

第14章 Oracle 8i 附加主题

本章要点：

新的ROWID

口令管理的增强

恢复管理器概念

高级队列概念

约束、国家语言支持和 sys 安全性

在 Oracle 8.x 中有几个新特性或增强。这些包括所有的增强类型；然而，实现的增强主要包括在第 20 章中。本章主要包括的有新的 ROWID 格式、新的口令管理功能、恢复管理器和高級队列；也简要地包含国家语言支持、约束和 sys 安全性的改进。

14.1 新的ROWID

在 Oracle 8.x 中，新的 ROWID 格式已被改变，主要是为了适应分区和对象的新特性以及增加数据库限制（参见第 18 章），提供比 Oracle 7.x 更多的数据库文件和表空间。ROWID 是 Oracle 中所有存储寻址的基础，因此这个改进很重要。

ROWID 被用做在规则的 Oracle 索引（B-树）中唯一识别一行，并在整个 Oracle 8 的内核中使用。在 Oracle 7.x 中，ROWID 包括文件号、块号和行号。它仅用一个号一致性地识别文件。换句话说，在 Oracle 7.x 中文件号是绝对的。在 Oracle 8.x 中，文件号是相对的。一个 Oracle 8.x 数据文件的文件号由两部分组成：

绝对部分在数据库内（和在 Oracle 7.x 中一样）唯一，由 dba_data_files 表的 FILE_NO 列代表。

相对部分在表空间中唯一，由 dba_files 表中的 RELATIVE_FNO 代表。

因为文件号对表空间是相对的，新的 ROWID 也必须存储段的一些指示。在 Oracle 8.x 中，这是用来识别表或分区的数据对象号。通过确定这个段，然后 Oracle 8 能确定包含的表空间，和使用 dba_tablespaces 一样。

能够确定数据对象号，例如从 dba_objects 的 DATA_OBJECT_ID 列中确定。这个数据对象号不是对象标识符（参见第 10 章），它被包含在每一块中，并且是唯一的。当一个表被截断或分区被移走时，数据对象号的版本增长。使用这种方法，Oracle 8 能验证在块中的号匹配（也就是，是正确版本），也能够用它与回滚段中的块做比较来保证版本的准确性。

新的 ROWID 格式使用基于 64 位的编码，它的字符宽度是 18。这就意味着在任何给定的 Oracle 8 数据库中有 64^{18} 个可能的行。新的 Oracle 8.x ROWID 组成见表 14-1。

表 14-1 新的 ROWID 组成

数据对象号	相对文件号	块号	位置号
000000	FFF	BBBBBB	SSS

从左到右，前六个字符代表数据对象号，接下来 3 个字符代表在表空间中的相对文件号，

再接下来的6个字符代表在一个文件中的块号，最后的3个字符代表块中的位置段（行）号。与Oracle 7.x对比，真正的不同之处是数据对象号和相对文件号。

在表14-2中，列出了Oracle 7.x ROWID的格式。

表14-2 旧的ROWID组成

块 号	行 号	文 件 号
BBBBBBBB.	RRRR.	FFFF

从左到右，前8个字符代表在文件中的块号，接下的4个字符代表块中行号，最后4个字符是绝对文件号。除了在分区表中涉及到全局索引（因为分区能跨多个分区表空间）外，这种类型的ROWID在多数情况下是足够的。

如何能够概括在Oracle 7.x的ROWID和Oracle 8.x新的ROWID之间的不同呢？表14-3会对你有些帮助。

表14-3 ROWID版本比较

版 本	描 述	字 节 数	显 示	文 件 号
7.x	限制	6	2点	绝对
8.x	扩展	10	无点	相对

相对表空间寻址是Oracle VLDB支持的基础。Oracle 8使用10个字节来存储这个新的ROWID，而Oracle 7.x使用6个字节。当相对ROWID已足够时，限制ROWID被使用。当绝对ROWID被要求（如在分区表中用全局索引）时，扩展（完全）ROWID被使用。对于无分区表上的无分区索引、在分区表上的相等分区索引、跨块的链接移植指针，使用限制ROWID已足够。扩展ROWID用于分区表上的全局索引、内核的使用和存储ROWID的格式。

