IBM® DB2® 通用数据库

数据恢复和高可用性指南与参考

版本 7

DATA-RCVR-00

IBM® DB2® 通用数据库

数据恢复和高可用性指南与参考

版本 7

DATA-RCVR-00

在使用本资料及其支持的产品之前，请阅读第467页的『附录K. 声明』中的一般信息。

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

通过您当地的 IBM 代表或 IBM 分部可订购出版物，或者，通过致电 1-800-879-2755（在美国）或 1-800-IBM-4YOU（在加拿大）来订购出版物。

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

目录

关于本书

本书提供了关于 IBM DB2 通用数据库 (UDB) 备份、复原和恢复实用程序的详细信息，并向您显示如何使用它们。本书还说明了高可用性的重要性并描述了几个平台上的 DB2 故障转移支持。

谁应使用本书

本手册的目标读者是数据库管理员、应用程序员以及其他负责或想了解 DB2 数据库系统上的备份、复原和恢复操作的 DB2 UDB 用户。

我们假设您已熟悉了 DB2 通用数据库、结构化查询语言 (SQL) 以及 DB2 UDB 所运行的操作系统环境。关于 DB2 UDB 的通用信息，请参阅*《管理指南》*。关于 SQL 的信息，参阅 *SQL Reference*。关于配置、调用以及使用 DB2 UDB 命令行处理器的信息，请参阅 *Command Reference*。关于 DB2 UDB 应用程序编程接口 (API) 的信息，请参阅 *Administrative API Reference*。关于创建包含 DB2 管理 API 的应用程序的通用信息，请参阅*《应用程序构建指南》*。此手册不包含安装 DB2 的指示信息，该指示信息的内容取决于您的操作系统。可在*《快速入门》*中找到适用于您的操作系统的安装信息。

此书的组织方式

包括下列主题:

数据恢复

第1章 制订一个良好的备份与恢复策略
讨论选择数据库和表空间恢复方法时要考虑的因素，包括备份和复原数据库或表空间，以及使用前滚恢复。

第2章 数据库备份
描述 DB2 备份实用程序，它用于创建数据库或表空间的备份副本。

第3章 数据库复原
描述 DB2 复原用程序，它用于重新构建先前已进行了备份的受损或毁坏的数据库或表空间。

第4章 前滚恢复
描述 DB2 前滚实用程序，它通过应用记录在数据库恢复日志文件中的事务，恢复数据库。

附录H. 数据库恢复的用户出口

讨论可如何将用户出口程序与数据库日志文件一起使用，并描述了某些样本用户出口程序。

附录I. 供应商产品的备份与复原 API

描述了 API 的函数和使用，这些 API 使 DB2 提供了与其他供应商软件的接口。

第1部分 数据恢复

第1章 制订一个良好的备份与恢复策略

数据库可能会因为硬件和／或软件故障而不可用。您可能会不时遇到存储问题、电源中断、应用程序故障以及各种需要采取不同恢复措施的故障情况。如果已准备了一个进行过彻底演习的恢复策略，它可保护您的数据免被丢失。在制订恢复策略时，您需要回答的一些问题是：数据库是可恢复的吗？恢复数据库可能花费多长时间？在备份操作之间将花费多长时间？可以为备份副本和归档日志分配多少存储空间？表空间级别的备份是否足够？或是否需要进行完整的数据库备份？

数据库恢复策略应确保在数据库恢复操作需要时，所有信息都可用。它应包括一个进行数据库备份的固定调度表，在分区数据库系统中则包括缩放系统时（添加或删除数据库分区服务或节点时）的备份。完整的策略还应包括恢复命令脚本、应用程序、用户定义的函数 (UDF)、操作系统库中的存储过程代码以及装入副本的过程。

在以下几节中还讨论了不同的恢复方法，您将发现最适用于您的商业环境的是哪种恢复方法。

数据库*备份*的概念与其他任何数据备份的概念一样：即，复制一份数据，然后将它存储在另一媒体上，以防原始媒体发生故障或毁坏。最简单的备份情况涉及关闭数据库（以确保不发生更多的事务），然后简单地对其进行备份。如果数据库已毁坏，就需要重新构建数据库。

数据库的重新构建称为*恢复*。*版本恢复*指的是使用备份操作期间创建的映象来复原数据库的先前版本。*前滚恢复*是指复原了数据库或表空间备份映象后，重新应用记录在数据库日志文件中的事务。

*崩溃恢复*是指在完成并落实所有更改（这些更改是一个或多个工作单元的一部分）或事务之前如果发生故障，会自动恢复数据库。这是通过回滚未完成的事务，并完成在发生崩溃时仍在内存中的已落实事务来实现的。

关于这些不同的恢复方法的详细信息，参见第18页的『版本恢复』、第19页的『前滚恢复』或第8页的『崩溃恢复』。

恢复日志文件和恢复历史记录文件是在创建数据库时自动创建的（第4页的图1）。不能直接修改恢复日志文件或恢复历史记录文件；但是万一您需要使用数据库备份映象来恢复丢失或损坏的数据，这两个文件会非常重要。

数据库对象 / 概念	等效物理对象

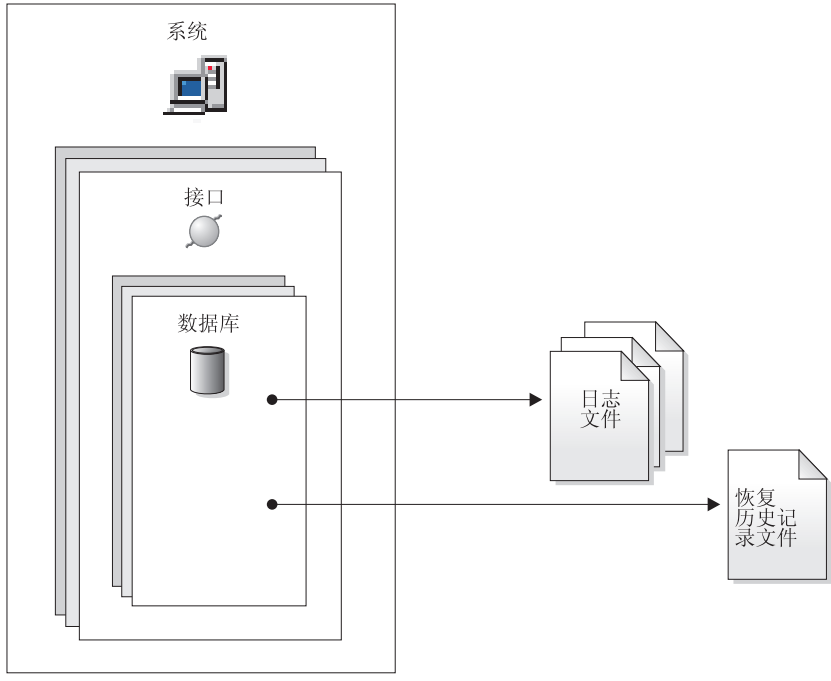


图 1. 恢复日志文件和恢复历史记录文件

每个数据库都包括*恢复日志*，它们用来从应用程序或系统错误中恢复。与数据库备份一起，它们用来将数据库的一致性恢复到出错时的时间点。

*恢复历史记录文件*包含特定备份信息的摘要，当必须将整个数据库或数据库的一部分恢复至给定时间点时，可以使用这些信息。恢复历史记录文件用来跟踪其他操作中与恢复相关的事件，如备份与复原操作。

那些很容易重新创建的数据可存储在不可恢复的数据库中。这些数据包括：用于只读应用程序的外部源中的数据以及不常进行更新的表；由于对它们进行的日志记录量较小，如果在复原操作之后还要进行复杂的日志文件文件管理工作和前滚操作，就不太合理了。在*不可恢复数据库*中禁用了 *logretain* 和 *userexit* 数据库配置参数。这表明只有保存的日志才是崩溃恢复所必需的。这些日志称为*活动日志*，它们包含当前事务数据。使用*脱机*备份的版本恢复是解决不可恢复数据库的恢复问题的主要手段。（脱机备份表示当备份操作正在进行时，其他应用程序无法使用该数据库。）这样的数据库只能进行脱机复原。它被复原为创建备份映象时的状态。

那些不容易重新创建的数据应存储在可恢复的数据库中。这些数据包括在装入后它的源已被破坏的数据、手工输入到表中的数据以及在装入数据库后由应用程序或用户修改的数据。*可恢复数据库*将 *logretain* 数据库配置参数设置为 "RECOVERY" 或启用了 *userexit* 数据库配置参数，或两种方法同时使用。活动日志仍可用于崩溃恢复，但您还有*已归档日志*，它包含已落实的事务数据。这样的数据库只能进行脱机复原。它被复原为创建备份映象时的状态。但对于前滚恢复，可通过使用活动日志和已归档日志来将数据库*前滚*（即，越过创建备份映象的时间）至特定的时间点，或者滚动至活动日志末尾。

可恢复数据库备份操作可以脱机执行，也可以*联机*执行（联机表示在备份操作期间其他应用程序可与该数据库连接）。数据库复原和前滚操作必须始终以脱机方式执行。联机备份操作期间，前滚恢复确保捕捉了*所有*表更改，且在复原该备份时重新应用这些更改。

若有一个可恢复数据库，则可备份、复原并将个别表空间前滚，而不必对整个数据库操作。当联机备份表空间时它仍然可用，同时发生的更新记录在日志中。当对表空间执行联机复原或前滚操作时，在该操作完成之前，该表空间本身不可用，但不阻止用户访问其他表空间中的表。

确定备份的频率

因为备份数据库需要时间和系统资源，所以恢复计划应该允许定期的备份操作。计划可能包括完整数据库备份和增量备份操作的组合（参见第21页的『增量备份与恢复』）。

即使已将日志归档，也应定期执行完整的数据库备份（以便允许前滚恢复）。从表空间备份映象的集合来重新构建数据库要比从完整的数据库备份映象来恢复数据库要困难得多。表空间备份映象在从独立的磁盘故障或应用程序错误进行恢复时很有用。

还应考虑不要覆盖备份映象和日志，应保存至少两个完整的数据库备份映象及其相关的日志，作为额外的预防措施。

若在恢复和前滚非常活跃的数据库时，应用归档日志所需的时间是主要关注的问题，应考虑进行更频繁的数据库备份所需的成本。进行更频繁的数据库备份可减少前滚时需要应用的归档日志数目。

数据库在*联机*或*脱机*时都可启动备份操作。如果它是联机的，在运行备份操作的同时，其他应用程序或进程可以与该数据库连接并读取和修改数据。如果备份操作是脱机运行的，其他应用程序则*不能*与数据库连接。

确定备份的频率

要缩短数据库处于不可用状态的时间，应考虑使用联机备份操作。仅当启用前滚恢复时，才支持联机备份操作。若启用前滚恢复且有一组完整的恢复日志，可以在需要时重建数据库。如果您的日志跨越了运行备份操作的时间段，则只能使用联机备份映象进行恢复。

脱机备份操作比联机备份操作快。

若一个数据库包含大量的长整数字段和大对象 (LOB) 数据，则备份该数据库可能会占用非常多的时间。备份实用程序使您可备份所选的表空间。若使用 DMS 表空间，则可以将不同类型的数据存储在它们各自的表空间中，以减少备份操作所需的时间。可以将表数据保存在一个表空间中，将长整数字段和 LOB 数据保存在另一个表空间中，再将索引保存在一个表空间中。通过将长整数字段和 LOB 数据存储在单独的表空间中，不选择备份包含该长整数字段和 LOB 数据的表空间，可以减少完成一项备份操作所需的时间。若长整数字段和 LOB 数据对于您的业务很重要，应参照完成这些表空间的复原操作所需的时间考虑备份这些表空间。若可从单独的源复制 LOB 数据，在创建或改变一个表以包括 LOB 列时选择 NOT LOGGED 选项。

注: 以下是如果要将长字段数据、LOB 数据和索引保留在单独的表空间中，但又不想将它们备份在一起时的特殊注意事项：若备份未包含所有表数据的一个表空间，则不能对该表空间执行时间点前滚恢复。包含一个表的任何类型数据的所有表空间都必须同时前滚至同一个时间点。

若重组一个表，应该在该操作完成后备份受影响的表空间。若不得不复原这些表空间，将不必通过数据重组来前滚。

恢复一个数据库所需的时间由两部分组成：复原备份所需的时间；以及，若允许数据库进行前滚恢复，在前滚操作期间应用日志所需的时间。当确定恢复计划时，应考虑这些恢复成本和它们对商业运作的影响。测试整体恢复计划有助于确定恢复数据库所需的时间是否适合给定的业务需求。在每次测试之后，可能要增加建立备份的频率。若前滚恢复是策略的一部分，这将会减少备份之间归档的日志数，因此也减少了复原操作之后前滚数据库所需的时间。

存储器注意事项

当确定要使用哪种恢复方法时，考虑必需的存储空间。

版本恢复方法需要空间来容纳数据库的副本和复原的数据库。前滚恢复方法需要空间来容纳数据库或表空间的副本、复原的数据库和归档的数据库日志。

若一个表包含长整数字段或大对象 (LOB) 列，应考虑将此数据置于单独的表空间中。这将会影响存储空间注意事项，并影响恢复计划。如果使用单独的表空间来

存储长整数字段和 LOB 数据，并知道备份长整数字段和 LOB 数据所需的时间，那么可以决定使用这样一个恢复计划，它以较低的频率保存此表空间的备份。也可选择在创建或改变一个表以包括 LOB 列时，不记录对那些列的更改。这将减少必需的日志空间和对应的日志归档空间的大小。

包含 LOB 的 SMS 表空间的备份大小可能比原始表空间的大小要大。根据表空间中的 LOB 数据大小，该备份可以增大 40%。例如，若建立了一个 1 GB 的 SMS 表空间（含有 LOB），当复原它时将需要大于 1GB 的磁盘空间。只在支持稀疏分配的文件系统（例如，基于 UNIX 的操作系统）上才会出现这种情况。

要防止媒体故障损坏数据库且使您无法重建它，应该在不同的设备上保存数据库备份、数据库日志和数据库本身。鉴于此原因，极力建议您使用 *newlogpath* 配置参数，这样一旦创建了该数据库，就将数据库日志置于单独的设备中。（与日志记录相关的其他配置参数在第31页的『数据库记录的配置参数』中讨论。）

数据库日志可能会占用大量的存储器。若计划使用前滚恢复方法，则必须确定管理归档日志的方法。以下是您的选择:

- 在数据库日志路径目录中提供足够的空间来保存这些日志。
- 当某些日志不再出现在活动的日志集中后，以人工方式将这些日志复制到存储设备或非数据库日志路径目录的一个目录中。
- 使用用户出口程序来将这些日志复制到您环境中的另一个存储设备中。例如，在 OS/2 上 DB2 支持用户出口程序将数据库备份映象和数据库日志存储在标准和非标准的设备上。（关于更多详细信息，参见第411页的『附录H. 数据库恢复的用户出口』。）

将相关的数据保存在一起

作为数据库设计的一部分，您将了解表之间存在的关系。这些关系可以在应用程序级表示（当事务更新多个表时），也可以在数据库级表示（表之间存在参考完整性，或者一个表上的触发器影响另一个表）。当制定恢复计划时，应该考虑这些关系。您将希望把相关的数据集备份在一起。可以在表空间级或数据库级建立这样的集合。通过将相关的数据集保存在一起，可以恢复至所有数据都一致的时间点。若您希望能够对表空间执行时间点前滚恢复，这一点尤其重要。

使用不同的操作系统

在具有多个操作系统的一个环境中工作时，必须注意: 在大多数情况下，备份和恢复计划是不能集成的。也就是说，通常不能在一个操作系统上备份数据库，然后在另一操作系统上复原该数据库。在这种情况中，应使每个操作系统的恢复计划分开并互不相关。

使用不同的操作系统

但在 Sun Solaris 和 HP 之间，可支持交叉平台的备份与复原操作。在系统间传送备份映象时，必须采用二进制方式。在目标系统上，创建数据库时使用的代码页和地区必须与创建原始数据库的系统相同。

如果必须将表从一个操作系统移动到另一操作系统，但在您的环境中又不支持交叉平台的备份与复原操作，可以使用 **db2move** 命令，或导出实用程序（后跟导入或装入实用程序）。关于这些实用程序的更多信息，请参阅 *Data Movement Utilities Guide and Reference*。

崩溃恢复

对数据库执行的事务（也称工作单元）可能被意外中断。若在作为工作单元一部分的所有更改完成和落实之前发生故障，则该数据库就会处于不一致和不可用的状态。*崩溃恢复*是将数据库移动回一致并可用状态的进程。为此，回滚未完成的事务，并完成当发生崩溃时仍在内存中的已落实事务（图2）。当数据库处于一致和可用状态时，它处于一种称为“一致点”的状态。

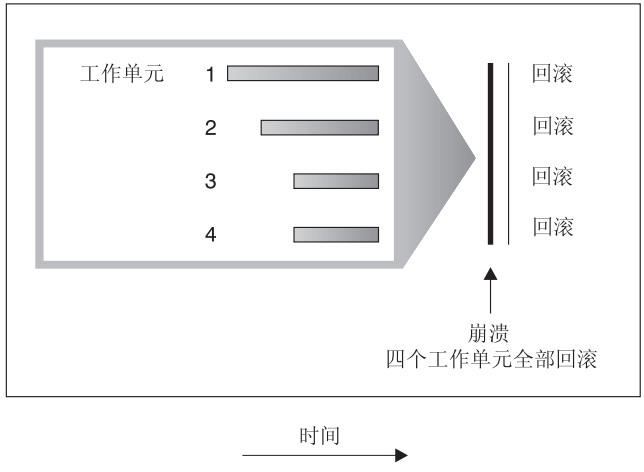


图 2. 回滚工作单元（崩溃恢复）

*事务处理失败*是由于出现了严重错误或导致数据库或数据库管理器异常结束的情况。部分完成的工作单元或发生故障时未刷新到磁盘中的 UOW 使数据库处于不一致和不可用状态。在事务处理故障之后必须恢复数据库。导致事务处理故障的情况有:

- 机器上的掉电故障，它会导致使用该机器的数据库管理器和数据库分区崩溃
- 导致 DB2 崩溃的严重操作系统错误。

如果您希望不完整工作单元的回滚是由数据库管理器自动完成的，则应将 *autorestart* 数据库配置参数设置为 ON，以启用该自动重新启动参数。（这是缺省值。关于此参数的更多信息，请参阅《*管理指南: 性能*》一书。）如果不想重新启动，则需要在发生数据库故障时发出 RESTART DATABASE 命令。db2diag.log 文件记录了数据库重新启动操作开始的时间。

如果对允许正向恢复的数据库应用崩溃恢复（即，将 *logretain* 配置参数设置为 RECOVERY，或启用 *userexit* 配置参数），且在崩溃恢复期间因个别表空间而发生错误，则必须使该表空间脱机，直到修复后才能访问它。崩溃恢复继续进行。在崩溃恢复完成时，该数据库中的其他表空间通常仍是可使用的，且可以与该数据库建立连接。

恢复已损坏的表空间

损坏的表空间带有一个或多个不能访问的容器。这通常是由媒体问题引起的: 这些问题或者是永久性的（例如，坏的磁盘），或者是临时性的（例如，脱机磁盘或未安装的文件系统）。

若损坏的表空间是系统目录表空间，则不能重新启动数据库。若不能修复容器问题但要使原始数据不受影响，可用的选项包括:

- 要复原数据库
- 要复原目录表空间。（因为数据库必须前滚，所以表空间复原只对可恢复数据库有效。）

如果损坏的表空间*不是*系统目录表空间，DB2 会尝试使尽可能多的数据库可用。

如果损坏的表空间只是临时的表空间，应在可进行与数据库的连接后，尽快创建新的临时表空间。创建后，可使用这个新的临时表空间，并且需要临时表空间的正常数据库操作也可以继续执行。若希望的话，可删除脱机的临时表空间。使用系统临时表空间的表重组有特殊的注意事项:

- 若数据库或数据库管理器配置参数 *indexrec* 设置为 RESTART，在数据库活动期间必须重建所有无效的索引；这包括构建阶段期间发生故障的重组中的索引。
- 若在损坏的临时表空间中有未完成的重组请求，可能必须将 *indexrec* 配置参数设置为 ACCESS 以避免重新启动失败。

恢复可恢复数据库中的表空间

损坏的表空间被置于脱机和不可访问状态，并且，因为必须执行崩溃恢复，所以还处于前滚暂挂状态。若没有其他问题，则重新启动操作将成功。可以再次使用损坏的表空间的条件是:

- 修复损坏的容器而不丢失原始数据，然后完成表空间前滚操作。（前滚操作将首先尝试使该表空间由脱机状态转移至正常状态。）

- 修复损坏的容器（丢失或不丢失原始数据）之后，执行表空间复原操作，然后执行前滚操作。

恢复不可恢复的数据库中的表空间

因为崩溃恢复是必需的，且日志不会无限期保存，因此仅当用户希望删除损坏的表空间时重新启动操作才会成功。（恢复成功完成意味着将损坏的表空间恢复到一致状态所必需的运行记录不再存在；因此，对这类表空间可以采取的唯一有效操作是删除它们。）

可通过调用未限定的重新启动数据库操作来做到这一点。若没有损坏的表空间，它将成功。若它失败 (SQL0290N)，则可查看 db2diag.log 文件，以获得当前已损坏的表空间的完整列表。

- 如果您愿意在重新启动数据库操作完成时删除所有这些表空间，则可启动另一重新启动数据库操作，使用 DROP PENDING TABLESPACES 选项列示所有已损坏的表空间。若损坏的表空间包括在 DROP PENDING TABLESPACES 列表中，则表空间进入删除暂挂状态，在恢复之后，唯一的选项是删除该表空间。重新启动操作继续，而不恢复此表空间。若损坏的表空间未包括在 DROP PENDING TABLESPACES 列表中，则重新启动数据库操作失败，并返回 SQL0290N。
- 若您不想删除这些表空间并因此丢失其中的数据，则有以下选项:
  - 等待并修复损坏的容器（没有丢失原始数据），然后再次尝试重新启动操作
  - 执行数据库复原操作。

注: 将表空间名放入 DROP PENDING TABLESPACES 列表中并不表示该表空间将处于删除暂挂状态。仅当在重新启动操作期间发现表空间损坏时才会是这样。重新启动操作成功后，应发出 DROP TABLESPACE 语句来删除每个处于删除暂挂状态的表空间（调用 LIST TABLESPACES 命令来了解哪些表空间处于此状态）。这样，可收回该空间，或者可重新创建那些表空间。

降低媒体故障的影响

要降低媒体故障的可能性，并简化从此类故障的恢复:

- 镜像或复制保存重要数据库的数据和日志的磁盘。
- 使用 “冗余独立磁盘阵列” (RAID) 配置，例如 RAID 级别 5。关于 RAID 更多信息，参见第11页的『防止磁盘故障』。
- 在分区数据库环境中，要设置一个严格的过程，来处理该目录节点上的数据和日志。因为此节点是维护数据库的关键:
  - 确保它驻留在可靠的磁盘上
  - 复制它

- 频繁地进行备份
- 不要在此节点上存放用户数据。

防止磁盘故障

若您担心由于磁盘故障而使数据或日志损坏的可能性，则请考虑使用某种形式磁盘故障容错。通常，这通过使用*磁盘阵列*来实现。磁盘阵列由一组磁盘驱动器组成，对于一个应用程序，磁盘阵列表现为单一的大容量磁盘驱动器。

磁盘阵列包含*磁盘条带化*（即将一个文件分布至多个磁盘）、磁盘的镜像和数据奇偶校验。

磁盘阵列有时只是指 RAID（冗余独立磁盘阵列）。磁盘阵列也可以通过处于操作系统级或应用程序级的软件来提供。硬件磁盘阵列和软件磁盘阵列的不同点在于处理输入／输出 (I/O) 请求的 CPU 处理的方式。对于硬件磁盘阵列，I/O 活动由磁盘控制器管理，对于软件磁盘阵列，由操作系统或应用程序完成。

硬件磁盘阵列: 在硬件磁盘阵列中，一个磁盘控制器可使用和管理多个磁盘，且使用它自己的 CPU 来完成。管理构成此阵列的磁盘所需的所有逻辑都包含在磁盘控制器中；因此，此实现与操作系统无关。

有几种类型的 RAID 体系结构，在功能和性能上有所不同，但只有 RAID 级别 1 和级别 5 是现在常用的。

RAID 级别 1 也称为磁盘镜像和双工技术。*磁盘镜像技术*使用单个磁盘控制器将数据（一个完整的文件）从一个磁盘复制到另一个磁盘。*磁盘双工技术*类似于磁盘镜像技术，但磁盘是与另一个磁盘控制器连接的（与两个 SCSI 适配器相似）。因此可较好地保护数据：任何一个磁盘发生故障时，仍可从另一个磁盘访问数据。使用磁盘双工技术时虽然磁盘控制器可能会发生故障，但不会影响可提供的数据保护。它的性能很好，但实现它需要的磁盘数是通常磁盘数的两倍。

RAID 级别 5 涉及到跨所有磁盘，按扇区进行的数据和奇偶性校验条带化。奇偶性校验与数据交错，而不是存储在专用的驱动器上。数据保护性能很好：如果有任何磁盘发生故障，仍然可以使用其他磁盘上的信息以及已条带化的奇偶性校验信息访问数据。读性能良好，但写性能较差。RAID 级别 5 配置需要最少三个相同的磁盘。开销所需的磁盘空间量随阵列中磁盘的数量而变化。如果 RAID 级别 5 配置了 5 个磁盘，空间开销将是百分之二十。

使用 RAID（但不是 RAID 级别 0）磁盘阵列时，有故障的磁盘不会影响您访问阵列上的数据。当在该阵列中使用可热插入的或可热交换的磁盘时，可以在该阵列正在使用时将替换磁盘与故障磁盘交换。使用 RAID 级别 5 时，如果有两个磁盘同时发生故障，所有数据都会丢失（但同时发生磁盘故障的可能性非常小）。

您可能会考虑对日志使用 RAID 级别 1 硬件磁盘阵列或软件磁盘阵列（参见『软件磁盘阵列』），原因是：该 RAID 级别可对故障点提供可恢复性并提供良好的写性能（这对日志来说很重要）。如果（由于不能将时间花在磁盘故障后的数据恢复上）可靠性很关键，而写性能不是，则可以考虑使用 RAID 级别 5 硬件磁盘阵列。另外，如果写性能很重要，而不太关心所需的附加磁盘空间量，则可以考虑对您的数据以及日志使用 RAID 级别 1 硬件磁盘阵列。

关于可用的 RAID 级别的详细信息，请访问：

```
http://www.acnc.com/04_01_00.html
```

软件磁盘阵列: 软件磁盘阵列在许多方面与硬件磁盘阵列相同（参见第11页的『硬件磁盘阵列』），但磁盘流量是由操作系统或在服务器上运行的应用程序管理的。软件阵列象其他程序一样，必须争用 CPU 和系统资源。对于 CPU 受约束的系统，它不是一个好的选项，同时要记住磁盘阵列的总体性能取决于服务器的 CPU 负荷和能力。

典型的软件磁盘阵列提供磁盘镜像（参见第11页的『硬件磁盘阵列』）。虽然冗余磁盘是必需的，但是由于不需要高成本的磁盘控制器，所以软件磁盘阵列的实现相对便宜。

注意:
如果操作系统引导驱动器在磁盘阵列中，则当该驱动器发生故障时，系统将无法启动。若在磁盘阵列运行之前该驱动器发生故障，则磁盘阵列就不允许访问该驱动器。引导驱动器应与磁盘阵列分开。

降低事务故障的影响

要降低事务故障的影响，应尽量确保：

- 不间断电源
- 有足够的磁盘空间用于存储数据库日志
- 在分区数据库环境中的数据库分区服务器之间有可靠的通信链路
- 在分区数据库环境中的系统时钟同步（参见第124页的『使分区数据库系统中的时钟同步』）。

从分区数据库环境中的事务处理故障中恢复

如果事务处理失败发生在分区数据库环境中，通常需要对发生了故障的数据库分区服务器和参予了该事务的任何其他数据库分区服务器都进行数据库恢复：

- 对发生了故障的数据库分区服务器的崩溃恢复发生在更正了先前的错误情况后。

- 对其他（仍活动的）数据库分区服务器的*数据库分区故障恢复*紧接在检测到故障后发生。

在分区数据库环境中，提交应用程序的数据库分区服务器是协调程序节点，而为该应用程序工作的第一个代理程序是代理程序节点。协调代理程序负责将工作分发至其他数据库分区服务器上，并跟踪哪些服务器参与了该事务。当应用程序对一个事务发出 COMMIT 语句时，该协调代理程序使用两阶段落实协议来落实该事务。在第一阶段期间，协调程序节点将 PREPARE 请求分发至参与该事务的所有其他的数据库分区服务器。然后，这些服务器用下列其中一项应答:

READ-ONLY	在此服务器中未发生任何数据更改
YES	在此服务器中发生了数据更改
NO	由于错误，服务器未准备落实

若其中一个服务器应答 NO，则回滚该事务。否则，协调程序节点开始第二个阶段。

在第二阶段期间，协调程序节点写入一条 COMMIT 日志记录，然后将 COMMIT 请求分发至应答了 YES 的所有服务器。在所有其他数据库分区服务器都已落实后，它们会将 COMMIT 的确认发送至协调程序节点。当协调代理程序从所有参与的服务器接收到所有 COMMIT 确认时，该事务完成。在此时间点，协调代理程序会写入一条 FORGET 日志记录。

有关两阶段落实的信息，请参阅*《管理指南: 计划》*一书。

活动数据库分区服务器上的事务处理故障恢复

若任何数据库分区服务器检测到另一个服务器停机，则与该失效的数据库分区服务器相关的所有工作都会停止:

- 若仍为活动的数据库分区服务器是一个应用程序的协调程序节点，且该应用程序是在失效的数据库分区服务器上运行（尚未准备 COMMIT），则会中断该协调代理程序，以便执行故障恢复。如果该协调代理程序处于 COMMIT 处理的第二个阶段，会将 SQL0279N 返回给应用程序，应用程序随之会丢失它的数据库连接。否则，协调代理程序将一个 ROLLBACK 请求分发至所有其他参与该事务的服务器，并将 SQL1229N 返回至该应用程序。
- 若失效的数据库分区服务器是该应用程序的协调程序节点，仍在活动服务器上为该应用程序工作的代理程序会被中断，以便执行故障恢复。除非当前事务已准备就绪且正在等待事务结果，否则在每个服务器上以本地方式回滚当前事务。在这种情况下，该事务在活动的数据库分区服务器上处于未确定状态，而协调程序节点未意识到这个情况（因为它不可用）。

  关于如何解决不确定事务的更多信息，请参阅*《管理指南: 计划》*一书。

- 若该应用程序与失效的数据库分区服务器连接（在它失效之前），但是本地数据库分区服务器和失效的数据库分区服务器都不是协调程序节点，则会中断为此应用程序工作的代理程序。协调程序节点将向其他数据库分区服务器发送 ROLLBACK 或断开消息。若协调程序节点返回 SQL0279，则事务将只在仍然活动的数据库分区服务器上是未确定的。

试图向该失效服务器发送请求的任何进程（如，代理程序或死锁检测器）都会得到通知：它不能发送该请求。

有故障的数据库分区服务器上的事务处理失败恢复

如果事务处理失败导致数据库管理器异常结束，可使用 RESTART 选项发出 **db2start** 命令，以在重新启动了处理器后立即重新启动数据库管理器。如果不能重新启动处理器，可发出 **db2start** 在不同的处理器上重新启动数据库管理器。关于更多信息，请参阅 *Command Reference*。

如果数据库管理器异常结束，则服务器上的数据库分区可能会处于不一致状态。要使它们可用，可在数据库分区服务器上触发崩溃恢复。

- 通过 RESTART DATABASE 命令显式触发
- 在 *autorestart* 数据库配置参数已设置为 ON 后，通过 CONNECT 请求隐式触发

崩溃恢复将日志记录重新应用到活动的日志文件中，以确保所有已完成的事务的结果都在数据库中。重新应用了这些更改后，除不确定事务外的所有未落实的事务都将本地回滚。分区数据库环境中有两种类型的不确定事务：

- 在不是协调程序节点的数据库分区服务器上，已就绪但未落实的事务就是未确定的。
- 在协调程序节点上，已落实但还未被记录为完成（即，还未写入 FORGET 记录）的事务是未确定的。当协调代理程序未从为该应用程序工作的所有服务器接收到全部 COMMIT 确认时，会发生此情况。

崩溃恢复试图通过下列其中一项操作解决所有未确定的事务。要执行的操作取决于数据库分区服务器是否是应用程序的协调程序节点：

- 若重新启动的服务器不是该应用程序的协调程序节点，它会将一个查询消息发送至该协调代理程序，以找到该事务的结果。
- 若重新启动的服务器是该应用程序的协调程序节点，它会将一个消息发送至协调代理程序仍在等待它们的 COMMIT 确认的所有其他代理程序（从属代理程序）。

有可能崩溃恢复不能解决所有不确定事务（例如，某些数据库分区服务器可能会不可用）。在这种情况中，会返回 SQL 警告消息 SQL1061W。由于不确定事务保留了资源（例如锁和活动日志空间），有可能会处于不能对数据库进行任何更改

的一点，因为不确定事务保留了活动日志空间。因此，应确定在崩溃恢复之后是否还有不确定事务，并尽快恢复解决这些不确定事务所需的所有数据库分区服务器。

若解决不确定事务所需的一个或多个服务器不能及时恢复，且需要访问其他服务器上的数据库分区，可以通过作出试探性决策来人工解决这些不确定事务。可以使用 LIST INDOUBT TRANSACTIONS 命令（参阅 *Command Reference*）来查询、落实和回滚服务器上的不确定事务。

注: LIST INDOUBT TRANSACTIONS 命令还可用于分布式事务环境中。要区分这两种类型的不确定事务，LIST INDOUBT TRANSACTIONS 命令所返回的输出中的 *originator* 字段显示下列其中一项:

- DB2 通用数据库扩充企业版，它指示该事务始发于分区数据库环境中。
- XA，它指示该事物始发于分布式环境中。

关于分布式环境的更多信息，请参阅*《管理指南: 计划》*一书。

标识有故障的数据库分区服务器

当一个数据库分区服务器发生故障时，应用程序通常会接收到下列其中一个 SQLCODE。检测哪个数据库管理器发生故障的方法取决于接收到的 SQLCODE:

SQL0279N
当在 COMMIT 处理期间终止的事务中涉及了数据库分区服务器时，会接收到此 SQLCODE。

SQL1224N
当发生故障的数据库分区服务器是该事务的协调程序节点时，会接收到此 SQLCODE。

SQL1229N
当发生故障的数据库分区服务器不是该事务的协调程序节点时，会接收到此 SQLCODE。

确定哪个发生了故障的数据库分区服务器是一个两阶段进程。与 SQLCODE SQL1229N 相关的 SQLCA 在 *sqlerrd* 字段的第六个数组位置包含检测到错误的服务器的节点号。（为服务器写入的节点号与 `db2nodes.cfg` 文件中的节点号对应。）在检测到错误的数据库分区服务器上，将指示发生了故障的服务器的节点号的一条消息写到 `db2diag.log` 文件。

注: 若正在一个处理器上使用多个逻辑节点，则一个逻辑节点的失效可能导致同一个处理器上的其他逻辑节点失效。

要从数据库分区服务器的故障中恢复:

1. 校正导致该失效的问题。
2. 通过从任何数据库分区服务器发出 **db2start** 命令，重新启动数据库管理器。
3. 通过在有故障的一个或多个数据库分区服务器上发出 RESTART DATABASE 命令，重新启动数据库。

恢复主机上的不确定事务

若在一个事务期间您的应用程序访问了主机或 AS/400 数据库服务器，则恢复不确定事务的方法会有所不同。

要访问主机或 AS/400 数据库服务器，须使用 DB2 Connect。若 DB2 Connect 配置了 DB2 同步点管理器，则恢复的步骤就会不同。

当 DB2 Connect 配置了 DB2 同步点管理器时的恢复

恢复主机或 AS/400 服务器上的不确定事务通常由“事务管理程序” (TM) 和 DB2 同步点管理器 (SPM) 自动执行。主机或 AS/400 服务器上的不确定事务不占用本地 DB2 位置的任何资源，但是只要该事务在该位置处是未确定的，它就会占用主机或 AS/400 服务器上的资源。若主机或 AS/400 服务器的管理员确定必须作出试探性决定，则管理员可能会与本地 DB2 数据库管理员联系（例如，通过电话）来确定是落实还是回滚主机或 AS/400 服务器上的这个事务。若发生这种情况，可以使用 LIST DRDA INDOUBT TRANSACTIONS 命令来确定 DB2 Connect 实例中的这个事务的状态。可以使用下列步骤作为涉及到 SNA 通信环境的大多数情况的一个参考。

1. 与 SPM 连接，如下所示:

```
db2 => connect to db2spm

 数据库连接信息

 数据库产品         = SPM0500
 SQL 授权标识       = CRUS
 本地数据库别名     = DB2SPM
```

2. 发出 LIST DRDA INDOUBT TRANSACTIONS 命令，以显示 SPM 识别的不确定事务。以下示例显示一个 SPM 识别的不确定事务。db_name 是主机或 AS/400 服务器的本地别名。partner_lu 是主机或 AS/400 服务器的全限定 luname。它为主机或 AS/400 服务器提供了最佳标识，它应由主机或 AS/400 服务器的调用程序提供。luwid 提供事务的唯一标识符，且在所有主机和 AS/400 服务器上可用。若显示未确定的事务，而且若 `uow_status` 字段的值是 C（落实）或 R（回滚），可以使用该值来确定事务的输出。若您发出 LIST DRDA INDOUBT TRANSACTIONS 命令且带有 WITH PROMPTING 参数，则您可以用交互式方式落实、回滚或忘记该事务。关于更多信息，请参阅 *Command Reference*。

```
db2 => list drda indoubt transactions
 DRDA Indoubt Transactions:
 1.db_name: DBAS3    db_alias: DBAS3    role: AR
   uow_status: C  partner_status: I  partner_lu: USIBMSY.SY12DQA
 corr_tok: USIBMST.STB3327L
 luwid: USIBMST.STB3327.305DFDA5DC00.0001
 xid: 53514C2000000017 00000000544D4442 0000000000305DFD A63055E962000000
      00035F
```

3. 若未显示 partner_lu 和 luwid 的不确定事务，或者若 LIST DRDA INDOUBT TRANSACTIONS 命令返回下列内容:

```
db2 => list drda indoubt transactions
SQL1251W  No data returned for heuristic query.
```

则该事务已回滚。

另一个情况不大可能但是也许已经发生。若显示带有 partner_lu 的适当 luwid 的不确定事务，但 uow_status 是 ″I″，则 SPM 不能确定是落实还是回滚该事务。在这种情况下，您应该使用 WITH PROMPTING 参数在 DB2 Connect 工作站上来落实或回滚该事务。然后允许 DB2 Connect 根据试探性决定与主机或 AS/400 服务器再同步。

当 DB2 Connect 不使用 DB2 同步点管理器时的恢复

若在从 “DB2 Connect 个人版” 或 “DB2 Connect 企业版” 发出的一个多站点更新中使用 TCP/IP 连通性来更新 DB2 OS/390 版，但未使用 DB2 同步点管理器，则参考本节中的信息。此情况中的不确定事务的恢复不同于使用 DB2 同步点管理器的不确定事务的恢复。当在此环境中出现不确定事务时，会在客户机、数据库服务器和 / 或 “事务管理程序” (TM) 数据库上生成一个警报项，这取决于谁检测到该问题。该警报项被置于 db2alert.log 文件中。关于警告的更多信息，请参阅 *Troubleshooting Guide*。

一旦 TM 和参与数据库及其连接再次全部可用，就会自动执行任何不确定事务的再同步。您应该允许在数据库服务器中自动执行再同步，而不是试探性地强制决定。但是，若您必须这样做，可参考下列步骤。

注: 因为未涉及到 DB2 同步点管理器，所以您不能使用 LIST DRDA INDOUBT TRANSACTIONS 命令。

1. 在 OS/390 主机上，发出命令 DISPLAY THREAD TYPE(INDOUBT)。

   从此列表中，标识您要试探性完成的事务。关于 DISPLAY 命令的详细信息，请参阅 *DB2 for OS/390 Command Reference*。显示的 LUWID 可以与 “事务管理程序数据库” 中的同一个 luwid 匹配。

2. 根据您要执行的操作，发出 RECOVER THREAD(<LUWID>) ACTION(ABORT|COMMIT) 命令。

关于 RECOVER 命令的详细信息，请参阅 *DB2 for OS/390 Command Reference*。

灾难恢复

术语*灾难恢复*用于描述在火灾、地震、恶意破坏或其他大灾害事件中复原数据库所需要执行的活动。灾难恢复的计划可以包括下列其中一项或多项:

- 在紧急情况中要使用的场地
- 用于恢复数据库的另一台机器
- 数据库备份和归档日志的非现场存储器。

若灾难恢复计划是在另一台机器上恢复整个数据库，则至少需要一个完整的数据库备份和该数据库的所有归档日志。可选择通过将归档日志应用于数据库，以便备用数据库保持最新。或者，可选择将数据库备份和日志归档保存在备用场地，并只在发生灾难之后执行复原和前滚操作。（在这种情况中，最新的数据库备份无疑是所希望的。）但是，当发生灾难时，将所有事务恢复至灾难发生时的状态一般是不可能的。

灾难恢复的表空间备份的有用性取决于发生故障的范围。通常，灾难恢复需要复原整个数据库，因此，应在备用场所保存一份完整的数据库备份。即使有每个表空间的单独备份映象，也不能使用它们来恢复该数据库。若该灾难是磁盘损坏，则可以使用该磁盘上每个表空间的表空间备份来恢复。若由于磁盘故障（或任何其他原因）而无法访问一个容器，可以将该容器复原至不同的位置。有关其他信息，参见第93页的『在复原操作期间重新定义表空间容器（重定向复原）』。

表空间备份和完整的数据库备份都可以在任何灾难恢复计划中起到一定的作用。可用于备份、复原和前滚数据的 DB2 设施为灾难恢复计划提供了一个基础。应确保已测试了恢复过程，它可以保护您的业务。

版本恢复

*版本恢复*指的是使用备份操作期间创建的映象来复原数据库的先前版本。对不可恢复数据库（即，该数据库没有归档日志）使用此方法。还可对 RESTORE DATABASE 命令使用 WITHOUT ROLLING FORWARD 选项，对可恢复数据库使用此方法。数据库复原操作将使用先前创建的备份映象来重建整个数据库。数据库备份使您可以将数据库复原到与进行备份时完全相同的状态。但是，从备份时间到故障时间之间的每个工作单元都将丢失（参见第19页的图3）。

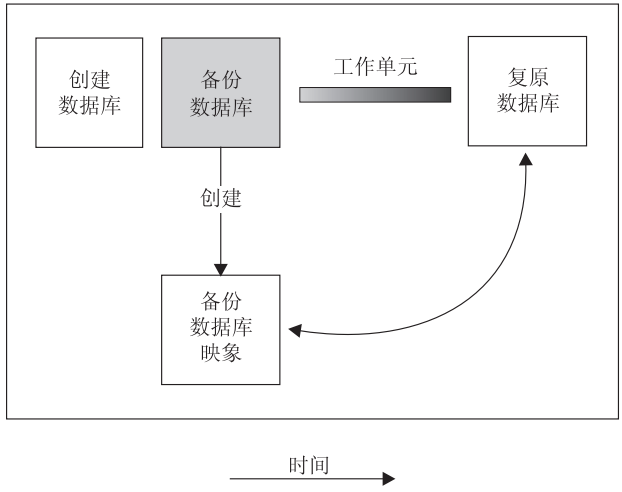


图 3. 版本恢复. 从最新的备份映象复原数据库，但在备份时间和故障时间之间处理的所有工作单元都将丢失。

使用版本恢复方法，必须定期调度和执行完整数据库备份。

在一个分区数据库环境中，数据库位于许多数据库分区服务器（或节点）上。必须复原所有分区，而且用于复原数据库操作的所有备份映象必须是同时建立的。（将备份每个数据库分区并单独复原。）同时建立的每个数据库分区的备份称为*版本备份*。

前滚恢复

要使用*前滚恢复*方法，必须已建立了数据库的备份，并且已将日志归档（通过启用 *logretain* 和 / 或 *userexit* 数据库配置参数）。有关您必须对所用的记录过程作出的决策的信息，参见第25页的『了解恢复日志』。复原数据库并指定 WITHOUT ROLLING FORWARD 选项等效于使用版本恢复方法。此数据库被复原到创建脱机备份映象时的状态。若复原数据库，但*没有*对复原数据库操作指定 WITHOUT ROLLING FORWARD 选项，则该数据库在复原操作结束时将处于前滚暂挂状态。这允许进行前滚恢复。

要考虑的两种恢复类型是:

- *数据库前滚恢复*。在此类型的前滚恢复中，记录在数据库日志中的事务应用到以下数据库复原操作中（参见第20页的图4）。该数据库日志记录了对该数据库所作的所有更改。这种方法将数据库恢复到在某特定时间点的状态，或恢复到故障前的状态（即，恢复到活动日志的末尾）。

在一个分区数据库环境中，数据库位于许多数据库分区上。若执行时间点前滚恢复，则必须前滚所有数据库分区，以确保所有分区都在同一级别。若需要复原单个数据库分区，则可以执行前滚恢复至日志末尾，以将它复原至与数据库中的其他分区相同的级别。如果前滚一个数据库分区，则只可使用对日志末尾的恢复。时间点恢复适用于*所有*数据库分区。

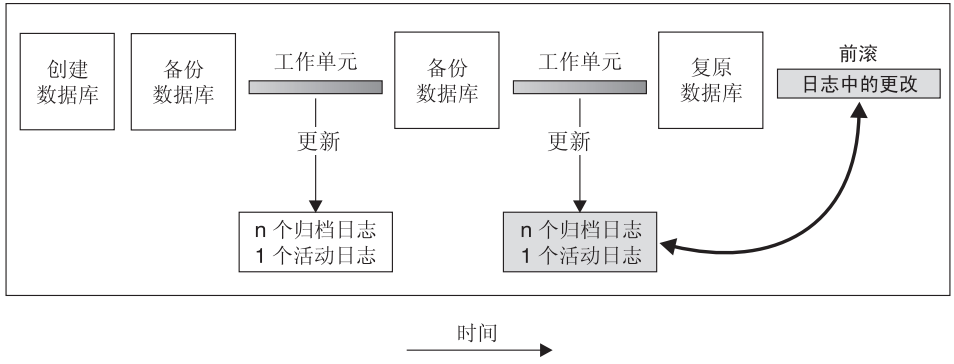


图 4. 数据库前滚恢复. 在运行时间较长的事务中，可以有多个活动的日志。

- *表空间前滚恢复*。如果启用了某个数据库对它进行前滚恢复，也可对它进行备份、复原并前滚表空间（参见第21页的图5）。要执行表空间复原和前滚操作，需要整个数据库（即所有表空间）或一个或多个个别表空间的备份映象。还需要影响要恢复的表空间的运行记录。可在日志中前滚至以下两点之一:
  - 日志末尾；或
  - 一个特定时间点（称为*时间点*恢复）。

可在下列两种情况下使用表空间前滚恢复:

- 在一个表空间复原操作后，该表空间始终处于前滚暂挂状态，且必须将它前滚。调用 ROLLFORWARD DATABASE 命令（参见 第126页的『ROLLFORWARD DATABASE 命令』）将日志应用于表空间，使其恢复到某个时间点或日志末尾。
- 若一个或多个表空间在崩溃恢复后处于*前滚暂挂*状态，首先应校正该表空间的问题。某些情况下，校正表空间的问题不涉及复原数据库操作。例如，掉电可能导致表空间处于前滚暂挂状态。在这种情况下，复原数据库操作并不是必需的。一旦校正了该表空间的问题，可使用 ROLLFORWARD DATABASE 命令将日志应用于表空间，使其恢复到日志末尾。若在崩溃恢复之前校正此问题，崩溃恢复将足以使数据库恢复到一致可用的状态。

  注: 若出错的表空间包含系统目录表，将不能启动数据库。必须复原 SYSCATSPACE 表空间，然后执行前滚恢复直至日志末尾。

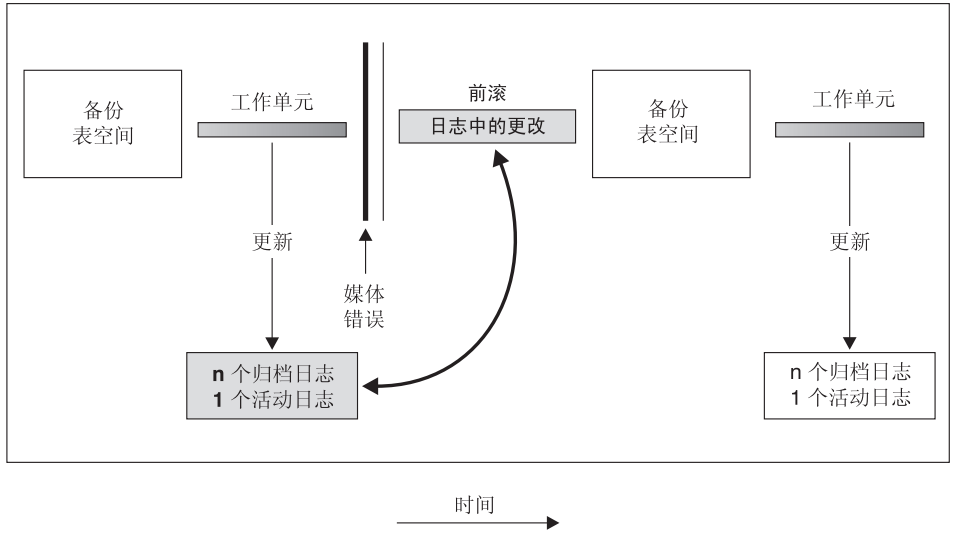


图 5. 表空间前滚恢复. 在运行时间较长的事务中，可以有多个活动的日志。

在分区数据库环境中，若要将表空间前滚至一个*时间点*，不必提供该表空间所驻留的节点（数据库分区）的列表。DB2 将前滚请求提交给所有分区。这意味着必须在表空间驻留的所有数据库分区上复原表空间。

在分区数据库环境中，若要将表空间前滚至*日志末尾*，而又*不*希望在所有的分区上前滚表空间，则必须提供数据库分区的列表。若要将（所有分区上）所有处于前滚暂挂状态的表空间前滚到日志末尾，不必提供数据库分区的列表。缺省情况下，会将数据库前滚请求发送至所有分区。

增量备份与恢复

由于数据库，尤其是仓库的大小持续扩充到太字节和拍字节的范围，备份和恢复这些数据库所需的时间及硬件资源也有了大幅的增长。因为对大数据库进行多个备份所需的存储量是巨大的，所以完整的数据库和表空间备份并不总是处理大数据库的最佳途径。请考虑以下问题:

- 仓库中只有一小部分数据更改时，应不需要备份整个数据库。
- 将表空间附加到现有数据库然后只备份表空间是危险的作法，原因是无法保证在表空间备份之间，已备份的表空间外没有任何更改。

DB2 现在可支持增量备份与恢复（但对长字段或大对象数据无效）。*增量备份*是一个备份映象，它只包含自上次进行备份以来有过更新的页。除了更新过的数据和

索引页，每个增量备份映象还包含通常存储在完整备份映象中的初始数据库元数据（例如数据库配置、表空间定义和数据库历史记录等等）。

支持两种类型的增量备份:

- *增量* - 增量备份映象是自从上次最新的、成功的完整备份操作以来，更改过的所有数据库数据的副本。也称为累积备份映象，因为进行的一系列增量备份中的每一个都会有上次增量备份映象的内容。增量备份映象的前身通常是同一对象最新的、成功的完整备份。
- *Delta* - delta 备份映象或增量 delta 备份映象是自从上次相关表空间的成功备份（包括完整、增量或 delta 备份）以来，已更改过的所有数据库数据的副本。也称为差异备份映象或非累积备份映象。delta 备份映象的前身是最新的成功备份，包括 delta 备份映象中每个表空间的备份。

增量备份映象和 delta 备份映象的关键差别在于: 对连续不断更改的对象进行持续备份时，它们的行为不同。每个持续增量映象都包含了前一个增量映象的完整内容以及自上次产生备份以来新的或更改过的任何数据。Delta 备份映象只包含自上次产生映象以来更改过的页。

允许以联机和脱机操作方式组合数据库和表空间增量备份。计划备份策略时应格外小心，因为将数据库增量备份和表空间增量备份结合在一起即暗示了数据库备份（或多个表空间的表空间备份）的前身不再需要是一个单独的映象，它可以是在不同时间进行的先前数据库和表空间备份的唯一集合。

要将数据库或表空间重建到一致状态，恢复进程必须以要复原的完整对象（数据库或表空间）的一致映象开始，然后必须以以下描述的次序（参见第23页的『从增量备份映象复原』）应用每个相应的增量备份映象。

要启用对数据库更新的跟踪，DB2 支持一个新数据库配置参数 *trackmod*，它可以有以下两个可接受的值之一:

- NO - 增量备份不允许使用此配置。不会以任何方式跟踪或记录数据库页跟踪。
- YES - 增量备份允许使用此配置。启用了更新跟踪后，更改会对与实例中任何数据库的第一个成功连接生效。在进行增量备份前必需一个完整的数据库备份。

对现有数据库的缺省 *trackmod* 设置是 NO；对于新数据库，它是 YES。

对于 SMS 表空间，此跟踪的粒度是表空间级。对于 DMS 表空间，粒度为数据和索引页的扩展级，对于其他页类型则是表空间级。

对数据库的更新跟踪会对更新或插入数据的事务的运行时性能产生较小影响。

从增量备份映象复原

从增量备份映象执行复原操作通常由以下步骤构成:

1. 标识增量目标映象。

   DBA 必须首先确定要复原的最终映象，并从 DB2 复原实用程序请求一个增量复原操作。此映象也称为增量复原的目标映象，因为它将成为要复原的最后映象。对此映象的增量复原命令可能会启动新数据库的创建过程，创建时使用来自此目标映象的配置和表空间定义。增量目标映象是使用 RESTORE DATABASE 命令中的 TAKEN AT 参数指定的。
2. 复原最新的完整数据库或表空间映象以建立一个基线，以根据它来应用每个后继的增量备份映象。
3. 按产生时的次序，在“步骤 2”中复原的基线映象的顶部，恢复需要的各个完整增量备份映象或表空间增量备份映象。
4. 重复“步骤 3”直到“步骤 1”中的目标映象读了两次。在整个增量复原操作期间会访问两次目标映象。在第一次访问期间，只从映象读取初始数据，而不读取任何用户数据。只在第二次访问期间才读取并处理完整的映象。

   必须访问两次增量复原操作的目标映象，以确保数据库最初是使用正确的历史记录、数据库配置以及将在复原操作期间创建的数据库表空间定义来配置的。如果自从进行了最初的完整数据库备份映象以来已删除了表空间，将从备份映象中读取该映象的表空间数据，但在增量复原处理期间会忽略该数据。

要复原一组增量备份映象，应对 RESTORE DATABASE 命令指定 TAKEN AT *timestamp* 选项。指定您要复原的最后一个映象的时间戳记。例如:

```
db2 restore db sample incremental automatic taken at 20001228152133
```

这样就可以让 DB2 复原实用程序自动执行上述每个步骤。在处理的最初阶段，将读取时间戳记为 20001228152133 的备份映象，复原实用程序将验证数据库、其历史记录和表空间定义是否存在并有效。

在处理的第二阶段，将查询数据库历史记录来构建执行请求的复原操作所需的备份映象链。如果由于某些原因上述操作不可能实现，DB2 无法构建所需映象的完整链，复原操作会终止并返回错误消息。此时不可能进行自动复原，您必须使用手工复原过程继续。

注: 强烈建议不要使用 PRUNE HISTORY 命令的 FORCE 选项。此命令的缺省操作可防止您删除历史记录条目（从最新的完整数据库备份映象恢复时可能会需要它们），但如果使用 FORCE 选项，就有可能删除自动复原操作所需的条目。

如果数据库历史记录不可用，您可以手工执行增量复原操作，请遵循本节开始处描述的步骤。例如:

1. db2 restore database sample incremental taken at <ts>

其中:
<ts> 指向要复原的最后一个增量备份映象

2. db2 restore database sample incremental taken at <ts1>

其中:
<ts1> 指向初始完整数据库（或表空间）映象

3. db2 restore database sample incremental taken at <tsX>

其中:
<tsX> 指向某个序列中的每个增量备份映象

4. 重复“步骤 3”，复原每个增量备份映象并包括 <ts>

如果尝试了数据库复原操作并且已产生了表空间增量备份映象，则必须以表空间映象的备份时间戳记的流年次序进行复原。

自动增量复原的限制

1. 如果自从想进行复原的源备份以来已重命名了表空间，并使用该新名称发出了一个表空间级的复原命令，将不能正确产生使用该数据库历史记录的备份映象链，并出现错误。

   示例:

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 rename tablespace from userspace1 to t1
   db2 restore db sample tablespace ('t1') incremental automatic taken at <ts2>
   ```

   建议的解决方法是: 使用手工增量复原。

2. 如果删除数据库，历史记录也会删除。如果复原已删除的数据库，数据库历史记录将复原到它在复原备份时的状态，而自那以后的所有历史记录都将丢失。如果稍后尝试执行需要使用任何这些已丢失的历史记录条目的自动增量复原，RESTORE 实用程序将尝试恢复一个不正确的备份链，并返回“序列混乱”错误。

   示例:

   ```
   db2 backup db sample -> <ts1>
   db2 backup db sample incremental -> <ts2>
   db2 backup db sample incremental delta -> <ts3>
   db2 backup db sample incremental delta -> <ts4>
   db2 drop db sample
   db2 restore db sample incremental automatic taken at <ts2>
   db2 restore db sample incremental automatic taken at <ts4>
   ```

   建议的解决方法:

- 使用手工增量复原。
- 在发出自动增量复原命令前，首先从映象 <ts4> 复原历史记录文件。

了解恢复日志

所有数据库都有相关的日志。这些日志保存了有关数据库更改的记录。若需要将数据库复原至上一完整、脱机备份之前的一个点，则需要日志才能将数据前滚至故障点。

有三类 DB2 记录: *循环*、*捕获*和*归档*，每种类型都提供了不同级别的恢复能力。

- 当创建新数据库时，*循环*记录是缺省行为。(*logretain* 数据库配置参数设置为 NO。) 对于这种类型的记录，只有完整、脱机数据库备份才有效。正如它的名称所表示的那样，循环记录使用一个联机日志“环”，提供对事务故障和系统崩溃的恢复。仅使用和保留日志到确保当前事务的完整性这样一个程度。循环记录不允许将数据库在上次完整备份操作后执行的事务中前滚。上次备份操作后发生的所有更改都将丢失。进行完整备份时，数据库必须脱机（用户不可访问）。因为这种类型的复原操作将数据恢复至进行完整备份的特定时间点，所以它称为*版本恢复*。

  图6显示当循环记录活动时，活动日志使用一个日志文件环。

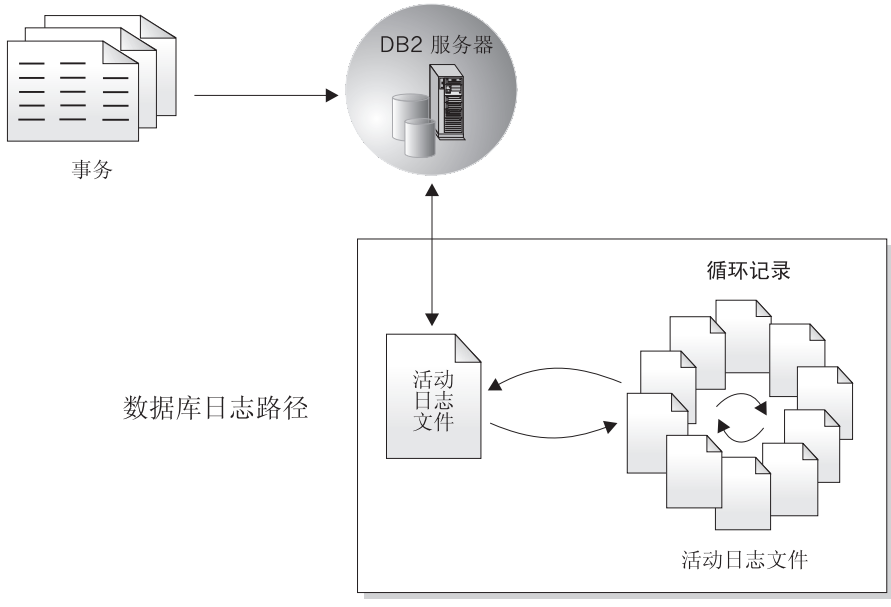


图 6. 循环记录

崩溃恢复期间，使用*活动*日志来防止故障（系统电源或应用程序错误）使数据库处于不一致的状态。RESTART DATABASE 命令在需要时使用活动日志来将数据库转变为一致和可用状态。在崩溃恢复期间将回滚记录在这些日志中的未落实的更改。那些已落实，但尚未从内存（缓冲池）写到磁盘（数据库容器）的更改将重做。这些操作确保了数据库的完整性。活动日志位于数据库日志路径目录中。

- 通过将 *logretain* 数据库配置参数设置为 CAPTURE，可配置*捕获*记录。捕获记录用于复制处理。将保留日志文件直到复制处理完成，完成后会删除日志文件。DB2 实用程序处理这种记录方式的方法与处理循环记录方式时一样；即，既不允许联机备份操作，也不允许表空间备份与复原操作，或前滚操作；且指定了 RECOVERY NO 选项的装入操作不会将表空间置于备份暂挂状态。
- *归档*记录是针对前滚恢复使用的。可通过将 *logretain* 数据库配置参数设置为 RECOVERY 来配置归档记录。归档日志可以是:

  联机归档日志 当活动日志中的更改不再需要用于正常处理时，该日志就会关闭，然后变成归档日志。当归档日志存储在数据库日志路径目录中时，就称它为*联机的*（参见第27页的图7）。

  脱机归档日志 当在数据库日志路径目录中再找不到归档日志时，就称该归档日志为*脱机的*（参见第28页的图8）。也可以使用用户出口程序将归档日志存储在数据库日志路径目录之外的位置中。（关于其他信息，参见第411页的『附录H. 数据库恢复的用户出口』。）

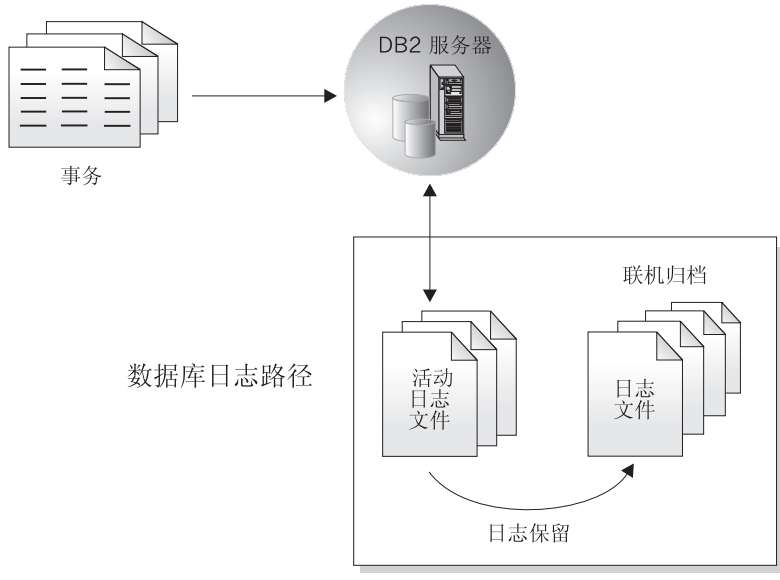


图 7. 联机归档记录

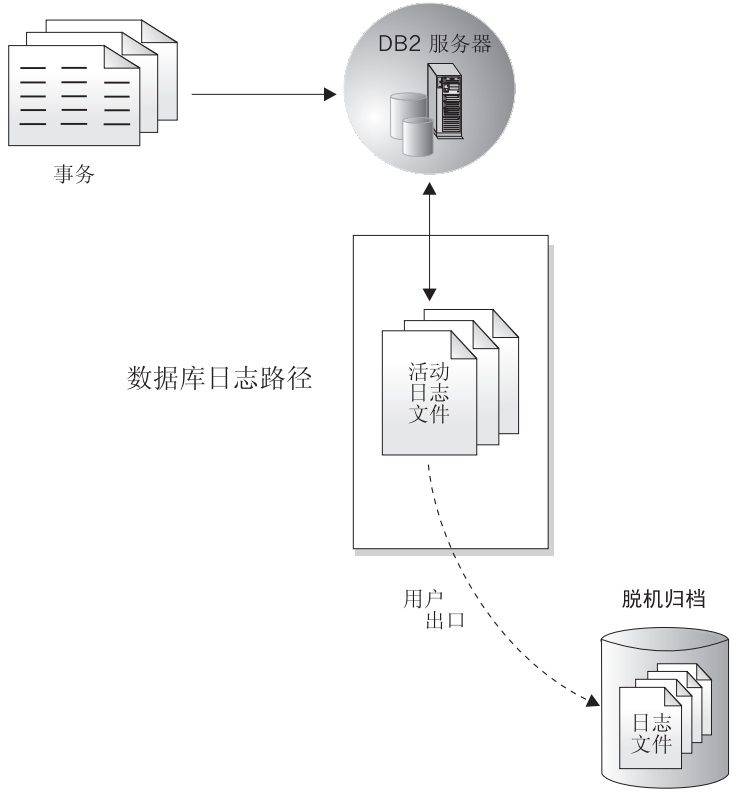
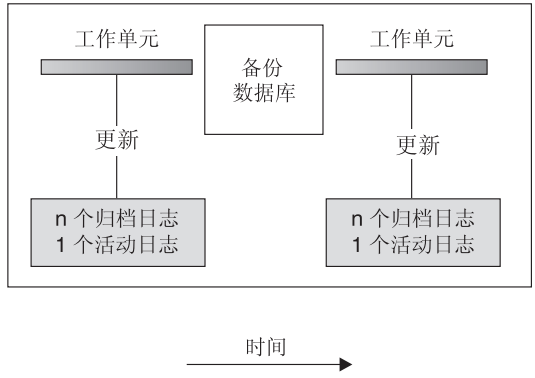


图 8. 脱机归档记录

前滚恢复可使用归档日志和活动日志来重建数据库，既可恢复到日志的末尾，也可恢复到特定的时间点。为此，前滚实用程序将在归档日志和活动日志中发现的已落实更改重新应用于复原的数据库。

前滚恢复还能使用日志重建表空间，方法是重新应用在归档日志和活动日志中已落实的更新。可以将表空间恢复至日志末尾或特定的时间点。

在联机备份操作期间，将记录对数据库的所有活动。复原联机备份映象时，必须至少将日志前滚至完成备份的时间点。为此，日志在复原数据库时必须已归档并可用。在联机备份完成后，DB2 将强制关闭当前活动的日志，从而使日志归档。这样就可确保您的联机备份有一套完整的归档日志可用于恢复。



日志在备份操作之间使用，以跟踪对数据库的更改。

图 9. 前滚恢复中活动的和已归档的数据库日志. 在运行时间较长的事务中，可以有多个活动的日志。

有两个数据库配置参数可用来更改归档日志的存储位置: *newlogpath* 参数和 *userexit* 参数。更改 *newlogpath* 参数还会影响活动日志的存储位置。关于这些配置参数的更多信息，请参阅《*管理指南: 性能*》一书。

要确定数据库日志日志路径目录中的哪些日志*范围*是归档日志，应检查数据库配置参数 *loghead* 的值。此参数指示活动的最低编号的日志。那些序号小于 *loghead* 的日志是归档日志且可移动。可使用控制中心检查此参数的值；也可使用命令行处理器和 GET DATABASE CONFIGURATION 命令来查看 “首个活动日志文件”。关于此配置参数的更多信息，请参阅《*管理指南: 性能*》一书。

注:

1. 若擦除了一个活动日志，则该数据库就会成为不可使用的，且必须将它复原之后，才能再次使用。只能前滚至擦除的第一个日志为止。
2. 如果担心活动日志（由于磁盘崩溃）可能已损坏，您应考虑对要用于进行记录的磁盘制作镜像，以使用操作系统级的镜像技术；或通过使用 NEWLOGPATH2 注册表变量，使用 DB2 级的镜像技术。

日志镜像

DB2 现在支持数据库级别的日志镜像。镜像日志文件有助于使您的数据免受:

- 活动日志的意外删除
- 硬件故障导致的数据毁坏

若担心活动日志可能（由于磁盘失效导致）已损坏，应该考虑使用一个新的 DB2 注册表变量 NEWLOGPATH2 来指定要管理活动日志副本的数据库辅助路径，从而为存储这些日志的卷建立镜像。

NEWLOGPATH2 注册表变量允许数据库将相同的第二份日志文件复制到另一路径。建议您将第二个日志路径设置到在物理上独立的磁盘（最好该磁盘也在另一磁盘控制器上）。在那种情况下，磁盘控制器不能是单个故障点。

注: 因为 Windows NT 不允许设备“安装”在任意路径名下，所以（在此平台上）不可能在单独设备上指定辅助路径。

可启用 NEWLOGPATH2（设置为 1）或禁用它（设置为 0）。缺省值为 0。如果此变量设置为 1，则辅助路径名是LOGPATH 变量的当前值后跟字符 2。例如，在 SMP 环境中，如果 LOGPATH是 /u/dbuser/sqllogdir/logpath，辅助路径将是/u/dbuser/sqllogdir/logpath2。在 MPP 环境中，如果 LOGPATH 是 /u/dbuser/sqllogdir/logpath，DB2 将把节点指示符附加到路径后，并使用 /u/dbuser/sqllogdir/logpath/NODE0000 作为主日志路径。此时，辅助日志路径将是 /u/dbuser/sqllogdir/logpath2/NODE0000。

首次启用 NEWLOGPATH2 时，实际上并不使用它，直到下次数据库启动时当前日志文件完成后才使用。这与 LOGPATH 和 NEWLOGPATH 当前的使用方法相同。

如果写到路径 1 或路径 2 时出错，数据库将把有问题的路径标记为“坏”路径，并在 db2diag.log 文件中写一条消息，然后将后继的日志记录只写到其他的“好”日志路径中。DB2 将不会再次尝试使用“坏”路径，直到当前日志文件完成。当 DB2 需要打开下一个日志文件时，将会验证此路径是否有效，如果有效就开始使用它。如果无效，DB2 将不会尝试再次使用该路径，直到下一个日志文件被第一次访问时。不会尝试使日志路径同步，但 DB2 保留了关于发生的访问错误的信息，因此在归档日志文件时可使用正确的路径。如果在写到其他“好”路径时出现故障，数据库会异常终止。

减少工作表的记录

若您的应用程序根据主表创建和填充了工作表，且您不担心这些工作表的可恢复性（因为它们可以从主表很容易地重新创建），则您可能希望在 CREATE TABLE 语句上指定 NOT LOGGED INITIALLY 参数创建工作表。使用 NOT LOGGED INITIALLY 参数的优点是，不记录在创建该表的同一个工作单元中对该表所作的任何更改（包括插入、删除、更新或创建索引操作）。这不仅减少了记录工作量，也可能提高应用程序的性能。还可以对现存表使用带 NOT LOGGED INITIALLY 参数的 ALTER TABLE 语句，来获得同样的结果。（为此，必须已用 NOT LOGGED INITIALLY 选项创建了表。）

注:

1. 可以在同一个工作单元中使用 NOT LOGGED INITIALLY 参数创建多个表。
2. 仍会记录对目录表和其他用户表的更改。

因为不记录表的更改，所以当决定使用 NOT LOGGED INITIALLY 参数时应该考虑下列事宜:

- 在落实时，必须将对表的*所有*更改写到磁盘。这意味着该落实可能占用更长的时间。
- 在创建该表的工作单元中执行任何操作而返回的错误将导致整个工作单元回滚 (SQLCODE -1476, SQLSTATE 40506)。
- 当前滚时，不能恢复这些表。若前滚操作遇到使用 NOT LOGGED INITIALLY 选项创建的表，会将该表标记为不可用。在恢复了数据库之后，访问该表的任何尝试都将返回 SQL1477N。

  注: 当创建了一个表时，将会保持对目录表的行锁定，直到执行 COMMIT 为止。要利用不记录行为，必须在创建该表的同一个工作单元中填充该表。这就存在并发情况。关于更多信息，请参阅*《管理指南: 性能》*一书中的『并发』一节。

关于创建表的更多信息，请参阅 *SQL Reference*。

若计划使用已说明临时表作为工作表，注意下列各项:

- 已说明临时表不是在目录中创建的；因此，不挂起锁定。
- 不对已说明临时表执行记录，甚至在第一个 COMMIT 之后也如此。
- 使用 ON COMMIT PRESERVE 选项来使各行在 COMMIT 之后仍留在表中；否则，将删除所有行。
- 只有创建已说明临时表的应用程序才能访问表的那个实例。
- 当删除应用程序与数据库的连接时，隐式删除该表。
- 工作单元中使用一个已说明临时表的操作中的错误不会导致该工作单元完全回滚。然而，更改已说明临时表内容的语句中的操作错误将删除该表中的所有行。回滚工作单元（或保存点）将删除已说明临时表中被该工作单元（或保存点）修改过的所有行。

有关已说明临时表及其限制的更多信息，参见 *SQL Reference* 中的 DECLARE GLOBAL TEMPORARY TABLE 语句。

数据库记录的配置参数

数据库配置文件包含与前滚恢复相关的参数。这些参数的缺省值不支持此类型的恢复，如果您计划使用它则必须更改某些缺省值。关于配置 DB2 通用数据库的更多信息，请参阅*《管理指南: 性能》*一书。

主日志 (logprimary)

此参数指定将要创建的主日志的数量。

主日志，无论是空的还是满的，都需要相同的磁盘空间容量。因此，若配置的日志多于需要的日志，将会不必要地占用磁盘空间。若配置的日志太少，可能会遇到日志满载的情况。当选择要配置的日志数时，必须考虑建立的每个日志的大小，以及应用程序是否可以处理日志满载的情况。

若允许现存数据库进行前滚恢复，则要将主记录数更改为主日志和辅助日志的数量之和，再加 1。为启用前滚恢复的数据库中的 LONG VARCHAR 和 LOB 字段记录了其他信息。

整个日志文件大小限制为 32 GB。即日志文件数 (*logprimary* + *logsecond*) 乘以每个日志文件的大小（以字节计）(*logfilsiz* * 4096) 必须小于 32 GB。

关于此配置参数的更多信息，请参阅《*管理指南: 性能*》一书。

辅助日志 (logsecond)

此参数指定创建并用于恢复（如果需要的话）的辅助日志文件的数目。

如果主日志文件已满，可按需要一次分配一个辅助日志文件（大小为 *logfilsiz*），最多可分配由此参数指定的最大数目。如果需要多于已配置的辅助日志文件数的辅助日志文件，将返回一个错误并且对数据库的活动将停止。

关于此配置参数的更多信息，请参阅《*管理指南: 性能*》一书。

日志大小 (logfilsiz)

此参数以 4 KB 的页数指定每个配置日志的大小。

在总的活动日志空间中，您可配置的日志大小有 32 GB 的逻辑限制。此限制源自 *logfilsiz* 上的高端限制 65535，以及 (*logprimary* + *logsecond*) 上的高端限制 128。因此，((*logprimary* + *logsecond*) * *logfilsiz*) < (32 GB / 4096)。

日志文件的大小对性能有直接的影响。如果某个数据库配置为保留日志，则每次填写日志时都会发出一个分配并初始化新日志的请求。增加日志的大小可减少分配和初始化新日志所需的请求的数量。（但是要记住，使用较大的日志大小，会花费更多的时间来格式化每个新日志）。新日志的格式化对于与数据库连接的应用程序是透明的，以使数据库性能不受格式化的影响。

假定一个应用程序将数据库保持为打开以使打开数据库时的处理时间最短（参见第46页的『增强恢复性能』），则日志文件大小应由建立脱机归档日志副本所花的时间确定。

用于存储脱机归档日志的设备和用于建立副本的软件的数据传送速度，最少必须与数据库管理器日志中写入数据的平均速率匹配。若该传送速度跟

不上新日志数据生成的速度，而且记录活动持续足够长的时间（由可用磁盘空间的容量确定），可能会用尽磁盘空间。若发生这种情况，数据库处理将停止。

当使用磁带或光盘媒体时，数据传送速度最为重要。（关于使用不同媒体来存储日志的信息，参见第411页的『附录H. 数据库恢复的用户出口』。）某些磁带机需要相同的时间来复制文件，而与它的大小无关。必须确定归档设备的能力。

磁带设备还有其他注意事项。归档请求的频率是很重要的。例如，如果用于完成任何复制操作的时间是 5 分钟，则日志应该足够大，以便在工作负荷高峰期间可保存 5 分钟的日志数据。磁带机可能存在设计限制，它会限制每天的操作数。当确定日志大小时，必须考虑这些因素。

将日志文件的丢失降低至最小程度，也是设置日志大小时的一个重要注意事项。归档会占用整个日志。若使用单个的大日志，则会增加归档之间的时间。若包含该日志的媒体失效，某些事务信息将可能丢失。减小日志大小会增加归档的频率，但可以减少由于媒体失效而丢失信息，因为可以使用丢失的日志之前的那些较小的日志。

日志缓冲区 (logbufsz)

此参数允许您指定在将日志记录写入磁盘之前用作日志缓冲区的数据库共享内存的容量。当发生下列任何一项事件时会将日志记录写入磁盘:

- 事务落实
- 日志缓冲区已满
- 发生了某些其他的内部数据库管理器事件。

增加日志缓冲区的大小可使与记录日志操作相关的输入／输出 (I/O) 活动更有效，因为将日志记录写到磁盘中的频率更低，而每次写入的记录却更多。

对组的落实次数 (mincommit)

此参数允许您延迟将日志记录写入磁盘，直到执行了最小数目的落实为止。此延迟可有助于减少与写入日志记录相关的数据库管理员器额外开销，这样若您有多个应用程序对数据库运行，且在很短的内该应用程序请求了许多落实，则可改进性能。

仅当此参数的值大于 1，且与该数据库连接的应用程序的数量大于此参数的值时，才会对落实进行这种分组。落实组合生效时，保持应用程序落实请求，直到经过 1 秒钟或落实请求数等于此参数的值为止。

新日志路径 (newlogpath)

数据库日志最初是在 SQLOGDIR 中创建的， SQLOGDIR 是数据库目录的子目录。通过更改此配置参数的值以指向另一目录或另一设备，可以更

改放置活动日志及以后的归档日志的位置。若数据库被配置为允许前滚恢复，则不要将当前存储在数据库日志路径目录中的归档日志移至新位置。

因为可以更改该日志路径的位置，因此前滚恢复所需的日志可以存在于不同的目录中或不同的设备上。在前滚操作期间可更改此配置参数的值，以允许您访问位于多个位置的日志。

必须跟踪这些日志的位置。

只有数据库处于一致状态时才会应用所作的更改。配置参数 *database_consistent* 将返回数据库的状态。关于此配置参数的更多信息，请参阅《*管理指南: 性能*》一书。有关在数据库处于不一致的状态时数据库日志所起的作用的信息，参见『管理日志文件』。

日志保留 (logretain)

如果 logretain 设置为 RECOVERY，归档日志将保留在数据库日志路径目录中，而将数据库视为可恢复的数据库则意味着启用了前滚恢复。

如果 logretain 设置为 CAPTURE，复制捕获程序将在捕获程序完成后，调用 PRUNE LOGFILE 命令来删除日志文件。如果想对数据库进行前滚恢复，就不应将 logretain 设置为 CAPTURE。

用户出口 (userexit)

此参数使数据库管理器调用用户出口程序来归档和检索日志。归档日志文件的位置不是活动日志路径。如果 *userexit* 设置为 ON，将启用前滚恢复。关于用户出口程序的更多信息，参见第411页的『附录H. 数据库恢复的用户出口』。

管理日志文件

管理数据库日志时，应考虑以下事项:

- 归档日志的编号方案以 S0000000.LOG 开始，直到 S9999999.LOG，符合日志文件的最大潜在大小 10000000。如果发生以下情况，数据库管理器将复位到 S0000000.LOG:
  - 数据库配置文件更改为启用前滚恢复
  - 数据库配置文件更改为*禁用*前滚恢复
  - 已使用了 S9999999.LOG。

  在复原数据库之后（执行或不执行前滚恢复），DB2 重新使用日志名。数据库管理器会确保在前滚恢复期间不应用不正确的日志，但是它无法检测到必需的日志的位置。必须确保前滚恢复可以找到正确的日志。

  前滚操作成功完成后，使用的最后一个日志会截断，而记录日志操作会从下一个按顺序的日志开始。日志路径目录中，序列号大于用于前滚恢复的最后一个

日志的任何日志都将重新使用。截断的日志中，跟在截断点后的任何条目都将用 0 覆盖。确保在调用前滚实用程序前建立了日志的副本。（可以调用用户出口程序，将日志复制到另一位置。关于用户出口程序的信息，参见第411页的『附录H. 数据库恢复的用户出口』。）

- 如果某个数据库尚未激活（通过 ACTIVATE DATABASE 命令），DB2 会在所有应用程序已从该数据库断开连接后截断当前日志文件。下次有应用程序与该数据库连接时，DB2 开始把日志记录到一个新日志文件中。如果系统上产生了很多小的日志文件，则可能需要考虑使用 ACTIVATE DATABASE 命令。使用该命令不仅能节省应用程序连接时初始化数据库所用的开销，还可节省分配大日志文件、截断它然后再分配新的大日志文件所用的开销。
- 因为日志文件名是重复使用的（参见第36页的图10），所以归档日志可能会与某个数据库的两个或多个不同的*日志序列*相关。例如，如果您要恢复“备份 2”，可以使用两种可能的日志序列。如果在完整的数据库恢复期间前滚至某个时间点，然后在到达日志末尾前停止，则您已创建了一个新的日志序列。两个日志序列不能合在一起。如果联机备份映象跨越了第一个日志序列，则必须使用此日志序列来完成前滚恢复。

  如果在恢复之后创建了新的日志序列，则旧日志序列中的任何表空间备份映象都是无效的。如果数据库复原操作后紧跟着表空间复原操作，则复原实用程序将无法识别旧日志序列上的表空间备份映象。在实际上前滚了数据库前，将使用的日志序列都是未知的。如果表空间在旧日志序列上，它一定会由表空间前滚操作“捕获”。使用无效备份映象的复原操作可能会成功完成，但表空间前滚操作将失败，而表空间将置于前滚暂挂状态。

  例如，假设一个表空间级的备份操作“备份 3”在顶级日志序列的 S0000013.LOG 和 S0000014.LOG 之间完成（参见第36页的图10）。如果要使用数据库级的备份映象“备份 2”进行复原和前滚，您将需要前滚 S0000012.LOG。然后，可以继续前滚到顶端日志序列或（较新的）底端日志序列。如果前滚底端日志序列，将不能使用表空间级的备份映象“备份 3”来执行表空间复原和前滚恢复。

  要使用表空间级的备份映象“备份 3”完成到日志末尾的表空间前滚操作，需要复原数据库级的备份映象“备份 2”，然后使用顶端日志序列前滚。一旦复原了表空间级的备份映象“备份 3”，就可以启动到日志末尾的前滚操作。

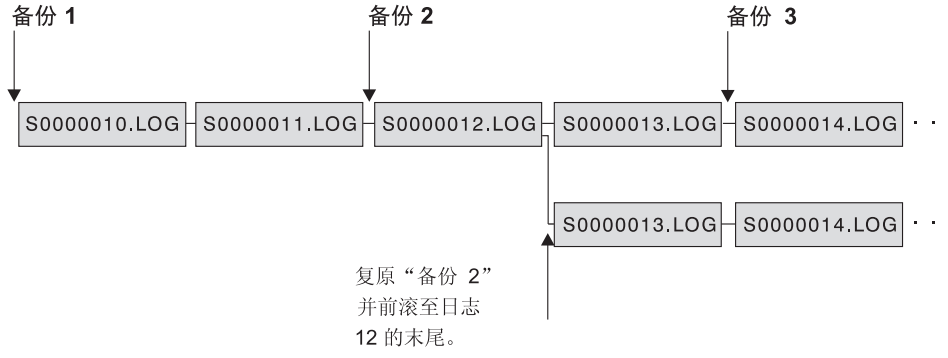


图 10. 重新使用日志文件名

通过用户出口程序管理日志文件

在调用用户出口程序来归档和检索日志文件时应注意以下注意事项:

- 数据库配置文件参数 *userexit* 指定在数据库的前滚恢复期间，数据库管理器是否调用用户出口程序来归档文件或检索日志文件。当前滚实用程序需要一个在日志路径目录中找不到的日志文件时，会发出检索日志文件的请求。

  注: 在 Windows NT 上，不能使用 REXX 用户出口来归档日志。
- 在归档过程中，当一个日志文件满时，即使该日志文件仍是活动的且需要用于正常的处理，仍将它传送至用户出口。这使得数据的副本可以尽快地从易丢失数据的媒体中转移。传送至用户出口的日志文件保存在日志路径目录中，直到不再需要它用于正常处理为止。这样就重新使用了磁盘空间。
- DB2 在启动用户出口程序来归档文件时，将日志文件打开为“读”方式。在某些平台上，这可防止用户出口程序能删除日志文件。其他平台（如 AIX）允许进程（包括用户出口程序）删除日志文件。用户出口程序在日志文件归档后永远不能删除它，因为该文件可能仍是活动的并是崩溃恢复所需的。DB2 管理在归档日志文件时重新使用的磁盘空间。
- 当日志文件已归档并且不是活动的时，DB2 不会删除该文件但在需要这样的文件时将它重命名为下一个日志文件。这将改进性能，因为创建新的日志文件（而不是重命名文件）会写出所有页以保证有足够的磁盘空间。重新使用磁盘上的页，比释放这些页再重新获得需要的页更好。
- DB2 在崩溃恢复或回滚期间，将不会调用用户出口程序来检索日志文件。
- 用户出口程序不保证可前滚恢复至故障点，只试图将故障范围减小。日志文件填写完后，排队等待用户出口例程使用。若在一个日志文件写完之前包含该日

志的磁盘失效，则该日志文件中的数据将会丢失。另外，由于文件是排除等待归档的，磁盘可能会在备份了所有文件之前就发生故障，从而导致队列中的任何日志文件都丢失。

- 每个日志文件的配置大小对用户出口都有直接影响。若每个日志文件都非常大，磁盘失效时将会丢失大量数据。如果数据库是由小的日志文件配置的，会导致更频繁地将数据传送至用户出口例程。

  然而，若您要将数据移到慢速设备（如磁带）上，您可能希望有较大的日志文件以防建立该队列。若该队列变满时，将不处理归档与检索请求。只有在队列上有空间时，处理才继续。未处理的请求将不会自动重新排队。

- 只有配置了 *userexit* 才会执行向用户出口程序发出的归档请求，且每次都要填写一个活动的日志文件。有可能当最后与数据库断开时活动的日志文件仍然未满，但仍然会对填写了一部分的活动日志文件调用用户出口程序。

  注: 要释放未用的日志空间，在将日志文件归档前要将它截断。

- 应将该日志的副本保存到另一个物理设备上，以便在包含该日志文件的设备出现媒体故障时，前滚恢复可使用这个脱机日志文件。这个设备不应是包含该数据库数据文件的同一设备。
- 在某些情况下，若在接收到用户出口程序对归档请求的肯定应答前数据库已经关闭，那么数据库管理器会在数据库打开时发送另一个请求。因此，一个日志文件会多次归档。
- 若用户出口程序接收到对一个不存在的文件进行归档的请求（因为有多个归档请求，而在第一次归档操作成功后即将该文件删除），或接收到对一个不存在的文件进行检索的请求（因为此文件位于另一个目录中，或已到达日志的末尾），则应忽略此请求并发送一个成功的返回码。
- 用户出口程序应允许在某个时间点恢复后存在具有相同名称的不同的日志文件；该程序应该写成保留日志文件，同时将那些日志文件与正确的恢复路径关联（参见第34页的『管理日志文件』）。
- 若两个或更多数据库正在同时使用一个设备，且其中一个操作涉及到前滚操作，则在当前驱动器中的媒体上可能不存在前滚恢复所需的日志文件。可能有两种情况:
  - 若用户出口程序将 0（成功）返回码传回给数据库管理器，且未检索到请求的日志文件，则数据库管理器认为前滚操作已完成到日志的末尾，于是前滚操作停止。但前滚处理可能并未进行到日志的末尾。
  - 若返回了非零返回码，数据库将处于前滚暂挂状态，这时您必须继续或停止前滚处理。

为防止这两种情况发生，可以确保在前滚操作期间在调用用户出口程序的节点上没有任何其他数据库是打开的，或者编写一个用户出口程序来处理这种情况。

在日志目录已满时将事务分块

可设置新的 DB2 注册表变量 DB2_BLOCK_ON_LOG_DISK_FULL 以防止当 DB2 不能在活动的日志路径中创建新日志文件时出现“磁盘已满”的错误。

DB2 会在每 5 分钟即尝试创建日志文件直到成功为止。如果配置数据库时 *userexit* 参数设置为 ON，DB2 还会检查日志文件的归档操作是否已完成。如果有不活动的日志文件已成功归档，DB2 会将该不活动的日志文件重命名为新的日志文件名并继续。每次尝试后，DB2 都会将一条消息写到 db2diag.log 文件中。要确认您的应用程序是否由于“日志磁盘已满”而挂起，唯一的方法就是监视 db2diag.log 文件。

在成功创建日志文件之前，尝试更新表表数据的任何用户应用程序都不能落实事务。只读查询可能不会直接受影响，但如果查询需要访问被更新请求锁定的数据或由更新应用程序在缓冲池中修正的数据页时，只读查询的状态也会象挂起一样。

按需进行的日志归档

DB2 现在支持在任何时候关闭（如果启用了用户出口选项，还包括归档）可恢复数据库的活动日志。因此您可以到某个已知点为止的一组完整的日志文件，然后使用这些日志文件来更新一个备用数据库。

通过调用 ARCHIVE LOG 命令或调用 **db2ArchiveLog** API 可以启动一个按需进行的日志归档操作，在第285页的『ARCHIVE LOG』 和 第298页的『db2ArchiveLog - 归档活动日志 API』 中分别对它们进行了描述。

使用原始日志

可使用原始设备来存储数据库日志。这样做既有优点，又有缺点。

- 优点是:
  - 可以将 26 个以上的物理驱动器与一个系统连接。
  - 文件 I/O 路径的长度较短。这可提高系统的性能。应执行基准测试，以评估对于工作负荷是否有可度量的效益。
- 缺点是:
  - 该设备不能被其他应用程序共享；*必须*将整个设备分配给 DB2。
  - 任何从该设备中备份或复制的操作系统实用程序或第三方工具不能在该设备上操作。

– 若指定了错误的物理驱动器号，可以很容易地擦除现存驱动器上的文件系统。

可用 *newlogpath* 数据库配置参数配置原始日志。有关用于指定原始设备的语法示例，请参阅《*管理指南: 实现*》一书的『原始 I/O』一节。但在这样做以前，要考虑以上列出的优点和缺点，以及以下列出的附加注意事项:

- 仅允许一个设备。在操作系统级别，可在多个磁盘上定义设备。DB2 将进行操作系统调用以确定以 4 KB 页为单位的设备大小。

  如果使用多个磁盘，这样做可提供更大的设备，而由此产生的条带化可通过提供更快的 I/O 吞吐量来改进性能。

- DB2 将试图写入设备的最后一个 4 KB 页。若设备大小大于 2 GB，则在不支持 2 GB 以上的设备的操作系统上，写至最后一页的尝试将失败。在这种情况下，DB2 将尝试使用所有页，直至达到受支持的极限。

  关于设备大小的信息用来指示在操作系统的支持下可用于 DB2 的设备的大小（以 4 KB 页为单位）。DB2 可写入的磁盘空间容量称为*可用的设备大小*。

  DB2 不使用设备的第一个 4 KB 页（此空间通常由操作系统用于其他用途）。这意味着可用于 DB2 的总空间是*设备大小* = *可用设备大小* - *1*。

- 不使用 *logsecond* 参数。DB2 将不分配辅助日志。活动日志空间的大小是由 *logprimary* x *logfilsiz* 而得的 4 KB 页的倍数。

- 仍将日志记录分组为日志块，每个日志块具有 4 KB 页的日志文件大小 (*logfilsiz*)。日志块在原始设备中是连续的。每个块还包括用于存放块标题的额外两页。这意味着设备可支持的*可用日志块数*是*设备大小* / (*logfilsiz* + 2)

- 设备必须足够大，以支持活动日志空间。即，*可用日志块数*必须大于（或等于）为 *logprimary* 配置参数指定的值。

- 若使用的是循环记录，*logprimary* 配置参数将确定要写入设备的日志块的数量。这可能导致设备上出现未使用的空间。

- 若使用日志保留 (*logretain*) 而不用用户出口程序，当用完所有*可用数目的日志块*后，产生更新的所有操作将接收到日志已满的错误。此时，必须关闭数据库并执行脱机备份以确保可恢复性。在数据库备份操作之后，写至设备的日志记录将丢失。这表明不能使用更早的数据库备份映象来复原该数据库，然后将其前滚。若在*可用数目的日志块*全部用完之前建立了一个数据库备份，则可以复原并前滚该数据库。

- 若将日志保留 (*logretain*) 和用户出口程序一起使用，在用日志记录填写每个日志块时，为每个日志块调用用户出口程序。该用户出口必须能够读取设备，并将归档的日志存储为文件。DB2 将不调用用户出口程序来将日志文件检索到原始设备。而是，在前滚恢复期间，DB2 读取块标题以确定原始设备是否包含所需的日志文件。若在原始设备中找不到必需的日志文件，DB2 将搜索溢出日志路径。若

仍找不到日志文件，DB2 将调用用户出口以便在溢出日志路径中检索日志文件。如果不对前滚操作指定溢出日志路径，DB2 将不会调用用户出口程序来检索日志文件。关于调用用户出口程序的其他信息，参见第413页的『调用格式』。

- 如果正在使用 DPROP 并将日志写入原始设备，则读取日志 API 将不会调用用户出口程序来检索日志文件。然而，若在该设备上可执行请求的日志记录，仍将返回它们。若请求的日志比设备上最旧的日志更早，则不返回它们（此行为类似于 DB2 找不到包含请求的日志记录的日志文件）。

  注:

  1. 建议在使用原始设备来记录时不要使用 DPROP。
  2. 如果使用 **sqlurlog** API，则不应使用原始设备来记录。

丢失日志

删除数据库会擦除当前数据库日志路径目录中的所有日志。在删除数据库之前，应考虑建立这些日志的副本。

如果将一个数据库前滚至某特定时间点，会重新使用最后一个日志及该日志后的所有现有日志：您将失去恢复此特定时间点以外的其他日志的能力。因此，在开始对某时间点的恢复操作*之前*，应复制当前数据库日志路径目录中的所有日志。

完成前滚操作时，就将最后一个落实事务的日志文件截断，并从下一个顺序日志开始记录。如果没有日志在截断前的副本，以及具有较高序列号的日志副本，将不能恢复某特定时间点以外的数据库。（一旦正常的数据库活动恢复后，将创建可用于任何后继恢复操作的新日志。）

若更改了日志路径目录，然后除去了该子目录或擦除了在该日志路径中需要的子目录中的任何日志，则当打开数据库时数据库管理器将会在缺省日志路径 SQLOGDIR 中查找这些日志。若找不到这些日志，数据库将被置于备份暂挂状态，而且只有备份该数据库后数据库才可使用。即使该子目录中没有任何日志，也必须建立此备份。

若您丢失了包含联机备份结尾时间点的日志，又要将对应的已复原映象前滚，该数据库将不可使用。要使该数据库可使用，必须根据另一个备份和所有相关的日志来复原该数据库。

可能会遇到类似如下的情况：想要对一个完整的数据库执行时间点恢复，但您担心在恢复过程期间可能丢失日志。（若在上一次备份数据库映象时到您希望恢复数据库时这段时间有相当多的归档日志，可能会发生此情况。）

首先，应该将所有适用的日志复制到“安全”位置。然后可以运行 RESTORE 命令并使用前滚恢复方法，将数据库前滚至您希望的时间点。若在此过程期间需要的任何日志损坏或丢失，在其他位置还有所有这些日志的副本。

了解恢复历史记录文件

恢复历史记录文件是与每个数据库一起创建的，且在发生下列情况时自动更新:

- 备份了数据库或表空间
- 复原了数据库或表空间
- 前滚了数据库或表空间
- 创建了表空间
- 改变了表空间
- 停顿表空间
- 重命名表空间
- 删除表空间
- 装入表
- 删除表
- 重新组织表
- 更新了表统计信息

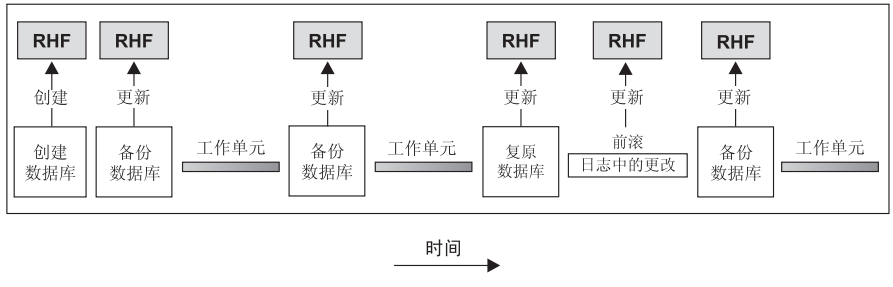


图 11. 创建和更新恢复历史记录文件

可以使用此文件中汇总的备份信息，将数据库的全部或部分恢复至给定的时间点。该文件中的信息包括:

- 唯一地标识每个条目的标识符 (ID) 字段
- 已复制的部分数据库和复制方法
- 建立副本的时间

- 副本的位置（指示设备信息和访问副本的逻辑方式）
- 进行上次复原操作的时间
- 重命名表空间的时间，显示了该表空间的先前名称和当前名称
- 备份操作的状态：活动、不活动、到期的或删除的
- 数据库备份保存的或前滚恢复操作期间处理的最后一个日志序号。

要查看恢复历史记录文件中的条目，使用 LIST HISTORY 命令。关于此命令的更多信息，参见 第288页的『LIST HISTORY』。

注：当复原和前滚操作执行到日志的末尾时，调用 LIST HISTORY 命令后显示的备份标识表示了结束时间：即备份标识值是 99991231235959。当执行前滚操作时，仅以此方式变换备份标识。

每个备份操作（数据库备份、表空间备份或增量备份）都包括复制恢复历史记录文件。将该恢复历史记录文件链接至数据库。删除数据库会删除恢复历史记录文件。将数据库复原至新位置会复原该恢复历史记录文件。复原不会覆盖现存的历史恢复文件。

若当前数据库不能使用或不可用，且相关的恢复历史记录文件被损坏或被删除，则 RESTORE 命令中的一个选项只允许复原恢复历史记录文件。这样，可以复查该恢复历史记录文件，以提供有关要将哪个备份用于复原该数据库的信息。

该文件的大小由 *rec_his_retentn* 配置参数控制，该参数指定该文件中条目的保存期（以天计）。即使将此参数的值设置为零 (0)，仍会保留最新的完整数据库备份（加上它的复原集）。（除去此副本的唯一方法是使用带 FORCE 选项的 PRUNE。）保存期的缺省值是 366 天。可使用 -1 将保存期设置为无限多天。在这种情况中，需要显式删除该文件。关于此配置参数的更多信息，请参阅《*管理指南：性能*》一书。

垃圾收集

虽然可在任何时间使用 PRUNE HISTORY 命令（参见 第291页的『PRUNE HISTORY/LOGFILE』）从历史记录文件中除去条目，仍建议您将该任务留给 DB2 来处理。记录在恢复历史记录文件中的 DB2 数据库备份的数目由 DB2 *垃圾收集* 来自动监视。DB2 垃圾收集在数据库备份操作成功完成后调用；也在数据库复原操作成功完成后调用。配置参数 *num_db_backups* 定义保留了多少个活动的完整（而不是增量）数据库备份映象。此参数的值用于从上一个条目开始扫描历史记录文件。

在执行每个完整的数据库备份操作后，可使用 *rec_his_retentn* 配置参数从历史记录文件中删除到期的条目。

*活动的数据库备份*是可使用当前日志进行复原和前滚，以恢复该数据库的当前状态的的一种备份。而*不活动的数据库备份*是在复原时会将数据库移动回先前状态的一种备份。

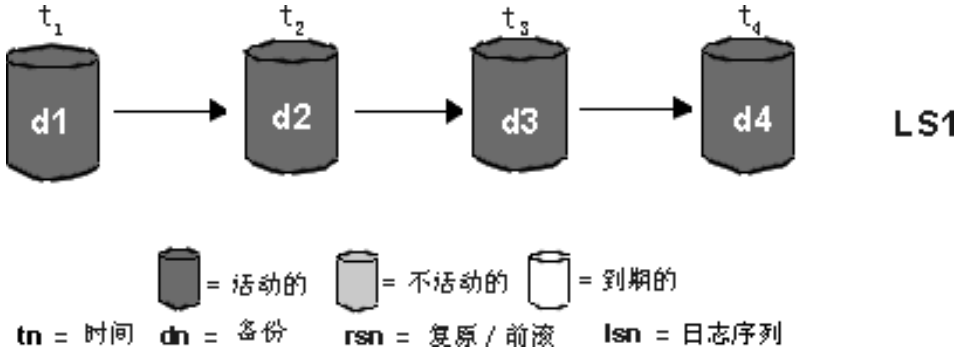


图 12. 活动数据库备份. num_db_backups 的值已设置为 4。

不再需要的所有活动数据库备份映象都标记为“到期的”。因为有更新的备份映象可用，所以可将这些映象视为不需要的。在该数据库备份映象到期前所建立的所有表空间备份映象和装入副本也标记为“到期的”。

标记为“不活动的”所有数据库备份映象以及在建立已到期的数据库备份映象前已建立的备份映象，也将标记为“到期的”。所有相关的“不活动”表空间备份映象和装入副本也标记为“到期的”。

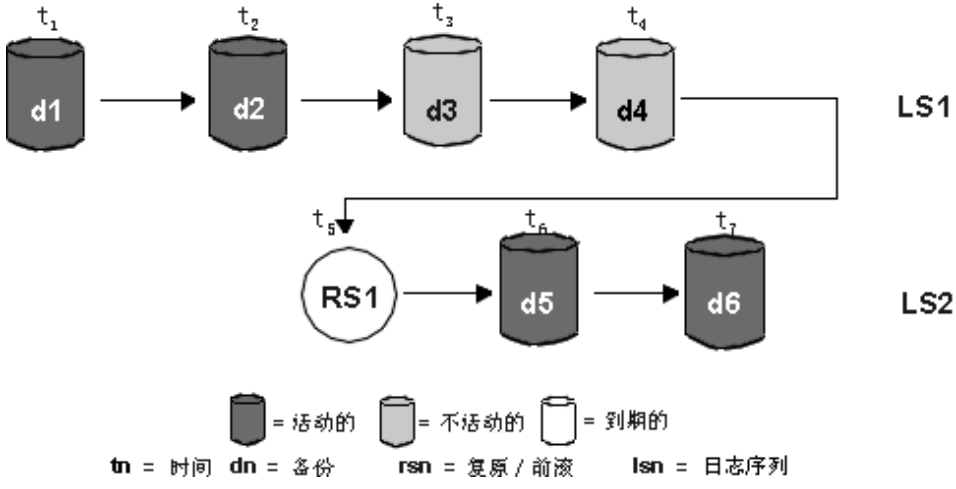


图 13. 不活动数据库备份

若复原了一个活动的数据库备份映象，但它不是历史记录文件中记录的最新数据库备份，则属于同一个日志序列的任何后续数据库备份映象将被标记为“不活动的”。

若复原了一个不活动的数据库备份映象，则属于当前日志序列的任何不活动的数据库备份都被再次标记为“活动的”。不再在当前日志序列中的所有数据库备份映象都标记为“不活动的”。

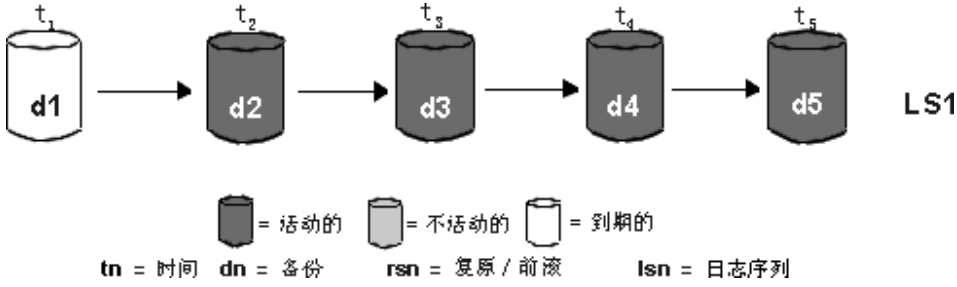


图 14. 到期数据库备份

DB2 垃圾收集也负责将 DB2 数据库或表空间备份映象的历史记录文件条目标记为“不活动”，条件是该备份与当前*日志序列*（也称为当前*日志链*）不对应。当前日志序列由已复原的 DB2 数据库备份映象和已处理的日志文件确定。一旦数据库备份映象复原后，所有后继的数据库备份映象都变得“不活动”，因为已复原的映象将开始一个新的日志链。（在不通过前滚来复原备份映象时实现。如果已发生了前滚操作，则日志链中在中断后建立的所有数据库备份都将标记为“不活动”。可以想像，由于已对包含已损坏的当前备份映象的整个日志序列执行了前滚实用程序，所以必须复原旧的数据库备份映象。）

如果在复原表空间级的备份映象后，数据库的当前状态不能通过应用当前日志序列来达到，则表空间级的备份映象会变为“不活动”。

如果备份映象包含 DATALINK 列，将联系负责运行 DB2 Data Links Manager 的所有 Data Links 服务器以请求进行垃圾收集。然后 DB2 垃圾收集会删除包含在到期备份中的相关 Data Links 服务器文件备份，但不删除在下次数据库备份操作之前解链的那些备份。

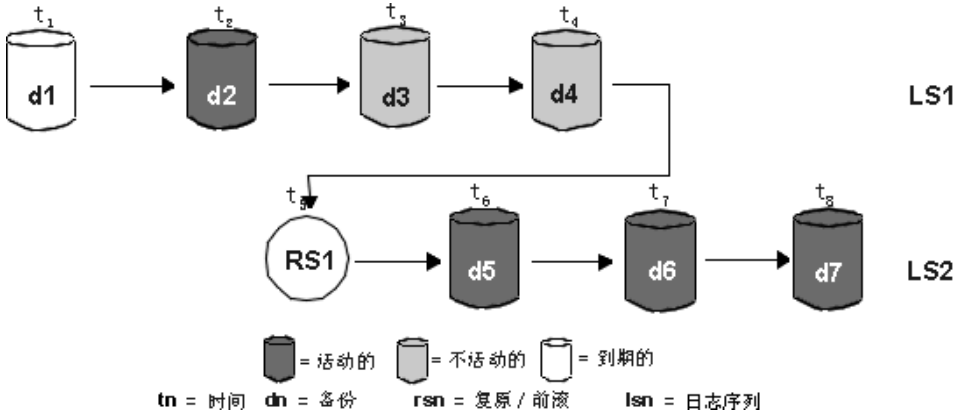


图 15. 混合活动的、不活动的和到期的数据库备份

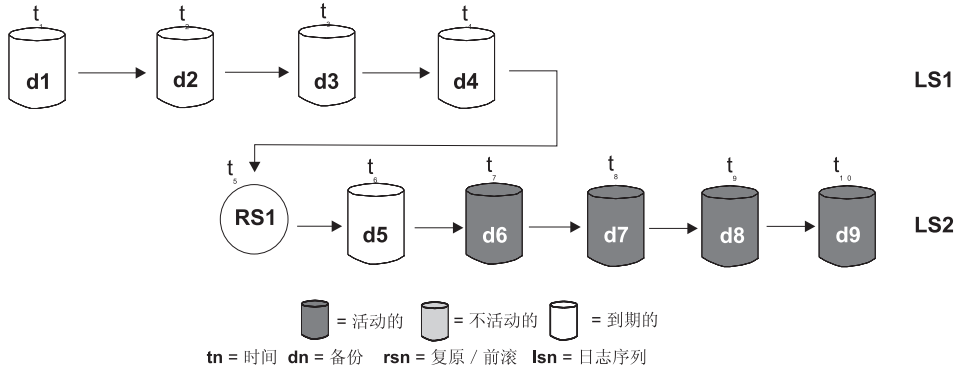


图 16. 到期日志序列

了解表空间状态

表空间的当前状态由它的状态反映。与恢复相关的最常见表空间状态是:

- *前滚暂挂*。表空间在复原后可发生了输入 / 输出 (I/O) 错误后被置于此状态。复原后，表空间可前滚到日志的末尾可某时间点。发生了 I/O 错误后，表空间必须前滚到日志的末尾。
- *正在前滚*。表空间在对它进行的前滚操作正在进行中时，被置于此状态。一旦前滚操作成功完成，表空间就不再处于“正在前滚”状态。如果取消了对表空间的前滚操作，表空间也会结束此状态。
- *复原暂挂*。如果取消了对表空间的前滚操作，或对表空间的前滚操作遇到了不可恢复的错误（此时必须再次复原并前滚表空间），会将表空间置于此状态。

- *备份暂挂*。在前滚操作的某个时间点后，或不带有复制选项的装入操作后，表空间将置于些状态。在可使用该表空间之前必须对其备份。（如果未进行备份就不能更新表空间，但允许只读操作。）

增强恢复性能

当考虑恢复性能时，应注意下列各项:

- 可以将日志置于单独的设备上，以提高频繁更新的数据库的性能。在联机事务处理 (OLTP) 环境中，更常见的是需要 I/O 来将数据写至日志而不是存储数据行。将日志置于单独的设备上，可将在日志和数据库文件之间进行移动所需的磁盘臂移动最小化。

  还应该考虑该磁盘上的其他文件。例如，将日志移至一个系统中用于系统分页的磁盘，而该磁盘没有足够的实内存，这样会破坏调整。

- 要缩短完成一次复原操作所需的时间:
  - 调整复原缓冲区大小。缓冲区大小必须是在备份操作期间使用的缓冲区大小的倍数。
  - 增加缓冲区的数量。

    若使用多个缓冲区和 I/O 通道，应该使用的通道数至少是缓冲区数的两倍，以确保通道不必等待数据。所用的缓冲区大小也影响复原操作的性能。理想的复原缓冲区大小应该是表空间的范围大小的倍数。

    若有多个具有不同范围大小的表空间，则指定是最大范围大小的倍数的一个值。

    建议的*最小*缓冲区数目是媒体设备或容器的数目加上对 PARALLELISM 选项指定的数目。
  - 使用多个源设备。
  - 对于复原操作，将 PARALLELISM 选项设置为至少比源设备数大 1。

- 若一个表包含大量的长整数字段和 LOB 数据，则复原它可能会占用非常多的时间。若允许数据库进行前滚恢复，则 RESTORE 命令能够复原选择的表空间。若该长整数字段和 LOB 数据对于您的业务很重要，应参照完成这些表空间的备份任务所需的时间考虑复原这些表空间。通过将长整数字段和 LOB 数据存储在单独的表空间中，不选择复原包含该长整数字段和 LOB 数据的表空间，可以减少完成复原操作所需的时间。若可从单独的源复制 LOB 数据，则当创建或改变一个表以包括 LOB 列时选择 NOT LOGGED 选项。若选择不复原包含长整数字段和 LOB 数据的表空间，但需要复原包含该表的表空间，则必须前滚到日志末尾，以便所有包含表数据的表空间都是一致的。

注: 若备份包含表数据的一个表空间，而未备份相关的长整数或 LOB 字段，则不能对该表空间执行时间点前滚恢复。必须将一个表的所有表空间同时前滚至同一个时间点。

- 下列说明适用于备份和复原操作:
  - 应使用多个 I/O 缓冲区和设备。
  - 分配至少两倍于正在使用的设备数量的缓冲区。
  - 不要使 I/O 设备控制器带宽超载。
  - 使用数量多的小缓冲区，而不使用数量少的大缓冲区。
  - 根据系统资源调整缓冲区的数量和大小。

并行恢复

DB2 现在使用多个代理程序来执行崩溃恢复和数据库前滚恢复。您可能会发现执行这些操作期间的性能更佳（尤其是在对称多处理机 (SMP) 上）；在数据库恢复期间使用多个代理程序可利用 SMP 机器上可用的多余 CPU。

此项增强功能引进的新代理程序类型是 db2agnsc。DB2 根据机器上 CPU 的数量，选择将用于数据库恢复的代理程序的数量。对于 SMP 机器，使用的代理程序的数量是（CPU 数 + 1）。在具有单个 CPU 的机器上，可使用多个代理程序以更有效的读取日志、处理日志记录以及预取数据页。

DB2 将日志记录分发到这些代理程序以使它们可在适当的时候并发地重新应用。例如，可通过此方法使那些与插入、删除、更新、添加键和删除键操作相关的日志记录的处理并行化。因为日志记录是在页级并行化的（相同数据页上的日志记录由相同的代理程序处理），所以即使所有工作都是对一个表完成的，性能还是会增强。

DB2 Data Links Manager 注意事项

下面几节提供的信息适用于包含 DATALINK 列的表。有关 DATALINK 列的完整描述，请参阅 *SQL Reference* 中的 CREATE TABLE 语句。

崩溃恢复注意事项

当一个应用程序发出的 SQL 请求涉及到运行 DB2 Data Links Manager 的 Data Links 服务器时（请求使用具有 FILE LINK CONTROL 属性的 DATALINK 列），数据库管理程序会将该工作分发到 Data Links 服务器。它也跟踪哪些 Data Links 服务器参与了该事务。当该应用程序对一个事务发出 COMMIT 时，数据库管理程序会使用两阶段落实协议来落实该事务。在第一个阶段，数据库管理程序写下一条 PREPARE 日志记录，并将 PREPARE 请求分发到所有 Data Links 服务器。然后每个 Data Links 服务器用下列其中一项应答:

- YES; 表示 Data Links 服务器落实准备就绪
- NO; 由于错误，Data Links 服务器未准备落实。

若所有 Data Links 服务器都回答 ″YES″，则认为第一个阶段成功。

第二个阶段的处理取决于第一个阶段的结果。若至少有一个 Data Links 服务器回答了 ″NO″，数据库管理程序会将 ABORT 请求分发到参与的所有 Data Links 服务器。回滚该事务，并将原因码为 ″03″ 的错误消息 SQL0903N 返回给该应用程序。否则，数据库管理程序按照通常 Data Links 服务器不参与的情况，继续落实该事务。在此处理结束时，它将一个 COMMIT 请求分发到参与该事务的所有 Data Links 服务器。

若某个 Data Links 服务器上发生故障，使某些事务处于 PREPARED 状态，则这些事务称为*不确定事务*。数据库管理程序负责跟踪这些事务的结果，并最后在 Data Links 服务器上解决它们。当数据库管理程序确定某个故障很有可能会在 Data Links 服务器上创建不确定事务时，它将把该 Data Links 服务器的状态标记为需要崩溃恢复。当该 Data Links 服务器处于此状态时，数据库管理程序禁止任何 SQL 请求涉及此服务器。将原因码为 ″03″ 的 SQL0357N 返回给提出该 SQL 请求的应用程序。

当执行 RESTART、ACTIVATE DATABASE 或进行第一个 CONNECT 处理时，数据库管理程序会尝试与配置的每个 Data Links 服务器连接，并尝试通过异常终止或落实它们来解决不确定事务。若一个 Data Links 服务器的所有不确定事务都已解决，除了在数据库管理程序上也是未确定的那些事务外，则将该服务器的状态标记为可用的。在可用状态下，允许涉及此 Data Links 服务器的 SQL 请求。当解决不确定事务的尝试结束时，若数据库管理程序确定某个 Data Links服务器可能仍有需要解决的不确定事务，它将把该 Data Links 服务器的状态标记为需要崩溃恢复。例如，若在 RESTART、ACTIVATE DATABASE 或第一个 CONNECT 处理期间某个 Data Links 服务器不可用；或 Data Links 服务器在处理期间遇到故障，则可能出现这种情况。

当配置给某个数据库的 Data Links 服务器处于需要崩溃恢复的状态时，数据库管理程序将禁止 SQL 请求涉及该特定的 Data Links 服务器。仍然允许涉及该数据库中其他数据的 SQL 请求。数据库管理程序将启动一个进程，来以异步方式尝试在需要恢复的每个 Data Links 服务器上完成崩溃恢复。当该进程成功完成崩溃恢复时，Data Links 服务器的状态被标记为可用的，以允许其他 SQL 请求涉及到它。

备份实用程序注意事项

DB2 确保在备份操作完成时，也备份运行 DB2 Data Links Manager 的 Data Links 服务器上的链接文件。（备份实用程序可以联机或脱机运行，备份映象可以是数据

库的，也可以是表空间的。）后面的描述只适用于其 RECOVERY 参数被设置为 YES 的 DATALINK 列链接的文件。（指定了 RECOVERY=NO 的 DATALINK 列引用的文件不会被备份。）

当链接文件时，Data Links 服务器会安排以异步方式将这些文件复制到归档服务器（如 TSM）或磁盘上。当运行备份实用程序时，DB2 确保预定复制的所有文件都已被复制。在备份处理开始时，DB2 会联系在 DB2 配置文件可指定的所有 Data Links 服务器。如果对数据库配置了一个或多个 Data Links 服务器，即使 Data Links 服务器不可用，备份操作也会继续。Data Links 服务器重新启动时，备份处理将在 Data Links 服务器再次对数据库可用之前，在 Data Links 服务器上可用。如果 Data Links 服务器有一个或多个链接文件且不在运行，或在备份操作期间停止运行，备份映象将不会包含完整的 DATALINK 信息。备份操作将成功完成。该备份操作的历史记录文件中的注释字段将标识失败的 DLM 服务器，或在有多于四个 DLM 服务器失败时标识失败的 DLM 服务器的号码。一旦成功地备份了 DLM，注释字段将复位为它的先前值。

在将 Data Links 服务器标记为再次对数据库可用前，必须成功完成对所有主要备份的备份处理。（该操作由异步进程自动完成。）如果备份操作启动时，已经有两倍于 *num_db_backups* 值（参见以下部分）的主要备份在 Data Links 服务器上等待完成，该备份操作将失败。必须重新启动 Data Links 服务器，在其他备份之前完成的主要备份是允许的。

当断开一个文件时，会根据为 ON UNLINK 参数指定的值删除它或恢复它先前的许可权。成功的备份操作可使 Data Links 服务器清除归档服务器（磁盘或 TSM）上文件的归档版本；参见第42页的『垃圾收集』）。*num_db_backups* 数据库配置参数指定在除去文件（已解链）的归档版本前，DB2 数据库备份的数量。关于此配置参数的更多信息，请参阅*《管理指南: 性能》*一书。

当除去断开的文件时，也会从 Data Links 服务器注册表中除去有关那些断开文件的信息。

为 AIX 上的 DB2 Data Links Manager 选择备份方法

除了“磁盘复制”和 XBSA 外，还可以使用 Tivoli Storage Manager (TSM) 来备份驻留在 Data Links 服务器上的文件。

要将 Tivoli Storage Manager 作为归档服务器使用:

1. 在 Data Links 服务器上安装 Tivoli Storage Manager。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。
2. 向 Tivoli Storage Manager 服务器注册 Data Links 服务器客户机应用程序。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。

3. 向 Data Links Manager Administrator 的 db2profile 或 db2cshrc 脚本文件中添加以下环境变量:

```
（对于 Bash、Bourne 或 Korn shell）
export DSMI_DIR=/usr/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:$DSMI_DIR

（对于 C shell）
setenv DSMI_DIR /usr/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:$DSMI_DIR
```

4. 确保 dsm.sys TSM 系统选项文件位于 $DSMI_DIR 目录中。
5. 确保 dsm.opt TSM 用户选项文件位于 *INSTHOME*/tsm 目录中，其中 *INSTHOME* 是 Data Links Manager Administrator 的主目录。
6. 将 /usr/tivoli/tsm/client/api/bin/dsm.sys Tivoli Storage Manager 系统选项文件中的 *PASSWORDACCESS* 选项设置为 generate。
7. 在第一次启动 Data Links File Manager *之前*，向生成选项注册 TSM 密码。这样，在 Data Links File Manager 启动与 TSM 服务器的连接时就不需要提供密码。关于更多信息，请参阅 TSM 产品文档。
8. 将 DLFM_BACKUP_TARGET 注册表变量设置为 TSM。在这种情况中，将忽略 DLFM_BACKUP_DIR_NAME 注册表变量的值。从而可激活 Tivoli Storage Manager 备份选项。

   注:

   a. 如果在运行时更改 TSM 和磁盘间 DLFM_BACKUP_TARGET 注册表变量的值，您应了解归档文件是不会移动到新指定的归档位置中去的。例如，如果在 DLFM_BACKUP_TARGET 注册表值设置为 TSM 时启动 Data Links File Manager 并更改对应于磁盘位置的注册表值，则所有新归档的文件都将存储在磁盘上的新位置中。而先前归档到 TSM 中的文件将不会移动到新磁盘位置。

   b. 要覆盖缺省 TSM 管理类，可使用一个称为 DLFM_TSM_MGMTCLASS 的新注册表变量。如果此注册表变量保留为未设置，将使用缺省 TSM 管理类。

9. 通过输入 **dlfm stop** 命令，停止 Data Links File Manager。
10. 通过输入 **dlfm start** 命令，启动 Data Links File Manager。

为 Solaris 操作环境中的 DB2 Data Links Manager 选择备份方法

除了 “磁盘复制” 和 XBSA 外，还可以使用 Tivoli Storage Manager (TSM) 来备份驻留在 Data Links 服务器上的文件。

要将 Tivoli Storage Manager 作为归档服务器使用:

1. 在 Data Links 服务器上安装 Tivoli Storage Manager。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。
2. 向 Tivoli Storage Manager 服务器注册 Data Links 服务器客户机应用程序。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。
3. 向 Data Links Manager Administrator 的 db2profile 或 db2cshrc 脚本文件中添加以下环境变量:

```
（对于 Bash、Bourne 或 Korn shell）
export DSMI_DIR=/opt/tivoli/tsm/client/api/bin
export DSMI_CONFIG=$HOME/tsm/dsm.opt
export DSMI_LOG=$HOME/dldump
export PATH=$PATH:/opt/tivoli/tsm/client/api/bin

（对于 C shell）
setenv DSMI_DIR /opt/tivoli/tsm/client/api/bin
setenv DSMI_CONFIG ${HOME}/tsm/dsm.opt
setenv DSMI_LOG ${HOME}/dldump
setenv PATH=${PATH}:/opt/tivoli/tsm/client/api/bin
```

4. 确保 dsm.sys TSM 系统选项文件位于 /opt/tivoli/tsm/client/api/bin 目录中。
5. 确保 dsm.opt TSM 用户选项文件位于 *INSTHOME*/tsm 目录中，其中 *INSTHOME* 是 Data Links Manager Administrator 的主目录。
6. 将 /opt/tivoli/tsm/client/api/bin/dsm.sys Tivoli Storage Manager 系统选项文件中的 *PASSWORDACCESS* 选项设置为 generate。
7. 在第一次启动 Data Links File Manager *之前*，向生成选项注册 TSM 密码。这样，在 Data Links File Manager 启动与 TSM 服务器的连接时就不需要提供密码。关于更多信息，请参阅 TSM 产品文档。
8. 将 DLFM_BACKUP_TARGET 注册表变量设置为 TSM。在这种情况中，将忽略 DLFM_BACKUP_DIR_NAME 注册表变量的值。从而可激活 Tivoli Storage Manager 备份选项。

   注:

   a. 如果在运行时更改 TSM 和磁盘间 DLFM_BACKUP_TARGET 注册表变量的值，您应了解归档文件是不会移动到新指定的归档位置中去的。例如，如果在 DLFM_BACKUP_TARGET 注册表值设置为 TSM 时启动 Data Links File Manager 并更改对应于磁盘位置的注册表值，则所有新归档的文件都将存储在磁盘上的新位置中。而先前归档到 TSM 中的文件将不会移动到新磁盘位置。

   b. 要覆盖缺省 TSM 管理类，可使用一个称为 DLFM_TSM_MGMTCLASS 的新注册表变量。如果此注册表变量保留为未设置，将使用缺省 TSM 管理类。

9. 通过输入 **dlfm stop** 命令，停止 Data Links File Manager。
10. 通过输入 **dlfm start** 命令，启动 Data Links File Manager。

为 Windows NT 上的 DB2 Data Links Manager 选择备份方法

无论何时将为进行恢复操作而定义的 DATALINK 值插入具有 DATALINK 列的表中时，都会对 Data Links 服务器上相应的 DATALINK 文件进行调度，以备份到归档服务器上。目前，“磁盘备份”（缺省方法）和 Tivoli Storage Manager 是将文件备份到归档服务器的两种受支持的方法。DB2 Data Links Manager Windows NT 版的未来发行版将支持其他供应商的备份媒体和软件。

磁盘复制（缺省方法）

在对 DB2 服务器调用备份实用程序时，应确保数据库中链接的文件备份在 Data Links 服务器上、由 DLFM_BACKUP_DIR_NAME 环境变量指定的目录中。此变量的缺省值是 `c:\dlfmbackup`，其中 `c:\` 代表 Data Links Manager 备份安装驱动器。

要将此变量设置为 c:\dlfmbackup，输入以下命令:

```
db2set -g DLFM_BACKUP_DIR_NAME=c:\dlfmbackup
```

由 DLFM_BACKUP_DIR_NAME 环境变量指定的位置一定*不能*位于使用 Data Links Filesystem Filter 的文件系统上，且所需的空间在您为备份文件指定的目录中可用。

还要通过输入以下命令，确保 DLFM_BACKUP_TARGET 变量设置为 LOCAL:

```
db2set -g DLFM_BACKUP_TARGET=LOCAL
```

在设置或更改了这些变量后，可使用 **dlfm stop** 和 **dlfm start** 命令，停止并重新启动 Data Links File Manager。

Tivoli Storage Manager

要将 Tivoli Storage Manager 作为归档服务器使用:

1. 在 Data Links 服务器上安装 Tivoli Storage Manager。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。
2. 向 Tivoli Storage Manager 服务器注册 Data Links 服务器客户机应用程序。关于更多信息，请参阅 Tivoli Storage Manager 产品文档。
3. 单击**开始**，然后选择**设置 —> 控制面板 —> 系统**。“系统特性”窗口打开。选择**环境**选项卡，并输入以下环境变量和对应值:

变量	值
DSMI_DIR	c:\tsm\baclient

变量	值
DSMI_CONFIG	c:\tsm\baclient\dsm.opt
DSMI_LOG	c:\tsm\dldump

4. 确保 dsm.sys TSM 系统选项文件位于 c:\tsm\baclient 目录中。
5. 确保 dsm.opt TSM 用户选项文件位于 c:\tsm\baclient 目录中。
6. 将c:\tsm\baclient\dsm.sys Tivoli Storage Manager 系统选项文件中的 *PASSWORDACCESS* 选项设置为 generate。
7. 在第一次启动 Data Links File Manager *之前*，向生成选项注册 TSM 密码。这样，在 Data Links File Manager 启动与 TSM 服务器的连接时就不需要提供密码。关于更多信息，请参阅 TSM 产品文档。
8. 使用以下命令，将 DLFM_BACKUP_TARGET 环境变量设置为 TSM:

   ```
   db2set -g DLFM_BACKUP_TARGET=TSM
   ```

   在这种情况中，将忽略 DLFM_BACKUP_DIR_NAME 环境变量的值。从而可激活 Tivoli Storage Manager 备份选项。

   注:

   a. 如果在运行时更改 TSM 和 LOCAL 间 DLFM_BACKUP_TARGET 环境变量的值，您应了解归档文件是不会移动到新指定的归档位置中去的。例如，如果在 DLFM_BACKUP_TARGET 环境变量设置为 TSM 时启动 Data Links File Manager 并将它的值更改为 LOCAL，则所有新归档的文件都将存储在磁盘上的新位置中。而先前归档到 TSM 中的文件将不会移动到新磁盘位置。

   b. 要覆盖缺省 TSM 管理类，可使用一个称为 DLFM_TSM_MGMTCLASS 的新环境变量。如果此变量保留为未设置，将使用缺省 TSM 管理类。
9. 通过输入 **dlfm stop** 命令，停止 Data Links File Manager。
10. 通过输入 **dlfm start** 命令，启动 Data Links File Manager。

在 AIX 上备份日志文件系统

在停止了 Data Links Manager 后，可在 AIX 上执行日志文件系统的脱机备份。向那些要求有更高可用性的用户推荐以下方法，这些方法无需停止 Data Links Manager。

1. 访问 CLI 源文件 quiesce.c 和 shell 脚本 online.sh。这些文件位于 /samples/dlfm 目录中。
2. 编译 quiesce.c:

   ```
   xlC -o quiesce  -L$HOME/sqllib/lib -I$HOME/sqllib/include -c quiesce.c
   ```

3. 作为 root 用户，在具有 DLFS 文件系统的节点上运行脚本。

shell 脚本 online.sh 假设在 Data Link Manager 节点上，对每个向 Data Link Manager 进行了注册的数据库都有一个目录条目。还假设 /etc/filesystems 有表示 DLFS 文件系统的完整条目。shell 脚本执行以下操作:

- 使向 Data Links Manager 进行了注册的数据库中的所有表停顿。从而停止任何新的活动。
- 卸装并重新安装文件系统作为只读文件系统。
- 执行文件系统备份。
- 卸装并重新安装文件系统作为读 / 写文件系统。
- 复位 DB2 表，即结束它们的停顿状态。

必须修改脚本以适应您的环境，如下所示:

1. 选择备份命令并将它放到脚本的 do_backup 函数中。
2. 在脚本中设置以下环境变量:
   - DLFM_INST: 设置为 DLFM 实例名。
   - PATH_OF_EXEC: 设置为 ″quiesce″ 可执行文件所驻留的路径。

调用脚本，如下所示:

```
online.sh <filesystem_name>
```

复原和前滚实用程序注意事项

若有一个或多个 DATALINK 列，它们是用 RECOVERY=YES 选项为一个表定义的，则应用随后的信息。若一个表的某 DATALINK 列是使用 RECOVERY=NO 选项定义的，在复原操作结束时该表将处于 datalink 协调暂挂状态。要获取更多信息，参见第63页的『协调 Data Links』。

在复原操作期间，可能会将带有 DATALINK 列的表置于以下两种状态之一:

- *Datalink 协调不可能*

  当表处于 “datalink 协调不可能” 状态时，可对不是 DATALINK 列的列执行不受限操作。当 SELECT 语句中涉及到 DATALINK 列时，就会返回警告。可对 DATALINK 列发出 UPDATE 调用（也有一些限制: 要获取更多信息，参见第62页的『取消表的 “Datalink 协调不可能” 状态』）。不能发出 INSERT 和 DELETE 语句，因为它们涉及到 DATALINK 列。

- *Datalink 协调暂挂*

  当表处于 “datalink 协调暂挂” 状态时，可对不是 DATALINK 列的列执行不受限操作。当 SELECT 语句中涉及到 DATALINK 列时，就会返回警告。不能发出类似于 UPDATE、INSERT 或 DELETE 的任何 DML 语句。

当复原或前滚实用程序运行时，会在 db2diag.log 文件中报告这些状态。也可以使用 **db2dart** 命令来获得此信息。

复原数据库或表空间时，必须满足以下条件才能使复原操作成功:

- 如果记录在备份文件中的任何 Data Links Server 未在运行，复原操作仍将成功完成。

  受已丢失的 Data Links 服务器影响的、带有 DATALINK 列信息的表，将在复原操作（或前滚操作，如果使用的话）完成后，置于 “datalink 协调暂挂” 状态。在 Data Links 服务器可再次对数据库标记为可用之前，复原处理必须成功完成。一旦 DLM 可用即可完成备份处理的异步进程（参见第48页的『备份实用程序注意事项』）也会完成复原处理。
- 如果备份文件中记录的任何 Data Links Server 在复原操作期间停止运行，复原操作将失败。Data Links Server 当机时仍可重新启动复原操作（参见上述内容）。
- 如果 Data Links 服务器上仍有先前的数据库复原操作未完成，后继的数据库或表空间复原操作将失败，直到重新启动了那些 Data Links 服务器，且未完成的复原操作已完成。
- 在该备份文件中记录的关于所有 DATALINK 列的信息必须存在于适当的 Data Links 服务器的注册表中。

  若关于 DATALINK 列的所有信息未全部记录在注册表中，则在复原操作（或前滚操作，若使用的话）完成之后，丢失了 DATALINK 列信息的表会被置于 “datalink 协调不可能” 状态。

  若注册表中未记录该备份，可能意味着提供的备份文件先于 num_db_backups 的值生成，且执行了 “垃圾收集”。这表示根据这个较早的备份归档的文件已被除去，且不能复原。含有 DATALINK 列的所有表都处于 datalink 协调暂挂状态。

  如果注册表中未记录该备份，可能意味着备份处理由于 Data Links 服务器不在运行而尚未完成。含有 DATALINK 列的所有表都处于 datalink 协调暂挂状态。重新启动 Data Links 服务器后，备份处理将在复原处理前完成。

  该表继续可供用户使用，但是 DATALINK 列中的值可能无法准确地引用这些文件（例如，可能找不到与 DATALINK 列的值匹配的文件）。如果不希望出现此情况，可以发出 ″SET CONSTRAINTS for tablename TO DATALINK RECONCILE PENDING″ 语句，将表置于 “检查暂挂状态”。

若在复原操作之后有一个表处于 “datalink 协调不可能” 状态，可以使用第62页的『取消表的 “Datalink 协调不可能” 状态』下建议的方式之一修正 DATALINK 列数据。

注: 在将文件从断开状态标记为链接状态的过程中，该文件可能不得不从归档服务器检索至文件系统。若在此过程期间发生错误（例如，由于文件名重复而无法将文件复制到文件系统中），则对应的表会被置于 “datalink 协调暂挂” 状态。

由于数据库备份映象中的 datalink.cfg 文件只将 datalink.cfg 作为备份时间来反映，强烈建议归档 datalink.cfg 文件已备出现某些不正常的恢复情况。需要最新的 datalink.cfg 文件以涵盖所有的恢复情况。因此在每次 ADD Data Links MANAGER 或 DROP Data Links MANAGER 命令调用后，都必须备份 datalink.cfg 文件。最新的 datalink.cfg 文件不在磁盘上时，这将有助于检索最新的 datalink.cfg 文件。

如果磁盘上没有最新的 datalink.cfg 文件，可使用在运行前滚操作前归档的最新 datalink.cfg 文件替换现有的（从备份映象中复原的） datalink.cfg 文件。在复原数据库后执行此操作。

从脱机备份复原数据库而不前滚

只能在数据库级而不能在表空间级执行复原而不执行前滚。要复原数据库而不前滚，可以复原不可恢复的数据库（即，使用循环记录的数据库），或在 RESTORE DATABASE 命令上指定 WITHOUT ROLLING FORWARD 参数。

若在使用复原实用程序时指定 WITHOUT DATALINK 选项，则含 DATALINK 列的所有表都会被置于 “datalink 协调暂挂 (DRP)” 状态，且在复原操作期间不对 Data Links 服务器执行任何协调。

如果不使用 WITHOUT DATALINK 选项，且不再对数据库定义记录在备份文件中的 Data Links 服务器（即，已使用 DROP Data Links MANAGER 命令删除），则包含引用已删除的 Data Links 服务器的 DATALINK 数据的表由复原实用程序置于 DRP 状态。

若不使用 WITHOUT DATALINK 选项，且所有 Data Links 服务器都可用，所有关于 DATALINK 列的信息都完整地记录在注册表中，则记录在该备份文件中的每个 Data Links 会发生下列情况:

- 在数据库复原选项所用的备份映象之后链接的所有文件被标记为断开的（因为它们在该备份映象中未记录为已链接）。
- 在备份映象之后断开但在建立该备份映象之前链接的所有文件，会被标记为链接的（因为它们在该备份映象中记录为已链接）。若该文件后来与另一个数据库中的另一个表链接，则复原的表会被置于 “datalink 协调暂挂” 状态。

注: 如果用于数据库复原操作的备份映象是在至少有一个 Data Links 服务器不在运行时建立的，会由于该备份中的 DATALINK 信息不完整而使上述操作不能

完成。如果用于数据库复原操作的备份映象是在通过或不通过前滚恢复进行的数据库复原操作后建立的，则上述操作也无法完成。在这两种情况中，具有 DATALINK 列的所有表都将置于 datalink 协调暂挂状态，且在复原操作期间不通过 Data Links 服务器执行任何协调。

复原数据库和表空间，并前滚至日志末尾

如果复原，然后将数据库或表空间前滚至日志末尾（意味着提供了所有日志），则不需要执行协调检查，除非在复原操作期间至少有一个记录在备份文件中的 Data Links 服务器不在运行。如果不确定是否向前滚操作提供了所有日志，或认为您可能需要协调 DATALINK 值，可执行以下操作:

1. 对涉及的一个或多个表发出以下 SQL 语句:

   ```
   SET CONSTRAINTS FOR tablename TO DATALINK RECONCILE PENDING
   ```

   这会将表同时置于“datalink 协调暂挂”状态以及“检查暂挂”状态。

2. 如果不希望表处于“检查暂挂”状态，可发出以下 SQL 语句:

   ```
   SET CONSTRAINTS FOR tablename IMMEDIATE CHECKED
   ```

   这将使表结束其“检查暂挂”状态，但仍保留“datalink 协调暂挂”状态。必须使用协调实用程序以使该表脱离此状态。

如果备份文件包含了引用（即，在建立备份时，向数据库注册了 DB2 Data Links Manager）已从数据库删除的 DB2 Data Links Manager 的 DATALINK 数据，则可能会发生上述情况。对于要前滚的每个表空间（其中至少有一个表的 DATALINK 数据引用了已删除的 DB2 Data Links Manager），具有 DATALINK 列的所有表都将由前滚实用程序置于 DRP 状态。

复原数据库和表空间，并前滚至某个时间点

当使用 Data Links 表时，可前滚至日志末尾或指定的时间点。

在前滚操作结束时，表空间中前滚到某个时间点的表处于“datalink 协调暂挂” 状态。应该使用协调实用程序将它们从这种状态中解脱。要获取更多信息，参见第63页的『协调 Data Links』。

时间点前滚示例

以下是一个简单方案，它显示为处理备份和恢复需要保留的文件。该示例显示类型为 DATALINK 的列中单行值的更改，以及 DB2 Data Links Manager 为支持恢复而需要保留的文件。对于此示例，假定不支持将这些文件恢复至上一个备份之前的时间点。Data Links 服务器运行 DB2 Data Links Manager 时则没有这种限制。可以看到 fileA 一直存在，直到时间 3（即删除它的时间），原因是: 在时间 2 它被断开，而此示例中数据库的策略是将断开的文件保存至运行下一个备份

（即，将 *num_db_backups* 数据库配置参数设置为 1）时。

时间	1	2	3	4	5	6	7
活动	创建	更新	备份	更新	更新	删除	复原至 5
列值	valueA	valueB	valueB	valueC	valueD	-	valueD
链接文件	fileA	fileB	fileB	fileC	fileD	-	fileD
由 Data Links File Manager 保存的附加文件		fileA		fileB	fileB, fileC	fileB, fileC, fileD	fileB, fileC

注: 链接文件的恢复始终与数据库的其余部分一起执行。

DB2 Data Links Manager 和恢复交互作用

下表显示可以执行的不同类型的恢复、复原和前滚处理期间发生的 DB2 Data Links Manager 处理，以及是否需要在完成恢复之后调用协调实用程序:

恢复类型	复原期间的 DB2 Data Links Manager 处理	前滚期间的 DB2 Data Links Manager 处理	协调
不可恢复的数据库			
完整备份的数据库复原，所有 Data Links Server 已启动	执行快速协调	N/A	若怀疑文件链路有问题，可以选择运行
使用 WITHOUT DATALINK 选项的数据库复原	置于协调暂挂状态的表	N/A	必需的
对完整备份的数据库复原，至少一个 Data Links 服务器当机	只对那些（没有与已当机的 Data Links 服务器的链接的）表空间中的表执行快速协调，其他表则置于“datalink 协调暂挂”状态	NA	是（具有与已当机的 Data Links 的链接的）表空间中的表所需的

恢复类型	复原期间的 DB2 Data Links Manager 处理	前滚期间的 DB2 Data Links Manager 处理	协调
不完整备份的数据库复原，所有 Data Links 服务器已启动	不执行快速协调，具有 DATALINK 列的所有表都置于“datalink 协调暂挂”状态	NA	必需的
可恢复的数据库			
使用 WITHOUT ROLLING FORWARD 选项的数据库复原，使用完整的备份，所有 Data Links 服务器已启动	执行快速协调	N/A	可选的
使用 WITHOUT ROLLING FORWARD 和 WITHOUT DATALINK 选项的数据库复原，使用完整或不完整的备份，Data Links 服务器已启动或当机	表置于“datalink 协调暂挂”状态	N/A	必需的
使用 WITHOUT ROLLING FORWARD 选项的数据库复原，使用完整备份，至少有一个 Data Links 服务器当机	只对那些（没有与已当机的那些 Data Links 服务器的链接的）表空间中的表执行快速协调，其他表则置于 “datalink 协调暂挂”状态	N/A	是（具有与已当机的那些 Data Links 的链接的）表空间中的表所需的
使用 WITHOUT ROLLING FORWARD 选项的数据库复原，使用不完整的备份，Data Links 服务器已启动或当机	不执行快速协调，具有 DATALINK 列的所有表都置于“datalink 协调暂挂”状态	N/A	必需的

恢复类型	复原期间的 DB2 Data Links Manager 处理	前滚期间的 DB2 Data Links Manager 处理	协调
数据库复原以及前滚至日志末尾，使用完整备份，所有 Data Links 服务器已启动	无操作	无操作	可选的
数据库复原以及前滚至日志末尾，使用完整备份，前滚处理期间至少有一个 Data Links 服务器当机	无操作	无操作	可选的
数据库复原以及前滚及日志末尾，使用完整或不完整备份，复原期间有 Data Links 服务器当机	无操作	具有 DATALINK 列的所有表都置于 datalink 协调暂挂状态	是具有 DATALINK 列的所有表所需的
数据库复原以及前滚及日志末尾，使用不完整的备份，复原期间所有 Data Links 服务器都已启动	无操作	无操作	可选的
数据库复原以及前滚至日志末尾，使用完整或不完整的备份，所有 Data Links 服务器都已启动，备份在任何 Data Links 服务器上都不可识别	无操作	（具有与不识别备份的 Data Links 服务器的链接的）表空间中的所有表置于 datalink 协调暂挂状态	必需的
表空间复原以及前滚到日志末尾，使用完整备份，所有 Data Links 服务器已启动	无操作	无操作	可选的
表空间复原以及前滚至日志末尾，使用完整备份，至少有一个 Data Links 服务器在前滚处理期间当机	无操作	无操作	可选的

恢复类型	复原期间的 DB2 Data Links Manager 处理	前滚期间的 DB2 Data Links Manager 处理	协调
表空间复原以及前滚至日志末尾，使用完整或不完整备份，在复原处理期间有任何 Data Links 服务器当机	无操作	（具有与已当机的任何 Data Links 服务器的链接的）表空间中的所有表置于“datalink 协调暂挂”状态	是（具有与已当机的任何 Data Links 服务器的链接的）表空间中的表所需的
表空间复原以及前滚到日志末尾，使用不完整备份，所有 Data Links 服务器已启动	无操作	无操作	可选的
数据库复原以及前滚至某时间点，使用完整或不完整的备份，Data Links 服务器在复原和／或前滚处理期间已启动或当机	无操作	表置于“datalink 协调暂挂”状态	必需的
表空间复原以及前滚至某时间点，使用完整或不完整的备份，Data Links 服务器在复原和／或前滚处理期间已启动或当机	无操作	表置于“datalink 协调暂挂”状态	必需的
将数据库复原到另一个数据库名、别名、主机名或实例，而不进行前滚（参见 第62页的1 中的“注”）	表置于“datalink 协调不可能”状态	N/A	可选，但必须手工修正置于“datalink 协调不可能”状态的表
将数据库复原到另一个数据库名、别名、主机名或实例，并进行前滚	无操作	表置于“datalink 协调不可能”状态	可选，但必须手工修正置于“datalink 协调不可能”状态的表

恢复类型	复原期间的 DB2 Data Links Manager 处理	前滚期间的 DB2 Data Links Manager 处理	协调
使用或不使用 WITHOUT DATALINK，将数据库从不可用的备份（在 Data Links 服务器上对映象执行了垃圾收集）复原，且不进行前滚（参见 1 中的“注”）	表置于“datalink 协调暂挂”状态	无操作	必需的
使用或不使用 WITHOUT DATALINK 选项，将数据库从不可用的备份（在 Data Links 服务器上对映象执行了垃圾收集）复原，且进行前滚	无操作	表置于“datalink 协调暂挂”状态	必需的
将表空间从不可用的备份（在 Data Links 服务器上对映象执行了垃圾收集）复原，且进行前滚	无操作	表置于“datalink 协调暂挂”状态	必需的

注:

1. 使用联机备份映象和 WITHOUT ROLLING FORWARD 选项进行的复原操作，使用脱机备份映象进行的复原操作。
2. 完整部分是在所有需要的 Data Links 服务器正在运行时建立的备份。不完整的备份是在至少有一个所需的 Data Links 服务器不在运行时建立的。
3. 如果用于数据库复原操作的备份映象是在数据库复原后，使用或不使用前滚操作建立的，将不会执行快速协调处理。在这种情况下，具有 DATALINK 列的所有表都将置于“datalink 协调暂挂”状态。

取消表的“Datalink 协调不可能”状态

如果表空间是从早于对 *num_db_backups* 数据库配置参数指定的值的备份中复原的，则具有 DATALINK 列的已复原的表（或表空间）将置于“datalink 协调不可能”状态。关于此配置参数的更多信息，请参阅*《管理指南: 性能》*一书。

即使 DATALINK 列的值可能是无效的，DB2 仍允许访问表。若要阻止访问含有可能不一致的 DATALINK 列值的一个表，发出 SET CONSTRAINTS for *tablename* TO DATALINK RECONCILE PENDING 命令。可以更新 DATALINK 值，如下所示:

- 使用 SQL UPDATE 语句，若一个 DATALINK 列不可空，则将该列的数据位置部分设置为长度为零的 URL，否则若该列是可空的，则设置为 NULL。
- 在适当的 Data Links 服务器上复原这些文件。然后运行应用程序，以发出 SELECT 语句来读取 DATALINK 列的值，并发出 UPDATE 语句以使用相同的值更新 DATALINK 列。应注意，“datalink 协调不可能”状态必须在更新 DATALINK 列值时生效。在完成更新操作之后，在适当的 Data Links 服务器上这些文件被标记为已链接。

然后，通过发出以下命令，复位“datalink 协调不可能”状态:

```
SET CONSTRAINTS FOR tablename DATALINK RECONCILE PENDING IMMEDIATE UNCHECKED
```

协调 Data Links

可使用协调实用程序来协调 Data Links。该实用程序是从 DB2 启动的，它涉及 DATALINK 列值引用的、运行 DB2 Data Links Manager 的所有 Data Links 服务器。它验证引用的文件存在于 Data Links 服务器中，或者可以重新建立这些链路。下面几节描述 DB2 如何检测您是否需要协调 Data Links，及如何协调它们。

若一个 Data Links 服务器文件引用不存在或不能重新建立，协调实用程序将把出错行的副本及每个错误的原因一起放进一个异常表（若指定的话）中，然后修改出错的行。若未指定该异常表，将把无法为之重新建立文件引用的 DATALINK 列值连同一个列 ID 和原因复制到异常报告文件中。可以使用异常表信息（若指定的话）或报告来更新这些行，以进行必要的校正。在协调实用程序中使用的异常表与装入实用程序使用的异常表完全相同。关于装入实用程序的更多信息，请参阅 *Data Movement Utilities Guide and Reference*。该报告使用命名约定 *report*.exp（.exp 扩展名由协调实用程序提供）。例如，可使用以下语句调用协调实用程序:

```
db2 RECONCILE dept DLREPORT  /u/scottba/report FOR EXCEPTION excptab
```

此命令协调称为 dept 的表，并将异常情况写到异常表 excptab 中，该表是由用户创建的。将协调期间断开的文件的信息写入文件 report.ulk 中，该文件是在 /u/scottba 目录中创建的。若未指定 *FOR EXCEPTION excptab*，则将异常信息写入文件 report.exp，该文件是在 /u/scottba 目录中创建的。关于协调实用程序的更多信息，请参阅 *Command Reference*。

检测需要协调的情况

以下是可能需要运行协调实用程序的某些情况:

- 整个数据库被复原并前滚至一个时间点。因为整个数据库前滚至一个落实的事务，所以没有表将处于“检查暂挂”状态（由于参考约束或检查约束）。该数据库中的所有数据都会转入一致的状态。但是，DATALINK 列与 DB2 Data Links Manager 中的元数据可能不同步，因此需要进行协调。

  在这种情况下，含有 DATALINK 列数据的表已经处于 DRP 状态。应该对其中每一个表调用该协调实用程序。
- 运行 DB2 Data Links Manager 的某 Data Links 服务器失去对其元数据的跟踪。这可能是不同原因导致的。例如:
  - Data Links 服务器曾经冷启动过。
  - Data Links 服务器的元数据曾被复原至后备级状态。

  在某些情况下，如 SQL UPDATE 和 DELETE 期间，DB2 可能可以检测出 Data Links 服务器中元数据的问题。在这些情况中，SQL 语句将失败。应使用 SET CONSTRAINTS 语句将该表置于 DRP 状态，然后对该表运行协调实用程序。
- 文件系统不可用（例如，由于磁盘划伤），且未复原至当前状态。在这种情况下，文件可能丢失。
- DB2 Data Links Manager 从数据库删除，有 DATALINK FILE LINK CONTROL 值引用该 DB2 Data Links Manager。应该对这类表运行协调实用程序。

协调过程摘要

若由于时间点恢复，或由于运行 DB2 Data Links Manager 的 Data Links 服务器与 DB2 控制信息不匹配，而需要协调 Data Links:

1. 发出 SET CONSTRAINTS 语句，将该表置于“datalink 协调暂挂”状态。（在某些情况下，DB2 将为您执行此操作。）
2. 使用协调实用程序来处理这些链路，并对异常表或异常报告中的异常情况采取适当的操作。

第2章 数据库备份

本节描述了 DB2 UDB 备份实用程序，它用于创建数据库或表空间的备份副本。

包括下列主题:

备份概述

DB2 BACKUP DATABASE 命令的最简单的形式只需要您指定要备份的数据库的别名。例如:

```
db2 backup db sample
```

如果命令成功完成，您将获得一个新的备份映象，它位于发出命令的路径或目录中。它位于此目录中的原因是，此示例中的命令不会显式指定备份映象的目标位置。例如，在 Windows NT/2000 上，此命令（在从根目录发出此命令时）会创建一个出现在目录列表中的映象，如下所示:

```
 Directory of D:\SAMPLE.0\DB2\NODE0000\CATN0000\20010320

03/20/2001  12:26p      <DIR>          .
03/20/2001  12:26p      <DIR>          ..
03/20/2001  12:27p          12,615,680 122644.001
```

备份映象创建在您在调用备份实用程序时，用选项指定的目标位置中。位置可以是:

- 目录（对于备份到磁盘或软盘）
- 设备（对于备份到磁带）
- Tivoli Storage Manager (TSM) 服务器（参见第405页的『附录G. Tivoli Storage Manager』）
- 另一供应商服务器
- 通过用户出口程序指定（仅在 OS/2 上）。

无论何时调用一个完整的数据库备份操作，都会使用摘要信息自动更新恢复历史记录文件。此文件在数据库配置文件所在的目录中创建。关于恢复历史记录文件的更多信息，参见第41页的『了解恢复历史记录文件』。

在基于 UNIX 的系统上，在磁盘上创建的备份映象的文件名由连续几个元素组成，由句号分隔:

```
DB_alias.Type.Inst_name.NODEnnnn.CATNnnnn.timestamp.Seq_num
```

例如:

```
STAFF.0.DB201.NODE0000.CATN0000.19950922120112.001
```

在其他平台上，使用一个有 4 层的子目录树:

```
DB_alias.Type\Inst_name.NODEnnnn\CATNnnnn\yyyymmdd\hhmmss.Seq_num
```

例如 (Windows NT/2000):

```
SAMPLE.0\DB2\NODE0000\CATN0000\20010320\122644.001
```

数据库别名
在调用备份实用程序时指定的，由 1 至 8 个字符组成的数据库别名。

类型
备份操作的类型，其中: 0 表示完整的数据库级备份、3 表示表空间级的备份而 4 表示由 LOAD...COPY TO 命令生成的备份映象。

实例名
从 **DB2INSTANCE** 环境变量中提取的，由 1 至 8 个字符组成的当前实例名。

节点号
节点的编号。在非分区数据库系统上，始终为 NODE0000。在分区数据库系统上，它是 NODE*xxxx*，其中 *xxxx* 是分配给 db2nodes.cfg 文件中的节点的号码。

目录节点号
数据库的目录节点的节点号。在非分区数据库系

统中，始终为 CATN0000。在分区数据库系统中，它是 CATN*xxxx*，其中 *xxxx* 是分配给 db2nodes.cfg 文件中的节点的号码。

时间戳记

执行备份操作时的日期与时间的 14 个字符表示法。该时间戳记的格式为 *yyyymmddhhnnss*，其中:

- *yyyy* 表示年度（1995 至 9999）
- *mm* 表示月份（01 到 12）
- *dd* 表示某月中的某一天（01 到 31）
- *hh* 表示小时（00 到 23）
- *nn* 表示时间（00 到 59）
- *ss* 表示秒（00 到 59）

序列号

用作文件扩展名的一个 3 位的数字。

将备份映象写到磁带时:

- 不创建文件名，但以上描述的信息会存储在备份头中以在验证时使用。
- 必须有一个磁带机可通过标准操作系统接口来使用。在大分区数据库系统上，如果每个数据库分区服务器都有一个专用的磁带机可能不太实际。可以将这些磁带机与一个或多个 TSM 服务器连接，以便将对这些磁带机的访问权提供给每个数据库分区服务器。关于 TSM 的更多信息，参见第405页的『附录G. Tivoli Storage Manager』。
- 在一个分区数据库系统上，还可以使用提供虚拟磁带机功能的产品，如 REELlibrarian 4.2 或 CLIO/S。可使用这些产品来访问通过伪磁带机与其他节点（数据库分区服务器）连接的磁带机。对远程磁带机的访问权是透明地提供的，而伪磁带机可以通过标准的操作系统接口来访问。

使用备份所需的特权、权限和授权

特权使用户能够创建或访问数据库资源。权限级别提供将特权分组的方法，以及更高级的数据库和实用程序操作。这两者一起用于控制对数据库管理器和它的数据库对象的访问。用户只能访问那些他们具有适当授权（即必需的特权或权限）的对象。

必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限才能使用备份实用程序。

使用备份

使用备份前

不应连接到将要备份的数据库；备份实用程序自动建立与指定数据库的连接，而此连接会在备份操作完成时终止。

数据库可以是本地数据库或远程数据库。备份映象保留在数据库服务器上，除非您使用的是存储管理产品，如 Tivoli Storage Manager (TSM)。

在分区数据库系统上，将分别备份数据库分区。此操作对于您调用实用程序的数据库分区服务器是本地的。但您可以从实例中的一个数据库分区服务器发出 **db2_all**，以对由节点号标识的服务器列表调用备份实用程序。（使用 LIST NODES 命令来标识具有用户表的节点或数据库分区服务器。关于 LIST NODES 命令的信息，请参阅 *Command Reference*。）为此，必须先备份目录节点，然后，备份其他数据库分区。也可以使用“命令中心”来备份数据库分区。因为此方法不支持前滚恢复，应定期备份这些节点上的数据库。还应与您建立的任何备份副本一起，保留一份 `db2nodes.cfg` 文件，以保护可能对此文件造成的损坏。

在分布式请求系统上，备份操作适用于分布式请求数据库和存储在数据库目录（包装器、服务器和别名等等）中的元数据。不备份数据源对象（表和视图），除非它们也存储在分布式请求数据库中。

若一个数据库是使用数据库管理器的先前发行版创建的且该数据库还未迁移，则必须在将该数据库迁移之后，才能备份它。关于迁移数据库的信息，请参阅《*管理指南: 计划*》一书。

调用备份

备份实用程序可通过以下各项调用:

- 命令行处理器 (CLP)。

  以下是通过 CLP 发出的 BACKUP DATABASE 命令的示例:

  ```
  db2 backup database sample to c:\DB2Backups
  ```

- “控制中心”中的“备份数据库”笔记本或向导。要打开“备份数据库”笔记本或向导:

  1. 从“控制中心”中展开对象树，直到找到“数据库”文件夹。
  2. 单击“数据库”文件夹。任何现有的数据库都会显示在窗口右边的窗格（内容窗格）中。
  3. 在内容窗格中右键单击需要的数据库，然后从弹出菜单中选择“备份数据库”或“使用向导备份数据库”。“备份数据库”笔记本或“备份数据库”向导打开。

关于“控制中心”的通用信息，请参阅《*管理指南*》。通过“控制中心”中的联机帮助设施提供了详细信息。

- 应用程序编程接口 (API) **sqlubkp**。关于此 API 的信息，参见第75页的『备份数据库 API』。关于创建包含 DB2 管理 API 的应用程序的通用信息，请参阅《*应用程序构建指南*》。

显示备份信息

可通过 **db2ckbkp** 显示关于现有备份映象的信息。此实用程序允许您:

- 测试备份映象的完整性并确定是否可复原它。
- 显示存储在备份头中的信息。

关于此实用程序的详细信息，参见 第276页的『db2ckbkp - 检查备份』。

备份到磁带

备份数据库或表空间时，必须正确地设置块大小和缓冲区大小，尤其在使用变量块大小（例如在 AIX 上，如果块大小已设置为 0）时。

对备份时可使用的固定块大小的数量有限制。因为 DB2 将备份映象头写作 4 KB 的块，所以有此限制。DB2 支持的固定块大小只有 512、1024、2048 和 4096 字节。若使用固定块大小，则可以指定任何备份缓冲区大小。但是您可能会发现，如果固定块大小不是 DB2 所支持的大小之一，则备份操作将不能成功完成。

如果数据库很大，使用固定块大小意味着您的备份操作将花费很长时间。您可能要考虑使用变量块大小。

注: 使用变量块大小当前不受支持。如果必须使用此选项，应确保您有经过良好测试的过程，以使您（使用用变量块大小创建的备份映象）进行成功的恢复。

使用变量块大小时，必须指定备份缓冲区大小小于或等于正在使用的磁带设备的最大限制。要获得优化的性能，缓冲区大小必须等于正在使用的设备的最大块大小限制。

使用变量块大小从备份映象进行复原时可能会返回错误。若发生这种情况，可能需要使用适当的块大小重写该映象。以下是 AIX 上的一个示例:

```
tcl -b 0 -Bn -f /dev/rmt0 read > backup_filename.file
dd if=backup_filename.file of=/dev/rmt0/ obs=4096 conv=sync
```

备份映象转储到文件 backup_filenam.file 中。**dd** 命令使用块大小 4096，将映象转储到磁带上。

如果映象太大，不能转储到文件中，此方法可能会出现问题。有一种可能的解决方案是：使用 **dd** 命令将该映象从一个磁带机转储至另一磁带机。只要映象在一卷磁带上放得下，就可以使用这种方法。使用两个磁带机时，**dd** 命令是：

```
dd if=/dev/rmt1 of=/dev/rmt0 obs=4096
```

若不可能使用两台磁带机，可以使用 **dd** 命令将映象转储至原始设备，然后再将该映象从原始设备转储至磁带。使用此方法的问题在于 **dd** 命令*必须*跟踪转储至原始设备的块数。此块数必须在将映象移动回磁带时指定。如果使用 **dd** 命令将映象从原始设备转储至磁带，该命令会将原始设备的整个内容转储至磁带。**dd** 实用程序不能确定使用了多少原始设备空间来保留映象。

当使用备份实用程序时，您需要知道磁带机的最大块大小限制。以下是一些示例：

设备	连接	块大小限制	DB2 缓冲区大小限制（以 4 KB 页计）
8 毫米	scsi	131,072	32
3420	s370	65,536	16
3480	s370	65,536	16
3490	s370	65,536	16
3490E	s370	65,536	16
7332（4 毫米）[1]	scsi	262,144	64
3490e	scsi	262,144	64
3590[2]	scsi	2,097,152	512
3570 (magstar MP)		262,144	64

注：

1. 7332 没有块大小的限制。256 KB 只是一个建议的值。块大小限制由父适配器确定。
2. 在 3590 支持 2 MB 的块大小时，如果性能完全可以满足您的需要，就可以尝试使用较低的值（例如 256 KB）。
3. 关于设备限制的信息，请查看设备文档或向设备供应商咨询。

备份到命名管道

现在已支持将数据库备份到基于 UNIX 的系统上的本地命名管道，或从基于 UNIX 的系统上的本地命名管道复原数据库。命名管道的编写器与阅读器必须是同一台机器。管道必须存在并位于本地文件系统上。因为将命名管道视为本地设备，所以不需要指定目标是命名管道。以下是 AIX 上的一个示例:

1. 创建命名管道:

   ```
   mkfifo /u/dmcinnis/mypipe
   ```

2. 使用此管道作为数据库备份操作的目标:

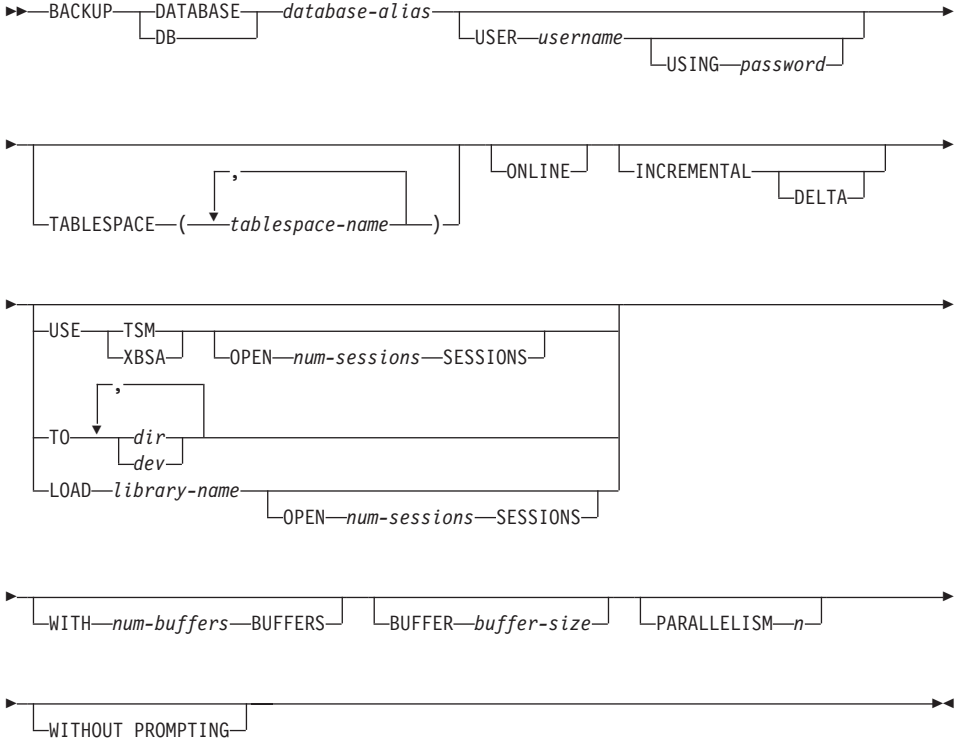
   ```
   db2 backup db sample to /u/dmcinnis/mypipe
   ```

3. 如果打算由复原实用程序来使用备份映象，则复原操作必须在备份操作*之前*调用，才不会丢失任何数据:

   ```
   db2 restore db sample into mynewdb from /u/dmcinnis/mypipe
   ```

BACKUP DATABASE 命令

命令语法



命令参数

DATABASE database-alias
指定要备份的数据库的别名。

USER username
标识备份数据库时要使用的用户名。

USING password
用于认证用户名的密码。如果省略了密码，将会提示用户输入。

TABLESPACE tablespace-name
用于指定要备份的表空间的名称列表。

ONLINE
指定联机备份。缺省值是脱机备份。联机备份只对配置为启用了 *logretain* 或 *userexit* 的数据库可用。

注: 因为 DB2 备份实用程序对包含 LOB 的对象需要 S 锁，如果 sysibm.systables 上有一个 IX 锁，则联机备份操作可能会时间不够。

INCREMENTAL

指定累积（增量）备份映象。增量备份映象是自从最新的成功完整备份操作以来，进行过更改的所有数据库数据的一份副本。

DELTA

指定非累积 (delta) 备份映象。delta 备份映象是自从任何类型的最新成功备份操作以来，更改过的所有数据库数据的一份副本。

USE TSM

指定备份将使用 Tivoli Storage Manager（以前称为 ADSM）输出。

OPEN num-sessions SESSIONS

要在 DB2 和 TSM 或另一备份供应商产品之间创建的 I/O 会话数。

注: 该参数在备份磁带、磁盘或其他本地设备时无效。

USE XBSA

指定将使用 XBSA 接口。 “备份服务 API” (XBSA) 是一个开放式的应用程序编程接口，供应用程序或设施需要数据存储管理进行备份或归档时使用。Legato NetWorker 是一个当前支持 XBSA 接口的存储管理器。

TO dir/dev

目录或磁带机名称列表。必须指定目录所驻留的完整路径。目标必须驻留在数据库服务器上。可重复此参数以指定备份映象将跨越的目标目录和设备。如果指定了多个目标（例如，target1、target2 和 target3），target1 将首先打开。媒体头和特殊文件（包括配置文件、表空间和历史记录文件）放在 target1 中。所有其他目标都是打开的，并在备份操作期间并行使用。由于对 OS/2 或 Windows 操作系统没有通用的磁带支持，所以每种类型的磁带机都需要一个唯一的设备驱动程序。要备份到 OS/2 或 Windows 操作系统的 FAT 文件系统上，用户必须符合 8.3 命名限制。

使用磁带机或软盘可能会生成需要用户进行输入的消息和提示。有效的响应选项是:

c 继续 - 继续使用生成了警告消息的设备（例如，已安装了新磁带时）

d 设备终止 - 只停止使用那个生成了警告消息（例如，没有磁带了）的设备

t 终止 - 异常终止备份操作。

在 OS/2 上不支持磁带。在 OS/2 上，可指定 0 或 0:，以使备份操作调用用户出口程序，（参见第411页的『附录H. 数据库恢复的用户出口』）。数据库在启动了用户出口程序的联机数据库备份之前停顿。备份实用程序会等待所有事务已落实或已回滚。在实用程序运行时，所有新的事务都将等待备份操作完成。

如果磁带系统不支持唯一地引用一个备份映象，建议不要将同一数据库的多个备份件副本保留在同一磁带上。

LOAD library-name

共享库的名称（OS/2 或 Windows 操作系统上的 DLL），包含要使用的供应商备份与复原 I/O 函数。也可以包含完整路径。如果未提供完整路径，将缺省为用户出口程序所驻留的那个路径。

WITH num-buffers BUFFERS

将要使用的缓冲区数量。缺省为 2。但是在将一个备份创建到多个位置时，具有较多的缓冲区有助于改进性能。

BUFFER buffer-size

以 4 KB 页计的缓冲区大小，在构建备份映象时使用。此参数的最小值是 8 页，缺省值是 1024 页。如果指定了缓冲区大小为 0，则 数据库管理器配置参数 *backbufsz* 的值将用作缓冲区分配大小。

如果使用具有变量块大小的磁带，将缓冲区大小减小到磁带机所支持的范围。否则，备份操作虽可以成功，但生成的映象可能是不可恢复的。

在 SCO UnixWare 7 上使用磁带机时，指定缓冲区大小为 16。

在 Linux 的大多数版本中，如果对向 SCSI 磁带机进行的备份操作使用 DB2 缺省缓冲区大小，会导致错误 SQL2025N，原因码 75。要防止 Linux 内部 SCSI 缓冲区溢出，使用此公式:

```
bufferpages <= ST_MAX_BUFFERS * ST_BUFFER_BLOCKS / 4
```

其中 *bufferpages* 是 *backbufsz* 或 *restbufsz* 的值，而 ST_MAX_BUFFERS 和 ST_BUFFER_BLOCKS 在 drivers/scsi 目录下的 Linux 内核中定义。

PARALLELISM n

确定可由备份实用程序并行读取的表空间的数量。缺省值为 1。

WITHOUT PROMPTING

指定备份将以无人照管方式运行，而通常需要用户干涉的任何操作都将返回一个错误消息。

备份数据库 API

C API 语法

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlubkp (
    char * pDbAlias,
    sqluint32 BufferSize,
    sqluint32     BackupMode,
    sqluint32     BackupType,
    sqluint32     CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32     NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

一般 API 语法

```
/* File: sqlutil.h */
/* API: Backup Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgbkp (
   unsigned short DbAliasLen,
   unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short * pReserved1,
    char * pDbAlias,
    sqluint32 BufferSize,
    sqluint32 BackupMode,
    sqluint32 BackupType,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    sqluint32 NumBuffers,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaTargetList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    sqluint32 * pBackupSize,
    void * pReserved4,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

API 参数

DbAliasLen

输入。一个 2 字节的无符号整数，代表以字节计的数据库别名的长度。

UserNameLen

输入。一个 2 字节的无符号整数，代表以字节计的用户名的长度。如果不提供用户名，则设置为 0。

PasswordLen

输入。一个 2 字节的无符号整数，代表以字节计的密码的长度。如果不提供密码，则设置为 0。

pReserved1.

保留以备将来使用。

pDbAlias

输入。一个包含要备份的数据库的数据库别名的字符串（如系统数据库目录中编目的那样）。

BufferSize

输入。以 4 KB 分配单位（页）计的备份缓冲区大小。最小值是 8 个单元。缺省值是 1024 个单元。

BackupMode

输入。指定备份方式。有效的值（在 sqlutil 中定义）是:

SQLUB_OFFLINE

脱机提供了与数据库的独占式连接。

SQLUB_ONLINE

联机使数据库在进行备份操作的过程中可由其他应用程序访问。

注: 如果 sysibm.systables 上有一个 IX 锁，则联机备份操作可能会超时，因为 DB2 备份实用程序需要 SMS LOB 对象上有 S 锁，而所有其他对象上有 IN 锁。

BackupType

输入。指定要建立的备份的类型。有效值（在 sqlutil 中定义）是:

SQLUB_FULL

指定完整（非增量）数据库备份操作。以后将不再支持此值。而是使用 SQLUB_DB 来指定完整数据库备份操作。

SQLUB_DB

指定数据库中所有表空间的备份。

SQLUB_TABLESPACE

指定表空间级的备份操作。在 *pTablespaceList* 参数中提供将要备份的表空间的列表。

SQLUB_INCREMENTAL

指定累积（增量）备份映象。增量备份映象是自从最新的成功完整备份操作以来更改过的所有数据库数据的副本。

SQLUB_DELTA

指定非累积 (delta) 备份映象。delta 备份映象是自从任何类型的最新成功备份操作以来更改过的所有数据库数据的副本。

CallerAction

输入。指定要采取的操作。有效的值（在 sqlutil 中定义）是:

SQLUB_BACKUP
启动备份操作。

SQLUB_NOINTERRUPT
启动备份操作。指定备份操作将以无人照管方式运行。将尝试那些通常需要用户干预的方案而不首先返回到调用程序，或者这些方案将产生错误。如果已知备份操作所需的所有媒体都已安装，且不希望出现实用程序提示时，使用此调用程序操作。

SQLUB_CONTINUE
在用户已执行了实用程序请求的某些操作后继续备份操作（例如在放入新磁带后继续）。

SQLUB_TERMINATE
在用户执行某些由实用程序请求的操作失败后，终止备份操作。

SQLUB_DEVICE_TERMINATE
从由备份实用程序使用的设备列表中除去某个特定的设备。当某个特定的媒体已满时，备份实用程序会返回一个警告给调用程序（在使用其他设备继续处理时）。使用此调用程序操作再次调用备份实用程序，以从正使用的设备列表中除去生成警告的设备。

SQLUB_PARM_CHECK
用于验证参数而不执行备份操作。此选项在返回调用后不终止数据库连接。在成功返回此调用后，需要用户发出指定了 SQLUB_CONTINUE 的调用以使操作继续。

SQLUB_PARM_CHECK_ONLY
用于验证参数而不执行备份操作。在此调用返回前，将终止由此调用建立的数据库连接，且不需要后续调用。

pApplicationId
输出。提供缓冲区长度 SQLU_APPLID_LEN+1（在 sqlutil 中定义）。此 API 将返回一个标识为应用程序提供服务的代理程序的字符串。可用于通过数据库监视器获取关于备份操作进度的信息。

pTimestamp
输出。提供缓冲区长度 SQLU_TIME_STAMP_LEN+1（在 sqlutil 中定义）。该 API 将返回备份映象的时间戳记。

NumBuffers
输入。指定要使用的备份缓冲区数。

pTablespaceList
输入。列出要备份的表空间。只有表空间级的备份操作需要该参数。参见第87页的『数据结构: SQLU-TABLESPACE-BKRST-LIST』。

pMediaTargetList

输入。此结构允许调用程序指定备份操作的目标。所提供的信息取决于 *media_type* 字段的值。*media_type* 的有效值（在 sqlutil 中定义）是:

SQLU_LOCAL_MEDIA

本地设备（磁带、磁盘或软盘的组合）。提供一个 *sqlu_media_entry* 结构的列表。在 OS/2 或 Windows 操作系统上，只可输入目录路径，而不能是磁带机的名称。

SQLU_TSM_MEDIA

TSM. 如果未使用 *sqlu_media_entry* 结构来指定备份映象的路径，应将 *sqlu_media_list_targets* 结构中的 *media* 指针初始化为 NULL。使用 DB2 附带提供的 TSM 共享库。如果希望使用 TSM 共享库的另一版本，可使用 SQLU_OTHER_MEDIA 并提供共享库名。

SQLU_OTHER_MEDIA

供应商产品。提供 *sqlu_vendor* 结构中的共享库的名称。

SQLU_USER_EXIT

用户出口。不需要其他任何输入（只在 OS/2 上可用）。

参见第83页的『数据结构: SQLU-MEDIA-LIST』。

pUserName

输入。包含尝试连接时将使用的用户名的字符串。

pPassword

输入。包含将与用户名一起使用的密码的字符串。

pReserved2

保留以备将来使用。

VendorOptionsSize

输入。*pVendorOptions* 字段的长度不能超出 65535 字节。

pVendorOptions

输入。用于将信息从应用程序发送给供应商函数。此数据结构必须是平面的，因此不支持间接级别。应注意字节转换未完成，且未检查此数据的代码页。

Parallelism

输入。并行性的程度（缓冲区操纵器的数量）。

pBackupSize

输出。备份映象的大小（以 MB 计）。可设置为 NULL。

pReserved4
保留以备将来使用。

pReserved3
保留以备将来使用。

pSqlca
输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

REXX API 语法

```
BACKUP DATABASE dbalias USING :value [USER username USING password]

[TABLESPACE :tablespacenames] [ONLINE]

[LOAD vendor-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 TO :target-area |
 USE TSM [OPEN num-sessions SESSIONS] |
 USER_EXIT]

[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[PARALLELISM parallelism-degree]
```

REXX API 参数

dbalias
要备份的数据库的别名。

value 组合 REXX 主机变量，对它返回数据库备份信息。在以下示例中，XXX 代表主机变量名:

XXX.0 变量中的元素号（始终为 2）

XXX.1 备份映象的时间戳记

XXX.2 标识为应用程序提供服务的代理程序的应用程序标识。

username
标识备份数据库时要使用的用户名。

password
用于认证用户名的密码。

tablespacenames
组合 REXX 主机变量，包含将备份的表空间的列表。在以下示例中 XXX 是主机变量的名称:

XXX.0 要备份的表空间号

XXX.1 第一个表空间的名称

XXX.2 第二个表空间的名称

XXX.3 以此类推。

vendor-library

共享库（Windows 操作系统或 OS/2 上的 DLL）的名称，它包含将使用的供应商备份与复原 I/O 函数。它可包含全路径。如果提供全路径，将缺省为用户出口程序所驻留的路径。

vendor-options

供应商函数所需要的信息。

num-sessions

将与 TSM 或供应商产品一起使用的 I/O 会话数。

target-area

本地设备。始终为磁带、磁盘或软盘的组合。在第83页的『数据结构: SQLU-MEDIA-LIST』中提供了一个列表。在 OS/2 或 Windows 操作系统上，只可输入目录路径，而不能是磁带机的名称。

caller-action

指定要采取的操作。有效的值为:

SQLUB_BACKUP

启动备份操作。

SQLUB_NOINTERRUPT

启动备份操作。指定备份操作将以无人照管方式运行。将尝试那些通常需要用户干预的方案而不首先返回到调用程序，或者这些方案将产生错误。如果已知备份操作所需的所有媒体都已安装，且不希望出现实用程序提示时，使用此调用程序操作。

SQLUB_CONTINUE

在用户已执行了实用程序请求的某些操作后继续备份操作（例如在放入新磁带后继续）。

SQLUB_TERMINATE

在用户执行某些由实用程序请求的操作失败后，终止备份操作。

SQLUB_DEVICE_TERMINATE

从由备份实用程序使用的设备列表中除去某个特定的设备。当某个特定的媒体已满时，备份实用程序会返回一个警告给调用程序（在使用其他设备继续处理时）。使用此调用程序操作再次调用备份实用程序，以从正使用的设备列表中除去生成警告的设备。

SQLUB_PARM_CHECK
用于验证参数而不执行备份操作。

num-buffers
将要使用的备份缓冲区数。

buffer-size
以 4 KB 分配单位计的备份缓冲区大小。最小值是 8 个单元。

parallelism-degree
并行性的程度（缓冲区操纵器的数量）。

数据结构: SQLU-MEDIA-LIST

此结构用于:

- 为备份映象保留一个*目标*媒体列表（参见第75页的『备份数据库 API』）。
- 为备份映象保留一个*源*媒体列表（参见第101页的『复原数据库 API』）。
- 将信息发送到 DB2 装入实用程序。

表 1. SQLU-MEDIA-LIST 结构中的字段

字段名称	数据类型	描述
MEDIA_TYPE	CHAR(1)	一个表示媒体类型的字符。
SESSIONS	INTEGER	表示由此结构的*目标*字段所指向的数组中的元素数。
TARGET	Union	此字段是指向三种结构类型之一的指针。所指向的结构类型由*媒体类型*字段的值确定。关于在此字段中提供的值的更多信息，参见相应的 API。

表 2. SQLU-MEDIA-LIST-TARGETS 结构中的字段

字段名称	数据类型	描述
MEDIA	Pointer	指向 *sqlu_media_entry* 结构的指针。
VENDOR	Pointer	指向 *sqlu_vendor* 结构的指针。
LOCATION	Pointer	指向 *sqlu_location_entry* 结构的指针。

表 3. SQLU-MEDIA-ENTRY 结构中的字段

字段名称	数据类型	描述
RESERVE_LEN	INTEGER	*media_entry* 字段的长度。用于 C 以外的其他语言。
MEDIA_ENTRY	CHAR(215)	备份与复原实用程序使用的备份映象的路径。

表 4. SQLU-VENDOR 结构中的字段

字段名称	数据类型	描述
RESERVE_LEN1	INTEGER	*shr_lib* 字段的长度。用于 C 以外的其他语言。
SHR_LIB	CHAR(255)	供应商提供的、用于存储和检索数据的共享库的名称。
RESERVE_LEN2	INTEGER	*filename* 字段的长度。用于 C 以外的其他语言。
FILENAME	CHAR(255)	使用共享库时，标识装入输入源的文件名。

表 5. SQLU-LOCATION-ENTRY 结构中的字段

字段名称	数据类型	描述
RESERVE_LEN	INTEGER	*location_entry* 字段的长度。用于 C 以外的其他语言。
LOCATION_ENTRY	CHAR(256)	用于装入实用程序的输入数据文件的名称。

MEDIA_TYPE 的有效值（在 `sqlutil` 中定义）是:

SQLU_LOCAL_MEDIA
本地设备（磁带、磁盘或软盘）

SQLU_SERVER_LOCATION
服务器设备（磁带、磁盘或软盘; 仅用于装入）。只能对 *pDataFileList* 参数指定。

SQLU_TSM_MEDIA
TSM

SQLU_OTHER_MEDIA
供应商库

SQLU_USER_EXIT
用户出口（仅用于 OS/2）

SQLU_PIPE_MEDIA
命名管道（仅用于供应商 API）

SQLU_DISK_MEDIA
磁盘（仅用于供应商 API）

SQLU_DISKETTE_MEDIA
软盘（仅用于供应商 API）

SQLU_TAPE_MEDIA
磁带（仅用于供应商 API）

语言语法

C 结构

```
/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_media_list
{
  char             media_type;
  char             filler[3];
  sqlint32             sessions;
  union sqlu_media_list_targets target;
} sqlu_media_list;
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-LIST-TARGETS */
/* ... */
union sqlu_media_list_targets
{
  struct sqlu_media_entry        *media;
  struct sqlu_vendor             *vendor;
  struct sqlu_location_entry     *location;
};
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-MEDIA-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_media_entry
{
  sqluint32       reserve_len;
  char            media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-VENDOR */
/* ... */
typedef SQL_STRUCTURE sqlu_vendor
{
  sqluint32       reserve_len1;
  char            shr_lib[SQLU_SHR_LIB_LEN+1];
  sqluint32       reserve_len2;
  char            filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLU-LOCATION-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_location_entry
{
  sqluint32       reserve_len;
  char            location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;
/* ... */
```

COBOL 结构

```
* File: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*


* File: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*
```

数据结构: SQLU-TABLESPACE-BKRST-LIST

此结构用于提供表空间名称的列表。

表 6. SQLU-TABLESPACE-BKRST-LIST 结构中的字段

字段名称	数据类型	描述
NUM_ENTRY	INTEGER	列表中由*tablespace* 字段所指向的条目数。
TABLESPACE	Pointer	指向 *sqlu_tablespace_entry* 结构的指针。

表 7. SQLU-TABLESPACE-ENTRY 结构中的字段

字段名称	数据类型	描述
RESERVE_LEN	INTEGER	*tablespace_entry* 字段中提供的字符串的长度。用于 C 以外的其他语言。
TABLESPACE_ENTRY	CHAR(19)	表空间的名称。

语言语法

C 结构

```
/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-BKRST-LIST */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
  long            num_entry;
  struct sqlu_tablespace_entry *tablespace;
} sqlu_tablespace_bkrst_list;
/* ... */


/* File: sqlutil.h */
/* Structure: SQLU-TABLESPACE-ENTRY */
/* ... */
typedef SQL_STRUCTURE sqlu_tablespace_entry
{
  sqluint32       reserve_len;
  char            tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
  char            filler[1];
} sqlu_tablespace_entry;
/* ... */
```

COBOL 结构

数据结构: SQLU-TABLESPACE-BKRST-LIST

```
* File: sqlutil.cbl
01 SQLU-TABLESPACE-BKRST-LIST.
    05 SQL-NUM-ENTRY          PIC S9(9) COMP-5.
    05 SQL-TABLESPACE         USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQLU-TABLESPACE-ENTRY.
    05 SQL-TBSP-LEN           PIC 9(9) COMP-5.
    05 SQL-TABLESPACE-ENTRY   PIC X(18).
    05 FILLER                 PIC X.
    05 SQL-FILLER             PIC X(1).
*
```

示例备份会话

CLP 示例

```
db2 backup database sample use tsm open 2 sessions with 4 buffers

db2 backup database payroll tablespace syscatspace, userspace1 to
   /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

以下是对可恢复数据库每周进行的增量备份策略的样本。包括一个每周进行的完整数据库备份操作、一个每天进行的非累积 (delta) 备份操作以及一个在每周中期进行的累积（增量）备份操作:

```
(Sun) db2 backup db kdr use tsm
(Mon) db2 backup db kdr online incremental delta use tsm
(Tue) db2 backup db kdr online incremental delta use tsm
(Wed) db2 backup db kdr online incremental use tsm
(Thu) db2 backup db kdr online incremental delta use tsm
(Fri) db2 backup db kdr online incremental delta use tsm
(Sat) db2 backup db kdr online incremental use tsm
```

在第397页的『附录F. 恢复 CLP 脚本』中提供了样本 DB2 命令脚本以及如何使用它的信息。

API 示例

在第331页的『附录E. 恢复样本程序』中提供了包含 DB2 API 和嵌入式 SQL 调用的样本程序，以及如何使用它们的信息。

优化备份性能

要减少完成备份操作所需的时间量:

- 指定表空间备份。

  对 BACKUP DATABASE 命令使用选项 TABLESPACE，可以备份（继而恢复）部分数据库。这样便于对表数据、索引和单独表空间中的长字段或大对象 (LOB) 数据进行管理。

- 增大 BACKUP DATABASE 命令上 PARALLELISM 参数的值，以使它反映正在备份的表空间数。

  PARALLELISM 参数定义在从数据库读取数据时已启动的进程或线程数。每个进程或线程都指定给一个特定的表空间。备份完此表空间后，它会请求另一个表空间。但是应注意: 每个进程或线程都需要内存和 CPU 开销，对于负荷较重的系统，应该使 PARALLELISM 参数保持为缺省值 1。

- 增加备份缓冲区大小。

理想的备份缓冲区大小是表空间范围大小的倍数。若有多个具有不同范围大小的表空间，则指定是最大范围大小的倍数的一个值。

- 增加缓冲区的数量。

  若使用多个缓冲区和 I/O 通道，应该使用的通道数至少是缓冲区数的两倍，以确保通道不必等待数据。
- 使用多个目标设备。

备份限制

以下限制适用于备份实用程序:

- 表空间备份操作和表空间复原操作不能同时运行，即使涉及的是不同的表空间。
- 如果希望能够在分区数据库环境中执行前滚恢复，必须在节点列表中的节点上定期备份数据库，并且必须有系统中其余节点上的数据库的至少一份备份映象（即使备份映象中不包含那个数据库的用户数据）。下列两种情况都需要在不包含数据库的用户数据的数据库分区服务器中存在数据库分区的备份映象:
  - 在建立上一个备份之后已将一个数据库分区服务器添加到数据库系统，因此需要在此数据库分区服务器上执行正向恢复。
  - 使用时间点恢复，它要求系统中的所有数据库分区都处于前滚暂挂状态。

故障诊断备份

不能备份处于不可用状态的数据库，除非该数据库处于备份暂挂状态。如果有任何表空间处于异常状态，则不能备份该数据库或该表空间，除非它处于备份暂挂状态。

如果数据库或表空间由于在执行复原操作期间发生了系统崩溃而处于部分复原状态，必须成功地复原该数据库或表空间才能对它们进行备份。

如果要备份的表空间的列表包含了临时表空间的名称，备份操作会失败。

备份实用程序可对建立不同数据库的备份副本的多个进程提供并行控制。此并行控制可使备份目标设备保持打开，直到所有备份操作结束。如果在备份操作期间发生错误，有一个打开的容器不能关闭，则其他备份操作的相同目标驱动器可能会接收到访问错误。要校正这类访问错误，必须终止导致了该错误的备份操作，并从目标设备断开。如果正在对并行的备份到磁带的操作使用备份实用程序，应确保这些进程的目标驱动器不是同一磁带。

第3章 数据库复原

这部分描述了 DB2 UDB 复原实用程序，它用于重新构建先前进行了备份的已受损的或毁坏的数据库或表空间。

包括下列主题:

复原概述

DB2 RESTORE DATABASE 命令的最简单的格式只需要您指定要复原的数据库的别名。例如:

```
db2 restore db sample
```

在此示例中，因为 SAMPLE 数据库已存在，将返回以下消息:

```
SQL2539W 警告！正在复原到一个与备份映象数据库相同的现有数据库。将删除数据库文件。
您想继续吗？(y/n)
```

如果指定 y，而 SAMPLE 数据库的备份映象已存在，则复原操作应成功完成。

数据库复原操作需要一个独占连接: 即，启动该任务后，复原实用程序会防止其他应用程序访问数据库，直到复原操作成功完成，所以不能再对该数据库运行任何应用程序。但表空间复原可以联机完成。

表空间将不可用直到复原操作（后跟前滚恢复）成功完成。

如果有跨越多个表空间的表，则应该一起备份并复原这个表空间集合。

执行部分或子集复原操作时，可以使用表空间级的备份映象或完整数据库级的备份映象，并从该映象中选择一个或多个表空间。从建立备份映象时开始的所有与这些表空间相关的日志文件必须存在。

使用复原所需的特权、权限和授权

特权使用户能够创建或访问数据库资源。权限级别提供将特权分组的方法，以及更高级的数据库和实用程序操作。这两者一起用于控制对数据库管理器和它的数据库对象的访问。用户只能访问那些他们具有适当授权（即必需的特权或权限）的对象。

要从完整的数据库备份复原至*现有*数据库，必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限。要复原至*新*数据库，必须具有 SYSADM 或 SYSCTRL 权限。

使用复原

使用复原前

复原至*现有*数据库时，不应连接至要复原的数据库：复原实用程序自动建立与特定数据库的连接，并且此连接在复原操作完成时终止。复原到*新*数据库时，需要实例连接才能创建数据库。复原到*新远程*数据库时，必须首先连接至新数据库将驻留的实例。然后创建新数据库，指定服务器的代码页和地区。

数据库可以是本地的也可以是远程的。

调用复原

复原实用程序可通过以下各项调用:

- 命令行处理器 (CLP)。

  以下是通过 CLP 发出的 RESTORE DATABASE 命令的示例:

  ```
  db2 restore db sample from D:\DB2Backups taken at 20010320122644
  ```

- “控制中心”中的“复原数据库”笔记本或向导。要打开“复原数据库”笔记本或向导:

  1. 从“控制中心”中，展开对象树，直到找到“数据库”文件夹。
  2. 单击“数据库”文件夹。所有现有的数据库都显示在该窗口右边的窗格（内容窗格）中。

3. 在内容窗格中，用鼠标右按钮单击需要的数据库，然后从弹出菜单中选择“复原数据库”或“使用向导复原数据库”。“复原数据库”笔记本或“复原数据库”向导打开。

关于“控制中心”的通用信息，请参阅《*管理指南*》。在“控制中心”中通过联机帮助设施提供了详细信息。

- 应用程序编程接口 (API)，**sqlrestore**。关于此 API 的信息，参见第101页的『复原数据库 API』。关于创建包含 DB2 管理 API 的应用程序的通用信息，请参阅《*应用程序构建指南*》。

在复原操作期间重新定义表空间容器（重定向复原）

在数据库备份操作期间，保留了一个记录，它记录了与正在备份的表空间相关的所有表空间容器。在复原操作期间，会检查备份映象中列示的所有容器，以确定它们是否存在并可访问。若这些容器中的一个或多个由于媒体故障（或由于任何其他原因）而不可访问，复原操作将失败。在这种情况下，复原操作需要重定向至不同的容器。DB2 支持添加、更改或除去表空间容器。

通过调用 RESTORE DATABASE 命令并指定 REDIRECT 参数，或通过使用“控制中心”中的“复原数据库”笔记本的“容器”页，可重新定义表空间容器。关于使用命令行处理器、命令脚本或应用程序编程接口的重定向复原示例，参见第111页的『示例复原会话』。

在重定向复原操作期间，如果目录和文件容器不存在，将自动创建。数据库管理程序不会自动创建设备容器。

容器重定向为管理表空间容器提供了相当大的灵活性。例如，即使不支持向 SMS 表空间添加容器，您也可以通过在调用重定向复原操作时指定其他容器来达到此目的。同样的，可以将 DMS 表空间从文件容器移至设备容器。

复原至现存的数据库

可以将一个完整的数据库备份映象复原至现有的数据库。备份映象可能在别名、数据库名或数据库起始值等方面与现存的数据库有所不同。

数据库*种子值*是数据库的唯一标识符，它在该数据库的整个生命期永不更改。种子值是在创建数据库时由数据库管理器指定的。它在复原操作后保持不变，即使备份映象有一个不同的数据库种子值。DB2 始终使用备份映象中的种子值。

复原至现有的数据库时，复原程序会:

复原至现有的数据库

- 删除现有数据库中的表、索引和长字段数据，并用备份映象中的数据来替换它们。
- 替换表示要复原的每个表空间的表条目。
- 保留恢复历史记录文件，除非它已损坏。如果恢复历史记录文件已损坏，数据库管理器会从备份映象中复制文件。
- 保留现有数据库的认证类型。
- 保留现有数据库的数据库目录。该目录定义了数据库的驻留位置与编目方式。
- 比较数据库种子值。如果种子值不同:
  - 删除与现有的数据库相关的日志。
  - 从备份映象中复制数据库配置文件。
  - 如果对 RESTORE DATABASE 命令指定了 NEWLOGPATH，应将 NEWLOGPATH 设置为 *logpath* 数据库配置参数的值。

  如果数据库种子值相同:
  - 如果映象是不可复原的数据库的，则删除日志。
  - 保留当前的数据库配置文件，除非该文件已毁坏，在这种情况下将从备份映象中复制此文件。
  - 如果对 RESTORE DATABASE 命令指定了 NEWLOGPATH，应将 NEWLOGPATH 设置为 *logpath* 数据库配置参数的值；否则应将当前日志路径复制到数据库配置文件。验证日志路径：如果该路径不能由数据库使用，应更改数据库配置以使用缺省日志路径。

复原至新的数据库

可以创建一个新数据库并向它复原完整的数据库备份映象。备份映象的代码页和目标数据库必须匹配。

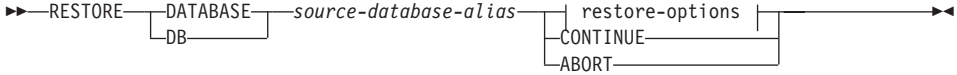
复原到新数据库时，复原实用程序:

- 使用通过目标数据库别名参数指定的数据库别名创建新数据库。（如果未指定目标数据库别名，复原实用程序会使用通过源数据库别名参数指定的相同别名来创建数据库。）
- 从备份映象复原数据库配置文件。
- 如果在 RESTORE DATABASE 命令上指定了 NEWLOGPATH，应将 NEWLOGPATH 设置为 *logpath* 数据库配置参数的值。验证日志路径：如果路径不能由数据库使用，则应更改数据库配置以使用缺省日志路径。
- 从备份映象复原认证类型。
- 从备份映象中的数据库目录复原注释。

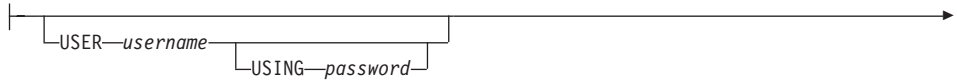
- 复原数据库的恢复历史记录文件。

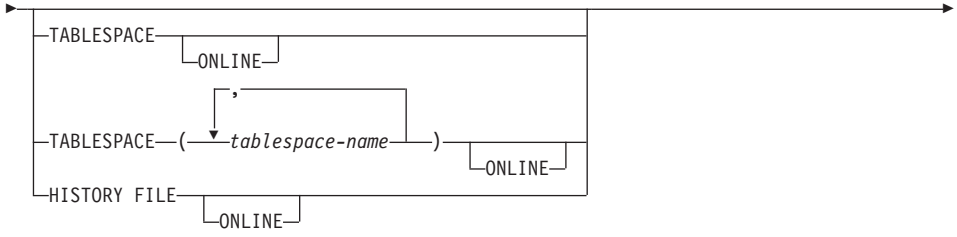
RESTORE DATABASE 命令

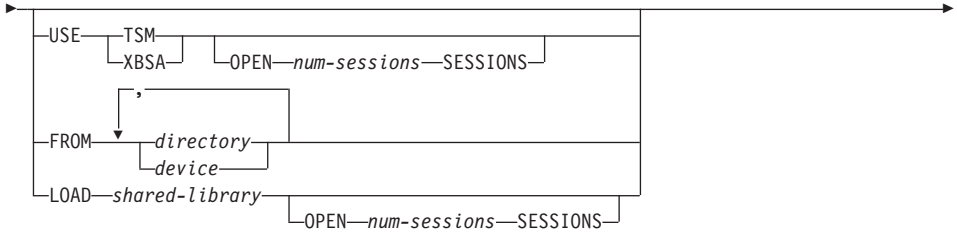
命令语法

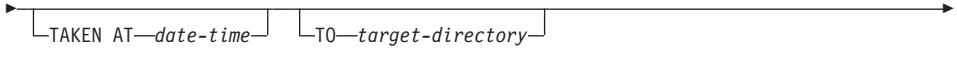


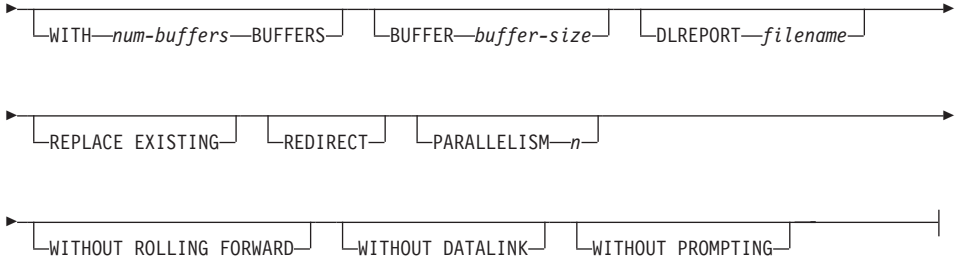
restore-options:











命令参数

DATABASE source-database-alias
从其建立备份的源数据库的别名。

CONTINUE
指定已重新定义了容器，应执行重定向复原操作中的最后一步。

ABORT
此参数:

- 停止重定向复原操作。当发生需要重复一个或多个步骤的错误时此参数很有用。发出了带 ABORT 选项的 RESTORE DATABASE 后，必须重复重定向复原操作的每一步，包括带有 REDIRECT 选项的 RESTORE DATABASE。
- 在增量复原操作完成前终止它。

USER username
标识复原数据库时要使用的用户名。

USING password
用于认证用户名的密码。如果省略密码，会提示用户输入。

TABLESPACE tablespace-name
用于指定要复原的表空间的名称列表。

ONLINE
此关键字只可在执行表空间级的复原操作时应用，指定该关键字使备份映象可联机复原。这意味着在对备份映象进行复原时其他代理程序可连接至数据库，而复原指定的表空间时其他表空间中的数据将是可用的。

HISTORY FILE
指定此关键字，以从备份映象中只复原历史记录。

INCREMENTAL
指定手工累积复原操作。用户必须手工发出每个复原命令。

AUTOMATIC/AUTO

指定自动累积（增量）复原操作。

USE TSM

指定将从 TSM 管理的输出复原数据库。

OPEN num-sessions SESSIONS

指定将与 TSM 或供应商产品一起使用的 I/O 会话数。

USE XBSA

指定将使用的 XBSA 接口。“备份服务 API”(XBSA) 是一个开放的应用程序编程接口，由需要数据存储管理的应用程序或设施进行备份或归档时使用。Legato NetWorker 是当前支持 XBSA 接口的一种存储管理器。

FROM directory/device

备份映象所驻留的目录或设备。如果省略 USE TSM、FROM 和 LOAD，缺省值是当前目录。

在 Windows 操作系统或 OS/2 上，指定的目录一定不能是 DB2 生成的目录。例如，提供以下命令:

```
db2 backup database sample to c:\backup
db2 restore database sample from c:\backup
```

DB2 在 `c:\backup` 目录下生成子目录，该目录应被忽略。要精确地指定要复原哪个备份映象，可使用 TAKEN AT 参数。可能会有数个备份映象存储在同一路径中。

如果指定了若干项，而最后一项是磁带机，会提示用户放入另一磁带。有效的响应是:

c 继续 - 继续使用生成了警告消息的设备（例如在安装了新磁带后继续）。

d 设备终止 - 只停止使用生成了警告消息的设备（例如，在没有更多磁带时终止）。

t 终止 - 在用户执行某些由实用程序请求的操作失败后，异常终止复原操作。

在 OS/2 上不支持磁带。在 OS/2 上，可指定 `0` 或 `0:`，使复原实用程序调用用户出口程序。（仅当使用用户出口程序来备份数据库时才会发生这种情况。）通过用户出口程序复原时，数据库的路径是用于定位容器的唯一参考，因此将复原那个数据库的所有容器。

使用用户出口程序时，不允许重定向复原。

LOAD shared-library

共享库（Windows 操作系统或 OS/2 上的 DLL）的名称，它包含将使用的供应商备份与复原 I/O 函数。该名称可包含全路径。如果未提供全路径，该值将缺省为用户出口程序所驻留的路径。

TAKEN AT date-time

数据库备份映象的时间戳记。时间戳记在成功完成备份操作后显示，并且是备份映象的路径名的一部分。以格式 *yyyymmddhhmmss* 指定。还可指定部分时间戳记。例如，如果有两个不同的备份映象，时间戳记分别为 19971001010101 和 19971002010101，指定 19971002 会导致使用时间戳记为 19971002010101 的那个映象。如果未指定此参数的值，源媒体上必须只有一个备份映象。

TO target-directory

目标数据库目录。如果实用程序复原到一个现有数据库，将忽略此参数。

注: 在 Windows 操作系统或 OS/2 上，只指定使用此参数时的盘符。若指定较长的路径，将返回错误。

INTO target-database-alias

目标数据库别名。如果目标数据库不存在，将创建它。

NEWLOGPATH directory

将用于在复原操作后归档日志文件的目录的全限定名。此参数与数据库配置参数 *newlogpath* 有相同的功能，但此参数的影响只限于它所指定的复原操作。当备份映象中的日志路径不适合在复原操作后使用时可使用此参数，例如，在该路径不再有效时或由另一数据库使用时。

WITH num-buffers BUFFERS

将使用的缓冲区数。缺省值是 2。但是在正从多个源读取数据时，或已增加了 PARALLELISM 的值时，可使用较大的缓冲区数以改进性能。

BUFFER buffer-size

将用于复原操作的，以页计的缓冲区大小。此参数的最小值是 8 页；缺省值是 1024 页。如果指定缓冲区大小为 0，将使用数据库管理器配置参数 *restbufsz* 的值作为缓冲区分配大小。

复原缓冲区大小必须是在备份操作期间指定的备份缓冲区大小的正整数倍数。如果指定了不正确的缓冲区，则分配的缓冲区将是最小的可接受大小。

在 SCO UnixWare 7 上使用磁带机时，指定缓冲区大小 16。

DLREPORT filename

文件名必须是全限定名称（如果指定的话）。在复原操作期间，作为快速

协调的结果，报告变为不再链接的文件。仅当正在复原的表有 DATALINK 列类型和链接文件时，才使用此选项。

REPLACE EXISTING

如果存在一个别名与目标数据库别名相同的数据库，此参数指定复原实用程序将使用复原数据库替换现有数据库。此参数对于调用复原实用程序的脚本很有用，因为命令行处理器将不会提示用户验证是否删除了现有数据库。如果指定了 WITHOUT PROMPTING 参数，则不需要指定 REPLACE EXISTING，但在这种情况中，如果发生了通常需要用户干预的事件，操作会失败。

REDIRECT

指定重定向复原操作。要完成重定向复原操作，在此命令后应跟有一个或多个 SET TABLESPACE CONTAINERS 命令，然后跟有指定了 CONTINUE 选项的 RESTORE DATABASE 命令。

注: 必须从同一窗口或 CLP 会话调用与一个重定向复原操作相关的所有命令。

WITHOUT ROLLING FORWARD

指定在成功复原数据库后，将把该数据库置于前滚暂挂状态。

如果在成功地进行了复原操作后数据库处于前滚暂挂状态，必须调用 第126页的『ROLLFORWARD DATABASE 命令』才能再次使用该数据库。

WITHOUT DATALINK

指定将把任何带有 DATALINK 列的表置于 “DataLink 协调暂挂” (DRP) 状态，以及将不协调任何链接文件。

PARALLELISM n

指定将分散在复原操作期间的缓冲区操纵器的数量。缺省值为 1。

WITHOUT PROMPTING

指定复原操作将以无人照管方式运行。那些通常需要用户干预的操作将返回一条错误消息。如果使用可更换的媒体设备（例如磁带或软盘），即使未指定此选项也会在设备结束时提示用户。

复原数据库 API

C API 语法

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurestore (
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    void * pRestoreInfo,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

一般 API 语法

```
/* File: sqlutil.h */
/* API: Restore Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqlgrestore (
    unsigned short SourceDbAliasLen,
    unsigned short TargetDbAliasLen,
    unsigned short TimestampLen,
    unsigned short TargetPathLen,
    unsigned short UserNameLen,
    unsigned short PasswordLen,
    unsigned short ReportFileLen,
    unsigned short Reserved2Len,
    char * pSourceDbAlias,
    char * pTargetDbAlias,
    sqluint32 BufferSize,
    sqluint32 RollforwardMode,
    sqluint32 DatalinkMode,
    sqluint32 RestoreType,
    sqluint32 RestoreMode,
    sqluint32 CallerAction,
    char * pApplicationId,
    char * pTimestamp,
    char * pTargetPath,
    sqluint32 NumBuffers,
    char * pReportFile,
    struct sqlu_tablespace_bkrst_list * pTablespaceList,
    struct sqlu_media_list * pMediaSourceList,
    char * pUserName,
    char * pPassword,
    void * pReserved2,
    sqluint32 VendorOptionsSize,
    void * pVendorOptions,
    sqluint32 Parallelism,
    unsigned short RestoreInfoSize,
    void * pRestoreInfo,
    unsigned short ContainerPageListSize,
    void * pContainerPageList,
    void * pReserved3,
    struct sqlca * pSqlca);
/* ... */
```

API 参数

SourceDbAliasLen

输入。一个 2 字节的无符号整数，代表以字节计的源数据库别名的长度。

TargetDbAliasLen

输入。一个 2 字节的无符号整数，代表以字节计的目标数据库别名的长度。如果未指定目标数据库别名，则设置为 0。

TimestampLen

输入。一个 2 字节的无符号整数，代表以字节计的时间戳记的长度。如果未提供时间戳记，则设置为 0。

TargetPathLen

输入。一个 2 字节的无符号整数，代表以字节计的目标目录的长度。如果未提供目标路径，则设置为 0。

UserNameLen

输入。一个 2 字节的无符号整数，代表以字节计的用户名的长度。如果不提供用户名，则设置为 0。

PasswordLen

输入。一个 2 字节的无符号整数，代表以字节计的密码的长度。如果不提供密码，则设置为 0。

ReportFileLen

输入。一个 2 字节的无符号整数，代表以字节计的报告文件名的长度。如果未提供报告文件名，则设置为 0。

Reserved2Len

输入。一个 2 字节的无符号整数，代表以字节计的保留区域的长度。设置为 0。

pSourceDbAlias

输入。包含了源数据库备份映象的数据库别名的字符串。

pTargetDbAlias

输入。包含了目标数据库别名的字符串。如果此参数的值为 NULL，则使用 *pSourceDbAlias* 的值。

BufferSize

输入。以 4 KB 分配单位（页）计的备份缓冲区大小。最小值是 8 个单元。缺省值是 1024 个单元。

指定的缓冲区大小必须等于用于产生备份映象的缓冲区大小，或是它的整数倍数。

RollforwardMode

输入。指定在复原操作结束时，是否将数据库置于前滚暂挂状态。有效的值（在 `sqlutil` 中定义）是:

SQLUD_ROLLFWD
在成功复原后，将数据库置于前滚暂挂状态。

SQLUD_NOROLLFWD
在成功复原后，不将数据库置于前滚暂挂状态。

如果在成功地进行了复原操作后数据库处于前滚暂挂状态，必须调用第131页的『前滚数据库 API』 才能再次使用该数据库。

DatalinkMode
输入。指定是否将把任何带有 DATALINK 列的表置于 “DataLink 协调暂挂” (DRP) 状态，以及是否将协调链接文件。有效的值（在 sqlutil 中定义）是:

SQLUD_DATALINK
执行协调操作。带有已定义的 DATALINK 列的表必须指定了 RECOVERY YES 选项。

SQLUD_NODATALINK
不执行协调操作。具有 DATALINK 列的表被置于 DRP 状态。带有已定义的 DATALINK 列的表必须指定了 RECOVERY YES 选项。

RestoreType
输入。指定复原操作的类型。有效的值（在 sqlutil 中定义）是:

SQLUD_FULL
从备份映象复原所有内容。可以脱机运行。以后将不再支持此值。而是使用 SQLUD_DB 来指定完整数据库复原操作。

SQLUD_DB
复原数据库中的所有表空间。可以脱机运行。

SQLUD_ONLINE_TABLESPACE
只复原表空间级的备份映象。可以联机运行。以后将不再支持此值。使用 SQLUD_TABLESPACE | SQLUD_ONLINE 来指定联机表空间级的复原操作。

SQLUD_TABLESPACE
只复原表空间级的备份映象。可以联机或脱机运行。

SQLUD_HISTORY
只复原恢复历史记录文件。

SQLUD_INCREMENTAL
执行手工累积复原操作。

SQLUD_AUTOMATIC
执行自动累积（增量）复原操作。必须用 SQLUD_INCREMENTAL 来指定；即 SQLUD_INCREMENTAL | SQLUD_AUTOMATIC。

RestoreMode
输入。指定复原操作将脱机执行还是联机执行。有效的值（在 sqlutil 中定义）是:

SQLUD_OFFLINE
执行脱机复原操作。

SQLUD_ONLINE
执行联机复原操作。

CallerAction
输入。指定要采取的操作的类型。有效的值（在 sqlutil 中定义）是:

SQLUD_RESTORE
启动复原操作。

SQLUD_TERMINATE_INCREMENTAL
在增量复原操作完成前终止它。

SQLUD_NOINTERRUPT
启动复原操作。指定复原操作将以无人照管方式运行。将尝试那些通常需要用户干预的方案而不首先返回到调用程序，或者这些方案将产生错误。例如在已知复原操作所需的所有媒体都已安装，且不希望出现实用程序提示时，使用此调用程序操作。

SQLUD_CONTINUE
继续使用生成了警告消息的设备（例如在安装了新磁带后继续）。

SQLUD_TERMINATE
在用户执行某些由实用程序请求的操作失败后，异常终止复原操作。

SQLUD_DEVICE_TERMINATE
从由复原实用程序使用的设备列表中除去设备。在设备已用完了它的输入时，复原实用程序会返回警告给调用程序。再次使用此调用程序操作来调用复原实用程序。生成警告的设备将从正在使用的设备的列表中除去。

SQLUD_PARM_CHECK
验证参数而不执行复原操作。

SQLUD_RESTORE_STORDEF
初始调用。请求重新定义表空间容器。

第一次调用时，*CallerAction* 必须设置为 SQLUD_RESTORE、SQLUD_NOINTERRUPT、SQLUD_RESTORE_STORDEF 或 SQLUD_PARM_CHECK。

pApplicationId
输出。提供缓冲区长度 SQLU_APPLID_LEN+1（在 sqlutil 中定义）。复原实用程序返回一个字符串，标识正在为应用程序提供服务的代理程序。可与数据库系统监视器一起使用，以监视应用程序。

pTimestamp
输入。一个代表备份映象的时间戳记的字符串。如果指定的源中只有一个备份映象，则些字段是可选的。

pTargetPath
输入。一个包含目标数据库目录的相对名称或全限定名称的字符串。如果在复原操作期间将创建新数据库，可使用此参数。

NumBuffers
输入。将用于复原操作的缓冲区数。

pReportFile
文件名必须是全限定名称（如果指定的话）。在复原操作期间，作为快速协调的结果，报告变为不再链接的文件。仅当正在复原的表有 DATALINK 列类型和链接文件时，才使用此选项。

pTablespaceList
指定将复原一个或多个表空间。在复原备份映象的子集时，或从表空间级备份映象复原表空间时使用。

具有以下局限性:

- 数据库必须是可恢复的。可恢复数据库将 *logretain* 数据库配置参数设置为 "RECOVERY"、启用 *userexit* 数据库配置参数，或这两者同时进行。
- 正在复原的数据库必须与用于创建备份映象的数据库是同一数据库。即，表空间不能通过表空间复原函数来添加到数据库。
- 从 OS/2 上的用户出口复原时，此函数不可用。
- 前滚实用程序确保在 MPP 环境中复原的表空间与包含相同表空间的任何其他节点同步。

注: 复原自从备份以来已重命名的表空间时，必须在调用复原实用程序时使用新的表空间名。如果使用旧名称，将找不到表空间。

pMediaSourceList

输入。备份映象的源媒体。参见第83页的『数据结构: SQLU-MEDIA-LIST』。调用程序需要在此结构中提供的信息取决于 *media_type* 字段的值。此字段的有效值（在 sqlutil 中定义）是:

SQLU_LOCAL_MEDIA

本地设备（磁带、磁盘或软盘的组合）。提供一个 *sqlu_media_entry* 结构的列表。在 Windows 操作系统或 OS/2 上，只可输入目录路径，而不能是磁带机的名称。

SQLU_TSM_MEDIA

TSM。不需要其他任何输入。使用 DB2 附带提供的 TSM 共享库。如果希望使用 TSM 的另一版本，可使用 SQLU_OTHER_MEDIA 并提供共享库名。

SQLU_OTHER_MEDIA

供应商产品。提供 *sqlu_vendor* 结构中的共享库的名称。

SQLU_USER_EXIT

用户出口。不需要其他任何输入（只在 OS/2 上可用）。

pUserName

输入。包含将用于连接的用户名的字符串。

pPassword

输入。包含将用于认证用户名的密码的字符串。

pReserved2

保留以备将来使用。

VendorOptionsSize

输入。供应商选项字段的长度。长度不能超出 65535 个字节。

pVendorOptions

输入。由供应商用于将信息从应用程序发送给供应商函数。此数据结构必须是平面的，因此不支持间接级别。应注意字节转换未完成，且未检查此数据的代码页。

Parallelism

输入。分区内并行性的程度（缓冲区操纵器的数量）。

RestoreInfoSize

保留以备将来使用。

pRestoreInfo

保留以备将来使用。

ContainerPageListSize
保留以备将来使用。

pContainerPageList
保留以备将来使用。

pReserved3
保留以备将来使用。

pSqlca
输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

REXX API 语法

```
RESTORE DATABASE source-database-alias [USING :value] [USER username USING password]
[TABLESPACE :tablespacenames] [ONLINE | HISTORY FILE ]
[LOAD shared-library [OPTIONS vendor-options] [OPEN num-sessions SESSIONS] |
 FROM :source-area | USE TSM [OPEN num-sessions SESSIONS] | USER_EXIT]
[TAKEN AT timestamp] [TO target-directory] [INTO target-database-alias]
[ACTION caller-action] [WITH num-buffers BUFFERS] [BUFFERSIZE buffer-size]
[WITHOUT ROLLING FORWARD] [PARALLELISM parallelism-degree]
```

REXX API 参数

source-database-alias
从其建立数据库备份映象的源数据库的别名。

value 组合 REXX 主机变量，对它返回数据库复原信息。在以下示例中，XXX 代表主机变量名:

XXX.0 变量中的元素号（始终为 1）

XXX.1 标识为应用程序提供服务的代理程序的应用程序标识。

username
标识将用于连接的用户名。

password
用于认证用户名的密码。

tablespacenames
组合 REXX 主机变量，包含将复原的表空间的列表。在以下示例中 XXX 是主机变量的名称:

XXX.0 要复原的表空间号

XXX.1 第一个表空间的名称

XXX.2 第二个表空间的名称

XXX.3 以此类推。

HISTORY FILE
指定后，只从备份映象复原历史记录文件。

shared-library
包含将使用的供应商复原 I/O 函数的共享库（Windows 操作系统或 OS/2 上的 DLL）名称。它可包含全路径。如果提供全路径，将缺省为用户出口程序所驻留的路径。

vendor-options
供应商函数所需要的信息。

num-sessions
将与 TSM 或供应商产品一起使用的 I/O 会话数。

source-area
组合 REXX 主机变量，它表示备份映象将驻留在的目录和设备。缺省值是当前目录。在 Windows 操作系统或 OS/2 上，只可输入目录路径，而不能是磁带机的名称。

timestamp
数据库备份映象的时间戳记。

target-directory
目标数据库目录。

target-database-alias
目标数据库别名。如果目标数据库不存在，将创建它。

caller-action
指定要采取的操作。有效的值为:

SQLUD_RESTORE
启动复原操作。

SQLUD_NOINTERRUPT
启动复原操作。指定复原操作将以无人照管方式运行。将尝试那些通常需要用户干预的方案而不首先返回到调用程序，或者这些方案将产生错误。例如在已知复原操作所需的所有媒体都已安装，且不希望出现实用程序提示时，使用此调用程序操作。

SQLUD_CONTINUE
继续使用生成了警告消息的设备（例如在安装了新磁带后继续）。

SQLUD_TERMINATE
在用户执行某些由实用程序请求的操作失败后，异常终止复原操作。

SQLUD_DEVICE_TERMINATE
从由复原实用程序使用的设备列表中除去设备。在设备已用完了它的输入时，复原实用程序会返回警告给调用程序。再次使用此调用程序操作来调用复原实用程序。生成警告的设备将从正在使用的设备的列表中除去。

SQLUD_PARM_CHECK
验证参数而不执行复原操作。

SQLUD_RESTORE_STORDEF
初始调用。请求重新定义表空间容器。

num-buffers
将使用的备份缓冲区数。

buffer-size
以 4 KB 分配单位计的备份缓冲区大小。最小值是 16 个单元。

parallelism-degree
分区内并行性的程度（缓冲区操纵器的数量）。

示例复原会话

CLP 示例

以下是一个典型的重定向复原方案，用于别名为 MYDB 的数据库:

1. 发出 RESTORE DATABASE 命令，使用 REDIRECT 选项。

   ```
   db2 restore db mydb replace existing redirect
   ```

   成功地完成了步骤 1 后，在完成步骤 3 前，可发出以下命令来异常终止复原操作:

   ```
   db2 restore db mydb abort
   ```

2. 对必须重新定义容器的每个表空间发出 SET TABLESPACE CONTAINERS 命令。例如，在 OS/2 发出:

   ```
   db2 set tablespace containers for 5 using
      (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   要验证复原数据库的容器是否是在此步骤中指定的那些容器，可发出 LIST TABLESPACE CONTAINERS 命令。

3. 在成功地完成了步骤 1 和 2 后，发出:

   ```
   db2 restore db mydb continue
   ```

   这是重定向复原操作的最后一步。

4. 如果步骤 3 失败，或者如果已异常终止了复原操作，则可从步骤 1 开始重新启动重定向的复原。

以下是对可恢复数据库每周进行的增量备份策略的样本。包括一个每周进行的完整数据库备份操作、一个每天进行的非累积 (delta) 备份操作以及一个在每周中期进行的累积（增量）备份操作:

```
(Sun) backup db kdr use tsm
(Mon) backup db kdr online incremental delta use tsm
(Tue) backup db kdr online incremental delta use tsm
(Wed) backup db kdr online incremental use tsm
(Thu) backup db kdr online incremental delta use tsm
(Fri) backup db kdr online incremental delta use tsm
(Sat) backup db kdr online incremental use tsm
```

要对在星期五早上创建的映象自动进行数据库复原，可发出:

```
restore db kdr incremental automatic taken at (Thu)
```

要对在星期五早上创建的映象手工进行数据库复原，可发出:

示例复原会话

```
restore db kdr incremental taken at (Thu)
restore db kdr incremental taken at (Sun)
restore db kdr incremental taken at (Wed)
restore db kdr incremental taken at (Thu)
```

在第397页的『附录F. 恢复 CLP 脚本』中提供了一个样本 DB2 命令脚本以及如何使用它的信息。

API 示例

在第331页的『附录E. 恢复样本程序』中提供了包含 DB2 API 和嵌入式 SQL 调用的样本程序以及如何使用它们的信息。

优化复原性能

要缩短完成一次复原操作所需的时间:

- 增加复原缓冲区大小。

  复原缓冲区大小必须是在备份操作期间指定的备份缓冲区大小的正整数倍数。如果指定了不正确的缓冲区，则分配的缓冲区将是最小的可接受大小。
- 增加缓冲区的数量。

  指定的值必须是为备份缓冲区指定的页数的倍数。最小页数为 16。

复原局限性

复原实用程序有以下局限性:

- 仅当先前已使用 DB2 备份实用程序备份了数据库时，才能使用复原实用程序。
- 如果正在使用 “DB2 控制中心”，则不能复原用先前版本的 DB2 创建的备份映象。
- 数据库复原操作在正在运行前滚进程时不能启动。
- 仅当表空间当前已存在且是同一表空间时才能复原表空间; “同一表空间”表示该表空间未删除，然后在备份和复原操作之间重新创建。
- 不能将表空间级的备份复原到新数据库。
- 不能执行涉及系统目录表的联机表空间级复原操作。
- 在 OS/2 上调用用户出口程序时，部分或子集复原操作是不可能的。
- 如果正在复原的数据库的代码页与可用于应用程序的代码页不匹配，或如果数据库管理器不支持从数据库代码页到可用于应用程序的代码页的代码页转换，复原的数据库将不可用。

复原故障诊断

如果在尝试将备份映象复原到另一系统上的新数据库时返回了 SQL0970N，且您的原始数据库中有使用绝对路径定义的表空间，应使用重定向的复原操作来定义新系统上的表空间容器。复原实用程序会尝试使用新系统上不存在的绝对路径和容器。

如果在数据库复原操作期间发生了系统故障，将不能连接到数据库直到再次调用复原实用程序并成功地完成了复原操作。如果在表空间复原操作期间发生了系统故障，则只有正在复原的表空间是不可用的。数据库中的其他表空间仍可以使用。

第4章 前滚恢复

这部分描述了 DB2 UDB 前滚实用程序，它通过应用记录在数据库恢复日志文件中的事务来恢复数据库。

包括下列主题:

前滚概述

DB2 ROLLFORWARD DATABASE 命令的最简单的格式只需要指定您想执行前滚恢复的数据库的别名。例如:

```
db2 rollforward db sample stop
```

在此示例中，命令返回:

```
                                 Rollforward Status

 Input database alias                   = sample
 Number of nodes have returned status   = 1

 Node number                            = 0
 Rollforward status                     = not pending
 Next log file to be read               =
```

```
 Log files processed                  =  -
 Last committed transaction           = 2001-03-11-02.39.48.000000

DB20000I  ROLLFORWARD 命令成功完成。
```

关于这些字段的说明，参见 第126页的『ROLLFORWARD DATABASE 命令』。

前滚恢复的常见方法涉及:

1. 调用前滚实用程序，而不使用 STOP 选项
2. 调用前滚实用程序，使用 QUERY STATUS 选项

   如果指定恢复到日志末尾，而返回的时间点又早于您预期的时间点，QUERY STATUS 选项会指示丢失了一个或多个日志文件。

   如果指定时间点恢复，QUERY STATUS 选项将有助于您确保前滚操作已在正确的时间点完成。
3. 使用 STOP 选项调用前滚实用程序。在操作停止后，不可能前滚其他更改。

必须已成功地复原了数据库（使用复原实用程序）才能前滚它，但对表空间则不是这样。表空间可以暂时置于前滚暂挂状态，但不需要复原操作撤销它（例如在电源中断后）。

调用前滚实用程序时:

- 如果数据库处于前滚暂挂状态，将前滚该数据库。如果表空间也处于前滚暂挂状态，则必须在数据库前滚操作完成后，再次调用前滚实用程序来前滚该表空间。
- 如果数据库*不*处于前滚暂挂状态，但该数据库中的表空间*处于*前滚暂挂状态:
  - 如果指定表空间列表，将只前滚那些表空间。
  - 如果不指定表空间列表，将前滚处于前滚暂挂状态的所有表空间。

数据库前滚操作是脱机运行的。直到前滚操作成功完成数据库才可用，除非在调用实用程序时指定了 STOP 选项，否则该操作不能完成。

表空间前滚操作可脱机运行。直到前滚操作成功完成数据库才可用。当达到日志末尾时或在调用实用程序时指定了 STOP 选项，会发生这种情况。

只要表空间中包括了 SYSCATSPACE，就可对表空间执行*联机*前滚操作。对表空间执行联机前滚操作时，该表空间不可用，但数据库中的其他表空间是可用的。

首次创建数据库时，只对它启用了循环记录。这意味着日志是可重用的，而不是被保存或归档。使用循环记录，是不可能进行前滚恢复的；只可进行崩溃复原或版本复原（参见第25页的『了解恢复日志』）。归档日志将建立备份后对数据库进行的更改记录成文档。通过将 *logretain* 数据库配置参数设置为 RECOVERY 或将

userexit 数据库配置参数设置为 YES，或同时设置这两个参数，启用日志归档（和前滚恢复）。这两个参数的缺省值是 NO 因为最初没有备份映象可用于复原数据库。更改这两个参数的值或其中一个的值时，数据库将处于备份暂挂状态，必须脱机备份该数据库才能再次使用它。

关于与记录相关的数据库配置参数的更多信息，参见第31页的『数据库记录的配置参数』。

使用前滚所需的特权、权限和授权

特权使用户能够创建或访问数据库资源。权限级别提供将特权分组的方法，以及更高级的数据库和实用程序操作。这两者一起用于控制对数据库管理器和它的数据库对象的访问。用户只能访问那些他们具有适当授权（即必需的特权或权限）的对象。

必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限才能使用前滚实用程序。

使用前滚

使用前滚之前

不应连接到要进行前滚恢复的数据库：前滚实用程序会自动建立与特定数据库的连接，且此连接在前滚操作完成时终止。

数据库可以是本地的或远程的。

调用前滚

前滚实用程序可通过以下各项调用:

- 命令行处理器 (CLP)。

  以下是通过 CLP 发出的 ROLLFORWARD DATABASE 命令的示例:

  ```
  db2 rollforward db sample to end of logs and stop
  ```

- “控制中心”中的“前滚数据库”笔记本。要打开“前滚数据库”笔记本:
  1. 从“控制中心”中展开对象树，直到找到“数据库”文件夹。
  2. 单击“数据库”文件夹。所有现有的数据库都显示在该窗口右边的窗格（内容窗格）中。
  3. 在内容窗格中，用鼠标右按钮单击需要的数据库，然后从弹出菜单中选择“前滚”。“前滚数据库”笔记本打开。

  关于“控制中心”的通用信息，请参阅《*管理指南*》。在“控制中心”中通过联机帮助设施提供了详细信息。

- 应用程序编程接口 (API), **sqluroll**。关于此 API 的信息，参见第131页的『前滚数据库 API』。关于创建包含 DB2 管理 API 的应用程序的通用信息，请参阅《*应用程序构建指南*》。

在分区数据库环境中，必须从数据库的目录节点发出前滚实用程序。

前滚表空间中的更改

如果数据库启用了前滚恢复，您可以选择备份、复原和前滚表空间，而不必前滚整个数据库。可能需要对个别表空间实现恢复策略，因为这可节省时间：复原数据库的一部分所需的时间比恢复整个数据库所需的时间少。例如，若一个磁盘损坏且它只包含一个表空间，则可以复原并前滚该表空间，而不必复原整个数据库，也不会影响用户访问该数据库的其余部分，除非损坏的表空间包含了系统目录表；在这种情况中，您不能连接至数据库。（如果包含系统目录表空间的表空间级备份映象是可用的，则可以独立复原系统目录表空间。）表空间级的备份也允许您更频繁地备份数据库的关键部分，因此所需的时间比备份整个数据库的时间少。

复原表空间后，它始终处于前滚暂挂状态。要使该表空间可以使用，必须对它执行前滚恢复。在大多数情况中，可以选择前滚至日志末尾或前滚至特定的时间点。但是不能将包含系统目录表的表空间前滚至某时间点。这些表空间必须前滚至日志末尾，以确保数据库中的所有表空间保持一致。

在前滚表空间之前，调用 LIST TABLESPACES SHOW DETAIL 命令。此命令将返回*最小恢复时间*，这是表空间可前滚至的最早时间点。对表空间或表空间中的表运行数据定义语言（DDL）语句时，将更新此最小恢复时间。必须将该表空间至少前滚至最小恢复时间，以便与系统目录表中的信息同步。如果恢复多个表空间，表空间必须至少前滚至要恢复的所有表空间中的最高“最小恢复时间”。在分区数据库环境中，对所有分区发出 LIST TABLESPACES SHOW DETAIL 命令。表空间必须前滚至所有分区上的所有表空间中的最高“最小恢复时间”。

如果要将表空间前滚到某时间点，而且有一个表包含在多个表空间中，则必须同时前滚所有这些表空间。例如，若该表数据包含在一个表空间中，而该表的索引包含在另一个表空间中，则必须同时将这两个表空间前滚至相同的时间点。

若表中的数据和长整数对象位于不同的表空间中，且已将该表重组，则必须一起复原和前滚数据和长整数对象的表空间。在重组该表之后，应该为受影响的表空间建立备份。

若要将一个表空间前滚到某时间点，则表空间中的表是:

- 另一个表空间中的摘要表的基础表，或

- 另一个表空间中的表的摘要表

应该将两个表空间前滚到同一时间点。否则，前滚操作结束时后，摘要表将处于检查暂挂状态。

若要将一个表空间前滚至一个时间点，而该表空间中的一个表与另一个表空间所包含的另一个表存在参考完整性关系，则应同时将这两个表空间前滚至相同的时间点。否则，这两个表空间将在时间点前滚操作结束时处于检查暂挂状态。若同时前滚这两个表空间，则该约束将在该时间点前滚操作结束保持有效。

应确保时间点表空间前滚操作不会导致在某些表空间中回滚一个事务，而在另一些表空间中落实该事务。这种情况可能在下列时候发生:

- 对一个事务已更新的表空间的子集执行时间点前滚，且该时间点在事务落实的时间之前。
- 要前滚至某个时间点的表空间中所包含的任何表有一个相关的触发器，或者这个表由一个触发器来更新，该触发器影响的表空间不是要前滚的表空间。

解决方案是: 找到一个可阻止这种情况发生的适当的时间点。

可以发出 QUIESCE TABLESPACES FOR TABLE 命令来创建事务一致的时间点，以将表空间前滚。停顿请求（是共享的，用于更新或独占方式）等待（通过锁定）对那些表空间运行的所有事务完成，并阻拦新请求。准许停顿请求时，表空间处于一致状态。要确定停止前滚操作的适当时间，可查看恢复历史记录文件以找出停顿点，并检查它是否是在最小复原时间后发生的。

表空间时间点前滚操作完成后，表空间将置于备份暂挂状态。必须建立该表空间的备份，因为在前滚到的时间点与当前时间之间对该表空间所做的所有更新都已被除去。再也不能从先前的数据库级或表空间级备份映象将该表空间前滚到当前时间。以下示例显示表空间级的备份映象为什么是必需的，以及如何使用它。（要使表空间可用，可以备份整个数据库、处于备份暂挂状态的表空间，或包括处于备份暂挂状态的表空间的表空间集合。）

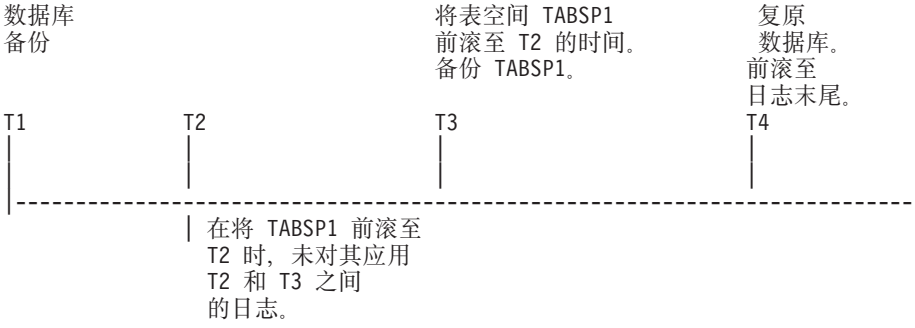


图 17. 表空间备份需求

在上述示例中，在 T1 时备份数据库。然后在 T3 时，表空间 TABSP1 前滚至某特定的时间点 (T2)，表空间在时间 T3 后备份。因为表空间处于备份暂挂状态，所以备份操作是必需的。表空间备份映象的时间戳记在 T3 后，但表空间是在 T2。不将 T2 和 T3 之间的日志记录应用于 TABSP1。在时间 T4 处，将使用在 T1 处创建的备份映象复原数据库，并前滚至日志末尾。表空间 TABSP1 将在时间 T3 处置于复原暂挂状态，因为数据库管理器假设在 T3 和 T4 之间，对 TABSP1 执行操作，而无需已对表空间应用了 T2 和 T3 之间的日志更改。如果这些日志更改是作为对数据库的前滚操作的一部分应用的，该假设将不成立。必须在表空间前滚到某时间点后建立的表空间级的备份，使您可将那个表空间前滚过前一时间点前滚操作（在上例中为 T3）。

假定要将表空间 TABSP1 复原至 T4，应该从 T3 之后建立的备份映象（必需的备份或稍后的一个）复原该表空间，然后将 TABSP1 前滚至日志末尾。

在前面的示例中，将数据库复原至时间 T4 最有效的方法应是按下列次序执行必需的步骤:

1. 复原数据库。
2. 复原表空间。
3. 前滚数据库。
4. 前滚表空间。

因为您在前滚数据库之前复原表空间，所以在前滚数据库时将不使用资源向表空间应用日志记录。

如果找不到时间 T3 后的 TABSP1 备份映象，或者想将 TABSP1 复原到 T3（或更早），可以:

- 将表空间前滚至 T3。无需再次复原表空间，因为它已根据数据库备份映象复原。

- 再次使用在时间 T1 建立的数据库备份来复原表空间，然后将该表空间前滚至时间 T3 之前的一个时间。
- 删除表空间。

在分区数据库环境中:

- 必须同时将一个表空间的所有部分前滚至同一时间点。这可确保该表空间在各数据库分区中是一致的。
- 如果某些数据库分区处于前滚暂挂状态，而且在其他数据库分区上，某些表空间处于前滚暂挂状态（但数据库分区不是），您必须首先前滚数据库分区，然后前滚表空间。
- 如果打算将表空间前滚到日志末尾，不必在每个数据库分区上复原: 只需在需要恢复的数据库分区上复原即可。但如果打算将表空间前滚到某时间点，则必须在每个数据库分区上复原它。

恢复删除的表

可能您偶然会删除仍需要其数据的表。如果是这样，您应该考虑在删除表操作后使关键的表成为可复原的。

可通过调用数据库复原操作复原表数据，后跟一个前滚到删除表前的某时间点的数据库前滚操作。如果数据库很大，这可能会花很多时间，并使您的数据在恢复期间不可用。

DB2 删除的表恢复功能使您可使用表空间级的复原和前滚操作来恢复删除的表数据。这样可比数据库级的恢复要快，且您的数据库将对用户保持可用。

要使删除的表可以复原，必须对该表驻留的表空间启用 DROPPED TABLE RECOVERY 选项。可在表空间创建时完成，或通过调用 ALTER TABLESPACE 语句完成（请参阅 *SQL Reference*）。DROPPED TABLE RECOVERY 选项是表空间特定的，并限于对常规表空间使用。要确定是否对表空间启用了删除的表恢复，可以查询 SYSCAT.TABLESPACES 目录表中的 DROP_RECOVERY 列。

对表（对该表的表空间启用了删除的表恢复）运行 DROP TABLE 语句时，将在日志文件中建立另一条目（标识删除的表）。还会在恢复历史记录文件中建立一个条目，包含可用于重新创建表的信息。

一次只能复原一个删除的表。可执行下列操作来复原删除的表:

1. 通过调用 LIST HISTORY DROPPED TABLE 命令来标识删除的表（参见 第288页的『LIST HISTORY』）。删除的表标识列示在“备份标识”列中。
2. 复原在删除该表前所建立的数据库级或表空间级备份映象。

3. 创建包含表数据的文件将写至的导出目录。该目录必须可对所有数据库分区访问，或存在于每个分区上。此导出目录下的子目录是由每个数据库分区自动创建的。这些子目录的名称是 NODE*nnnn*，其中 *nnnn* 代表数据库分区或节点号。包含删除的表数据的数据文件（就如它存在于每个数据库分区上那样）将导出到称为 data 的较低子目录中。例如 \export_directory\NODE0000\data。
4. 删除表后前滚至某时间点，对 ROLLFORWARD DATABASE 命令使用 RECOVER DROPPED TABLE 选项。也可前滚至日志末尾，以使对表空间或数据库中的其他表进行的更新不会丢失。
5. 使用 CREATE TABLE 语句从恢复历史记录文件重新创建表。
6. 将在前滚操作期间导出的表数据导入表中。

对可从删除的表中复原的数据类型有一些限制。不可能复原:

- 大对象 (LOB) 或长字段数据。对长表空间不支持 DROPPED TABLE RECOVERY 选项。如果尝试复原包含 LOB 或 LONG VARCHAR 列的删除的表，这些列将在生成的导出文件中设置为 NULL。只能对常规表空间使用 DROPPED TABLE RECOVERY 选项，而不能对临时或长表空间使用。
- 与行类型相关的元数据。（数据已复原，但不是元数据。）将复原类型表的分级结构表中的数据。此数据包含的信息可能比已删除的类型表中出现的信息多。

*可以*复原与 DATALINK 列相关的链接文件的名称。导入表数据后，应使用 DB2 Data Links Manager 来协调该表。DB2 Data Links Manager 可能或可以复原这些文件的备份映象，也可能无法复原，这取决于垃圾收集是否已删除它们。

使用装入副本位置文件

DB2LOADREC 注册表变量用于标识包含装入副本位置信息的文件。此文件在前滚恢复期间用于查找该装入副本。它包含以下信息:

- 媒体类型
- 要使用的媒体设备的数目
- 在表的装入操作期间生成的装入副本的位置
- 装入副本的文件名（若适用的话）

如果位置文件不存在，或文件中找不到匹配的条目，将使用日志记录中的信息。

在进行前滚恢复之前，该文件中的信息可能会被覆盖。

注:

1. 在一个分区数据库环境中，DB2LOADREC 注册表变量必须在 db2profile 文件中。

2. 在一个分区数据库环境中，在每个数据库分区服务器中都必须存在该装入副本文件，且文件名（包括路径）必须相同。
3. 若 DB2LOADREC 注册表变量标识的文件中的一个条目无效，将使用旧的装入副本位置文件来提供替换无效条目的信息。

该位置文件提供下列信息。前五个参数必须具有有效值，它们用于标识装入副本。对于记录的每个装入副本，其整体结构是相同的。例如:

```
TIMestamp       19950725182542          * 装入时生成的时间戳记
SCHema          PAYROLL                 * 装入的表的模式
TABlename       EMPLOYEES               * 表名
DATabasename    DBT                     * 数据库名
DB2instance     TORONTO                 * DB2INSTANCE
BUFfernumber    NULL                    * 要用于恢复的缓冲区的数量
SESsionnumber   NULL                    * 要用于恢复的会话的数量
TYPeofmedia     L                       * 媒体的类型 - L 表示本地设备
                                                      A 表示 TSM
                                                      O 表示其他供应商
LOCationnumber 3                        * 位置数
   ENTry        /u/toronto/dbt.payroll.employes.001
   ENT          /u/toronto/dbt.payroll.employes.002
   ENT          /dev/rmt0
TIM             19950725192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             TORONTO
SES             NULL
BUF             NULL
TYP             A
TIM             19940325192054
SCH             PAYROLL
TAB             DEPT
DAT             DBT
DB2             TORONTO
SES             NULL
BUF             NULL
TYP             O
SHRlib          /@sys/lib/backup_vendor.a
```

注:

1. 每个关键字的前三个字符很重要的。所有关键字必须以指定的次序排列。不接受任何空行。
2. 时间戳记的格式是 *yyyymmddhhmmss*。
3. 所有字段都是必需的，BUF 和 SES 除外（它们可以是 NULL）。如果 SES 为 NULL，将使用由 *numloadrecses* 配置参数指定的值。如果 BUF 为 NULL，缺省值是 SES+2。
4. 如果位置文件中即使只有一个条目是无效的，也会使用先前的装入副本位置文件来提供那些值。

5. 媒体类型可以是本地设备（L 指磁带、磁盘或软盘）、TSM (A) 或其他供应商 (0)。如果类型为 L，则需要位置数后跟位置条目。如果类型 A，不需要进一步的输入。如果类型是 0，需要共享库名。关于使用 TSM 和其他供应商产品作为备份媒体的详细信息，参见第405页的『附录G. Tivoli Storage Manager』。
6. SHRlib 参数指向一个库，该库具有存储装入副本数据的函数。
7. 如果调用装入操作时指定 COPY NO 或 NONRECOVERABLE 选项，并且在操作完成后不建立数据库或受影响的表空间的备份副本，不能将数据库或表空间复原到装入操作后的某时间点。即，不能使用前滚恢复将数据库或表空间重新构建到后面的装入操作中的状态。只能将数据库或表空间复原到装入操作前的某时间点。

如果要使用特定的装入副本，可使用数据库的恢复历史记录文件，以确定特定装入操作的时间戳记。在分区数据库环境中，恢复历史记录文件对每个数据库分区都是本地的。

关于装入实用程序的详细信息，请参阅 *Data Movement Utilities Guide and Reference*。

使分区数据库系统中的时钟同步

应该使所有数据库分区服务器的系统时钟保持相对同步，以确保数据库操作顺利进行以及正向可复原性不受限制。数据库分区服务器之间的时间差加上一个事务的任何潜在的操作和通信延迟，应小于对 *max_time_diff*（节点之间的最大时间差）数据库管理器配置参数指定的值。

为了确保运行记录的时间戳记反映分区数据库系统中的事务的顺序，DB2 使用每台机器上的系统时钟作为运行记录中时间戳记的基准。但是，若将系统时钟设置得提前，也自动将日志时钟设置得提前。虽然可以将系统时钟设置得落后，但是日志时钟却不能这样设置，它会保持*相同的*超前时间，直至系统时钟与此时间相匹配为止。于是，这两个时钟便同步了。这表示一个数据库节点上的短期系统时钟错误可能对数据库日志的时间戳记产生长期的影响。

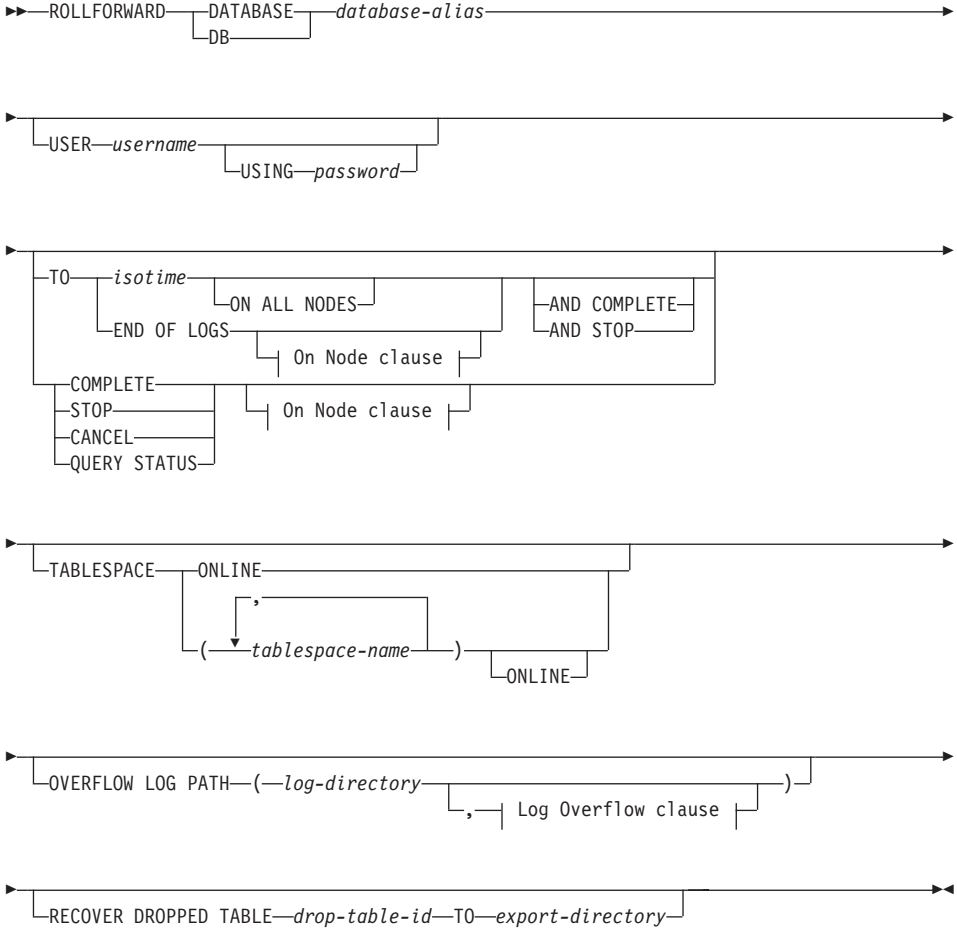
例如，假定数据库分区服务器 A 上的系统时钟被错误地设置为 1999 年 11 月 7 日，而当前年份是 1997 年，并假定在数据库分区服务器上的分区中落实了更新事务之后将该错误校正。若继续使用该数据库，且过一段时间便定期更新它，则 1997 年 11 月 7 日至 1999 年 11 月 7 日之间的任何点实际上是不可通过前滚恢复达到的。当完成数据库分区服务器 A 上的 COMMIT 时，数据库日志中的时间戳记被设置为 1999，而日志的时钟会停留在 1999 年 11 月 7 日，直到系统时钟与此时间相匹配为止。若试图前滚至此时间范围内的一个时间点，则操作将在超过指定的停止点（即 1997 年 11 月 7 日）的第一个时间戳记处停止。

虽然 DB2 不能控制对系统时钟的更新，但是 *max_time_diff* 数据库管理器配置参数降低了发生此类问题的机会:

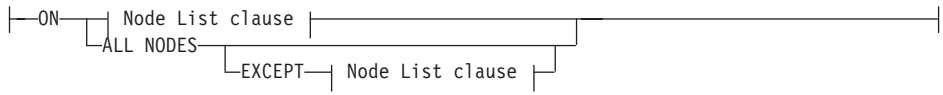
- 此参数的可配置值的范围是 1 分钟至 24 小时。关于设置 *max_time_diff* 的更多信息，请参阅《*管理指南: 性能*》一书。
- 当对非目录节点发出第一个连接请求时，数据库分区服务器会将它的时间发送至该数据库的目录节点。该目录节点就会检查请求该连接的节点上的时间以及它自己的时间是否在 *max_time_diff* 参数指定的范围之内。若超过此范围，则拒绝该连接。
- 涉及到数据库中两个以上数据库分区服务器的更新事务，必须先验证参与的数据库分区服务器上的时钟是否同步，然后才可落实该更新。若两个或多个数据库分区服务器的时差超过了 *max_time_diff* 允许的限制，则会回滚该事务，以防止将不正确的时间传播至其他数据库分区服务器。

ROLLFORWARD DATABASE 命令

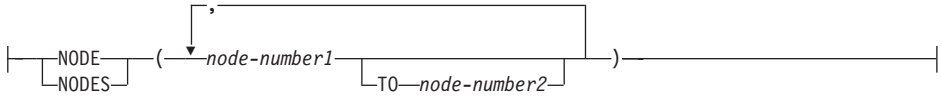
命令语法



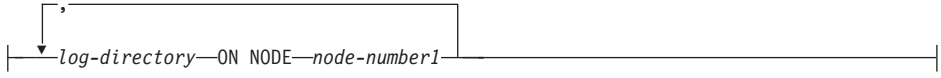
On Node clause:



Node List clause:



Log Overflow clause:



命令参数

DATABASE database-alias
将对其进行前滚恢复的数据库别名。

USER username
对数据库执行前滚恢复时使用的用户名。

USING password
用于认证用户名的密码。如果省略密码，会提示用户输入。

TO

isotime
所有已落实的事务将前滚至的某个时间点（包括在该时间点落实的事务以及之前落实的事务）。

此值指定为时间戳记，即一个标识日期和时间的 7 部分字符串。格式为 *yyyy-mm-dd-hh.mm.ss.nnnnnn*（年、月、日、小时、分钟、秒和微秒），以“世界时坐标”(UTC) 表示。UTC 有助于避免存在与不同日志相关的相同时间戳记（例如，与夏时制相关的时间的更改）。备份映象中的时间戳记基于启动备份操作时的本地时间。CURRENT TIMEZONE 特殊注册器指定 UTC与应用程序服务器上的本地时间的不同。这种不同由一个持续时间来表示（它是一个十进制数，前两位表示小时数，接下来的两位表示分钟数而最后两位表示秒数。）从本地时间中减去 CURRENT TIMEZONE 可将本地时间转换为 UTC。

END OF LOGS
指定将应用来自数据库配置参数 *logpath* 中列示的所有联机归档日志文件的所有已落实事务。

ALL NODES

指定事务将在 db2nodes.cfg 文件中指定的所有节点上前滚。如果未指定节点子句，则这是缺省值。

EXCEPT

指定事务将在 db2nodes.cfg 文件中指定的所有节点上前滚，在节点列表中指定的那些节点除外。

ON NODE / ON NODES

在节点集合上前滚数据库。

node-number1

指定节点列表中的节点号。

node-number2

指定第二个节点号，因此从 *node-number1* 到 *node-number2*（包括 *node-number2*）都将包括在节点列表中。

COMPLETE / STOP

停止前滚日志记录，通过回滚所有未完成的事务并结束数据库的前滚暂挂状态，来完成前滚恢复进程。因此可访问正在前滚的数据库或表空间。这两个关键字是等效的，应指定其中一个而不应同时指定两个。关键字 AND 允许一次指定多个操作，例如 db2 rollforward db sample to end of logs and complete。

注: 将表空间前滚到某时间点时，表空间将被置于备份暂挂状态。

CANCEL

取消前滚恢复操作。它会将已启动了前滚恢复的所有节点上的数据库或一个或多个表空间置于复原暂挂状态:

- 如果*数据库*前滚操作不在进行中（即数据库处于前滚暂挂状态），此选项会将数据库置于复原暂挂状态。
- 如果*表空间*前滚操作不在进行中（即表空间处于前滚暂挂状态），则必须指定表空间列表。列表中的所有表空间都将处于复原暂挂状态。
- 如果表空间前滚操作*正在*进行中（即至少有一个表空间处于“前滚进行中”状态），处于“前滚进行中”状态的所有表空间都将置于复原暂挂状态。如果指定了表空间列表，该列表必须包括处于“前滚进行中”状态的所有表空间。该列表上的所有表空间都将置于复原暂挂状态。
- 如果前滚至某时间点，则传入的任何表空间名都将被忽略，且处于“前滚进行中”状态的所有表空间都将置于复原暂挂状态。
- 如果使用表空间列表前滚至日志末尾，只有列示的表空间将置于复原暂挂状态。

此选项不能用于取消*实际已在运行的*前滚操作。它只能用于取消已在进行中，但此时并未实际运行的前滚操作。前滚操作可在进行中，但并未运行，如果:

- 它异常终止。
- 未指定 STOP 选项。
- 错误导致前滚操作失败。某些错误，例如对不可恢复的装入操作前滚，会使表空间处于复原暂挂状态。

注: 使用此选项时应小心，只有在因为某些表空间已处于前滚暂挂状态或复原暂挂状态，已在进行中的前滚操作才不能完成时使用该选项。如果对状态有疑问，可使用 LIST TABLESPACES 命令来判断表空间是处于“前滚进行中”状态还是前滚暂挂状态。

QUERY STATUS

列示数据库管理器已前滚的日志文件、所需的下一个归档文件，以及自从前滚处理开始以来最后一个落实的事务的时间戳记（格式为 CUT）。在分区数据库环境中，将对每个节点返回此状态信息。返回的信息包含以下字段:

Node number

Rollforward status

状态可以是: 数据库或表空间前滚暂挂、数据库或表空间前滚进行中、数据库或表空间前滚处理 STOP（停止）或未暂挂。

Next log file to be read

包含需要的下一日志文件名的字符串。在分区数据库环境中，如果前滚实用程序失败并有返回码指出: 日志文件丢失或已发生了日志信息的不匹配，则使用此信息。

Log files processed

包含已处理的日志文件名的字符串，恢复操作不再需要这些文件，因此可从目录中除去。例如，如果最旧的未落实事务是在日志文件 *x* 中开始的，则过时日志文件的范围将不包括 *x*; 且范围在 *x* - 1 处结束。

Last committed transaction

包含格式为 ISO 的时间戳记 (*yyyy-mm-dd-hh.mm.ss*) 的字符串。该时间戳记标记完成前滚恢复后，最后一个已落实的事务。时间戳记适用于数据库。对于表空间前滚恢复，它是对数据库落实的最后一个事务的时间戳记。

注: 如果省略 TO、STOP、COMPLETE 或 CANCEL 子句，QUERY STATUS 将是缺省值。如果指定了 TO、STOP 或 COMPLETE，将在命令成功完成时显示状态信息。如果指定了个别表空间，将忽略它们；状态请求不能只对指定的表空间应用。

TABLESPACE

对表空间级的前滚恢复指定此关键字。

tablespace-name

对于表空间级的前滚恢复至某时间点，此选项是必需的。它允许为前滚恢复到日志末尾指定表空间的子集。在分区数据库环境中，正在前滚的每个节点上，列表中的每个表空间不必都存在。如果它*存在*，就必须处于正确的状态。

ONLINE

指定此关键字，以允许联机完成表空间级的前滚恢复。这意味着正在进行前滚恢复时，允许另一代理程序连接。

OVERFLOW LOG PATH log-directory

指定在恢复期间，将在其中搜索归档日志的备用日志路径。如果日志文件移动到由 *logpath* 数据库配置参数指定的位置以外的位置，则使用此参数。在分区数据库环境中，这是*所有节点*的（全限定）缺省溢出日志路径。可对单分区数据库指定相对溢出日志路径。如果前滚实用程序找不到它需要的下一日志，将在 SQLCA 中返回日志名，而前滚恢复会结束。如果没有更多日志可用，则使用 STOP 选项来终止前滚恢复。将回滚未完成的事务，以确保数据库或表空间处于一致状态。

log-directory ON NODE

在分区数据库环境中，允许不同的日志路径覆盖某特定节点的缺省溢出日志路径。

RECOVER DROPPED TABLE drop-table-id

恢复前滚操作期间删除的表。可使用 第288页的『LIST HISTORY』 获取表标识。

TO export-directory

指定包含表数据的文件将被写至的目录。目录必须对所有节点可访问。

前滚数据库 API

C API 语法

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_FN
  sqluroll (
    struct rfwd_input * pRfwdInput,
    struct rfwd_output * pRfwdOuput,
    struct sqlca * pSqlca);
/* ... */
```

一般 API 语法

```
/* File: sqlutil.h */
/* API: Rollforward Database */
/* ... */
SQL_API_RC SQL_API_RN
  sqlgroll (
    struct grfwd_input * grfwdin,
    struct rfwd_output * rfwdout,
    struct sqlca * sqlca);

SQL_STRUCTURE grfwd_input
{
   unsigned short DbAliasLen,
   unsigned short StopTimeLen,
   unsigned short UserNameLen,
   unsigned short PasswordLen,
   unsigned short OverflowLogPathLen,
   unsigned short ReportFileLen,      /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   sqluint32 Version,
   char * pDbAlias,
   unsigned short CallerAction,
   char * pStopTime,
   char * pUserName,
   char * pPassword,
   char * pOverflowLogPath,
   unsigned short NumChngLgOvrflw,
   struct sqlurf_newlogpath * pChngLogOvrflw,
   unsigned short ConnectMode,
   struct sqlu_tablespace_bkrst_list * pTablespaceList,
   short AllNodeFlag,
   short NumNodes,
   SQL_PDB_NODE_TYPE * pNodeList,
   short NumNodeInfo,
   unsigned short DlMode,             /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pReportFile,                /* NOTE: This parameter is no longer used */
                                      /*       for the DB2 Data Links Manager. */
   char * pDroppedTblID,
   char * pExportDir
}
/* ... */
```

API 参数

pRfwdInput

输入。指向 *rfwd_input* 结构的指针。关于此结构的更多信息，参见第139页的『数据结构: RFWD-INPUT』。

pRfwdOutput

输出。指向 *rfwd_output* 结构的指针。关于此结构的更多信息，参见第142页的『数据结构: RFWD-OUTPUT』。

DbAliasLen

输入。一个 2 字节的无符号整数，代表以字节计的数据库别名的长度。

StopTimeLen

输入。一个 2 字节的无符号整数，代表以字节计的停止时间参数的长度。如果不提供停止时间，则设置为 0。

UserNameLen

输入。一个 2 字节的无符号整数，代表以字节计的用户名的长度。如果不提供用户名，则设置为 0。

PasswordLen

输入。一个 2 字节的无符号整数，代表以字节计的密码的长度。如果不提供密码，则设置为 0。

OverflowLogPathLen

输入。一个 2 字节的无符号整数，代表以字节计的溢出日志路径的长度。如果不提供溢出日志路径，则设置为 0。

ReportFileLen

输入。此参数当前不可使用，应设置为 0。

Version

输入。前滚参数的版本标识。定义为 SQLUM_RFWD_VERSION。

pDbAlias

输入。包含了数据库别名的字符串。它是编目在系统数据库目录中的别名。

CallerAction

输入。指定要采取的操作。有效值（在 `sqlutil` 中定义）是:

SQLUM_ROLLFWD

前滚至由 *pPointInTime* 指定的某时间点。对于数据库前滚恢复，数据库将保持前滚暂挂状态。对于表空间级的前滚至某时间点，表空间将保持“前滚进行中”状态。

SQLUM_STOP

结束前滚恢复。不处理新日志记录并备份未落实的事务。将取消数据库或表空间的*前滚暂挂*状态。等效参数是 SQLUM_COMPLETE。

SQLUM_ROLLFWD_STOP
前滚至由 *pPointInTime* 指定的某时间点中，并结束前滚恢复。将停止数据库或表空间的前滚暂挂状态。等效参数是 SQLUM_ROLLFWD_COMPLETE。

SQLUM_QUERY
pNextArcFileName、*pFirstDelArcFileName*、*pLastDelArcFileName* 和 *pLastCommitTime* 的查询值。返回数据库状态和节点号。

SQLUM_PARM_CHECK
验证参数而不执行前滚操作。

SQLUM_CANCEL
取消当前正在运行的前滚操作。数据库或表空间被置于恢复暂挂状态。

注: 当前滚操作实际上正在运行时不能使用此选项。如果操作暂停（即等待 STOP），或在前滚操作期间发生了系统故障，则不能使用该参数。使用时应小心。

前滚数据库可能会需要使用磁带设备的装入恢复操作。如果用户对所需的设备进行干预，前滚 API 会返回一个警告消息。可使用以下 3 个调用程序操作之一来再次调用 API。

SQLUM_LOADREC_CONTINUE
继续使用生成了警告消息的设备（例如在安装了新磁带后）。

SQLUM_LOADREC_DEVICE_TERMINATE
停止使用生成了警告消息的设备（例如没有更多磁带时）。

SQLUM_LOADREC_TERMINATE
终止正由装入恢复使用的所有设备。

pStopTime
输入。包含格式为 ISO 的时间戳记的字符串。超出此时间戳记时，数据库恢复将停止。指定 SQLUM_INFINITY_TIMESTAMP 以尽可能多地前滚。对于 SQLUM_QUERY、SQLUM_PARM_CHECK 以及任何装入恢复 (SQLUM_LOADREC_*xxx*) 调用程序操作可能为 NULL。

pUserName
输入。包含应用程序用户名的字符串。可能为 NULL。

pPassword
输入。包含提供的用户名（如果有的话）密码的字符串。可能为 NULL。

pOverflowLogPath

输入。此参数用于指定要使用的备用日志路径。除活动的日志文件以外，归档日志文件需要（由用户）移动到 *logpath*（请参阅 *Administrative API Reference* 中的 ″sqlfxdb - Get Database Configuration″）才能由此实用程序使用。如果用户在 *logpath* 中没有足够的空间，可能会导致问题。因此提供了溢出日志路径。在前滚恢复期间，将首先在 *logpath* 中，然后在溢出日志路径中搜索需要的日志文件。表空间前滚恢复所需的日志文件可导向 *logpath* 或溢出日志路径。如果调用程序不指定溢出日志路径，缺省值将是 *logpath*。在分区数据库环境中，溢出日志路径必须是有效的全限定路径，而缺省路径是每个节点的缺省溢出日志路径。在单分区环境中，如果服务器是本地的，溢出日志路径可以是相对的。

NumChngLgOvrflw

仅用于 MPP。更改的溢出日志路径数。这些新日志路径只覆盖指定节点的缺省溢出日志路径。

pChngLogOvrflw

仅用于 MPP。指向一个结构的指针，该结构包含已更改的溢出日志路径的全限定名。这些新日志路径只覆盖指定节点的缺省溢出日志路径。

ConnectMode

输入。有效的值（在 `sqlutil` 中定义）是:

SQLUM_OFFLINE

脱机前滚。必须对数据库前滚恢复指定该值。

SQLUM_ONLINE

联机前滚。

pTablespaceList

输入。指向一个结构的指针，该结构包含将要前滚至日志末尾或特定时间点的表空间的名称。如果未指定，将选择需要前滚的表空间。

AllNodeFlag

仅用于 MPP。输入。指示是否将前滚操作应用到 `db2nodes.cfg` 中定义的所有节点。有效的值为:

SQLURF_NODE_LIST

对发送到 *pNodeList* 中的节点列表中的节点应用。

SQLURF_ALL_NODES

应用到所有节点。*pNodeList* 应为 NULL。这是缺省值。

SQLURF_ALL_EXCEPT

对发送到 *pNodeList* 的节点列表中的节点以外的所有节点应用。

SQLURF_CAT_NODE_ONLY
只应用到目录节点。*pNodeList* 应为 NULL。

NumNodes
输入。指定 *pNodeList* 数组中的节点号。

pNodeList
输入。指向要在其上执行前滚恢复的节点号数组的指针。

NumNodeInfo
输入。定义输出参数 *pNodeInfo* 的大小，它必须足够大才能保留正在前滚的每个节点的状态信息。在单分区环境中，此参数应设置为 1。此参数的值应当与正对其调用此 API 的节点号相同。

DlMode
输入。此参数当前不可使用，应设置为 0。

pReportFile
输入。此参数当前不可使用，应设置为 NULL。

pDroppedTblID
输入。字符串，它包含正对其尝试恢复的已删除的表的标识。

pExportDir
输入。删除的表数据将导入至的目录。

pSqlca
输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

REXX API 语法

```
ROLLFORWARD DATABASE database-alias [USING :value] [USER username USING password]
[rollforward_action_clause | load_recovery_action_clause]
where rollforward_action_clause stands for:
    { TO point-in-time [AND STOP] |
      {
        [TO END OF LOGS [AND STOP] | STOP | CANCEL | QUERY STATUS | PARM CHECK }
        [ON {:nodelist | ALL NODES [EXCEPT :nodelist]}]
      }
    }
    [TABLESPACE {ONLINE |:tablespacenames [ONLINE]} ]
    [OVERFLOW LOG PATH default-log-path [:logpaths]]
and load_recovery_action_clause stands for:
    LOAD RECOVERY { CONTINUE | DEVICE_TERMINATE | TERMINATE }
```

REXX API 参数

database-alias
要前滚的数据库的别名。

value 包含输出值的组合 REXX 主机变量。在以下示例中，XXX 代表主机变量名:

XXX.0	变量中的元素号
XXX.1	应用程序标识
XXX.2	从节点接收到的应答数
XXX.2.1.1	第一个节点号
XXX.2.1.2	第一个状态信息
XXX.2.1.3	第一个需要的下一归档文件
XXX.2.1.4	第一个将删除的第一归档文件
XXX.2.1.5	第一个将删除的最后归档文件
XXX.2.1.6	第一个最后落实时间
XXX.2.2.1	第二个节点号
XXX.2.2.2	第二个状态信息
XXX.2.2.3	第二个所需的下一归档文件
XXX.2.2.4	第二个将删除的第一归档文件
XXX.2.2.5	第二个将删除的最后归档文件
XXX.2.2.6	第二个最后落实时间
XXX.2.3.x	以此类推。

username
标识前滚数据库时要使用的用户名。

password
用于认证用户名的密码。

时间点 格式为 ISO 的时间戳记 *yyyy-mm-dd-hh.mm.ss.nnnnnn*（年、月、日、小时、分钟、秒和微秒），以“世界时坐标”(UTC) 表示。

tablespacenames
包含了要前滚的表空间列表的组合 REXX 主机变量。在以下示例中 XXX 是主机变量的名称:

XXX.0	要前滚的表空间号

XXX.1 第一个表空间的名称

XXX.2 第二个表空间的名称

XXX.x 以此类推。

default-log-path

在恢复期间，将在其中搜索归档日志的缺省溢出日志路径。

logpaths

组合 REXX 主机变量，包含了要在恢复期间在其中搜索归档日志的备用日志路径的列表。在以下示例中 XXX 是主机变量的名称:

XXX.0 更改的溢出日志路径号

XXX.1.1 第一个节点

XXX.1.2 第一个溢出日志路径

XXX.2.1 第二个节点

XXX.2.2 第二个溢出日志路径

XXX.3.1 第三个节点

XXX.3.2 第三个溢出日志路径

XXX.x.1 以此类推。

nodelist

包含了节点列表的组合 REXX 主机变量。在以下示例中 XXX 是主机变量的名称:

XXX.0 节点号

XXX.1 第一个节点

XXX.2 第二个节点

XXX.x 以此类推。

数据结构: RFWD-INPUT

此结构用于将信息发送给第131页的『前滚数据库 API』。

表 8. RFWD-INPUT 结构中的字段

字段名称	数据类型	描述
VERSION	sqluint32	前滚版本。
PDBALIAS	Pointer	数据库别名。
CALLERACTION	UNSIGNED SHORT	操作。
PSTOPTIME	Pointer	停止时间。
PUSERNAME	Pointer	用户名。
PPASSWORD	Pointer	密码。
POVERFLOWLOGPATH	Pointer	溢出日志路径。
NUMCHNGLGOVRFLW	UNSIGNED SHORT	已更改的溢出日志路径数（仅 MPP）。
PCHNGLOGOVRFLW	Structure	已更改的溢出日志路径（仅 MPP）。
CONNECTMODE	UNSIGNED SHORT	连接方式。
PTABLESPACELIST	Structure	指向表空间名称列表的指针。关于此结构的信息，参见第87页的『数据结构: SQLU-TABLESPACE-BKRST-LIST』。
ALLNODEFLAG	SHORT	所有节点标志。
NUMNODES	SHORT	节点列表的大小。
PNODELIST	Pointer	节点号列表。
NUMNODEINFO	SHORT	第142页的『数据结构: RFWD-OUTPUT』 中 *pNodeInfo* 的大小。
DLMODE	UNSIGNED SHORT	此参数不是当前使用的。
PREPORTFILE	Pointer	此参数不是当前使用的。
PDROPPEDTBLID	Pointer	该字符串包含了正试图恢复的已删除的表的标识。
PEXPORTDIR	Pointer	已删除的表数据将导出至的目录。
NODENUM	SQL_PDB_NODE_TYPE	节点号。
PATHLEN	UNSIGNED SHORT	新日志路径的长度。
LOGPATH	CHAR(255)	新溢出日志路径。

Data Structure: RFWD-INPUT

语言语法

C 结构

```
/* File: sqlutil.h */
/* Structure: RFWD-INPUT */
/* ... */
SQL_STRUCTURE rfwd_input
{
  sqluint32              version;
  char                   *pDbAlias;
  unsigned short         CallerAction;
  char                   *pStopTime;
  char                   *pUserName;
  char                   *pPassword;
  char                   *pOverflowLogPath;
  unsigned short         NumChngLgOvrflw;
  struct sqlurf_newlogpath *pChngLogOvrflw;
  unsigned short         ConnectMode;
  struct sqlu_tablespace_bkrst_list *pTablespaceList;
  short                  AllNodeFlag;
  short                  NumNodes;
  SQL_PDB_NODE_TYPE      *pNodeList;
  short                  NumNodeInfo;
  unsigned short         DlMode;         /* This parameter is not */
                                         /*   currently used. */
  char                   *pReportFile;   /* This parameter is not */
                                         /*   currently used. */
  char                   *pDroppedTblID;
  char                   *pExportDir;
};
/* ... */


/* File: sqlutil.h */
/* Structure: SQLURF-NEWLOGPATH */
/* ... */
SQL_STRUCTURE sqlurf_newlogpath
{
  SQL_PDB_NODE_TYPE      nodenum;
  unsigned short         pathlen;
  char                   logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};
/* ... */
```

COBOL 结构

```
* File: sqlutil.cbl
01 SQL-RFWD-INPUT.
    05 SQL-VERSION            PIC 9(9) COMP-5.
    05 SQL-DBALIAS            USAGE IS POINTER.
    05 SQL-CALLERACTION       PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-STOPTIME           USAGE IS POINTER.
    05 SQL-USERNAME           USAGE IS POINTER.
    05 SQL-PASSWORD           USAGE IS POINTER.
    05 SQL-OVERFLOWLOGPATH    USAGE IS POINTER.
    05 SQL-NUMCHANGE          PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-CHNG-LOG-OVRFLW  USAGE IS POINTER.
    05 SQL-CONNECTMODE        PIC 9(4) COMP-5.
    05 FILLER                 PIC X(2).
    05 SQL-P-TABLESPACE-LIST  USAGE IS POINTER.
    05 SQL-ALLNODEFLAG        PIC S9(4) COMP-5.
    05 SQL-NUMNODES           PIC S9(4) COMP-5.
    05 SQL-NODELIST           USAGE IS POINTER.
    05 SQL-NUMNODEINFO        PIC S9(4) COMP-5.
    05 SQL-DLMODE             PIC 9(4) COMP-5.  * This parameter is not
                                                *   currently used.
    05 SQL-REPORTFILE         USAGE IS POINTER. * This parameter is not
                                                *   currently used.
    05 SQL-DROPPEDTBLID       USAGE IS POINTER.
    05 SQL-EXPORTDIR       USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQLURF-NEWLOGPATH.
    05 SQL-NODENUM            PIC S9(4) COMP-5.
    05 SQL-PATHLEN            PIC 9(4) COMP-5.
    05 SQL-LOGPATH            PIC X(254).
    05 FILLER                 PIC X.
    05 FILLER                 PIC X(1).
*
```

数据结构: RFWD-OUTPUT

此结构用于从第131页的『前滚数据库 API』中发送信息。

表 9. RFWD-OUTPUT 结构中的字段

字段名称	数据类型	描述
PAPPLICATIONID	Pointer	长度为 SQLU_APPLID_LEN+1（在 sqlutil 中定义）的缓冲区的地址，它用于保留从 API 返回的应用程序标识符。此标识符可与数据库系统监控器 API 一起使用，用于监控应用程序。如果此信息不重要，可指定空指针。在分区数据库环境中，只返回目录节点的应用程序标识符。
PNUMREPLIES	Pointer	接收到的节点应答数。应答的每个节点都填写在 *pNodeInfo* 中的 *sqlurf_info* 结构中。在单个分区环境中，此参数的值是 1。
PNODEINFO	Structure	节点应答信息。*sqlurf_info* 结构中用户定义的的数组 *NumNodeInfo*。

表 10. SQLURF-INFO 结构中的字段

字段名称	数据类型	描述
NODENUM	SQL_PDB_NODE_TYPE	节点号。
STATE	LONG	状态信息。
NEXTARCLOG	UNSIGNED CHAR(13)	一个 12 字节的缓冲区，用于保留下一个需要的归档日志文件的返回名称。如果指定了除 SQLUM_QUERY 之外的调用操作，此字段中返回的值指示访问文件时发生了错误。可能的原因有: • 在数据库日志目录中未找到该文件，或在由溢出日志路径参数指定的路径上未找到该文件。 • 用户出口程序未成功返回归档文件。

表 10. SQLURF-INFO 结构中的字段 (续)

字段名称	数据类型	描述
FIRSTARCDEL	UNSIGNED CHAR(13)	一个 12 字节的缓冲区，用于保留恢复操作不再需要的第一个归档日志文件的返回名称。此文件，以及到 *lastarcdel* 为止（并包括*lastarcdel*）的所有文件都将移动，以节省磁盘上的空间。 例如，如果 *firstarcdel* 和 *lastarcdel* 中返回的值是 S0000001.LOG 和 S0000005.LOG，则可移动以下日志文件: • S0000001.LOG • S0000002.LOG • S0000003.LOG • S0000004.LOG • S0000005.LOG
LASTARCDEL	UNSIGNED CHAR(13)	一个 12 字节的缓冲区，用于保留可从数据库日志目录中除去的最后一个归档日志文件的返回名称。
LASTCOMMIT	UNSIGNED CHAR(27)	一个 26 个字符的字符串，其中包含格式为 ISO 的时间戳记。此值表示在前滚操作终止后，最后一个已落实的事务的时间戳记。

STATE 的可能值（在 sqlutil 中定义）是:

SQLURFQ_NOT_AVAILABLE
无法与节点连接。

SQLURFQ_NOT_RFW_PENDING
数据库不处于前滚暂挂状态。

SQLURFQ_DB_RFW_PENDING
数据库处于前滚暂挂状态。

SQLURFQ_TBL_RFW_PENDING
表空间处于前滚暂挂状态。

SQLURFQ_DB_RFW_IN_PROGRESS
数据库处于“正在前滚”状态。

SQLURFQ_TBL_RFW_IN_PROGRESS
表空间处于“正在前滚”状态。

SQLURFQ_DB_RFW_STOPPING
数据库前滚操作在正在处理 STOP 请求时中断。

SQLURFQ_TBL_RFW_STOPPING
表空间前滚操作在正在处理 STOP 请求时中断。

语言语法

C 结构

```
/* File: sqlutil.h */
/* Structure: RFWD-OUTPUT */
/* ... */
SQL_STRUCTURE rfwd_output
{
  char            *pApplicationId;
  long            *pNumReplies;
  struct sqlurf_info *pNodeInfo;
};
/* ... */
```

```
/* File: sqlutil.h */
/* Structure: SQLURF-INFO */
/* ... */
SQL_STRUCTURE sqlurf_info
{
  SQL_PDB_NODE_TYPE nodenum;
  long            state;
  unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
  unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};
/* ... */
```

COBOL 结构

```
* File: sqlutil.cbl
01 SQL-RFWD-OUTPUT.
    05 SQL-APPLID             USAGE IS POINTER.
    05 SQL-NUMREPLIES         USAGE IS POINTER.
    05 SQL-P-NODE-INFO        USAGE IS POINTER.
*
```

```
* File: sqlutil.cbl
01 SQLURF-INFO.
    05 SQL-NODENUM           PIC S9(4) COMP-5.
    05 FILLER                PIC X(2).
    05 SQL-STATE             PIC S9(9) COMP-5.
    05 SQL-NEXTARCLOG        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-FIRSTARCDEL       PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTARCDEL        PIC X(12).
    05 FILLER                PIC X.
    05 SQL-LASTCOMMIT        PIC X(26).
    05 FILLER                PIC X.
    05 FILLER                PIC X(2).
*
```

前滚会话示例

CLP 示例

示例 1

ROLLFORWARD DATABASE 命令一次允许多个操作的规范，每个规范都由关键字 AND 隔开。例如，要前滚至日志末尾，然后完成，可将以下单独的命令

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

组合为:

```
db2 rollforward db sample to end of logs and complete
```

虽然两种方式是等效的，但建议您以两个步骤来完成此类操作。非常重要的是，应验证前滚操作是否在按预期的方式执行，免得需要停止该操作从而可能导致日志丢失。如果在前滚恢复期间发现了错误的日志，而该错误的日志却被解释为“日志结束”，上述验证尤其重要。此时，可使用该日志的一个未损坏的备份副本来继续对更多的日志进行前滚操作。

示例 2

前滚至日志末尾（已复原了两个表空间）:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

这两个语句是等效的。需要 AND STOP 或 AND COMPLETE 以使表空间前滚恢复到日志末尾。不需要表空间名称。如果未指定的话，将包括所有需要前滚恢复的表空间。如果将只前滚这些表空间的一个子集，则必须指定它们的名称。

示例 3

已复原了 3 个表空间后，将其中一个前滚到日志末尾，另两个前滚到某时间点，全部联机进行:

```
db2 rollforward db sample to end of logs tablespace(TBS1) online

db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

应注意，两个前滚操作不能并行运行。只有在成功地完成了第一个前滚操作后，才能调用第二个命令。

示例 4

复原数据库后，前滚到某时间点，使用 OVERFLOW LOG PATH 来指定用户出口保存归档日志的目录:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

示例 5 (MPP)

有 3 个节点: 0、1 和 2。在所有节点上定义表空间 TBS1，在节点 0 和 2 上定义表空间 TBS2。在节点 1 复原了数据库并在 0 和 2 上复原了 TBS1 后，在节点 1 上前滚数据库:

```
db2 rollforward db sample to end of logs and stop
```

会返回警告 SQL1271（“数据库已恢复，但节点 0 和 2 上的一个或多个表空间已脱机”）。

```
db2 rollforward db sample to end of logs
```

该命令在节点 0 和 2 上前滚 TBS1。在这种情况中，子句 TABLESPACE(TBS1) 是可选的。

示例 6 (MPP)

只在节点 0 和 2 上复原表空间 TBS1 后，在节点 0 和 2 上前滚 TBS1:

```
db2 rollforward db sample to end of logs
```

忽略节点 1。

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

该命令失败，因为 TBS1 未准备好在节点 1 上进行前滚恢复。报告 SQL4906N。

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

成功完成。

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

该命令失败，因为 TBS1 未准备好在节点 1 上进行前滚恢复；必须将所有部分前滚到一起。

注: 随着表空间前滚到某时间点，将不接受节点子句。前滚操作必须在表空间所驻留的所有节点上进行。

在节点 1 上复原 TBS1 后:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS1)
```

成功完成。

示例 7 (MPP)

在所有节点上复原表空间后前滚至 PIT2，但不指定 AND STOP。前滚操作仍在进行中。取消并前滚至 PIT1:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all nodes **

db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

示例 8 (MPP)

前滚恢复 db2nodes.cfg 文件中列出的 8 个节点（3 至 10）上驻留的表空间:

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

前滚恢复至日志末尾（而不是时间点）成功完成。不必指定表空间所驻留的节点。实用程序缺省到 db2nodes.cfg 文件。

示例 9 (MPP)

前滚恢复驻留在单节点节点组（在节点 6 上）上的 6 个小表空间。

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

前滚恢复至日志末尾（而不是时间点）成功完成。

在第397页的『附录F. 恢复 CLP 脚本』中提供了一个样本 DB2 命令脚本以及如何使用它的信息。

API 示例

在第331页的『附录E. 恢复样本程序』中提供了包含 DB2 API 和嵌入式 SQL 调用的样本程序以及如何使用它们的信息。

前滚局限性

前滚实用程序有以下局限性:

- 一次只能调用一个前滚操作。如果有多个要复原的表空间，可在同一操作中指定所有表空间。

- 若在最近的备份操作后有重命名的表空间，应确保在前滚表空间时使用新名称。先前表空间名将不被识别。
- 不能取消正在运行的前滚操作。只能取消已完成的前滚操作，对那些尚未指定 STOP 选项或在完成前已失败的前滚操作则不能取消。
- 指定的时间戳记小于前一时间戳记时，不能继续将表空间前滚至某时间点。如果未指定时间点，将使用前一时间点。可以只指定 STOP 来启动前滚至某时间点的操作，但只有在涉及的表空间都是从同一脱机备份映象复原时才能这样做。在这种情况中，不需要日志处理。如果在正在进行的前滚操作已完成或取消前对另一表空间启动前滚操作，会返回一条错误消息 (SQL4908)。对所有节点调用 LIST TABLESPACES 命令，以确定当前有哪个表空间正在前滚（前滚进行中状态），哪些表空间正准备前滚（前滚暂挂状态）。有 3 种选择:
  - 完成所有表空间上正在进行的前滚操作。
  - 完成表空间子集上正在进行的前滚操作。（如果前滚操作将继续到某特定的时间点，从而需要涉及到所有节点，则可能无法完成表空间子集上正在进行的前滚操作。
  - 取消正在进行的前滚操作。

前滚故障诊断

不要在复原表空间时不取消正在进行的前滚操作；否则您可能会在一个表空间集合中有某些表空间处于“前滚进行中”状态，而另一些表空间则处于“前滚暂挂”状态。正在进行中的前滚操作将只在处于“前滚进行中”状态的表空间上执行。

如果前滚实用程序遇到不可复原的操作，（例如，装入时没有副本），相关的表空间将置于复原暂挂状态。要取消表空间的复原暂挂状态，必须从一个更近期的数据库级或表空间级备份映象进行复原。

返回警告 SQL1271（即使尚未请求 AND STOP），指示有表空间处于前滚暂挂状态或复原暂挂状态。在数据库前滚操作期间，有一个表空间是由前滚实用程序脱机建立的。在*表空间*前滚操作期间，它可以表示有一个正在前滚的表空间是由前滚实用程序脱机建立的，或有另一（未列示的）表空间仍然需要前滚。

如果所有正在前滚的表空间（在列表中指定的或因为它们处于前滚暂挂状态）都是由前滚实用程序脱机建立的，则返回 SQL1272。这可能表示这些表空间中有的表已经过了不可复原的装入操作，或有的表经过 NOT LOGGED INITIALLY 处理。如果返回错误，则应停止表空间前滚操作（即不应再处于进行中）。

第2部分 高可用性

第5章 高可用性和故障转移简介

成功的电子商务取决于事务处理系统不间断的高可用性，而事务处理系统是由数据库管理系统，如 DB2 驱动的，数据库管理系统必须每周 7 天，每天 24 小时可用（即 '24 x 7'）。

高可用性

高可用性 (HA) 是一个术语，用于描述那些可随时对客户运行并可用的系统。为此:

- 必须高效地处理事务，且在高峰操作期内不能有显著的性能下降（甚至是失去了可用性）。在分区数据库环境中，DB2 可同时利用分区内和分区间的并行性来高效地处理事务。*分区内并行性*可用于 SMP 环境中，用来同时处理复杂的 SQL 语句的各种组件。*分区间并行性*在分区数据库环境中，另一方面它指的是在所有分区节点上同时处理一个查询。关于并行性的更多信息，请参阅《*管理指南*》。
- 在发生硬件或软件故障，或出现灾难性的事故时系统必须能够快速恢复。使用日志文件系统可能会有所帮助。*日志文件系统*是一种会自动记录对其结构进行的更改的文件系统。从而确保，只要存储媒体不丢失或毁坏，对该文件系统结构进行的任何更改就可免受系统崩溃的影响。在系统崩溃后，这些日志中记录的更改会以最初记录的次序，应用到文件系统中。日志文件系统按快速恢复的次数划分。它们尤其适用于具有大文件系统的环境，或数据完整性非常重要时。

  可快速恢复的能力关键取决于是否已准备好了一个经过验证的备份与恢复策略。关于恢复策略的更多信息，参见第3页的『第1章 制订一个良好的备份与恢复策略』。
- 增强了企业数据库能力的软件必须持续运行并可用于事务处理。要保持数据库管理器运行，必须确保有另一个数据库管理器可在该数据库管理器发生故障时代替它。这称作故障转移。*故障转移*功能允许在发生硬件故障时，自动将工作负荷从一个系统转移至另一个系统。

在永远前滚日志文件的另一机器上保留您的数据库的一份副本，可达到故障转移保护的效果。*日志交付*是一个将整日志文件复制到备用机器上的进程，可从归档设备复制，也可通过对主数据库运行的用户出口程序复制。通过这种方法，主数据库可使用 DB2 复原实用程序或分割镜像功能，复原到备用机器上。可使用新的暂挂 I/O 支持来快速初始化数据库（参见第155页的『通过联机分割镜像以及暂挂

I/O 支持实现的高可用性』)。备用机器上的辅助数据库将继续前滚日志文件。如果主数据库发生故障，任何剩余的日志文件都将复制到备用机器上。前滚到日志末尾并停止操作后，所有的客户机都将重新连接到备用机器上的辅助数据库。

也可通过您可添加到系统中的特定于平台的软件来提供故障转移支持。例如:

- 高可用性群集多重处理，增强的可缩放性 AIX 版。

  关于 HACMP/ES 的详细信息，参见第159页的『第6章 AIX 上的高可用性』，或标题为 “IBM DB2 Universal Database Enterprise Edition for AIX and HACMP/ES” 的白皮书，可从 “DB2 UDB and DB2 Connect Online Support” 的 Web 站点 (http://www.ibm.com/software/data/pubs/papers/) 获得。

- Microsoft Cluster Server Windows NT 版或 Windows 2000 版。

  关于 MSCS 的详细信息，参见第199页的『第7章 Windows 操作系统上的高可用性』。

- Sun Cluster 或 VERITAS Cluster Server，用于 Solaris 操作环境。

  关于 Sun Cluster 2.x 的信息，参见第229页的『第8章 Solaris 操作环境中的高可用性』；关于 Sun Cluster 3.0 的信息，参见标题为 “DB2 and High Availability on Sun Cluster 3.0” 的白皮书，可从 “DB2 UDB and DB2 Connect Online Support” Web 站点 (http://www.ibm.com/software/data/pubs/papers/) 获得。关于 VERITAS Cluster Server 的信息，参见标题为 “DB2 and High Availability on VERITAS Cluster Server” 的白皮书，也可从 “DB2 UDB and DB2 Connect Online Support” Web 站点获得。

- 惠普公司的 Multi-Computer/ServiceGuard。

  关于 HP MC/ServiceGuard 的详细信息，参见标题为 “IBM DB2 EE v.7.1 Implementation and Certification With Hewlett-Packard’s MC/ServiceGuard High Availability Software” 的白皮书，可从 “DB2 UDB and DB2 Connect Online Support” Web 站点 (http://www.ibm.com/software/data/pubs/papers/) 获得。

故障转移策略通常是基于群集系统的。一个*群集*就是一组互相连接的系统，可一起工作，相当于单个系统。每个处理器都称为该群集中的一个节点。群集技术可将故障服务器上的工作负荷承担过来，从而使服务器可在发生故障时互相备份。

IP 地址接管（或 IP 接管）是一项在服务器当机时，将服务器 IP 地址从一台机器传送到另一台机器的能力；对于客户机应用程序，这两台机器在不同的时候表现为同一服务器。

故障转移软件可能会在系统间使用*心跳监视*或*保持活动信息包*以确认可用性。心跳监视涉及到在一个群集中的所有节点间维护持续通信的系统服务。如果未检测到心跳，就会对备份系统启动故障转移。最终用户通常没有意识到系统已发生故障。

市场上最常见的两种故障转移策略称为*空闲备用*和*相互接管*，虽然取决于各个供应商，与这些术语相关的配置可能会涉及到其他不同的术语。

空闲备用

在这种配置中，使用一个系统来运行 DB2 实例，而第二个处理器处于“空闲”状态或备用方式，准备在第一个系统运行的环境中发生操作系统或硬件故障时接管该实例。因为备用系统一直处于空闲状态直到需要它为止，所以整体系统性能不会受到影响。

相互接管

在此配置中，将每个系统都指定为另一系统的备份。因为备份系统在故障转移后必须做一些额外的工作：它必须做它自己的工作加上那些已前由那个发生了故障的系统完成的工作，所以整体系统性能可能会受到影响。

故障转移策略可用于对实例、分区或多个逻辑节点进行故障转移。

通过联机分割镜像以及暂挂 I/O 支持实现的高可用性

暂挂 I/O 支持通过对联机分割镜像处理提供一套完整的实现方案（即，分割镜像而不关闭数据库），使系统有持续的可用性。*分割镜像*是数据库的一个“瞬间”副本，可通过建立包含数据的磁盘的镜像，然后在需要副本时分割该镜像来实现。*磁盘镜像*是一个将所有数据写到两个单独的硬盘中（一个硬盘是另一个的镜像）的进程。*分割*镜像是建立该镜像的备份副本的进程。

如果您不希望使用 DB2 备份实用程序来备份大数据库，可以通过使用暂挂 I/O 和分割镜像功能，从所镜像的映象中建立副本。此方法还可以:

- 消除来自生产机器的备份操作开销
- 提供了克隆系统的一个快捷方式
- 提供了对空闲备用故障转移的快速实现。不需要初始复原操作，如果证实前滚操作太慢或遇到了错误，重新初始化会非常快。

db2inidb 命令初始化分割镜像，因此可用于:

- 制作克隆数据库

  可使用主数据库的只读克隆，例如创建报告。
- 作为备用数据库
- 作为备份映象

只能对已分割的镜像发出此命令，已分割的镜像必须首先运行 **db2inidb** 才能使用（参见 第283页的『db2inidb - 初始化镜像数据库』）。

在分区数据库环境中，**db2inidb** 命令必须先在每个分区中运行，才能使用来自任何这些分区的分割映象。该工具可对所有分区同时运行。

制作克隆数据库

数据库克隆可提供主（活动）数据库的一种脱机“备份”。但不能备份该克隆数据库、不能将此映象复原到原始系统上也不能在此原始系统上产生的日志文件中前滚。

要克隆一个数据库，应遵循以下步骤:

1. 暂挂主数据库上的 I/O ：

   ```
   db2 set write suspend for database
   ```

2. 使用适当的的操作系统级的命令，从主数据库分割镜像。
3. 恢复主数据库上的 I/O:

   ```
   db2 set write resume for database
   ```

4. 从另一机器上连接到镜像的数据库。
5. 启动数据库实例:

   ```
   db2start
   ```

6. 初始化镜像的数据库，作为主数据库的克隆:

   ```
   db2inidb database_alias as snapshot
   ```

   注: 此命令将回滚执行分割时正在进行中的事务。

使用分割镜像作为备用数据库

随着镜像的（备用）数据库持续在日志中前滚，将不断地从主系统中访存由主数据库创建的新日志。要使用分割镜像作为备用数据库，应遵循这些步骤:

1. 暂挂主数据库上的 I/O ：

   ```
   db2 set write suspend for database
   ```

2. 使用适当的的操作系统级的命令，从主数据库分割镜像。
3. 恢复主数据库上的 I/O:

   ```
   db2 set write resume for database
   ```

4. 将镜像的数据库连接到另一实例。
5. 将镜像的数据库置于前滚暂挂状态:

   ```
   db2inidb database_alias as standby
   ```

   如果有 DMS 表空间（数据库管理的空间），则可以建立一个完整的数据库备份，以抵消在生产数据库上建立备份的开销。

6. 设置用户出口程序，从主系统中检索最新的日志文件。
7. 将数据库前滚到日志末尾。
8. 继续检索日志文件、将数据库前滚至日志末尾，直到主数据库当机。

使用分割镜像作为备份映象

要使用分割镜像作为“备份映象”，应遵循这些步骤:

1. 暂挂主数据库上的 I/O ：

   ```
   db2 set write suspend for database
   ```

2. 使用适当的的操作系统级的命令，从主数据库分割镜像。
3. 恢复主数据库上的 I/O:

   ```
   db2 set write resume for database
   ```

4. 使用操作系统级的命令，复制主系统上的镜像数据和日志。
5. 启动数据库实例:

   ```
   db2start
   ```

6. 将镜像的数据库初始化为“备份映象”，该备份映象可用于将已分割的磁盘上的数据复制回原始系统上的磁盘。（上述复制操作中不要包括包含了日志文件的文件系统，因为在前滚进程中将需要这些日志。）

   ```
   db2inidb database_alias as mirror
   ```

7. 将（原始系统上的）数据库前滚至日志末尾。

第6章 AIX 上的高可用性

“增强的可伸缩性 (ES)” 是 AIX 版 “高可用性群集多重处理方式 (HACMP)” 的一个功能部件。此功能部件提供的故障转移与 HACMP 提供的相同，并与 HACMP 有相同的事件结构，（请参阅 *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*）。增强的可伸缩性还提供了:

- 较大的 HACMP 群集。
- 通过*用户定义事件*扩大错误检测范围。受监控的区域可触发用户定义事件，这些事件各不相同，如一个进程停止或分页空间接近其容量限制。若有必要，这类事件包括可添加至故障转移进程的前事件和后事件。可将专用于其他实现的额外功能放在 HACMP 的前事件和后事件流中。

  规则文件 (/usr/sbin/cluster/events/rules.hacmprd) 包含 HACMP 事件。用户定义事件被添加至此文件。事件发生时要运行的脚本文件是此定义的一部分。

  有关用户定义事件和规则文件的更多信息，参见 第177页的『HACMP ES 事件监控和用户定义事件』。
- HACMP 客户机实用程序，它们用于从 HACMP 群集外的 AIX 物理节点监控和检测（一个或多个群集中的）状态更改。

HACMP ES 群集中的节点交换的消息称为*心跳*或*保持活动的信息包*，每个节点通过此消息通知其他节点有关它的可用性的信息。已停止应答的节点导致该群集中的其余节点调用恢复。该恢复过程称为 *node_down 事件*，也可称为*故障转移*。恢复过程完成后，将节点重新集成到该群集中。这称为 *node_up 事件*。

有两类事件: 一类是在 HACMP ES 的操作内预计的标准事件，另一类是与硬件和软件部件中的参数监控相关的用户定义事件。

其中一个标准事件是 node_down 事件。当计划应执行什么操作来作为恢复过程的一部分时，HACMP 允许两个故障转移选项: 热（或空闲）备用和相互接管。

群集配置

*在热备用*配置中，作为接管节点的 AIX 处理器节点*不*运行任何其他工作负荷。在*相互接管*配置中，作为接管节点的 AIX 处理器节点要运行其他工作负荷。

通常，“DB2 通用数据库扩充企业版” (UDB EEE) 以相互接管方式运行，且每个节点上都有分区。例外的方案有: 目录节点是热备用配置一部分。

当在使用 HACMP ES 的一个 RS/6000 SP 上计划大型 DB2 安装时，需要考虑如何在 RS/6000 SP 大型机内部或之间划分群集的节点。将节点和它的备份置于不同的 SP 大型机中，这样可以在一个大型机停机（即大型机电源／配电板发生故障）时进行接管。但是，发生此类故障的可能性非常小，因为每个 SP 大型机中有 *N*+1 个电源，每个 SP 交换机都有备用电路以及 *N*+1 个风扇和电源。在大型机发生故障的情况下，可能需要人工干预以复原其余的大型机。在 SP Administration Guide 中记载了此恢复过程。HACMP ES 确保了 SP 节点故障的恢复；计算机故障的恢复取决于一个或多个 SP 大型机内部群集的正确布局。

另一个计划的注意事项是如何管理大型群集。管理一个小的群集比管理一个大的群集容易得多；然而，管理一个大的群集又比管理许多小的群集容易得多。在计划过程中，要考虑在群集环境中如何使用应用程序。例如，若有一单个的、大型的、同类应用程序在 16 个节点上运行，则将配置作为单个群集管理比管理 8 个两节点群集可能要容易些。若相同的 16 个节点包含与不同网络、磁盘和节点相关的许多不同的应用程序，则可能最好将这些节点分组到更小的群集中。记住，一次只能将一个节点集成到 HACMP 群集中；启动多个群集的一个配置比启动一个大群集快得多。如果某个节点及其备份在同一群集中，那么 HACMP ES 可同时支持单个和多个群集。

HACMP ES 故障转移允许资源组对物理节点的预定义（也称作*级联*）分配。该故障转移过程还允许资源组对物理节点的浮动（也称作*旋转*）分配。IP 地址和外部磁盘卷组，或文件系统，或 NFS 文件系统，以及每个资源组内的应用服务器指定应用程序或应用程序组件，它们可由 HACMP ES 通过故障转移和重新集成在物理节点间进行操纵。故障转移和重新集成行为由所创建的资源组类型和该资源组中的节点数来指定。

例如，考虑一个 DB2 数据库分区（逻辑节点）。若其日志和表空间容器位于外部磁盘上，并且其他节点与那些磁盘链接，则其他那些节点可能可以访问这些磁盘并重新启动该数据库分区（在接管节点上）。这是 HACMP 自动执行的操作类型。HACMP ES 也可用来复原 DB2 实例主用户目录所使用的 NFS 文件系统。

作为使用 DB2 UDB EEE 来进行恢复的计划的一部分，要仔细阅读 HACMP ES 文档。应阅读 Concepts, Planning, Installation, and Administration Guides，然后为您的环境构建恢复体系结构。对于根据已知故障点来标识的要恢复的每个子系统，标识您需要的 HACMP 群集，以及恢复节点（热备用或相互接管）。这是填写该文档中包括的 HACMP 工作表的起点。

强烈建议在外部磁盘配置中建立磁盘和适配器的镜像。对于为 HACMP 配置的 DB2 物理节点，需要小心确保卷组上的节点可从共享外部磁盘联机。在相互接管配

置中，这种方案需要某些附加计划，以便配对的节点可相互访问对方的卷组而不造成冲突。对于 DB2 UDB EEE，这表示在所有数据库中的所有容器名必须是唯一的。

实现唯一性的一种方式是包括分区号作为名称的一部分。当创建 SMS 或 DMS 容器时，可指定容器字符串语法的节点表达式。当指定表达式时，节点号可以是容器名的一部分，或者，若指定了附加自变量，则那些自变量的结果可以是容器名的一部分。使用自变量 ″ $N″ ([blank]$N) 来指示节点表达式。自变量必须出现在容器字符串的末尾，且只可使用下列形式之一:

表 11. 用于创建容器的自变量. 假定该节点号为 5。

语法	示例	值
[blank]$N	″ $N″	5
[blank]$N+[number]	″ $N+1011″	1016
[blank]$N%[number]	″ $N%3″	2
[blank]$N+[number]%[number]	″ $N+12%13″	4
[blank]$N%[number]+[number]	″ $N%3+20″	22

注:

1. % 是系数。
2. 在所有情况下，从左往右对运算符求值。

下面是如何使用此特殊自变量来创建容器的一些示例:

- 创建在两节点系统上使用的容器。

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE USING
   (device '/dev/rcont $N' 20000)
```

