname
      |  SQL Access Level = accesslevel
      |  Language = lang
      |  Parameter Style = parmstyle
      |  Fenced                        Not Deterministic
      |  Called on NULL Input          Disallow Parallel
      |  Not Federated                 Not Threadsafe
```

特定名称唯一标识调用的表函数。其余行详细描述函数的属性。

相关概念:

- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

连接信息

有三种类型的连接:

- 散列连接
- 合并连接
- 嵌套循环连接。

当在一节中执行到要执行连接时，会显示下列其中一个语句:

```
Hash Join
Merge Join
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
Left Outer Merge Join
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，可以构建一个探测表来提高连接的性能。在存取外部表时用以下语句指示探测表的构建:

```
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
```

- 构建探测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*组合内连接*。

如果一个连接涉及两个以上的表，则应从头到尾读取说明步骤。例如，假设说明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 中存在其它行，则转至步骤 4。
6. 如果 Y 中存在其它行，则转至步骤 3。
7. 如果 X 中存在其它行，则转至步骤 2。
8. 如果 W 中存在其它行，则转至步骤 1。

相关概念:

- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

数据流信息

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标识。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
```

相关概念:

- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

插入、更新和删除信息

这些 SQL 语句的说明文本是自我说明的。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: Materialized Query Table = schema.name ID = ts,n
Update: Materialized Query Table = schema.name ID = ts,n
Delete: Materialized Query Table = 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
```

相关概念:

• 第 517 页的『db2expln 和 dynexpln 输出的描述』
• 第 534 页的『db2expln 和 dynexpln 输出的示例』

块和行标识准备信息

对于某些存取方案，如果对限定行和块标识（标识）排序并除去了重复的标识（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标识（如果是索引 AND 运算），则会更有效。正如说明语句所指示，标识准备有下列三种主要用途:

- 以下语句指示“索引 OR 运算”用于准备限定标识的列表:

  ```
  Index ORing Preparation
  Block Index ORing 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标识的一种技术。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。
- 标识准备的另一个用途是准备在列表预取期间要使用的输入数据，如下列任一项所指示:

  ```
  List Prefetch Preparation
  Block 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标识的一种技术。索引 AND 运算的处理从下列其中一个语句开始:

  ```
  Index ANDing
  Block 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标识，并使用位过滤器技术来确定在存取的每个索引中出现的标识。以下语句指示正在为索引 AND 运算处理标识:

  ```
  Index ANDing Bitmap Build Using Row IDs
  Index ANDing Bitmap Probe Using Row IDs
  Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Build Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Block IDs
  Block Index ANDing Bitmap Build and Probe Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs and Build Using Row IDs
  Block Index ANDing Bitmap Probe Using Block IDs
  Block Index ANDing Bitmap Probe Using Row IDs
  ```

  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
  ```

对于任何类型的标识准备，如果可以执行列表预取，则将用以下语句指示它:

```
Prefetch: Enabled
```

相关概念:

• 第 517 页的『db2expln 和 dynexpln 输出的描述』
• 第 534 页的『db2expln 和 dynexpln 输出的示例』

聚集信息

对满足指定标准的那些行执行聚集，如果存在这样的行，则它们是由 SQL 语句谓词提供的。如果要执行某种聚集函数，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
```

谓词聚集指示在实际存取数据时，将聚集操作压入为一个谓词来处理。

在上述任何一种聚集语句之下将是要执行的聚集函数的类型指示:

```
Group By
Column Function(s)
Single Record
```

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。

如果使用谓词聚集，则在出现了聚集的表存取语句之后，将是聚集“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
```

相关概念:

• 第 517 页的『db2expln 和 dynexpln 输出的描述』
• 第 534 页的『db2expln 和 dynexpln 输出的示例』

并行处理信息

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，将使用几个子代理进程同时执行该方案的各个部分。这些子代理进程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，该节被分为几个小节。每个小节被发送至一个或多个节点，以便运行。一个重要的小节是*协调程序小节*。协调程序小节是每个方案中的第一个小节。它首先获得控制权，并负责分发其它小节，然后将结果返回至调用应用程序。

  小节的分发由以下语句指示:

  ```
  Distribute Subsection #n
  ```

  可用八种方式的其中一种来确定接收小节的节点:

  – 以下示例指示将根据列值把小节发送至数据库分区组内的一个节点。

```
Directed by Hash
|  #Columns = n
|  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
```

– 以下示例指示将把小节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
```

– 以下示例指示将把小节发送至与给定数据库分区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
```

– 以下示例指示将把小节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小节。

```
Directed to Single Node
|  Node Number = n
```

– 以下示例之一指示将在协调程序节点上执行小节。

```
Directed to Application Coordinator Node
Directed to Local Coordinator Node
```

– 以下示例指示将把小节发送到已列示的所有节点。

```
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小节。

```
Directed to Any Node
```

• 表队列用于在一个分区数据库环境中的小节之间或一个对称多处理器（SMP）环境中的子代理进程之间移动数据。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:

将所有行发送至协调程序节点:

```
Broadcast to Coordinator Node
```

将所有的行发送至给定小节运行所在的每个数据库分区:

```
Broadcast to All Nodes of Subsection n
```

根据行中的值，将每一行发送至一个数据库分区:

```
Hash 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 Specific Node
```

将每一行发送到随机确定的节点:

```
Send to Random Node
```

– 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下推操作的表存取之后，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
```

这些消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消息:

```
Output Sorted
Output Sorted and Unique
```

这些消息之后是有关用于排序操作的键数的指示。

```
#Key Columns = n
```

对于排序键中的每一列，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，则会显示下列消息:

```
Residual Predicate(s)
|  #Predicates = n
```

• 一个分区数据库环境中的某些小节明确地循环回至该小节的开始处，这由以下语句指示:

```
Jump Back to Start of Subsection
```

相关概念:

• 第 517 页的『db2expln 和 dynexpln 输出的描述』
• 第 534 页的『db2expln 和 dynexpln 输出的示例』

联合查询信息

在联合数据库中执行 SQL 语句需要有对其它数据源执行部分语句的能力。

以下语句指示将要读取的数据源:

```
Ship 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
```

发生在数据源的插入、更新或删除操作将由适当的消息指示:

```
Ship Distributed Insert #n
Ship Distributed Update #n
Ship Distributed Delete #n
```

如果在数据源显式锁定了表，将用以下语句指示它:

```
Ship Distributed Lock Table #n
```

针对数据源的 DDL 语句分割成两部分。在数据源上调用的部分由以下指示:

```
Ship Distributed DDL Statement #n
```

如果联合服务器是分区数据库，那么 DDL 语句的部分必须在目录节点上运行。这由以下指示:

```
Distributed DDL Statement #n Completion
```

单独提供每个分布式子语句的详细信息。分布式语句的选项描述如下:

- 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
  ```

- 如果数据源是关系式的，则子语句的 SQL 显示为:

  ```
  SQL Statement:
       statement
  ```

  非关系数据源用下列项指示:

  ```
  Non-Relational Data Source
  ```

- 子语句中引用的昵称列示如下:

  ```
  Nicknames Referenced:
     schema.nickname ID = n
  ```

  如果数据源是关系式的，昵称的基本表显示为:

  ```
  Base = baseschema.basetable
  ```

  如果数据源是非关系数据源，则昵称的源文件显示为:

  ```
  Source File = filename
  ```

- 在执行子语句之前，如果将值从联合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
  ```

- 在执行子语句之后，如果将值从数据源传送至联合服务器，则值的数目将由下面这一项显示:

  ```
  #Output Columns: n
  ```

相关概念:

- 第 165 页的『分析在何处对联合查询求值的准则』
- 第 517 页的『db2expln 和 dynexpln 输出的描述』

其它信息

- 数据定义语言语句的节将在输出中用下列语句指示:

  ```
  DDL Statement
  ```

  不对 DDL 语句提供其它说明输出。

- 用于可更新的专用寄存器的 SET 语句（如 **CURRENT EXPLAIN SNAPSHOT**）的节将在输出中用以下语句指示:

  ```
  SET Statement
  ```

不对 SET 语句提供其它说明输出。

- 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。如果数据库管理器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，且索引键中的所有列都是 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，将出现以下语句:

```
Positioned Operation
```

如果位置操作是针对联合数据源的，则该语句为:

```
Distributed 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)
```

表构造函数可以用于将一个集合中的值变换成可以传送至后续操作的一系列行。当提示表构造函数输入下一行时，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:
```

条件判定用于实现象 SQL CASE 语句这样的活动，或象引用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，它将需要一个完成阶段。此阶段显示为:

```
Return Data Completion
```

- 如果正在调用存储过程，则将出现下列信息:

```
Call Stored Procedure
|    Name = schema.funcname
|    Specific Name = specificname
|    SQL Access Level = accesslevel
|    Language = lang
|    Parameter Style = parmstyle
|    Expected Result Sets = n
|    Fenced                        Not Deterministic
|    Called on NULL Input          Disallow Parallel
|    Not Federated                 Not Threadsafe
```

- 如果正在释放一个或多个 LOB 定位器，则将出现下列信息:

```
Free LOB Locators
```

相关概念:

- 第 517 页的『db2expln 和 dynexpln 输出的描述』
- 第 534 页的『db2expln 和 dynexpln 输出的示例』

db2expln 和 dynexpln 输出示例

下列输出示例显示为特定环境中的特定查询收集的说明信息。

db2expln 和 dynexpln 输出的示例

以下显示的五个示例可以帮助您了解 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。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。

相关概念:

- 第 517 页的『dynexpln』
- 第 534 页的『示例一: 无并行性』
- 第 536 页的『示例二: 具有分区内并行性的单一分区方案』
- 第 538 页的『示例三: 具有分区间并行性的多分区方案』
- 第 540 页的『示例四: 具有分区间并行性和分区内并行性的多分区方案』
- 第 542 页的『示例五: 联合数据库方案』

相关参考:

- 第 512 页的『db2expln – SQL 说明』

示例一: 无并行性

本示例只请求一个列表，它包含所有职员的姓名、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散列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，故存取方案在连接每个表时对它们进行关系扫描。

```
******************** PACKAGE ***************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:05:00

 Bind Timestamp = 2002-01-04-14.05.00.415403

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 120.518692
Estimated Cardinality = 221.535980

(    6) Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  #Columns = 3
        |  Relation Scan
        |  |  Prefetch: Eligible
        |  Lock Intents
        |  |  Table: Intent Share
        |  |  Row  : Next Key Share
(    6) |  Process Build Table for Hash Join
(    2) Hash Join
        |  Estimated Build Size: 7111
        |  Estimated Probe Size: 9457
(    5) |  Access Table Name = DOOLE.PROJECT  ID = 2,7
        |  |  #Columns = 2
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    5) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 5737
        |  |  Estimated Probe Size: 6421
(    4) |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
        |  |  |  #Columns = 3
        |  |  |  Relation Scan
        |  |  |  |  Prefetch: Eligible
        |  |  |  Lock Intents
        |  |  |  |  Table: Intent Share
        |  |  |  |  Row  : Next Key Share
(    4) |  |  |  Process Probe Table for Hash Join
(    1) Return Data to Application
        |  #Columns = 5

