Oracle 数据库培训手册

— DBA

文档作者:

创建日期:

确认日期:

控制编码:

当前版本: 1.0

文档控制

Part I: Oracle 数据库基础

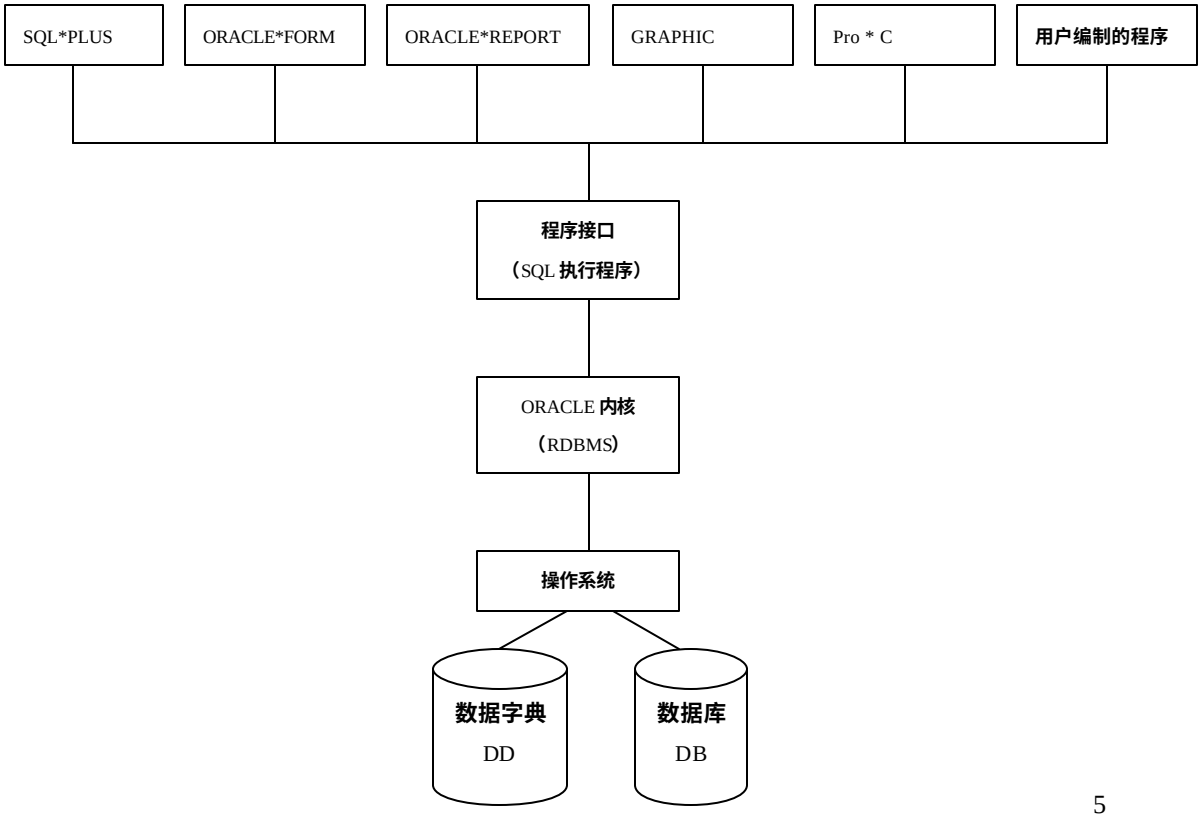
Chapter 1　Oracle 数据库概述

单元培训目标

- **初步了解** Oracle**产品结构及组成**
- **了解** Oracle **数据库的特点**
- **初步了解** Oracle**数据库的体系结构**
- **了解** Oracle **公司的其他产品**
-

§1.1 Oracle 产品结构的组成

ORACLE 系统是由以 RDBMS 为核心的一批软件产品构成，其产品结构轮廓下图所示：



§1.2 Oracle 产品结构的特点

ORACLE 公司于 1979 年，首先推出基于 SQL 标准的关系数据库产品，可在 100 多种硬件平台上运行（所括微机、工作站、小型机、中型机和大型机），支持很多种操作系统。用户的 ORACLE 应用可方便地从一种计算机配置移至另一种计算机配置上。ORACLE 的分布式结构可将数据和应用驻留在多台计算机上，而相互间的通信是透明的。1992 年 6 月 ORACLE 公司推出的 ORACLE7 协同服务器数据库，使关系数据库技术迈上了新台阶。根据 IDG（国际数据集团）1992 年全球 UNIX 数据库市场报告，ORACLE 占市场销售量 50%。它之所以倍受用户喜爱是因为它有以下突出的特点：

- 支持大数据库、多用户的高性能的事务处理。ORACLE支持最大数据库，其大小可到几百千兆，可充分利用硬件设备。支持大量用户同时在同一数据上执行各种数据应用，并使数据争用最小，保证数据一致性。系统维护具有高的性能，ORACLE 每天可连续 24 小时工作，正常的系统操作（后备或个别计算机系统故障）不会中断数据库的使用。可控制数据库数据的可用性，可在数据库级或在子数据库级上控制。

- ORACLE遵守数据存取语言、操作系统、用户接口和网络通信协议的工业标准。所以它是一个开放系统，保护了用户的投资。美国标准化和技术研究所（NIST）对 ORACLE7 SERVER 进行检验，100%地与ANSI/ISO SQL89标准的二级相兼容。

- 实施安全性控制和完整性控制。ORACLE为限制各监控数据存取提供系统可靠的安全性。ORACLE实施数据完整性，为可接受的数据指定标准。

- 支持分布式数据库和分布处理。ORACLE为了充分利用计算机系统和网络，允许将处理分为数据库服务器和客户应用程序，所有共享的数据管理由数据库管理系统的计算机处理，而运行数据库应用的工作站集中于解释和显示数据。通过网络连接的计算机环境，ORACLE将存放在多台计算机上的数据组合成一个逻辑数据库，可被全部网络用户存取。分布式系统像集中式数据库一样具有透明性和数据一致性。

- 具有可移植性、可兼容性和可连接性。由于 ORACLE 软件可在许多不同的操作系统上运行，以致 ORACLE 上所开发的应用可移植到任何操作系统，只需很少修改或不需修改。ORACLE 软件同工业标准相兼容，包括许多工业标准的操作系统，所开发应用系统可在任何操作系统上运行。可连接性是指 ORALCE 允许不同类型的计算机和操

作系统通过网络可共享信息。

§1.3 Oracle 数据库体系结构

要了解 Oracle 的体系结构，必须先了解两个基本概念：数据库和实例。下面两节将详细描述这两个基本概念及其在Oracle 中的实现。

§1.3.1 数据库

数据库是一个数据集和。Oracle 能够提供按照一致性方式定义的定义模型（称作关系模型）存储和访问数据的方法，因此 Oracle 被认为是一种关系数据库管理系统（RDBMS)。对“数据库”一词的大多数引用不仅是指物理的数据，也指下面描述的物理、内存及进程对象的组合。

数据库中的数据存储在表中。关系表由列定义，并赋予一个列名。数据在表中以行的方式存储。表可以相互关联，数据库可用来实施这些关联。

除了按关系格式存储数据外，Oralce 支持面向对象的结构。对象既可以与其他对象建立关系，也能包含其他对象。

无论是采用关系结构还是面向对象的结构，Oracle数据库都将数据存储在文件中。在其内部，数据库结构提供数据对文件的逻辑映射，允许不同类型的数据分开存储。这些逻辑划分被称为表空间。

§1.3.1.1 表空间

表空间是数据库的逻辑划分，每个数据库至少有一个表空间。为便于管理和提高运行效率，可以使用一些附加表空间来划分用户和应用程序。例如：USER 表空间供一般用户使用，RBS 表空间供回滚段使用。一个表空间只能属于一个数据库。

§1.3.1.2 文件

每个表空间由同一磁盘上的一个或多个文件组成，这些文件叫数据文件。一个数据文件只能属于一个表空间。在 Oracle7.2 种，数据文件创建后可改变大小。创建新的表空间需要创建新的数据文件。

数据文件一旦加入到表空间中，就不能从这个表空间中移走，也不能与其他表空间发生关系。

如果数据库对象存储在多个表空间中，那么就可以通过把它们各自的数据文件存放在不同磁盘上来对其进行物理分割。这样不仅可以缓解每个磁盘的存储压力，也有利于分散数据库的 I/O请求，提高数据库读写速度。

§1.3.2 实例

数据库实例（instance）也称作服务器（server），是用来访问数据库文件的存储结构及后台进程的集合。在 ORACLE 系统中，首先是实例启动，然后由实例装配（MOUNT）一数据库。在松耦合系统中，在具有 ORACLE PARALLEL SERVER 选项时，单个数据库可被多个实例装配，即多个实例共享同一物理数据库。

实例使用一组所有用户共享的后台进程和一些存储结构来访问数据库中的数据。

§1.3.2.1 进程结构

这里的进程指系统为了使性能最好和协调多个用户，在多进程系统中使用一些的附加进程。在许多操作系统中，后台进程是在实例启动时自动地建立。一个 ORACLE 实例可以有许多后台进程，但它们不是一直存在。这些进程包括：

- 数据库书写器（Database Writer，缩写 DRWn）

数据库书写器将修改过的块从数据库告诉缓冲存储器中写入数据文件。虽然一个数据库书写器进程(DBWO)对于大多数系统已经足够了，但是用户可以配置附加的进程(DBW1 到 DBW9)来提高有大量数据修改的系统写入性能。初始化参数 DB_WRITER_PROCESSES 用来确定 DBWn 进程的个数。

- 日志书写器(Log Writer,缩写 LGWR)

日志书写器进程将重做日志纪录写入磁盘。重做日志纪录是在系统全局区的重做日志缓冲器中产生的。并且 LGWR 将重做日志纪录按顺序写入联机重做日志文件中。如果数据库有多种重做日志，LGWR 将重做日志纪录写入联机重做日志文件组中。

- 检查点(Checkpoint,缩写 CKPT)

在特定的时间，所有修改的系统全局区数据库缓冲器由 DBWn 将它们写入数据文件；这种情况叫做检查点。检查点进程在检查点响应 DBWn 信号，更新所有的数据库的数据文件和控制文件到大多数最近的检查点。

- 系统监控器（System Monitor，缩写 SMON）

系统监控器在一个失败的实例再次启动是执行崩溃恢复。在多实例系统（使用 Oracle 并行服务器的系统）中，实例的 SMON 的进程可以为其他已经失败的实例执行实例恢复。SMON 也清除不再使用的临时段，恢复在崩溃期间忽略的事务和由于文件读取或掉线错误引

起的实例恢复。这些事务最终由 SMON在表空间或文件重新联机是恢复。

SMON也在数据库字典管理表空间中合并自由区域，产生连续的自由空间，使之容易分配。

- 进程监视器(Process Monitor,缩写 PMON)

进程监视器是在用户进程失败是执行进程恢复。PMON 响应清除高速缓冲存储器和释放进程正在使用的资源。PMON 也检查调度器和服务器进程，如果他们已经失败则重新启动。

- 存档器(Archiver， 缩写 ARCn)

一个或多个存档器进程在联机重做日志满了或日志切换发生时，将它们拷贝到归档存储中。

- 恢复器（Recover， 缩写 RECO）

恢复器用来解决在分布数据库中由于网络或系统失败而挂起的分布事务。在一定的时间间隔里，本地的 RECO 视图连接远程得数据库，并自动地完成提交或任何挂起的分布事务的本地端口的回滚。

- 调度器(Dispatcher,缩写 Dnnnn)

调度器是后台进程的选项，只有在多线程服务器(MTS)配置使用时才给出。

- 锁(Lock，缩写 LCK0)

在 Oracle 并行服务器中，一个锁进程提供内部事务的锁定。

- 作业队列(Job Queue，缩写 SNPn)

在分布的数据库配置中，多至 36个作业队列进程可以自动得刷新表的快照。它们周期性地被唤醒，按计划刷新快照。

与大多数后台进程不同，如果 SNP进程失败，并不产生实例失败。

- 队列监视器(Queue Monitor,缩写 QMNn)

队列监视器进程是针对 Oracle高级队列的可选的后台进程。用户可以配置多至 10 个的监视器进程。如果这些进程失败，它们也不产生实例失败。

§1.3.2.2 内存结构

ORACLE 在内存存储下列信息：

- 执行的程序代码。
- 连接的会话信息
- 程序执行期间所需数据和共享的信息
- 存储在外存储上的缓冲信息。

ORACLE 具有下列基本的内存结构：

- 软件代码区
- 系统全局区,包括数据库缓冲存储区、日志缓冲区和共享池.
- 程序全局区,包括栈区和数据区.
- 排序区

软件代码区

用于存储正在执行的或可以执行的程序代码。

软件区是只读，可安装成共享或非共享。ORACLE 系统程序是可共享的，以致多个 ORACLE 用户可存取它，而不需要在内存有多个副本。用户程序可以共享也可以不共享。

系统全局区

为一组由 ORACLE 分配的共享的内存结构，可包含一个数据库实例的数据或控制信息。如果多个用户同时连接到同一实例时，在实例的 SGA 中数据可为多个用户所共享，所以又称为共享全局区。当实例起动时，SGA 的存储自动地被分配；当实例关闭时，该存储被回收。所有连接到多进程数据库实例的全部用户可自动地被分配；当实例关闭时，该存储被回收。所有连接到多进程数据库实例的全部用户可使用其 SGA 中的信息,但仅仅有几个进程可写入信息。在 SGA 中存储信息将内存划分成几个区:数据库缓冲存储区、日志缓冲区、共享池、请求和响应队列、数据字典存储区和其它各种信息。

程序全局区

PGA 是一个内存区，包含单个进程的数据和控制信息，所以又称为进程全局区（PROCESS GLOBAL AREA)。

排序区

排序需要内存空间，ORACLE 利用该内存排序数据，这部分空间称为排序区。排序区存在于请求排序的用户进程的内存中，该空间的大小为适就排序数据量的大小，可增长，但受初始化参数 SORT-AREA-SIZER 所限制。

§1.4 其他产品简介

除了数据库外，Oracle 公司还有很多其他有名的产品，下面主要介绍一些和数据库相关的工具：

§1.4.1 SQL*PLUS

SQL*PLUS 是 Oracle RDBMS 的主要的字符模式专用接口。使用 SQL*PLUS，用户可以定义和操作 Oracle 数据库中的数据。SQL 是所有数据库产品供应商都遵循的工业标准。Oracle 的 SQL*PLUS 是标准 SQL 的一个超集，除了符合 SQL 标准的语句外，他还提供一些 Oracle 特定的外加语句，该产品的名称很形象的表达了这个意思(SQL 和 PLUS)。

使用 SQL*PLUS，可以很容易的产生各种字符模式的报表来。使用 SQL*PLUS，还可以创建动态 SQL*PLUS 脚本，甚至可以创建动态的用于特定操作系统的命令语言程序。SQL*PLUS 可以被用来执行某些 Oracle 的管理动能，可以被编成者在特定的终端会话期间进行交互。SQL*PLUS 还可以处理 ANSI SQL 和 PL/SQL 块。

§1.4.2 Server Manager

为了帮助进行 Oracle 数据库的设置、管理和日常操作，Oracle 提供了一种名为 Server Manager 的工具。尽管这种工具具有许多功能，但是其最主要的功能还是用来启动和关闭本地或远程数据库。这种通用性是数据库管理员能够非常灵活的管理数据库。

Server Manager 的第二个功能包括：改变数据库和进行系统设计，修改数据库的特征，管理用户和安全机制，恢复数据库和操作属于数据库的数据文件。Server Manager 的命令行模式是一个非图形接口，它允许用户通过输入命令进行交互。而输出则显示在用户屏幕上。这种模式对于那些不需要 GUI 接口的数据库管理任务是非常有用的。

Server Manager 的命令模式与命令行模式除了使用的方式不同外基本上是等价的。命令行模式从本质上来说是交互式的，而命令模式却意味着运行成批的命令。一般来说，命令模式被用来运行由用户创建的脚本或命令的集合。

§1.4.3 Oracle Enterprise Manager(OEM)

Oracle Enterprise Manager(OEM)是 Oracle 公司所提供的工具中使用的最多的工具之一。Enterprise Manager 使数据库管理员和用户可以很容易的察看和操作 Oracle 数据库的各方面特征。以前，Oracle 提供两种基本工具：SQL*DBA 和 Server Manager。这两种

1

工具主要用于服务器段的管理，所以在功能和可用性上有所限制。OEM 实在这些工具指上的一个重大突破。它允许用户管理、监控和调整不同机器、平台和版本上的数据库。Enterprise Manager 被设计为与几乎所有的 Oracle 活动相联系的唯一的接触点。OEM 的许多功能都是可以在命令行实用程序上运行的简单命令。这意味着尽管 OEM 更易于应用，但是当不能使用 OEM 时，也可以使用其它工具。

Enterprise Manager 实际上是一组应用程序，如下所示：

应用程序	描述
Enterprise Manager	Enterprise Manager 的主要控制台
Backup Manager	备份与恢复
Data Manager	导入、导出和载入数据
Instance Manager	启动 / 关闭、初始化参数
Schema Manager	管理所有的用户对象
Security Manager	管理用户、角色和配置文件
Software Manager	在客户机 / 服务器环境中分配软件
SQL Worksheet	SQL 行命令，与 SQL*PLUS 相似
Storage Manager	管理数据文件和表空间

§1.4.3 SQL*Loader

目前，DBA 所面临的最具挑战性的问题就是将数据从外部数据源移植到 Oracle 数据库中。而数据仓库的引入增加了这项任务的复杂性；要进行移植的数据从 Mb 增加到了 GB，在某些情况下，甚至到了 TB。Oracle 使用 SQL*Loader 实用程序来满足这方面的需要，这是一种可以将外部数据载入到 Oracle 数据库的表中去的灵活工具。SQL*Loader 不仅灵活，而且还是可配置的，他常常可以使用户避免去开发嵌入式 SQL 的 3GL 程序。当面临将外部数据转换为 Oracle 格式的任务时，用户在求助于其他方法之前都应该首先考虑使用 SQL*Loader。

§1.4.4 导入与导出

导入与导出是 Oracle 支持的两个补充实用程序。这两个工具主要用来实现备份和恢复数据，把数据移至其他 Oracle 数据库，以及将数据从 Oracle 的老版本中迁移到更新的版本中。导入与导出的其他的一些应用如下所示：

- 在操作系统文件中存储数据以用于存档。
- 存储数据库对象的定义
- 有选择的备份数据库中的一部分。
- 将数据从一个 Oracle 用户模式移至另一个用户模式。
- 将数据从一个硬件平台或操作系统移至另一个硬件平台或操作系统中。
- 通过减少存储碎片来节省空间并提高性能
- 增量备份并恢复数据

导入和导出使用程序的操作时非常简单易学的。导出实用程序先写出有关表或数据库对象的信息以及来源于 Oracle 表本身的数据信息，如表的创建、索引的创建、表的授权以及表的大小信息等。导出实用程序随后将这些信息保存到已命名的操作系统文件中。这些由导出实用程序创建的操作系统文件被称之为转储文件。通常，以 Oracle 二进制格式存储的转储文件进能被导入实用程序使用。无论操作系统是否允许，都可以给转储文件命名。除非指定了另一个名称，否则导出的信息在缺省的情况下将被存入 ERPDAT.DMP 文件中。

