
Oracle9i 数据库管理基础 I

第 1 册 • 学生指南

D11321CN11
产品版 1.1
2002 年 3 月
D34488

ORACLE®

作者

Marie St. Gelais
S Matt Taylor Jr

技术审稿人

Paulo Barqueira
Charles Fabrie
Lilian Hobbs
Dominique Jeunot
Donna Keesling
Simon Law
Howard Ostrow
Ashesh Parekh
Gabriela Stanescu

出版商

John B Dawson

版权所有 © Oracle Corporation, 2001。保留所有权利。

本文档包含 Oracle 公司的专有权信息；根据许可证协议提供该文档，该许可证协议含有对使用和公开本文档的各种限制；本文档还受到版权法的保护。严禁对本文档所涉及的软件进行逆向工程设计。如果将本文档交付给美国国防部下属的某个政府机构，则根据“受限制权利”进行提供并且必须符合以下规定：

受限制权利说明

对商用计算机软件的限制同样适用于政府的使用、复制或泄露行为，并且根据联邦法律，本软件将被视为“受限制权利”软件，相关规定请参见 DFARS 252.227-7013 《技术数据和计算机软件中的权利》（1988 年 10 月）中的段落 (c)(1)(ii)。

未经 Oracle 公司事先明确的书面许可，不得以任何形式或通过任何途径复制本文档或文档的任何部分。任何其它复制行为都被视为对版权法的触犯，违者可能须负民事和（或）刑事责任。

如果将本文档交付给美国国防部之外的某个政府机构，则根据“受限制权利”进行提供，该权利在 FAR 52.227-14 《数据权利-通则》（包括 1987 年 6 月的《附则》III）中有所规定。

本文档中的信息如有更改，恕不另行通知。如果您在此文档中发现任何问题，请书面通知 Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065。Oracle 公司不保证此文档中没有错误。

SQL*Loader、SQL*Net、SQL*Plus、Net8、Oracle Call Interface、Oracle7、Oracle8、Oracle 8i、Developer/2000、Developer/2000 Forms、Designer/2000、Oracle Enterprise Manager、Oracle Parallel Server、PL/SQL、Pro*C、Pro*C/C++ 和 Trusted Oracle 都是 Oracle 公司的商标和注册商标。

所有其它产品或公司名称仅用于标识，可能是其各自所有者的商标。

目录

前言

I 简介

课程目标 1-2

Oracle9i 企业版 1-3

数据库管理员的任务 1-4

1 Oracle 体系结构组件

目标 1-2

基本组件概览 1-3

Oracle 服务器 1-5

Oracle 例程 1-6

建立连接和创建会话 1-7

Oracle 数据库 1-9

物理结构 1-10

内存结构 1-11

系统全局区 1-12

共享池 1-15

库高速缓存 1-16

数据字典高速缓存 1-17

数据库缓冲区高速缓存 1-18

重做日志缓冲区 1-21

大型共享池 1-22

Java 池 1-24

程序全局区 1-25

进程结构 1-28

用户进程 1-29

服务器进程 1-30

后台进程 1-31

数据库写入器 (DBWn) 1-32

日志写入器 (LGWR) 1-33

系统监视器 (SMON) 1-34

进程监视器 (PMON) 1-35

检查点 (CKPT) 1-36

归档程序 (ARCn) 1-37

逻辑结构 1-39

处理 SQL 语句 1-42

小结 1-44

练习 1 概览 1-45

2 Oracle 服务器入门

目标 2-2

数据库管理工具 2-3

Oracle Universal Installer 2-4

启动 Universal Installer 2-5

使用响应文件进行非交互式安装 2-6

Oracle Database Configuration Assistant 2-9

数据库管理员用户 2-10

SQL*Plus 2-11

Oracle Enterprise Manager 2-12

Oracle Enterprise Manager 体系结构 2-13

控制台 2-15

小结 2-17

练习 2 概览 2-18

3 管理 Oracle 例程

目标 3-2

初始化参数文件 3-3

PFILE initSID.ora 3-6

创建 PFILE 3-7

PFILE 示例 3-8

SPFILE spfileSID.ora 3-9

创建 SPFILE 3-10

SPFILE 示例 3-13

STARTUP 命令行为 3-14

修改 SPFILE 中的参数 3-15

启动数据库 NOMOUNT 3-19

启动数据库 MOUNT 3-20

启动数据库 OPEN 3-21

STARTUP 命令 3-22

ALTER DATABASE 命令 3-25

以受限模式打开数据库 3-26

以只读模式打开数据库 3-29

关闭数据库 3-31

关闭选项 3-32

使用诊断文件监视例程 3-36

警报日志文件 3-37

后台跟踪文件 3-39

用户跟踪文件 3-40

启用或禁用用户跟踪 3-41

小结 3-43

练习 3 概览 3-44

4 创建数据库

目标 4-2

管理和组织数据库 4-3

最佳灵活体系结构 (OFA) 4-4

Oracle 软件和文件位置 4-5

创建的前提条件 4-6

数据库管理员的验证方法 4-7

使用口令文件验证 4-8

创建数据库 4-10

操作系统环境 4-11

Database Configuration Assistant 4-12

使用 Database Configuration Assistant 创建数据库 4-13

手动创建数据库 4-17

创建数据库 4-20

使用 Oracle 管理文件 (OMF) 创建数据库 4-23

故障排除 4-27

数据库的创建结果 4-28

小结 4-29

练习 4 概览 4-30

5 使用数据字典和动态性能视图

目标 5-2

内置数据库对象 5-3

数据字典 5-4

基表和数据字典视图 5-5

创建数据字典视图 5-6

数据字典内容 5-7

数据字典的使用方式 5-8

数据字典视图类别 5-9

数据字典示例 5-11

动态性能表 5-12

动态性能表示例 5-13

管理脚本命名约定 5-15

小结 5-16

练习 5 概览 5-17

6 维护控制文件

目标 6-2

控制文件 6-3

控制文件的内容 6-5

对控制文件进行多元备份 6-7

使用 SPFILE 时对控制文件进行多元备份 6-8

使用 PFILE 时对控制文件进行多元备份 6-9

使用 OMF 管理控制文件 6-10

获取控制文件信息 6-11

小结 6-14

练习 6 概览 6-15

7 维护重做日志文件

目标 7-2

使用重做日志文件 7-3

重做日志文件的结构 7-4

重做日志文件如何发挥作用 7-6

强制执行日志切换和检查点 7-8

添加联机重做日志文件组 7-9

添加联机重做日志文件成员 7-10

删除联机重做日志文件组 7-12

删除联机重做日志文件成员 7-13

重定位或重命名联机重做日志文件 7-15

联机重做日志文件的配置 7-17

使用 OMF 管理联机重做日志文件 7-19

获取组和成员信息 7-20

归档的重做日志文件 7-22

小结 7-26

练习 7 概览 7-27

8 管理表空间和数据文件

目标 8-2

表空间和数据文件 8-3

表空间类型 8-4

创建表空间 8-5

表空间的空间管理 8-9

本地管理的表空间 8-10

字典管理的表空间 8-12

还原表空间 8-13

临时表空间 8-14

- 缺省临时表空间 8-17
- 创建缺省临时表空间 8-18
- 缺省临时表空间的限制 8-21
- 只读表空间 8-22
- 使表空间脱机 8-25
- 更改存储设置 8-28
- 调整表空间大小 8-30
- 启用数据文件自动扩展 8-31
- 手动调整数据文件的大小 8-34
- 向表空间添加数据文件 8-35
- 移动数据文件的方法 8-37
- 删除表空间 8-40
- 使用 OMF 管理表空间 8-43
- 使用 OMF 管理表空间 8-44
- 获取表空间信息 8-45
- 小结 8-46
- 练习 8 概览 8-47

9 存储结构和关系

- 目标 9-2
- 存储和关系结构 9-3
- 段类型 9-4
- 存储子句优先级 9-8
- 区的分配与回收 9-9
- 已用区和空闲区 9-10
- 数据库块 9-11
- 多种块大小支持 9-12
- 标准块大小 9-13
- 非标准块大小 9-14
- 创建非标准块大小的表空间 9-16
- 多种块大小的规则 9-18
- 数据库块内容 9-19
- 块空间使用参数 9-20
- 数据块管理 9-22
- 自动段空间管理 9-23
- 配置自动段空间管理 9-25
- 手动数据块管理 9-26
- 块空间使用率 9-27
- 获取存储信息 9-29
- 小结 9-32
- 练习 9 概览 9-33

10 管理还原数据

- 目标 10-2
- 管理还原数据 10-3
- 还原段 10-4
- 还原段：用途 10-5
- 读一致性 10-6
- 还原段的类型 10-7
- 自动还原管理：概念 10-9
- 自动还原管理：配置 10-10
- 自动还原管理：初始化参数 10-11
- 自动还原管理：UNDO 表空间 10-12
- 自动还原管理：改变 UNDO 表空间 10-14
- 自动还原管理：切换 UNDO 表空间 10-16
- 自动还原管理：删除 UNDO 表空间 10-18
- 自动还原管理：其它参数 10-21
- 还原数据统计信息 10-23
- 自动还原管理：调整 UNDO 表空间的大小 10-24
- 自动还原管理：还原限额 10-26
- 获取还原段信息 10-27
- 小结 10-29
- 练习 10 概览 10-30

11 管理表

- 目标 11-2
- 存储用户数据 11-3
- Oracle 内置数据类型 11-6
- ROWID 格式 11-10
- 行的结构 11-12
- 创建表 11-13
- 创建表：原则 11-17
- 创建临时表 11-18
- 设置 PCTFREE 和 PCTUSED 11-19
- 行移植和行链接 11-20
- 更改存储和块使用参数 11-21
- 手动分配区 11-24
- 重新组织非分区表 11-25
- 截断表 11-26
- 删除表 11-27
- 删除列 11-29
- 使用 UNUSED 选项 11-31

获取表信息 11-33
小结 11-35
练习 11 概览 11-36

12 管理索引

目标 12-2
索引分类 12-3
B 树索引 12-5
位图索引 12-7
比较 B 树索引和位图索引 12-9
创建正常的 B 树索引 12-10
创建索引：原则 12-13
创建位图索引 12-15
更改索引的存储参数 12-18
分配和回收索引空间 12-20
重建索引 12-21
联机重建索引 12-23
合并索引 12-24
检查索引及其有效性 12-25
删除索引 12-27
标识未用索引 12-29
获取索引信息 12-30
小结 12-31
练习 12 概览 12-32

13 维护数据完整性

目标 13-2
数据完整性 13-3
约束的类型 13-5
约束的状态 13-6
约束检查 13-8
将约束定义为立即或延迟 13-9
执行主键和唯一键约束 13-10
外键注意事项 13-11
创建表时定义约束 13-13
约束定义原则 13-17
启用约束 13-18
使用 EXCEPTIONS 表 13-23
获取约束信息 13-26
小结 13-29
练习 13 概览 13-30

14 管理口令安全性和资源

- 目标 14-2
- 配置文件 14-3
- 口令管理 14-5
- 启用口令管理 14-6
- 口令帐户锁定 14-7
- 口令失效和过期 14-8
- 口令历史记录 14-9
- 口令校验 14-10
- 用户提供的口令函数 14-11
- 口令校验函数 `VERIFY_FUNCTION` 14-12
- 创建配置文件：口令设置 14-13
- 改变配置文件：口令设置 14-17
- 删除配置文件：口令设置 14-19
- 资源管理 14-21
- 启用资源限制 14-22
- 在会话级设置资源限制 14-23
- 在调用级设置资源限制 14-24
- 创建配置文件：资源限制 14-25
- 使用“数据库资源管理器”管理资源 14-28
- 资源计划指令 14-31
- 获取口令和资源限制信息 14-33
- 小结 14-35
- 练习 14 概览 14-36

15 管理用户

- 目标 15-2
- 用户和安全性 15-3
- 数据库方案 15-5
- 创建用户操作的核对清单 15-6
- 创建新用户：数据库验证 15-7
- 创建新用户：操作系统验证 15-10
- 更改用户的表空间限额 15-12
- 删除用户 15-14
- 获取用户信息 15-16
- 小结 15-17
- 练习 15 概览 15-18

16 管理权限

- 目标 16-2
- 管理权限 16-3
- 系统权限 16-4
- 系统权限：示例 16-5
- 授予系统权限 16-6
- SYSDBA 和 SYSOPER 权限 16-8**
- 系统权限限制 16-9
- 撤消系统权限 16-10
- 撤消通过 **ADMIN OPTION** 授予的系统权限 16-12
- 对象权限 16-13
- 授予对象权限 16-14
- 撤消对象权限 16-17
- 撤消对象权限 **WITH GRANT OPTION** 16-20
- 获取权限信息 16-21
- 小结 16-22
- 练习 16 概览 16-23

17 管理角色

- 目标 17-2
- 角色 17-3
- 角色的优点 17-4
- 创建角色 17-5
- 预定义角色 17-7
- 修改角色 17-8
- 分配角色 17-10
- 设置缺省角色 17-13
- 应用程序角色 17-15
- 启用和禁用角色 17-16
- 撤消用户角色 17-19
- 删除角色 17-21
- 角色创建原则 17-23
- 使用口令与缺省角色的原则 17-24
- 获取角色信息 17-25
- 小结 17-26
- 练习 17 概览 17-27

18 审计

- 目标 18-2
- 审计 18-3
- 审计原则 18-4

- 审计类别 18-6
- 数据库审计 18-8
- 审计选项 18-10
- 获取审计信息 18-12
- 获取审计记录信息 18-13
- 小结 18-14
- 练习 18 概览 18-15

19 将数据加载到数据库中

- 目标 19-2
- 数据加载方法 19-3
 - 直接加载 19-4
 - 串行直接加载 19-6
 - 并行直接加载 19-7
 - SQL*Loader 19-9
 - 使用 SQL*Loader 19-11
 - SQL*Loader 控制文件 19-13
 - 与控制文件的语法有关的注意事项 19-17
 - 输入数据和数据文件 19-18
 - 逻辑记录 19-22
 - 加载方法 19-23
 - 直接路径加载和常规路径加载的比较 19-26
 - 并行直接路径加载 19-28
 - 数据转换 19-29
 - 被废弃或拒绝的记录 19-30
 - 日志文件的内容 19-34
 - SQL*Loader 原则 19-36
 - 小结 19-37
 - 练习 19 概览 19-38

20 使用“全球化支持”

- 目标 20-2
- “全球化支持”功能 20-3
- 编码方案 20-5
- 数据库字符集和国家字符集 20-8
- 选择 Oracle 数据库字符集的原则 20-9
- 选择 Oracle 国家字符集的原则 20-11
- 选择 Unicode 解决方案: Unicode 数据库 20-12
- 选择 Unicode 解决方案: Unicode 数据类型 20-13

指定语言相关行为 20-14
指定服务器的语言相关行为 20-15
相关语言和地域缺省值 20-16
指定会话的语言相关行为 20-18
客户机-服务器体系结构中的字符集 20-19
指定会话的语言相关行为 20-21
文字排序 20-22
NLS 排序 20-23
在 SQL 函数中使用 NLS 参数 20-26
文字索引支持 20-30
使用 NLS 导入和加载数据 20-31
获取字符集信息 20-32
获取 NLS 设置信息 20-33
在 SQL 函数中使用 NLS 参数 20-37
小结 20-38
练习 20 概览 20-39

- A 如何在 Unix 环境中创建 Oracle9i 数据库**
- B 手动管理还原数据（回退段）**
- C SQL*Plus 练习解答**
- D Oracle Enterprise Manager 练习解答**

前言

概要

本课程旨在使 Oracle 数据库管理员 (DBA) 在基本管理任务方面奠定牢固的基础。本课程的主要目的是为 DBA 提供建立和维护 Oracle 数据库及解决数据库问题的必要知识和技能。本课程专为初级数据库管理员、技术支持分析人员、系统管理员、应用程序开发人员、MIS 主管和其他 Oracle 的用户而设计。

本前言包括以下部分：

- 课前须知
- 前提条件
- 本课程的组织结构
- 相关出版物
- 印刷惯例

课前须知

要想从本课程中获得最大的收益，必须具有以下特定的技能：

- 熟悉关系数据库的概念
- 精通 SQL、SQL*Plus 和操作系统命令（Unix 和 NT）
- 基本的操作系统知识
- 在 Oracle 环境中工作过的一些经验

前提条件

- Oracle 简介

本课程的组织结构

Oracle 9i 数据库管理基础 I 由教师讲解，同时还包括实际操作练习。本课程允许利用 SQL*Plus 或 Oracle Enterprise Manager (OEM) 完成各项练习。此外，本课程还包含为准备 Oracle 专家认证考试而设计的明确目标。

相关出版物

Oracle 出版物

书名	编号
Oracle9i Backup and Recovery Concepts	A90133-02
Oracle9i Database Administrator's Guide	A90117-01
Oracle9i Database Concepts	A88856-02
Oracle9i Database Error Messages	A90202-02
Oracle9i Database New Features	A90120-02
Oracle9i Database Reference	A90190-02
Oracle9i Database Utilities	A90192-01
Oracle9i Enterprise Manager Administrator's Guide	A88767-02
Oracle9i Enterprise Manager Concepts Guide	A88770-01
Oracle9i Enterprise Manager Configuration Guide	A88769-01
Oracle9i Net Services Administrator's Guide	A90154-01
Oracle9i Net Services Reference Guide	A90155-01
Oracle9i Recovery Manager Reference	A90136-02
Oracle9i Recovery Manager User's Guide	A90135-01
Oracle9i Database Reference	A90190-02
Oracle9i SQL Reference	A90125-01
Oracle9i User-Managed Backup and Recovery Guide	A90134-01

其它出版物

- 系统发行公告牌
- 安装指南和用户指南
- *read.me* 文件
- International Oracle User's Group (IOUG) 文章
- *Oracle Magazine*

印刷惯例

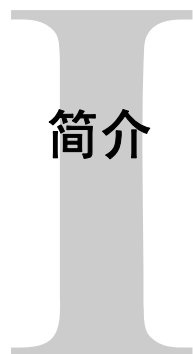
文本印刷惯例

惯例	元素	示例
粗斜体	词汇表术语（如果存在词汇表）	这种 <i>算法</i> 插入了新的键值。
大小写	按钮、复选框、触发器、窗口	单击“可执行文件”按钮。 选中“无法删除卡”复选框。 将 When-Validate-Item 触发器分配给 ORD 块。 打开“主调度”窗口。
Courier new 字体、区分大小写（缺省为小写）	代码输出、目录名、文件名、口令、路径名、URL、用户输入、用户名	代码输出: <code>ebug.set ('I',300);</code> 目录: <code>bin (DOS)</code> 、 <code>\$FMHOME (UNIX)</code> 文件名: 找到 <code>init.ora</code> 文件。 口令: <code>User tiger</code> 作为您的口令。 路径名: 打开 <code>c:\my_docs\projects</code> URL: 转到 <code>http://www.oracle.com</code> 用户输入: 输入 <code>300</code> 用户名: 以 <code>scott</code> 的身份登录
首字母大写	图形标签（除非术语是专有名词）	客户地址（ <i>但</i> Oracle Payables <i>除外</i> ）
斜体	强调的字词、书名和课程名、变量	不要保存对数据库所做的更改。 有关详细信息，请参阅 <i>Oracle7 Server SQL Language Reference Manual</i> 。 输入 <code>user_id@us.oracle.com</code> ，其中 <i>user_id</i> 是用户名。
引号	名称长且只有首字母大写的界面元素； 交叉引用的课程标题和章节标题	选择“包括一个可以重新使用的模块组件”，然后单击“完成”。 本主题在第 3 课“使用对象”的第 II 单元中加以讨论。
大写	SQL 列名、命令、函数、方案、表名	使用 <code>SELECT</code> 命令看存储在 EMP 表的 <code>LAST_NAME</code> 列中的信息。

惯例	元素	示例
箭头	菜单路径	选择“文件”(File) > “保存”(Save)。
逗号	按键顺序	依次按下并松开这些键： [Alternate]、[F]、[D]
加号	按键组合	同时按下并按住这些键： [Ctrl]+[Alt]+[Del]

代码印刷惯例

惯例	元素	示例
大小写	Oracle Forms 触发器	When-Validate-Item
小写	列名、表名	SELECT last_name FROM s_emp;
	口令	DROP USER scott IDENTIFIED BY tiger;
	PL/SQL 对象	OG_ACTIVATE_LAYER (OG_GET_LAYER ('prod_pie_layer'))
小写斜体	语法变量	CREATE ROLE <i>role</i>
大写	SQL 命令和函数	SELECT userid FROM emp;



ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

课程目标

完成这一课的学习后，您应该能达到下列目标：

- 了解 Oracle 体系结构中的各种组件
- 启动和关闭 Oracle 数据库
- 创建可操作的数据库
- 管理 Oracle 控制文件、重做日志文件、数据文件、表空间、段、区和块
- 管理用户、权限和资源
- 使用“全球化支持”功能

ORACLE

I-2

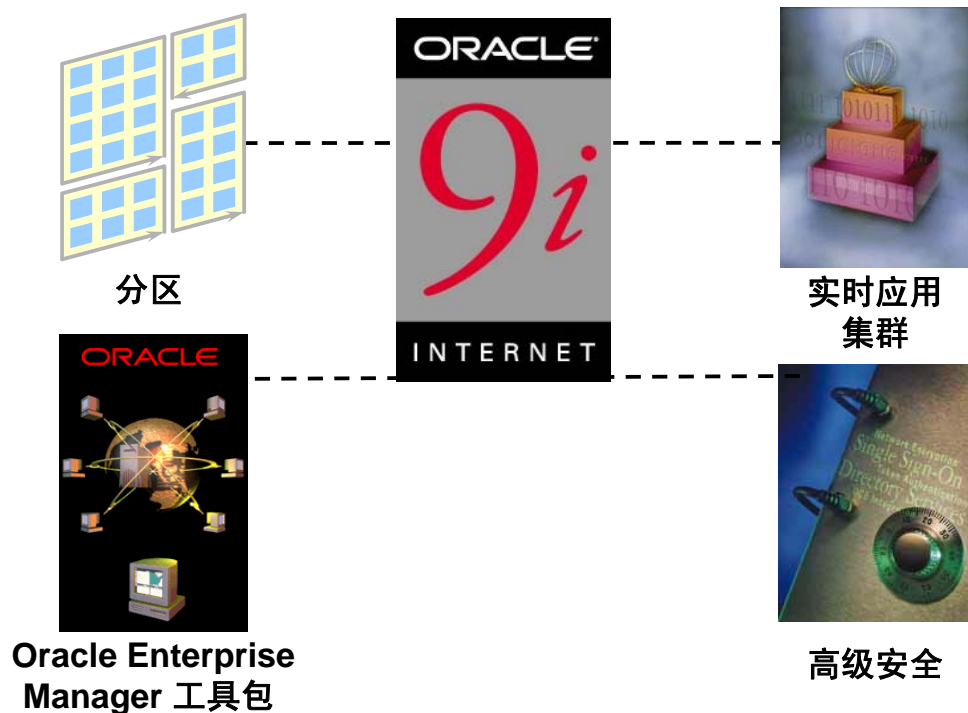
Copyright © Oracle Corporation, 2001. All rights reserved.

本课程涉及的内容

本课程是介绍核心数据库管理员任务的系列课程中的入门课程。本课程涉及以下任务：

- 概述 Oracle 体系结构
- 计划和创建数据库
- 管理内存结构、进程结构、物理结构和逻辑结构
- 通过监控数据库用户的操作来对其进行管理
- 使用“全球化支持”功能

Oracle9i 企业版



什么是 Oracle9i 企业版？

Oracle9i 企业版是一种可伸缩且易于管理的对象关系数据库。本课程就最基本的企业版的管理进行了探讨。但是，下列选件还提供了其它功能。

- 分区 (Partitioning)：提供了一些可用于执行大型、可伸缩的应用程序的实用工具。通过它，您可以在比基本企业版中更低的粒度级别上控制表和索引
- 实时应用集群 (Real Application Clusters)：改善数据库的可伸缩性和可用性，这是通过允许 Oracle 软件的多个副本访问单个数据库来实现的
- Oracle Enterprise Manager 工具包：构建在 Oracle Enterprise Manager 的基础上。Oracle Enterprise Manager 的诊断包 (Diagnostics Pack)、优化包 (Tuning Pack) 和更改管理包 (Change Management Pack) 均是附件，可以为 DBA 提供一整套工具，用来对 Oracle 环境进行高级诊断、监视、优化以及管理更改。
- 高级安全 (Advanced Security)：使用加密和数据完整性检查来提供客户机-服务器和服务器-服务器的网络安全，同时可使用第三方安全服务来支持增强的用户验证服务。

注：以上选件均需购买许可证。

数据库管理员的任务

- 计划和创建数据库
- 管理数据库可用性
- 管理物理结构和逻辑结构
- 根据设计管理存储
- 管理安全
- 网络管理
- 备份和恢复
- 数据库优化



数据库管理员的任务

数据库管理员负责维护 Oracle 服务器，使该服务器可以处理用户的请求。为了进行有效维护，了解 Oracle 体系结构是必不可少的。本课程着重概述了 Oracle 体系结构，另外还涉及了一些管理员任务，如计划和创建数据库、管理数据库可用性、管理内存/物理/逻辑等结构，以及管理用户和权限。

其它课程中涉及的数据库管理员任务：

下面的任务是在其它课程中讨论的：

- 备份和恢复，在 *Oracle9i 数据库管理基础II* 中讨论
- 网络管理，在 *Oracle9i 数据库管理基础II* 中讨论
- 数据库优化，在 *Oracle9i 数据库性能优化* 中讨论

1

Oracle 体系结构组件

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

目标

完成这一课的学习后，您应该能达到下列目标：

- 概括 **Oracle** 体系结构及其主要组件
- 列出在用户连接到 **Oracle** 例程过程中涉及的结构

ORACLE

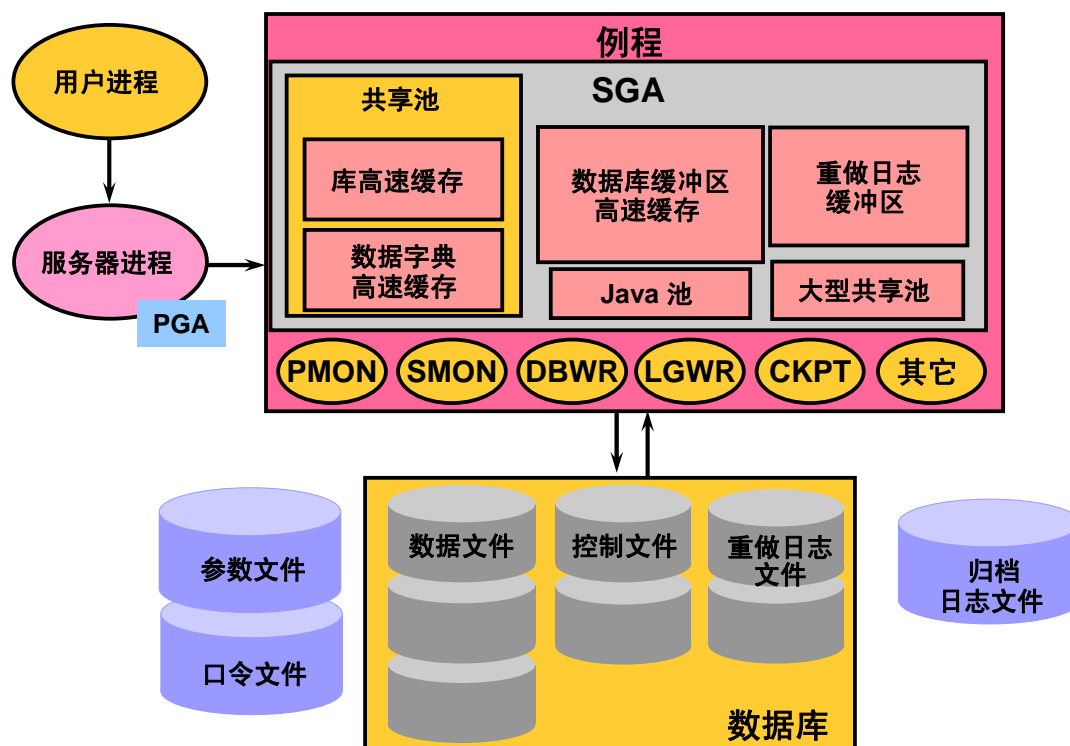
1-2

Copyright © Oracle Corporation, 2001. All rights reserved.

目标

本课通过分析建立数据库连接、创建会话和执行 SQL 命令时涉及的物理结构、内存结构、进程结构和逻辑结构，向您介绍了 Oracle 服务器的体系结构。

基本组件概览



1-3

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

基本组件概览

Oracle 体系结构包括很多基本组件，本课将详细介绍这些组件。

- **Oracle 服务器：**Oracle 服务器中包含多种文件结构、进程结构和内存结构；但是，处理 SQL 语句时，并非所有这些结构都会用到。某些结构用于提高数据库的性能，确保该数据库在遇到软件或硬件错误时可以恢复，或者执行维护该数据库所需的其它任务。Oracle 服务器包括一个 Oracle 例程和一个 Oracle 数据库。
- **Oracle 例程：**Oracle 例程是后台进程和内存结构的组合。只有启动例程后，才能访问数据库中的数据。每次启动例程时，会分配系统全局区 (SGA) 并启动 Oracle 后台进程。后台进程代表调用进程执行各种功能。它们把为每个用户运行的多个 Oracle 程序所处理的功能统一起来。后台进程执行输入/输出 (I/O)，并监视其它 Oracle 进程来提高并行性，从而使性能和可靠性更加优越。
- **Oracle 数据库：**Oracle 数据库包含操作系统文件（也称为数据库文件），这些文件为数据库信息提供了实际的物理存储。数据库文件用于确保数据一致性并能在例程失败时得以恢复。

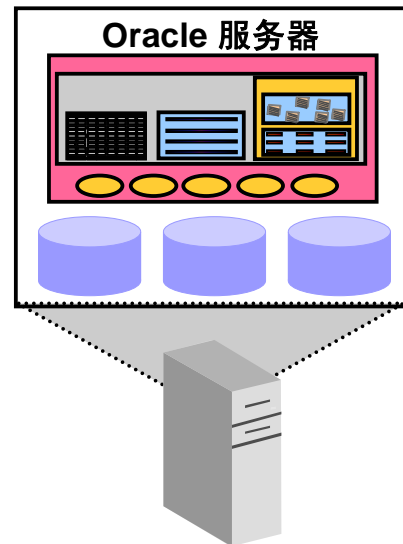
基本组件概览（续）

- 其它关键文件：非数据库文件用于配置例程、验证特权用户以及在磁盘出现故障时恢复数据库。
- 用户进程和服务器进程：执行 SQL 语句时，用户进程和服务器进程是其中涉及的主要进程；但是，其它进程也会有助于服务器完成 SQL 语句的处理。
- 其它进程：还有很多供其它选件使用的其它进程，例如，高级排队 (Advanced Queuing)、实时应用集群 (Real Application Clusters)、共享服务器 (Shared Server) 和高级复制 (Advanced Replication) 等。这些进程将在相应的课程中分别进行讨论。

Oracle 服务器

Oracle 服务器：

- 是一个数据库管理系统，它为信息管理提供了开放、综合和集成的方法
- 包括 Oracle 例程和 Oracle 数据库



ORACLE

1-5

Copyright © Oracle Corporation, 2001. All rights reserved.

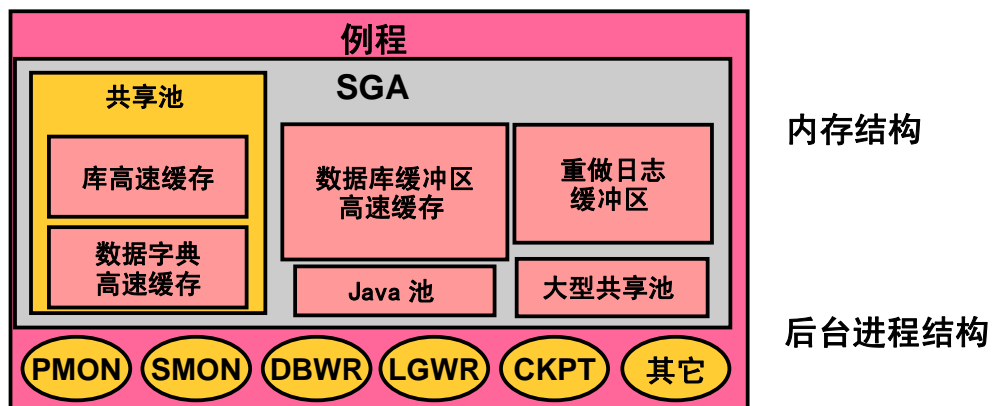
Oracle 服务器

数据库服务器是信息管理的关键。一般来说，服务器必须可靠地管理多用户环境中的大量数据，以便多个用户能够同时访问同一数据。所有这一切都必须在保证高性能的同时进行。数据库服务器还必须防止未经授权的访问，并为故障恢复提供有效的解决方案。

Oracle 例程

Oracle 例程：

- 是一种访问 Oracle 数据库的方式
- 始终打开一个，并且只打开一个数据库
- 由内存结构和后台进程结构组成



ORACLE

1-6

Copyright © Oracle Corporation, 2001. All rights reserved.

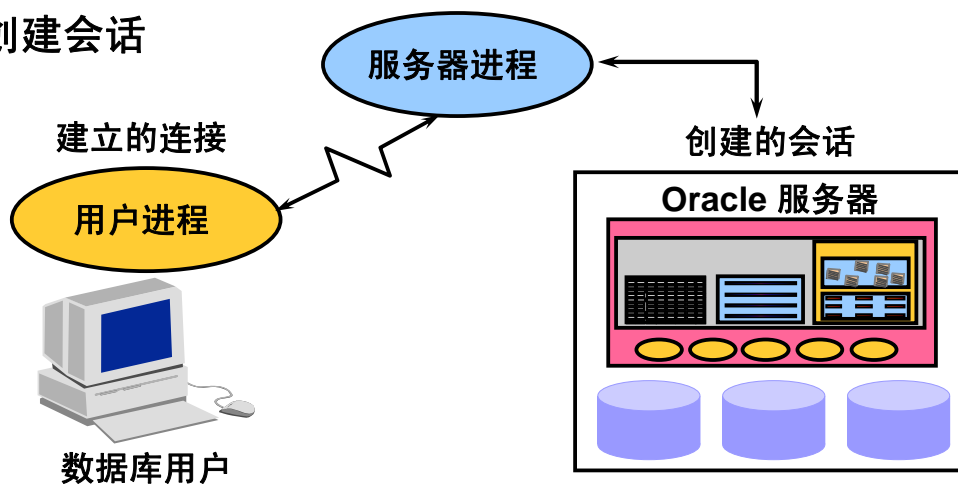
Oracle 例程

Oracle 例程由系统全局区 (SGA) 内存结构和用于管理数据库的后台进程组成。例程是通过使用特定于每个操作系统的方法来标识的。例程一次只能打开和使用一个数据库。

建立连接和创建会话

连接到 Oracle 例程包括：

- 建立用户连接
- 创建会话



ORACLE

1-7

Copyright © Oracle Corporation, 2001. All rights reserved.

建立连接和创建会话

用户只有在连接到例程后，才能向 Oracle 数据库提交 SQL 语句。

- 用户启动 SQL*Plus 之类的工具，或者运行使用 Oracle Forms 之类的工具开发的应用程序。该应用程序或工具将作为用户进程来执行。
- 在最基本的配置中，用户登录到 Oracle 服务器时，运行 Oracle 服务器的计算机上就会创建一个进程。这个进程称为服务器进程。服务器进程代表客户机上运行的用户进程与 Oracle 例程进行通信。服务器进程代表用户执行 SQL 语句。

连接：

连接是用户进程和 Oracle 服务器之间的通信路径。数据库用户可以采用以下三种方式之一连接到 Oracle 服务器：

- 用户登录到运行 Oracle 例程的操作系统上，然后启动访问该系统中的数据库的应用程序或工具。通信路径是使用主机操作系统上的进程间通信机制建立的。

建立连接和创建会话

连接（续）

- 用户在本地计算机上启动应用程序或工具，然后通过网络连接到运行 Oracle 例程的计算机。在这个称为客户机/服务器的配置中，用户和 Oracle 服务器使用网络软件进行通信。
- 在三层连接中，用户计算机通过网络与应用程序服务器或网络服务器进行通信，这些服务器又通过网络与运行 Oracle 例程的计算机连接。例如，用户在网络中的一台计算机上运行浏览器来使用位于 NT 服务器上的应用程序，该 NT 服务器又从 UNIX 主机上运行的 Oracle 数据库中检索数据。

会话

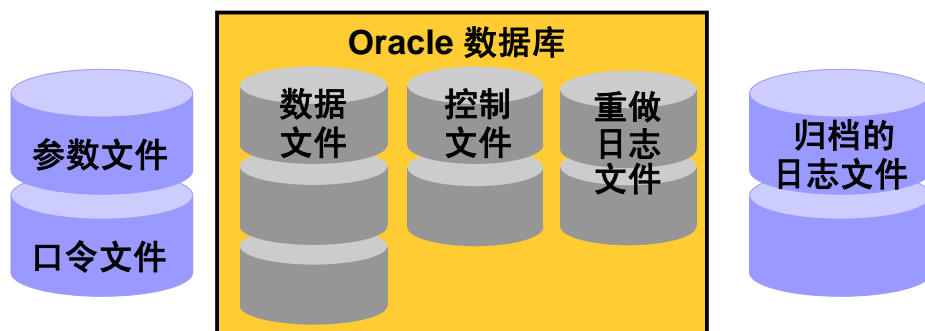
会话是用户与 Oracle 服务器的一种特定连接。会话在 Oracle 服务器验证用户后启动，当用户退出或出现异常终止时会话结束。对某个具体的数据库用户来说，如果该用户从很多工具、应用程序或者终端同时登录，则可能有很多并发会话。除了一些专用的数据库管理工具以外，启动数据库会话还要求 Oracle 服务器可供使用。

注：在此处所讲的连接类型中，用户进程和服务器进程之间存在一对一的通信关系，这称作专用服务器连接。使用共享服务器 (Shared Server) 配置时，多个用户进程可以共享服务器进程。

Oracle 数据库

Oracle 数据库：

- 是一个被统一处理的数据集合
- 包括三类文件



ORACLE

1-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle 数据库

数据库的一般用途是存储和检索相关信息。Oracle 数据库具有逻辑结构和物理结构。数据库的物理结构是数据库中操作系统文件的集合。Oracle 数据库包含以下三类文件：

- 数据文件，包含数据库中的实际数据
- 重做日志文件，包含数据库的更改记录，可以在出现故障时恢复数据
- 控制文件，包含维护和验证数据库完整性所需的信息

其它关键文件结构

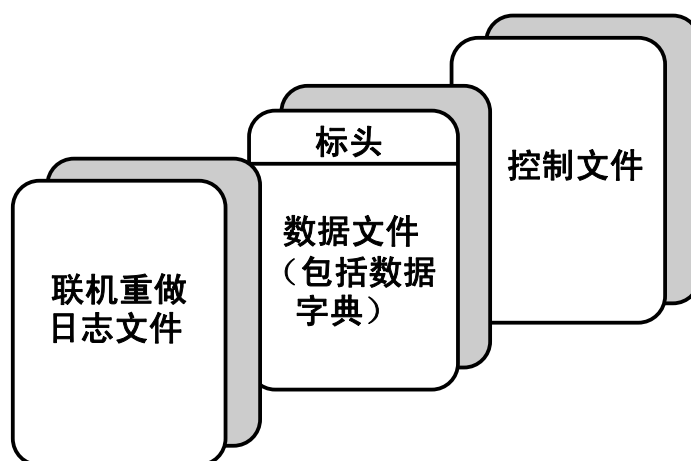
Oracle 服务器还使用一些其它文件，这些文件并不是数据库的一部分：

- 参数文件，用来定义 Oracle 例程的特性。例如，它包含调整 SGA 中一些内存结构大小的参数。
- 口令文件验证哪些用户有权启动和关闭 Oracle 例程。
- 归档的重做日志文件是重做日志文件的脱机副本，当必须从介质失败中进行恢复时可能会需要这些副本。

物理结构

物理结构包括以下三种文件类型：

- 控制文件
- 数据文件
- 重做日志文件



物理结构

Oracle 数据库的物理结构包括以下三种文件类型：控制文件、数据文件和重做日志文件。

内存结构

Oracle 的内存结构由两个内存区组成，分别是：

- **系统全局区 (SGA)**：在例程启动时分配，是 **Oracle** 例程的基本组件
- **程序全局区 (PGA)**：在服务器进程启动时分配

ORACLE

系统全局区

- **SGA 包括以下几种内存结构：**
 - 共享池
 - 数据库缓冲区高速缓存
 - 重做日志缓冲区
 - 其它结构（例如锁定和栓锁管理以及统计数据）
- **在 SGA 中还可配置其它两种内存结构：**
 - 大型共享池
 - Java 池

ORACLE

1-12

Copyright © Oracle Corporation, 2001. All rights reserved.

系统全局区 (SGA)

SGA 也称作共享全局区，用于存储数据库进程共享的数据库信息。它包含有关 Oracle 服务器的数据和控制信息，在 Oracle 服务器所在计算机的虚拟内存中分配。

要查看 SGA 内存的分配情况，可以使用下面的语句：

```
SQL> SHOW SGA;
```

Total System Global Area	36437964	bytes
Fixed Size	6543794	bytes
Variable Size	19521536	bytes
Database Buffers	16777216	bytes
Redo Buffers	73728	bytes

系统全局区（续）

动态 SGA:

从 Oracle9i 开始，动态 SGA 实施了一种体系结构，可以对 SGA 配置进行更改，而不必关闭例程。这样，在不关闭例程的情况下，就可以更改数据库缓冲区高速缓存和共享池的大小。因此，开始时可将数据库缓冲区高速缓存和共享池配置为一个较小的值，然后根据它们各自的工作量增减，但最大值不能超过由 SGA_MAX_SIZE 指定的值。

调整 SGA 的大小:

SGA 的大小由几个初始化参数决定。对 SGA 的大小影响最大的参数有:

DB_CACHE_SIZE: 标准块的高速缓存大小。对于 UNIX，缺省值是 48 MB，对于 NT，缺省值是 52 MB

LOG_BUFFER: 为重做日志缓冲区分配的字节数

SHARED_POOL_SIZE: 专用于共享 SQL 和 PL/SQL 的内存区的大小（以字节为单位）。缺省值是 16 MB。如果是 64 位，则缺省值是 64 MB

LARGE_POOL_SIZE: 大型共享池的大小。缺省值是零（如果未将 init.ora 参数 PARALLEL_AUTOMATIC_TUNING 设置为 TRUE，则自动计算缺省值。）

JAVA_POOL_SIZE: Java 池的大小。缺省值是 24 MB

因此，SGA 的大小不能超过 SGA_MAX_SIZE - DB_CACHE_SIZE - LOG_BUFFER - SHARED_POOL_SIZE - LARGE_POOL_SIZE - JAVA_POOL_SIZE。

注：在 *Oracle9i 数据库性能优化* 课程中详细介绍了动态 SGA 及其大小调整方面的信息。

系统全局区

- **SGA 是动态的**
- **大小由 SGA_MAX_SIZE 参数指定**
- **由 SGA 组件以粒组为单位进行分配和跟踪**
 - 连续的虚拟内存分配
 - 粒组大小由估算的 SGA_MAX_SIZE 总计大小确定

ORACLE

1-14

Copyright © Oracle Corporation, 2001. All rights reserved.

系统全局区（续）

分配单位：

粒组是一个连续虚拟内存分配单位。粒组的大小取决于估算的 SGA 的总大小，这个总大小是根据参数 SGA_MAX_SIZE 的值计算的。

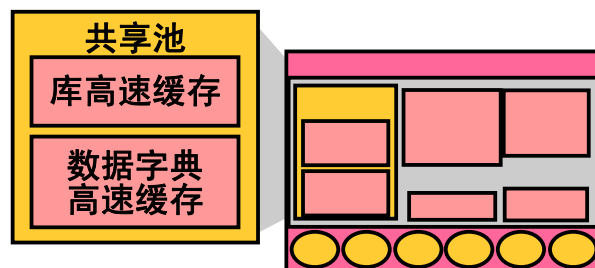
- 如果估算的 SGA 的大小小于 128 MB，那么粒组大小是 4 MB
- 否则为 16 MB

组件（数据库缓冲区高速缓存和共享池）能够以粒组为单位进行增减。对于拥有粒组的每个组件而言，分配给该组件的粒组数、对该组件执行但暂时挂起的任何操作（例如，通过 ALTER SYSTEM 分配粒组、通过 ALTER SYSTEM 释放粒组，相应的自身优化），以及以粒组为目标大小的目标大小都将由 V\$BUFFER_POOL 视图来跟踪和显示。启动例程时，Oracle 服务器将分配粒组条目，每个粒组使用一个条目来支持 SGA_MAX_SIZE 字节的地址空间。继续启动时，每个组件将根据需要获取足够的粒组。SGA 的最低配置是三个粒组（一个粒组用于固定的 SGA（包括重做缓冲区）；一个粒组用于数据库缓冲区高速缓存；另一个粒组用于共享池）。

共享池

- 用于存储：
 - 最近执行的 SQL 语句
 - 最近使用的数据定义
- 它包括以下两个与性能相关的关键内存结构：
 - 库高速缓存
 - 数据字典高速缓存
- 其大小由 SHARED_POOL_SIZE 参数确定

```
ALTER SYSTEM SET  
SHARED_POOL_SIZE = 64M;
```



ORACLE

1-15

Copyright © Oracle Corporation, 2001. All rights reserved.

共享池

共享池环境既包含固定结构，也包含可变结构。固定结构的大小相对保持不变，而可变结构的大小会根据用户和程序的需求增减。固定结构和可变结构的实际大小由一个初始化参数和 Oracle 内部算法来确定。

调整共享池大小：

由于共享池用于可以全局共享的对象，如可重复使用的 SQL 执行计划、PL/SQL 程序包、过程、函数和游标信息，所以必须对它的大小进行调整，以满足固定区和可变区的需要。分配给共享池的内存由 SHARED_POOL_SIZE 初始化参数确定。使用 ALTER SYSTEM SET，便可以动态地重新调整它的大小。经过性能分析后，就可以调整它的大小，但 SGA 的总大小不能超过 SGA_MAX_SIZE。

库高速缓存

- 存储有关最近使用的 **SQL** 和 **PL/SQL** 语句的信息
- 启用常用语句共享
- 由“最近最少使用算法” (LRU) 管理
- 包括以下两个结构：
 - 共享的 **SQL** 区
 - 共享的 **PL/SQL** 区
- 大小由共享池的大小确定

ORACLE

1-16

Copyright © Oracle Corporation, 2001. All rights reserved.

库高速缓存

库高速缓存的大小视所定义的共享池大小而定。内存分配是在对语句进行语法分析或调用程序单元时进行。如果共享池的大小太小，就会将语句连续重新载入库高速缓存，从而使性能受到影响。库高速缓存由算法 LRU 来管理。高速缓存填满时，将从库高速缓存中删除最近很少使用的执行路径和语法分析树，以便为新条目腾出空间。如果某些 **SQL** 或 **PL/SQL** 语句未再次使用，它们最终会被删除。

库高速缓存包括以下两个结构：

- 共享 **SQL**：共享 **SQL** 为针对数据库运行的 **SQL** 语句存储并共享执行计划和语法分析树。下次运行同一 **SQL** 语句时，这个语句就能利用共享 **SQL** 提供的语法分析信息来加快其执行速度。要确保 **SQL** 语句随时可以使用共享 **SQL** 区，文本、方案和绑定变量必须完全相同。
- 共享 **PL/SQL**：共享 **PL/SQL** 区存储并共享最近执行的 **PL/SQL** 语句。经过语法分析和编译的程序单元和过程（函数、程序包和触发器）都存储在这个区中。

数据字典高速缓存

- 数据库中最最近使用的定义的集合
- 包括与数据库文件、表、索引、列、用户、权限和其它数据库对象相关的信息
- 在语法分析阶段，服务器进程会在数据字典中查找用于解析对象名和验证访问的信息
- 将数据字典信息高速缓存到内存中，可缩短查询和 DML 的响应时间
- 大小由共享池的大小决定

ORACLE

1-17

Copyright © Oracle Corporation, 2001. All rights reserved.

