

## 第40章 分布式数据库管理

本章要点：

- 理解分布式数据库
- 使用一个分布数据库
- 使用分布式事务
- 理解只读快照
- 在列级创建一个只读快照
- 快速快照刷新

### 40.1 理解分布式数据库

分布式数据库是作为一个单独的数据库但位于不同场所的系统。这些场所可以在任何地方，从紧邻的办公室到世界的另一端。在一个具有不同结点连接起来的网络环境中，分布式数据库担当一个单独的系统。为了完全理解一个分布式系统是如何工作的，数据库管理员必须具备一定的知识，理解多硬件系统、网络、客户 / 服务器技术与数据库管理。

涉及到分布式数据库系统管理的信息很少。随着 Oracle 8i 版本与未来开发的数据仓库的出现，分布式数据库的使用有了极大的改变。许多 DBA 可能没有机会管理这种分布式系统，所以并不熟悉这种系统是什么。

本章描述各种类型的分布式系统，它们是怎样使用的，怎样管理它们。Oracle 8i 提供了一种创建多个分布式系统的简单的、实际的方法。然而，不是所有的公司都有能力支持一个 GUI 环境，或如果进行远程支持，只能使用服务器端的行命令模式。由于这个原因，本章描述的处理不提及分布式选项的 GUI 部分，只提及当鼠标点下之后发生什么，以及在没有 GUI 环境的情况下如何管理一个系统。Oracle 8 与 Oracle 8i 中出现一个名为 Oracle 企业管理器的 GUI 工具，用于管理分布式数据库。在 Oracle 8 中，企业管理器只在 Windows NT 上可用。在 Oracle 8i 中，企业管理器是一个基于 Java 的工具，可以从任何主要的平台上运行。

#### 40.1.1 描述每种类型的数据库

分布式数据库实际有两种类型：

- 具有远程查询、数据维护与两阶段提交的分布式数据库。

- 通过数据管理方法，诸如快照与触发器或其他非数据库管理方法（如 SQL\*PLUS 中的 COPY）的复制数据库。

最简单的分布式数据库是一系列独立的数据库在逻辑上通过网络链接到一起以形成一个单独的视图。复制数据库含有通过网络连接从其他远程数据库拷贝的信息。

复制数据库可简单按在它们之间传递信息所使用的方法进行分类。下面是这种拷贝处理的两种主要的方法（通常称为传播）：

- 分布式事务。

快照刷新。

分布式事务是这样一种处理：在其中用户对一个结点进行更新，通过触发器或过程将这个改变传送给另一个结点。快照是表的副本（或子集），可以从主结点传播到每个远程结点。

在分布式数据库系统中，有必要将数据跨系统进行连接，糟糕的数据存取技巧会导致在数据库服务器之间的用户数据流成为全部性能的主要因素。为了给一个最优化的执行计划决定最好的存取技巧，优化器首先必须能够决定足够数量的替代路径。在 Oracle 8i 中加强的远程连接表现出更多的选项，允许产生一个伴有相应性能增长的更好的执行计划。

#### 40.1.2 数据库命名规定

数据库命名、存取与管理的方式开始对存取因特网与基于因特网的数据库有较大的影响。Oracle 建议数据库与它们的链的命名规定遵循标准因特网域名命名规定，这个规定可以有許多部分，每一部分如 IP 地址一样，由点进行分隔。

数据库的名字从右向左读，这个命名规定可以有几个部分，第一部分（最右边的部分）是基本的或根域。缺省情况下，域名为 world。这在 Net 8 中并不要求，但为了一致性原因，最好还是包括它以支持 SQL\*Net 的老版本。

域名可以基于公司或所处位置的结构。例如，在德国的数据库的命名规定可以有几种方式，最简单的形式是，名字可以是 GERMANY.WORLD。如果数据库名字基于位置，名字将是 GERMANY.BWC，其中 BWC 是域。另一种方法将是使用大陆扩展位置，名字将是 GERMANY.EUROPE.BWC。不论是哪种命名规定，在将来的维护中，名字对程序员与用户必须是透明的而且易于理解。

**警告** 在命名一个域与数据库时还有一些限制。在常用数据库命名规定之外（没有空间、没有特殊的字符等等），操作系统或网络会限制名字的长度。参考 Oracle 指定的文档以查找名字的长度限制。

#### 40.1.3 归档透明性

这点很重要：当为远程系统上的表或对象提供名字时，命名规定允许系统的程序员与用户使用名字存取远程表，就像它们在本地图一样。透明性的概念是可以获取所有对象，这些对象对数据库管理员与用户是相同的。

Oracle 通过使用 SQL\*Net 与透明网关，使系统的开发看起来是相同的，而不管数据库的厂商、类型与位置。为获取更多的与数据透明性有关的信息，参考第 34 章。

透明性的目的是提供对所有数据库的无缝访问。例如，用户可以以相同的方式访问任何一个表（前提是它们具有安全性），可以使用与访问本地表相同的语法访问在 SYBASE 数据库上的表。可以使用与在底特律的本地表相同的语法访问在德国一个 Oracle 数据库中的表。要获取更多的透明网关信息，阅读 Oracle 8i 在相应的网关上的文档。例如，通过使用一个透明网关，SYBASE 系统现在看起来就像一个 Oracle 数据库，可以使用数据库链进行指引。

异构服务代理在 Oracle 8i 中处理为多线程。当系统中有大量用户会话并行访问相同的非 Oracle 系统时，这种新的功能将减少系统消耗的资源总数。这种更有效地使用系统资源的方式允许大量的并发用户会话。

