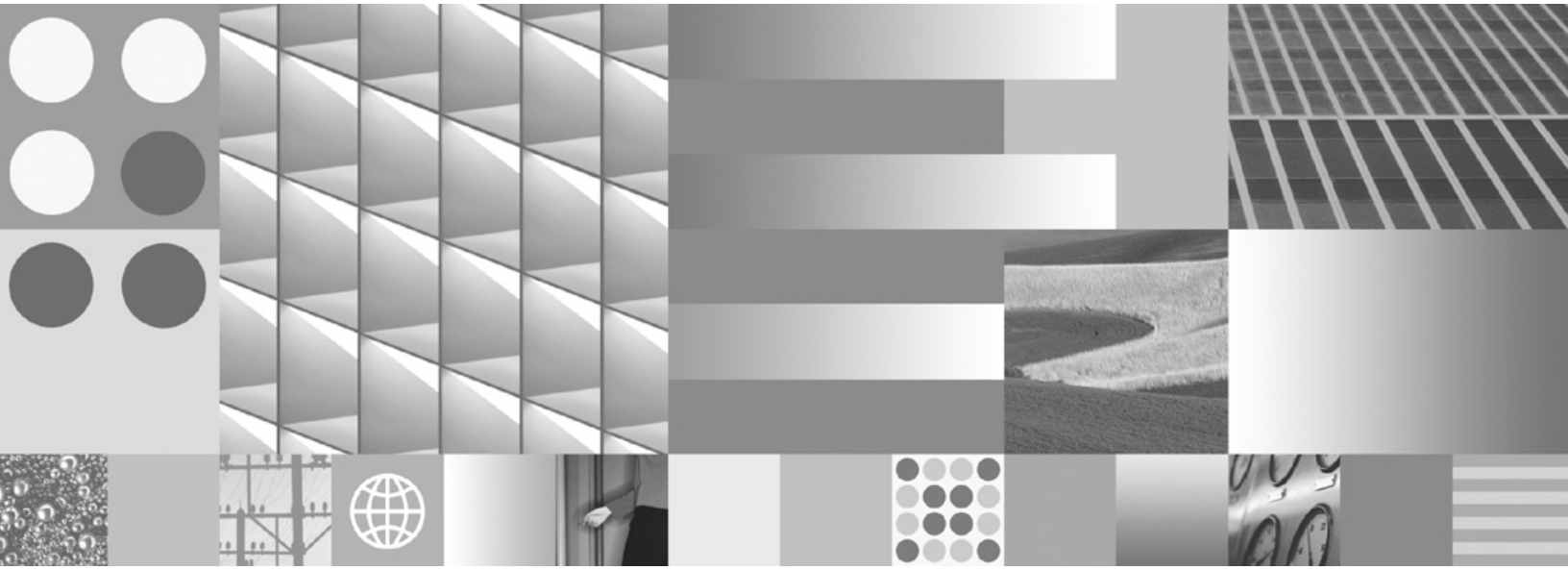
IBM DB2 9.7
Linux 版、UNIX 版和 Windows 版

版本 9.7



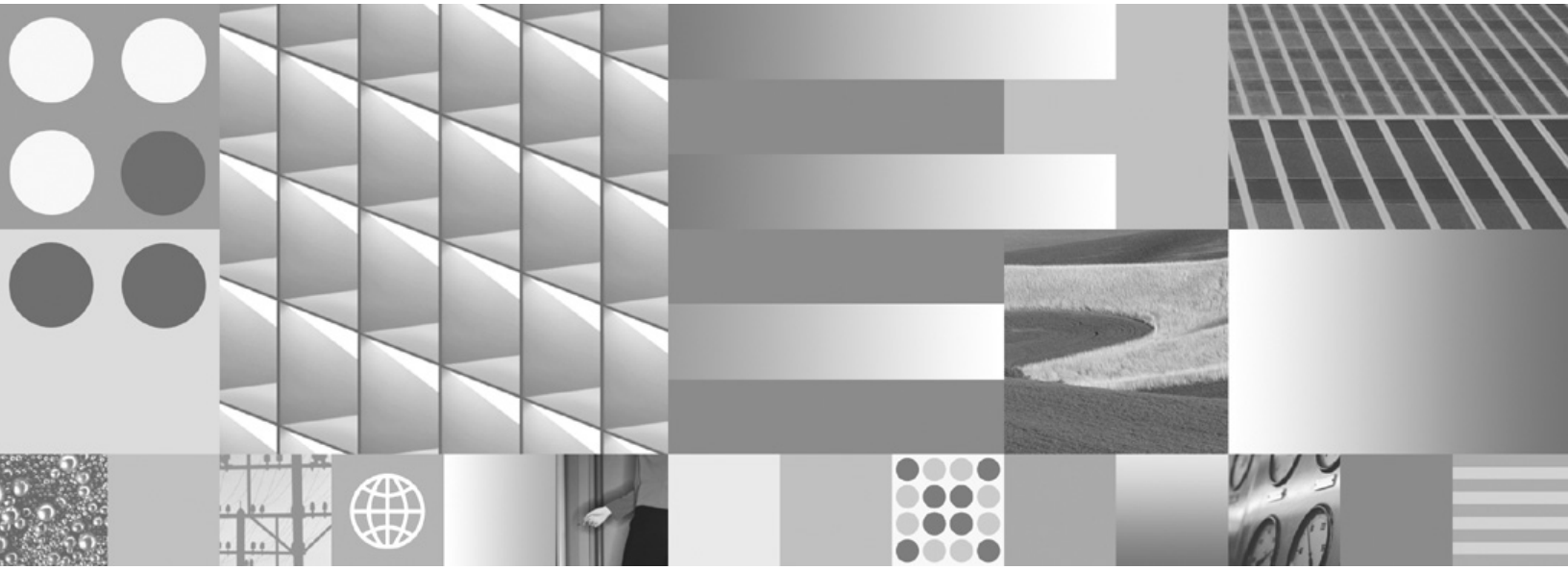
故障诊断和调整数据库性能

2009 年 11 月更新

S151-1164-01

IBM DB2 9.7
Linux 版、UNIX 版和 Windows 版

版本 9.7



故障诊断和调整数据库性能

2009 年 11 月更新

S151-1164-01

注意

使用此信息及其支持的产品前，请先阅读第 505 页的附录 B，『声明』下的常规信息。

修订版声明

此文档包含 IBM 的所有权信息。它在许可协议中提供，且受版权法的保护。本出版物中包含的信息不包括对任何产品的保证，且提供的任何语句都不需要如此解释。

您可在线或通过当地的 IBM 代表处订购 IBM 出版物。

- 要在线订购出版物，请转至 IBM 出版物中心，网址为: www.ibm.com/shop/publications/order
- 要查找当地的 IBM 代表处，请转至 IBM 全球联系人目录，网址为: www.ibm.com/planetwide

要从美国或加拿大的 DB2 市场和销售部订购 DB2 出版物，请致电 1-800-IBM-4YOU（426-4968）。

当您向 IBM 发送信息时，即同意授予 IBM 独一无二的权力以它认为适当且不会对您造成任何影响的方式使用或分发该信息。

© Copyright International Business Machines Corporation 2006, 2009.

目录

© Copyright IBM Corp. 2006, 2009

关于本书

此指南提供有关调整数据库性能和解决 DB2® 数据库客户机和服务器问题的信息。

它可帮助您执行下列操作:

- 制定性能监视和调整策略
- 制定日常操作的故障诊断策略
- 调整数据库服务器的配置
- 对使用数据库服务器的应用程序进行更改
- 简要地确定问题与错误
- 根据症状解决问题
- 了解可用的诊断工具

本书的读者用户

此指南适用于对数据库性能调整以及 DB2 数据库客户机和服务器问题故障诊断感兴趣的客户、用户、系统管理员、数据库管理员（DBA）、通信专家、应用程序开发者和技术支持代表。要使用本书，您应熟悉以下方面:

- 通信、关系数据库和局域网（LAN）概念
- 硬件和软件要求与选项
- 网络的整体配置
- 在网络上运行的应用程序和其他功能
- 基本的 DB2 数据库管理任务
- 有关适用于您安装的产品的《快速入门》指南中所描述的安装和早期任务的信息

本书的结构

为了帮助您对数据库系统进行性能监视和调整，此处提供的信息包含对影响数据库性能的因素进行了解所必需的背景资料，以及用来帮助您调整系统性能的指示信息。为了帮助您了解、找出和解决 DB2 软件的问题，故障诊断和支持信息包含有关如何使用 DB2 产品附带提供的问题确定资源的指示信息。

第 1 部分 调整数据库性能

作为数据库管理员，可能会遇到以下情况：用户时常报告其数据库应用程序的运行速度很慢。此处提供的信息描述如何制定性能监视策略以获取对数据库系统性能的客观评价（与历史结果比较），如何调整数据库服务器的配置以及如何对使用数据库服务器的应用程序进行更改；在不增加处理成本和不对用户降级服务的情况下提高数据库系统性能这一目标涉及的所有方面。

- 第 1 章『性能调整工具和方法』描述如何设计和实现基准测试程序来帮助您提高性能。
- 第 2 章『性能监视工具和方法』提供有关可运行监视策略的重要性的信息，该策略定期收集关键的系统性能数据。

© Copyright IBM Corp. 2006, 2009

- 第 3 章『影响性能的因素』包含有关可以影响数据库系统性能的各个因素的信息。可以对这些因素的其中一些进行调整或重新配置。
- 第 4 章『制定性能调整策略』描述可帮助您显著提高工作负载性能的 DB2 设计顾问程序工具。

第 2 部分 对问题进行故障诊断

为了帮助您独立解决问题，本部分中包含的信息描述如何确定问题根源、如何收集诊断信息、从何处获取修订以及要搜索哪些知识库以获取其他信息。如果必须与 IBM 软件支持机构联系，那么此处提供的信息描述了如何与支持机构联系以及服务技术人员帮助您解决问题所需的诊断信息。

- 第 5 章『用于故障诊断的工具』描述可以用来帮助执行解决问题的系统方法的故障诊断工具。其目标是，确定某项功能未按预期方式工作的原因以及解决问题的方式。
- 第 6 章『对 DB2 数据库进行故障诊断』提供了有关可能出现的各种已知问题以及如何对它们进行故障诊断的信息。
- 第 7 章『对 DB2® Connect™ 进行故障诊断』提供了有关可能出现的各种已知问题以及如何对它们进行故障诊断的信息。
- 第 8 章『搜索知识库』提供了有关如何通过搜索 IBM 知识库来查找问题的解决方案的信息。本章描述如何通过使用可用的资源、支持工具和搜索方法来优化结果。
- 第 9 章『获取 DB2 产品修订』提供了有关获取可能已可用来解决问题的产品修订的信息。可以通过执行此处概括的步骤来获取修订。
- 第 10 章『了解有关故障诊断的更多信息』对以下内容进行了描述: 下列主题可以如何帮助您获取高效地对 DB2 数据库服务器问题进行故障诊断所需的概念性信息。
- 第 11 章『与 IBM 软件支持机构联系』提供了有关如何与 IBM 软件支持机构联系及其在帮助您解决产品缺陷和数据库问题时将要求您提供哪些资料的信息。

第 3 部分 附录

- 附录 A『DB2 技术信息概述』
- 附录 B『声明』

第 1 部分 性能概述

*性能*是指计算机系统响应特定工作负载时的行为方式。按照系统响应时间、吞吐量和资源利用率来测量性能。

性能还受下列因素影响:

- 系统中的可用资源量
- 充分利用和共享那些资源的程度

通常，您希望对系统进行调整以改善其成本/效益比率。具体目标可能包括:

- 处理更大或更紧迫的工作负载，而不增加处理成本
- 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，而不降低对用户的服务质量

性能调整的某些好处，例如更高效地利用资源以及能够将更多用户添加到系统，是有形的。而其他好处是无形的，例如，由于响应速度更快而令用户更加满意。

性能调整准则

在制订性能调整总体方案时，请牢记下列准则。

- **记住递减返回定律**：最大的性能效益通常来自于最初的努力。
- **不要只为调整而调整**：进行调整以解除已确定的约束。如果所调整的资源不是造成性能问题的主要原因，那么实际上会使后续调整工作更加困难。
- **对整个系统进行全盘考虑**：不能片面地调整一个参数或资源。在进行调整前，务必考虑所作的更改对整个系统的影响。性能调整要求在各种系统资源之间进行权衡。例如，您可能会增大缓冲池大小以提高 I/O 性能，但是，缓冲池越大，所需的内存量就越多，这将会导致其他方面的性能下降。
- **每次更改一个参数**：每次不要更改多个因素。即使您确定所有更改都有好处，也没有任何办法来评估每项更改的贡献。
- **按级别进行测量和配置**：每次调整系统的一个级别。系统级别包括:
  - 硬件
  - 操作系统
  - 应用程序服务器和请求者
  - 数据库管理器
  - SQL 和 XQuery 语句
  - 应用程序
- **检查是否存在硬件和软件问题**：某些性能问题可以通过维修硬件和/或修订软件来解决。在维修硬件或修订软件之前，请不要花费过多时间来监视和调整系统。
- **在升级硬件前搞清楚问题**：尽管增加存储器或提高处理器能力能立即提高性能，但您还是应该花时间了解系统的瓶颈所在。否则，您可能会在耗费资金购置磁盘存储器之后，才发现系统没有利用该存储器所需的处理能力或通道。

© Copyright IBM Corp. 2006, 2009

- **在开始进行调整前，安排好后退过程**：如果调整努力导致性能意外下降，那么在尝试替代方案前，应该将所作的更改撤销。您应该保存原始设置，以便轻松方便地撤销不想保留的更改。

制订性能提高过程

性能提高过程是一种可反复的方案，用于监视和调整性能的各个方面。您可以根据性能监视结果来调整数据库服务器的配置以及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应该以您对使用数据的应用程序类型的了解以及您对数据访问模式的理解为依据。不同类型的应用程序有不同的性能要求。

任何性能提高过程都包含下列基本步骤:

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. 制订并执行性能监视方案。
4. 持续分析监视结果，以确定需要调整的资源。
5. 每次进行一项调整。

到了某个时候，如果不再能够通过调整数据库服务器或应用程序来提高性能，那么表明需要升级硬件了。

用户可以提供的性能信息

需要对系统进行调整的第一个征兆可能是，用户对性能有所抱怨。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进行监视和调整，那么可以通过倾听用户的意见来解决性能问题。首先，可以向用户询问几个简单的问题，例如:

- 您所谓的“响应慢”达到何种程度？是比预期速度慢 10% 还是慢数十倍？
- 您什么时候开始注意到此问题？此问题是最近出现的，还是一直都存在？
- 其他用户是否遇到相同的问题？这些用户是一两个人还是整个组？
- 如果是一组用户遇到同一个问题，那些这些用户是否连接到同一个局域网？
- 此问题是否可能与特定类型的事务或应用程序相关？
- 您是否注意到此问题的出现具有任何模式？例如，此问题是在一天的特定时间发生还是持续发生？

性能调整限制

性能调整的好处是有限的。在考虑要投入多少时间和费用来提高系统性能时，务必评估要投入多少额外的时间和费用来帮助系统的用户。

如果系统遇到响应时间或吞吐量问题，那么通常可以通过进行调整来提高性能。但是，有一个临界点，超过这个临界点再进行调整将无助于事。达到这个临界点之后，您应该考虑修改目标和期望值。要更显著地提高性能，可能需要添加更多磁盘存储器、更快的 CPU、更多 CPU、更多主存储器、更快的通信链路或者它们的组合。

第 1 章 性能调整工具和方法

基准程序测试

基准程序测试是应用程序开发生命周期的常规组成部分。这是由应用程序开发者和数据库管理员（DBA）等团队成员参与的工作。

基准程序测试对系统执行，用于确定当前性能，并可用于提高应用程序的性能。即使已将应用程序代码编写得尽可能效率高，也可以通过调整数据库和数据库管理器配置参数来进一步提高性能。

可以通过不同类型的基准程序测试来发现特定类型的信息。例如:

- *基础结构基准程序*确定数据库管理器在特定受限实验室条件下的吞吐量能力。
- *应用程序基准程序*确定数据库管理器在更接近于生产环境的条件下的吞吐量能力。

进行基准程序测试以调整配置参数基于受控的条件。此类测试涉及在不断更改系统配置（并可能更改 SQL 语句）的情况下反复运行应用程序中的 SQL 语句，直到应用程序尽可能高效运行为止。

同一方法也可用于调整其他将对性能产生影响的因素，例如索引、表空间配置和硬件配置。

基准程序测试可以帮助您了解数据库管理器如何对各种不同条件作出反应。您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添加更多用户后事务执行速率的特征，甚至还可测试使用新发行版的数据库产品对应用程序产生的影响。

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。您可以通过在正常环境中运行测试应用程序入手。随着您缩小性能问题的范围，可以开发专用的测试用例，以便限制所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即可获得有价值的信息。从简单的评估开始，只有在必要时才提高复杂程度。

良好的基准程序应具有下列特征:

- 测试可重复
- 测试的每次迭代都在相同系统状态下开始
- 系统中不存在任何意外地处于活动状态的其他功能或应用程序
- 用于基准程序测试的硬件和软件与生产环境匹配

注意，已启动的应用程序即使处于空闲状态也会耗用内存。这将增大页面调度导致基准程序的运行结果产生偏差和违反可重复性条件的概率。

基准程序准备

在可以开始执行性能基准程序测试之前，必须满足一些先决条件。

在开始执行性能基准程序测试之前，请执行下列操作:

© Copyright IBM Corp. 2006, 2009

- 完成应用程序运行时所面向的数据库的逻辑设计和物理设计
- 创建表、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；请确保有适当的统计信息可用
- 进行规划，准备对具有生产环境中的大小的数据库运行，以使应用程序能够测试有代表性的内存需求；如果不可能做到这一点，那么请尝试确保可用系统资源量与数据量的比例在测试系统和生产系统中相同（例如，如果测试系统包含 10% 的数据，那么请使用可供生产系统使用的处理器时间的 10% 以及内存量的 10%）
- 将数据库对象置于其最终磁盘位置，确定日志文件大小，确定工作文件和备份映像的位置，测试备份过程
- 检查程序包，以确保在有可能的情况下启用性能选项，例如行分块

尽管基准程序测试期间可能会揭示应用程序的实际限制，但基准程序的用途是测量性能，而不是检测缺陷。

基准程序测试程序应该在能够准确代表最终生产环境的环境中运行。理想情况下，它应该在具有相同内存和磁盘配置的同一型号服务器上运行。如果该应用程序最终将服务于大量用户并处理大量数据，那么这一点尤其重要。操作系统以及基准程序测试程序直接使用的任何通信或存储工具先前也应调整完毕。

要进行基准程序测试的 SQL 语句应该是典型 SQL 语句或恶劣 SQL 语句，如下所述。

典型 SQL 语句

典型 SQL 语句包括正在进行基准程序测试的应用程序的典型操作期间执行的那些语句。所选择的语句将取决于该应用程序的性质。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要测试 FETCH、UPDATE 和多个 INSERT 语句。

恶劣 SQL 语句

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，在用户等候电话回音期间运行以便检索和更新客户信息的应用程序中的语句。
- 包含大量连接的语句或者应用程序中最复杂的语句。例如，为所有客户帐户生成每月活动综合报表的银行业务应用程序中的语句。一个公共表可能会列示客户的地址和帐号；但是，还必须连接其他多个表，以便处理和集成所有必需的帐户事务信息。
- 访问路径不良的语句，例如可用的索引所不支持的语句
- 执行时间过长的语句
- 仅在应用程序初始化期间执行，但资源需求与此不相称的语句。例如，生成当天必须处理的帐户工作列表的应用程序中的语句。该应用程序启动时，第一个主 SQL 语句将产生 7 路连接，这将创建一个非常大的列表，该列表包含此应用程序用户所负责的所有帐户。此语句每天可能只运行几次，但如果未将其调整好，那么它每次都要耗费几分钟才能运行完毕。

基准程序测试创建

在设计和实现基准程序测试程序时，您需要考虑各种因素。

由于测试程序的主要用途是模拟用户应用程序，所以程序的整体结构将有所变化。您可以将整个应用程序用作基准程序，并且只需引入某种方法对所要分析的 SQL 语句进行计时。对于大型应用程序或复杂应用程序而言，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，可以只将那些语句与必需的 CONNECT、PREPARE、OPEN、其他语句和计时机制一起包括在基准程序测试程序中。

另一个要考虑的因素是所要使用的基准程序的类型。其中一个选择是，在一个特定时间间隔内反复运行一组 SQL 语句。在此时间间隔内执行的语句数反映了应用程序的吞吐量。另一个选择是，只确定执行各个 SQL 语句所需的时间。

对于所有基准程序测试，您都需要一种可靠而适当的方法来测量耗用时间。要模拟其中的各个 SQL 语句以隔离方式执行的应用程序，最好测量每个语句的 PREPARE、EXECUTE 或者 OPEN、FETCH 或 CLOSE 时间。对于其他应用程序，测量从第一个 SQL 语句到 COMMIT 语句的事务时间可能更为合适。

尽管每个查询的耗用时间是性能分析中的一个重要因素，但却不一定会揭示瓶颈。例如，有关 CPU 使用情况、锁定和缓冲池 I/O 的信息可能表明应用程序遇到 I/O 限制，而不是 CPU 的使用达到满负荷。基准程序测试程序应该使您能够获取此类数据，以便有需要时进行更详细的分析。

并非所有应用程序都会将通过查询检索到的行集发送至某个输出设备。例如，该结果集可能是另一个应用程序的输入。格式化屏幕输出数据常常会产生很高的 CPU 成本，且可能无法反映用户需要。为了提供准确的模拟，基准程序测试程序应该反映应用程序的特定行处理活动。如果将行发送至输出设备，那么效率不高的格式化可能会耗用大量 CPU 时间，并且会误报 SQL 语句本身的实际性能。

虽然 DB2 命令行处理器（CLP）使用起来非常方便，但由于它将产生处理开销，因此并不适合于进行基准程序测试。在实例 `sqllib` 目录的 `bin` 子目录中提供了基准程序工具（db2batch）。此工具可以从平面文件或标准输入读取 SQL 语句，动态地准备和执行这些语句，然后返回结果集。它还使您能够对返回给 db2batch 的行数以及显示的行数进行控制。您可以指定所返回的性能信息的级别，其中包括耗用时间、处理器时间、缓冲池使用情况、锁定以及从数据库监视器收集的其他统计信息。如果正在对一组 SQL 语句进行计时，那么 db2batch 还将对性能结果进行汇总并提供算术和几何平均数。

通过将 db2batch 调用包装在 Perl 或 Korn shell 程序脚本中，您可以轻松方便地模拟多用户环境。请通过选择适当的 db2batch 选项确保连接属性（例如隔离级别）相同。

注意，在分区数据库环境中，db2batch 仅适合于测量耗用时间；返回的其他信息只与协调数据库分区中的活动相关。

您可以编写一个驱动程序，以帮助您进行基准程序测试。在 Linux® 或 UNIX® 系统上，可以使用 shell 程序来编写驱动程序。驱动程序可以执行基准程序、传递适当的参数、通过多次迭代来驱动该测试、将环境复原为一致的状态、使用新的参数值设置下一个测试以及收集和整合测试结果。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、分析结果以及提供给定测试的最佳参数值报告。

基准程序测试执行

在最常见的基准程序测试类型中，您选择一个配置参数并使用该参数的不同值运行该测试，直至达到最佳效果为止。

单个测试应该包括使用同一个参数值反复执行应用程序（例如，5 或 10 次迭代）。这使您能够获取更为可靠的平均性能值，以便将其他参数值所生成的结果与之进行比较。

您应该将第一次运行（称为“热身运行”）视为与后续运行（称为“常规运行”）有所不同。热身运行包括一些启动活动（例如初始化缓冲池），因此，完成热身运行所耗用的时间比常规运行略长。就统计而言，热身运行所获得的信息无效。在计算一组特定参数值的平均值时，将仅使用常规运行所生成的结果。通常，在计算平均值之前，最好先删除最大值和最小值。

为了最大程度地提高各次运行之间的一致性，请确保缓冲池在每次新运行后都恢复为已知的状态。执行测试时，在缓冲池中可能会装入数据，这将致使后续运行由于需要执行的磁盘 I/O 减少而提高速度。通过将其他无关的数据读入缓冲池，或者通过临时除去所有数据库连接来取消分配缓冲池，可以强制清除缓冲池的内容。

使用一组参数值完成测试之后，可以更改一个参数的值。在两次迭代之间，执行下列任务，以便将基准程序的环境复原为原始状态:

- 如果由于测试的需要更新了目录统计信息，那么请确保每次迭代都使用相同的统计值。
- 如果要在测试期间更新测试数据，那么这些数据必须一致。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。此方法使您能够只复原受影响的数据。应该对包含此数据的表和索引运行 REORG 和 RUNSTATS 实用程序。

概括而言，请执行以下步骤对数据库应用程序执行基准测试:

第一步 保留 DB2 注册表、数据库和数据库管理器配置参数以及缓冲池的标准建议值不变，这些值可以包括:

- 已知对于应用程序的无错误、正确执行而言必需的值
- 在先前调整期间能够提高性能的值
- AUTOCONFIGURE 命令提供的建议值
- 缺省值；但是，下列各项的缺省值可能不合适:
  - 对于工作负载和测试目标而言至关重要的参数
  - 日志大小（此大小应该在应用程序的单元测试和系统测试期间确定）
  - 任何必须进行更改才能使应用程序能够运行的参数

对此初始情况运行一组迭代，然后计算平均耗用时间、吞吐量或处理器时间。结果应该尽可能一致，理想情况下，各次运行之间的差别不到几个百分点。在各次运行之间显著变化的性能度量值可能会导致难以进行调整。

第二步 选择一种且唯一一种方法或调整参数进行测试，并更改它的值。

第三步 运行另一组迭代，然后计算平均耗用时间或处理器时间。

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，那么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。
- 如果性能下降或保持不变，那么将该参数返回为原来的值，返回至第二步并选择新的参数。重复此过程，直到所有参数都测试完毕为止。

基准程序测试分析示例

基准程序测试程序的输出应该包括每个测试的标识、迭代号、语句号以及每次执行的耗用时间。

注意，这些样本报告中的数据仅作演示之用。这些数据并不代表实际的测量结果。

基准程序测试结果的摘要可能类似于:

```
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 1. 样本基准程序测试结果

分析表明，完成 CONNECT（语句 01）耗用 1.34 秒，OPEN CURSOR（语句 10）耗用 2 分钟又 8.15 秒，FETCH（语句 15）返回 7 行并且延迟时间最长为 0.28 秒，CLOSE CURSOR（语句 20）耗用 0.84 秒，而完成 CONNECT RESET（语句 99）耗用 0.03 秒。

如果您的程序能够以定界 ASCII 格式输出数据，那么以后可以将该数据导入到数据库表或电子表格中，以便进行进一步统计分析。

基准程序摘要报告可能类似于:

```
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 2. 样本基准程序计时报告

第 2 章 性能监视工具和方法

对系统性能进行运作监视

运作监视是指，随着时间的推移定期收集关键的系统性能指标。此信息不仅提供了用于优化该初始配置以使其更适合于需求的关键数据，还使您为解决自行出现或者由于软件升级、数据量或用户数上升或者部署新应用程序而出现的新问题作好准备。

运作监视注意事项

运作监视策略需要考虑多个注意事项。

运作监视必须非常轻量级（不会大量耗用它所测量的系统的资源）并且一般化（广泛关注系统中任何位置可能出现的潜在问题）。

由于您计划在系统的生命周期内定期收集运作度量值，因此，能够通过某种方法管理所有那些数据至关重要。对于该数据的许多可能用法（例如监视性能的长期趋势）而言，您希望能够在有可能相隔多个月的任意数据集合之间进行比较。DB2 产品本身很好地简化了此类数据管理工作。对监视数据进行分析和比较变得非常直接，您已有一个健壮的基础结构来进行长期数据存储和组织。

DB2 数据库（“DB2”）系统提供了一些极佳的监视数据来源。主要来源包括快照监视器以及 DB2 版本 9.5 和更高版本中提供的工作负载管理（WLM）数据聚集表函数。这两个来源都侧重于摘要数据，即，计数器、计时器和柱形图之类的工具维护系统中活动的累积总计。通过随着时间的推移对这些监视元素进行采样，可以推断开始时间与结束时间之间发生的平均活动，这将透露出许多信息。

没有理由将您自己局限于 DB2 产品所提供的度量值。实际上，非 DB2 数据绝对不只“可有可无”。在确定性能问题时，上下文信息极为关键。用户、应用程序、操作系统、存储子系统和网络全都能够提供有关系统性能的宝贵信息。在生成有关系统性能的全面而整体的写照时，包括 DB2 数据库软件外部的度量值是重要的一环。

在最近的几个 DB2 数据库产品发行版中，有一个趋势，即通过 SQL 接口提供越来越多的监视数据供您使用。这使您能够非常直接地通过 DB2 来管理监视数据，其原因在于，可以方便地将该数据从管理视图重定向到其他位置，例如重定向回到 DB2 表。为了更深入地进行探查，还可以将事件和活动监视器数据写入 DB2 表，这将提供类似的益处。由于可以轻松方便地将非常多的监视数据存储在 DB2 中，在 DB2 中存储系统度量值（例如 vmstat 所返回的 CPU 利用率）所需的投资也变得小而可控。

为运作监视收集的数据类型

对于进行中的运作监视，最好收集多类数据。

- 一组基本的 DB2 系统性能监视度量值。
- DB2 配置信息

  定期创建数据库和数据库管理器配置、DB2 注册表变量和模式定义的副本有助于提供已进行的任何更改的历史记录，并有助于说明监视数据中出现的变化。

© Copyright IBM Corp. 2006, 2009

- 整体系统负载

  如果允许 CPU 或 I/O 利用率接近饱和，那么可能会引起仅仅使用 DB2 快照难以检测的系统瓶颈。因此，最好定期使用 vmstat 和 iostat（基于 UNIX 的系统）以及 perfmon（Windows®）来监视系统负载；对于基于 UNIX 的系统，还可以使用 netstat 来监视网络问题。您还可以使用管理视图（例如 ENV_SYS_RESOURCES）来检索与系统相关的操作系统信息、CPU 信息、内存信息和其他信息。通常，您查找系统中正常内容的变化，而不是查找特定通用值的变化。

- 在业务逻辑级测量的吞吐量和响应时间

  在 DB2 之上的业务逻辑级测量的应用程序性能视图具有与最终用户最为相关这一优点，并且通常包括所有可能会引起瓶颈的内容，例如表示逻辑、应用程序服务器、Web 服务器以及多个网络层等等。此数据对于服务级别协议（SLA）的设置或验证过程而言至关重要。

DB2 系统性能监视元素和系统负载数据已进行足够程度的压缩，即使您每 5 到 15 分钟收集一次这些数据，长期的总数据量在大多数系统中也无关紧要。同样，收集此数据的开销通常只会导致 CPU 时间耗用量增加 1% 到 3%，此成本对于保留重要系统度量值的持续历史记录而言一点都不高。通常，配置信息的更改频率相对较低，因此每天收集一次此信息已足够，并且不会导致数据量过大。

系统性能监视元素的基本集合

系统性能度量值的基本集合由大约 10 个度量值组成，可以在持续进行的运行监视工作中使用。

共有数以百计的度量值可供您选择，但收集全部这些度量值将生成相当多的数据，从而严重影响生产力。您需要具有下列特性的度量值:

- 易于收集 – 您不希望必须使用复杂或成本高昂的工具来执行日常监视，并且不希望监视活动对系统造成沉重负担。
- 易于理解 – 您不希望每次查看度量值时都必须查找该度量值的含义。
- 与系统相关 – 并非所有度量值在所有环境中都提供有意义的信息。
- 灵敏而不过分灵敏 – 度量值的变化应该指示系统中的实际变化；度量值不应自行波动。

这个入门集合由大约 10 个度量值组成:

- 已执行的事务数:

  ```
  TOTAL_COMMITS
  ```

  此度量值提供了有关系统活动的优秀的基本级别度量。

- 分别针对数据、索引和临时数据测量的缓冲池命中率:

  ```
  100 * (POOL_DATA_L_READS – POOL_DATA_P_READS) / POOL_DATA_L_READS
  100 * (POOL_INDEX_L_READS – POOL_INDEX_P_READS) / POOL_INDEX_L_READS
  100 * (POOL_TEMP_DATA_L_READS - POOL_TEMP_DATA_P_READS) / POOL_TEMP_DATA_L_READS
  100 * (POOL_TEMP_INDEX_L_READS - POOL_TEMP_INDEX_P_READS) /
  POOL_TEMP_INDEX_L_READS
  ```

缓冲池命中率是其中一个最基本的度量值，它提供有关系统利用内存来避免磁盘 I/O 的效率的重要整体度量。对于 OLTP 环境而言，数据命中率 80-85% 或更高以及索引命中率 90-95% 或更高通常被认为是良好情况，当然，您可以使用缓冲池快照中的数据针对各个缓冲池计算这些命中率。

虽然这些度量值通常很有用，但对于频繁执行大型表扫描的系统（例如数据仓库）而言，数据命中率通常相当低；其原因在于，数据被读入缓冲池之后，在被逐出以便为其他数据腾出空间之前不再被使用。

- 每个事务的缓冲池物理读写次数:

```
(POOL_DATA_P_READS + POOL_INDEX_P_READS +
  POOL_TEMP_DATA_P_READS + POOL_TEMP_INDEX_P_READS)
  / TOTAL_COMMITS

(POOL_DATA_WRITES + POOL_INDEX_WRITES)
  / TOTAL_COMMITS
```

这些度量值与缓冲池命中率紧密相关，但用途略有不同。虽然您可以考虑命中率的目标值，但每个事务的读写次数没有可能的目标。那么，为何要关心这些计算呢？这是因为，磁盘 I/O 在数据库性能方面是极为重要的因素，所以最好从多个角度对其进行审视。此外，这些计算包括写操作，而命中率只涉及读操作。最后，难以单独确定 94% 之类的索引命中率是否值得您尝试改进。如果每小时只有 100 次逻辑索引读操作，并且其中的 94 次在缓冲池内完成，那么为了使另外 6 次避免转变为物理读而耗费时间并不明智。但是，如果伴随 94% 索引命中率的统计信息表明每个事务都执行 20 次物理读（这可以进一步分为数据读和索引读以及常规读和临时读），那么缓冲池命中率可能值得您进行调查。

这些度量值不仅仅是物理读写次数，并且已针对每个事务进行规范化。此趋势是通过许多度量值形成的。其目的在于，使度量值与所收集数据的时间长度无关，并且与系统在该时间的繁忙程度无关。通常，这有助于确保获取类似的度量值，而不考虑收集监视数据的方式和时间。使数据收集操作在计时和持续时间方面具有一定程度的一致性确实不错；但是，规范化使其不再成为优良措施的关键所在。

- 读取数据库行数与选择数据库行数之比:

```
ROWS_READ / ROWS_RETURNED
```

此计算指示为了查找符合条件的行而从数据库表中读取的平均行数。数值较小表明查找数据的效率很高，这通常表示正在有效地使用索引。例如，在系统执行许多表扫描，并且需要检查数百万行才能确定它们是否符合结果集要求的情况下，此数值可能非常大。另一方面，通过全限定唯一索引来访问表时，此统计值可能非常小。纯索引访问方案（不需要从表中读取任何行）不会导致 ROWS_READ 增大。

在 OLTP 环境中，此度量值通常不会大于 2 或 3，这表明大部分访问操作通过索引而非表扫描完成。此度量值是监视方案在一段时间内的稳定性的简单方法 – 此值意外增大通常表明不再使用索引，在这种情况下，您应该进行调查。

- 每个事务的排序操作耗用时间量:

```
TOTAL_SORT_TIME / TOTAL_COMMITS
```

这是一种很有效率的排序统计信息处理方法，其原因在于，由于排序溢出而造成的任何额外开销都将自动地包括在此计算中。即，您可能还想收集 TOTAL_SORTS 和 SORT_OVERFLOWS 以便于进行分析，当系统中存在排序问题的历史记录时尤其如此。

- 每 1000 个事务的累积锁定等待时间量:

  ```
  1000 * LOCK_WAIT_TIME / TOTAL_COMMITS
  ```

  锁定等待时间过长通常表现为响应速度较慢，因此，监视此时间十分重要。由于单一事务的锁定等待时间通常非常小，因此，此值按 1000 个事务为单位进行规范化。放大到 1000 个事务仅仅是为了使测量值更易于处理。

- 每 1000 个事务的死锁和锁定超时次数:

  ```
  1000 * (DEADLOCKS + LOCK_TIMEOUTS) / TOTAL_COMMITS
  ```

  虽然死锁在大多数生产系统中相对罕见，但锁定超时情况可能比较常见。应用程序通常必须以类似方式对其进行处理：从头开始重新执行事务。监视这种情况的发生比率有助于避免由于死锁过多或锁定超时过长而大幅增加系统负载但不被 DBA 所了解的情况。

- 每 1000 个事务的脏窃用触发器数目:

  ```
  1000 * POOL_DRTY_PG_STEAL_CLNS / TOTAL_COMMITS
  ```

  “脏窃用”是最不可取的触发器缓冲池清除方法。实际上，将所清除页的更新写入磁盘时，需要新缓冲池页的 SQL 语句的处理将被中断。如果允许频繁地发生脏窃用，那么会对吞吐量和响应时间产生显著的影响。

- 每 1000 个事务的程序包高速缓存插入次数:

  ```
  1000 * PKG_CACHE_INSERTS / TOTAL_COMMITS
  ```

  程序包高速缓存插入操作是系统的常规执行过程的组成部分；但是，如果此数目较大，那么表明耗用 CPU 时间过多。在许多设计精良的系统中，系统以稳定状态运行之后，执行的程序包高速缓存插入操作将非常少，这是因为系统将使用或重复使用静态 SQL 语句或者先前准备的动态 SQL 语句。在临时性动态 SQL 语句的流量较高的系统中，SQL 语句编译和程序包高速缓存插入操作不可避免。但是，此度量值用于监视第三类情况，即，应用程序未重复使用已准备的语句或者未在频繁执行的 SQL 语句中使用参数标记，从而意外导致程序包高速缓存内容不断流失。

- 代理程序等待日志记录被清仓到磁盘时耗用的时间:

  ```
  LOG_WRITE_TIME
    / TOTAL_COMMITS
  ```

  事务日志很有可能由于活动级别较高、配置不正确或者其他原因而成为系统瓶颈。通过监视日志活动，您可以检测 DB2 端引起的问题（应用程序驱动的日志请求数增加）或系统端引起的问题（通常是由于硬件或配置问题导致日志子系统性能下降）。

- 分区数据库环境中两个分区之间发送和接收的快速通信管理器（FCM）缓冲区的数目:

  ```
  FCM_SENDS_TOTAL，FCM_RECVS_TOTAL
  ```

  这些度量值提供集群中不同分区之间的数据流的速率，尤其是流是否平衡。从不同分区接收到的缓冲区数目如果有显著的差别，那么可能表明散列到每个分区的数据量不均匀。

分区数据库环境中的跨分区监视

在上面提到的各个监视元素值中，几乎每一个都是以分区为基础报告的。

通常，您期望大多数监视统计信息在同一个 DB2 分区组中的所有分区之间相当一致。显著的差别可能表明存在数据不均匀情况。要跟踪的跨分区比较样本包括:

- 数据、索引和临时表的逻辑和物理缓冲池读次数
- 分区级以及大型表的读取行数
- 排序时间以及排序溢出次数
- 发送和接收 FCM 缓冲区的数目
- CPU 和 I/O 利用率

监视数据中的异常值

对性能问题进行故障诊断时，能够确定异常值是解释系统性能监视数据的关键。

如果监视元素的值比正常情况差（即，值不正常），那么它将提供有关性能问题性质的线索。通常，较差的值是指高于预期的值，例如较长的锁定等待时间。但是，异常值也可能是低于预期的值，例如较低的缓冲池命中率。根据情况的不同，您可以使用一种或多种方法来确定某个值是否比正常情况差。

一种方法是，依赖于业界的首选法则或最佳实践。例如，其中一条首选法则是，对于 OLTP 环境，数据的缓冲池命中率 80-85% 或更高通常被认为不错。注意，此首选法则适用于 OLTP 环境，对于数据仓库而言并不是有用的准则（数据仓库的数据命中率通常由于系统的性质而远低于此值）。

另一种方法是，将当前值与先前收集的基线值作比较。此方法通常最为明确并依赖于您有足够的操作监视策略来收集和存储正常情况下的关键性能指标。例如，您可能会注意到，当前缓冲池命中率是 85%。根据业界规范，这将被认为是正常的值，但与报告性能问题前记录的值 99% 相比却不正常。

最后一种方法是，将当前值与相似系统上的当前值作比较。例如，如果相似系统上的缓冲池命中率为 99%，那么当前缓冲池命中率 85% 将被认为不正常。

GOVERNOR 实用程序

控制器用于监视对数据库运行的应用程序的行为并可以更改该行为，这取决于控制器配置文件中指定的规则。

要点: 由于 DB2 版本 9.5 引入了新的策略性 DB2 工作负载管理功能部件，因此版本 9.7 建议您不要使用 DB2 GOVERNOR 实用程序，将来的发行版可能会将其除去。有关不推荐使用GOVERNOR 实用程序的更多信息，请参阅“建议您不要使用 DB2 控制器和 Query Patroller”。要了解更多关于 DB2 工作负载管理器及其如何替换 GOVERNOR 实用程序的信息，请参阅“DB2 工作负载管理器概念简介”和“关于 DB2 工作负载管理器的常见问题”。

控制器实例由一个前端实用程序和一个或多个守护程序组成。控制器的每个实例都特定于数据库管理器的一个实例。缺省情况下，启动控制器时，将对分区数据库的每个数据库分区启动一个控制器守护程序。但是，您可以指定对所要监视的单一数据库分区启动守护程序。

控制器按照控制器配置文件中的规则来管理应用程序事务。例如，应用某个规则可能会表明应用程序正在使用过多的特定资源。规则还将指定要执行的操作，例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，那么控制器将在发生资源违例的数据库分区中更改代理程序的优先级。在分区数据库中，如果强制将应用程序与数据库断开连接，那么将执行该操作，即使检测到违例的守护程序正在该应用程序的协调程序节点上运行也是如此。

控制器将记录它所执行的任何操作。

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，请增大控制器唤醒时间间隔以降低其 CPU 使用率。

启动和停止控制器

GOVERNOR 实用程序监视已连接至数据库的应用程序，并根据您在该数据库的控制器配置文件中指定的规则来更改那些应用程序的行为。

要点: 由于 DB2 版本 9.5 中引入了新的工作负载管理功能部件，所以版本 9.7 中已经不推荐使用 DB2 控制器实用程序，并可能在以后的发行版中将其除去。 有关更多信息，请参阅《DB2 版本 9.7 新增内容》一书中的“已经不推荐使用 DB2 控制器和 Query Patroller”主题。

在启动控制器之前，您必须创建控制器配置文件。

要启动控制器，您必须具有 *sysadm* 或 *sysctrl* 权限。

要启动控制器，请使用 db2gov 命令并指定下列必需参数:

START *database-name*
指定的数据库名称必须与控制器配置文件中的数据库名称匹配。

config-file
此数据库的控制器配置文件的名称。如果该文件不在缺省位置（sqllib 目录）中，那么必须包括文件路径和文件名。

log-file
此控制器的日志文件的基本名称。对于分区数据库而言，对于每个在其中为此控制器实例运行守护程序的数据库分区，将添加该数据库分区的分区号。

要对分区数据库的单个数据库分区启动控制器，请指定 **dbpartitionnum** 选项。

例如，要对数据库 SALES 的数据库分区 3 启动控制器，并使用名为 salescfg 的配置文件以及名为 saleslog 的日志文件，请输入以下命令:

```
db2gov start sales dbpartitionnum 3 salescfg saleslog
```

要对所有数据库分区启动控制器，请输入以下命令:

```
db2gov start sales salescfg saleslog
```

控制器守护程序

控制器守护程序用于收集关于对数据库运行的应用程序的信息。

控制器守护程序在启动后将运行以下任务循环。

1. 守护程序检查其控制器配置文件是否已更改或者是否尚未读取该文件。如果满足任何一个条件，那么该守护程序将读取该文件中的规则。这允许您在控制器守护程序运行时更改其行为。

2. 守护程序请求收集关于每个正在使用数据库的应用程序和代理程序的资源使用统计信息的快照信息。
3. 守护程序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。如果某个规则适用，那么控制器将执行指定的操作。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，那么与该违例相关的规则将在下一个控制器时间间隔内应用于该应用程序。
4. 守护程序对于它执行的任何操作，都将在控制器日志文件中写入一条记录。

当控制器完成其任务后，它按照配置文件中指定的时间间隔进行休眠。该时间间隔经过之后，控制器将唤醒并再次开始任务循环。

如果控制器遇到错误或停止信号，那么它在停止之前将先执行清理处理。通过使用已设置其优先级的应用程序的列表，清理处理将复位所有应用程序代理程序的优先级。然后，它将复位任何不再处理应用程序的代理程序的优先级。这确保在控制器结束后，代理程序不会一直以非缺省优先级运行。如果发生错误，那么控制器将一条消息写入管理通知日志，以指示它异常结束。

如果 **agentpri** 数据库管理器配置参数的值不是系统缺省值，那么无法使用控制器来调整代理程序优先级。

尽管控制器守护程序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。因为控制器守护程序可以发出快照请求，所以它可以检测数据库管理器何时结束。

控制器配置文件

控制器配置文件包含用于控制对数据库运行的应用程序的规则。

控制器对每条规则进行求值并在规则求值为 true 时执行指定的操作。

控制器配置文件包含常规的子句，这些子句用于标识所要监视的数据库（必需）、以何时间间隔写包含 CPU 使用统计信息的帐户记录以及控制器守护程序的休眠时间间隔。配置文件还可以包含一个或多个可选的应用程序监视规则语句。下列准则既适用于常规子句也适用于规则语句:

- 使用花括号（{ }）对常规注释进行定界。
- 在大多数情况下，请使用大写、小写或大小写混合字符来指定值。在 `applname` 子句后面指定的应用程序名例外，此名称区分大小写。
- 使用分号（;）来终止每个常规子句或规则语句。

如果需要更新规则，那么请编辑配置文件，而不必停止控制器。每个控制器守护程序都将检测到该文件已更改并重新读取该文件。

在分区数据库环境中，必须在一个跨所有数据库分区安装的目录中创建控制器配置文件，以使每个数据库分区中的控制器守护程序都可以读取同一配置文件。

常规子句

下列子句不能在控制器配置文件中多次指定。

dbname

要监视的数据库的名称或别名。此子句是必需的。

account *n*

以分钟计的时间间隔，在此时间间隔经过后，将写包含每个连接的 CPU 使用统计信息的帐户记录。此选项在 Windows 操作系统上不可用。在某些平台上，无法从快照监视器获取 CPU 统计信息。在这种情况下，account 子句将被忽略。

如果一个短期会话完全发生在帐户时间间隔内，那么不会写任何日志记录。所写的日志记录包含 CPU 统计信息，这些信息反映自从该连接的上一个日志记录以来 CPU 的使用情况。如果停止并接着重新启动控制器，那么将在两个日志记录中反映 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识别。

interval *n*

以秒计的时间间隔，在此时间间隔经过后，守护程序将唤醒。如果未指定此子句，那么将使用缺省值，即 120 秒。

规则子句

规则语句指定如何控制应用程序，这些语句由称为“规则子句”的小型组件组成。如果使用规则子句，那么必须按特定顺序在规则语句中输入这些子句，如下所示:

1. **desc**: 关于规则的注释，括在单引号或双引号中
2. **time**: 规则的求值时间
3. **authid**: 应用程序执行语句时采用的一个或多个授权标识
4. **applname**: 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。如果应用程序名包含空格，那么必须将该名称括在双引号中。
5. **setlimit**: 对控制器检查进行限制；例如，只检查 CPU 时间、所返回的行数或者空闲时间。在某些平台上，无法从快照监视器获取 CPU 统计信息。在这种情况下，setlimit 子句将被忽略。
6. **action**: 达到限制时要执行的操作。如果未指定操作，那么达到限制时，控制器会将为应用程序工作的代理程序的优先级降低 10。可以对应用程序执行的操作包括降低其代理程序优先级、强制它与数据库断开连接或者为它的操作设置调度选项。

对规则子句进行组合以构成规则语句，并且，特定子句在每条语句中不能使用多次。

```
desc "Allow no UOW to run for more than an hour"
setlimit uowtime 3600 action force;
```

如果有多条规则适用于某个应用程序，那么将应用所有这些规则。通常，与首先遇到的限制相关联的操作是第一个要应用的操作。对规则子句指定值 -1 时的情况例外: 以后对同一子句指定的值只能覆盖先前指定的值；先前规则语句中的其他子句仍有效。

例如，一条规则语句使用 rowssel 100000 和 uowtime 3600 子句来指定: 如果应用程序的耗用时间大于 1 小时，或者它选择 100000 行以上的数据，那么应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的耗用时间不受限制。在这种情况下，即使该应用程序运行 1 小时以上，也不会更改其优先级。也就是说，uowtime -1 覆盖 uowtime 3600。但是，如果它选择 100000 行以上的数据，那么由于 rowssel 100000 仍有效，因此它的优先级将下降。

规则的应用顺序

控制器按从上到下顺序处理配置文件中的规则。但是，如果特定规则语句中的 setlimit 子句比先前规则语句中的同一子句更宽松，那么将应用限制性较强的规则。在以下示例中，仍将 ADMIN 限制为 5000 行，原因是第一条规则的限制性更强。

```
desc "Force anyone who selects 5000 or more rows."
setlimit rowssel 5000 action force;

desc "Allow user admin to select more rows."
authid admin setlimit rowssel 10000 action force;
```

为了确保限制性较弱的规则能够覆盖限制较强的先前规则，请先指定 -1 以清除先前的规则，然后再应用新规则。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，第三条规则对 ADMIN 将限制重设为 10000 行。

```
desc "Force anyone who selects 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;
```

控制器配置文件示例

```
{ The database name is SAMPLE; do accounting every 30 minutes;
  wake up once a second. }
dbname sample; account 30; interval 1;

desc "CPU restrictions apply to everyone 24 hours a day."
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 the 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 department."
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 the DB2 CLP by the novice user."
authid novice
applname db2bp.exe
setlimit cpu 5 locks 100 rowssel 250;

desc "During the day, 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 the 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 "Increase the priority of an important application so it always
      completes quickly."
applname V1app setlimit cpu 1 locks 1 rowssel 1 action priority -20;

desc "Some people, such as the database administrator (and others),
     should not be limited. Because 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;
```

控制器规则子句

控制器配置文件中的每条规则都由一些子句组成，这些子句指定了规则所适用的条件以及导致该规则求值为真的操作。

规则子句必须按以下所示顺序指定。

可选的开始子句

desc 指定此规则的描述。此描述必须括在单引号或双引号中。

time 指定对此规则进行求值的时间段。必须按以下格式来指定时间段: `time hh:mm hh:mm`; 例如 `time 8:00 18:00`。如果未指定此子句，那么每天 24 小时内都将对此规则进行求值。

authid 指定应用程序在执行时使用的一个或多个授权标识。如果指定多个授权标识，那么必须用逗号（,）进行分隔; 例如: `authid gene, michael, james`。如果未指定此子句，那么此规则将应用于所有授权标识。

applname

指定连接到数据库的可执行文件或对象文件的名称。如果指定多个应用程序名，那么必须用逗号（,）进行分隔; 例如: `applname db2bp, batch, geneprog`。如果未指定此子句，那么此规则将应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。您应该确保要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。控制器配置文件中指定的应用程序名将被截断为 20 个字符，以便与其内部表示匹配。

限制子句

setlimit

指定控制器要检查的一个或多个限制。这些限制必须是 -1 或大于 0（例如，`cpu -1 locks 1000 rowssel 10000`）。您必须至少指定一个限制，并且，任何在规则语句中未指定的限制都不受该规则限制。控制器可以检查下列限制:

cpu *n* 指定应用程序可耗用的 CPU 秒数。如果指定 -1，那么应用程序的 CPU 使用情况不受限制。

idle *n* 指定允许连接处于空闲状态的秒数。如果指定 -1，那么连接的空闲时间不受限制。

注: 某些数据库实用程序（例如 BACKUP 和 RESTORE）与数据库建立连接，然后通过引擎可分配单元（EDU）执行控制器不可视的工作。这些数据库连接看起来处于空闲状态，并且可能会超出空闲时间限制。为了防止控制器对这些实用程序执行操作，请通过调用这些实用程序的授权标识对它们指定 -1。例如，为了防止控制器对使用授权标识 DB2SYS 运行的实用程序执行操作，请指定 `authid DB2SYS setlimit idle -1`。

locks *n*

指定应用程序可挂起的锁定数。如果指定 -1，那么应用程序挂起的锁定数不受限制。

rowsread *n*

指定应用程序可选择的行数。如果指定 -1，那么可选择的行数不受限制。可以指定的最大值是 4294967298。

注: 此限制与 `rowssel` 不同。区别在于，`rowsread` 是为了返回结果集而必须读取的行数。此数目包括引擎在目录表中读取的行数，使用索引时，此数目将下降。

rowssel *n*

指定可以返回给应用程序的行数。只有在协调数据库分区中，此值才是非零值。如果指定 -1，那么可以返回的行数不受限制。可以指定的最大值是 4294967298。

uowtime *n*

指定工作单元（UOW）从第一次进入活动状态开始可耗用的秒数。如果指定 -1，那么耗用时间不受限制。

注: 如果已使用 sqlmon API 来停用工作单元监视开关或时间戳记监视开关，那么这将影响控制器根据工作单元耗用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系统的信息。如果应用程序的工作单元（UOW）在控制器启动之前就已经启动，那么控制器将不会控制该 UOW。

操作子句

action 指定超出一个或多个指定的限制后要执行的操作。如果超出限制，并且未指定 `action` 子句，那么控制器会将为该应用程序工作的代理程序的优先级降低 10。

force 指定对为应用程序提供服务的代理程序执行强制操作。（FORCE APPLICATION 命令将终止协调代理程序。）

注: 在分区数据库环境中，仅当正在应用程序的协调数据库分区中运行控制器守护程序时，才会执行 `force` 操作。因此，如果正在分区 A 中运行控制器守护程序，而某些协调数据库分区为数据库分区 B 的应用程序超出限制，那么将跳过 `force` 操作。

nice *n* 指定为应用程序工作的代理程序的相对优先级更改。有效值的范围是 -20 到 +20（对于基于 UNIX 的系统）和 -1 到 6（对于 Windows 平台）。

- 在基于 UNIX 的系统上，必须将 **agentpri** 数据库管理器配置参数设置为缺省值；否则，它将覆盖 `nice` 值。

- 在 Windows 平台上，可以将 **agentpri** 数据库管理器配置参数与 `nice` 值配合使用。

可以使用控制器来控制在缺省用户服务超类 SYSDEFAULTUSERCLASS 中运行的应用程序的优先级。如果使用控制器来降低在此服务超类中运行的应用程序的优先级，那么代理程序将解除与其出站相关因子的关联（如果它与某个出站相关因子相关联的话），并且将根据控制器指定的代理程序优先级来设置其相对优先级。您无法使用控制器来更改用户定义的服务超类和子类中的代理程序优先级。而是，必须使用该服务超类或子类的代理程序优先级设置来控制在这些服务类中运行的应用程序。但是，可以使用控制器在任何服务类中强制连接。

注: 在 AIX® 5.3 上，实例所有者必须具有 CAP_NUMA_ATTACH 能力才能提高为应用程序工作的代理程序的相对优先级。要授予此能力，请以 root 用户身份登录并运行以下命令:

```
chuser capabilities=CAP_NUMA_ATTACH,CAP_PROPAGATE
```

schedule [class]

进度调度将提高为应用程序工作的代理程序的优先级。其目标在于最大程度地缩短平均响应时间，同时在应用程序之间保持平稳。

控制器根据以下条件来选择前几个要调度的应用程序:

- 挂起锁定数目最多的应用程序（尝试减少锁定等待数目）
- 最旧的应用程序
- 估计的剩余运行时间最短的应用程序（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）

每个条件中排名前三位的应用程序比所有其他应用程序具有更高的优先级。即，使每个条件组中排名第一的应用程序具有最高的优先级，使排名第二的应用程序具有第二高的优先级，并使排名第三的应用程序具有第三高的优先级。如果某个应用程序在多个条件中排名前三位，那么它将具有排名最高的那个条件的相应优先级，而另一条件中排名次高的应用程序将具有次高优先级。例如，如果应用程序 A 挂起的锁定最多，但具有第三短的估计剩余运行时间，那么它将具有第一个条件的最高优先级。在最短估计剩余运行时间条件中排名第四的应用程序将具有该条件的第三高优先级。

此控制器规则选择的应用程序最多分为三类。对于每一类，控制器根据上述条件选择 9 个应用程序，它们是每个类中排名前三位的应用程序。如果您指定了 `class` 选项，那么此规则选择的所有应用程序都将被看作单个类，并且将选择 9 个应用程序并使它们具有较高的优先级，如上所述。

如果在多个控制器规则中选择了某个应用程序，那么该应用程序由最后一个选择该应用程序的规则控制。

注: 如果已使用 sqlmon API 来停用该语句开关，那么这将影响控制器根据语句耗用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，那么将对整个实例关闭这些开关，并且控制器将不再接收到此信息。

调度操作可以:

- 确保不同组中的每个应用程序都获得时间，而不必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（3 个短、5 个中等长度和 6 个长）同时运行，那么它们可能会因为共用 CPU 而导致响应时间都很长。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。通过使用优先级，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以及一条针对长应用程序的规则。

  以下示例给出了一个控制器配置文件的部分内容以阐明这一点:

  ```
  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）中的每个组都获得相同的优先级。如果一个组正在运行大量的应用程序，那么管理员可以确保其他组仍能够获得合理的响应时间，以运行他们的应用程序。例如，对于涉及到三个部门（财务、库存和计划）的情况，可以将所有“财务”用户置于一个组，将所有“库存”用户置于第二个组，并将所有“计划”用户置于第三个组。处理能力将在这三个部门之间或多或少地均分。

  以下示例给出了一个控制器配置文件的部分内容以阐明这一点:

  ```
  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应用程序。

  如果未指定 `class` 选项，那么控制器将根据调度操作下活动应用程序的数目来创建自己的类，并且将根据查询编译器为该应用程序运行的查询所作的成本估计将这些应用程序置于不同的类。管理员可以通过不限定所选择的应用程序（即，不指定 `applname`、`authid` 或 `setlimit` 子句）来调度所有应用程序。

控制器日志文件

每当控制器守护程序执行操作时，它将一条记录写入它的日志文件。

操作包括下列各项:

- 启动或停止控制器
- 读取控制器配置文件
- 更改应用程序的优先级
- 强制应用程序
- 遇到错误或警告

每个控制器守护程序都有一个不同的日志文件，以防由于多个控制器守护程序同时尝试写同一个文件而产生文件锁定瓶颈。要查询控制器日志文件，请使用 db2govlg 命令。

日志文件存储在 sqllib 目录的 log 子目录中，但在 Windows 操作系统上除外，在该操作系统中，log 子目录位于 Windows 操作系统用于存储应用程序日志文件的 Common Application Data 目录下。在使用 db2gov 命令来启动控制器时，应提供日志文件的基本名称。请确保日志文件名包含数据库名称，以便对每个受控制的数据库分区上的日志文件进行区分。为了确保分区数据库环境中的每个控制器都使用唯一的文件名，在日志文件名中，将自动追加控制器守护程序运行所在的数据库分区的编号。

日志文件记录格式

日志文件中的每个记录都具有以下格式:

```
Date Time DBPartitionNum RecType Message
```

Date 和 *Time* 字段的格式为 *yyyy-mm-dd-hh.mm.ss*。您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。*DBPartitionNum* 字段包含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记录的类型。可以记录的值包括:

- ACCOUNT: 应用程序记帐统计信息
- ERROR: 发生错误
- FORCE: 已强制应用程序
- NICE: 已更改应用程序的优先级
- READCFG: 控制器已读取配置文件
- SCHEDGRP: 代理程序优先级发生更改
- START: 控制器已启动
- STOP: 控制器已停止
- WARNING: 发生警告

下面对这些值中的部分值作更详细的描述。

ACCOUNT

在下列情况下，将写 ACCOUNT 记录:

- 自从上次为应用程序写 ACCOUNT 记录以来，此应用程序的 **agent_usr_cpu_time** 或 **agent_sys_cpu_time** 监视元素值已更改。
- 应用程序不再处于活动状态。

ACCOUNT 记录具有以下格式:

```
<auth_id> <appl_id> <applname> <connect_time> <agent_usr_cpu_delta>
<agent_sys_cpu_delta>
```

ERROR

当控制器守护程序需要关闭时，将写 ERROR 记录。

FORCE

当控制器根据控制器配置文件中的规则来强制应用程序时，将写 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数据库分区的编号。

cfg_line

指定控制器配置文件中导致强制应用程序的规则所在的行号。

restriction_exceeded

提供有关违反规则情况的详细信息。有效值为:

- CPU: 应用程序 USR CPU 时间与 SYS CPU 时间的总计（以秒计）
- Locks: 应用程序挂起的锁定的总数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序处于空闲状态的时间长度
- ET: 自从应用程序的当前工作单元启动以来的耗用时间（已超出 `uowtime setlimit`）

NICE 当控制器根据控制器配置文件中的规则来更改应用程序的优先级时，将写 NICE 记录。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

指定要应用于应用程序的代理程序进程优先级值的增量或减量。

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

提供有关违反规则情况的详细信息。有效值为:

- CPU: 应用程序 USR CPU 时间与 SYS CPU 时间的总计（以秒计）
- Locks: 应用程序挂起的锁定的总数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序处于空闲状态的时间长度
- ET: 自从应用程序的当前工作单元启动以来的耗用时间（已超出 `uowtime setlimit`）

SCHEDGRP

将应用程序添加到调度组或者将其从一个调度组移至另一个调度组时，将写 SCHEDGRP 记录。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

提供有关违反规则情况的详细信息。有效值为:

- CPU: 应用程序 USR CPU 时间与 SYS CPU 时间的总计（以秒计）
- Locks: 应用程序挂起的锁定的总数
- Rowssel: 应用程序选择的总行数
- Rowsread: 应用程序读取的总行数
- Idle: 应用程序处于空闲状态的时间长度
- ET: 自从应用程序的当前工作单元启动以来的耗用时间（已超出 `uowtime setlimit`）

START

当控制器启动时，将写 START 记录。START 记录具有以下格式:

```
Database = <database_name>
```

STOP 当控制器停止时，将写 STOP 记录。此记录具有以下格式:

```
Database = <database_name>
```

WARNING

在下列情况下，将写 WARNING 记录:

- 已调用 sqlefrce API 以强制应用程序，但该 API 返回了正数 SQLCODE。
- 快照调用返回了并非 1611（SQL1611W）的正数 SQLCODE。
- 快照调用返回了并非 -1224（SQL1224N）或 -1032（SQL1032N）的负数 SQLCODE。这些返回码在先前活动的实例关闭后出现。
- 在基于 UNIX 的环境中，已尝试安装信号处理程序，但该尝试失败。

由于写入的是标准值，因此您可以在日志文件中查询不同类型的操作。*Message* 字段提供其他非标准信息，这些信息随记录类型的不同而有所变化。例如，FORCE 或 NICE 记录的 *Message* 字段包含应用程序信息，而 ERROR 记录的该字段包含错误消息。

控制器日志文件可能与以下示例类似:

```
2007-12-11-14.54.52    0 START      Database = TQTEST
2007-12-11-14.54.52    0 READCFG    Config = /u/db2instance/sqllib/tqtest.cfg
2007-12-11-14.54.53    0 ERROR      SQLMON Error: SQLCode = -1032
2007-12-11-14.54.54    0 ERROR      SQLMONSZ Error: SQLCode = -1032
```

停止控制器

GOVERNOR 实用程序监视已连接至数据库的应用程序，并根据您在该数据库的控制器配置文件中指定的规则来更改那些应用程序的行为。

要点: 由于 DB2 版本 9.5 中引入了新的工作负载管理功能部件，所以版本 9.7 中已经不推荐使用 DB2 控制器实用程序，并可能在以后的发行版中将其除去。 有关更多信息，请参阅《DB2 版本 9.7 新增内容》一书中的“已经不推荐使用 DB2 控制器和 Query Patroller”主题。

要停止控制器，您必须具有 *sysadm* 或 *sysctrl* 权限。

要停止控制器，请使用 db2gov 命令并指定 STOP 选项。

例如，要对 SALES 数据库的所有数据库分区停止控制器，请输入以下命令:

```
db2gov STOP sales
```

要仅对数据库分区 3 停止控制器，请输入以下命令:

```
db2gov START sales nodenum 3
```

第 3 章 影响性能的因素

系统体系结构

DB2 体系结构和进程概述

在客户机端，本地或远程应用程序与 DB2 客户机库链接。本地客户机使用共享内存和信号进行通信；远程客户机使用协议（例如命名管道（NPIPE）或 TCP/IP）进行通信。在服务器端，活动由引擎可分派单元（EDU）控制。

图 3 显示了 DB2 体系结构和进程的一般概述。

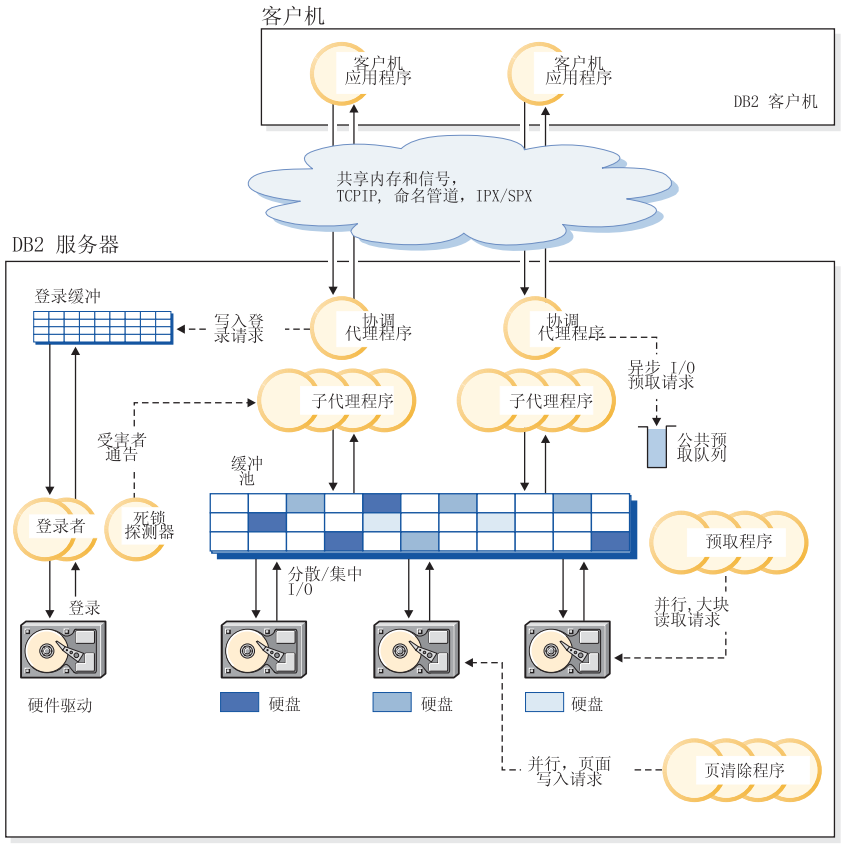


图 3. 客户机连接和数据库服务器组件

EDU 显示为圆圈或圆圈组。

© Copyright IBM Corp. 2006, 2009

EDU 在所有平台上都作为线程实现。DB2 代理程序是最常见的 EDU 类型。这些代理程序代表应用程序执行大量 SQL 和 XQuery 处理。其他常见的 EDU 包括预取程序和页清除程序。

可以指定一组子代理程序来处理客户机应用程序请求。如果服务器所在的机器包含多个处理器或者是分区数据库环境的组成部分，那么可以指定多个子代理程序。例如，在对称多处理（SMP）环境中，多个 SMP 子代理程序可以利用多个处理器。

所有代理程序和子代理程序都由一个共享算法管理，该算法将最大程度地减少创建和破坏 EDU 的操作。

缓冲池是数据库服务器内存中的一个区域，用户数据页、索引数据页和目录数据页被临时地移至该区域，并可以在该处被修改。由于访问内存中的数据比访问磁盘中的数据快得多，因此缓冲池是数据库性能的重要因素。

缓冲池以及预取程序和页清除程序 EDU 的配置决定了应用程序能够多快地访问数据。

- *预取程序*在应用程序需要数据之前从磁盘检索该数据，并将其移入缓冲池。例如，如果没有数据预取程序，那么需要扫描大量数据的应用程序将必须等待数据从磁盘移入缓冲池。应用程序的代理程序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将请求的页从磁盘读入缓冲池，从而实现那些请求。如果使用多个磁盘来存储数据，那么可以采用条带分割技术将数据分布到那些磁盘上。条带分割技术使预取程序能够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。页清除程序是独立于应用程序代理程序的后台 EDU。它们将查找已被修改的页，并将那些已更改的页写入磁盘。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，那么应用程序代理程序将必须执行缓冲池与磁盘存储器之间的所有数据读取和写入操作。

DB2 进程技术模型

DB2 进程技术模型方面的知识可以帮助您理解数据库管理器与其相关联的组件的交互方式，并且可以帮助您在发生问题时进行故障诊断。

所有 DB2 数据库服务器使用的进程技术模型都旨在简化数据库服务器与客户机之间的通信。它还确保数据库应用程序独立于数据库控制块和关键数据库文件之类的资源。

DB2 数据库服务器必须执行各种不同的任务，例如处理数据库应用程序请求或确保将日志记录写入磁盘。通常，每项任务都由一个独立的*引擎可分派单元*（EDU）执行。

采用多线程体系结构对于 DB2 数据库服务器而言有很多优点。由于同一进程内的所有线程可以共享一些操作系统资源，因此，新线程需要的内存和操作系统资源比进程要少。此外，在某些平台上，线程的上下文切换时间比进程短，这有助于提高性能。在所有平台上使用线程模型使得 DB2 数据库服务器更易于配置，因为这样更容易根据需要分配更多 EDU，并且可以动态分配必须由多个 EDU 共享的内存。

对于正在访问的每个数据库，将启动不同的 EDU 以处理各种数据库任务，例如预取、通信和日志记录。数据库代理程序是一类特殊的 EDU，创建它们是为了处理应用程序对数据库的请求。

通常，您可以依靠 DB2 数据库服务器来管理 EDU 集合。但是，也可以通过一些 DB2 工具来管理 EDU。例如，可以使用带有 **-edus** 选项的 db2pd 命令来列示所有活动的 EDU 线程。

每个客户机应用程序连接都有一个对数据库执行操作的协调代理程序。*协调代理程序*代表应用程序工作，并根据需要使用专用内存、进程间通信（IPC）或远程通信协议与其他代理程序进行通信。

DB2 体系结构提供了一个防火墙，以使应用程序与 DB2 数据库服务器在不同的地址空间中运行（图 4）。防火墙将数据库和数据库管理器与应用程序、存储过程和用户定义的函数（UDF）隔开。此防火墙有助于维护数据库中数据的完整性，这是因为，它将阻止应用程序编程错误覆盖内部缓冲区或数据库管理器文件。此防火墙还提高了可靠性，原因是应用程序错误不会导致数据库管理器崩溃。

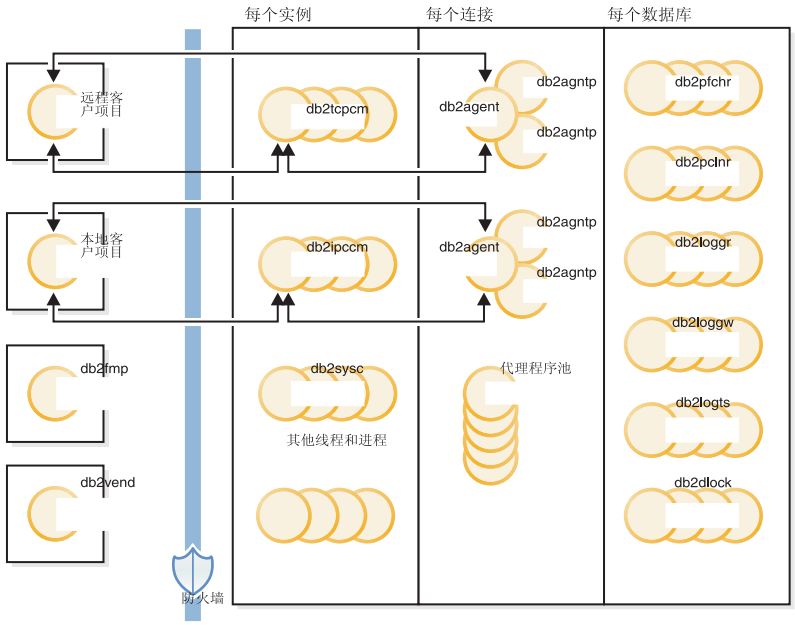


图 4. DB2 数据库系统的进程技术模型

客户机程序

客户机程序可以是远程程序，也可以是在数据库服务器所在机器上运行的本地程序。客户机程序首先通过通信侦听器与数据库联系。

侦听器

通信侦听器在 DB2 数据库服务器启动时启动。每种已配置的通信协议都有一个侦听器，本地客户机程序使用进程间通信（IPC）侦听器（db2ipccm）。侦听器包括:

- db2ipccm，用于本地客户机连接
- db2tcpcm，用于 TCP/IP 连接
- db2tcpdm，用于 TCP/IP 发现工具请求

代理程序

将为所有来自本地或远程客户机程序（应用程序）的连接请求分配相应的协调代理程序（db2agent）。创建协调代理程序之后，它将代表该应用程序执行所有数据库请求。

在分区数据库环境或者已启用*查询内并行性*的系统中，协调代理程序会将数据库请求分发给子代理程序（分别为 db2agntp 和 db2agnts）。与应用程序相关联但当前处于空闲状态的子代理程序名为 db2agnta。

协调代理程序可能:

- 已通过别名连接到数据库；例如，db2agent (DATA1) 将连接到数据库别名 DATA1。
- 已连接到实例；例如 db2agent (user1) 将连接到实例 user1。

DB2 数据库服务器还会将其他类型的代理程序（例如独立的协调代理程序或子协调代理程序）实例化，以便执行特定的操作。例如，独立的协调代理程序 db2agnti 用于运行事件监视器，而子协调代理程序 db2agnsc 用于在异常关闭后以并行方式执行数据库重新启动操作。

空闲代理程序驻留在代理程序池中。这些代理程序可用于处理来自代表客户机程序运行的协调代理程序或来自代表现有协调代理程序运行的子代理程序的请求。当存在大量应用程序工作负载时，配备大小适当的空闲代理程序池有助于提高性能。在这种情况下，可以根据需要立即使用空闲代理程序，而不需要为每个应用程序连接分配全新的代理程序，后一种情况涉及创建线程以及分配并初始化内存和其他资源。DB2 数据库服务器自动管理空闲代理池的大小。

db2fmp

受保护方式进程负责在防火墙外执行受防护的存储过程和用户定义的函数。db2fmp 进程始终是独立的进程，但可能是多线程进程，这取决于它执行的例程的类型。

db2vend

这是代表 EDU 执行供应商代码的进程；例如，执行用户出口程序以进行日志归档（仅适用于 UNIX）。

数据库 EDU

以下列表包括每个数据库使用的一些重要 EDU:

- db2dlock，用于死锁检测。在分区数据库环境中，使用另一个线程（db2glock）来协调 db2dlock EDU 从每个分区中收集的信息；db2glock 仅对目录分区运行。
- db2hadrp，高可用性灾难恢复（HADR）主服务器线程
- db2hadrs，HADR 备用服务器线程
- db2lfr，用于处理各个日志文件的日志文件阅读器
- db2loggr，用于处理日志文件以处理事务处理和恢复
- db2loggw，用于将日志记录写入日志文件
- db2logmgr，用于日志管理器。管理可恢复数据库的日志文件。
- db2logts，用于跟踪哪些表空间在哪些日志文件中有日志记录。此信息记录在数据库目录中的 DB2TSCHG.HIS 文件中。
- db2lused，用于更新对象用途

- db2pfchr，用于缓冲池预取程序
- db2pclnr，用于缓冲池页清除程序
- db2redom，用于重做主进程。在恢复期间，它处理重做日志记录并将日志记录指定给重做工作程序来进行处理。
- db2redow，用于重做工作程序。在恢复期间，它按照重做主进程的请求来处理重做日志记录。
- db2shred，用于处理日志页中的各个日志记录
- db2stmm，用于自调整内存管理功能
- db2taskd，用于分发后台数据库任务。这些任务由名为 db2taskp 的线程执行。
- db2wlmd，用于自动收集工作负载管理统计信息
- 事件监视器线程的标识方式如下:
  - db2evm*%1%2* (*%3*)

    其中，*%1* 可以是:

    - g – 全局文件事件监视器
    - gp – 全局管道事件监视器
    - l – 本地文件事件监视器
    - lp – 本地管道事件监视器
    - t – 表事件监视器

    %2 可以是:

    - i – 协调程序
    - p – 不是协调程序

    而 *%3* 是事件监视器名称
- 备份和复原线程的标识方式如下:
  - db2bm.*%1.%2*（备份和复原缓冲区操纵程序）和 db2med.*%1.%2*（备份和复原介质控制器），其中:

    - *%1* 是用于控制备份或复原会话的代理程序的 EDU 标识
    - *%2* 是用于区分属于特定备份或复原会话的线程（可能有许多个）的顺序值

    例如: **db2bm.13579.2** 标识具有 EDU 标识为 13579 的 db2agent 线程控制的第二个 db2bm 线程。

数据库服务器线程和进程

系统控制器（在 UNIX 上为 db2sysc，在 Windows 操作系统上为 db2syscs.exe）必须存在，这样数据库服务器才能工作。下列线程和进程将执行各种任务:

- db2acd，用于主管运行状况监视器、自动维护实用程序和管理任务调度程序的自主计算守护程序。此进程以前称为 db2hmon。
- db2aiothr，用于管理数据库分区的异步 I/O 请求（仅适用于 UNIX）
- db2alarm，用于在 EDU 请求的计时器到期时通知 EDU（仅适用于 UNIX）
- db2cart，用于在 **userexit** 数据库配置参数处于启用状态时归档日志文件
- db2disp，客户机连接集中器分派器
- db2fcms，快速通信管理器发送方守护程序

- db2fcmr，快速通信管理器接收方守护程序
- db2fmd，故障监视器守护程序
- db2fmtlg，用于在 **logretain** 数据库配置参数处于启用状态并且 **userexit** 数据库配置参数处于禁用状态时格式化日志文件
- db2licc，管理已安装的 DB2 许可证
- db2panic，紧急代理程序，用于在特定数据库分区达到代理程序的限制时处理紧急请求（仅用于分区数据库环境）
- db2pdbc，并行系统控制器，用来处理来自远程数据库分区的并行请求（仅用于分区数据库环境）
- db2resync，扫描全局再同步列表的再同步代理进程
- db2srvlst，用于管理 DB2 z/OS® 版之类的系统的地址列表
- db2sysc，主系统控制器 EDU；它处理关键的 DB2 服务器事件
- db2thcln，在 EDU 终止时重新启动资源（仅适用于 UNIX）
- db2wdog，在 UNIX 和 Linux 操作系统上处理异常终止的看守程序

数据库代理程序

应用程序访问数据库时，将有多个进程或线程开始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。数据库代理程序是数据库管理器中用来处理应用程序请求的线程。在版本 9.5 中，代理程序在所有平台上都作为线程运行。

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都由单个工作程序代理程序协调。*工作程序代理程序*执行应用程序请求，但不与任何特定应用程序永久相连。*协调代理程序*与应用程序之间的关联时间最长，这是因为它们一直与应用程序相连，直到应用程序断开连接为止。引擎集中器处于启用状态时的情况是此规则的唯一例外，在这种情况下，协调代理程序可以在事务边界（COMMIT 或 ROLLBACK）终止该关联。

工作程序代理程序分为三类:

- 空闲代理程序

  这是最简单的工作程序代理程序形式。它没有出站连接，并且没有本地数据库连接或实例连接。

- 活动协调代理程序

  客户机应用程序的每个数据库连接都有一个在数据库上协调其工作的活动代理程序。在创建协调代理程序之后，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他代理程序进行通信。每个代理程序都使用自己的专用内存来运行，并与其他代理程序共享数据库管理器和数据库全局资源，例如缓冲池。事务完成后，活动协调代理程序可以变为不活动代理程序。当客户机与数据库断开连接或者与实例拆离时，其协调代理程序将:

  – 成为活动的协调代理程序（如果其他连接正在等待）
  – 被释放并标记为空闲（如果没有任何连接正在等待，并且正在自动管理池代理程序的最大数目或尚未达到该最大数目）
  – 被终止并且其存储器被释放（如果没有任何连接正在等待，并且已达到缓冲池代理程序的最大数目）

- 子代理程序

  协调代理程序将数据库请求分发至子代理程序，然后由这些子代理程序为应用程序执行请求。在协调代理程序创建之后，它将通过协调对数据库执行请求的子代理程序，来代表它的应用程序处理所有数据库请求。在 DB2 版本 9.5 中，子代理程序还可以存在于非分区环境以及未启用查询内并行性的环境中。

不为任何应用程序执行工作且正在等待分配的代理程序被认为是空闲代理程序，它驻留在*代理程序池*中。这些代理程序可以处理来自代表客户机程序运行的协调代理程序的请求，也可以处理来自代表现有协调代理程序运行的子代理程序的请求。可用代理程序的数目取决于 **num_poolagents** 数据库管理器配置参数的值。

需要代理程序时，如果没有空闲的代理程序，那么将动态地创建新的代理程序。因为创建新代理程序将引起特定数量的开销，因此，如果可以为客户机激活空闲的代理程序，那么可提高 CONNECT 和 ATTACH 性能。

为应用程序执行工作的子代理程序与该应用程序相关联。它完成指定的工作后，可以被放入代理程序池，但它仍然与原始应用程序相关联。当该应用程序请求执行其他工作时，数据库管理器在创建新代理程序之前，将首先在空闲池中查找相关联的代理程序。

数据库代理程序管理

大多数应用程序在连接的应用程序数与数据库服务器可以处理的应用程序请求数之间建立一对一关系。但是，您的环境可能要求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独立地控制这些因素:

- **max_connections** 参数指定所连接的应用程序的最大数目
- **max_coordagents** 参数指定可以同时处理的应用程序请求的最大数目

当 **max_connections** 的值大于 **max_coordagents** 的值时，将启用连接集中器。由于每个活动的协调代理程序都需要全局数据库资源开销，因此这些代理程序的数目越大，达到可用全局资源量上限的机会也就越大。为了防止发生这种情况，请将 **max_connections** 的值设置为大于 **max_coordagents** 的值，或者将这两个参数都设置为 AUTOMATIC。

在两种特定情况下，最好将这两个参数设置为 AUTOMATIC:

- 如果您确信系统能够处理可能需要的所有连接，但是又想通过限制协调代理程序数来限制所使用的全局资源量，那么请将 **max_connections** 设置为 AUTOMATIC。当 **max_connections** 大于 **max_coordagents** 时，将启用连接集中器。
- 如果您不想限制最大连接数或最大协调代理程序数，但是您知道系统要求所连接应用程序数与所处理应用程序请求数之间存在多对一关系或者能够受益于这种关系，那么请将这两个参数都设置为 AUTOMATIC。这两个参数都设置为 AUTOMATIC 时，数据库管理器将使用您指定的值作为连接数与协调代理程序数之间的理想比率。注意，这两个参数都支持同时配置起始值和 AUTOMATIC 设置。例如，以下命令使值 200 和 AUTOMATIC 都与 **max_coordagents** 参数关联: `update dbm config using max_coordagents 200 automatic`。

示例

请考虑以下方案:

- **max_connections** 参数设置为 AUTOMATIC 并且当前值为 300
- **max_coordagents** 参数设置为 AUTOMATIC 并且当前值为 100

max_connections 与 **max_coordagents** 的比率是 300:100。当连接增加时，数据库管理器将创建新的协调代理程序，仅当有需要时才进行连接集中。这些设置将引起下列操作:

- 第 1 到 100 个连接将创建新的协调代理程序
- 第 101 到 300 个连接不会创建新的协调代理程序；它们共享已创建的那 100 个代理程序
- 第 301 到 400 个连接将创建新的协调代理程序
- 第 401 到 600 个连接不会创建新的协调代理程序；它们共享已创建的那 200 个代理程序
- 依此类推...

此示例假定已连接的应用程序正在执行足够多的工作来保证在每一步骤中创建新的协调代理程序。经过一段时间之后，如果已连接的应用程序不再执行足够数量的工作，那么协调代理程序将进入不活动状态并可能终止。

如果连接数减少，但是由剩余连接执行的工作量还很大，那么协调代理程序数可能不会立即减少。**max_connections** 和 **max_coordagents** 参数不会直接影响代理程序共享或代理程序终止。正常的代理程序终止规则仍然适用，这意味着连接数与协调代理程序数之间的比率可能并非正好是您指定的值。代理程序在终止之前可能会返回到代理程序池以供复用。

如果需要进行详细程度更高的控制，那么请指定较简单的比率。例如，可以将上一个示例中的 300:100 比率表达为 3:1。如果 **max_connections** 设置为 3（AUTOMATIC），并且 **max_coordagents** 设置为 1（AUTOMATIC），那么对于每 3 个连接可以创建 1 个协调代理程序。

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个数据库。远程应用程序是从作为数据库服务器所在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接。

客户机连接的管理方式取决于连接集中器是否处于打开状态。每当 **max_connections** 数据库管理器配置参数的值大于 **max_coordagents** 配置参数的值时，连接集中器处于打开状态。

- 如果连接集中器处于关闭状态，那么每个客户机应用程序都将被指定一个称为*协调代理程序*的唯一引擎可分派单元（EDU），后者负责协调该应用程序的处理并与之进行通信。
- 如果连接集中器处于打开状态，那么每个协调代理程序都可以管理多个客户机连接（每次管理一个），并可以协调其他工作程序代理程序以完成这项工作。对于具有许多相对短暂的连接的因特网应用程序或具有许多相对小型的事务的应用程序而言，连接集中器通过允许更多客户机应用程序同时进行连接来提高系统性能。它还可以减少每个连接所使用的系统资源量。

在图 5 中，DB2 服务器中的每个圆圈都代表一个使用操作系统线程实现的 EDU。

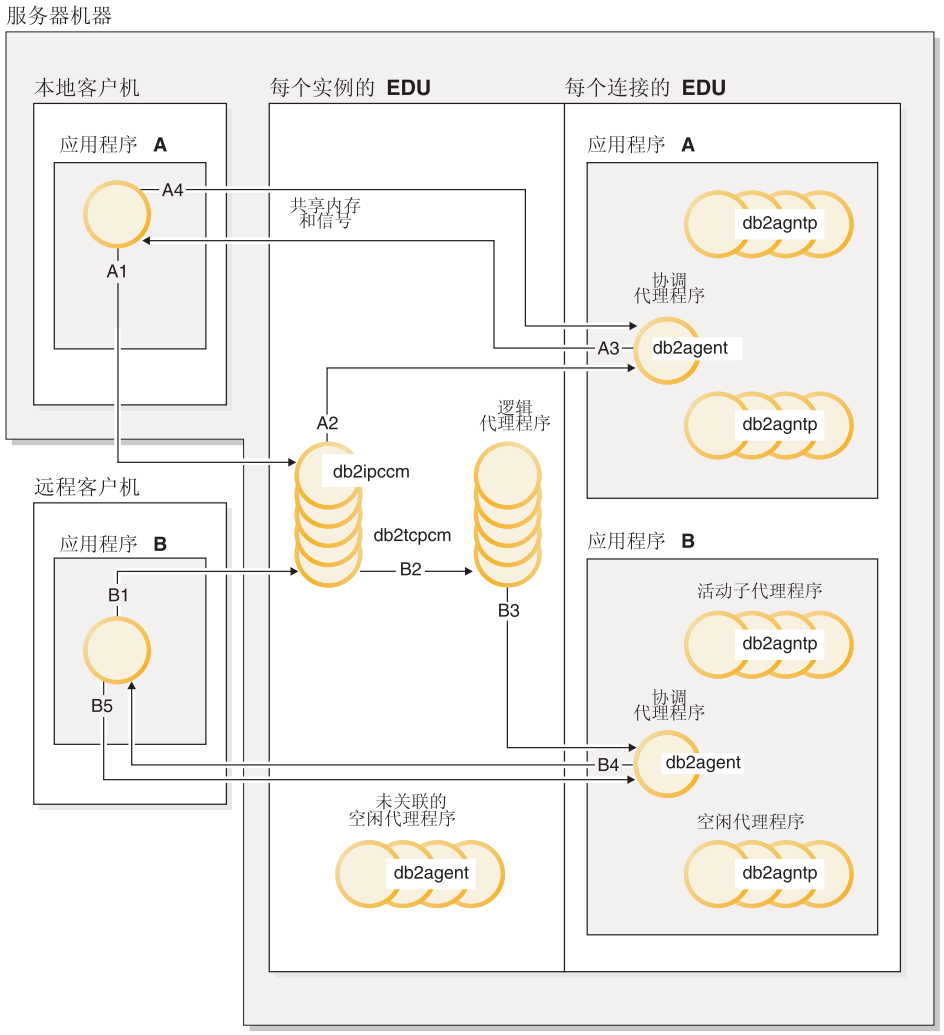


图 5. 客户机/服务器处理模型概述

- 在 A1 处，本地客户机通过 db2ipccm 建立通信。
- 在 A2 处，db2ipccm 与 db2agent EDU 配合工作，后者成为来自本地客户机的应用程序请求的协调代理程序。
- 在 A3 处，该协调代理程序联系该客户机应用程序，以便在该客户机应用程序与协调程序之间建立共享内存通信。
- 在 A4 处，本地客户机上的应用程序连接到数据库。
- 在 B1 处，远程客户机通过 db2tcpcm 建立通信。如果选择另一种通信协议，那么将使用相应的通信管理器。
- 在 B2 处，db2tcpcm 与 db2agent（后者成为应用程序的协调代理程序）配合工作并将连接传递至此代理程序。
- 在 B4 处，协调代理程序与远程客户机应用程序联系。
- 在 B5 处，远程客户机应用程序连接到数据库。

另请注意:

- 工作程序代理程序执行应用程序请求。工作程序代理程序共有四种: 活动协调代理程序、活动子代理程序、相关联子代理程序和空闲代理程序。
- 每个客户机连接均与活动协调代理程序相链接。
- 在分区数据库环境或者启用了分区内并行性的环境中，协调代理程序将数据库请求分发到子代理程序（db2agntp）。
- 有一个代理程序池（db2agent），空闲代理程序在此位置等待新工作。
- 其他 EDU 管理客户机连接、日志、两阶段落实操作、备份和复原操作以及其他任务。

图 6 显示了作为服务器机器环境组成部分的其他 EDU。每个活动数据库都有自己的预取程序（db2pfchr）和页清除程序（db2pclnr）的共享池，并有自己的记录器（db2loggr）和死锁检测器（db2dlock）。

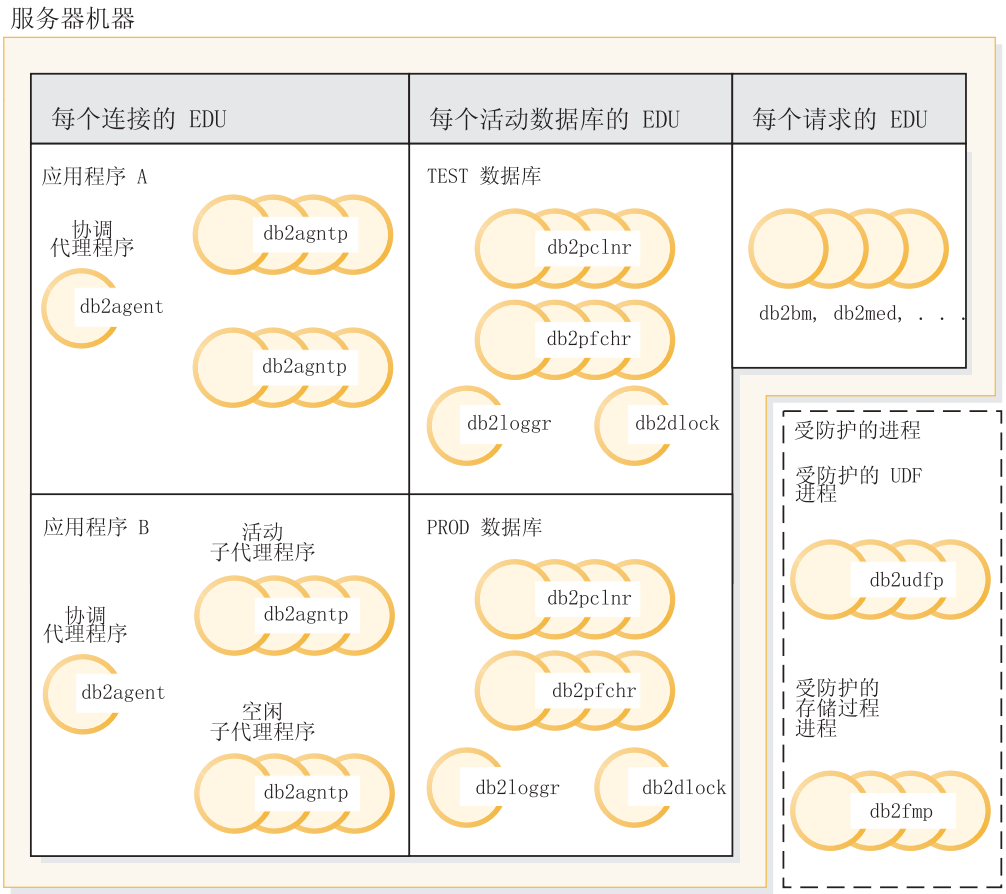


图 6. 数据库服务器中的 EDU

将对受保护的用户定义的函数（UDF）和存储过程（图中未显示）进行管理，以便尽可能减少与它们的创建和删除相关联的成本。**keepfenced** 数据库管理器配置参数的缺省值为 YES，这使得下次进行过程调用时可以重复使用存储过程进程。

注: 为了提高性能，不受保护的 UDF 和存储过程直接在代理程序的地址空间中运行。但是，由于它们对代理程序的地址空间的访问权不受限制，因此，在使用它们之前，需要对它们进行严格的测试。

图 7 显示了单一数据库分区处理模型与多数据库分区处理模型之间存在的相似性与差别。

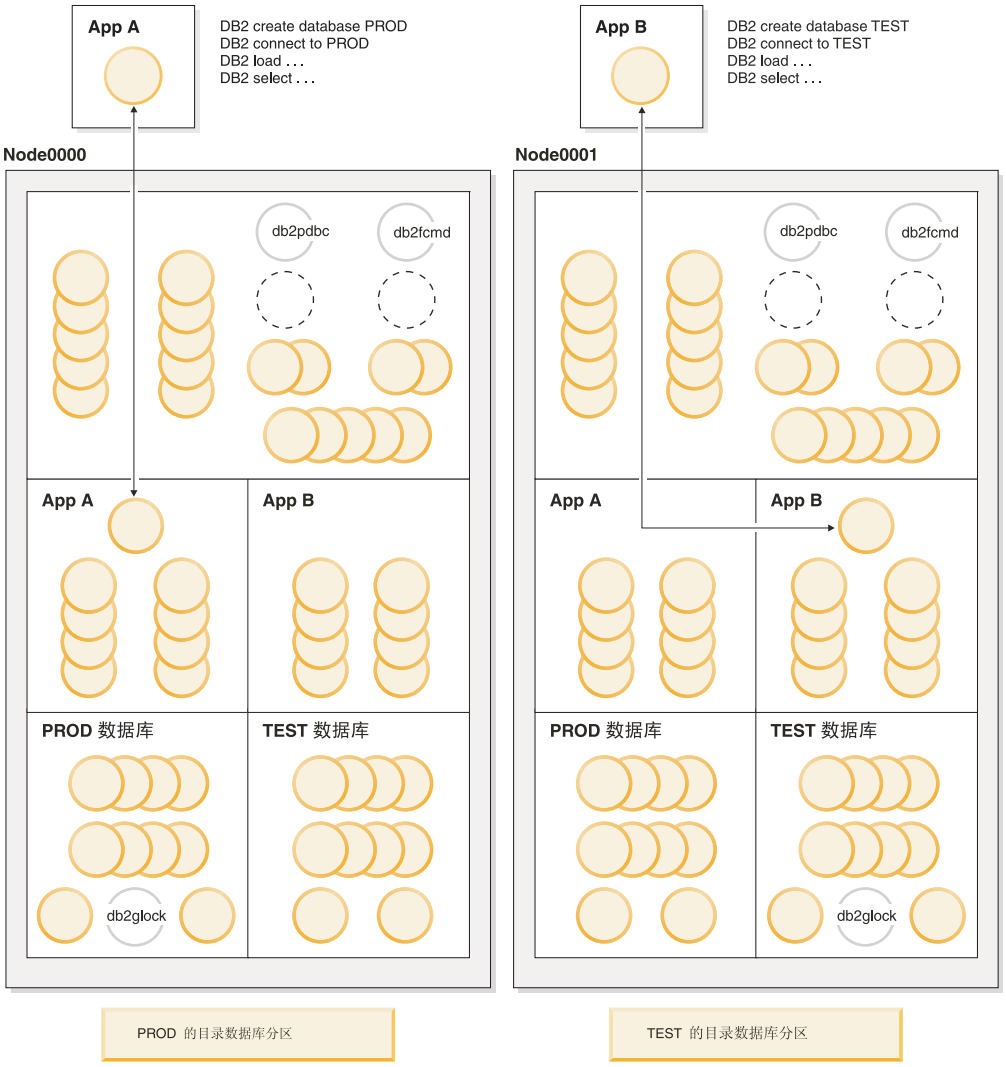


图 7. 多数据库分区的进程技术模型

在多数据库分区环境中，从中发出 CREATE DATABASE 命令的数据库分区被称为*目录数据库分区*。系统目录表存储在此数据库分区中。系统目录是所有关于数据库中对象的信息的存储库。

如图 7 所示，由于应用程序 A 在 Node0000 中创建 PROD 数据库，因此还在此数据库分区中创建 PROD 数据库的目录。同样，由于应用程序 B 在 Node0001 中创建 TEST 数据库，因此还在此数据库分区中创建 TEST 数据库的目录。最好在不同的数据库分区中创建数据库，以便在环境中各个数据库分区之间平衡与每个数据库的目录相关联的额外活动。

在多数据库分区环境的每个数据库分区中，可以找到与实例相关联的其他 EDU（db2pdbc 和 db2fcmd）。这些 EDU 用于在数据库分区之间协调请求以及启用快速通信管理器（FCM）。

还有一个与目录数据库分区相关联的附加 EDU，即 db2glock。此 EDU 控制活动数据库所在的各个数据库分区之间的全局死锁。

每个来自应用程序的连接请求都由协调代理程序的相关联连接表示。*协调代理程序*是指与应用程序进行通信（即接收请求和发送应答）的代理程序。它可以独立地处理请求，也可以协调多个子代理程序对请求进行处理。协调代理程序所在的数据库分区被称为该应用程序的*协调程序数据库分区*。

协调程序数据库分区将来自应用程序的部分数据库请求发送至其他数据库分区中的子代理程序。接着，在协调程序数据库分区中对所有结果进行合并，然后发送回给应用程序。

可以配置任意数目的数据库分区在同一机器上运行。这称为*多逻辑分区*配置。在主存储器容量非常大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，可以优化数据库分区之间的通信，以使用共享内存和信号。

连接集中器对客户机连接的改善

对于频繁建立连接但连接持续时间相对短暂的应用程序，连接集中器通过允许高效处理许多并发客户机连接来提高这些应用程序的性能。它还将减少每个连接期间的内存使用量以及减少上下文切换次数。

当 **max_connections** 数据库管理器配置参数的值大于 **max_coordagents** 配置参数的值时，连接集中器处于启用状态。

在需要许多并发用户连接的环境中，可以启用连接集中器以便更高效地使用系统资源。此功能部件结合了以前只有 DB2 Connect 连接池才具有的优点。首次连接后，连接集中器能够缩短连接到主机所需的时间。当请求与主机断开连接时，将断开入站连接，但在存储池中保留与主机的出站连接。接收到新的连接请求时，数据库管理器将尝试复用存储池中的现有出站连接。

为了最大程度地提高使用连接池或连接集中器的应用程序的性能，请调整用于控制高速缓存数据块大小的参数。有关更多信息，请参阅 DB2 Connect 产品文档。

示例

- 假定有一个单一分区数据库，并且，平均有 1000 个用户同时连接到此数据库。有时，连接的用户数可能会更多。并发事务的数目可能高达 200，但不会高于 250。这些事务都比较短暂。

  对于此工作负载，可以设置下列数据库管理器配置参数:

  – 将 **max_coordagents** 设置为 250，以便支持最大的并发事务数
  – 将 **max_connections** 设置为 AUTOMATIC 并且值为 1000，以确保支持任意数目的连接；在此示例中，任何大于 250 的值都将确保启用连接集中器。
  – 保持 **num_poolagents** 设置为缺省值，但是应确保有数据库代理程序可用来为入局客户机请求提供服务，并确保由于创建新代理程序而引起的开销较低。
- 假定有一个单一分区数据库，并且，平均有 1000 个用户同时连接到此数据库。有时，连接的用户数可能会达到 2000。在任何给定时间，预期平均有 500 个用户执行工作。并发事务数大约为 250。通常，500 个协调代理程序将会过多；对于所连接的 1000 个用户，250 个协调代理程序应该足够了。

  对于此工作负载，可以按如下方式更新数据库管理器配置:

```
update dbm cfg using max_connections 1000 automatic
update dbm cfg using max_coordagents 250 automatic
```

这意味着，随着连接数在超过 1000 以后继续增加，将根据需要来创建更多的协调代理程序，其最大值由连接总数确定。随着工作负载增加，数据库管理器将尝试在连接数与协调代理程序数之间维持相对稳定的比率关系。

- 假定您不想启用连接集中器，但却希望限制所连接的用户数。例如，要将同时连接的用户数限制为 250，可以设置下列数据库管理器配置参数:
  - 将 **max_coordagents** 设置为 250。
  - 将 **max_connections** 设置为 250。
- 假定您不想启用连接集中器，并且不想限制所连接的用户数。那么，可以按如下方式更新数据库管理器配置:

```
update dbm cfg using max_connections automatic
update dbm cfg using max_coordagents automatic
```

分区数据库中的代理程序

在分区数据库环境或者启用了分区内并行性的环境中，每个数据库分区都有自己的代理程序池，可以从中抽取子代理程序。

由于存在这个池，因此不必在每次需要子代理程序时将其创建或者在它完成工作时将其破坏。这些子代理程序仍然可以作为此池中的相关联代理程序，并可以由数据库管理器使用，以执行来自与它们相关联的应用程序或者新应用程序的新请求。

对系统性能和内存耗用量的影响与代理程序池的设置有很大关系。有关代理程序池大小的数据库管理器配置参数（**num_poolagents**）将影响可以与一个数据库分区中的应用程序保持关联的代理程序和子代理程序的总数。如果池太小并且该池已满，那么子代理程序将解除自己与它所处理应用程序的关联并终止。由于必须经常创建子代理程序并重新使它们与应用程序相关联，因此性能将下降。

缺省情况下，**num_poolagents** 设置为 AUTOMATIC 并且值为 100，数据库管理器将自动管理池中的空闲代理程序数。

如果以手动方式将 **num_poolagents** 的值设置得过小，那么一个应用程序的相关联子代理程序就可能填满整个池。因此，当另一个应用程序需要新的子代理程序，并且在其代理程序池中没有子代理程序时，它将从其他应用程序的代理程序池中重新启动不活动的子代理程序。这种行为将确保充分利用资源。

如果以手动方式将 **num_poolagents** 的值设置得过大，那么相关联的子代理程序可能会长时间停留在池中未被使用并耗用数据库管理器资源，导致那些资源不可用于其他任务。

当连接集中器处于启用状态时，**num_poolagents** 的值不一定反映任何时候在池中可能处于空闲状态的代理程序的准确数目。可能会临时需要代理程序以处理工作负载更高的活动。

除数据库代理程序以外，其他异步数据库管理器活动也作为独立的进程或线程运行，其中包括:

- 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
- 通信和 IPC 侦听器
- 表空间容器重平衡程序

进行配置以实现良好的性能

某些类型的 DB2 部署的配置高度可调，InfoSphere™ Balanced Warehouse™（BW）或者 SAP 系统中的那些部署就是如此。

对于 BW 而言，硬件因素（例如 CPU 数目、内存量相对于 CPU 数目的比率、磁盘的数目和配置以及版本）将根据确定最佳配置时进行的全面测试预先指定。对于 SAP 而言，未准确地指定硬件配置；但是，提供了许多样本配置。此外，SAP 最佳实践提供了建议的 DB2 配置设置。如果您正在将 DB2 部署用于提供了经过精心测试的配置准则的系统，那么通常应该利用那些准则以代替更通用的经验法则。

假定您有一个未掌握其详细硬件配置的建议系统。您的目标是，确定几项关键的配置决策，以使系统实现良好的性能。此步骤通常在系统启动并运行前执行，因此，您对系统的实际运行情况的了解可能非常有限。在某种程度上，您必须根据您对该系统将要执行的任务的了解进行“最佳猜测”。

硬件配置

在配置系统以提高性能时，CPU 容量是其中一项主要的独立变量。由于所有其他硬件配置通常都由 CPU 容量决定，因此，预测给定工作负载所需的 CPU 容量并不容易。在商业智能（BI）环境中，合理的估算是，每个处理器核心处理 200-300 GB 的活动原始数据。对于其他环境而言，比较好的方法是根据一个或多个现有 DB2 系统来测量所需的 CPU 容量。例如，如果新系统需要处理的用户数增加 50%，并且运行的每个 SQL 语句都至少与现有系统中的 SQL 语句一样复杂，那么假定所需的 CPU 容量也增加 50% 比较合理。同样，还应该考虑其他在 CPU 使用率方面预计有所变化的因素，例如不同的吞吐量需求或者在使用触发器或引用完整性方面的变动。

在根据可用的信息很好地认识 CPU 需求之后，就可以逐步确定硬件配置的其他方面。虽然您必须考虑数以千兆字节或兆兆字节计的所需系统磁盘容量，但性能方面的最重要因素是每秒 I/O 次数（IOPS）这一容量，即每秒的数据传输兆字节数。实际上，此因素由涉及到的各个磁盘的数目确定。

情况为何如此？在过去这十年，CPU 在速度方面取得了难以置信的巨大进步，但磁盘在其容量和降低成本方面取得的进步更大。磁盘寻道时间和传输率方面也有所改善，但却落后于 CPU 速度方面的进步。因此，为了实现现代系统所需的总体性能，使用多个磁盘比以往更加重要，对于将要执行大量随机磁盘 I/O 操作的系统而言更是如此。通常，我们总是希望使用尽量少的磁盘来存储系统中的全部数据，但这通常会导致性能非常差。

对于 RAID 存储器，或者对于可独立寻址的驱动器，经验法则是为每个处理器核心至少配置 10 到 20 个磁盘。对于存储服务器，建议采用类似的数目；但是，在这种情况下，需要更加小心谨慎。存储服务器上的空间分配通常与容量而非吞吐量更为相关。您最好充分了解数据库存储器的物理布局，以确保逻辑上相分离的存储器不会意外重叠。例如，对于 4 路系统而言，合理的分配可能是 8 个各包含 8 个驱动器的阵列。但是，

如果全部这 8 个阵列共享相同的 8 个底层物理驱动器，那么与 8 个阵列分布于 64 个物理驱动器的情况相比，此配置的吞吐量将大幅下降。

最好为 DB2 事务日志预留一些专用（非共享）磁盘。这是因为，日志的 I/O 特征与 DB2 容器有很大的不同，日志 I/O 与其他类型 I/O 之间的竞争关系会引起日志记录瓶颈，在需要执行大量写活动的系统中尤其如此。

通常，RAID-1 磁盘对每秒能够为多达 400 个具有合理写密度的 DB2 事务提供足够的日志记录吞吐量。吞吐率越高或者日志记录量越大（例如批量插入期间的情况），就需要越大的日志吞吐量，此吞吐量可以由 RAID-10 配置中通过写高速缓存磁盘控制器连接到系统的附加磁盘提供。

由于 CPU 与磁盘实际上按不同的时标工作（分别为纳秒和毫秒），因此，需要使它们相分离才能实现合理的处理性能。这就是内存所起的作用。在数据库系统中，内存的主要用途是避免执行 I/O，因此在某种程度上，系统的内存越多，性能就越好。幸运的是，内存成本在过去几年间大幅下降，现在，带有数十到数百吉字节（GB）的 RAM 的系统已很常见。通常，每个处理器核心 4 到 8 吉字节对于大多数应用程序而言已足够。

AIX 配置

为了实现良好性能而需更改的 AIX 参数相对较少。下列建议假定您使用的 AIX 的级别为 5.3 或更高级别。同样，如果存在已适用于您的系统的特定设置（例如 BW 或 SAP 配置），那么这些设置应该优先于下列通用准则。

- VMO 参数 **LRU_FILE_REPAGE** 应该设置为 0。此参数控制 AIX 是否牺牲计算页或文件系统高速缓存页。此外，**minperm** 应该设置为 3。这两个值都是 AIX 6.1 中的缺省值。
- 最初，可以保留 AIO 参数 **maxservers** 的缺省值，即每个 CPU 运行 10 个服务器。系统进入活动状态后，将按如下方式对 **maxservers** 进行调整:
  1. 收集 `ps -elfk | grep aio` 命令的输出并确定是否所有异步 I/O（AIO）内核进程（aioserver）都耗用相同的 CPU 时间。
  2. 如果是这种情况，那么表明 **maxservers** 的设置可能太小。请将 **maxservers** 增大 10%，然后重复步骤 1。
  3. 如果某些 aioserver 耗用的 CPU 时间少于其他 aioserver，那么表明系统至少已有所需的 aioserver。如果使用较少 CPU 时间的 aioserver 所占的百分比超过 10%，那么请将 **maxservers** 减小 10% 并重复步骤 1。
- AIO 参数 **maxreqs** 应该设置为 `MAX( NUM_IOCLEANERS x 256, 4096 )`。此参数控制未完成的 AIO 请求的最大数目。
- hdisk 参数 **queue_depth** 应该基于阵列中的物理磁盘数。例如，对于 IBM® 磁盘，**queue_depth** 的缺省值是 3，建议的值是 3 x *number-of-devices*。此参数控制可排队磁盘请求的数目。
- 磁盘适配器参数 **num_cmd_elems** 应该设置为所有与适配器相连接的设备的 **queue_depth** 之和。此参数控制可以进行排队以便发送到适配器的请求数。

Solaris 和 HP-UX 配置

对于在 Solaris 或 HP-UX 上运行的 DB2 而言，可以使用 db2osconf 实用程序来检查内核参数以及根据系统大小提供建议。db2osconf 实用程序允许您根据内存和 CPU 来指定内核参数，此外，也可以借助于将当前系统配置与预期的将来配置作比较的常规缩

放因子来进行指定。一种好的做法是使用缩放因子 2，或者，如果正在运行大型系统（例如 SAP 应用程序），那么请使用更大的值。通常，db2osconf 为您配置 Solaris 和 HP-UX 提供了良好的起始点，但它未提供最佳的值，这是因为它无法考虑当前以及将来的工作负载。

Linux 配置

将 Linux 系统用作 DB2 服务器时，可能必须更改某些 Linux 内核参数。由于 Linux 分发有所变化，并且由于此环境高度灵活，因此我们将只考虑一些需要根据 Linux 实现进行验证的最重要设置。

在 64 位系统上，必须将 **SHMMAX**（共享内存段的最大大小）设置为最小值 1 GB（1073741824 字节），并且应该将 **SHMALL** 参数设置为数据库服务器上可用内存量的 90%。缺省情况下，**SHMALL** 是 8 GB。对于 DB2 而言，其他重要的 Linux 内核配置参数及其建议值如下所示:

- **kernel.sem**（指定 4 个内核信号设置 – SEMMSL、SEMMNS、SEMOPM 和 SEMMNI）: 250 256000 32 1024
- **kernel.msgmni**（消息队列标识的数目）: 1024
- **kernel.msgmax**（消息的最大大小，以字节计）: 65536
- **kernel.msgmnb**（消息队列的缺省大小，以字节计）: 65536

DB2 数据库分区功能

有关是否使用 DB2 数据库分区功能（DPF）的决策通常并非仅仅基于数据量，而更多地基于工作负载。作为一般性的准则，大多数 DPF 部署隶属于数据仓储和商业智能领域。对于大型的复杂查询环境，强烈建议您使用 DPF，这是因为它的无共享体系结构具有出色的可伸缩性。对于不大可能迅速增大的较小型数据集市（最大可达 300 GB）而言，通常最好选择 DB2 企业服务器版（ESE）配置。但是，DPF 能够使大型或快速增长的 BI 环境受益匪浅。

通常，在典型的分区数据库系统中，每个数据分区有一个处理器核心。例如，包含 *n* 个处理器核心的系统有可能将目录存放在分区 0 中，并有 *n* 个附加的数据分区。如果目录分区的使用量较大（例如，用于存放单分区维表），那么也可以为其分配一个处理器核心。如果该系统将支持非常多的并发活动用户，那么每个分区可能需要两个核心。

作为一般性的指南，您应该在每个分区大约存储 250 GB 活动原始数据的情况下进行规划。

InfoSphere Balanced Warehouse 文档提供了有关分区数据库配置最佳实践的深入信息。此文档还包含有关非 Balanced Warehouse 部署的实用信息。

代码页和整理顺序选项

代码页或代码集以及整理顺序选项除了影响数据库行为以外，还将对性能产生重大影响。Unicode 的使用非常广泛，这是因为，与传统的单字节代码页相比，它使您能够在数据库中表示更多的字符串。对于 DB2 版本 9.5 中的新数据库而言，Unicode 是缺省代码集。但是，由于 Unicode 代码集使用多个字节来表示某些单个的字符，因此会增加磁盘和内存需求。例如，UTF-8 代码集（这是其中一种最常用的 Unicode 代码集）使用 1 到 4 个字节来表示每个字符。从单字节代码集迁移到 UTF-8 所引起的平均字符串扩

展系数非常难以估算，这是因为，此系数取决于多字节字符的使用频率。对于典型的北美内容而言，通常不会进行扩展。对于大多数西欧语言而言，使用重音字符通常会导致扩展接近 10%。

除此之外，相对于单字节代码页，使用 Unicode 会导致耗用额外的 CPU 时间。首先，如果发生扩展，那么较长的字符串将要求执行更多的工作。其次，更重要的一点是，与单字节代码页所使用的典型 SYSTEM 整理顺序相比，更为复杂的 Unicode 整理顺序所使用的算法（例如 UCA500R1_NO）的成本更高。成本提高的原因是，以文化方面正确的方式对 Unicode 字符串进行排序比较复杂。受影响的操作包括排序、字符串比较、LIKE 处理和索引创建。

如果需要使用 Unicode 才能正确地表示数据，那么请谨慎地选择整理顺序。

- 如果数据库将包含多种语言的数据，并且该数据的正确排序顺序极为重要，那么请使用其中一种在文化方面正确的整理顺序（例如 UCA500R1_*）。根据数据和应用程序的不同，相对于 IDENTITY 顺序，这可能会导致性能开销提高 1.5 到 3 倍。
- 在文化方面正确的整理顺序既有规范化版本也有非规范化版本。规范化整理顺序（例如 UCA500R1_NO）将执行附加的检查以处理格式不正确的字符，而非规范化整理顺序（例如 UCA500r1_NX）不执行这些检查。除非处理格式不正确的字符将引起问题，否则请使用非规范化版本，这是因为避免执行规范化代码有助于提高性能。也就是说，即使在文化方面正确的非规范化整理顺序的成本也非常高。
- 如果正在将数据库从单字节环境移至 Unicode 环境，但对是否包含各种语言的内容没有严格要求（大多数部署都是这种情况），那么可感知语言的整理顺序可能比较适合。*可感知语言的整理顺序*（例如 SYSTEM_819_BE）利用了许多 Unicode 数据库只包含一种语言的数据这一事实。它们与单字节整理顺序（例如 SYSTEM_819）使用同一种基于查找表的整理算法，因此非常高效。通常，如果原始单字节数据库中的整理行为可接受，那么只要语言内容在移至 Unicode 后未显著更改，那么就应该考虑使用在文化方面有感知能力的整理顺序。相对于在文化方面正确的整理顺序，这将产生非常大的性能增益。

物理数据库设计

- 通常，与系统管理存储器（SMS）常规表空间相比，基于文件的数据库管理存储器（DMS）常规表空间的性能更好。SMS 通常用于临时表空间，当临时表非常小时尤其如此；但是，在这种情况下，SMS 的性能优势将随着时间的推移而下降。
- 过去，DMS 原始设备表空间与 DMS 文件表空间相比具有相当显著的性能优势；但是，随着直接 I/O 的引入（现在，CREATE TABLESPACE 和 ALTER TABLESPACE 语句中的 NO FILE SYSTEM CACHING 子句已使其成为缺省选项），DMS 文件表空间的性能实际上与 DMS 原始设备表空间相同。

初始 DB2 配置设置

DB2 配置顾问程序（也称为 AUTOCONFIGURE 命令）采用您提供的基本系统准则并确定一组良好的起始 DB2 配置值。AUTOCONFIGURE 命令能够提供相对于缺省配置设置的实质改进，这是获取初始配置值的建议方法。通常，您需要根据系统特征对 AUTOCONFIGURE 命令生成的建议进行一些附加的微调工作。

下面是一些有关使用 AUTOCONFIGURE 命令的建议:

- 从 DB2 版本 9.1 开始，尽管 AUTOCONFIGURE 命令将在数据库创建时自动运行，但您最好还是显式地运行 AUTOCONFIGURE 命令。这是因为，通过显式地运行此命令，您可以指定“关键字/值”对，从而有助于为系统定制结果。
- 在数据库中填充适当数量的活动数据之后，请运行（或重新运行）AUTOCONFIGURE 命令。这将为此工具提供更多有关数据库性质的信息。用于填充数据库的数据量十分重要，其原因在于，这将影响缓冲池大小计算之类的工作。数据过多或过少都将导致这些计算的准确性下降。
- 对于 AUTOCONFIGURE 命令的重要关键字（例如 **mem_percent**、**tpm** 和 **num_stmts**），尝试使用不同的值，以便确定受这些更改影响的配置值以及影响程度。
- 如果您正在试验不同的关键字和值，那么请使用 APPLY NONE 选项。这使您有机会将建议与当前设置进行比较。
- 由于缺省值可能不适合于您的系统，因此请对所有关键字指定值。例如，**mem_percent** 的缺省值为 25%，这对于专用的 DB2 服务器而言太小；在这种情况下，建议的值是 85%。

DB2 自主与自动参数

最近的 DB2 数据库产品发行版显著增加了在实例或数据库启动时自动进行设置或者在操作期间动态地进行调整的参数的数目。对于大多数系统而言，自动设置所提供的性能将仅次于非常仔细地进行手动调整的系统。这尤其得益于 DB2 自调整内存管理器（STMM），此管理器动态地调整数据库内存分配总量以及 DB2 系统中的四个主要内存使用者：缓冲池、锁定列表、程序包高速缓存和排序堆。

由于这些参数将逐个分区地进行应用，因此，在分区数据库环境中使用 STMM 时应该小心谨慎。在分区数据库系统中，STMM 将持续测量单一分区的内存需求（此分区由 DB2 系统自动选择，但所作的选择可以被覆盖）并将堆大小更新“推送”到所有已启用 STMM 的分区。由于所有分区都使用相同的值，因此在分区之间的数据量、内存需求和一般活动水平非常一致的分区数据库环境中，STMM 的效用最佳。如果少数分区的数据量或内存需求有所偏斜，那么应该对那些分区禁用 STMM 并允许 STMM 对更为一致的分区进行调整。例如，对于目录分区，通常应该将 STMM 禁用。

对于数据分布不均匀的分区数据库环境而言（在这种环境中，建议您不要进行持续的跨集群内存调整），可以在“调整阶段”有选择地临时使用 STMM 来帮助确定良好的手动堆设置：

- 对一个“典型”的分区启用 STMM。对于其他分区，STMM 仍处于禁用状态。
- 在内存设置稳定之后，请将 STMM 禁用并以手动方式将受影响的参数“固定”为经过调整的值。
- 在其他具有类似数据量和内存需求的数据库分区（例如同一个分区组的分区）中，部署经过调整的值。
- 如果系统中有多组相互分离的数据库分区包含类似的数据量和数据类型并扮演类似的角色，那么重复此过程。

通常，配置顾问程序在适用的情况下将选择启用自主设置。这包括 RUNSTATS 命令执行的自动统计信息更新（非常有用），但不包括自动重组和自动备份。后两项功能可能也非常有用，但您必须根据环境对其进行配置和调度才能获得最佳结果。缺省情况下，自动统计信息概要分析功能应该保持处于禁用状态。此功能的开销相当高，因此只应该在受控环境下临时用于分析复杂的语句。

显式的配置设置

某些参数没有自动设置，配置顾问程序将不会设置这些参数。您需要明确地设置这些参数。这里，只讨论对性能有所影响的参数。

- **logpath** 或 **newlogpath** 确定事务日志的位置。即使配置顾问程序也无法为您确定日志的存储位置。如上所述，最重要的一点是，它们不应与其他 DB2 对象（例如表空间）共享磁盘设备，也不应留在数据库路径下的缺省位置。理想情况下，应该将事务日志放在吞吐量足以确保不会引起瓶颈的专用存储器上。
- **logbufsz** 确定事务记录器内部缓冲区的大小，以 4-KB 页计。缺省值（只有 8 页）过小，无法在生产环境中提供良好的性能。配置顾问程序始终会增大此参数，但可能并不足够，这取决于输入参数。通常，256-1000 页的值是不错的范围，并且在数据库服务器整体方案的内存总量中只占非常小的一部分。
- **mincommit** 控制*组落实*，这将致使 DB2 系统尝试同时对 *n* 个落实事务进行批处理。对于当前事务记录器设计而言，这不大可能是期望的行为。请保留 **mincommit** 的缺省值 1。
- **buffpage** 确定分配给每个将大小定义为 -1 的缓冲池的页数。最佳实践是，忽略 **buffpage**，并明确地设置在 SYSCAT.BUFFERPOOLS 中有条目的缓冲池的大小或者让 STMM 自动调整缓冲池大小。
- **diagpath** 确定各种非常有用的 DB2 诊断文件的位置。此参数对性能的影响通常非常有限，但在分区数据库环境中的情况可能并非如此。在所有分区中，**diagpath** 的缺省位置通常是通过 NFS 安装的共享路径。最佳实践是，对于每个分区，覆盖 **diagpath** 并指定本地的非 NFS 目录。这将防止各个分区尝试更新同一个文件并写入诊断消息。而是，这些消息将保存在每个分区的本地位置，从而大大减少争用情况。
- **DB2_PARALLEL_IO** 不是配置参数，而是 DB2 注册表变量。DB2 系统经常使用由磁盘阵列（操作系统将其视为单一设备）组成的存储器或者跨多个设备的文件系统。因此，在缺省情况下，DB2 数据库系统每次只对表空间容器发出一个预取请求。此行为基于对以下事实的了解：对单一设备发出的多个请求将被序列化。但是，如果容器驻留在磁盘阵列中，那么就有机会同时向其分派多个预取请求，而不会进行序列化。这就是 **DB2_PARALLEL_IO** 所起的作用。此参数告知 DB2 系统，可以并行地向单一容器发出多个预取请求。最简单的设置是 `DB2_PARALLEL_IO=*`（表示所有容器都驻留在多个磁盘中，在本例中，假定磁盘数为 7），但其他设置也控制并行程度以及哪些表空间受影响。例如，如果您知道容器驻留在由 4 个磁盘组成的 RAID-5 阵列中，那么可以将 **DB2_PARALLEL_IO** 设置为 *:3。特定的值能否提高性能还取决于扩展数据块大小、RAID 段大小以及使用同一组磁盘的容器数。

与 SAP 和其他 ISV 环境相关的注意事项

如果正在为 SAP 之类的 ISV 应用程序运行 DB2 数据库服务器，那么可能有一些兼顾到特定应用程序的最佳实践准则。最直接的机制是 DB2 注册表变量 **DB2_WORKLOAD**，可以将它设置为一个值以便启用要针对特定环境和工作负载进行优化的聚集注册表变量。**DB2_WORKLOAD** 的有效设置包括：1C、CM、COGNOS_CS、FILENET_CM、MAXIMO、MDM、SAP、TPM、WAS、WC 和 WP。

其他建议和最佳实践也可能适用，例如代码页或代码集以及整理顺序选项，这是因为它们必须设置为预先确定的值。有关详细信息，请参阅应用程序供应商的文档。

对于许多 ISV 应用程序（例如 SAP Business One）而言，AUTOCONFIGURE 命令可以成功地用于定义初始配置。但是，此命令不应用于 SAP NetWeaver 安装，原因是 SAP

安装期间应用了一组初始的 DB2 配置参数。此外，SAP 有一种功能强大的备用最佳实践方法（SAP 注意事项），此方法描述了首选的 DB2 参数设置；例如，SAP 注意事项 1086130 – DB6: DB2 9.5 标准参数设置。

使用 DB2 DPF 功能时，必须特别注意 SAP 应用程序。SAP 主要将 DPF 用于它的 SAP NetWeaver Business Intelligence（Business Warehouse）产品。建议的布局将 DB2 系统目录、维和主表以及 SAP 基本表放在分区 0 中。与其他 DB2 DPF 安装相比，这将在此分区中产生另一工作负载。由于 SAP 应用程序服务器在此分区中运行，因此可能会将多达 8 个处理器分配到此分区。随着 SAP BW 工作负载的并行度变得更高，并且在同时运行许多短查询的情况下，SAP BI 的分区数通常小于其他应用程序情况下的分区数。换而言之，每个数据分区需要多个 CPU。

实例配置

启动新的 DB2 实例时，您可以通过多个步骤来建立基本配置。

- 您可以使用配置顾问程序来获取有关缓冲池大小、数据库配置参数和数据库管理器配置参数的初始值的建议。要使用配置顾问程序，请对现有数据库指定 AUTOCONFIGURE 命令，或者在 CREATE DATABASE 命令中指定 AUTOCONFIGURE 选项。您可以显示建议的值，也可以使用 CREATE DATABASE 命令的 APPLY 选项来应用建议值。这些建议是根据您提供的输入以及顾问程序收集的系统信息生成的。
- 您可以使用配置助手来配置和维护数据库对象、添加新对象、绑定应用程序、设置数据库管理器配置参数以及导入和导出配置信息。要打开配置助手，请调用 db2ca 命令。对于实例配置，配置助手将帮助您设置数据库管理器配置参数、设置 DB2 注册表变量、配置另一个实例或者复位配置。
- 请查阅摘要表（请参阅“配置参数摘要”），该表列示并简要地描述了每个可用于数据库管理器或数据库的配置参数。在这些摘要表中，有一列指示调整特定参数是有可能导致高性能、中等性能、低性能还是无变化。请使用这些表来查找可以帮助您在环境中实现最大性能增益的参数。
- 使用 ACTIVATE DATABASE 命令来激活数据库并启动所有必需的数据库服务，以使任何应用程序能够连接到该数据库以及使用该数据库。在分区数据库环境中，此命令将在所有数据库分区中激活该数据库，并且将消除第一个应用程序进行连接时初始化数据库所需耗用的启动时间。

表空间设计

磁盘存储器性能因素

硬件特征（例如磁盘存储器配置）会对系统性能产生很大影响。

磁盘存储器配置方面的下列一个或多个因素会影响性能:

- 存储器的划分

  在索引与数据之间以及在各个表空间之间划分有限存储器容量的良好程度，在很大程度上决定了系统在不同情况下的性能表现。
- 磁盘 I/O 的分布

在多个设备和控制器之间平衡磁盘 I/O 需求的良好程度，将影响数据库管理器从磁盘中检索数据的速度。

- 磁盘子系统核心性能指标

  每秒钟的磁盘操作数或者每秒传输的数据量（以兆字节计）会对整体系统的性能产生非常大的影响。

表空间对查询优化的影响

表空间的某些特征会对查询编译器选择的访问方案产生影响。

这些特征包括:

- 容器特征

  容器特征会显著影响查询执行期间的 I/O 成本。查询优化器在选择访问方案时将考虑这些 I/O 成本，其中包括访问不同表空间中的数据时产生的任何成本差别。优化器使用 SYSCAT.TABLESPACES 目录视图中的两列来帮助估计访问表空间中的数据时的 I/O 成本:

  – OVERHEAD 提供容器所需时间的估计（以毫秒计），在该时间过后才将任何数据读入内存。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。

    您可以使用以下公式来估计开销成本:

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

    其中:

    - 0.5 表示半转的平均开销
    - 对每个完整旋转计算旋转等待时间（以毫秒计），如下所示:

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

      其中:

      - 除以每分钟转数以获得每转所需的分钟数
      - 乘以 60 秒/分
      - 乘以 1000 毫秒/秒

    例如，假定磁盘速度为每分钟 7200 转。使用旋转延迟时间公式的结果如下:

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

    此值可用于估计开销（假定平均寻道时间为 11 毫秒）:

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

  – TRANSFERRATE 提供将一页数据读入内存所需时间的估计（以毫秒计）。

    如果每个表空间容器都是单个的物理磁盘，那么可使用以下公式来估计传输成本（以毫秒/页计）:

    ```
    TRANSFERRATE = (1 / spec_rate) * 1000 / 1024000 * page_size
    ```

    其中:

- 除以 *spec_rate*（这是磁盘的传输速率规格，以兆字节/秒计），以获取每兆字节所需的秒数
- 乘以 1000 毫秒/秒
- 除以 1024000 个字节/兆字节
- 乘以页大小（以字节计）；例如，4-KB 页的大小为 4096 字节

例如，假定磁盘的速率规格为每秒 3 兆字节。那么:

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

即，每页大约 1.3 毫秒。

如果表空间容器不是单个的物理磁盘，而是磁盘阵列（例如 RAID），那么估计 TRANSFERRATE 时必须考虑其他事项。

如果阵列相对较小（即，瓶颈处于磁盘级），那么可用磁盘数乘以 *spec_rate*。但是，如果阵列较大，那么瓶颈可能不是在磁盘级别，而是存在于其他某个 IO 子系统组件上，例如磁盘控制器、IO 总线或系统总线。在这种情况下，不能假定 IO 的吞吐量能力是 *spec_rate* 与磁盘数目的乘积。而是，您必须测量顺序扫描期间的实际 IO 速率（以兆字节计）。例如，`select count(*) from big_table` 所产生的顺序扫描的大小可能是几兆字节。在这种情况下，请将结果除以 BIG_TABLE 所在表空间所包含的容器数。然后，使用该结果替换以上公式中的 *spec_rate*。例如，在扫描包含 4 个容器的表空间中的某个表时，所测得的顺序 IO 速率 100 MB 意味着每个容器 25 兆字节，即，TRANSFERRATE 为 (1/25) * 1000 / 1024000 * 4096 = 0.16 毫秒/页。

分配给一个表空间的各个容器可能在不同的物理磁盘上。为了获得最佳结果，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，那么在设置 OVERHEAD 和 TRANSFERRATE 时，您应使用平均值。

您可以根据硬件规格或通过试验来获得这些列的特定于介质的值。可以在 CREATE TABLESPACE 和 ALTER TABLESPACE 语句中指定这些值。

- 预取

在考虑访问表空间中的数据时的 I/O 成本时，优化器还将考虑从磁盘预取数据页和索引页可能对查询性能产生的潜在影响。预取操作可以降低将数据读入缓冲池所引起的开销。

优化器将使用 SYSCAT.TABLESPACES 目录视图中 PREFETCHSIZE 和 EXTENTSIZE 列的信息来估计预取量。

– 仅当创建表空间时，才可以设置 EXTENTSIZE。通常，大小为 4 页或 8 页的扩展数据块已足够。
– 在创建或更改表空间时，可设置 PREFETCHSIZE。缺省预取大小由 **dft_prefetch_sz** 数据库配置参数值确定。请查看有关估算此参数的建议并按需要进行更改，或者将其设置为 AUTOMATIC。

对表空间进行更改之后，请考虑执行 RUNSTATS 实用程序以收集有关索引的最新统计信息，从而确保查询优化器选择有可能的最佳数据访问方案，然后再重新绑定应用程序。

数据库设计

表

标准表的表和索引管理

在标准表中，数据在逻辑上按数据页的列表进行组织。这些数据页根据表空间的扩展数据块大小在逻辑上分组到一起。

例如，如果扩展数据块大小是 4，那么第 0 至 3 页是第一个扩展数据块的组成部分，第 4 至 7 页是第二个扩展数据块的组成部分，依此类推。

根据数据页大小以及记录大小的不同，每个数据页中包含的记录数也会有所变化。大多数页仅包含用户记录。但是，少数页包含由数据服务器用于管理表的特殊内部记录。例如，在标准表中，每 500 个数据页就有一个可用空间控制记录（FSCR），如图 8 所示。这些记录用于映射后续每 500 个数据页（直到下一个 FSCR 为止）中可供新记录使用的可用空间。

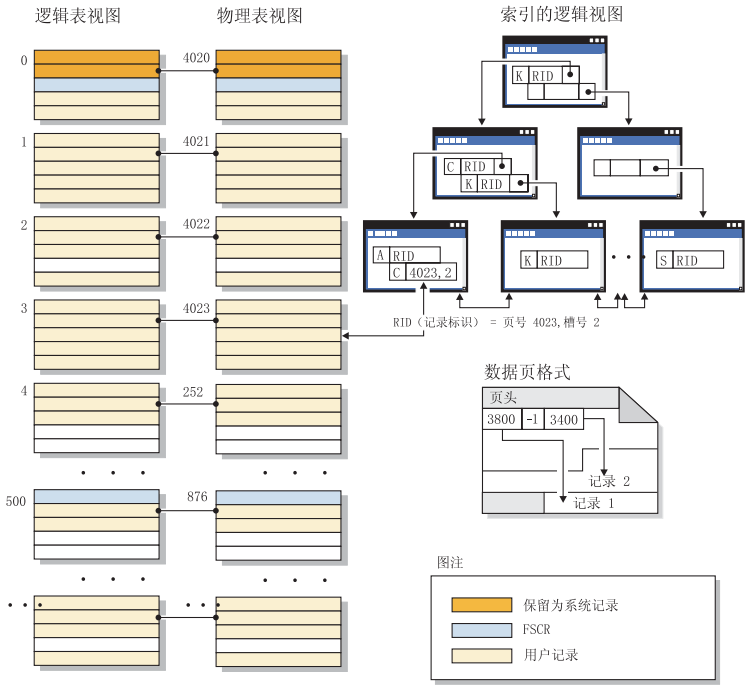


图 8. 标准表的逻辑表、记录和索引结构

在逻辑上，索引页组织成 B 树，这可以有效地找到具有特定键值的表记录。索引页中的项数不固定，而是取决于键的大小。对于数据库管理的空间（DMS）表空间中的表而

言，索引页中的记录标识（RID）使用相对于表空间的页号，而不是相对于对象的页号。这使索引扫描操作能够直接访问数据页，而不需要扩展数据块映像页（EMP）来进行映射。

每个数据页都具有相同的格式。每一页的最前面都是页头，后面跟着槽目录。槽目录中的每个条目都与该页中的另一个记录相对应。槽目录中的条目代表记录开始位置在数据页中的字节偏移。值为 -1 的条目与已删除的记录相对应。

记录标识和页

记录标识由页号及随后的槽号组成（图 9）。索引记录还包含名为 ridFlag 的附加字段。ridFlag 存储关于索引中键的状态信息，例如它们是否被标记为“已删除”。在使用索引来标识 RID 之后，便使用该 RID 来标识正确的数据页以及该页中的槽号。对记录指定 RID 之后，该 RID 在表被重组之前将不会更改。

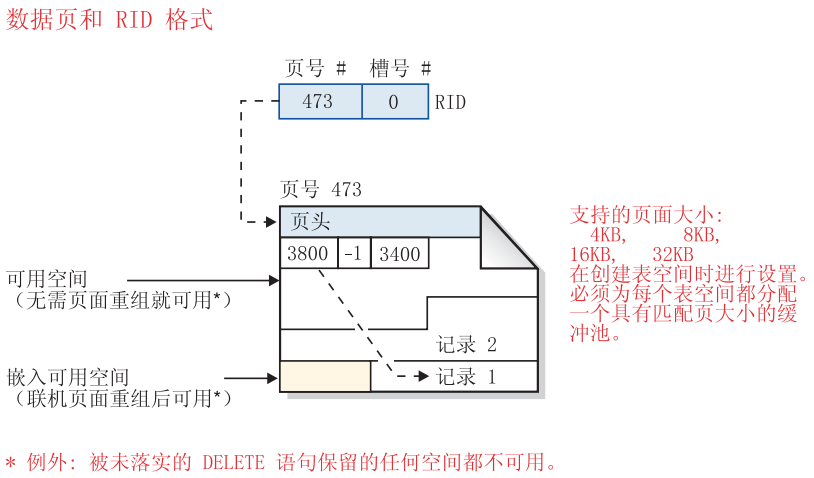


图 9. 数据页和记录标识（RID）格式

重组表页时，实际删除记录后在页中留下的嵌入式可用空间将被转换为可使用的可用空间。

DB2 数据服务器支持不同的页大小。对于有可能按顺序访问行的工作负载，请使用较大的页大小。例如，顺序访问通常用于决策支持应用程序或大量使用临时表的场合。对于有可能随机访问行的工作负载，请使用较小的页大小。例如，在联机事务处理（OLTP）环境中，通常执行随机访问。

标准表中的索引管理

DB2 索引使用经过优化的 B 树实现，此实现基于一种进行预写记录的高效率高并行性索引管理方法。B 树索引安排成平衡的页层次结构，从而通过在项被插入或删除时重新排列数据键来最大程度地缩短访问时间。

经过优化的 B 树实现的叶子页包含双向指针，这使单一索引同时支持正向或反向扫描。除进行 90/10 分割（这意味着，索引键的最高 10% 放在新页中）的高键页以外，索引

页通常进行对半分割。这种类型的索引页分割对于特定工作负载而言非常有用，这些工作负载的插入操作通常使用新的高键值完成。

只有在已对表挂起 X 锁定的情况下，才会从索引页中除去已删除的索引键。如果无法立即除去键，那么会将其标记为“已删除”而稍后将其除去。

如果在创建索引时已通过对 MINPCTUSED 指定正数值启用联机索引整理碎片功能，那么可以通过联机方式来合并索引叶子页。MINPCTUSED 表示索引叶子页中已用空间量所占的最小百分比。除去某个键后，如果索引页中的已用空间量低于此值，那么数据库管理器会尝试将其余键与相邻页中的键合并。如果有足够的空间，那么执行合并，并删除一个索引叶子页。由于仅当从索引页中除去键时才会以联机方式整理碎片，所以如果键仅仅被标记为“已删除”但并未实际地从页中除去，那么不会以联机方式整理碎片。联机索引整理碎片功能可以改进空间复用情况，但如果 MINPCTUSED 值过大，那么执行合并所需的时间将增加，并且成功进行合并的可能性将降低。建议的 MINPCTUSED 值是 50% 或更小。

CREATE INDEX 语句的 INCLUDE 子句允许对索引叶子页指定除键列以外的一个或多个列。这些“包括列”与索引 B 树的排序操作无关，但可以增加适合于纯索引访问的查询数。但是，它们也会增加索引空间需求，并且，如果所包括的列被频繁更新，那么还可能会增加索引维护成本。更新包括列的维护成本低于更新键列，但高于更新并非作为索引组成部分的列。

MDC 表的表和索引管理

多维集群（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 表按页组织，这些页包含分为许多列的数据行。每一页中的行由记录标识（RID）标识。但是，MDC 表的各个页还分组为具有扩展数据块大小的块。例如，第 52 页的图 10 显示了扩展数据块大小为 4 的表。前四页（编号为 0 到 3）是表中的第一个块。接着的四页（编号为 4 到 7）是表中的第二个块。

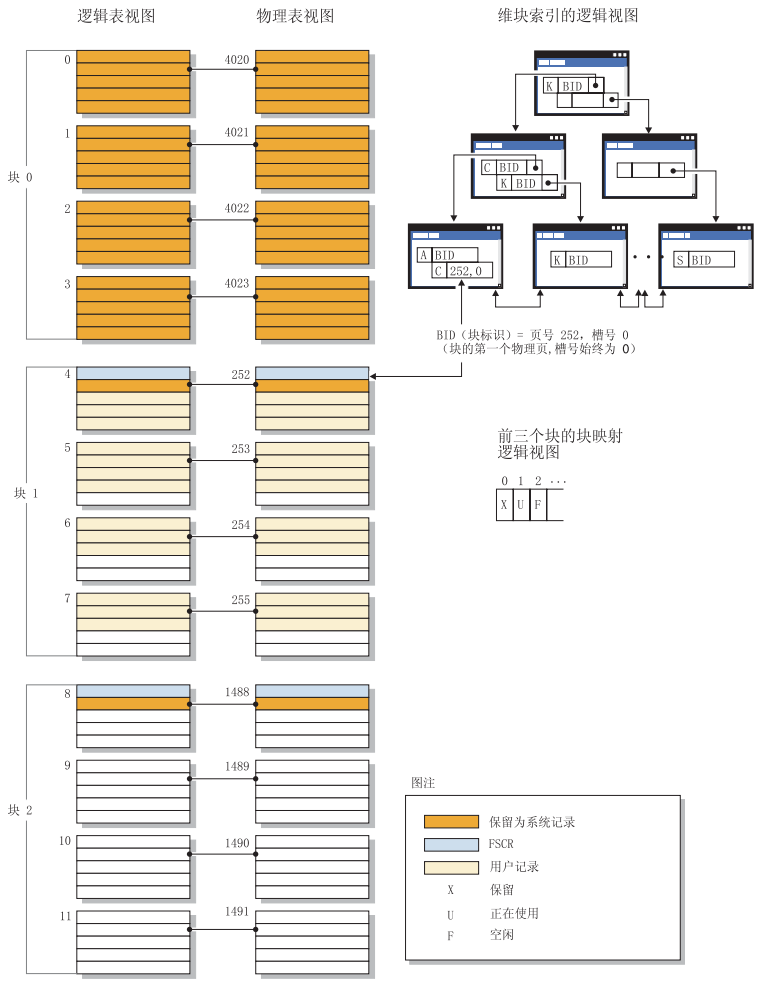


图 10. MDC 表的逻辑表、记录和索引结构

第一个块包含 DB2 服务器用来管理表的特殊内部记录，其中包括可用空间控制记录（FSCR）。在后续块中，第一页包含 FSCR。FSCR 为新记录映射块中每一页中存在的可用空间。将记录插入到表中时，将使用这部分可用空间。

顾名思义，MDC 表对多个维的数据进行集群。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。创建 MDC 表时，将自动创建下面这两个索引:

- 维块索引，它包含指向单个维的每个被占用块的指针
- 组合块索引，它包含所有维键列并用于在插入和更新活动期间维护集群

当优化器确定特定查询的最高效访问方案时，它将考虑使用维块索引的访问方案。当查询包含应用于维值的谓词时，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以及访存那些扩展数据块的内容。由于扩展数据块在物理上是磁盘中相邻的页，因此这将最大程度地减少 I/O 操作，从而提高性能。

如果对数据访问方案进行的分析表明特定 RID 索引有助于提高查询性能，那么您可以创建这样的索引。

索引

索引结构

数据库管理器使用 B+ 树结构来存储索引。

如图 11 所示，B+ 树由多层组成；“rid”表示记录标识（RID）。

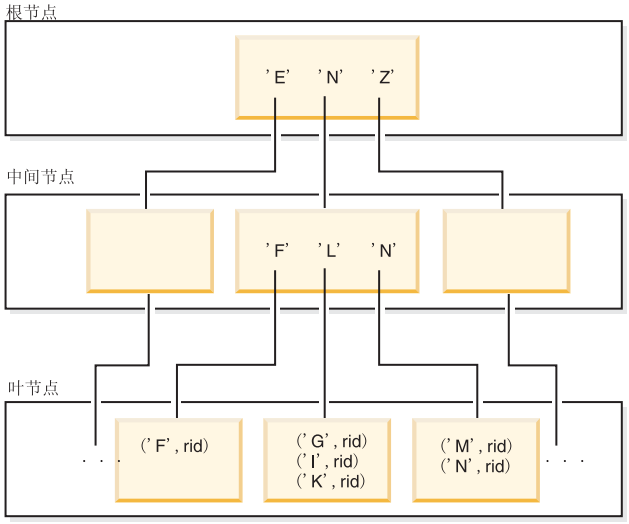


图 11. B+ 树索引的结构

顶层称为*根节点*。底层由*叶节点*组成，这些节点存储索引键值，并通过指针指向包含相应数据的表行。根节点层与叶节点层之间的那些层称为*中间节点*。

当索引管理器查找特定的索引键值时，它从根节点开始搜索索引树。根节点对于下一层中的每个（中间）节点都包含一个键。每个键的值都是下一层的相应节点的最大现有键值。例如，假定索引包含三层（如上图所示）。为了查找特定的索引键值，索引管理器将搜索根节点，以查找第一个大于或等于搜索键值的键值。根节点键指向特定的中间节点。索引管理器将按照此过程遍历每个中间节点，直到找到包含所需索引键的叶节点为止。

假定图 11 中正在查找的键是“I”。根节点中第一个大于或等于“I”的键是“N”，它指向下一层的中间节点。该中间节点中第一个大于或等于“I”的键是“L”，它指向“I”的索引键及其相应 RID 所在的特定叶节点。RID 标识基本表中相应的行。

叶节点层也可以包含指向上一层叶节点的指针。这些指针使索引管理器能够在找到范围内的某个值之后按任一方向扫描叶节点以检索某个范围内的值。只有在创建索引时指定了 ALLOW REVERSE SCANS 选项的情况下，才可能有按任一方向执行扫描的能力。

对于多维集群（MDC）表而言，将为您对表指定的每个集群维自动创建块索引。还将创建一个组合块索引；此索引包含每一列中涉及任何表维的键部分。这种索引包含指向块标识（BID）而不是 RID 的指针，并有助于改进数据访问。

为索引的叶子页中每个 RID 存储的单字节 *ridFlag* 用于将该 RID 标记为“已被以逻辑方式删除”，以便稍后以物理方式将其除去。对于索引中的每个可变长度列，将有另外一个字节存储列值的实际长度。在落实更新或删除操作之后，可以将已标记为“已删除”的键除去。

索引清除和维护

在创建索引之后，除非您保持索引紧凑并且组织有序，否则性能会随着时间推移而下降。

下列建议将帮助您使索引尽可能小并保持高效:

- 启用联机索引整理碎片功能

  创建索引时，指定 MINPCTUSED 子句。必要时，删除并重新创建现有的索引。
- 频繁地执行落实；如果不可能频繁地执行落实，那么以显式方式或者通过锁定升级来获取表级别的 X 锁定。

  执行落实后，可以从表中实际地除去被标记为“已删除”的索引键。将所要删除的键标记为“已删除”之后，对表的 X 锁定将允许实际地删除这些键，如下所述。
- 使用 REORGCHK 命令来帮助确定何时重组索引或表以及何时使用带有 CLEANUP ONLY 子句的 REORG INDEXES 命令。

  为了允许在重组期间对索引进行读写访问，请使用带有 ALLOW WRITE ACCESS 选项的 REORG INDEXES 命令。

  为了允许在清除期间对索引进行读写访问，请使用带有 ALLOW WRITE ACCESS 选项的 REORG INDEXES 命令。对于数据分区表，如果还指定了 CLEANUP ONLY 子句，那么可以对 REORG INDEXES...ALL 命令指定 ALLOW WRITE ACCESS 子句。

  对于 DB2 版本 9.7 修订包 1 和更高版本的发行版，对数据分区表发出带有 ON DATA PARTITION 子句的 REORG INDEXES 命令，以重组所指定分区的分区索引。在重组索引期间，不受影响的分区将保持可读，而只有受影响的分区才可写。

在下列情况下，将清除被标记为“已删除”的索引键:

- 在后续插入、更新或删除活动期间

  在插入键期间，将清除被标记为“已删除”以及确定已落实的键（如果这样可以避免需要执行页分割并防止索引增大的话）。

  在删除键期间，如果某页中的所有键都被标记为“已删除”，那么将尝试查找另一个其中所有键都被标记为“已删除”并且所有那些删除操作都已落实的索引页。如果找到这样的页，那么会将其从索引树中删除。删除键时，如果存在对表的 X 锁定，那么将实际地删除该键，而不只是将其标记为“已删除”。在物理删除期间，如果同一页上的任何已删除的键被标记为“已删除”且已知将要落实，那么也会除去这些键。
- 执行 REORG INDEXES 命令并指定了 CLEANUP 选项时

CLEANUP ONLY PAGES 选项将搜索并释放其中的所有键都被标记为“已删除”并已知要落实的索引页。

CLEANUP ONLY ALL 选项不仅释放其中所有键都被标记为“已删除”并已知要落实的索引页，还将从包含一些未删除的记录标识（RID）的页中除去被标记为“已删除”并已知要落实的 RID。此选项还将尝试合并相邻叶子页，前提是合并后的叶子页中的可用空间量至少为 PCTFREE。PCTFREE 值是在创建索引时定义的。缺省的 PCTFREE 值是 10%。如果可以合并两页，那么将释放其中一页。

对于数据分区表，建议您在完成异步清除索引操作之后调用 RUNSTATS 命令。要确定表中是否存在已拆离的数据分区，请查询 SYSCAT.DATAPARTITIONS 目录视图中的 STATUS 字段，然后查找值“L”（逻辑上已拆离）、“D”（具有拆离从属表（例如，具体化查询表）的已拆离分区）或“I”（清除索引）。

- 重建索引时（或者，对于数据分区索引，重建索引分区时）

  重建索引的实用程序包括:
  - REORG INDEXES（未指定任何 CLEANUP 选项时）
  - 带有 ON DATA PARTITION 子句的 REORG INDEXES
  - 带有 ON DATA PARTITION 子句的 REORG TABLE
  - REORG TABLE（未指定 INPLACE 选项时）
  - IMPORT（指定了 REPLACE 选项时）
  - LOAD（指定了 INDEXING MODE REBUILD 选项时）

异步索引清除

异步索引清除（AIC）是在导致索引条目失效的操作后执行的延迟型索引清除操作。根据索引类型的不同，条目可以是记录标识（RID）或块标识（BID）。在后台以异步方式运行的索引清除程序将除去无效的索引条目。

AIC 将加快从分区表中拆离数据分区这一进程的速度，并且在分区表包含一个或多个非分区索引时启动。在这种情况下，AIC 将除去所有引用了已拆离的数据分区的非分区索引条目以及任何伪删除的条目。在清除所有索引之后，将从系统目录中除去与已拆离的数据分区相关联的标识。在 DB2 版本 9.7 修订包 1 和更高版本的发行版中，通过异步分区拆离任务来启动 AIC。

在 DB2 版本 9.7 修订包 1 之前，如果该分区表有从属具体化查询表（MQT），那么要在执行 SET INTEGRITY 语句之后才会启动 AIC。

在 AIC 执行过程中，将维持正常的表访问。访问索引的查询将忽略尚未清除的任何无效条目。

在大多数情况下，将对与分区表相关联的每个非分区索引启动一个清除程序。内部任务分发守护程序负责将 AIC 任务分发到适当的表分区并指定数据库代理程序。分发守护程序和清除程序代理程序是内部系统应用程序，它们分别以应用程序名称 db2taskd 和 db2aic 出现在 LIST APPLICATIONS 命令输出中。为了防止意外中断，不能强制执行系统应用程序。只要数据库处于活动状态，分发守护程序就一直处于联机状态。清除程序在完成清除之前将保持活动状态。如果数据库在清除期间被停用，那么在您重新激活数据库之后，AIC 将继续进行。

AIC 对性能的影响

AIC 对性能的影响很小。

需要进行瞬时行锁定测试以确定是否落实了伪删除的条目。但是，由于从未获取锁定，因此并行性不会受影响。

每个清除程序都获取最小表空间锁定（IX）和表锁定（IS）。如果清除程序确定其他应用程序正在等待这些锁定，就会释放这些锁定。如果发生这种情况，那么清除程序就会暂挂处理 5 分钟。

清除程序与实用程序调速功能集成。缺省情况下，每个清除程序的实用程序影响优先级为 50。您可以使用 SET UTIL_IMPACT_PRIORITY 命令或 db2UtilityControl API 来更改此优先级。

监视 AIC

可以使用 LIST UTILITIES 命令来监视 AIC。每个索引清除程序都作为一个单独的实用程序出现在输出中。以下是 LIST UTILITIES SHOW DETAIL 命令输出的示例:

```
标识                                 = 2
类型                                 = ASYNCHRONOUS INDEX CLEANUP
数据库名称                           = WSDB
分区号                               = 0
描述                                 = 表: USER1.SALES, 索引: USER1.I2
开始时间                             = 12/15/2005 11:15:01.967939
状态                                 = 正在执行
调用类型                             = 自动
正在调速:
   优先级                            = 50
进度监视:
      总计工作                       = 5 页
      已完成的工作                   = 0 页
      开始时间                       = 12/15/2005 11:15:01.979033

标识                                 = 1
类型                                 = ASYNCHRONOUS INDEX CLEANUP
数据库名称                           = WSDB
分区号                               = 0
描述                                 = 表: USER1.SALES, 索引: USER1.I1
开始时间                             = 12/15/2005 11:15:01.978554
状态                                 = 正在执行
调用类型                             = 自动
正在调速:
   优先级                            = 50
进度监视:
      总计工作                       = 5 页
      已完成的工作                   = 0 页
      开始时间                       = 12/15/2005 11:15:01.980524
```