数据字典高速缓存

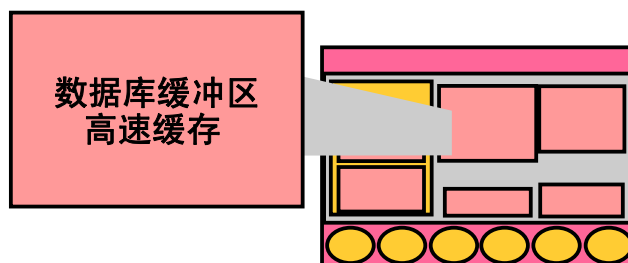
数据字典高速缓存也称作字典高速缓存或行高速缓存。将数据字典信息同时高速缓存到数据库缓冲区和共享池内存中，可以提高性能。有关数据库（用户帐户数据、数据文件名、段名、区的位置、表的说明和用户权限）的信息都存储在数据字典表中。当服务器需要用到这类信息时，将会读取数据字典表，返回的数据将存储在数据字典高速缓存中。

调整数据字典的大小：

数据字典的整体大小取决于共享池的大小，并由数据库进行内部管理。如果数据字典高速缓存太小，那么数据库必须对数据字典表进行反复地查询，才能获得服务器所需要的信息。这些查询称作递归调用，它在速度上要慢于对数据字典高速缓存所进行的直接查询，因为直接查询不使用 SQL。

数据库缓冲区高速缓存

- 存储已从数据文件中检索到的数据块的副本
- 能够大幅提高获取和更新数据时的性能
- 通过 LRU 算法管理
- 主块的大小由 DB_BLOCK_SIZE 确定



ORACLE

1-18

Copyright © Oracle Corporation, 2001. All rights reserved.

数据库缓冲区高速缓存

处理查询时，Oracle 服务器进程在数据库缓冲区高速缓存中查找任何所需的块。如果未在数据库缓冲区高速缓存中找到这个块，服务器进程就从数据文件读取这个块，并在数据库缓冲区高速缓存中放置一个副本。由于对同一个块的后续请求可以在内存中找到这个块，因此这些请求可能不需要进行物理读取。Oracle 服务器使用 LRU 算法来释放近期未被访问的缓冲区，以便在数据库缓冲区高速缓存中为新块腾出空间。

数据库缓冲区高速缓存

- 由独立的子高速缓存组成：
 - DB_CACHE_SIZE
 - DB_KEEP_CACHE_SIZE
 - DB_RECYCLE_CACHE_SIZE

- 大小可以进行动态调整

```
ALTER SYSTEM SET DB_CACHE_SIZE = 96M;
```

- 设置 DB_CACHE_ADVICE 可收集用于预测不同高速缓存大小行为的统计信息
- 统计信息由 V\$DB_CACHE_ADVICE 显示

ORACLE

1-19

Copyright © Oracle Corporation, 2001. All rights reserved.

数据库缓冲区高速缓存

调整数据库缓冲区高速缓存的大小：

数据库缓冲区高速缓存中每个缓冲区的大小等于一个 Oracle 块的大小，它由 DB_BLOCK_SIZE 参数来指定。数据库缓冲区高速缓存由独立的子高速缓存组成，子高速缓存用于缓冲区池和多个块大小。参数 DB_BLOCK_SIZE 确定主块的大小，主块的大小用于 SYSTEM 表空间。

以下三个参数定义了数据库缓冲区高速缓存的大小：

- DB_CACHE_SIZE：只调整缺省缓冲区高速缓存的大小，这个参数始终存在且不能设置为零
- DB_KEEP_CACHE_SIZE：调整保留缓冲区高速缓存的大小，用于保留内存中很可能会重新使用的块
- DB_RECYCLE_CACHE_SIZE：调整循环缓冲区高速缓存的大小，用于删除内存中重新使用的可能性很小的块

数据库缓冲区高速缓存（续）

缓冲区高速缓存咨询功能：

缓冲区高速缓存咨询功能可以启用和禁用统计信息的收集，这些统计信息用于预测不同高速缓存大小的行为。这些统计信息所提供的信息可以帮助您针对具体的工作量，调整数据库缓冲区高速缓存的大小，以达到最佳性能。缓冲区高速缓存咨询信息通过 `V$DB_CACHE_ADVICE` 视图收集和显示。

可以通过初始化参数 `DB_CACHE_ADVICE` 启用缓冲区高速缓存咨询功能。这个参数是一个动态参数，可以使用 `ALTER SYSTEM` 来更改。可供使用的三个值是 `OFF`、`ON` 和 `READY`。

`DB_CACHE_ADVICE` 参数值：

`OFF`：咨询功能关闭，并且没有为咨询功能分配内存

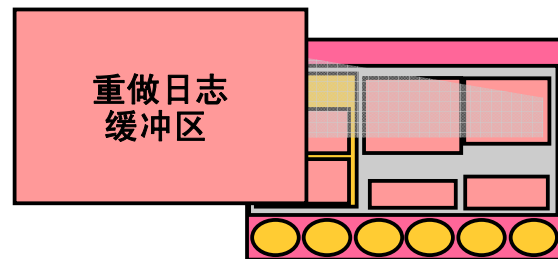
`ON`：咨询功能打开，并产生 CPU 和内存开销

如果参数处于 `OFF` 状态，在尝试将它设置为 `ON` 状态时会产生 `ORA-4031` 错误：当参数切换到 `ON` 时，无法从共享池分配内存。如果参数处于 `READY` 状态，则能正确地设置为 `ON`，因为此时已为咨询功能分配内存。

`READY`：咨询功能关闭，但已经为咨询功能分配内存。在实际打开咨询功能前分配内存可以避免产生 `ORA-4031` 错误。如果将参数从 `OFF` 状态直接切换到 `ON`，可能会产生 `ORA-4031` 错误。

重做日志缓冲区

- 记录对数据库数据块所做的全部更改
- 主要用于恢复
- 其中记录的更改称作重做条目
- 重做条目包含用于重新构造或重做更改的信息
- 大小由 LOG_BUFFER 定义



ORACLE

1-21

Copyright © Oracle Corporation, 2001. All rights reserved.

重做日志缓冲区

重做日志缓冲区是一个循环缓冲区，它包含对数据文件块所做的各种更改。此信息存储在重做条目中。重做条目包含将数据恢复到使用 INSERT、UPDATE、DELETE、CREATE、ALTER、或 DROP 操作进行更改前的状态所需要的信息。

调整重做日志缓冲区的大小：

重做日志缓冲区的大小由初始化参数 LOG_BUFFER 定义。

注：*Oracle9i 数据库性能优化* 课程将详细介绍调整重做日志缓冲区的大小。有关重做日志文件的详细信息，请参见“管理重做日志文件”这一课。

大型共享池

- **SGA 中的可选内存区**
- **分担了共享池的一部分工作**
- **用于：**
 - 共享服务器的会话内存 (UGA)
 - I/O 服务器进程
 - 备份和恢复操作或 RMAN
 - 并行执行消息缓冲区
 - 将 `PARALLEL_AUTOMATIC_TUNING` 设置为 `TRUE`
- **不使用 LRU 列表**
- **大小由 `LARGE_POOL_SIZE` 确定**

ORACLE

1-22

Copyright © Oracle Corporation, 2001. All rights reserved.

大型共享池

通过从大型共享池为共享服务器、Oracle XA 或并行查询缓冲区分配会话内存，Oracle 可将共享池主要用于高速缓存共享的 SQL 语句。这样，便减轻了共享池中各区的工作负担。共享池无需再提供内存以高速缓存 SQL 语法分析树，来支持共享服务器会话信息、I/O 以及备份和恢复进程。由于增减共享 SQL 高速缓存的开销降低，性能得以提高。

备份和恢复：

如果设置了 `BACKUP_DISK_IO= n` 和 `BACKUP_TAPE_IO_SLAVE = TRUE` 参数，则恢复管理器 (RMAN) 将使用大型共享池。如果大型共享池已经配置，但不够大，则在大型共享池中分配内存就会失败。RMAN 将错误消息写入警报日志文件，而且不使用 I/O 操作进行备份或恢复操作。

并行执行：

如果将 `PARALLEL_AUTOMATIC_TUNING` 设置为 `TRUE`，将使用大型共享池。否则，将把这些缓冲区分配至共享池。

大型共享池（续）

调整大型共享池的大小：

大型共享池的大小由 `LARGE_POOL_SIZE` 参数定义，单位是字节。该参数不是动态参数。

大型共享池和 **LRU** 列表：

大型共享池并不包含 **LRU** 列表。它与使用 **LRU** 列表的共享池中的保留空间不同。

Java 池

- 存储 Java 命令的服务分析要求
- 在安装并使用 Java 时是必需的
- 大小由 `JAVA_POOL_SIZE` 参数确定

ORACLE

1-24

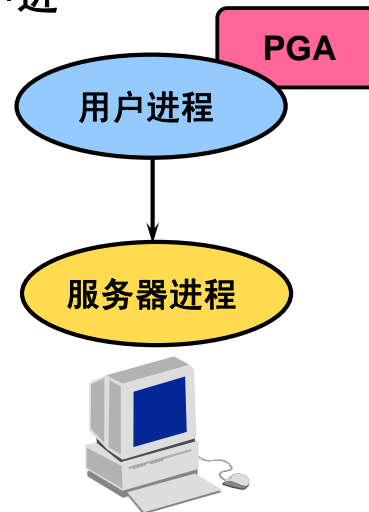
Copyright © Oracle Corporation, 2001. All rights reserved.

Java 池

Java 池是一个可选设置，但如果安装并使用 Java，则需要使用 Java 池。Java 池大小可以使用 `JAVA_POOL_SIZE` 参数设置，单位为字节。在 Oracle9i，Java 池的缺省大小是 24 MB。

程序全局区

- 为连接到 Oracle 数据库的每个用户进程保留的内存
- 在创建进程时分配
- 在终止进程时回收
- 仅供一个进程使用



ORACLE

1-25

Copyright © Oracle Corporation, 2001. All rights reserved.

程序全局区 (PGA)

程序全局区或进程全局区 (PGA) 是内存区，它包含有关单个服务器进程或单个后台进程的数据和控制信息。PGA 在创建进程时分配，并在终止进程时回收。与由若干个进程共享的 SGA 相比，PGA 是仅供一个进程使用的区。

PGA 的内容：

PGA 内存的内容会因不同情况而变化，这取决于例程是在专用服务器配置还是在共享服务器配置下运行。一般来讲，PGA 内存包括下列组件：

- 专用 SQL 区：包含绑定信息和运行时内存结构之类的的数据。发出 SQL 语句的每个会话均拥有一个专用 SQL 区。提交同一 SQL 语句的每个用户都拥有自己的使用单个共享 SQL 区的专用 SQL 区。因此，许多专用 SQL 区都与同一个共享 SQL 区相关联。一个游标的专用 SQL 区可以分成以下两个区：
 - 永久区：包含绑定信息，并且只在关闭游标时释放
 - 运行时区：在执行请求时的第一步创建。对于 INSERT、UPDATE 和 DELETE 命令，该区在执行语句后释放，对于查询操作，该区只在提取所有行或取消查询后释放。

程序全局区（续）

- 专用 SQL 区（续）：专用 SQL 区的位置取决于为会话建立的连接类型。在专用服务器环境中，专用 SQL 区位于各自服务器进程的 PGA 中。在共享服务器环境中，专用 SQL 区位于 SGA 中。

管理专用 SQL 区是用户进程的职责。用户进程可以分配的专用 SQL 区的数目始终由初始化参数 OPEN_CURSORS 来限制。该参数的缺省值是 50。

- 会话内存：包含为保留会话变量以及与该会话相关的其它信息而分配的内存。对于共享服务器环境，该会话是共享的而不是专用的。
- SQL 工作区：用于大量占用内存的操作，如排序、散列联接、位图合并和位图创建。工作区的大小可进行控制和调整。

自 Oracle9i 起，工作区的大小可以进行自动的全局管理。要实现这种功能，可将 WORKAREA_SIZE_POLICY 参数设置成缺省情况下的 AUTO，同时设置 PGA_AGGREGATE_TARGET 初始化参数。DBA 可以对 PGA_AGGREGATE_TARGET 参数进行设置，以指定该例程可以使用的 PGA 内存的目标聚集数目。此参数只是一个指标，可以由 DBA 在例程一级动态修改。设置值的单位可以是字节数、千字节数、兆字节数或吉字节数。设置完这些参数后，工作区的大小即可自动调整，这些会话将忽略所有的 *_AREA_SIZE 参数。

在 Oracle9i 之前的版本中，由 DBA 控制着 SQL 工作区的最大大小，方法是设置以下参数：SORT_AREA_SIZE、HASH_AREA_SIZE、BITMAP_MERGE_AREA_SIZE 和 CREATE_BITMAP_AREA_SIZE。设置这些参数是很困难的，因为工作区的最大大小在理论上是基于输入的数据大小以及系统中活动工作区的数目来选择的。但是，以上两种因素在不同工作区和不同时间差别很大。因此，很难在最佳环境下调整这些参数。

程序全局区（续）

专用服务器和共享服务器之间的内存分配差异：

PGA 内存的内容会因不同情况而变化，这取决于例程是在专用服务器配置下还是在共享服务器配置下运行。一般来讲，PGA 内存包括下列组件：

内存区	专用服务器	共享服务器
会话内存的特性	专用	共享
永久区的位置	PGA	SGA
运行时区的位置 (SELECT)	PGA	SGA
运行时区的位置 (DML/DDL)	PGA	PGA

进程结构

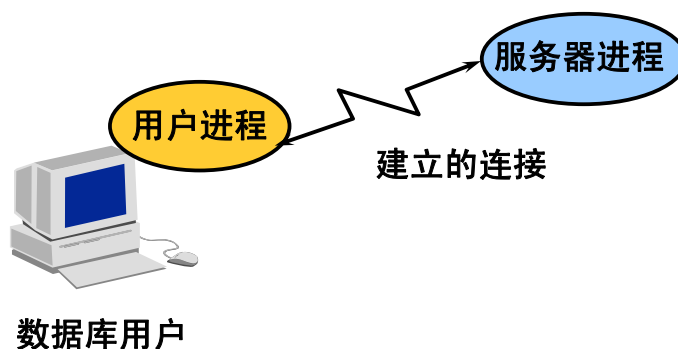
Oracle 利用了以下几种进程的优势：

- **用户进程：** 在数据库用户请求连接到 **Oracle** 服务器时启动
- **服务器进程：** 与 **Oracle** 例程相连接，在用户建立会话时启动
- **后台进程：** 在 **Oracle** 例程启动时启动

ORACLE

用户进程

- 请求与 Oracle 服务器交互的程序
- 必须先建立连接
- 不与 Oracle 服务器直接交互



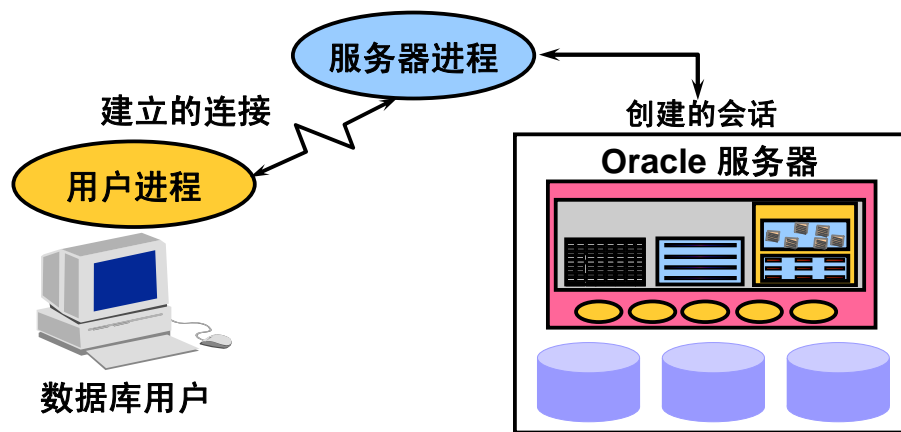
ORACLE

用户进程

需要从数据库请求信息的数据库用户必须先建立与 Oracle 服务器的连接。连接由 SQL*Plus 这类的数据库接口工具请求，并开始用户进程。用户进程并不与 Oracle 服务器直接进行交互操作，而是通过用户程序接口 (UPI) 生成各种调用。用户程序接口则可以创建会话并启动服务器进程。

服务器进程

- 直接与 Oracle 服务器交互的程序
- 执行生成的调用并返回相关结果
- 可以是专用服务器或共享服务器



ORACLE

1-30

Copyright © Oracle Corporation, 2001. All rights reserved.

服务器进程

用户建立连接后，服务器进程便会启动，以处理用户进程的请求。服务器进程可以是专用服务器进程或共享服务器进程。在专用服务器环境中，服务器进程只处理一个用户进程的请求。用户进程断开连接后，服务器进程就会终止。在共享服务器环境中，服务器进程将处理多个用户进程的请求。服务器进程可以通过 Oracle 程序接口 (OPI) 与 Oracle 服务器进行通信。

注：*Oracle9i 数据库性能优化* 课程将详细介绍服务器进程在专用服务器环境与共享服务器环境中的分配情况。

后台进程

维护并加强物理结构与内存结构之间的关系

- 必备的后台进程:

- DBWn PMON CKPT
- LGWR SMON

- 可选的后台进程:

- ARCn LMDn RECO
- CJQ0 LMON Snnn
- Dnnn Pnnn
- LCKn QMNn

ORACLE

1-31

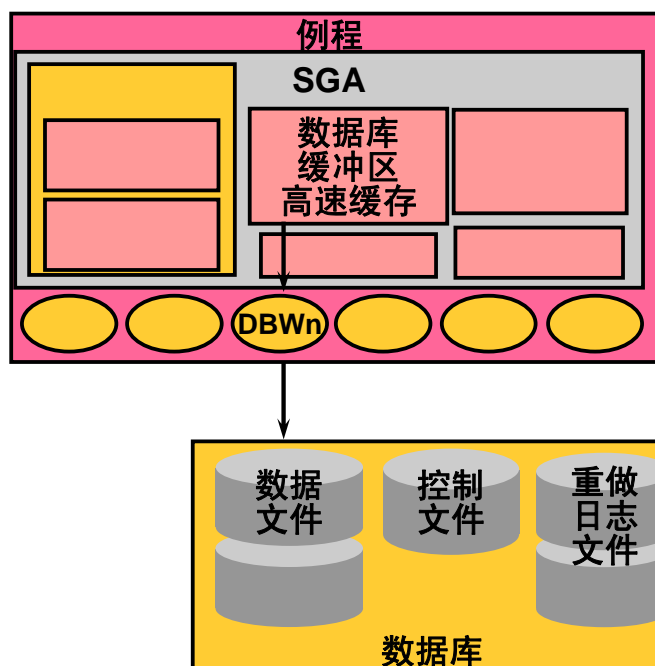
Copyright © Oracle Corporation, 2001. All rights reserved.

后台进程

Oracle 体系结构有五个必备的后台进程，本课将详细介绍这些进程。除了必备的后台进程列表以外，Oracle 体系结构中还有很多可选的后台进程。如果选择使用这些可选的后台进程，它们便会启动。除了 ARCn 后台进程之外，这些可选的进程不在本课所涉及的范围之内。下面列出了一些可选的后台进程：

- RECO: 恢复程序
- QMNn: 高级排队
- ARCn: 归档程序
- LCKn: RAC 锁管理器 – 例程锁
- LMON: RAC DLM 监控程序 – 全局锁
- LMDn: RAC DLM 监控程序 – 远程锁
- CJQ0: 协调程序作业队列后台进程
- Dnnn: 调度程序
- Snnn: 共享服务器
- Pnnn: 并行查询从属

数据库写入程序 (DBWn)



在以下情况下写入:

- 出现检查点
- 灰数据缓冲区达到阈值
- 没有空闲缓冲区
- 出现超时
- 执行了 RAC ping 请求
- 表空间处于 OFFLINE 状态
- 表空间处于 READ ONLY 状态
- 对表执行 DROP 或 TRUNCATE 操作
- 对表空间执行 BEGIN BACKUP 操作

ORACLE

1-32

Copyright © Oracle Corporation, 2001. All rights reserved.

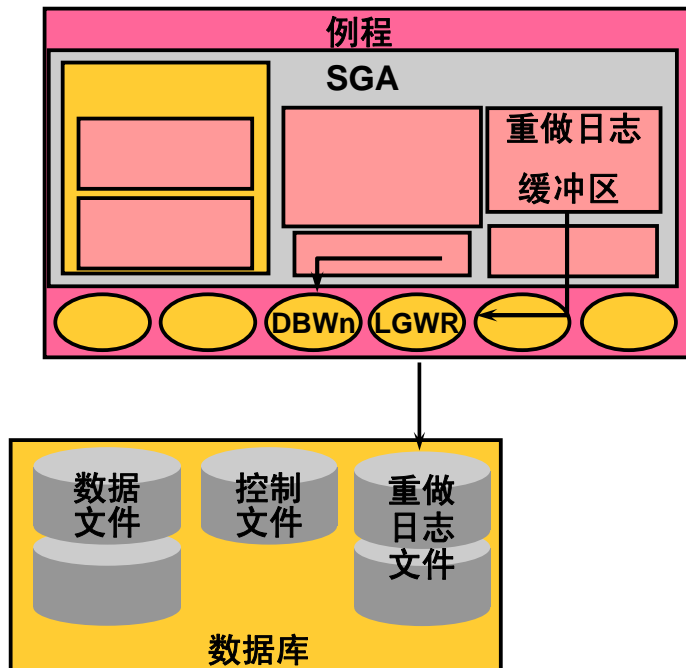
数据库写入程序 (DBWn)

服务器进程在数据库缓冲区高速缓存中记录对还原块和数据块所做的更改。DBWn 将数据库缓冲区高速缓存中的灰数据缓冲区写入数据文件。这可确保数据库缓冲区高速缓存中有足够数量的空闲缓冲区（即当服务器进程需要读取数据文件中的块时可以覆盖的缓冲区）可用。由于服务器进程只在数据库缓冲区高速缓存中进行更改，因此提高了数据库的性能。

DBWn 延迟写入数据文件, 直到发生下列事件之一:

- 增量或正常检查点
- 灰数据缓冲区的数量达到阈值
- 进程扫描指定数量的块而无法找到任何空闲缓冲区时
- 出现超时
- 实时应用集群 (Real Application Clusters, RAC) 环境中出现 ping 请求
- 使一般表空间或临时表空间处于脱机状态
- 使表空间处于只读模式
- 删除或截断表
- 执行 ALTER TABLESPACE 表空间名 BEGIN BACKUP 操作

日志写入器 (LGWR)



LGWR 在以下情况下写入:

- 提交时
- 三分之一填满时
- 有 1 MB 的重做时
- 每隔三秒
- DBWn 写入前

日志写入器 (LGWR)

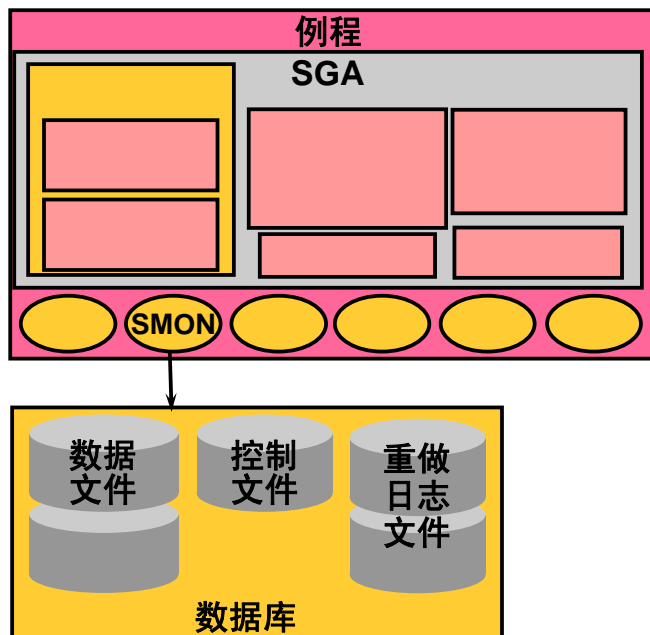
LGWR 在下列情况下执行从重做日志缓冲区到重做日志文件的连续写入:

- 当提交事务时
- 当重做日志缓冲区的三分之一填满时
- 当重做日志缓冲区中记录了超过 1 MB 的更改时
- 在 DBWn 将数据库缓冲区高速缓存中修改的块写入数据文件以前
- 每隔三秒

因为恢复操作需要重做, 所以 LGWR 只在重做写入磁盘后确认提交操作。

LGWR 还可以调用 DBWn 来写入数据文件。

系统监控程序 (SMON)



职责:

- 例程恢复
 - 前滚重做日志中的更改
 - 打开数据库供用户访问
 - 回退未提交的事务处理
- 合并空闲空间
- 回收临时段

ORACLE

1-34

Copyright © Oracle Corporation, 2001. All rights reserved.

系统监控程序 (SMON)

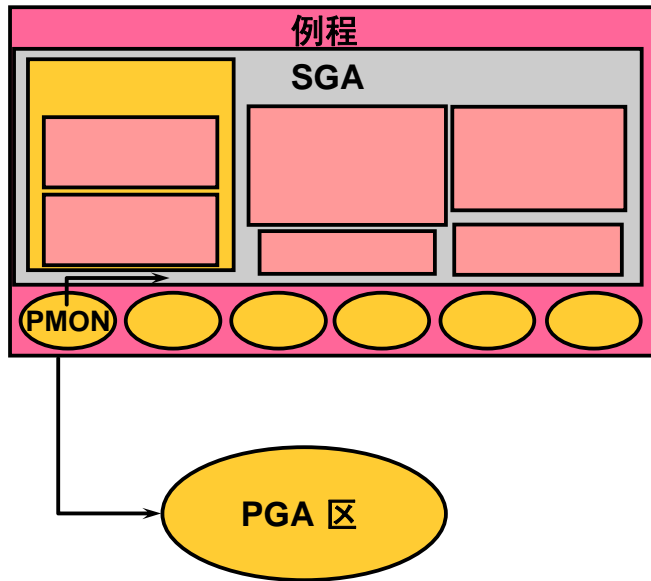
如果 Oracle 例程失败，那么 SGA 中尚未写入磁盘的所有信息都会丢失。例如，操作系统的失败导致例程失败。例程丢失后，后台进程 SMON 在数据库重新打开时自动执行例程恢复。恢复例程需要执行以下步骤：

1. 前滚以恢复尚未记入数据文件但已经记入联机重做日志中的数据。由于例程失败时 SGA 的丢失，所以尚未将这些数据写入磁盘。在该进程中，SMON 读取重做日志文件并将重做日志中记录的更改应用到数据块中。由于提交的所有事务处理都已写入重做日志，因此该进程完全恢复了这些事务处理。
2. 打开数据库以便用户可以登录。未被未恢复事务处理锁定的任何数据都立即可用。
3. 回退未提交的事务处理。它们由 SMON 回退，或在访问锁定的数据时由单个服务器进程回退。

SMON 也执行一些空间维护功能：

- 它联合或合并数据文件中空闲空间的邻近区域。
- 它回收临时段，将它们作为数据文件中的空闲空间返回。

过程监视器 (PMON)



例程失败后，通过以下方法进行清理：

- 回退事务处理
- 释放锁
- 释放其它资源
- 重新启动已失效的调度程序

ORACLE

1-35

Copyright © Oracle Corporation, 2001. All rights reserved.

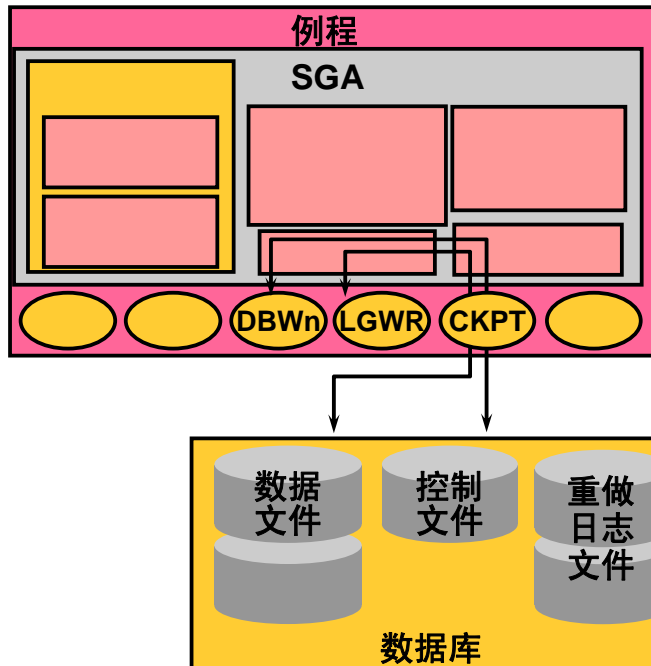
过程监视器 (PMON)

进程失败后，后台进程 PMON 通过下面的方法进行清理：

- 回退用户的当前事务处理
- 释放当前保留的所有表锁或行锁
- 释放用户当前保留的其它资源
- 重新启动已失效的调度程序

Oracle9i 数据库管理基础 II 课程中详细介绍了调度程序。

检查点 (CKPT)



职责包括：

- 在检查点发信号给 DBWn
- 使用检查点信息更新数据文件的标头
- 使用检查点信息更新控制文件

ORACLE

1-36

Copyright © Oracle Corporation, 2001. All rights reserved.

检查点 (CKPT)

每隔三秒，CKPT 进程就会向控制文件存储数据，以标识重做日志文件中恢复操作的起始位置，该操作称作检查点。检查点的用途是确保数据库缓冲区高速缓存中在时间点之间发生修改的所有缓冲区内容都已写入数据文件。这个时间点（称作检查点位置）是例程失败时开始恢复数据库的位置。DBWn 应将数据库缓冲区高速缓存中在该时间点之前发生修改的所有缓冲区内容写入数据文件。对于 Oracle9i 之前的版本，这项操作在重做日志的结尾处执行。切换日志时，CKPT 还将这个检查点的信息写入数据文件的头部。

启动检查点的原因如下：

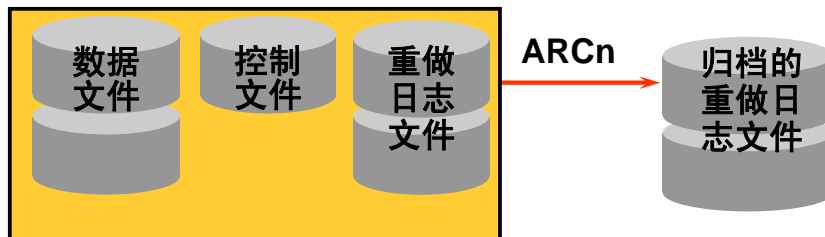
- 确保定期向磁盘写入内存中发生修改的数据块，以便在系统或数据库失败时不会丢失数据
- 缩短例程恢复所需的时间。只需处理最后一个检查点后面的重做日志条目以启动恢复操作
- 确保提交的所有数据在关闭期间均已写入数据文件

由 CKPT 写入的检查点信息包括检查点位置、系统更改号、重做日志中恢复操作的起始位置以及有关日志的信息等等。

注：CKPT 并不将数据块写入磁盘，或将重做块写入联机重做日志。

归档程序 (ARCn)

- 可选的后台进程
- 设置 ARCHIVELOG 模式时自动归档联机重做日志
- 保留数据库的全部更改记录



ORACLE

1-37

Copyright © Oracle Corporation, 2001. All rights reserved.

归档程序 (ARCn)

所有其它后台进程都是可选的，这将取决于数据库的配置；但是，其中的 ARCn 进程对于丢失磁盘数据后的数据库恢复起着至关重要的作用。当联机重做日志文件填满时，Oracle 服务器开始写入下一个联机重做日志文件。从一个重做日志到另一个重做日志的切换过程称为日志切换。ARCn 进程在每次日志切换时备份或归档已满的日志组。在日志能够重新使用之前，它自动将联机重做日志归档，从而保留对数据库所做的全部更改。这样，即使磁盘驱动器损坏，DBA 也能够将数据库恢复到出现故障前的状态。

归档重做日志文件：

DBA 必须做出的一个重要决策是配置数据库以 ARCHIVELOG 模式还是以 NOARCHIVELOG 模式操作。

NOARCHIVELOG 模式：在 NOARCHIVELOG 模式下，每次发生日志切换时，就会覆盖联机重做日志文件。只有重做日志组的检查点完成后，LGWR 才会覆盖该日志组。这就确保发生例程崩溃时提交的数据能够得以恢复。在例程崩溃的过程中，只会丢失 SGA 中的数据。磁盘数据不会丢失，而只会丢失内存中的数据。例如，当操作系统的崩溃引起例程崩溃时。

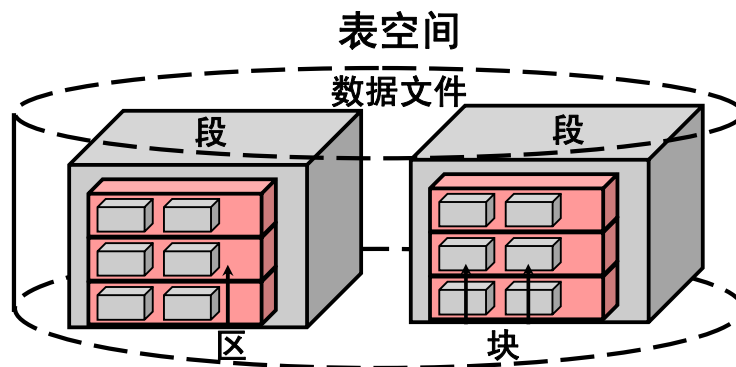
将重做日志文件归档（续）

ARCHIVELOG 模式：如果将数据库配置为以 ARCHIVELOG 模式运行，那么已满的联机重做日志文件的非活动组必须归档之后才能够再次使用。因为对数据库所做的更改记录在联机重做日志文件中，所以数据库管理员能够使用数据文件的物理备份和归档的联机重做日志文件来恢复数据库，而不会由于任何单点故障（包括磁盘数据的丢失）而丢失任何已提交的数据。通常应将生产数据库配置为以 ARCHIVELOG 模式运行。

Oracle9i 数据库管理基础 II 课程中详细介绍了归档日志模式。

逻辑结构

- 指示数据库物理空间的使用情况
- 层次结构由表空间、段、区和块组成



ORACLE

1-39

Copyright © Oracle Corporation, 2001. All rights reserved.

逻辑结构

逻辑结构的层次如下所述：

- Oracle 数据库至少包含一个表空间。
- 表空间包含一个或多个段。
- 段由区组成。
- 区由逻辑块组成。
- 块是读写操作的最小单元。

Oracle 数据库体系结构包括构成数据库的逻辑结构和物理结构。

- 物理结构包括构成数据库的控制文件、联机重做日志文件和数据文件。
- 逻辑结构包括表空间、段、区和数据块。

Oracle 服务器通过表空间和逻辑存储结构（包括段、区和数据块），使您能够实现磁盘空间使用的小粒度控制。

逻辑结构（续）

表空间：

Oracle 数据库内的数据存储在表空间内。

- Oracle 数据库可以从逻辑上分组到称为表空间的更小的逻辑空间区。
- 一个表空间在某一时刻只能属于一个数据库。
- 每个表空间由一个或多个称为数据文件的操作系统文件组成。
- 表空间可能包含一个或多个段。
- 表空间可以在数据库运行时联机。
- 除了 SYSTEM 表空间或者有活动还原段的表空间，可将其它表空间置于脱机状态而不会影响数据库运行。
- 表空间可以在可读写和只读状态之间切换。

数据文件（不是逻辑结构）：

- Oracle 数据库内的每个表空间由一个或者多个称为数据文件的文件组成。这些物理结构与在其上运行 Oracle 服务器的操作系统是一致的。
- 一个数据文件只能属于一个表空间。
- 通过分配指定数量的磁盘空间加上少量的开销，Oracle 服务器创建表空间数据文件。
- 数据文件创建后，数据库管理员可以更改其大小或者指定数据文件应随着表空间内对象的增长而动态增长。

段：

- 段是在表空间中为特定逻辑存储结构分配的空间。
- 表空间可以由一个或多个段组成。
- 段无法跨越表空间；但是段可以跨越属于同一表空间的多个数据文件。
- 每个段由一个或多个区组成。

区：

按区向段分配空间。

- 一个或多个区组成一个段。
 - 当段创建后，它至少由一个区组成。
 - 随着段的增长，需要向该段添加区。
 - DBA 可以手动向段添加区。
- 一个区就是一组连续的 Oracle 块。
- 区无法跨越数据文件，所以，它必须存在于一个数据文件内。

逻辑结构（续）

数据块：

Oracle 服务器以 Oracle 块或者数据块为单位来管理数据文件中的存储空间。

- Oracle 数据库内的数据存储在数据块内，数据块为最精细的粒度等级。
- Oracle 数据块是 Oracle 服务器能够分配、读或写的最小存储单元。
- 一个数据块对应一个或多个从现有数据文件中分配的操作系统块。
- 每个 Oracle 数据库的标准数据块大小是在创建数据库时由初始化参数 `DB_BLOCK_SIZE` 指定的。
- 数据块大小应当是操作系统块大小的整数倍以避免不必要的 I/O。
- 数据块大小最大值取决于操作系统。

处理 SQL 语句

- 通过以下进程连接到一个例程：
 - 用户进程
 - 服务器进程
- 所用的 Oracle 服务器组件取决于 SQL 语句的类型：
 - 查询语句返回行
 - DML 语句记录更改
 - 提交操作确保事务处理的恢复
- 有些 Oracle 服务器组件不参与 SQL 语句的处理

ORACLE

1-42

Copyright © Oracle Corporation, 2001. All rights reserved.

处理 SQL 语句

处理查询：

- 语法分析：
 - 搜索同一语句
 - 检查语法、对象名和权限
 - 锁定语法分析过程中使用的对象
 - 创建和存储执行计划
- 绑定：获取变量值
- 执行：处理语句
- 提取：将结果行返回用户进程

处理 SQL 语句（续）

处理 DML 语句：

- 语法分析：与处理查询时的语法分析阶段相同。
- 绑定：与处理查询时的绑定阶段相同。
- 执行：
 - 如果数据库缓冲区高速缓存中不存在某些数据块和还原块，服务器进程就从数据文件将它们读入数据库缓冲区高速缓存。
 - 服务器进程锁定要进行修改的行。还原块用于存储数据的前像，以便在需要时回退 DML 语句。
 - 数据块记录数据的新值。
 - 服务器进程将数据的前像记录到回退块中，并更新数据块。这两种更改都是在数据库缓冲区高速缓存中进行的。数据库缓冲区高速缓存中所有已更改的块都标记为灰数据缓冲区，即与磁盘中相应的块不同的缓冲区。
 - DELETE 或 INSERT 命令的处理使用类似的步骤。DELETE 命令的前像包含已删除行中的列值，而 INSERT 命令的前像中包含行的位置信息。

处理 DDL 语句：

DDL（数据定义语言）语句的执行与 DML（数据操纵语言）语句和查询的执行不尽相同，因为成功执行 DDL 语句需要对数据字典具有写权限。对于这些语句，语法分析阶段实际上包括分析、数据字典查找和执行。事务处理管理 SQL 语句、会话管理 SQL 语句和系统管理 SQL 语句在语法分析和执行阶段处理。要重新执行这些语句，再次进入执行阶段即可。

小结

在这一课中，您应该能够掌握：

- 解释数据库文件：数据文件、控制文件和联机重做日志
- 解释 **SGA** 内存结构：数据库缓冲区高速缓存、共享池和重做日志缓冲区
- 解释主要的后台进程：**DBWn**、**LGWR**、**CKPT**、**PMON**、**SMON**
- 解释后台进程 **ARCn** 的用法
- 列出可选后台进程和条件后台进程
- 解释逻辑层次结构

ORACLE

练习 1 概览

此练习涉及以下主题：

- 复习体系结构组件
- 用户连接到 **Oracle** 例程的过程中所涉及的结构

ORACLE

练习 1: Oracle 体系结构组件

- 1 下面哪个描述是正确的?
 - a Oracle 服务器是由三类文件组成的数据集合。
 - b 用户通过启动 Oracle 例程建立与数据库的连接。
 - c 连接是 Oracle 服务器和 Oracle 例程之间的通信路径。
 - d 会话在 Oracle 服务器验证用户后启动。
- 2 以下哪个内存区不属于 SGA?
 - a 数据库缓冲区高速缓存
 - b PGA
 - c 重做日志缓冲区
 - d 共享池
- 3 下面哪两个关于共享池的描述是正确的?
 - a 共享池包含库高速缓存、数据字典高速缓存、共享 SQL 区、Java 池和大型共享池。
 - b 共享池用于存储最近执行的 SQL 语句。
 - c 共享池用于可以全局共享的对象。
 - d 库高速缓存包含共享 SQL 区和共享 PL/SQL 区。
- 4 以下哪个内存区用于高速缓存数据字典信息?
 - a 数据库缓冲区高速缓存
 - b PGA
 - c 重做日志缓冲区
 - d 共享池
- 5 重做日志缓冲区的主要用途是记录对数据库的数据块所做的所有更改。
 - a 对
 - b 错
- 6 PGA 是一个内存区, 它包含有关多个服务器进程或多个后台进程的数据和控制信息。
 - a 对
 - b 错
- 7 启动 Oracle 例程后, 下面哪个进程是可用的?
 - a 用户进程
 - b 服务器进程
 - c 后台进程
- 8 列出五个必备的后台进程。

练习 1: Oracle 体系结构组件 (续)

- 9 请将下列处理过程与它们所执行的任务相连。
- | | |
|-----------|-----------------|
| a 数据库写入程序 | E 帮助写入到数据文件头部 |
| b 日志写入器 | C 负责例程恢复 |
| c 系统监控程序 | D 进程失败后进行清理 |
| d 进程监控程序 | B 记录数据库更改以便进行恢复 |
| e 检查点 | A 将灰数据缓冲区写入数据文件 |
- 10 Oracle 数据库的物理结构包括控制文件、数据文件和重做日志文件。
- a 对
 - b 错
- 11 从数据库开始按组成层次排列下列结构。
- a 表空间
 - b 区
 - c 段
 - d 数据库
 - e 块

- 12 列出 Oracle 服务器的组件。

- 13 列出 Oracle 例程的组件。

- 14 列出组成 Oracle 数据库的三种文件类型。

2

Oracle 服务器入门

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

目标

完成这一课的学习后，您应该能：

- 了解 DBA 可以使用的常见数据库管理工具
- 了解 Oracle Universal Installer 的特性
- 使用 SQL*Plus 操作和控制 Oracle 数据库
- 列举 Oracle Enterprise Manager 的主要组件

ORACLE

数据库管理工具

工具	说明
Oracle Universal Installer (OUI)	用于安装、升级或删除软件组件
Oracle Database Configuration Assistant	与 OUI 进行交互的图形用户界面工具，也可单独使用，用于创建、删除或修改数据库
SQL*Plus	用于访问 Oracle 数据库中数据的实用程序
Oracle Enterprise Manager	用于管理、监视和优化一个或多个数据库的图形界面

ORACLE

2-3

Copyright © Oracle Corporation, 2001. All rights reserved.

常见数据库管理工具的示例

本课程将论述以上列出的工具，但它们只是 Oracle 提供的实用程序的一部分。

Oracle Universal Installer

- 用于安装、升级或删除软件组件，还用于创建数据库
- 基于 Java 引擎
- 具有以下特性：
 - 自动的相关性解析
 - 可以进行基于 Web 的安装
 - 跟踪组件和套件安装的清单
 - 可卸装已安装的组件
 - 支持多个 Oracle 主目录
 - 支持全球化技术

ORACLE

2-4

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Universal Installer 的特性

基于 Java 的 Oracle Universal Installer 为所有支持 Java 的平台提供了安装解决方案，允许使用通用的安装过程，并且不要求用户具有平台方面的经验。

Universal Installer 具有以下功能：

- 检测组件之间的相关性并根据检测结果执行安装。
- 可用于指向定义了发布或登台区域的 URL，并可通过 HTTP 远程安装软件。
- 可用于删除已安装的产品。卸装操作是对安装操作的“撤消”。
- 维护目标计算机上所有 Oracle 主目录的清单，其中包括主目录的名称、产品以及主目录上安装的产品版本。
- 检测操作系统的语言，并用该语言运行安装会话。
- 可在交互模式或静默模式下运行。Oracle Universal Installer 使用响应文件在静默（或非交互）模式下运行。

启动 Universal Installer

- 在 UNIX 上，使用以下命令启动 Oracle Universal Installer:

```
$ ./runInstaller
```

- 在 NT 上，使用以下命令启动 Oracle Universal Installer:

```
D:\> setup
```

ORACLE

2-5

Copyright © Oracle Corporation, 2001. All rights reserved.

交互式安装

UNIX:

安装程序名为 `runInstaller`，位于 `INSTALL\install\solaris` 目录下。

在 UNIX 上，请不要以根用户身份运行 Installer。

NT:

安装程序名为 `setup.exe`，位于 `install/win32` 目录下。

注：有关在您所用的平台上安装 Oracle Server 的信息，请参阅专用于该操作系统的 Oracle 文档。

使用响应文件进行非交互式安装

- 无需用户交互
- 响应文件：
 - 必须进行编辑的模板
 - 包含变量和值的文本文件
 - 自定义的参数
- 使用以下命令以非交互模式启动 **Universal Installer**:

```
./runInstaller -responsefile myrespfile -silent
```

ORACLE

2-6

Copyright © Oracle Corporation, 2001. All rights reserved.