End of section
```

```
Optimizer Plan:

               RETURN
               (   1)
                 |
               HSJOIN
               (   2)
              /      \
        HSJOIN        TBSCAN
        (   3)        (   6)
       /      \          |
   TBSCAN    TBSCAN   Table:
   (   4)    (   5)   DOOLE
     |         |      EMPLOYEE
 Table:      Table:
 DOOLE       DOOLE
 DEPARTMENT  PROJECT
```

该方案的第一部分存取 DEPARTMENT 和 PROJECT 表，并用散列连接来连接它们。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一分区方案

本示例显示与第一个示例相同的 SQL 语句，但是此查询是为一个 4 路 SMP 机器编译的。

```
******************** PACKAGE ***************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:12:38

 Bind Timestamp = 2002-01-04-14.12.38.732627

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 133.934692
Estimated Cardinality = 221.535980

(    2) Process Using 4 Subagents
(    7) |  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
(    7) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 9457
(    6) |  |  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
(    6) |  |  |  Process Build Table for Hash Join
(    4) |  |  Hash Join
        |  |  |  Estimated Build Size: 5737
        |  |  |  Estimated Probe Size: 6421
(    5) |  |  |  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
(    5) |  |  |  |  Process Probe Table for Hash Join
(    2) |  Insert Into Asynchronous Local Table Queue  ID = q1
(    2) Access Local Table Queue  ID = q1  #Columns = 5
(    1) Return Data to Application
        |  #Columns = 5

End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
                LTQ
               (   2)
                 |
               HSJOIN
               (   3)
              /      \
       HSJOIN         TBSCAN
       (   4)         (   7)
      /      \          |
  TBSCAN    TBSCAN   Table:
  (   5)    (   6)   DOOLE
    |         |      EMPLOYEE
 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进程，而在该方案结束时表队列要收集每个子代理进程的工作结果，然后将这些结果返回至应用程序。

示例三：具有分区间并行性的多分区方案

本示例显示与第一个示例相同的 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"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:54:57

 Bind Timestamp = 2002-01-04-14.54.57.033666

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 118.483406
Estimated Cardinality = 474.720032

Coordinator Subsection:(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

Subsection #1:(    8)    Access Table Queue  ID = q2  #Columns = 2
(    3)    Hash Join
           |  Estimated Build Size: 5737
           |  Estimated Probe Size: 8015
(    6)    |  Access Table Queue  ID = q3  #Columns = 3
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5333
           |  |  Estimated Probe Size: 6421
(    5)    |  |  Access Table Name = DOOLE.DEPARTMENT  ID = 2,4
           |  |  |  #Columns = 3
           |  |  |  Relation Scan
           |  |  |  |  Prefetch: Eligible
           |  |  |  Lock Intents
           |  |  |  |  Table: Intent Share
           |  |  |  |  Row  : Next Key Share
```

```
(    5)    |  |  |  Process Probe Table for Hash Join
(    2)    Insert Into Asynchronous Table Queue  ID = q1
           |  Broadcast to Coordinator Node
           |  Rows Can Overflow to Temporary Table

        Subsection #2:
(    9)    Access Table Name = DOOLE.PROJECT  ID = 2,7
           |  #Columns = 2
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    9)    |  Insert Into Asynchronous Table Queue  ID = q2
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    8)    Insert Into Asynchronous Table Queue Completion  ID = q2

Subsection #3:
(    7)    Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
           |  #Columns = 3
           |  Relation Scan
           |  |  Prefetch: Eligible
           |  Lock Intents
           |  |  Table: Intent Share
           |  |  Row  : Next Key Share
(    7)    |  Insert Into Asynchronous Table Queue  ID = q3
           |  |  Hash to Specific Node
           |  |  Rows Can Overflow to Temporary Tables
(    6)    Insert Into Asynchronous Table Queue Completion  ID = q3

End of section


Optimizer Plan:

               RETURN
               (   1)
                 |
               BTQ                 (   2)
                 |
               HSJOIN
               (   3)
              /      \
       HSJOIN         DTQ
       (   4)        (   8)
      /      \          |
  TBSCAN      DTQ     TBSCAN
  (   5)     (   6)   (   9)
    |           |        |
 Table:      TBSCAN   Table:
 DOOLE       (   7)   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，但是该节分成四个小节。这些小节具有下列任务:

- **协调程序小节**。此小节协调其它小节。在此方案中，它导致分发其它小节，然后使用表队列来收集要返回至应用程序的结果。
- **小节 #1**。此小节扫描表队列 q2，并使用散列连接将它与表队列 q3 的数据连接。接着第二个散列连接将表 DEPARTMENT 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小节。

- **小节 #2**。此小节扫描 PROJECT 表，然后将结果散列到特定节点。这些结果由小节 #1 读取。
- **小节 #3**。此小节扫描 EMPLOYEE 表，然后将结果散列到特定节点。这些结果由小节 #1 读取。

示例四：具有分区间并行性和分区内并行性的多分区方案

本示例显示与第一个示例相同的 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"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/04
 Prep Time = 14:58:35

 Bind Timestamp = 2002-01-04-14.58.35.169555

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = Yes
 Intra-Partition Parallel = Yes (Bind Degree = 4)

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno

Intra-Partition Parallelism Degree = 4

Estimated Cost        = 145.198898
Estimated Cardinality = 474.720032

Coordinator Subsection:(-----)    Distribute Subsection #2
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #3
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(-----)    Distribute Subsection #1
           |  Broadcast to Node List
           |  |  Nodes = 10, 33, 55
(    2)    Access Table Queue  ID = q1  #Columns = 5
(    1)    Return Data to Application
           |  #Columns = 5

Subsection #1:(    3)    Process Using 4 Subagents
(   10)    |  Access Table Queue  ID = q3  #Columns = 2
(    4)    |  Hash Join
           |  |  Estimated Build Size: 5737
           |  |  Estimated Probe Size: 8015
(    7)    |  |  Access Table Queue  ID = q5  #Columns = 3
(    5)    |  |  Hash Join
```

```
                |  |  |  Estimated Build Size: 5333
                |  |  |  Estimated Probe Size: 6421
(    6)         |  |  |  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
(    6)         |  |  |  |  Process Probe Table for Hash Join
(    3)         |  Insert Into Asynchronous Local Table Queue  ID = q2
(    3)      Access Local Table Queue  ID = q2  #Columns = 5
(    2)      Insert Into Asynchronous Table Queue  ID = q1
             |  Broadcast to Coordinator Node
             |  Rows Can Overflow to Temporary Table

        Subsection #2:
(   11)      Process Using 4 Subagents
(   12)      |  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
(   11)      |  Insert Into Asynchronous Local Table Queue  ID = q4
(   11)      Access Local Table Queue  ID = q4  #Columns = 2
(   10)      Insert Into Asynchronous Table Queue  ID = q3
             |  Hash to Specific Node
             |  Rows Can Overflow to Temporary Tables

        Subsection #3:
(    8)      Process Using 4 Subagents
(    9)      |  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
(    8)      |  Insert Into Asynchronous Local Table Queue  ID = q6
(    8)      Access Local Table Queue  ID = q6  #Columns = 3
(    7)      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)
                 |
               HSJOIN
               (   4)
              /      \
        HSJOIN        DTQ
        (   5)       (  10)
```

```
        /       \         |
    TBSCAN     DTQ      LTQ
    (   6)    (   7)   (  11)
      |         |         |
  Table:       LTQ     TBSCAN
  DOOLE       (   8)   (  12)
  DEPARTMENT    |         |
              TBSCAN   Table:
              (   9)   DOOLE
                |      PROJECT
             Table:
             DOOLE
             EMPLOYEE
```

此方案与第三个示例中的方案相似，不同的是多个子代理进程执行每个小节。另外，在每个小节结束时，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进程的结果。

示例五：联合数据库方案

本示例显示与第一个示例相同的 SQL 语句，但已在一个联合数据库上编译此查询，在这个联合数据库中，表 DEPARTMENT 和 PROJECT 在数据源上，而表 EMPLOYEE 在联合服务器上。

```
******************** PACKAGE ***************************************

Package Name = "DOOLE"."EXAMPLE"  Version = ""

 Prep Date = 2002/01/11
 Prep Time = 13:52:48

 Bind Timestamp = 2002-01-11-13.52.48.325413

 Isolation Level          = Cursor Stability
 Blocking                 = Block Unambiguous Cursors
 Query Optimization Class = 5

 Partition Parallel       = No
 Intra-Partition Parallel = No

 SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "DOOLE"

-------------------- SECTION ---------------------------------------
Section = 1


SQL Statement:
  DECLARE EMPCUR CURSOR
  FOR
     SELECT e.lastname, e.job, d.deptname, d.location, p.projname
     FROM employee AS e, department AS d, project AS p
     WHERE e.workdept = d.deptno AND e.workdept = p.deptno


Estimated Cost        = 1804.625000
Estimated Cardinality = 112000.000000

(    7) Ship Distributed Subquery #2
        |  #Columns = 2
(    2) Hash Join
        |  Estimated Build Size: 48444
        |  Estimated Probe Size: 232571
(    6) |  Access Table Name = DOOLE.EMPLOYEE  ID = 2,5
        |  |  #Columns = 3
```

```
        |  |  Relation Scan
        |  |  |  Prefetch: Eligible
        |  |  Lock Intents
        |  |  |  Table: Intent Share
        |  |  |  Row  : Next Key Share
(    6) |  |  Process Build Table for Hash Join
(    3) |  Hash Join
        |  |  Estimated Build Size: 7111
        |  |  Estimated Probe Size: 64606
(    4) |  |  Ship Distributed Subquery #1
        |  |  |  #Columns = 3
(    1) Return Data to Application
        |  #Columns = 5

Distributed Substatement #1:
(    4) Server: REMOTE  (DB2/UDB 8.1)
SQL Statement:

             SELECT A0."DEPTNO", A0."DEPTNAME", A0."LOCATION"
             FROM "DOOLE"."DEPARTMENT" A0

        Nicknames Referenced:
           DOOLE.DEPARTMENT  ID = 32768  Base = DOOLE.DEPARTMENT
        #Output Columns = 3

Distributed Substatement #2:
(    7) Server: REMOTE  (DB2/UDB 8.1)
SQL Statement:

             SELECT A0."DEPTNO", A0."PROJNAME"
             FROM "DOOLE"."PROJECT" A0

        Nicknames Referenced:
           DOOLE.PROJECT  ID = 32769  Base = DOOLE.PROJECT
        #Output Columns = 2

End of section

Optimizer Plan:

                RETURN
                (   1)
                  |
                HSJOIN
                (   2)
               /      \
        HSJOIN          SHIP
        (   3)         (   7)
       /      \           |
   SHIP      TBSCAN   Nickname:
  (   4)     (   6)   DOOLE
    |          |      PROJECT
 Nickname:   Table:   DOOLE       DOOLE
 DEPARTMENT  EMPLOYEE
```

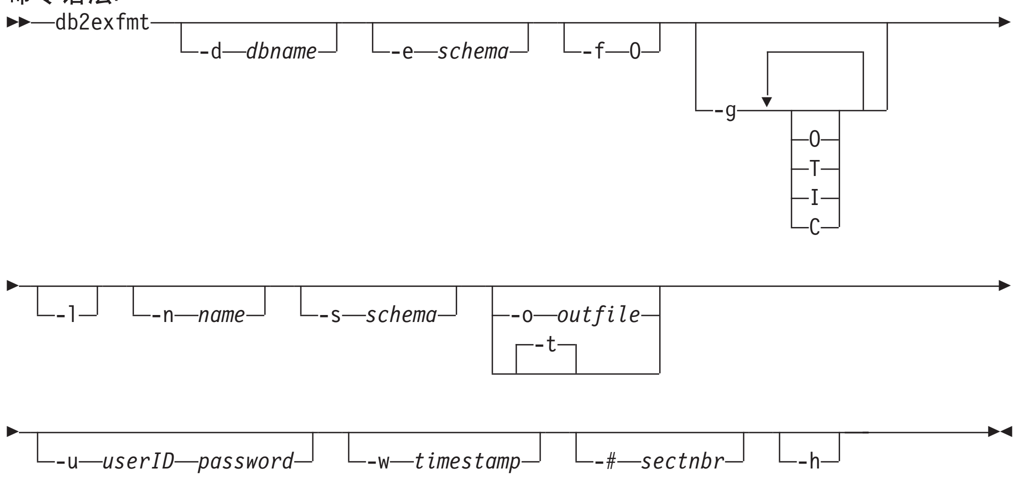
此方案与第一个示例中的方案有完全相同的块，但有两个表的数据来自数据源。这两个表是通过分布式子查询存取的，在本案例中，这些分布式子查询只是从那些表中选择所有行。在将数据返回至联合服务器之后，它便与来自本地表的数据连接到一起。

附录 D. db2exfmt – 说明表格式化

可以使用 db2exfmt 工具来格式化说明表的内容。此工具位于实例 `sqllib` 目录的 `misc` 子目录中。

要使用该工具，需要有要格式化的说明表的读存取权。

命令语法:



命令参数:

-d *dbname*
: 包含程序包的数据库的名称。

-e *schema*
: 说明表的模式。

-f
: 格式化标志。在此发行版中，唯一支持的值是 `O`（运算符总结）。

-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*
: “说明”请求的源名（SOURCE_NAME）。

-s *schema*
: “说明”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 *userID password*
当与一个数据库连接时，使用提供的用户标识和密码。

用户标识和密码都必须符合命名约定，且是数据库可识别的。

-w *timestamp*
说明时间戳记。指定 `-1` 来获取最新的说明请求。

-# *sectnbr*
源中的节号。要请求所有的节，指定零。

-h 显示帮助信息。当指定了此选项时，其它所有的选项都会被忽略，且只显示帮助信息。

使用注意事项:

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。

如果未提供说明表模式，则环境变量 **USER** 的值就被用作缺省值。如果找不到此变量，则会提示用户提供说明表的模式。

源名、源模式和说明时间戳记可用 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。对于最新的说明语句，可将说明时间指定为 `-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。

相关概念:

- 第 172 页的『说明工具』
- 第 173 页的『使用说明信息的准则』
- 第 179 页的『捕获说明信息的准则』

附录 E. 使用 db2adutl 命令以及 logarchopt1 和 vendoropt 数据库配置参数来进行跨节点恢复

下列示例说明如何使用 db2adutl 命令以及 *logarchopt1* 和 *vendoropt* 数据库配置参数来执行跨节点恢复。

在下列示例中，计算机 1 名为 bar，它正在运行 AIX。此机器的所有者是 roecken。bar 上的数据库名为 zample。计算机 2 名为 dps。此机器也在运行 AIX 并由 regress9 所有

PASSWORDACCESS = generate:

计算机 1:

1. 设置数据库以便将日志归档至 TSM。更新 zample 数据库的数据库配置参数 *logarchmeth1*:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   将返回以下信息:

   ```
   成功完成 DB20000I UPDATE DATABASE CONFIGURATION 命令。
   ```

   注: 在更新数据库配置之前，可能必须对数据库进行脱机备份。
2. 对数据库进行联机备份:

   ```
   db2 backup db zample online use tsm
   ```

   将返回以下信息:

   ```
   备份成功。此备份映像的时间戳记是: 20040216151025
   ```
3. 连接至 zample 数据库，然后在该数据库中创建一个表。
4. 将数据装入新表中。在此示例中，表名为 a，从定界 ASCII 文件中装入的数据为 mr。指定了 COPY YES 选项来生成装入的数据的副本，并且 USE TSM 选项指定数据的副本存储在 Tivoli Storage Manager 上。

   注: 仅当启用了数据库进行前滚恢复时才能指定 COPY YES 选项；即，必须对数据库启用 *logretain* 和／或 *userexit* 数据库配置参数。

   ```
   bar:/home/roecken> db2 load from mr of del modified by noheader replace into a
   copy yes use tsm
   ```

   实用程序返回了一系列消息来指示它的进度:

   ```
     SQL3109N  实用程序正开始从“/home/roecken/mr”文件中装入数据。

   SQL3500W  实用程序在“02/16/2004 15:12:13.392633”时进入“装入”阶段。
   SQL3519W  开始装入一致点。输入记录数 = “0”。

   SQL3520W  装入一致点成功。

   SQL3110N  实用程序已完成处理。从输入文件中读取了“1”行。
   SQL3519W  开始装入一致点。输入记录数 = “1”。

   SQL3520W  装入一致点成功。
   ```



```
SQL3515W  实用程序在“02/16/2004 15:12:13.445718”时已完成“装入”阶段。
读取的行数         = 1
跳过的行数         = 0
装入的行数         = 1
拒绝的行数         = 0
删除的行数         = 0
落实的行数         = 1
```

现在，在 TSM 上应该具有一个备份映像、一个装入副本映像和一个日志文件。可以按如下所示对 zample 数据库运行查询:

```
bar:/home/roecken> db2adutl query db zample
```

将返回以下信息:

```
正在检索 FULL DATABASE BACKUP 信息。
    1 时间: 20040216151025  最早的日志: S0000000.LOG  数据库分区号: 0    会话数: 1
正在检索 INCREMENTAL DATABASE BACKUP 信息。
  找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像


正在检索 DELTA DATABASE BACKUP 信息。
  找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像


正在检索 TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 TABLESPACE BACKUP 映像


正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像


正在检索 DELTA TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像


正在检索 LOAD COPY 信息。
    1 时间:  20040216151213


正在检索 LOG ARCHIVE 信息。
   日志文件: S0000000.LOG, 链号: 0, 数据库分区号: 0 生成时间: 2004-02-16-15.10.38
```

5. 要启用跨节点恢复，必须为另一个节点和帐户提供对 bar 计算机上的对象的存取权。在此示例中，对 dps 节点和 regress9 用户提供了存取权。

   ```
   bar:/home/roecken> db2adutl grant user regress9 on nodename dps for db zample
   ```

   将返回以下信息:

   ```
   成功添加了 regress9 存取 dps 节点上的 ZAMPLE 的许可权。
   ```

   要查询 **db2adutl grant** 操作的结果，发出以下命令:

   ```
   bar:/home/roecken> db2adutl queryaccess
   ```

   将返回以下信息:

   ```
   节点                 用户名              数据库名称    类型
   -------------------------------------------------------------
   DPS                  regress9            ZAMPLE         A
   -------------------------------------------------------------
    存取类型:   B - 备份映像   L - 日志   A - 同时使用这两种存取类型
   ```

PASSWORDACCESS = 生成环境:

计算机 2:

尚未设置第 2 台计算机 dps。在 dps 上对 zample 数据库的 **db2adutl** 查询返回了下列结果:

```
dps:/home/regress9> db2adutl query db zample
--- 数据库目录是空的 ---
警告: 在 ADSM 服务器上 DB2 没有创建文件空间
警告: 在 ADSM 中找不到任何别名的 DB2 备份映像。


dps:/home/regress9> db2adutl query db zample nodename bar owner roecken
--- 数据库目录是空的 ---

对 ZAMPLE 数据库的查询


正在检索 FULL DATABASE BACKUP 信息。
    1 时间: 20040216151025  最早的日志: S0000000.LOG  数据库分区号: 0    会话数: 1
 正在检索 INCREMENTAL DATABASE BACKUP 信息。
  找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像


 正在检索 DELTA DATABASE BACKUP 信息。
  找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像


 正在检索 TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 TABLESPACE BACKUP 映像


 正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像


 正在检索 DELTA TABLESPACE BACKUP 信息。
  找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像


 正在检索 LOAD COPY 信息。
    1 时间:  20040216151213


 正在检索 LOG ARCHIVE 信息。
   日志文件: S0000000.LOG, 链号: 0, 数据库分区号: 0 生成时间: 2004-02-16-15.10.38
```

在 dps 计算机上尚不存在 zample 数据库。

1. 将 zample 数据库复原至 dps 计算机:

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
    "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   将返回以下信息:

   ```
   DB20000I  成功完成 RESTORE DATABASE 命令。
   ```

   注: 如果 dps 上已经存在 zample 数据库, 则将省略 OPTIONS 参数, 并且将使用数据库配置参数 *vendoropt*。此配置参数会覆盖备份或复原操作的 OPTIONS 参数。

   对 zample 数据库的前滚操作将失败, 原因是前滚实用程序找不到日志文件。前滚操作如下所示:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

将返回以下错误:

```
SQL4970N  数据库“ZAMPLE”上的前滚恢复不能达到指定的停止点（日志结束或时间点），
原因是在节点“0”上丢失了日志文件。
```

2. 要强制前滚实用程序查找另一台机器上的日志文件，必须配置正确的 *logarchopt* 值，在此情况下，为 *logarchopt1* 数据库配置参数:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

3. 要使前滚实用程序能够使用装入副本映像，还必须设置 *vendoropt* 数据库配置参数:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

4. 现在，可以对 zample 数据库进行前滚了:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

将返回以下信息:

```
                                 前滚状态

输入数据库别名                    = zample
返回了状态的节点数                = 1

节点号                            = 0
前滚状态                          = 未暂挂
要读取的下一个日志文件            =
已处理的日志文件                  = S0000000.LOG - S0000000.LOG
最后落实的事务                    = 2004-02-16-20.10.38.000000

DB20000I  成功完成 ROLLFORWARD 命令。
```

PASSWORDACCESS = 提示环境:

在“提示”环境中，需要额外的信息，尤其是创建了对象的机器的 TSM 节点名和密码。

对于 **db2adutl**，更新 dsm.sys 文件（在基于 Windows 的平台上，更新 *dsm.opt* 文件），并将 NODENAME bar（因为 bar 是源计算机的名称）添加至服务器子句:

```
dps:/home/regress9> db2adutl query db zample nodename bar owner roecken password *******
```

将返回以下信息:

```
    对 ZAMPLE 数据库的查询


正在检索 FULL DATABASE BACKUP 信息。
    1 时间: 20040216151025  最早的日志: S0000000.LOG  数据库分区号: 0    会话数: 1
  正在检索 INCREMENTAL DATABASE BACKUP 信息。
    找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像


  正在检索 DELTA DATABASE BACKUP 信息。
    找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像