然后，可以把导出实用程序产生的输出文件存入磁盘或脱机存储，以后导入实用程序可以利用这些文件来重建导出的数据，这样就能达到恢复或者维护系统的目的。

导出格式如下：

exp userid=[user]/[password] OWNER=[owner].. [other options]

导入格式如下：

imp userid=[user]/[password] file=[Import file].. [other options]

Chapter 2　创建数据库

单元培训目标

- ◆创建数据库之前应考虑的因素
- ◆使用 Oracle 数据库配置代理创建数据库
- ◆手工创建数据库
- ◆安装参数
- ◆创建数据库之后应考虑的因素

§2.1　创建数据库的准备

在创建数据库之前，对需要建立的数据库必须有一个整体的规划。同时，用户必须保证创建数据库的一些先决条件已经得到满足。

§2.1.1 规划

创建数据库前的规划主要包括：

- 首先是要创建的数据库的安装参数，比较重要的有：
  - DB_NAME/DB_DOMAIN
  - CONTROL_FILES
  - DB_BLOCK_BUFFERS
  - PROCESSES
  - ROLLBACK_SEGMENTS
  - 许可参数

关于这些参数的意义及如何配置，请参看 2.3 节“安装参数”，以及其他一些相关的章节。或者可以参见“Oracle8i Reference”。

- 规划数据库的表及索引，估计它们大概要占用多大的空间。请参看相关章节，估算这些数据库对象的占用空间。或参考“Oracle8i Concepts”。

- 规划怎样防止用户的新数据库创建失败，以及日志模式（是否归档，联机重做日志如何配置，需要多大空间等等）和备份策略。请参看 4.4 节“日志文件”，了解日志文件及如何配置。备份策略请参见“Oracle8i Backup And Recovery Guide”。

- 决定数据库的字符集。在创建数据库时，用户必须确定数据库的字符集。所有的字符数据，包括数据字典中的数据，都存储在数

据字符集中。如果用户用不同的字符集来访问数据库，数据库字符集应该与所使用的字符集相同或是它的超集。

- 选择数据库块的大小。这确定为初始化参数，在不进行数据库的重新创建时，它是不可更改的。

- 选择用户的全局数据库名，这个名字在 Create Database 中指定，而且它也可以由安装参数指定。

另外，要熟悉一个实例的启动和关闭，数据库的安装和打开的原则和选项。

§2.1.2 前提条件

在创建一个新的数据库之前，用户必须保证一下的前提条件得到满足：

- 安装期望的 Oracle 软件。包括设置各种环境变量，这些变量对于用户的操作系统是唯一的；建立软件和数据库文件的结构目录。

- 用户拥有操作系统与整个数据库管理操作相关的权限。用户必须由操作系统或通过一个密码文件来进行特别授权，使得用户可以在数据库创建或打开之前能够启动或关闭某个实例。

- 充足的内存用于启动 Oracle 实例。

- 在运行 Oracle 数据库的计算机上又刮化数据库所需的充足的磁盘存储空间。

§2.2 Oracle 数据库配置代理(DBCA)

Oracle数据库配置代理是用来创建数据库的一个图形用户界面(GUI)的工具，下面介绍使用这种工具创建数据库的优点及其内部机制。

§2.2.1 Oracle 数据库配置代理的特点

作为一种图形几面的工具，Oracle数据库配置代理具有以下优点：

？为用户设置了最佳缺省参数，不需用户干预；它使用最佳柔性结构(OFA, Optimal Flexible Architecture)，而数据库文件和管理文件，包括安装文件，按照标准的命名和布局实践。

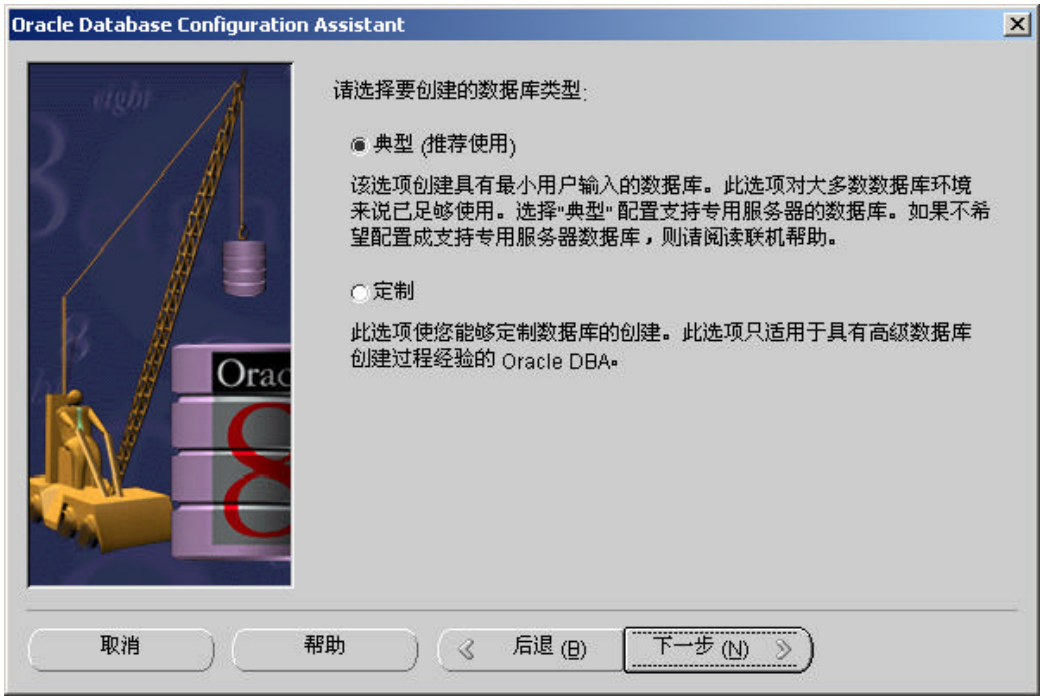
? 速度快；用户可以选择直接从安装光盘上拷贝一个已创建的数据库，而不是进行一个冗长的创建过程。这也是创建数据库的最快方法。

? 还可以用来修改、删除现有数据库。在创建数据库之后，根据用户的需求，还可以使用 Oracle 配置代理来更改或者删除现有的数据库。

? 根据用户不同的选择，Oracle 配置代理自动帮你设置所需的数据库类型。用户只需回答几个问题，Oracle 配置代理自动生成创建用户数据库的 OLTP 的脚本、数据仓库、多目标环境等。虽然使用 Oracle 配置代理来创建数据库的确比较方便，但是由于所有的过程都是在后台进行，用户无法了解创建数据库的内部机制。

§2.2.2 使用 DBCA 创建数据库的选项及步骤

在进入 Oracle 数据库配置代理后，选择“创建数据库”后，系统提示所需的数据库的类型，包括“典型”和“定制”（图 2.1）。



以上版本为 Oracle8.1.7，在 Oracle8.1.6 中还有一选项为“最小”。

§2.2.2.1 典型

选择“典型安装”后，Oracle 数据库配置代理将在安装的最后阶段自动启动，并创建一个经过配置的，备用的多目标启动器，它具有如下特点：

- 缺省安装参数

- Oracle选项和 interMedia元件的自动安装和配置
- 先进的复制能力
- 多线程服务器模式的数据库配置
- NOARCHIVELOG 的归档模式
- 用户只需输入全局数据库名，选择一下所需数据库中使用的应用程序的主要类型及所需的安装组件，DBCA 自动进行安装，无需用户干预。

使用典型安装的步骤如下：

1. 进入 Oracle数据库配置代理，选择“ 创建数据库 ”、“ 典型安装 ”；

2. 选择从ＣＤ拷贝现有数据库文件或是创建新的数据库文件。

拷贝现有数据库文件需要 Oracle8i 的安装光盘。这是创建数据库的最快办法。使用该选项用户只需输入新数据库的全局数据库名和 Oracle系统标识符，即可开始进行正式安装（缺省的 Oracle 系统标识符与全局数据名一致）。

创建新的数据库文件速度较慢。它根据在客户计算机上检测到的内存可用数量，和用户提供的有关数据库将在其中操作的环境信息，创建一个由缺省和定制参数设置联合组成的新数据库。

3. 如果选择了“ 创建新的数据库文件 ”，下一步用户必须选择在该数据库中使用的应用程序的主要类型。

＊ 联机事务处理（OLTP）

选择此选项可创建一种适合 OLTP 环境的数据库环境。OLTP 环境中的数据库每天必须处理来自许多并发 （同时连接的）用户的成千乃至上百万的事务。用户必须能够快速访问最新数据。因此，数据库的性能取决于数据吞吐量 （速度）和数据可用性。

＊ 数据仓库

选择此选项可创建一种适合数据仓库环境的数据库环境。 数据仓库环境中的数据库必须处理各种各样的查询（通常是只读查

询，如 SELECT 语句），包括从几条纪录的简单读取到许多不同的表中查询数千条纪录的大量复杂查询。因此，数据库性能取决于响应时间。

数据仓库数据库使用经过优化的各种快速、高效的数据检索方案，如星型方案。一个星型方案由超大事实表（包含基本信息）和大量小得多的维表（或查找表，包含事实表中特定的属性条目信息）组成。

＊ 多用途

选择侧选项可以创建适用于混合工作负荷环境（OLTP 和数据仓库）的数据库。这是要安装的缺省数据库。

4. 输入任意时刻并发连接到该数据库的用户数目。如果在上一页选择了 OLTP 并在下面输入了２０或更多的并发用户，将启用 Oracle 多线程服务器支持。

选择所需数据库的插件和选件。

- **Oracle Time Series**

能够通过对象数据类型 (ODT) 存储和检索时间戳数据。例如，应用程序可以使用此插件处理金融市场交易 (如股票、证券和互助基金股份交易) 的历史数据。

- **Oracle Spatial**

使用户能够更加便捷和自然地存储、检索和处理空间数据。

- **Oracle JServer**

提供符合 Java 虚拟机(JVM) 规范和 Java 语言规范的完整的 Java 执行环境。JVM JServer (使用 JServer) 将标准的 Java 名称空间嵌入 RDBMS 方案。这使 Java 程序能够在整个企业内访问存放在 Oracle 数据库和应用程序服务器中的 Java 对象。

- **Oracle interMedia**

Oracle interMedia 添加的支持使 Oracle8i 能够提供更高的可靠性、可用性和查询能力，以及 Internet 和其它应用程序的多媒体内容所需的数据管理服务。该插件包括本地数据类型服务、元数据管理工具、以及支持音频、视频、图象和文本的运算符。

- **Oracle Visual Information Retrieval**

能够对 Oracle8i 数据库管理的图象数据进行存储、检索和处理。支持使用“二进制大对象”(BLOBS)的图象存储和对驻留在外部文件 (BFILE) 中的图象数据的引用。此插件是用于各种成像应用程序的一个构件，而不是一个最终用户应用程序。此插件的一些示 例是数字美术陈列室、房地产交易、文档成像和备用照片集 (如时装设计师和建筑师使用的)。

· **Advanced Replication**

使您能够对分布式环境中不同站点上的多个数据副本进行维护和操作。此插件支持一种同步的、随处更新的复制模式；即数据的所有副本都可被潜在更新，使所有站点最终都使用相同的数据。

5. 输入新数据库的全局数据库名和 SID。SID 标识正在运行的 Oracle8i 软件的特定例程。

6. 立即创建数据库或保留脚本以后运行。

§2.2.2.2 定制

选择此选项可以定制方式创建数据库。选择“定制”还可以自动为数据库配置共享服务器支持 (又称为多线程支持)。

对于熟悉高级数据库创建过程的 Oracle DBA 可以使用此选项进行定制：

＊ 数据文件、控制文件和重做日志文件的设置

＊ 表空间大小

＊ 区大小

＊ 数据库存储容量参数

＊ 归档格式和目标

＊ 跟踪文件目标

＊ 字符集值

＊ 选择要创建的数据库版本

步骤如下：

1. 进入 Oracle 数据库配置代理，选择“创建数据库”、“定制安装”；

2. 选择在该数据库中使用的应用程序的主要类型。

3. 选择并发连接的用户数目。

4. 选择数据库缺省操作的模式。

在下列情况下应选择专用服务器模式：

* 在“数据仓库”环境中使用数据库

* 只有少数客户机连接您的数据库

* 客户机将对数据库发出持久的、长时间运行的请求。

在下列情况下应选择共享服务器模式 (共享服务器模式也称为多线程服务器模式)：

* 在联机事务处理 (OLTP) 环境中使用数据库。通过使用 MTS 可使联机事务处理进程应用程序大大受益。

* 有大量用户需要连接数据库并有效地使用可用的系统资源。

* 有内存限制。与专用服务器相比当用户数量增加时 MTS 减少了大量内存的使用。在后一模式中，内存的使用几乎与用户的数量成比例增加。MTS 可以使您调整和优化全部系统性能，因此如果对优化数据库进行高度控制成为一个重要准则，您就需要 MTS。

* 希望使用连接共享、连接集中和负载均衡等 Net8 功能。

* 最大限度地管理和使用系统资源成为一个重要准则。

* 可预测的和快速的连接次数十分重要，例如对 Web 应用程序。

5. 选择所需数据库的插件和选件。

6. 输入新数据库的全局数据库名和 SID。SID 标识正在运行的 Oracle8i 软件的特定例程。

这里可以设置初始化文件名，更改字符集。

7. 输入控制文件的目录路径，或采用缺省路径。在装载、打开和访问数据库时都要求有一个控制文件。控制文件存储所有数据库、日志文件、恢复所需的同步信息以及数据库名。只要数据库发生结构变化，就必须备份控制文件。

推荐的配置是至少在不同的磁盘上保存三个控制文件。初

始化文件的 CONTROL_FILES 参数标识控制文件的目录路径。Oracle Database Configuration Assistant 不会自动将两个控制文件放置在不同的磁盘上，因此必须手动完成这一任务。

8. 输入几个基本的表空间的信息。您可以在此屏上定制区大小。Oracle8i 以数据块、区和段控制磁盘空间的使用。Oracle8i 数据库中的数据是以数据块方式进行物理存储的。数据文件中的一组连续数据块构成区。

表空间中一组包含特定结构数据的区组成段。

9. 输入重做日志信息。

* 重做日志文件

输入两个重做日志文件的目录路径，或采用缺省路径。重做日志文件用于记录所有数据更改。如果在此数据写入磁盘之前数据库发生故障，所做的更改可以在重做日志文件中找到。这样可以防止数据丢失。

* 重做日志 1 文件大小 (KB)

输入重做日志文件 1 的大小，或者采用缺省输入项。

* 重做日志 2 文件大小 (KB)

输入重做日志文件 2 的大小，或者采用缺省输入项。

10. 输入检查点信息。

* 检查点间隔

输入新填写的触发检查点所需的重做日志文件块数，或采用缺省输入项。检查点是数据库写入器进程 (DBWR) 将 SGA 中所有修改过的数据库缓冲区写入相应的数据文件的时间点。检查点进程负责在检查点通知 DBWR 并更新数据库的所有数据文件和控制文件，以便指明最近的检查点。不管此值为何，当从一个联机重做日志文件切换到另一个重做日志文件时，总会出现检查点。如果此值超过实际的重做日志文件大小，则仅在切换日志文件时才出现检查点。此值将赋给初始化文件的 LOG_CHECKPOINT_INTERVAL 参数。其范围从 2 到无穷大。

* 检查点超时 (秒)

输入出现下一个检查点之前所经历的时间 (秒)，或采用缺省值。输入零值将禁用基于时间的检查点。此值将赋给初始化

文件的 LOG_CHECKPOINT_TIMEOUT 参数。

＊ 启用归档日志

若选择此选项，则将数据库置为 ARCHIVELOG 模式，并使填满的重做日志文件在再次使用之前归档。这样可以使数据库从例程和磁盘故障中完全恢复。如果选定此项，则必须启用自动归档并提供下列四个域的信息。

? 归档格式

输入归档日志文件的格式，或采用缺省输入项。此值将赋给初始化文件的 LOG_ARCHIVE_FORMAT 参数。

? 归档目标

输入存放脱机重做日志文件的目录，或采用缺省输入项。此值将赋给初始化文件的 LOG_ARCHIVE_DEST 参数。

１１．输入 MTS 调度程序参数和一些高级信息。关于这些参数的定义，请参看 2.4 节“安装参数”。

１２．配置 SGA 信息。

每次启动 Oracle8i 数据库例程时，内存中都会分配 SGA。SGA 是一组共享内存结构，包含一个 Oracle8i 数据库系统的数据和控制信息。如果多个用户并行连接到同一个数据库例程，则 SGA 中的数据将由这些用户共享。

在此处显示的 SHARED_POOL_SIZE、DB_BLOCK_BUFFERS、DB_BLOCK_SIZE 和 PROCESSES 等参数的缺省值是由您在前几屏上选定的并行用户数量和数据库环境类型 (OLTP、数据仓库或多用途) 确定的。

１３．输入跟踪目录信息。

＊ 用于用户进程

输入要在其中写入用户进程调试跟踪文件的目录路径，或采用缺省路径。此值由初始化文件参数 USER_DUMP_DEST 标识。用户进程将 SQL 语句传送给服务器进程，然后再接收结果并运行应用程序 (如 Server Manager 或 Pro*C)。

＊ 用于后台进程

输入要在其中写入 Oracle8i 后台进程调试跟踪文件的目

2

录路径，或采用缺省路径。此值由初始化文件参数 USER_DUMP_DEST 标识。

＊ 用于核心转储 (UNIX)

输入要在其中写入核心转储文件的目录，或采用缺省路径。此值由初始化文件参数 CORE_DUMP_DEST 标识。

１４. 立即创建数据库或保留脚本以后运行。

§2.3 手工创建数据库

手工创建数据库可以让用户对创建数据库的内部机制有所了解。通过在创建数据库时解决一些问题，对用户掌握数据库的结构也大有裨益。

§2.3.1 创建数据库的步骤

１. 确定用户的实例标志（DB_NAME 和 SID）。

Oracle 实例标志隐该与数据库名字（DB_NAME）一致。该实例标志用来避免与其他 Oracle 实例混淆，而这里所指的其他 Oracle 实例你可在系统上创建和运行。

２. 创建安装参数文件

对于人已一个 Oracle 数据库的实例（系统全局区域和后台进程）使用安装参数文件来启动。要为数据库创建参数文件，用户将要使用用户的操作系统来制作一个 Oracle 在其发行介质上的安装参数文件的一份拷贝。给这个新的拷贝起一个新的文件名。然后用户可以编辑并定制数据库的这个新文件。请参阅２.4 节“安装参数”获得各种安装参数的信息。

用户系统上的每一个数据库应该至少有一个安装参数文件与该数据库对应，不要几个数据库使用相同的参数文件。

３. 启动 Server Manager，以 SYSDBA 连接到数据库实例。

```
$ svrmgrl

svrmgr> connect username/password AS sysdba
```