下面是一个异构型环境的示例图：

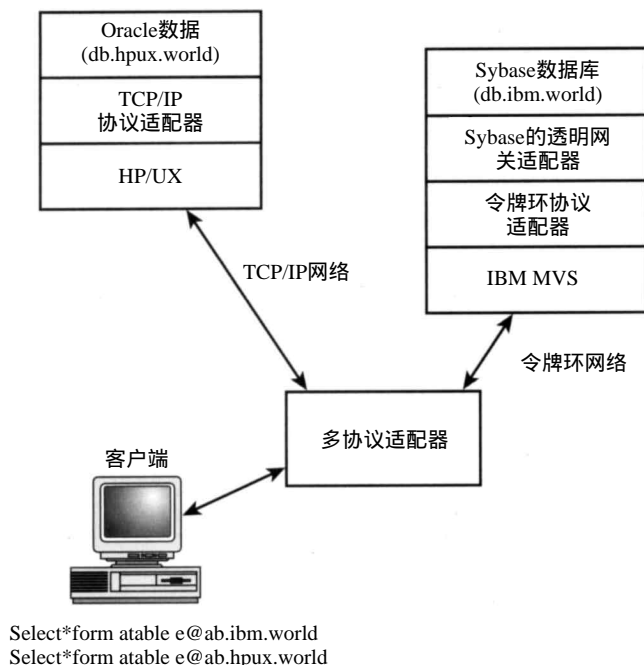


图40-1 分布式异构环境

必须有一个方法论以将任何数据库地址的改变传播到其他的分布式结点。这最终是数据库管理员的职责。许多结点将 TNSNAMES.ORA 与 SQLNET.ORA 放在一个集中的应用服务器上，这会以一种较方便的方法管理 TNSNAMES。然而，这并不总是最优的解决策略。在一个分布式环境中，每个结点上可能具有其他的独立的数据库（如测试数据库、该结点上特定的其他数据库等等），并且无法拥有一个本地化的 TNSNAMES。如果情况是这样，公司应该决定使用 Oracle NAMES 而不是 Net8 作为管理连接的方法。在创建一个分布式环境时，模式名、用户 ID 与口令也必须是可被管理的。例如，如果在创建数据库链时含有口令，口令就不能改变，如果口令改变了，必须告诉对受该模式影响的数据库链进行维护的数据库管理员。

**注意** 如果使用 Net 8，结点之间的通信很重要。必须有一个形式化的方法用于移动分布式数据库的 IP 地址或重命名数据库。

在 Oracle 8.0 中引入了代理自注册，它减少或消除了配置异构型服务时需要数据库管理员的介入。在 Oracle 8i 中，重写了代码使自注册处理更加有效。

Oracle 8i 中的新特性是代理指定，HS 对象文件的共享库而不是共享磁盘，它是在连接代理执行文件时的替代品。这个改变的益处是与平台有关的，可以通过为所有类型的代理使用一个单独的代理库（extproc、hsalloci、hssqlpss、hsdepaxa 及 hsots）增强它们的可伸缩性。而且对内存的需要将减少，因为代理执行程序会变得非常小。

#### 40.1.4 使用 Oracle 安全服务器与全局用户

Oracle 安全服务器是使数据库管理员或安全管理员管理全局环境中的安全的工具。安全服

务器使系统管理员可以创建一个用户，利用个人证明而不是口令进行验证。这种证明类似于请求访问支持电子商务的结点，这种证明代理之一是 VeriSign。

当注册到某一结点时，产生一个错误，警告用户在进入结点之前必须具备个人证明，他们的证明通过名字、地址与其他唯一鉴别所需要的信息鉴别他们。Oracle安全服务器第1版只对Web服务器执行证明。在第2版中，改进了这个能力，包括对Net 8客户与服务器的证明。依照Oracle文档，这种由Oracle安全服务器执行的证明遵守电子商务的国际标准（X.509）。

为确保只有具有相应特权的那些人才能访问数据库，Oracle创建了一个两层的安全级别。数据库管理员必须使用Oracle安全服务器创建用户，并且必须作为全局用户存在于数据库中。如果没有用户ID与证明，访问被拒绝。在创建全局用户时必须小心，用户ID不能既是全局的又是本地的。

#### 40.1.5 SQL\*Net

数据库管理员必须对怎样有效地设置与使用Net8管理分布式系统有深刻的理解。Net8及它的管理在第33章中讨论。

通过系统管理员和网络管理员协力，数据库管理员将必须建立有能力支持分布式系统的服务器。控制邮件或企业内部网系统的服务器不适于用作分布式数据库。网络负载将对任何数据库的性能有直接的影响，特别是分布式系统。

### 40.2 使用一个分布数据库

分布式数据库由通过数据库链与网络连接链接起来的多个数据库组成。对这些系统的管理可以大不相同。在大型、高度集中的企业中，每个位置必须有它自己的数据库管理员；在一个高度集中的企业中，一个数据库管理员将负责位于不同州，有些情况下，不同国家中的数据库。对这些数据库的管理与调整与自治的数据库系统相同，只是指向它们的链接与视图有所不同。

有许多原因需要使用一个分布式数据库系统，下面就是一些：

应用设计——一定的软件设计深化为一个分布式系统。具有多个位置，每个位置上有分离的IT/IS部门或数据库的公司将使用相同的或相似的软件。每个地点将维护它自己环境中唯一的数据。当公司发展时，每个地点将会要求访问其他结点位置上的数据，这最终会发展成为一个分布式系统。

改进的性能——本地数据库将比一个大型集中的数据库小。结果是，该数据库上的查询与其他的事务将会运行得更快。网络活动将显著降低，改进整个系统的性能。

较少的、不太昂贵的硬件需求——通过减少每个系统上用户的总数，支持这种系统的硬件可以相当地小（依赖于每个结点上的实际用户数）。例如，每个本地数据库可能会运行在一个单CPU NT服务器上，而不是运行在一个大型主机上。

改进的可靠性与可用性——通过将数据分布到多个结点，系统更有可能处于运行状态。当一个结点停机时，系统的其他结点仍处于运行状态并且是可访问的。这种软件的分布增加了另一维可靠性。通过具有拥有软件副本的结点，每个结点可以是一个镜像结点以支持一个灾难恢复计划。

改进的大量开发支持与前台应用。

在8.1 (8i) 版本中, Oracle给它的后台应用和前台应用提供了高级复制。后台应用要求实时复制数据, 前台应用区域是一个正在增长的市场, 特别对大量开发而言, 此时高级复制功能具有许多优势。