  将使用下列容器:

```
/dev/rcont0   - 在节点 0 上
/dev/rcont1   - 在节点 1 上
```

- 创建在有四个节点的系统上使用的容器。

```
CREATE TABLESPACE TS2 MANAGED BY DATABASE USING
   (file '/DB2/containers/TS2/container $N+100' 10000)
```

  将使用下列容器:

```
/DB2/containers/TS2/container100   - 在节点 0 上
/DB2/containers/TS2/container101   - 在节点 1 上
/DB2/containers/TS2/container102   - 在节点 2 上
/DB2/containers/TS2/container103   - 在节点 3 上
```

- 创建在两节点系统上使用的容器。

```
CREATE TABLESPACE TS3 MANAGED BY SYSTEM USING
   ('/TS3/cont $N%2, '/TS3/cont $N%2+2')
```

将使用下列容器:

```
/TS3/cont0   - 在节点 0 上
/TS3/cont2   - 在节点 0 上
/TS3/cont1   - 在节点 1 上
/TS3/cont3   - 在节点 1 上
```

图18和第163页的图19显示了 DB2 SSA I/O 子系统配置的一个示例，并显示了一个方案的一部分，该方案是确保高度可用的外部磁盘配置并能够访问所有卷组而不发生冲突所必需的。

DB2 SSA I/O 子系统配置 — 没有单个故障点

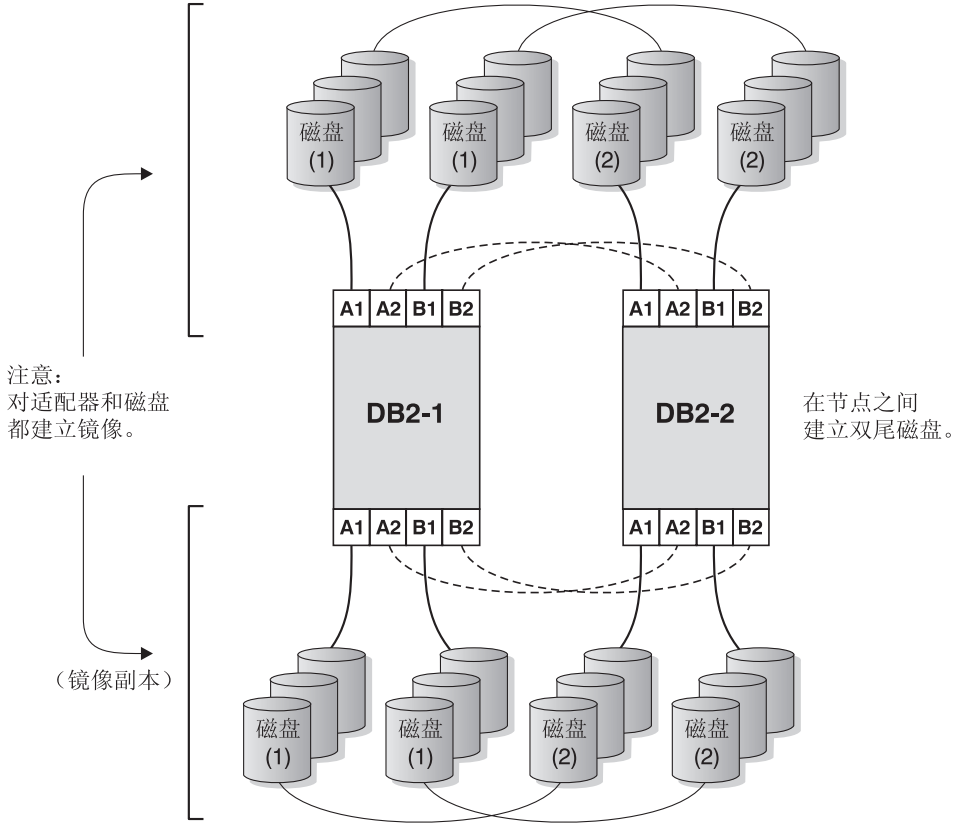


图 18. 没有单个故障点

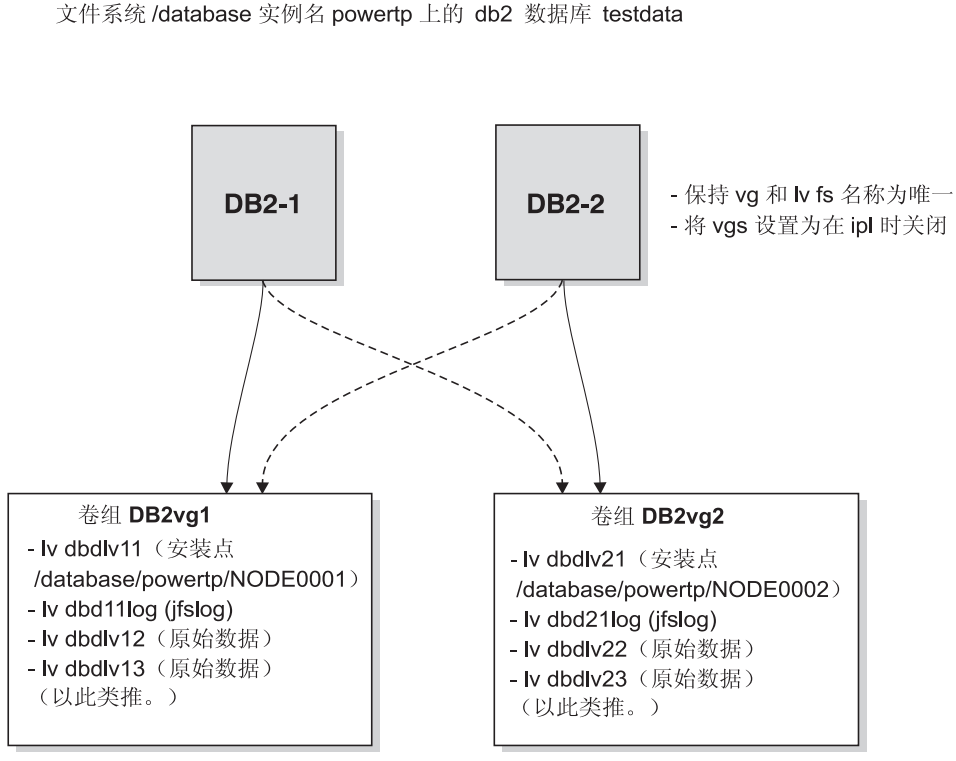


图 19. 卷组和逻辑卷设置

配置 DB2 数据库分区

一旦完成配置，HACMP ES 就启动实例中的每个数据库分区，一次启动一个物理节点。建议使用多个群集来启动多于四个节点的并行 DB2 配置。注意，在 64 节点的并行 DB2 配置中，并行启动 32 个两节点 HACMP 群集比启动四个 16 节点群集要快。

脚本文件 `rc.db2pe` 与 DB2 UDB EEE 封装在一起（并安装在每个节点的 `/usr/bin`）中），以对热备用或相互接管节点中的 HACMP ES 故障转移或恢复进行辅助配置。另外，可在相互接管配置中的故障转移阶段从 `rc.db2pe` 内定制 DB2 缓冲池大小。（当两个数据库分区在一个物理节点上运行时，需要配置缓冲池大小以确保合适的性能。）

当在 DB2 数据库分区的 HACMP 配置中创建应用服务器时，将 rc.db2pe 指定为启动和停止脚本，如下所示:

```
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> start <use switch>
/usr/bin/rc.db2pe <instance> <dpn> <secondary dpn> stop <use switch>
```

其中:

```
<instance> 是实例名。
<dpn> 是数据库分区号。
<secondary dpn> 仅是相互接管配置中的 “伙伴” 数据库分区号;
   在热备用配置中，它与 <dpn> 相同。
<use switch> 通常是空白；当它为空白时，指示
   SP 交换网络用于 db2nodes.cfg 文件中的主机名字段
   （DB2 的所有通信都通过 SP 交换来按路径发送）;
   若不是空白，则使用的名称是要使用的 SP 节点的主机名。
```

从 rc.db2pe 内使用 DB2 命令 LIST DATABASE DIRECTORY，来查找为此数据库分区配置的所有数据库。然后，该脚本文件查找 /usr/bin/reg.parms.*DATABASE* 文件和 /usr/bin/failover.parms.*DATABASE* 文件，其中 *DATABASE* 是为此数据库分区配置的每个数据库。在相互交换配置中，建议您创建参数文件 reg.parms.*xxx* 和 failover.parms.*xxx*。在 failover.parms.*xxx* 文件中，应调整 BUFFPAGE、DBHEAP 和影响缓冲池的任何其他参数的设置，以确保多个缓冲池的可能性。提供了样本文件 reg.parms.SAMPLE 和 failover.parms.SAMPLE 供您使用。

此环境中的其中一个重要参数是 *start_stop_time* 数据库管理程序配置缺省值，它的缺省值为 10 分钟。然而，rc.db2pe 将此参数设置为两分钟。您应通过 rc.db2pe 将此参数的值设置为 10 分钟，或稍微多些。在此上下文中，指定的持续时间是分区的故障与复原之间的时间间隔。若在分区上运行的应用程序频繁地发出 COMMIT，则在数据库分区失效后，10 分钟时间应足够回滚未落实的事务，并达到该分区上数据库的一致性点。若工作负荷很重，或者您拥有许多分区，则可能需要增加持续时间，以降低在回滚操作完成之前发生超时的可能性。

下面是热备用配置和相互交换配置的一个示例。在这两个示例中，资源组都包含一个 “服务 IP” 交换别名地址。此交换别名地址用于:

- 对 DB2 实例所有者文件系统的文件服务器的 NFS 访问
- 在故障转移、TSM（Tivoli Storage Manager，以前称为 ADSM）连接或其他类似的操作情况下需要维护的其他客户机访问。

若您的实现不需要这些别名，则可除去它们。若除去了它们，则要确保在 rc.db2pe 脚本文件中将 *MOUNT_NFS* 参数设置为 NO。

热备用配置示例

此示例假设物理节点 1 与 2 之间存在热备用配置，且 DB2 实例名是 POWERTP。数据库分区是 1，数据库是 TESTDATA，该数据库驻留在文件系统 /database上。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1      (<<< 这是交换别名地址)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 1 stop
```

相互接管配置示例

此示例假设物理节点 1 与 2 之间存在相互接管配置，且 DB2 实例名是 POWERTP。数据库分区是 1 和 2，数据库是 TESTDATA，该数据库驻留在文件系统 /database上。

```
Resource group name: db2_dp_1
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_switch_1      (<<< 这是交换别名地址)
Filesystems: /database/powertp/NODE0001
Volume Groups: DB2vg1
Application Servers: db2_dp1_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 1 2 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 1 2 stop

Resource group name: db2_pd_2
Node Relationship: cascading
Participating nodenames: node2_eth, node1_eth
Service_IP_label: nfs_switch_2      (<<< 这是交换别名地址)
Filesystems: /database/powertp/NODE0002
Volume Groups: DB2vg2
Application Servers: db2_dp2_app
Application Server Start Script: /usr/bin/rc.db2pe powertp 2 1 start
Application Server Stop Script: /usr/bin/rc.db2pe powertp 2 1 stop
```

NFS 服务器节点的配置

还可使用 rc.db2pe 脚本使 DB2 并行实例用户目录的 NFS 安装的目录可用。这可通过在 rc.db2pe 脚本文件中将 *MOUNT_NFS* 参数设置为 YES，并配置 NFS 故障转移服务器对来完成，如下所示:

- 配置该主目录，并使用 /etc/exports 和 **exportfs** 命令将它作为“根”导出至一个 IP 地址，该地址是与 NFS 服务器的 IP 地址相同的子网中的节点上使用的 IP 地址。包括 HACMP 引导地址和服务地址。NFS 服务器的 IP 地址与可

用备份替换的 HACMP 中的服务地址相同。DB2 实例所有者的主目录应是 NFS 直接安装的，而不是自动安装的。（作为 DB2 实例所有者主目录的脚本不支持使用自动安装程序。）

- 通过使用 SMIT 或一个命令行配置，为此文件系统创建一个单独的 /etc/filesystems 项，以便该 DB2 并行分组中的所有节点（包括文件服务器）可使用 NFS 文件系统命令进行安装。

  例如，可将 /nfshome JFS 文件系统作为 /dbhome 导出至所有节点。每个节点都创建一个 NFS 文件系统 /dbname，它是 nfs_server:/nfshome。因此，当实例名为 "powertp" 时，DB2 实例所有者的主目录将是 /dbhome/powertp。

  确保 /etc/filesystems 中用于安装的 NFS 参数是 "hard"、"bg"、"intr" 和 "rw"。

- 确保与 /etc/passwd 中的主目录 /dbhome/powertp 相关的 DB2 实例所有者定义在所有节点上相同。

  在 SP 环境中的用户定义通常是在控制工作站上创建的，并且 "supper" 或 "pcp" 用来将 /etc/passwd、/etc/security/passwd、/etc/security/user 和 /etc/security/group 分发至所有节点。

- 不要在已导出的卷组和文件系统的 HACMP 资源组中配置 "要导出的 nfs_filesystems"。而是以正常方式将它配置给 NFS。NFS 服务器的脚本将控制文件系统的导出。
- 确保文件系统所驻留的卷组的主要编号在主节点和接管节点上是相同的。这通过使用带 **-V** 选项的 importvg 来完成。
- 验证 rc.db2pe 中的 *MOUNT_NFS* 参数是否被设置为 YES，并且每个节点在 /etc/filesystems 中是否安装有 NFS 文件系统。若不是这样，则 rc.db2pe 将不能安装该文件系统并启动 DB2。
- 若已创建了 DB2 实例所有者，并且正将用户的目录结构复制到正创建的文件系统，则确保对该目录使用 **tar** (-cvf) 命令。这确保预留符号链接。
- 不要忘记对所创建的文件系统的逻辑卷和文件系统日志的适配器和磁盘进行镜像。

NFS 服务器接管配置的示例

此示例假定在 IP 地址 "nfs_server" 上的卷组 nfsvg 中，存在一个 NFS 服务器文件系统 /nfshome。该 DB2 实例名是 POWERTP，主目录是 /dbhome/powertp。

```
Resource group name: nfs_server
Node Relationship: cascading
Participating nodenames: node1_eth, node2_eth
Service_IP_label: nfs_server      (<<< 这是交换别名地址)
Filesystems: /nfshome
Volume Groups: nfsvg
```

```
Application Servers: nfs_server_app
Application Server Start Script: /usr/bin/rc.db2pe powertp NFS SERVER start
Application Server Stop Script: /usr/bin/rc.db2pe powertp NFS SERVER stop
```

在此示例中:

- 在安装 nfs_server:/nfshome 时，所有节点上的 /etc/filesystems 都将包含表示 /dbhome 的一项。nfs_server 是一个“服务 IP”交换别名地址。
- nfs_server 节点和备份节点上的 /etc/exports 将包括引导地址和服务地址，并包含表示 /nfsfs -root=nfs_switch_1, nfs_switch_2, ... 的一项。

配置 SP 交换机时的注意事项

当用 SP 交换机实现 HACMP ES 时，考虑下列情况:

- SP 交换机上有“基地址”和“别名地址”。基地址是那些在“SP 系统数据仓库”(SDR) 中定义的地址，并当“引导”系统时由 rc.switch 配置。别名地址是除基地址外、通过带别名属性的 ifconfig 命令配置的、至 **css0** 接口中的 IP 地址。例如:

  ```
  ifconfig css0 inet alias sw_alias_1 up
  ```

- 当配置 DB2 db2nodes.cfg 文件时，应对“主机名”和“网络名”字段使用 SP 交换机“基本”IP 地址名。交换机 IP 地址别名只用来维护 NFS 连接性。使用 **db2start** (RESTART) 命令重新启动 DB2（这会更新 db2nodes.cfg），来完成 DB2 故障转移。
- 不要将交换机地址与 etc/hosts 别名混淆。SP 交换机地址和 SP 交换机别名地址实际存在于 etc/hosts 或 DNS 中。交换机别名地址不是 SP 交换基地址的另一个名称；它们具有各自的独立地址。
- 当一个节点启动时，SP 交换机基地址始终存在于该节点上。HACMP ES 不配置或在节点间移动这些地址。
- 若计划使用 SP 交换机别名地址，则将这些地址作为“心跳”和 IP 地址接管的引导地址和服务地址配置给 HACMP。若不想使用 SP 交换机别名地址，则将基本 SP 交换机地址仅（不进行 IP 地址接管）作为“心跳”的服务地址配置给 HACMP。不要在任何配置中配置别名地址*和*交换机基地址；HACMP ES 不支持此配置。
- 对于 IP 接管配置，仅在节点之间移动 SP 交换机别名地址（而不移动 SP 交换机基地址）。
- 因为每个节点只能有一个 SP 交换机适配器，因此需要 SP 交换机别名。使用别名地址允许一个节点接管另一个节点的交换机别名 IP 地址，而不用添加另一个交换适配器。这在“插槽约束”节点中很有用。有关处理从 SP 交换机适配器故障中进行恢复的更多信息，参见 第180页的『HACMP ES 脚本文件』下的网络故障一节。

- 若为进行 IP 地址接管而配置 SP 交换机，需要为每个节点创建两个额外的别名 IP 地址：一个作为引导地址，另一个作为服务地址。
- 不要忘记在 SP 交换机基本 IP 地址或 SP 交换机别名 IP 地址的 HACMP ES 网络名定义中使用 ″HPS″。
- 当启动 HACMP 时，HACMP 中的 `rc.cluster` 自动在 SP 交换机引导地址中执行 **ifconfig**。除创建 IP 地址和名称并将它们定义给 HACMP 外，不需要附加的配置。
- SP 交换的 Eprimary 节点是实现 Estart、Efence 和 Eunfence 命令的服务器。HACMP 脚本尝试在启动 HACMP 时对一个节点执行 Eunfence 或 Estart，并使该交换可用（若将该交换定义为其网络中的一员）。因此，确保 Eprimary 节点在启动 HACMP 时可用。在 HACMP 节点由于出错而退出之前，它最多等待 12 分钟以便 Eprimary 故障转移完成。
- 由 “SP 并行系统支持程序” (PSSP) 而不是 HACMP 在节点间移动 SP 交换的 Eprimary 节点。若一个 Eprimary 节点脱机，则 PSSP 自动让一个备份节点执行该 Eprimary 节点的功能。该交换网络不受此更改的影响，且保持运行。

DB2 HACMP 配置示例

下列示例举例说明其他的故障转移支持配置，并显示当发生故障时出现的情况。

如果是 DB2 HACMP 相互接管配置（第169页的图20、第170页的图21和第171页的图22）：

- HACMP 适配器以及 SP 交换别名的引导别名和服务别名是为以太网定义的 - 不影响基地址。记住在 HACMP 网络名中使用 ″HPS″ 字符串。
- 通过交换别名，将 `NFS_server/nfshome` 作为 `/dbhome` 安装在所有节点上。
- `db2nodes.cfg` 文件包含 SP 交换基地址。在 DB2 数据库分区（逻辑节点）故障转移后，**db2start** (RESTART) 命令会更改 `db2nodes.cfg` 文件。
- 不显示 SP 交换别名引导地址。
- 节点可存在于不同的 SP 计算机中。

使用 NFS 故障转移的 DB2 HACMP 相互接管 — 正常

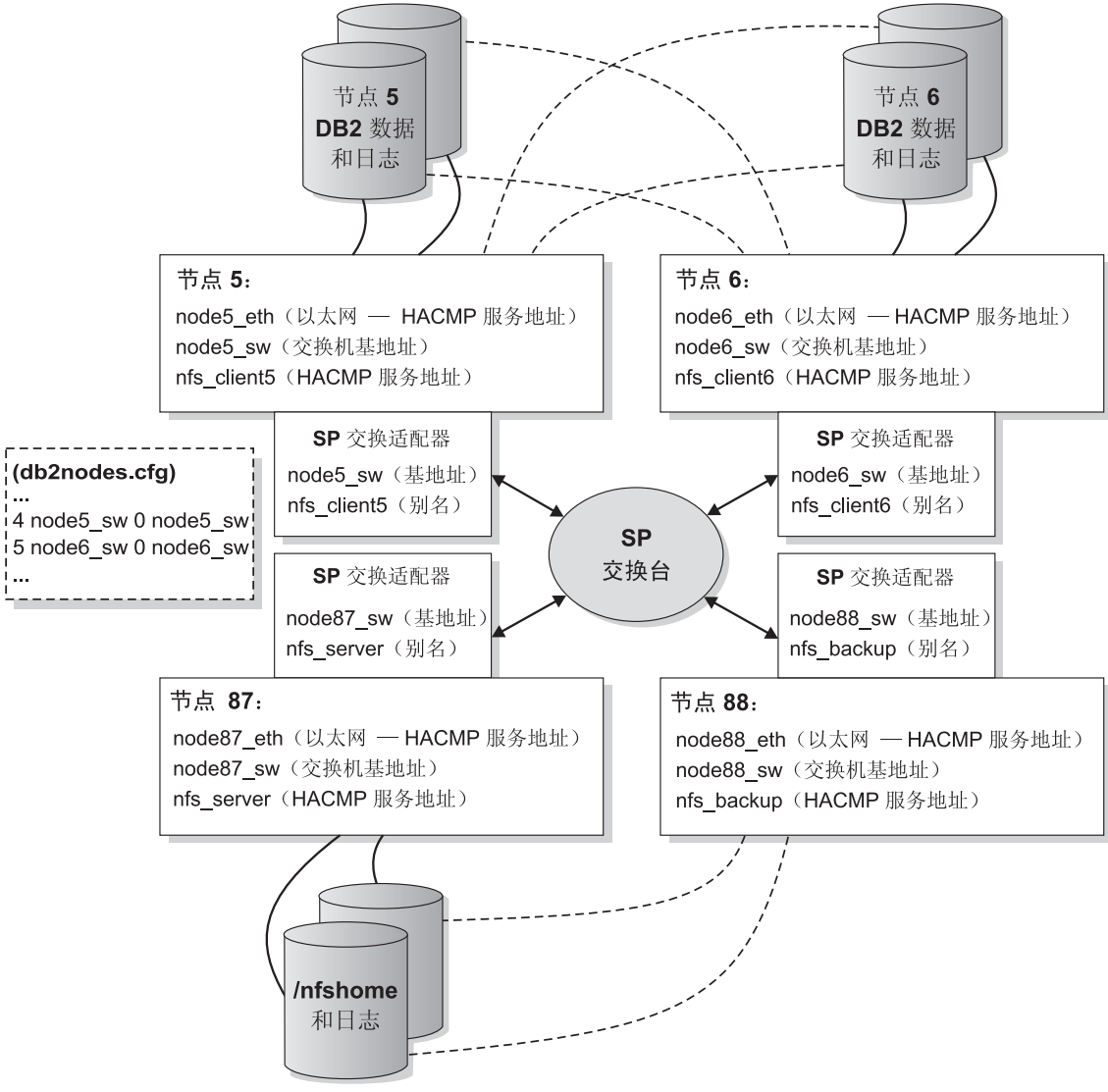


图 20. 使用 NFS 故障转移的相互接管 - 正常

使用 NFS 故障转移的 DB2 HACMP 的相互接管 — NFS 故障转移

— nfs_server SP 交换机别名 IP 地址和 nfs 安装的 /nfshome 从节点 87 移动到节点 88。
— SP 交换机 arp 代码具有通过此移动更新所有交换机 arp 高速缓存的功能。

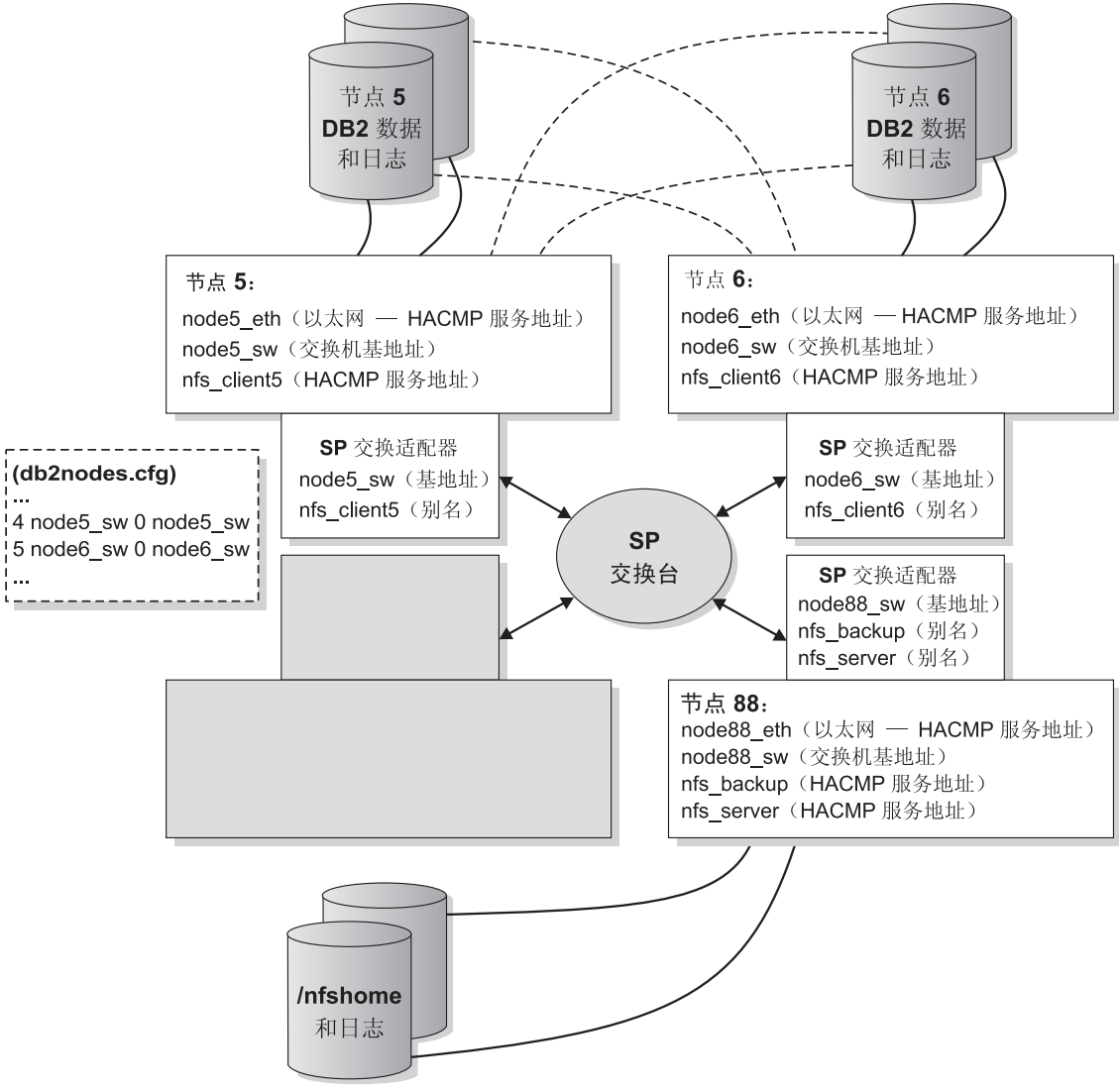


图 21. 使用 NFS 故障转移的相互接管 - NFS 故障转移

使用 NFS 故障转移的 DB2 HACMP 相互接管 — DB2 故障转移

— 交换机 IP 地址接管允许其他服务器（如 ADSM）保持联网。
— 节点 5 运行 DB2 的 2 个逻辑节点。

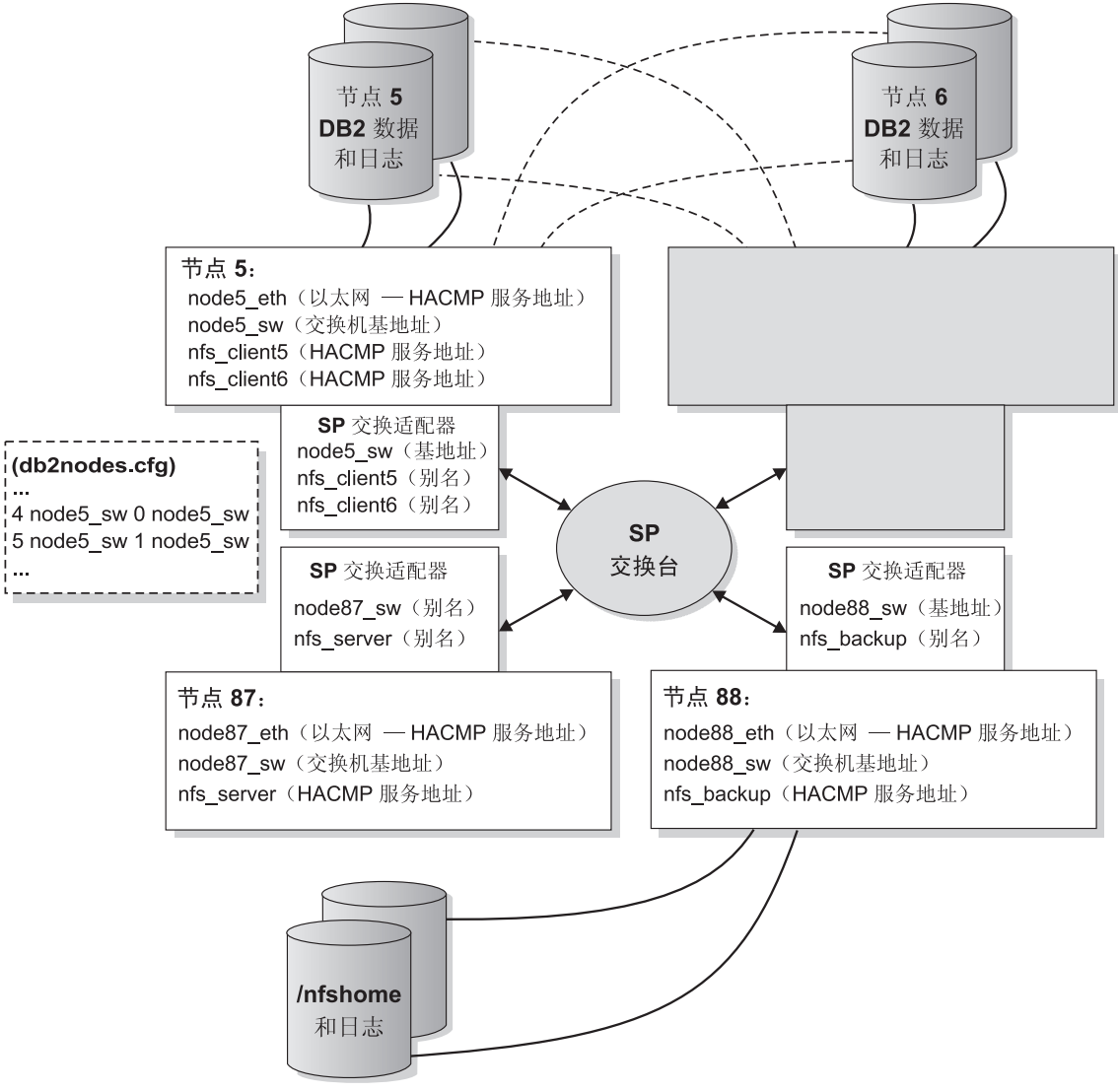


图 22. 使用 NFS 故障转移的相互接管 - DB2 故障转移

如果是 DB2 HACMP 热备用配置（第173页的图23和第174页的图24）：

- HACMP 适配器以及 SP 交换别名的引导别名和服务别名是为以太网定义的 - 不影响基地址。记住在 HACMP 网络名中使用 ″HPS″ 字符串。
- 通过交换别名，将 NFS_server/nfshome 作为 /dbhome 安装在所有节点上。
- db2nodes.cfg 文件包含 SP 交换基地址。在 DB2 数据库分区（逻辑节点）故障转移后，**db2start** (RESTART) 命令会更改 db2nodes.cfg 文件。
- 不显示 SP 交换别名引导地址。

使用 **NFS** 故障转移的 **Db2 HACMP** 热备用 — 正常

注意： 取决于磁盘布线，热备用节点可以备份多个节点。

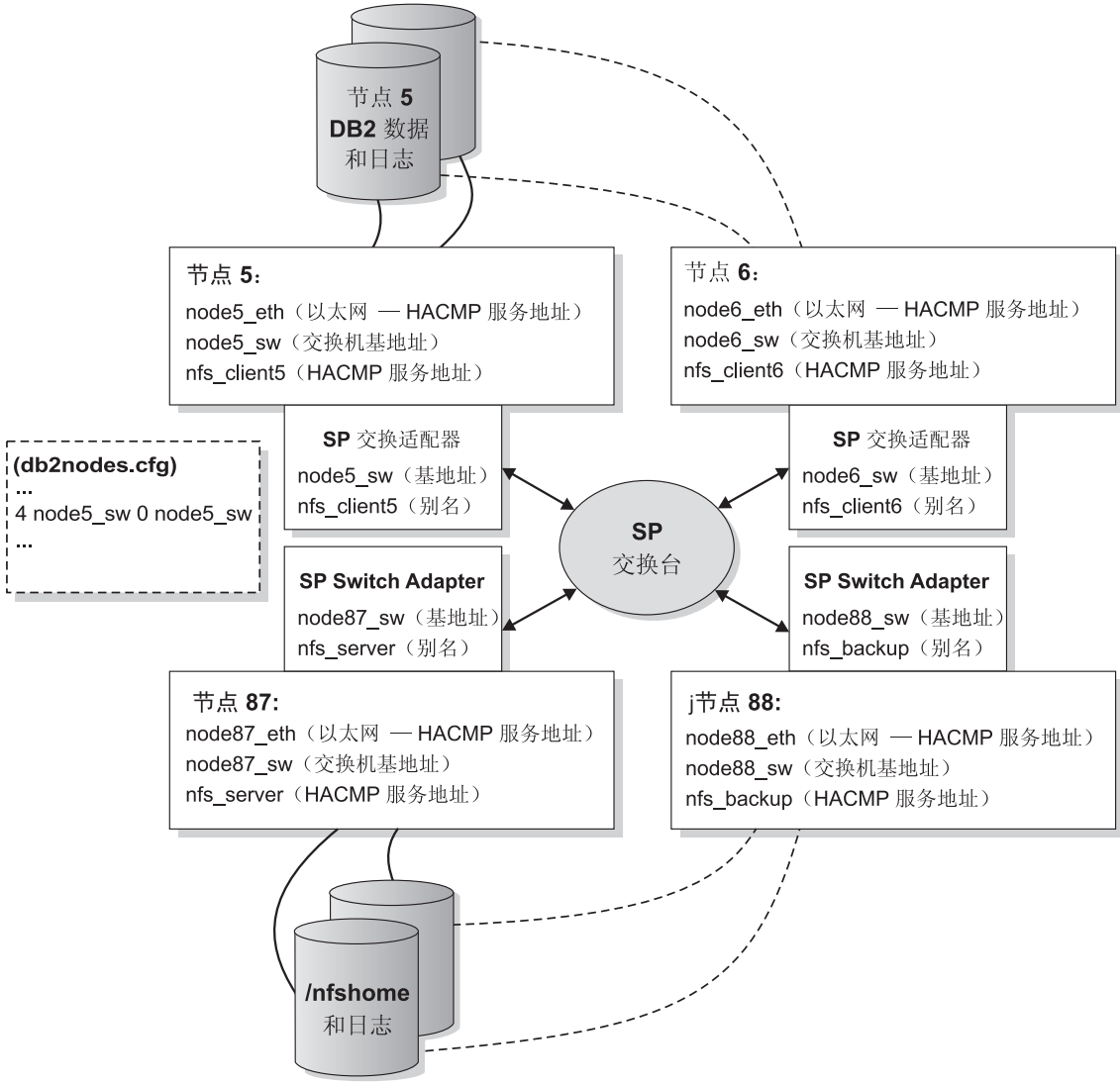


图 23. 使用 NFS 故障转移的热备用 - 正常

使用 NFS 故障转移的 DB2 HACMP 热备用 — DB2 故障转移

注意：取决于磁盘布线，热备用节点可备份多个节点。

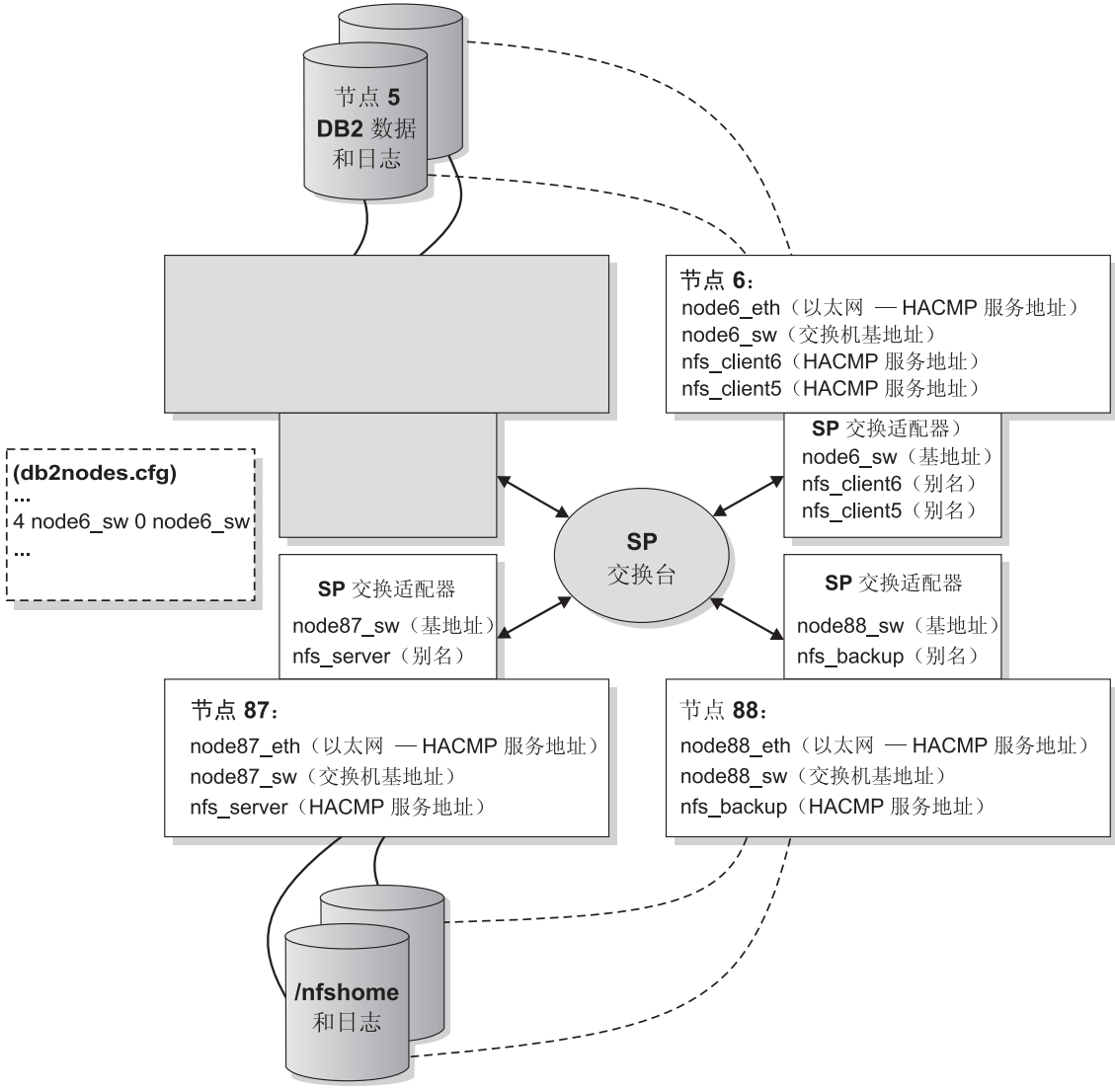


图 24. 使用 NFS 故障转移的热备用 - DB2 故障转移

如果是不带 NFS 故障转移的 DB2 HACMP 相互接管配置（第175页的图25和第176页的图26）：

- HACMP 适配器和 SP 交换基地址是为以太网定义的。记住，当将基地址作为服务地址配置给 HACMP 时，不存在引导地址（只有“心跳”）。不要忘记在 SP 交换的 HACMP 网络名中使用 ″HPS″ 字符串。
- db2nodes.cfg 文件包含 SP 交换基地址。在 DB2 数据库分区（逻辑节点）故障转移后，**db2start** (RESTART) 命令会更改 db2nodes.cfg 文件。
- 不显示 NFS 故障转移功能。
- 节点可存在于不同的 SP 计算机中。

不用 **NFS** 故障转移的 **DB2 HACMP** 相互接管 — 正常

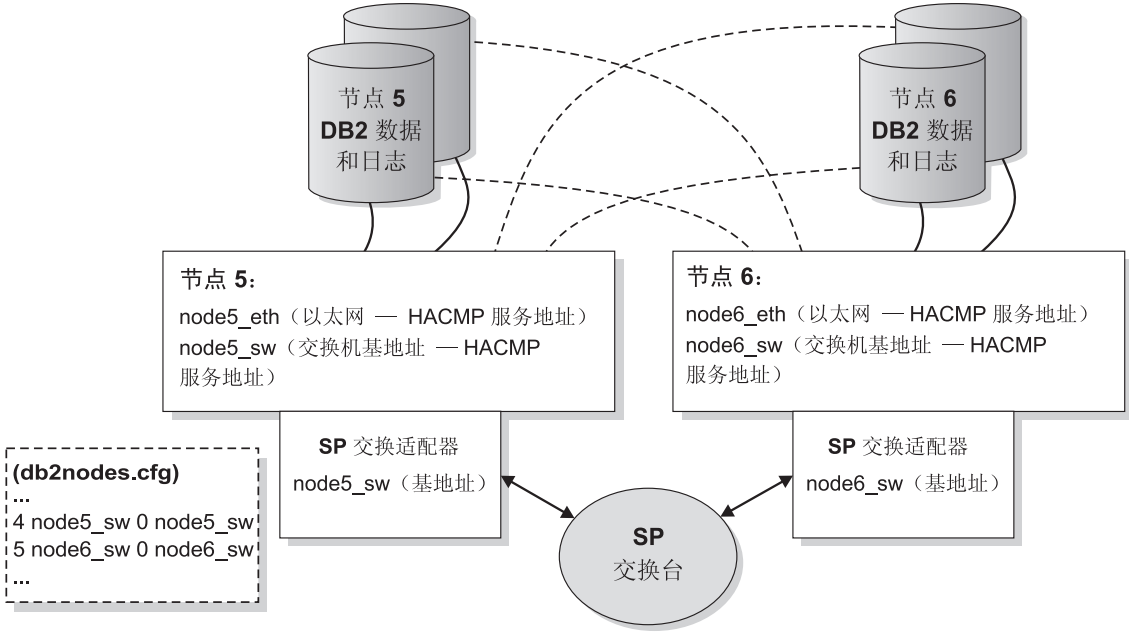


图 25. 不用 NFS 故障转移的相互接管 - 正常

不用 **NFS** 故障转移的 **DB2 HACMP** 相互接管 — **DB2** 故障转移

— 节点 5 运行 DB2 的 2 个逻辑节点。

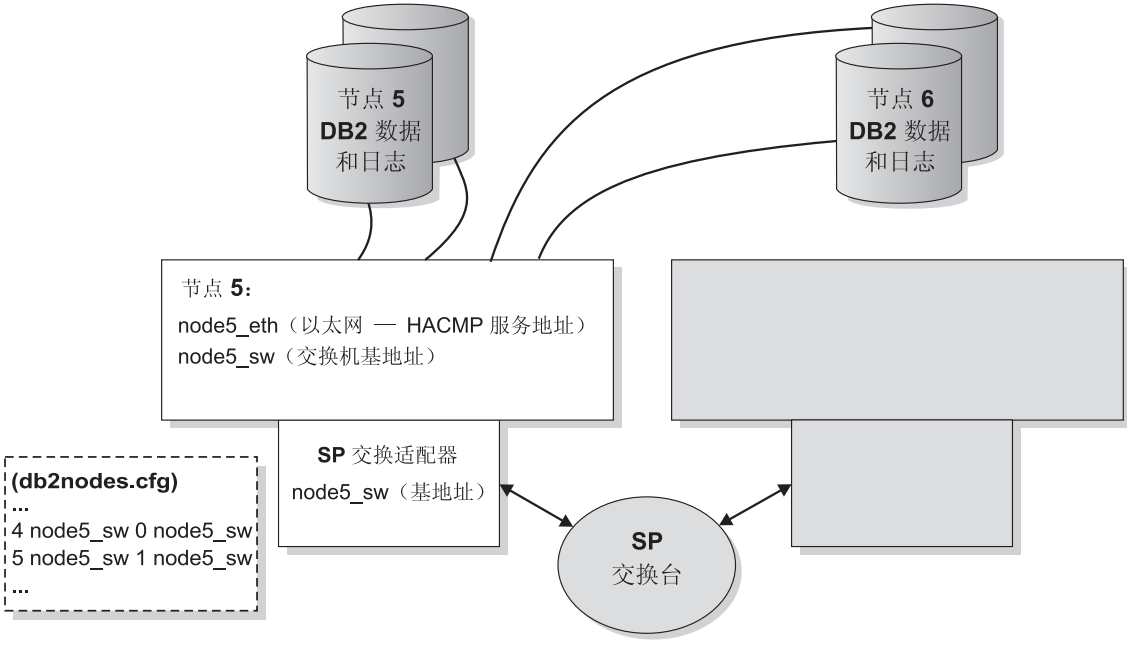


图 26. 不用 NFS 故障转移的相互接管 - DB2 故障转移

DB2 HACMP 启动建议

建议不要在 /etc/inittab 中指定在引导时启动 HACMP。应在引导这些节点后人工启动 HACMP。这允许对失效的节点进行非中断维护。

以“中断维护”为例，假定一个节点有硬件故障并已崩溃。HACMP 自动启动故障转移，复原成功完成。然而，需要修正失效的节点。若 /etc/inittab 中将 HACMP 配置为在重新引导时启动，则引导完成后，此节点将尝试重新集成，在此情况下，这并不是所期望的。

对于“非中断维护”，假定在每个节点上人工启动 HACMP。借此，可修正和重新集成失效节点，而不会影响其他节点。提供 ha_cmd 脚本，以便从控制工作站控制 HACMP 启动和停止命令。

注: 第一次创建 DB2 实例时，下一项被追加至 /etc/inittab 文件:

```
rcdb2:2:once:/etc/rc.db2 > /dev/console 2>&1 # Autostart DB2 Services
```

若启用 HACMP 或 HACMP ES，则通过在 HACMP 项前面加上上面这一行来更新 /etc/inittab 文件。下面是 /etc/inittab 文件中的一个样本 HACMP 项:

```
clinit:a:wait:touch /usr/sbin/cluster/.telinit # HACMP for AIX
```

此项必须是 /etc/inittab 文件中的最后一项。

HACMP ES 事件监控和用户定义事件

下面是用户定义事件的两个示例: 在调页空间达到某一填充百分比时关闭 AIX 物理节点上的一个 DB2 数据库分区，并重新启动 DB2 数据库分区，或在某进程在给定节点上结束时启动故障转移操作。可以在 samples 子目录中找到说明用户定义事件的示例，如关闭数据库分区并强制事务异常终止来释放调页空间。

规则文件 /user/sbin/cluster/events/rules.hacmprd 包含 HACMP 事件。此文件中的每个事件描述都有以下 9 个部件:

- 事件名，它必须是唯一的。
- 状态，或事件的限定符。事件名和状态是规则触发器。仅当 “HACMP ES 群集管理器” 找到具有与该事件名和状态对应的触发器的规则时，它才启动恢复。
- 资源程序路径，这是包含恢复程序的 *xxx*.rp 文件的完整路径说明。
- 恢复类型。它被保留以备后用。
- 恢复级别。它被保留以备后用。
- 资源变量名，它用于 “事件管理器” 事件。
- 实例向量，它用于 “事件管理器” 事件。这是一组格式为 “名称=值” 的元素。这些值唯一地标识系统中该资源的副本，并用扩展名唯一标识该资源变量的副本。
- 谓词，它用于 “事件管理器” 事件。这是资源变量与其他元素之间的关系表达式。当此表达式为真时，“事件管理” 子系统生成一个事件以通知 “群集管理器” 和适当的应用程序。
- Rearm 谓词，它用于 “事件管理器” 事件。这是用来生成改变主谓词状态的事件的谓词。此谓词一般与主要谓词相反。它也可与该事件谓词一起使用来为您感兴趣的条件建立上限和下限。

每个对象在事件定义中都需要一行，即使不使用该行。若除去了这些行，则 “HACMP ES 群集管理器” 不能正确地对该事件定义进行语法分析，这可能会导致系统挂起。将任何以 ″#″ 开始的行视为注释行。

注: 该规则文件要求每个事件定义只能有九行（未计算任何注释行）。当在该规则文件的底部添加一个用户定义事件时，除去该文件末尾不需要的空行很重要，否则该节点将挂起。

下面是 node_up 的事件定义的一个示例:

```
##### Beginning of the Event Definition: node_up
#
TE_JOIN_NODE
0
/usr/sbin/cluster/events/node_up.rp
2
0
# 6) Resource variable - only used for event management events

# 7) Instance vector - only used for event management events

# 8) Predicate  - only used for event management events

# 9) Rearm predicate - only used for event management events

###### End of the Event Definition: node_up
```

此示例仅是可以在 rules.hacmprd 文件中找到的其中一个事件定义。在此示例中，当发生 node_up 事件时，将调用恢复程序 /usr/sbin/cluster/events/node_up.rp。对状态、恢复类型和恢复级别指定了值。有四个空行表示: 资源变量、实例变量、谓词和 rearm 谓词。

您可定义其他事件，以便对非标准 HACMP ES 事件作出反应。例如，要定义 /tmp 文件系统超过 90% 负载的事件，必须修改 rules.hacmprd 文件

在 “IBM 并行系统支持程序” (PSSP) 中预定义了许多事件。这些事件可被利用（当在用户定义事件内使用时），如下所示:

1. 停止该群集。
2. 编辑 rules.hacmprd 文件。在修改该文件之前备份它。人工添加预定义的 PSSP 事件。若需要在该群集中的所有节点将点同步化，则在恢复程序中使用 **barrier** 命令。（在 HACMP Concepts, Installation, and Administration Guides 中有关于 barrier 命令和恢复程序的同步化的更多信息。）
3. 重新启动该群集。当启动 “群集管理器” 时，将 rules.hacmprd 文件存储在内存中。要准确地实现这些更改，重新启动所有的群集。群集中不应存在任何不兼容的规则。
4. “群集管理器” 使用 rules.hacmprd 文件中的所有事件。

HACMP ES 使用 PSSP 事件检测以处理用户定义事件。 “PSSP 事件管理” 子系统通过监控不同硬件和软件资源来提供综合的事件检测。

资源状态由资源变量表示。资源条件表示为称为谓词的表达式。

“事件管理”从“资源监控程序”接收资源变量，该监控程序观察特定系统资源的状态，并将此状态转换为几个资源变量。定期将这些变量传送至“事件管理”。“事件管理”应用“HACMP ES 群集管理器”在 rules.hacmprd 中对每个资源变量指定的谓词。当评估该谓词为真时，会生成一个事件并将该事件发送至“群集管理器”。“群集管理器”启动该表决协议，并在由恢复程序中的“节点集”指定的一组节点上（根据事件优先级）运行该恢复程序文件 (xxx.rp)。

该恢复程序文件 (*xxx*.rp) 由一个或多个恢复程序行组成。每一行以下列格式说明:

```
relationship     command_to_run     expected_status     NULL
```

在该行中，每个值之间必须至少有一个空格。″Relationship″ 是一个值，用来决定哪个程序应在哪种节点上运行。支持三种关系:

- 全部。在当前 HACMP 群集的所有节点上运行指定的命令或程序。
- 事件。仅在发生该事件的节点上运行指定的命令或程序。
- 其他。在所有未发生该事件的节点上运行指定的命令或程序。

″Command_to_run″ 是一个使用引号定界的字符串，带或不带可执行程序的完整路径说明。仅有 HACMP 传递的事件脚本才可使用相对路径定义。其他脚本或程序必须使用完整路径说明，即使它们与 HACMP 事件脚本位于相同的目录中。

″Expected_states″ 是指定的命令或程序的返回码。它或者是整数值，或者是 ″x″。若使用 ″x″，则表示“群集管理器”不关心该返回码。所有其他代码都必须与期望的返回码相等，否则“群集管理器”将检测到事件故障。此事件的处理“挂起”该进程，直到发生恢复（通过人工干预）为止。若不进行人工干预，该节点就不能与其他节点同步。所有节点同步是“群集管理器”控制所有节点的必要条件。

″NULL″ 是被保留以备后用的字段。单词 ″NULL″ 必须出现在每行（障碍行除外）的末尾。若在两个障碍命令之间或在第一个障碍命令之前指定多个恢复命令，则在该节点自身上和节点之间以并行方式运行这些恢复命令。

该障碍命令用来将所有群集节点中的所有命令同步。当一个节点命中该恢复程序中的障碍语句时，“群集管理器”启动此节点上的障碍协议。因为障碍协议是一个两阶段协议，当所有节点都已遇到该恢复程序中的障碍，并且已“表决”赞成该协议时，则通知所有节点两阶段已完成。

此过程概述如下:

1. “组服务／ES”（对于预定义事件）或“事件管理”（对于用户定义事件）将该事件通知“HACMP ES 群集管理器”。
2. “群集管理器”读取 rules.hacmprd 文件并确定映射至该事件的恢复程序。

3. “群集管理器”运行由一系列恢复命令组成的恢复程序。
4. 恢复程序执行可能是 shell 脚本或二进制命令的恢复命令。(在 HACMP AIX 版中，恢复命令与 HACMP 事件脚本相同。)
5. “群集管理器”从恢复命令接收返回状态。一个意外的状态会“挂起”该群集，直到进行人工干预(使用 smit cm_rec_aids 或 /usr/sbin/cluster/utilities/clruncmd 命令)为止。

HACMP ES 脚本文件

DB2 UDB EEE 附带包括了以下用于故障转移和用户定义事件的样本脚本。这些脚本文件位于 $INSTNAME/sqllib/samples/hacmp/es 目录中。这些脚本将“按原样”运行，您也可定制恢复操作。

- DB2 数据库分区恢复脚本 rc.db2pe。这是用于启动和停止数据库分区中的 HACMP 配置的脚本文件。它还充当 DB2 实例所有者的 NFS 服务器的 HACMP 启动和停止脚本。
- HACMP ES 的 DB2 专用的用户定义事件。包括六个缺省事件: 一个用于进程恢复，两个用于调页空间，三个用于 NFS 和自动安装程序恢复。
- DB2 实例 NFS 文件服务器故障转移。此脚本提供 DB2 实例的文件系统服务器至备份的故障转移。
- 网络故障转移。脚本 network_up_complete、network_back 、network_down_complete 和 network_down 允许 SP DB2 数据库分区在其 SP 交换适配器失效时执行故障转移。
- 定义“SP GUI 透视”的监控事件的脚本。通过“事件和硬件透视”，可以监控故障转移和用户定义复原。阅读“PSSP 管理”的文档，以了解有关“透视”的更多信息。
- 安装和除去 HACMP ES 节点上的核心脚本和事件的安装脚本。
- 脚本文件，用来创建和除去用于监控 HACMP 和 DB2 配置的 “SP 透视”问题管理 (pman) 资源。

必须在将运行恢复操作的每个节点上安装该恢复脚本。脚本文件可从 SP 控制工作站或其他指定的 SP 节点集中安装:

1. 将这些脚本从 $INSTNAME/sqllib/samples/hacmp/es 目录复制到可运行 **pcp** 和 **pexec** 命令的 SP 控制工作站或另一个 SP 节点。这些命令是安装操作所必需的。
2. 通过为故障转移配置设置关键字参数(如 BUFFPAGE)来为环境定制 reg.parms.SAMPLE 和 failover.parms.SAMPLE 文件。通常情况下，对于相互接管配置，会将故障设置调低至常规设置大小的一半或更小。而且，将使用用自己的名称重命名的这些文件的副本(取代 ″SAMPLE″)。

3. 定制 rc.db2pe 文件中的 5 个参数: NFS_RETRIES、START_RETRIES、MOUNT_NFS、STOP_RETRIES 和 FAILOVER（如有必要的话）。重试和故障转移设置应能满足大多数实现的需要。应根据您是否将使用 NFS 服务器可用性软件包来配置 MOUNT_NFS 设置。若希望 rc.db2pe 为您安装并验证 DB2 实例所有者的 NFS 主目录，应指定此设置。将 FAILOVER 参数设置为 "YES" 将调用 db2_proc_restart，并尝试重新启动 DB2 数据库分区。若重新启动操作不成功，则 HACMP 将关闭，并进行故障转移。
4. 定制事件文件中的 db2_paging_action、db2_proc_recovery 和 nfs_auto_recovery。编辑 pwq，将它更改为 DB2 实例所有者。定制 db2_paging_action，以指定当调页空间填充程度超过 90% 时要执行的操作。（若真的发生此情况，则停止 DB2 数据库分区。）若需要附加的恢复操作，修改该脚本。
5. 使用 db2_inst_ha 在指定的节点上安装这些脚本和事件。（在开始之前，必须在这些节点上预安装 HACMP ES。）db2_inst_ha 的语法是:

   ```
   db2_inst_ha $INSTNAME/sqllib/samples/hacmp/es <nodelist> <DATABASENAME>

   其中,
      $INSTNAME/sqllib/samples/hacmp/es 是脚本与事件所在的目录
      <nodelist> 是节点的 pcp 或 pexec 样式; 例如
         1-16 或 1,2,3,4
      <DATABASENAME> 是常规和故障转移参数文件的数据库名。
   ```

   将 reg.parms.SAMPLE 和 failover.parms.SAMPLE 文件复制到每个节点并重命名为 reg.parms.DATABASENAME。 db2_inst_ha 将文件复制至 /usr/bin 中的每个节点，并更新 HACMP 事件文件:

   ```
   /usr/sbin/cluster/events/rules.hacmprd
   /usr/sbin/cluster/events/network_up_complete
   /usr/sbin/cluster/events/network_down_complete
   ```

6. 用 HACMP 配置系统和脚本。
7. 使用 **create_db2_events** 命令为问题管理资源 (pman) 和 “SP GUI 透视” 安装监控事件。 “透视” 中需要附加的配置和定制。有关 “透视” 的更多信息，阅读 PSSP Administration Guide。
8. 使用 ha_db2stop 命令，可关闭数据库分区，而无需进行 HACMP ES 故障转移。要使用此命令，将该文件复制到数据库用户的主目录中，并确保为该用户设置了许可权和拥有权。要停止数据库而不进行故障转移，则可作为该用户来输入:

   ```
   ha_db2stop
   ```

   注: 必须等待该命令返回。使用 ctrl-C 中断或通过停止进程来退出，可能会导致过早地再次启用故障转移，并且，某些数据库分区可能不停止。

使用 HACMP ES 执行 DB2 恢复脚本操作

HACMP ES 以下列方式调用 DB2 恢复脚本:

- node_up_local （启动节点）

  HACMP 运行 node_up 序列，以获取在此节点拥有的（通过级联）或赋予此节点的（通过旋转）资源组中指定的卷组、逻辑卷、文件系统和 IP 地址。

  当运行 node_up_local_complete 时，启动包含 rc.db2pe 的应用服务器定义，以启动在此物理节点上的应用服务器定义中指定的数据库分区。

  注: 当 rc.db2pe 以启动方式运行时，它调整在 reg.parms.DATABASE 中对与参数 (parms) 文件相匹配的数据库目录中的每个 DATABASE 指定的 DB2 参数。

  每个节点在启动时都遵循此序列。若有多个 HACMP 群集且并行启动它们，则一次启动多个节点。

- node_down_remote（故障转移）

  HACMP 获取在指定的接管节点上的资源组中指定的卷组、逻辑卷、文件系统和 IP 地址。

  当运行 node_down_remote_complete 时，HACMP 将把 rc.db2pe 作为在此数据库分区的资源组中指定的应用服务器启动脚本来运行。

  注: 当 rc.db2pe 以相互接管方式运行时，它停止在其上运行的 DB2 数据库分区，调整 failover.parms.DATABASE 中对与参数 (parms) 文件相匹配的数据库目录中的每个数据库指定的 DB2 参数，然后启动该物理替换节点上的两个数据库分区。

- node_up_remote（重新集成失效节点 - 级联相互接管资源组）

  当在旧的接管节点上运行 node_up_remote 时，该应用服务器定义导致 rc.db2pe 以停止方式运行。

  注: 当 rc.db2pe 以重新集成方式（相互接管）运行时，它停止在其上运行的两个数据库分区，调整在 failover.parms.DATABASE 中对与参数 (parms) 文件相匹配的数据库目录中的每个数据库指定的 DB2 参数，然后只启动此物理替换节点上要保留的数据库分区。

  旧的接管节点释放在重新集成节点要拥有的资源组中指定的卷组、逻辑卷、文件系统和 IP 地址。

  HACMP 重新获取在重新集成节点现在拥有的资源组中指定的卷组、逻辑卷、文件系统和 IP 地址。

当运行 node_up_local_complete 时，启动包含 rc.db2pe 的应用服务器定义，以启动在此重新集成的物理节点上的应用服务器定义中指定的 DB2 数据库分区。

注: 当 rc.db2pe 以启动方式运行时，它调整在 reg.parms.DATABASE 中对与参数 (parms) 文件相匹配的数据库目录中的每个 DATABASE 指定的 DB2 参数。

- node_down_local（节点停止或由于接管停止）

  当在正在停止的节点上运行 node_down_local 时，该应用服务器定义导致 rc.db2pe 以停止方式运行。

  注: 当 rc.db2pe 以停止方式运行时，它调整 failover.parms.DATABASE 中对与参数 (parms) 文件相匹配的数据库目录中的每个 DATABASE 指定的 DB2 参数，然后停止该数据库分区（这用于接管）。

  HACMP 释放在该节点现在拥有的资源组中指定的卷组、逻辑卷、文件系统和 IP 地址。

- db2_proc_recovery（db2 进程停止）

  所有节点都运行 db2_proc_restart 脚本。发生故障的节点重新启动正确的 DB2 数据库分区。

- db2_paging_recovery（调页空间复原）

  所有节点都运行 db2_paging_action 脚本。若一个节点的调页空间被填充了 70% 以上，则发出 wall 命令。若一个节点的调页空间被填充了 90% 以上，则停止并重新启动此物理节点上的 DB2 数据库分区。

- nfs_auto_recovery（nfs 或自动安装进程失败）

  所有节点都以 NFS 方式运行 rc.db2pe 脚本。若一个 NFS 进程停止运行，则重新启动它。类似地，若自动安装过程停止运行，则重新启动它。

- network_down_complete（网络故障 - SP 交换）

  调用 net_down 脚本。这验证该网络是否为 SP 交换网络，并验证它是否已关闭。若是这样，则它等待用户定义的时间间隔。缺省时间间隔是 100 秒。

  若该 SP 交换网络重新启动（由 network_up_complete 事件指示），则没有恢复生效。

  若达到时间限制，则用故障转移停止 HACMP。

注: 可通过 SP 问题管理和 “SP 透视 GUI” 监控所有事件。

其他脚本实用程序

您还可使用其他脚本实用程序，包括:

- ha_cmd，这是为了从控制工作站启动 SP 节点上的 HACMP 而提供的命令。语法是:

  ```
  ha_cmd <noderange> <START|STOP|TAKE|FORCE>

  其中,
     <noderange> 是 SP 节点范围的 pcp 或 pexec 式样。
        例如, "ha_cmd 3-6 START" 将在节点 3、4、5 和 6 上启动 HACMP。
              "ha_cmd 5 TAKE" 将在节点 5 上为相互接管关闭 HACMP。
  ```

- ha_mon，这是用于从 SP 控制工作站监控 HACMP hacmp_out 文件的命令。语法是:

  ```
  ha_mon <node>

  其中,
     <node> 是要监控的 SP 节点。
     ha_mon 将对您指定的节点上的 /tmp/hacmp.out 文件执行 "tail -f" 操作。
  ```

- db2_turnoff_recov，这是用于临时禁用所有 HACMP 复原（非故障转移）的命令，它是专门为极为罕见的情况设计的。不启动 DB2 进程、调页、NFS 和自动安装程序恢复。此功能从 HACMP 规则文件中除去该恢复的事件节。必须停止并重新启动 HACMP。语法是:

  ```
  db2_turnoff_recov <nodelist>
  ```

- db2_turnon_recov，这是用于重新启用 HACMP（非故障转移）复原的命令。此命令在 db2_turnoff_recov 后使用，以复原 HACMP 规则文件，这样可恢复用户定义事件。必须停止并重新启动 HACMP。语法是:

  ```
  db2_turnon_recov <nodelist>
  ```

监控 HACMP 群集

除了在 HACMP ES 中已存在的那些监控实用程序外，还提供了一些脚本，用于创建 SP 问题管理 (pman) 事件以监控 DB2 HACMP ES 配置。要从 SP 控制工作站监控 HACMP 状态:

- 在该控制工作站上安装 HACMP 客户机代码。
- 编辑 /usr/sbin/cluster/etc/clhosts 文件，并包括您想要监控的这些节点的 SP 以太网 IP 地址。
- 调用命令 startsrc -s clinfo 来启动监控群集。

HACMP 提供了一个用于监控群集的接口（/usr/sbin/cluster/clstat）。

要使用“用于 HACMP RS 的 SP 透视 GUI”和用户定义事件来执行问题管理监控:

1. 调用 create_db2_events <nodelist>，其中，*nodelist* 包含 pcp 或 pexec 样式节点。此脚本创建由“透视”监控的 5 个 pman 事件。

注: 在创建这些事件时使用资源变量 PSSP.pm.User_state12-16。若这些资源变量已有其他用途，则必须更新 create_db2_events 和 update_db2_events，以使用不同的资源变量。

2. 在控制工作站上启动“透视”。从启动板，选择事件透视。您应该能看到 5 个事件: db2_hacmp_recovery、db2_process_recovery、db2_paging_err、db2_nfs_err 和 Errlog_PERM_entry。
3. 双击每个事件。在出现的屏幕上，注册（在“定义表”内）该事件的条件。单击名称: “未命名”旁边的向下箭头，并选择与您指定为条件的事件相同的名称。选择“应答选项”标签。单击屏幕顶部的按钮（“将消息发送至透视事件会话”）。可指定命令、错误日志条目以及 SNMP 软中断来触发这些事件。仅通过“透视”会话维护事件日志屏幕；因此，可能要为每个会话创建 AIX 错误日志条目。选择**确认**，并关闭该窗口。
4. 从“透视”启动板，选择硬件“透视”。
5. 当硬件框架 GUI 出现时，选择“视图”，然后选择“监控”。将提供对于 SP 可监控的事件列表。卷动至该列表的底部，将看到两个附加的事件: 一个用于 HACMP DB2 恢复 (db2_ha_ind)，另一个用于 SP 节点 PERM 错误 (Errlog_PERM_mon)。选择您想要监控的那些事件。（当一个事件发生时，节点显示一个红色的 ″X″。若所有被监控条件都良好，则节点屏幕是绿色的。）通常情况下，使用 host_responds、switch_responds 和 node_power_LED。也可监控该节点上的 DB2 HACMP 恢复以及 PERM 错误。

注: 用于 pman 和“透视”的 db2_hacmp_mon 和 db2_hacmp_recovery 变量不反映 HACMP 群集状态。相反，这些变量反映启动或停止 DB2 的 rc.db2pe 操作的状态。“真实的” HACMP 状态显示在 HACMP clstat 监控程序中，该状态反映 HACMP 群集状态。若想要让 db2_hacmp_ind 反映类似于 HACMP 状态的监控，则将下面这一行添加至 /etc/inittab 文件:

```
haind:2:wait:/usr/bin/db2_update_events HAIND OFF 2>&1 >/dev/null
```

若正计划将 NetView 用于您的实现，考虑使用 HAVIEW（它是 HACMP 的一部分）来监控配置。有关配置该产品的信息，参考 NetView 文档。

DB2 SP HACMP ES 安装

为了帮助您规划配合“DB2 通用数据库”安装 HACMP ES，下面提供了安装和迁移过程的逐步骤概述。

首次安装 DB2 SP HACMP ES

要安装 HACMP ES:

1. 在每个 SP 节点上安装 AIX 操作系统（参考 SP Installation and Administration Guides）。确保在控制工作站和每个 SP 节点上都有适当的调页空间可用。确保已考虑并实现了交换配置，以及任何其他可修改的配置参数。适当的设置您想要使用的 SP 监控（透视）。确保 SP dsh、pcp 和 pexec 命令运行。
2. 设计数据库布局。这至少应包括要使用的节点数、DB2 数据库分区至物理节点的映射、每个节点或分区的磁盘需求和表空间注意事项。还应考虑主 DB2 实例所有者将是谁，以及此用户和其他用户将需要什么访问授权。
3. 计划外部 SSA 磁盘配置，包括冗余适配器、镜像磁盘和双尾磁盘。
4. 使用数据库布局和 SSA 配置，填写 HACMP Planning, Installation, and Administration Guides 中的 HACMP 工作表。
5. 实现外部 SSA 磁盘配置。确保微代码级别在所有驱动器上是一致的，并使用 Maymap 实用程序验证并填充工作表中的任何间隙。
6. 在每个 SP 节点上安装 DB2 UDB EEE。
7. 在每个 SP 节点上安装 HACMP ES。
8. 使用 **db2_inst_ha** 命令安装 DB2 UDB EEE HACMP ES on SP 程序包。
9. 创建 DB2 主实例用户，并确保它可以访问所有节点。在这方面，这是一个可用性不高的用户。这可能是 SP 控制工作站上的一个临时 SP 用户。
10. 创建 DB2 实例和数据库。在转至下一步之前，通过调用 **db2start**，然后调用 **db2stop** 来确保该实例和数据库可操作。
11. 若想要在添加 HACMP 之前装入数据库，现在就应执行此操作。
12. 根据 HACMP 工作表和此文档中的信息，在 SP 节点拓扑和资源组上配置 HACMP ES。
13. 以 DB2 主实例用户的 NFS 服务器节点为起点，根据在此文档中的指定，在所有节点上更改此用户（通过修改 `/etc/security/user` 和 `/etc/passwd`）。此用户将成为可用性很高的 NFS 用户；并且此节点和其备份将更新 `/etc/exports`。通过交换别名 IP 地址，所有节点都将可以使用 NFS 来安装此目录（在每个节点上的 `/etc/filesystems` 中具有一项）。
14. “打包”(tar) 该主实例用户的主目录，并在新位置“解取”(untar) 该主目录。
15. 在每个 SP 节点上创建一个 NFS 文件系统，以安装新的主实例主目录。
16. 在 NFS 服务器节点上启动 HACMP。通过查看 `/tmp/hacmp.out`，验证它已成功启动。当写入此文件时，可使用 **ha_mon** 命令来监控它。
17. 一次启动一个其他节点，通过查看 `/tmp/hacmp.out` 来验证每个节点是否已成功启动。当写入此文件时，可使用 **ha_mon** 命令来监控它。
18. 通过“透视和问题管理”设置可选的监控。

19. 通过在每个节点上模拟并行维护操作，来验证每个节点上的故障转移功能。可使用 **ha_cmd** 命令（指定 TAKE 选项）来通过接管平缓停止 HACMP。通过查看 /tmp/hacmp.out 和使用监控工具来验证接管和重新集成是否成功。

DB2 SP HACMP ES 迁移

若要从一个非 HACMP 安装迁移至带 HACMP 的安装，则考虑下列概述:

1. 将现存的外部磁盘转换为高度可用的、双尾的、镜像配置。添加任何额外的硬件和磁盘以完成此配置，记住，当不同节点上的不同逻辑卷为双尾时，它们的名称*必须*是唯一的。这适用于卷组、逻辑卷和文件系统。
2. 完成该 HACMP 计划和相关的工作表，包括本文档中的工作表。
3. 实现外部 SSA 磁盘配置更改。确保微代码级别在所有驱动器上是一致的，并使用 Maymap 实用程序验证并消去工作表中的任何间隙。

   注: 支持 RAID5 配置中的 SSA 磁盘。在同一个 RAID 循环中存在两个 SSA 适配器是唯一允许的配置。对于具有 RAID 磁盘双尾的 HACMP 配置，只支持每个节点一个适配器。在这种配置中，该适配器是访问磁盘的单个故障点，建议用额外的配置检测适配器的运转中断问题，并借助于它来完成 HACMP 故障转移事件。AIX 出错通知是在 SSA 适配器失效时配置一个节点用于故障转移的最简单方法。有关 AIX 错误通知的更多信息，参考 *HACMP for AIX, V4.2.2, Enhanced Scalability Installation and Administration Guide*。
4. 在每个 SP 节点上安装 HACMP ES。
5. 使用 **db2_inst_ha** 命令安装 DB2 UDB EEE HACMP ES on SP 程序包。
6. 根据 HACMP 工作表和此文档中的信息，在 SP 节点拓扑和资源组上配置 HACMP ES。
7. 以 DB2 主实例用户的 NFS 服务器节点为起点，根据在此文档中的指定，在所有节点上更改此用户（通过修改 /etc/security/user 和 /etc/passwd）。此用户将成为可用性很高的 NFS 用户；并且此节点和其备份将更新 /etc/exports。通过交换别名 IP 地址，所有节点都将可以使用 NFS 来安装此目录（在每个节点上的 /etc/filesystems 中具有一项）。
8. “打包”该主实例用户的主目录，并在新位置“解包”该主目录。
9. 在每个 SP 节点上创建一个 NFS 文件系统，以安装新的主实例主目录。
10. 在 NFS 服务器节点上启动 HACMP。通过查看 /tmp/hacmp.out，验证它已成功启动。当写入此文件时，可使用 **ha_mon** 命令来监控它。
11. 一次启动一个其他节点，通过查看 /tmp/hacmp.out 来验证每个节点是否已成功启动。当写入此文件时，可使用 **ha_mon** 命令来监控它。
12. 通过“透视和问题管理”设置可选的监控。

13. 通过在每个节点上模拟并行维护操作，来验证每个节点上的故障转移功能。可使用 **ha_cmd** 命令（指定 TAKE 选项）来通过接管平缓停止 HACMP。通过查看 `/tmp/hacmp.out` 和使用监控工具来验证接管和重新集成是否成功。

DB2 SP HACMP ES 工作表

下列工作表设计成配合您在准备外部 SSA 磁盘配置时应该完成的 HACMP 工作表使用（那些工作表在 HACMP Planning, Installation, and Administration Guides 中）。在每种情况下，都提供了一个完成的示例和一个空白的工作表。

下图显示第一个样本工作表中所记载的外部磁盘上的数据库配置。用来创建该数据库的语句是:

```
db2 create database pwq on /newdata
```

为 SSA 外部适配器和外部 SSA 磁盘建立镜像和双尾，以使逻辑卷没有单点故障。此框图描述了一个类似于 **maymap** 命令的输出的配置。Maymap 是一个显示外部 SSA 磁盘配置的实用程序（通过 AIXTOOLS 使用），应在计划设置时使用。

DB2 四节点数据库外部磁盘设置样本

- 显示双尾以获取高可用性。

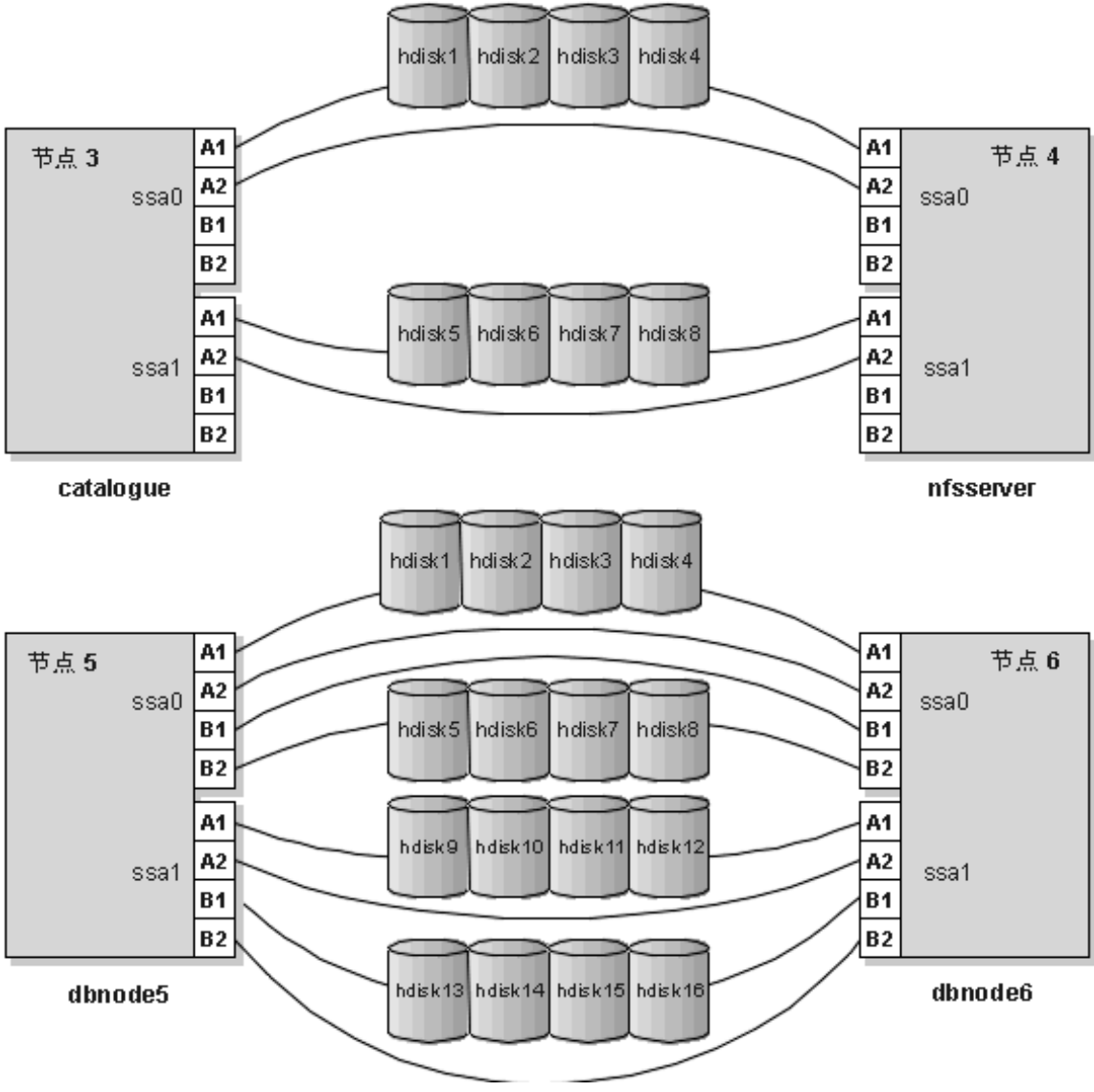


图 27. DB2 四节点数据库外部磁盘设置样本

在复查下表之前，您应仔细阅读 HACMP 文档中有关卷组上的定额设置及逻辑卷上的镜像化写一致性设置的内容。以上两个设置将直接影响可用性和性能。务必复查这些设置并理解其内在含义。“定额”和“镜像化写一致性”的典型设置都是 ″off″。

表 12. HACMP 卷组、逻辑卷和文件系统

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者
3	havg3	8	hlv300	10	2	hdisk1 hdisk5	/newdata /pwq /NODE0003	hlog301	目录节点安装点; 节点 4	root *
3	havg3	8	hlog301	1	2	hdisk1 hdisk5	N/A	N/A	目录节点 jfslog; 节点 4	root *
3	havg3	8	hlv301	10	2	hdisk2 hdisk6	N/A	N/A	目录节点原始临时空间; 节点 4	pwq **
4	havg4	8	hlv400	10	2	hdisk3 hdisk7	/dbmnt	hlog401	nfsserver pwq home; 节点 3	root *
4	havg4	8	hlog401	1	2	hdisk3 hdisk7	N/A	N/A	nfsserver jfslog; 节点 3	root *
5	havg5	8	hlv500	10	2	hdisk1 hdisk9	/newdata/ pwq/ NODE0005	HLOG501	Dbnode5 安装点; 节点 6	root *
5	havg5	8	hlog501	1	2	hdisk1 hdisk9	N/A	N/A	Dbnode5 jfslog; 节点 6	root *
5	havg5	8	hlv501	10	2	hdisk2 hdisk10	N/A	N/A	Dbnode5 原始临时空间; 节点 6	pwq **
5	havg5	8	hlv502	100	2	hdisk2 hdisk10	N/A	N/A	Dbnode5 原始表空间; 节点 6	pwq **

表 12. HACMP 卷组、逻辑卷和文件系统 (续)

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者
5	havg5	8	halv503	100	2	hdisk3 hdisk11	N/A	N/A	Dbnode5 原始表空间; 节点 6	pwq **
5	havg5	8	halv504	100	2	hdisk3 hdisk11	N/A	N/A	Dbnode5 原始表空间; 节点 6	pwq **
5	havg5	8	halv505	100	2	hdisk4 hdisk12	/dbdata5	hlog501	Dbnode6 系统表空间; 节点 6	root *
6	havg6	8	hlv600	10	2	hdisk5 hdisk13	/newdata/ pwq/ NODE0006	hlog601	Dbnode6 安装点; 节点 5	root *
6	havg6	8	hlog601	1	2	hdisk5 hdisk13	N/A	N/A	Dbnode6 jfslog; 节点 5	root *
6	havg6	8	hlv601	10	2	hdisk6 hdisk14	N/A	N/A	Dbnode6 原始临时空间; 节点 5	pwq **
6	havg6	8	hlv602	100	2	hdisk6 hdisk14	N/A	N/A	Dbnode6 原始表空间; 节点 5	pwq **
6	havg6	8	hlv603	100	2	hdisk7 hdisk15	N/A	N/A	Dbnode6 原始表空间; 节点 5	pwq **
6	havg6	8	hlv604	100	2	hdisk7 hdisk15	N/A	N/A	Dbnode6 原始表空间; 节点 5	pwq **
6	havg6	8	hlv605	100	2	hdisk8 hdisk16	/dbdata6	hlog601	Dbnode6 系统表空间; 节点 5	root *

表 12. HACMP 卷组、逻辑卷和文件系统 (续)

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者
注: 1. * jfs 文件系统逻辑卷和日志保留 root 用户许可权。 2. ** 原始数据库空间获取 /dev 原始文件条目 (/dev/rxxxx) 的数据库用户许可权。										

表 13. HACMP 卷组、逻辑卷和文件系统 – 空白

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者

表 13. HACMP 卷组、逻辑卷和文件系统 – 空白 (续)

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者

表 13. HACMP 卷组、逻辑卷和文件系统 – 空白 (续)

SP 节点	卷组名	PP 大小 (MB)	逻辑卷名	PP 数	份数	硬盘列表	文件系统安装点	文件系统日志逻辑卷	节点描述和备份	/dev 逻辑设备的用户所有者

表 14. 计划 HACMP NFS 服务器

SP 节点	外部文件系统	备份节点	SP 交换引导和服务 IP 别名对	要安装的文件系统 (/etc/filesystems)	要指定成数据库主目录的文件系统	作为文件系统导出目的地的地址 (/etc/exports)
3	/dbmnt	4	nfs_boot_3 nfs_client_3	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
4	/dbmnt	3	nfs_server_boot nfs_server	nfs_server:/ dbmnt as /dbi	/dbi/pwq	nfs_boot_3 nfs_client_3 nfs_server_boot nfs_server nfs_boot_5 nfs_client_5 nfs_boot_6 nfs_client_6
5	N/A	N/A	nfs_boot_5 nfs_client_5	nfs_server:/ dbmnt as /dbi	/dbi/pwq	N/A
6	N/A	N/A	nfs_boot_6 nfs_client_6	nfs_server:/ dbmnt as /dbi	/dbi/pwq	N/A

注:

1. /etc/passwd 必须在所有节点上都相同可从控制工作站将它同步。
2. 确保外部文件系统具有数据库实例所有者的许可权。
3. /etc/filesystems 必须具有安装参数: hard、bg、intr 和 rw。
4. /etc/exports 将仅在服务器和其备份上具有

   -root=ip1:ip2:ip3

   。

表 15. 计划 HACMP NFS 服务器 - 空白

SP 节点	外部文件系统	备份节点	SP 交换引导和服务 IP 别名对	要安装的文件系统 (/etc/filesystems)	要指定成数据库主目录的文件系统	作为文件系统导出目的地的地址 (/etc/exports)

表 15. 计划 HACMP NFS 服务器 - 空白 (续)

SP 节点	外部文件系统	备份节点	SP 交换引导和服务 IP 别名对	要安装的文件系统 (/etc/filesystems)	要指定成数据库主目录的文件系统	作为文件系统导出目的地的地址 (/etc/exports)

第7章 Windows 操作系统上的高可用性

您可设置数据库系统，以便当一台机器发生故障时，在这台机器上运行的数据库服务器可在另一台机器上运行。在 Windows NT 上，故障转移支持可通过“Microsoft 群集服务器” (MSCS) 实现。要使用 MSCS，您需要 Windows NT 企业版的版本 4.0，并安装了 MSCS 功能部件。

在一个群集环境中，MSCS 可同时执行故障检测和重新启动资源，如对物理磁盘和 IP 地址的故障转移支持。（当有故障的机器再次联机时，除非您先前配置资源这样做，否则资源不会自动回退到原来的状态。有关更多信息，参见第210页的『回退注意事项』。）

在启用 DB2 实例以支持故障转移之前，执行下列计划步骤:

1. 确定要使用哪些磁盘来存储数据。应给每个数据库服务器分配至少一个磁盘供它自己使用。用于存储数据的磁盘必须与一个共享磁盘子系统连接，且必须配置为 MSCS 磁盘资源。
2. 确保要用于支持远程请求的每个数据库服务器都有一个 IP 地址。

当设置故障转移支持时，它可以用于现有实例，或可以在实现故障转移支持时创建一个新实例。

要启用故障转移支持，执行下列步骤:

1. 为 DB2MSCS 实用程序创建一个输入文件。
2. 调用 **db2mscs** 命令。
3. 若正在使用一个分区数据库系统，则注册数据库驱动器映射以启用相互接管。参见第211页的『在分区数据库环境中，为进行相互接管配置而注册数据库驱动器映射』。

在启用了实例以使用故障转移支持后，配置将类似于第200页的图28。

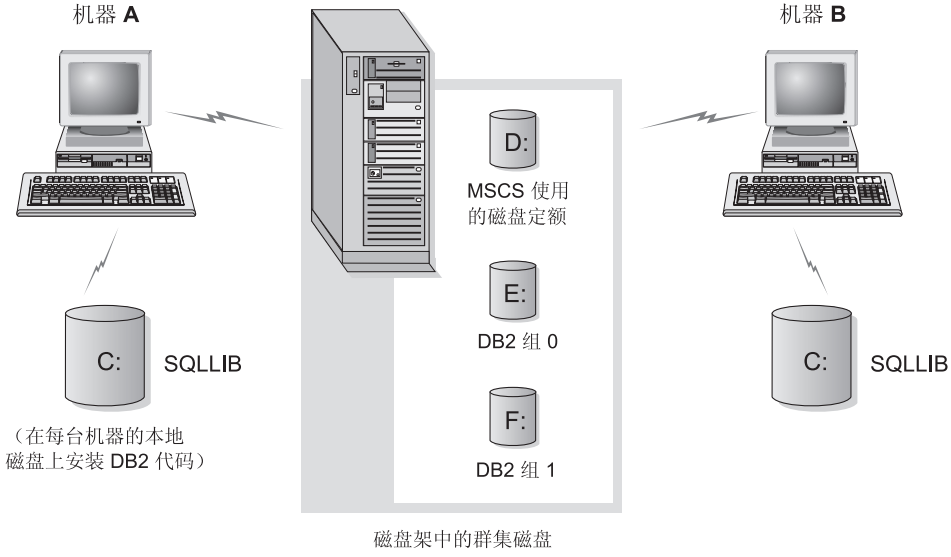


图 28. 示例 MSCS 配置

下列几节描述不同类型的故障转移支持，以及如何实现它们。在执行以下描述的任何步骤前，您必须已经在 MSCS 群集中要使用的每台机器上安装了 MSCS 软件。另外，还必须在每台机器上安装了 DB2。

故障转移配置

可使用两种类型的配置:

- 热备用
- 相互接管

目前，MSCS 支持两台机器的群集。

在一个分区数据库环境中，群集不必全部具有相同类型的配置。您可将一些群集设置为使用热备用，而将其他群集设置为使用相互接管。例如，若您的 DB2 实例由五个工作站组成，可以将两台机器设置为使用相互接管配置，将另外两个设置为使用热备用配置，而将剩余的一台机器配置为不支持故障转移。

热备用配置

在一个热备用配置中，MSCS 群集中的一台机器提供专用的故障转移支持，而另一台机器参与该数据库系统。若参与该数据库系统的机器发生故障，则该机器上的数据库服务器将在故障转移机器上启动。若在一个分区数据库系统中，您正在一

台机器上运行多个逻辑节点，而该机器发生故障，则这些逻辑节点将在故障转移机器上启动。图29 显示了一个热备用配置的示例。

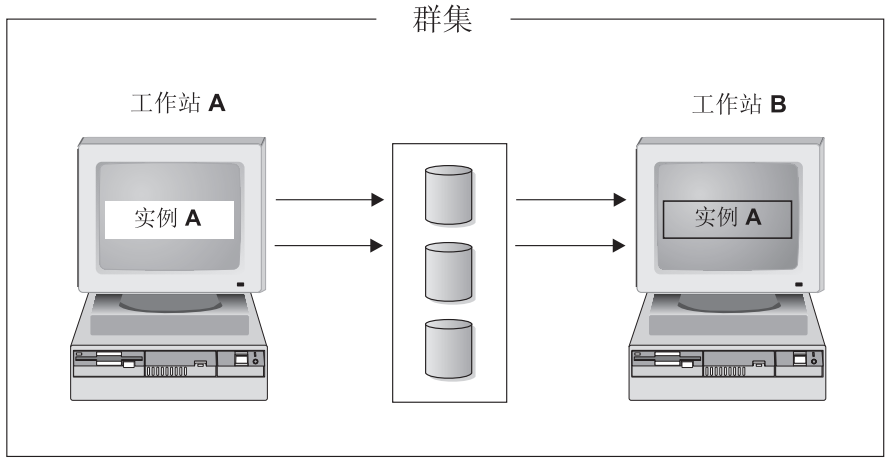


图 29. 热备用配置

相互接管配置

在一个相互接管配置中，两个工作站都参与数据库系统（即，每台机器上至少有一个数据库服务器在运行）。若 MSCS 群集中的一个工作站失效，则该失效机器上的数据库服务器将在另一台机器上启动以运行。在一个相互接管配置中，一台机器上的数据库服务器发生故障时不会影响另一台机器上的数据库服务器。在任何给定的时间点，任何数据库服务器可在任何机器上是活动的。第202页的图30 显示了一个相互接管配置的示例。

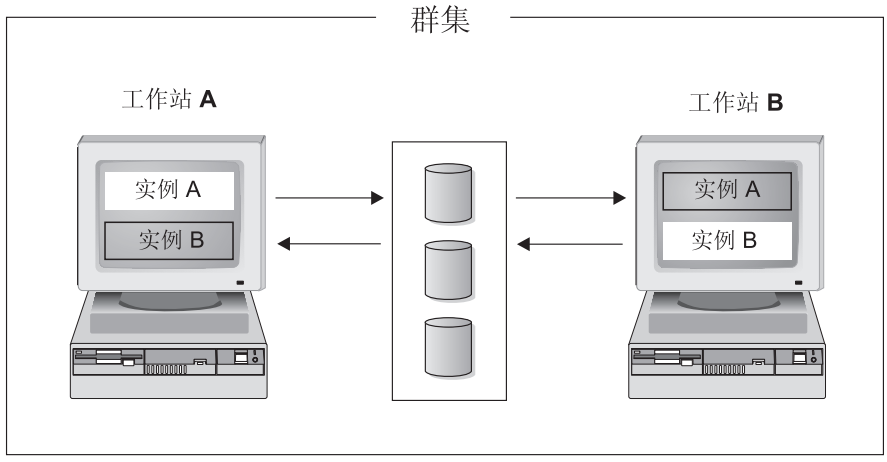


图 30. 相互接管配置

使用 DB2MSCS 实用程序

使用 DB2MSCS 实用程序创建基础结构，以便 DB2 在 Windows NT 环境中使用“Microsoft 群集服务 (MSCS) 支持”来支持故障转移。可使用此实用程序以便在单分区和分区数据库环境中启用故障转移。

为使 DB2MSCS 实用程序成功运行，“群集服务”必须能够定位资源 DLL `db2wolf.dll`，它驻留在 `%ProgramFiles%\SQLLIB\bin` 目录下。“DB2 UDB 版本 7 安装程序”将 PATH 系统环境变量设置为指向 `%ProgramFiles%\SQLLIB\bin` 目录。但如果是在 Windows 2000 操作系统上运行，则不需要在安装后重新引导机器。如果想要运行 DB2MSCS 实用程序，则必须重新引导机器，以使 PATH 环境变量更新为适用于“群集服务”。

对于每个实例，在它们拥有实例的机器上调用一次 **db2mscs** 命令。若在 MSCS 群集中的一台机器上只有一个 DB2 实例在运行，该操作将设置热备用配置。若在 MSCS 群集中的每台机器上都有一个实例在运行，则应在每个拥有实例的机器上运行 DB2MSCS 一次，以设置相互接管配置。

DB2MSCS 实用程序:

1. 从名为 `DB2MSCS.CFG` 的输入文件中读取必需的 MSCS 和 DB2 参数。有关全套输入参数的信息，参见第203页的『指定 DB2MSCS.CFG 文件』。
2. 验证输入文件中的参数。
3. 注册 DB2 资源类型。
4. 创建一个或多个 MSCS 组，以包含 MSCS 和 DB2 资源。
5. 创建 IP 资源。

6. 创建 “网络名” 资源。
7. 将 MSCS 磁盘移至组中。
8. 创建一个或多个 DB2 资源。
9. 添加 DB2 资源的所有必需的从属部件。
10. 将非群集 DB2 实例转换为群集实例。
11. 使所有资源联机。

该命令语法如下所示:

其中:

-f:*input_file*

指定 MSCS 实用程序要使用的 DB2MSCS.CFG 输入文件。若未指定此参数，DB2MSCS 实用程序会读取当前目录中的 DB2MSCS.CFG 文件。

指定 DB2MSCS.CFG 文件

DB2MSCS.CFG 文件是 ASCII 文本文件，它包含由 DB2MSCS 实用程序读取的参数。使用下列格式在单独的一行上指定每个输入参数: PARAMETER_KEYWORD=*parameter_value*。例如:

```
CLUSTER_NAME=WOLFPACK
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89
```

在 /SQLLIB 目录的 /CFG 子目录中有两个示例配置文件。第一个文件 DB2MSCS.EE 是单分区数据库环境的示例。第二个文件 DB2MSCS.EEE 是分区数据库环境的示例。

DB2MSCS.CFG 文件的参数如下所示:

DB2_INSTANCE

DB2 实例的名称。若未指定实例名，则使用缺省实例（DB2INSTANCE 环境变量的值）。

此参数具有全局作用域，且只在 DB2MSCS.CFG 文件中指定一次。

此参数是可选的。

示例:

```
DB2_INSTANCE=DB2
```

该实例必须已存在。有关创建实例的信息，参考 *DB2 Enterprise - Extended Edition for Windows Quick Beginnings* 一书。

DB2_LOGON_USERNAME

DB2 服务的登录帐户名。

此参数具有全局作用域，且只在 DB2MSCS.CFG 文件中指定一次。

只有 DB2 扩充企业版实例需要此参数。

示例:

```
DB2_LOGON_USERNAME=db2user
```

DB2_LOGON_PASSWORD

DB2 服务的登录帐户的密码。若提供了 DB2_LOGON_USERNAME 参数，但没有提供 DB2_LOGON_PASSWORD 参数，则 DB2MSCS 实用程序会提示您输入该密码。在命令行输入密码时不会显示出来。

此参数具有全局作用域，且只在 DB2MSCS.CFG 文件中指定一次。

只有 DB2 扩充企业版实例需要此参数。

示例:

```
DB2_LOGON_PASSWORD=xxxxxx
```

CLUSTER_NAME

MSCS 群集名。只有指定了另一个 CLUSTER_NAME 标记后，才在此群集中创建在此行后指定的所有资源。

对每个群集指定此参数一次。

此参数是可选的。若未指定，则使用本地机器上的 MSCS 群集的名称。

示例:

```
CLUSTER_NAME=WOLFPACK
```

GROUP_NAME

MSCS 组名。若指定了此参数，当 MSCS 组不存在时，会创建一个新的 MSCS 组。如果该组已存在，则将它用作目标组。只有在指定了另一个 GROUP_NAME 关键字之后，才在此组中创建在此行后创建的任何 MSCS 资源。

对每个组，指定此参数一次。

此参数是必需的。

示例:

```
GROUP_NAME=DB2 Group
```

DB2_NODE

当前 MSCS 组将包括的数据库分区服务器（节点）的节点号。若同一台机器上存在多个逻辑节点，则每个节点需要一个单独的 DB2_NODE 关键字。

在 GROUP_NAME 参数后指定此参数，以便在正确的 MSCS 组中创建 DB2 资源。

只有 DB2 扩充企业版实例需要此参数。

示例:

```
DB2_NODE=0
```

IP_NAME

“IP 地址”资源的名称。IP_NAME 的值是任意的，但必须是唯一的。当指定此参数时，会创建一个类型为“IP 地址”的 MSCS 资源。

远程 TCP/IP 连接需要此参数。您必须为在分区数据库环境中的拥有实例的机器指定此参数。在单分区数据库环境中，此参数是可选的。

示例:

```
IP_NAME=IP Address for DB2
```

注: DB2 客户机应使用此 IP 资源的 TCP/IP 地址来编目 TCP/IP 节点项。通过使用 MSCS IP 地址，当数据库服务器执行故障转移到另一机器时，DB2 客户机仍可与数据库服务器连接，因为在故障转移机器上 IP 地址是可用的。

IP 资源的属性如下所示:

IP_ADDRESS

IP 资源的 TCP/IP 地址。指定此关键字，以设置上述 IP 资源的 TCP/IP 地址。

若指定了 IP_NAME 参数，则此参数是必需的。

示例:

```
IP_ADDRESS=9.21.22.34
```

IP_SUBNET

上述 IP 资源的子网掩码。

若指定了 IP_NAME 参数，则此参数是必需的。

示例:

```
IP_SUBNET=255.255.255.0
```

IP_NETWORK

上述 IP 资源所属的 MSCS 网络的名称。若未指定此参数，则使用系统检测到的第一个 MSCS 网络。

此参数是可选的。

示例:

```
IP_NETWORK=Token Ring
```

NETNAME_NAME

“网络名”资源的名称。指定此参数以创建“网络名”资源。

对于单分区数据库环境，此参数是可选的。对于分区数据库环境，此参数是必需的。

示例:

```
NETNAME_NAME=Network name for DB2
```

“网络名”资源的属性如下所示:

NETNAME_VALUE

“网络名”的值。

若指定了 NETNAME_NAME 参数，则此参数是必需的。

示例:

```
NETNAME_VALUE=DB2SRV
```

NETNAME_DEPENDENCY

“网络名”资源的从属项列表。每个“网络名”资源都必须从属于一个“IP 地址”资源。若未指定此参数，则“网络名”资源从属于该组中的第一个 IP 资源。

此参数是可选的。

示例:

```
NETNAME_DEPENDENCY=IP Address for DB2
```

DISK_NAME

要移动到当前组中的物理磁盘资源的名称。指定所需的尽量多的磁盘资源。

注:

1. 磁盘资源必须已存在。
2. 当 DB2MSCS 实用程序为 MSCS 支持配置 DB2 实例时，将该实例目录复制到该组中的第一个 MSCS 磁盘。要为该实例目录指定不同的 MSCS 磁盘，可使用 INSTPROF_DISK 参数。

示例:

```
DISK_NAME=Disk E:
DISK_NAME=Disk F:
```

INSTPROF_DISK

一个可选的参数，指定包含 DB2 实例目录的 MSCS 磁盘。若未指定此参数，DB2MSCS 实用程序使用该实例目录所在的同一个组中的第一个 MSCS 磁盘。

该 DB2 实例目录创建在 MSCS 磁盘上的 X:\DB2PROFS 目录下（其中，X 是 MSCS 磁盘的盘符）。

示例:

```
INSTPROF_DISK=Disk E:
```

TARGET_DRVMAP_DISK

用来指定数据库驱动器映射的目标 MSCS 磁盘的可选参数。如果在与节点不属于同一组的 MSCS 磁盘上创建数据库，则使用目标驱动器映射磁盘来包含数据库分区。如果未指定此参数，必须使用 DB2DRVMP 实用程序手工注册数据库驱动器映射。

示例:

```
TARGET_DRVMAP_DISK = Disk E:
```

为单分区数据库系统设置故障转移

当您对单分区数据库系统运行 DB2MSCS 实用程序时，一个 MSCS 组包含 DB2 和所有从属的 MSCS 资源（IP 地址、网络名和磁盘）。例如，一个单分区数据库系统的 DB2MSCS.CFG 文件的内容将类似于如下内容:

```
#
#  DB2MSCS.CFG for a single-partition database system
#
DB2_INSTANCE=DB2
CLUSTER_NAME=MSCS
GROUP_NAME=DB2 Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

为两个单分区数据库系统设置相互接管配置

您可设置两个单分区数据库系统，每台机器上一个，这样若任何一台机器上的数据库系统发生故障，都可在另一 MSCS 节点上重新启动它。

要为此配置设置故障转移支持，需要在每个拥有实例的机器上将 DB2MSCS 实用程序运行一次。必须为每个数据库系统定制配置文件。

假设 DB2 实例称为 DB2A 和 DB2B。DB2A 实例的 DB2MSCS.CFG 文件如下所示:

```
#
#  DB2MSCS.CFG for first single-partition database system
#
DB2_INSTANCE=DB2A
CLUSTER_NAME=MSCS
GROUP_NAME=DB2A Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk E:
```

DB2B 实例的 DB2MSCS.CFG 文件可能会如下所示:

```
#
#  DB2MSCS.CFG for second single-partition database system
#
DB2_INSTANCE=DB2B
CLUSTER_NAME=MSCS
GROUP_NAME=DB2B Group
IP_NAME=...
IP_ADDRESS=...
IP_SUBNET=...
IP_NETWORK=...
NETNAME_NAME=...
NETNAME_VALUE=...
DISK_NAME=Disk F:
```

有关完整的示例，参见第213页的『示例 - 为相互接管设置两个单分区实例』。

为分区数据库系统设置多个 MSCS 群集

对多分区数据库系统运行 DB2MSCS 实用程序时，将为参与该系统的每个物理机器创建一个 MSCS 组。DB2MSCS.CFG 文件必须包含多节，且对于 GROUP_NAME 参数以及该组的所有必需的从属资源，每一节必须具有不同的值。

而且，必须对每个 MSCS 组中的每个数据库分区服务器指定 DB2_NODE 参数。若您有多个逻辑节点，每个逻辑节点都需要一个单独的 DB2_NODE 关键字。

例如，假设有一个多分区数据库系统，由四台机器上的四个数据库分区服务器组成，而您想将其中两个 MSCS 群集配置为使用相互接管配置。应按如下所示设置 DB2MSCS.CFG 配置文件:

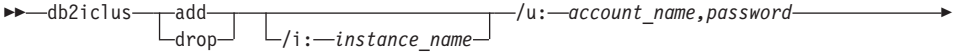
```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple clusters
DB2_INSTANCE=DB2MPP
DB2_LOGON_USERNAME=db2user
DB2_LOGON_PASSWORD=xxxxxx
CLUSTER_NAME=MSCS1
# Group 1
GROUP_NAME=DB2 Group 1
DB2_NODE=0
IP_NAME=...

...
# Group 2
GROUP_NAME=DB2 Group 2
DB2_NODE=1
IP_NAME=...

...

CLUSTER_NAME=MSCS2
# Group 3
GROUP_NAME=DB2 Group 3
DB2_NODE=2
IP_NAME=...

...
# Group 4
GROUP_NAME=DB2 Group 4
DB2_NODE=3
IP_NAME=...

...
```

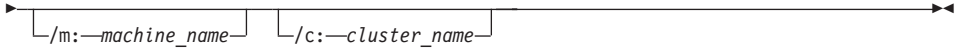
有关完整的示例，参见第215页的『示例 - 为相互接管设置四节点分区数据库系统』。

维护 MSCS 系统

当运行 DB2MSCS 实用程序时，它将为 MSCS 群集中的所有机器创建故障转移支持的基础结构。要从机器上除去该支持，使用带 ″drop″ 选项的 **db2iclus** 命令。要对机器重新启用该支持，可使用 ″add″ 选项。

该命令语法如下所示:





其中:

add
将故障转移支持添加至一个 MSCS 群集，从而在机器上启用故障转移支持。这样 DB2 资源（数据库服务器）就可在此机器上执行故障转移。

drop
从 MSCS 群集中删除故障转移支持，以从机器中除去它。

/i: *instance_name*
实例名称。（此参数覆盖 DB2INSTANCE 环境变量的设置值。）

/u: *account_name*, *password*
用作“DB2 服务”的登录帐户名的域帐户。例如:

```
/u:domainA\db2nt,password
```

只在使用 ″add″ 选项时，此参数才是必需的。

/m:*machine_name*
要添加至 MSCS 群集的或要从 MSCS 群集中删除的机器的计算机名。若您是从另一台机器而不是正在对其修改故障转移支持的机器运行该命令，必须指定此选项。

/c: *cluster_name*
在 LAN 上使用的 MSCS 群集的名称。此名称是在第一次创建 MSCS 群集时指定的。

回退注意事项

缺省情况下，将组设置为不回退到原始的（发生故障的）机器。除非将 DB2 组手工配置为在故障转移后回退，否则在解决了造成故障转移的问题后，它将继续在备用的 MSCS 节点上运行。

若将 DB2 组配置为自动回退到原始机器，则一旦原始机器可用，DB2 组中的所有资源，包括 DB2 资源将会尽快回退。若在回退期间存在一个数据库连接，则 DB2 资源不能脱机，而回退处理将失败。

若要在回退处理期间强行中断所有数据库连接，则将 DB2_FALLBACK 注册表变量设置为 ON。此变量必须按如下所示的格式设置:

```
db2set DB2_FALLBACK=ON
```

您不必在设置此注册表变量后重新引导或重新启动该群集服务。

在分区数据库环境中，为进行相互接管配置而注册数据库驱动器映射

当在分区数据库环境中创建数据库时，可指定一个盘符，以指示要在何处创建数据库。

注: 不要对单分区数据库环境设置数据库驱动器映射。

当 CREATE DATABASE 命令运行时，它希望您指定的驱动器将同时对所有参与该实例的机器可用。因为这是不可能的，所以 DB2 使用数据库驱动器映射，为每台机器的相同驱动器指定不同的名称。

例如，假设名为 DB2 的一个 DB2 实例包含两个数据库分区服务器:

```
NODE0 在机器 WOLF_NODE_0 上是活动的
NODE1 在机器 WOLF_NODE_1 上是活动的
```

还假设共享磁盘 E: 与 NODE0 属于同一个组，而共享磁盘 F: 与 NODE1 属于同一个组。

要在共享磁盘 E 上创建数据库:

```
db2 create database mppdb on e:
```

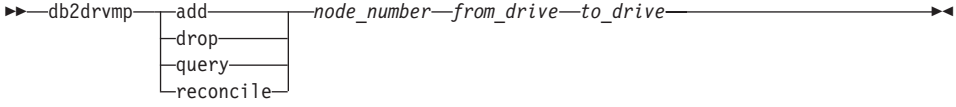
为了使该命令成功，驱动器 E: 必须同时对两台机器可用。在一个相互接管配置中，每个数据库分区服务器可在不同的机器上是活动的，而群集磁盘 E: 只对一台机器是可用的。在这种情况中，CREATE DATABASE 命令将总是失败。

要解决这个问题，数据库驱动器应按如下所示进行映射:

```
对于 NODE0，该映射是从驱动器 F: 至驱动器 E:
对于 NODE1，该映射是从驱动器 E: 至驱动器 F:
```

然后，将 NODE0 对驱动器 F: 的任何数据库访问映射至驱动器 E:，将 NODE1 对驱动器 E: 的任何数据库访问映射至驱动器 F:。使用驱动器映射，CREATE DATABASE 命令将在 NODE0 的驱动器 E: 和 NODE1 的驱动器 F: 上创建数据库文件。

使用 **db2drvmp** 命令来设置驱动器映射。该命令语法如下所示:



这些参数如下所示:

add 指定新的数据库驱动器映射。

drop 除去现有的数据库驱动器映射。

query 查询数据库映射。

reconcile 当注册表内容损坏时，修复数据库映射驱动器。有关更多信息，参见『协调数据库驱动器映射』。

node_number 节点号。添加和删除操作需要此参数。

from_drive 映射的源盘符。添加和删除操作需要此参数。

to_drive 映射的目标盘符。添加操作需要此参数。该参数对其他操作不适用。

若要设置从 F: 至 NODE0 的 E: 的数据库驱动器映射，应使用如下命令:

```
db2drvmp add 0 F E
```

注: 数据库驱动器映射不适用于表空间、容器或任何其他数据库存储对象。

类似地，要设置从 E: 至 NODE1 的 F: 的数据库驱动器映射，应发出如下命令:

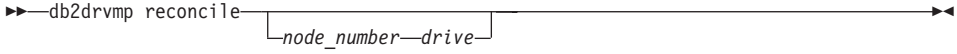
```
db2drvmp add 1 E F
```

注: 任何对数据库驱动器映射的设置或更改都不会立即生效。要激活数据库驱动器映射，可使用“群集管理员”工具将 DB2 资源脱机，然后再联机。

使用 DB2MSCS.CFG 文件中的 TARGET_DRVMAP_DISK 关键字将使驱动器映射自动完成。

协调数据库驱动器映射

当在具有生效的数据库驱动器映射的机器上创建数据库时，该映射保存在驱动器上的隐藏文件中。这样可防止创建数据库后数据库驱动器被除去。例如，当您意外地删除了数据库驱动器映射时，您将希望*协调*数据库驱动器映射。要协调该映射，对包含数据库的每个数据库分区服务器运行 **db2drvmp reconcile** 命令。该命令语法如下所示:



这些参数如下所示:

node_number 要修复的节点的节点号。若未指定 *node_number*，该命令会协调所有节点的映射。

drive 要协调的驱动器。若未指定驱动器，该命令会协调所有驱动器的映射。

db2drvmp 命令扫描机器上的所有驱动器以查找数据库分区服务器管理的数据库分区，然后按需要对注册表重新应用数据库驱动器映射。

示例 - 为相互接管设置两个单分区实例

此示例的目的是在一个相互接管配置中设置两个具有故障转移支持的单分区数据库实例。在此示例中，将四个服务器配置为两个 MSCS 群集。通过使用相互接管配置，当任何一台机器发生故障时，为该机器配置的数据库服务器将在备用机器上执行故障转移（通过使用 MSCS 软件配置），然后在备用机器上运行。

在结果配置中有两个 MSCS 群集。每个群集具有:

- 两个服务器，分别具有 64 MB 内存和一个 2 GB 的本地 SCSI 磁盘
- 一个 SCSI 磁盘塔，具有三个共享的 SCSI 磁盘，每个有 2 GB。

另外，每台机器都安装了一个 100X 以太网适配卡。

每台机器都安装了下列软件:

- 安装了 MSCS 功能部件的 “Windows NT 企业版的版本 4.0”
- DB2 通用数据库企业版的版本 7。

所产生的网络配置如下所示:

服务器 1:	服务器 2:
• 机器名: db2test1	• 机器名: db2test2
• TCP/IP 主机名: db2test1	• TCP/IP 主机名: db2test2
• IP 地址: 9.9.9.1	• IP 地址: 9.9.9.2
子网掩码: 255.255.255.0	子网掩码: 255.255.255.0
• MSCS 群集名: ClusterA	• MSCS 群集名: ClusterA

网络中的两台机器都配置为使用 TCP/IP，且使用以太网 100 T-base 集线器与专用 LAN 连接。若缺少 “域名服务器” (DNS)，则所有机器都有一个本地 TCP/IP hosts 文件，该文件包含下列各个条目:

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 ClusterA # for MSCS ClusterA
9.9.9.4 db2tcp1 # for DB2 remote client connection to Server 1
9.9.9.5 db2tcp2 # for DB2 remote client connection to Server 2
```

初步任务

在执行下列任务前，假设两台机器属于同一个域 DB2NTD:

1. 在 DB2 将运行的那些机器上，为 DB2 创建一个作为“本地管理员”组的域帐户。使用该帐户来执行所有任务:
   - 将用户名设置为 db2nt。
   - 将密码设置为 db2nt。
2. 在机器 db2test1 和 db2test2 上安装 MSCS 功能部件:
   - 将 MSCS 群集命名为 ClusterA。
   - 群集 IP 地址为 9.9.9.3。
   - MSCS 软件将使用共享磁盘 D:。
   - DB2 将使用共享磁盘 E: 和 F:。
3. 在机器 db2test1 上安装“DB2 通用数据库企业版版本 7”。在本地驱动器 C:\SQLLIB 上安装该软件。
4. 在机器 db2test2 上安装“DB2 通用数据库企业版版本 7”。在本地驱动器 C:\SQLLIB 上安装该软件。

下一步是为每个实例设置 DB2MSCS.CFG 文件，然后对每个实例运行 DB2MSCS 实用程序。

运行 DB2MSCS 实用程序

要设置 db2test1 机器，应执行下列任务:

1. 在机器 db2test1 上，作为用户 db2nt 登录。密码是 db2nt。
2. 若 DB2 实例 DB2A 尚不存在，则创建它。创建该实例的命令是:

   ```
   db2icrt DB2A
   ```

3. 在机器 db2test1 上为 DB2 实例设置 DB2MSCS.CFG 文件:

   ```
   #
   #  DB2MSCS.CFG for database system
   #  on machine db2test1
   DB2_INSTANCE=DB2A
   CLUSTER_NAME=ClusterA
   #
   # Group 1
   GROUP_NAME=DB2A Group
   IP_NAME=IP Address for DB2A
   IP_ADDRESS=9.9.9.4
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Network name for DB2A
   NETNAME_VALUE=DB2SRV1
   NETNAME_DEPENDENCY=IP Address for DB2A
   DISK_NAME=Disk E:
   INSTPROF_DISK=Disk E:
   ```

4. 运行 DB2MSCS 实用程序，如下所示:

```
db2mscs -f:DB2MSCS.CFG
```

5. 从 db2nt 帐户注销。
6. 在机器 db2test2 上，作为用户 db2nt 登录，它属于“本地管理员”组。密码是 db2nt。
7. 若 DB2 实例 DB2B 尚不存在，则创建它。创建该实例的命令是:

   ```
   db2icrt DB2B
   ```

8. 在机器 db2test2 上为 DB2 实例设置 DB2MSCS.CFG 文件:

   ```
   #
   #  DB2MSCS.CFG for database system
   #  on machine db2test2
   DB2_INSTANCE=DB2B
   CLUSTER_NAME=ClusterA
   #
   # Group 1
   GROUP_NAME=DB2B Group
   IP_NAME=IP Address for DB2B
   IP_ADDRESS=9.9.9.5
   IP_SUBNET=255.255.255.0
   IP_NETWORK=ClusterA
   NETNAME_NAME=Network name for DB2B
   NETNAME_VALUE=DB2SRV2
   NETNAME_DEPENDENCY=IP Address for DB2B
   DISK_NAME=Disk F:
   INSTPROF_DISK=Disk F:
   ```

9. 运行 DB2MSCS 实用程序，如下所示:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

10. 从 db2nt 帐户注销。

示例 - 为相互接管设置四节点分区数据库系统

此示例的目的是在一个相互接管配置中设置具有故障转移支持的四节点分区数据库系统。在此示例中，将四个服务器配置为两个 MSCS 群集。通过使用相互接管配置，如果任何一台机器发生故障，则为该机器配置的数据库分区服务器将对备用机器执行故障转移（通过使用 MSCS 软件配置），然后在备用机器上作为一个逻辑节点运行。

在结果配置中有两个 MSCS 群集。每个群集具有:

- 两个服务器，分别具有 64 MB 内存和一个 2 GB 的本地 SCSI 磁盘
- 一个 SCSI 磁盘塔，具有三个共享的 SCSI 磁盘，每个 2 GB。

另外，每台机器都安装了一个 100X 以太网适配卡。

每台机器都安装了下列软件:

- 安装了 MSCS 功能部件的 “Windows NT 企业版的版本 4.0”
- DB2 通用数据库扩充企业版的版本 7。

产生的网络配置如下所示:

服务器 1: • 机器名: db2test1 • TCP/IP 主机名: db2test1 • IP 地址: 9.9.9.1 子网掩码: 255.255.255.0 • MSCS 群集名: ClusterA	服务器 2: • 机器名: db2test2 • TCP/IP 主机名: db2test2 • IP 地址: 9.9.9.2 子网掩码: 255.255.255.0 • MSCS 群集名: ClusterA
服务器 3: • 机器名: db2test3 • TCP/IP 主机名: db2test3 • IP 地址: 9.9.9.3 子网掩码: 255.255.255.0 • MSCS 群集名: ClusterB	服务器 4: • 机器名: db2test4 • TCP/IP 主机名: db2test4 • IP 地址: 9.9.9.4 子网掩码: 255.255.255.0 • MSCS 群集名: ClusterB

网络中的所有机器都配置为使用 TCP/IP，且使用以太网 100 T-base 集线器与专用 LAN 连接。若缺少 “域名服务器” (DNS)，则所有机器都有一个本地 TCP/IP `hosts` 文件，该文件包含下列各个条目:

```
9.9.9.1 db2test1 # for Server 1
9.9.9.2 db2test2 # for Server 2
9.9.9.3 db2test3 # for Server 3
9.9.9.4 db2test4 # for Server 4
9.9.9.5 ClusterA # for MSCS Cluster 1
9.9.9.6 ClusterB # for MSCS Cluster 2
9.9.9.7 db2tcp # for DB2 remote client connection
```

初步任务

在执行下列任务前，假设这四台机器都属于同一个域 `DB2NTB`:

1. 在 DB2 将运行的那些机器上，为 DB2 创建一个作为 “本地管理员” 组的域帐户。使用该帐户来执行所有任务:
   - 将用户名设置为 `db2nt`。
   - 将密码设置为 `db2nt`。
2. 创建具有 “*密码永不到期*” 特征的第二个域帐户。此帐户将与 DB2 服务相关:
   - 将用户名设置为 `db2mpp`。
   - 将密码设置为 `db2mpp`。

3. 在机器 db2test1 和 db2test2 上安装 MSCS 功能部件:
   - 将 MSCS 群集命名为 ClusterA。
   - 该群集的 “IP 地址” 为 9.9.9.5。
   - MSCS 软件将使用共享磁盘 D:。
   - DB2 将使用共享磁盘 E: 和 F:。
4. 在机器 db2test3 和 db2test4 上安装 MSCS 功能部件:
   - 将 MSCS 群集命名为 ClusterB。
   - 该群集的 “IP 地址” 为 9.9.9.6。
   - MSCS 软件将使用共享磁盘 D:
   - DB2 将使用共享磁盘 E: 和 F:。
5. 在机器 db2test1 上安装 DB2 扩充企业版:
   - 选择 “*此机器将是拥有实例的数据库分区服务器*” 选项。
   - 该 DB2 服务的帐户是 db2mpp。密码是 db2mpp。
   - 在本地驱动器 C:\SQLLIB 上安装该软件。
6. 在机器 db2test2、db2test3 和 db2test4 上安装 DB2 扩充企业版:
   - 选择 “*此机器将是现存分区数据库系统上的一个新节点*” 选项。
   - 选择 db2test1 作为拥有实例的机器。
   - 该 DB2 服务的帐户是 db2mpp。密码是 db2mpp。
   - 在本地驱动器 C:\SQLLIB 上安装该软件。

下一步是设置 DB2MSCS.CFG 文件，然后运行 DB2MSCS 实用程序。

运行 DB2MSCS 实用程序

要设置 db2test1 机器，执行下列任务:

1. 作为用户 db2nt 登录，它属于 “本地管理员” 组。密码是 db2nt。
2. 设置 DB2MSCS.CFG 文件:

```
#
#  DB2MSCS.CFG for one partitioned database system with
#  multiple MSCS clusters
DB2_INSTANCE=DB2MPP
CLUSTER_NAME=ClusterA
DB2_LOGON_USERNAME=db2mpp
DB2_LOGON_PASSWORD=db2mpp
# Group 1
# for DB2 node 0
GROUP_NAME=DB2NODE0
DB2_NODE=0
IP_NAME=IP Address for DB2
IP_ADDRESS=9.9.9.7
```

```
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
NETNAME_NAME=Network name for DB2
NETNAME_VALUE=DB2WOLF
NETNAME_DEPENDENCY=IP Address for DB2
DISK_NAME=Disk E:
INSTPROF_DISK=Disk E:
#

# Group 2
# for DB2 node 1
GROUP_NAME=DB2NODE1
DB2_NODE=1
DISK_NAME=Disk F:
#

CLUSTER_NAME=ClusterB
# Group 3
# for DB2 node 2
GROUP_NAME=DB2NODE2
DB2_NODE=2
DISK_NAME=Disk E:

#
# Group 4
# for DB2 node 3
GROUP_NAME=DB2NODE3
DB2_NODE=3
DISK_NAME=Disk F:
```

3. 运行 DB2MSCS 实用程序，如下所示:

   ```
   db2mscs -f:DB2MSCS.CFG
   ```

4. 从 db2nt 帐户注销。

最后的步骤是注册两个 MSCS 群集的数据库驱动器映射。

注册 ClusterA 的数据库驱动器映射

要注册 MSCS 群集 ClusterA 的数据库驱动器映射，执行下列任务:

1. 在机器 db2test1 上，作为用户 db2mpp 登录，它是与 DB2 服务相关的帐户。密码是 db2mpp。
2. 要注册数据库驱动器映射，输入下列命令:

   ```
   db2drvmp add 0 F E

   db2drvmp add 1 E F
   ```

3. 将所有 DB2 资源脱机，然后将它们联机。

注册 ClusterB 的数据库驱动器映射

要注册 MSCS 群集 ClusterB 的数据库驱动器映射，执行下列任务:

1. 在机器 db2test3 上，作为用户 db2mpp 登录，它是与 DB2 服务相关的帐户。密码是 db2mpp。
2. 要注册数据库驱动器映射，输入下列命令:

```
db2drvmp add 2 F E
db2drvmp add 3 E F
```

3. 将所有 DB2 资源脱机，然后将它们联机。

在 MSCS 环境中管理 DB2

若正在使用 MSCS 群集，则您的 DB2 实例需要制订关于日常操作、数据库部署和数据库配置的附加计划。为了使 DB2 在任何 MSCS 节点上透明地执行，必须执行附加管理任务。所有与 DB2 相关的操作系统资源必须在所有 MSCS 节点上可用。其中一些操作系统资源不在 MSCS 作用域之内。即，不能将它们定义为 MSCS 资源。必须确保配置每个系统，以便相同的操作系统资源在所有 MSCS 节点上都可用。后面几节描述必须执行的附加工作。

启动和停止 DB2 资源

必须从“群集管理员”工具启动和停止 DB2 资源。有几种方法可用于启动 DB2 实例，如 **db2start** 命令和“控制面板”中的**服务**选项。但是，若未从“群集管理员”启动 DB2，则 MSCS 软件将不知道 DB2 实例的状态。如果 DB2 实例是使用“群集管理员”启动的，并使用 **db2stop** 命令停止，MSCS 软件会将 **db2stop** 命令解释为软件故障并尝试重新启动 DB2。（当前 MSCS 界面不支持对资源状态的通知。）

类似地，若使用 **db2start** 启动 DB2 实例，则 MSCS 不能检测该资源是否联机；若数据库服务器发生故障，则 MSCS 将无法在群集中的故障转移机器上使 DB2 资源联机。

有三个操作可用于 DB2 实例:

联机 此操作等效于使用 **db2start** 命令。若 DB2 已经是活动的，则此操作可以只用于通知 MSCS: DB2 是活动的。将把此操作执行期间的任何错误写入“Windows NT 事件日志”。

脱机 此操作等效于使用 **db2stop** 命令。若实例有任何活动的连接，此操作将失败。这与 **db2stop** 的特性是一致的。

使资源失效
此操作等效于使用指定了 **force** 选项的 **db2stop** 命令。DB2 将所有应用程序与 DB2 系统断开，并停止所有数据库服务器。

运行脚本

在将 DB2 资源联机前和联机后都可运行脚本。这些脚本*必须*驻留在为 DB2INSTPROF 环境变量指定的实例简要表目录中。此目录是 **db2icrt** 命令的 "-p" 参数所指定的目录路径。通过发出如下命令可获取此值:

```
db2set -i:instance_name DB2INSTPROF
```

此文件路径必须在一个群集磁盘上，以便该实例目录在所有群集节点上可用。

这些脚本文件不是必需的，且仅当它们在实例目录中存在时才运行。它们由 "MSCS 群集服务" 在后台启动。这些脚本文件必须重定向标准输出，以便捕捉脚本文件中命令返回的任何信息。该输出不显示到屏幕上。

在一个分区数据库环境中，缺省情况下，实例中的每个数据库分区服务器都使用相同的脚本。若需要区分该实例中的不同数据库分区服务器，则使用 DB2NODE 环境变量的不同赋值以标记特定的节点号（例如，在 db2cpre.bat 和 db2cpost.bat 文件中使用 IF 语句）。

在将 DB2 资源联机前运行脚本

若要在将 DB2 资源联机*前*运行一个脚本，则*必须*将该脚本命名为 db2cpre.bat。在将 DB2 资源联机前，DB2 调用函数，以便从 Windows NT 命令行处理器 (CLP) 启动此批处理文件，然后等待 CLP 完成执行。您可使用此批处理文件执行一些任务，如修改 DB2 数据库管理程序配置。若故障转移系统是受约束的，则可能要更改某些数据库管理程序的参数值，且必须减少 DB2 所占用的系统资源。

位于 db2cpre.bat 脚本中的命令应当同步执行。否则，可能在脚本中的所有任务完成前将 DB2 资源联机，这将引起意外的行为。特别地，**db2cmd** 不应该在 db2cpre.bat脚本中调用，因为这样的话它会启动另一个命令处理器，该命令处理器将与 **db2cmd** 程序异步运行 DB2 命令。

若要在 db2cpre.bat 脚本中使用 DB2 CLP 命令，应将这些命令放入一个文件，并在为 DB2 命令行处理器初始化 DB2 环境的一个程序内，将该文件作为 CLP 批处理文件来运行，然后等待 DB2 命令行处理器的完成。例如:

```
#include <windows.h>

int WINAPI DB2SetCLPEnv_api(DWORD pid);

void main (int argc, char *argv [ ] )
{
      STARTUPINFO         startInfo   = {0};
      PROCESS_INFORMATION pidInfo     = {0};
      char     title  [32]   = "Run Synchronously";
      char     runCmd [64]  =
                              "DB2 -z c:\\run.out -tvf c:\\run.clp";
/* Invoke API to setup a CLP Environment */
```

```
      if ( DB2SetCLPEnv_api (GetCurrentProcessId ()) == 0 ) 1
      {
         startInfo.cb          = sizeof(STARTUPINFO);
         startInfo.lpReserved  = NULL;
         startInfo.lpTitle     = title;
         startInfo.lpDesktop   = NULL;
         startInfo.dwX         = 0;
         startInfo.dwY         = 0;
         startInfo.dwXSize     = 0;
         startInfo.dwYSize     = 0;
         startInfo.dwFlags     = 0L;
         startInfo.wShowWindow = SW_HIDE;
         startInfo.lpReserved2 = NULL;
         startInfo.cbReserved2 = 0;
if ( CreateProcessA( NULL,
                              runCmd, 2
                              NULL,
                              NULL,
                              FALSE,
                              NORMAL_PRIORITY_CLASS CREATE_NEW_CONSOLE,
                              NULL,
                              NULL,
                              &startInfo,
                              &pidInfo))
        {
           WaitForSingleObject (pidInfo.hProcess, INFINITE);
           CloseHandle (pidInfo.hProcess);
           CloseHandle (pidInfo.hThread);
        }
      }
      return;
}
```

1 通过导入库 DB2API.LIB 解析 API DB2SetCLPEnv_api。此 API 设置一个允许调用 CLP 命令的环境。若从 db2cpre.bat 脚本调用此程序，命令处理器将等待 CLP 命令完成。

2 runCmd 是包含 DB2 CLP 命令的脚本文件的名称。

一个名为 db2clpex.exe 的样本程序可在 DB2 安装路径的 MISC 子目录中找到。此可执行文件与提供的示例相似，但它接受 DB2 CLP 命令作为命令行自变量。若要使用此样本程序，将它复制到 BIN 子目录中。可在 db2cpre.bat 脚本中使用此可执行文件，如下所示 （INSTHOME 是实例目录）：

```
db2clpex "DB2 -Z INSTHOME\pre.log -tvf INSTHOME\pre.clp"
```

所有 DB2 ATTACH 命令或 CONNECT 语句应当明确指定用户，否则，将在与群集服务相关的用户帐户下执行它们。CLP 脚本也应当以 TERMINATE 命令完成，以结束 CLP 后台进程。

以下是 db2cpre.bat 文件的一个示例:

```
db2cpre.bat : 1
------------------------
db2clpex "db2 -z INSTHOME\pre-%DB2NODE%.log -tvf INSTHOME\pre.clp" 2 - 5
------------------------

PRE.CLP 6
------------------------
update dbm cfg using MAXAGENTS 200;
get dbm cfg;
terminate;
------------------------
```

1 db2cpre.bat 脚本在与“群集服务”相关的用户帐户下执行。若必需 DB2 操作，则与“群集服务”相关的用户帐户必须是一个有效的 SQL 标识符，它由 DB2 定义。

2 INSTHOME 是实例目录。

3 对于每个节点，日志文件的名称必须不同，以避免当两个逻辑节点同时联机时可能产生的文件争用。

4 db2clpex.exe 是一个样本程序，它使用命令行自变量来指定要调用的 CLP 命令。

5 必须使 db2clpex.exe 样本程序在所有 MSCS 群集节点上可用

6 此示例中的 CLP 命令设置代理程序数的限制。

将 DB2 资源联机后运行脚本

若要在将 DB2 资源联机*后*运行脚本，则*必须*将脚本命名为 db2cpost.bat。当 DB2 资源已成功联机后，将从 MSCS 异步运行该脚本。可在此脚本中使用 **db2cmd** 命令，以执行 DB2 CLP 脚本文件。使用 **db2cmd** 命令的 ″-c″ 参数，来指定一旦该任务完成后实用程序应关闭所有窗口。例如:

```
db2cmd -c db2 -tvf mycmds.clp
```

″-c″ 参数必须是 **db2cmd** 命令的第一个自变量，因为它防止后台存在孤立的命令处理器。

若要在 DB2 资源进行故障转移且成为活动后立即执行数据库活动，则 db2cpost.bat 脚本是有用的。例如，您可重新启动或激活该实例中的数据库，以便它们准备好用于用户访问。

以下是 db2cpost.bat 脚本的一个示例:

```
db2cpost.bat 1
------------------------
db2cmd -c db2 -z INSTHOME\post-%DB2NODE%.log -tvf INSTHOME\post.clp 2 - 4
------------------------

POST.CLP 5
```

```
-----------------------
restart database SAMPLE;
connect reset;
activate database SAMPLE;
terminate;
-----------------------
```

1 db2cpost.bat 脚本在与“群集服务”相关的用户帐户下运行。若必需 DB2 操作，则与“群集服务”相关的用户帐户必须是一个有效的 SQL 标识符，它由 DB2 定义。

2 INSTHOME 是实例目录。

3 对于每个节点，日志文件的名称必须不同，以避免当两个逻辑节点同时联机时可能产生的文件争用。

4 可使用 **db2cmd** 命令，因为 db2cpost.bat 脚本可异步运行。必须使用 ″-c″ 参数来终止命令处理器。

5 此示例中的 CLP 脚本包含重新启动和激活数据库的命令。此脚本将数据库返回到紧接在数据库管理程序启动后的一个活动状态。在一个分区数据库系统中，您应当除去 ACTIVATE DATABASE 命令，因为可将多个 DB2 资源同时联机。RESTART DATABASE 命令可能失败，因为另一个节点正在激活该数据库。若发生这种情况，重新运行该脚本，以确保正确地重新启动了数据库。

数据库注意事项

当创建一个数据库时，确保数据库路径指向一个共享磁盘。这允许在所有 MSCS 节点上查看该数据库。所有日志和其他数据库文件必须也指向群集磁盘，以便 DB2 成功地进行故障转移。若不执行这些步骤，则 DB2 系统将发生故障，因为在 DB2 看来好象是文件已删除或不可用。

还要确保设置数据库管理程序和数据库配置参数，以便在任何一个 MSCS 节点上支持 DB2 所占用的系统资源量。应将 *autorestart* 数据库配置参数设置为 ON，以便在进行故障转移时的第一个数据库连接将使数据库处于一致的状态。（*autorestart* 的缺省设置为 ON。）也可使用 db2cpost.bat 脚本来重新启动并激活数据库，使该数据库处于就绪状态。此方法更好，因为这样不会依赖于 *autorestart*，且使数据库处于就绪状态，而不受用户连接请求影响。

用户和组支持

DB2 依赖于 Windows NT 进行用户认证和组支持。要使一个 DB2 实例以无缝方式从一个 MSCS 节点到另一个进行故障转移，每个 MSCS 节点都必须对相同的 Windows NT 安全性数据库具有访问权。您可使用“Windows NT 域安全性”来达到此目的。

在“域安全性”数据库中定义所有 DB2 用户和组。MSCS 节点必须是此域的成员，或此域必须是“可信赖域”。然后，DB2 将使用“域安全性”数据库来进行认证和组支持，与 DB2 正在哪个 MSCS 节点上运行无关。

若您使用的是本地帐户，必须在每个 MSCS 节点上复制该帐户。不建议使用此方法，因为它容易出错，且需要双重维护。

“DCE 安全性”也是一种受支持的认证模式，只要所有 MSCS 节点为相同 DCE 单元中的客户机。

您应当使 MSCS 服务与符合 DB2 命名约定的一个用户帐户相关。这允许 MSCS 服务对 DB2 执行在 `db2cpre.bat` 和 `db2cpost.bat` 脚本中可能必需的操作。

有关 Windows NT 用户和组支持的更多信息，请参阅*《管理指南: 实现》*一书中的『DB2 Windows NT 版的用户认证』。

通信注意事项

DB2 在 MSCS 环境中支持两个 LAN 协议:

- TCP/IP
- NetBIOS

支持 TCP/IP，因为它是受支持的群集资源类型。要允许 DB2 使用 TCP/IP 作为一个分区数据库系统的通信协议，创建一个“IP 地址”资源，然后将它置于 DB2 资源所在的同一个组中，该组表示您要用作远程应用程序的协调程序节点的数据库分区服务器。然后，使用“群集管理员”工具创建从属性，以确保在启动 DB2 资源前 IP 资源是联机的。从而，DB2 客户机可编目 TCP/IP 节点目录条目以使用此 TCP/IP 地址。

与 *svcename* 数据库管理程序配置参数相关的 TCP/IP 端口必须保留，以供参与该实例的所有机器上的 DB2 实例使用。与该端口号码相关的服务名也必须在所有机器上的 `services` 文件中是相同的。

虽然 NetBIOS 不是受支持的群集资源，但是可使用 NetBIOS 作为 LAN 协议，因为该协议确保 NetBIOS 名在 LAN 上是唯一的。当 DB2 注册 NetBIOS 名时，NetBIOS 确保该名称没有在 LAN 上使用。在一个故障转移方案中，当将 DB2 从一个系统移到另一个系统时，将从 MSCS 群集中的一个伙伴机器上注销 DB2 使用的 *nname*，而在另一台机器上注册。

DB2 NetBIOS 支持使用“NetBIOS 帧”(NBF)。此协议堆栈可与不同的逻辑适配卡号 (LANA) 相关。要确保 NetBIOS 对服务器的一致性访问，与 NBF 协议堆栈相

关的 LANA 应当在所有群集节点上相同。可使用“控制面板”中的**网络**选项来配置它。您应当将 NBF 与 LANA 0 相关，因为它是 DB2 期望的缺省设置。

系统时间注意事项

DB2 使用系统时间来记录特定操作的时间戳记。所有参与 DB2 故障转移的 MSCS 节点必须使系统时区和系统时间同步，以确保 DB2 在所有机器上的操作取得一致。

使用“控制面板”对话中的**日期／时间**选项，来设置系统时区。MSCS 具有一个时间服务，当 MSCS 节点连接为一个群集时，该服务用来校准日期和时间。但是，该时间服务只是每 12 个小时校准时间一次，若在一个系统上更改了时间，而在将时间同步前 DB2 进行了故障转移，这可能引起问题。

若在 MSCS 群集的一个节点上更改了时间，则应当使用如下命令在其他群集节点上人工校准时间:

```
net time /set /y \\remote_node
```

其中 *remote_node* 是群集节点的机器名。

分区数据库环境中的管理服务器和控制中心注意事项

“DB2 管理服务器”是在安装 DB2 通用数据库期间（可选地）创建的。它不是分区数据库系统。控制中心使用“管理服务器”提供的服务，来管理 DB2 实例和数据库。

在一个分区数据库系统中，一个 DB2 实例可驻留在多个 MSCS 节点上。这意味着一个 DB2 实例必须在控制中心下的多个系统上编目，以便无论该 DB2 实例在哪个 MSCS 节点上是活动的，该实例都是可访问的。

“管理服务器”实例目录不是共享的。您必须将“管理服务器”目录中的所有用户定义文件镜像到所有 MSCS 节点上，以对所有 MSCS 节点提供相同级别的管理。特别是，必须使用户脚本和调度的可执行文件在所有节点上可用。还必须确保调度的活动在一个 MSCS 群集中的所有机器上进行调度。

或者，代替在所有机器上复制“管理服务器”的方法是，您可能要使“管理服务器”进行故障转移。对于以下示例，假设该群集中有两个 MSCS 节点，它们称为 MACH0 和 MACH1。 MACH0 具有对“管理服务器”将使用的一个群集磁盘的访问权。还假设 MACH0 和 MACH1 都具有“管理服务器”。您将执行下列步骤，以使“管理服务器”高度可用:

1. 在每台机器上调用 **db2admin stop** 命令，以同时将两台机器上的“管理服务器”停止。

2. 在所有管理客户机上，使用 UNCATALOG NODE 命令，以取消编目对 MACH0 和 MACH1 上的“管理服务器”的所有引用。（可在客户机上使用 LIST NODE DIRECTORY 命令，以确定是否存在对“管理服务器”的任何引用。）
3. 通过从 MACH1 上调用 **db2admin drop** 命令，将“管理服务器”从 MACH1 上删除。（若两台机器上都具有“管理服务器”，则只执行此步骤。）
4. 从 MACH0 上发出 **db2admin** 命令，确定“管理服务器”的名称。（缺省名称是 DB2DAS00。）
5. 使用 DB2MSCS 实用程序，为“管理服务器”设置故障转移支持。这必然会在 MSCS 上创建一个名为 DB2DAS00 的 DB2 资源，它依赖于 IP 和磁盘资源。（若您具有一个接管配置，应当将该资源放在为 NODE0 保存 DB2 资源的组中。）此资源将用作支持“管理服务器”的 MSCS 资源。DB2MSCS.ADMIN 文件应是如下所示的格式:

```
#
# db2mscs.admin for Administration Server
# run db2mscs -f:db2mscs.admin
#
DB2_INSTANCE=DB2DAS00
CLUSTER_NAME=CLUSTERA
DB2_LOGON_USERNAME=db2admin
DB2_LOGON_PASSWORD=db2admin
# put Administration server in the same group as DB2 Node 0
GROUP_NAME=DB2NODE0 1
DISK_NAME=DISK E:
INSTPROF_DISK=DISK E:
IP_NAME= IP Address for Administration Server
IP_ADDRESS=9.9.9.8
IP_SUBNET=255.255.255.0
IP_NETWORK=Ethernet
```

   1 此组可与现存组相同。这样，就不需要附加磁盘来用于实例简要表目录。

6. 在 MACH1 上调用如下命令，以将 DB2DAS00 设置为“管理服务器”：

```
db2set -g db2adminserver=DB2DAS00
```

7. 在 MACH0 上，通过 Services 程序修改 DB2DAS00 的启动特性，以便人工而不是自动地启动它，因为 DB2DAS00 现在由 MSCS 控制。

当使“管理服务器”可进行故障转移时，所有远程访问都应当使用 MSCS IP 资源来与“管理服务器”通信。现在，“管理服务器”将具有下列特性:

- “管理服务器”实例目录将使用“管理服务器”来进行故障转移。
- 客户机将只编目单个节点以与“管理服务器”通信，而不管它在哪个 MSCS 节点上是活动的。
- 只需要对“管理服务器”调度作业一次。

- 仅当“管理服务器”在本地实例所在的相同 MSCS 节点上活动时，“管理服务器”才可控制本地实例。
- 若“群集服务”不是活动的，则“管理服务器”不可访问。

限制

当在 MSCS 环境中运行 DB2 时:

- 除非共享磁盘在两个 MSCS 节点上具有相同的物理磁盘号，否则，不能在共享磁盘上使用物理 I/O。可使用逻辑 I/O，因为磁盘是使用分区标识符来访问的。
- 必须为 MSCS 支持配置所有 DB2 资源。若没有这样做，DB2 运行期间将发生系统错误（在缺少系统资源的情况下，DB2 不能正确操作）。例如，若数据库日志没有在 MSCS 共享磁盘上，则 DB2 不能重新启动数据库。
- 必须从“群集管理员”工具管理 DB2 实例。 MSCS 将把用于启动和停止数据库管理程序的其他方法看作是软件不一致性。例如，若您使用 MSCS 来启动 DB2，而使用 **db2stop** 命令来停止 DB2，则 MSCS 将把这种情况检测为软件错误，并重新启动实例。这还意味着您不应当使用控制中心来启动和停止 DB2。
- 要卸装 DB2，必须首先停止 MSCS。

第8章 Solaris 操作环境中的高可用性

Solaris 操作环境中的高可用性可通过将 DB2 与 Sun Cluster 2.x (SC2.x)、Sun Cluster 3.0 (SC3.0) 或 Veritas Cluster Server (VCS) 配合使用来获得。关于 Sun Cluster 3.0 的更多信息，可参见名为 ″DB2 and High Availability on Sun Cluster 3.0″ 的白皮书，可从 ″DB2 UDB and DB2 Connect Online Support″ Web 站点 (http://www.ibm.com/software/data/pubs/papers/) 获得。关于 VERITAS Cluster Server 的信息，参见名为 ″DB2 and High Availability on VERITAS Cluster Server″ 的白皮书，可从 ″DB2 UDB and DB2 Connect Online Support″ Web 站点获得。

本章详细描述 DB2 如何使用 Sun Cluster 2.x (SC2.x) 来达到高可用性，包括高可用性代理程序的说明，该代理程序用作两个软件产品之间的中介（参见图31）。

图 31. DB2、Sun Cluster 2.x 和高可用性

高可用性

提供数据服务的计算机系统包含许多独特的部件，每个部件都有一个关联的 “平均无故障时间” (MTBF)。MTBF 是部件保持可用的平均时间。高质量硬盘驱动器的 MTBF 大约是一百万小时（大约 114 年）。虽然这看起来象是很长一段时间，但每 200 个硬盘中，就有一个有可能在 6 个月内失效。

虽然有很多方法可以提高数据服务的可用性，但最常用的方法还是 HA 群集。当用于高可用性时，群集由两台或更多机器、一组专用网络接口、一个或多个公用网络接口以及一些共享磁盘组成。这种特殊的配置允许将数据服务从一台机器移至另一机器。通过将数据服务移至群集中的另一机器，应该能够继续提供对其数据的访问。将数据服务从一台机器移至另一机器称为*故障转移*，第230页的图32 对其进行了说明。

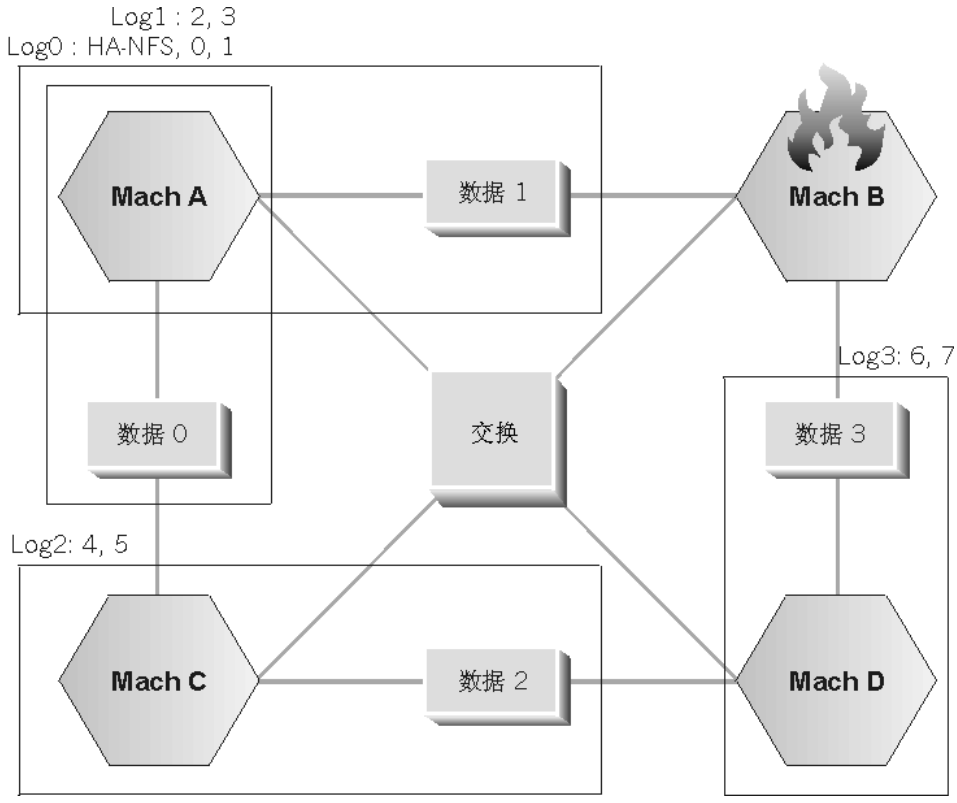


图 32. 故障转移

专用网络接口用来在群集中的机器之间发送心跳消息和控制消息。公用网络接口用来直接与 HA 群集的客户机通信。HA 群集中的磁盘与群集中的两台或更多机器相连，因此，当一台机器失效时，另一机器可以访问它们。

在 HA 群集上运行的数据服务有一个或多个逻辑公用网络接口和一组磁盘与其关联。HA 数据服务的客户机只通过 TCP/IP 与该数据服务的逻辑网络接口相连。如果执行故障转移，则该数据服务及其逻辑网络接口和磁盘组便移至另一机器。

HA 群集的其中一个好处是可以在没有支持人员辅助的情况下恢复数据服务，并且，可以随时这样做。另一个好处是冗余。群集中的所有部件都应该是冗余的，包括机器本身。群集应当能够经受住任何单个故障点。

虽然高度可用的数据服务在本质上可以有非常大的区别，但它们还是有一些公共的需求。高度可用数据服务的客户机期望该数据服务的网络地址和主机名保持不变，并期望能够以同一方式发出请求，而不考虑该数据服务具体是在哪一台机器上。

假定一个 Web 浏览器正在访问高度可用 Web 服务器。请求是使用 URL（统一资源定位器）发出的，该 URL 包含主机名和 Web 服务器上文件的路径。浏览器期望该主机名和路径在 Web 服务器进行故障转移后保持不变。如果浏览器从 Web 服务器下载文件，而该服务器正在进行故障转移，则该浏览器将需要重新发出该请求。

数据服务的可用性是通过数据服务可供其用户使用的时间量测量的。最常用的可用性计量单位是“运行时间”的百分比；此百分比通常用数字“9”来表示:

```
99.99% => 服务（最多）停止 52.6 分钟／年
99.999% => 服务（最多）停止 5.26 分钟／年
99.9999% => 服务（最多）停止 31.5 秒／年
```

当设计和测试 HA 群集时:

1. 确保群集管理员熟悉系统并了解发生故障转移时应发生的情况。
2. 确保群集的每一部分都确实是冗余的，当它失效时，可以被快速替换。
3. 强制一个测试系统在受控环境中失效，并确保它每次都能正确地进行故障转移。
4. 跟踪每次故障转移的原因。虽然这应该不会经常发生，但找出任何导致群集不稳定的问题非常重要。例如，如果群集的一个部分一个月导致了 5 次故障转移，则找出原因并修正它。
5. 当发生故障转移时，务必通知该群集的支持人员。
6. 不能使群集过载。确保其余的系统在发生故障转移之后，仍能够处理适量的工作负荷。
7. 经常检查容易引起故障的部件（如磁盘），以便可以在发生问题之前进行替换。

故障容错和连续可用性

另一种提高数据服务可用性的方法是故障容错。*故障容错*机器内置了其所有冗余，应该能够经受住任何部件（包括 CPU 和内存）的单一故障。故障容错机器最常用在小型市场中，其实现成本比较高。机器位于不同地理位置的 HA 群集还有一个优点，即可以从仅影响那些位置的一部分的灾难中恢复。

*连续可用性*是一个基于高可用性的步骤。它保护其客户机不受已计划的和未曾计划的停机时间的影响。有了连续可用性配置，即使其中一台提供数据服务的机器失效或为了进行维护而被关闭，客户机也完全不受影响。连续可用性配置非常复杂，实现成本更高。

由于 HA 群集可伸缩、易于使用以及实现成本相对较低，所以它是最常见的提高可用性的解决方案。

Sun Cluster 2.2

Sun Cluster 2.2 (SC2.2) 是 Sun Microsystems 的群集和高可用性产品。SC2.2 在一个群集中最多支持四台机器。通过使用四个 Ultra Enterprise 10000, 一个群集最多可以带有 256 个 CPU 和 256 GB 内存。

受支持的系统

系统	UltraSPARC	内存容量	I/O
Ultra Enterprise 1	1	64MB-1GB	3 SBus
Ultra Enterprise 2	1-2	64MB-2GB	4 SBus
Ultra Enterprise 450	1-4	32MB-4GB	10 PCI
Ultra Enterprise 3000	1-6	64MB-6GB	9 SBus
Ultra Enterprise 4000	1-14	64MB-14GB	21 SBus
Ultra Enterprise 5000	1-14	64MB-14GB	21 SBus
Ultra Enterprise 6000	1-30	64MB-30GB	45 SBus
Ultra Enterprise 10000	1-64	512MB-64GB	64 SBus

代理程序

Sun Cluster 软件包括许多受 SC2.2 产品支持并随其一起交付的高可用性代理程序。其他 HA 代理程序, 如用于 DB2 的代理程序, 不是由 Sun 开发的, 它们未随 Sun Cluster 软件一起交付。HA 代理程序 DB2 版是 DB2 附带的, 受 IBM 支持, 并由 DB2 免费提供。

Sun Cluster 软件通过提供注册特定方法（脚本或程序）的机会来使用高度可用数据服务, 这些方法与 Sun Cluster 软件的各种部件相对应。利用这些方法, SC2.2 软件便可以控制数据服务, 而无需对其进行深入了解。这些方法包括:

START
用来在逻辑网络接口联机之前启动一部分数据服务。

START_NET
用来在逻辑网络接口联机之后启动一部分数据服务。

STOP 用来在逻辑网络接口脱机之后停止一部分数据服务。

STOP_NET
用来在逻辑网络接口脱机之前停止一部分数据服务。

ABORT
与 STOP 方法类似, 不同的是它刚好在群集软件关闭机器之前运行。在此

情况下，该机器的“健康”是有疑问的，在该机器关闭之前，数据服务可能想要执行“临终愿望”请求。在逻辑网络接口脱机后运行。

ABORT_NET

与 ABORT 方法类似，不同的是它在逻辑网络接口脱机之前运行。

FM_INIT

用来初始化故障监控程序。

FM_START

用来启动故障监控程序。

FM_STOP

用来停止故障监控程序。

FM_CHECK

由 **hactl** 命令调用。返回对应的数据服务的当前状态。

DB2 代理程序包含下列脚本：START_NET、STOP_NET、FM_START 和 FM_STOP。群集重新配置期间，不运行下列脚本：ABORT、ABORT_NET 和 FM_CHECK。

高可用性代理程序由这些方法中的一个或多个组成。这些方法通过 **hareg** 命令向 SC2.2 注册。一旦注册，Sun Cluster 软件就将调用相对应的方法来控制数据服务。

重要的是记住不能调用服务的 ABORT 和 STOP 方法。这些方法用于数据服务的受控关闭，如果机器失效，而未调用这些方法，则该数据服务必须能够恢复。

要获取更多信息，参考 Sun Cluster 文档。

逻辑主机

SC2.2 软件使用了逻辑主机概念。*逻辑主机*由一组磁盘和一个或多个逻辑公用网络接口组成。高度可用数据服务与一个逻辑主机关联，并需要该逻辑主机的磁盘组中的磁盘。逻辑主机可以驻留在群集中不同的机器上，并可以“借用”运行它们的机器的 CPU 和内存。

逻辑网络接口

与其他基于 UNIX 的操作系统一样，除用于网络接口的主 IP 地址外，Solaris 还有能力具有附加的 IP 地址。附加的 IP 地址驻留在一个逻辑接口上，其驻留方式与驻留在物理网络接口上的主 IP 地址的方式相同。下面是群集中两台机器上的逻辑接口的一个示例。共有两个逻辑主机，它们当前都在机器 ″thrash″ 上。

```
scadmin@crackle(202)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 289966 0 289966 0 0 0
```

```
hme0 1500 9.21.55.0 9.21.55.98 121657 6098 764122 0 0 0
scid0 16321 204.152.65.0 204.152.65.1 489307 0 476479 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.33 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.17 347317 0 348073 0 0 0
```

1. lo0 是反馈接口
2. hme0 是公用网络接口（以太网）
3. scid0 是第一个专用网络接口（SCI 或可缩放 Coherent 接口）
4. scid0:1 是 Sun Cluster 软件内部使用的逻辑网络接口
5. scid1 是第二个专用网络接口

```
scadmin@thrash(203)# netstat -in
Name Mtu Net/Dest Address Ipkts Ierrs Opkts Oerrs Collis Queue
lo0 8232 127.0.0.0 127.0.0.1 1128780 0 118780 0 0 0
hme0 1500 9.21.55.0 9.21.55.92 1741422 5692 757127 0 0 0
hme0:1 1500 9.21.55.0 9.21.55.109 0 0 0 0 0 0
hme0:2 1500 9.21.55.0 9.21.55.110 0 0 0 0 0 0
scid0 16321 204.152.65.0 204.152.65.2 476641 0 489476 0 0 0
scid0:1 16321 204.152.65.32 204.152.65.34 0 0 0 0 0 0
scid1 16321 204.152.65.16 204.152.65.18 348199 0 347444 0 0 0
```

1. hme0:1 是逻辑主机的逻辑网络接口
2. hme0:2 是另一逻辑主机的逻辑网络接口

逻辑主机可以有一个或多个逻辑接口与其关联。这些逻辑接口随逻辑主机一起在机器间移动，它们用来访问与该逻辑主机关联的数据服务。因为这些逻辑接口随逻辑主机一起移动，所以客户机可以独立于数据服务所在的机器来访问数据服务。

高度可用数据服务应联编至 TCP/IP 地址 INADDR_ANY。这确保系统上的每个 IP 地址都可以接受数据服务连接。相反，若数据服务联编至某一特定 IP 地址，则它必须联编至与提供数据服务的逻辑主机关联的逻辑接口。联编至 INADDR_ANY 还有一个好处，即当系统上新增了数据服务所需的新 IP 地址时，无需重新联编至该地址。

注: HA 实例的客户机应当使用逻辑主机的逻辑 IP 地址的主机名来编目数据库。它们决不应使用一台机器的主主机名，因为不能保证 DB2 将在该机器上运行。

磁盘组和文件系统

数据服务的磁盘与组中的一个逻辑主机关联。如果群集正在运行 Sun StorEdge Volume Manager (Veritas)，则 Sun Cluster 软件使用 Veritas ″vxdg″ 实用程序来为每个逻辑主机导入和导出磁盘组。下面是两个逻辑主机 ″log0″ 和 ″log1″ 的磁盘组的示例，这两个逻辑主机驻留在名为 ″thrash″ 的机器上。名为 ″crackle″ 的机器当前并未存放任何逻辑主机。

```
scadmin@crackle(206)# vxdg list
NAME STATE ID
rootdg enabled 899825206.1025.crackle

scadmin@thrash(205)# vxdg list
NAME STATE ID
rootdg enabled 924176206.1025.thrash
data0 enabled 925142028.1157.crackle=
data1 enabled 899826248.1108.crackle
```

磁盘组 ″data0″ 和 ″data1″ 分别与逻辑主机 ″log0″ 和 ″log1″ 相对应。磁盘组 ″data0″ 可以从 ″thrash″ 中导出，方法是运行

```
vxdg deport data0
```

并可导入 ″crackle″，方法是运行

```
vxdg import data1
```

这是由 Sun Cluster 软件自动完成的，不应在活动群集上人工执行。

每个磁盘组都包含许多可以由群集中的两台或更多机器共享的磁盘。逻辑主机只能移至另一能够对其拥有的磁盘组中的磁盘进行物理访问的机器。

有两个文件控制着每个逻辑主机的文件系统:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
/etc/opt/SUNWcluster/conf/hanfs/dfstab.<logical_host>
```

其中，*logical_host* 是关联的逻辑主机的名称。

`vfstab` 文件与 `/etc/vfstab` 文件类似，不同的是它包含在为逻辑主机导入磁盘组之后要安装的文件系统的条目。`dfstab` 文件与 `/etc/dfs/dfstab` 文件类似，不同的是它包含要通过 HA-NFS 为逻辑主机导出的文件系统的条目。每台机器都有这些文件的它们自己的副本，您应仔细确保这些文件在群集中每台机器上的内容都相同。

注: 逻辑主机的 `vfstab` 和 `dfstab` 文件的路径容易令人误解，因为它们包含目录 `hanfs`。只有逻辑主机的 `dfstab` 文件才用于 HA-NFS。即使未配置 HA-NFS，也会使用 `vfstab` 文件。

下面是来自在相互接管配置中运行 “DB2 通用数据库扩充企业版” (EEE) 的群集的示例:

```
scadmin@thrash(217)# ls -l /etc/opt/SUNWcluster/conf/hanfs
total 8
-rw-r--r-- 1 root build 173 Apr 14 15:01 dfstab.log0
-rw-r--r-- 1 root build 316 Apr 26 12:07 vfstab.log0
-rw-r--r-- 1 root build 389 Apr 13 21:04 vfstab.log1
```

```
scadmin@thrash(218)# cat dfstab.log0
share -F nfs -o root=crackle:thrash:\
jolt:bump:crackle.torolab.ibm.com:thrash.torolab.ibm.com:\
jolt.torolab.ibm.com:bump.torolab.ibm.com /log0/home
```

主机（被授予安装文件系统 /log0/home 的许可权）来自群集中每台机器上的所有网络接口（逻辑接口和物理接口）。文件系统使用 root 用户许可权导出。

```
scadmin@thrash(220)# cat vfstab.log0
#device to mount             device  to fsck               mount
#                                                         point

/dev/vx/dsk/data0/data1-stat /dev/vx/rdsk/data0/data1-stat /log0
/dev/vx/dsk/data0/vol01      /dev/vx/rdsk/data0/vol01      /log0/home
/dev/vx/dsk/data0/vol02      /dev/vx/rdsk/data0/vol02      /log0/data

scadmin@thrash(221)# cat vfstab.log1
#device to mount             device  to fsck               mount
#                                                         point
/dev/vx/dsk/data1/data1-stat /dev/vx/rdsk/data1/data1-stat /log1
/dev/vx/dsk/data1/vol01      /dev/vx/rdsk/data1/vol01      /log1/home
/dev/vx/dsk/data1/vol02      /dev/vx/rdsk/data1/vol02      /log1/data
/dev/vx/dsk/data1/vol03      /dev/vx/rdsk/data1/vol03      /log1/data1



FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -


FS   fsck mount   options
type pass at boot

ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
ufs  2    no      -
```

vfstab.log0 文件包含 /log0 目录下的文件系统的三个有效条目。注意，逻辑主机 log0 的文件系统使用逻辑卷设备，这些逻辑设备是与该逻辑主机关联的磁盘组 data0 的一部分。

vfstab 文件中的文件系统按从上到下的次序安装，因此重要的是确保以正确的次序列示文件系统。安装在特定文件系统下面的那些文件系统应列示在该文件系统下面。逻辑主机所需的实际文件系统取决于数据服务的需要，与这些示例会有很大的不同。

故障转移期间，SC2.2 软件负责确保与一个逻辑主机关联的磁盘组和逻辑接口跟随着该逻辑主机在群集中的机器间移动。在故障转移之后，高度可用数据服务预期这些资源中至少有一个在新系统上可用。事实上，许多数据服务甚至不知道它们是高度可用的，在故障转移之后，必须使这些资源“看起来”丝毫未变。

控制方法

注册控制方法应使用

```
hareg(1m)
```

在注册 HA 服务之后，SC2.2 就负责在群集重新配置或故障转移期间的适当时候调用已对 HA 服务注册的方法。

在群集重新配置（受控故障转移）期间，（以给出的次序）执行下列操作：如果机器崩溃，将不执行步骤 5c 之前的操作。（有关群集重新配置的更多信息，参考 SC2.2 文档。）

1. 运行 FM_STOP 方法。
2. 运行 STOP_NET 方法。
3. 使逻辑主机的逻辑接口脱机。
   - ifconfig hme0:1 0.0.0.0 down
4. 运行 STOP 方法。
5. 移动磁盘组和文件系统。
   a. 卸装逻辑主机文件系统。
   b. 在一台机器上对磁盘组执行 vxdg deport 操作。

   - - 如果机器崩溃，则只运行下面的步骤 - -

   c. 在另一机器上对磁盘组执行 vxdg import 操作。
   d. 对逻辑主机文件系统执行 fsck。
   e. 安装逻辑主机文件系统。
6. 运行 START 方法。
7. 使逻辑主机的逻辑接口联机。
   - ifconfig hme0:1 <ip address> up
8. 运行 START_NET 方法。
9. 运行 FM_INIT 方法。
10. 运行 FM_START 方法。

控制方法是使用下列命令行自变量运行的:

```
METHOD <logical hosts being hosted> <logical hosts not being hosted> <time-out>
```

第一个自变量是当前存放的逻辑主机的逗号定界列表，第二个自变量是未存放的逻辑主机的逗号定界列表。最后一个自变量是该方法的超时，即 SC2.2 软件在异常终止该方法之前允许它运行的时间量。

磁盘和文件系统配置

SC2.2 支持两个卷管理器: Sun StorEdge Volume Manager (Veritas) 和 Solstice Disk Suite。虽然这两个软件的表现都很优良，但 StorEdge Volume Manager 在群集环境中更有优势。在某些群集配置中，群集中每台机器的磁盘罩的控制器编号可以不同。若控制器编号不同，则该控制器的磁盘设备的路径也将不同。因为 Disk Suite 直接使用磁盘设备路径，所以它在此环境中不能很好地工作。StorEdge Volume Manager 使用磁盘本身，而不考虑控制器编号，因而即使控制器编号不同也不受影响。

因为 HA 的目标是提高数据服务的可用性，所以重要的是确保所有文件系统和磁盘设备都被镜像，或采用 RAID 配置。这能预防因磁盘失效而导致的故障转移，并能提高群集的稳定性。

HA-NFS

当跨多台机器配置一个实例时，DB2 UDB EEE 需要一个共享文件系统。典型的 DB2 UDB EEE 配置是通过 NFS 从一台机器导出主目录，并将其安装在所有参与 EEE 实例的机器上。对于相互接管配置，DB2 UDB EEE 依靠 HA-NFS 来提供共享的、高度可用的文件系统。其中一个逻辑主机通过 HA-NFS 导出一个文件系统，然后，群集中的每台机器安装该文件系统作为 EEE 实例的主目录。有关 HA-NFS 的更多信息，参考 Sun Cluster 文档。

cconsole 和 ctelnet 实用程序

SC2.2 附带提供了两个很有用的实用程序: `cconsole` 和 `ctelnet`。这些实用程序可以用来同时向群集中的数台机器发出单一命令。使用这些实用程序编辑配置文件可以确保该文件在群集中的所有机器上保持完全相同。这些实用程序还可用来以完全相同的方式在每台机器上安装软件。有关这些实用程序的更多信息，参考 Sun Cluster 文档。

校园群集和大陆群集

当一个群集中的数台机器不在同一建筑物中时，该群集被称为*校园群集*。校园群集对于将建筑本身作为单个故障点而除去时很有用。例如，如果群集中的机器全都在同一建筑物中，而该建筑物烧毁了，整个群集都受到影响。但如果机器在不同的建筑物中，即使有一座建筑物烧毁了，群集也可以幸免于难。

*大陆群集*的机器分布在不同的城市中。在这种情况下，目的是要将某个地理区域作为单个故障点而除去。这种类型的群集提供了针对灾难事件（如地震和海啸）的保护。

目前，Sun Cluster 可支持机器最多相距 10 公里（即大约 6 英里）。这使校园群集对于那些需要在两个不同站点之间进行高速连接的人而言成为可行的选项。群集需要两个专用互连，并需要许多光纤电缆用于共享磁盘。两个站点之间进行高速连接的成本可能会抵消掉它的优点。

常见问题

SC2.2 软件使用“群集配置数据库”（即 CCD(4)）来为群集配置提供单群集范围的储存库。CCD 有一个专用 API，它存储在 /etc/opt/SUNWcluster/conf 目录下面。在极为罕见的情况下，CCD 会变得不同步，可能需要修复。在这种情况下，最好的 CCD 修复方法是从副本复原它。

要备份 CCD，需在群集中的所有机器上关闭群集软件，“压缩”(tar) /etc/opt/SUNWcluster/conf 目录，并将 tar 文件存储在安全的地方。如果在进行备份时未关闭群集软件，则复原 CCD 时可能会发生问题。在每次更改群集配置时都执行新备份，以确保副本总是最新的。要复原 CCD，需在群集中的所有机器上关闭群集软件，将 conf 目录移至 conf.old，并“解压”(untar) 副本。之后，便可以使用新的 CCD 启动群集。

DB2 注意事项

本节包括下列主题:

连接 HA 实例的应用程序

依赖高度可用 DB2 实例的应用程序必须能够在发生故障转移时进行重新连接。因为逻辑主机的主机名和 IP 地址保持不变，所以无需连接另一主机名或重新编目数据库。

假设有一个群集带有两台机器和一个 “DB2 通用数据库企业版” (EE) 实例。EE 实例通常驻留在群集中的其中一台机器上。HA 实例的客户机将连接与该 HA 实例关联的逻辑主机的逻辑 IP 地址（或主机名）。

依照 HA 客户机，有两种类型的故障转移。其中一种类型的故障转移在存放 HA 实例的机器崩溃时发生。另一种类型的故障转移在 HA 有机会平缓关闭时发生。

若一台机器崩溃并关闭 HA 实例，则与数据库的现存连接和新连接都将挂起。连接挂起的原因是网络上没有任何机器具有客户机用于该数据库的 IP 地址。若数据库平缓关闭，则 `db2stop force` 断开与数据库的现存连接，并返回错误消息。

故障转移期间，与数据库关联的逻辑 IP 地址脱机，或者是因为 SC2.2 软件使其脱机，或者是因为存放逻辑主机的机器崩溃。此时，与该数据库的任何新连接都将被短暂挂起。

在 DB2 启动之前，与该数据库关联的逻辑 IP 地址最终在另一机器上联机。在此阶段，与该数据库的连接不被挂起，但将接收到通信错误，因为 DB2 尚未在系统上启动。仍与该数据库相连的 DB2 客户机也将开始接收通信错误。虽然那些客户机仍相信它们是连接着的，但开始存放该逻辑 IP 地址的机器却不了解任何现存的连接。连接仅仅是进行复位，DB2 客户机接收到通信错误。片刻之后，DB2 将在该机器上启动，现在可以成功地连接该数据库了。此时，数据库可能不一致，客户机可能必须要等待它恢复。

当针对 HA 环境设计应用程序时，不必为数据库连接的挂起阶段编写特殊的代码。连接只是在 Sun Cluster 软件移动逻辑 IP 地址时被短暂挂起。在此阶段，在 Sun Cluster 上运行的任何数据服务都将遇到同一连接挂起问题。无论数据库是如何关闭的，客户机都将接收到错误消息，必须尝试重新连接，直到成功为止。从客户机的角度而言，就象是 HA 实例先是关闭，然后在同一机器上重新打开。在受控故障转移中，在客户机看来，就象是 HA 实例被强制关闭，而客户机稍后可以重新连接同一机器上的数据库。在不受控故障转移中，在客户机看来，就象是数据库服务器崩溃，然后很快在同一机器上重新启动。

EE 和 EEE 实例的磁盘布局

DB2 期望它所需的磁盘设备或文件系统在群集中的每台机器上看起来都是相同的。要确保发生这种情况，应将必需的磁盘或文件系统配置为：它们跟随与 HA 实例关联的逻辑主机，并在群集中的每台机器上都有相同的路径名。

DMS 和 SMS 表空间在 HA 环境中都受支持。DMS 表空间的设备容器必须使用由卷管理器创建的原始设备，它们或者是镜像的，或者采用 RAID 配置。不应使用规则磁盘设备，如 `/dev/rdsk/c20t0d0s0`，原因是：

- 它增加了同时从多台机器写该设备的可能性。

- 控制器编号在另一机器上可能不同。

如果 DB2 在此情况下进行故障转移，则它所需的磁盘设备看起来与它们在另一机器上时不同，并且 DB2 也不启动。DMS 表空间的文件容器以及 SMS 表空间的容器必须驻留在已安装的文件系统上。当一个逻辑主机的 vfstab 文件中包括了该逻辑主机的文件系统时，便会自动安装那些文件系统。

一个逻辑主机的 vfstab 文件在以下路径中:

```
/etc/opt/SUNWcluster/conf/hanfs/vfstab.<logical_host>
```

其中，*logical_host* 是与该 vfstab 文件关联的逻辑主机的名称。

每个逻辑主机都有它自己的 vfstab 文件，该文件包含将该逻辑主机的磁盘组传送至当前机器之后（但在启动 HA 服务之前）要安装的文件系统。Sun Cluster 软件在运行 **fsck**（文件系统检查）以确保文件系统健康之后，它将尝试安装任何正确定义的文件系统。若 **fsck** 失败，则不安装文件系统，并记录错误消息。

注: 若一个进程打开了文件，或其当前工作目录在安装点之下，则安装将失败。为了预防这种情况，确保逻辑主机的 vfstab 文件中包含的安装点下面没有留下任何进程。

当使用 SMS 表空间时，可以对 EEE 实例的文件系统布局使用任何约定。下面是 hadb2_setup 实用程序所使用的约定:

```
scadmin@crackle(190)# pwd
/export/ha_home/db2eee/db2eee
scadmin@crackle(191)# ls -l
total 18
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0000 -> /log0/disks/db2eee/NODE0000
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0001 -> /log0/disks/db2eee/NODE0001
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0002 -> /log0/disks/db2eee/NODE0002
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0003 -> /log0/disks/db2eee/NODE0003
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0004 -> /log0/disks/db2eee/NODE0004
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0005 -> /log1/disks/db2eee/NODE0005
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0006 -> /log1/disks/db2eee/NODE0006
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0007 -> /log1/disks/db2eee/NODE0007
lrwxrwxrwx 1 root build 28 Aug 12 19:08 NODE0008 -> /log1/disks/db2eee/NODE0008
scadmin@crackle(192)#
```

实例所有者是 db2eee，db2eee 实例的缺省数据库目录是 /export/ha_home/db2eee。逻辑主机 log0 存放数据库分区 0、1、2 和 3，而逻辑主机 log1 存放数据库分区 4、5、6、7 和 8。

对于每个数据库分区，都有一个对应的 NODE*xxxx* 目录。数据库分区的节点目录指向关联的逻辑主机文件系统下面的一个目录。

当选择路径约定时，确保:

1. 文件系统的磁盘在特定逻辑主机的磁盘组中，该逻辑主机对需要这些磁盘的数据库分区负责。
2. 通过逻辑主机的 vfstab 文件安装存放容器的文件系统。

EE 和 EEE 实例的主目录布局

对于 EE 实例，主目录应该是逻辑主机的 vfstab 文件中定义的文件系统。此目录在 DB2 启动之前便可用，并随 DB2 一起被传送至群集中逻辑主机所移到的位置。每台机器都有它自己的 vfstab 文件副本，您应仔细确保该文件在每台机器上的内容都相同。下面是一个 EE 实例的主目录的示例:

```
/log0/home/db2ee
```

其中，/log0 是逻辑主机 log0 的逻辑主机文件系统，而 db2ee 是 DB2 实例的名称。应将这个主目录路径放在群集中可以存放 "db2ee" 实例的每台机器的 /etc/passwd 文件中。

对于 EEE 实例，有两种设置主目录的方法。对于热备用配置，设置主目录的方法可以与用于 EE 实例的方法相同。对于相互接管配置，必须将 HA-NFS 用于主目录，且必须在设置 EEE 实例*之前*正确配置它。

群集中的其中一台机器必须使用所选逻辑主机的 dfstab 文件来为 EEE 实例导出文件系统。dfstab 文件包含当一台机器存放逻辑主机时应通过 NFS 导出的文件系统。每台机器都有它自己的 dfstab 文件副本，您应仔细确保该文件在每台机器上的内容都相同。

有关 HA-NFS 文件系统的信息（通过 hadb2_setup 程序）放在 hadb2tab文件中。当 HA 代理程序读取实例的信息时，它自动为该实例安装 HA-NFS 文件系统（参见 第245页的『hadb2tab 文件』）。

HA-NFS 文件系统的安装点通常是 /export/ha_home。在群集中的每台机器上，这将是从导出 HA-NFS 目录的逻辑主机安装的 NFS。EEE 实例所有者的主目录放在此目录下面，名为:

```
/export/ha_home/<instance>
```

其中，*instance* 是实例所有者的名称。

实例在每台机器上都可以有一个主目录，这样可以避免反复安装和卸装。但这样将需要进行额外的管理，才能确保主目录在每台机器上都是完全相同的。未能这样做会导致 DB2 不能正确启动，或导致它使用另一配置启动。这*不是*受支持的配置。

逻辑主机和 DB2 UDB EEE

通常选择逻辑主机来存放一个或多个数据库分区，以及导出 HA-NFS 文件系统。例如，若群集中有四个数据库分区和两台机器，则每台机器都应该有一个逻辑主机（图33）。其中一个逻辑主机可以存放两个数据库分区并导出 HA-NFS 文件系统，而另一个逻辑主机可以存放其余两个数据库分区。

在缺省情况下，DB2 UDB EEE 实例分配足够的资源，以成功地向已带有一个或多个用于该实例的活动数据库分区的机器添加最多两个数据库分区。例如，若群集中的单个实例有四个数据库分区，则仅当每个逻辑主机有一个数据库分区，或一个逻辑主机存放了三个数据库分区时，才会有所影响。在任何一种情况下，都有可能将三个数据库分区执行故障转移至已存放有一个用于同一实例的数据库分区的机器。

可使用 DB2_NUM_FAILOVER_NODES 注册表变量来增加为进行故障转移的数据库分区保留的资源量。

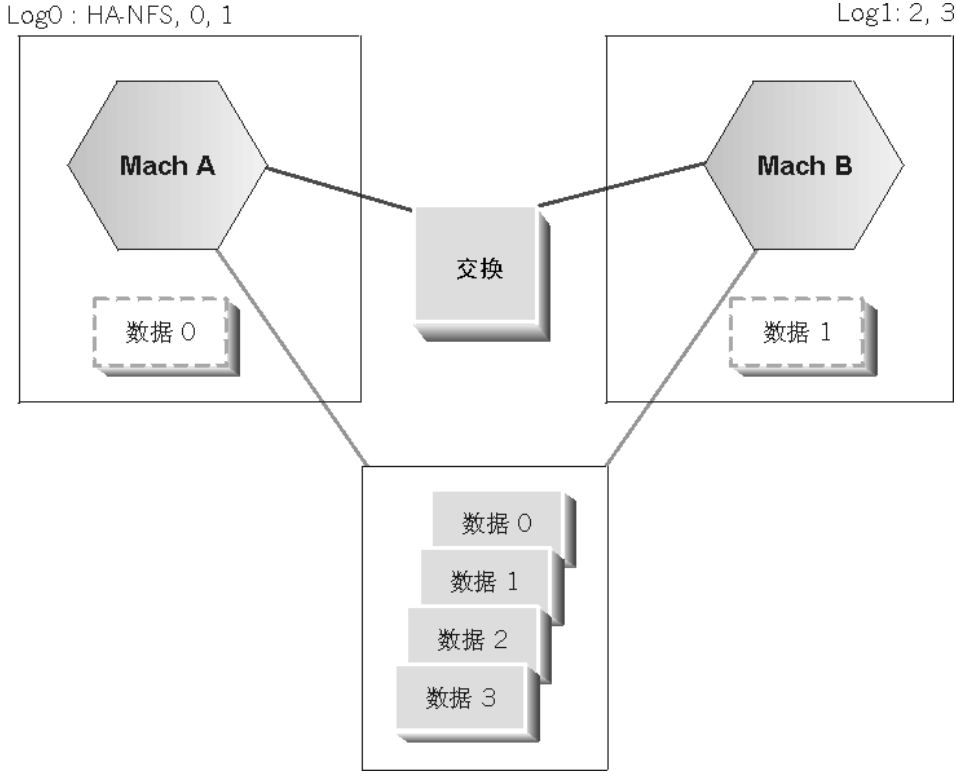


图 33. 每台机器一个逻辑主机

DB2 安装位置和选项

应对安装有 DB2 的文件系统执行镜像，或者，它至少应采用 RAID 配置。若 DB2 安装在规则磁盘上，则更有可能发生磁盘故障；所导致的故障转移被认为是可以预防的，这降低了群集的稳定性。

因为 HA 代理程序始终需要对 DB2 库具有访问权，所以不能将 DB2 安装在用于逻辑主机的磁盘组中的磁盘上。若 HA 代理程序对 DB2 库不具有访问权，则它们将失败。必须将 DB2 正常地安装在群集中的每台机器上。

数据库和数据库管理程序配置参数

可以使用 pre_db2start 脚本来在故障转移之后但在 DB2 启动之前更改数据库管理程序配置参数（参见第247页的『用户脚本』）。这个可执行脚本在实例所有者的主目录的 sqllib/ha 目录下运行（如果它存在的话）。正如其名称所暗示的那样，它刚好在 **db2start** 之前运行。除非该实例是 EEE 实例，否则将传送给控制方法的相同自变量传送给 pre_db2start 脚本。对于 EEE 实例，还将 **db2start** 命令的节点号传送给 pre_db2start 脚本。

崩溃恢复

HA 环境中的崩溃恢复与规则环境中的崩溃恢复相同。即使启动 HA 实例的机器与其发生崩溃时所在的机器不同，该实例的文件和磁盘设备看起来也是一样的，而恢复数据库所需的操作也不会有什么不同。关于崩溃恢复和其他形式的数据库恢复的更多信息，请参阅第3页的『第1章 制订一个良好的备份与恢复策略』。

虽然可以人工地（或通过其中一个用户脚本）重新启动数据库，但还是建议您将 *autorestart* 数据库配置参数设置为 ON，特别是对于 EEE 实例而言更是如此。这将把数据库处于不一致状态的时间长度缩至最短。

通过数据复制的高可用性

也可以通过复制增强数据可用性。通过在两台服务器之间复制数据，可以达到某种形式的高可用性。如果其中一台服务器关闭，则另一服务器应能够替换它并继续提供数据服务。

但是因为复制是异步执行的，因此当另一服务器关闭时，可能尚未将某些更改传播至该服务器。

Sun Cluster 2.2 上的 DB2 Connect 先决条件

如果满足下列条件，DB2 Connect 可在 Sun Cluster 2.2 上受支持:

- 主机使用的协议是 TCP/IP（不是 SNA）

- 不使用两阶段落实。如果用户将 SPM 日志配置到共享磁盘上（可通过 *spm_log_path* 数据库管理器配置参数来实现），而故障转移机器有相同的 TCP/IP 配置（相同的主机名、IP 地址等等），此限制将不生效。

DB2 高可用性代理程序

DB2 高可用性代理程序在 DB2 与 SC2.x 之间起中介作用。它为 Sun Cluster 2.2 软件提供了一种控制群集环境中的 DB2 的方法，而无需深入了解 DB2。EE 和 EEE 实例都有一个代理程序。代理程序既支持管理实例，又支持数据库实例。

注册 hadb2 服务

要使用 SC2.2，必须注册 DB2 HA 代理程序。注册数据服务告知 SC2.2 有哪些控制方法可用，以及它们驻留在哪一个目录中。HA 代理程序附带交付了一个名为 hadb2_reg 的特殊脚本，它可以为 EE 和 EEE 实例注册 hadb2 服务。只需要对整个群集运行一次 hadb2_reg 脚本。

虽然只有一组用于 DB2 HA 代理程序的控制方法，但它们的注册方式取决于是否将在相互接管配置中使用 EEE 实例。对于热备用配置中的 EE 实例或 EEE 实例，不使用 HA-NFS；因此，不需要 "-d nfs" 开关（它告知 SC2.2 软件 hadb2 服务依赖于 HA-NFS）。

hadb2_reg 用来为 EEE 实例注册 DB2 V7.1 控制方法的实际命令是:

```
hareg -r hadb2 -b /opt/IBMdb2/V7.1/ha -m
START=hadb2_start,START_NET=hadb2_startnet,STOP_NET=hadb2_stopnet,
FM_START=hadb2_fmstart,FM_STOP=hadb2_fmstop
-t START_NET=$TIMEOUT,STOP_NET=$TIMEOUT -d nfs
```

-b 开关告知 SC2.x 在 opt/IBMdb2/V7.1/ha 目录中查找所有控制方法。-m 开关定义 hadb2 服务的实际控制方法。-t 开关定义 START_NET 和 STOP_NET 控制方法的超时。有关每个控制方法的详细描述，参考 Sun Cluster 文档。

可使用 hadb2_unreg 脚本来注销 hadb2 服务，并且与 hadb2_reg 一样，它只需要对群集运行一次。

hadb2tab 文件

hadb2tab 文件是 DB2 HA 代理程序的主配置文件。每个控制方法都查阅此文件，以了解哪些实例是高度可用的。hadb2tab 文件位于 DB2 UDB 版本 7.1 的 /var/db2/v71/ 目录下面。此文件支持多个实例，每一个非注释行都表示一个不同的 HA 实例。下面是 hadb2tab 文件的一个示例:

```
<scadmin@thrash(203)# cat hadb2tab
EEE DATA db2eee jolt ON /export/ha_home /log0/home #Added by DB2 HA software
EE ADMIN db2ee log1 ON  -               -          #Added by DB2 HA software
```

第一个字段向 DB2 HA 代理程序指示该实例是 EE 实例还是 EEE 实例。第二个字段指示该实例是数据实例还是管理实例。第三个字段包含该 HA 实例的用户名。第四个字段是该实例的逻辑主机或 HA-NFS 主机，这取决于它是 EE 实例还是 EEE 实例。第五个字段指示该实例的故障监控是打开的还是关闭的。最后两个字段分别是本地安装点和远程 HA-NFS 目录。如果不使用这些字段，则应将它们设置为 -（连字符），并且，这些字段只应配合 EEE 相互接管配置使用。hadb2tab 文件中允许注释，如果一行中 ″#″ 标记前面的信息的长度为零，或该信息不是一个实例的有效定义，则该行被视为注释。

控制方法

SC2.2 代理程序的控制方法可以是一组脚本或程序。Solaris 上 DB2 的代理程序是一组包括下列方法的程序:

START_NET
: hadb2_startnet，用来启动 DB2

STOP_NET
: hadb2_stopnet，用来停止 DB2

FM_START
: hadb2_fmstart，用来启动 DB2 的故障监控程序

FM_STOP
: hadb2_fmstop，用来停止 DB2 的故障监控程序

有关这些控制方法的更多信息，参考 Sun Cluster 文档。

对于 EE 实例，与该实例关联的逻辑主机就是在 hadb2tab 文件中定义的。然而，对于 EEE 实例，还必须在以下文件中查找控制方法:

```
~<instance>/sqllib/ha/hadb2-eee.cfg
```

其中，~<instance> 是实例所有者的主目录。此文件对每个数据库分区都包含一行，它用来将数据库分区与逻辑主机关联。有效的 hadb2-eee.cfg 文件的一个示例是:

```
crackle % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

实例或数据库分区跟随着相对应的逻辑主机围绕在群集周围。逻辑主机可以移至群集中任何受下层硬件和 SC2.2 支持的机器。若配置设置正确，则 DB2 将支持任何受 SC2.2 软件支持的拓扑结构。

在读取了一个实例的所有信息之后，控制方法便知道有哪些逻辑主机与该实例关联。在对命令行自变量进行了语法分析之后，控制方法还知道当前机器存放了哪些逻辑主机，以及未存放哪些逻辑主机。

下表显示了各控制方法在运行时所执行的操作，以及当前机器上是否存放了与数据库分区或实例关联的逻辑主机。

控制方法	存放关联的逻辑主机	不存放关联的逻辑主机
START_NET	启动 DB2 实例或数据库分区	无操作
STOP_NET	无操作	停止 DB2 实例或数据库分区
FM_START	启动实例的故障监控程序	无操作
FM_STOP	无操作	停止实例的故障监控

执行启动操作的控制方法只关心当前存放的逻辑主机，执行停止操作的控制方法只关心当前未存放的逻辑主机。

若是在使用 HA-NFS，则控制方法还需要以一种特殊的方法安装 HA-NFS 目录。若未将 HA-NFS 的本地安装点和目录定义为 -（连字符），则控制方法在本地安装点上运行 statvfs(2)。若本地安装点的文件系统类型不是 nfs，则代理程序尝试使用 hadb2tab 行中的信息来安装文件系统。若已将 HA-NFS 的安装点和目录定义为 -（连字符），则需要对应的逻辑主机的 vfstab 文件来安装包含该实例的主目录的文件系统。对于 EE 和 EEE 热备用配置，只应将 HA-NFS 的本地安装点和远程目录定义为 -（连字符）。

用户脚本

这些脚本从控制方法中运行，以添加附加的功能，它们与控制方法传送相同的命令行自变量，此外，它们是由系统管理员或数据库管理员编写的。

若必须从不在后台运行的脚本中运行一个程序，则考虑使用 nohup(1) 使该程序在后台运行。nohup 程序保护执行的程序不受 SIGHUP（或挂断）信号干扰。若没有 nohup，则脚本中以后台方式运行的程序可能会被该脚本完成时发出的 SIGHUP 信号结束。

控制方法运行下列脚本:

- /var/db2/v61/failover
- ~<instance>/sqllib/ha/pre_db2start
- ~<instance>/sqllib/ha/post_db2start
- ~<instance>%s/sqllib/ha/post_failover
- ~<instance>/sqllib/ha/pre_db2stop

- ~<instance>/sqllib/ha/fm_warning

其中，*~instance* 是 HA 实例的主目录。

除 fm_warning 脚本之外，每个用户脚本都是使用调用它的控制方法的自变量运行的。使用 EEE 实例时，还将数据库分区编号（作为最后一个自变量）传送给用户脚本。

/var/db2/v71/failover 脚本是在 START_NET 方法开始时调用的，它在后台运行。举个例子，这样的脚本可用来在发生故障转移时向支持人员发送电子邮件。下面是故障转移脚本的一个示例:

```
#!/bin/ksh

# E-mail or page support staff to notify them that a failover has occurred.

echo "Failover occurred on machine 'hostname':Running $0!"
   |/bin/mail admin@sphere.torolab.ibm.com
```

要成功地从脚本中发送电子邮件，必须在系统上正确地配置 sendmail(1m)。

正如其名称所暗示的那样，pre_db2start 脚本刚好在调用 **db2start** 之前运行。此脚本可用于诸如更改数据库管理程序配置参数之类的任务。它必须在 20 秒之内完成。对于 EEE 实例，在每个数据库分区上调用 **db2start** 之前运行此脚本。此脚本仅适用于数据实例，而不适用于管理实例。

类似地，刚好在调用 **db2start** 之后运行 post_db2start 脚本。此脚本可用于诸如重新启动数据库之类的任务。它在后台运行，以确保其执行时间不会干扰其他实例。此脚本仅适用于数据实例，而不适用于管理实例。

在处理实例之后，运行实例所有者主目录下面的 post_failover 脚本。可使用此脚本来通知客户机应用程序: DB2 现在已开始工作、来激活数据库或将状态文件发送给管理员。它在后台运行，以防止其执行时间延迟其他 HA 实例的操作。下面是后故障转移脚本的一个示例:

```
#!/bin/ksh
#

# Send the status file to the administrato-r.
mail admin@sphere.torolab.ibm.com </tmp/HA.info.db2eee
```

DB2 HA 代理程序的 START_NET 和 STOP_NET 方法都在处理每个实例之后创建一个状态文件。状态文件的名称是:

```
/tmp/HA.info.<instance>
```

其中，*instance* 是实例所有者的用户名。此状态文件包含该实例的启动和停止报告，以及运行控制方法所花费的时间。下面是状态文件的一个示例:

```
scadmin@crackle(173)# cat /tmp/HA.info.db2eee
-----  Elapsed Time: 00:00:18           -----
-----  Elapsed Time: 00:00:00  (HA-NFS) -----

NODE     ACTION     RESULT     TRIES     RC
----     ------     ------     -----     --
   4     stop       success        3    1064
   5     stop       success        1    1064
   6     stop       success        2    1064
   7     stop       success        2    1064
   8     stop       success        1    1064
-------------------------------------------
```

pre_db2stop 脚本刚好在调用 **db2stop** 之前运行。可使用此脚本来通知客户机应用程序: DB2 将要停止。它必须在 20 秒之内完成。此脚本仅适用于数据实例，而不适用于管理实例。

当 DB2 因为意外关闭而重新启动时，故障监控程序也将运行一个用户脚本。此脚本名为:

```
~<instance>/sqllib/ha/fm_warning
```

可使用 fm_warning 脚本来通知系统管理员: 故障监控程序重新启动了 DB2。系统管理员应尝试查出 DB2 意外关闭的原因，并采取适当的措施来防止它再次发生。fm_warning 脚本在后台运行。

其他注意事项

若关闭了 HA 数据服务，则故障转移或群集重新配置期间只运行停止方法；仅当正确注册并打开了 HA 数据服务时，才运行其他方法。

确保群集中的每台机器都有足够的资源来运行它所负责的所有数据服务。在将群集投入生产环境之前，必须要考虑资源，如 CPU 负荷、内存、交换和核心参数。例如，若群集中的一台机器可能需要运行两个 DB2 实例，则该机器的核心参数需求将是这两个实例的需求之和。

故障监控程序

若打开故障监控，则群集重新配置或故障转移期间，将启动故障监控程序。若 START_NET 脚本未启动 DB2，则故障监控程序将自行启动 DB2。故障监控程序可以检测出 DB2 是尚未启动，还是因为未知的原因而关闭了。正因为如此，重要的是不要在打开故障监控程序时人工关闭 DB2。故障监控程序将把这视为意外关闭，并重新启动 DB2。如果发生的次数太多，则它将对适当的逻辑主机执行故障转移。

当对实例启用了故障监控时，正确的人工启动或停止实例的方法是首先关闭故障监控或 hadb2 服务。这两个操作都可以通过 **hadb2_setup** 命令使用 -f 和 -s 开关启动（参见 第254页的『hadb2_setup 命令』）。

注: 不要对同一逻辑主机使用多个实例。如果有多个实例与一个逻辑主机关联，则健康的实例可能会与不健康的实例一起进行故障转移。

EEE 注意事项

当决定将哪些数据库分区与一个逻辑主机关联时，重要的是考虑它们将如何进行故障转移。假设一个由两台机器组成的群集要配合这两台机器之间的四个数据库分区使用，如图34 所示。

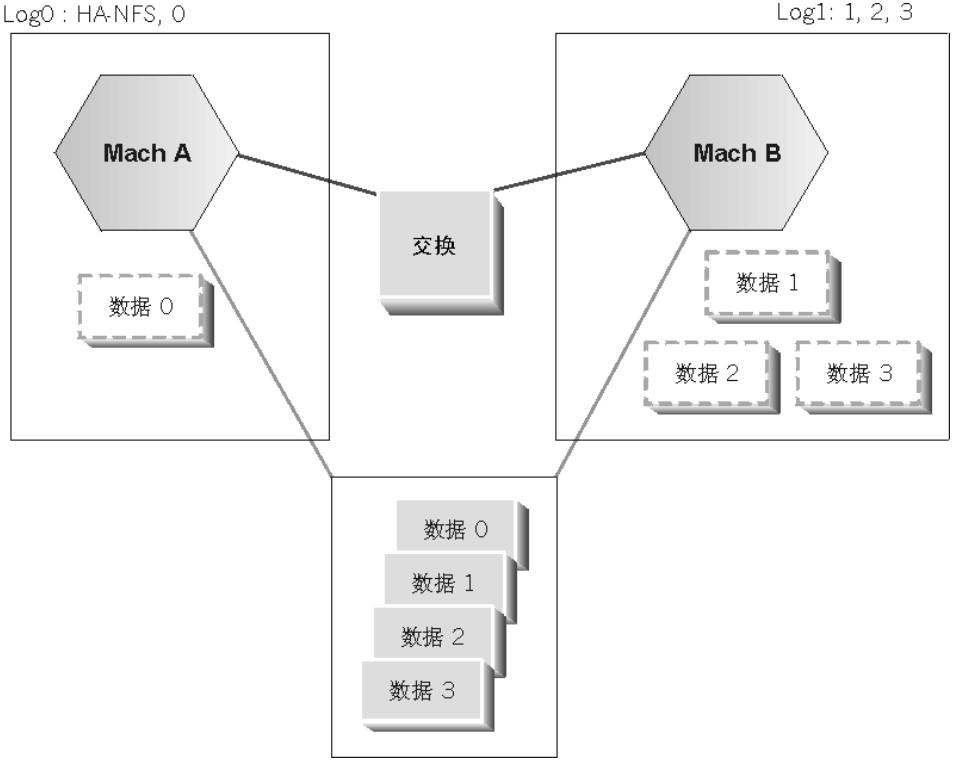


图 34. 由两台机器组成的群集与四个数据库分区

可将一个逻辑主机与每个数据库分区关联，并将一个逻辑主机用于 HA-NFS。在这种情况下，若所有逻辑主机都驻留在一个系统上，则会有问题。若该系统失效，则必须同时将所有逻辑主机移出该系统。不幸的是，Sun Cluster 软件并非按任何可预测的次序移动逻辑主机，在移动与 HA-NFS 关联的逻辑主机之前，可能会移动带有关联的数据库分区的逻辑主机。通常，最好是根据单一系统上将要存放的数

据库分区来将那些数据库分区分组到一起。这意味着正常情况下存放在一台机器上的两个数据库分区应与单一逻辑主机关联。

将更新 EEE 实例使用的 db2nodes.cfg 文件，以指示数据库分区所在的机器。例如，若所有数据库分区都在一台名为 "crackle" 的机器上，则 db2nodes.cfg 文件类似于:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 crackle 4 204.152.65.33
5 crackle 5 204.152.65.33
6 crackle 6 204.152.65.33
7 crackle 7 204.152.65.33
8 crackle 8 204.152.65.33
```