在本示例中，有两个清除程序正在对 USERS1.SALES 表进行操作。一个清除程序正在处理索引 I1，另一个清除程序正在处理索引 I2。进度监视部分显示需要清除的估计总索引页数和当前的干净索引页数。

状态字段指示清除程序的当前状态。正常状态是“正在执行”，但如果清除程序正在等待被指定到可用的数据库代理程序，或者清除程序由于锁定争用而临时暂挂，那么清除程序可能处于“正在等待”状态。

注意，由于每个数据库分区仅对该数据库分区中运行的任务指定标识，因此不同数据库分区中的不同任务可能具有相同的实用程序标识。

MDC 表的异步索引清除

您可以采用异步索引清除（AIC）功能来提高转出删除性能。转出删除是一种从多维集群（MDC）表中删除合格数据块的高效方法。AIC 是指在执行使索引条目失效的操作之后延迟清除索引。

在标准转出删除期间，将以同步方式清除索引。如果表包含许多记录标识（RID）索引，那么需要花费相当多时间来除去引用了正被删除的表行的索引键。您可以通过指定在删除操作落实后清除这些索引来提高转出速度。

要对 MDC 表利用 AIC，必须显式地启用*延迟索引清除转出*机制。可以通过两种方法来指定延迟转出：将注册表变量 **DB2_MDC_ROLLOUT** 设置为 DEFER 或者发出 SET CURRENT MDC ROLLOUT MODE 语句。在执行延迟索引清除转出操作期间，会将各个块标记为已转出，但不会更新 RID 索引，直到该事务落实之后才会进行更新。在删除操作期间将清除块标识（BID）索引，这是因为它们并不需要执行行级别处理。

转出删除操作落实时将调用 AIC 转出；如果数据库已关闭，那么在数据库重新启动之后首次访问该表时也会调用 AIC 转出。在 AIC 进行期间，对索引执行的查询（包括那些需要访问正被清除的索引的查询）将成功。

每个 MDC 表都有一个协调清除程序。多个转出的索引清除操作都合并在清除程序中，后者将为每个 RID 索引生成一个清除代理程序。清除代理程序以并行方式更新 RID 索引。清除程序还与实用程序调速功能集成。缺省情况下，每个清除程序的实用程序影响优先级为 50（可接受的值为 1 到 100，0 表示无调速功能）。您可以使用 SET UTIL_IMPACT_PRIORITY 命令或 db2UtilityControl API 来更改此优先级。

注：在 DB2 版本 9.7 和更高版本的发行版中，不支持对具有分区 RID 索引的数据分区 MDC 表执行延迟清除转出。仅支持 NONE 和 IMMEDIATE 方式。如果 **DB2_MDC_ROLLOUT** 注册表变量设置为 DEFER，或者 CURRENT MDC ROLLOUT MODE 专用寄存器设置为 DEFERRED 以覆盖 **DB2_MDC_ROLLOUT** 设置，那么清除转出类型将为 IMMEDIATE。

如果 MDC 表仅存在非分区 RID 索引，那么支持执行延迟索引清除转出。MDC 块索引可以是分区索引，也可以是非分区索引。

监视延迟索引清除转出操作的进度

由于直到完成清除之后才能复用 MDC 表中已转出的块，因此监视延迟索引清除转出操作的进度将很有用。请使用 LIST UTILITIES 命令来显示被清除的每个索引的实用程序监视器条目。您还可以通过使用 SYSPROC.ADMIN_GET_TAB_INFO_V95 表函数或 GET SNAPSHOT 命令，来检索数据库中在转出删除后处于暂挂异步清除状态的 MDC 表块的总数（BLOCKS_PENDING_CLEANUP）。

在 LIST UTILITIES SHOW DETAILS 命令的以下样本输出中，进度由每个索引中已被清除的页数指示。每个阶段都代表一个 RID 索引。

```
标识                                 = 2
类型                                 = MDC ROLLOUT INDEX CLEANUP
数据库名称                           = WSDB
分区号                               = 0
描述                                 = TABLE.<schema_name>.<table_name>
```

```
开始时间                          = 06/12/2006 08:56:33.390158
状态                              = 正在执行
调用类型                          = 自动
正在调速:
   优先级                         = 50
进度监视:
   估计完成百分比                 = 83
   阶段号                         = 1
      描述                        = <schema_name>.<index_name>
      工作总计                    = 13 页
      已完成的工作                = 13 页
      开始时间                    = 06/12/2006 08:56:33.391566
   阶段号                         = 2
      描述                        = <schema_name>.<index_name>
      工作总计                    = 13 页
      已完成的工作                = 13 页
      开始时间                    = 06/12/2006 08:56:33.391577
   阶段号                         = 3
      描述                        = <schema_name>.<index_name>
      总计工作                    = 9 页
      已完成的工作                = 3 页
      开始时间                    = 06/12/2006 08:56:33.391587
```

联机索引整理碎片

联机索引整理碎片功能由索引叶子页的最小已用空间量阈值启用，此阈值可由用户定义。

从叶子页中删除索引键时，如果超出此阈值，那么将检查相邻的索引叶子页以确定能否合并这两个叶子页。如果某一页有足够的空间，并且有可能合并两个相邻的页，那么将立即在后台进行合并，并且将删除所产生的空索引叶子页。

如果现有索引要求有能力进行联机合并，那么必须将其删除，然后使用 CREATE INDEX 语句重新创建这些索引并指定 MINPCTUSED 子句。MINPCTUSED 的建议值是小于 50，这是因为，我们的目标是将两个相邻的索引叶子页合并。零值（缺省值）将禁用联机整理碎片功能。

在联机索引整理碎片期间，不会合并索引非叶子页。但是，将删除空的非叶子页以供同一个表的其他索引复用。要为数据库管理的空间（DMS）存储模型中的其他对象释放这些非叶子页，或者要释放系统管理的空间（SMS）存储模型中的磁盘空间，请对表和索引执行完全重组，这将使索引尽可能小。在联机索引整理碎片期间，索引中的层数不会减少。

如果已对表挂起 X 锁定，那么删除键期间将从索引页中实际地除去键；在这种情况下，联机索引整理碎片有效。但是，如果删除键期间未对表挂起 X 锁定，那么将把这些键标记为“已删除”但不会从索引页中实际地将其除去，并且不会尝试进行索引整理碎片。

要对索引整理碎片，而不考虑 MINPCTUSED 的值，请调用 REORG INDEXES 命令并指定 CLEANUP ONLY ALL 选项。将对两个相邻的叶子页进行合并，前提是合并后的页中的可用空间量至少为 PCTFREE。您可以在创建索引时指定 PCTFREE；它的缺省值是 10（百分比）。

使用关系索引来提高性能

访问表数据时，可使用索引来提高性能。访问关系数据时，将使用关系索引；访问 XML 数据时，将使用基于 XML 数据的索引。

尽管查询优化器决定是否使用关系索引来访问关系表数据，但您负责确定哪些索引可以提高性能并创建那些索引。例外情况只有维块索引和组合块索引，当您创建多维集群（MDC）表时，系统将为每个维自动创建这些索引。

在创建关系索引或更改预取大小之后，请执行 RUNSTATS 实用程序以收集新的索引统计信息。您应该定期执行 RUNSTATS 实用程序以保持统计信息最新；如果没有有关索引的最新统计信息，那么优化器将无法确定查询的最佳数据访问方案。

要确定是否在特定的程序包中使用了关系索引，请使用说明工具。要获取有关一个或多个 SQL 语句可以利用的关系索引的建议，请使用 db2advis 命令来启动设计顾问程序。

关系索引相对于无索引的优点

如果不存在表的索引，那么必须对 SQL 查询中引用的每个表执行表扫描。表越大，此扫描所耗用的时间就越长，这是因为表扫描要求按顺序访问每一行。虽然对于需要表中的大多数行的复杂查询来说，使用表扫描效率可能更高，但对于只返回部分表行的查询而言，索引扫描可以更有效地访问表行。

如果在 SELECT 语句中引用了关系索引列，并且优化器估计索引扫描比表扫描快，那么优化器将选择索引扫描。索引文件通常较小，因此读取它所需的时间比读取整个表所需的时间要少，在表较大时尤其如此。并且，可能不必扫描整个索引。应用于索引的谓词将减少必须从数据页读取的行数。

如果对输出的排序要求可以与某个索引列匹配，那么按列顺序扫描索引将按正确顺序检索各行，而不需要执行排序操作。注意，存在所查询的表的关系索引并不保证结果集已进行排序。只有 ORDER BY 子句才能确保结果集进行排序。

关系索引还可以包含“包括列”，这些列是已建立索引的行中未建立索引的列。这样的列使优化器能够只从索引中检索所需的信息，而不必访问表本身。

关系索引相对于无索引的缺点

尽管索引可以显著缩短访问时间，但是它们也会对性能产生负面影响。在创建索引之前，请考虑多个索引对磁盘空间和处理时间的影响。您应该谨慎地选择索引以满足应用程序的需要。

- 每个索引都需要存储空间。确切的容量取决于表的大小以及关系索引中列的大小和数目。
- 对表执行的每个插入或删除操作都要求对该表的每个索引进行额外的更新。对于每个更改了索引键值的更新操作而言，情况亦如此。
- 每个关系索引都代表另一个可供优化器考虑的潜在访问方案，这将延长查询编译时间。

关系索引规划技巧

设计精良的索引可以使查询能够更方便地访问关系数据。

请使用设计顾问程序（db2advis 命令）来查找特定查询的最佳索引或者一组定义了工作负载的查询的最佳索引。此工具可以提供性能增强建议，例如包括支持反向扫描的列或索引。

下列准则也可以帮助您创建有用的关系索引。

- 高效地检索数据
  - 为了提高数据检索效率，请将*包括列*添加至唯一索引。良好的候选包括符合下列条件的列:
    - 被频繁访问，因此将从纯索引访问中受益
    - 不需要用来限制索引扫描的范围
    - 不影响索引键的排序或唯一性

    例如:

    ```
    create unique index idx on employee (workdept) include (lastname)
    ```

    指定 LASTNAME 作为包括列而不是索引键组成部分意味着，LASTNAME 只存储在索引的叶子页上。
  - 对频繁运行的查询的 WHERE 子句中使用的列创建关系索引。

    在以下示例中，除非 WORKDEPT 列包含许多重复的值，否则 WORKDEPT 的索引有可能使 WHERE 子句受益。

    ```
    where workdept='A01' or workdept='E21'
    ```
  - 创建具有复合键的关系索引，并在该复合键中指定查询中引用的每个列。通过这种方式指定索引时，可以只从索引检索关系数据，这比访问表更高效。

    例如，考虑如下查询:

    ```
    select lastname
      from employee
      where workdept in ('A00','D11','D21')
    ```

    如果对 EMPLOYEE 表的 WORKDEPT 和 LASTNAME 列定义了关系索引，那么通过扫描索引而不是扫描整个表，可以更有效地处理此查询。因为谓词引用了 WORKDEPT，所以此列应该是关系索引的第一个键列。
- 高效地搜索表

  请决定是采用升序键顺序还是降序键顺序，这取决于最常使用的顺序。尽管在 CREATE INDEX 语句中指定 ALLOW REVERSE SCANS 选项后可以按反向方向来搜索值，但是，按指定的索引顺序执行扫描的速度比反向扫描略快。
- 高效地访问较大型的表

  使用关系索引来优化对含有较多数据页的表频繁执行的查询（数据页数记录在 SYSCAT.TABLES 目录视图的 NPAGES 列中）。您应该:
  - 对任何将要用于连接表的列创建索引。
  - 对任何将用于定期搜索特定值的列创建索引。
- 提高更新或删除操作的性能
  - 要提高对父表执行的此类操作的性能，请对外键创建关系索引。
  - 要提高对 REFRESH IMMEDIATE 和 INCREMENTAL 具体化查询表（MQT）执行的此类操作的性能，请对 MQT 的隐式唯一键创建唯一关系索引，该键由 MQT 定义的 GROUP BY 子句中的列组成。
- 提高连接性能

如果多列关系索引中的第一个键列有多个选择，那么请使用等式连接谓词（*expression1 = expression2*）最常指定的那一列，或者使用像第一个键列那样具有最多相异值的那一列。

- 排序
  - 为了最大程度地提高排序操作的速度，请对频繁用于执行关系数据排序的列创建关系索引。
  - 为了避免执行某些排序，在有可能时，请使用 CREATE INDEX 语句来定义主键和唯一键。
  - 创建关系索引，以便按照频繁运行的查询所要求的顺序对行进行排序。DISTINCT、GROUP BY 和 ORDER BY 子句要求执行排序。

    以下示例使用 DISTINCT 子句:

    ```
    select distinct workdept
      from employee
    ```

    数据库管理器可以使用对 WORKDEPT 列定义的索引来消除重复的值。这个索引也可用于对值进行分组，如以下使用 GROUP BY 子句的示例所示:

    ```
    select workdept, average(salary)
      from employee
      group by workdept
    ```

- 对新插入的行进行集群并避免页分割

  定义一个集群索引，这将显著减少需要对表执行重组的机会。使用 CREATE TABLE 语句的 PCTFREE 选项来指定应该在每一页中留下的可用空间量，以便可以适当地插入行。此外，也可以在 LOAD 命令中指定 pagefreespace 文件类型修饰符。

- 降低索引维护成本和存储空间量
  - 避免创建作为其他现有索引的不完整键的索引。例如，如果已对列 A、B 和 C 定义索引，那么对列 A 和 B 定义的另一个索引一般用处不大。
  - 不要对多个列创建任意的索引。不必要的索引不仅浪费空间，还会导致准备时间过长。
    - 对于联机事务处理（OLTP）环境，请为每个表创建一个或两个索引。
    - 对于只读查询环境，可以为每个表创建 5 个以上索引。
    - 对于混合查询和 OLTP 环境，最好为每个表创建 2 到 5 个索引。

- 启用联机索引整理碎片功能

  创建关系索引时，请使用 MINPCTUSED 选项。MINPCTUSED 将启用联机索引整理碎片功能；它指定在索引叶子页中必须使用的最小空间量。

关系索引的性能技巧

您可以通过执行多项操作来确保关系索引的效能。

- 指定大型实用程序堆

  对于您正在为其创建或重组关系索引的表，如果您预期要对该表执行大量更新活动，那么请考虑配置大型实用程序堆（**util_heap_sz** 数据库配置参数），这将有助于提高这些操作的速度。

- 为了避免在对称多处理器（SMP）环境中执行排序时发生溢出，请增大 **sheapthres** 数据库管理器配置参数的值。

- 为关系索引创建不同的表空间

  您可以在速度较快的物理设备上创建索引表空间，此外，也可以将索引表空间指定到另一个缓冲池以避免索引页与表数据页发生竞争，从而延长索引页在缓冲区中的停留时间。

  如果对索引使用另一个表空间，那么可以针对索引优化该表空间的配置。由于索引通常比表小，且分布在更少的容器中，因此索引经常具有较小的扩展数据块大小。查询优化器在选择访问方案时，它将考虑包含表空间的设备的速度。

- 确保高度集群

  如果 SQL 语句要求对结果进行排序（例如，它包含 ORDER BY、GROUP BY 或 DISTINCT 子句），那么在下列情况下，优化器可能不会选择索引:

  – 索引集群程度较低。有关特定索引中的集群程度的信息，请查询 SYSCAT.INDEXES 目录视图的 CLUSTERRATIO 和 CLUSTERFACTOR 列。
  – 由于表很小，因此扫描该表并在内存中对结果集进行排序的成本更低。
  – 有多个索引可用于访问该表。

  集群索引将尝试维护数据的特定顺序，从而改善 RUNSTATS 实用程序所收集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。创建集群索引之后，请执行脱机表重组操作。通常，只能根据一个索引对表进行集群。请在构建集群索引之后构建其他索引。

  表的 PCTFREE 值确定在页面中要为将来插入数据而留下的可用空间量，以便能够适当地对插入的数据进行集群。如果未对表指定 PCTFREE 值，那么重组操作将消除所有额外空间。

  除范围集群表的情况以外，在更新操作期间不会对数据集群进行维护。即，如果更新一条记录，导致集群索引中它的键值发生更改，那么不一定会将该记录移至新页以维护集群顺序。要维护集群，请删除该记录，然后插入该记录的更新版本，而不是使用更新操作。

- 使表和索引统计信息保持最新

  在创建新的关系索引之后，请执行 RUNSTATS 实用程序以收集索引统计信息。这些统计信息可以帮助优化器确定使用索引是否能够提高数据访问性能。

- 启用联机索引整理碎片功能

  如果将关系索引的 MINPCTUSED 设置为大于零的值，那么将启用联机索引整理碎片功能。当页面中的可用空间量低于指定的 MINPCTUSED 值时，联机索引整理碎片功能将允许通过合并索引叶子页来压缩索引。

- 必要时重组关系索引

  为了借助索引最大程度地提高性能，请考虑定期重组索引，这是因为对表执行的更新会导致索引页预取效率降低。

  要重组索引，请将其删除并重新创建，或者使用 REORG 实用程序。

  为了避免需要频繁地执行重组，请在 CREATE INDEX 语句中指定适当的 PCTFREE 值，以便创建每个索引叶子页时在该页中保留足够的可用空间。在将来的活动期

间，将记录插入到索引时导致索引页分割的可能性较小，从而提高索引页预取效率（索引页分割将使索引页的临近性下降）。重组关系索引时，将根据您创建该索引时指定的 PCTFREE 值来保留空间。

- 分析关于关系索引使用情况的说明信息

  请对最常用的查询定期发出 EXPLAIN 语句，并验证每个关系索引是否至少使用了一次。如果某个索引未被任何查询使用，那么请考虑删除该索引。

  说明信息还使您能够确定，正在扫描的大型表是否作为嵌套循环连接的内表进行处理。如果是这样，那么连接谓词列的索引或者丢失，或者被认为应用于该连接谓词的效率不高。

- 将大小变化范围很大的表声明为“易失”

  *易失表*是指基数在运行时变化很大的表。对于这种表，优化器可能会生成执行表扫描而不是索引扫描的访问方案。

  请使用带有 VOLATILE 子句的 ALTER TABLE 语句将此类表声明为“易失”。对于这样的表，优化器在下列情况下将执行索引扫描代替表扫描，而不考虑统计信息:

  – 所引用的所有列都是索引的组成部分
  – 该索引可以在索引扫描期间应用谓词

  对于类型表而言，只有类型表层次结构的根表才支持 ALTER TABLE...VOLATILE 语句。

分区与集群

分区表的索引行为

分区表的索引的工作方式与非分区表的索引类似，但是，它们使用另一种存储模型进行存储，这取决于它们是分区索引还是非分区索引。

虽然常规非分区表的索引全都驻留在共享的索引对象中，但分区表的*非分区索引*将在单一表空间中它自己的索引对象中进行创建，即使数据分区跨多个表空间亦如此。数据库管理的空间（DMS）和系统管理的空间（SMS）表空间都支持使用不同于表数据所在位置的位置中的索引。可以将每个非分区索引放入它自己的表空间，其中包括大型表空间。每个索引表空间都必须使用与数据分区相同的存储机制，即 DMS 或 SMS。大型表空间中的索引可以包含多达 2^{29} 页。所有表空间都必须在同一个数据库分区组中。

*分区索引*使用了索引组织方案，即，索引数据根据表的分区方案划分到多个*索引分区*中。每个索引分区都只引用相应数据分区中的表行。给定数据分区的所有索引分区都驻留在同一个索引对象中。

从 DB2 版本 9.7 修订包 1 开始，用户根据分区表中 XML 列的 XML 数据创建的索引可以是分区索引或非分区索引。缺省情况下为分区索引。系统生成的 XML 区域索引始终为分区索引，而系统生成的列路径索引始终为非分区索引。在 DB2 V9.7 中，基于 XML 数据的索引为非分区索引。

非分区索引的优势包括:

- 能够为每个索引定义不同的表空间特征（例如，不同的页大小可能有助于确保更好地利用空间）
- 可以相互独立地对各个索引进行重组
- 能够提高删除索引操作的性能
- 减少 I/O 争用，这有助于更高效地对索引数据进行并发访问
- 删除各个索引时，空间将立即可供系统使用，而无需进行索引重组

分区索引的优势包括:

- 能够提高数据滚入和滚出性能
- 由于索引进行分区，因此能够减少对索引页的争用
- 每个索引分区均采用索引 B 树结构，这有如下作用:
  - 提高插入、更新、删除和扫描性能，这是因为，索引分区的 B 树所包含的层数通常少于引用表中所有数据的索引
  - 在采用分区消除功能期间提高扫描性能和并行性；尽管分区消除功能既可用于分区索引扫描也可用于非分区索引扫描，但用于分区索引扫描却更为有效，这是因为，每个索引分区都只包含相应数据分区的键。这样，所扫描的键数和索引页数要少于对非分区索引执行的类似查询。

虽然非分区索引始终保留索引列的顺序，但分区索引在某些情况下可能会在各分区之间丢失一些顺序；例如，如果分区列与索引列不匹配，并且将访问多个分区。

在联机索引创建期间，允许对表进行并发读写访问。在构建此类索引之后，在索引构建期间对该表所作的更改将应用于新索引。对该表所作的写访问将被阻塞，直到索引创建完成并且事务落实为止。对于分区索引而言，仅当应用创建索引分区期间对数据分区所作的更改时，才会停顿每个数据分区以便进行只读访问。

当您使用 ALTER TABLE...ATTACH PARTITION 语句滚入数据时，分区索引支持变得特别有用。如果存在非分区索引（不包括 XML 列路径索引，如果表包含 XML 数据的话），请在连接分区之后发出 SET INTEGRITY 语句。这对于非分区索引维护、范围验证、约束检查和具体化查询表（MQT）维护而言十分有必要。非分区索引的维护工作可能相当耗时并需要大量日志空间。请使用分区索引来避免此维护成本。

第 65 页的图 12 显示了分区表的两个非分区索引，这两个索引驻留在不同的表空间中。

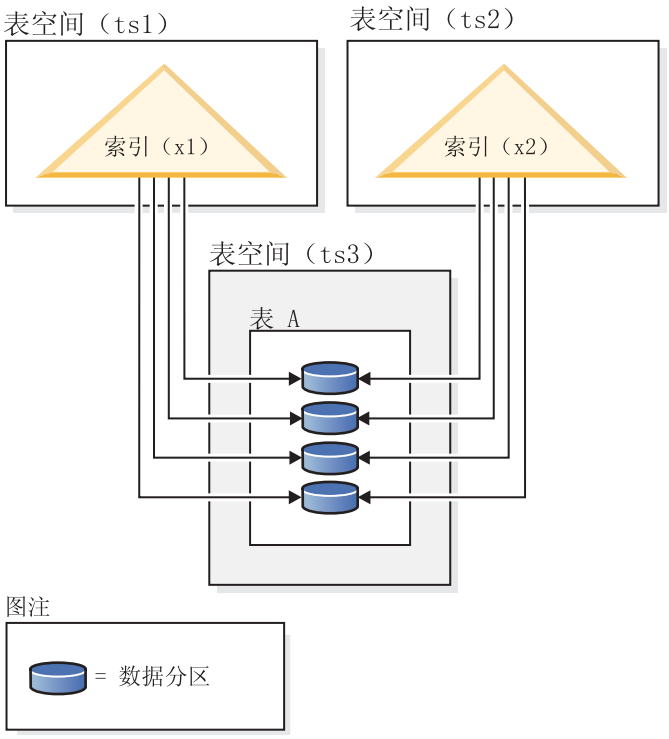


图 12. 分区表的非分区索引

第 66 页的图 13 显示了分区表的分区索引，此索引跨两个数据库分区并驻留在单一表空间中。

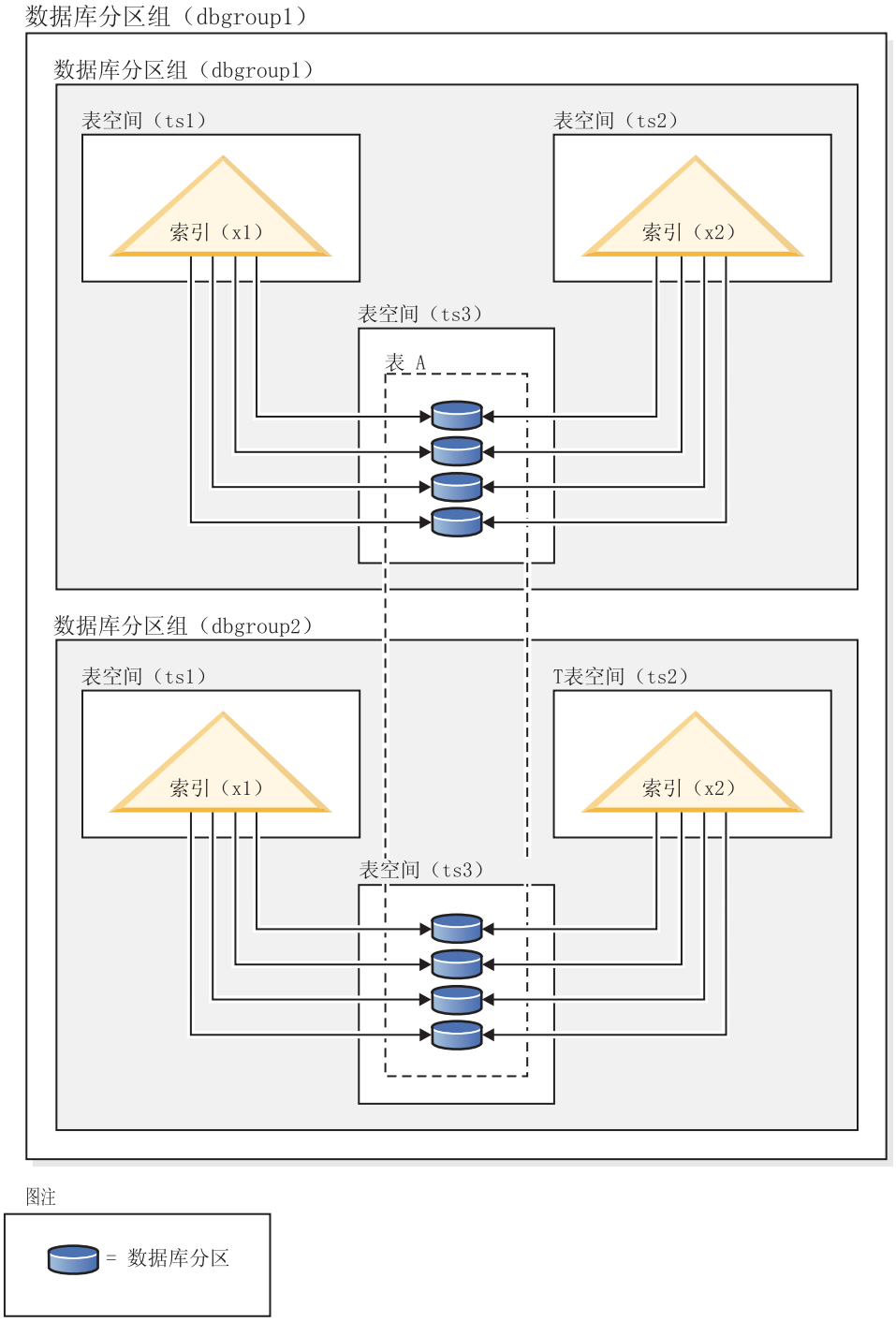


图 13. 分布式分区表的非分区索引

第 67 页的图 14 显示了分区表的混合分区索引和非分区索引。

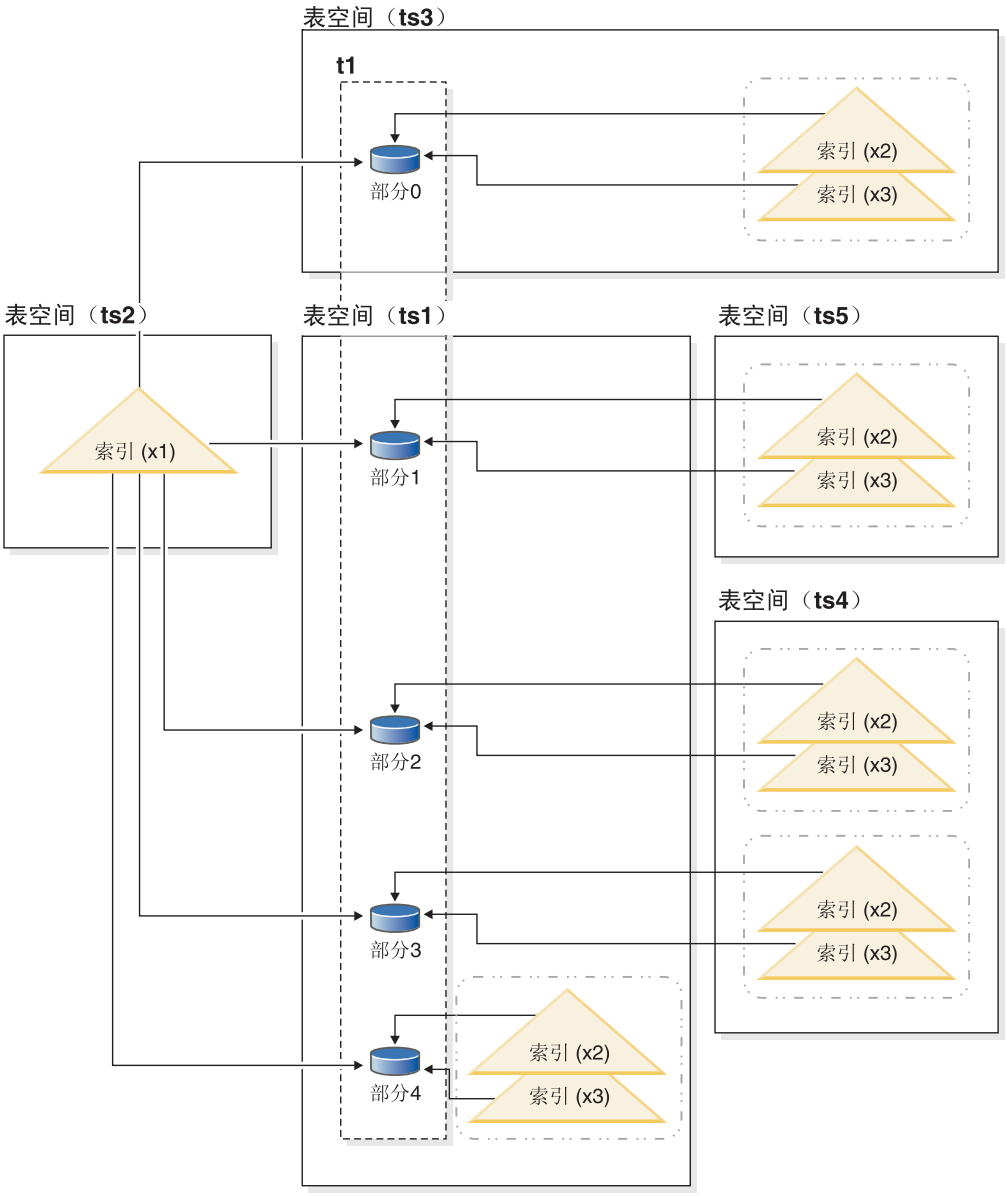


图 14. 分区表的分区索引和非分区索引

非分区索引 X1 引用所有数据分区中的行。相反，分区索引 X2 和 X3 只引用与其相关联的数据分区中的行。表空间 TS3 还显示了一些索引分区，这些索引分区共享与其相关联的数据分区的表空间。对于分区索引而言，这是缺省情况。

您可以覆盖非分区索引和分区索引的缺省位置，尽管为这两种索引执行此操作的方法有所不同。对于非分区索引，可以在创建该索引时指定表空间；对于分区索引，您需要在创建该表时确定用于存储索引分区的表空间。

非分区索引

要覆盖非分区索引的索引位置，请使用 CREATE INDEX 语句的 IN 子句，这将使您能够为索引指定另一个表空间位置。根据需要，可以将不同的索引放入不同的表空间。如果创建分区表时未指定它的非分区索引的放置位置，并且使用未指定表空间的 CREATE INDEX 语句来创建索引，那么将在已连接的第一个数据分区或可视数据分区的表空间中创建该索引。按顺序对下面三种可能情

况进行评估（从情况 1 开始），以确定创建索引的位置。找到匹配的情况时，此项用于确定索引的表空间位置的评估即停止。

情况 1:
如果在 CREATE INDEX...IN *tbspace* 语句中指定了索引表空间，那么将指定的表空间用于此索引。
情况 2:
如果在 CREATE TABLE...INDEX IN *tbspace* 语句中指定了索引表空间，那么将指定的表空间用于此索引。
情况 3:
未指定任何表空间时，选择已连接的第一个数据分区或可视数据分区所使用的表空间。

分区索引

缺省情况下，索引分区将被放入它们所引用的数据分区所在的表空间。要覆盖这种缺省行为，必须对您使用 CREATE TABLE 语句定义的每个数据分区使用 INDEX IN 子句。换而言之，如果您计划对分区表使用分区索引，那么必须在创建该表时预测索引分区的存储位置。如果您尝试在创建分区索引时使用 INDEX IN 子句，那么将接收到错误消息。

示例 1: 给定分区表 SALES (a int, b int, c int)，创建唯一索引 A_IDX。

```
create unique index a_idx on sales (a)
```

由于表 SALES 是分区表，因此索引 a_idx 也将被创建为分区索引。

示例 2: 创建索引 B_IDX。

```
create index b_idx on sales (b)
```

示例 3: 覆盖分区索引中索引分区的缺省位置，对您创建分区表时定义的每个分区使用 INDEX IN 子句。在以下示例中，将在表空间 TS3 中创建表 Z 的索引。

```
create table z (a int, b int)
      partition by range (a) (starting from (1)
      ending at (100)  index in ts3)

   create index c_idx on z (a) partitioned
```

分区表的非分区索引的集群

对分区表使用集群索引可获得与对常规表使用集群索引相同的好处。但是，在选择集群索引时，必须谨慎地对待表分区键定义。

您可以使用任何集群键对分区表创建集群索引。数据库服务器将尝试使用集群索引以本地方式对每个数据分区中的数据进行集群。在集群插入操作期间，将执行索引查找操作以查找适合的记录标识（RID）。将从此 RID 开始在表中寻找空间以插入记录。为了实现性能优良的最佳集群，索引列与表分区键列之间应该存在关联。确保这种关联的一种方法是将表分区键列置于索引列之前，如以下示例所示:

```
partition by range (month, region)
create index...(month, region, department) cluster
```

虽然数据库服务器不强制此关联，但还是希望索引中的所有键按分区标识进行聚集，以实现优良的集群。例如，假定一个表按 QUARTER 进行分区，并且对 DATE 定义了集群索引。在 QUARTER 与 DATE 之间存在关系，由于任何数据分区的所有键在该索引中都聚集在一起，因此能够实现性能良好的最佳数据集群。第 69 页的图 15 表明，仅

当集群与表分区键相关时，才能实现最佳的扫描性能。

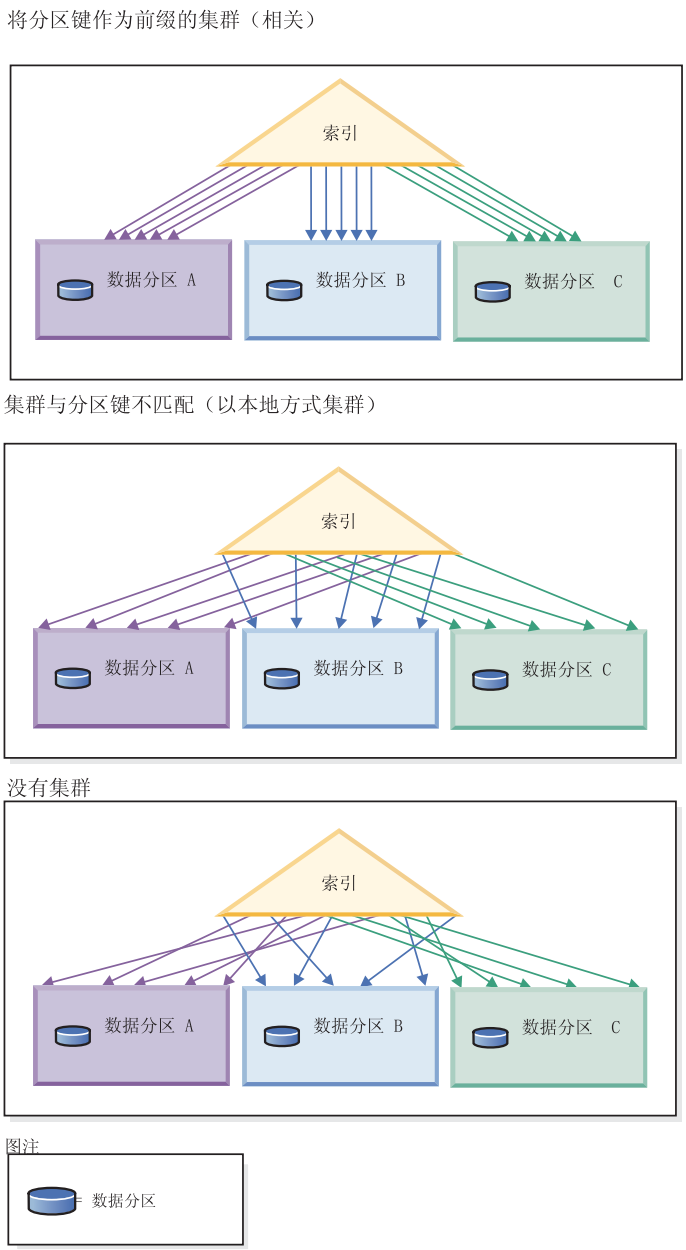


图 15. 集群索引可能对分区表产生的影响。

集群的好处包括:

- 在每个数据分区中，各个行按键顺序排列。
- 集群索引能够提高按键顺序遍历表的扫描的性能，这是因为扫描者将先访存第一页的第一行，接着访存该页的每一行，然后再移至下一页。这意味着，在任何给定时

间，缓冲池都只需要包含表的一页。相反，如果未对表进行集群，那么有可能访存不同的页中的行。除非缓冲池能够容纳整个表，否则大多数页都可能被访存多次，从而导致扫描速度显著下降。

如果集群键与表分区键不相关，但数据以局部方式进行集群，那么当缓冲池有足够空间来容纳每个数据分区的一页时，仍然可以获得集群索引的全部好处。这是因为，从特定数据分区中访存的每一行都在先前从该分区中访存的行附近（参见第 69 页的图 15 中的第二个示例）。

联合数据库

影响联合数据库的服务器选项

联合数据库系统由 DB2 数据服务器（联合数据库）以及一个或多个数据源组成。您在发出 CREATE SERVER 语句时向联合数据库标识数据源。您还可以指定服务器选项，以便优化和控制联合系统操作的各个方面。

必须安装分布式连接安装选项，并将数据库管理器配置参数 **federated** 设置为 YES，然后才能创建服务器并指定服务器选项。以后，要更改服务器选项，请使用 ALTER SERVER 语句。

您在 CREATE SERVER 语句中指定的服务器选项值将影响查询下推分析、全局优化以及联合数据库操作的其他方面。例如，可以指定性能统计信息作为服务器选项值。*cpu_ratio* 选项指定数据源处理器和联合服务器处理器的相对速度，*io_ratio* 选项指定数据源和联合服务器上的数据 I/O 的相对速率。

服务器选项值将被写入系统目录（SYSCAT.SERVEROPTIONS），优化器在为数据源开发访问方案时，将使用此信息。如果统计信息发生更改（例如，升级了数据源处理器），那么请使用 ALTER SERVER 语句来更新该目录以使其包含新值。

资源利用率

内存分配

内存分配和释放在各个时间进行。可以在发生特定事件（例如应用程序建立连接）时将内存分配给特定内存区，也可以为了响应配置更改而重新分配内存。

第 71 页的图 16 显示了数据库管理器为不同用途分配的各个内存区以及允许您控制这些内存区的大小的配置参数。请注意，在分区数据库环境中，每个数据库分区都将设置自己的数据库管理器共享内存。

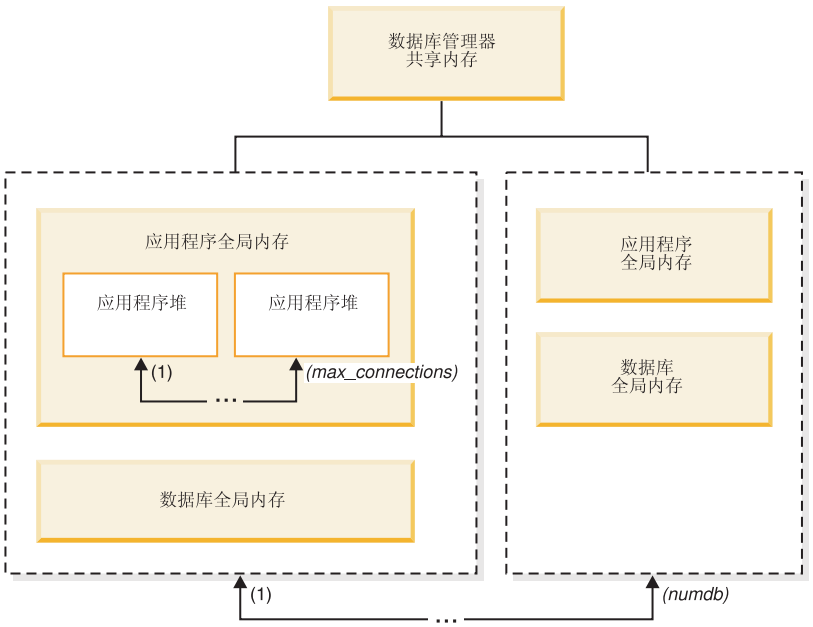


图 16. 数据库管理器分配的内存的类型

每当发生下列其中一个事件时，数据库管理器都将分配内存:

数据库管理器启动时（db2start）

数据库管理器共享内存（也称为*实例共享内存*）在数据库管理器停止（db2stop）前将保持处于已分配状态。此区域包含数据库管理器在管理通过所有数据库连接进行的活动时所需的信息。DB2 将自动控制数据库管理器共享内存的大小。

第一次激活数据库或者连接到数据库时

所有与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** 数据库配置参数指定。缺省情况下，此参数设置为 automatic，从而允许 DB2 计算为数据库分配的初始内存量以及在运行时根据数据库的需要自动配置数据库内存大小。

可以对下列内存区进行动态调整:

- 缓冲池（使用 ALTER BUFFERPOOL 语句）
- 数据库堆（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
- 锁定列表

此外，还可以动态地更新 **sortheap**、**sheapthres_shr** 和 **sheapthres** 配置参数。唯一的限制是，不能动态地将 **sheapthres** 由 0 更改为大于 0 的值，反之亦然。

缺省情况下，将执行共享排序操作，排序内存使用者在任何一个时间可以使用的数据库共享内存量由 **sheapthres_shr** 数据库配置参数值确定。仅当分区内并行性、数据库分区和连接集中器都处于禁用状态，并且 **sheapthres** 数据库管理器配置参数设置为非零值时，才能执行专用排序操作。

应用程序连接至数据库时

每个应用程序都有自己的*应用程序堆*，这是*应用程序全局内存*的组成部分。您可以使用 **applheapsz** 数据库配置参数来限制任何一个应用程序可以分配的内存量，也可以使用 **appl_memory** 数据库配置参数来限制应用程序内存总耗用量。

创建代理程序时

当分区数据库环境中出现连接请求或新的 SQL 请求时，系统将指定代理程序并为其分配*代理程序专用内存*。代理程序专用内存包含仅供此特定代理程序使用的内存。如果已启用专用排序操作，那么将从代理程序专用内存中分配专用排序堆。

下列配置参数用于限制为每种类型的内存区分配的内存量。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

此数据库管理器配置参数指定各个应用程序可以使用的并发活动数据库的最大数目。因为每个数据库都有自己的全局内存区，所以增大此参数的值将增加可以分配的内存量。

maxappls

此数据库配置参数指定可以同时连接到特定数据库的应用程序的最大数目。此参数的值将影响可以为该数据库分配的代理程序专用内存量和应用程序全局内存量。

max_connections

此数据库管理器配置参数用于限制任何时候可以访问数据服务器的数据库连接或实例连接的数目。

max_coordagents

此数据库管理器配置参数用于限制，一个实例的所有活动数据库中可以同时存在的数据库管理器协调代理程序的数目（在分区数据库环境中，将对每个数据库分区实施此限制）。与 **maxappls** 和 **max_connections** 相配合，此参数将限制为代理程序专用内存和应用程序全局内存分配的内存量。

db2mtrk 命令所调用的内存跟踪程序使您能够查看实例中的当前内存分配量。您还可以使用 ADMIN_GET_DBP_MEM_USAGE 表函数来确定整个实例或单一数据库分区的内存总耗用量。GET SNAPSHOT 命令使您能够在实例、数据库或应用程序级别检查当前内存使用情况。

数据库管理器共享内存

数据库管理器共享内存组织成若干个不同的内存区。配置参数使您能够控制这些区域的大小。

第 73 页的图 17 显示了数据库管理器共享内存的分配方式。

图 17. 数据库管理器的内存使用方式

监视器堆

此内存区用于存储数据库系统监视器数据。此内存区的大小由 **mon_heap_sz** 数据库管理器配置参数确定。

审计缓冲区

此内存区用于数据库审计活动。此缓冲区的大小由 **audit_buf_sz** 数据库管理器配置参数确定。

快速通信管理器（FCM）缓冲池

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，**fcm_num_buffers** 的值较大时尤其如此。所需的 FCM 内存将从 FCM 缓冲池中进行分配。

数据库全局内存

数据库全局内存受缓冲池大小以及下列数据库配置参数影响:

- **catalogcache_sz**
- **database_memory**
- **dbheap**
- **locklist**
- **pckcachesz**
- **sheapthres_shr**
- **util_heap_sz**

应用程序全局内存

应用程序全局内存可由 **appl_memory** 配置参数控制。可以使用下列数据库配置参数来限制任何一个应用程序可耗用的内存量:

- **applheapsz**
- **stat_heap_sz**
- **stmtheap**

代理程序专用内存

每个代理程序都需要自己的专用内存区。数据服务器将根据需要以及已配置的内存资源来创建任意数目的代理程序。您可以使用 **max_coordagents** 数据库管理器配置参数来控制协调代理程序的最大数目。每个代理程序的专用内存区最大大小由下列配置参数的值确定:

- **agent_stack_sz**
- **sheapthres** 和 **sortheap**

代理程序/应用程序共享内存

本地客户机的代理程序/应用程序共享内存段的总数受下面这两个值的较小者限制:

- 所有活动数据库的 **maxappls** 数据库配置参数值的总和
- **max_coordagents** 数据库配置参数的值

注: 在启用了引擎集中功能的配置中（**max_connections** > **max_coordagents**），应用程序内存耗用量受 **max_connections** 限制。

代理程序/应用程序共享内存还受下列数据库配置参数影响:

- **aslheapsz**
- **rqrioblk**

FCM 缓冲池和内存需求

在分区数据库系统中，快速通信管理器（FCM）缓冲区共享内存将从数据库管理器共享内存中进行分配。

图 18 对此作了说明。

图 18. 使用多逻辑分区时的 FCM 缓冲池

每个数据库分区的 FCM 缓冲区数由 **fcm_num_buffers** 数据库管理器配置参数控制。缺省情况下，此参数设置为 automatic。要手动调整此参数，请使用 **buff_free** 和 **buff_free_bottom** 系统监视元素中的数据。

每个数据库分区的 FCM 通道数由 **fcm_num_channels** 数据库管理器配置参数控制。缺省情况下，此参数设置为 automatic。要手动调整此参数，请使用 **ch_free** 和 **ch_free_bottom** 系统监视元素中的数据。

DB2 数据库管理器能够通过根据需要分配更多 FCM 缓冲区和通道来自动管理 FCM 内存资源。这将提高性能并防止“FCM 资源不足”运行时错误。在 Linux 操作系统上，数据库管理器可以为 FCM 缓冲区和通道预先分配更大量的系统内存，大小可达最大缺省容量 2 GB。仅当需要附加的 FCM 缓冲区或通道时，内存空间才受影响。要启用此行为，请将 **DB2_FCM_SETTINGS** 注册表变量的 FCM_MAXIMIZE_SET_SIZE 选项设置为 YES（或 TRUE）。缺省值是 YES。

有关调整将会影响内存使用情况的参数的准则

以手动方式调整内存时（即，不使用自调整内存管理器时），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，同时更改内存参数的值，直至找到性能开始下降的那一点为止。这个点指示进一步分配内存并不会提高应用程序的性能。

多个参数的内存分配上限可能超出现有硬件和操作系统的容量范围。这些限制是为将来的增长作好准备。您最好不要将内存参数设置为它们的最大值，除非可以证明那些值恰当。这甚至适用于带有大量可用内存的系统。此建议的意图是，避免数据库管理器迅速耗尽系统上的所有可用内存。此外，管理大量内存也会引起附加的开销。

对于大多数配置参数而言，仅在需要时才会落实内存，并且参数设置确定特定内存堆的最大大小。但是，对于缓冲池和下列配置参数而言，将分配所有指定的内存:

- **aslheapsz**
- **fcm_num_buffers**
- **fcm_num_channels**
- **locklist**

某些操作系统在进程分配内存时（而不是在需要对该内存进行页面调度以将其内容输出到交换空间时）分配交换空间。对于这些系统，通常建议您提供容量至少为系统总内存量两倍的调页空间。

要了解参数的有效范围，请参阅每个参数的详细信息。

自调整内存功能概述

自调整内存功能通过自动设置内存配置参数值以及调整缓冲池大小来简化内存配置任务。启用此功能之后，内存调整器将在下列内存使用者之间动态分配可用的内存资源: 缓冲池、锁定内存、程序包高速缓存和排序内存。

要启用自调整内存功能，请使用 **self_tuning_mem** 数据库配置参数。

可以自动调整下列与内存相关的数据库配置参数:

- database_memory - 数据库共享内存大小
- locklist - 锁定列表的最大存储量
- maxlocks - 锁定升级前锁定列表的最大百分比
- pckcachesz - 程序包高速缓存大小
- sheapthres_shr - 共享排序的排序堆阈值
- sortheap - 排序堆大小

自调整内存功能

从 DB2 版本 9 开始，内存调整功能通过自动设置一些内存配置参数的值来简化内存配置任务。启用此功能之后，内存调整器将在下列内存使用者之间动态分配可用的内存资源: 缓冲池、锁定内存、程序包高速缓存和排序内存。

调整器在 **database_memory** 配置参数所定义的内存限制范围内工作。此参数的值也可以自动调整。启用自调整功能（将 **database_memory** 的值设置为 AUTOMATIC）之后，调整器将确定数据库的整体内存需求并根据当前数据库需求来增加或减少分配给数据库共享内存的内存量。例如，如果当前数据库需求很高，并且系统上有足够的可用内存，那么将为数据库共享内存分配较多的内存。如果数据库内存需求下降，或者系统上的可用内存量变得过低，那么将释放一些数据库共享内存。

如果 **database_memory** 配置参数未设置为 AUTOMATIC，那么数据库将使用您对此参数指定的内存量，从而根据需要在内存使用者之间分配内存。您可以通过两种方法来指定此内存量: 将 **database_memory** 设置为某个数值或者将其设置为 COMPUTED。在后一种情况下，总内存量基于数据库启动时的数据库内存堆初始值之和。

您还可以对内存使用者启用自调整功能，如下所示:

- 对于缓冲池，使用 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 语句（指定 AUTOMATIC 关键字）
- 对于锁定内存，使用 **locklist** 或 **maxlocks** 数据库配置参数（指定 AUTOMATIC 值）
- 对于程序包高速缓存，使用 **pckcachesz** 数据库配置参数（指定 AUTOMATIC 值）
- 对于排序内存，使用 **sheapthres_shr** 或 **sortheap** 数据库配置参数（指定 AUTOMATIC 值）

自调整操作所作的更改将记录在 stmmlog 子目录中的内存调整日志文件中。这些日志文件包含每个内存使用者在特定调整时间间隔内的资源需求摘要，这些时间间隔由日志条目中的时间戳记确定。

如果可用内存量不多，那么自调整功能的性能增益有限。由于调整决策基于数据库工作负载，因此内存需要快速变化的工作负载可能会限制自调整内存管理器（STMM）的效能。如果工作负载的内存特征不断变化，那么 STMM 将以较低的频率在多变的目标条件下进行调整。在这种情况下，STMM 将无法实现绝对汇合，而是尝试维护针对当前工作负载进行调整的内存配置。

启用自调整内存功能

自调整内存功能通过自动设置内存配置参数值以及调整缓冲池大小来简化内存配置任务。

关于此任务

启用此功能之后，内存调整器将在多个内存使用者（其中包括缓冲池、锁定内存、程序包高速缓存和排序内存）之间动态分配可用的内存资源。

过程

1. 要对数据库启用自调整内存功能，请使用 UPDATE DATABASE CONFIGURATION 命令或 db2CfgSet API 将 **self_tuning_mem** 数据库配置参数设置为 ON。
2. 要对内存配置参数所控制的内存区启用自调整功能，请使用 UPDATE DATABASE CONFIGURATION 命令或 db2CfgSet API 将相关配置参数设置为 AUTOMATIC。
3. 要对缓冲池启用自调整功能，请使用 CREATE BUFFERPOOL 语句或 ALTER BUFFERPOOL 语句将缓冲池大小设置为 AUTOMATIC。在分区数据库环境中，缓冲池在 SYSCAT.BUFFERPOOLDBPARTITIONS 中不应该有任何条目。

结果

注:

1. 由于在不同的内存使用者之间分配自调整内存，因此在任意给定时间，必须至少对两个内存区（例如锁定内存和数据库共享内存）同时启用自调整功能。当满足下列其中一个条件时，内存调整器将主动调整系统上的内存（**self_tuning_mem** 数据库配置参数的值为 ON）：
   - 一个配置参数或缓冲池大小设置为 AUTOMATIC，并且 **database_memory** 数据库配置参数设置为数字值或者 AUTOMATIC
   - **locklist**、**sheapthres_shr**、**pckcachesz** 或缓冲池大小中的任意两个设置为 AUTOMATIC

- **sortheap** 数据库配置参数设置为 AUTOMATIC

2. **locklist** 数据库配置参数的值将与 **maxlocks** 数据库配置参数一起进行调整。对 **locklist** 参数禁用自调整功能将自动地对 **maxlocks** 参数禁用自调整功能，而对 **locklist** 参数启用自调整功能将自动地对 **maxlocks** 参数启用自调整功能。
3. 仅当数据库管理器配置参数 **sheapthres** 设置为 0 时，才允许自动调整 **sortheap** 或 **sheapthres_shr** 数据库配置参数。
4. **sortheap** 的值将与 **sheapthres_shr** 一起进行调整。对 **sortheap** 参数禁用自调整功能将自动地对 **sheapthres_shr** 参数禁用自调整功能，而对 **sheapthres_shr** 参数启用自调整功能将自动地对 **sortheap** 参数启用自调整功能。
5. 自调整内存功能只能在高可用性灾难恢复（HADR）主服务器上运行。在 HARD 系统上激活自调整内存功能后，永远不会在辅助服务器上运行此功能，并且，只有在正确设置配置的情况下，此功能才会在主服务器上运行。如果切换 HADR 数据库角色，那么自调整内存操作也将进行切换，从而在新的主服务器上运行。在主数据库启动之后，或者在备用数据库通过接管操作转换为主数据库之后，自调整内存管理器（STMM）引擎可分派单元（EDU）可能直到第一个客户机建立连接后才会启动。

禁用自调整内存功能

可以对整个数据库或者一个或多个配置参数或缓冲池禁用自调整内存功能。

即使对整个数据库禁用自调整内存功能，设置为 AUTOMATIC 的内存配置参数和缓冲池也仍然支持自动调整；但是，内存区将保持当前大小不变。

1. 要对数据库禁用自调整内存功能，请使用 UPDATE DATABASE CONFIGURATION 命令或 db2CfgSet API 将 **self_tuning_mem** 数据库配置参数设置为 OFF。
2. 要对内存配置参数所控制的内存区禁用自调整功能，请使用 UPDATE DATABASE CONFIGURATION 命令或 db2CfgSet API 将相关配置参数设置为 MANUAL 或指定数字参数值。
3. 要对缓冲池禁用自调整功能，请使用 ALTER BUFFERPOOL 语句将缓冲池大小设置为特定的值。

注:

- 在某些情况下，要对一个内存配置参数启用自调整功能，还必须同时对另一个相关的内存配置参数启用此功能。例如，这意味着对 **locklist** 或 **sortheap** 数据库配置参数禁用自调整内存功能时，还将分别对 **maxlocks** 或 **sheapthres_shr** 数据库配置参数禁用自调整内存功能。

确定已启用自调整功能的内存使用者

您可以查看由配置参数控制或者应用于缓冲池的自调整内存功能设置。

- 要从命令行查看配置参数的设置，请使用 GET DATABASE CONFIGURATION 命令并指定 SHOW DETAIL 选项。在输出中，可以启用自调整功能的内存使用者将分组到一起，如下所示:

```
描述                                 参数                  当前值                 延迟的值
-------------------------------------------------------------------------------------------
自调整内存功能                       (SELF_TUNING_MEM) =   ON（活动）             ON
数据库共享内存大小（4KB）            (DATABASE_MEMORY) =   AUTOMATIC(37200)       AUTOMATIC(37200)
最大锁定列表存储器（4KB）            (LOCKLIST) =          AUTOMATIC(7456)        AUTOMATIC(7456)
每个应用程序的锁定列表百分比         (MAXLOCKS) =          AUTOMATIC(98)          AUTOMATIC(98)
程序包高速缓存大小（4KB）            (PCKCACHESZ) =        AUTOMATIC(5600)        AUTOMATIC(5600)
共享排序的排序堆阈值（4KB）          (SHEAPTHRES_SHR) =    AUTOMATIC(5000)        AUTOMATIC(5000)
排序列表堆（4KB）                    (SORTHEAP) =          AUTOMATIC(256)         AUTOMATIC(256)
```

- 也可以使用 db2CfgGet API 来确定是否已启用调整功能。将返回下列值:

  ```
  SQLF_OFF                  0
  SQLF_ON_ACTIVE            2
  SQLF_ON_INACTIVE          3
  ```

  SQLF_ON_ACTIVE 表明自调整功能已启用并处于活动状态，而 SQLF_ON_INACTIVE 表明自调整功能已启用但当前处于不活动状态。

要查看缓冲池的自调整设置，请使用下列其中一种方法。

- 要从命令行检索已启用自调整功能的缓冲池列表，请使用以下查询:

  ```
  SELECT BPNAME, NPAGES FROM SYSCAT.BUFFERPOOLS
  ```

  对缓冲池启用自调整功能之后，该特定缓冲池的 SYSCAT.BUFFERPOOLS 视图中的 NPAGES 字段将设置为 -2。当自调整功能处于禁用状态时，NPAGES 字段将设置为缓冲池的当前大小。
- 要确定已启用自调整功能的缓冲池的当前大小，请使用 GET SNAPSHOT 命令并检查缓冲池的当前大小（**bp_cur_buffsz** 监视元素的值）:

  ```
  GET SNAPSHOT FOR BUFFERPOOLS ON database-alias
  ```

  对特定数据库分区指定缓冲池大小的 ALTER BUFFERPOOL 语句将在 SYSCAT.BUFFERPOOLDBPARTITIONS 目录视图中为该缓冲池创建例外条目或更新现有条目。如果某个缓冲池的例外条目已存在，并且缺省缓冲池大小设置为 AUTOMATIC，那么该缓冲池将不会参与自调整操作。

注意，内存调整器的反应受调整内存使用者的内存使用量所需时间的限制，这一点十分重要。例如，减小缓冲池大小的过程可能非常长，因此，为排序内存调整缓冲池内存大小所产生的性能增益可能不会立即体现。

分区数据库环境中的自调整内存功能

在分区数据库环境中使用自调整内存功能时，有一些因素决定该功能是否能适当地调整系统。

对分区数据库启用自调整内存功能时，会将一个数据库分区指定为调整分区，所有内存调整决定都根据该数据库分区的内存和工作负载特征作出。在该分区中作出调整决策之后，会将内存调整分发到其他数据库分区，以确保所有数据库分区都维护类似的配置。

单调整分区模型假定，仅当所有数据库分区具有类似内存需求时，才会使用该功能。在确定是否对分区数据库启用自调整内存功能时，请使用下列准则。

建议对分区数据库使用自调整内存功能的情况

当所有数据库分区都具有类似内存需求并且正在类似硬件上运行时，可以不进行任何修改就启用自调整内存功能。这些类型的环境共享下列特征:

- 所有数据库分区都在完全相同的硬件上运行，并且多个逻辑数据库分区均匀地分布在多个物理数据库分区中
- 数据分布情况最佳或者接近最佳
- 工作负载均匀地分布在各个数据库分区中，这意味着，各个数据库分区中一个或多个堆的内存需求均相同

在这种环境中，如果所有数据库分区的配置相同，那么自调整内存功能将正确地配置系统。

建议对分区数据库使用自调整内存功能并进行限定的情况

在环境中的大部分数据库分区具有类似内存需求并且正在类似硬件上运行的情况下，可以使用自调整内存功能，但进行初始配置时要小心。这些系统可能有一组数据库分区用于数据，并且有一组少得多的协调程序分区和目录分区。在这些环境中，将协调程序分区和目录分区配置为与包含数据的数据库分区不同可能会有好处。

应该对所有包含数据的数据库分区启用自调整内存功能，并且应该将其中的一个数据库分区指定为调整分区。由于协调程序分区和目录分区的配置可能不同，因此应对那些分区禁用自调整内存功能。要对协调程序分区和目录分区禁用自调整内存功能，请对这些分区将 **self_tuning_mem** 数据库配置参数设置为 OFF。

建议不要对分区数据库使用自调整内存功能的情况

如果各个数据库分区的内存需求有所不同，或者不同的数据库分区正在极不相同的硬件上运行，那么最好禁用自调整内存功能。要禁用此功能，请对所有分区将 **self_tuning_mem** 数据库配置参数设置为 OFF。

比较不同数据库分区的内存需求

确定不同数据库分区的内存需求是否非常相近的最佳方法是查看快照监视器。如果下列快照元素在所有数据库分区中都相近（差别不超过 20%），那么可以认为这些数据库分区的内存需求极为相近。

通过发出以下命令来收集下列数据：`get snapshot for database on <dbname>`

```
当前挂起的锁定数                        = 0
锁定等待数                              = 0
数据库等待锁定的时间（毫秒）            = 0
正在使用的锁定列表内存（以字节计）      = 4968
锁定升级次数                            = 0
互斥锁定升级次数                        = 0

已分配的共享排序堆总数                  = 0
共享排序堆高水位标记                    = 0
超出阈值后的排序次数（共享内存）        = 0
排序溢出数                              = 0

程序包高速缓存查询数                    = 13
程序包高速缓存插入数                    = 1
程序包高速缓存溢出数                    = 0
程序包高速缓存高水位标记（以字节计）    = 655360

散列连接数                              = 0
散列循环数                              = 0
散列连接溢出数                          = 0
小散列连接溢出数                        = 0
后阈值散列连接数（共享内存）            = 0

OLAP 功能数目                           = 0
OLAP 功能溢出数目                       = 0
活动 OLAP 功能数                        = 0
```

通过发出以下命令来收集下列数据：`get snapshot for bufferpools on <dbname>`

```
缓冲池数据逻辑读取次数                  = 0
缓冲池数据物理读取次数                  = 0
缓冲池索引逻辑读取次数                  = 0
缓冲池索引物理读取次数                  = 0
缓冲池总计读取时间（毫秒）              = 0
缓冲池总计写入时间（毫秒）              = 0
```

在分区数据库环境中使用自调整内存功能

在分区数据库环境中启用自调整内存功能之后，将出现一个单独的数据库分区（称为*调整分区*），此分区将监视内存配置的情况，并将任何配置更改传播到所有其他数据库分区以使所有参与数据库分区的配置保持一致。

调整分区是根据多个特征选择的，例如分区组中的数据库分区数以及已定义的缓冲池数。

- 要确定当前已指定为调整分区的数据库分区，请调用 ADMIN_CMD 过程，如下所示:

  ```
  CALL SYSPROC.ADMIN_CMD('get stmm tuning dbpartitionnum')
  ```

- 要更改调整分区，请调用 ADMIN_CMD 过程，如下所示:

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning dbpartitionnum <partitionnum>')
  ```

  将以异步方式或者在数据库下次启动时更新调整分区。要让内存调整器自动选择调整分区，请输入 -1 作为 *partitionnum* 的值。

在分区数据库环境中启动内存调整器

由于自调整内存功能要求所有分区都处于活动状态，因此在分区数据库环境中，仅当数据库由显式的 ACTIVATE DATABASE 命令激活时，才会启动内存调整器。

对特定数据库分区禁用自调整内存功能

- 要对部分数据库分区禁用自调整内存功能，请对那些数据库分区将 **self_tuning_mem** 数据库配置参数设置为 OFF。
- 要对特定数据库分区中由配置参数控制的部分内存使用者禁用自调整内存功能，请对该数据库分区将相关配置参数值或缓冲池大小设置为 MANUAL 或某个特定值。建议使自调整内存功能的配置参数值在所有运行中的分区中保持一致。
- 要对特定数据库分区中的特定缓冲池禁用自调整内存功能，请发出 ALTER BUFFERPOOL 语句并指定大小值以及要在其中禁用自调整内存功能的分区。

  对特定数据库分区指定缓冲池大小的 ALTER BUFFERPOOL 语句将在 SYSCAT.BUFFERPOOLDBPARTITIONS 目录视图中为该缓冲池创建例外条目或更新现有条目。如果某个缓冲池的例外条目已存在，并且缺省缓冲池大小设置为 AUTOMATIC，那么该缓冲池将不会参与自调整操作。要除去例外条目，以便可以对缓冲池启用自调整功能:

  1. 通过发出 ALTER BUFFERPOOL 语句并将缓冲池大小设置为特定值，对此缓冲池禁用自调整功能。
  2. 发出另一个 ALTER BUFFERPOOL 语句，以便将此数据库分区中缓冲池的大小设置为缺省大小。
  3. 通过发出另一个 ALTER BUFFERPOOL 语句并将缓冲池大小设置为 AUTOMATIC，对此缓冲池启用自调整功能。

在不均匀的环境中启用自调整内存功能

理想情况下，数据应该均匀地分布在所有数据库分区中，并且每个分区中运行的工作负载的内存需求应该比较接近。如果数据分布不均匀，以致一个或多个数据库分区包含的数据显著多于或少于其他数据库分区，那么就不应该对这些不规则的数据库分区启用自调整功能。这也适用于不同数据库分区中的内存需求不均匀的情况。例如，如果只在一个分区中执行需要大量资源的排序操作，或者某些数据库分区使用的硬件与其他分区不同并且有更多的可用内存，那么将发生这种情况。在此类环境中，仍然可以对某些数据库分区启用自调整内存功能。要在不均匀环境中利用自调整内存功能，请确定一组具有相似数据和内存需求的数据库分区并对它们启用自调整功能。对于其余分区，应该以手动方式进行内存配置。

缓冲池管理

缓冲池为数据库页提供工作内存和高速缓存。

缓冲池通过允许从内存（而不是磁盘）中读取数据来提高数据库系统的性能。由于大多数页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。

当应用程序访问表行时，数据库管理器将在缓冲池中查找包含该行的页。如果在缓冲池中找不到该页，那么数据库管理器将从磁盘中读取该页并将其放入缓冲池。然后，可以使用该数据来处理查询。

激活数据库时，将为缓冲池分配内存。第一个执行连接的应用程序可能以隐式方式激活数据库。此外，还可以在数据库管理器运行期间创建和删除缓冲池以及调整缓冲池大小。可使用 ALTER BUFFERPOOL 语句来增大缓冲池的大小。缺省情况下，如果有足够的可用内存，那么执行此语句时将立即调整缓冲池大小。如果执行此语句时没有足够的可用内存，那么将在重新激活数据库时分配内存。如果减小缓冲池的大小，那么将在事务落实时释放内存。缓冲池内存将在数据库停用时被释放。

为了确保所有情况下都有适当的缓冲池可用，DB2 将创建多个小型的系统缓冲池，它们每一个的页大小分别为 4 KB、8 KB、16 KB 和 32 KB。每个缓冲池的大小为 16 页。这些缓冲池处于隐藏状态；它们未包含在系统目录或缓冲池系统文件中。您不能直接使用或更改这些缓冲池，但 DB2 在下列情况下将使用它们:

- 指定的缓冲池由于在创建时指定了 DEFERRED 关键字而未启动，或者由于没有足够的内存可用来创建所需页大小的缓冲池而导致该缓冲池处于不活动状态

  一条消息将被写入到管理通知日志。必要时，表空间将重新映射到系统缓冲池。性能可能明显下降。
- 在数据库连接尝试期间无法启动缓冲池

  此问题通常是由于严重的原因（例如内存不足情况）所致。虽然 DB2 将因为系统缓冲池的存在而继续充分发挥其功能，但性能将明显下降。您应该立即解决此问题。发生此问题时，您将接收到警告，并且系统会将一条消息写入管理通知日志。

创建缓冲池时，除非明确地指定另一页大小，否则页大小将是创建数据库时指定的大小。由于仅当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，对缓冲池指定的页大小应该由表空间的页大小确定。在创建缓冲池后，就无法更改它的页大小。

通过发出 db2mtrk 命令调用的内存跟踪程序使您能够查看已分配给缓冲池的数据库内存量。您还可以使用 GET SNAPSHOT 命令并检查缓冲池的当前大小（**bp_cur_buffsz** 监视元素的值）。

作为 DB2 工作负载管理器所提供的更大的工作负载管理功能集的组成部分，可以对活动的缓冲池优先级进行控制。有关更多信息，请参阅“DB2 工作负载管理器概念简介”和“服务类的缓冲池优先级”。

数据页的缓冲池管理

缓冲池页可能在使用中，也可能未被使用，并且可以是脏页或干净页。

- *使用中的页*就是当前正被读取或更新的页。如果某页正被更新，那么只有更新者才能访问该页。但是，如果该页并非正被更新，那么可以同时有许多并发的阅读者。
- *脏页*包含已被更改但尚未写入磁盘的数据。

这些页将一直保留在缓冲池中，直到数据库关闭、其他页需要使用另一页所占用的空间或者该页被显式地从缓冲池中清除（例如，在删除对象的过程中执行此清除）为止。以下条件确定除去哪一页以使其空间可供另一页使用:

- 该页在多久以前被引用？
- 该页再次被引用的可能性有多大？
- 该页包含什么类型的数据？
- 该页是否已在内存中更改但尚未写入磁盘？

已更改的页在被覆盖之前，始终会写入磁盘。除非需要空间，否则不会从缓冲池中自动除去已写入磁盘的已更改页。

页清除程序代理程序

在经过认真调整的系统中，通常由页清除程序代理程序将已更改的页（即脏页）写入磁盘。页清除程序代理程序作为后台进程执行 I/O，它们使应用程序能够更快地运行，这是因为，其代理程序可以执行实际的事务工作。页清除程序代理程序有时被称为*异步页清除程序*或*异步缓冲区写程序*，这是因为它们不与其他代理程序的工作协调进行，而只是在有需要时才工作。

为了提高需要执行大量更新操作的工作负载的性能，可以启用*主动页清除*功能，以使页清除程序在任何给定时刻选择要写哪些脏页时更为主动。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如此。

第 84 页的图 19 举例说明页清除程序代理程序与数据库代理程序之间如何共享缓冲池管理工作。

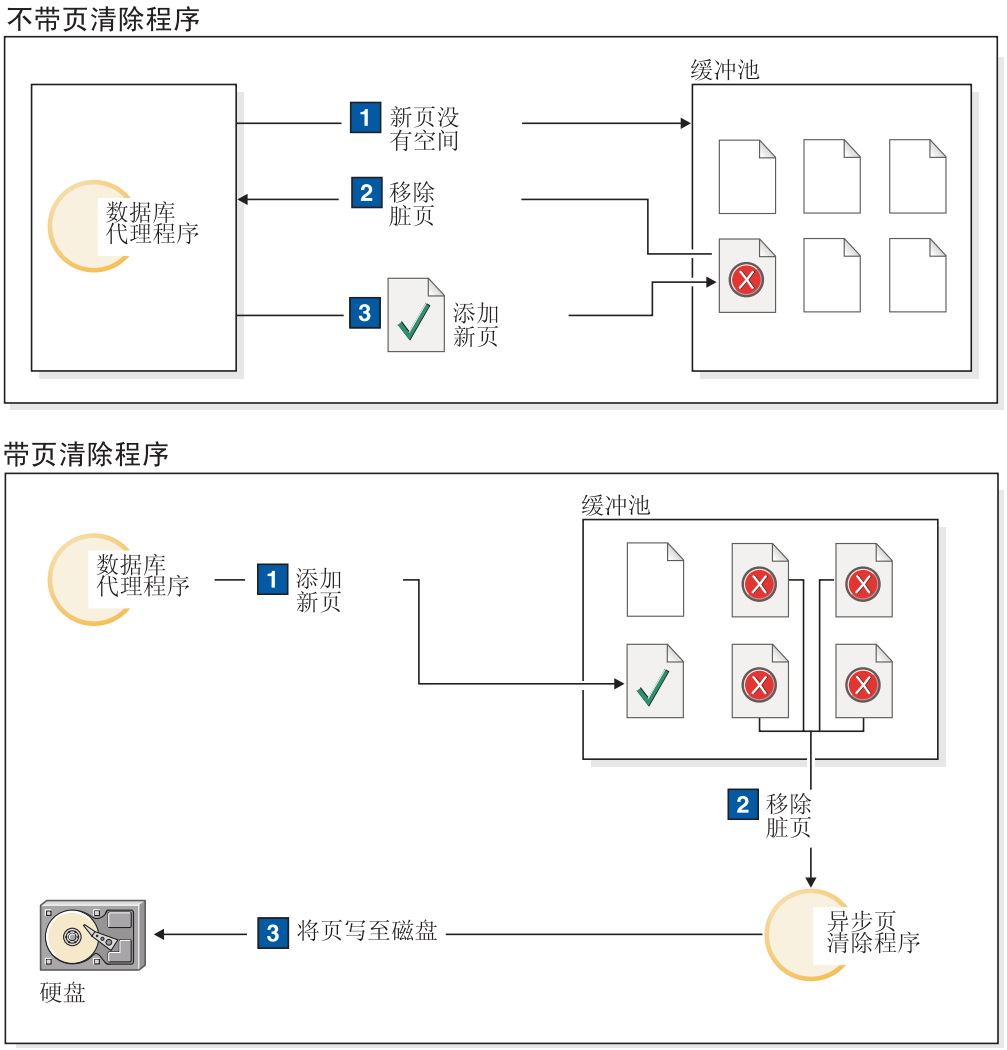


图 19. 异步页清除. 脏页被写入磁盘。

页清除和快速恢复

如果将较多的页写入磁盘，那么系统崩溃后的数据库恢复速度将加快，这是因为数据库管理器可以从磁盘重建较多的缓冲池，而不必根据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:

- 最近写入的日志记录
- 描述对缓冲池数据所作的最早更改的日志记录

页清除程序将脏页写入磁盘时，会确保恢复期间需要回放的日志的大小不超过以下值:

```
logfilsiz * softmax / 100（以 4-KB 页计）
```

其中:

- **logfilsiz** 表示日志文件的大小
- **softmax** 表示数据库崩溃后要恢复的日志文件所占的百分比；例如，如果 **softmax** 的值为 250，那么发生崩溃后，将有 2.5 个日志文件包含需要恢复的更改

要将恢复期间日志读取时间缩至最短，请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功能，那么 **pool_lsn_gap_clns**（触发缓冲池日志空间清除程序次数）监视元素将提供此信息。如果已启用主动页清除页功能，那么应该不会发生这种情况，并且 **pool_lsn_gap_clns** 的值为 0。

可以使用 **log_held_by_dirty_pages** 监视元素来确定是否页清除程序未清除足够多的页数，因此无法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**，那么需要提供更多页清除程序或者调整 **softmax**。

管理多个数据库缓冲池

尽管每个数据库都至少需要一个缓冲池，但您可以为表空间具有多页大小的单一数据库创建多个缓冲池，每个缓冲池可以具有不同的大小或使用不同的页大小。

您可以使用 ALTER BUFFERPOOL 语句来调整缓冲池大小。

新数据库具有名为 IBMDEFAULTBP 的缺省缓冲池，其缺省页大小基于创建数据库时指定的页大小。缺省页大小作为 **pagesize** 参考数据库配置参数进行存储。如果您创建具有缺省页大小的表空间，并且未对其指定特定的缓冲池，那么该表空间将被指定到缺省缓冲池。您可以调整缺省缓冲池的大小并更改其属性，但不能将其删除。

缓冲池的页大小

在创建或升级数据库之后，可以创建其他缓冲池。如果创建使用 8-KB 页大小作为缺省值的数据库，那么将创建具有缺省页大小（在本例中，这是 8 KB）的缺省缓冲池。此外，还可以创建具有 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。此方法不要求您在创建数据库时更改 4-KB 缺省页大小。不能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（例如 8 KB、16 KB 或 32 KB）的表空间，那么需要对其指定具有相同页大小的缓冲池。如果此缓冲池当前未处于活动状态，那么 DB2 会尝试临时地将该表空间指定给另一个使用相同页大小的活动缓冲池，或者指定给第一次将客户机连接至数据库时 DB2 创建的某个缺省系统缓冲池。当该数据库再次被激活时，如果原来指定的缓冲池处于活动状态，那么 DB2 会将该表空间指定给该缓冲池。

使用 CREATE BUFFERPOOL 语句创建缓冲池时，如果未指定大小，那么缓冲池大小将设置为 AUTOMATIC 并由 DB2 管理。以后，要更改缓冲池大小，请使用 ALTER BUFFERPOOL 语句。

在分区数据库环境中，一个数据库的每个缓冲池在所有数据库分区中的缺省定义都相同，除非在 CREATE BUFFERPOOL 语句中另行指定或者特定数据库分区的缓冲池大小被 ALTER BUFFERPOOL 语句更改。

大型缓冲池的优点

大型缓冲池具有下列优点:

- 它们使频繁被请求的数据页能够保留在缓冲池中，从而提高访问速度。较少的 I/O 操作可以减少 I/O 争用情况，从而缩短响应时间并减少 I/O 操作所需的处理器资源。
- 它们提供了在保持响应时间不变的前提下实现更高事务处理速率的机会。

- 它们能够防止频繁使用的磁盘存储设备（例如用于存储目录表以及频繁引用的用户表和索引的设备）发生 I/O 争用情况。由于包含临时表空间的磁盘存储设备上的 I/O 争用情况减少，因此查询所需执行的排序操作也能受益。

多个缓冲池的优点

如果系统中存在下列任何一种情况，那么应该只使用一个缓冲池:

- 缓冲池总空间量小于 10000 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员
- 您正在测试系统中工作

在所有其他情况下，请考虑使用多个缓冲池，原因如下:

- 可以将临时表空间指定到独立的缓冲池，以便提高需要临时存储器的查询（尤其是执行大量排序操作的查询）的性能。
- 如果数据必须由很多小型的更新事务应用程序反复并快速地访问，那么请考虑将包含该数据的表空间指定到独立的缓冲池。如果此缓冲池的大小合适，那么它的页被找到的机会将更大，这有利于缩短响应时间和降低事务成本。
- 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，您可能想将频繁更新的表和索引放在单独的缓冲池中，以便与那些频繁查询但很少更新的表和索引分开。
- 对于很少使用的应用程序（尤其是需要对一个很大的表非常随机地进行访问的应用程序）所访问的数据，可使用较小的缓冲池。在这种情况下，不需要使数据保存在缓冲池中的时间超出单个查询的运行时间。最好为此类数据保留一个小缓冲池，以便腾出额外的内存供其他缓冲池使用。

将数据分到不同的缓冲池之后，可以根据统计信息和记帐跟踪来生成良好并且成本相对较低的性能诊断数据。

自调整内存管理器（STMM）非常适合于调整包含多个缓冲池的系统。

启动时的缓冲池内存分配

在创建缓冲池或更改缓冲池时，所有缓冲池所需的全部内存必须可供数据库管理器使用，以便数据库启动时能够分配所有缓冲池。如果在数据库管理器处于联机状态时创建或更改缓冲池，那么数据库全局内存中应该有附加的内存可用。如果创建新缓冲池或增大现有缓冲池的大小时指定了 DEFERRED 关键字，但无法获得所需的内存，那么数据库管理器将在下次激活该数据库时执行更改。

如果数据库启动时无法获得此内存，那么数据库管理器将仅使用最小大小为 16 页的系统缓冲池（每个页大小对应一个系统缓冲池）并返回警告。该数据库将一直保持处于此状态，直到您更改其配置并且该数据库可以完全重新启动为止。虽然性能可能欠佳，但您可以连接到该数据库、重新配置缓冲池大小或者执行其他关键任务。完成这些任务后，请重新启动该数据库。不要在这种状态下长时间运行数据库。

为了避免在仅使用系统缓冲池的情况下启动数据库，请使用 **DB2_OVERRIDE_BPF** 注册表变量来优化对可用内存的使用。

主动页清除

从 DB2 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行为。使用此方法时，页清除程序在任何给定时刻都将更为主动地选择要写出的脏页。

这种主动的页清除方法与缺省页清除方法主要在以下两方面有所不同:

- 页清除程序不再响应 **chngpgs_thresh** 数据库配置参数的值。

  当良好牺牲页的数目减少到低于可接受的值时，页清除程序将搜索整个缓冲池，写出潜在的牺牲页，并将这些页的位置告知代理程序。
- 页清除程序不再响应记录器发出的日志序号（LSN）间隔触发器。

  如果更新缓冲池中最旧页的日志记录与当前日志位置之间的日志空间量超出 **softmax** 数据库配置参数所允许的值，那么称数据库遇到 *LSN 间隔*。

  使用缺省的页清除方法时，检测到 LSN 间隔的记录器将触发页清除程序，以便写出所有引起 LSN 间隔的页；即，页清除程序将写出那些比 **softmax** 值所允许的时效旧的页。页清除程序将在空闲状态与执行大量写页活动之间交替。这将使 I/O 子系统达到饱和状态，从而影响其他正在对页执行读写操作的代理程序。并且，检测到 LSN 间隔时，页清除程序可能无法足够快地执行清除，并且 DB2 可能会耗尽日志空间。

  主动页清除方法通过将相同次数的写操作分布到更长的时间段来调整此行为。页清除程序通过根据当前活动级别不仅清除引起 LSN 间隔的页，而且清除不久之后有可能引起 LSN 间隔的页来完成此调整。

要使用新的页清除方法，请将 **DB2_USE_ALTERNATE_PAGE_CLEANING** 注册表变量设置为 ON。

提高更新性能

当代理程序更新页时，数据库管理器使用协议来最大程度地减少事务所需的 I/O 并确保可恢复性。

此协议包括下列步骤:

1. 使用互斥锁定将所要更新的页锁定。将一条日志记录写入日志缓冲区，从而描述如何撤销和重做更改。作为此操作的一部分，获取日志序号（LSN）并将其存储到正在更新的页的页头。
2. 对该页进行更新。
3. 将该页解锁。因为尚未将对该页所作的更改写入磁盘，所以认为该页是“脏”页。
4. 更新日志缓冲区。将日志缓冲区中的数据和“脏”数据页都写入磁盘。

为了提高性能，这些 I/O 操作将被延迟到系统负载较低时或者有必要执行这些操作以确保可恢复性或限制恢复时间时进行。确切而言，脏页在下列情况下被写入磁盘:

- 另一个代理程序选择它作为牺牲页
- 页清除程序处理该页。这可能在以下情况下发生:
  - 另一个代理程序选择它作为牺牲页
  - 超出 **chngpgs_thresh** 数据库配置参数值，从而导致异步页清除程序被唤醒并将更改后的页写入磁盘。如果已对数据库启用主动页清除功能，那么此值无关紧要，并且不会触发页清除。

- 超出 **softmax** 数据库配置参数值，从而导致异步页清除程序被唤醒并将更改后的页写入磁盘。如果已对数据库启用主动页清除功能，并且已经为数据库正确地配置页清除程序数，那么始终都不会超出此值。
- 干净页的数目下降到过低水平。只有采用主动页清除方法时，页清除程序才会对这种情况作出反应。
- 脏页当前导致或者预期会导致 LSN 间隔情况。只有采用主动页清除方法时，页清除程序才会对这种情况作出反应。

• 该页是使用 NOT LOGGED INITIALLY 定义的表的一部分，并且该更新后面跟有 COMMIT 语句。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。

将数据预取至缓冲池

*预取*页是指，在应用程序需要一页或多页之前，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缩短 I/O 等待时间，因此有助于提高性能。此外，并行 I/O 也有助于提高预取效率。

预取分为两类:

• *顺序预取*在应用程序需要这些页之前，将连续的页读入缓冲池。
• *列表预取*（有时称为*列表顺序预取*）高效地预取一组不连续的数据页。

预取数据页与数据库管理器代理程序读方法不同，后一种方法用于检索一页或少量连续页但只将一页数据传送给应用程序时的情况。

预取与分区内并行性

分区内并行性使用多个子代理程序来扫描索引或表，预取会对分区内并行性的性能产生重要的影响。此类并行扫描将产生较高的数据使用率，从而要求较高的预取率。

并行扫描与串行扫描相比，预取不足的成本更高。如果串行扫描期间未进行预取，那么由于代理程序需要等待 I/O，所以查询的运行速度更慢。如果并行扫描期间未进行预取，那么当一个子代理程序等待 I/O 时，所有子代理程序都可能需要进行等待。

在此上下文中，分区内并行性下的预取极为重要，因此更为主动地执行；顺序检测机制允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程序数目的增加而增大。

顺序预取:

使用单一 I/O 操作将多个连续的页读入缓冲池可以大大降低应用程序开销。

当数据库管理器确定顺序 I/O 合适并且预取操作有助于提高性能时，预取操作就会启动。对于表扫描和表排序这类情况，数据库管理器将自动选择顺序预取。以下示例（可能要求执行表扫描）适合于顺序预取:

```
SELECT NAME FROM EMPLOYEE
```

顺序检测

有时，并不能立即确定顺序预取能否提高性能。在此类情况下，数据库管理器可以监视 I/O 并在顺序页读操作发生时激活预取。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。您可以使用 **seqdetect** 数据库配置参数来控制数据库管理器是否执行顺序检测。

例如，如果启用顺序检测，那么顺序预取有可能使以下 SQL 语句受益:

```
SELECT NAME FROM EMPLOYEE
WHERE EMPNO BETWEEN 100 AND 3000
```

在本示例中，优化器可能已开始使用基于 EMPNO 列的索引来扫描表。如果该表相对于此索引而言高度集群，那么数据页读操作将几乎按顺序进行，并且预取有助于提高性能。同样，如果必须检查许多索引页，并且数据库管理器检测到正在按顺序读取索引页，那么有可能进行索引页预取。

表空间的 PREFETCHSIZE 选项的含义

CREATE TABLESPACE 或 ALTER TABLESPACE 语句的 PREFETCHSIZE 子句允许您指定执行数据预取时将从表空间中读取的预取页数。您指定的值（或者“AUTOMATIC”）将存储在 SYSCAT.TABLESPACES 目录视图的 PREFETCHSIZE 列中。

您最好将 PREFETCHSIZE 值明确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）与表空间的 EXTENTSIZE 值（数据库管理器在使用另一个容器前写入容器的页数）的乘积。例如，如果扩展数据块大小为 16 页，并且表空间包含两个容器，那么可以将预取大小设置为 32 页。如果每个容器有 5 个物理磁盘，那么可以将预取大小设置为 160 页。

数据库管理器将监视缓冲池的使用情况，以确保预取不会从缓冲池中除去另一个工作单元所需的页。为了避免引起问题，数据库管理器可将预取的页数限制为小于您对表空间指定的数目。

预取大小可能会显著影响性能，对于大型表扫描而言尤其如此。请使用数据库系统监视器和其他系统监视工具来帮助您调整表空间的预取大小。您可以收集有关是否存在下列情况的信息:

- 查询等待 I/O（通过使用适用于操作系统的监视工具来检查这种情况）
- 正在进行预取（通过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）数据元素来检查这种情况）

如果查询预取数据时发生 I/O 等待情况，那么请增大 PREFETCHSIZE 的值。如果预取程序不是这些 I/O 等待情况的原因，那么增大 PREFETCHSIZE 值对提高查询性能没有帮助。

对于所有类型的预取而言，如果预取大小是表空间的扩展数据块大小的倍数，并且这些扩展数据块在多个不同的容器中，那么可以并行地执行多个 I/O 操作。为了提高性能，请将容器配置为使用多个不同的物理设备。

用于提高顺序预取性能的基于块的缓冲池:

从磁盘预取页这一操作由于 I/O 开销而成本高昂。如果处理与 I/O 交迭，那么可以显著提高吞吐量。

大多数平台提供了将连续页从磁盘读入非连续内存部分的高性能基本方法。这些基本方法通常称为*散射读*或*向量化 I/O*。在某些平台上，这些基本方法的性能不能与采用较大块大小执行 I/O 相比。

缺省情况下，缓冲池基于页，这意味着将磁盘上的连续页预取到内存中的非连续页。如果可以将连续页从磁盘读取到缓冲池的连续页中，那么可以提高顺序预取性能。

您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；每个块都包含指定数量的连续页，这称为*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 服务器在考虑执行基于块的 I/O 时采用的粒度。扩展数据块是在容器中对表空间进行条带分割的粒度。由于可以将多个具有不同扩展数据块大小的表空间与使用同一个块大小定义的缓冲池绑定，因此，请考虑扩展数据块大小如何与块大小进行交互以便高效使用缓冲池内存。在下列情况下，可能会浪费缓冲池内存:

- 扩展数据块大小（用于确定预取请求大小）小于对缓冲池指定的块大小
- 预取请求中的某些页已存在于缓冲池的页区域中

I/O 服务器允许在每个缓冲池块中浪费一些页，但是，如果浪费的页过多，那么 I/O 服务器将执行并非基于块的预取以便将数据装入到缓冲池的页区域，从而导致性能达不到最佳水平。

为了获得最佳性能，请将扩展数据块大小相同的表空间与块大小等于表空间扩展数据块大小的缓冲池绑定。在扩展数据块大小大于块大小时，而不是在扩展数据块大小小于块大小时，可以实现良好性能。

要创建基于块的缓冲池，请使用 CREATE BUFFERPOOL 或 ALTER BUFFERPOOL 语句。

注: 基于块的缓冲池用于顺序预取。如果应用程序不使用顺序预取方法，那么将浪费缓冲池的块区域。

列表预取:

列表预取（即*列表顺序预取*）是一种高效访问数据页的方法，即使那些数据页不连续亦如此。

列表预取功能可以与单索引访问或多索引访问结合使用。

如果优化器使用索引来访问行，那么它可将数据页读取操作推迟到从该索引中获取所有行标识（RID）之后进行。例如，优化器可以执行索引扫描以确定要检索的行和数据页。

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

然后，使用以下搜索条件:

```
WHERE NAME BETWEEN 'A' and 'I'
```

如果数据未根据此索引进行集群，那么列表预取过程将包括对索引扫描操作所获取的 RID 列表进行排序的步骤。

有关预取和并行性的 I/O 服务器配置:

要启用预取功能，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 活动发来的预取请求。这些预取请求包含有关满足查询所需的 I/O 的描述。

通过使用 **num_ioservers** 数据库配置参数配置足够的 I/O 服务器，可以大大提高能够受益于预取的查询的性能。为了最大程度地提高并行 I/O 的机会，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 服务器数目要好于低估此数目。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，因此不会影响性能。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小的进程，因此，某些编号较大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程序的数目
- I/O 服务器可并行工作的最大程度

请考虑将 **num_ioservers** 的值设置为 AUTOMATIC，以使数据库管理器能够根据系统配置智能地选择值。

使用并行 I/O 预取示例:

I/O 服务器用于将数据预取到缓冲池。

此过程如第 92 页的图 20 所示。

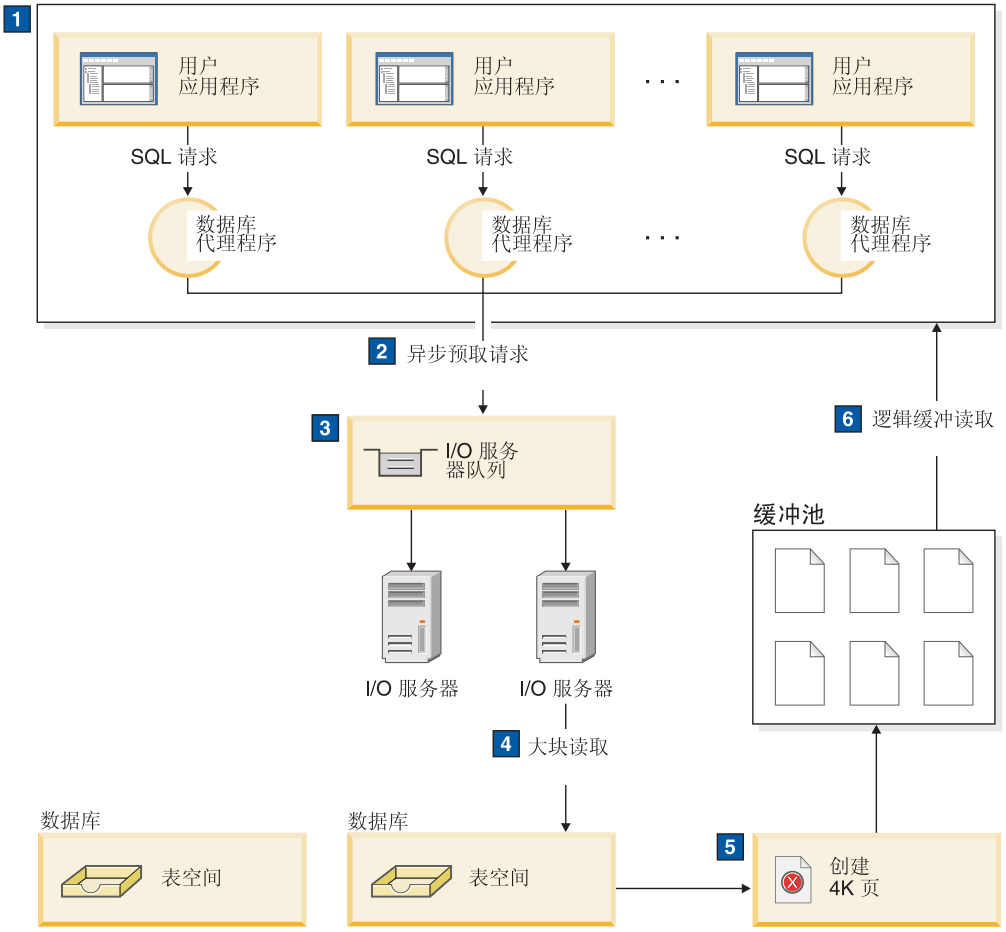


图 20. 使用 I/O 服务器来预取数据

1 用户应用程序将请求传递到已由数据库管理器指定给用户应用程序的数据库代理程序。

2, **3** 数据库代理程序确定应使用预取方法来获取满足该请求所需的数据，并将预取请求写入 I/O 服务器队列。

4, **5** 第一个可用的 I/O 服务器从这个队列读取预取请求，然后将数据从表空间读入缓冲池。可以同时在表空间中访存数据的 I/O 服务器的数目取决于队列中的预取请求数以及 **num_ioservers** 数据库配置参数指定的 I/O 服务器数。

6 数据库代理程序对缓冲池中的数据页执行必要的操作并将结果返回给用户应用程序。

并行 I/O 管理:

如果表空间中存在多个容器，那么数据库管理器可以启动*并行 I/O*，即，数据库管理器使用多个 I/O 服务器来处理单个查询的 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多个容器。并行 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，可以执行并行 I/O 的实际 I/O 服务器数限制为所请求数据分布于的物理设备数。因此，所需的 I/O 服务器数就是物理设备数。

在下列情况下，以不同方式启动并行 I/O:

- 对于*顺序预取*，当预取大小是表空间的扩展数据块大小的倍数时，将启动并行 I/O。每个预取请求都按扩展数据块边界分为较小的请求。然后，将这些小型的请求指定到不同的 I/O 服务器。
- 对于*列表预取*，根据存储数据页的容器，将每个页列表分成较小的列表。然后，将这些小列表指定到不同的 I/O 服务器。
- 对于*数据库或表空间备份和复原*，并行 I/O 请求数等于备份缓冲区大小除以扩展数据块大小，最大大小为容器数。
- 对于*数据库或表空间复原*，将启动并行 I/O 请求，并且划分方式与顺序预取的划分方式相同。数据不会被复原到缓冲池，而是直接从复原缓冲池移至磁盘。
- 装入数据时，可以使用 DISK_PARALLELISM 命令选项来指定 I/O 并行性级别。如果未指定此选项，那么数据库管理器将使用缺省值，此值基于所有与该表相关联的表空间的表空间容器累计数目。

为了使并行 I/O 的性能最为理想，请确保:

- 有足够的 I/O 服务器。请将 I/O 服务器的数目指定为稍大于数据库中所有表空间所使用的容器的数目。
- 扩展数据块大小和预取大小适合于表空间。为了防止过度使用缓冲池，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相当小，合适的值在 8 至 32 页之间。
- 容器在不同的物理驱动器上。
- 为了确保一致的并行度，所有容器的大小应该相同。

  如果一个或多个容器比其他容器小，那么它们会降低优化并行预取的可能性。请考虑以下示例:

  – 在一个较小的容器装满后，其他数据将存储到其余容器中，从而导致容器变得不平衡。容器不平衡会导致并行预取性能下降，这是因为，可从中预取数据的容器数可能小于容器的总数。
  – 如果以后添加了一个较小的容器并对数据进行重新平衡，那么这个较小容器中包含的数据将少于其他容器。相对于其他容器，此容器的少量数据将不会优化并行预取。
  – 如果某个容器较大，并且所有其他容器都已装满，那么这个较大的容器是唯一可用于存储更多数据的容器。数据库管理器无法使用并行预取方法来访问这项附加的数据。
- 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查询来缩短该查询的耗用时间。需要足够的 I/O 容量以使每个处理器都忙于工作。为了提供该 I/O 容量，通常需要附加的物理驱动器。

  预取大小必须较大，以使预取操作以较高速率进行以及确保高效地使用 I/O 容量。

  所需的物理驱动器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。

在 AIX 上配置 IOCP:

AIX 5.3 TL9 SP2 和 AIX 6.1 TL2 包含作为基本安装版本组成部分提供的 I/O 完成端口（IOCP）文件集。但是，如果通过升级操作系统（而不是通过安装新操作系统）来应用最低操作系统要求，那么必须单独配置 I/O 完成端口（IOCP）。

1. 输入 lslpp 命令，以检查系统上是否已安装 IOCP 模块。

   ```
   $ lslpp -l bos.iocp.rte
   ```

   结果输出应该类似于以下示例:

   ```
   Fileset                      Level     State       Description
   ----------------------------------------------------------------------------
   Path: /usr/lib/objrepos
     bos.iocp.rte               5.3.9.0   APPLIED     I/O Completion Ports API

   Path: /etc/objrepos
     bos.iocp.rte               5.3.0.50  COMMITTED   I/O Completion Ports API
   ```

2. 输入 lsdev 命令，以检查 IOCP 端口的状态是否为“可用”（Available）。

   ```
   $ lsdev -Cc iocp
   ```

   结果输出应该与以下示例匹配:

   ```
    iocp0 Available  I/O Completion Ports
   ```

   如果 IOCP 端口状态为“已定义”（Defined），那么请将状态更改为“可用”（Available）。

   a. 以 root 用户身份登录至服务器并发出以下命令:

      ```
      # smitty iocp
      ```

   b. 选择更改/显示 I/O 完成端口的特征。

   c. 将已配置的系统重新启动时的状态由“已定义”（Defined）更改为“可用”（Available）。

   d. 再次输入 lsdev 命令以确认 IOCP 端口的状态已更改为“可用”（Available）。

初始用户连接方案中的数据库停用行为

用户第一次连接到数据库时，将激活该数据库。在单一分区环境中，该数据库将被装入到内存并保持处于此状态，直到最后一个用户断开连接为止。同一行为也适用于多分区环境，在此环境中，任何初始用户连接都将在该数据库的本地分区和目录分区中激活该数据库。

当最后一个用户断开连接时，将在本地分区以及此用户作为数据库的最后一个活动用户连接的任何远程分区中关闭该数据库。这种根据第一个连接和最后一个断开连接进行的数据库激活和停用称为*隐式激活*。激活由第一个用户连接启动，并且此激活将一直保持有效，直到用户执行 CONNECT RESET 或者终止或删除该连接（这将导致以隐式方式停用该数据库）为止。

将数据库装入到内存的过程也非常复杂。此过程包括初始化所有数据库组件（其中包括缓冲池），这是应该最大程度减少的处理类型，在对性能敏感的环境中尤其如此。此行为在多分区环境中尤其重要，在这些环境中，从一个数据库分区中发出的查询将到达其他包含目标数据集部件的分区。那些数据库分区将被激活或停用，这取决于用户应用程序的连接和断开连接行为。当用户发出的查询第一次到达某个数据库分区时，该查询将产生第一次激活该分区的成本。当该用户断开连接时，将停用该数据

库，除非先前已对该远程分区建立其他连接。如果下一个传入查询需要访问该远程分区，那么必须先在该分区中激活该数据库。每次激活和停用数据库（或者数据库分区，如果适用的话）都会产生此成本。

此行为的唯一例外是，用户选择通过发出 ACTIVATE DATABASE 命令以显式方式激活数据库。在此命令成功完成之后，数据库将一直存在于内存中，即使最后一个用户断开连接亦如此。这既适用于单一分区环境，也适用于多分区环境。要停用这样的数据库，请发出 DEACTIVATE DATABASE 命令。这两个命令都具有全局作用域，这意味着，如果情况适用的话，将在所有数据库分区中激活或停用数据库。由于将数据库装入到内存这一操作的处理量非常大，请考虑使用 ACTIVATE DATABASE 命令以显式方式激活数据库，而不是依靠通过数据库连接以隐式方式进行激活。

调整排序性能

由于查询通常需要经过排序或分组的结果，因此正确地配置排序堆对于良好的查询性能而言至关重要。

在下列情况下，需要进行排序:

- 不存在能够满足所请求顺序的索引（例如，使用了 ORDER BY 子句的 SELECT 语句）
- 存在索引，但执行排序比使用索引更有效
- 创建索引
- 删除索引，这将导致对索引页号进行排序

影响排序的元素

下列因素将影响排序性能:

- 下列配置参数的设置:
  - 排序堆大小（**sortheap**），此参数指定用于执行每次排序的内存量
  - 排序堆阈值（**sheapthres**）和共享排序的排序堆阈值（**sheapthres_shr**），这些参数控制整个实例中可用于执行排序的内存总量
- 工作负载中需要执行大量排序操作的语句数
- 是否存在有助于避免执行不必要的排序操作的索引
- 使用了未最大程度减少排序需求的应用程序逻辑
- 并行排序，这可以提高排序性能，但仅当语句使用分区内并行性时才可行
- 排序是否*溢出*。如果经过排序的数据在排序堆（这是每次执行排序时分配的内存块）中放不下，那么数据将溢出到数据库所拥有的临时表中。
- 排序结果是否*管道式*结果。如果经过排序的数据可直接返回，而不需要一个临时表来存储经过排序的列表，那么这是管道式排序。

  对于管道式排序，在应用程序关闭与排序相关联的游标之前，排序堆不会被释放。在游标关闭之前，管道式排序可以持续耗用内存。

尽管可以完全在排序内存中执行排序，但这可能会导致过度进行页交换。在这种情况下，将失去大型排序堆的优势。因此，每当您调整排序配置参数时，应该使用操作系统监视器来跟踪系统页面调度方面的任何变化。

用于管理排序性能的技术

确定排序对性能产生重大影响的特定应用程序和语句:

1. 在应用程序级别和语句级别设置事件监视器，以帮助您确定排序总时间最长的应用程序。
2. 对于其中的每个应用程序，查找*排序总时间*最长的语句。

   您也可以搜索说明表，以确定执行了排序操作的查询。
3. 使用这些语句作为设计顾问程序的输入，该程序将标识索引并可创建索引以减少排序需求。

可以使用自调整内存管理器（STMM）自动根据需要动态地分配和取消分配排序所需的内存资源。要使用此功能:

- 通过将 **self_tuning_mem** 配置参数设置为 ON，对数据库启用自调整内存功能。
- 将 **sortheap** 和 **sheapthres_shr** 配置参数设置为 AUTOMATIC。
- 将 **sheapthres** 配置参数设置为 0。

您还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**、**sheapthres_shr** 和 **sheapthres** 配置参数。对于每个数据库管理器以及每个数据库:

1. 设置并运行一个典型的工作负载。
2. 对于每个适用的数据库，收集下列性能变量在基准程序工作负载周期内的平均值:
   - 使用中排序堆总大小（**sort_heap_allocated** 监视元素的值）
   - 活动排序次数和活动散列连接数（**active_sorts** 和 **active_hash_joins** 监视元素的值）
3. 将 **sortheap** 设置为每个数据库的平均*使用中排序堆总大小*。

   注: 如果将长键用于排序，那么可能需要增大 **sortheap** 配置参数的值。
4. 设置 **sheapthres**。要估算适当的大小:
   a. 确定实例中哪个数据库的 **sortheap** 值最大。
   b. 确定此数据库的排序堆的平均大小。

      如果太难以确定，那么使用最大排序堆大小的 80%。
   c. 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，您可以使用基准测试技术来优化此值。

数据组织

随着时间的推移，当记录分布在越来越多的数据页上时，表中的数据将变成碎片，从而增大表和索引的大小。这会增加执行查询时需要读取的页数。重组表和索引会压缩数据，从而回收浪费的空间并改善数据访问情况。

用于执行表或索引重组的步骤如下所示:

1. 确定是否需要重组任何表或索引。
2. 选择重组方法。
3. 对所标识的对象执行重组。
4. 可选: 监视重组进度。

5. 确定重组是否成功。对于脱机表重组以及任何索引重组，操作将以同步方式进行，并且结果将在操作完成后显示。对于联机表重组而言，操作以异步方式进行，详细信息将包含在历史记录文件中。
6. 收集有关所重组的对象的统计信息。
7. 重新绑定需要访问所重组的对象的应用程序。

表重组

对表数据进行大量更改之后，逻辑上按顺序排列的数据可能会驻留在非顺序数据页中，从而导致数据库管理器必须执行附加的读操作才能访问数据。此外，如果已经删除了许多行，那么还需要执行其他读操作。在这种情况下，您可以考虑重组表以使其与索引匹配并回收空间。

您还可以重组系统目录表。

由于重组表的时间通常比更新统计信息的时间长，因此您可以执行 RUNSTATS 命令以刷新数据的当前统计信息，然后重新绑定应用程序。如果刷新的统计信息并未提高性能，那么重组可能会有所帮助。

下列因素可能表明需要重组表:

- 对查询所访问的表进行了大量的插入、更新和删除活动。
- 对于使用集群率较高的索引的查询，其性能发生显著变化。
- 执行 RUNSTATS 命令以刷新表统计信息后，性能未得到改善。
- REORGCHK 命令的输出表明需要重组表。

注: 借助 DB2 版本 9.7 修订包 1 和更高版本的发行版，通过重组特定数据分区的数据，只具有分区索引（由系统生成的 XML 路径索引除外）的数据分区表就可以获得更高的数据可用性。分区级别重组将对所指定的数据分区执行表重组，同时使该表的其他数据分区保持可访问。对分区表执行 REORGCHK 命令所获得的输出包含有关执行分区级别重组的统计信息和建议。

可以对数据分区表发出 REORG TABLE 命令和 REORG INDEXES ALL 命令，以便同时重组不同的数据分区或者重组分区中的分区索引。同时重组不同的数据分区或者重组分区的分区索引时，用户可以访问不受影响的分区，但是无法访问受影响的分区。必须满足下列所有条件，才能发出同时对同一个表运行的 REORG 命令:

- 每个 REORG 命令必须对 ON DATA PARTITION 子句指定不同的分区。
- 每个 REORG 命令必须使用 ALLOW NO ACCESS 方式来限制访问数据分区。
- 如果发出 REORG TABLE 命令，那么分区表必须只有分区索引。不能对表定义非分区索引（由系统生成的 XML 路径索引除外）。

选择重组表的方法

可以通过两种方法来重组表: *传统重组*（脱机）和*原位置重组*（联机）。

缺省行为是脱机重组。要指定联机重组操作，请使用 REORG TABLE 命令的 INPLACE 选项。

另一种方法是，使用联机表移动存储过程进行原位置重组。请参阅“使用 ADMIN_MOVE_TABLE 过程以联机方式移动表”。

每种方法都有其优点和缺点，概述如下。选择重组方法时，应考虑哪种方法提供的优点是您优先要解决的方面。例如，如果发生故障时进行恢复比性能更重要，那么最好使用联机重组方法。

脱机重组的优点

此方法具有下列优点:

- 最快速的表重组操作，未包括大对象（LOB）或长字段数据时尤其如此
- 完成后将生成集群情况完美的表和索引
- 在表之后自动重建的索引已被重组；不需要执行单独的步骤来重建索引
- 使用临时表空间来构建影子副本；这降低了包含目标表或索引的表空间的空间需求
- 使用除集群索引以外的索引对数据进行重新集群

脱机重组的缺点

此方法具有下列缺点:

- 对表的访问受限：在重组操作的排序和构建阶段，只能进行读访问
- 正在重组的表的影子副本需要大量空间
- 对重组过程的控制能力较低；脱机重组操作无法暂停和重新启动

联机重组的优点

此方法具有下列优点:

- 除重组操作的截断阶段以外，允许对表进行全面的访问
- 对重组过程的控制能力较高，该过程以异步方式在后台运行，并可以被暂停、继续或停止；例如，如果正在对表运行大量的更新或删除操作，那么您可以暂停进行中的重组操作
- 重组过程在发生故障后可恢复
- 由于以递增方式处理表，因此降低了工作存储空间需求
- 重组的好处将立即体现，甚至可以在重组操作完成前体现

联机重组的缺点

此方法具有下列缺点:

- 数据或索引的集群情况并不完美，这取决于在重组操作期间访问表的事务的类型
- 与脱机重组操作相比，性能欠佳
- 日志记录需求可能较高，这取决于移动的行数、对表定义的索引数以及那些索引的大小
- 由于此过程执行的是索引维护操作而非重建操作，因此可能需要执行后续的索引重组操作

表 1. 联机重组与脱机重组的比较

特征	脱机重组	联机重组
性能	快	慢
完成后数据的集群因子	良好	非最佳集群
并行性（对表的访问）	从不能访问到只读访问	从只读访问到完全访问

表 1. 联机重组与脱机重组的比较 (续)

特征	脱机重组	联机重组
数据存储空间需求	非常大	不是非常大
日志记录存储空间需求	不是非常大	非常大
用户控制（暂停和重新启动重组过程的能力）	控制能力较低	控制能力较高
可恢复性	不可恢复	可恢复
索引重建	进行	不进行
支持所有类型的表	是	否
是否能够指定除集群索引以外的索引	是	否
是否使用临时表空间	是	否

表 2. 联机重组和脱机重组支持的表类型

表类型	是否受脱机重组支持	是否受联机重组支持
多维集群表（MDC）	是[1]	否
范围集群表（RCT）	否[2]	否
追加方式表	否	否[3]
包含长字段或大对象（LOB）数据的表	是[4]	否
系统目录表：SYSIBM.SYSDBAUTH、SYSIBM.SYSROUTINEAUTH、SYSIBM.SYSSEQUENCES 和 SYSIBM.SYSTABLES	是	否

注意:

1. 由于通过 MDC 块索引自动维护集群，所以 MDC 表的重组只涉及空间回收。不能指定索引。
2. RCT 的范围区域始终保持集群。
3. 可以在将追加方式禁用后执行联机重组。
4. 重组长字段或大对象（LOB）数据相当耗时，并且无法提高查询性能；仅当需要回收空间时，才应该重组这些数据。

监视表重组操作的进度

有关当前表重组操作的进度信息将写入历史记录文件。历史记录文件包含每个重组事件的记录。要查看此文件，请对正在重组的表所在的数据库执行 LIST HISTORY 命令。

此外，也可使用表快照来监视表重组操作的进度。无论如何设置数据库系统监视器表开关，系统均会记录表重组监视数据。

如果发生错误，那么 SQLCA 消息将写入历史记录文件。对于原位置表重组操作而言，状态将被记录为 PAUSED。

传统（脱机）表重组

传统表重组使用影子副本方法，从而构建正在重组的表的完整副本。

传统的脱机表重组操作包括四个阶段:

1. SORT - 在此阶段，如果 REORG TABLE 命令中指定了索引，或者已对表定义集群索引，那么首先根据该索引对表行进行排序。如果指定了 INDEXSCAN 选项，那么使用索引扫描方法对表进行排序；否则，使用表扫描排序方法。此阶段仅适用于集群表重组操作。空间回收重组操作从构建阶段开始。
2. BUILD - 在此阶段，将构建整个表的重组副本，此操作在该表的表空间或者 REORG TABLE 命令中指定的临时表空间中执行。
3. REPLACE - 在此阶段，将原始表对象替换为临时表空间中的副本，或者在所重组表的表空间中创建指向新构建的对象的指针。
4. RECREATE ALL INDEXES - 在此阶段，重新创建先前对该表定义的所有索引。

您可以使用快照监视器或快照管理视图来监视表重组操作的进度并确定当前阶段。

脱机方式下的锁定条件的限制性与联机方式更强。构建副本期间，可以对表进行读访问。但是，在将原始表替换为已重组的副本期间以及在重建索引期间，需要对表进行互斥访问。

在整个表重组过程中，需要挂起 IX 表空间锁定。在构建阶段，将获取 U 锁定并对该表挂起该锁定。U 锁定允许锁定所有者更新表中的数据。虽然没有其他应用程序能够更新数据，将允许进行读访问。替换阶段开始后，U 锁定将升级为 Z 锁定。在此阶段，任何其他应用程序都无法访问数据。此锁定将一直挂起到表重组操作完成为止。

脱机重组过程将创建多个文件。这些文件存储在数据库目录中。它们的名称以表空间标识和对象标识为前缀；例如，0030002.ROR 是表空间标识为 3 且表标识为 2 的表重组操作的状态文件。

以下列表提供了脱机表重组操作期间在系统管理的空间（SMS）表空间中创建的临时文件:

- .DTR - 数据影子副本文件
- .LFR - 长字段文件
- .LAR - 长字段分配文件
- .RLB - LOB 数据文件
- .RBA - LOB 分配文件
- .BMR - 多维集群（MDC）表的块对象文件

在索引重组操作期间，将创建以下临时文件:

- .IN1 - 影子副本文件

以下列表提供了排序阶段在系统临时表空间中创建的临时文件:

- .TDA - 数据文件
- .TIX - 索引文件
- .TLF - 长字段文件
- .TLA - 长字段分配文件
- .TLB - LOB 文件

- .TBA – LOB 分配文件
- .TBM – 块对象文件

您不应以手动方式从系统中除去与重组过程相关联的文件。

以脱机方式重组表:

以脱机方式重组表是整理表碎片的最快方法。重组可减少表所需的空间量并提高数据访问和查询性能。

您必须具有 SYSADM、SYSCTRL、SYSMAINT、DBADM 或 SQLADM 权限或者对所要重组的表具有 CONTROL 特权。您还必须建立数据库连接才能重组表。

在标识需要重组的表之后，可以对那些表运行 REORG 实用程序，并且可以选择对那些表的任何索引运行该实用程序。

1. 要使用 REORG TABLE 命令来重组表，只需指定该表的名称。例如:

   ```
   reorg table employee
   ```

   您可以使用特定的临时表空间来重组表。例如:

   ```
   reorg table employee use mytemp
   ```

   您可以重组表并根据特定索引对行进行重新排序。例如:

   ```
   reorg table employee index myindex
   ```

2. 要使用 SQL CALL 语句来重组表，请通过 ADMIN_CMD 过程来指定 REORG TABLE 命令。例如:

   ```
   call sysproc.admin_cmd ('reorg table employee')
   ```

3. 要使用管理应用程序编程接口来重组表，请调用 db2Reorg API。

在重组表之后，请收集有关该表的统计信息，以便优化器掌握最准确的数据来评估查询访问方案。

脱机表重组的恢复:

在替换阶段开始之前，脱机表重组是一个完全成功或完全失败的过程。如果系统在排序或构建阶段崩溃，那么重组操作将回滚，并且不会在崩溃恢复期间重新执行。

如果系统在替换阶段开始之后崩溃，那么重组操作必须完成，这是因为已完成所有重组工作，原始表可能不再可用。在崩溃恢复期间，需要已重组的表对象的临时文件，而不是用于执行排序的临时表空间。恢复操作将完全重新开始替换阶段，并且需要恢复副本对象中的所有数据。在这种情况下，系统管理的空间（SMS）表空间与数据库管理的空间（DMS）表空间之间有一个差别: 必须将 SMS 中已重组的表对象从一个对象复制到另一个对象; 但在 DMS 中，如果在同一个表空间中执行重组，那么只需指向已重组的表对象，并且将删除原始表。将不重建索引，但在崩溃恢复期间会将索引标记为无效。数据库将根据一般规则确定重建索引的时间: 在数据库重新启动时或第一次访问索引时。

如果在索引重建阶段发生崩溃，那么由于新的表对象已存在，因此不会重新执行任何操作。将按先前描述的方式来处理索引。

在前滚恢复期间，如果旧版本的表在磁盘上，那么重新进行重组操作。前滚实用程序使用构建阶段记录的记录标识（RID）来重新应用创建了已重组的表的操作，从而重复构建和替换阶段。将按先前描述的方式来处理索引。仅当最初使用了临时表空间时，已重组的对象的副本才需要临时表空间。在前滚恢复期间，可以同时重新执行多个重组操作（并行恢复）。

提高脱机表重组的性能:

脱机表重组的性能在很大程度上由数据库环境的特征决定。

以 ALLOW NO ACCESS 方式运行的重组操作与以 ALLOW READ ACCESS 方式运行的重组操作在性能方面几乎没有任何差别。差别在于，在 ALLOW READ ACCESS 方式的重组操作期间，实用程序可能必须先等待其他应用程序完成扫描并释放它们的锁定，然后才能替换该表。在以任何一种方式运行的重组操作的索引重建阶段，该表都不可用。

有关提高性能的技巧

- 如果有足够的空间，那么请对原始表和表的重组副本使用同一个表空间，以代替使用临时表空间。这将节省从临时表空间复制已重组的表所需的时间。
- 考虑在重组表之前删除不必要的索引，以便减少重组操作期间需要维护的索引数目。
- 确保正确设置已重组的表所在的表空间的预取大小。
- 调整 **sortheap** 和 **sheapthres** 数据库配置参数，以便控制可用于排序的空间量。由于每个处理器都将执行私有排序，因此 **sheapthres** 的值至少应该是 **sortheap** x *使用的处理器数*。
- 调整页清除程序数，以确保尽快清除缓冲池中的脏索引页。

原位置（联机）表重组

原位置表重组在您能够对表数据进行全面访问的情况下重组该表。数据访问不中断的代价是，表重组操作的运行速度较慢。

在原位置（即联机）表重组操作期间，将按顺序重组表的各个部分。不会将数据复制到临时表空间；而是，在现有表对象中移动行以重新建立集群、回收可用空间并消除溢出行。

联机表重组操作分为 4 个主要阶段:

1. 选择 *n* 页

   在此阶段，数据库管理器将选择由 *n* 页组成的范围，其中 *n* 是至少包含 32 个要进行重组处理的顺序页的扩展数据块的大小。

2. 腾出范围

   REORG 实用程序将此范围内的所有行移至表中的可用页。每个被移动的行都保留一条重组表指针（RP）记录，该记录包含该行的新位置的记录标识（RID）。将行作为包含数据的重组表溢出（RO）记录被放入到表的可用页。此实用程序移动一组行完成后，它将等待所有访问该表中数据的应用程序完成。这些“旧扫描者”在访问表数据时，将使用旧 RID。在等待阶段开始的任何表访问（“新扫描者”）都将使用新 RID 来访问数据。在所有旧扫描者都完成后，REORG 实用程序将清除已移动的行、删除 RP 记录并将 RO 记录转换为常规记录。

3. 填充范围

   腾出特定范围内的所有行之后，将采用已重组的格式、根据先前使用的任何索引进行排序后的顺序并遵循先前定义的任何预取限制写回这些行。重写该范围内的所有页之后，将选择该表中的后续 *n* 个顺序页并重复以上过程。

4. 截断表

   缺省情况下，重组表中的所有页后，缺省情况下将截断表以回收空间。如果指定了 NOTRUNCATE 选项，那么不会截断已重组的表。

联机表重组操作期间创建的文件

联机表重组操作期间，将为每个数据库分区创建一个 .OLR 状态文件。这个二进制文件的名称格式为 xxxxyyyy.OLR，其中 *xxxx* 是表空间标识，*yyyy* 是十六进制的对象标识。此文件包含从暂停状态继续执行联机重组操作所需的下列信息:

- 重组操作的类型
- 正在重组的表的生存日志序号（LSN）
- 要腾出的下一个范围
- 重组操作是为了对数据进行集群还是仅仅为了回收空间
- 正在用于对数据进行集群的索引的标识

将对 .OLR 文件执行校验和检查。如果该文件已损坏并导致校验和错误，或者表 LSN 与生存 LSN 不匹配，那么将启动新的重组操作并创建新的状态文件。

如果 .OLR 状态文件被删除，那么重组过程无法继续，将返回 SQL2219，并且必须启动新的重组操作。

您不应以手动方式从系统中除去与重组过程相关联的文件。

以联机方式重组表:

联机或原位置表重组允许用户在重组表的同时访问该表。

您必须具有 SYSADM、SYSCTRL、SYSMAINT、DBADM 或 SQLADM 权限或者对所要重组的表具有 CONTROL 特权。您还必须建立数据库连接才能重组表。

在标识需要重组的表之后，可以对那些表运行 REORG 实用程序，并且可以选择对那些表的任何索引运行该实用程序。

1. 要使用 REORG TABLE 命令以联机方式重组表，只需指定表名和 INPLACE 选项。例如:

   ```
   reorg table employee inplace
   ```

2. 要使用 SQL CALL 语句以联机方式重组表，请通过 ADMIN_CMD 过程来指定 REORG TABLE 命令。例如:

   ```
   call sysproc.admin_cmd ('reorg table employee inplace')
   ```

3. 要使用管理应用程序编程接口以联机方式重组表，请调用 db2Reorg API。

在重组表之后，请收集有关该表的统计信息，以便优化器掌握最准确的数据来评估查询访问方案。

联机表重组的恢复:

联机表重组的故障通常是由于处理错误（例如磁盘已满或日志记录错误）所致。如果联机表重组失败，那么系统会将 SQLCA 消息写入历史记录文件。

如果运行时发生故障，那么联机表重组操作将暂停，然后在崩溃恢复期间回滚。以后，您可以通过在 REORG TABLE 命令中指定 RESUME 选项来继续执行重组操作。由于联机表重组过程进行全面的日志记录，因此保证可恢复。

在某些情况下，联机表重组操作可能会超出 **num_log_span** 数据库配置参数所设置的限制。在这种情况下，数据库管理器将强制控制 REORG 实用程序并将其置于 PAUSE 状态。在快照监视器输出中，REORG 实用程序的状态将显示为 PAUSED。

联机表重组暂停由中断驱动，这意味着它既可以由用户触发（使用 REORG TABLE 命令的 PAUSE 选项或者 FORCE APPLICATION 命令），也可以由数据库管理器在某些情况（例如系统崩溃）下触发。

如果分区数据库环境中的一个或多个数据库分区遇到错误，那么返回的 SQLCODE 将是来自第一个报告错误的数据库分区的 SQLCODE。

暂停和重新启动联机表重组:

用户可以暂停和重新启动正在进行中的联机表重组。

您必须具有 SYSADM、SYSCTRL、SYSMAINT、DBADM 或 SQLADM 权限或者对要暂停或重新启动联机重组操作的表具有 CONTROL 特权。您还必须建立数据库连接才能暂停或重新启动联机表重组。

1. 要使用 REORG TABLE 命令来暂停联机表重组，请指定表名、INPLACE 选项和 PAUSE 选项。例如:

   ```
   reorg table employee inplace pause
   ```

2. 要重新启动已暂停的联机表重组，请指定 RESUME 选项。例如:

   ```
   reorg table employee inplace resume
   ```

   暂停联机表重组操作之后，无法开始对该表执行新的重组。您必须继续执行或停止已暂停的操作，然后才能开始新的重组过程。

   发出 RESUME 请求之后，重组过程将考虑当前 RESUME 请求所指定的截断选项。例如，如果当前 RESUME 请求未指定 NOTRUNCATE 选项，那么将忽略原始 REORG TABLE 命令（或者任何先前的 RESUME 请求）所指定的 NOTRUNCATE 选项。

   执行复原和前滚操作后，无法恢复表重组操作。

联机表重组的锁定和并行性注意事项:

联机表重组的其中一个重要方面是如何控制锁定，其原因在于，这对应用程序并行性而言至关重要。

联机表重组操作可以挂起下列锁定:

- 为了确保能够对表空间进行写访问，获取对重组操作所影响的表空间的 IX 锁定。
- 在整个重组操作期间获取并挂起表锁定。锁定级别取决于重组期间实施的访问方式:
  - 如果指定了 ALLOW WRITE ACCESS，那么获取 IS 表锁定。

  - 如果指定了 ALLOW READ ACCESS，那么获取 S 表锁定。
- 在截断阶段，将请求对表挂起 S 锁定。在获取 S 锁定之前，并发事务可以插入行。这些插入的行可能不会被 REORG 实用程序检测到，并可能导致表无法被截断。获取 S 表锁定之后，导致表无法被截断的行将被移走，以便能够对表进行压缩。在压缩表之后，会将其截断，但仅当确定所有在截断点访问该表的事务都完成后才会执行此操作。
- 可能会获取行锁定，这取决于表锁定的类型:
  - 如果已对该表挂起 S 锁定，那么不需要单独的行级别 S 锁定，并且不必进一步执行锁定。
  - 如果已对该表挂起 IS 锁定，那么移动行之前将获取 NS 行锁定，并在移动完成后释放该锁定。
- 在联机表重组操作期间，还可能获取某些内部锁定。

锁定会影响联机表重组操作和并发用户应用程序的性能。您可以使用锁定快照数据来帮助您了解联机表重组期间发生的锁定活动。

监视表重组

您可以使用 GET SNAPSHOT 命令、SNAPTAB_REORG 管理视图或 SNAP_GET_TAB_REORG 表函数来获取有关表重组操作的状态信息。

- 要使用 SQL 来访问有关重组操作的信息，请使用 SNAPTAB_REORG 管理视图。 例如，以下查询将返回有关对当前所连接数据库的所有数据库分区执行的表重组操作的详细信息。如果未重组任何表，那么不会返回任何行。

```
select
    substr(tabname, 1, 15) as tab_name,
    substr(tabschema, 1, 15) as tab_schema,
    reorg_phase,
    substr(reorg_type, 1, 20) as reorg_type,
    reorg_status,
    reorg_completion,
    dbpartitionnum
  from sysibmadm.snaptab_reorg
  order by dbpartitionnum
```

- 要使用快照监视器来访问有关重组操作的信息，请使用 GET SNAPSHOT FOR TABLES 命令并检查表重组监视元素的值。

由于脱机表重组操作是同步操作，所以错误将返回给实用程序的调用者（应用程序或命令行处理器）。并且，由于联机表重组操作是异步操作，所以这种情况下不会将错误消息返回给 CLP。要查看执行联机表重组操作期间返回的 SQL 错误消息，请使用 LIST HISTORY REORG 命令。

联机表重组操作作为 db2Reorg 进程在后台运行。即使调用应用程序终止其数据库连接，此进程也将继续运行。

索引重组

表被更新后，索引性能可能会下降。

这种下降表现在下列方面:

- 叶子页碎片化。叶子页碎片化之后，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那些页中键的顺序相匹配，从而产生*不良集群索引*。叶子页出现不良集群情况后，顺序预取操作的效率将降低，I/O 等待数将增加。
- 索引发展到层数过多。在这种情况下，应重组索引。

索引重组要求:

- 对表及其索引具有 SYSADM、SYSMAINT、SYSCTRL、DBADM 或 SQLADM 权限，或者具有 CONTROL 特权
- 存储索引的表空间中的可用空间量等于索引的当前大小。发出 CREATE TABLE 语句时，考虑将索引放在大型表空间中。
- 其他日志空间。索引重组实用程序将记录它自己的活动。

如果在 CREATE INDEX 语句中指定了 MINPCTUSED 选项，那么在某个键被删除且可用空间量变为小于指定的值时，数据库服务器将自动合并索引叶子页。此过程称为*联机索引整理碎片*。

要复原索引集群、释放空间以及降低叶级别，可以使用下列其中一种方法:

- 删除并重新创建索引。
- 使用 REORG TABLE 命令，并指定允许以脱机方式重组表及其索引的选项。
- 使用 REORG INDEXES 命令以联机方式重组索引。在生产环境中，您可以选择此方法，因为这将允许用户在重建表索引期间对表执行读写操作。

联机索引重组

如果使用带有 ALLOW WRITE ACCESS 选项的 REORG INDEXES 命令，那么重建所有基于所指定表的索引时，对该表进行的读写操作将继续进行。在重组期间，将记录对基础表所作的任何将影响索引的更改。重组操作将在重建索引时处理所记录的这些更改。

如果有内部内存缓冲区可供使用，那么对基础表所作的将影响索引的更改还将记录到该空间。内部缓冲区是根据需要从实用程序堆中分配的指定内存区域。使用内存缓冲区空间使索引重组实用程序能够按以下方式处理更改：先直接从内存读取，然后在必要时通过日志读取，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。

不支持对空间索引或多维集群（MDC）表执行 ALLOW WRITE ACCESS 方式（无论是否指定了 CLEANUP ONLY 选项）的联机索引重组。

对于 DB2 V9.7 修订包 1 和更高版本的发行版，对数据分区表使用 REORG INDEXES ALL 命令并使用 ON DATA PARTITION 子句指定某个分区，就会重组单个数据分区的分区索引。在重组索引期间，不受影响的分区将保持可读，而只有受影响的分区才可写。

可以对数据分区表发出 REORG TABLE 命令和 REORG INDEXES ALL 命令，以便同时重组不同的数据分区或者重组分区中的分区索引。同时重组不同的数据分区或者重组分区的分区索引时，用户可以访问不受影响的分区。必须满足下列所有条件，才能发出同时对同一个表运行的 REORG 命令:

- 每个 REORG 命令必须对 ON DATA PARTITION 子句指定不同的分区。
- 每个 REORG 命令必须使用 ALLOW NO ACCESS 方式来限制访问数据分区。

- 如果发出 REORG TABLE 命令，那么分区表必须只有分区索引。不能对表定义非分区索引（由系统生成的 XML 路径索引除外）。

注: REORGCHK 命令的输出包含有关重组索引的统计信息和建议。对于分区表，此输出包含有关重组分区索引和非分区索引的统计信息和建议。

确定何时重组表和索引

对表数据进行大量更改之后，在逻辑上连续的数据可能会存储在不连续的物理数据页中，在许多更新操作创建溢出记录后尤其如此。如果数据以此方式组织，那么数据库管理器必须执行附加的读操作才能访问所需的数据。另外，在删除大量的行之后，也需要执行其他读操作。

表重组操作用于对数据整理碎片，从而避免浪费空间。并且，此操作还将对行进行重新排序以合并溢出记录，从而改进数据访问并最终提高查询性能。您可以指定应该根据特定索引对数据进行重新排序，以便最大程度地减少查询访问数据时执行的读操作的次数。

对表数据进行大量更改会导致索引性能下降。索引叶子页可能会碎片化或者集群程度不佳，并且索引所形成的层数可能会超出为了获得最佳性能而必需的层数。所有这些问题都会导致 I/O 增加以及性能下降。

下列任何一个因素都可能表明应该重组表或索引:

- 自从上次重组表之后，已对该表执行大量的插入、更新和删除活动
- 查询使用集群率较高的索引，但其性能发生显著变化
- 在执行 RUNSTATS 命令以刷新统计信息后，性能未得到改善
- REORGCHK 命令的输出表明可以通过重组表或它的索引来提高性能

  在某些情况下，REORGCHK 实用程序始终建议重组表，即使在执行表重组操作之后也是如此。例如，如果使用 32-KB 页大小，并且平均记录长度为 15 字节，每页最多包含 253 条记录，那么每页有 32700 - (15 x 253) = 28905 个不可用字节。这意味着，大约 88% 的页空间是空闲空间。您应该分析 REORGCHK 实用程序的建议并评估潜在增益与执行重组的成本之间的对比关系。

REORGCHK 命令返回有关数据组织的统计信息，并且可以在是否需要重组特定表或索引这一问题上为您提供建议。但是，通过定期地或者在特定时间对 SYSSTAT 视图运行特定查询，可以构造一个历史记录，这有助于您确定可能对性能产生重大影响的趋势。

要确定是否需要重组表或索引，请查询 SYSSTAT 视图并监视下列统计信息:

- 行的溢出情况

  请查询 SYSSTAT.TABLES 视图中的 OVERFLOW 列以监视溢出值。此值表示在原始页中放不下的行数。当可变长度列导致记录长度扩展到某个程度，以致于某一行在数据页上分配给它的位置中放不下时，该行的数据将溢出。对表添加列时，行长度也会更改。在这种情况下，在该行中的原始位置将保留一个指针，而实际值存储在由该指针指示的另一位置。这可能会影响性能，因为数据库管理器必须根据该指针来查找该列的内容。这个包括两个步骤的过程增加了处理时间，并且还可能增加所需执行的 I/O 次数。重组表数据将消除任何行溢出情况。
- 访存统计信息

查询 SYSSTAT.INDEXES 目录视图中的下列各列，以便确定按索引顺序访问表时预取程序的效率。这些统计信息体现对基础表执行预取程序时的平均性能特征。

- AVERAGE_SEQUENCE_FETCH_PAGES 列存储可以按顺序访问的平均页数。可以按顺序访问的页适合于预取。数目较小表明预取程序未充分发挥作用，原因是它们无法读入由表空间的 PREFETCHSIZE 值指定的所有页数。较大的数指示预取程序将有效地执行。对于集群索引和表，此数目应该接近 NPAGES（包含一些行的页数）的值。
- AVERAGE_RANDOM_FETCH_PAGES 列存储使用索引来访存表行时，在两次顺序页访问之间访存的随机表页的平均数目。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。这种混乱情况通常是由于在表末尾或溢出页中执行无序的插入所致，当使用索引来访问某个范围的值时，查询性能将受影响。
- AVERAGE_SEQUENCE_FETCH_GAP 列存储使用索引访存表行时表页序列之间的平均间隔。此间隔是通过扫描索引叶子页进行检测的，每个间隔都表示在各个表页序列之间必须随机访存的表页的平均数目。随机访问许多页时就会发生这种情况，从而中断预取程序。数目较大表明表比较混乱或者对于索引而言集群情况较差。

- 包含已被标记为“已删除”但尚未被除去的记录标识（RID）的索引叶子页的数目

  将 RID 标记为“已删除”时，通常不会以物理方式将其删除。这意味着，可用空间可能会被这些在逻辑上已被删除的 RID 占用。要检索其中每个 RID 都被标记为“已删除”的叶子页的数目，请查询 SYSSTAT.INDEXES 视图的 NUM_EMPTY_LEAFS 列。对于并非其中所有 RID 都被标记为“已删除”的叶子页，在逻辑上已被删除的 RID 的总数存储在 NUMRIDS_DELETED 列中。

  使用此信息来估算通过调用带有 CLEANUP ALL 选项的 REORG INDEXES 命令可以回收的空间量。如果您只想回收其中所有 RID 都被标记为“已删除”的页中的空间，那么请调用带有 CLEANUP ONLY PAGES 选项的 REORG INDEXES 命令。

- 索引的集群比率和集群因子统计信息

  通常，表只有一个索引可以具有较高的集群度。集群比率统计信息存储在 SYSCAT.INDEXES 目录视图的 CLUSTERRATIO 列中。此值介于 0 与 100 之间，它表示索引中的数据集群程度。如果收集详细的索引统计信息，那么在 CLUSTERFACTOR 列中将存储较详细的集群因子统计信息（介于 0 与 1 之间），并且 CLUSTERRATIO 的值为 -1。在这两种集群统计信息中，只有一种可以记录在 SYSCAT.INDEXES 目录视图中。要将 CLUSTERFACTOR 值与 CLUSTERRATIO 值进行比较，请将 CLUSTERFACTOR 值乘以 100 以获得一个百分比值。

  在集群比率较高的情况下，并非只访问索引的索引扫描可能执行得更好。较低的集群比率将使此类扫描执行更多的 I/O，这是因为再次访问数据页时，该数据页仍在缓冲池中的可能性较小。增大缓冲区大小可以提高集群程度不高的索引的性能。

  如果表数据最初根据某个索引进行集群，并且集群统计信息表明数据现在的集群情况相对于该索引而言并不好，那么您可能想重组该表以便再次对数据进行集群。

- 叶子页的数目

  请查询 SYSCAT.INDEXES 视图中的 NLEAF 列，以便确定索引所占用的叶子页的数目。此数目指示对索引进行完整扫描所需执行的索引页 I/O 次数。

理想情况下，索引所占用的空间量应该尽可能少，以便减少索引扫描所需执行的 I/O 次数。随机的更新活动会导致页分割，从而增大索引大小。在表重组操作期间，可以重建每个索引并使其占用的空间量最少。

缺省情况下，构建索引时，将在每个索引页中保留 10% 的可用空间。要增加可用空间量，请在创建索引时指定 PCTFREE 选项。每当重组索引时，都将使用指定的 PCTFREE 值。大于 10% 的可用空间值可以降低索引重组频率，这是因为，有额外的空间可以容纳所插入的附加索引内容。

- 空数据页的数目

  要计算表中的空页数，请查询 SYSCAT.TABLES 视图中的 FPAGES 和 NPAGES 列，然后将 FPAGES 值（使用中的总页数）减去 NPAGES 值（包含行的页数）。空页可能是由于整个范围内的行被删除而产生。

  随着空页数的增加，就需要进行表重组。重组表时，将回收空页并减少表所使用的空间量。另外，因为表扫描期间会将空页读入缓冲池，所以回收未使用的页可以提高扫描性能。

  如果表的使用中的总页数（FPAGES）小于或等于 (NPARTITIONS * 1 个扩展数据块大小)，那么建议您不要进行表重组。对于分区表，NPARTITIONS 表示数据分区数；否则，它的值是 1。在分区数据库环境中，如果 FPAGES <= (表的数据库分区组中的数据库分区数) * (NPARTITIONS * 1 个扩展数据块大小)，那么建议您不要进行表重组。

在重组表或索引之前，请考虑查询性能不断降低所浪费的成本与重组表或索引所需的成本（其中包括处理时间、耗用时间和并行性降低）哪个更高，以确定是否重组。

重组表和索引的成本

决定是否要重组表或索引时，必须考虑对这些对象执行重组所产生的开销。

重组表和索引的成本包括:

- 执行实用程序所需的处理时间。
- 运行 REORG 实用程序时，锁定将导致并行性下降。
- 需要额外的存储器。
  - 脱机表重组需要较多的存储空间来存放表的影子副本。
  - 联机或原位置表重组需要的日志空间较多。
  - 脱机索引重组需要的日志空间较少，并且不涉及影子副本。
  - 联机索引重组需要的日志空间较多，并且需要较多的存储空间来存放索引的影子副本。

在某些情况下，已重组的表可能比原始表要大。在下列情况下，重组后的表可能会增大:

- 在使用索引来确定行顺序的集群重组表操作中，如果表记录的长度可变，那么可能需要更多空间，这是因为在重组后的表中，某些页中包含的行数可能比原始表中的行数少。
- 在上次重组之后，已增加在每一页上留下的可用空间量（由 PCTFREE 值表示）。

脱机表重组的空间需求

由于脱机重组使用影子副本方法，因此需要足够的附加存储器来容纳表的另一个副本。将在原始表所在的表空间或用户指定的临时表空间中构建影子副本。

如果使用表扫描排序方法，那么可能需要其他临时表空间存储器来进行排序处理。需要的其他空间可能与要重组的表一样大。如果集群索引具有系统管理的空间（SMS）类型或唯一的数据库管理的空间（DMS）类型，那么重新创建此索引时，不需要进行排序。但是，将通过扫描刚刚重组的数据来重建此索引。重新创建的任何其他索引都需要进行排序，并可能需要大小可达所重组表的大小的临时空间。

脱机表重组操作生成的控制日志记录很少，因此耗用的日志空间量相对较少。如果 REORG 实用程序未使用索引，那么将只创建表数据日志记录。如果指定了索引，或者已对该表定义集群索引，那么将按照记录标识（RID）放入新版本的表的顺序来记录这些 RID。每个 RID 日志记录最多包含 8000 个 RID，每个 RID 耗用 4 个字节。这可能加剧脱机表重组操作期间的日志空间问题。请注意，仅当数据库可恢复时，才会记录 RID。

联机表重组的日志空间需求

联机表重组操作所需的日志空间通常比脱机表重组操作所需的日志空间要多。需要的空间量由所要重组的行数、索引数、索引键的大小以及开始时表的组织情况决定。一种不错的做法是，为表的日志空间耗用情况确定一个典型的基准。

表中的每一行都有可能在联机表重组操作期间被移动两次。对于每个索引，每个表行都必须更新索引键以反映新位置，并且在完成所有对旧位置的访问后，将再次更新索引键以除去对旧 RID 的引用。移回行时，将再次执行对索引键的更新。系统将记录所有这些活动，以使联机表重组操作完全可恢复。假定存在一个索引，对于每一行，最少有两个数据日志记录（每个都包含行数据）和四个索引日志记录（每个都包含键数据）。集群索引尤其容易填满索引页，从而导致索引分割和合并，这些分割和合并活动也必须进行记录。

由于联机表 REORG 实用程序发出频繁的内部 COMMIT 语句，因此通常不会产生大量活动日志。截断阶段的情况例外，在此阶段，此实用程序将请求获取 S 表锁定。如果此实用程序无法获取该锁定，那么它将等待，其他事务在此期间可能会迅速填满日志。

减少重组表和索引的需要

您可以使用不同的策略来减少重组表和索引的需要以及相关联的成本。

减少重组表的需要

要减少重组表的需要:

- 使用多分区表。
- 创建多维集群（MDC）表。对于 MDC 表，将对您在 CREATE TABLE 语句的 ORGANIZE BY DIMENSIONS 子句中指定的列维护集群。但是，如果 REORGCHK 实用程序确定存在太多未使用的块或应该对块进行压缩，那么可能仍会建议重组 MDC 表。
- 对表启用 APPEND 方式。例如，如果新行的索引键值总是新的大键值，那么表的集群属性会尝试将其放在表的末尾。在这种情况下，启用 APPEND 方式可能优于使用集群索引。

为了进一步减少重组表的需要，请在创建表后执行下列任务:

- 对该表进行更改，以指定装入或表重组操作期间要在每一页中保留的可用空间所占的百分比（PCTFREE）
- 创建集群索引并指定 PCTFREE 选项
- 将数据装入到表中之前对数据进行排序

执行这些任务后，表的集群索引和 PCTFREE 设置有助于保持原来的排序顺序。如果表页中有足够的空间，那么可以将新数据插入正确的页，以保持索引的集群特征。但是，随着越来越多的数据被插入，表页会因此而变满，记录会被追加至表的末尾，因而表将逐渐失去集群特性。

如果在创建集群索引后执行表重组操作或者执行排序和装入操作，那么索引将尝试维护数据的顺序，这将改善 RUNSTATS 实用程序所搜集的 CLUSTERRATIO 或 CLUSTERFACTOR 统计信息。

减少重组索引的需要

要减少重组索引的需要:

- 创建集群索引，并指定 PCTFREE 或 LEVEL2 PCTFREE 选项。
- 创建索引并指定 MINPCTUSED 选项。此外，请考虑使用 REORG INDEXES 命令的 CLEANUP ONLY ALL 选项来合并叶子页。

自动重组

对表数据进行许多更改后，表及其索引可能会碎片化。逻辑上按顺序排列的数据可能会驻留在非顺序页中，从而导致数据库管理器必须执行附加的读操作才能访问数据。

RUNSTATS 实用程序收集的统计信息将指示数据在表中的分布情况。通过分析这些统计信息可以知道何时需要执行哪种类型的重组。

自动重组过程通过使用 REORGCHK 实用程序提供的公式来确定是否需要执行表或索引重组。它会定期评估已经更新了统计信息的表和索引，以便了解是否需要重组并在有必要执行这些操作时进行安排。

可以通过 **auto_reorg**、**auto_tbl_maint** 和 **auto_maint** 数据库配置参数来启用或禁用自动重组功能部件。

在分区数据库环境中，自动重组操作的启动在主目录数据库分区中进行，因此只需要对该分区启用这些配置参数。但是，将在目标表所在的所有数据库分区中运行重组操作。

如果您不太确定何时以及如何重组表和索引，那么可以将自动重组作为整个数据库维护方案的一部分。

您还可以对多维集群（MDC）表进行重组，以便回收空间。只有 DMS 表空间中的 MDC 表才支持释放 MDC 表中的扩展数据块。可以在数据库的自动维护活动期间释放 MDC 表中的扩展数据块。

自动重组数据分区表

对于 DB2 版本 9.7 修订包 1 和更低版本的发行版，自动重组功能支持重组整个表的数据分区表。对于 DB2 版本 9.7 修订包 1 和更高版本的发行版，自动重组功能支持重组分区表的数据分区以及重组分区表的数据分区的分区索引。

为了避免使整个数据分区表进入 ALLOW NO ACCESS 方式，自动重组功能将在数据分区级别对需要重组的分区索引执行 REORG INDEXES ALL 操作。自动重组功能将对任何需要重组的非分区索引执行 REORG INDEX 操作。

自动重组功能将对数据分区表执行下列 REORG TABLE 操作:

- 如果对表定义了任何非分区索引（系统生成的 XML 路径索引除外），并且只有一个分区需要重组，那么自动重组功能将通过使用 ON DATA PARTITION 子句指定需要重组的分区，然后执行 REORG TABLE 操作。否则，在未使用 ON DATA PARTITION 子句的情况下，自动重组功能将对整个表执行 REORG TABLE 操作。
- 如果没有对表定义任何非分区索引（系统生成的 XML 路径索引除外），那么自动重组功能将通过使用 ON DATA PARTITION 子句指定需要重组的每个分区的数据分区，然后执行 REORG TABLE 操作。

允许自动重组表和索引

使用自动化的表和索引重组功能，以使您无需担忧何时使用何种方法来重组数据。

为了进行高效率的数据访问和实现最佳工作负载性能，具有组织良好的表和索引数据极为关键。在执行大量插入、更新和删除操作之后，逻辑上按顺序排列的表数据可能会驻留在非顺序数据页中，从而导致数据库管理器必须执行附加的读操作才能访问数据。此外，在访问已从中删除大量行的表中的数据时，也需要执行附加的读操作。您可以允许 DB2 服务器重组系统目录表以及用户表。

要允许对数据库进行自动重组，请将下列配置参数设置为 ON:

- **auto_maint**
- **auto_tbl_maint**
- **auto_reorg**

应用程序设计

数据库应用程序设计是其中一项将会影响应用程序性能的因素。请查看本节的内容，以了解有关可以帮助您最大程度地提高数据库应用程序性能的应用程序设计注意事项的详细信息。

应用程序进程、并行性与恢复

所有 SQL 程序都作为*应用程序进程*或代理程序的组成部分执行。应用程序进程涉及执行一个或多个程序，并且是数据库管理器分配资源和锁定时面向的单元。不同的应用程序进程可能涉及执行不同的程序或者同一个程序的不同执行。

多个应用程序进程可能会同时请求访问同一数据。*锁定*机制用于在此类情况下维护数据完整性，例如，防止两个应用程序进程同时更新同一行数据。

数据库管理器获取锁定，以防一个应用程序进程所作的未落实更改意外地被任何其他进程察觉。数据库管理器在应用程序进程结束时释放所有为该进程获取并保留的锁

定。但是，应用程序进程可以显式地请求更快地释放该锁定。此任务通过*落实*操作完成，该操作将释放工作单元期间获取的锁定，并且还将落实在该工作单元期间进行的数据库更改。

工作单元（UOW）是应用程序进程中可恢复的操作序列。工作单元在应用程序进程启动时或者先前 UOW 由于除应用程序进程终止以外的原因而结束时启动。执行落实操作、执行回滚操作或者应用程序进程结束时，工作单元将结束。落实或回滚操作只影响该操作正在结束的 UOW 中进行的数据库更改。

数据库管理器提供了一种方法来撤销由应用程序进程进行的未落实更改。如果应用程序进程的其中一部分发生故障，或者发生死锁或锁定超时情况，那么可能有必要撤销未落实的更改。应用程序进程可以显式地请求取消它所作的数据库更改。此任务通过*回滚*操作完成。

只要这些更改保持处于未落实状态，它们就不会被其他应用程序进程看到，并且可以回滚。但是，如果实施中的隔离级别是“未落实的读”（UR），那么情况并非如此。这些数据库更改在落实之后，它们可供其他应用程序进程访问，并且不再可以回滚。

DB2 调用级接口（CLI）和嵌入式 SQL 都支持称为*并发事务*的连接方式，该方式支持多个连接，并且每个连接都是一个独立的事务。一个应用程序可以有多个与同一数据库的并发连接。

数据库管理器为应用程序进程获取的锁定将挂起到 UOW 结束为止，除非隔离级别是“游标稳定性”（即 CS，在这种情况下，锁定在游标从一行移至另一行时被释放）或“未落实的读”（UR）。

应用程序进程从来不会由于自己的锁定而导致无法执行操作。但是，如果一个应用程序使用并发事务，那么一个事务的锁定可能会影响并发事务的操作。

UOW 的启动和终止定义了应用程序进程中的一致点。例如，银行事务可能涉及将资金从一个帐户转到另一帐户。这样的事务要求从第一个帐户中扣除这些资金，然后将其加入第二个帐户。执行扣除步骤之后，数据变得不一致。只有将资金加入第二个帐户之后，才重新进入一致状态。这两个步骤都完成后，可以使用落实操作来结束 UOW，从而使这些更改对其他应用程序进程可视。如果在 UOW 结束前发生故障，那么数据库管理器将回滚任何未落实的更改以恢复数据一致性。

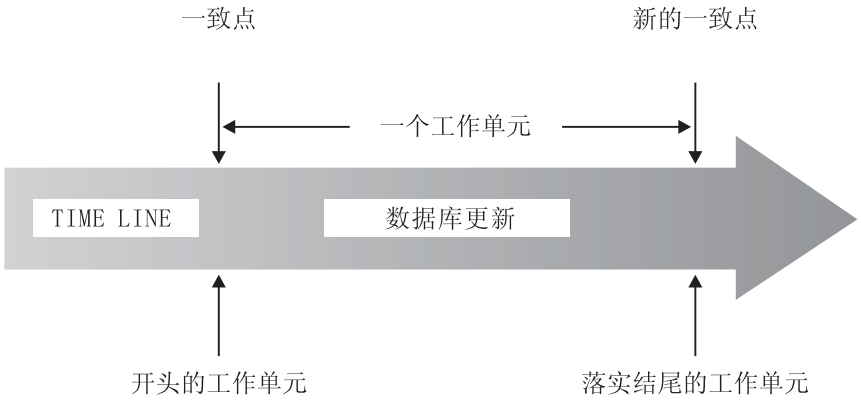


图 21. 包含 COMMIT 语句的工作单元

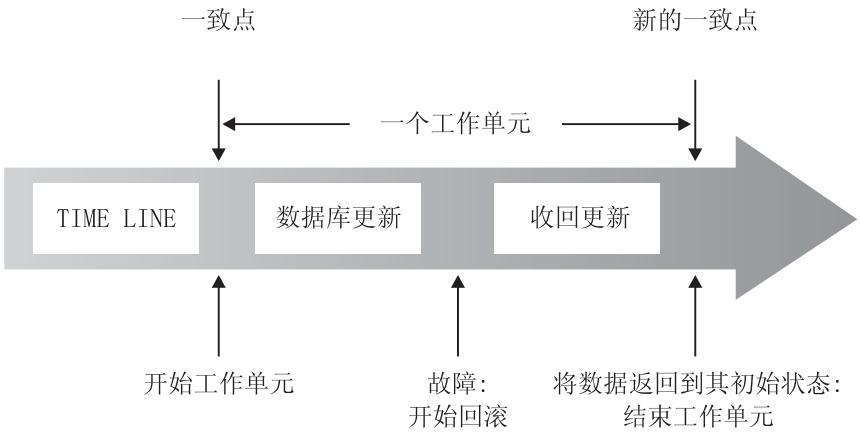


图 22. 包含 ROLLBACK 语句的工作单元

并行性问题

因为许多用户访问和更改关系数据库中的数据，所以数据库管理器必须允许用户进行这些更改并确保数据完整性。

*并行性*是指可以同时由多个交互式用户或应用程序共享资源。数据库管理器通过控制此访问来防止出现意外的后果，例如:

- **丢失更新**。两个应用程序 A 和 B 可能同时读取同一行并根据这些应用程序读取的数据来计算其中一列的新值。如果 A 更新该行，而 B 随后也更新该行，那么 A 所作的更新将丢失。
- **访问未落实的数据**。应用程序 A 可能更新某个值，而应用程序 B 可能在该值被落实前读取该值。接着，如果 A 撤销该更新，那么 B 执行的计算可能基于无效的数据。
- **不可重复读**。应用程序 A 可能在处理其他请求前读取某一行。与此同时，B 修改或删除该行并落实更改。以后，如果 A 试图再次读取原始行，那么它将看到已修改的行或发现原始行已被删除。
- **幻像读**。应用程序 A 可能执行一个查询，该查询根据某个搜索条件读取一组行。应用程序 B 插入新数据或更新现有数据，以满足应用程序 A 的查询。应用程序 A 在同一个工作单元中再次执行它的查询，这将返回一些附加的“幻像”值。

对于全局临时表而言，不会发生并行性问题，原因是这些表仅可供声明或创建它们的应用程序使用。

联合数据库系统中的并行控制

*联合数据库系统*支持应用程序和用户提交在单个语句中引用两个或更多个数据库管理系统（DBMS）的 SQL 语句。为了引用这样的数据源（每个数据源都由 DBMS 和数据组成），DB2 服务器使用昵称。*昵称*是其他 DBMS 中对象的别名。在联合系统中，DB2 服务器依赖于主管所请求数据的数据库管理器的并行控制协议。

DB2 联合系统为数据库对象提供*位置透明性*。例如，如果关于表和视图的信息已移动位置，那么可以更新通过昵称对该信息的引用，而无需更改请求获取此信息的应用程序。当应用程序通过昵称来访问数据时，DB2 服务器依靠数据源的并行控制协议来确保强制实施隔离级别。尽管 DB2 服务器尝试将数据源上请求的隔离级别与等同的逻辑级

别匹配，但是，结果可能因数据源功能不同而有所变化。

隔离级别

与应用程序进程相关联的*隔离级别*确定该进程所访问的数据被锁定或者与其他同时执行的进程相隔离的程度。该隔离级别在工作单元运行期间生效。

因此，应用程序进程的隔离级别指定:

- 应用程序读取或更新的行可供其他同时执行的应用程序进程使用的程度
- 其他同时执行的应用程序进程的更新活动可以影响应用程序的程度

静态 SQL 语句的隔离级别是作为程序包的属性指定的，并且将应用于使用该程序包的应用程序进程。要指定隔离级别，请在程序准备过程中设置 ISOLATION 绑定或预编译选项。对于动态 SQL 语句而言，缺省隔离级别是对准备该语句的程序包指定的隔离级别。请使用 SET CURRENT ISOLATION 语句对会话中发出的动态 SQL 语句指定另一隔离级别。有关更多信息，请参阅“CURRENT ISOLATION 专用寄存器”。对于静态 SQL 语句和动态 SQL 语句而言，*select-statement* 中的 *isolation-clause* 都将覆盖专用寄存器（如果已设置的话）和绑定选项值。有关更多信息，请参阅“SELECT 语句”。

隔离级别由锁定实施，所使用的锁定的类型将限制或阻止并发应用程序进程访问数据。已声明临时表以及它们的行无法被锁定，这是因为它们仅可供声明它们的应用程序访问。

数据库管理器支持三种一般类别的锁定:

共享（S）
挂起 S 锁定之后，并发应用程序进程只能对数据执行只读操作。

更新（U）
挂起 U 锁定之后，如果并发应用程序进程未声明它们要更新行，那么它们只能对数据执行只读操作。数据库管理器假定当前正在查看行的进程可能会更新该行。

互斥（X）
挂起 X 锁定之后，并发应用程序进程将无法以任何方式访问数据。这不适用于隔离级别为“未落实的读”（UR）的应用程序进程，这些进程能够读取但无法修改数据。

无论采用哪种隔离级别，数据库管理器都将对插入、更新或删除的每一行挂起互斥锁定。因此，所有隔离级别都将确保应用程序进程在工作单元运行期间更改的任何行在该工作单元完成前不会被任何其他应用程序进程更改。

数据库管理器支持四种隔离级别。

- 第 116 页的『可重复读（RR）』
- 第 116 页的『读稳定性（RS）』
- 第 117 页的『游标稳定性（CS）』
- 第 117 页的『未落实的读（UR）』

注: 某些主机数据库服务器支持*不落实（NC）*隔离级别。对于其他数据库服务器而言，此隔离级别的行为与“未落实的读”隔离级别相同。

下面是每种隔离级别的详细描述，这些描述按隔离级别对性能的影响程度的降序排列，但按您访问或更新数据时需要加以关注的程度的升序排列。

可重复读（RR）

*可重复读*隔离级别将锁定应用程序在工作单元（UOW）运行期间引用的所有行。如果应用程序在同一个工作单元中发出 SELECT 语句两次，那么每次将返回相同的结果。对于 RR 而言，不可能出现丢失更新、访问未落实的数据、不可重复读以及幻像读等情况。

在 RR 隔离级别下，应用程序在 UOW 完成前可以任意次地检索和处理行。但是，在该 UOW 完成之前，其他应用程序均无法更新、删除或插入将会影响结果集的行。在 RR 隔离级别下运行的应用程序无法看到其他应用程序所作的未落实更改。此隔离级别确保返回的所有数据在被应用程序看到前保持不变，即使使用了临时表或行分块方法也是如此。

所引用的每一行都将被锁定，而不仅仅是锁定所检索的行。例如，如果扫描 10000 行并对它们应用谓词，尽管可能只有 10 行满足条件，但仍会锁定全部的 10000 行。其他应用程序无法插入或更新再次执行查询时将被添加到该查询所引用的行列表中的行。这将防止出现幻像读情况。

由于 RR 可能会获取相当多的锁定，所以此数目可能会超出 **locklist** 和 **maxlocks** 数据库配置参数所指定的限制。为了避免锁定升级，在有可能发生锁定升级的时候，优化器可能会选择获取单个表级别锁定用于索引扫描。如果您不希望进行表级别锁定，那么请使用“读稳定性”隔离级别。

评估引用约束时，DB2 服务器有时会将用于外表扫描的隔离级别升级到 RR，而不考虑用户先前设置的隔离级别。这将导致其他锁定一直被挂起到落实为止，从而增加发生死锁或锁定超时情况的可能性。为了避免这些问题，请创建只包含外键列的索引以供引用完整性扫描使用。

读稳定性（RS）

*读稳定性*隔离级别只锁定应用程序在工作单元运行期间检索的那些行。RS 确保在 UOW 完成之前，在该 UOW 运行期间读取的任何合格行不会被其他应用程序进程更改，并确保任何由另一个应用程序进程更改的行在该进程落实更改前无法被读取。对于 RS 而言，不可能出现访问未落实的数据以及不可重复读等情况。但是，有可能进行幻像读。

此隔离级别确保返回的所有数据在被应用程序看到前保持不变，即使使用了临时表或行分块方法也是如此。

RS 隔离级别既提供了高度的并行性，也提供了稳定的数据视图。所以，优化器确保在发生锁定升级前不获取表级别锁定。

RS 隔离级别适合于符合下列条件的应用程序:

- 在并发环境中运行
- 要求合格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同一个查询，或者在一个工作单元中多次发出同一个查询时并不要求结果集相同

游标稳定性（CS）

*游标稳定性*隔离级别将在游标定位于事务执行期间所访问的任何行上时锁定该行。此锁定在下一行被访存或者事务终止之前将保持有效。但是，如果更改了该行中的任何数据，那么在落实更改之前将一直挂起该锁定。

在此隔离级别下，其他应用程序无法在可更新游标定位于某一行上时更新或删除该行。在 CS 下，无法访问其他应用程序未落实的数据。但是，有可能进行不可重复读和幻像读。

CS 是缺省隔离级别。如果您希望最大程度地提高并行性，并且只需要查看已落实的数据时，此隔离级别适用。

注: 在版本 9.7 所引入的*当前已落实*语义下，将像以前那样只返回已落实的数据，但读取者现在不等待更新者释放行锁定。而是，读取者返回基于当前已落实版本的数据；即，写操作启动前的数据。

未落实的读（UR）

*未落实的读*隔离级别允许应用程序访问其他事务未落实的更改。并且，UR 不会阻止其他应用程序访问正被读取的行，除非该应用程序尝试更改或删除表。

在 UR 下，可能会出现访问未落实的数据、不可重复读以及幻像读等情况。如果对只读的表运行查询，或者只发出 SELECT 语句并且查看其他应用程序尚未落实的数据不会引起问题时，此隔离级别适用。

对于只读游标和可更新游标，UR 的工作方式有所不同。

- 只读游标可访问其他事务未落实的大部分更改。
- 在事务处理期间，正由其他事务创建或删除的表、视图和索引不可用。其他事务进行的任何其他更改在落实或回滚前都可被读取。在 UR 下，可更新游标的工作方式就像隔离级别为 CS 一样。

如果未落实的读应用程序使用了模糊游标，那么它可以在运行期间使用 CS 隔离级别。如果 PREP 或 BIND 命令的 BLOCKING 选项值为 UNAMBIG（缺省值），那么模糊游标可以升级为 CS。要避免这种升级，请执行下列操作:

- 将该应用程序中的游标修改为明确游标。将 SELECT 语句更改为包括 FOR READ ONLY 子句。
- 保留应用程序中的模糊游标，但对该程序进行预编译或者对其进行绑定并指定 BLOCKING ALL 和 STATICREADONLY YES 选项，以便允许该程序运行时将模糊游标视为只读游标。

隔离级别的比较

表 3 对受支持的隔离级别作了摘要。

表 3. 隔离级别的比较

	UR	CS	RS	RR
应用程序是否能够看到其他应用程序进程未落实的更改？	是	否	否	否

表 3. 隔离级别的比较 (续)

	UR	CS	RS	RR
应用程序是否能够更新其他应用程序进程未落实的更改？	否	否	否	否
重新执行语句时是否会被其他应用程序进程影响？[1]	是	是	是	否[2]
已更新的行是否能够被其他应用程序进程更新？[3]	否	否	否	否
已更新的行是否能够被其他在除 UR 以外的隔离级别下运行的应用程序进程读取？	否	否	否	否
已更新的行是否能够被其他在 UR 隔离级别下运行的应用程序进程读取？	是	是	是	是
已访问的行是否能够被其他应用程序进程更新？[4]	是	是	否	否
已访问的行是否能够被其他应用程序进程读取？	是	是	是	是
当前行是否能够被其他应用程序进程更新或删除？[5]	是/否[6]	是/否[6]	否	否

注:

1. *幻像读现象*的一个示例如下所示: 工作单元 UW1 读取 *n* 行满足特定搜索条件的数据。工作单元 UW2 插入一行或多行满足同一搜索条件的数据，然后执行落实。接着，如果 UW1 再次执行读操作并指定相同的搜索条件，那么它将看到另一个结果集: 先前读取的行以及 UW2 插入的行。
2. 如果基于标号的访问控制（LBAC）凭证在两次读操作之间发生更改，那么第二次读操作的结果可能会因为您访问不同的行而有所变化。
3. 如果应用程序既读取表也写表，那么此隔离级别无法为该应用程序提供保护。例如，应用程序对一个表打开一个游标，然后对该表执行插入、更新或删除操作。使用打开的游标来访存更多的行时，该应用程序可能会看到不一致的数据。
4. *不可重复读现象*的一个示例如下所示: 工作单元 UW1 读取某一行。工作单元 UW2 修改该行，然后执行落实。接着，如果 UW1 再次读取该行，那么它可能会看到另一个值。
5. *脏读现象*的一个示例如下所示: 工作单元 UW1 修改某一行。工作单元 UW2 在 UW1 执行落实前读取该行。以后，如果 UW1 回滚更改，那么 UW2 所读取的将是不存在的数据。
6. 在 UR 或 CS 下，如果游标不可更新，那么当前行在某些情况下可以被其他应用程序进程更新或删除。例如，如果执行缓存，那么可能会导致客户机上的当前行与服务器上的当前行不同。并且，在 CS 下使用"当前已落实"语义时，正在读取的行可能带有处于暂挂状态的未落实更新。在这种情况下，始终将该行的当前已落实版本返回给应用程序。

隔离级别摘要

表 4 列示了与不同隔离级别相关联的并行性问题。

表 4. 隔离级别摘要

隔离级别	访问未落实的数据	不可重复读	幻像读
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

因为获取和释放锁定所需的处理和内存资源随隔离级别的不同而有所变化，所以隔离级别不但影响应用程序之间的隔离程度，而且还影响各个应用程序的性能特征。发生死锁的可能性也随隔离级别的不同而有所变化。表 5 提供了简单的试探方法，可以帮助您为应用程序选择初始隔离级别。

表 5. 选择隔离级别的准则

应用程序类型	要求数据高度稳定	不要求数据高度稳定
读写事务	RS	CS
只读事务	RR 或 RS	UR

指定隔离级别

因为隔离级别确定访问数据时如何使该数据不受其他进程影响，所以您应该选择在并行性需求与数据完整性需求之间进行平衡的隔离级别。

您指定的隔离级别在工作单元（UOW）运行期间生效。使用下列试探方法来确定编译 SQL 或 XQuery 语句时将使用的隔离级别:

- 对于静态 SQL:
  - 如果在语句中指定了 *isolation-clause*，那么使用该子句的值。
  - 如果在语句中未指定 *isolation-clause*，那么使用将程序包与数据库绑定时对该程序包指定的隔离级别。
- 对于动态 SQL:
  - 如果在语句中指定了 *isolation-clause*，那么使用该子句的值。
  - 如果在语句中未指定 *isolation-clause*，并且已在当前会话中发出 SET CURRENT ISOLATION 语句，那么使用 CURRENT ISOLATION 专用寄存器的值。
  - 如果在语句中未指定 *isolation-clause*，并且在当前会话中未发出 SET CURRENT ISOLATION 语句，那么使用将程序包与数据库绑定时对该程序包指定的隔离级别。
- 对于静态或动态 XQuery 语句，环境的隔离级别确定对 XQuery 表达式进行求值时使用的隔离级别。

注: 许多以商用为目的编写的应用程序提供了用于选择隔离级别的方法。有关信息，参阅应用程序文档。

您可以通过多种不同的方法来指定隔离级别。

- **在语句级别:**

  注: 无法在语句级别指定 XQuery 语句的隔离级别。

  使用 WITH 子句。不能在子查询中使用 WITH 子句。WITH UR 选项只适用于只读操作。在其他情况下，语句将自动由 UR 更改为 CS。

  此隔离级别覆盖对语句所在的程序包指定的隔离级别。您可以对下列 SQL 语句指定隔离级别:

  - DECLARE CURSOR
  - 搜索型 DELETE
  - INSERT

- SELECT
- SELECT INTO
- 搜索型 UPDATE

- **对于当前会话中的动态 SQL**:

  请使用 SET CURRENT ISOLATION 语句对会话中发出的动态 SQL 语句设置隔离级别。发出此语句将对 CURRENT ISOLATION 专用寄存器设置一个值，该值用来指定当前会话中发出的任何动态 SQL 语句的隔离级别。一旦设置 CURRENT ISOLATION 专用寄存器，此寄存器就会对会话中编译的任何后续动态 SQL 语句设置隔离级别，而不考虑该语句由哪个程序包发出。此隔离级别将一直生效到会话结束或者发出 SET CURRENT ISOLATION...RESET 语句为止。

- **在预编译或绑定时**:

  对于使用受支持的编译型语言编写的应用程序，请使用 PREP 或 BIND 命令的 ISOLATION 选项。此外，也可以使用 sqlaprep 或 sqlabndx API 来指定隔离级别。

  - 如果在预编译时创建绑定文件，那么隔离级别存储在绑定文件中。如果在绑定时未指定隔离级别，那么缺省值是预编译期间使用的隔离级别。
  - 如果未指定隔离级别，那么将使用缺省级别“游标稳定性”。

  要确定程序包的隔离级别，请执行以下查询:

  ```
  select isolation from syscat.packages
    where pkgname = 'pkgname'
      and pkgschema = 'pkgschema'
  ```

  其中 *pkgname* 是程序包的未限定名称，而 *pkgschema* 是程序包的模式名。这两个名称都必须以大写字符指定。

- **在运行时使用 JDBC 或 SQLJ 时**:

  注: JDBC 和 SQLJ 通过 DB2 服务器上的 CLI 实现，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。

  要创建使用 SQLJ 的程序包并指定其隔离级别，请使用 SQLJ 概要文件定制程序（db2sqljcustomize 命令）。

- **在运行时，从 CLI 或 ODBC 中**:

  使用 CHANGE ISOLATION LEVEL 命令。借助 DB2 调用级接口（CLI），可以在 CLI 配置中更改隔离级别。在运行时，将 SQLSetConnectAttr 函数与 SQL_ATTR_TXN_ISOLATION 属性配合使用，以便设置 *ConnectionHandle* 自变量所引用的当前连接的事务隔离级别。此外，也可以在 `db2cli.ini` 文件中使用 TXNISOLATION 关键字。

- **在支持 REXX™ 的数据库服务器上**:

  创建数据库时，将有多个绑定文件与该数据库绑定，这些文件支持 REXX 中 SQL 的不同隔离级别。创建数据库时，其他命令行处理器（CLP）程序包也会与该数据库绑定。

  REXX 和 CLP 使用缺省的 CS 隔离级别来连接到数据库。更改此隔离级别并不会更改连接状态。

要确定 REXX 应用程序所使用的隔离级别，请检查 SQLISL 预定义的 REXX 变量的值。此值在 CHANGE ISOLATION LEVEL 命令每次执行时被更新。

“当前已落实”语义提高了并行性

在 CS 隔离级别下，进行行级别锁定可能会引起锁定超时和死锁，对于未设计成能够防止此类问题的应用程序而言尤其如此。某些高吞吐量数据库应用程序无法容忍等待事务处理期间发出的锁定，并且，某些应用程序无法容忍处理未落实的数据，但仍要求读事务不会被阻塞。

在新的*当前已落实*语义下，将像以前那样只返回已落实的数据，但读取者现在不等待写入者释放行锁定。而是，读取者返回基于当前已落实版本的数据；即，写操作启动前的数据。

缺省情况下，对于新数据库，将打开“当前已落实”语义。这允许任何应用程序利用新行为，而不需要更改应用程序本身。您可以使用新的数据库配置参数 **cur_commit** 来覆盖此行为。这很有用，例如，如果应用程序要求阻塞写程序以便对内部逻辑进行同步，那么可以使用此参数。

同样，已升级的数据库在缺省情况下将 **cur_commit** 置于禁用状态，以防应用程序万一要求阻塞写程序以便对它们的内部逻辑进行同步；以后，您可以根据需要打开此参数。

“当前已落实”语义仅适用于不涉及目录表的只读扫描或者用于评估或实施约束的内部扫描。注意，由于“当前已落实”是在扫描级决定的，因此写程序的访问方案可能包括当前已落实的扫描。例如，只读子查询的扫描可以涉及“当前已落实”语义。由于“当前已落实”语义遵守隔离级别语义，因此在“当前已落实”语义下运行的应用程序将继续遵守隔离级别。

“当前已落实”语义要求增加日志空间供写入者使用。附加的空间用于记录事务期间对数据行进行的第一次更新。为了检索该行的当前已落实映像，此数据是必需的。根据工作负载的不同，这将对使用的日志总空间量产生不显著的影响或重大影响。当 **cur_commit** 处于禁用状态时，不需要附加的日志空间。

示例

请考虑以下通过“当前已落实”语义来避免死锁的方案。在此方案中，两个应用程序更新两个不同的表，但尚未进行落实。然后，这两个应用程序都尝试使用只读游标来读取另一个应用程序所更新的表。

步骤	应用程序 A	应用程序 B
1	update T1 set col1 = ? where col2 = ?	update T2 set col1 = ? where col2 = ?
2	select col1, col3, col4 from T2 where col2 >= ?	select col1, col5, from T1 where col5 = ? and col2 = ?
3	commit	commit

如果没有“当前已落实”语义，那么在“游标稳定性”隔离级别下运行的这些应用程序可能会引起死锁，从而导致其中一个应用程序失败。当这两个应用程序需要读取另一个应用程序正在更新的数据时，将发生这种死锁情况。

在“当前已落实”语义下，如果步骤 2 中任何一个应用程序的查询刚好需要当前正被另一应用程序更新的数据，那么该应用程序将不会等待该锁定被释放，因此不可能发生死锁情况。而是，将找到并使用先前落实的数据版本。

用于忽略未落实的插入的选项

DB2_SKIPINSERTED 注册表变量控制对于使用游标稳定性（CS）或读稳定性（RS）隔离级别的语句，是否可以忽略未落实的数据插入。

根据 **DB2_SKIPINSERTED** 注册表变量的值不同，将以两种方式中的一种来处理未落实的插入。

- 如果值为 ON，那么 DB2 服务器将忽略未落实的插入，这在许多情况下能够提高并行性，并且是大多数应用程序的首选行为。未落实的插入被视为尚未发生。
- 如果值为 OFF（缺省值），那么 DB2 服务器将等待插入操作完成（落实或回滚），然后相应地处理数据。这在某些情况下合适。例如:
  - 假定两个应用程序使用一个表在它们之间传递数据，第一个应用程序将数据插入到表中，第二个应用程序从表中读取数据。第二个应用程序必须按照数据在表中的出现顺序来处理数据，如果要读取的下一行正在由第一个应用程序插入，那么第二个应用程序必须等待插入操作落实。
  - 应用程序通过删除数据并接着插入该数据的新映像来避免使用 UPDATE 语句。

通过锁定延迟对未落实的数据进行求值

为了提高并行性，数据库管理器在某些情况下允许将 CS 或 RS 隔离扫描的行锁定延迟，直到已知某一行满足查询的谓词条件为止。

缺省情况下，在表扫描或索引扫描期间执行行级别锁定时，数据库管理器将先锁定每个已被扫描但其落实状态未知的行，然后再确定该行是否满足查询的谓词条件。

为了提高此类扫描的并行性，请启用 **DB2_EVALUNCOMMITTED** 注册表变量，以便能够对未落实的数据执行谓词求值。包含未落实的更新的行可能不满足查询条件，但如果将谓词求值延迟到事务完成之后进行，那么该行可能会确实满足该查询的条件。

未落实的已删除行在表扫描期间将被跳过，并且，如果已启用 **DB2_SKIPDELETED** 注册表变量，那么数据库管理器在索引扫描期间将跳过已删除的键。

DB2_EVALUNCOMMITTED 注册表变量设置在编译时应用于动态 SQL 或 XQuery 语句，而在绑定期间应用于静态 SQL 或 XQuery 语句。这意味着，即使在运行时启用此注册表变量，除非在绑定时启用 **DB2_EVALUNCOMMITTED**，否则也不会使用避免锁定策略。即使在绑定时启用此注册表变量，但在运行时未启用此变量，避免锁定策略也仍然生效。对于静态 SQL 或 XQuery 语句，如果重新绑定程序包，那么绑定时生效的注册表变量设置就是所应用的设置。以隐式方式重新绑定静态 SQL 或 XQuery 语句时，将使用 **DB2_EVALUNCOMMITTED** 注册表变量的当前设置。

对不同访问方案的未落实数据进行求值的适用性

表 6. RID 纯索引访问

谓词	对未落实的数据进行求值
无	否
SARGable	是

表 7. 纯数据访问（关系 RID 列表或延迟 RID 列表）

谓词	对未落实的数据进行求值
无	否
SARGable	是

表 8. RID 索引和数据访问

谓词		对未落实的数据进行求值	
索引	数据	索引访问	数据访问
无	无	否	否
无	SARGable	否	否
SARGable	无	是	否
SARGable	SARGable	是	否

表 9. 块索引和数据访问

谓词		对未落实的数据进行求值	
索引	数据	索引访问	数据访问
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未落实数据求值行为进行比较。此表是 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
```

在缺省的游标稳定性（CS）隔离级别下，将执行下列事务。

表 10. 在 CS 隔离级别下对 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 语句将对表中的第一行挂起互斥锁定，从而导致会话 2 中的查询无法返回结果集，即使会话 1 中正在更新的行当前不满足会话 2 中查询的条件

亦如此。CS 隔离级别要求当游标定位在查询所访问的任何行上时，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 才能获取对该行的锁定。

通过使用对未落实的数据进行求值这一功能，可以避免在会话 2 中等待锁定，该功能首先对谓词进行求值，然后再锁定行。这样，会话 2 中的查询将不会尝试锁定表中的第一行，从而提高了应用程序并行性。注意，这还意味着会话 2 中的谓词求值根据会话 1 中 `deptnumb=5` 的未落实值进行。会话 2 中的查询将省略其结果集中的第一行，尽管在会话 1 中回滚更新将导致该行满足会话 2 中的查询条件。

即使掉转操作顺序，对未落实的数据进行求值这一功能也仍然能够提高并行性。使用缺省锁定行为时，会话 2 将首先获取行锁定，从而禁止在会话 1 中执行搜索型 UPDATE，即使会话 1 中的 UPDATE 语句不更改被会话 2 的查询锁定的行亦如此。如果会话 1 中的搜索型 UPDATE 首先尝试检查行，然后仅当这些行符合条件时才将其锁定，那么会话 1 中的查询将不会被阻塞。

限制

- 必须启用 **DB2_EVALUNCOMMITTED** 注册表变量。
- 隔离级别必须是 CS 或 RS。
- 行级别锁定生效。
- 存在 SARGable 求值谓词。
- 未落实数据求值功能不适用于对系统目录表执行的扫描。
- 对于多维集群（MDC）表，可以对索引扫描延迟块级锁定；但是，不能对表扫描延迟块级锁定。
- 对于正在执行原位置表重组操作的表，不会发生锁定延迟。
- 对于扫描预取方案而言，不会将行级别锁定延迟到数据访问期间进行；而是，在移至表中的某行之前在索引访问期间锁定该行。
- 在表扫描期间将无条件地跳过已删除的行，但仅当 **DB2_SKIPDELETED** 注册表变量处于启用状态时，才会跳过已删除的索引键。

编写和调整查询以便最大程度地提高性能

您可以通过多种方法来最大程度地降低 SQL 语句对 DB2 数据库性能的影响。

为了最大程度地降低此影响，您可以:

- 编写更方便于 DB2 优化器进行优化的 SQL 语句。DB2 优化器可能无法高效地运行包含非等式连接谓词、连接列的数据类型不匹配、包含非必需外连接以及包含其他复杂搜索条件的 SQL 语句。
- 正确地配置 DB2 数据库，以利用 DB2 优化功能。如果您有准确的目录统计信息，并且为工作负载选择最佳的优化类，那么 DB2 优化器就能够选择最佳的查询访问方案。
- 使用 DB2 说明功能来查看可能的查询访问方案并确定如何调整查询以便最大程度地提高性能。

最佳实践适用于常规工作负载、仓库工作负载和 SAP 工作负载。

虽然可以在应用程序编写完毕后通过多种方法来处理特定的查询性能问题，但您可以提前广泛应用良好的基本编写和调整实践措施以帮助提高 DB2 数据库性能。

查询性能并不是一次性的考虑事项。在应用程序开发生命周期的设计、开发和生产阶段，都应该对查询性能加以考虑。

SQL 是非常灵活的语言，这意味着，可以通过多种方法来获取相同的正确结果。这种灵活性还意味着，某些查询在利用 DB2 优化器的能力方面优于其他查询。

在查询执行期间，DB2 优化器将为每个 SQL 语句选择查询访问方案。优化器对许多备用访问方案的执行成本进行建模，并且将选择估算成本最低的访问方案。如果一个查询包含许多复杂的搜索条件，那么 DB2 优化器在某些情况下能够重写谓词，但在另外一些情况下却无法这样做。

对于复杂查询，例如商业智能（BI）应用程序中使用的那些查询，准备或编译 SQL 语句所需的时间可能较长。正确地设计和配置数据库有助于最大程度地缩短语句编译时间。这包括选择正确的优化类以及正确地设置其他注册表变量。

优化器还需要准确的输入才能作出准确的访问方案决策。这意味着，您需要收集准确的统计信息并有可能需要使用高级统计功能部件，例如统计视图和列组统计信息。

您可以使用 DB2 工具（尤其是 DB2 说明工具）来调整查询。DB2 编译器能够捕获关于静态或动态查询的访问方案和环境的信息。通过所捕获的此信息，您可以了解各个语句的运行情况，从而调整语句和数据库管理器配置以提高性能。

编写 SQL 语句

SQL 是一种功能十分强大的语言，它允许您以语法虽有不同但语义完全相同的方式来指定关系表达式。但是，某些在语义上等同的变体比其他变体更易于优化。虽然 DB2 优化器具有强大的查询重写功能，但是，它并非始终能够将 SQL 语句重写成最优的形式。

某些 SQL 构造会对查询优化器所考虑的访问方案造成限制，您应该尽可能避免或替换这些构造。

避免在搜索条件中使用复杂的表达式:

避免在搜索条件中使用复杂的表达式，这些表达式将导致优化器无法使用目录统计信息来估算精确的选择性。

表达式还可能限制对可用于应用谓词的访问方案的选择。在优化的查询重写阶段，优化器可以重写许多表达式以便估算精确的选择性；它无法处理所有的可能性。

避免对表达式使用连接谓词:

如果对表达式使用连接谓词，那么连接方法只能是嵌套循环连接。

并且，估算的基数有可能不准确。包含表达式的连接的一些示例如下所示:

```
WHERE SALES.PRICE * SALES.DISCOUNT = TRANS.FINAL_PRICE
WHERE UPPER(CUST.LASTNAME) = TRANS.NAME
```

避免在局部谓词中使用基于列的表达式:

不要在局部谓词中使用基于列的表达式，而是，请使用表达式的翻转形式。

请考虑以下示例:

```
XPRESSN(C) = 'constant'
INTEGER(TRANS_DATE)/100 = 200802
```

您可以重写这些语句，如下所示:

```
C = INVERSEXPRESSN('constant')
TRANS_DATE BETWEEN 20080201 AND 20080229
```

使用基于列的表达式将导致无法使用索引开始键和停止键，从而导致估算的选择性不正确，并要求在执行查询时进行额外的处理。

这些表达式还将导致无法进行查询重写优化，例如识别各列何时等同、将列替换为常量以及识别何时最多返回一行。在可以证实最多返回一行之后，还可以进一步进行优化，因此失去的优化机会进一步增多。请考虑以下查询:

```
SELECT LASTNAME, CUST_ID, CUST_CODE FROM CUST
WHERE (CUST_ID * 100) + INT(CUST_CODE) = 123456 ORDER BY 1,2,3
```

可以将此查询重写为:

```
SELECT LASTNAME, CUST_ID, CUST_CODE FROM CUST
WHERE CUST_ID = 1234 AND CUST_CODE = '56' ORDER BY 1,2,3
```

如果已定义基于 CUST_ID 的唯一索引，那么重写的查询版本将使查询优化器能够认识到最多返回一行。这将避免引入不必要的 SORT 操作。这还使 CUST_ID 和 CUST_CODE 列能够被替换为 1234 和“56”，从而避免从数据页或索引页复制值。最后，这还使作用于 CUST_ID 的谓词能够作为索引开始键或停止键进行应用。

谓词中存在表达式这一情况并不一定明显。当视图列由表达式定义时，引用这些视图的查询通常就是这种情况。例如，考虑以下视图定义和查询:

```
CREATE VIEW CUST_V AS
  (SELECT LASTNAME, (CUST_ID * 100) + INT(CUST_CODE) AS CUST_KEY
    FROM CUST)

SELECT LASTNAME FROM CUST_V WHERE CUST_KEY = 123456
```

查询优化器将把该查询与视图定义合并，从而生成以下查询:

```
SELECT LASTNAME FROM CUST
WHERE (CUST_ID * 100) + INT(CUST_CODE) = 123456
```

这就是上一个示例中描述的那个有问题的谓词。您可以使用说明工具来显示经过优化的 SQL 语句，从而观察视图合并结果。

如果翻转功能难以表达，那么请考虑使用生成列。例如，如果要查找符合 LASTNAME IN ('Woo', 'woo', 'WOO', 'WOo',...) 所表达的条件的名字，那么可以创建生成列 UCASE (LASTNAME) = 'WOO'，如下所示:

```
CREATE TABLE CUSTOMER (
  LASTNAME VARCHAR(100),
  U_LASTNAME VARCHAR(100) GENERATED ALWAYS AS (UCASE(LASTNAME))
)

CREATE INDEX CUST_U_LASTNAME ON CUSTOMER(U_LASTNAME)
```

在 DB2 数据库 Linux 版、UNIX 版和 Windows 版的版本 9.5 修订包 1 中，对不区分大小写搜索的支持就是为了解决此特定示例所述的情况。您可以使用 UCA500R1 整理顺序名称中的 _Sx 属性来控制整理强度。例如，UCA500R1_LFR_S1 是忽略大小写和重音符的法语整理顺序。

避免连接列之间数据类型不匹配:

在某些情况下，数据类型不匹配将导致无法使用散列连接。

与其他连接方法相比，散列连接对连接谓词有一些额外的限制。尤其是，连接列的数据类型必须完全相同。例如，如果一个连接列是 FLOAT，而另一个是 REAL，那么不支持散列连接。另外，如果连接列数据类型是 CHAR、GRAPHIC、DECIMAL 或 DECFLOAT，那么长度必须相同。

避免在谓词中使用空操作表达式来更改优化器估算:

格式为 `COALESCE(X, X) = X` 的空操作 coalesce() 谓词将导致任何使用它的查询的规划发生估算错误。目前，DB2 查询编译器无法详细分析该谓词并确定所有行都确实满足它的要求。

因此，该谓词将人为减少来自某个查询规划部分的估算行数。行估算值减小通常会导致该查询规范中其余部分的行和成本估算值减小，有时还会导致选择另一个方案，这是因为不同候选方案之间的相对估算值已更改。

为何这种空操作谓词有时能够提高查询性能？添加空操作 coalesce() 谓词将引入一个错误，该错误将屏蔽某些其他导致性能无法达到最佳水平的内容。

某些性能增强工具执行的是强制测试：工具反复地将该谓词引入到查询中的不同位置，从而对不同的列执行操作，以便尝试找到能够通过引入错误无意中发现性能更好的方案的情况。对于在查询中手动编码“空操作”谓词的查询开发者而言，情况亦如此。通常，开发者对数据有一定程度的了解，这有助于确定谓词的位置。

使用此方法来提高查询性能是一种短期的解决方案，这无法解决根本原因，并且可能会造成以下影响:

- 可能能够提高性能的领域未显现。
- 无法保证此变通方法能够永久提高性能，这是因为，DB2 查询编译器最终可能能够更好地处理谓词，其他随机因素也可能会对其产生影响。
- 其他查询也可能会受同一根本原因影响，这通常会导致系统的性能有所下降。

如果您已遵循最佳实践建议，但相信仍未实现最佳性能，那么可以为 DB2 优化器提供明确的优化准则，而不是引入空操作谓词。请参阅“优化概要文件和准则”。

避免使用非等式连接谓词:

因为连接方法只能是嵌套循环连接，所以应该避免使用比较运算符不是等式的连接谓词。

另外，优化器可能无法准确计算连接谓词的选择性估算值。但是，并非始终能够避免使用非等式连接谓词。有必要使用非等式连接谓词时，请确保存在基于任何一个表的适当索引，这是因为连接谓词将应用于嵌套循环连接的内表。

非等式连接谓词的一个常见示例是，必须对星型模式中的维数据进行版本化才能准确地反映某个维在不同时间点的状态。这通常称为*缓慢变化的维*。其中一种缓慢变化的维涉及包括每个维行的有效开始和结束日期。事实表与维表之间的连接除了要求根据维主键进行连接以外，还要求检查与事实相关联的日期是否在维的开始日期与结束日期之间。这通常称为*第 6 类缓慢变化的维*。通过某个事实事务日期向后连接到事实表以便进一步限定维版本的范围连接成本很高。例如:

```
SELECT...
  FROM PRODUCT P, SALES F
  WHERE
    P.PROD_KEY = F.PROD_KEY AND
    F.SALE_DATE BETWEEN P.START_DATE AND
    P.END_DATE
```

在这种情况下，请确保存在基于 (F.PROD_KEY, F.SALE_DATE) 的索引。

请考虑创建统计视图以帮助优化器为此方案计算更好的选择性估算值。例如:

```
CREATE STATISTICAL VIEW V_PROD_FACT AS
  SELECT P.*
    FROM PRODUCT P, SALES F
    WHERE
      P.PROD_KEY = F.PROD_KEY AND
      F.SALE_DATE BETWEEN P.START_DATE AND
      P.END_DATE

ALTER VIEW V_PROD_FACT ENABLE QUERY OPTIMIZATION

RUNSTATS ON TABLE DB2USER.V_PROD_FACT WITH DISTRIBUTION
```

如果查询块中存在任何非等式连接谓词，那么不考虑专用的星型模式连接，例如使用索引与（AND）运算的星型连接和集线器连接。（请参阅“确保查询符合星型模式连接的必需条件”。）

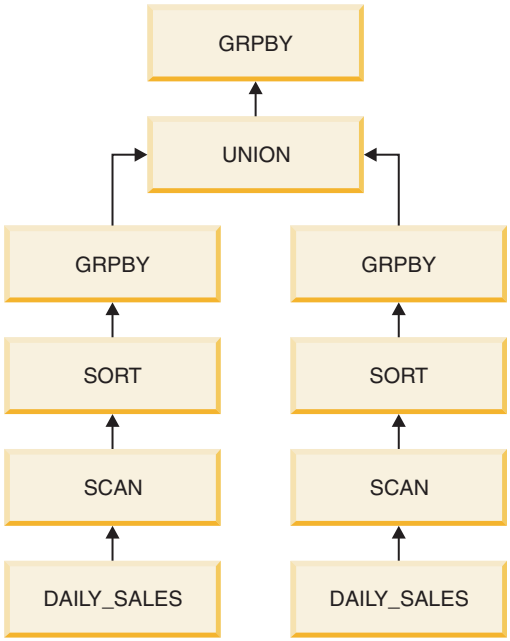
避免使用多个带有 DISTINCT 关键字的聚集操作:

请避免使用在同一个子查询中执行多次 DISTINCT 聚集操作的查询，这些查询的运行成本高昂。

请考虑以下示例:

```
SELECT SUM(DISTINCT REBATE), AVG(DISTINCT DISCOUNT)
  FROM DAILY_SALES
  GROUP BY PROD_KEY
```

要确定相异 REBATE 值和相异 DISCOUNT 值的集合，可能需要对来自 PROD_KEY 表的输入流进行两次排序。此查询的查询访问方案可能类似于:



优化器将原始查询重写为不同的聚集并对每个聚集指定 DISTINCT 关键字，然后使用 UNION 关键字对多个聚集进行组合。在内部重写的语句如下:

```
SELECT Q8.MAXC0, (Q8.MAXC1 / Q8.MAXC2)
FROM
  (SELECT MAX(Q7.C0) AS MAXC0, MAX(Q7.C1) AS MAXC1, MAX(Q7.C2) AS MAXC2
    FROM
      (SELECT SUM(DISTINCT Q2.REBATE) AS C0, CAST(NULL AS INTEGER) AS C1,
        0 AS C2, Q2.PROD_KEY
      FROM
        (SELECT Q1.PROD_KEY, Q1.REBATE
          FROM DB2USER.DAILY_SALES AS Q1) AS Q2
      GROUP BY Q2.PROD_KEY
      UNION ALL
      SELECT CAST(NULL AS INTEGER) AS C0, SUM(DISTINCT Q5.DISCOUNT) AS C1,
          COUNT(DISTINCT Q5.DISCOUNT) AS C2, Q5.PROD_KEY
        FROM
          (SELECT Q4.PROD_KEY, Q4.DISCOUNT
            FROM DB2USER.DAILY_SALES AS Q4) AS Q5
        GROUP BY Q5.PROD_KEY) AS Q7
      GROUP BY Q7.PROD_KEY) AS Q8
```

如果无法避免使用多个 DISTINCT 聚集，那么请考虑将 **DB2_EXTENDED_OPTIMIZATION** 注册表变量与 ENHANCED_MULTIPLE_DISTINCT 选项配合使用。此选项将导致读一次以多个相异聚集为目标的输入流，然后对 UNION 的每个分支重复使用该输入流。此选项可以提高这些类型的查询的性能，其中，处理器数与数据库分区数的比率较低（例如，比率小于或等于 1）。在不包含对称多处理器（SMP）的分区数据库环境中，应该使用此设置。此优化扩展功能并不能在所有环境中提高查询性能。您应执行测试，以确定各个查询的性能提高情况。

避免使用不必要的外连接:

某些查询的语义需要外连接（左连接、右连接或全连接）。但是，如果查询语义不需要外连接，并且该查询正被用于处理不一致的数据，那么它最适合于处理数据不一致问题的根本原因。

例如，在具有星型模式的数据集市中，事实表可能包含事务的行，但由于数据一致性问题，某些维没有匹配的父维。发生此问题的原因可能是，抽取、变换和装入（ETL）过程由于某种原因而未能对某些业务键进行协调。在此方案中，事实表行以左外连接方式与维连接，以确保它们即使没有父代也会被返回。例如:

```
SELECT...
  FROM DAILY_SALES F
    LEFT OUTER JOIN CUSTOMER C ON F.CUST_KEY = C.CUST_KEY
    LEFT OUTER JOIN STORE S ON F.STORE_KEY = S.STORE_KEY
  WHERE
    C.CUST_NAME = 'SMITH'
```

左外连接也会导致无法进行多项优化，其中包括使用专用的星型模式连接访问方法。但是，在某些情况下，查询优化器可以自动地将左外连接重写成内连接。在本示例中，由于谓词 C.CUST_NAME = 'SMITH' 将除去任何在此列中包含空值的行，从而使左外连接在语义上非必需，所以 CUSTOMER 与 DAILY_SALES 之间的左外连接可以转换为内连接。因此，由于存在外连接而无法进行某些优化可能不会对所有查询产生负面影响。但是，了解这些限制并避免使用外连接（除非绝对有需要使用外连接）至关重要。

将 OPTIMIZE FOR N ROWS 子句与 FETCH FIRST N ROWS ONLY 子句配合使用:

OPTIMIZE FOR *n* ROWS 子句通知优化器，应用程序计划只检索 *n* 行，但是查询将返回完整的结果集。FETCH FIRST *n* ROWS ONLY 子句指示查询应该只返回 *n* 行。

对外子查询指定 FETCH FIRST *n* ROWS ONLY 之后，DB2 数据服务器不会自动采用 OPTIMIZE FOR *n* ROWS。请尝试同时指定 OPTIMIZE FOR *n* ROWS 和 FETCH FIRST *n* ROWS ONLY，以鼓励使用直接从所引用的表返回行而不首先执行缓存操作（例如插入到临时表、执行排序或插入到散列连接散列表）的查询访问方案。

应用程序如果指定了 OPTIMIZE FOR *n* ROWS 以鼓励使用避免缓存操作的查询访问方案，但却检索整个结果集，那么性能可能会欠佳。这是因为，返回前 *n* 行速度最快的查询访问方案在检索整个结果集时可能不是最好的查询访问方案。

确保查询符合星型模式连接的必需条件:

对于星型模式，优化器将考虑两种专用的连接方法，即星型连接或集线器连接，它们有助于显著提高性能。

但是，查询必须符合以下条件。

- 对于每个查询块
  - 必须至少连接三个不同的表
  - 所有连接谓词都必须是等式谓词
  - 不能存在子查询
  - 在各个表之间或查询块外部不能存在相关性或依赖性
  - 对于索引与（AND）运算，不能存在不确定函数，这是因为索引必须应用事实表谓词以便于进行半连接
  - 事实表
    - 是查询块中最大的表
    - 至少包含 10000 行

- 被认为仅仅是一个表
- 必须至少连接至两个维表或者称为“雪花”的组

– 维表
- 不是事实表
- 可以逐个连接到事实表或者在“雪花”中进行连接

– 维表或“雪花”
- 必须对事实表进行过滤（过滤基于优化器的估算值。）
- 必须存在一个用于事实表并且使用事实表索引中的前导列的连接谓词。必须符合此条件才能考虑使用星型连接或集线器连接，尽管集线器连接将只需要使用单一事实表索引。

表示左外连接或右外连接的查询块只能引用两个表，因此星型模式连接不符合条件。

不需要显式地声明引用完整性即可使优化器识别星型模式连接。

避免使用冗余的谓词:

避免使用冗余的谓词，当它们跨不同的表出现时尤其如此。在某些情况下，优化器无法检测谓词是否冗余。这可能导致低估基数。

例如，在 SAP 商业智能（BI）应用程序中，将带有事实表和维表的雪花模式用作查询优化数据结构。在某些情况下，对事实表和维表定义了冗余的时间特征列（用于月份的“SID_0CALMONTH”或者用于年份的“SID_0FISCPER”）。

SAP BI 联机分析处理（OLAP）处理器将对维表和事实表的时间特征列生成冗余的谓词。

这些冗余的谓词可能会导致查询运行时间延长。

下一节提供了一个示例，在此示例中，SAP BI 查询的 WHERE 条件中定义了两个冗余的谓词。对时间维（DT）和事实（F）表定义了完全相同的谓词:

```
    AND (     "DT"."SID_0CALMONTH" = 199605
          AND "F". "SID_0CALMONTH" = 199605
           OR "DT"."SID_0CALMONTH" = 199705
          AND "F". "SID_0CALMONTH" = 199705 )
AND NOT (     "DT"."SID_0CALMONTH" = 199803
          AND "F". "SID_0CALMONTH" = 199803 )
```

DB2 优化器不会将这些谓词识别为等同，而是将它们视为相互独立。这将导致低估基数、查询访问方案欠佳以及查询运行时间延长。

因此，特定于 DB2 数据库平台的软件层将除去冗余的谓词。

上述谓词将转换为如下所示的谓词。只保留了应用于事实表列“SID_0CALMONTH”的谓词:

```
    AND (     "F". "SID_0CALMONTH" = 199605
           OR "F". "SID_0CALMONTH" = 199705 )
AND NOT (     "F". "SID_0CALMONTH" = 199803 )
```

请应用 SAP 注意事项 957070 和 1144883 中的指示信息以除去冗余的谓词。

使用约束来提高查询优化程度

请考虑定义唯一约束、检查约束和引用完整性约束。这些约束将提供语义信息，这些信息使 DB2 优化器能够重写查询以消除连接、通过连接下推聚集、通过连接下推 FETCH FIRST *n* ROWS、除去不必要的 DISTINCT 操作以及执行许多其他优化。

当应用程序本身能够确保关系时，还可以使用参考约束来代替检查约束和引用完整性约束。在这种情况下，可以进行相同的优化。在插入、更新或删除行时，数据库管理器强制实施的约束可能会引起大量系统开销，更新大量具有引用完整性约束的行时尤其如此。如果应用程序在更新行之前已验证信息，那么使用参考约束可能比使用常规约束效率高。

例如，假定有两个表 DAILY_SALES 和 CUSTOMER。CUSTOMER 表中的每一行都有唯一的客户键（CUST_KEY）。DAILY_SALES 包含 CUST_KEY 列，并且每一行都引用 CUSTOMER 表中的客户键。您可以创建引用完整性约束以表示 CUSTOMER 与 DAILY_SALES 之间的这种 1:N 关系。如果应用程序强制实施此关系，那么可以将该约束定义为参考约束。于是，以下查询可以避免在 CUSTOMER 与 DAILY_SALES 之间执行连接，这是因为，不会从 CUSTOMER 中检索任何列，并且 DAILY_SALES 中的每一行都将在 CUSTOMER 中找到匹配项。查询优化器将自动除去连接。

```
SELECT AMT_SOLD, SALE PRICE, PROD_DESC
  FROM DAILY_SALES, PRODUCT, CUSTOMER
  WHERE
    DAILY_SALES.PROD_KEY = PRODUCT.PRODKEY AND
    DAILY_SALES.CUST_KEY = CUSTOMER.CUST_KEY
```

应用程序必须强制实施参考约束，否则查询可能会返回不正确的结果。在此示例中，如果 DAILY_SALES 中的任何行在 CUSTOMER 表中没有相应的客户键，那么此查询将不正确地返回那些行。

在复杂查询中将 REOPT 绑定选项与输入变量配合使用

在语句通常比较简单并且查询访问方案的选择较为直接的联机事务处理（OLTP）环境中，输入变量对于缩短语句准备时间而言十分关键。

使用不同输入变量值多次执行同一个查询时，可以重复使用动态语句高速缓存中已编译的访问节，从而避免在输入值每次更改时执行成本高昂的 SQL 语句编译工作。

但是，输入变量可能导致复杂的查询工作负载发生问题，对于这些工作负载而言，查询访问方案的选择较为复杂，因此优化器需要更多的信息才能作出良好的决策。此外，语句编译时间通常只是整体执行时间的一小部分，通常不会反复执行的商业智能（BI）查询无法受益于动态语句高速缓存。

如果需要在复杂查询工作负载中使用输入变量，那么请考虑使用 REOPT(ALWAYS) 绑定选项。REOPT 绑定选项将语句编译工作从准备阶段推迟到打开或执行阶段进行，在那些阶段，输入变量值已确定。这些值将被传递到 SQL 编译器，因此优化器可以使用这些值来更准确地估算选择性。REOPT(ALWAYS) 指定每次执行语句时都应该对其进行重新编译。REOPT(ALWAYS) 还可以用于引用了专用寄存器的复杂查询，例如 `WHERE TRANS_DATE =CURRENT DATE - 30 DAYS`。如果输入变量导致为 OLTP 工作负载选择的访问方案欠佳，并且 REOPT(ALWAYS) 由于语句编译工作而引起大量开销，那么请考虑对所选查询使用 REOPT(ONCE)。REOPT(ONCE) 将语句编译工作推迟到绑定第一个输入变量值时进行。于是，将使用第一个输入变量值来编译和优化 SQL 语句。以后使用不同的值来执行该语句时，将重复使用根据第一个输入值编译的访问节。如果第一

个输入变量值能够代表后续的值，那么这是一种好方法，并且它提供的查询访问方案优于输入变量值未知时基于缺省值的查询访问方案。

可以通过多种方法来指定 REOPT:

- 对于 C/C++ 应用程序中的嵌入式 SQL，使用 REOPT 绑定选项。此绑定选项将同时影响静态 SQL 和动态 SQL 的重新优化行为。
- 对于 CLP 程序包，使用 REOPT 绑定选项来重新绑定 CLP 程序包。例如，要使用 REOPT ALWAYS 来重新绑定用于隔离级别 CS 的 CLP 程序包，请指定以下命令:

  ```
  rebind nullid.SQLC2G13 reopt always
  ```

- 对于使用旧款 JDBC 驱动程序的 CLI 应用程序或 JDBC 应用程序，请在 db2cli.ini 配置文件中使用 REOPT 关键字设置。这些值和相应的选项是:
  - 2 - NONE
  - 3 - ONCE
  - 4 - ALWAYS
- 对于使用 JCC 通用驱动程序的 JDBC 应用程序，使用下列其中一种方法:
  - 使用 SQL_ATTR_REOPT 连接属性或语句属性。
  - 使用 SQL_ATTR_CURRENT_PACKAGE_SET 连接属性或语句属性来指定 NULLID、NULLIDR1 或 NULLIDRA 程序包集。NULLIDR1 和 NULLIDRA 是保留的程序包集名称。使用这两个名称时，它们分别表示 REOPT ONCE 或 REOPT ALWAYS。这些程序包集必须由下列命令显式地创建:

    ```
    db2 bind db2clipk.bnd collection NULLIDR1
    db2 bind db2clipk.bnd collection NULLIDRA
    ```

- 对于 SQL PL 过程，请使用下列其中一种方法:
  - 使用 SET_ROUTINE_OPTS 存储过程来设置要在当前会话中用于创建 SQL PL 过程的绑定选项。例如，调用:

    ```
    sysproc.set_routine_opts('reopt always')
    ```

  - 使用 **DB2_SQLROUTINE_PREPOPTS** 注册表变量在实例级别设置 SQL PL 过程选项。使用 SET_ROUTINE_OPTS 存储过程设置的值将覆盖通过 **DB2_SQLROUTINE_PREPOPTS** 指定的值。

还可以使用优化概要文件为静态语句和动态语句设置 REOPT，如以下示例所示:

```
<STMTPROFILE ID="REOPT example ">
     <STMTKEY>
       <![CDATA[select acct_no from customer where name = ? ]]>
        </STMTKEY>        <OPTGUIDELINES>        <REOPT VALUE='ALWAYS'/>
       </OPTGUIDELINES>  </STMTPROFILE>
```

使用参数标记来缩短动态查询的编译时间

DB2 数据服务器可以将访问节和语句文本存储在动态语句高速缓存中，从而避免重新编译先前已运行的动态 SQL 语句。

对此语句发出的后续准备请求将尝试在动态语句高速缓存中查找访问节，从而避免进行编译。但是，仅在谓词中使用的字面值方面有所不同的语句不匹配。例如，下面这两个语句在动态语句高速缓存中被认为不相同:

```
SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 26790
SELECT AGE FROM EMPLOYEE WHERE EMP_ID = 77543
```

即使频繁运行相对简单的 SQL 语句，也会由于语句编译工作而导致系统 CPU 使用率过高。如果系统遇到此类性能问题，那么请考虑更改应用程序，以便使用参数标记将谓词值传递给 DB2 编译器，而不要显式地将它们包括在 SQL 语句中。但是，对于在谓词中使用了参数标记的复杂查询而言，此访问方案可能并非最佳。有关更多信息，请参阅“将 REOPT 绑定选项与复杂查询中的输入变量配合使用”。

设置 DB2_REDUCED_OPTIMIZATION 注册表变量

如果设置优化类无法充分缩短应用程序的编译时间，那么请尝试设置 **DB2_REDUCED_OPTIMIZATION** 注册表变量。

与设置优化类相比，此注册表变量提供了对优化器搜索空间的更多控制权。此注册表变量允许您请求使用精简优化功能或者严格按照指定的优化类来使用优化功能。如果减少所使用的优化技术的数目，那么还将减少优化期间耗用的时间和资源。

尽管可以减少优化期间的时间和资源耗用量，但这也会增大生成非最佳查询访问方案的风险。

首先，尝试将此注册表变量设置为 YES。如果优化类为 5（缺省值）或更低，那么优化器将禁用某些优化技术，这些技术可能会耗用大量的准备时间和资源，但通常无法生成更好的查询访问方案。如果优化类正好是 5，那么优化器将精简或禁用其他一些技术，这可以进一步缩短优化时间和资源耗用量，但是也将进一步增大生成非最佳查询访问方案的风险。对于低于 5 的优化类，其中某些技术可能并非在任何情况下都有效。但是，如果它们有效，那么它们将保持有效。

如果 YES 设置未充分缩短编译时间，那么请尝试将此注册表变量设置为整数值。效果与 YES 相同，但对于使用优化类 5 优化的动态准备的查询，存在下列附加行为。如果任何查询块中的连接总数超出此设置，那么优化器将切换到贪婪联合枚举算法，而不是禁用其他优化技术。结果是，将在类似于优化类 2 的级别优化查询。

提高插入性能

将数据插入到表中之前，插入搜索算法将检查可用空间控制记录（FSCR），以查找空间足以存储新数据的页。

但是，即使 FSCR 指示某页的可用空间足够，该空间也可能因为已被另一个事务中未落实的删除操作保留而不可用。

DB2MAXFSCRSEARCH 注册表变量指定将记录添加到表时要搜索的 FSCR 数目。缺省情况是搜索 5 个 FSCR。修改此值使您能够在插入速度与空间复用之间进行平衡。使用较大的值将优化空间复用。使用较小的值将优化插入速度。将值设置为 -1 表示强制数据库管理器搜索所有 FSCR。如果搜索 FSCR 时找不到足够的空间，那么数据将被追加到表的末尾。

ALTER TABLE 语句的 APPEND ON 选项指定将追加表数据，并指定不保留关于页中可用空间的信息。这样的表不能带有集群索引。对于只增大不减小的表，此选项能够提高性能。

如果已对该表定义集群索引，那么数据库管理器会尝试将记录插入到其他具有类似索引键值的记录所在的页。如果该页上没有空间，那么将考虑周围的页。如果那些页不

合适，那么将搜索 FSCR，如上所述。但是，在这种情况下，将使用“最差匹配”方法来代替“首先匹配”方法。最差匹配方法往往选择包含更多可用空间的页。此方法将为具有类似键值的行建立新的集群区。

如果已对一个表定义集群索引，那么在装入或重组该表之前，请使用 ALTER TABLE 语句的 PCTFREE 子句。PCTFREE 子句指定执行装入或重组操作后应该在数据页中保留的可用空间所占的百分比。这将提高集群索引操作能在适当的页中找到可用空间的可能性。

高效的 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类，性能可能相差很大。

请考虑下列有关创建高效 SELECT 语句的准则:

- 仅指定需要的列。使用星号（*）指定所有列将产生不必要的处理。
- 使用谓词将答案集限制为仅包括所需的行。
- 如果需要的行数大大小于可能返回的总行数，那么请指定 OPTIMIZE FOR 子句。此子句将影响对访问方案的选择以及在通信缓冲区中分块的行数。
- 要利用行分块方法并提高性能，请指定 FOR READ ONLY 或 FOR FETCH ONLY 子句。并且，因为不会对检索到的行挂起互斥锁定，所以并行性也有所改进。此外，还可能会发生附加的查询重写。同样，通过指定这些子句以及 BLOCKING ALL 绑定选项，可以提高对联合数据库系统中的昵称运行的查询的性能。
- 对于将与定位型更新操作配合使用的游标而言，指定 FOR UPDATE OF 子句将使数据库管理器能够选择更合适的初始锁定级别并避免发生潜在的死锁。注意，FOR UPDATE 游标无法利用行分块方法。
- 对于将与搜索型更新操作配合使用的游标而言，指定 FOR READ ONLY 和 USE AND KEEP UPDATE LOCKS 子句将对受影响的行强制挂起 U 锁定，从而避免死锁并仍允许进行行分块。
- 尽可能避免进行数字数据类型转换。在比较值时，请尝试使用具有相同数据类型的项。如果需要执行转换，那么会由于精度受限而导致结果不准确，并且会由于进行运行时转换而导致性能下降。

  在有可能的时候，请使用下列数据类型:
  - 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、小数或 DECFLOAT
  - 尽量使用 DECFLOAT，而不是小数
  - 尽量使用日期时间，而不是字符
  - 尽量使用数字，而不是字符
- 为了减小发生排序操作的可能性，请省略 DISTINCT 或 ORDER BY 之类的子句（如果此类操作不是必需的）。
- 要检查表中是否存在行，请选择单一的行。请打开游标并访存一行，或执行单行 SELECT INTO 操作。记住，如果找到多行，那么需检查是否发生 SQLCODE -811 错误。

  除非您知道表非常小，否则不要使用以下语句来检查非零值:

```
select count(*) from <table-name>
```

对于大型的表，对所有行计数将影响性能。

- 如果更新活动较少且表较大，那么请对谓词中频繁使用的列定义索引。
- 如果同一列出现在多个谓词中，那么请考虑使用 IN 列表。对于配合主变量使用的大型 IN 列表，循环部分主变量可能会提高性能。

下列建议只适用于访问多个表的 SELECT 语句。

- 使用连接谓词来连接表。*连接谓词*是指一个连接中不同表的两个列之间的比较。
- 对连接谓词中的列定义索引，以便更高效地处理连接。对于包含访问多个表的 SELECT 语句的 UPDATE 和 DELETE 语句而言，索引也有助于提高性能。
- 有可能时，避免使用包含连接谓词的 OR 子句或表达式。
- 在分区数据库环境中，建议根据连接列对连接的表进行分区。

有关限制 SELECT 语句的准则

优化器假定应用程序必须检索所有由 SELECT 语句标识的行。此假定最适合于联机事务处理（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很大的答案集，但它们只检索前几行，通常只检索填满特定显示格式所需数目的行。

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 子句来指定可以由后续定位型 UPDATE 语句更新的列。
- 使用 FOR READ 或 FETCH ONLY 子句使返回的列成为只读列。
- 使用 OPTIMIZE FOR *n* ROWS 子句来指定优先检索整个结果集的前 *n* 行。
- 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。

下列各节描述每种方法的性能优点。

FOR UPDATE 子句

FOR UPDATE 子句通过仅包括可由后续定位型 UPDATE 语句更新的列来限制结果集。如果在未指定列名的情况下指定 FOR UPDATE 子句，那么将包括表或视图中所有可更新的列。如果指定了列名，那么每个名称都必须是非限定的，并且必须标识该表或视图的某一列。

在下列情况下，不能使用 FOR UPDATE 子句:

- 无法删除与 SELECT 语句相关联的游标。
- 在选择的列中，至少有一列是目录表中的不可更新列，但尚未在 FOR UPDATE 子句中将其排除。

在 DB2 CLI 应用程序中，可以使用 CLI 连接属性 SQL_ATTR_ACCESS_MODE 来实现同一目标。

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。对于允许执行更新和删除操作的结果表，如果数据库管理器可以检索数据块来代替进行互斥锁

定，那么指定 FOR READ ONLY 子句可以提高访存操作的性能。请不要在定位型 UPDATE 或 DELETE 语句中使用的查询中指定 FOR READ ONLY 子句。

在 DB2 CLI 应用程序中，可以使用 CLI 连接属性 SQL_ATTR_ACCESS_MODE 来实现同一目标。

OPTIMIZE FOR *n* ROWS 子句

OPTIMIZE FOR 子句声明只想检索部分结果或者优先检索前几行。于是，优化器可以选择能够将检索前几行所需的响应时间缩至最短的访问方案。此外，作为单个块发送到客户机的行数由 *n* 的值限制。因此，OPTIMIZE FOR 子句既影响服务器从数据库检索合格行的方式，又影响将那些行返回给客户机的方式。

例如，假定您定期查询 EMPLOYEE 表以确定哪些职员的薪水最高:

```
select lastname, firstnme, empno, salary
  from employee
  order by salary desc
```

虽然您先前已对 SALARY 列定义了一个降序索引，但因为职员按职员编号排序，所以此索引的集群度可能不佳。为了尽量避免许多随机的同步 I/O，优化器将可能选择列表预取访问方法，此方法要求对所有合格行的行标识进行排序。此排序可能会导致在将前几个合格行返回给应用程序之前发生延迟。为了防止此延迟，请在语句中添加 OPTIMIZE FOR 子句，如下所示:

```
select lastname, firstnme, empno, salary
  from employee
  order by salary desc
  optimize for 20 rows
```

在这种情况下，优化器可能选择直接使用 SALARY 索引，这是因为只检索薪水最高的 20 位职员。无论可以对多少行进行分块，都只将包含 20 行的行块返回给客户机。

使用 OPTIMIZE FOR 子句，优化器优先选择能够避免大批操作或流中断（例如排序操作所引起的情况）的访问方案。您最有可能使用 OPTIMIZE FOR 1 ROW 子句来影响访问路径。使用此子句有下列作用:

- 降低了连接序列包含组合内表的可能性，这是因为它们需要临时表。
- 连接方法可能会有所更改。最有可能的选项是嵌套循环连接，这是因为它的开销成本较低，并且在检索少量行时通常更有效率。
- 更有可能存在与 ORDER BY 子句匹配的索引，这是因为 ORDER BY 不要求进行排序。
- 降低了列表预取的可能性，这是因为此访问方法要求进行排序。
- 降低了顺序预取的可能性，这是因为只需要少量的几行。
- 在连接查询中，有可能将包含 ORDER BY 子句中的列的表选作外表，前提是该外表的某个索引提供了 ORDER BY 子句所需的排序。

虽然 OPTIMIZE FOR 子句适用于所有优化级别，但它在优化类 3 和更高优化类下工作得最好，这是因为低于 3 的优化类使用*贪婪联合枚举*搜索策略。此方法有时会产生无法使它们自己快速检索前几行的多表连接访问方案。

如果已打包的应用程序使用调用级接口（DB2 CLI 或 ODBC），那么可以在 db2cli.ini 配置文件中使用 **OPTIMIZEFORNROWS** 关键字，让 DB2 CLI 在每个查询语句的末尾自动追加 OPTIMIZE FOR 子句。

从昵称选择数据时，结果可能随数据源支持的不同而有所变化。如果昵称所引用的数据源支持 OPTIMIZE FOR 子句，并且 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，或者优化器确定最低成本方案是本地执行，那么将在本地应用 OPTIMIZE FOR 子句。在这种情况下，DB2 优化器将优先选择能够最大程度缩短检索某个查询前几行的响应时间的访问方案，但可供优化器用于生成方案的选项略微受限，并且 OPTIMIZE FOR 子句对性能的改善幅度可能微不足道。

如果同时指定 OPTIMIZE FOR 子句和 FETCH FIRST 子句，那么两个 *n* 值中的较小者将影响通信缓冲区大小。为了进行优化，这两个值被视为互不相关。

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有助于提高性能。无论结果集可能另外包含多少行，也只检索 *n* 行。

如果同时指定 FETCH FIRST 子句和 OPTIMIZE FOR 子句，那么两个 *n* 值中的较小者将影响通信缓冲区大小。为了进行优化，这两个值被视为互不相关。

DECLARE CURSOR WITH HOLD 语句

如果使用包含 WITH HOLD 子句的 DECLARE CURSOR 语句来声明游标，那么打开的游标在事务落实后将继续处于打开状态，并且所有锁定都将被释放，但那些用于保护当前游标位置的锁定除外。如果回滚事务，那么将关闭所有打开的游标、释放所有锁定并释放任何 LOB 定位器。

在 DB2 CLI 应用程序中，可以使用 CLI 连接属性 SQL_ATTR_CURSOR_HOLD 来实现同一目标。如果已打包的应用程序使用调用级接口（DB2 CLI 或 ODBC），那么请在 db2cli.ini 配置文件中使用 **CURSORHOLD** 关键字，让 DB2 CLI 自动为每个已声明的游标假设 WITH HOLD 子句。

指定行分块以降低开销

所有语句和数据类型（其中包括 LOB 数据类型）都支持行分块，行分块通过在单个操作中检索行块来降低游标的数据库管理器开销。

这个行块代表内存中的多个页。它不是多维（MDC）表块，后者在物理上映射到磁盘上的扩展数据块。

行分块由 BIND 或 PREP 命令的下列选项指定:

BLOCKING ALL
在声明时指定了 FOR READ ONLY 子句或者未被指定为 FOR UPDATE 的游标将被分块。

BLOCKING NO
游标将不会被分块。

BLOCKING UNAMBIG

在声明时指定了 FOR READ ONLY 子句的游标将被分块。在声明时未指定 FOR READ ONLY 子句或 FOR UPDATE 子句的游标、非模糊游标或者只读游标将被分块。模糊游标将不会被分块。

在计算块大小时，将使用下列数据库管理器配置参数。

- **aslheapsz** 参数指定本地应用程序的应用程序支持层堆的大小。它用来确定打开分块游标时使用的 I/O 块大小。
- **rqrioblk** 参数指定远程应用程序与数据库服务器上其数据库代理程序之间的通信缓冲区的大小。它还用来确定打开分块游标时数据服务器运行时客户机中使用的 I/O 块大小。

对 LOB 数据类型的行数据启用分块之前，务必了解这样做对系统资源产生的影响。返回 LOB 列时，在服务器上将耗用更多共享内存来存储对每个数据块中 LOB 值的引用。此类引用的数目随 **rqrioblk** 配置参数值的不同而有所变化。

要增加分配给堆的内存量，请通过以下方法来修改数据库配置参数 **database_memory**:

- 将它的值设置为 AUTOMATIC
- 如果此参数当前设置为用户定义的数值，那么将该值增大 256 页

为了提高引用了 LOB 值的现有嵌入式 SQL 应用程序的性能，可以使用 BIND 命令来重新绑定应用程序，并指定 BLOCKING ALL 子句或 BLOCKING UNAMBIG 子句以请求进行分块。嵌入式应用程序将从服务器检索行块，然后以每次一行的方式检索 LOB 值。返回 LOB 结果的用户定义的函数（UDF）可能会导致 DB2 服务器恢复为以单行方式检索 LOB 数据，从而导致在服务器上耗用大量内存。

要指定行分块，请执行下列操作:

1. 使用 **aslheapsz** 和 **rqrioblk** 配置参数的值来估算为每个块返回的行数。在这两个公式中，*orl* 是以字节计的输出行长度。
   - 对于本地应用程序，使用以下公式:

     ```
     每块行数 = aslheapsz * 4096 / orl
     ```

     每页的字节数为 4096。
   - 对于远程应用程序，使用以下公式:

     ```
     每块行数 = rqrioblk / orl
     ```
2. 要启用行分块，请对 BIND 或 PREP 命令的 BLOCKING 选项指定适当的值。

   如果未指定 BLOCKING 选项，那么缺省行分块类型为 UNAMBIG。对于命令行处理器（CLP）和调用级接口（CLI），缺省行分块类型为 ALL。

查询中的数据采样

通常，访问所有与查询相关的数据并不实际，有时也没有必要。在某些情况下，在部分数据中查找整体趋势或模式已经足够。完成此任务的其中一种方法是，对数据库的随机样本运行查询。

DB2 产品使您能够高效地为 SQL 和 XQuery 查询进行数据采样，从而有可能大幅提升大型查询的性能，同时保持高度的准确性。

采样功能通常用于聚集查询，例如 AVG、COUNT 和 SUM，对于这些查询，可以从数据样本中获取相当准确的值以进行聚集。采样功能也可用于获取表中各行的随机子集，以便进行审计或者提高数据挖掘和分析速度。

可以通过两种方法进行采样：行级采样和页级采样。

行级别伯努利采样

行级别伯努利采样通过使用 SARGable 谓词（它包含样本中每一行的概率为 *P*/100，排除样本中每一行的概率为 1 - *P*/100）来获取表行的 *P*% 样本。

行级别伯努利采样总是生成有效的随机样本，而不考虑数据集群程度。但是，如果没有可用的索引，那么这种采样的性能会很低，这是因为必须检索每一行并对其应用采样谓词。如果没有任何索引，那么与不进行采样而执行查询相比，不会节省任何 I/O。如果有可用的索引，那么性能将有所提高，这是因为将对索引叶子页内的 RIDS 应用采样谓词。一般情况下，每个所选 RID 要求执行一次 I/O，并且每个索引叶子页要求执行一次 I/O。

系统页级采样

系统页级采样与行级采样类似，但是对页采样而不是对行采样。每一页包括在样本中的概率都是 *P*/100。如果包括某一页，那么将包括该页中所有的行。

系统页级采样的性能很好，这是因为，样本中包括的每一页只需要执行一次 I/O。与不采样相比，页级采样将大幅提高性能。但是，页级采样与行级采样相比，聚集估算的准确性可能较差。当每一页包含许多行，或者查询中引用的列在各页中展现较高的集群度时，这种差别最为显著。

指定采样方法

使用 TABLESAMPLE 子句对表的随机数据样本执行查询。TABLESAMPLE BERNOULLI 指定执行行级别伯努利采样。TABLESAMPLE SYSTEM 指定执行系统页级采样，除非优化器确定执行行级别伯努利采样更为高效。

应用程序的并行处理

DB2 产品支持并行环境，确切而言，在对称多处理器（SMP）机器上支持此环境。

在 SMP 机器中，允许多个处理器访问数据库，从而将复杂 SQL 请求的执行分布到各个处理器。这种*分区内并行性*是指将单一数据库操作（例如，创建索引）分为多个部分，然后以并行方式在单一数据库分区中执行那些部分。

要在编译应用程序时指定并行度，请使用 CURRENT DEGREE 专用寄存器或 DEGREE 绑定选项。*度*是指可以同时执行的查询部分数。在处理器数目与选择的并行度值之间没有严格的关系。您可以指定比机器上的处理器数目多或少的值。甚至对于单处理器机器，也可以设置大于 1 的并行度以提高性能。但请注意，每个并行度都将增加系统内存和处理器开销。

如果要以并行方式执行查询，那么必须修改某些配置参数以提高性能。在具有高并行度的环境中，您应该查看并修改控制共享内存量和预取的配置参数。

下列配置参数控制和管理并行处理。

- **intra_parallel** 数据库管理器配置参数用于启用或禁用并行性。
- **max_querydegree** 数据库管理器配置参数用于对数据库中的任何查询设置并行度上限。此值覆盖 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项。
- **dft_degree** 数据库配置参数用于设置 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项的缺省值。

如果编译查询时指定了 DEGREE = ANY，那么数据库管理器将根据许多因素（其中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。如果系统很忙，那么在执行查询前可能会降低并行度。

请使用 DB2 说明工具来显示关于优化器选择的并行度的信息。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

即使在未使用 SMP 机器的情况下，也可以指定并行度。例如，在单处理器机器上，受 I/O 约束的查询可能能够通过将并行度声明为 2 或更大的值而受益。在这种情况下，处理器可能不必等待 I/O 任务完成就可以开始处理新的查询。LOAD 之类的实用程序可以独立地控制 I/O 并行性。

锁定管理

锁定管理是其中一项将会影响应用程序性能的因素。请查看本节的内容，以了解有关可以帮助您最大程度地提高数据库应用程序性能的锁定管理注意事项的详细信息。

锁定与并行性控制

为了提供并行控制并防止数据访问不受控，数据库管理器将锁定缓冲池、表、数据分区、表块或表行。

*锁定*使数据库管理器资源与应用程序（称为*锁定所有者*）相关联，以便控制其他应用程序访问同一资源的方式。

数据库管理器根据下列各项来适当地使用行级别锁定或表级别锁定:

- 在预编译时或者将应用程序与数据库绑定时指定的隔离级别。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）

  各种隔离级别用于控制对未落实的数据的访问、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。为了最大程度地降低对性能的影响，请使用能够满足应用程序需要的最低隔离级别。
- 优化器选择的访问方案。表扫描、索引扫描和其他数据访问方法都需要不同的数据访问类型。
- 表的 LOCKSIZE 属性。ALTER TABLE 语句的 LOCKSIZE 子句指示访问表时使用的锁定的粒度。选项是: ROW（表示行锁定）、TABLE（表示表锁定）或

BLOCKINSERT（仅表示对多维集群（MDC）表的块锁定）。对 MDC 表使用 BLOCKINSERT 子句时，除了执行块级锁定的插入操作以外，将执行行级别锁定。当事务将对分离的单元执行大型插入时，请对 MDC 表使用 ALTER TABLE...LOCKSIZE BLOCKINSERT 语句。对于只读的表，请使用 ALTER TABLE...LOCKSIZE TABLE 语句。这将减少数据库活动所需的锁定数。对于分区表，将按所要访问的数据指示的那样首先获取表锁定，然后获取数据分区锁定。

- 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，那么性能可能会因为锁定升级以及数据库中共享对象之间的并行性降低而下降。如果锁定升级频繁发生，那么请增大 **locklist** 和/或 **maxlocks** 的值。要减少同时挂起的锁定数，请确保事务频繁地落实。

每当创建、更改或删除缓冲池时，将设置互斥的缓冲池锁定。在收集系统监视数据时，您可能会遇到这种类型的锁定。此锁定的名称是缓冲池本身的标识。

通常，除非存在下列其中一种情况，否则将使用行级别锁定:

- 隔离级别是“未落实的读”
- 隔离级别是“可重复读”，并且访问方案要求在不使用索引范围谓词的情况下执行扫描
- 表的 LOCKSIZE 属性是 TABLE
- 锁定列表填满，从而引起锁定升级
- 已通过 LOCK TABLE 语句获取显式的表锁定，这将导致并发应用程序进程无法更改或使用表

对于 MDC 表，在下列情况下，将使用块级锁定来代替行级别锁定:

- 表的 LOCKSIZE 属性是 BLOCKINSERT
- 隔离级别是“可重复的读”，并且访问方案涉及谓词
- 搜索型更新或删除操作仅涉及应用于维列的谓词

行锁定的持续时间随使用的隔离级别不同而有所变化:

- 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: 通常，仅当游标定位在行上时，才挂起行锁定。注意，在某些情况下，CS 扫描期间可能完全不挂起锁定。
- RS 扫描: 只在事务执行期间挂起合格的行锁定。
- RR 扫描: 在事务执行期间挂起所有行锁定。

锁定粒度

如果某个应用程序对某个数据库对象挂起锁定，那么另一个应用程序可能无法访问该对象。因此，与块级锁定、数据分区级锁定或表级别锁定相比，锁定最少量数据并使这些数据不可访问的行级别锁定更有利于最大程度地提高并行性。

但是，锁定需要存储器和处理时间，因此单个表锁定能够最大程度地降低锁定开销。

ALTER TABLE 语句的 LOCKSIZE 子句指定行级别锁定、数据分区级锁定、块级锁定或表级别锁定的粒度。缺省情况下，将使用行锁定。在表定义中使用此选项并不会阻止正常的锁定升级发生。

ALTER TABLE 语句以全局方式指定锁定，从而影响所有访问该表的应用程序和用户。各个应用程序可以使用 LOCK TABLE 语句在应用程序级别指定表锁定。

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，且始终只需要 S 锁定。其他用户也可以获取对该表的 S 锁定。
- 表通常由只读应用程序访问，但有时由单个用户访问以进行短暂维护，而该用户需要 X 锁定。当维护程序运行时，只读应用程序将无法访问该表，但在其他情况下，只读应用程序可以同时访问该表并且锁定开销最小。

对于多维集群（MDC）表而言，可以同时指定 LOCKSIZE 子句和 BLOCKINSERT 以便只在插入操作期间使用块级锁定。如果指定了 BLOCKINSERT，那么将对所有其他操作执行行级别锁定，但极少会对插入操作执行行级别锁定。即，在插入行时使用块级锁定，但如果更新记录标识（RID）索引时在索引中遇到可重复读（RR）扫描，那么使用行级别锁定来锁定下一个键。BLOCKINSERT 锁定在下列情况下可能有用:

- 多个事务正在对不同单元执行大量插入操作
- 未发生多个事务以并发方式对同一个单元执行插入的情况，或者已发生这种情况，但由于每个事务向每个单元插入足够的数据，所以用户并不关心每个事务是否对不同的块执行插入

锁定属性

数据库管理器锁定具有多项基本属性。

这些属性包括:

方式 允许锁定所有者进行的访问类型以及允许被锁定对象的并发用户进行的访问类型。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。数据库管理器还会锁定其他类型的资源，例如行和表空间。此外，还可以对多维集群（MDC）表进行块锁定以及对分区表进行数据分区锁定。正被锁定的对象确定了锁定的*粒度*。

锁定数 挂起锁定的时间长度。查询运行时所处的隔离级别将影响锁定数。

表 11 按照对资源的控制性递增顺序列示锁定方式并描述它们的效果。

表 11. 锁定方式摘要

锁定方式	适用的对象类型	描述
IN（无意向）	表空间、块、表和数据分区	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。其他并发应用程序可以读取或更新该表。
IS（意向共享）	表空间、块、表和数据分区	锁定所有者可以读取已锁定的表中的数据，但不能更新此数据。其他应用程序可以读取或更新该表。
IX（意向互斥）	表空间、块、表和数据分区	锁定所有者和并发应用程序可以读取和更新数据。其他并发应用程序既可以读取也可以更新该表。
NS（扫描共享）	行	锁定所有者和所有并发应用程序可以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。

表 11. 锁定方式摘要 (续)

锁定方式	适用的对象类型	描述
NW（下一键弱互斥）	行	将行插入到索引时，获取对下一行的 NW 锁定。仅当下一行当前被 RR 扫描锁定时，才会发生这种情况。锁定所有者可以读取但不可更新已锁定的行。除了与 NS 锁定兼容之外，此锁定方式类似于 X 锁定。
S（共享）	行、块、表和数据分区	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。
SIX（在意向互斥下共享）	表、块和数据分区	锁定所有者可以读取和更新数据。其他并发应用程序可以读取该表。
U（更新）	行、块、表和数据分区	锁定所有者可以更新数据。其他工作单元可以读取锁定的对象中的数据，但不能对其进行更新。
X（互斥）	行、块、表、缓冲池和数据分区	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”（UR）应用程序才能访问已锁定的对象。
Z（超级互斥）	表空间、表和数据分区	在特定情况下，例如更改或删除表、创建或删除表的索引或者执行某些类型的表重组时，将获取对表的这种锁定。其他并发应用程序都无法读取或更新该表。

影响锁定的因素

数据库管理器锁定的方式和粒度受多个因素影响。

这些因素包括:

- 应用程序执行的处理类型
- 数据访问方法
- 各种配置参数的值

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分类为下列其中一种类型：只读、有意更改、更改以及受游标控制。

- 只读

  此处理类型包括所有具有只读本质、包含显式 FOR READ ONLY 子句或者虽然具有二义性但查询编译器因为 PREP 或 BIND 命令指定了 BLOCKING 选项值而假定它们是只读语句的 SELECT 语句。此类型只需要共享锁定（IS、NS 或 S）。

- 有意更改

  此处理类型包括所有包含 FOR UPDATE 子句、包含 USE AND KEEP UPDATE LOCKS 子句、包含 USE AND KEEP EXCLUSIVE LOCKS 子句或者虽然具有二义性但查询编译器假定有意执行更改的 SELECT 语句。此类型使用共享和更新锁定（对于行，为 S、U 或 X；对于块，为 IX、S、U 或 X；对于表，为 IX、U 或 X）。

- 更改

  此处理类型包括 UPDATE、INSERT 和 DELETE 语句，但不包括 UPDATE WHERE CURRENT OF 或 DELETE WHERE CURRENT OF。此类型需要互斥锁定（IX 或 X）。

- 受游标控制

此处理类型包括 UPDATE WHERE CURRENT OF 和 DELETE WHERE CURRENT OF。此类型需要互斥锁定（IX 或 X）。

根据子选择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数据的表的锁定。更改处理规则确定对目标表的锁定。

锁定和数据访问方法

*访问方案*是优化器选择的用于从特定表中检索数据的方法。访问方案会对锁定方式产生重大影响。

如果使用索引扫描方法来查找特定的行，那么优化器通常为该表选择行级别锁定（IS）。例如，如果 EMPLOYEE 表有一个基于职员编号（EMPNO）的索引，那么可以通过该索引进行访问，以便选择单个职员的信息:

```
select * from employee
  where empno = '000310'
```

如果未使用索引，那么必须按顺序扫描整个表才能找到所需的行，并且优化器有可能会选择单个表级别锁定（S）。例如，如果没有基于 SEX 列的索引，那么可以通过表扫描方法来选择所有男性职员，如下所示:

```
select * from employee
  where sex = 'M'
```

注: 受游标控制的处理将使用底层游标的锁定方式，直到应用程序找到要更新或删除的行为止。对于这种类型的处理，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操作。

范围集群表中的锁定与标准键锁定的工作方式略有不同。访问范围集群表中某个范围的行时，该范围内的所有行都会被锁定，即使其中某些行是空行亦如此。在标准键锁定中，将只锁定包含现有数据的行。

延迟型数据页访问意味着对行的访问分两步进行，这将导致锁定情况更为复杂。锁定获取计时和锁定持久性取决于隔离级别。由于可重复读（RR）隔离级别将保持所有锁定直到事务结束，因此将保持第一步中获取的锁定，而不必在第二步另外获取锁定。对于读稳定性（RS）和游标稳定性（CS）隔离级别而言，必须在第二步期间获取锁定。为了最大程度地提高并行性，在第一步期间不获取锁定，并通过重新应用所有谓词来确保只返回合格行。

锁定类型兼容性

当一个应用程序对某个对象挂起锁定，而另一个应用程序请求锁定同一个对象时，就会发生锁定兼容性问题。当两种锁定方式兼容时，可以同意对该对象的第二个锁定请求。

如果请求的锁定的锁定方式与已挂起的锁定不兼容，那么不能同意锁定请求。相反，请求要等到第一个应用程序释放其锁定，并且释放所有其他现有不兼容锁定为止。

第 146 页的表 12 指示了兼容的锁定类型（由“是”指示）以及不兼容的类型（由“否”指示）。注意，当请求者正等待锁定时，可能出现超时。

表 12. 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**
无	是	是	是	是	是	是	是	是	是	是	是
IN（无意向）	是	是	是	是	是	是	是	是	是	否	是
IS（意向共享）	是	是	是	是	是	是	是	是	否	否	否
NS（扫描共享）	是	是	是	是	是	否	否	是	否	否	是
S（共享）	是	是	是	是	是	否	否	是	否	否	否
IX（意向互斥）	是	是	是	否	否	是	否	否	否	否	否
SIX（在意向互斥下共享）	是	是	是	否	否	否	否	否	否	否	否
U（更新）	是	是	是	是	是	否	否	否	否	否	否
X（互斥）	是	是	否	否	否	否	否	否	否	否	否
Z（超级互斥）	是	否	否	否	否	否	否	否	否	否	否
NW（下一键弱互斥）	是	是	否	是	否	否	否	否	否	否	否

下一键锁定

在将一个键插入到索引期间，对于该索引中将跟在新键后面的键所对应的行，只有在该行当前被可重复读（RR）索引扫描锁定的情况下，才会锁定该行。当发生这种情况时，将延迟到完成了执行 RR 扫描的事务之后才会插入新的索引键。

用于下一键锁定的锁定方式是 NW（下一键弱互斥）。下一键锁定将在键插入操作发生前被释放；即，在将行插入到表之前被释放。

如果更新某一行导致更改该行的索引键值，那么也会发生插入键这种情况，这是因为，需要将原始键值标记为“已删除”并将新的键值插入到索引中。对于只影响索引的包括列的更新，可以在原位置更新键，而不会进行下一键锁定。

在 RR 扫描期间，将以 S 方式锁定与跟在扫描范围末尾后面的键相对应的行。如果没有任何键跟在扫描范围末尾后面，那么将获取表结束锁定以锁定索引的末尾。对于分区表的分区索引而言，将获取多个锁定以锁定每个索引分区的末尾，而不是仅获取一个锁定以锁定索引末尾。如果扫描范围末尾之后的键已被标记为“已删除”，那么将执行下列其中一项操作:

- 扫描将继续进行以锁定相应的行，直到找到未被标记为“已删除”的键为止
- 扫描将锁定该键的相应行
- 扫描将锁定索引的末尾

标准表的锁定方式和访问方案

标准表获取的锁定类型取决于生效中的隔离级别以及所使用的数据访问方案。

下列各表列示不同访问方案在各种隔离级别下获取的标准表锁定类型。每个条目都分为两部分: 表锁定和行锁定。连字符表明特定的锁定粒度不可用。

表 7-12 列示了将数据页读取操作推迟时获取的锁定类型（推迟目的是，允许列示要使用多个索引进一步限定的行或者要进行排序以便高效预取的行）。

- 表 1. 不使用谓词的表扫描的锁定方式
- 表 2. 使用谓词的表扫描的锁定方式
- 表 3. 不使用谓词的 RID 索引扫描的锁定方式
- 表 4. 使用单个合格行的 RID 索引扫描的锁定方式
- 表 5. 仅使用 Start 和 Stop 谓词的 RID 索引扫描的锁定方式
- 表 6. 仅使用索引和其他谓词（sargs 和 resids）的 RID 索引扫描的锁定方式
- 表 7. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的 RID 索引扫描
- 表 8. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的 RID 索引扫描之后
- 表 9. 用于延迟型数据页访问的索引扫描的锁定方式: 使用谓词（sargs 和 resids）的 RID 索引扫描
- 表 10. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用谓词（sargs 和 resids）的 RID 索引扫描之后
- 表 11. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的 RID 索引扫描
- 表 12. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

注:

1. 块级锁定也适用于多维集群（MDC）表。
2. 可以使用 SELECT 语句的 *lock-request-clause* 显式地更改锁定方式。

表 13. 不使用谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 14. 使用谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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 隔离级别下，如果存在应用于索引中包括列的谓词，那么隔离级别将升级到 CS，并且锁定将升级到 IS 表锁定或 NS 行锁定。

表 15. 不使用谓词的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 16. 使用单个合格行的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 17. 仅使用 Start 和 Stop 谓词的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 18. 仅使用索引和其他谓词（sargs 和 resids）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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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
UR	IN/-	IX/U	IX/X	IX/U	IX/X

表 19. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S	IX/S		X/-	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 20. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 21. 用于延迟型数据页访问的索引扫描的锁定方式: 使用谓词（sargs 和 resids）的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S	IX/S		IX/S	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 22. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用谓词（sargs 和 resids）的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 23. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S	IX/S		IX/X	
RS	IN/-	IN/-		IN/-	
CS	IN/-	IN/-		IN/-	
UR	IN/-	IN/-		IN/-	

表 24. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

MDC 表和 RID 索引扫描的锁定方式

多维集群（MDC）表在表或 RID 索引扫描期间获取的锁定类型取决于生效中的隔离级别以及所使用的数据访问方案。

下列各表列示不同访问方案在各种隔离级别下获取的 MDC 表锁定类型。每个条目都分为三部分：表锁定、块锁定和行锁定。连字符表明特定的锁定粒度不可用。

表 9-14 列示了将数据页读取操作推迟时获取的 RID 索引扫描锁定类型。在 UR 隔离级别下，如果存在应用于索引中包括列的谓词，那么隔离级别将升级到 CS，并且锁定将升级到 IS 表锁定、IS 块锁定或 NS 行锁定。

- 表 1. 不使用谓词的表扫描的锁定方式
- 表 2. 仅使用维列上的谓词的表扫描的锁定方式
- 表 3. 使用其他谓词（sargs 和 resids）的表扫描的锁定方式
- 表 4. 不使用谓词的 RID 索引扫描的锁定方式
- 表 5. 使用单个合格行的 RID 索引扫描的锁定方式
- 表 6. 仅使用 Start 和 Stop 谓词的 RID 索引扫描的锁定方式
- 表 7. 仅使用索引谓词的 RID 索引扫描的锁定方式
- 表 8. 使用其他谓词（sargs 和 resids）的 RID 索引扫描的锁定方式
- 表 9. 用于延迟型数据页访问的索引扫描的锁定方式：不使用谓词的 RID 索引扫描
- 表 10. 用于延迟型数据页访问的索引扫描的锁定方式：在不使用谓词的 RID 索引扫描之后
- 表 11. 用于延迟型数据页访问的索引扫描的锁定方式：使用谓词（sargs 和 resids）的 RID 索引扫描
- 表 12. 用于延迟型数据页访问的索引扫描的锁定方式：在使用谓词（sargs 和 resids）的 RID 索引扫描之后
- 表 13. 用于延迟型数据页访问的索引扫描的锁定方式：仅使用 Start 和 Stop 谓词的 RID 索引扫描
- 表 14. 用于延迟型数据页访问的索引扫描的锁定方式：在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

注： 可以使用 SELECT 语句的 *lock-request-clause* 显式地更改锁定方式。

表 25. 不使用谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-

表 26. 仅使用维列上的谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-

表 26. 仅使用维列上的谓词的表扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-

表 27. 使用其他谓词（sargs 和 resids）的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 28. 不使用谓词的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	IX/IX/S	IX/IX/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

表 29. 使用单个合格行的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 30. 仅使用 Start 和 Stop 谓词的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 31. 仅使用索引谓词的 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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 32. 使用其他谓词（sargs 和 resids）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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 33. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-	

表 34. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的 RID 索引扫描之后

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

表 35. 用于延迟型数据页访问的索引扫描的锁定方式: 使用谓词（sargs 和 resids）的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-	

表 36. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用谓词（sargs 和 resids）的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 37. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-	

表 38. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

MDC 块索引扫描的锁定方式

多维集群（MDC）表在块索引扫描期间获取的锁定类型取决于生效中的隔离级别以及所使用的数据访问方案。

下列各表列示不同访问方案在各种隔离级别下获取的 MDC 表锁定类型。每个条目都分为三部分: 表锁定、块锁定和行锁定。连字符表明特定的锁定粒度不可用。

表 5-12 列示了将数据页读取操作推迟时获取的块索引扫描锁定类型。

- 表 1. 不使用谓词的索引扫描的锁定方式
- 表 2. 仅使用维列上的谓词的谓词扫描的锁定方式
- 表 3. 仅使用 Start 和 Stop 谓词的索引扫描的锁定方式
- 表 4. 使用谓词的索引扫描的锁定方式
- 表 5. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的块索引扫描
- 表 6. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的块索引扫描之后

- 表 7. 用于延迟型数据页访问的索引扫描的锁定方式: 仅对维列使用谓词的块索引扫描
- 表 8. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅对维列使用谓词的块索引扫描之后
- 表 9. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的块索引扫描
- 表 10. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的块索引扫描之后
- 表 11. 用于延迟型数据页访问的索引扫描的锁定方式: 使用其他谓词（sargs 和 resids）的块索引扫描
- 表 12. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用其他谓词（sargs 和 resids）的块索引扫描之后

注: 可以使用 SELECT 语句的 *lock-request-clause* 显式地更改锁定方式。

表 39. 不使用谓词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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--

表 40. 仅使用维列上的谓词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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-

表 41. 仅使用 Start 和 Stop 谓词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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/-

表 42. 使用谓词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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 42. 使用谓词的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		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

表 43. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--	

表 44. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--

表 45. 用于延迟型数据页访问的索引扫描的锁定方式: 仅对维列使用谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 46. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅对维列使用谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--

表 47. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/--		IX/X/--	

表 47. 用于延迟型数据页访问的索引扫描的锁定方式：仅使用 Start 和 Stop 谓词的块索引扫描 (续)

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 48. 用于延迟型数据页访问的索引扫描的锁定方式：在仅使用 Start 和 Stop 谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--	

表 49. 用于延迟型数据页访问的索引扫描的锁定方式：使用其他谓词（sargs 和 resids）的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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/--	

表 50. 用于延迟型数据页访问的索引扫描的锁定方式：在使用其他谓词（sargs 和 resids）的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
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

对分区表的锁定行为

除锁定整个表以外，还可以锁定分区表的每个数据分区。

与非分区表相比，这支持更高的粒度并提高了并行性。db2pd 命令、事件监视器、管理视图和表函数的输出都将标识数据分区锁定。

访问表时，将首先获取表锁定，然后根据需要获取数据分区锁定。访问方法和隔离级别可能要求锁定结果集未涉及到的数据分区。获取这些数据分区锁定之后，在表锁定

保持期间可能会一直挂起这些锁定。例如，对索引执行的游标稳定性（CS）扫描可能保持对先前访问的数据分区的锁定，以便降低以后重新获取数据分区锁定的成本。

数据分区锁定还承担用于确保对表空间进行访问的成本。对于非分区表而言，表空间访问由表锁定处理。即使在表级别存在互斥锁定或共享锁定，也会进行数据分区锁定。

更高的粒度允许一个事务对特定数据分区具有互斥访问权并避免进行行锁定，而其他事务能够访问其他数据分区。这可能是为批量更新选择的方案或者将锁定升级到数据分区级别的结果。许多访问方法的表锁定通常是意向锁定，即使以共享或互斥方式锁定数据分区亦如此。这将提高并行性。但是，如果在数据分区级别需要非意向锁定并且方案表明可能会访问所有数据分区，那么可能会在表级别选择非意向锁定，以防止并发事务之间发生数据分区死锁。

LOCK TABLE 语句

对于分区表而言，LOCK TABLE 语句获取的唯一锁定是表级别锁定。这将防止后续数据操作语言（DML）语句执行行锁定，并避免在行、块或数据分区级别出现死锁情况。更新索引时，可以使用 IN EXCLUSIVE MODE 选项来保证互斥访问，这对于在大型更新期间限制索引增大很有用。

ALTER TABLE 语句的 LOCKSIZE TABLE 选项的影响

LOCKSIZE TABLE 选项确保以共享或互斥方式来锁定表，而不进行意向锁定。对于分区表而言，这种锁定策略将同时应用于表锁定和数据分区锁定。

行级别锁定和块级锁定升级

分区表中的行级别锁定和块级锁定可以升级到数据分区级别。发生这种情况后，其他事务可以更容易地访问该表，即使数据分区锁定升级到共享、互斥或超级互斥方式亦如此，这是因为其他数据分区保持不受影响。升级操作的通知日志条目指示了所影响的数据分区以及该表的名称。

锁定升级无法确保对非分区索引进行互斥访问。要进行互斥访问，必须符合下列其中一个条件:

- 语句必须使用互斥表级别锁定
- 必须发出显式的 LOCK TABLE IN EXCLUSIVE MODE 语句
- 该表必须具有 LOCKSIZE TABLE 属性

对于分区索引而言，对索引分区的互斥访问由数据分区到互斥或超级互斥访问方式的锁定升级确保。

解释锁定信息

SNAPLOCK 管理视图可以帮助您解释对某个分区表返回的锁定信息。以下是脱机索引重组期间捕获的 SNAPLOCK 管理视图。

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE,
LOCK_MODE, LOCK_ESCALATION FROM SYSIBMADM.SNAPLOCK where TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE

TABNAME         TAB_FILE_ID          TBSP_NAME       DATA_PARTITION_ID LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_ESCALATION
--------------- -------------------- --------------- ----------------- ------------------ ---------- ---------------
```

```
TP1                         32768 -                         -1 TABLE_LOCK        Z                    0
TP1                             4 USERSPACE1                 0 TABLE_PART_LOCK    Z                    0
TP1                             5 USERSPACE1                 1 TABLE_PART_LOCK    Z                    0
TP1                             6 USERSPACE1                 2 TABLE_PART_LOCK    Z                    0
TP1                             7 USERSPACE1                 3 TABLE_PART_LOCK    Z                    0
TP1                             8 USERSPACE1                 4 TABLE_PART_LOCK    Z                    0
TP1                             9 USERSPACE1                 5 TABLE_PART_LOCK    Z                    0
TP1                            10 USERSPACE1                 6 TABLE_PART_LOCK    Z                    0
TP1                            11 USERSPACE1                 7 TABLE_PART_LOCK    Z                    0
TP1                            12 USERSPACE1                 8 TABLE_PART_LOCK    Z                    0
TP1                            13 USERSPACE1                 9 TABLE_PART_LOCK    Z                    0
TP1                            14 USERSPACE1                10 TABLE_PART_LOCK    Z                    0
TP1                            15 USERSPACE1                11 TABLE_PART_LOCK    Z                    0
TP1                             4 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                             5 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                             6 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                             7 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                             8 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                             9 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            10 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            11 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            12 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            13 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            14 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            15 USERSPACE1                 - TABLE_LOCK         Z                    0
TP1                            16 USERSPACE1                 - TABLE_LOCK         Z                    0

  已选择 26 个记录。
```

在此示例中，类型为 TABLE_LOCK 且 DATA_PARTITION_ID 为 -1 的锁定对象用于控制对分区表 TP1 的访问权以及并行性。类型为 TABLE_PART_LOCK 的锁定对象用于控制每个数据分区的大多数访问权和并行性。

此输出还捕获了其他类型为 TABLE_LOCK 的锁定对象（TAB_FILE_ID 4 到 16），这些锁定对象没有 DATA_PARTITION_ID 值。如果某种类型的锁定中对象的 TAB_FILE_ID 和 TBSP_NAME 与分区表中的数据分区或索引相对应，那么可以使用这种类型的锁定来控制脱机备份实用程序的并行性。

锁定转换

更改已挂起的锁定的方式称为*锁定转换*。

当进程访问它已锁定的数据对象，并且访问方式需要比所挂起的锁定更为严格的锁定时，将进行锁定转换。一个进程在任何给定时间对一个数据对象只能挂起一个锁定，尽管它可以通过查询间接地多次请求锁定同一个数据对象。

某些锁定方式仅适用于表，而有些锁定方式仅适用于行、块或数据分区。对于行或块，如果需要 X 锁定并且已挂起 S 或 U 锁定，那么通常将进行转换。

IX 和 S 锁定是锁定转换的特殊情况。这两种锁定被认为严格程度相当，所以，如果挂起一个锁定而需要另一个锁定，那么转换将生成 SIX（具有互斥意向的共享）锁定。如果所请求的方式更严格，那么所有其他转换都将导致请求的锁定方式变成挂起的锁定方式。

当查询更新行时，还可能会发生双重转换。如果通过索引访问来读取行，并且锁定方式为 S，那么包含该行的表带有覆盖意向锁定。但是，如果锁定类型是 IS 而不是 IX，并且随后更改该行，那么表锁定将转换为 IX，而行锁定将转换为 X。

执行查询时，锁定转换通常以隐式方式进行。系统监视元素 **lock_current_mode** 和 **lock_mode** 可提供有关数据库中发生的锁定转换的信息。

锁定等待和超时

锁定超时检测是一项数据库管理器功能，旨在防止应用程序无限地等待锁定被释放。

例如，一个事务可能正在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，并且未允许该应用程序落实将会释放该锁定的事务。为了避免在这种情况下停止应用程序，请将 **locktimeout** 数据库配置参数设置为任何应用程序应等待获取锁定的最长时间。

设置此参数有助于避免全局死锁，在分布式工作单元（DUOW）应用程序中尤其如此。如果锁定请求处于暂挂状态的时间长度大于 **locktimeout** 值，那么将返回一个错误给发出请求的应用程序并将其事务回滚。例如，如果 APPL1 尝试获取已由 APPL2 挂起的锁定，那么 APPL1 在超时时间段到期时将接收到 SQLCODE -911（SQLSTATE 40001），原因码为 68。**locktimeout** 的缺省值为 -1，这表示禁用锁定超时检测功能。

对于表、行、数据分区和多维集群（MDC）块锁定而言，应用程序可以通过更改 CURRENT LOCK TIMEOUT 专用寄存器的值来覆盖 **locktimeout** 的值。

要生成关于锁定超时的报告文件，请将 **DB2_CAPTURE_LOCKTIMEOUT** 注册表变量设置为 ON。锁定超时报告将包含关于导致锁定超时的锁定争用所涉及关键应用程序的信息以及关于锁定的详细信息，例如锁定名称、锁定类型、行标识、表空间标识和表标识。注意，由于已有通过使用 CREATE EVENT MONITOR FOR LOCKING 语句来收集锁定超时事件的新方法，因此建议您不要使用此变量，并且将来的发行版可能会将其除去。

要在 db2diag 日志文件中记录更多关于锁定请求超时的信息，请将 **diaglevel** 数据库管理器配置参数的值设置为 4。记录的信息包括被锁定对象的名称、锁定方式以及挂起该锁定的应用程序。还可能会记录当前的动态 SQL 或 XQuery 语句或静态程序包名称。仅当 **diaglevel** 为 4 时，才会记录动态 SQL 或 XQuery 语句。

您可以通过锁定等待信息系统监视元素或 **db.apps_waiting_locks** 运行状况指示器获取更多有关锁定等待数和锁定超时的信息。

指定锁定等待方式策略

会话可以指定锁定等待方式策略，该策略在会话需要无法立即获取的锁定时使用。

该策略指示会话是否将:

- 在无法获取锁定时返回 SQLCODE 和 SQLSTATE
- 无限等待锁定
- 为获取锁定等待一段指定的时间
- 等待锁定时使用 **locktimeout** 数据库配置参数的值

锁定等待方式策略通过 SET CURRENT LOCK TIMEOUT 语句指定，此语句更改 CURRENT LOCK TIMEOUT 专用寄存器的值。此专用寄存器指定在返回指示无法获取锁定的错误之前等待锁定的秒数。

传统的锁定方法会导致应用程序相互阻塞。当一个应用程序必须等待另一个应用程序释放其锁定时，就会发生阻塞。用于处理这种阻塞的影响的策略通常会提供一种机制来指定可接受的最大阻塞持续时间。这是应用程序在无法获取锁定的情况下返回之前等待的时间。以前，只能在数据库级别通过更改 **locktimeout** 数据库配置参数的值来指定此时间。

locktimeout 的值将应用于所有锁定，但是此锁定等待方式策略只影响下列锁定类型: 行、表、索引键和多维集群（MDC）块锁定。

死锁

死锁是指两个应用程序彼此锁定对方所需的数据，这将导致这两个应用程序都无法继续执行。

例如，在图 23 中，有两个同时运行的应用程序: 应用程序 A 和应用程序 B。应用程序 A 的第一个事务是更新表 1 的第一行，第二个事务是更新表 2 的第二行。应用程序 B 先更新表 2 的第二行，然后再更新表 1 的第一行。在时间点 T1，应用程序 A 锁定表 1 的第一行。同时，应用程序 B 锁定表 2 的第二行。在时间点 T2，应用程序 A 请求锁定表 2 的第二行。但是，与此同时，应用程序 B 尝试锁定表 1 的第一行。由于应用程序 A 要等到更新表 2 的第二行完成之后才会释放对表 1 中第一行的锁定，而应用程序 B 要等到更新表 1 的第一行完成之后才会释放对表 2 中第二行的锁定，因此将发生死锁。这两个应用程序都将等待对方释放对数据的锁定。

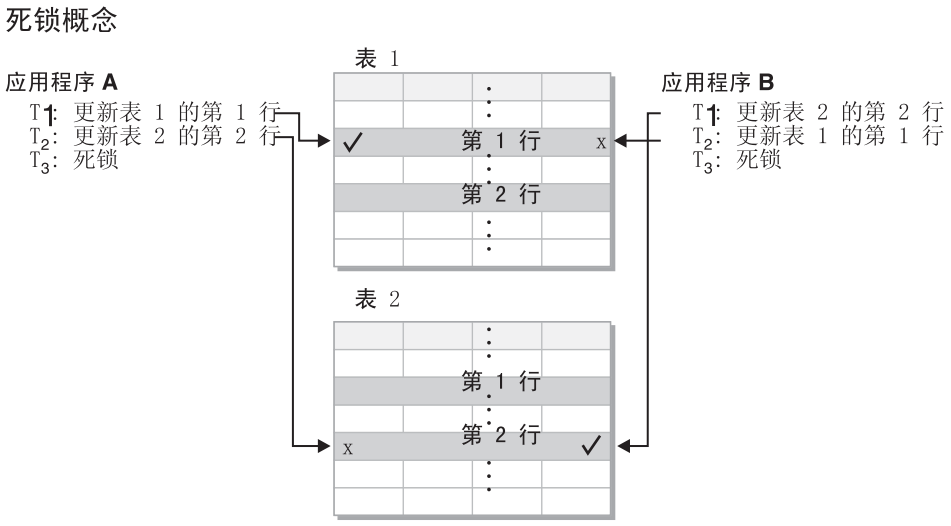


图 23. 应用程序之间的死锁

由于应用程序不会自动释放对它们所需的数据的锁定，因此需要死锁检测器进程来解决死锁情况。死锁检测器将监视关于正在等待锁定的代理程序的信息，并且将根据 **dlchktime** 数据库配置参数指定的时间间隔唤醒。

如果死锁检测器检测到死锁情况，那么它将任意选择一个已死锁的进程作为*牺牲进程*来回滚。牺牲进程将被唤醒并将 SQLCODE -911（SQLSTATE 40001）返回给调用应用程序，原因码为 2。数据库管理器将自动回滚所选进程中未落实的事务。回滚操作完成后，将释放属于牺牲进程的锁定，而该死锁所涉及的其他进程可以继续进行。

为了确保性能良好，请为 **dlchktime** 选择适当的值。过短的时间间隔将引起不必要的开销，而过长的时间间隔将导致死锁情况的持续时间延长。

在分区数据库环境中，**dlchktime** 的值仅应用于目录数据库分区。如果发生大量死锁，那么请增大 **dlchktime** 的值以满足锁定等待和通信等待的需要。

为了避免应用程序在读取它们打算随后更新的数据时发生死锁:

- 执行选择操作时，请使用 FOR UPDATE 子句。此子句确保在进程尝试读取数据时设置 U 锁定，并且不允许对行进行分块。
- 在查询中使用 WITH RR 或者 WITH RS 和 USE AND KEEP UPDATE LOCKS 子句。这些子句确保在进程尝试读取数据时设置 U 锁定，并且允许对行进行分块。

在联合系统中，应用程序所请求的数据可能会由于数据源处发生死锁而无法获得。发生这种情况时，DB2 服务器将依赖于数据源处的死锁处理工具。如果在多个数据源之间发生死锁，那么 DB2 服务器将依靠数据源超时机制来解决死锁情况。

要记录关于死锁的更多信息，请将 **diaglevel** 数据库管理器配置参数的值设置为 4。记录的信息包括被锁定对象的名称、锁定方式以及挂起该锁定的应用程序。还可能会记录当前的动态 SQL 和 XQuery 语句或静态程序包名称。

查询优化

查询优化是其中一项将会影响应用程序性能的因素。请查看本节的内容，以了解有关可以帮助您最大程度地提高数据库应用程序性能的查询优化注意事项的详细信息。

SQL 和 XQuery 编译器过程

SQL 和 XQuery 编译器执行多个步骤来生成可以执行的访问方案。

*查询图模型*是驻留在内存中的内部数据库，它代表按第 162 页的图 24 所示的步骤处理的查询（描述如下）。注意，某些步骤只适用于将对联合数据库运行的查询。

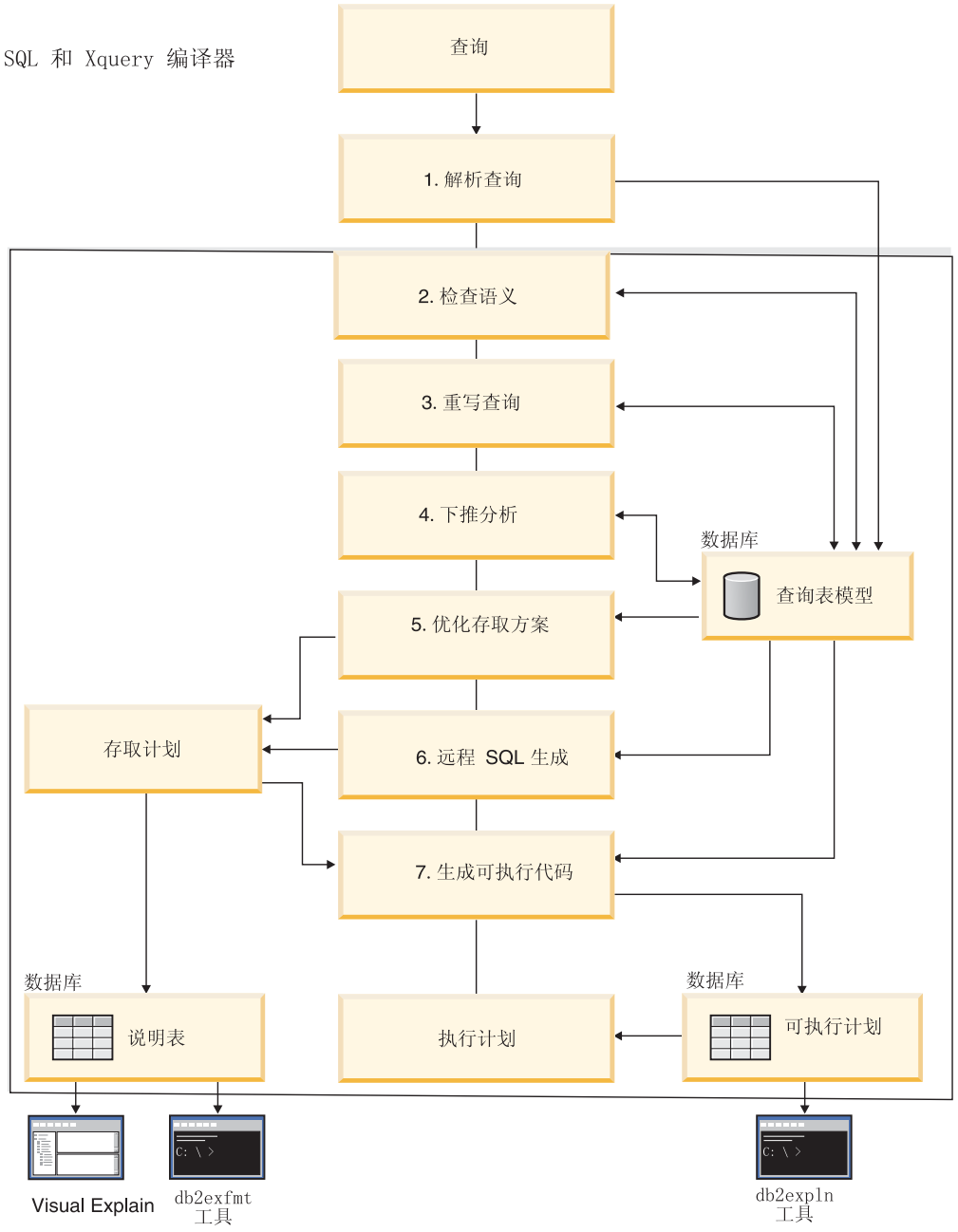


图 24. SQL 和 XQuery 编译器执行的步骤

1. 解析查询

   SQL 和 XQuery 编译器分析查询以验证其语法。如果检测到任何语法错误，那么查询编译器将停止处理，并将相应的错误返回至提交该查询的应用程序。解析完成时，将创建该查询的内部表示并将其存储在查询图模型中。

2. 检查语义

   编译器确保语句的各个部分之间不存在不一致情况。例如，编译器将验证对 YEAR 标量函数指定的列是否已被定义为具有日期时间数据类型。

编译器还会对查询图模型添加行为语义，其中包括引用约束、表检查约束、触发器和视图的作用。查询图模型包含查询的所有语义，其中包括查询块、子查询、关联、派生的表、表达式、数据类型、数据类型转换、代码页转换和分布键。

3. 重写查询

   编译器使用查询图模型中存储的全局语义将该查询变换为更易于优化的形式。然后，它将结果存储在查询图模型中。

   例如，编译器可能会移动谓词，更改其应用级别，从而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
   - 行的重新分布
   - 相关子查询，这些子查询包含对该子查询外部的表列的引用

   对于分区数据库环境中的某些查询而言，在重写查询的过程中，可能会发生解除相关。

4. 下推分析（仅限于联合数据库）

   此步骤中的主要任务是，向优化器建议是否可以在数据源处以远程方式对操作进行求值（即*下推*）。此类下推活动仅适用于数据源查询，这是对一般谓词下推操作的扩展。

5. 优化访问方案

   通过使用查询图模型作为输入，编译器的优化器部分将生成许多能够满足查询的备用执行方案。为了估算每个方案的执行成本，优化器将使用有关表、索引、列和函数的统计信息。然后，它选择估算执行成本最低的方案。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，其中包括索引、基本表、派生的表、子查询、关联和递归。

   优化器还可以考虑另一类下推操作，即*聚集与排序*，这可以将这些操作的求值下推到数据管理服务（DMS）组件，从而提高性能。

   在确定如何选择页大小时，优化器还会考虑是否有不同大小的缓冲池。它将考虑数据库是否分区或者是否可以选择对称多处理器（SMP）环境中的查询内并行性选项。优化器使用此信息来帮助选择该查询的最佳访问方案。

   此步骤的输出是一个访问方案，并且已在说明表中捕获关于此访问方案的详细信息。可以通过说明快照来捕获用于生成访问方案的信息。

6. 远程 SQL 生成（仅限于联合数据库）

   优化器选择的最终方案可以由一组对远程数据源执行操作的步骤组成。远程 SQL 生成步骤将根据数据源的 SQL 方言，为每个数据源执行的操作创建高效的 SQL 语句。

7. 生成可执行的代码

   在最后一个步骤中，编译器使用访问方案和查询图模型来创建查询的可执行访问方案或节。此代码生成步骤使用查询图模型中的信息，以避免重复执行只需计算一次的表达式。对于代码页转换以及在使用了主变量的情况下，有可能进行此类优化。

要对包含主变量、专用寄存器或参数标记的静态/动态 SQL 或 XQuery 语句启用查询优化或重新优化功能，在绑定程序包时，应该指定 REOPT 绑定选项。将使用主变量、专用寄存器或参数标记的值（而不是编译器选择的缺省估算值）来优化属于该程序包并包含这些变量的语句的访问路径。如果可以获得这些值，那么执行查询时就会进行此优化。

有关静态 SQL 和 XQuery 语句访问方案的信息存储在系统目录表中。执行程序包时，数据库管理器将使用系统目录中存储的信息来确定如何访问该数据以及提供该查询的结果。此信息由 db2expln 工具使用。

注: 按适当的时间间隔对经常更改的表执行 RUNSTATS 命令。优化器需要有关表及其数据的最新统计信息以创建最高效的访问方案。请重新绑定应用程序以利用更新后的统计信息。如果未执行 RUNSTATS，或者优化器假定您对空表或几乎为空的表执行此命令，那么它可能使用缺省值或者尝试根据磁盘上存储该表所用的文件页数来推断特定的统计信息。另请参阅“自动收集统计信息”。

查询重写方法和示例

在查询重写阶段，查询编译器将 SQL 和 XQuery 语句变换为更易于优化的格式；这有助于改进可能的访问方案。重写查询对于非常复杂的查询（包括那些包含许多子查询或许多连接的查询）而言特别重要。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 或 XQuery 语句的查询重写规则的数目，请更改优化类。要查看查询重写过程的某些结果，请使用说明工具或 Visual Explain。

可以通过下列任何一种方法来重写查询:

- 操作合并

  为了构造所含操作（尤其是 SELECT 操作）数目最少的查询，SQL 和 XQuery 编译器将重写查询以合并查询操作。下列示例举例说明一些可合并的操作:

  - 示例 - 视图合并

    使用视图的 SELECT 语句可能会限制表的连接顺序，并有可能引入冗余的表连接。如果在查询重写期间合并视图，那么可撤销这些限制。

  - 示例 - 从子查询到连接的变换

    如果 SELECT 语句包含子查询，那么可能会限制对表的顺序处理所作的选择。

  - 示例 - 消除冗余连接

    在查询重写期间，可以除去冗余的连接，从而简化 SELECT 语句。

  - 示例 - 共享聚集

    如果查询使用不同的函数，那么重写可减少需要执行的计算数。

- 操作移动

  为了构造所含操作和谓词数目最少的查询，编译器将重写查询以移动查询操作。下列示例举例说明一些可移动的操作:

  - 示例 - 消除 DISTINCT

    在查询重写期间，优化器可以移动 DISTINCT 操作的执行位置，从而降低此操作的成本。在某些情况下，可以将 DISTINCT 操作完全除去。

– 示例 - 常规谓词下推

在查询重写期间，优化器可以更改谓词的应用顺序，以便尽早将选择性更强的谓词应用于查询。

– 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。减小必须广播至其他数据库分区的内容的大小并且/或者减少广播次数，是查询重写过程的目标。

- 谓词转换

SQL 和 XQuery 编译器将重写查询，以便将现有谓词转换为更优的形式。下列示例举例说明一些可转换的谓词:

– 示例 - 添加隐式谓词

在查询重写期间，可以对查询添加谓词，以使优化器在选择该查询的最佳访问方案时能够考虑其他表连接。

– 示例 - 从 OR 到 IN 的变换

在查询重写期间，可以将 OR 谓词转换为 IN 谓词，以便提高访问方案的效率。SQL 和 XQuery 编译器还可以将 IN 谓词转换为 OR 谓词，但前提是此变换将创建效率更高的访问方案。

编译器重写示例: 视图合并:

使用视图的 SELECT 语句可能会限制表的连接顺序，并有可能引入冗余的表连接。如果在查询重写期间合并视图，那么可撤销这些限制。

假定您可以访问下面这两个基于 EMPLOYEE 表的视图: 一个视图显示受过高等教育的职员，另一个视图显示年薪高于 $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
```

以下用户编写的查询将列示受过高等教育并且年薪高于 $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，可以合并这两个视图以创建以下查询:

```
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，优化器在选择访问方案时将有更多的选项可供考虑。此外，如果已合并的这两个视图使用同一个基本表，那么还可以另外进行重写。

示例 - 从子查询到连接的变换

SQL 和 XQuery 编译器将接收一个包含子查询的查询，例如:

```
select empno, firstnme, lastname, phoneno
  from employee
  where workdept in
    (select deptno
      from department
      where deptname = 'OPERATIONS')
```

并将其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'
```

通常，连接的执行效率高于子查询。

示例 - 消除冗余连接

有时，查询包含不必要的连接。

```
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
```

SQL 和 XQuery 编译器可以消除连接并将此查询简化为:

```
select empno, firstnme, lastname, edlevel, salary
  from employee
  where
    edlevel > 17 and
    salary  > 35000
```

以下示例假定在 EMPLOYEE 与 DEPARTMENT 表之间存在基于部门号的引用约束。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
```

于是，类似于以下的查询:

```
select lastname, salary
  from peplview
```

将变为:

```
select lastname, salary
  from employee
  where workdept not null
```

注意，在这种情况下，即使您知道可以重写查询，您也可能因为无权访问基础表而无法进行重写。您可能只对以上所示的视图具有访问权。所以，此类优化必须由数据库管理器执行。

在下列情况下，引用完整性连接中有可能出现冗余情况:

- 通过连接来定义视图
- 自动生成查询

示例 - 共享聚集

在一个查询中使用多个函数可能会导致多次执行计算，这将耗用时间。减少所需的计算次数有助于改进访问方案。编译器接收使用多个函数的查询，例如:

```
select sum(salary+bonus+comm) as osum,
    avg(salary+bonus+comm) as oavg,
    count(*) as ocount
  from employee
```

并将其变换为:

```
select osum, osum/ocount ocount
  from (
    select sum(salary+bonus+comm) as osum,
        count(*) as ocount
      from employee
  ) as shared_agg
```

经过此重写后，所需执行的 sum 和 count 次数均减半。

编译器重写示例: 消除 DISTINCT:

在查询重写期间，优化器可以移动 DISTINCT 操作的执行位置，从而降低此操作的成本。在某些情况下，可以将 DISTINCT 操作完全除去。

例如，如果 EMPLOYEE 表的 EMPNO 列被定义为主键，那么以下查询:

```
select distinct empno, firstnme, lastname
  from employee
```

通过除去 DISTINCT 子句，可以重写为:

```
select empno, firstnme, lastname
  from employee
```

在此示例中，由于正在选择主键，因此编译器知道返回的每一行都唯一。在这种情况下，DISTINCT 关键字多余。如果不重写该查询，那么优化器将必须构建包含必需处理（例如排序）的方案，以确保列值唯一。

示例 - 常规谓词下推

更改谓词的正常应用级别可提高性能。例如，以下视图提供部门 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'
```

对此视图执行的以下查询未充分发挥效率:

```
select firstnme, phoneno
  from d11_employee
  where lastname = 'BROWN'
```

在查询重写期间，编译器将 lastname = 'BROWN' 谓词下推到 D11_EMPLOYEE 视图。这使得可以更快并可能更高效地应用该谓词。在此示例中，可能执行的实际查询如下所示:

```
select firstnme, phoneno
  from employee
  where
    lastname = 'BROWN' and
    workdept = 'D11'
```

谓词下推并不限于视图。可以下推谓词的其他情况包括 UNION、GROUP BY 和派生的表（嵌套表表达式或公共表表达式）。

示例 - 解除相关

在分区数据库环境中，编译器可以重写以下查询（用于查找所有服务于编程项目并且所得报酬偏低的职员）。

```
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)
```

由于此查询是相关的，并且因为 PROJECT 和 EMPLOYEE 不可能根据 PROJNO 进行分区，因此有可能将每个项目广播至每个数据库分区。并且，子查询将不得不进行多次求值。

编译器可以按如下方式重写此查询:

- 确定服务于程序设计项目的职员的相异值列表并将其命名为 DIST_PROJS。这必须是相异值列表，以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%')
  ```

- 将 DIST_PROJS 与 EMPLOYEE 表连接，以获得每个项目的平均补偿额 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)
  ```

- 重写后的查询是:

  ```
  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
```

此查询计算每个项目的 avg_comp（avg_per_proj）。然后，可以将结果广播到所有包含 EMPLOYEE 表的数据库分区。

编译器重写示例: 组合型 SQL/XQuery 语句的谓词下推:

优化关系 SQL 查询的一项基本技术是，将外层查询块的 WHERE 子句中的谓词移入所包含的较低层查询块（例如视图），从而能够提早进行数据过滤以及有可能更好地利用索引。

这在分区数据库环境中甚至更为重要，其原因在于，提早进行过滤有可能减少必须在数据库分区之间传递的数据量。

类似的技术可用于移动 XQuery 中的谓词或 XPath 过滤器。基本策略是，始终将过滤表达式尽可能移至靠近数据源的位置。此优化技术在 SQL 中被称为*谓词下推*，在 XQuery 中被称为*抽取下推*（用于过滤器和 XPath 抽取）。

由于 SQL 和 XQuery 所利用的数据模型有所不同，因此必须跨这两种语言之间的边界来移动谓词、过滤器或抽取。将 SQL 谓词变换为语义等同的过滤器并将其下推到 XPath 抽取时，必须考虑数据映射和强制类型转换规则。下列示例阐述如何将关系谓词下推到 XQuery 查询块。

请考虑下面这两个包含客户信息的 XML 文档:

文档 1

```
<customer>
  <name>John</name>
  <lastname>Doe</lastname>
  <date_of_birth>
   1976-10-10
  </date_of_birth>
  <address>
    <zip>95141.0</zip>
  </address>
<volume>80000.0</volume>
</customer>
<customer>
  <name>Jane</name>
  <lastname>Doe</lastname>
  <date_of_birth>
    1975-01-01
  </date_of_birth>
  <address>
    <zip>95141.4</zip>
  </address>
<volume>50000.00</volume>
</customer>
```

文档 2

```
<customer>
  <name>Michael</name>
  <lastname>Miller </lastname>
  <date_of_birth>
  1975-01-01
  </date_of_birth>
  <address>
    <zip>95142.0</zip>
  </address>
<volume>100000.00</volume>
</customer>
<customer>
  <name>Michaela</name>
  <lastname>Miller</lastname>
  <date_of_birth>
   1980-12-23
  </date_of_birth>
  <address>
    <zip>95140.5</zip>
  </address>
<volume>100000</volume>
</customer>
```

示例 - 推送整型谓词

请考虑以下查询:

```
select temp.name, temp.zip
  from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
    columns name varchar(20) path 'customer/name',
    zip integer path 'customer/zip'
    ) as temp
  where zip = 95141
```

为了使用基于 T.XMLDOC 的可能索引以及提早过滤掉不想要的人员，zip = 95141 谓词将以内部方式转换为以下等同的 XPATH 过滤表达式:

```
T.XMLCOL/customer/zip[. >= 95141.0 and . < 95142.0]
```

由于编译器不使用 XML 段的模式信息，因此不能假定 ZIP 只包含整数。有可能存在其他具有小数部分的数字值以及基于此特定 XPath 抽取的相应双精度 XML 索引。XML2SQL 强制类型转换功能将处理此变换，即，先截断小数部分，然后再将值的类型强制转换为 INTEGER。此行为必须在下推过程中有所反映，并且必须更改该谓词以保持语义正确。

示例 - 推送 VARCHAR(*n*) 谓词

请考虑以下查询:

```
select temp.name, temp.lastname
  from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
    columns name varchar(20) path 'customer/name',
    lastname varchar(20) path 'customer/lastname'
    ) as temp
  where lastname = 'Miller'
```

为了使用基于 T.XMLDOC 的可能索引以及提早过滤掉不想要的人员，lastname = 'Miller' 谓词将以内部方式转换为以下等同的 XPATH 过滤表达式:

```
T.XMLCOL/customer/lastname[. > rtrim("Miller")
and . < blank_padd("Miller", max(20,length("Miller"))]
```

在 SQL 中，处理尾部空格的方式与 XPath 或 XQuery 有所不同。原始 SQL 谓词将不会对名字均为“Miller”的两个客户加以区分，即使他们中的一个（Michael）带有尾部空格亦如此。因此，这两个客户都将被返回，这与下推未更改的谓词时的情况不同。

解决方案是，将该谓词变换为范围过滤器。

- 通过使用 RTRIM() 函数从比较值中截断所有尾部空格，创建第一个边界。
- 第二个边界必须大于或等于所有包含尾部空格的可能“Miller”字符串。将对原始字符串填充空格，以使其长度达到最大列长度或者比较字符串的长度（以较长者为准）。

示例 - 推送 DECIMAL(*x*,*y*) 谓词

请考虑以下查询:

```
select temp.name, temp.volume
  from xmltable('db2-fn:xmlcolumn("T.XMLDOC")'
    columns name varchar(20) path 'customer/name',
    volume decimal(10,2) path 'customer/volume'
    ) as temp
  where volume = 100000.00
```

为了使用基于 T.XMLDOC 的可能双精度索引以及提早过滤掉不想要的人员，volume = 100000.00 谓词将以内部方式转换为以下等同的 XPATH 过滤表达式:

```
T.XMLCOL/customer/volume[.=100000.00]
```

不必将此谓词变换为范围过滤器，这是因为，强制类型转换限制强制使 XML 值的小数部分精度和长度与目标 SQL 类型相同。违反此约束将返回错误。将 DOUBLE 值强制转换为 DECIMAL(*x*,*y*) 时，精度不会下降。不必对比较值进行取整或截断。

编译器重写示例：隐式谓词:

在查询重写期间，可以对查询添加谓词，以使优化器在选择该查询的最佳访问方案时能够考虑其他表连接。

以下查询返回一个列表，该列表包含其部门向部门 E01 报告的经理以及那些经理所负责的项目:

```
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
```

可以使用以下隐式谓词（称为*传递闭包的谓词*）来重写此查询:

```
dept.mgrno = proj.respemp
```

现在，优化器在尝试选择该查询的最佳访问方案时，可以考虑其他连接。

在查询重写阶段，将根据等式谓词所暗指的可转换性来派生其他局部谓词。例如，以下查询将返回部门号大于 E00 的部门以及在这些部门工作的职员的姓名。

```
select empno, lastname, firstname, deptno, deptname
  from employee emp, department dept
  where
    emp.workdept = dept.deptno and
    dept.deptno > 'E00'
```

可以使用以下隐式谓词来重写此查询:

```
emp.workdept > 'E00'
```

此重写将减少需要连接的行数。

示例 - 从 OR 到 IN 的变换

假定 OR 子句将两个或更多个应用于同一列的简单等式谓词连接到一起，如以下示例所示:

```
select *
  from employee
  where
    deptno = 'D11' or
    deptno = 'D21' or
    deptno = 'E21'
```

如果没有基于 DEPTNO 列的索引，那么使用 IN 谓词代替 OR 有助于更高效地处理此查询:

```
select *
  from employee
  where deptno in ('D11', 'D21', 'E21')
```

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，以便能够执行索引 OR 运算。

谓词类型和访问方案

*谓词*是搜索条件的元素，用于表达或暗指比较运算。谓词通常出现在查询的 WHERE 子句中，用于缩小查询所返回的结果集的范围。

根据求值过程如何以及何时使用谓词，可以将谓词分为四类。这些类别列示如下（按性能最佳到最差排列）：

1. 范围定界谓词
2. 索引 SARGable 谓词
3. 数据 SARGable 谓词
4. 残留谓词

术语 *SARGable* 是可用作*搜索自变量*的术语。

表 51 对这些谓词类别的特征作了摘要。

表 51. 谓词类型的特征摘要

特征	谓词类型			
	范围定界	索引 SARGable	数据 SARGable	残留
减少索引 I/O	是	否	否	否
减少数据页 I/O	是	是	否	否
减少内部传递的行数	是	是	是	否
减少合格行的数目	是	是	是	是

范围定界谓词和索引 SARGable 谓词

范围定界谓词用于限制索引扫描范围。它们提供索引搜索的开始和停止键值。索引 SARGable 谓词无法限制搜索范围，但可根据索引进行求值，这是因为该谓词中引用的列是索引键的组成部分。例如，考虑以下索引:

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

另外，考虑包含以下 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 SARGable 谓词。

优化器对这些谓词进行求值时，将使用索引数据，而不是读取基本表。索引 SARGable 谓词可减少需要从表中读取的行数，但不会影响所访问的索引页数。

数据 SARGable 谓词

无法由索引管理器求值但可以由数据管理服务（DMS）求值的谓词称为数据 SARGable 谓词。这些谓词通常需要访问表中的各行。有需要时，DMS 将检索对该谓词进行求值所需的列，并且将检索任何其他由 SELECT 列表所需但未能从索引中获取的列。

例如，考虑对 PROJECT 表定义的以下索引:

```
INDEX IX0:  PROJNO ASC
```

在以下查询中，deptno = 'D11' 被视为数据 SARGable 谓词。

```
select projno, projname, respemp
  from project
  where deptno = 'D11'
  order by projno
```

残留谓词

在 I/O 成本方面，残留谓词的成本比访问表更高。这样的谓词可以:

- 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，此数据与表存储在不同的文件中

这种谓词由关系数据服务（RDS）进行求值。

某些只应用于索引的谓词在数据页被访问时必须进行重新应用。例如，当数据页被访问时，使用索引 OR 运算或索引 AND 运算的访问方案始终将这些谓词作为残留谓词进行重新应用。

联合数据库查询编译器阶段

联合数据库下推分析:

对于要对联合数据库运行的查询，优化器将执行下推分析以确定能否在远程数据源处执行特定操作。

操作可以是函数，例如关系运算符、系统函数或用户函数；操作也可以是 SQL 运算符，例如 ORDER BY 或 GROUP BY。

务必定期更新本地目录信息，以使 DB2 查询编译器能够访问关于远程数据源处的 SQL 支持的准确信息。请使用 DB2 数据定义语言（DDL）语句（例如 CREATE FUNCTION MAPPING 或 ALTER SERVER）来更新目录。

如果无法将函数下推到远程数据源，那么它们可能会显著影响查询性能。请考虑强制在本地而不是数据源处对选择性谓词进行求值的影响。此类求值可能要求 DB2 服务器从远程数据源检索整个表，然后在本地根据谓词对其进行过滤。网络约束和大型表可能会导致性能下降。

未下推的运算符也可能显著影响查询性能。例如，让 GROUP BY 运算符在本地聚集远程数据也会要求 DB2 服务器从远程数据源检索整个表。

例如，假定昵称 N1 引用 DB2 z/OS 版数据源中的数据源表 EMPLOYEE。此表包含 10000 行，其中一列包含职员的姓，另一列包含薪水。优化器在处理以下语句时有多个选项，这取决于本地整理顺序与远程整理顺序是否相同:

```
select lastname, count(*)  from n1
  where
    lastname > 'B' and
    salary > 50000
  group by lastname
```

- 如果整理顺序相同，那么有可能可以将查询谓词下推到 DB2 z/OS 版。在数据源处对结果进行过滤和分组通常比复制整个表并在本地执行操作更高效。对于此查询而言，谓词和 GROUP BY 操作可以在数据源处执行。
- 如果整理顺序不同，那么无法在数据源处对这两个谓词进行求值。但是，优化器可能会决定下推 `salary > 50000` 谓词。范围比较仍必须在本地进行。
- 如果整理顺序相同，并且优化器知道本地 DB2 服务器非常快，那么优化器可能会确定在本地执行 GROUP BY 操作是成本最低的方法。该谓词将在数据源处进行求值。这是下推分析与全局优化相结合的示例。

通常，目标是确保优化器在远程数据源处对函数和运算符进行求值。许多因素将影响到能否在远程数据源处对函数或 SQL 运算符求值，其中包括:

- 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。通常，存在这些因素的原因是 DB2 产品支持丰富的 SQL 方言。DB2 数据服务器可弥补另一数据服务器在功能方面的不足，但这样做可能要求在 DB2 服务器中执行操作。

- 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，大部分数据源都支持 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 列表中是否允许表达式有所限制。如果远程数据源处存在限制，那么 DB2 服务器可能必须在本地执行 GROUP BY 操作。

- SQL 约束

  每个数据源的 SQL 限制可能有所不同。例如，某些数据源需要参数标记才能将值与远程 SQL 语句绑定。因此，必须检查参数标记限制以确保每个数据源可支持此类绑定机制。如果 DB2 服务器无法确定一种好方法来绑定函数的值，那么必须在本地对此函数进行求值。

- SQL 限制

  虽然 DB2 服务器可能允许使用超出远程数据源所允许限制的整数，但不能在将要发送到数据源的语句中嵌入超出远程限制的值，并且必须在本地对任何受影响的函数或运算符进行求值。

- 服务器细节

  此类别包含多个因素。例如，如果数据源与 DB2 服务器以不同的方式对空值进行排序，那么对可空表达式执行的 ORDER BY 操作无法以远程方式进行求值。

- 整理顺序

检索数据以进行本地排序和比较通常会导致性能下降。如果将联合数据库配置为使用数据源所使用的整理顺序，然后将 COLLATING_SEQUENCE 服务器选项设置为 Y，那么优化器可考虑下推许多查询操作。如果整理顺序相同，那么可以下推下列操作:

– 字符或数字数据的比较
– 字符范围比较谓词
– 排序

但是，如果联合数据库与数据源之间空字符的加权有所不同，那么可能会产生不正常的结果。如果将语句提交到不区分大小写的数据源，那么比较操作可能会返回意外的结果。在不区分大小写的数据源中，对字符“I”和“i”指定的权重是相同的。缺省情况下，DB2 服务器区分大小写，并且将对这些字符指定不同的权重。

为了提高性能，联合服务器允许在数据源处执行排序和比较。例如，在 DB2 z/OS 版中，ORDER BY 子句定义的排序由基于 EBCDIC 代码页的整理顺序实现。要使用联合服务器来检索根据 ORDER BY 子句排序的 DB2 z/OS 版数据，请配置联合数据库以使其使用基于 EBCDIC 代码页的预定义整理顺序。

如果联合数据库与数据源的整理顺序不同，那么 DB2 服务器会将数据检索到联合数据库。由于用户希望看到按照对联合服务器定义的整理顺序排序的查询结果，所以在本地对数据进行排序将确保实现此期望。如果您需要查看以数据源的整理顺序排序的数据，那么请以传递方式提交查询或者在数据源视图中定义该查询。

- 服务器选项

  多个服务器选项会影响下推机会，其中包括 COLLATING_SEQUENCE、VARCHAR_NO_TRAILING_BLANKS 和 PUSHDOWN。

- DB2 类型映射和函数映射因素

  DB2 服务器上的缺省本地数据类型映射能够为每种数据源数据类型提供足够的缓冲区空间，从而避免丢失数据。您可以为特定数据源定制类型映射以适合于特定的应用程序。例如，如果要访问具有 DATE 数据类型（缺省情况下，它映射至 DB2 TIMESTAMP 数据类型）的 Oracle 数据源列，那么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 服务器可以补偿数据源不支持的函数:

- 此函数在远程数据源中不存在。
- 此函数存在，但操作数特征违反函数限制。IS NULL 关系运算符是此情况的一个示例。大部分数据源支持此运算符，但某些数据源可能有限制，例如只允许列名出现在 IS NULL 运算符左边。
- 如果以远程方式对该函数进行求值，那么它可能返回另一结果。大于（“>”）运算符是这种情况的一个示例。对于具有不同整理顺序的数据源而言，如果由 DB2 服务器在本地对大于运算符进行求值，那么此运算符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

请确保列的本地数据类型不会妨碍在数据源处对谓词进行求值。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 以何方式绑定较长的列。这种情况将影响由 DB2 优化器确定的连接顺序中存在的可能性数目。例如，使用 INTEGER 或 INT 数据类型创建的 Oracle 数据源列将被赋予类型 NUMBER(38)。此 Oracle 数据类型的昵称列将被赋予本地数据类型 FLOAT，其原因在于，DB2 整数的范围是 2**31 到 (-2**31)-1，这大致等同于 NUMBER(9)。在这种情况下，无法在 DB2 数据源处将 DB2 整数列与 Oracle 整数列连接（这是由于连接列较短）；但是，如果 DB2 INTEGER 数据类型可容纳此 Oracle 整数列的域，那么请使用 ALTER NICKNAME 语句来更改其本地数据类型，以便可以在 DB2 数据源处进行该连接。

- 列选项

  使用 ALTER NICKNAME 语句来添加或更改昵称的列选项。

  使用 VARCHAR_NO_TRAILING_BLANKS 选项来标识未包含尾部空格的列。在检查对这种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。DB2 服务器可能生成另一个形式等同的谓词，以便用于将要发送到数据源的远程 SQL 语句。您可能会看到依靠数据源进行求值的另一个谓词，但最终结果应该会等同。

  使用 NUMERIC_STRING 选项来指示该列中的值是否总是不带尾部空格的数值。

  表 52 描述这些选项。

表 52. 列选项及其设置

选项	有效设置	缺省设置
NUMERIC_STRING	Y: 指定此列只包含数字数据串。此列不包含可能干扰列数据排序的空白字符。当数据源的整理顺序与 DB2 服务器的不同时，此选项很有用。使用此选项标记的那些列不会因为整理顺序不同而被本地（数据源）求值过程排除。如果此列只包含后跟尾部空格字符的数字串，那么请不要指定 Y。 N: 指定此列并非只包含数字数据串。	N
VARCHAR_NO_TRAILING_BLANKS	Y: 指定此数据源使用非空白填充 VARCHAR 比较语义，这与 DB2 数据服务器类似。对于不包含尾部空格字符的可变长度字符串，某些数据服务器的非空白填充比较语义返回与 DB2 比较语义相同的结果。如果您确定数据源处的所有 VARCHAR 表或视图列都不包含尾部空格字符，那么请指定此值。 N: 指定此数据源不使用非空白填充 VARCHAR 比较语义，这与 DB2 数据服务器类似。	N

影响下推机会的查询特征

查询所引用的 SQL 运算符可以涉及来自多个数据源的昵称。此操作必须在 DB2 服务器上执行，以便对来自两个使用一个运算符（例如集合运算符 UNION）的被引用数据源的结果进行组合。不能在远程数据源处直接对此运算符进行求值。

有关确定在何处对联合查询进行求值的准则:

通过调用 db2expln 命令启动的 DB2 说明实用程序将显示查询的求值位置。每个运算符的执行位置都包括在命令输出中。

- 如果查询被下推，那么您应该会看到 RETURN 运算符（这是一个标准的 DB2 运算符）。如果 SELECT 语句从昵称检索数据，那么您还会看到联合数据库操作所特有的 SHIP 运算符: 它更改数据流的服务器属性，并且将本地运算符与远程运算符分隔开。该 SELECT 语句是使用数据源所支持的 SQL 方言生成的。
- 如果可以完全将 INSERT、UPDATE 或 DELETE 语句下推至远程数据源，那么 SHIP 运算符可能不会出现在访问方案中。对于 RETURN 运算符，将显示所有以远程方式执行的 INSERT、UPDATE 或 DELETE 语句。但是，如果无法完全下推查询，那么 SHIP 运算符将显示以远程方式执行的操作。

了解查询为何由数据源而不是 DB2 服务器进行求值

在寻求增加下推机会的方法时，请考虑以下关键问题:

- 为何未对此谓词进行远程求值？

  当使用选择性很强的谓词来过滤行并减少网络流量时，会出现此问题。远程谓词求值还将影响能否对同一数据源的两个表之间的连接进行远程求值。

  要检查的领域包括:
  - 子查询谓词。此谓词是否包含与另一个数据源相关的子查询？此谓词是否包含涉及此数据源不支持的 SQL 运算符的子查询？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。此谓词是否包含此远程数据源无法求值的函数？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，此谓词是否要求绑入某些值？是否会违反此数据源的 SQL 限制？
  - 全局优化。优化器可能已经确定本地处理更节省成本。
- 为何不对 GROUP BY 运算符进行远程求值？

  要检查的领域包括:
  - 是否已对 GROUP BY 运算符的输入进行远程求值？如果回答是否定的，那么请检查输入。
  - 数据源是否对此运算符有任何限制？示例包括:
    - GROUP BY 的项数受限
    - 组合的 GROUP BY 项的字节数受限
    - 在 GROUP BY 列表中只能指定列
  - 数据源是否支持此 SQL 运算符？
  - 全局优化。优化器可能已经确定本地处理更节省成本。
  - GROUP BY 子句是否包含字符表达式？如果包含，那么请验证远程数据源是否与 DB2 服务器一样区分大小写。
- 为何未对集合运算符进行远程求值？

  要检查的领域包括:

- 它的两个操作数是否完全在同一远程数据源中进行求值？答案应该是肯定的，如果不是，那么请检查每个操作数。
- 该数据源是否对此集合运算符有任何限制？例如，大对象（LOB）或 LONG 字段数据是否是此特定集合运算符的有效输入？

- 为何未对 ORDER BY 操作进行远程求值？

  要检查的领域包括:

  - 是否已对 ORDER BY 操作的输入进行远程求值？如果回答是否定的，那么请检查输入。
  - ORDER BY 子句是否包含字符表达式？如果包含，那么远程数据源的整理顺序或区分大小写性是否与 DB2 服务器不同？
  - 远程数据源是否对此运算符有任何限制？例如，ORDER BY 的项数是否受限制？远程数据源是否限制 ORDER BY 列表中只能指定列？

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，访问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。这样的远程 SQL 生成有助于为查询生成全局优化访问策略。

优化器使用下推分析输出来确定是在 DB2 服务器上以本地方式对每个操作进行求值，还是在数据源处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，该输出不仅包括对操作进行求值的成本，还包括在 DB2 服务器与远程数据源之间传递数据和消息的成本。

虽然目标是生成经过优化的查询，但下列因素将显著影响全局优化，从而影响查询性能。

- 服务器特征
- 昵称特征

影响全局优化的服务器选项

下列数据源服务器选项可能影响全局优化:

- 处理速度的相对比率

  使用 CPU_RATIO 服务器选项来指定，数据源处理速度相对于 DB2 服务器处理速度的快慢。比率较低表明数据源处理速度高于 DB2 服务器处理速度；在这种情况下，DB2 优化器很可能考虑将需要耗用大量处理器资源的操作下推到数据源。

- I/O 速度的相对比率

  使用 IO_RATIO 服务器选项来指定，数据源系统 I/O 速度相对于 DB2 服务器系统 I/O 速度的快慢。比率较低表明数据源 I/O 速度高于 DB2 服务器 I/O 速度；在这种情况下，DB2 优化器很可能考虑将需要耗用大量 I/O 资源的操作下推到数据源。

- DB2 服务器与数据源之间的通信速率

  使用 COMM_RATE 服务器选项来指定网络容量。低速率（这表明 DB2 服务器与数据源之间的网络通信速度较慢）将促使 DB2 优化器减少向此数据源发送或从此数据源接收的消息数。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访问方案。

- 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，那么 DB2 优化器将认为任何从此数据源检索到的数据均未经排序。

- 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。缺省情况下，DB2 服务器不将任何方案提示发送至数据源。

  方案提示是用于向数据源处的优化器提供额外信息的语句片段。对于某些查询而言，此信息可以提高性能。方案提示可帮助数据源处的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，那么发送到数据源的查询将包含其他信息。例如，发送到 Oracle 优化器的带有方案提示的语句可能如下所示:

  ```
  select /*+ INDEX (table1, t1index)*/
      col1
    from table1
  ```

  方案提示是字符串: `/*+ INDEX (table1, t1index)*/`

- DB2 优化器知识库中的信息

  DB2 服务器有一个包含本机数据源数据的优化器知识库。DB2 优化器不会生成特定数据库管理系统（DBMS）无法生成的远程访问方案。换而言之，DB2 服务器将避免生成远程数据源处的优化器无法理解或接受的方案。

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。

- 索引考虑事项

  DB2 服务器可以使用关于数据源处的索引的信息来优化查询。因此，可用的索引信息应该是最新的，这一点至关重要。昵称的索引信息最初是在昵称被创建时获取的。视图昵称的索引信息不会被收集。

- 创建昵称的索引规范

  您可以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。请使用 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基于本地表创建索引的语法相似。在下列情况下，请考虑创建索引规范:

  – 当 DB2 服务器在昵称创建期间无法从数据源检索任何索引信息时
  – 当需要视图昵称的索引时
  – 当您想促使 DB2 优化器将特定昵称用作嵌套循环连接的内表时。如果不存在基于连接列的索引，那么您可以进行创建。

  对视图的昵称发出 CREATE INDEX 语句之前，请考虑是否需要索引。如果该视图是对带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的索引匹配的索引可显著提高查询性能。但是，如果在本地基于并非简单 SELECT 语句的视图（例如，通过连接两个表创建的视图）创建索引，那么可能会导致查询性能下降。例如，如果基于两个表之间的连接所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元素。该查询的性能将会很差，这是因为要对该连

接进行多次求值。另一种方法是，为数据源视图中引用的每个表创建昵称，然后在 DB2 服务器上创建引用这两个昵称的局部视图。

- 目录统计信息考虑事项

  系统目录统计信息描述相关列中昵称的总大小以及值的范围。优化器在计算用于处理包含昵称的查询的最低成本路径时，将使用这些统计信息。昵称统计信息与表统计信息存储在同一个目录视图中。

  虽然 DB2 服务器可以检索数据源中存储的统计数据，但它无法自动检测对该数据的更新。并且，DB2 服务器无法自动检测对数据源处对象的定义所作的更改。如果对象的统计数据或定义已更改，那么您可以执行下列操作:

  – 在数据源处运行与 RUNSTATS 命令等同的命令，删除当前昵称，然后重新创建该昵称。如果对象的定义已更改，那么请使用此方法。
  – 手动更新 SYSSTAT.TABLES 目录视图中的统计信息。此方法需要的步骤较少，但如果对象的定义已更改，那么此方法无效。

联合数据库查询的全局分析:

您可以通过调用 db2expln 命令来启动 DB2 说明实用程序，此实用程序将显示远程优化器为远程说明功能所支持的那些数据源生成的访问方案。每个运算符的执行位置都包括在命令输出中。

根据查询类型，您还可以在 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 优化器为每个运算符的输入和输出估算的行数。

了解 DB2 优化决策

在寻求提高性能的方法时，请考虑以下关键问题:

- 为何未对同一数据源的两个昵称之间的连接进行远程求值？

  要检查的领域包括:

  – 连接操作。远程数据源能否支持这些操作？
  – 连接谓词。能否在远程数据源处对连接谓词进行求值？如果回答是否定的，那么检查连接谓词。

- 为何未对 GROUP BY 运算符进行远程求值？

  请检查运算符语法并验证能否在远程数据源处对该运算符进行求值。

- 为何远程数据源未对该语句进行完全求值？

  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以在远程数据源处进行求值，优化器也根据其估算的成本来生成全局优化方案。有很多因素会对该方案产生影响。例如，即使远程数据源能够处理原始查询中的每项操作，但其处理速度可能远低于 DB2 服务器的处理速度，因此在 DB2 服务器上执行这些操作可能更好。如果结果不理想，那么请验证 SYSCAT.SERVEROPTIONS 目录视图中的服务器统计信息。

- 一个由优化器生成并且完全在远程数据源处进行求值的方案，其性能为何比直接在远程数据源处执行的原始查询的性能差很多？

要检查的领域包括:

– DB2 优化器生成的远程 SQL 语句。请确保此语句与原始查询完全相同。检查谓词顺序方面的更改。良好的查询优化器应该不会对查询中谓词的顺序敏感。远程数据源处的优化器可能根据输入谓词的顺序生成了另一个方案。请考虑修改 DB2 服务器的输入的谓词排序，或者与远程数据源的服务机构联系以寻求帮助。

  您还可以检查是否已替换谓词。良好的查询优化器应该不会对等效的谓词替换敏感。远程数据源处的优化器可能根据输入谓词生成了另一个方案。例如，某些优化器无法为谓词生成传递闭包语句。

– 其他函数。远程 SQL 语句是否包含原始查询中不存在的函数？可能已使用某些此类函数来转换数据类型；您务必验证它们是否必需。

数据访问方法

查询优化器在编译 SQL 或 XQuery 语句时，它将估算满足查询的不同方法的执行成本。

优化器根据这些估算值来选择最佳的访问方案。*访问方案*指定解析 SQL 或 XQuery 语句所需的操作顺序。绑定应用程序时，将创建程序包。这个*程序包*包含该应用程序中所有静态 SQL 和 XQuery 语句的访问方案。动态 SQL 和 XQuery 语句的访问方案在运行时创建。

可以通过三种方法来访问表中的数据:

- 通过按顺序扫描整个表
- 通过访问表的索引以查找特定的行
- 通过进行扫描共享

可以根据谓词中定义的条件对行进行过滤（通常在 WHERE 子句中指定这些谓词）。在所访问的表中选择的行将进行连接以生成结果集，通过对输出进行分组或排序，可以对此数据进行进一步处理。

从 DB2 9.7 开始，缺省行为是*扫描共享*，即，扫描能够使用另一个扫描的缓冲池页。扫描共享能够提高工作负载并发性和性能。通过进行扫描共享，系统能够支持更多的并发应用程序，查询可以更高效地执行，系统吞吐量得以提升，甚至连未参与扫描共享的查询也能从中受益。如果环境中的应用程序对大型表执行表扫描或多维集群（MDC）块索引扫描之类的扫描，那么扫描共享功能的效用尤其明显。编译器根据扫描类型、扫描目的、隔离级别以及对每个记录完成的工作量等因素来确定一个扫描是否适合于参与扫描共享。

通过索引扫描进行数据访问

*索引扫描*是指数据库管理器在访问基本表之前访问索引，并通过扫描该索引中指定范围内的行来缩小合格行的集合；索引扫描的意图是，对输出进行排序或者直接检索所请求的列数据（*纯索引访问*）。

在扫描索引中指定范围内的行时，*索引扫描范围*（扫描的起点和终点）由查询中与索引列进行比较的值确定。对于纯索引访问，由于所请求的所有数据都在索引中，因此不需要访问已建立索引的表。

如果创建索引时指定了 ALLOW REVERSE SCANS 选项，那么还可以按与定义的方向相反的方向来执行扫描。

如果未创建适当的索引，或者索引扫描的成本将更高，那么优化器将选择表扫描。如果表较小、索引集群比率较低、查询需要大多数表行或者使用分区索引时需要进行额外的排序（分区索引在某些情况下无法保留顺序），那么索引扫描的成本可能更高。要确定访问方案是使用表扫描还是索引扫描，请使用 DB2 说明工具。