这些功能对企业是有用的。例如, 假设大型装饰品公司 ( Big Widget Company, BWC ) 是一个大的国际化的装饰品制造企业, 在密执安州的底特律、意大利的米兰、德国的法兰克福都有办事处。BWC开始一个进取性的研究与发展计划, 开发一种新产品用于国际使用。因为每个国家的标准不同, 每个国家的装饰品定义略有差异。这些区别保存在一个名为 SITE\_SPECIFICATION的分离的表中。因为这是一个研究与发展项目, 定义的改变是高度动态的, 每个国家需要临时访问其他国家的定义, 所以需要使用分布式数据库。

#### 40.2.1 设置一个分布式系统

分布式系统的设置必须独立于应用。如果软件使用分布式数据库, 许多由数据库管理员执行的工作, 诸如创建数据库链, 将在软件安装的过程中自动实现。然而, 在 BWC案例中, 由数据库管理员负责设置分布式数据库。因为这是一个研究与发展计划, BWC已决定在项目的开发中进行投资。结果是, 维护数据库的责任依赖于一个数据库管理员。

每个数据库将作为一个自治的数据库被安装。安装将基于每个国家的数据库上的并发用户的数量。其他在安装数据库时要考虑的因素如下所示:

- 数据库的所有使用。

- 使用数据库的其他应用。

- 标准DBA调整技术。

- 用于连接的网络协议的类型。

- 数据是如何动态变化的, 它将如何影响网络传输量。

- 当前网络传输量的级别以及新的数据库将对它产生什么影响。

注意 如果系统被大量地访问, 在决定并发用户时应该包括远程用户。

##### 1. 使用数据库链

数据库链组成了Oracle用于访问远程数据库对象的方法, 有三种类型的数据库链:

- 公有。

- 私有。

- 全局。

公有数据库链与公共同义词相似, 所有的用户都可以引用它。私有数据库链由链的拥有者 (模式) 访问, 公有数据库链在使用Oracle NAMES时自动创建。第33章详细解释Oracle Names。

在Oracle 8数据库链的语法中有一些显著的增加。创建一个公有数据库链或私有数据库链的主要部分是相同的。

```
CREATE
[SHARED]
[PUBLIC]
DATABASE LINK dblink
[authenticated clause] [(CONNECT TO [CURRENT_USER|user
IDENTIFIED BY password] [authenticated clause] USING '{connect string}');
```

这个语法在Oracle 8与Oracle 7中有一些不同, 在Oracle 8中增加了一个新的进程, 另一个

进程也被改变。

SHARED是一个全新的进程。为了使用这个进程，数据库必须使用一个多线程服务器。这使创建数据库链时可以使用现存的可用的连接。除非数据库管理员已完全理解不能共享数据库链与多线程服务器的隐含概念，否则不能这样做。如果设置不正确，将严重影响系统的性能。为获得有关共享数据库链及怎样有效地使用它们的详细信息，阅读《Oracle8 Server Distributed Database Systems》中的第2章。

验证子句与数据库链的 SHARED部分有关。当使用 SHARED时，必须使用它。下面是验证子句的语法：

```
AUTHENTICATED BY username IDENTIFIED BY PASSWORD
```

在验证子句中用户并不执行任何操作，只要求用户名是远程服务器上的一个合法用户与口令。数据库管理员可以专为这个目的创建一个哑帐号。

CURRENT\_USER使用新的Oracle8的全局用户类型，这个强有力的新功能能够创建一个用户，该用户可以使用一个单独的注册访问结点上的任何数据库。这并不会与本地用户产生混淆。这个用户必须使用Oracle8的安全服务器创建。

使用BWC示例，下面的代码将从底特律创建一个到德国的简单的公有数据库链。

```
CREATE PUBLIC DATABASE LINK germany.bwc.com  
USING 'GERMANY';
```

注意数据库名字对大小写是敏感的。如果不是由 Oracle Names进行的定义，必须以别名的形式或者以 TNSNAMES.ORA中的真实名字的形式显示 GERMANY名字。TNSNAMES.ORA必须在客户和服务上。使用本语法，用户当前必须注册到在德国和底特律的数据库中。假设模式名在德国与底特律是相同的，访问 SITE\_SPECIFICATION的语法如下：

```
SELECT * FROM SITE_SPECIFICATION@GERMANY.BWC.COM;
```

注意 SQL语句内的链接不是大小写敏感的。

提示 为确保透明性，创建一个同义词或视图，对用户隐藏数据库链。

```
CREATE SYNONYM germany_specification  
FOR site_specification@germany.bwc.com;
```

可以创建简单的数据库链，也可以创建复杂的数据库链，这由数据库管理员决定。例如，可以创建一个连接到欧洲的链，如果 BWC扩展到欧洲的其他国家，这将非常有利。米兰和法兰克福的数据库链为：

```
CREATE DATABASE LINK europe using 'EUROPE.BWC';
```

然而，在创建一个同义词时，这个初始化连接将不起作用，数据库管理员通过增加国家扩展这个名字。

```
CREATE SYNONYM germany_specification FOR site_specification@europe@germany;
```