  正在检索 TABLESPACE BACKUP 信息。
    找不到 ZAMPLE 的 TABLESPACE BACKUP 映像


  正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
    找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像


  正在检索 DELTA TABLESPACE BACKUP 信息。
    找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像
```

```
正在检索 LOAD COPY 信息。
    1 时间:  20040216151213


正在检索 LOG ARCHIVE 信息。
   日志文件: S0000000.LOG, 链号: 0, 数据库分区号: 0 生成时间: 2004-02-16-15.10.38
```

1. 如果该数据库不存在，则创建空的 zample 数据库。如果 zample 数据库已存在，则可以跳过此步骤和用来更新数据库配置的下面两个步骤。

   ```
   dps:/home/regress9> db2 create db zample
   ```

2. 更新 zample 数据库的数据库配置参数 *tsm_nodename*:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

3. 更新 zample 数据库的数据库配置参数 *tsm_password*:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password ********
   ```

4. 复原 zample 数据库:

   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   成功完成复原操作，但是发出了一条警告:

   ```
   SQL2540W  复原成功，但是在以“无中断”方式进行处理时，
   在“数据库复原”期间遇到了警告“2523”。
   ```

   此时，前滚实用程序又找不到正确的日志文件:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   将返回以下错误消息:

   ```
   SQL1268N  由于检索节点“0”上的数据库“ZAMPLE”的日志文件“S0000000.LOG”时
   发生错误“-2112880618”，前滚恢复已停止。
   ```

5. 因为数据库复原操作替换了数据库配置文件，所以必须将 TSM 数据库配置值设置为正确的值。首先，必须复位 *tsm_nodename* 配置参数:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
   ```

6. 必须复位 *tsm_password* 数据库配置参数:

   ```
   dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
   ```

7. 必须复位 *logarchopt1* 数据库配置参数，以便前滚实用程序可以找到正确的日志文件:

   ```
   dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-fromnode=bar -fromowner=roecken'"
   ```

8. 还必须复位 *vendoropt* 数据库配置参数，以便还可以使用装入恢复文件:

   ```
   dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-fromnode=bar -fromowner=roecken'"
   ```

9. 当设置数据库配置参数时，可以对数据库进行前滚:

   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   对 zample 数据库运行 ROLLFORWARD QUERY STATUS 命令将显示以下结果:

   ```
                                   前滚状态

   输入数据库别名                   = zample
   返回了状态的节点数               = 1
   ```

```
节点号                              = 0
前滚状态                            = 未暂挂
要读取的下一个日志文件              =
已处理的日志文件                    = S0000000.LOG - S0000000.LOG
最后落实的事务                      = 2004-02-16-20.10.38.000000