使用响应文件进行非交互式安装

如果不希望由用户进行操作或使用非图形终端进行安装时，将执行非交互式安装。

安装参数是使用响应文件自定义的。响应文件是一个文本文件，其中包含 Oracle Universal Installer 在安装过程中使用的变量和值。例如，安装参数可以是 ORACLE_HOME 的值以及安装类型，如“典型安装”或“自定义安装”。

用户需首先复制和编辑响应文件以指定要安装的组件。

UNIX:

响应文件模板位于 stage/response 目录中。

```
./runInstaller -responsefile filename [-silent] [-nowelcome]
```

NT:

响应文件模板位于 CD-ROM 上的 Response 目录中。

```
setup.exe -responsefile filename [-silent]
```

注：这不是字符模式。

以非交互模式启动 Oracle Universal Installer

示例：以非交互模式启动 Oracle Universal Installer

UNIX:

```
./runInstaller -responsefile FILENAME [-SILENT] [-NOWELCOME]
```

其中:

- **FILENAME:** 指定响应文件
- **SILENT:** 以静默模式运行 Oracle Universal Installer
- **NOWELCOME:** 不显示“欢迎”(Welcome) 窗口。如果指定了 SILENT, 则不必使用此参数

UNIX 上的响应文件示例:

```
[General]
```

```
RESPONSEFILE_VERSION=1.7.0
```

```
[Session]
```

```
UNIX_GROUP_NAME="dba"
```

```
FROM_LOCATION="/u01/stage/products.jar"
```

```
ORACLE_HOME="/u01/app/oracle/ora9i"
```

```
ORACLE_HOME_NAME="Ora9i"
```

```
TOPLEVEL_COMPONENT={"oracle.server", "9.0.1.1.1"}
```

```
SHOW_COMPONENT_LOCATIONS_PAGE=false
```

```
SHOW_SUMMARY_PAGE=false
```

```
SHOW_INSTALL_PROGRESS_PAGE=false
```

```
SHOW_REQUIRED_CONFIG_TOOL_PAGE=false
```

```
SHOW_OPTIONAL_CONFIG_TOOL_PAGE=false
```

```
SHOW_END_SESSION_PAGE=false
```

```
NEXT_SESSION=true
```

```
SHOW_SPLASH_SCREEN=true
```

```
SHOW_WELCOME_PAGE=false
```

```
SHOW_ROOTSH_CONFIRMATION=true
```

```
SHOW_EXIT_CONFIRMATION=true
```

```
INSTALL_TYPE="Typical"
```

```
s_GlobalDBName="u01.us.oracle.com"
```

```
s_mountPoint="/u01/app/oracle/ora9i/dbs"
```

```
s_dbSid="db09"
```

```
b_createDB=true
```

以非交互模式启动 Oracle Universal Installer

响应文件示例（续）：

- General 部分指定响应文件的版本号。
- Sessions 部分列出了 Universal Installer 的各种对话框。其中包括：
 - FROM_LOCATION 指定了要安装的产品源文件位置
 - ORACLE_HOME, ORACLE_HOME 的值
 - ORACLE_HOME_NAME, ORACLE_HOME_NAME 的值
 - SHOW_INSTALL_PROGRESS 是在安装阶段出现的安装进度页
 - 如果运行 root.sh 脚本前需要显示确认对话框，则将 SHOW_ROOTISH_CONFIRMATION 设置为 TRUE
 - 如果退出安装程序前需要显示确认对话框，则将 SHOW_EXIT_CONFIRMATION 设置为 TRUE

注：有关设置响应文件的完整信息，请参考针对各操作系统的安装指南。

Oracle Database Configuration Assistant

Oracle Database Configuration Assistant 可用于：

- 创建数据库
- 配置数据库选件
- 删除数据库
- 管理模板

ORACLE

2-9

Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Database Configuration Assistant

如何使用 Oracle Database Configuration Assistant 创建数据库在“创建数据库”一课中讲述。

数据库管理员用户

- 自动创建用户 **SYS** 和 **SYSTEM**
- 在创建数据库期间创建
- 授予 **DBA** 角色

SYS

- 口令: `change_on_install`
- 数据库数据字典的所有者

SYSTEM

- 口令: `manager`
- **Oracle** 工具使用的附加内部表和视图的所有者

ORACLE

2-10

Copyright © Oracle Corporation, 2001. All rights reserved.

数据库管理员用户

对 Oracle 服务器执行管理任务（如创建用户）需要特别的权限。**SYS** 和 **SYSTEM** 这两个数据库用户帐户是随数据库自动创建的，并被授予 **DBA** 角色。这种角色是随每个数据库自动创建的预定义角色。**DBA** 角色具有所有数据库系统权限。

SYS

创建数据库时，将创建用户 **SYS**，其初始标识口令为 `change_on_install`。**SYS** 拥有极其重要的数据字典。以 **SYS** 身份连接时应具有 **SYSDBA** 或 **SYSOPER** 角色。如果在不具有 **SYSDBA** 或 **SYSOPER** 权限的情况下尝试连接，将接收到错误消息：ORA-28009 应以 **SYSDBA** 或 **SYSOPER** 角色连接至 **SYS**。

SYSTEM

创建数据库时，还会自动创建 **SYSTEM** 用户，其初始标识口令为 `manager`。用户 **SYSTEM** 拥有的附加表和视图也随之创建。这些表和视图包含 Oracle 工具使用的管理信息。

根据创建数据库时所用模式的不同，即，手动创建还是使用 Database Creation Assistant 创建，可能还会创建一些附加用户。至少应创建一个附加管理员用户名，供执行日常管理任务时使用。

为安全起见，**SYS** 和 **SYSTEM** 的缺省口令应在创建数据库后立即更改。

SQL*Plus

- 一种提供下列功能的 Oracle 工具：
 - 操作和控制数据库
 - 启动和关闭数据库、创建和运行查询、添加行、修改数据和编写自定义报表
- 它是具有特定附加内容的标准 SQL 语言的一部分
- 连接至 SQL*Plus

```
sqlplus /nolog  
  
connect / as sysdba  
Connected to an idle instance.
```

ORACLE

SQL*Plus

SQL*Plus 是 Oracle 的命令行工具，可用于运行标准的 SQL（Structured Query Language，结构化查询语言）命令集。SQL 是一种功能性语言，用户可使用它与 Oracle 数据库进行通信，来检索、添加、更新或修改数据库中的数据。

Oracle Enterprise Manager

- 用作 DBA 的集中系统管理工具
- 管理、诊断和优化多个数据库的工具
- 从多个位置管理多个网络节点和服务的工具
- 用于与其他管理员共享任务
- 提供管理并行服务器和复制的数据库的工具

ORACLE

2-12

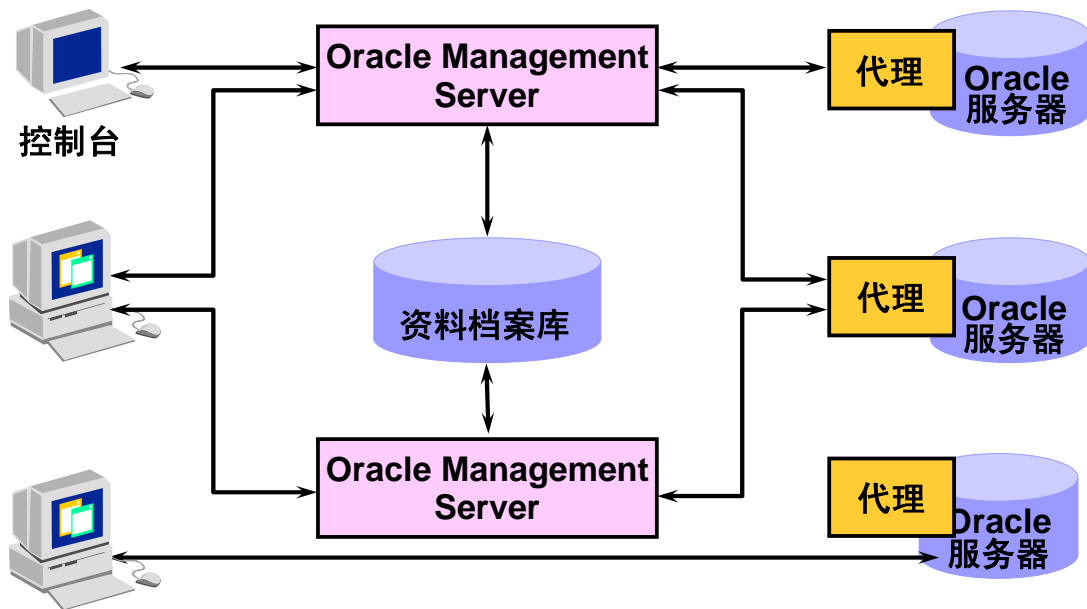
Copyright © Oracle Corporation, 2001. All rights reserved.

Oracle Enterprise Manager

Oracle Enterprise Manager 是一个统一的管理框架，由基于 Java 的控制台、一套工具和服务、一个管理服务器和智能代理的网络构成。它包括系统中对象及其关系的层次树和图形表示。公用服务，包括作业调度和管理、事件管理、数据库发现和管理、服务发现和管理构成了 Oracle Enterprise Manager 的完整框架。

此外，Oracle Enterprise Manager 还包括集成的应用程序，使用这些应用程序，可执行例行程序和高级管理任务。这些应用程序包括可选程序包，如：诊断包 (Diagnostics Pack)、优化包 (Tuning Pack) 和更改管理包 (Change Management Pack) 以及 Oracle Net Manager、空间索引审查程序 (Spatial Index Advisor) 和文本管理器 (Text Manager) 等其它应用程序。

Oracle Enterprise Manager — 体系结构



2-13

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

Oracle Enterprise Manager 体系结构

Oracle Enterprise Manager 使用三层体系结构，其中包括：

第一层：控制台客户机和集成工具，为管理员提供图形界面。

第二层：Oracle Management Servers 和数据库资料档案库，为处理系统管理任务提供可伸缩的中层结构。

第三层：安装在每个节点上的智能代理，监视 Oracle Enterprise Manager 服务并执行 Management Server 上的任务。

并非在所有情况都需要采用三层系统的 Oracle Enterprise Manager，Oracle Enterprise Manager 也可以仅有两层体系结构，该结构直接连接至数据库。使用独立启动的控制台，一个用户可以使用一个或多个应用程序，而无需使用 Oracle Management Server 或智能代理。

如果要执行无需使用作业、事件或组系统的基本管理任务，可以使用独立控制台。

Oracle Enterprise Manager 体系结构（续）

控制台：

这是第一层，由客户端应用程序组成，如控制台和管理应用程序，它们为管理员提供图形用户界面以执行所有管理任务。第一层依赖第二层的 Oracle Management Server 来处理其大量的应用程序逻辑。

注：自 Oracle9i 起，可以独立连接至控制台。对于 Oracle9i 以前的版本，所有与控制台的连接都需要通过 Oracle Management Server 进行。

Oracle Management Server：

Oracle Enterprise Manager 的第二层组件是 Oracle Management Server (OMS)。OMS 是 Enterprise Manager 框架的核心，它提供管理用户帐户、处理作业和事件等活动、管理控制台（第一层）和受管节点（第三层）之间的信息流。

OMS 使用资料档案库来存储所有系统数据、应用程序数据、有关受管节点的状态的信息以及有关所有系统管理的程序包的信息。

Oracle Enterprise Manager 资料档案库：

该资料档案库是一组表，是在设置 OMS 时创建的。OMS 使用资料档案库作为其永久的后端存储。如果必要，可使用多个 OMS。多个 OMS 共享一个资料档案库并提供可靠性和容错功能。

节点：

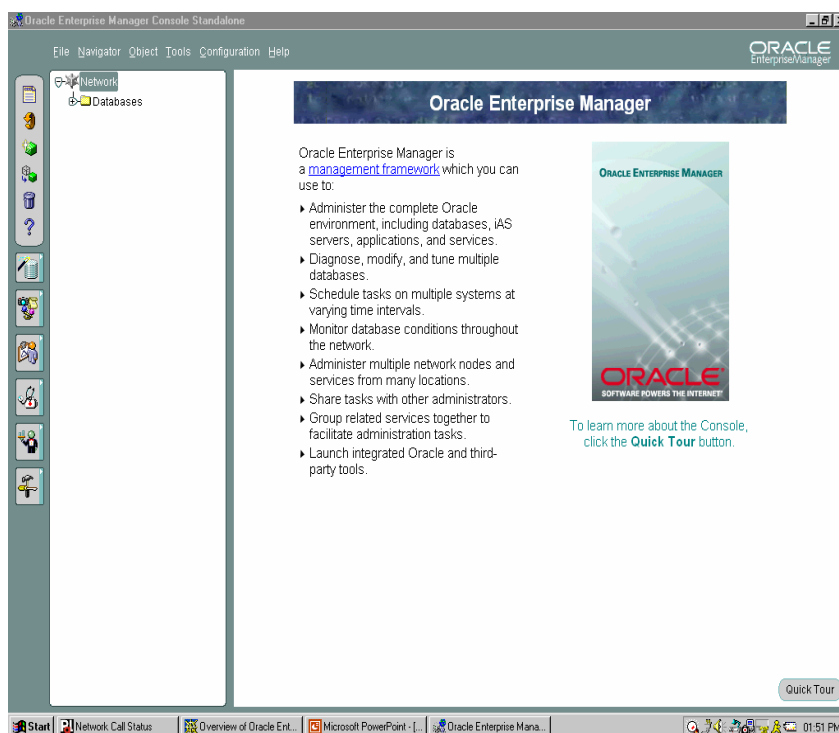
第三层由受管节点组成，其中包含数据库和其它受管服务等目标。每个节点上都驻留着一个 Oracle 智能代理，它与 OMS 进行通信并执行控制台和客户端应用程序所发送的任务。

每个节点只需要一个智能代理。

智能代理是独立于数据库、控制台以及 Management Server 运行的。由于它独立于其它组件运行，智能代理可以执行如下任务：启动和关闭数据库、在系统的另一部分关闭时仍保持运行状态。智能代理的 ID 是 db snmp。

控制台

- 中心启动点
- 可在瘦客户机或胖客户机上运行
- 可以独立启动，也可通过 OMS 启动



ORACLE

2-15

Copyright © Oracle Corporation, 2001. All rights reserved.

控制台特性

控制台为管理员提供图形界面，是所有管理应用程序和工具的中心启动点。另外，SQL*Plus Worksheet 也可以通过控制台启动。

控制台可通过 Web 以瘦客户机模式运行，也可以作为胖客户机运行。瘦客户机使用 Web 浏览器与安装有控制台文件的服务器相连接，而胖客户机需要在本地安装控制台文件。

控制台可独立启动，也可以通过连接到 Oracle Management Server 来启动。

注：本课程的目标并不在于为您提供有关 Oracle Enterprise Manager、控制台或 Oracle Management Server 的详细信息。有关使用 Oracle Enterprise Manager 的完整信息，请参考 *Oracle Enterprise Manager 9i* 课程。

如何启动 Oracle Enterprise Manager 控制台

示例：启动 Oracle Enterprise Manager 控制台。

1. 请按以下步骤启动控制台：
“开始” (Start) > “程序” (Programs) > Oracle-OraHome90 >
“Enterprise Manager 控制台”
2. 选择以下选项之一来启动控制台：
 - “登录到 Oracle Management Server” (Login to the Oracle Management Server)
 - “独立启动” (Launch standalone)
3. 单击“确定” (OK)。

小结

在这一课中，您应该能够掌握：

- 了解数据库管理工具
- 了解 Oracle Universal Installer 的特性
- 使用 SQL*Plus 操作和控制数据库
- 了解 Oracle Enterprise Manager 的主要组件

ORACLE

练习 2 概览

此练习涉及以下主题：

- 连接到 **SQL*Plus**
- 连接到 **Enterprise Manager 控制台**

ORACLE[®]

练习 2: Oracle 服务器入门

本练习需在教师指导下进行。教师将为您提供注册帐户并指导您登录到该帐户。
将教师提供的信息填写在以下位置。

主机名: _____

SID 名: _____

1 以 SYSDBA 用户身份连接到 SQL*Plus。

提示:

- 按照教师的指导连接到您的帐户
- 启动 SQL*Plus
- 以 SYS AS SYSDBA 用户身份连接

```
$ sqlplus /nolog
SQL*Plus: Release 9.0.1.0.0 - Production on Thu Nov 29
15:44:51 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
SQL> CONNECT / AS SYSDBA
Connected.
```

2 使用 SQL*Plus 运行以下查询，验证是否已连接到该数据库。

```
SQL> SELECT * FROM DUAL;
D
-
X
```

3 以独立模式启动 Oracle Enterprise Manager。

- 导航到“开始”(Start) > “程序”(Programs) > Oracle-OraHome90 > “Enterprise Manager 控制台”(Enterprise Manager Console)
- 选择“独立启动”(Launch standalone) 选项。
- 选择“确定”(OK)。

练习 2: Oracle 服务器入门

4 使用 Oracle Management 服务启动 Oracle Enterprise Manager。

- 仅用于 Oracle Classroom:

如果您处在 Oracle Classroom 环境中，则必须执行以下专用于 Oracle Classroom 启动的四个步骤：

1. 单击桌面上的 omsconfig 文件更新图标

输入您的班级使用的 Unix 服务器的名称。教师将为您提供该名称。请按原样输入。此名称是区分大小写的。

2. 打开一个 MSDOS 窗口。

3. 在命令行提示符下输入：oemctl start oms. 等待出现以下消息：

“The Oracleoracle901_homeManagementServer service
服务已成功启动。”

4. 关闭 MSDOS 窗口。

- 启动 OEM 控制台并选择“登录到 Oracle Management Server” (Login to the Oracle Management Server) 选项。使用以下用户名和口令登录：

管理员 (Administrator): sysman 注：区分大小写。

口令 (Password): oem_temp 注：区分大小写。

出现提示后，将口令更改为 oracle。注：区分大小写。

管理服务器 (Management Server): (由教师提供)

- 打开 OEM 控制台后，进入主菜单并选择以下菜单项：

“导航器” (Navigator) > “发现节点” (Discover Nodes)。出现“发现向导” (Discovery Wizard) 对话框。

- 选择“下一步” (Next) 以继续。

- 输入您要管理的节点的名称：即指定的数据库服务器主机名。（由教师提供）

- 选择“下一步” (Next)。

- 发现操作完成后，选择“确定” (OK)。注：如果发现操作未成功，请向教师报告。

- 展开导航树中的“数据库” (Database) 文件夹。

- 双击教师提供的目标数据库。

(转至下页)

练习 2: Oracle 服务器入门

4 使用 Oracle Management 服务启动 Oracle Enterprise Manager。 (续)

- 提供以下连接信息:
 - 用户 (User): (由教师提供)
 - 口令 (Password): (由教师提供)
 - 身份 (As): **SYSDBA**
- 接着设置运行作业的节点身份证明。
- 从主菜单选择以下菜单项: “配置” (Configuration) > “首选项” (Preferences)。
- 选择 “首选身份证明” (Preferred Credentials) 页。
- 滚动到窗口底部并选择指定数据库的条目。
- 提供下列信息:
 - 用户名 (Username): (由教师提供)
 - 口令 (Password): (由教师提供)
 - 确认口令 (Confirm Password):
 - 角色 (Role): **SYSDBA**
- 选择 “确定” (OK)。

5 启动 SQL*Plus Worksheet

SQL*Plus Worksheet 可通过执行以下操作从 Oracle Enterprise Manager 控制台启动:

- 导航到 “工具” (Tools) > “数据库应用程序” (Database Applications) > SQL*Plus Worksheet

SQL*Plus Worksheet 还可通过执行以下操作从 Windows NT 菜单启动:

- 导航到 “开始” (Start) > “程序” (Programs) > Oracle-OraHome90 > “集成管理工具” (Integrated Management Tools) > SQLPlus Worksheet
 - 直接连接到教师定义的数据库。
 - 输入: 用户名 (Username)、口令 (Password) 和服务 (Service)
 - 连接身份 (Connect as): **SYSDBA**
 - 单击 “确定” (OK)。

注: 每次以另一用户身份登录 (在 SQL*Plus Worksheet 内) 时, 必须在连接字符串中包括服务名。

3

管理 Oracle 例程

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

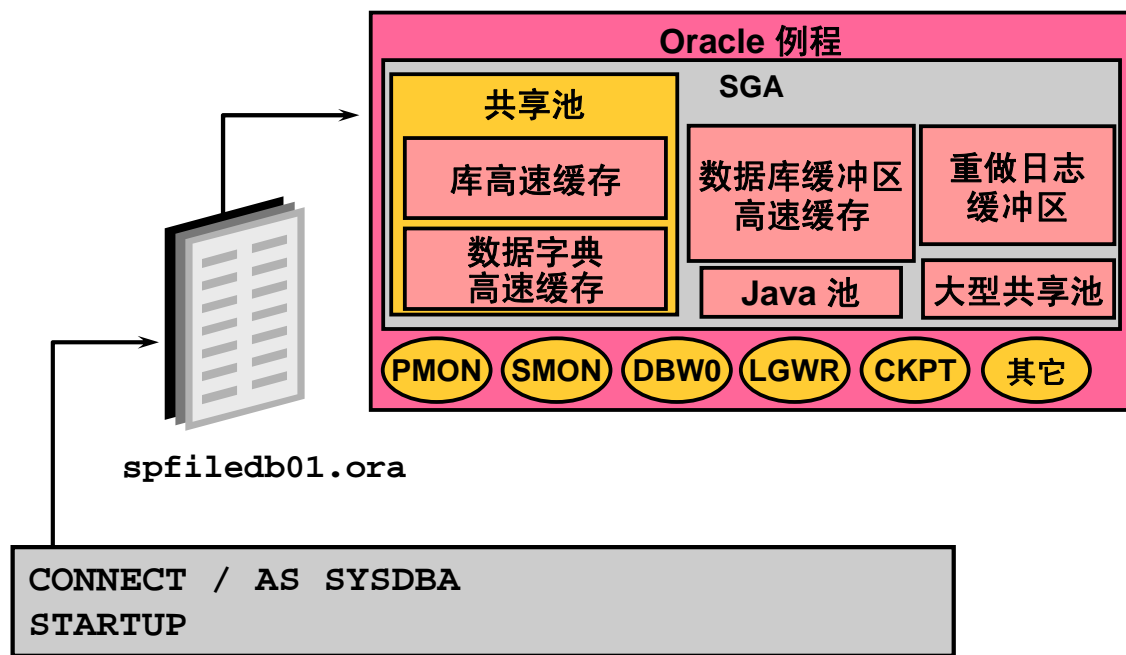
目标

完成这一课的学习后，您应该能：

- 创建和管理初始化参数文件
- 启动和关闭例程
- 监视和使用诊断文件

ORACLE

初始化参数文件



ORACLE

3-3

Copyright © Oracle Corporation, 2001. All rights reserved.

初始化参数文件

要启动一个例程，Oracle 服务器必须读取初始化参数文件。

初始化参数文件

- 文件中的条目专用于要启动的例程
- 有两种类型的参数：
 - 显式：文件中有一个条目
 - 隐式：文件中没有条目，但假定取 Oracle 缺省值
- 可存在多个初始化参数文件
- 对文件中条目的更改的生效时间，取决于使用的初始化参数文件类型
 - 静态参数文件 PFILE
 - 永久参数文件 SPFILE

ORACLE

3-4

Copyright © Oracle Corporation, 2001. All rights reserved.

初始化参数文件

Oracle 服务器在启动例程时读取初始化参数文件。共有两种类型的初始化参数文件：

- 静态参数文件 PFILE，一般名为 initSID.ora。
- 永久参数文件 SPFILE，一般名为 spfileSID.ora。

初始化参数文件内容：

- 例程参数列表
- 与该例程相关联的数据库的名称
- 系统全局区 (SGA) 的内存结构的分配
- 如何处理已满的联机重做日志文件
- 控制文件的名称和位置
- 有关撤消段的信息

为在各种不同情况下优化性能，一个例程可有多多个初始化参数文件。

初始化参数文件

使用 Oracle Enterprise Manager 查看初始化参数

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “例程”(Instance) > “配置”(Configuration)。
2. 从“常规”(General) 页选择“全部初始化参数”(All Initialization Parameters)。

PFILE

initSID.ora

- 文本文件
- 使用操作系统编辑器进行修改
- 手动进行修改
- 所作更改在下次启动时生效
- 仅在例程启动过程中打开
- 缺省位置为 `$ORACLE_HOME/dbs`

ORACLE

3-6

Copyright © Oracle Corporation, 2001. All rights reserved.

PFILE

PFILE 是可使用标准的操作系统编辑器进行维护的文本文件。PFILE 在例程启动过程中是只读的。如果文件发生修改，则必须关闭然后重新启动例程以使新的参数值生效。缺省情况下，该文件位于 `$ORACLE_HOME/dbs` 目录中，文件名是 `initSID.ora`。

创建 PFILE

- 使用样本 `init.ora` 文件创建
 - 样本文件由 Oracle Universal Installer 安装
 - 使用操作系统复制命令复制样本
 - 由数据库 SID 唯一标识

```
cp init.ora $ORACLE_HOME/dbs/initdba01.ora
```

- 修改 `initSID.ora`
 - 编辑参数
 - 针对数据库要求

ORACLE

3-7

Copyright © Oracle Corporation, 2001. All rights reserved.

创建 PFILE

样本 `init.ora` 文件由 Universal Installer 在安装过程中创建。该样本 `init.ora` 文件可用于创建特定于某一例程的 `initSID.ora`。可使用文本编辑器修改 `initSID.ora` 文件中的参数。

PFILE 示例

```
# Initialization Parameter File: initdba01.ora
db_name              = dba01
instance_name        = dba01
control_files         = (
    home/dba01/ORADATA/u01/control01dba01.ctl,
    home/dba01/ORADATA/u02/control01dba02.ctl)
db_block_size         = 4096
db_cache_size         = 4M
shared_pool_size      = 50000000
java_pool_size        = 50000000
max_dump_file_size    = 10240
background_dump_dest  = /home/dba01/ADMIN/BDUMP
user_dump_dest        = /home/dba01/ADMIN/UDUMP
core_dump_dest        = /home/dba01/ADMIN/CDUMP
undo_management       = AUTO
undo_tablespace       = UNDOTBS
. . .
```

ORACLE

3-8

Copyright © Oracle Corporation, 2001. All rights reserved.

PFILE 示例

- 以这样的格式指定值：keyword=value（关键字 = 值）。
- 服务器为每个参数都设置了缺省值。根据参数的不同，缺省值可能与操作系统相关。
- 可以按任意顺序指定参数，但也存在例外。
- 注释行以 # 符号开头。
- 参数中如果包括字符文字，可将参数用双引号括起。
- 可以使用关键字 IFILE 使参数中包括其它文件。
- 如果使用的操作系统区分大小写，那么文件名也区分大小写。
- 如果有多个值，应该用圆括号将它们括起来，用逗号隔开。

注：请为参数的列出顺序指定一个标准：按字母顺序列出或按功能进行分组。PFILE 根据例程的不同而变化，不一定与上例相同。

SPFILE spfileSID.ora

- 二进制文件
- 由 Oracle 服务器进行维护
- 始终驻留在服务器端
- 所做更改永久有效，不受关闭和启动的影响
- 可以自行调节参数值
- 使恢复管理器能够备份初始化参数文件

ORACLE

3-9

Copyright © Oracle Corporation, 2001. All rights reserved.

SPFILE

SPFILE 是 Oracle9i 中新增的二进制文件。该文件不能手动修改，且必须始终驻留在服务器端。创建该文件后，即由 Oracle 服务器进行维护。如果进行手动修改，SPFILE 将无效。SPFILE 具有对数据库进行永久更改的功能，不受关闭和启动操作的影响，它还提供自动调节记录在文件中的参数值的功能。使用 SPFILE，RMAN 可以支持初始化参数文件的备份，因为 SPFILE 驻留在服务器端。缺省情况下，它位于 \$ORACLE_HOME/dbs 目录中，缺省名称为 spfileSID.ora。

创建 SPFILE

- 从 PFILE 文件创建

```
CREATE SPFILE = '$ORACLE_HOME/dbs/spfileDBA01.ora'  
FROM PFILE = '$ORACLE_HOME/dbs/initDBA01.ora';
```

其中

- SPFILE-NAME: 要创建的 SPFILE
- PFILE-NAME: 用于创建 SPFILE 的 PFILE

- 可在例程启动之前或之后执行

ORACLE

3-10

Copyright © Oracle Corporation, 2001. All rights reserved.

创建 SPFILE

SPFILE 是使用 CREATE SPFILE 命令从 PFILE 文件创建的。该命令需要具有 SYSDBA 权限才能执行。该命令可在例程启动之前或之后执行。

```
SQL> CREATE SPFILE [= 'SPFILE-NAME']  
2 FROM PFILE [= 'PFILE-NAME']
```

其中:

- SPFILE-NAME: 要创建的 SPFILE 的名称
- PFILE-NAME: 用于创建 SPFILE 的 PFILE 的名称。PFILE 必须在服务器端可用

如果在语法中未包括 SPFILE-NAME 和 PFILE-NAME, Oracle 将使用缺省 PFILE 来生成 SPFILE (其名称由系统生成)。

```
SQL> CREATE SPFILE FROM PFILE;
```


创建 SPFILE (续)

导出 SPFILE:

可将 SPFILE 的内容导出到 PFILE 中。

```
SQL> CREATE PFILE FROM SPFILE;
```

以上命令在服务器端创建了一个文本文件格式的 PFILE。该命令可在例程启动之前或之后执行。这样就提供了一种查看 SPFILE 并进行修改的简单方法:

- 将 SPFILE 导出到 PFILE
- 编辑 PFILE
- 从编辑过的 PFILE 重新创建 SPFILE

将 SPFILE 导出到 PFILE 还可用作创建永久参数文件的备份的备用方法。

注: 使用 Oracle9i, RMAN 还可备份永久参数文件。

V\$SPPARAMETER

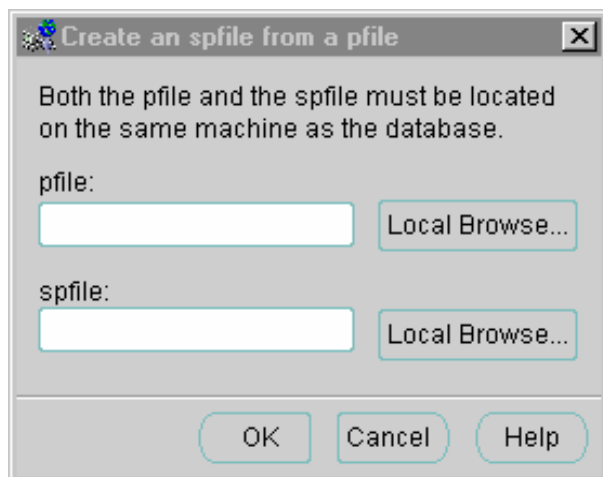
如上所述, 查看 SPFILE 内的参数设置时有几个选项。V\$SPPARAMETER 是显示和查看 SPFILE 的内容的另一种方法。

创建 SPFILE

使用 Oracle Enterprise Manager 创建 SPFILE

从 OEM 控制台：

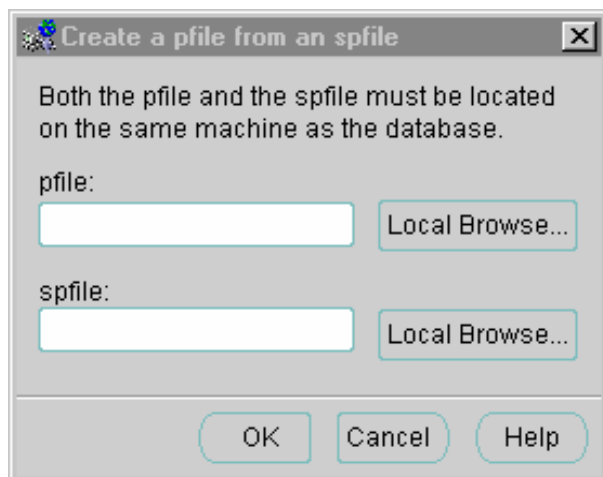
1. 从主菜单选择“对象” (Object) > “创建 spfile” (Create spfile)。



使用 Oracle Enterprise Manager 导出 SPFILE

从 OEM 控制台：

1. 从主菜单选择“对象” (Object) > “创建 pfile” (Create pfile)。



SPFILE 示例

```
*.background_dump_dest='/home/dba01/ADMIN/BDUMP'  
*.compatible='9.0.0'  
*.control_files='/home/dba01/ORADATA/u01/ctrl01.ctl'  
*.core_dump_dest='/home/dba01/ADMIN/CDUMP'  
*.db_block_size=4096  
*.db_name='dba01'  
*.db_domain='world'  
*.global_names=TRUE  
*.instance_name='dba01'  
*.remote_login_passwordfile='exclusive'  
*.java_pool_size=50000000'  
*.shared_pool_size=50000000  
*.undo_management='AUTO'  
*.undo_tablespace='UNDOTBS'  
. . .
```

ORACLE

3-13

Copyright © Oracle Corporation, 2001. All rights reserved.

SPFILE 示例

PFILE 中的参数设置行上指定的注释保留在 SPFILE 中。所有其它注释都被忽略。尽管 SPFILE 中的文本在 UNIX 中易于查看，但 SPFILE 是一个二进制文件，对 SPFILE 进行手动修改将使之无效。如果需要查看 SPFILE 的特定内容或进行一些更改，可将 SPFILE 导出到 PFILE。

STARTUP 命令行为

- 优先顺序

- spfileSID.ora
- 缺省 SPFILE
- initSID.ora
- 缺省 PFILE

- 指定的 PFILE 可覆盖优先顺序

```
STARTUP PFILE = $ORACLE_HOME/dbs/initDBA1.ora
```

- PFILE 可指示要使用 SPFILE

```
SPFILE = /database/startup/spfileDBA1.ora
```

ORACLE

3-14

Copyright © Oracle Corporation, 2001. All rights reserved.

STARTUP 命令行为

优先顺序:

- 使用命令 STARTUP 时, 服务器端的 spfileSID.ora 用于启动例程。
- 如果找不到 spfileSID.ora, 则使用服务器端的缺省 SPFILE 来启动例程。
- 如果找不到缺省 SPFILE, 将使用服务器端的 initSID.ora 来启动例程。

指定的 PFILE 可覆盖缺省 SPFILE 来启动例程。

可在 PFILE 中包含一个定义以指示要使用 SPFILE。这是在非缺省位置使用 SPFILE 启动例程的唯一方法。要使用非缺省位置的 SPFILE 启动数据库, 必须在 PFILE 中指定 SPFILE=<完整路径和文件名>。

示例: SPFILE=\$HOME/ADMIN/PFILE/\$ORACLE_SID.ora。

修改 SPFILE 中的参数

- 使用 ALTER SYSTEM 更改参数值

```
ALTER SYSTEM SET undo_tablespace = 'UNDO2';
```

- 指定所做更改是临时的还是永久的

```
ALTER SYSTEM SET undo_tablespace = 'UNDO2'  
SCOPE=BOTH;
```

- 删除或重置值

```
ALTER SYSTEM RESET undo_suppress_errors  
SCOPE=BOTH SID='*';
```

ORACLE

3-15

Copyright © Oracle Corporation, 2001. All rights reserved.

修改 SPFILE 中的参数

ALTER SYSTEM SET 命令用于更改例程参数的值。

```
ALTER SYSTEM SET parameter_name = parameter_value  
[COMMENT 'text'] [SCOPE = MEMORY|SPFILE|BOTH]  
[SID= 'sid'|'*']
```

其中

parameter_name: 要更改的参数名称

parameter_value: 要将参数更改为的值

COMMENT: 添加在 SPFILE 中被更改的参数旁的注释

SCOPE: 确定应在内存中、在 SPFILE 中还是同时在这两个位置进行更改

MEMORY: 只能在当前运行的例程中更改参数值

SPFILE: 只能在 SPFILE 中更改参数值

BOTH: 在当前运行的例程和 SPFILE 中均可更改参数值

SID: 标识要使用的 SPFILE 的 ORACLE_SID

'sid': 更改 SPFILE 时使用的特定 SID

'*': 使用缺省 SPFILE

修改 SPFILE 中的参数（续）

示例：

```
SQL> SHOW PARAMETERS undo_suppress_errors
```

NAME	TYPE	VALUE
-----	-----	-----
undo_suppress_errors	boolean	FALSE

```
SQL> ALTER SYSTEM SET undo_suppress_errors = TRUE
      2 COMMENT = 'temporary testing' SCOPE=BOTH
      3 SID='DBA01';
```

```
SQL> SHOW PARAMETERS undo_suppress_errors
```

NAME	TYPE	VALUE
-----	-----	-----
undo_suppress_errors	boolean	TRUE

ALTER SYSTEM RESET 命令用于删除或还原为缺省值。

```
SQL> ALTER SYSTEM RESET parameter_name [SCOPE =
      MEMORY|SPFILE|BOTH] [SID= 'sid' | '*' ]
```

示例：

```
SQL> ALTER SYSTEM RESET undo_suppress_errors
      2 SCOPE=BOTH SID='dba01';
```

从 SPFILE 中删除一个参数有以下几种方法：

- 将参数重设为缺省值来模拟使用 ALTER SYSTEM SET 的删除操作。
- 使用 CREATE SPFILE FROM PFILE 重新创建 SPFILE。
- 使用 ALTER SYSTEM RESET 从 SPFILE 删除参数。

修改 SPFILE 中的参数（续）

使用 Oracle Enterprise Manager 修改 SPFILE 配置

从 OEM 控制台：

1. 导航到“数据库” (Databases) > “例程” (Instance)。
2. 单击“配置” (Configuration)。
3. 在“常规” (General) 页上，单击“全部初始化参数” (All Initialization Parameters)。
4. 在参数值栏中修改参数。
5. 单击“确定” (OK)。

应在初始化参数文件中指定的参数

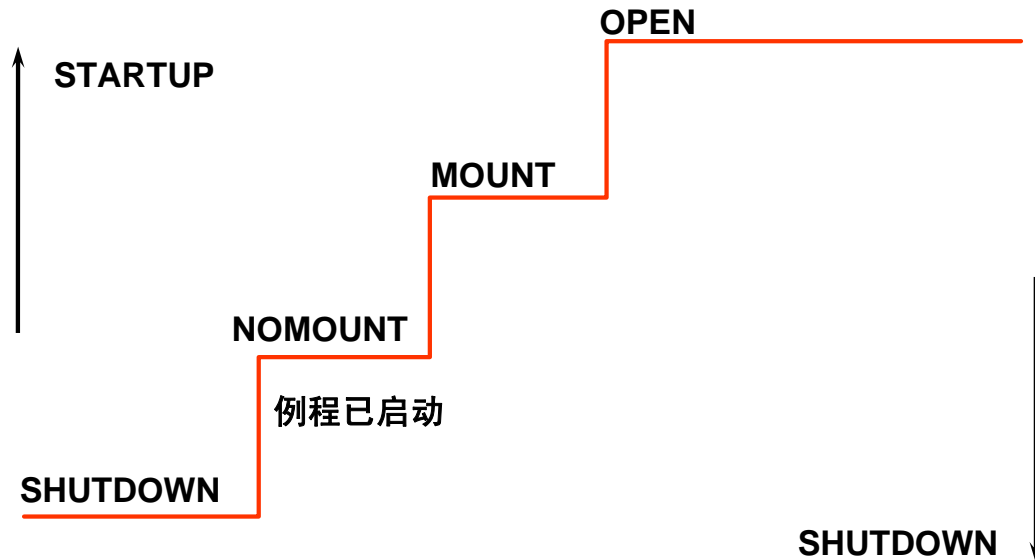
参数	说明
BACKGROUND_DUMP_DEST	写入后台进程跟踪文件的位置（LGWR、DBWn 等）。这也是存储警报日志文件的位置
COMPATIBLE	该例程应与之兼容的服务器版本
CONTROL_FILES	控制文件的名称
DB_CACHE_SIZE	指定标准块大小缓冲区的高速缓存大小
DB_NAME	由八个或更少字符组成的数据库标识符。这是创建新数据库时要求设置的唯一参数
SHARED_POOL_SIZE	共享池的大小（以字节为单位）
USER_DUMP_DEST	代表用户进程创建用户调试跟踪文件的位置

注：缺省值取决于 Oracle 服务器的版本。

常修改的参数

参数	说明
IFILE	在当前参数文件内嵌入的另一个参数文件的名称。最多可以有三级嵌套
LOG_BUFFER	分配给 SGA 中重做日志缓冲区的字节数
MAX_DUMP_FILE_SIZE	以操作系统块数指定的跟踪文件的最大大小
PROCESSES	能够同时连接该例程的最大操作系统进程数
SQL_TRACE	启用或禁用每个用户会话的 SQL 跟踪设备
TIMED_STATISTICS	启用或禁用跟踪文件和监视器屏幕中的计时

启动数据库 NOMOUNT



ORACLE

3-19

Copyright © Oracle Corporation, 2001. All rights reserved.

启动数据库

启动数据库时，选择启动状态。下面将说明启动例程的各阶段的情况。

启动例程 (NOMOUNT)：

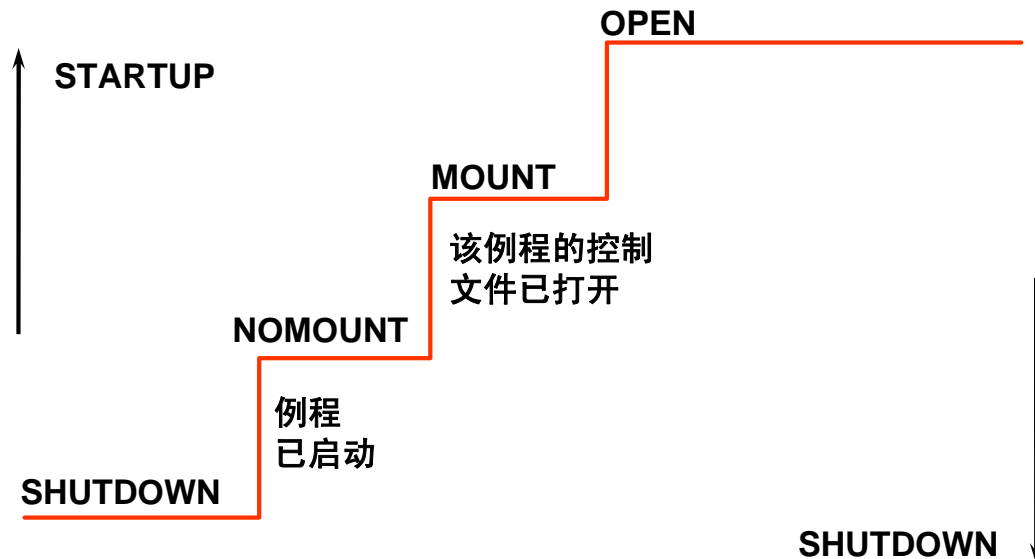
仅在创建数据库或重新创建控制文件过程中，例程才会在 NOMOUNT 阶段启动。

启动例程包括下列任务：

- 按以下顺序从 \$ORACLE_HOME/dbs 读取初始化文件：
 - 首先读取 spfileSID.ora
 - 如果找不到，则读取 spfile.ora
 - 如果仍然找不到，则读取 initSID.ora使用 STARTUP 指定 PFILE 参数以覆盖缺省行为。
- 分配 SGA
- 启动后台进程
- 打开 alertSID.log 文件和跟踪文件

必须在初始化文件中使用 DB_NAME 参数对数据库命名，或使用 STARTUP 命令命名。

启动数据库 MOUNT



ORACLE

3-20

Copyright © Oracle Corporation, 2001. All rights reserved.

启动数据库

加载数据库 (MOUNT)：

若要执行特定的维护操作，可启动例程并加载数据库，但不要打开数据库。

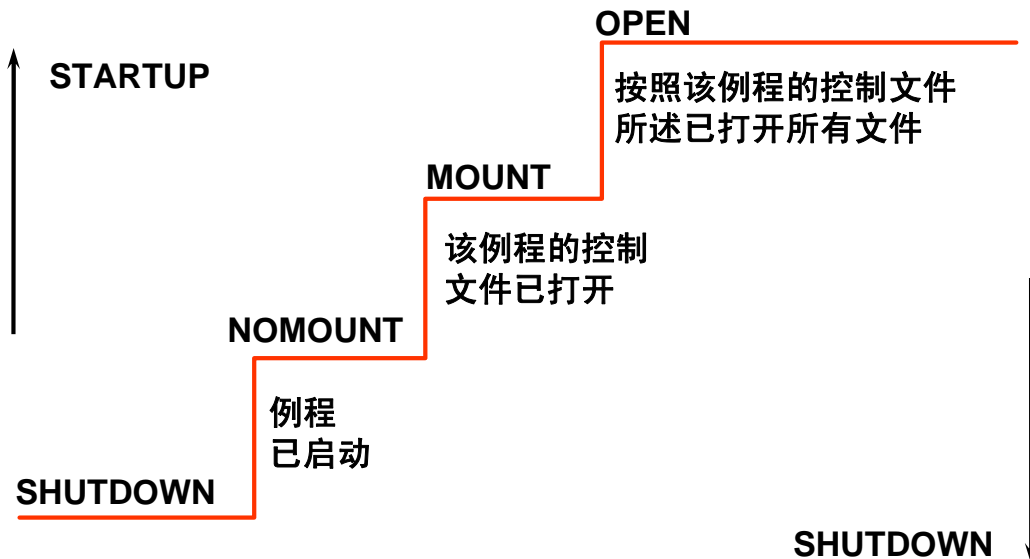
例如，在以下任务中必须加载数据库但不要打开数据库：

- 重命名数据文件
- 启用和禁用重做日志归档选项
- 执行完全数据库恢复

加载数据库包括以下任务：

- 使数据库与以前启动的例程关联
- 定位并打开参数文件中指定的控制文件
- 读取控制文件以获取数据文件和重做日志文件的名称和状态。但是，在此时不进行数据文件和联机重做日志文件是否存在的检查。

启动数据库 OPEN



ORACLE

3-21

Copyright © Oracle Corporation, 2001. All rights reserved.

启动数据库

打开数据库 (OPEN):

正常的数据库操作指启动例程、加载数据库和打开数据库。通过正常的数据库操作，任何有效用户都可以连接到数据库并执行一般的数据访问操作。

打开数据库包括以下任务：

- 打开联机数据文件
- 打开联机重做日志文件

如果在尝试打开数据库时有任何数据文件或联机重做日志文件不存在，Oracle 服务器将返回错误消息。

在这个最后阶段中，Oracle 服务器验证所有数据文件和联机重做日志文件是否可以打开，并检查数据库的一致性。如果需要，系统监视 (SMON) 后台进程将启动例程恢复操作。

STARTUP 命令

启动例程并打开数据库：

```
STARTUP
```

```
STARTUP PFILE=$ORACLE_HOME/dbs/initdb01.ora
```

ORACLE

3-22

Copyright © Oracle Corporation, 2001. All rights reserved.

STARTUP 命令

若要启动例程，请使用以下命令：

```
STARTUP [FORCE] [RESTRICT] [PFILE=filename]
        [OPEN [RECOVER][database]
        | MOUNT
        | NOMOUNT]
```

（注：这不是完整的语法。）

其中：

- OPEN：使用户能够访问数据库
- MOUNT：为某些 DBA 活动加载数据库，但不允许用户访问数据库
- NOMOUNT：创建 SGA 并启动后台进程，但不允许访问数据库
- PFILE=parfile：允许使用非缺省参数文件配置例程

启动（续）

- **FORCE:** 执行正常启动之前终止运行的例程。
- **RESTRICT:** 只允许具有 **RESTRICTED SESSION** 权限的用户访问数据库。
- **RECOVER:** 在数据库启动时开始进行介质恢复。

自动启动数据库:

在 UNIX 上:

自动启动和关闭数据库可由特定的操作系统文件进行控制, 例如, `/var/opt/oracle` 目录下的 `oratab`。

注: 有关详细信息, 请参考您所用操作系统的安装指南。

故障排除:

如果在发出 **STARTUP** 命令时出现错误, 则在再次发出 **STARTUP** 命令前必须先发出 **SHUTDOWN** 命令。

注: **STARTUP** 和 **SHUTDOWN** 命令都是 **SQL*Plus** 命令, 而不是 **SQL** 命令。

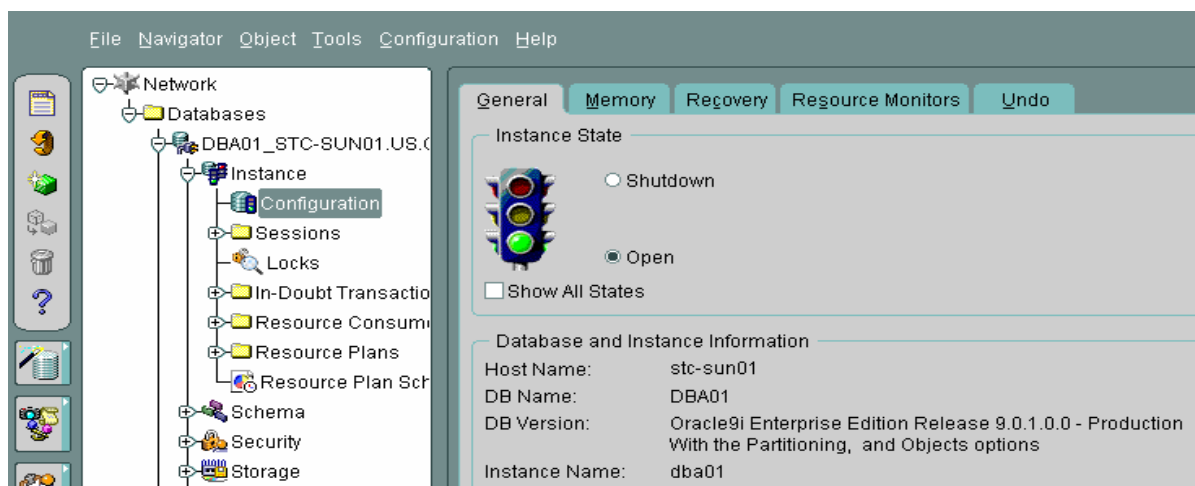
关闭选项（续）

使用 Oracle Enterprise Manager 启动数据库

从 OEM 控制台：

1. 导航到“数据库” (Databases) > “例程” (Instance)
2. 单击“配置” (Configuration)
3. 从“常规” (General) 选项卡，选择“打开” (Open) 选项。
4. 单击“应用” (Apply)。

注：必须以 SYSDBA 权限连接到数据库才能执行启动操作。



ALTER DATABASE 命令

- 将数据库状态从 NOMOUNT 更改为 MOUNT:

```
ALTER DATABASE db01 MOUNT;
```

- 将数据库作为只读数据库打开:

```
ALTER DATABASE db01 OPEN READ ONLY;
```

ORACLE

3-25

Copyright © Oracle Corporation, 2001. All rights reserved.