当从Oracle 8获取Oracle 7 ROWID时，ROWID以限制的格式出现，如同它们在Oracle 7.x中一样。但是，当从Oracle 7.x获取Oracle 8.x ROWID时，你需要使用DBMS_ROWID包来解释扩展格式。Oracle 7的表可以被导出到Oracle 8中，但是包含有新ROWID格式的Oracle 8.x表不能被导出到Oracle 7中。更进一步，如果ROWID被以任何存储格式保持，在它们导入之后，需要重新计算，因为它们的内容将是过时的，需要更新。

只在应用是部分存储或获取ROWID行时，发生应用可移植性问题。另一方面，在整体上，使用ROWID的这些应用将受影响。当Oracle 7表被导出或被移植（使用移植工具）到Oracle 8时，存储ROWID的列的宽度自动地变宽以适应新的比较宽的Oracle 8.x ROWID。前述的DBMS_ROWID包由dbmsutil.sql脚本创建，被catproc.sql脚本调用。表14-4列出了DBMS_ROWID包的函数和返回值。清单14-1显示了从限制ROWID转换到扩展ROWID的例子。

表14-4 DBMS_ROWID 包的函数

功 能	返 回
rowid_create	一个新的ROWID
rowid_info	类型和组成
rowid_type	类型（0=限制，1=扩展）
rowid_object	数据对象号（段）
rowid_relative_fno	相对文件号
rowid_block_number	块号
rowid_row_number	行（位置）号

(续)

功 能	返 回
rowid_to_absolute_fno	绝对文件号
rowid_to_extended	扩展ROWID
rowid_to_restricted	限制ROWID
rowid_verify	0=可以被扩展, 1=不能被扩展

清单 14-1 从限制ROWID转换到扩展ROWID的例子

```
SQL> select * from mytable where mykey=12;
mykey myROWID          myvalue
-----
12 00000001A.0011.0009      102
SQL> update mytable set myrowid=
1 dbms_rowid.rowid_to_extended(myrowid, 'JOHN', 'MYTABLE', 0)
2 where mykey=12;
SQL> select * from mytable where mykey=12;
mykey myROWID          myvalue
-----
12 AAAAbCAACAAAAAbAAAA      102
```

rowid_to_extended函数接收旧的（限制的）ROWID、拥有者、表和转换类型（conversion_type）。这个转换类型既可是 0 用于存储（内部）转换，又可是 1 用于显示（外部）转换。它返回一个新的（扩展）ROWID。

14.2 口令管理的增强

在Oracle 8中提供的口令管理的增强包括帐户锁定、时效和到期、历史机制和复杂的验证（激活检查）。为了启用口令管理，运行utlpwdmg.sql脚本。你能建立环境资源文件并将用户指定给它们。然而，它不能够像在 init.ora文件中那样用resource_limit参数打开和关闭对资源的限制，口令限制总是被强迫执行。

用帐户锁定，在指定数量的登录失败后，企图再次登录时，Oracle 8自动锁定帐户。可以基于用户或组设定这个数。在超过一定的时间间隔或者手动解锁之后，帐户锁定被打开。可以根据需要手动锁定，在这种情况下，只能使用手动解锁（不是通过一个时限解除封锁）。

使用口令时效和到期，你能指定一个截止周期或活动时间，它对应于口令保持多长时间的有效性。你也能指定一个宽限期来使用户有时间在口令到期时改变它的口令。这些特性可以被建立在用户或组上。例如 A组可能有90天期限加上5天的宽限，而B组可能有30天期限再加上1天宽限，另外用户C可能有单独的期限和宽限，分别是60天和10天。

使用口令历史机制，一个用户不能再用过去的时间间隔中使用过的任何口令，这个时间间隔被指定在该用户的环境资源文件中。另外，要记住的重要的一点是，和资源限制不一样，它和其他口令限制总是被强迫执行（除非被修改）。