4．启动数据库的一个实例

用户不安装数据库就可以启动一个实例。一般来说，用户只在数据库创建期间这样做。使用代NOMOUNT选项的STARTUP语句，如果不指定安装参数，从操作系统的缺省安装中读取。

SVRMGR> startup nomount pfile=

5．创建数据库

使用语句CREATE DATABASE来创建所需数据库，同时根据需要，可以使用另外的一些设置参数来命名数据库、建立文件的最多数值、命名文件并设置大小等等。

6．创建附加的数据库对象

创建了数据库后，对于不同的用途，还需要创建不同的数据库对象。两个必须运行的脚本有：

CATELOG.SQL-创建数据字典表的视图和动态性能视图

CATPROC.SQL-建立PL/SQL功能的应用，并创建Oracle提供的PL/SQL包。

7．备份数据库

§2.3.2 创建数据库的脚本简介

创建数据库用到的主要的SQL语句如下：

1． 创建数据库 –CREATE DATABASE

例：

```
CREATE DATABASE test
 CONTROLFILE REUSE
 LOGFILE
  '/u01/oracle/test/redo01.log 'SIZE 1M REUSE,
  '/u01/oracle/test/redo02.log 'SIZE 1M REUSE,
 '/u01/oracl e/test/redo03.log 'SIZE 1M REUSE,
 '/u01/oracle/test/redo04.log 'SIZE 1M REUSE,
DATAFILE  '/u01/oracle/test/system01.dbf 'SIZE 10M
REUSE
AUTOEXTEND ON
NEXT 10M MAXSIZE 200M
CHARACTER SET WE8ISO8859P1;
```

上面这段话的作用如下：

① **创建数据库的数据文件；**
② **创建数据库的控制文件；**
③ **创建数据库的重做日志文件；**
④ **创建** SYSTEM **表空间及回滚段；**
⑤ **创建数据目录；**
⑥ **创建用户** SYS**和** SYSTEM**；**
⑦ **指定数据库中存储数据的字符集；**
⑧ **装入、打开数据库。**

2. 创建系统回滚段

CREATE ROLLBACK SEGMENT rb_temp
STORAGE(INITIAL 100k NEXT 250k);

ALTER ROLLBACK SEGMENT rb1 ONLINE;

创建一个临时的回滚段来支持数据库的创建。

3. 为回滚段创建表空间

CREATE TABLESPACE rbs DATAFILE
'/u01/oracle/test/rbs01.dbf'SIZE 5M REUSE
AUTOEXTEND ON NEXT 5M MAXSIZE 150M;

4. 创建用户表空间

CREATE TABLESPACE users DATAFILE
'/u01/oracle/test/user01.dbf'SIZE 3M REUSE
AUTOEXTEND ON NEXT 5M MAXSIZE 150M;

5. 创建临时表空间

CREATE TABLESPACE temp DATAFILE
'/u01/oracle/test/temp01.dbf'SIZE 2M REU SE
AUTOEXTEND ON NEXT 5M MAXSIZE 150M;

6. 创建回滚段

CREATE ROLLBACK SEGMENT rb1 STORAGE (INITIAL
50k NEXT 250k) TABLESPACE rbs;

CREATE ROLLBACK SEGMENT rb2 STORAGE (INITIAL
50k NEXT 250k) TABLESPACE rbs;

CREATE ROLLBACK SEGMENT rb3 STORAGE (INITIAL

50k NEXT 250k) TABLESPACE rbs;

CREATE ROLLBACK SEGMENT rb4 STORAGE (INITIAL 50k NEXT 250k) TABLESPACE rbs;

ALTER ROLLBACK SEGMENT rb1 ONLINE;
ALTER ROLLBACK SEGMENT rb2 ONLINE;
ALTER ROLLBACK SEGMENT rb3 ONLINE;
ALTER ROLLBACK SEGMENT rb4 ONLINE;
ALTER ROLLBACK SEGMENT rb_temp OFFLINE;
DROP ROLLBACK SEGMENT rb_temp;

创建用户事务所需的回滚段，然后脱机临时回滚段，并删除。

§2.4 安装参数

在创建或启动数据库时，系统会从安装参数文件中读一些缺省的参数。用户可以更改安装参数文件，修改其中的参数来调整数据库创建或启动时的缺省值。或者当用户对数据库比较熟悉时，可以通过 ALTER SYSTEM **来直接动态的修改这些参数。**

§2.4.1 安装参数简介

在安装参数文件中，比较重要的参数有：

1. DB_NAME **和** DB_DOMAIN

DB_NAME **指的是数据库的本地名称，**DM_DOMAIN **指的是数据库位于网络结构中的主域名称。两者结合起来就确定了在网络中的唯一的数据库名称。**

2. CONTROL_FILES

指定创建或启动数据库时的控制文件位置及名称。

3. DB_BLOCK_SIZE

每个数据库块的大小，一般为 2k **或** 4k**。一般来说，提高数据库块的大小能提高磁盘和内存**IO **的效率。**

4. DB_BLOCK_BUFFERS

这个参数指在系统全局区域中数据库缓冲器的大小。

5. PROCESSES

连到 Oracle 数据库上的操作系统进程的最大数量。

6. ROLLBACK_SEGMENTS

Oracle 实例回滚段及用户回滚段的列表。

Chapter 3　Oracle 实例

单元培训目标

- **了解什么是实例**
- **了解 Oracle 实例的进程结构**
- **了解 Oracle 实例的内存结构**
- **了解 Oracle 实例的各个阶段**
- **掌握数据库的启动与关闭**

§3.1　简介

每一个运行的 ORACLE 数据库与一个 ORACLE 实例（INSTANCE）相联系。一个 ORACLE 实例为存取和控制一数据库的软件机制。每一次在数据库服务器上启动一数据库时，称为系统全局区（SYSTEM GLOBAL AREA）的一内存区（简称 SGA）被分配，有一个或多个 ORACLE 进程被启动。该 SGA 和 ORACLE 进程的结合称为一个 ORACLE 数据库实例。一个实例的 SGA 和进程为管理数据库数据、为该数据库一个或多个用户服务而工作。

在 ORACLE 系统中，首先是实例启动，然后由实例装配（MOUNT）一数据库。在松耦合系统中，在具有 ORACLE PARALLEL SERVER 选项时，单个数据库可被多个实例装配，即多个实例共享同一物理数据库。

§3.2　进程结构

进程是操作系统中的一种机制，它可执行一系列的操作步。在有些操作系统中使用作业(JOB)或任务(TASK)的术语。一个进程通常有它自己的专用存储区。ORACLE 进程的体系结构设计使性能最大。

ORACLE 实例有两种类型：单进程实例和多进程实例。

单进程 ORACLE（又称单用户 ORACLE）是一种数据库系统，一个进程执行全部 ORACLE 代码。由于 ORACLE 部分和客户应用程序不能分别以进程执行，所以 ORACLE 的代码和用户的数据库

应用是单个进程执行。

在单进程环境下的 ORACLE 实例，仅允许一个用户可存取。例如在 MS-DOS 上运行 ORACLE 。

多进程 ORACLE 实例（又称多用户 ORACLE）使用多个进程来执行 ORACLE 的不同部分，对于每一个连接的用户都有一个进程。

在多进程系统中，进程分为两类：用户进程和 ORACLE 进程。当一用户运行一应用程序,如 PRO*C 程序或一个 ORACLE 工具（如 SQL*PLUS），为用户运行的应用建立一个用户进程。ORACLE 进程又分为两类：服务器进程和后台进程。服务器进程用于处理连接到该实例的用户进程的请求。当应用和 ORACELE 是在同一台机器上运行，而不再通过网络，一般将用户进程和它相应的服务器进程组合成单个的进程，可降低系统开销。然而，当应用和 ORACLE 运行在不同的机器上时，用户进程经过一个分离服务器进程与 ORACLE 通信。它可执行下列任务：

对应用所发出的 SQL 语句进行语法分析和执行。

从磁盘（数据文件）中读入必要的数据块到 SGA 的共享数据库缓冲区（该块不在缓冲区时）。

将结果返回给应用程序处理。

系统为了使性能最好和协调多个用户，在多进程系统中使用一些附加进程，称为后台进程。在许多操作系统中，后台进程是在实例启动时自动地建立。一个 ORACLE 实例可以有许多后台进程，但它们不是一直存在。后台进程的名字为：

DBWR 数据库写入程序

LGWR 日志写入程序

CKPT 检查点

SMON 系统监控

PMON 进程监控

ARCH 归档

RECO 恢复

LCKn 封锁

Dnnn 调度进程

Snnn 服务器

每个后台进程与ORACLE数据库的不同部分交互。

下面对后台进程的功能作简单介绍：

DBWR 进程：该进程执行将缓冲区写入数据文件，是负责缓冲存储区管理的一个 ORACLE 后台进程。当缓冲区中的一缓冲区被修改，它被标志为“弄脏”，DBWR 的主要任务是将“弄脏”的缓冲区写入磁盘，使缓冲区保持“干净”。由于缓冲存储区的缓冲区填入数据库或被用户进程弄脏，未用的缓冲区的数目减少。当未用的缓冲区下降到很少，以致用户进程要从磁盘读入块到内存存储区时无法找到未用的缓冲区时，DBWR 将管理缓冲存储区，使用户进程总可得到未用的缓冲区。

ORACLE 采用 LRU（LEAST RECENTLY USED）算法（最近最少使用算法）保持内存中的数据块是最近使用的，使 I/O 最小。在下列情况预示 DBWR 要将弄脏的缓冲区写入磁盘：

当一个服务器进程将一缓冲区移入“弄脏”表，该弄脏表达到临界长度时，该服务进程将通知 DBWR 进行写。该临界长度是为参数 DB-BLOCK-WRITE-BATCH 的值的一半。

当一个服务器进程在 LRU 表中查找 DB-BLOCK-MAX-SCAN-CNT 缓冲区时，没有查到未用的缓冲区，它停止查找并通知 DBWR 进行写。

出现超时（每次 3 秒），DBWR 将通知本身。

当出现检查点时，LGWR 将通知 DBWR

在前两种情况下，DBWR 将弄脏表中的块写入磁盘，每次可写的块数由初始化参数 DB-BLOCK-WRITE-BATCH 所指定。如果弄脏表中没有该参数指定块数的缓冲区，DBWR 从 LUR 表中查找另外一个弄脏缓冲区。

如果 DBWR 在三秒内未活动，则出现超时。在这种情况下 DBWR 对 LRU 表查找指定数目的缓冲区，将所找到任何弄脏缓冲区写入磁盘。每当出现超时，DBWR 查找一个新的缓冲区组。每次由 DBWR 查找的缓冲区的数目是为寝化参数 DB-BLOCK-WRITE-BATCH 的值的二倍。如果数据库空运转，DBWR 最终将全部缓冲区存储区写入磁盘。

在出现检查点时，LGWR 指定一修改缓冲区表必须写入到磁

盘。DBWR 将指定的缓冲区写入磁盘。

在有些平台上，一个实例可有多个 DBWR。在这样的实例中，一些块可写入一磁盘，另一些块可写入其它磁盘。参数 DB-WRITERS 控制 DBWR 进程个数。

LGWR 进程：该进程将日志缓冲区写入磁盘上的一个日志文件，它是负责管理日志缓冲区的一个 ORACLE 后台进程。LGWR 进程将自上次写入磁盘以来的全部日志项输出，LGWR 输出：

当用户进程提交一事务时写入一个提交记录。

每三秒将日志缓冲区输出。

当日志缓冲区的 1 / 3 已满时将日志缓冲区输出。

当 DBWR 将修改缓冲区写入磁盘时则将日志缓冲区输出。

LGWR 进程同步地写入到活动的镜象在线日志文件组。如果组中一个文件被删除或不可用，LGWR 可继续地写入该组的其它文件。

日志缓冲区是一个循环缓冲区。当 LGWR 将日志缓冲区的日志项写入日志文件后，服务器进程可将新的日志项写入到该日志缓冲区。LGWR 通常写得很快，可确保日志缓冲区总有空间可写入新的日志项。

注意：有时候当需要更多的日志缓冲区时，LWGR 在一个事务提交前就将日志项写出，而这些日志项仅当在以后事务提交后才永久化。

ORACLE 使用快速提交机制，当用户发出 COMMIT 语句时，一个 COMMIT 记录立即放入日志缓冲区，但相应的数据缓冲区改变是被延迟，直到在更有效时才将它们写入数据文件。当一事务提交时，被赋给一个系统修改号（SCN），它同事务日志项一起记录在日志中。由于 SCN 记录在日志中，以致在并行服务器选项配置情况下，恢复操作可以同步。

CKPT 进程：该进程在检查点出现时，对全部数据文件的标题进行修改，指示该检查点。在通常的情况下，该任务由 LGWR 执行。然而，如果检查点明显地降低系统性能时，可使 CKPT 进程运行，将原来由 LGWR 进程执行的检查点的工作分离出来，由 CKPT 进程实现。对于许多应用情况，CKPT 进程是不必要的。只有当数据库有许多数据文件，LGWR 在检查点时明显地降低性能才使

CKPT 运行。CKPT 进程不将块写入磁盘，该工作是由 DBWR 完成的。

初始化参数 CHECKPOINT-PROCESS控制 CKPT 进程的使能或使不能。缺省时为 FALSE，即为使不能。

SMON 进程：该进程实例启动时执行实例恢复，还负责清理不再使用的临时段。在具有并行服务器选项的环境下，SMON 对有故障 CPU 或实例进行实例恢复。SMON 进程有规律地被呼醒，检查是否需要，或者其它进程发现需要时可以被调用。

PMON 进程：该进程在用户进程出现故障时执行进程恢复，负责清理内存储区和释放该进程所使用的资源。例：它要重置活动事务表的状态，释放封锁，将该故障的进程的 ID 从活动进程表中移去。PMON 还周期地检查调度进程（DISPATCHER）和服务器进程的状态，如果已死，则重新启动（不包括有意删除的进程）。

PMON 有规律地被呼醒，检查是否需要，或者其它进程发现需要时可以被调用。

RECO 进程：该进程是在具有分布式选项时所使用的一个进程，自动地解决在分布式事务中的故障。一个结点 RECO 后台进程自动地连接到包含有悬而未决的分布式事务的其它数据库中，RECO 自动地解决所有的悬而不决的事务。任何相应于已处理的悬而不决的事务的行将从每一个数据库的悬挂事务表中删去。

当一数据库服务器的 RECO 后台进程试图建立同一远程服务器的通信，如果远程服务器是不可用或者网络连接不能建立时，RECO 自动地在一个时间间隔之后再次连接。

RECO 后台进程仅当在允许分布式事务的系统中出现，而且 DISTRIBUTED – TRANSACTIONS 参数是大于 0。

ARCH 进程：该进程将已填满的在线日志文件拷贝到指定的存储设备。当日志是为 ARCHIVELOG 使用方式、并可自动地归档时 ARCH 进程才存在。

LCKn进程：是在具有并行服务器选件环境下使用，可多至 10 **个进程（**LCK0**，**LCK1……**，**LCK9**），用于实例间的封锁。**

Dnnn进程（调度进程）：该进程允许用户进程共享有限的服务器进程（SERVER PROCESS**）。没有调度进程时，每个用户进程需要一个专用服务进程（**DEDICATEDSERVER PROCESS**）。对于多线索服务器（**MULTI-THREADED SERVER **）可支持多个用户进程。如果在系统中具有大量用户，多线索服务器可支持大量用户，尤其在客户 _服务器环境中。**

在一个数据库实例中可建立多个调度进程。对每种网络协议至少建立一个调度进程。数据库管理员根据操作系统中每个进程可连接数目的限制决定启动的调度程序的最优数，在实例运行时可增加或删除调度进程。多线索服务器需要 SQL*NET **版本** 2 **或更后的版本。在多线索服务器的配置下，一个网络接收器进程等待客户应用连接请求，并将每一个发送到一个调度进程。如果不能将客户应用连接到一调度进程时，网络接收器进程将启动一个专用服务器进程。该网络接收器进程不是** ORACLE **实例的组成部分，它是处理与** ORACLE **有关的网络进程的组成部分。在实例启动时，该网络接收器被打开，为用户连接到** ORACLE **建立一通信路径，然后每一个调度进程把连接请求的调度进程的地址给予于它的接收器。当一个用户进程作连接请求时，网络接收器进程分析请求并决定该用户是否可使用一调度进程。如果是，该网络接收器进程返回该 调度进程的地址，之后用户进程直接连接到该调度进程。有些用户进程不能调度进程通信（如果使用** SQL*NET **以前的版本的用户），网络接收器进程不能将如此用户连接到一调度进程。在这种情况下，网络接收器建立一个专用服务器进程，建立一种合适的连接。**

§3.3 内存结构

ORACLE **在内存存储下列信息：**

1. **执行的程序代码**;

2. **连接的会话信息**

3. **程序执行期间所需数据和共享的信息**

4. **存储在外存储上的缓冲信息。**

ORACLE **具有下列基本的内存结构：**

1. **软件代码区**

3

2. 系统全局区,包括数据库缓冲存储区、日志缓冲区和共享池.

3. 程序全局区,包括栈区和数据区.