DB20000I  成功完成 ROLLFORWARD 命令。
```

相关参考:

- 『db2adutl - Managing DB2 objects within TSMCommand』（*Command Reference*）
- 第 367 页的『logarchopt1 － 主日志归档选项』
- 第 373 页的『vendoropt － 供应商选项』

附录 F. DB2 通用数据库技术信息

DB2 文档和帮助

DB2® 技术信息可通过下列工具和方法获得:

- DB2 信息中心
  - 主题
  - DB2 工具的帮助
  - 样本程序
  - 教程
- 可下载的 PDF 文件、CD 上的 PDF 文件和印刷书籍
  - 指南
  - 参考手册
- 命令行帮助
  - 命令帮助
  - 消息帮助
  - SQL 状态帮助
- 已安装的源代码
  - 样本程序

可以在线访问 ibm.com® 上的其它 DB2 Universal Database™（DB2 通用数据库）技术信息，例如，技术说明、白皮书和 Redbooks™（红皮书）。访问位于以下网址的 DB2 信息管理软件资料库站点：www.ibm.com/software/data/pubs/。

DB2 文档更新

IBM® 可能会定期提供 DB2 信息中心的文档修订包和其它文档更新。如果访问 http://publib.boulder.ibm.com/infocenter/db2help/ 网址中的 DB2 信息中心，则将始终可以查看最新的信息。如果本地安装了 DB2 信息中心，则需要手工安装所有更新才能查看它们。文档更新允许您在新信息可供使用时更新从 *DB2 信息中心 CD* 安装的信息。

信息中心的更新比 PDF 或硬拷贝书籍的更新要频繁。要获得最新的 DB2 技术信息，一提供文档更新时就安装它们，或者访问 www.ibm.com 站点上的 DB2 信息中心。

相关概念:

- 『CLI sample programs』（*CLI Guide and Reference, Volume 1*）
- 『Java 样本程序』（*《应用程序开发指南：构建和运行应用程序》*）
- 第 554 页的『DB2 信息中心』

相关任务:

- 第 569 页的『从 DB2 工具调用上下文帮助』
- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 570 页的『从命令行处理器调用消息帮助』

- 第 570 页的『从命令行处理器调用命令帮助』
- 第 571 页的『从命令行处理器调用 SQL 状态帮助』

相关参考:

- 第 563 页的『DB2 PDF 和印刷文档』

DB2 信息中心

DB2® 信息中心使您可以访问充分利用 DB2 系列产品（包括 DB2 Universal Database™（DB2 通用数据库）、DB2 Connect™、DB2 Information Integrator 和 DB2 Query Patroller™）所需的所有信息。 DB2 信息中心还包含主要的 DB2 功能部件和组件（包括复制、数据仓储和 DB2 extender）的信息。

如果是在 Mozilla 1.0（或更新版本）或 Microsoft® Internet Explorer 5.5（或更新版本）中查看的话，则 DB2 信息中心具有下列功能部件。某些功能部件需要您启用对 JavaScript™ 的支持:

灵活安装选项
: 可选择使用最适合您的需要的选项来查看 DB2 文档:
- 要轻松确保文档始终是最新的，可直接从 IBM® Web 站点上的 DB2 信息中心访问所有文档，网址为: http://publib.boulder.ibm.com/infocenter/db2help/
- 要将更新工作量减至最少并使网络通信保持在内部网内，可将 DB2 文档安装在内部网上的单台服务器上
- 要使您有最大的灵活性并减少对网络连接的依赖，可将 DB2 文档安装在您自己的计算机上

搜索 可通过在**搜索**文本字段中输入搜索术语来搜索 DB2 信息中心中的所有主题。可通过用引号将术语括起来以检索确定匹配项，还可以使用通配运算符（* 和 ?）和布尔运算符（AND、NOT 和 OR）细化搜索。

面向任务的目录
: 可从单个目录查找 DB2 文档中的主题。目录主要是按想要执行的任务的种类组织的，同时也包括有关产品概述、目标、参考信息、索引和词汇表的条目。
- 产品概述描述 DB2 系列中的可用产品之间的关系、其中每个产品提供的功能部件以及其中每个产品的最新发行信息。
- 目标类别（例如，安装、管理和开发）包括一些主题，这些主题使您能够快速地完成任务并且更好地理解完成这些任务的背景信息。
- 参考主题提供有关主题的详细信息，包括语句和命令语法、消息帮助以及配置参数。

显示目录中的当前主题
: 可通过单击目录框架中的**刷新／显示当前主题**按钮或通过单击内容框架中的**在目录中显示**按钮来显示当前主题在目录中的位置。如果访问了指向若干个文件中的相关主题的若干个链接，或者是从搜索结果到达主题的，此功能会非常有用。

索引 可从索引访问全部文档。索引是按索引项的拼音顺序组织的。

词汇表 可使用词汇表来查找在 DB2 文档中使用的术语的定义。词汇表是按词汇表术语的拼音顺序组织的。

集成的本地化信息

DB2 信息中心以您在浏览器首选项中设置的首选语言显示信息。如果主题不是以首选语言提供的，则 DB2 信息中心将显示该主题的英文版。

有关 iSeries™ 技术信息，参阅 IBM eServer™ iSeries 信息中心，网址为 www.ibm.com/eserver/iseries/infocenter/。

相关概念:

- 第 555 页的『DB2 信息中心安装方案』

相关任务:

- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 562 页的『以首选语言显示 DB2 信息中心中的主题』
- 第 561 页的『调用 DB2 信息中心』
- 第 557 页的『使用"DB2 安装"向导来安装 DB2 信息中心（UNIX）』
- 第 559 页的『使用"DB2 安装"向导来安装 DB2 信息中心（Windows）』

DB2 信息中心安装方案

对于如何访问 DB2® 信息，不同的工作环境可有不同的需求。可以使用三种方法访问 DB2 信息中心：从 IBM® Web 站点访问、从组织网络的服务器访问或从安装在计算机上的版本访问。在所有三种情况中，文档都包含在 DB2 信息中心中，DB2 信息中心是基于主题的信息的结构化 Web，可使用浏览器来查看。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 信息中心。但是，如果想要从内部网服务器或从您自己的计算机访问 DB2 信息中心，必须使用产品"介质包"中的 DB2 信息中心 CD 来安装 DB2 信息中心。参阅下面的用于访问 DB2 文档的选项的总结及三个方案来帮助确定访问 DB2 信息中心时使用哪个方法最适合您和您的工作环境，以及可能需要考虑哪些安装问题。

用于访问 DB2 文档的选项的总结:

下表提供了有关哪些选项可能用来在您的工作环境中访问 DB2 信息中心中的 DB2 产品文档的建议。

因特网访问	内部网访问	建议
是	是	访问 IBM Web 站点上的 DB2 信息中心，或者访问安装在内部网服务器上的 DB2 信息中心。
是	否	访问 IBM Web 站点上的 DB2 信息中心。
否	是	访问安装在内部网服务器上的 DB2 信息中心。
否	否	访问本地计算机上的 DB2 信息中心。

方案：访问您的计算机上的 DB2 信息中心:

Tsu-Chen 在一个小镇上开了一家工厂，而小镇没有本地 ISP，所以他不能访问因特网。他购买了 DB2 Universal Database™（DB2 通用数据库）来管理库存、产品订单、银行帐户信息和业务开销。由于以前从未使用过 DB2 产品，所以 Tsu-Chen 需要从 DB2 产品文档了解产品的使用方法。

使用典型安装选项在计算机上安装了 DB2 通用数据库之后，Tsu-Chen 尝试访问 DB2 文档。但是，浏览器显示一条错误消息，提示找不到他尝试打开的页。Tsu-Chen 查阅了 DB2 产品的安装手册，发现如果想要访问计算机上的 DB2 文档，则必须安装 DB2 信息中心。他在介质包中找到了 *DB2 信息中心 CD* 并安装了它。

Tsu-Chen 现在能够从操作系统的应用程序启动程序访问 DB2 信息中心，也能够了解如何使用 DB2 产品来增加业务的成功率。

方案：访问 IBM Web 站点上的 DB2 信息中心：

Colin 是培训公司的一名信息技术顾问。他精通数据库技术和 SQL 并对全北美使用 DB2 通用数据库的企业提供有关这些主题的讲座。Colin 的部分讲座包括将 DB2 文档用作教学工具。例如，在讲授有关 SQL 的课程时，Colin 使用有关 SQL 的 DB2 文档作为教授数据库查询的基本和高级语法的方法。

Colin 授课的大部分企业都访问因特网。当 Colin 安装了 DB2 通用数据库的最新版本时，他会决定配置其移动式计算机以访问 IBM Web 站点上的 DB2 信息中心。此配置允许 Colin 在授课期间在线访问最新的 DB2 文档。

但是，有时在旅行时 Colin 不能访问因特网。这对他来说是个问题，尤其是在他需要访问 DB2 文档来备课时。为避免类似情况，Colin 在他的移动式计算机上安装了 DB2 信息中心的副本。

Colin 可以随时很方便的获得 DB2 文档的副本。通过使用 **db2set** 命令，他可以根据所在位置很容易地将其移动式计算机上的注册表变量配置为访问 IBM Web 站点上或他的移动式计算机上的 DB2 信息中心。

方案：访问内部网服务器上的 DB2 信息中心：

Eva 是一家人寿保险公司的高级数据库管理员。她的管理职责包括在公司的 UNIX® 数据库服务器上安装和配置 DB2 通用数据库的最新版本。她的公司最近通知职员，为安全起见，在工作时间将不提供因特网访问。因为公司具有联网环境，所以 Eva 决定在内部网服务器上安装 DB2 信息中心，以便公司中经常使用公司数据仓库的所有职员（销售代表、销售经理和业务分析人员）都可以访问 DB2 文档。

Eva 会指导她的数据库小组使用响应文件在所有职员的计算机上安装 DB2 通用数据库的最新版本，以确保每台计算机都配置为使用内部网服务器的主机名和端口号来访问 DB2 信息中心。

但是，Eva 的小组中的初级数据库管理员 Migual 误解了 Eva 的意思，她在一些职员的计算机上安装了 DB2 信息中心的副本，但没有将 DB2 通用数据库配置为访问内部网服务器上的 DB2 信息中心。为了更正这种情况，Eva 告诉 Migual 使用 **db2set** 命令在这些计算机中的每一台上更改 DB2 信息中心注册表变量（DB2_DOCHOST 表示主机名，而 DB2_DOCPORT 表示端口号）。现在，该网络上的所有相应计算机都可以访问 DB2 信息中心，而且职员可在 DB2 文档中查找有关 DB2 问题的答案。

相关概念：

- 第 554 页的『DB2 信息中心』

相关任务：

- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』

- 第 557 页的『使用“DB2 安装”向导来安装 DB2 信息中心（UNIX）』
- 第 559 页的『使用“DB2 安装”向导来安装 DB2 信息中心（Windows）』
- 『设置访问 DB2 信息中心的位置：公共 GUI 帮助』

相关参考:

- 『db2set - DB2 Profile Registry Command』（*Command Reference*）

使用“DB2 安装”向导来安装 DB2 信息中心（UNIX）

可使用三种方法访问 DB2 产品文档：从 IBM Web 站点访问、从内部网服务器访问或从计算机上安装的版本访问。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 文档。如果想要从内部网服务器或您自己的计算机访问 DB2 文档，必须从 *DB2 信息中心 CD* 安装文档。使用“DB2 安装”向导，您可以定义安装首选项并在使用 UNIX 操作系统的计算机上安装 DB2 信息中心。

先决条件:

本节列示了在 UNIX 计算机上安装 DB2 信息中心的硬件、操作系统、软件以及通信需求。

- **硬件需求**

  需要下列其中一种处理器:
  - PowerPC（AIX）
  - HP 9000（HP-UX）
  - Intel 32 位（Linux）
  - Solaris UltraSPARC 计算机（Solaris Operating Environment）
- **操作系统需求**

  需要下列其中一个操作系统:
  - IBM AIX 5.1（在 PowerPC 上）
  - HP-UX 11i（在 HP 9000 上）
  - Red Hat Linux 8.0（在 Intel 32 位上）
  - SuSE Linux 8.1（在 Intel 32 位上）
  - Sun Solaris V8（在 Solaris Operating Environment UltraSPARC 计算机上）

  注: DB2 信息中心在支持 DB2 客户机的一部分 UNIX 操作系统上运行。因此，建议从 IBM Web 站点访问 DB2 信息中心，或者在内部网服务器上安装并访问 DB2 信息中心。
- **软件需求**
  - 支持下列浏览器:
    - Mozilla V1.0 或更高版本
- “DB2 安装”向导是一个图形安装程序。必须实现能够呈示图形用户界面的 X Window System 软件才能使“DB2 安装”向导在计算机上运行。必须确保正确导出了显示内容，才能运行“DB2 安装”向导。例如，在命令提示符处输入以下命令:

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **通信需求**
  - TCP/IP

过程:

要使用“DB2 安装”向导安装 DB2 信息中心:

1. 登录系统。
2. 在系统上放入并装上 DB2 信息中心产品 CD。
3. 通过输入以下命令切换到装上 CD 的目录:

   ```
   cd /cd
   ```

   其中 */cd* 表示 CD 的安装点。
4. 输入 **./db2setup** 命令来启动“DB2 安装”向导。
5. “IBM DB2 安装启动板”打开。要直接进至 DB2 信息中心的安装，单击**安装产品**。联机帮助可指导您完成其余步骤。要调用联机帮助，单击**帮助**。可随时单击**取消**来结束安装。
6. 在**选择您想要安装的产品**页中，单击**下一步**。
7. 在**欢迎使用“DB2 安装”向导**页中，单击**下一步**。“DB2 安装”向导将指导您完成程序安装过程。
8. 要继续安装，必须接受许可协议。在**许可协议**页中，选择**我接受许可协议中的条款**，然后单击**下一步**。
9. 在**选择安装操作**页中，选择**在此计算机上安装 DB2 信息中心**。如果想要在稍后使用响应文件在此计算机或其它计算机上安装 DB2 信息中心，则选择**将设置保存在响应文件中**。单击**下一步**。
10. 在**选择要安装的语言**页中，选择将用来安装 DB2 信息中心的语言。单击**下一步**。
11. 在**指定 DB2 信息中心端口**页中，配置 DB2 信息中心的入局通信。单击**下一步**继续安装。
12. 在**开始复制文件**页中复查您作出的安装选择。要更改任何设置，单击**上一步**。单击**安装**以将 DB2 信息中心文件复制到计算机上。

还可以使用响应文件安装 DB2 信息中心。

缺省情况下，安装日志 `db2setup.his`、`db2setup.log` 和 `db2setup.err` 位于 `/tmp` 目录中。

`db2setup.log` 文件会捕获所有 DB2 产品安装信息（包括错误）。`db2setup.his` 文件会记录计算机上的所有 DB2 产品安装。DB2 将 `db2setup.log` 文件追加至 `db2setup.his` 文件。`db2setup.err` 文件捕获 Java 返回的任何错误输出，例如，异常和陷阱信息。

当安装完成后，DB2 信息中心将安装在下列其中一个目录中，这取决于您的 UNIX 操作系统:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris Operating Environment: /opt/IBM/db2/V8.1

相关概念:

- 第 554 页的『DB2 信息中心』
- 第 555 页的『DB2 信息中心安装方案』

相关任务:

- 『使用响应文件安装 DB2（UNIX）』（*《安装与配置补充手册》*）
- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 562 页的『以首选语言显示 DB2 信息中心中的主题』
- 第 561 页的『调用 DB2 信息中心』
- 第 559 页的『使用“DB2 安装”向导来安装 DB2 信息中心（Windows）』

使用“DB2 安装”向导来安装 DB2 信息中心（Windows）

可使用三种方法访问 DB2 产品文档：从 IBM Web 站点访问、从内部网服务器访问或从计算机上安装的版本访问。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 文档。如果想要从内部网服务器或您自己的计算机访问 DB2 文档，必须从 *DB2 信息中心 CD* 安装 DB2 文档。使用“DB2 安装”向导，可以定义安装首选项并在使用 Windows 操作系统的计算机上安装 DB2 信息中心。

先决条件:

本节列示了在 Windows 上安装 DB2 信息中心的硬件、操作系统、软件以及通信需求。

- **硬件需求**

  需要下列其中一种处理器:

  – 32 位计算机：奔腾或与奔腾兼容的 CPU
- **操作系统需求**

  需要下列其中一个操作系统:

  – Windows 2000

  – Windows XP

  注: DB2 信息中心在支持 DB2 客户机的一部分 Windows 操作系统上运行。因此，建议从 IBM Web 站点访问 DB2 信息中心，或者在内部网服务器上安装并访问 DB2 信息中心。
- **软件需求**
  – 支持下列浏览器:
    - Mozilla 1.0 或更高版本
    - Internet Explorer V5.5 或 V6.0（对于 Windows XP，则为 Internet Explorer V6.0）
- **通信需求**
  – TCP/IP

限制:

- 需要具有安装 DB2 信息中心的管理特权的帐户。

过程:

要使用“DB2 安装”向导安装 DB2 信息中心:

1. 使用为 DB2 信息中心安装定义的帐户登录至系统。

2. 将 CD 插入到驱动器中。如果启用了自动运行功能，则它将启动“IBM DB2 安装启动板”。
3. “DB2 安装”向导会确定系统语言并启动该语言的安装程序。如果想要运行英语之外的语言的安装程序，或者安装程序无法自动启动，则可以手工启动“DB2 安装”向导。

   要手工启动“DB2 安装”向导:

   a. 单击**开始**并选择**运行**。
   b. 在**打开**字段中，输入以下命令:

      x:\setup.exe /i *2-letter language identifier*

      其中 *x:* 表示 CD 驱动器，而 *2-letter language identifier* 表示将用来运行安装程序的语言。
   c. 单击**确定**。
4. “IBM DB2 安装启动板”打开。要直接进至 DB2 信息中心的安装，单击**安装产品**。联机帮助可指导您完成其余步骤。要调用联机帮助，单击**帮助**。可随时单击**取消**来结束安装。
5. 在**选择您想要安装的产品**页中，单击**下一步**。
6. 在**欢迎使用“DB2 安装”向导**页中，单击**下一步**。“DB2 安装”向导将指导您完成程序安装过程。
7. 要继续安装，必须接受许可协议。在**许可协议**页中，选择**我接受许可协议中的条款**，然后单击**下一步**。
8. 在**选择安装操作**页中，选择**在此计算机上安装 DB2 信息中心**。如果想要在稍后使用响应文件在此计算机或其它计算机上安装 DB2 信息中心，则选择**将设置保存在响应文件中**。单击**下一步**。
9. 在**选择要安装的语言**页中，选择将用来安装 DB2 信息中心的语言。单击**下一步**。
10. 在**指定 DB2 信息中心端口**页中，配置 DB2 信息中心的入局通信。单击**下一步**继续安装。