注意 与Oracle以前的版本一样，Oracle8不支持从远程数据库选择LONG数据类型。

## 2. 延迟事务

延迟事务是稍后执行的事务。例如，在主结点上进行的改变存储在一个队列中，并且被延迟直到复制机制取出改变并将它们应用到远程结点（参见表 40-1）。

### 3. 对分布式系统使用初始化参数

除在分布式数据库一节指明的参数以外，更多的参数如表 40-1 所示。

表40-1 被延迟事务数据库影响的参数

参 数	描 述
COMMIT_POINT_STRENGTH(0-255)	这个参数用于设置在两阶段提交中的提交点场地。具有最高提交点强度的结点是提交点场地。每个使用这个结点作为提交点场地的结点必须处于相同的结点上。决定哪个数据库将成为提交点场地的因素是数据的拥有者（驱动器）、数据的关键程度与系统的可用性。有关提交的信息存在提交点的系统上。建议所有的结点还要具有相同的值或只有一个较大的值，这确保了万一失效，其他的结点可以记录这个数据。这个参数的缺省值与操作系统有关，参考 Oracle 8 特定操作系统文档以找出这些值
DISTRIBUTED_TRANSACTIONS (0-TRANSACTIONS)	并发分布式事务的数量的限制。如果设为 0，不激活 RECO 进程（Oracle 的恢复进程），禁用数据库的分布式能力
GLOBAL_NAMES	如果设为 TRUE，链中所引用的名字必须与数据库名匹配，而不是与别名匹配。为使用 Oracle 8 的高级复制功能，必须使用这个值
DML_LOCKS(20-无限)	在一个单独的事务中限制 DML 锁数量
ENQUEUE_RESOURCES(10-65535)	允许几个并发进程共享资源。为确定这个值是否设置得当，查看 v\$sysstat 中的 enqueue_waits 值。如果它是非零值，增大 enqueue 资源。
MAX_TRANSACTION_BRANCHES(1-32)	这个参数的最大值可以从 8~32。分支是在一个单独的分布式事务中可以访问的不同服务器（或服务器组）的数量。减小这个数可能会降低共享值使用的总数
OPEN_LINKS(0-255)	一个会话中打开的到其他数据库的并发连接的数量。在分布式环境中，必须小心确保这个值不小于在一个单独的 SQL 语句中可以引用的远程表的数量。如果将它设为 0，将不再有分布式事务
OPEN_LINKS_PER_INSTANCE (0-UB4MAXVAL)	这是 Oracle 8 的新参数，它限制了由外部事务管理器创建的打开链接的数量，要获得详细信息，参见 Oracle 8i 文档

#### 40.2.2 识别分布式系统中的潜在问题

在管理一个简单的分布式系统中，系统改变是最大的问题，这与管理一个单独的、自治的数据库相似，但区别在于每个数据库可以看到其他数据库中的对象，这样就创建了一个依赖于连续性的系统。下面是必须在分布式系统内部进行调整的变化。

**结构的改变**——当远程表作了结构上的修改或被删除后，使用过程与触发器更新或复制特定的远程表将会失败。指向一个被删除的表的数据库链将会失败。

**注意** 如果用户删除了另一个系统链接引用的表，将不会有引用完整性警告，只有当表中含有被引用的元素时才发生这些警告。

**模式的改变**——除表被移走之外，数据库链中的引用用户必须具有正确的权限。对资源环境文件与角色的改变必须进行调整以确保它们不会影响分布式系统的其余部分。这包括口令！

**对 SQL\*Net 对象的改变**——文件诸如 TNSNAMES、PROTOCOL 与 TNSNAV 将影响分布式数据库系统。

**系统停机**——在停机之间进行调整至关重要，特别是对依赖于分布式更新、快照或复制的系统。有些情况下，如果这个处理被打破，必须进行手工干涉以修复这个问题。

除非数据库管理员被告知数据库或网络停机，否则分布式系统的完整性将受到危害。

组织之间的交流至关重要，特别是在开发者与数据库管理员之间的交流。一个数据库或IP地址结构的改变会影响整个企业及分布式系统的完整性。这种调整通常由DBA管理。

如果在数据库之间的连接丢失了，可能发生另一个问题。首先DBA负有决定如果连接丢失会发生什么的职责。DBA必须能够容易地识别错误信息，特别是与SQL\*net或Net8相关的错误信息。更多的有关连接问题解决策略的信息，参见本书第33章。

### 40.2.3 调整分布式系统

分布式数据库的调整方法与自治数据库相同，请参阅第四部分并记住以下规则：

如果分布执行，书写拙劣的SQL仍会执行较差。

本地基于成本的优化不会将分区的远程表看作是分区的。

避免在一个单独的SQL语句中使用多个数据库链。

## 40.3 使用分布式事务

当提及分布式事务与两阶段提交时，复杂性似乎是压倒一切的。记住一个主要的思想：Oracle 7与Oracle 8i自动管理分布式事务。大多数DBA没有机会真正看到一个两阶段提交，因为它们是基于触发器的进程。下面的例子用图解说明了一个两阶段提交。

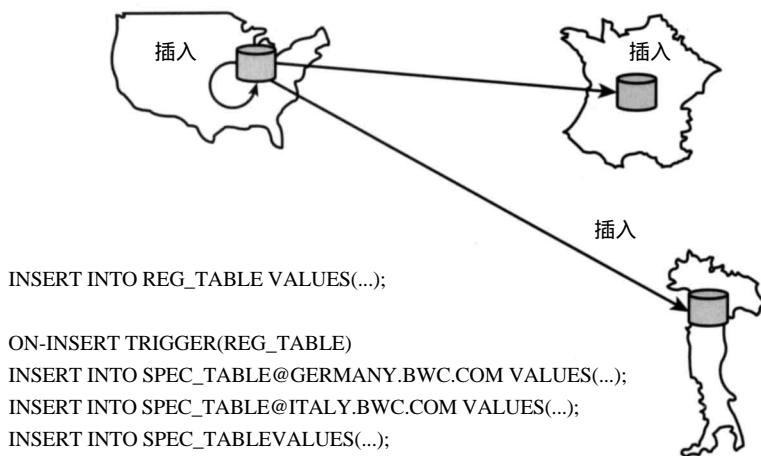


图40-2 两阶段提交

BWC有一个规范表 (SPEC\_TABLE)，依赖于美国 (US\_REG) 中的规则表。当US\_REG表中插入新的数据或进行更新时，应用更新并向在德国、意大利与底特律的SPEC\_TABLE表中插入数据。