4. 排序区

软件代码区

用于存储正在执行的或可以执行的程序代码。

软件区是只读，可安装成共享或非共享。ORACLE 系统程序是可共享的，以致多个 ORACLE 用户可存取它，而不需要在内存有多个副本。用户程序可以共享也可以不共享。

系统全局区

为一组由 ORACLE 分配的共享的内存结构，可包含一个数据库实例的数据或控制信息。如果多个用户同时连接到同一实例时，在实例的 SGA 中数据可为多个用户所共享，所以又称为共享全局区。当实例起动时，SGA 的存储自动地被分配；当实例关闭时，该存储被回收。所有连接到多进程数据库实例的全部用户可自动地被分配；当实例关闭时，该存储被回收。所有连接到多进程数据库实例的全部用户可使用其 SGA 中的信息，但仅仅有几个进程可写入信息。在 SGA 中存储信息将内存划分成几个区：数据库缓冲存储区、日志缓冲区、共享池、请求和响应队列、数据字典存储区和其它各种信息。

程序全局区

PGA 是一个内存区，包含单个进程的数据和控制信息，所以又称为进程全局区（PROCESS GLOBAL AREA）。

排序区

排序需要内存空间，ORACLE 利用该内存排序数据，这部分空间称为排序区。排序区存在于请求排序的用户进程的内存中，该空间的大小为适就排序数据量的大小，可增长，但受初始化参数 SORT-AREA-SIZER 所限制。

§3.4 启动与关闭

当我们说启动某个数据库时，实际上是先启动改数据库的一个实例，然后再装入和打开该数据库。启动或关闭数据库有三种方法:Server Manager、Revery Manager 或者 Oracle Enterprise Manager。下面我们简单介绍一下用 Server Manager启动数据库的过程。

首先，在操作系统上启动 Server Manager：

1. 设置环境参数：

> set ORACLE_SID=test

2. 后动 Server Manager:

> svrmgrl

显示 SVRMGR> ，表明已进入了 Server Manager。

3. 以 SYSDBA 方式连接 Oracle：

SVRMGR> connect username/password as sysdba

现在用户就连接到数据库，可以进行数据库的启动或关闭操作了。

§3.4.1 启动

§3.4.1.1 启动阶段

启动数据库分为三个阶段：

1. 启动一个实例；

启动一实例的处理包含分配一个 SGA（数据库信息使用的内存共享区）和后台进程的建立。实例起动的执行先于该实例装配一数据库。如果仅启动实例,则没有数据库与内存储结构和进程相联系。命令为：

SVRMGR> startup nomount

2. 装配一数据库

装配数据库是将一数据库与已启动的实例相联。当实例安装一数据库之后，该数据库保持关闭，仅 DBA 可存取。命令为：

SVRMGR> alter database mount;

3. **打开一数据库**

打开一数据库是使数据库可以进行正常数据库操作的处理。当一数据库打开所有用户可连接到该数据库用存取其信息。在数据库打开时，在线数据文件和在线日志文件也被打开。如果一表空间在上一次数据库关闭时为离线，在数据库再次打开时，该表空间与它所相联的数据文件还是离线的。

SVRMGR> alter database open;

在实际启动数据库时，除非需要，否则可以直接从数据库关闭阶段启动到打开阶段，命令为：

SVRMGR> startup

§3.4.1.2 启动选项

在启动数据库时有几个选项：

1. **以** Restrict **方式打开数据库：**

命令：

SVRMGR> startup restrict

以 restrict **方式打开的数据库只允许具有** RESTRICTED SESSION **的数据库管理员访问数据库，而一般的数据库允许具有** CREATE SESSION **的用户访问数据库。**

2. **以强制方式打开数据库**

命令：

SVRMGR> startup force

一般来说，该选项只在用正常方式（SHUTDOWN NORMAL、SHUTDOWN IMMEDIATE、SHUTDOWN TRANSACTIONAL**）无法关闭数据库时，才用来强制启动数据库的另一实例（在启动前以** SHUTDOWN ABORT **关闭数据库），然后再正常关闭数据库。**

3. **启动并恢复**

如果用户知道所要求的介质恢复，可以启动一个数据库实例，装入该数据库，并自动的启动恢复程序。命令：

SVRMGR> startup open recover;

4. 以独占或并行方式启动数据库

如果用户的 Oracle 服务器允许多个实例来并发的访问一个数据库（Oracle 并行服务器选项），选择是独占或是并行的装入数据库。命令为：

并行：

SVRMGR> startup open test pfile=init.ora parallel;

独占：

SVRMGR> startup open test pfile=init.ora exclusive

§3.4.2 关闭

§3.4.2.1 关闭阶段

关闭一实例以及它所连接的数据库也有三步操作：

1. **关闭数据库**

数据库停止的第一步是关闭数据库。当数据库关闭后，所有在 SGA 中的数据库数据和恢复数据相应地写入到数据文件和日志文件。在这操作之后，所有联机数据文件和联机的日志文件也被关闭，任何离线表空间中数据文件夹是已关闭的。在数据库关闭后但还安装时，控制文件仍保持打开。

2. **卸下数据库**

停止数据库的第二步是从实例卸下数据库。在数据库卸下后，在计算机内存中仅保留实例。在数据库卸下后，数据库的控制文件也被关闭。

3. **停止实例**

停止数据库的最后一步是停止实例。当实例停止后，SAG 是从内存中撤消，后台进程被中止。

与启动数据库不同，关闭数据库不能从打开阶段回到装入阶段或者非装入阶段，而只能直接从打开阶段回到关闭阶段。命令为：

SVRMGR> shutdown

§3.4.2.2 关闭选项

关闭数据库主要有四个选项：

1. 一般方式(normal)

在缺省或一般方式下关闭数据库，有以下特点：

- 在语句发出后不允许新的连接；
- 在数据库关闭前，等待所有用户与数据库断开连接；
- 数据库下一次启动不需要任何实例恢复程序。

2. 立即方式(immediate)

仅在下列情况下使用立即方式关闭数据库：

- 很快就要关闭电源；
- 数据库或其他应用程序功能不正常。

立即方式有以下特点：

- 未授权的事务退回（如果长期有未授权的事务存在，可能不会很快）；
- 退回所有激活事务，并断开所有的连接用户；
- 数据库下一次启动不需要任何实例恢复程序。

3. 事务方式(transactional)

事务方式的特点：

- 等待用户提交事务，与立即方式相比，这点可防止客户端丢失工作；
- 数据库下一次启动不需要任何实例恢复程序。

4. Abort 方式

在下列情况下使用 Abort 方式：

- 很快就要关闭电源；
- 数据库或其他应用程序功能不正常；
- 启动数据时遭遇问题时。

特点：

- **立即中断与所有客户端的连接；**
- **不会回退或提交为提交事务；**
- **数据库下一次启动将要求实例恢复程序。**

Chapter4　Oracle 物理结构

单元培训目标

- ◆掌握 Oracle 物理结构的组成
- ◆了解数据文件的创建和管理
- ◆察看数据文件的信息
- ◆了解控制文件的创建和管理
- ◆了解联机日志文件的管理
- ◆了解存档日志文件的管理

§4.1　简介

一个 ORACLE 数据库是数据的集合，被处理成一个单位。一个 ORACLE 数据库有一个物理结构和一个逻辑结构。

物理数据库结构（physical database structure）是由构成数据库的操作系统文件所决定。每一个 ORACLE 数据库是由三种类型的文件组成：数据文件、日志文件和控制文件。数据库的文件为数据库信息提供真正的物理存储。

每一个 ORACLE 数据库有一个或多个物理的数据文件(data file)。一个数据库的数据文件包含全部数据库数据。逻辑数据库结构(如表、索引)的数据物理地存储在数据库的数据文件中。数据文件有下列特征：

- 一个数据文件仅与一个数据库联系。
- 一旦建立，数据文件不能改变大小
- 一个表空间（数据库存储的逻辑单位）由一个或多个数据文件组成。

数据文件中的数据在需要时可以读取并存储在 ORACLE 内存储区中。例如：用户要存取数据库一表的某些数据，如果请求信息不在数据库的内存存储区内，则从相应的数据文件中读取并存储在内存。当修改和插入新数据时，不必立刻写入数据文件。为了减少磁盘输出的总数，提高性能，数据存储在内存，然后由 ORACLE 后台进程 DBWR 决定如何将其写入到相应的数据文件。

每一个数据库有两个或多个日志文件（redo log file）的组，每一个日志文件组用于收集数据库日志。日志的主要功能是记录对数据所作的修改，所以对数据库作的全部修改是记录在日志中。在出现故障时，如果不能将修改数据永久地写入数据文件，则可利用日志得到该修改，所以从不会丢失已有操作成果。

日志文件主要是保护数据库以防止故障。为了防止日志文件本身的故障，ORACLE允许镜象日志(mirrored redo log)，以致可在不同磁盘上维护两个或多个日志副本。

日志文件中的信息仅在系统故障或介质故障恢复数据库时使用，这些故障阻止将数据库数据写入到数据库的数据文件。然而任何丢失的数据在下一次数据库打开时，ORACLE自动地应用日志文件中的信息来恢复数据库数据文件。

每一 ORACLE 数据库有一个控制文件(control file)，它记录数据库的物理结构，包含下列信息类型：

- 数据库名；
- 数据库数据文件和日志文件的名字和位置；
- 数据库建立日期。

为了安全起见，允许控制文件被镜象。

每一次 ORACLE数据库的实例启动时，它的控制文件用于标识数据库和日志文件，当着手数据库操作时它们必须被打开。当数据库的物理组成更改时，ORACLE 自动更改该数据库的控制文件。数据恢复时，也要使用控制文件。

§4.2 数据文件

在 Oracle 中，每个文件都有两个相关的文件编号：一个是绝对文件号，一个是相对文件号。

绝对文件号是在数据库中的数据文件的唯一标志。在早期的 Oracle版本中，绝对文件号被简单的称作“文件号”。

相对文件号是在一个表空间中的数据文件的唯一标志。对于中小型数据库，相对文件与绝对文件号相同。然而，当数据库中的数据文件超过某个极限时（一般为 1023），相对文件号与绝对文件号不同。

§4.2.1 数据文件的创建与更改

§4.2.1.1 创建

我们这里指的创建数据文件指向某个数据库表空间增加数据文件，因为数据文件总是属于某个表空间的。为了达到这个目的，要使用 ALTER TABLESPACE … ADD DATAFILE 语句。因此，创建数据文件前，必须保证你的数据库用户有向一个表空间增加数据文件的 ALTER TABLESPACE的系统许可权。

§4.2.1.2 更改

更改一个数据文件主要包括：

- 允许和禁止数据文件的自动扩展；
- 手动调整数据文件的尺寸；
- 改变数据文件的可用性；
- 重命名和重新部署数据文件。

用户可以指定某个数据文件在数据库需要更多空间时是否自动增加文件尺寸。这些文件以指定的增长量增长到指定的最大值。设置数据文件自动扩展的优点在于：

- 不需要人工干预数据表空间扩展；
- 不会因为分配区域失败而中断应用。

更改数据文件的自动扩展格式：

```
ALTER DATABASE DATAFILE [Datafile] AUTOEXTEND ON/OFF;
```

当一个数据文件快要达到指定的最大值时，用户可以根据需要手工调大数据文件的尺寸；而当最初指定数据文件的尺寸过大时，为了节省空间，可以手动调小数据文件的尺寸。但是请注意：一个数据文件可以调整的最小值是受表空间中存储对象多少确定的，并不是可以任意调整的。

调整数据文件大小的语句：

```
ALTER DATABASE DATAFILE [Datafile] RESIZE [size]M/K;
```

改变数据文件的可用性时,用户必须有 ALTER DATABASE 的系统许可权，只有当数据库在高级模式下打开，才能完成这些操作。只有在极个别情况下，用户需要把数据文件脱机或联机。一般

只在一个数据文件发生问题时，手工使他脱机然后再手工使他联机。在改变数据文件可用性时应考虑不同的数据库模式。

在 ARCHIVELOG模式下,可以手工使一个数据文件脱机或联机。格式如下：

```
ALTER DATABASE DATAFILE [Datafile] ONLINE/OFFLINE;
```

但在 NOARCHIVELOG 模式下,应注意如果此时使一个数据文件脱机,很有可能由于不一致而不能再次联机。因此在 NOARCHIVELOG 模式下脱机一般只用来丢弃某个确定不用并且尚未被备份过的数据文件。格式如下：

```
ALTER DATABASE DATAFILE [Datafile] OFFLINE DROP;
```

重命名或者重新部署数据文件可分为下面两种情况：

１. 重命名或者重新部署一个非 SYSTEM 的表空间中的数据文件

这种情况下可直接针对表空间进行操作，步骤如下

1) 使用 ALTER TABLESPACE [Tablespace] OFFLINE 使该表空间脱机；
2) 在操作系统上重命名或者重新部署该表空间中的数据文件；
3) 使用 ALTER TABLESPACE [Tablespace] RENAME [Old Datafile] TO [New Datafile]在数据库中反映数据文件的改变。

２. 重命名或者重新部署多个表空间或者是 SYSTEM表空间中的数据文件

重命名或者重新部署多个表空间的数据文件也可把他们分为单个的表空间进行重命名或重部署。这里介绍的是一步完成重命名或重部署的方法。

1) 确保数据库被安装但未打开 ；
2) 在操作系统上重命名或者重新部署表空间中的数据文件；
3) 使用 ALTER DATABASE [Tablespace] RENAME [Old Datafile] TO [New Datafile]在数据库中反映数据文件的改变。

§4.2.2 数据文件的信息

在Oracle数据库中有几个数据字典视图可以察看数据文件的有用信息：

视图	描述
DBA_DATA_FILES	数据文件的描述信息，包括属于哪个表空间及文件号
USER_EXTENTS,DBA_EXTENTS	数据文件包含的区域
USER_FREE_SPACE,DBA_FREE_SPACE	数据文件的空闲区域
V$DATAFILE	控制文件的数据文件信息
V$DATAFILE_HEADER	数据文件头信息

§4.3 控制文件

§4.3.1 控制文件简介

在Oracle数据库中，用控制文件来记录数据库名称、相关数据文件及联机重做日志文件的名称及位置、数据库创建的时间戳、日志序列号及检查点信息等物理结构。

为了安全起见，Oracle会创建控制文件的一个或多个拷贝。而且为了防止磁盘文件损坏，应把多个拷贝放置于不同的磁盘驱动器上。这样如果某个磁盘文件损坏后，实例将关闭，此时可以拷贝其他拷贝来重新启动实例，而不需要做介质恢复。另外，如果控制文件意外丢失或者用户希望改变控制文件的设置，可能需要人工重建控制文件。

控制文件必须按照初始化参数CONTROL_FILES的格式来命名。在数据库启动后，实例维护并把相关信息写入所有列出的控制文件。

影响控制文件大小的参数主要有：MAXDATAFILES，MAXLOGFILES，MAXLOGMEMBERS，MAXLOGHISTORY，MAXINSTANCES。

§4.3.2 创建控制文件

在数据库创建前通过指定初始化参数文件中的初始化参数CONTROL_FILES来创建Oracle数据库的初始控制文件。

在数据库创建时，如果在系统中存在指定名称的文件，则必须在CREATE DATABASE中使用CONTROL REUSE选项。如果用户在创建数据库前未指定CONTROL_FILES的值，Oracle会创建一

4

个缺省的控制文件。这个缺省控制文件的名称和位置根据不同的系统而有所不同。

§4.3.3 更改控制文件

更改控制文件包括：

- 创建附加的控制文件拷贝
- 重命名现有的控制文件
- 重新部署现有的控制文件
- 重新创建所有的控制文件
- 删除控制文件

在所有的这些更改中，创建附加的控制文件、重命名或者重部署现有的控制文件、删除控制文件都可以使用以下的方式进行：

1. 关闭数据库；
2. 在操作系统一级对控制文件进行所需更改；
3. 编辑数据库初始化文件中的 CONTROL_FILES 参数，反映对控制文件所作的更改；
4. 重新启动数据库。

在某些条件下，用户可能需要重新创建所有的控制文件，如：所有的控制文件同时损坏或者需要对所有的控制文件进行更改，后者主要是更改一个在 CREATE DATABASE 中指定的永久的数据库设置，例如数据库名称、MAXLOGFILES、MAXLOGMEMBERS、MAXLOGHISTORY、MAXDATAFILES 或者 MAXINSTANCES。此时要进行的操作如下：

1. 关闭数据库；
2. 备份所有数据库的数据文件和联机重做日志文件；
3. 启动数据库至 NOMOUNT 状态；
4. 使用 CREATE CONTROLFILE语句重新创建所有的控制文件；下面是一个更改数据库名称的 CREATE CONTROLFILE 语句：

```
CREATE CONTROLFILE
    SET DATABASE prod
```

LOGFILE GROUP 1 ('log11', log12) SIZE 50K

GROUP 2 (log21 ', 'log22) SIZE 50K

NORESETLOGS

DATAFILE 'data1 'SIZE 3M, 'data2 'SIZE 3M

MAXLOGFILES 50

MAXLOGMEMBERS 3

MAXDATAFILES 200

MAXINTANCES 6

ARCHIVELOG;

5．在存储设备上备份新的控制文件；

6．更改数据库的初始化参数文件。这里涉及到的参数主要有 CONTROL_FILES 和 DB_NAME。根据上面的步骤对其进行相应的更改；

7．如果需要，恢复数据库。如果用户作为恢复来创建控制文件则恢复数据库。如果新的控制文件用 NORESETLOGS 选项来创建，用户可以完整的恢复数据库，关闭数据库恢复。如果新的数据库用 RESETLOGS 选项来创建，用户必须制定 USING BACKUP CONTROLFILES。如果用户已经丢失了联机或存档的重做日志文件或数据文件，使用恢复这些文件的程序；

8．打开数据库：

- 如果用户没有执行恢复，正常打开数据库；
- 如果用户完全执行，按步骤 7 关闭了数据库恢复，启动数据库；
- 如果用户在创建控制文件时指定 RESETLOGS，使用 ALTER DATABASE 语句，显示 RESETLOGS。

§4.4 日志文件

§4.4.1 日志文件简介

每一个数据库有两个或多个日志文件（redo log file）的组，每一个日志文件组用于收集数据库日志。日志的主要功能是记录对数

据所作的修改，所以对数据库作的全部修改是记录在日志中。在出现故障时，如果不能将修改数据永久地写入数据文件，则可利用日志得到该修改，所以从不会丢失已有操作成果。

日志文件主要是保护数据库以防止故障。为了防止日志文件本身的故障，ORACLE允许镜象日志(mirrored redo log)，以致可在不同磁盘上维护两个或多个日志副本。

日志文件中的信息仅在系统故障或介质故障恢复数据库时使用，这些故障阻止将数据库数据写入到数据库的数据文件。然而任何丢失的数据在下一次数据库打开时,ORACLE 自动地应用日志文件中的信息来恢复数据库数据文件。

日志文件分为联机重做日志文件和存档重做日志文件。

§4.4.2 联机重做日志文件

在进行数据库恢复时，最重要的结构就是联机重做日志。每个数据库实例有其自己的联机重做日志组（Redo log groups）。这些联机重做日志组，不管是否多工的，都叫作实例的联机重做线程。在典型的配置中，只有一个数据库实例访问 Oracle 数据库，所以指给出一个线程。当运行并发服务器时，多个实例并发的访问 Oracle 数据库，此时每个实例有它自己的线程，。

§4.4.2.1 简介

联机重做日志中充满了重做纪录（redo records）。重做记录有一组更改向量（change vectors）组成，每一个都描述数据库中一个块的改变。用户可以用重做记录来重建所有数据库所作的改变，甚至包括会滚段。因此，联机日志也保护回滚段。当用户恢复使用重做数据的数据库时，Oracle 读取在重做记录中的更改向量，对相应块进行更改。

重做记录在 SGA 的重做日志缓冲器中用循环方式放入缓冲器，通过 Oracle 后台进程日志书写器(LGWR)来写入联机重做日志文件之中的一个文件。不论什么时候提交事务，LGWR将从 SGA 的日志缓冲器中奖事务的重做记录写入联机重做日志文件中，并且，分配系统更改号（System Change Number，SCN）来识别每一个递交事务的重做记录。只有当所有的与给定的事务相关的重做记录安全的位于磁盘的联机文件中，系统才会通知用户进程：事务已经提交。

重做日志也能在相应的事务提交之前写到联机重做文件中。如果重做日志缓冲器已经满了，或有其他的事务提交，即使一些重做记录可能不被提交，LGWR 也将所有的重做日志缓冲器中的重做日志刷新到重做日志文件。

数据库的联机重做日志由两个以上的联机重做日志文件组成。Oracle 要求至少要有要有两个联机重做日志文件，一个用于写操作，另一个用于存档。

在循环方式中 LGWR 写联机重做日志。当当前的联机重做日志满了时，LGWR 开始写道第一个联机重做日志文件。当最后的联机重做日志文件满了时，LGWR回到第一个联机重做日志，并向之写入，并再次启动循环。