激活口令检查是通过一个缺省的 SYS PL/SQL函数提供的。你能用你自己的增加改进的 PL/SQL函数替换它，缺省的函数提供复杂的验证来检查每一个口令，如下：

- 1) 至少4个字符长度。
- 2) 不等于用户ID。
- 3) 至少有以下每种字符中的一个：字母、数字、标点符号。

4) 至少有3个字符不同于老口令。

你自己的口令检查函数在 SYS图表中创建，依附在用户或组的环境资源文件中，并且必须附加如下声明：

```
myfunction (
  p_userid IN varchar2(30),
  p_new_password IN varchar2(30),
  p_old_password IN varchar2(30) )
return boolean;
```

CREATE PROFILE 和 ALTER PROFILE 语句处理所有前述的口令限制的设置，对 CREATE PROFILE 语句的缺省口令设置如表 14-5 所示。

表14-5 CREATE PROFILE的缺省口令设置

设 置	缺 省 值	单 位
failed_login_attempts	3	次
password_life_time	60	天
password_reuse_time	180	天
password_reuse_max	无限制	个
password_lock_time	1	个
password_grace_time	10	天
password_verify_function	缺省	函数

这些设置中的大部分已经在前面的段落中解释过，现在来讨论一些可能不太清楚的地方。password_reuse_max 代表在一个口令能够再被使用之前，改变口令的最大个数。缺省值是无限制，但是口令再使用还受 password_reuse_time 控制。这两个参数都能被设置为无限制；然而实际上仅有一个能被设置为数值。在这方面，它们之间的设置是互斥的。

password_lock_time 指定在特定次数的企图登录失败之后，用户帐户被锁定的天数。用缺省的设置，如果用户三次登录失败以后，用户帐户将被锁定一天。最后，缺省的 password_verify_function 的实际名字是简单的 verify_function。

清单 14-2 给出一个例子，这个例子创建一个环境资源文件，设置口令的限制，然后创建一个用户以拥有这个环境资源文件。

清单 14-2 为用户创建口令环境资源文件的例子

```
SQL> create profile secure90 limit
  2 password_life_time 90
  3 password_reuse_time 366
  4 password_grace_time 5;
SQL> create user john
  2 identified by all4one?
  3 default tablespace scratch
  4 temporary tablespace temp2
  5 quota 10M on scratch
  6 password expire
  7 profile secure90;
```

注意，CREATE USER 语句在为新用户帐户 john 预定截止日期之前，CREATE USER 和 ALTER USER 语句将你的定制的环境资源文件用于一个特定的用户。你也可以用组来替代这个用户。还有 ALTER USER 语句被用来手动加锁和解除一个帐户的锁，如下所示：

```
SQL> alter user john
  2 account lock | unlock;
```

DBA_USERS和USER_USERS包含反映这些新的口令管理特性的新列，如表 14-6所示。

表14-6 新的用户口令管理特性

新 列	意 义
account_status	锁定、截止或打开
grace_data	改变的日期+password_grace_time
lock_data	锁定的日期
expiry_data	创建的日期+password_life_time

DBA_PROFILES还列出了口令设置，并包含一个新的 resource_type列，指明内核或口令。一个新的user_password_limits视图包含resource_name和limit列。

14.3 恢复管理器概念

恢复管理器（RMAN）是对Oracle 8备份、恢复和重建进行管理的一段软件。RMAN可以是独立的或分布的。RMAN对Oracle 8服务器使用PL/SQL接口，它调用三个主要函数：BACKUP、RECOVER和RESTORE。RMAN使用一个称为恢复目录的知识库。在 8.0.4中，第三方存储子系统如EMC/Epoch、Legato、IBM等等能够被集成到 RMAN解决方案中。例如，磁带驱动器以自动方式使用，能够以并行方式使用诸如那些来自 Exabyte的磁带单元。

RMAN结构的四个基本块是RMAN接口（一个OEM applet图形接口或命令行解释器，它能在OEM 1.4或更高版本中获得）、BACKUP/RECOVER/RESTORE服务器函数、恢复目录和存储子系统（操作系统的、第三方的或两者都是）。