11. 在**开始复制文件**页中复查您作出的安装选择。要更改任何设置，单击**上一步**。单击**安装**以将 DB2 信息中心文件复制到计算机上。

可以使用响应文件来安装 DB2 信息中心。还可以使用 **db2rspgn** 命令来根据现有安装生成响应文件。

有关安装期间遇到的错误的信息，请参阅 ’My Documents’\DB2LOG\ 目录中的 db2.log 和 db2wi.log 文件。’My Documents’ 目录的位置将取决于计算机的设置。

db2wi.log 文件会捕获最新的 DB2 安装信息。db2.log 会捕获 DB2 产品安装的历史。

相关概念:

- 第 554 页的『DB2 信息中心』
- 第 555 页的『DB2 信息中心安装方案』

相关任务:

- 『使用响应文件安装 DB2 产品（Windows）』（*《安装与配置补充手册》*）
- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 562 页的『以首选语言显示 DB2 信息中心中的主题』

- 第 561 页的『调用 DB2 信息中心』
- 第 557 页的『使用“DB2 安装”向导来安装 DB2 信息中心（UNIX）』

相关参考:

- 『db2rspgn - Response File Generator Command (Windows)』（*Command Reference*）

调用 DB2 信息中心

DB2 信息中心允许您访问使用用于 Linux、UNIX 和 Windows 操作系统的 DB2 系列产品（例如，DB2 通用数据库、DB2 Connect、DB2 Information Integrator 和 DB2 Query Patroller）所需的所有信息。

可以从下列其中一个位置调用 DB2 信息中心;

- 安装了 DB2 UDB 客户机或服务器的计算机
- 安装了 DB2 信息中心的内部网服务器或本地计算机
- IBM Web 站点

先决条件:

在调用 DB2 信息中心之前:

- *可选*: 配置浏览器以便用首选语言来显示主题
- *可选*: 配置 DB2 客户机以使用安装在计算机或内部网服务器上的 DB2 信息中心

过程:

要调用安装了 DB2 UDB 客户机或服务器的计算机上的 DB2 信息中心:

- 从“开始菜单”（Windows 操作系统）: 单击**开始 → 程序 → IBM DB2 → 信息 → 信息中心**。
- 从命令行提示符:
  - 对于 Linux 和 UNIX 操作系统，发出 **db2icdocs** 命令。
  - 对于 Windows 操作系统，发出 **db2icdocs.exe** 命令。

要在 Web 浏览器中打开安装在内部网服务器或本地计算机上的 DB2 信息中心:

- 打开 Web 页面（网址为: http://<host-name>:<port-number>/），其中 <host-name> 表示主机名，而 <port-number> 表示提供 DB2 信息中心的端口号。

要在 Web 浏览器中打开 IBM Web 站点上的 DB2 信息中心:

- 打开 Web 页面（网址为: publib.boulder.ibm.com/infocenter/db2help/）。

相关概念:

- 第 554 页的『DB2 信息中心』
- 第 555 页的『DB2 信息中心安装方案』

相关任务:

- 第 562 页的『以首选语言显示 DB2 信息中心中的主题』
- 第 569 页的『从 DB2 工具调用上下文帮助』
- 第 562 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 570 页的『从命令行处理器调用命令帮助』

- 『设置访问 DB2 信息中心的位置: 公共 GUI 帮助』

相关参考:

- 『HELP Command』(*Command Reference*)

更新安装在计算机或内部网服务器上的 DB2 信息中心

http://publib.boulder.ibm.com/infocenter/db2help/ 提供的 DB2 信息中心将会用新的或更改过的文档定期更新。IBM 还可能提供 DB2 信息中心更新，可以下载这些更新并将它们安装在计算机或内部网服务器上。更新 DB2 信息中心不会更新 DB2 客户机或服务器产品。

先决条件:

必须能够访问连接至因特网的计算机。

过程:

要更新安装在计算机或内部网服务器上的 DB2 信息中心:

1. 打开位于 IBM Web 站点上的 DB2 信息中心: http://publib.boulder.ibm.com/infocenter/db2help/
2. 在欢迎页面的“服务与支持”标题下面的“下载”部分，单击 **DB2 通用数据库文档**链接。
3. 通过将最新刷新的文档映像级别与已安装的文档级别进行比较来确定 DB2 信息中心的版本是否已过时。已安装的文档级别列示在 DB2 信息中心欢迎页面上。
4. 如果有较新版本的 DB2 信息中心可用，则下载适用于您的操作系统的最新刷新的 *DB2 信息中心*映像。
5. 要安装刷新过的 *DB2 信息中心*映像，遵循 Web 页面上提供的指示信息。

相关概念:

- 第 555 页的『DB2 信息中心安装方案』

相关任务:

- 第 561 页的『调用 DB2 信息中心』
- 第 557 页的『使用“DB2 安装”向导来安装 DB2 信息中心（UNIX）』
- 第 559 页的『使用“DB2 安装”向导来安装 DB2 信息中心（Windows）』

以首选语言显示 DB2 信息中心中的主题

DB2 信息中心尝试以您在浏览器首选项中指定的语言显示主题。如果主题未翻译为首选语言，则 DB2 信息中心将显示该主题的英文版。

过程:

要在 Internet Explorer 浏览器中以您的首选语言显示主题:

1. 在 Internet Explorer 中，单击**工具** —> **Internet 选项** —> **语言...** 按钮。“语言首选项”窗口打开。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮。

注: 添加语言并不能保证计算机具有以首选语言显示主题所需的字体。

- 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一项。

3. 刷新该页面以便以首选语言显示 DB2 信息中心。

要在 Mozilla 浏览器中以首选语言显示主题:

1. 在 Mozilla 中，选择**编辑** —> **首选项** —> **语言**按钮。“语言”面板将显示在“首选项”窗口中。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮以从“添加语言”窗口中选择一种语言。
   - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一项。
3. 刷新该页面以便以首选语言显示 DB2 信息中心。

相关概念:

- 第 554 页的『DB2 信息中心』

DB2 PDF 和印刷文档

下列各表提供正式书名、书号和 PDF 文件名。要订购硬拷贝书籍，必须知道正式书名。要打印 PDF 文件，必须知道 PDF 文件名。

DB2 文档按下列标题分类:

- 核心 DB2 信息
- 管理信息
- 应用程序开发信息
- 商业智能信息
- DB2 Connect 信息
- 入门信息
- 教程信息
- 可选组件信息
- 发行说明

对于 DB2 资料库中的每本书籍，下表描述了订购该书籍的硬拷贝、打印或查看该书籍的 PDF 所需的信息。DB2 资料库中的每本书籍的完整描述可从 IBM 出版物中心（IBM Publications Center）获取，网址为 www.ibm.com/shop/publications/order。

核心 DB2 信息

这些书籍中的信息对所有 DB2 用户来说都是基础知识，不管您是程序员、数据库管理员或是使用 DB2 Connect、DB2 仓库管理器或其它 DB2 产品的人员，都将会发现此信息很有用。

表 73. 核心 DB2 信息

书名	书号	PDF 文件名
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x81

表 73. 核心 DB2 信息 (续)

书名	书号	PDF 文件名
《IBM DB2 通用数据库词汇表》	无书号	db2t0c81
《IBM DB2 通用数据库消息参考第 1 卷》	G152-0177，未提供硬拷贝	db2m1c81
《IBM DB2 通用数据库消息参考第 2 卷》	G152-0178，未提供硬拷贝	db2m2c81
《IBM DB2 通用数据库新增内容》	S152-0176	db2q0c81

管理信息

这些书籍中的信息包括有效地设计、实现和维护 DB2 数据库、数据仓库和联合系统所需的那些主题。

表 74. 管理信息

书名	书号	PDF 文件名
《IBM DB2 通用数据库管理指南：计划》	S152-0167	db2d1c81
《IBM DB2 通用数据库管理指南：实现》	S152-0165	db2d2c81
《IBM DB2 通用数据库管理指南：性能》	S152-0166	db2d3c81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
《IBM DB2 通用数据库数据恢复和高可用性指南与参考》	S152-0181	db2hac81
《IBM DB2 通用数据库数据仓库中心管理指南》	S152-0188	db2ddc81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

应用程序开发信息

这些书籍中的信息对于应用程序开发者或使用 DB2 通用数据库（DB2 UDB）的程序员特别有用。您将找到有关受支持的语言和编译器的信息，以及使用各种受支持的编程接口（例如，嵌入式 SQL、ODBC、JDBC、SQLJ 和 CLI）访问 DB2 UDB 所需的文档。如果正在使用 DB2 信息中心，还可访问 HTML 版本的源代码以获取样本程序。

表 75. 应用程序开发信息

书名	书号	PDF 文件名
《IBM DB2 通用数据库应用程序开发指南：构建和运行应用程序》	S152-0168	db2axc81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx81

商业智能信息

这些书籍中的信息描述如何使用将增强 DB2 通用数据库的数据仓储功能和分析功能的组件。

表 76. 商业智能信息

书名	书号	PDF 文件名
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	G152-0187	db2idc81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

DB2 Connect 信息

此类别中的信息描述如何使用 DB2 Connect 企业版或 DB2 Connect 个人版来存取大型机和中型机服务器上的数据。

表 77. DB2 Connect 信息

书名	书号	PDF 文件名
IBM Connectivity Supplement	无书号	db2h1x81
《*IBM DB2 Connect 快速入门 DB2 Connect 企业版*》	G152-0271	db2c6c81
《*IBM DB2 Connect 快速入门 DB2 Connect 个人版*》	G152-0171	db2c1c81
《*IBM DB2 Connect 用户指南*》	S152-0172	db2c0c81

入门信息

安装和配置服务器、客户机以及其它 DB2 产品时，此类别中的信息非常有用。

表 78. 入门信息

书名	书号	PDF 文件名
《*IBM DB2 通用数据库快速入门 DB2 客户机版*》	G152-0170，未提供硬拷贝	db2itc81
《*IBM DB2 通用数据库快速入门 DB2 服务器版*》	G152-0173	db2isc81
《*IBM DB2 通用数据库快速入门 DB2 个人版*》	G152-0175	db2i1c81
《*IBM DB2 通用数据库安装与配置补充手册*》	G152-0174，未提供硬拷贝	db2iyc81
《*IBM DB2 通用数据库快速入门 DB2 Data Links Manager 版*》	G152-0169	db2z6c81

教程信息

教程信息介绍 DB2 功能部件并指导如何执行各种任务。

表 79. 教程信息

书名	书号	PDF 文件名
《*商业智能教程：数据仓库简介*》	无书号	db2tuc81
《*商业智能教程：数据仓储扩展课程*》	无书号	db2tac81
Information Catalog Center Tutorial	无书号	db2aix81
Video Central for e-business Tutorial	无书号	db2twx81
《*Visual Explain 教程*》	无书号	db2tvc81

可选组件信息

此类别中的信息描述如何使用可选 DB2 组件。

表 80. 可选组件信息

书名	书号	PDF 文件名
《IBM DB2 Cube Views 指南与参考》	S152–0532	db2aac81
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference	SC27-1226	db2sbx81
IBM DB2 Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
《DB2 Net Search Extender 管理和用户指南》 **注**: 此文档的 HTML 不是从"HTML 文档"CD 安装的。	S152-0596	不适用

发行说明

发行说明提供了特定于产品发行版和修订包级别的附加信息。发行说明还提供了并入到每个发行版、更新和修订包中的文档更新的总结。

表 81. 发行说明

书名	书号	PDF 文件名
《DB2 发行说明》	请参阅"注"。	请参阅"注"。
《DB2 安装说明》	仅在产品 CD-ROM 上提供。	未提供。

注: 提供有下列格式的发行说明:

- XHTML 和文本格式(在产品 CD 上)
- PDF 格式(在 PDF 文档 CD 上)

此外,《发行说明》中讨论*已知问题和变通方法*和*发行版之间的不兼容性*的部分还会出现在 DB2 信息中心中。

要在基于 UNIX 的平台上查看文本格式的发行说明,请参阅 `Release.Notes` 文件。此文件位于 `DB2DIR/Readme/%L` 目录中,其中 `%L` 表示语言环境名称,`DB2DIR` 表示:

- 对于 AIX 操作系统: `/usr/opt/db2_08_01`
- 对于所有其它基于 UNIX 的操作系统: `/opt/IBM/db2/V8.1`

相关概念:

- 第 553 页的『DB2 文档和帮助』

相关任务:

- 第 568 页的『从 PDF 文件打印 DB2 书籍』
- 第 568 页的『订购印刷的 DB2 书籍』
- 第 569 页的『从 DB2 工具调用上下文帮助』

从 PDF 文件打印 DB2 书籍

可从 *DB2 PDF 文档* CD 上的 PDF 文件打印 DB2 书籍。通过使用 Adobe Acrobat Reader，可打印整本书或特定范围的那些页。

先决条件:

确保安装了 Adobe Acrobat Reader。如果需要安装 Adobe Acrobat Reader，则可从 Adobe Web 站点获得它，网址为 www.adobe.com。

过程:

要从 PDF 文件打印 DB2 书籍:

1. 插入 *DB2 PDF 文档* CD。在 UNIX 操作系统上，安装 “DB2 PDF 文档” CD。有关如何在 UNIX 操作系统上安装 CD 的详细信息，参阅《*快速入门*》一书。
2. 打开 index.htm。文件将在浏览器窗口中打开。
3. 单击想要查看的 PDF 的标题。该 PDF 将在 Acrobat Reader 中打开。
4. 选择**文件** → **打印**以打印想要的书籍的任何部分。

相关概念:

- 第 554 页的『DB2 信息中心』

相关任务:

- 『装上 CD-ROM（AIX）』（《*DB2 服务器快速入门*》）
- 『装上 CD-ROM（HP-UX）』（《*DB2 服务器快速入门*》）
- 『装上 CD-ROM（Linux）』（《*DB2 服务器快速入门*》）
- 第 568 页的『订购印刷的 DB2 书籍』
- 『安装 CD-ROM（Solaris Operating Environment）』（《*DB2 服务器快速入门*》）

相关参考:

- 第 563 页的『DB2 PDF 和印刷文档』

订购印刷的 DB2 书籍

如果喜欢使用硬拷贝书籍，可以用以下三种方式中的一种订购它们。

过程:

可在某些国家或地区订购印刷版书籍。访问您所在国家或地区的 IBM 出版物 Web 站点，以了解您所在国家或地区是否提供此项服务。如果可以订购这些出版物，则您可以:

- 与 IBM 授权经销商或市场营销代表联系。要查找您当地的 IBM 代表，查看 IBM 全球联系人目录（IBM Worldwide Directory of Contacts），网址为 www.ibm.com/planetwide。

- 访问 IBM 出版物中心（IBM Publications Center），网址为 http://www.ibm.com/shop/publications/order。可能未在所有国家或地区提供从 IBM 出版物中心订购书籍这项功能。

DB2 产品可用时，印刷书籍与 *DB2 PDF 文档 CD* 上以 PDF 格式提供的那些书籍是相同的。印刷书籍中的内容出现在 *DB2 信息中心 CD* 中时也是相同的。但是，DB2 信息中心 CD 中有一些附加内容未出现在 PDF 书籍中的任何位置（例如，SQL 管理例程和 HTML 样本）。并非 DB2 PDF 文档 CD 上提供的所有书籍都可以订购硬拷贝。

注: DB2 信息中心的更新比 PDF 或硬拷贝书籍的更新要频繁得多；一提供文档更新就安装它们，或者参阅网址如下的 DB2 信息中心以获取最新信息：http://publib.boulder.ibm.com/infocenter/db2help/。

相关任务:

- 第 568 页的『从 PDF 文件打印 DB2 书籍』

相关参考:

- 第 563 页的『DB2 PDF 和印刷文档』

从 DB2 工具调用上下文帮助

上下文帮助提供有关与特定窗口、笔记本、向导或顾问程序相关联的任务或控件的信息。上下文帮助可从具有图形用户界面的 DB2 管理和开发工具获得。有两种类型的上下文帮助:

- 通过位于每个窗口或笔记本上的**帮助**按钮访问的帮助
- 弹出信息，即将鼠标光标放到字段或控件上或在窗口、笔记本、向导或顾问程序中选择了字段或控件并按 F1 键时显示的弹出信息窗口。

帮助按钮允许您访问概述、先决条件和任务信息。弹出信息描述各个字段和控件。

过程:

要调用上下文帮助:

- 要获取窗口和笔记本帮助，启动其中一个 DB2 工具，然后打开任意窗口或笔记本。单击窗口或笔记本右下角的**帮助**按钮以调用上下文帮助。

  还可从位于每个 DB2 工具中心上方的**帮助**菜单项访问上下文帮助。

  在向导和顾问程序中，单击第一页上的“任务概述”链接以查看上下文帮助。
- 要获取有关窗口或笔记本的各个控件的弹出信息帮助，单击该控件，然后按 **F1**。包含有关控件的详细信息的弹出信息将显示在黄色窗口中。

  注: 如果希望只要将鼠标光标放在字段或控件上就显示弹出信息，在“工具设置”笔记本的**文档**页上选择**自动显示弹出信息**复选框。

  与弹出信息类似，诊断弹出信息是另一种形式的上下文相关帮助；它们包含数据输入规则。诊断弹出信息显示在输入的数据无效或不充分时出现的紫色窗口中。会对以下各项显示诊断弹出信息:

  – 必填字段。

  – 其数据遵照精确格式的字段，例如，日期字段。

相关任务:

- 第 561 页的『调用 DB2 信息中心』
- 第 570 页的『从命令行处理器调用消息帮助』
- 第 570 页的『从命令行处理器调用命令帮助』
- 第 571 页的『从命令行处理器调用 SQL 状态帮助』
- 『访问 DB2 信息中心: 概念帮助』
- 『如何使用 DB2 UDB 帮助: 公共 GUI 帮助』
- 『设置访问 DB2 信息中心的位置: 公共 GUI 帮助』
- 『设置对 DB2 上下文帮助和文档的访问权: 公共 GUI 帮助』

从命令行处理器调用消息帮助

消息帮助描述产生消息的原因并描述为解决错误而应采取的任何操作。

过程:

要调用消息帮助，打开命令行处理器并输入:

```
? XXXnnnnn
```

其中 *XXXnnnnn* 表示有效的消息标识。

例如，`? SQL30081` 会显示有关 SQL30081 消息的帮助。

相关概念:

- 『消息介绍』(*《消息参考》第 1 卷*)

相关参考:

- 『db2 - Command Line Processor Invocation Command』(*Command Reference*)

从命令行处理器调用命令帮助

命令帮助说明命令行处理器中命令的语法。

过程:

要调用命令帮助，打开命令行处理器并输入:

```
? command
```

其中 *command* 表示一个关键字或整条命令。

例如，`? catalog` 显示所有 CATALOG 命令的帮助，而 `? catalog database` 只显示 CATALOG DATABASE 命令的帮助。

相关任务:

- 第 569 页的『从 DB2 工具调用上下文帮助』
- 第 561 页的『调用 DB2 信息中心』
- 第 570 页的『从命令行处理器调用消息帮助』

- 第 571 页的『从命令行处理器调用 SQL 状态帮助』

相关参考:

- 『db2 - Command Line Processor Invocation Command』（*Command Reference*）

从命令行处理器调用 SQL 状态帮助

DB2 通用数据库返回可作为 SQL 语句结果的条件的 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

过程:

要调用 SQL 状态帮助，打开命令行处理器并输入:

```
? sqlstate 或 ? class code
```

其中，*sqlstate* 表示有效的 5 位 SQL 状态，*class code* 表示该 SQL 状态的前 2 位。

例如，`? 08003` 显示 08003 SQL 状态的帮助，而 `? 08` 显示 08 类代码的帮助。

相关任务:

- 第 561 页的『调用 DB2 信息中心』
- 第 570 页的『从命令行处理器调用消息帮助』
- 第 570 页的『从命令行处理器调用命令帮助』

DB2 教程

DB2® 教程帮助您了解 DB2 通用数据库的各个方面。教程提供了开发应用程序、调整 SQL 查询性能、使用数据仓库、管理元数据和使用 DB2 开发 Web 服务等方面的课程，这些课程中还提供了逐步指示信息。

开始之前:

可从“信息中心”查看 XHTML 版本的教程，网址如下: http://publib.boulder.ibm.com/infocenter/db2help/。

某些教程课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述，请参阅每个教程。

DB2 通用数据库教程:

单击以下列表中的教程标题以查看该教程。

《商业智能教程: 数据仓库中心介绍》
: 使用“数据仓库中心”来执行介绍性的数据仓储任务。

《商业智能教程: 数据仓储扩展课程》
: 使用“数据仓库中心”来执行高级数据仓储任务。

《信息目录中心教程》
: 使用“信息目录中心”来创建和管理信息目录以查找并使用元数据。

《Visual Explain 教程》
使用 Visual Explain 来分析、优化和调整 SQL 语句以获取更好的性能。

DB2 故障诊断信息

提供有大量故障诊断和问题确定信息，可帮助您使用 DB2® 产品。

DB2 文档
DB2 信息中心以及构成 DB2 资料库的 PDF 书籍中处处可找到故障诊断信息。可参阅 DB2 信息中心导航树（在浏览器窗口的左窗格中）的“支持和故障诊断”分支以查看 DB2 故障诊断文档的完整列表。

DB2 技术支持 Web 站点
如果您遇到了问题并且想要获取查找可能的原因和解决方案的帮助，请参阅 DB2 技术支持（DB2 Technical Support）Web 站点。该“技术支持”站点具有指向最新 DB2 出版物、技术说明、授权程序分析报告（APAR）、修订包的链接、内部 DB2 错误代码的最新列表以及其它资源。可搜索此知识库并查找问题的可能解决方案。

访问位于以下网址的 DB2 技术支持 Web 站点：http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 问题确定教程系列
要查找有关如何快速标识和解决在使用 DB2 产品时可能遇到的问题的信息，参阅 DB2 问题确定教程系列 Web 站点。有一个教程介绍可用的 DB2 问题确定设施和工具并帮助您决定何时使用它们。其它教程处理相关主题，例如“数据库引擎问题确定”、“性能问题确定”和“应用程序问题确定”。

查看 DB2 技术支持站点上的 DB2 问题确定教程的完整集合，网址如下：http://www.ibm.com/software/data/support/pdm/db2tutorials.html

相关概念:
- 第 554 页的『DB2 信息中心』
- 『问题确定 － DB2 技术支持教程简介』（*故障诊断指南*）

辅助功能

辅助功能部件可帮助那些身体有某些缺陷（如活动不方便或视力不太好）的用户成功地使用软件产品。以下列表指定 DB2® V8 产品中的主要辅助功能部件:
- 所有 DB2 功能可使用键盘（而不是鼠标）导航来实现。有关更多信息，请参阅第 573 页的『键盘输入和导航』。
- 可定制 DB2 用户界面上的字体大小和颜色。有关更多信息，请参阅第 573 页的『界面显示的辅助功能』。
- DB2 产品支持使用 Java™ Accessibility API 的辅助功能应用程序。有关更多信息，请参阅第 573 页的『与辅助技术的兼容性』。
- DB2 文档是以易使用格式提供的。有关更多信息，请参阅第 573 页的『文档的辅助功能』。

键盘输入和导航

键盘输入

只使用键盘就可以操作 DB2 工具。使用键或键组合就可以执行使用鼠标所能完成的操作。标准操作系统击键用于标准操作系统操作。

有关使用键或键组合执行操作的更多信息，请参阅 键盘快捷方式和加速键：公共 GUI 帮助。

键盘导航

可使用键或键组合来导航 DB2 工具用户界面。

有关使用键或键组合来导航 DB2 工具的更多信息，请参阅 键盘快捷方式和加速键：公共 GUI 帮助。

键盘焦点

在 UNIX® 操作系统中，击键操作起作用的活动窗口的区域将突出显示。

界面显示的辅助功能

DB2 工具所具有的功能部件使视力不太好的用户更易使用。这些辅助功能方面的增强包括了对可定制字体属性的支持。

字体设置

可使用“工具设置”笔记本来选择菜单和对话框窗口中文本的颜色、大小和字体。

有关指定字体设置的更多信息，请参阅 更改菜单和文本的字体：公共 GUI 帮助。

不依赖于颜色

不需要分辨颜色就可以使用此产品中的任何功能。

与辅助技术的兼容性

DB2 工具界面支持 Java Accessibility API，它使您能够将屏幕阅读器和其它辅助技术与 DB2 产品配合使用。

文档的辅助功能

DB2 的相关文档是以 XHTML 1.0 格式提供的，它在大部分 Web 浏览器中是可查看的。XHTML 允许您根据浏览器中设置的显示首选项来查看文档。还允许您使用屏幕阅读器和其它辅助技术。

语法图是以点分十进制格式提供的。仅当使用屏幕阅读器访问联机文档时，此格式才可用。

相关概念:

- 第 574 页的『点分十进制语法图』

相关任务:

- 『键盘快捷方式和加速键：公共 GUI 帮助』
- 『更改菜单和文本的字体：公共 GUI 帮助』

点分十进制语法图

语法图是以点分十进制的格式为使用屏幕阅读器访问信息中心的用户提供的。

在点分十进制格式中，每个语法元素写在单独的一行上。如果两个或多个语法元素总是一起出现（或总是一起不出现），它们可显示在同一行上，这是因为可将它们视作单个复合语法元素。

每一行以点分十进制编号开始；例如，3、3.1 或 3.1.1。要正确地听到这些数字，确保屏幕阅读器设置为读出标点。具有相同点分十进制编号的所有语法元素（例如，具有编号 3.1 的所有语法元素）是互斥的替代项。如果听到行 3.1 USERID 和 3.1 SYSTEMID，就知道语法可能包括 USERID 或 SYSTEMID，但不会同时包括这两者。

点分十进制编号级别表示嵌套级别。例如，如果具有点分十进制编号 3 的语法元素后跟点分十进制编号为 3.1 的一系列语法元素，则编号为 3.1 的所有语法元素是编号为 3 的语法元素的下级。

某些单词和符号用在点分十进制编号的旁边以添加有关这些语法元素的信息。这些单词和符号有时可能会出现在元素本身的开头。为易于识别，如果该单词或符号是语法元素的一部分，它的前面会加上反斜杠（\）字符。* 符号可用在点分十进制编号的旁边以指示该语法元素重复。例如，点分十进制编号为 3 的语法元素 *FILE 的格式为 3 * FILE。3* FILE 这一格式指示语法元素 FILE 重复。格式 3* * FILE 指示语法元素 * FILE 重复。

用来分隔一串语法元素的字符（例如，逗号）在语法中刚好显示在它们要分隔的项之前。这些字符可与每一项显示在同一行上，或显示在单独一行上并带有与相关项相同的点分十进制编号。该行还可显示另一个符号，该符号给出有关语法元素的信息。例如，行 5.1*、5.1 LASTRUN 和 5.1 DELETE 意味着如果使用多个 LASTRUN 和 DELETE 语法元素，必须用逗号分隔这些元素。如果未指定分隔符，则假定使用空格来分隔每个语法元素。

如果语法元素前面有 % 符号，这表示在别处定义的引用。% 符号之后的字符串是语法段的名称，而非文字。例如，行 2.1 %OP1 意味着您应引用单独的语法分段 OP1。

下列单词和符号用在点分十进制编号的旁边:

- ? 表示可选语法元素。后跟 ? 符号的点分十进制编号指示具有相应点分十进制编号的所有语法元素及任何下级语法元素都是可选的。如果只有一个带有点分十进制编号的语法元素，则 ? 符号与该语法元素显示在同一行上（例如，5? NOTIFY）。如果有多个带有点分十进制编号的语法元素，则 ? 符号单独显示在一行上，后跟可选语法元素。例如，如果您听到行 5 ?、5 NOTIFY 和 UPDATE，就知道语法元素 NOTIFY 和 UPDATE 是可选的；即，您可选择其中一项或全部都不选。? 符号相当于路线图中的支路。
- ! 表示缺省语法元素。后跟 ! 符号的点分十进制编号和语法元素指示该语法元素是共享同一点分十进制编号的所有语法元素的缺省选项。只有共享同一点分十进制编号的语法元素的其中一个可指定 ! 符号。例如，如果听到行 2? FILE、2.1! (KEEP) 和 2.1 (DELETE)，就知道 (KEEP) 是 FILE 关键字的缺省选项。在此示例中，如果包括 FILE 关键字但未指定选项，将应用缺省选项 KEEP。缺省选项还会应用于下一个较高的点分十进制编号。在此示例中，如果省略了 FILE 关键字，将使用缺省值 FILE(KEEP)。但是，如果听到行 2? FILE、2.1、2.1.1! (KEEP) 和 2.1.1 (DELETE)，

则缺省选项 KEEP 仅应用于下一个较高的点分十进制编号 2.1（它没有相关联的关键字），而不会应用于 2? FILE。如果省略了关键字 FILE，则不会使用任何值。

- * 表示可重复零次或多次的语法元素。后跟 * 符号的点分十进制编号指示此语法元素可使用零次或多次；即，它是可选的而且可以重复。例如，如果听到行 5.1* data area，就知道可以包括一个数据区、多个数据区或者不包括数据区。如果听到行 3*、3 HOST 和 3 STATE，就知道可包括 HOST 和／或 STATE 或者不包括任何内容。

  注:

  1. 如果点分十进制编号的旁边有星号（*）且只有一项带有该点分十进制编号，可重复同一项多次。
  2. 如果点分十进制编号的旁边有星号且有若干项带有该点分十进制编号，可使用列表中的多项，但每项只能使用一次。在先前示例中，可以写为 HOST STATE，但不能写为 HOST HOST。
  3. * 符号相当于路线语法图中的回路。

- + 表示必须被包括一次或多次的语法元素。后跟 + 符号的点分十进制编号指示此语法元素必须被包括一次或多次；即，它必须至少被包括一次，而且可以重复。例如，如果听到行 6.1+ data area，就知道必须至少包括一个数据区。如果听到行 2+、2 HOST 和 2 STATE，就知道必须包括 HOST 和／或 STATE。与 * 符号类似，如果 + 符号是带有该点分十进制编号的唯一项，则它只能重复特定项。与 * 符号一样，+ 符号相当于路线语法图中的回路。

相关概念:

- 第 572 页的『辅助功能』

相关任务:

- 『键盘快捷方式和加速键: 公共 GUI 帮助』

相关参考:

- 『How to read the syntax diagrams』（*SQL Reference, Volume 2*）

DB2 通用数据库产品的 Common Criteria 认证

DB2 通用数据库正在进行 Common Criteria 评估保证级别 4（EAL4）的评估认证。有关 Common Criteria 的更多信息，请参阅 Common Criteria Web 站点: http://niap.nist.gov/cc-scheme/

附录 G. 声明

IBM 可能在所有国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含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；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：（i）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，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario

L6G 1C7
CANADA

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息可能包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年*）。此部分代码是根据 IBM 公司的样本程序衍生出来的。
© Copyright IBM Corp. （*输入年份*）. All rights reserved.

商标

下列各项是国际商业机器公司在美国和／或其他国家或地区的商标，且已在 DB2 UDB 文档库中的至少一份文档中使用。

ACF/VTAM	iSeries
AISPO	LAN Distance
AIX	MVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	NetView
BookManager	OS/390
C Set++	OS/400
C/370	PowerPC
CICS	pSeries
Database 2	QBIC
DataHub	QMF
DataJoiner	RACF
DataPropagator	RISC System/6000
DataRefresher	RS/6000
DB2	S/370
DB2 Connect	SP
DB2 Extender	SQL/400
DB2 OLAP Server	SQL/DS
DB2 Information Integrator	System/370
DB2 Query Patroller	System/390
DB2 Universal Database	SystemView
Distributed Relational	Tivoli
Database Architecture	VisualAge
DRDA	VM/ESA
eServer	VSE/ESA
Extended Services	VTAM
FFST	WebExplorer
First Failure Support Technology	WebSphere
IBM	WIN-OS/2
IMS	z/OS
IMS/ESA	zSeries

下列各项是其他公司的商标或注册商标，且已在 DB2 UDB 文档库中的至少一份文档中使用:

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其他国家或地区的商标。

Intel 和 Pentium 是 Intel Corporation 在美国和／或其他国家或地区的商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

[A]

[B]

[C]

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[N]

[P]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

U

V

W

X

与 IBM 联系

在中国，请致电下列其中一个号码以与 IBM 联系：

- 800-810-1818 或 (010) 84981188 分机 5151，可获得售前客户服务
- 800-810-1818 或 (010) 84981188 分机 5200，可获得售后客户服务
- 800-810-1818 或 (010) 84981188 分机 5017，可获得市场营销与销售的信息

要查找您所在国家或地区的 IBM 营业处，可在网上查看 IBM 全球联系人目录（Directory of Worldwide Contacts），网址为：http://www.ibm.com/planetwide

产品信息

有关 DB2 通用数据库产品的信息可通过万维网获取，网址为：http://www-900.ibm.com/cn/software/db2/

此站点包含有关 DB2 产品家族、DB2 解决方案、技术前沿与趋势、DB2 服务、成功案例、市场活动、培训与认证、DB2 开发者园地、合作伙伴、下载中心、资料库、第三方分析报告、殊荣与奖项、DB2 新闻以及如何购买 DB2 的最新信息。

有关如何在中国以外的国家或地区与 IBM 联系的信息，请访问 IBM Worldwide 页面，网址为：www.ibm.com/planetwide

中国印刷

S152-0166-01

Spine information:

IBM® DB2™ Universal Database
DB2 通用数据库

管理指南: 性能

版本 8.2