充满的联机重做日志文件对于LGWR 在重新使用时是否可用，取决于是否启用了存档：

- 如果存档没有启用(NOARCHIVELOG)模式，一个充满的联机重做日志文件在内部更改写入数据文件即可用。
- 如果存档被启用（ARCHIVELOG模式），一个充满的联机重做日志文件在内部更改已经写入数据文件，且数据文件已经存档后即对 LGWR 可用。

在任何给定的时间，Oracle只使用联机重做日志文件中的一个来存储由重做日志缓冲器写入的重做记录。LGWR正在写的联机重做日志文件叫做当前联机重做日志文件。

由实例恢复要求的联机重做日志文件叫做激活的 (active)联机重做日志文件。不是由实例恢复要求的联机重做日志文件叫做非激活的(inactive)联机重做日志文件。

如果用户启用存档，Oracle不能重新使用或覆盖一个激活的联机日志文件，直到 ARCn已经存档了它的内容为止。如果存档没有启用，在最后一个联机重做日志充满了以后，通过覆盖第一个可用的激活文件，写入继续运行。

日志切换时一个点，在这个点，Oracle结束向一个联机重做日志文件的写入，开始向另一个写入。日志切换总是在当前的联机重做日志文件完全充满了时发生，并且继续向下一个联机重做日志写入。用户也可以手工地强制进行日志切换。

在每次日志切换发生时，Oracle给每一个联机重做日志文件分配一个新的日志序列号，并且 LGWR 开始向它写入。如果 Oracle存档联机重做日志，那么，存档的日志保留它的日志序列号。循环使用的联机重做日志文件得到下一个可用的日志序列号。

每一个联机或存档的重做日志文件由它的日志序列号唯一的判别。

§4.4.2.2 规划

在配置数据库日志的联机重做日志时，用户应该考虑一下准则：

- 多工联机重做日志文件

为了减小重做日志失败的几率，Oracle提供多工实例的联机重做日志文件的能力，以防止联机重做日志文件损坏。在多工联机重做日志时，LGWR并发的将相同的重做日志信息写入多个同样的联机重做日志文件中。

- 在不同的磁盘上放置联机重做日志成员

在设置多工联机重做日志时，应在不同的盘上放置的成员。如果只是一个磁盘失败，那么组中只有一个成员对LGWR不可用，而其他成员对于LGWR仍然可以继续它的功能。

如果用户对联机日志进行存档，将在磁盘之间传播联机重做日志来清除LGWR和ARCn后台进程的连接。例如，如果用户有两个组的双工联机重做日志成员，则将每个成员放在不同的盘上，并设置存档目标为第五个盘。因此，不会再有LGWR（写入成员）和ARCn（读取成员）之间的竞争。

数据文件和联机重做日志文件也应该放在不同的盘上，来缩减在写数据块和写重做日志记录之间的竞争。

- 设置联机重做日志成员的大小

在考虑联机重做日志文件的大小时，考虑是否要存档重做日志。联机重做日志文件应该确定大小，一边充满的组可以被存档到脱机存储介质的单一单元中，并使介质中未用空间量最小。例如，假设只有一个充满的联机重做日志组可以适用于磁带，并且磁带有49%的存储空间保留未用，在这种情况下，最好稍微减少一下联机重做日志文件的大小，这样时每个磁带上可以存档两个日志组。

- 选择联机重做日志文件的数量

确定数据库实例的联机重做日志文件的合适数量的最好方法是检测不同的配置。优化的配置有最少的组，不妨碍LGWR书写重做日志信息。

在设置和更改实例的联机重做日志配置之前，应考虑可以限制联机重做日志文件数量的参数。下面的参数限制可由用户添加进数据库的联机重做日志文件的数量：

1. 参数MAXLOGFILES用于CREATE DATABASE语句中，确定每个数据库的联机重做日志文件组的最大数量。组值范围为1到

MAXLOGFILES。替换这个上限的唯一方法是重新创建数据库或它的控制文件。这样，在创建数据库之前考虑这个参数是非常重要的。

2．初始化参数 LOG_FILES可以临时地减少当前实例期间的联机重做日志文件组的最大数量。然而，LOG_FILES不能替换 MAXLOGIFLES 来增加这个极限。如果 LOG_FILES没有设置，Oracle 使用特定操作系统的缺省值。

3．参数 MAXLOGMEMBERS用于 CREATE DATABASE 语句，确定每一个组的最大成员数量。和使用 MAXLOGFILES一样，替换这个上限的唯一方法是重新创建数据库或控制文件。这样，在数据库创建之前考虑这个参数是非常重要的。

§4.4.2.3 管理联机重做日志

联机重做日志的管理主要包括联机重做日志的创建、更改和删除。

在创建数据库时，要求创建所有要求的联机重做日志文件的组和成员。例如，在联机重做日志上添加组可以纠正重做日志组的可用性问题。

要创建新的联机重做日志文件组和成员，用户必须有 ALTER DATABASE 的系统权限。创建时使用带有ADD LOGFILE子句的 SQL 语句 ALTER DATABASE。例如下面的语句给数据库添加一个新的重做日志文件组：

```
ALTER DATABASE ADD LOGFILE
( ‘ /oracle/dbs/log1c.rdo ’ , ’ /oracle/dbs/log2c.rdo ’ ) SIZE
500K;
```

用户也可以指定组的号码：

```
ALTER DATABASE ADD LOGFILE GROUP 10
( ‘ /oracle/dbs/log1c.rdo ’ , ’ /oracle/dbs/log2c.rdo ’ ) SIZE
500K;
```

如果用户要添加新的联机重做日志成员，使用带有ADD LOG MEMBER参数的 SQL 语句 ALTER DATABASE。下面的语句将向 2 号重做日志组中添加新的重做日志成员：

```
ALTER DATABASE ADD LOG MEMBER ‘ /oracle/dbs/log2b.rdo ’ TO
GROUP 2;
```

或者用户可以指定 TO 参数中组的其他成员来识别目标组。如下：

```
ALTER DATABASE ADD LOG MEMBER ‘ /oracle/dbs/log2b.rdo ’ TO
```

(‘ /oracle/dbs/log1c.rdo ’ , ’ /oracle/dbs/log2c.rdo ’);

如果当前有一些联机重做日志文件使用的磁盘设备被删除，或者在数据文件和大量的联机重做日志文件存储于一个磁盘上，应该分开来减少争用时，用户可以通过重命名联机重做日志文件来改变他们的位置。要重新命名联机重做日志成员,用户必须有ALTER DATABASE系统权限。另外，用户可能需要操作系统权限来将文件拷贝到希望的地方，以及打开和关闭数据库的权限。

在重命名联机重做日志成员之前，保证新的联机重做日志成员已经存在。重命名步骤如下：

１．备份数据库；

２．拷贝联机重做日志文件到新的地址；

３．使用带有RENAME FILE子句的 ALTER DATABASE 语句来重命名联机重做日志成员；

４．开发数据库进行一般操作；

５．备份控制文件。

在LGWR停止向一个联机重做日志组中写入，并开始向另一个联机重做日志组中写入时，发生日志切换。缺省情况下，日志切换在当前联机重做日志文件组写满时自动发生。但是用户也可以使用下面的语句来强制日志切换：

ALTER SYSTEM SWITCH LOGFILE;

要强制日志切换，用户必须有ALTER SYSTEM权限。

用户可以设置初始化参数LOG_BLOCK_CHECKSUM为TRUE来启用重做日志块的检查。如果个别的重做日志块在组中所有的成员都有问题，存档停止。在这种情况下，用户可用ALTER DATABASE … CLEAR LOGFILE来清除有错的重做日志，避免将之存档。例如：

ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP 3;

§4.4.2.4 察看联机重做日志信息

使用VLOG,VLOGFILE,V$THREAD视图来察看关于数据库的联机重做日志信息。V$THREAD 视图对于并行服务器系统管理员有特殊的用处。

下面的查询返回所有的不在并行服务器上使用的数据库的联机重做日志信息：

SELECT group#,bytes,members FROM sys.v$log;

要看组中的成员，使用一下查询：

SELECT * FROM v$logfile WHERE group#=2;

§4.4.3 存档重做日志文件

§4.4.3.1 简介

Oracle允许用户将充满的称为存档重做日志(archived redo logs)的联机重做日志文件组，保存到一个或多个脱机的目标中。存档（Archiving）是将联机重做日志转变为存档的重做日志。后台进程ARCn时存档操作自动进行。用户可以使用存档日志来进行恢复数据库、更新一个备用的数据库或者借助于日志采集器来获得关于数据库历史的信息。

一个存档的重做日志文件是一联机重做日志文件组的同一个充满的成员的一份拷贝。它包括在组中同一成员给出的重做入口，并保存唯一的重做日志序列号。

如果用户启用存档，直到已经完成存档时，LGWR才能允许重新使用，并继而重写联机重做日志组。因此，存档的重做日志由于用户起用了存档而包含所创建的组的一个拷贝。

相对于NOARCHIVELOG模式下运行的数据库，ARCHIVELOG模式具有以下优点：

1. 数据库的备份和联机与存档的重做日志文件一起，保证了用户可以恢复在操作系统和磁盘失败中的递交的事务；
2. 如果用户保存一个存档的日志，用户可以使用在数据库打开和一般系统使用中完成的备份；
3. 通过连续地把原始存档重做日志应用到备用数据库，用户可以保存一个与它的原时数据库同步的备用数据库。

在存档时，用户可以配置一个实例来自动存档充满的联机重做日志文件，或者用户可以手工的存档。为了方便和高效，自动存档一般是最好的。

§4.4.3.2 控制存档模式

作为CREATE DATABASE语句中数据库创建的一部分，用户可以设置数据库的初始化存档模式。通常，用户可以在数据库创建时使用缺省的NOARCHIVELOG模式，这是因为那时生成的重做信息不需要存档。

在创建数据库后，如果用户想把数据库的存档模式从 NOARCHIVELOG 模式变为ARCHIVELOG模式，执行下面的操作：

１．关闭数据库实例；

２．备份数据库；

３．启动一个新的实例，装入但不打开数据库；

４．使用下面的语句切换数据库的存档模式：

```
ALTER DATABASE ARCHIVELOG;
```

如果用户的操作系统允许，用户可以启用联机重做日志的自动存档。用户可以在实例启动之前或之后来启用自动存档。要在实例启动之后启用自动存档，用户必须用系统管理员权限来连接 Oracle。

要在实例启动时启用充满的组的自动存档，应该在数据库的初始化参数中包括初始化参数LOG_ARCHIVE_START，并将它设置为 true：

```
LOG_ARCHIVE_START=TRUE
```

此设置将在重起数据库后起作用。
另外，用户还可以在数据库启动后切换到自动存档模式：

```
ALTER SYSTEM ARCHIVE START;
```

反之，如果用户想禁用自动存档方式，也有如下两种办法：

１．在实例启动时禁用自动存档模式

用户可以设置数据库的初始化参数中包括 LOG_ARCHIVE_START，并将它设置为 FALSE：

```
LOG_ARCHIVE_START=FALSE
```

重起数据库后就不会使用自动存档模式了。

２．在实例启动后禁用自动存档模式

如果用户具有 ALTER SYSTEM 权限，那么用户可以使用以下的命令来禁用自动存档模式：

```
ALTER SYSTEM ARCHIVE LOG STOP;
```

§4.4.3.3 存档选项

在启用自动存档模式后，有几个选项是用户值得注意的，这些

选项对于提高数据库性能以及正确使用存档方式有一定帮助。

1．存档目标

Oracle数据库可以使用两种办法来确定存档目标，这里的目标主要是指存档文件的服务器上的存档路径，而存档文件的名称则是由数据库初始化文件中的LOG_ARCHIVE_FORMAT参数指定的。这两种办法都是在数据库初始化文件设置相应的参数：

1．第一种办法是设置LOG_ARCHIVE_DEST_n参数指定存档文件的位置，具体格式如下：

LOG_ARCHIVE_DEST_n=’LOCATION=[路径名]’

2．第二种办法是使用LOG_ARCHIVE_DEST和LOG_ARCHIVE_DUPLEX_DEST参数；

LOG_ARCHIVE_DEST=’[路径名]’

LOG_ARCHIVE_DUPLEX_DEST=’ [路径名]’

其中，LOG_ARCHIVE_DEST指定存档文件的首要存档地址，LOG_ARCHIVE_DUPLEX_DEST 指定存档文件的次要存档地址。

另外，使用参数LOG_ARCHIVE_FORMAT参数来设置存档文件的名称的格式如下：

ARCHIVE_LOG_FORMAT=[命名规则]

其中命名规则不要包括路径名,可以使用线程号和序列号来命名，规则如下：

%s 文件序列号，1,2,3…；

%t 线程号，1,2,3…；

%S 文件序列，零填充在左边；

%T 线程号，零填充在左边。

2．日志存档方式

Oracle 数据库中用户可以指定日志存档时是保存在本地的磁盘或是脱机存储介质中，还是把存档目标保存至本地或远程的、和原数据库保持同步的备用数据库中。

a) 如果用户只是保存在本地磁盘或介质中，如上面介绍的，用户通过指定数据库初始化文件参数 LOG_ARCHIVE_DEST_n 或是 LOG_ARCHIVE_DEST参数来指定存档目标；

b) 如果用户要将存档文件传送至一个备用数据库，以便同原数据库保持同步的话，用户可以通过指定 STANDBY_ARCHIVE_DEST 参数来指定创建备用数据库上的文件名。

３．指定多个ARCn进程

通过指定数据库初始化参数 LOG_ARCHIVE_MAX_PROCESS，用户允许数据库在 ARCn进程数量不足以处理当前工作量时可以自动增加的 ARCn进程的最大数量。

４．调整存档缓冲器参数

和存档缓冲器相关的参数由两个：LOG_ARCHIVE_BUFFERS 和 LOG_ARCHIVE_BUFFERS_SIZE。前者指定了存档缓冲器的个数，后者指定了每个存档缓冲器的大小。在刚创建好数据库后，用户可以先将 LOG_ARCHIVE_BUFFERS设为 1，并将 LOG_ARCHIVE_BUFFERS_SIZE 设的尽可能大，然后慢慢加大 LOG_ARCHIVE_BUFFERS，同时减小 LOG_ARCHIVE_BUFFERS_SIZE，使得系统性能达到最佳。

§4.4.3.4 存档信息

在 Oracle数据库中，用户可以通过在 Server Managerz 中运行命令：

ARCHIVE LOG LIST

或是通过数据库中的 v$视图来察看相关信息。

１．在 Server Manager 中运行命令 ARCHIVE LOG LIST 可以得到以下信息：

Database log mode:数据库是否在 ARCHIVE LOG模式下运行；

Automatic archival: 是否自动存档；

Archive Destination: 存档目标；

Oldest online log sequence: 最老的已经填充的联机重做日志组的序列号；

5

Next log sequence to archive: 下一个要存档的联机重做日志组的序列号；

Current log sequence Number: 当前联机重做日志文件的序列号。

2．在数据库中有几个 V$视图对于用户察看存档重做日志是比较有用的，包括 V$DATABASE，V$ARCHIVED，V$ARCHIVE_DEST，V$ARCHIVE_PROCESSES，V$BACKUP_REDOLOG，V$LOG，V$LOG_HISTORY。

Chapter5　Oracle 逻辑结构

单元培训目标

- **掌握** Oracle **逻辑结构的组成**
- **了解数据表空间的创建和管理**
- **了解段、范围、块的概念**
- **了解回滚段的管理**

§5.1　简介

数据库逻辑结构包含表空间、段、区域(extent)**、数据块和模式对象。本章主要介绍除模式对象外的其他逻辑结构，模式对象将在下一章进行介绍。**

一个数据库划分为一个或多个逻辑单位，该逻辑单位称为表空间（TABLESPACE）**。一个表空间可将相关的逻辑结构组合在一起。**

每个数据库可逻辑划分为一个或多个表空间。

每一个表空间是由一个或多个数据文件组成，该表空间物理地存储表空间中全部逻辑结构的数据。DBA **可以建立新的表空间，可为表空间增加数据文件或可删除数据文件，设置或更改缺省的段存储位置。**

ORACLE **数据库中一表空间是由一个或多个物理数据文件组成，一个数据文件只可与一个表空间想联系。当为一表空间建立一数据文件时，**ORACLE **建立该文件，分配指定的磁盘空间容量。在数据文件初时建立后，所分配的磁盘不包含任何数据。表空间可以在线或离线。 在** ORACLE **中还允许单独数据文件在线或离线。**

ORACLE **通过段、范围和数据块逻辑数据结构可更细地控制磁盘空间的使用。**

段（SEGMENT**）包含表空间中一种指定类型的逻辑存储结构，是由一组范围组成。在** ORACLE **数据库中有几种类型的段：数据段、牵引段、回滚段和临时段。**

ORACLE **对所有段的空间分配，以范围为单位。**

一个范围（EXTENT**）是数据库存储空间分配的一个逻辑单位，它由连续数据块所组成。每一个段是由一个或多个范围组成。当一段**

中间所有空间已完全使用时，ORACLE 为该段分配一个新的范围。

为了维护的目的，在数据库的每一段含有段标题块（segment header block）说明段的特征以及该段中的范围目录。

数据块（data block）是 ORACLE 管理数据文件中存储空间的单位，为数据库使用的 I/O 的最小单位，其大小可不同于操作系统的标准 I/O 块大小。

§5.2 表空间

Oracle数据库中的表空间按照其重要性和功能不同可以区分为非 SYSTEM 表空间和 SYSTEM 表空间。每一个 ORACLE 数据库包含有一个名为 SYSTEM 的表空间，在数据库建立是自动建立。在该表空间中总包含有整个数据库的数据字典表。最小的数据库可只需要 SYSTEM 表空间。该表空间必须总是在线。表信息和存储的 PL/SQL 程序单元（过程、函数、包和触发器）的全部存储数据是存储在 SYSTEM 表空间中。如果这些 PL/SQL 对象是为数据库建的，DBA 在 SYSTEM 表空间中需要规划这些对象所需要的空间。

DBA 可利用表空间作下列工作：

- 控制数据库数据的磁盘分配；
- 将确定的空间份额分配给数据库用户；
- 通过使单个表空间在线或离线，控制数据的可用性；
- 执行部分数据库后备或恢复操作；
- 为提高性能，跨越设备分配数据存储。

§5.2.1 创建表空间

无论在任何数据库中，第一个、且从数据库创建时就存在的表空间是 SYSTEM 表空间。并且在数据库中的第一个数据文件也总是定位在 SYSTEM 表空间中。SYSTEM 表空间是一个数据库中最重要的表空间，用户不能创建第二个 SYSTEM 表空间，同时也不能删除此表空间。

Oracle8i 中，用户可以创建两种管理类型的非 system 表空间：

- 依靠 SQL 字典表跟踪磁盘空间利用的字典管理类型表空间，是创建表空间的缺省类型，在 Oracle8i 版本以前一直使用；
- 使用位图而非 SQL 字典列表来跟踪已经使用和尚未使用空间

的本地管理表空间，在 Oracle8i 版本中建议用户使用。

在数据库创建后，如果用户具有 CREATE TABLESPACE权限，并且在 SYSTEM 表空间获得了包括 SYSTEM 回滚段在内的两个以上的回滚段时，用户打开数据库并可以使用以下命令来创建一个新的非 SYSTEM 表空间：