### 40.3.1 理解两阶段提交

所有的数据操作语言 (DML) 伴随发生两阶段提交，并且如名字所意味的，它是由两个截然不同的阶段组成的。为了确保解释的一致性，本段使用的DML是更新。

两阶段提交的第一阶段称为准备阶段。起始的数据库称为全局协调器，它向所有其他的数据库发送信息，通知它们将要发生一个更新。这个信息并不具有真正的更新，只是一个很

小的总量以使其他数据库知道将更新什么。如果更新发生，接收的数据库将响应全局协调器更新。接收的数据库可以三种方式进行响应：

准备好——已准备好进行更新。

只读——没有准备的必要。

退出——子处理不能执行更新。

注意 全局协调器将决定数据库被更新的顺序。

接收的数据库也必须进行一些系统检查，包括发送一个准备语句给它自己的所有协同的数据库。这包括在将被更新的行放置锁（如果不是只读的）。在所有的接收数据库进行响应之后，全局协调器将事务置于悬而未决状态。如果所有的子响应都是准备好或只读的，全局协调器决定提交点。

DBA根据数据库的稳定性决定哪个数据库将作为提交点。例如，在 UNIX服务器上的数据库将比在PC上的数据库更加稳定。主要的目标是选出一个不会由于悬而未决事务而创建一个锁的数据库。被提交或回滚的事务称为悬而未决事务。

DBA由参数文件中的COMMIT\_POINT\_STRENGTH的值分配提交点。分布树中的较高的值决定了提交点强度，如果系统是高度分布的，可以有一个以上的数据库被分配作为提交点。如果事务影响不同的数据库，这也适用。提交点是第一个将被提交的数据库。它保持事务的状态直到所有的事务完成。

第二个阶段，或提交阶段，是提交数据的过程。在所有接收数据库被成功地提交并将它们成功传给全局协调器后，全局协调器向提交点发出一个信息，提交点移走悬而未决状态。每个子处理将提交或释放记录上的行级锁。

当子响应发出一个退出信息时，整个事务被回滚。事务是一个要么全有要么全无的进程。为确保数据的一致性，所有的子响应必须提交或回滚主事务。DBA必须可以指出并改正一个不成功的提交以有效地管理这种类型的分布式系统。

注意 当提交不成功时，必须通报执行这个提交的程序或人发生一个错误，应该通过设计产生这个通报。

注意 如果事务在分布式系统的一个结点提交失败，事务在所有的结点回滚。当创建回滚段时，分布式系统中的所有的主机（结点）必须有能力供给分布式系统上预期的最大的事务。

#### 40.3.2 悬而未决事务的处理

网络问题是最有可能导致悬而未决事务的问题。与管理两阶段提交相同，Oracle 7与Oracle8i自动管理在网络故障中检测出的问题。与允许进行灾难恢复处理的自动恢复一样简单，Oracle建议DBA允许悬而未决事务的自动恢复。这个建议有一些例外。由于在网络故障过程中放置在悬而未决事务上的行级锁有可能产生问题，在故障过程中，其他用户试图访问这些行就成为问题，这些数据成为其他用户不可访问的数据。DBA必须要手工管理这个事务以清除这些锁。管理事务的过程称为强制一个事务执行。

如果网络丢失，它不只会显现给DBA，还会显现给所有受那个网络影响的用户集团。其他没有受到网络停机影响的数据库可以继续工作。当用户试图在一个不可访问的数据库上执

行一个DDL时，他们将会发现一个问题，这个DBA也注意到的问题就是锁。有早期SQL\*Net版本经验的DBA熟悉当删除一个连接时，会发生一个端口监督程序锁。这与在分布式事务的过程中网络掉了的情况下发生的锁的类型相同。主要的区别在于如果不管它，分布式事务将在网络恢复时修正它自己。如果网络要关闭很长一段时间，DBA应该准备强制任何产生锁的显著的事务执行。

强制事务执行为回滚或提交的过程很简单。问题是DBA必须在本地强迫事务执行。这意味着如果一个DBA负责管理多个数据库，每个事务将不得不进行手工调查。例如，在底特律的一个用户更新了表中的一行，这需要同时更新米兰与法兰克福中相同的表。如果事务在写任何系统之前失败，DBA应该登录到每个计算机并在每个场所强迫事务执行。这个强制执行可以是提交或回滚的形式。虽然这很麻烦，但幸好这不是经常要做的工作。

当手工强制事务执行时要非常小心。它应该是要么全部要么没有的解决策略。当执行时，DBA应该与其他的DBA协同以确保这个修改与其他的环境匹配或手工验证所有数据库上的一致性。这意味着查询每个受影响的数据库，并确保数据在每个结点是完全相同的。如果没有这样做，会发生数据的不一致性。手工强制事务执行的步骤如下：

- 1) 跟随事务到它失败或开始失败的地方。
- 2) 如果可能，找出一个已完成的事务，或者回滚这个事务，或者提交这个事务。
- 3) 使用这个事务作为强制其他事务执行的模板，开始执行的逻辑位置是提交点。
- 4) 查询DBA\_2PC\_PENDING表，在TRAN\_COMMENT列将有事务从何处开始与做了什么的信息，例如：

```
COMMIT COMMENT 'transaction name';
```

注意 需要使用单引号，事务名是大小写敏感的。

在ADVICE列也有信息，它指示DBA事务在哪里，例如：

```
ALTER SESSION ADVISE COMMIT;
```

DBA\_2PC\_PENDING表的GLOBAL\_TRAN\_ID列提供给DBA事务的ID，这个ID在分布式系统包括的所有数据库中是相同的。如果这个数据库是全局协调器，GLOBAL\_TRAN\_ID的最后一部分将与DBA\_2PC\_PENDING表的LOCAL\_ID相匹配。

- 5) 查看DBA\_2PC\_NEIGHBORS，这个表看起来像这样：

LOCAL\_TRAN\_ID——这个分布式事务的本地事务ID。

IN\_OUT——指出事务是IN还是OUT。

DATABASE——如果事务是IN，这是传送事务的数据库的名字。如果事务是OUT，这是事务正发送到的数据库的数据库链的名字。

DBUSER\_OWNER——如果事务是IN，这是本地用户的名字。如果事务是OUT，这是数据库链拥有者的名字。

INTERFACE——C代表提交请求，N代表准备好或只读。

DBID——Oracle对连接的数据库的唯一的标识符。

SESS#——连接的会话ID。

BRANCH——这个分支的事务ID。

- 6) 使用这个表（数据库与LOCAL\_TRAN\_ID）中的信息，遵循这个线程到另一个数据库。

7) 查询DBA\_2PC\_PENDING表, 如果LOCAL\_TRAN\_ID与GLOBAL\_TRAN\_ID相匹配, 这是一个全局协调器。是DBA应该开始事务线程的地方。如果在这个追踪过程中, DBA遇到了不能解决的问题, 使用它作为显著事务的模板。