若这些数据库分区中的某一些被移至一台名为 "thrash" 的机器，则 db2nodes.cfg 文件被更新为:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 crackle 2 204.152.65.33
3 crackle 3 204.152.65.33
4 thrash 0 204.152.65.34
5 thrash 1 204.152.65.34
6 thrash 2 204.152.65.34
7 thrash 3 204.152.65.34
8 thrash 4 204.152.65.34
```

注意，主机名和交换机名都更改为反映机器名 "thrash"，端口编号也不同了。

HA.config 文件

若 /etc/HA.config 文件存在的话，它可以包含许多配置选项，包括下列各项:

```
scadmin@thrash(204)# cat /etc/HA.config
SYSLOG_FACILITY=LOG_LOCAL3
SYSLOG_LPRIORITY=LOG_INFO
SYSLOG_EPRIORITY=LOG_ERR
USE_INTERCONNECT=auto
SWITCH_NAME=204.152.65.18
DEBUG_LEVEL=2
FAILS_PER_HOUR=2
FAILS_PER_DAY=4
FAILS_PER_WEEK=10
FM_FAIL_SEV=soft
DB2START_TIMEOUT=60
DB2STOP_TIMEOUT=500
SCRIPT_USER=bin
```

注: 若 HA.config 文件不存在，则使用缺省值。

SYSLOG_FACILITY 变量设置用于记录消息和错误的 SYSLOG 设施。SYSLOG_LPRIORITY 和 SYSLOG_EPRIORITY 变量分别设置用于记录资料式信息和错误消息的 SYSLOG 优先级。

可能需要进行一些更改，以使 SYSLOG 精灵程序能够记录来自 DB2 HA 代理程序的信息。例如，将下列两行之一添加至 /etc/syslog.conf 文件将告知 SYSLOG 精灵程序把信息写至一个日志文件。

```
*.notice                                        /var/adm/SC.x
local3.info                                     /var/adm/SC.LOG_LOCAL3
```

Sun Cluster 通常使用高速互连。要对 DB2 使用高速互连，请将 USE_INTERCONNECT 设置为 auto 或 override。auto 设置（缺省值）使用 Sun 内部逻辑网络接口。若初始接口失效，则此接口将被传送至另一物理接口。若将 USE_INTERCONNECT 设置为 override，则交换机名从 SWITCH_NAME 变量中获取。另一个选项是将 USE_INTERCONNECT 设置为 no，它指定不使用高速互连。

DEBUG_LEVEL 指定故障转移期间要记录多少信息。它是一个介于 0 与 10 之间的数字，其中，10 为最高调试级别。使用指定的 SYSLOG 优先级和设施记录信息。若遇到任何问题，则将调试级别设置为最高级别，将 SYSLOG 配置为记录 HA 代理程序的输出，并将 SYSLOG 输出发送至 IBM 服务。

有三个变量可以帮助 DB2 故障监控程序确定何时对逻辑主机执行故障转移: FAILS_PER_HOUR、FAILS_PER_DAY 和 FAILS_PER_WEEK。每一个 HA 环境都是不同的；您必须确定可以接受的 DB2 故障数量。在每个“可接受的”故障之后，在同一机器上重新启动 DB2。当超过这三个故障阈值之一时，与实例或数据库分区关联的逻辑主机执行故障转移。

FM_FAIL_SEV 变量指定故障转移是“软的”还是“硬的”。要获取更多信息，参考 Sun Cluster 文档中有关 hactl(1m) 的部分。

DB2START_TIMEOUT 和 DB2STOP_TIMEOUT 变量指定允许 **db2start** 和 **db2stop** 运行的最大秒数。在指定的时间间隔过去之后，HA 代理程序认为该操作已失败，并尝试重新启动实例。

有些用户脚本不与任何特定实例关联。正常情况下，这些脚本作为 root 用户运行；这一点可以被 SCRIPT_USER 变量覆盖，可设置此变量以指定可以运行这些脚本的用户标识。

控制方法是如何运行 DB2 命令的

DB2 HA 代理程序使用 **su** 命令来作为实例所有者运行命令。实际的命令将类似于:

```
su - <instance> -c "db2stop"
```

其中，*instance* 是实例的用户名。

重要的是确保实例所有者的 .profile 文件支持 **su**。若不支持，则 **su** 命令不能正确工作。请人工调用 **su** 命令，或从脚本调用，以验证该命令是否能够成功运行。

设置

在阅读本节之前，您务必要熟悉 SC2.2 软件。本节假定您了解如何设置 SC2.2 和 HA-NFS，并且了解如何使用卷管理器。除 DB2 所必需的其他补丁程序之外，HA 代理程序还需要下列补丁程序:

```
Solaris 2.6:
   105210-17（或更新版本）
   105786-05（或更新版本）
```

注: Solaris 7 (Solaris 2.7) 没有必需的补丁程序。

公共安装步骤

1. 在群集中的所有机器上安装 SC2.2。安装期间，SC2.2 将询问要安装哪些代理程序。因为 DB2 不是由 SC2.2 附带交付的，所以它不在代理程序列表中。用于 DB2 的代理程序将随 DB2 一起安装，并通过 **hadb2_reg** 命令注册。
2. 使用磁盘组和逻辑 IP 地址配置逻辑主机。

DB2 UDB 企业版上的设置

1. 在逻辑主机的逻辑主机文件系统下面为实例创建主目录。
2. 在群集中的所有机器上安装 DB2。
3. 在群集中当前带有用于该实例的主目录的机器上创建该实例。
4. 向群集中的其他机器添加用于该实例的用户，确保数字用户标识相同。
5. 使用 **hadb2_reg** 命令注册 hadb2 服务。
6. 运行 **hadb2_setup** 命令来为该实例设置 HA。

DB2 UDB 扩充企业版上的设置

1. 为 HA 实例所有者创建主目录:
   a. 对于热备用，在逻辑主机的逻辑主机文件系统下面为实例创建主目录。

b. 对于相互接管配置，配置 HA-NFS，并从其中一个逻辑主机导出主目录。在其中一台机器上，在选择的安装点下面安装 HA-NFS 目录。

2. 在群集中的所有机器上安装 DB2。
3. 在安装有 HA-NFS 文件系统的机器上创建实例。
4. 向群集中的其他机器添加用于该实例的用户，确保数字用户标识相同。
5. 使用 **hadb2_reg** 命令注册 hadb2 服务。
6. 运行 **hadb2_setup** 命令来为该实例设置 HA。

注: 因为 NIS 会引入单个故障点，所以不建议使用 NIS 来定义 HA 实例的信息。

hadb2_setup 命令

hadb2_setup 命令是 DB2 HA 代理程序附带的程序的中心。可以使用它来设置、修改或删除实例。可以使用它来打开和关闭 hadb2_setup 服务。有了此命令，您无需人工编辑 hadb2tab 文件。

注: **hadb2_setup** 命令只对运行它的机器执行操作。对一台机器所作的更改也应应用于群集中的其他机器。

支持下列自变量:

添加 EE 实例:

```
---------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <logical_host> -p [DATA|ADMIN] -t EE

   例如:
      hadb2_setup -a -i db2ee -f off -h log1 -p DATA -t EE
```

添加 EEE 实例:

```
----------------------
   hadb2_setup -a -i <instance> -f [on|off] -h <nfs_host> -l <mount_point> \
      -r <ha-nfs_dir> -p [DATA|ADMIN] -t EEE -n "<node_info>"

   例如:
      hadb2_setup -a -i db2eee -f off -h ha-sun1 -l /export/ha_home \
         -r /log0/home -p DATA -t EEE -n "log0[0,10,20],log1[30,40,50]"
```

删除实例:

```
---------------------
   hadb2_setup -d -i <instance>
```

修改实例:

```
---------------------
```

```
hadb2_setup -m -i <instance> [-f [on|off] | -l <mount_point> | \
   -h <host> | -p [DATA|ADMIN] | -r <ha-nfs_dir> | -t [EE|EEE] ]
```

其他选项:

```
-------------
   -s   <on|off>              （对所有 HA 实例）启动或关闭 hadb2
   -y                         假设 yes，以进行安全性检查
```

要打开或关闭 hadb2 服务，指定 -s 开关。这与使用带有 -n 和 -y 开关，并指定 hadb2 服务的 **hareg** 命令等效。有关 **hareg(1m)** 命令的更多信息，参考 Sun Cluster 文档。

可使用 -f 开关来关闭实例的故障监控程序。其作用为在本地机器上停止该实例的故障监控程序，以及修改 hadb2tab 文件，以反映故障监控已关闭这一事实。

对于 EE 实例，建议在实例进行故障转移时在所有机器上关闭故障监控。对于 EEE 实例，在人工关闭该实例之前，必须在所有存放有该实例的数据库分区的机器上关闭故障监控。

要删除实例，请使用 -d 开关。这只从 hadb2tab 文件中除去该实例，而不除去或修改任何其他文件或目录。因为 hadb2tab 文件是 HA-DB2 代理程序的主配置文件，所以从此文件中除去一个实例将使控制方法不知道它的存在。

要修改实例，请使用 -m 开关。这只更改 hadb2tab 文件中的信息，而不除去或修改任何其他文件或目录。-m 开关可以与任何处理 hadb2tab 文件中的信息的开关一起使用。因为 **hadb2_setup** 命令不支持修改 db2nodes.cfg 文件和 hadb2-eee.cfg 文件，所以在初始设置之后，必须人工更改这些文件。

添加实例稍微有些棘手。

对于 EE 实例，需要下列自变量:

```
hadb2_setup -a -i <instance> -f <fm> -h <logical_host> -t <EEE_or_EE>
   -p <purpose>
```

其中，*instance* 是要添加的实例的名称，*fm* 指定最初是打开还是关闭故障监控，*logical_host* 是关联的逻辑主机，*EEE_or_EE* 设置为 EE，而 *purpose* 可以是 DATA 或 ADMIN。

对于 EEE 实例，需要下列自变量:

```
hadb2_setup -a -i <instance> -f <fm> -h <nfs_host> -t <EEE_or_EE> -p
   <purpose> -l <mount_point> -r <HA-NFS_directory> -n <node_info>
```

其中，*instance* 是要添加的实例的名称，*fm* 指定最初是打开还是关闭故障监控，*nfs_host* 是导出 HA-NFS 文件系统的逻辑主机的主机名，*EEE_or_EE* 设置为 EEE，*purpose* 可以是 DATA 或 ADMIN，*mount_point* 是 HA-NFS 目录的本地安装点，*HA-NFS_directory* 是 HA-NFS 目录，而 *node_info* 是将数据库分区与一个逻辑主机关联的信息。例如:

```
hadb2_setup -a -i db2eee -f on -h jolt -l /export/ha_home -p DATA -t EEE -r
   /log1/home -n "log0[0,1],log1[2,3]"
```

添加 EEE 实例时，必须将节点信息括在引号中。在此示例中，实例 ″db2eee″ 将与两个逻辑主机 ″log0″ 和 ″log1″ 关联。″db2eee″ 实例的数据库分区 ″0″ 和 ″1″ 将与逻辑主机 ″log0″ 关联，数据库分区 ″2″ 和 ″3″ 将与逻辑主机 ″log1″ 关联。

使用 **hadb2_setup** 命令来向群集中的所有机器添加实例。然后，可以通过强制群集重新配置，或通过先关闭再打开 hadb2 服务来启动实例。这可以通过 **hareg** 命令完成，也可以通过 **hadb2_setup** 命令的 -s 开关完成。若该实例未启动，则参见第259页的『故障诊断』。

当 **hadb2_setup** 命令添加 EEE 实例时，以透明方式执行下列操作:

- 检查指定的信息。这包括确保系统上存在该用户，且 HA-NFS 正在运行。
- 创建一个 db2nodes.cfg 文件。
- 创建一个 hadb2-eee.cfg 文件。
- 为 EEE 实例创建 .rhosts 文件。
- 创建从缺省数据库路径到关联的逻辑主机数据目录的符号链接。
- 向 hadb2tab 文件添加一行。

为了预防配置错误，并确保在运行 **hadb2_setup** 命令之后能够启动 HA 实例，该命令在添加新实例之前要执行相当大量的测试。

创建 db2nodes.cfg 文件，并将关于当前群集状态的信息存入其中。例如，若逻辑主机 ″log0″ 正驻留在机器 ″crackle″ 上，则与 ″log0″ 关联的数据库分区的条目将包含机器名 ″crackle″ 和用于 ″crackle″ 的高速互连:

```
scadmin@crackle(193)# cat db2nodes.cfg
0 crackle 0 204.152.65.33
1 crackle 1 204.152.65.33
2 thrash 0 204.152.65.34
3 thrash 1 204.152.65.34
```

仅根据命令上指定的节点信息创建 hadb2-eee.cfg 文件。对于每个数据库分区都有一行:

```
sphere % cat hadb2-eee.cfg
NODE:log0 0
NODE:log0 1
NODE:log1 2
NODE:log1 3
```

DB2 UDB EEE 需要 .rhost 文件，该文件应包含群集中每台机器的所有主机名（或 IP 地址）。例如:

```
crackle db2eee
204.152.65.1 db2eee
204.152.65.17 db2eee
thrash db2eee
204.152.65.2 db2eee
204.152.65.18 db2eee
crackle db2eee
jolt db2eee
bump db2eee
thrash.torolab.ibm.com db2eee
crackle.torolab.ibm.com db2eee
```

根据 SMS 表空间的文件系统布局，**hadb2_setup** 命令设置许多目录和符号链接。它们包括:

- 每个逻辑主机的逻辑主机文件系统下面的一个名为 ″data″ 的目录。
- 与该逻辑主机关联的每个数据库分区的节点目录（在这个 ″data″ 目录下面）。
- 缺省数据库路径中的符号链接，它们位于 ~<instance> 下面，其中，*~instance* 是实例的主目录。对于每个数据库分区，都有一个符号链接指向相对应的节点目录。要获取更多信息，参见 第240页的『EE 和 EEE 实例的磁盘布局』。

故障转移时间

故障转移时间从数据最初不可用时算起，直到它再次可用为止。故障转移期间发生的许多事件会对故障转移时间产生显著影响:

- 磁盘导出和导入。

  虽然导出和导入磁盘对整体停机时间确实有所影响，但与其他事件相比，它们通常不会花费很长时间。故障转移期间需要从一台机器移至另一机器的磁盘越多，此过程所花费的时间也越长。若有磁盘损坏，则此过程还要长一些。

- 对为逻辑主机安装的文件系统执行 fsck。

  在可以安装逻辑主机的文件系统之前，这些文件系统必须通过 fsck，以确保它们健康。文件系统越大，此过程就越长。通过使用日志文件系统，可以大幅缩短此时间。因为日志文件系统通常是在 HA 环境中使用的，所以 fsck 时间通常不是一个问题。

- 从 HA 代理程序调用的用户脚本。

HA 代理程序将调用用户脚本（如果它们存在且可执行的话）。这些脚本中的一些是同步运行的，它们可能会增加启动 HA 实例所需的时间。确保它们能够尽可能快地运行；请考虑在后台运行由这些脚本调用的任何外部程序。

- HA-NFS。

  对于相互接管配置中的单一 EEE 实例，必须将 HA-NFS 用于实例所有者的主目录。因为 *lockd* 有宽限期（在 HA-NFS 的 HA 代理程序中定义，当运行 HA-NFS 时，为 90 秒），所以 HA-NFS 会增加故障转移时间。在故障转移之后，因为任何在 HA-NFS 文件系统上锁定文件的进程都必须等待宽限期结束，所以这会影响故障转移时间。在故障转移之后，DB2 的 HA 代理程序是第一个在实例所有者的主目录下锁定文件的进程，它记录获取第一个锁定所花费的时间。在故障转移之后，此时间显示在状态报告中。

- 启动 DB2。

  启动 DB2 对故障转移时间的影响不大。对于 EE 实例，平均大约增加 5-15 秒。对于 EEE 实例，大约增加 10 秒，每个正在执行故障转移的数据库分区另外还要增加 5 秒。举个例子，如果有三个数据库分区正在进行故障转移，则启动这三个数据库分区将会使故障转移时间增加大约 25 秒。这不包括该实例的数据库的崩溃恢复。

- 数据库崩溃恢复。

  崩溃恢复通常导致与故障转移关联的停机时间中的绝大部分。复原数据库所需时间的长短取决于许多因素，包括:

  – 客户机工作负荷。事务日志中只记录对数据库的更改。若客户机工作负荷主要是只读操作，则崩溃恢复期间，必须对数据库应用的事务就相对少一些。

  – 磁盘和机器速度。磁盘以及存放 HA 实例的机器的速度对复原数据库所需时间的长短也有影响。系统越快，崩溃恢复时间也就越短。

  – *softmax* 数据库配置参数的值。*softmax* 的值是日志文件大小的百分比，达到此百分比时，设置一个软检查点，并写日志控制文件。崩溃恢复期间，使用日志控制文件来确定哪些日志记录是将数据库复原为一致状态真正必需的。减小此值将导致数据库管理程序更频繁地触发页清除程序，并设置更多的软检查点；虽然性能会降低，但数据库恢复却更快。

  – 该实例是 EE 还是 EEE。若该实例是 EEE 实例，则数据库重新启动操作将并行完成。每个数据库分区都负责重新启动它自己的数据库部分。若数据库有 50 GB 的数据，则带有四个数据库分区的实例将能够以大约四倍于 EE 实例的速度复原数据库。

故障诊断

下表标识了您可能会遇到的问题、其可能的原因以及为了解决它们而需执行的操作。

表 16. 在 Sun Cluster 2.2 上排除高可用性的故障

症状	可能的原因	操作
不能安装逻辑主机文件系统	逻辑主机的故障转移期间，已正常安装和卸装逻辑主机文件系统。故障转移期间，逻辑主机文件系统下面不应有活动的进程或打开的文件。在非常罕见的情况下，不能被删掉的进程在逻辑主机文件系统下面有它们的当前工作目录。要了解安装点下面是否有进程，使用 `fuser(1m)`，或使用名为 `lsof` 的 GNU 实用程序。当不能安装逻辑主机文件系统时，会生成错误消息。[a]	重新引导系统，或将该逻辑主机文件系统转移为另一名称，并重新创建它。这样做允许冻结的进程停留在该目录下面（因为不能删掉它），并使安装能够发生。[b]
db2start 或 db2stop 超时不起作用	SIGALRM 信号不能中断阻塞的系统调用。相反，该系统调用将重新启动，就象使用 `sigaction()` 设置了 SA_RESTART 标志一样。这导致 DB2 HA 代理程序的超时被忽略，代理程序方法将挂起，而不是从挂起的 **db2start** 或 **db2stop** 命令恢复。	对 Solaris 2.6 应用必需的补丁程序，即 105210-17（或更新版本）。
对实例的登录挂起	虽然发生此问题的原因相当多，但最常见的原因包括 NFS 问题和 `/usr/sbin/quota` 程序。	检查 NFS 安装，确保它们是健康的，并寻找由实例所有者拥有的限额进程。根据系统管理员的判断，将限额程序更改为指向 `/bin/true` 的符号链接可以解决此问题。这不是建议的解决方法，但可能管用。
我刚刚设置了 EEE 实例，但它不启动	**`hadb2_setup`** 命令未向 `/etc/services` 文件添加端口；期望管理员人工添加它们。返回了错误消息。[c]	确保 `/etc/services` 文件中命名了适当的端口。

表 16. 在 Sun Cluster 2.2 上排除高可用性的故障 (续)

症状	可能的原因	操作
START_NET 方法不能启动 DB2		关闭故障监控，以确保实例不执行故障转移。作为实例所有者登录，并尝试人工启动 DB2。 1. 确保 `hadb2tab` 配置文件指定了正确的实例类型。例如，将 `db2nodes.cfg` 文件用于 EE 管理实例将导致问题，HA 代理程序方法将无法从此问题复原。 2. 确保 `.rhosts` 文件存在，且其中带有有效的条目。 3. 确保群集中的所有机器使用 root 用户许可权来共享 HA-NFS 文件系统。 4. 检查核心参数，并确保它们正确。 5. 确保 `/etc/services` 文件包含用于该实例的条目。
实例只在一台机器上工作	• 实例的数字 *uid* 可能没有在群集中的每台机器上都相同。 • 核心参数可能没有在群集中的每台机器上都有效。 • `hadb2tab` 文件可能没有在群集中的每台机器上都相同。 • 其他配置文件（如逻辑主机 `vfstab` 文件）可能没有在群集中的每台机器上都相同。	若这些原因似乎都不适用，尝试作为实例所有者登录，并人工启动 DB2。对于 EE 实例，若存放实例的逻辑主机驻留在当前机器上，则此方法应管用。对于 EEE 实例，此方法可以对群集中任何可以存放数据库分区的机器起作用。
su - <instance> -c "db2start" 不工作	• 该实例的 `.profile` 可能不支持 **su**。 • Bourne shell (`/bin/sh`) 有一个已知的问题，即 **su** 命令能以人工方式工作，但不能通过 HA 代理程序工作。	• 尝试作为 root 用户人工运行此命令，在通过 HA 代理程序再试之前，确保它能工作。 • 如有必要，切换至 Korn shell (`/bin/ksh`)。
我的 EEE 实例不能启动，但主目录已安装	可能未使用“root 用户”许可权来将 HA-NFS 目录导出至群集中的机器。DB2 和 HA 代理程序都需要此目录才能正确运行。	要进行测试，尝试（作为 root 用户）在实例所有者的主目录下面创建一个文件。

表 16. 在 Sun Cluster 2.2 上排除高可用性的故障 (续)

症状	可能的原因	操作
尝试访问 EEE 实例目录时返回“NFS 文件句柄已过时”错误	实例所有者的主目录下面可能仍有进程。	卸装实例所有者的主目录，并允许 HA 代理程序重新安装它。若先关闭并再次打开 hadb2 服务（参见 第254页的『hadb2_setup 命令』中有关 **hadb2_setup** 命令上的 -s 开关的描述），则 HA 代理程序将重新安装该目录。
未能通过 SC2.2 成功运行控制方法	可能未向 Sun Cluster 软件注册 hadb2 服务，或者可能是未打开它。	若该控制方法似乎能从命令行正常运行，则检查 SYSLOG 文件中的错误消息可能会有助于您解释该问题。确保向 Sun Cluster 软件注册了 hadb2 服务，且它已打开。 人工运行方法对于调试问题而言很有用。[d] 应作为 root 用户来运行方法，并给出适当的命令行自变量。若逻辑主机列表为空，则给出自变量 ""。不带空格分隔符的双引号指示空白自变量。例如: `hadb2_startnet log0,log1 "" 600` 第一个自变量 log0,log1 告知 hadb2_startnet 方法: 逻辑主机 log0 和 log1 驻留在当前机器上。第二个自变量为空，它告知 hadb2_startnet 方法: 没有将其他逻辑主机存放在群集中的其他机器上（它们全都在当前机器上）。第三个自变量告知该方法: SC2.2 将在 600 秒之后超时。
用户脚本不运行	仅当用户脚本存在于适当的目录中且可执行时，它们才能运行。	检查文件所有权和属性。若脚本仍未能运行，则与 IBM 服务联系。寄上未运行的脚本的目录列表，以及应该已运行了该脚本的故障转移或群集重新配置的 SYSLOG 输出。

表 16. 在 Sun Cluster 2.2 上排除高可用性的故障 (续)

症状	可能的原因	操作
未将信息记录至 /etc/syslog.conf 中指定的文件		使用 touch(1) 来创建 /etc/syslog.conf 文件中指定的文件，然后重新启动 SYSLOG 精灵程序。

[a]在不能安装逻辑主机文件系统时，生成的错误消息可能类似于:

```
Aug 17 11:14:01 rash ID[SUNWcluster.loghost.1170]: importing data1
Aug 17 11:14:06 rash ID[SUNWcluster.scnfs.3040]: mount -F ufs -o ""
     /dev/vx/dsk/data1/data1-stat /log1 failed.
Aug 17 11:14:07 rash ID[SUNWcluster.ccd.ccdd.5304]: error freeze cmd =
     /opt/SUNWcluster/bin/loghost_sync
CCDSYNC_POST_ADDU LOGHOST_CM:log1:rash /etc/opt/SUNWcluster/conf/ccd.database
     2 "0 1" 1 error code = 1
```

[b] 例如:

```
scadmin@rash(218)# ps -fe | egrep db2
db2ee 1984 1 0 0:01 <defunct>
```

解决方法:

```
scadmin@rash(229)# cd /
scadmin@rash(230)# mv /log1 /log1.bkp
scadmin@rash(231)# mkdir /log1
```

[c] 错误消息可能类似于:

```
SQL6030N START or STOP DATABASE MANAGER failed. Reason code "13".
```

[d] 例如，若 hadb2_startnet 方法找不到 libdb2.so.1，但它能通过 Sun Cluster 软件正常运行，则不会报告错误。人工运行该方法将生成下列信息:

```
scadmin@crackle(213)# hadb2_startnet '''log0,log1' 600
ld.so.1: hadb2_startnet: fatal: libdb2.so.1: open failed:
   No such file or directory
Killed
```

第3部分 附录

附录A. 如何阅读语法图

语法图显示了应如何指定一个命令，以使操作系统能正确地解释输入了什么内容。

从左至右，从上至下，沿着水平线（主路径）读语法图。如果该行以箭头结束，命令语法将继续，且下一行也以箭头开始。用一个垂直的条标记命令语法结束。

从语法图输入信息时，应确保包括了标点符号，如引号和等号。

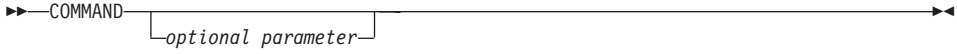
参数分类为关键字或变量:

- 关键字代表常量，以大写字母显示；但在命令提示符处，可用大写、小写或混合大小写输入关键字。例如，命令名就是一个关键字。
- 变量代表用户提供的名称或值，由小写字母显示；但在命令提示符处，变量可用大写、小写或混合大小写输入，除非显式声明了大小写限制。例如，文件名就是一个变量。

参数可以是关键字和变量的组合。

所需的参数显示在主路径上:

可选参数显示在主路径下:



参数的缺省值显示在路径上方:

第一个参数显示在主路径上的一个参数堆栈，表示必须选择一个参数:

第一个参数显示在主路径下的一个参数堆栈，表示可以选择一个参数:

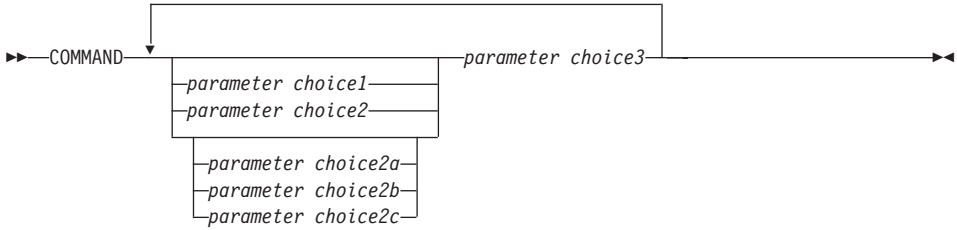
路径上返回到左边的一个箭头表示如果符合以下约定，就可重复该项:

- 如果箭头是不中断的，则可在各项由空格隔开的列表中重复该项:

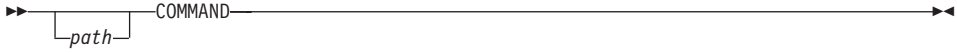
- 如果箭头中有逗号，则可在各项由逗号隔开的列表中重复该项:

如果符合前面讨论过的对需的参数和可选参数的堆栈约定，就可重复参数堆栈中的项。

某些语法图包含了其他参数堆栈中的参数堆栈。只有符合前面讨论过的约定才能重复堆栈中的项。即，如果内部堆栈上没有重复箭头，但外部堆栈上有，则只能从内部堆栈选择一个参数，并将它与外部堆栈中的任何参数组合在一起，可重复该组合值。例如，以下语法图显示了可以将参数 *choice2a* 与参数 *choice2* 组合，然后再次重复该组合值（*choice2* 加 *choice2a*）:



某些命令前有可选路径参数:



如果未提供此参数，系统将在当前目录中搜索该命令。如果找不到该命令，系统会继续在 .profile 中列示的路径上的所有目录中搜索该命令。

某些命令有等效的语法变体:

附录B. 警告、错误和完成消息

由各种备份和恢复实用程序生成的消息包括在 SQL 消息中。这些消息是由数据库管理器在已检测到警告或出错状态时生成。每条消息都有一个消息标识符，由前缀 (SQL) 和一个 4 位或 5 位的消息号组成。有 3 种消息类型: 通知、警告和关键。已 N 结束的消息标识符是错误消息。那些以 W 结束的消息标识符表示警告或信息性消息。以 C 结束的消息标识符表示关键系统错误。

消息号也称为 *SQLCODE*。取决于它的消息类型（N、W 或 C），发送给应用程序的 SQLCODE 是正数或负数。N 和 C 产生负值，而 W 产生正值。DB2 将 SQLCODE 返回给应用程序，而应用程序可获得与该 SQLCODE 相关的消息。DB2 也会对可能是 SQL 语句结果的情况返回 *SQLSTATE* 值。某些 SQLCODE 值有相关的 SQLSTATE 值。

关于所有 DB2 消息的详细信息，请参阅《*消息参考*》。可使用此书中的信息来标识错误或问题，并通过适当的恢复操作来解决问题。此信息还可用于了解消息从何处产生并记录在何处。

也可从操作系统命令行访问 SQL 消息以及与 SQLSTATE 值相关的消息文本。要访问这些错误消息的帮助，可在操作系统命令提示符处输入:

```
db2 ? SQLnnnnn
```

其中 *nnnnn* 代表消息号。在基于 UNIX 的系统上，建议使用双引号分界，这将避免目录中有单字符文件名时的问题:

```
db2 "? SQLnnnnn"
```

因为 **db2** 命令是不区分大小写的，且不需要终止字母，所以接受消息标识符作为参数。因此，以下命令将产生相同的结果:

```
db2 ? SQL0000N
db2 ? sql0000
db2 ? SQL0000n
```

如果消息文本太长，在屏幕上放不下，可使用以下命令（在基于 UNIX 的操作系统上以及其他支持 ″more″ 管道的操作系统上）:

```
db2 ? SQLnnnnn | more
```

也可将输出重定向到一个可在稍后进行浏览的文件中。

也可从交互式输入方式调用帮助。要访问此方式，可在操作系统命令提示符处输入以下命令:

```
db2
```

要以这种方式获得 DB2 消息，可在命令提示符处输入以下命令 (db2 =>):

```
? SQLnnnnn
```

发出以下命令，可检索与 SQLSTATE 相关的消息文本:

```
db2 ? nnnnn
或
db2 ? nn
```

其中 *nnnnn* 是一个 5 字符的 SQLSTATE 值（字母数字值），而 *nn* 是一个 2 位的 SQLSTATE 类代码（SQLSTATE 值的前两位）。

附录C. 附加 DB2 命令

db2adutl - 处理 TSM 归档的映象

允许用户查询、抽取、验证及删除备份映象、日志和使用 Tivoli Storage Manager（以前称为 ADSM）保存的装入复制映像。

在基于 UNIX 的系统上，此实用程序位于 INSTHOME/sqllib/misc 目录中。在 Windows 操作系统和 OS/2 上，它位于 \sqllib\misc 目录中。

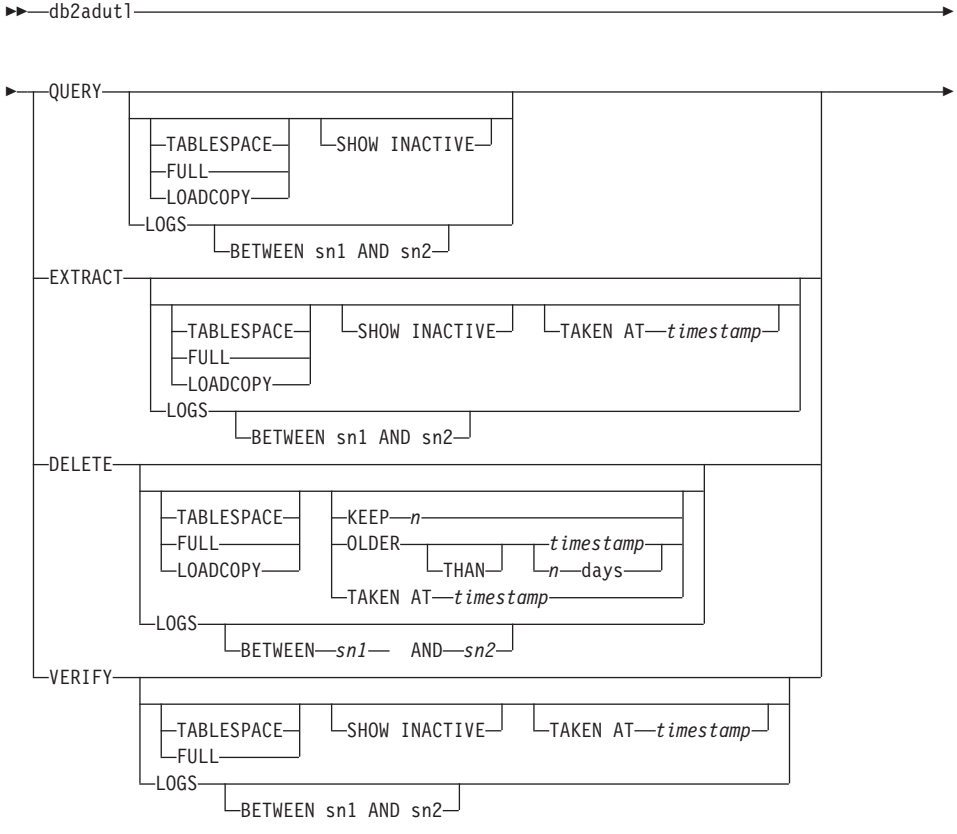
权限

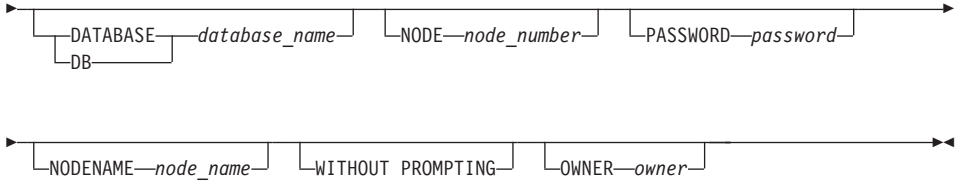
无

需要的连接

无

命令语法





命令参数

QUERY
向 TSM 服务器查询 DB2 对象。

EXTRACT
从 TSM 服务器复制 DB2 对象到本地机器上的当前目录中。

DELETE
在 TSM 服务器上，停用备份对象或删除日志归档。

VERIFY
对服务器上的副本执行一致性检查。

注: 此参数使整个备份映象基于网络传送。

TABLESPACE
只包括表空间备份映象。

FULL 只包括完整的数据库备份映象。

LOADCOPY
只包括装入副本映象。

LOGS 只包括日志归档映象

BETWEEN *sn1* **AND** *sn2*
指定将使用日志序列号 1 和日志序列号 2 之间的日志。

SHOW INACTIVE
包括已被停用的备份对象。

TAKEN AT *timestamp*
按时间戳记指定备份映象。

KEEP *n*
按时间戳记，取消激活指定类型的全部对象，除最新的 *n* 之外。

OLDER THAN *timestamp* **或** *n* **days**
指定将取消激活时间戳记早于 *timestamp* 或 *n* 天的对象。

DATABASE *database_name*
只考虑那些与指定的数据库名相关的对象。

NODE *node_number*
只考虑那些由指定的节点号创建的对象。

PASSWORD *password*
指定此节点的 TSM 客户机密码，如果需要的话。如果指定了一个数据库，但未提供密码，对 *tsm_password* 数据库配置参数指定的值将发送给 TSM；否则将不使用密码。

NODENAME *node_name*
只考虑那些与特定 TSM 节点名相关的映象。

WITHOUT PROMPTING
将不提示用户在删除对象前进行验证。

OWNER *owner*
只考虑那些由指定的所有者创建的对象。

示例

以下是来自 db2 backup database rawsampl use tsm 的样本输出:

```
Backup successful. The timestamp for this backup is : 19970929130942
  db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 1, Time: 19970929130942,
                                  Oldest log: S0000053.LOG, Sessions used: 1
   full database backup image: 2, Time: 19970929142241,
                              Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   table space backup image: 1, Time: 19970929094003,
                              Oldest log: S0000051.LOG, Sessions used: 1
   table space backup image: 2, Time: 19970929093043,
                              Oldest log: S0000050.LOG, Sessions used: 1
   table space backup image: 3, Time: 19970929105905,
                              Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

以下是来自 db2adutl delete full taken at 19950929130942 db rawsampl 的样本输出:

```
Query for database RAWSAMPL

Retrieving full database backup information.  Please wait.

   full database backup image: RAWSAMPL.0.db26000.0.19970929130942.001

    Do you want to deactivate this backup image (Y/N)? y

      Are you sure (Y/N)? y

  db2adutl query

Query for database RAWSAMPL

Retrieving full database backup information.
   full database backup image: 2, Time: 19950929142241,
                               Oldest log: S0000054.LOG, Sessions used: 1

Retrieving table space backup information.
   tablespace backup image: 1, Time: 19950929094003,
                               Oldest log: S0000051.LOG, Sessions used: 1
   tablespace backup image: 2, Time: 19950929093043,
                               Oldest log: S0000050.LOG, Sessions used: 1
   tablespace backup image: 3, Time: 19950929105905,
                               Oldest log: S0000052.LOG, Sessions used: 1

Retrieving log archive information.
   Log file: S0000050.LOG
   Log file: S0000051.LOG
   Log file: S0000052.LOG
   Log file: S0000053.LOG
   Log file: S0000054.LOG
   Log file: S0000055.LOG
```

db2ckbkp - 检查备份

此实用程序可用于测试备份映象的完整性，并确定是否可复原该映象。还可用它来显示存储在备份头中的元数据。

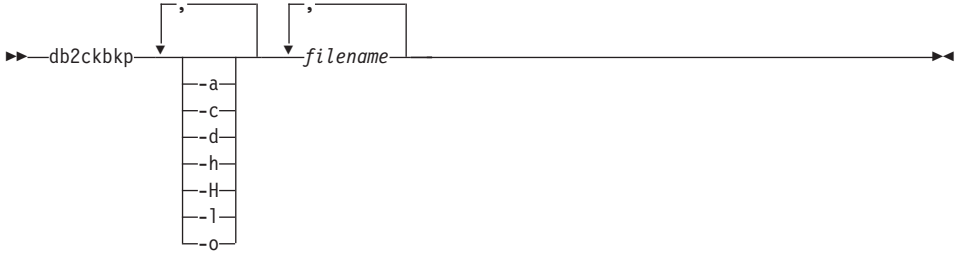
权限

任何人都可以访问该实用程序，但用户必须对映象备份有读许可权才能对它们执行此实用程序。

需要的连接

无

命令语法



命令参数

-a 显示所有可用的信息。

-c 显示检查位和检查和的结果。

-d 显示来自 DMS 表空间数据页的头部分的信息。

-h 显示媒体头信息，包括复原实用程序预期的映象的名称或路径。

-H 只显示媒体头信息。

注:

1. 此选项不验证映象。如果未指定该选项，会对整个映象执行验证。
2. 此选项在于其他任何选项组合时无效。

-l 显示日志文件头数据。

-o 显示对象头部分中的详细信息。

filename

备份映象文件的名称。可同时检查一个或多个文件。

注:

1. 如果完整的备份由多个对象构成，只有在将 **db2ckbkp** 用于同时验证所有对象时，该验证才会成功。
2. 检查一个映象的多个部分时，必须首先指定第一个备份映象 (.001)。

示例

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.*
[1] Buffers processed:  ##
[2] Buffers processed:  ##
[3] Buffers processed:  ##
Image Verification Complete - successful.

db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001


====================
MEDIA HEADER REACHED:
====================
        Server Database Name           -- SAMPLE2
        Server Database Alias          -- SAMPLE2
        Client Database Alias          -- SAMPLE2
        Timestamp                      -- 19990818122909
        Node                           -- 0
        Instance                       -- krodger
        Sequence Number                -- 1
        Release ID                     -- 900
        Database Seed                  -- 65E0B395
        DB Comment's Codepage (Volume) -- 0
        DB Comment (Volume)            --
        DB Comment's Codepage (System) -- 0
        DB Comment (System)            --
        Authentication Value           -- 255
        Backup Mode                    -- 0
        Backup Type                    -- 0
        Backup Gran.                   -- 0
        Status Flags                   -- 11
        System Cats inc                -- 1
        Catalog Node Number            -- 0
        DB Codeset                     -- ISO8859-1
        DB Territory                   --
        Backup Buffer Size             -- 4194304
        Number of Sessions             -- 1
        Platform                       -- 0

The proper image file name would be:
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

[1] Buffers processed:  ####
Image Verification Complete - successful.
```

用法注释

如果备份映象使用多个会话创建的，**db2ckbkp** 可同时检查所有文件。用户负责确保序列号为 001 的会话是指定的第一个文件。

此实用程序也可以验证存储在磁带上的备份映象（那些用变量块大小创建的映象除外）。通过为复原操作准备磁带、然后调用实用程序再指定磁带机名称来实现。例如，在基于 UNIX 的系统上:

```
db2ckbkp -h /dev/rmt0
```

在 Windows NT 上:

```
db2ckbkp -d \\.\tape1
```

如果备份映象驻留在 TSM 上，则参见 第272页的『db2adutl - 处理 TSM 归档的映象』。

db2ckrst - 检查增量复原映象序列

查询数据库历史记录，并生成进行增量复原操作所需的备份映象的时间戳记列表。同时还生成用于手工增量复原的简化了的复原语法。

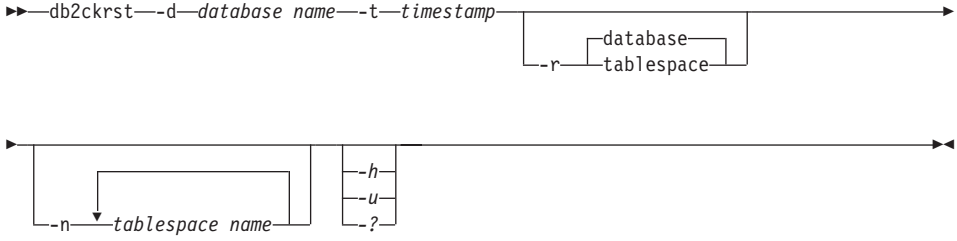
权限

无

需要的连接

无

命令语法



命令参数

-d database namefile-name
指定将复原的数据库的别名。

-t timestamp
指定将增量复原的备份映象的时间戳记。

-r 指定将执行的复原类型。缺省值是数据库复原。

注: 如果选择了表空间但未给出表空间名，则实用程序会查找指定映象的历史记录条目并使用列出的要执行复原的表空间名。

-n tablespace name
指定将复原的一个或多个表空间的名称。

注: 如果选择了数据库复原类型并指定了表空间名的列表，则实用程序将使用指定的表空间名继续执行表空间复原操作。

-h/-u/-?
显示帮助信息。当指定了此选项时，其他所有的选项都会被忽略，且只显示帮助信息。

示例

```
db2ckrst -d mr -t 20001015193455 -r database
db2ckrst -d mr -t 20001015193455 -r tablespace
db2ckrst -d mr -t 20001015193455 -r tablespace -n tbsp1 tbsp2

> db2 backup db mr

Backup successful. The timestamp for this backup image is : 20001016001426

> db2 backup db mr incremental

Backup successful. The timestamp for this backup image is : 20001016001445

> db2ckrst -d mr -t 20001016001445

Suggested restore order of images using timestamp 20001016001445 for database mr.
==================================================================
  db2 restore db mr incremental taken at 20001016001445
  db2 restore db mr incremental taken at 20001016001426
  db2 restore db mr incremental taken at 20001016001445
==================================================================

> db2ckrst -d mr -t 20001016001445 -r tablespace -n userspace1
Suggested restore order of images using timestamp 20001016001445 for database mr.
==================================================================
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001426
  db2 restore db mr tablespace ( USERSPACE1 ) incremental taken at 20001016001445
==================================================================
```

用法注释

数据库历史记录必须存在，此实用程序才能使用。如果数据库历史记录不存在，则在使用实用程序前在 RESTORE DATABASE 命令中指定 HISTORY FILE 选项。

如果使用 PRUNE HISTORY 命令的 FORCE 选项，可能会删除从最新的完整数据库备份映象进行恢复所需的条目。PRUNE HISTORY 命令的缺省操作可防止所需的条目被删除。建议不要使用 PRUNE HISTORY 命令的 FORCE 选项。

保留备份操作的完整记录，并使用此实用程序作为指导是一个好办法。

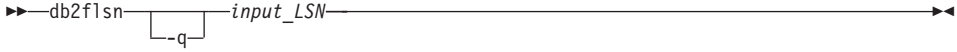
db2flsn - 查找日志序列号

返回包含由指定的日志序列号 (LSN) 标识的日志记录的文件名。

权限

无

命令语法



命令参数

-q 指定只打印日志文件名。将不打印任何错误或警告消息，状态只能通过返回码来确定。有效的错误码是:

- -100 无效输入
- -101 打不开 LFH 文件
- -102 未成功读取 LFH 文件
- -103 无效的 LFH
- -104 数据库不是可复原
- -105 LSN 太大
- -500 逻辑错误。

其他有效的返回码是:

- 0 成功执行
- 99 警告: 结果基于最后一个已知的日志文件大小。

input_LSN

一个 12 字节的字符串，它代表有前导零的内部（6 字节）十六进制值。

示例

```
db2flsn 000000BF0030
   Given LSN is contained in log file S0000002.LOG

db2flsn -q 000000BF0030
   S0000002.LOG

db2flsn 000000BE0030
   Warning: the result is based on the last known log file size.
   The last known log file size is 23 4K pages starting from log extent 2.
```

```
   Given LSN is contained in log file S0000001.LOG

db2flsn -q 000000BE0030
   S0000001.LOG
```

用法注释

日志头控制文件 sqlogctl.lfh 必须驻留在当前目录中。由于此文件位于数据库目录中，所以工具可以从数据库目录运行，或者控制文件可从运行工具的目录复制到该目录。

工具使用 *logfilsiz* 数据库配置参数。DB2 记录此参数的 3 个最近值，以及用每个 *logfilsiz* 值创建的第一个日志文件，从而使工具可在 *logfilsiz* 更改时正确工作。如果指定的 LSN 比 *logfilsiz* 的最早记录值要早，工具会使用该值并返回一个警告。该工具可与 UDB 版本 5.2 之前的数据库管理器一起使用；此时，即使结果正确（如果 *logfilsiz* 的值保持不变）也会返回警告。

此工具只能用于可复原的数据库。如果在配置数据库时，*logretain* 设置为 RECOVERY 或 *userexit* 设置为 ON，该数据库就是可复原的。

db2inidb - 初始化镜像数据库

在分割的镜像环境中，此命令用于为不同的目的而初始化镜像数据库。

权限

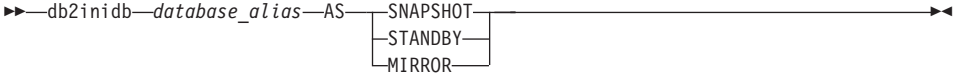
以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*

需要的连接

无

命令语法



命令参数

database_alias

指定要初始化的数据库的别名。

SNAPSHOT

指定要初始化为主数据库的克隆的镜像数据库。该数据库是只读数据库。

STANDBY

指定要置于前滚暂挂状态的数据库。可访存主数据库中的新日志，并应用到备用数据库。然后可在主数据库当机时，将该备用数据库用作主数据库。

MIRROR

指定将用作备份映象的镜像数据库，该备份映象可用于复原主数据库。

db2mscs - 设置 Windows NT 故障转移实用程序

创建此基础结构，使 DB2 在使用 Microsoft Cluster Server (MSCS) 的 Windows NT/2000 上支持故障转移。可使用此实用程序，以在单分区和分区数据库环境中都支持故障转移。

权限

用户必须登录到属于 MSCS 群集中每个机器的 Administrators 组的域用户帐户。

命令语法

命令参数

-f:input_file

指定 MSCS 实用程序要使用的 DB2MSCS.CFG 输入文件。若未指定此参数，则 DB2MSCS 实用程序读取当前目录中的 DB2MSCS.CFG 文件。

ARCHIVE LOG

关闭并截断某个可恢复数据库的活动日志文件。如果启用了用户出口，发出归档请求。

作用域

在 MPP 环境中，此命令关闭并截断所有节点上的活动日志；但是可指定节点的子集。

权限

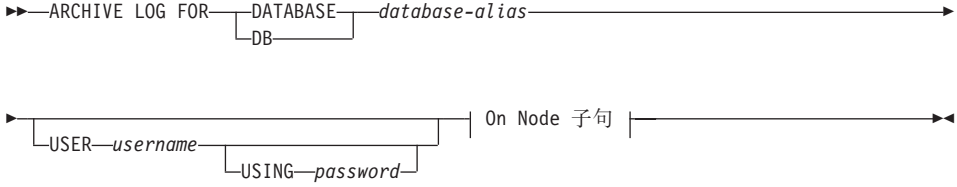
以下各项之一：

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

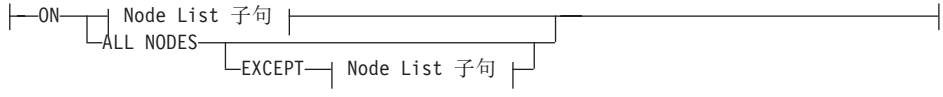
需要的连接

此命令自动建立与指定数据库的连接。如果连接已存在，会返回错误。

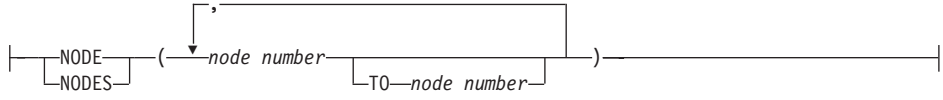
命令语法



On Node 子句:



Node List clause:



命令参数

DATABASE database-alias
指定要归档其活动日志的数据库的别名。

USER username
标识尝试连接时要使用的用户名。

USING password
指定认证用户名的密码。

ON ALL NODES
指定应对 db2nodes.cfg 文件中的所有节点发出该命令。如果未指定节点子句，则这是缺省值。

EXCEPT
指定应对 db2nodes.cfg 文件中的所有节点发出该命令，在节点列表中指定的那些节点除外。

ON NODE/ON NODES
指定应为一组节点上的指定数据库归档日志。

node number
指定指定节点中的一个节点。

TO node number
在指定要归档日志的节点范围时使用。从指定的第一个节点号到（并包括）指定的第二个节点号之前的所有节点都包括在节点列表中。

用法注释

此命令可用于收集到某个已知点为止的一套完整的日志文件。然后可将该日志文件用于更新备用数据库。

如果在调用此命令时，其他应用程序有正在进行中的事务，当日志缓冲区刷新到磁盘中，而其他事务尝试将日志记录写到缓冲区就必须等到刷新完成，此时会发现有轻微的性能降低。

此命令会导致数据库丢失部分 LSN 空间，从而加速了对有效 LSN 的消耗。

INITIALIZE TAPE

DB2 Windows NT/2000 版支持对流式磁带机的备份与复原操作。使用此命令进行磁带初始化。

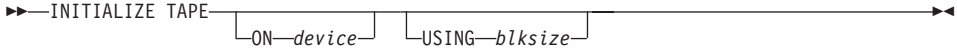
权限

无

需要的连接

无

命令语法



命令参数

ON device

指定有效的磁带机名称。缺省值是 \\.\TAPE0。

USING blksize

以字节为单位，指定设备的块大小。如果该值在设备受支持的块大小范围内，则初始化设备以使用指定的块大小。

注: 在 第72页的『BACKUP DATABASE 命令』和 第96页的『RESTORE DATABASE 命令』中指定的缓冲区大小必须是此处指定的块大小可分的。

如果未指定此参数的值，将初始化设备以使用它的缺省块大小。如果指定了值 0，会初始化该设备以使用变量长度块大小；如果设备不支持变量长度块方式，将返回一个错误。

另见

第293页的『REWIND TAPE』

第294页的『SET TAPE POSITION』.

LIST HISTORY

列示历史记录文件中的条目。历史记录文件包含对恢复和管理事件的记录。恢复事件包括完整的数据库和表空间级的备份、增量备份以及复原和前滚操作。所记录的其他事件包括：创建、改变或重命名表空间、运行统计信息、重组表、删除表和装入。

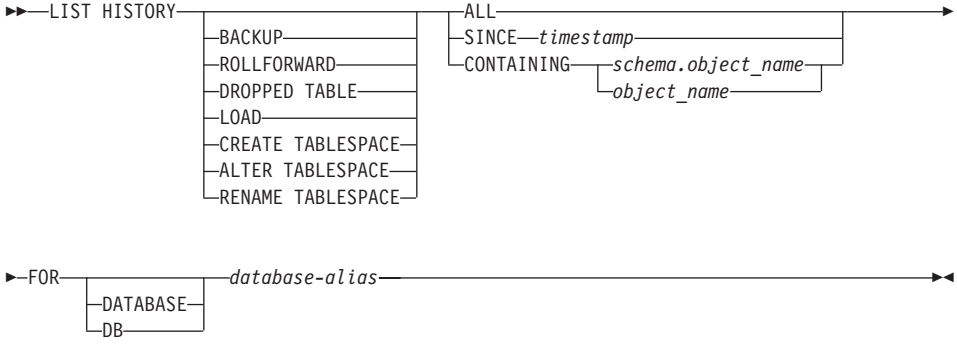
权限

无

需要的连接

实例。不需要显式连接。如果数据库列示为远程数据库，会为命令执行期间建立与远程节点的实例连接。

命令语法



命令参数

HISTORY
列出当前记录在历史记录文件中的全部事件。

BACKUP
列出备份与复原操作。

ROLLFORWARD
列出前滚操作。

DROPPED TABLE
列出已删除的表记录。

LOAD 列出装入操作。

CREATE TABLESPACE
列出表空间创建和删除运操作。

RENAME TABLESPACE
列出表空间重命名操作。

ALTER TABLESPACE
列出改变表空间操作。

ALL 列示历史记录文件中具有指定类型的所有条目。

SINCE timestamp
可指定的完整时间戳记（格式为 *yyyymmddhhnnss*），或初始前缀（至少为 *yyyy*）。时间戳记等于或大于所提供的时间戳记的所有条目都将列出。

CONTAINING schema.object_name
此限定名唯一地标识表。

CONTAINING object_name
此不限定名唯一地标识表空间。

FOR DATABASE database-alias
用于标识将列出其恢复历史记录文件的数据库。

示例

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

用法注释

此命令生成的报告包含以下符号:

操作
- A - 创建表空间
- B - 备份
- C - 装入副本
- D - 删除的表
- F - 前滚
- G - 重组表
- L - 装入
- N - 重命名表空间
- O - 删除表空间
- Q - 停顿
- R - 复原
- S - 运行统计信息
- T - 改变表空间
- U - 卸装

类型
备份类型:

F - 脱机
N - 联机
I - 增量脱机
O - 增量联机
D - Delta 脱机
E - Delta 联机

前滚类型:

E - 日志结束
P - 时间点

装入类型:

I - 插入
R - 替换

改变表空间类型:

C - 添加容器
R - 重新平衡

停顿类型:

S - 停顿共享
U - 停顿更新
X - 停顿铬占
Z - 停顿复位

如果已将前滚操作执行到了所有日志的末尾，而备份标识表示结束时间，则备份标识值为 99991231235959。

PRUNE HISTORY/LOGFILE

用于从恢复历史记录文件中删除条目，或从活动的日志文件路径中删除日志文件。如果文件变得特别大而保留期又特别长，可能会需要从恢复历史记录文件中删除条目。如果日志是手工归档的（而不是通过用户出口程序归档的），则可能需要从活动的日志文件路径中删除日志文件。

权限

以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

需要的连接

数据库

命令语法

命令参数

HISTORY timestamp

标识将删除的恢复历史记录文件中条目的范围。可指定一个完整的时间戳记（格式为 *yyyymmddhhmmss*）或初始前缀（最少为 *yyyy*）。时间戳记等于或小于所提供的时间戳记的所有条目都将从恢复历史记录文件中删除。

WITH FORCE OPTION

指定将根据指定的时间戳记来修剪条目，即使会从该文件中删除最新复原设置中的某些条目。复原集是最新的完整数据库备份，包括该备份映象的任何复原。如果未指定此参数，来自先前备份映象的所有条目都将保留在该历史记录中。

LOGFILE PRIOR TO log-file-name

指定一个字符串作为日志文件名，例如 S0000100.LOG。将删除指定的日志文件前（但不包括）的所有日志文件。LOGRETAIN 数据库配置参数必须设置为 RECOVERY 或 CAPTURE。

示例

要从恢复历史记录文件中除去在 1994 年 12 月 1 日前（包括这一天）建立的所有复原、装入、表空间备份以及完整数据库备份的条目，可输入:

```
db2 prune history 199412
```

注: 将 199412 解释为 19941201000000。

用法注释

来自历史记录文件的修剪备份条目会导致 DB2 Data Links Manager 服务器上的相关文件备份被删除。

REWIND TAPE

DB2 Windows NT/2000 版支持对流式磁带机的备份与复原操作。使用此命令进行磁带倒带。

权限

无

需要的连接

无

命令语法

命令参数

ON device
指定有效的磁带机名称。缺省值是 `\\.\TAPE0`。

另见

第287页的『INITIALIZE TAPE』

第294页的『SET TAPE POSITION』.

SET TAPE POSITION

DB2 Windows NT/2000 版支持对流式磁带机的备份与复原操作。使用此命令进行磁带定位。

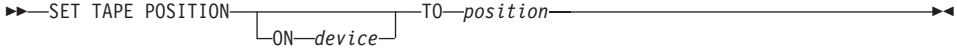
权限

无

需要的连接

无

命令语法



命令参数

ON device

指定有效的磁带机名称。缺省值是 `\\.\TAPE0`。

TO position

指定将定位磁带的标记。DB2 Windows NT/2000 版在每次备份映象后写磁带标记。值为 1 指定第一个位置，而 2 则指定第二个位置，以此类推。例如，如果将磁带定位在磁带标记 1，会定位归档 2 以进行复原。

另见

第287页的『INITIALIZE TAPE』

第293页的『REWIND TAPE』.

UPDATE HISTORY FILE

更新历史记录文件条目中的位置、设备类型或注释。

权限

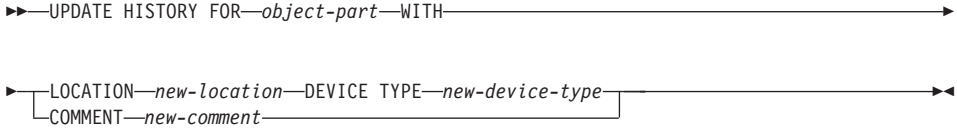
以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

需要的连接

数据库

命令语法



命令参数

FOR object-part
指定备份或复制映象的标识符。它是一个时间戳记，并带有从 001 到 999 的可选序列号。

LOCATION new-location
指定备份映象的新物理位置。对此参数的解释取决于设备类型。

DEVICE TYPE new-device-type
指定用于存储备份映象的新设备类型。有效的设备类型是:

D 磁盘

K 软盘

T 磁带

A TSM

U 用户出口

O 其他

COMMENT new-comment
指定描述该条目的新注释。

示例

要更新在 1997 年 4 月 13 日，上午 10:00 所建立的完整数据库备份的历史记录文件条目，可输入:

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

用法注释

该历史记录文件由数据库管理员用于保留记录。它由 DB2 在内部用于自动恢复增量备份。

另见

第291页的『PRUNE HISTORY/LOGFILE』.

附录D. 附加 API 及相关数据结构

db2ArchiveLog - 归档活动日志 API

关闭或截断某个可恢复数据库的活动日志文件。如果启用了用户出口，发出归档请求。

作用域

在 MPP 环境中，此 API 关闭并截断所有节点上的活动日志。

权限

以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

需要的连接

此 API 自动建立与指定数据库的连接。如果连接已存在，会返回错误。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2ArchiveLog (
      db2Uint32 version,
      void * pDB2ArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Archive Active Log */
/* ... */
SQL_API_RC SQL_API_FN
   db2gArchiveLog (
      db2Uint32 version,
      void * pDB2gArchiveLogStruct,
      struct sqlca * pSqlca);

typedef struct
{
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char * piDatabaseAlias;
   char * piUserName;
   char * piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE * piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
/* ... */
```

API 参数

version

输入。指定作为第二个参数 *pDB2ArchiveLogStruct* 传入的变量的版本和发行级别。

pDB2ArchiveLogStruct

输入。指向 *db2ArchiveLogStruct* 结构的指针。

pSqlca

输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

iAliasLen

输入。一个 4 字节的无符号整数，代表以字节计的数据库别名的长度。

iUserNameLen

输入。一个 4 字节的无符号整数，代表以字节计的用户名的长度。如果不使用用户名，则设置为 0。

iPasswordLen

输入。一个 4 字节的无符号整数，代表以字节计的密码的长度。如果不使用密码，则设置为 0。

piDatabaseAlias

输入。一个包含数据库别名的字符串（如系统数据库目录中编目的那样），将对该数据库进行活动日志归档。

piUserName

输入。包含尝试连接时将使用的用户名的字符串。

piPassword

输入。包含尝试连接时将使用的密码的字符串。

iAllNodeFlag

输入。仅用于 MPP。标志，它指示是否应对 `db2nodes.cfg` 文件中的所有节点应用该操作。有效的值为:

DB2ARCHIVELOG_ALL_NODES

对所有节点应用（piNodeList 应为 NULL）。这是缺省值。

DB2ARCHIVELOG_NODE_LIST

对发送到 *piNodeList* 中的节点列表中指定的所有节点应用。

DB2ARCHIVELOG_ALL_EXCEPT

对除发送到 *piNodeList* 中的节点列表中指定的那些节点以外的所有节点应用。

iNumNodes

输入。仅用于 MPP。指定 *piNodeList* 数组中的节点号。

piNodeList

输入。仅用于 MPP。指向要应用归档日志操作的节点号数组的指针。

iOptions

输入。保留以备将来使用。

用法注释

此 API 可用于收集到某个已知点为止的日志文件的完整集合。然后日志文件可用于更新备用数据库。

如果调用此 API 时，其他应用程序已有事务在进行中，将日志缓冲区刷新到磁盘中时，会发现性能略有降低；其他事务尝试将日志记录写到缓冲区必须等待刷新完成。

此 API 会导致数据库丢失其 LSN 空间的一部分，因此加速消耗有效的 LSN。

db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API

结束扫描恢复历史记录文件的操作，并释放扫描所需的 DB2 资源。必须在成功地调用了 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』 之后才能调用该 API。

权限

无

需要的连接

实例。在调用此 API 前不需要调用 **sqleatin**。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2HistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Close Recovery History File Scan */
/* ... */
SQL_API_RC  SQL_API_FN
  db2GenHistoryCloseScan (
    db2Uint32 version,
    void * piHandle,
    struct sqlca * pSqlca);
/* ... */
```

API 参数

version

输入。指定第二个参数 *piHandle* 的版本和发行等级。

piHandle

输入。指定由 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』 返回的扫描访问的句柄。

pSqlca

输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

REXX API 语法

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX API 参数

scanid

包含从 OPEN RECOVERY HISTORY FILE SCAN 返回的扫描标识符的主机变量。

用法注释

关于使用恢复历史记录文件 API 的详细描述，参见 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』。

另见

第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』

第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』

第316页的『db2Prune API』

第313页的『db2HistoryUpdate - 更新恢复历史记录文件 API』.

db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API

获取恢复历史记录文件中的下一个条目。必须在成功地调用了 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』 之后才能调用该 API。

权限

无

需要的连接

实例。在调用此 API 前不需要调用 **sqleatin**。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
    db2Uint32 version,
    void * pDB2HistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2HistoryGetEntryStruct;
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Get Next Recovery History File Entry */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryGetEntry (
    db2Uint32 version,
    void * pDB2GenHistoryGetEntryStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint16 iHandle,
  db2Uint16 iCallerAction,
  struct db2HistData * pioHistData
} db2GenHistoryGetEntryStruct;
/* ... */
```

API 参数

version

输入。指定作为第二个参数 *pDB2HistoryGetEntryStruct* 传入的结构的版本和发行等级。

pDB2HistoryGetEntryStruct

输入。指向 *db2HistoryGetEntryStruct* 结构的一个指针。

pSqlca

输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

iHandle

输入。包含由 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』 返回的扫描访问的句柄。

iCallerAction

输入。指定要采取的操作的类型。有效值（在 db2ApiDf 中定义）是:

DB2HISTORY_GET_ENTRY

获取下一个条目，但不带有任何命令数据。

DB2HISTORY_GET_DDL

只获取前一次访存中的命令数据。

DB2HISTORY_GET_ALL

获取下一个条目，包括所有数据。

pioHistData

输入。指向 *db2HistData* 结构的一个指针。关于此结构的更多信息，请参阅第323页的『数据结构: db2HistData』。

REXX API 语法

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

REXX API 参数

scanid

包含从 OPEN RECOVERY HISTORY FILE SCAN 返回的扫描标识符的主机变量。

value 一个组合 REXX 主机变量，将向它返回恢复历史记录文件条目信息。在下例中，XXX 表示主机变量名:

XXX.0	变量中第一级元素的数目（总为 15）
XXX.1	表空间元素的数目
XXX.2	已使用的表空间元素的数目
XXX.3	OPERATION（执行的操作类型）
XXX.4	OBJECT（操作的粒度）
XXX.5	OBJECT_PART（时间戳记和序列号）
XXX.6	OPTYPE（操作限定符）
XXX.7	DEVICE_TYPE（所使用的设备的类型）
XXX.8	FIRST_LOG（最早的日志标识）
XXX.9	LAST_LOG（当前日志标识）
XXX.10	BACKUP_ID（备份标识符）
XXX.11	SCHEMA（表名限定符）
XXX.12	TABLE_NAME（装入的表的名称）
XXX.13.0	NUM_OF_TABLESPACES（备份或复原过程中所涉及的表空间号）
XXX.13.1	备份／复原的第一个表空间的名称
XXX.13.2	备份／复原的第二个表空间的名称

XXX.13.3 以此类推

XXX.14 LOCATION（存储备份或副本的位置）

XXX.15 COMMENT（用于描述条目的文本）。

用法注释

返回的记录应已使用在调用 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』 时指定的值进行了选择。

关于使用恢复历史记录文件 API 的详细描述，参见 第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』。

另见

第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』

第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』

第316页的『db2Prune API』

第313页的『db2HistoryUpdate - 更新恢复历史记录文件 API』.

db2HistoryOpenScan - 打开恢复历史记录文件扫描 API

开始扫描恢复历史记录文件。

权限

无

需要的连接

实例。在调用此 API 前不需要调用 **sqleatin**。如果数据库编目为远程数据库，会建立与远程节点的实例连接。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
    db2Uint32 version,
    void * pDB2HistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2HistoryOpenStruct;
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Open Recovery History File Scan */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryOpenScan (
    db2Uint32 version,
    void * pDB2GenHistoryOpenStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piDatabaseAlias,
  char * piTimestamp,
  char * piObjectName,
  db2Uint32 oNumRows,
  db2Uint16 iCallerAction,
  db2Uint16 oHandle
} db2GenHistoryOpenStruct;
/* ... */
```

API 参数

version

输入。指定作为第二个参数 *pDB2HistoryOpenStruct* 传入的结构的版本和发行等级。

pDB2HistoryOpenStruct

输入。指向 *db2HistoryOpenStruct* 结构的一个指针。

pSqlca

输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

piDatabaseAlias

输入。指向包含数据库别名的字符串的指针。

piTimestamp

输入。指向字符串的一个指针，该字符串指定要用于选择记录的时间戳记。将选择那些时间戳记等于或大于此值的记录。将此参数设置为 NULL，或指向零，可防止过滤使用时间戳记的条目。

piObjectName

输入。指向字符串的一个指针，该字符串指定要用于选择记录的对象名。对象可以是表或表空间。如果是表，必须提供全限定表名。将此参数设置为 NULL，或指向零，可防止过滤使用对象名的条目。

oNumRows

输出。在 API 返回的基础上，此参数包含匹配的恢复历史记录文件条目数。

iCallerAction

输入。指定要采取的操作的类型。有效值（在 db2ApiDf 中定义）是:

DB2HISTORY_LIST_HISTORY

列出当前记录在历史记录文件中的全部事件。

DB2HISTORY_LIST_BACKUP

列出备份与复原操作。

DB2HISTORY_LIST_ROLLFORWARD

列出前滚操作。

DB2HISTORY_LIST_DROPPED_TABLE

列出已删除的表记录。不返回与条目相关的 DDL 字段。要检索某个条目的 DDL 信息，应在访存了该条目这后立即调用 第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』，并带有调用程序操作 DB2HISTORY_GET_DDL。

DB2HISTORY_LIST_LOAD

列出装入操作。

DB2HISTORY_LIST_CRT_TABLESPACE

列出表空间创建和删除运操作。

DB2HISTORY_LIST_REN_TABLESPACE

列出表空间重命名操作。

DB2HISTORY_LIST_ALT_TABLESPACE

列出改变表空间操作。不返回与条目相关的 DDL 字段。要检索某个条目的 DDL 信息，应在访存了该条目这后立即调用 第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』，并带有调用程序操作 DB2HISTORY_GET_DDL。

DB2HISTORY_LIST_RUNSTATS

列出 RUNSTATS 操作。此值当前不受支持。

DB2HISTORY_LIST_REORG

列出 REORGANIZE TABLE 操作。此值当前不受支持。

oHandle

输出。在 API 返回的基础上，此参数包含扫描访问的句柄。它随后会在 第304页的

304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』和 第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』中使用。

REXX API 语法

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

REXX API 参数

database_alias
要列出其历史记录文件的数据库的别名。

objname
指定要用于选择记录的对象名称。对象可以是表或表空间。如果是表，必须提供全限定表名。将此参数设置为 NULL 可防止过滤使用 *objname* 的条目。

timestamp
指定要用于选择记录的时间戳记。将选择那些时间戳记等于或大于此值的记录。将此参数设置为 NULL 可防止过滤使用 *timestamp* 的条目。

value 一个组合 REXX 主机变量，会将恢复历史记录文件信息返回给它。以下例中，XXX 表示主机变量名。

XXX.0 变量的元素数（总为 2）

XXX.1 未来扫描访问的标识符（句柄）

XXX.2 匹配的恢复历史记录文件条目数。

用法注释

时间戳记、对象名称和调用程序操作的组合可用于过滤记录。只返回发送所有指定的过滤器的记录。

对象名称的过滤效果取决于指定的值:

- 因为装入操作的记录是历史记录文件中表的仅有信息，所以指定一个表将返回装入操作的记录。
- 指定表空间将返回表空间的备份、复原和装入操作的记录。

注: 要返回表的记录，必须将表指定为 *schema.tablename*。指定 *tablename* 将只返回表空间的返回记录。

允许每个进程最多扫描 8 个历史记录文件。

要列出历史记录文件中的每个条目，典型的应用程序通常会执行以下步骤:

1. 调用 **db2HistoryOpenScan**，它将返回 *oNumRows*。
2. 分配 *db2HistData* 结构，它的 *n oTablespace* 字段表示空间，其中 *n* 是任意数。
3. 将 *db2HistData* 的 *iDB2NumTablespace* 字段设置为 *n*。
4. 在循环中，执行以下操作:
   - 调用 **db2HistoryGetEntry** 以从历史记录文件进行访存。
   - 如果 **db2HistoryGetEntry** 返回 SQLCODE 为 SQL_RC_OK，则使用 *db2HistData* 结构的 *sqld* 字段来确定返回的表空间条目数。
   - 如果 **db2HistoryGetEntry** 返回 SQLCODE 为 SQLUH_SQLUHINFO_VARS_WARNING，表示未对 DB2 尝试返回的所有表空间分配足够的空间；应释放并重新分配 *db2HistData* 结构，使它的 *oDB2UsedTablespace* 表空间条目有足够的空间，然后将 *iDB2NumTablespace* 设置为 *oDB2UsedTablespace*。
   - 如果 **db2HistoryGetEntry** 返回 SQLCODE 为 SQLE_RC_NOMORE，表示已检索了所有恢复历史记录文件条目。
   - 其他任何 SQLCODE 都表示有问题。
5. 访存了所有信息后，可调用 第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』 以释放通过调用 **db2HistoryOpenScan** 而分配的资源。

提供了在 sqlutil 中定义的宏 SQLUHINFOSIZE(*n*)，以帮助您确定在使用 *n oTablespace* 字段表示空间的 *db2HistData* 结构中需要多少内存。

另见

第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』

第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』

第316页的『db2Prune API』

第313页的『db2HistoryUpdate - 更新恢复历史记录文件 API』.

db2HistoryUpdate - 更新恢复历史记录文件 API

更新历史记录文件条目中的位置、设备类型或注释。

权限

以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

需要的连接

数据库。要更新非缺省数据库的历史记录文件中的条目，必须在调用该 API 之前先建立与该数据库的连接。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
    db2Uint32 version,
    void * pDB2HistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2HistoryUpdateStruct;
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Update Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenHistoryUpdate (
    db2Uint32 version,
    void * pDB2GenHistoryUpdateStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piNewLocation,
  char * piNewDeviceType,
  char * piNewComment,
  db2Uint32 iEID
} db2GenHistoryUpdateStruct;
/* ... */
```

API 参数

version
输入。指定作为第二个参数 *pDB2HistoryUpdateStruct* 传入的结构的版本和发行等级。

pDB2HistoryUpdateStruct
输入。指向 *db2HistoryUpdateStruct* 结构的一个指针。

pSqlca
输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

piNewLocation
输入。指向字符串的一个指针，该字符串指定备份、复原或装入副本映象的新位置。将此参数设置为 NULL，或指向零，使该值保持不变。

piNewDeviceType
输入。指向字符串的一个指针，该字符串指定存储备份、复原或装入副本映象的新设备类型。将此参数设置为 NULL，或指向零，使该值保持不变。

piNewComment
输入。指向字符串的一个指针，该字符串指定描述条目的新注释。将此参数设置为 NULL，或指向零，使该注释保持不变。

iEID 输入。可用于更新历史记录文件中特定条目的唯一标识符。

REXX API 语法

```
UPDATE RECOVERY HISTORY USING :value
```

REXX API 参数

value 一个组合 REXX 主机变量，包含与恢复历史记录文件条目的新位置相关的信息。以下例中，XXX 表示主机变量名:

XXX.0 变量的元素数（必须在 1 和 4 之间）

XXX.1 OBJECT_PART（序列号从 001 到 999 的时间戳记）

XXX.2 备份或副本映象的新位置（此参数是可选的）

XXX.3 用于存储备份或副本映象的新设备（此参数是可选的）

XXX.4 新注释（此参数是可选的）。

用法注释

这是一个更新功能，在此更改之前的所有信息都将被替换并不可重新创建。这些更改不会记入日志。

历史记录文件只用于进行记录。它不会由复原或前滚功能直接使用。在复原操作期间，可指定备份映象的位置，也可将历史记录文件用于跟踪位置。然后可向备份实用程序提供信息（参见第75页的『备份数据库 API』）。同样，如果移动了装入副本映象的位置，则必须提供带有新位置和存储媒体类型的前滚实用程序。有关附加信息，参见第131页的『前滚数据库 API』。

另见

第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』

第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』

第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』

第316页的『db2Prune API』.

db2Prune API

从恢复历史记录文件或日志文件中删除条目，或从活动的日志路径中删除日志文件。

权限

以下各项之一:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

需要的连接

数据库。要从任何非缺省数据库的恢复历史记录文件中删除条目，必须在调用该 API 之前先建立与该数据库的连接。

API 包含文件

db2ApiDf.h

C API 语法

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2Prune (
    db2Uint32 version,
    void * pDB2PruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2PruneStruct;
/* ... */
```

一般 API 语法

```
/* File: db2ApiDf.h */
/* API: Prune Recovery History File */
/* ... */
SQL_API_RC SQL_API_FN
  db2GenPrune (
    db2Uint32 version,
    void * pDB2GenPruneStruct,
    struct sqlca * pSqlca);

typedef struct
{
  db2Uint32 iStringLen;
  char * piString,
  db2Uint32 iEID,
  db2Uint32 iCallerAction,
  db2Uint32 iOptions
} db2GenPruneStruct;
/* ... */
```

API 参数

version

输入。指定作为第二个参数 *pDB2PruneStruct* 传入的结构的版本和发行等级。

pDB2PruneStruct

输入。指向 *db2PruneStruct* 结构的一个指针。

pSqlca

输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

iStringLen

输入。指定 *piString* 的长度，以字节为单位。

piString

输入。指向字符串的一个指针，它指定时间戳记或日志序列号 (LSN)。时间戳记或时间戳记的一部分（最少是 *yyyy* 或 year）用于选择要删除的记录。所有等于或小于该时间戳记的条目都将被删除。必须提供有效的时间戳记；NULL 参数没有缺省行为。

此参数还可用于发送 LSN，因此可修剪不活动的日志。

iEID 输入。指定可用于修剪历史记录文件中的单个条目的唯一标识符。

iCallerAction

输入。指定要采取的操作的类型。有效值（在 db2ApiDf 中定义）是:

DB2PRUNE_ACTION_HISTORY

除去历史记录文件条目。

DB2PRUNE_ACTION_LOG

从活动的日志路径中除去日志文件。

iOptions

输入。有效值（在 db2ApiDf 中定义）是:

DB2PRUNE_OPTION_FORCE

强制除去上一个备份。

DB2PRUNE_OPTION_LSNSTRING

指定 *piString* 的值是 LSN，在指定了 DB2PRUNE_ACTION_LOG 的调用程序操作时使用。

REXX API 语法

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

REXX API 参数

timestamp

包含了时间戳记的主机变量。时间戳记等于或小于所提供的时间戳记的所有条目都将从恢复历史记录文件中删除。

WITH FORCE OPTION

如果指定该选项，将根据指定的时间戳记来修剪恢复历史记录文件，即使会从该文件中删除最新复原设置中的某些条目。如果不指定该选项，将保留最新的复原设置，即使时间戳记小于或等于作为输入指定的时间戳记。

用法注释

修剪历史记录文件并不会删除实际的备份或装入文件。用户必须手工删除这些文件才能释放它们在存储媒体上所消耗的空间。

注意:
如果从媒体上删除了最新的完整数据库备份（不仅是从历史记录文件中对它进行了修剪），用户必须确保所有表空间，包括目录表空间和用户表空间都已备份。未成功进行备份可能会导致数据库无法恢复，或数据库中用户数据的某些部分丢失。

另见

第302页的『db2HistoryCloseScan - 关闭恢复历史记录文件扫描 API』

第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』

第308页的『db2HistoryOpenScan - 打开恢复历史记录文件扫描 API』

第313页的『db2HistoryUpdate - 更新恢复历史记录文件 API』.

sqlurlog - 异步读日志 API

使调用程序可以从数据库日志中抽取某些日志记录，并向“日志管理器”查询当前日志状态信息。只能对可恢复的数据库使用此 API。如果在配置数据库时，*logretain* 设置为 RECOVERY 或 *userexit* 设置为 ON，该数据库就是可恢复的。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sqlutil.h

C API 语法

```
/* File: sqlutil.h */
/* API: Asynchronous Read Log */
/* ... */
SQL_API_RC SQL_API_FN
  sqlurlog (
    sqluint32 CallerAction,
    SQLU_LSN * pStartLsn,
    SQLU_LSN * pEndLsn,
    char * pLogBuffer,
    sqluint32 LogBufferSize,
    SQLU_RLOG_INFO * pReadLogInfo,
    struct sqlca * pSqlca);
/* ... */
```

API 参数

CallerAction

输入。指定要执行的操作。

SQLU_RLOG_READ

按照日志序列号，从最开始到最末尾读取数据库日志，然后返回该范围内所有可传播的日志记录。

SQLU_RLOG_READ_SINGLE
从起始日志序列号读取单个的日志记录（无论该日志记录是可传播的还是不可传播的）。

SQLU_RLOG_QUERY
查询数据库日志。查询结果将通过 SQLU_RLOG_INFO 结构发送回来（参见第329页的『数据结构: SQLU-RLOG-INFO』）。

pStartLsn
输入。起始日志序列号指定了读取日志的起始关系字节地址。该值必须是某个实际日志记录的开始。

pEndLsn
输入。结束日志序列号指定了读取日志的结束关系字节地址。此值必须大于 *startLsn*，且不必是某个实际日志记录的末尾。

pLogBuffer
输出。按顺序存储在指定范围内读取的所有可传播日志记录的缓冲区。该缓冲区的大小必须至少能保留一个日志记录。建议此缓冲区应有最少 32 个字节。它的最大大小取决于要求的范围的大小。缓冲区中的每个日志记录都有一个 6 字节日志序列号 (LSN) 的前缀，表示后跟的日志记录的 LSN。

LogBufferSize
输出。以字节为单位指定日志缓冲区的大小。

pReadLogInfo
输出。与调用和数据库日志相关的结构详细信息。关于此结构的更多信息，参见第329页的『数据结构: SQLU-RLOG-INFO』。

pSqlca
输出。指向 *sqlca* 结构的一个指针。关于此结构的更多信息，请参阅 *Administrative API Reference* 或 *SQL Reference*。

样本程序

C \sqllib\samples\c\asynrlog.sqc

用法注释

如果请求的操作是读取日志，调用程序会提供日志序列号范围以及用于保留日志记录的缓冲区。此 API 按顺序读取日志（该顺序是由请求的 LSN 范围确定的），并返回与具有 DATA CAPTURE 选项 CHANGES 的表相关的日志记录，以及提供了当前活动的日志信息的 SQLU_RLOG_INFO 结构。如果请求的操作是查询，则此 API 会返回提供了当前活动的日志信息的 SQLU_RLOG_INFO 结构。

要使用“异步日志阅读器”，首先应查询数据库日志是否有有效的起始 LSN。在查询调用之后，读取日志信息结构 (SQLU-RLOG-INFO) 可获得一个将用于读取调用的有效起始 LSN（在 initialLSN 成员中）。当前活动日志的末尾将在读取日志信息结构的 curActiveLSN 成员中。用作读取操作的结束 LSN 的值可以是以下其中一个:

- curActiveLSN 的值
- 大于 initialLSN 的值
- FFFF FFFF FFFF，由异步日志阅读器解释为当前日志的末尾。

关于读取日志信息结构的更多信息，参见第329页的『数据结构: SQLU-RLOG-INFO』。

在起始和结束 LSN 范围内读取的可传播日志记录都将返回到日志缓冲区中。日志记录不包含它的 LSN，它包含在缓冲区中实际的日志记录前。**sqlurlog** 返回的各种 DB2 日志记录的描述都可以在 *Administrative API Reference*中找到。

要按顺序读取初始读取后的下一个日志记录，可以向 SQLU-RLOG-INFO 中返回的最后一个读取的 LSN 加 1。重新提交该调用，使用新的起始 LSN 和有效的结束 LSN。然后会读取下一个记录块。*sqlca* 代码 SQLU_RLOG_READ_TO_CURRENT 表示日志阅读器已读到了当前活动日志的结尾。

数据结构: db2HistData

此结构用于在调用 第304页的『db2HistoryGetEntry - 获取下一个恢复历史记录文件条目 API』 后返回信息。

表 17. db2HistData 结构中的字段

字段名称	数据类型	描述
ioHistDataID	char(8)	一个 8 字节的结构标识符和存储器转储的 “眼球捕获器”。唯一有效的值是 ″SQLUHINF″。此字符串不存在符号定义。
oObjectPart	db2Char	首 14 个字符是格式为 *yyyymmddhhnnss* 的时间戳记，它表示操作开始的时间。接着的 3 个字符是序列号。当备份映象保存在多个文件或多个磁带上时，每个备份操作都会在此文件中产生多个条目。序列号允许指定多个位置。复原和装入操作在此文件中只有一个条目，对应于相应备份的序列号 ’001’。时间戳记与序列号组合在一起，必须是唯一的。
oEndTime	db2Char	格式为 *yyyymmddhhnnss* 的时间戳记，它表示操作完成的时间。
oFirstLog	db2Char	最早的日志文件标识（范围从 S0000000 到 S9999999）： • 是对联机备份应用前滚恢复所需的 • 是对脱机备份应用前滚恢复所需的 • 复原当装入操作开始时处于当前状态的完整的数据库级或表空间级备份后应用
oLastLog	db2Char	最新的日志文件标识（范围从 S0000000 到 S9999999）： • 是对联机备份应用前滚恢复所需的 • 是对脱机备份将前滚恢复应用到当前时间点所需的 • 复原当装入操作结束时处于当前状态的完整数据库级或表空间级备份时应用（如果未应用前滚恢复，将与 *oFirstLog* 相同）。
oID	db2Char	唯一的备份或表标识符。
oTableQualifier	db2Char	表限定符。
oTableName	db2Char	表名。

表 17. db2HistData 结构中的字段 (续)

字段名称	数据类型	描述
oLocation	db2Char	对于备份和装入副本，此字段表示保存数据的位置。对于需要本文件中的多个条目的操作，由 *oObjectPart* 定义的序列号标识了在指定的位置中找到了哪部分的备份。对于复原和装入操作，位置总是标识已保存的复原或装数据的第一部分（对应于多部分备份的序列号 '001'。*oLocation* 中的数据取决于 *oDeviceType*，有不同的解释: • 对于磁盘或软盘（D 或 K），它是全限定文件名 • 对于磁带 (T)，它是卷标 • 对于 TSM (A)，它是服务器名称 • 对于用户出口或其他（U 或 O），它是自由格式文本。
oComment	db2Char	自由格式文本注释。
oCommandText	db2Char	命令文本或 DDL。
oLastLSN	SQLU_LSN	上一日志序列号。
oEID	Structure	唯一条目标识符。
poEventSQLCA	Structure	已记录的事件的结果 *sqlca*。关于 *sqlca* 结构的信息，请参阅 *Administrative API Reference* 和 *SQL Reference*。
poTablespace	db2Char	表空间名称的列表。
ioNumTablespaces	db2Uint32	*poTablespace* 列表中的条目数。每个表空间备份都包含一个或多个表空间。每个表空间复原操作都替换一个或多个表空间。如果此字段不为零（表示表空间级的备份和复原），此文件中的下一行包含了备份或复原的表空间名称，由 18 个字符的字符串表示。一个表空间名称出现在一行上。
oOperation	char	参见第325页的表18。
oObject	char	操作的粒度: D 表示完整的数据库、P 表示表空间而 T 表示表。
oOptype	char	参见第325页的表19。
oStatus	char	条目状态: D 表示已删除（将来使用）、E 表示已到期、I 表示不活动的、N 表示尚未落实而 Y 表示已落实或活动的。
oDeviceType	char	设备类型。此字段确定如何解释 *oLocation* 字段: A 表示 TSM、C 表示客户机、D 表示磁盘、K 表示软盘、L 表示本地、O 表示其他（其他供应商设备支持）、P 表示管道、S 表示服务器、T 表示磁带而 U 表示用户出口。

表 18. db2HistData 结构中的有效 oOperation 值

值	描述	C 定义	COBOL/FORTRAN 定义
A	添加表空间	DB2HISTORY_OP_ADD_TABLESPACE	DB2HIST_OP_ADD_TABLESPACE
B	备份	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	装入副本	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	删除的表	DB2HISTORY_OP_DROPPED_TABLE	DB2HIST_OP_DROPPED_TABLE
F	前滚	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	识别表	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	装入	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	重命名表空间	DB2HISTORY_OP_REN_TABLESPACE	DB2HIST_OP_REN_TABLESPACE
O	删除表空间	DB2HISTORY_OP_DROP_TABLESPACE	DB2HIST_OP_DROP_TABLESPACE
Q	停顿	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	复原	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
S	运行统计信息	DB2HISTORY_OP_RUNSTATS	DB2HIST_OP_RUNSTATS
T	改变表空间	DB2HISTORY_OP_ALT_TABLESPACE	DB2HIST_OP_ALT_TBS
U	卸装	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD

表 19. db2HistData 结构中的有效 oOptype 值

oOperation	oOptype	描述	C/COBOL/FORTRAN 定义
B	F	脱机	DB2HISTORY_OPTYPE_OFFLINE
	N	联机	DB2HISTORY_OPTYPE_ONLINE
	I	增量脱机	DB2HISTORY_OPTYPE_INCR_OFFLINE
	O	增量联机	DB2HISTORY_OPTYPE_INCR_ONLINE
	D	delta 脱机	DB2HISTORY_OPTYPE_DELTA_OFFLINE
	E	delta 联机	DB2HISTORY_OPTYPE_DELTA_ONLINE
F	E	日志结束	DB2HISTORY_OPTYPE_EOL
	P	时间点	DB2HISTORY_OPTYPE_PIT
L	I	插入	DB2HISTORY_OPTYPE_INSERT
	R	替换	DB2HISTORY_OPTYPE_REPLACE

表 19. db2HistData 结构中的有效 oOptype 值 (续)

oOperation	oOptype	描述	C/COBOL/FORTRAN 定义
Q	S	停顿共享	DB2HISTORY_OPTYPE_SHARE
	U	停顿更新	DB2HISTORY_OPTYPE_UPDATE
	X	停顿独占	DB2HISTORY_OPTYPE_EXCL
	Z	停顿复位	DB2HISTORY_OPTYPE_RESET
R	F	脱机	DB2HISTORY_OPTYPE_OFFLINE
	N	联机	DB2HISTORY_OPTYPE_ONLINE
	I	增量脱机	DB2HISTORY_OPTYPE_INCR_OFFLINE
	O	增量联机	DB2HISTORY_OPTYPE_INCR_ONLINE
T	C	添加容器	DB2HISTORY_OPTYPE_ADD_CONT
	R	重新平衡	DB2HISTORY_OPTYPE_REB

表 20. db2Char 结构中的字段

字段名称	数据类型	描述
pioData	char	指向字符数据缓冲区的指针。如果为 NULL，将不返回任何数据。
iLength	db2Uint32	输入。*pioData* 缓冲区的大小。
oLength	db2Uint32	输出。*pioData* 缓冲区中有效的数据字符数。

表 21. db2HistoryEID 结构中的字段

字段名称	数据类型	描述
ioNode	SQL_PDB_NODE_TYPE	节点号。
ioHID	db2Uint32	本地历史记录文件入口标识。

语言语法

C 结构

```
/* File: db2ApiDf.h */
/* ... */
typedef SQL_STRUCTURE db2HistoryData
{
  char ioHistDataID[8];
  db2Char oObjectPart;
  db2Char oEndTime;
  db2Char oFirstLog;
  db2Char oLastLog;
  db2Char oID;
  db2Char oTableQualifier;
  db2Char oTableName;
  db2Char oLocation;
  db2Char oComment;
  db2Char oCommandText;
  SQLU_LSN oLastLSN;
  db2HistoryEID oEID;
  struct sqlca * poEventSQLCA;
  db2Char * poTablespace;
  db2Uint32 ioNumTablespaces;
  char oOperation;
  char oObject;
  char oOptype;
  char oStatus;
  char oDeviceType
} db2HistoryData;

typedef SQL_STRUCTURE db2Char
{
  char * pioData;
  db2Uint32 ioLength
} db2Char;

typedef SQL_STRUCTURE db2HistoryEID
{
  SQL_PDB_NODE_TYPE ioNode;
  db2Uint32 ioHID
} db2HistoryEID;
/* ... */
```

数据结构: SQLU-LSN

此并集由 第320页的『sqlurlog - 异步读日志 API』 使用，包含了日志序列号的定义。日志序列号 (LSN) 代表数据库日志中一个相对的字节地址。所有日志记录都由此号码来标识。它代表日志记录从数据库日志开始处的字节偏移。

表 22. SQLU-LSN 并集中的字段

字段名称	数据类型	描述
lsnChar	Array of UNSIGNED CHAR	指定由 6 个成员组成的字符数组日志序列号。
lsnWord	Array of UNSIGNED SHORT	指定由 3 个成员组成的短数组日志序列号。

语言语法

C 结构

```
typedef union SQLU_LSN
{
unsigned char  lsnChar  [6] ;
unsigned short lsnWord  [3] ;
} SQLU_LSN;
```

数据结构: SQLU-RLOG-INFO

此结构中的信息包括对 第320页的『sqlurlog - 异步读日志 API』 的调用的状态, 以及数据库日志。

表 23. SQLU-RLOG-INFO 结构中的字段

字段名称	数据类型	描述
initialLSN	SQLU_LSN	指定在发出第一个数据库 CONNECT 语句后写的第一个日志记录的 LSN 值。有关更多信息, 参见第328页的『数据结构: SQLU-LSN』。
firstReadLSN	SQLU_LSN	指定读取的第一个日志记录的 LSN 值。
lastReadLSN	SQLU_LSN	指定读取的最后一个日志记录的 LSN 值。
curActiveLSN	SQLU_LSN	指定当前（活动的）日志的 LSN 值。
logRecsWritten	sqluint32	指定写到缓冲区中的日志记录数。
logBytesWritten	sqluint32	指定写到缓冲区中的字节数。

语言语法

C 结构

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
SQLU_LSN       initialLSN ;
SQLU_LSN       firstReadLSN ;
SQLU_LSN       lastReadLSN ;
SQLU_LSN       curActiveLSN ;
sqluint32      logRecsWritten ;
sqluint32      logBytesWritten ;
} SQLU_RLOG_INFO;
```

附录E. 恢复样本程序

不带嵌入式 SQL 的样本程序 (backrest.c)

以下样本程序显示了如何使用 DB2 备份与复原 API 来:

- 备份数据库
- 复原数据库
- 前滚恢复数据库

关于 SAMPLE 数据库的详细信息，请参阅 *SQL Reference*。

此样本程序的源文件 (backrest.c) 可在 Windows 操作系统和 OS/2 上的 \sqllib\samples\c 目录中找到，对于基于 UNIX 的系统，该文件在 /sqllib/samples/c 目录中。它包含了一些 DB2 API。makefile 位于相同的目录中，包含构建这个和其他样本程序的命令。关于创建包含 DB2 管理 API 的应用程序的通用信息，以及关于编译和链接选项的详细信息，请参阅《*应用程序构建指南*》。要在 Windows NT 或 Windows 2000 上，从源文件 backrest.c 构建样本程序 backrest:

1. 将文件 backrest.c、makefile、utilapi.c 和 utilapi.h 复制到工作目录中。
2. 如果数据库管理器不是正在运行，可从 DB2 命令窗口发出 **db2start** 命令。要打开启用了 CLP 的 DB2 窗口，并初始化操作系统上的 DB2 命令行环境，可从命令提示符发出 **db2cmd**。
3. 输入 nmake backrest。生成以下文件:

```
backrest.exe
backrest.ilk
backrest.obj
backrest.pdb
utilapi.obj
```

要运行样本程序（可执行文件），输入:

```
backrest database userID password
```

例如:

```
backrest sample db2admin db2admin
```

以下内容是此程序返回的输出示例:

不带嵌入式 SQL 的样本程序 (backrest.c)

```
This is sample program : backrest.c

NOTE:   Ensure the database is not in use prior to running this program.

Updating the sample database configuration parameter LOGRETAIN
to 'ON' to enable rollforward recovery.

Backing up the 'sample' database.
The database has been successfully backed up.

Updating the sample database configuration parameter LOGRETAIN
to 'OFF'.

Restoring the database 'sample' as 'TESTBACK' (1st pass).
Should get returned value = SQLUD_INACCESSABLE_CONTAINER.
SQLUD_INACCESSABLE_CONTAINER is returned.

Need to SET TABLESPACES CONTAINERS
Tablespace container information for tablespace 1 obtained.
Tablespace containers have been set for tablespace 1.

Restoring the database 'sample' as 'TESTBACK' (2nd pass).
Database sample has been restored as 'TESTBACK'.

The database 'TESTBACK' has been successfully rolled forward.

The database 'TESTBACK' has been successfully dropped.
```