CREATE [TEMPORARY] TABLESPACE [tablespace name]
DATAFILE/TEMPFILE [file] [option];

如果使用了 TEMPORARY 关键字，表示用户要创建一个临时表空间，这种表空间一般包含一个或多个临时文件，用来在一个连续的进程中存储数据库中的模式对象，但是在临时表空间中用户不能显式的创建数据库对象。这种表空间可以改进多重排序的并行性，减少它们的总开销，或者完全避免 Oracle 空间管理操作。临时表空间可以是字典类型或是本地类型的。

如果用户需要创建一般的表空间，在一些特定的操作系统中，用户必须先创建该表空间将要使用的数据文件，然后再创建表空间时指定该数据文件。Oracle 将按照给定的条件自动的定位并格式化数据文件。

如果用户需要创建一个本地管理表空间，那就必须制定 CREATE TABLESPACE中的 LOCAL 项。用户将有两个选项：

- AUTOALLOCATE：Oracle 自动管理区域，当表空间中预定包含各不相同的数据库对象，这些对象需要容量也各不相同的区域时使用，这是用户空间会被浪费掉一些。使用以下命令：

  CREATE TABLESPACE [tablespace] DATAFILE [other options]
  LOCAL AUTOALLOCATE;

- UNIFORM: 如果用户需要严格控制不用的表空间，并且能够准确的预知分配给数据对象的空间以及区域的数量和容量时适用。这样可以保证在用户表空间中不会有无用的空间，使用以下格式的语句：

  CREATE TABLESPACE [tablespace] DATAFILE [other options]
  LOCAL UNIFORM [size] K/M;

在缺省状态下，刚创建的表空间为可读写的。以后根据用户需求，用户可以手工将其该为只读的。

§5.2.2 更改表空间

对表空间的更改主要包括四种类型：更改表空间存储参数、更改表

空间地址分配、使表空间联机或脱机、更改表空间为读写或只读

§5.2.2.1 更改存储参数

表空间相关的存储参数有以下一些：

INITIAL	以字节（K 或 M）定义数据段里第一个区域的尺寸；
NEXT	定义第二区域字节的大小（K 或 M）；
PCTINCRESE	第二区域后各个区域的增长率；
MINEXTENTS	表空间中数据段最初创建时分配的区域数目
MAXEXTENTS	确定数据库可以使用的区域数量的最大值。可以为 UNLIMITED。

存储参数影响两个方面：访问存储在数据库中的数据需要的时间，数据库中空间的使用效率。通过改动 NEXT、MAXEXTENTS、PCTINCREATE 参数（在表空间创建后 INITIAL 和 MINEXTENTS 参数不能改动），可以控制表空间中自由空间存储碎片。使用如下格式语句：

```
ALTER TABLESPACE [tablespace] DEFAULT STORAGE (
 [ NEXT [size]k ]
 [ MAXEXTENTS n ]
 [ PCTINCREASE m ]
```

§5.2.2.2 更改表空间地址分配

在 Oracle 数据库中，表空间区域是由一些邻接的自由存储块组成的。如果一些区域没有存储数据，就被称为自由区域。这些区域在尺寸上并不一定都相同，在为一个表空间数据段分配新的区域，Oracle 数据库先在所有的自由区域中找到一块足够大的区域，用来存储数据。如果在现有的自由区域中找不到一块足够大的区域，那么 Oracle 将找到一块较大的区域，并合并相邻的自由区域来形成一个新的自由区域，以便能用来存储数据。如果合并以后，新的自由区域仍然不能满足要求，Oracle 数据库就必须寻找另一块较大的区域，并合并相邻自由区域。所以，如果在数据库的表空间中充满了尺寸较小的自由区域，在为大批量数据寻找存储区域时会花费较多的时间。

反之，如果在 Oracle 数据库中，相邻的自由区域都已经被合并，那么在寻址时就可以较容易地找到大小合适的自由区域。所以，在

Oracle数据库，可以定时运行一下命令来合并自由区域：

ALTER TABLESPACE [tablespace] COALESCE;

在执行合并命令时，应注意：

1. 本地管理的表空间不需要合并自由空间；

2. 如果 PCTINCREASE 不为 0，用户也不需要运行合并命令，因为后台进程 SMON 会定期合并相邻的自由空间区域。但是如果 PCTINCREASE 为 0，用户就必须定期运行合并命令，同时还可以减小 SMON 进程的总开销；

3. 当一个数据段因为某种原因被切断，此时不管表空间 PCTINCREASE 是否为 0，SMON 也会运行限定形式的合并操作来重新形成新的数据段；

4. COALESCE选项是排它的,当执行合并操作时不能同时指定其它选项。

§5.2.2.3 使表空间脱机或联机

表空间利用增加数据文件扩大表空间，表空间的大小为组成该表空间的数据文件大小的和。

DBA 可以使 ORACLE 数据库中任何表空间（除 SYSTEM 表空间外）在线（ONLINE）或离线（OFFLINE）。表空间通常是在线，以致它所包含的数据对数据库用户是可用的。当表空间为离线时，其数据不可使用。在下列情况下，DBA 可以使其离线。

- 使部分数据不可用，而剩余的部分允许正常存取；
- 执行离线的表空间后备；
- 为了修改或维护一应用，使它和它的一组表临时不可用。

包含有正在活动的回滚段的表空间不能被离线，仅当回滚段不正在使用时，该表空间才可离线。

在数据字典中记录表空间的状态，在线还是离线。如果在数据库关闭时一表空间为离线，那么在下次数据库装配和重新打开后，它仍然保持离线。

当出现某些错误时，一个表空间可自动地由在线改变为离线。通过使用多个表空间，将不同类型的数据分开，更方便DBA 来管理数

据库。

在 Oracle 数据库中，使用以下语句使表空间脱机：

ALTER TABLESPACE [tablespace] OFFLINE [option];

这里用户可以采用的选项有：

- NORMAL

当一个表空间的所有的数据文件都不存在错误时，可以采用 NORMAL 选项来对它进行脱机，脱机时，Oracle 为数据文件生成正常检验点；

- TEMPORARY

如果一个表空间的数据文件存在错误，那么用户就不能再使用正常选项进行脱机，而必须使用 TEMPORARY 选项，改选项在对数据文件进行脱机时，使用临时脱机次序为每个数据文件生成检验点。在对表空间进行临时脱机前，如果该表空间中的数据文件因存在错误已经脱机，那么在重新使表空间联机时需要进行介质恢复；

- IMMEDIATE

在 ARCHIVELOG 模式下，用户也可以采用 IMMEDIATE 对表空间进行脱机。此时 Oracle 不会对其数据文件生成任何检验点，所以在重新联机表空间时需要进行介质恢复；

- FOR RECOVER

使用表空间的时间点恢复时可以采用此选项。

在使用脱机命令时，有几点需要注意的：

1. SYSTEM 表空间不能进行脱机；
2. 如果某个表空间包含正在使用的回滚段，那么它也不能被脱机；
3. 如果该表空间正在作为临时表空间或有排序区域在使用，那么在脱机前，用户需要更改这些空间分配。

在需要时，用户也可以重新让某个脱机的表空间重新联机：

ALTER TABLESPACE [tablespace] ONLINE;

§5.2.2.4 使表空间只读

在 Oracle数据库创建之后，缺省的所有表空间都是可读可写的。但是在一些情况下，用户可能希望把现有的表空间进行脱机：

- 某表空间中包含大量的静态数据,并且在一段时间内这些数据不会被改动，为了节省备份和恢复的时间和空间，希望对这些表空间只做一次备份；
- 保护数据，希望所有人都只能察看某个表空间上的数据，但不能进行更改；
- 希望把某表空间的数据移至另一数据库使用。

在以上情况下，用户都可以通过把表空间设为只读来实现：

ALTER TABLESPACE [tablespace] READ ONLY;

在使用以上语句时请注意：

1. 更改前表空间必须处于联机状态；
2. SYSTEM 表空间不能更改为只读；
3. 任何包含正在使用的回滚段的表空间不能被更改为只读；
4. 请勿将已更改的表空间包含在联机备份中，因为在备份后所有数据文件的主文件将被更改；
5. 用户可以通过数据字典 V$SQLAREA, V$TRANSACTIONS 来观察更改是否结束。

用户也可以使用下面的语句使表空间重新可以写入：

ALTER TABLESPACE [tablespace] READ WRITE;

§5.2.3 删除表空间

除了 SYSTEM 表空间外，Oracle数据库中的所有的表空间都可以被删除，使用以下语句进行删除：

DROP TABLESPACE [tablespace] [options];

这里经常的选项有：

- INCLUDING CONTENTS

使用这个选项将删除表空间及其包含所有的数据库对象，如：表、

视图和序列等；

- CASCADE CONTRANTS

如果表空间外的表的完整性约束引用了欲删除表空间上的数据，那么使用这个选项将把这些完整性约束一并删除。

在删除请注意：

1. 用户必须具有 DROP TABLESPACE权限；

2. SYSTEM 表空间不能被删除；

3. 包含正在使用的回滚段的表空间不能被删除 ；

4. 包含正在使用的数据段的表空间不能被删除 ；

5. 表空间删除后，仅是在控制文件中删除了该表空间的相关信息，在数据字典中仍然可以看到该表空间，不过其状态为INVALID。

§5.2.4 察看表空间信息

Oracle数据库中可以用来察看表空间信息的数据字典有：

- V$TABLESPACE

所有表空间的名称及相关信息；

- DBA_TABLESPACE（USER_TABLESPACE）

所有(用户能访问的) 表空间的描述；

- DBA_SEGMENTS(USER_SEGMENTS)

所有(用户能访问的)表空间内的数据段的信息；

- DBA_EXTENTS(USER_EXTENTS)

所有(用户能访问的)表空间内数据区域的信息；

- DBA_FREE_SPACE(USER_FREE_SPACE)

所有(用户能访问的)表空间内自由区域的信息；

- V$DATAFILE

所有数据文件的信息；

- V$TEMPFILE

所有临时文件的信息；

- DBA_DATA_FILES

属于表空间的数据文件的信息；

- DBA_TEMP_FILES

属于临时表空间的临时文件信息；

§5.3 段、区域及块

ORACLE 通过段、区域和数据块逻辑数据结构可更细地控制磁盘空间的使用。

段

段（SEGMENT）包含表空间中一种指定类型的逻辑存储结构，是由一组范围组成。在 ORACLE 数据库中有几种类型的段：数据段、牵引段、回滚段和临时段。

数据段：对于每一个非聚集的表有一数据段，表的所有数据存放在该段。每一聚集有一个数据段，聚集中每一个表的数据存储在该段中。

索引段：每一个索引有一索引段，存储索引数据。

回滚段：是由 DBA 建立，用于临时存储要撤消的信息，这些信息用于生成读一致性数据库信息、在数据库恢复时使用、回滚未提交的事务。

临时段：当一个 SQL 语句需要临时工作区时，由 ORACLE 建立。当语句执行完毕，临时段的区域退回给系统。

ORACLE 对所有段的空间分配，以区域为单位。

区域

一个区域（EXTENT）是数据库存储空间分配的一个逻辑单位，它由连续数据块所组成。每一个段是由一个或多个区域组成。当一段中间所有空间已完全使用时，ORACLE 为该段分配一个新的区域。

为了维护的目的，在数据库的每一段含有段标题块（segment header block）说明段的特征以及该段中的区域目录。

数据块

数据块（data block**）是** ORACLE **管理数据文件中存储空间的单位，为数据库使用的** I/O **的最小单位，其大小可不同于操作系统的标准** I/O **块大小。**

数据块的格式由公用的变长标题、表目录、行目录、未用空间和行数据构成。

§5.4 回滚段

§5.4.1 简介

回滚段是 Oracle **数据库中一种比较特殊的段,它仅仅是用来临时存储事务处理，使每个事务处理可以被回退或提交。在一个数据库刚创建后，系统中** SYSTEM **表空间中包含一个名为** SYSTEM **的回滚段。缺省地，其他表空间中不含有回滚段。**

在创建数据库时，或者在创建数据库后，回滚段的使用都会对整个数据库的性能有重要影响，因此在使用时用户需注意：

1. **在创建数据库时，应对需要创建的回滚段的大小及数量有一个规划。理论上讲，较多的回滚段将会减少争用分配回滚段的可能，并改进系统性能；**

2. **在数据库初始化参数中，**TRANSACTIONS **规定了在** Oracle **数据库中预期的同时并发的事务处理数，**TRANSACTIONS_PER_ROLLBACK_SEGMENT **则指定了每个回滚段至少必须处理的事务处理量，因此创建数据库后，你应该创建**(TRANSACTIONS/PER_TRANSACTION_ROLLBACK_SEGMENT)**个回滚段；**

3. **如果数据库运行于并行服务器模式，那么可以在初始化参数文件中** ROLLBACK_SEGMENTS **参数后指明某实例能使用的回滚段，称为实例的私有回滚段。反之，每个实例都可以访问的回滚段称为公用回滚段。如果数据库不处于并行服务器模式，那么所有的回滚段都是一样的，没有公有和私有之分；**

4. **一般来说，小的回滚段被放置在主存储器甚至是** SGA **中，这样的回滚段使用时由于具有较小**I/O**，将获得较好的数据库性能；但是如果数据库中存在较大的事务处理量时，如果此时仍然使用较小的回滚段，那么将回增加“**Snap too old**”的可能性。因此应尽量使回滚段的大小与经常产生的事务处理量一致。如果同时存在小的事务处理和大的事务处理，那么就应该建立多个**

6

尺寸不一的回滚段；

5．在事务处理量较稳定的情况下，应该尽量使所有回滚段的尺寸相等，一般来说，10 个到 20 个大小相同的回滚段能使数据库获得较好的 I/O 性能；

6．尽量把回滚段和数据段分开，不要把回滚段和数据段放置在同一个表空间中。这样有利于对数据段所属的表空间进行操作，如脱机和更改为只读等。含有回滚段的表空间必须始终保持联机。

§5.4.2 创建回滚段

使用下面的语句创建一个新的回滚段：

```
CREATE [PUBLIC / PRIVATE] ROLLBACK SEGMENT [rollback segment]

TABLESPACE [tablespace]

[storage parameters];
```

其中，storage parameters 的格式如下：

```
STORAGE(

  INITIAL [initial_size]K

  NEXT [next_size]K

  OPTIMAL [optimal_size ]M

  MINEXTENTS n

  MAXEXTENTS m)
```

其中，INITIAL 是分配给回滚段的初始区域，NEXT 是分配给回滚段的第二个区域大小，OPTIMAL 是回滚段的优化大小，MINEXTENTS 是回滚段创建时的最小区域数，MAXEXTENTS 是回滚段能分配的最大区域数（包括初始区域数）。

在使用时请注意：

1．最好将 INITIAL 和 NEXT 设为一样的值,以保证回滚段中所有的区域尺寸相同；

2．创建时 INITIAL 尺寸应该尽量大，避免经常需要分配区域；

3．一般来说，M=20能获得最佳性能；

4．避免设置M=UNLIMITED，否则如果在一段程序错误的循环导致回滚段或者数据文件不必要的扩展，将对数据库性能产生影响；

5．刚创建回滚段后，该回滚段是脱机的，必须使其首先联机，然后才能使用。

§5.4.3 更改回滚段

在使用回滚段过程中，用户会经常对回滚段进行改动。这要求用户具有 ALTER ROLLBACK SEGMENT 权限。这些可能的改动包括：更改回滚段的存储参数、收缩回滚段的尺寸和更改回滚段的状态。

回滚段创建后，用户可以对回滚段的存储参数进行更改。这在调试数据库性能时经常要用到这种更改。比如改动 USER_RS1 回滚段的 OPTIMAL：

```
ALTER ROLLBACK SEGMENT USER_RS1
STORAGE(OPTIMAL 4M);
```

在回滚段联机的前提下，用户可以将回滚段的大小缩回至某个特定值，如：

```
ALTER ROLLBACK SEGMENT rbs1 SHRINK TO 100K;
```

这将把回滚段 rbs1 的尺寸回滚到 100K。

只有联机的回滚段才能对事务处理有效，但是用户也有可能需要使一个回滚段脱机：

```
ALTER ROLLBACK SEGMENT [rollback segment] OFFLINE;
```

在脱机前，该回滚段必须是联机的。如果一个回滚段包含有疑问或被恢复的分布式业务处理数据，那么此时该回滚段的状态为 PARTLY AVAILABLE。

使回滚段联机：

```
ALTER ROLLBACK SEGMENT [rollback segment] ONLINE;
```

启动数据库后，在数据库初始化参数 ROLLBACK_SEGMENTS 指定的回滚段将被自动联机。

§5.4.4 删除回滚段

在用户具有 DROP ROLLBACK SEGMENT**权限的前提下，如果一个回滚段是联机的，那么我们就可以使用以下语句将之删除：**

DROP ROLLBACK SEGMENT [rollback segment];

但是请注意，删除后在数据字典中仍然可以看到此回滚段，必须重其数据库后才能使其消失。

Chapter6　模式对象

单元培训目标

- **了解 Oracle 中的模式与模式对象**
- **掌握数据表的创建与管理**
- **掌握索引的创建与管理**
- **了解簇的概念与使用**
- **了解散列簇的概念**
- **掌握视图的创建**
- **掌握序列和同义词的使用**

§6.1　模式与模式对象

一个模式(schema)**为模式对象**(scehma object)**的一个集合，每一个数据库用户对应一个模式。模式对象为直接引用数据库数据的逻辑结构，模式对象包含如表、视图、索引、簇、序列、同义词、数据库连接、过程和包等结构。模式对象是逻辑数据存储结构，每一种模式对象在磁盘上没有一个相应文件存储其信息。**

一个模式对象逻辑地存储在数据库的一个表空间中，每一个对象的数据物理地包含在表空间的一个或多个数据文件中。对于某些对象，如表、索引和簇，你可以指定在表空间的数据文件中 Oracle 应为其分配多大的空间。

一个模式被一个数据库用户所拥有，并且和该数据库用户具有相同的名称。

§6.2　数据表

§6.2.1 简介

表是 Oracle 数据库中数据存储的基本单位。数据以某行某列的方式存储在表中。定义数据表时，用户必须定义表名和表中的所有列。对于每一列，用户必须指定列名、数据类型（VARCHAR2, DATE, NUMBER**）和宽度。有些数据类型的宽度已经被预先指定了，如 DATE；而对于使用 NUMBER 类型的列，用户必须指定其精度和比例。表中的一行就是所有列的信息的一个集合。**

在表中，你可以为每一个列指定一个约束规则，这些规则称为完整性约束。最普遍的完整性约束是指定一个列的数值不能为空（NOT NULL），要求该表中的每条记录在存储时必须有一个值。

表创建后，用户可以使用 SQL语句把数据按照要求插入到该表中，然后用户就能使用 SQL语句对表中的数据进行查询、更改和删除了。

§6.2.2 创建表

如果用户具有 CREATEBLE权限，那么用户可以在自己的模式下创建一个新表；如果用户具有 CREATE ANY TABLE权限，那么用户可以在其他用户的模式下创建一张新表。此外，表的拥有者必须必须有该表所在表空间的配额，或者具有 UNLIMIIED TABLESPACE的系统许可权；如果用户想把新创建的表让别人访问，用户必须赋予 EXECUTE并附带 GRANT OPTION，或者具有 EXECUTE ANY TYPE 系统权限并有 ADMIN OPTION。如果没有这些权限，用户不可能赋权给其他用户来访问这张表。

用户可以一下的语句来创建表：

CREATE [GLOBALTEMPORARY] TABLE [schema.]table

(column_name column_type column_constraints,

...)