为什么要使用RMAN管理你的Oracle 8备份？它能自动执行许多管理功能和职责。备份能发生在多级上：数据库、表空间或者数据文件级。数据文件能够被特指为基于用户定义容限的备份。最重要的是，真正的增量备份是可能的；换句话说，在任何级别的备份能够执行只对已改变的块备份；更进一步说，未被使用的块不被备份。备份能够很容易地建立时间表和控制。存储子系统处理更容易管理，它的操作可能被并行化。另外，RMAN能够与已经建立起来的第三方厂商硬件或软件集成。

RMAN有两个主接口：命令行解释（CLI）接口或OEM图形用户接口（GUI）。CLI接口是一个与SQL*Plus相似的应用，能够用交互或非交互的方式执行。GUI在OEM中被集成，提供能够与第三方应用（例如存储子系统）相链接的API库。

14.3.1 恢复目录

恢复目录由catrman.sql脚本创建，通常在专用数据库中对它进行存储，它十分像 OEM知识库。恢复数据库是存放恢复目录的数据库，目标数据库是任何 RMAN要备份的目标数据库的应用或用户数据库。恢复目录必须和你的其他数据库一起备份。为备份恢复数据库，你必须反转它和一个目标数据库的角色，一个目标数据库于是成为恢复数据库的备份数据库，它是暂时的目标数据库，另一些目标数据库是恢复数据库，这意味着它至少有两个恢复目录——一个是为所有用户数据库的，另外至少有一个为恢复目录自身的目录。

提示 你应把主要的RMAN知识库放入分开的（恢复）数据库中，并放在与你的核心应用（目标）数据库分开的机器上。另外你能只为处理重要的 RMAN仓库在一个应用数据库中创建一个RMAN知识库。用这种方法，你能失去任何数据库并能恢复它，包括

恢复数据库自身。

恢复目录维护在它的数据库仓库中的信息，这些信息包括数据文件备份设置、归档日志备份设置、备份块、数据库文件拷贝、归档重做日志、归档重做日志拷贝、目标数据库的控制文件信息（也就是它们的物理结构）和通常可获得的备份命令。

RMAN静态地存储关于所有它的目标数据库的控制文件信息。当备份操作发生时，RMAN与需要备份的服务于目标数据库上的 Oracle 8服务器进行通信，在它的核心里已经有了备份功能。RMAN初始化和管理工作，但是实际上每一个目标数据库备份它自己。因此，当任何目标数据库结构改变时（例如，一个数据文件被增加或重命名），在下一个备份操作发生之前，除了必须从数据库的数据字典中进行查询（这意味着目标数据库必须打开）的回滚段消息外，几乎所有对更新恢复目录必要的信息都能在目标数据库的控制文件中找到。

然而RMAN能够在没有恢复目录时进行操作，当你仅有一个或少量几个备份数据库，或者表空间按时间点恢复（PTR）不是本质的时，这是适合的。为什么你不能有 PTR？回滚段信息应该存储在恢复目录中，它被从目标数据库中查询，但是没有恢复目录，这儿可以没有 PTR。对无恢复目录操作的其他限制包括不能使用存储脚本（它应被存储在目录中）并且当控制文件被丢失或毁坏时自动恢复不一定可行（只是部分如此）。

14.3.2 RMAN命令和功能

恢复管理器依赖于（目标）服务器备份功能，这个功能驻留在包括所有必须的备份、恢复和复原子程序的数据库包中。RMAN的前期产品，External（或Enterprise）Backup Utility（EBU）仅能调用第三方工具。集成是最高程序层。现在 RMAN能通过API功能控制操作系统（OS）和第三方存储子系统。集成是最低的功能层。