ALTER DATABASE 命令

要将数据库从 NOMOUNT 更改为 MOUNT 阶段或者从 MOUNT 更改为 OPEN 阶段，请使用 ALTER DATABASE 命令：

```
ALTER DATABASE { MOUNT | OPEN }
```

若要防止数据被用户事务修改，可以以只读模式打开数据库。

若要启动例程，请使用以下命令：

```
ALTER DATABASE OPEN [READ WRITE | READ ONLY]
```

其中：

- READ WRITE：以读写模式打开数据库，以便用户生成重做日志。
- READ ONLY：将用户限制为只能执行只读事务，防止用户生成重做日志信息。

以受限模式打开数据库

- 使用 **STARTUP** 命令限制对数据库的访问：

```
STARTUP RESTRICT
```

- 使用 **ALTER SYSTEM** 命令将例程置于受限模式：

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

ORACLE

3-26

Copyright © Oracle Corporation, 2001. All rights reserved.

以受限模式打开数据库

受限会话十分有用，例如，当执行结构维护或数据库导出和导入时。以受限模式启动数据库，这样，只有具有 **RESTRICTED SESSION** 权限的用户才可以使用该数据库。使用 **ALTER SYSTEM SQL** 命令也可以将数据库设为受限模式：

```
ALTER SYSTEM [ {ENABLE|DISABLE} RESTRICTED SESSION ]
```

其中：

- **ENABLE RESTRICTED SESSION**：仅允许具有 **RESTRICTED SESSION** 权限的用户在以后登录
- **DISABLE RESTRICTED SESSION**：禁用 **RESTRICTED SESSION** 以允许没有该权限的用户可以登录

终止会话：

将例程置于受限模式后，在执行管理任务前可能想终止所有当前用户会话。此操作可通过以下命令来实现：

```
ALTER SYSTEM KILL SESSION 'integer1,integer2'
```

其中：

- **integer1**：V\$SESSION 视图中的 **SID** 列的值
- **integer2**：V\$SESSION 视图中的 **SERIAL#** 列的值

以受限模式打开数据库（续）

注：会话 ID 和序列号用来唯一地标识会话。这样，即使用户注销身份并且新会话使用相同的会话 ID，也可确保 ALTER SYSTEM KILL SESSION 命令能够应用于正确的会话。

终止会话的影响：

ALTER SYSTEM KILL SESSION 命令一执行，将使后台进程 PMON 立即执行以下步骤：

- 回退用户的当前事务
- 释放所有当前持有的表或行锁定
- 释放用户当前保留的所有资源

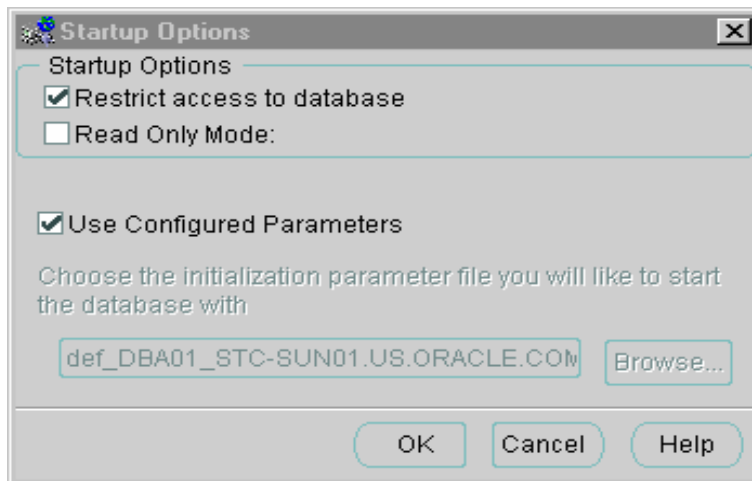
关闭选项（续）

使用 Oracle Enterprise Manager 以受限模式打开数据库

从 OEM 控制台：

1. 导航到“例程” (Instance) > “配置” (Configuration)。
2. 选择“常规” (General) 页。
3. 在“例程状态” (Instance State), 选择“关闭” (Shutdown) 选项。
4. 选择“应用” (Apply)。
5. 出现“关闭选项” (Shutdown Options) 对话框。选择“立即” (Immediate) 选项。
6. 选择“确定” (OK)。
7. 处理完成后，选择“关闭” (Close)。
8. 在“例程状态” (Instance State) 下，选择“打开” (Open) 选项。
9. 选择“确定” (OK)。
10. 出现“启动选项” (Startup Options) 对话框。选择“只限访问数据库” (Restrict access to database) 选项。
11. 选择“确定” (OK)。
12. 处理完成后，单击“关闭” (Close)。

注：您必须以 SYSDBA 权限连接到数据库。



以只读模式打开数据库

- 以只读模式打开数据库

```
STARTUP MOUNT  
ALTER DATABASE OPEN READ ONLY;
```

- 此模式可用于：
 - 执行查询
 - 使用本地管理的表空间执行磁盘排序
 - 使数据文件（而不是表空间）脱机和联机
 - 执行脱机数据文件和表空间的恢复

ORACLE

3-29

Copyright © Oracle Corporation, 2001. All rights reserved.

以只读模式打开数据库

只要数据库尚未以读写模式打开，就能以只读模式打开。该功能对于备用数据库从生产数据库卸载查询处理尤其有用。

如果查询需要使用临时表空间（例如，进行磁盘排序），当前用户必须将本地管理的表空间分配为缺省的临时表空间；否则查询会失败。对于用户 **SYS**，需要有本地管理的表空间。

注：后面的课程将讨论本地管理的表空间。

只读模式不限制无需生成重做数据即可更改数据库状态的数据库恢复或操作。例如，在只读模式中：

- 可使数据文件脱机和联机。
- 可以执行脱机数据文件和表空间的恢复。

磁盘在写入其它文件，如控制文件、操作系统审计线索、跟踪文件和警报日志文件时，可仍然保持只读模式。

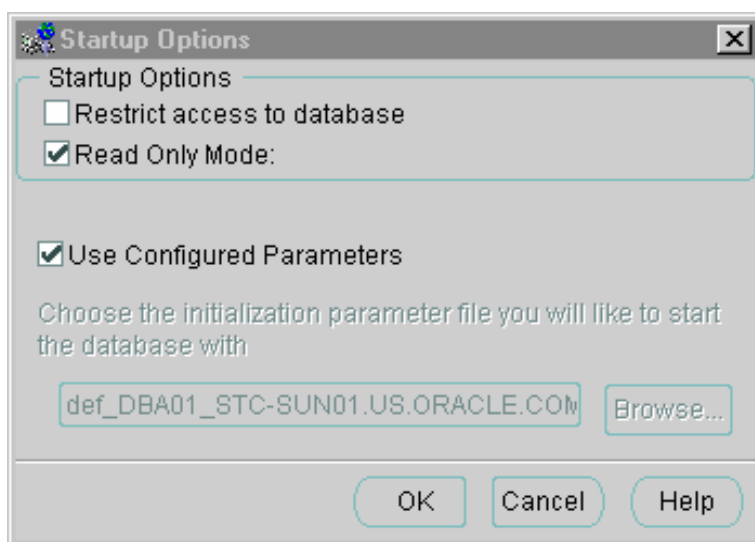
关闭选项（续）

使用 Oracle Enterprise Manager 以只读模式启动数据库

从 OEM 控制台：

1. 导航到“例程” (Instance) > “配置” (Configuration)。
2. 选择“常规” (General) 页。
3. 在“例程状态” (Instance State) 下，选择“关闭” (Shutdown) 选项。
4. 选择“应用” (Apply)。
5. 出现“关闭选项” (Shutdown Options) 对话框。选择“立即” (Immediate) 选项。
6. 选择“确定” (OK)。
7. 处理完成后，选择“关闭” (Close)。
8. 在“例程状态” (Instance State) 下，选择“打开” (Open) 选项。
9. 选择“确定” (OK)。
10. 出现“启动选项” (Startup Options) 对话框。选择“只读模式” (Read Only Mode) 选项。
11. 选择“确定” (OK)。
12. 处理完成后，单击“关闭” (Close)。

注：您必须以 SYSDBA 权限连接到数据库。



关闭数据库

关闭模式	A	I	T	N
允许建立新连接	否	否	否	否
等待到当前会话结束	否	否	否	是
等待到当前事务处理结束	否	否	是	是
强制执行检查点操作并关闭文件	否	是	是	是

关闭模式：

- **A = ABORT**
- **I = IMMEDIATE**
- **T = TRANSACTIONAL**
- **N = NORMAL**

ORACLE

3-31

Copyright © Oracle Corporation, 2001. All rights reserved.

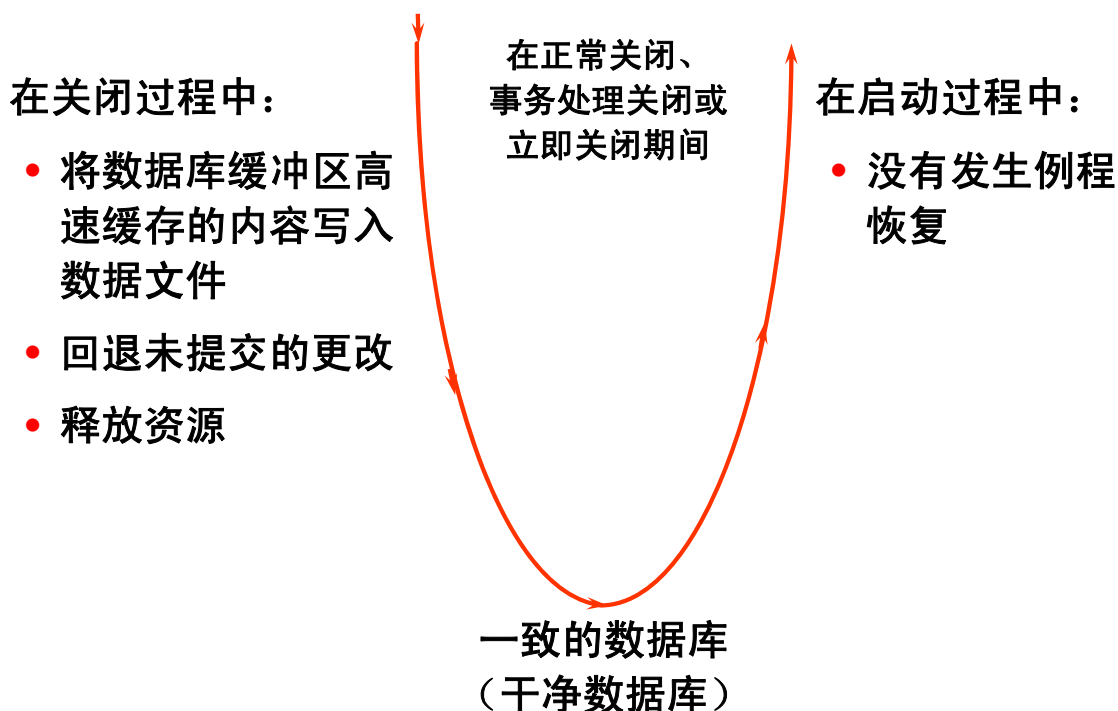
关闭数据库

关闭数据库这一操作将导致对所有物理结构进行操作系统脱机备份，并使修改过的静态初始化参数在重新启动后生效。

要关闭例程，必须使用以下命令以 SYSOPER 或 SYSDBA 身份进行连接：

SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT]

关闭选项



ORACLE

3-32

Copyright © Oracle Corporation, 2001. All rights reserved.

关闭选项

正常关闭：

“正常”是缺省的关闭模式。正常的数据库关闭在下列情况下进行：

- 未建立新连接。
- Oracle 服务器等待所有用户断开后才完成关闭。
- 数据库和重做缓冲区中的内容已写入磁盘。
- 后台进程已终止，SGA 已从内存中删除。
- Oracle 在关闭例程前将关闭并卸装数据库。
- 下一次启动将不要求例程恢复。

事务处理关闭：

事务处理关闭防止客户机丢失工作。事务处理数据库关闭在下列情况下进行：

- 没有客户机可以在此特定例程上启动新事务。
- 当客户机结束正在进行的事务时，断开客户机。
- 当所有事务都已完成后立即关闭。
- 下一次启动将不要求例程恢复。

关闭选项（续）

立即关闭：

立即关闭数据库在下列情况下进行：

- 由 Oracle 处理的当前 SQL 语句未完成。
- Oracle 服务器不等待当前连接到数据库的用户断开。
- Oracle 回退活动的事务并断开所有连接的用户。
- Oracle 在关闭例程前将关闭并卸装数据库。
- 下一次启动将不要求例程恢复。

关闭选项

在关闭过程中:

- 发生修改的缓冲区内容不写入数据文件
- 不回退未提交的更改

在关闭中止、
例程失败或
强制启动期间

在启动过程中:

- 使用重做日志重新应用更改
- 使用撤消段回退未提交的更改
- 释放资源

不一致的数据库
(灰数据库)

ORACLE

3-34

Copyright © Oracle Corporation, 2001. All rights reserved.

关闭选项

关闭中止:

如果“正常”和“立即关闭”选项不起作用，可以中止当前数据库例程。中止例程在下列情况下进行:

- Oracle 服务器所处理的当前 SQL 语句被立即终止。
- Oracle 不等待当前连接数据库的用户断开。
- 数据库和重做缓冲区中的内容不写入磁盘。
- 未提交的事务不回退。
- 在不关闭文件的情况下例程被终止。
- 数据库不关闭或被卸装。
- 下次启动要求恢复例程，该操作将自动进行。

注: 建议您不要备份处于不一致状态的数据库。

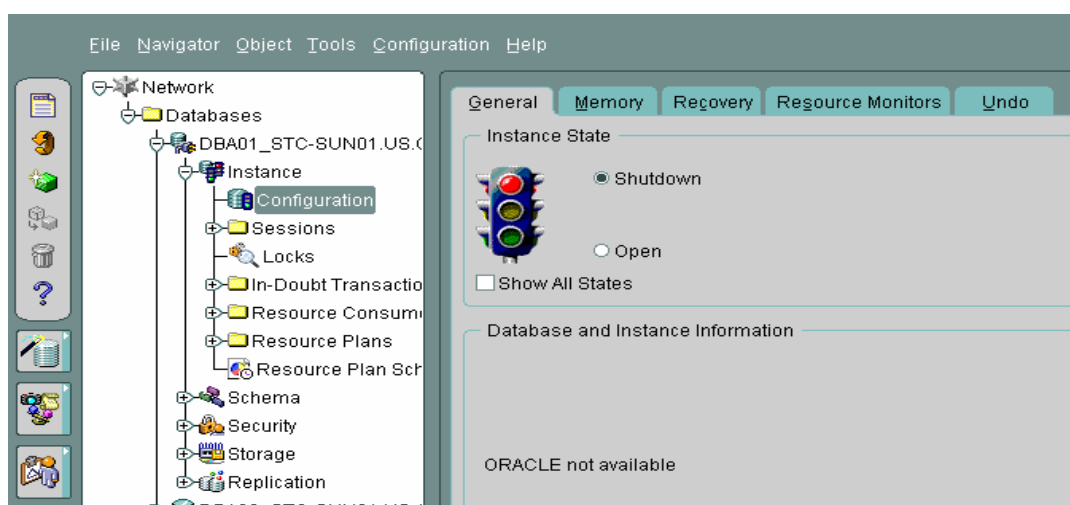
关闭选项（续）

使用 Oracle Enterprise Manager 关闭数据库

从 OEM 控制台：

1. 导航到“数据库” (Databases) > “例程” (Instance)
2. 单击“配置” (Configuration)
3. 从“常规” (General) 选项卡，选择“打开” (Open) 选项。
4. 单击“应用” (Apply)。

注：必须以 SYSDBA 权限连接到数据库才能执行关闭操作。



使用诊断文件监视例程

- 诊断文件
 - 包含有关出现的重要事件的信息
 - 用于解决问题
 - 用于更好地进行数据库的日常管理
- 共有以下几种类型：
 - alertSID.log 文件
 - 后台跟踪文件
 - 用户跟踪文件

ORACLE

3-36

Copyright © Oracle Corporation, 2001. All rights reserved.

使用诊断文件监视例程

诊断文件是获取有关数据库活动的信息的一种方法。同时也是管理例程的有用工具。诊断文件有几种类型。创建的诊断文件的类型取决于出现的问题或需要传播的信息。

- alertSID.log 文件：记录数据库日常操作的信息
- 后台跟踪文件：记录 SMON、PMON、DBWn 和其它后台进程失败时产生的重要信息
- 用户跟踪文件：记录出现严重用户错误或用户强制执行跟踪文件时产生的重要信息

警报日志文件

- **alertSID.log 文件:**
 - 记录命令
 - 记录主要事件结果
 - 用于记录日常操作信息
 - 用于诊断数据库错误
- 每个条目都带有与之相关联的时间戳
- 必须由 **DBA** 进行管理
- 存储位置由 **BACKGROUND_DUMP_DEST** 定义

ORACLE

3-37

Copyright © Oracle Corporation, 2001. All rights reserved.

警报日志文件

每个 Oracle 例程都有一个警报日志文件。如果该文件尚未创建，将在例程启动过程中进行创建。警报日志文件由您进行管理，并随着数据库的继续运行而不断增长。诊断日常操作或错误时，应该首先查看警报日志文件。警报日志文件还包含指向跟踪文件的指针，从而可获得更详细的信息。

警报日志文件记录了以下信息：

- 数据库启动或关闭的时间
- 所有非缺省初始化参数的列表
- 后台进程的启动
- 例程使用的线程
- 正在向其中写入信息的日志序列号 LGWR
- 有关日志切换的信息
- 表空间的创建和撤消段
- 已发出的警报声明
- 有关 ORA-600 等错误消息和区错误的信息

警报日志文件（续）

alert_SID.log 的存储位置由 BACKGROUND_DUMP_DEST 初始化参数定义。

后台跟踪文件

- 后台跟踪文件
 - 记录所有后台进程检测到的错误
 - 用于诊断并排除错误
- 在后台进程遇到错误时创建
- 存储位置由 `BACKGROUND_DUMP_DEST` 定义

ORACLE

3-39

Copyright © Oracle Corporation, 2001. All rights reserved.

后台跟踪文件

后台跟踪文件用于记录后台进程（如 SMON、PMON、DBWn 和其它后台进程）遇到的错误。只有出现需要写入跟踪文件的错误时，才会创建后台跟踪文件。您可使用它们来诊断和解决问题。最初创建后台跟踪文件后，文件内包含指示数据服务器和操作系统的版本号的标头信息。

用户跟踪文件的命名约定：`sid_processname_PID.trc` (`db01_lgwr_23845.trc`)。其存储位置由 `BACKGROUND_DUMP_DEST` 初始化参数定义。

用户跟踪文件

- 用户跟踪文件
 - 由用户进程生成
 - 可由服务器进程生成
 - 包含跟踪的 SQL 语句的统计信息
 - 包含用户错误消息
- 在用户遇到会话错误时创建
- 存储位置由 `USER_DUMP_DEST` 定义
- 大小由 `MAX_DUMP_FILE_SIZE` 定义

ORACLE

3-40

Copyright © Oracle Corporation, 2001. All rights reserved.

用户跟踪文件

用户跟踪文件包含跟踪的 SQL 语句的统计信息，这对于 SQL 优化非常有用。此外，用户跟踪文件还包含用户错误消息。

用户跟踪文件的命名约定：`sid_ora_PID.trc(db01_ora_23845.trc)`。

其存储位置由 `USER_DUMP_DEST` 初始化参数定义。

启用或禁用用户跟踪

- 会话级别：

- 使用 ALTER SESSION 命令：

- ```
ALTER SESSION SET SQL_TRACE = TRUE
```

- 执行 DBMS 过程：

- ```
dbms_system.SET_SQL_TRACE_IN_SESSION
```

- 例程级别

- 设置初始化参数：

- ```
SQL_TRACE = TRUE
```

ORACLE

3-41

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用或禁用用户跟踪

注：在例程级别设置 SQL\_TRACE=TRUE 后将生成大量跟踪数据。因此，要谨慎使用此选项。

用户跟踪的详细信息将在 *Oracle9i SQL 语句优化* 课程中讲述。

## 启用或禁用用户跟踪

### 使用 Oracle Enterprise Manager 启用或禁用用户跟踪

从 OEM 控制台：

1. 导航到 “数据库” (Databases) > “例程” (Instance) > “配置” (Configuration)。
2. 从 “常规” (General) 页选择 “全部初始化参数” (Initialization Parameters)。
3. 设置参数 `SQL_TRACE = TRUE`。
4. 选择 “确定” (OK)。



## 小结

在这一课中，您应该能够掌握：

- 创建和管理初始化参数文件
- 启动和关闭例程
- 监视和使用诊断文件

ORACLE<sup>®</sup>

## 练习 3 概览

此练习涉及以下主题：

- 创建 SPFILE
- 以不同模式启动和关闭数据库

ORACLE

3-44

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 3 概览

注：本练习可以使用 SQL\*Plus 或使用 OEM 和 SQL\*Plus Worksheet 来完成。

### 练习 3: 管理 Oracle 例程

- 1 以 SYS 用户身份连接到数据库并关闭数据库。
- 2 关闭数据库后, 从 PFILE 创建 SPFILE。将 SPFILE 以 spfileSID.ora 文件名格式存放在目录 \$HOME/ADMIN/PFILE 中。使用例程名称来代替 SID。从 \$HOME/ADMIN/PFILE 目录中的 PFILE 创建 SPFILE。
- 3 从操作系统查看 SPFILE。
- 4 以 SYS 用户身份连接, 然后使用 SPFILE 启动数据库。
- 5
  - a 关闭数据库然后以只读模式打开。
  - b 以用户名 HR 和口令 HR 连接, 然后在 REGIONS 表中插入以下一行:  

```
INSERT INTO regions VALUES (5, 'Mars');
```

结果如何?
  - c 将数据库重置回读写模式。
- 6
  - a 以用户名 HR 和口令 HR 连接, 在 REGIONS 表中插入以下行; 不要提交或退出。  

```
INSERT INTO regions VALUES (5, 'Mars');
```
  - b 使用一个新的 Telnet 会话启动 SQL\*Plus。以 SYS AS SYSDBA 用户身份连接并执行关闭事务处理。
  - c 回退 HR 会话中的插入操作并退出。HR 会话有何变化? SYS 会话又有何变化?
- 7
  - a 以 SYS 用户身份启动数据库。
  - b 以 HR 用户身份启动另一个会话。  
注: 保持打开两个 SQL\*Plus 会话, 一个会话用于用户 SYS, 另一个会话用于用户 HR。
  - c 以 SYS 用户身份启用受限会话。
  - d 以 HR 用户身份对 REGIONS 表执行 SELECT 命令。SELECT 命令是否成功? 退出会话, 然后重新以 HR 用户身份连接。结果如何?
  - e 以 SYS 用户身份禁用受限会话。



# 4

## 创建数据库

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 了解创建数据库的前提条件
- 使用 **Oracle Database Configuration Assistant** 创建数据库
- 手动创建数据库
- 使用 “**Oracle 管理文件**” (**Oracle Managed Files**) 创建数据库

ORACLE

## 管理和组织数据库

- **计划数据库是管理数据库系统的第一步**
  - 确定数据库的用途
  - 确定数据库的类型
  - 概括数据库的体系结构设计
  - 选择数据库名称
- **创建数据库**
- **使用 Oracle Data Migration Assistant 可以从较早的数据库版本进行移植**

ORACLE

4-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### 计划和组织数据库

计划数据库是组织和实施数据库系统的第一步。首先要确定数据库的用途，这就需要根据业务要求来确定应该创建哪种数据库类型。数据库类型包括数据仓库、用于高效的联机事务处理或用于通用目的的数据库。确定了用途和类型后，接下来就是概括要应用的数据库体系结构。例如：如何组织和存储数据文件、控制文件和重做日志文件？Oracle 的“最佳灵活体系结构” (Optimal Flexible Architecture) 可以帮助您安排数据库文件的结构和位置。定义了体系结构之后，您必须为新的数据库选择数据库名称和系统标识名。

数据库创建这项任务可用于准备一些操作系统文件，创建只需一次，这与数据库中的数据文件数量无关。

从较早版本的 Oracle 数据库进行移植时，除非需要一个全新的数据库，否则不必创建数据库。如果不必创建数据库，则可以使用移植实用程序。Oracle Data Migration Assistant 就是这样的工具，它可以协助您移植当前的数据库系统。

## 最佳灵活体系结构 (OFA)

- **Oracle 建议使用的标准数据库体系结构布局**
- **OFA 涉及三个主要规则：**
  - 建立一个目录结构，在该目录结构中，任何数据库文件都可以存储在任意磁盘资源上。
  - 将具有不同行为的对象分放到不同的表空间。
  - 通过将数据库组件分别安装到不同的磁盘资源上，使数据库获得最高的可靠性和最佳的性能。

ORACLE

4-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### 最佳灵活体系结构 (OFA)

所有支持的平台上的安装和配置都符合最佳灵活体系结构 (OFA)。OFA 按类型和用途来组织数据库文件。二进制文件、控制文件、日志文件和管理文件可分装在多个磁盘上。

一致的命名约定具有以下优点：

- 可以很容易地将数据库文件与其它文件区别开。
- 易于识别控制文件、重做日志文件和数据文件。
- 通过将文件分装在不同磁盘和目录中，对同一台计算机上的多个 Oracle 主目录的管理变得更加容易。
- 可实现更好的性能，因为数据文件、二进制文件和管理文件现在分别驻留在不同的目录和磁盘上，这样就减少了它们对磁盘的争用。



## Oracle 软件和文件的位置

| oracle_base 软件               | 文件                         |
|------------------------------|----------------------------|
| <code>/product</code>        | <code>oradata/</code>      |
| <code>/release_number</code> | <code>db01/</code>         |
| <code>/bin</code>            | <code>system01.dbf</code>  |
| <code>/dbs</code>            | <code>control01.ctl</code> |
| <code>/rdbms</code>          | <code>redo0101.log</code>  |
| <code>/sqlplus</code>        | <code>...</code>           |
| <code>/admin</code>          | <code>db02/</code>         |
| <code>/inst_name</code>      | <code>system01.dbf</code>  |
| <code>/pfile</code>          | <code>control01.ctl</code> |
|                              | <code>redo0101.log</code>  |
|                              | <code>...</code>           |

### Oracle 软件的位置

上面的目录树是符合 OFA 的数据库示例。

#### 最佳灵活体系结构：

安装和创建数据库过程中的另一个重要问题就是如何组织文件系统，以便于对数据库的增长进行管理。数据库的增长体现在向现有数据库添加数据、添加用户、创建新数据库、添加硬件、在众多的驱动器之间适当地分配输入/输出 (I/O) 负载等。

## 创建的前提条件

要新建数据库，您必须具备以下条件：

- 已授权的帐户，通过以下方式之一验证：
  - 操作系统
  - 口令文件
- 有足够的内存可用于启动例程
- 有足够的磁盘空间可用于计划要创建的数据库

ORACLE

4-6

Copyright © Oracle Corporation, 2001. All rights reserved.

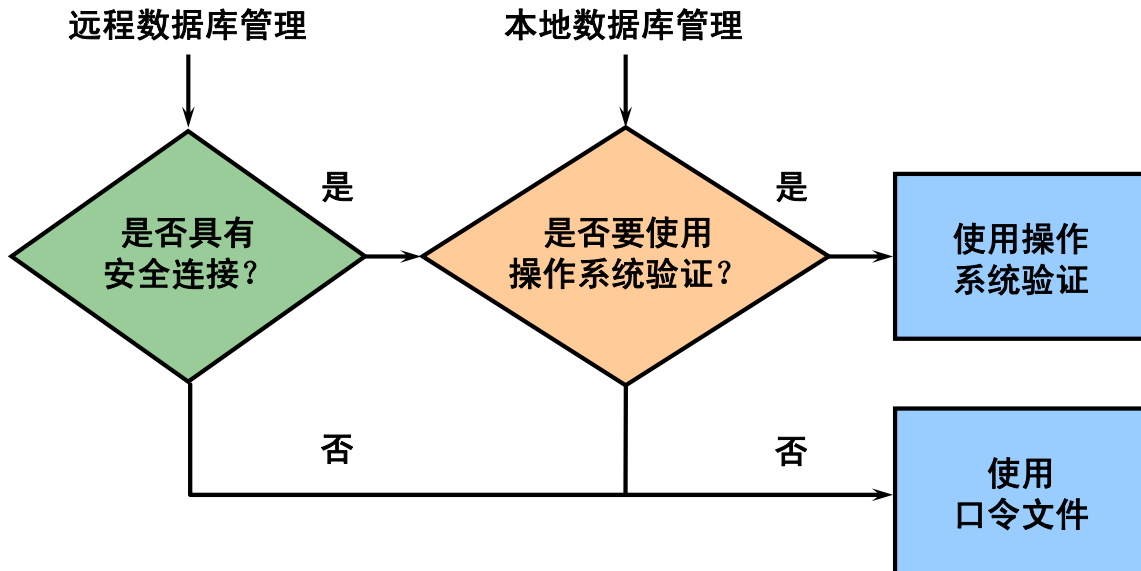
### 创建的前提条件

创建数据库需要具备 SYSDBA 权限。使用操作系统验证或口令文件验证即可授予这些权限。

创建数据库之前，确保有足够的内存可用于 SGA、Oracle 可执行程序 and 进程。请参考操作系统安装和管理指南。

计算数据库所需的磁盘空间，包括联机重做日志文件、控制文件和数据文件。

# 数据库管理员的验证方法



ORACLE

4-7

Copyright © Oracle Corporation, 2001. All rights reserved.

## 数据库管理员的验证方法

您可能需要从数据库所驻留的计算机上对数据库进行本地管理，或者需要从一个远程客户端管理许多不同的数据库服务器，根据实际情况，请选择使用操作系统还是口令文件来验证数据库管理员。

注：有关操作系统验证的信息，请参考针对各操作系统的手册。

## 使用口令文件验证

- 使用口令实用程序创建口令文件

```
$ orapwd file=$ORACLE_HOME/dbs/orapwU15
password=admin entries=5
```

- 在初始化参数文件中设置  
`REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE`
- 向口令文件添加用户
- 向每个用户分配适当的权限

```
GRANT SYSDBA TO HR;
```

ORACLE

4-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### 如何使用口令文件验证

Oracle 提供了一个口令实用程序 `orapwd` 来创建口令文件。使用 `SYSDBA` 权限进行连接时，您将以 `SYS` 方案而不是与您的用户名关联的方案进行连接。对于 `SYSOPER`，将连接到 `PUBLIC` 方案。

使用口令文件访问数据库的权限通过特权用户发出的特殊 `GRANT` 命令来提供。

注：有关授予权限的信息，请参考“管理权限”一课。

## 如何使用口令文件验证（续）

### 使用口令文件：

1. 使用口令实用程序 orapwd 创建口令文件。

```
orapwd file=filename password=password entries=max_users
```

其中：

filename: 口令文件的名称（必需）

password: SYSOPER 和 SYSDBA 的口令（必需）

entries: 允许作为 SYSDBA 或 SYSOPER 连接的不同用户的最大数目。如果超过该值，则必须创建一个新的口令文件。所以设置较大的值较为稳妥。等号 (=) 字符的两侧不应有空格。

示例: orapwd file=\$ORACLE\_HOME/dbs/orapwU15  
password=admin entries=5

其中：

filename: \$ORACLE\_HOME/dbs/orapwU15

password: admin

entries: 5

2. 将 REMOTE\_LOGIN\_PASSWORDFILE 参数设置为 EXCLUSIVE

其中：

EXCLUSIVE 表示只有一个例程可以使用口令文件，并且该口令文件包含 SYS 以外的名称。使用 EXCLUSIVE 口令文件可以向单个用户授予 SYSDBA 或 SYSOPER 权限。

3. 使用上面创建的口令文件连接数据库。

```
CONNECT sys/admin AS SYSDBA
```

### 口令文件位置：

UNIX: \$ORACLE\_HOME/dbs

NT: %ORACLE\_HOME%/database

### 维护口令文件：

使用操作系统命令删除现有口令文件，然后使用口令实用程序创建一个新的口令文件。

# 创建数据库

可通过以下方式创建 Oracle 数据库：

- **Oracle Universal Installer**
- **Oracle Database Configuration Assistant**
  - 图形用户界面
  - 基于 Java
  - 由 Oracle Universal Installer 启动
  - 可独立使用
- **CREATE DATABASE 命令**

ORACLE

4-10

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建数据库

共有三种创建数据库的方式：使用 Oracle Universal Installer 在 Oracle9i 安装中自动创建；使用 Oracle Database Configuration Assistant (DBCA)；或使用 CREATE DATABASE 命令通过创建 SQL 脚本来创建数据库。

Database Configuration Assistant 是一个用来简化数据库创建操作的图形用户界面，它既能与 Oracle Universal Installer 交互使用，也可以独立使用。DBCA 基于 Java，可以从任何带 Java 引擎的平台启动。

安装 Oracle Server 的过程中，Oracle Universal Installer 会启动 DBCA，接着 DBCA 将自动创建一个初始数据库。对于用或不用 DBCA，创建或不创建初始数据库，以及是否在安装后作为独立的应用程序启动 DBCA 来创建数据库，您都有充分的选择自由。

如果使用的是较早版本的 Oracle 软件，还可以对现有数据库进行移植或升级。

# 操作系统环境

设置以下环境变量：

- ORACLE\_BASE
- ORACLE\_HOME
- ORACLE\_SID
- ORA\_NLS33
- PATH
- LD\_LIBRARY\_PATH

ORACLE

4-11

Copyright © Oracle Corporation, 2001. All rights reserved.

## 操作系统环境

手动创建数据库或使用 Database Configuration Assistant 创建数据库之前，必须正确配置操作系统环境。

ORACLE\_BASE：指定 Oracle 软件的顶级目录。

示例：/u01/app/oracle

ORACLE\_HOME：指定 Oracle 软件的安装目录。OFA 建议的目录为  
\$ORACLE\_BASE/product/release

示例：/u01/app/oracle/product/9.1.1

ORACLE\_SID：指定例程名称，同一台计算机上运行的 Oracle 例程的名称必须唯一。

ORA\_NLS33：创建带有非 US7ASCII 字符集的数据库时必须指定。

示例：\$ORACLE\_HOME/ocommon/nls/admin/data

PATH：指定操作系统查找可执行程序（如 SQL\*Plus）时所搜索的路径。Oracle9i 可执行程序位于 \$ORACLE\_HOME/bin 目录下，需要添加到 PATH 变量中。

LD\_LIBRARY\_PATH：指定操作系统和 Oracle 库文件所在的目录。示例：  
\$ORACLE\_HOME/lib

# Database Configuration Assistant

**Database Configuration Assistant 可用于：**

- 创建数据库
- 配置数据库选件
- 删除数据库
- 管理模板
  - 使用预定义模板设置创建新模板
  - 从现有数据库创建新模板
  - 删除数据库模板

ORACLE

4-12

Copyright © Oracle Corporation, 2001. All rights reserved.

## Database Configuration Assistant

管理模板是 Oracle9i 中的新增功能。您可以利用一些预定义的模板，也可以将现有数据库用作副本来创建新的数据库或模板。数据库参数以 XML 格式存储。

使用模板的好处：

- 节省创建数据库的时间
- 可共享模板
- 可根据需要变换数据库选件

有关模板的详细信息，请参考 Oracle Database Configuration Assistant 联机帮助。



## 使用 Database Configuration Assistant 创建数据库

- 选择要从预定义模板创建的数据库类型
- 指定全局数据库名称和 SID
- 选择要在数据库中使用的功能
- 确定数据库创建后要运行的任何脚本
- 选择数据库的运行模式

ORACLE

4-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 Database Configuration Assistant 创建数据库

- 启动 Database Configuration Assistant: 程序 (Programs) > Oracle-OraHome90 > 配置和移植工具 (Configuration and Migration Tools) > Database Configuration Assistant。
- 选择“创建数据库”(Create a Database) 选项。
- 从预定义模板列表中选择要创建的数据库类型。
  - 数据仓库
  - 通用
  - 新数据库
  - 事务处理

使用“显示详细资料”(Show Details) 选项查看要创建的数据库。创建的模板可带有数据文件，也可不带数据文件。

- 不带数据文件：只包含数据库的结构。可以指定和更改所有数据库参数。
- 带有数据文件：既包含数据库的结构也包含其中的物理数据文件。自动为数据库创建所有日志文件和控制文件，并且可添加/删除控制文件、日志组，还可以更改数据文件的目标位置和名称。无法添加或删除数据文件、表空间或回退段。无法更改初始化参数。

## 创建数据库（续）

- 指定“全局数据库名称” (Global Database Name) 和 SID。
- 指定要在数据库中使用功能，如：
  - Oracle Spatial
  - Oracle OLAP services
  - 示例模式 (Example Schemas)

示例模式 (Example Schemas) 包含以下类型的表的脚本：

人力资源

订单输入

产品介质

销售历史记录

装运

- 确定数据库创建后要运行的任何脚本。
- 选择数据库的运行模式
  - 专用服务器模式
  - 共享服务器模式

## 使用 Database Configuration Assistant 创建数据库

- 指定内存、归档、数据库大小和文件位置等选项
- 定义数据库存储参数
- 按需更改文件位置变量
- 选择一个数据库创建选项完成数据库的创建

ORACLE

4-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 Database Configuration Assistant 创建数据库

- 指定以下选项

- 内存

选择“典型”(Typical) 或“自定义”(Custom) 数据库

选择“典型”选项创建的数据库涉及的用户输入最少。选择此选项后，可以指定以下任一运行数据库的环境：联机事务处理 (Online Transaction Processing, OLTP)、多用途和数据仓库。

“自定义”选项允许您自定义数据库的创建。该选项仅供具有高级数据库创建经验的数据库管理员使用。

- 归档

此选项将数据库置于 ARCHIVELOG 模式，并在重新使用重做日志文件前对其进行归档。

- DB 大小

此参数用于定义数据库的块大小和排序区大小。只能在创建数据库时指定数据块的大小。SORT\_AREA\_SIZE 指可用于排序操作的最大内存空间。

## 创建数据库（续）

- 文件位置

用于指定跟踪文件的位置，以及指定初始化参数文件的路径。

- 定义数据库存储参数。此页显示一个树列表以及一个摘要视图（多栏列表），允许您更改和查看以下对象：控制文件、表空间、数据文件、还原段和重做日志组。
- 单击“文件位置变量” (File Location Variables) 按钮可更改任意文件位置变量。
- 选择一个数据库创建选项完成数据库的创建。
  - 创建数据库 (Create Database)：此选项可用于立即创建数据库。
  - 另存为数据库模板 (Save as a Database Template)：此选项可用于将数据库创建参数保存为模板。此模板随即将添加到可用模板列表中。
  - 生成数据库创建脚本 (Generate Database Creation Scripts)：此选项可用于将数据库创建参数保存为脚本文件，供以后使用。
- 选择“完成” (Finish)。

## 手动创建数据库

- 为例程和数据库选择唯一的名称。
- 选择一个数据库字符集。
- 设置操作系统变量。
- 创建初始化参数文件。
- 在 NOMOUNT 阶段启动该例程。
- 创建并执行 CREATE DATABASE 命令。
- 打开数据库。
- 运行脚本以生成数据字典并完成创建后的步骤。
- 根据需要创建其它表空间。

ORACLE

4-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### 手动创建数据库

- 为例程和数据库选择唯一的名称。
- 选择一个数据库字符集。  
必须定义一个数据库字符集。同时还可选择定义一个国家字符集。例如：
  - 字符集 Character set AL32UTF16
  - 国家字符集 AL16UTF16 (National character set AL16UTF16)

有关可供使用的各种字符集的信息，请参考“使用全球化支持”一课。

- 设置操作系统变量。

需要设置四个环境变量：ORACLE\_HOME、ORACLE\_SID、PATH、LD\_LIBRARY\_PATH。

- ORACLE\_HOME：安装 Oracle9i 服务器的顶级目录。
- ORACLE\_SID：可由用户定义的、分配给数据库例程的名称。用于区分运行在同一台机器上的不同数据库例程
- PATH：定义操作系统查找可执行程序时要搜索的目录。
- LD\_LIBRARY\_PATH：定义所需的库文件的存储目录。

## 手动创建数据库（续）

- 创建初始化参数文件。

初始化参数文件是通过随安装过程安装的 `init.ora` 示例文件而创建的。复制 `init.ora` 示例文件，将其命名为 `initSID.ora`。针对要创建的数据库的具体需要来修改该文件。如果要使用 `SPFILE`，则必须首先创建 `PFILE`。有关如何创建数据库特定的 `initSID.ora` 文件和 `SPFILE` 的指导说明，请参考“管理 Oracle 例程”一课。

- 在 `NOMOUNT` 阶段启动该例程。

以具有 `SYSDBA` 权限的用户 `SYS` 身份连接。要创建数据库，数据库必须处于 `NOMOUNT` 状态。有关如何让数据库处于 `NOMOUNT` 状态的指导说明，请参考“管理 Oracle 例程”一课。

- 创建并执行 `CREATE DATABASE` 命令。

- 创建包含 `CREATE DATABASE` 命令的 `SQL` 脚本。以具有 `SYSDBA` 权限的用户 `SYS` 身份连接到 `SQL*Plus`。当数据库处于 `NOMOUNT` 状态时，执行该脚本。
- 如果要创建的数据库是通过“Oracle 管理文件” (Oracle Managed Files, OMF) 来管理操作系统文件的，那么 `CREATE DATABASE` 命令的简化程度将非常明显。有关 OMF 的信息，请参考“管理 Oracle 例程”一课。

- 打开数据库。

必须首先打开数据库，然后再运行脚本，创建数据字典并完成创建后的步骤。有关如何从 `NOMOUNT` 状态下打开数据库的指导说明，请参考“管理 Oracle 例程”一课。

- 运行脚本。

- 创建数据库后必须运行两个脚本：`catalog.sql` 和 `catproc.sql`。这两个脚本都必须以具有 `SYSDBA` 权限的用户 `SYS` 身份运行。执行脚本前，数据库必须处于 `OPEN` 状态。
- `catalog.sql`：在基表和动态性能视图上创建视图及其同义词。它还启动其它脚本，为以下各项创建对象：
  - PL/SQL 基本环境，包括 PL/SQL 数据类型的声明、预定义异常、内置过程和函数、SQL 操作等
  - 审计
  - 导入/导出
  - SQL\*Loader
  - 已安装选项

## 手动创建数据库（续）

- 运行脚本（续）
  - `catproc.sql`: 创建使用 PL/SQL 所需的程序包和过程。此外，此脚本还创建用于扩展 RDBMS 功能的若干 PL/SQL 程序包，以及用于预警、管道、logminer、大对象、对象、排队、复制和其它内置选项的程序包视图。
  - `pupbld.sql`: 创建名为“产品用户配置文件” (Product User Profile) 的表以及相关的过程。运行此脚本将在用户每次连接到 SQL\*Plus 时防止生成警告消息。
- 注：必须以用户 SYSTEM 的身份运行此脚本。
- 创建其它表空间。
  - 应该根据数据库的需要创建其它表空间。

注：附录 A 提供了在 UNIX 环境中手动创建数据库的分步说明。此外，有关在特定的平台上创建数据库的信息，还可参考针对不同操作系统的 Oracle 文档。

# 创建数据库

```
CREATE DATABASE user01
LOGFILE
 GROUP 1 ('/$HOME/ORADATA/u01/redo01.log') SIZE 100M,
 GROUP 2 ('/$HOME/ORADATA/u02/redo02.log') SIZE 100M,
 GROUP 3 ('/$HOME/ORADATA/u03/redo03.log') SIZE 100M
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
DATAFILE '/$HOME/ORADATA/u01/system01.dbf' SIZE 325M
UNDO TABLESPACE undotbs
DATAFILE '/$HOME/ORADATA/u02/undotbs01.dbf' SIZE 200M
 AUTOEXTEND ON NEXT 5120K MAXSIZE UNLIMITED
DEFAULT TEMPORARY TABLESPACE temp
CHARACTER SET US7ASCII
NATIONAL CHARACTER SET AL16UTF16
SET TIME_ZONE= 'America/New_York'
```

ORACLE

4-20

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建数据库

若要创建数据库，请使用下列 SQL 命令：

```
CREATE DATABASE [database]
[CONTROLFILE REUSE]
[LOGFILE [GROUP integer] filespec]
[MAXLOGFILES integer]
[MAXLOGMEMBERS integer]
[MAXLOGHISTORY integer]
[MAXDATAFILES integer]
[MAXINSTANCES integer]
[ARCHIVELOG|NOARCHIVELOG]
[CHARACTER SET charset]
[NATIONAL CHARACTER SET charset]
[DATAFILE filespec [autoextend_clause]]
```



## 创建数据库（续）

```
filespec ::= 'filename' [SIZE integer][K|M] [REUSE]
 autoextend_clause ::=
 [AUTOEXTEND {OFF|ON [NEXT integer[K|M]]
 [MAXSIZE {UNLIMITED|integer[K|M]} }}]
[DEFAULT TEMPORARY TABLESPACE tablespace filespec
 [temp_tablespace_extent_clause]
temp_tablespace_extent_clause::=
EXTENT MANAGEMENT LOCAL UNIFORM [SIZE integer][K|M]]
[UNDO TABLESPACE tablespace DATAFILE filespec
 [autoextend_clause]]
[SET TIME_ZONE [time_zone_region]]
}
]
```

其中：

- DATABASE：是要创建的数据库的名称（如果省略了数据库的名称，则使用初始化参数 DB\_NAME 的值）。
- CONTROLFILE REUSE：指定应重新使用参数文件中确定的现有控制文件
- LOGFILE GROUP：指定要使用的日志文件的名称及其所属的组
- MAXLOGFILES：指定可以为数据库创建的重做日志文件组的最大数量
- MAXLOGMEMBERS：指定日志文件组的日志文件成员的最大数量
- MAXLOGHISTORY：指定 Oracle Real Application Clusters 的自动介质恢复能够恢复的归档重做日志的最大数量

## 创建数据库（续）

- **DATAFILE:** `filespec` 指定要使用的数据文件
- **AUTOEXTEND:** 启用或禁用数据文件的自动扩展
- **MAXDATAFILES:** 指定在执行 **CREATE DATABASE** 或 **CREATE CONTROLFILE** 时控制文件的数据文件段的初始大小调整。如果尝试添加新文件（其数量大于 **MAXDATAFILES**，但小于或等于 **DB\_FILES**），将引起控制文件自动扩展，以便数据文件段能够容纳更多的文件。
- **MAXINSTANCES:** 是能够同时装载和打开数据库的例程的最大数量
- **ARCHIVELOG:** 指定重新使用重做日志前必须归档
- **NOARCHIVELOG:** 指定不必对重做日志的内容进行归档即可重新使用这些日志
- **CHARACTER SET:** 是数据库用来存储数据的字符集
- **NATIONAL CHARACTER SET:** 指定在定义为 **NCHAR**、**NCLOB** 或 **NVARCHAR2** 的列中存储数据时所使用的国家字符集。如果未指定，则国家字符集与数据库字符集相同
- **DEFAULT TEMPORARY TABLESPACE:** 为数据库创建缺省临时表空间。Oracle 会将所有未指定其它临时表空间的用户都分配到此临时表空间
- **UNDO TABLESPACE:** 创建还原表空间，并创建指定的数据文件，作为还原表空间的一部分
- **SET TIME\_ZONE:** 设置数据库所在的时区

## 使用“Oracle 管理文件”(OMF) 创建数据库

- 使用 OMF 可简化操作系统上的文件管理
- OMF 由 Oracle 服务器通过 SQL 命令创建和删除
- OMF 是通过设置以下两个参数来建立的：
  - DB\_CREATE\_FILE\_DEST: 设置该参数以提供数据文件的缺省位置
  - DB\_CREATE\_ONLINE\_LOG\_DEST\_N: 设置该参数以提供联机重做文件和控制文件的缺省位置
    - 最多可设置五个位置

ORACLE

4-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用“Oracle 管理文件”(OMF) 创建数据库

OMF 免除了对 Oracle 数据库中的文件进行直接管理的必要，从而简化了文件管理。

OMF 按如下方式命名：

- 控制文件：ora\_%u.ctl
- 重做日志文件：ora\_%g\_%u.log
- 数据文件：ora\_%t\_%u.dbf
- 临时数据文件：ora\_%t\_%u.tmp

其中的一些字符定义如下：

- %u 是一个八个字符长的字符串，可以确保唯一性。
- %t 是表空间名，如有必要，可按照文件名的最大长度要求将其截断。将表空间名放在唯一性字符串之前，意味着表空间内的所有数据文件按照字母顺序排列显示。
- %g 是重做日志文件组号。
- 带 .dbf 扩展名的 ora\_ 表明该文件是 OMF。

## 使用“Oracle 管理文件”(OMF) 创建数据库（续）

还原文件没有特殊的扩展名。

不必同时设置参数 DB\_CREATE\_FILE\_DEST 和 DB\_CREATE\_ONLINE\_LOG\_DEST\_N，可以使用其中任意一个，也可以同时使用。

## 使用“Oracle 管理文件”(OMF) 创建数据库

- 在初始化参数文件中定义 OMF 参数。示例：
  - DB\_CREATE\_FILE\_DEST=/ \$HOME/ORADATA/u05
  - DB\_CREATE\_ONLINE\_DEST\_1=/ \$HOME/ORADATA/u01
  - DB\_CREATE\_ONLINE\_DEST\_2=/ \$HOME/ORADATA/u02
- CREATE DATABASE 命令得到了简化：

```
@cddb01.sql
> CREATE DATABASE dba01;
```

ORACLE

4-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用“Oracle 管理文件”(OMF) 创建数据库

要使用 OMF 创建数据库，需要在初始化参数文件中定义参数 DB\_CREATE\_FILE\_DEST 和 DB\_CREATE\_ONLINE\_DEST\_n。一旦定义了 OMF 参数，您就不必再定义文件名或位置，创建数据库的语法因而得到了简化。

上面的示例中创建的数据库使用了如下 OMF：

- 位于目录 / \$HOME/ORADATA/u05 中的 SYSTEM 表空间数据文件，大小为 100 MB，可无限限制地自动扩展。
- 两个联机重做日志组，组中各有两个大小为 100 MB 的成员，分别位于 / \$HOME/ORADATA/u01 和 / \$HOME/ORADATA/u02 中
- 如果启用了自动还原管理模式，目录 / \$HOME/ORADATA/u05 中还会有一个还原表空间数据文件，大小为 10 MB，可无限限制地自动扩展。同时创建一个名为 SYS\_UNDOTBS 的还原表空间。
- 如果未指定 CONTROL\_FILES 初始化参数，则会生成两个控制文件，分别位于 / \$HOME/ORADATA/u01 和 / \$HOME/ORADATA/u02 中。位于 / \$HOME/ORADATA/u01 中的控制文件是主控制文件。

## 使用“Oracle 管理文件”(OMF) 创建数据库（续）

如果 CREATE DATABASE 命令失败，创建的所有 OMF 都将被删除。当用户从 DBA\_DATAFILES、V\$DATAFILE、V\$CONTROLFILE 和 V\$LOGFILE 中进行选择时，可看到内部生成的文件名。

可通过 ALTER SYSTEM SET 命令动态修改 DB\_CREATE\_FILE\_DEST 和 DB\_CREATE\_ONLINE\_LOG\_DEST\_N。

## 故障排除

如果存在以下情况，数据库创建将无法成功：

- **SQL 脚本中存在语法错误**
- **要创建的文件已经存在**
- **出现操作系统错误，如文件或目录权限错误，或空间不足错误**

ORACLE

4-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### 故障排除

如果幻灯片中显示的三个问题中的任何一个问题发生，CREATE DATABASE 语句将以失败告终。应该删除 CREATE DATABASE 语句创建的所有文件，改正错误，然后尝试重新创建。

## 数据库的创建结果

数据库应包含：

- 数据文件、控制文件和重做日志文件
- 用户 SYS，口令为 change\_on\_install
- 用户 SYSTEM，口令为 manager
- 内部表（但没有数据字典视图）

ORACLE

4-28

Copyright © Oracle Corporation, 2001. All rights reserved.