[temporary options]

[physical properties]

[table properties];

其中可以使用的选项有：

一. GLOBAL TEMPORARY

创建一张临时表，该表对所有用户都是可见的，但是临时表中的数据只有当会话向表中插入数据时才是可见的。

二. Temporary options

这里可以和创建临时表搭配的选项有两个：

1. ON COMMIT DELETE ROWS: 说明临时表是事务指定，每次提交后 Oracle 将截短表（删除所有行）；

2. ON COMMIT PRESERVE ROWS: 说明临时表是会话指定，当中断会话时 Oracle将截短表（删除所有行）.

三. Physical options

这里最常用的表物理选项包括：

PCTFREE: 用于设置存储块中备用空间的比率。存储块中数据行中的更新需要保留一定的空间。此选项将标明表的数据段中将保持多大比率的空间自由和可用。在指定时需注意较小的自由空间能减小整个数据库的储存空间，但是将会降低更新操作的性能；

PCTUSED: 按照 PCTFREE 的设定，数据块被认为是满了之后，Oracle不会考虑向表中插入新的行，直到存储块的使用率下降到 PCTUSED 参数值之下。

INITRANS: 根据 DML 事务项的数量，以在最初为其在数据库头部为其保留空间。保留空间位于关联数据或者索引段中所有数据块的头部。数据块的缺省值是１，簇和索引的缺省值是２。

MAXTRANS: 在多重事务同时访问相同数据块的行列时，在存储块中将为每个 DML 事务项分配空间。一旦 INITRANS保留的空间被耗尽，如果可以，将在存储块自由空间之外为额外事务项分 配空间。一经分配，这个空间将永久性的称为存储块头的一部分。MAXTRANS参数限制了可以同时使用数据块种数据的事务项的数量。

TABLESPACE tablespace: 当用户具有适当的许可权和表空间配额，你可以指定新表所处的表空间。这样，可以使数据库的性能增加，减少管理数据库所需要的时间。

四. Table properties

在创建新表时，用户可以指定表的储存参数，以优化数据库的性能，减少数据库必需的储存空间，常用的表储存参数有：

１. INITIAL

创建数据段时分配的第一个区域的容量，以字节为单位。这个参数不能在 ALTER TABLE 中指定。

２. NEXT

数据段中增加的下一个区域的容量，以字节为单位。第二个区域的容量等于 NEXT的初始设置。从这以后，NEXT 的值设置为前一容量乘以(1+PCTCREASE / 100)。

3．PCTCREASE

相对于最后增加的区域，数据段每个新增区域大小的增大比率。如果 PCTCREASE 为 0，那么所有的区域容量相同。

4．MINEXTENTS

当创建数据段时分配区域的总数。这个参数是为创建时大的空间分配而设，即使连续的存储空间不可用。

5．MAXEXTENTS

用于指定数据段分配的区域总数，包括第一个区域。

6．FREELIST GROUPS

用户创建的数据库对象的自由列表组的数量。

7．FREELISTS

表中每一自由组的自由表数量。

§6.2.3 改变表

如果用户具有 ALTER TABLE 或者 ALTER ANY TABLE 权限，那么用户可以使用下面的命令变更表：

```
ALTER TABLE table

[ADD (new_column_name new_column_type
new_column_constraints,

... )]

[MODIFY [old_column_name new_type new_constraints]]

[MOVE [table properties]]

[DROP (old_column_name, ... )];
```

这里需要指出的是：

1. 用户只能增加现有列的长度，不能将其缩小，除非表中没有数据；

2. 改变表存储参数后，变更将应用于表中已分配的和将来要分配的数据。参数改变后，变更不是马上在表上进行，只有才需要的时候才进行；

3. 当 INITRANS 改变后，新的参数值只应用于以后增加地新数据块，而 MAXTRANS 的新值将应用于表中所有的数据块；

4. 参数 INITIAL 和 MINEXTENTS 不能改变。

§6.2.4 删除表

如果用户具有 DROP TABLE 或者 DROP ANY TABLE 权限，那么用户可以删除不再需要的表：

DROP TABLE table [CASCADE CONTRAINTS];

如果表中的主关键字或者唯一关键字涉及到其他表的外部关键字，而用户需要删除关联表中的外部关键字，那么用户必须指定 CASCADE CONTRANTS 选项。

§6.2.5 分区表

当一张表中的数据很大时，用户可以把表中的数据分解为一些容易管理的数据块，每个块称为一个分区(PARTITION)或子分区（SUBPARTITION）。这时候，表中的的一些参数是共享的，如逻辑属性和行、约束定义，但是这些分区可以拥有不同的物理属性，如不同的分区属于不同的表空间。

将表进行分区的好处在于：

1．减少在多个分区破坏数据的可能性；

2．独立的备份和恢复各个分区；

3．控制分区到驱动器的映射；

4．提高易管理性、实用性和使用性能。

用户可以使用的分区方法有三种：

1．范围分区

使用范围分区法将表中的行映射到基于列值范围的分区，这种分区在处理数据在一定的逻辑范围内均匀存在的表时性能很好，但是如果数据并不是均匀分布，就必须考虑使用别的分区方法了。例如：

```
CREATE TABLE cux_gl_balance
  ( code_combination_id      NUMBER,
```

period_year NUMBER,

period_num NUMBER,

period_net_dr NUMBER,

period_net_cr NUMBER,

balance NUMBER)

PARTITION BY RANGE(period_year, period_num)

(PARTITION 2001_spring VALUES LESS THAN (2001, 4) TABLESPACE cuxd1,

PARTITION 2001_summer VALUES LESS THAN (2001, 7) TABLESPACE cuxd2,

PARTITION 2001_automn VALUES LESS THAN (2001, 10) TABLESPACE cuxd3,

PARTITION 2001_winter VALUES LESS THAN (2002, 1) TABLESPACE cuxd4);

上例中将把２００１年各个季度的数据放在不同的表空间中，有利于用户对数据的管理；

２．散列（hash）分区

如果数据不易进行范围分区，但是出于性能原因又一定要进行分区的时候，用户可以考虑使用散列分区。这种分区法根据分区健的散列值来将行映射到分区上，也能使数据平均分配，提高系统性能。例如：

CREATE TABLE cux_users

(user_id NUMBER,

user_name VARCHAR2(150))

PARTITION BY HASH(user_id)

PARTITIONS 4

STORE IN (cuxd1, cuxd2, cuxd3, cuxd4);

此时散列表中的分区被保留在不同的表空间中。

3．复合分区

经常使用而且效果较好的一种方法是先使用范围分区法分区，然后再在各个子分区中使用散列分区法来存储数据，这样做存储数据对历史数据和带数据都比较理想。例如：

```
CREATE TABLE cux_users (
  User_id  NUMBER,
  User_number NUMBER,
  User_name VARCHAR2(1000))
  PARTITION BY RANGE(user_number)
  SUBPARTITION BY HASH(user_name)
  SUBPARTITIONS 8 STORE IN (cuxd1, cuxd2, cuxd3, cuxd4)
  (PARTITION p1 VALUES LESS THAN (1000),
   PARTITION p2 VALUES LESS THAN (2000),
   PARTITION p3 VALUES LESS THAN (MAXVALUE)),
```

§6.3 索引

§6.3.1 简介

索引(index)是与表和簇相关的一种选择结构。索引是为提高数据检索的性能而建立，利用它可快速地确定指定的信息。ORACLE 索引为表数据提供快速存取路径。索引适用于一范围的行查询或指定行的查询。

索引可建立在一表的一列或多列上，一旦建立，由 ORACLE 自动维护和使用，对用户是完全透明的。索引是逻辑地和物理地独立于数据，它们的建立或删除对表没有影响，应用可继续处理。索引数据的检索性能几乎保持常数，而当一表上存在许多索引时，修改、删除和插入操作的性能会下降。

Oracle 中的索引模式包括：B*-树索引（目前最常用的）、B*-树簇索引、散列簇索引，反向关键字索引和位图索引。此外，Oracle 还支持基于函数的索引和专门用于应用程序和字体盒的域索引。

索引有唯一索引和非唯一索引。唯一索引保证表中没有两行在定义索引的列上具有重复值。ORACLE 在唯一码上自动地定义唯一索引实施 UNIQUE 完整性约束。

组合索引是在表的某个列上所建立的一索引。组全索引可加快 SELECT 语句的检索速度，在其 WHERE 子句中可引用组合索引的全部或主要部分 。所以在定义中给出列的次序，将经常存取的或选择最多的列放在首位。

在建立索引时，将在表空间自动地建立一索引段，索引段空间分配和保留空间的使用受下列方式控制：

索引段范围的分配常驻该索引段的存储参数控制。

其数据块中未用空间可受该段的 PCTFREE 参数设置所控制。

§6.3.2 创建索引

用户可以为数据表或者簇建立索引，本节主要介绍对数据表建立索引的情况。

如果用户具有 CREATE INDEX 权限，或者编入索引的表属于用户自己的模式，或者用户被赋予对目标表具有 INDEX 授权，那么用户可以在自己的模式下的表建立索引；如果用户具有 CREATE ANY INDEX 权限，那么用户可以为别人模式下的表创建索引。

用户可以明式和暗式地创建一个索引。暗式创建索引只能是在用户指定了一张表的情况下，系统会自动为该表创建一个唯一性索引，该索引一般以 SYS 开头，后面接一个序列号。

在其他情况下，用户必须使用 SQL 语句为表创建索引。该语句的格式如下：

```
CREATE [UNIQUE/BITMAP] INDEX [SCHEMA.]index
  ON [SCHEMA.]table/cluster [alias]
  (column_name/column_expression [ASC/DESC],
    ... )
  [other index clause]
```

其中，需要注意的是：

1. 关键字 UNIQUE 用于申明该索引是唯一性索引，索引字段在表中不能重复（空出外），但它不使用于位图索引和域索引；

7

2．关键字 BITMAP 用于申明该索引是位图索引，即把表中的记录与一作为位图的关键字相关联，位图索引中的每个位都对应着表中的一条记录的 rowid。只要在位图中设置了一个位，就表示在表中的某条记录的 rowid 对应着相应的键值。位图索引适合于并发事务较少的表。

3．用户使用基于函数的索引。假如用户经常查询某个子公司的销售数据，可以考虑建立下面的索引：

CREATE INDEX com1_sales_index ON sales(get_Org='A 公司');

4．在最后，其它可以使用的参数还有：

- 全局分区参数：按照索引字段不同的方位对索引进行分区，关于对索引分区的描述请参见后面的小节；

- 本地分区参数：按照和目标表中进行分区的相同字段，按照和目标表相同的范围和规则对索引进行分区；

- 存储参数：和表类似，用户可以指定索引的 PCTFREE, PCTUSED, INITRANS, MAXTRANS 等参数；

- 此外，还有一些特别的参数值得一提：

① ONLINE: 以前，用户在创建索引时不能对基表进行 DML 操作，现在，用户可以指定联机，使创建索引和更改集表同时进行称为可能；

② COMPRESS: 使用关键字压缩创建索引使用户能够消除关键列标头值的重复出现。压缩时，索引关键字被断开为前缀和后缀，所有的后缀通过共享前缀而使存储空间得到节省。

③ COMPALLEL: 用户可以并行的创建索引，因为多个进程索引合作创建索引，比单个进程顺序的创建索引快。

④ NOLOGGING: 使用 NOLOGGING 参数，用户可以使创建索引时生成的重做日志最小。

§6.3.3 改变索引

在满足以下条件之一时，用户可以改变重建索引，改变索引的存储参数和物理属性，但是用户不能更改索引的数据列结构:

1． 用户模式中包含该索引；

2． 用户具有 ALTER ANY INDEX 的系统授权。

改变索引时使用如下的语句：

ALTER INDEX [schema.]index [option]

这里比较常用的选项主要有：

1．改变存储参数

这里可以改变的参数有 INITRANS, MAXTRANS, MAXEXTENTS 等参数，但是用户不能改变 INITIAL 和 MINEXTENTS，如：

```
ALTER INDEX emp_ename
  INITRANS 5
  MAXTRANS 10
  STORAGE(PCTCREASE 50);
```

2．改变并行参数 PARALLEL；

3．改变 LOGGING, NOLOGGING 等参数；

4．重建 INDEX, 如：

```
ALTER INDEX artist_ix REBUILD PARTITION p063
NOLOGGING;
```

5．使索引不可用：

```
ALTER INDEX index1 UNUSABLE;
```

6．收集统计数据

```
ALTER INDEX emp_index REBUILD COMPUTE
STATICS;
```

§6.3.4 删除索引

要删除一个索引，索引必须存在于用户的模式中，或者用户具有 DROP ANY INDEX 的系统权限。在这个前提下，用户可以使用以下的 命令对索引进行删除：

DROP INDEX [schema.]index;

这里，请注意:

1. 只有当索引是明式地使用 CREATE INDEX 创建时，索引残能被删除；对于在创建表时由于申明了 UNIQUE 或者 PRIMARY KEY 而被建立的索引，用户必须先禁用或者删除这种完整性约束，然后才能删除索引；

2. 删除表时，该表上所有的索引都将被删除。

§6.3.5 分区索引

和表一样，索引也可应使用分区或者子分区来共享相同的索引选项，同时又具有不同的物理属性。索引也可以使用和表一样的三种分区方法：范围分区、散列分区和复合分区。

但是对于全局索引来说，要么不使用分区，要么只能使用范围分区，但是它也有它的优点: 它可以定义在任何类型的分区或者分区表上。它通常具有更多的维护要求。 下面是一个建立分区的全局索引的例子：

```
CREATE INDEX stock_ix ON stock_xactions
  (stock_symbol, stock_series)
   GLOBAL PARTITION BY RANGE (stock_symbol)
      (PARTITION  VALUES LESS THAN ('N') TABLESPACE
ts3,
       PARTITION VALUES LESS THAN (MAXVALUE)
TABLESPACE ts4);
```

为了反映基表的结构而构造了本地索引，它被基表均匀分区，即意味着在基表的相同列上对索引分区，创建相同数量的分区和子分区，并给它们与基表的分区对应相同的分区边界。对于本地索引，在分区被维护活动时自动维护索引分区。这保证了索引一直被基表均分。在对本地索引进行分区时，三种分区方法都可以使用，下面即是使用三种方法对本地索引进行分区的例子：

1. 范围分区：

```
  CREATE INDEX emp_dept ON emp(deptno)
 LOCAL (PARTITION p1 TABLESPACE tbs1,
        PARTITION p2 TABLESPACE tbs2);
```

2. **散列分区**

```
CREATE INDEX sales_idx ON sales(item) LOCAL
   STORE IN (tbs1, tbs2);
```

3. **复合分区：**

```
CREATE INDEX sales_idx ON sales(sale_date, item)
  STORAGE (INITIAL 1M, MAXEXTENTS UNLIMITED)
   LOCAL
  STORE IN (tbs1, tbs2, tbs3, tbs4, tbs5)
  (PARTITION q1_1997, PARTITION q2_1997,
    PARTITION q3_1997
     (SUBPARTITION q3_1997_s1 TABLESPACE ts2,
       SUBPARTITION q3_1997_s2 TABLESPACE ts4,
       SUBPARTITION q3_1997_s3 TABLESPACE ts6,
       SUBPARTITION q3_1997_s4 TABLESPACE ts8),
    PARTITION q4_1997,
    PARTITION q1_1998);
```

§6.4 簇

簇（cluster**）是存储表数据的可选择的方法。一个簇是一组表，将具有同一公共列值的行存储在一起，并且它们经常一起使用。这些公共列构成簇关键字。例如：**EMP **表各** DEPT **表共享** DEPTNO **列，所以** EMP **表和** DEPT **表可聚集在一起，簇关键字的列为** DEPTNO **列，该簇将每个部门的全部职工行各该部门的行物理地存储在同一数据块中。**

使用簇的好处在于：

1. **在存取簇中的数据时减少了磁盘输入和输出的时间，使数据库性能增加；**

2. **簇关键字的值在簇和簇索引中只存放一次，节省了存储簇中**

1

相关表和索引数据要求的存储空间。

如果用户拥有 CREATE CLUSTER 的系统权限和欲包含该簇的表空间的配额，或者具有 UNLIMITED TABLESPACE的系统权限，那么用户可以在自己的模式中创建簇。如果用户具有 CREATE ANY CLUSTER 的系统权限和欲包含该簇的表空间的配额，或者具有 UNLIMITED TABLESPACE的系统权限，那么可以在其他用户的模式下创建簇。

下面是一个建立簇的例子：

```
CREATE CLUSTER emp_dept(deptno NUMBER(3))
  PCTUSED 80
  PCTFREE 5
  SIZE 600
  TABLESPACE users
 STORAGE (INITIAL 200K
    NEXT 300K
    MINEXTENTS 2
    MAXEXTENTS 20
    PCTINCREASE 33);
```

创建了簇后，就可以在创建表指定使用已经创建的簇。在创建簇表时，必须指定要使用的簇名，同时在表中必须有一列和簇关键字同名的数据库列。如上例，创建了簇 emp_dept 后，可以按如下语句创建簇表 emp 和 dept：

```
CREATE TABLE dept (
  Deptno NUMBER(3) PRIMARY KEY,
  ... )
  CLUSTER emp_dept(deptno);
CREATE TABLE emp (
 Empno NUMBER(5) PRIMARY KEY,
```

```
 Ename VARCHAR2(15) NOT NULL,
 ...
 Deptno NUMBER(3) REFERENCES dept)
  CLUSTER emp_dept(deptno);
```

如果用户具有 CREATE INDEX **或** CREATE ANY INDEX **的系统权限，同时具有创建表空间的配额，或者** UNLIMITED TABLESPACE **的系统权限，那么用户可以在簇关键字上创建索引：**

```
CREATE INDEX emp_dept_index
  ON CLUSTER emp_dept
  INITRANS 2
  MAXTRANS 5
  TABLESPACE users
  STORAGE (
     INITIAL 50K
     NEXT 50K
     MINEXTENTS 2
     MAXEXTENTS 10
     PCTINCREASE 33
     PCTFREE 5);
```

如果将散列函数应用于簇关键字上，这时就成为散列簇。在以下情况下，用户可以考虑建立散列簇来增加系统性能：

- **大多数查询都集中在簇关键字上；**
- **散列簇中的表的大小基本都保持不变,所以用户可以确定簇中的表的大小。**

下面是创建散列簇的一个例子：

```
CREATE CLUSTER emp_cluster(deptno NUMBER)
```

SIZE 620

HASH IS deptno HASHKEYS 1009;