用于限制范围的索引扫描

为了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进行求值，以了解它是否可用来满足 WHERE 子句中的等式和其他谓词。*谓词*是 WHERE 子句中搜索条件的元素，用于表达或暗指比较运算。在下列情况下，可以使用谓词来限制索引扫描范围:

- 测试 IS NULL 或 IS NOT NULL
- 测试严格和相容不等式
- 测试是否等于常量、主变量、求值为常量的表达式或者关键字
- 测试是否等于基本子查询（这是不包含 ANY、ALL 或 SOME 的子查询）；此子查询不能包含对其直接父查询块（即，此子查询所属的 SELECT）的相关列引用。

下列示例说明如何使用索引来限制扫描范围。

- 考虑具有以下定义的索引:

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

  可以使用下列谓词来限制使用索引 IX1 的扫描的范围:

```
where
  name = :hv1 and
  dept = :hv2
```

  或者

```
where
  mgr = :hv1 and
  name = :hv2 and
  dept = :hv3
```

  第二个 WHERE 子句表明，不必按键列在索引中的出现顺序来指定谓词。尽管这些示例使用主变量，但您也可以改为使用其他变量，例如参数标记、表达式或常量。

  在以下 WHERE 子句中，将只使用引用了 NAME 和 DEPT 的谓词来限制索引扫描范围:

```
where
  name = :hv1 and
  dept = :hv2 and
  salary = :hv4 and
  years = :hv5
```

  由于存在将这些列与最后两个索引键列分隔开的键列（MGR），因此不执行排序。但是，在 `name = :hv1` 和 `dept = :hv2` 谓词确定范围之后，就可以根据其余索引键列对其他谓词进行求值。

- 考虑使用 ALLOW REVERSE SCANS 选项创建的索引:

```
create index iname on tname (cname desc) allow reverse scans
```

在这种情况下，索引（INAME）基于 CNAME 列中的降序值。虽然此索引是为按降序顺序运行的扫描定义的，但也可以按升序顺序执行扫描。对此索引的使用由优化器在创建和考虑访问方案时控制。

用于测试不等式的索引扫描

某些不等式谓词可以限制索引扫描范围。共有两种类型的不等式谓词:

- 严格不等式谓词

  用于范围限制谓词的严格不等式运算符是大于（>）和小于（<）。

  在限制索引扫描范围时，将只考虑一个带有严格不等式谓词的列。在以下示例中，应用于 NAME 和 DEPT 列的谓词都可用于限制范围，但应用于 MGR 列的谓词不能用于该用途。

  ```
  where
    name = :hv1 and
    dept > :hv2 and
    dept < :hv3 and
    mgr < :hv4
  ```

- 相容不等式谓词

  用于范围限制谓词的相容不等式运算符是:

  – >= 和 <=

  – BETWEEN

  – LIKE

  在限制索引扫描范围时，可以考虑多个带有相容不等式谓词的列。在以下示例中，所有谓词都可用于限制范围。

  ```
  where
    name = :hv1 and
    dept >= :hv2 and
    dept <= :hv3 and
    mgr <= :hv4
  ```

  假定 `:hv2 = 404`、`:hv3 = 406` 并且 `:hv4 = 12345`。数据库管理器将扫描索引以查找部门 404 和 405，但当它遇到第一位其职员号（MGR 列）大于 12345 的经理时，它将停止扫描部门 406。

用于对数据进行排序的索引扫描

如果查询需要已排序的输出，并且，如果各个排序列从第一个索引键列开始在该索引中依序出现，那么可以使用该索引对数据进行排序。排序操作可以由 ORDER BY、DISTINCT、GROUP BY、“= ANY”子查询、“> ALL”子查询、“< ALL”子查询、INTERSECT 或 EXCEPT 以及 UNION 之类的操作引起。这方面的例外情况包括:

- 对于分区索引而言，仅当索引键列以表分区键列作为前缀，或者分区消除功能消除了除一个分区以外的所有分区时，它才可用于对数据进行排序。
- 在测试索引键列是否等于“常量值”或者任何求值为常量的表达式时，排序列可以与第一个索引键列不同。

请考虑以下查询:

```
where
  name = 'JONES' and
  dept = 'D93'
order by mgr
```

对于此查询，可以使用索引对行进行排序，这是因为 NAME 和 DEPT 始终是相同的值，因此将进行排序。即，前导 WHERE 和 ORDER BY 子句等同于:

```
where
  name = 'JONES' and
  dept = 'D93'
order by name, dept, mgr
```

唯一索引也可用于截断排序顺序要求。请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
```

由于 IX0 索引确保 PROJNO 唯一，因此不需要另外根据 PROJNAME 列进行排序: 对于每个 PROJNO 值，只有一个 PROJNAME 值。

索引访问类型

在某些情况下，优化器可能会发现，可以从表的索引中检索查询所需的所有数据。在其他情况下，优化器可以使用多个索引来访问表。对于范围集群表，可以通过“虚拟”索引来访问数据，“虚拟”索引将计算数据记录的位置。

纯索引访问

在某些情况下，可以从索引检索所有必需的数据，而不必访问表。这称为*纯索引访问*。例如，考虑以下索引定义:

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

仅仅通过访问该索引（而不必读取基本表）即可满足以下查询:

```
select name, dept, mgr, salary
  from employee
  where name = 'SMITH'
```

但是，所需的列经常未出现在索引中。要检索这些列中的数据，必须读取表行。要使优化器能够选择纯索引访问方法，请创建包含包括列的唯一索引。例如，考虑以下索引定义:

```
create unique index ix1 on employee
  (name asc)
  include (dept, mgr, salary, years)
```

此索引实现 NAME 列的唯一性，并且还存储并维护 DEPT、MGR、SALARY 和 YEARS 列的数据。这样，仅仅通过访问该索引即可满足以下查询:

```
select name, dept, mgr, salary
  from employee
  where name = 'SMITH'
```

您务必考虑包括列所需的附加存储空间和维护成本是否合理。如果很少执行利用包括列的查询，那么成本可能不合理。

多索引访问

优化器可以选择通过扫描同一个表的多个索引来满足 WHERE 子句的谓词。例如，考虑下面这两个索引定义:

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

通过使用这两个索引，可以满足下列谓词:

```
where
  dept = :hv1 or
  (job = :hv2 and
  years >= :hv3)
```

扫描索引 IX2 将生成满足 `dept = :hv1` 谓词的记录标识（RID）列表。扫描索引 IX3 将生成满足 `job = :hv2 and years >= :hv3` 谓词的 RID 列表。在访问该表之前，将组合这两个 RID 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，如以下示例所示:

```
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。如果应用程序对同一个表中相应的列创建多个索引，并且对该表运行包含多个 AND 谓词的查询，那么可能会发生索引 AND 运算。对每个已建立索引的列执行多索引扫描时，将生成进行散列以创建位图的值。第二个位图用于探测第一个位图，以便为最终结果集生成合格的行。例如，下列查询:

```
INDEX IX4: SALARY   ASC
INDEX IX5: COMM     ASC
```

可用于解析下列谓词:

```
where
  salary between 20000 and 30000 and
  comm between 1000 and 3000
```

在本示例中，扫描索引 IX4 将生成满足 `salary between 20000 and 30000` 谓词的位图。扫描 IX5 和探测 IX4 的位图将生成满足这两个谓词的合格 RID 的列表。这称为*动态位图 AND 运算*。只有满足下列条件才会发生这种情况：表具有足够大的基数，它的列有足够多的值在合格范围内，或者使用等式谓词时重复值足够多。

为了在扫描多个索引时实现动态位图的性能增益，可能有必要更改 **sortheap** 数据库配置参数和 **sheapthres** 数据库管理器配置参数的值。在访问方案中使用动态位图时，需要附加的排序堆空间。将 **sheapthres** 设置为比较接近 **sortheap** 时（即，低于每个并发查询所使用空间量的两到三倍），那么采用多索引访问方法的动态位图所耗用的内存量肯定显著低于优化器的预期。解决方案是，相对于 **sortheap** 增大 **sheapthres** 的值。

优化器在访问单个表时，不会对索引 AND 运算和索引 OR 运算进行组合。

范围集群表中的索引访问

与标准表不同，范围集群表不需要那些将键值映射至某一行的物理索引（类似于传统的 B 型树索引）。而是，它利用列域的顺序性质并使用有效映射在表中生成特定行的位置。在此类映射的最简单示例中，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集群属性来根据最佳集群索引（它的唯一成本就是计算范围集群函数）生成访问方案。表中的行的集群有保证，这是因为，范围集群表将保留它们的原始键值排序。

索引访问和集群比率

优化器在选择访问方案时，它估算将所需的页从磁盘读入缓冲池所需执行的 I/O 次数。此估算包括预测缓冲池的使用情况，这是因为从已包含在缓冲池中的页读取行时，无需另外执行 I/O。

对于索引扫描而言，系统目录中的信息可以帮助优化器估算将数据页读入缓冲池的 I/O 成本。它将使用 SYSCAT.INDEXES 视图的以下各列中的信息:

- CLUSTERRATIO 信息指示表数据相对于此索引的集群度。此数值越大，各行按索引键顺序排列的情况越好。如果表行顺序接近于索引键顺序，那么当数据页包含在缓冲区中时，可以从该页读取行。如果此列的值为 -1，并且可以获得 PAGE_FETCH_PAIRS 和 CLUSTERFACTOR 信息，那么优化器将使用该信息。
- PAGE_FETCH_PAIRS 列包含多对数值以及 CLUSTERFACTOR 信息，这些数值模拟将数据页读入各种大小的缓冲池时所需执行的 I/O 次数。只有在指定 DETAILED 子句的情况下对索引调用 RUNSTATS 命令时，才会为这些列收集数据。

如果没有可用的索引集群统计信息，那么优化器将使用缺省值，即假定数据相对于索引而言集群情况不佳。数据的集群度将显著影响性能，您应该尝试使其中一个对表定义的索引的集群度接近 100%。通常，只有一个索引可达到 100% 集群度，但下列情况除外: 索引的键是集群索引键的超集，或者两个索引的键列之间存在实际的关联。

重组表时，您可以指定一个索引，此索引将用于对行进行集群并在插入处理期间保持这些行处于集群状态。由于更新和插入操作会降低表相对于索引而言的集群度，因此您可能需要定期重组表。为了减少频繁执行插入、更新或删除操作的表的重组次数，请在 ALTER TABLE 语句中指定 PCTFREE 子句。

扫描共享

*扫描共享*是指一个扫描利用另一扫描所完成的工作的能力。共享工作的示例包括磁盘页读、磁盘查找、缓冲池内容复用以及解压等等。

工作量繁重的扫描，例如对大型表进行的表扫描或多维集群（MDC）块索引扫描，有时适合与其他扫描共享页读。这样的共享扫描可以从表中的任意一点开始，以便利用已包含在缓冲池中的页。当共享扫描到达表的末尾时，它将从开头继续并在到达开始点时完成操作。这称为*回绕扫描*。第 187 页的图 25 显示了表和索引的常规扫描与回绕扫描之间的区别。

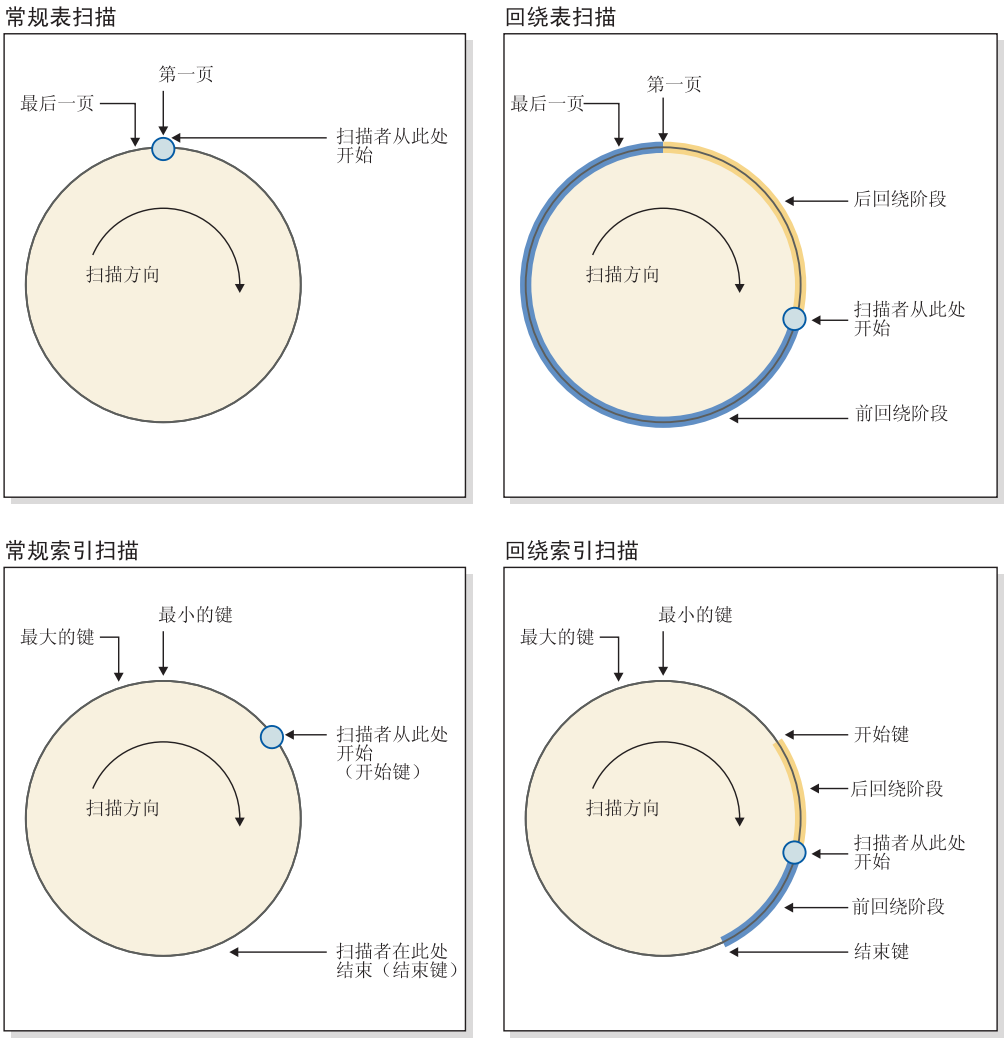


图 25. 常规扫描和回绕扫描的概念性视图

缺省情况下，扫描共享功能处于启用状态，是否适合于扫描共享和回绕由 SQL 编译器自动确定。在运行时，合格的扫描可能参与也可能不参与共享或回绕，这取决于编译时未知的因素。

共享的扫描者在*共享组*中进行管理。这些组尽可能将它们的成员置于一起，以便最大程度地增加共享的好处。如果一个扫描比另一个扫描快，那么页共享的好处可能会消失。在这种情况下，缓冲池中由第一个扫描访问的缓冲池页在共享组中的另一个扫描能够访问它们之前可能会被清除。数据服务器按同一个共享组中两个扫描之间的缓冲池页数测量它们之间的距离。数据服务器还监视扫描速度。如果同一个共享组中两个扫描之间的距离增大到太大，那么它们可能无法共享缓冲池页。为了减少这种情况，可以对较快的扫描进行调速，以允许较慢的扫描在数据页被清除前访问那些页。第 188 页的图 26 显示了两个共享集，其中一个用于表，另一个用于块索引。*共享集*是通过同一种访问机制（例如表扫描或块索引扫描）访问同一个对象（例如一个表）的共享组的集合。对于表扫描而言，页读顺序按页标识递增；对于块索引扫描而言，页读顺序按键值递增。

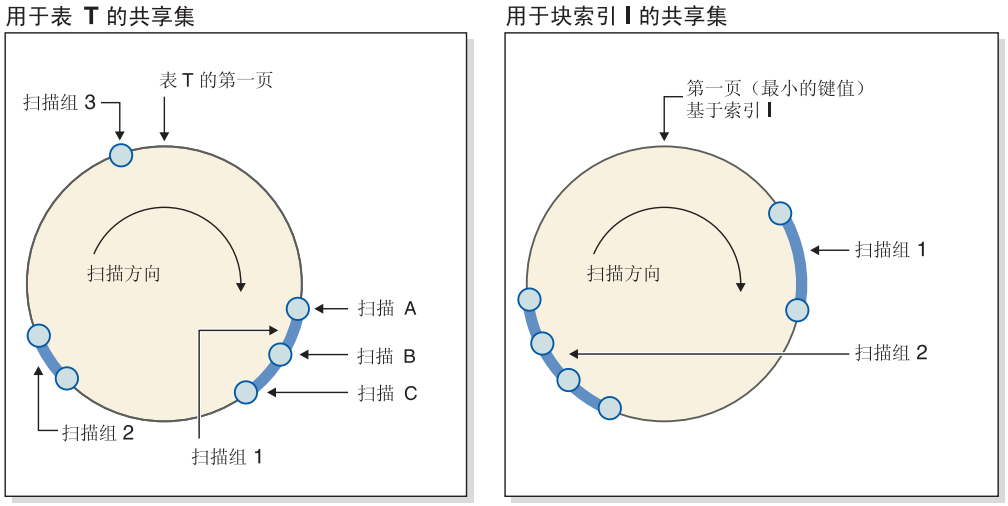


图 26. 表扫描和块索引扫描共享的共享集

此图还显示了如何在组中复用缓冲池内容。请考虑扫描 C，这是组 1 的第一个扫描。后续扫描（A 和 B）与 C 分组到一起，这是因为它们与 C 接近并有可能复用 C 读入缓冲池的页。

高优先级扫描者从来不会被低优先级扫描者调速，而是移至另一个共享组。高优先级扫描者可能会被放入某个组并由于该组中低优先级扫描者所完成的工作而受益。只要受益持续，它将一直停留在该组中。通过对快速扫描者进行调速或者将其移至更快的共享组（如果该扫描者遇到这样的组），那么数据服务器将调整共享组以确保共享保持最优。

您可以使用 db2pd 命令来查看关于扫描共享的信息。例如，对于单个的共享扫描，db2pd 输出将显示扫描速度和扫描调节时间量之类的数据。对于共享组，命令输出将显示组中的扫描数以及该组共享的页数。

在 EXPLAIN_ARGUMENT 表中，有一些新行包含关于表扫描和索引扫描的扫描共享信息（您可以使用 db2exfmt 命令来格式化和查看此表的内容）。

您可以使用优化器概要文件来覆盖编译器所作的关于扫描共享的决策（请参阅“访问类型”）。这样的覆盖仅用于存在特殊需求时的情况；例如，如果要求结果集中记录的顺序可重复，但必须避免使用 ORDER BY 子句（这可能将触发排序），那么回绕提示很有用。否则，建议您不要使用这些优化概要文件，除非 DB2 服务机构要求您进行使用。

连接

*连接*是指根据信息的某些公共领域对来自两个或更多个表的数据进行组合的过程。如果连接条件（*连接谓词*）确定对应的行中的信息匹配，那么一个表中的行就会与另一个表中的行配对。

例如，考虑下面这两个表:

TABLE1

PROJ	PROJ_ID
A	1
B	2

TABLE2

PROJ_ID	NAME
1	Sam
3	Joe

TABLE1

PROJ	PROJ_ID
C	3
D	4

TABLE2

PROJ_ID	NAME
4	Mary
1	Sue
2	Mike

要将 TABLE1 与 TABLE2 连接，以使 PROJ_ID 列包含相同的值，请使用以下 SQL 语句:

```
select proj, x.proj_id, name
  from table1 x, table2 y
  where x.proj_id = y.proj_id
```

在这种情况下，适当的连接谓词是: `where x.proj_id = y.proj_id`。

此查询将生成以下结果集:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

根据任何连接谓词的性质以及通过表和索引统计信息确定的任何成本，优化器将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

连接两个表时，将选择其中一个表作为外表，而将另一个表视为该连接的内表。首先访问外表，并且只对其执行一次扫描。是否对内表执行多次扫描取决于该连接的类型以及可用的索引。即使一个查询连接两个以上的表，优化器每次也只连接两个表。必要时，将创建临时表来保存中间结果。

您可以提供显式的连接运算符（例如 INNER 或 LEFT OUTER JOIN），以确定如何在连接中使用表。但是，以这种方式更改查询之前，应允许优化器确定如何连接表，然后分析查询性能以确定是否添加连接运算符。

连接方法

当查询要求连接表时，优化器可以选择三种基本连接策略中的一种策略：嵌套循环连接、合并连接或散列连接。

嵌套循环连接

嵌套循环连接通过下列两种方式中的一种方式执行:

- 对于外表中每个被访问的行，扫描内表

例如，表 T1 中的列 A 和表 T2 中的列 A 包含下列值:

外表 T1: 列 A	内表 T2: 列 A
2	3
3	2
3	2
	3
	1

要在表 T1 与 T2 之间完成嵌套循环连接，数据库管理器将执行下列步骤:

1. 读取 T1 中的第一行。A 的值是 2。
2. 扫描 T2，直到找到匹配项（2）为止，然后连接这两行。
3. 重复步骤 2，直到到达表的末尾为止。
4. 返回至 T1 并读取下一行（3）。
5. 从第一行开始扫描 T2，直到找到匹配项（3）为止，然后连接这两行。
6. 重复步骤 5，直到到达表的末尾为止。
7. 返回至 T1 并读取下一行（3）。
8. 像前面那样扫描 T2，连接所有匹配的行（3）。

- 对于外表中每个被访问的行，对内表执行索引查找

  如果存在以下格式的谓词，那么可以使用此方法:

  ```
  expr(outer_table.column) relop inner_table.column
  ```

  其中，relop 是比较运算符（例如 =、>、>=、< 或 <=），而 expr 是基于外表的有效表达式。例如:

  ```
  outer.c1 + outer.c2 <= inner.c1
  outer.c4 < inner.c3
  ```

  此方法可以显著减少每次访问外表时要在内表中访问的行数；受益程度取决于许多因素，其中包括连接谓词的选择性。

优化器对嵌套循环连接进行求值时，在执行连接前还将确定是否对外表进行排序。如果它根据连接列对外表进行排序，那么可以减少为了访问磁盘上的内表页而对内表执行读操作的次数，这是因为这些页很可能已在缓冲池中。如果该连接使用高度集群的索引来访问内表，并且外表已进行排序，那么所访问的索引页的数目可减至最少。

如果优化器预期该连接将执行成本更高的后期排序，那么它还可能选择在连接前执行排序。为了支持 GROUP BY、DISTINCT、ORDER BY 或合并连接操作，可能有必要执行后期排序。

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*）需要格式为 table1.column = table2.column 的谓词。这称为*等式连接谓词*。合并连接需要已排序的连接列输入，此排序可通过访问索引或执行排序实现。如果连接列是 LONG 字段列或大对象（LOB）列，那么无法使用合并连接。

在合并连接中，将同时扫描所连接的表。合并连接的外表只被扫描一次。除非外表中出现重复的值，否则内表也只被扫描一次。如果出现重复的值，那么可能再次扫描内表中的一组行。

例如，表 T1 中的列 A 和表 T2 中的列 A 包含下列值:

外表 T1: 列 A	内表 T2: 列 A
2	1
3	2
3	2
	3
	3

要在表 T1 与 T2 之间完成合并连接，数据库管理器将执行下列步骤:

1. 读取 T1 中的第一行。A 的值是 2。
2. 扫描 T2，直到找到匹配项（2）为止，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 的末尾时，返回到 T1 以获取下一行。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。数据库管理器将记住此位置。

散列连接

散列连接需要一个或多个格式为 `table1.columnX = table2.columnY` 的谓词，在此格式中，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。对于类型为 DECFLOAT 的列，精度必须相同。该列不能是 LONG 字段列或 LOB 列。

首先，扫描指定的内表并且将行复制到从排序堆中划出的内存缓冲区，该排序堆由 **sortheap** 数据库配置参数指定。根据使用连接谓词的列计算而得的散列值，将内存缓冲区分为若干部分。如果内表的大小超出可用的排序堆空间，那么将所选部分的缓冲区写入临时表。

处理内表完毕后，通过首先比较对连接谓词的列计算的散列值，扫描第二个表（外表）并将该表的行与内表的行匹配。如果外表行列的散列值与内表行列的散列值匹配，那么将比较实际的连接谓词列值。

与未被写入临时表的表部分相对应的外表行将立即与内存中的内表行匹配。如果内表的对应部分已被写入临时表，那么也将外表行写入临时表。最后，从临时表中读取匹配的表部分对，比较它们的行的散列值，然后检查连接谓词。

为了全面实现散列连接的性能增益，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。

如果能够避免散列循环以及溢出到磁盘，那么散列连接的性能最佳。要调整散列连接性能，请估算可供 **sheapthres** 使用的最大内存量，然后调整 **sortheap** 参数。增大它的设置，尽可能避免散列循环和磁盘溢出，但不要达到 **sheapthres** 参数指定的限制。

增大 **sortheap** 值应该还能提高需要执行多次排序的查询的性能。

有关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。由查询的优化类确定的这些方法包括多种搜索策略、星型模式连接、早期外连接和组合表。

联合枚举算法是优化器所探查的方案组合数的重要决定因子。

- 贪婪联合枚举
  - 在空间和时间需求方面效率较高
  - 使用单向枚举；即，一旦为两个表选择连接方法，在进一步优化期间就不进行更改
  - 连接多个表时，可能会错过最佳访问方案。如果您的查询只连接两个或三个表，那么贪婪联合枚举选择的访问方案与动态规划联合枚举选择的访问方案相同。如果该查询包含多个应用于同一列的连接谓词（以显式方式指定或者以隐式方式通过谓词传递闭包生成），那么更是如此。
- 动态规划联合枚举
  - 在空间和时间需求方面效率不高，随着所连接的表的数目增加，空间和时间需求按指数规律增加
  - 搜索最佳访问方案时，高效而详尽
  - 类似于 DB2 z/OS 版所使用的策略

星型模式连接

在查询中引用的表几乎始终通过连接谓词相互相关。如果在不使用连接谓词的情况下连接两个表，那么将形成这两个表的笛卡尔乘积。在笛卡尔乘积中，第一个表的每个合格行都与第二个表的每个合格行连接。这将创建一个通常非常大的结果表，其大小是两个源表大小的交叉乘积。由于这种方案的性能不可能很好，因此优化器甚至会避免确定此类访问方案的成本。

在优化类设置为 9 时或者在星型模式的特殊情况下，将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，而其他表称为*维表*。维表只有一个连接，该连接将这些维表与事实表连接，而与查询无关。每个维表都包含附加的值，这些值展开有关事实表中特定列的信息。典型的查询由多个引用维表中的值的局部谓词组成，并包含将维表与事实表连接的连接谓词。对于这些查询而言，在访问大型的事实表之前，对多个小型维表计算笛卡尔乘积可能有好处。当多个连接谓词与一个多列索引匹配时，这种技术很有用。

DB2 数据服务器能够识别对采用星型模式设计并且至少带有两个维表的数据库执行的查询，并且能够增大搜索空间以包括那些计算维表的笛卡尔乘积的可能方案。如果计算笛卡尔乘积的方案的估算成本最低，那么优化器将选择该方案。

这种星型模式连接策略假定连接使用主键索引。另一个方案涉及外键索引。如果事实表中的外键列是单列索引，并且在所有维表中都具有相对较高的选择性，那么可以使用以下星型模式连接技术:

1. 通过下列操作处理每个维表:
   - 在基于维表和事实表的外键索引之间执行半连接
   - 对记录标识（RID）值进行散列，以便动态地创建位图
2. 对于每个位图，对上一个位图使用 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 执行排序。
5. 访存一个基本表行。
6. 将事实表与其每个维表重新连接，从而访问 SELECT 子句在维表中所需的列。
7. 重新应用残留谓词。

这种技术不需要多列索引。不要求事实表与维表之间存在显式的引用完整性约束，但建议您设置这些约束。

星型模式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 **sortheap** 数据库配置参数指定。

早期外连接

当优化器发现一个表中的每一行最多只需要与另一个表中的一行进行连接时，它可以选择早期外连接。

当存在应用于其中一个表的键列的连接谓词时，就可以进行早期外连接。例如，请考虑以下查询，此查询返回职员及其直接主管的姓名。

```
select employee.name as employee_name,
    manager.name as manager_name
  from employee as employee, employee as manager
  where employee.manager_id = manager.id
```

假定 ID 列是 EMPLOYEE 表中的一个键，并且每个职员最多只有一个主管，那么此连接将避免在 MANAGER 表中搜索后续的匹配行。

当查询包含 DISTINCT 子句时，也可以进行早期外连接。例如，请考虑以下查询，此查询返回车型售价超过 30000 美元的汽车制造商的名称。

```
select distinct make.name
  from make, model
  where
    make.make_id = model.make_id and
    model.price > 30000
```

对于每家汽车制造商，我们只需要确定是否其生产的任何一个型号的售价超过 30000 美元。不需要将汽车制造商与其生产的所有售价超过 30000 美元的型号都连接起来，因为这样做对提高查询结果的准确性并无帮助。

对 GROUP BY 子句添加 MIN 或 MAX 聚集函数时，也可以进行早期外连接。例如，请考虑以下查询，此查询返回股票代码以及在 2000 年之前发生以下情况的最近日期:特定股票的收盘价格至少比开盘价格高 10%。

```
select dailystockdata.symbol, max(dailystockdata.date) as date
  from sp500, dailystockdata
  where
```

```
    sp500.symbol = dailystockdata.symbol and
    dailystockdata.date < '01/01/2000' and
    dailystockdata.close / dailystockdata.open >= 1.1
group by dailystockdata.symbol
```

*合格集*是 DAILYSTOCKDATA 表中满足日期和价格要求的行集，它与 SP500 表中的特定股票代码进行连接。对于 SP500 表中的每个股票代码行，如果 DAILYSTOCKDATA 表中的合格集按 DATE 的降序排列，那么只需要对每个股票代码返回合格集中的第一行，这是因为第一行代表特定股票的最近日期。合格集中的其他行不是必需的。

组合表

如果一对表的连接结果是一个新表（称为*组合表*），那么此表通常成为与另一个内表的连接的外表。这称为*组合外连接*。在某些情况下，特别是在使用贪婪联合枚举技术时，使两个表的连接结果成为后续连接的内表将很有用。当一个连接的内表由连接两个或更多个表的结果组成时，此方案称为*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 与 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表，选择第二个连接结果作为内表。在最终方案（T1xT2）x（T3xT4）中，连接结果（T3xT4）称为*组合内连接*。根据查询优化类，优化器对可以作为连接内表的表的最大数目设置不同的约束。优化类 5、7 和 9 允许进行组合内连接。

分区数据库环境中的复制型具体化查询表

复制型具体化查询表（MQT）允许数据库管理预先计算的表数据值，从而提高分区数据库环境中频繁执行的连接的性能。

注意，在此上下文中，复制型 MQT 与数据库内复制相关。数据库间复制与预订、控制表以及不同数据库和不同操作系统中的数据相关。

在以下示例中:

- SALES 表在多分区表空间 REGIONTABLESPACE 中，并且根据 REGION 列进行分割。
- 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型 MQT。

```
create table r_employee as (
  select empno, firstnme, midinit, lastname, workdept
    from employee
  )
  data initially deferred refresh immediate
  in regiontablespace
  replicated
```

更新复制型 MQT 的内容:

```
refresh table r_employee
```

使用 REFRESH 语句之后，应该对复制的表调用 RUNSTATS 实用程序，就像对任何其他表执行此调用一样。

以下查询计算职员的销售额、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
```

数据库管理器并非使用仅驻留在一个数据库分区中的 EMPLOYEE 表，而是使用 R_EMPLOYEE，这是在每个存储 SALES 表的数据库分区中均进行复制的 MQT。这将提高性能，其原因在于，执行连接时不必通过网络将职员信息传送到每个数据库分区。

并置连接中的复制型具体化查询表

复制型 MQT 也有助于并置连接。例如，如果星型模式包含分布于 20 个数据库分区中的大型事实表，那么对事实表和维表进行并置后，事实表与维表之间的连接效率最高。如果所有表都在同一个数据库分区组中，那么对于一个并置连接，最多能够对一个维表进行正确分区。其他维表不能在并置连接中使用，这是因为事实表中的连接列与事实表的分布键不对应。

假定存在根据 C1 进行分割的表 FACT (C1, C2, C3, ...)、根据 C1 进行分割的表 DIM1 (C1, dim1a, dim1b, ...) 以及根据 C2 进行分割的表 DIM2 (C2, dim2a, dim2b, ...) 等等。在这种情况下，FACT 与 DIM1 之间的连接最好，这是因为将并置谓词 `dim1.c1 = fact.c1`。这两个表都根据列 C1 进行分割。

但是，由于 FACT 根据列 C1 进行分割，而不是根据列 C2 进行分割，因此不能对涉及 DIM2 和谓词 `dim2.c2 = fact.c2` 的连接进行并置。在这种情况下，可以在事实表的数据库分区组中复制 DIM2，以便在每个数据库分区中以局部方式进行连接。

在创建复制型 MQT 时，源表可以是数据库分区组中的单一分区表或多分区表。在大多数情况下，复制的表不大，并且可以放入单一分区数据库分区组。通过只指定表中的部分列或者使用谓词来限制合格行的数目，可以限制所要复制的数据。

也可以在多分区数据库分区组中创建复制型 MQT，以便在所有数据库分区中创建源表的副本。与采用广播方式将源表传送到所有数据库分区相比，在此环境中，大型事实表与维表之间的连接更有可能以局部方式进行。

不会自动创建基于所复制的表的索引。您可以创建与基于源表的索引不同的索引。但是，为了防止未出现在源表中的约束违例，不能对复制的表创建唯一索引或定义约束，即使源表中存在相同的约束亦如此。

可以在查询中直接引用复制的表，但不可以通过将 DBPARTITIONNUM 标量函数用于复制的表以便查看特定数据库分区中的表数据。

请使用 DB2 说明工具来确定查询的访问方案是否已使用复制型 MQT。优化器选择的访问方案是否使用复制型 MQT 取决于要连接的数据。如果优化器确定以广播方式向数据库分区组中的其他数据库分区传送原始源表的成本更低，那么可能不会使用复制型 MQT。

分区数据库的连接策略

分区数据库环境的连接策略可以与未分区数据库环境的策略不同。您可以将其他技术应用于标准连接方法以提高性能。

对于频繁连接的表，应该考虑进行表并置。在分区数据库环境中，*表并置*是指两个表将相同数目的兼容分区键存储到同一个数据库分区组时出现的状态。发生这种情况时，可以在存储数据的数据库分区中执行连接处理，并且只需要将结果集移至协调程序数据库分区。

表队列

分区数据库环境中连接技术的描述使用下列术语:

- *表队列*（有时称为 TQ）是一种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机制。
- *定向型表队列*（有时称为 DTQ）对行进行散列，以便将其发送到其中一个接收数据库分区。
- *广播表队列*（有时称为 BTQ）将行发送到所有接收数据库分区，而不进行散列。

表队列用于传递表数据:

- 使用分区间并行性时，将数据从一个数据库分区传递到另一个数据库分区
- 使用分区内并行性时，在数据库分区中传递数据
- 使用单一分区数据库时，在数据库分区中传递数据

每个表队列都按单一方向传递数据。编译器决定何处需要使用表队列并将其包括在方案中。执行该方案时，数据库分区之间的连接将启动表队列。表队列在进程结束时关闭。

表队列分为多种类型:

- 异步表队列

  这些表队列被称为异步队列的原因是，它们在应用程序发出任何访存请求之前读取行。发出 FETCH 语句时，将从表队列中检索该行。

  如果您在 SELECT 语句中指定了 FOR FETCH ONLY 子句，那么将使用异步表队列。如果您只访存行，那么异步表队列的速度较快。
- 同步表队列

  这些表队列被称为同步队列的原因是，对于应用程序发出的每个 FETCH 语句，这些队列读取一行。在每个数据库分区中，游标都定位在要从该数据库分区读取的下一行上。

  如果您未在 SELECT 语句中指定 FOR FETCH ONLY 子句，那么将使用同步表队列。在分区数据库环境中，如果您要更新行，那么数据库管理器将使用同步表队列。
- 合并表队列

  这些表队列维护顺序。
- 非合并表队列

这些表队列也被称为*常规表队列*，它们不维护顺序。

- 侦听器表队列（有时称为 LTQ）

  这些表队列与相关的子查询配合使用。使用这种类型的表队列时，将关联值向下传递至子查询，然后将结果向上传递回给父查询块。

用于分区数据库的连接方法

有多种连接方法可用于分区数据库环境，其中包括：并置连接、广播外表连接、定向外表连接、定向内表和外表连接、广播内表连接以及定向内表连接。

在下面的图中，q1、q2、和 q3 表示表队列。所引用的表包含在两个数据库分区中，箭头指示表队列的发送方向。协调程序数据库分区是数据库分区 0。

并置连接

并置连接以本地方式在数据所在的数据库分区中发生。该数据库分区在完成连接以后将数据发送到其他数据库分区。要使优化器考虑并置连接，必须对所连接的表进行并置，并且所有各对相应分布键都必须参与等式连接谓词。图 27 提供了一个示例。

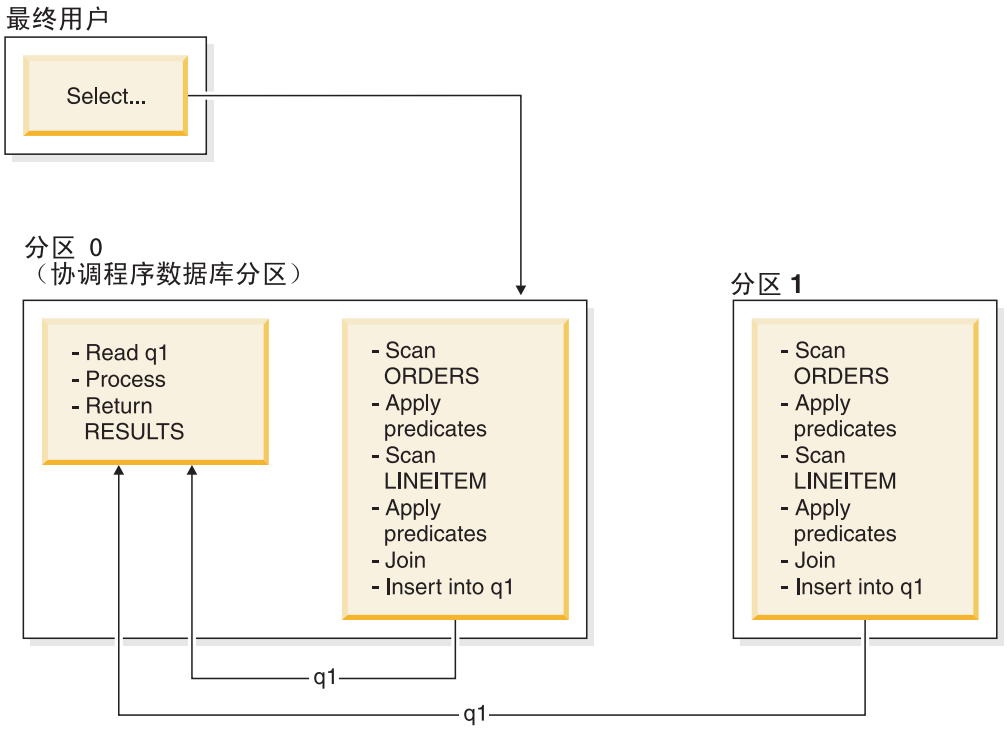


图 27. 并置连接示例

LINEITEM 和 ORDERS 表都根据 ORDERKEY 列进行分区。此连接以本地方式在每个数据库分区中执行。在此示例中，假定连接谓词为：`orders.orderkey = lineitem.orderkey`。

复制型具体化查询表（MQT）能够提高并置连接的可能性。

广播外表连接

广播外表连接代表一种并行连接策略，如果所连接的表之间没有等式连接谓词，那么可以使用此连接。此连接也可用于其他证实此连接方法最合乎成本效益的情况。例如，当有一个很大的表和一个很小的表，并且未根据连接谓词列对任何一个表进行分割时，可能会发生广播外表连接。低成本方法是将较小的表广播至较大的表，而不是分割这两个表。图 28 提供了一个示例。

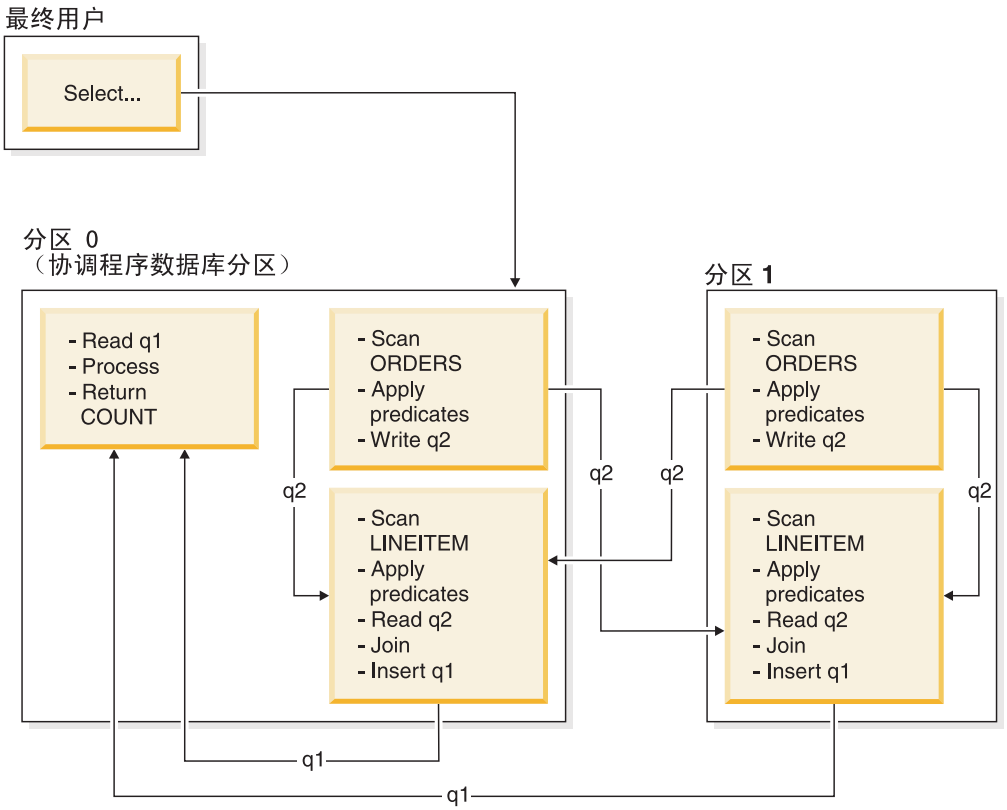
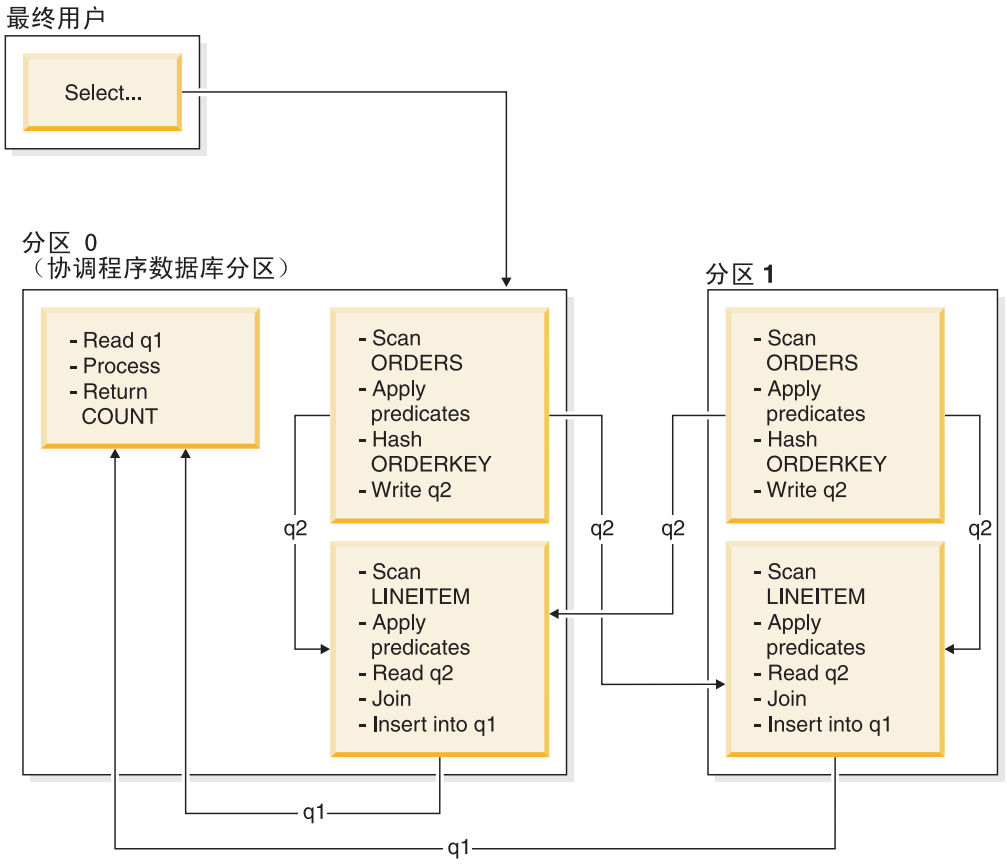


图 28. 广播外表连接示例

ORDERS 表被发送到所有包含 LINEITEM 表的数据库分区。表队列 q2 被广播至内表的所有数据库分区。

定向外表连接

在定向外表连接策略中，根据内表的分割属性将外表的每一行发送至内表的一个部分。此连接在此数据库分区中进行。第 199 页的图 29 提供了一个示例。



LINEITEM 表根据 ORDERKEY 列进行分区。ORDERS 表根据另一个列进行分区。ORDERS 表将进行散列并被发送到 LINEITEM 表的正确数据库分区。在此示例中，假定连接谓词为: `orders.orderkey = lineitem.orderkey`。

图 29. 定向外表连接示例

定向内表和外表连接

在定向内表和外表连接策略中，根据连接列的值，将外表和内表的行定向到一组数据库分区。此连接在这些数据库分区中进行。第 200 页的图 30 提供了一个示例。

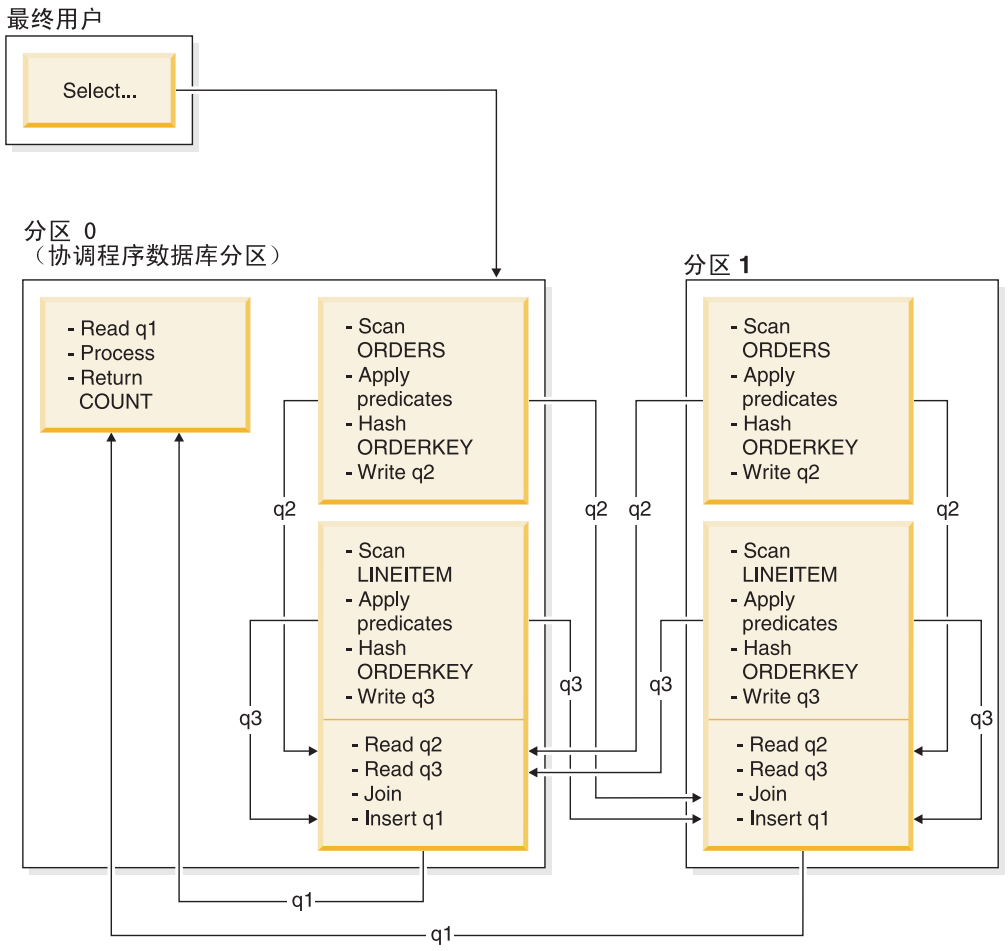
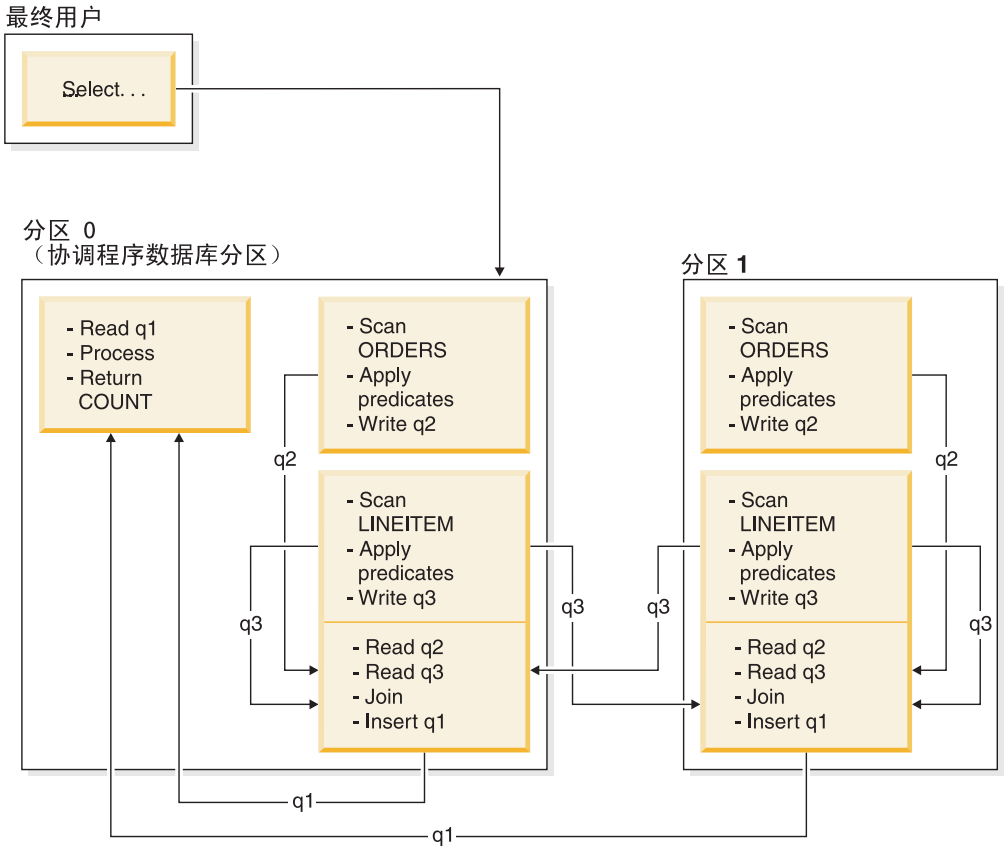


图 30. 定向内表和外表连接示例

两个表均未根据 ORDERKEY 列进行分区。这两个表都将进行散列并被发送到新的数据库分区，它们将在那些数据库分区中进行连接。表队列 q2 和 q3 都将被定向。在此示例中，假定连接谓词为：`orders.orderkey = lineitem.orderkey`。

广播内表连接

在广播内表连接策略中，将内表广播至外表的所有数据库分区。第 201 页的图 31 提供了一个示例。

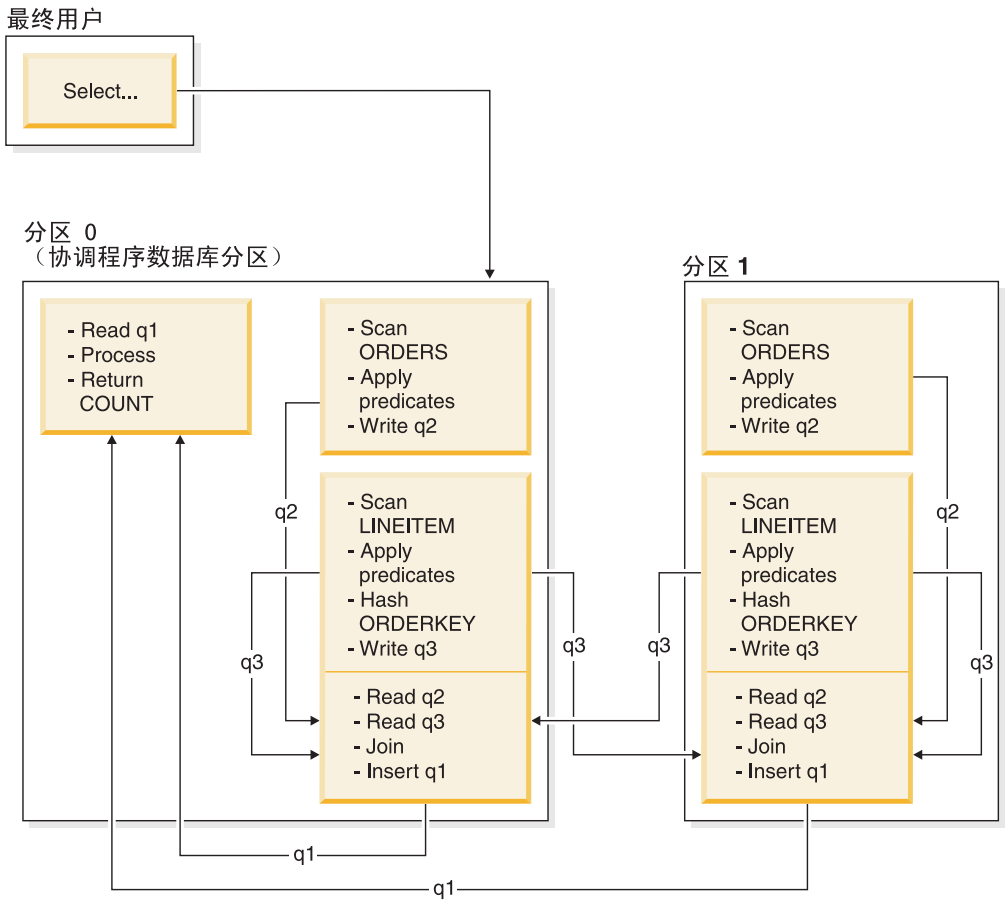


LINEITEM 表被发送到所有包含 ORDERS 表的数据库分区。表队列 q3 被广播至外表的所有数据库分区。

图 31. 广播内表连接示例

定向内表连接

在定向内表连接策略中，根据外表的分割属性，将内表的每一行发送至外表的一个数据库分区。此连接在此数据库分区中进行。第 202 页的图 32 提供了一个示例。



ORDERS 表根据 ORDERKEY 列进行分区。LINEITEM 表根据另一个列进行分区。LINEITEM 表将进行散列并被发送到 ORDERS 表的正确数据库分区。在此示例中，假定连接谓词为: `orders.orderkey = lineitem.orderkey`。

图 32. 定向内表连接示例

排序和分组对查询优化的影响

当优化器选择访问方案时，它将考虑数据排序操作对性能的影响。如果没有任何索引按请求的顺序对访存的行进行排序，那么将执行排序。如果优化器确定排序成本低于索引扫描成本，那么也会执行排序。

优化器采用下列其中一种方式来处理已排序的数据:

- 在查询执行时，通过管道传送排序结果
- 让数据库管理器以内部方式处理排序

管道传送排序与非管道传送排序

如果可以通过单一顺序遍历来读取数据的最终排序列表，那么可以对这些结果进行*管道传送*。管道传送是比非管道传送更快的排序结果传送方式。优化器将尽可能选择对排序结果进行管道传送。

无论是否对排序进行管道传送，排序时间都取决于多种因素，其中包括要排序的行数、键大小和行宽。如果要排序的行占用的空间超出排序堆中的可用空间量，那么将

执行多遍排序。每一遍都对整个行集的某个子集进行排序。每一遍的结果都存储在缓冲池中的一个临时表中。如果缓冲池中没有足够的空间，那么可以将此临时表中的页写入磁盘。所有各遍排序完成后，将这些已排序的子集合并成单一已排序的行集。如果对排序进行管道传送，那么合并那些行时，将直接把它们传递给关系数据服务（RDS）。（RDS 是一个 DB2 组件，负责处理对数据库内容的访问或操作请求。）

下推分组和排序运算符

在某些情况下，优化器可以选择将排序或聚集操作从 RDS 下推到数据管理服务（DMS）。（DMS 是一个 DB2 组件，用于控制数据库中表和表数据的创建、除去、维护以及访问。）下推这些操作使 DMS 能够直接将数据传递至排序或聚集例程，从而提高性能。如果不进行下推，那么 DMS 首先将此数据传递给 RDS，后者再与排序或聚集例程进行交互。例如，此类优化可以使以下查询受益:

```
select workdept, avg(salary) as avg_dept_salary
  from employee
  group by workdept
```

排序中的分组操作

如果排序操作能够生成 GROUP BY 操作所需的顺序，那么优化器可以在进行排序时执行部分或全部 GROUP BY 聚集。如果每个组中的行数较大，那么这种做法就有利。如果在排序期间执行一些分组以减少或消除排序溢出到磁盘的情况，就会更加有利。

在排序期间进行聚集要求执行下列三个聚集阶段中的一个或多个阶段，以确保返回正确的结果。

- 聚集过程的第一阶段（*部分聚集*）计算聚集值，直到填满排序堆为止。在部分聚集期间，接受未聚集的数据并生成部分聚集。如果填满排序堆，那么其余数据将溢出到磁盘，其中包括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。
- 聚集过程的第二阶段（*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进行聚集。由于分组键列是分布键列的子集，因此无法完成聚集。中间聚集使用现有的部分聚集来生成新的部分聚集。并不总是会执行此阶段。此阶段既用于分区内并行性也用于分区间并行性。在分区内并行性的情况下，如果全局分组键可用，那么将完成分组。在分区间并行性的情况下，如果分组键是用来在数据库分区间进行分组的分布键的子集，从而要求执行重新分布才能完成聚集，那么将执行此阶段。对于分区内并行性，如果每个代理程序在缩减为单个代理程序以完成聚集之前都对其溢出的排序运行结果完成合并，那么将出现类似的情况。
- 聚集过程的最后阶段（*最终聚集*）使用所有部分聚集并生成最终聚集。此步骤始终通过 GROUP BY 运算符执行。排序操作无法执行完整的聚集，这是因为无法保证排序不会被分割。完整的聚集接受未聚集的数据，然后生成最终聚集。这种聚集方法通常用于对已处于正确顺序的数据进行分组。

优化策略

分区内并行性的优化策略

如果编译 SQL 语句时指定了并行度，那么优化器可以选择访问方案，以便在单一数据库分区中以并行方式执行查询。

在运行时，将创建多个称为“子代理程序”的数据库代理程序来执行该查询。子代理程序的数目将小于或等于编译该 SQL 语句时指定的并行度。

为了将访问方案并行化，优化器将它划分为两个部分，每个子代理程序运行一部分，协调代理程序运行另一部分。子代理程序通过表队列将数据传递至协调代理程序或其他子代理程序。在分区数据库环境中，子代理程序与其他数据库分区中的子代理程序之间能够通过表队列来发送或接收数据。

分区内并行扫描策略

可以采用并行方式对同一个表或索引执行关系扫描和索引扫描。要进行并行关系扫描，需将表划分为由页或行组成的范围，然后将范围分配给子代理程序。子代理程序将扫描分配给它的范围，处理当前范围完毕后，它将被分配另一个范围。

要进行并行索引扫描，需根据索引键值以及键值的索引条目数将索引划分为多个记录范围。并行索引扫描的执行方式类似于并行表扫描，即，将某个范围内的记录分配给子代理程序。子代理程序处理当前范围完毕后，将被分配新的范围。

优化器确定扫描单位（页或行）和扫描粒度。

并行扫描在各个子代理程序之间均匀地分布工作。并行扫描的目标是，平衡所有子代理程序的负载并使它们保持相同的繁忙程度。如果繁忙子代理程序数等于可用处理器数，且磁盘未过度处理 I/O 请求，那么表明机器资源得到高效利用。

执行查询时，其他访问方案策略可能会导致数据不平衡。优化器选择并行策略，以便在子代理程序之间维持数据平衡。

分区内并行排序策略

优化器可以选择下列其中一种并行排序策略:

- 循环排序

  这也称为*再分布排序*。这种方法使用共享内存，高效地将数据尽可能均匀地重新分布到所有子代理程序。它使用轮询算法来确保分布均匀。首先，为每个子代理程序创建单个排序。在插入阶段，子代理程序以循环方式对每个排序执行插入，以使数据分布更加均匀。
- 分区排序

  这类似于循环排序，即，为每个子代理程序创建一个排序。子代理程序将一个散列函数应用于排序列，以便确定应该将行插入到哪个排序。例如，如果合并连接的内表和外表是分区排序，那么子代理程序可使用合并连接来连接相应的表部分并以并行方式执行。
- 复制排序

  如果每个子代理程序都需要所有排序输出，那么使用这种排序。将创建一个排序，并且在将行插入到排序时使各个子代理程序同步。排序完成后，每个子代理程序都读取整个排序。如果行数较小，那么可使用此排序对数据流进行重新平衡。
- 共享排序

  此排序与复制排序相同，只是子代理程序要对已排序的结果打开一个并行扫描，以便采用一种与循环排序相似的方式在子代理程序之间分布数据。

分区内并行临时表

子代理程序可以协同工作，通过将行插入到同一个表来生成临时表。这称为*共享临时表*。根据是要复制还是要分割数据流，子代理程序可以对共享临时表进行专用扫描或并行扫描。

分区内并行聚集策略

子代理程序可以采用并行方式来执行聚集操作。聚集操作要求根据分组列对数据进行排序。如果可以保证一个子代理程序接收一组分组列值的所有行，那么该程序可以执行完整的聚集。发生这种情况的条件是，先前已执行分区排序，致使已根据分组列对数据流进行分割。

否则，子代理程序可以执行部分聚集，并使用另一种策略来完成该聚集。某些策略如下所示:

- 通过合并表队列将部分聚集的数据发送至协调代理程序。协调代理程序完成聚集。
- 将部分聚集的数据插入到分区排序。该排序根据分组列进行分割，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，那么可将部分聚集的数据插入到复制排序。每个子代理程序都使用该复制排序来完成聚集，并接收完全相同的聚集结果副本。

分区内并行连接策略

子代理程序可以采用并行方式来执行连接操作。并行连接策略由数据流的特征确定。

通过对连接的内表和/或外表进行分区或者复制数据流，可以将连接并行化。例如，如果因为并行扫描已将嵌套循环连接的外流分区，而且内流由每个子代理程序独立重新求值，那么可以将该连接并行化。如果合并连接的内流和外流由于分区排序而按值分区，那么可将该连接并行化。

MDC 表的优化策略

如果创建多维集群（MDC）表，那么可以提高许多查询的性能，这是因为优化器可以应用附加的优化策略。这些策略主要依赖于块索引效率有所提高，但根据多个维进行集群这一优点还能提高数据检索速度。

MDC 表优化策略还可以利用分区内并行性和分区间并行性的性能优点。请考虑 MDC 表的下列具体优点:

- 维块索引查找操作可以标识表的所需部分，并且能够快速地仅扫描所需的块。
- 因为块索引小于记录标识（RID）索引，所以查找速度更快。
- 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 相结合。
- 保证在扩展数据块内集群数据，这有助于提高检索速度。
- 如果可以使用转出方法，那么删除行的速度将更快。

请考虑名为 SALES 的 MDC 表的以下简单示例，这个表对 REGION 和 MONTH 列定义了维:

```
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操作，以寻找月份为三月且地区为 SE 的块。然后，它可以只扫描那些块，以便快速地访存结果集。

转出删除

当条件允许使用转出方法来进行删除时，将使用这种从 MDC 表中删除行的更高效方法。必需的条件包括:

- 该 DELETE 语句是搜索型 DELETE，而不是定位型 DELETE（该语句不使用 WHERE CURRENT OF 子句）。
- 没有 WHERE 子句（将删除所有行），或者 WHERE 子句只包含应用于维的条件。
- 定义表时，未指定 DATA CAPTURE CHANGES 子句。
- 该表不是引用完整性关系中的父表。
- 未对该表定义 ON DELETE 触发器。
- 未在任何立即刷新的 MQT 中使用该表。
- 如果级联删除操作的外键是该表的维列的子集，那么它可能适合于转出。
- 在由 CREATE TRIGGER 语句的 OLD TABLE AS 子句指定的触发 SQL 操作之前，该 DELETE 语句不能出现在对临时表执行并标识了受影响行集的 SELECT 语句中。

在转出删除期间，不会记录所删除的记录。而是，将通过重新格式化页的某些部分使包含这些记录的页表现为空页。将会记录对重新格式化的部分所作的更改，但不会记录这些记录本身。

*立即清除转出*这一缺省行为是指，在删除时清除 RID 索引。还可以通过将注册表变量 **DB2_MDC_ROLLOUT** 设置为 IMMEDIATE，或者通过对 SET CURRENT MDC ROLLOUT MODE 语句指定 IMMEDIATE 来指定此方式。与标准删除操作相比，索引更新的日志记录没有变化，因此，性能提高取决于 RID 索引的数目。RID 索引越少，性能就越好，衡量标准是总时间和日志空间所占的百分比。

可以使用以下公式来估算可以节省的日志空间量:

```
S + 38*N - 50*P
```

其中，*N* 是已删除的记录数，*S* 是已删除的记录的总大小（包括空指示符和 VARCHAR 长度之类的开销），*P* 是包含已删除的记录的块中的页数。此数值是实际日志数据的缩减量。节省的所需活动日志空间量是此值的两倍，这是因为，还将节省为回滚操作保留的空间。

另外，在落实事务之后，可以使用*延迟清除转出*方法来更新 RID 索引。还可以通过将注册表变量 **DB2_MDC_ROLLOUT** 设置为 DEFER，或者通过对 SET CURRENT MDC ROLLOUT MODE 语句指定 DEFERRED 来指定此方式。在延迟转出方式下，将在删除操作落实后在后台以异步方式清除 RID 索引。在删除任务非常大型或者已对表定义大量 RID 索引的情况下，使用这种转出方法可以非常快速地执行删除。整体清除操作的速度也有所提高，这是因为，执行延迟索引清除时将以并行方式清除索引，而执行立即索引清除时将逐行清除索引中的每一行。并且，DELETE 语句的事务日志空间需求显著降低，这是因为，索引按索引页而不是按索引键来更新异步索引清除日志。

注: 延迟清除转出操作需要更多内存资源，这些内存资源将从数据库堆中获取。如果数据库管理器无法分配它所需的内存结构，那么延迟清除转出操作将失败，并将一条消息写入管理通知日志。

何时使用延迟清除转出方法

如果删除性能对于您而言是最重要的因素，并且已对表定义 RID 索引，那么应使用延迟清除转出方法。注意，在进行索引清除之前，对已转出的块进行基于索引的扫描会稍微降低性能，这取决于已转出的数据量。在决定执行立即索引清除操作和延迟索引清除操作时，还应该考虑下列问题:

- 删除操作的规模

  对于非常大型的删除任务，请选择延迟清除转出方法。在对许多小型 MDC 表频繁发出维 DELETE 语句的情况下，异步清除索引对象所产生的开销要比删除操作期间节省的时间的价值更高。

- 索引的数目和类型

  如果表包含大量 RID 索引，并且需要对这些索引执行行级别处理，那么应使用延迟清除转出方法。

- 块可用性

  如果您希望由删除操作释放的块空间在 DELETE 语句落实后立即可用，那么请使用立即清除转出方法。

- 日志空间

  如果日志空间有限，那么应对大型删除任务使用延迟清除转出方法。

- 内存约束

  对于所有已暂挂延迟清除操作的表，延迟清除转出操作将耗用更多的数据库堆空间。

要在删除期间禁止转出行为，请将 **DB2_MDC_ROLLOUT** 注册表变量设置为 OFF，或者对 SET CURRENT MDC ROLLOUT MODE 语句指定 NONE。

注: 在 DB2 版本 9.7 和更高版本的发行版中，不支持对具有分区 RID 索引的数据分区 MDC 表执行延迟清除转出。仅支持 NONE 和 IMMEDIATE 方式。如果 **DB2_MDC_ROLLOUT** 注册表变量设置为 DEFER，或者 CURRENT MDC ROLLOUT MODE 专用寄存器设置为 DEFERRED 以覆盖 **DB2_MDC_ROLLOUT** 设置，那么清除转出类型将为 IMMEDIATE。

如果 MDC 表仅存在非分区 RID 索引，那么支持执行延迟索引清除转出。

分区表的优化策略

“数据分区消除”功能是指数据库服务器能够根据查询谓词确定，只需要访问表的部分数据分区即可应答查询。对分区表运行决策支持查询时，“数据分区消除”功能特别有用。

分区表使用了数据组织方案，即，表数据根据该表中一个或多个表分区键列中的值分布到多个存储对象（称为数据分区或范围）中。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在同一个表空间中。

以下示例演示数据分区消除功能在性能方面的好处。

```
create table custlist(
  subsdate date, province char(2), accountid int)
  partition by range(subsdate) (
    starting from '1/1/1990' in ts1,
    starting from '1/1/1991' in ts1,
    starting from '1/1/1992' in ts1,
    starting from '1/1/1993' in ts2,
    starting from '1/1/1994' in ts2,
    starting from '1/1/1995' in ts2,
    starting from '1/1/1996' in ts3,
    starting from '1/1/1997' in ts3,
    starting from '1/1/1998' in ts3,
    starting from '1/1/1999' in ts4,
    starting from '1/1/2000' in ts4,
    starting from '1/1/2001'
    ending '12/31/2001' in ts4)
```

假定您只对 2000 年的客户信息感兴趣。

```
select * from custlist
  where subsdate between '1/1/2000' and '12/31/2000'
```

如图 33 所示，数据库服务器确定只需要访问表空间 TS4 中的一个数据分区即可解决此查询。

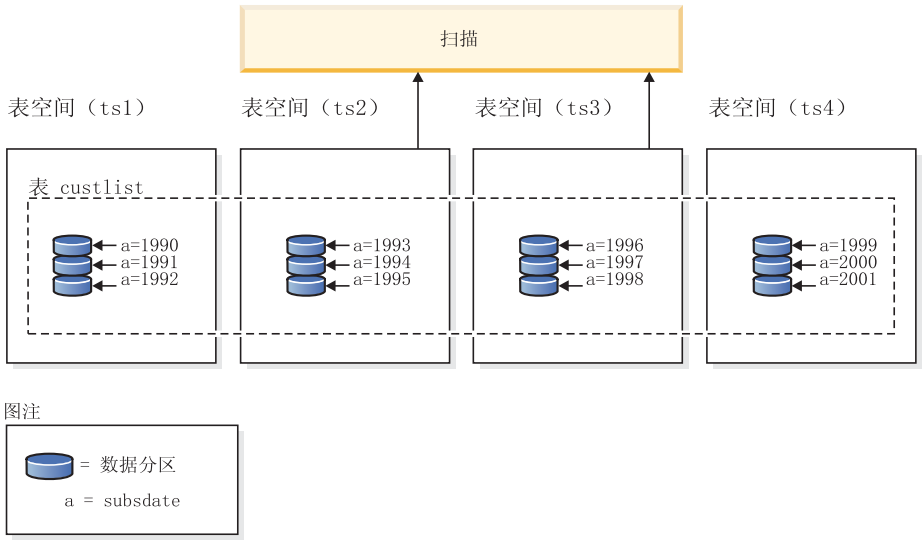


图 33. 数据分区消除功能在性能方面的好处

另一个数据分区消除示例基于以下方案:

```
create table multi (
  sale_date date, region char(2))
  partition by (sale_date) (
    starting '01/01/2005'
    ending '12/31/2005'
    every 1 month)

create index sx on multi(sale_date)

create index rx on multi(region)
```

假定您发出以下查询:

```
select * from multi
  where sale_date between '6/1/2005'
    and '7/31/2005' and region = 'NW'
```

在不进行表分区时，一种可能的方案是索引“与”（AND）。索引“与”（AND）执行下列任务:

- 读取每个索引中的所有相关索引条目
- 保存两组行标识（RID）
- 对 RID 进行匹配，以确定哪些 RID 同时出现在这两个索引中
- 使用 RID 对行进行访存

如图 34 所示，在进行表分区的情况下，将读取索引以查找 REGION 和 SALE_DATE 的匹配项，从而快速检索匹配的行。

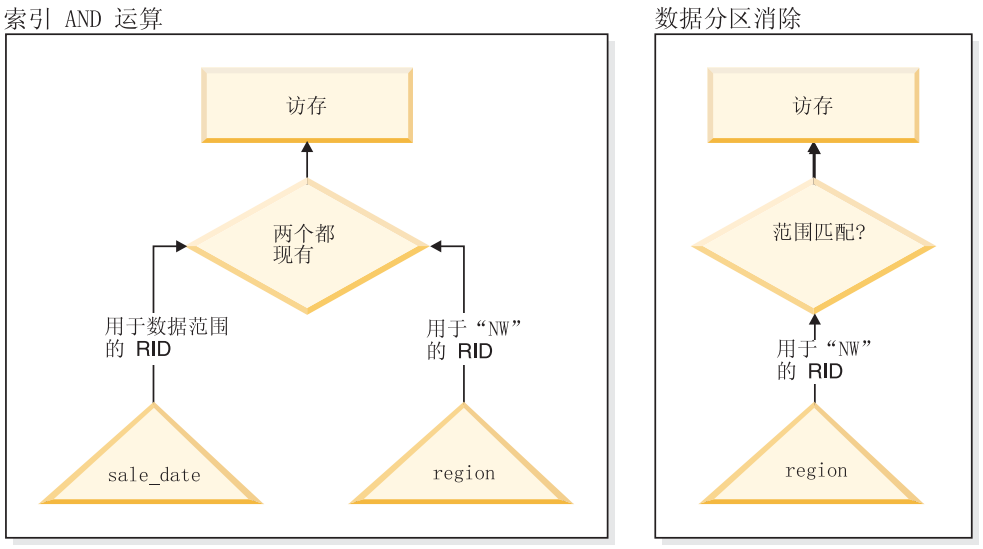


图 34. 用于进行表分区和索引“与”（AND）操作的优化器决策路径

DB2 说明

您还可以使用说明工具来确定查询优化器所选择的数据分区消除方案。“DP Elim Predicates”信息显示扫描了哪些数据分区来解决以下查询:

```
   select * from custlist
     where subsdate between '12/31/1999' and '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
     FALSE
DPLSTPRT: (List of data partitions accessed)
     9-11
DPNUMPRT: (Number of data partitions accessed)
   3

DP Elim Predicates:
------------------
Range 1)
   Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)
```

```
Objects Used in Access Plan:
 ---------------------------

   Schema: MRSRINI
   Name:    CUSTLIST
   Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
   Last statistics update:    2005-11-30-14.21.34.339392
     Number of columns:      3
     Number of rows:      100000
   Width of rows:      19
   Number of buffer pool pages:    1200
     Number of data partitions:    12
     Distinct row values:      No
    Tablespace name:      <VARIOUS>
```

多列支持

在将多个列用作表分区键的情况下，数据分区消除功能将起作用。例如:

```
create table sales (
  year int, month int)
 partition by range(year, month) (
    starting from (2001,1)
    ending at (2001,3) in ts1,
   ending at (2001,6) in ts2,
   ending at (2001,9) in ts3,
   ending at (2001,12) in ts4,
   ending at (2002,3) in ts5,
   ending at (2002,6) in ts6,
   ending at (2002,9) in ts7,
   ending at (2002,12) in ts8)

select * from sales where year = 2001 and month < 8
```

查询优化器判断只需要访问 TS1、TS2 和 TS3 中的数据分区即可解决此查询。

注: 在表分区键由多个列构成的情况下，仅当存在作用于组合键的前导列的谓词时才能实现数据分区消除，这是因为，用于表分区键的非前导列不是独立的。

多范围支持

可以对包含多个范围的数据分区（即，通过“或”（OR）运算聚集到一起的数据分区）实现数据分区消除。在使用上一个示例中创建的 SALES 表的情况下，执行下列查询:

```
select * from sales
  where (year = 2001 and month <= 3)
    or (year = 2002 and month >= 10)
```

数据库服务器将只访问 2001 年第一季度和 2002 年最后一个季度的数据。

生成列

可以将生成列用作表分区键。例如:

```
create table sales (
  a int, b int generated always as (a / 5))
  in ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
  partition by range(b) (
    starting from (0)
    ending at (1000) every (50))
```

在本示例中，将作用于生成列的谓词用于数据分区消除。此外，如果用于生成列的表达式是单调的，那么数据库服务器将把作用于源列的谓词转换为作用于生成列的谓词，从而对生成列启用数据分区消除。例如:

```
select * from sales where a > 35
```

数据库服务器根据作用于 a 的谓词（a > 35）来生成作用于 b 的额外谓词（b > 7），从而允许进行数据分区消除。

连接谓词

如果将连接谓词下推到表访问级别，那么连接谓词也可以用于数据分区消除。只有嵌套循环连接（NLJN）的内连接才会将连接谓词下推到表访问级别。

请考虑下列表:

```
create table t1 (a int, b int)
  partition by range(a,b) (
    starting from (1,1)
    ending (1,10) in ts1,
    ending (1,20) in ts2,
    ending (2,10) in ts3,
    ending (2,20) in ts4,
    ending (3,10) in ts5,
    ending (3,20) in ts6,
    ending (4,10) in ts7,
    ending (4,20) in ts8)

create table t2 (a int, b int)
```

将使用下面这两个谓词:

```
P1: T1.A = T2.A
P2: T1.B > 15
```

在本示例中，由于不知道连接的外值，因此无法确定将在编译时访问的确切数据分区。在这种情况下，以及在使用主变量或参数标记的情况下，将在运行时绑定必需的值时进行数据分区消除。

在运行时，当 T1 是 NLJN 的内表时，将根据 T2.A 的每个外值的谓词自动进行数据分区消除。在运行时，将对外值 T2.A = 3 应用谓词 T1.A = 3 和 T1.B > 15，这样就限定了在表空间 TS6 中要访问的数据分区。

假定表 T1 和 T2 中的列 A 包含下列值:

外表 **T2**: 列 **A**	内表 **T1**: 列 **A**	内表 **T1**: 列 **B**	内表 **T1**: 数据分区位置
2	3	20	TS6
3	2	10	TS3
3	2	18	TS4
	3	15	TS6
	1	40	TS3

要执行嵌套循环连接（假定对内表进行表扫描），数据库管理器执行下列步骤:

1. 读取 T2 中的第一行。A 的值是 2。
2. 在连接谓词 T1.A = T2.A 中将 T2.A 值（此值为 2）与列 T2.A 绑定。该谓词变成 T1.A = 2。

3. 使用谓词 T1.A = 2 和 T1.B > 15 来应用数据分区消除。这将限定表空间 TS4 中的数据分区。
4. 在应用 T1.A = 2 和 T1.B > 15 之后，扫描表 T1 的表空间 TS4 中的数据分区，直到找到一行为止。找到的第一个合格行是 T1 的行 3。
5. 连接匹配的行。
6. 扫描表 T1 的表空间 TS4 中的数据分区，直到找到下一个匹配项（T1.A = 2 并且 T1.B > 15）为止。再也找不到其他行。
7. 对 T2 的下一行（将 A 的值替换为 3）重复步骤 1 至 6，直到处理完 T2 的所有行为止。

基于 XML 数据的索引

从 DB2 版本 9.7 修订包 1 开始，可以对分区表创建基于 XML 数据的分区索引或者非分区索引。缺省情况下将创建分区索引。

在表插入、更新和删除操作期间，数据库管理器将按照维护任何其他基于分区表的关系索引的方式来维护分区 XML 索引和非分区 XML 索引。为了提高查询处理速度，将按照使用基于非分区表中 XML 数据的索引的方式来使用基于分区表中 XML 数据的非分区索引。通过使用查询谓词，有可能确定只需要访问分区表中的部分数据分区即可应答查询。

基于 XML 列的数据分区消除功能和索引可以配合工作，以提高查询性能。请考虑以下分区表:

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

现在，考虑以下查询:

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

优化器可以根据谓词 a > 21 立即消除前两个分区。如果优化器在查询方案中选择基于列表 B 中 XML 数据的非分区索引，那么使用基于 XML 数据的索引的索引扫描将能够利用优化器的数据分区消除结果，并且将只返回属于关系数据分区消除谓词未消除的分区的结果。

使用具体化查询表改进查询优化

具体化查询表（MQT）能够显著缩短复杂查询的响应时间。

对于需要执行下列一项或多项操作的查询而言尤其如此:

- 基于一个或多个维聚集数据
- 连接和聚集涉及一组表的数据
- 数字来自通常访问的数据子集；即，来自“热”水平或垂直数据库分区
- 在分区数据库环境中对表或表的一部分中的数据进行重新分区

MQT 的知识已集成到 SQL 和 XQuery 编译器中。在编译器中，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 取代访问基本表的查询。如果使用 MQT，那么说明工具可以提供关于选择了哪个 MQT 的信息。在这种情况下，用户必须对基本表（而不是重新路由的 MQT）具有访问特权。

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和索引来优化数据访问以及通过调用 RUNSTATS 实用程序优化数据访问的准则也适用于 MQT。

为了帮助您理解 MQT 的功能，以下示例说明多维分析查询可以如何利用 MQT。假定数据库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。每项交易都包含一批一起购买的商品。此模式属于多星型模式，这是因为共有两个大型表，一个包含交易商品，另一个标识采购交易。

每一项交易由三个分层维描述：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系列的标准表中。位置层次结构包含城市、州或省以及国家或地区信息，并且存储在单个非标准表中。时间层次结构包含日、月和年信息，并且在单一日期字段中进行编码。日期维是使用内置函数从交易的日期字段中抽取的。此模式中的其他表表示客户的帐户信息以及客户信息。

在下列层次结构的每一层创建用于存储销售信息的 MQT:

- 产品
- 位置
- 时间（由年、月和日组成）

存储的这些聚集数据可以满足许多查询。以下示例说明如何创建一个 MQT，该 MQT 按照产品组和产品系列维、按照城市、州或省和国家或地区维以及按照时间维来计算销售金额和销售件数。在它的 GROUP BY 子句中，还包括多个其他的列。

```
create table dba.pg_salessum
  as (
    select l.id as prodline, pg.id as pgroup,
      loc.country, loc.state, loc.city,
      l.name as linename, pg.name as pgname,
      year(pdate) as year, month(pdate) as month,
      t.status,
      sum(ti.amount) as amount,
      count(*) as count
    from cube.transitem as ti, cube.trans as t,
      cube.loc as loc, cube.pgroup as pg, cube.prodline as l
    where
      ti.transid = t.id and
      ti.pgid = pg.id and
      pg.lineid = l.id and
      t.locid = loc.id and
      year(pdate) > 1990
    group by l.id, pg.id, loc.country, loc.state, loc.city,
      year(pdate), month(pdate), t.status, l.name, pg.name
  )
data initially deferred refresh deferred;

refresh table dba.pg_salessum;
```

可以利用此类预先计算的总额的查询包括:

- 按月和产品组的销售额
- 1990 年以来的总销售额
- 1995 年或 1996 年的销售额

- 特定产品组或产品系列的销售总额
- 1995 年和 1996 年特定产品组或产品系列的销售总额
- 特定国家或地区的销售总额

尽管该 MQT 未包括以上任何一个查询的准确答案，但使用 MQT 计算答案的成本可能显著低于使用大型基本表的成本，这是因为此答案的其中一部分已计算完毕。MQT 可以减少成本高昂的基本数据连接、排序和聚集需求。

下列样本查询可以使用示例 MQT 中已经计算出的结果，因此性能将显著提高。

第一个查询返回 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);
```

第二个查询按产品组返回 1995 和 1996 年的销售总额:

```
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，使用 MQT 在响应时间方面的潜在改进就越显著。MQT 能够有效消除各个查询之间重叠的工作。这些计算只是在 MQT 构建时执行一次并在 MQT 每次被刷新时执行一次，MQT 的内容可以在多个查询的执行期间重复使用。

说明工具

DB2 说明工具提供有关优化器为 SQL 或 XQuery 语句选择的访问方案的详细信息。

此信息描述用于选择访问方案的决策条件，并可以帮助您调整语句或实例配置以提高性能。更确切而言，说明信息可以帮助您:

- 了解数据库管理器如何访问表和索引以满足您的查询
- 评估性能调整操作。在更改语句或配置之后，请检查新的说明信息，以确定该操作对性能的影响。

捕获的信息包括:

- 用于处理查询的操作的序列
- 成本信息

- 谓词和每个谓词的选择性估算值
- 捕获说明信息时在 SQL 或 XQuery 语句中引用的所有对象的统计信息
- 用于重新优化 SQL 或 XQuery 语句的主变量、参数标记或专用寄存器的值

要调用说明工具，请发出 EXPLAIN 语句；此语句将捕获有关为特定可说明语句选择的访问方案的信息并将此信息写入说明表。在发出 EXPLAIN 之前，您必须创建说明表。您还可以设置 CURRENT EXPLAIN MODE 或 CURRENT EXPLAIN SNAPSHOT，它们是用于控制说明工具的行为的专用寄存器。

有关使用 EXPLAIN 实用程序所需的特权和权限的信息，请参阅 EXPLAIN 语句的描述。可以将 EXPLAIN 权限授予需要访问说明信息但不需要访问数据库中存储的数据的个人。此权限是数据库管理员权限的子集，不具有访问表中存储的数据所需的固有特权。

要显示说明信息，可以使用命令行工具或 Visual Explain。使用的工具确定了如何设置用于控制说明工具的行为的专用寄存器。例如，如果您期望只使用 Visual Explain，那么只需要捕获快照信息。如果您期望使用其中一个命令行实用程序或者定制 SQL 或 XQuery 语句对说明表执行详细分析，那么应捕获所有说明信息。

使用说明工具来调整 SQL 语句

说明工具用于显示查询优化器选择的用于运行 SQL 语句的查询访问方案。

它包含有关用于运行该 SQL 语句的关系操作的全部详细信息，例如方案运算符、它们的自变量、执行顺序和成本。由于查询访问方案是查询性能的其中一项最关键因素，因此，在诊断查询性能问题时，理解说明工具的输出至关重要。

通常，说明信息用于:

- 了解应用程序性能的变化原因
- 评估性能调整工作

分析性能变化

为了帮助您了解查询性能发生变化的原因，请执行下列步骤以获取“之前的和之后的”说明信息:

1. 在进行任何更改前，捕获查询的说明信息并保存所生成的说明表。此外，也可以保存 db2exfmt 实用程序的输出。但是，将说明信息保存到说明表使您能够方便地通过 SQL 对其进行查询，并且有助于执行更复杂的分析。并且，这还将提供在关系 DBMS 中存储数据所具有的全部明显维护优势。您可以在任何时候运行 db2exfmt 工具。
2. 如果您无法通过访问 Visual Explain 来查看此信息，那么请保存或打印当前的目录统计信息。此外，还可以使用 db2look 命令来帮助执行此任务。在 DB2 版本 9.7 中，可以在说明表填充完毕后收集说明快照。说明快照包含说明语句时的所有相关统计信息。db2exfmt 实用程序将自动格式化快照中包含的统计信息。这在使用自动或实时收集统计信息功能时尤其重要，其原因在于，用于进行查询优化的统计信息可能还不在系统目录表中，或者，在语句被说明直到从系统目录中检索统计信息的这段时间内，这些信息被更改。
3. 保存或打印数据定义语言（DDL）语句，其中包括 CREATE TABLE、CREATE VIEW、CREATE INDEX 和 CREATE TABLESPACE 的那些语句。db2look 命令也将执行此任务。

以此方式收集的信息为将来的分析提供参考点。对于动态 SQL 语句，可以在首次运行应用程序时收集此信息。对于静态 SQL 语句，也可以在绑定时收集此信息。在进行重大系统更改（例如安装新的服务级别或 DB2 发行版）之前或者在重大配置更改（例如添加或删除数据库分区以及重新分布数据）之前收集此信息尤其重要。这是因为，这些类型的系统更改可能导致对访问方案进行负面更改。虽然访问方案回归的情况很罕见，但提供此信息可以帮助您更快地解决性能回归问题。要进行性能变化分析，请将先前收集的信息与您启动分析时收集的关于查询和环境的信息作比较。

举一个简单的例子，您进行的分析可能指出，不再将某个索引用作访问路径的一部分。通过使用 Visual Explain 或 db2exfmt 显示的目录统计信息，您可能注意到，索引层数（NLEVELS 列）现在大大高于最初将查询与数据库绑定时的情况。于是，您可以选择执行下列其中一项操作:

- 重组该索引
- 收集表和索引的新统计信息
- 在重新绑定查询时收集说明信息

执行其中一项操作之后，再次检查访问方案。如果再次使用该索引，那么查询性能可能不再有问题。如果仍未使用该索引，或者性能仍有问题，那么请执行第二次操作并检查结果。重复这些步骤，直到解决问题为止。

评估性能调整工作

您可以执行多项操作来帮助提高查询性能，例如调整配置参数、添加容器或者收集新的目录统计信息。

在其中任何一个领域进行更改后，可以使用说明工具来确定该更改对所选访问方案的影响。例如，如果您根据索引准则来添加索引或具体化查询表（MQT），那么说明数据可以帮助您确定是否已实际地按预期方式使用该索引或具体化查询表。

尽管说明输出提供的信息使您能够确定所选访问方案及其相关成本，但准确测量查询的性能提高情况的唯一方法是使用基准程序测试技术。

有关捕获说明信息的准则

在编译 SQL 或 XQuery 语句时，可以请求捕获说明数据。

如果在运行时编译增量绑定 SQL 或 XQuery 语句，那么在运行时而不是绑定时将数据放入说明表。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。

仅当编译 SQL 或 XQuery 语句时才会捕获说明信息。在初始编译之后，当环境更改要求再次编译动态查询语句时，或者当说明工具处于活动状态时，将再次进行编译。如果对同一个查询语句发出相同的 PREPARE 语句，那么每当准备或执行此语句时，都将编译该查询并捕获说明数据。

如果使用 REOPT ONCE 或 ALWAYS 绑定选项来绑定程序包，那么将编译包含主变量、参数标记、全局变量或专用寄存器的 SQL 或 XQuery 语句；并且，如果在编译时知道这些变量的实际值，那么将使用这些值来创建访问路径，如果不知道实际值，那么将使用缺省的估算值来创建访问路径。

如果使用了 REOPT ONCE 选项，那么将尝试使指定的 SQL 或 XQuery 语句与程序包高速缓存中的同一语句匹配。将使用这个已重新优化并高速缓存的查询语句的值来

重新优化所指定查询语句。如果用户具有必需的访问特权，那么说明表将包含新重新优化的访问方案以及用于执行重新优化的值。

在多分区数据库系统中，应该在最初编译和使用 REOPT ONCE 重新优化该语句时所在的数据库分区中说明该语句，否则将返回错误。

在说明表中捕获信息

- 静态或增量绑定 SQL 和 XQuery 语句

  请在 BIND 或 PREP 命令中指定 EXPLAIN ALL 或 EXPLAIN YES 选项，或者在源程序中包括静态的 EXPLAIN 语句。

- 动态 SQL 和 XQuery 语句

  在下列任何一种情况下，都将捕获说明表信息。

  – 如果 CURRENT EXPLAIN MODE 专用寄存器设置为:
    - YES: SQL 和 XQuery 编译器将捕获说明数据并执行该查询语句。
    - EXPLAIN: SQL 和 XQuery 编译器将捕获说明数据，但不执行该查询语句。
    - RECOMMEND INDEXES: SQL 和 XQuery 编译器将捕获说明数据并将建议的索引放入 ADVISE_INDEX 表，但不执行该查询语句。
    - EVALUATE INDEXES: SQL 和 XQuery 编译器将使用用户放入 ADVISE_INDEX 表的索引进行评估。在此方式下，将说明所有动态语句，就像这些虚拟索引可用一样。如果虚拟索引能够提高语句性能，那么查询编译器将选择使用这些索引。否则，将忽略这些索引。要了解所建议的索引是否有用，请查看 EXPLAIN 结果。
    - REOPT: 如果可以获得主变量、参数标记、全局变量或专用寄存器的实际值，那么在执行时的重新优化语句期间，查询编译器将捕获静态或动态 SQL 或 XQuery 语句的说明数据。

- 如果已在 BIND 或 PREP 命令中指定 EXPLAIN ALL 选项，那么查询编译器在运行时将捕获动态 SQL 和 XQuery 语句的说明数据，即使 CURRENT EXPLAIN MODE 专用寄存器设置为 NO 亦如此。

捕获说明快照信息

当您请求获取说明快照时，说明信息将采用 Visual Explain 所需的格式存储在 EXPLAIN_STATEMENT 表的 SNAPSHOT 列中。其他应用程序无法使用此格式。有关说明快照的其他信息（包括关于数据对象和数据运算符的信息）由 Visual Explain 本身提供。

在编译 SQL 或 XQuery 语句并已请求获取说明数据时，将捕获说明快照数据，如下所示:

- 静态或增量绑定 SQL 和 XQuery 语句

  如果在 BIND 或 PREP 命令中指定了 EXPLSNAP ALL 或 EXPLSNAP YES 子句，或者源程序包含使用 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的静态 EXPLAIN 语句，那么将捕获说明快照。

- 动态 SQL 和 XQuery 语句

  在下列任何一种情况下，都将捕获说明快照。

– 您发出带有 FOR SNAPSHOT 或 WITH SNAPSHOT 子句的 EXPLAIN 语句。对于前者，将只捕获说明快照信息；对于后者，将捕获所有说明信息。
– 如果 CURRENT EXPLAIN SNAPSHOT 专用寄存器设置为:
  - YES: SQL 和 XQuery 编译器将捕获说明快照数据并执行该查询语句。
  - EXPLAIN: SQL 和 XQuery 编译器将捕获说明快照数据，但不执行该查询语句。
– 在 BIND 或 PREP 命令中指定 EXPLSNAP ALL 选项。查询编译器在运行时将捕获说明快照数据，即使 CURRENT EXPLAIN SNAPSHOT 专用寄存器设置为 NO 亦如此。

关于捕获部分说明信息的准则

部分说明功能将捕获（直接捕获或者通过工具捕获）有关仅使用运行时部分内容的语句的说明信息。部分说明类似于 db2expln 命令所提供的功能，但是部分说明提供的详细信息级别接近于说明工具所提供的详细信息级别。

通过使用运行时部分的内容来说明语句，就可以获得有关实际上将运行的对象的信息和诊断（如果是在执行之后才捕获此部分，那么将获得已经运行的对象的信息和诊断）；而如果是发出 EXPLAIN 语句则相反，发出 EXPLAIN 语句时可能会生成另一种访问方案（例如，对于动态 SQL 语句，可能自从上一次执行该语句以来已经更新了统计信息，从而导致在 EXPLAIN 语句编译要说明的语句时将选择另一种访问方案）。

部分说明接口将为说明表填充的信息与 EXPLAIN 语句所生成的信息相似，但是也存在一些差别。将数据写入说明表之后，可由您想使用的任何现有说明工具（例如，db2exfmt 命令）来进行处理。

部分说明接口

以下列表中有四个接口过程，它们可以执行部分说明。这些过程之间的唯一区别就是所提供的输入不同（即，用来查找部分的方法不同）：

EXPLAIN_FROM_ACTIVITY
将应用程序标识、活动标识、工作单元标识和活动事件监视器名称作为输入。此过程将在活动事件监视器中搜索与此活动相对应的部分（一个 SQL 活动就是具体执行某个部分）。由于具体执行了该部分，因此使用此接口的部分说明包含部分实际值。

EXPLAIN_FROM_CATALOG
将程序包名、程序包模式、唯一标识和部分编号作为输入。此过程将搜索特定部分的目录表。

EXPLAIN_FROM_DATA
将可执行文件标识、部分和语句文本作为输入。

EXPLAIN_FROM_SECTION
将可执行文件标识和位置作为输入，而位置是通过使用下列其中一种方法指定的:

- 内存程序包高速缓存
- 程序包高速缓存事件监视器名称

此过程将在给定位置搜索部分。

可执行文件标识将唯一而且一致地标识部分。可执行文件标识是在数据服务器中为每个已经执行的部分生成的不透明的二进制标记。可执行文件标识被用作输入来查询部分的监视数据以及执行部分说明。

在每种情况下，此过程都将使用所标识的运行时部分中包含的信息来执行说明，然后将说明信息写入由 *explain_schema* 输入参数所标识的说明表中。在调用此过程之后，由调用者负责执行落实。

部分说明与 EXPLAIN 语句输出之间的差别:

发出部分说明之后所获得的结果与运行 EXPLAIN 语句之后所收集的结果相似。按照受影响的说明表和隐含意义（如果有）描述了上述结果与 db2exfmt 实用程序所生成的输出之间的细微差别。

存储过程输出参数
EXPLAIN_REQUESTER、EXPLAIN_TIME、SOURCE_NAME、SOURCE_SCHEMA 和 SOURCE_VERSION 由用来查找有关说明表所包含部分的信息的键组成。对任何现有说明工具（例如，db2exfmt）使用这些参数，以格式化从该部分检索到的说明信息。

EXPLAIN_INSTANCE 表

对于由部分说明所生成的行，按不同方式设置下面各列:

- EXPLAIN_OPTION 设置为值 S
- SNAPSHOT_TAKEN 始终设置为 N
- REMARKS 始终为 NULL

EXPLAIN_STATEMENT 表

当部分说明生成了说明输出时，EXPLAIN_LEVEL 列设置为值 S。请您一定要注意，EXPLAIN_LEVEL 列是此表的主键的一部分，也是其他大多数说明表的外键的一部分；因此，其他那些表中也存在此 EXPLAIN_LEVEL 值。

在 EXPLAIN_STATEMENT 表中，当 EXPLAIN_LEVEL = S 时，将提供通常与 EXPLAIN_LEVEL = P 的行相关联的其余列值（SNAPSHOT 除外）。当 EXPLAIN_LEVEL 为 S 时，SNAPSHOT 始终为 NULL。

如果在生成部分说明时原始语句不可用（例如，如果未对 EXPLAIN_FROM_DATA 过程提供语句文本，就会出现这种上述情况），那么在 EXPLAIN_LEVEL 设置为 0 时，STATEMENT_TEXT 将设置为字符串 UNKNOWN。

在部分说明的 db2exfmt 输出中，在已优化的语句后面将显示下面这一额外的行:

```
Explain level:    Explain from section
```

EXPLAIN_OPERATOR 表

就用于记录成本的所有列来说，在发出部分说明之后，只有 TOTAL_COST 和 FIRST_ROW_COST 列会填充值。所有其他用于记录成本的列的值都为 -1。

在部分说明的 db2exfmt 输出中，获得了下列差异:

- 在访问方案图中，I/O 成本显示为 NA

- 在每个运算符的详细信息中，显示的成本只有累积总成本（Cumulative Total Cost）和第一行的累积成本（Cumulative First Row Cost）

EXPLAIN_PREDICATE 表

无差别。

EXPLAIN_ARGUMENT 表

当发出部分说明时，只有少量的自变量类型未写入 EXPLAIN_ARGUMENT 表。

EXPLAIN_STREAM 表

在发出部分说明之后，下列各列没有值:

- COLUMN_NAMES
- SINGLE_NODE
- PARTITION_COLUMNS
- SEQUENCE_SIZES

在发出部分说明之后，下列各列的值始终为 -1:

- COLUMN_COUNT
- PREDICATE_ID

在部分说明的 db2exfmt 输出中，在每个操作程序的输入流和输出流部分中都将省略所列示的这些列中的信息。

EXPLAIN_OBJECT 表

在发出部分说明之后，STATS_SRC 列始终设置为空字符串，CREATE_TIME 列设置为 NULL。

在发出部分说明之后，下列各列的值始终为 -1:

- COLUMN_COUNT
- WIDTH
- FIRSTKEYCARD
- FIRST2KEYCARD
- FIRST3KEYCARD
- FIRST4KEYCARD
- SEQUENTIAL_PAGES
- DENSITY
- AVERAGE_SEQUENCE_GAP
- AVERAGE_SEQUENCE_FETCH_GAP
- AVERAGE_SEQUENCE_PAGES
- AVERAGE_SEQUENCE_FETCH_PAGES
- AVERAGE_RANDOM_PAGES
- AVERAGE_RANDOM_FETCH_PAGES
- NUMRIDS

- NUMRIDS_DELETED
- NUM_EMPTY_LEAFS
- ACTIVE_BLOCKS
- NUM_DATA_PART

在部分说明的 db2exfmt 输出中，从接近输出底部找到的每个表和每个索引的统计信息中，将省略所列示的这些列中的信息。

部分说明不会在其输出中包含编译器所引用的对象（即，OBJECT_TYPE 以 + 开头的那些行）。这些对象不会显示在 db2exfmt 输出中。

捕获和访问部分实际值:

部分实际值是在执行访问方案的某个部分期间收集的运行时统计信息。要捕获具有实际值的部分，请使用活动事件监视器。要访问部分实际值，请使用 EXPLAIN_FROM_ACTIVITY 存储过程执行部分说明。

为了能够查看部分实际值，必须对捕获了其部分实际值的部分执行部分说明（也就是说，该部分和部分实际值都是说明工具的输入）。此处提供了有关启用、捕获和访问部分实际值的信息。

启用部分实际值

仅当已经启用了部分实际值，才会在运行时更新这些部分实际值。可使用 **DB2_SYSTEM_MONITOR_SETTINGS** 注册表变量的 SECTION_ACTUALS 参数来启用部分实际值。要启用部分实际值，请将此参数设置为 TRUE（缺省值为 FALSE）。例如:

```
db2set DB2_SYSTEM_MONITOR_SETTINGS=SECTION_ACTUALS:TRUE
```

此注册表变量设置是动态设置。将此注册表变量设置为 TRUE 之后，将在执行部分期间更新部分实际值。

注: 在更新此注册表变量之前由应用程序执行的任何语句，在同一应用程序中再次运行该语句时将保持其原始的部分实际值。也就是说，如果应用程序在禁用了部分实际值的情况下发出某个语句，接着通过该注册表变量启用了部分实际值，然后重新发出该语句，那么第二次执行该语句时仍然会禁用这些部分实际值。首次发出该语句的其他应用程序将为该语句启用部分实际值。

捕获部分实际值

用于捕获部分以及部分实际值的机制是使用活动事件监视器。如果启用了收集活动信息，那么在活动执行完毕时，活动事件监视器就会写出该活动的详细信息。通过对工作负载、服务类、阈值或工作操作使用 COLLECT ACTIVITY DATA 子句来启用收集活动信息。要指定收集某个部分和实际值（如果可用并且已启用），请使用 COLLECT ACTIVITY DATA 子句的 SECTION 选项。例如，以下语句指示当该语句完成时，由与 WL1 工作负载相关联的连接发出的任何 SQL 语句将让任何活动的活动事件监视器收集信息（包括部分和实际值）:

```
ALTER WORKLOAD WL1 COLLECT ACTIVITY DATA WITH DETAILS,SECTION
```

在分区数据库环境中，如果对要执行的语句应用了 COLLECT ACTIVITY DATA 子句，并且 COLLECT ACTIVITY DATA 子句同时指定了 SECTION 关键字和 ON ALL

DATABASE PARTITIONS 子句，那么活动事件监视器将在执行了活动的所有分区中捕获部分实际值。如果未指定 ON ALL DATABASE PARTITIONS 子句，那么将仅捕获协调程序分区中的实际值。

局限性 捕获部分实际值时存在下列局限性:

- 使用 WLM_CAPTURE_ACTIVITY_IN_PROGRESS 存储过程将有关当前正在执行的活动的信息发送至活动事件监视器时，将不会捕获部分实际值。由 WLM_CAPTURE_ACTIVITY_IN_PROGRESS 存储过程生成的所有活动事件监视器记录的 partial_record 列值都为 1。
- 在违反了后触发阈值的情况下，将仅捕获协调程序分区中的部分实际值。
- 必须将说明表迁移到 DB2 版本 9.7 修订包 1 或更高版本之后，才能使用部分说明来访问部分实际值。如果尚未迁移说明表，虽然部分说明将起作用，但是在说明表中将不会填充部分实际值信息。在这种情况下，会将条目写入 EXPLAIN_DIAGNOSTIC 表。
- 必须在重新创建现有 DB2 V9.7 活动事件监视器表（尤其是活动表）之后，才能由活动事件监视器捕获部分实际值数据。如果活动逻辑组不包含 SECTION_ACTUALS 列，可能仍然会使用活动事件监视器所捕获的部分来执行部分说明，但是该说明将不包含任何部分实际值数据。

访问部分实际值

可以使用 EXPLAIN_FROM_ACTIVITY 过程来访问部分实际值。当您对捕获了其部分实际值的活动执行部分说明时，将为 EXPLAIN_ACTUALS 说明表填充实际值信息。

注: 仅当使用 EXPLAIN_FROM_ACTIVITY 过程执行了部分说明时，部分实际值才可用。

EXPLAIN_ACTUALS 表是现有 EXPLAIN_OPERATOR 说明表的子表。调用 EXPLAIN_FROM_ACTIVITY 时，如果部分实际值可用，那么将在 EXPLAIN_ACTUALS 表中填充实际值数据。如果收集了多个数据库分区中的部分实际值，那么每个操作程序的每个数据库分区在 EXPLAIN_ACTUALS 表中都有一行。

获取部分说明和实际值以调查查询性能较低的原因:

要解决 SQL 查询性能下降问题，可从获取包含部分实际值信息的部分说明开始。然后，可以将部分实际值与优化器所生成的估计访问方案值进行比较，以评估此访问方案的有效性。此任务将指导您完成获取部分实际值以调查查询性能较低这一过程。

开始前

您已完成调查工作的诊断阶段，确定确实存在 SQL 查询性能下降问题，并且已经确定了哪个语句可能导致了性能下降。

关于此任务

此任务将指导您完成获取部分实际值以调查查询性能较低这一过程。与优化器所生成的估计值进行比较时，部分实际值中包含的信息可帮助您解决查询性能下降问题。

限制

请参阅“捕获和访问部分实际值”中说明的局限性。

过程

对于 myApp.exe 应用程序所执行的查询，要调查其查询性能较低的原因，请完成下列步骤:

1. 启用部分实际值:

   ```
   db2set DB2_SYSTEM_MONITOR_SETTINGS=SECTION_ACTUALS:TRUE
   ```

2. 使用 SYSINSTALLOBJECTS 过程在 MYSCHEMA 模式中创建说明表:

   ```
   CALL SYSINSTALLOBJECTS( 'EXPLAIN', 'C', NULL, 'MYSCHEMA' )
   ```

   注: 如果您已经创建了说明表，那么可以跳过此步骤。

3. 通过发出以下命令来创建工作负载 MYCOLLECTWL 以收集由 myApp.exe 应用程序提交的活动，并启用收集这些活动的部分数据:

   ```
   CREATE WORKLOAD MYCOLLECTWL APPLNAME( 'MYAPP.EXE')
    COLLECT ACTIVITY DATA WITH DETAILS,SECTION
    GRANT USAGE ON WORKLOAD MYCOLLECTWL TO PUBLIC
   ```

   注: 如果选择使用单独的工作负载，那么将限制活动事件监视器所捕获的信息量。

4. 通过发出以下语句来创建称为 ACTEVMON 的活动事件监视器:

   ```
   CREATE EVENT MONITOR ACTEVMON FOR ACTIVITIES WRITE TO TABLE
   ```

5. 通过执行以下语句来激活活动事件监视器 ACTEVMON:

   ```
   SET EVENT MONITOR ACTEVMON STATE 1
   ```

6. 运行 myApp.exe 应用程序。该活动事件监视器将捕获由此应用程序发出的所有语句。

7. 通过发出以下语句来查询活动事件监视器表，以查找感兴趣的语句的标识信息:

   ```
   SELECT APPL_ID,
          UOW_ID,
          ACTIIVTY_ID,
          STMT_TEXT
   FROM ACTIVITYSTMT_ACTEVMON
   ```

   以下是由所发出的 SELECT 语句生成的输出示例:

   ```
   APPL_ID                    UOW_ID    ACTIVITY_ID     STMT_TEXT
   -------------------------  --------  --------------  ---------------
   *N2.DB2INST1.0B5A12222841         1               1  SELECT * FROM ...
   ```

8. 将活动标识信息用作 EXPLAIN_FROM_ACTIVITY 过程的输入，以获取部分说明和实际值，如以下 CALL 语句所示:

   ```
   CALL EXPLAIN_FROM_ACTIVITY( '*N2.DB2INST1.0B5A12222841', 1, 1, 'ACTEVMON',
    'MYSCHEMA', ?, ?, ?, ?, ? )
   ```

   以下是 EXPLAIN_FROM_ACTIVITY 调用所生成的样本输出:

   ```
   Value of output parameters
   --------------------------
   Parameter Name : EXPLAIN_SCHEMA
   Parameter Value : MYSCHEMA

   Parameter Name : EXPLAIN_REQUESTER
   Parameter Value : SWALKTY

   Parameter Name : EXPLAIN_TIME
   Parameter Value : 2009-08-24-12.33.57.525703
   ```

```
Parameter Name : SOURCE_NAME
Parameter Value : SQLC2H20

Parameter Name : SOURCE_SCHEMA
Parameter Value : NULLID

Parameter Name : SOURCE_VERSION
Parameter Value :

Return Status = 0
```

9. 使用 db2exfmt 命令，并将作为 EXPLAIN_FROM_ACTIVITY 过程输出而返回的说明实例键指定为输入来调整说明数据的格式，如下所示:

```
db2exfmt -d test -w 2009-08-24-12.33.57.525703 -n SQLC2H20 -s NULLID -# 0 -t
```

以下是说明实例输出:

```
  ******************** EXPLAIN INSTANCE ********************

DB2_VERSION:       09.07.1
SOURCE_NAME:       SQLC2H20
SOURCE_SCHEMA:     NULLID
SOURCE_VERSION:
EXPLAIN_TIME:      2009-08-24-12.33.57.525703
EXPLAIN_REQUESTER: SWALKTY

Database Context:
----------------
        Parallelism:          None
        CPU Speed:            4.000000e-05
        Comm Speed:           0
        Buffer Pool size:     198224
        Sort Heap size:       1278
        Database Heap size:   2512
        Lock List size:       6200
        Maximum Lock List:    60
        Average Applications: 1
        Locks Available:      119040

Package Context:
---------------
        SQL Type:           Dynamic
        Optimization Level: 5
        Blocking:           Block All Cursors
        Isolation Level:    Cursor Stability



---------------- STATEMENT 1  SECTION 201 ----------------
        QUERYNO:       0
        QUERYTAG:      CLP
        Statement Type:   Select
        Updatable:        No
        Deletable:        No
        Query Degree:     1

Original Statement:
------------------
select *
from syscat.tables

Optimized Statement:
-------------------
SELECT Q10.$C67 AS "TABSCHEMA", Q10.$C66 AS "TABNAME", Q10.$C65 AS "OWNER",
        Q10.$C64 AS "OWNERTYPE", Q10.$C63 AS "TYPE", Q10.$C62 AS "STATUS",
        Q10.$C61 AS "BASE_TABSCHEMA", Q10.$C60 AS "BASE_TABNAME", Q10.$C59 AS
```

```
        "ROWTYPESCHEMA", Q10.$C58 AS "ROWTYPENAME", Q10.$C57 AS "CREATE_TIME",
        Q10.$C56 AS "ALTER_TIME", Q10.$C55 AS "INVALIDATE_TIME", Q10.$C54 AS
        "STATS_TIME", Q10.$C53 AS "COLCOUNT", Q10.$C52 AS "TABLEID", Q10.$C51
        AS "TBSPACEID", Q10.$C50 AS "CARD", Q10.$C49 AS "NPAGES", Q10.$C48 AS
        "FPAGES", Q10.$C47 AS "OVERFLOW", Q10.$C46 AS "TBSPACE", Q10.$C45 AS
        "INDEX_TBSPACE", Q10.$C44 AS "LONG_TBSPACE", Q10.$C43 AS "PARENTS",
        Q10.$C42 AS "CHILDREN", Q10.$C41 AS "SELFREFS", Q10.$C40 AS
        "KEYCOLUMNS", Q10.$C39 AS "KEYINDEXID", Q10.$C38 AS "KEYUNIQUE",
        Q10.$C37 AS "CHECKCOUNT", Q10.$C36 AS "DATACAPTURE", Q10.$C35 AS
        "CONST_CHECKED", Q10.$C34 AS "PMAP_ID", Q10.$C33 AS "PARTITION_MODE",
        '0' AS "LOG_ATTRIBUTE", Q10.$C32 AS "PCTFREE", Q10.$C31 AS
        "APPEND_MODE", Q10.$C30 AS "REFRESH", Q10.$C29 AS "REFRESH_TIME",
...

Explain level:    Explain from section

Access Plan:
-----------
        Total Cost:             154.035
        Query Degree:           1

                                                                  Rows
                                                                 Rows Actual
                                                                  RETURN
                                                                  (1)
                                                                   Cost
                                                                    I/O
                                                                    |
                                                                    54
                                                                    396
                                                                  >^HSJOIN
                                                                  (   2)
                                                                   153.056
                                                                     NA
                                                          /----------+----------\
                                                        54                        20
                                                        396                        0
                                                      >^HSJOIN                   TBSCAN
                                                      (   3)                     (  12)
                                                       140.872                   11.0302
                                                         NA                        NA
                                                  /---------+----------\           |
                                                54                       6         20
                                                396                      0         NA
                                              >^HSJOIN                 IXSCAN    TABLE: SYSIBM
                                              (   4)                   (  11)   SYSAUDITPOLICIES
                                               138.033                 2.01136
                                                 NA                      NA
                                           /---------+---------\           |
                                         54                      6         -1
                                         396                     0         NA
                                       >^HSJOIN                IXSCAN    INDEX: SYSIBM
                                       (   5)                  (  10)   INDCOLLATIONS04
                                        135.193                2.01136
                                          NA                     NA
                                    /-------+--------\             |
                                  54                    6          -1
                                  396                   0          NA
                                >^HSJOIN              IXSCAN    INDEX: SYSIBM
                                (   6)                (   9)   INDCOLLATIONS04
                                 132.354              2.01136
                                   NA                   NA
                             /-----+-----\              |
                           54              6           -1
                           396             0           NA
                         TBSCAN          IXSCAN    INDEX: SYSIBM
                         (   7)          (   8)   INDCOLLATIONS04
```

```
      129.57          2.01136
        NA              NA
        |               |
        54              -1
        NA              NA
  TABLE: SYSIBM   INDEX: SYSIBM
    SYSTABLES    INDCOLLATIONS04
```

...

10. 检查说明输出中的部分实际值信息。将部分实际值与优化器所生成访问方案的估计值进行比较。如果部分实际值与访问方案的估计值之间存在差异，那么请弄清楚导致此差异的原因并执行相应的操作。作为要讨论的一个示例，您将弄清楚要查询的其中一个表的表统计信息已过期的原因。此错误会导致优化器选择一个不正确的访问方案，这可能就会导致查询性能下降。在这种情况下，要采用的措施是对此表运行 RUNSTATS 命令以更新表统计信息。
11. 再次尝试运行该应用程序，以确定查询性能是否仍然较低。

分析说明输出中的部分实际值信息:

可用的部分实际值显示在说明输出的不同部分。本节描述了在何处查找说明输出中的部分实际值信息和运算符详细信息。

db2exfmt 图输出中的部分实际值

如果说明实际值可用，那么实际值将显示在图中的估计行。说明仅支持运算符的实际值，而不支持对象的实际值。对于图中的对象将显示 NA（不适用）。以下是示例 db2exfmt 图输出:

```
       Rows
     Rows Actual
       RETURN
       (1)
        Cost
         I/O
         |
       3.21948  << The estimated rows used by optimizer
         301    << The actuals rows collected in runtime
        DTQ
       (   2)
       75.3961
         NA
         |
       3.21948
         130
       HSJOIN
       (   3)
       72.5927
         NA
      /--+---\
    674        260
    220        130
  TBSCAN     TBSCAN
  (   4)     (   5)
  40.7052    26.447
    NA         NA
    |          |
    337        130
    NA         NA  << Explain does not support actuals for objects
  TABLE: FF  TABLE: FF
    T1         T2
```

在分区数据库环境中，图中所显示的基数是在其中收集实际值的所有数据库分区的平均基数。显示的是平均值，因为这是优化器所估计的值。提供实际平均值比提供估计平均值更有意义。在分区数据库环境中，在运算符详细信息输出中按数据库分区提供了对于部分实际值的统计分析。用户可以检查这些详细信息以确定其他信息，例如，（所有分区的）总计、最小值和最大值等等。

db2exfmt 输出中的运算符详细信息

运算符的实际基数将显示在包含 Estimated number of rows 的那一行后面的流部分（说明输出中的 Actual number of rows）。如果是在对多个数据库成员运行运算符，那么所显示的实际基数将是分区数据库环境的平均基数。按数据库分区列示的值将显示在一个单独的部分 Explain Actuals。只有在分区数据库环境中才会显示 Explain Actuals 部分，但不是按串行方式显示。如果没有为特定数据库分区提供实际值，那么在按数据库分区列示的值列表中，在分区号旁边将显示 NA。Output Streams 部分中的 Actual number of rows 也将为 NA。下面是 db2exfmt 输出中的运算符详细信息的示例:

```
9) UNION : (Union)
   Cumulative Total Cost:     10.6858
   Cumulative First Row Cost:    9.6526

   Arguments:
   ---------
   UNIONALL: (UnionAll Parameterized Base Table)
      DISJOINT

   Input Streams:
   -------------
      5) From Operator #10

         Estimated number of rows:  30
         Actual number of rows:     63
         Partition Map ID:          3

      7) From Operator #11

         Estimated number of rows:  16
         Actual number of rows:     99
         Partition Map ID:          3


   Output Streams:
   --------------
      8) To Operator #8

         Estimated number of rows:  30
         Actual number of rows:     162
         Partition Map ID:          3

   Explain Actuals:      << This section will only show in a partitioned database environment
   ----------------
      DB Partition number   Cardinality
      -------------------   -----------
               1               193
               2               131
```

有关使用说明信息的准则

您可以使用说明信息来了解应用程序性能变化的原因或者评估性能调整效果。

性能变化分析

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:

1. 在进行任何更改前，捕获查询的说明信息并保存所生成的说明表。此外，也可以保存 db2exfmt 说明工具的输出。
2. 如果您无法通过访问 Visual Explain 来查看此信息，那么请保存或打印当前的目录统计信息。可以使用 db2look 生产力工具来帮助执行此任务。
3. 保存或打印数据定义语言（DDL）语句，其中包括 CREATE TABLE、CREATE VIEW、CREATE INDEX 或 CREATE TABLESPACE。

以此方式收集的信息为将来的分析提供参考点。对于动态 SQL 或 XQuery 语句，可以在首次运行应用程序时收集此信息。对于静态 SQL 和 XQuery 语句，可以在绑定时收集此信息。要进行性能变化分析，请将您收集的信息与先前收集的此参考信息作比较。

例如，您所作的分析可能表明，确定访问路径时不再使用某个索引。通过使用 Visual Explain 中的目录统计信息，您可能注意到，索引层数（NLEVELS 列）现在大大高于最初将查询与数据库绑定时的情况。于是，您可以选择执行下列其中一项操作:

- 重组该索引
- 收集表和索引的新统计信息
- 在重新绑定查询时收集说明信息

执行其中一项操作之后，再次检查访问方案。如果再次使用该索引，那么查询性能可能不再有问题。如果仍未使用该索引，或者性能仍有问题，那么请选择此列表中的另一项操作并检查结果。重复这些步骤，直到解决问题为止。

评估性能调整效果

您可以执行多项操作来帮助提高查询性能，例如更新配置参数、添加容器以及收集新的目录统计信息等等。

在其中任何一个领域进行更改后，请使用说明工具来确定该更改对所选访问方案的影响。例如，如果您根据索引准则来添加索引或具体化查询表（MQT），那么说明数据可以帮助您确定是否已实际地按预期方式使用该索引或 MQT。

尽管说明输出使您能够确定所选访问方案及其相关成本，但准确测量特定查询的性能提高情况的唯一方法是使用基准程序测试技术。

有关分析说明信息的准则

说明信息的主要用途是分析查询语句的访问路径。您可以通过多种方法来分析说明数据，以帮助调整查询和环境。

请考虑下列几种分析:

- 索引使用

  适当的索引可以显著提高性能。通过使用说明输出，您可以确定是否正在使用为了帮助进行一组特定的查询而创建的索引。请在下列区域查找索引的使用:

  – 连接谓词

- 局部谓词
- GROUP BY 子句
- ORDER BY 子句
- WHERE XMLEXISTS 子句
- 选择列表

也可以使用说明工具来评估使用另一个索引或者完全不使用索引是否更好。创建新索引后，请使用 RUNSTATS 命令来收集该索引的统计信息，然后重新编译查询。随着时间的推移，您可能会通过说明数据注意到，系统使用了表扫描方法而非索引扫描方法。这可能是由于表数据的集群发生更改所致。如果先前使用的索引现在具有较低的集群比率，那么您可能想执行下列操作:

- 重组该表，以便根据该索引对表数据进行集群
- 使用 RUNSTATS 命令来收集索引和表的统计信息
- 重新编译查询

要确定重组表是否已改进访问方案，请检查重新编译后的查询的说明输出。

- 访问类型

分析说明输出，并查找对于正在运行的应用程序类型而言通常并非最优的数据访问类型。例如:

- 联机事务处理（OLTP）查询

  OLTP 应用程序是使用范围定界谓词进行索引扫描的主要候选者，这是因为它们倾向于通过对键列应用等式谓词达到只返回少数合格行的目标。如果您的 OLTP 查询使用表扫描方法，那么您可能想分析说明数据以确定不使用索引扫描方法的原因。

- 仅浏览查询

  “浏览”类型的查询的搜索条件可能很含糊，从而导致产生大量合格的行。如果用户通常只查看少数输出数据屏幕，那么您可以指定在返回一些结果前，不需要计算整个答案集。在这种情况下，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（而不只是前几个数据屏幕）的资源耗用量减少到最低程度。

  例如，如果说明输出表明访问方案使用了合并扫描连接和排序运算符，那么在将任何行返回给应用程序之前，将在一个临时表中实现整个答案集。在这种情况下，您可以尝试在 SELECT 语句中使用 OPTIMIZE FOR 子句来更改访问方案。如果指定此选项，那么优化器可尝试选择一个访问方案，该方案在将前面几行返回给应用程序前，不在临时表中生成整个答案集。

- 连接方法

如果一个查询连接了两个表，那么检查所使用的连接的类型。涉及行数较多的连接，例如决策支持查询中的那些连接，使用散列连接或合并连接通常会运行得更快。只涉及几行的连接，例如 OLTP 查询中的那些连接，通常使用嵌套循环连接会运行得更快。但是，在这两种情况下都可能会出现运行减慢问题（例如使用局部谓词或索引），这可能会更改这些典型连接的工作方式。

使用访问方案对 REFRESH TABLE 和 SET INTEGRITY 语句的性能问题进行自诊断

通过对 REFRESH TABLE 和 SET INTEGRITY 语句调用说明实用程序，可以生成可用来对这些语句的性能问题进行自诊断的访问方案。这可以帮助您更好地维护具体化查询表（MQT）。

要获取 REFRESH TABLE 或 SET INTEGRITY 语句的访问方案，请使用下列任何一种方法:

- 在 EXPLAIN 语句中使用 EXPLAIN PLAN FOR REFRESH TABLE 或 EXPLAIN PLAN FOR SET INTEGRITY 选项
- 在发出 REFRESH TABLE 或 SET INTEGRITY 语句前将 CURRENT EXPLAIN MODE 专用寄存器设置为 EXPLAIN，然后将 CURRENT EXPLAIN MODE 专用寄存器设置为 NO。

限制

- REFRESH TABLE 和 SET INTEGRITY 语句不适合进行重新优化；因此，REOPT 说明方式（或说明快照）不适用于这两个语句。
- EXPLAIN 语句的 WITH REOPT ONCE 子句也指示将重新优化指定的可说明语句，它不适用于 REFRESH TABLE 和 SET INTEGRITY 语句。

方案

此方案说明如何从 EXPLAIN 和 REFRESH TABLE 语句生成访问方案并使用该方案来自诊断性能问题的原因。

1. 创建并填充表。例如:

```
create table t (
  i1 int not null,
  i2 int not null,
  primary key (i1)
);

insert into t values (1,1), (2,1), (3,2), (4,2);

create table mqt as (
  select i2, count(*) as cnt from t group by i2
)
data initially deferred
refresh deferred;
```

2. 发出 EXPLAIN 和 REFRESH TABLE 语句，如下所示:

```
explain plan for refresh table mqt;
```

   此步骤可替换为在 SET CURRENT EXPLAIN MODE 专用寄存器中设置 EXPLAIN 方式，如下所示:

```
set current explain mode explain;
refresh table mqt;
set current explain mode no;
```

3. 使用 db2exfmt 命令来格式化说明表的内容并获取访问方案。此工具位于实例 sqllib 目录的 misc 子目录中。

```
db2exfmt –d dbname -o refresh.exp -1
```

4. 分析访问方案以确定性能问题的原因。在以上示例中，如果 T 是大型表，那么表扫描成本将相当高昂。创建索引可以提高查询的性能。

用于收集和分析说明信息的工具

DB2 数据库服务器提供了一个综合全面的说明工具，此工具可提供有关优化器为 SQL 或 XQuery 语句选择的访问方案的详细信息。

存储说明数据的表在所有受支持的平台上均可访问，并包含静态和动态 SQL 和 XQuery 语句的信息。多个工具使您能够灵活地捕获、显示和分析说明信息。

详细的查询优化器信息使您能够对访问方案进行深入分析，此信息存储在说明表中，独立于实际的访问方案本身。要从说明表中获取信息，请使用下列其中一种或多种方法:

- 使用 db2exfmt 工具在格式化的输出中显示说明信息。
- 编写您自己的针对说明表的查询。通过编写自己的查询，您可以轻松方便地处理输出、比较不同的查询或者对一段时间内同一查询的多次执行进行比较。

使用 db2expln 工具来查看静态 SQL 或 XQuery 语句的一个或多个程序包的访问方案信息。此实用程序显示所选访问方案的实际实现，而未显示优化器信息。通过检查所生成的访问方案，db2expln 工具提供了在运行时将要执行的操作的相对紧凑、非正式视图。

您可以在 `sqllib` 目录的 `misc` 子目录中找到这些命令行说明工具。

下表对可以与 DB2 说明工具配合使用的各种工具作了概述。请参考此表来选择最适合于您的环境和需求的工具。

表 53. 说明工具

期望的特征	说明表	db2expln	db2exfmt
文本输出		是	是
“快速和脏的”静态 SQL 和 XQuery 分析		是	
静态 SQL 和 XQuery 支持	是	是	是
动态 SQL 和 XQuery 支持	是	是	是
CLI 应用程序支持	是		是
可用于 DRDA® 应用程序请求器	是		
详细的优化器信息	是		是
适合于分析多个语句	是	是	是
可以从应用程序中访问信息	是		

显示在说明时有效的目录统计信息

说明工具捕获在说明语句时有效的统计信息。这些统计信息可能与系统目录中存储的统计信息不同，实时统计信息收集功能处于启用状态时尤其如此。如果已填充说明表，但未创建说明快照，那么只有部分统计信息记录在 EXPLAIN_OBJECT 表中。

要捕获所有与正在说明的语句相关的目录统计信息，请在填充说明表的同时创建说明快照，然后使用 SYSPROC.EXPLAIN_FORMAT_STATS 标量函数来格式化该快照中的目录统计信息。

如果使用 db2exfmt 工具来格式化说明信息，并且已收集说明快照，那么此工具将自动使用 SYSPROC.EXPLAIN_FORMAT_STATS 函数来显示目录统计信息。

说明信息的说明表和组织

所有说明信息都围绕“说明实例”这一概念进行组织。*说明实例*代表说明工具对一个或多个 SQL 或 XQuery 语句的一次调用。在一个说明实例中捕获的说明信息包括编译环境以及为满足正在编译的 SQL 或 XQuery 语句而选择的访问方案。

例如，说明实例可以由下列任何一项组成:

- 对于静态查询语句: 一个程序包中所有合格的 SQL 或 XQuery 语句。对于 SQL 语句（包括用于查询 XML 数据的语句）而言，可以捕获 CALL、复合 SQL（动态）、DELETE、INSERT、MERGE、REFRESH TABLE、SELECT、SET INTEGRITY、SELECT INTO、UPDATE、VALUES 和 VALUES INTO 语句的说明信息。对于 XQuery 语句而言，可以获取 XQUERY db2-fn:xmlcolumn 和 XQUERY db2-fn:sqlquery 语句的说明信息。

  注: REFRESH TABLE 和 SET INTEGRITY 语句只以动态方式进行编译。
- 对于增量绑定 SQL 语句: 一个特定的 SQL 语句
- 对于动态 SQL 语句: 一个特定的 SQL 语句
- 每个 EXPLAIN 语句（动态或静态）

（通过发出 EXPLAIN 语句或者通过使用部分说明接口调用的）说明工具将捕获有关为特定可说明语句选择的访问方案的信息并将此信息写入说明表。在发出 EXPLAIN 之前，您必须创建说明表。要创建这些表，请运行 `sqllib` 子目录的 `misc` 子目录中的 `EXPLAIN.DDL` 脚本。

您还可以使用 SYSPROC.SYSINSTALLOBJECTS 过程来创建、删除和验证说明表。可以在特定模式和表空间中创建这些表。您可以在 `EXPLAIN.DDL` 文件中找到示例。

说明表可能由多个用户共用。可以为一个用户定义说明表，然后为每个附加用户创建指向已定义的表的别名。此外，也可以在 SYSTOOLS 模式下定义说明表。如果在用户会话标识下（对于动态 SQL 或 XQuery 语句）或语句授权标识下（对于静态 SQL 或 XQuery 语句）找不到其他说明表或别名，那么说明工具将使用缺省的 SYSTOOLS 模式。每个共享公共说明表的用户都必须对那些表具有插入特权。

表 54. 说明表摘要

表名	描述
ADVISE_INDEX	存储有关建议的索引的信息。此表可以由查询编译器、db2advis 命令或用户填充。此表用于: • 获取建议的索引 • 根据有关建议的索引的输入对索引进行求值
ADVISE_INSTANCE	包含有关 db2advis 执行情况的信息，其中包括开始时间。每次执行 db2advis 都会生成相应的一行。
ADVISE_MQT	包含用于定义每个所建议的具体化查询表（MQT）的查询、每个 MQT 的列统计信息（例如 COLSTATS（采用 XML 格式））以及 NUMROWS 等等，此外，还包含用于获取每个 MQT 的详细统计信息的采样查询。
ADVISE_PARTITION	存储由 db2advis 生成并评估的虚拟数据库分区。

表 54. 说明表摘要 (续)

表名	描述
ADVISE_TABLE	通过使用设计顾问程序提供的有关 MQT、多维集群表（MDC）和数据库分区的最终建议，存储用于创建表的数据定义语言（DDL）。
ADVISE_WORKLOAD	此表中的每一行代表工作负载中的一个 SQL 或 XQuery 语句。db2advis 命令使用此表来收集并存储工作负载信息。
EXPLAIN_ACTUALS	包含说明部分实际值信息。
EXPLAIN_ARGUMENT	包含有关每个运算符（如果存在的话）的唯一特征的信息。
EXPLAIN_DIAGNOSTIC	对于为 EXPLAIN_STATEMENT 表中的所说明语句的特定实例生成的每条诊断消息，此表包含一个条目。
EXPLAIN_DIAGNOSTIC_DATA	包含 EXPLAIN_DIAGNOSTIC 表中记录的特定诊断消息的消息标记。这些消息标记提供特定于生成该消息的 SQL 语句的执行的更多信息。
EXPLAIN_INSTANCE	所有说明信息的主控制表。说明表中的每一行都显式地链接至此表中唯一的一行。有关正在说明的 SQL 或 XQuery 语句源代码的基本信息以及环境信息保存在此表中。
EXPLAIN_OBJECT	标识为了满足 SQL 或 XQuery 语句而生成的访问方案所需的数据对象。
EXPLAIN_OPERATOR	包含查询编译器为了满足 SQL 或 XQuery 语句而需要的所有运算符。
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 或 XQuery 语句的文本。用户输入的原始 SQL 或 XQuery 语句与优化器用于选择访问方案的版本一起存储在此表中。 当请求获取说明快照时，将记录其他说明信息以描述查询优化器选择的访问方案。此信息采用 Visual Explain 所需的格式存储在 EXPLAIN_STATEMENT 表的 SNAPSHOT 列中。其他应用程序无法使用此格式。
EXPLAIN_STREAM	表示各个运算符与数据对象之间的输入和输出数据流。数据对象本身在 EXPLAIN_OBJECT 表进行表示。数据流中涉及的运算符在 EXPLAIN_OPERATOR 表中进行表示。

数据对象的说明信息:

单个访问方案可以使用一个或多个数据对象来满足 SQL 或 XQuery 语句。

对象统计信息

说明工具将记录有关每个对象的信息，例如:

- 创建时间

- 上次收集该对象的统计信息的时间
- 该对象中的数据是否已排序（仅限于表或索引对象）
- 该对象中的列数（仅限于表或索引对象）
- 该对象中的估计行数（仅限于表或索引对象）
- 该对象在缓冲池中占用的页数
- 对于该对象存储所在的指定表空间，所发生的单一随机 I/O 的估计总开销（以毫秒计）
- 从指定的表空间读取 4-KB 页的估计传输速率（以毫秒计）
- 预取大小和扩展数据块大小（以 4-KB 页计）
- 索引中的数据集群程度
- 该对象的索引使用的叶子页数以及树中的层数
- 该对象的索引中相异完整键值的数目
- 表中溢出记录的总数

数据运算符的说明信息:

单个访问方案可以对数据执行多项操作，从而满足 SQL 或 XQuery 语句并返回结果。查询编译器确定必需的操作，例如表扫描、索引扫描、嵌套循环连接或者 GROUP BY 运算符。

除显示关于访问方案中使用的每个运算符的信息以外，说明输出还显示访问方案的累积效果。

估算的成本信息

将记录运算符的下列累积成本估算值。这些成本是所选访问方案的成本，包括被捕获信息的运算符的成本。

- 总成本（以 timeron 计）
- 页 I/O 次数
- 处理指令数
- 访存第一行的成本（以 timeron 计），其中包括任何必需的初始开销
- 通信成本（以帧计）

timeron 是发明的相对计量单位。Timeron 值由优化器根据数据库被使用时将会更改的内部值（例如统计信息）确定。因此，不能保证每次确定以 timeron 计的估算成本时 SQL 或 XQuery 语句的 timeron 值都相同。

运算符属性

说明工具将记录下列用于描述每个运算符的属性的信息:

- 已访问的表的集合
- 已访问的列的集合
- 数据排序所依据的列（如果优化器已确定后续运算符可使用此排序的话）
- 已应用的谓词的集合
- 将返回的估计行数（基数）

实例的说明信息:

说明实例信息存储在 EXPLAIN_INSTANCE 表中。有关实例中的每个查询语句的其他特定信息存储在 EXPLAIN_STATEMENT 表中。

说明实例标识

下列信息帮助您标识特定的说明实例以及使关于特定语句的信息与说明工具的特定调用相关联:

- 请求获取说明信息的用户
- 说明请求的开始时间
- 包含所说明语句的程序包的名称
- 包含所说明语句的程序包的 SQL 模式
- 包含该语句的程序包的版本
- 是否已收集快照信息

环境设置

关于查询编译器优化查询时所处的数据库管理器环境的信息也将被捕获。环境信息包括下列各项:

- DB2 产品的版本和发行版号
- 编译查询时所处的并行度

  CURRENT DEGREE 专用寄存器、DEGREE 绑定选项、SET RUNTIME DEGREE 命令和 **dft_degree** 数据库配置参数确定编译特定查询时所处的并行度。
- 该语句是动态的还是静态的
- 用于编译该查询的查询优化类
- 编译查询时发生的游标行分块的类型
- 该查询运行时所处的隔离级别
- 编译该查询时各种配置参数的值。获取说明快照时，将记录下列参数的值:
  - 排序堆大小（**sortheap**）
  - 活动应用程序的平均数目（**avg_appls**）
  - 数据库堆（**dbheap**）
  - 锁定列表的最大存储量（**locklist**）
  - 升级之前锁定列表的最大百分比（**maxlocks**）
  - CPU 速度（**cpuspeed**）
  - 通信带宽（**comm_bandwidth**）

语句标识

每个说明实例可能已说明多个语句。除了用于唯一地标识说明实例的信息以外，下列信息可以帮助标识各个查询语句:

- 语句的类型: SELECT、DELETE、INSERT、UPDATE、定位型 DELETE、定位型 UPDATE 或 SET INTEGRITY
- 程序包中发出该语句的语句号和节号，它们记录在 SYSCAT.STATEMENTS 目录视图中

EXPLAIN_STATEMENT 表中的 QUERYTAG 和 QUERYNO 字段包含说明过程中设置的标识。当 EXPLAIN MODE 或 EXPLAIN SNAPSHOT 处于活动状态，并且在命令行处理器（CLP）或调用级别接口（CLI）会话期间提交动态说明语句时，QUERYTAG 值将分别设置为“CLP”或“CLI”。在这种情况下，对于每个语句，QUERYNO 将缺省为按 1 或更大的值递增的数值。对于所有其他并非来自 CLP 或 CLI 或者未使用 EXPLAIN 语句的动态说明语句，QUERYTAG 值将设置为空白，并且 QUERYNO 始终为 1。

成本估算

对于所说明的每个语句，优化器将记录执行所选访问方案的相对成本估算值。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，不提供耗用时间估算值:

- 查询优化器不估算耗用时间，而只估算资源消耗量。
- 优化器并非将可影响耗用时间的所有因素都考虑在模型内。它将忽略不影响访问方案效率的那些因素。许多运行时因素会影响耗用时间，其中包括：系统工作负载、资源争用量、并行处理量和 I/O 量、将行返回至用户的成本以及客户机与服务器之间的通信时间。

语句文本

对于所说明的每个语句，将记录语句文本的两个版本。一个版本是查询编译器从应用程序接收的代码。另一个版本是对该查询的内部（编译器）表示进行反向转换而生成的版本。尽管此转换类似于其他查询语句，但它既不必遵循正确的查询语言语法，也不必在整体上反映内部表示的实际内容。此转换只是为了使您能够了解优化器选择访问方案时所处的上下文。要了解编译器如何重写查询以便更好地进行优化，请将用户编写的语句文本与查询语句的内部表示作比较。重写的语句还将揭示其他影响语句的因素，例如触发器和约束。在这个“经过优化”的文本中，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。

$CONSTRAINT$
此标记用于标识编译期间对原始语句添加的约束，您应该结合 $WITH_CONTEXT$ 前缀来查看此标记的内容。

$DERIVED.T*n*
派生的表的名称，其中 *n* 表示整数值。

$INTERNAL_FUNC$
此标记指示存在编译器用于所说明查询但不可通用的函数。

$INTERNAL_PRED$
此标记指示存在编译所说明查询期间添加的但不可通用的谓词。编译器使用内部谓词来满足由于触发器和约束而对原始语句添加的附加上下文。

$INTERNAL_XPATH$
此标记指示存在内部表函数，该函数将单个带注释的 XPath 模式作为输入参数，并返回有一列或多列与该模式匹配的表。

RID 此标记用于标识特定行的行标识（RID）列。

$TRIGGER$
此标记用于标识编译期间对原始语句添加的触发器，您应该结合 $WITH_CONTEXT$ 前缀来查看此标记的内容。

$WITH_CONTEXT$(...)