8) 要强制执行一个事务, 使用DBA\_2PC\_PENDING表中的TRANSACTION\_ID, 如下所示:

```
Commit force 'transaction id';  
Rollback force 'transaction_id';
```

注意 需要使用单引号。

9) 如果有问题, 将事务回滚, 然后将这个回滚通知用户集团。作为预防, 检查受影响的表的状态以确保数据的一致性。虽然这很麻烦, 但它不是经常发生的, 并且总会保证数据库的完整性。

## 40.4 理解只读快照

快照是主表的一个副本(或子集), 以特定的间隔复制到其他子结点。只读快照是比分布式查询(一个指向远程表的SELECT语句)快的解决方案。除不占网络开销之外, 在远程表上的命中明显降低。如果信息不需要是实时的, 那么快照可以是一个为其他远程结点提供信息的最好的方法。快照通常在一个预定义的时间表内进行刷新。每个快照或快照集可以有它自己的更新计划表。快照可以按照每个结点上的需求复制到不同的位置, 除预定的快照刷新之外, DBA还可以手工刷新一个快照, 使用DBMS\_REFRESH过程完成, 这种刷新的语法为:

```
EXECUTE DBMS_REFRESH. REFRESH ( 'Snapshotname' );
```

注意 使用这种形式, 快照将使用缺省的快速刷新。如果需要完全刷新, 必须将它加到过程调用中。

提供快照的目的与简单的分布式系统的目的相同, 下面是使用快照而不是动态数据库链的一些特殊原因:

改进的性能——快照实际上是一个本地对象。用户可以查询快照而没有与网络相关的开销。

更高的可靠性——如果主结点出了故障, 并不会限制对快照对象的访问。

例如, 由于对美国装饰品定义的依赖, BWC决定分布式数据库不再支持这个业务。装饰器规定的快照被建立。出于商务原因, BWC决定意大利与德国不需要知道其他国家的装饰品定义是什么, 但美国需要知道这两个国家的装饰品定义。所有的装饰品设计被不同的表持有, 所以不需要连接这些表。为减少混淆, 表的拥有者在全世界范围是相同的。

## 40.5 在列级创建一个只读快照

可更新快照现在可以是水平的(选中的行)子集或垂直的(选中的列)的子集。以前的版本只允许垂直子集。

垂直分区允许远程结点需要最小数量的数据, 因而降低连接时间。它还保护快照结点不修改与它们相关的主结点。可以在不影响快照结点的情况下向主结点增加列, 也可以在不影响快照结点的情况下在主结点删除快照没有引用的列。

#### 40.5.1 设置一个快照

创建快照与创建表一样简单。事实上，这正是 DBA所做的——创建一个表的副本，不同之处在于副本成为一个自动的过程而不是人工的过程。在表被拷贝后，快照按规定的时间表进行更新以反映表的改变。快照的规划将基于与创建表所使用的一样的规则。

创建一个快照的语法如下：

```
CREATE SNAPSHOT [schema.]SNAPSHOT [Physical attributes *]  
[TABLESPACE tablespace]  
[LOB_storage_clause]  
[CACHE|NOCACHE]  
[CLUSTER (cluster column{,})]  
[table partition clause]  
[parallel clause]  
[USING INDEX physical attributes clause [LOGGING|NOLOGGING]TABLESPACE tablespace]  
[FAST|COMPLETE|FORCE]  
[START WITH date]  
[NEXT date]  
[WITH PRIMARY KEY|ROWID]  
[USING DEFAULT {MASTER|LOCAL}] ROLLBACK SEGMENT]  
[LOCAL|MASTER ROLLBACK SEGMENT rollback_segment]  
FOR UPDATE AS select_command;
```

例如，如下语句创建一个简单快照：

```
CREATE SNAPSHOT milan_specification  
REFRESH FORCE START WITH SYSDATE next SYSDATE + 7 ;
```

CREATE SNAPSHOT命令的START WITH与NEXT基于SYSDATE。整个数代表一个单独的日期。在前例中，下一次刷新日期是第一次刷新之后的第七天。如果不使用 NEXT，快照将永远不会刷新。如果刷新发生在一个时间段，如每小时或每半小时，它会成为片断的。例如，每小时的刷新率为  $\text{SYSDATE} + 1/24$ ，每半小时的刷新率为  $1/48$ ，每15分钟的刷新率为  $1/64$ 等等。如果总是在星期一进行刷新，语法稍微复杂一些，如下所示：

```
NEXT_DAY(TRUNC(SYSDATE), 'MONDAY')+3/24
```

指定NEXT\_DAY代替SYSDATE，24上面的3代表早晨3点。

使用索引允许创建一个索引，当从快照进行查询时，索引可以改进性能。

CREATE SNAPSHOT命令的FORCE/FAST/COMPLETE是复制的方法。COMPLETE是时间最集中的，它重建这个快照，并通过执行一个 INSERT INTO ... AS SELECT \*完成，COMPLETE也是创建最初的快照的命令。如果表比较大，在初始创建快照之后，应该避免使用这个命令。

FAST是缺省值。它使用日志比较两个快照，并只增加在日志中的改变。

注意 在7版本数据库中，如果快速刷新失败，快照刷新被中断，如果没有手工干涉就无法恢复。在8版本中，如果快速刷新失败，将执行完全刷新。只在快照具有快照日志时才能进行快速刷新。

如果不能执行一个快速刷新的话，就强制执行它（例如，一个被中断的刷新）。这个选项在8版本中被取消。

快照的管理在7版本与8版本中并没有显著的改变。最大的改变是在创建子句中增加了选项，大多数的改变与快照处理并不直接有关，而是由于新增加的对象类型，新的创建表命令与新的创建索引命令。不同之处包括：