上述命令将创建簇关键字为 deptno的一个散列簇，SIZE为 620 指容纳所有行的簇关键字所需的空间为 620 字节，HASHKEYS：1009指在散列簇中可分配的行为 1009。

§6.5 视图

一个视图（view)是由一个或多个表（或其他视图）中的数据的一种定制的表示，是用一个查询定义，所以可认为是一个存储的查询（stored query）或是一个虚表(virtual table)。视图可在使用表的许多地方使用。

由于视图是由表导出的，视图和表存在许多类似，视图象表最多可定义 254 列。视图可以被查询，而在修改、插入或删除时具有一定的限制，在视图上执行的全部操作真正地影响视图的基本表中的数据，受到基本表的完整性约束和触发器的限制。

视图与表不同，一个视图不分配任何存储空间，视图不真正地包含数据。由查询定义的视图相应于视图引用表中的数据。视图只在数据字典中存储其定义。

引入视图有下列好处：

- 通过限制对表的行预定义集合的存取，为表提供附加的安全性；
- 隐藏数据复杂性；
- 为用户简化命令；
- 为基本表的数据提供另一种观点；
- 可将应用隔离基本表定义的修改；
- 用于不用视图无法表示的查询；
- 可用于保存复杂查询。

创建视图时经常采用的命令格式如下：

CREATE [OR REPLACE] [FORCE] VIEW view_name AS
Subquery [with option] ;

其中我们经常使用的一些 WITH **选项：**

- WITH READ ONLY

保证用户不能通过视图来对表进行插入、更新和删除；

- WITH CHECK OPTION

通过视图只能更改视图中定义的行；

- CONSTRAINT constraint

对表中的数据进行约束。

§6.6 序列

在 Oracle**数据库中，可以使用序列来获得一个唯一的整数。在多用户环境下该序列特别有用，可生成各返回序列号而不需要磁盘** I / O **或事务封锁。最常见的用法是使用序列来生成一张表中的关键字段（如一个标识）的值。**

序列号为 ORACLE **整数，最多可有** 38 **个数字。一个序列定义指出一般信息：序列的名字、上升或下降、序列号之间间距和其它信息。对所有序列的确的定义以行存储在** SYSTEM **表空间中的数据字典表中，所以所有序列定义总是可用。由引用序列号的** SQL **语句使用序列号，可生成一个新的序列号或使用当前序列号。一旦在用户会话中的** SQL **语句生成一序列号，该序列号仅为该会话可用。序列号生成是独立于表，所以同一序列生成器可用于一个和多个表。所生成序列号可用于生成唯一的主码。**

使用如下命令来生成一个序列：

```
CREATE SEQUENCE sequence_name
  [INCREMENT BY h]
   [STA RT WITH i]
  [NOMAXVALUE / MAXVALUE max_value]
  [NOMINVALUE / MINVALUE min_value]
   [CYCLE / NOCYLE]
  [NOCACHE / CACHE cash_value]
```

其中：

- INCREMENT BY

序列每次的增量，缺省为１；

- START WITH

序列刚创建后第一个可以使用的值，缺省为１；

- NOMAXVALUE/MAXVALUE

序列可以取的最大值；

- NOMINVALUE/MINVALUE

序列可以取得最小值，需小于或等于 START WITH **和** MAXVALUE**；**

- CYCLE/NOCYCLE

当序列达到最大值后，序列是否循环；

- CACHE/NOCACHE

为了快速获得序列值，在内存中预先分配序列值的数量。

§6.7 同义词

一个同义词(synonym)**为任何表、视图、快照、序列、过程、函数或包的别名，其定义存储在数据字典中。同义词因安全性和方便原因而经常使用，可用于：**

- **蔽对象的名字及其持有者**;
- **式数据库的远程对象提供位置透明性**;
- **用户简化** SQL **语句。**

有两种同义词：公用和专用。一个公用同义词为命名为 PUBLIC **特殊用户组所持有，可为数据库中每一个用户所存取。一个专用同义词是包含在指定用户的模式中，仅为该用户和授权的用户所使用。**

如果用户具有 CREATE SYNONYM **或者** CREATE ANY SYNONYM **权限，那么可以在自己的模式或者别人的模式下创建同义词：**

CREATE SYNONYM synonym_name FOR schema.table;

Chapter7　备份与恢复

- 了解备份与恢复的概念
- 了解备份和恢复的基本策略
- 掌握无归档日志的恢复
- 了解有归档日志的恢复

§7.1　概念及基本策略

§7.1.1 概念

什么是备份？备份就是为了防止数据丢失或者应用程序错误，把数据库中的文件拷贝至另一位置保存。这样，在发生数据丢失或者程序错误后，可以使用这些拷贝来重新构造丢失的数据和程序。

用户可以使用 Oracle Recovery Manager(简称 RMAN)或者是操作系统命令来进行备份。根据备份时数据库的状态，备份可分为不一致备份（inconsistent backup）和一致备份(consistent backup)。不一致备份是数据库打开或数据库非正常关闭后进行的一个或多个数据库文件的备份；一致备份是数据库完全关闭后所进行的一个或多个数据库文件的备份。

备份分为物理备份和逻辑备份：物理备份指对数据库物理文件进行备份，而逻辑备份指对数据库某一部分的逻辑结构进行备份。

物理备份主要是指数据文件、表空间或者是整个数据库的备份。在备份后，如果用户要使用这些备份进行介质恢复，那么在恢复期间，如果备份是一致备份，那么用户可以直接重建备份，不需要进行恢复；如果备份不是一致备份，用户可以把重复记录和增量备份应用到一个重建的备份中，使数据库前进。本章将主要介绍和物理备份相关的内容和方法。

根据备份的对象不同，常见的物理备份类型有整个数据库的备份，表空间的备份，数据文件的备份和控制文件的备份。

通常，恢复就是使用备份的文件，包括数据文件、控制文件等，根据实际需要，利用重做日志，使数据库恢复到某一时间点状态的过程。

一般的介质恢复包含两个部分：重建物理备份和利用重做日志记录或者增量备份更新。无论是哪种，最后都要保证所有的数据文件有相同的 SCN，并且在任何情况下，介质恢复总是使用重建的备份来实施恢复。

Oracle在一个数据库的实例失败后，自动对数据库进行两种备份：崩溃备份和实例备份。其中，实例备份中包含以下两个自动进程：

- 通过联机重做日志，使数据库向前滚动至最接近失败的时间点；
- 回滚所有尚未提交事务。

使用重做日志记录和增量备份，重建一个数据库、表空间或数据文件的组合来备份更新为最当前点，称为完全备份。

通过重建一个备份，并应用特定的增量备份或是重做记录，使数据库向前滚动至某一非当前的时间点或检查点，这种恢复称为不完全恢复（incomplete recovery）。在不完全恢复后，用户必须使用RESETLOGS选项打开数据库以切换联机重做日志。

§7.1.2 备份与恢复基本策略

从数据库文件 – 数据文件、控制文件或联机重做日志文件故障中恢复所需的文件集称为冗余集（redundancy set）。备份与恢复的黄金法则是：包含冗余集的磁盘集或者其他介质应当位于与数据文件、控制文件和联机重做日志所在磁盘不同的磁盘。

在做备份与恢复时，最基本的策略是：

1. 联机重做日志

   在不同磁盘上保留多个联机重做日志的相同拷贝；

2. 归档重做日志

   在多个位置归档重做日志，或者频繁备份归档重做日志；

3. 控制文件

   维护控制文件的多个并行拷贝；

4. 数据文件

8

频繁备份数据文件，并把它们放置在多种介质的安全地方。

在考虑备份策略时，用户需要考虑的主要是采用什么类型的备份、是否需要一致备份和采取什么备份方法：

1．备份类型主要分为数据库备份、表空间备份、数据文件备份和控制文件备份。

整个数据库备份是最常见的类型，包括控制文件及所有数据文件。如果重建备份后，用户应用重做日志，那么该备份是不一致的，反之则是一致的。

表空间备份是指备份一个或多个表空间的数据文件。这种备份类型要求数据库运行于 ARCHIVELOG 模式下,因为在恢复时，这些文件肯定必须用重做日志记录才能使其和其它数据文件保持一致。

同理，数据文件备份也需要运行于 ARCHIVELOG 模式下。

控制文件备份可以在数据库装载后，通过命令 ALTER DATABASE 语句将控制文件备份于指定路径下。

2．是否采取一致备份

一致备份所得到的所有数据文件的头信息是一致的，包含相同的 SCN。因此用户可以重建数据库的一致备份，而不需要进行介质恢复，也不需要进行实例恢复。对数据库进行一致备份的唯一方式是使用 NORMAL 或 IMMEDIATE 选项关闭数据库，在数据库关闭后进行备份。如果数据库运行于 NOARCHIVELOG 模式，那么一致备份是唯一有效的备份选项,因此也需要重复来创建一致性。因为在 NOARCHIVELOG 模式下并不存档重做日志，而是直接改写。

不一致备份是其中所有读写数据文件和控制文件没有对同一个 SCN 点进行点监测的备份。在这种情况下，Oracle 将应用重做日志滚动数据库，直到所有文件的 SCN 全都一致，用户才能打开数据库。不一致备份对于数据库必须一直处于打开状态的系统而言，是一种很有效的备份方法。用户可以每天对数据库进行备份，也可以每天备份一部分数据库数据文件，最后在恢复时应用重做日志来使 SCN 保持一致。

3．备份方法

对于 Oracle8i 数据库而言，用户可以采用的备份方法有以下几种：

- **操作系统级备份**

用户可以使用 UNIX cp **命令来制作备份，或者编写脚本让操作系统进行自动备份；**

- Recovery Manager(RMAN)

恢复管理器(RMAN)**是** Oracle **附带的一个进行备份、映像拷贝和恢复的工具。它使用专用格式备份一个或多个数据文件、控制文件或归档日志，所以在恢复时，用户也必须使用恢复管理器来进行恢复 ；**

- Export

Export **是** Oracle**的一个实用程序，通过输入有关为数据库创建的模式对象的信息来制作备份。输入后，用户可以使用** Import **再将输出文件导入到数据库。**

而对于恢复，用户需要考虑的因素有：采用完全恢复或不完全恢复，采用何种恢复方法。

1. **完全恢复和不完全恢复**

介质恢复指重做记录和增量备份来把数据文件重建为当前或指定的非当前时间。如果恢复到当前时间称为完全恢复，反之称为不完全恢复。

要使数据库或某个表空间、数据文件完全恢复，用户需要重建数据库备份，打开数据库，应用重做记录恢复。

不完全恢复应用在以下情况：

- **联机或存档日志丢失或损坏；**
- **用户错误导致数据丢失；**
- **控制文件丢失。**

恢复完成后，用户需要用 RESETLOGS **方式打开数据库。**

2. **恢复方法**

针对不同的备份方法，用户也可以采用不同的恢复方法：

- **操作系统** + Server Manager

如果用户使用操作系统备份，那么用户可以使用操作系统使用程序备份，然后执行 Server Manager **语句来手工恢复；**

- RMAN

RMAN 可以自动进行备份重建和恢复工作。

§7.2 备份

在掌握了基本的概念和策略后，我们就可以开始具体的讨论如何对数据库进行备份操作了。整个备份的过程如下：

§7.2.1 确定哪些文件需要备份

用户一般需要的文件有数据文件、日志文件和控制文件。用户可以通过数据字典察看这些文件的相关信息：

1. **数据文件**

```
SELECT name FROM v$datafile;

NAME
------------------------------------------------------------
/oracle/dbs/system.dbf
/oracle/dbs/rbs.dbf
/oracle/dbs/user.dbf
```

或者用户连接 v$tablespace **和** v$datafile**，察看表空间及其相关的数据文件：**

```
SELECT t.name tablespace,
       f.name  datafile
FROM  v$tablespace t, v$datafile f
WHERE t.ts# = f.ts#
ORDER BY t.name

Tablespace                     DataFile
------------------------------ ----------------------------------------
SYSTEM                         /oracle/dbs/system.dbf
RBS                            /oracle/dbs/rbs.dbf
USER                           /oracle/dbs/user.dbf
```

2．日志文件

SELECT member FROM v$logfile;

MEMBER

/oracle/dbs/log1.log

/oracle/dbs/log2.log

3．控制文件

SELECT value FROM v$parameter WHERE
name='control_files';

VALUE

/oracle/dbs/control1.dbf, /u01/dbs/control2.dbf

§7.2.2 备份

确定了备份的对象之后，用户可以使用不同的备份方法对数据库进行备份，如 Recovery Manager 和操作系统备份，这里我们简单介绍一下操作系统备份。

前面介绍了备份可以针对整个数据库、控制文件、表空间和数据文件。

针对整个数据库的备份步骤如下：

1．如果数据库是打开的，在 Server Manager 中使用 NORMAL, IMMEDIATE 或 TRANSACTIONAL 关闭数据库：

SVRMGR> shutdown normal;

或

SVRMGR> shutdown immediate;

或

SVRMGR> shutdown transactional;

2．使用操作系统命令复制所有的数据文件、日志文件和控制文件，并尽可能备份初始化参数文件和其他 Oracle 产品初始化文件。如下例把 /oracle/dbs/ 目录下所有的文件复制至备份目录：

% cp /oracle/dbs/* /u01/backup

3．重新启动数据库：

STARTUP

备份文件可以直接通过SQL **语句来产生二进制的备份文件,用户可以使用不同的选项来把备份的控制文件输出到不同的路径下：**

1．备份控制文件至跟踪文件产生目录：

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

2．备份控制文件至指定目录：

ALTER DATABASE BACKUP CONTROLFILE TO
'u01/backup/control.bak 'REUSE;

REUSE **选项允许用户用新控制文件改写当前已存在的控制文件。**

备份表空间及其数据文件有两种方法，一种是联机备份，一种是脱机备份。联机备份常用的步骤是：

1．察看要备份的表空间的数据文件：

SELECT file_name

FROM sys.dba_data_files

WHERE tablespace_name='USERS';

FILE_NAME

/oracle/user.dbf

2．标识联机表空间备份的开始：

ALTER TABLESPACE users BEGIN BACKUP;

3．使用操作系统命令备份该表空间的数据文件：

% cp /oracle/user.dbf /u01/backup

4．标识联机表空间备份结束：

ALTER TABLESPACE users END BACKUP;

脱机备份的步骤是：

1．察看要备份的表空间的数据文件：

```
SELECT file_name
FROM sys.dba_data_files
WHERE tablespace_name='USERS';
 FILE_NAME
 ----------------------------------------------------------------
 /oracle/user.dbf
```

2．使用 normal **选项使表空间脱机：**

```
ALTER TABLESPACE users OFFLINE NORMAL;
```

3．使用操作系统命令备份该表空间的数据文件：

```
% cp /oracle/user.dbf /u01/backup
```

4．使表空间联机：

```
ALTER TABLESPACE users ONLINE;
```

§7.3 恢复

§7.3.1 确定哪些文件需要恢复

V$RECOVER_FILE **中列出了数据库中需要恢复的文件及要恢复它们的原因：**

```
SELECT * FROM V$RECOVER_FILE;
```

FILE#	ONLINE	ERROR	CHANGE	TIME
14	ONLINE		0	
15	ONLINE	FILE NOT FOUND	0	
21	OFFLINE	OFFLINE NORMAL	0	

然后通过V$DATAFILE **和** V$TABLESPACE**确定需要恢复的数据文件和表空间名：**

```
SELECT d.name, t.name tablespace

FROM v$datafile d, v$tablespace t

WHERE t.ts# = d.ts#

AND d.file# IN (14, 15, 21);

NAME                                          TABLESPACE

--------------------------------------------- ----------

/oracle/dbs/tbs1.dbf                          TBS1

/oracle/dbs/tbs2.dbf                          TBS2

/oracle/dbs/tbs3.dbf                          TBS3
```

§7.3.2 重建文件

在恢复时，第一步总是使用备份重建所需的文件，这一步包括重建数据文件、控制文件、日志文件以及其他如参数文件等。具体步骤如下：

1．用上一节的技术确定哪些文件需要恢复；

2．用操作系统命令把文件拷贝至缺省位置，如下例把备份文件拷贝至 /oracle/dbs:

```
% cp /u01/back/* /oracle/dbs
```

§7.3.3 基本恢复过程

通常采用的恢复方式有：

- **使用介质恢复语句**

  使用 SQL**的语句** RECOVER DATABASE**，** RECOVER TABLESPACE, RECOVER DATAFILE **来进行恢复，如：**

  ```
  RECOVER DATABASE;

  RECOVER TABLESPACE tbs1;

  RECOVER DATAFILE '/oracle/user.dbf';
  ```

- **应用归档重做日志**

  应用归档重做日志又有四种方式：

1. 提出归档重做日志名称

步骤为：

① 改变初始化参数，如：

LOG_ARCHIVE_DEST_1 = /oracle/new_arc

② 移动日志文件至新位置，如：

% cp /oracle/arc/* /oracle/new_arc

③ 装载数据库：

STARTUP MOUNT;

④ 恢复数据库：

RECOVER DATABASE;

2. 使用 Server Manager **的** RECOVER **语句自动应用日志文件**

步骤为：

① 用备份重建数据文件：

% cp /u01/backup/tbs* /oracle/dbs

② 装载数据库：

STARTUP MOUNT;

③ 打开自动恢复：

SET AUTORECOVERY ON;

④ 恢复数据库：

RECOVER DATABASE;

⑤ Oracle **自动提示并应用相应的日志文件**

3. 使用 ALTER DATABASE RECOVER **语句单独应用日志**

例如恢复一个表空间的步骤为：

① 标识恢复表空间：

ALTER TABLESPACE RECOVER TABLESPACE users;

然后数据库会提示应用某个日志文件来做恢复。

② 根据提示应用单独的日志文件 ：

ALTER DATABASE RECOVER LOGFILE
'/oracle/arc/log1.arc';

如果不对，数据库会提示应用另一个日志文件来做恢复。

③ 表空间联机：

ALTER TABLESPACE users ONLINE;

4. 使用 ALTER DATABASE RECOVER **语句自动应用日志**

如果备份文件已经重建，用户有管理员权限，那么用户可以使用以下命令自动应用日志：

ALTER DATABASE RECOVER AUTOMATIC
TABLESPACE users;

ALTER TABLESPACE users ONLINE;

- NOARCHIVELOG**模式下恢复**

例如在缺省位置重建整个数据库最近的备份：

1. 关闭数据库

SHUTDOWN ABORT;

2. 修复介质错误；

3. 重建数据库备份；

4. 不完全恢复：

ALTER DATABASE RECOVER DATABASE UNTIL
CANCEL;

5. 开数据库，重置日志序列：

ALTER DATABASE OPEN RESETLOGS;

- ARCHIVE**模式下恢复**

以下是恢复的一个例子：

1．如果数据库是打开的，关闭：

SHUTDOWN ABORT;

2．现场备份

3．重建备份；

4．装载数据库：

STARTUP NOMOUNT;

5．基于取消的恢复：

RECOVER DATABASE UNTIL CANCEL;

或基于时间的恢复：

RECOVER DATABASE UNTIL TIME 2001-12-13 01:32:10 ’USING BACKUP CONTROL;

或基于改变的恢复：

RECOVER DATABASE UNTIL CHANGE 100;