RMAN命令包括注册一个目标数据库、重置目标数据库的状态（甚至对以前备份过的版本）、当目标数据库改变时更新恢复目录（甚至从前备份过的控制文件的状态）的能力。目录同步是周期性更新目标数据库信息：所有控制文件和回滚信息。这是重要的，不仅因为可能有结构性变化，而且因为日志文件现在记录在控制文件里。在 Oracle 8.x中，控制文件改进为追踪日志文件转换和归档重做日志文件拷贝。因此同步变得十分重要。

RMAN提供直接使用 BACKUP、RECOVER和RESTORE命令或在脚本中嵌入的能力。RMAN脚本被RUN命令调用。CREATE SCRIPT命令被用来创建脚本（把它存储在恢复目录中）。为了安全起见，每个你备份的脚本都保存一个操作系统拷贝（例如用于丢失恢复目录的情况）。REPORT和LIST命令提供恢复目录信息和RMAN状态的日志输出，REPORT报告基于用户容量相关的冗余、增量或时间可能需要备份文件的数据库文件；备份文件可能被删除备份文件不能恢复。LIST列出关于在备份集或复制中用户特定数据文件的信息，部分或全部匹配查找准则。

14.3.3 RMAN术语

备份集或者是数据文件（可能还包括控制文件）或者是归档日志的集合。换句话说，多个文件被写入单个备份集中。它与单个文件简单地被复制为镜像是相对地。在备份集中被备份的文件在能够被使用之前必须被恢复（和提取）。而镜像能够直接被使用，只是简单地重新命名为丢失的数据文件。备份集支持完全或增量备份。

并行化能够被指定以匹配可获得磁带驱动器数。备份集能够从磁盘到磁带进行，在恢复

时相反执行。数据文件能够被保存在包括多个磁带的备份（多磁带卷）之中，并被多路复用，在每个数据文件中有用户指定的块数。控制文件块首先被完全写入，如果被包括，它不能在使用数据文件块时交织使用。备份集能够被写入磁带或磁盘中。

备份块是属于备份集的单个文件。这个文件能包含来自多个数据文件的块并能够被指明作为操作系统文件或磁带卷。备份集是它的所有备份块之和。

镜像复制仅只能够写入到磁盘，它与源文件形成一对一的对应。如果所有的镜像文件的总和包含你的整个数据库，那么你有一个完全的冷备份。你也能注册外来备份，那也就是说，这些镜像文件可以不是 RMAN 创建的。

完全备份包含要备份的所有特定的数据块。增量备份仅包含自上次备份以来被修改的块。累加增量备份是可能的。但是它们浪费空间和时间，因此它们的优点被量的积累所消除。你也可以做一个完全备份作为基准，它被所有随后的增量所忽略。多级增量就像 UNIX ufs 备份规划。0级代表完全备份，你能指定最高 7级的增量。例如，2级可能代表周增量，1级代表日增量。用这种规划，增量仅备份自上次同级或更低级备份以来的增量。例如，2级备份仅备份自上次2级、1级或0级备份以来的改变。多级增量增强了恢复，因为你从任何给定的级仅需要恢复最多一个增量。

标识符是能够被指定到备份集或镜像文件的字符串（最多 30个字符）。标识符是被存储子系统用来帮助用户记忆一些存储单元（例如磁带卷或磁盘分区）的符号名。它的用途与 RMAN 没有区别。标识符能够跨越备份集或镜像文件。标识符不必对每个备份集或镜像文件唯一。这能有助于分组某些备份集，在感觉上组成一个整体确定的备份集，例如一个表空间。

备份和恢复的并行化操作由 RMAN 在内部处理，它建立多个并行用户会话，每个设备一个会话。RMAN 在用户会话内多路复用独立命令的发生。因此尽管在 RMAN 内语句执行是串行化的（如 SQL*Plus），通过虚拟的多用户会话可以实现并行化。尽管一个用户能够指定分组，当多路复用它们到备份集时，数据文件被 RMAN 分组。除 SYSTEM 表空间外，RMAN 支持任何表空间的 PTR。为了实现后者，Oracle 要求数据库仅用技术支持助手复制和操作。