LOB\_STORAGE\_CLAUSE与大对象定义有关，大对象的详细信息参见《Oracle8i A Server Application Developer's Guide》。

LOGGING/NOLOGGING指明是否在创建快照时记录重做日志信息。这与 CREATE INDEX NO RECOVER命令相似。它也影响直接装载与直接装载插入。在恢复过程中也会显著影响索引。有关CREATE INDEX的详细信息参阅Oracle8i SQL文档。

CACHE/NOCACHE指明在一个全表扫描之后，快照是否被保存在内存中。更详细的信息参阅《Oracle8i Concepts Guide》中有关内存结构、高速缓存与LRU算法部分。

WITH PRIMARY KEY是在Oracle 8i中的缺省值。通过指明创建一个主键索引快照，主表可以重新组织而不影响快照刷新，涉及使用 PRIMARY KEY快照的限制如下：

在被引用的表中必须有一个主键。必须是在 CREATE TABLE命令中使用 PRIMARY KEY约束。

必须使用主键中的所有列。

**警告** 如果没有在创建快照命令中使用主键，将使用记录的行 ID。如果由于数据库恢复而使主表得到恢复，这会产生潜在的问题。恢复将改变行 ID的值，可能会使快照无效。为确保不产生这样的问题，在任何数据库恢复之后使用完全选项刷新快照。

USING DEFAULT MASTER | LOCAL指明回滚段的类型。DEFAULT将使Oracle进行选择；MASTER将使用远程结点上的回滚段；LOCAL将使用本地数据库的回滚段。

#### 40.5.2 使用快照刷新组

许多快照，虽然与不同的主表相关，可能具有相同的刷新频率，使它们可以组成一个组。虽然Oracle8i为创建这些快照使用了一个点击的方法，但有必要知道在鼠标后面发生什么。

Oracle将这个能力称为API调用。复制对象的操作不是由标准 SQL命令执行的。这些调用通过使用过程执行。有关过程、它们的语法与怎样使用它们的详细信息，参见《Oracle 8i™ Application Developer's Guide》。

### 40.6 快速快照刷新

快照刷新已被优化以支持大的刷新组。有改进的对子查询快照及空刷新（自从上次刷新以来主表没有改变）的支持。一个单独的刷新组现在可以含有 400个快照，在快照组中刷新快照所需要的往返次数也被减少（在 8.0.5版本中第一次增加这个功能）。

#### 40.6.1 确定快照中的潜在问题

在数据库管理过程中，活动前的管理是最正确的方法。要为活动前的管理快照过程，DBA必须能够指出潜在的问题，并设计系统防止这些问题的发生，本节指出了这样的一些问题：

##### 1. 大小

在创建快照的过程中，应该避免一种缺陷。虽然快照只用于查询，但它不是一个静态表。大小应基于表副本的更新率，而不是快照自身。例如，如果 GERMANY\_SPECS表是高度动态的，应该具有一个较大的使用百分比（PCTUSED）以适应更新。表应该调整尺寸以确保不会

产生碎片。

## 2. 不小心删除了对象

一些快照在快照组中。为找与它们相关的工作号，查询 DBA\_REFRESH\_CHILDREN表。这些组可以被删除，然而，必须小心以确保没有同时删除对象。因此，删除任何类型的复制组的方法应该由SQL代码完成，而不是由企业管理器完成。企业管理器使 DBA可以通过点击对象删除它们。

删除刷新组的方法使用这个过程：

```
execute DBMS_REPCAT.DROP_SNAPSHOT_REPGROUP(gname=>'[name of group]'
```

drop\_contents缺省为FALSE。当使用过程时，必须作为一行输入，否则它们不会被调用。

警告 如果设置drop\_contents为TRUE，与该组相关的表也被删除。

## 3. 丢失快照刷新

当一个主数据库不能完成快照的刷新时，在刷新试图停止之前将再试 16次，这16次并不意味着16次刷新间隔。第一次刷新失败后，快照处理将等待一分钟，然后再试。如果失败，时间成倍增长为两分钟，然后四分钟，时间成指数增长直至到达或超过真正的刷新间隔。当发生这个时，它将按刷新率进行。在 16次之后，Oracle将更新USER\_REFRESH表与USER\_REFRESH\_CHILDREN表中的BROKEN列。BROKEN列表明快照刷新发生了错误。不只在这个表中反映这个问题，Oracle SNP进程也在追踪文件与alert.log中指出刷新问题。

减少一个中断快照的可能性的方法是使用 FORCE选项而不是 FAST选项创建快照。FORCE将选择使用FAST，除非发生了一些事情阻止成功的快速刷新；然后它将执行一个完全刷新。使用这个选项的一个缺点是万一是一个非常大的表，刷新将进行很长时间。如果本次刷新仍在进行，又发生了按时间表规定的下次刷新，这将产生问题。

在DBA确定刷新被打断并矫正了中断快照的问题后，快照可以被手工刷新。Oracle8i提供了一些手工刷新快照的方法，一个方法是执行快照的工作。要这样做，在子结点记录日志并执行以下查询：

```
SELECT RNAME,JOB FROM DBA_REFRESH;
```

结果与下面相似：

RNAME	JOB
GERMANY SPECS	142

运行中断快照的工作，工作名与快照名相同（本地/子）。要这样做，执行下面的过程：

```
Execute DBMS_JOB.RUN(142);
```

响应如下：

```
PL/SQL procedure successfully completed.
```

当这个完成时，验证 USER\_REFRESH与USER\_REFRESH\_CHILDREN表中的BROKEN状态不再是Y。如果BROKEN状态仍然是Y，重复上述过程。如果仍不能修复问题，确信没有其他的问题（如丢失连接），它会阻止刷新进行。

为预活动地监控快照处理并阻止另一个中断的快照，比较快照时间与快照日志中的时间。它们应该相匹配，如果不匹配，可能表明导致快照丢失的问题仍没有解决。

在主结点，使用下面的查询：

```
SELECT MASTER,SUBSTR  
(TO_CHAR  
(CURRENT_SNAPSHOTS,'MM-DD-YYYY HH:MI:SS'),  
.....
```

```
1,20)  
TIME FROM DBA_SNAPSHOT_LOGS;
```