以下是样本程序的源列表:

```
/*****************************************************************************
**
** Source File Name = backrest.c
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 1995, 2000
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**    PURPOSE :
**       an example showing how to use BACKUP & RESTORE APIs in order to:
**          - backup a database
**          - restore a database
**          - roll forward database to return the database to a state it was
**            in prior to the damage...
**
**    APIs USED :
**       BACKUP DATABASE                  sqlubkp()
**       RESTORE DATABASE                 sqlurestore()
**       UPDATE DATABASE CONFIGURATION    sqlfudb()
**       GET TABLESPACE CONTAINER QUERY   sqlbtcq()
**       SET TABLESPACE CONTAINERS        sqlbstsc()
```

```
**       ROLLFORWARD DATABASE              sqluroll()
**
**    STRUCTURES USED :
**       sqlu_tablespace_bkrst_list
**       SQLB_TBSCONTQRY_DATA
**       sqlu_media_list
**       rfwd_input
**       rfwd_output
**       sqlurf_info
**       sqlca
**
**    OTHER FUNCTIONS DECLARED :
**       'C' COMPILER LIBRARY :
**          stdio.h  -  printf
**
**       internal :
**
**       external :
**          check_error :     Checks for SQLCODE error, and prints out any
**          [in UTIL.C]          related information available.
**                               This procedure is located in the UTIL.C file.
**
**    EXTERNAL DEPENDENCIES :
**       - Ensure existence of database for precompile purposes.
**       - Compile and link with the IBM Cset++ compiler (AIX and OS/2)
**         or the Microsoft Visual C++ compiler (Windows)
**         or the compiler supported on your platform.
**
** For more information about these samples see the README file.
**
** For more information on programming in C, see the:
**   -  "Programming in C and C++" section of the Application Development Guide
** For more information on building C applications, see the:
**   -  "Building C Applications" section of the Application Building Guide.
**
** For more information on the SQL language see the SQL Reference.
**
******************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlutil.h>
#include <sqlenv.h>
#include "utilapi.h"

/* You can change the following path to another directory for which */
/* you have permission. create the "backup" directory in the same path. */

   #define TEMPDIR "."

int main (int argc, char *argv[]) {

   /* Variables for the BACKUP API */
   sqluint32 buff_size;
```

```
sqluint32 num_buff;
struct sqlu_tablespace_bkrst_list *tablespace_list;
struct SQLB_TBSCONTQRY_DATA *cont;
struct sqlca sqlca;
struct sqlu_media_entry media_entry;
struct sqlu_media_list media_list;
char applid[SQLU_APPLID_LEN+1];
char timestamp[SQLU_TIME_STAMP_LEN+1];
char *target_path;

sqluint32 tsc_id;
sqluint32 num_cont;

/* variables for the Update Database Configuration API */
struct sqlfupd itemList;
short log_retain;

/* variables for the ROLLFORWARD API */
struct rfwd_input rfwdInput;
char dbAlias[] = "TESTBACK";
char overFlowLogPath[SQL_PATH_SZ] = TEMPDIR;

struct rfwd_output rfwdOutput;
sqlint32   numReplies;
struct sqlurf_info nodeInfo;

printf ("\nThis is sample program : backrest.c\n\n");

printf ("NOTE:   Ensure the database is not in use prior to running
 this program.\n\n");

if (argc != 4) {
   printf ("\nUSAGE: backrest database userID password\n\n");
   return 1;
} /* endif */

/* Altering the default Database Configuration to enable rollforward
   recovery of the TESTBACK database. */

log_retain = 1;
itemList.token = SQLF_DBTN_LOG_RETAIN;

itemList.ptrvalue = (char *) &log_retain;


printf ("Updating the %s database configuration parameter
 LOGRETAIN \n", argv[1]);
printf ("to 'ON' to enable rollforward recovery.\n\n");

/*****************************************\
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");
```

```
/* Initialize the BACKUP API variables */
 buff_size = 16;
 num_buff = 1;
 tablespace_list = NULL;
 media_list.media_type = 'L';
 media_list.sessions = 1;

 strcpy (media_entry.media_entry, TEMPDIR);

 media_list.target.media = &media_entry;
 printf ("Backing up the '%s' database.\n", argv[1]);
 /*****************\
 * BACKUP DATABASE *
 \*****************/
 sqlubkp (argv[1],
           buff_size,
           SQLUB_OFFLINE,
           SQLUB_FULL,
           SQLUB_BACKUP,
           applid,
           timestamp,
           num_buff,
           tablespace_list,
           &media_list,
           argv[2],
           argv[3],
                            NULL,
           0,
                            NULL,
           1,
                            NULL,
                            NULL,
                            NULL,
           &sqlca);

 API_SQL_CHECK("backing up the database");
 printf ("The database has been successfully backed up.\n\n");



 /* Altering the default Database Configuration to disable rollforward
    recovery of the TESTBACK database. */

 log_retain = 0;
 itemList.token = SQLF_DBTN_LOG_RETAIN;
 itemList.ptrvalue = (char *) &log_retain;


 printf ("Updating the %s database configuration parameter
  LOGRETAIN \n", argv[1]);
 printf ("to 'OFF'.\n\n");

 /*****************************************\
```

```
* UPDATE DATABASE CONFIGURATION API called *
\*****************************************/
sqlfudb(argv[1], 1, &itemList, &sqlca);
API_SQL_CHECK("updating the database configuration");



/* Initialize the variables for the RESTORE API */
buff_size = 1024;
target_path = NULL;
printf ("Restoring the database '%s' as 'TESTBACK' (1st pass).\n", argv[1]);
printf ("Should get returned value = SQLUD_INACCESSABLE_CONTAINER.\n");
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore(argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_RESTORE_STORDEF,
          applid,
          timestamp,
          target_path,
          num_buff,
                          NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
                          NULL,
          0,
                          NULL,
          1,
                          NULL,
                          NULL,
                          NULL,
          &sqlca);
if (sqlca.sqlcode != SQLUD_INACCESSABLE_CONTAINER)
   API_SQL_CHECK("restoring database");

printf ("SQLUD_INACCESSABLE_CONTAINER is returned.\n\n");
printf ("Need to SET TABLESPACES CONTAINERS\n");
/* Set the containers structure to a new list of containers */
tsc_id = 1;
cont = (struct SQLB_TBSCONTQRY_DATA *) malloc
   (sizeof(struct SQLB_TBSCONTQRY_DATA));

/*******************************\
* GET TABLESPACE CONTAINER QUERY *
\*******************************/
sqlbtcq (&sqlca, tsc_id, &num_cont, &cont);
```

```
API_SQL_CHECK("GET TABLESPACE CONTAINER QUERY");
printf ("Tablespace container information for tablespace 1 obtained.\n");
/**************************\
* SET TABLESPACE CONTAINERS *
\**************************/
sqlbstsc (&sqlca,
          SQLB_SET_CONT_INIT_STATE,
          tsc_id,
          num_cont,
          cont);
API_SQL_CHECK("SET TABLESPACE CONTAINERS");
printf ("Tablespace containers have been set for tablespace 1.\n\n");

printf ("Restoring the database '%s' as 'TESTBACK' (2nd pass).\n", argv[1]);
/******************\
* RESTORE DATABASE *
\******************/
sqlurestore (argv[1],
          "TESTBACK",
          buff_size,
          SQLUD_ROLLFWD,
          SQLUD_DATALINK,
          SQLUD_FULL,
          SQLUD_OFFLINE,
          SQLUD_CONTINUE,
          applid,
          timestamp,
          target_path,
          num_buff,
                           NULL,
          tablespace_list,
          &media_list,
          argv[2],
          argv[3],
                           NULL,
          0,
                           NULL,
          1,
                           NULL,
                           NULL,
                           NULL,
          &sqlca);
API_SQL_CHECK("restoring database 2");
printf ("Database %s has been restored as 'TESTBACK'.\n\n", argv[1]);


/**********************\
* ROLLFORWARD DATABASE *
\**********************/
rfwdInput.version=SQLUM_RFWD_VERSION;
rfwdInput.pDbAlias=dbAlias;
rfwdInput.CallerAction=SQLUM_ROLLFWD;
rfwdInput.pStopTime=SQLUM_INFINITY_TIMESTAMP;
rfwdInput.pUserName=argv[2];
rfwdInput.pPassword=argv[3];
```

不带嵌入式 SQL 的样本程序 (backrest.c)

```
    rfwdInput.pOverflowLogPath=overFlowLogPath;
    rfwdInput.NumChngLgOvrflw=0;
    rfwdInput.pChngLogOvrflw=NULL;
    rfwdInput.ConnectMode=SQLUM_OFFLINE;
    rfwdInput.pTablespaceList=NULL;
    rfwdInput.AllNodeFlag= SQLURF_ALL_NODES;
    rfwdInput.NumNodes=0;
    rfwdInput.pNodeList=NULL;
    rfwdInput.NumNodeInfo=1;
    rfwdInput.DlMode=0;
    rfwdInput.pReportFile=NULL;
    rfwdInput.pDroppedTblID=NULL;
    rfwdInput.pExportDir=NULL;


    rfwdOutput.pApplicationId=applid;
    rfwdOutput.pNumReplies=&numReplies;
    rfwdOutput.pNodeInfo=&nodeInfo;


    sqluroll( &rfwdInput,
              &rfwdOutput,
              &sqlca);
    API_SQL_CHECK("rolling database forward");
    printf ("The database 'TESTBACK' has been successfully rolled
     forward.\n\n", argv[1]);


    /***************\
    * DROP DATABASE *
    \***************/
    sqledrpd ("TESTBACK",
              &sqlca);
    API_SQL_CHECK("DROP DATABASE");
    printf ("The database 'TESTBACK' has been successfully dropped.\n");


    return 0;
}
```

带有嵌入式 SQL 的样本程序 (dbrecov.sqc)

以下样本程序显示了如何使用 DB2 备份与复原 API 来:

- 备份数据库
- 复原数据库
- 前滚恢复数据库

关于 SAMPLE 数据库的详细信息，请参阅 *SQL Reference*。

注: 在 DB2 版本 7.2 中未安装 dbrecov 样本文件。只可以从 Web 上下载。

要下载 dbrecov 样本程序所需的文件:

1. 转至 http://www.ibm.com/software/data/db2/udb/ad/v7/abg.html。
2. 在 ″Recent Updates″ 下，单击 ″dbrecov sample files by platform″。
3. 在出现的页上，选择适当的链接，以下载适用于您的平台的文件。对于 Windows 操作系统和 OS/2，该文件包含在一个已压缩的可执行文件中；而对于基于 UNIX 的系统，该文件包含在 tar.gz 文件中。

注意:
不要在现有 DB2 UDB 版本 7 样本文件所驻留的目录中运行此可执行文件，因为新实用程序文件将覆盖现有版本的 DB2 UDB 版本 7 实用程序文件。新实用程序文件与其他样本程序不兼容。

此样本程序 (dbrecov.sqc) 的源文件包含了 DB2 API 和 嵌入式 SQL 调用。关于创建包含 DB2 管理 API 的应用程序的通用信息，以及关于编译和链接选项的详细信息，请参阅*《应用程序构建指南》*。

以下是在 Windows 操作系统上构建和运行 dbrecov 程序的指示信息。要在另一受支持的操作系统上构建程序，可参见样本文件中附带的自述文件。

1. 在您的工作目录上运行下载的可执行文件 dbrecov_win.exe。从而解压缩以下文件:

```
dbrecov.sqc - 用于 dbrecov 程序的嵌入式 SQL 源文件
utilapi.c   - 用于 DB2 API 程序的错误检查实用程序
utilapi.h   - utilapi.c 的头文件
utilemb.sqc - 用于嵌入式 SQL 程序的错误检查实用程序文件
utilemb.h   - utilemb.sqc 的头文件
bldmapp.bat - 构建 Microsoft Visual C++ 应用程序
bldvapp.bat - 构建 VisualAge C++ 应用程序
embprep.bat - 预编译并绑定嵌入式 SQL 程序
README      - 包含关于该程序文件的信息
```

2. 如果数据库管理器不是正在运行，可从 DB2 命令窗口发出 **db2start** 命令。要打开启用了 CLP 的 DB2 窗口，并初始化操作系统上的 DB2 命令行环境，可从命令提示符发出 **db2cmd**。
3. 取决于您的编译器，输入 bldmapp dbrecov 或 bldvapp dbrecov。生成以下文件:

```
dbrecov.exe
dbrecov.ilk
dbrecov.c
dbrecov.obj
dbrecov.bnd
dbrecov.pdb
utilemb.c
utilemb.obj
```

要运行样本程序（可执行文件），输入:

```
dbrecov
```

取决于您的操作系统环境，输出会不同。以下内容是该程序在 Windows 系统上返回的输出的示例:

```
THIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.

USE THE DB2 API:
  sqlfxdb -- Get Database Configuration
TO GET THE DATABASE CONFIGURATION AND DETERMINE
THE SERVER WORKING PATH.

NOTE: Backup images will be created on the server
      in the directory D:\DB2,
      and will not be deleted by the program.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
CONNECT  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

**************************************
*** BACK UP AND RESTORE A DATABASE ***
**************************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurestore -- Restore Database
TO BACK UP AND RESTORE A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162032
```

```
  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162032
    - target database        : sample

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database restore -- start
line                = 506
file                = dbrecov.sqc
SQLCODE             = 2539

SQL2539W  Warning!  Restoring to an existing database that is the same as the
backup image database.  The database files will be deleted.

---- end warning report ----------------------

  Continuing the restore operation...

  Restore finished.

*************************
*** REDIRECTED RESTORE ***
*************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqlbmtsq -- Tablespace Query
  sqlbtcq -- Tablespace Container Query
  sqlbstsc -- Set Tablespace Containers
  sqlefmem -- Free Memory
  sqledrpd -- Drop Database
TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.

  Update 'sample' database configuration:
    - Disable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = OFF/NO

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162125

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162125
    - target database        : RRDB

-- The following warning report is expected! --
```

```
---- warning report ---------------------------

application message = database restore -- start
line               = 776
file               = dbrecov.sqc
SQLCODE            = 1277

SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.

---- end warning report ----------------------

  Continuing the restore operation...

  Find and redefine inaccessible containers.

    Redefine inaccessible container:
      - table space name: SYSCATSPACE
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0000.0
      - container type: path
      - new container name: D:\DB2\SQLT0000.0

    Redefine inaccessible container:
      - table space name: TEMPSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0001.0
      - container type: path
      - new container name: D:\DB2\SQLT0001.0

    Redefine inaccessible container:
      - table space name: USERSPACE1
      - default container name: D:\DB2\NODE0000\SQL00003\SQLT0002.0
      - container type: path
      - new container name: D:\DB2\SQLT0002.0

  Restore finished.

  Drop the 'RRDB' database.

****************************
*** ROLLFORWARD RECOVERY ***
****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlecrea -- Create Database
  sqlurestore -- Restore Database
  sqluroll -- Rollforward Database
  sqledrpd -- Drop Database
TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD.

  Update 'sample' database configuration:
    - Enable the configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES
```

```
  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162223

  Restoring a database ...
    - source image alias     : sample
    - source image time stamp: 20010506162223
    - target database        : RFDB

  Restore finished.

  Rolling 'RFDB' database forward ...
  Rollforward finished.

  Drop the 'RFDB' database.

*****************************
*** ASYNCHRONOUS READ LOG ***
*****************************

USE THE DB2 APIs:
  sqlfudb -- Update Database Configuration
  sqlubkp -- Backup Database
  sqlurlog -- Asynchronous Read Log
AND THE SQL STATEMENTS:
CONNECT  ALTER TABLE
COMMITINSERT
DELETE
ROLLBACK
  CONNECT RESET
TO READ LOG RECORDS FOR THE CURRENT CONNECTION.

  Update 'sample' database configuration:
    - Enable the database configuration parameter LOGRETAIN
        i.e., set LOGRETAIN = RECOVERY/YES

  Backing up the 'sample' database...
  Backup finished.
    - backup image size      : 9 MB
    - backup image path      : D:\DB2
    - backup image time stamp: 20010506162247

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Invoke the following SQL statements:
    ALTER TABLE emp_resume DATA CAPTURE CHANGES;
    COMMIT;
    INSERT INTO emp_resume
      VALUES('000777', 'ascii', 'This is a new resume.');
            ('777777', 'ascii', 'This is another new resume');
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000777';
```

```
    DELETE FROM emp_resume WHERE empno = '777777';
    COMMIT;
    DELETE FROM emp_resume WHERE empno = '000140';
    ROLLBACK;
    ALTER TABLE emp_resume DATA CAPTURE NONE;
    COMMIT;

  Start reading database log.

-- The following warning report is expected! --
---- warning report ---------------------------

application message = database log info -- get
line               = 1457
file               = dbrecov.sqc
SQLCODE            = 2653

SQL2653W  A Restore, Forward or Crash Recovery may have reused log sequence
number ranges.  Reason code "1".

---- end warning report ----------------------

    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: ON

    Record type: Normal
      component ID: DMS log record
      function ID: Update Record
        oldRID: 2
        old subrecord length: 76
        old subrecord offset: 0
        subrecord type: Updatable, Internal control
        newRID: 2
        new subrecord length: 76
        new subrecord offset: 2916
        subrecord type: Updatable, Internal control

    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK

    Record type: Normal
      component ID: DMS log record
      function ID: Insert Record
        RID: 12
        subrecord length: 88
        subrecord offset: 486
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        30 30 30 37 37 37 0F 00 05 00 *000777....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
        49 06 01 00 00 00 00 00 15 00 *I.........*
```

```
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Insert Record
    RID: 13
    subrecord length: 88
    subrecord offset: 398
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 12
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 37 37 37 0F 00 05 00 *000777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 15 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 15 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 13
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
```

```
    37 37 37 37 37 37 0F 00 05 00 *777777....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 1A 00 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 3C 00 01 00 00 00 1A 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 01 00 00 00 *..........*

Record type: Normal commit
  UTC transaction committed (in seconds since 01/01/70): 989180591
  authorization ID of the application: MELNYK

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 6
    subrecord length: 88
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 05 00 *000140....*
    14 00 3C 00 00 61 73 63 69 69 *.....ascii*
    49 06 01 00 00 00 00 00 24 05 *I.........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 01 3C 00 02 00 00 00 24 05 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 14 00 00 00 *..........*

Record type: Normal
  component ID: DMS log record
  function ID: Delete Record
    RID: 7
    subrecord length: 89
    subrecord offset: 0
    subrecord type: Updatable, Formatted user data
    user data fixed length: 15
    user data:
    30 30 30 31 34 30 0F 00 06 00 *000140....*
    15 00 3C 00 00 73 63 72 69 70 *.....scrip*
    74 49 06 01 00 00 00 00 00 56 *tI.......V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 01 3C 00 02 00 00 00 56 *.........V*
    07 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 00 00 00 *..........*
    00 00 00 00 00 00 00 16 00 00 *..........*
    00                            *.         *

Record type: Compensation
  component ID: DMS log record
  function ID: Undo Delete Record
    RID: 7
    subrecord length: 89
```

```
        subrecord offset: 397
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        30 30 30 31 34 30 0F 00 06 00 *000140....*
        15 00 3C 00 00 73 63 72 69 70 *.....scrip*
        74 49 06 01 00 00 00 00 00 56 *tI.......V*
        07 00 00 00 00 00 00 00 00 00 *..........*
        00 00 01 3C 00 02 00 00 00 56 *.........V*
        07 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 16 00 00 *..........*
        00                            *.         *

    Record type: Compensation
      component ID: DMS log record
      function ID: Undo Delete Record
        RID: 6
        subrecord length: 88
        subrecord offset: 309
        subrecord type: Updatable, Formatted user data
        user data fixed length: 15
        user data:
        30 30 30 31 34 30 0F 00 05 00 *000140....*
        14 00 3C 00 00 61 73 63 69 69 *.....ascii*
        49 06 01 00 00 00 00 00 24 05 *I.........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 01 3C 00 02 00 00 00 24 05 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 00 00 00 00 *..........*
        00 00 00 00 00 00 14 00 00 00 *..........*

    Record type: Normal abort
      authorization ID of the application: MELNYK

    Record type: Normal
      component ID: DMS log record
      function ID:  Alter Table Attribute
        Propagation attribute is changed to: OFF

    Record type: Local pending list
      UTC transaction committed (in seconds since 01/01/70): 989180591
      authorization ID of the application: MELNYK

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

*******************************************
*** READ A DATABASE RECOVERY HISTORY FILE ***
*******************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryCloseScan -- Close Recovery History File Scan
```

```
TO READ A DATABASE RECOVERY HISTORY FILE.

  Open recovery history file for 'sample' database.

  Read entry number 0.

  Display entry number 0.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 1.

  Display entry number 1.
    object part: 20010506162058001
    end time: 200105061621
    first log: S0000000
    last log: S0000000
    ID: 20010506162032
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 RESTORE SAMPLE NO RF
    command text:
    history file entry ID: 49
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: R
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 2.
```

```
  Display entry number 2.
    object part: 20010506162125001
    end time: 200105061622
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 50
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 3.

  Display entry number 3.
    object part: 20010506162223001
    end time: 200105061622
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 51
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Read entry number 4.

  Display entry number 4.
    object part: 20010506162247001
    end time: 200105061623
    first log: S0000000
```

```
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 52
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close recovery history file for 'sample' database.

*****************************************************
*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***
*****************************************************

USE THE DB2 APIs:
  db2HistoryOpenScan -- Open Recovery History File Scan
  db2HistoryGetEntry -- Get Next Recovery History File Entry
  db2HistoryUpdate -- Update Recovery History File
  db2HistoryCloseScan -- Close Recovery History File Scan
TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.

  Open the recovery history file for 'sample' database.

  Read the first entry in the recovery history file.

  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: D:\DB2\SAMPLE.0\DB2\NODE0000\CATN0000\20010506
    comment: DB2 BACKUP SAMPLE OFFLINE
    command text:
    history file entry ID: 48
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
```

```
      sqlcode: 0
      sqlstate:
      message:

  Connecting to 'sample' database...
  Connected to 'sample' database.

  Update the first entry in the history file:
    new location = 'this is the NEW LOCATION'
    new comment = 'this is the NEW COMMENT'

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.

  Close recovery history file for 'sample' database.

  Read the first recovery history file entry.

  Display the first entry.
    object part: 20010506162032001
    end time: 200105061620
    first log: S0000000
    last log: S0000000
    ID:
    table qualifier:
    table name:
    location: this is the NEW LOCATION
    comment: this is the NEW COMMENT
    command text:
    history file entry ID: 48
    table spaces:
SYSCATSPACEUSERSPACE1    type of operation: B
    granularity of the operation: D
    operation type: F
    entry status: A
    device type: D
    SQLCA:
      sqlcode: 0
      sqlstate:
      message:

  Close the recovery history file for 'sample' database.

**************************************
*** PRUNE THE RECOVERY HISTORY FILE ***
**************************************

USE THE DB2 API:
  db2Prune -- Prune Recovery History File
AND THE SQL STATEMENTS:
CONNECT  CONNECT RESET
TO PRUNE THE RECOVERY HISTORY FILE.

  Connecting to 'sample' database...
  Connected to 'sample' database.
```

```
  Prune the recovery history file for 'sample' database.

  Disconnecting from 'sample' database...
  Disconnected from 'sample' database.
```

以下是样本程序的源列表:

```
/*****************************************************************************
**
** Source File Name: dbrecov.sqc
**
** Licensed Materials - Property of IBM
**
** (C) COPYRIGHT International Business Machines Corp. 2001
** All Rights Reserved.
**
** US Government Users Restricted Rights - Use, duplication or
** disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
**
**
**    PURPOSE:
**    This sample shows how to recover a database.
**
**    APIs USED:
**        db2HistoryCloseScan -- Close Recovery History File Scan
**        db2HistoryGetEntry -- Get Next Recovery History File Entry
**        db2HistoryOpenScan -- Open Recovery History File Scan
**        db2HistoryUpdate -- Update Recovery History File
**        db2Prune -- Prune Recovery History File
**        sqlbmtsq -- Tablespace Query
**        sqlbstsc -- Set Tablespace Containers
**        sqlbtcq -- Tablespace Container Query
**        sqledrpd -- Drop Database
**        sqlefmem -- Free Memory
**        sqlfudb -- Update Database Configuration
**        sqlfxdb -- Get Database Configuration
**        sqlubkp -- Backup Database
**        sqlurestore -- Restore Database
**        sqlurlog -- Asynchronous Read Log
**        sqluroll -- Rollforward Database
**
** For detailed information about database backup and recovery, see the
** "Data Recovery and High Availability Guide and Reference". This manual will
** help you to determine which database and table space recovery methods are
** best suited to your business environment.
**
** For more information about the sample programs, see the README file.
**
** For more information about programming in C, see the
** "Programming in C and C++" section of the "Application Development Guide".
**
** For more information about building C applications, see the
** section for your compiler in the "Building Applications" chapter
** for your platform in the "Application Building Guide".
```

```
**
** For more information about SQL, see the "SQL Reference".
**
** For more information about DB2 APIs, see the "Administrative API Reference".
**
** For the latest information on programming, compiling, and running DB2
** applications, visit the DB2 application development Web site at
**     http://www.software.ibm.com/data/db2/udb/ad
**
******************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlutil.h>
#include <db2ApiDf.h>
#include "utilemb.h"

int DbRecoveryHistoryFilePrune(char *, char *, char *);
int DbBackupAndRestore(char *, char *, char *, char *, char *);
int DbBackupAndRedirectedRestore(char *, char *, char *, char *, char *);
int DbBackupRestoreAndRollforward(char *, char *, char *, char *, char *);
int DbLogRecordsForCurrentConnectionRead(char *, char *, char *, char *);
int DbRecoveryHistoryFileRead(char *);
int DbFirstRecoveryHistoryFileEntryUpdate(char *, char *, char *);


/* support function called by the main() */
int ServerWorkingPathGet(char *, char *);

/* support function called by DbBackupAndRedirectedRestore() */
int InaccessableContainersRedefine(char *);

/* support function called by DbBackupAndRedirectedRestore() and
   DbBackupRestoreAndRollforward() */
int DbDrop(char *);

/* support function called by DbLogRecordsForCurrentConnectionRead() */
int LogBufferDisplay(char *, sqluint32);
int LogRecordDisplay(char *, sqluint32, sqluint16, sqluint16);
int SimpleLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32);
int ComplexLogRecordDisplay(sqluint16, sqluint16, char *, sqluint32,
                            sqluint8, char *, sqluint32);
int LogSubRecordDisplay(char *, sqluint16);
int UserDataDisplay(char *, sqluint16);

/* support functions called by DbRecoveryHistoryFileRead() and
   DbFirstRecoveryHistoryFileEntryUpdate() */
int HistoryEntryDataFieldsAlloc(struct db2HistoryData *);
int HistoryEntryDisplay(struct db2HistoryData);
int HistoryEntryDataFieldsFree(struct db2HistoryData *);

/* DbCreate will create a new database on the server with the server's
    code page.
   Use this function only if you want to restore a remote database.
```

```
   This support function is being called by DbBackupAndRedirectedRestore()
   and DbBackupRestoreAndRollforward().  */
int DbCreate(char *, char *);

int main(int argc, char *argv[])
{
  int rc = 0;
  char nodeName[SQL_INSTNAME_SZ + 1];
  char serverWorkingPath[SQL_PATH_SZ + 1];
  char restoredDbAlias[SQL_ALIAS_SZ + 1];
  char redirectedRestoredDbAlias[SQL_ALIAS_SZ + 1];
  char rolledForwardDbAlias[SQL_ALIAS_SZ + 1];
  sqluint16 savedLogRetainValue;
  char dbAlias[SQL_ALIAS_SZ + 1];
  char user[USERID_SZ + 1];
  char pswd[PSWD_SZ + 1];

  /* check the command line arguments */
  rc = CmdLineArgsCheck3(argc, argv, dbAlias, nodeName, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  printf("\nTHIS SAMPLE SHOWS HOW TO RECOVER A DATABASE.\n");

  strcpy(restoredDbAlias, dbAlias);
  strcpy(redirectedRestoredDbAlias, "RRDB");
  strcpy(rolledForwardDbAlias, "RFDB");

  /* attach to a local or remote instance */
  rc = InstanceAttach(nodeName, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  printf("\nUSE THE DB2 API:\n");
  printf("  sqlfxdb -- Get Database Configuration\n");
  printf("TO GET THE DATABASE CONFIGURATION AND DETERMINE\n");
  printf("THE SERVER WORKING PATH.\n");

  /* get the server working path */
  rc = ServerWorkingPathGet(dbAlias, serverWorkingPath);
  if (rc != 0)
  {
    return rc;
  }

  printf("\nNOTE: Backup images will be created on the server\n");
  printf("      in the directory %s,\n", serverWorkingPath);
  printf("      and will not be deleted by the program.\n");

  /* call the sample functions */
  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);
```

```
  rc = DbBackupAndRestore(dbAlias,
                          restoredDbAlias,
                          user,
                          pswd,
                          serverWorkingPath);

  rc = DbBackupAndRedirectedRestore(dbAlias,
                                    redirectedRestoredDbAlias,
                                    user,
                                    pswd,
                                    serverWorkingPath);

  rc = DbBackupRestoreAndRollforward(dbAlias,
                                     rolledForwardDbAlias,
                                     user,
                                     pswd,
                                     serverWorkingPath);

  rc = DbLogRecordsForCurrentConnectionRead(dbAlias,
                                            user,
                                            pswd,
                                            serverWorkingPath);

  rc = DbRecoveryHistoryFileRead(dbAlias);

  rc = DbFirstRecoveryHistoryFileEntryUpdate(dbAlias, user, pswd);

  rc = DbRecoveryHistoryFilePrune(dbAlias, user, pswd);

  /* detach from the local or remote instance */
  rc = InstanceDetach(nodeName);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* end main */

int ServerWorkingPathGet(char dbAlias[], char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbConfigFields[1];
  char serverLogPath[SQL_PATH_SZ + 1];
  int len;

  /* get the server log path */
  /* SQLF_DBTN_LOGPATH is a token of the non-updatable database configuration
     parameter 'logpath'; it is used to get the server log path */
  dbConfigFields[0].token = SQLF_DBTN_LOGPATH;
  dbConfigFields[0].ptrvalue =
    (char *)malloc((SQL_PATH_SZ + 1) * sizeof(char));
```

```
  /* get database configuration */
  /* the API sqlfxdb returns the values of individual entries in a
     database configuration file */
  sqlfxdb(dbAlias, 1, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("server log path -- get");

  strcpy(serverLogPath, dbConfigFields[0].ptrvalue);
  free(dbConfigFields[0].ptrvalue);

  /* choose the server working path; if, for example, serverLogPath =
     "C:\DB2\NODE0001\....", we'll keep "C:\DB2" for the serverWorkingPath
     variable; backup images created in this sample will be placed under
     the 'serverWorkingPath' directory */
  len = (int)(strstr(serverLogPath, "NODE") - serverLogPath - 1);
  memcpy(serverWorkingPath, serverLogPath, len);
  serverWorkingPath[len] = '\0';

  return 0;
} /* ServerWorkingPathGet */

int DbCreate(char existingDbAlias[], char newDbAlias[])
{
  struct sqlca sqlca;
  char dbName[SQL_DBNAME_SZ + 1];
  char dbLocalAlias[SQL_ALIAS_SZ + 1];
  char dbPath[SQL_PATH_SZ + 1];
  struct sqledbdesc dbDescriptor;
  struct sqledbcountryinfo countryInfo;
  struct sqlfupd dbConfigFields[2];

  printf("\n  Create '%s' empty database with the same code set as
   '%s' database.\n",
         newDbAlias, existingDbAlias);

  /* initialize dbConfigFields */
  dbConfigFields[0].token = SQLF_DBTN_TERRITORY;
  dbConfigFields[0].ptrvalue = (char *)malloc(10 * sizeof(char));
  memset(dbConfigFields[0].ptrvalue, '\0', 10);
  dbConfigFields[1].token = SQLF_DBTN_CODESET;
  dbConfigFields[1].ptrvalue = (char *)malloc(20 * sizeof(char));
  memset(dbConfigFields[1].ptrvalue, '\0', 20);

  /* get two database configuration fields */
  sqlfxdb(existingDbAlias, 2, &dbConfigFields[0], &sqlca);
  DB2_API_CHECK("DB Config. -- Get");

  /* create a new database */
  strcpy(dbName, newDbAlias);
  strcpy(dbLocalAlias, newDbAlias);
  strcpy(dbPath, "");

  strcpy(dbDescriptor.sqldbdid, SQLE_DBDESC_2);
  dbDescriptor.sqldbccp = 0;
  dbDescriptor.sqldbcss = SQL_CS_NONE;
```

```
  strcpy(dbDescriptor.sqldbcmt, "");
  dbDescriptor.sqldbsgp = 0;
  dbDescriptor.sqldbnsg = 10;
  dbDescriptor.sqltsext = -1;
  dbDescriptor.sqlcatts = NULL;
  dbDescriptor.sqlusrts = NULL;
  dbDescriptor.sqltmpts = NULL;

  strcpy(countryInfo.sqldbcodeset, (char *)dbConfigFields[1].ptrvalue);
  strcpy(countryInfo.sqldblocale, (char *)dbConfigFields[0].ptrvalue);

  /* create database */
  sqlecrea(dbName,
           dbLocalAlias,
           dbPath,
           &dbDescriptor,
           &countryInfo,
           '\0',
                              NULL,
           &sqlca);
  DB2_API_CHECK("Database -- Create");

  /* free the allocated memory */
  free(dbConfigFields[0].ptrvalue);
  free(dbConfigFields[1].ptrvalue);

  return 0;
} /* DbCreate */

int DbDrop(char dbAlias[])
{
  struct sqlca sqlca;

  printf("\n  Drop the '%s' database.\n", dbAlias);

  /* drop and uncatalog the database */
  sqledrpd(dbAlias, &sqlca);
  DB2_API_CHECK("Database -- Drop");

  return 0;
} /* DbDrop */

int DbBackupAndRestore(char dbAlias[],
                       char restoredDbAlias[],
                       char user[],
                       char pswd[],
                       char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
```

```
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
sqluint32 restoreBufferSize;
sqluint32 rollforwardPendingState;
sqluint32 datalinkMode;
sqluint32 restoreType;
sqluint32 restoreMode;
sqluint32 restoreCallerAction;
char restoreAppId[SQLU_APPLID_LEN + 1];
char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
char *restoreTargetPath;
sqluint32 restoreBuffersNb;
struct sqlu_tablespace_bkrst_list restoreTablespaceList;
struct sqlu_media_list restoreSourceMediaList;
struct sqlu_media_entry restoreSourceMediaEntries[1];
sqluint32 restoreParallelismDegree;

printf("\n************************************\n");
printf("*** BACK UP AND RESTORE A DATABASE ***\n");
printf("************************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("TO BACK UP AND RESTORE A DATABASE.\n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN\n");
printf("        i.e., set LOGRETAIN = OFF/NO\n");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE     */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);
```

```
backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
                            NULL,
        0,
                            NULL,
        backupParallelismDegree,
        &backupImageSize,
                            NULL,
                            NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */

  /* depending on the sqlca.sqlcode value, user action may be */
  /* required, such as mounting a new tape */

  printf("\n  Continuing the backup operation...\n");

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
```

```
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
                           NULL,
          0,
                           NULL,
          backupParallelismDegree,
          &backupImageSize,
                           NULL,
                           NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/*****************************/
/*    RESTORE THE DATABASE    */
/*****************************/

restoreBufferSize = 1024; /*  1024 x 4KB */
rollforwardPendingState = SQLUD_NOROLLFWD;
datalinkMode = SQLUD_NODATALINK;
restoreType = SQLUD_FULL;
restoreMode = SQLUD_OFFLINE;
strcpy(restoreTimestamp, backupStartTimestamp);
restoreTargetPath = NULL;
restoreBuffersNb = 1;
restoreTablespaceList.num_entry = 0; /* number of table spaces */
restoreTablespaceList.tablespace = NULL;
restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
restoreSourceMediaList.sessions = 1; /* number of elements in the target */
restoreSourceMediaList.target.media = restoreSourceMediaEntries;
strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
restoreParallelismDegree = 1;

printf("\n  Restoring a database ...\n");
printf("    - source image alias     : %s\n", dbAlias);
printf("    - source image time stamp: %s\n", restoreTimestamp);
printf("    - target database        : %s\n", restoredDbAlias);

restoreCallerAction = SQLUD_RESTORE;
```

```
  /* The API sqlurestore is used to restore a database that has been backed
     up using the API sqlubkp. */
  sqlurestore(dbAlias,
              restoredDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
                              NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
                              NULL,
              0,
                              NULL,
              restoreParallelismDegree,
                              NULL,
                              NULL,
                              NULL,
              &sqlca);
/*  DB2_API_CHECK("database restore -- start");  */
  EXPECTED_WARN_CHECK("database restore -- start");

  while (sqlca.sqlcode != 0)
  {
    /* continue the restore operation */
    printf("\n  Continuing the restore operation...\n");

    /* depending on the sqlca.sqlcode value, user action may be
       required, such as mounting a new tape */

    restoreCallerAction = SQLUD_CONTINUE;

    /* restore the database */
    sqlurestore(dbAlias,
                restoredDbAlias,
                restoreBufferSize,
                rollforwardPendingState,
                datalinkMode,
                restoreType,
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                              NULL,
                &restoreTablespaceList,
```

```
                &restoreSourceMediaList,
                user,
                pswd,
                             NULL,
                0,
                             NULL,
                restoreParallelismDegree,
                             NULL,
                             NULL,
                             NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  return 0;
} /* DbBackupAndRestore */

int DbBackupAndRedirectedRestore(char dbAlias[],
                                 char restoredDbAlias[],
                                 char user[],
                                 char pswd[],
                                 char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;
```

```
printf("\n*************************\n");
printf("*** REDIRECTED RESTORE ***\n");
printf("*************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlecrea -- Create Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("  sqlbmtsq -- Tablespace Query\n");
printf("  sqlbtcq -- Tablespace Container Query\n");
printf("  sqlbstsc -- Set Tablespace Containers\n");
printf("  sqlefmem -- Free Memory\n");
printf("  sqledrpd -- Drop Database\n");
printf("TO BACK UP AND DO A REDIRECTED RESTORE OF A DATABASE.\n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Disable the database configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = OFF/NO\n");

/* SQLF_DBTN_LOG_RETAIN is a token of the updatable database configuration
   parameter 'logretain'; it is used to update the database configuration
   file */
dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* disable the database configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_DISABLE;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Disable");

/*******************************/
/*    BACK UP THE DATABASE     */
/*******************************/
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
```

```
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
                            NULL,
        0,
                            NULL,
        backupParallelismDegree,
        &backupImageSize,
                            NULL,
                            NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape */

  printf("\n  Continuing the backup operation...\n");

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
                            NULL,
          0,
                            NULL,
          backupParallelismDegree,
          &backupImageSize,
                            NULL,
                            NULL,
```

```
            &sqlca);
    DB2_API_CHECK("Database -- Backup");
  }

  printf("  Backup finished.\n");
  printf("    - backup image size      : %d MB\n", backupImageSize);
  printf("    - backup image path      : %s\n",
         backupTargetMediaEntries[0].media_entry);
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);

  /* To restore a remote database, you will first need to create an empty
     database if the client's code page is different from the server's
     code page.
     If this is the case, uncomment the call to DbCreate(). It will create
     an empty database on the server with the server's code page. */

/*
  rc = DbCreate(dbAlias, restoredDbAlias);
  if (rc != 0)
  {
  return rc;
  }
*/

  /*****************************/
  /*    RESTORE THE DATABASE    */
  /*****************************/

  restoreBufferSize = 1024; /*  1024 x 4KB */
  rollforwardPendingState = SQLUD_NOROLLFWD;
  datalinkMode = SQLUD_NODATALINK;
  restoreType = SQLUD_FULL;
  restoreMode = SQLUD_OFFLINE;
  strcpy(restoreTimestamp, backupStartTimestamp);
  restoreTargetPath = NULL;
  restoreBuffersNb = 1;
  restoreTablespaceList.num_entry = 0; /* number of table spaces */
  restoreTablespaceList.tablespace = NULL;
  restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
  restoreSourceMediaList.sessions = 1; /* number of elements in the target */
  restoreSourceMediaList.target.media = restoreSourceMediaEntries;
  strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
  restoreParallelismDegree = 1;

  printf("\n  Restoring a database ...\n");
  printf("    - source image alias     : %s\n", dbAlias);
  printf("    - source image time stamp: %s\n", restoreTimestamp);
  printf("    - target database        : %s\n", restoredDbAlias);

  restoreCallerAction = SQLUD_RESTORE_STORDEF;

  /* The API sqlurestore is used to restore a database that has been backed
     up using the API sqlubkp. */
  sqlurestore(dbAlias,
```

```
                    restoredDbAlias,
                    restoreBufferSize,
                    rollforwardPendingState,
                    datalinkMode,
                    restoreType,
                    restoreMode,
                    restoreCallerAction,
                    restoreAppId,
                    restoreTimestamp,
                    restoreTargetPath,
                    restoreBuffersNb,
                                    NULL,
                    &restoreTablespaceList,
                    &restoreSourceMediaList,
                    user,
                    pswd,
                                    NULL,
                    0,
                                    NULL,
                    restoreParallelismDegree,
                                    NULL,
                                    NULL,
                                    NULL,
                    &sqlca);
  /*  DB2_API_CHECK("database restore -- start"); */
    EXPECTED_WARN_CHECK("database restore -- start");

    while (sqlca.sqlcode != 0)
    {
      /* continue the restore operation */
      printf("\n  Continuing the restore operation...\n");

      /* depending on the sqlca.sqlcode value, user action may be
         required, such as mounting a new tape */

      if (sqlca.sqlcode == SQLUD_INACCESSABLE_CONTAINER)
      {
        /* redefine the table space container layout */
        printf("\n  Find and redefine inaccessable containers.\n");
        rc = InaccessableContainersRedefine(serverWorkingPath);
        if (rc != 0)
        {
          return rc;
        }
      }

      restoreCallerAction = SQLUD_CONTINUE;

      /* restore the database */
      sqlurestore(dbAlias,
                  restoredDbAlias,
                  restoreBufferSize,
                  rollforwardPendingState,
                  datalinkMode,
                  restoreType,
```

```
                restoreMode,
                restoreCallerAction,
                restoreAppId,
                restoreTimestamp,
                restoreTargetPath,
                restoreBuffersNb,
                              NULL,
                &restoreTablespaceList,
                &restoreSourceMediaList,
                user,
                pswd,
                              NULL,
                0,
                              NULL,
                restoreParallelismDegree,
                              NULL,
                              NULL,
                              NULL,
                &sqlca);
    DB2_API_CHECK("database restore -- continue");
  }

  printf("\n  Restore finished.\n");

  /* drop the restored database */
  rc = DbDrop(restoredDbAlias);

  return 0;
} /* DbBackupAndRedirectedRestore */

int InaccessableContainersRedefine(char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  sqluint32 numTablespaces;
  struct SQLB_TBSPQRY_DATA **ppTablespaces;
  sqluint32 numContainers;
  struct SQLB_TBSCONTQRY_DATA *pContainers;
  int tspNb;
  int contNb;
  char pathSep[2];

  /* The API sqlbmtsq provides a one-call interface to the table space query
     data. The query data for all table spaces in the database is returned
     in an array. */
  sqlbmtsq(&sqlca,
           &numTablespaces,
           &ppTablespaces,
           SQLB_RESERVED1,
           SQLB_RESERVED2);
  DB2_API_CHECK("tablespaces -- get");

  /* refedine the inaccessible containers */
  for (tspNb = 0; tspNb < numTablespaces; tspNb++)
  {
```

```
/* The API sqlbtcq provides a one-call interface to the table space
   container query data. The query data for all the containers in a table
   space, or for all containers in all table spaces, is returned in an
   array. */
sqlbtcq(&sqlca, ppTablespaces[tspNb]->id, &numContainers, &pContainers);
DB2_API_CHECK("tablespace containers -- get");

for (contNb = 0; contNb < numContainers; contNb++)
{
  if (!pContainers[contNb].ok)
  {
    /* redefine inaccessible container */
    printf("\n    Redefine inaccessible container:\n");
    printf("      - table space name: %s\n",
           ppTablespaces[tspNb]->name);
    printf("      - default container name: %s\n",
           pContainers[contNb].name);
    if (strstr(pContainers[contNb].name, "/"))
    { /* UNIX */
      strcpy(pathSep, "/");
    }
    else
    { /* Intel */
      strcpy(pathSep, "\\");
    }
    switch (pContainers[contNb].contType)
    {
      case SQLB_CONT_PATH:
        printf("      - container type: path\n");

        sprintf(pContainers[contNb].name, "%s%sSQLT%04d.%d",
                                          serverWorkingPath, pathSep,
                                          ppTablespaces[tspNb]->id,
                                          pContainers[contNb].id);
        printf("      - new container name: %s\n",
         pContainers[contNb].name);
        break;
      case SQLB_CONT_DISK:
      case SQLB_CONT_FILE:
      default:
        printf("    Unknown container type.\n");
        break;
    }
  }
}

/* The API sqlbstsc is used to set or redefine table space containers
   while performing a 'redirected' restore of the database. */
sqlbstsc(&sqlca,
         SQLB_SET_CONT_FINAL_STATE,
         ppTablespaces[tspNb]->id,
         numContainers,
         pContainers);
DB2_API_CHECK("tablespace containers -- redefine");
```

```
    /* The API sqlefmem is used here to free memory allocated by DB2 for use
       with the API sqlbtcq (Tablespace Container Query). */
    sqlefmem(&sqlca, pContainers);
    DB2_API_CHECK("tablespace containers memory -- free");
  }

  /* The API sqlefmem is used here to free memory allocated by DB2 for
     use with the API sqlbmtsq (Tablespace Query). */
  sqlefmem(&sqlca, ppTablespaces);
  DB2_API_CHECK("tablespaces memory -- free");

  return 0;
} /* InaccessibleContainersRedefine */

int DbBackupRestoreAndRollforward(char dbAlias[],
                                  char rolledForwardDbAlias[],
                                  char user[],
                                  char pswd[],
                                  char serverWorkingPath[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct sqlfupd dbCfgParameters[1];
  unsigned short logretain;
  sqluint32 backupBufferSize;
  sqluint32 backupMode;
  sqluint32 backupType;
  sqluint32 backupCallerAction;
  char backupAppId[SQLU_APPLID_LEN + 1];
  char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
  sqluint32 backupBuffersNb;
  struct sqlu_tablespace_bkrst_list backupTablespaceList;
  struct sqlu_media_list backupTargetMediaList;
  struct sqlu_media_entry backupTargetMediaEntries[1];
  sqluint32 backupParallelismDegree;
  sqluint32 backupImageSize;
  sqluint32 restoreBufferSize;
  sqluint32 rollforwardPendingState;
  sqluint32 datalinkMode;
  sqluint32 restoreType;
  sqluint32 restoreMode;
  sqluint32 restoreCallerAction;
  char restoreAppId[SQLU_APPLID_LEN + 1];
  char restoreTimestamp[SQLU_TIME_STAMP_LEN + 1];
  char *restoreTargetPath;
  sqluint32 restoreBuffersNb;
  struct sqlu_tablespace_bkrst_list restoreTablespaceList;
  struct sqlu_media_list restoreSourceMediaList;
  struct sqlu_media_entry restoreSourceMediaEntries[1];
  sqluint32 restoreParallelismDegree;
  struct rfwd_input rfwdInput;
  struct rfwd_output rfwdOutput;
  char rollforwardAppId[SQLU_APPLID_LEN + 1];
  sqlint32 numReplies;
  struct sqlurf_info nodeInfo;
```

```
printf("\n****************************\n");
printf("*** ROLLFORWARD RECOVERY ***\n");
printf("****************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlecrea -- Create Database\n");
printf("  sqlurestore -- Restore Database\n");
printf("  sqluroll -- Rollforward Database\n");
printf("  sqledrpd -- Drop Database\n");
printf("TO BACK UP, RESTORE, AND ROLL A DATABASE FORWARD. \n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");


dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable the configuration parameter 'logretain' */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
```

```
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
                              NULL,
        0,
                              NULL,
        backupParallelismDegree,
        &backupImageSize,
                              NULL,
                              NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("\n  Continuing the backup operation...\n");

  /* depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
                              NULL,
          0,
                              NULL,
          backupParallelismDegree,
          &backupImageSize,
                              NULL,
                              NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
```

```
  printf("    - backup image time stamp: %s\n",
         backupStartTimestamp);

  /* To restore a remote database, you will first need to create an empty
     database if the client's code page is different from the server's
     code page.
     If this is the case, uncomment the call to DbCreate(). It will create
     an empty database on the server with the server's code page. */

/*
  rc = DbCreate(dbAlias, rolledForwardDbAlias);
  if (rc != 0)
  {
    return rc;
  }
*/

  /*****************************/
  /*    RESTORE THE DATABASE    */
  /*****************************/

  restoreBufferSize = 1024; /*  1024 x 4KB */
  rollforwardPendingState = SQLUD_ROLLFWD;
  datalinkMode = SQLUD_NODATALINK;
  restoreType = SQLUD_FULL;
  restoreMode = SQLUD_OFFLINE;
  strcpy(restoreTimestamp, backupStartTimestamp);
  restoreTargetPath = NULL;
  restoreBuffersNb = 1;
  restoreTablespaceList.num_entry = 0; /* number of table spaces */
  restoreTablespaceList.tablespace = NULL;
  restoreSourceMediaList.media_type = SQLU_LOCAL_MEDIA;
  restoreSourceMediaList.sessions = 1; /* number of elements in the target */
  restoreSourceMediaList.target.media = restoreSourceMediaEntries;
  strcpy(restoreSourceMediaEntries[0].media_entry, serverWorkingPath);
  restoreParallelismDegree = 1;

  printf("\n  Restoring a database ...\n");
  printf("    - source image alias     : %s\n", dbAlias);
  printf("    - source image time stamp: %s\n", restoreTimestamp);
  printf("    - target database        : %s\n", rolledForwardDbAlias);

  restoreCallerAction = SQLUD_RESTORE;

  /* The API sqlurestore is used to restore a database that has been backed
     up using the API sqlubkp. */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
```

```
            restoreTimestamp,
            restoreTargetPath,
            restoreBuffersNb,
                            NULL,
            &restoreTablespaceList,
            &restoreSourceMediaList,
            user,
            pswd,
                            NULL,
            0,
                            NULL,
            restoreParallelismDegree,
                            NULL,
                            NULL,
                            NULL,
            &sqlca);
DB2_API_CHECK("database restore -- start");

while (sqlca.sqlcode != 0)
{
  /* continue the restore operation */
  printf("\n  Continuing the restore operation...\n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  restoreCallerAction = SQLUD_CONTINUE;

  /* restore the database */
  sqlurestore(dbAlias,
              rolledForwardDbAlias,
              restoreBufferSize,
              rollforwardPendingState,
              datalinkMode,
              restoreType,
              restoreMode,
              restoreCallerAction,
              restoreAppId,
              restoreTimestamp,
              restoreTargetPath,
              restoreBuffersNb,
                            NULL,
              &restoreTablespaceList,
              &restoreSourceMediaList,
              user,
              pswd,
                            NULL,
              0,
                            NULL,
              restoreParallelismDegree,
                            NULL,
                            NULL,
                            NULL,
              &sqlca);
  DB2_API_CHECK("database restore -- continue");
```

```
  }

  printf("\n  Restore finished.\n");


  /*****************************/
  /*    ROLLFORWARD RECOVERY    */
  /*****************************/

  printf("\n  Rolling '%s' database forward ...\n", rolledForwardDbAlias);

  rfwdInput.version = SQLUM_RFWD_VERSION;
  rfwdInput.pDbAlias = rolledForwardDbAlias;
  rfwdInput.CallerAction = SQLUM_ROLLFWD_STOP;
  rfwdInput.pStopTime = SQLUM_INFINITY_TIMESTAMP;
  rfwdInput.pUserName = user;
  rfwdInput.pPassword = pswd;
  rfwdInput.pOverflowLogPath = serverWorkingPath;
  rfwdInput.NumChngLgOvrflw = 0;
  rfwdInput.pChngLogOvrflw = NULL;
  rfwdInput.ConnectMode = SQLUM_OFFLINE;
  rfwdInput.pTablespaceList = NULL;
  rfwdInput.AllNodeFlag = SQLURF_ALL_NODES;
  rfwdInput.NumNodes = 0;
  rfwdInput.pNodeList = NULL;
  rfwdInput.NumNodeInfo = 1;
  rfwdInput.DlMode = 0;
  rfwdInput.pReportFile = NULL;
  rfwdInput.pDroppedTblID = NULL;
  rfwdInput.pExportDir = NULL;

  rfwdOutput.pApplicationId = rollforwardAppId;
  rfwdOutput.pNumReplies = &numReplies;
  rfwdOutput.pNodeInfo = &nodeInfo;

  /* rollforward database */
  /* The API sqluroll rollforward recovers a database by
     applying transactions recorded in the database log files. */
  sqluroll(&rfwdInput, &rfwdOutput, &sqlca);
  DB2_API_CHECK("rollforward -- start");

  printf("  Rollforward finished.\n");

  /* drop the restored database */
  rc = DbDrop(rolledForwardDbAlias);

  return 0;
} /* DbBackupRestoreAndRollforward */

int DbLogRecordsForCurrentConnectionRead(char dbAlias[],
                                         char user[],
                                         char pswd[],
                                         char serverWorkingPath[])
{
  int rc = 0;
```

```
struct sqlca sqlca;
struct sqlfupd dbCfgParameters[1];
unsigned short logretain;
sqluint32 backupBufferSize;
sqluint32 backupMode;
sqluint32 backupType;
sqluint32 backupCallerAction;
char backupAppId[SQLU_APPLID_LEN + 1];
char backupStartTimestamp[SQLU_TIME_STAMP_LEN + 1];
sqluint32 backupBuffersNb;
struct sqlu_tablespace_bkrst_list backupTablespaceList;
struct sqlu_media_list backupTargetMediaList;
struct sqlu_media_entry backupTargetMediaEntries[1];
sqluint32 backupParallelismDegree;
sqluint32 backupImageSize;
SQLU_LSN startLSN;
SQLU_LSN endLSN;
char *logBuffer;
sqluint32 logBufferSize;
SQLU_RLOG_INFO readLogInfo;
int i;

printf("\n*****************************\n");
printf("*** ASYNCHRONOUS READ LOG ***\n");
printf("*****************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  sqlfudb -- Update Database Configuration\n");
printf("  sqlubkp -- Backup Database\n");
printf("  sqlurlog -- Asynchronous Read Log\n");
printf("AND THE SQL STATEMENTS:\n");
printf("  CONNECT\n");
printf("  ALTER TABLE\n");
printf("  COMMIT\n");
printf("  INSERT\n");
printf("  DELETE\n");
printf("  ROLLBACK\n");
printf("  CONNECT RESET\n");
printf("TO READ LOG RECORDS FOR THE CURRENT CONNECTION.\n");

printf("\n  Update \'%s\' database configuration:\n", dbAlias);
printf("    - Enable the database configuration parameter LOGRETAIN \n");
printf("        i.e., set LOGRETAIN = RECOVERY/YES\n");

dbCfgParameters[0].token = SQLF_DBTN_LOG_RETAIN;
dbCfgParameters[0].ptrvalue = (char *)&logretain;

/* enable LOGRETAIN */
logretain = SQLF_LOGRETAIN_RECOVERY;

/* The API sqlfudb is used to modify individual entries in a specific
   database configuration file. */
sqlfudb(dbAlias, 1, dbCfgParameters, &sqlca);
DB2_API_CHECK("Db Log Retain -- Enable");

/* start the backup operation */
```

```
printf("\n  Backing up the '%s' database...\n", dbAlias);

backupBufferSize = 16; /*  16 x 4KB */
backupMode = SQLUB_OFFLINE;
backupType = SQLUB_FULL;
backupBuffersNb = 1;
backupTablespaceList.num_entry = 0; /* number of table spaces */
backupTablespaceList.tablespace = NULL;
backupTargetMediaList.media_type = SQLU_LOCAL_MEDIA;
backupTargetMediaList.sessions = 1; /* number of elements in the target */
backupTargetMediaList.target.media = backupTargetMediaEntries;
strcpy(backupTargetMediaEntries[0].media_entry, serverWorkingPath);
backupParallelismDegree = 1;

backupCallerAction = SQLUB_BACKUP;

/* The API sqlubkp creates a backup copy of a database.
   This API automatically establishes a connection to the specified
    database.
   (This API can also be used to create a backup copy of a table space). */
sqlubkp(dbAlias,
        backupBufferSize,
        backupMode,
        backupType,
        backupCallerAction,
        backupAppId,
        backupStartTimestamp,
        backupBuffersNb,
        &backupTablespaceList,
        &backupTargetMediaList,
        user,
        pswd,
                            NULL,
        0,
                            NULL,
        backupParallelismDegree,
        &backupImageSize,
                            NULL,
                            NULL,
        &sqlca);
DB2_API_CHECK("Database -- Backup");

while (sqlca.sqlcode != 0)
{
  /* continue the backup operation */
  printf("\n  Continuing the backup operation...\n");

  /* Depending on the sqlca.sqlcode value, user action may be
     required, such as mounting a new tape. */

  backupCallerAction = SQLUB_CONTINUE;

  /* back up the database */
  sqlubkp(dbAlias,
          backupBufferSize,
```

```
          backupMode,
          backupType,
          backupCallerAction,
          backupAppId,
          backupStartTimestamp,
          backupBuffersNb,
          &backupTablespaceList,
          &backupTargetMediaList,
          user,
          pswd,
                           NULL,
          0,
                           NULL,
          backupParallelismDegree,
          &backupImageSize,
                           NULL,
                           NULL,
          &sqlca);
  DB2_API_CHECK("Database -- Backup");
}

printf("  Backup finished.\n");
printf("    - backup image size      : %d MB\n", backupImageSize);
printf("    - backup image path      : %s\n",
       backupTargetMediaEntries[0].media_entry);
printf("    - backup image time stamp: %s\n",
       backupStartTimestamp);

/* connect to the database */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}

/* invoke SQL statements to fill database log */
printf("\n  Invoke the following SQL statements:\n"
       "    ALTER TABLE emp_resume DATA CAPTURE CHANGES;\n"
       "    COMMIT;\n"
       "    INSERT INTO emp_resume\n"
       "      VALUES('000777', 'ascii', 'This is a new resume.');\n"
       "            ('777777', 'ascii', 'This is another new resume');\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000777';\n"
       "    DELETE FROM emp_resume WHERE empno = '777777';\n"
       "    COMMIT;\n"
       "    DELETE FROM emp_resume WHERE empno = '000140';\n"
       "    ROLLBACK;\n"
       "    ALTER TABLE emp_resume DATA CAPTURE NONE;\n"
       "    COMMIT;\n");

EXEC SQL ALTER TABLE emp_resume DATA CAPTURE CHANGES;
EMB_SQL_CHECK("SQL statement 1 -- invoke");

EXEC SQL COMMIT;
```

```
  EMB_SQL_CHECK("SQL statement 2 -- invoke");

  EXEC SQL INSERT INTO emp_resume
    VALUES('000777', 'ascii', 'This is a new resume.'),
          ('777777', 'ascii', 'This is another new resume');
  EMB_SQL_CHECK("SQL statement 3 -- invoke");

  EXEC SQL COMMIT;
  EMB_SQL_CHECK("SQL statement 4 -- invoke");

  EXEC SQL DELETE FROM emp_resume WHERE empno = '000777';
  EMB_SQL_CHECK("SQL statement 5 -- invoke");

  EXEC SQL DELETE FROM emp_resume WHERE empno = '777777';
  EMB_SQL_CHECK("SQL statement 6 -- invoke");

  EXEC SQL COMMIT;
  EMB_SQL_CHECK("SQL statement 7 -- invoke");

  EXEC SQL DELETE FROM emp_resume WHERE empno = '000140';
  EMB_SQL_CHECK("SQL statement 8 -- invoke");

  EXEC SQL ROLLBACK;
  EMB_SQL_CHECK("SQL statement 9 -- invoke");

  EXEC SQL ALTER TABLE emp_resume DATA CAPTURE NONE;
  EMB_SQL_CHECK("SQL statement 10 -- invoke");

  EXEC SQL COMMIT;
  EMB_SQL_CHECK("SQL statement 11 -- invoke");

  printf("\n  Start reading database log.\n");

  logBuffer = NULL;
  logBufferSize = 0;

  /* The API sqlurlog (Asynchronours Read Log) is used to extract records
     from the database logs, and to query the log manager for current
     log state information.
     This API can only be used on recoverable databases. */

  /* Query the log manager for current log state information: */
  rc = sqlurlog(SQLU_RLOG_QUERY,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &sqlca);
/*  DB2_API_CHECK("database log info -- get"); */
  EXPECTED_WARN_CHECK("database log info -- get");

  logBufferSize = 64 * 1024;
  logBuffer = (char *)malloc(logBufferSize);
```

```
memcpy(&startLSN, &(readLogInfo.initialLSN), sizeof(startLSN));
memcpy(&endLSN, &(readLogInfo.curActiveLSN), sizeof(endLSN));

/* Extract a log record from the database logs, and
   read the first log sequence asynchronously: */
rc = sqlurlog(SQLU_RLOG_READ,
              &startLSN,
              &endLSN,
              logBuffer,
              logBufferSize,
              &readLogInfo,
              &sqlca);
if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  DB2_API_CHECK("database logs -- read");
}
else
{
  if (readLogInfo.logRecsWritten == 0)
  {
    printf("\n  Database log empty.\n");
  }
}

/* display log buffer */
rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);

while (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
{
  /* read the next log sequence */

  memcpy(&startLSN, &(readLogInfo.lastReadLSN), sizeof(startLSN));
  /* increase startLSN by 1 */
  startLSN.lsnChar[5] = startLSN.lsnChar[5] + 1;
  i = 5;
  while (startLSN.lsnChar[i] == 0 && i > 0)
  {
    startLSN.lsnChar[i - 1] = startLSN.lsnChar[i - 1] + 1;
    i = i - 1;
  }

  /* Extract a log record from the database logs, and
   read the next log sequence asynchronously: */
  rc = sqlurlog(SQLU_RLOG_READ,
                &startLSN,
                &endLSN,
                logBuffer,
                logBufferSize,
                &readLogInfo,
                &sqlca);
  if (sqlca.sqlcode != SQLU_RLOG_READ_TO_CURRENT)
  {
    DB2_API_CHECK("database logs -- read");
  }
```

```
    /* display log buffer */
    rc = LogBufferDisplay(logBuffer, readLogInfo.logRecsWritten);
  }

  /* free the log buffer */
  free(logBuffer);

  /* disconnect from the database */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbLogRecordsForCurrentConnectionRead */

int LogBufferDisplay(char *logBuffer, sqluint32 numLogRecords)
{
  int rc = 0;
  sqluint32 logRecordNb;
  sqluint32 recordSize;
  sqluint16 recordType;
  sqluint16 recordFlag;
  char *recordBuffer;

  /* initialize recordBuffer */
  recordBuffer = logBuffer + sizeof(SQLU_LSN);

  for (logRecordNb = 0; logRecordNb < numLogRecords; logRecordNb++)
  {
    recordSize = *(sqluint32 *)(recordBuffer);
    recordType = *(sqluint16 *)(recordBuffer + 4);
    recordFlag = *(sqluint16 *)(recordBuffer + 6);

    rc = LogRecordDisplay(recordBuffer, recordSize, recordType, recordFlag);
    /* update recordBuffer */
    recordBuffer = recordBuffer + recordSize + sizeof(SQLU_LSN);
  }

  return 0;
} /* LogBufferDisplay */

int LogRecordDisplay(char *recordBuffer,
                     sqluint32 recordSize,
                     sqluint16 recordType,
                     sqluint16 recordFlag)
{
  int rc = 0;
  sqluint32 logManagerLogRecordHeaderSize;
  char *recordDataBuffer;
  sqluint32 recordDataSize;
  char *recordHeaderBuffer;
  sqluint8 componentIdentifier;
  sqluint32 recordHeaderSize;
```

```
/* determine logManagerLogRecordHeaderSize */
if ((char)recordType == 'C')
{ /* compensation */
  if (recordFlag & 0x0002)
  { /* propagatable */
    logManagerLogRecordHeaderSize = 32;
  }
  else
  {
    logManagerLogRecordHeaderSize = 26;
  }
}
else
{ /* non compensation */
  logManagerLogRecordHeaderSize = 20;
}

switch ((char)recordType)
{
  case 'E':
  case 'M':
  case 'A':
    recordDataBuffer = recordBuffer + logManagerLogRecordHeaderSize;
    recordDataSize = recordSize - logManagerLogRecordHeaderSize;
    rc = SimpleLogRecordDisplay(recordType,
                                recordFlag,
                                recordDataBuffer,
                                recordDataSize);
    break;
  case 'N':
  case 'C':
    recordHeaderBuffer = recordBuffer + logManagerLogRecordHeaderSize;
    componentIdentifier = *(sqluint8 *)recordHeaderBuffer;
    switch (componentIdentifier)
    {
      case 1:
        recordHeaderSize = 6;
        break;
      default:
        printf("    Unknown complex log record: %lu %c %u\n",
               recordSize, recordType, componentIdentifier);
        return 1;
    }
    recordDataBuffer = recordBuffer +
                       logManagerLogRecordHeaderSize +
                       recordHeaderSize;
    recordDataSize = recordSize -
                     logManagerLogRecordHeaderSize -
                     recordHeaderSize;
    rc = ComplexLogRecordDisplay(recordType,
                                 recordFlag,
                                 recordHeaderBuffer,
                                 recordHeaderSize,
                                 componentIdentifier,
```

```
                                    recordDataBuffer,
                                    recordDataSize);
      break;
    default:
      printf("    Unknown log record: %lu %c\n",
             recordSize, (char)recordType);
      break;
  }

  return 0;
} /* LogRecordDisplay */

int SimpleLogRecordDisplay(sqluint16 recordType,
                           sqluint16 recordFlag,
                           char *recordDataBuffer,
                           sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint32 timeTransactionCommited;
  sqluint16 authIdLen;
  char authId[129];

  switch ((char)recordType)
  {
    case 'E':
      printf("\n    Record type: Local pending list\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = *(sqluint16 *)(recordDataBuffer + 4);
      memcpy(authId, (char *)(recordDataBuffer + 6), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'M':
      printf("\n    Record type: Normal commit\n");
      timeTransactionCommited = *(sqluint32 *)(recordDataBuffer);
      authIdLen = (sqluint16) (recordDataSize - 4);
      memcpy(authId, (char *)(recordDataBuffer + 4), authIdLen);
      authId[authIdLen] = '\0';
      printf("      %s: %lu\n",
             "UTC transaction committed (in seconds since 01/01/70)",
             timeTransactionCommited);
      printf("      authorization ID of the application: %s\n", authId);
      break;
    case 'A':
      printf("\n    Record type: Normal abort\n");
      authIdLen = (sqluint16) (recordDataSize);
      memcpy(authId, (char *)(recordDataBuffer), authIdLen);
      authId[authIdLen] = '\0';
      printf("      authorization ID of the application: %s\n", authId);
      break;
    default:
      printf("    Unknown simple log record: %c %lu\n",
```

```
            (char)recordType, recordDataSize);
      break;
  }

  return 0;
} /* SimpleLogRecordDisplay */

int ComplexLogRecordDisplay(sqluint16 recordType,
                            sqluint16 recordFlag,
                            char *recordHeaderBuffer,
                            sqluint32 recordHeaderSize,
                            sqluint8 componentIdentifier,
                            char *recordDataBuffer,
                            sqluint32 recordDataSize)
{
  int rc = 0;
  sqluint8 functionIdentifier;
  /* for insert, delete, undo delete */
  sqluint32 RID;
  sqluint16 subRecordLen;
  sqluint16 subRecordOffset;
  char *subRecordBuffer;
  /* for update */
  sqluint32 newRID;
  sqluint16 newSubRecordLen;
  sqluint16 newSubRecordOffset;
  char *newSubRecordBuffer;
  sqluint32 oldRID;
  sqluint16 oldSubRecordLen;
  sqluint16 oldSubRecordOffset;
  char *oldSubRecordBuffer;
  /* for alter table attributes */
  sqluint32 alterBitMask;
  sqluint32 alterBitValues;

  switch ((char)recordType)
  {
    case 'N':
      printf("\n    Record type: Normal\n");
      break;
    case 'C':
      printf("\n    Record type: Compensation\n");
      break;
    default:
      printf("\n    Unknown complex log record type: %c\n", recordType);
      break;
  }

  switch (componentIdentifier)
  {
    case 1:
      printf("      component ID: DMS log record\n");
      break;
    default:
      printf("      unknown component ID: %d\n", componentIdentifier);
```

```
    break;
}

functionIdentifier = *(sqluint8 *)(recordHeaderBuffer + 1);
switch (functionIdentifier)
{
  case 106:
    printf("      function ID: Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 111:
    printf("      function ID: Undo Delete Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 118:
    printf("      function ID: Insert Record\n");
    RID = *(sqluint32 *)(recordDataBuffer + 2);
    subRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    subRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    printf("        RID: %lu\n", RID);
    printf("        subrecord length: %u\n", subRecordLen);
    printf("        subrecord offset: %u\n", subRecordOffset);
    subRecordBuffer = recordDataBuffer + 12;
    rc = LogSubRecordDisplay(subRecordBuffer, subRecordLen);
    break;
  case 120:
    printf("      function ID: Update Record\n");
    oldRID = *(sqluint32 *)(recordDataBuffer + 2);
    oldSubRecordLen = *(sqluint16 *)(recordDataBuffer + 6);
    oldSubRecordOffset = *(sqluint16 *)(recordDataBuffer + 10);
    newRID = *(sqluint32 *)(recordDataBuffer +
                            12 + oldSubRecordLen + recordHeaderSize + 2);
    newSubRecordLen = *(sqluint16 *)(recordDataBuffer +
                                     12 + oldSubRecordLen +
                                     recordHeaderSize + 6);
    newSubRecordOffset =
      *(sqluint16 *)(recordDataBuffer + 12 + oldSubRecordLen +
                     recordHeaderSize + 10);
    printf("        oldRID: %lu\n", oldRID);
    printf("        old subrecord length: %u\n", oldSubRecordLen);
    printf("        old subrecord offset: %u\n",
```

```
        oldSubRecordOffset);
  oldSubRecordBuffer = recordDataBuffer + 12;
  rc = LogSubRecordDisplay(oldSubRecordBuffer, oldSubRecordLen);
  printf("        newRID: %lu\n", newRID);
  printf("        new subrecord length: %u\n", newSubRecordLen);
  printf("        new subrecord offset: %u\n",
        newSubRecordOffset);
  newSubRecordBuffer = recordDataBuffer +
    12 + oldSubRecordLen + recordHeaderSize + 12;
  rc = LogSubRecordDisplay(newSubRecordBuffer, newSubRecordLen);
  break;
case 124:
  printf("      function ID:  Alter Table Attribute\n");
  alterBitMask = *(sqluint32 *)(recordDataBuffer + 2);
  alterBitValues = *(sqluint32 *)(recordDataBuffer + 6);
  if (alterBitMask & 0x00000001)
  {
    /* Alter the value of the 'propagation' attribute: */
    printf("        Propagation attribute is changed to: ");
    if (alterBitValues & 0x00000001)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00000002)
  {
    /* Alter the value of the 'pending' attribute: */
    printf("        Pending attribute is changed to: ");
    if (alterBitValues & 0x00000002)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00010000)
  {
    /* Alter the value of the 'append mode' attribute: */
    printf("        Append Mode attribute is changed to: ");
    if (alterBitValues & 0x00010000)
    {
      printf("ON\n");
    }
    else
    {
      printf("OFF\n");
    }
  }
  if (alterBitMask & 0x00200000)
```

```
      {
        /* Alter the value of the 'LF Propagation' attribute: */
        printf("        LF Propagation attribute is changed to: ");
        if (alterBitValues & 0x00200000)
        {
          printf("ON\n");
        }
        else
        {
          printf("OFF\n");
        }
      }
      if (alterBitMask & 0x00400000)
      {
        /* Alter the value of the 'LOB Propagation' attribute: */
        printf("        LOB Propagation attribute is changed to: ");
        if (alterBitValues & 0x00400000)
        {
          printf("ON\n");
        }
        else
        {
          printf("OFF\n");
        }
      }
      break;
    default:
      printf("      unknown function identifier: %u\n",
             functionIdentifier);
      break;
  }

  return 0;
} /* ComplexLogRecordDisplay */

int LogSubRecordDisplay(char *recordBuffer, sqluint16 recordSize)
{
  int rc = 0;
  sqluint8 recordType;
  sqluint8 updatableRecordType;
  sqluint16 userDataFixedLength;
  char *userDataBuffer;
  sqluint16 userDataSize;

  recordType = *(sqluint8 *)(recordBuffer);
  if (recordType != 0 && recordType != 4)
  {
    printf("        Unknown subrecord type.\n");
  }
  else if (recordType == 4)
  {
    printf("        subrecord type: Special control\n");
  }
  else
  {
```

```
    /* recordType == 0 */
    printf("        subrecord type: Updatable, ");
    updatableRecordType = *(sqluint8 *)(recordBuffer + 4);
    if (updatableRecordType != 1)
    {
      printf("Internal control\n");
    }
    else
    {
      printf("Formatted user data\n");
      userDataFixedLength = *(sqluint16 *)(recordBuffer + 6);
      printf("        user data fixed length: %u\n",
             userDataFixedLength);
      userDataBuffer = recordBuffer + 8;
      userDataSize = recordSize - 8;
      rc = UserDataDisplay(userDataBuffer, userDataSize);
    }
  }

  return 0;
} /* LogSubRecordDisplay */

int UserDataDisplay(char *dataBuffer, sqluint16 dataSize)
{
  int rc = 0;

  sqluint16 line, col;

  printf("        user data:\n");

  for (line = 0; line * 10 < dataSize; line = line + 1)
  {
    printf("        ");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        printf("%02X ", dataBuffer[line * 10 + col]);
      }
      else
      {
        printf("   ");
      }
    }
    printf("*");
    for (col = 0; col < 10; col = col + 1)
    {
      if (line * 10 + col < dataSize)
      {
        if (isalpha(dataBuffer[line * 10 + col]) ||
            isdigit(dataBuffer[line * 10 + col]))
        {
          printf("%c", dataBuffer[line * 10 + col]);
        }
        else
```

```
        {
          printf(".");
        }
      }
      else
      {
        printf(" ");
      }
    }
    printf("*");
    printf("\n");
  }

  return 0;
} /* UserDataDisplay */

int DbRecoveryHistoryFileRead(char dbAlias[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint32 numEntries;
  sqluint16 recoveryHistoryFileHandle;
  sqluint32 entryNb;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;

  printf("\n*******************************************\n");
  printf("*** READ A DATABASE RECOVERY HISTORY FILE ***\n");
  printf("*******************************************\n");
  printf("\nUSE THE DB2 APIs:\n");
  printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
  printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
  printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
  printf("TO READ A DATABASE RECOVERY HISTORY FILE.\n");

  /* initialize the data structures */
  dbHistoryOpenParam.piDatabaseAlias = dbAlias;
  dbHistoryOpenParam.piTimestamp = NULL;
  dbHistoryOpenParam.piObjectName = NULL;
  dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;

  dbHistoryEntryGetParam.pioHistData = &histEntryData;
  dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
  rc = HistoryEntryDataFieldsAlloc(&histEntryData);
  if (rc != 0)
  {
    return rc;
  }

  /*****************************************/
  /* OPEN THE DATABASE RECOVERY HISTORY FILE */
  /*****************************************/
  printf("\n  Open recovery history file for '%s' database.\n", dbAlias);
```

```
  /* open the recovery history file to scan */
  db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
  DB2_API_CHECK("database recovery history file -- open");

  numEntries = dbHistoryOpenParam.oNumRows;

  /* dbHistoryOpenParam.oHandle returns the handle for scan access */
  recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
  dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

  /*********************************************/
  /* READ AN ENTRY IN THE RECOVERY HISTORY FILE */
  /*********************************************/
  for (entryNb = 0; entryNb < numEntries; entryNb = entryNb + 1)
  {
    printf("\n  Read entry number %u.\n", entryNb);

    /* get the next entry from the recovery history file */
    db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
    DB2_API_CHECK("database recovery history file entry -- read")

    /* display the entries in the recovery history file */
    printf("\n  Display entry number %u.\n", entryNb);
    rc = HistoryEntryDisplay(histEntryData);
  }

  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

  /* The API db2HistoryCloseScan ends the recovery history file scan and
     frees DB2 resources required for the scan. */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbRecoveryHistoryFileRead */

int HistoryEntryDataFieldsAlloc(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  strcpy(pHistEntryData->ioHistDataID, "SQLUHINF");

  pHistEntryData->oObjectPart.pioData = malloc(17 + 1);
  pHistEntryData->oObjectPart.iLength = 17 + 1;

  pHistEntryData->oEndTime.pioData = malloc(12 + 1);
  pHistEntryData->oEndTime.iLength = 12 + 1;
```

```
  pHistEntryData->oFirstLog.pioData = malloc(8 + 1);
  pHistEntryData->oFirstLog.iLength = 8 + 1;

  pHistEntryData->oLastLog.pioData = malloc(8 + 1);
  pHistEntryData->oLastLog.iLength = 8 + 1;

  pHistEntryData->oID.pioData = malloc(128 + 1);
  pHistEntryData->oID.iLength = 128 + 1;

  pHistEntryData->oTableQualifier.pioData = malloc(128 + 1);
  pHistEntryData->oTableQualifier.iLength = 128 + 1;

  pHistEntryData->oTableName.pioData = malloc(128 + 1);
  pHistEntryData->oTableName.iLength = 128 + 1;

  pHistEntryData->oLocation.pioData = malloc(128 + 1);
  pHistEntryData->oLocation.iLength = 128 + 1;

  pHistEntryData->oComment.pioData = malloc(128 + 1);
  pHistEntryData->oComment.iLength = 128 + 1;

  pHistEntryData->oCommandText.pioData = malloc(128 + 1);
  pHistEntryData->oCommandText.iLength = 128 + 1;

  pHistEntryData->poEventSQLCA =
    (struct sqlca *)malloc(sizeof(struct sqlca));

  pHistEntryData->poTablespace = (db2Char *)malloc(3 * sizeof(db2Char));
  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    pHistEntryData->poTablespace[tsNb].pioData = malloc(18 + 1);
    pHistEntryData->poTablespace[tsNb].iLength = 18 + 1;
  }

  pHistEntryData->iNumTablespaces = 3;

  return 0;
} /* HistoryEntryDataFieldsAlloc */

int HistoryEntryDisplay(struct db2HistoryData histEntryData)
{
  int rc = 0;
  char buf[129];
  sqluint32 tsNb;

  memcpy(buf, histEntryData.oObjectPart.pioData,
              histEntryData.oObjectPart.oLength);
  buf[histEntryData.oObjectPart.oLength] = '\0';
  printf("    object part: %s\n", buf);

  memcpy(buf, histEntryData.oEndTime.pioData,
              histEntryData.oEndTime.oLength);
  buf[histEntryData.oEndTime.oLength] = '\0';
  printf("    end time: %s\n", buf);
```

```
memcpy(buf, histEntryData.oFirstLog.pioData,
            histEntryData.oFirstLog.oLength);
buf[histEntryData.oFirstLog.oLength] = '\0';
printf("    first log: %s\n", buf);

memcpy(buf, histEntryData.oLastLog.pioData,
            histEntryData.oLastLog.oLength);
buf[histEntryData.oLastLog.oLength] = '\0';
printf("    last log: %s\n", buf);

memcpy(buf, histEntryData.oID.pioData, histEntryData.oID.oLength);
buf[histEntryData.oID.oLength] = '\0';
printf("    ID: %s\n", buf);

memcpy(buf, histEntryData.oTableQualifier.pioData,
            histEntryData.oTableQualifier.oLength);
buf[histEntryData.oTableQualifier.oLength] = '\0';
printf("    table qualifier: %s\n", buf);

memcpy(buf, histEntryData.oTableName.pioData,
            histEntryData.oTableName.oLength);
buf[histEntryData.oTableName.oLength] = '\0';
printf("    table name: %s\n", buf);

memcpy(buf, histEntryData.oLocation.pioData,
            histEntryData.oLocation.oLength);
buf[histEntryData.oLocation.oLength] = '\0';
printf("    location: %s\n", buf);

memcpy(buf, histEntryData.oComment.pioData,
            histEntryData.oComment.oLength);
buf[histEntryData.oComment.oLength] = '\0';
printf("    comment: %s\n", buf);

memcpy(buf, histEntryData.oCommandText.pioData,
            histEntryData.oCommandText.oLength);
buf[histEntryData.oCommandText.oLength] = '\0';
printf("    command text: %s\n", buf);
printf("    history file entry ID: %u\n", histEntryData.oEID.ioHID);
printf("    table spaces:\n");

for (tsNb = 0; tsNb < histEntryData.oNumTablespaces; tsNb = tsNb + 1)
{
  memcpy(buf, histEntryData.poTablespace[tsNb].pioData,
              histEntryData.poTablespace[tsNb].oLength);
  buf[histEntryData.poTablespace[tsNb].oLength] = '\0';
  printf("      %s\n", buf);
}

printf("    type of operation: %c\n", histEntryData.oOperation);
printf("    granularity of the operation: %c\n", histEntryData.oObject);
printf("    operation type: %c\n", histEntryData.oOptype);
printf("    entry status: %c\n", histEntryData.oStatus);
printf("    device type: %c\n", histEntryData.oDeviceType);
printf("    SQLCA:\n");
```

```
  printf("      sqlcode: %ld\n", histEntryData.poEventSQLCA->sqlcode);
  memcpy(buf, histEntryData.poEventSQLCA->sqlstate, 5);
  buf[5] = '\0';
  printf("      sqlstate: %s\n", buf);
  memcpy(buf, histEntryData.poEventSQLCA->sqlerrmc,
              histEntryData.poEventSQLCA->sqlerrml);
  buf[histEntryData.poEventSQLCA->sqlerrml] = '\0';
  printf("      message: %s\n", buf);

  return 0;
} /* HistoryEntryDisplay */

int HistoryEntryDataFieldsFree(struct db2HistoryData *pHistEntryData)
{
  int rc = 0;
  sqluint32 tsNb;

  free(pHistEntryData->oObjectPart.pioData);
  free(pHistEntryData->oEndTime.pioData);
  free(pHistEntryData->oFirstLog.pioData);
  free(pHistEntryData->oLastLog.pioData);
  free(pHistEntryData->oID.pioData);
  free(pHistEntryData->oTableQualifier.pioData);
  free(pHistEntryData->oTableName.pioData);
  free(pHistEntryData->oLocation.pioData);
  free(pHistEntryData->oComment.pioData);
  free(pHistEntryData->oCommandText.pioData);
  free(pHistEntryData->poEventSQLCA);

  for (tsNb = 0; tsNb < 3; tsNb = tsNb + 1)
  {
    free(pHistEntryData->poTablespace[tsNb].pioData);
  }

  free(pHistEntryData->poTablespace);

  return 0;
} /* HistoryEntryDataFieldsFree */

int DbFirstRecoveryHistoryFileEntryUpdate(char dbAlias[],
                                          char user[],
                                          char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2HistoryOpenStruct dbHistoryOpenParam;
  sqluint16 recoveryHistoryFileHandle;
  struct db2HistoryGetEntryStruct dbHistoryEntryGetParam;
  struct db2HistoryData histEntryData;
  char newLocation[DB2HISTORY_LOCATION_SZ + 1];
  char newComment[DB2HISTORY_COMMENT_SZ + 1];
  struct db2HistoryUpdateStruct dbHistoryUpdateParam;

  printf("\n************************************************\n");
  printf("*** UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY ***\n");
```

```
printf("*****************************************************\n");
printf("\nUSE THE DB2 APIs:\n");
printf("  db2HistoryOpenScan -- Open Recovery History File Scan\n");
printf("  db2HistoryGetEntry -- Get Next Recovery History File Entry\n");
printf("  db2HistoryUpdate -- Update Recovery History File\n");
printf("  db2HistoryCloseScan -- Close Recovery History File Scan\n");
printf("TO UPDATE A DATABASE RECOVERY HISTORY FILE ENTRY.\n");

/* initialize data structures */
dbHistoryOpenParam.piDatabaseAlias = dbAlias;
dbHistoryOpenParam.piTimestamp = NULL;
dbHistoryOpenParam.piObjectName = NULL;
dbHistoryOpenParam.iCallerAction = DB2HISTORY_LIST_HISTORY;
dbHistoryEntryGetParam.pioHistData = &histEntryData;
dbHistoryEntryGetParam.iCallerAction = DB2HISTORY_GET_ALL;
rc = HistoryEntryDataFieldsAlloc(&histEntryData);
if (rc != 0)
{
  return rc;
}

/*****************************************/
/* OPEN THE DATABASE RECOVERY HISTORY FILE */
/*****************************************/
printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);

/* The API db2HistoryOpenScan starts a recovery history file scan */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

/* dbHistoryOpenParam.oHandle returns the handle for scan access */
recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;
dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;

/*****************************************************/
/* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE */
/*****************************************************/
printf("\n  Read the first entry in the recovery history file.\n");

/* The API db2HistoryGetEntry gets the next entry from the recovery
   history file. */
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
DB2_API_CHECK("first recovery history file entry -- read");
printf("\n  Display the first entry.\n");

/* HistoryEntryDisplay is a support function used to display the entries
   in the recovery history file. */
rc = HistoryEntryDisplay(histEntryData);

/* update the first history file entry */
rc = DbConn(dbAlias, user, pswd);
if (rc != 0)
{
  return rc;
}
```

```
strcpy(newLocation, "this is the NEW LOCATION");
strcpy(newComment, "this is the NEW COMMENT");
printf("\n  Update the first entry in the history file:\n");
printf("    new location = '%s'\n", newLocation);
printf("    new comment = '%s'\n", newComment);
dbHistoryUpdateParam.piNewLocation = newLocation;
dbHistoryUpdateParam.piNewDeviceType = NULL;
dbHistoryUpdateParam.piNewComment = newComment;
dbHistoryUpdateParam.iEID.ioNode = histEntryData.oEID.ioNode;
dbHistoryUpdateParam.iEID.ioHID = histEntryData.oEID.ioHID;

/* The API db2HistoryUpdate can be used to update the location,
   device type, or comment in a history file entry. */

/* Call this API to update the location and comment of the first
   entry in the history file: */
db2HistoryUpdate(db2Version710, &dbHistoryUpdateParam, &sqlca);
DB2_API_CHECK("first history file entry -- update");

rc = DbDisconn(dbAlias);
if (rc != 0)
{
  return rc;
}

/*******************************************/
/* CLOSE THE DATABASE RECOVERY HISTORY FILE */
/*******************************************/
printf("\n  Close recovery history file for '%s' database.\n", dbAlias);

/* The API db2HistoryCloseScan ends the recovery history file scan and
   frees DB2 resources required for the scan. */
db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
DB2_API_CHECK("database recovery history file -- close");


/*******************************************/
/* RE-OPEN THE DATABASE RECOVERY HISTORY FILE */
/*******************************************
printf("\n  Open the recovery history file for '%s' database.\n", dbAlias);

/* starts a recovery history file scan */
db2HistoryOpenScan(db2Version710, &dbHistoryOpenParam, &sqlca);
DB2_API_CHECK("database recovery history file -- open");

recoveryHistoryFileHandle = dbHistoryOpenParam.oHandle;

dbHistoryEntryGetParam.iHandle = recoveryHistoryFileHandle;
printf("\n  Read the first recovery history file entry.\n");

/*************************************************************************/
/* READ THE FIRST ENTRY IN THE RECOVERY HISTORY FILE AFTER MODIFICATION */
/*************************************************************************/
db2HistoryGetEntry(db2Version710, &dbHistoryEntryGetParam, &sqlca);
```

```
  DB2_API_CHECK("first recovery history file entry -- read");

  printf("\n  Display the first entry.\n");
  rc = HistoryEntryDisplay(histEntryData);

  /*******************************************/
  /* CLOSE THE DATABASE RECOVERY HISTORY FILE */
  /*******************************************/
  printf("\n  Close the recovery history file for '%s' database.\n",
         dbAlias);

  /* ends the recovery history file scan */
  db2HistoryCloseScan(db2Version710, &recoveryHistoryFileHandle, &sqlca);
  DB2_API_CHECK("database recovery history file -- close");

  /* free the allocated memory */
  rc = HistoryEntryDataFieldsFree(&histEntryData);

  return 0;
} /* DbFirstRecoveryHistoryFileEntryUpdate */

int DbRecoveryHistoryFilePrune(char dbAlias[], char user[], char pswd[])
{
  int rc = 0;
  struct sqlca sqlca;
  struct db2PruneStruct histPruneParam;
  char timeStampPart[14 + 1];

  printf("\n*************************************\n");
  printf("*** PRUNE THE RECOVERY HISTORY FILE ***\n");
  printf("*************************************\n");
  printf("\nUSE THE DB2 API:\n");
  printf("  db2Prune -- Prune Recovery History File\n");
  printf("AND THE SQL STATEMENTS:\n");
  printf("  CONNECT\n");
  printf("  CONNECT RESET\n");
  printf("TO PRUNE THE RECOVERY HISTORY FILE.\n");

  /* Connect to the database: */
  rc = DbConn(dbAlias, user, pswd);
  if (rc != 0)
  {
    return rc;
  }

  /* Prune the recovery history file: */
  printf("\n  Prune the recovery history file for '%s' database.\n",
         dbAlias);

  /* timeStampPart is a pointer to a string specifying a time stamp or
     log sequence number. Time stamp is used here to select records for
     deletion. All entries equal to or less than the time stamp will be
     deleted. */
  histPruneParam.piString = timeStampPart;
  strcpy(timeStampPart, "2010"); /* year 2010 */
```

```
  /* The action DB2PRUNE_ACTION_HISTORY removes history file entries: */
  histPruneParam.iAction = DB2PRUNE_ACTION_HISTORY;

  /* The option DB2PRUNE_OPTION_FORCE forces the removal of the last backup: */
  histPruneParam.iOptions = DB2PRUNE_OPTION_FORCE;

  /* db2Prune can be called to delete entries from the recovery history file
     or log files from the active log path. Here we call it to delete
     entries from the recovery history file.
     You must have SYSADM, SYSCTRL, SYSMAINT, or DBADM authority to prune
     the recovery history file. */
  db2Prune(db2Version710, &histPruneParam, &sqlca);
  DB2_API_CHECK("recovery history file -- prune");

  /* Disconnect from the database: */
  rc = DbDisconn(dbAlias);
  if (rc != 0)
  {
    return rc;
  }

  return 0;
} /* DbRecoveryHistoryFilePrune */
```

附录F. 恢复 CLP 脚本

以下 DB2 命令显示了如何使用 CLP 命令来:

- 备份数据库
- 复原数据库
- 前滚恢复数据库

确保 SAMPLE 数据库存在，且没有正在使用。关于 SAMPLE 数据库的详细信息，请参阅 *SQL Reference*。关于 DB2 命令行处理器的通用信息，请参阅 *Command Reference*。

对脚本的与 Windows 兼容的版本以及与 UNIX 兼容的版本都进行了描述。

用于 Windows 操作系统的样本命令脚本

要在 Windows NT 上或 Windows 2000 上运行脚本:

1. 将脚本保存为一个文件，例如文件名为 `backrest.db2`。
2. 如果数据库管理器不是正在运行，可从 DB2 命令窗口发出 **db2start** 命令。要打开启用了 CLP 的 DB2 窗口，并初始化操作系统上的 DB2 命令行环境，可从命令提示符发出 **db2cmd**。
3. 输入 `db2 -f backrest.db2 -t`。

以下是此脚本返回的输出的示例:

```
D:\>db2 -f backrest.db2 -t
This is CLP script: backrest.db2


Deleting old SAMPLE database backup images...

process SAMPLE.0\DB2\NODE0000\CATN0000
process 20010403

Updating the database configuration parameter LOGRETAIN to 'ON'...

DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.


Backing up the SAMPLE database...
```

```
Backup successful. The timestamp for this backup image is : 20010403131027


Restoring the SAMPLE database as TESTBACK (1st pass)...

SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.

Listing the table spaces for the TESTBACK database...


           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined


Defining new table space containers for Tablespace 2...

DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

Listing table space containers for Tablespace 2 (TESTBACK database)...


            Tablespace Containers for Tablespace 2
```

```
 Container ID                         = 0
 Name                                 = c:\ts2con1
 Type                                 = Path


Restoring the SAMPLE database as TESTBACK (2nd pass)...

DB20000I  The RESTORE DATABASE command completed successfully.

Rolling the TESTBACK database forward...


                                Rollforward Status

 Input database alias                 = testback
 Number of nodes have returned status = 1

 Node number                          = 0
 Rollforward status                   = not pending
 Next log file to be read             =
 Log files processed                  =  -
 Last committed transaction           = 2001-04-03-03.16.07.000000

DB20000I  The ROLLFORWARD command completed successfully.

Dropping the TESTBACK database...

DB20000I  The DROP DATABASE command completed successfully.

Terminating the command line processor's back-end process...

DB20000I  The TERMINATE command completed successfully.


D:\>
```

以下是该脚本的源列表:

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--              from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.

!ECHO This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;
```

```
!ECHO Deleting old SAMPLE database backup images...;
!rd! SAMPLE.0\DB2\NODE0000\CATN0000;

!ECHO Updating the database configuration parameter LOGRETAIN to 'ON'...;
update db cfg for sample using logretain on;

!ECHO Backing up the SAMPLE database...;
backup db sample;

!ECHO Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

!ECHO Listing the table spaces for the TESTBACK database...;
list tablespaces;

!ECHO Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "c:\ts2con1");

!ECHO Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

!ECHO Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

!ECHO Rolling the TESTBACK database forward...;
rollforward db testback stop;

!ECHO Dropping the TESTBACK database...;
drop db testback;

!ECHO Terminating the command line processor's back-end process...;
   terminate;

-- End file
```

用于基于 UNIX 的系统的样本命令脚本

要在基于 UNIX 的操作系统上运行脚本:

1. 将脚本保存为一个文件，例如文件名为 backrest.db2。
2. 如果数据库管理器不是正在运行，可从命令提示符发出 **db2start** 命令。
3. 输入 db2 -f backrest.db2 -t。

以下是此脚本返回的输出的示例:

```
sunfish /export/home2/falexand/samples/clp>db2 -f backrest.db2 -t
This is CLP script: backrest.db2

Deleting old SAMPLE database backup images...