### 数据库的创建结果

创建数据库后，该例程将运行起来，数据库也处于打开状态，可正常操作。数据库中包含用户 SYS 和 SYSTEM。根据数据库的创建方法，即使用 DBCA 创建或手动创建，可以相应地创建其他用户。创建数据库后应立即更改 SYS 和 SYSTEM 的口令。

可以查看动态性能视图，如 V\$LOGFILE、V\$CONTROLFILE 和 V\$DATAFILE，但没有创建数据字典视图。



## 小结

在这一课中，您应该能够掌握：

- 创建数据库的前提条件
- 使用 **Oracle Database Configuration Assistant** 创建数据库
- 手动创建数据库
- 使用 “**Oracle 管理文件**” (**Oracle Managed Files**) 创建数据库

ORACLE

## 练习 4 概览

- 本课提供了两种创建数据库的方式：
  - 使用 **Database Configuration Assistant** 通过图形化的步骤创建数据库。请按如下方式启动：  
开始(Start) > 程序(Programs) > Oracle-OraHome90 > 配置和移植工具(Configuration and Migration Tools)。
  - 附录 A 提供了在 UNIX 系统中手动创建数据库的分步指导。
- 复习这些步骤，选择手动创建数据库或使用 **Database Configuration Assistant** 来创建数据库。

ORACLE

# 5

## 使用数据字典和动态性能视图

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 了解内置数据库对象
- 了解数据字典的内容和使用
- 了解数据字典视图的创建方式
- 了解数据字典视图的类别
- 查询数据字典和动态性能视图
- 了解管理脚本命名约定

ORACLE

# 内置数据库对象

随数据库一起创建的其它对象：

- 数据字典
- 性能表
- PL/SQL 程序包
- 数据库事件触发器

ORACLE

5-3

Copyright © Oracle Corporation, 2001. All rights reserved.

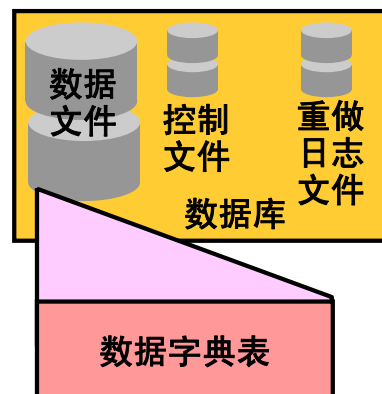
## 内置数据库对象

创建数据库时，除创建数据库文件外，还将创建其它几种结构。

- 数据字典：包含对数据库中对象的说明
- 动态性能表：包含由数据库管理员 (DBA) 用来监视和优化数据库及例程的信息
- PL/SQL 程序包：向数据库添加功能的程序单元。这些程序包在执行 `CREATE DATABASE` 命令之后运行 `catproc.sql` 脚本时创建。PL/SQL 程序包不在本课讨论的范围之内。
- 数据库事件触发器：触发器是在表或视图发生修改，或执行某些用户操作或数据库系统操作时隐式执行的过程。数据库事件触发器不在本课讨论的范围之内。

# 数据字典

- 每个 Oracle 数据库的中心
- 描述数据库以及数据库对象
- 包含只读表和视图
- 存储在 SYSTEM 表空间内
- 由用户 SYS 拥有
- 由 Oracle 服务器进行维护
- 通过 SELECT 访问



ORACLE

## 数据字典

数据字典是 Oracle 数据库最重要的部分之一，它是一组只读表和视图，提供有关其相关数据库的信息。

只要执行数据定义语言 (DDL) 命令，Oracle 服务器就会更新数据字典。此外，数据操纵语言 (DML) 命令（如引起表扩展的命令）也可以更新数据字典。

数据字典不仅是每个 Oracle 数据库最重要的部分，它还是所有用户（从最终用户到应用程序设计者以及数据库管理员）的重要信息来源。

要访问数据字典，请使用 SQL 语句。因为数据库是只读的，所以只能对数据字典的表和视图发出查询。

# 基表和数据字典视图

数据字典包含以下两个部分：

- 基表
  - 存储数据库的说明
  - 使用 `CREATE DATABASE` 命令创建
- 数据字典视图
  - 用于简化基表信息
  - 通过公共同义词访问
  - 使用 `catalog.sql` 脚本创建

ORACLE

5-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## 基表和数据字典视图

数据字典包含对数据库中的对象的说明。它包括两种对象类型。

**基表：**

基表是存储有关数据库的信息的底层表。基表是在任何 Oracle 数据库中首先创建的对象。在使用 `CREATE DATABASE` 创建数据库时，只要 Oracle 服务器运行 `sql.bsq` 脚本，就会自动创建这些对象。只有 Oracle 服务器才能对这些基表执行写入操作。用户很少直接访问基表，因为其中的数据大多数都是以隐含格式存储的。切勿使用 `DML` 命令直接更新基表，但 `AUD$` 表除外。例如，`IND$` 表就是一个基表，它包含有关数据库中的索引的信息。

**数据字典视图：**

数据字典视图是基表的汇总，可以更有效地显示基表信息。例如，在数据字典视图中，除了显示对象编号外还会使用对象名。数据字典视图是在运行 `CREATE DATABASE` 命令之后使用 `catalog.sql` 脚本创建的。

## 创建数据字典视图

| 脚本          | 用途                   |
|-------------|----------------------|
| catalog.sql | 创建常用的数据字典视图和同义词      |
| catproc.sql | 运行服务器端 PL/SQL 所必需的脚本 |

ORACLE

5-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建数据字典视图

创建数据库时，会自动创建数据字典的基表。使用 Oracle Universal Installer 创建数据库时，会自动运行用于创建数据字典和动态性能视图的脚本以及 Oracle 服务器选项的脚本。手动创建新数据库时，需要手动运行这些脚本。此外，将 Oracle 服务器升级为新版本时，可能也需要重新运行这些脚本。只有具有 SYSDBA 权限的用户 SYS 才能运行这些脚本。

以上脚本位于下列目录中：

**UNIX:** \$ORACLE\_HOME/rdbms/admin

**NT:** %ORACLE\_HOME%\rdbms\admin



# 数据字典内容

数据字典提供有关以下方面的信息：

- 逻辑数据库结构和物理数据库结构
- 对象的定义和空间分配
- 完整性约束
- 用户
- 角色
- 权限
- 审计

ORACLE

5-7

Copyright © Oracle Corporation, 2001. All rights reserved.

## 数据字典内容

数据字典包含以下内容：

- 数据库内所有方案对象的定义，这些对象包括表、视图、索引、簇、同义词、序列、过程、函数、程序包、触发器等等
- 已为方案对象分配的空间量以及它们当前使用的空间量
- 列的缺省值
- 完整性约束信息
- Oracle 用户的名称
- 已授予每个用户的权限和角色
- 审计信息，如有哪些人访问或更新了各种方案对象

# 数据字典的使用方式

## 主要用途：

- **Oracle 服务器使用它来查找有关以下内容的信息：**
  - 用户
  - 方案对象
  - 存储结构
- **执行 DDL 语句时，Oracle 服务器会对它进行修改。**
- **用户和 DBA 可将它作为数据库相关信息的只读参考**

ORACLE

5-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## 数据字典的使用方式

### Oracle 服务器使用数据字典的方式：

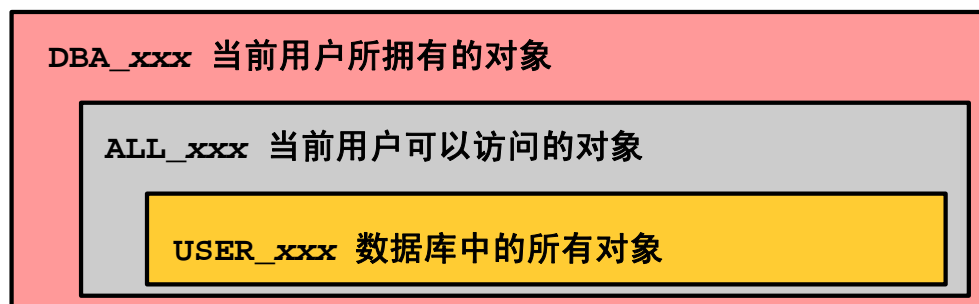
数据字典基表中的数据是 Oracle 服务器运行所必需的。因此，应该只能由 Oracle 服务器来写入或更改数据字典信息。在数据库操作过程中，Oracle 服务器通过读取数据字典来确定方案对象是否存在，以及用户是否对它们具有适当的访问权限。Oracle 服务器还会不断地更新数据字典，及时反映数据库结构方面的变化。

### 用户和数据库管理员使用数据字典的方式：

数据字典的视图可以作为所有数据库用户的参考。某些视图可由所有 Oracle 用户访问；另外一些视图只供数据库管理员使用。

## 数据字典视图的类别

- 三种静态视图集
- 按范围分类为：
  - DBA: 所有方案中的视图
  - ALL: 用户可以访问的视图
  - USER: 用户方案中的视图



ORACLE

5-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 数据字典视图的类别

#### 带有 DBA 前缀的视图：

带有 DBA 前缀的视图显示整个数据库的全局视图。只有数据库管理员才能查询它们。如果用户被授予了系统权限 `SELECT ANY TABLE`，则也可以查询数据字典中前缀为 DBA 的视图。

若要查询数据库中的所有对象，DBA 可以发出下列语句：

```
SELECT owner, object_name, object_type
FROM dba_objects;
```

#### 带有 ALL 前缀的视图：

带有 ALL 前缀的视图指的是从用户角度看到的完整数据库视图。这些视图返回有关方案对象的信息，除了用户所拥有的方案对象外，还包括那些通过公开或显式授予权限和角色的方式让用户获得访问权限的方案对象。

例如，以下查询返回用户具有访问权限的全部对象的信息：

```
SELECT owner, object_name, object_type
FROM all_objects;
```

## 数据字典视图的类别（续）

### 带有 **USER** 前缀的视图：

典型数据库用户最可能感兴趣的视图是带有 **USER** 前缀的视图。

这些视图：

- 涉及数据库中用户自己独有的环境
- 通常涉及当前用户所拥有的对象
- 除 **OWNER** 列暗指当前用户外，具有与其它视图一致的列
- 返回 **ALL** 视图中信息的子集
- 为方便起见，可具有缩写的公共同义词

例如，下列查询返回用户方案中包含的所有对象：

```
SELECT owner, object_name, object_type
FROM users_objects;
```

### 数据字典视图：

数据字典视图是静态视图，从这些视图中用户可以知道：

- 是否曾经创建了某个对象？
- 该对象是什么的一部分？
- 谁拥有该对象？
- 用户具有哪些权限？
- 对该对象有哪些限制？

注：有关全部数据字典视图的列表，请参考 *Oracle9i Database Reference* 文档。

# 数据字典示例

- 总览

- `DICTIONARY`, `DICT_COLUMNS`

- 方案对象

- `DBA_TABLES`, `DBA_INDEXES`, `DBA_TAB_COLUMNS`,  
`DBA_CONSTRAINTS`

- 空间分配

- `DBA_SEGMENTS`, `DBA_EXTENTS`

- 数据库结构

- `DBA_TABLESPACES`, `DBA_DATA_FILES`

ORACLE

5-11

Copyright © Oracle Corporation, 2001. All rights reserved.

## 数据字典示例

要获取数据字典视图的概览，可以查询 `DICTIONARY` 视图或其同义词 `DICT` 视图。  
例如：

```
SELECT * FROM dictionary;
```

用 `where` 子句来缩小查询范围：

```
SELECT * FROM dictionary WHERE table_name LIKE 'dba_seg%'
```

要获得视图中各列的列表，请使用 `DESCRIBE` 关键字：

```
DESCRIBE dba_users;
```

要获得数据字典视图中各列的概览，可以查询 `DICT_COLUMNS` 视图。

要查看数据字典视图的内容，请使用 `SELECT` 命令。

```
SELECT * FROM dba_users;
```

注：有关数据字典视图及其中各列的完整列表，请参考“Oracle9i Database Reference”文档。

# 动态性能表

- 虚拟表
- 记录当前的数据库活动
- 在数据库可操作时不断更新
- 通过内存和控制文件访问信息
- 用于监控和优化数据库
- 由 `sys` 用户拥有
- 同义词以 `v$` 开头
- 在 `V$FIXED_TABLE` 中列出

ORACLE

5-12

Copyright © Oracle Corporation, 2001. All rights reserved.

## 动态性能表

在 Oracle 服务器的整个操作过程中，它将当前数据库活动记录在称为动态性能视图的一组虚拟表中。只有数据库处于运行状态时，这些虚拟表才驻留在内存中，反映数据库操作的实时状况。它们指向内存和控制文件中的实际信息源。

这些表不是真正的表，大多数用户都无法访问它们；但是数据库管理员可以在这些视图上查询、授予 `SELECT` 权限并创建视图。这些视图有时称为固定视图，因为数据库管理员无法更改或删除这些视图。

动态性能表由 `sys` 拥有，它们的名称均以 `v_$` 开头。在这些表上先创建视图，然后再为这些视图创建公共同义词。同义词名以 `v$` 开头。例如，`V$DATAFILE` 视图包含有关数据库中数据文件的信息，而 `V$FIXED_TABLE` 视图包含有关数据库中所有动态性能表和视图的信息。

动态性能表可使用户了解到以下信息：

- 该对象是否处于联机状态并可用？
- 该对象是否已打开？
- 目前持有哪些锁？
- 该会话是否处于活动状态？

## 动态性能表示例

- V\$CONTROLFILE
- V\$DATABASE
- V\$DATAFILE
- V\$INSTANCE
- V\$PARAMETER
- V\$SESSION
- V\$SGA
- \$SPPARAMETER
- V\$TABLESPACE
- V\$THREAD
- V\$VERSION

ORACLE

5-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 动态性能视图示例

要获得动态性能视图的概览，可以查询 DICTIONARY 视图或其同义词 DICT 视图。  
例如：

```
SELECT * FROM dictionary;
```

用 where 子句来缩小查询范围：

```
SELECT * FROM dictionary WHERE table_name like 'V$data%'
```

也可以查询 V\$FIXED\_TABLE 视图获得动态性能视图列表：

```
SELECT * FROM V$FIXED_TABLE;
```

要获得视图中各列的列表，请使用 DESCRIBE 关键字：

```
DESCRIBE V$INSTANCE;
```

要获得动态性能视图中各列的概览，可以查询 DICT\_COLUMNS 视图。

要查看视图的内容，请使用 SELECT 命令。

```
SELECT * from V$INSTANCE;
```

## 动态性能视图示例（续）

示例：

- V\$CONTROLFILE: 列出控制文件的名称
- V\$DATABASE: 包含控制文件中的数据库信息
- V\$DATAFILE: 包含控制文件中的数据文件信息
- V\$INSTANCE: 显示当前例程的状态
- V\$PARAMETER: 列出会话的当前有效参数和值
- V\$SESSION: 列出当前每个会话的会话信息
- V\$SGA: 包含有关系统全局区 (SGA) 的摘要信息
- V\$SPFILE: 列出 SPFILE 的内容
- V\$TABLESPACE: 显示控制文件中的表空间信息
- V\$THREAD: 包含控制文件中的线程信息
- V\$VERSION: Oracle 服务器中核心库组件的版本号

注：有关动态性能视图及其中各列的完整列表，请参考“Oracle9i Database Reference”文档。



## 管理脚本命名约定

| 约定                     | 说明           |
|------------------------|--------------|
| <code>cat*.sql</code>  | 目录和数据字典信息    |
| <code>dbms*.sql</code> | 数据库程序包说明     |
| <code>prvt*.plb</code> | 封装的数据库程序包代码  |
| <code>utl*.sql</code>  | 数据库实用程序的视图和表 |

ORACLE

5-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 管理脚本命名约定

管理脚本可以根据各自的文件名划分为以下几类：

**cat\*.sql：**这些脚本用于创建数据字典视图。除 `catalog.sql` 和 `catproc.sql` 脚本之外，还存在创建有关 Oracle 实用程序信息的其它脚本。例如，`catadt.sql` 脚本用于创建数据字典视图以显示 ORDBMS 中的类型和其它对象特性的元数据信息。`catnoadt.sql` 脚本用于删除这些表和视图。

**dbms\*.sql 和 prvt\*.plb：**这些脚本用于创建预定义的 Oracle 程序包的對象，这些程序包扩展了 Oracle 服务器功能。这些程序简化了管理数据库的任务。大多数 SQL 脚本在 `catproc.sql` 脚本的执行过程中运行。少数其它脚本必须由数据库管理员执行。例如，`dbmspool.sql` 脚本使您可以显示共享池中对象的大小，并在 SGA 中将它们标记为保留或删除，以减少共享池碎片。

**utl\*.sql：**这些脚本必须在数据库需要其它视图和表时运行。例如，脚本 `utlxplan.sql` 创建用于查看 SQL 语句的执行计划的表。

**注：**大多数脚本必须以具有 SYSDBA 权限的 SYS 用户身份运行。数据库管理员应检查脚本以确定必须使用哪个用户帐户来运行脚本。

## 小结

在这一课中，您应该能够掌握：

- 了解内置数据库对象
- 了解数据字典的内容和使用
- 了解数据字典视图的创建方式
- 了解数据字典视图的类别
- 查询数据字典和动态性能视图
- 了解管理脚本命名约定

ORACLE

## 练习 5 概览

此练习涉及以下主题：

- 了解数据字典的组件和内容
- 查询数据字典

ORACLE

5-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 5 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus 工作表 (SQL\*Plus Worksheet) 完成练习。

## 练习 5：使用数据字典和动态性能视图

- 1 下面哪些关于数据字典的描述是正确的？
  - a 数据字典描述了数据库及其对象。
  - b 数据字典包括两种类型的对象：基表和数据字典视图。
  - c 数据字典是表的集合。
  - d 数据字典记录和验证了它所关联的数据库的有关信息。
- 2 基表是使用 `catalog.sql` 脚本创建的。
  - a 对
  - b 错
- 3 下面哪三个关于数据字典用法的描述是正确的？
  - a 执行 DML 语句的时候，Oracle 服务器会对数据字典进行修改。
  - b 数据字典用于查找有关用户、方案对象和存储结构的信息。
  - c 用户和 DBA 将数据字典用作参考。
  - d 数据字典是数据库正常运行的必要组成部分。
- 4 数据字典视图是静态视图。
  - a 对
  - b 错
- 5 动态性能视图的信息是从控制文件收集的。
  - a 对
  - b 错
- 6 动态性能视图有可能使用户了解到下面哪些信息？
  - a 该对象是否处于联机状态并可用？
  - b 目前持有哪些锁？
  - c 谁拥有该对象？
  - d 用户具有哪些权限？
  - e 该会话是否处于活动状态？
- 7 请查找数据字典视图的列表。
- 8 识别数据库名称、例程名和数据库块的大小。

**提示：** 查询动态性能视图 `V$DATABASE`，`V$THREAD`，和 `V$PARAMETER`。
- 9 列出数据文件的名称。

**提示：** 查询动态性能视图 `V$DATAFILE`。
- 10 列出组成 `SYSTEM` 表空间的数据文件。

**提示：** 查询数据字典视图 `DBA_DATA_FILES` 来确定 `SYSTEM` 表空间的数据文件。

## 练习 5：使用数据字典和动态性能视图（续）

**11** 数据库中有多少空闲空间？已占用了多大空间？

**提示**

- 查询数据字典视图 `DBA_FREE_SPACE`，显示数据库中可用的空闲空间量。
- 查询数据字典视图 `DBA_SEGMENTS` 以显示已使用的空间量。

**12** 列出数据库用户的名称和创建日期。

**提示：** 查询数据字典视图 `DBA_USERS`，列出数据库用户的姓名和创建日期。



# 6

## 维护控制文件

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

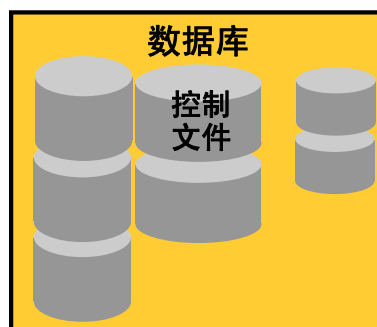
- 解释控制文件的用途
- 列出控制文件的内容
- 对控制文件进行多元备份和管理
- 使用“Oracle 管理文件” (Oracle Managed Files, OMF) 管理控制文件
- 获取控制文件信息

ORACLE



# 控制文件

- 小型二进制文件
- 定义物理数据库的当前状态
- 维护数据库的完整性
- 要求：
  - 在启动数据库时处于 MOUNT 状态
  - 能够操作数据库
- 只链接至一个数据库
- 丢失数据后可能需要恢复
- 最初由 CREATE DATABASE 确定大小



## 控制文件

控制文件是一个小型二进制文件，是成功启动和操作数据库所必需的。每个控制文件只与一个 Oracle 数据库相关联。在打开一个数据库之前，系统将读取控制文件以确定该数据库是否处于有效状态以供使用。

因为 Oracle 服务器在数据库使用的过程中会不断更新控制文件，所以控制文件必须在数据库打开时随时都可供写入。只有 Oracle 服务器才能修改控制文件中的信息；DBA 或最终用户不能编辑控制文件。

如果由于某些原因控制文件无法访问，则数据库将无法正确运行。如果数据库控制文件的所有副本都丢失，则必须先恢复数据库，然后才能将其打开。

## 控制文件（续）

### 确定控制文件的大小：

在创建数据库过程中指定的关键字会影响控制文件的大小。当参数的值较大时，这一点尤其明显。CREATE DATABASE 或 CREATE CONTROLFILE 命令中的以下关键字会影响控制文件的大小：

- MAXLOGFILES
- MAXLOGMEMBERS
- MAXLOGHISTORY
- MAXDATAFILES
- MAXINSTANCES

# 控制文件的内容

控制文件中包含以下条目：

- 数据库名称和标识符
- 创建数据库的时间戳
- 表空间的名称
- 数据文件和重做日志文件的名称和位置
- 当前重做日志的序列号
- 检查点信息
- 还原段的开始和结尾
- 重做日志归档信息
- 备份信息

ORACLE

6-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## 控制文件的内容

控制文件中的信息包括下列内容：

- 数据库名称取自初始化参数 DB\_NAME 所指定的名称或 CREATE DATABASE 语句中所用的名称。
- 当创建数据库时会记录数据库标识符。
- 创建数据库时还会记录创建数据库的时间戳。
- 当在数据库中添加、重命名或删除数据文件或重做日志时，会更新相关数据文件和联机重做日志文件的名称和位置。
- 当添加或删除表空间时会更新表空间信息。
- 在日志切换过程中会记录重做日志历史信息。
- 归档日志的位置和状态会在归档时记录。
- 备份的位置和状态由“恢复管理器”(Recovery Manager) 实用程序记录。
- 在进行日志切换时记录当前日志序列号。
- 在建立检查点时记录检查点信息。

## 控制文件的内容（续）

控制文件由以下两种类型的部分组成：

- 可重用
- 不可重用

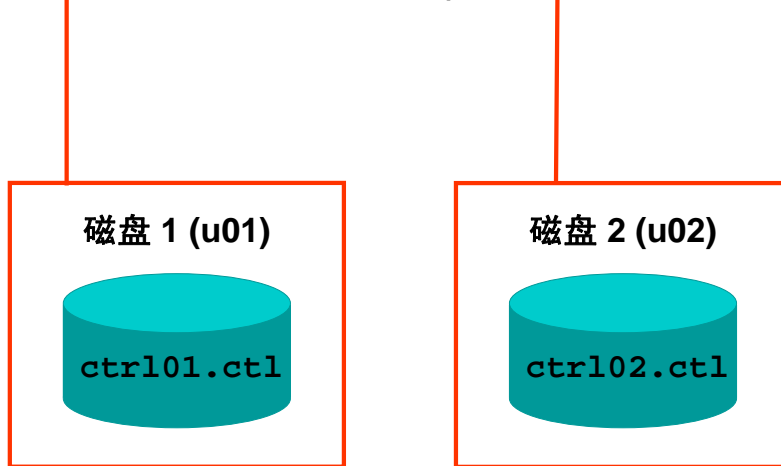
可重用部分存储“恢复管理器”(Recovery Manager)的信息，如备份数据文件名和备份重做日志文件名。只有“恢复管理器”(Recovery Manager)才能以循环方式重新使用这些部分。

**注：***Oracle9i 数据库管理基础 II* 课程中详细介绍了“恢复管理器”(Recovery Manager)。

## 对控制文件进行多元备份

`CONTROL_FILES=`

`$HOME/ORADATA/u01/ctrl01.ctl, $HOME/ORADATA/u02/ctrl02.ctl`



ORACLE

6-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### 对控制文件进行多元备份

要避免控制文件的单点故障所带来的损失，我们强烈建议您对控制文件进行多元备份，即在不同的物理磁盘上存储一个副本。如果某个控制文件丢失，可以使用控制文件的多元备份副本重新启动例程，而不必恢复数据库。

通过以下方式，最多可以对控制文件进行八次多元备份：

- 创建数据库时，通过在初始化参数文件中指定控制文件的名称和完整路径来创建多个控制文件：

```
CONTROL_FILES=$HOME/ORADATA/u01/ctrl01.ctl,
$HOME/ORADATA/u02/ctrl02.ctl
```

- 创建数据库后添加控制文件

#### 备份控制文件：

由于控制文件记录数据库的物理结构，所以在更改数据库的物理结构后应当立即备份控制文件。在 *Oracle9i 数据库管理基础 II* 课程中详细介绍了控制文件的备份和恢复。

## 使用 SPFILE 时对控制文件 进行多元备份

### 1. 改变 SPFILE:

```
ALTER SYSTEM SET control_files =
'$HOME/ORADATA/u01/ctrl01.ctl',
'$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
```

### 2. 关闭数据库:

```
shutdown immediate
```

### 3. 创建控制文件副本:

```
cp $HOME/ORADATA/u01/ctrl01.ctl
 $HOME/ORADATA/u02/ctrl02.ctl
```

### 4. 启动数据库:

```
startup
```

ORACLE

6-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## 使用 SPFILE 时对控制文件进行多元备份

1. **改变 SPFILE:** 使用 ALTER SYSTEM SET 命令改变 SPFILE，使其包括要用的所有控制文件的列表：主控制文件和多元备份副本。
2. **关闭数据库:** 关闭数据库，以便在操作系统上创建控制文件副本。
3. **创建控制文件副本:** 使用操作系统复制命令，根据需要创建控制文件副本并验证副本已存在于相应目录中。
4. **启动数据库:** 启动数据库时，系统将读取 SPFILE，同时 Oracle 服务器将维护 CONTROL\_FILES 参数中列出的所有控制文件。

## 使用 PFILE 时对控制文件 进行多元备份

### 1. 关闭数据库:

```
shutdown immediate
```

### 2. 创建控制文件副本:

```
cp $HOME/ORADATA/u01/ctrl01.ctl
$HOME/ORADATA/u02/ctrl02.ctl
```

### 3. 向 PFILE 添加控制文件名:

```
CONTROL_FILES = (/DISK1/control01.ctl,
/DISK3/control02.ctl)
```

### 4. 启动数据库:

```
startup
```

ORACLE

6-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## 使用 PFILE 时对控制文件进行多元备份

1. 关闭数据库: 关闭数据库, 以便在操作系统上创建控制文件副本。
2. 创建控制文件副本: 使用操作系统复制命令, 根据需要创建控制文件副本并验证副本已存在于相应目录中。
3. 向 PFILE 添加控制文件名: 改变 PFILE, 使其包括所有控制文件的列表。
4. 启动数据库: 启动数据库时, 系统将读取 PFILE, 同时 Oracle 服务器将维护 CONTROL\_FILES 参数中列出的所有控制文件。

## 使用 OMF 管理控制文件

- 如果未指定 `CONTROL_FILES` 参数，则创建 OMF
- 位置由 `DB_CREATE_ONLINE_LOG_DEST_n` 指定
- 生成的名称是唯一的并在 `alertSID.log` 中显示

ORACLE

6-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 OMF 管理控制文件

如果在初始化参数文件中未指定 `CONTROL_FILES` 参数，则在创建数据库时将自动把控制文件创建为 OMF。

如果使用 `init.ora` 文件，必须将 `CONTROL_FILES` 参数设置为 OMF 生成的名称，这些名称可在 `V$CONTROLFILE` 或 `alertSID.log` 中找到。如果使用 `SPFILE`，创建数据库时将自动设置和保存 `CONTROL_FILES` 参数。

控制文件的位置由 `DB_CREATE_ONLINE_LOG_DEST_n` 参数确定。如果未设置这个参数，控制文件将位于 `DB_CREATE_FILE_DEST` 参数定义的位置。如果未设置这两个参数，控制文件将不属于 OMF。如果控制文件不属于 OMF，将需要在初始化参数文件中设置 `CONTROL_FILES` 参数，否则将会收到错误消息。

创建文件时，将生成唯一的控制文件名称 (`ora_cmr7t30p.ctl`)，并在 `lertSID.log` 中显示。



## 获取控制文件信息

有关控制文件状态和位置的信息，可以通过查询以下视图来检索。

- **V\$CONTROLFILE**: 列出与该例程相关联的所有控制文件的名称和状态
- **V\$PARAMETER**: 列出所有参数的状态和位置
- **V\$CONTROLFILE\_RECORD\_SECTION**: 提供有关控制文件记录部分的信息
- **SHOW PARAMETER CONTROL\_FILES**: 列出控制文件的名称、状态和位置

ORACLE

6-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取控制文件信息

要获取控制文件的位置和名称，请查询 V\$CONTROLFILE 视图。

```
SELECT name FROM V$CONTROLFILE;

NAME

/u01/home/db03/ORADATA/u01/ctrl01.ctl
/u01/home/db03/ORADATA/u01/ctrl01.ctl
2 rows selected.
```

也可以使用 V\$PARAMETER 视图。

```
SELECT name, value from V$PARAMETER
WHERE name = 'control_files';

NAME Value

control_files /u01/home/db03/ORADATA/u01/ctrl01.ctl
```

获取控制文件信息（续）

要获取有关控制文件不同部分的信息，请查询 V\$CONTROLFILE\_RECORD\_SECTION 视图。

```
SQL> SELECT type, record_size, records_total, records_used
2 FROM v$controlfile_record_section
3 WHERE TYPE='DATAFILE';
```

| TYPE     | RECORD_SIZE | RECORDS_TOTAL | RECORDS_USED |
|----------|-------------|---------------|--------------|
| DATAFILE | 180         | 40            | 10           |

1 row selected.

RECORDS\_TOTAL 列指定分配给某特定部分的记录数。例如，在以上示例中，您可以看到数据文件的最大数为 30，这个数字由 CREATE DATABASE 命令中的 AXDATAFILES 参数确定。

也可以使用 SHOW PARAMETER 命令找到控制文件的位置。

```
SQL> SHOW PARAMETER control_files;
NAME TYPE VALUE

control_files string $HOME/ORADATA/u01/ctrl01.ctl,
 $HOME/ORADATA/u02/ctrl02.ctl
```

几个动态性能视图中的信息可以从控制文件中获得。下面是一些示例：

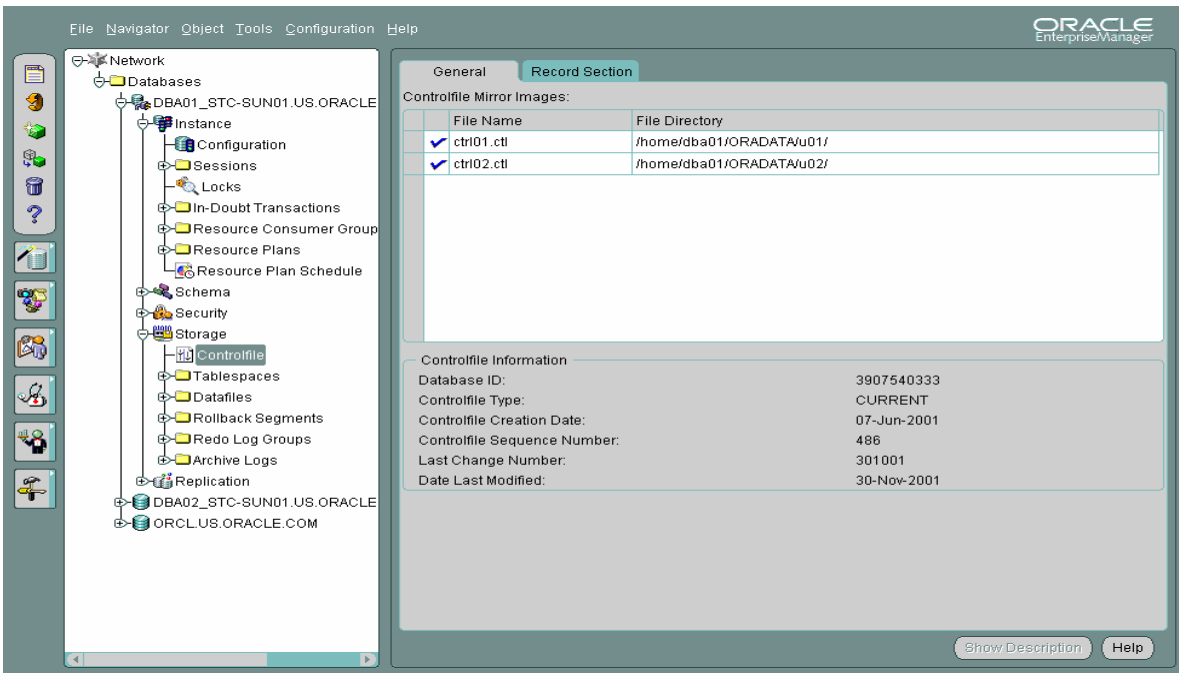
- V\$BACKUP
- V\$DATAFILE
- V\$TEMPFILE
- V\$TABLESPACE
- V\$ARCHIVE
- V\$LOG
- V\$LOGFILE
- V\$LOGHIST
- V\$ARCHIVED\_LOG
- V\$DATABASE

获取控制文件信息（续）

使用 Oracle Enterprise Manager 查看有关控制文件的信息

从 OEM 控制台：

- 1. 导航到 “数据库” (Databases) > “存储” (Storage)。
- 2. 单击 “控制文件” (Controlfile)。
- 3. 在 “常规” (General) 选项卡上，单击 “全部初始化参数” (All Initialization Parameters)。
- 4. 在参数值栏中修改参数。
- 5. 单击 “确定” (OK)。



## 小结

在这一课中，您应该能够掌握：

- 使用 SPFILE 对控制文件进行多元备份
- 使用 init.ora 对控制文件进行多元备份
- 使用 OMF 管理控制文件

ORACLE

## 练习 6 概览

此练习涉及以下主题：

- 在没有控制文件的情况下启动数据库
- 对现有控制文件进行多元备份

ORACLE

6-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 6 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 6：管理控制文件

### 1 现有控制文件的位置及其名称是什么？

**提示：** 查询动态性能视图 V\$CONTROLFILE。**注：** 您还可以使用 V\$PARAMETER，或者执行 SHOW PARAMETER 命令以显示控制文件的名称和位置。

### 2 尝试在没有任何控制文件的情况下启动数据库。通过更改参数文件中的控制文件名或只是更改控制文件名，便可模仿此操作。结果如何？

### 3 使用目录 u02 和新控制文件 ctrl02.ctl 对现有的控制文件进行多元备份。确保 Oracle 服务器可以写入新的控制文件。例如，在 UNIX 上使用 chmod 660 命令。确认两个控制文件都在使用。

**提示：**

- 关闭数据库前，改变 SPFILE (SCOPE=SPFILE) 以将新的控制文件添加到初始化文件中。
- 关闭数据库，将现有的控制文件复制到目录 u02 中并将名称更改为 ctrl02.ctl。在 UNIX 上使用 chmod 660 命令。通常，文件的权限不会发生更改，这种情况适合于教室环境。
- 启动数据库。
- 查询动态性能视图 V\$CONTROLFILE 或 V\$PARAMETER，或使用 SHOW PARAMETER 命令确认两个控制文件都在使用。

### 4 控制文件中数据文件部分的初始大小是多少？

**提示：** 查询动态性能视图 V\$CONTROLFILE\_RECORD\_SECTION。



# 维护重做日志文件

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 解释联机重做日志文件的用途
- 概述联机重做日志文件的结构
- 控制日志切换和检查点
- 对联机重做日志文件进行多元备份和维护
- 使用 **OMF** 管理联机重做日志文件

ORACLE



# 使用重做日志文件

重做日志文件具有以下特征：

- 记录对数据所做的所有更改
- 提供恢复机制
- 可以划分成组
- 至少需要两个组



ORACLE

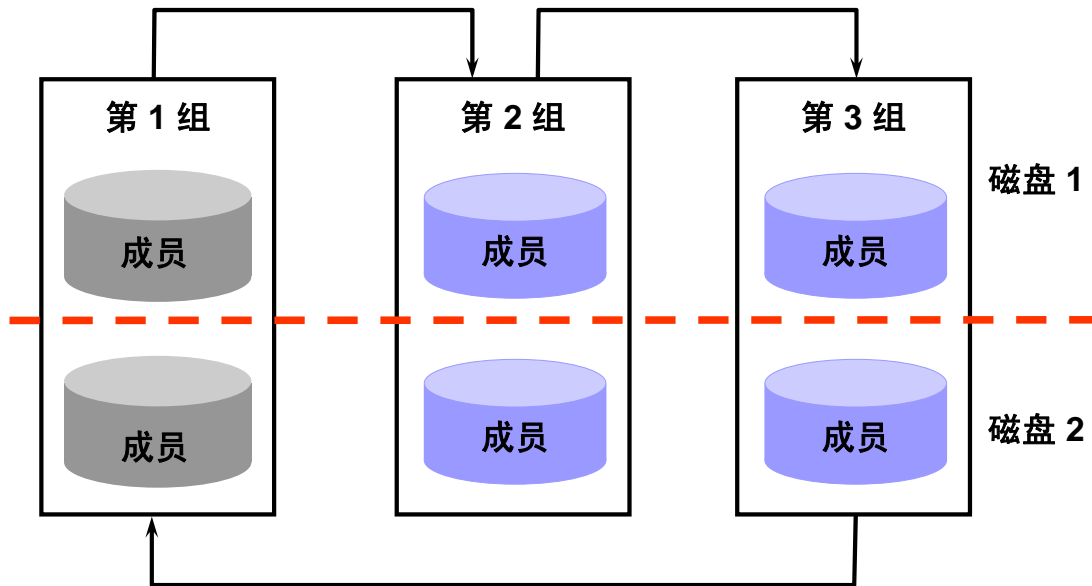
7-3

Copyright © Oracle Corporation, 2001. All rights reserved.