从本地/子结点，使用下面的查询：

```
SELECT NAME,SUBSTR  
(TO_CHAR  
(LAST_REFRESH,'MM-DD-YYYY HH:MISS'),  
1,20)  
TIME FROM DBA_SNAPSHOTS;
```

这些时间应该相同。如果不同，需要对快照进行手工管理直至完全解决问题。

#### 4. 控制快照日志增长

快照日志含有主快照的 DML 改变。这决定了向远程结点发送什么。在使用这个日志的所有快照完成后，日志被净化。然而，如果快照有问题，或者其中一个快照不定期地刷新，快照日志会不受控制地增长，这会在两个方面产生问题。DBA 应该设置日志大小以使它可以容纳一定的增长总数。如果快照的稳定性是首要任务，建议使用无限的区间，这会消除达到最大区间数，并填满日志所在表空间的可能性。DBA 应该定期检查这个日志以确保不会发生这种问题。如果无法管理日志的大小，DBA 应该净化日志，这可以由企业管理器工具完成，或者 DBA 可以使用存储过程或 API 调用 DBMS\_SNAPSHOT.PURGE\_LOG 执行，语法如下：

```
Execute dbms_snapshot.purge_log('[master table name]',number,'flag');
```

number 是要清除的历史快照的数量。如果 DBA 想要清除所有的快照日志，这个数会非常大。当这样使用时要小心，如果清除了所有的日志，快照将不得不进行完全刷新。如果表非常大，这会是一个问题。

注意 标志是一个可覆盖的值。如果设为 DELETE，或 number 设为 0，将从最近最少刷新的快照删除日志。

### 40.6.2 理解快照的限制

一个大表快照的初始创建将可能超出系统的能力。例如，如果一个有几千万条记录的大表的快照不得不通过不稳定的连线进行复制的话，最好执行一个脱机实例化。要完成这项工作，需要执行一些步骤。必须小心进行这些处理，如果可能，在一个有很大存储空间的测试环境中进行。步骤如下：

- 1) 在产品数据库中，为每个主表（将在远程结点具有快照的表）创建一个快照日志。
- 2) 使用一个新的模式，并且在测试数据库中，创建一个指向含有主表的产品数据库的快照，名字必须是唯一的，理想上应该是远程结点上将使用的名字。
- 3) 如果这个处理必须在产品数据库中完成（不建议这样做），在 CREATE SNAPSHOT 语句中的链接将指向当前数据库，Oracle 将这个能力称为回环链接。
- 4) 导出这个新的模式，与新的快照拥有者相同的模式将执行这个导出。
- 5) 删除这个新建的快照，确保这是手工完成的（使用 API 调用）。如果不是手工完成的，确保只创建快照，而没有创建相应的对象。

在远程结点创建一个快照：

- 1) 创建一个空的快照组，要支持下一步的过程调用，这一步必须进行。
- 2) 使用下面的过程：

```
DBMS_OFFLINE_SNAPSHOT.BEGIN_LOAD  
(gname='[groupname]'.sname=>'snapshotname',)
```

这将为将被导入的数据创建一个快照“壳”。

3) 导入这个快照的基表，由前缀为 SNAP\$ 的表标识。

4) 导入完成后，使用过程 DBMS\_OFFLINE\_SNAPSHOT.END\_LOAD 暗示系统导入完成。实现这个处理的最简单的方法是热备份处理，其中执行命令 ALTER TABLESPACE BEGIN-  
BACKUP。

注意 与不能在 LONG 数据类型上使用 SELECT 创建表一样，快照也不能支持这个 SELECT 语句。

### 40.6.3 调整快照

由多个表连接组成的复杂快照会严重降低性能。完全测试与调整用于创建快照的查询是必不可少的。这包括在创建快照之前使用 explain plan 与 TKPROF。如果需要，在子结点创建一些单独的表快照，然后创建一个基于物理快照的本地视图。性能将会显著增强，网络负载将降低。

为在库缓存中保留用于刷新数据库的包，最好将这些包钉在内存中。这将帮助阻止从内存中移走包。为了钉住一个包，调用这个包，然后使用 DBMS\_SHARED\_POOL 包。在钉一个包之前，必须先引用它。最简单的引用包的方法是重新编译它，语法如下：

```
ALTER PACKAGE DBMS_SNAPSHOT.I_AM_A_REFRESH COMPILE;
```

响应为：

Package altered.

现在，使用下面的语句打包：

```
Execute DBMS_SHARED_POOL.KEEP('DBMS_SNAPSHOT');
```

响应为：

PL/SQL procedure successfully completed.

这将改进性能，但可能不会显著增强性能，然而，这确实有帮助——特别是对那些经常刷新的快照。

可以因性能被钉的其他包为：

- DBMS\_REFRESH
- DBMS\_REPUTIL
- DBMS\_REPCAT
- DBMS\_DEFER

这些包的详细信息，参阅《Oracle 8i Server Replication——Replication Manager API Reference》。

#### 初始化复制包

通过将 PL/SQL 复制包内在化与优化快照刷新，会实现显著的性能改进。

从版本 8.0 开始持续地趋势是，将更多的复制代码移入数据库引擎内。PL/SQL 生成的用于在远程结点复制事务的包已被内在化。这允许复制事务更为有效地在远程结点应用，因为包没有生成，结点可以被更快地实例化。内部的包也更安全，因为它们是不会被乱动的。

40.6.4 使用快照的初始化参数

init.ora表含有一些直接影响快照处理的参数，这些参数在表 40-2中定义。

表40-2 快照的初始化参数

参 数	描 述
JOB_QUEUE_INTERVAL(1-3600)	快照处理被唤醒的间隔。必须小心以确保这个值没有设得过高，不会妨碍快照自己的时间间隔
JOB_QUEUE_PROCESSES (0-36)	快照处理进程数的限制。通常，1就足够了，除非有几个快照，或可能会影响快照进程的大的快照