备份本质上要求数据库被打开（以获得回滚信息），可是恢复仅要求实例被启动。备份和恢复都检查记录和报告块的毁坏。它们也为破碎块探测进行校验和。破碎的块是当它被修改时，正好赶上被备份操作读取，因此它必须被重读来得到一致性版本。

在 Oracle 8.x 中，控制文件随日志转换和归档文件拷贝增长。可是这个增长能够通过设置 init.ora 文件中的参数 control_life_record_keep_time 进行控制，这个参数被表示保持最旧的可重用条目或记录的天数，而 Oracle 7.x 没有可重用条目。如果你设置这个参数为 0，无论你何时需要，条目都可再用，它们无时间限制。

所有保存 RMAN 备份和恢复信息的数据字典视图都以 V\$BACKUP_* 作为前缀，如下所示：

```
V$BACKUP_CORRUPTION
V$BACKUP_DATAFILE
V$BACKUP_REDOLOG
V$BACKUP_SET
V$BACKUP_PIECE
V$BACKUP_MEMBER
V$BACKUP_RESTORE_STAT
V$BACKUP_PARMS
```

查看这些视图和V\$BACKUP_LOG以及V\$DATAFILE可监控RMAN的活动。

14.4 高级队列概念

在分布式环境中主要有两种主要类型的执行：立即（同步）或延时（异步）。在大多数情况下，立即执行是在许多分布式系统中被采用的。但是在其他一些情况下，例如基于复制的分布式系统，延时执行是适当的。队列是延时执行机制。队列是先进先出（FIFO）或先来先服务（FCFS）数据结构，首次到来的事情是首先得到服务并退出。可是，存在纯队列的变体，它们全都或多或少地基于优先级这一概念。在队列中的一个任务单元称之为一个工作。一个工作在队列中等待被服务或被处理。即使系统失败，持续队列也能保证在队列中的所有工作将在系统成功重新启动后被处理。工作控制和任务流管理是两个很适合队列化的领域。事务处理（TP）监控器和面向消息的中间软件（MOM）依靠或者模仿消息队列功能。Oracle 8.x提供高级队列（AQ）能力而无需第三方软件。

消息是能被处理的最小工作。和网络包一样，它包括用户数据和控制信息。单个事务能创建（产生）或处理（消耗）多个消息。ENQUEUE创建消息，DEQUEUE使消息被处理。消息队列是消息的队列或消息的集合。由于不同的原因，多消息队列能够被创建，例如从功能上分开一些消息任务、分开独立的消息活动和可能增加进程的吞吐量。消息表是用来存放多消息队列的Oracle表。

Oracle 8.x高级队列的一些比较重要的特性包括如下：

消息排列和优先级。

对象作为用户数据。

响应和异常队列。

事务性和非事务性消息。

一个消息的多个接收者。

消息分组。

Oracle高级队列的许多用户功能管理功能是通过系统包 dbms_aq和dbmsaqadm提供的。dbms_aq包含两个基本过程enqueue和dequeue。用户必须被授予aq_user_role。消息的payload是它的用户数据。看一看enqueue和dequeue声明。

```
enqueue (
queue_name      IN      varchar2,
enqueue_options IN      enqueue_option_t,
message-properties IN    message_properties_t,
payload         IN      <object_type | raw>,
msgid          OUT     raw )

dequeue (
queue_name      IN      varchar2,
dequeue_options IN      dequeue_option_t,
message-properties OUT   message_properties_t,
payload         OUT     <object_type | raw>,
msgid          OUT     raw )
```

因为payload能被指定为一对象类型，它从本质上讲可以是任何对象类型，它的范围很宽（参考第10章）。enqueue_options、dequeue_options和message_properties全部都是预定义的记录类型。queue_name唯一地识别一个队列。msgid唯一地识别一个消息。请参考Oracle 8.x文献，那里包括所有enqueue和dequeue选项的详细说明。看一个简单enqueue和dequeue调用：

```
dbms_aq.enqueue (  
  queue_name => 'john.jq1',  
  payload => employee_type  
  msgid => msgh );  
  
dbms_aq.dequeue (  
  queue_name => 'john.jq1',  
  payload => employee_type  
  msgid => msgh );
```