Updating the database configuration parameter LOGRETAIN to 'ON'...
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

```
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.


Backing up the SAMPLE database...

Backup successful. The timestamp for this backup image is : 20010531172525


Restoring the SAMPLE database as TESTBACK (1st pass)...
SQL1277N  Restore has detected that one or more table space containers are
inaccessible, or has set their state to 'storage must be defined'.
DB20000I  The RESTORE DATABASE command completed successfully.

Listing the table spaces for the TESTBACK database...

           Tablespaces for Current Database

 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = System Temporary data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined

 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x2001100
   Detailed explanation:
     Restore pending
     Storage must be defined
     Storage may be defined


Defining new table space containers for Tablespace 2...
DB20000I  The SET TABLESPACE CONTAINERS command completed successfully.

Listing table space containers for Tablespace 2 (TESTBACK database)...

            Tablespace Containers for Tablespace 2
```

```
 Container ID                         = 0
 Name                                 = /export/home2/falexand/falexand...
                                          .../NODE0000/SQL00020/ts2con1
 Type                                 = Path


Restoring the SAMPLE database as TESTBACK (2nd pass)...
DB20000I  The RESTORE DATABASE command completed successfully.

Rolling the TESTBACK database forward...

                               Rollforward Status

 Input database alias                 = testback
 Number of nodes have returned status = 1

 Node number                          = 0
 Rollforward status                   = not pending
 Next log file to be read             =
 Log files processed                  =  -
 Last committed transaction           = 2001-05-29-21.20.16.000000

DB20000I  The ROLLFORWARD command completed successfully.

Dropping the TESTBACK database...
DB20000I  The DROP DATABASE command completed successfully.

Terminating the command line processor's back-end process...
DB20000I  The TERMINATE command completed successfully.
```

以下是该脚本的源列表:

```
-- Before proceeding, ensure that:
--   The database manager is running
--   The SAMPLE database exists and is not in use.

-- Run the script by issuing:
--   db2 -f backrest.db2 -t
--     where -f tells the command line processor to read command input
--               from a file instead of from standard input, and
--           -t tells the command line processor to use a semicolon (;)
--              as the statement termination character.

echo This is CLP script: backrest.db2;

-- Ensure that the DB2 profile registry variable DB2_ENABLE_LDAP
--   is set to 'NO':
!db2set DB2_ENABLE_LDAP=NO;

echo Deleting old SAMPLE database backup images...;
!rm -f ./SAMPLE.0.*;

echo Updating the database configuration parameter LOGRETAIN to 'ON'...;
```

```
update db cfg for sample using logretain on;

echo Backing up the SAMPLE database...;
backup db sample;

echo Restoring the SAMPLE database as TESTBACK (1st pass)...;
restore db sample into testback redirect;

echo Listing the table spaces for the TESTBACK database...;
list tablespaces;

echo Defining new table space containers for Tablespace 2...;
set tablespace containers for 2 using (path "ts2con1");

echo Listing table space containers for Tablespace 2 (TESTBACK database)...;
list tablespace containers for 2;

echo Restoring the SAMPLE database as TESTBACK (2nd pass)...;
restore db sample continue;

echo Rolling the TESTBACK database forward...;
rollforward db testback stop;

echo Dropping the TESTBACK database...;
drop db testback;

echo Terminating the command line processor's back-end process...;
   terminate;

-- End file
```

附录G. Tivoli Storage Manager

调用 DB2 备份或复原实用程序时，可指定您要使用 Tivoli Storage Manager（TSM，以前称为 ADSM）产品来管理备份或复原操作。可将 TSM 客户机的版本 3.1.x.3 以及更高版本与 DB2 配合使用。

在基于 UNIX 的平台上设置 Tivoli Storage Manager 客户机

在数据库管理器可使用 TSM 选项前，必须执行以下设置活动:

1. 在 SunOS 和 Solaris 环境上，执行下列步骤。（对于其他基于 UNIX 的平台，从步骤 2 开始。）
   a. 确保安装了所需的操作系统级别: SunOS 5.5.1 或 Solaris 2.5.1。
   b. 安装 TSM 客户机的版本 3.1.x.3 或更高版本。确保在安装此版本的客户机之前，除去了*所有*先前的 TSM 程序包。
   c. 验证 TSM 安装在目录 /opt/IBMDSMap5、/opt/IBMDSMba5 和 /opt/IBMDSMsa5 中。
   d. 在目录 /usr/lib 中创建下列符号链接（若它们尚不存在）:

      ```
      libApiDS.so -> libApiDS.so.1
      libApiDS.so.1 -> /opt/IBMDSMap5/api/libApiDS.so.2
      ```

2. 创建或修改 TSM 用户配置选项文件 /usr/sbin/dsm.opt 和 TSM 系统配置选项文件 /usr/sbin/dsm.sys 以适合您的环境。
3. 在 SunOS 和 Solaris 环境上，执行下列步骤。（对于其他基于 UNIX 的平台，继续步骤 4。
   a. 将 /usr/sbin/dsm.opt 和 /usr/sbin/dsm.sys 复制到目录 /opt/IBMDSMap5。
   b. 将 /opt/IBMDSMap5/solaris/dsmaptica 复制到目录 /opt/IBMDSMap5。
4. 设置 TSM 使用的环境变量:

   DSMI_DIR 标识 API 可信的的代理程序文件（dsmapicta 或 dsmtca）所在的用户定义目录路径。在 SunOS 和 Solaris 环境中，应该将它设置为 /opt/IBMDSMap5。

   DSMI_CONFIG 标识 dsm.opt 文件（它包含 TSM 用户选项）的用户定义目录路径。与另外两个变量不同，此变量应包含全限定路径和文件名。在 SunOS 和 Solaris 环境中，应该将它设置为 /opt/IBMDSMap5/dsm.opt。

DSMI_LOG 标识将在其中创建错误日志 (dsierror.log) 的用户定义目录路径。

5. 建立 TSM 密码。

   要使一个 Tivoli 客户机能够与 TSM 服务器连接，它必须有该服务器的密码。将可执行文件 dsmapipw 安装在实例所有者的 INSTHOME/sqllib/adsm 目录。这个可执行文件允许您建立和重设 TSM 密码。

   要执行 dsmapipw 命令，您必须作为 root 用户登录。当执行此命令时，将提示您输入下列信息:

   - *旧密码*，它是 TSM 服务器认可的该 TSM 节点的当前密码。第一次执行此命令时，密码是 TSM 管理员在 TSM 服务器上注册节点时所提供的密码。
   - *新密码*，它是 TSM 节点的新密码，存储在 TSM 服务器上。（注意: 将提示您输入新密码两次，以检查输入错误。）

   注: 调用备份或复原实用程序的用户不需要知道此密码。只需要在为初始连接建立了密码时，以及密码已在 TSM 服务器上复位后，运行 dsmapipw 命令。

6. 如果数据库管理器正在运行，应该:
   - 使用 **db2stop** 命令停止数据库管理器。
   - 使用 **db2start** 命令启动数据库管理器。

在其他平台上设置 Tivoli Storage Manager 客户机

在数据库管理器可使用 TSM 选项前，必须执行以下设置活动:

1. 设置 TSM 使用的环境变量:

   DSMI_DIR 标识 API 可信赖的代理程序文件（dsmapicta 或 dsmtca）所在的用户定义目录路径。

   DSMI_CONFIG
   标识 dsm.opt 文件（它包含 TSM 用户选项）的用户定义目录路径。与另外两个变量不同，此变量应包含全限定路径和文件名。

   DSMI_LOG 标识将在其中创建错误日志(dsierror.log) 的用户定义目录路径。

2. 若适合您的操作系统的话，则创建（或修改）TSM 系统配置选项文件 (dsm.sys)。
3. 创建（或修改）dsm.opt TSM 用户配置选项文件。环境变量 DSMI_CONFIG 指向此文件。

4. 建立 TSM 密码。

   要使一个 Tivoli 客户机能够与 TSM 服务器连接，它必须有该服务器的密码。将可执行文件 dsmapipw 安装在实例所有者的 \sqllib\adsm 目录。这个可执行文件允许您建立和重设 TSM 密码。

   要执行 dsmapipw 命令，必须作为本地管理员登录。运行此命令时，将提示您输入下列信息:

   - *旧密码*，它是 TSM 服务器认可的该 TSM 节点的当前密码。第一次调用此命令时，密码是 TSM 管理员在 TSM 服务器上注册节点时所提供的密码。
   - *新密码*，它是 TSM 节点的新密码，存储在 TSM 服务器上。（注意: 将提示您输入新密码两次，以检查输入错误。）

   注: 调用备份或复原实用程序的用户不需要知道此密码。只需要在为初始连接建立了密码时，以及密码已在 TSM 服务器上复位后，运行 dsmapipw 命令。

5. 如果数据库管理器正在运行，应该:
   - 使用 **db2stop** 命令停止数据库管理器。
   - 使用 **db2start** 命令启动数据库管理器。

使用 Tivoli Storage Manager 的注意事项

要使用 TSM 内的特定功能部件，可能需要给出使用该功能部件的对象的全限定路径名。（记住在 Windows NT 操作系统和 OS/2 上，将使用 \ 来代替 /。）全限定路径名是:

- 完整的数据库备份对象
  为: /<database>/NODEnnnn/FULL_BACKUP.timestamp.seq_no
- 表空间备份对象为: /<database>/NODEnnnn/TSP_BACKUP.timestamp.seq_no
- 装入副本对象为: /<database>/NODEnnnn/LOAD_COPY.timestamp.seq_no

其中 <database> 是数据库别名，NODEnnnn 是节点号。大写名称必须按显示的样子输入。

- 对于多个备份使用同一个数据库别名的情况，时间戳记和序号就成为全限定名中可区别的部分。需要查询 TSM，以确定要使用的备份版本。
- TSM 图形用户界面不认识个别的备份。备份映象存储到 TSM 管理的文件空间中。个别备份只能通过 TSM API 或通过使用这些 API 的 **db2adutl** 来管理（参见 第272页的『db2adutl - 处理 TSM 归档的映象』）。
- 若 Tivoli 客户机在服务器配置文件中的 COMMTIMEOUT 参数指定的时间内没有应答，则 TSM 服务器执行的会话将超时。有 3 个因素会导致超时问题:

  - 在 TSM 服务器上 COMMTIMEOUT 参数设置得太低。例如，如果在复原期间创建大的 DMS 表空间，就可能发生超时。此参数的建议值为 6000 秒。
  - DB2 备份或复原缓冲区可能太大。
  - 联机备份操作期间的数据库活动可能太频繁。
- 数据库管理器使用 TSM 完整备份选项；不支持 TSM 增量备份操作。
- 使用多个会话来增加处理能力。
- 在非 UNIX 平台上，DB2 备份和复原实用程序不允许多于一个的 TSM 会话。

当前 Windows 操作系统和 OS/2 上的 Tivoli 客户机支持重新进入，因此可以从单个机器，使用备份、复原或装入实用程序安全地创建多个 I/O 会话。然而，用户必须确认已安装的 TSM 客户机版本确实支持此功能。

在单节点配置中，若用户尝试发出 BACKUP DATABASE 命令，如:

```
db2 backup db sample use tsm open 3 sessions
```

DB2 将检测到 TSM 不支持多个会话，并返回错误消息。（这也适用于通过 COPY YES 选项，使用 TSM 调用的装入操作。）

但是，您应该知道，在 Windows NT 上的多逻辑节点 (MLN) 配置中，若每个逻辑节点只尝试创建一个会话，DB2 可能无法在单台机器上检测到多个会话的使用情况。基于此原因，对于 MLN 配置而言，验证它们的 TSM 客户机是否支持重新进入非常重要。若使用 TSM 同时备份、复原或装入多个逻辑节点，在每个节点尝试使用单个会话的条件下， DB2 将允许该操作继续，即使这些逻辑节点实际驻留在相同的硬件上。这可能导致备份尝试失败，并挂起装入进程，因此若没有最新的 TSM 客户机，便不应作这样的尝试。

在 TSM 上管理备份和日志归档

db2adutl 实用程序允许您查询、抽取和删除使用 TSM 保存的备份映象、日志和装入副本映象。该实用程序安装在基于 UNIX 的平台上的 INSTHOME/sqllib/misc 目录中和 Windows 操作系统及 OS/2 上的 \sqllib\misc 目录中。关于此实用程序的详细信息，参见 第272页的『db2adutl - 处理 TSM 归档的映象』。

QUERY 选项使您可列出备份映象、装入备份映象或日志。可以选择列出某个范围内的日志。还可以请求查看不活动的备份映象。

EXTRACT 选项使您可从 TSM 将备份映象或日志复制到当前目录。您可以选择要抽取哪个备份映象或日志。

DELETE 选项使您可取消激活备份映象或从 TSM 删除日志。您可以选择要处理哪个备份映象或日志。可以使用 KEEP *n* 选项来保留最新的 *n* 个备份映象。可以使用 OLDER THAN *timestamp* 或 *n* DAYS 选项来定制 DELETE 请求。

Tivoli 空间管理器与 Data Links 的集成

DB2 Data Links Manager 可使用 Tivoli Space Manager (TSM) 及其虚拟文件系统 (FSM)，FSM 将它自己置于本地日志文件系统 (JFS) 的顶层。可通过与 JFS 相同的方式访问并配置 FSM。

此功能对于那些文件系统中有必须定期移动到第三方存储设施的用户很有益处。必须定期管理这些文件系统的空间。DB2 Data Links Manager 对 TSM 的支持为管理 DATALINK 文件的空间提供了更大的灵活性。TSM 允许定期调整对 Data Links 管理的文件系统的分配，而没有在正常使用期间无意中就已填满了文件系统的风险；这与在 DB2 Data Links Manager 文件系统中，对可能存储在此的所有文件都预分配足够的存储空间不同。

限制与局限性

- 此功能当前只支持在 AIX 上使用。
- 只能由 root 用户来完成选择性迁移 (**dsmmigrate**)以及重调用 FC （读取许可权数据库）链接的文件。

  选择性迁移只能由文件所有者自己执行，在读许可权数据库文件中，为 DataLink Manager Administrator (dlfm)。要访问这样的文件，需要从主机数据库获得一个令牌。唯一不需要令牌的用户是 ″root″ 用户。″root″ 用户较容易执行选择性迁移和调用这类文件。dlfm 用户第一次只能使用有效的令牌来迁移 FC 文件。第二次尝试迁移时（重调用后），操作会失败，并返回错误消息 “ANS1028S 内部程序错误。请咨询您的服务代表。” 如果非 root 用户对 FC 文件尝试运行 **dsmmigrate**，操作会失败。由于通常是由管理器来访问文件服务器上的文件，所以该局限性的影响不大。
- **stat** 和 **statfs** 系统调用将 *Vfs-type* 显示为 `fsm`，而不是 `dlfs`，即使 dlfs 安装在 fsm 之上。

  此行为支持 **dsmrecalld** 守护程序的正常功能，该守护程序在文件系统上运行 **statfs** 以确定它的 *Vfs-type* 是否是 `fsm`。
- 如果具有最小 *inode* 数的文件是 FC（读许可权数据库）链接的，**dsmls** 命令不会显示任何输出。

  dsmls 类似于 **ls** 命令：它列示了由 TSM 管理的文件。不需要任何用户操作。

附录H. 数据库恢复的用户出口

您可以开发一个*用户出口程序*以自动执行日志文件的归档和检索。（在 OS/2 上，用户出口程序还可用于备份与复原操作。）在调用用户出口程序进行日志文件的归档或检索前，应确保 *userexit* 数据库配置参数已设置为 YES。它还使数据库可进行前滚恢复。

调用了用户出口程序后，数据库管理器将控制发送给可执行文件 db2uext2。（在 OS/2 上，备份和复原操作先调用 db2uexit.cmd，然后该程序再调用 db2uext2。）数据库管理器将参数发送到 db2uext2，完成后程序再将返回码发送回数据库管理器。由于数据库管理器只能处理有限的返回条件，所以用户出口程序应能够处理错误条件（参见第415页的『错误处理』）。而且由于在一个数据库管理器实例中只能调用一个用户出口程序，因此必须对可能要它执行的每个操作都有一个程序节。

包括下列主题:

- 『样本用户出口程序』
- 第413页的『调用格式』
- 第415页的『备份与复原注意事项（仅对于 DB2 OS/2 版）』
- 第415页的『错误处理』

样本用户出口程序

对所有受支持的平台都提供了样本用户出口程序。您可以修改这些程序以适应您的特定要求。样本程序中有很好的注释信息，它有助于您进行更有效的使用。

您应该了解用户出口程序必须从活动的日志路径*复制*日志文件到归档日志路径中。不要从活动的的日志路径除去日志文件。（这可能会在数据库恢复期间导致问题。）DB2 将从活动的日志路径中除去恢复操作不再需要的归档日志文件。

以下是 DB2 附带的样本用户出口程序的描述。

- **基于 UNIX 的系统**

  “DB2 基于 UNIX 的系统版”中的用户出口样本程序在 sqllib/samples/c 子目录中。虽然提供的样本是用 C 语言编码的，您的用户出口程序仍可以不同的编程语言编写。

  您的用户出口程序必须是可执行文件，名称为 db2uext2。

这里有四个可用于基于 UNIX 的系统的样本用户出口程序:

– db2uext2.cadsm

该样本使用 Tivoli Storage Manager 来归档和检索数据库日志文件。

– db2uext2.ctape

该样本使用磁带媒体来归档和检索数据库日志文件。

– db2uext2.cdisk

该样本使用操作系统的 COPY 命令以及磁盘媒体来归档和检索数据库日志文件。

– db2uxt2.cxbsa

此样本使用 Legato NetWorker** Version 4.2.5 程序（可从 Legato** Systems 公司获得）来归档和检索数据库日志文件。该样本只在 AIX 上受支持。

- **Windows 操作系统**

可在 sqllib\samples\c 子目录中找到用于 DB2 Windows 操作系统版的用户出口样本程序。虽然提供的样本是用 C 语言编码的，您的用户出口程序仍可以不同的编程语言编写。

您的用户出口程序必须是可执行文件，名称为 db2uext2。

对于 Windows 操作系统，有两个样本用户出口程序:

– db2uext2.cadsm

该样本使用 Tivoli Storage Manager 来归档和检索数据库日志文件。

– db2uext2.cdisk

该样本使用操作系统的 COPY 命令以及磁盘媒体来归档和检索数据库日志文件。

- **OS/2**

可在 \sqllib\samples\rexx 目录的实例子目录中找到用于 DB2 OS/2 版的用户出口样本程序。（dbuexit.CAD 程序例外: 它在 \sqllib\samples\c 目录的实例子目录中。）虽然提供的样本大多数是 REXX 命令文件，但是您的用户出口程序可用另一种程序设计语言来编写。

您选择要实现的样本应重命名为 db2uexit，且扩展名为 .cmd 或 .exe。将已重命名的文件移动到目录 \sqllib\bin中。

提供了 5 个 OS/2 样本用户出口程序:

– db2uexit.ex1

此样本使用可从 Seagate** Software 公司获得的 Sytos Premium** 版本 2.2 程序。它可用来将数据存储在 IBM 外部磁带机上，并可从该磁带机检索数据。当前仅支持 Sytos Premium 产品的版本 2.2。（需要 OS/2 修订包 26 才能使用此产品。）复查样本程序列表，以了解其他需求。

– db2uexit.ex2

  此样本使用可从 Mountain** 公司获得的 Filesafe** 程序。它可用于将数据存储在 Mountain 磁带机上，并从该磁带机检索数据。会为数据库的每个备份副本分配一个唯一的卷标，以便一个或多个数据库的多个备份映象可存储在相同的磁带上。

– db2uexit.ex3

  此样本使用可从 Maynard** 公司的获得的 MaynStream** 程序。它可用于将数据存储在 Maynard 磁带机上，并从该磁带机检索数据。MaynStream 不支持将数据库复原操作重定向到备份了数据库的那个驱动器以外的其他驱动器上。

– db2uexit.ex4

  此样本使用 OS/2 XCOPY 命令。存储设备可以是 OS/2 支持的任何一个设备，如硬盘、软盘或光盘。如果工作站配置正确，这些设备可成为 LAN 重定向的驱动器。

  XCOPY 不能用于备份和复原数据库。

– db2uexit.CAD

  以 C 语言编写的此样本等效于 Tivoli Storage Manager (TSM) 样本程序。可将它用于归档和检索数据库日志文件。

调用格式

数据库管理器调用用户出口程序时，将一组参数（数据类型为 CHAR）发送到程序。调用格式取决于您的操作环境。

- 用于基于 UNIX 的操作系统或 Windows NT/2000 的调用格式:

```
db2uext2 -OS<os> -RL<db2rel> -RQ<request> -DB<dbname>
-NN<nodenum> -LP<logpath> -LN<logname> -AP<tsmpasswd>
-SP<startpage> -LS<logsize>
```

os 指定实例是在哪个平台是运行的。有效的值是: AIX、Solaris、HP-UX、SCO、Linux、Dynix/ptx、SGI 和 NT。

db2rel 指定 DB2 发行级别。例如，SQL07020。

request 指定请求类型。有效的值是: ARCHIVE 和 RETRIEVE。

dbname 指定数据库名。

nodenum 指定本地节点号，例如 5。

logpath 指定日志文件的全限定路径。该路径必须包含尾部路径分隔符。例如 /u/database/log/path/ 或 d:\logpath\。

logname 指定要归档或检索的日志文件的名称，例如 S0000123.LOG。

tsmpasswd 指定 TSM 密码。（如果先前已指定了数据库配置参数 *tsm_password* 的值，该值将发送到用户出口程序。）

startpage 指定日志范围开始的那个设备的 4 KB 偏移页的数量。

logsize 指定日志范围的大小，以 4 KB 页为单位。只有将原始设备用于记录时，此参数才有效。

- OS/2 的调用格式:

```
action drive db_alias log_path log_file indicator
```

action 有效的值是: BACKUP、RESTORE、ARCHIVE 和 RETRIEVE。

drive 对于备份操作，指定要备份数据库的驱动器的位置。对于复原操作，指定复原数据库的目标驱动器。对于归档或检索操作，指定数据库所在的驱动器。在每种情况中，都指定一个后跟冒号的盘符（例如 C:）。

db_alias 指定数据库别名（或在不存在别名时，指定数据库名）。

log_path 对于备份或复原操作，指定一个响应文件的全限定名，该文件包含要备份或复原的一个文件列表。该列表中的每个名称都是一个可能包含了通配符的全限定文件名。对于复原操作，该盘符和路径代表了备份数据库文件时的源驱动器和路径。例如，如果 C:\SQLUTIL\dbname.MH1 包含在该响应文件中，则表示 dbname.MH1 文件是从 C:\SQLUTIL 备份的。

对于归档或检索操作，指定日志路径目录。例如 C:\SQL00001\SQLOGDIR\。

log_file 对于备份操作，指定由备份实用程序生成的媒体标号。此标号由数据库别名和时间戳记组成。对于复原操作，指定要将文件复原到的数据库子目录的路径名。不包括盘符，因为已通过 *drive* 参数指定了它。格式为 \SQL*nnnnn*\。

对于归档或检索操作，指定日志文件名称。例如 S0000001.LOG。

indicator 指定可用于在备份或复原操作期间支持多个调用的指示符。第一个调用的字符值是 1 而后继调用的字符值是 2。

在一个备份或一个复原操作期间，会多次调用用户出口程序。第一个调用会备份或复原媒体头 (.MH*n*) 文件，而第二个调用会备份或复原整套数据库文件。

不会将此参数用于归档或检索操作。

备份与复原注意事项（仅对于 DB2 OS/2 版）

如果正在编写从备份或复原实用程序调用的用户出口程序，则以下注意事项适用:

- 用户出口程序返回的非零返回码导致该实用程序失败，且不尝试重试。
- 在全限定文件名中只使用受支持的通配符。例如，C:\SQL00001*.* 和 C:*.MH* 都是可接受的搜索规则。
- 用户出口程序必须处理每行一个全限定文件名且每行以一个回车和换行符终止的响应文件格式。该文件中没有文件结束字符。
- 如果同一数据库的多个备份映象将放在一个媒体上，该用户出口程序应能够复原操作期间选择正确的映象。（参见第411页的『样本用户出口程序』中的 db2uexit.ex2 的描述。）
- 共享一个备份设备的两个同时运行的备份操作必须串行化。
- 如果一个备份映象跨越多个媒体，则必须由用户出口程序或它调用的应用程序来处理对媒体的提示。要支持此功能，备份和复原实用程序会打开一个操作系统前景会话以调用用户出口程序。
- 用户出口程序不能备份数据库目录内的任何子目录。
- 使用用户出口程序来复原数据库时，复原实用程序要求完全控制该数据库。然而，工作站可以与非复原的数据库建立活动的连接。
- 若正使用用户出口程序来备份或复原一个数据库，且另一个操作正在使用同一个磁带机，则该备份或复原操作可能会失败并需要重新启动。要避免这种情况，您应确保当一个备份或复原操作正在进行时，没有其他调用用户出口程序以进行记录的数据库在使用中；或者确保在设备未就绪时，确保用户出口程序可在稍后重新尝试备份或复原操作。
- 在复原操作期间，盘符和路径可以与备份操作期间指定的盘符和路径不同。例如，如果文件 dbname.MH1 是从 C:\SQLUTIL 备份的，可将它复原到 D:\SQLUTIL2。

错误处理

用户出口程序应设计为提供特定并有意义的返回码，以使数据库管理器可正确地解释它们。因为用户出口程序由基本的操作系统命令处理器调用，操作系统本身可以返回出错码。并且由于未重新映射这些出错码，使用操作系统消息将有助于实用程序获得关于这些出错码的信息。

在 OS/2 上，由用户出口程序返回的任何非 0 代码会导致备份或复原实用程序失败，且不尝试重试。实用程序会报告一个通用 SQLCODE -2029，它的消息文本显示了由用户出口程序或操作系统返回的代码。

表24显示了可由用户出口程序返回的代码，并描述数据库管理器是如何解释这些代码的。若返回码在该表中列出，则将其值当作 32 处理。

表 24. 用户出口程序返回码. 只适用于归档和检索操作。

返回码	说明
0	成功。
4	遇到临时资源错误。[a]
8	需要操作员介入。[a]
12	硬件错误。[b]
16	用户出口程序或该程序使用的软件功能出错。[b]
20	传送给用户出口程序一个或多个参数出错。验证用户出口程序是否正确地处理指定的参数。[b]
24	找不到用户出口程序。在 OS/2 上此错误消息还表示完成复原操作所需的一个文件在当前备份媒体上找不到。[b]
28	输入／输出 (I/O) 故障或操作系统导致的错误。[b]
32	用户出口程序由用户终止。[b]
255	由用户出口程序无法为可执行文件装入库文件而导致的错误。[c]

[a] 对于归档与检索请求，返回码 4 或 8 导致在五分钟后重试。如果用户出口程序继续对向同一日志文件发出的检索请求返回 4 或 8，DB2 会挂起。（适用于前滚操作或对 **sqlurlog** API（由复制实用程序使用）的调用。）

[b] 用户出口请求将暂挂 5 分钟。在此期间，将忽略所有的请求，包括导致出错状态的请求。在这个时间为 5 分钟的暂挂后，再处理下一个请求。如果处理此请求时没有错误，将继续处理新的用户出口请求，且 DB2 会重新发出失败过的或先前被暂挂的归档请求。若在重试期间生成一个大于 8 的返回码，则会将请求再暂停 5 分钟。此 5 分钟的暂挂将继续，直到该校正该问题或直到停止并重新启动该数据库。一旦已从该数据库断开了与所有应用程序的连接，DB2 会对先前可能未成功归档的任何日志文件发出归档请求。如果用户出口程序对日志文件归档失败，磁盘可能会填满了日志文件，从而可能使性能降低。一旦磁盘变满，数据库管理器将不再接受应用程序对更新数据库的请求。如果调用了用户出口程序来检索日志文件，则前滚恢复会暂挂但不停止，除非指定了 ROLLFORWARD STOP 选项。如果未指定 STOP 选项，可以校正问题并继续恢复。

[c] 如果用户出口程序返回错误码 255，可能是程序无法装入可执行文件所需的库文件。要进行验证，可手工调用用户出口程序。会显示更多信息。

注: 对于归档与检索操作，除返回码 0、4 和 24 外，对所有其他返回码都发出警告消息。该警告消息包含来自用户出口程序的返回码和提供给用户出口程序的输入参数的副本。

附录I. 供应商产品的备份与复原 API

DB2 提供了一些接口，可由第三方媒体管理产品用于存储和检索执行备份与复原操作的数据。此功能设计为将备份与复原操作的数据目标扩大到软盘、磁盘、磁带和 Tivoli Storage Manager（已将它作为 DB2 的一个标准部件来支持）。

这些第三方媒体管理产品在此附录的余下部分都将称为“供应商产品”。

DB2 定义了一组函数原型，它提供了一个用于备份和复原的通用数据接口，可由多个供应商使用。这些函数由供应商在基于 UNIX 的系统上的共享库中提供，或在 OS/2 或 Windows 操作系统上的 DLL 上提供。当 DB2 调用这些函数时，装入由调用备份或复原例程所指定的共享库或 DLL，并调用由供应商提供的函数以执行所需的任务。

附录分为以下四个部分:

- DB2 与供应商产品的交互作用的操作概述。
- DB2 供应商 API 的详细描述。
- API 调用中使用的关于数据结构的信息。
- 使用供应商产品来调用备份与复原的详细信息。

操作概述

定义了 5 个函数，以在 DB2 和供应商产品之间提供接口:

- sqluvint - 初始化与链接到设备
- sqluvget - 从设备读取数据
- sqluvput - 向设备写数据
- sqluvend - 解链设备
- sqluvdel - 删除落实的会话

DB2 将调用这些函数，它们应当由供应商产品在基于 UNIX 的系统上的共享库中提供，或在 OS/2 或 Windows 操作系统上的 DLL 中提供。

注: 将运行共享库或 DLL 代码，作为数据库引擎代码的一部分。因此，必须重新输入并彻底调试。函数如果有错就会影响到数据库的数据完整性。

DB2 在特定的备份或复原操作期间调用函数的顺序取决于:

- 将利用的会话数。
- 是备份操作还是复原操作。
- 对备份或复原操作指定的 PROMPTING 方式。
- 在其中存储数据的设备的特征。
- 在操作期间可能遇到的错误。

会话数

DB2 支持使用一个或多个数据流或会话来备份和复原数据库对象。使用 3 个会话的备份或复原需要有 3 个物理或逻辑设备可用。供应商设备支持正在使用中时，由供应商的函数负责管理对每个物理或逻辑设备的接口。DB2 只发送或接收流向或来自供应商提供的函数的数据缓冲区。

要使用的会话数由调用备份或复原数据库函数的应用程序作为参数来指定。此值在由 **sqluvint** 使用的 INIT-INPUT 结构中提供（参见 第425页的『sqluvint - 初始化与链接到设备』）。

DB2 将继续初始化会话直到达到指定的会话数，或接收到来自 **sqluvint** 调用的 SQLUV_MAX_LINK_GRANT 警告返回码。为了警告 DB2 已达到了它可支持的最大会话数，供应商产品将需要代码来跟踪活动的会话数。未成功警告 DB2 可能会使 DB2 初始化会话请求失败，从而导致所有会话的终止以及整个备份或复原操作的失败。

如果是备份操作，DB2 会在每个会话开始时写一个媒体头记录。该记录包含了 DB2 用于在复原操作期间标识会话的信息。通过将序列号附加到备份映象的名称后，DB2 可唯一地标识每个会话。会话数从 1 开始，对第一个会话使用 1，然后在每次通过 **sqluvint** 调用启动另一会话以进行备份或复原操作时加 1。要获取更多的详细信息，参见 第444页的『INIT-INPUT』。

备份操作成功完成后，DB2 会对它关闭的最后一个会话写一个媒体跟踪文件。跟踪文件中包含的信息告诉 DB2 使用了多少个会话来执行备份操作。在复原操作期间，这些信息将用于确保已复原所有会话或数据流。

没有错误、警告或提示时的操作

对于备份操作，DB2 会对每个会话发出以下顺序的调用。

```
sqluvint, action = SQLUV_WRITE
```

n 后加 1

```
sqluvput
```

加 1

```
sqluvend, action = SQLUV_COMMIT
```

DB2 发出 **sqluvend** 调用（操作 SQLUV_COMMIT）时，它希望供应商产品会适当地保存输出数据。对 DB2 发出返回码 SQLUV_OK 表示成功。

DB2-INFO 结构对 **sqluvint** 调用使用，它包含了标识备份所需的信息（参见 第440页的『DB2-INFO』）。提供了序列号。可能会选择供应商产品来保存此信息。DB2 将在复原期间用它来标识将要复原的备份。

对于复原操作，要对每个会话发出的调用序列是:

```
sqluvint, action = SQLUV_READ
```

n 后加 1

```
sqluvget
```

加 1

```
sqluvend, action = SQLUV_COMMIT
```

对 **sqluvint** 调用使用的 DB2-INFO 结构中的信息将包含标识备份所需的信息。未提供序列号。DB2 希望所有的备份对象（在备份期间落实的会话输出）都将返回。返回的第一个备份对象是使用序列号 1 生成的对象，而其他所有对象都不按特定的次序复原。DB2 将检查媒体末端对象以确保处理了所有对象。

注: 并非所有供应商产品都将保留一个备份对象的名称的记录。只有在备份到磁带上或其他容量有限的媒体上时，才最可能发生这种情况。在复原会话的初始化期间，可利用标识信息来登台需要的备份对象，以使它们在有需要时可用；这在使用 juke box 或机械人系统来存储备份时最有用。DB2 总是会检查媒体头（每个会话的输出中的第一个记录），以确保复原了正确的数据。

PROMPTING 方式

启动了备份或复原操作后，有两种可能的提示方式:

- WITHOUT PROMPTING 或 NOINTERRUPT，此时供应商产品没有机会向用户写消息，或用户也没有机会作出响应。
- PROMPTING 或 INTERRUPT，此时可接收并响应来自供应商产品的消息。

对于 PROMPTING 方式，备份与复原定义了三种可能的用户响应:

- 继续

  对设备的读或写数据操作将继续。
- 设备终止

  设备将不再接收数据，会话终止。

- 终止

  终止整个备份或复原操作。

本节中的稍后部分讨论了 PROMPTING 和 WITHOUT PROMPTING 方式的使用。

设备特征

为了使供应商设备支持 API，定义了两种常用的设备类型:

- 容量有限的设备，需要用户操作才能更换媒体；例如磁带机、软盘或 CDROM 驱动器。
- 容量很大的设备，在正常操作中不需要用户处理媒体；例如 juke box 或一种智能机械人媒体处理设备。

容量有限的设备可能需要在备份或复原操作期间提示用户装入附加媒体。通常，DB2 对于备份或复原操作中媒体装入的次序并不敏感。它还为向用户发送供应商媒体处理消息提供了设施。这种提示需要启动备份或复原操作时 PROMPTING 为 on。媒体处理消息文本在返回码结构的描述字段中指定。

如果 PROMPTING 为 on，而 DB2 从 **sqluvput**（写）或 **sqluvget**（读）调用接收到 SQLUV_ENDOFMEDIA 或 SQLUV_ENDOFMEDIA_NO_DATA 返回码，DB2 将:

- 如果调用是 **sqluvput**，则标记发送给该会话的最后那个缓冲区，以重新发送。稍后它将会放到会话中。
- 使用 **sqluvend** (action = SQLUV_COMMIT) 调用该会话。如果成功（返回码 SQLUV_OK），DB2 将:
  - 从指示媒体结束情况的返回码结构，发送供应商媒体处理消息给用户。
  - 提示用户发出继续、设备终止或终止响应。
- 如果响应为*继续*，DB2 将使用 **sqluvint** 调用来初始化另一会话，如果成功，则从会话读数据或向会话写数据。要在写数据时唯一地标识会话，DB2 会对序列号进行递增。通过 **sqluvint** 使用时，DB2-INFO 结构中的序列号可用，并且在媒体头记录中（该记录是发送给会话的第一个数据记录）。

  DB2 不会启动多于启动备份或复原操作时请求的会话，或由供应商产品用 **sqluvint** 调用时的 SQLUV_MAX_LINK_GRANT 警告来指示的会话。
- 如果响应为*设备终止*，DB2 不会尝试初始化另一会话，而活动的会话数由此减 1。DB2 不允许通过设备终止响应来终止所有的会话，至少在备份或复原操作完成前要有一个会话保持活动。
- 如果响应是*终止*，DB2 将终止备份或复原操作。关于 DB2 究竟为何不终止某些会话的更多信息，参见第421页的『如果向 DB2 返回了出错状态』。

因为备份或复原性能通常取决于正使用的设备数，所以维护并行性很重要。对于备份操作，鼓励用户响应为继续，除非他们知道余下的活动会话将保留仍将写出的数据。对于复原操作，也鼓励用户响应为继续，直到已处理完所有媒体。

如果备份或复原方式为 WITHOUT PROMPTING，而 DB2 接收到来自某个会话的 SQLUV_ENDOFMEDIA 或 SQLUV_ENDOFMEDIA_NO_DATA 返回码，它将终止会话，且不尝试打开另一会话。如果在备份或复原操作完成前，所有会话都对 DB2 返回“媒体结束”消息，操作将失败。因此，对容量有限的设备使用 WITHOUT PROMPTING 时应特别小心，但在对容量很大的设备操作时，该方法可行。

供应商产品有可能会对 DB2 隐藏媒体安装和转换操作，因此设备看起来象是有无穷大的容量。有一些容量非常大的设备就是以这种方式操作的。在这些情况中非常关键的一点是：将已备份的所有数据返回给 DB2 时所采用的次序与数据在复原操作时的次序相同。未成功地做到这一点会导致数据丢失，但 DB2 由于已无法检测到数据丢失，而假设复原操作成功。

DB2 将数据写到供应商产品时，假设每个缓冲区都包含在一个且只有一个媒体上（例如，一盒磁带）。供应商产品会将这些缓冲区分割为跨多个媒体，而不让 DB2 知道。在这种情况中，复原操作期间处理媒体的次序就变得很关键，因为供应商产品将负责把重新构造的缓冲区从多个媒体返回给 DB2。未成功地做到这一点将导致复原操作失败。

如果向 DB2 返回了出错状态

执行备份或复原操作时，DB2 希望所有会话都成功完成，否则整个备份或复原操作都将失败。会话通过 **sqluvend** 调用（操作 = SQLUV_COMMIT）上的 SQLUV_OK 返回码，向 DB2 发出成功完成的信号。

如果遇到了不可恢复错误，DB2 将终止该会话。它们可能是 DB2 错误，或是由供应商产品返回给 DB2 的错误。因为所有会话必须成功落实才能有一个完整的备份或复原操作，一个会话失败会导致 DB2 终止与该操作相关的其他会话。

如果供应商产品使用一个不可恢复的返回码响应来自 DB2 的调用，供应商产品可通过使用 RETURN-CODE 结构的描述字段中放置的消息文本，自选提供附加信息。将对用户出现此消息文本，同时还有 DB2 信息，从而可采取校正操作。

在有的备份方案中，已成功落实了一个会话，但与备份操作相关的另一会话却遇到了不可恢复的错误。因为必须在所有会话都成功完成后才能将备份操作视为成功，所以 DB2 必须删除已落实的会话中的输出数据：DB2 发出 **sqluvdel** 调用以请求删除该对象。不将此调用视为 I/O 会话，它负责初始化并终止删除备份对象时可能需要的任何连接。

DB2-INFO 结构将不包含序列号，**sqluvdel** 将删除与 DB2-INFO 结构中余下的参数匹配的所有备份对象。

警告状态

如果设备未就绪，或发生了其他某些可校正的状态，DB2 可能会从供应商产品接收到警告返回码。对读或写操作都如此。

在 **sqluvput** 和 **sqluvget** 调用上，供应商可将返回码设置为 SQLUV_WARNING，并通过使用 RETURN-CODE 结构的描述字段中放置的消息文本，自选提供附加信息。将对用户出现此消息文本，从而可采取校正操作。用户可用以下三种方法之一来响应，继续、设备终止或终止:

- 如果响应为*继续*，DB2 将在备份操作期间，尝试使用 **sqluvput** 来重写缓冲区。在复原操作期间，DB2 将发出 **sqluvget** 调用来读下一个缓冲区。
- 如果响应为*设备终止*或*终止*，DB2 将以在发生了不可恢复的错误时采取的同样方式（例如，它将终止活动的会话并删除已落实的会话），终止整个备份或复原操作。

操作提示与技巧

本部分提供了构建供应商产品时的某些提示与技巧。

恢复历史记录文件

恢复历史记录文件可用作数据库恢复操作的一个辅助方法。它与每个数据库相关，并且随着每次备份或复原操作而自动更新。关于恢复历史记录文件的更多信息，参见第41页的『了解恢复历史记录文件』。可通过以下设施，对此文件中的信息进行查看、更新或修剪:

- 控制中心
- 命令行处理器 (CLP)
  - LIST HISTORY 命令
  - UPDATE HISTORY FILE 命令
  - PRUNE HISTORY 命令
- API
  - db2HistoryOpenScan
  - db2HistoryGetEntry
  - db2HistoryCloseScan
  - db2HistoryUpdate
  - db2Prune

关于该文件的布局的信息，参见第323页的『数据结构: db2HistData』。

备份操作完成后，将把一个或多个记录写到该文件中。如果将备份操作的输出引导到供应商设备，则历史记录中的 DEVICE 字段会包含一个 0，且 LOCATION 字段会包含:

- 调用备份操作时指定的供应商文件名。
- 共享库的名称（如果未指定供应商文件名）。

关于指定此选项的更多信息，参见第449页的『使用供应商产品调用备份或复原操作』。

可使用“控制中心”、CLP 或 API 来更新 LOCATION 字段。如果已使用了容量有限的设备（例如可更换的媒体）来保留备份映象，就可以更新备份信息的位置，从而使媒体物理移动到另一（可能是离站）存储位置。如果是这种情况，就可以在需要进行恢复操作时，使用恢复历史记录文帮助您定位备份映象。

函数和数据结构

以下部分描述了可供供应商产品使用的一般函数和数据结构。

用于供应商产品的 API 是:

- 第425页的『sqluvint - 初始化与链接到设备』
- 第429页的『sqluvget - 从设备读取数据』
- 第432页的『sqluvput - 向设备写数据』
- 第435页的『sqluvend - 解链设备并释放其资源』
- 第438页的『sqluvdel - 删除落实的会话』

由供应商 API 使用的数据结构是:

第440页的『DB2-INFO』
包含向供应商设备标识 DB2 的信息。

第443页的『VENDOR-INFO』
包含标识供应商和设备版本的信息。

第444页的『INIT-INPUT』
在 DB2 和供应商设备之间设置逻辑链路。

第446页的『INIT-OUTPUT』
包含来自设备的输出。

第447页的『DATA』
包含在 DB2 和供应商设备间传送的数据。

第448页的『RETURN-CODE』
包含返回码和错误说明。

sqluvint - 初始化与链接到设备

调用此函数可提供关于初始化的信息，以及在 DB2 和供应商设备之间建立逻辑链路的信息。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sql.h

C API 语法

```
/* File: sqluvend.h */
/* API: Initialize and Link to Device */
/* ... */
int sqluvint (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 参数

Init_input
输入。包含由 DB2 提供的、用于与供应商设备建立逻辑链路的信息的结构。

Init_output
输出。包含由供应商设备返回的输出的结构。

Return_code
输出。包含要发送给 DB2 的返回码和简要文本说明的结构。

用法注释

对于每个媒体 I/O 会话，DB2 都将调用此函数以获得设备句柄。如果出于任何原因，供应商函数在初始化期间遇到了错误，它都将通过返回码指出该错误。如果返回码表示出现了错误，DB2 可能会通过调用 **sqluvend** 函数，选择终止该操作。关于可能的返回码的详细信息，以及 DB2 对每个返回码的反应都包含在返回码表中（参见表25）。

INIT-INPUT 结构中包含的元素可由供应商产品用于确定是否可进行备份或复原:

- size_HI_order and size_LOW_order

  这是备份的估计大小。它们可用于确定供应商设备是否可处理大小为这么多的备份映象。它们可用于估计保留备份所需的可更换媒体的数量。如果预计会出现问题，在第一次 **sqluvint** 调用失败时，该返回码可能会有用。

- req_sessions

  用户请求的会话数，可与估计大小和提示级别一起使用，以确定是否可进行备份或复原操作。

- prompt_lvl

  提示级别告诉供应商是否可以对某些操作进行提示，例如更换可更换的媒体（例如，将另一盒磁带放入磁带机中）。它可能会建议由于无法提示用户，所以该操作不能进行。

  如果提示级别是 WITHOUT PROMPTING，而可更换的媒体的数量大于所请求的会话数，DB2 将不能成功完成该操作（参见 第419页的『PROMPTING 方式』和第420页的『设备特征』获取更多信息）。

DB2 为正在写的备份或将通过 DB2-INFO 结构中的字段读取的复原命名。在操作 = SQLUV_READ 的情况下，供应商产品必须检查已命名的对象是否存在。如果找不到，返回码应设置为 SQLUV_OBJ_NOT_FOUND，从而使 DB2 采取相应的措施。

初始化成功完成后，DB2 会通过发出其他数据传输函数而继续，但是可能会通过 **sqluvend** 调用在任何时候终止会话。

返回码

表 25. 有效的 sqluvint 返回码及相应的 DB2 操作

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_OK	操作成功。	sqluvput、sqluvget（参见注释）	如果 action = SQLUV_WRITE，下一个调用将是（对 BACKUP 数据）调用 sqluvput。如果 action = SQLUV_READ，则验证返回 SQLUV_OK 之前，命名对象是否存在；下一个调用将是对 RESTORE 数据调用 sqluvget。

表 25. 有效的 sqluvint 返回码及相应的 DB2 操作 (续)

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_LINK_EXIST	先前激活的会话。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_COMM_ ERROR	与设备通信错误。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INV_VERSION	DB2 和供应商的产品不兼容。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INV_ACTION	请求了一个无效的操作。还可用来表示参数的组合产生了不可能的操作。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_NO_DEV_ AVAIL	此时没有设备可用。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_OBJ_NOT_ FOUND	找不到指定的对象。应在对 sqluvint 调用的操作为 'R'（读），且根据 DB2-INFO 结构中指定的标准找不到请求的对象时，使用该返回码。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_OBJS_FOUND	有多于 1 个对象与指定的准则匹配。当对 sqluvint 调用的操作为 'R'（读取）且有多个对象与 DB2-INFO 结构中的准则相匹配时产生此调用。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INV_USERID	指定了无效的用户标识。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INV_ PASSWORD	提供了无效的密码。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INV_OPTIONS	在供应商选项字段中遇到了无效的选项。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_INIT_FAILED	初始化失败，会话将终止。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_DEV_ERROR	设备错误。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。

表 25. 有效的 sqluvint 返回码及相应的 DB2 操作 (续)

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_MAX_LINK_GRANT	建立了最大链接数。	sqluvput 和 sqluvget（参见注释）	将它视为 DB2 提供的警告。此警告告诉 DB2 因为已达到了它可以支持的最大会话数（注: 这可能是由于设备的可用性），所以不要再打开与供应商产品的其他会话。如果 action = SQLUV_WRITE (BACKUP)，则下一个调用将是 sqluvput。如果 action = SQLUV_READ，则验证返回 SQLUV_MAX_LINK_GRANT 之前，命名对象是否存在；下一个调用将是对 RESTORE 数据调用 sqluvget。
SQLUV_IO_ERROR	I/O 错误。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。
SQLUV_NOT_ENOUGH_SPACE	没有足够的空间用于存储整个备份映象，所提供的估计大小是以字节计的 64 位值。	没有进一步的调用	会话初始化失败。释放为此会话分配的内存，然后终止。因为从未建立会话，所以将不会接收到 sqluvend 调用。

sqluvget - 从设备读取数据

在初始化后，可调用此函数以从设备读取数据。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sqluvend.h

C API 语法

```
/* File: sqluvend.h */
/* API: Reading Data from Device */
/* ... */
int sqluvget (
  void * pVendorCB,
  struct Data         *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API 参数

pVendorCB

输入。指向为 DATA 结构（包括数据缓冲区）和 Return_代码分配的空间的指针。

Data 输入／输出。指向 *data* 结构的指针。

Return_code
输出。来自 API 调用的返回码。

obj_num
指定应检索哪个备份对象。

buff_size
指定要使用的缓冲区大小。

actual_buff_size
指定读或写的实际字节。此值应设置为输出，以指示实际读取的数据字节数。

dataptr
指向数据缓冲区的指针。

reserve
保留以备将来使用。

用法注释

此函数由复原实用程序使用。

返回码

表 26. 有效的 sqluvget 返回码及相应的 DB2 操作

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_OK	操作成功。	sqluvget	DB2 处理数据
SQLUV_COMM_ERROR	与设备通信错误。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_ACTION	请求了一个无效的操作。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_DEV_HANDLE	无效的设备句柄。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_BUFF_SIZE	指定了无效的缓冲区大小。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_DEV_ERROR	设备错误。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_WARNING	警告。不应使用它来表示 DB2 的媒体结束；应使用 SQLUV_ENDOFMEDIA 或 SQLUV_ENDOFMEDIA_NO_DATA 来表示 DB2 的媒体结束。但可使用此返回码来表示设备未就绪的情况。	sqluvget 或 sqluvend, action = SQLU_ABORT	参见第422页的『警告状态』中 DB2 处理警告的说明。
SQLUV_LINK_NOT_EXIST	当前不存在链接。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_MORE_DATA	操作成功；有更多的数据可用。	sqluvget	

表 26. 有效的 sqluvget 返回码及相应的 DB2 操作 (续)

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_ENDOFMEDIA_NO_DATA	媒体结束，读取了 0 字节（例如磁带已结束）。	sqluvend	参见 第419页的『PROMPTING 方式』和第420页的『设备特征』下 DB2 处理媒体结束情况的说明。
SQLUV_ENDOFMEDIA	媒体结束， > 读取了 0 字节（例如磁带已结束）。	sqluvend	DB2 处理数据，然后如 第419页的『PROMPTING 方式』和第420页的『设备特征』中描述的那样，处理媒体结束情况。
SQLUV_IO_ERROR	I/O 错误。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
下一个调用: [a] 如果下一个调用是 sqluvend, action = SQLU_ABORT，此会话和所有其他活动的会话将会终止。			

sqluvput - 向设备写数据

在初始化后，可调用此函数以向设备写数据。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sqluvend.h

C API 语法

```
/* File: sqluvend.h */
/* API: Writing Data to Device */
/* ... */
int sqluvput (
  void * pVendorCB,
  struct Data  *,
  struct Return_code  *);
/* ... */

typedef struct Data
}
   sqlint32  obj_num;
   sqlint32  buff_size;
   sqlint32  actual_buff_size;
   void      *dataptr;
   void      *reserve;
{ Data;
```

API 参数

pVendorCB

输入。指向为 DATA 结构（包括数据缓冲区）和 Return_代码分配的空间的指针。

Data 输出。用将要写出的数据填充的数据缓冲区。

Return_code
输出。来自 API 调用的返回码。

obj_num
指定应检索哪个备份对象。

buff_size
指定要使用的缓冲区大小。

actual_buff_size
指定读或写的实际字节。此值应设置为指示实际读取的数据字节数。

dataptr
指向数据缓冲区的指针。

reserve
保留以备将来使用。

用法注释

此函数由备份实用程序使用。

返回码

表 27. 有效的 sqluvput 返回码及相应的 DB2 操作

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_OK	操作成功。	如果完成的话（例如，DB2 已没有更多的数据），为 sqluvput 或 sqluvend	通知其他进程操作成功。
SQLUV_COMM_ERROR	与设备通信错误。	sqluvend，action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_ACTION	请求了一个无效的操作。	sqluvend，action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_DEV_HANDLE	无效的设备句柄。	sqluvend，action = SQLU_ABORT[a]	会话将终止。
SQLUV_INV_BUFF_SIZE	指定了无效的缓冲区大小。	sqluvend，action = SQLU_ABORT[a]	会话将终止。
SQLUV_ENDOFMEDIA	已达到了媒体的末端，例如磁带的末端。	sqluvend	参见 第419页的『PROMPTING 方式』和第420页的『设备特征』下 DB2 处理媒体结束情况的说明。
SQLUV_DATA_RESEND	设备请求再次发送缓冲区。	sqluvput	DB2 将重新发送上一个缓冲区。只应进行一次该操作。
SQLUV_DEV_ERROR	设备错误。	sqluvend，action = SQLU_ABORT[a]	会话将终止。

表 27. 有效的 sqluvput 返回码及相应的 DB2 操作 (续)

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_WARNING	警告。不应使用它来表示 DB2 的媒体结束；应使用 SQLUV_ENDOFMEDIA 来表示。但可使用此返回码来表示设备未就绪的情况。	sqluvput	参见第422页的『警告状态』中 DB2 处理警告的说明。
SQLUV_LINK_NOT_EXIST	当前不存在链接。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
SQLUV_IO_ERROR	I/O 错误。	sqluvend, action = SQLU_ABORT[a]	会话将终止。
下一个调用: [a] 如果下一个调用是 sqluvend, action = SQLU_ABORT，此会话和所有其他活动的会话将会终止。已使用 sqluvint、sqluvdel 和 sqluvend 调用顺序删除了落实的会话。（参见第421页的『如果向 DB2 返回了出错状态』）。			

sqluvend - 解链设备并释放其资源

结束或解链设备，并释放它的所有相关资源。供应商必须在返回到 DB2 前，释放未使用的资源（例如已分配的空间和文件句柄）。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sql.h

C API 语法

```
/* File: sqluvend.h */
/* API: Unlink the Device and Release its Resources */
/* ... */
int sqluvend (
  sqlint32 action,
  void * pVendorCB,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 参数

操作 输入。用于落实或异常终止会话:

- SQLUV_COMMIT （ 0 = 要落实）
- SQLUV_ABORT （ 1 = 要异常终止）

pVendorCB

输入。指向 Init_output 结构的指针。

Init_output

输出。取消分配的 Init_output 的空间。如果操作是“要落实”，数据已落实到稳定存储，以进行备份。如果操作是“要异常终止”，则清除数据以进行备份。

Return code

输出。来自 API 调用的返回码。

用法注释

对已打开的每个会话调用此函数。有两个可能的操作代码:

- Commit

  将数据输出到此会话，或从此会话读取数据已完成。

  对于写（备份）会话，如果供应商返回到 DB2 时带有返回码 SQLUV_OK，DB2 会假设供应商产品已正确地保存了输出数据，并且如果在稍后的 **sqluvint** 调用中引用输出数据时，它是可访问的。

  对于读（复原）操作，如果供应商返回到 DB2 时带有返回码 SQLUV_OK，则因为可能会再次使用数据，所以不应删除数据。

  如果供应商返回 SQLUV_COMMIT_FAILED，DB2 会假设整个备份或复原操作可能有问题。所有活动的会话都会由 **sqluvend** 调用（并且 action = SQLUV_ABORT）终止。对于备份操作，落实的会话接收调用顺序 **sqluvint**、**sqluvdel** 和 **sqluvend**（请参阅第421页的『如果向 DB2 返回了出错状态』）。

- Abort

  DB2 遇到了问题，不会再对会话读或写数据。

  对于写（备份）会话，供应商应删除部分输出数据集，并在删除了部分输出时使用 SQLUV_OK 返回码。DB2 会假设整个备份有问题。所有活动的会话都将由 **sqluvend** 调用（且 action = SQLUV_ABORT）终止，落实的会话接收调用顺序 **sqluvint**、**sqluvdel** 和 **sqluvend**（参见第421页的『如果向 DB2 返回了出错状态』）。

  对于读（复原）会话，供应商不应删除数据（因为可能会再次需要它），但应进行清除，并返回给 DB2 SQLUV_OK 返回码。DB2 通过 **sqluvend** 调用（且 action = SQLUV_ABORT）终止所有的复原会话。如果供应商将 SQLUV_ABORT_FAILED 返回给 DB2，将不会向调用程序通知此错误，原因是 DB2 将返回第一个致命的故障，并忽略后继的故障。此时，如果 DB2 已调用了 **sqluvend**（并且 action = SQLUV_ABORT），一定会发生初始致命错误。

返回码

表 28. 有效的 sqluvend 返回码及相应的 DB2 操作

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_OK	操作成功。	没有进一步的调用	释放为此会话分配的所有内存，然后终止。
SQLUV_COMMIT_FAILED	落实请求失败。	没有进一步的调用	释放为此会话分配的所有内存，然后终止。

表 28. 有效的 sqluvend 返回码及相应的 DB2 操作 (续)

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_ABORT_FAILED	异常终止请求失败。	没有进一步的调用	

sqluvdel - 删除落实的会话

删除落实的会话。

权限

以下各项之一:

- *sysadm*
- *dbadm*

需要的连接

数据库

API 包含文件

sqluvend.h

C API 语法

```
/* File: sqluvend.h */
/* API: Delete Committed Session */
/* ... */
int sqluvdel (
  struct Init_input   *,
  struct Init_output  *,
  struct Return_code  *);
/* ... */
```

API 参数

Init_input
输入。为 Init_input 和 Return_code 分配的空间。

Return_code
输出。来自 API 调用的返回码。将删除 Init_input 结构所指向的对象。

用法注释

如果打开了多个会话且落实了某些会话，但其中一个会话失败了，将调用此函数以删除落实的会话。未指定序列号；**sqluvdel** 负责查找在特定的备份操作期间创建的所有对象并删除它们。INIT-INPUT 结构中的信息用于标识将删除的输出数据。对 **sqluvdel** 的调用负责建立从供应商设备删除备份对象所需的任何连接或会话。如果此调用的返回码是 SQLUV_DELETE_FAILED，DB2 不会通知调用程序，原

因是 DB2 将返回第一个致命的故障并忽略后继的故障。此时，如果 DB2 已调用了 **sqluvdel**，一定会发生初始致命错误。

返回码

表 29. 有效的 sqluvdel 返回码及相应的 DB2 操作

头文件中的文字	描述	可能的下一个调用	其他注释
SQLUV_OK	操作成功。	没有进一步的调用	
SQLUV_DELETE_FAILED	删除请求失败。	没有进一步的调用	

DB2-INFO

此结构包含向供应商标识 DB2 的信息。

表 30. DB2-INFO 结构中的字段. 所有字段都是以零结束的字符串。

字段名称	数据类型	描述
DB2_id	char	DB2 产品的标识符。它指向的字符串的最大长度是 8 个字符。
version	char	DB2 产品的当前版本。它指向的字符串的最大长度是 8 个字符。
release	char	DB2 产品的当前发行版。如果该字段不太重要，可设置为 NULL。它指向的字符串的最大长度是 8 个字符。
level	char	DB2 产品的当前级别。如果该字段不太重要，可设置为 NULL。它指向的字符串的最大长度是 8 个字符。
action	char	指定要进行的操作。它指向的字符串的最大长度是 1 个字符。
filename	char	用于标识备份映象的文件名。如果是 NULL，*server_id*、*db2instance*、*dbname* 和 *timestamp* 将唯一地标识备份映象。它指向的字符串的最大长度是 255 个字符。
server_id	char	标识数据库所驻留的服务器的唯一名称。它指向的字符串的最大长度是 8 个字符。
db2instance	char	db2 实例标识。这是调用命令的用户标识。它指向的字符串的最大长度是 8 个字符。
type	char	指定正在进行的备份的类型或正在执行的复原的类型。以下是可能的值: 当操作是 SQLUV_WRITE 时: 0 - 完整数据库备份 3 - 表空间级的备份 当操作是 SQLUV_READ 时: 0 - 完整复原 3 - 联机表空间复原 4 - 表空间复原 5 - 历史记录文件复原
dbname	char	要进行备份或复原的数据库的名称。它指向的字符串的最大长度是 8 个字符。
alias	char	要进行备份或复原的数据库的别名。它指向的字符串的最大长度是 8 个字符。
timestamp	char	用于标识备份映象的时间戳记。它指向的字符串的最大长度是 26 个字符。

表 30. DB2-INFO 结构中的字段 (续). 所有字段都是以零结束的字符串。

字段名称	数据类型	描述
sequence	char	指定备份映象的文件扩展名。对于写操作，第一个会话的值是 1，而每次通过 sqluvint 调用启动另一会话时，该值就会加 1。对于读取操作，该值总为零。它指向的字符串的最大长度是 3 个字符。
obj_list	struct sqlu_gen_list	列出备份映象的对象。向供应商提供该值，仅供他们参考。
max_bytes_per_txn	sqlint32	以字节为单位，向供应商指定由用户指定的传送缓冲区大小。
image_filename	char	保留以备将来使用。
reserve	void	保留以备将来使用。
nodename	char	生成备份的节点的名称。
password	char	生成备份的节点的密码。
owner	char	备份创始人的标识。
mcNameP	char	管理类。
nodeNum	SQL_PDB_NODE_TYPE	节点号。供应商接口支持大于 255 的号码。

filename 或 *server_id*、*db2instance*、*type*、*dbname* 和 *timestamp* 唯一地标识备份映象。由 *sequence* 指定的序列号标识文件扩展名。要复原某个备份映象时，必须相同的值才能检索该备份映象。取决于供应商产品，如果使用 *filename*，则其他参数可以设置为 NULL，反之亦然。

语言语法

C 结构

```
/* File: sqluvend.h */
/* ... */
typedef struct DB2_info
{
  char                  *DB2_id;
  char                  *version;
  char                  *release;
  char                  *level;
  char                  *action;
  char                  *filename;
  char                  *server_id;
  char                  *db2instance;
  char                  *type;
  char                  *dbname;
  char                  *alias;
  char                  *timestamp;
  char                  *sequence;
  struct sqlu_gen_list  *obj_list;
  long                  max_bytes_per_txn;
  char                  *image_filename;
  void                  *reserve;
  char                  *nodename;
  char                  *password;
  char                  *owner;
  char                  *mcNameP;
  SQL_PDB_NODE_TYPE     nodeNum;
} DB2_info;
/* ... */
```

VENDOR-INFO

此结构中的信息标识设备的供应商和版本。

表 31. VENDOR-INFO 结构中的字段. 所有字段都是以零结束的字符串。

字段名称	数据类型	描述
vendor_id	char	供应商的标识符。它指向的字符串的最大长度是 64 个字符。
version	char	供应商产品的当前版本。它指向的字符串的最大长度是 8 个字符。
release	char	供应商产品的当前发行版。如果该字段不太重要，可设置为 NULL。它指向的字符串的最大长度是 8 个字符。
level	char	供应商产品的当前级别。如果该字段不太重要，可设置为 NULL。它指向的字符串的最大长度是 8 个字符。
server_id	char	标识数据库所驻留的服务器的唯一名称。它指向的字符串的最大长度是 8 个字符。
max_bytes_per_txn	sqlint32	受支持的最大传送缓冲区大小。由供应商以字节指定。只有在供应商初始化函数的返回码为 SQLUV_BUFF_SIZE 时才使用它，表示指定了一个无效的缓冲区大小。
num_objects_in_backup	sqlint32	用于进行完整备份的会话数。用于确定在复原操作期间，何时处理了所有备份映象。
reserve	void	保留以备将来使用。

语言语法

C 结构

```
typedef struct Vendor_info
{
  char       *vendor_id;
  char       *version;
  char       *release;
  char       *level;
  char       *server_id;
  sqlint32   max_bytes_per_txn;
  sqlint32   num_objects_in_backup;
  void       *reserve;
} Vendor_info;
```

INIT-INPUT

此结构中包含的信息由 DB2 提供，用于设置并建立与供应商设备的逻辑链路。

表 32. INIT-INPUT 结构中的字段. 所有字段都是以零结束的字符串。

字段名称	数据类型	描述
DB2_session	struct DB2_info	以 DB2 的角度，对会话进行的描述。
size_options	unsigned short	选项字段的长度。使用 DB2 备份或复原函数时，此字段中的数据从 *VendorOptionsSize* 参数直接发送。
size_HI_order	sqluint32	以字节数估计的 DB 大小的高阶 32 位，总共 64 位。
size_LOW_order	sqluint32	以字节数估计的 DB 大小的低阶 32 位，总共 64 位。
options	void	调用备份或复原函数时从应用程序发送此信息。此数据结构必须是平面，也就是说，不支持间接级别。字节反转未完成，而此数据的代码页也未检查。使用 DB2 备份或复原函数时，此字段中的数据从 *pVendorOptions* 参数直接发送。
reserve	void	保留以备将来使用。
prompt_lvl	char	调用备份或复原操作时，用户请求的提示级别。它指向的字符串的最大长度是 1 个字符。
num_sessions	unsigned short	调用备份或复原操作时，用户请求的会话数。

语言语法

C 结构

```
typedef struct Init_input
{
  struct DB2_info  *DB2_session;
  unsigned short   size_options;
  sqluint32        size_HI_order;
  sqluint32        size_LOW_order;
  void             *options;
  void             *reserve;
  char             *prompt_lvl;
  unsigned short   num_sessions;
} Init_input;
```

INIT-OUTPUT

此结构包含由供应商设备返回的输出。

表 33. INIT-OUTPUT 结构中的字段

字段名称	数据类型	描述
vendor_session	struct Vendor_info	包含向 DB2 标识供应商的信息。
pVendorCB	void	供应商控制块。
reserve	void	保留以备将来使用。

语言语法

C 结构

```
typedef struct Init_output
{
  struct Vendor_info  *vendor_session;
  void                *pVendorCB;
  void                *reserve;
} Init_output;
```

DATA

此结构包含在 DB2 和供应商设备间传输的数据。

表 34. DATA 结构中的字段

字段名称	数据类型	描述
obj_num	sqlint32	DB2 在备份操作期间指定的序列号。
buff_size	sqlint32	缓冲区的大小。
actual_buf_size	sqlint32	发送或接收的实际字节数。一定不能超出 *buff_size*。
dataptr	void	指向数据缓冲区的指针。DB2 为缓冲区分配空间。
reserve	void	保留以备将来使用。

语言语法

C 结构

```
typedef struct Data
{
  sqlint32  obj_num;
  sqlint32  buff_size;
  sqlint32  actual_buff_size;
  void      *dataptr;
  void      *reserve;
} Data;
```

RETURN-CODE

此结构包含返回给 DB2 的错误的返回码和简短说明。

表 35. RETURN-CODE 结构中的字段

字段名称	数据类型	描述
return_code[a]	sqlint32	供应商函数中的返回码。
description	char	返回码的简短描述。
reserve	void	保留以备将来使用。
[a] 这是供应商特定的返回码，与由不同 DB2 API 返回的值不同。参见供应商产品接受的返回码的个别 API 描述。		

语言语法

C 结构

```
typedef struct Return_code
{
  sqlint32  return_code,
  char      description[60],
  void      *reserve,
} Return_code;
```

使用供应商产品调用备份或复原操作

可在调用 DB2 备份实用程序或 DB2 复原实用程序时，从以下各项指定供应商产品:

- 控制中心
- 命令行处理器 (CLP)
- 应用程序编程接口 (API)。

控制中心

“控制中心”是随 DB2 附带的，用于进行数据库管理的图形用户界面。

要指定	用于备份或复原操作的“控制中心”输入变量
使用供应商设备和库名	是 *Use Library*。指定库名（在基于 UNIX 的系统上）或 DLL 名（在 Windows 操作系统或 OS/2 上）。
会话数	是 *Sessions*。
供应商选项	不受支持。
供应商文件名	不受支持。
传送缓冲区大小	（对于备份操作）是 *Size of each Buffer*，而（对于复原操作）不适用。

命令行处理器 (CLP)

命令行处理器 (CLP) 可用来调用 DB2 BACKUP DATABASE 或 RESTORE DATABASE 命令。

要指定	命令行处理器参数	
	用于备份	用于复原
使用供应商设备和库名	*library-name*	*shared-library*
会话数	*num-sessions*	*num-sessions*
供应商选项	不受支持	不受支持
供应商文件名	不受支持	不受支持
传送缓冲区大小	*buffer-size*	*buffer-size*

应用程序编程接口 (API)

有两个 API 函数调用支持备份与复原操作: **sqlubkp** 可用于备份（参见第75页的『备份数据库 API』），而 **sqlurestore** 可用于复原（参见第101页的『复原数

据库 API』)。

要指定	（用于 **sqlubkp** 和 **sqlurestore** 的）**API** 参数是
使用供应商设备和库名	如下所示: 在 *sqlu_media_list* 结构中，指定媒体类型 SQLU_OTHER_MEDIA，然后在 *sqlu_vendor* 结构的 *shr_lib* 中指定共享库或 DLL。
会话数	如下所示: 在结构 *sqlu_media_list* 中指定 *sessions*。
供应商选项	*PVendorOptions*
供应商文件名	如下所示: 在结构 *sqlu_media_list* 中，指定媒体类型 SQLU_OTHER_MEDIA，然后在结构 *sqlu_vendor* 的 *filename* 中指定文件名。
传送缓冲区大小	*BufferSize*

附录J. 使用 DB2 资料库

“DB2 通用数据库”资料库由联机帮助、书籍（PDF 和 HTML）和 HTML 格式的样本程序组成。本节描述所提供的信息以及如何访问这些信息。

要访问联机产品信息，可以使用“信息中心”。有关更多信息，参见第463页的『通过“信息中心”访问信息』。可以查看任务信息、DB2 书籍、故障诊断信息、样本程序和 Web 上的 DB2 信息。

DB2 PDF 文件和打印的书籍

DB2 信息

下表将 DB2 书籍分为四个类别:

DB2 指南和参考信息
这些书籍包含所有平台的公共 DB2 信息。

DB2 安装和配置信息
这些书籍是针对特定平台上的 DB2 的。例如，有分别针对 OS/2 平台、Windows 平台和基于 UNIX 的平台上 DB2 的《*快速入门*》书籍。

HTML 格式的跨平台样本程序
这些样本是与“应用程序开发客户机”一起安装的样本程序的 HTML 版本。样本仅供参考，并不替代实际程序。

发行说明
这些文件包含 DB2 书籍中未能包括的最新信息。

HTML 格式的安装手册、发行说明和教程可直接在产品 CD-ROM 上看到。大部分书籍在产品 CD-ROM 上都有 HTML 格式以便查看，而在 DB2 出版物 CD-ROM 上则有 Adobe Acrobat (PDF) 格式以便查看和打印。还可从 IBM 订购打印的副本；请参阅第460页的『订购打印书籍』。下表列示了可订购的书籍。

在 OS/2 和 Windows 平台上，可在 `sqllib\doc\html` 目录下安装 HTML 文件。DB2 信息被翻译成各种语言；但是，并非所有的信息都有每一种语言的翻译版本。每当信息不能以某种特定语言表示出来时，就会提供英语信息。

在 UNIX 平台上，可在 `doc/%L/html` （其中 *%L* 表示语言环境）目录下安装多种语言版本的 HTML 文件。有关更多信息，参考适当的《*快速入门*》书籍。

您可以各种方法来获取 DB2 书籍并访问信息:

- 第462页的『查看联机信息』
- 第466页的『搜索联机信息』
- 第460页的『订购打印书籍』
- 第459页的『打印 PDF 书籍』

表 36. DB2 信息

名称	描述	书号 **PDF 文件名**	**HTML 目录**
	DB2 指南和参考信息		
《管理指南》	*《管理指南: 计划》*提供数据库概念的概述、有关设计问题（如逻辑和物理数据库设计）的信息，以及高可用性的讨论。 *《管理指南: 实现》*提供有关实现问题（如实现设计、访问数据库、审核、备份和恢复）的信息。 *《管理指南: 性能》*提供有关数据库环境以及应用程序性能评估和调整的信息。 在北美，可使用书号 SBOF-8934 来订购三卷英文版的*《管理指南》*。	SB84-0219 db2d1x70 SB84-0218 db2d2x70 SB84-0243 db2d3x70	db2d0
Administrative API Reference	描述 DB2 应用程序编程接口 (API) 以及您可以用来管理数据库的数据结构。此书还说明如何在应用程序中调用 API。	SC09-2947 db2b0x70	db2b0
《应用程序构建指南》	提供环境设置信息和关于如何在 Windows、OS/2 和基于 UNIX 的平台上编译、链接和运行 DB2 应用程序的循序渐进说明。	SB84-0220 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	提供关于使用 “DB2 通用数据库” 产品时可能遇到的 APPC、CPI-C 和 SNA 检测码的一般信息。 仅有 HTML 格式的版本。	无书号 db2apx70	db2ap

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
Application Development Guide	说明如何开发使用嵌入式 SQL 或 Java（JDBC 和 SQLJ）来访问 DB2 数据库的应用程序。讨论主题包括在分区环境或联合体系统中编写存储过程、编写用户定义函数、创建用户定义类型、使用触发器和开发应用程序。	SC09-2949 db2a0x70	db2a0
CLI Guide and Reference	说明如何开发使用 “DB2 调用层接口”（一个与 Microsoft ODBC 规范兼容的可调用 SQL 接口）来访问 DB2 数据库的应用程序。	SC09-2950 db2l0x70	db2l0
Command Reference	说明如何使用 “命令行处理器”，并描述可用来管理数据库的 DB2 命令。	SC09-2951 db2n0x70	db2n0
Connectivity Supplement	提供有关以下各项的设置和参考信息：如何将作为 DRDA 应用程序请求器的 DB2 AS/400 版、DB2 OS/390 版、DB2 MVS 版、DB2 VM 版与 “DB2 通用数据库” 服务器配合使用。此书还详述了如何将 DRDA 应用服务器与 DB2 Connect 应用程序请求器配合使用。 仅有 HTML 和 PDF 格式。	无书号 db2h1x70	db2h1
Data Movement Utilities Guide and Reference	说明如何使用 DB2 实用程序（如导入、导出、装入、自动装入程序和 DPROP）来使数据移动易于进行。	SC09-2955 db2dmx70	db2dm
《数据仓库中心管理指南》	提供有关如何使用 “数据仓库中心” 构建和维护数据仓库的信息。	SB84-0226 db2ddx70	db2dd
Data Warehouse Center Application Integration Guide	提供帮助程序员将应用程序与 “数据仓库中心” 和 “信息目录管理器” 集成的信息。	SC26-9994 db2adx70	db2ad
《DB2 Connect 用户指南》	提供 DB2 Connect 产品的概念、编程以及一般用法信息。	SB84-0221 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	提供 DB2 Query Patroller 系统的操作概述、特定操作和管理信息以及管理图形用户界面实用程序的任务信息。	SC09-2958 db2dwx70	db2dw

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
《DB2 Query Patroller 用户指南》	描述如何使用 DB2 Query Patroller 的工具和功能。	SB84-0222 db2wwx70	db2ww
《词汇表》	提供 DB2 及其组件中使用的术语的定义。 有 HTML 格式可用且在 *SQL Reference* 中。	无书号 db2t0x70	db2t0
《Image、Audio 和 Video Extenders 管理和编程》	提供有关 DB2 Extender 的一般信息，有关 Image、Audio 和 Video (IAV) Extender 的管理和配置的信息，以及有关使用 IAV Extender 进行编程的信息。它包括参考信息、诊断资料（带有消息）和样本。	SB84-0247 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	提供有关管理信息目录的指南。	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	提供“信息目录管理器”的体系结构接口的定义。	SC26-9997 db2bix70	db2bi
《信息目录管理器用户指南》	提供有关使用“信息目录管理器”用户界面的信息。	SB84-0227 db2aix70	db2ai
《安装和配置补遗》	指导您了解计划、安装和设置特定于平台的 DB2 客户机。此补遗还包含关于联编、设置客户机和服务器通信、DB2 GUI 工具、DRDA AS、分布式安装、配置分布式请求和访问多机种数据源的信息。	GB84-0127 db2iyx70	db2iy
《消息参考》	列出由 DB2、“信息目录管理器”和“数据仓库中心”发出的消息和代码，并描述应执行的操作。 在北美，您可订购两卷英文版的《消息参考》（使用书号 SBOF-8932）。	第 1 卷 GB84-0216 db2m1x70 第 2 卷 GB84-0217 db2m2x70	db2m0
OLAP Integration Server Administration Guide	说明如何使用“OLAP 集成服务器”的“管理器”组件。	SC27-0782 db2dpx70	n/a

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
OLAP Integration Server Metaoutline User's Guide	说明如何使用标准 "OLAP 元轮廓" 接口（而非通过使用 "元轮廓辅助程序"）创建和植入 OLAP 元轮廓。	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	说明如何使用标准 "OLAP 模型接口"（而非使用 "模型辅助程序"）来创建 OLAP模型。	SC27-0783 db2lpx70	n/a
《OLAP 安装与用户指南》	提供 OLAP Starter Kit 的配置和设置信息。	SA40-1755 db2ipx70	db2ip
《OLAP Spreadsheet Add-in 用户指南 Excel 版》	描述如何使用 Excel 电子表格程序来分析 OLAP 数据。	SA40-1756 db2epx70	db2ep
《OLAP Spreadsheet Add-in 用户指南 Lotus 1-2-3 版》	描述如何使用 Lotus 1-2-3 电子表格程序来分析 OLAP 数据。	SA40-1757 db2tpx70	db2tp
Replication Guide and Reference	提供随 DB2 提供的 "IBM 复制" 工具的计划、配置、管理和用法信息。	SC26-9920 db2e0x70	db2e0
《Spatial Extender 用户指南和参考》	提供关于 Spatial Extender 的安装、配置、管理、编程和故障诊断的信息。还提供对空间数据概念的重要描述，并提供 Spatial Extender 特定的参考资料（消息和 SQL）。	SB84-0249 db2sbx70	db2sb
《SQL 入门》	介绍 SQL 概念，并提供许多构造和任务的示例。	SB84-0223 db2y0x70	db2y0
SQL Reference，第 1 卷和第 2 卷	描述 SQL 语法、语义和语言规则。此书还包括关于发行版间的不兼容性、产品限制和目录视图的信息。 在北美，可使用书号 SBOF-8933 来订购两卷英文版的 *SQL Reference*。	第 1 卷 SC09-2974 db2s1x70 第 2 卷 SC09-2975 db2s2x70	db2s0
System Monitor Guide and Reference	描述如何收集关于数据库和数据库管理程序的各种信息。此书说明如何利用信息来了解数据库活动、提高性能和确定问题的原因。	SC09-2956 db2f0x70	db2f0

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
《Text Extender 管理和编程》	提供有关 DB2 Extender 的一般信息，有关 Text Extender 的管理和配置的信息，以及有关使用 Text Extender 进行编程的信息。它包括参考信息、诊断资料（带有消息）和样本。	SB84-0248 desu9x70	desu9
Troubleshooting Guide	帮助您确定错误源、从问题中恢复并向“DB2 客户服务”咨询以使用诊断工具。	GC09-2850 db2p0x70	db2p0
《新增内容》	描述“DB2 通用数据库的版本 7”中的新特性、函数和增强功能。	SB84-0224 db2q0x70	db2q0
	DB2 安装和配置信息		
DB2 Connect Enterprise Edition for OS/2 and Windows Quick Beginnings	提供 OS/2 和Windows 32 位操作系统上的DB2 Connect 企业版的计划、迁移、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2953 db2c6x70	db2c6
DB2 Connect Enterprise Edition for UNIX Quick Beginnings	提供基于 UNIX 的平台上的 DB2 Connect 企业版的计划、迁移、安装、配置和任务信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2952 db2cyx70	db2cy
《DB2 Connect 个人版快速入门》	提供 OS/2 和 Windows 32 位操作系统上的DB2 Connect 个人版的计划、迁移、安装、配置和任务信息。此书还包含所有受支持的客户机的安装和设置信息。	GB84-0212 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	在进行所有受支持的 Linux 分发时，提供“DB2 Connect 个人版”的计划、安装、迁移和配置信息。	GC09-2962 db2c4x70	db2c4
《DB2 Data Links Manager 快速入门》	提供“DB2 Data Links Manager AIX 版”和 Windows 32 位操作系统的计划、安装、配置和任务信息。	GB84-0211 db2z6x70	db2z6
《DB2 扩充企业版 UNIX 版快速入门》	提供在基于 UNIX 的平台上的 DB2 扩充企业版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0209 db2v3x70	db2v3

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
DB2 Enterprise - Extended Edition for Windows Quick Beginnings	提供 DB2 扩充企业版 Windows 32 位操作系统版的计划、安装和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2963 db2v6x70	db2v6
DB2 for OS/2 Quick Beginnings	提供 OS/2 操作系统上的 “DB2 通用数据库” 的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GC09-2968 db2i2x70	db2i2
《DB2 UNIX 版快速入门》	提供在基于 UNIX 的平台上的 “DB2 通用数据库” 的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0214 db2ixx70	db2ix
《DB2 Windows 版快速入门》	提供 Windows 32 位操作系统上的 “DB2 通用数据库” 的计划、安装、迁移和配置信息。此书还包含许多受支持的客户机的安装和设置信息。	GB84-0215 db2i6x70	db2i6
《DB2 个人版快速入门》	提供 OS/2 和Windows 32 位操作系统上的 “DB2 通用数据库个人版” 的计划、安装、迁移和配置信息。	GB84-0213 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	在进行所有受支持的 Linux 分发时，提供 “DB2 通用数据库个人版” 的计划、安装、迁移和配置信息。	GC09-2972 db2i4x70	db2i4
《DB2 Query Patroller 安装指南》	提供有关 DB2 Query Patroller 的安装信息。	GB84-0208 db2iwx70	db2iw
《DB2 仓库管理器安装指南》	提供仓库代理程序、仓库转换程序和 “信息目录管理器” 的安装信息。	GB84-0122 db2idx70	db2id
	HTML 格式的跨平台样本程序		
HTML 格式的样本程序	为所有受 DB2 支持的平台上的编程语言提供 HTML 格式的样本程序。提供的样本程序仅供参考。并非所有样本都有所有编程语言的版本。HTML 样本仅当安装了 “DB2 应用程序开发客户机” 时才可用。 有关这些程序的更多信息，参考 *《应用程序构建指南》*。	无书号	db2hs

表 36. DB2 信息 (续)

名称	描述	书号 PDF 文件名	HTML 目录
	发行说明		
DB2 Connect 发行说明	提供 DB2 Connect 书籍中未能包括的最新信息。	参见注释 2。	db2cr
DB2 安装说明	提供 DB2 书籍中未能包括的最新安装特定信息。	仅在产品 CD-ROM 上提供。	
DB2 发行说明	提供 DB2 书籍中未能包括的、有关所有 DB2 产品和功能部件的最新信息。	参见注释 2。	db2ir

注:

1. 文件名第六个位置的字符 *x* 指示书籍的语言版本。例如，文件名 db2d0e70 标识英语版本的《*管理指南*》，而文件名 db2d0f70 标识同一本书的法语版本。下列字母用在文件名的第六个位置以指示语言版本:

语言	标识符
巴西葡萄牙语	b
保加利亚语	u
捷克语	x
丹麦语	d
荷兰语	q
英语	e
芬兰语	y
法语	f
德语	g
希腊语	a
匈牙利语	h
意大利语	i
日语	j
韩国语	k
挪威语	n
波兰语	p
葡萄牙语	v
俄语	r
简体中文	c
斯洛文尼亚语	l
西班牙语	z
瑞典语	s
繁体中文	t
土耳其语	m

2. DB2 书籍中未能包括的最新信息以 HTML 格式在“发行说明”中提供，或作为 ASCII 文件提供。在“信息中心”中和产品 CD-ROM 上都提供了 HTML 版本。要查看 ASCII 文件:
   - 在基于 UNIX 的平台上，请参阅 Release.Notes 文件。此文件位于 DB2DIR/Readme/%L 目录中，其中 %L 表示语言环境名，而 DB2DIR 表示:
     - 在 AIX 上，是 /usr/lpp/db2_07_01
     - 在 HP-UX、PTX、Solaris 和 Silicon Graphics IRIX 上，是 /opt/IBMdb2/V7.1
     - 在 Linux 上，是 /usr/IBMdb2/V7.1。
   - 在其它平台上，请参阅 RELEASE.TXT 文件。此文件在安装了产品的目录中。在 OS/2 平台上，还可双击 **IBM DB2** 文件夹，然后双击**发行说明**图符。

打印 PDF 书籍

如果想要书籍的打印副本，则可打印 DB2 出版物 CD-ROM 上的 PDF 文件。使用 Adobe Acrobat Reader，可打印整本书籍或特定范围内的页。有关资料库中每本书的文件名，参见第452页的表36。

可从 Adobe Web 站点（网址 http://www.adobe.com）获取 Adobe Acrobat Reader 的最新版本。

这些 PDF 文件包括在 DB2 出版物 CD-ROM 上，文件扩展名为 PDF。要访问这些 PDF 文件:

1. 插入 DB2 出版物 CD-ROM。在基于 UNIX 的平台上，安装 DB2 出版物 CD-ROM。参考《*快速入门*》以了解安装过程。
2. 启动 Acrobat Reader。
3. 从下列位置之一打开期望的 PDF 文件:
   - 在 OS/2 和 Windows 平台上:

     x:\doc*language* 目录，其中 *x* 表示 CD-ROM 驱动器而 *language* 表示两个字符的国家或地区代码，它表示您所用的语言（例如，EN 表示英语）。
   - 在基于 UNIX 的平台上:

     CD-ROM 上的 */cdrom*/doc/*%L* 目录，其中 */cdrom* 表示 CD-ROM 的安装点而 *%L* 表示期望的语言环境的名称。

还可从 CD-ROM 将 PDF 文件复制至本地或网络驱动器并从该处读取它们。

订购打印书籍

通过使用销售单 (SBOF) 书号，可单独订购或成套订购已打印的 DB2 书籍（仅限北美）。要订购书籍，与 IBM 授权经销商或市场代表联系，或致电 1-800-879-2755（美国）或 1-800-IBM-4YOU（加拿大）。还可从 Publications Web 页面（网址为 http://www.elink.ibmlink.ibm.com/pbl/pbl）订购这些书籍。

有两套书籍。SBOF-8935 提供了“DB2 仓库管理器”的参考和用法信息。SBOF-8931 提供了所有其他“DB2 通用数据库”产品和功能部件的参考和用法信息。每个 SBOF 的内容列示在下表中:

表 37. 订购打印书籍

SBOF 号	包括的书籍	
SBOF-8931	• 管理指南: 计划 • 管理指南: 实现 • 管理指南: 性能 • Administrative API Reference • 应用程序构建指南 • Application Development Guide • CLI Guide and Reference • Command Reference • Data Movement Utilities Guide and Reference • 数据仓库中心管理指南 • Data Warehouse Center Application Integration Guide • DB2 Connect 用户指南 • 安装和配置补遗 • Image、Audio 和 Video Extenders 管理和编程 • 消息参考，第 1 卷和第 2 卷	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User’s Guide • OLAP Integration Server Model User’s Guide • OLAP Integration Server User’s Guide • OLAP 安装和用户指南 • OLAP Spreadsheet Add-in Excel 版用户指南 • OLAP Spreadsheet Add-in Lotus 1-2-3 版用户指南 • Replication Guide and Reference • Spatial Extender Administration and Programming Guide • SQL 入门 • SQL Reference，第 1 卷和第 2 卷 • System Monitor Guide and Reference • Text Extender 管理和编程 • Troubleshooting Guide • 新增内容

表 37. 订购打印书籍 (续)

SBOF 号	包括的书籍
SBOF-8935	• Information Catalog Manager Administration Guide • 信息目录管理器用户指南 • Information Catalog Manager Programming Guide and Reference • Query Patroller Administration Guide • Query Patroller 用户指南

DB2 联机文档

访问联机帮助

随所有 DB2 组件都附带提供了联机帮助。下表描述了各种类型的联机帮助。

帮助类型	内容	如何访问...
命令帮助	说明命令行处理器中命令的语法。	从命令行处理器，以交互方式输入: `? command` 其中 *command* 表示一个关键字或整个命令。 例如，`? catalog` 显示所有 CATALOG 命令的帮助，而 `? catalog database` 显示 CATALOG DATABASE 命令的帮助。
客户机配置辅助程序帮助 *命令中心帮助* *控制中心帮助* *数据仓库中心帮助* *事件分析程序帮助* *信息目录管理器帮助* *卫星管理中心帮助* *脚本中心帮助*	说明您可在窗口或笔记本中执行的任务。此帮助包括您需要知道的概述和先决条件信息，并描述如何使用窗口或笔记本控件。	从窗口或笔记本，单击**帮助**按钮或按 **F1** 键。

帮助类型	内容	如何访问...

消息帮助	描述消息的起因以及您应该执行的任何操作。	从命令行处理器，以交互方式输入:
		`? XXXnnnnn`
		其中 *XXXnnnnn* 表示有效的消息标识符。
		例如，`? SQL30081` 显示关于 SQL30081 消息的帮助。
		要每次查看一屏消息帮助，可输入:
		`? XXXnnnnn | 尚有`
		要在文件中保存消息帮助，可输入:
		`? XXXnnnnn > filename.ext`
		其中 *filename.ext* 表示想要保存消息帮助的文件。

SQL 帮助	说明 SQL 语句的语法。	从命令行处理器，以交互方式输入: `help statement` 其中，*statement* 表示 SQL 语句。 例如，`help SELECT` 显示有关 SELECT 语句的帮助。 **注**: 在基于 UNIX 的平台上，SQL 帮助不可用。
SQLSTATE 帮助	说明 SQL 状态及类代码。	从命令行处理器，以交互方式输入: `? sqlstate` 或 `? class code` 其中，*sqlstate* 表示有效的五位 SQL 状态，而 *class code* 表示该 SQL 状态的头两位。 例如，`? 08003` 显示 08003 SQL状态的帮助，而 `? 08` 显示 08 类代码的帮助。

查看联机信息

此产品中的书籍为超文本标记语言 (HTML) 软拷贝格式。软拷贝格式使您可搜索或浏览信息，并提供访问相关信息的超文本链接。它还使得在站点间共享资料库更容易。

可使用遵循 HTML 版本 3.2 规范的任何浏览器来查看联机书籍或样本程序。

要查看联机书籍或样本程序:

- 如果正在运行 DB2 管理工具，则使用 “信息中心”。

- 从浏览器，单击**文件 —> 打开页**。打开的页中包含 DB2 信息的描述和至 DB2 信息的链接:
  - 在基于 UNIX 的平台上，打开以下页:

    INSTHOME/sqllib/doc/*%L*/html/index.htm

    其中 *%L* 表示语言环境名称
  - 在其它平台上，打开以下页:

    sqllib\doc\html\index.htm

    该路径位于安装了 DB2 的驱动器上。

    如果尚未安装“信息中心”，则可通过双击 **DB2 信息**图符来打开该页。视您正在使用的系统不同，图符在主产品文件夹中或在“Windows 开始”菜单中。

安装 Netscape 浏览器

如果还未安装 Web 浏览器，则可从产品包装箱中的 Netscape CD-ROM 安装 Netscape。要获取如何安装它的详细指示信息，执行:

1. 插入 Netscape CD-ROM。
2. 安装 CD-ROM（仅限于在基于 UNIX 的平台上）。参考《*快速入门*》以了解安装过程。
3. 关于安装说明，可参考 CDNAV *nn*.txt 文件，其中 *nn* 表示两字符语言标识符。该文件位于 CD-ROM 的根目录下。

通过“信息中心”访问信息

“信息中心”提供对 DB2 产品信息的快速访问。在所有装有 DB2 管理工具的平台上，都提供了“信息中心”。

可通过双击“信息中心”图符来打开“信息中心”。视正在使用的系统的不同，该图符在主产品文件夹的“信息”文件夹中，或在 Windows 的**开始**菜单中。

还可通过使用工具栏和 DB2 Windows 平台上的**帮助**菜单来访问“信息中心”。

“信息中心”提供了六种类型的信息。单击适当的标签来查看提供给该类型的主题。

任务 可使用 DB2 执行的关键任务。

参考 DB2 参考信息，如关键字、命令以及 API。

书籍 DB2 书籍。

故障诊断 错误消息类别及其恢复操作。

样本程序 随“DB2 应用程序开发客户机”一起提供的样本程序。如果未安装“DB2 应用程序开发客户机”，则不显示此标签。

Web 万维网（WWW）上的 DB2 信息。要访问此信息，必须从系统连接至 Web。

当选择其中一个列表中的项时，“信息中心”启动一个查看器来显示信息。视所选择的信息种类的不同，查看器可能是系统帮助查看器、编辑器或 Web 浏览器。

“信息中心”提供了查找功能部件，因此您不用浏览这些列表就能查找特定主题。

对于全文本搜索，请遵循“信息中心”中指向**搜索 DB2 联机信息**搜索表格的超文本链接。

HTML 搜索服务器通常是自动启动的。如果 HTML 信息中的搜索不起作用，则可能必须使用下列其中一个方法来启动搜索服务器:

在 Windows 上

单击**开始**并选择**程序 —> IBM DB2 —> 信息 —> 启动 HTML 搜索服务器**。

在 OS/2 上

双击 **DB2 OS/2 版**文件夹，然后双击**启动 HTML 搜索服务器**图符。

如果在搜索 HTML 信息时遇到任何其它问题，可参考发行说明。

注: 搜索功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。

使用 DB2 向导

向导通过让您一次一步地完成每一个任务来协助您完成特定管理任务。可通过“控制中心”和“客户机配置辅助程序”来获取向导。下表列出了这些向导并描述了它们的用途。

注: “创建数据库”、“创建索引”、“配置多站点更新”和“性能配置”向导对分区数据库环境可用。

向导	帮助您...	如何访问...
添加数据库	在客户机工作站上编目数据库。	从“客户机配置辅助程序”单击**添加**。
备份数据库	确定、创建并调度备份计划。	从“控制中心”，右键单击想要备份的数据库并选择**备份 —> 数据库（使用向导）**。

向导	帮助您...	如何访问...
配置多站点更新	配置多站点更新、分布式事务或两阶段落实。	从“控制中心”，右键单击**数据库**文件夹并选择**多站点更新**。
创建数据库	创建数据库并执行一些基本配置任务。	从“控制中心”，右键单击**数据库**文件夹，并选择**创建 —> 数据库（使用向导）**。
创建表	选择基本数据类型并创建表的主键。	从“控制中心”，右键单击**表**图符，并选择**创建 —> 表（使用向导）**。
创建表空间	创建新的表空间。	从“控制中心”，右键单击**表空间**图符，并选择**创建 —> 表空间（使用向导）**。
创建索引	建议对于所有查询要创建和删除哪些索引。	从“控制中心”，右键单击**索引**图符，并选择**创建 —> 索引（使用向导）**。
性能配置	通过更新配置参数来调整数据库性能以满足您的业务需求。	从“控制中心”，右键单击想要调整的数据库并选择**使用向导配置性能**。 对于分区数据库环境，从“数据库分区”视图，右键单击想要调整的首个数据库分区并选择**使用向导配置性能**。
复原数据库	在故障之后恢复数据库。它帮助您了解要使用的备份及要重放的纪录。	从“控制中心”，右键单击想要复原的数据库并选择**恢复 —> 数据库（使用向导）**。

设置文档服务器

在缺省情况下，DB2 信息安装在本地系统上。这表示需要访问 DB2 信息的每个人都必须安装相同的文件。要将 DB2 信息存储在单个位置中，执行下列步骤:

1. 将所有文件和子目录从本地系统上的 `\sqllib\doc\html` 复制至 Web 服务器。每一本书都有其自己的子目录，该子目录包含构成该书的所有必需的 HTML 和 GIF 文件。确保目录结构仍相同。
2. 配置 Web 服务器以查找新位置中的文件。有关信息，可参考*《安装和配置补遗》*中的 NetQuestion 附录。
3. 如果正在使用“信息中心”的 Java 版本，可为所有 HTML 文件指定基本的 URL。您应将该 URL 用于书籍列表。
4. 当能够查看书籍文件时，可将经常查看的主题做成书签。您可能想把下列各页做成书签:
   - 书籍列表

- 经常使用的书籍的目录
- 经常引用的文章，如 ALTER TABLE 主题
- 搜索格式

有关如何从中央机器处理“DB2 通用数据库”联机文档文件的信息，参考*《安装和配置补遗》*中的 NetQuestion 附录。

搜索联机信息

要查找 HTML 文件中的信息，使用下列方法之一:

- 在顶部框中单击**搜索**。使用搜索格式来查找特定的主题。此功能在 Linux、PTX 和 Silicon Graphics IRIX 环境中不可用。
- 在顶部框中单击**索引**。使用索引来查找书中的特定主题。
- 显示帮助或 HTML 书籍的目录或索引，然后使用 Web 浏览器的查找功能查找书中的特定主题。
- 使用 Web 浏览器的书签功能来快速返回至特定的主题。
- 使用“信息中心”的搜索功能来查找特定的主题。请参见第463页的『通过“信息中心”访问信息』以获取详细信息。

附录K. 声明

IBM 可能未在所有国家或地区中提供本文档中讨论的产品、服务或功能部件。关于您所在区域目前可用的产品及服务的信息，请向当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并不明示或默示只能使用 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，都可以用来代替 IBM 产品、程序或服务。然而，评估和验证任何非 IBM 产品、程序或服务均由用户自行负责。

IBM 可能已经申请或正在申请与本文档有关的各项专利权。提供本文档并不表示允许您使用这些专利。 您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节 (DBCS) 信息的许可证查询，请与您的国家或地区的 “IBM 知识产权部” 联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于联合王国或该条款与当地法律不一致的任何国家或地区: 国际商业机器公司以 “仅此状态” 的基础提供此出版物，不附有任何形式的(（无论是明示的还是默示的）保证，包括（但不限于）不侵犯、适销性或适用于某特定用途的默示保证或条件。一些国家或地区允许否认某些事务中的明示或默示保证，因此，此声明可能不适用于您。

本资料可能会包含技术错误或印刷错误。此处的信息会定期得到更改；这些更改将编入本出版物的新版本中。IBM 可能随时对此出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

在此信息中对非 IBM Web 站点的任何引用仅是出于方便起见，不以任何方式提供对这些 Web 站点的保证。这些 Web 站点中的资料不是此 IBM 产品资料的一部分，使用这些 Web 站点时风险自负。

IBM 对于您以任何方式提供的任何信息，有权利以任何它认为适当的方式使用或分发，而不必对您负任何责任。

为了以下目的: (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 (ii) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的被许可方请与以下地址联系:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付款，都可获取这方面的信息。

本资料中描述的许可程序和它可用的全部许可证材料均由 IBM 根据 “IBM 客户协议”、 “IBM 国际程序许可证协议”或任何与客户之间的等效协议中的条款提供。

此处包含的所有性能数据都是在受控环境中确定的。因此，在其他操作环境中获得的结果可能与之相差很大。某些测量可能是在开发级的系统上进行的，不能保证这些测量方法在通用系统上同样可用。此外，某些测量方法可能是通过外推法归纳来估计的。实际结果可能会有所不同。此文档的用户应针对他们的特定环境验证数据是否适用。

关于非 IBM 产品的信息是从那些产品的供应商、他们发布的声明或其他公用来源获得的。IBM 未测试那些产品，不能确认与非 IBM 产品相关的性能、兼容性或任何其他声明的准确性。如有关于非 IBM 产品的功能的问题，应向那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能随时更改或撤销，而不作任何通知，并且仅代表发展目标。

本资料中可能包含用于日常业务运作的数据或报表的示例。为了尽可能完整地说明问题，这些示例可能包含个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址相似，纯属巧合。

版权许可证:

本资料可能包含源语言的样本应用程序，它们举例说明各种操作平台上的编程技术。为了开发、使用、市场营销或分发符合编写这些样本程序所针对的操作系统的应用程序编程接口的应用程序，您可以以任何形式复制、修改和分发这些样本

程序，而不必向 IBM 付款。尚未在所有条件下彻底测试这些示例。因此，IBM 不能保证或默示这些程序的可靠性、适用性或功能。

这些样本程序或任何派生产品的每个副本或任何部分都必须包括如下版权声明:

©（您的公司名）（年份）。本代码的某些部分是从 “IBM 公司样本程序” 派生的。
© Copyright IBM Corp. _输入年份_。All rights reserved.

商标

以星号 (*) 标出的下列各项是国际商业机器公司在美国和 / 或其他国家或地区的商标。

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extender
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

下列各项是其他公司的商标或注册商标:

Microsoft、Windows 和 Windows NT 是 Microsoft Corporation 的商标或注册商标。

Java 或所有基于 Java 的商标和徽标以及 Solaris 是 Sun Microsystems, Inc. 在美国和 / 或其他国家或地区的商标。

Tivoli 和 NetView 是 Tivoli Systems Inc. 在美国和 / 或其他国家或地区的商标。

UNIX 是经 X/Open Company Limited 唯一许可的在美国和 / 或其他国家或地区的注册商标。

以双星号 (**) 标出的其他公司、产品或服务名称，可能是其他公司的商标或服务标记。

索引

[G]

[H]

[J]

[K]

[L]

[M]

[P]

[Q]

[R]

[S]

[T]

E

G

H

I

L

M

N

O

P

X

与 IBM 联系

如果有技术问题，请在与“DB2 客户支持中心”联系之前复查并执行 *Troubleshooting Guide* 所建议的操作。本指南提出了一些建议，指导您收集一些信息从而帮助“DB2 客户支持中心”更好地为您服务。

要获取信息或订购任何“DB2 通用数据库”产品，与当地分支机构的 IBM 代表联系或与任何授权的 IBM 软件经销商联系。

您如果住在美国，请致电下列其中一个号码:

- 1-800-237-5511，可获得客户支持
- 1-888-426-4343，可了解所提供的服务项目

产品信息

您如果住在美国，请致电下列其中一个号码:

- 1-800-IBM-CALL (1-800-426-2255) 或 1-800-3IBM-OS2 (1-800-342-6672)，可订购产品或获取一般信息。
- 1-800-879-2755，可订购出版物。

http://www.ibm.com/software/data/

DB2 万维网网页提供关于新闻、产品描述和培训计划等等的当前 DB2 信息。

http://www.ibm.com/software/data/db2/library/

“DB2 产品和服务技术库”可供您访问常见问题、修正、书籍以及最新的 DB2 技术信息。

注: 此信息可能只有英文版。

http://www.elink.ibmlink.ibm.com/pbl/pbl/

“国际出版物”Web 订购站点提供关于如何订购书籍的信息。

http://www.ibm.com/education/certify/

IBM Web 站点中的“专业证书程序”提供各种 IBM 产品（包括 DB2）的证书测试信息。

ftp.software.ibm.com

以匿名形式登录。可在目录 `/ps/products/db2` 中找到有关 DB2 和许多其他产品的演示、修正、信息和工具。

comp.databases.ibm-db2, bit.listserv.db2-l

这些因特网新闻组可供用户来讨论使用 DB2 产品的经验。

On Compuserve: GO IBMDB2

输入此命令来访问 IBM DB2 系列论坛。这些论坛支持所有的 DB2 产品。

有关如何在美国以外的地区与 IBM 联系的信息，请参阅 *IBM Software Support Handbook* 的附录 A。要访问此文档，访问以下 Web 页面：http://www.ibm.com/support/，然后选择该页面底部附近的 IBM Software Support Handbook 链接。

注：在某些国家或地区，IBM 授权的经销商应与他们的经销商支持机构联系，而不是与“IBM 支持中心”联系。

DATA-RCVR-00

Spine information:

IBM® DB2® 通用数据库 数据恢复和高可用性指南与参考 版本 7