如果已对原始查询语句添加附加的触发器或约束，那么此前缀将出现在文本开始位置。在此前缀之后，将出现任何影响该语句的编译和解析的触发器或约束的名称列表。

SQL 和 XQuery 说明工具

db2expln 命令描述为 SQL 或 XQuery 语句选择的访问方案。

当未捕获到说明数据时，您可以使用此工具来获取所选访问方案的快速说明。对于静态 SQL 和 XQuery 语句，db2expln 将检查存储在系统目录中的程序包。对于动态 SQL 和 XQuery 语句，db2expln 将检查查询高速缓存中的各个部分。

说明工具位于实例 sqllib 目录的 bin 子目录中。如果 db2expln 不在当前目录中，那么它必须在 PATH 环境变量中出现的某个目录中。

首次访问数据库时，db2expln 命令将使用 db2expln.bnd、db2exsrv.bnd 和 db2exdyn.bnd 文件将自己与该数据库绑定。

db2expln 输出的描述:

db2expln 命令的说明输出包括程序包信息以及每个程序包的部分信息。

- 程序包信息包括绑定操作的日期以及相关的绑定选项
- 部分信息包括部分号以及正在说明的 SQL 或 XQuery 语句

关于该 SQL 或 XQuery 语句的所选访问方案的说明输出将出现在部分信息后面。

访问方案的步骤（即“部分”）按数据库管理器执行它们的顺序显示。每个主要步骤都显示为向左对齐的标题，有关该步骤的信息以缩进形式显示在该标题之下。在访问方案的说明输出的左页边距中，将出现缩进条。这些条还用于标记每项操作的作用域。缩进层较低（更靠近右边）的操作将在上一缩进层中的操作之前被处理。

选择的访问方案基于输出中显示的原始 SQL 语句（如果已启用语句集中器，那么是*有效的 SQL 语句*）或 XQuery 语句的扩充版本。由于查询编译器的查询重写组件可能会将 SQL 或 XQuery 语句转换为等效但效率更高的格式，因此，说明输出中显示的访问方案可能与您的期望出入很大。说明工具（包括说明表、SET CURRENT EXPLAIN MODE 语句和 Visual Explain）显示的是用于优化的实际 SQL 或 XQuery 语句，这些语句具有 SQL 或 XQuery 的风格，并且是通过对查询的内部表示进行反向转换创建的。

在将 db2expln 的输出与说明工具的输出进行比较时，运算符标识选项（-opids）非常有用。每当 db2expln 开始处理来自说明工具的新运算符时，就会在所说明方案的左边打印运算符标识号。运算符标识可以用于对该访问方案的各种表示中的步骤进行比较。注意，在说明工具输出中的运算符与 db2expln 显示的操作之间并非始终存在一一对应的关系。

表访问信息:

db2expln 输出中的一个语句将提供正在访问的表的名称和类型。

关于常规表的信息包括下列其中一个表访问语句:

```
Access Table Name = schema.name  ID = ts,n
Access Hierarchy Table Name = schema.name  ID = ts,n
Access Materialized Query Table Name = schema.name  ID = ts,n
```

其中:

- *schema.name* 是正在访问的表的标准名称
- ID 是该表的 SYSCAT.TABLES 目录视图条目中的相应 TABLESPACEID 和 TABLEID

关于临时表的信息包括下列其中一个表访问语句:

```
Access Temp Table  ID = tn
Access Global Temp Table  ID = ts,tn
```

其中，ID 是该表（*ts*）的 SYSCAT.TABLES 目录视图条目中的相应 TABLESPACEID 或者由 db2expln 指定的相应标识（*tn*）。

在表访问语句之后，将提供下列附加语句以便进一步描述该访问。

- 列数
- 块访问
- 并行扫描
- 扫描方向
- 行访问
- 锁定意图
- 谓词
- 其他

列数语句

以下语句指示在表的每一行中使用的列数:

```
#Columns = n
```

块访问语句

以下语句指示已对表定义一个或多个维块索引:

```
Clustered by Dimension for Block Index Access
```

如果未出现此语句，那么表明创建该表时未指定 ORGANIZE BY DIMENSIONS 子句。

并行扫描语句

以下语句指示数据库管理器将使用多个子代理程序以并行方式读取该表:

```
Parallel Scan
```

如果未出现此语句，那么表明该表将只被一个代理程序（或子代理程序）读取。

扫描方向语句

以下语句指示数据库管理器将按逆序读取行:

```
Scan Direction = Reverse
```

如果未出现此语句，那么表明扫描方向是正向，这是缺省情况。

行访问语句

下列其中一个语句将指示如何访问表中的合格行。

- Relation Scan 语句指示正在按顺序扫描表以查找合格行。
  - 以下语句指示不执行数据预取:

    ```
    Relation Scan
    |  Prefetch: None
    ```

  - 以下语句指示优化器已确定将要预取的页数:

    ```
    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来标识和访问合格行:

  ```
  Index Scan:  Name = schema.name  ID = xx
  | Index type
  | Index Columns:
  ```

  其中:

  - *schema.name* 是正在扫描的索引的标准名称
  - ID 是 SYSCAT.INDEXES 目录视图中的相应 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
    Index over XML data
    ```

  接着，对于索引中的每一列，都有一行输出。此信息的有效格式如下:

  ```
  n: column_name (Ascending)
  n: column_name (Descending)
  n: column_name (Include Column)
  ```

  将提供下列语句以指示索引扫描类型。

  - 通过下列语句指示索引的范围定界谓词:

    ```
    #Key Columns = n
    |  Start Key: xxxxx
    |  Stop Key: xxxxx
    ```

    其中，*xxxxx* 是下列其中一项:

    - Start of Index
    - End of Index
    - Inclusive Value: 或 Exclusive Value:

      包括键值将会包括在索引扫描中。排除键值将不会包括在扫描中。键值由该键的每一部分的下列其中一项确定:

      ```
      n: 'string'
      n: nnn
      n: yyyy-mm-dd
      ```

```
n: hh:mm:ss
n: yyyy-mm-dd hh:mm:ss.uuuuuu
n: NULL
n: ?
```

将只显示文字串的前 20 个字符。如果字符串长度超过 20 个字符，那么这种情况由该文字串末尾的省略号（...）指示。某些键只有在该部分执行之后才能确定。这种情况由作为值的问号（?）指示。

– `Index-Only Access`

如果能够从索引键中获得所有需要的列，那么将显示此语句，并且不会访问任何表数据。

– 以下语句指示不会对索引页进行预取:

```
Index Prefetch: None
```

– 以下语句指示应该预取索引页:

```
Index Prefetch: Eligible
```

– 以下语句指示不执行数据页预取:

```
Data Prefetch: None
```

– 以下语句指示应该预取数据页:

```
Data Prefetch: Eligible
```

– 如果存在可以传递至索引管理器以帮助限定索引条目的谓词，那么将使用以下语句来显示这些谓词的数目:

```
Sargable Index Predicate(s)
|  #Predicates = n
```

- 如果正在通过访问方案中先前准备的行标识（RID）来访问合格行，那么这种情况将由以下语句指示:

```
Fetch Direct Using Row IDs
```

如果已对该表定义一个或多个块索引，那么可以通过块标识或行标识来访问行。这种情况由以下语句指示:

```
Fetch Direct Using Block or Row IOs
```

锁定意图语句

对于每个表访问，都将通过以下语句显示将在表和行级别获取的锁定的类型:

```
Lock Intents
|  Table: xxxx
|  Row  : xxxx
```

可能的表锁定值包括:

- Exclusive
- Intent Exclusive
- Intent None
- Intent Share
- Share
- Share Intent Exclusive
- Super Exclusive

- Update

可能的行锁定值包括:

- Exclusive
- Next Key Weak Exclusive
- None
- Share
- Update

谓词语句

共有三种类型的语句提供有关访问方案中使用的谓词的信息。

- 以下语句指示对于从分块索引中检索的每个数据块，将要进行求值的谓词的数目:

  ```
  Block Predicates(s)
  |  #Predicates = n
  ```

- 以下语句指示访问数据时，将要进行求值的谓词的数目。此数目不包括下推操作，例如聚集或排序:

  ```
  Sargable Predicate(s)
  |  #Predicates = n
  ```

- 以下语句指示返回数据后，将要进行求值的谓词的数目:

  ```
  Residual Predicate(s)
  |  #Predicates = n
  ```

这些语句中显示的谓词数可能未反映查询语句中提供的谓词数，这是因为:

- 在同一个查询中可以多次应用同一个谓词
- 在查询优化过程中，可以通过添加隐式谓词来变换和扩展谓词
- 在查询优化过程中，可以对谓词进行变换和压缩以减少谓词数目

其他表语句

- 以下语句指示将只访问一行:

  ```
  Single Record
  ```

- 如果用于进行表访问的隔离级别与语句的隔离级别不同，那么将出现以下语句:

  ```
  Isolation Level: xxxx
  ```

  对于这种情况，有多种可能的原因。例如:

  – 使用“可重复读”（RR）隔离级别绑定的程序包影响到某些引用完整性约束；为了检查这些约束而对父表进行的访问将会降级到“游标稳定性”（CS）隔离级别，以避免对此表挂起不必要的锁定。
  – 使用“未落实的读”（UR）隔离级别绑定的程序包包含 DELETE 语句；为了执行删除操作而对表进行的访问将会升级到 CS。

- 以下语句指示，如果有足够的 **sortheap** 内存可用，那么将在缓冲池外部对从临时表读取的部分或全部行进行高速缓存:

  ```
  Keep Rows In Private Memory
  ```

- 以下语句指示已对表设置易变基数属性:

  ```
  Volatile Cardinality
  ```

临时表信息:

在访问方案执行期间，将使用一个临时表作为工作表。通常，如果需要在访问方案中提前对子查询进行求值，或者中间结果在可用内存中装不下，那么将使用临时表。

如果需要临时表，那么 db2expln 命令输出将包含下列其中一个语句。

```
   Insert Into Temp Table  ID = tn     --> 普通的临时表
   Insert Into Shared Temp Table  ID = tn     --> 多个子代理程序将以并行方式创建
                                                  普通的临时表
Insert Into Sorted Temp Table  ID = tn     --> 已排序的临时表
   Insert Into Sorted Shared Temp Table  ID = tn     --> 多个子代理程序将以并行方式创建
                                                        已排序的临时表
Insert Into Global Temp Table  ID = ts,tn     --> 已声明的全局临时表
   Insert Into Shared Global Temp Table  ID = ts,tn     --> 多个子代理程序将以并行方式创建
                                                           已声明的全局临时表
Insert Into Sorted Global Temp Table  ID = ts,tn     --> 已排序的已声明全局临时表
   Insert Into Sorted Shared Global Temp Table  ID = ts,tn     --> 多个子代理程序将以并行方式创建
                                                                  已排序的已声明全局临时表
```

ID 是由 db2expln 指定的标识，旨在便于引用临时表。此 ID 以字母“t”为前缀，以指示该表是临时表。

上述每个语句后面都跟着以下语句:

```
#Columns = n
```

此语句指示要插入到临时表的每一行所包含的列数。

已排序的临时表

已排序的临时表可以由类似如下的操作生成:

- ORDER BY
- DISTINCT
- GROUP BY
- 合并连接
- “= ANY”子查询
- “<> ALL”子查询
- INTERSECT 或 EXCEPT
- UNION（不带 ALL 关键字）

在 db2expln 命令输出中，可能会出现许多与已排序的临时表相关联的语句。

- 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，将显示下列其中一行:

  ```
  Key n: column_name (Ascending)
  Key n: column_name (Descending)
  Key n: (Ascending)
  Key n: (Descending)
  ```

- 下列语句提供对行数和行大小的估算，以便在运行时可以分配最优的排序堆:

  ```
  Sortheap Allocation Parameters:
  |  #Rows     = n
  |  Row Width = n
  ```

- 如果只需要已排序的结果的前几行，那么将显示以下语句:

  ```
  Sort Limited To Estimated Row Count
  ```

- 对于对称多处理器（SMP）环境中执行的排序，要执行的排序的类型由下列其中一个语句指示:

```
Use Partitioned Sort
Use Shared Sort
Use Replicated Sort
Use Round-Robin Sort
```

- 下列语句指示已排序的结果是否留在排序堆中:

```
Piped
Not Piped
```

  如果指示管道式排序，那么数据库管理器会将已排序的输出保留在内存中，而不是将其放入另一个临时表。

- 以下语句指示排序操作期间将除去重复的值:

```
Duplicate Elimination
```

- 如果排序操作期间执行了聚集，那么将显示下列其中一个语句:

```
Partial Aggregation
Intermediate Aggregation
Buffered Partial Aggregation
Buffered Intermediate Aggregation
```

临时表完成

每当在表访问的作用域内创建临时表时，将显示一个完成语句。此语句可以是下列其中一项:

```
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

表函数是将数据以表形式返回给语句的用户定义的函数（UDF）。表函数由下列语句指示，这些语句将提供有关函数属性的详细信息。特定名称将唯一地标识所调用的表函数。

```
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
```

连接信息:

db2expln 命令输出可能包含关于所说明语句中的连接的信息。

每当执行连接时，将显示下列其中一个语句:

```
Hash Join
Merge Join
Nested Loop Join
```

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
```

对于合并连接或嵌套循环连接而言，连接的外表将是输出中显示的上一个访问语句中引用的表。连接的内表将是该连接语句的作用域内包含的访问语句中引用的表。对于散列连接而言，将访问语句反向：外表包含在该连接的作用域之内，内表出现在该连接之前。

对于散列连接或合并连接，可能出现下列附加语句:

- `Early Out: Single Match Per Outer Row`

  在某些环境中，连接只需要确定内表中是否有任何行与外表中的当前行匹配。

- ```
  Residual Predicate(s)
  |  #Predicates = n
  ```

  可以在连接完成之后应用谓词。此语句显示正在应用的谓词的数目。

对于散列连接，可能出现下列附加语句:

- `Process Hash Table For Join`

  散列表根据内表进行构建。此语句显示是否已在访问内表期间将散列表的构建工作下推到某个谓词。

- `Process Probe Table For Hash Join`

  访问外表时，可以构建一个探测表以提高连接性能。此语句显示是否已在访问外表期间构建探针表。

- `Estimated Build Size: n`

  此语句显示构建散列表所需的估计字节数。

- `Estimated Probe Size: n`

  此语句显示构建探测表所需的估计字节数。

对于嵌套循环连接，在连接语句之后可能会立即出现以下语句:

```
Piped Inner
```

此语句指示连接的内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，那么应从头到尾阅读说明步骤。例如，假定说明输出包含以下数据流:

```
Access ..... W
Join
|  Access ..... X
Join
|  Access ..... Y
Join
|  Access ..... Z
```

执行步骤将是:

1. 从表 W 中获取合格行。
2. 将 W 中的某一行与表 X 中的下一行连接，并将结果命名为 P1（表示编号为 1 的部分连接结果）。
3. 将 P1 与表 Y 中的下一行连接，以创建 P2。
4. 将 P2 与表 Z 中的下一行连接，以创建一个完整的结果行。
5. 如果 Z 中存在其他行，那么转至步骤 4。
```

6. 如果 Y 中存在其他行，那么转至步骤 3。
7. 如果 X 中存在其他行，那么转至步骤 2。
8. 如果 W 中存在其他行，那么转至步骤 1。

数据流信息:

在一个访问方案内，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”这一概念使您能够将访问方案中的一组操作作为一个单元进行控制。

数据流的开始由 db2expln 输出中的以下语句指示:

```
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都被认为是同一个数据流的组成部分。

数据流具有许多特征，在最初的数据流语句后面，可以有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访问方案中先前生成的值，那么数据流将被标记为:

  ```
  Correlated
  ```

- 与已排序的临时表类似，下列语句指示是否将数据流的结果保留在内存中:

  ```
  Piped
  Not Piped
  ```

  如果执行时没有足够的内存，那么可以将管道式数据流写入磁盘。访问方案支持这两种可能性。

- 以下语句指示此数据流只需要单个记录:

  ```
  Single Record
  ```

访问数据流时，在输出中将出现以下语句:

```
Access Data Stream n
```

INSERT、UPDATE 和 DELETE 信息:

INSERT、UPDATE 或 DELETE 语句的说明文本是自解释性文本。

在 db2expln 输出中，这些 SQL 操作的语句文本可以是:

```
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
```

块和行标识准备信息:

对于某些访问方案而言，在访问表之前，如果对合格行和块标识进行排序并除去重复项（对于索引 OR 运算），或者采用某种技术来确定在所有正在访问的索引中都出现的标识（对于索引 AND 运算），那么将提高效率。

说明输出中显示的标识准备信息有三个主要用途:

- 以下语句指示已使用索引 OR 运算来准备合格标识的列表:

  ```
  Index ORing Preparation
  Block Index ORing Preparation
  ```

  *索引 OR 运算*是指这样一种技术: 访问多个索引并对结果进行组合，以便包括出现于任何索引中的相异标识。当通过 OR 关键字来连接谓词或者存在 IN 谓词时，优化器将考虑进行索引 OR 运算。
- 以下语句指示已准备输入数据以供列表预取期间使用:

  ```
  List Prefetch Preparation
  Block List Prefetch RID Preparation
  ```
- *索引 AND 运算*是指这样一种技术: 访问多个索引并对结果进行组合，以便包括出现于所访问的所有索引中的标识。索引 AND 运算以下列任何一个语句开头:

  ```
  Index ANDing
  Block Index ANDing
  ```

  如果优化器已估算结果集的大小，那么将通过以下语句显示估算值:

  ```
  Optimizer Estimate of Set Size: n
  ```

  索引 AND 运算过滤操作处理标识，并使用位过滤技术来确定出现于所访问的每个索引中的标识。以下语句指示已经为索引 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
  ```

  如果优化器已估算位图的结果集大小，那么此估计值将由以下语句指示:

  ```
  Optimizer Estimate of Set Size: n
  ```

如果可以为任何类型的标识准备工作执行预取，那么这种情况由以下语句指示:

```
Prefetch: Enabled
```

聚集信息:

聚集对满足 SQL 语句中谓词所代表的条件的行执行。

如果执行聚集函数，那么在 db2expln 输出中将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
```

谓词聚集意味着，在访问数据时，已将该聚集操作作为谓词进行处理。

在聚集语句后面跟着另一个声明，该语句标识已执行的聚集函数的类型:

```
Group By
Column Function(s)
Single Record
```

特定的列函数可以由原始 SQL 语句派生而来。将从索引读取单个记录，以满足 MIN 或 MAX 操作。

如果已执行谓词聚集，那么将有聚集完成操作和相应的输出:

```
Aggregation Completion
   Partial Aggregation Completion
   Intermediate Aggregation Completion
   Final Aggregation Completion
```

并行处理信息:

以并行方式执行 SQL 语句（使用分区内或分区间并行性）要求执行一些特殊的访问方案操作。

- 运行分区内并行方案时，将使用多个子代理程序来同时执行该方案的各个部分。这些子代理程序的创建由 db2expln 命令输出中的以下语句指示:

  ```
  Process Using n Subagents
  ```

- 运行分区间并行方案时，该节将分为多个子节。每个子节都被发送到一个或多个数据库分区以便运行。一个重要的子节是*协调程序子节*。协调程序子节是每个方案中的第一个子节。它首先获得控制权并负责分发其他子节，然后将结果返回给调用应用程序。
  - 子节的分发由以下语句指示:

    ```
    Distribute Subsection #n
    ```

  - 以下语句指示将根据列值把子节发送到数据库分区组中的数据库分区。

    ```
    Directed by Hash
    |  #Columns = n
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  - 以下语句指示将子节发送到预先确定的数据库分区。（当语句使用 DBPARTITIONNUM() 标量函数时，这种情况很常见。）

    ```
    Directed by Node Number
    ```

  - 以下示例指示将子节发送到与数据库分区组中预先确定的数据库分区号相对应的数据库分区。（当语句使用 HASHEDVALUE 标量函数时，这种情况很常见。）

    ```
    Directed by Partition Number
    |  Partition Map ID = n, Nodegroup = ngname, #Nodes = n
    ```

  - 以下语句指示将子节发送到为应用程序的游标提供当前行的数据库分区。

    ```
    Directed by Position
    ```

  - 以下语句指示只有编译该语句时确定的一个数据库分区将接收该子节。

    ```
    Directed to Single Node
    |  Node Number = n
    ```

- 下列任何一个语句都指示将在协调程序数据库分区中执行子节。

```
Directed to Application Coordinator Node
Directed to Local Coordinator Node
```

- 以下语句指示将子节发送到列示的所有数据库分区。

```
Broadcast to Node List
|  Nodes = n1, n2, n3, ...
```

- 以下语句指示只有执行该语句时确定的一个数据库分区将接收该子节。

```
Directed to Any Node
```

- 表队列用于在分区数据库环境中的子节之间或者对称多处理器（SMP）环境中的子代理程序之间移动数据。
  - 下列语句指示正在将数据插入到表队列中:

```
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，插入到表队列中的行的目标由下列其中一个语句描述:

    将每一行都发送到协调程序数据库分区:

```
Broadcast to Coordinator Node
```

    将每一行都发送到每个运行给定子节的数据库分区:

```
Broadcast to All Nodes of Subsection n
```

    根据行中的值，将每一行都发送到一个数据库分区:

```
Hash to Specific Node
```

    将每一行都发送到执行语句时确定的数据库分区:

```
Send to Specific Node
```

    将每一行都发送到随机确定的数据库分区:

```
Send to Random Node
```

  - 在某些情况下，数据库分区表队列必须将某些行溢出至临时表。这种可能性由以下语句标识:

```
Rows Can Overflow to Temporary Table
```

  - 在包含将行插入到表队列的下推操作的表访问之后，有一个“完成”语句，此语句处理未能立即发送的行。在这种情况下，将显示下列其中一行:

```
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
```

  - 下列语句指示正在从表队列检索数据:

```
Access Table Queue  ID = qn
Access Local Table Queue  ID = qn
```

    在这些语句后面，始终是所检索的列数。

```
#Columns = n
```

  - 如果表队列在接收端对行进行排序，那么将出现下列其中一个语句:

```
Output Sorted
Output Sorted and Unique
```

在这些语句后面是用于执行排序操作的键的数目。

```
#Key Columns = n
```

对于排序键中的每一列，将显示下列其中一个语句:

```
Key n: (Ascending)
Key n: (Descending)
```

– 如果将在表队列的接收端将谓词应用于行，那么将显示以下语句:

```
Residual Predicate(s)
|  #Predicates = n
```

- 在分区数据库环境中，某些子节将明确地循环回到该子节的开头，这种情况下将显示以下语句:

```
Jump Back to Start of Subsection
```

联合查询信息:

在联合数据库中执行 SQL 语句时，要求能够对其他数据源执行该语句的某些部分。

db2expln 命令的以下输出表明将读取数据源:

```
Ship Distributed Subquery #n
|   #Columns = n
```

如果将谓词应用于从分布式子查询返回的数据，那么所应用的谓词的数目由下列语句指示:

```
Residual Predicate(s)
|   #Predicates = n
```

发生在数据源处的插入、更新或删除操作由下列其中一个语句指示:

```
Ship Distributed Insert #n
Ship Distributed Update #n
Ship Distributed Delete #n
```

如果在数据源处显式地锁定了表，那么将出现以下语句:

```
Ship Distributed Lock Table #n
```

对数据源执行的数据定义语言（DDL）语句分为两部分。在数据源处调用的部分由以下语句指示:

```
Ship Distributed DDL Statement #n
```

如果联合服务器是分区数据库，那么 DDL 语句的一部分必须在目录数据库分区中运行。这种情况由以下语句指示:

```
Distributed DDL Statement #n Completion
```

每个分布式子语句的详细信息都单独显示。

- 子查询的数据源由下列其中一个语句指示:

```
Server: server_name (type, version)
Server: server_name (type)
Server: server_name
```

- 对于关系数据源，子语句的 SQL 将显示为:

```
SQL Statement:
     statement
```

非关系数据源由以下语句指示:

```
Non-Relational Data Source
```

- 子语句中引用的昵称按以下方式列示:

```
Nicknames Referenced:
   schema.nickname ID = n
```

对于关系数据源，昵称的基本表按以下方式显示:

```
Base = baseschema.basetable
```

对于非关系数据源，昵称的源文件按以下方式显示:

```
Source File = filename
```

- 在执行子语句之前，如果将值从联合服务器传递到数据源，那么值的数目由以下语句指示:

```
#Input Columns: n
```

- 在执行子语句之后，如果将值从数据源传递到联合服务器，那么值的数目由以下语句指示:

```
#Output Columns: n
```

其他说明信息:

db2expln 命令的输出还包含其他不太容易分类的有用信息。

- 在输出中，数据定义语言（DDL）语句的各个节由以下语句指示:

```
DDL Statement
```

除此以外，不提供有关 DDL 语句的其他说明输出。

- 在输出中，用于可更新的专用寄存器的 SET 语句（例如 CURRENT EXPLAIN SNAPSHOT）的各个节由以下语句指示:

```
SET Statement
```

除此以外，不提供有关 SET 语句的其他说明输出。

- 如果 SQL 语句包含 DISTINCT 子句，那么在输出中可能会出现以下语句:

```
Distinct Filter  #Columns = n
```

其中，*n* 是获取相异行时涉及的列数。要检索相异的行值，必须先对行执行排序以便消除重复的行。如果数据库管理器不必显式地消除重复行，例如下列情况，那么此语句将不会出现:

  – 存在唯一的索引，且索引键中的所有列都是 DISTINCT 操作的组成部分
  – 在排序期间可以消除重复行

- 如果下一个操作取决于特定的记录标识，那么将出现以下语句:

```
Positioned Operation
```

如果此定位型操作是对联合数据源执行的，那么此语句将变为:

```
Distributed Positioned Operation
```

对于任何使用 WHERE CURRENT OF 语法的 SQL 语句，都将出现此语句。

- 如果存在必须应用于结果但不能作为另一个操作的组成部分进行应用的谓词，那么将出现以下语句:

```
Residual Predicate Application
|  #Predicates = n
```

- 如果 SQL 语句包含 UNION 运算符，那么将出现以下语句:

```
UNION
```

- 如果访问方案中某个操作的唯一用途是生成供后续操作使用的行值，那么将出现以下语句:

```
Table Constructor
|  n-Row(s)
```

可以使用表构造函数将一个集合中的各个值变换成可以传递到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
```

- 如果存在只有在特定情况下才会被处理的操作，那么将出现以下语句:

```
Conditional Evaluation
|  Condition #n:
|  #Predicates = n
|  Action #n:
```

条件判定用于实现 CASE 语句之类的活动，或者用于实现引用完整性约束或触发器之类的内部机制。如果未显示任何操作，那么该条件成立时，将只执行数据处理操作。

- 如果访问方案正在处理 ALL、ANY 或 EXISTS 子查询，那么将出现下列其中一个语句:

```
ANY/ALL Subquery
EXISTS Subquery
EXISTS SINGLE Subquery
```

- 在某些更新或删除操作之前，有必要确定特定行在表中的位置。这种情况由以下语句指示:

```
Establish Row Position
```

- 对于对多维集群表执行的符合转出优化条件的删除操作，将出现下列其中一个语句:

```
CELL DELETE with deferred cleanup
CELL DELETE with immediate cleanup
```

- 如果正在将行返回给应用程序，那么将出现以下语句:

```
Return Data to Application
|  #Columns = n
```

如果已下推此操作以便在访问表时执行，那么在输出中将出现完成阶段语句:

```
Return Data Completion
```

- 如果正在调用存储过程，那么将出现以下语句:

```
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大对象（LOB）定位器，那么将出现以下语句:

```
Free LOB Locators
```

优化查询访问方案

语句集中器降低了编译开销

语句集中器在数据库服务器上修改动态 SQL 语句，以使类似而不等同的 SQL 语句可以共享同一个访问方案。

在联机事务处理（OLTP）系统中，可能会反复生成包含不同字面值的简单语句。在此类工作负载中，重新编译语句的成本会导致开销大幅增加。语句集中器通过允许重复使用已编译的语句（而不考虑字面值）来消除此开销。

缺省情况下，语句集中器处于禁用状态。要对数据库中的所有动态语句启用语句集中器，请将 **stmt_conc** 数据库配置参数设置为 LITERALS。只有前 100000 个字面值才会被替换；其余字面值保持不变。

语句集中器通过修改传入动态 SQL 语句来提高性能。在适合于语句集中器的工作负载中，相对于通过复用程序包高速缓存中已包含的语句所节省的成本，与修改传入 SQL 语句相关联的开销微不足道。

如果语句集中过程导致修改动态语句，那么原始语句和修改后的语句都将显示在说明输出中。如果语句集中器已修改原始语句文本，那么事件监视器逻辑监视元素以及 MON_GET_ACTIVITY_DETAILS 表函数的输出都将显示原始语句。其他监视器界面将仅显示修改后的语句文本。

请考虑以下示例，其中，**stmt_conc** 数据库配置参数设置为 LITERALS，并且执行了下面这两个语句:

```
select firstnme,lastname from employee where empno='000020'
select firstnme,lastname from employee where empno='000070'
```

这些语句共享程序包高速缓存中的同一个条目，并且该条目使用以下语句:

```
select firstnme,lastname from employee where empno=:L0000000000
```

数据服务器将根据原始语句中使用的字面值来提供 `:L0000000000` 的值（“000020”或“000070”）。

由于语句集中过程将更改语句文本，因此会对访问方案的选择产生影响。如果程序包高速缓存中的类似语句具有类似的访问方案，那么应该使用语句集中器。如果一个语句中的不同字面值导致访问方案显著不同，那么不应对该语句启用语句集中器。

访问方案复用

对于为程序包中的静态 SQL 语句选择的访问方案，您可以请求使其在多次绑定或重新绑定操作期间与现有访问方案保持相同或非常相似。

访问方案复用功能可以避免在未经您明确核准的情况下发生重大方案更改。虽然这可能意味着查询无法通过潜在的访问方案改进受益，但访问方案复用功能所提供的控制使您能够在一切就绪时测试和实现那些改进。在此之后，您可以继续使用现有的访问方案并实现稳定并且可预测的性能。

要启用访问方案复用功能，请使用 ALTER PACKAGE 语句，或者使用 BIND、REBIND 或 PRECOMPILE 命令的 APREUSE 选项。对于启用了访问方案复用功能的程序包，SYSCAT.PACKAGES 目录视图的 APREUSE 列包含值 Y。

ALTER_ROUTINE_PACKAGE 过程是一种对编译型 SQL 对象（例如 SQL 过程）启用访问方案复用功能的简便方法。但是，访问方案在编译型对象重新生效期间不可复用，这是因为该对象在重新绑定前将被删除。在这种情况下，APREUSE 将只在下次绑定或重新绑定该程序包时生效。

将模式或编译环境更改控制在最低程度时，访问方案复用功能最为有效。如果进行重大更改，那么可能无法重新创建先前的访问方案。此类重大更改的示例包括，删除访问方案所使用的索引或者在另一优化级别重新编译 SQL 语句。对查询编译器的语句分析所作的重大更改也可能导致先前访问方案不再可复用。

您可以将访问方案复用功能与优化准则相结合。语句级准则优先于它所应用于的静态 SQL 语句的访问方案复用功能。没有语句级准则的静态语句的访问方案可复用，但它们不能与任何已指定的通用优化准则发生冲突。您可以使用具有空准则的语句概要文件对特定语句禁用访问方案复用功能，而使访问方案复用功能可用于程序包中的其他静态语句。

注: 版本 9.7 以前的发行版所生成的程序包中的访问方案不可复用。

即使无法复用某个访问方案，编译也将继续，但将返回警告（SQL20516W），并且原因码将指示复用访问方案的尝试不成功的原因。有时，说明工具所提供的诊断消息还包含其他信息。

优化类

编译 SQL 或 XQuery 语句时，可以指定优化类以确定优化器如何为该语句选择最高效的访问方案。

各个优化类在编译查询期间考虑的优化策略数目和类型方面有所区别。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。

您可以在编译 SQL 或 XQuery 语句时指定下列其中一种优化类。

0 此类通知优化器在生成访问方案时进行最少量的优化，并具有下列特征:

- 优化器不考虑高频值统计信息。
- 仅应用基本的查询重写规则。
- 使用了贪婪联合枚举。
- 仅允许使用嵌套循环连接及索引扫描访问方法。
- 在生成的访问方法中不使用列表预取。
- 不考虑星型连接策略。

此类应该只用于需要最低的查询编译开销的情况。查询优化类 0 适用于以下应用程序: 完全由访问索引结构良好的表的非常简单的动态 SQL 或 XQuery 语句组成。

1 此优化类具有下列特征:

- 优化器不考虑高频值统计信息。

- 只应用查询重写规则的一个子集。
- 使用了贪婪联合枚举。
- 在生成的访问方法中不使用列表预取。

除了合并扫描连接及表扫描也可用以外，优化类 1 类似于类 0。

2 此类通知优化器使用比类 1 显著高的优化程度，而使复杂查询的编译成本显著低于类 3 及更高的类。此优化类具有下列特征:

- 使用所有可用的统计信息，其中包括高频值和分位数统计信息。
- 除只在极少情况下才适用的计算密集型规则外，将应用所有其他查询重写规则，其中包括具体化查询表路由。
- 使用了贪婪联合枚举。
- 考虑各种访问方法，包括列表预取和具体化查询表路由。
- 如果适用，请考虑星型连接策略。

优化类 2 除了使用贪婪联合枚举而不是动态规划联合枚举以外，类似于类 5。在所有使用贪婪联合枚举算法的类中，此类具有最高的优化程度，与类 3 及更高的类相比，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。建议将类 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，因此访问方案不太可能在高速缓存中停留到出现下一个查询为止。

3 此类表示中等程度的优化，与 DB2 z/OS 版的查询优化特征最为接近。此优化类具有下列特征:

- 如果有可用的高频值统计信息，那么使用那些统计信息。
- 应用大部分查询重写规则，包括子查询至连接的变换。
- 使用动态规划联合枚举，并且:
  - 组合内表的有限使用
  - 涉及查找表的星型模式的笛卡尔乘积的有限使用
- 考虑各种访问方法，包括列表预取、索引 AND 运算和星型连接。

此类适用于大量应用程序，能够改进具有 4 个或更多连接的查询的访问方案。

5 此类指导优化器使用相当多的优化来生成访问方案，并具有下列特征:

- 使用所有可用的统计信息，其中包括高频值和分位数统计信息。
- 除只在极少情况下才适用的计算密集型规则外，将应用所有其他查询重写规则，其中包括具体化查询表路由。
- 使用动态规划联合枚举，并且:
  - 组合内表的有限使用
  - 涉及查找表的星型模式的笛卡尔乘积的有限使用
- 考虑各种访问方法，包括列表预取、索引 AND 运算和具体化查询表路由。

对于同时包含事务处理和复杂查询的混合环境而言，优化类 5（缺省类）是一个很好的选择。此优化类设计成可以用高效的方式应用最有价值的查询变换和其他查询优化技术。

如果优化器检测到无法保证用于复杂动态 SQL 或 XQuery 语句的其他资源和处理时间，那么将减少优化。减少的范围取决于机器大小和谓词数目。当优化

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，它使用贪婪联合枚举法，并减少了考虑的访问方案的组合数。

7 此类指导优化器使用相当多的优化来生成访问方案。此类与优化类 5 类似，但优化器不考虑减少用于复杂动态 SQL 或 XQuery 语句的查询优化量。

9 此类指导优化器使用所有可用的优化技术。这些主要功能包括:

- 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
- 所有访问方法

此类将增加由优化器考虑的可能的访问方案数量。对于使用大表的很复杂或者运行时间很长的查询，可以使用此类来确定更全面优化是否将生成更好的访问方案。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。

选择优化类:

相对于明确指定优化技术，设置优化类有一些优点。

在下列情况下尤其如此:

- 管理非常小型的数据库或非常简单的动态查询
- 在编译时适应数据库服务器上的内存限制
- 缩短查询编译时间；例如，在语句准备期间缩短此时间

通过使用缺省优化类 5，可以借助合理的资源量对大多数语句进行充分优化。查询编译时间和资源耗用量主要受查询的复杂性影响，尤其受连接数和子查询数影响。但是，编译时间和资源耗用量也受所执行的优化量影响。

查询优化类 1、2、3、5 和 7 都比较通用。仅当您需要进一步缩短查询编译时间，并且 SQL 和 XQuery 语句非常简单时，您才应考虑使用查询优化类 0。

技巧: 要分析长时间运行的查询，请使用 db2batch 来运行该查询，以便确定编译和执行该查询所耗用的时间。如果编译时间过长，那么请降低优化类。如果执行需要是一个问题，那么请考虑更高的优化类。

在选择优化类时，请考虑下列一般准则:

- 通过使用缺省查询优化类 5 入手。
- 选择除缺省优化类以外的优化类时，请先尝试优化类 1、2 或 3。优化类 0、1 和 2 使用贪婪联合枚举算法。
- 如果有许多表，并且有许多应用于同一个列的连接谓词，并且您关注编译时间，那么请使用优化类 1 或 2。
- 对运行时间非常短（不到一秒）的查询，请使用低优化类（0 或 1）。这些查询趋向于:
  - 访问单个表或仅访问少量的表
  - 访存一行或仅访存少量的行
  - 使用标准的唯一索引
  - 涉及联机事务处理（OLTP）
- 对于运行时间较长（超过 30 秒）的查询，请使用较高的优化类（3、5 或 7）。

- 优化类 3 或更高的优化类使用动态编程联合枚举算法，与优化类 0、1 和 2 相比，此算法考虑更多备用方案并可能使编译时间显著延长，表的数目增加时尤其如此。
- 仅当查询有特别的优化需求时，才应使用优化类 9。

复杂的查询可能要求进行不同程度的优化才能选择最佳访问方案。对于具有下列特征的查询，请考虑使用较高的优化类:

- 访问大型表
- 查询大量视图
- 包含大量谓词
- 包含许多子查询
- 包含许多连接
- 包含许多集合运算符，例如 UNION 或 INTERSECT
- 很多合格行
- 包含 GROUP BY 和 HAVING 操作
- 包含嵌套表表达式

对完全规范化数据库执行的决策支持查询或月结报告查询是复杂查询的典型示例，对于这些查询，至少应该使用缺省查询优化类。

对于由查询生成器生成的 SQL 和 XQuery 语句，请使用较高的查询优化类。许多查询生成器创建的查询的运行效率不高。编写得很差的查询需要进行额外的优化才能选择良好的访问方案。使用查询优化类 2 或更高优化类可以提高此类查询的性能。

对于 SAP 应用程序，始终应该使用优化类 5。此优化类将启用许多针对 SAP 优化的 DB2 功能，例如设置 **DB2_REDUCED_OPTIMIZATION** 注册表变量。

在联合数据库中，优化类不适用于远程优化器。

设置优化类:

指定优化级别时，请考虑查询是使用静态 SQL 和 XQuery 语句还是动态 SQL 和 XQuery 语句，以及是否重复执行同一个动态查询。

对于静态 SQL 和 XQuery 语句，只需耗费一次查询编译时间和资源，而生成的方案可以使用许多次。通常，静态 SQL 和 XQuery 语句应该始终使用缺省查询优化类（5）。因为动态语句在运行时进行绑定和执行，所以，请考虑对动态语句进行附加的优化开销是否提高整体性能。但是，如果重复执行相同的动态 SQL 或 XQuery 语句，那么将对所选访问方案进行高速缓存。这种语句可以使用与静态 SQL 或 XQuery 语句相同的优化级别。

如果您不确定查询能否从附加优化获益，或者您关心编译时间和资源使用情况，那么请考虑进行基准程序测试。

要指定查询优化类，请完成下列步骤:

1. 分析性能因素。
   - 对于动态查询语句，测试应该对该语句的平均运行时间进行比较。请使用以下公式来估算平均运行时间:

```
编译时间 + 所有迭代的执行时间之和
------------------------------------------------------------
                    迭代次数
```

迭代次数表示每次编译语句时，预期该语句将要执行的次数。

注: 在初始编译之后，当环境更改要求重新编译动态 SQL 和 XQuery 语句时，将进行重新编译。对语句进行高速缓存之后，如果环境未更改，那么后续 PREPARE 语句将复用高速缓存的语句。

- 对于静态 SQL 和 XQuery 语句，请比较语句运行时间。

  尽管您可能对静态 SQL 和 XQuery 语句的编译时间也感兴趣，但静态语句的编译和执行总时间在任何有意义的上下文中都难以估计。比较总时间并不能认识到这样一个事实，即每次绑定静态语句，它可以执行多次，而在运行时，一般不对其进行绑定。

2. 指定优化类。
   - 动态 SQL 和 XQuery 语句使用 CURRENT QUERY OPTIMIZATION 专用寄存器指定的优化类。例如，以下语句将优化类设置为 1:

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

     要确保动态 SQL 或 XQuery 语句始终使用同一个优化类，可以在应用程序中指定 SET 语句。

     如果尚未设置 CURRENT QUERY OPTIMIZATION 专用寄存器，那么将使用缺省查询优化类来绑定动态语句。动态查询和静态查询的缺省值都由数据库配置参数 **dft_queryopt** 的值确定（缺省值为 5）。绑定选项和专用寄存器的缺省值也从 **dft_queryopt** 数据库配置参数读取。
   - 静态 SQL 和 XQuery 语句使用 PREP 和 BIND 命令指定的优化类。SYSCAT.PACKAGES 目录视图中的 QUERYOPT 列记录用于绑定程序包的优化类。如果以隐式方式或使用 REBIND PACKAGE 命令来重新绑定程序包，那么此优化类将用于静态语句。要更改这种静态 SQL 和 XQuery 语句的优化类，请使用 BIND 命令。如果未指定优化类，那么数据服务器将使用 **dft_queryopt** 数据库配置参数指定的缺省优化类。

在其他调整选项未产生可接受的结果时使用优化概要文件

如果您已遵循最佳实践建议，但相信仍未实现最佳性能，那么可以为 DB2 优化器提供明确的优化准则。

这些优化准则包含在称为优化概要文件的 XML 文档中。概要文件定义 SQL 语句及其相关联的优化准则。

如果广泛使用优化概要文件，那么需要进行大量维护工作。更重要的是，使用优化概要文件只能提高现有 SQL 语句的性能。始终如一地遵循最佳实践有助于确保所有查询（包括将来的查询）实现稳定的查询性能。

优化概要文件和准则

优化概要文件是包含一个或多个 SQL 语句的优化准则的 XML 文档。通过使用 SQL 文本和明确标识一个 SQL 语句所需的其他信息建立每个 SQL 语句与其关联的优化准则之间的通信。

DB2 优化器是业界最先进的、基于成本的优化器之一。但是，在极少情况下，此优化器可能会选择不是最佳的执行方案。作为熟悉数据库的 DBA，您可以使用 db2advis、RUNSTATS 和 db2expln 之类的实用程序以及优化类设置来帮助调整优化器，以获得更好的数据库性能。如果在使用了所有调整选项后未获得期望的结果，那么可以为 DB2 优化器提供显式优化准则。

例如，假定即使在更新数据库统计信息并执行所有其他调整步骤之后，优化器仍未选择 I_SUPPKEY 索引来访问以下子查询中的 SUPPLIERS 表:

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P.P_SIZE = 39
  AND P.P_TYPE = 'BRASS'
  AND S.S_NATION = 'MOROCCO'
  AND S.S_NATION IN ('MOROCCO', 'SPAIN')
  AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                          FROM PARTSUPP PS1, SUPPLIERS S1
                          WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                            AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                            AND S1.S_NATION = S.S_NATION))
```

在这种情况下，可以使用显式的优化准则来影响优化器。例如:

```
<OPTGUIDELINES><IXSCAN TABLE="S" INDEX="I_SUPPKEY"/></OPTGUIDELINES>
```

优化准则是通过简单的 XML 规范指定的。 DB2 优化器将每个元素都解释为优化准则。在此示例中，有一个优化准则元素。IXSCAN 元素请求优化器使用索引访问方法。IXSCAN 元素的 TABLE 属性通过使用表引用的显示名来指示目标表引用，INDEX 属性指定索引。

以下示例基于上一个查询，它说明如何使用优化概要文件将优化准则传递给 DB2 优化器。

```
<?xml version="1.0" encoding="UTF-8">

<OPTPROFILE VERSION="9.1.0.0">
<STMTPROFILE ID="Guidelines for TPCD Q9">
  <STMTKEY SCHEMA="TPCD">
    SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
    FROM PARTS P, SUPPLIERS S, PARTSUPP PS
    WHERE P_PARTKEY = PS.PS_PARTKEY
      AND S.S_SUPPKEY = PS.PS_SUPPKEY
      AND P.P_SIZE = 39
      AND P.P_TYPE = 'BRASS'
      AND S.S_NATION = 'MOROCCO'
      AND S.S_NATION IN ('MOROCCO', 'SPAIN')
      AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                              FROM PARTSUPP PS1, SUPPLIERS S1
                              WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                                AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                                AND S1.S_NATION = S.S_NATION))
  </STMTKEY>
  <OPTGUIDELINES><IXSCAN TABLE="S" INDEX="I_SUPPKEY"/></OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>
```

每个 STMTPROFILE 元素都为一个应用程序语句提供一组优化准则。目标语句由 STMTKEY 子元素标识。然后，为优化概要文件提供模式限定名并将此概要文件插入到数据库中。通过在 BIND 或 PRECOMPILE 命令中指定优化概要文件的名称，即可使此优化概要文件对该语句生效。

优化概要文件允许为优化器提供优化准则，而不需要更改应用程序或数据库配置。您只需编写一个简单的 XML 文档，将此文档插入到数据库中，然后在 BIND 或 PRECOMPILE 命令中指定优化概要文件的名称。优化器会自动使优化准则与相应的语句匹配。

优化准则不必是全面的，但应以期望的执行方案为目标。DB2 优化器仍考虑使用基于成本的现有方法来选择其他可能的访问方案。以特定表引用为目标的优化准则不能覆盖一般优化设置。例如，指定表 A 与表 B 之间的合并连接的优化准则在优化类 0 无效。

优化器将忽略无效或不适用的优化准则。如果有任何优化准则被忽略，那么将会创建执行方案并返回 SQL0437W（原因码为 13）。然后，可以使用 EXPLAIN 语句来获取有关优化准则处理的详细诊断信息。

优化概要文件:

优化概要文件的结构:

优化概要文件可以包含全局准则，这些准则将应用于该概要文件生效期间执行的所有数据操作语言（DML）语句。优化概要文件还可以包含特定准则，这些准则将应用于程序包中的各个 DML 语句。

例如:

- 您可以编写一个全局优化准则，它指定对于当前优化概要文件处于活动状态时处理的每个语句，请求优化器引用具体化查询表（MQT）Test.SumSales 和 Test.AvgSales。
- 您可以编写一个语句级优化准则，它指定每当优化器遇到所指定语句时，使用 I_SUPPKEY 索引来访问 SUPPLIERS表。

优化概要文件包含两个主要的节，您可以在这两节中指定这两种类型的准则：全局优化准则节可以包含一个 OPTGUIDELINES 元素，语句概要文件节可以包含任意数目的 STMTPROFILE 元素。优化概要文件还必须包含一个 OPTPROFILE 元素，此元素包含元数据和处理伪指令。

以下代码是用于 DB2 版本 9.1 的有效优化概要文件的示例，它包含全局优化准则节以及带有一个 STMTPROFILE 元素的语句概要文件节。

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="9.1.0.0">

  <!--
       Global optimization guidelines section.
       Optional but at most one.
  -->
  <OPTGUIDELINES>
    <MQT NAME="Test.AvgSales"/>
    <MQT NAME="Test.SumSales"/>
  </OPTGUIDELINES>

  <!--
       Statement profile section.
       Zero or more.
  -->
  <STMTPROFILE ID="Guidelines for TPCD Q9">
    <STMTKEY SCHEMA="TPCD">
      <![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
S.S_COMMENT FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY
AND P.P_SIZE = 39 AND P.P_TYPE = 'BRASS'
AND S.S_NATION = 'MOROCCO' AND S.S_NATION IN ('MOROCCO', 'SPAIN')
AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
FROM PARTSUPP PS1, SUPPLIERS S1
WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                            AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                            AND S1.S_NATION  = S.S_NATION)]]>
    </STMTKEY>
```

```
    <OPTGUIDELINES>
      <IXSCAN TABID="Q1" INDEX="I_SUPPKEY"/>
    </OPTGUIDELINES>
  </STMTPROFILE>

</OPTPROFILE>
```

OPTPROFILE 元素

优化概要文件以 OPTPROFILE 元素开头。在上一个示例中，此元素包含一个 VERSION 属性，该属性指定优化概要文件的版本为 9.1。

全局优化准则节

全局优化准则将应用于优化概要文件所作用于的所有语句。全局优化准则节由全局 OPTGUIDELINES 元素表示。在上一个示例中，此节包含一个全局优化准则。该准则指定，在处理此优化概要文件所作用于的任何语句时，应考虑 MQT Test.AvgSales 和 Test.SumSales。

语句概要文件节

语句概要文件定义适用于特定语句的优化准则。优化概要文件中可以没有语句概要文件，也可以有多个语句概要文件。语句概要文件节由 STMTPROFILE 元素表示。在上一个示例中，此节包含优化概要文件所作用于的特定语句的准则。

每个语句概要文件都包含分别由 STMTKEY 和 OPTGUIDELINES 元素表示的语句关键字和语句级优化准则。

- 语句关键字用于标识语句级优化准则所应用于的语句。在此示例中，STMTKEY 元素包含原始语句文本以及明确标识此语句所需的其他信息。优化器使用语句关键字使语句概要文件与相应的语句匹配。此关系使您能够为语句提供优化准则，而不必修改应用程序。
- 语句概要文件的语句级优化准则节由 OPTGUIDELINES 元素表示。此节由一个或多个访问请求或连接请求构成，这些请求指定语句中访问和连接表的方法。在与语句概要文件中的语句关键字成功匹配之后，优化器将在优化语句时引用相关联的语句级优化准则。此示例包含一个访问请求，该请求指定，嵌套子选择中引用的 SUPPLIERS 表使用名为 I_SUPPKEY 的索引。

创建优化概要文件:

优化概要文件是一个 XML 文档，它包含一个或多个数据操作语言（DML）语句的优化准则。

由于优化概要文件可以包含许多准则组合，因此以下信息仅包含创建任何优化概要文件时都必须执行的公共步骤。

要创建优化概要文件，请执行下列操作:

1. 启动 XML 编辑器。如果有可能话，请使用具有模式验证功能的 XML 编辑器。优化器不会执行 XML 验证。根据当前优化概要文件模式，优化概要文件必须有效。
2. 使用有意义的名称创建新的 XML 文档。您可能希望名称描述文档所适用于的语句范围。例如: `inventory_db.xml`。
3. 在此文档中添加 XML 声明。如果未指定编码格式，那么假定为 UTF-8。如果有可能话，请使用 UTF-16 编码来保存文档。数据服务器处理此编码时的效率更高。

```
<?xml version="1.0" encoding="UTF-16"?>
```

4. 在此文档中添加优化概要文件节。

```
<OPTPROFILE VERSION="9.1.0.0">
</OPTPROFILE>
```

5. 在 OPTPROFILE 元素中，适当地创建全局或语句级优化准则，然后保存文件。

配置数据服务器以使用优化概要文件:

在创建优化概要文件并且针对当前优化概要文件模式（COPS）验证该概要文件的内容之后，这些内容必须与唯一的模式限定名相关联，并且必须存储在 SYSTOOLS.OPT_PROFILE 表中。

要配置数据服务器以使用优化概要文件，请执行下列操作:

1. 创建优化概要文件表。 此表的每一行都可以包含一个优化概要文件: SCHEMA 和 NAME 列标识优化概要文件，而 PROFILE 列包含优化概要文件的文本。
2. 可选: 您可以授予任何对此表的权限或特权，以便满足数据库安全性要求。这不会影响优化器读取该表的能力。
3. 在此表中插入要使用的优化概要文件。

优化概要文件中的 STMTKEY 字段:

在 STMTPROFILE 中，目标语句由 STMTKEY 子元素标识。STMTKEY 字段中定义的语句必须与应用程序所执行的语句完全匹配，从而允许 DB2 明确地标识目标语句。但是，容许语句中存在空格。

一旦 DB2 找到与当前编译关键字相匹配的语句关键字，它就会停止查找；因此，如果优化概要文件中有多个语句概要文件的语句关键字与当前编译关键字相匹配，那么将只使用第一个这样的语句概要文件（按照文档顺序）。此外，在这种情况下不会发出错误或警告。

如果编译关键字具有缺省模式“COLLEGE”以及函数路径 SYSIBM,SYSFUN,SYSPROC,DAVE，那么语句关键字将与“select * from orders where *foo(orderkey)*>20”语句相匹配。

```
<STMTKEY SCHEMA='COLLEGE' FUNCPATH='SYSIBM,SYSFUN,SYSPROC,DAVE'>
<![CDATA[select * from orders where foo(orderkey)>20[[>
</stmtkey>
```

指定优化器将使用的优化概要文件:

使用 OPTPROFILE 绑定选项来指定要在程序包级别使用优化概要文件，或者使用 CURRENT OPTIMIZATION PROFILE 专用寄存器来指定要在语句级别使用优化概要文件。

此专用寄存器包含动态准备好进行优化的语句所使用的优化概要文件的限定名。对于 CLI 应用程序，可以使用 CURRENTOPTIMIZATIONPROFILE 客户机配置选项为每个连接设置此专用寄存器。

OPTPROFILE 绑定选项设置还指定 CURRENT OPTIMIZATION PROFILE 专用寄存器的缺省优化概要文件。缺省值的优先顺序如下所示:

- 无论任何其他设置为何，OPTPROFILE 绑定选项都适用于所有静态语句。
- 对于动态语句，CURRENT OPTIMIZATION PROFILE 专用寄存器的值由下列内容确定，其优先顺序从低到高:

– OPTPROFILE 绑定选项
– CURRENTOPTIMIZATIONPROFILE 客户机配置选项
– 应用程序中的最新 SET CURRENT OPTIMIZATION PROFILE 语句

在应用程序中设置优化概要文件:

通过使用 SET CURRENT OPTIMIZATION PROFILE 语句，您可以控制应用程序中动态语句的当前优化概要文件的设置。

您在此语句中提供的优化概要文件名必须是模式限定的名称。如果未提供模式名，那么 CURRENT SCHEMA 专用寄存器的值将用作隐式模式限定符。

指定的优化概要文件将应用于所有后续动态语句，直到遇到另一个 SET CURRENT OPTIMIZATION PROFILE 语句为止。静态语句不受影响，这是因为，在对此设置进行求值之前，已对这些语句进行预处理和打包。

要在应用程序中设置优化概要文件，请执行下列操作:

- 可以在应用程序中的任何位置使用 SET CURRENT OPTIMIZATION PROFILE 语句。例如，将根据 JON.SALES 优化概要文件来优化以下序列中的最后一个语句。

```
EXEC SQL  SET CURRENT OPTIMIZATION PROFILE = 'NEWTON.INVENTDB';

/* The following statements are both optimized with 'NEWTON.INVENTDB' */
EXEC SQL PREPARE stmt FROM SELECT ... ;
EXEC SQL EXECUTE stmt;

EXEC SQL EXECUTE IMMEDIATE SELECT ... ;

EXEC SQL SET CURRENT OPTIMIZATION PROFILE = 'JON.SALES';

/* This statement is optimized with 'JON.SALES' */
EXEC SQL EXECUTE IMMEDIATE SELECT ... ;
```

- 如果要让优化器使用应用程序开始运行时有效的缺省优化概要文件，请指定 NULL 值。例如:

```
EXEC SQL SET CURRENT OPTIMIZATION PROFILE = NULL;
```

- 如果不想让优化器使用优化概要文件，请指定空字符串。例如:

```
EXEC SQL SET CURRENT OPTIMIZATION PROFILE = '';
```

- 如果您正在使用调用级接口（CLI）应用程序，那么可以使用配置助手或 UPDATE CLI CONFIGURATION 命令在 db2cli.ini 文件中添加 CURRENTOPTIMIZATIONPROFILE 参数。例如:

```
update cli cfg for section sanfran using currentoptimizationprofile jon.sales
```

这将导致在 db2cli.ini 文件中生成以下条目:

```
[SANFRAN]
CURRENTOPTIMIZATIONPROFILE=JON.SALES
```

注: 应用程序中的任何 SET CURRENT OPTIMIZATION PROFILE 语句都将覆盖此设置。

将优化概要文件与程序包绑定:

通过使用 **BIND** 或 **PRECOMPILE** 命令来准备程序包时，可以使用 OPTPROFILE 选项来指定要用于该程序包的优化概要文件。

这是将优化概要文件应用于静态语句的唯一方法，并且指定的概要文件将应用于该程序包中的所有静态语句。通过这种方式指定的优化概要文件还将是用于该程序包中动态语句的缺省优化概要文件。

您可以使用 API（例如 sqlaprep）或命令行处理器（CLP）在 SQLJ 或嵌入式 SQL 中绑定优化概要文件。
例如，以下代码说明如何从 CLP 中将库存数据库优化概要文件与库存应用程序绑定:

```
db2 prep inventapp.sqc bindfile optprofile newton.inventdb
db2 bind inventapp.bnd
db2 connect reset
db2 terminate
xlc -I$HOME/sqllib/include -c inventapp.c -o inventapp.o
xlc -o inventapp inventapp.o -ldb2 -L$HOME/sqllib/lib
```

如果未对优化概要文件指定模式名，那么 QUALIFIER 选项将用作隐式限定符。

修改优化概要文件:

您可以通过编辑文档、针对当前优化概要文件模式（COPS）验证此文档并将 SYSTOOLS.OPT_PROFILE 表中的原始文档替换为新版本来修改优化概要文件。

引用优化概要文件时，将对其进行编译并在内存中对其进行高速缓存；因此，还必须除去这些引用。请使用 FLUSH OPTIMIZATION PROFILE CACHE 语句从优化概要文件高速缓存中除去旧的优化概要文件，并使动态方案高速缓存中任何使用旧概要文件准备的语句都失效（*逻辑失效*）。要修改优化概要文件，请执行下列操作:

1. 编辑优化概要文件以进行必需的更改并验证 XML。
2. 使用新概要文件来更新 SYSTOOLS.OPT_PROFILE 表。
3. 如果未创建用于对优化概要文件高速缓存执行清仓的触发器，那么请发出 FLUSH OPTIMIZATION PROFILE CACHE 语句，以除去优化概要文件高速缓存中可能包含的任何优化概要文件版本。

   注: 对优化概要文件高速缓存执行清仓时，还将使动态方案高速缓存中任何使用旧优化概要文件准备的动态语句失效。

对此优化概要文件的任何后续引用都将导致优化器读取新概要文件并将其重新装入到优化概要文件高速缓存中。此外，由于将以逻辑方式使通过旧优化概要文件准备的语句失效，因此，任何对那些语句的调用都将通过新优化概要文件进行准备并在动态方案高速缓存中重新进行高速缓存。

删除优化概要文件:

通过从 SYSTOOLS.OPT_PROFILE 表中删除不再需要的优化概要文件，可以除去此优化概要文件。引用优化概要文件时，将对其进行编译并在内存中对其进行高速缓存；因此，如果曾使用原始概要文件，那么还必须从优化概要文件高速缓存中清除已删除的优化概要文件。

要删除优化概要文件，请执行下列操作:

1. 从 SYSTOOLS.OPT_PROFILE 表中删除优化概要文件。例如:

```
delete from systools.opt_profile
  where schema = 'NEWTON' and name = 'INVENTDB'
```

2. 如果未创建用于对优化概要文件高速缓存执行清仓的触发器，那么请发出 FLUSH OPTIMIZATION PROFILE CACHE 语句，以除去优化概要文件高速缓存中可能包含的任何优化概要文件版本。

   注: 对优化概要文件高速缓存执行清仓时，还将使动态方案高速缓存中任何使用旧优化概要文件准备的动态语句失效。

对此优化概要文件的任何后续引用都将导致优化器返回 SQL0437W（原因码为 13）。

优化准则:

优化准则的类型:

DB2 优化器分两个阶段来处理语句: 查询重写优化阶段和方案优化阶段。

已优化语句由*查询重写优化阶段*确定，此阶段将原始语句变换为语义上等价但更容易在方案优化阶段优化的语句。*方案优化阶段*通过枚举许多备用方案并选择使执行成本估计最低的备用方案来确定*已优化语句*的最佳访问方法、连接方法和连接顺序。

在这两个优化阶段期间考虑的查询变换、访问方法、连接方法、连接顺序和其他优化备用方法由各种 DB2 参数（例如 CURRENT QUERY OPTIMIZATION 专用寄存器、REOPT 绑定选项和 **DB2_REDUCED_OPTIMIZATION** 注册表变量）控制。这一组优化备用方法被称为*搜索空间*。

支持下列类型的语句优化准则:

- 首先应用*一般优化准则*，这些准则可能会影响搜索空间，并可用于影响一般优化参数的设置。
- 接着应用*查询重写准则*，这些准则可能会影响方案优化阶段优化的语句，并可用于影响查询重写优化阶段考虑的变换。
- 最后应用*方案优化准则*，这些准则可用于影响方案优化阶段考虑的访问方法、连接方法和连接顺序。

一般优化准则:

一般优化准则可用于设置一般优化参数。

其中的每个准则都具有语句级作用域。

查询重写优化准则:

查询重写准则可用于影响查询重写优化阶段考虑的变换，该阶段将原始语句变换为语义等同并且经过优化的语句。

然后，在方案优化阶段确定已优化语句的最佳执行方案。因此，查询重写优化准则可以影响方案优化准则的适用性。

每个查询重写优化准则都与优化器的其中一条查询变换规则相对应。查询重写优化准则可以影响下列查询变换规则:

- IN-LIST 到连接
- 子查询到连接
- NOT-EXISTS 子查询到反连接

- NOT-IN 子查询到反连接

查询重写优化准则并非始终适用。一次强制执行一条查询重写规则。因此，在后续规则之前实施的查询重写规则可以影响与该规则相关联的查询重写优化准则。环境配置可以影响某些重写规则的行为，这些规则又将影响查询重写优化准则是否适用于特定规则。

要每次都获得相同的结果，在实施查询重写规则之前，这些规则必须满足一些条件。如果查询重写组件尝试将规则应用于查询时未满足与该规则关联的条件，那么将忽略该规则的查询重写优化准则。如果查询重写优化准则不适用并且此准则是启用准则，那么将返回 SQL0437W（原因码为 13）。如果查询重写优化准则不适用并且此准则是禁用准则，那么不会返回消息。在这种情况下，不会应用此查询重写规则，因为此规则被视为已禁用。

可以将查询重写优化准则分为两种类别：语句级准则和谓词级准则。所有查询重写优化准则都支持语句级类别。仅 INLIST2JOIN 支持谓词级类别。语句级查询重写优化准则将应用于整个查询。谓词级查询重写优化准则仅应用于特定谓词。如果同时指定语句级和谓词级查询重写优化准则，那么谓词级准则将覆盖特定谓词的语句级准则。

每个查询重写优化准则都由优化准则模式中的相应重写请求元素表示。

以下示例说明 INLIST2JOIN 重写请求元素所表示的“IN-LIST 到连接”查询重写优化准则。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P_SIZE IN (35, 36, 39, 40)
  AND S.S_NATION IN ('INDIA', 'SPAIN')
  AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                          FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                          WHERE P.P_PARTKEY  = PS1.PS_PARTKEY AND
S1.S_SUPPKEY = PS1.PS_SUPPKEY
AND S1.S_NATION = S.S_NATION)
ORDER BY S.S_NAME
<OPTGUIDELINES><INLIST2JOIN TABLE='P'/></OPTGUIDELINES>;
```

这个特定的查询重写优化准则指定，应该将谓词 P_SIZE IN (35, 36, 39, 40) 中的常量列表变换为表表达式。于是，这个表表达式适合于驱动对主要子选择中的 PARTS 表进行的带索引嵌套循环连接访问。TABLE 属性用于指示此谓词所应用于的表引用，从而标识目标 IN-LIST 谓词。如果所标识的表引用有多个 IN-LIST 谓词，那么 INLIST2JOIN 重写请求元素将被认为不明确，并且将被忽略。

在此类情况下，可以添加 COLUMN 属性以便对目标 IN-LIST 谓词进行进一步限定。例如:

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P_SIZE IN (35, 36, 39, 40)
  AND P_TYPE IN ('BRASS', 'COPPER')
  AND S.S_NATION IN ('INDIA', 'SPAIN')
  AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                          FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                          WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                            AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
```

```
                              AND S1.S_NATION = S.S_NATION)
ORDER BY S.S_NAME
<OPTGUIDELINES><INLIST2JOIN TABLE='P' COLUMN='P_SIZE'/></OPTGUIDELINES>;
```

INLIST2JOIN 元素的 TABLE 属性用于标识主要子选择中的 PARTS 表引用。COLUMN 属性用于将作用于 P_SIZE 列的 IN-LIST 谓词标识为目标。通常，COLUMN 属性的值可以包含目标 IN-LIST 谓词中引用的列的非限定名称。如果在未提供 TABLE 属性的情况下提供 COLUMN 属性，那么查询重写优化准则将被视为无效并被忽略。

可以使用 OPTION 属性来启用或禁用特定的查询重写优化准则。在以下示例中，由于 OPTION 属性设置为 DISABLE，因此谓词 P_SIZE IN (35, 36, 39, 40) 中的常量列表将不会被变换为表表达式。OPTION 属性的缺省值是 ENABLE。ENABLE 和 DISABLE 都必须以大写字符指定。

```
<OPTGUIDELINES>  <INLIST2JOIN TABLE='P' COLUMN='P_SIZE' OPTION='DISABLE'/>
</OPTGUIDELINES>
```

在以下示例中，INLIST2JOIN 重写请求元素不具有 TABLE 属性。优化器将此解释为请求对语句中的所有 IN-LIST 谓词禁用“IN-LIST 到连接”查询变换。

```
<OPTGUIDELINES><INLIST2JOIN OPTION='DISABLE'/></OPTGUIDELINES>
```

以下示例说明 SUBQ2JOIN 重写请求元素所表示的“子查询到连接”查询重写优化准则。“子查询到连接”变换将子查询转换为等同的表表达式。此变换将应用于由 EXISTS、IN、=SOME、=ANY、<>SOME 或 <>ANY 限定的子查询谓词。“子查询到连接”查询重写优化准则并不确保合并子查询。此查询重写优化准则无法面向特定的子查询。只能在语句级别启用或禁用此变换。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P_SIZE IN (35, 36, 39, 40)
  AND P_TYPE = 'BRASS'
  AND S.S_NATION IN ('INDIA', 'SPAIN')
  AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
                          FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
                          WHERE P.P_PARTKEY = PS1.PS_PARTKEY
                            AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
                            AND S1.S_NATION = S.S_NATION)
ORDER BY S.S_NAME
<OPTGUIDELINES><SUBQ2JOIN OPTION='DISABLE'/></OPTGUIDELINES>;
```

以下示例说明 NOTEX2AJ 重写请求元素所表示的“NOT-EXISTS 到反连接”查询重写优化准则。“NOT-EXISTS 到反连接”变换将子查询转换为一个表表达式，该表达式使用反连接语义来连接到其他表（只返回不匹配的行）。“NOT-EXISTS 到反连接”查询重写优化准则将应用于由 NOT EXISTS 限定的子查询谓词。“NOT-EXISTS 到反连接”查询重写优化准则并不确保合并子查询。此查询重写优化准则无法面向特定的子查询。只能在语句级别启用或禁用此变换。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P_SIZE IN (35, 36, 39, 40)
  AND P_TYPE = 'BRASS'
  AND S.S_NATION IN ('INDIA', 'SPAIN')
  AND NOT EXISTS (SELECT 1
                  FROM "Tpcd".SUPPLIERS S1
```

```
                    WHERE S1.S_SUPPKEY = PS.PS_SUPPKEY)
ORDER BY S.S_NAME
<OPTGUIDELINES><NOTEX2AJ OPTION='ENABLE'/></OPTGUIDELINES>;
```

注: 在语句级别启用查询变换规则并不会确保将该规则应用于该语句的特定部分。将使用常规标准来确定是否进行查询变换。例如，如果查询块包含多个 NOT EXISTS 谓词，那么优化器将不会考虑将任何这些谓词转换为反连接。在语句级别显式地启用查询变换并不会更改此行为。

以下示例说明 NOTIN2AJ 重写请求元素所表示的“NOT-IN 到反连接”查询重写优化准则。“NOT-IN 到反连接”变换将子查询转换为一个表表达式，该表达式使用反连接语义来连接到其他表（只返回不匹配的行）。“NOT-IN 到反连接”查询重写优化准则将应用于由 NOT IN 限定的子查询谓词。“NOT-IN 到反连接”查询重写优化准则并不确保合并子查询。此查询重写优化准则无法面向特定的子查询。只能在语句级别启用或禁用此变换。

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY
  AND S.S_SUPPKEY = PS.PS_SUPPKEY
  AND P_SIZE IN (35, 36, 39, 40)
  AND P_TYPE = 'BRASS'
  AND S.S_NATION IN ('INDIA', 'SPAIN')
  AND PS.PS_SUPPKEY NOT IN (SELECT S1.S_SUPPKEY
                            FROM "Tpcd".SUPPLIERS S1
                            WHERE S1.S_NATION = 'CANADA')
ORDER BY S.S_NAME
<OPTGUIDELINES><NOTIN2AJ OPTION='ENABLE'/></OPTGUIDELINES>
```

在应用于语句的其他查询重写变换的上下文中考虑特定的查询重写优化准则时，该准则可能不适用。即，如果无法应用要启用某个变换的准则请求，那么将返回警告。例如，如果 INLIST2JOIN 重写启用请求元素面向由另一个查询变换从查询中消除的谓词，那么此元素不适用。并且，成功地应用一个查询重写优化准则可能会更改其他查询重写变换规则的适用性。例如，请求将 IN-LIST 变换为表表达式可能会导致无法将另一个 IN-LIST 变换为表表达式。这是因为，对于每个查询块，优化器只应用单一“IN-LIST 到连接”变换。

方案优化准则:

方案优化准则将在基于成本的优化阶段进行应用，在此阶段，将确定语句的访问方法、连接方法、连接顺序和其他执行方案详细信息。

方案优化准则不需要指定执行方案的所有方面。未指定的执行方案方面由优化器以基于成本的方式确定。

方案优化准则分为两类:

- accessRequest - 这是一个访问请求，它指定用于满足语句中的表引用的访问方法。
- joinRequest - 这是一个连接请求，它指定用于执行连接操作的方法和顺序。连接请求由访问请求或其他连接请求组成。

访问请求优化准则与优化器的数据访问方法（例如表扫描、索引扫描和列表预取）相对应。连接请求准则与优化器的连接方法（例如嵌套循环连接、散列连接以及合并连接）相对应。每个访问请求和连接请求都由语句优化准则模式中相应的访问请求元素和连接请求元素表示。

以下示例说明 IXSCAN 访问请求元素所表示的索引扫描访问请求。这个特定的请求指定，优化器将使用 I_SUPPKEY 索引来访问语句的主要子选择中的 SUPPLIERS 表。可选的 INDEX 属性用于标识期望的索引。TABLE 属性用于标识此访问请求所应用于的表引用。TABLE 属性必须使用目标表引用的显示名来标识该引用，在本例中，这是相关名 S。

SQL 语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
  where p_partkey = ps.ps_partkey and
    s.s_suppkey = ps.ps_suppkey and
    p_size = 39 and
    p_type = 'BRASS' and
    s.s_nation in ('MOROCCO', 'SPAIN') and
    ps.ps_supplycost = (select min(ps1.ps_supplycost)
                          from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                          where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                          s1.s_suppkey = ps1.ps_suppkey and
                          s1.s_nation = s.s_nation)
  order by s.s_name
```

优化准则:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SUPPKEY'/>
</OPTGUIDELINES>
```

以下索引扫描访问请求元素指定，对于语句的主要子选择中的 PARTS 表，优化器将使用索引访问方法。由于未指定 INDEX 属性，因此优化器将以基于成本的方式来选择索引。由于没有相关联的相关名，因此 TABLE 属性使用限定的表名来引用目标表引用。

```
<OPTGUIDELINES>
  <IXSCAN TABLE='"Tpcd".PARTS'/>
</OPTGUIDELINES>
```

以下列表预取访问请求由 LPREFETCH 访问请求元素表示。这个特定的请求指定，优化器将使用 I_SNATION 索引来访问语句的嵌套子选择中的 SUPPLIERS 表。TABLE 属性使用相关名 S1，这是因为，此相关名是标识了嵌套子选择中的 SUPPLIERS 表引用的显示名。

```
<OPTGUIDELINES>
  <LPREFETCH TABLE='S1' INDEX='I_SNATION'/>
</OPTGUIDELINES>
```

以下索引扫描访问请求元素指定，优化器将使用 I_SNAME 索引来访问主要子选择中的 SUPPLIERS 表。FIRST 属性指定，此表将是为相应 FROM 子句选择的连接序列中访问的第一个表。可以对任何访问或连接请求添加 FIRST 属性；但是，最多只能有一个访问或连接请求的 FIRST 属性引用同一个 FORM 子句中的表。

SQL 语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
  where p_partkey = ps.ps_partkey
    s.s_suppkey = ps.ps_suppkey and
    p_size = 39 and
    p_type = 'BRASS' and
    s.s_nation in ('MOROCCO', 'SPAIN') and
    ps.ps_supplycost = (select min(ps1.ps_supplycost)
                          from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                          where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                            s1.s_suppkey = ps1.ps_suppkey and
```

```
                                s1.s_nation = s.s_nation)
  order by s.s_name
  optimize for 1 row
```

优化准则:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SNAME' FIRST='TRUE'/>
</OPTGUIDELINES>
```

以下示例说明如何在单一语句优化准则中传递多个访问请求。TBSCAN 访问请求元素表示表扫描访问请求。这个特定的请求指定，将使用全表扫描方法来访问嵌套子选择中的 SUPPLIERS 表。LPREFETCH 访问请求元素指定，优化器在对主要子选择中的 SUPPLIERS 表进行列表预取索引访问期间，将使用 I_SUPPKEY 索引。

```
<OPTGUIDELINES>
  <TBSCAN TABLE='S1'/>
  <LPREFETCH TABLE='S' INDEX='I_SUPPKEY'/>
</OPTGUIDELINES>
```

以下示例说明 NLJOIN 连接请求元素所表示的嵌套循环连接请求。通常，连接请求元素包含两个子元素。第一个子元素表示连接操作的期望外输入，第二个子元素表示连接操作的期望内输入。子元素可以是访问请求、其他连接请求或者访问请求与连接请求的组合。在本例中，第一个 IXSCAN 访问请求元素指定，主要子选择中的 PARTS 表将是连接操作的外表。它还指定使用索引扫描方法来执行 PARTS 表访问。第二个 IXSCAN 访问请求元素指定，主要子选择中的 PARTSUPP 表将是连接操作的内表。它也指定使用索引扫描方法来访问该表。

```
<OPTGUIDELINES>
  <NLJOIN>
    <IXSCAN TABLE='"Tpcd".Parts'/>
    <IXSCAN TABLE="PS"/>
  </NLJOIN>
</OPTGUIDELINES>
```

以下示例说明 HSJOIN 连接请求元素所表示的散列连接请求。ACCESS 访问请求元素指定，嵌套子选择中的 SUPPLIERS 表将是连接操作的外表。在主要目标是指定连接顺序时，此访问请求元素十分有用。IXSCAN 访问请求元素指定，嵌套子选择中的 PARTSUPP 表将是连接操作的内表，并且优化器将选择索引扫描方法来访问该表。

```
<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1'/>
    <IXSCAN TABLE='PS1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

以下示例说明如何通过对连接请求进行嵌套来构造较大的连接请求。此示例包含由 MSJOIN 连接请求元素表示的合并连接请求。连接操作的外输入是对主要子选择的 PARTS 和 PARTSUPP 表进行连接（由 NLJOIN 连接请求元素表示）的结果。连接请求元素的内输入是主要子选择中的 SUPPLIERS 表（由 IXSCAN 访问请求元素表示）。

```
<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='"Tpcd".Parts'/>
      <IXSCAN TABLE="PS"/>
    </NLJOIN>
    <IXSCAN TABLE='S'/>
  </MSJOIN>
</OPTGUIDELINES>
```

如果连接请求将有效，那么直接或间接嵌套在其中的所有访问请求元素都必须引用优化后的语句中同一个 FROM 子句中的表。

创建语句级优化准则:

语句概要文件的语句级优化准则节由一个或多个访问请求或连接请求构成，这些请求指定语句中访问和连接表的方法。

使用所有其他调整选项。例如:

1. 确保 RUNSTATS 实用程序最近更新了数据分布统计信息。
2. 确保数据服务器正在运行，并且对工作负载设置了正确的优化类。
3. 确保优化器有适当的索引来访问查询中引用的表。

要创建语句级优化准则，请执行下列操作:

1. 创建优化概要文件，以便在其中插入语句级准则。
2. 对该语句运行说明工具，以确定优化准则是否有用。如果情况似乎如此，那么继续。
3. 通过运行类似于以下的查询，获取*原始*语句:

```
select statement_text
  from explain_statement
  where explain_level = '0' and
    explain_requester = 'SIMMEN' and
    explain_time      = '2003-09-08-16.01.04.108161' and
    source_name       = 'SQLC2E03' and
    source_version    = '' and
    queryno           = 1
```

4. 编辑优化概要文件并创建语句概要文件，并在语句键中插入语句文本。例如:

```
<STMTPROFILE ID="Guidelines for TPCD Q9">
   <STMTKEY SCHEMA="TPCD"><![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE,
     S.S_COMMENT
     FROM PARTS P, SUPPLIERS S, PARTSUPP PS
     WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY
     AND P.P_SIZE  = 39 AND P.P_TYPE  = 'BRASS' AND S.S_NATION
     = 'MOROCCO' AND
     PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST)
     FROM PARTSUPP PS1, SUPPLIERS S1
     WHERE P.P_PARTKEY  = PS1.PS_PARTKEY
     AND S1.S_SUPPKEY = PS1.PS_SUPPKEY
     AND S1.S_NATION  = S.S_NATION)]]>
   </STMTKEY>
</STMTPROFILE>
```

5. 在语句键后面插入语句级优化准则。使用显示名来标识访问请求和连接请求中引用的对象。以下是连接请求的示例:

```
<OPTGUIDELINES>
  <HSJOIN>
    <TBSCAN TABLE='PS1'/>
    <IXSCAN TABLE='S1'
      INDEX='I1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

6. 验证并保存文件。

如果没有达到预期的结果，请更改准则或创建其他准则，并适当地更新优化概要文件。

在优化准则中构造表引用:

术语*表引用*是指任何表、视图、表表达式或者是别名在 SQL 语句或视图定义中引用的表。优化准则可以使用原始语句中表引用的显示名或者与已优化语句中的表引用相关联的唯一相关名来标识表引用。

扩展名（即显示名的序列）用来帮助唯一地标识视图中嵌入的表引用。不能将别名用作优化准则中的表引用；在这种情况下，将忽略所有以表引用作为目标的准则。如果优化准则所标识的显示名或扩展名在整个语句的上下文中不是唯一的，那么认为这些优化准则具有二义性而不会应用。此外，如果多个优化准则标识相同的表引用，那么认为标识表引用的所有优化准则相冲突，将不应用这些优化准则。查询变换的可能性导致无法保证显示名或扩展名在优化期间仍存在；因此，将忽略任何标识了此类表引用的准则。

使用原始语句中的显示名来标识表引用

表引用由表的显示名标识。指定显示名的方式与在 SQL 语句中限定表的方式相同。

用于指定 SQL 标识的规则也适用于优化准则的 TABLE 属性值。TABLE 属性值相当于语句中的每个显示名。在此 DB2 发行版中，只允许单个匹配项。如果 TABLE 属性值由模式限定，那么它与任何等价的显示限定表名匹配。如果 TABLE 属性值未被限定，那么它与任何等价的相关名或显示表名匹配。因此，认为 TABLE 属性值由语句的有效缺省模式隐式限定。以下示例对这些概念作了说明。假定使用缺省模式 Tpcd 来优化语句。

```
select s_name, s_address, s_phone, s_comment
  from parts, suppliers, partsupp ps
  where p_partkey = ps.ps_partkey and
    s.s_suppkey = ps.ps_suppkey and
    p_size = 39 and
    p_type = 'BRASS'
```

用于标识语句中的表引用的 TABLE 属性值包括“″Tpcd″.PARTS”、“PARTS”和“Parts”（因为标识未定界，所以它将转换为大写字符）。无法标识语句中的表引用的 TABLE 属性值包括“″Tpcd2″.SUPPLIERS”、“PARTSUPP”（不是显示名）和“Tpcd.PARTS”（必须定界标识 Tpcd；否则它将转换为大写字符）。

可以使用显示名来以原始语句、视图、SQL 函数或触发器中的任何表引用为目标。

使用原始语句中的显示名来标识视图中的表引用

优化准则可以使用扩展语法来标识视图中嵌入的表引用，如以下示例所示:

```
create view "Rick".v1 as
  (select * from employee a where salary > 50000)

create view "Gustavo".v2 as
  (select * from "Rick".v1
     where deptno in ('52', '53', '54')

select * from "Gustavo".v2 a
  where v2.hire_date > '01/01/2004'

<OPTGUIDELINES>
  <IXSCAN TABLE='A/"Rick".V1/A'/>
</OPTGUIDELINES>
```

IXSCAN 访问请求元素指定，对于视图 ″Gustavo″.V2 和 ″Rick″.V1 中嵌入的 EMPLOYEE 表引用，使用索引扫描方法。用于在视图中标识表引用的扩展语法是一系列由斜杠字符分隔的显示名。TABLE 属性值 A/″Rick″.V1/A 说明了此扩展语法。序列中的最后一个显示名（A）用于标识作为优化准则目标的表引用。序列中的第一个显示名（A）用于标识原始语句中直接引用的视图。中间的显示名（″Rick″.V1）用于标识视图引用以及从直接视图引用到目标表引用的路径。有关从优化准则中引用显示名的规则（如上一节所述）适用于扩展语法的每个步骤。

如果视图中 EMPLOYEE 表引用的显示名对于语句直接或间接引用的所有表而言都唯一，那么没必要采用扩展名称语法。

可以使用扩展语法来以原始语句、SQL 函数或触发器中的任何表引用为目标。

使用已优化语句中的相关名来标识表引用

优化准则还可以使用与已优化语句中的表引用关联的唯一相关名来标识表引用。已优化语句是语义上与原始语句等价的语句，它由查询重写优化阶段确定。可以从说明表中检索已优化的语句。优化准则的 TABID 属性用于标识已优化的语句中的表引用。例如:

原始语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from sm_tpcd.parts p, sm_tpcd.suppliers s, sm_tpcd.partsupp ps
  where p_partkey = ps.ps_partkey and
    s.s_suppkey = ps.ps_suppkey and
    p.p_size = 39 and
    p.p_type = 'BRASS' and
    s.s_nation in ('MOROCCO', 'SPAIN') and
    ps.ps_supplycost = (select min(ps1.ps_supplycost)
                          from sm_tpcd.partsupp ps1, sm_tpcd.suppliers s1
                          where p.p_partkey = ps1.ps_partkey and
                            s1.s_suppkey = ps1.ps_suppkey and
                            s1.s_nation = s.s_nation)

<OPTGUIDELINES>
  <HSJOIN>
    <TBSCAN TABLE='S1'/>
    <IXSCAN TABID='Q2'/>
  </HSJOIN>
</OPTGUIDELINES>
```

优化后的语句:

```
select q6.s_name as "S_NAME", q6.s_address as "S_ADDRESS",
    q6.s_phone as "S_PHONE", q6.s_comment as "S_COMMENT"
  from (select min(q4.$c0)
          from (select q2.ps_supplycost
                  from sm_tpcd.suppliers as q1, sm_tpcd.partsupp as q2
                  where q1.s_nation = 'MOROCCO' and
                    q1.s_suppkey = q2.ps_suppkey and
                    q7.p_partkey = q2.ps_partkey
                  ) as q3
          ) as q4, sm_tpcd.partsupp as q5, sm_tpcd.suppliers as q6,
              sm_tpcd.parts as q7
  where p_size = 39 and
    q5.ps_supplycost = q4.$c0 and
    q6.s_nation in ('MOROCCO', 'SPAIN') and
    q7.p_type = 'BRASS' and
    q6.s_suppkey = q5.ps_suppkey and
    q7.p_partkey = q5.ps_partkey
```

此优化准则显示了一个散列连接请求，其中，嵌套子选择中的 SUPPLIERS 表是外表（由 TBSCAN 访问请求元素指定），而嵌套子选择中的 PARTSUPP 表是内表（由 IXSCAN 访问请求元素指定）。通过适用原始语句中相应的显示名，TBSCAN 访问请求元素使用 TABLE 属性来标识 SUPPLIERS 表引用。另一方面，通过使用与优化语句中的表引用相关联的唯一相关名，IXSCAN 访问请求元素使用 TABID 属性来标识 PARTSUPP 表引用。

如果单个优化准则同时指定 TABLE 和 TABID 属性，那么它们必须标识相同的表引用或者将忽略此优化准则。

注: 当前无法保证在升级到新发行版的 DB2 产品时已优化的语句中的相关名将保持稳定状态。

模糊表引用

如果某个优化准则与多个显示名或扩展名匹配，那么认为此优化准则无效而不会应用。例如:

```
create view v1 as
  (select * from employee
     where salary > (select avg(salary) from employee)

select * from v1
  where deptno in ('M62', 'M63')

<OPTGUIDE>
  <IXSCAN TABLE='V1/EMPLOYEE'/>
</OPTGUIDE>
```

优化器认为 IXSCAN 访问请求不明确，这是因为显示名 EMPLOYEE 在视图 V1 的定义中不唯一。

要消除歧义，可以重写视图以使用唯一相关名，也可以使用 TABID 属性。由 TABID 属性标识的表引用决不会是模糊的，因为已优化的语句中的所有相关名都是唯一的。

相冲突的优化准则

多个优化准则不能标识相同的表引用。例如:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_PTYPE'/>
  <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_SIZE'/>
</OPTGUIDELINES>
```

每个 IXSCAN 元素都引用主要子选择中的 ″Tpcd″.PARTS 表。

当两个或更多准则引用同一个表时，仅应用第一个准则；将忽略所有其他准则并返回错误。

对于每个查询，在谓词级只能启用一个 INLIST2JOIN 查询重写请求元素。以下示例演示了不受支持的查询重写优化准则，此准则在谓词级别启用了两个 IN-LIST 谓词。这两个准则都将被忽略，并且将返回警告。

```
<OPTGUIDELINES>
  <INLIST2JOIN TABLE='P' COLUMN='P_SIZE'/>
 <INLIST2JOIN TABLE='P' COLUMN='P_TYPE'/>
</OPTGUIDELINES>
```

验证是否已使用优化准则:

优化器尽最大可能遵循优化概要文件中指定的优化准则；但是，优化器可以拒绝无效或不适用的准则。

在可以使用说明工具之前，说明表必须已存在。用于创建说明表的数据定义语言（DDL）包含在 EXPLAIN.DDL 中，此文件在 sqllib 目录的 misc 子目录中。

要验证是否已使用有效的优化准则，请执行下列操作:

1. 对那些准则所应用于的语句发出 EXPLAIN 语句。 如果存在作用于使用优化概要文件的语句的优化准则，那么优化概要文件名将作为 RETURN 运算符参数出现在 EXPLAIN_ARGUMENTS 表中。并且，如果优化准则包含与当前语句匹配的 SQL 嵌入式优化准则或语句概要文件，那么该语句概要文件的名称将作为 RETURN 运算符参数出现。这两个新参数值的类型为 OPT_PROF 和 STMTPROF。
2. 检查所说明的语句的结果。 可以修改下列针对说明表的查询以返回优化概要文件名以 及
   EXPLAIN_REQUESTER、EXPLAIN_TIME、SOURCE_NAME、SOURCE_VERSION 与 QUERYNO 的特定组合的语句概要文件名:

```
SELECT VARCHAR(B.ARGUMENT_TYPE,   9) as TYPE,
     VARCHAR(B.ARGUMENT_VALUE, 24) as VALUE

FROM   EXPLAIN_STATEMENT A, EXPLAIN_ARGUMENT B

WHERE  A.EXPLAIN_REQUESTER = 'SIMMEN'
  AND  A.EXPLAIN_TIME      = '2003-09-08-16.01.04.108161'
  AND  A.SOURCE_NAME       = 'SQLC2E03'
  AND  A.SOURCE_VERSION    = ''
  AND  A.QUERYNO           = 1

  AND  A.EXPLAIN_REQUESTER = B.EXPLAIN_REQUESTER
  AND  A.EXPLAIN_TIME      = B.EXPLAIN_TIME
  AND  A.SOURCE_NAME       = B.SOURCE_NAME
  AND  A.SOURCE_SCHEMA     = B.SOURCE_SCHEMA
  AND  A.SOURCE_VERSION    = B.SOURCE_VERSION
  AND  A.EXPLAIN_LEVEL     = B.EXPLAIN_LEVEL
  AND  A.STMTNO            = B.STMTNO
  AND  A.SECTNO            = B.SECTNO

  AND  A.EXPLAIN_LEVEL     = 'P'

  AND  (B.ARGUMENT_TYPE  = 'OPT_PROF' OR ARGUMENT_TYPE = 'STMTPROF')
  AND  B.OPERATOR_ID = 1
```

   如果优化准则处于活动状态，并且所说明的语句与优化准则的 STMTKEY 元素中包含的语句相匹配，那么与先前示例类似的查询将生成类似于以下的输出。STMTPROF 参数的值与 STMTPROFILE 元素中 ID 属性的值相同。

```
TYPE      VALUE
--------- -------------------------
OPT_PROF  NEWTON.PROFILE1
STMTPROF  Guidelines for TPCD Q9
```

优化概要文件和准则的 XML 模式:

当前优化概要文件模式:

给定 DB2 发行版的有效优化概要文件内容由称为当前优化概要文件模式（COPS）的 XML 模式描述。优化概要文件仅适用于 DB2 数据库 Linux 版、UNIX 版和 Windows 版服务器。

以下列表代表当前 DB2 产品发行版的 COPS。还可以在 DB2OptProfile.xsd 中（它位于 sqllib 目录的 misc 子目录下）找到 COPS。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" version="1.0">
<!--*******************************************************************************************-->
<!-- Licensed Materials - Property of IBM                                                      -->
<!-- (C) Copyright International Business Machines Corporation 2009. All rights reserved.    -->
<!-- U.S. Government Users Restricted Rights; Use, duplication or disclosure restricted by   -->
<!-- GSA ADP Schedule Contract with IBM Corp.                                                  -->
<!--*******************************************************************************************-->
<!--*******************************************************************************************-->
<!-- Definition of the current optimization profile schema for V9.7.0.0                        -->
<!--                                                                                           -->
<!-- An optimization profile is composed of the following sections:                            -->
<!--                                                                                           -->
<!-- + A global optimization guidelines section (at most one) which defines optimization     -->
<!-- guidelines affecting any statement for which the optimization profile is in effect.     -->
<!--                                                                                           -->
<!-- + Zero or more statement profile sections, each of which defines optimization           -->
<!-- guidelines for a particular statement for which the optimization profile                -->
<!-- is in effect.                                                                             -->
<!--                                                                                           -->
<!-- The VERSION attribute indicates the version of this optimization profile                -->
<!-- schema.                                                                                   -->
<!--*******************************************************************************************-->
<xs:element name="OPTPROFILE">
   <xs:complexType>
      <xs:sequence>
         <!-- Global optimization guidelines section. At most one can be specified. -->
         <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType" minOccurs="0"/>
         <!-- Statement profile section. Zero or more can be specified -->
         <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- Version attribute is currently optional -->
      <xs:attribute name="VERSION" use="optional"/>
   </xs:complexType>
</xs:element>

<!--*******************************************************************************************-->
<!-- Global optimization guidelines supported in this version:                                 -->
<!-- + MQTOptimizationChoices elements influence the MQTs considered by the optimizer.       -->
<!-- + computationalPartitionGroupOptimizationsChoices elements can affect repartitioning    -->
<!-- optimizations involving nicknames.                                                        -->
<!-- + General requests affect the search space which defines the alternative query          -->
<!-- transformations, access methods, join methods, join orders, and other optimizations,    -->
<!-- considered by the compiler and optimizer.                                                 -->
<!-- + MQT enforcement requests specify semantically matchable MQTs whose usage in access    -->
<!-- plans should be enforced regardless of cost estimates.                                   -->
<!-- ******************************************************************************************-->
<xs:complexType name="globalOptimizationGuidelinesType">
   <xs:sequence>
      <xs:group ref="MQTOptimizationChoices" />
      <xs:group ref="computationalPartitionGroupOptimizationChoices" />
      <xs:group ref="generalRequest"/>
      <xs:group ref="mqtEnforcementRequest" />
   </xs:sequence>
   </xs:complexType><!-- ************************************************************************************************-->
<!-- Elements for affecting materialized query table (MQT) optimization.                       -->
<!--                                                                                           -->
<!-- + MQTOPT - can be used to disable materialized query table (MQT) optimization.          -->
<!-- If disabled, the optimizer will not consider MQTs to optimize the statement.            -->
<!--                                                                                           -->
<!-- + MQT - multiple of these can be specified. Each specifies an MQT that should be        -->
<!-- considered for optimizing the statement. Only specified MQTs will be considered.        -->
<!--                                                                                           -->
<!--*******************************************************************************************-->
<xs:group name="MQTOptimizationChoices">
   <xs:choice>
      <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
```

```
            </xs:complexType>      </xs:element>
         <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
               <xs:attribute name="NAME" type="xs:string" use="required"/>
               </xs:complexType>      </xs:element>
      </xs:choice>
   </xs:group>
   <!-- ********************************************************************************************-->
   <!-- Elements for affecting computational partition group (CPG) optimization.                    -->
   <!--                                                                                             -->
   <!-- + PARTOPT - can be used disable the computational partition group (CPG) optimization        -->
   <!-- which is used to dynamically redistributes inputs to join, aggregation,                     -->
   <!-- and union operations when those inputs are results of remote queries.                       -->
   <!--                                                                                             -->
   <!-- + PART - Define the partition groups to be used in CPG optimizations.                       -->
   <!--                                                                                             -->
   <!-- ********************************************************************************************-->
   <xs:group name="computationalPartitionGroupOptimizationChoices">
      <xs:choice>
         <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
            <xs:complexType>
               <xs:attribute name="OPTION" type="optionType" use="optional"/>
            </xs:complexType>      </xs:element>
         <xs:element name="PART" minOccurs="0" maxOccurs="1">
            <xs:complexType>
               <xs:attribute name="NAME" type="xs:string" use="required"/>
               </xs:complexType>      </xs:element>
      </xs:choice>
   </xs:group>
   <!-- ********************************************************************************************-->
   <!-- Definition of a statement profile.                                                           -->
   <!-- Comprised of a statement key and optimization guidelines.                                    -->
   <!-- The statement key specifies semantic information used to identify the statement to           -->
   <!-- which optimization guidelines apply. The optional ID attribute provides the statement       -->
   <!-- profile with a name for use in EXPLAIN output.                                               -->
   <!-- ********************************************************************************************-->
   <xs:complexType name="statementProfileType">
      <xs:sequence>
         <!-- Statement key element -->
         <xs:element name="STMTKEY" type="statementKeyType"/>
         <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
      </xs:sequence>
      <!-- ID attribute.Used in explain output to indicate the statement profile was used. -->
      <xs:attribute name="ID" type="xs:string" use="optional"/>
      </xs:complexType><!--*************************************************************************************************-->
   <!-- Definition of the statement key. The statement key provides semantic information used       -->
   <!-- to identify the statement to which the optimization guidelines apply.                       -->
   <!-- The statement key is comprised of:                                                          -->
   <!-- + statement text (as written in the application)                                            -->
   <!-- + default schema (for resolving unqualified table names in the statement)                   -->
   <!-- + function path (for resolving unqualified types and functions in the statement)            -->
   <!-- The statement text is provided as element data whereas the default schema and function      -->
   <!-- path are provided via the SCHEMA and FUNCPATH elements, respectively.                       -->
   <!--********************************************************************************************-->
   <xs:complexType name="statementKeyType" mixed="true">
      <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
      <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
      </xs:complexType>
   <!--********************************************************************************************-->
   <!--                                                                                             -->
   <!-- Optimization guideline elements can be chosen from general requests, rewrite               -->
   <!-- requests access requests, or join requests.                                                 -->
   <!--                                                                                             -->
   <!-- General requests affect the search space which defines the alternative query                -->
   <!-- transformations, access methods, join methods, join orders, and other optimizations,        -->
   <!-- considered by the optimizer.                                                                -->
   <!--                                                                                             -->
   <!-- Rewrite requests affect the query transformations used in determining the optimized         -->
   <!-- statement.                                                                                  -->
   <!--                                                                                             -->
   <!-- Access requests affect the access methods considered by the cost-based optimizer,          -->
   <!-- and join requests affect the join methods and join order used in the execution plan.        -->
   <!--                                                                                             -->
   <!-- MQT enforcement requests specify semantically matchable MQTs whose usage in access          -->
   <!-- plans should be enforced regardless of cost estimates.                                      -->
   <!--                                                                                             -->
   <!--********************************************************************************************-->
   <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
   <xs:complexType name="optGuidelinesType">
```

```
   <xs:sequence>
      <xs:group ref="generalRequest" minOccurs="0" maxOccurs="1"/>
      <xs:choice maxOccurs="unbounded">
         <xs:group ref="rewriteRequest" />
         <xs:group ref="accessRequest"/>
         <xs:group ref="joinRequest"/>
         <xs:group ref="mqtEnforcementRequest"/>
      </xs:choice>
   </xs:sequence>
   </xs:complexType><!--*****************************************************************************************  -->
<!-- Choices of general request elements.                                                                    -->
<!-- REOPT can be used to override the setting of the REOPT bind option.                                     -->
<!-- DPFXMLMOVEMENT can be used to affect the optimizer's plan when moving XML documents                     -->
<!-- between database partitions. The value can be NONE, REFERENCE or COMBINATION. The                       -->
<!-- default value is NONE.                                                                                  -->
<!--***************************************************************************************** -->
<xs:group name="generalRequest">
   <xs:sequence>
      <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DPFXMLMOVEMENT" type="dpfXMLMovementType" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:group>
<!--*****************************************************************************************-->
<!-- Choices of rewrite request elements.                                                  -->
<!--*****************************************************************************************-->
<xs:group name="rewriteRequest">
   <xs:sequence>
      <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
      <xs:element name="SUBQ2JOIN" type="subqueryToJoinType" minOccurs="0"/>
      <xs:element name="NOTEX2AJ" type="notExistsToAntiJoinType" minOccurs="0"/>
      <xs:element name="NOTIN2AJ" type="notInToAntiJoinType" minOccurs="0"/>
   </xs:sequence>
</xs:group>
<!--***************************************************************************************** -->
<!-- Choices for access request elements.                                                    -->
<!-- TBSCAN - table scan access request element                                              -->
<!-- IXSCAN - index scan access request element                                              -->
<!-- LPREFETCH - list prefetch access request element                                        -->
<!-- IXAND - index ANDing access request element                                             -->
<!-- IXOR - index ORing access request element                                               -->
<!-- XISCAN - xml index access request element                                               -->
<!-- XANDOR - XANDOR access request element                                                  -->
<!-- ACCESS - indicates the optimizer should choose the access method for the table          -->
<!--***************************************************************************************** -->
<xs:group name="accessRequest">
   <xs:choice>
      <xs:element name="TBSCAN" type="tableScanType"/>
      <xs:element name="IXSCAN" type="indexScanType"/>
      <xs:element name="LPREFETCH" type="listPrefetchType"/>
      <xs:element name="IXAND" type="indexAndingType"/>
      <xs:element name="IXOR" type="indexOringType"/>
      <xs:element name="XISCAN" type="indexScanType"/>
      <xs:element name="XANDOR" type="XANDORType"/>
      <xs:element name="ACCESS" type="anyAccessType"/>
   </xs:choice>
</xs:group>
<!--***************************************************************************************** -->
<!-- Choices for join request elements.                                                      -->
<!-- NLJOIN - nested-loops join request element                                              -->
<!-- MSJOIN - sort-merge join request element                                                -->
<!-- HSJOIN - hash join request element                                                      -->
<!-- JOIN - indicates that the optimizer is to choose the join method.                       -->
<!--***************************************************************************************** -->
<xs:group name="joinRequest">
   <xs:choice>
      <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
      <xs:element name="HSJOIN" type="hashJoinType"/>
      <xs:element name="MSJOIN" type="mergeJoinType"/>
      <xs:element name="JOIN" type="anyJoinType"/>
   </xs:choice>
</xs:group>
<!--*****************************************************************************************-->
<!-- MQT enforcement request element.                                                      -->
<!-- MQTENFORCE - This element can be used to specify semantically matchable MQTs whose  -->
<!--   usage in access plans should be enforced regardless of Optimizer cost estimates.  -->
<!--   MQTs can be specified either directly with the NAME attribute or generally using  -->
```

```
<!--   the TYPE attribute.                                                          -->
<!--   Only the first valid attribute found is used and all subsequent ones are ignored. -->
<!--   Since this element can be specified multiple times, more than one MQT can be      -->
<!--   enforced at a time.                                                          -->
<!--   Note however, that if there is a conflict when matching two enforced MQTs to the  -->
<!--   same data source (base-table or derived) an MQT will be picked based on existing  -->
<!--   tie-breaking rules, i.e., either heuristic or cost-based.                    -->
<!--   Finally, this request overrides any other MQT optimization options specified in   -->
<!--   a profile, i.e., enforcement will take place even if MQTOPT is set to DISABLE or  -->
<!--   if the specified MQT or MQTs do not exist in the eligibility list specified by    -->
<!--   any MQT elements.                                                            -->
<!--*******************************************************************************-->
<xs:group name="mqtEnforcementRequest">
   <xs:sequence>
      <xs:element name="MQTENFORCE" type="mqtEnforcementType" minOccurs="0" maxOccurs="unbounded"/>
   </xs:sequence>
</xs:group>
<!--******************************************************************************* -->
<!-- REOPT general request element. Can override REOPT setting at the package, db,            -->
<!-- dbm level.                                                                   -->
<!--******************************************************************************* -->
<xs:complexType name="reoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ONCE"/>
            <xs:enumeration value="ALWAYS"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   </xs:complexType><!--*************************************************************************************************-->
<!-- RTS general request element to enable, disable or provide a time budget for              -->
<!-- real-time statistics collection.                                                          -->
<!-- OPTION attribute allows enabling or disabling real-time statistics.                       -->
<!-- TIME attribute provides a time budget in milliseconds for real-time statistics collection.-->
<!--*******************************************************************************-->
<xs:complexType name="rtsType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of an "IN list to join" rewrite request                                -->
<!-- OPTION attribute allows enabling or disabling the alternative.                      -->
<!-- TABLE attribute allows request to target IN list predicates applied to a            -->
<!-- specific table reference. COLUMN attribute allows request to target a specific IN list   -->
<!-- predicate.                                                                          -->
<!--*******************************************************************************-->
<xs:complexType name="inListToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="TABLE" type="xs:string" use="optional"/>
   <xs:attribute name="COLUMN" type="xs:string" use="optional"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of a "subquery to join" rewrite request                                -->
<!-- The OPTION attribute allows enabling or disabling the alternative.                  -->
<!--*******************************************************************************-->
<xs:complexType name="subqueryToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of a "not exists to anti-join" rewrite request                         -->
<!-- The OPTION attribute allows enabling or disabling the alternative.                  -->
<!--*******************************************************************************-->
<xs:complexType name="notExistsToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of a "not IN to anti-join" rewrite request                             -->
<!-- The OPTION attribute allows enabling or disabling the alternative.                  -->
<!--*******************************************************************************-->
<xs:complexType name="notInToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional" default="ENABLE"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Effectively the superclass from which all access request elements inherit.          -->
<!-- This type currently defines TABLE and TABID attributes, which can be used to tie an      -->
<!-- access request to a table reference in the query.                                   -->
<!-- The TABLE attribute value is used to identify a table reference using identifiers        -->
<!-- in the original SQL statement. The TABID attribute value is used to identify a table     -->
<!-- referece using the unique correlation name provided via the                         -->
<!-- optimized statement. If both the TABLE and TABID attributes are specified, the TABID     -->
<!-- field is ignored. The FIRST attribute indicates that the access should be the first      -->
<!-- access in the join sequence for the FROM clause.                                    -->
<!-- The SHARING attribute indicates that the access should be visible to other concurrent    -->
```

```
<!-- similar accesses that may therefore share bufferpool pages. The WRAPPING attribute       -->
<!-- indicates that the access should be allowed to perform wrapping, thereby allowing it to  -->
<!-- start in the middle for better sharing with other concurrent accesses. The THROTTLE      -->
<!-- attribute indicates that the access should be allowed to be throttled if this may        -->
<!-- benefit other concurrent accesses. The SHARESPEED attribute is used to indicate whether  -->
<!-- the access should be considered fast or slow for better grouping of concurrent accesses. -->
<!--***************************************************************************************** -->
<xs:complexType name="accessType" abstract="true">
   <xs:attribute name="TABLE" type="xs:string" use="optional"/>
   <xs:attribute name="TABID" type="xs:string" use="optional"/>
   <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
   <xs:attribute name="SHARING" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="WRAPPING" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="THROTTLE" type="optionType" use="optional" default="ENABLE"/>
   <xs:attribute name="SHARESPEED" type="shareSpeed" use="optional"/>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of an table scan access request method.                                    -->
<!--*****************************************************************************************-->
<xs:complexType name="tableScanType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
   </xs:complexType><!-- ***************************************************************************************-->
<!-- Definition of an index scan access request element. The index name is optional.         -->
<!--***************************************************************************** -->
<xs:complexType name="indexScanType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of a list prefetch access request element. The index name is optional.    -->
<!--***************************************************************************** -->
<xs:complexType name="listPrefetchType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of an extended access element which will be used by IXAND and ACCESS      -->
<!-- requests.                                                                            -->
<!-- A single index scan be specified via the INDEX attribute. Multiple indexes           -->
<!-- can be specified via INDEX elements. The index element specification supersedes the  -->
<!-- attribute specification. If a single index is specified, the optimizer will use the  -->
<!-- index as the first index of the index ANDing access method and will choose addi-     -->
<!-- tional indexes using cost. If multiple indexes are specified the optimizer will      -->
<!-- use exactly those indexes in the specified order. If no indexes are specified        -->
<!-- via either the INDEX attribute or INDEX elements, then the optimizer will choose     -->
<!-- all indexes based upon cost.                                                         -->
<!-- Extension for XML support:                                                           -->
<!-- TYPE: Optional attribute. The allowed value is XMLINDEX. When the type is not        -->
<!-- specified, the optimizer makes a cost based decision.                                -->
<!-- ALLINDEXES: Optional attribute. The allowed value is TRUE. The default               -->
<!-- value is FALSE.                                                                      -->
<!--***************************************************************************** -->
<xs:complexType name="extendedAccessType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:sequence minOccurs="0">
            <xs:element name="INDEX" type="indexType" minOccurs="2" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
         <xs:attribute name="TYPE" type="xs:string" use="optional" fixed="XMLINDEX"/>
         <xs:attribute name="ALLINDEXES" type="boolType" use="optional" fixed="TRUE"/>
      </xs:extension>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of an index ANDing access request element.                                -->
<!-- Extension for XML support:                                                           -->
<!-- All attributes and elements in extendedAccessType are included.                    -->
<!-- Note that ALLINDEXES is a valid option only if TYPE is XMLINDEX.                    -->
<!-- STARJOIN index ANDing: Specifying STARJOIN='TRUE' or one or more NLJOIN elements     -->
<!-- identifies the index ANDing request as a star join index ANDing request. When that   -->
<!-- is the case:                                                                         -->
<!--   TYPE cannot be XMLINDEX (and therefore ALLINDEXES cannot be specified).            -->
<!--   Neither the INDEX attribute nor INDEX elements can be specified.                   -->
<!--   The TABLE or TABID attribute identifies the fact table.                            -->
```

```
<!--   Zero or more semijoins can be specified using NLJOIN elements.                    -->
<!--   If no semijoins are specified, the optimizer will choose them.                     -->
<!--   If a single semijoin is specified, the optimizer will use it as the first semijoin -->
<!--   and will choose the rest itself.                                                   -->
<!--   If multiple semijoins are specified the optimizer will use exactly those semijoins -->
<!--   in the specified order.                                                            -->
<!--************************************************************************************ -->
<xs:complexType name="indexAndingType">
   <xs:complexContent>
      <xs:extension base="extendedAccessType">
         <xs:sequence minOccurs="0">
            <xs:element name="NLJOIN" type="nestedLoopJoinType" minOccurs="1" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="STARJOIN" type="boolType" use="optional"/>
      </xs:extension>
   </xs:complexContent>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of an INDEX element method. Index set is optional. If specified,          -->
<!-- at least 2 are required.                                                             -->
<!--************************************************************************************ -->
<xs:complexType name="indexType">
   <xs:attribute name="IXNAME" type="xs:string" use="optional"/>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of an XANDOR access request method.                                       -->
<!--*************************************************************************************-->
<xs:complexType name="XANDORType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Use for derived table access or other cases where the access method is not of        -->
<!-- consequence.                                                                         -->
<!-- Extension for XML support:                                                           -->
<!-- All attributes and elements in extendedAccessType are included.                      -->
<!-- Note that INDEX attribute/elements and ALLINDEXES are valid options only if TYPE     -->
<!-- is XMLINDEX.                                                                         -->
<!--*************************************************************************************-->
<xs:complexType name="anyAccessType">
   <xs:complexContent>
      <xs:extension base="extendedAccessType"/>
   </xs:complexContent>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of an index ORing access                                                  -->
<!-- Cannot specify more details (e.g indexes). Optimizer will choose the details based   -->
<!-- upon cost.                                                                           -->
<!--************************************************************************************ -->
<xs:complexType name="indexOringType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Effectively the super class from which join request elements inherit.                -->
<!-- This type currently defines join element inputs and also the FIRST attribute.        -->
<!-- A join request must have exactly two nested sub-elements. The sub-elements can be    -->
<!-- either an access request or another join request. The first sub-element represents   -->
<!-- outer table of the join operation while the second element represents the inner      -->
<!-- table. The FIRST attribute indicates that the join result should be the first join   -->
<!-- relative to other tables in the same FROM clause.                                    -->
<!--************************************************************************************ -->
<xs:complexType name="joinType" abstract="true">
   <xs:choice minOccurs="2" maxOccurs="2">
      <xs:group ref="accessRequest"/>
      <xs:group ref="joinRequest"/>
   </xs:choice>
   <xs:attribute name="FIRST" type="xs:string" use="optional" fixed="TRUE"/>
   </xs:complexType><!--************************************************************************************************** -->
<!-- Definition of nested loop join access request. Subclass of joinType.                 -->
<!-- Does not add any elements or attributes.                                             -->
<!--************************************************************************************ -->
<xs:complexType name="nestedLoopJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
   </xs:complexType><!--*************************************************************************************************-->
<!-- Definition of merge join access request. Subclass of joinType.                       -->
<!-- Does not add any elements or attributes.                                             -->
<!--************************************************************************************ -->
<xs:complexType name="mergeJoinType">
   <xs:complexContent>
```

```
      <xs:extension base="joinType"/>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of hash join access request. Subclass of joinType.                        -->
<!-- Does not add any elements or attributes.                                             -->
<!--************************************************************************************* -->
   <xs:complexType name="hashJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Any join is a subclass of binary join. Does not extend it in any way.                -->
<!-- Does not add any elements or attributes.                                             -->
<!--************************************************************************************* -->
<xs:complexType name="anyJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
   </xs:complexType><!--*****************************************************************************************-->
<!-- The MQTENFORCE element can be specified with one of two attributes:                  -->
<!--   NAME: Specify the MQT name directly as a value to this attribute.                  -->
<!--   TYPE: Specify the type of the MQTs that should be enforced with this attribute.    -->
<!--   Note that only the value of the first valid attribute found will be used. All      -->
<!--   subsequent attributes will be ignored.                                             -->
<!--*************************************************************************************-->
<xs:complexType name="mqtEnforcementType">
   <xs:attribute name="NAME" type="xs:string"/>
   <xs:attribute name="TYPE" type="mqtEnforcementTypeType"/>
   </xs:complexType><!--*****************************************************************************************************-->
<!-- Allowable values for the TYPE attribute of an MQTENFORCE element:                        -->
<!--   NORMAL: Enforce usage of all semantically matchable MQTs, except replicated MQTs.      -->
<!--   REPLICATED: Enforce usage of all semantically matchable replicated MQTs only.          -->
<!--   ALL: Enforce usage of all semantically matchable MQTs.                                 -->
<!--************************************************************************************** -->
<xs:simpleType name="mqtEnforcementTypeType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="NORMAL"/>
      <xs:enumeration value="REPLICATED"/>
      <xs:enumeration value="ALL"/>
   </xs:restriction>
</xs:simpleType>
<!--*****************************************************************************************-->
<!-- Allowable values for a boolean attribute.                                              -->
<!--************************************************************************************** -->
<xs:simpleType name="boolType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="TRUE"/>
      <xs:enumeration value="FALSE"/>
   </xs:restriction>
</xs:simpleType>
<!--*****************************************************************************************-->
<!-- Allowable values for an OPTION attribute.                                              -->
<!--************************************************************************************** -->
<xs:simpleType name="optionType">
   <xs:restriction base="xs:string">
      <xs:enumeration value="ENABLE"/>
      <xs:enumeration value="DISABLE"/>
   </xs:restriction>
</xs:simpleType>
<!--*****************************************************************************************-->
<!-- Allowable values for a SHARESPEED attribute.                                           -->
<!--************************************************************************************** -->
<xs:simpleType name="shareSpeed">
   <xs:restriction base="xs:string">
      <xs:enumeration value="FAST"/>
      <xs:enumeration value="SLOW"/>
   </xs:restriction>
</xs:simpleType>
<!--*****************************************************************************************-->
<!-- Definition of the qryopt type: the only values allowed are 0, 1, 2, 3, 5, 7 and 9       -->
<!--*****************************************************************************************-->
<xs:complexType name="qryoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
```

```
            <xs:enumeration value="5"/>
            <xs:enumeration value="7"/>
            <xs:enumeration value="9"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of the degree type: any number between 1 and 32767 or the strings ANY or -1  -->
<!--************************************************************************** -->
<xs:simpleType name="intStringType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="32767"></xs:maxInclusive>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ANY"/>
            <xs:enumeration value="-1"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>

<xs:complexType name="degreeType">
        <xs:attribute name="VALUE" type="intStringType"></xs:attribute>
   </xs:complexType><!--*****************************************************************************************-->
<!-- Definition of DPF XML movement types                                                  -->
<!--************************************************************************** -->
<xs:complexType name="dpfXMLMovementType">
    <xs:attribute name="VALUE" use="required">
     <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="REFERENCE"/>
        <xs:enumeration value="COMBINATION"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
    </xs:complexType>
</xs:schema>
```

OPTPROFILE 元素的 XML 模式:

OPTPROFILE 元素是优化概要文件的根。

此元素的定义如下:

XML 模式

```
   <xs:element name="OPTPROFILE">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="OPTGUIDELINES" type="globalOptimizationGuidelinesType"
              minOccurs="0"/>
            <xs:element name="STMTPROFILE" type="statementProfileType" minOccurs="0"
              maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="VERSION" use="optional">
            <xs:simpleType>
               <xs:restriction base="xs:NMTOKEN">
                  <xs:enumeration value="9.7.0.0"/>
               </xs:restriction>
            </xs:simpleType>
         </xs:attribute>
      </xs:complexType>
   </xs:element>
```

描述

可选的 OPTGUIDELINES 子元素用于定义优化概要文件的全局优化准则。每个 STMTPROFILE 子元素都定义一个语句概要文件。VERSION 属性用于标识当前优化概要文件模式，特定的优化概要文件将针对该模式进行创建和验证。

全局 OPTGUIDELINES 元素的 XML 模式:

OPTGUIDELINES 元素定义优化概要文件的全局优化准则。

此元素由复杂类型 globalOptimizationGuidelinesType 定义。

XML 模式

```
<xs:complexType name="globalOptimizationGuidelinesType">
   <xs:sequence>
      <xs:group ref="MQTOptimizationChoices"/>
      <xs:group ref="computationalPartitionGroupOptimizationChoices"/>
      <xs:group ref="generalRequest"/>
      <xs:group ref="mqtEnforcementRequest"/>
   </xs:sequence>
</xs:complexType>
```

描述

可以使用 MQTOptimizationChoices、computationalPartitionGroupChoices 或 generalRequest 组中的元素来定义全局优化准则。

- MQTOptimizationChoices 组元素可用于影响 MQT 替换。
- computationalPartitionGroupOptimizationChoices 组元素可用于影响计算分区组优化，后者涉及动态分布从远程数据源读取的数据。此元素仅适用于分区的联合数据库配置。
- generalRequest 组元素并不特定于优化过程的特定阶段，并且可用来更改优化器的搜索空间。可以全局指定这些元素，也可以在语句级别指定。
- MQT 实施请求指定可在语义上匹配的具体化查询表（MQT），应该在访问方案中强制使用这些表，而不考虑成本估算值。

MQT 优化选项:

MQTOptimizationChoices 组定义一组可用于影响具体化查询表（MQT）优化的元素。特别是，您可以使用这些元素来允许或禁止考虑 MQT 替换，也可以指定优化器要考虑的全部 MQT。

XML 模式

```
<xs:group name="MQTOptimizationChoices">
   <xs:choice>
      <xs:element name="MQTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
         </xs:complexType>
      </xs:element>
      <xs:element name="MQT" minOccurs="0" maxOccurs="unbounded">
         <xs:complexType>
            <xs:attribute name="NAME" type="xs:string" use="required"/>
         </xs:complexType>
      </xs:element>
   </xs:choice>
</xs:group>
```

描述

MQTOPT 元素用于启用或禁用考虑 MQT 优化。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。

MQT 元素的 NAME 属性标识优化器要考虑的 MQT。在 NAME 属性中用于构造对 MQT 的引用的规则与构造对显示表名的引用的规则相同。如果指定了一个或多个 MQT 元素，那么优化器只考虑那些 MQT。仍然会根据成本决定是否使用一个或多个指定的 MQT 来执行 MQT 替换。

示例

以下示例说明如何禁用 MQT 优化。

```
<OPTGUIDELINES>
   <MQTOPT OPTION='DISABLE'/>
</OPTGUIDELINES>
```

以下示例说明如何将 MQT 优化限制到 Tpcd.PARTSMQT 表和 COLLEGE.STUDENTS 表。

```
<OPTGUIDELINES>
   <MQT NAME='Tpcd.PARTSMQT'/>
   <MQT NAME='COLLEGE.STUDENTS'/>
<OPTGUIDELINES>
```

计算分区组优化选项:

computationalPartitionGroupOptimizationChoices 组定义一组可用于影响计算分区组优化的元素。特别是，您可以使用这些元素来启用或禁用计算分区组优化，也可以指定用于计算分区组优化的分区组。

XML 模式

```
<xs:group name="computationalPartitionGroupOptimizationChoices">
   <xs:choice>
      <xs:element name="PARTOPT" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="OPTION" type="optionType" use="optional"/>
         </xs:complexType>
      </xs:element>
      <xs:element name="PART" minOccurs="0" maxOccurs="1">
         <xs:complexType>
            <xs:attribute name="NAME" type="xs:string" use="required"/>
         </xs:complexType>
      </xs:element>
   </xs:choice>
</xs:group>
```

描述

PARTOPT 元素用于启用或禁用考虑计算分区组优化。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。

可以使用 PART 元素来指定用于计算分区组优化的分区组。NAME 属性必须标识现有的分区组。将根据成本决定是否使用指定的分区组执行动态重新分布。

示例

以下示例说明如何禁用计算分区组优化。

```
<OPTGUIDELINES>
   <PARTOPT OPTION='DISABLE'/>
</OPTGUIDELINES>
```

以下示例说明如何指定将 WORKPART 分区组用于计算分区组优化。

```
<OPTGUIDELINES>
   <MQT NAME='Tpcd.PARTSMQT'/>
   <PART NAME='WORKPART'/>
<OPTGUIDELINES>
```

作为全局请求的一般优化准则:

generalRequest 组定义不特定于优化过程的特定阶段的准则，并可用于更改优化器的搜索空间。

可以在全局级别和语句级别指定一般优化准则。对于全局优化准则和语句级别优化准则来说，它们的一般优化准则元素的描述和语法都相同。有关更多信息，请参阅“一般优化准则的 XML 模式”。

STMTPROFILE 元素的 XML 模式:

STMTPROFILE 元素用于在优化概要文件中定义语句概要文件。

此元素由复杂类型 statementProfileType 定义。

XML 模式

```
<xs:complexType name="statementProfileType">
   <xs:sequence>
      <xs:element name="STMTKEY" type="statementKeyType"/>
      <xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
   </xs:sequence>
   <xs:attribute name="ID" type="xs:string" use="optional"/>
</xs:complexType>
```

描述

语句概要文件指定特定语句的优化准则，它包含下列部分。

- 语句关键字

  优化概要文件可以对应用程序中的多个语句生效。优化器使用语句关键字使每个语句概要文件自动与应用程序中相应的语句匹配。这使您能够为语句提供优化准则，而不必编辑应用程序。语句关键字包含应用程序中编写的语句文本以及明确标识正确的语句所需的其他信息。STMTKEY 子元素表示语句关键字。

- 语句级优化准则

  语句概要文件的此部分指定作用于语句关键字所标识的语句的优化准则。有关信息，请参阅“语句级 OPTGUIDELINES 元素的 XML 模式”。

- 语句概要文件名

  出现在诊断输出中的用户指定名称，用于标识特定的语句概要文件。

STMTKEY 元素的 XML 模式:

STMTKEY 元素使优化器能够将语句概要文件与应用程序中的相应语句匹配。

此元素由复杂类型 statementKeyType 定义。

XML 模式

```
<xs:complexType name="statementKeyType" mixed="true">
   <xs:attribute name="SCHEMA" type="xs:string" use="optional"/>
   <xs:attribute name="FUNCPATH" type="xs:string" use="optional"/>
</xs:complexType>
</xs:schema>
```

描述

可选的 SCHEMA 属性可用于指定语句关键字的缺省模式部分。

可选的 FUNCPATH 属性可用于指定语句关键字的函数路径部分。各个路径必须由逗号分隔，指定的函数路径必须与编译关键字中指定的函数路径完全匹配。

示例

以下示例提供了一个语句关键字定义，此定义使特定语句与缺省模式“COLLEGE”和函数路径“SYSIBM,SYSFUN,SYSPROC,DAVE”相关联。

```
<STMTKEY SCHEMA='COLLEGE' FUNCPATH='SYSIBM,SYSFUN,SYSPROC,DAVE'>
   <![CDATA[select * from orders" where foo(orderkey) > 20]]>
</STMTKEY>
```

由于语句文本包含特殊的 XML 字符“>”，因此 CDATA 标记（以 <![CDATA[开头并以]]> 结尾）是必需的。

语句关键字和编译关键字匹配:

语句关键字用于标识语句级优化准则所应用于的应用程序语句。

编译 SQL 语句时，有多种因素会影响编译器在语义方面解释语句的方式。SQL 语句以及 SQL 编译器参数设置共同构成编译关键字。语句关键字的每一部分都对应于编译关键字的某个部分。

语句关键字由下列部分组成:

- 语句文本，这是应用程序中编写的语句的文本
- 缺省模式，这是用作未限定表名的隐式限定符的模式名；此部分是可选的，但如果语句包含未限定的表名，那么应该提供此部分
- 函数路径，这是解析未限定的函数和数据类型引用时使用的函数路径；此部分是可选的，但如果语句包含未限定的用户定义的函数或用户定义的类型，那么应该提供此部分

当数据服务器编译 SQL 语句并查找活动的优化概要文件时，它将尝试使优化概要文件中的每个语句关键字与当前编译关键字匹配。如果语句关键字的每个指定部分都与编译关键字的相应部分匹配，那么认为语句关键字和编译关键字匹配。如果未指定语句关键字的某个部分，那么缺省情况下认为省略的部分匹配。语句关键字的每个未指定部分都被视为通配符，它与任何编译关键字的相应部分匹配。

数据服务器找到与当前编译关键字匹配的语句关键字后，它就停止搜索。如果有多个语句概要文件的语句关键字与当前编译关键字匹配，那么将只使用第一个语句概要文件（基于文档顺序）。

语句级 OPTGUIDELINES 元素的 XML 模式:

语句概要文件的 OPTGUIDELINES 元素定义作用于相关联语句关键字所标识的语句的优化准则。此元素由类型 optGuidelinesType 定义。

XML 模式

```
<xs:element name="OPTGUIDELINES" type="optGuidelinesType"/>
<xs:complexType name="optGuidelinesType">
   <xs:sequence>
      <xs:group ref="general request" minOccurs="0" maxOccurs="1"/>
      <xs:choice maxOccurs="unbounded">
         <xs:group ref="rewriteRequest"/>
         <xs:group ref="accessRequest"/>
         <xs:group ref="joinRequest"/>
         <xs:group ref="mqtEnforcementRequest"/>
      </xs:choice>
  </xs:sequence>
</xs:complexType>
```

描述

optGuidelinesType 组定义 OPTGUIDELINES 元素的有效子元素集。DB2 优化器将每个子元素都解释为优化准则。可以将子元素分类为一般请求元素、重写请求元素、访问请求元素或连接请求元素。

- *一般请求元素*用于指定一般优化准则，这些准则可用于更改优化器的搜索空间。
- *重写请求元素*用于指定查询重写优化准则，这些准则可用于影响确定优化语句时应用的查询变换。
- *访问请求元素*和*连接请求元素*是方案优化准则，这些准则可用于影响已优化语句的执行方案中使用的访问方法、连接方法和连接顺序。
- *MQT 实施请求元素*指定在语义上匹配的具体化查询表（MQT），应该在访问方案中强制使用这些表，而不考虑成本估算值。

注：语句概要文件中指定的优化准则优先于优化概要文件的全局节中指定的优化准则。

一般优化准则的 XML 模式:

generalRequest 组定义不特定于优化过程的特定阶段的准则，并可用于更改优化器的搜索空间。

```
<!--****************************************************************************** --> \
<!-- Choices of general request elements.                                         --> \
<!-- REOPT can be used to override the setting of the REOPT bind option.          --> \
<!-- DPFXMLMOVEMENT can be used to affect the optimizer's plan when moving XML documents  --> \
<!-- between database partitions. The allowed value can be REFERENCE or COMBINATION. The  --> \
<!-- default value is NONE.                                                       --> \
<!--****************************************************************************** --> \
<xs:group name="generalRequest">
   <xs:sequence>
      <xs:element name="REOPT" type="reoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DEGREE" type="degreeType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="QRYOPT" type="qryoptType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="RTS" type="rtsType" minOccurs="0" maxOccurs="1"/>
      <xs:element name="DPFXMLMOVEMENT" type="dpfXMLMovementType" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:group>
```

注：可以在全局级别和语句级别指定一般优化准则。对于全局优化准则和语句级别优化准则来说，它们的一般优化准则元素的描述和语法都相同。

描述

一般请求元素可用于定义一般优化准则，后者影响优化搜索空间，从而可以影响重写优化准则以及基于成本的优化准则的适用性。

DEGREE 请求:

您可以使用 DEGREE 一般请求元素来覆盖 DEGREE 绑定选项设置、**dft_degree** 数据库配置参数值或先前 SET CURRENT DEGREE 语句的结果。

仅当已对实例进行分区内并行性配置时，才会考虑 DEGREE 一般请求元素；否则，将返回警告。此元素由复杂类型 degreeType 定义。

XML 模式

```
<xs:simpleType name="intStringType">
   <xs:union>
      <xs:simpleType>
         <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"></xs:minInclusive>
            <xs:maxInclusive value="32767"></xs:maxInclusive>
         </xs:restriction>
      </xs:simpleType>
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="ANY"/>
            <xs:enumeration value="-1"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:union>
</xs:simpleType>

<xs:complexType name="degreeType">
   <xs:attribute name="VALUE"
      type="intStringType"></xs:attribute>
</xs:complexType>
```

描述

DEGREE 一般请求元素包含必需的 VALUE 属性，此属性指定 DEGREE 选项的设置。此属性可以具有 1 到 32767 之间的整数值或者字符串值 -1 或 ANY。值 -1（或 ANY）指定并行度由数据服务器确定。值 1 指定查询不应使用分区内并行性。

DPFXMLMOVEMENT 请求:

在分区数据库环境中，可以使用 DPFXMLMOVEMENT 一般请求元素来覆盖优化器在选择方案时所作的决策，即，是在数据库分区之间移动类型为 XML 的列还是仅移动对该列的引用。此元素由复杂类型 dpfXMLMovementType 定义。

```
<xs:complexType name="dpfXMLMovementType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string"
            <xs:enumeration value="REFERENCE"/>
            <xs:enumeration value="COMBINATION"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
```

描述

在分区数据库环境中执行语句时，有时必须在数据库分区之间移动数据。对于 XML 列，优化器可以选择移动那些列中包含的实际文档，也可以只移动对原始数据库分区中源文档的引用。

DPFXMLMOVEMENT 一般请求元素具有必需的 VALUE 属性，此属性的可能值为 REFERENCE 或 COMBINATION。如果需要将包含 XML 列的行从一个数据库分区移至另一个数据库分区:

- REFERENCE 指定通过表队列（TQ）运算符来移动对 XML 文档的引用。文档本身仍在源数据库分区中。
- COMBINATION 指定移动某些 XML 文档，对于其余 XML 文档，只通过 TQ 运算符来移动对它们的引用。

移动文档还是仅移动对那些文档的引用这一决策取决于运行查询时的条件。如果未指定 DPFXMLMOVEMENT 一般请求元素，那么优化器将进行基于成本的决策，以便最大程度地提高性能。

QRYOPT 请求:

您可以使用 QRYOPT 一般请求元素来覆盖 QUERYOPT 绑定选项设置、**dft_queryopt** 数据库配置参数值或先前 SET CURRENT QUERY OPTIMIZATION 语句的结果。此元素由复杂类型 qryoptType 定义。

XML 模式

```
<xs:complexType name="qryoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
         <xs:restriction base="xs:string">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="2"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="5"/>
            <xs:enumeration value="7"/>
            <xs:enumeration value="9"/>
         </xs:restriction>
      </xs:simpleType>
   </xs:attribute>
</xs:complexType>
```

描述

QRYOPT 一般请求元素包含必需的 VALUE 属性，此属性指定 QUERYOPT 选项的设置。此属性可以具有下列任何一个值: 0、1、2、3、5、7 或 9。有关这些值的含义的详细信息，请参阅“优化类”。

REOPT 请求:

可以使用 REOPT 一般请求元素来覆盖 REOPT 绑定选项的设置，后者影响包含参数标记或主变量的语句的优化。此元素由复杂类型 reoptType 定义。

XML 模式

```
<xs:complexType name="reoptType">
   <xs:attribute name="VALUE" use="required">
      <xs:simpleType>
```

```
            <xs:restriction base="xs:string">
               <xs:enumeration value="ONCE"/>
               <xs:enumeration value="ALWAYS"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:attribute>
   </xs:complexType>
```

描述

REOPT 一般请求元素包含必需的 VALUE 属性，此属性指定 REOPT 选项的设置。此属性可以具有值 ONCE 或 ALWAYS。ONCE 指定应该为第一组主变量或参数标记值优化语句。ALWAYS 指定应该为每一组主变量或参数标记值优化语句。

RTS 请求:

RTS 一般请求元素可用于启用或禁用实时统计信息收集功能。它还可用于限制实时统计信息收集功能所耗费的时间量。

对于某些查询或工作负载而言，最好对实时统计信息收集功能进行限制，以避免语句编译期间产生额外的开销。RTS 一般请求元素由复杂类型 rtsType 定义。

```
<!--*****************************************************************************************--> \
<!-- RTS general request element to enable, disable or provide a time budget for             --> \
<!-- real-time statistics collection.                                                        --> \
<!-- OPTION attribute allows enabling or disabling real-time statistics.                     --> \
<!-- TIME attribute provides a time budget in milliseconds for real-time statistics collection.--> \
<!--***************************************************************************************** --> \
<xs:complexType name="rtsType">
   <xs:attribute name="OPTION" type="optionType" use="optional"
   default="ENABLE"/>
   <xs:attribute name="TIME" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
```

描述

RTS 一般请求元素具有两个可选属性。

- OPTION 属性用于启用或禁用实时统计信息收集功能。此属性的值可以是 ENABLE（缺省值）或 DISABLE。
- TIME 属性指定语句编译期间收集每个语句的实时统计信息时可以耗用的最大时间量（以毫秒计）。

如果对 OPTION 属性指定了 ENABLE，那么必须通过相应的配置参数来启用自动收集统计信息功能和实时统计信息收集功能。否则，将不会应用该优化准则，并且将返回 SQL0437W（原因码为 13）。

查询重写优化准则的 XML 模式:

rewriteRequest 组定义将影响优化过程的查询重写阶段的准则。

XML 模式

```
   <xs:group name="rewriteRequest">
      <xs:sequence>
         <xs:element name="INLIST2JOIN" type="inListToJoinType" minOccurs="0"/>
         <xs:element name="SUBQ2JOIN"   type="subqueryToJoinType" minOccurs="0"/>
         <xs:element name="NOTEX2AJ"    type="notExistsToAntiJoinType" minOccurs="0"/>
         <xs:element name="NOTIN2AJ"    type="notInToAntiJoinType" minOccurs="0"/>
      </xs:sequence>
   </xs:group>
```

描述

如果使用 INLIST2JOIN 元素来同时指定语句级和谓词级优化准则，那么谓词级准则将覆盖语句级准则。

“IN-LIST 到连接”查询重写请求:

您可以使用 INLIST2JOIN 查询重写请求元素来启用或禁用“IN-LIST 谓词到连接”重写变换。可以将其指定为语句级优化准则，也可以将其指定为谓词级优化准则。在后一种情况下，对每个查询只能启用一个准则。INLIST2JOIN 请求元素由复杂类型 inListToJoinType 定义。

XML 模式

```
<xs:complexType name="inListToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional"
   default="ENABLE"/>
   <xs:attribute name="TABLE"  type="xs:string"  use="optional"/>
   <xs:attribute name="COLUMN" type="xs:string"  use="optional"/>
</xs:complexType>
```

描述

INLIST2JOIN 查询重写请求元素有三个可选属性，但是没有子元素。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。TABLE 和 COLUMN 属性用于指定 IN-LIST 谓词。如果未指定这些属性，或者对这些属性指定空字符串（『』）值，那么该准则将作为语句级准则处理。如果指定这两个属性的其中一个或全部，那么该准则将作为谓词级准则处理。如果未指定 TABLE 属性，或者对此属性指定空字符串值但指定了 COLUMN 属性，那么此优化准则将被忽略，并且将返回 SQL0437W（原因码为 13）。

“NOT-EXISTS 到反连接”查询重写请求:

您可以使用 NOTEX2AJ 查询重写请求元素来启用或禁用“NOT-EXISTS 谓词到反连接”重写变换。只能将其指定为语句级优化准则。NOTEX2AJ 请求元素由复杂类型 notExistsToAntiJoinType 定义。

XML 模式

```
<xs:complexType name="notExistsToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional"
   default="ENABLE"/>
</xs:complexType>
```

描述

NOTEX2AJ 查询重写请求元素有一个可选属性，但是没有子元素。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。

“NOT-IN 到反连接”查询重写请求:

您可以使用 NOTIN2AJ 查询重写请求元素来启用或禁用“NOT-IN 谓词到反连接”重写变换。只能将其指定为语句级优化准则。NOTIN2AJ 请求元素由复杂类型 notInToAntiJoinType 定义。

XML 模式

```
<xs:complexType name="notInToAntiJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional"
   default="ENABLE"/>
</xs:complexType>
```

描述

NOTIN2AJ 查询重写请求元素有一个可选属性，但是没有子元素。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。

“子查询到连接”查询重写请求:

您可以使用 SUBQ2JOIN 查询重写请求元素来启用或禁用“子查询到连接”重写变换。只能将其指定为语句级优化准则。SUBQ2JOIN 请求元素由复杂类型 subqueryToJoinType 定义。

XML 模式

```
<xs:complexType name="subqueryToJoinType">
   <xs:attribute name="OPTION" type="optionType" use="optional"
   default="ENABLE"/>
</xs:complexType>
```

描述

SUBQ2JOIN 查询重写请求元素有一个可选属性，但是没有子元素。OPTION 属性的值可以是 ENABLE（缺省值）或 DISABLE。

方案优化准则的 XML 模式:

方案优化准则可以由访问请求或连接请求组成。

- *访问请求*指定表引用的访问方法。
- *连接请求*指定用于执行连接操作的方法和顺序。连接请求由其他访问请求或连接请求组成。

大多数可用的访问请求与优化器的数据访问方法（例如表扫描、索引扫描和列表预取）相对应。大多数可用的连接请求与优化器的连接方法（例如嵌套循环连接、散列连接和合并连接）相对应。每个访问请求或连接请求元素都可用于影响方案优化。

访问请求:

accessRequest 组定义有效访问请求元素的集合。访问请求指定表引用的访问方法。

XML 模式

```
<xs:group name="accessRequest">
   <xs:choice>
      <xs:element name="TBSCAN" type="tableScanType"/>
      <xs:element name="IXSCAN" type="indexScanType"/>
      <xs:element name="LPREFETCH" type="listPrefetchType"/>
      <xs:element name="IXAND" type="indexAndingType"/>
      <xs:element name="IXOR" type="indexOringType"/>
      <xs:element name="XISCAN" type="indexScanType"/>
      <xs:element name="XANDOR" type="XANDORType"/>
      <xs:element name="ACCESS" type="anyAccessType"/>
   </xs:choice>
</xs:group>
```

描述

- TBSCAN、IXSCAN、LPREFETCH、IXAND、IXOR、XISCAN 和 XANDOR

  这些元素对应于 DB2 数据访问方法，并且只能应用于语句中引用的本地表。它们不能引用昵称（远程表）或派生的表（子选择的结果）。

- ACCESS

  当连接顺序（而不是访问方法）是主要的关注事项时，可以使用此元素，这将使优化器选择访问方法。当目标表引用是派生的表时，必须使用 ACCESS 元素。对于 XML 查询而言，此元素还可以在属性 TYPE 设置为 XMLINDEX 的情况下使用，以便指定让优化器选择 XML 索引访问方案。

访问类型:

TBSCAN、IXSCAN、LPREFETCH、IXAND、IXOR、XISCAN、XANDOR 和 ACCESS 元素的公共方面由抽象类型 accessType 定义。

XML 模式

```
   <xs:complexType name="accessType" abstract="true">
      <xs:attribute name="TABLE" type="xs:string" use="optional"/>
      <xs:attribute name="TABID" type="xs:string" use="optional"/>
      <xs:attribute name="FIRST" type="xs:string" use="optional"
      fixed="TRUE"/>
      <xs:attribute name="SHARING" type="optionType" use="optional"
      default="ENABLE"/>
<xs:attribute name="WRAPPING" type="optionType" use="optional"
      default="ENABLE"/>
<xs:attribute name="THROTTLE" type="optionType" use="optional"/>
      <xs:attribute name="SHARESPEED" type="shareSpeed" use="optional"/>
   </xs:complexType>

   <xs:complexType name="extendedAccessType">
      <xs:complexContent>
         <xs:extension base="accessType">
            <xs:sequence minOccurs="0">
               <xs:element name="INDEX" type="indexType" minOccurs="2"
                 maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="INDEX" type="xs:string" use="optional"/>
            <xs:attribute name="TYPE" type="xs:string" use="optional"
            fixed="XMLINDEX"/>
<xs:attribute name="ALLINDEXES" type="boolType" use="optional"
fixed="TRUE"/>
</xs:extension>
      </xs:complexContent>
   </xs:complexType>
```

描述

所有访问请求元素都扩展复杂类型 accessType。每个这种元素都必须使用 TABLE 或 TABID 属性来指定目标表引用。有关如何根据访问请求元素来构造正确的表引用的信息，请参阅“在优化准则中构造表引用”。

访问请求元素还可以指定可选的 FIRST 属性。如果指定了 FIRST 属性，那么此属性的值必须为 TRUE。对访问请求元素添加 FIRST 属性表明，执行方案应该包含指定的表作为相应 FROM 子句的连接顺序中的第一个表。在每个 FROM 子句中，只能有一个访问或连接请求指定 FIRST 属性。如果同一个 FROM 子句的多个访问或连接请求目标表指定了 FIRST 属性，那么将忽略除第一个请求以外的所有请求并返回警告（SQL0437W，原因码为 13）。

新的优化器准则使您能够影响编译器的扫描共享决策。在编译器将允许共享扫描、回绕扫描或调速的情况下，指定适当的准则将阻止共享扫描、回绕扫描或调速。共享扫

描可以被其他参与扫描共享的扫描所见，那些扫描可以根据该信息进行某些决策。回绕扫描可以从表中的任意位置开始，以便利用已包含在缓冲池中的页。调速的扫描是为了提高整体共享级别已延迟的扫描。

对于 SHARING、WRAPPING 和 THROTTLE 属性而言，有效的 optionType 值是 DISABLE 和 ENABLE（缺省值）。如果编译器选择禁用 SHARING 和 WRAPPING，那么不能将其启用。在那些情况下，使用 ENABLE 将不起作用。可以启用或禁用 THROTTLE。有效的 SHARESPEED 值（用于编译器估算的扫描速度）是 FAST 和 SLOW。缺省情况是，允许编译器根据估算结果来确定值。

TYPE 属性所支持的值只有 XMLINDEX，此值向优化器表明，必须使用其中一种 XML 索引访问方法（例如 IXAND、IXOR、XANDOR 或 XISCAN）来访问表。如果未指定此属性，那么优化器在为指定的表选择访问方案时，将进行基于成本的决策。

可以使用可选的 INDEX 属性来指定索引名。

可以使用可选的 INDEX 元素将索引的两个或更多个名称指定为索引元素。如果同时指定 INDEX 属性和 INDEX 元素，那么 INDEX 属性将被忽略。

仅当 TYPE 属性的值为 XMLINDEX 时，才能指定可选的 ALLINDEXES 属性（此属性所支持的值只有 TRUE）。如果指定了 ALLINDEXES 属性，那么优化器必须使用所有适用的关系索引以及基于 XML 数据的索引来访问指定的表，而不考虑成本。

任何访问请求:

可以使用 ACCESS 访问请求元素来指定，优化器将根据成本来选择用于访问表的适当方法；在引用派生的表时，必须使用此元素。派生的表是另一个子选择的结果。此访问请求元素由复杂类型 anyAccessType 定义。

XML 模式

```
<xs:complexType name="anyAccessType">
   <xs:complexContent>
      <xs:extension base="extendedAccessType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 anyAccessType 是抽象类型 extendedAccessType 的简单扩展。未添加新元素或属性。

TYPE 属性所支持的值只有 XMLINDEX，它向优化器表明，必须使用其中一种 XML 索引访问方法（例如 IXAND、IXOR、XANDOR 或 XISCAN）来访问表。如果未指定此属性，那么优化器在为指定的表选择访问方案时，将进行基于成本的决策。

仅当 TYPE 属性的值为 XMLINDEX 时，才可以使用可选的 INDEX 属性来指定索引名。如果指定了此属性，那么优化器可能会选择下列其中一个方案:

- 使用基于 XML 数据的所指定索引的 XISCAN 方案
- XANDOR 方案，以使基于 XML 数据的所指定索引成为 XANDOR 下的其中一个索引；优化器将在 XANDOR 方案中使用所有基于 XML 数据的适用索引
- IXAND 方案，以使指定的索引成为 IXAND 的前导索引；优化器将以基于成本的方式对 IXAND 方案添加更多索引。

- 基于成本的 IXOR 方案

仅当 TYPE 属性的值为 XMLINDEX 时，才可以使用可选的 INDEX 元素来指定两个或更多个索引名作为索引元素。如果指定了此元素，那么优化器可能会选择下列其中一个方案:

- XANDOR 方案，以使基于 XML 数据的所指定索引出现在 XANDOR 下；优化器将在 XANDOR 方案中使用所有基于 XML 数据的适用索引
- IXAND 方案，以使指定的索引按所指定顺序成为 IXAND 的索引
- 基于成本的 IXOR 方案

如果同时指定 INDEX 属性和 INDEX 元素，那么 INDEX 属性将被忽略。

仅当 TYPE 属性的值为 XMLINDEX 时，才能指定可选的 ALLINDEXES 属性（此属性所支持的值只有 TRUE）。如果指定了此属性，那么优化器必须使用所有适用的关系索引以及基于 XML 数据的索引来访问指定的表，而不考虑成本。优化器将选择下列其中一个方案:

- XANDOR 方案，所有基于 XML 数据的适用索引都将在 XANDOR 运算符下出现
- IXAND 方案，所有适用的关系索引以及基于 XML 数据的索引都将在 IXAND 运算符下出现
- IXOR 方案
- XISCAN 方案（仅当对表定义了单一索引并且该索引的类型为 XML 时）

示例

以下准则是任何访问请求的一个示例:

```
<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1'/>
    <IXSCAN TABLE='PS1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

以下示例说明 ACCESS 准则，此准则指定应该对 SECURITY 表进行某些 XML 索引访问。优化器可能会选择任何 XML 索引方案，例如 XISCAN、IXAND、XANDOR 或 IXOR 方案。

```
SELECT * FROM security
  WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
    StockInformation[Industry= "OfficeSupplies"]')

<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX'/>
</OPTGUIDELINES>
```

以下示例说明 ACCESS 准则，此准则指定应该对 SECURITY 表进行所有可能的索引访问。优化器负责选择访问方法。假定两个 XML 索引（SEC_INDUSTRY 和 SEC_SYMBOL）与两个 XML 谓词匹配。优化器将以基于成本的方式选择 XANDOR 或 IXAND 访问方法。

```
SELECT * FROM security
  WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
      StockInformation[Industry= "Software"]') AND
    XMLEXISTS('$SDOC/Security/Symbol[.="IBM"]')
```

```
<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX' ALLINDEXES='TRUE'/>
</OPTGUIDELINES>
```

以下示例说明 ACCESS 准则，此准则指定至少应该使用 SEC_INDUSTRY XML 索引来访问 SECURITY 表。优化器将以基于成本的方式选择下列其中一个访问方案:

- 使用 XML 索引 SEC_INDUSTRY 的 XISCAN 方案
- 将 SEC_INDUSTRY 索引作为 IXAND 的第一个分支的 IXAND 方案。优化器可以通过进行基于成本的分析在 IXAND 方案中随意使用更多的关系索引或 XML 索引。例如，如果存在基于 TRANS_DATE 列的关系索引，那么该索引可能会作为 IXAND 的另一个分支出现（如果优化器认为这样有好处的话）。
- 使用 SEC_INDUSTRY 索引和其他适用的 XML 索引的 XANDOR 方案。

```
SELECT * FROM security
  WHERE trans_date = CURRENT DATE AND
    XMLEXISTS('$SDOC/Security/SecurityInformation/
      StockInformation[Industry= "Software"]') AND
    XMLEXISTS('$SDOC/Security/Symbol[.="IBM"]')

<OPTGUIDELINES>
  <ACCESS TABLE='SECURITY' TYPE='XMLINDEX' INDEX='SEC_INDUSTRY'/>
</OPTGUIDELINES>
```

索引 AND 运算访问请求:

可以使用 IXAND 访问请求元素来指定，优化器将使用索引 AND 运算数据访问方法来访问本地表。此元素由复杂类型 indexAndingType 定义。

XML 模式

```
<xs:complexType name="indexAndingType">
   <xs:complexContent>
      <xs:extension base="extendedAccessType">
         <xs:sequence minOccurs="0">
            <xs:element name="NLJOIN" type="nestedLoopJoinType" minOccurs="1"
              maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="STARJOIN" type="boolType" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 indexAndingType 是 extendedAccessType 的扩展。当未指定 STARJOIN 属性和 NLJOIN 元素时，indexAndingType 将成为 extendedAccessType 的简单扩展。extendedAccessType 类型通过添加可选的 INDEX 属性、可选的 INDEX 子元素、可选的 TYPE 属性以及可选的 ALLINDEXES 属性来扩展抽象类型 accessType。INDEX 属性可用于指定索引 AND 运算中使用的第一个索引。如果使用了 INDEX 属性，那么优化器将以基于成本的方式来选择其他索引和访问顺序。INDEX 子元素可用于指定确切的一组索引和访问顺序。INDEX 子元素的显示顺序指示执行各个索引扫描时应采用的顺序。指定 INDEX 子元素将取代指定 INDEX 属性。

- 如果未指定索引，那么优化器将以基于成本的方式选择索引和访问顺序。
- 如果通过使用属性或子元素指定了索引，那么这些索引必须是对 TABLE 或 TABID 属性所标识的表定义的索引。
- 如果未对该表定义任何索引，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

TYPE 属性所支持的值只有 XMLINDEX，它向优化器表明，必须使用一个或多个基于 XML 数据的索引来访问表。

仅当 TYPE 属性的值为 XMLINDEX 时，才可以使用可选的 INDEX 属性来指定 XML 索引名。可以在可选的 INDEX 属性中指定关系索引，而与如何指定 TYPE 属性无关。优化器会将指定的索引用作 IXAND 方案的前导索引。优化器将以基于成本的方式对 IXAND 方案添加更多索引。

仅当 TYPE 属性的值为 XMLINDEX 时，才可以使用可选的 INDEX 元素来指定两个或更多个基于 XML 数据的索引的名称作为索引元素。可以在可选的 INDEX 属性中指定关系索引，而与如何指定 TYPE 元素无关。优化器将按指定的顺序将指定的索引用作 IXAND 方案的索引。

即使未指定 TYPE 属性，INDEX 属性和 INDEX 元素对于关系索引而言也仍然有效。

如果同时指定 INDEX 属性和 INDEX 元素，那么 INDEX 属性将被忽略。

仅当 TYPE 属性的值为 XMLINDEX 时，才能指定可选的 ALLINDEXES 属性（此属性所支持的值只有 TRUE）。如果指定了此属性，那么优化器必须在 IXAND 方案中使用所有适用的关系索引以及基于 XML 数据的索引来访问指定的表，而不考虑成本。

如果指定了 TYPE 属性，但未指定 INDEX 属性、INDEX 元素和 ALLINDEXES 属性，那么优化器将选择至少包含一个基于 XML 数据的索引的 IXAND 方案。此方案中的其他索引可以是关系索引或基于 XML 数据的索引。索引的顺序和选项由优化器以基于成本的方式确定。

在索引 AND 运算访问请求中，块索引必须出现在记录索引前面。如果未符合此要求，那么将返回 SQL0437W（原因码为 13）。索引 AND 运算访问方法要求，对于每个索引，至少能够对一个谓词建立索引。如果由于必需的谓词不存在而导致索引 AND 运算不符合条件，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。如果该索引 AND 运算访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

可以使用访问请求元素 IXAND 来请求星型连接索引 AND 运算方案。IXAND 元素的可选 STARJOIN 属性指定将 IXAND 用于星型连接索引 AND 运算方案。NLJOIN 可以是 IXAND 的子元素，并且必须是正确构造的星型连接半连接。`STARJOIN="FALSE"` 指定对于常规基本访问索引 AND 运算方案的请求。`STARJOIN="TRUE"` 指定对于星型连接索引 AND 运算方案的请求。缺省值由上下文确定：如果 IXAND 有一个或多个半连接子元素，那么缺省值为 TRUE；否则，缺省值为 FALSE。如果指定了 `STARJOIN="TRUE"`：

- 不能指定 INDEX、TYPE 和 ALLINDEXES 属性
- 不能指定 INDEX 元素

如果指定了 NLJOIN 元素：

- 不能指定 INDEX、TYPE 和 ALLINDEXES 属性
- 不能指定 INDEX 元素
- STARJOIN 属性唯一支持的值为 TRUE

以下示例说明索引 AND 运算访问请求：

SQL 语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
  where p_partkey = ps.ps_partkey and
    s.s_suppkey = ps.ps_suppkey and
    p_size = 39 and
    p_type = 'BRASS' and
    s.s_nation in ('MOROCCO', 'SPAIN') and
    ps.ps_supplycost = (select min(ps1.ps_supplycost)
                          from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                          where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                            s1.s_suppkey = ps1.ps_suppkey and
                            s1.s_nation = s.s_nation)
  order by s.s_name
  optimize for 1 row
```

优化准则:

```
<OPTGUIDELINES>
  <IXAND TABLE='"Tpcd".PARTS' FIRST='TRUE'>
    <INDEX IXNAME='ISIZE'/>
    <INDEX IXNAME='ITYPE'/>
  </IXAND>
</OPTGUIDELINES>
```

索引 AND 运算请求指定，将使用索引 AND 运算数据访问方法来满足主要子选择中的 PARTS 表。第一个索引扫描将使用 ISIZE 索引，第二个索引扫描将使用 ITYPE 索引。索引由 INDEX 元素的 IXNAME 属性指定。FIRST 属性设置指定，PARTS 表将是同一个 FROM 子句包含 SUPPLIERS、PARTSUPP 和派生表的连接序列中的第一个表。

以下示例说明了星型连接索引 AND 运算准则，它指定了第一个半连接，但是允许优化器选择其余半连接。它还允许优化器选择所指定半连接中的外部表以及内部表的索引的特定访问方法。

```
<IXAND TABLE="F">
  <NLJOIN>
    <ACCESS TABLE="D1"/>
    <IXSCAN TABLE="F"/>
  </NLJOIN>
</IXAND>
```

以下准则将指定所有半连接（其中包括详细信息），让优化器没有用于 IXAND 和低于 IXAND 的方案的选项。

```
<IXAND TABLE="F" STARJOIN="TRUE">
  <NLJOIN>
    <TBSCAN TABLE="D1"/>
    <IXSCAN TABLE="F" INDEX="FX1"/>
  </NLJOIN>
  <NLJOIN>
    <TBSCAN TABLE="D4"/>
    <IXSCAN TABLE="F" INDEX="FX4"/>
  </NLJOIN>
  <NLJOIN>
    <TBSCAN TABLE="D3"/>
    <IXSCAN TABLE="F" INDEX="FX3"/>
  </NLJOIN>
</IXAND>
```

索引 OR 运算访问请求:

可以使用 IXOR 访问请求元素来指定，优化器将使用索引 OR 运算数据访问方法来访问本地表。此元素由复杂类型 indexOringType 定义。

XML 模式

```
<xs:complexType name="indexOringType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 indexOringType 是抽象类型 accessType 的简单扩展。未添加新元素或属性。如果该索引 OR 运算访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。优化器将以基于成本的方式选择索引 OR 运算中使用的谓词和索引。索引 OR 运算访问方法要求至少能够对一个 IN 谓词建立索引，或者能够对带有项的谓词建立索引并通过逻辑 OR 运算对其进行连接。如果由于必需的谓词或索引不存在而导致索引 OR 运算不符合条件，那么将忽略该请求并返回 SQL0437W（原因码为 13）。

以下示例说明索引 OR 运算访问请求:

SQL 语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
  from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
  where p_partkey = ps.ps_partkey and
    s.s_suppkey = ps.ps_suppkey and
    p_size = 39 and
    p_type = 'BRASS' and
    s.s_nation in ('MOROCCO', 'SPAIN') and
    ps.ps_supplycost = (select min(ps1.ps_supplycost)
                          from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                          where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                            s1.s_suppkey = ps1.ps_suppkey and
                            s1.s_nation = s.s_nation)
  order by s.s_name
  optimize for 1 row
```

优化准则:

```
<OPTGUIDELINES>
  <IXOR TABLE='S'/>
</OPTGUIDELINES>
```

此索引 OR 运算访问请求指定，将使用索引 OR 运算数据访问方法来访问主要子选择中引用的 SUPPLIERS 表。优化器将以基于成本的方式为索引 OR 运算选择适当的谓词和索引。

索引扫描访问请求:

可以使用 IXSCAN 访问请求元素来指定，优化器将使用索引扫描方法来访问本地表。此元素由复杂类型 indexScanType 定义。

XML 模式

```
<xs:complexType name="indexScanType">
   <xs:complexContent>
      <xs:extension base="accessType">
```

```
        <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 indexScanType 通过添加可选的 INDEX 属性来扩展抽象类型 accessType。INDEX 属性指定要用于访问表的索引的名称。

- 如果该索引扫描访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果指定了 INDEX 属性，那么它必须标识对 TABLE 或 TABID 属性所指定的表定义的索引。如果该索引不存在，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果未指定 INDEX 属性，那么优化器将以基于成本的方式选择索引。如果未对目标表定义任何索引，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

以下准则是索引扫描访问请求的一个示例:

```
<OPTGUIDELINES>
  <IXSCAN TABLE='S' INDEX='I_SUPPKEY'/>
</OPTGUIDELINES>
```

列表预取访问请求:

可以使用 LPREFETCH 访问请求元素来指定，优化器将使用列表预取索引扫描方法来访问本地表。此元素由复杂类型 listPrefetchType 定义。

XML 模式

```
<xs:complexType name="listPrefetchType">
   <xs:complexContent>
      <xs:extension base="accessType">
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 listPrefetchType 通过添加可选的 INDEX 属性来扩展抽象类型 accessType。INDEX 属性指定要用于访问表的索引的名称。

- 如果该列表预取访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 列表预取访问方法要求至少能够对一个谓词建立索引。如果由于必需的谓词不存在而导致列表预取访问方法不符合条件，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果指定了 INDEX 属性，那么它必须标识对 TABLE 或 TABID 属性所指定的表定义的索引。如果该索引不存在，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果未指定 INDEX 属性，那么优化器将以基于成本的方式选择索引。如果未对目标表定义任何索引，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

以下准则是列表预取访问请求的一个示例:

```
<OPTGUIDELINES>
  <LPREFETCH TABLE='S1' INDEX='I_SNATION'/>
</OPTGUIDELINES>
```

表扫描访问请求:

可以使用 TBSCAN 访问请求元素来指定，优化器将使用顺序表扫描方法来访问本地表。此元素由复杂类型 tableScanType 定义。

XML 模式

```
<xs:complexType name="tableScanType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 tableScanType 是抽象类型 accessType 的简单扩展。未添加新元素或属性。如果该表扫描访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

以下准则是表扫描访问请求的一个示例:

```
<OPTGUIDELINES>
  <TBSCAN TABLE='S1'/>
</OPTGUIDELINES>
```

XML 索引 AND 运算和索引 OR 运算访问请求:

可以使用 XANDOR 访问请求元素来指定，优化器将使用多个基于 XML 数据的 XANDOR 索引扫描方法来访问本地表。此元素由复杂类型 XANDORType 定义。

XML 模式

```
<xs:complexType name="XANDORType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 XANDORType 是抽象类型 accessType 的简单扩展。未添加新元素或属性。

示例

给定以下查询:

```
SELECT * FROM security
  WHERE trans_date = CURRENT DATE AND
    XMLEXISTS('$SDOC/Security/SecurityInformation/
      StockInformation[Industry = "Software"]') AND
    XMLEXISTS('$SDOC/Security/Symbol[.="IBM"]')
```

以下 XANDOR 准则指定，应该通过对所有适用的 XML 索引执行 XANDOR 运算来访问 SECURITY 表。由于关系索引无法与 XANDOR 运算符配合使用，因此不会考虑使用 SECURITY 表的任何关系索引。

```
<OPTGUIDELINES>
  <XANDOR TABLE='SECURITY'/>
</OPTGUIDELINES>
```

XML 索引扫描访问请求:

可以使用 XISCAN 访问请求元素来指定，优化器将使用基于 XML 数据的索引扫描方法来访问本地表。此元素由复杂类型 indexScanType 定义。

XML 模式

```
<xs:complexType name="indexScanType">
   <xs:complexContent>
      <xs:extension base="accessType"/>
         <xs:attribute name="INDEX" type="xs:string" use="optional"/>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 indexScanType 通过添加可选的 INDEX 属性来扩展抽象类型 accessType。INDEX 属性指定要用于访问表的基于 XML 数据的索引的名称。

- 如果该基于 XML 数据的索引扫描访问方法不在作用于该语句的搜索空间中，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果指定了 INDEX 属性，那么它必须标识对 TABLE 或 TABID 属性所指定的表定义的基于 XML 数据的索引。如果该索引不存在，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。
- 如果未指定 INDEX 属性，那么优化器将以基于成本的方式选择基于 XML 数据的索引。如果未对目标表定义任何基于 XML 数据的索引，那么将忽略该访问请求并返回 SQL0437W（原因码为 13）。

示例

给定以下查询:

```
SELECT * FROM security
  WHERE XMLEXISTS('$SDOC/Security/SecurityInformation/
    StockInformation[Industry = "OfficeSupplies"]')
```

以下 XISCAN 准则指定，应该使用名为 SEC_INDUSTRY 的 XML 索引来访问 SECURITY 表。

```
<OPTGUIDELINES>
  <XISCAN TABLE='SECURITY' INDEX='SEC_INDUSTRY'/>
</OPTGUIDELINES>
```

连接请求:

joinRequest 组定义有效连接请求元素的集合。连接请求指定用于连接两个表的方法。

XML 模式

```
<xs:group name="joinRequest">
   <xs:choice>
      <xs:element name="NLJOIN" type="nestedLoopJoinType"/>
      <xs:element name="HSJOIN" type="hashJoinType"/>
```

```
      <xs:element name="MSJOIN" type="mergeJoinType"/>
      <xs:element name="JOIN" type="anyJoinType"/>
    </xs:choice>
  </xs:group>
```

描述

- NLJOIN、MSJOIN 和 HSJOIN

  这些元素分别对应于嵌套循环连接方法、合并连接方法和散列连接方法。

- JOIN

  当连接顺序不是主要的关注事项时，可以使用此元素，这将使优化器选择连接方法。

所有连接请求元素都包含两个子元素，它们表示连接操作的输入表。连接请求还可以指定可选的 FIRST 属性。

以下准则是连接请求的示例:

```
<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1'/>
    <IXSCAN TABLE='PS1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

嵌套顺序最终确定连接顺序。以下示例说明如何根据较小的连接请求来构造较大的连接请求:

```
<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='"Tpcd".Parts'/>
      <IXSCAN TABLE="PS"/>
    </NLJOIN>
    <IXSCAN TABLE='S'/>
  </MSJOIN>
</OPTGUIDELINES>
```

连接类型:

所有连接请求元素的公共方面由抽象类型 joinType 定义。

XML 模式

```
<xs:complexType name="joinType" abstract="true">
   <xs:choice minOccurs="2" maxOccurs="2">
      <xs:group ref="accessRequest"/>
      <xs:group ref="joinRequest"/>
   </xs:choice>
   <xs:attribute name="FIRST" type="xs:string" use="optional"
   fixed="TRUE"/>
</xs:complexType>
```

描述

扩展复杂类型 joinType 的连接请求元素必须正好包含两个子元素。其中的任何一个子元素都可以是从 accessRequest 组中选择的访问请求元素，另一个子元素可以是从 joinRequest 组中选择的连接请求元素。连接请求中出现的第一个子元素指定连接操作的外表，第二个元素指定内表。

如果指定了 FIRST 属性，那么此属性的值必须为 TRUE。对连接请求元素添加 FIRST 属性表示，您希望有这样一个执行方案，其中连接请求的目标表是相应 FROM 子句的连接顺序中最外部的表。在每个 FROM 子句中，只能有一个访问或连接请求指定 FIRST 属性。如果以同一个 FROM 子句中的表为目标的多个访问或连接请求指定了 FIRST 属性，那么将忽略除初始请求以外的所有请求，并且将返回 SQL0437W（原因码为 13）。

任何连接请求:

可以使用 JOIN 连接请求元素来指定优化器要选择适当的方法按特定顺序连接两个表。

这两个表可以是本地表或派生的表（由访问请求子元素指定），也可以是连接操作的结果（由连接请求子元素指定）。派生的表是另一个子选择的结果。此连接请求元素由复杂类型 anyJoinType 定义。

XML 模式

```
<xs:complexType name="anyJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 anyJoinType 是抽象类型 joinType 的简单扩展。未添加新元素或属性。

以下示例说明如何使用 JOIN 连接请求元素对一组表强制使用特定连接顺序:

SQL 语句:

```
select s.s_name, s.s_address, s.s_phone, s.s_comment
   from "Tpcd".parts, "Tpcd".suppliers s, "Tpcd".partsupp ps
   where p_partkey = ps.ps_partkey and
     s.s_suppkey = ps.ps_suppkey and
     p_size = 39 and
     p_type = 'BRASS' and
     s.s_nation in ('MOROCCO', 'SPAIN') and
     ps.ps_supplycost = (select min(ps1.ps_supplycost)
                           from "Tpcd".partsupp ps1, "Tpcd".suppliers s1
                           where "Tpcd".parts.p_partkey = ps1.ps_partkey and
                             s1.s_suppkey = ps1.ps_suppkey and
                             s1.s_nation = s.s_nation)
   order by s.s_name
```

优化准则:

```
<OPTGUIDELINES>
  <JOIN>
    <JOIN>
      <ACCESS TABLE='Tpcd".PARTS'/>
      <ACCESS TABLE='S'/>
    </JOIN>
    <ACCESS TABLE='PS'>
  </JOIN>
</OPTGUIDELINES>
```

JOIN 连接请求元素指定，主要子选择中的 PARTS 表将与 SUPPLIERS 表进行连接，并指定此结果将与 PARTSUPP 表进行连接。优化器将以基于成本的方式为这个特定的连接序列选择连接方法。

散列连接请求:

可以使用 HSJOIN 连接请求元素来指定，优化器采用散列连接方法来连接两个表。

这两个表可以是本地表或派生的表（由访问请求子元素指定），也可以是连接操作的结果（由连接请求子元素指定）。派生的表是另一个子选择的结果。此连接请求元素由复杂类型 hashJoinType 定义。

XML 模式

```
<xs:complexType name="hashJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 hashJoinType 是抽象类型 joinType 的简单扩展。未添加新元素或属性。如果该散列连接方法不在作用于该语句的搜索空间中，那么将忽略连接请求并返回 SQL0437W（原因码为 13）。

以下准则是散列连接请求的示例:

```
<OPTGUIDELINES>
  <HSJOIN>
    <ACCESS TABLE='S1'/>
    <IXSCAN TABLE='PS1'/>
  </HSJOIN>
</OPTGUIDELINES>
```

合并连接请求:

可以使用 MSJOIN 连接请求元素来指定，优化器采用合并连接方法来连接两个表。

这两个表可以是本地表或派生的表（由访问请求子元素指定），也可以是连接操作的结果（由连接请求子元素指定）。派生的表是另一个子选择的结果。此连接请求元素由复杂类型 mergeJoinType 定义。

XML 模式

```
<xs:complexType name="mergeJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 mergeJoinType 是抽象类型 joinType 的简单扩展。未添加新元素或属性。如果该合并连接方法不在作用于该语句的搜索空间中，那么将忽略连接请求并返回 SQL0437W（原因码为 13）。

以下准则是合并连接请求的示例:

```
<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='"Tpcd".Parts'/>
      <IXSCAN TABLE="PS"/>
    </NLJOIN>
    <IXSCAN TABLE='S'/>
  </MSJOIN>
</OPTGUIDELINES>
```

嵌套循环连接请求:

可以使用 NLJOIN 连接请求元素来指定，优化器采用嵌套循环连接方法来连接两个表。

这两个表可以是本地表或派生的表（由访问请求子元素指定），也可以是连接操作的结果（由连接请求子元素指定）。派生的表是另一个子选择的结果。此连接请求元素由复杂类型 nestedLoopJoinType 定义。

XML 模式

```
<xs:complexType name="nestedLoopJoinType">
   <xs:complexContent>
      <xs:extension base="joinType"/>
   </xs:complexContent>
</xs:complexType>
```

描述

复杂类型 nestedLoopJoinType 是抽象类型 joinType 的简单扩展。未添加新元素或属性。如果该嵌套循环连接方法不在作用于该语句的搜索空间中，那么将忽略连接请求并返回 SQL0437W（原因码为 13）。

以下准则是嵌套循环连接请求的示例:

```
<OPTGUIDELINES>
  <NLJOIN>
    <IXSCAN TABLE='"Tpcd".Parts'/>
    <IXSCAN TABLE="PS"/>
  </NLJOIN>
</OPTGUIDELINES>
```

SYSTOOLS.OPT_PROFILE 表:

SYSTOOLS.OPT_PROFILE 表包含所有优化概要文件。

可以采用两种方法来创建此表:

- 调用 SYSINSTALLOBJECTS 过程:

  ```
  db2 "call sysinstallobjects('opt_profiles', 'c', '', '')"
  ```

- 发出 CREATE TABLE 语句:

  ```
  create table systools.opt_profile (
    schema  varchar(128) not null,
    name    varchar(128) not null,
    profile blob (2m)    not null,
    primary key (schema, name)
  )
  ```

按如下所示定义 SYSTOOLS.OPT_PROFILE 表中的列:

SCHEMA
指定优化概要文件的模式名。此名称最多可以包含 30 个字母数字或下划线字符，但它定义为如上所示的 VARCHAR(128)。

NAME 指定优化概要文件的基本名称。此名称最多可以包含 128 个字母数字或下划线字符。

PROFILE
指定用于定义优化概要文件的 XML 文档。

用于清空优化概要文件高速缓存的触发器:

每当 SYSTOOLS.OPT_PROFILE 表中的条目被更新或删除时，都将自动清空优化概要文件高速缓存。

必须创建以下 SQL 过程和触发器，这样才能自动清空概要文件高速缓存。

```
CREATE PROCEDURE SYSTOOLS.OPT_FLUSH_CACHE( IN SCHEMA VARCHAR(128),
                                           IN NAME VARCHAR(128) )
LANGUAGE SQL
MODIFIES SQL DATA
BEGIN ATOMIC
 -- FLUSH stmt (33) + quoted schema (130) + dot (1) + quoted name (130) = 294
 DECLARE FSTMT VARCHAR(294) DEFAULT 'FLUSH OPTIMIZATION PROFILE CACHE '; --

 IF NAME IS NOT NULL THEN
   IF SCHEMA IS NOT NULL THEN
     SET FSTMT = FSTMT || '"' || SCHEMA || '".'; --
   END IF; --

   SET FSTMT = FSTMT || '"' || NAME || '"'; --

   EXECUTE IMMEDIATE FSTMT; --
 END IF; --
END;

CREATE TRIGGER SYSTOOLS.OPT_PROFILE_UTRIG AFTER UPDATE ON SYSTOOLS.OPT_PROFILE
REFERENCING OLD AS O
FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );

CREATE TRIGGER SYSTOOLS.OPT_PROFILE_DTRIG AFTER DELETE ON SYSTOOLS.OPT_PROFILE
REFERENCING OLD AS O
FOR EACH ROW
  CALL SYSTOOLS.OPT_FLUSH_CACHE( O.SCHEMA, O.NAME );
```

管理 SYSTOOLS.OPT_PROFILE 表:

优化概要文件必须与唯一的模式限定名关联，并且必须存储在 SYSTOOLS.OPT_PROFILE 表中。您可以使用 LOAD、IMPORT 和 EXPORT 命令来管理该表中的文件。

例如，可以从任何 DB2 客户机中使用 IMPORT 命令在 SYSTOOLS.OPT_PROFILE 表中插入或更新数据。可以使用 EXPORT 命令将 SYSTOOLS.OPT_PROFILE 表中的概要文件复制到文件。

以下示例说明如何将三个新的概要文件插入到 SYSTOOLS.OPT_PROFILE 表。假定文件在当前目录中。

1. 创建一个输入文件（例如 profiledata），并在不同的行中输入每个概要文件的模式、名称和文件名:

   ```
   "ROBERT","PROF1","ROBERT.PROF1.xml"
   "ROBERT","PROF2","ROBERT.PROF2.xml"
   "DAVID", "PROF1","DAVID.PROF1.xml"
   ```

2. 执行 IMPORT 命令:

   ```
   import from profiledata of del
     modified by lobsinfile
     insert into systools.opt_profile
   ```

要更新现有的行，请使用 IMPORT 命令的 INSERT_UPDATE 选项:

```
import from profiledata of del
  modified by lobsinfile
  insert_update into systools.opt_profile
```

要将 ROBERT.PROF1 概要文件复制到 ROBERT.PROF1.xml（假定概要文件长度小于 32700 字节），请使用 EXPORT 命令:

```
export to robert.prof1.xml of del
  select profile from systools.opt_profile
    where schema='ROBERT' and name='PROF1'
```

有关更多信息，包括如何导出长度超过 32700 字节的数据，请参阅“EXPORT 命令”。

数据库分区组对查询优化的影响

在分区数据库环境中，优化器在确定查询的最佳访问方案时，它能够识别并使用表的并置。

如果在连接查询中频繁涉及某些表，那么应该将那些表划分到各个数据库分区，以使每个所连接的表中的行位于同一个数据库分区中。在执行连接操作期间，所连接的两个表中数据的并置能够避免将数据从一个数据库分区移至另一个数据库分区。请将这两个表置于同一个数据库分区组中，以确保对这两个表中的数据进行并置。

根据表的大小，将数据分布到多个数据库分区有助于缩短执行查询所需的估计时间。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。

收集准确的目录统计信息（包括高级统计信息功能部件）

准确的数据库统计信息对于查询优化而言十分关键。您应该定期地对任何对于查询性能而言十分关键的表执行 RUNSTATS 操作。

如果应用程序直接查询系统目录表，并且存在大量目录更新活动（例如由于执行数据定义语言（DDL）语句而产生的目录更新活动），那么您还可能想收集这些表的统计信息。您可以启动自动收集统计信息功能，以允许 DB2 数据服务器自动执行 RUNSTATS 操作。可以启用实时收集统计信息功能，以便允许 DB2 数据服务器在优化查询前一刻收集统计信息，从而提供时效性更强的统计信息。

如果您正在使用 RUNSTATS 命令以手动方式收集统计信息，那么至少应该使用下列选项:

```
RUNSTATS ON TABLE DB2USER.DAILY_SALES
  WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
```

分布统计信息使优化器能够了解数据分布不均匀情况。使用特定的索引来访问表时，详细的索引统计信息能够提供更多有关访存数据页所需的 I/O 的详细信息。对于大型表而言，收集详细的索引统计信息将耗用相当多的处理时间和内存。SAMPLED 选项能够提供几乎同样准确的详细索引统计信息，但所需的 CPU 和内存只是几分之一。没有为表提供统计概要文件时，自动收集统计信息功能也将使用这些缺省值。

为了提高查询性能，请考虑收集更高级的统计信息，例如列组统计信息或 LIKE 统计信息，或者创建统计视图。

列组统计信息

如果查询包含多个用于连接两个表的连接谓词，那么 DB2 优化器在选择用于执行该查询的方案之前，将计算每个谓词的选择性。

例如，假定一个制造商利用各种颜色、弹性和品质的原料来生产产品。制成品与所用的原料具有相同的颜色和弹性。此制造商发出以下查询:

```
SELECT PRODUCT.NAME, RAWMATERIAL.QUALITY
  FROM PRODUCT, RAWMATERIAL
  WHERE
    PRODUCT.COLOR = RAWMATERIAL.COLOR AND
    PRODUCT.ELASTICITY = RAWMATERIAL.ELASTICITY
```

此查询将返回所有产品的名称和原料品质。共有两个连接谓词:

```
PRODUCT.COLOR = RAWMATERIAL.COLOR
PRODUCT.ELASTICITY = RAWMATERIAL.ELASTICITY
```

优化器假定这两个谓词相互独立，即，对于每种颜色，存在各种弹性。于是，优化器将使用每个表的目录统计信息并根据弹性级别数和不同颜色数来估算每一对谓词的总体选择性。例如，根据此估算，它可能选择嵌套循环连接而不是合并连接，反之亦然。

但是，这两个谓词可能并非相互独立。例如，可能只有几种颜色的高弹性材料，并且只有另外几种颜色的低弹性材料。在这种情况下，谓词的组合选择性将消除较少的行，因此查询将返回较多的行。在没有此信息的情况下，优化器可能不再选择最佳方案。

要收集 PRODUCT.COLOR 和 PRODUCT.ELASTICITY 的列组统计信息，请发出以下 RUNSTATS 命令:

```
RUNSTATS ON TABLE PRODUCT ON COLUMNS ((COLOR, ELASTICITY))
```

优化器将使用这些统计信息来检测关联情况并动态地调整相关谓词的组合选择性，从而更正确地估算连接大小和成本。

当查询使用 GROUP BY 或 DISTINCT 之类的关键字对数据进行分组时，列组统计信息还使优化器能够计算相异分组的数目。

请考虑以下查询:

```
SELECT DEPTNO, YEARS, AVG(SALARY)
       FROM EMPLOYEE   GROUP BY DEPTNO, MGR, YEAR_HIRED
```

如果没有任何索引或列组统计信息，那么优化器估算的分组数（以及本例中返回的行数）将是 DEPTNO、MGR 和 YEAR_HIRED 中相异值数的乘积。这个估算假定各个分组键列相互独立。但是，如果每位经理都正好管理一个部门，那么这个假定可能不正确。并且，不可能每个部门每年都聘请职员。因此，估算的 DEPTNO、MGR 和 YEAR_HIRED 相异值乘积可能超出实际的相异组数。

对 DEPTNO、MGR 和 YEAR_HIRED 收集的列组统计信息为优化器提供了上一个查询的相异分组的确切数目:

```
RUNSTATS ON TABLE EMPLOYEE ON COLUMNS ((DEPTNO, MGR, YEAR_HIRED))
```

优化器除管理 JOIN 谓词关联以外，还管理与简单等式谓词的关联，例如:

```
DEPTNO = 'Sales' AND  MGR = 'John'
```

在此示例中，应用于 EMPLOYEE 表中 DEPTNO 列的谓词可能与应用于 YEAR 列的谓词无关。但是，应用于 DEPTNO 和 MGR 的谓词肯定不会相互独立，这是因为，每个部门在某个时刻通常都由一位经理管理。优化器使用关于列的统计信息来确定组合的相异值数，然后调整所估算的基数以便考虑各列之间的关联。

简单等式谓词的关联

除了使用连接谓词关联以外，优化器还使用类型为 COL = 的简单等式谓词来管理关联。

例如，假定一个表包含各种类型的轿车，每一种轿车都具有 MAKE（即制造商）、MODEL、YEAR、COLOR 和 STYLE（例如私家车、旅行车或赛车）。因为几乎每家制造商每年生产的各种型号和款式的轿车都具有相同的标准颜色，所以 COLOR 的谓词很可能独立于 MAKE、MODEL、STYLE 或 YEAR 的谓词。但是，因为只有一家轿车制造商才会生产具有特定名称的型号，所以基于 MAKE 和 MODEL 的谓词并非相互独立。同一个型号名几乎不可能由两家或两家以上的轿车制造商使用。

如果存在基于 MAKE 和 MODEL 这两列的索引，或者已收集列组统计信息，那么优化器将使用关于该索引或这两列的统计信息来确定组合的相异值数，并调整这两列之间的关联的选择性或基数估算。如果这些谓词是局部的等式谓词，那么优化器不需要唯一索引即可进行调整。

统计视图

基于成本的 DB2 优化器使用访问方案运算符所处理的估算行数（即“基数”）来准确地估算该运算符的成本。此基数估算是优化器的成本模型的唯一最重要输入，其准确性在很大程度上取决于 RUNSTATS 实用程序从数据库收集的统计信息。

需要更复杂的统计信息来表示更复杂的关系，例如涉及表达式的比较（例如 price > MSRP + Dealer_markup）、跨多个表的关系（例如 product.name = 'Alloy wheels' and product.key = sales.product_key）或者除涉及独立属性和简单比较运算的谓词以外的任何其他内容。统计视图能够表示这些类型的复杂关系，这是因为，统计信息是根据视图返回的结果集收集的，而不是根据视图所引用的基本表收集的。

编译查询时，优化器将使该查询与可用的统计视图匹配。当优化器估算中间结果集的基数时，它将使用来自视图的统计信息来更好地进行估算。

查询不必直接引用统计视图即可让优化器使用统计视图。优化器将使用用于具体化查询表（MQT）的匹配机制使查询与统计视图匹配。在这方面，统计视图除了不会被永久存储、不耗用磁盘空间以及不必进行维护以外，与 MQT 非常相似。

要创建统计视图，请先创建一个视图，然后使用 ALTER VIEW 语句对该视图启用优化功能。然后，对该统计视图运行 RUNSTATS 命令，从而在系统目录表中填充该视图的统计信息。例如，要创建一个统计视图以表示星型模式中 TIME 维表与事实表之间的连接，请执行以下操作:

```
CREATE VIEW SV_TIME_FACT AS (
  SELECT T.* FROM TIME T, SALES S
    WHERE T.TIME_KEY = S.TIME_KEY)

ALTER VIEW SV_TIME_FACT ENABLE QUERY OPTIMIZATION

RUNSTATS ON TABLE DB2DBA.SV_TIME_FACT WITH DISTRIBUTION
```

这个统计视图可用来改进基数估算，从而改进类似于以下的查询的访问方案并提高查询性能:

```
SELECT SUM(S.PRICE)
  FROM SALES S, TIME T, PRODUCT P
  WHERE
```

```
T.TIME_KEY = S.TIME_KEY AND
T.YEAR_MON = 200712 AND
P.PROD_KEY = S.PROD_KEY AND
P.PROD_DESC = 'Power drill'
```

如果没有统计视图，那么优化器将假定所有与特定 TIME 维 YEAR_MON 值对应的事实表 TIME_KEY 值都均匀地出现在事实表中。但是，十二月份的销售情况极佳，使得销售交易数显著多于其他月份。

目前，自动收集统计信息功能不适用于统计视图。因此，每次对统计视图所引用的基本表进行大量更新之后，请对这些视图收集统计信息。

使用统计视图

在将视图的统计信息用于优化查询之前，必须允许对该视图进行优化。支持进行优化的视图被称为*统计视图*。

非统计视图不能进行优化，并且称为*常规视图*。视图在最初创建时，不允许进行优化。请使用 ALTER VIEW 语句来允许对视图进行优化。有关执行此任务所需的特权和权限，请参阅 ALTER VIEW 语句的描述。有关对视图使用 RUNSTATS 实用程序所需的特权和权限，请参阅 RUNSTATS 命令的描述。

如果下列任何一个条件成立，那么无法对视图启用优化:

- 该视图直接或间接地引用具体化查询表（MQT）。（MQT 或统计视图可以引用统计视图。）
- 该视图不再有效。
- 该视图是带类型视图。
- 在同一 ALTER VIEW 语句中存在另一个视图更改请求。

如果正在进行更改以启用优化功能的视图的定义包含下列任何一项，那么将返回警告，并且优化器将不会利用该视图的统计信息:

- 聚集操作或相异值操作
- 并集、差集或交集操作
- OLAP 规范

1. 对该视图启用优化功能。

   可以使用 ALTER VIEW 语句的 ENABLE OPTIMIZATION 子句对视图启用优化功能。以后，可以使用 DISABLE OPTIMIZATION 子句对已启用优化功能的视图禁用优化功能。例如，要对 MYVIEW 启用优化功能，请输入以下命令:

   ```
   alter view myview enable query optimization
   ```

2. 调用 RUNSTATS 命令。 例如，要对 MYVIEW 收集统计信息，请输入以下命令:

   ```
   runstats on table db2dba.myview
   ```

   收集视图统计信息（其中包括分布统计信息）时，要以 10% 为标准进行行级采样，请输入以下命令:

   ```
   runstats on table db2dba.myview with distribution tablesample bernoulli (10)
   ```

   收集视图统计信息（其中包括分布统计信息）时，要以 10% 为标准进行页级别采样，请输入以下命令:

   ```
   runstats on table db2dba.myview with distribution tablesample system (10)
   ```

3. 可选: 如果受视图定义影响的查询是静态 SQL 程序包的组成部分，那么请重新绑定那些程序包，以便利用由于新统计信息而对访问方案进行的更改。

与优化相关的视图统计信息

在查询优化期间，将仅考虑那些能够体现定义了统计视图（例如 CARD 和 COLCARD）的查询的数据分布特征的统计信息。

可以收集下列与视图记录相关联的统计信息供优化器使用。

- 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES）
  - CARD - 视图结果中的行数
- 列统计信息（SYSCAT.COLUMNS 和 SYSSTAT.COLUMNS）
  - COLCARD - 视图结果的某一列中相异值的数目
  - AVGCOLLEN - 视图结果的某一列的平均长度
  - HIGH2KEY - 视图结果的某一列中的次大值
  - LOW2KEY - 视图结果的某一列中的次小值
  - NUMNULLS - 视图结果的某一列中空值的数目
  - SUB_COUNT - 视图结果的某一列中子元素的平均数目
  - SUB_DELIM_LENGTH - 每个用于分隔子元素的定界符的平均长度
- 列分布统计信息（SYSCAT.COLDIST 和 SYSSTAT.COLDIST）
  - DISTCOUNT - 小于或等于 COLVALUE 统计信息的相异分位数值的数目
  - SEQNO - 按频率排列的序号，用于帮助唯一地标识表中的行
  - COLVALUE - 要收集其频率或分位数统计信息的数据值
  - VALCOUNT - 数据值在视图列中的出现频率；对于分位数，这是小于或等于数据值（COLVALUE）的值的数目

可以收集并非描述数据分布情况的统计信息（例如 NPAGES 和 FPAGES），但优化器将忽略这些信息。

方案: 使用统计视图来提高基数估算准确性

在数据仓库中，事实表中的信息经常更改，而维表数据则是静态的。这表示维属性数据可能与事实表属性数据一致，也可能不一致。

当前可用于优化器的传统基本表统计信息不允许优化器辨别各个表之间的关系。可以使用统计视图（和 MQT）的列和表分布统计信息为优化器提供更正这些类型的基数估算错误所需的信息。

请考虑以下查询，它计算每年七月份高尔夫球棍的销售收入:

```
select  sum(f.sales_price), d2.year
  from  product d1, period d2, daily_sales f
  where  d1.prodkey = f.prodkey
    and  d2.perkey = f.perkey
    and  d1.item_desc = 'golf club'
    and  d2.month = 'JUL'
  group by  d2.year
```

如果优化器可以确定是涉及 PRODUCT 和 DAILY_SALES 的半连接还是涉及 PERIOD 和 DAILY_SALES 的半连接最具选择性，那么对于此查询而言，星型连接查询执行方案是一个不错的选择。为了生成有效的星型连接方案，优化器必须能够选择最具选择性的半连接作为索引 AND 运算的外支路。

数据仓库通常包含货架上已没有的产品的记录。这将导致连接后 PRODUCT 列的分布与连接前的分布极不相同。由于缺少更好的信息，优化器将只能根据基本表统计信息来确定本地谓词的选择性，因此优化器可能对谓词 `item_desc = 'golf club'` 的选择性过于乐观。

例如，如果高尔夫球棍在历史上占所生产产品量的 1%，但现在占销售量的 20%，那么优化器很可能高估 `item_desc = 'golf club'` 的选择性，这是因为没有描述进行连接后 *item_desc* 的分布情况的统计信息。如果十二个月的销售量都大致相同，那么谓词 `month = 'JUL'` 的选择性大约为 8%，因此错误估计谓词 `item_desc = 'golf club'` 的选择性将导致优化器将看似更具选择性的 PRODUCT 与 DAILY_SALES 之间的半连接作为星型连接方案的索引 AND 运算的外支路。

以下示例提供了有关如何设置统计视图来解决此类问题的逐步说明。

以下是典型数据仓库中的一个数据库，其中 STORE、CUSTOMER、PRODUCT、PROMOTION 和 PERIOD 是维表，而 DAILY_SALES 是事实表。下列各表提供了这些表的定义。

表 55. STORE（63 行）

列	storekey	store_number	city	state	district	...
属性	integer 非空主键	char(2)	char(20)	char(5)	char(14)	...