## 使用重做日志文件

利用重做日志文件，在数据库发生故障时，可以重新处理事务。每个事务在处理的同时也会写入重做日志缓冲区，然后刷新到重做日志文件，这样，如果发生介质故障，重做日志文件将提供恢复机制。（但也存在例外情况，例如，在启用 NOLOGGING 子句的情况下对象中的直接加载插入。）写入的信息包括尚未提交的事务处理、还原段信息以及方案和对象管理语句。重做日志文件用来在例程失败等情况下恢复尚未写入数据文件的提交数据。重做日志文件只用于恢复。

## 重做日志文件的结构



ORACLE

7-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### 重做日志文件的结构

数据库管理员可设置 Oracle 数据库以维护联机重做日志文件副本，来避免由于单点故障丢失数据库信息。

#### 联机重做日志文件组：

- 一组相同的联机重做日志文件副本称作联机重做日志组。
- LGWR 后台进程向组内所有联机重做日志文件并发写入相同信息。
- 为保证数据库的正常操作，Oracle 服务器最少需要两个联机重做日志文件组。

#### 联机重做日志文件成员：

- 组内的每个联机重做日志文件称为成员。
- 组内的每个成员都有相同的日志序列号和同样的大小。Oracle 服务器每次写入日志组时，都分配一个日志序列号以唯一地识别每个重做日志文件。当前日志序列号存储在控制文件和所有数据文件的标头内。

## 重做日志文件的结构（续）

### 创建初始重做日志文件：

联机重做日志文件组和成员的初始集是在数据库创建时创建的。

下面的参数限制了联机重做日志文件的数量：

- CREATE DATABASE 命令中的 MAXLOGFILES 参数指定联机重做日志文件组的绝对最大数量。
- MAXLOGFILES 的最大值和缺省值取决于您的操作系统。
- CREATE DATABASE 命令所使用的 MAXLOGMEMBERS 参数决定每个组的成员的最大数量。MAXLOGMEMBERS 的最大值和缺省值取决于您的操作系统。

## 重做日志文件如何发挥作用

- 重做日志文件是以循环方式使用的。
- 一旦某个重做日志文件被写满，LGWR 就会移动到下一个日志组。
  - 这称为日志切换
  - 同时还将执行检查点操作
  - 将信息写入控制文件

ORACLE

7-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 重做日志文件如何发挥作用

Oracle 服务器将对数据库所做的所有更改按顺序记录到重做日志缓冲区中。LGWR 进程把重做条目从重做日志缓冲区写入联机重做日志组的其中一个组，这个组叫做当前联机重做日志组。LGWR 进程将在以下情况下写入：

- 当提交事务处理时
- 当重做日志缓冲区被写满三分之一时
- 当重做日志缓冲区内的已更改记录超过 1 MB 时
- 在 DBWn 将数据库缓冲区高速缓存中修改的块写入数据文件之前

重做日志文件是以循环方式使用的。每个重做日志文件组用一个日志序列号来标识，每次重新使用日志时就会覆盖原来的序列号。

#### 日志切换：

LGWR 按顺序向联机重做日志文件写入重做信息。一旦当前联机重做日志文件组被写满，LGWR 就开始写入下一个组。这称为日志切换。

当最后一个可用联机重做日志文件已满时，LGWR 将返回第一个联机重做日志文件组并开始重新写入。

## 重做日志文件如何发挥作用（续）

### 检查点：

在检查点期间：

- 大量的灰数据库缓冲区数据（由正在经历检查点事件的日志所覆盖）被 DBWn 写入到数据文件中。DBWn 写入的缓冲区的数量是由参数 FAST\_START\_MTTR\_TARGET 决定的（如果已指定）。缺省值为零。

注：Oracle9i 数据库管理基础 II 课程中详细介绍了 FAST\_START\_MTTR\_TARGET 参数。

- 检查点后台进程 CKPT 更新控制文件以反映该进程已成功完成。如果检查点是由日志切换引起的，CKPT 还会更新数据文件的标头。

可以针对数据库中的所有数据文件执行或者只针对特定数据文件执行检查点操作。

例如，检查点可发生在下面情况中：

- 每次日志切换时
- 当已通过正常、事务处理或者立即选项关闭例程时
- 通过设置初始化参数 FAST\_START\_MTTR\_TARGET 强制执行时
- 数据库管理员通过手动方式请求时
- ALTER TABLESPACE [OFFLINE NORMAL|READ ONLY|BEGIN BACKUP] 命令导致对特定数据文件执行检查点操作时

如果初始化参数 LOG\_CHECKPOINTS\_TO\_ALERT 设置为 TRUE，则有关每个检查点的信息都记录在 alert\_SID.log 文件内。该参数缺省值为 FALSE，表示不记录检查点。

## 强制执行日志切换和检查点

- 强制执行日志切换:

```
ALTER SYSTEM SWITCH LOGFILE;
```

- 可使用以下方式强制执行检查点操作:

- 设置 FAST\_START\_MTTR\_TARGET 参数

```
FAST_START_MTTR_TARGET = 600
```

- ALTER SYSTEM CHECKPOINT 命令

```
ALTER SYSTEM CHECKPOINT;
```

ORACLE

7-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### 强制执行日志切换和检查点

如前所述，日志切换和检查点操作是在数据库运行中的某些特定点自动执行的，但 DBA 可以强制执行日志切换或检查点操作。

#### 强制执行检查点:

FAST\_START\_MTTR\_TARGET 参数取代了不赞成使用的参数:

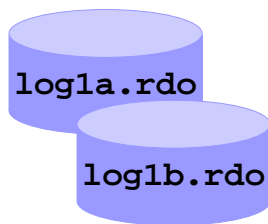
- FAST\_START\_IO\_TARGET
- LOG\_CHECKPOINT\_TIMEOUT

如果使用参数 FAST\_START\_MTTR\_TARGET，就不能再使用这两个不赞成使用的参数。

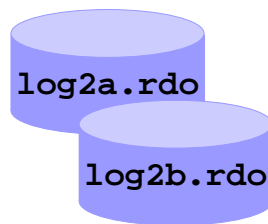
在上面的示例中，已设置了 FAST\_START\_MTTR\_TARGET 参数，因此例程恢复所用的时间不应超过 600 秒。数据库将根据这一目标来调整其它参数。

## 添加联机重做日志文件组

```
ALTER DATABASE ADD LOGFILE GROUP 3
('$HOME/ORADATA/u01/log3a.rdo',
 '$HOME/ORADATA/u02/log3b.rdo')
SIZE 1M;
```



第 1 组



第 2 组



第 3 组

ORACLE

7-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 添加联机重做日志文件组

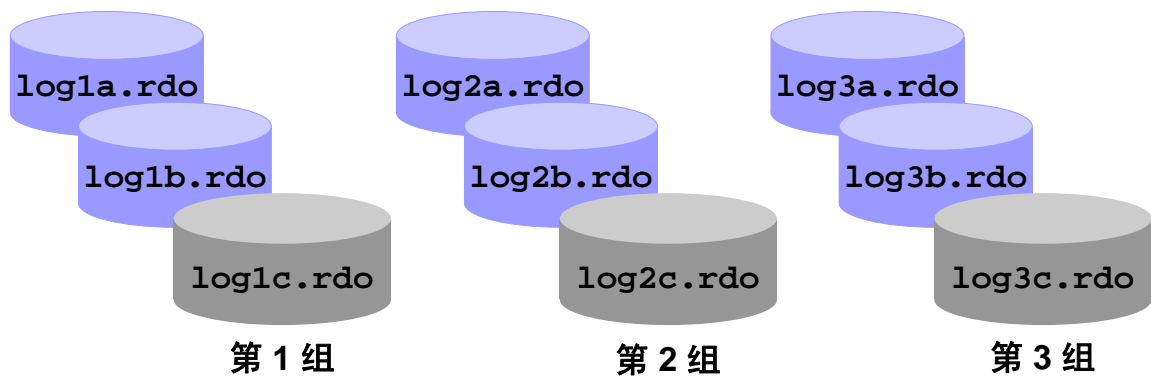
在某些情况下，您可能需要创建其它日志文件组。例如，添加组可以解决可用性问题。要创建一个新的联机重做日志文件组，请使用下面的 SQL 命令：

```
ALTER DATABASE [database]
ADD LOGFILE [GROUP integer] filespec
[, [GROUP integer] filespec]...
```

您可以通过文件说明来指定成员名称和位置。可以选择每个重做日志文件组的 GROUP 参数值。如果您省略了该参数，Oracle 服务器自动生成其值。

## 添加联机重做日志文件成员

```
ALTER DATABASE ADD LOGFILE MEMBER
'$HOME/ORADATA/u04/log1c.rdo' TO GROUP 1,
'$HOME/ORADATA/u04/log2c.rdo' TO GROUP 2,
'$HOME/ORADATA/u04/log3c.rdo' TO GROUP 3;
```



ORACLE

7-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 添加联机重做日志文件成员

您可以使用下面的 ALTER DATABASE ADD LOGFILE MEMBER 命令向现有的重做日志文件组添加新成员：

```
ALTER DATABASE [database]
ADD LOGFILE MEMBER
['filename' [REUSE]
[, 'filename' [REUSE]]...
TO {GROUP integer
| ('filename'[, 'filename']...)
}
]...
```

请使用日志文件成员的完全指定名；否则将在数据库服务器缺省目录下创建该文件。

如果该文件已经存在，其大小必须与指定值相同，并且必须指定 REUSE 选项。您可以通过指定一个或多个组内成员或者指定组号来识别目标组。

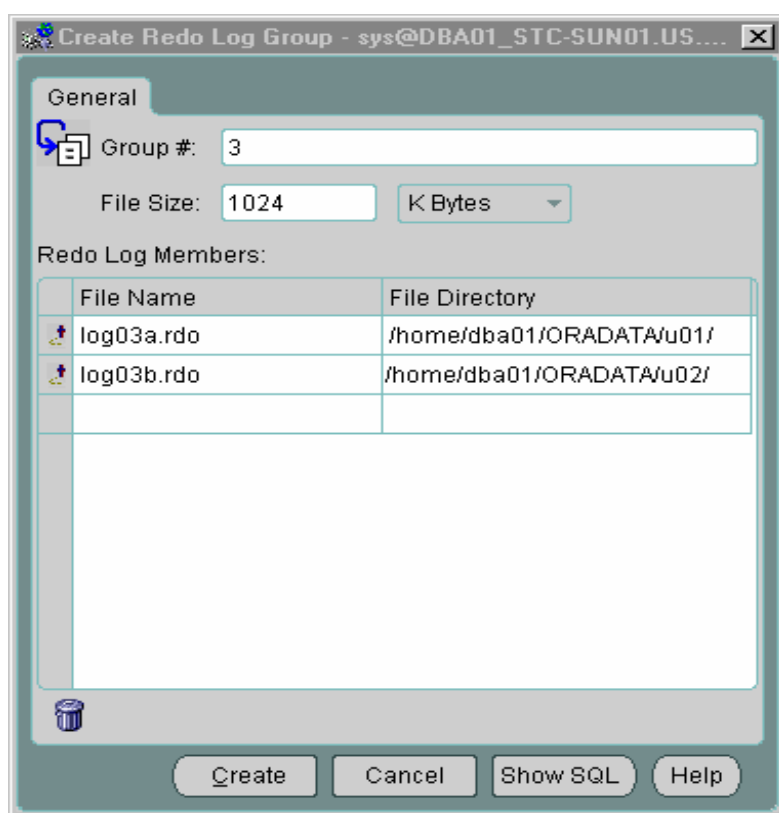


## 添加联机重做日志文件成员（续）

使用 **Oracle Enterprise Manager** 添加重做日志文件组和成员

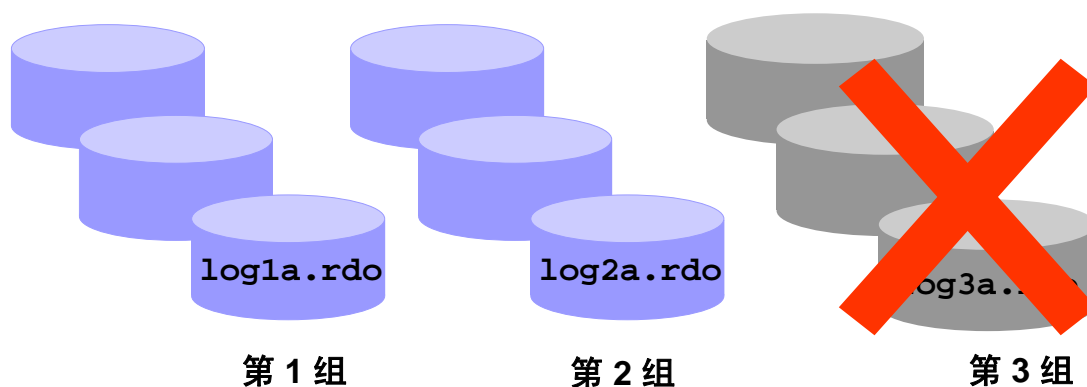
从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage)。
2. 单击“重做日志组”(Redo Log Groups) 文件夹。
3. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
4. 在“常规”(General) 选项卡中，填写创建重做日志文件组和成员所需的信息。
5. 单击“创建”(Create)。



## 删除联机重做日志文件组

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```



ORACLE

7-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 删除联机重做日志文件组

若要增大或者减小联机重做日志文件组的大小，请添加新的联机重做日志文件组（具有新的大小），然后删除旧组。

可以使用下面的 ALTER DATABASE DROP LOGFILE 命令删除整个联机重做日志文件组：

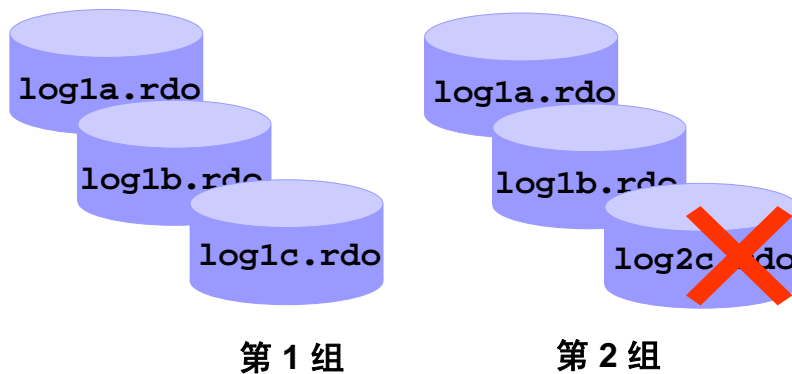
```
ALTER DATABASE [database]
DROP LOGFILE {GROUP integer|('filename'[, 'filename']...)}
 [, {GROUP integer|('filename'[,
 'filename']...)}]...
```

#### 限制：

- 一个例程至少需要两组联机重做日志文件。
- 无法删除活动组或者当前组。
- 删除联机重做日志文件组时并不删除操作系统文件。

## 删除联机重做日志文件成员

```
ALTER DATABASE DROP LOGFILE MEMBER
'$HOME/ORADATA/u04/log3c.rdo';
```



ORACLE

7-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 删除重做日志文件成员

如果联机重做日志文件成员无效，则最好删除它。如果要删除一个或多个特定的联机重做日志文件成员，请使用下面的 ALTER DATABASE DROP LOGFILE MEMBER 命令：

```
ALTER DATABASE [database]
DROP LOGFILE MEMBER 'filename'[, 'filename']...
```

#### 限制：

- 如果要删除的是组内的最后一个有效成员，那么您不能删除该成员。
- 如果该组是当前组，那么必须先强制执行日志文件切换，然后才能删除该成员。
- 如果数据库正运行在 ARCHIVELOG 模式下并且未将该成员所属日志文件组归档，那么您无法删除该成员。
- 删除联机重做日志文件成员时，如果没有使用 OMF 功能，则不会删除操作系统文件。

## 使用“存储管理器”(Storage Manager)删除重做日志文件组和成员

### 使用 Oracle Enterprise Manager 删除重做日志文件组和成员：

从“OEM 控制台”(OEM Console)：

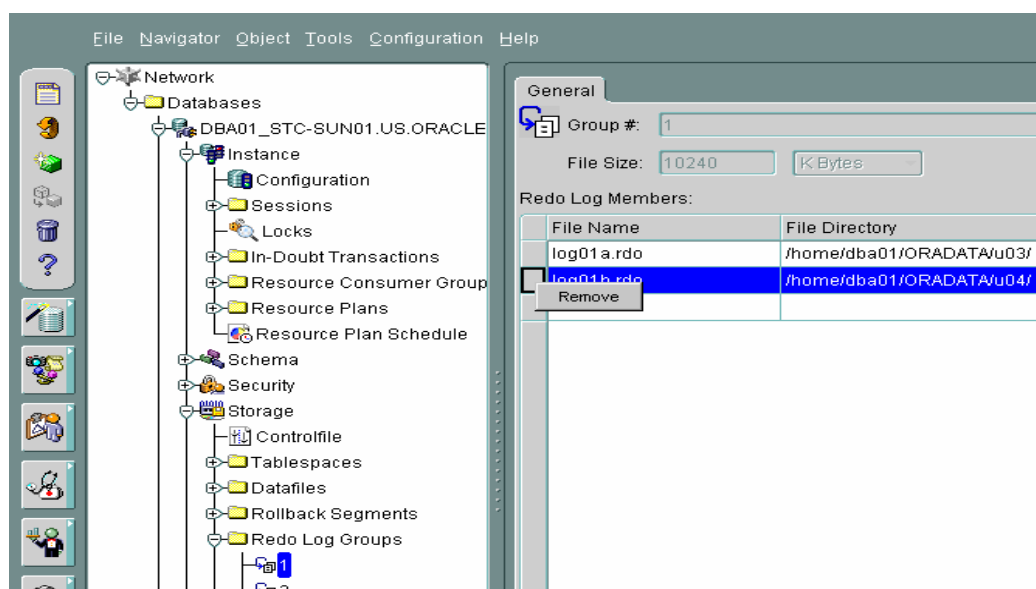
1. 导航到“数据库”(Databases) > “存储”(Storage)。

要删除组，请执行以下操作：

1. 展开“重做日志组”(Redo Log Groups) 文件夹，然后选择要删除的重做日志文件组。
2. 单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
3. 确认删除。

要删除成员，请执行以下操作：

1. 展开“重做日志组”(Redo Log Groups) 文件夹，然后导航到包含要删除的成员的组。
2. 在“常规”(General) 页中，突出显示该成员，然后单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
3. 确认删除。



## 重定位或重命名 联机重做日志文件

使用以下两种方法之一重定位或重命名联机重做日志文件：

- **ALTER DATABASE CLEAR LOGFILE 命令**
  - 将联机重做日志文件复制到新的位置
  - 执行该命令

```
ALTER DATABASE CLEAR LOGFILE
'$HOME/ORADATA/u01/log2a.rdo';
```

- 添加新成员并删除旧成员

ORACLE

7-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 重定位或重命名联机重做日志文件

可以通过重命名联机重做日志文件来更改联机重做日志文件的位置。在重命名联机重做日志文件之前，请确保新的联机重做日志文件已存在。Oracle 服务器仅更改控制文件内的指针，并不从物理上重命名或创建任何操作系统文件。

下面的 ALTER DATABASE RENAME FILE 命令可更改联机重做日志文件的名称：

```
SQL> ALTER DATABASE [database]
2 RENAME FILE 'filename' [, 'filename']...
3 TO 'filename']...
```

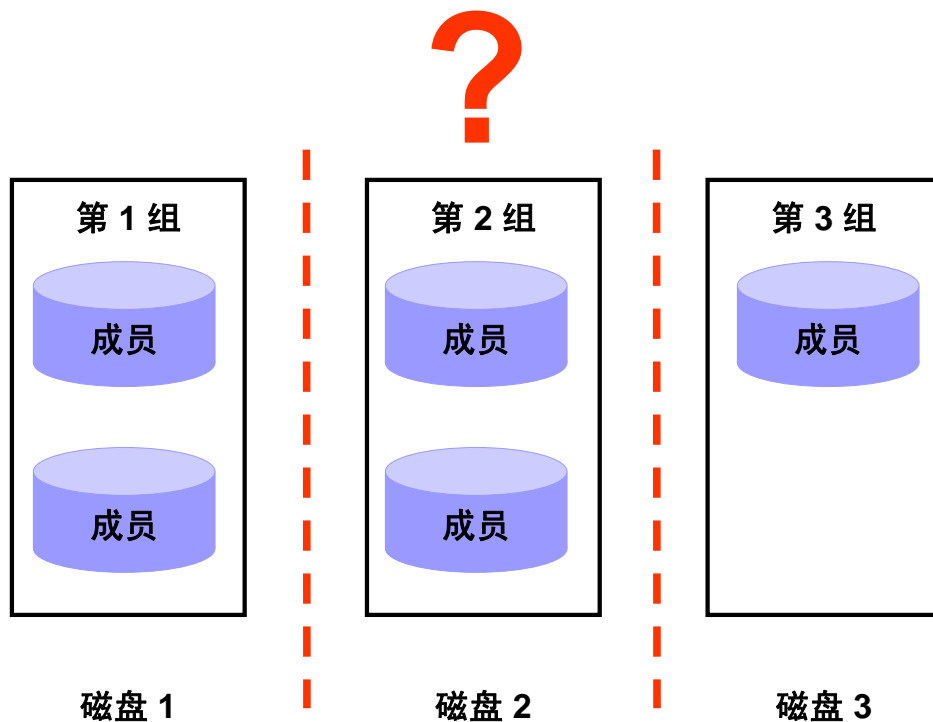
## 使用“存储管理器”(Storage Manager) 重定位或重命名重做日志文件成员

### 使用 Oracle Enterprise 重定位或重命名重做日志文件组和成员

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “重做日志组”(Redo Log Groups)。
2. 选择一个重做日志文件组。
3. 修改重做日志文件成员的“文件名”(File Name) 或“文件目录”(File Directory) 以重定位或重命名该成员。
4. 单击“应用”(Apply)。

## 联机重做日志文件的配置



ORACLE

7-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### 联机重做日志文件的配置

要确定一个数据库例程的联机重做日志文件的合适数量，您必须测试不同的配置。

在某些情况下，数据库例程可能只需要两个组。在其它情况下，数据库例程可能需要更多的组以保证各个组始终可供 LGWR 使用。例如，如果 LGWR 跟踪文件或警报文件中的消息表明 LGWR 经常不得不因为检查点操作尚未完成或者组尚未归档而等待，您就需要添加组。

尽管 Oracle 服务器允许多元备份的组可以包含不同数量的成员，但应该尽量建立对称配置。不对称配置应只是非常情况（如磁盘故障）的临时结果。

#### 联机重做日志文件的位置：

对联机重做日志文件进行多元备份时，请将组内的成员放置在不同磁盘上。这样，即使一个成员不可用而其它成员可用，该例程也不会关闭。

将归档日志文件和联机重做日志文件分放在不同磁盘上，以减少 ARCn 和 LGWR 后台进程之间的争用。

## 联机重做日志文件的配置（续）

数据文件和联机重做日志文件应当放置在不同的磁盘上以减少 LGWR 和 DBWn 的争用，并降低发生介质故障时同时丢失数据文件和联机重做日志文件的风险。

### 调整联机重做日志文件的大小：

联机重做日志文件最小为 50 KB，最大文件大小视操作系统而定。不同组的成员可以有不同的大小；但是，大小不同的组不会带来任何好处。

只有当需要更改联机重做日志组的成员大小时，才需要大小不同的组作为临时结果。在这种情况下，必须先创建新的大小不同的联机重做日志文件组，然后删除旧组。

下面的情况可能影响联机重做日志文件的配置：

- 日志切换和检查点的数量
- 重做条目的量和个数
- 存储介质的空间量；例如，启用归档时磁带上的空间量



## 使用 OMF 管理联机重做日志文件

- 定义 DB\_CREATE\_ONLINE\_LOG\_DEST\_n 参数:

```
DB_CREATE_ONLINE_LOG_DEST_1
DB_CREATE_ONLINE_LOG_DEST_2
```

- 可以在没有任何文件说明的情况下添加组:

```
ALTER DATABASE ADD LOGFILE;
```

- 删除组:

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

ORACLE

7-19

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 OMF 管理联机重做日志文件

**定义 DB\_CREATE\_ONLINE\_LOG\_DEST\_n 参数:** 要创建由 OMF 管理的联机重做日志文件, 必须定义 DB\_CREATE\_ONLINE\_LOG\_DEST\_n 参数。必须给每个多元备份副本 (由 *n* 值指定) 设置该参数。

在上面的示例中, 创建了两个组, 每个组分别包含两个成员。组名称是自动生成的 (如 ora\_1\_wo94n2xi.log) 并显示在 alertSID.log 中。缺省大小为 100 MB。要创建新的联机重做日志文件组, DBA 应当使用 ALTER DATABASE ADD LOGFILE 命令。该命令已经过修改, 不再需要文件说明。

本幻灯片中显示了以下示例: 添加了一个日志文件, 该日志文件有两个成员, 其中一个成员位于 DB\_CREATE\_ONLINE\_LOG\_DEST\_1 定义的位置, 另一个成员位于 DB\_CREATE\_ONLINE\_LOG\_DEST\_2。日志文件成员的唯一文件名是自动生成的, 并且显示在 alertSID.log 中。缺省大小为 100 MB。

#### 删除组:

在上面的示例中, 删除了第 3 个日志文件组以及与第 3 组中每个 OMF 日志文件成员关联的操作系统文件。

#### 归档的重做日志文件和 OMF:

归档的重做日志文件不能是 OMF。

## 获取组和成员的信息

可通过查询以下视图来获取有关组及其成员的信息：

- V\$LOG
- V\$LOGFILE

ORACLE

7-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取组和成员的信息

#### V\$LOG 视图：

下面的查询返回控制文件中关于联机重做日志文件的信息：

```
SQL> SELECT group#, sequence#, bytes, members, status
2 FROM v$log;
```

| GROUP# |     | SEQUENCE# | BYTES | MEMBERS  | STATUS |
|--------|-----|-----------|-------|----------|--------|
| 1      | 688 | 1048576   | 1     | CURRENT  |        |
| 2      | 689 | 1048576   | 1     | INACTIVE |        |

2 rows selected.

下面的项是 STATUS 列的常见值：

- UNUSED：表示从未对联机重做日志文件组进行写入。这是刚添加的联机重做日志文件的状态。
- CURRENT：表示当前的联机重做日志文件组。这说明该联机重做日志文件组是活动的。
- ACTIVE：表示联机重做日志文件组是活动的，但是并非当前联机重做日志文件组。崩溃恢复需要该状态。它可用于块恢复。它可能已归档，也可能未归档。

## 获取组和成员的信息（续）

- **CLEARING**: 表示在执行 `ALTER DATABASE CLEAR LOGFILE` 命令后正在将该日志重建为一个空日志。日志清除后，其状态更改为 **UNUSED**。
- **CLEARING\_CURRENT**: 表示正在清除当前日志文件中的已关闭线程。如果切换时发生某些故障，如写入新日志标头时发生了输入/输出 (I/O) 错误，则日志可能处于此状态。
- **INACTIVE**: 表示例程恢复不再需要联机重做文件日志组。它可能已归档，也可能未归档。

### **V\$LOGFILE** 视图:

若要获取组内所有成员的名称，请查询 **V\$LOGFILE** 视图。

```
SQL> SELECT member FROM V$LOGFILE;
```

MEMBER

-----  
/u01/home/db03/ORADATA/u03/log02a.rdo

/u01/home/db03/ORADATA/u03/log01a.rdo

**STATUS** 列的值可以为下列之一:

- **INVALID**: 表明该文件不可访问
- **STALE**: 表示文件内容不完全
- **DELETED**: 表明该文件已不再使用
- 空白表明文件正在使用中

## 归档的重做日志文件

- 已满的联机重做日志文件可以归档。
- 在 ARCHIVELOG 模式下运行数据库并对重做日志文件进行归档有两个好处：
  - 恢复：数据库备份连同联机重做日志文件和归档重做日志文件可共同确保恢复所有已提交的事务处理。
  - 备份：可在数据库打开时执行备份。
- 缺省情况下，数据库是在 NOARCHIVELOG 模式下创建的。

ORACLE

7-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### 归档的重做日志文件

数据库管理员 (DBA) 必须做出的一个重要决策是：将数据库配置为在 ARCHIVELOG 模式下还是在 NOARCHIVELOG 模式下操作。

#### **NOARCHIVELOG 模式：**

在 NOARCHIVELOG 模式下，每次联机重做日志文件已满并发生日志切换时，都要覆盖联机重做日志文件。直到对重做日志文件组的检查点操作完成后，LGWR 才覆盖该重做日志文件组。

#### **ARCHIVELOG 模式：**

如果数据库配置为在 ARCHIVELOG 模式运行下，那么必须将已满的联机重做日志文件的不活动组归档。因为对数据库所做的所有更改都记录在联机重做日志文件内，数据库管理员可以使用物理备份和归档的联机重做日志文件恢复数据库，而不会丢失任何已提交数据。

归档联机重做日志文件有两种方法：

- 手动
- 自动（建议采用的方法）

## 归档的重做日志文件（续）

### ARCHIVELOG 模式（续）：

LOG\_ARCHIVE\_START 初始化参数表明例程启动时，使用手动还是自动归档。

- **TRUE:** TRUE 表示归档是自动的。ARCn 将在每次日志切换时开始将已满的日志组归档。
- **FALSE:** FALSE 是缺省值，表示 DBA 将手动归档已满的重做日志文件。每次归档联机重做日志文件时，DBA 必须手动执行一条命令。可以对所有或特定的联机重做日志文件进行手动归档。

## 归档的重做日志文件

- 由 ARCn 自动完成
- 通过 SQL 语句手动完成
- 成功归档后：
  - 将在控制文件中加入一个条目
  - 记录：归档日志名、日志序列号以及高和低系统更改号 (SCN)
  - 在执行以下操作之前，不能再使用已满的重做日志文件：
  - 执行检查点操作
  - 已通过 ARCn 将文件归档
- 可以进行多元备份
- 由 DBA 维护

ORACLE

7-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 归档的重做日志文件

有关归档日志的信息可从 V\$INSTANCE 获取。

```
SQL> SELECT archiver
 2 FROM v$instance;
ARCHIVE

STOPPED
1 row selected.
```

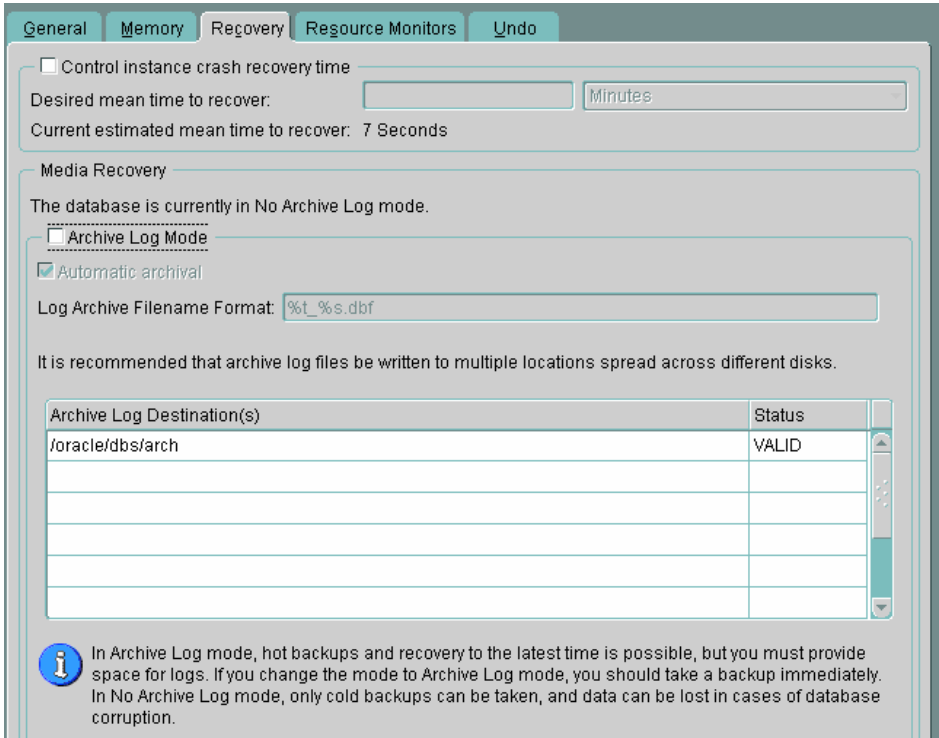
注：Oracle9i 数据库管理基础 II 课程中详细介绍了归档。

归档的重做日志文件（续）

使用 Oracle Enterprise Manager 获取归档信息

从“OEM 控制台”(OEM Console):

- 1. 导航到“数据库”(Databases) > “例程”(Instance)。
- 2. 单击“配置”(Configuration)。
- 3. “常规”(General) 页指定：
  - “数据库和例程信息 — 归档日志模式”(Database and Instance Information-Archive Log Mode): 指定数据库运行的模式
  - “全部初始化参数”(All Initialization Parameters): 确定给归档设置的任何参数
- 4. “恢复”(Recovery) 页用于设置和确定归档的细节，例如：模式、文件名格式以及日志目标位置。



## 小结

在这一课中，您应该能够掌握：

- 解释联机重做日志文件的用途
- 获取重做日志文件信息
- 控制日志切换和检查点
- 对联机重做日志文件进行多元备份和维护
- 使用 **OMF** 管理联机重做日志文件

ORACLE



## 练习 7 概览

此练习涉及以下主题：

- 创建联机重做日志文件组和成员
- 维护联机重做日志文件组和成员
- 使用 **OMF** 管理联机重做日志文件

ORACLE

7-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 7 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成此处的练习。

## 练习 7：维护重做日志文件

- 1 列出现有日志文件的数量和位置，并显示您的数据库所拥有的重做日志文件组及成员的数量。

提示：

- 查询动态视图 V\$LOGFILE。
- 使用动态视图 V\$LOG。

- 2 您的数据库是在哪种数据库模式下配置的？是否启用了归档？

提示：

- 查询动态性能视图 V\$DATABASE。
- 查询动态性能视图 V\$INSTANCE。

- 3 使用以下命名约定，向数据库中位于 u04 内的每个组添加重做日志文件成员：

向第 1 组添加成员：log01b.rdo

向第 2 组添加成员：log02b.rdo

验证结果。

提示：

- 执行 ALTER DATABASE ADD LOGFILE MEMBER 命令，将重做日志文件成员添加到每个组中。
- 查询动态性能视图 V\$LOGFILE 以对结果进行验证。

- 4 使用以下命名约定在您的数据库中添加重做日志组，该日志组有两个成员，分别位于 u03 和 u04 上：

添加第 3 组：log03a.rdo 和 log03b.rdo

验证结果。

提示：

- 执行 ALTER DATABASE ADD LOGFILE 命令创建新组。
- 查询动态性能视图 V\$LOGFILE 以显示新组中的新成员名称。
- 查询动态性能视图 V\$LOG 以显示重做日志文件组和成员的数量。

## 练习 7：维护重做日志文件（续）

### 5 删除第 4 步中创建的重做日志文件组。

提示：

- 执行 ALTER DATABASE DROP LOGFILE GROUP 命令删除日志组。
- 查询动态视图 V\$LOG 对结果进行验证。
- 删除该组的操作系统文件。

### 6 将所有联机重做日志文件的大小调整为 1024 KB。（因为无法调整日志文件的大小，所以必须添加新日志并删除旧日志。）

提示：

- 执行 ALTER DATABASE ADD LOGFILE GROUP 命令添加大小为 1024 KB 的新组。
- 查询动态视图 V\$LOG 检查活动组。
- 执行 ALTER SYSTEM SWITCH LOGFILE 命令强制执行日志切换，并将组状态更改为非活动状态。需要日志切换的次数将发生变化。
- 执行 ALTER DATABASE DROP LOGFILE 删除非活动状态组。
- 查询动态视图 V\$LOG 对结果进行验证。



# 8

## 管理表空间和数据文件

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

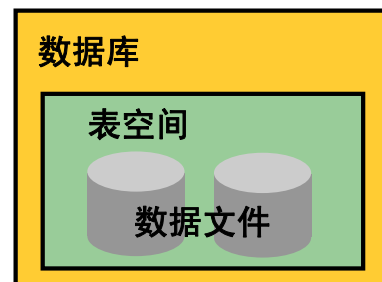
- 确定表空间和数据文件的用途
- 创建表空间
- 管理表空间
- 使用“Oracle 管理文件” (Oracle Managed Files, OMF)  
创建和管理表空间

ORACLE

# 表空间和数据文件

**Oracle** 在逻辑上以表空间存储数据，而实际上以数据文件进行存储。

- **表空间：**
  - 某一时刻只能属于一个数据库
  - 由一个或多个数据文件组成
  - 可进一步划分为逻辑存储单元
- **数据文件：**
  - 只能属于一个表空间和一个数据库
  - 是方案对象数据的资料档案库



## 表空间和数据文件

数据库、表空间和数据文件是紧密相关的，但它们之间又有着重要区别：

- Oracle 数据库由一个或多个称为表空间的逻辑存储单元组成，表空间作为一个整体存储数据库中的所有数据。
- Oracle 数据库内的每个表空间由一个或多个称为数据文件的文件组成，这些数据文件是与 Oracle 运行所在的操作系统一致的物理结构。
- 数据库的所有数据都存储在数据文件中，数据库的每个表空间都由这些数据文件组成。例如，最简单的 Oracle 数据库只有一个表空间和一个数据文件。而另一个数据库可具有三个表空间，每个表空间由两个数据文件组成（共有六个数据文件）。

# 表空间类型

- **SYSTEM 表空间**
  - 随数据库创建
  - 包含数据字典
  - 包含 SYSTEM 还原段
- **非 SYSTEM 表空间**
  - 用于分开存储段
  - 易于空间管理
  - 控制分配给用户的空间量

ORACLE

8-4

Copyright © Oracle Corporation, 2001. All rights reserved.

## 表空间类型

为加强控制和方便维护，DBA 创建了表空间。Oracle 服务器识别两种类型的表空间：SYSTEM 和所有其它表空间。

### SYSTEM 表空间：

- 随数据库创建
- 所有数据库均需要
- 包括数据字典（内含存储程序单元）
- 包含 SYSTEM 还原段
- 应不包括用户数据，尽管允许这样做

### 非 SYSTEM 表空间：

- 支持更灵活地管理数据库
- 将还原段、临时段、应用程序数据段和应用程序索引段分开
- 根据备份要求将数据分开
- 分开动态和静态数据
- 控制分配给用户对象的空间量



# 创建表空间

使用以下命令创建表空间：

**CREATE TABLESPACE**

```
CREATE TABLESPACE userdata
DATAFILE '/u01/oradata/userdata01.dbf' SIZE 100M
AUTOEXTEND ON NEXT 5M MAXSIZE 200M;
```

ORACLE

8-5

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建表空间

使用 CREATE TABLESPACE 命令可创建表空间：

```
CREATE TABLESPACE tablespace
[DATAFILE clause]
[MINIMUM EXTENT integer[K|M]]
[BLOCKSIZE integer [K]]
[LOGGING|NOLOGGING]
[DEFAULT storage_clause]
[ONLINE|OFFLINE]
[PERMANENT|TEMPORARY]
[extent_management_clause]
[segment_management_clause]
```

## 创建表空间（续）

其中：

**Tablespace**：是要创建的表空间的名称

**DATAFILE**：指定组成表空间的一个或多个数据文件

**MINIMUM EXTENT**：确保表空间内每个占用区的大小是整数 (integer) 的倍数。使用 K 或 M 以千字节或兆字节为单位指定该大小。

**BLOCKSIZE**：BLOCKSIZE 指定表空间的非标准块大小。要指定该子句，必须具有 DB\_CACHE\_SIZE，并至少设置一个 DB\_nK\_CACHE\_SIZE 参数，在该子句中指定的整数 (integer) 必须与一个 DB\_nK\_CACHE\_SIZE 参数设置相对应。

**LOGGING**：指定在缺省情况下，表空间内的所有表、索引和分区的所有更改都写入重做日志文件。LOGGING 为缺省设置。

**NOLOGGING**：指定在缺省情况下，表空间内的所有表、索引和分区的所有更改都不写入重做日志文件。NOLOGGING 只影响某些 DML 和 DDL 命令，如直接加载。

**DEFAULT**：DEFAULT 指定表空间内创建的所有对象的缺省存储参数。

**OFFLINE**：指定表空间从创建后就不可用。

**PERMANENT**：指定表空间可用于保留永久对象。

**TEMPORARY**：指定表空间仅用于保留临时对象，如：由 ORDER BY 子句引起的隐式排序所使用的段。不能指定 EXTENT MANAGEMENT LOCAL 或 BLOCKSIZE 子句。

**extent\_management\_clause**：该子句指定如何管理表空间内的区。该子句在本课的后续部分中讨论。

**segment\_management\_clause**：这只与永久的、且在本地管理的表空间相关。通过它可指定 Oracle 是否应使用空闲列表或位图来跟踪表空间段中的已占用空间和空闲空间。

**datafile\_clause ::= filename [SIZE integer[K|M] [REUSE]  
[ autoextend\_clause ]**

**filename**：是表空间中的数据文件的名称。

**SIZE**：指定文件大小。使用 K 或 M 以千字节或兆字节为单位指定大小。

**REUSE**：允许 Oracle 服务器重新使用现有文件。

**autoextend\_clause**：该子句启用或禁用数据文件的自动扩展。

**NEXT**：以字节为单位指定在需要更多区时自动分配的磁盘空间下一增量的大小。

## 创建表空间（续）

其中：

**MAXSIZE**：指定数据文件可以自动扩展到的最大磁盘空间。

**UNLIMITED**：指定可分配给数据文件或 **Tempfile** 的磁盘空间是不受限制的。

另请参阅“Oracle9i SQL Reference”和“Oracle9i Concepts”以获取更多信息。

## 创建表空间（续）

### 使用 Oracle Enterprise Manager 创建表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General) 和 “存储”(Storage) 选项卡中填写创建表空间所需的信息。
4. 单击“创建”(Create)。

The screenshot shows the 'Create Tablespace' dialog box in Oracle Enterprise Manager, with the 'Storage' tab selected. The 'Name' field is set to 'USERDATA'. Below it, the 'Datafiles' section contains a table with one row: 'USERDATA01.dbf' in the 'File Name' column, '/home/dba01/ORADATA/u01/' in the 'File Directory' column, and '100 MB' in the 'Size' column. The 'Status' section has 'Online' selected and 'Normal' in the dropdown. The 'Type' section has 'Permanent' selected. At the bottom are buttons for 'Create', 'Cancel', 'Show SQL', and 'Help'.

| File Name      | File Directory           | Size   |
|----------------|--------------------------|--------|
| USERDATA01.dbf | /home/dba01/ORADATA/u01/ | 100 MB |

# 表空间的空间管理

- **本地管理的表空间：**
  - 在表空间内管理空闲区
  - 使用位图来记录空闲区
  - 每一位与一个块或一组块相对应
  - 位的数值指明是空闲还是已占用
- **字典管理的表空间：**
  - 由数据字典管理空闲区
  - 在分配或回收区时更新对应的表

ORACLE

8-9

Copyright © Oracle Corporation, 2001. All rights reserved.

## 表空间的空间管理

表空间以区为单位分配空间。可使用以下两种不同方法来跟踪创建的表空间中的空闲空间和已占用空间：

**本地管理的表空间：**在表空间内通过位图管理区。位图中的每个位对应于一个块或一组块。分配了某个区或释放了某个区可重新使用时，Oracle 服务器更改位图值以显示块的新状态。从 Oracle9i 开始，在本地管理已成为缺省设置。

**字典管理的表空间：**由数据字典管理区。Oracle 服务器将在分配或回收区时更新数据字典中对应的表。

## 本地管理的表空间

- 减少了对数据字典表的争用
- 分配或回收空间时不生成还原数据
- 无需合并

```
CREATE TABLESPACE userdata
DATAFILE '/u01/oradata/userdata01.dbf' SIZE 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K;
```

ORACLE

8-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 本地管理的表空间

EXTENT MANAGEMENT 子句的 LOCAL 选项指定表空间在本地管理。缺省情况下，表空间在本地管理。

extent\_management\_clause:

```
[EXTENT MANAGEMENT [DICTIONARY | LOCAL
[AUTOALLOCATE | UNIFORM [SIZE integer[K|M]]]]]
```

其中:

DICTIONARY: 指定使用字典表来管理表空间。

LOCAL: 指定在本地通过位图管理表空间。如果指定了 LOCAL，则不能再指定 DEFAULT storage\_clause、MINIMUM EXTENT 或 TEMPORARY。

AUTOALLOCATE: 指定表空间由系统管理。用户无法指定区大小。这是缺省设置。

UNIFORM: 指定按照大小统一为 SIZE 字节数的各个区来管理表空间。使用 K 或 M 以千字节或兆字节来指定区大小。缺省大小为 1 MB。

## 本地管理的表空间（续）

EXTENT MANAGEMENT 子句可用于各种 CREATE 命令中：

- 对于非 SYSTEM 的永久表空间，您可以在 CREATE TABLESPACE 命令中指定 EXTENT MANAGEMENT LOCAL。
- 对于临时表空间，您可以在 CREATE TEMPORARY TABLESPACE 命令中指定 EXTENT MANGEMENT LOCAL。

## 在本地管理表空间的优点：

本地管理的表空间相对于字典管理的表空间有如下优点：

- 本地管理可以避免循环空间管理操作，但是这种操作在字典管理的表空间中却有可能发生。一旦消耗或释放某个区的空间会产生另一个消耗或释放操作（消耗或释放还原段或数据字典表内的空间）时，它就会发生。
- 由于本地管理的表空间在数据字典表中不记录空闲空间，从而减少了对这些表的争用。
- 区的本地管理可自动跟踪邻近的空闲空间，因而无须合并空闲区。
- 本地管理的区大小可由系统自动确定。
- 对区的位图进行更改不会生成还原信息，因为它们不更新数据字典中的表（表空间限额信息等特殊情况除外）。

## 字典管理的表空间

- 在数据字典中管理区
- 存储在表空间中的每个段都可以有不同的存储子句
- 需要合并

```
CREATE TABLESPACE userdata
DATAFILE '/u01/oradata/userdata01.dbf'
SIZE 500M EXTENT MANAGEMENT DICTIONARY
DEFAULT STORAGE
(initial 1M NEXT 1M PCTINCREASE 0);
```

ORACLE

8-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 字典管理的表空间

字典管理的表空间内的段可具有自定义的存储设置。这比本地管理的表空间更灵活，但效率要低得多。



## 还原表空间

- 用于存储还原段
- 不能包含任何其它对象
- 其中的区要在本地管理
- 只能使用 DATAFILE 和 EXTENT MANAGEMENT 子句

```
CREATE UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undo01.dbf' SIZE 40M;
```

ORACLE

8-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 还原表空间

还原表空间使用“自动还原管理”(Automatic Undo Management)的方式。有关“自动还原管理”的更多信息，请参考“管理还原数据”一课。

```
CREATE UNDO TABLESPACE tablespace
[DATAFILE clause]
```

## 临时表空间

- 用于排序操作
- 不能包含任何永久对象
- 建议在本地管理区

```
CREATE TEMPORARY TABLESPACE temp
TEMPFILE '/u01/oradata/temp01.dbf' SIZE 500M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4M;
```

ORACLE

8-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### 临时表空间

通过指定专门用于排序段的临时表空间，您可以更有效地管理用于排序操作的空间。在临时表空间内不能驻留永久方案对象。

当一个段由多个排序操作共享时，就使用排序段或者临时段。当多个排序太大而无法装入内存时，使用临时表空间能改进性能。给定临时表空间的排序段在例程首次执行排序操作时创建。排序段通过分配更多的区来扩展，直到段大小等于或者大于该例程上运行的所有活动排序的存储要求总和。

## 临时表空间（续）

本地管理的临时表空间具有临时数据文件 (Tempfile)，它与普通数据文件很相似，只有以下几点不同：

- Tempfile 始终设为 NOLOGGING 模式。
- 无法将 Tempfile 设置为只读。
- 无法重命名 Tempfile。
- 无法通过 ALTER DATABASE 命令创建 Tempfile。
- Tempfile 对于只读数据库是必需的。
- 介质恢复不恢复 Tempfile。

若要优化临时表空间内的排序性能，可将 UNIFORM SIZE 设置为 SORT\_AREA\_SIZE 参数的整数倍。

## 临时段（续）

### 使用 Oracle Enterprise Manager 创建临时表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General) 选项卡中提供详细信息。
4. 在“类型”(Type) 区域中选择“临时”(Temporary) 选项。
5. 单击“存储”(Storage) 选项卡，输入存储信息。
6. 单击“创建”(Create)。

## 缺省临时表空间

- 指定数据库范围内的缺省临时表空间
- 避免使用 `SYSTEM` 表空间存储临时数据
- 可使用以下命令进行创建：
  - `CREATE DATABASE`
  - 在本地管理
  - `ALTER DATABASE`

```
ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE temp;
```

ORACLE

8-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### 缺省临时表空间

创建未指定缺省临时表空间的数据库时，分配给任意用户的，未使用 `TEMPORARY TABLESPACE` 子句创建的表空间是 `SYSTEM` 表空间。此时，`alert_sid.log` 中会记录一条警告消息，指出 `SYSTEM` 表空间是缺省临时表空间。在创建数据库期间创建缺省临时表空间可防止将 `SYSTEM` 表空间用作临时空间。

创建数据库后，可通过创建临时表空间然后改变数据库来设置缺省临时表空间。

```
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp;
```

定义后，未显式分配到临时表空间的用户将被分配到该缺省临时表空间。

缺省临时表空间可通过使用 `ALTER DATABASE DEFAULT TEMPORARY TABLESPACE` 命令随时进行更改。更改缺省临时表空间后，分配到该缺省临时表空间的所有用户将被重新分配到新的缺省表空间。

# 创建缺省临时表空间

- 在创建数据库期间:

```
CREATE DATABASE DBA01
LOGFILE
GROUP 1 ('/$HOME/ORADATA/u01/redo01.log') SIZE 100M,
GROUP 2 ('/$HOME/ORADATA/u02/redo02.log') SIZE 100M,
MAXLOGFILES 5
MAXLOGMEMBERS 5
MAXLOGHISTORY 1
MAXDATAFILES 100
MAXINSTANCES 1
DATAFILE '/$HOME/ORADATA/u01/system01.dbf' SIZE 325M
UNDO TABLESPACE undotbs
DATAFILE '/$HOME/ORADATA/u02/undotbs01.dbf' SIZE 200
DEFAULT TEMPORARY TABLESPACE temp
TEMPFILE '/$HOME/ORADATA/u03/temp01.dbf' SIZE 4M
CHARACTER SET US7ASCII
```

ORACLE

8-18

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建缺省临时表空间

在创建数据库期间:

创建未指定缺省临时表空间的数据库时, 分配给任意用户的, 未使用 TEMPORARY TABLESPACE 子句创建的缺省表空间是 SYSTEM 表空间。此时, alertSID.log 会记录一条警告消息, 指示出 SYSTEM 表空间是缺省临时表空间。

在创建数据库期间创建缺省临时表空间可防止将 SYSTEM 表空间用作临时空间。使用 CREATE DATABASE 命令创建缺省临时表空间时, 其类型是在本地管理。

## 创建缺省临时表空间

- 创建数据库后：

```
ALTER DATABASE
DEFAULT TEMPORARY TABLESPACE default_temp2;
```

- 查询 DATABASE\_PROPERTIES 以确定数据库的缺省临时表空间

```
SELECT * FROM DATABASE_PROPERTIES;
```

ORACLE

8-19

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建缺省临时表空间（续）

创建数据库后：

缺省临时表空间可通过下列方法创建和设置：

- 使用 CREATE TABLESPACE 命令创建临时表空间
- 使用 ALTER DATABASE 命令，如上所述

定义后，未显式分配到临时表空间的用户将被分配到该缺省临时表空间。

缺省临时表空间可通过使用 ALTER DATABASE DEFAULT TEMPORARY TABLESPACE 命令随时进行更改。更改缺省临时表空间后，已分配到该缺省临时表空间的所有用户将被重新分配到新的缺省表空间。

## 创建缺省临时表空间（续）

使用 **Oracle Enterprise Manager** 创建临时表空间：

从“OEM 控制台”(OEM Console)：

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General) 选项卡中选择“临时”(Temporary)，然后选择“设置为缺省临时表空间”(Set as Default Temporary Tablespace)。
4. 在“存储”(Storage) 选项卡中输入必要的信息。
5. 单击“创建”(Create)。



## 缺省临时表空间的限制

不能对缺省临时表空间执行下列操作：

- 将其删除，除非已经有新的缺省临时表空间
- 使其脱机
- 更改为永久表空间

ORACLE

8-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### 缺省临时表空间的限制

#### 删除缺省临时表空间

您只有指定了一个新的缺省表空间后，才能删除旧的缺省临时表空间。必须使用 `ALTER DATABASE` 命令才能将缺省临时表空间更改为新的缺省值。旧的缺省临时表空间仅在新的缺省临时表空间可用时才会被删除。分配到旧的缺省临时表空间的用户将被自动重新分配到新的缺省临时表空间。

#### 更改缺省临时表空间的类型

由于缺省临时表空间必须是 `SYSTEM` 表空间或临时表空间，因此，无法将缺省临时表空间更改为永久类型。

#### 使缺省临时表空间脱机

使表空间脱机后，例如在进行脱机备份、维护或更改使用该表空间的应用程序时，其他用户将无法使用对应的这部分数据库内容。由于上述情况都不适用于临时表空间，因此无法使缺省临时表空间脱机。

## 只读表空间

- 使用以下命令可将表空间置于只读模式

```
ALTER TABLESPACE userdata READ ONLY;
```

- 导致检查点操作
- 数据仅用于读操作
- 可从表空间删除对象

ORACLE

8-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### 只读表空间

ALTER TABLESPACE [tablespace] READ ONLY 命令将表空间置于过渡只读模式。除了以前修改过该表空间中的块的已有事务处理回退，这种过渡状态不允许再对该表空间进行任何写入操作。当已有的所有事务处理提交或者回退后，只读命令完成，该表空间置于只读模式。

您可以删除只读表空间内的表和索引等项，因为这些命令只影响数据字典。之所以可以这样操作，是因为 DROP 命令只更新数据字典，而不更新构成表空间的物理文件。对于本地管理的表空间，删除的段将改为临时段以避免更新位图。要使只读表空间可写，表空间内的所有数据文件都必须联机。将表空间设为只读将导致对表空间的数据文件执行检查点操作。

## 只读表空间（续）

将表空间设为只读可防止对表空间中的数据文件进行任何写操作。为此，数据文件可驻留在只读介质上，如 CD-ROM 或一次性写入 (WORM) 驱动器。只读表空间可以免去对数据库大量的静态分配执行备份。

在一次性写入 (WORM) 设备上创建只读表空间：

1. ALTER TABLESPACE...READ ONLY
2. 使用操作系统命令将表空间的数据文件移动到只读设备上。
3. ALTER TABLESPACE...RENAME DATAFILE

## 只读表空间（续）

### 使用 **Oracle Enterprise Manager** 将表空间设为只读

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 选择表空间。
3. 在“General”(常规)选项卡的“Status”(状态)区域选择“Read Only”(只读)复选框。
4. 单击“应用”(Apply)。

# 使表空间脱机

- 无法访问数据
- 不能设为脱机的表空间：
  - SYSTEM 表空间
  - 具有活动的还原段的表空间
  - 缺省临时表空间
- 使用以下命令可使表空间脱机：

```
ALTER TABLESPACE userdata OFFLINE;
```

- 使用以下命令可使表空间联机：

```
ALTER TABLESPACE userdata ONLINE;
```

ORACLE

8-25

Copyright © Oracle Corporation, 2001. All rights reserved.