用这个enqueue调用，一个消息有msgid、msgsh和指定为employee_type对象的page load，这个消息被放进 john拥有的队列表 jq1中。后续的 dequeue调用处理同一个消息。所有 enqueue_options、dequeue_options和message_properties缺省值被采用。同时，注意赋值操作符=>的使用。

提示 当任何消息有基于时间相关的属性（例如延时、截止或保持间隔）时，要求有一个AQ计时器进程或时间管理器（tm）。设置init.ora参数AQ_TM_PROCESSES为1，这将启动一个时间管理器进程。当前，它仅能被置为1；但是在未来的版本中，它可能增加为使能多个时间管理器。

DBMS_AQADM包含用于管理创建和控制队列和队列表以及管理订阅者和启动 /停止时间管理器进程的子程序，你必须有AQ_ADMINISTRATOR_ROLE以执行以下这些过程：

```
CREATE_ | DROP_ QUEUE_TABLE  
CREATE_ | DROP_ | ALTER_ | START_ | STOP_ QUEUE  
START_ | STOP_ TIME_MANAGER  
ADD_ | REMOVE_ | QUEUE_ SUBSCRIBER(S)
```

SYS必须首先执行grant_type_access过程以给AQ管理者对AQ对象类型的访问权。当需要时AQ管理者能更进一步授权给用户，看一看简单创建和删除队列表的例子：

```
dbms_adqam.create_queue_table (  
  queue_table => 'jq1',  
  queue_payload => employee_type );  
  
dbms_adqam.create_queue_table (  
  queue_table => 'jq1');
```

采用缺省值时，这些语句可以十分简单。但是和创建队列处理队列操作一样，在DBMS_AQADM过程的执行中可以指定许多选项，特别是处理产生的过程。作为一个例子，一个最后关键概念涉及订阅者。一个队列可以允许每个消息有多个接收者（这是队列表创建时的一个选项），可以处理来自这种队列的消息的用户称为订阅者（subscriber）。DBMS_AQ用户和DBMS_AQADM管理包的所有可用选项的细节已超出这本书的范围。但是可以说 Oracle在它的数据库引擎中集成了很强壮、也复杂的队列机制。

DBA_QUEUE和USER_QUEUE表包含表队列信息，DBA_QUEUE和USER_QUEUE包含队列信息。AQ\$<queue_table>视图对于每个队列表可用。

14.5 约束、国家语言支持和SYS安全性

在Oracle 8.x中约束检查能够被拖延直到事务结束前，也就是服务器只在提交时检查约束是否被满足。如果约束在事务期间被违反，这是没有问题的。如果在提交时违反，整个事务被回滚。

与之相对的是立即约束检查，即服务器在每个语句结尾检查约束。如果语句违反约束，

这条语句被回滚，不是整个事务。

还有，替代使约束无效或有效，它们能够被强迫。一个强迫约束强加于所有新修改语句，但是忽略当前已存在的数据（它实际上可能违反约束）。尽管可能不是一个好主意，Oracle 8.x现在可以使用非唯一索引使唯一性约束和主键约束可以被处理。

用于国家语言支持（NLS）的NCHAR列是新的。它是一个定长字符、多字符数据类型，能容纳最多2000个字符。NVARCHAR2与varchar2的关系就与同NCHAR与char的关系一样。但是NVARCHAR2能容纳最多4000个字符。最后，为了存储和提取NCHAR和NVARCHAR2数据类型，实际值用字符N作为前缀，不用引号。

在Oracle 8.x中，除非你属于OSDBA组或你有SYS口令/内部口令，否则如果没有SYSDBA身份，你将不能链接到SYS图表中。这些有使用ANY限定（例如SELECT ANY TABLE）的系统权限的用户，在Oracle 8.x将不能像在Oracle 7.x中那样有权访问SYS图表。在Oracle 8.x中必须有SELECT_CATALOG_ROLE或EXECUTE_CATALOG_ROLE才能选择或执行SYS对象。