表 56. CUSTOMER（1000000 行）

列	custkey	name	address	age	gender	...
属性	integer 非空主键	char(30)	char(40)	smallint	char(1)	...

表 57. PRODUCT（19450 行）

列	prodkey	category	item_desc	price	cost	...
属性	integer 非空主键	integer	char(30)	decimal(11)	decimal(11)	...

表 58. PROMOTION（35 行）

列	promokey	promotype	promodesc	promovalue	...
属性	integer 非空主键	integer	char(30)	decimal(5)	...

表 59. PERIOD（2922 行）

列	perkey	calendar_date	month	period	year	...
属性	integer 非空主键	date	char(3)	smallint	smallint	...

表 60. DAILY_SALES（754069426 行）

列	storekey	custkey	prodkey	promokey	perkey	sales_price	...
属性	integer	integer	integer	integer	integer	decimal(11)	...

假定公司的经理希望确定如果他们为回头客打折，那么消费者是否会再次购买该产品。此外，假定仅对在全国有 18 家分店的商店“01”进行调查。表 61 显示了关于不同促销类别的信息。

表 61. PROMOTION（35 行）

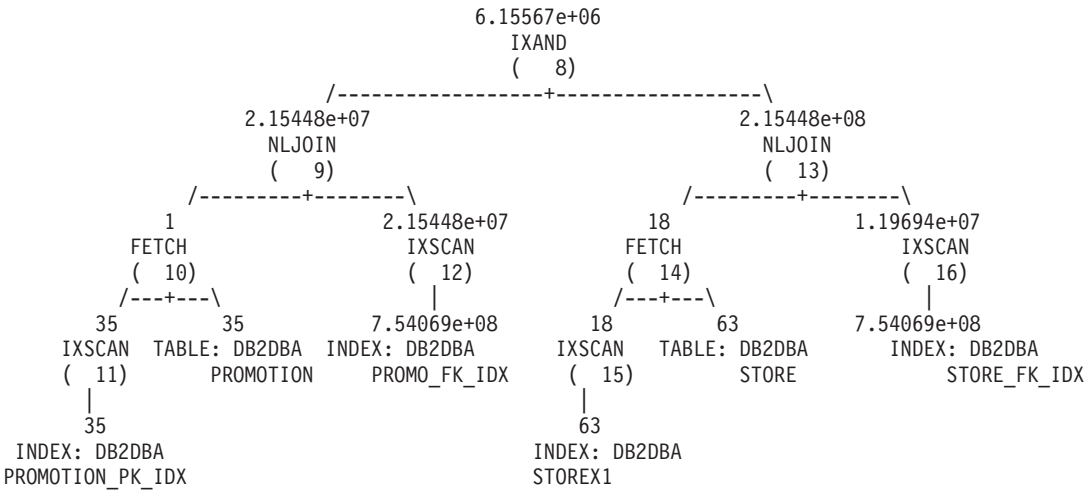
promotype	promodesc	COUNT（promotype）	在总数中所占的百分比
1	回头客	1	2.86%
2	优惠券	15	42.86%
3	广告	5	14.29%
4	经理特选	3	8.57%
5	库存过多的商品	4	11.43%
6	通道末端展示	7	20.00%

此表指出，回头客折扣仅占所提供的 35 种促销的 2.86%。

以下查询将返回计数 12889514:

```
select  count(*)
  from  store d1, promotion d2, daily_sales f
  where  d1.storekey = f.storekey
    and  d2.promokey = f.promokey
    and  d1.store_number = '01'
    and  d2.promotype = 1
```

此查询根据优化器生成的下列方案执行。在此图的每个节点中，第一行是估算的基数，第二行是运算符类型，第三行（括号中的数字）是运算符标识。



在嵌套循环连接（编号 9）中，优化器估计销售的产品中，大约 2.86% 来自顾客再次以折扣价格购买相同产品（2.15448e+07÷7.54069e+08≈0.0286）。请注意，此值在连接 PROMOTION 表与 DAILY_SALES 表前后没有变化。第 315 页的表 62 概述了连接前后的基数估计及其百分比（过滤效果）。

表 62. 与 DAILY_SALES 连接前后的基数估计。

	连接前		连接后	
谓词	计数	合格行百分比	计数	合格行百分比
store_number = '01'	18	28.57%	2.15448e+08	28.57%
promotype = 1	1	2.86%	2.15448e+07	2.86%

因为 promotype = 1 的可能性小于 store_number = '01' 的可能性，所以优化器选择 PROMOTION 与 DAILY_SALES 之间的半连接作为星型连接方案的索引 AND 运算的外支路。这将得出使用促销类型 1 大约售出 6155670 件产品的估算计数，但这不是正确的基数估算，其偏离因子为 2.09%（12889514÷6155670≈2.09）。

什么原因导致优化器只能估算出满足两个谓词的实际记录数的一半？商店“01”大约占所有商店的 28.57%。如果其他商店的销售量大于商店“01”又会怎样（小于 28.57% 吗）？或者，如果商店“01”实际销售了大部分产品又会怎样（大于 28.57% 吗）？同样，表 62 中显示的使用促销类型 1 销售的 2.86% 的产品可能是误导。DAILY_SALES 中的实际百分比很可能与显示的数字不同。

可以使用统计视图来帮助优化器更正其估算。首先，需要创建两个统计视图，分别表示先前查询中的每个半连接。第一个统计视图提供所有日常销售的商店的分布。第二个统计视图表示所有日常销售的促销类型的分布。请注意，每个统计视图都能提供有关任何特定商店编号或促销类型的分布信息。在本示例中，使用 10% 的采样率来检索各个视图的 DAILY_SALES 中的记录，并将这些记录保存在全局临时表中。然后，查询这些表以收集更新两个统计视图所必需的统计信息。

1. 创建一个视图来表示 STORE 与 DAILY_SALES 的连接。

```
create view sv_store_dailysales as
  (select s.*
    from store s, daily_sales ds
    where s.storekey = ds.storekey)
```

2. 创建一个视图来表示 PROMOTION 与 DAILY_SALES 的连接。

```
create view sv_promotion_dailysales as
  (select p.*
    from promotion.p, daily_sales ds
    where p.promokey = ds.promokey)
```

3. 通过启用视图来进行查询优化，从而使该视图成为统计视图:

```
alter view sv_store_dailysales enable query optimization
alter view sv_promotion_dailysales enable query optimization
```

4. 执行 RUNSTATS 命令，以收集视图的统计信息:

```
runstats on table db2dba.sv_store_dailysales with distribution
runstats on table db2dba.sv_promotion_dailysales with distribution
```

5. 再次运行查询，以便可以重新优化查询。进行重新优化之后，优化器将使 SV_STORE_DAILYSALES 和 SV_PROMOTION_DAILYSALES 与查询匹配，并使用视图统计信息来调整事实表与维表之间半连接的基数估计，这使得它反转在没有这些统计信息的情况下选择的半连接的原始顺序。新的方案如下所示:

```
                1.04627e+07
                  IXAND
                  (   8)
      /-----------------+-----------------\
6.99152e+07                               1.12845e+08
  NLJOIN                                    NLJOIN
  (   9)                                    (  13)
```

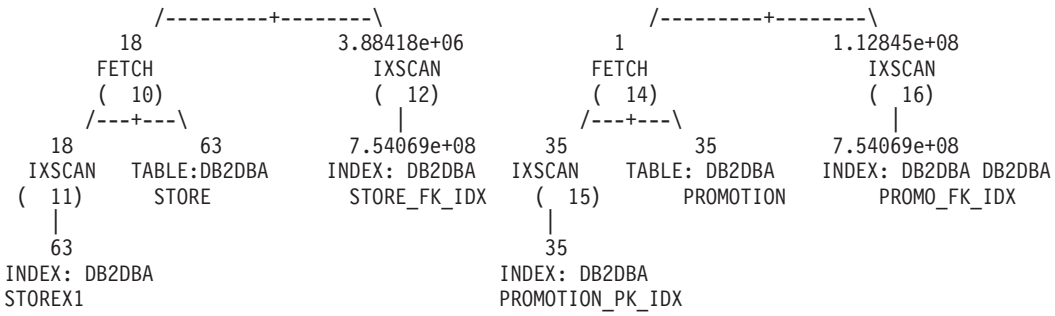


表 63 总结了每个半连接的连接前后的基数估计及其百分比（过滤效果）。

表 63. 与 DAILY_SALES 连接前后的基数估计。

	连接前		连接后（*没有统计视图*）		连接后（*有统计视图*）	
谓词	计数	合格行百分比	计数	合格行百分比	计数	合格行百分比
store_number = '01'	18	28.57%	2.15448e+08	28.57%	6.99152e+07	9.27%
promotype = 1	1	2.86%	2.15448e+07	2.86%	1.12845e+08	14.96%

请注意，此时 STORE 和 DAILY_SALES 之间的半连接在索引 AND 运算方案的外支路中执行。这是因为两个统计视图实质上告知优化器谓词 `store_number = '01'` 过滤的行数比 `promotype = 1` 要多。这一次，优化器估算的已售产品件数大约为 10462700。此估计的偏差因子为 1.23（12889514÷10462700≈1.23），这比不使用统计视图时的估算（第 315 页的表 62）准确很多。

目录统计信息

当查询编译器优化查询方案时，它所作的决策主要受有关数据库表大小、索引大小和统计视图大小的统计信息的影响。此信息存储在系统目录表中。

如果使用表、索引和统计视图的特定列来选择行或连接表，那么优化器还将使用有关这些列中的数据分布情况的信息。优化器使用此信息来估计每个查询的备用访问方案的成本。

还可以收集下列统计信息：索引的集群比率、索引中叶子页的数目、溢出其原始页的表行数以及表中的已填充页数和空页数。您可以使用此信息来决定何时重组表或索引。

在分区数据库环境中收集表统计信息时，将只收集运行实用程序所针对的数据库分区中那部分表的统计信息，或者只收集数据库分区组中第一个包含该表的数据库分区的统计信息。有关统计视图的信息将针对所有数据库分区进行收集。

RUNSTATS 实用程序所更新的统计信息

目录统计信息由 RUNSTATS 实用程序更新，您可以通过发出 RUNSTATS 命令、调用 ADMIN_CMD 过程或调用 db2Runstats API 来启动该实用程序。可以手动地启动更新，也可以自动启动更新。

有关已声明临时表的统计信息并非存储在系统目录中，而是存储在代表已声明临时表的目录信息的内存结构中。您可以对已声明临时表执行 RUNSTATS（在某些情况下，此功能很有用）。

RUNSTATS 实用程序将收集有关表和索引的以下信息:

- 包含行的页的数目
- 使用中的页的数目
- 表中的行数（*基数*）
- 溢出的行数
- 对于多维集群（MDC）表: 包含数据的块的数目
- 对于分区表: 单一数据分区中数据的集群度
- 数据分布统计信息 - 对于其中的数据未均匀分布并且其中的列包含大量重复值的表和统计视图，优化器将使用此统计信息来估计那些表和视图的有效访问方案
- 详细的索引统计信息 - 优化器将使用此统计信息来确定通过索引访问表数据的效率
- LIKE 谓词（特别是那些用于在字符串中搜索模式的 LIKE 谓词，例如 `LIKE %disk%`）的子元素统计信息 - 这些统计信息也由优化器使用

RUNSTATS 实用程序将收集表中每个数据分区的下列统计信息。这些统计信息仅用于确定是否需要对分区进行重组:

- 包含行的页的数目
- 使用中的页的数目
- 表中的行数（基数）
- 溢出的行数
- 对于 MDC 表，包含数据的块数

在下列情况下，不收集分布统计信息:

- **num_freqvalues** 和 **num_quantiles** 数据库配置参数设置为 0
- 数据的分布已知，例如，每个数据值都唯一时
- 列包含 LONG、LOB 或结构化数据类型
- 对于子表中的行类型（不收集表级别统计信息 NPAGES、FPAGES 和 OVERFLOW）
- 如果请求分位数分布，但该列只包含一个非空值
- 对于已扩展的索引或已声明临时表

RUNSTATS 实用程序将收集有关表或统计视图中每一列以及索引键中第一列的下列信息:

- 列的基数
- 列的平均长度（在数据库内存或临时表中存储该列时所需的平均空间量，以字节计）
- 列中的次高值
- 列中的次低值
- 列中的空值数

对于包含大对象（LOB）或 LONG 数据类型的列而言，RUNSTATS 实用程序将只收集该列的平均长度以及该列中的空值数。该列的平均长度代表数据描述符的长度，LOB 数据以直接插入方式存储到数据页时的情况除外。在磁盘上存储该列所需的平均空间量可能与此统计信息的值不同。

RUNSTATS 实用程序将收集有关每个 XML 列的下列信息:

- 空 XML 文档数
- 非空 XML 文档数
- 相异路径数
- 每条相异路径的节点数之和
- 每条相异路径的文档数之和
- 具有最大节点数的 *k* 对（路径，节点数）
- 具有最大文档数的 *k* 对（路径，文档数）
- 具有最大节点数的 *k* 个三元组（路径，值，节点数）
- 具有最大文档数的 *k* 个三元组（路径，值，文档数）
- 对于通往文本或属性值的每条相异路径:
  - 此路径可以接受的相异值数
  - 最大值
  - 最小值
  - 文本或属性节点数
  - 包含文本或属性节点的文档数

XML 列中的每一行都存储一个 XML 文档。路径或“路径/值”对的节点数是指该路径或“路径/值”对可达的节点数。路径或“路径/值”对的文档数是指包含该路径或“路径/值”对的文档数。

对于 DB2 V9.7 修订包 1 和更高版本的发行版，下列各项适用于收集 XML 列的分布统计信息:

- 收集基于 XML 列中所指定 XML 数据的每个索引的分布统计信息。
- RUNSTATS 实用程序必须同时收集分布统计信息和表统计信息，才能收集基于 XML 数据的索引的分布统计信息。由于 XML 分布统计信息与表统计信息存储在一起，因此必须收集表统计信息以便收集分布统计信息。

  如果仅收集索引统计信息或者在创建索引期间收集索引统计信息，那么将不会收集基于 XML 数据的索引的分布统计信息。

  缺省情况下，RUNSTATS 实用程序最多可以对每个基于 XML 数据的索引的分布统计信息收集 250 个分位数。可以在执行 RUNSTATS 实用程序时指定某列的最大分位数。
- 收集基于 XML 数据类型为 VARCHAR、DOUBLE、TIMESTAMP 和 DATE 的索引的分布统计信息。将不收集基于 XML 数据类型为 VARCHAR HASHED 的索引的 XML 分布统计信息。
- 将在对表自动执行 RUNSTATS 操作时收集 XML 分布统计信息。
- 使用 STATISTICS 选项来装入数据时不会创建 XML 分布统计信息。
- 将不收集基于分区表中所定义 XML 数据的分区索引的 XML 分布统计信息。

RUNSTATS 实用程序将收集有关列组的下列信息:

- 列组的基于时间戳记的名称
- 列组的基数

RUNSTATS 实用程序将收集有关索引的下列信息:

- 索引条目的数目（*索引基数*）
- 叶子页数
- 索引层数
- 表数据相对于索引的集群度
- 与数据分区相关的索引键的集群度
- 磁盘上按索引键顺序排列的叶子页与索引所占用页范围中页数的比率
- 索引的第一列中的相异值数
- 索引的前二、三和四列中的相异值数
- 索引的所有列中的相异值数
- 磁盘上按索引键顺序排列的叶子页的数目，在这些叶子页之间只有很少甚至没有大的间隔
- 平均叶子键大小，不含包括列
- 平均叶子键大小，含包括列
- 页中所有记录标识（RID）都被标记为“已删除”的页数
- 并非所有 RID 都被标记为“已删除”的页中已被标记为“已删除”的 RID 数

如果您请求收集详细的索引统计信息，那么还将收集关于表数据相对于索引的集群度的附加信息以及不同缓冲区大小的页访存估算值。

对于分区索引而言，这些统计信息与单一索引分区相关，但以下情况例外：索引的第一列中的相异值；索引的前两列、前三列和前四列中的相异值；以及索引的全部各列中的相异值。另外，还将收集每个索引分区的统计信息，以便确定是否需要对索引分区进行重组。

目录统计表

有关数据库表大小、索引大小和统计视图大小的统计信息存储在系统目录表中。

下列各表提供有关此统计信息的简要描述及其存储位置。

- “表”列指示，未指定 RUNSTATS 命令的 FOR INDEXES 或 AND INDEXES 选项时是否收集特定统计信息。
- “索引”列指示，指定了 FOR INDEXES 或 AND INDEXES 选项时是否收集特定统计信息。

某些统计信息只能由表提供，某些只能由索引提供，某些可以由这两者提供。

- 表 1. 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES）
- 表 2. 列统计信息（SYSCAT.COLUMNS 和 SYSSTAT.COLUMNS）
- 表 3. 多列统计信息（SYSCAT.COLGROUPS 和 SYSSTAT.COLGROUPS）
- 表 4. 多列分布统计信息（SYSCAT.COLGROUPDIST 和 SYSSTAT.COLGROUPDIST）
- 表 5. 多列分布统计信息（SYSCAT.COLGROUPDISTCOUNTS 和 SYSSTAT.COLGROUPDISTCOUNTS）
- 表 6. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES）
- 表 7. 列分布统计信息（SYSCAT.COLDIST 和 SYSSTAT.COLDIST）

表 4. 多列分布统计信息（SYSCAT.COLGROUPDIST 和 SYSSTAT.COLGROUPDIST）以及表 5. 多列分布统计信息（SYSCAT.COLGROUPDISTCOUNTS 和 SYSSTAT.COLGROUPDISTCOUNTS）中列示的多列分布统计信息并非由 RUNSTATS 实用程序收集。您无法手动更新这些统计信息。

表 64. 表统计信息（SYSCAT.TABLES 和 SYSSTAT.TABLES）

统计信息	描述	RUNSTATS 选项	
		表	索引
FPAGES	表所使用的页数	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）已占用的块的总数	是	否

注:

1. 如果尚未定义该表的索引，而您请求获取索引统计信息，那么不会更新 CARD 统计信息。而是，将保留先前的 CARD 统计信息。

表 65. 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	列中的次大值	是	是（注 1）
LOW2KEY	列中的次小值	是	是（注 1）
NUMNULLS	列中的空值数	是	是（注 1）
SUB_COUNT	子元素的平均数目	是	否（注 2）
SUB_DELIM_LENGTH	每个用于分隔子元素的定界符的平均长度	是	否（注 2）

注:

1. 列统计信息是针对索引键中的第一列收集的。
2. 这些统计信息提供关于列中数据的信息，这些列包含一系列由空格定界的子字段或子元素。将只对类型为 CHAR 和 VARCHAR 并且代码页属性为单字节字符集（SBCS）、FOR BIT DATA 或 UTF-8 的列收集 SUB_COUNT 和 SUB_DELIM_LENGTH 统计信息。

表 66. 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

表 67. 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 = 高频值 Q = 分位数值	是	否
ORDINAL	组中列的序数	是	否
SEQNO	序号 *n*，它表示第 *n* 个 TYPE 值	是	否
COLVALUE	作为字符文字的数据值或者空值	是	否

表 68. 多列分布统计信息（SYSCAT.COLGROUPDISTCOUNTS 和 SYSSTAT.COLGROUPDISTCOUNTS）

统计信息	描述	RUNSTATS 选项	
		表	索引
TYPE	F = 高频值 Q = 分位数值	是	否
SEQNO	序号 *n*，它表示第 *n* 个 TYPE 值	是	否
VALCOUNT	如果 TYPE = F，那么 VALCOUNT 是具有此 SEQNO 的列组的 COLVALUE 出现次数。如果 TYPE = Q，那么 VALCOUNT 是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，那么此列包含小于或等于具有此 SEQNO 的列组的 COLVALUE 的相异值数目。如果无法获得此数目，那么为空。	是	否

表 69. 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	表数据的集群度	否	是（注 2）
CLUSTERFACTOR	更精确的集群度	否	详细（注 1 和 2）

表 69. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES）(续)

统计信息	描述	RUNSTATS 选项	
		表	索引
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	索引的第一列中的相异值数	否	是
FIRST2KEYCARD	索引的前两列中的相异值数	否	是
FIRST3KEYCARD	索引的前三列中的相异值数	否	是
FIRST4KEYCARD	索引的前四列中的相异值数	否	是
FULLKEYCARD	索引的所有各列中的相异值数，不包括其所有记录标识（RID）都已被标记为“已删除”的索引中的任何键值	否	是
PAGE_FETCH_PAIRS	不同缓冲区大小的页访存估算值	否	详细（注 1 和 2）
AVGPARTITION_CLUSTERRATIO	单个数据分区内数据的集群度	否	是（注 2）
AVGPARTITION_CLUSTERFACTOR	单个数据分区内集群度的更精确度量	否	详细（注 1 和 2）
AVGPARTITION_PAGE_FETCH_PAIRS	根据单个数据分区生成的不同缓冲区大小的页访存估算值	否	详细（注 1 和 2）
DATAPARTITION_CLUSTERFACTOR	扫描索引期间的数据分区引用数	否（注 6）	是（注 6）
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	可按顺序访问的索引页的平均数目；这是可以被预取程序检测为按顺序排列的索引页数	否	是
AVERAGE_RANDOM_PAGES	在两次顺序页访问之间随机访问的索引页的平均数目	否	是
AVERAGE_SEQUENCE_GAP	序列之间的间隔	否	是

表 69. 索引统计信息（SYSCAT.INDEXES 和 SYSSTAT.INDEXES）(续)

统计信息	描述	RUNSTATS 选项	
		表	索引
AVERAGE_SEQUENCE_FETCH_PAGES	可按顺序访问的表页的平均数目；这是预取程序使用索引来访存表行时可以被预取程序检测为按顺序排列的表页数	否	是（注 4）
AVERAGE_RANDOM_FETCH_PAGES	使用索引来访存表行时，在两次顺序页访问之间随机访问的表页的平均数目	否	是（注 4）
AVERAGE_SEQUENCE_FETCH_GAP	使用索引来访存表行时，序列之间的间隔	否	是（注 4）
NUMRIDS	索引中的 RID 的数目，其中包括已删除的 RID	否	是
NUMRIDS_DELETED	索引中已被标记为“已删除”的 RID 的总数，不包括其中所有 RID 都被标记为“已删除”的叶子页中的 RID	否	是
NUM_EMPTY_LEAFS	其中所有 RID 都被标记为“已删除”的叶子页的总数	否	是
INDCARD	索引条目的数目（索引基数）	否	是

注:

1. 要收集详细的索引统计信息，请在 RUNSTATS 命令中指定 DETAILED 子句。
2. 除非表足够大（约大于 25 页），否则不使用 DETAILED 子句来收集 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS。在这种情况下，CLUSTERRATIO 是 -1（未收集）。如果表相对较小，那么 RUNSTATS 实用程序只收集 CLUSTERRATIO，而不会收集 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS。如果未指定 DETAILED 子句，那么只收集 CLUSTERRATIO。
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而言，DENSITY 应为 100。优化器使用 DENSITY 来估计如果预取了索引页，那么平均可能从其他索引读取多少个不相关的页。
4. 当表在 DMS 表空间中时，无法计算此统计信息。
5. 即使调用命令时指定要收集统计信息，在装入或创建索引操作期间也不会收集预取统计信息。如果 **seqdetect** 数据库配置参数设置为 NO，那么也不会收集预取统计信息。
6. 如果表的 RUNSTATS 选项为“否”，那么收集表统计信息时不收集统计信息；如果索引的 RUNSTATS 选项为“是”，那么使用带 INDEXES 选项的 RUNSTATS 命令时将收集统计信息。

表 70. 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 = Q，那么 DISTCOUNT 是小于或等于 COLVALUE 统计信息的相异值的数目	分布（注 2）	否
TYPE	指示行是提供高频值还是分位数统计信息	分布	否
SEQNO	序号的频率排名，用于帮助在表中唯一地标识该行	分布	否
COLVALUE	所收集的频率或分位数统计信息的相应数据值	分布	否
VALCOUNT	数据值在列中的出现频率；对于分位数，这是小于或等于该数据值（COLVALUE）的值的数目	分布	否

注:

1. 要收集列分布统计信息，请在 RUNSTATS 命令中指定 WITH DISTRIBUTION 子句。除非列值严重不均匀，否则无法收集分布统计信息。
2. 仅对索引中的第一个键列收集 DISTCOUNT。

自动收集统计信息

DB2 优化器使用目录统计信息来确定查询的最佳访问方案。过期或者不完整的表或索引统计信息可能会导致优化器选择并非最佳的方案，从而导致查询执行速度下降。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，使这些统计信息保持最新也是一项很花费时间的任务。

通过使用自动收集统计信息功能（这是 DB2 的自动表维护功能的组成部分），可以让数据库管理器确定是否需要更新统计信息。可以使用实时统计信息（RTS）功能在编译语句时*以同步方式*执行自动收集统计信息操作，也可以启用 RUNSTATS 实用程序以便在后台运行此程序以执行*同步*收集。虽然可以在实时收集统计信息功能处于禁用状态时启用后台收集统计信息功能，但必须启用后台收集统计信息功能才能执行实时收集统计信息功能。缺省情况下，当您创建新的数据库时，就会启用后台自动收集统计信息功能。要启用实时自动收集统计信息功能，请将 **auto_stmt_stats** 数据库配置参数的值设置为 ON。此参数是 **auto_runstats** 配置参数的子代。

了解异步收集和实时收集统计信息

启用实时收集统计信息功能之后，还可以使用一些元数据来生成统计信息。*生成*表示派生或创建统计信息，而不是作为正常 RUNSTATS 活动的一部分来收集统计信息。例如，如果知道一个表中的行数、页大小和平均行宽，那么可以知道该表中包含的行

数。在某些情况下，并未真正派生统计信息，而是由索引和数据管理器维护统计信息，并且可以直接将这些统计信息存储在目录中。例如，索引管理器将在每个索引中维护叶子页数和层数。

查询优化器根据查询需求和表更新活动数量（更新、插入或删除操作的数目）来确定统计信息的收集方式。

实时收集统计信息功能能够提供更及时更准确的统计信息。准确的统计信息可以产生更好的查询执行方案并提高性能。未启用实时收集统计信息功能时，将每隔两小时执行一次异步收集统计信息操作。要为某些应用程序提供准确的统计信息，此收集频率可能不够高。

在启用实时收集统计信息功能之后，仍将每隔两个小时执行一次异步收集统计信息检查操作。在下列情况下，实时收集统计信息功能还将引起发出异步收集请求:

- 表活动频率对于同步收集而言不够高，但对于异步收集而言却又过高
- 由于表太大，因此同步收集统计信息操作进行了采样
- 已生成同步统计信息
- 同步收集统计信息操作由于超出收集时间而失败

最多可以同时处理两个异步请求，但必须针对不同的表处理这些请求。一个请求必须由实时收集统计信息功能发起，而另一个请求必须由异步收集统计信息检查操作发起。

通过使用下面几种方法，可以将自动收集统计信息功能对性能的影响降到最低:

- 通过使用已调速的 RUNSTATS 实用程序执行异步收集统计信息操作。调速功能根据当前数据库活动来控制 RUNSTATS 实用程序耗用的资源量: 随着数据库活动的增加，此实用程序的运行速度将变慢，从而减少资源需求。
- 对于每个查询，同步收集统计信息操作的时间限制为 5 秒。此值可由 RTS 优化准则控制。如果同步收集操作超出时间限制，那么将提交异步收集请求。
- 同步收集统计信息操作不会将统计信息存储在系统目录中。而是，此操作将统计信息存储在统计信息高速缓存中，以后再通过异步操作存储在系统目录中。这样就可以避免更新系统目录时产生的开销和可能的锁定争用。后续 SQL 编译请求可以使用统计信息高速缓存中的统计信息。
- 对于每个表，只能执行一个同步收集统计信息操作。其他需要执行同步收集统计信息操作的代理程序将在可能的情况下生成统计信息，并继续编译语句。在分区数据库环境中也会强制执行此行为，此环境中的不同数据库分区中的代理程序可能需要同步统计信息。
- 要定制所收集的统计信息的类型，您可以启用统计信息概要分析功能（此功能使用有关先前数据库活动的信息来确定数据库工作负载需要的统计信息），也可以为特定的表创建自己的统计信息概要文件。
- 只对缺少统计信息或具有高级别活动（根据更新、插入或删除操作的数目衡量）的表收集统计信息。即使表符合统计信息收集条件，也不会收集同步统计信息，除非查询优化需要这些信息。在某些情况下，查询优化器可以在不使用统计信息的情况下选择访问方案。
- 对于异步收集统计信息检查操作而言，还将对大型表（超过 4000 页的表）进行采样，以确定高级表活动是否更改了统计信息。只有在保证已更改统计信息的情况下，才会收集这种大型表的统计信息。

- 对于异步收集统计信息操作而言，自动将 RUNSTATS 实用程序安排在维护策略中指定的最佳维护时间段运行。此策略还指定自动收集统计信息作用域内的一组表，这进一步减少了不必要的资源消耗。
- 同步收集和生成统计信息操作不会按照维护策略指定的最佳维护时间段进行，这是因为同步请求必须立即执行并且收集时间有限。同步收集和生成统计信息操作遵循特定策略，该策略指定了自动收集统计信息作用域内的一组表。
- 执行自动收集统计信息操作时，受影响的表仍然可用于常规的数据库活动（更新、插入或删除操作）。
- 不收集昵称的实时统计信息（同步统计信息或生成的统计信息）。要为异步统计信息收集操作自动刷新系统目录中的昵称统计信息，请调用 SYSPROC.NNSTAT 过程。

对常规表、具体化查询表（MQT）和全局临时表执行实时同步统计信息收集操作。不收集全局临时表的异步统计信息。

不会对下列对象执行自动收集统计信息（同步或异步）操作:

- 统计视图
- 已被标记为 VOLATILE 的表（在 SYSCAT.TABLES 目录视图中设置了 VOLATILE 字段的表）
- 您已通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新其统计信息的表

  以手动方式修改表统计信息时，数据库管理器假定您现在负责维护其统计信息。为了让数据库管理器维护已经以手动方式更新其统计信息的表的统计信息，请使用 RUNSTATS 命令来收集统计信息，或者在使用 LOAD 命令时指定收集统计信息。在版本 9.5 之前创建并且在升级前已手动更新其统计信息的表不受影响，其统计信息将由数据库管理器自动维护，直到您以手动方式更新这些统计信息为止。

不会对下列对象生成统计信息:

- 统计视图
- 您已通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新其统计信息的表。注意，即使未启用实时收集统计信息功能，也仍然会对已手动更新其统计信息的表生成某些统计信息。

在分区数据库环境中，将在单个数据库分区中收集统计信息然后进行推测。数据库管理器始终在数据库分区组的第一个数据库分区中收集统计信息（同步和异步）。

至少在数据库激活 5 分钟之后才会执行非实时统计信息收集活动。

启用实时收集统计信息功能之后，应安排已定义的维护时间段；缺省情况下，未定义维护时间段。如果没有已定义的维护时间段，那么将只执行同步统计信息收集操作。在这种情况下，收集的统计信息仅保存在内存中，并且通常以采样方式收集（小型表除外）。

将对静态和动态 SQL 执行实时统计信息处理。

已使用 IMPORT 命令截断的表将自动重组为具有旧统计信息。

对于已收集其统计信息的表，同步和异步自动收集统计信息操作将使引用了这些表并已进行高速缓存的动态语句失效。这样做是为了能够使用最新统计信息来重新优化已进行高速缓存的动态语句。

实时统计信息和说明处理

不会对仅仅使用说明工具进行说明（而未执行）的查询进行实时处理。下表概述了 CURRENT EXPLAIN MODE 专用寄存器具有不同值时的行为。

表 71. 作为 CURRENT EXPLAIN MODE 专用寄存器值的应变量的实时收集统计信息操作

CURRENT EXPLAIN MODE 值	是否考虑实时收集统计信息
YES	是
EXPLAIN	否
NO	是
REOPT	是
RECOMMEND INDEXES	否
EVALUATE INDEXES	否

自动收集统计信息和统计信息高速缓存

DB2 版本 9.5 引入了统计信息高速缓存，以便使以同步方式收集的统计信息可用于所有查询。此高速缓存是目录高速缓存的一部分。在分区数据库环境中，此高速缓存仅存在于目录数据库分区中。目录高速缓存可以存储同一个 SYSTABLES 对象的多个条目，这将增大所有数据库分区中的目录高速缓存大小。在启用实时收集统计信息功能的情况下，应考虑增大 **catalogcache_sz** 数据库配置参数的值。

从 DB2 版本 9 开始，可以使用配置顾问程序来确定新数据库的初始配置。配置顾问程序建议将 **auto_stmt_stats** 数据库配置参数设置为 ON。

自动收集统计信息和统计概要文件

同步和异步统计信息是根据表的有效统计概要文件收集的，但下列情况例外:

- 为了使同步收集统计信息操作的开销降至最低，数据库管理器可以通过进行采样来收集统计信息。在这种情况下，采样率和方法可能与统计概要文件中指定的采样率和方法不同。
- 同步收集统计信息操作可以选择生成统计信息，但是可能无法生成统计概要文件中指定的所有统计信息。例如，无法生成 COLCARD、HIGH2KEY 和 LOW2KEY 之类的列统计信息，除非该列在某些索引中是前导列。

如果同步收集统计信息操作无法收集统计概要文件中指定的所有统计信息，那么将提交异步收集请求。

尽管实时收集统计信息功能旨在使收集统计信息的开销降至最低，但是请先在测试环境中尝试使用此功能，以确保不会对性能产生负面影响。对某些联机事务处理（OLTP）方案使用此功能可能会对性能产生负面影响，对查询运行时间设置了上限的情况下尤其如此。

允许自动收集统计信息:

为了进行高效率的数据访问和实现最佳工作负载性能，具有准确而完整的数据库统计信息极为关键。您可以使用自动化表维护功能的自动收集统计信息功能来更新和维护相关的数据库统计信息。

对于在单一处理器上运行单一数据库分区的环境，可以通过以下方法增强此功能：收集查询数据并生成统计信息概要文件，这可以帮助 DB2 服务器自动收集工作负载所需的一组准确统计信息。在分区数据库环境、某些联合数据库环境或者启用了分区内并行性的环境中，此选项不可用。

要允许自动收集统计信息，请执行下列操作：

1. 通过将 **auto_maint**、**auto_tbl_maint** 和 **auto_runstats** 数据库配置参数设置为 ON 来配置数据库实例。 **auto_maint** 参数是 **auto_tbl_maint** 和 **auto_runstats** 的父代。
2. 可选：要允许自动生成统计信息概要文件，请将 **auto_stats_prof** 和 **auto_prof_upd** 数据库配置参数设置为 ON。 **auto_maint** 参数是 **auto_stats_prof** 和 **auto_prof_upd** 的父代。
3. 可选：要允许收集实时统计信息，请将 **auto_stmt_stats** 配置参数设置为 ON。 每当需要表统计信息以优化查询时，都会在编译语句时自动收集这些统计信息。**auto_maint** 参数是 **auto_stmt_stats** 的父代。

使用统计信息概要文件来收集统计信息：

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，该文件指定要对特定表收集的统计信息的类型；例如，表统计信息、索引统计信息或分布统计信息。此功能使您能够存储 RUNSTATS 选项以方便将来使用，从而简化了统计信息收集工作。

要同时注册概要文件和收集统计信息，请发出带有 SET PROFILE 选项的 RUNSTATS 命令。要仅注册概要文件，请发出带有 SET PROFILE ONLY 选项的 RUNSTATS 命令。要使用已注册的概要文件来收集统计信息，请发出带有 USE PROFILE 选项的 RUNSTATS 命令。

要了解特定表的统计信息概要文件中当前指定的选项，请查询 SYSCAT.TABLES 目录视图。例如：

```
SELECT STATISTICS_PROFILE FROM SYSCAT.TABLES WHERE TABNAME = 'EMPLOYEE'
```

自动进行统计信息概要分析

统计信息概要文件也可以通过 DB2 自动统计信息概要分析功能自动生成。启用此功能之后，有关数据库活动的信息将被收集并存储到查询反馈仓库中。然后，将根据此数据来生成统计信息概要文件。启用此功能可以缓解统计信息是否与特定工作负载相关的不确定性问题。

自动统计信息概要分析功能可以与自动收集统计信息功能配合使用，后一项功能根据自动生成的统计信息概要文件中包含的信息来安排统计信息维护操作。

要启用自动统计信息概要分析功能，请确保已通过设置适当的数据库配置参数启用了自动表维护功能。有关更多信息，请参阅“auto_maint - 自动维护配置参数”。**auto_stats_prof** 配置参数将激活查询反馈数据收集操作，而 **auto_prof_upd** 配置参数将激活统计信息概要文件生成功能以供自动收集统计信息功能使用。

在分区数据库环境、某些联合数据库环境或者分区内并行性处于启用状态的环境中，不支持自动生成统计信息概要文件。

自动统计信息概要分析功能最适合于需要运行包含许多谓词、使用了大型连接或指定了大量分组的大型复杂查询的系统。此功能不太适合于主要处理事务性工作负载的系统。

在可以轻易承受运行时监视功能的性能开销的开发环境中，请将 **auto_stats_prof** 和 **auto_prof_upd** 配置参数设置为 ON。当测试系统使用现实的数据和查询时，可以将适当的统计信息概要文件传送至生产系统以使查询受益，并且不会产生附加的监视开销。

在生产环境中，如果检测到与一组特定查询相关的性能问题（由于统计信息不正确而引起的问题），那么可以将 **auto_stats_prof** 配置参数设置为 ON 并将目标工作负载执行一段时间。自动统计信息概要分析功能将分析查询反馈并在 SYSTOOLS.OPT_FEEDBACK_RANKING 表中创建建议。您可以检查这些建议并相应地手动优化统计信息概要文件。要让 DB2 服务器按照这些建议自动更新统计信息概要文件，请在启用 **auto_stats_prof** 时启用 **auto_prof_upd**。

创建查询反馈仓库

自动统计信息概要分析功能所必需的查询反馈仓库由 SYSTOOLS 模式中的五个表组成。这些表存储关于查询执行期间所遇到的谓词的信息以及关于如何收集统计信息的建议。这五个表是:

- OPT_FEEDBACK_PREDICATE
- OPT_FEEDBACK_PREDICATE_COLUMN
- OPT_FEEDBACK_QUERY
- OPT_FEEDBACK_RANKING
- OPT_FEEDBACK_RANKING_COLUMN

请使用 SYSINSTALLOBJECTS 过程来创建查询反馈仓库。此过程用于在 SYSTOOLS 模式中创建或删除对象，有关此过程的更多信息，请参阅“SYSINSTALLOBJECTS”。

自动收集统计信息和概要分析功能使用的存储器:

自动收集统计信息和重组功能将工作数据存储在数据库的表中。这些表是在 SYSTOOLSPACE 表空间中创建的。

激活数据库时，将自动使用缺省选项来创建 SYSTOOLSPACE。这些表的存储器需求与数据库中表的数目成比例，可以按每个表大约 1 KB 来估算。如果这对于您的数据库而言过大，那么您可能想删除并亲自重新创建表空间，从而适当地分配存储器。虽然将自动重新创建表空间中的自动维护和运行状况监视表，但当您删除表空间时，在那些表中捕获的任何历史记录都将丢失。

对自动收集统计信息活动进行日志记录:

统计信息日志是对特定数据库进行的所有统计信息收集活动（既包括手动活动也包括自动活动）的记录。

统计信息日志的缺省名称为 db2optstats.*number*.log。此日志在 $DIAGPATH/events 目录中。统计信息日志是旋转日志。日志行为由 **DB2_OPTSTATS_LOG** 注册表变量控制。

您可以直接查看统计信息日志，也可以使用 SYSPROC.PD_GET_DIAG_HIST 表函数对其进行查询。此表函数将返回一定数目的列，这些列包含关于任何已记录的事件的标准信息，例如时间戳记、DB2 实例名称、数据库名称、进程标识、进程名称和线程标识。日志还包含供不同日志记录功能使用的通用列。下表描述了统计信息日志如何使用这些通用列。

表 72. 统计信息日志文件中的通用列

列名	数据类型	描述
OBJTYPE	VARCHAR(64)	事件所应用于的对象的类型。对于统计信息日志记录，这是要收集的统计信息的类型。OBJTYPE 可以在统计信息收集后台进程启动或停止时引用该进程。它还可以引用由自动收集统计信息功能执行的活动，例如采样测试、初始采样和表求值活动。 可能的统计信息收集活动值包括: **TABLE STATS** 将收集表统计信息。 **INDEX STATS** 将收集索引统计信息。 **TABLE AND INDEX STATS** 将同时收集表统计信息和索引统计信息。 可能的自动收集统计信息值包括: **EVALUATION** 自动收集统计信息后台进程已开始求值阶段。在此阶段，将检查表以确定它们是否需要更新后的统计信息，并在必要时收集统计信息。 **INITIAL SAMPLING** 正在使用采样方法来收集表的统计信息。所采样的统计信息将存储在系统目录中。这将允许自动收集统计信息功能快速处理没有统计信息的表。后续操作将在不进行采样的情况下收集统计信息。初始采样在自动收集统计信息的求值阶段执行。 **SAMPLING TEST** 正在使用采样方法来收集表的统计信息。所采样的统计信息将不会存储在系统目录中。所采样的统计信息将与当前目录统计信息进行比较，以便确定是否以及何时应收集此表的完整统计信息。采样在自动收集统计信息过程的求值阶段执行。 **STATS DAEMON** 统计信息守护程序是用来处理实时统计信息处理所提交的请求的后台进程。当后台进程启动或停止时，将对此对象类型进行日志记录。

表 72. 统计信息日志文件中的通用列 (续)

列名	数据类型	描述
OBJNAME	VARCHAR(255)	事件所应用于的对象的名称（如果可用）。对于统计信息日志记录，这是表名或索引名。如果 OBJTYPE 是 STATS DAEMON 或 EVALUATION，那么 OBJNAME 是数据库名称，而 OBJNAME_QUALIFIER 是 NULL。
OBJNAME_QUALIFIER	VARCHAR(255)	对于统计信息日志记录，这是表或索引的模式。
EVENTTYPE	VARCHAR(24)	事件类型是与此事件相关联的操作。统计信息日志记录的可能值包括: **COLLECT** 为统计信息收集操作对此操作进行日志记录。 **START** 当实时统计信息后台进程（OBJTYPE = STATS DAEMON）或自动收集统计信息求值阶段（OBJTYPE = EVALUATION）启动时，对此操作进行日志记录。 **STOP** 当实时统计信息后台进程（OBJTYPE = STATS DAEMON）或自动收集统计信息求值阶段（OBJTYPE = EVALUATION）停止时，对此操作进行日志记录。 **ACCESS** 当尝试访问一个表以收集统计信息时，将记录此操作。当对象不可用时，此事件类型用于对不成功的访问尝试进行日志记录。 **WRITE** 将先前已收集并且存储在统计信息高速缓存中的统计信息写入系统目录时，将记录此操作。
FIRST_EVENTQUALIFIERTYPE	VARCHAR(64)	第一个事件限定符的类型。事件限定符用于描述受事件影响的对象。对于统计信息日志记录，第一个事件限定符是事件发生时的时间戳记。对于第一个事件限定符类型，值为 AT。
FIRST_EVENTQUALIFIER	CLOB(16k)	事件的第一个限定符。对于统计信息日志记录，第一个事件限定符是统计信息事件发生时的时间戳记。统计信息事件的时间戳记可能与 TIMESTAMP 列所表示的日志记录的时间戳记不同。
SECOND_EVENTQUALIFIERTYPE	VARCHAR(64)	第二个事件限定符的类型。对于统计信息日志记录，值可以为 BY 或 NULL。此字段不用于其他事件类型。

表 72. 统计信息日志文件中的通用列 (续)

列名	数据类型	描述
SECOND_EVENTQUALIFIER	CLOB(16k)	事件的第二个限定符。对于统计信息日志记录，它表示收集 COLLECT 事件类型的统计信息的方式。可能的值包括: **用户** 统计信息收集操作由 DB2 用户通过调用 LOAD、REDISTRIBUTE 或 RUNSTATS 命令或通过发出 CREATE INDEX 语句执行。 **同步** 统计信息收集操作由 DB2 服务器在编译 SQL 语句时执行。统计信息将存储在统计信息高速缓存中，而不是系统目录中。 **同步采样** 统计信息收集操作由 DB2 服务器在编译 SQL 语句时使用采样方法执行。统计信息将存储在统计信息高速缓存中，而不是系统目录中。 **生成** 在编译 SQL 语句时通过使用数据和索引管理器维护的信息来生成统计信息。统计信息将存储在统计信息高速缓存中，而不是系统目录中。 **部分生成** 在编译 SQL 语句时通过使用数据和索引管理器维护的信息仅生成某些统计信息。特别是，仅生成某些列的 HIGH2KEY 和 LOW2KEY 值。统计信息将存储在统计信息高速缓存中，而不是系统目录中。 **异步** 统计信息由 DB2 后台进程收集并存储在系统目录中。 此字段不用于其他事件类型。
THIRD_EVENTQUALIFIERTYPE	VARCHAR(64)	第三个事件限定符的类型。对于统计信息日志记录，值可以为 DUE TO 或 NULL。

表 72. 统计信息日志文件中的通用列 (续)

列名	数据类型	描述
THIRD_EVENTQUALIFIER	CLOB(16k)	事件的第三个限定符。对于统计信息日志记录，它表示统计信息活动无法完成的原因。可能的值包括: **超时** 同步收集统计信息操作超过时间预算。 **错误** 由于发生错误，统计信息活动失败。 **RUNSTATS 错误** 由于发生 RUNSTATS 错误，同步收集统计信息失败。对于某些错误，即使未能收集统计信息，SQL 语句编译操作也可能成功完成。例如，在没有足够的内存来收集统计信息时，将继续编译 SQL 语句。 **对象不可用** 由于无法访问数据库对象，未能收集该对象的统计信息。某些可能的原因包括: • 对象以超级互斥（Z）方式被锁定 • 对象所在的表空间不可用 • 需要重新创建表索引 **冲突** 由于另一个应用程序已在收集同步统计信息，所以未能执行同步统计信息收集。 请检查 FULLREC 列或 db2diag 日志文件以了解错误详细信息。
EVENTSTATE	VARCHAR(255)	由于事件导致的对象或操作的状态。对于统计信息日志记录，它指示统计信息操作的状态。可能的值包括: • 开始 • 成功 • 失败

示例

在此示例中，查询通过调用 PD_GET_DIAG_HIST 返回最多当前时间戳记前一年的事件的统计信息日志记录。

```
select pid, tid,
    substr(eventtype, 1, 10),
    substr(objtype, 1, 30) as objtype,
    substr(objname_qualifier, 1, 20) as objschema,
    substr(objname, 1, 10) as objname,
    substr(first_eventqualifier, 1, 26) as event1,
    substr(second_eventqualifiertype, 1, 2) as event2_type,
    substr(second_eventqualifier, 1, 20) as event2,
    substr(third_eventqualifiertype, 1, 6) as event3_type,
    substr(third_eventqualifier, 1, 15) as event3,
    substr(eventstate, 1, 20) as eventstate
  from table(sysproc.pd_get_diag_hist
    ('optstats', 'EX', 'NONE',
      current_timestamp - 1 year, cast(null as timestamp))) as sl
  order by timestamp(varchar(substr(first_eventqualifier, 1, 26), 26));
```

结果按 FIRST_EVENTQUALIFIER 列中存储的时间戳记进行排序，该时间戳记表示统计信息事件的时间。

```
PID   TID          EVENTTYPE OBJTYPE              OBJSCHEMA OBJNAME    EVENT1                    EVENT2_ EVENT2              EVENT3_ EVENT3   EVENTSTATE
                                                                                                  TYPE                        TYPE
----- ------------ --------- -------------------  --------- ---------- ------------------------- ------- ------------------- ------- -------- ----------
28399  1082145120   START      STATS DAEMON    -   PROD_DB  2007-07-09-18.37.40.398905 -   -   -     -   成功
28389  183182027104 COLLECT  TABLE AND INDEX STATS   DB2USER   DISTRICT   2007-07-09-18.37.43.261222 BY  同步      -       -      启动
28389  183182027104 COLLECT  TABLE AND INDEX STATS   DB2USER   DISTRICT   2007-07-09-18.37.43.407447 BY  异步      -       -      成功
28399  1082145120   COLLECT  TABLE AND INDEX STATS   DB2USER   CUSTOMER   2007-07-09-18.37.43.471614 BY  异步      -       -      启动
28399  1082145120   COLLECT  TABLE AND INDEX STATS   DB2USER   CUSTOMER   2007-07-09-18.37.43.524496 BY  异步      -       -      成功
28399  1082145120   STOP     STATS DAEMON -                    PROD_DB    2007-07-09-18.37.43.526212 -    -         -       -      成功
28389  183278496096 COLLECT  TABLE STATS             DB2USER   ORDER_LINE 2007-07-09-18.37.48.676524 BY  同步采样  -       -      启动
28389  183278496096 COLLECT  TABLE STATS             DB2USER   ORDER_LINE 2007-07-09-18.37.53.677546 BY  同步采样  DUE TO  超时   失败
28389  1772561034   START    EVALUATION                -         PROD_DB    2007-07-10-12.36.11.092739 -    -         -       -      成功
28389  8231991291   COLLECT  TABLE AND INDEX STATS   DB2USER   DISTRICT   2007-07-10-12.36.30.737603 BY   异步      -       -      启动
28389  8231991291   COLLECT  TABLE AND INDEX STATS   DB2USER   DISTRICT   2007-07-10-12.36.34.029756 BY   异步      -       -      成功
28389  1772561034   STOP     EVALUATION                -         PROD_DB    2007-07-10-12.36.39.685188 -    -         -       -      成功
28399    1504428165 START     STATS DAEMON            -         PROD_DB    2007-07-10-12.37.43.319291 -      -                 -       -      成功
28399    1504428165 COLLECT   TABLE AND INDEX STATS  DB2USER   CUSTOMER   2007-07-10-12.37.43.471614 BY      异步        -     -       启动
28399    1504428165 COLLECT   TABLE AND INDEX STATS  DB2USER   CUSTOMER   2007-07-10-12.37.44.524496 BY      异步        -     -       故障
28399    1504428165 STOP      STATS DAEMON           -         PROD_DB    2007-07-10-12.37.45.905975 -       -                 -       -      成功
28399    4769515044 START     STATS DAEMON           -         PROD_DB    2007-07-10-12.48.33.319291 -       -                 -       -      成功
28389    4769515044 WRITE     TABLE AND INDEX STATS  DB2USER   CUSTOMER   2007-07-10-12.48.33.969888 BY      异步        -     -       启动
28389    4769515044 WRITE     TABLE AND INDEX STATS  DB2USER   CUSTOMER   2007-07-10-12.48.34.215230 BY      异步        -     -       成功
```

提高大型统计信息日志的查询性能:

如果统计信息日志文件非常大，那么通过将日志记录复制到表中、创建索引并收集统计信息，可以提高查询性能。

过程

1. 为日志记录创建包含适当的列的表。

```
create table db2user.stats_log (
  pid         bigint,
  tid         bigint,
  timestamp   timestamp,
  dbname      varchar(128),
  retcode     integer,
  eventtype   varchar(24),
  objtype     varchar(30),
  objschema   varchar(20),
  objname     varchar(30),
  event1_type varchar(20),
  event1      timestamp,
  event2_type varchar(20),
  event2      varchar(40),
  event3_type varchar(20),
  event3      varchar(40),
  eventstate  varchar(20))
```

2. 声明一个游标，以便用于对 SYSPROC.PD_GET_DIAG_HIST 执行的查询。

```
declare c1 cursor for
  select pid, tid, timestamp, dbname, retcode, eventtype,
      substr(objtype, 1, 30) as objtype,
      substr(objname_qualifier, 1, 20) as objschema,
      substr(objname, 1, 30) as objname,
      substr(first_eventqualifiertype, 1, 20),
      substr(first_eventqualifier, 1, 26),
      substr(second_eventqualifiertype, 1, 20),
      substr(second_eventqualifier, 1, 40),
      substr(third_eventqualifiertype, 1, 20),
      substr(third_eventqualifier, 1, 40),
      substr(eventstate, 1, 20)
    from table (sysproc.pd_get_diag_hist
      ('optstats', 'EX', 'NONE',
        current_timestamp - 1 year, cast(null as timestamp ))) as sl
```

3. 将统计日志记录装入到表中。

```
load from c1 of cursor replace into db2user.stats_log
```

4. 对该表创建索引，然后收集有关该表的统计信息。

```
create index sl_ix1 on db2user.stats_log(eventtype, event1);
create index sl_ix2 on db2user.stats_log(objtype, event1);
create index sl_ix3 on db2user.stats_log(objname);

runstats on table db2user.stats_log
  with distribution and sampled detailed indexes all;
```

收集和更新统计信息的准则

RUNSTATS 实用程序用于收集有关表、索引和统计视图的统计信息，以便为优化器提供准确的信息进行访问方案选择。

在下列情况下，请使用 RUNSTATS 实用程序来收集统计信息:

- 将数据装入表并创建适当的索引之后
- 对表创建新索引之后
- 使用 REORG 实用程序重组表之后
- 通过更新、插入或删除操作对表及其索引进行重大修改之后
- 在绑定性能非常重要的应用程序之前
- 当您想要将当前统计信息与先前统计信息进行比较时
- 更改预取值之后
- 执行 REDISTRIBUTE DATABASE PARTITION GROUP 命令之后
- 存在 XML 列时。使用 RUNSTATS 只收集 XML 列的统计信息时，将保留执行装入操作或上次执行 RUNSTATS 操作期间收集的非 XML 列的现有统计信息。如果先前已收集有关某些 XML 列的统计信息，并且当前 RUNSTATS 操作未包括那些列，那么将替换或删除那些统计信息。

为了提高 RUNSTATS 性能并节省用来存储统计信息的磁盘空间，请考虑仅指定那些应该收集其数据分布统计信息的列。

执行 RUNSTATS 后，应该重新绑定应用程序。如果有新的统计信息可用，那么查询优化器可能会选择不同的访问方案。

如果无法一次收集全套统计信息，那么请对部分对象使用 RUNSTATS 实用程序。如果由于正在对那些对象执行活动而引起不一致性，那么查询优化期间将返回警告消息（SQL0437W，原因码 6）。如果发生这种情况，那么请再次使用 RUNSTATS 以更新分布统计信息。

要确保索引统计信息与相应的表同步，那么请同时收集表和索引统计信息。如果自从上次收集统计信息以来已对该表作了大量修改，那么只更新该表的索引统计信息将使这两组统计信息相互不同步。

在生产系统上使用 RUNSTATS 实用程序可能会对工作负载的性能产生负面影响。此实用程序现在支持调速选项，在数据库活动级别较高的时间段，可以使用此选项来限制执行 RUNSTATS 对性能的影响。

在分区数据库环境中收集某个表的统计信息时，RUNSTATS 将只对从中执行该实用程序的数据库分区执行操作。在此数据库分区中获得的结果将被推广到其他数据库分区。如果此数据库分区未包含该表的所需部分，那么该请求将被发送到数据库分区组中第一个包含所需数据的数据库分区。

收集统计视图的统计信息时，将对所有包含该视图所引用的基本表的数据库分区收集统计信息。

请考虑下列技巧，以提高 RUNSTATS 的效率以及统计信息的有效性:

- 仅对用来连接表的列或者 WHERE、GROUP BY 以及查询的类似子句中引用的列收集统计信息。如果已对这些列建立索引，那么可以使用 RUNSTATS 命令的 ONLY ON KEY COLUMNS 子句来指定这些列。
- 为特定的表和列定制数据库配置参数 **num_freqvalues** 和 **num_quantiles** 的值。
- 使用 SAMPLE DETAILED 子句来收集详细的索引统计信息，以减少为详细索引统计信息执行的后台计算量。SAMPLE DETAILED 子句将缩短收集统计信息所需的时间，并且在大多数情况下能够产生足够的精度。
- 为已填充的表创建索引时，请使用 COLLECT STATISTICS 子句在创建该索引时创建统计信息。
- 在添加或除去大量的表行之后，或者如果已更新要收集其统计信息的列中的数据，那么请再次使用 RUNSTATS 以更新统计信息。
- 由于 RUNSTATS 只对单一数据库分区收集统计信息，因此，如果数据并未一致地分布在所有数据库分区中，那么统计信息可能不太准确。如果您怀疑数据分布不均匀，那么在使用 RUNSTATS 实用程序之前，请考虑使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令在各个数据库分区之间重新分布数据。
- 对于 DB2 V9.7 修订包 1 和更高版本的发行版，可以收集 XML 列的分布统计信息。收集基于 XML 列中所指定 XML 数据的每个索引的分布统计信息。缺省情况下，对每个基于 XML 数据的索引的分布统计信息使用最大值 - 250 个分位数。

  收集 XML 列的分布统计信息时，可以更改最大分位数。您可以减小最大分位数，以根据特定数据大小来减少 XML 分布统计信息的空间需求；或者，如果使用 250 个分位数不足以捕获基于 XML 数据的索引的数据集的分布统计信息，那么可以增大最大分位数。

收集目录统计信息:

使用 RUNSTATS 实用程序来收集关于表、索引和统计视图的目录统计信息。查询优化器将使用此信息为查询选择最佳访问方案。

有关使用此实用程序所需的特权和权限的信息，请参阅 RUNSTATS 命令的描述。要收集目录统计信息，请执行下列操作:

1. 对于要收集其统计信息的表、索引或统计视图，连接至该表、索引或视图所在的数据库。
2. 从 DB2 命令行，执行 RUNSTATS 命令并指定适当的选项。这些选项使您能够为那些对表、索引或统计视图运行的查询定制所要收集的统计信息。
3. RUNSTATS 操作完成后，发出 COMMIT 语句以释放锁定。
4. 对于已更新其统计信息的表、索引或统计视图，重新绑定任何访问那些表、索引或统计视图的程序包。

注:

1. RUNSTATS 命令不支持使用昵称。如果查询访问联合数据库，请使用 RUNSTATS 命令来更新所有数据库中的表的统计信息，然后删除并重新创建访问远程表的昵称，以使新统计信息可供优化器使用。

2. 在分区数据库环境中收集某个表的统计信息时，RUNSTATS 将只对从其中执行该实用程序的数据库分区执行操作。在此数据库分区中获得的结果将被推广到其他数据库分区。如果此数据库分区未包含该表的所需部分，那么该请求将被发送到数据库分区组中第一个包含所需数据的数据库分区。

   收集统计视图的统计信息时，将对所有包含该视图所引用的基本表的数据库分区收集统计信息。
3. 对于 DB2 V9.7 修订包 1 和更高版本的发行版，下列各项适用于收集类型为 XML 的列的分布统计信息:
   - 收集基于 XML 列中所指定 XML 数据的每个索引的分布统计信息。
   - RUNSTATS 命令必须同时收集分布统计信息和表统计信息，才能收集基于 XML 数据的索引的分布统计信息。
   - 缺省情况下，RUNSTATS 命令最多可以对每个基于 XML 数据的索引的分布统计信息收集 250 个分位数。可以在执行 RUNSTATS 命令时指定某列的最大分位数。
   - 收集基于 XML 数据类型为 VARCHAR、DOUBLE、TIMESTAMP 和 DATE 的索引的分布统计信息。将不收集基于 XML 数据类型为 VARCHAR HASHED 的索引的 XML 分布统计信息。
   - 将不收集基于分区表中所定义 XML 数据的分区索引的分布统计信息。

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为查询选择最佳的访问方案，因此，使统计信息始终保持最新至关重要。随着数据库不断增大，高效地收集统计信息将变得更具挑战性。

一种有效的方法是，对表数据的随机样本收集统计信息。对于 I/O 受限制或处理器受限制的系统而言，性能将极大提升。

DB2 产品使您能够高效地为收集统计信息而进行数据采样，从而有可能大幅提升 RUNSTATS 实用程序的性能，同时保持高度的准确性。

可以通过两种方法进行采样: 行级采样和页级采样。要获取这些样本方法的描述，请参阅“查询中的数据采样”。

页级采样的性能很好，原因是对于选择的每一页，只需要执行一次 I/O 操作。对于行级采样而言，I/O 成本没有降低，原因是完整表扫描过程将扫描每个表页。但是，尽管 I/O 次数未减少，但行级采样的性能还是有很大提高，这是因为收集统计信息这一操作需要耗用大量处理器资源。

在数据值集群程度很高的情况下，行级采样将比页级采样提供更好的样本。与页级采样相比，行级别样本集有可能更好地反映数据，这是因为此样本集将包括每个数据页中 *P%* 的行。对于页级采样而言，样本集将包括 *P%* 的页的所有行。如果这些行随机分布在表中，那么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别不大。

除非使用了 REPEATABLE 选项（重新生成上一个样本），否则反复调用 RUNSTATS 命令将随机生成各个样本。当需要数据保持不变的表的一致统计信息时，此选项非常有用。

子元素统计信息:

如果在除模式末尾以外的任何位置使用 % 通配符来指定 LIKE 谓词，那么您应该收集有关子元素结构的基本信息。

除了通配 LIKE 谓词（例如 `SELECT...FROM DOCUMENTS WHERE KEYWORDS LIKE '%simulation%'`）以外，列和查询必须符合特定条件才能受益于子元素统计信息。

表列应该包含由空格分隔的子字段或子元素。例如，假定 DOCUMENTS 表包含四行数据，并且这个表的 KEYWORDS 列包含用于执行文本检索的相关关键字列表。KEYWORDS 中的值是:

```
'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，每个列值都由 5 个子元素构成，其中每个子元素都是一个单词（关键字）并通过空格与其他子元素分隔开。

查询应该在 WHERE 子句中引用这些列。

优化器总是估算与每个谓词匹配的行数。对于这些通配的 LIKE 谓词，优化器假定正在匹配的列包含一系列并置在一起的元素，并且，它根据字符串长度估算每个元素的长度（不包括前导和结尾 % 字符）。如果收集子元素统计信息，那么优化器将获得关于每个子元素的长度以及定界符的信息。它可以使用这些附加信息来更准确地估算将与该谓词匹配的行数。

要收集子元素统计信息，请执行带有 LIKE STATISTICS 选项的 RUNSTATS 命令。

关于子元素的 RUNSTATS 统计信息:

如果指定了 LIKE STATISTICS 子句，那么对于类型为 CHAR 和 VARCHAR 并且代码页属性为单字节字符集（SBCS）、FOR BIT DATA 或 UTF-8 的列，RUNSTATS 实用程序将收集统计信息。

SUB_COUNT
子元素的平均数目。

SUB_DELIM_LENGTH
每个用于分隔子元素的定界符的平均长度。在此上下文中，定界符是指一个或多个连续的空格字符。

DB2_LIKE_VARCHAR 注册表变量影响优化器处理以下格式的 谓词的方式: `<column> like '%<character-string>%'`。有关此注册表变量的更多信息，请参阅“查询编译器变量”。

要检查子元素统计信息的值，请查询 SYSCAT.COLUMNS 目录视图。例如:

```
select substr(colname, 1, 16), sub_count, sub_delim_length
  from syscat.columns where tabname = 'DOCUMENTS'
```

如果指定了 LIKE STATISTICS 子句，那么完成 RUNSTATS 实用程序所需的时间将延长。如果您正在考虑使用此选项，那么请评估耗费此附加开销对于查询性能方面的改进而言是否值得。

有关手动更新目录统计信息的一般规则:

在更新目录统计信息时，最重要的一般规则是确保将各种统计信息的有效值、范围和格式存储在那些统计信息的视图中。

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（列中的相异值数目不能大于表中的行数）。假定要将 COLCARD 从 100 减少到 25，并将 CARD 从 200 减少到 50。如果先更新 SYSSTAT.TABLES，那么将返回错误，这是因为 CARD 将小于 COLCARD。

但是，在某些情况下难以检测冲突，因此可能不会返回错误，受影响的统计信息存储在不同目录表中时尤其如此。

在更新目录统计信息之前，请至少确保:

- 数字统计信息为 -1 或者大于等于零。
- 表示百分比的数字统计信息（例如 SYSSTAT.INDEXES 中的 CLUSTERRATIO）介于 0 与 100 之间。

创建表时，目录统计信息将设置为 -1，以指示该表没有统计信息。DB2 服务器将使用缺省值来编译和优化 SQL 或 XQuery 语句，直到收集了统计信息为止。更新表或索引统计信息时，如果新值与缺省值不一致，那么更新操作可能会失败。因此，在创建表之后并在尝试更新表或索引的统计信息之前，建议您运行 RUNSTATS 实用程序。

注:

1. 对于行类型，子表的表级别统计信息 NPAGES、FPAGES 和 OVERFLOW 不可更新。
2. 分区级别的表和索引统计信息不可更新。

有关手动更新列统计信息的规则:

在更新 SYSSTAT.COLUMNS 目录视图中的统计信息时，您应该遵循特定的准则。

- 以手动方式更新 HIGH2KEY 或 LOW2KEY 值时，请确保:
  - 值对于相应用户列的数据类型而言有效。
  - 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，后者不包括附加的引号（引号可使字符串的长度增大到 68）。这意味着，在确定 HIGH2KEY 或 LOW2KEY 值时，将只考虑相应用户列中的值的前 33 个字符。
  - 存储值之后，就可以将其与 UPDATE 语句的 SET 子句配合使用以及将其用于成本计算。对于字符串而言，这意味着在字符串开头和末尾添加单引号，并且对字符串中已存在的每个引号另外添加一个引号。在表 73 中，提供了用户列值以及它们在 HIGH2KEY 或 LOW2KEY 中的相应值的示例。

表 73. HIGH2KEY 和 LOW2KEY 值（按数据类型排列）

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

  - 只要相应的列中有 3 个以上相异值，HIGH2KEY 就大于 LOW2KEY。

- 列的基数（SYSSTAT.COLUMNS 中的 COLCARD）不能大于其相应的表或统计视图的基数（SYSSTAT.TABLES 中的 CARD）。
- 列中的空值数（SYSSTAT.COLUMNS 中的 NUMNULLS）不能大于其相应的表或统计视图的基数（SYSSTAT.TABLES 中的 CARD）。
- 被定义为 LONG 或大对象（LOB）数据类型的列不支持统计信息。

有关手动更新表和昵称统计信息的规则:

在更新 SYSSTAT.TABLES 目录视图中的统计信息时，您应该遵循特定的准则。

- 在 SYSSTAT.TABLES 中，您可以更新的统计值只有 CARD、FPAGES、NPAGES 和 OVERFLOW；而对于多维集群（MDC）表，还能更新 ACTIVE_BLOCKS。
- CARD 必须大于或等于 SYSSTAT.COLUMNS 中相应的表的所有 COLCARD 值。
- CARD 必须大于 NPAGES。
- FPAGES 必须大于 NPAGES。
- 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）。
- 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）。

在联合数据库系统中，在手动方式更新基于远程视图的昵称的统计信息时，务必小心谨慎。统计信息（例如昵称将要返回的行数）可能未反映评估此远程视图的实际成本，因此可能会误导 DB2 优化器。但是，在某些情况下，更新统计信息可以使远程视图受益；这包括基于单一基本表来定义并且未对 SELECT 列表应用任何列函数的远程视图。复杂视图可能要求执行复杂的调整过程以便调整每个查询。请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断那些视图的成本。

详细的索引统计信息

如果对索引执行的 RUNSTATS 操作指定了 DETAILED 选项，那么收集到的统计信息将允许优化器根据缓冲池大小来估算需要执行的数据页访存次数。此附加信息帮助优化器更好地估算通过索引访问表的成本。

详细统计信息提供关于在不同缓冲区大小下执行完整索引扫描时，访问表的数据页所需执行的物理 I/O 次数的简明信息。当 RUNSTATS 实用程序扫描索引的页时，它将模拟不同的缓冲区大小并估算缺页故障发生频率。例如，如果只有一个缓冲区页可用，那么索引所引用的每个新页都将导致缺页故障。在最坏的情况下，每行都可能引用不同的页，从而导致 I/O 次数与已建立索引的表中的行数相同。另一种极端情况是，缓冲区大小足以容纳整个表（但不超过最大缓冲区大小），因此所有表页都被一次性读取。这种情况的结果是，物理 I/O 次数是缓冲区大小的无变化、非递增应变量。

此统计信息还有助于更精确地估算表行相对于索引顺序的集群度。集群度越低，通过索引访问表行时所需执行的 I/O 次数就越多。优化器在估算通过索引访问表的成本时，将同时考虑缓冲区大小和集群度。

在下列情况下，请收集详细的索引统计信息:

- 查询引用了未包括在索引中的列
- 表有多个具有不同集群度的非集群索引
- 各个键值的集群度不均匀

- 以不均匀的方式更新索引值

如果事先不了解这些情况或者未强制在不同缓冲区大小下扫描索引并接着监视所引起的物理 I/O，那么难以确定这些情况。确定是否发生其中任何一种情况的最低成本方法可能是，收集并检查索引的详细统计信息，如果得到的 PAGE_FETCH_PAIRS 非线性，那么保留这些统计信息。

收集详细的索引统计信息时，完成 RUNSTATS 操作所需耗用的时间较长，并且需要较多的内存和处理时间。例如，SAMPLED DETAILED 选项需要 2 MB 的统计信息堆。请将另外 488 个 4-KB 页分配给 **stat_heap_sz** 数据库配置参数设置，以便满足此内存需求。如果该堆太小，那么 RUNSTATS 实用程序在尝试收集统计信息前将返回错误。

除非表足够大（约大于 25 页），否则不会收集 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS。在这种情况下，CLUSTERFACTOR 将是介于 0 与 1 之间的值；而 CLUSTERRATIO 将是 -1（不收集）。如果表相对较小，那么 RUNSTATS 实用程序只收集 CLUSTERRATIO（值介于 0 与 100 之间），而不会收集 CLUSTERFACTOR 和 PAGE_FETCH_PAIRS。如果未指定 DETAILED 子句，那么只收集 CLUSTERRATIO。

收集索引统计信息:

收集索引统计信息可以帮助优化器确定是否应该使用特定索引来解析查询。

以下示例基于名为 SALES 的数据库，该数据库包含具有索引 CUSTIDX1 和 CUSTIDX2 的表 CUSTOMERS。

有关使用 RUNSTATS 实用程序所需的特权和权限的信息，请参阅 RUNSTATS 命令的描述。

要收集索引的详细统计信息，请执行下列操作:

1. 连接至 SALES 数据库。
2. 根据需要，从 DB2 命令行执行下列其中一个命令:
   - 要同时收集有关 CUSTIDX1 和 CUSTIDX2 的详细统计信息:

     ```
     runstats on table sales.customers and detailed indexes all
     ```

   - 要收集这两个索引的详细统计信息，但进行采样而不是对每个索引条目执行详细计算:

     ```
     runstats on table sales.customers and sampled detailed indexes all
     ```

     SAMPLED DETAILED 选项需要 2 MB 的统计信息堆。请将另外 488 个 4-KB 页分配给 **stat_heap_sz** 数据库配置参数设置，以便满足此内存需求。如果该堆太小，那么 RUNSTATS 实用程序在尝试收集统计信息前将返回错误。

   - 要收集所采样的索引的详细统计信息以及表的分布统计信息，以使索引统计信息与表统计信息一致:

     ```
     runstats on table sales.customers
       with distribution on key columns
         and sampled detailed indexes all
     ```

有关手动更新索引统计信息的规则:

在更新 SYSSTAT.INDEXES 目录视图中的统计信息时，您应该遵循特定的准则。

- 下列规则适用于 PAGE_FETCH_PAIRS:
  - PAGE_FETCH_PAIRS 统计信息中的各个值不能超过 10 位，并且必须小于最大整数值（2 147 483 647）。
  - PAGE_FETCH_PAIRS 统计信息中的各个值必须由空格字符定界符分隔。
  - 如果 CLUSTERFACTOR 大于零，那么必须始终存在有效的 PAGE_FETCH_PAIRS 统计信息。
  - 在单个 PAGE_FETCH_PAIRS 统计信息中，必须正好有 11 对值。
  - PAGE_FETCH_PAIRS 统计信息中的缓冲区大小值（每一对值中的第一个值）必须按升序出现。
  - PAGE_FETCH_PAIRS 统计信息中的任何缓冲区大小值都不能大于 MIN(NPAGES, 524 287)（对于 32 位操作系统）或 MIN(NPAGES, 2 147 483 647)（对于 64 位操作系统），其中 NPAGES（存储在 SYSSTAT.TABLES 中）是相应表中的页数。
  - PAGE_FETCH_PAIRS 统计信息中的页访存值（每一对值中的第二个值）必须按降序出现，并且没有任何一个值小于相应表的 NPAGES 或大于 CARD。
  - 如果两个连续值对中的缓冲区大小值相同，那么这两对值中的页访存值也必须相同。

  下面是有效 PAGE_FETCH_PAIRS 统计信息的一个示例:

  ```
  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
  ```

- 下列规则适用于 CLUSTERRATIO 和 CLUSTERFACTOR:
  - CLUSTERRATIO 的有效值是 -1 或者介于 0 与 100 之间。
  - CLUSTERFACTOR 的有效值是 -1 或者介于 0 与 1 之间。
  - 在 CLUSTERRATIO 和 CLUSTERFACTOR 值中，至少一个必须始终是 -1。
  - 如果 CLUSTERFACTOR 是正数值，那么它必须伴有一个有效的 PAGE_FETCH_PAIRS 值。
- 对于关系索引，下列规则适用于 FIRSTKEYCARD、FIRST2KEYCARD、FIRST3KEYCARD、FIRST4KEYCARD、FULLKEYCARD 和 INDCARD:
  - 对于单列索引，FIRSTKEYCARD 必须等于 FULLKEYCARD。
  - FIRSTKEYCARD 必须等于相应列的 SYSSTAT.COLUMNS.COLCARD。
  - 如果其中任何索引统计信息不相关，那么请将其设置为 -1。例如，如果索引只包含 3 列，那么请将 FIRST4KEYCARD 设置为 -1。
  - 对于多列索引，如果所有统计信息都相关，那么它们之间的关系必须是:

    ```
    FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
      <= FULLKEYCARD <= INDCARD == CARD
    ```

- 对于基于 XML 数据的索引，FIRSTKEYCARD、FIRST2KEYCARD、FIRST3KEYCARD、FIRST4KEYCARD、FULLKEYCARD 和 INDCARD 之间的关系必须是:

```
FIRSTKEYCARD <= FIRST2KEYCARD <= FIRST3KEYCARD <= FIRST4KEYCARD
  <= FULLKEYCARD <= INDCARD
```

- 下列规则适用于 SEQUENTIAL_PAGES 和 DENSITY:
  - SEQUENTIAL_PAGES 的有效值是 -1 或者介于 0 与 NLEAF 之间。
  - DENSITY 的有效值是 -1 或者介于 0 与 100 之间。

分布统计信息

您可以收集两种数据分布统计信息: 高频值统计信息和分位数统计信息。

- *高频值统计信息*提供关于某一列以及具有最高重复次数、次高重复次数直至 **num_freqvalues** 数据库配置参数值所指定级别的数据值的信息。要禁止收集高频值统计信息，请将 **num_freqvalues** 设置为 0。还可以对特定的表、统计视图或列使用 RUNSTATS 命令的 NUM_FREQVALUES 子句。
- *分位数统计信息*提供关于数据值相对于其他值的分布情况的信息。这些统计信息被称为 *K* 分位数，它们表示值 *V*，至少有 *K* 个值等于或小于该值。您可以通过按升序对值进行排序来计算 *K* 分位数。*K* 分位数值是从范围低端起第 *K* 个位置中的值。

  要指定应该将列数据值分组成的“部分”数（分位数），请将 **num_quantiles** 数据库配置参数设置为 2 与 32767 之间的某个值。缺省值为 20，它确保对于任何等式、小于或大于谓词，优化器估算误差最大为正或负 2.5%，而对于任何 BETWEEN 谓词，最大误差为正或负 5%。要禁止收集分位数统计信息，请将 **num_quantiles** 设置为 0 或 1。

  可以对特定的表、统计视图或列设置 **num_quantiles**。

注: 使用的 **num_freqvalues** 和 **num_quantiles** 值越大，RUNSTATS 实用程序耗用的处理资源和内存（由 **stat_heap_sz** 数据库配置参数指定）也越多。

何时收集分布统计信息

要确定表或统计视图的分布统计信息是否有用，首先确定:

- 应用程序中的查询是否使用主变量。

  分布统计信息对于不使用主变量的动态和静态查询而言最为有用。在访问包含主变量的查询时，优化器只能有限地利用分布统计信息。

- 列中的数据是否均匀分布。

  如果表中至少有一列的数据分布非常“不均匀”，并且该列频繁出现在等式或范围谓词中（即，出现在类似于以下的子句中），那么请创建分布统计信息:

```
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;
```

可能会出现两类数据分布不均匀情况，并且很可能同时出现。

- 数据可能高度集群，而不是在最大与最小数据值之间均匀分布。请考虑下面这一列，其中的数据在范围（5,10）内集群:

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

分位数统计信息可以帮助优化器处理这种数据分布。

查询可以帮助您确定列数据的分布是否不均匀。例如:

```
select c1, count(*) as occurrences
  from t1
group by c1
order by occurrences desc
```

- 重复的数据值经常出现。请考虑数据按下列频率分布的列:

表 74. 某一列中数据值的频率

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

高频值和分位数统计信息都可以帮助优化器处理众多的重复值。

何时仅收集索引统计信息

在下列情况下，您可以考虑只收集基于索引数据的统计信息:

- 在运行 RUNSTATS 实用程序之后已创建新索引，并且您不想再次收集有关表数据的统计信息。
- 存在许多影响索引第一列的数据更改。

要指定哪个级别的统计精度

使用 **num_quantiles** 和 **num_freqvalues** 数据库配置参数来指定以何精度来存储分布统计信息。也可以在收集表或列的统计信息时使用相应的 RUNSTATS 命令选项来指定精度。这些值设置得越大，RUNSTATS 实用程序在创建和更新分布统计信息时使用的精度就越大。但是，精度越大，在 RUNSTATS 操作本身执行期间以及在目录表中存储更多数据时需要的资源就越多。

对于大部分数据库而言，请指定介于 10 与 100 之间的值作为 **num_freqvalues** 数据库配置参数的值。理想情况下，创建高频值统计信息时，应该使其余值的频率大致相等或者与最高频值的频率相比可以忽略不计。数据库管理器收集的数目可能低于此数目，其原因在于，将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，那么请将 **num_freqvalues** 的值设置为 0。

要指定分位数的数目，请将 **num_quantiles** 数据库配置参数设置为介于 20 与 50 之间的值。

- 首先确定估算任何范围查询的行数时可接受的最大误差，即百分比 *P*。
- 对于 BETWEEN 谓词，分位数的数目应该近似于 100/*P*，对于任何其他类型的范围谓词（<、<=、> 或 >=），应该近似于 50/*P*。

例如，对于 BETWEEN 谓词而言，25 个分位数导致的最大估算误差为 4%，而对于“>”谓词则为 2%。通常，请至少指定 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，那么请将 **num_quantiles** 设置为 0。如果将此参数设置为“1”，那么因为整个值范围适合于一个分位数，因此不收集分位数统计信息。

优化器如何使用分布统计信息:

优化器将使用分布统计信息，以便更好地估算不同查询访问方案的成本。

除非优化器有关于值在低值与高值之间的分布情况的附加信息，否则将假定数据值均匀分布。如果数据值彼此差异较大、在范围的某些部分集群或者包含许多重复值，那么优化器将选择次于最优的访问方案。

请考虑以下示例：为了选择成本较低的访问方案，优化器需要估算某个列值满足等式或范围谓词的行数。估算越准确，优化器选择最优访问方案的可能性就越大。对于以下查询:

```
select c1, c2
  from table1
  where c1 = 'NEW YORK'
  and c2 <= 10
```

假定对列 C1 和 C2 都定义了索引。一种可能的访问方案是，使用基于 C1 的索引来检索所有符合条件 C1 = 'NEW YORK' 的行，然后对检索到的每一行检查是否符合条件 C2 <= 10。另一种方案是使用基于 C2 的索引来检索所有符合条件 C2 <= 10 的行，然后对检索到的每一行检查是否符合条件 C1 = 'NEW YORK'。因为执行查询的主要成本通常是检索行的成本，所以检索次数最少的方案最好。选择此方案意味着需要估算满足每个谓词的行数。

如果没有分布统计信息，但已对表或统计视图运行 RUNSTATS 实用程序，那么可供优化器使用的信息只有列中的次高数据值（HIGH2KEY）、次低数据值（LOW2KEY）、相异值数目（COLCARD）和行数（CARD）。假定列中的数据值具有相等的频率，并且各个数据值在 LOW2KEY 与 HIGH2KEY 之间均匀分布，以此为前提估算满足等式或范围谓词的行数。具体而言，满足等式谓词（C1 = KEY）的行数估算为 CARD/COLCARD，而满足范围谓词（C1 BETWEEN KEY1 AND KEY2）的行数估算为:

```
    KEY2 - KEY1
------------------  x CARD
HIGH2KEY - LOW2KEY
```

仅当数据值在列中的真实分布相当均匀时，这些估算才准确。如果没有分布统计信息，并且数据值的频率变动很大，或者数据值的分布非常不均匀，那么估算可能偏差很大，并且优化器选择的访问方案可能并非最优。

存在可用的分布统计信息时，可通过以下方法大大减少此类误差的可能性：使用高频值统计信息来估算满足等式谓词的行数，使用高频值统计信息和分位数统计信息来估算满足范围谓词的行数。

收集特定列的分布统计信息:

为了提高 RUNSTATS 操作和后续查询方案分析的效率，可以只对查询在 WHERE、GROUP BY 和类似子句中引用的那些列收集分布统计信息。此外，还可以收集有关所组合的各组列的基数统计信息。优化器在为引用了某个组中的列的查询估算选择性时，它将使用此类信息来检测列关联。

以下示例基于名为 SALES 的数据库，该数据库包含具有索引 CUSTIDX1 和 CUSTIDX2 的表 CUSTOMERS。

有关使用 RUNSTATS 实用程序所需的特权和权限的信息，请参阅 RUNSTATS 命令的描述。

在分区数据库环境中收集某个表的统计信息时，RUNSTATS 将只对从中执行该实用程序的数据库分区执行操作。在此数据库分区中获得的结果将被推广到其他数据库分区。如果此数据库分区未包含该表的所需部分，那么该请求将被发送到数据库分区组中第一个包含所需数据的数据库分区。

要收集特定列的统计信息，请执行下列操作:

1. 连接至 SALES 数据库。
2. 根据需要，从 DB2 命令行执行下列其中一个命令:
   - 要同时收集有关 ZIP 和 YTDTOTAL 列的分布统计信息:

     ```
     runstats on table sales.customers
       with distribution on columns (zip, ytdtotal)
     ```

   - 要收集这两列的分布统计信息，但使用不同的分布选项进行收集:

     ```
     runstats on table sales.customers
       with distribution on columns (
         zip, ytdtotal num_freqvalues 50 num_quantiles 75)
     ```

   - 要收集已在 CUSTIDX1 和 CUSTIDX2 中建立索引的列的分布统计信息:

     ```
     runstats on table sales.customer
       on key columns
     ```

   - 要收集列 ZIP 和 YTDTOTAL 以及包含 REGION 和 TERRITORY 的列组的统计信息:

     ```
     runstats on table sales.customers
       on columns (zip, (region, territory), ytdtotal)
     ```

   - 假定先前已使用带有 STATISTICS 选项的 LOAD 命令收集非 XML 列的统计信息。要收集 XML 列 MISCINFO 的统计信息:

     ```
     runstats on table sales.customers
       on columns (miscinfo)
     ```

   - 要只收集非 XML 列的统计信息:

     ```
     runstats on table sales.customers
       excluding xml columns
     ```

     EXCLUDING XML COLUMNS 子句优先于所有其他指定了 XML 列的子句。

- 对于 DB2 V9.7 修订包 1 和更高版本的发行版，以下命令将通过对 XML 列 MISCINFO 使用最多 50 个分位数来收集分布统计信息。将对表中所有其他列使用缺省值，即，20 个分位数:

```
runstats on table sales.customers
   with distribution on columns ( miscinfo num_quantiles 50 )
   default num_quantiles 20
```

  注: 要收集 XML 列 MISCINFO 的分布统计信息，需要满足下列条件:
  - 必须同时收集表统计信息和分布统计信息。
  - 必须对此列定义基于 XML 数据的索引，并且为索引指定的数据类型必须为 VARCHAR、DOUBLE、TIMESTAMP 或 DATE。

分布统计信息用法的扩展示例:

分布统计信息提供有关表数据的频率和分布情况的信息，此信息可以帮助优化器在数据未均匀分布并且存在许多重复值时构建查询访问方案。

下列示例将帮助您理解优化器使用分布统计信息的方式。

使用高频值统计信息的示例

请考虑包含格式为 C1 = KEY 的等式谓词的查询。如果有可用的高频值统计信息，那么优化器可以使用那些统计信息来选择适当的访问方案，如下所示:

- 如果 KEY 是频率最高的 *N* 个值中的一个，那么优化器将使用目录中存储的 KEY 的频率。
- 如果 KEY 不是频率最高的 *N* 个值中的一个，那么优化器在假定 (COLCARD - *N*) 个非高频值均匀分布的前提下估算满足谓词的行数。即，按以下公式（1）来估算行数:

```
CARD - NUM_FREQ_ROWS
--------------------
    COLCARD - N
```

其中，CARD 是表中的行数，COLCARD 是列的基数，NUM_FREQ_ROWS 是值等于 *N* 个最高频值中某个值的总行数。

例如，考虑数据值具有下列频率的列 C1:

数据值	频率
1	2
2	3
3	40
4	4
5	1

表中的行数为 50，列基数为 5。正好有 40 行满足谓词 C1 = 3。如果优化器假定数据均匀分布，那么它估计满足谓词的行数为 50/5 = 10，并且误差为 -75%。但是，如果有仅基于最高频值（即，*N* = 1）的高频值统计信息，那么估算的行数为 40，并且没有误差。

考虑另一个示例，即，共有两行满足谓词 C1 = 1。在没有高频值统计信息的情况下，估算的满足谓词的行数为 10，误差为 400%:

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

   10 - 2
   ------  X 100 = 400%
      2
```

在使用高频值统计信息（*N* = 1）的情况下，优化器将使用上面给出的公式（1）来估算包含此值的行数，如下所示:

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

并且误差降低了一个数量级:

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

使用分位数统计信息的示例

下列有关分位数统计信息的讨论使用了术语“K 分位数”。列的 *K 分位数*是最小的数据值 *V*，即至少有 *K* 行的数据值小于或等于 *V*。要计算 K 分位数，请按升序对列值进行排序；*K* 分位数是已排序列中的第 *K* 行中的数据值。

如果有分位数统计信息，那么优化器可以更好地估算满足范围谓词的行数，如下列示例所示。请考虑包含下列值的列 C1:

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

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

- 正好有 7 行满足谓词 C <= 8.5。假定数据均匀分布，那么以下公式（2）:

  ```
     KEY2 - KEY1
  ------------------  X CARD
  HIGH2KEY - LOW2KEY
  ```

  在 LOW2KEY 替代 KEY1 的情况下，满足此谓词的行数将估算为:

```
 8.5 - 5.1
----------  X 10 ≈ 0
93.6 - 5.1
```

其中 ≈ 表示“几乎相等”。在此估算中，误差接近 -100%。

如果有分位数统计信息，那么优化器将通过与 8.5（其中一个分位数中的最大值）相对应的 *K* 值 7 来估算满足此谓词的行数。在这种情况下，误差缩小为 0。

- 正好有 8 行满足谓词 C <= 10。如果优化器假定数据均匀分布并使用公式（2），那么满足此谓词的行数估算为 1，且误差为 -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将满足此谓词的行数估算为 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。与上一个示例相同，r_1 = 7。为了估算 r_2，优化器使用线性插值法:

```
       10 - 8.5
r_2 ≈ ----------  X（值 > 8.5 且 <= 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
```

  最终估算是 r_1 + r_2 ≈ 7，误差仅为 -12.5%。

在这些示例中，分位数能够提高估算的准确性，这是因为实际数据值“集群”在范围 5 - 10 之间，但标准估算公式假定数据值均匀地分布在 0 与 100 之间。

当不同数据值的频率有很大差异时，使用分位数也能够提高准确性。假定某一列中的数据值具有下列频率:

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

K	*K* 分位数
95	70
100	80

并且，假定存在基于三个最高频值的高频值统计信息。

正好有 10 行满足谓词 C BETWEEN 20 AND 30。假定数据均匀分布，在使用公式（2）的情况下，满足此谓词的行数估算如下:

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

误差为 150%。

在同时使用高频值统计信息和分位数统计信息的情况下，满足此谓词的行数估算为 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。在使用公式（1）时，r_1 估算如下:

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

使用线性插值法时，r_2 估算如下:

```
  30 - 20
  -------  X（值 > 20 且 <= 40 的行数）
  40 - 20

  30 - 20
= -------  X (25 - 5)
  40 - 20

= 10
```

得到的最终估算值为 15，并使误差降低到原来的三分之一。

有关手动更新分布统计信息的规则:

在更新 SYSSTAT.COLDIST 目录视图中的统计信息时，您应该遵循特定的准则。

- 高频值统计信息:
  - VALCOUNT 值必须保持不变或者随 SEQNO 值的递增而减小。
  - COLVALUE 值的数目必须小于或等于该列中相异值的数目，后者存储在 SYSSTAT.COLUMNS.COLCARD 中。
  - VALCOUNT 中的各个值之和必须小于或等于该列中的行数，后者存储在 SYSSTAT.TABLES.CARD 中。
  - 在大多数情况下，COLVALUE 值应该介于该列的次高数据值与次低数据值之间，这两个值分别存储在 SYSSTAT.COLUMNS 中的 HIGH2KEY 和 LOW2KEY 中。可以有一个大于 HIGH2KEY 的高频值以及一个小于 LOW2KEY 的高频值。
- 分位数统计信息:
  - COLVALUE 值必须保持不变或者随 SEQNO 值的递增而减小。
  - VALCOUNT 值必须随 SEQNO 值的递增而递增。
  - 最大的 COLVALUE 值在 VALCOUNT 中必须要有等于该列中行数的相应条目。

– 在大多数情况下，COLVALUE 值应该介于该列的次高数据值与次低数据值之间，这两个值分别存储在 SYSSTAT.COLUMNS 中的 HIGH2KEY 和 LOW2KEY 中。

假定包含 *R* 行的列 C1 的分布统计信息可供使用，并假定您希望修改该统计信息，以使其对应于具有相同的数据值相对比例但包含 (*F* x *R*) 行的列。要将高频值或分位数统计信息上调 *F* 倍，请将每个 VALCOUNT 条目乘以 *F*。

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，请编辑 SYSSTAT.ROUTINES 目录视图。

RUNSTATS 实用程序不会收集 UDF 的统计信息。如果有 UDF 统计信息，那么优化器在估算各种访问方案的成本时就可以使用这些统计信息。如果没有这些统计信息，那么优化器将使用缺省值，这些缺省值假定一个简单 UDF。

表 75 列示了目录视图列，您可以为这些列提供估算值以提高性能。注意，用户只能修改 SYSSTAT.ROUTINES（而不是 SYSCAT.ROUTINES）中的列值。

表 75. 函数统计信息（SYSCAT.ROUTINES 和 SYSSTAT.ROUTINES）

统计信息	描述
IOS_PER_INVOC	每次调用函数时执行的读或写请求的估计数目
INSTS_PER_INVOC	每次调用函数时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执行的读或写请求的估计数目
INSTS_PER_ARGBYTE	对每个输入自变量字节执行的机器指令的估计数目
PERCENT_ARGBYTES	函数实际处理的输入自变量字节所占的估计平均百分比
INITIAL_IOS	第一次或最后一次调用函数时执行的读或写请求的估计数目
INITIAL_INSTS	第一次或最后一次调用函数时执行的机器指令的估计数目
CARDINALITY	表函数生成的估计行数

例如，考虑 EU_SHOE，这个 UDF 将美国鞋码转换为等同的欧洲鞋码。对于此 UDF，可以按如下方式设置 SYSSTAT.ROUTINES 中统计信息列的值:

- 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。设置为将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。全都设置为 0，这是因为此 UDF 只执行计算

PERCENT_ARGBYTES 将由并非始终处理整个输入字符串的函数使用。例如，考虑 LOCATE，这个 UDF 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

中第一次出现时的起始位置。假定第一个自变量的长度小得相对于第二个自变量而言并不重要，那么平均来说将搜索第二个自变量的 75%。根据此信息以及下列假定，应将 PERCENT_ARGBYTES 设置为 75:

- 有一半的机会找不到第一个自变量，这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位置出现的可能性相同，这将导致找到第一个自变量时搜索第二个自变量的一半（平均值）

您可以使用 INITIAL_INSTS 或 INITIAL_IOS 来记录第一次或最后一次调用函数时，所执行的机器指令或者读/写请求的估计数目；这可能代表成本，例如，设置暂存区的成本。

要获得关于某个 UDF 使用的 I/O 和指令的信息，请使用编程语言编译器或可用于操作系统的监视工具所提供的输出。

用于执行建模和假设情况计划的目录统计信息

在更改系统目录中某些用于计划用途的统计信息之后，您可以观察对数据库性能的影响。

更新所选系统目录统计信息的能力使您能够:

- 在开发系统上，使用生产系统统计信息对查询性能进行建模
- 执行“假设情况”查询性能分析

请不要手动更新生产系统上的统计信息。否则，对于包含动态 SQL 或 XQuery 语句的产品查询，优化器可能不会选择最佳的访问方案。

要修改表和索引及其组件的统计信息，您必须对该数据库具有显式的 DBADM 权限。具有 DATAACCESS 权限的用户可以对 SYSSTAT 模式中定义的视图执行 UPDATE 语句，以便更改这些统计信息列中的值。

不具有 DATAACCESS 权限的用户只能查看某些行，这些行包含那些用户对其具有 CONTROL 特权的对象的统计信息。如果您不具有 DATAACCESS 权限，但对各个数据库对象具有下列特权，那么也可以更改该对象的统计信息:

- 对表的显式 CONTROL 特权。您还可以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。您还可以更新这些昵称的列和索引的统计信息。注意，这些更新只影响本地元数据（不更改数据源表统计信息），并且只影响由 DB2 优化器生成的全局访问策略。
- 用户定义的函数（UDF）的所有权

以下代码示例更新 EMPLOYEE 表的统计信息:

```
update sysstat.tables
  set
    card = 10000,
    npages = 1000,
    fpages = 1000,
    overflow = 2
  where tabschema = 'MELNYK'
    and tabname = 'EMPLOYEE'
```

以手动方式更新目录统计信息时，务必小心谨慎。任意进行更改可能会严重影响后续查询的性能。您可以使用下列任何方法使开发系统上的统计信息恢复到一致状态:

- 回滚您在其中进行手动更改的工作单元（假定尚未落实该工作单元）。
- 使用 RUNSTATS 实用程序来刷新目录统计信息。
- 更新目录统计信息，以指定尚未收集统计信息；例如，将 NPAGES 列值设置为 -1 表明尚未收集此统计信息。
- 撤销您所作的更改。仅当您在进行任何更改之前已使用 db2look 命令来捕获统计信息时，才能使用此方法。

如果优化器确定某个值或值组合无效，那么它将使用缺省值并返回警告。但是，这种情况很罕见，原因是更新统计信息时已执行大部分验证工作。

用于对生产数据库建模的统计信息:

有时，您可能希望开发系统包含生产系统中的部分数据。但是，在开发系统上选择的访问方案不一定与生产系统上选择的访问方案相同。

在这些情况下，有必要更新开发系统的目录统计信息和配置，以使它们与生产系统上的目录统计信息和配置相匹配。

要生成使开发数据库的目录统计信息与生产数据库的目录统计信息相匹配所必需的数据操作语言（DML）语句，可以使用处于模仿方式（指定 -m 选项）的 db2look 命令。

对开发系统运行 db2look 所生成的 UPDATE 语句之后，可使用该系统来验证要在生产系统上生成的访问方案。由于优化器使用表空间的配置来估算 I/O 成本，因此开发系统上表空间的类型（SMS 或 DMS）和容器数必须与生产系统上表空间的类型和容器数相同。

避免手动更新目录统计信息

通过对 SYSSTAT 模式中的视图发出 UPDATE 语句，DB2 数据服务器支持手动更新目录统计信息。

在测试系统上模仿生产数据库以便检查查询访问方案时，此功能很有用。如果要捕获对 SYSSTAT 模式中的视图执行的 DDL 和 UPDATE 语句以便在另一个系统上回放，那么 db2look 实用程序非常有用。

请避免通过手动提供不正确的统计信息强制使用特定查询访问方案来影响查询优化器。虽然这种做法能够提高某些查询的性能，但也会导致其他查询的性能下降。在采用此方法之前，请考虑使用其他调整选项（例如使用优化准则和概要文件）。如果有必要采用此方法，那么务必记录原始统计信息以防需要将其复原。

最大程度地减轻 RUNSTATS 的影响

可以通过多种方法来提高 RUNSTATS 的性能。

要最大程度地减轻此实用程序对性能的影响，请完成下列步骤:

- 通过使用 COLUMNS 子句，限制应该收集统计信息的列。许多列从不会被查询工作负载中的谓词引用，因此不需要这些列的统计信息。
- 限制当数据通常均匀分布时要收集分布统计信息的列。与收集基本列统计信息相比，收集分布统计信息需要耗用更多的 CPU 和内存。但是，要确定某个列的值是否均匀分布，必须要有现成的统计信息，否则必须查询数据。此方法还假定数据在表被修改后将保持均匀分布。

- 通过进行页级或行级采样（指定 TABLESAMPLE SYSTEM 或 BERNOULLI 子句）来限制所处理的页数和行数。首先，通过指定 TABLESAMPLE SYSTEM(10) 进行 10% 页级采样。检查统计信息的准确性以及系统性能是否已由于访问方案的更改而下降。如果性能已下降，那么请尝试通过指定 TABLESAMPLE BERNOULLI(10) 进行 10% 行级采样。如果统计信息的准确性欠佳，那么请增加采样量。进行 RUNSTATS 页级或行级采样时，请对所连接的表使用相同的采样率。这对于确保连接列统计信息具有相同的准确性级别而言至关重要。
- 通过在 CREATE INDEX 语句中指定 COLLECT STATISTICS 选项，在索引创建期间收集索引统计信息。与创建索引后执行独立的 RUNSTATS 操作相比，此方法的速度要快。此方法还确保创建新索引之后立即生成它的统计信息，以使优化器能够准确估算使用该索引的成本。
- 在执行 LOAD 命令时，通过指定 REPLACE 选项来收集统计信息。与完成装入操作后执行独立的 RUNSTATS 操作相比，此方法的速度要快。此方法还确保表在装入数据后立即就有最新的统计信息，以使优化器能够准确估算使用该表的成本。

在分区数据库环境中，RUNSTATS 实用程序将从单一数据库分区中收集统计信息。如果对该表所在的数据库分区发出 RUNSTATS 命令，那么将从该位置收集统计信息。否则，将对该表的数据库分区组中的第一个数据库分区收集统计信息。为了获得一致的统计信息，请确保从同一个数据库分区中收集所连接的表的统计信息。

数据压缩与性能

数据压缩功能可用于减少必须对磁盘读写的数据量，从而降低 I/O 成本。

目前，可以执行两种形式的数据压缩:

- *值压缩*涉及除去一个值的重复条目，只存储一个副本，并跟踪任何对所存储值进行的引用的位置。
- *行压缩*涉及将一行中跨多个列值的重复模式替换为较短的符号字符串。行压缩逻辑将扫描要针对重复数据进行压缩的表。压缩字典包含该数据的短数字键，在压缩的行中，这些键替换实际数据。

在 DB2 版本 9.1 之前，您必须通过执行脱机表重组以手动方式创建压缩字典。在版本 9.5 和更高版本中，将为支持数据压缩功能的表自动创建数据压缩字典。

在本发行版中，版本 9.5 为永久表引入的自动行压缩功能已扩展为包括所有临时表。通过对临时表进行数据压缩:

- 减少了大型查询和复杂查询所需的临时磁盘空间量
- 提高了查询性能

如果已安装 DB2 存储器优化功能部件，那么将自动启用临时表数据压缩功能。

每个适合于进行行压缩的临时表都另外需要 2-3 MB 内存来创建它的压缩字典；此内存将保持分配到压缩字典创建完毕为止。

此外，还可以对基于已压缩的临时表的索引对象和索引进行压缩，以降低存储成本。对于通常包含许多非常大的索引的大型联机事务处理（OLTP）和数据仓库环境而言，此功能特别有用。在这两种情况下，索引压缩功能都能够在 I/O 受约束的环境中大幅提高性能，并且在 CPU 受约束的环境中只会引起性能小幅下降甚至完全不下降。

如果已对包含 XML 列的表启用压缩功能，那么还将对 XDA 对象中存储的 XML 数据进行压缩。在 XDA 对象中，将存储一个独立的 XML 数据压缩字典。这也适用于包含使用当前 DB2 产品版本添加的 XML 列的表。如果表包含使用先前版本创建的 XML 列，那么不支持 XDA 压缩功能；对于这样的表，将仅压缩数据对象。

降低日志记录开销以提高 DML 性能

数据库管理器将维护用于记录所有数据库更改的日志文件。共有两种日志记录策略：循环日志记录和归档日志记录。

- 使用*循环日志记录*策略时，在可用的文件装满后，将从初始日志文件开始复用日志文件。被覆盖的日志记录不可恢复。
- 使用*归档日志记录*策略时，将在日志文件装满日志记录后对其进行归档。通过保留日志，就可以进行日志前滚恢复，即，在灾难恢复期间重新应用日志文件中记录的数据库更改（已完成的工作单元或事务）。

对常规数据和索引页所作的所有更改在被记录器进程写入磁盘之前都会被写入日志缓冲区。SQL 语句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
- 直到相应数据页写入磁盘为止，这是因为 DB2 服务器使用预写记录功能，即，使用 COMMIT 语句来落实事务时，并不需要将所有已更改的数据页和索引页都写入磁盘
- 直到更改元数据完毕为止（此更改主要是由于执行数据定义语言语句所致）
- 日志缓冲区装满时

数据库管理器以这种方式将日志数据写入磁盘，以便最大程度地缩短处理延迟。如果有许多短小的事务同时进行处理，那么大部分延迟由必须等待日志数据写入磁盘的 COMMIT 语句造成。因此，记录器进程频繁地将少量日志数据写入磁盘。其他延迟由日志 I/O 引起。要在应用程序响应时间与日志记录延迟之间进行平衡，请将 **mincommit** 数据库配置参数设置为大于 1 的值。虽然此设置可能会导致某些应用程序的 COMMIT 延迟时间更长，但是一次操作可以写的日志数据将更多。

通过影子页面调度来跟踪对大对象（LOB）和 LONG VARCHAR 所作的更改。除非您指定保留日志，并且在 CREATE TABLE 语句中未指定 NOT LOGGED 子句的情况下定义 LOB 列，否则不会记录 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。直接插入 LOB 值完全参与更新、插入或删除日志记录，就像它们是 VARCHAR 值一样。

直接插入 LOB 可以提高性能

某些应用程序广泛使用大对象（LOB）。在许多情况下，这些 LOB 并不是非常大 – 大小最多几千个字节。现在，通过将此类 LOB 数据放入数据页中的格式化行（而不是放入 LOB 存储器对象），可以提高 LOB 数据访问性能。

这样的 LOB 被称为*直接插入 LOB*。以前，处理此类 LOB 可能会导致应用程序中出现瓶颈。直接插入 LOB 可以提高访问 LOB 数据的查询的性能，这是因为，不需要执行附加的 I/O 即可访存、插入或更新此数据。并且，还可以对直接插入 LOB 数据进行行压缩。

要启用此功能，请使用 CREATE TABLE 语句或 ALTER TABLE 语句的 INLINE LENGTH 选项。INLINE LENGTH 选项将应用于结构化类型、XML 类型或 LOB 列。对于 LOB 列，直接插入长度指示可以存储在基本表行中的 LOB 值的最大字节大小（包括 4 个开销字节）。

另外，还将隐式地对新表或现有表（在添加 LOB 列时）中的所有 LOB 列启用此功能，并且将在数据库升级期间对所有现有的 LOB 列启用此功能。每个 LOB 列都将根据对其定义的最大大小保留行空间。每个 LOB 列的隐式 INLINE LENGTH 值都是自动定义的，并且将存储下来，就像您已显式地指定此值一样。

无法以直接插入方式存储的 LOB 值将单独存储在 LOB 存储器对象中。

注意，当表带有包含直接插入 LOB 的列时，每一页所能容纳的行数将减少，只返回非 LOB 数据的查询的性能可能会有所下降。对于大多数语句包含一个或多个 LOB 列的工作负载，LOB 直接插入功能十分有益。

虽然不必对 LOB 数据进行日志记录，但系统始终对直接插入 LOB 进行日志记录，因此可能会增大日志记录开销。

第 4 章 制定性能调整策略

设计顾问程序

DB2 设计顾问程序是可以帮助您显著提高工作负载性能的工具。选择要为复杂工作负载创建哪些索引、具体化查询表（MQT）、集群维或数据库分区的任务可能会令人十分头痛。设计顾问程序可以确定提高工作负载性能所需的所有对象。

如果工作负载中存在一组 SQL 语句，那么设计顾问程序将为下列各项提供一些建议:

- 新的索引
- 新的集群索引
- 新的 MQT
- 对多维集群（MDC）表的转换
- 表的重新分布

设计顾问程序可以立即实现部分或全部这些建议，您也可以安排稍后运行这些建议。

要启动“设计顾问程序”实用程序，请使用 db2advis 命令。

设计顾问程序可以帮助简化下列任务:

规划和建立新的数据库

在设计数据库时，可以使用设计顾问程序来生成索引、MQT、MDC 表或数据库分区在测试环境中的替代设计。

在分区数据库环境中，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前确定适当的数据库分区策略
- 帮助从单一分区数据库升级到多分区数据库
- 帮助从另一个数据库产品迁移到多分区 DB2 数据库

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
- 通过使用样本工作负载的性能作为衡量标准，提高总体数据库性能
- 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
- 确定如何优化新查询的性能
- 对运行状况中心提供的建议作出响应，这些建议针对执行大量排序操作的工作负载的共享内存使用情况或排序堆问题
- 查找工作负载中未使用的对象

设计顾问程序输出

缺省情况下，设计顾问程序的输出将写至标准输出并保存在 ADVISE_* 表中:

- 每次运行设计顾问程序时，都将更新 ADVISE_INSTANCE 表并添加新的一行:
  - START_TIME 和 END_TIME 字段分别显示实用程序的启动时间和停止时间。
  - 如果此实用程序成功结束，那么 STATUS 字段将包含 COMPLETED 值。

© Copyright IBM Corp. 2006, 2009

- MODE 字段指示是否使用了 db2advis 命令的 -m 选项。
- COMPRESSION 字段指示使用的压缩类型。

- 如果已提供 MQT、MDC 表或数据库分区策略建议，那么 ADVISE_TABLE 表中的 USE_TABLE 列将包含 Y 值。

  MQT 建议包含在 ADVISE_MQT 表中；MDC 建议包含在 ADVISE_TABLE 表中；数据库分区策略建议包含在 ADVISE_PARTITION 表中。这些表中的 RUN_ID 列包含一个值，此值与 ADVISE_INSTANCE 表中某一行的 START_TIME 相对应，从而将其链接到设计顾问程序的同一次运行。

  如果已提供 MQT、MDC 或数据库分区建议，那么相关的 ALTER TABLE 存储过程调用将放在 ADVISE_TABLE 表的 ALTER_COMMAND 列中。ALTER TABLE 存储过程调用可能会因对 ALTOBJ 存储过程的表的限制而失败。
- 如果已提供索引建议，那么 ADVISE_INDEX 表中的 USE_INDEX 列将包含 Y 值（已建议或评估索引）或 R 值（已建议将现有的集群 RID 索引取消集群）。
- ADVISE_MQT 表的 COLSTATS 列包含 MQT 的列统计信息。这些统计信息包含在 XML 结构中，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>
```

您可以使用 db2advis 命令的 -o 选项将设计顾问程序的建议保存到文件。保存的设计顾问程序输出包含下列元素:

- 与任何新索引、MQT、MDC 表或数据库分区策略相关联的 CREATE 语句
- 用于 MQT 的 REFRESH 语句
- 用于新对象的 RUNSTATS 命令

此输出的示例如下所示:

```
--<?xml version="1.0"?>
--<design-advisor>
--<mqt>
--<identifier>
--<name>MQT612152202220000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<statementlist>3</statementlist>
--<benefit>1013562.481682</benefit>
--<overhead>1468328.200000</overhead>
--<diskspace>0.004906</diskspace>
--</mqt>
.....
--<index>
```

```
--<identifier>
--<name>IDX612152221400000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<table><identifier>
--<name>PART</name>
--<schema>TPCD </schema>
--</identifier></table>
--<statementlist>22</statementlist>
--<benefit>820160.000000</benefit>
--<overhead>0.000000</overhead>
--<diskspace>9.063500</diskspace>
--</index>
.....
--<statement>
--<statementnum>11</statementnum>
--<statementtext>
--
-- select
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice,
-- sum(l_quantity) from tpcd.customer, tpcd.orders,
-- tpcd.lineitem where o_orderkey in( select
-- l_orderkey from tpcd.lineitem group by l_orderkey
-- having sum(l_quantity) > 300 ) and c_custkey
-- = o_custkey and o_orderkey = l_orderkey group by
-- c_name, c_custkey, o_orderkey, o_orderdate, o_totalprice
-- order by o_totalprice desc, o_orderdate fetch first
-- 100 rows only
--</statementtext>
--<objects>
--<identifier>
--<name>MQT612152202490000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>ORDERS</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>CUSTOMER</name>
--<schema>TPCD </schema>
--</identifier>
--<identifier>
--<name>IDX612152235020000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152235030000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--<identifier>
--<name>IDX612152211360000</name>
--<schema>ZILIO2 </schema>
--</identifier>
--</objects>
--<benefit>2091459.000000</benefit>
--<frequency>1</frequency>
--</statement>
```

此 XML 结构可以包含多个列。这包括列基数（即，每个列中值的数目）以及可选的 HIGH2KEY 和 LOW2KEY 值。

此外，还包括定义的索引所基于的基本表。可使用 benefit 值对索引和 MQT 进行排名。此外，也可以使用 (benefit - overhead) 对索引进行排名以及使用 (benefit - 0.5 * overhead) 对 MQT 进行排名。

索引和 MQT 的列表后面是工作负载中的语句列表，其中包括 SQL 文本、语句编号、根据建议估计的性能提高（改进）程度以及语句使用的表、索引和 MQT 的列表。此输出示例保留了 SQL 文本中的原始间隔，但为了便于阅读，通常将 SQL 文本分为 80 个字符的带注释行。

如果正在使用现有的索引或 MQT 来执行工作负载，那么它们将出现在输出中。

MDC 和数据库分区建议未明确显示在此 XML 输出示例中。

在进行一些小幅修改之后，您可以将此输出文件作为 CLP 脚本来运行，以便创建建议的对象。您可能想执行的修改包括:

- 对所有 RUNSTATS 命令进行组合，使其成为对新对象或修改后的对象执行的单一 RUNSTATS 调用
- 提供更适用的对象名来代替系统生成的标识
- 除去或注释掉您不想立即实现的对象的任何数据定义语言（DDL）

使用设计顾问程序

您可以通过调用 db2advis 命令来运行设计顾问程序。

1. 定义工作负载。请参阅“为设计顾问程序定义工作负载”。
2. 对此工作负载运行 db2advis 命令。

   注: 如果有关数据库的统计信息不是最新的，那么生成的建议就没那么可靠。
3. 解释 db2advis 的输出并进行必要的修改。
4. 根据情况，实现设计顾问程序的建议。

为设计顾问程序定义工作负载

设计顾问程序分析特定工作负载时，它将考虑工作负载中包括的语句的类型、特定语句的出现频率以及数据库特征之类的因素，以便生成一些能够最大程度降低工作负载的运行总成本的建议。

*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。可以对下列各项运行设计顾问程序:

- 通过 db2advis 命令以直接插入方式输入的单个 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

您可以创建新的工作负载文件，也可以修改先前存在的工作负载文件。可以从多种来源将语句导入到文件中，这些来源包括:

- 定界文本文件
- 事件监视器表
- Query Patroller 历史记录数据表（通过从命令行使用 -qp 选项）
- EXPLAIN_STATEMENT 表中的说明语句
- 使用 DB2 快照捕获到的最新 SQL 语句
- 工作负载管理器活动表
- 工作负载管理器事件监视器表（通过从命令行使用 -wlm 选项）

将 SQL 语句导入到工作负载文件之后，可以添加、更改、修改或除去语句以及修改它们的频率。

- 要对动态 SQL 语句运行设计顾问程序，请执行下列操作:
  1. 使用以下命令来复位数据库监视器:

     ```
     db2 reset monitor for database database-name
     ```

  2. 等待适当的时间，以便对数据库执行动态 SQL 语句。
  3. 调用 db2advis 命令并指定 -g 选项。如果要将动态 SQL 语句保存在 ADVISE_WORKLOAD 表中以便将来引用，那么还应该使用 -p 选项。
- 要对工作负载文件中包含的一组 SQL 语句运行设计顾问程序，请执行下列操作:
  1. 手动创建工作负载文件并将各个 SQL 语句用分号隔开，或者从上面列示的一个或多个来源导入 SQL 语句。
  2. 设置工作负载中语句的频率。缺省情况下，对工作负载文件中每个语句指定的频率均为 1。SQL 语句的频率表示该语句相对于其他语句在工作负载中出现的次数。例如，一个特定的 SELECT 语句在工作负载中出现了 100 次，而另一个 SELECT 语句出现了 10 次。为了表示这两个语句的相对频率，您可以指定第一个 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以手动方式更改该语句在工作负载中的频率或权重: `- - # SET FREQUENCY n`，其中 *n* 是要对该语句指定的频率值。
  3. 调用 db2advis 命令，并通过 -i 选项来指定工作负载文件名。
- 要对 ADVISE_WORKLOAD 表中包含的工作负载运行设计顾问程序，请调用 db2advis 命令并通过 -w 选项来指定工作负载名称。

使用设计顾问程序将单分区数据库转换为多分区数据库

您可以使用设计顾问程序帮助您将单分区数据库转换为多分区数据库。

关于此任务

除了提供有关新索引、具体化查询表（MQT）和多维集群（MDC）表的建议以外，设计顾问程序还可以提供有关如何分布数据的建议。

过程

1. 使用 db2licm 命令来注册数据库分区功能部件（DPF）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: 设计顾问程序只能提供有关如何将数据重新分布到现有表空间的建议。
3. 运行设计顾问程序，并在 db2advis 命令中指定分区选项。
4. 在运行设计顾问程序生成的 DDL 语句之前，请稍微修改 db2advis 输出文件。 由于必须先设置数据库分区方式，然后才能运行设计顾问程序所生成的 DDL 脚本，因此在所返回的脚本中，已将建议注释掉。您负责按照建议对表进行变换。

设计顾问程序的局限性和限制

设计顾问程序提供的关于索引、具体化查询表（MQT）、多维集群（MDC）表和数据库分区的建议有一些局限性和限制。

有关索引建议的限制

- 对于 MQT，提供的索引建议旨在提高工作负载性能而不是 MQT 刷新性能。

- 仅对于 MDC 表，才会提供集群 RID 索引建议。设计顾问程序将包括集群 RID 索引作为选项，而不是为表创建 MDC 结构。
- 版本 9.7 的设计顾问程序不会建议您对分区表创建分区索引。建议的所有索引都将使用显式的 NOT PARTITIONED 子句进行创建。

有关 MQT 建议的限制

- 设计顾问程序不会推荐使用增量 MQT。如果您想创建增量 MQT，那么可将 REFRESH IMMEDIATE MQT 转换为使用所选登台表的增量 MQT。
- 对于 MQT，提供的索引建议旨在提高工作负载性能而不是 MQT 刷新性能。
- 如果工作负载不包含更新、插入或删除操作，那么不考虑更新所建议的 REFRESH IMMEDIATE MQT 对性能产生的影响。
- 建议 REFRESH IMMEDIATE MQT 对隐式的唯一键创建唯一索引，隐式的唯一键由 MQT 查询定义的 GROUP BY 子句中的列组成。

有关 MDC 建议的限制

- 对于现有的表，在设计顾问程序为该表考虑 MDC 之前，必须在该表中填充足够的数据。建议最少填充 20 到 30 兆字节的数据。小于 12 个扩展数据块的表将不会被考虑。
- 除非在 db2advis 命令中使用了采样选项 -r，否则不会为新的 MQT 考虑 MDC 建议。
- 设计顾问程序不会为类型表、临时表或联合表生成 MDC 建议。
- 必须要有足够的存储空间（对于大型表，大约为表数据的 1%）可用于存储 db2advis 命令执行期间使用的采样数据。
- 尚未收集统计信息的表将不会被考虑。
- 设计顾问程序不会为多列维生成建议。

有关数据库分区建议的限制

设计顾问程序只能为 DB2 企业服务器版生成数据库分区建议。

其他限制

设计顾问程序在运行时，将创建临时模拟目录表。未完成的运行可能会导致未删除其中的某些表。在这种情况下，可以通过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，指定未完成的执行时使用的用户名）。如果未指定 -f 选项，那么设计顾问程序将只生成除去这些表所需要的 DROP 语句；而不会实际地除去这些表。

注: 从版本 9.5 开始，-f 是缺省选项。这意味着，如果对选择的 MQT 运行 db2advis，那么数据库管理器将使用与模式名相同的用户标识自动删除所有本地模拟目录表。

您应该在目录数据库分区中另外创建一个表空间来存储这些模拟的目录表，并将 CREATE 或 ALTER TABLESPACE 语句的 DROPPED TABLE RECOVERY 选项设置为 OFF。这使得清除工作更容易执行，并且设计顾问程序的执行速度将更快。

第 2 部分 对问题进行故障诊断

为了准确地分析问题，第一步要做的就是完整地描述问题。如果没有问题描述，您就不知道从什么地方开始调查造成问题的原因。

此步骤包括询问自己如下基本问题:

- 症状是什么？
- 问题是在哪里发生的？
- 问题是在何时发生的？
- 发生问题的条件是什么？
- 问题是否可以再现？

通过回答上述及其他问题，就得到了对大多数问题的准确描述，并且也是找出问题解决方案的最好办法。

症状是什么?

开始描述问题时，最明显的问题是“发生了什么问题？”。这看起来像一个很直观的问题；但是，它可以分为若干其他问题，从而更好地描述该问题。这些问题包括:

- 是什么人或什么工具报告该问题的？
- 错误代码和错误消息是什么？
- 怎么失败的？例如: 循环、中止、停止、性能下降或结果错误。
- 对业务有何影响？

问题是在哪里发生的?

确定问题发生的位置并不总是那么容易，但它是解决问题的最重要步骤之一。报告组件和失败组件之间可能存在多层技术。网络、磁盘和驱动程序只是调查问题时要考虑的几个部分。

- 是在特定平台上还是在多个平台上都发生了该问题？
- 是否支持当前环境和配置？
- 应用程序是在数据库服务器本地运行还是在远程服务器上运行？
- 是否涉及网关？
- 数据库是存储在各个磁盘上，还是存储磁盘 RAID 磁盘阵列上？

这些类型的问题可帮助您隔离问题层，并且是确定问题来源所必需的。记住，不能只因为某层报告问题而总是断定那就是问题根源所在。

标识发生问题的位置时应了解发生问题的环境。总是应该花一些时间来完整描述问题环境，包括操作系统、操作系统版本、所有相应软件及版本和硬件信息。确认您正在配置受支持的环境中运行，这是因为许多问题会解释为发现若干软件级别不能在一起运行，或者未在一起经过完整测试。

© Copyright IBM Corp. 2006, 2009

问题是在何时发生的?

问题分析中的另一个必需步骤是创建导致故障的事件的详细时间线，对于从前发生的那些案例尤其如此。可以通过回溯工作过程很轻松地完成此任务：从报告错误时开始（尽可能精确，甚至精确到毫秒），然后通过可用日志和信息回溯工作过程。通常您只需要进行观察，直到您在任何诊断日志中发现的第一个值得怀疑的事件，但是，这并不总是那么容易，通常需要您有实践经验。如果同时存在多层技术，每层都有自己的诊断信息，那么要知道停止的时间就特别困难。

- 问题是否仅在日间或夜间的特定时间发生？
- 问题多久发生一次？
- 导致问题的事件的发生顺序是什么？
- 问题是否发生在环境更改（如升级现有软件或硬件或者安装新的软件或硬件）之后？

回答这类问题可帮助您创建事件的详细时间线，并且为您提供用来进行调查的参考框架。

发生问题的条件是什么?

要完整地描述问题，知道发生问题时还有什么别的软件在运行是非常重要的。如果问题是在特定环境或特定条件下发生的，那么这可能是找出问题原因的关键线索。

- 问题是否总是在执行同一任务时发生？
- 事件是否要按一定顺序发生，问题才会再现？
- 是否存在其他应用程序同一时间失败？

回答这些类型的问题可帮助您说明发生问题的环境，并且能够将所有从属项关联起来。记住，不能只因为同一时间发生了多个问题就表示它们总是相关的。

问题是否可以再现?

从问题描述和调查角度来看，“理想”的问题是可以再现的。对于可再现的问题，您几乎总是可以将它们与提供的一大堆工具或过程配合使用，以帮助进行调查。因此，可再现问题通常比较容易调试和解决。

但是，可再现问题也有缺点：如果该问题对企业有很严重的影响，那么您可能不想让它再现。在这种情况下，最好尽可能在测试或开发环境中再现该问题。

- 能否在测试机器上再现问题？
- 是否多个用户或应用程序都遇到同一类型的问题？
- 能否通过运行单个命令、一组命令、特定应用程序或独立应用程序来再现问题？
- 能否通过从 DB2 命令行输入等价命令/查询来再现问题？

因为在调查时测试或开发环境有更好的灵活性，也更便于控制，所以最好在测试或开发环境中再现容易发生的单个问题。

第 5 章 用于故障诊断的工具

下列工具可以帮助您收集、格式化或分析诊断数据。

- db2dart

  可以使用 db2dart 命令来验证数据库及其对象的体系结构是否正确。还可以使用它来显示数据库控制文件的内容，以便从其他情况下可能无法访问的表中抽取数据。

- db2diag

  db2diag 工具用于对 db2diag 日志文件中的大量信息进行过滤和格式化。通过对 db2diag 日志文件中的记录进行过滤，可以缩短诊断问题时查找所需记录的时间。

- db2greg

  可使用 db2greg 工具来查看和编辑全局注册表。

- db2level

  db2level 命令帮助您确定 DB2 实例的版本和服务级别（构建级别和修订包级别）。

- db2look

  如果能够创建结构类似另一数据库的数据库，有时会非常有利。例如，与在生产系统上测试新应用程序或恢复计划相比，创建结构和数据类似的测试系统然后对其进行测试（这不会对生产系统产生负面影响）更有意义。可使用 db2look 工具来抽取必需的 DDL 语句，这些语句是在一个数据库中再现另一个数据库中的数据库对象所需的。该工具还可生成将统计信息从一个数据库复制至另一个数据库所需的 SQL 语句，以及复制数据库配置、数据库管理器配置和注册表变量所需的语句。

- db2ls

  由于能够在系统上安装 DB2 产品的多个副本，并且能够灵活地在您选择的路径中安装 DB2 产品和功能部件，所以需要使用一个工具来帮助跟踪已经安装了哪些 DB2 产品及其安装位置。在受支持的 Linux 和 UNIX 操作系统上，db2ls 命令将列示安装在系统上的 DB2 产品和功能部件，还会列示 DB2 版本 9 HTML 文档。

- db2pd

  因为 db2pd 工具可从 DB2 内存集合迅速返回即时信息，所以该工具可用于故障诊断。

- db2support

  对 DB2 问题收集信息时，您必须运行的最重要的 DB2 实用程序是 db2support。db2support 实用程序将自动收集所有可用的 DB2 诊断信息和系统诊断信息。它还有一个可选的交互式“问与答”会话，该会话会提出有关问题的详情。

- db2val

  db2val 工具将通过验证安装文件、实例、数据库创建、与该数据库的连接以及分区数据库环境的状态，来验证 DB2 副本的核心功能。

- 跟踪

© Copyright IBM Corp. 2006, 2009

如果遇到与 DB2 相关的反复出现并且可再现的问题，那么执行跟踪有时能够捕获有关该问题的其他信息。在正常情况下，仅当 IBM 软件支持机构要求时，才应该使用跟踪。执行跟踪的过程包括设置跟踪工具、再现错误与收集数据。

- 特定于平台的工具（Windows）（Linux 和 UNIX）

  可以使用 Windows、Linux 和 UNIX 操作系统所附带提供的有用诊断工具来收集和处理数据，这些数据有助于确定系统问题的原因。

db2dart 工具概述

可以使用 db2dart 命令来验证数据库及其对象的体系结构是否正确。还可以使用它来显示数据库控制文件的内容，以便从其他情况下可能无法访问的表中抽取数据。

要显示所有可能的选项，需发出不带任何参数的 db2dart 命令。如果命令行中未显式指定一些需要参数的选项（如表空间标识），那么会提示输入这些参数。

缺省情况下，db2dart 实用程序将创建名为 databaseName.RPT 的报告文件。对于单一分区数据库分区环境，将在当前目录中创建该文件。对于多分区数据库分区环境，将在诊断目录的子目录中创建该文件。该子目录称为 DART####，其中 #### 是数据库分区号。

db2dart 实用程序通过直接从磁盘中读取数据库中的数据和元数据来对其进行访问。因此，决不能对仍具有活动连接的数据库运行该工具。如果存在活动连接，那么该工具将不知道缓冲池中的页面或内存中的控制结构（此处是举例说明），可能会报告假的错误结果。同样，如果对需要进行崩溃恢复或尚未完成前滚恢复的数据库运行 db2dart，由于磁盘上的数据性质不一致，可能会导致类似的不一致情况。

比较 INSPECT 和 db2dart

INSPECT 命令用于检查数据库的体系结构完整性，从而检查数据库的各页是否一致。INSPECT 命令会检查表对象的结构和表空间的结构是否有效。交叉对象验证会创建索引与数据一致性的联机检查。db2dart 命令会检查数据库以了解体系结构是否正确，并报告遇到的所有错误。

INSPECT 命令与 db2dart 命令的相似之处在于它允许您检查数据库、表空间和表。这两个命令之间的主要差别是，在运行 db2dart 之前需要停用数据库，而 INSPECT 需要与数据库连接，并且可以在同时有多个活动数据库连接时运行。

如果不停用数据库，那么 db2dart 将产生不可靠的结果。

下列各表列示了 db2dart 命令和 INSPECT 命令执行的测试之间的差别。

表 76. 针对表空间比较 db2dart 和 INSPECT 的功能

执行的测试	db2dart	INSPECT
SMS 表空间		
检查表空间文件	YES	NO
验证内部页标题字段的内容	YES	YES
DMS 表空间		
检查多个对象指向的扩展数据块映像	YES	NO

表 76. 针对表空间比较 db2dart 和 INSPECT 的功能 (续)

执行的测试	db2dart	INSPECT
检查每个扩展数据块映像页以找出一致性位错误	NO	YES
检查每个空间映射页以找出一致性位错误	NO	YES
验证内部页标题字段的内容	YES	YES
验证扩展数据块映像是否与表空间映射一致	YES	NO

表 77. 针对数据对象比较 db2dart 和 INSPECT 的功能

执行的测试	db2dart	INSPECT
检查数据对象以找出一致性位错误	YES	YES
检查特殊控制行的内容	YES	NO
检查可变长度列的长度和位置	YES	NO
检查表行中的 LONG VARCHAR、LONG VARGRAPHIC 和大对象（LOB）描述符	YES	NO
检查总计页数、已使用的页数和可用空间百分比	NO	YES
验证内部页标题字段的内容	YES	YES
验证每行的记录类型及其长度	YES	YES
验证行是否未重叠	YES	YES

表 78. 针对索引对象比较 db2dart 和 INSPECT 的功能

执行的测试	db2dart	INSPECT
检查一致性位错误	YES	YES
检查索引键的位置和长度以及是否存在重叠	YES	YES
检查索引中键的顺序	YES	NO
确定总计页数和已使用的页数	NO	YES
验证内部页标题字段的内容	YES	YES
验证唯一键的唯一性	YES	NO
检查给定索引条目的数据行是否存在	NO	YES
验证数据值的每个键	NO	YES

表 79. 针对块映射对象比较 db2dart 和 INSPECT 的功能

执行的测试	db2dart	INSPECT
检查一致性位错误	YES	YES
确定总计页数和已使用的页数	NO	YES
验证内部页标题字段的内容	YES	YES

表 80. 针对长整型字段和 LOB 对象比较 db2dart 和 INSPECT 的功能

执行的测试	db2dart	INSPECT
检查分配结构	YES	YES
确定总计页数和已使用的页数（仅适用于 LOB 对象）	NO	YES

此外，还可以使用 db2dart 命令执行下列操作:

- 格式化和转储数据页
- 格式化和转储索引页
- 将数据行格式化为定界 ASCII
- 将索引标记为无效

INSPECT 命令不能用于执行这些操作。

使用 db2diag 工具来分析 db2diag 日志文件

供数据库和系统管理员使用的主日志文件为管理通知日志。db2diag 日志文件旨在供 IBM 软件支持机构用于进行故障诊断。

管理通知日志消息也以标准化消息格式记录到 db2diag 日志文件。

db2diag 工具用于对 db2diag 日志文件中的大量信息进行过滤和格式化。过滤 db2diag 日志文件记录可缩短诊断问题时查找所需记录的时间。

示例 1: 按数据库名称过滤 db2diag 日志文件

如果实例中有若干数据库，并且您只希望显示与数据库“SAMPLE”有关的信息，那么可以按如下所示过滤 db2diag 日志文件:

```
db2diag -g db=SAMPLE
```

因此，将仅显示包含“DB: SAMPLE”的 db2diag 日志文件记录，如:

```
2006-02-15-19.31.36.114000-300 E21432H406          级别: 错误
PID: 940                  TID: 660          PROC: db2syscs.exe
实例: DB2                      节点: 000          数据库: SAMPLE
APPHDL: 0-1056                 APPID: *LOCAL.DB2.060216003103
函数: DB2 UDB, 基本系统实用程序, sqleDatabaseQuiesce, 探测点: 2
消息: ADM7507W  数据库停顿请求成功完成。
```

示例 2: 按进程标识过滤 db2diag 日志文件

以下命令可用来显示进程标识（PID）为 2200，并且在分区 0、1、2 或分区 3 上运行的进程生成的所有严重错误消息:

```
db2diag -g level=Severe,pid=2200 -n 0,1,2,3
```

注意，此命令可能以不同方式写入，包括 db2diag -l severe -pid 2200 -n 0,1,2,3。还要注意的是，**-g** 选项指定区分大小写的搜索，所以此处“Severe”会起作用，但如果使用了“severe”则会失败。满足如下要求时，这些命令将成功检索 db2diag 日志文件记录:

```
2006-02-13-14.34.36.027000-300 I18366H421          级别: 严重
PID     : 2200                 TID  : 660          PROC : db2syscs.exe
实例: DB2                      节点: 000           数据库: SAMPLE
APPHDL: 0-1433               APPID: *LOCAL.DB2.060213193043
函数: DB2 UDB, 数据管理, sqldPoolCreate, 探测点: 273
返回码: ZRC=0x8002003C=-2147352516=SQLB_BAD_CONTAINER_PATH
                    "错误的容器路径"
```

示例 3: 格式化 db2diag 工具输出

以下命令过滤 2006 年 1 月 1 日之后发生，并且包含分区 0、1 或2 上记录的所有非严重错误和严重错误的所有记录。它会输出匹配的记录，因此时间戳记、分区号和级别将出现在第一行上，而 pid、tid 和实例名将出现在第二行上，之后是错误消息:

```
db2diag -time 2006-01-01 -node "0,1,2" -level "Severe, Error" |
db2diag -fmt "Time: %{ts}
分区: %node Message Level: %{level} \nPid: %{pid}  Tid: %{tid}
实例: %{instance}\nMessage: @{msg}\n"
```

生成的输出示例如下所示:

```
时间: 2006-02-15-19.31.36.099000 分区: 000 消息级别: 错误
Pid: 940 Tid: 40 实例: DB2
消息: ADM7506W 已经请求了数据库停顿。
```

有关更多信息，请发出下列命令:

- db2diag -help 提供了所有可用选项的简短描述
- db2diag -h brief 提供对所有不带示例的选项的描述
- db2diag -h notes 提供用法说明和限制
- db2diag -h examples 提供一小组示例以帮助您入门
- db2diag -h tutorial 提供所有可用选项的示例
- db2diag -h all 提供最完整的选项列表

示例 4: 过滤来自不同工具的消息

下列示例显示如何仅查看来自数据库管理器中的特定工具（或所有工具）的消息。受支持的工具包括:

- ALL，这会返回来自所有工具的记录
- MAIN，这会返回来自 DB2 常规诊断日志的记录，例如 db2diag 日志文件和管理通知日志
- OPSTATS，这会返回与优化器统计信息有关的记录

要读取来自 MAIN 工具的消息，请使用以下命令:

```
db2diag -facility MAIN
```

要显示来自 OPSTATS 工具的消息并滤出级别为 Severe 的记录，请使用以下命令:

```
db2diag -fac OPSTATS -level Severe
```

要显示来自所有可用工具的消息并滤出实例为 harmistr 级别为 Error 的记录，请使用以下命令:

```
db2diag -fac all -g instance=harmistr,level=Error
```

要显示 OPSTATS 工具中级别为 Error 并且以特定格式输出时间戳记和 PID 字段的所有消息，请使用以下命令:

```
db2diag -fac opstats -level Error -fmt " Time :%{ts} Pid :%{pid}"
```

示例 5: 根据时间戳记来合并文件和对记录进行排序

此示例说明了如何根据时间戳记来合并两个或更多 db2diag 日志文件以及对记录进行排序。

要合并的两个 db2diag 日志文件为如下所示:

- db2diag.0.log; 它包含具有以下时间戳记的 Level:Error 的记录:
  - 2009-02-26-05.28.49.822637
  - 2009-02-26-05.28.49.835733
  - 2009-02-26-05.28.50.258887
  - 2009-02-26-05.28.50.259685
- db2diag.1.log; 它包含具有以下时间戳记的 Level:Error 的记录:
  - 2009-02-26-05.28.11.480542
  - 2009-02-26-05.28.49.764762
  - 2009-02-26-05.29.11.872184
  - 2009-02-26-05.29.11.872968

要根据时间戳记来合并两个诊断日志文件以及对记录进行排序，请执行以下命令:

```
db2diag -merge db2diag.0.log db2diag.1.log -fmt %{ts} -level error
```

执行合并以及对记录进行排序的结果为如下所示:

- 2009-02-26-05.28.11.480542
- 2009-02-26-05.28.49.764762
- 2009-02-26-05.28.49.822637
- 2009-02-26-05.28.49.835733
- 2009-02-26-05.28.50.258887
- 2009-02-26-05.28.50.259685
- 2009-02-26-05.29.11.872184
- 2009-02-26-05.29.11.872968

时间戳记是按时间顺序进行合并和排序的。

示例 6: 按时间戳记合并分割诊断目录路径文件以及对记录进行排序

此示例说明了如何合并当前主机上的三个数据库分区中的文件。为了获得分割诊断目录路径，按以下方式设置了 **diagpath** 数据库管理器配置参数:

```
db2 update dbm cfg using diagpath '"$n"'
```

以下是要合并的三个 db2diag 日志文件的列表:

- ~/sqllib/db2dump/NODE0000/db2diag.log
- ~/sqllib/db2dump/NODE0001/db2diag.log
- ~/sqllib/db2dump/NODE0002/db2diag.log

要根据时间戳记来合并这三个诊断日志文件以及对记录进行排序，请执行以下命令:

```
db2diag -merge
```

使用 db2greg 显示和更改全局注册表（UNIX）

在 UNIX 和 Linux 平台上，可以使用 db2greg 命令来查看全局注册表。

在 DB2 版本 9.7 和更高版本中，DB2 全局概要文件注册表并非记录在文本文件 <DB2DIR>/default.env 中。现在，使用全局注册表文件 global.reg 来注册与当前 DB2 安装版本相关的 DB2 全局概要文件设置。

全局注册表只存在于 UNIX 和 Linux 平台上:

- 对于 root 安装，全局注册表文件为 /var/db2/global.reg（HP-UX 上为 /var/opt/db2/global.reg）。
- 对于非 root 安装，全局注册表文件为 $HOME/sqllib/global.reg，其中 $HOME 是非 root 用户的主目录。

全局注册表由以下三种不同记录类型组成:

- “服务”: 服务记录包含产品级别的信息，如版本和安装路径。
- “实例”: 实例记录包含实例级别的信息，如实例名、实例路径、版本和“引导时启动”标志。
- “变量”: 变量记录包含变量级别的信息，如变量名和变量值。
- 注释。

可使用 db2greg 工具查看全局注册表。此工具位于 sqllib/bin 和 bin 下的 install 目录中（以供作为 root 用户登录时使用）。

可使用 db2greg 工具编辑全局注册表。在 root 安装中编辑全局注册表需要 root 权限。

仅当 IBM 软件支持机构要求使用 db2greg 工具时，您才必须使用该工具。

标识产品的版本和服务级别

db2level 命令帮助您确定 DB2 实例的版本和服务级别（构建级别和修订包级别）。

要确定 DB2 实例是否为最新服务级别，请比较 db2level 输出和 DB2 支持机构 Web 站点 www.ibm.com/support/docview.wss?rs=71&uid=swg27007053 上的修订包下载页面中的信息。

在 Windows 系统上运行 db2level 命令的典型结果为:

```
DB21085I 实例"DB2"使用"32"位和 DB2 代码发行版"SQL09010"，其级别标识为
"01010107"。
参考标记为"DB2 v9.1.0.189"、"n060119"、""和修订包"0"。
产品安装在"c:\SQLLIB"中，DB2 副本名称为"db2build"。
```

这四个参考标记的组合唯一地标识 DB2 实例的精确服务级别。在与 IBM 软件支持机构联系以获取帮助时，此信息非常重要。

对于 JDBC 或 SQLJ 应用程序，如果使用的是用于 SQLJ 和 JDBC 的 IBM DB2 驱动程序，可通过运行 db2jcc 实用程序来确定驱动程序的级别:

```
db2jcc -version

IBM DB2 JDBC Driver Architecture 2.3.63
```

使用 db2look 模拟数据库

如果能够创建结构类似另一数据库的数据库，有时会非常有利。例如，与在生产系统上测试新应用程序或恢复计划相比，创建结构和数据类似的测试系统然后对其进行测试更有意义。

这样生产系统不会受到测试的负面性能影响，或者因为错误的应用程序导致的意外数据毁坏而造成的影响。而且，在调查问题（如无效结果、性能问题等）时，会更容易在与生产系统完全相同的测试系统上调试该问题。

可使用 db2look 工具来抽取必需的 DDL 语句，这些语句是在一个数据库中再现另一个数据库中的数据库对象所需的。该工具还可生成将统计信息从一个数据库复制至另一个数据库所需的 SQL 语句，以及复制数据库配置、数据库管理器配置和注册表变量所需的语句。因为新数据库可能未包含与原始数据库完全相同的一组数据，但您仍然要对两个系统选择相同的访问计划，所以这一点是非常重要的。只应对正在版本 9.5 和更高级别的 DB2 服务器上运行的数据库发出 db2look 命令。

db2look 工具在 *DB2 Command Reference* 中作了详细描述，但您可以通过执行该工具（不带任何参数）来查看选项列表。可使用 -h 选项来显示更多详细用法。

使用 db2look 来模拟数据库中的表

要在数据库中抽取表的 DDL，使用 -e 选项。例如，创建 SAMPLE 数据库的副本 SAMPLE2，这样会在新数据库中创建第一个数据库中的所有对象。

```
C:\>db2 create database sample2
DB20000I 成功完成了 CREATE DATABASE 命令。
C:\>db2look -d sample -e > sample.ddl
-- 用户为:
-- 为表创建 DDL
-- 正在自动绑定程序包 ...
-- 绑定成功
-- 正在自动绑定程序包 ...
-- 绑定成功
```

注: 如果还想对用户定义的空间、数据库分区组和缓冲池生成 DDL，那么在上述命令中 -e 的后面添加 -l 标志。将不会抽取缺省数据库分区组、缓冲池和表空间。这是因为在缺省情况下它们已经存在于每个数据库中。如果希望模拟它们，那么必须手动对其进行更改。

在文本编辑器中打开 sample.ddl 文件。因为您要对新数据库运行此文件中的 DDL，所以必须将 CONNECT TO SAMPLE 语句更改为 CONNECT TO SAMPLE2。如果使用了 -l 选项，那么可能需要更改与表空间命令相关联的路径，这样它们也会指向适当的路径。请抽空查看文件内容的余下部分。您应该会看到样本数据库中的所有用户表的 CREATE TABLE、ALTER TABLE 和 CREATE INDEX 语句:

```
...
------------------------------------------------
-- 表 "DB2"."ORG" 的 DDL 语句
------------------------------------------------

  CREATE TABLE "DB2"."ORG"  (
```

```
    "DEPTNUMB" SMALLINT NOT NULL ,
    "DEPTNAME" VARCHAR(14) ,
    "MANAGER" SMALLINT ,
    "DIVISION" VARCHAR(10) ,
    "LOCATION" VARCHAR(13) )
   IN "USERSPACE1" ;
...
```

更改连接语句后，运行以下语句:

```
C:\>db2 -tvf sample.ddl > sample2.out
```

查看 sample2.out 输出文件 - 每一项都应该已经成功执行。如果发生了错误，错误消息应指示问题所在。修正这些问题并再次运行这些语句。

如输出所示，将导出所有用户表的 DDL。这是缺省行为，但对于包括的表，还有更具体的其他选项可用。例如，如果希望仅包括 STAFF 和 ORG 表，那么使用 -t 选项:

```
C:\>db2look -d sample -e -t staff org > staff_org.ddl
```

如果希望仅包括带有模式 DB2 的表，那么使用 -z 选项:

```
C:\>db2look -d sample -e -z db2 > db2.ddl
```

模拟表的统计信息

如果测试数据库用于测试性能或调试性能问题，那么对两个数据库生成的访问方案应该完全相同。优化器根据统计信息、配置参数、注册表变量和环境变量来生成访问方案。如果两个系统间的这些设置完全相同，那么访问方案很可能是相同的。

如果将完全相同的数据装入到两个数据库中，并且对它们两个执行相同的 RUNSTATS 选项，那么统计信息也应该完全相同。但是，如果数据库包含不同数据，或者如果测试数据库中只使用数据的子集，那么统计信息可能会有显著差别。在这种情况下，可使用 db2look 来从生产数据库收集统计信息并将它们放在测试数据库中。通过对可更新的目录表的 SYSSTAT 集合创建 UPDATE 语句并对所有表创建 RUNSTATS 命令来执行此操作。

用于创建统计信息语句的选项为 -m。返回至 SAMPLE/SAMPLE2 示例，从 SAMPLE 收集统计信息并将其添加至 SAMPLE2:

```
C:\>db2look -d sample -m > stats.dml
-- 用户为:
-- 以模拟方式运行 db2look
```

像以前一样，必须编辑输出文件以便将 CONNECT TO SAMPLE 语句更改为 CONNECT TO SAMPLE2。而且再查看一下文件内容的余下部分，以了解某些 RUNSTATS 和 UPDATE 语句包含的内容:

```
...
-- 模拟表 ORG
RUNSTATS ON TABLE "DB2"."ORG" ;
UPDATE SYSSTAT.INDEXES
SET NLEAF=-1,
        NLEVELS=-1,
        FIRSTKEYCARD=-1,
        FIRST2KEYCARD=-1,
        FIRST3KEYCARD=-1,
        FIRST4KEYCARD=-1,
        FULLKEYCARD=-1,
        CLUSTERFACTOR=-1,
        CLUSTERRATIO=-1,
```

```
        SEQUENTIAL_PAGES=-1,
        PAGE_FETCH_PAIRS='',
        DENSITY=-1,
        AVERAGE_SEQUENCE_GAP=-1,
        AVERAGE_SEQUENCE_FETCH_GAP=-1,
        AVERAGE_SEQUENCE_PAGES=-1,
        AVERAGE_SEQUENCE_FETCH_PAGES=-1,
        AVERAGE_RANDOM_PAGES=-1,
        AVERAGE_RANDOM_FETCH_PAGES=-1,
        NUMRIDS=-1,
        NUMRIDS_DELETED=-1,
        NUM_EMPTY_LEAFS=-1
WHERE TABNAME = 'ORG' AND TABSCHEMA = 'DB2   ';
...
```

使用抽取 DDL 的 -e 选项的同时，可使用 -t 和 -z 选项来指定一组表。

抽取配置参数和环境变量

优化器根据统计信息、配置参数、注册表变量和环境变量来选择计划。可将统计信息与 db2look 一起使用来生成必需的配置更新和设置语句。此操作使用 -f 选项完成。例如:

```
c:\>db2look -d sample -f>config.txt
-- 用户为: DB2INST1
-- 正在自动绑定程序包 ...
-- 绑定成功
-- 正在自动绑定程序包 ...
-- 绑定成功
```

config.txt 包含类似如下的输出:

```
-- 此 CLP 文件是使用 DB2LOOK 版本 9.1 创建的
-- 时间戳记: 2/16/2006 7:15:17 PM
-- 数据库名称: SAMPLE
-- 数据库管理器版本: DB2/NT 版本 9.1.0
-- 数据库代码页: 1252
-- 数据库整理顺序为: UNIQUE


CONNECT TO SAMPLE;

------------------------------------------------------
-- 数据库和数据库管理器配置参数
------------------------------------------------------

UPDATE DBM CFG USING cpuspeed 2.991513e-007;
UPDATE DBM CFG USING intra_parallel NO;
UPDATE DBM CFG USING comm_bandwidth 100.000000;
UPDATE DBM CFG USING federated NO;

...