## 使表空间脱机

表空间一般是联机的，这样可方便数据库用户使用其中包含的数据。但是，数据库管理员可以让表空间脱机以便：

- 使数据库的一部分不可用，但允许正常访问数据库的其余部分
- 执行脱机表空间备份（尽管表空间可以在联机使用时备份）
- 在数据库打开时恢复表空间或数据文件
- 在数据库打开时移动数据文件

### 表空间的脱机状态

当表空间脱机后，Oracle 不允许随后有任何 SQL 语句引用该表空间含有的对象。试图对脱机表空间内的对象进行访问的用户将收到一条错误消息。

当表空间脱机或者重新联机后，该事件记录在数据字典和控制文件内。如果关闭数据库时表空间仍然脱机，则当随后数据库装载并重新打开时，该表空间仍保持脱机且不会被检查。

## 使表空间脱机（续）

如果遇到某些错误（例如，当数据库写入程序进程 DBWn 几次试图向某表空间的数据文件写入都失败时），Oracle 例程自动将表空间从联机切换为脱机。不同的错误情况在 *Oracle9i 数据库管理基础 II* 课程内有更为详细的讨论。

### 使表空间脱机

只要数据库打开，数据库管理员就可以使任何表空间脱机（SYSTEM 表空间和任何具有活动还原段或临时段的表空间除外）。当一个表空间脱机后，Oracle 服务器将使与之相关联的所有数据文件脱机

```
ALTER TABLESPACE tablespace
```

```
{ONLINE | OFFLINE [NORMAL | TEMPORARY | IMMEDIATE | FOR RECOVER]}
```

其中：

**NORMAL：** 将该表空间中所有数据文件内的所有块从 SGA 中清空。这是缺省设置。在使该表空间重新联机之前，您无须对其执行介质恢复。尽可能使用 NORMAL 子句。

**TEMPORARY：** 对表空间内的所有联机数据文件执行检查点操作，即使某些文件无法写入。所有脱机文件可能都需要进行介质恢复。

**IMMEDIATE：** 不保证表空间文件可用，而且不执行检查点操作。在使表空间重新联机前，您必须对其执行介质恢复操作。

**FOR RECOVER：** 使表空间脱机以进行表空间时间点恢复。

## 使表空间脱机（续）

### 使用 Oracle Enterprise Manager 使表空间脱机

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 选择表空间。
3. 在“常规”(General) 选项卡的“状态”(Status) 区域，选择“脱机”(Offline)。
4. 从下拉菜单中选择“模式”(Mode)。
5. 单击“应用”(Apply)。

## 更改存储设置

- 使用 ALTER TABLESPACE 命令更改存储设置

```
ALTER TABLESPACE userdata MINIMUM EXTENT 2M;
```

```
ALTER TABLESPACE userdata
DEFAULT STORAGE (INITIAL 2M NEXT 2M
MAXEXTENTS 999);
```

- 不能更改在本地管理的表空间的存储设置

ORACLE

8-28

Copyright © Oracle Corporation, 2001. All rights reserved.

### 更改存储设置

使用 ALTER TABLESPACE 命令可以改变表空间的缺省存储定义：

```
ALTER TABLESPACE tablespace
[MINIMUM EXTENT integer[K|M]
|DEFAULT storage_clause]
```



## 更改缺省存储设置（续）

### 使用 Oracle Enterprise Manager 更改存储设置

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 使用鼠标右键单击表空间，从弹出的菜单中选择“查看/编辑详细资料”(View/Edit Details)。
3. 单击“存储”(Storage) 选项卡并进行必要的更改。
4. 单击“应用”(Apply)。

# 调整表空间大小

表空间大小可通过以下方法进行调整：

- 更改数据文件的大小：
  - 使用 `AUTOEXTEND` 自动调整
  - 使用 `ALTER TABLESPACE` 手动调整
- 使用 `ALTER TABLESPACE` 添加数据文件

ORACLE

8-30

Copyright © Oracle Corporation, 2001. All rights reserved.

## 调整表空间大小

您可以通过下面两种方法增大表空间：

- 选择自动或手动更改数据文件的大小。
- 向表空间添加数据文件。

## 启用数据文件自动扩展

- 可使用以下命令自动调整大小：
  - CREATE DATABASE
  - CREATE TABLESPACE
  - ALTER TABLESPACE ... ADD DATAFILE

- 示例：

```
CREATE TABLESPACE user_data
DATAFILE
 '/u01/oradata/userdata01.dbf' SIZE 200M
 AUTOEXTEND ON NEXT 10M MAXSIZE 500M;
```

- 查询 DBA\_DATA\_FILES 视图以确定是否启用了 AUTOEXTEND。

ORACLE

8-31

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用数据文件自动扩展

为新数据文件指定 AUTOEXTEND

通过 AUTOEXTEND 子句可启用或禁用数据文件的自动扩展。文件将按指定的增量增加直到达到指定的最大值。

使用 AUTOEXTEND 子句的优点：

- 当表空间的空间用尽时无需过多的直接干预
- 确保应用程序不会由于未能分配区而暂停

创建数据文件后，可使用下列 SQL 命令启用数据文件的自动扩展：

- CREATE DATABASE
- CREATE TABLESPACE ... DATAFILE
- ALTER TABLESPACE ... ADD DATAFILE

## 启用数据文件自动扩展（续）

### 为新数据文件指定 **AUTOEXTEND**（续）

使用 ALTER DATABASE 命令可修改数据文件并启用自动扩展：

```
ALTER DATABASE DATAFILE filespec [autoextend_clause]
autoextend_clause ::= [AUTOEXTEND { OFF|ON[NEXT integer[K|M]]
 [MAXSIZE UNLIMITED | integer[K|M]] }]
```

其中：

**AUTOEXTEND OFF**：禁用数据文件的自动扩展

**AUTOEXTEND ON**：启用数据文件的自动扩展

**NEXT**：指定在需要更多区时分配给数据文件的磁盘空间

**MAXSIZE**：指定允许分配给该数据文件的最大磁盘空间

**UNLIMITED**：将分配给数据文件的磁盘空间设为不受限

### 为现有数据文件指定 **AUTOEXTEND**

使用 SQL 命令 **ALTER DATABASE** 可对现有数据文件启用或禁用自动文件扩展：

```
ALTER DATABASE [database]
 DATAFILE 'filename'[, 'filename']... autoextend_clause
```

### 确定 **AUTOEXTEND** 已启用还是已禁用

查询 DBA\_DATA\_FILES 视图以确定是否启用了 **AUTOEXTEND** 并检查 **AUTOEXTENSIBLE** 列。

```
SQL> select tablespace_name, file_name, autoextensible
 2 from dba_data_files;
```

| TABLESPACE_NAME | FILE_NAME                            | AUTOEXTENSIBLE |
|-----------------|--------------------------------------|----------------|
| SYSTEM          | /home/dba01/ORADATA/u01/system01.dbf | YES            |
| DATA01          | /home/dba01/ORADATA/u04/data01.dbf   | NO             |
| USERS           | /home/dba01/ORADATA/u03/users01.dbf  | NO             |
| INDX            | /home/dba01/ORADATA/u06/indx01.dbf   | NO             |
| SAMPLE          | /home/dba01/ORADATA/u02/sample01.dbf | YES            |
| DATA02          | /home/dba01/ORADATA/u03/data02.dbf   | NO             |
| INDEX01         | /home/dba01/ORADATA/u06/index01.dbf  | YES            |
| UNDO2           | /home/dba01/ORADATA/u01/UNDO2.dbf    | NO             |

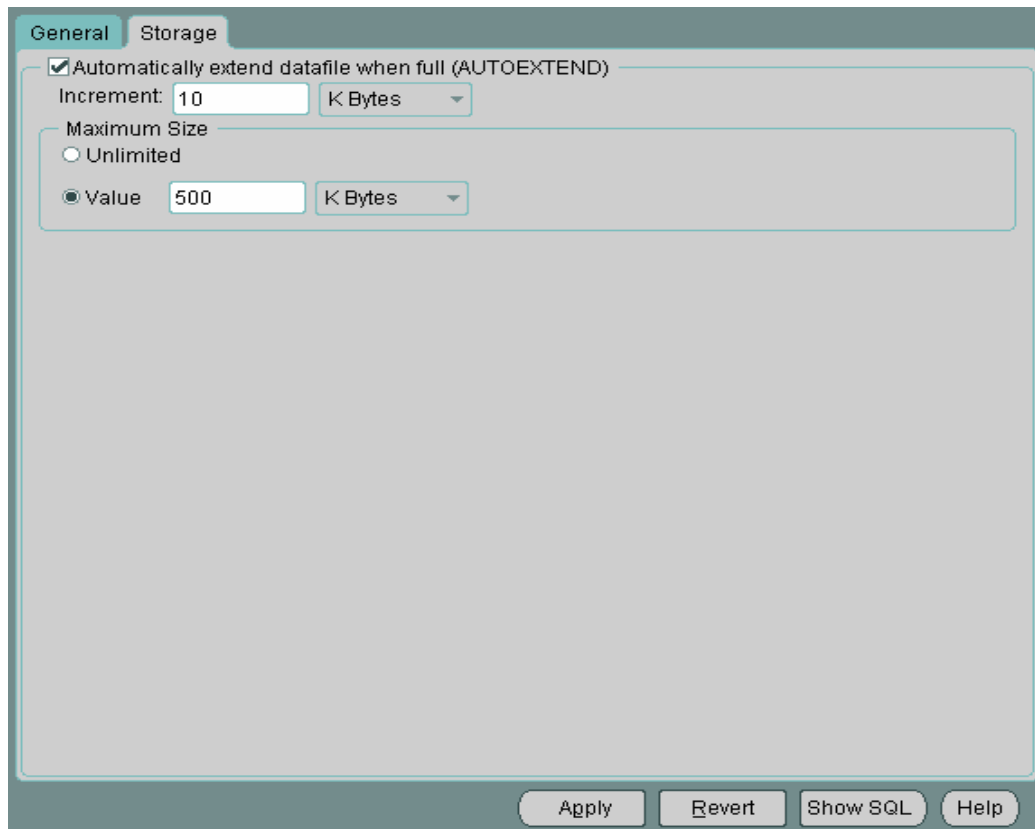
8 rows selected.

## 为新数据文件指定 AUTOEXTEND（续）

### 使用 Oracle Enterprise Manager 启用自动调整大小功能

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “数据文件”(Datafiles)。
2. 选择数据文件。
3. 在“存储”(Storage) 选项卡中, 选择“数据文件已满后自动扩展”(Automatically extend datafile when full) 复选框。
4. 设置“增量”(Increment) 和“最大大小”(Maximum Size) 的值。
5. 单击“应用”(Apply)。



## 手动调整数据文件的大小

- 使用 ALTER DATABASE 可手动增加或减少数据文件的大小
- 调整数据文件大小可在无需添加更多数据文件的情况下添加更多空间
- 手动调整数据文件大小将回收数据库中的未用空间
- 示例:

```
ALTER DATABASE
DATAFILE '/u03/oradata/userdata02.dbf'
RESIZE 200M;
```

ORACLE

8-34

Copyright © Oracle Corporation, 2001. All rights reserved.

### 手动调整数据文件的大小

DBA 能够更改数据文件的大小，他们可以使用 ALTER DATABASE 命令手动增加或减少数据文件的大小，而不必通过添加数据文件来向数据库添加空间：

```
ALTER DATABASE [database]
 DATAFILE 'filename'[, 'filename']...
 RESIZE integer[K|M]
```

其中：

Integer：以字节为单位表示的结果数据文件的绝对大小

如果存储的数据库对象超过指定大小，那么数据文件大小只能减少到数据文件内最后一个对象的最后的块为止。

## 向表空间添加数据文件

- 通过添加其它数据文件来增加分配给表空间的空间
- 通过 `ADD DATAFILE` 子句可添加数据文件
- 示例:

```
ALTER TABLESPACE user_data
ADD DATAFILE '/u01/oradata/userdata03.dbf'
SIZE 200M;
```

ORACLE

8-35

Copyright © Oracle Corporation, 2001. All rights reserved.

### 向表空间添加数据文件

您可以通过 `ALTER TABLESPACE ADD DATAFILE` 命令，向表空间添加数据文件以增加分配给表空间的磁盘空间总量：

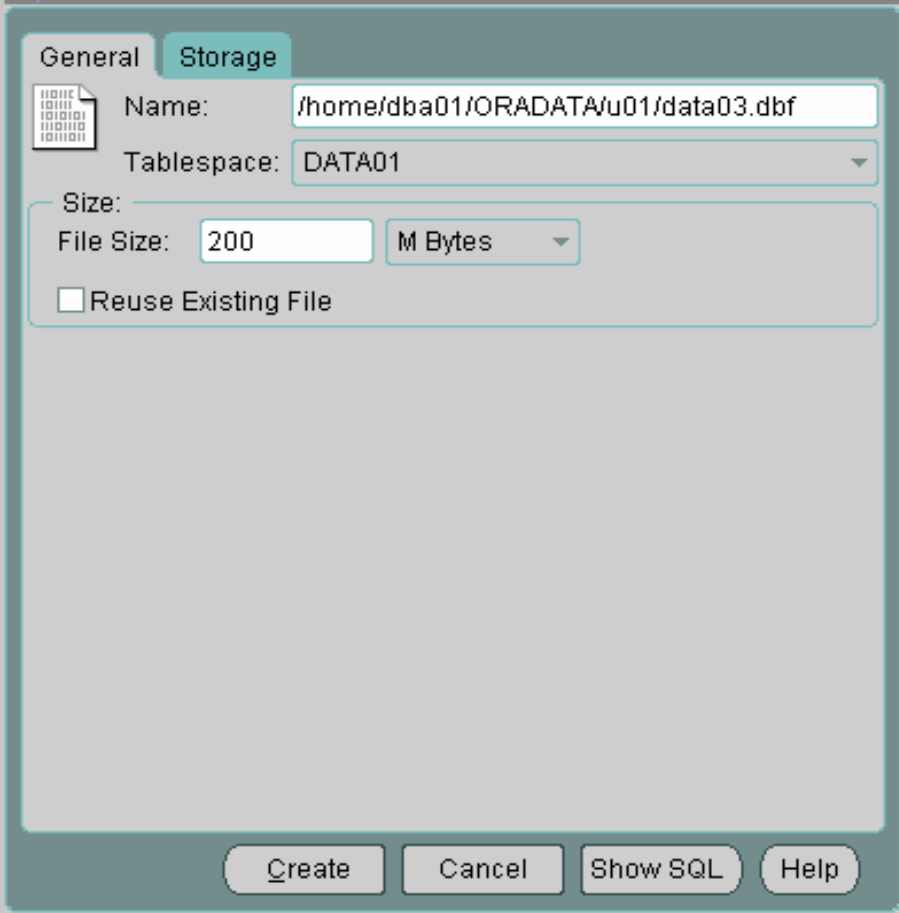
```
ALTER TABLESPACE tablespace
ADD DATAFILE
filespec [autoextend_clause]
```

## 向表空间添加数据文件（续）

### 使用 Oracle Enterprise Manager 添加数据文件

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 选择表空间。
3. 选择“添加数据文件”(Add Datafile)。
4. 在“常规”(General) 选项卡中输入文件信息。
5. 单击“创建”(Create)。



The screenshot shows the 'Add Datafile' dialog box in Oracle Enterprise Manager, with the 'Storage' tab selected. The dialog has two tabs: 'General' and 'Storage'. The 'Storage' tab contains the following fields and controls:

- Name:** A text field containing the path `/home/dba01/ORADATA/u01/data03.dbf`.
- Tablespace:** A dropdown menu showing `DATA01`.
- Size:** A section containing:
  - File Size:** A text field with the value `200`.
  - Unit:** A dropdown menu showing `M Bytes`.
- Reuse Existing File:** An unchecked checkbox.

At the bottom of the dialog, there are four buttons: `Create`, `Cancel`, `Show SQL`, and `Help`.



## 移动数据文件的方法

- **ALTER TABLESPACE**

- 表空间必须脱机
- 目标数据文件必须存在

```
ALTER TABLESPACE userdata RENAME
DATAFILE '/u01/oradata/userdata01.dbf'
TO '/u02/oradata/userdata01.dbf';
```

- **重命名数据文件的步骤：**

- 使表空间脱机。
- 使用操作系统命令移动或复制文件。
- 执行 **ALTER TABLESPACE RENAME DATAFILE** 命令。
- 使表空间联机。
- 必要时使用操作系统命令删除该文件。

ORACLE

8-37

Copyright © Oracle Corporation, 2001. All rights reserved.

### 移动数据文件的方法

根据表空间类型的不同，数据库管理员可使用以下两种方法之一来移动数据文件：

#### **ALTER TABLESPACE 命令**

下面显示了 **ALTER TABLESPACE** 命令，它仅适用于不含活动还原段或临时段的非 **SYSTEM** 表空间中的数据文件：

```
ALTER TABLESPACE tablespace
 RENAME DATAFILE 'filename'[, 'filename']...
 TO 'filename'[, 'filename']...
```

源文件名必须与存储在控制文件内的名称匹配。

# 移动数据文件的方法

- **ALTER DATABASE**

- 数据库必须已装载
- 目标数据文件必须存在

```
ALTER DATABASE RENAME
FILE '/u01/oradata/system01.dbf'
TO '/u03/oradata/system01.dbf';
```

ORACLE

8-38

Copyright © Oracle Corporation, 2001. All rights reserved.

## 移动数据文件的方法（续）

### ALTER DATABASE 命令

ALTER DATABASE 命令可用来移动任意类型的数据文件：

```
ALTER DATABASE [database]
 RENAME FILE 'filename'[, 'filename']...
 TO 'filename'[, 'filename']...
```

因为 SYSTEM 表空间无法脱机，您必须使用该方法移动 SYSTEM 表空间内的数据文件。

使用如下进程重命名无法脱机的表空间内的文件：

1. 关闭数据库。
2. 使用操作系统命令移动文件。
3. 装载数据库。
4. 执行 ALTER DATABASE RENAME FILE 命令。
5. 打开数据库。

## 移动数据文件的方法（续）

### 使用 Oracle Enterprise Manager 移动数据文件

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 选择要移动的数据文件所驻留的表空间。
3. 在“常规”(General) 页，选择“脱机”(Offline)。
4. 选择“应用”(Apply)。
5. 表空间脱机后，立即更新“常规”(General) 页中的“文件目录”(File Directory) 信息。
6. 单击“应用”(Apply)。

#### 注:

- 这些命令验证该文件存在于新位置；它们并不创建或者移动文件。
- 请始终提供完整的文件名（包括其路径）以标识旧的和新的数据文件。

## 删除表空间

- 不能删除下列表空间：
  - SYSTEM 表空间
  - 具有活动段的表空间
- INCLUDING CONTENTS 将删除段
- INCLUDING CONTENTS AND DATAFILES 将删除数据文件
- CASCADE CONSTRAINTS 将删除所有引用完整性约束

```
DROP TABLESPACE userdata
INCLUDING CONTENTS AND DATAFILES;
```

ORACLE

8-40

Copyright © Oracle Corporation, 2001. All rights reserved.

### 删除表空间

当不再需要表空间及其内容时，可以通过下面的 DROP TABLESPACE SQL 命令从数据库中删除表空间：

```
DROP TABLESPACE tablespace
[INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]]
```

其中：

tablespace：指定要删除的表空间的名称

INCLUDING CONTENTS：删除表空间内的所有段

AND DATAFILES：删除关联的操作系统文件

CASCADE CONSTRAINTS：如果要删除的表空间之外的表引用了该表空间内表的主键和唯一键，则删除这种引用完整性约束

## 删除表空间（续）

### 原则：

- 不使用 INCLUDING CONTENTS 选项，将无法删除仍包含数据的表空间。当表空间包含许多对象时，该选项可能会生成许多还原数据。
- 删除表空间后，其数据将不再包含在数据库内。
- 在删除表空间时，只删除关联数据库控制文件内的文件指针。操作系统文件仍然存在，如果未使用 AND DATAFILES 子句或数据文件是 OMF，则必须使用适当的操作系统命令明确删除这些文件。
- 即使将表空间切换到只读状态，仍可以删除该表空间以及其中的段。
- 删除表空间之前，建议您将表空间脱机，以确保没有事务处理访问该表空间内的任何段。

## 删除表空间（续）

### 使用 Oracle Enterprise Manager 删除表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 选择表空间。
3. 单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
4. 单击“是”(Yes) 确认删除。

## 使用 OMF 管理表空间

- 使用下列方法之一定义 DB\_CREATE\_FILE\_DEST 参数：
  - 初始化参数文件
  - 使用 ALTER SYSTEM 命令动态设置

```
ALTER SYSTEM SET
db_create_file_dest = '/u01/oradata/dba01';
```

- 创建表空间时：
  - 自动创建数据文件并存放在由 DB\_CREATE\_FILE\_DEST 指定的目录下
  - 缺省大小是 100 MB
  - AUTOEXTEND 设置为 UNLIMITED

ORACLE

8-43

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 OMF 管理表空间

配置 OMF 以创建表空间时，需指定一个初始化参数 DB\_CREATE\_FILE\_DEST，这样就不再需要 DATAFILE 子句。所有数据文件都将自动创建，其位置由 B\_CREATE\_FILE\_DEST 定义。数据文件名由 Oracle 服务器自动生成（如：ora\_tbs1\_2ixfh90q.dbf）。

## 使用 OMF 管理表空间

- 创建 OMF 表空间：

```
CREATE TABLESPACE text_data DATAFILE SIZE 20M;
```

- 向现有表空间添加 OMF 数据文件：

```
ALTER TABLESPACE text_data ADD DATAFILE;
```

- 动态更改缺省文件位置：

```
ALTER SYSTEM SET
db_create_file_dest = '/u01/oradata/dba01';
```

- 删除表空间也将删除操作系统文件：

ORACLE

8-44

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 OMF 管理表空间

#### 创建 OMF 表空间

使用 OMF 创建表空间时无需使用 DATAFILE 子句。忽略 DATAFILE 子句的表空间的缺省设置是大小为 100M 的数据文件，并且该文件设置 AUTOEXTEND 且不受 MAXSIZE 限制。也可以选择指定文件大小。

```
CREATE TABLESPACE tablespace
[DATAFILE [filename] [SIZE integer [K|M]]];
```

#### 向 OMF 表空间添加数据文件

可向现有表空间添加数据文件。使用 ADD DATAFILE 命令时不再需要文件说明。

#### 动态更改缺省文件位置

应对 DB\_CREATE\_ONLINE\_LOG\_DEST\_n 进行设置，以确保日志文件和控制文件与数据文件不在同一个目录下。可以通过 ALTER SYSTEM SET 命令对目标位置进行动态更改。

#### 删除 OMF 表空间

对于利用 OMF 创建的表空间内的数据文件，当删除关联的表空间时，将在操作系统一级上删除这些文件。



## 获取表空间信息

通过以下查询可获取表空间和数据文件的信息：

- 表空间：
  - DBA\_TABLESPACES
  - V\$TABLESPACE
- 数据文件信息：
  - DBA\_DATA\_FILES
  - V\$DATAFILE
- 临时文件信息：
  - DBA\_TEMP\_FILES
  - V\$TEMPFILE

ORACLE

## 小结

在这一课中，您应该能够掌握：

- 使用表空间分隔数据
- 创建各种类型的表空间
- 管理表空间
- 使用 **OMF** 管理表空间

ORACLE

## 练习 8 概览

此练习涉及以下主题：

- 创建表空间
- 修改表空间
- 使用 **OMF** 配置和创建表空间

ORACLE

8-47

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 8 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 8：管理表空间和数据文件

1 创建具有下列名称和存储设置的永久表空间：

- a DATA01 是数据字典管理的表空间。
- b DATA02 是具有统一的区大小的本地管理的表空间。确保表空间内使用的每个区的大小都是 100 KB 的倍数。
- c INDEX01 是区大小统一为 4K 的本地管理的表空间。当需要更多区时，允许自动扩展 500 KB，最大大小为 2 MB。
- d RONLY 是具有缺省存储设置且包含只读表的表空间。不要在此时将表空间设为只读。
- e 显示数据字典中的信息。

**提示：**可通过下列查询查看有关表空间的信息。

- DBA\_TABLESPACES
- V\$TABLESPACE
- V\$DATAFILE

2 为表空间 DATA02 再分配 500 KB 的磁盘空间。验证结果。

3 将表空间 INDEX01 重定位到子目录 u06。验证 INDEX01 的重定位结果及该表空间的状态。

**提示：**

- 使 INDEX01 表空间脱机。
- 使用 V\$DATAFILE 验证状态。
- 使用操作系统移动命令将表空间移动到 u06。
- 使用 ALTER TABLESPACE 命令重定位表空间。
- 使 INDEX01 表空间联机。
- 使用 V\$DATAFILE 验证状态。

4 a 在表空间 RONLY 中创建一个表。将表空间 RONLY 设为只读。运行查询验证结果。

b 尝试创建另一个表 TABLE2。删除创建的第一个表 TABLE1。结果如何？

5 删除表空间 RONLY 及关联的数据文件。验证结果。

6 只在内存中将 DB\_CREATE\_FILE\_DEST 设置为 \$HOME/ORADATA/u05。创建大小为 5 MB 的表空间 DATA03。不要指定文件位置。验证数据文件的创建结果。

# 9

## 存储结构和关系

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 描述数据库的逻辑结构
- 列出段类型及其用途
- 列出控制块空间使用率的关键字
- 从数据字典获取有关存储结构的信息

ORACLE

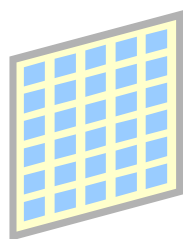
# 存储和关系结构

|                       |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
|-----------------------|--|---------------|--|---------------------|--|-------------|---------------------|--------------------|--|----------------------------------|--|--------------|--------------------|-----------|--|------------------|--|------------------|--|-----------|--|-------------|--|
| Database              |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
| PROD                  |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
| TABLESPACES           |  |               |  | USER_DATA           |  |             |                     |                    |  | RBS                              |  |              |                    | TEMP      |  |                  |  |                  |  |           |  |             |  |
| SYSTEM                |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
| DATAFILES             |  |               |  | DISK2/<br>USER1.dbf |  |             | DISK3/<br>USER2.dbf |                    |  | DISK1/<br>ROLL1.dbf              |  |              | DISK1/<br>TEMP.dbf |           |  |                  |  |                  |  |           |  |             |  |
| SEGMENTS              |  |               |  | S_DEPT              |  | S_EMP       |                     | S_DEPT<br>(cont'd) |  | S_EMP<br>FIRST N<br>AME<br>Index |  | RBS1         |                    | RBS2      |  | RBS1<br>(cont'd) |  | RBS2<br>(cont'd) |  | Temp      |  |             |  |
| D.D.<br>Table         |  | D.D.<br>Index |  | RB<br>Seg           |  | Data<br>Seg |                     | Data<br>Seg        |  | Data<br>Seg                      |  | Index<br>Seg |                    | RB<br>Seg |  | RB<br>Seg        |  | RB<br>Seg        |  | RB<br>Seg |  | Temp<br>Seg |  |
| EXTENTS               |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
| 1 2 1 2 1 2 1 1 2 2 1 |  |               |  | FREE 1 1 2 2 1      |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |
| Oracle DATA BLOCKS    |  |               |  |                     |  |             |                     |                    |  |                                  |  |              |                    |           |  |                  |  |                  |  |           |  |             |  |

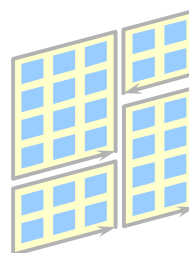
## 数据库体系结构

上一课讨论了数据库、其表空间和数据文件的存储结构。本课通过分析段、区和数据块来继续讨论数据库存储。

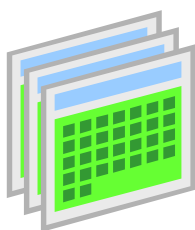
## 段类型



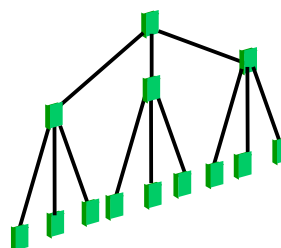
表



表分区



簇



索引

ORACLE

9-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### 段类型

段是数据库内占用空间的对象。它们使用数据库中数据文件内的空间。本部分介绍不同类型的段。

#### 表：

表是在数据库内存储数据的最常用方法。表段用于存储非集簇且未分区的表中的数据。表段中的数据并不按特定顺序存储，因此，数据库管理员 (DBA) 很难控制表中块内行的位置。表段中的所有数据都必须存储在一个表空间内。

#### 表分区：

当数据库内表的并发使用率很高时，主要关注点将是伸缩性和可用性。在这种情况下，表内数据可以存储在几个分区内，每个分区驻留在不同的表空间。Oracle 服务器当前支持通过键值范围、散列算法以及值列表来分区。表分区后，每个分区都是一个段，可以指定存储参数单独对它们进行控制。使用这种类型的段需要在 Oracle9i 企业版内选择分区 (Partitioning) 选件。



## **段类型（续）**

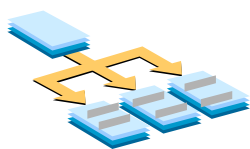
### **簇：**

簇与表一样，是一种数据段类型。簇内的行是基于键列值存储的。一个簇可以包含一个或多个表。一个簇内的表属于同一个段并且共享相同的存储特性。可以通过索引或者散列算法来访问集簇表内的行。

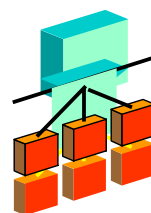
### **索引：**

一个特定索引的所有条目都存储在一个索引段内。如果一个表有三个索引，则使用三个索引段。使用索引段的目的是根据指定的关键字来查找行在表内的位置。

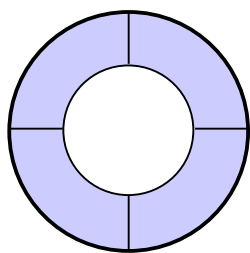
## 段类型



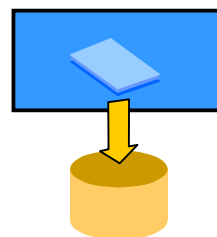
按索引组织的表



索引分区



还原段



临时段

ORACLE

9-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 段类型

#### 按索引组织的表：

在按索引组织的表内，数据基于键值存储在索引内。按索引组织的表无需在表中进行查找，因为所有数据都可以直接从索引树中检索到。

#### 索引分区：

索引可以分区并跨多个表空间。在这种情况下，索引内每个分区都对应一个段并且无法跨越多个表空间。分区索引的主要用途在于，通过分散索引输入/输出 (I/O) 来最大限度地降低争用。使用这种类型的段需要在 Oracle9i 企业版内选择分区 (Partitioning) 选项。

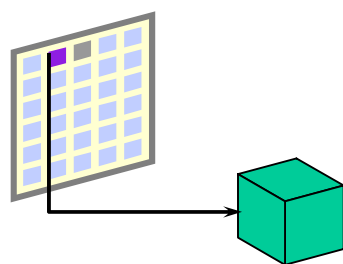
#### 还原段：

还原段由正在对数据库进行更改的事务处理使用。在更改数据或者索引块之前，旧值存储在还原段内。因此，允许用户还原所做的更改。

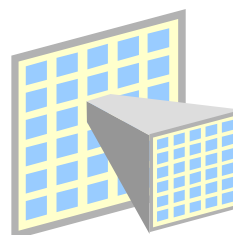
#### 临时段：

当用户执行 CREATE INDEX、SELECT DISTINCT 和 SELECT GROUP BY 等命令时，Oracle 服务器就会在内存中执行排序。如果排序所需空间大于内存中的可用空间，则将中间结果写入到磁盘上。临时段用来存储这些中间结果。

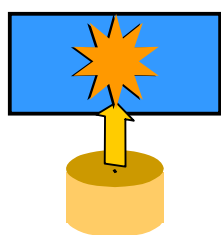
## 段类型



LOB 段



嵌套表



引导程序段

ORACLE

9-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### 段类型

#### LOB 段：

表中的一列或者多列可以用来存储大型对象 (LOB)，如文本文档、图像或者视频。如果列很大，Oracle 服务器将把这些值存储在独立的段（称为“LOB 段”）中。表中只包含一个定位器或者指针，指向对应的 LOB 数据所在的位置。

#### 嵌套表：

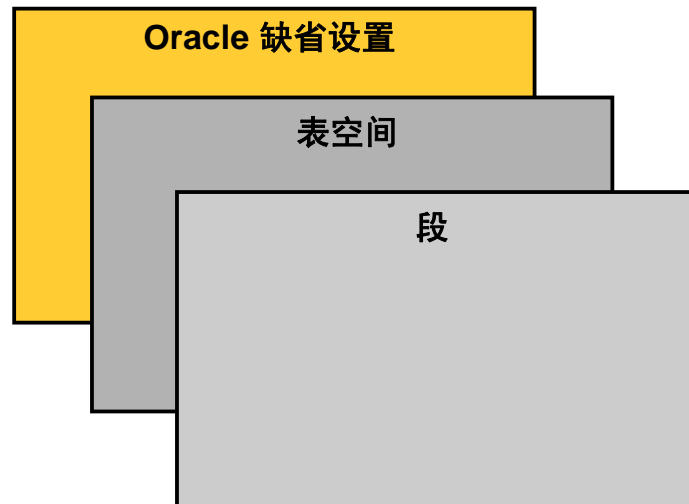
表中的列可以由用户定义的表构成，如订单中的项。在这种情况下，内表（即嵌套表）将存储为独立的段。

#### 引导程序段：

引导程序段，也称为高速缓存段，是在创建数据库时由 sql.bsq 脚本创建。在例程打开数据库时，该段可帮助初始化数据字典高速缓存。

引导程序段无法查询或者更新，并且不需要数据库管理员维护。

# 存储子句优先级



ORACLE

9-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## 存储参数

可以在段级别指定存储子句，以控制如何向段分配区。

- 除了 MINIMUM EXTENT 和 UNIFORM SIZE 表空间参数外，在段级别指定的任何存储参数覆盖在表空间级别设置的相应选项。
- 如果没有在段级别明确设置存储参数，那么存储参数缺省设置为在表空间级别所设置的值。
- 如果没有在表空间级别明确设置存储参数，那么将使用 Oracle 服务器系统的缺省设置。

### 其它注意事项：

- 如果存储参数改变，新选项只适用于尚未分配的区。
- 某些参数无法在表空间级别指定，而只能在段级别指定。
- 如果指定了表空间的最小区大小，则该大小适用于将来分配给该表空间内段的所有区。

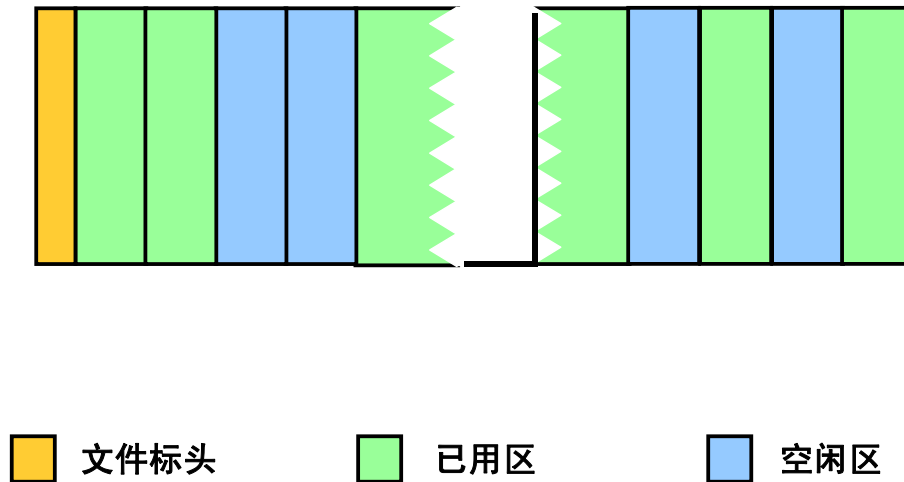
## 区的分配与回收

- 区是表空间内某个段使用的一块空间。
- 当段处于以下情况时分配区：
  - 已创建
  - 已扩展
  - 已改变
- 当段处于以下情况时回收区：
  - 已删除
  - 已改变
  - 已截断

ORACLE

## 已用区和空闲区

### 数据文件



### 区

创建表空间时，表空间中的数据文件包含一个标头，它是文件中的第一个块或前几个块。创建段时，从表空间的空闲区为这些段分配空间。段所使用的连续空间称为“已用区”。当段释放空间时，将把所释放的区添加到表空间的可用空闲区池中。

## 数据库块

- I/O 的最小单位
- 由一个或多个操作系统块组成
- 在创建表空间时设置
- DB\_BLOCK\_SIZE 指定了缺省块大小

ORACLE

## 多种块大小支持

- 数据库既可以按照标准的块大小创建，也可以按照非标准的块大小创建，非标准的块大小最多为四种。
- 块大小可以是介于 2 KB 和 32 KB 之间的 2 的幂值。

ORACLE

9-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 多种块大小支持

Oracle9i 支持创建具有多种块大小的数据库。此功能在下列情况下非常有用：

- 将表空间从联机事务处理 (OLTP) 数据库传送到企业数据仓库时。使用 Oracle9i，可方便地在具有不同块大小的数据库之间传送数据
- 要求能够在具有相应块大小的表空间中定位对象以最大限度地提高 I/O 性能时

SYSTEM 表空间的块大小指的是标准块大小。它是在创建数据库时设置的。使用 Oracle9i，除了标准的块大小外，还可以指定最多四种非标准的块大小。在初始化文件中，可以在缓冲区高速缓存中为每个这样的块大小配置子高速缓存。也可以在例程运行过程中配置子高速缓存。可以创建具有其中任意块大小的表空间。标准块大小用于系统表空间和大多数其它表空间。



## 标准块大小

- 在创建数据库时使用 **DB\_BLOCK\_SIZE** 参数设置；除非重新创建该数据库，否则无法更改
- 用于 **SYSTEM** 和 **TEMPORARY** 表空间
- **DB\_CACHE\_SIZE** 指定标准块大小的 **DEFAULT** 缓冲区高速缓存大小：
  - 最小大小 = 一个粒组（4 MB 或 16 MB）
  - 缺省值 = 48 MB

ORACLE

9-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 标准块大小

**DB\_BLOCK\_SIZE** 初始化参数用于指定数据库的标准块大小。该块大小用于 **SYSTEM** 表空间，以及任何临时表空间。除非进行指定，否则，标准块大小还用作表空间的缺省块大小。Oracle 最多支持四种附加的非标准块大小。

应该将最常用的块大小设置为标准块大小。在很多情况下，这是唯一需要指定的块大小。通常，将 **DB\_BLOCK\_SIZE** 设置为 4 KB 或 8 KB。如果未指定，则使用缺省数据块大小，缺省数据块大小取决于所用的操作系统，并且通常是合适的块大小。

除非重新创建数据库，否则在创建数据库后将不能更改块大小。

**DB\_CACHE\_SIZE** 初始化参数替代以前版本中使用的 **DB\_BLOCK\_BUFFERS** 初始化参数。**DB\_CACHE\_SIZE** 参数用于指定标准块大小缓冲区的高速缓存大小，其中的标准块大小是由 **DB\_BLOCK\_SIZE** 指定的。

为了保持向后兼容性，**DB\_BLOCK\_BUFFERS** 参数仍然可以使用，但它是一个静态参数，不能与任何可动态调整大小的参数一起使用。

**注：**粒组是一个连续虚拟内存分配单位。粒组的大小取决于估算的 SGA 的总大小，这个总大小是根据 **SGA\_MAX\_SIZE** 的参数值计算的：如果估算的 SGA 的大小 < 128 MB，则为 4 MB；否则为 16 MB。

## 非标准块大小

- 使用以下动态参数配置附加高速缓存：
  - `DB_2K_CACHE_SIZE` 用于 2 KB 块
  - `DB_4K_CACHE_SIZE` 用于 4 KB 块
  - `DB_8K_CACHE_SIZE` 用于 8 KB 块
  - `DB_16K_CACHE_SIZE` 用于 16 KB 块
  - `DB_32K_CACHE_SIZE` 用于 32 KB 块
- 如果 `nK` 是标准块大小，则不允许使用 `DB_nK_CACHE_SIZE`
- 每个高速缓存的最小大小：一个粒组

ORACLE

9-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### 非标准块大小

数据库缓冲区高速缓存初始化参数决定了 SGA 数据库缓冲区高速缓存组件的大小。可以使用这些参数为数据库使用的各种块大小指定高速缓存大小。如果要在数据库中使用多种块大小，则必须设置 `DB_CACHE_SIZE` 和至少一个 `DB_nK_CACHE_SIZE` 参数。每个参数为相应的块大小指定了缓冲区高速缓存大小。`DB_nK_CACHE_SIZE` 参数的缺省值为零。如果存在块大小为 `n KB` 的联机表空间，则不要将此参数设置为零。

平台特定的块大小具有一些限制。例如，如果平台上的最大块大小小于 32 KB，则不能设置 `DB_32K_CACHE_SIZE`。此外，如果最小块大小大于 2 KB，则不能设置 `DB_2K_CACHE_SIZE`。

**注：**这些参数不能用于调整标准块大小的高速缓存大小。例如，如果 `DB_BLOCK_SIZE` 的值为 2 KB，则设置 `DB_2K_CACHE_SIZE` 是非法的。标准块大小的高速缓存大小始终由 `DB_CACHE_SIZE` 的值确定。

粒组是一个连续的虚拟内存分配单位。粒组的大小取决于估算的 SGA 的总大小，这个总大小根据 `SGA_MAX_SIZE` 参数值进行计算。

## 非标准块大小

### 使用 Oracle Enterprise Manager 配置附加高速缓存

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “配置”(Configuration)。
2. 选择“全部初始化参数”(All Initialization Parameters)。
3. 对以下参数进行相应的更改:
  - DB\_2K\_CACHE\_SIZE
  - DB\_4K\_CACHE\_SIZE
  - DB\_8K\_CACHE\_SIZE
  - DB\_16K\_CACHE\_SIZE
  - DB\_32K\_CACHE\_SIZE
4. 单击“确定”(OK)。

## 创建非标准块大小的表空间

```
CREATE TABLESPACE tbs_1
DATAFILE 'tbs_1.dbf'
SIZE 10M BLOCKSIZE 4K;
```

```
DESCRIBE dba_tablespaces
Name Null? Type

TABLESPACE_NAME NOT NULL VARCHAR2(30)
BLOCK_SIZE NOT NULL NUMBER
...
```

ORACLE

9-16

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建非标准块大小的表空间

使用 BLOCKSIZE 子句为表空间指定非标准块大小。可以用字节或千字节（使用 K 后缀）指定大小。

要指定该子句，必须设置 DB\_CACHE\_SIZE 和至少一个 DB\_nK\_CACHE\_SIZE 参数，在该子句中指定的整数必须与某个 DB\_nK\_CACHE\_SIZE 参数的设置值对应。

**限制：**不能为临时表空间（即，如果同时指定了 TEMPORARY，或者要将该空间作为任何用户的临时表空间）指定非标准块大小。

上面的第一条语句创建一个名为 tbs\_1 的表空间，使用的数据文件 tbs\_1.dbf 具有 4 KB 的块大小。要成功执行这条语句，当前必须在缓冲区高速缓存中配置了 4 KB 大小的缓冲区。

**注：**已将一个新列添加到 \*\_TABLESPACES 字典视图中，以反映特定表空间中使用的相应块大小。

## 创建非标准块大小的表空间

### 使用 Oracle Enterprise Manager 创建非标准块大小的表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 要在创建表空间的过程中设置块大小，请选择“存储”(Storage) 页，并指定“块大小”(Block Size) 的值。
4. 单击“创建”(Create)。

The screenshot shows the 'Storage' tab in the Oracle Enterprise Manager console. It contains the following settings:

- Extent Management:** ☒ Locally managed ☐ Managed in the dictionary
- Allocation:** ☒ Automatic Allocation ☐ Uniform Allocation
- Size:** [ ] K Bytes
- Enable logging:** ☒ Yes - Generates redo logs and recoverable ☐ No - Faster updates, no redo logs generated and not recoverable
- Block Size:** [ 4096 ] Bytes

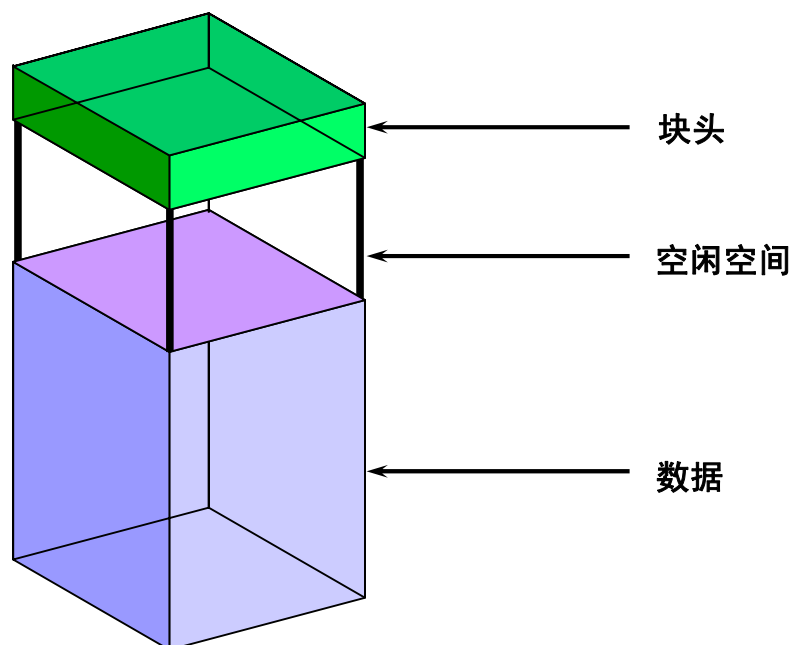
Buttons at the bottom: Create, Cancel, Show SQL, Help.