---------------------------------
-- 环境变量设置
---------------------------------


COMMIT WORK;

CONNECT RESET;
```

注: 将仅包括影响 DB2 编译器的那些参数和变量。如果影响编译器的注册表变量设置为其缺省值，那么它不会显示在“环境变量设置”下面。

列示系统上安装的 DB2 数据库产品（Linux 和 UNIX）

在受支持的 Linux 和 UNIX 操作系统上，db2ls 命令将列示安装在系统上的 DB2 数据库产品和功能部件，其中包括 DB2 版本 9.7 HTML 文档。

开始前

root 用户必须至少安装了一个 DB2 版本 9（或更高版本）数据库产品，/usr/local/bin 目录中才会提供指向 db2ls 命令的符号链接。

关于此任务

由于能够在系统上安装 DB2 数据库产品的多个副本，并且能够灵活地在您选择的路径中安装 DB2 数据库产品和功能部件，所以需要使用一个工具来帮助跟踪已经安装了哪些 DB2 产品及其安装位置。在受支持的 Linux 和 UNIX 操作系统上，db2ls 命令将列示安装在系统上的 DB2 产品和功能部件，其中包括 DB2 HTML 文档。

在安装介质中以及系统上的 DB2 安装副本中，都可以找到 db2ls 命令。可以从其中任何一个位置运行 db2ls 命令。可以从所有产品（IBM 数据服务器驱动程序包除外）的安装介质中运行 db2ls 命令。

可以使用 db2ls 命令来列示:

- DB2 数据库产品在系统上的安装位置，并且会列示 DB2 数据库产品级别
- 特定安装路径中的所有或特定 DB2 数据库产品和功能部件

限制

根据所使用的标识，db2ls 命令列示的输出会有所不同:

- 当使用 root 用户权限运行 db2ls 命令时，仅查询 root 用户 DB2 安装。
- 当使用非 root 用户标识运行 db2ls 命令时，将查询 root 用户 DB2 安装和匹配的非 root 用户标识拥有的非 root 用户安装。不查询由其他非 root 用户标识拥有的 DB2 安装。

db2ls 命令是可用来查询 DB2 数据库产品的唯一方法。您*不能*使用 Linux 或 UNIX 操作系统本机实用程序（如 pkginfo、rpm、SMIT 或 swlist）来查询 DB2 数据库产品。包含用于查询 DB2 安装并与其进行交互的本机安装实用程序的任何现有脚本必须进行更改。

在 Windows 操作系统上*不能*使用 db2ls 命令。

过程

- 要列示 DB2 数据库产品在系统上的安装路径和 DB2 数据库产品级别，请输入以下命令:

  ```
  db2ls
  ```

  该命令列示安装在系统上的每个 DB2 数据库产品的下列信息:

  – 安装路径
  – 级别
  – 修订包
  – 特殊安装编号。此列由 IBM DB2 支持人员使用。

- 安装日期。此列显示上次修改 DB2 数据库产品的时间。
- 安装程序用户标识。此列显示用于安装 DB2 数据库产品的用户标识。

- 要列示有关特定安装路径中的 DB2 数据库产品或功能部件的信息，必须指定 **q** 参数:

  ```
  db2ls -q -p -b baseInstallDirectory
  ```

  其中:

  - **q** 指定您要查询产品或功能部件。此参数是必需的。如果查询了 DB2 版本 8 产品，那么会返回空白值。
  - **p** 指定列表显示产品而不是列示功能部件。
  - **b** 参数指定产品或功能部件的安装目录。此参数是固定的，如果不是从安装目录运行该命令。

结果

根据所提供的参数，该命令将列示以下信息:

- 安装路径。安装路径只需指定一次，无须对每个功能部件都指定它。
- 将显示以下信息:
  - 已安装功能部件的响应文件标识，或者如果指定了 **p** 选项，已安装产品的响应文件标识。例如，ENTERPRISE_SERVER_EDITION。
  - 功能部件名称，或者如果指定了 **p** 选项，产品名。
  - 产品版本、发行版、修改级别和修订包级别（VRMF）。例如，9.5.0.0
  - 修订包（如果有）。例如，如果安装了修订包 1，那么显示的值是 1。这包括临时修订包，例如修订包 1a。
- 如果任何产品的 VRMF 信息不匹配，在输出列表末尾都将显示一条警告消息。该消息建议应用此修订包。

使用 db2pd 命令进行监视和故障诊断

因为 db2pd 命令可从 DB2 内存集合迅速返回即时信息，所以该命令可用于故障诊断。

概述

该工具不需要获得任何锁存器或使用任何引擎资源就可以收集信息。因此，在 db2pd 收集信息时，有可能（并且预计）会检索到正在更改的信息；这样，数据可能不是十分准确。如果遇到正在更改的内存指针，可使用信号处理程序来防止 db2pd 异常结束。这可能会导致输出中出现诸如以下的消息: “正在更改的数据结构已强制终止命令”。虽然如此，该工具对于故障诊断却非常有用。在不锁存的情况下收集信息有两个好处: 检索速度更快并且不会争用引擎资源。

如果要在出现特定 SQLCODE、ZRC 代码或 ECF 代码时捕获关于数据库管理系统的信息，那么可以使用 db2pdcfg -catch 命令完成此操作。捕获到错误时，将启动 db2cos（调出脚本）。db2cos 脚本可以动态地进行更改，以便运行解决问题所需的任何 db2pd 命令、操作系统命令或任何其他命令。在 UNIX 和 Linux 上，模板 db2cos 脚本文件在 sqllib/bin 中。在 Windows 操作系统上，db2cos 在 $DB2PATH\bin 目录中。

添加新节点时，您可以监视数据库分区服务器上的操作进度；即，添加节点时，请使用带有可选的 oldviewapps 和 detail 参数的 db2pd -addnode 命令来获取更详细的信息。

要获取当前处于活动状态或者曾处于活动状态但由于某种原因而被停用的事件监视器的列表，请运行 db2pd -gfw 命令。此命令还将返回关于每个快速写程序 EDU 的目标的统计信息和信息（事件监视器将数据写入那些目标）。

示例

以下是使用 db2pd 命令提高故障诊断速度的一组示例:

- 示例: 诊断锁定等待
- 示例 2: 使用 **-wlocks** 参数来捕获正在等待的所有锁定
- 示例 3: 使用 **-apinfo** 参数来捕获关于锁定所有者和锁定等待者的详细运行时信息
- 示例 4: 在考虑锁定问题时使用调出脚本
- 示例 5: 将应用程序映射到动态 SQL 语句
- 示例 6: 监视内存使用情况
- 示例 7: 确定哪个应用程序耗尽表空间
- 示例 8: 监视恢复
- 示例 9: 确定事务正在使用的资源量
- 示例 10: 监视日志使用情况
- 示例 11: 查看综合系统（Sysplex）列表
- 示例 12: 生成堆栈跟踪
- 示例 13: 查看数据库分区的内存统计信息

示例 1: 诊断锁定等待

如果运行 db2pd -db *databasename* -locks -transactions -applications -dynamic，那么结果将类似于:

```
Locks:
Address            TranHdl Lockname                      Type Mode Sts Owner Dur HldCnt Att    ReleaseFlg
0x07800000202E5238 3       00020002000000040000000052 Row  ..X  G   3     1   0      0x0000 0x40000000
0x07800000202E4668 2       00020002000000040000000052 Row  ..X  W*  2     1   0      0x0000 0x40000000
```

对于使用 -db 数据库名称选项指定的数据库，开头的结果会显示该数据库的锁定。此结果表明，TranHdl 2 正在等待 TranHdl 3 挂起的锁定。

```
Transactions:
Address           AppHandl [nod-index] TranHdl Locks State Tflag      Tflag2     Firstlsn       Lastlsn        LogSpace SpaceReserved TID           AxRegCnt GXID
0x0780000020251B80 11       [000-00011] 2       4     READ  0x00000000 0x00000000 0x000000000000 0x000000000000 0        0             0x0000000000B7 1        0
0x0780000020252900 12       [000-00012] 3       4     WRITE 0x00000000 0x00000000 0x000000FA000C 0x000000FA000C 113      154           0x0000000000B8 1        0
```

您会发现 TranHdl 2 与 AppHandl 11 相关联，而 TranHdl 3 与 AppHandl 12 相关联。

```
Applications:
Address           AppHandl [nod-index] NumAgents CoorPid Status       C-AnchID C-StmtUID L-AnchID L-StmtUID Appid

0x07800000006879E0 12       [000-00012] 1         1073336 UOW-Waiting   0        0         17       1         *LOCAL.burford.060303225602
0x0780000000685E80 11       [000-00011] 1         1040570 UOW-Executing 17       1         94       1         *LOCAL.burford.060303225601
```

您会发现 AppHandl 12 最后运行动态语句 17, 1。ApplHandl 11 是当前正在运行的动态语句 17, 1，而最后运行的语句是 94, 1。

```
Dynamic SQL Statements:
Address            AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x07800000209FD800 17     1       1      1      2      2      update pdtest set c1 = 5
0x07800000209FCCC0 94     1       1      1      2      2      set lock mode to wait 1
```

您会发现，文本列显示与锁定超时相关联的 SQL 语句。

示例 2: 使用 **-wlocks** 参数来捕获正在等待的所有锁定

如果运行 db2pd -wlocks -db pdtest，那么将生成类似于以下的结果。这些结果表明，第一个应用程序（AppHandl 47）正在对表执行插入，并且第二个应用程序（AppHandl 46）正在对该表执行选择:

```
venus@boson:/home/venus =>db2pd -wlocks -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:22

Locks being waited on :
AppHandl [nod-index] TranHdl    Lockname                   Type    Mode Conv Sts CoorEDU    AppName  AuthID   AppID
47       [000-00047] 8          00020004000000000840000652 Row     ..X       G   5160       db2bp    VENUS    *LOCAL.venus.071207213730
46       [000-00046] 2          00020004000000000840000652 Row     .NS       W   5913       db2bp    VENUS    *LOCAL.venus.071207213658
```

示例 3: 使用 **-apinfo** 参数来捕获关于锁定所有者和锁定等待者的详细运行时信息

以下样本输出的生成条件与示例 2 相同:

```
venus@boson:/home/venus =>db2pd -apinfo 47 -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:30

Application :
  Address :                0x0780000001676480
  AppHandl [nod-index] :   47       [000-00047]
  Application PID :        876558
  Application Node Name :  boson
  IP Address:              n/a
  Connection Start Time :  (1197063450)Fri Dec  7 16:37:30 2007
  Client User ID :         venus
  System Auth ID :         VENUS
  Coordinator EDU ID :     5160
  Coordinator Partition :  0
  Number of Agents :       1
  Locks timeout value :    4294967294 seconds
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 2
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     UOW-Waiting
  Application Name :       db2bp
  Application ID :         *LOCAL.venus.071207213730

  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         n/a
  ClientAccntng :          n/a

  List of inactive statements of current UOW :
    UOW-ID :          2
    Activity ID :     1
    Package Schema :  NULLID
    Package Name :    SQLC2G13
    Package Version :
    Section Number :  203
    SQL Type :        Dynamic
    Isolation :       CS
```

```
    Statement Type :  DML, Insert/Update/Delete
    Statement :       insert into pdtest values 99


venus@boson:/home/venus =>db2pd -apinfo 46 -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:39

Application :
  Address :                0x0780000000D77A60
  AppHandl [nod-index] :   46       [000-00046]
  Application PID :        881102
  Application Node Name :  boson
  IP Address:              n/a
  Connection Start Time :  (1197063418)Fri Dec  7 16:36:58 2007
  Client User ID :         venus
  System Auth ID :         VENUS
  Coordinator EDU ID :     5913
  Coordinator Partition :  0
  Number of Agents :       1
  Locks timeout value :    4294967294 seconds
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 1
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     Lock-wait
  Application Name :       db2bp
  Application ID :         *LOCAL.venus.071207213658

  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         n/a
  ClientAccntng :          n/a

  List of active statements :
   *UOW-ID :          3
    Activity ID :     1
    Package Schema :  NULLID
    Package Name :    SQLC2G13
    Package Version :
    Section Number :  201
    SQL Type :        Dynamic
    Isolation :       CS
    Statement Type :  DML, Select (blockable)
    Statement :       select * from pdtest
```

示例 4: 在考虑锁定问题时使用调出脚本

要使用调出脚本，请查找 db2cos 输出文件。该文件的位置由数据库管理器配置参数 **diagpath** 控制。输出文件的内容将随您在 db2cos 脚本文件中输入的命令不同而有所变化。当 db2cos 脚本文件包含 db2pd -db sample -locks 命令时，提供的输出示例如下所示:

```
Lock Timeout Caught
Thu Feb 17 01:40:04 EST 2006
Instance DB2
Datbase: SAMPLE
Partition Number: 0
PID: 940
TID: 2136
Function: sqlplnfd
Component: lock manager
Probe: 999
Timestamp: 2006-02-17-01.40.04.106000
```

```
AppID: *LOCAL.DB2...
AppHdl:
...
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:06:53
Locks:
Address    TranHdl Lockname                   Type Mode Sts Owner Dur HldCnt Att Rlse
0x402C6B30 3       000200030000000040000000052 Row  ..X  W*  3     1   0      0   0x40
```

在输出中，W* 指示发生超时的锁定。在本例中，已发生锁定等待。当锁定转换为更高级的方式时，也会发生锁定超时。这由输出中的 C* 指示。

可以使用 db2cos 文件中的其他 db2pd 命令所提供的输出将结果映射至事务、应用程序、代理程序甚至 SQL 语句。可以缩小输出范围或使用其他命令来收集需要的信息。例如，可以使用 db2pd **-locks wait** 参数以便只打印处于等待状态的锁定。另外，还可以使用 **-app** 和 **-agent** 参数。

示例 5: 将应用程序映射至动态 SQL 语句

db2pd -applications -dynamic 命令报告动态 SQL 语句的当前和最后一个锚点标识和语句唯一标识。这允许直接从应用程序映射至动态 SQL 语句。

```
Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid  Status
0x00000002006D2120 780      [000-00780] 1          10615    UOW-Executing

C-AnchID C-StmtUID  L-AnchID L-StmtUID  Appid
163      1          110      1          *LOCAL.burford.050202200412

Dynamic SQL Statements:
Address            AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x0000000220A02760 163    1       2      2      2      1      CREATE VIEW MYVIEW
0x0000000220A0B460 110    1       2      2      2      1      CREATE VIEW YOURVIEW
```

示例 6: 监视内存使用情况

当您尝试了解内存使用情况时，db2pd -memblock 命令非常有用，如以下样本输出所示:

```
All memory blocks in DBMS set.

Address            PoolID   PoolName   BlockAge   Size(Bytes) I LOC   File
0x0780000000740068 62       resynch    2          112         1 1746  1583816485
0x0780000000725688 62       resynch    1          108864      1 127   1599127346
0x07800000001F4348 57       ostrack    6          5160048     1 3047  698130716
0x07800000001B5608 57       ostrack    5          240048      1 3034  698130716
0x07800000001A0068 57       ostrack    1          80          1 2970  698130716
0x07800000001A00E8 57       ostrack    2          240         1 2983  698130716
0x07800000001A0208 57       ostrack    3          80          1 2999  698130716
0x07800000001A0288 57       ostrack    4          80          1 3009  698130716
0x0780000000700068 70       apmh       1          360         1 1024  3878879032
0x07800000007001E8 70       apmh       2          48          1 914   1937674139
0x0780000000700248 70       apmh       3          32          1 1000  1937674139
...
```

接下来是已排序的“性能池”输出:

```
Memory blocks sorted by size for ostrack pool:
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
57         ostrack    5160048              1          3047  698130716
57         ostrack    240048               1          3034  698130716
57         ostrack    240                  1          2983  698130716
57         ostrack    80                   1          2999  698130716
57         ostrack    80                   1          2970  698130716
57         ostrack    80                   1          3009  698130716
Total size for ostrack pool: 5400576 bytes

Memory blocks sorted by size for apmh pool:
```

```
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
70         apmh       40200                2          121   2986298236
70         apmh       10016                1          308   1586829889
70         apmh       6096                 2          4014  1312473490
70         apmh       2516                 1          294   1586829889
70         apmh       496                  1          2192  1953793439
70         apmh       360                  1          1024  3878879032
70         apmh       176                  1          1608  1953793439
70         apmh       152                  1          2623  1583816485
70         apmh       48                   1          914   1937674139
70         apmh       32                   1          1000  1937674139
Total size for apmh pool: 60092 bytes
...
```

最后一部分输出对整个内存集的内存使用者进行排序:

```
All memory consumers in DBMS memory set:
PoolID     PoolName   TotalSize(Bytes)     %Bytes TotalCount %Count LOC   File
57         ostrack    5160048              71.90  1          0.07   3047  698130716
50         sqlch      778496               10.85  1          0.07   202   2576467555
50         sqlch      271784               3.79   1          0.07   260   2576467555
57         ostrack    240048               3.34   1          0.07   3034  698130716
50         sqlch      144464               2.01   1          0.07   217   2576467555
62         resynch    108864               1.52   1          0.07   127   1599127346
72         eduah      108048               1.51   1          0.07   174   4210081592
69         krcbh      73640                1.03   5          0.36   547   4210081592
50         sqlch      43752                0.61   1          0.07   274   2576467555
70         apmh       40200                0.56   2          0.14   121   2986298236
69         krcbh      32992                0.46   1          0.07   838   698130716
50         sqlch      31000                0.43   31         2.20   633   3966224537
50         sqlch      25456                0.35   31         2.20   930   3966224537
52         kerh       15376                0.21   1          0.07   157   1193352763
50         sqlch      14697                0.20   1          0.07   345   2576467555
...
```

在 UNIX 和 Linux 操作系统上，还可以报告专用内存的内存块。例如，如果运行 db2pd -memb pid=159770，那么将生成类似于以下的结果:

```
All memory blocks in Private set.

Address            PoolID     PoolName     BlockAge   Size(Bytes) I LOC   File
0x0000000110469068 88         private      1          2488        1 172   4283993058
0x0000000110469A48 88         private      2          1608        1 172   4283993058
0x000000011046A0A8 88         private      3          4928        1 172   4283993058
0x000000011046B408 88         private      4          7336        1 172   4283993058
0x000000011046D0C8 88         private      5          32          1 172   4283993058
0x000000011046D108 88         private      6          6728        1 172   4283993058
0x000000011046EB68 88         private      7          168         1 172   4283993058
0x000000011046EC28 88         private      8          24          1 172   4283993058
0x000000011046EC68 88         private      9          408         1 172   4283993058
0x000000011046EE28 88         private      10         1072        1 172   4283993058
0x000000011046F288 88         private      11         3464        1 172   4283993058
0x0000000110470028 88         private      12         80          1 172   4283993058
0x00000001104700A8 88         private      13         480         1 1534  862348285
0x00000001104702A8 88         private      14         480         1 1939  862348285
0x0000000110499FA8 88         private      80         65551       1 1779  4231792244
Total set size: 94847 bytes

Memory blocks sorted by size:
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
88         private    65551                1          1779  4231792244
88         private    28336                12         172   4283993058
88         private    480                  1          1939  862348285
88         private    480                  1          1534  862348285
Total set size: 94847 bytes
```

示例 7: 确定哪个应用程序耗尽表空间

通过使用 db2pd -tcbstats，可以确定对表执行的插入操作数。以下是用户定义的全局临时表 TEMP1 的样本信息:

```
TCB Table Information:
Address           TbspaceID TableID PartID MasterTbs MasterTab TableName  SchemaNm ObjClass DataSize  LfSize LobSize XMLSize
0x0780000020B62AB0 3        2       n/a    3         2         TEMP1      SESSION  Temp     966       0      0       0
```

```
TCB Table Stats:
Address            TableName   Scans   UDI   PgReorgs   NoChgUpdts   Reads   FscrUpdates Inserts   Updates   Deletes   OvFlReads OvFlCrtes
0x0780000020B62AB0 TEMP1       0       0     0          0            0       0           43968     0         0         0         0
```

然后，可以通过使用 db2pd -tablespaces 命令获取表空间 3 的信息。样本输出如下所示:

```
Tablespace 3 Configuration:
Address            Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe  LastConsecPg Name
0x0780000020B1B5A0 DMS  UsrTmp  4096   32       Yes  32       1     1         On  1        0          31           TEMPSPACE2

Tablespace 3 Statistics:
Address            TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM        State      MinRecTime NQuiescers
0x0780000020B1B5A0 5000       4960       1088       0          3872       1088       0x00000000 0          0

Tablespace 3 Autoresize Statistics:
Address            AS  AR  InitSize    IncSize     IIP MaxSize     LastResize     LRF
0x0780000020B1B5A0 No  No  0           0           No  0           None           No

Containers:
Address            ContainNum Type    TotalPgs   UseablePgs StripeSet  Container
0x0780000020B1DCC0 0          File    5000       4960       0          /home/db2inst1/tempspace2a
```

FreePgs 列表明空间已耗尽。因为可用页数值下降，所以可用空间减少。另请注意，FreePgs 值加上 UsedPgs 值将等于 UsablePgs 值。

在了解这一点之后，可以通过运行 db2pd -db sample -dyn 来确定正在使用表 TEMP1 的动态 SQL 语句:

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:13:06

Dynamic Cache:
Current Memory Used           1022197
Total Heap Size               1271398
Cache Overflow Flag           0
Number of References          237
Number of Statement Inserts   32
Number of Statement Deletes   13
Number of Variation Inserts   21
Number of Statements          19

Dynamic SQL Statements:
Address            AnchID StmtUID  NumEnv  NumVar  NumRef  NumExe  Text
0x0000000220A08C40 78     1        2       2       3       2       declare global temporary table temp1 (c1 char(6)) not logged
0x0000000220A8D960 253    1        1       1       24      24      insert into session.temp1 values('TEST')
```

最后，可以通过运行 db2pd -db sample -app 将以上输出中的信息映射到应用程序输出，以便确定应用程序。

```
Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid Status
0x0000000200661840 501      [000-00501] 1          11246   UOW-Waiting

C-AnchID C-StmtUID  L-AnchID L-StmtUID  Appid
0        0          253      1          *LOCAL.db2inst1.050202160426
```

可以使用用于标识动态 SQL 语句的锚点标识（AnchID）值来标识相关联的应用程序。结果表明，最后一个锚点标识（L-AnchID）值与锚点标识（AnchID）值相同。一次运行 db2pd 产生的结果将在下一次运行 db2pd 时使用。

db2pd -agent 的输出将指示应用程序读取的行数（Rowsread 列）和写入的行数（Rowswrtn 列）。这些值将显示应用程序已完成的部分及尚未完成的部分，如以下样本输出所示:

```
Address            AppHandl [nod-index] AgentPid  Priority  Type DBName
0x0000000200698080 501      [000-00501] 11246     0         Coord SAMPLE

State       ClientPid  Userid   ClientNm Rowsread   Rowswrtn   LkTmOt
Inst-Active 26377      db2inst1 db2bp    22         9588       NotSet
```

可以将运行 db2pd -agent 命令时生成的 AppHandl 和 AgentPid 值映射到运行 db2pd -app 命令时生成的相应 AppHandl and CoorPiid 值。

如果您怀疑内部临时表占满了表空间，那么这些步骤会稍有不同。但是，仍然可以使用 db2pd -tcbstats 来标识具有最大插入数目的表。以下是隐式临时表的样本信息:

```
TCB Table Information:
Address            TbspaceID TableID PartID MasterTbs MasterTab TableName          SchemaNm ObjClass    DataSize  ...
0x0780000020CC0D30 1         2       n/a    1         2         TEMP (00001,00002) <30>     <JMC Temp   2470      ...
0x0780000020CC14B0 1         3       n/a    1         3         TEMP (00001,00003) <31>     <JMC Temp   2367      ...
0x0780000020CC21B0 1         4       n/a    1         4         TEMP (00001,00004) <30>     <JMC Temp   1872      ...

TCB Table Stats:
Address            TableName          Scans   UDI   PgReorgs   NoChgUpdts Reads   FscrUpdates Inserts ...
0x0780000020CC0D30 TEMP (00001,00002) 0       0     0          0          0       0           43219   ...
0x0780000020CC14B0 TEMP (00001,00003) 0       0     0          0          0       0           42485   ...
0x0780000020CC21B0 TEMP (00001,00004) 0       0     0          0          0       0           0       ...
```

在此示例中，使用命名约定 TEMP (TbspaceID, TableID) 的表中有大量插入。这些是隐式临时表。SchemaNm 列中的值的命名约定为 AppHandl 的值与 SchemaNm 的值并置，这使得它能够标识正在工作的应用程序。

然后，可以将这些信息映射至 db2pd -tablespaces 产生的输出，以查看表空间 1 的已使用空间。请记录以下输出中的表空间统计信息中 UsedPgs 值与 UsablePgs 值之间的关系:

```
Tablespace Configuration:
Address            Id    Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe LastConsecPg Name
0x07800000203FB5A0 1     SMS  SysTmp  4096   32       Yes  320      1     1         On  10       0         31           TEMPSPACE1

Tablespace Statistics:
Address            Id    TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM    State      MinRecTime NQuiescers
0x07800000203FB5A0 1     6516       6516       6516       0          0          0      0x00000000 0          0

Tablespace Autoresize Statistics:
Address            Id    AS  AR  InitSize    IncSize     IIP MaxSize     LastResize  LRF
0x07800000203FB5A0 1     No  No  0           0           No  0           None        No

Containers:
...
```

然后，可以使用命令 db2pd -app 标识应用程序句柄 30 和 31（因为它们出现在 -tcbstats 输出中）:

```
Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid    Status          C-AnchID C-StmtUID  L-AnchID   L-StmtUID  Appid
0x07800000006FB880 31       [000-00031] 1          4784182    UOW-Waiting     0        0          107        1          *LOCAL.db2inst1.051215214142
0x07800000006F9CE0 30       [000-00030] 1          8966270    UOW-Executing   107      1          107        1          *LOCAL.db2inst1.051215214013
```

最后，将以上输出中的信息映射到通过运行 db2pd -dyn 命令获取的动态 SQL 输出:

```
Dynamic SQL Statements:
Address             AnchID StmtUID    NumEnv     NumVar     NumRef     NumExe     Text
0x0780000020B296C0 107    1          1          1          43         43         select c1, c2 from test group by c1,c2
```

示例 8: 监视恢复

如果运行命令 db2pd -recovery，那么输出将显示多个计数器，这些计数器可用于验证是否正在执行恢复，如以下样本输出所示。当前日志和当前 LSN 值提供了日志位置。已完成的工作值是迄今为止完成的字节数。

```
Recovery:
Recovery Status     0x00000401
Current Log         S0000005.LOG
Current LSN         000002551BEA
Job Type            ROLLFORWARD RECOVERY
Job ID              7
Job Start Time      (1107380474) Wed Feb  2 16:41:14 2005
Job Description     Database Rollforward Recovery
Invoker Type        User
Total Phases        2
Current Phase       1

Progress:
Address            PhaseNum Description StartTime                CompletedWork  TotalWork
0x0000000200667160 1        Forward     Wed Feb  2 16:41:14 2005 2268098 bytes  Unknown
0x0000000200667258 2        Backward    NotStarted                0 bytes        Unknown
```

示例 9: 确定事务正在使用的资源量

如果运行命令 db2pd -transactions，那么输出将提供锁定数、第一个日志序号（LSN）、最后一个 LSN、已用日志空间量和保留空间量，如以下样本输出所示。这对于了解事务行为很有用。

```
Transactions:
Address            AppHandl [nod-index] TranHdl  Locks  State  Tflag
0x0000000022026D980 797      [000-00797] 2        108    WRITE  0x00000000
0x0000000022026E600 806      [000-00806] 3        157    WRITE  0x00000000
0x0000000022026F280 807      [000-00807] 4        90     WRITE  0x00000000

Tflag2     Firstlsn       Lastlsn        LogSpace   SpaceReserved
0x00000000 0x000001072262 0x0000010B2C8C 4518       95450
0x00000000 0x000001057574 0x0000010B3340 6576       139670
0x00000000 0x00000107CF0C 0x0000010B2FDE 3762       79266

TID            AxRegCnt   GXID
0x000000000451 1          0
0x0000000003E0 1          0
0x000000000472 1          0
```

示例 10: 监视日志使用情况

命令 db2pd -logs 对于监视数据库的日志使用情况很有用。通过使用以下样本输出中的写入页数值，可以确定日志使用量是否正在增加:

```
Logs:
Current Log Number            2
Pages Written                 846
Method 1 Archive Status       Success
Method 1 Next Log to Archive  2
Method 1 First Failure        n/a
Method 2 Archive Status       Success
Method 2 Next Log to Archive  2
Method 2 First Failure        n/a

Address            StartLSN       State      Size  Pages  Filename
0x0000000023001BF58 0x000001B58000 0x00000000 1000  1000   S0000002.LOG
0x0000000023001BE98 0x000001F40000 0x00000000 1000  1000   S0000003.LOG
0x0000000230008F58 0x000002328000 0x00000000 1000  1000   S0000004.LOG
```

通过使用此输出，可以标识两类问题:

- 如果最近大多数的日志归档操作都失败，那么归档状态的值将设置为失败。如果正在发生的归档失败导致完全无法对日志进行归档，那么归档状态值将设置为首次失败。
- 如果日志归档速度非常慢，那么下一个要归档的日志值将小于当前日志编号值。如果归档速度非常慢，那么可能表明活动日志的空间已耗尽，这将导致无法在数据库中进行任何数据更改。

示例 11: 查看综合系统（Sysplex）列表

如果不使用显示以下样本输出的 db2pd -sysplex 命令，那么报告综合系统（sysplex）列表的唯一方法是使用 DB2 跟踪。

```
Sysplex List:
Alias:         HOST
Location Name: HOST1
Count:         1

IP Address      Port       Priority   Connections Status     PRDID
 1.2.34.56      400        1          0           0
```

示例 12: 生成堆栈跟踪

可以使用 db2pd -stack all 命令（对于 Windows 操作系统）或 -stack 命令（对于 UNIX 操作系统）来生成当前数据库分区中所有进程的堆栈跟踪。如果您怀疑某个进程或线程正在循环或正被挂起，那么可能要反复使用此命令。

可以通过发出命令 db2pd -stack *eduid* 来获取特定引擎可分派单元（EDU）的当前调用堆栈，如以下示例所示:

```
Attempting to dump stack trace for eduid 137.
See current DIAGPATH for trapfile.
```

如果需要所有 DB2 进程的调用堆栈，请使用 db2pd -stack all 命令，例如，在 Windows 操作系统上:

```
Attempting to dump all stack traces for instance.
See current DIAGPATH for trapfiles.
```

如果您正在使用具有多个物理节点的分区数据库环境，那么通过使用命令 db2_all ″; db2pd -stack all″ 可以获取所有分区中的信息。但是，如果分区是同一机器上的所有逻辑分区，那么使用 db2pd -alldbp -stacks 时的速度会更快。

示例 13: 查看数据库分区的内存统计信息

db2pd -dbptnmem 命令显示 DB2 服务器当前消耗的内存量，并在较高级别显示使用这些内存的服务器区域。

以下是在 AIX 机器上运行 db2pd -dbptnmem 的输出示例:

```
Database Partition Memory Controller Statistics

Controller Automatic: Y
Memory Limit:   122931408 KB
Current usage:     651008 KB
HWM usage:         651008 KB
Cached memory:     231296 KB
```

这些数据字段和列的描述如下:

控制器自动
: 指示内存控制器设置。如果 **instance_memory** 配置参数设置为 AUTOMATIC，那么它将显示值“Y”。这意味着数据库管理器自动确定内存耗用量的上限。

内存限制
: 如果强制施加了实例内存限制，那么 **instance_memory** 配置参数的值是可以耗用的 DB2 服务器内存的上限。

当前使用量
: 服务器当前耗用的内存量。

HWM 使用量
: 自激活数据库分区（在 db2start 命令运行时）以来消耗的内存高水位标记（HWM）或峰值。

高速缓存的内存
: 当前使用量中未使用但为了提高将来内存请求的性能而高速缓存的内存量。

以下是在 AIX 操作系统上运行 db2pd -dbptnmem 的样本输出:

```
Individual Memory Consumers:
Name            Mem Used (KB)   HWM Used (KB)   Cached (KB)
========================================================
APPL-DBONE         160000          160000         159616
DBMS-name           38528           38528           3776
FMP_RESOURCES       22528           22528              0
PRIVATE             13120           13120            740
FCM_RESOURCES       10048           10048              0
LCL-p606416           128             128              0
DB-DBONE           406656          406656          67200
```

列示了 DB2 服务器中所有已注册的内存“使用者”以及它们消耗的内存总量。列描述如下:

名称 内存“使用者”的简短专有名称，例如:

APPL-*dbname*
为数据库 *dbname* 耗用的应用程序内存

DBMS-*name*
全局数据库管理器内存需求

FMP_RESOURCES
与 db2fmps 进行通信所需的内存

PRIVATE
其他专用内存需求

FCM_RESOURCES
快速通信管理器资源

LCL-*pid*
用于与本地应用程序进行通信的内存段

DB-*dbname*
为数据库 *dbname* 耗用的数据库内存

使用的内存（KB）
当前分配给使用者的内存量

使用的 HWM（KB）
使用者曾耗用的内存量的高水位标记（HWM），即内存量峰值

已高速缓存（KB）
在“使用的内存”中，当前未使用但立即可用于将来的内存分配的内存量

使用 db2support 命令来收集环境信息

对 DB2 问题收集信息时，您需要运行的最重要的 DB2 实用程序是 db2support。db2support 实用程序将自动收集所有可用的 DB2 诊断信息和系统诊断信息。它还有一个可选的交互式“问与答”会话，该会话会提出有关问题的详情。

使用 db2support 实用程序可以避免可能的用户错误，这是因为您不必手动输入 GET DATABASE CONFIGURATION FOR *database-name* 或 LIST TABLESPACES SHOW DETAIL 之类的命令。而且，您不需要有关要运行的命令或要收集的文件的指示信息，因此收集数据的速度会比较快。

- 执行 db2support -h 命令以生成完整的命令选项列表。
- 使用相应的 db2support 命令收集数据。

db2support 实用程序应该由具有 SYSADM 权限的用户（如实例所有者）运行，以便该实用程序可以收集所有必需的信息而不发生错误。如果没有 SYSADM 权限的用户运行 db2support，那么在实用程序运行 QUERY CLIENT 或 LIST ACTIVE DATABASES 之类的命令时，可能会产生 SQL 错误（SQL1092）。

如果使用 db2support 实用程序来帮助将信息传送至 IBM 软件支持机构，那么在系统遇到问题时运行 db2support命令。这样工具就会及时地收集信息，例如操作系统性能详细信息。如果在出现问题时无法运行实用程序，您仍可以在问题停止之后发出 db2support 命令，因为会自动生成某些首次出现数据捕获（FODC）诊断文件。

对于调试问题所需的大多数信息的收集而言，以下基本调用通常已经足够（注意，如果使用 -c 选项，那么该实用程序将建立与数据库的连接）：

```
db2support <output path> -d <database name> -c
```

输出的收集非常方便，并且会存储在压缩的 ZIP 归档 db2support.zip 中，以便可以很轻松地在任何系统上对其进行传送和解压缩。

db2support 捕获的信息类型取决于调用命令的方式、是否启动了数据库管理器以及能否连接至数据库。

db2support 实用程序在所有条件下收集以下信息:

- db2diag 日志文件
- 所有陷阱文件
- 锁定列表文件
- 转储文件
- 各种与系统有关的文件
- 各种系统命令的输出
- db2cli.ini

根据情况，db2support 实用程序还有可能收集以下信息:

- 活动日志文件
- 缓冲池和表空间（SQLSPCS.1 和 SQLSPCS.2）控制文件（使用 -d 选项）
- db2dump 目录的内容
- 扩展系统信息（使用 -s 选项）
- 数据库配置设置（使用 -d 选项）
- 数据库管理器配置设置文件
- 日志文件头文件（使用 -d 选项）
- 恢复历史记录文件（使用 -d 选项）

HTML 报告 db2support.html 将总是包括下列信息:

- 问题记录（PMR）编号（如果指定了 -n）
- 操作系统和级别（如 AIX 5.1）
- DB2 发行版信息
- 是 32 位还是 64 位环境的指示信息
- DB2 安装路径信息

- db2nodes.cfg 的内容
- CPU 和磁盘数目及内存量
- 实例上的数据库列表
- 注册表信息和环境，包括 PATH 和 LIBPATH
- UNIX 的当前文件系统和索引节点的磁盘可用空间
- Java™ SDK 级别
- 数据库管理器配置
- 数据库恢复历史记录文件列表
- sqllib 目录的 ls -lR 输出（或 Windows 上的相应项）
- LIST NODE DIRECTORY 命令的结果
- LIST ADMIN NODE DIRECTORY 命令的结果
- LIST DCS DIRECTORY 命令的结果
- LIST DCS APPLICATIONS EXTENDED 命令的结果
- 所有已安装软件的列表

指定 -s 选项时，以下信息将出现在 db2support.html 文件中:

- 详细的磁盘信息（分区布局、类型、LVM 信息等等）
- 详细的网络信息
- 内核统计信息
- 固件版本
- 其他特定于操作系统的命令

如果已经启动了 DB2，那么 db2support.html 文件包含下列附加信息:

- 客户机连接状态
- 数据库和数据库管理器配置（数据库配置需要 -d 选项）
- CLI 配置
- 内存池信息（大小和耗用大小）。如果使用 -d 选项，那么会收集完整数据
- LIST ACTIVE DATABASES 命令的结果
- LIST DCS APPLICATIONS 命令的结果

如果已指定 -c 选项，并且已成功连接至数据库，那么 db2support.html 文件包含以下信息:

- 用户表的数目
- 数据库数据的大概大小
- 数据库快照
- 应用程序快照
- 缓冲池信息
- LIST APPLICATIONS 命令的结果
- LIST COMMAND OPTIONS 命令的结果
- LIST DATABASE DIRECTORY 命令的结果
- LIST INDOUBT TRANSACTIONS 命令的结果
- LIST DATABASE PARTITION GROUPS 命令的结果

- LIST DBPARTITIONNUMS 命令的结果
- LIST ODBC DATA SOURCES 命令的结果
- LIST PACKAGES/TABLES 命令的结果
- LIST TABLESPACE CONTAINERS 命令的结果
- LIST TABLESPACES 命令的结果
- LIST DRDA IN DOUBT TRANSACTIONS 命令的结果
- DB2 工作负载管理器信息

db2support.zip 文件内容示例

为了获取 db2support.zip 文件的内容示例，已执行以下命令:

```
db2support . -d sample -c -f -st "select * from staff"
```

通过将 db2support.zip 文件解压缩，收集了下列文件和目录:

- DB2CONFIG/ – 配置信息（例如数据库、数据库管理器、BP、CLI、Java 开发者套件及其他）
- DB2DUMP/ – 过去三天生成的 db2diag.log 文件内容
- DB2MISC/ – sqllib 目录的列表
- DB2SNAP/ – DB2 命令的输出（例如，db2set、LIST TABLES、LIST INDOUBT TRANSACTIONS、LIST APPLICATIONS 及其他）
- db2supp_opt.zip – 优化器问题的诊断信息
- db2supp_system.zip – 操作系统信息
- db2support.html – 格式化为 HTML 节的诊断信息
- db2support_options.in – 用来启动 db2support 收集的命令行选项

db2supp_opt.zip 文件包含关于优化器的信息。将此文件解压缩将生成下列目录:

- OPTIMIZER/ – 优化器问题的诊断信息
- OPTIMIZER/optimizer.log – 此文件包含所有活动的日志
- OPTIMIZER/CATALOGS – 下列子目录中所有包含 LOB 的目录:
  - FUNCTIONS
  - ROUTINES
  - SEQUENCES
  - TABLES
  - VIEWS
- OPTIMIZER/DB2DUMP – db2serv 输出（serv.* 和 serv2.* 输出文件）

db2supp_system.zip 文件包含系统信息。将此文件解压缩将生成下列文件和目录:

- DB2CONFIG/ – db2cli.ini（~/sqllib/cfg 中的文件）
- DB2MISC/ – DB2SYSTM 文件（二进制）及其他
- OSCONFIG/ – 不同的操作系统信息文件（例如，netstat、services、vfs、ulimit、hosts 及其他）
- OSSNAP/ – 操作系统快照（例如，iostat、netstat、uptime、vmstat、ps_elf 及其他）
- SQLDBDIR/ – 重要的缓冲池元文件（~/sqllib/sqldbdir）

- SQLGWDIR/ - DCS 目录（~/sqllib/sqlgwdir 中的文件）
- SQLNODIR/ - 节点目录（~/sqllib/sqlnodir 中的文件）
- SPMLOG/ - ~/sqllib/spmlog 中的文件
- report.log - 所有收集活动的日志

验证 DB2 副本

db2val 命令确保 DB2 副本正常工作。

db2val 工具将通过验证安装文件、实例、数据库创建、与该数据库的连接以及 PDF 环境的状态，来验证 DB2 副本的核心功能。如果您使用 tar.gz 文件手动将 DB2 副本部署到 Linux 和 UNIX 操作系统上，那么此验证可能会有用。db2val 命令可以快速确保已正确进行了所有配置，并确保 DB2 副本为您所希望的副本。可以指定实例或数据库，也可以针对所有实例运行 db2val。可以在 *DB2 install path*\bin 目录和 sqllib/bin 目录中找到 db2val 命令。

例如，要验证 DB2 副本的所有实例，运行以下命令:

```
db2val -a
```

有关db2val 命令的完整详细信息以及更多示例，请参阅“db2val - DB2 副本验证工具命令”主题。

基本跟踪诊断

如果遇到与 DB2 相关的反复出现并且可再现的问题，那么执行跟踪有时能够捕获有关该问题的其他信息。在正常情况下，仅当 IBM 软件支持机构要求时，才应该使用跟踪。执行跟踪的过程包括设置跟踪工具、再现错误与收集数据。

跟踪收集的信息量增长得相当快。当您执行跟踪时，请尽可能地只捕获错误状态并避免任何其他活动。在执行跟踪时，请尽可能地使用最省力的方案来再现问题。

收集跟踪信息会对 DB2 实例的性能带来负面影响。性能的降低程度取决于问题的类型以及使用了多少资源来收集跟踪信息。

在请求跟踪时，IBM 软件支持机构应提供下列信息:

- 简单的逐步过程
- 进行每次跟踪的位置的说明
- 跟踪内容的说明
- 请求跟踪的理由的说明
- 回退过程（例如，如何禁用所有跟踪）

尽管您应该听从 IBM 软件支持机构有关获取哪些跟踪的建议，但以下仍然提供要求您获取特定跟踪时的一些一般准则:

- 如果安装期间发生问题，并且缺省安装日志不足以确定问题原因，那么比较适合进行安装跟踪。
- 如果在其中一个 GUI（图形用户界面）工具中发生问题，并且通过 DB2 命令窗口中的一些显式命令成功地执行了某些操作，那么比较适合进行控制中心跟踪。注意，这只会使用可从控制中心启动的工具捕获问题。

- 如果 CLI 应用程序中出现问题，并且该问题不能在该应用程序之外再现，那么比较适合进行 CLI 跟踪。
- 如果 JDBC 应用程序中出现问题，并且该问题不能在该应用程序之外再现，那么比较适合进行 JDBC 跟踪。
- 如果该问题与要在 DRDA 层进行通信的信息直接有关，那么进行 DRDA 跟踪比较适合。
- 对于可进行跟踪的所有其他情况，最适合进行 DB2 跟踪。

在诊断错误时，跟踪信息并不总是对您有帮助。例如，在下列情况下可能无法捕获错误情况:

- 您指定的跟踪缓冲区大小不够大，无法存放完整的跟踪事件集，当跟踪停止写入文件或合并时，丢失了有用的信息。
- 跟踪的方案没有重新创建错误状态。
- 重新创建了错误状态，但对问题发生位置的假定不正确。例如，跟踪是在客户机工作站上收集到的，但实际错误发生在服务器上。

DB2 跟踪

使用 db2trc 获取 DB2 跟踪

db2trc 命令控制随 DB2 提供的跟踪工具。跟踪工具记录有关操作的信息并将此信息格式化为可读格式。

记住，运行跟踪时会增加开销，所以启用跟踪工具可能会影响系统性能。

通常，IBM 软件支持机构和开发团队使用 DB2 跟踪进行故障诊断。您可以运行跟踪来获取有关正在调查的问题的信息，但如果您不了解 DB2 源代码，它的用途将十分有限。

即使只要求您获取跟踪文件，您也应该知道如何正确地启动跟踪以及如何转储跟踪文件。

注: 您需要 SYSADM、SYSCTRL 或 SYSMAINT 权限的其中一种权限来使用 db2trc。

要大致了解可用的选项，执行不带任何参数的 db2trc 命令:

```
C:\>db2trc
用法: db2trc (chg|clr|dmp|flw|fmt|inf|off|on) options
```

有关特定 db2trc 命令参数的更多信息，请使用 -u 选项。例如，要查看有关启动跟踪的更多信息，请执行以下命令:

```
db2trc on -u
```

这会提供有关可在启动 DB2 跟踪时指定的所有附加选项（标注为“工具”）的信息。

启用跟踪时，最重要的选项是 -L。它指定用于存储跟踪信息的内存缓冲区的大小。缓冲区大小可以字节或兆字节为单位来指定。要指定兆字节，那么在值后面追加“M”或“m”。跟踪缓冲区大小必须是 2 兆字节的幂。如果指定的大小不符合此要求，那么缓冲区大小将自动舍入为最接近的 2 的幂。

如果缓冲区太小，那么信息可能会丢失。在缺省情况下，如果缓冲区变满，那么只会保留最新的跟踪信息。如果缓冲太大，可能难以将文件发送至 IBM 软件支持团队。

如果跟踪时间相对较短的操作（如数据库连接），那么大概 8 MB 大小通常就已足够:

```
C:\> db2trc on -l 8M
已启动跟踪
```

但是，如果跟踪规模较大的操作或者同时在进行大量工作，那么可能需要较大的跟踪缓冲区。

在大多数平台上，跟踪可按如上所述随时打开并工作。但是，有些特定情况需要注意:

1. 在多数据库分区系统上，必须对每个物理（相对于逻辑）数据库分区运行跟踪。
2. 在 HP-UX 上，Linux 和 Solaris 平台上，如果在启动实例后关闭跟踪，那么不管指定大小如何，下一次启动跟踪时都会使用非常小的缓冲区。例如，昨天您通过使用 db2trc on -l 8m 启动了跟踪，然后收集了跟踪信息，然后停止了跟踪（db2trc off）。今天您希望在不关闭并重新启动实例的情况下，运行跟踪并将内存缓冲区设置为 32 MB（db2trc on -l 32m）。您会发现在此情况下，跟踪仅获得很小的缓冲区。为了在这些平台上有效地运行跟踪，应在启动实例前以所需大小缓冲区启动跟踪，并在以后必要时“清除”缓冲区。

转储 DB2 跟踪文件

使用 ON 选项启用跟踪工具后，将跟踪实例所作的所有后续工作。

跟踪运行时，可使用 clr 选项来清除跟踪缓冲区。将除去跟踪缓冲区中的所有现有信息。

```
C:\>db2trc clr
已清除跟踪
```

要跟踪的操作完成后，使用后跟跟踪文件名的 dmp 选项将内存缓冲池转储至磁盘。例如:

```
C:\>db2trc dmp trace.dmp
跟踪已转储至文件
```

跟踪缓冲区转储至磁盘后，跟踪工具将继续运行。要关闭跟踪，使用 OFF 选项:

```
C:\>db2trc off
已关闭跟踪
```

格式化 DB2 跟踪文件

db2trc dmp 命令创建的转储文件为二进制格式，并且不可读取。要验证是否可读取跟踪文件，请对二进制跟踪文件进行格式化以显示流量控制并将格式化输出发送至空设备。

下面的示例显示了用于执行此任务的命令:

```
db2trc flw example.trc nul
```

其中 example.trc 使用 dmp 选项生成的二进制文件。

此命令的输出将显式地指出读取文件是否有问题，以及跟踪是否已合并。

此时，可将转储文件发送至 IBM 软件支持机构。他们会根据您的 DB2 服务级别来对其进行格式化。但有时可能会要求您在发送转储文件之前将其转换为 ASCII 格式。这是通过 flw 和 fmt 选项完成的。必须提供二进制转储文件的名称及要创建的 ASCII 文件的名称:

```
C:\>db2trc flw trace.dmp trace.flw
C:\Temp>db2trc flw trace.dmp trace.flw
总跟踪记录数: 18854
截断跟踪: NO
合并跟踪: NO
格式化的跟踪记录数: 1513 (pid: 2196 tid 2148 节点: -1)
格式化的跟踪记录数: 100 (pid: 1568 tid 1304 节点: 0)
...

C:\>db2trc fmt trace.dmp trace.fmt
C:\Temp>db2trc fmt trace.dmp trace.fmt
截断跟踪: NO
合并跟踪: NO
总跟踪记录数: 18854
格式化的跟踪记录数: 18854
```

如果此输出指示“合并跟踪”为“YES”，那么表示跟踪缓冲区不够大，无法包含跟踪时间段收集的所有信息。根据情况，合并跟踪也许是可行的。如果您关心的是最新信息（除非指定了 -i 选项，否则这是缺省情况下保留的信息），那么跟踪文件中的内容可能已经足够。但是，如果您关心的是跟踪时间段开始时发生的情况，或者关心发生的所有情况，那么您可能想要使用更大的跟踪缓冲区重做该操作。

将二进制文件格式化为可读文本文件时，有一些选项可用。例如，可使用 `db2trc fmt -xml trace.dmp trace.fmt` 转换二进制数据并以 XML 可解析格式输出结果。其他选项显示在跟踪命令（db2trc）的详细描述中。

要注意的另一件事是: 在 Linux 和 UNIX 操作系统上，DB2 在因为严重错误而关闭实例时会自动将跟踪缓冲区转储至磁盘。因此，如果实例异常结束时启用了跟踪，那么会在诊断目录中创建一个文件，其名称为 `db2trdmp.###`，其中 `###` 是数据库分区号。Windows 平台上不会发生这种情况。在这些情况下，您必须手动转储跟踪。

总之，以下是 db2trc 命令的常用顺序示例:

```
db2trc on -l 8M
   db2trc clr
   <Execute problem recreation commands>
   db2trc dump db2trc.dmp
   db2trc off
   db2trc flw db2trc.dmp <filename>.flw
   db2trc fmt db2trc.dmp <filename>.fmt
   db2trc fmt -c db2trc.dmp <filename>.fmtc
```

DRDA 跟踪文件

在分析 DRDA 跟踪之前，必须了解 DRDA 是数据和通信结构的定义的开放式标准。例如，DRDA 包含有关数据传输的组织方式以及有关该信息的通信方式的一组规则。

这些规则是在以下参考手册中定义的:

- DRDA V3 Vol. 1: Distributed Relational Database Architecture™
- DRDA V3 Vol. 2: 格式化数据对象内容体系结构
- DRDA V3 Vol. 3: 分布式数据管理体系结构

这些手册的 PDF 版本可从 www.opengroup.org 获取。

db2drdat 实用程序记录 DRDA Application Requestor（AR）与 DB2 DRDA Application Server（AS）之间（如 DB2 Connect 与主机或 Power Systems® Servers 数据库服务器之间）的数据交换。

TRACE 实用程序

db2drdat 实用程序记录 DB2 Connect 服务器（代表 IBM 数据服务器客户机）与 IBM 大型机数据库服务器之间交换的数据。

作为数据库管理员（或应用程序开发者），您可能会发现了解此数据流如何工作是很有用的，因为这些知识可以帮助您确定特定问题的起源。假设您遇到以下情况：对 IBM 大型机数据库服务器发出了 CONNECT TO 数据库语句，但是该命令失败了并且您接收到不成功的返回码。如果准确了解到哪些信息被传送到 IBM 大型机数据库服务器管理系统，那么或许能够确定故障的原因，即使返回码信息是一般信息。很多故障是由简单的用户错误造成的。

db2drdat 的输出列示 DB2 Connect 工作站与 IBM 大型机数据库服务器管理系统之间交换的数据流。发送到 IBM 大型机数据库服务器的数据标记为 SEND BUFFER，而从 IBM 大型机数据库服务器接收到的数据标记为 RECEIVE BUFFER。

如果接收缓冲区中包含 SQLCA 信息，那么它将后接此数据的已格式化的解释并标记为 SQLCA。SQLCA 的 SQLCODE 字段是 IBM 大型机数据库服务器所返回的*未映射的*值。在文件中，发送缓冲区和接收缓冲区是按从最旧到最新的顺序来排列的。每个缓冲区都具有：

- 进程标识。
- SEND BUFFER、RECEIVE BUFFER 或 SQLCA 标号。缓冲区中的第一个 DDM 命令或对象被标记为 DSS TYPE。

在发送缓冲区和接收缓冲区中的其他数据被分成五列，包括：

- 字节数。
- 第 2 列和第 3 列表示在两个系统之间所交换的DRDA 数据流，采用 ASCII 或 EBCDIC 格式。
- 第 2 列和第 3 列的 ASCII 表示。
- 第 2 列和第 3 列的 EBCDIC 表示。

跟踪输出

db2drdat 实用程序将下列信息写入到*跟踪文件*中：

- -r
  - DRDA 应答/对象的类型
  - 接收缓冲区
- -s
  - DRDA 请求的类型
  - 发送缓冲区
- -c
  - SQLCA
- TCP/IP 错误信息
  - 接收函数返回码
  - 严重性
  - 使用的协议
  - 使用的 API

– 功能
– 错误号

注:

1. 出口码的值为零，指示成功完成了命令，非零值指示没有成功完成命令。
2. 返回的字段将随使用的 API 不同而不同。
3. 返回的字段将随运行 DB2 Connect 的平台不同而不同，即使对于同一 API 也是如此。
4. 如果 db2drdat 命令将输出发送至已存在的文件，那么将擦除旧文件，除非文件的许可权不允许擦除。

跟踪输出文件分析

在 db2drdat 跟踪中捕获到下列信息:

- 客户机应用程序的进程标识（PID）
- 在数据库连接服务（DCS）目录中编目的 RDB_NAME
- DB2 Connect CCSID
- IBM 大型机数据库服务器 CCSID
- DB2 Connect 系统正在与其通信的 IBM 大型机数据库服务器管理系统。

首个缓冲区中包含发送到 IBM 大型机数据库服务器管理系统的“交换服务器属性”（EXCSAT）和“访问 RDB”（ACCRDB）命令。它将这些命令作为 CONNECT TO 数据库命令的结果来发送。下一个缓冲区中包含 DB2 Connect 从 IBM 大型机数据库服务器管理系统中接收到的应答。它包含“交换服务器属性应答数据”（EXCSATRD）和“访问 RDB 应答消息”（ACCRDBRM）。

EXCSAT

EXCSAT 命令包含由“服务器名”（SRVNAM）对象指定的客户机的工作站名，按照 DDM 规范，它是代码点 X’116D’。EXCSAT 命令在第一个缓冲区中。在 EXCSAT 命令内，一旦除去了 X’116D’，值 X’9481A292’（按 CCSID 500 编码）就被转换为*掩码*。

EXCSAT 命令还包含 EXTNAM（外部名）对象，通常将该对象放在 IBM 大型机数据库管理系统上的诊断信息中。它由 20 个字节的应用程序标识、后接 8 个字节的进程标识（或者是 4 个字节的进程标识和 4 个字节的线程标识）组成。它由代码点 X’115E’ 表示，在此示例中，其值为 db2bp 并用空格填充，后接 000C50CC。在 Linux 或 UNIX IBM 数据服务器客户机 上，可以将此值与 ps 命令相关联，该命令将与活动进程有关的进程状态信息返回到标准输出中。

ACCRDB

ACCRDB 命令包含 RDBNAM 对象中的 RDB_NAME，它是代码点 X’2110’。在第一个缓冲区中，ACCRDB 命令跟在 EXCSAT 命令后面。在 ACCRDB 命令内，一旦除去了 X’2110’，值 X’E2E3D3C5C3F1’ 就会被转换为 STLEC1。这对应于 DCS 目录中的目标数据库名称字段。

记帐字符串具有代码点 X’2104’。

通过在 ACCRDB 命令中查找代码点为 X’119C’ 的 CCSID 对象 CCSIDSBC（用于单字节字符的 CCSID），就可显示为 DB2 Connect 工作站配置的代码集。在此示例中，CCSIDSBC 为 X’0333’，代码集为 819。

在 ACCRDB 命令中还存在附加对象 CCSIDDBC（双字节字符的 CCSID）和 CCSIDMBC（混合字节字符的 CCSID），它们的代码点分别为 X'119D' 和 X'119E' 。在此示例中，CCSIDDBC 为 X'04B0'，代码集为 1200；CCSIDMBC 为 X'0333'，代码集为 819。

EXCSATRD 和 ACCRDBRM

CCSID 值也是从 IBM 大型机数据库服务器的第二个缓冲区内的"访问 RDB 应答消息"（ACCRDBRM）中返回的。此缓冲区中包含 EXCSATRD，后接 ACCRDBRM。示例输出文件中包含 IBM 大型机数据库服务器系统的两个 CCSID 值。这两个值分别为 1208（对于单字节字符和混合字节字符）和 1200（对于双字节字符）。

如果 DB2 Connect 不识别从 IBM 大型机数据库服务器返回的代码页，那么将对用户返回 SQLCODE -332 以及源和目标代码页。如果 IBM 大型机数据库服务器不识别从 DB2 Connect 发送的代码集，那么它将返回 VALNSPRM（不受支持的参数值，DDM 代码点为 X'1252'），并对用户转换为 SQLCODE -332。

ACCRDBRM 还包含参数 PRDID（特定产品标识，代码点为 X'112E'）。值为 X'C4E2D5F0F8F0F1F5'，用 EBCDIC 表示为 DSN08015。按照标准，DSN 为 DB2 z/OS 版。还指示了版本号。ARI 是 DB2 服务器 VSE 版和 VM 版，SQL 是 DB2 数据库或 DB2 Connect，而 QSQ 则是 DB2 IBM i 版。

跟踪输出文件样本

下列各图显示了样本输出，说明在 DB2 Connect 工作站与主机或 System i 数据库服务器之间交换的一些 DRDA 数据流。从用户的角度来看，已经使用命令行处理器（CLP）发出了 `CONNECT TO` 数据库命令。

第 397 页的图 35 通过 TCP/IP 连接使用 DB2 Connect 企业版版本 9.1 和 DB2 z/OS 版版本 8。

```
1 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 0 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233

2 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 19532 probe 1177
 bytes 250

 SEND BUFFER(AR):

           EXCSAT RQSDSS                    (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  00C3D041000100BD 1041007F115E8482   ...A.....A...^..  .C}........".;db
    0010  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
    0020  4040F0F0F0C3F5F0 C3C3F0F0F0000000   @@..............    000C50CC000...
    0030  0000000000000000 0000000000000000   ................  ................
    0040  0000000000000000 000000000060F0F0   ............`..  ..............-00
    0050  F0F1A2A495404040 4040404040404040   .....@@@@@@@@@@@  01sun
    0060  4040404040404040 4040404040404040   @@@@@@@@@@@@@@@@
    0070  C4C5C3E5F8404040 F0A2A49540404040   .....@@@....@@@@  DECV8   0sun
    0080  4040404040404040 4000181404140300   @@@@@@@@@.......           .......
    0090  0724070008147400 05240F0008144000   .$....t..$....@.  .............. .
    00A0  08000E1147D8C4C2 F261C1C9E7F6F400   ....G....a......  .....QDB2/AIX64.
    00B0  08116D9481A29200 0C115AE2D8D3F0F9   ..m.......Z.....  .._mask...]SQL09
    00C0  F0F0F0                              ...               000

           ACCSEC RQSDSS                    (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0026D00100020020 106D000611A20003   .&..... .m......  ..}......_...s..
   0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
    0020  404040404040                        @@@@@@

3 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110546200 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 105

4 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110549755 probe 1178
 bytes 122

 RECEIVE BUFFER(AR):

           EXCSATRD OBJDSS                  (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0059D04300010053 1443000F115EE5F8   .Y.C...S.C...^..  ..}.........;V8
    0010  F1C14BE2E3D3C5C3 F100181404140300   ..K.............  1A.STLEC1.......
    0020  0724070007147400 05240F0007144000   .$....t..$....@.  .............. .
    0030  0700081147D8C4C2 F20014116DE2E3D3   ....G.......m...  .....QDB2..._STL
    0040  C5C3F14040404040 4040404040000C11   ...@@@@@@@@@@...  EC1          ...
    0050  5AC4E2D5F0F8F0F1 F5                 Z........         ]DSN08015

           ACCSECRD OBJDSS                  (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0010D0030002000A 14AC000611A20003   ................  ..}.........s..

5 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110656806 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233
```

图 35. 跟踪输出的示例（TCP/IP 连接）

```
6 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110659711 probe 1177
 bytes 250

 SEND BUFFER(AR):

           SECCHK RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  003CD04100010036 106E000611A20003   .<.A...6.n......  ..}......>...s..
    0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
    0020  4040404040040000C 11A1D9858799F485   @@@@@@..........        ....Regr4e
    0030  A599000A11A09585 A6A39695           ............      vr....newton
ACCRDB RQSDSS                   (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  00ADD001000200A7 20010006210F2407   ........ ...!.$.  ..}....x........
    0010  00172135C7F9F1C1 F0C4F3C14BD7C1F8   ..!5........K...  ....G91A0D3A.PA8
    0020  F806030221064600 162110E2E3D3C5C3   ....!.F..!......  8..........STLEC
    0030  F140404040404040 4040404040000C11   .@@@@@@@@@@@@...  1            ...
    0040  2EE2D8D3F0F9F0F0 F0000D002FD8E3C4   ............/...  .SQL09000....QTD
    0050  E2D8D3C1E2C30016 00350006119C0333   .........5.....3  SQLASC.........
    0060  0006119D04B00006 119E0333003C2104   ...........3.

7 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259908001 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 176

8 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259911584 probe 1178
 bytes 193

 RECEIVE BUFFER(AR):

           SECCHKRM RPYDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0015D0420001000F 1219000611490000   ...B.........I..  ..}.............
    0010  000511A400                         .....             ...u.
ACCRDBRM RPYDSS                 (ASCII)          (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  009BD00200020095 2201000611490000   ........"....I..  ..}....n........
    0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
    0020  2EC4E2D5F0F8F0F1 F500160035000611   ............5...  .DSN08015.......
    0030  9C04B80006119E04 B80006119D04B000   ................  ................
    0040  0C11A0D5C5E6E3D6 D540400006212524   .........@@..!%$  ...NEWTON  .....
    0050  34001E244E000624 4C00010014244D00   4..$N..$L....$M.  ....+...<.....(.
    0060  06244FFFFF000A11 E8091E768301BE00   .$O........v....  ..!.....Y...c...
    0070  2221030000000005 68B3B8C7F9F1C1F0   "!......h.......  ...........G91A0
    0080  C4F3C1D7C1F8F840 4040400603022106   .......@@@@...!.  D3APA88    .....
    0090  46000A11E8091E76 831389             F......v...       ....Y...c.i

9 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364420503 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
```

图 36. 跟踪输出的示例（TCP/IP 连接） - 续

```
10 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364440751 probe 1177
 bytes 27

 SEND BUFFER(AR):

           RDBCMM RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200E               ........ .        ..}.......

11 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475009631 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54

12 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475014579 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

           ENDUOWRM RPYDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
    0020  4040404040400005 211501             @@@@@@..!..             .....
SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  000BD00300010005 2408FF             ........$..       ..}........

13 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721710319 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 126

14 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721727276 probe 1177
 bytes 143

 SEND BUFFER(AR):

           EXCSQLIMM RQSDSS                 (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0053D0510001004D 200A00442113E2E3   .S.Q...M ..D!...  ..}....(......ST
    0010  D3C5C3F140404040 4040404040404040   ....@@@@@@@@@@@@  LEC1
    0020  D5E4D3D3C9C44040 4040404040404040   ......@@@@@@@@@@  NULLID
    0030  4040E2D8D3C3F2C6 F0C1404040404040   @@........@@@@@@    SQLC2F0A
    0040  4040404041414141 41484C5600CB0005   @@@@AAAAAHLV....      ......<.....
  0050  2105F1                               !..               ..1
SQLSTT OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  002BD00300010025 2414000000001B64   .+.....%$......d  ..}.............
    0010  656C657465206672 6F6D206464637375   elete from ddcsu  .%......?_......
    0020  73312E6D79746162 6C65FF             s1.mytable.       ..._`./.%..

15 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832901261 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 102
```

图 37. 跟踪输出的示例（TCP/IP 连接） - 续

```
16 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832906528 probe 1178
 bytes 119

 RECEIVE BUFFER(AR):

           SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  0066D00300010060 240800FFFFFF3434   .f.....`$.....44  ..}....-........
    0010  3237303444534E58 4F544C2000FFFFFE   2704DSNXOTL ....  ......+.!.<.....
    0020  0C00000000000000 00FFFFFFFF000000   ................  ................
    0030  0000000000572020 2057202020202020   .....W   W       ................
    0040  001053544C454331 2020202020202020   ..STLEC1          ....<...........
    0050  2020000F44444353 5553312E4D595441     ..DDCSUS1.MYTA  ............(...
    0060  424C450000FF                         BLE...            .<....
17 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833156953 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10

18 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833159843 probe 1177
 bytes 27

 SEND BUFFER(AR):

           RDBRLLBCK RQSDSS                 (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200F               ........ .        ..}.......
19 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943302832 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54

20 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943306288 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

           ENDUOWRM RPYDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
    0020  4040404040400005 211502             @@@@@@..!..             .....
SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
    0000  000BD00300010005 2408FF             ........$..       ..}........
```

图 38. 跟踪输出的示例（TCP/IP 连接） - 续

DRDA 跟踪的后续缓冲区信息

可以分析后续发送和接收缓冲区以获取附加信息。下一个请求包含一个落实。commit 命令指示 IBM 大型机数据库服务器管理系统落实当前工作单元。第四个缓冲区是作为落实或回滚的结果从 IBM 大型机数据库服务器数据库管理系统中接收到的。它包含“结束工作单元应答消息”（ENDUOWRM），它指示当前工作单元已经结束。

在此示例中，跟踪条目 12 包含一个空的 SQLCA，由后接 X'FF' 的 DDM 代码点 X'2408' 指示。空的 SQLCA（X'2408FF'）表示成功（SQLCODE 0）。

第 397 页的图 35 在跟踪条目 16 中显示了包含错误 SQLCA 的接收缓冲区的示例。

控制中心跟踪

尝试在控制中心中跟踪问题之前，建议先确保通过从 DB2 命令提示符处输入显式命令执行操作时未发生同样的问题。

通常当您在控制中心（或可从控制中心启动的一种其他的 GUI 工具）中执行任务时，您会看到"显示命令"按钮，它将提供该工具使用的命令的确切语法。如果从 DB2 命令提示符处成功执行了这一确切命令，但在 GUI 工具中执行该命令时失败了，那么应该获取控制中心跟踪。

为了获取只能在控制中心中再现的问题的跟踪，按如下所示启动控制中心:

```
db2cc -tf filename
```

这会打开控制中心跟踪并且将跟踪输出保存至指定的文件。输出文件将保存至 Windows 上的 `<DB2 install path>\sqllib\tools` 以及 UNIX 和 Linux 上的 `/home/<userid>/sqllib/tools`。

注: 启动控制中心并且启用跟踪后，使用尽可能少的步骤来再现问题。尽量避免在工具中单击不必要或不相关的项。再现问题后，关闭控制中心（以及为再现问题而打开的任何其他 GUI 工具）。

您必须将生成的跟踪文件发送给 IBM 软件支持机构以进行分析。

JDBC 跟踪文件

获取使用 DB2 JDBC 2 类驱动程序 Linux 版、UNIX 版和 Windows 版的应用程序的跟踪

本任务描述如何获取使用用于 Linux、UNIX 和 Windows 系统的 DB2 JDBC 2 类驱动程序的应用程序的跟踪。

此类型的跟踪适用于以下遇到问题的情况:

- 使用 DB2 JDBC 2 类驱动程序 Linux 版、UNIX 版和 Windows 版（DB2 JDBC 2 类驱动程序）的 JDBC 应用程序
- DB2 JDBC 存储过程

注: 有许多关键字可添加至 `db2cli.ini` 文件并影响应用程序行为。这些关键字可以解决应用程序问题，也可能是造成应用程序问题的原因。还有一些关键字在 CLI 文档中并未讨论。这些关键字只能从 DB2 支持机构处获取。如果 `db2cli.ini` 文件包含未讨论的关键字，那么它们可能是 DB2 支持机构建议使用的关键字。DB2 JDBC 2 类驱动程序在内部使用 DB2 CLI 驱动程序进行数据库访问。例如，DB2 JDBC 2 类驱动程序在内部将 Java getConnection() 方法映射至DB2 CLI SQLConnect() 函数。因此，Java 开发者可能会发现 DB2 CLI 跟踪是 DB2 JDBC 跟踪的一个很好的补充。

1. 为跟踪文件创建路径。 应该创建每个用户都有写入权限的路径。

   例如，在 Windows 上:

```
mkdir c:\temp\trace
```

在 Linux 和 UNIX 上:

```
mkdir /tmp/trace
          chmod 777 /tmp/trace
```

2. 更新 CLI 配置关键字。解决此冲突有两种方法:
   - 手动编辑 db2cli.ini 文件。根据是否使用 Microsoft® ODBC 驱动程序管理器、所使用的数据源名称（DSN）的类型、所安装的客户机或驱动程序的类型以及是否已设置注册表变量 **DB2CLIINIPATH**，db2cli.ini 文件的位置可能会有所变化。有关更多信息，请参阅*Call Level Interface Guide and Reference, Volume 1*中的『db2cli.ini 初始化文件』主题。
     a. 在纯文本编辑器中打开 db2cli.ini 文件。
     b. 将以下一段添加至该文件（如果 COMMON 段已存在，那么只追加变量）:

        ```
        [COMMON]
              JDBCTrace=1
              JDBCTracePathName=<path>
              JDBCTraceFlush=1
        ```

        例如，其中 <path> 在 Windows 上为 C:\temp\trace，在 Linux 或 UNIX 操作系统上为 /tmp/trace。
     c. 保存该文件时至少保存文件结尾的一个空白行。（这可以避免某些语法分析错误。）
   - 使用 UPDATE CLI CFG 命令更新 db2cli.ini 文件。发出下列命令:

     ```
     db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 1
     db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTracePathName <path>
     ```

     例如，其中 <path> 在 Windows 上为 C:\temp\trace，在 Linux 或 UNIX 操作系统上为 /tmp/trace。

     ```
     db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTraceFlush 1
     ```

   当您使用跟踪工具来诊断应用程序问题时，记住它对应用程序性能会有影响，并且会影响所有应用程序而不仅仅是测试程序。这就是一定要记住在标识问题后要关闭跟踪工具的原因。
3. 发出以下命令来验证是否设置并选择了正确的关键字:

   ```
   db2 GET CLI CFG FOR SECTION COMMON
   ```

4. 重新启动应用程序。

   仅当应用程序启动时才读取 db2cli.ini 文件，因此，要使任何更改生效，都必须重新启动应用程序。

   如果要跟踪 JDBC 存储过程，那么表示要重新启动 DB2 实例。
5. 捕获错误。 运行该应用程序直到生成错误，然后终止该应用程序。尽可能减少这种情况，在跟踪时只运行与问题再现有关的 JDBC 应用程序。这会使跟踪文件更加清晰明了。
6. 禁用 JDBC 跟踪。

   在 db2cli.ini 的 [COMMON] 段手动设置 JDBCTrace=0 关键字，或者发出:

   ```
   db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
         db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 0
   ```

7. 重新启动正在运行和跟踪的所有应用程序。
8. 收集跟踪文件。

   JDBC 跟踪文件将写至 JDBCTracePathName 关键字指定的路径。生成的文件名都会以 .trc 扩展名结尾。问题再现时在跟踪路径中生成的所有文件都是必需的。

获取使用 DB2 通用 JDBC 驱动程序的应用程序的跟踪

本任务描述对于使用 DB2 通用 JDBC 驱动程序的应用程序，如何获取其跟踪。

如果已经有正在使用 DB2 通用 JDBC 驱动程序的 SQLJ 或 JDBC 应用程序，那么可使用以下几种不同方法来启用 JDBC 跟踪:

- 如果使用 DataSource 接口连接至数据源，那么使用 DataSource.setTraceLevel() 和 DataSource.setTraceFile() 方法来启用跟踪。
- 如果使用 DriverManager 接口连接至数据源，那么启用跟踪的最简单方法是获取连接之前在 DriverManager 上设置 logWriter。

  例如:

  ```
  DriverManager.setLogWriter(new PrintWriter(new FileOutputStream("trace.txt")));
  ```

- 如果在使用 DriverManager 接口，在装入驱动程序时还可以在 URL 中指定 traceFile 和 traceLevel 属性。

  例如:

  ```
  String databaseURL =
  "jdbc:db2://hal:50000/sample:traceFile=c:/temp/foobar.txt;" ;
  ```

CLI 跟踪文件

CLI 跟踪捕获有关访问 DB2 CLI 驱动程序的应用程序的信息。CLI 跟踪基本上不提供有关 DB2 CLI 驱动程序的内部工作情况的信息。

此类型的跟踪适用于以下遇到问题的情况:

- CLI 应用程序
- ODBC 应用程序（因为 ODBC 应用程序使用 DB2 CLI 接口来访问 DB2）
- DB2 CLI 存储过程
- JDBC 应用程序和存储过程

在诊断 ODBC 应用程序时，通常最容易确定问题的方法就是使用 ODBC 跟踪或 DB2 CLI 跟踪。如果在使用 ODBC 驱动程序管理器，那么可以进行 ODBC 跟踪。查阅驱动程序管理器文档以确定启用 ODBC 跟踪的方式。DB2 CLI 跟踪是特定于 DB2 的，并且通常包含的信息比一般 ODBC 跟踪要多。这两种跟踪通常极为相似，都会列示某个应用程序中所有 CLI 调用的入口点和出口点；包括这些调用的所有参数和返回码。

DB2 JDBC 2 类驱动程序 Linux 版、UNIX 版和 Windows 版（DB2 JDBC 2 类驱动程序）取决于访问数据库的 DB2 CLI 驱动程序。因此，Java 开发者可能还想启用 DB2 CLI 跟踪，以获取有关应用程序如何通过各个软件层与数据库交互的其他信息。尽管都是在 db2cli.ini 文件中设置的，但 DB2 JDBC 和 DB2 CLI 跟踪选项相互独立。

获取 CLI 跟踪

要打开 CLI 跟踪功能，必须启用一组 CLI 配置关键字。

开始前

注: 有许多关键字可添加至 db2cli.ini 文件并影响应用程序行为。这些关键字可以解决应用程序问题，也可能是造成应用程序问题的原因。还有一些关键字在 CLI 文档中并未讨论。这些关键字只能从 IBM 软件支持机构处获取。如果 db2cli.ini 文件包含未讨论的关键字，那么它们可能是 IBM 软件支持团队建议使用的关键字。

关于此任务

当您使用跟踪工具来诊断应用程序问题时，记住它对应用程序性能会有影响，并且会影响所有应用程序而不仅仅是测试程序。这就是一定要记住在标识问题后要关闭跟踪工具的原因。

过程

要获取 CLI 跟踪，请完成下列步骤:

1. 为跟踪文件创建路径。

   应该创建每个用户都有写入权限的路径。 例如，在 Windows 操作系统上:

   ```
   mkdir c:\temp\trace
   ```

   在 Linux 和 UNIX 操作系统上:

   ```
   mkdir /tmp/trace
   chmod 777 /tmp/trace
   ```

2. 更新 CLI 配置关键字。

   可通过手动编辑 db2cli.ini 文件或使用 UPDATE CLI CFG 命令来执行此操作。

   - 要手动编辑 db2cli.ini 文件:
     a. 在纯文本编辑器中打开 db2cli.ini 文件。根据是否使用 Microsoft ODBC 驱动程序管理器、所使用的数据源名称（DSN）的类型、所安装的客户机或驱动程序的类型以及是否已设置注册表变量 **DB2CLIINIPATH**，db2cli.ini 文件的位置可能会有所变化。有关更多信息，请参阅*Call Level Interface Guide and Reference, Volume 1*中的『db2cli.ini 初始化文件』主题。
     b. 将以下一段添加至该文件（或者如果 COMMON 段已存在，那么只追加变量）:

        ```
        [COMMON]
        Trace=1
        TracePathName=path
        TraceComm=1
        TraceFlush=1
        TraceTimeStamp=1
        ```

        例如，其中 *path* 在 Windows 上为 C:\temp\trace，在 Linux 或 UNIX 上为 /tmp/trace。
     c. 保存该文件时至少保存文件结尾的一个空白行。（这可以避免某些语法分析错误。）
   - 要使用 UPDATE CLI CFG 命令来更新 CLI 配置关键字，请发出下列命令:

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName path
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3
```

例如，其中 *path* 在 Windows 上为 `C:\temp\trace`，在 Linux 或 UNIX 上为 `/tmp/trace`。

3. 确认 `db2cli.ini` 配置。

   发出以下命令来验证是否设置并选择了正确的关键字:

   ```
   db2 GET CLI CFG FOR SECTION COMMON
   ```

4. 重新启动应用程序。

   仅当应用程序启动时才读取 `db2cli.ini` 文件，因此，要使任何更改生效，都必须重新启动应用程序。

   如果要跟踪 CLI 存储过程，那么表示要重新启动 DB2 实例。

5. 捕获错误。

   运行该应用程序直到生成错误，然后终止该应用程序。如果可以减少这种情况，在跟踪时只运行与问题再现有关的应用程序，那么跟踪分析会清晰得多。

6. 禁用 CLI 跟踪。

   在 `db2cli.ini` 的 `[COMMON]` 节中，手动地将 **Trace** 关键字设置为值 0，或者发出以下命令:

   ```
   db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
   ```

7. （可选）重新启动任何可能正在运行和跟踪的应用程序。

结果

CLI 跟踪文件将写至 **TracePathName** 关键字指定的路径。文件名的格式为 p*pid*t*tid*.cli，其中 *pid* 是操作系统指定的进程标识，而 *tid* 是应用程序进程生成的每个线程的数字计数器（从 0 开始）。例如，`p1234t1.cli`。如果您正在与 IBM 软件支持机构配合诊断问题，他们会想要查看跟踪路径中生成的所有文件。

解释 CLI 跟踪文件中的输入和输出参数

与任何常规函数一样，DB2 CLI 函数也有输入和输出参数。在 DB2 CLI 跟踪中，可以看到这些输入和输出参数，它们提供有关每个应用程序如何调用特定 CLI API 的大量详细信息。CLI 跟踪中显示的任何 CLI 函数的输入和输出参数都可与文档的“CLI 参考”部分中的 CLI 函数的定义进行比较。

以下是 CLI 跟踪文件中的某个片段:

```
SQLConnect( hDbc=0:1, szDSN="sample", cbDSN=-3, szUID="",
                      cbUID=-3, szAuthStr="", cbAuthStr=-3 )
        ---> Time elapsed - +6.960000E-004 seconds

SQLRETURN   SQLConnect       (
            SQLHDBC           ConnectionHandle,  /* hdbc */
            SQLCHAR      *FAR ServerName,        /* szDSN */
            SQLSMALLINT       NameLength1,       /* cbDSN */
            SQLCHAR      *FAR UserName,          /* szUID */
```

```
            SQLSMALLINT       NameLength2,       /* cbUID */
            SQLCHAR      *FAR Authentication,    /* szAuthStr */
            SQLSMALLINT       NameLength3);      /* cbAuthStr */
```

CLI 函数的初始调用显示输入参数和指定给它们的值（如果适当）。

在 CLI 函数返回时，它们将显示作为结果的输出参数，例如:

```
SQLAllocStmt( phStmt=1:1 )
    <--- SQL_SUCCESS   Time elapsed - +4.444000E-003 seconds
```

在此情况下，CLI 函数 SQLAllocStmt() 将返回输出参数 phStmt，其值为“1:1”（连接句柄 1，语句句柄 1）。

分析 CLI 跟踪中的动态 SQL

DB2 CLI 跟踪还会显示如何通过声明并在 SQLPrepare() 和 SQLBindParameter() 中使用参数标记来执行动态 SQL。这使得您能够确定运行时执行的 SQL 语句。

以下跟踪条目显示 SQL 语句的准备过程（问号 ? 或者后跟名称的冒号（:*name*）表示参数标记）:

```
 SQLPrepare( hStmt=1:1, pszSqlStr=
   "select * from employee where empno = ?",
   cbSqlStr=-3 )
    ---> Time elapsed - +1.648000E-003 seconds
( StmtOut="select * from employee where empno = ?" )
SQLPrepare( )
  <--- SQL_SUCCESS   Time elapsed - +5.929000E-003 seconds
```

以下跟踪条目显示将参数标记绑定为最大长度为 7 的 CHAR:

```
SQLBindParameter( hStmt=1:1, iPar=1, fParamType=SQL_PARAM_INPUT,
fCType=SQL_C_CHAR, fSQLType=SQL_CHAR, cbColDef=7, ibScale=0,
 rgbValue=&00854f28, cbValueMax=7, pcbValue=&00858534 )
    ---> Time elapsed - +1.348000E-003 seconds
SQLBindParameter( )
    <--- SQL_SUCCESS   Time elapsed - +7.607000E-003 seconds
```

这就执行了动态 SQL 语句。rbgValue=″000010″ 显示应用程序在运行时用来取代参数标记的值:

```
SQLExecute( hStmt=1:1 )
    ---> Time elapsed - +1.317000E-003 seconds
( iPar=1, fCType=SQL_C_CHAR, rgbValue="000010" - X"303030303130",
 pcbValue=6, piIndicatorPtr=6 )
    sqlccsend( ulBytes - 384 )
    sqlccsend( Handle - 14437216 )
    sqlccsend( ) - rc - 0, time elapsed - +1.915000E-003
    sqlccrecv( )
    sqlccrecv( ulBytes - 1053 ) - rc - 0, time elapsed - +8.808000E-003
SQLExecute( )
    <--- SQL_SUCCESS   Time elapsed - +2.213300E-002 seconds
```

解释 CLI 跟踪中的计时信息

有几种方法可用来从 DB2 CLI 跟踪中收集计时信息。在缺省情况下，CLI 跟踪会捕获应用程序中自上次对特定线程执行 CLP API 调用后所花的时间。

它包括客户机与服务器之间的网络时间以及 DB2 中所花的时间。例如:

```
SQLAllocStmt( hDbc=0:1, phStmt=&0012ee48 )
        ---> Time elapsed - +3.964187E+000 seconds
```

（此时间值指示应用程序中自上次调用 CLI API 后所花的时间）

```
SQLAllocStmt( phStmt=1:1 )
        <--- SQL_SUCCESS   Time elapsed - +4.444000E-003 seconds
```

（自函数完成后，此时间值指示 DB2 中所花的时间，包括网络时间）

捕获计时信息的另一个方法是使用 CLI 关键字 TraceTimeStamp。此关键字将对每次调用和 DB2 CLI API 调用的结果生成时间戳记。关键字有 4 个显示选项：没有时间戳记信息、处理器标记和 ISO 时间戳记、处理器标记或者 ISO 时间戳记。

在处理与计时有关的问题（如 CLI0125E - 函数顺序错误）时，它会非常有用。如果要在使用多线程应用程序时尝试确定最先发生的事件，它也会非常有用。

解释 CLI 跟踪中的未知值

DB2 CLI 函数可返回“未知值”作为 CLI 跟踪中的输入参数。

如果 DB2 CLI 驱动程序要查找特定于该输入参数的内容，而应用程序提供了另一个值，那么可能发生这种情况。例如，如果遵循 CLI 函数的过时定义或使用已经不推荐使用的 CLI 函数，那么可能会发生这种情况。

您还可能会看到 CLI 函数调用返回“选项值已更改”或“键集解析器返回码”。这是显示消息的键集游标的结果，如游标因为某些特定原因降级为静态游标。

```
SQLExecDirect( hStmt=1:1, pszSqlStr="select * from org", cbSqlStr=-3 )
        ---> Time elapsed - +5.000000E-002 seconds
( StmtOut="select * from org" )
( COMMIT=0 )
( StmtOut=" SELECT A.TABSCHEMA, ..... )
( StmtOut=" SELECT A.TABSCHEMA, ..... )
( Keyset Parser Return Code=1100 )

SQLExecDirect( )
        <--- SQL_SUCCESS_WITH_INFO   Time elapsed - +1.06E+001 seconds
```

在上述 CLI 跟踪中，键集解析器指示返回码为 1100，这表示该表没有唯一索引或主键，因此未能创建键集游标。这些返回码未外部化，所以此时如果您想要有关返回码的含义的更多信息，那么必须与 IBM 软件支持机构联系。

调用 SQLError 或 SQLDiagRec 将指示游标类型已更改。那么，该应用程序就应该查询游标类型和并行性以确定更改的属性。

解释多线程 CLI 跟踪输出

CLI 跟踪可跟踪多线程应用程序。跟踪多线程应用程序的最好方法是使用 CLI 关键字 TracePathName。这将生成跟踪文件 p<pid>t<tid>.cli，其中 <tid> 是应用程序的实际线程标识。

如果必须知道实际线程标识，可在 CLI 跟踪头中看到此信息:

```
[ Process: 3500, Thread: 728 ]
[ Date & Time:          02/17/2006 04:28:02.238015 ]
[ Product:              QDB2/NT DB2 v9.1.0.190 ]
...
```

还可使用 CLI 关键字 TraceFileName 将多线程应用程序跟踪写至一个文件。此方法将生成一个您选择的文件，但阅读此文件会很困难，这是因为在执行一个线程中的特定 API 的同时会执行另一个线程中的另一个 API，因此可能会导致复查跟踪时造成混乱。

通常建议将 TraceTimeStamp 设置为 ON 以便您可以通过查看执行特定 API 的时间来确定事件的真实顺序。对于调查一个线程在另一个线程中导致问题的情况，这会非常有用（如 CLI0125E - 函数顺序错误）。

特定于平台的工具

诊断工具（Windows）

本节描述三个用于 Windows 系统的有用诊断工具。

下列诊断工具可用于 Windows 操作系统:

事件查看器、性能监视器和其他管理工具

"管理工具"文件夹提供了各种诊断信息，包括访问事件日志和访问性能信息。

任务管理器

"任务管理器"显示正在 Windows 服务器上运行的所有进程以及有关内存使用情况的详细信息。使用此工具来查明哪些 DB2 进程正在运行并诊断性能问题。通过使用此工具，可以确定进程的内存使用情况、内存限制、已使用的交换文件空间以及内存泄漏。

要打开"任务管理器"，按 Ctrl + Alt + Delete，并从可用选项中单击**任务管理器**。

Dr. Watson

Dr. Watson 实用程序是在"常规故障保护"（GPF）事件中调用的。它记录可能有助于诊断问题的数据，并将此信息保存至文件。必须通过在命令行上输入 `drwatson` 以启动此实用程序。

诊断工具（Linux 和 UNIX）

本节描述有关在 Linux 和 UNIX 平台上进行故障诊断和性能监视的一些基本命令。

有关其中任何一个命令的详细信息，请在命令行上输入它并在前面加上"man"。使用这些命令来收集和处理数据，这些数据有助于确定系统问题的原因。在收集数据之后，可以由熟悉问题的人员对其进行检查，也可以将其提供给 IBM 软件支持机构。

故障诊断命令（AIX）

下列 AIX 系统命令对于 DB2 故障诊断很有用:

errpt errpt 命令报告系统错误，如硬件错误和网络故障。

- 要获取每个错误显示一行的概述，使用 errpt
- 要获取每个错误显示一页的更为详细的视图，使用 errpt -a
- 要获取错误号为"1581762B"的错误，使用 errpt -a -j 1581762B
- 要查明过去是否用光了调页空间，使用 errpt | grep SYSVMM
- 要查明是否存在令牌环卡或磁盘问题，检查 errpt 输出中的短语"disk"和"tr0"

lsps lsps -a 命令监视并显示调页空间的使用情况。

lsattr 此命令显示各种操作系统参数。例如，使用以下命令来查明数据库分区上的实内存量:

```
lsattr -l sys0 -E
```

xmperf

对于使用 Motif 的 AIX 系统，此命令启动用于收集和显示与系统相关的性能数据的图形监视器。对于每个数据库分区，监视器都会在单个窗口中显示三维图，这对于高级监视有好处。但是，如果活动很低，那么此监视器的输出将是有限值。

spmon

如果使用系统分区作为 Parallel System Support Programs（PSSP）的一部分，那么可能需要检查 SP 交换机是否在所有工作站上运行。要查看所有数据库分区的状态，从控制工作站使用下列其中一个命令:

- spmon -d，用于 ASCII 输出
- spmon -g，用于图形用户界面

或者，从数据库分区工作站使用命令 netstat -i 来查看是否关闭了该交换机。如果已关闭高性能交换机，那么数据库分区名旁边会有一个星号（*）。例如:

```
css0* 65520 <Link>0.0.0.0.0.0
```

如果启动了高性能交换机，那么不会出现星号。

故障诊断命令（Linux 和 UNIX）

下列系统命令用于所有 Linux 和 UNIX 系统（包括 AIX），除非另行说明。

df df 命令让您查看文件系统是否已满。

- 要查看所有文件系统（包括已安装的文件系统）中有多少可用空间，使用 df
- 要查看名称包含“dev”的所有文件系统中有多少可用空间，使用 df | grep dev
- 要查看 home 文件系统中有多少可用空间，使用 df /home
- 要查看文件系统“tmp”中有多少可用空间，使用 df /tmp
- 要查看机器上是否有足够的可用空间，检查下列命令的输出：df /usr、df /var、df /tmp 和 df /home

truss 此命令对于跟踪一个或多个进程中的系统调用很有用。

pstack

可用于 Solaris 2.5.1 或更新版本，/usr/proc/bin/pstack 命令显示堆栈回溯信息。/usr/proc/bin 目录包含用于调试可能处于暂挂状态的进程的其他工具。

性能监视工具

下列工具可用于监视系统的性能。

vmstat

此命令对于确定某个对象是处于暂挂状态还是在长时间运行是很有用的。您可以监视在页面调进（pi）和页面调出（po）列中找到的页面调度速率。其他很重要的列是已分配虚拟存储器量（avm）和可用虚拟存储量（fre）。

iostat 此命令对于监视 I/O 活动很有用。您可以使用读写速率来估计某些 SQL 操作所需的时间量（如果它们是系统上唯一的活动）。

netstat

此命令让您了解每个数据库分区上的网络流量以及遇到的错误信息包的数目。它对于隔离网络问题是很有用的。

system file

可用于 Solaris 操作系统，`/etc/system` 文件包含内核配置限制的定义，例如系统上允许同时存在的最大用户数、每个用户的最大进程数以及有关资源大小和资源量的进程间通信（IPC）限制。这些限制非常重要，因为它们会影响 Solaris 操作系统机器上的 DB2 性能。

第 6 章 DB2 数据库故障诊断

通常，故障诊断过程要求您先找出并确定问题，然后再寻求解决方法。本节将提供关于 DB2 产品特定功能的故障诊断信息。

对于已知常见问题，找出的解决方法在本节以核对表的形式提供。如果您无法在该核对表中找到解决方法，那么可以自己收集其他诊断数据并进行分析，或者将该数据提交给 IBM 软件支持机构进行分析。

下列问题将引导您执行相应的故障诊断任务:

1. 是否已经应用了所有已知修订包？如果尚未应用，请考虑《*安装 DB2 服务器*》中的“应用修订包”。
2. 是否在以下情况下出现问题:
   - 安装 DB2 数据库服务器或客户机时？如果是，请参阅本书其他位置处的“收集有关安装问题的数据”主题。
   - 创建、删除、更新或升级实例或 DB2 管理服务器（DAS）时？如果是，请参阅本书其他位置处的“收集有关 DAS 和实例管理问题的数据”主题。
   - 使用 EXPORT、IMPORT、LOAD 或 db2move 命令移动数据时？如果是，请参阅本书其他位置处的“收集有关数据移动问题的数据”主题。

如果您的问题不属于上面任一类别并且要与 IBM 软件支持机构联系，那么可能仍需要基本的诊断数据。您必须。

收集 DB2 数据

有时只对症状进行故障诊断不能解决问题。在这种情况下，必须收集诊断数据。必须收集的诊断数据以及所收集的数据的来源取决于正在调查的问题类型。这些步骤表示如何收集在将问题提交给 IBM 软件支持机构时通常必须提供的基本信息集。

要获取最完整的输出，实例所有者应调用 db2support 实用程序。

要收集基本诊断信息集并将其存储在压缩的文件归档中，请输入 db2support 命令:

```
db2support <output_directory> -s -d <database_name> -c
```

使用 -s 会给出有关使用的硬件和操作系统的系统详细信息。使用 -d 会给出有关指定数据库的详细信息。使用 -c 允许尝试连接至指定数据库。
输出的收集非常方便，并且会存储在压缩的 ZIP 归档 db2support.zip 中，以便可以很轻松地在任何系统上对其进行传送和解压缩。

对于特定症状或产品的特定部分中的问题，可能需要收集其他数据。请参阅特定于问题的“收集数据”文档。

接下来，您可以执行下列任何任务:

- 分析数据
- 将数据提交给 IBM 软件支持机构

© Copyright IBM Corp. 2006, 2009

收集关于数据移动问题的数据

如果您在执行数据移动命令时遇到问题并且不能确定问题的原因，请收集您或 IBM 软件支持机构可用来诊断和解决问题的诊断数据。

请按照以下列表中适合于您所遇到的情况的数据收集指示信息执行操作:

- 要收集与 db2move 命令相关的问题的数据，请转至发出了此命令的目录。根据在此命令中指定的操作找到下列文件:
  - 对于 COPY 操作，查找名为 COPY.*timestamp*.ERR 和 COPYSCHEMA.*timestamp*.MSG 的文件。如果还指定了 LOAD_ONLY 或 DDL_AND_LOAD 方式，还需查找名为 LOADTABLE.*timestamp*.MSG 的文件。
  - 对于 EXPORT 操作，查找名为 EXPORT.out 的文件。
  - 对于 IMPORT 操作，查找名为 IMPORT.out 的文件。
  - 对于 LOAD 操作，查找名为 LOAD.out 的文件。
- 要收集与 EXPORT、IMPORT 或 LOAD 命令相关的问题的数据，确定命令是否包括 MESSAGES 参数。如果包括此参数，那么收集输出文件。如果您不指定其他目录和驱动器，那么这些实用程序使用当前目录和缺省驱动器作为目标。
- 要收集与 REDISTRIBUTE 命令相关的问题的数据，请查找称为“*databasename.database_partition_groupname*”的文件。 Linux 和 UNIX 上，查找名为“*databasename.database_partition_groupname.timestamp*”的文件；在 Windows 上，查找名为“*databasename.database_partition_groupname.date.time*”的文件。此文件分别在 *$HOME*/sqllib/db2dump 目录或 $DB2PATH\sqllib\redist 目录中，其中 *$HOME* 是实例所有者的主目录。

收集关于 DAS 和实例管理问题的数据

如果您在执行 DB2 管理服务器（DAS）或实例管理时遇到问题并且不能确定问题的原因，请收集您或 IBM 软件支持机构可用来诊断和解决问题的诊断数据。

这些步骤仅适用于您在 Linux 或 UNIX 上使用 DB2 时重现问题的情况。

要收集关于 DAS 或实例管理问题的诊断数据:

1. 在启用跟踪或调试方式的情况下重复失败的命令。示例命令:

```
db2setup -t trace.out
dascrt -u DASUSER -d
dasdrop -d
dasmigr -d
dasupdt -d
db2icrt -d INSTNAME
db2idrop INSTNAME -d
db2iupgrade -d INSTNAME
db2iupdt -d INSTNAME
```

2. 找到诊断文件。可能会有多个文件，请比较时间戳记以确保获取所有合适的文件。

   缺省情况下，输出内容将出现在 /tmp 目录中。

   示例文件名为: dascrt.log、dasdrop.log、 dasupdt.log、db2icrt.log.PID、db2idrop.log.PID、db2iupgrade.log.PID 和 db2iupdt.log.PID，其中，PID 是进程标识。

3. 将诊断文件提供给 IBM 软件支持机构。

如果是因为 db2start 或 START DATABASE MANAGER 命令失败而导致出现问题，请在 insthome/sqllib/log 目录（其中 insthome 是实例所有者的主目录）中查找名为 db2start.*timestamp*.log 的文件。同样，如果是因为 db2stop 或 STOP DATABASE MANAGER 命令失败而导致出现问题，那么查找名为 db2stop.*timestamp*.log 的文件。仅当数据库管理器未在 **start_stop_time** 数据库管理器配置参数所指定的时间内响应命令时，才会找到这些文件。

分析 DB2 数据

在收集数据后，您必须确定这些数据如何帮助您解决特定问题。分析类型取决于正在调查的问题类型和已收集的数据。这些步骤表示如何开始调查任何基本 DB2 诊断数据。

要分析诊断数据，请执行下列操作:

- 弄清楚各个数据相互之间的关系。例如，如果数据在多个系统上，那么有条理地组织好数据，以便知道每个数据的来源。
- 通过检查时间戳记确认每个诊断数据与问题的计时有关。注意，来自不同源的数据可能采用不同的时间戳记格式；一定要了解每种时间戳记格式中不同元素的序列，以便知道不同事件的发生时间。
- 确定最有可能包含与问题有关的信息的数据源，并从那里开始分析。例如，如果问题与安装有关，那么从安装日志文件（如果存在）开始分析，而不是从一般产品或操作系统日志文件开始分析。
- 每个数据源的特定分析方法都是唯一的，但大多数跟踪文件和日志文件都适用的一个技巧是，一开始就要确定数据中出现问题的时间点。在确定此时间点后，您可以从此时间点往前分析数据，以便弄清楚问题的根本原因。
- 如果您正在调查一个问题，并且拥有两个环境中几乎一样多的数据（其中一个环境正在运行，而另一个环境未运行），那么应从比较每个环境的操作系统和产品配置详细信息开始。

诊断和解决锁定问题

要解决锁定问题，首先必须诊断导致 SQL 查询性能下降或查询无法完成的锁定事件的类型以及所涉及的 SQL 语句。本主题提供可以帮助您诊断锁定问题类型的步骤以及有助于解决锁定问题的步骤。

简介

如果应用程序由于遇到故障而无法完成任务，或者 SQL 查询性能由于锁定而下降，那么锁定问题是正确的诊断。因此，理想的目标是，避免在数据库系统上发生任何锁定超时或死锁，这两种情况都将导致应用程序无法完成它们的任务。

锁定等待是可以预期的正常事件，但如果等待锁定时耗用的时间过多，那么锁定等待会导致 SQL 查询性能下降以及应用程序完成时间过长。过长的锁定等待持续时间还有成为锁定超时的风险，这将导致应用程序无法完成它的任务。

锁定升级如果导致锁定超时，那么将被认为是锁定问题。在理想情况下，我们的目标是避免任何锁定升级，但如果不会产生负作用，少量的锁定升级可接受。

建议您持续监视锁定等待、锁定超时和死锁锁定事件；通常，在工作负载级别监视锁定等待，并在数据库级别监视锁定超时和死锁。

诊断所发生的锁定问题的类型以及确定其解决方案时，首先应收集信息并查找诊断指示器。下列各节将引导您完成此过程。

收集信息

通常，为了能够有针对性地评估系统行为是否不正常（这可能包括处理延迟以及性能不佳），您必须要有描述了系统典型行为（基线）的信息。然后，可以将您观察到的可疑异常行为与基线进行比较。通过安排定期的运作监视任务来收集基线数据是故障诊断过程的关键组成部分。有关确定系统的基线操作的更详细信息，请参阅第 9 页的『对系统性能进行运作监视』.

要确认哪一类锁定问题是 SQL 查询性能下降或查询无法完成的原因，有必要收集一些信息，以便帮助您确定所涉及的锁定事件类型、哪个应用程序正在请求或挂起此锁定、应用程序在此事件期间执行的操作以及速度显著变慢的 SQL 语句。

要收集此类信息，您可以创建锁定事件监视器、使用表函数或者使用 db2pd 命令。锁定事件监视器所收集的信息可以分为三大类:

- 关于所涉及的锁定的信息
- 关于请求此锁定的应用程序及其当前活动的信息。对于死锁而言，这是关于被称为“牺牲者”的语句的信息。
- 关于拥有此锁定的应用程序及其当前活动的信息。对于死锁而言，这是关于被称为“参与者”的语句的信息。

有关如何监视锁定等待、锁定超时和死锁锁定事件的指示信息，请参阅《*数据库监视指南和参考*》中的『监视锁定事件』。

查找诊断指示器

锁定事件监视器、表函数或者运行 db2pd 命令都可以收集有助于您确定锁定问题性质的信息。确切而言，下列主题包含在诊断方面有指示性的信息，可以帮助您诊断和确认您所遇到的特定类型锁定问题。

- 如果等待时间较长，并且不存在锁定超时情况， 那么可能发生了锁定等待问题。要进行确认: 诊断锁定等待问题
- 如果死锁数增大到超出基线数目，那么很可能存在死锁问题。要进行确认: 诊断死锁问题
- 如果锁定超时数增大，并且 **locktimeout** 数据库配置参数设置为非零时间值，那么很可能存在锁定超时问题。要进行确认（并考虑锁定等待问题）: 诊断锁定超时问题
- 如果锁定等待数超出正常情况，并且锁定事件监视器表明发生锁定升级（是），那么很可能存在锁定升级问题。要进行确认: 诊断锁定升级问题

诊断锁定等待问题

当一个事务尝试对已被另一事务占用的资源获取锁定时，将发生锁定等待。如果锁定等待的持续时间过长，那么将导致 SQL 查询的执行速度下降。如果您发现锁定等待时间较长或超出预期，并且未发生锁定超时情况，那么可能是存在锁定等待问题。

开始前

通常，为了能够有针对性地评估系统行为是否不正常（这可能包括处理延迟以及性能不佳），您必须要有描述了系统典型行为（基线）的信息。然后，可以将您观察到的

可疑异常行为与基线进行比较。通过安排定期的运作监视任务来收集基线数据是故障诊断过程的关键组成部分。有关确定系统的基线操作的更详细信息，请参阅第 9 页的『对系统性能进行运作监视』.

有关如何监视锁定等待锁定事件的指示信息，请参阅《*数据库监视指南和参考*》中的『监视锁定事件』。

过程

诊断 当一个事务（由一个或多个 SQL 语句组成）尝试获取一个锁定，但锁定方式与另一事务挂起的锁定有冲突时，将发生锁定等待。锁定等待时间过长通常表现为响应速度较慢，因此，监视此时间十分重要。因为单一事务的锁定等待时间通常非常短，并且规范化的度量值易于处理，所以锁定等待时间按 1000 个事务为单位进行最佳规范化。

在尝试确认每个锁定等待的诊断方法时，必须考虑锁定等待的不同性质。以下是三种不同性质的锁定等待及其最佳诊断方法的列表:

- 单个锁定等待的时间较长
  - 检查服务类和工作负载的锁定等待时间峰值。请对工作负载设置锁定事件监视器以获取该值。
- 锁定等待时间较长，但各个锁定等待的时间较短
  - 通常由锁定防护所致。请使用 db2pd -locks wait 命令来检测锁定链。
- 正在等待的锁定的类型
  - 检查此类型可能有助于确定问题。查找正在等待锁定的代理程序，以获取有关锁定类型的信息。使用锁定类型信息来确定是否发生了某些显而易见的情况。例如，程序包锁定可能表明 BIND/REBIND 命令或 DDL 与该程序包的用户有冲突；内部 c（目录高速缓存）锁定可能表明 DDL 与语句编译有冲突。

指示信号

请查找下列锁定等待指示信号:

- 锁定等待数不断增加（**lock_waits** 监视元素的值不断增大）。
- 正在等待锁定的活动代理程序所占的百分比较高（例如，活动代理程序总数的 20% 或更高）。有关如何获取此信息的信息，请参阅下一节，即“要监视的内容”。
- 在数据库或工作负载级别捕获的锁定等待时间值（**lock_wait_time** 监视元素）不断增大。

要监视的内容

与许多其他类型的 DB2 监视器数据不同，锁定信息具有非常强的瞬态性。除 **lock_wait_time**（这是累积总计）以外，大部分其他锁定信息在锁定本身被释放后就会消失。因此，如果在一段时间内定期收集锁定和锁定等待事件数据以便能够更好地了解演进情况，那么最能反映这些数据的价值。

要收集关于正在等待锁定的活动代理程序的信息，请使用 WLM_GET_SERVICE_CLASS_AGENTS_V97 表函数。正在等待锁定的代理程序由代理程序通过下列“属性-值”对指示:

- EVENT_OBJECT = LOCK

- EVENT_TYPE = ACQUIRE

您还可以使用应用程序快照、锁定管理性视图或者 db2pd -wlocks 命令的锁定等待选项来获取关于正在等待锁定的活动代理程序的信息。

关键的指示器监视元素如下所示:

- **lock_waits** 值不断增大
- **lock_wait_time** 值较大

如果您观察到这里列示的一个或多个指示信号，那么很有可能存在锁定等待问题。请遵循“下一步做什么”一节中的链接来解决此问题。

下一步任务

在诊断出您所遇到的问题可能由锁定等待所致之后，请执行步骤来解决问题: 『解决锁定等待问题』

解决锁定等待问题

在诊断锁定等待问题之后，下一步是尝试解决由于应用程序等待锁定时耗时过长而引起的问题。这里提供的准则帮助您解决锁定等待问题，并且将帮助您预防将来发生此类意外情况。

开始前

通过执行第 413 页的『诊断和解决锁定问题』中概述的针对锁定问题的必要诊断步骤，确认您遇到锁定等待问题。

关于此任务

这里提供的准则可以帮助您解决您所遇到的锁定等待问题，并可以帮助您预防将来发生此类意外。

过程

请使用下列步骤来诊断不可接受的锁定等待问题的原因并应用修正措施。

1. 借助于管理通知日志，获取关于代理程序在其中等待锁定时耗时过长的表的信息。
2. 使用管理通知日志中的信息来决定如何解决锁定等待问题。您可以通过遵循多个准则来减少锁定争用情况并缩短锁定等待时间。请考虑下列选项:
   - 有可能的话，避免使用非常长的事务和 WITH HOLD 游标。挂起锁定的时间越长，它们导致与其他应用程序发生争用的机会就越大。仅当使用高隔离级别时，这才是问题。
   - 最佳实践是，尽早落实下列操作:
     - 写操作，例如删除、插入和更新
     - 数据定义语言（DDL）语句，例如 ALTER、CREATE 和 DROP 语句
     - BIND 和 REBIND 命令
   - 在发出 ALTER 或 DROP DDL 语句之后，运行 SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS 过程以使任何数据对象重新生效，并运行 db2rbind 命令以重新绑定任何程序包。
   - 避免访存大于所需的结果集，在可重复读（RR）隔离级别下尤其要避免这种情况。所访问的行数越多，挂起的锁定越多，遇到被别人挂起的锁定的机会就越

大。实际上，这通常意味着将行选择标准下推到 SELECT 语句的 WHERE 子句中，而不是返回较多的行并在应用程序中对其进行过滤。例如:

```
exec sql declare curs for
  select c1,c2 from t
  where c1 not null;
exec sql open curs;
do {
  exec sql fetch curs
    into :c1, :c2;
} while( P(c1) != someVar );

==>

exec sql declare curs for
  select c1,c2 from t
  where c1 not null
  and myUdfP(c1) = :someVar;
exec sql open curs;
exec sql fetch curs
    into :c1, :c2;
```

- 避免使用高于所需的隔离级别。为了在应用程序中维护结果集完整性，可能需要使用“可重复读”隔离级别；但是，此隔离级别将在挂起的锁定和潜在锁定冲突方面产生额外的成本。
- 如果适合于应用程序中的业务逻辑，请考虑通过 **DB2_EVALUNCOMMITTED**、**DB2_SKIPDELETED** 和 **DB2_SKIPINSERTED** 注册表变量来修改锁定行为。这些注册表变量使 DB2 数据库管理器能够在某些情况下延迟或避免挂起锁定，从而减少争用情况并有可能提高吞吐量。
- 尽可能消除锁定升级。

下一步任务

请重新运行应用程序，然后在管理通知日志中查找与锁定相关的条目，或者检查相应工作负载、连接、服务子类、工作单元和活动级别的锁定等待和锁定等待时间度量值，确保已消除锁定问题。

诊断死锁问题

死锁是指两个应用程序彼此锁定对方所需的数据，这将导致在死锁检测器不介入的情况下，这两个应用程序都无法继续执行。死锁将导致参与者事务由于等待死锁检测而减慢，将导致由于回滚牺牲事务而浪费系统资源，并且将导致在整个进程期间执行额外的系统工作和事务日志访问。如果死锁数增大到超出基线数目，并且事务被重新执行，那么可能是发生了死锁问题。

开始前

通常，观察到的任何死锁都被认为是异常情况。为了能够有针对性地评估系统行为是否不正常（这可能包括处理延迟以及性能不佳），您必须要有描述了系统典型行为（基线）的信息。然后，可以将您观察到的可疑异常行为与基线进行比较。通过安排定期的运作监视任务来收集基线数据是故障诊断过程的关键组成部分。有关确定系统的基线操作的更详细信息，请参阅第 9 页的『对系统性能进行运作监视』.

有关如何监视死锁锁定事件的指示信息，请参阅《*数据库监视指南和参考*》中的『监视锁定事件』。

过程

诊断 死锁是指两个应用程序彼此锁定对方所需的数据，这将导致在死锁检测器不介入的情况下，这两个应用程序都无法继续执行。在系统自动回滚先前发生死锁的事务之后，牺牲应用程序必须从头开始重新执行该事务。监视这种情况的发生比率有助于避免由于死锁过多而大幅增加系统负载但不被 DBA 所了解的情况。

指示信号

请查找下列死锁指示信号:

- 一个或多个应用程序偶尔重新执行事务
- 管理通知日志中的死锁消息条目
- **deadlocks** 监视元素指示死锁数增加
- **int_deadlock_rollbacks** 监视元素指示回滚数增加
- 代理程序等待日志记录被清仓到磁盘时耗用的时间增加，这由 **log_disk_wait_time** 监视元素指示

要监视的内容

死锁的成本会有所变化，并且与所回滚的事务的长度成正比。虽然如此，任何死锁通常都表明发生问题。

实际上，可以通过三种方法来检测死锁事件:

1. 设置锁定事件监视器，并设置 **mon_deadlock** 数据库配置参数以捕获有关数据库中发生的所有锁定事件的详细信息
2. 在管理通知日志中监视死锁消息以及附带的基本信息

   注: 为了能够将死锁消息写入管理通知日志文件，请将 **mon_lck_msg_lvl** 数据库管理器配置参数的值设置为 2。
3. 通过表函数来监视指示器监视元素

大多数用户采用第一种方法。通过监视关键的指示器监视元素来检测死锁的发生时间，用户可以通过检查事件监视器所收集的信息来获取详细信息。

关键的指示器监视元素如下所示:

- **deadlocks** 值为非零
- **int_deadlock_rollbacks** 表明回滚数由于死锁事件而增加
- **log_disk_wait_time** 表明代理程序等待日志被清仓到磁盘时耗用的时间增加

如果您观察到这里列示的一个或多个指示信号，那么很有可能存在死锁问题。请遵循“下一步做什么”一节中的链接来解决此问题。

下一步任务

在诊断出您所遇到的问题可能由死锁所致之后，请执行步骤来解决问题: 第 419 页的『解决死锁问题』

解决死锁问题

在诊断死锁问题之后，下一步是尝试解决两个同时运行并相互锁定对方所需资源的应用程序之间的死锁问题。这里提供的准则可以帮助您解决您所遇到的死锁问题，并可以帮助您预防将来发生此类意外。

开始前

通过执行第 413 页的『诊断和解决锁定问题』中概述的针对锁定问题的必要诊断步骤，确认您遇到死锁问题。

关于此任务

这里提供的准则可以帮助您解决您所遇到的死锁问题，并可以帮助您预防将来发生此类意外。

过程

请使用下列步骤来诊断不可接受的死锁问题的原因并应用修正措施。

1. 借助于锁定事件监视器或管理通知日志，获取关于代理程序在其中遇到死锁问题的表的信息。
2. 使用管理通知日志中的信息来决定如何解决死锁问题。您可以通过遵循多个准则来减少锁定争用情况并缩短锁定等待时间。请考虑下列选项:
   - 每个应用程序连接都应该处理它自己的一组行，以避免锁定等待。
   - 有时，可以通过确保所有应用程序按相同顺序访问公共数据来降低死锁频率 – 例如，这意味着它们访问（并因此锁定）表 A 中的行，接着访问表 B 中的行，接着访问表 C 中的行，依此类推。如果两个应用程序按不同的顺序对相同对象挂起不兼容的锁定，那么它们发生死锁的风险将显著加大。
   - 锁定超时情况并不比死锁好多少，这两者都会导致事务回滚，但如果您必须最大程度地减少死锁数，那么可以通过确保锁定超时通常在相关的潜在死锁能够被检测到之前发生来实现目标。要做到这一点，请将 **locktimeout** 数据库配置参数值（单位为秒）设置为远小于 **dlchktime** 数据库配置参数值（单位为毫秒）。否则，如果 **locktimeout** 大于 **dlchktime** 时间间隔，那么死锁检测器可能会在死锁情况发生后立即被唤醒并在锁定超时发生前检测到死锁。
   - 尽可能避免并发 DDL 操作。例如，DROP TABLE 语句可能会引起大量的目录更新操作，这是因为，除了为表本身删除行以外，还可能必须为表索引、主键和检查约束等等删除行。如果其他 DDL 操作是删除或创建对象，那么可能会发生锁定冲突，偶尔甚至会发生死锁。
   - 最佳实践是，尽早落实下列操作:
     – 写操作，例如删除、插入和更新
     – 数据定义语言（DDL）语句，例如 ALTER、CREATE 和 DROP
     – BIND 和 REBIND 命令
3. 死锁检测器无法了解和解决下列情况，因此应用程序设计必须对此进行预防。应用程序（尤其是多线程应用程序）可能会发生涉及 DB2 锁定等待以及等待非 DB2 资源（例如信号）的死锁。例如，连接 A 可能正在等待连接 B 所挂起的锁定，而 B 可能正在等待 A 占用的信号。

下一步任务

请重新运行应用程序，然后在管理通知日志中查找与锁定相关的条目，确保已消除锁定问题。

诊断锁定超时问题

如果等待资源锁定的事务的等待时间过长，导致超出 **locktimeout** 数据库配置参数所指定的等待时间值，那么将发生锁定超时。这将耗用时间，从而导致 SQL 查询性能下降。如果锁定超时数增大，并且 **locktimeout** 数据库配置参数设置为非零时间值，那么很可能存在锁定超时问题。

开始前

通常，为了能够有针对性地评估系统行为是否不正常（这可能包括处理延迟以及性能不佳），您必须要有描述了系统典型行为（基线）的信息。然后，可以将您观察到的可疑异常行为与基线进行比较。通过安排定期的运作监视任务来收集基线数据是故障诊断过程的关键组成部分。有关确定系统的基线操作的更详细信息，请参阅第 9 页的『对系统性能进行运作监视』.

有关如何监视锁定超时锁定事件的指示信息，请参阅《*数据库监视指南和参考*》中的『监视锁定事件』。

过程

诊断 有时，锁定等待情况将引起锁定超时，从而导致事务回滚。锁定等待导致锁定超时前经过的时间段由数据库配置参数 **locktimeout** 指定。数目过多的锁定超时将像死锁一样对系统造成破坏。虽然死锁在大多数生产系统中相对罕见，但锁定超时情况可能比较常见。应用程序通常必须以类似方式对其进行处理：从头开始重新执行事务。监视这种情况的发生比率有助于避免由于锁定超时过多而大幅增加系统负载但不被 DBA 所了解的情况。

指示信号

请查找下列锁定超时指示信号:

- 应用程序频繁地重新执行事务
- **lock_timeouts** 监视元素的值不断增大
- 管理通知日志中的锁定超时消息条目

要监视的内容

锁定事件具有相对瞬态的性质，因此，如果在一段时间内定期收集锁定事件数据以便能够更好地了解演进情况，那么最能反映这些数据的价值。

您可以在管理通知日志中监视锁定超时消息。

注：为了能够将锁定超时消息写入管理通知日志文件，请将 **mon_lck_msg_lvl** 数据库管理器配置参数的值设置为 3。

创建事件监视器，以便捕获工作负载或数据库的锁定超时数据。

关键的指示器监视元素如下所示:

- **lock_timeouts** 值不断增大
- **int_rollbacks** 值不断增大

如果您观察到这里列示的一个或多个指示信号，那么很有可能存在锁定超时问题。请遵循“下一步做什么”一节中的链接来解决此问题。

下一步任务

在诊断出您所遇到的问题可能由锁定超时所致之后，请执行步骤来解决问题: 『解决锁定超时问题』

解决锁定超时问题

在诊断锁定超时问题之后，下一步是尝试解决由于应用程序等待锁定直至发生锁定超时而引起的问题。这里提供的准则可以帮助您解决您所遇到的锁定超时问题，并可以帮助您预防将来发生此类意外。

开始前

通过执行第 413 页的『诊断和解决锁定问题』中概述的针对锁定问题的必要诊断步骤，确认您遇到锁定超时问题。

关于此任务

这里提供的准则可以帮助您解决您所遇到的锁定超时问题，并可以帮助您预防将来发生此类意外。

过程

请使用下列步骤来诊断不可接受的锁定超时问题的原因并应用修正措施。

1. 借助于锁定事件监视器或管理通知日志，获取关于代理程序在其中遇到锁定超时问题的表的信息。
2. 使用管理通知日志中的信息来决定如何解决锁定超时问题。您可以通过遵循多个准则来减少锁定争用情况以及缩短锁定等待时间，这有助于减少锁定超时数。请考虑下列选项:
   - 将 **locktimeout** 数据库配置参数调整为适合于数据库环境的秒数。
   - 有可能的话，避免使用非常长的事务和 WITH HOLD 游标。挂起锁定的时间越长，它们导致与其他应用程序发生争用的机会就越大。
   - 最佳实践是，尽早落实下列操作:
     - 写操作，例如删除、插入和更新
     - 数据定义语言（DDL）语句，例如 ALTER、CREATE 和 DROP
     - BIND 和 REBIND 命令
   - 避免访存大于所需的结果集，在可重复读（RR）隔离级别下尤其要避免这种情况。所访问的行数越多，挂起的锁定越多，遇到被别人挂起的锁定的机会就越大。实际上，这通常意味着将行选择标准下推到 SELECT 语句的 WHERE 子句中，而不是返回较多的行并在应用程序中对其进行过滤。例如:

```
exec sql declare curs for
  select c1,c2 from t
  where c1 not null;
exec sql open curs;
do {
  exec sql fetch curs
    into :c1, :c2;
} while( P(c1) != someVar );

==>

exec sql declare curs for
```

```
  select c1,c2 from t
  where c1 not null
  and myUdfP(c1) = :someVar;
exec sql open curs;
exec sql fetch curs
    into :c1, :c2;
```

- 避免使用高于所需的隔离级别。为了在应用程序中维护结果集完整性，可能需要使用“可重复读”隔离级别；但是，此隔离级别将在挂起的锁定和潜在锁定冲突方面产生额外的成本。
- 如果适合于应用程序中的业务逻辑，请考虑通过 **DB2_EVALUNCOMMITTED**、**DB2_SKIPDELETED** 和 **DB2_SKIPINSERTED** 注册表变量来修改锁定行为。这些注册表变量使 DB2 数据库管理器能够在某些情况下延迟或避免挂起锁定，从而减少争用情况并有可能提高吞吐量。

下一步任务

请重新运行应用程序，然后在管理通知日志中查找与锁定相关的条目，或者检查相应工作负载、连接、服务子类、工作单元和活动级别的锁定等待和锁定等待时间度量值，确保已消除锁定问题。

诊断锁定升级问题

锁定升级是指，为了减少分配给锁定的内存（锁定空间），将多个行级别锁定升级为能够节省内存的单一表锁定。这种情况虽然自动发生并且可以节省供锁定专用的内存空间，但会导致一致性级别降低到不可接受的程度。如果锁定等待数超出典型水平，并且管理通知日志条目表明发生了锁定升级，那么很可能存在锁定升级问题。

开始前

通常，为了能够有针对性地评估系统行为是否不正常（这可能包括处理延迟以及性能不佳），您必须要有描述了系统典型行为（基线）的信息。然后，可以将您观察到的可疑异常行为与基线进行比较。通过安排定期的运作监视任务来收集基线数据是故障诊断过程的关键组成部分。有关确定系统的基线操作的更详细信息，请参阅第 9 页的『对系统性能进行运作监视』.

过程

诊断 发生从多个行级别锁定到单一表级别锁定的锁定升级的原因包括:

- 对一个表挂起的多个行级别锁定所耗用的内存总量超出为了存储锁定而分配的内存总量百分比
- 锁定列表耗尽空间。导致锁定列表被耗尽的应用程序将通过锁定升级过程强制实施其锁定，尽管该应用程序并不是大多数锁定的挂起者。

为了存储锁定而分配的内存总量百分比阈值（应用程序必须超出此阈值才会导致锁定升级）由 **maxlocks** 数据库配置参数定义，分配给锁定的内存量由 **locklist** 数据库配置参数定义。在配置良好的数据库中，很少发生锁定升级。如果锁定升级导致并行性降低到不可接受的程度，那么您可以分析问题并确定最佳应对措施。

从内存空间的角度看，如果锁定内存由自调整内存管理器（STMM）管理，而不是仅仅由 **locklist** 数据库配置参数分配，那么锁定升级并不是严重的问题。STMM 在万一耗尽空闲内存空间的情况下，将自动调整锁定内存空间。

指示信号

请查找下列锁定升级指示信号:

- 管理通知日志中的锁定升级消息条目

要监视的内容

锁定事件具有相对瞬态的性质，因此，如果在一段时间内定期收集锁定事件数据以便能够更好地了解演进情况，那么最能反映这些数据的价值。

请检查此监视元素，确定锁定升级是否是其中一个导致 SQL 查询性能下降的因素:

- **lock_escals**

如果您观察到这里列示的一个或多个指示信号，那么很有可能存在锁定升级问题。请遵循“下一步做什么”一节中的链接来解决此问题。

下一步任务

在诊断出您所遇到的问题可能由锁定升级所致之后，请执行步骤来解决问题: 『解决锁定升级问题』

解决锁定升级问题

在诊断锁定升级问题之后，下一步是尝试解决由于数据库管理器自动地将锁定由行级别升级到表级别而引起的问题。这里提供的准则可以帮助您解决您所遇到的锁定升级问题，并可以帮助您预防将来发生此类意外。

开始前

通过执行第 413 页的『诊断和解决锁定问题』中概述的针对锁定问题的必要诊断步骤，确认您遇到锁定升级问题。

关于此任务

这里提供的准则可以帮助您解决您所遇到的锁定升级问题，并可以帮助您预防将来发生此类意外。

过程

我们的目标是最大程度地减少锁定升级，有可能的话消除这些问题。良好的应用程序设计以及锁定处理方面的数据库配置可以最大程度地减少甚至消除锁定升级。锁定升级可能会导致一致性水平下降，并可能引起锁定超时，因此解决锁定升级是一项重要任务。您可以使用 **lock_escals** 监视元素以及写入管理通知日志的消息来确定和更正锁定升级。

首先，确保记录锁定升级信息。将 **mon_lck_msg_lvl** 数据库管理器配置参数的值设置为 1。这是缺省设置。当发生锁定升级事件时，将记录与锁定、工作负载、应用程序、表和错误 SQLCODE 有关的信息。如果当前正在执行动态 SQL 语句，那么还会记录查询。

请使用下列步骤来诊断不可接受的锁定升级问题的原因并应用修正措施。

1. 通过管理通知日志，收集关于所有发生锁定升级的表以及所涉及应用程序的信息。 此日志文件包含下列信息:

- 当前挂起的锁定数
- 在完成锁定升级之前所需的锁定数
- 每个正在进行锁定升级的表的表标识和表名
- 当前挂起的非表锁定的数目
- 在锁定升级过程中要获取的新表级锁定。通常，将获取 S 或 X 锁定
- 与新表级锁定的获取相关联的内部返回码

2. 使用关于锁定升级所涉及的应用程序的管理通知日志信息来确定如何解决升级问题。 请考虑下列选项:
   - 检查 **maxlocks** 和/或 **locklist** 数据库配置参数，有可能的话，对其进行调整。在分区数据库系统中，请对所有数据库分区进行此更改。**locklist** 配置参数的值对于当前工作负载而言可能太小。如果多个应用程序遇到锁定升级问题，那么可能表明需要增大锁定列表大小。增大工作负载或者添加新应用程序都可能会导致锁定列表变得过小。如果只有一个应用程序遇到锁定升级问题，那么调整 **maxlocks** 配置参数可能能够解决此问题。但是，您可能想同时增大 **maxlocks** 和 **locklist** - 如果允许一个应用程序使用较大份额的锁定列表，那么所有其他应用程序现在可能会耗尽锁定列表中的其余可用锁定并发生锁定升级问题。
   - 您可能想考虑运行应用程序和 SQL 语句时采用的隔离级别，例如 RR、RS、CS 或 UR。RR 和 RS 隔离级别有可能引起更多的锁定升级，这是因为锁定将一直挂起到发出 COMMIT 为止。CS 和 UR 隔离级别并不会将锁定一致挂起到发出 COMMIT 为止，因此发生锁定升级的可能性不大。请使用应用程序所能容忍的最低可能隔离级别。
   - 根据业务需要以及应用程序的设计，提高应用程序中的落实频率。提高落实频率将减少在任意给定时刻挂起的锁定数。这有助于防止应用程序达到 **maxlocks** 值（从而避免触发锁定升级），并有助于防止所有应用程序耗尽锁定列表。
   - 您可以修改应用程序，以便使用 LOCK TABLE 语句来获取表锁定。对于由多个应用程序和用户进行并发访问并不关键的表而言，这是一种良好的策略；例如，当应用程序使用对此应用程序实例唯一指定的永久工作表（例如，不是 DGTT）时，情况即如此。在这种情况下，获取表锁定是一种良好的策略，这将减少应用程序所挂起的锁定数并提高性能；其原因在于，对于在工作表中访问的行，不再需要获取和释放行锁定。

     如果应用程序没有工作表，并且您无法增大 **locklist** 或 **maxlocks** 配置参数的值，那么可以让应用程序获取表锁定。但是，在选择要锁定的表时，务必谨慎小心。请避免选择由许多应用程序和用户访问的表，原因是锁定这些表将引起并行性问题，这可能会影响响应时间，在最坏的情况下，有可能导致应用程序发生锁定超时。

下一步任务

请重新运行应用程序，然后在管理通知日志中查找与锁定相关的条目，确保已消除锁定问题。

从已承受的陷阱恢复

对于遇到的陷阱，DB2 实例将准备首次出现数据捕获（FODC）包。缺省情况下，已将 DB2 实例配置为具有陷阱弹性。DB2 实例还确定陷阱是否可承受。术语“可承受”的意思是，遇到陷阱的 DB2 引擎线程已被暂挂，而 DB2 实例将继续运行。

缺省情况下，已根据 **DB2RESILIENCE** 注册表变量的缺省设置将 DB2 实例配置为具有陷阱弹性。

过程

识别已承受的陷阱

承受陷阱的目的是，在遇到陷阱（DB2 编程错误）时，最大程度地降低对数据库系统的影响。已承受的陷阱将导致进行下列诊断:

1. 在 **diagpath** 数据库管理器配置参数所指定的标准路径中创建 FODC 目录。
2. 将错误消息 ADM14013C 记录到管理通知和 db2diag 日志文件。

   注: 如果未能承受陷阱，从而导致实例关闭，那么将记录 ADM14011C。
3. 将错误 sqlcode -1224 返回给应用程序。
4. EDU 线程暂挂，您可以在 db2pd -edus 的输出中观察到这种情况。

恢复 虽然已承受的陷阱应该不会妨碍实例的常规操作，但暂挂的 EDU 线程却占用着某些资源，因此建议您通过执行下列步骤在最方便的时候停止并重新启动该实例:

1. 要终止所有在超时时间段内发出 COMMIT 或 ROLLBACK 的活动应用程序（这将最大程度地缩小运行 db2start 命令时的崩溃恢复窗口），请发出以下命令:

   ```
   db2 quiesce instance instance_name user user_name
   defer with timeout minutes
   ```
2. [可选] 要终止任何未在步骤 1 的超时时间段内发出 COMMIT 或 ROLLBACK 的应用程序以及任何在超时时间段结束后访问该数据库的新应用程序，请发出以下命令:

   ```
   db2 quiesce instance instance_name user user_name immediate
   ```
3. 通过执行以下命令，强制关闭该实例以及暂挂的 EDU:

   ```
   db2_kill
   ```

   注: 发出 db2stop 命令并不会结束已承受陷阱的实例。
4. 使用下列任何一条命令，重新启动 DB2 实例:

   ```
   db2start
   ```

   或者

   ```
   START DATABASE MANAGER
   ```

诊断 找到 **diagpath** 数据库管理器配置参数所指定的 FODC 目录。另外，还可以通过查看管理通知或 db2diag 日志文件来确认 FODC 目录的位置。请将 FODC 信息转发给 IBM 软件支持机构。

对管理任务调度程序进行故障诊断

以下核对表可以帮助您对在管理任务调度程序中运行任务时出现的问题进行故障诊断:

过程

1. 如果任务未按照预期执行，那么您应该做的第一件事就是在 ADMIN_TASK_STATUS 管理视图中查找执行状态记录。

- 如果记录存在，请检查各个值。尤其要注意 STATUS，INVOCATION、SQLCODE，SQLSTATE，SQLERRMC 和 RC 列。这些值通常标志问题的根源。
- 如果视图中没有执行状态记录，那么说明任务未执行。对于此种情况，有多种可能解释:
  - 管理任务调度程序已禁用。如果管理任务调度程序已禁用，那么不会执行任务。要启用该调度程序，请设置 **DB2_ATS_ENABLE** 注册表变量。
  - 任务已除去。可能有人已将任务除去。请通过查询 ADMIN_TASK_LIST 管理视图来确认任务是否存在。
  - 调度程序不能识别任务。管理任务调度程序通过每隔五分钟就连接到每个活动数据库来查找新的和更新过的任务。此时间段过后，调度程序才会识别任务。请至少等待五分钟。
  - 数据库未处于活动状态。只有当数据库处于活动状态时，管理任务调度程序才能检索或执行任务。请激活数据库。
  - 事务未落实。管理任务调度程序会忽略未落实任务。请确保在添加、更新或除去任务后进行落实。
  - 时刻表无效。任务的时刻表可能会使任务无法运行。例如，任务可能已达到调用的最大数目。请检索 ADMIN_TASK_LIST 视图中任务的时刻表，并在需要时更新时刻表。

2. 如果通过参阅 ADMIN_TASK_STATUS 管理视图无法确定问题起因，那么请参阅 DB2 诊断日志。 所有严重错误都将记录在 db2diag 日志文件中。任务执行期间，管理任务调度程序守护程序也会记录参考事件消息。这些错误和消息通过“管理任务调度程序”组件进行了标识。

下一步任务

如果您遵循以上步骤但仍然无法确定问题的起因，那么请考虑向 IBM 软件支持机构开放问题管理记录（PMR）。请告诉他们，您已经遵循了这些指示信息，并将您所收集的诊断数据发送给他们。

对压缩进行故障诊断

未自动创建数据压缩字典

您有一个大型的表或者表的大型 XML 存储器对象，但未创建数据压缩字典。您希望了解未按预期创建数据压缩字典的原因。本资料既适用于表对象的压缩字典也适用于 XML 存储器对象的压缩字典。

您可能会处于以下情况:

- 某个表的 COMPRESS 属性设置为 YES。
- 该表已存在一段时间，并且曾添加和除去数据。
- 表的大小接近阈值大小。您希望自动创建数据压缩字典。
- 您运行了表数据填充操作（例如 INSERT、LOAD INSERT 或 REDISTRIBUTE），您预期这将使表大小增大并超出阈值大小。
- 未自动创建数据压缩字典。未创建数据压缩字典并将其放入表中。您期望对该时间点之后添加至表的数据进行压缩，但该数据仍处于未压缩状态。

- 对于 XML 数据而言，该数据处于 DB2 版本 9.7 存储格式。

  如果表包含使用 DB2 版本 9.5 或更低版本创建的 XML 列，那么不支持对此表的 XML 存储器对象中的数据进行压缩。如果对这样的表启用数据行压缩功能，那么将只对表对象中的表行数据进行压缩。如果无法在插入、装入或重组操作期间对 XML 存储器对象进行压缩，那么仅当 XML 列是使用 DB2 V9 或 DB2 V9.5 创建的时，才会将一条消息写入 db2diag 日志文件。

为何未执行数据压缩？

虽然大于阈值大小的数据将致使自动创建压缩字典，但还将检查另一个条件。此条件是，该对象必须包含足够的数据才能创建字典，消息 ADM5591W 将指示此要求。过去对此数据执行的活动可能也包括删除或除去数据。在该对象中，可能存在若干未包含数据的大块。因此，大对象可能超出或正好符合对象大小阈值，但该对象中可能没有足够的数据来创建字典。

如果对该对象执行了大量活动，那么需要定期重组该对象。对于 XML 数据而言，您需要使用 longlobdata 选项来重组该表。如果不进行重组，那么对象大小可能会很大，但其中填充的数据将很稀疏。重组对象将消除数据碎片并压缩该对象中的数据。完成重组后，该对象将变小，并且数据的填充将更紧密。重组后的对象将更准确地体现该对象中的数据量，并可能小于允许自动创建数据压缩字典的阈值大小。

如果该对象的填充较为稀疏，那么可以使用 REORG TABLE 命令来执行表重组（对于 XDA，请使用 LONGLOBDATA 选项），从而创建字典。缺省情况下，指定了 KEEPDICTIONARY。可以指定 RESETDICTIONARY 以强制创建字典。

使用 REORGCHK 命令来确定是否需要重组表。

如果未对一个表启用数据行压缩功能，那么将不会为该表自动创建字典（ADC）。如果已对数据库禁用 ADC 处理功能，那么将返回消息 ADM5594I，此消息将描述禁用原因。

如果该表包含使用 DB2 版本 9.5 或更低版本创建的 XML 列，请使用 ADMIN_MOVE_TABLE 存储过程来升级该表，然后启用数据行压缩功能。

行压缩功能未减少临时表的磁盘存储空间量

在一些已知的情况下，即使获得“存储器优化”功能部件许可证，也无法对临时表实现预期的节省磁盘存储空间目标。

症状

对临时表启用行压缩功能后，未实现所预期的节省磁盘空间量目标。

原因

- 发生这种情况的原因最有可能是，同时有大量应用程序正在运行和创建临时表，每个应用程序都耗用其中一部分数据库管理器内存。这将导致没有足够的内存可用于创建压缩字典。发生这种情况时，不会提供通知。
- 正在根据算法使用基于字典的方法对行进行压缩。如果临时表的行足够大，能够节省可观的磁盘空间，那么将对该行进行压缩。不会对临时表中较小的行进行压缩，这将导致无法按预期节省磁盘存储空间。发生这种情况时，不会提供通知。

风险

对于系统而言，除了未对行大小低于阈值的临时表使用行压缩功能以外，没有任何风险。但是，如果可用的内存量一直严重短缺，那么会对数据库管理器产生负面影响。

数据复制过程无法将处于压缩状态的行映像解压缩

在一些已知的情况下，数据复制解决方案无法成功地将包含处于压缩状态的行映像的日志记录解压缩。对于瞬时错误（暂时性错误）而言，返回的 SQL 代码将与错误原因相对应，而永久性错误通常由 SQL0204N 通知指示。只有在瞬时错误情况下，以后才有可能成功地将日志记录中的行映像解压缩。db2ReadLog API 将继续处理其他日志记录，即使它无法将某个日志记录解压缩亦如此。

症状

日志阅读器在读取包含处于压缩状态的用户数据的日志记录时，可能会遇到瞬时错误和永久性错误。下列并不全面的示例列表列示了两类错误，在读取包含处于压缩状态的数据（行映像）的日志记录时，可能会遇到这些错误。

瞬时错误:

- 不允许访问表空间
- 无法访问表（锁定超时）
- 装入和存储必需的字典时耗尽内存

永久性错误:

- 表所驻留在的表空间不存在
- 日志记录所属的表或表分区不存在
- 不存在表或表分区的字典
- 日志记录包含行映像，但用于压缩那些映像的字典比表中的字典旧

原因

复制解决方案或任何其他日志阅读器有可能落后于数据库活动，它们在读取包含处于压缩状态的用户数据的日志记录时将接收到错误（参见“方案 1”）。发生这种情况的原因可能是，所读取的日志记录包含处于压缩状态的用户数据，而用于压缩该数据的压缩字典在该日志被读取时比表中的压缩字典旧。

同样，如果表被删除，那么与该表相关联的字典也将被除去。在这种情况下，将无法对该表的处于压缩状态的行映像进行解压缩（参见“方案 2”）。注意，此限制不适用于未处于压缩状态的行映像，这是因为，这些行映像仍可被读取并复制，即使该表被删除亦如此。

对于任何一个表，只能有一个活动的数据压缩字典和一个历史字典。

方案 1:

已对表 t6 启用压缩功能。并且，已对该表启用用于执行复制的 DATA CAPTURE CHANGES 属性。数据复制应用程序正在复制该表，并且日志阅读器正在读取包含处于压缩状态的数据（行映像）的日志记录。在发出 REORG TABLE 命令（这将导致重建表的字典）之后，对表 t6 执行 LOAD 操作时，使用 db2ReadLog API 的客户机日志阅读器读取第一条 INSERT 语句的第一个日志记录。

对表 t6 执行下列语句，该表已包含压缩字典，并且已启用 DATA CAPTURE CHANGES 属性:

```
-> db2 alter table t6 data capture changes
-> db2 insert into t6 values (...)
-> db2 insert into t6 values (...)
```

由于已存在表 t6 的数据压缩字典，因此将使用表 t6 的压缩字典对 ALTER 之后的两条 INSERT 语句进行压缩。此时，日志阅读器尚未到达第一条 INSERT 语句。

以下 REORG TABLE 命令将致使为表 t6 构建新的压缩字典，并且将保留当前压缩字典作为历史字典，从而使日志阅读器比当前压缩字典落后一个字典（但是，在执行 REORG 之后，不会将历史字典装入到内存中）:

```
-> db2 reorg table t6 resetdictionary
```

当日志阅读器读取 INSERT 语句的 INSERT 日志时（此日志现在要求将历史字典装入内存），表 t6 正在执行 LOAD 操作:

```
-> db2 load from data.del of del insert into table t6 allow no access
```

对源表执行 LOAD 时，指定的 ALLOW NO ACCESS 选项将导致以 Z 方式锁定表 t6。日志阅读器必须将历史字典装入内存以便对 INSERT 日志记录中的行映像进行解压缩，但是，访存该字典要求挂起 IN 表锁定。在这种情况下，日志阅读器将无法获取该锁定。这将导致 db2ReadLogFilterData 结构的 sqlcode 成员返回 SQL 代码 SQL2048N。这与瞬时错误相对应（即，如果再次调用该 API，那么*可能*能够将该日志记录解压缩）。日志阅读器将返回日志记录中处于压缩状态的行映像并继续读取下一个日志记录。

方案 2:

已对表 t7 启用 DATA CAPTURE CHANGES 属性。并且，已对该表启用压缩功能以降低存储成本。数据复制应用程序正在复制该表，但是，日志阅读器落后于源表活动；并且，在日志阅读器再次读取日志记录之前，数据压缩字典已被重建两次。

在已启用 DATA CAPTURE CHANGES 属性、已启用表压缩功能以及已构建新字典的情况下，对表 t7 执行下列语句:

```
-> db2 alter table t7 compress yes
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
```

使用 db2ReadLog API 的客户机日志阅读器将要读取与以下第一个 INSERT 语句相对应的下一个日志:

```
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
```

在这种情况下，db2ReadLog API 无法将该日志记录的内容解压缩，这是因为日志阅读器已落后两个或更多个 REORG RESETDICTIONARY 操作。在表中找不到将该日志记录中的行映像解压缩所需的字典；只有第二个 REORG 的压缩字典以及最后一个 REORG 的压缩字典与表存储在一起。但是，db2ReadLog API 不会由于出错而失败。而是，将在用户缓冲区中返回处于未压缩状态的行映像，并且，在日志记录前面的 db2ReadLogFilterData 结构中，sqlcode 成员将返回 SQL 代码 SQL0204N。此代码与永

久性错误相对应（即，永远无法将该日志记录解压缩）。

环境

在任何平台上，只要数据复制解决方案使用 db2ReadLog API，并且已对表设置 DATA CAPTURE CHANGES 属性，就会由于缺少旧压缩字典而无法成功地将处于压缩状态的日志记录解压缩。

解决问题

用户响应:

对于瞬时错误，有可能重新发出读请求并成功地读取日志。例如，如果日志记录属于驻留在表空间中的表，并且不允许访问该表，那么字典可能不可访问，因此无法将日志记录解压缩（参见“方案 1”）。以后，该表空间可能会变为可用，届时重新发出日志读请求将能够成功地将该日志记录解压缩。

- 如果返回了瞬时错误（参见“方案 1”），请阅读错误信息以执行适当的操作。这可能包括等待表操作完成，此时可能能够重新读取该日志记录，并且解压缩将成功。
- 如果发生永久性错误（参见“方案 2”），那么表明由于用于压缩行映像的压缩字典不再可用而无法将日志记录中的行映像解压缩。对于这种情况，复制解决方案可能需要重新初始化受影响的目标表。

对全局变量问题进行故障诊断

一般来说，如果遇到问题的用户有权读取全局变量，他们就可以诊断与全局变量有关的应用程序。只需要具有读许可权，就可以通过发出 VALUES(Global Variable Name) 语句来知道全局变量的值。有时运行应用程序的用户无权读取全局变量。

第一个方案说明引用全局变量时的可能问题有一个简单的解决方案。第二个方案说明更有可能出现的情况：需要对适当用户授予针对全局变量的读许可权。

方案 1

对全局变量的引用必须正确限定。可能存在名称相同但模式不同的变量，并且之前在 PATH 寄存器值中遇到了不正确的模式。一个解决方案是确定对全局变量的引用使用的是标准名称。

方案 2

应用程序开发者（developerUser）根据只有他才有读许可权的一些全局变量的值创建了很复杂的一系列过程、视图、触发器等等。应用程序的最终用户（finalUser）登录并开始使用 developerUser 创建的环境发出 SQL。finalUser 向 developerUser 抱怨他看不到必须允许看到的数据。在诊断此问题时，developerUser 将其授权标识切换为 finalUser 的授权标识，作为 finalUser 登录并尝试发出 finalUser 所发出的 SQL。developerUser 发现 finalUser 所说的问题是事实。

developerUser 必须确定 finalUser 是否看到了他所看到的全局变量。developerUser 运行 SET SESSION USER 以查看 finalUser 所看到的全局变量值。以下是确定并解决此问题的建议方法。

developerUser 要求安全性管理员（secadmUser）授予其许可权以作为 finalUser 使用 SET SESSION USER。然后，developerUser 以本来身份登录并使用 SET SESSION AUTHORIZATION 语句以将 SESSION_USER 专用寄存器设置为 finalUser 的专用寄存器。运行出现该问题的 SQL 后，他使用 SET SESSION AUTHORIZATION 语句切换回 developerUser 身份。developerUser 现在发出 VALUES 语句并会看到全局变量的实际值。

下面是一个样本 SQL，显示 developerUser 在数据库中采取的操作。

```
#########################################################################
# developerUser connects to database and creates needed objects
#########################################################################

db2 "connect to sample user developerUser using xxxxxxxx"

db2 "create table security.users \
(userid varchar(10) not null primary key, \
firstname varchar(10), \
lastname varchar(10), \
authlevel int)"

db2 "insert into security.users values ('ZUBIRI', 'Adriana', 'Zubiri', 1)"
db2 "insert into security.users values ('SMITH', 'Mary', 'Smith', 2)"
db2 "insert into security.users values ('NEWTON', 'John', 'Newton', 3)"

db2 "create variable security.gv_user varchar(10) default (SESSION_USER)"
db2 "create variable security.authorization int default 0"

# Create a procedure that depends on a global variable
db2 "CREATE PROCEDURE SECURITY.GET_AUTHORIZATION() \
SPECIFIC GET_AUTHORIZATION \
RESULT SETS 1 \
LANGUAGE SQL \
   SELECT authlevel INTO security.authorization \
   FROM security.users \
   WHERE userid = security.gv_user"

db2 "grant all on variable security.authorization to public"
db2 "grant execute on procedure security.get_authorization to public"
db2 "terminate"

#########################################################################
# secadmUser grants setsessionuser
#########################################################################
db2 "connect to sample user secadmUser using xxxxxxxx"
db2 "grant setsessionuser on user finalUser to user developerUser"
db2 "terminate"

#########################################################################
# developerUser will debug the problem now
#########################################################################

echo "------------------------------------------------------------"
echo " Connect as developerUser "
echo "------------------------------------------------------------"
db2 "connect to sample user developerUser using xxxxxxxx"

echo "------------------------------------------------------------"
echo " SET SESSION AUTHORIZATION = finalUser "
echo "------------------------------------------------------------"
db2 "set session authorization = finalUser"

echo "--- TRY to get the value of gv_user as finalUser (we must not be able to)"
db2 "values(security.gv_user)"
```

```
echo "--- Now call the procedure---"
db2 "call security.get_authorization()"

echo "--- if it works it must return 3 ---"
db2 "values(security.authorization)"

echo "------------------------------------------------------------"
echo " SET SESSION AUTHORIZATION = developerUser "
echo "------------------------------------------------------------"

db2 "set session authorization = developerUser"

echo "--- See what the variable looks like ----"
db2 "values(security.gv_user)"

db2 "terminate"
```

对高可用性进行故障诊断

DB2 版本 9.5 GA 未在 AIX 6.1 上安装 Tivoli System Automation for Multiplatforms（SA MP）基本组件

DB2 版本 9.5 GA 高可用性功能中提供的 IBM Tivoli® SA MP 基本组件不支持 AIX 6.1 操作系统。要获取适用于 AIX 6.1 的适当版本的 SA MP 基本组件，安装 DB2 版本 9.5 修订包 1 或更高级别的修订包。

症状

如果将 DB2 版本 9.5 GA 数据库产品安装在 AIX 6.1 上，那么安装程序将检测您是否正在使用 AIX 6.1，并且不会安装 SA MP 基本组件。

原因

与 DB2 版本 9.5 GA 捆绑在一起的 SA MP 基本组件不支持 AIX 6.1。

解决问题

将 DB2 版本 9.5 修订包 1 或更高级别的修订包安装在 AIX 6.1 上时，将会成功安装 SA MP 基本组件。

对不一致性进行故障诊断

对数据不一致问题进行故障诊断

诊断数据库中出现数据不一致问题的位置是非常重要的。确定数据不一致的一种方法是使用 INSPECT 命令中的输出来标识存在问题的位置。发现不一致时，必须确定如何处理该问题。

一旦确定存在数据一致性问题，有两个选择:

- 与 IBM 软件支持机构联系，并要求他们帮助您从数据不一致的情况进行恢复
- 删除并重建存在数据一致性问题的数据库对象。

您将使用 INSPECT 命令的 INSPECT CHECK 变体来检查存在数据不一致的数据库、表空间或表。一旦生成 INSPECT CHECK 命令的结果，就应使用 db2inspf 命令格式化检查结果。

如果 INSPECT 命令未完成，请与 IBM 软件支持机构联系。

对索引与数据不一致问题进行故障诊断

索引必须是准确的，才能快速访问表中的正确数据，除非数据库已毁坏。

可使用 INSPECT 命令并通过在交叉对象检索子句中使用 INDEXDATA 选项，以对索引与数据不一致问题执行联机检查。使用 INSPECT 命令时，缺省情况下不会执行索引数据检查；必须明确请求才会执行此检查。

如果因为 INSPECT 执行 INDEXDATA 检查时索引数据不一致而发现错误，那么会返回错误消息 SQL1141N。在返回此错误消息的同时，会收集数据诊断信息并将其转储至 db2diag 日志文件。还会在管理通知日志中记录紧急消息。使用 db2diag 日志文件分析工具（db2diag）来过滤并格式化 db2diag 日志文件的内容。

锁定影响

使用 INSPECT 命令和 INDEXDATA 选项检查索引与数据不一致问题时，仅以 IS 方式锁定被检查的表。

指定 INDEXDATA 选项后，缺省情况下只会使用明确指定级别子句选项的值。对于未明确指定的任何级别子句选项，缺省级别（INDEX NORMAL 和 DATA NORMAL）会从 NORMAL 改写为 NONE。

对 DB2 数据库系统的安装进行故障诊断

如果安装 DB2 数据库产品时发生问题，那么请确认您的系统符合安装要求，并查看常见安装问题列表。

过程

要对 DB2 数据库系统的安装问题进行故障诊断，请完成下列步骤:

- 确保系统符合所有安装要求。
- 如果遇到许可错误，请确保已应用适当的许可证。

  查看“知识集合: DB2 许可证问题”技术说明中的常见问题列表: http://www.ibm.com/support/docview.wss?rs=71&uid=swg21322757
- 查看文档和 DB2 技术支持 Web 站点提供的安装问题列表: www.ibm.com/software/data/db2/support/db2_9/troubleshoot.html

下一步任务

如果您完成上述步骤后仍无法确定问题所在，请开始收集诊断数据以获取更多信息。

收集关于安装问题的数据

如果您遇到安装问题并且不能确定问题的原因，请收集您或 IBM 软件支持机构可用来诊断和解决问题的诊断数据。

要收集关于安装问题的诊断数据:

1. 可选: 在启用跟踪的情况下重复安装尝试。例如:

   在 Linux 和 UNIX 操作系统上

   ```
   db2setup -t /filepath/trace.out
   ```

   在 Windows 操作系统上

   ```
   setup -t \filepath\trace.out
   ```

2. 找到安装日志文件。
   - 在 Windows 上，缺省文件名为“DB2-*ProductAbbreviation-DateTime*.log”。例如: DB2-ESE-Wed Jun 21 11_59_37 2006.log。安装日志的缺省位置为"我的文档"\DB2LOG\ 目录。
   - 在 Linux 和 UNIX 上，缺省文件名为 db2setup.log、db2setup.his 和 db2setup.err。

     如果在启用跟踪（或调试方式）后问题再次出现，那么可能会创建其他文件，例如: dascrt.log、dasdrop.log、dasupdt.log、db2icrt.log.*PID*、db2idrop.log.*PID*、db2iupgrade.log.*PID* 和 db2iupdt.log.*PID*，其中 *PID* 是进程标识。

     所有这些文件的缺省位置都是 /tmp 目录。对于跟踪文件（trace.out）而言，未指定目录时没有缺省目录，因此必须指定要在其中创建跟踪输出文件的文件夹的文件路径。
3. 可选: 如果您打算将数据提交给 IBM 软件支持机构，还请收集 DB2 数据。有关其他信息，请参阅“收集 DB2 数据”。

分析安装问题的数据

收集有关安装问题的诊断数据后，可分析此数据以确定产生问题的原因。这些步骤是可选的。如果不能顺利确定问题原因，请将数据提交给 IBM 软件支持机构。

这些步骤假定您以具有收集安装问题的数据中描述的文件。

1. 确保您正在查看相应的安装日志文件。请检查文件的创建日期或文件名中包含的时间戳记（在 Windows 操作系统上）。
2. 确定是否已成功完成安装。
   - 在 Windows 操作系统上，安装日志文件的底部出现类似以下的消息指示安装成功:

     ```
     Property(C): INSTALL_RESULT = 已成功完成安装
     === 日志记录停止时间: 6/21/2006  16:03:09 ===
     MSI (c) (34:38) [16:03:09:109]:
     产品: DB2 企业服务器版 - DB2COPY1 -- 已成功完成安装操作。
     ```

   - 在 Linux 和 UNIX 操作系统上，安装日志文件（缺省情况下为 db2setup.log）底部的消息指示安装成功。
3. 可选: 确定是否发生了错误。如果安装成功完成，但您在安装过程中接收到错误消息，那么在安装日志文件中找到这些错误。
   - 在 Windows 操作系统上，大多数错误都以“错误: ”或“警告: ”开头。例如:

     ```
     1: 错误: 运行命令"D:\IBM\SQLLIB\bin\db2.exe CREATE TOOLS CATALOG SYSTOOLS
     USE EXISTING DATABASE TOOLSDB FORCE"以启动和/或迁移 DB2 工具目录数据
     库时出现错误。
     ```

返回值为"4"。
1: 警告: 在此计算机上安装"DB2 企业服务器版 - DB2COPY1"时
发生了小错误。某些功能可能不会正确地起作用。

- 在 Linux 和 UNIX 操作系统上，如果 Java 返回了任何错误（如异常和陷阱信息），那么将显示缺省名为 db2setup.err 的文件。

如果您启用了安装跟踪，安装日志文件中将具有更多的条目且条目会更详细。

如果分析此数据不能帮您解决问题，且如果您与 IBM 软件支持机构签有维护合同，那么可发出问题报告。IBM 软件支持机构将让您提交收集的任何数据，并可能需要您执行的任何分析。

如果您进行调查后仍无法解决问题，请将数据提交给 IBM 软件支持机构。

已知问题与解决方案

以非 root 用户身份将 DB2 数据库产品安装到系统 WPAR 上的缺省路径中时发生错误（AIX）

在 AIX 6.1 上，如果以非 root 用户身份将 DB2 数据库产品安装在系统工作负载分区（WPAR）上的缺省安装路径（/opt/IBM/db2/V9.7）中，那么可能发生各种错误。要避免这些问题，将 DB2 数据库产品安装在只有 WPAR 能访问的文件系统上。

症状

如果将 DB2 数据库产品安装在系统 WPAR 上的 /usr 或 /opt 目录中，那么可能会发生各种错误，这取决于您配置目录的方式。可以将系统 WPAR 配置为与全局环境共享 /usr 和 /opt 目录（在这种情况下，可以从 WPAR 中对 /usr 和 /opt 目录进行读访问，但不能进行写访问），或者可以将该系统配置为具有 /usr 和 /opt 目录的本地副本。

在第一种情况下，如果 DB2 数据库产品安装在全局环境中的缺省路径中，那么可以在系统 WPAR 中看到该安装。这使得看起来像 DB2 安装在 WPAR 上，但尝试创建 DB2 实例将导致以下错误：DBI1288E 执行程序 db2icrt 失败。由于您没有对目录或文件 /opt/IBM/db2/V9.7/profiles.reg 和 /opt/IBM/db2/V9.7/default.env 的写许可权，此程序失败。

在第二种情况下，如果 DB2 数据库产品安装在全局环境中的缺省路径中，那么当 WPAR 创建 /usr 和 /opt 目录的本地副本时，还将复制 DB2 数据库产品安装。如果系统管理员尝试使用数据库系统，那么这可能会导致意外问题。由于 DB2 数据库产品旨在用于另一个系统，因此可能会复制不准确的信息。例如，最初在全局环境中创建的所有 DB2 实例似乎都出现在 WPAR 中。这会使系统管理员弄不清楚系统上实际安装了哪些实例。

原因

这些问题是由于将 DB2 数据库产品安装在系统 WPAR 上的 /usr 或 /opt 目录中导致的。

解决问题

不要将 DB2 数据库产品安装在全局环境中的缺省路径中。

安装只有 WPAR 能访问的文件系统并在该文件系统上安装 DB2 数据库产品。

DB2 数据库产品的 Beta 版和非 Beta 版不能共存

DB2 副本可以包含一个或多个不同的 DB2 数据库产品，但不能同时包含 Beta 版和非 Beta 版产品。请不要将 Beta 版和非 Beta 版的 DB2 数据库产品安装在同一个位置。

此限制同时适用于 DB2 数据库产品的客户机和服务器组件。

解决问题

在安装非 Beta 版之前，请卸载 Beta 版的 DB2 版本 9.7，或者选择另一个安装路径。

安装 DB2 数据库产品时解决服务名称错误

如果您选择非缺省服务名称或端口号供 DB2 数据库产品或 DB2 信息中心使用，请确保不要指定已在使用中的值。

症状

当您尝试安装 DB2 数据库产品或*DB2 信息中心*时，DB2 安装向导报告了一个错误，指出“指定的服务名称在使用中”。

原因

当您安装下列产品时，DB2 安装向导将提示您选择端口号和服务名称:

- *DB2 信息中心*
- 接受来自客户机的 TCP/IP 通信的 DB2 数据库产品
- 将作为数据库分区服务器工作的 DB2 数据库产品

如果您选择服务名称和端口号，而不是接受缺省值，那么可能会发生此错误。如果您选择系统上的 `services` 文件中已存在的服务名称，并且只更改了端口号，那么将发生此错误。

解决问题

执行下列其中一项操作:

- 使用缺省值。
- 使用已同时存在于 `services` 文件中的服务名称和端口号。
- 在 `services` 文件中添加未使用的服务名称和未使用的端口号。请在 DB2 安装向导中指定这些值。

对许可证问题进行故障诊断

分析 DB2 许可证一致性报告

要验证 DB2 功能部件的许可证一致性，请分析 DB2 许可证一致性报告。如果存在任何许可证发放违例，那么可通过获取适当的许可证密钥或者除去问题 DB2 数据库产品或功能部件来解决这些违例问题。

开始前

以下步骤假定您使用了许可证中心或 db2licm 命令来生成 DB2 许可证一致性报告。

过程

1. 打开包含 DB2 许可证一致性报告的文件。
2. 在一致性报告中检查每个 DB2 功能部件的状态。 该报告针对每个功能部件显示下列其中一个值:

   一致 指示未检测到任何违例。该功能部件已使用且得到正确授权。

   未使用 指示您尚未执行任何需要此特定功能部件的活动。

   违例 指示该功能部件未得到授权，但已使用。
3. 如果存在任何违例，请使用许可证中心或 db2licm -l 命令来查看您的许可证信息。

   如果随"未获得授权"状态列示了 DB2 功能部件，那么您必须获取该功能部件的许可证。用于注册该功能部件的许可证密钥和指示信息位于您购买 DB2 功能部件时收到的激活 CD 上。

   一些 DB2 功能部件使用的是软停止策略；也就是说，即使发生违例，该功能部件也将继续工作，让您有时间来获取并应用许可证密钥。另一些功能部件使用的是硬停止策略，在发生违例的情况下，功能部件将停止工作。

   注: 在 DB2 工作组服务器版 和 DB2 Express™版 上，SAMPLE 数据库包含具体化查询表（MQT）和多维集群表（MDC），这将导致许可证违例。只能通过升级到 DB2 企业服务器版 来消除此违例情况。
4. 如果选择废弃或删除问题对象，而不是购买许可证，请使用以下命令来确定 DB2 数据库产品中的哪些对象或设置会导致许可证违例:
   - 对于 DB2 高级访问控制功能部件:

     检查使用了基于标号的访问控制（LBAC）的表。针对 DB2 副本中每个实例的每个数据库运行以下命令:

     ```
     SELECT TABSCHEMA, TABNAME
     FROM SYSCAT.TABLES
     WHERE SECPOLICYID>0
     ```
   - 对于 DB2 性能优化功能部件:
     - 检查是否存在任何具体化查询表。针对 DB2 副本中每个实例的每个数据库运行以下命令:

       ```
       SELECT OWNER, TABNAME
       FROM SYSCAT.TABLES WHERE TYPE='S'
       ```
     - 检查是否存在任何多维集群表。针对 DB2 副本中每个实例的每个数据库运行以下命令:

```
SELECT A.TABSCHEMA, A.TABNAME, A.INDNAME, A.INDSCHEMA
FROM SYSCAT.INDEXES A, SYSCAT.TABLES B
WHERE (A.TABNAME=B.TABNAME AND A.TABSCHEMA=B.TABSCHEMA)
AND A.INDEXTYPE='BLOK'
```

– 检查您的任何实例是否使用了查询并行性（也称为*查询内并行性*）。针对 DB2 副本中的每个实例运行以下命令一次:

```
SELECT NAME, VALUE
FROM SYSIBMADM.DBMCFG
WHERE NAME IN ('intra_parallel')
```

- 对于 DB2 存储器优化功能部件:

  检查是否有任何表启用了行级别压缩。针对 DB2 副本中每个实例的每个数据库运行以下命令:

```
SELECT TABSCHEMA, TABNAME
FROM SYSCAT.TABLES
WHERE COMPRESSION IN ('R', 'B')
```

下一步任务

一旦解决了违例（通过获取功能部件的许可证或除去发生违例的源），就可以通过从许可证中心或发出命令

`db2licm -x` 来重置许可证一致性报告。

故障诊断优化准则和概要文件

通过说明表提供了对优化准则的诊断支持（由优化概要文件传递）。

如果优化器未应用优化准则，您将接收到 SQL0437W 警告，原因码为 13。详细说明未应用优化准则的原因的诊断信息将被添加到说明表中。有两个说明表用于接收优化器诊断输出:

- EXPLAIN_DIAGNOSTIC – 此表中的每个条目都表示一条与特定语句的优化相关的诊断消息。每条诊断消息都使用一个数字代码表示。
- EXPLAIN_DIAGNOSTIC_DATA – 此表中的每个条目都是与 EXPLAIN_DIAGNOSTIC 表中的特定诊断消息相关的诊断数据。

用于创建诊断说明表的 DDL 显示在下面的第 439 页的图 39 中。

以下步骤可以帮助您诊断使用优化准则时出现的问题:

1. *《故障诊断和调整数据库性能》*中的“验证是否已使用优化准则”。
2. 使用内置 *Administrative Routines and Views*中的“EXPLAIN_GET_MSGS 表函数”来检查完整错误消息。

如果完成上述步骤后仍无法确定问题根源，请开始收集诊断数据并考虑与 IBM 软件支持机构联系。

```
CREATE TABLE EXPLAIN_DIAGNOSTIC
      ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
        EXPLAIN_TIME      TIMESTAMP    NOT NULL,
        SOURCE_NAME       VARCHAR(128) NOT NULL,
        SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
        SOURCE_VERSION    VARCHAR(64)  NOT NULL,
        EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
        STMTNO            INTEGER      NOT NULL,
        SECTNO            INTEGER      NOT NULL,
        DIAGNOSTIC_ID     INTEGER      NOT NULL,
        CODE              INTEGER      NOT NULL,
        PRIMARY KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO,
                     DIAGNOSTIC_ID),
        FOREIGN KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO)
        REFERENCES EXPLAIN_STATEMENT ON DELETE CASCADE);

   CREATE TABLE EXPLAIN_DIAGNOSTIC_DATA
      ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
        EXPLAIN_TIME      TIMESTAMP    NOT NULL,
        SOURCE_NAME       VARCHAR(128) NOT NULL,
        SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
        SOURCE_VERSION    VARCHAR(64)  NOT NULL,
        EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
        STMTNO            INTEGER      NOT NULL,
        SECTNO            INTEGER      NOT NULL,
        DIAGNOSTIC_ID     INTEGER      NOT NULL,
        ORDINAL           INTEGER      NOT NULL,
        TOKEN             VARCHAR(1000),
        TOKEN_LONG        BLOB(3M) NOT LOGGED,
        FOREIGN KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO,
                     DIAGNOSTIC_ID)
        REFERENCES EXPLAIN_DIAGNOSTIC ON DELETE CASCADE);
```

注: EXPLAIN_REQUESTOR、EXPLAIN_TIME、SOURCE_NAME、SOURCE_SCHEMA、SOURCE_VERSION、EXPLAIN_LEVEL、STMTNO 和 SECTNO 列同时包括在两个表中，以形成 EXPLAIN_STATEMENT 表的外键及 EXPLAIN_DIAGNOSTIC 与 EXPLAIN_DIAGNOSTIC_DATA 之间的父子关系。

图 39. 用于创建诊断说明表的 DDL

此 DDL 包括在 `sqllib` 目录的 `misc` 子目录下的 EXPLAIN.DDL 文件中。

对分区数据库环境进行故障诊断

与 127.0.0.2 有关的 FCM 问题（Linux 和 UNIX）

在分区数据库环境中，如果 /etc/hosts 文件中有对应 127.0.0.2 的条目，那么快速通信管理器（FCM）会遇到问题。

症状

根据情况，可能会出现不同的错误消息。例如，创建数据库时可能会发生以下错误: SQL1229N 当前事务已回滚，因为发生了系统错误。SQLSTATE=40504

原因

jk该问题是由于 /etc/hosts 文件中存在对应 IP 地址 127.0.0.2 的条目而导致的，其中 127.0.0.2 映射至机器的标准主机名。例如:

```
127.0.0.2 ServerA.ibm.com ServerA
```

其中“ServerA.ibm.com”是标准主机名。

环境

该问题仅限于带有DB2 数据库分区功能部件的DB2 企业服务器版。

解决问题

从 /etc/hosts 文件中除去该条目，或者将其转换为注释。例如:

```
# 127.0.0.2 ServerA.ibm.com ServerA
```

在加密文件系统上创建数据库分区（AIX）

AIX 6.1 支持加密 JFS2 文件系统或文件集的功能。DB2 数据库产品中的分区数据库环境不支持此功能。如果尝试在 AIX 上使用 EFS（加密文件系统）创建分区数据库环境，那么将会发生 SQL10004C 错误。

症状

如果尝试在多分区数据库环境中的加密文件系统上创建数据库，那么您将接收到以下错误: SQL10004C 在访问数据库目录时发生 I/O 错误。SQLSTATE=58031

原因

此时不能在 AIX 上使用 EFS（加密文件系统）创建分区数据库环境。由于分区数据库的分区使用 rsh 或 ssh，因此 EFS 中的密钥库将丢失，并且数据库分区无法访问存储在加密文件系统上的数据库文件。

诊断问题

DB2 诊断（db2diag）日志文件将包含错误消息和以下文本: OSERR : ENOATTR (112)" 找不到属性"。

解决问题

要在分区数据库环境中成功创建数据库，必须具有可用于所有机器的文件系统，并且该文件系统不能是加密文件系统。

对脚本进行故障诊断

您可能会有一些内部工具或脚本以数据库引擎中运行的进程为基础。因为所有代理程序、预取程序和页面清除程序现在被视为单个多线程进程内的线程，所以这些工具或脚本可能不再生效。

内部工具和脚本必须进行修改才能用于线程化进程。例如，某些脚本可能会启动 ps 命令来列示进程名称；然后对特定代理进程执行任务。必须重新编写这些脚本。

问题确定数据库命令 db2pd 会有一个新选项 -edu（“引擎可分派单元”的缩写），用于列示所有代理程序名称及其线程标识。db2pd -stack 命令继续与线程化引擎配合使用，以转储各个 EDU 堆栈或当前节点的所有 EDU 堆栈。

在应用修订包 1 之后重新编译静态部分以收集部分实际值

在应用 DB2 版本 9.7 修订包 1 之后，无法为在应用此修订包之前所编译的静态部分收集部分实际值。在应用修订包 1 之后，必须重新编译静态部分以收集部分实际值。

症状

执行 EXPLAIN_FROM_ACTIVITY 例程时未收集部分实际值。

原因

无法为在应用此修订包之前编译的静态部分收集部分实际值。

解决问题

在安装 DB2 V9.7 修订包 1 之后，验证自从应用了此修订包以来是否已经使用 REBIND 命令重新绑定了静态部分。为此，请检查 SYSCAT.PACKAGES 目录视图中的 LAST_BIND_TIME 列。

对存储器密钥支持进行故障诊断

存储器保护密钥（线程级的硬件密钥）用于避免无效内存访问尝试，以使 DB2 引擎具有更大的弹性。要解决启用此功能时遇到的错误，请按照下一节中的指示信息执行操作。

诊断注册表变量错误

设置注册表变量 *《数据库管理概念和配置参考》*中的『DB2_MEMORY_PROTECT』时，返回了值无效（DBI1301E）错误。因为以下原因之一发生了此错误:

- 对注册表变量给定的值无效。请参阅 **DB2_MEMORY_PROTECT** 的注册表变量用法，以了解有关有效值的信息。

- 硬件和操作系统可能不支持存储器保护密钥，因此无法启用此功能。存储器保护密钥在 POWER6™ 处理器中可用并且受具有 5300-06 技术级别的 AIX 5L™ V5.3 操作系统的支持。

第 7 章 对 DB2 Connect 进行故障诊断

DB2 Connect 环境涉及到多个软件、硬件和通信产品。通过排除和提炼可用的数据以得出结论（找出发生错误的位置）是故障诊断的最佳方法。

在收集相关信息之后，根据您选择的适当主题，转到可参考的章节。

诊断工具

当您遇到问题时，可以使用下列工具:

- 包括转储文件、陷阱文件、错误日志、通知文件和警报日志在内的所有诊断数据都在诊断数据目录路径（**diagpath**）数据库管理器配置参数指定的路径中:

  如果此配置参数的值为 null，那么诊断数据将写入下列目录或文件夹之一:

  – 对于 Linux 和 UNIX 环境: INSTHOME/sqllib/db2dump，其中 *INSTHOME* 是实例的主目录。

  – 对于受支持的 Windows 环境:

    - 如果未设置 **DB2INSTPROF** 环境变量，那么将使用 x:\SQLLIB\DB2INSTANCE，其中 x:\SQLLIB 是驱动器引用和在 **DB2PATH** 注册表变量中指定的目录，而 **DB2INSTANCE** 的值为实例的名称。

      注: 该目录没必要命名为 SQLLIB。

    - 如果设置了 **DB2INSTPROF** 环境变量，那么将使用 x:\DB2INSTPROF\DB2INSTANCE，其中 **DB2INSTPROF** 是实例概要文件目录的名称，**DB2INSTANCE** 是实例的名称（缺省情况下，在 Windows 32 位操作系统上为 **DB2INSTDEF** 的值）。

- 对于 Windows 操作系统，可以使用“事件查看器”来查看管理通知日志。
- 可以使用的有效诊断工具包括 **db2trc**、**db2pd**、**db2support** 和 **db2diag**
- 对于 Linux 和 UNIX 操作系统，**ps** 命令将关于活动进程的进程状态信息返回到标准输出中。
- 对于 UNIX 操作系统，核心文件是在发生服务器错误时，在当前目录中创建的。它包含已终止的进程的内存映像，可用来确定哪个功能导致该错误。

收集相关信息

故障诊断包括缩小问题的范围和调查可能的原因。正确的出发点是收集相关信息，确定您知道哪些信息，哪些数据尚未收集，以及可以排除哪些途径。至少要回答下列问题:

- 最初的连接已经成功了吗？
- 硬件运行正常吗？
- 通信路径正常吗？
- 有任何通信网络更改会使先前的目录条目无效吗？
- 已经启动数据库了吗？

© Copyright IBM Corp. 2006, 2009

- 一台或多台客户机与 DB2 Connect 服务器（网关）之间的通信、DB2 Connect 网关与 IBM 大型机数据库服务器之间的通信或 DB2 Connect 个人版与 IBM 大型机数据库服务器之间的通信中断了吗？
- 利用消息中所返回的消息内容和标记可以确定哪些问题？
- 此时将使用诊断工具（例如，db2trc、db2pd 或 db2support）来提供任何帮助吗？
- 执行类似任务的其他机器运行正确吗？
- 如果这是一个远程任务，那么在本地能成功执行它吗？

初始连接不成功

复查下列问题并确保已遵循了安装步骤:

1. *成功完成了安装过程吗？*
   - 所有必备软件产品都可用吗？
   - 内存和磁盘空间足够用吗？
   - 安装了远程客户机支持吗 ？
   - 完成了通信软件的安装并且没有任何错误情况吗？
2. *对于 UNIX 操作系统，创建了产品的实例吗？*
   - 作为 root 用户，您创建了将作为实例所有者和 sysadm 组的用户和组了吗？
3. *成功地处理了许可证信息吗？（如果适用的话）*
   - 对于 UNIX 操作系统，您编辑了节点锁定文件并输入了由 IBM 提供的密码吗？
4. *正确地配置了 IBM 大型机数据库服务器与工作站之间的通信吗？*
   - 必须考虑的三个配置:
     a. IBM 大型机数据库服务器配置向服务器标识应用程序请求器。IBM 大型机服务器数据库管理系统将具有系统目录条目，这些条目会根据位置、网络协议和安全性来定义请求器。
     b. DB2 Connect 工作站配置向服务器定义客户机成员并向客户机定义 IBM 大型机服务器。
     c. 客户机工作站配置必须定义工作站的名称和通信协议。
   - 关于未执行最初连接的问题分析包括: 验证 PU（物理单元）名称都是完整的且正确的，或者，对于 TCP/IP 连接，验证是否已经指定了正确的端口号和主机名。
   - IBM 大型机服务器数据库管理员和“网络”管理员都有可用来诊断问题的实用程序。
5. *您具有 IBM 大型机服务器数据库管理系统在使用 IBM 大型机服务器数据库时需要的权限级别吗？*
   - 考虑用户的访问权限、表限定符的规则以及预期的结果。
6. *尝试使用命令行处理器（CLP）来对 IBM 大型机数据库服务器发出 SQL 语句未成功吗？*
   - 遵循了将 CLP 绑定至 IBM 大型机数据库服务器的过程吗？

初始连接后遇到的问题

提供下列问题作为起始点，以帮助缩小问题的范围。

1. *有任何特殊的或不常见的操作环境吗？*
   - 这是新应用程序吗？
   - 正在使用新过程吗？
   - 最近是否执行了可能会影响系统的更改？例如，自从上次成功地运行了应用程序或方案之后，是否更改了某些软件产品或应用程序？
   - 对于应用程序，是使用哪个应用程序编程接口（API）来创建该应用程序的？
   - 是否有使用该软件或通信 API 的其他应用程序在用户系统上运行？
   - 最近安装了修订包吗？如果当用户试图在他们的操作系统上使用（或装入）一个自从安装以来从未使用过的功能部件时发生问题，那么应确定 IBM 的最新修订包，并在安装该功能部件后装入该修订包。
2. *以前发生过此错误吗？*
   - 是否有关于先前错误状态的已记录的解决方案？
   - 谁是参与者？他们能否提供对可能的操作方法的深入见解？
3. *您探索过使用通信软件命令来返回有关网络的信息吗？*
   - TCP/IP 可能会在使用 TCP/IP 命令和守护程序时检索到有用的信息。
4. *在 SQLCA（SQL 通信区）中是否返回了有帮助的信息？*
   - 问题处理过程应该包括检查 SQLCODE 和 SQLSTATE 字段内容的步骤。
   - SQLSTATE 允许应用程序员对 DB2 系列数据库产品的常见错误类进行测试。在分布式关系数据库网络中，此字段可以提供一个公共基础。
5. *在服务器中执行了 START DBM 吗？*另外，对于访问远程服务器的客户机，要确保正确地设置了 DB2COMM 环境变量。
6. *执行同一任务的其他机器能够成功与服务器相连吗？*试图与服务器相连的客户机数目可能已经达到了最大值。如果另一个客户机与服务器断开连接，那么先前不能连接的客户机现在能连接吗？
7. *机器是否有正确的地址？*验证该机器在网络中是否是唯一的。
8. *当远程连接时，已经为客户机授予了正确的权限吗？*可能成功与实例进行了连接，但是，可能未在数据库级别或表级别授予权限。
9. *这是第一台与远程数据库进行连接的机器吗？*在分布式环境中，网络之间的路由器或桥接器可能会阻塞客户机与服务器之间的通信。例如，当使用 TCP/IP 时，应确保可以对远程主机执行 PING。

不受支持的 DDM 命令

当 DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 充当 DRDA 应用程序服务器（DRDA AS）时，不支持 DDM 命令 BNDCPY、BNDDPLY、DRPPKG 和 DSCRDBTBL。

症状

如果 DRDA 应用程序请求器（DRDA AR）连接到 DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 且发出以下任何命令，那么该命令将失败:

表 81. 不受支持的 DDM 命令

DDM 命令	DDM 代码点	描述
BNDCPY	X'2011'	复制现有的关系数据库（RDB）包
BNDDPLY	X'2016'	部署现有的 RDB 包
DRPPKG	X'2007'	删除包
DSCRDBTBL	X'2012'	描述 RDB 表

另外，也不支持在单参数式（或单列式）数组输入的 SQLDTA 描述符中使用的以下代码点:

表 82. 不受支持的 FD:OCA 数据对象

FD:OCA 数据对象	DDM 代码点	描述
FDOEXT	X'147B'	格式化数据对象内容体系结构（FD:OCA）数据区间
FDOOFF	X'147D'	FD:OCA 数据位移

在这种情况下，最常见的错误消息是 SQL30020N（"由于分布式协议错误，执行失败，该错误将影响后续命令和 SQL 语句的成功执行"）。

原因

分布式数据管理体系结构（DDM）是 DRDA 协议的一部分。DDM 命令 BNDCPY、BNDDPLY、DRPPKG 和 DSCRDBTBL 存在于 DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 所支持的所有 DRDA 级别中，但 DRDA 应用程序服务器不支持这些 DDM 命令。

类似，DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 DRDA 应用程序服务器不支持 **FDOEXT** 和 **FDOOFF** 代码点。当您提交单列式数组输入请求时，在发送到服务器的 SQLDTA 描述符中使用这些代码点。

诊断问题

如果在 DRDA 应用程序服务器上获取 DB2 跟踪，那么将看到一条类似如下的对这些命令作出响应的消息: 错误消息 = 解析器: 命令不受支持。

解决问题

BNDCPY 和 BNDDPLY DDM 命令当前不存在任何受支持的替代命令。

要删除包，请使用 SQL 语句 DROP PACKAGE。例如，连接到 DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 DRDA 应用程序服务器并在 EXECUTE IMMEDIATE 请求中发送 DROP PACKAGE 语句。DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 将成功处理该请求。

要描述 RDB 表，请使用下列其中一个 DDM 命令: DSCSQLSTT（描述 SQL 语句）或 PRPSQLSTT（预编译 SQL 语句）。例如，如果需要表 TAB1 的描述，请描述或预编译以下语句: `SELECT * FROM TAB1`。

注: 当 DRDA AR 发出 PRPSQLSTT 命令时，必须同时指定带有值 TRUE 的实例变量 **RTNSQLDA**，否则服务器将不返回“SQLDA 应答数据”（SQLDARD）描述符。

要避免与 **FDOEXT** 和 **FDOOFF** 代码点相关的问题，请使用单行式数组输入请求取代单参数式（或单列式）数组输入请求。

DB2 Connect 常见问题

本主题列示使用 DB2 Connect 时遇到的常见的连接问题症状。在每种情况下，都为您提供:

- 消息号和与该消息相关联的返回码（或特定于协议的返回码）的组合。 每个消息和返回码组合都具有独立的标题，这些标题是先按消息号，再按返回码来排序的。
- 症状，通常采用样本消息列表的格式。
- 建议的解决方案，指示错误的可能原因。在某些情况下，可能会提供多种建议的解决方案。

SQL0965 或 SQL0969

症状 可发出消息 SQL0965 和 SQL0969，并且返回来自 DB2 IBM i 版、DB2 z/OS 版和 DB2 服务器 VM 和 VSE 版的许多不同返回码。

当您遇到任何其中一个消息时，应该在发出该消息的数据库服务器产品的文档中查找原始 SQL 代码。

解决方案

无法转换从 IBM 大型机数据库接收到的 SQL 代码。根据错误代码来更正该问题，然后重新提交失败的命令。

SQL5043N

症状 未能成功启动对一个或多个通信协议的支持。但是，已成功启动了核心数据库管理器功能。

DB2 Connect 服务器上可能未启动 TCP/IP 协议。先前可能已经有成功的客户机连接。

如果 `diaglevel = 4`，那么 db2diag 日志文件可能包含类似的条目，例如:

```
2001-05-30-14.09.55.321092   Instance:svtdbm5   Node:000
   PID:10296(db2tcpcm)   Appid:none
   common_communication  sqlcctcpconnmgr_child   Probe:46
   DIA3205E 在 TCP/IP services 文件中配置的且 TCP/IP
服务器支持所需的套接字地址"30090"正被另一进程使用。
```

解决方案

此警告是一种症状，它表示充当远程客户机的服务器的 DB2 Connect 在处理一个或多个客户机通信协议时遇到问题。这些协议可以是 TCP/IP 和其他协议，消息通常会指出为 DB2 Connect 定义的某个通信协议未正确配置。

原因通常是未定义 DB2COMM 概要文件变量，或者该变量未正确定义。该问题通常是 DB2COMM 变量与数据库管理器配置中定义的名称（例如，svcename 或 nname）之间不匹配的结果。

一种可能的情况是先前有成功的连接，然后得到 SQL5043 错误消息，但未更改任何配置。使用 TCP/IP 协议时，当远程系统因为某些原因而异常终止了连接，

就可能会发生这种情况。发生这种情况时，客户机上可能仍然显示连接存在，通过发出下面所显示的命令，就可复原连接，而无须进一步的操作。

与 DB2 Connect 服务器相连接的某个客户机很有可能在 TCP/IP 端口上仍有句柄。在与 DB2 Connect 服务器相连接的每一台客户机上，输入下列命令:

```
db2 terminate      db2stop
```

SQL30020

症状 SQL30020N 执行失败，原因是“分布式协议错误”，该错误将影响后续命令和 SQL 语句的成功执行。

解决方案

遇到此错误时，应该与服务中心联系。与服务中心联系之前请先运行 db2support 命令。

SQL30060

症状 SQL30060N *"<authorization-ID>"* 没有执行操作 *"<operation>"* 的特权。

解决方案

当连接至 DB2 z/OS 版时，尚未正确地更新“通信数据库”（CDB）表。

SQL30061

症状 连接至错误的 IBM 大型机数据库服务器位置 - 找不到任何目标数据库。

解决方案

可能在 DCS 目录条目中指定了错误的服务器数据库名称。当发生此情况时，将对应用程序返回 SQLCODE -30061。

检查 DB2 节点、数据库和 DCS 目录条目。DCS 目录条目中的目标数据库名称字段必须与基于平台的数据库名称相对应。例如，对于 DB2 z/OS 版数据库，要使用的名称应该与“引导数据集”（BSDS）“LOCATION=*locname*”字段中的名称相同，当启动了“分布式数据设施”（DDF）时，在 DSNL004I 消息（LOCATION=*location*）中也提供了该名称。

TCP/IP 节点的正确命令是:

```
db2 catalog tcpip node <node_name> remote <host_name_or_address>
            server <port_no_or_service_name>
db2 catalog dcs database <local_name> as <real_db_name>
db2 catalog database <local_name> as <alias> at <node node_name>
                     authentication server
```

要连接至数据库，您应该发出:

```
db2 connect to <alias> user <user_name> using <password>
```

带有返回码 79 的 SQL30081N

症状

```
SQL30081N  已检测到通信错误。
   所使用的通信协议: "TCP/IP"。
 所使用的通信 API: "SOCKETS"。
位置    检测到错误的位置: ""。   错误的
      通信功能:
"连接"。特定于协议的错误代码: "79"、"*"和"*"。
   SQLSTATE=08001
```

解决方案

当远程客户机未能与 DB2 Connect 服务器连接时，就可能发生此错误。当从 DB2 Connect 服务器连接至 IBM 大型机数据库服务器时，也可能发生此错误。

1. 在 DB2 Connect 服务器上，DB2COMM 概要文件变量可能设置得不正确。检查此变量。例如，当在 AIX 上运行 DB2 企业服务器版时，命令 db2set db2comm=tcpip 应该出现在 sqllib/db2profile 中。
2. IBM 数据服务器客户机 中指定的 TCP/IP 服务名称和端口号可能与 DB2 Connect 服务器中指定的不匹配。在以上两种机器中验证 TCP/IP services 文件中的各个条目。
3. 检查在 DB2 Connect 服务器上是否启动了 DB2。使用以下命令将“数据库管理器配置”diaglevel 设置为 4:

   ```
   db2 update dbm cfg using diaglevel 4
   ```

   在停止并重新启动 DB2 后，查看 db2diag 日志文件，以检查是否已经启动 DB2 TCP/IP 通信。您应该看到与下列信息类似的输出:

   ```
   2001-02-03-12.41.04.861119   Instance:svtdbm2   Node:00
   PID:86496(db2sysc)   Appid:none
   common_communication  sqlcctcp_start_listen   Probe:80
   DIA3000I "TCPIP"协议支持已成功启动。
   ```

带有特定于协议的错误代码 10032 的 SQL30081N

症状

```
   SQL30081N  已检测到通信错误。
      所使用的通信协议: "TCP/IP"。
   所使用的通信 API: "SOCKETS"。
位置    检测到错误的位置: "9.21.85.159"。   检测到
        错误的通信功能:
      "发送"。特定于协议的错误代码: "10032"、"*", "*".
      SQLSTATE=08001
```

解决方案

当试图与 TCP/IP 通信已经发生故障的机器断开连接时，可能会接收到此错误消息。用 TCP/IP 子系统更正该问题。

在大多数机器上，只须对机器重新启动 TCP/IP 协议就可以更正该问题。有时可能需要重新启动整个机器。

连接（CONNECT）期间出现 SQL30082 RC=24

症状 SQLCODE -30082 提供的用户名或密码不正确。

解决方案

确保在 CONNECT 语句上提供了正确的密码（必要时）。未提供要发送到目标服务器数据库的密码。必须将密码从 IBM 数据服务器客户机发送到目标服务器数据库中。在某些平台上，例如 AIX，仅当密码是在 CONNECT 语句中提供的时候，才能获得该密码。

第 8 章 搜索知识库

如何有效地搜索已知问题

有许多资源可用于描述已知问题，包括 DB2 APAR、白皮书、IBM Redbooks®（红皮书）出版物、技术说明和手册。所以，有效地搜索这些（及其他）资源以迅速确定您遇到的问题是否已有解决方案是非常重要的。

在搜索之前，应清楚地了解问题的情况。

在清楚地了解问题的情况之后，需要创建搜索关键字列表以增加找到现有解决方案的机率。以下是一些技巧:

1. 在搜索中使用多个字词。您使用的搜索项越切合，搜索结果越准确。
2. 从特定结果开始，然后在必要时扩大搜索结果的范围。例如，如果返回的结果太少，那么除去一些不太切合的搜索项，然后重试。或者，如果不确定要使用的关键字，可使用若干关键字进行广泛搜索，查看接收到的搜索结果类型，您就能够更准确地选择其他关键字。
3. 有时搜索特定短语会更有效。例如，如果输入 ″administration notification file″（加上引号），那么只会得到包含内容与您的输入内容和顺序完全相同的短语。（相对于包含这三个单词的任何组合的所有文档）。
4. 使用通配符。如果遇到特定 SQL 错误，那么搜索“SQL5005<wildcard>”，其中 <wildcard> 是您要搜索的资源的相应通配符。这比只搜索“SQL5005”或“SQL5005c ”返回的结果要多。
5. 如果遇到实例异常结束并且生成陷阱文件的情况，那么使用陷阱或核心文件的堆栈回溯中的前两个或前三个函数来搜索已知问题。如果返回的结果太多，那么尝试添加关键字“trap”、“abend”或“crash”。
6. 如果搜索特定于操作系统的关键字（如信号编号或错误值），那么尝试针对常量名称而不是值进行搜索。例如，搜索“EFGIG”而不搜索错误号 27。

一般情况下，成功搜索到的搜索项通常包括:

- 描述所运行命令的词汇
- 描述症状的词汇
- 诊断中的标记

故障诊断资源

提供有大量故障诊断信息，可帮助您使用 DB2 数据库产品。

DB2 文档

您可以在 *DB2 信息中心*以及构成 DB2 资料库的 PDF 书籍中找到故障诊断信息。

© Copyright IBM Corp. 2006, 2009

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）、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

访问 DB2 技术支持 Web 站点: www.ibm.com/software/data/db2/support/db2_9/

第 9 章 获取 DB2 产品修订

修订包中包含适用于 IBM 在产品测试期间发现的问题以及客户使用产品时发现的问题的代码更新和修订。将讨论如何找到最新修订包以及如何对数据库环境应用这些修订。

获取修订

IBM 可能已提供用于解决您所遇到的问题的产品修订。您可以通过执行下列步骤来获取修订。

1. 您可以分别在下列 Web 页面上查看修订列表和获取修订包:
   - DB2 9 Linux 版、UNIX 版和 Windows 版支持
   - DB2 Linux 版、UNIX 版和 Windows 版的修订（按版本排列）
2. 确定所需的修订包。通常，建议您安装最新的修订包，以避免遇到由已更正的已知软件缺陷引起的问题。
3. 下载修订包，然后通过双击自解压缩的可执行程序包将文件解压缩。打开 `SERVER/doc/your_language/readme.txt` 文档并顺着提供的链接转到 DB2 信息中心，以获取安装指示信息。
4. 应用修订。有关指示信息，请参阅《*安装 DB2 服务器*》中的“应用修订包”。

修订包、临时修订包和测试修订

授权程序分析报告（APAR）是当前未更改发行版的 IBM 程序中可疑缺陷导致的问题的书面报告。APAR 描述 IBM 测试期间发现的问题以及客户报告的问题。

可在修订包、临时修订包和测试修订中交付解决了 APAR 中所描述问题的已修改 DB2 代码。

修订包 修订包是 APAR 修订的累积集合。修订包尤其说明 DB2 的新发行版之间出现的 APAR。它们允许您上移至特定维护级别。修订包具有以下特征:

- 它们是累积的。特定 DB2 发行版的修订包将替代或包含该发行版的先前修订包和临时修订包中提供的所有 APAR 修订。
- 它们适用于所有受支持的操作系统和 DB2 数据库产品。
- 它们包含多个 APAR。
- 它们发布在 DB2 技术支持 Web 站点上，且通常可供购买了符合 Passport Advantage® 程序的产品的客户使用。
- 它们已经过 IBM 的完整测试。
- 它们的配套文档描述了对数据库产品所作的更改并描述了如何安装和除去修订包。

注: 当 APAR 修订提供在修订包中时，APAR 的状态将从“Open”改为“Closed”。可通过检查 DB2 技术支持 Web 站点上的 APAR 描述来确定单个 APAR 的状态。

© Copyright IBM Corp. 2006, 2009

临时修订包

临时修订包是在修订包之间出现的重要 APAR 修订的累积集合。APAR 必须被视为很普遍或者在其他方面特别重要，才能包含在临时修订包中。候选 APAR 由 DB2 技术支持团队的专家进行评估与核准。临时修订包具有以下特征:

- 它们是累积的。特定 DB2 发行版的临时修订包将替代或包含该发行版的先前修订包和临时修订包中提供的所有 APAR 修订。
- 它们适用于一部分操作系统和 DB2 数据库产品。
- 它们通常包含 20 到 30 个新 APAR。
- 它们发布在 DB2 技术支持 Web 站点上，且通常可供购买了符合 Passport Advantage 程序的产品的客户使用。
- 它们已经过 IBM 的完整测试。
- 它们的配套文档描述了如何安装和除去临时修订包。

临时修订包在发布后可在生产中使用两年。它们大约在两个修订包之间的中间推出，并预期作为测试修订的首选替代选择，这些测试修订未获得临时修订包所获得的测试级别或未享有临时修订包所享有的支持级别。

测试修订

测试修订是针对报告的问题提供给特定用户进行测试的临时解决方案。测试修订有时被称为“特殊构建”且具有以下特征:

- 它们通常包含单个 APAR。
- 它们通过 DB2 支持机构获得且通常不对公众开放。
- 它们经过了 IBM 的有限测试。
- 它们包含最少的记录，包括应如何应用测试修订的描述、修订的 APAR 以及除去测试修订的指示信息。

尚未解决新问题、问题没有变通方法或无法等到下一修订包或临时修订包可用时，就会提供测试修订。例如，如果问题导致对业务产生极大影响，将提供测试修订以缓解这种状况直到在修订包或临时修订包中处理了该 APAR。

建议使 DB2 环境始终在最新修订包级别运行，以确保操作不会出现问题。要接收可用的新修订包的通知，请向 DB2 技术支持 Web 站点（http://www.ibm.com/software/data/db2/support/db2_9/）上的“My notifications”电子邮件更新进行预订。

要了解有关 DB2 修订和修订包的角色和用途方面的更多信息，请参阅支持策略声明。

应用测试修订

测试修订是针对报告的问题提供给特定用户进行测试的临时修订。每个测试修订都有自述文件。测试修订自述文件提供了有关安装和卸载该测试修订的指示信息以及包括在测试修订中的 APAR 列表（如果存在）。

每个测试修订都有特定的先决条件。有关详细信息，请参阅该测试修订附带提供的自述文件。

有两种类型的测试修订:

- 单个 DB2 产品的测试修订。这些测试修订可应用于该产品的现有安装，也可用在未安装 DB2 的情况下执行完整产品安装。

- 通用测试修订（仅适用于 Linux 和 UNIX）。在已经安装了多个 DB2 产品的情况下进行安装时，就可以使用通用测试修订。

如果安装了本地语言，那么可能还需要一个单独的本地语言测试修订。仅当本地语言测试修订与已安装的 DB2 产品处于同一测试修订级别时，才能应用该本地语言测试修订。如果您要应用通用测试修订，那么必须同时应用通用测试修订和本地语言测试修订来更新 DB2 产品。

从 IBM 软件支持机构获取测试修订，并遵循自述文件中有关安装、测试和除去（如果需要）该测试修订的指示信息。
当在一个多分区数据库分区环境中安装测试修订时，该系统必须处于脱机状态，并且必须将参与实例的所有计算机都升级到同一测试修订级别。

第 10 章 了解有关故障诊断的更多信息

使用 DB2 数据库产品时，您有时可能会遇到问题。此问题可能由数据库管理器以及针对数据库运行的应用程序报告，也可能由用户向您提供数据库“某些地方不太正常”反馈来报告此问题。

此处展示的概念和工具将向您介绍对数据库操作中真实的或察觉到的问题进行故障诊断的任务，并帮助您执行该任务。这里强调了在正确时间捕获正确数据的重要性，因此首次出现数据捕获是第一个讨论的工具。将展示数据库管理器用于捕获有关数据库操作的数据的其他日志和文件，同时还提及了操作系统诊断工具。

了解更多信息

下列主题可以帮助您获取有效地对 DB2 产品问题进行故障诊断所需的概念性信息:

- 关于故障诊断

  故障诊断是解决问题的系统性方法。其目标是，确定某项功能未按预期方式工作的原因以及解决问题的方式。

- 关于诊断数据目录路径

  根据您使用的平台不同，可以在由 **diagpath** 数据库管理器配置参数指定的诊断数据目录中找到转储文件、陷阱文件、诊断日志文件、管理通知日志文件、警报日志文件和“首次出现数据集合”（FODC）程序包中包含的 DB2 诊断信息。

- 关于管理通知日志文件

  DB2 数据库管理器将下列类型的信息写入管理通知日志: DB2 实用程序（例如 REORG 和 BACKUP）的状态; 客户机应用程序错误、服务类更改、许可证发放活动、日志文件路径和存储问题、监视活动并为活动建立索引，以及表空间问题。数据库管理员可以使用这些信息来诊断问题、调整数据库或监视数据库。

- 关于 DB2 诊断（db2diag）日志文件

  虽然已使用标准化消息格式将管理通知日志消息记录至 db2diag 日志文件，但还是建议您先查看 db2diag 日志文件以了解数据库所发生的情况。

- 关于特定于平台的错误日志

  在 DB2 之外也提供了许多其他文件和实用程序，可用于帮助分析问题。通常，在确定问题的根本原因时，它们和 DB2 文件一样重要。

- 关于消息

  了解更多关于消息的信息可以帮助您确定错误或问题以及通过相应的恢复操作来解决问题。此信息还可用于了解消息的生成位置和记录位置。

- 关于内部返回码

  有两种类型的内部返回码: ZRC 值和 ECF 值。它们将出现在 DB2 跟踪输出和 db2diag 日志文件中。ZRC 和 ECF 值通常为负数，并用于表示错误状况。

- 关于转储文件

© Copyright IBM Corp. 2006, 2009

转储文件是在发生错误时创建的，它包含将有助于诊断问题（例如，内部控制块）的其他信息。写至转储文件的每个数据项都具有与其相关联的时间戳记，以帮助进行问题确定。转储文件使用二进制格式，目的是供 DB2 客户支持代表使用。

- 关于陷阱文件

  如果 DB2 由于陷阱、分段违例或异常而不能继续处理，它就会生成陷阱文件。DB2 接收到的所有信号或异常都会记录在陷阱文件中。陷阱文件还包含发生错误时正在运行的函数序列。此序列有时又称“函数调用堆栈”或“堆栈跟踪”。陷阱文件还包含有关捕获到信号或异常时进程的状态的其他信息。

- 关于首次出现数据捕获（FODC）

  首次出现数据捕获（FODC）是用来捕获有关 DB2 实例的基于方案的数据的过程。DB2 用户可根据特定症状手动调用 FODC，也可在检测到预定义方案或症状时自动调用 FODC。此信息减少了再现错误以获取诊断信息的需要。

- 关于调出脚本（db2cos）输出文件

  在缺省情况下，当数据库管理器因为应急启动、陷阱、分段违例或异常而不能继续处理时，将会调用 db2cos 脚本。

- 关于组合 DB2 和操作系统诊断

  在诊断与内存、交换文件、CPU、磁盘存储器和其他资源有关的一些问题时，需要完整地了解给定操作系统管理这些资源的方式。至少在确定与资源有关的问题时需要知道对于每个用户而言，该资源存在的限制及限制程度。

诊断数据目录路径

根据您使用的平台不同，可以在由 **diagpath** 数据库管理器配置参数指定的诊断数据目录中找到转储文件、陷阱文件、诊断日志文件、管理通知日志文件、警报日志文件和“首次出现数据集合”（FODC）程序包中包含的 DB2 诊断信息。

概述

通过使用 **diagpath** 数据库管理器配置参数指定诊断数据目录路径，可以确定使用下面哪种目录路径方法来诊断数据存储器:

单个诊断数据目录路径

不管数据库是否分区，DB2 实例的所有诊断数据都将存储在单个目录中。在分区数据库环境中，主机中的不同分区的诊断数据都将转储至此诊断数据目录路径。当 **diagpath** 值设置为 NULL 或者任何不带 `$h` 或 `$n` 模式标识的有效路径名时，此诊断数据目录路径是缺省条件。

分割诊断数据目录路径

对于分区数据库环境，可以将诊断数据单独存储在按照主机和/或数据库分区指定的目录中。因此，所给定诊断目录中每种类型的诊断文件将只包含一个主机中的诊断信息，或一个数据库分区中的诊断信息，或者同时包含一个主机和一个数据库分区中的诊断信息。

有关 **diagpath** 数据库管理器配置参数设置的信息，请参阅: *《数据库管理概念和配置参考》*中的『diagpath - 诊断数据目录路径配置参数』。

优点

指定诊断数据目录路径具有下列优点:

- 通过设置单个诊断数据目录路径，可以将多个数据库分区和主机中的诊断信息合并在一个中央位置以便于访问。
- 可以提高诊断日志记录性能；这是因为如果您按照主机或者数据库分区来分割诊断数据目录路径，就会更少争用 db2diag 日志文件。

合并文件以及对记录排序

在分割诊断数据目录路径的情况下，可以使用 db2diag -merge 命令并根据时间戳记来合并同一类型的多个诊断文件中的记录以及对这些记录排序。有关更多信息，请参阅: *Command Reference* 中的『db2diag - db2diag 日志分析工具命令』。

按主机和/或数据库分区分割诊断数据目录路径

采用 **diagpath** 数据库管理器配置参数的缺省 DB2 诊断数据目录路径设置时，会将所有诊断信息收集在单个诊断数据目录中。然而，可以分割诊断数据目录路径，以便按照主机和/或数据库分区创建和指定不同的目录。这样，就可以按照诊断数据转储所源自的主机或数据库分区，将先前存储在单个目录中的诊断转储文件存储在不同的目录中。

开始前

需要具备 DB2 版本 9.7 修订包 1 或更高版本的修订包。

关于此任务

您将能够按照启动了诊断数据转储的主机或数据库分区将诊断数据目录路径分割为单独存储的诊断信息。

限制

分割诊断数据目录路径以使多个诊断信息源保持独立，这在分区数据库环境中最有用。

过程

- **按物理主机分割诊断数据目录路径**
  - 要分割缺省诊断数据目录路径，请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照物理主机来分割缺省诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"$h"'
      ```

      此命令将在缺省诊断数据目录下使用主机名创建一个子目录，如下所示:

      Default_diagpath/HOST_*hostname*
  - 要分割用户指定的诊断数据目录路径（例如，/home/usr1/db2dump/），请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照物理主机来分割 /home/usr1/db2dump/ 诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $h"'
      ```

注: 必须用一个空格将 /home/usr1/db2dump/ 与 $h 隔开。

此命令将在 /home/usr1/db2dump/ 诊断数据目录下使用主机名创建一个子目录，如下所示:

/home/usr1/db2dump/HOST_*hostname*

- **按数据库分区分割诊断数据目录路径**
  - 要分割缺省诊断数据目录路径，请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照数据库分区来分割缺省诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"$n"'
      ```

      此命令将在缺省诊断数据目录下使用分区号为每个分区创建一个子目录，如下所示:

      Default_diagpath/NODE*number*
  - 要分割用户指定的诊断数据目录路径（例如，/home/usr1/db2dump/），请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照数据库分区来分割 /home/usr1/db2dump/ 诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $n"'
      ```

      注: 必须用一个空格将 /home/usr1/db2dump/ 与 $n 隔开。

      此命令将在 /home/usr1/db2dump/ 诊断数据目录下使用分区号为每个分区创建一个子目录，如下所示:

      /home/usr1/db2dump/NODE*number*
- **按物理主机以及按每个物理主机中的数据库分区来分割诊断数据目录路径**
  - 要分割缺省诊断数据目录路径，请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照物理主机以及按每个物理主机中的数据库分区来分割缺省诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"$h$n"'
      ```

      此命令将在缺省诊断数据目录下使用主机名和分区号为主机中的每个逻辑分区创建一个子目录，如下所示:

      Default_diagpath/HOST_*hostname*/NODE*number*
  - 要分割用户指定的诊断数据目录路径（例如，/home/usr1/db2dump/），请执行下列步骤:
    - 发出以下命令来设置 **diagpath** 数据库管理器配置参数，以按照物理主机以及按每个物理主机中的数据库分区来分割 /home/usr1/db2dump/ 诊断数据目录路径:

      ```
      db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $h$n"'
      ```

      注: 必须用一个空格将 /home/usr1/db2dump/ 与 hn 隔开。

      此命令将在 /home/usr1/db2dump/ 诊断数据目录下使用主机名和分区号为主机中的每个逻辑分区创建一个子目录，如下所示:

      /home/usr1/db2dump/HOST_*hostname*/NODE*number*

      例如，名为 boson 的 AIX 主机有 3 个数据库分区，每个数据库分区的节点号分别为 0、1 和 2。该目录的列表输出示例类似于以下内容:

```
usr1@boson /home/user1/db2dump->ls -R *
HOST_boson:

HOST_boson:
NODE0000 NODE0001 NODE0002

HOST_boson/NODE0000:
db2diag.log db2eventlog.000 db2resync.log db2sampl_Import.msg events usr1.nfy

HOST_boson/NODE0000/events:
db2optstats.0.log

HOST_boson/NODE0001:
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog

HOST_boson/NODE0001/stmmlog:
stmm.0.log

HOST_boson/NODE0002:
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

下一步任务

注:

- 如果指定了按数据库分区分割诊断数据目录路径（$n 或 hn），那么始终会为每个主机都创建 NODE0000 目录。如果在其中创建了 NODE0000 目录的主机中不存在数据库分区 0，那么可以忽略 NODE0000 目录。
- 要检查是否成功分割了诊断数据目录路径的设置，请执行以下命令:

  ```
  db2 get dbm cfg | grep DIAGPATH
  ```

  如果成功分割了诊断数据目录路径，那么将返回带有前导空格的值 $h、$n 或 hn。例如，返回的输出类似于以下内容:

  ```
  Diagnostic data directory path            (DIAGPATH) = /home/usr1/db2dump/ $h$n
  ```

要将不同的 db2diag 日志文件合并在一起以便更容易进行分析和故障诊断，请使用 db2diag -merge 命令。有关更多信息，请参阅: *Command Reference* 中的『db2diag - db2diag 日志分析工具命令』以及第 368 页的『使用 db2diag 工具来分析 db2diag 日志文件』。。

管理通知日志

管理通知日志（*instance_name*.nfy）是可以从中获取有关大量数据库管理和维护活动的信息的存储库。数据库管理员可以使用这些信息来诊断问题、调整数据库或仅监视数据库。

在 UNIX 和 Linux 操作系统平台上，DB2 数据库管理器将下列类型的信息写入管理通知日志（在 Windows 操作系统平台上，事件日志用来记录管理通知事件）:

- DB2 实用程序（例如 REORG 和 BACKUP）的状态
- 客户机应用程序错误
- 服务类更改
- 许可证发放活动
- 文件路径
- 存储器问题

- 监视活动
- 建立索引活动
- 表空间问题

管理通知日志消息也以标准化消息格式记录到 db2diag 日志文件。

通知消息提供了其他信息以补充提供的 SQLCODE。

管理通知日志文件有两种不同的形式:

单一管理通知日志文件

一个活动的管理通知日志文件，名为 *instance_name*.nfy，其大小将无限增大。这是缺省形式，每当 **diagsize** 数据库管理器配置参数的值为缺省值 0 时，都存在此文件。

旋转管理通知日志文件

单一活动日志文件（名为 *instance_name*.*N*.nfy，其中 *N* 是从 0 开始持续增大的文件名数字索引）。虽然可以在 **diagpath** 配置参数所定义的位置中找到一系列管理通知日志文件，但每个日志文件都将增大到所限制的大小为止。达到该大小时，系统将关闭该日志文件，然后创建并打开一个新的日志文件进行日志记录，新日志文件将具有增大的文件名索引（*instance_name*.*N+1*.nfy）。每当 **diagsize** 数据库管理器配置参数具有非零值时，都存在此文件。

注: 在 Windows 操作系统平台上，单一管理通知日志文件和旋转管理通知日志文件都不可用。

您可以通过适当地设置 **diagsize** 数据库管理器配置参数来选择系统上存在的日志文件形式。

配置

通过设置下列数据库管理器配置参数，可以在大小、位置和事件类型以及所记录详细信息级别方面对管理通知日志文件进行配置:

diagsize

diagsize 的值确定将采用的管理通知日志文件的形式。如果值为 0，那么将采用单一管理通知日志文件。如果值不为 0，那么将采用旋转管理通知日志文件，并且这个非零值还指定所有旋转诊断日志文件和所有旋转管理通知日志文件的总大小。必须重新启动实例才能使新的 **diagsize** 参数值生效。有关完整的详细信息，请参阅以下主题: diagsize －“诊断日志文件大小”配置参数。

diagpath

可以指定将诊断信息写入 **diagpath** 配置参数所定义位置中的管理通知日志文件。有关完整的详细信息，请参阅以下主题: diagpath －“诊断数据目录路径”配置参数。

notifylevel

可以使用 **notifylevel** 配置参数来指定事件类型以及写入管理通知日志文件的详细信息级别。有关完整的详细信息，请参阅以下主题: notifylevel －“通知级别”配置参数。

解释管理通知日志文件条目

您可以使用文本编辑器来查看怀疑发生了问题的机器上的管理通知日志文件。记录的最新事件在文件的最后面。

通常，每个条目包含下列部分:

- 时间戳记
- 报告错误的位置。应用程序标识允许您匹配在服务器和客户机的日志上与应用程序有关的条目。
- 说明错误的诊断消息（通常以“DIA”或“ADM”开头）。
- 任何可用的支持数据，例如，SQLCA 数据结构和指向任何其他转储文件或陷阱文件的位置的指针。

以下示例显示样本日志条目的头信息，且标识了日志的所有部分。

注: 不是所有日志条目都包括所有这些部分。

```
2006-02-15-19.33.37.630000 1     实例: DB2 2    节点: 000 3
PID: 940(db2syscs.exe) TID: 660 4    Appid: *LOCAL.DB2.020205091435 5
恢复管理器 6   sqlpresr 7   探测点: 1 8    数据库: SAMPLE   9
ADM1530E 10 已启动崩溃恢复。 11
```

图注:

1. 消息的时间戳记。
2. 生成该消息的实例的名称。
3. 对于多分区系统，此项为生成该消息的数据库分区。在非分区数据库中，该值为“000”。
4. 进程标识（PID），后跟进程名称，再后跟负责生成消息的线程标识（TID）。
5. 进程正在为其工作的应用程序的标识。在本示例中，生成消息的进程代表标识为 *LOCAL.DB2.020205091435 的应用程序工作。

   此值与 **appl_id** 监视元素数据相同。有关如何解释此值的详细信息，请参阅 **appl_id** 监视元素的文档。

   要标识关于特定应用程序标识的信息，执行下列其中一项操作:

   - 在 DB2 服务器上使用 LIST APPLICATIONS 命令或在 DB2 Connect 网关上使用 LIST DCS APPLICATIONS 命令来查看应用程序标识列表。可以根据此列表确定有关遇到错误的客户机的信息，例如其节点名以及其 TCP/IP 地址。
   - 使用 GET SNAPSHOT FOR APPLICATION 命令查看应用程序标识列表。
6. 写入消息的 DB2 组件。对于由使用 db2AdminMsgWrite API 的用户应用程序编写的消息，该组件将被称为“用户应用程序”。
7. 提供消息的函数的名称。此函数在写入消息的 DB2 组件中运行。对于由使用 db2AdminMsgWrite API 的用户应用程序编写的消息，该函数将被称为“用户函数”。
8. 唯一内部标识。此数字允许 DB2 客户支持和开发在报告了消息的 DB2 源代码中找到相应位置。
9. 发生错误的数据库。
10. 以十六进制代码指示错误类型和编号的消息（如果存在）。

11. 说明记录的事件的消息文本（如果存在）。

设置管理通知日志文件的错误捕获级别

本任务描述如何设置管理通知日志文件的错误捕获级别。

DB2 在管理通知日志中记录的信息由 **NOTIFYLEVEL** 设置确定。

- 要检查当前设置，请发出 GET DBM CFG 命令。

  查找以下变量:

  ```
  通知级别（NOTIFYLEVEL）                    = 3
  ```

- 要更改此设置，请使用 UPDATE DBM CFG 命令。 例如:

  ```
  DB2 UPDATE DBM CFG USING NOTIFYLEVEL X
  ```

  其中，*X* 是所要使用的通知级别。

DB2 诊断（db2diag）日志文件

DB2 诊断 db2diag 日志文件主要供 IBM 软件支持机构用于进行故障诊断。管理通知日志主要由数据库和系统管理员用于进行故障诊断。管理通知日志消息也以标准化消息格式记录到 db2diag 日志文件。

概述

由于 DB2 诊断和管理通知消息都将记录到 db2diag 日志文件中，因此这通常使 db2diag 日志文件成为您需要获取关于数据库操作的信息时应该首先检查的位置。“相关链接”部分中列示的主题提供了有关如何解释诊断日志文件内容的帮助。如果您进行的故障诊断尝试无法解决问题，并且您认为需要帮助，那么可以与 IBM 软件支持机构联系（有关详细信息，请参阅“与 IBM 软件支持机构联系”主题）。在收集将被请求发送到 IBM 软件支持机构的相关诊断信息时，可以将 db2diag 日志文件包括在其他信息来源中，这些来源包括其他相关日志、存储器转储以及跟踪输出等等。

db2diag 日志文件有两种不同的形式:

单一诊断日志文件

一个活动的诊断日志文件，名为 db2diag.log，其大小将无限增大。这是缺省形式，每当 **diagsize** 数据库管理器配置参数的值为缺省值 0 时，都存在此文件。

旋转诊断日志文件

单一活动日志文件（名为 db2diag.*N*.log，其中 *N* 是从 0 开始持续增大的文件名数字索引）。虽然可以在 **diagpath** 配置参数所定义的位置中找到一系列诊断日志文件，但每个日志文件都将增大到所限制的大小为止。达到该大小时，系统将关闭该日志文件，然后创建并打开一个新的日志文件进行日志记录，新日志文件将具有增大的文件名索引（db2diag.*N*+*1*.log）。每当 **diagsize** 数据库管理器配置参数具有非零值时，都存在此文件。

您可以通过适当地设置 **diagsize** 数据库管理器配置参数来选择系统上存在的日志文件形式。

配置

通过设置下列数据库管理器配置参数，可以在大小、位置以及所记录的诊断错误类型方面对 db2diag 日志文件进行配置:

diagsize

diagsize 的值确定将采用的诊断日志文件的形式。如果值为 0，那么将采用单一诊断日志文件。如果值不为 0，那么将采用旋转诊断日志文件，并且这个非零值还指定所有旋转诊断日志文件和所有旋转管理通知日志文件的总大小。必须重新启动实例才能使新的 **diagsize** 参数值生效。有关完整的详细信息，请参阅以下主题：diagsize －“诊断日志文件大小”配置参数。

diagpath

可以指定将诊断信息写入 **diagpath** 配置参数所定义位置中的 db2diag 日志文件。有关完整的详细信息，请参阅以下主题：diagpath －“诊断数据目录路径”配置参数。

diaglevel

可以使用 **diaglevel** 配置参数来指定要写入 db2diag 日志文件的诊断错误的类型。有关完整的详细信息，请参阅以下主题：diaglevel －“诊断错误捕获级别”配置参数。

解释诊断日志文件条目

您可以使用 db2diag 日志文件分析工具（db2diag）对 db2diag 日志文件进行过滤和格式化。虽然已使用标准化消息格式将管理通知日志消息记录至 db2diag 日志文件，但还是建议先查看 db2diag 日志文件以了解数据库所发生的情况。

除了使用 db2diag 之外，还可使用文本编辑器来查看怀疑发生了问题的机器上的诊断日志文件。记录的最新事件在文件的最后面。

注：管理通知（*instance_name*.nfy）和诊断（db2diag.log）日志将作为单一日志文件持续增大。如果 **diagsize** 数据库管理器配置参数设置为非零值，那么管理通知和 db2diag log 文件将变为一系列旋转日志文件（*instance_name.N*.nfy 和 db2diag.*N*.log），这些文件具有由 **diagsize** 配置参数值确定的有限总大小。

以下示例显示样本日志条目的头信息，且标识了日志的所有部分。

注：不是所有日志条目都包括所有这些部分。只有开头的一些字段（时间戳记至 TID）和函数才会显示在所有 db2diag 日志文件记录中。

```
2007-05-18-14.20.46.973000-240 1  I27204F655 2  LEVEL: Info 3
PID : 3228 4  TID : 8796 5  PROC : db2syscs.exe 6
INSTANCE: DB2MPP 7  NODE : 002 8  DB : WIN3DB1 9
APPHDL : 0-51 10  APPID: 9.26.54.62.45837.070518182042 11
AUTHID : UDBADM 12
EDUID : 8796 13  EDUNAME: db2agntp 14  (WIN3DB1) 2
FUNCTION: 15  DB2 UDB, data management, sqldInitDBCB, probe:4820
DATA #1 : 16  String, 26 bytes
Setting ADC Threshold to:
DATA #2 : unsigned integer, 8 bytes
1048576
```

图注:

1. 消息的时间戳记和时区。

注：db2diag 日志文件中的时间戳记包含时区。例如：2006-02-13-14.34.35.965000-300，其中"-300"是 UTC（全球标准时间，以前称为 GMT）与应用程序服务器中的当地时间（以分钟为单位）之间的偏差。因此，-300 表示 UTC - 5 小时，如 EST（东部标准时间）。

2. 记录标识字段。对于创建 DB2 诊断日志的平台，db2diag 日志文件的记录标识指定要记录的当前消息的文件位移（如“27204”）和消息长度（如“655”）。

3. 与错误消息相关联的诊断级别。例如，参考、警告、错误、严重或事件。

4 进程标识。

5. 线程标识。

6. 进程名称。

7. 生成该消息的实例的名称。

8. 对于多分区系统，此项为生成该消息的数据库分区。在非分区数据库中，该值为“000”。

9. 数据库名称。

10. 应用程序句柄。此值与 db2pd 输出和锁定转储文件中使用的值相对应。它包括后跟协调程序索引编号并且用破折号分开的协调程序分区号。

11. 进程正在为其工作的应用程序的标识。在本示例中，生成消息的进程代表标识为 9.26.54.62.45837.070518182042 的应用程序工作。

    TCP/IP 生成的应用程序标识由三个部分组成:

    1. **IP 地址**: 它表示为 32-bit 位数字，最大显示为 8 位十六进制字符。
    2. **端口号**: 显示为 4 位十六进制字符。
    3. 此应用程序的实例的**唯一标识**。

    注: 当 IP 地址或端口号的十六进制版本以 0 至 9 开头时，它们将分别转换为 G 至 P。例如，“0”映射到“G”，“1”映射到“H”，依此类推。IP 地址 AC10150C.NA04.006D07064947 表示为如下所示: IP 地址仍为 AC10150C，它将转换为 172.16.21.12。端口号为 NA04。第一个字符为“N”，它将映射为“7”。因此，端口号的十六进制为 7A04，它将转换为十进制格式 31236。

    此值与 *appl_id* 监视元素数据相同。有关如何解释此值的详细信息，请参阅 *appl_id* 监视元素的文档。

    要标识关于特定应用程序标识的信息，执行下列其中一项操作:

    - 在 DB2 服务器上使用 LIST APPLICATIONS 命令或在 DB2 Connect 网关上使用 LIST DCS APPLICATIONS 命令来查看应用程序标识列表。可以根据此列表确定有关遇到错误的客户机的信息，例如其数据库分区名以及其 TCP/IP 地址。
    - 使用 GET SNAPSHOT FOR APPLICATION 命令查看应用程序标识列表。
    - 使用 db2pd -applications -db <dbname> 命令。

12 授权标识。

13 引擎可分派单元标识。

14 引擎可分派单元的名称。

15. 产品名（“DB2”）、组件名（“数据管理”）、写入消息的函数名（“sqlInitDBCB”）以及函数内的探测点（“4820”）。

16. 被调用函数返回的信息。可能会返回多个数据字段。

既然已经看到了样本 db2diag 日志文件条目，以下是所有可能字段的列表:

```
<timestamp><timezone>       <recordID>         级别: <level> (<source>)
PID: <pid>              TID: <tid>       PROC: <procName>
实例: <instance>        节点: <node>     数据库: <database>
APPHDL: <appHandle>     APPID: <appID>
AUTHID  : <authID>
EDUID   : <eduID>           EDUNAME: <engine dispatchable unit name>
函数: <prodName>, <compName>, <funcName>, 探测点: <probeNum>
消息: <messageID>  <msgText>
已调用: <prodName>, <compName>, <funcName>   OSERR: <errorName> (<errno>)
返回码: <type>=<retCode> <errorDesc>
ARG #N: <typeTitle>, <typeName>, <size> bytes
... 参数 ...
DATA #N: <typeTitle>, <typeName>, <size> bytes
... 数据 ...
```

尚未在示例中说明的字段包括:

- <source> 指示已记录错误的源头。(可在样本的第一行结尾找到。)可能的值包括:
  - origin - 导致错误的函数(起始点)已记录消息
  - OS - 操作系统已生成错误
  - received - 已从另一个进程(客户机/服务器)接收到错误
  - sent - 错误已发送至另一进程(客户机/服务器)
- MESSAGE 包含要记录的消息。它包括:
  - <messageID> - 消息号, 如 ECF=0x9000004A 或 DIA8604C
  - <msgText> - 错误描述

  如果还有被调用(CALLED)字段, 那么 <msgText> 是被调用(CALLED)函数返回的错误对记录消息的函数(在"函数"字段中指定)的影响
- 被调用(CALLED) 这是返回错误的函数。它包括:
  - <prodName> - 产品名: "OS"、"DB2"、"DB2 Tools"或"DB2 Common"
  - <compName> - 组件名称(如果是系统调用则为"-")
  - <funcName> - 被调用函数名
- OSERR 这是被调用(CALLED)的系统调用返回的操作系统错误。(可在 CALLED 所在行的结尾找到。)它包括:
  - <errorName> - 特定于系统的错误名称
  - <errno> - 操作系统错误号
- ARG 本节列示返回错误的函数调用的自变量。它包括:
  - <N> - 对"被调用"函数的调用中的自变量的位置
  - <typeTitle> - 与第 N 个自变量类型名相关联的标注
  - <typeName> - 要记录的自变量的类型的名称
  - <size> - 要记录的自变量的大小
- DATA 它包含记录函数转储的所有其他数据。它包括:
  - <N> - 要转储的数据对象的序号

– <typeTitle> - 要转储的数据的标注
– <typeName> - 要记录的数据字段的类型的名称，如 PD_TYPE_UINT32 和 PD_TYPE_STRING
– <size> - 数据对象的大小

解释 db2diag 日志文件的信息记录

db2diag 日志文件中的第一个消息应该总是参考记录。

参考记录的示例如下所示:

```
2006-02-09-18.07.31.059000-300 I1H917              级别: 事件
PID: 3140                  TID: 2864        PROC: db2start.exe
实例: DB2                  节点: 000
函数: DB2 UDB, RAS/PD 组件, _pdlogInt, 探测点: 120
开始: 新的诊断日志文件
DATA #1: 构建级别, 124 字节
实例"DB2"使用"32"位和 DB2 代码发行版"SQL09010", 其级别标识为"01010107"。
参考标记为"DB2 v9.1.0.190"、"s060121"、""和修订包"0"。
DATA #2 : 系统信息, 1564 字节
系统: WIN32_NT MYSRVR Service Pack 2 5.1 x86 系列 15, 模型 2, 步骤 4
CPU: 总计: 1 联机: 1 每个套接字的核心数: 1 每个核心的线程化程度: 1
物理内存（MB）: 总计: 1024 空闲: 617 可用: 617
虚拟内存（MB）: 总计: 2462 空闲: 2830
交换内存（MB）: 总计: 1438 空闲: 2213
此记录中的信息仅在创建此文件时有效（请参阅此记录的时间戳记）
```

参考记录是 db2start 在每个逻辑分区上的输出。这会生成多个参考记录: 每个逻辑分区一个参考记录。因为参考记录包含的内存值在每个分区上都不同，所以此信息可能很有用。

设置诊断日志文件的错误捕获级别

DB2 诊断（db2diag）日志文件是包含 DB2 所记录的文本信息的文件。此信息用于进行故障诊断，并且主要供 IBM 软件支持机构使用。

db2diag 日志文件中记录的诊断错误的类型由 **diaglevel** 数据库管理器配置参数设置确定。

- 要检查当前设置，请发出 GET DBM CFG 命令。

  查找以下变量:

  ```
  诊断错误捕获级别（DIAGLEVEL）                = 3
  ```

- 要动态更改该值，请使用 UPDATE DBM CFG 命令。

  要以联机方式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
  ```

  例如:

  ```
  DB2 UPDATE DBM CFG USING DIAGLEVEL X
  ```

  其中 X 是期望的通知级别。 在诊断可再现的问题时，IBM 软件支持机构的人员可能会建议您在执行故障诊断时使用 **diaglevel** 值 4。

组合 DB2 数据库和操作系统诊断

在诊断与内存、交换文件、CPU、磁盘存储器和其他资源有关的一些问题时，需要完整地了解给定操作系统管理这些资源的方式。至少在确定与资源有关的问题时需要知道对于每个用户而言，该资源存在的限制及限制程度。（相关限制通常用于 DB2 实例所有者的用户标识。）

以下是必须获取的一些重要配置信息:

- 操作系统补丁级别、已安装软件和升级历史
- CPU 数目
- RAM 量
- 交换和文件高速缓存设置
- 用户数据和文件资源限制及每个用户的进程极限
- IPC 资源限制（消息队列、共享内存段和信号量）
- 磁盘存储器类型
- 机器还有什么功能？DB2 是否争用资源？
- 认证在何处进行？

大多数平台有直接的命令可用来检索资源信息。但是，您很少需要手动获取该信息，原因是 db2support 实用程序会收集此数据及更多其他信息。当指定了 -s 和 -m 选项时，db2support 生成的 `detailed_system_info.html` 文件包含用来收集此信息的许多操作系统命令的语法。

下列练习用于帮助您发现 DB2 诊断文件中的系统配置和用户环境信息。第一个练习让您熟悉运行 db2support 实用程序涉及的步骤。后续练习涉及陷阱文件，这些文件会提供更多 DB2 生成的数据，而这些数据在了解用户环境和资源限制时非常有用。

练习 1: 运行 db2support 命令

1. 使用 db2start 命令启动 DB2 实例。
2. 假定您已具有可用的 SAMPLE 数据库，请创建一个目录以存储 db2support 的输出。
3. 切换至该目录并发出:

   ```
   db2support <directory> -d sample -s -m
   ```

4. 复查控制台输出，特别是所收集信息的类型。

   在 Windows 上运行时，应该看到类似以下的输出:

   ```
   ...
   正在收集"系统文件"
            "db2cache.prf"
            "db2cos9402136.0"
            "db2cos9402840.0"
            "db2dbamr.prf"
            "db2diag.bak"
            "db2eventlog.000"
            "db2misc.prf"
            "db2nodes.cfg"
            "db2profile.bat"
            "db2systm"
            "db2tools.prf"
            "HealthRulesV82.reg"
            "db2dasdiag.log"
       ...
   正在收集"详细的操作系统和硬件信息"
   ```

```
正在收集"系统资源信息（磁盘、CPU和内存）"
正在收集"操作系统和级别"
正在收集"JDK 级别"
正在收集"DB2 发行版信息"
正在收集"DB2 安装路径信息"
正在收集"注册表信息"
...
正在创建最终输出归档
          "db2support.html"
          "db2_sqllib_directory.txt"
          "detailed_system_info.html"
          "db2supp_system.zip"
          "dbm_detailed.supp_cfg"
          "db2diag.log"
db2support 现在已完成。
 已生成归档文件"db2support.zip"
```

5. 现在使用 Web 浏览器来查看 detailed_system_info.html 文件。在每个系统上标识下列信息:
   - CPU 数目
   - 操作系统级别
   - 用户环境
   - 用户资源限制（UNIX ulimit 命令）

练习 2: 查找 DB2 陷阱文件中的环境信息

1. 确保 DB2 实例已启动，然后发出

   ```
   db2pd -stack all
   ```

   调用堆栈放在 **diagpath** 数据库管理器配置参数定义的诊断目录的文件中。
2. 在其中一个陷阱文件中查找下列内容:
   - DB2 代码级别
   - 数据段顶部（这是所需的最大专用地址空间）
   - 当前数据大小（这是最大专用地址空间限制）
   - 当前核心大小（这是最大核心文件限制）
   - 信号处理程序（此信息可能不会出现在所有陷阱文件中）
   - 环境变量（此信息可能不会出现在所有陷阱文件中）
   - 映射输出（显示装入的库）

Windows 中的示例陷阱文件（被截断）:

```
...
<DB2TrapFile version="1.0">
<Trap>
<Header>
DB2 build information: DB2 v9.1.0.190 s060121 SQL09010
timestamp: 2006-02-17-14.03.43.846000
uname: S:Windows
comment:
process id: 940
thread id: 3592
</Header>
<SystemInformation>
Number of Processors: 1
Processor Type: x86 Family 15 Model 2 Stepping 4
OS Version: Microsoft Windows XP, Service Pack 2 (5.1)
Current Build: 2600
</SystemInformation>
```

```
<MemoryInformation>
<Usage>
Physical Memory:    1023 total,     568 free.
Virtual Memory :    2047 total,    1882 free.
Paging File    :    2461 total,    2011 free.
Ext. Virtual   :       0 free.
</Usage>
</MemoryInformation>
<EnvironmentVariables>
<![CDATA[
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2_EXTSECURITY=YES
[g] DB2SYSTEM=MYSRVR
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
]]></EnvironmentVariables>
```

使 DB2 和系统事件或错误相关

通常会忽略系统消息和错误日志。如果在问题定义和调查的初始阶段花时间执行一个简单任务，就可以在解决问题时节省几小时、几天甚至几星期的时间。该任务将会比较不同日志中的各个条目，并且记录看起来与时间和各个条目引用的资源相关的所有内容。

虽然并非总是与问题诊断有关，但许多情况下系统日志中会提供最好的线索。如果可将报告的系统问题与 DB2 错误相关，通常就能确定直接导致 DB2 症状的原因。很明显的示例包括磁盘错误、网络错误和硬件错误。并不那么明显的示例包括在不同机器（如域控制器）上报告的错误，机器不同会影响连接时间或认证。

可以调查系统日志以评估稳定性，特别是在全新的系统上报告问题时尤其如此。在常用应用程序中间歇发生陷阱可能表示存在底层硬件问题。

以下是系统日志提供的一些其他信息。

- 重要事件，如重新引导系统的时间
- 系统上发生 DB2 陷阱（及失败的其他软件中的错误、陷阱或异常）的时间
- 内核应急启动、文件系统空间不足和交换空间不足错误（可能导致系统无法创建或派生新进程）

系统日志可帮助您在 db2diag 日志文件中排除作为考虑因素的崩溃条目。如果在 DB2 管理通知或 DB2 诊断日志中找到崩溃条目，但先前没有任何错误，那么 DB2 崩溃恢复可能由系统关闭所致。

这一关联信息原则扩展至来自任何源的日志和所有可标识用户症状。例如，它在标识和描述来自另一应用程序的日志的关联条目时很有用，即使您不能完整地解释它们也是如此。

概述此信息就能够很完整地了解服务器和发生问题时进行的各种事件。

db2cos（调出脚本）输出文件

在缺省情况下，当数据库管理器因为应急启动、陷阱、分段违例或异常而不能继续处理时，将会调用 db2cos 脚本。每个缺省 db2cos 脚本将调用 db2pd 命令以打开方式收集信息。

db2cos 脚本的名称的格式为 db2cos_hang、db2cos_trap 等等。每个脚本的行为方式类似，但 db2cos_hang 则有所不同，它是通过 db2fodc 工具调用的。

缺省 db2cos 脚本在 bin 目录中。在 UNIX 操作系统上，此目录是只读的。可将 db2cos 脚本文件复制至 adm 目录，必要时可在该位置修改该文件。如果 db2cos 脚本在 adm 目录中，那么会运行该脚本，否则会运行 bin 目录中的脚本。

在多分区配置中，将仅对遇到陷阱的分区上的陷阱代理程序调用该脚本。如果需要从其他分区收集信息，可更新 db2cos 脚本以使用 db2_all命令，或者如果所有分区在同一台机器上，那么在 db2pd 命令上指定 -alldbpartitionnums 选项。

还可通过 db2pdcfg -cos 命令来配置触发 db2cos 调用的信号类型。缺省配置用于要在发生应急启动或陷阱时运行的 db2cos 脚本。但是，在缺省情况下，生成的信号不会启动 db2cos 脚本。

发生应急启动、陷阱、分段违例或异常时，事件顺序如下所示:

1. 创建陷阱文件
2. 调用信号处理程序
3. 调用 db2cos 脚本（取决于启用的 db2cos 设置）
4. 在管理通知日志中记录相应条目
5. 在 db2diag 日志文件中记录相应条目

db2cos 脚本中的 db2pd 命令收集到的缺省信息包括有关操作系统、已安装 DB2 产品的版本和服务级别、数据库管理器和数据库配置的详细信息，以及有关以下各项的状态的信息: 代理程序、内存池、内存块、应用程序、实用程序、事务、缓冲池、锁定、事务日志、表空间和容器。此外，它还会提供有关下列各项的信息: 动态、静态和目录高速缓存的状态、表和索引统计信息、恢复状态以及重新优化的 SQL 语句及活动语句列表。如果必须收集进一步的信息，请更新 db2cos 脚本并指定其他命令。

调用缺省 db2cos 脚本时，它将在 DIAGPATH 数据库管理器配置参数指定的目录中生成输出文件。这些文件名为 XXX.YYY.ZZZ.cos.txt，其中 XXX 是进程标识（PID），YYY 是线程标识（TID），而 ZZZ 是数据库分区号（对于单分区数据库则为 000）。如果存在多线程陷阱，那么会对每个线程单独调用 db2cos 脚本。如果 PID 和 TID 组合多次出现，那么该数据将追加至文件。还会显示时间戳记，所以您可以区分输出的迭代。

根据 db2cos 脚本中指定的命令，db2cos 输出文件将包含不同信息。如果未更改缺省脚本，那么将显示类似于以下的条目（后跟详细 db2pd 输出）:

```
2005-10-14-10.56.21.523659
PID: 782348                    TID: 1             PROC: db2cos
实例: db2inst1                 节点: 0            数据库: SAMPLE
APPHDL  :                      APPID: *LOCAL.db2inst1.051014155507
函数: 操作系统服务，sqloEDUCodeTrapHandler，探测点: 999
事件: 从操作系统服务 sqloEDUCodeTrapHandler 调用
          /home/db2inst1/sqllib/bin/db2cos 捕获到陷阱
实例 db2inst1 使用 64 位和 DB2 代码发行版 SQL09010
...
操作系统信息:

操作系统名称: AIX
节点名: n1
版本: 5
```

```
发行版: 2
机器: 000966594C00

...
```

db2diag 日志文件还将包含与发生位置有关的条目。例如:

```
2005-10-14-10.42.17.149512-300 I19441A349         级别: 事件
PID: 782348               TID: 1           PROC: db2sysc
实例: db2inst1            节点: 000
函数: DB2 UDB, 跟踪服务, pdInvokeCalloutScript, 探测点: 10
开始: 从操作系统服务 sqloEDUCodeTrapHandler 调用 /home/db2inst1/sqllib/bin/db2cos
2005-10-14-10.42.23.173872-300 I19791A310         级别: 事件
PID: 782348               TID: 1           PROC: db2sysc
实例: db2inst1            节点: 000
函数: DB2 UDB, 跟踪服务, pdInvokeCalloutScript, 探测点: 20
停止: 完成 /home/db2inst1/sqllib/bin/db2cos 调用

2005-10-14-10.42.23.519227-300 E20102A509         级别: 严重
PID: 782348               TID: 1           PROC: db2sysc
实例: db2inst1            节点: 000
函数: DB2 UDB, 操作系统服务, sqloEDUCodeTrapHandler, 探测点: 10
消息: ADM0503C  发生了意外内部处理错误。已关闭所有与此实例相关联的
          DB2 进程。
          已经记录了诊断信息。有关进一步的帮助, 与"IBM 支持机构"联系。
2005-10-14-10.42.23.520111-300 E20612A642         级别: 严重
PID: 782348               TID: 1           PROC: db2sysc
实例: db2inst1            节点: 000
函数: DB2 UDB, 操作系统服务, sqloEDUCodeTrapHandler, 探测点: 20
DATA #1: 接收到信号编号, 4 字节
11
DATA #2: 信号信息, 64 字节
0x0FFFFFFFFFFFD5C0 : 0000 000B 0000 0000 0000 0009 0000 0000    ................
0x0FFFFFFFFFFFD5D0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5E0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5F0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
```

转储文件

转储文件是在发生错误时创建的，它包含将有助于诊断问题（例如，内部控制块）的其他信息。写至转储文件的每个数据项都具有与其相关联的时间戳记，以帮助进行问题确定。转储文件采用二进制格式，旨在供 IBM 软件支持机构代表使用。

创建或追加转储文件时，会在 db2diag 日志文件中建立一个条目，指示所写数据的时间和类型。这些 db2diag 日志条目类似于以下所示:

```
2007-05-18-12.28.11.277956-240 I24861950A192 LEVEL: Severe
PID:1056930 TID:225448 NODE:000 Title: dynamic memory buffer
Dump File:/home/svtdbm5/sqllib/db2dump/1056930.225448.000.dump.bin
```

注: 对于分区数据库环境，文件扩展名标识分区号。例如，以下条目指示转储文件是由在分区 10 上运行的 DB2 进程创建的:

```
Dump File: /home/db2/sqllib/db2dump/6881492.2.010.dump.bin
```

首次出现数据捕获信息

首次出现数据捕获（FODC）是用来捕获有关 DB2 实例的基于方案的数据的过程。DB2 用户可根据特定症状手动调用 FODC，也可在检测到预定义方案或症状时自动调用 FODC。此信息减少了再现错误以获取诊断信息的需要。

可在下列文件中找到 FODC 信息:

管理通知日志（"*instance_name*.nfy"）

- 操作系统: 所有操作系统
- 缺省位置:
  - Linux 和 UNIX: 位于 **diagpath** 数据库管理器配置参数所指定的目录中。
  - Windows: 使用事件查看器工具（**开始 > 控制面板 > 管理工具 > 事件查看器**）
- 创建实例时自动创建。
- 发生重大事件时，DB2 将信息写入管理通知日志。该信息供数据库和系统管理员使用。记录在此文件中的消息类型由 **notifylevel** 配置参数确定。

  注: 如果 **diagsize** 数据库管理器配置参数设置为非零值，那么单一管理通知日志文件行为（*instance_name*.nfy）将更改为旋转日志行为（*instance_name.N*.nfy）。

DB2 诊断日志（db2diag.log）

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所标识的目录中。
- 创建实例时自动创建。
- 此文本文件包含关于实例遇到的错误和警告的诊断信息。此信息用于进行故障诊断，且旨在供 IBM 软件支持机构的技术员使用。记录在此文件中的消息类型由 **diaglevel** 数据库管理器配置参数确定。

  注: 如果 **diagsize** 数据库管理器配置参数设置为非零值，那么单一诊断日志文件行为（db2diag.log）将更改为旋转日志行为（db2diag.*N*.log）。

DB2 管理服务器（DAS）诊断日志（db2dasdiag.log）

- 操作系统: 所有操作系统
- 缺省位置:
  - Linux 和 UNIX: 位于 `DASHOME/das/dump` 中，其中 `DASHOME` 是 DAS 所有者的主目录
  - Windows: 位于 DAS 主目录的"dump"文件夹中。例如，`C:\Program Files\IBM\SQLLIB\DB2DAS00\dump`
- 创建 DAS 时自动创建。
- 此文本文件包含关于 DAS 遇到的错误和警告的诊断信息。

DB2 事件日志（db2eventlog.xxx，其中 xxx 是数据库分区号）

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所指定的目录中。
- 创建实例时自动创建。
- DB2 事件日志文件是数据库管理器中发生的基础结构级事件的循环日志。该文件大小固定，并且充当在实例运行时记录的特定事件的循环缓冲区。每次停止实例时，就会替换先前的事件日志，而不是追加。如果实例捕获，那么还会生成 db2eventlog.XXX.crash 文件。这些文件旨在供 IBM 软件支持机构使用。

DB2 调出脚本（db2cos）输出文件

- 操作系统：所有操作系统
- 缺省位置：位于 **diagpath** 数据库管理器配置参数所指定的目录中。
- 如果由于 FODC 中断而执行 db2cos 脚本，那么 db2cos 输出文件将被置于在 **diagpath** 数据库管理器配置参数所指定位置中创建的 FODC 目录中。
- 出现应急启动、陷阱或分段违例时自动创建。还可以按使用 db2pdcfg 命令指定的那样在出现特定问题时创建。
- 缺省 db2cos 脚本将调用 db2pd 命令以打开方式收集信息。根据 db2cos 脚本所包含命令（例如操作系统命令和其他 DB2 诊断工具）的不同，db2cos 输出文件的内容也将有所变化。有关使用 db2cos 脚本执行的工具的更多详细信息，请在文本编辑器中打开该脚本文件。
- `bin/` 目录中提供了 db2cos 脚本。在 UNIX 上，此目录是只读的。要对此脚本创建您自己的可修改版本，请将 db2cos 脚本复制至 `adm/` 目录。您可自由修改脚本的此版本。如果脚本在 `adm/` 目录中，那么会运行该版本。否则，会运行 `bin/` 目录中的缺省版本。

转储文件

- 操作系统：所有操作系统
- 缺省位置：位于 **diagpath** 数据库管理器配置参数所指定的目录中。
- 如果在 FODC 中断期间转储这些文件，那么它们将被放在 FODC 目录中。
- 出现特定问题情况时自动创建。
- 对于某些错误情况，会将附加信息记录在以失败进程标识命名的二进制文件中。这些文件旨在供 IBM 软件支持机构使用。

陷阱文件

- 操作系统：所有操作系统
- 缺省位置：位于 **diagpath** 数据库管理器配置参数所指定的目录中。
- 如果在 FODC 中断期间转储这些文件，那么它们将被放在 FODC 目录中。
- 实例异常结束时自动创建。还可以使用 db2pd 命令创建。
- 如果数据库管理器由于陷阱、分段违例或异常而不能继续处理，那么会生成陷阱文件。

核心文件

- 操作系统：Linux 和 UNIX
- 缺省位置：位于 **diagpath** 数据库管理器配置参数所指定的目录中。
- 如果在 FODC 中断期间转储这些文件，那么它们将被放在 FODC 目录中。
- DB2 实例异常终止时由操作系统创建。
- 除其他内容以外，核心映像还将包含 DB2 的大多数或全部内存分配，这可能是描述问题时必需的内容。

根据常见中断问题收集诊断信息

发生影响实例的中断时，可自动将诊断信息收集到包中。包中的信息也可手动创建。

如果使用 DB2 实例和数据库时发生问题，应收集发生问题时的数据。首次出现数据收集（FODC）是一个术语，用于描述 DB2 环境中发生问题时应采取的操作。应使用

db2pdcfg 工具设置 DB2FODC 注册表变量中的选项，以控制中断期间收集的数据。应使用 db2pdcfg -fodc 来更改 DB2FODC 注册表变量选项。这些选项会影响 FODC 情况中有关数据捕获的数据库系统行为。

自动收集诊断信息

数据库管理器调用 db2fodc 命令以自动执行首次出现数据捕获（FODC）。

为了使中断与 DB2 诊断日志和其他故障诊断文件相关，会同时将诊断消息写至管理通知日志和 db2diag 日志文件。诊断消息包括 FODC 目录名和创建 FODC 目录时的时间戳记。FODC 包描述文件在新的 FODC 目录中。

表 83. 自动 FODC 类型和程序包

程序包	描述	调用类型	所执行脚本
FODC_Trap_*timestamp*	发生实例范围内的陷阱	自动	db2cos_trap(.bat)
FODC_Panic_*timestamp*	引擎检测到无关的异常情况，决定不继续执行	自动	db2cos_trap(.bat)
FODC_BadPage_*timestamp*	检测到坏页	自动	db2cos_datacorruption(.bat)
FODC_DBMarkedBad_*timestamp*	数据库发生错误，已被标记为“有问题”	自动	db2cos(.bat)
FODC_IndexError_*timestamp*_PID_EDUID_Node#	发生 EDU 范围内的索引错误。（db2cos_indexerror_short(.bat) 和/或 db2cos_indexerror_long(.bat) 将转储到该目录。）	自动	无

诊断信息的手动收集

首次出现数据收集命令（db2fodc）用于收集有关可能中止的信息或出现严重性能问题时的信息。运行 db2fodc 命令后，会在当前诊断路径下自动创建新目录 FODC_hang_*timestamp*。会运行 db2cos_hang 脚本。此脚本控制将收集并放在 FODC 子目录中的数据集合。FODC 子目录是否存在取决于 db2fodc 命令的运行方式或 DB2 注册表变量的配置。

表 84. 手动 FODC 类型和程序包

程序包	描述	调用类型	所执行脚本
FODC_Hang_*timestamp*	用户调用 db2fodc -hang 以收集用于对悬而未决情况进行故障诊断（或严重影响性能）的数据	手动	db2cos_hang(.bat)
FODC_Perf_*timestamp*	用户调用 db2fodc -perf 以收集用于对性能进行故障诊断的数据	手动	db2cos_perf(.bat)

表 84. 手动 FODC 类型和程序包 (续)

程序包	描述	调用类型	所执行脚本
位于 **FODC_IndexError_*timestamp*_PID_EDUID_Node#** 中的脚本	用户可发出 db2fodc -indexerror *FODC_IndexError_directory* [basic | full]（缺省值为 basic）以在脚本中调用 db2dart 命令。 在 DPF 上，请使用 db2_all "<<+node#< db2fodc -indexerror *FODC_IndexError_directory* [basic | full]"。node# 是 *FODC_IndexError_directory* 目录名中的最后一个数字。将 db2fodc -indexerror 与 db2_all 命令配合使用时，需要使用绝对路径。	手动	db2cos_indexerror_long(.bat)、db2cos_indexerror_short(.bat)

配置诊断信息的自动收集

必须先指示数据库管理器要采取的操作，数据库管理器才能自动执行操作。

系统会设置一些标志，来指示数据库操作期间遇到错误或警告时数据库管理器应采取的操作。要采取的操作包括:

- 在 db2diag 日志文件中生成堆栈跟踪。（缺省）
- 运行标注脚本 db2cos。（缺省）
- 停止跟踪（db2trc）命令。

更改首次出现数据捕获（FODC）选项

使用“配置 DB2 数据库的问题确定行为”（db2pdcfg）命令来更改首次出现数据捕获（FODC）选项。FODC 选项是使用 db2pdcfg 工具在 DB2FODC 注册表变量中设置的。这些选项会影响 FODC 情况中有关数据捕获的数据库系统行为。

作为 FODC 的一部分收集的数据及其放置

根据实例内的中断类型，首次出现数据捕获（FODC）会导致创建子目录及收集的特定内容。将创建一系列子目录及文件和日志的集合。

将在 FODC 目录下创建下列一个或多个子目录:

- DB2CONFIG，包含 DB2 配置输出和文件
- DB2PD，包含 db2pd 输出或输出文件
- DB2SNAPS，包含 DB2 快照
- DB2TRACE，包含 DB2 跟踪
- OSCONFIG，包含操作系统配置文件
- OSSNAPS，包含操作系统监视器信息
- OSTRACE，包含操作系统跟踪

根据 DB2FODC 的配置或 db2fodc 的运行方式，这些目录并非一直存在。

根据中断类型，可在 FODC 目录和子目录中找到以下内容:

- 陷阱文件
- 中断时进行数据捕获期间生成的和不同组件完成的所有不同二进制转储文件和纯文本转储文件
- db2evlog 的事件日志文件
- 中断时启动了跟踪的情况下生成的 DB2 跟踪转储
- 包含核心文件的目录
- DB2FODC 日志文件:
  - 只有一个“日志”文件用于手动 FODC: db2fodc_hang.log（用于中止）或 db2fodc_badpage.log（用于错误页面）
- 数据毁坏的相关信息
  - 进程信息: ps（在 UNIX 上）和 db2pd -edus 输出
  - 当前由 db2support 收集的其他信息（可选）:
    - errpt -a output（在 AIX 上）
    - UNIX 平台上的系统日志。例如，/var/adm/messages（用于 SunOS）和 /var/adm/syslog.log（在 HP/UX 上）。只要能够收集这些文件，就会完成此操作（在 Linux 上，必须有 root 用户访问权才能复制 syslog 文件）。

自动 FODC 数据生成

出现中断并自动启用首次出现数据捕获（FODC）后，会根据症状收集数据。收集的数据特定于诊断该中断所需的信息。

用于标记中断原因的一条或多条消息，包括定义为“关键”的消息。

陷阱文件包含如下信息:

- 可用的虚拟存储量
- 发生陷阱时与产品的配置参数及注册表变量相关的值
- 发生陷阱时 DB2 产品使用的估计内存量
- 提供中断上下文的信息

原始堆栈转储可能包括在 ASCII 陷阱文件中。

特定于数据库管理器内的组件的转储文件存储在相应 FODC 包目录中。

DB2 Query Patroller 和首次出现数据捕获（FODC）

如果发现需要调查 DB2 Query Patroller 问题，有一些日志会包含有关可能遇到的问题的可探测原因的信息。

qpdiag.log

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所标识的目录中。
- Query Patroller 系统变为活动状态时自动创建。
- 包含 Query Patroller 的参考记录和诊断记录。此信息用于故障诊断,且旨在供 IBM 软件支持机构使用。

qpmigrate.log

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所标识的目录中。
- 由 qpmigrate 实用程序自动创建。安装 Query Patroller（如果指定要运行 Query Patroller 的现有数据库）时可隐式运行 qpmigrate 命令，或安装之后显式运行该命令。
- 将 Query Patroller 从一个版本迁移到另一个版本时捕获信息和错误消息。它由 Query Patroller 管理员使用。

qpsetup.log

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所标识的目录中。
- 由 qpsetup 实用程序自动创建。安装 Query Patroller（如果指定要运行 Query Patroller 的现有数据库）时可隐式运行 qpsetup 命令，或安装之后显式运行该命令。
- 捕获 qpsetup 实用程序运行时的信息和错误消息。它由 Query Patroller 管理员使用。

qpuser.log

- 操作系统: 所有操作系统
- 缺省位置: 位于 **diagpath** 数据库管理器配置参数所标识的目录中。
- Query Patroller 系统变为活动状态时自动创建。
- 包含有关 Query Patroller 的参考消息；例如，指示 Query Patroller 何时停止和启动。它由 Query Patroller 管理员使用。

使用首次出现数据捕获（FODC）的监视和审计工具

如果发现需要调查监视和审计工具问题，有一些日志会包含有关可能遇到的问题的可探测原因的信息。

DB2 审计日志（"db2audit.log"）

- 操作系统: 所有操作系统
- 缺省位置:
  - Windows: 位于 `$DB2PATH\instance_name\security` 目录中
  - Linux 和 UNIX: 在 `$HOME\sqllib\security` 目录中，其中 $HOME 是实例所有者的主目录
- 启动 db2audit 工具时创建。
- 包含 DB2 审计工具为一系列预定义的数据库事件生成的审计记录。

DB2 控制器日志（"*mylog.x*"，其中 *x* 是控制器在其上运行的数据库分区号）

- 操作系统: 所有操作系统
- 缺省位置:
  - Windows: 位于 `$DB2PATH\instance_name\log` 目录
  - Linux 和 UNIX: 位于 `$HOME\sqllib\log` 目录中，其中 $HOME 是实例所有者的主目录
- 使用 GOVERNOR 实用程序时创建。在 db2gov 命令中指定了基本日志文件名。

- 有关控制器守护程序执行的操作（例如，强制执行一个应用程序、读取控制器配置文件、启动或结束该实用程序）的记录信息以及错误和警告。

事件监视器文件（例如，“00000000.evt”）

- 操作系统：所有操作系统
- 缺省位置：创建文件事件监视器时，所有事件记录都将被写入 CREATE EVENT MONITOR 语句指定的目录中。
- 事件发生时由事件监视器生成。
- 包含和事件监视器相关联的事件记录。

使用首次出现数据捕获（FODC）的图形工具

如果发现需要调查命令编辑器、数据仓库中心或信息目录中心问题，有一些日志会包含有关可能遇到的问题的可探测原因的信息。

命令编辑器日志

- 操作系统：所有操作系统
- 缺省位置：使用 DB2 工具栏的命令编辑器页指定此日志文件的名称和位置。如果未指定路径，那么此日志将存储在 Windows 上的 `$DB2PATH\sqllib\tools` 目录中或 Linux 和 UNIX 上的 `$HOME/sqllib/tools` 目录中，其中 HOME 是实例所有者的主目录。
- 当选择命令编辑器中的**文件的日志命令历史记录**并指定文件和位置时创建。
- 包含命令编辑器的命令和语句执行历史记录。

数据仓库中心 IWH2LOGC.log 文件

- 操作系统：所有操作系统
- 缺省位置：位于 VWS_LOGGING 环境变量指定的目录中。在 Windows 上，缺省路径为 `$DB2PATH\sqllib\logging` 目录；在 Linux 和 UNIX 上，缺省路径为 `$HOME/sqllib/logging` 目录，其中 HOME 是实例所有者的主目录。
- 记录器停止时由数据仓库中心自动创建。
- 包含记录器停止时无法发送的由数据仓库中心和 OLE 服务器编写的消息。可使用数据仓库中心中的“日志查看器”窗口查看此日志。

数据仓库中心 IWH2LOG.log 文件

- 操作系统：所有操作系统
- 缺省位置：位于 VWS_LOGGING 环境变量指定的目录中。在 Windows 上，缺省路径为 `$DB2PATH\sqllib\logging` 目录；在 Linux 和 UNIX 上，缺省路径为 `$HOME/sqllib/logging` 目录，其中 HOME 是实例所有者的主目录。
- 数据仓库中心无法启动或激活了跟踪时由数据仓库中心自动创建。
- 包含数据仓库中心记录器无法启动并无法写入数据仓库中心日志（IWH2LOGC.log）时的诊断信息。可使用数据仓库中心中的“日志查看器”窗口查看此日志。

数据仓库中心 IWH2SERV.log 文件

- 操作系统：所有操作系统
- 缺省位置：位于 VWS_LOGGING 环境变量指定的目录中。在 Windows 上，缺省路径为 `$DB2PATH\sqllib\logging` 目录；在 Linux 和 UNIX 上，缺省路径为 `$HOME/sqllib/logging` 目录，其中 HOME 是实例所有者的主目录。

- 由数据仓库中心服务器跟踪工具自动创建。
- 包含服务器跟踪工具创建的数据仓库中心启动消息和日志消息。可使用数据仓库中心中的“日志查看器”窗口查看此日志。

信息目录中心标记文件 EXPORT 日志

- 操作系统: 所有操作系统
- 缺省位置: 信息目录中心中“导出”工具的**选项**选项卡指定的已导出标记文件路径和日志文件名
- 由信息目录中心中的“导出”工具生成
- 包含标记文件导出信息，例如导出过程开始和停止的时间和日期。它还包括在导出操作期间遇到的任何错误消息。

信息目录中心标记文件 IMPORT 日志

- 操作系统: 所有操作系统
- 缺省位置: 信息目录中心中“导入”工具指定的已导入标记文件路径和日志文件名
- 由信息目录中心中的“导入”工具生成
- 包含标记文件导入历史记录信息，例如导入过程开始和停止的时间和日期。它还包括在导入操作期间遇到的任何错误消息。

内部返回码

有两种类型的内部返回码: ZRC 值和 ECF 值。这些返回码一般仅在供 IBM 软件支持机构使用的诊断工具中可视。

例如，它们显示在DB2 跟踪输出和 db2diag 日志文件中。

ZRC 和 ECF 值基本上具有相同的作用，但格式上稍有不同。每个 ZRC 值都具有以下特征:

- 类名
- 组件
- 原因码
- 相关联的 SQLCODE
- SQLCA 消息标记
- 描述

但是，ECF 值由以下部分组成:

- 集名
- 产品标识
- 组件
- 描述

ZRC 和 ECF 值通常为负数，并用于表示错误状况。ZRC 值根据它们表示的错误类型进行分组。这些分组称为“类”。例如，具有以“SQLZ_RC_MEMHEP”开头的名称的 ZRC 值通常是与内存不足相关的错误。相似地，ECF 值被分组为“集”。

以下是包含 ZRC 值的 db2diag 日志文件条目的示例:

```
2006-02-13-14.34.35.965000-300    I17502H435       级别: 错误
PID: 940                  TID: 660         PROC: db2syscs.exe
实例: DB2                 节点: 000        数据库: SAMPLE
APPHDL: 0-1433            APPID: *LOCAL.DB2.050120082811
函数: DB2 UDB, 数据包含, sqlpsize, 探测点: 20
返回码: ZRC=0x860F000A=-2045837302=SQLO_FNEX"找不到文件。"
          DIA8411C 找不到文件 ""。
```

可使用 db2diag 命令获取有关此 ZRC 值的完整详细信息, 例如:

```
c:\>db2diag -rc 0x860F000A

输入 ZRC 字符串"0x860F000A"被解析为 0x860F000A (-2045837302)。

要映射的 ZRC 值: 0x860F000A (-2045837302)
        V7 等同 ZRC 值: 0xFFFFE60A (-6646)

ZRC 类:
        关键介质错误 (类索引: 6)
组件:
        SQLO; 操作系统服务 (组件索引: 15)
原因码:
        10 (0x000A)

标识:
        SQLO_FNEX
        SQLO_MOD_NOT_FOUND
标识 (不带组件):
        SQLZ_RC_FNEX

描述:
        未找到文件。

相关信息:
        Sqlcode -980
SQL0980C  发生磁盘错误。无法处理后续的 SQL 语句。
sqlca 标记数: 0
        对话框消息号: 8411
```

如果发出命令 db2diag -rc -2045837302 或 db2diag -rc SQLO_FNEX, 将返回相同信息。

ECF 返回码的输出示例如下:

```
c:\>db2diag -rc 0x90000076

输入 ECF 字符串"0x90000076"被解析为 0x90000076 (-1879048074)

要映射的 ECF 值: 0x90000076 (-1879048074)

ECF 集:
        setecf (集索引: 1)
产品:
        DB2 Common
组件:
        OSSe
代码:
        118 (0x0076)

标识:
        ECF_LIB_CANNOT_LOAD

描述:
        不能装入指定的库。
```

db2diag 命令输出中最有价值的故障诊断信息是描述和相关信息 (仅对于 ZRC 返回码)。

要获取 ZRC 或 ECF 值的完整列表，请分别使用命令 db2diag -rc zrc 和 db2diag -rc ecf。

消息简介

假设您熟悉安装了 DB2 的操作系统的功能。可使用以下各章中包含的信息来识别错误或问题，并通过相应的恢复操作来解决问题。此信息还可用于了解消息的生成位置和记录位置。

消息结构

消息帮助描述了产生消息的原因以及响应该消息时应该执行的任何操作。

消息标识依次包含以下各项：由三个字符组成的消息前缀、四位或五位数字的消息号以及由单个字母表示的后缀。例如，*SQL1042C*。要获取消息前缀列表，请参阅『调用消息帮助』和第 485 页的『其他 DB2 消息』。由单个字母表示的后缀描述错误消息的严重性。

通常，以 *C* 结束的消息标识指示严重消息；以 *E* 结束的消息标识指示紧急消息；以 *N* 结束的消息标识指示错误消息；以 *W* 结束的消息标识指示警告消息；以 *I* 结束的消息标识指示参考消息。

对于 ADM 消息，以 *C* 结束的消息标识指示严重消息；以 *E* 结束的消息标识指示紧急消息；以 *W* 结束的消息标识指示重要消息；以 *I* 结束的消息标识指示参考消息。

对于 SQL 消息，以 *C* 结束的消息标识指示关键系统错误；以 *N* 结束的消息标识指示错误消息；以 *W* 结束的消息标识指示警告或参考消息。

一些消息包括标记，有时也称为消息变量。当包含标记的消息由 DB2 生成时，每个标记都会替换为特定于所遇到错误状态的值，以帮助用户对产生该错误消息的原因进行诊断。例如，DB2 消息 SQL0107N 如下所示：

- 来自命令行处理器：

  SQL0107N 名称“<name>”太长。最大长度为“<length>”。
- 来自 DB2 信息中心：

  SQL0107N 名称 *name* 太长。最大长度为 *length*。

此消息包括以下两个标记：“<name>”和“<length>”。当在运行时生成此消息时，这些消息标记将分别替换为导致错误的对象的实际名称，以及对于该对象类型允许的最大长度。

在一些情况下，标记不适用于错误的特定实例，会改为返回值 *N，例如：

```
SQL20416N 未能将提供的值（"*N"）转换为安全标号。对于策略标识为"1"的安全策略，
标号的长度应该为"8"个字符。该值的长度为"0"个字符。SQLSTATE=23523
```

调用消息帮助

可从命令行处理器访问下列 DB2 消息：

前缀　　描述

ADM 由许多 DB2 组件生成的消息。这些消息会被写入管理通知日志文件，并且旨在向系统管理员提供其他信息。

AMI 由 MQ 应用程序消息传递接口生成的消息

ASN 由 DB2 复制生成的消息

CCA 由配置助手生成的消息

CLI 由调用级接口生成的消息

DBA 由数据库管理工具生成的消息

DBI 由安装和配置生成的消息

DBT 由数据库工具生成的消息

DB2 由命令行处理器生成的消息

DQP 由 Query Patroller 生成的消息

EAS 由嵌入式应用程序服务器生成的消息

EXP 由 EXPLAIN 实用程序生成的消息

GSE 由 DB2 Spatial Extender 生成的消息

LIC 由 DB2 License Manager 生成的消息

MQL 由 MQ Listener 生成的消息

SAT 在卫星环境中生成的消息

SPM 由同步点管理器生成的消息

SQL 在检测到警告或错误状态时由数据库管理器生成的消息

XMR 由 XML 元数据存储库生成的消息。

要调用消息帮助，请打开命令行处理器并输入:

```
? XXXnnnnn
```

其中 *XXX* 表示有效消息前缀，而 *nnnnn* 则表示有效消息号。

通过发出以下命令，可检索与给定 SQLSTATE 值关联的消息文本:

```
? nnnnn
```

或者

```
? nn
```

其中 *nnnnn* 是一个五位的 SQLSTATE（字母数字），而 *nn* 则是一个两位的 SQLSTATE 类代码（SQLSTATE 值的前两位）。

注: 作为 **db2** 命令的参数接受的消息标识是不区分大小写的。此外，由单个字母表示的后缀为可选且被忽略。

因此，以下命令将产生相同的结果:

- ? SQL0000N
- ? sql0000
- ? SQL0000w

要在基于 UNIX 的系统的命令行上调用消息帮助，请输入:

```
db2 "? XXXnnnnn"
```

其中 *XXX* 表示有效消息前缀，而 *nnnnn* 则表示有效消息号。

如果消息文本太长而不能在屏幕上完全显示，请使用以下命令（在基于 Unix 的系统上以及其他支持“more”的系统上）:

```
db2 "? XXXnnnnn" | more
```

其他 DB2 消息

一些 DB2 组件返回不以在线方式提供或没有在此手册中描述的消息。其中一些消息前缀可包括下列各项:

AUD 由 DB2 审计设施生成的消息。

DIA 由许多 DB2 组件生成的诊断消息。这些消息会被写入 db2diag 日志文件，并且旨在当研究错误时向用户和 DB2 服务人员提供其他信息。

GOV 由 DB2 GOVERNOR 实用程序生成的消息。

在大多数情况下，这些消息会提供足够信息来确定产生警告或错误的原因。有关生成了消息的命令或实用程序的更多信息，请参阅在其中对该命令或实用程序进行了说明的相应手册。

其他消息来源

当在系统上运行其他程序时，接收到的消息可能带有在本参考资料中未提到的前缀。

有关这些消息的信息，请参阅针对该程序产品提供的信息。

特定于平台的错误日志信息

在 DB2 之外也提供了许多其他文件和实用程序，可用于帮助分析问题。通常，在确定问题的根本原因时，它们和 DB2 文件一样重要。

其他文件和实用程序允许访问与下列各方面有关的日志和跟踪中包含的信息。

- 操作系统
- 应用程序和第三方供应商
- 硬件

除此处描述的一些领域之外，还有许多其他领域包含这种信息，这取决于您的操作环境。因此，在调试系统中的问题时，要了解需要调查的所有潜在领域。

操作系统

每个操作系统都有一组自己的诊断文件，用于跟踪活动和故障。最常见的（通常也是最有用的）诊断文件是错误报告或事件日志。以下是收集此信息的方法列表:

- AIX: 使用 /usr/bin/errpt -a 命令
- Solaris: `/var/adm/messages*` 文件或 /usr/bin/dmesg 命令
- Linux: `/var/log/messages*` 文件或 /bin/dmesg 命令
- HP-UX: `/var/adm/syslog/syslog.log` 文件或 /usr/bin/dmesg 命令

- Windows: 使用系统日志文件、安全性日志文件、应用程序事件日志文件以及 `windir\drwtsn32.log` 文件（其中，windir 是 Windows 安装目录）

每个操作系统总是有更多的跟踪和调试实用程序。请参阅您的操作系统文档和支持资料，以确定提供了哪些进一步的信息。

应用程序和第三方供应商

每个应用程序都应该有它自己的记录和诊断文件。这些文件将补充 DB2 信息集，以便使您更准确地了解潜在问题领域。

硬件

硬件设备通常将信息记录到操作系统错误日志中。但是，有时需要其他信息。在这些情况下，必须确定哪些硬件诊断文件和实用程序可用于环境中的某个硬件。例如，在 DB2 报告页面不正确或某种类型的毁坏时。通常由于磁盘问题而报告此类错误，在这种情况下，必须调查硬件诊断文件。请参阅您的硬件文档和支持资料，以确定提供了哪些进一步的信息。

某些信息（如硬件记录中的信息）有时效性。发生错误时，应尽快从相关源收集尽可能多的信息。

总之，要完全了解并评估问题，必须从 DB2、应用程序、操作系统和底层硬件收集所有可用的信息。db2support 工具将自动收集您需要的大多数 DB2 和操作系统信息，但您仍然应该了解除此之外可能有助于调查的任何信息。

系统核心文件（Linux 和 UNIX）

如果程序异常终止，那么系统会创建一个核心文件以存储已终止进程的内存映像。诸如内存地址违例、非法指令、总线错误和用户生成的退出信号之类的错误会使核心文件转储。

核心文件名为“core”，除非使用 DB2FODC 注册表变量中的值进行配置，否则缺省情况下该文件放在 **diagpath** 数据库管理器配置参数指定的目录中。注意，系统核心文件与 DB2 陷阱文件不同。

访问系统核心文件信息（Linux 和 UNIX）

dbx 系统命令帮助您确定哪个函数导致创建系统核心文件。这是一个简单的检查，它将帮助您确定是数据库管理器有错误，或者帮助确定是操作系统还是应用程序错误导致该问题。

- 必须已安装 dbx 命令。此命令特定于操作系统：在 AIX 和 Solaris 上使用 dbx；在 HP-UX 上使用 xdb，在 Linux 上则使用 gdb。
- 在 AIX 上，使用 chdev 命令或 smitty 以确保启用了完整的核心选项。

可使用以下步骤以确定导致发生核心文件转储的函数。

1. 从 UNIX 命令提示符处输入以下命令：

   ```
   dbx program_name core_filename
   ```

   program_name 是异常终止的程序的名称，*core_filename* 是包含核心文件转储的文件的名称。*core_filename* 参数是可选的。如果不指定它，那么会使用缺省名称“core”。

2. 检查核心文件中的调用堆栈。可通过 UNIX 命令提示符发出 man dbx 来获取有关如何执行此操作的信息。
3. 要结束 dbx 命令，在 dbx 提示符下输入 quit。

以下示例显示如何使用 dbx 命令来读取称为“main”的程序的核心文件。

1. 在命令提示符下，输入:

```
dbx main
```

2. 屏幕上将出现类似如下的输出:

```
dbx version 3.1 for AIX.
Type 'help' for help.
reading symbolic information ...
[using memory image in core]
segmentation.violation in freeSegments at line 136
136         (void) shmdt((void *) pcAdress[i]);
```

3. 使核心转储的函数的名称为“freeSegments”。在 dbx 提示符下输入 where 以显示故障点的程序路径。

```
(dbx) where
freeSegments(numSegs = 2, iSetId = 0x2ff7f730, pcAddress = 0x2ff7f758, line
136
in "main.c"
main (0x1, 2ff7f7d4), line 96 in "main.c"
```

   在本示例中，freeSegments 的第 136 行出错，它是从 main.c 中的第 96 行调用的。

4. 要结束 dbx 命令，在 dbx 提示符下输入 quit。

访问事件日志（Windows）

本任务描述如何访问 Windows 事件日志。

Windows 事件日志还会提供有用的信息。虽然系统事件日志在发生 DB2 崩溃或有关系统资源的其他不明错误时好像最有用，但获取全部三种类型的事件日志仍是值得的:

- 系统
- 应用程序
- 安全性

使用 Windows 事件查看器查看事件日志。 根据您使用的 Windows 操作系统的不同，用于启动查看器的方法将会不同。
例如，要在 Windows XP 上打开“事件查看器”，请单击**开始 —> 控制面板**。选择**管理工具**，然后双击**事件查看器**。

导出事件日志（Windows）

本任务描述如何导出 Windows 事件日志。

可从 Windows 事件查看器导出两种格式的事件日志:

- 日志文件格式
- 文本或逗号定界格式。

使用 Windows 事件查看器导出事件日志。

- 您可以将日志文件格式（*.evt）数据装入回到事件查看器中（例如，在另一台工作站上进行装入）。此格式易于使用，这是因为您可以使用该查看器来切换时间顺序、过滤特定事件以及提前和推后事件。

- 可以在大多数文本编辑器中打开文本（*.txt）或逗号定界（*.csv）格式的日志。它们还避免了关于时间戳记的潜在问题。以 .evt 格式导出事件日志时，时间戳记将使用全球标准时间，并且会在查看器中转换为工作站的当地时间。如果不够仔细，可能会因为时差忽略关键事件。文本文件还易于搜索。

访问 Dr. Watson 日志文件（Windows）

本任务描述如何在 Windows 系统上访问 Dr. Watson 日志文件。

Dr. Watson 日志 `drwtsn32.log` 是系统上发生的所有异常的编年表。尽管 Dr. Watson 在评估整体系统稳定性及描述 DB2 陷阱的历史记录时很有帮助，但 DB2 陷阱文件比 Dr. Watson 日志更有用。

找到 Dr. Watson 日志文件。 缺省路径为 `<install_drive>:\Documents and Settings \All Users\Documents\DrWatson`

陷阱文件

如果 DB2 由于陷阱、分段违例或异常而不能继续处理，它就会生成陷阱文件。

DB2 接收到的所有信号或异常都会记录在陷阱文件中。陷阱文件还包含发生错误时正在运行的函数序列。此序列有时又称“函数调用堆栈”或“堆栈跟踪”。陷阱文件还包含有关捕获到信号或异常时进程的状态的其他信息。

当应用程序在运行受防护线程安全例程期间被强制停止时，也将生成陷阱文件。在进程关闭时，将发生陷阱。这不是致命错误，您不必担心。

文件位于由 **diagpath** 数据库管理器配置参数指定的目录中。

在所有平台上，陷阱文件名以进程标识（PID）开头，后跟线程标识（TID），再后跟分区号（在单分区数据库上为 000），并以“.trap.txt”结尾。

还有一些诊断陷阱，它们是在发生某些特定条件（这些条件不一定会使实例崩溃）时由代码生成的，在查看堆栈时非常有用。这些陷阱是使用 PID 以十进制格式命名的，后跟分区号（在单分区数据库中为 0）。

示例:

- `6881492.2.000.trap.txt` 是进程标识（PID）为 6881492，线程标识（TID）为 2 的陷阱文件。
- `6881492.2.010.trap.txt` 是进程和线程在分区 10 上运行的陷阱文件。

可将 db2pd 命令与 -stack all 或 -dump 选项配合使用，以根据需要生成陷阱文件。但是，一般情况下只有在 IBM 软件支持机构请求时才应完成此任务。

可使用 db2pd -stacks 或 db2pd -dumps 命令生成堆栈跟踪文件。这些文件与陷阱文件的内容相同，但仅供诊断使用。它们的名称类似于 `6881492.2.000.stack.txt`。

格式化陷阱文件（Windows）

可以使用名为 db2xprt 的命令来格式化陷阱文件（*.TRP）。它会将 DB2 数据库二进制陷阱文件格式化为人们可以阅读的 ASCII 文件。

db2xprt 工具使用 DB2 符号文件来格式化陷阱文件。这些 .PDB 文件的子集与 DB2 数据库产品包括在一起。

如果已在 **diagpath** 数据库管理器配置参数指定的目录中生成陷阱文件 “DB30882416.TRP”，那么可以按如下方式进行格式化:

```
db2xprt DB30882416.TRP DB30882416.FMT
```

第 11 章 与 IBM 软件支持机构联系

与 IBM 软件支持机构联系

IBM 软件支持机构可以帮助您解决产品缺陷。

在与 IBM 软件支持机构联系之前，贵公司必须签署有效的 IBM 软件维护合同，并且您必须得到向 IBM 提交问题的授权。有关可用的维护合同类型的信息，请参阅 *Software Support Handbook* 中的“Premium Support”，网址为：techsupport.services.ibm.com/guides/services.html。

完成下列步骤，以便与 IBM 软件支持联系并提交问题:

1. 定义问题、收集背景信息并确定问题的严重性。要获取帮助，请参阅 *Software Support Handbook* 中的“Contacting IBM”，网址为：techsupport.services.ibm.com/guides/beforecontacting.html。
2. 收集诊断信息。
3. 通过下列其中一种方法向 IBM 软件支持机构提交问题:
   - 在线方式：在 IBM Software Support Web 站点上，单击 **ESR**（电子服务请求）链接，打开 Service Request 站点：www.ibm.com/software/support/probsub.html。
   - 通过电话：要了解在您的国家/地区应该拨打的电话号码，请访问 *Software Support Handbook* 的 Contacts 页面：techsupport.services.ibm.com/guides/contacts.html。

如果您提交的问题与软件缺陷或者文档缺失或不准确相关，那么 IBM 软件支持机构将创建授权程序分析报告（APAR）。APAR 详细地描述问题。在有可能的时候，IBM 软件支持机构将提供一种变通方法，您可以实现该变通方法，直到 IBM 解决该 APAR 并交付修订为止。 IBM 每天都在 IBM Software Support Web 站点上发布已解决的 APAR，以使其他遇到同一问题的用户可以从同一解决方案中受益。

将数据提交给 IBM 软件支持机构

您可以通过 FTP 或电子服务请求（ESR）工具将数据提交给 IBM 软件支持机构。下面提供了有关指示信息。

下列步骤假定您已向 IBM 软件支持机构开放问题管理记录（PMR）。

可使用下列其中一种方法将日志文件和配置文件之类的诊断数据发送至 IBM 软件支持机构:

- FTP
- 电子服务请求（ESR）工具
- 要通过 FTP 向增强中央客户端数据存储库（EcuRep）提交文件:
  1. 将收集的数据文件打包成 ZIP 或 TAR 格式，并根据问题管理记录（PMR）标识指定压缩包的名称。

© Copyright IBM Corp. 2006, 2009

文件必须使用以下命名约定以便与 PMR 正确关联：xxxxx.bbb.ccc.yyy.yyy，其中 xxxxx 是 PMR 号，bbb 是 PMR 的分支号，ccc 是 PMR 的地域代码，而 yyy.yyy 是文件名。

2. 使用 FTP 实用程序连接至服务器 ftp.emea.ibm.com。
3. 作为用户标识“anonymous”登录并输入您的电子邮件地址作为密码。
4. 转至 `toibm` 目录。例如，cd toibm。
5. 转至其中一个特定于操作系统的子目录。例如，子目录包括 `aix`、`linux`、`unix` 或 `windows`。
6. 切换至二进制方式。例如，在命令提示符下输入 bin。
7. 使用 put 命令将文件放在服务器上。使用以下文件命名约定来指定文件的名称并将其放到服务器上。您的 PMR 将更新，以使用以下格式列示文件的存储位置：`xxxx.bbb.ccc.yyy.yyy`。（xxx 是 PMR 号，bbb 是分支，ccc 是地域代码，而 yyy.yyy 是 `tar.Z` 或 `xyz.zip` 之类的文件类型的描述。）可将文件发送至 FTP 服务器，但不能更新它们。如果以后任何时间必须更改该文件，那么必须创建新的文件名。
8. 输入 quit 命令。

- 要使用 ESR 工具提交文件，请执行下列操作：
  1. 登录 ESR。
  2. 在“欢迎”页面上的**输入报告编号**字段中输入 PMR 号，然后单击**执行**。
  3. 下滚至**附加相关文件**字段。
  4. 单击**浏览**以查找要提交至 IBM 软件支持机构的日志、跟踪或其他诊断文件的位置。
  5. 单击**提交**。文件将通过 FTP 传输至 IBM 软件支持机构，并且与您的 PMR 相关联。

有关 EcuRep 服务的更多信息，请参阅 IBM EMEA 中央客户数据存储服务。

有关 ESR 的更多信息，请参阅电子服务请求（ESR）帮助。

第 3 部分 附录

© Copyright IBM Corp. 2006, 2009

附录 A. DB2 技术信息概述

可以通过下列工具和方法获取 DB2 技术信息:

- DB2 信息中心
  - 主题（任务、概念和参考主题）
  - DB2 工具的帮助
  - 样本程序
  - 教程
- DB2 书籍
  - PDF 文件（可下载）
  - PDF 文件（在 DB2 PDF DVD 中）
  - 印刷版书籍
- 命令行帮助
  - 命令帮助
  - 消息帮助

注: DB2 信息中心主题的更新频率比 PDF 书籍或硬拷贝书籍的更新频率高。要获取最新信息，请安装可用的文档更新，或者参阅 ibm.com 上的 DB2 信息中心。

可以在线访问 ibm.com 上的其他 DB2 技术信息，如技术说明、白皮书和 IBM Redbooks 出版物。访问位于以下网址的 DB2 信息管理软件库站点: http://www.ibm.com/software/data/sw-library/。

文档反馈

我们非常重视您对 DB2 文档的反馈。如果您想就如何改善 DB2 文档提出建议，请将电子邮件发送至 db2docs@ca.ibm.com。 DB2 文档小组会阅读您的所有反馈，但不能直接答复您。请尽可能提供具体的示例，这样我们才能更好地了解您所关心的问题。如果您要提供有关具体主题或帮助文件的反馈，请加上标题和 URL。

请不要用以上电子邮件地址与 DB2 客户支持机构联系。如果您遇到文档不能解决的 DB2 技术问题，请与您当地的 IBM 服务中心联系以获得帮助。

硬拷贝或 PDF 格式的 DB2 技术库

下列各表描述 IBM 出版物中心（网址为 www.ibm.com/shop/publications/order）提供的 DB2 资料库。可从 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 下载 PDF 格式的 DB2 版本 9.7 手册的英文版本和翻译版本。

尽管这些表标识书籍有印刷版，但可能未在您所在国家或地区提供。

每次更新手册时，表单号都会递增。确保您正在阅读下面列示的手册的最新版本。

© Copyright IBM Corp. 2006, 2009

注: *DB2 信息中心*的更新频率比 PDF 或硬拷贝书籍的更新频率高。

表 85. DB2 技术信息

书名	书号	是否提供印刷版	最近一次更新时间
Administrative API Reference	SC27-2435-01	是	2009 年 11 月
Administrative Routines and Views	SC27-2436-01	否	2009 年 11 月
Call Level Interface Guide and Reference, Volume 1	SC27-2437-01	是	2009 年 11 月
Call Level Interface Guide and Reference, Volume 2	SC27-2438-01	是	2009 年 11 月
Command Reference	SC27-2439-01	是	2009 年 11 月
《数据移动指南和参考》	S151-1186-00	是	2009 年 8 月
《数据恢复及高可用性指南与参考》	S151-1187-01	是	2009 年 11 月
《数据库管理概念和配置参考》	S151-1163-01	是	2009 年 11 月
《数据库监视指南和参考》	S151-1165-00	是	2009 年 8 月
《数据库安全性指南》	S151-1188-01	是	2009 年 11 月
DB2 Text Search Guide	SC27-2459-01	是	2009 年 11 月
《开发 ADO.NET 和 OLE DB 应用程序》	S151-1167-00	是	2009 年 8 月
《开发嵌入式 SQL 应用程序》	S151-1168-00	是	2009 年 11 月
Developing Java Applications	SC27-2446-01	是	2009 年 11 月
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	否	2009 年 8 月
开发用户定义的例程（SQL 和外部例程）	S151-1169-00	是	2009 年 11 月
《数据库应用程序开发入门》	G151-1170-00	是	2009 年 11 月
《Linux 和 Windows 上的 DB2 安装和管理入门》	G151-1172-00	是	2009 年 8 月
《全球化指南》	S151-1189-00	是	2009 年 8 月
《安装 DB2 服务器》	G151-1174-01	是	2009 年 11 月
《安装 IBM 数据服务器客户端》	G151-1175-00	否	2009 年 8 月
《消息参考第 1 卷》	S151-1182-01	否	2009 年 11 月
《消息参考第 2 卷》	S151-1183-01	否	2009 年 11 月

表 85. DB2 技术信息 (续)

书名	书号	是否提供印刷版	最近一次更新时间
《Net Search Extender 管理和用户指南》	S151-1185-01	否	2009 年 11 月
《分区和集群指南》	S151-1190-01	是	2009 年 11 月
《pureXML 指南》	S151-1180-01	是	2009 年 11 月
Query Patroller Administration and User's Guide	SC27-2467-00	否	2009 年 8 月
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	否	2009 年 8 月
《SQL 过程语言: 应用程序启用和支持》	S151-1171-00	是	2009 年 8 月
SQL Reference, Volume 1	SC27-2456-01	是	2009 年 11 月
SQL Reference, Volume 2	SC27-2457-01	是	2009 年 11 月
《故障诊断和调整数据库性能》	S151-1164-01	是	2009 年 11 月
《升级到 DB2 版本 9.7》	S151-1173-01	是	2009 年 11 月
《Visual Explain 教程》	S151-1184-00	否	2009 年 8 月
《DB2 版本 9.7 新增内容》	S151-1179-01	是	2009 年 11 月
Workload Manager Guide and Reference	SC27-2464-00	是	2009 年 8 月
《XQuery 参考》	S151-1181-01	否	2009 年 11 月

表 86. 特定于 DB2 Connect 的技术信息

书名	书号	是否提供印刷版	最近一次更新时间
《安装和配置 DB2 Connect 个人版》	S151-1177-01	是	2009 年 11 月
《安装和配置 DB2 Connect 服务器》	S151-1178-01	是	2009 年 11 月
《DB2 Connect 用户指南》	S151-1176-01	是	2009 年 11 月

表 87. Information Integration 技术信息

书名	书号	是否提供印刷版	最近一次更新时间
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	是	2009 年 8 月
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	是	2009 年 8 月

表 87. Information Integration 技术信息 (续)

书名	书号	是否提供印刷版	最近一次更新时间
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	否	2009 年 8 月
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	是	2009 年 8 月
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	是	2009 年 8 月

订购印刷版的 DB2 书籍

如果您需要印刷版的 DB2 书籍，可以在许多（但不是所有）国家或地区在线购买。无论何时都可以从当地的 IBM 代表处订购印刷版的 DB2 书籍。请注意，*DB2 PDF 文档* DVD 上的某些软拷贝书籍没有印刷版。例如，*DB2 消息参考*的任何一卷都没有提供印刷版书籍。

只要支付一定费用，就可以从 IBM 获取 DB2 PDF 文档 DVD，该 DVD 包含许多 DB2 书籍的印刷版。 根据您下订单的位置，您可能能够从 IBM 出版物中心在线订购书籍。如果在线订购在您所在国家或地区不可用，您始终可以从当地的 IBM 代表处订购印刷版 DB2 书籍。注意，并非 DB2 PDF 文档 DVD 上的所有书籍都有印刷版。

注：最新最完整的 DB2 文档保留在 DB2 信息中心中，网址为：http://publib.boulder.ibm.com/infocenter/db2luw/v9r7。

要订购印刷版的 DB2 书籍：

- 要了解您是否可从所在国家或地区在线订购印刷版的 DB2 书籍，可查看 IBM 出版物中心站点，网址为：http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问出版物订购信息，然后再按照针对您所在位置的订购指示信息进行订购。
- 要从当地的 IBM 代表处订购印刷版的 DB2 书籍：
  1. 从下列其中一个 Web 站点找到当地代表处的联系信息：
     - IBM 全球联系人目录，网址为 www.ibm.com/planetwide。
     - IBM 出版物 Web 站点，网址为 http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问对应您的所在地的出版物主页。在此页面中访问“关于此站点”链接。
  2. 请在致电时说明您想订购 DB2 出版物。
  3. 请向您当地的代表提供想要订购的书籍的书名和书号。有关书名和书号的信息，请参阅第 495 页的『硬拷贝或 PDF 格式的 DB2 技术库』。

从命令行处理器显示 SQL 状态帮助

DB2 产品针对可能充当 SQL 语句结果的条件返回 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

要启动 SQL 状态帮助，请打开命令行处理器并输入:

```
? sqlstate or ? 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。

访问不同版本的 DB2 信息中心

对于 DB2 版本 9.7 主题，*DB2 信息中心* URL 为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/。

对于 DB2 版本 9.5 主题，*DB2 信息中心* URL 为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

对于 DB2 版本 9.1 主题，*DB2 信息中心* URL 为: http://publib.boulder.ibm.com/infocenter/db2luw/v9/。

对于 DB2 版本 8 主题，请转至 *DB2 信息中心* URL: http://publib.boulder.ibm.com/infocenter/db2luw/v8/。

在 DB2 信息中心中以您的首选语言显示主题

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。如果未提供主题的首选语言翻译版本，那么 DB2 信息中心将显示该主题的英文版。

- 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，请单击**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，请选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
  3. 刷新页面以便以首选语言显示 DB2 信息中心。
- 要在 Firefox 或 Mozilla 浏览器中以首选语言显示主题:
  1. 在**工具** —> **选项** —> **高级**对话框中的**语言**部分中选择按钮。“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，请单击**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，请选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。

3. 刷新页面以便以首选语言显示 DB2 信息中心。

在某些浏览器和操作系统组合上，可能还必须将操作系统的区域设置更改为您选择的语言环境和语言。

更新安装在您的计算机或内部网服务器上的 DB2 信息中心

本地安装的 DB2 信息中心必须定期进行更新。

开始前

必须已安装 DB2 版本 9.7 信息中心。有关详细信息，请参阅《安装 DB2 服务器》中的“使用 DB2 安装向导来安装 DB2 信息中心”主题。所有适用于安装信息中心的先决条件和限制同样适用于更新信息中心。

关于此任务

可自动或手动更新现有 DB2 信息中心:

- 自动更新 - 更新现有信息中心功能和语言。自动更新的一个优点是在更新期间，信息中心不可用的时间最短。另外，自动更新可设置为作为定期运行的其他批处理作业的一部分运行。
- 手动更新 - 应该在更新过程期间要添加功能或语言时使用。例如，如果本地信息中心最初安装的是英语和法语版，而现在还要安装德语版；那么手动更新将安装德语版，并更新现有信息中心的功能和语言。但是，手动更新要求您手动停止、更新和重新启动信息中心。在整个更新过程期间信息中心不可用。

过程

此主题详细说明了自动更新的过程。有关手动更新的指示信息，请参阅“手动更新安装在您的计算机或内部网服务器上的 DB2 信息中心”主题。

要自动更新安装在您的计算机或内部网服务器上的 DB2 信息中心:

1. 在 Linux 操作系统上,
   a. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 `/opt/ibm/db2ic/V9.7` 目录中。
   b. 从安装目录浏览至 `doc/bin` 目录。
   c. 运行 `ic-update` 脚本:

      ```
      ic-update
      ```
2. 在 Windows 操作系统上,
   a. 打开命令窗口。
   b. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 `<Program Files>\IBM\DB2 Information Center\Version 9.7` 目录中，其中 `<Program Files>` 表示 Program Files 目录的位置。
   c. 从安装目录浏览至 `doc\bin` 目录。
   d. 运行 `ic-update.bat` 文件:

      ```
      ic-update.bat
      ```

结果

DB2 信息中心自动重新启动。如果更新可用，那么信息中心会显示新的以及更新后的主题。如果信息中心更新不可用，那么会在日志中添加消息。日志文件位于 doc\eclipse\configuration 目录中。日志文件名称是随机生成的编号。例如，1239053440785.log。

手动更新安装在您的计算机或内部网服务器上的 DB2 信息中心

如果已经在本地安装了 DB2 信息中心，那么您可以从 IBM 获取文档更新并安装。

关于此任务

手动更新在本地安装的 *DB2 信息中心* 要求您:

1. 停止计算机上的 *DB2 信息中心*，然后以独立方式重新启动信息中心。如果以独立方式运行信息中心，那么网络上的其他用户将无法访问信息中心，因而您可以应用更新。DB2 信息中心的工作站版本总是以独立方式运行。
2. 使用“更新”功能部件来查看可用的更新。如果有您必须安装的更新，那么请使用“更新”功能部件来获取并安装这些更新。

   注: 如果您的环境要求在一台未连接至因特网的机器上安装 *DB2 信息中心*更新，那么通过使用一台已连接至因特网并有已安装的 *DB2 信息中心*的机器将更新站点镜像至本地文件系统。如果网络中有许多用户将安装文档更新，那么可以通过在本地也为更新站点制作镜像并为更新站点创建代理来缩短每个人执行更新所需要的时间。如果提供了更新包，请使用“更新”功能部件来获取这些更新包。但是，只有在单机方式下才能使用“更新”功能部件。
3. 停止独立信息中心，然后在计算机上重新启动 *DB2 信息中心*。

注: 在 Windows 2008、Windows Vista 和更高版本上，稍后列示在此部分的命令必须作为管理员运行。要打开具有全面管理员特权的命令提示符或图形工具，请右键单击快捷方式，然后选择**以管理员身份运行**。

过程

要更新安装在您的计算机或内部网服务器上的 *DB2 信息中心*:

1. 停止 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击 **DB2 信息中心**服务，并选择**停止**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 以独立方式启动信息中心。
   - 在 Windows 上:
     a. 打开命令窗口。
     b. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 *Program Files*\IBM\DB2 Information Center\Version 9.7 目录中，其中 *Program Files* 表示 Program Files 目录的位置。
     c. 从安装目录浏览至 doc\bin 目录。
     d. 运行 help_start.bat 文件:

        ```
        help_start.bat
        ```
   - 在 Linux 上:

a. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 /opt/ibm/db2ic/V9.7 目录中。
b. 从安装目录浏览至 doc/bin 目录。
c. 运行 help_start 脚本:

```
help_start
```

系统缺省 Web 浏览器将打开以显示独立信息中心。

3. 单击**更新**按钮 ( )。(必须在浏览器中启用 JavaScript™。) 在信息中心的右边面板上，单击**查找更新**。 将显示现有文档的更新列表。
4. 要启动安装进程，请检查您要安装的选项，然后单击**安装更新**。
5. 在安装进程完成后，请单击**完成**。
6. 要停止独立信息中心，请执行下列操作:
   - 在 Windows 上，浏览至安装目录的 doc\bin 目录并运行 help_end.bat 文件:

     ```
     help_end.bat
     ```

     注: help_end 批处理文件包含安全地停止使用 help_start 批处理文件启动的进程所需的命令。不要使用 Ctrl-C 或任何其他方法来停止 help_start.bat。
   - 在 Linux 上，浏览至安装目录的 doc/bin 目录并运行 help_end 脚本:

     ```
     help_end
     ```

     注: help_end 脚本包含安全地停止使用 help_start 脚本启动的进程所需的命令。不要使用任何其他方法来停止 help_start 脚本。
7. 重新启动 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击 **DB2 信息中心**服务，并选择**启动**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv97 start
     ```

结果

更新后的 *DB2 信息中心*将显示新的以及更新后的主题。

DB2 教程

DB2 教程帮助您了解 DB2 产品的各个方面。这些课程提供了逐步指示信息。

开始之前

可从信息中心查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2help/ 。

某些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，请参阅教程。

DB2 教程

要查看教程，请单击标题。

《pureXML 指南》中的“pureXML®”

设置 DB2 数据库以存储 XML 数据以及对本机 XML 数据存储执行基本操作。

《Visual Explain 教程》中的“Visual Explain”

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

提供了很多故障诊断和问题确定信息以帮助您使用 DB2 数据库产品。

DB2 文档

故障诊断信息可在《DB2 故障诊断指南》或 *DB2 信息中心*的“数据库基础”部分中找到。可在该处找到有关如何使用 DB2 诊断工具和实用程序来隔离和找出问题的信息、某些最常见问题的解决方案以及有关如何解决使用 DB2 数据库产品时可能遇到的问题的建议。

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 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 或错误修订）、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

请访问 DB2 技术支持 Web 站点: http://www.ibm.com/software/data/db2/support/db2_9/。

条款和条件

如果符合以下条款和条件，那么授予您使用这些出版物的许可权。

个人使用: 只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

商业使用: 只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作这些出版物的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论是明示的还是暗含的。

当使用这些出版物损害了 IBM 的利益，或者根据 IBM 的规定，未正确遵守上述指导说明时，那么 IBM 保留自主决定撤销本文授予的许可权的权利。

只有您完全遵循所有适用的法律和法规，包括所有的美国出口法律和法规，您才可以下载、出口或再出口该信息。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，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关于适销和适用于某种特定用途的保证。

附录 B. 声明

本信息是为在美国提供的产品和服务编写的。有关非 IBM 产品的信息是基于首次出版此文档时的可获信息且会随时更新。

IBM 可能在其他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.

有关双字节字符集（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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

本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 站点中的资料不是此 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:

© Copyright IBM Corp. 2006, 2009

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，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，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。此样本程序“按现状”提供，且不附有任何种类的保证。对于使用此样本程序所引起的任何损坏，IBM 将不承担责任。

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*）（*年份*）. 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

IBM、IBM 徽标和 ibm.com® 是 International Business Machines Corp. 在全球范围许多管辖区域内的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。Web 站点 www.ibm.com/legal/copytrade.shtml 上的“版权和商标信息”中提供了 IBM 商标的最新列表。

下列术语是其他公司的商标或注册商标

- Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。
- Java 和所有基于 Java 的商标和徽标是 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。
- Intel®、Intel 徽标、Intel Inside®、Intel Inside 徽标、Intel® Centrino®、Intel Centrino 徽标、Celeron®、Intel® Xeon®、Intel SpeedStep®、Itanium® 和 Pentium® 是 Intel 公司或其子公司在美国和其他国家或地区的商标或注册商标。
- 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。

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]

© Copyright IBM Corp. 2006, 2009

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[N]

[P]

[Q]

[R]

[S]

[T]

[W]

[Z]

A

C

D

E

F

H

I

J

L

M

N

O

P

Q

R

S

T

U

X

Z

[特别字符]

Printed in China

S151-1164-01

Spine information:

IBM DB2 9.7 Linux 版、UNIX 版和 Windows 版　　版本 9.7

故障诊断和调整数据库性能