## 多种块大小的规则

- 分区对象的所有分区必须位于具有相同块大小的表空间中。
- 所有临时表空间必须采用标准块大小，包括用作缺省临时表空间的永久表空间。
- 按索引组织的表溢出和外部 **LOB** 段可以存储在块大小与基表不同的表空间中。

ORACLE

## 数据库块内容



ORACLE

9-19

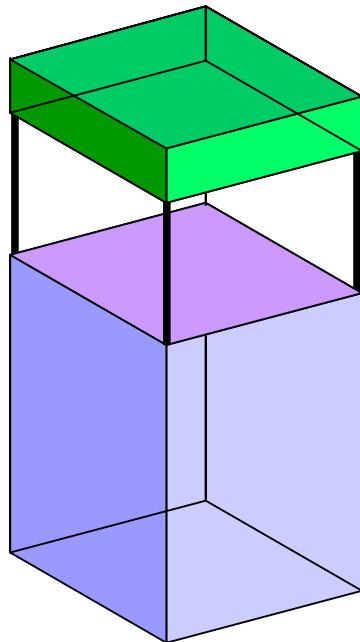
Copyright © Oracle Corporation, 2001. All rights reserved.

### 数据块

Oracle 数据块包含：

- 块头：块头包含数据块地址、表目录、行目录和事务处理对块中的行进行更改时所用的事务位置。块头从上往下增长。
- 数据空间：行数据从下往上插入到块中。
- 空闲空间：块内空闲空间位于块的中部。因此，必要时块头和行数据空间可以增长。最初，块内的空闲空间是连续的。但是，删除和更新会使块内的空闲空间产生碎片。Oracle 服务器可以在需要时合并块内的空闲空间。

## 块空间使用参数



**INITTRANS**

**MAXTRANS**

**PCTFREE**

**PCTUSED**

**ORACLE**

9-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### 块空间使用参数

块空间使用参数可用来控制对数据段和索引段空间的使用。

#### 控制并发性的参数：

**INITTRANS 和 MAXTRANS：**指定初始的和最大的事务位置数，这些事务位置在索引块或者数据块内创建。事务位置用来存储在某一时间点对块进行更改的事务处理的有关信息。一个事务处理只占用一个事务位置，即使它正在更改多行或者多个索引条目。

**INITTRANS：**保证最低级别的并发性。对于数据段和索引段，INITTRANS 的缺省值分别为 1 和 2，以保证最低级别的并发性。例如，如果 INITTRANS 设为 3，则保证至少有 3 个事务处理可以同时对该块进行更改。如果需要，也可以从块空闲空间内分配更多事务位置，以允许更多的事务处理并发修改块内的行。

**MAXTRANS：**缺省值为 255，它用于设置可更改数据块或者索引块的并发事务处理数的限制。进行设置后，该值将限制事务位置对空间的使用，从而保证块内有足够的空间供行或者索引数据使用。



## 块空间使用参数（续）

### 控制数据空间使用的参数：

**PCTFREE**：对于数据段而言，此参数用于指定每个数据块中保留空间所占的百分比，保留空间用于因更新块内的行而导致的生长。**PCTFREE** 的缺省值为 10%。

**PCTUSED**：对于数据段而言，此参数表示 Oracle 服务器试图为表内每个数据块所保持的已用空间的最低百分比。如果一个块的已用空间低于 **PCTUSED**，则将该块放回到空闲列表中。段的空闲列表是容纳将来所插入内容的候选块的列表。缺省情况下，每个段在创建时都有一个空闲列表。通过设置存储子句的 **FREELISTS** 参数，可以创建有更多空闲列表的段。**PCTUSED** 的缺省值为 40%。

**PCTFREE** 和 **PCTUSED** 都按可用数据空间百分比来计算，可用数据空间是从整个块大小减去块头空间后剩余的块空间。

注：“管理索引”一课详细讨论了这些参数在索引中的使用。

*Oracle9i：性能优化* 课程详细讨论了如何指定 **FREELISTS**。

# 数据块管理

可以使用两种方法来管理数据块：

- 自动段空间管理
- 手动管理

ORACLE

## 自动段空间管理

- 一种在数据库段内管理空闲空间的方法。
- 对段内空闲和已用空间的跟踪是使用位图完成的（与使用空闲列表相对）。
- 此方法提供了：
  - 更方便的管理
  - 更高的空间使用率
  - 改进的并发 INSERT 操作性能
- 限制：不能用于包含 LOB 的表空间。

ORACLE

9-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动段空间管理

#### 易于使用

PCTUSED、FREELISTS、FREELIST GROUPS 均是自动管理的。

#### 更高的空间使用率

所有对象都可以更有效地使用空间，尤其是行大小变化很大的对象。

#### 改进的并发处理性能

改进了对并发访问变化的运行时调整。

## 自动段空间管理

- 位图段包含一个位图，它描述了与段中的可用空间相关的每个块的状态。
- 该映射包含在单独的一组块中，这些块称为“位图块”(BMB)。
- 插入新行时，服务器就会在该映射中搜索具有足够空间的块。
- 当块中的可用空间数量发生变化时，位图中就会反映出它的新状态。

ORACLE

## 配置自动段空间管理

- 自动段空间管理仅能在表空间级别启用，用于在本地管理的表空间。

```
CREATE TABLESPACE data02
DATAFILE '/u01/oradata/data02.dbf' SIZE 5M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K
SEGMENT SPACE MANAGEMENT AUTO;
```

- 创建表空间后，这些规格将应用于在该表空间中创建的所有段。

ORACLE

9-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### 配置自动段空间管理

位图段是通过 CREATE TABLESPACE 命令的 SEGMENT SPACE MANGEMENT AUTO 子句指定的，此后不能更改这些段。如果定义了 PCTUSED、FREELIST 和 FREELIST GROUPS，则将其全部忽略。

可以用位图管理的段为：规则表、索引、按索引组织的表 (IOT) 以及 LOB。

# 手动数据块管理

- 允许使用参数手动配置数据块，例如：
  - PCTFREE
  - PCTUSED
  - FREELIST
- 在以前的 Oracle 版本中，这是唯一可用的方法

ORACLE

9-26

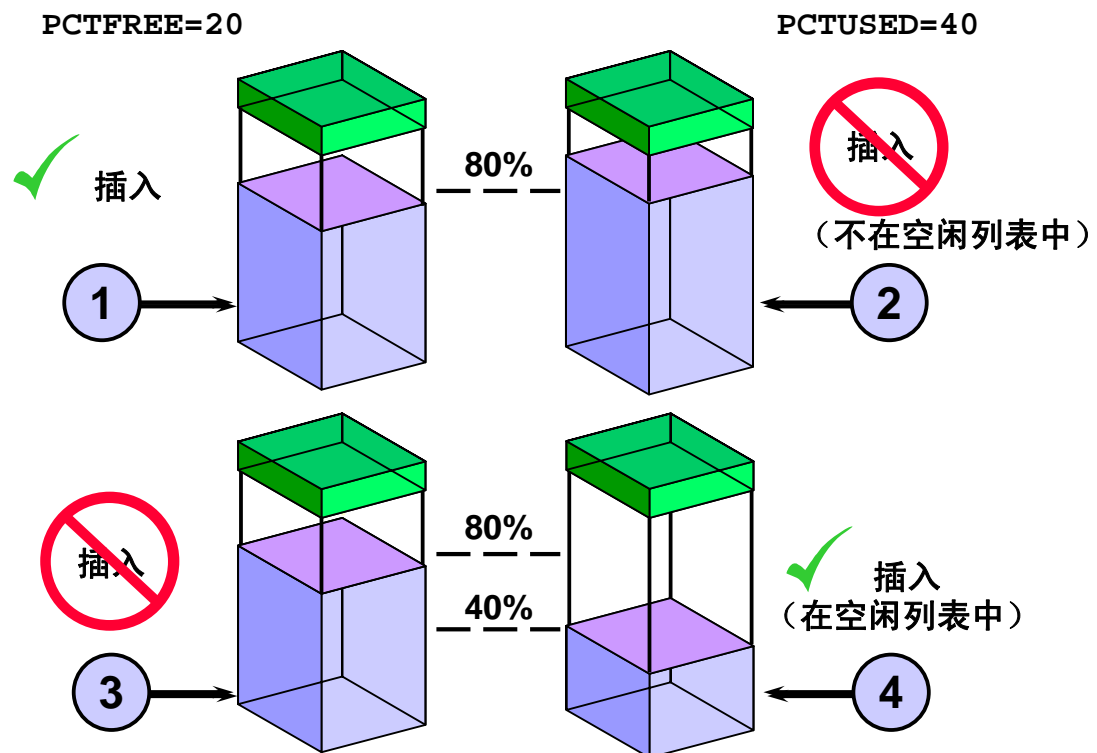
Copyright © Oracle Corporation, 2001. All rights reserved.

## 手动数据块管理

使用手动数据块管理，可以配置如何使用块空间以及块何时可用。可以在手动管理中使用 PCTFREE、PCTUSED 和 FREELIST 等参数。在以前的版本中，这是管理数据块的唯一方法。

手动方法是缺省设置。

## 块空间使用率



ORACLE

9-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### 块空间使用率

下面步骤介绍对于 `PCTFREE=20` 且 `PCTUSED=40` 的数据段如何管理块内的空间：

1. 向块中插入行，直到块内的空闲空间等于或者小于 20%。当行所占用的块内可用数据空间达到 80% ( $100 - \text{PCTFREE}$ ) 或者更多后，则无法在该块内插入。
2. 剩余的 20% 可在行大小增长时使用。例如，更新初始为 `NULL` 的列并分配一个值。这样，更新后的块使用率可能超过 80%。
3. 如果由于更新，删除了块内的行或者行大小减少，块使用率可能跌至 80% 以下。但是，仍然无法向块中插入，直到块使用率跌至 `PCTUSED` 以下，在本例中 `PCTUSED` 为 40%。
4. 当块使用率跌至 `PCTUSED` 以下时，该块可用于插入。随着向块内插入行，块使用率增长，此时又重复从步骤 1 开始的循环。

注：设置 `PCTFREE` 和 `PCTUSED` 参数的原则将在关于表和索引的“管理表”和“管理索引”两课中分别进行详细介绍。

## 块空间使用率

### 使用 Oracle Enterprise Manger 设置 PCTFREE 和 PCTUSED

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 要在创建表的过程中设置 PCTFREE 和 PCTUSED，请选择“存储”(Storage) 页，并完成填写“空间使用率”(Space Usage) 中的各项。
4. 单击“创建”(Create)。

The screenshot shows the 'Storage' tab in the Oracle Enterprise Manager console. The 'Explicit' radio button is selected under the 'Extents' section. The 'Space Usage' section contains the following fields:

- Initial Size: [ ] K Bytes
- Next Size: [ ] K Bytes
- Increase Size by: [ ] %
- Minimum Number: [ ]
- Maximum Number: ☐ Unlimited ☒ Value [ 0 ]
- Percentage free space reserved for updates: [ 20 ]
- Percentage used space threshold for row insertion candidacy: [ 40 ]
- Number of Transactions: Initial: [ ] Maximum: [ ]
- Free Lists: Free Lists: [ ] Groups: [ ]
- Buffer Pool: [ DEFAULT ]

At the bottom, there are buttons for 'Create', 'Cancel', 'Show SQL', and 'Help'.



## 获取存储信息

可以通过查询以下视图来获取有关存储的信息：

- **DBA\_EXTENTS**
- **DBA\_SEGMENTS**
- **DBA\_TABLESPACES**
- **DBA\_DATA\_FILES**
- **DBA\_FREE\_SPACE**

ORACLE

9-29

Copyright © Oracle Corporation, 2001. All rights reserved.

### 查询数据字典

表空间、数据文件、段和空闲区与已用区之间的关系可通过查询数据字典查看。

当创建具有一个或多个文件的表空间时，系统将在 DBA\_TABLESPACES 中添加一行。对于数据库中的每个文件，系统都会在 DBA\_DATA\_FILES 中添加一行。在此阶段，每个数据文件内的空间（除了文件标头）在 DBA\_FREE\_SPACE 中均显示为一个空闲区。

创建一个段后，在 DBA\_SEGMENTS 内将看到一行。可从 DBA\_EXTENTS 查看为该段中的区分配的空间，同时调整 DBA\_FREE\_SPACE 以显示为段创建的区所在文件内的空闲空间减少了。

文件内的所有空间（除了标题块）必须在 DBA\_FREE\_SPACE 或 DBA\_EXTENTS 中说明。

## 查询数据字典（续）

### DBA\_SEGMENTS 视图:

查询 DBA\_SEGMENTS 视图以获得分配给某个段的区和块的数目。

```
SQL> SELECT segment_name,tablespace_name,extents,blocks
2 FROM dba_segments
3 WHERE owner = 'HR';
```

```
SEGMENT_NAME TABLESPACE EXTENTS BLOCKS
```

```

```

```
REGIONS SAMPLE 1 8
LOCATIONS SAMPLE 1 8
DEPARTMENTS SAMPLE 1 8
JOBS SAMPLE 1 8
EMPLOYEES SAMPLE 1 8
JOB_HISTORY SAMPLE 1 8
5 rows selected.
```

### DBA\_EXTENTS 视图:

使用 DBA\_EXTENTS 视图以检查给定段的区。

```
SQL> SELECT extent_id,file_id,block_id,blocks
2 FROM dba_extents
3 WHERE owner='HR'
```

```
4 AND segment_name='EMPLOYEES';
```

```
EXTENT_ID FILE_ID BLOCK_ID BLOCKS
```

```

```

```
0 4 2 5
1 4 27 5
2 4 32 10
3 4 42 15
4 4 57 20
5 rows selected.
```

## 查询数据字典（续）

### DBA\_FREE\_SPACE 视图：

使用 DBA\_FREE\_SPACE 视图以检查给定段的区。

```
SQLPLUS> SELECT tablespace_name, count(*),
 2> max(blocks), sum(blocks)
 3> FROM dba_free_space
 4> GROUP BY tablespace_name;
```

| TABLESPACE_NAME | COUNT(*) | MAX(BLOCKS) | SUM(BLOCKS) |
|-----------------|----------|-------------|-------------|
| DATA01          | 2        | 1284        | 1533        |
| RBS             | 3        | 2329        | 2419        |
| SORT            | 1        | 1023        | 1023        |
| SYSTEM          | 1        | 5626        | 5626        |
| TEMP            | 1        | 2431        | 2431        |

5 rows selected.

## 小结

在这一课中，您应该能够掌握：

- 使用表空间执行以下操作：
  - 将段分开以便于管理
  - 控制用户的空间分配
- 按段中存储的信息类型对段进行分类
- 使用存储子句确定区大小
- 控制块空间的使用

ORACLE

## 练习 9 概览

此练习涉及以下主题：

- 了解和获取有关数据库中各种存储结构类型的信息

ORACLE

9-33

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 9 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 9：存储结构和关系

- 1 以用户 SYSTEM 的身份运行 lab09\_01.sql 脚本以创建表和索引。
- 2 标识数据库中不同类型的段。
- 3 编写一条查询语句，检查在哪些段中还有不足五个区就要达到最大区数。忽略引导程序段。在确定将来数据加载期间有可能产生错误的段时，该查询非常有用。
- 4 哪些文件已为 EMP 表分配了空间？
- 5 运行 lab09\_05.sql 脚本。
- 6 列出表空间的可用空闲空间。该查询应显示每个表空间中的碎片数、总空闲空间及最大空闲区。
- 7 列出在尝试分配附加的区时因空间不足而产生错误的段。

# 10

## 管理还原数据

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 说明还原数据的用途
- 进行自动还原管理
- 创建和配置还原段
- 从数据字典获取还原段信息

ORACLE



# 管理还原数据

- 管理还原数据的方法有两种：
  - 自动还原管理
  - 手动还原管理
- “还原”一词在以前的版本中称为“回退”。

ORACLE

10-3

Copyright © Oracle Corporation, 2001. All rights reserved.

## 管理还原数据

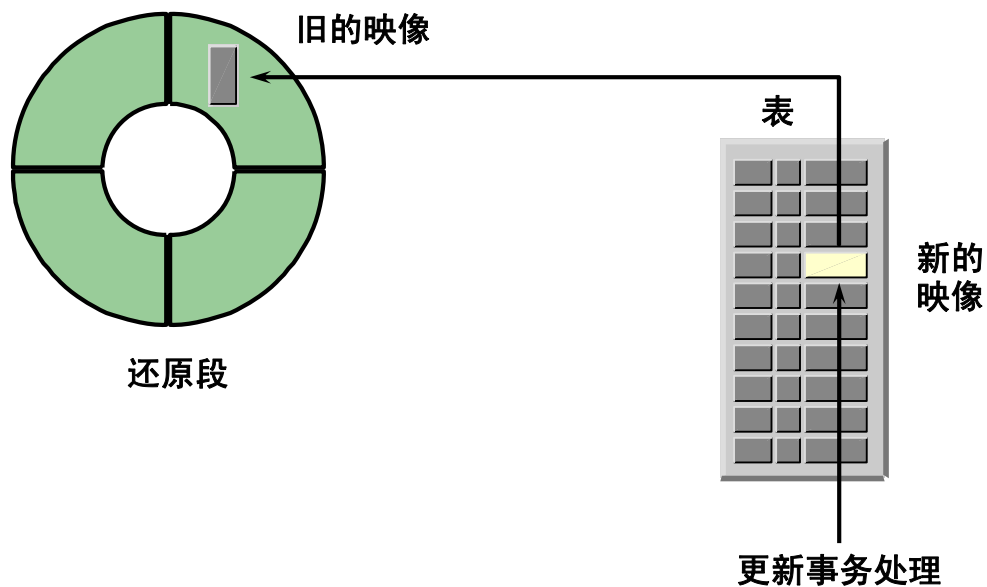
### 自动还原管理

Oracle 服务器可以自动管理还原段的创建、分配和优化。

### 手动还原管理

您可以手动管理还原段的创建、分配和优化。在 Oracle9i 以前的版本中，这是唯一可用的方法。有关手动还原管理方法的信息，请参考“附录 B：手动管理还原数据”。

## 还原段



ORACLE

10-4

Copyright © Oracle Corporation, 2001. All rights reserved.

### 还原段

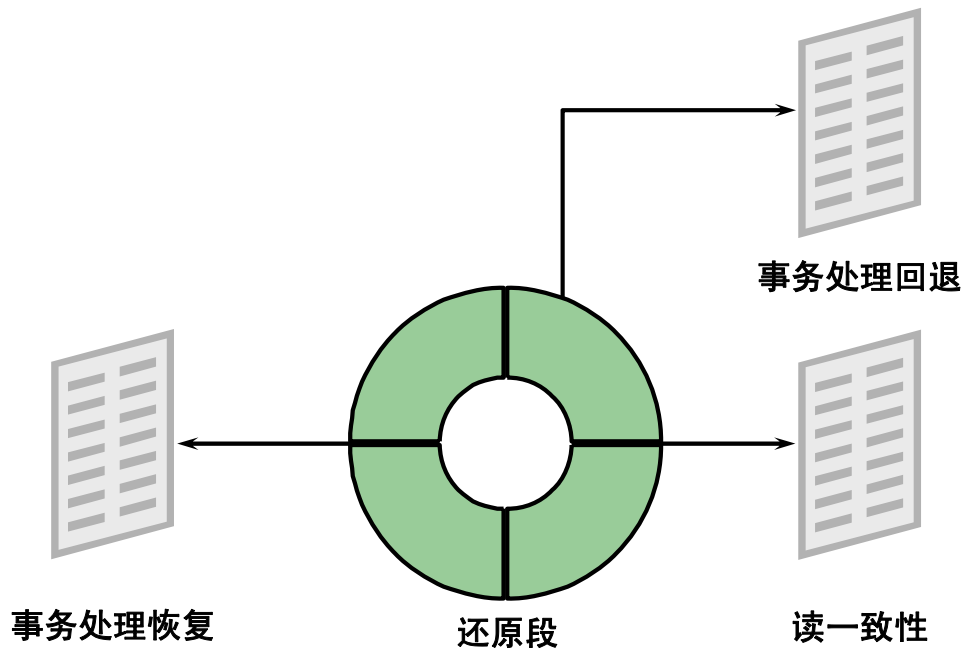
还原段用于在进程更改数据库中的数据时保存旧值（要还原的数据）。它按数据被修改之前的原样存储数据的位置及数据本身。

还原段的标头包含一个事务处理表，该表中存储着有关使用这个还原段的当前事务处理的信息。

一个连续的事务处理只使用一个还原段存储它的全部还原数据。

许多并发事务处理可以写入一个还原段。

## 还原段：用途



ORACLE

10-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### 还原段：用途

#### 事务处理回退

当某事务处理修改表中某行时，被修改的列的旧映像（要还原的数据）将存储在还原段中。如果将该事务处理回退，则 Oracle 服务器通过将还原段中的值写回到该行来恢复原始值。

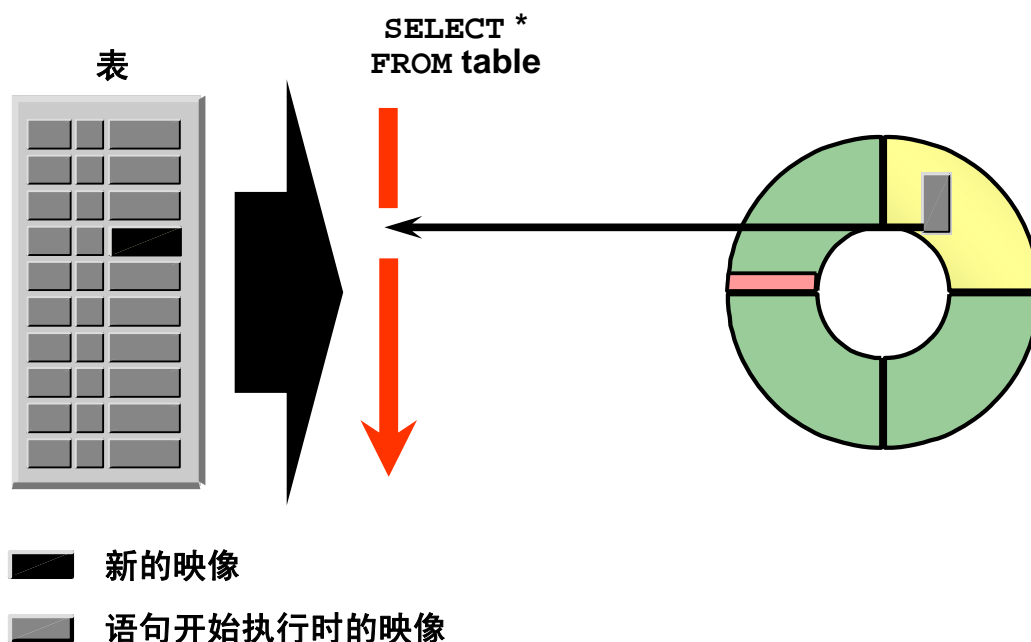
#### 事务处理恢复

如果例程在事务处理正在进行时失败，那么 Oracle 服务器需要在数据库再次打开时还原所有未提交的更改。这种回退操作是事务处理恢复的一部分。之所以有可能恢复事务处理，原因在于对还原段所做的更改同样受重做日志文件的保护。

#### 读一致性

在事务处理正在进行时，数据库中的其他用户不应看到这些事务处理所做的任何未提交更改。此外，也不应从某条语句中看到该语句开始执行后所提交的任何更改。还原段中的旧值（要还原的数据）也可用于为读者提供给定语句的一致映像。

## 读一致性



ORACLE

10-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 读一致性

Oracle 服务器保证一条语句所看到的数据来自一致的时间，即使其它事务处理修改了该数据。

当 Oracle 服务器开始执行 SELECT 语句时，它确定当前系统更改号 (SCN)，并确保这个 SCN 之前未提交的任何更改不会被这条语句处理。请考虑在进行多个更改的同时执行长时间运行的查询的情况。如果在这次查询开始时某行有未提交的更改，Oracle 服务器会构建该行的读一致性映像，方法是从还原段检索这些更改的前像，并将更改应用于内存中该行的副本。

#### 事务处理读一致性

始终为 SQL 语句提供读一致性。但是，您可以为只读事务处理请求读一致性，方法是在事务处理开始时发出下列命令：

```
SQL> SET TRANSACTION READ ONLY;
```

或者，可以为执行 DML 的事务处理请求读一致性，方法是在事务处理开始发出下列命令：

```
SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

无论是以上哪种情况，Oracle 服务器都提供从事务处理开始时就具有读一致性的数据。使用 SERIALIZABLE 会对性能造成负面影响。

## 还原段的类型

- **SYSTEM:** 用于 **SYSTEM** 表空间中的对象
- **非 SYSTEM:** 用于其它表空间中对象，包括：
  - 自动模式：需要 **UNDO** 表空间
  - 手动模式：
    - 专用：由一个例程获取
    - 公用：由任何例程获取
- **延迟：** 表空间处于立即脱机或临时脱机时使用，或用于恢复

ORACLE

10-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### 还原段的类型

#### **SYSTEM** 还原段

创建数据库时，将在 **SYSTEM** 表空间中创建 **SYSTEM** 还原段。这个还原段只用于对 **SYSTEM** 表空间中的对象所做的更改。**SYSTEM** 还原段在手动模式和自动模式下的存在和工作是一样的。

#### **非 SYSTEM** 还原段

具有多个表空间的数据库至少需要一个非 **SYSTEM** 还原段用于手动模式，或至少需要一个 **UNDO** 表空间用于自动模式。

#### 手动模式

在手动模式下，由数据库管理员创建的非 **SYSTEM** 还原段可用于对任何非 **SYSTEM** 表空间中的对象所做的更改。非 **SYSTEM** 还原段有以下两种类型。

#### 专用

因为专用还原段列在参数文件中，所以它们是通过例程联机的段。但是，通过发出 **ALTER ROLLBACK SEGMENT** 命令，可以使它们显式联机。

## 还原段的类型（续）

### 公用

公用还原段形成了数据库中可用还原段的池。公用还原段通常与 Oracle Real Application Clusters 一起使用来创建还原段池，这个池可以由任何“实时应用集群”(Real Application Clusters) 例程使用。

**注：**在 *Oracle9i Real Application Clusters and Administration* 手册中讨论了公用还原段的用法。

### 延迟还原段

当表空间脱机时，可能会创建延迟还原段。它们用于在表空间恢复联机时回退事务处理。当不再需要这些延迟还原段时，它们将自动被删除。

因为延迟还原段是由 Oracle 服务器来维护的，所以您无需进行维护。

## 自动还原管理： 概念

- 还原数据是使用 UNDO 表空间来管理的。
- 您可以为每个例程分配一个 UNDO 表空间，还要针对例程的工作量分配足够的空间。
- **Oracle** 服务器自动维护 UNDO 表空间内的还原数据。

ORACLE

10-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：概念

还原段是按照下面的命名约定创建的：

`_SYSSMUn$`

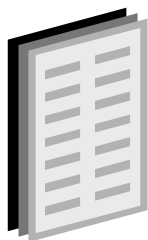
例如：

`_SYSSMU1$`

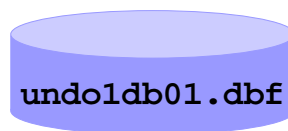
`_SYSSMU2$`

## 自动还原管理： 配置

- 配置初始化文件中的两个参数：
  - UNDO\_MANAGEMENT
  - UNDO\_TABLESPACE
- 至少创建一个 UNDO 表空间。



初始化文件



UNDO 表空间

ORACLE

10-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：配置

如果数据库中只有一个 UNDO 表空间，并且将 UNDO\_MANAGEMENT 设置为 AUTO，则 UNDO\_TABLESPACE 参数是可选的；Oracle 服务器将自动选择 UNDO 表空间。



## 自动还原管理： 初始化参数

- **UNDO\_MANAGEMENT**: 指定系统应该使用 AUTO 模式还是 MANUAL 模式
- **UNDO\_TABLESPACE**: 指定要使用的特定 UNDO 表空间

```
UNDO_MANAGEMENT=AUTO
UNDO_TABLESPACE=UNDOTBS
```

ORACLE

10-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：初始化参数

#### **UNDO\_MANAGEMENT 参数:**

UNDO\_MANAGEMENT 参数决定数据库的还原模式。该参数可以设置为 AUTO 和 MANUAL 这两个值中的任一个值，并且必须在初始化参数文件中设置。UNDO\_MANAGEMENT 不能在数据库启动后进行动态更改。AUTO 模式可以将数据库设置为自动还原管理，并需要 UNDO 表空间。在 MANUAL 模式（缺省值）下，可以根据需要在数据库中创建和管理还原段，这与以前的 Oracle 服务器版本中的操作相同。

#### **UNDO\_TABLESPACE 参数:**

指定要使用的 UNDO 表空间。此参数可以在初始化文件中设置，或使用 ALTER SYSTEM 命令来动态改变。

```
SQL> ALTER SYSTEM SET undo_tablespace = UNDOTBS;
```

## 自动还原管理： UNDO 表空间

在创建数据库时一起创建 UNDO 表空间，方法是，在 CREATE DATABASE 命令中添加一个子句

```
CREATE DATABASE db01
. . .
UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undodb01.dbf' SIZE 20M
AUTOEXTEND ON
```

或者稍后使用 CREATE UNDO TABLESPACE 命令创建它

```
CREATE UNDO TABLESPACE undo1
DATAFILE '/u01/oradata/undodb01.dbf'
SIZE 20M;
```

ORACLE

10-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：UNDO 表空间

自动还原管理需要一个 UNDO 表空间。数据库中可能有多个 UNDO 表空间，但只能有一个 UNDO 表空间处于活动状态。

您可以在创建数据库时一起创建 UNDO 表空间，方法是，在 CREATE DATABASE 命令中添加一个子句。

在创建数据库期间，如果将 UNDO\_MANAGEMENT 参数设置为 AUTO 并在 CREATE DATABASE 语句中省略了 UNDO 表空间子句，那么 Oracle 服务器将创建一个名为 SYS\_UNDOTBS 的 UNDO 表空间。数据文件表空间 SYS\_UNDOTS 的缺省数据文件名称是 'dbul<oracle\_sid>.dbf'。该文件位于 \$ORACLE\_HOME/dbs 中。文件的大小取决于操作系统。将 AUTOEXTEND 设置为 ON。

创建完数据库后，您可以使用 CREATE UNDO TABLESPACE 命令创建 UNDO 表空间。

## 自动还原管理：UNDO 表空间（续）

### 使用 Oracle Enterprise Manager 创建 UNDO 表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “表空间”(Tablespaces)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General) 选项卡中，输入文件名和文件大小。
4. 在“类型”(Type) 区域中，选择“还原”(Undo)。
5. 单击“创建”(Create)。

## 自动还原管理： 改变 UNDO 表空间

- 使用 ALTER TABLESPACE 命令，可以对 UNDO 表空间进行更改。
- 在下面的示例中，向 UNDO 表空间添加了其它数据文件：

```
ALTER TABLESPACE undotbs
ADD DATAFILE '/u01/oradata/undotbs2.dbf'
SIZE 30M
AUTOEXTEND ON;
```

ORACLE

10-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：改变 UNDO 表空间

改变 UNDO 表空间时，服务器为下列子句提供支持。

- ADD DATAFILE
- RENAME
- DATAFILE [ ONLINE | OFFLINE ]
- BEGIN BACKUP
- END BACKUP

## 自动还原管理：改变 UNDO 表空间（续）

### 使用 Oracle Enterprise Manager 改变 UNDO 表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage)。
2. 选择“表空间”(Tablespace) 文件夹，然后用鼠标右键单击 UNDO 表空间。
3. 选择“添加数据文件”(Add a data file)。

## 自动还原管理： 切换 UNDO 表空间

- 可以从使用一个 UNDO 表空间切换到使用另一个 UNDO 表空间。
- 一次只能将一个 UNDO 表空间分配给某个数据库。
- 一个例程中可以存在多个 UNDO 表空间，但只能有一个处于活动状态。
- 使用 ALTER SYSTEM 命令，可以在各个 UNDO 表空间之间进行动态切换。

```
ALTER SYSTEM SET UNDO_TABLESPACE=UNDOTBS2;
```

ORACLE

## 使用 Oracle Enterprise Manager 切换 UNDO 表空间

### 使用 Oracle Enterprise Manager 切换 UNDO 表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage) > “例程”(Instance)。
2. 单击“配置”(Configuration)。
3. 在“还原”(Undo) 页上, 从“当前还原表空间”(Current Undo Tablespace) 下拉列表中选择 UNDO 表空间。
4. 单击“应用”(Apply)。

## 自动还原管理： 删除 UNDO 表空间

- 使用 DROP TABLESPACE 命令，可以删除 UNDO 表空间。

```
DROP TABLESPACE UNDOTBS2;
```

- 某个 UNDO 表空间只有在当前未由任何例程使用的时候才能被删除。
- 要删除活动的 UNDO 表空间，请执行以下操作：
  - 切换到新的 UNDO 表空间
  - 完成当前所有事务处理后，删除该表空间

ORACLE

10-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：删除 UNDO 表空间

删除 UNDO 表空间时，该表空间不能再被例程使用，表空间内的所有事务处理必须均已完成。

如果表空间 UNDOTBS 是数据库当前活动的 UNDO 表空间，并且将被删除，那么在删除该表空间前必须设置一个新的 UNDO 表空间。如果某个 UNDO 表空间已不再存在，先创建另一个 UNDO 表空间。然后，使用 ALTER SYSTEM 命令更改当前的 UNDO 表空间。

```
SQL> ALTER SYSTEM SET undo_tablespace = UNDOTBS2;
```

您可以在表空间 UNDOTBS 内的所有事务处理都已完成后删除它。要确定是否存在任何一个活动的事务处理，请使用以下查询：

```
SQL> SELECT a.name,b.status
2 FROM v$rollname a, v$rollstat b
3 WHERE a.name IN (SELECT segment_name
4 FROM dba_segments
5 WHERE tablespace_name = 'UNDOTBS')
6 AND a.usn = b.usn;
```

| NAME       | STATUS          |
|------------|-----------------|
| -----      | -----           |
| _SYSSMU4\$ | PENDING OFFLINE |



### 自动还原管理：删除 UNDO 表空间（续）

状态为 PENDING OFFLINE 的某个还原段仍包含活动的事务处理。如果查询没有返回任何行，则表明所有事务处理均已完成，并且可以使用以下命令删除该表空间。

```
SQL> DROP TABLESPACE UNDOTBS;
```

切换到另一个 UNDO 表空间后，Oracle 服务器可以引用表空间 UNDOTBS 为查询提供读一致性。当表空间 UNDOTBS 无法再提供读一致性后，需要该表空间中信息的查询就会收到以下错误：ORA-1555 快照已过期。

## 自动还原管理：删除 UNDO 表空间（续）

### 使用 Oracle Enterprise Manager 删除 UNDO 表空间

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “存储”(Storage)。
2. 选择“表空间”(Tablespace)文件夹。
3. 右击 UNDO 表空间的名称，从弹出的菜单中选择“删除”(Remove)。
4. 确认删除。

## 自动还原管理： 其它参数

- **UNDO\_SUPPRESS\_ERRORS 参数：**
  - 将这个参数设置为 TRUE，可以避免试图以 AUTO 模式执行手动操作时产生错误。
- **UNDO\_RETENTION 参数：**
  - 此参数控制为提供读一致性而保留的还原数据量。

ORACLE

10-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：其它参数

#### UNDO\_SUPPRESS\_ERRORS 参数

使用 UNDO\_SUPPRESS\_ERRORS，用户可以避免在自动还原管理模式下执行手动还原管理模式操作（例如，ALTER ROLLBACK SEGMENT ONLINE）时出现错误。通过设置这个参数，用户可以在将所有应用程序和脚本转换成自动还原管理模式前使用还原表空间功能。例如，如果有一个使用 SET TRANSACTION USE ROLLBACK SEGMENT 语句的应用程序，则可以向该应用程序添加语句 ALTER SESSION SET UNDO\_SUPPRESS\_ERRORS = true 以避免 ORA-30019 错误。

ORA-30019：在自动还原模式下执行了非法的回退段

#### UNDO\_RETENTION 参数

确定还原数据的保留时间，以便提供读一致性。保留还原数据可以进行更长时间的查询，同时 UNDO 表空间所需的数据文件也越大。以秒为单位定义的 UNDO\_RETENTION 参数可以在初始化文件中设置，或使用 ALTER SYSTEM 命令来动态修改。

```
SQL> ALTER SYSTEM SET UNDO_RETENTION=900;
```

如果值为 900，可以使还原数据保留 15 分钟。

### **自动还原管理：其它参数（续）**

如果 UNDO 表空间设置得太小，则即使设置了 UNDO\_RETENTION，仍不能按指定的时间保留还原数据。Oracle 服务器使用一种算法在 UNDO 表空间内分配空间，并在产生新的事务处理失败之前，分配没有活动事务处理的未过期的空间。

## 还原数据统计信息

```
SELECT end_time,begin_time,undoblks
FROM v$undostat;
```

| END_TIME           | BEGIN_TIME         | UNDO |
|--------------------|--------------------|------|
| 22-JAN-01 13:44:18 | 22-JAN-01 13:43:04 | 19   |
| 22-JAN-01 13:43:04 | 22-JAN-01 13:33:04 | 1474 |
| 22-JAN-01 13:33:04 | 22-JAN-01 13:23:04 | 1347 |
| 22-JAN-01 13:23:04 | 22-JAN-01 13:13:04 | 1628 |
| 22-JAN-01 13:13:04 | 22-JAN-01 13:03:04 | 2249 |
| 22-JAN-01 13:03:04 | 22-JAN-01 12:53:04 | 1698 |
| 22-JAN-01 12:53:04 | 22-JAN-01 12:43:04 | 1433 |
| 22-JAN-01 12:43:04 | 22-JAN-01 12:33:04 | 1532 |
| 22-JAN-01 12:33:04 | 22-JAN-01 12:23:04 | 1075 |

ORACLE

10-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### 还原数据统计信息

#### V\$UNDOSTAT 视图:

此视图显示了统计数据的频率分布，以说明数据库的运行状况。视图中的每一行保留的是每隔 10 分钟就在例程中收集一次的统计数据。您可以使用此视图估算当前工作量所需的还原空间量。Oracle 服务器使用此视图优化系统对还原空间的使用。此视图在自动模式和手动模式下均可使用。

虽然时间间隔一般是 10 分钟，但最新的行自其时间间隔开始起返回的时间通常少于 10 分钟。

## 自动还原管理： 调整 UNDO 表空间的大小

确定 UNDO 表空间的大小时需要三条信息：

- (UR) 以秒为单位的 UNDO\_RETENTION
- (UPS) 每秒生成的还原数据块的数量
- (DBS) 开销随着区大小和文件大小 (db\_block\_size) 变化

$$\text{UndoSpace} = [\text{UR} * (\text{UPS} * \text{DBS})] + (\text{DBS} * 24)$$

ORACLE

10-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：调整 UNDO 表空间的大小

调整 UNDO 表空间的大小需要三个数据。其中两个数据可以从初始化文件中获取：UNDO\_RETENTION 和 DB\_BLOCK\_SIZE。第三个数据是公式，它需要对数据库执行查询。每秒生成的还原块的数量可以从 V\$UNDOSTAT 获得。下面的公式计算生成的还原块的总数，并用监视的时间长度（以秒计）除以得出的块总数：

```
SQL> SELECT (SUM(undoblks) / SUM) ((end_time - begin_time) *
86400) FROM v$undostat;
```

列 END\_TIME 和 BEGIN\_TIME 属于 DATE 数据类型。如果对 DATE 数据类型做减法，则计算结果是天数。要将天数转换成秒数，乘以一天所包含的秒数 86400 即可。

查询结果返回每秒生成的还原块的数量。这个值需要乘以还原块的大小，还原块大小与 DB\_BLOCK\_SIZE 中定义的数据库块的大小相同。下面的查询计算的是所需的字节数：

## 自动还原管理：调整 UNDO 表空间的大小（续）

```
SQL> SELECT (UR * (UPS * DBS)) + (DBS * 24) AS "Bytes"
 2 FROM (SELECT value AS UR
 3 FROM v$parameter
 4 WHERE name = 'undo_retention'),
 5 (SELECT (SUM(undoblks)/SUM(((end_time -
 begin_time)*86400))) AS UPS
 6 FROM v$undostat),
 7 (SELECT value AS DBS
 8 FROM v$parameter
 9 WHERE name = 'db_block_size');

 Bytes

19106213
```

要将字节数转换成兆字节数，请除以 1,048,576 字节。对于这个数据库，结果是 18.22 MB。

为了获得最佳结果，应该在一天中数据库负载最繁重的时候进行计算。

## 自动还原管理： 还原限额

- 长事务处理和未正确写入的事务处理会消耗宝贵的资源。
- 使用还原限额，可以对用户进行分组，并为用户组指定还原空间的最大限制。
- **UNDO\_POOL** 是“资源管理器” (Resource Manager) 指令，用于为资源组定义可用的空间量。
- 如果某个组超过其限制，这个组就不能进行任何新的事务处理，直到已完成或中止的当前事务处理释放了还原空间。

ORACLE

10-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### 自动还原管理：还原限额

使用资源计划，可以对用户进行分组，并对某个组可使用的资源数量加以限制。可以对某个组生成的还原数据量进行限制，方法是设置 **UNDO\_POOL** 的值：缺省值是无限大。如果某个组超过了它的限制，就会收到一个错误，同时这个组不能进行新的事务处理，直到当前事务处理完成或中止。

ORA-30027: “超出还原限额 – 无法获取 %s (字节)”

原因：已超出为此会话的使用者组分配的还原空间量。

操作：请求 **DBA** 增加还原限额，或者等待其它事务处理提交后再继续执行。

注：在“管理口令安全性和资源”一课中对“资源管理”进行了详细的讨论。



## 获取还原段信息

- 可以通过查询以下视图来获取有关还原段的信息：
  - DBA\_ROLLBACK\_SEGS
- 动态性能视图
  - V\$ROLLNAME
  - V\$ROLLSTAT
  - V\$UNDOSTAT
  - V\$SESSION
  - V\$TRANSACTION

ORACLE

10-27

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取还原段信息

要获取有关数据库中所有还原段的信息，请查询 DBA\_ROLLBACK\_SEGS 视图：

```
SQL> SELECT segment_name,owner,tablespace_name,status
 2 FROM dba_rollback_segs;
```

| SEGMENT_NAME | OWNER  | TABLESPACE | STATUS |
|--------------|--------|------------|--------|
| SYSTEM       | SYS    | SYSTEM     | ONLINE |
| _SYSSMU1\$   | PUBLIC | UNDO1      | ONLINE |
| _SYSSMU2\$   | PUBLIC | UNDO1      | ONLINE |
| _SYSSMU3\$   | PUBLIC | UNDO1      | ONLINE |
| _SYSSMU4\$   | PUBLIC | UNDO1      | ONLINE |

有关脱机的还原段的信息只能在 DBA\_ROLLBACK\_SEGS 中看到。动态性能视图只显示联机还原段。

OWNER 列指定还原段的类型：

- SYS：指专用还原段
- PUBLIC：指公用还原段

## 获取还原段信息（续）

### V\$ROLLSTAT 和 V\$ROLLNAME 视图：

联接 V\$ROLLSTAT 和 V\$ROLLNAME 视图以获取例程当前使用的还原段的统计数据。

示例：

```
SQL> SELECT n.name, s.extents, s.rssize,s.hwmsize,
 2 s.xacts, s.status
 3 FROM v$rollname n, v$rollstat s
 4 WHERE n.usn = s.usn;
```

| NAME       | EXTENTS | RSSIZE  | HWMSIZE | XACTS | STATUS |
|------------|---------|---------|---------|-------|--------|
| -----      | -----   | -----   | -----   | ----- | -----  |
| SYSTEM     | 7       | 425984  | 425984  | 0     | ONLINE |
| _SYSSMU1\$ | 5       | 2289664 | 2289664 | 0     | ONLINE |
| _SYSSMU2\$ | 10      | 6549504 | 6549504 | 0     | ONLINE |
| _SYSSMU3\$ | 7       | 4386816 | 4386816 | 0     | ONLINE |
| _SYSSMU4\$ | 6       | 4321280 | 4321280 | 0     | ONLINE |

### V\$TRANSACTION 和 V\$SESSION 视图：

要检查活动事务处理当前使用的还原段的情况，请联接 V\$TRANSACTION 和 V\$SESSION 视图。

示例：

```
SQL> SELECT s.username, t.xidusn, t.ubafil,
 2 t.ubablk, t.used_ublk
 3 FROM v$session s, v$transaction t
 4 WHERE s.saddr = t.ses_addr;
```

| USERNAME | XIDUSN | UBAFIL | UBABLK | USED_UBLK |
|----------|--------|--------|--------|-----------|
| -----    | -----  | -----  | -----  | -----     |
| HR       | 2      | 2      | 5005   | 1         |

## 小结

在这一课中，您应该能够掌握：

- 配置自动还原管理
- 创建 UNDO 表空间
- 正确调整还原表空间的大小

ORACLE®

## 练习 10 概览

此练习涉及以下主题：

- 创建 UNDO 表空间
- 在各个 UNDO 表空间之间进行切换
- 删除 UNDO 表空间

ORACLE

10-30

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 10 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成此处的练习。

## 练习 10：管理还原数据

- 1 以用户 SYSTEM/MANAGER 的身份进行连接，并列出表空间 UNDOTBS 中的还原段。
- 2 在 \$HOME/oradata/u03 中创建还原表空间 UNDO2，大小为 15 MB。列出表空间 UNDO2 中的回退段。
- 3 在一个新的 Telnet 会话中启动 SQL\*Plus 并以用户 HR 的身份进行连接，然后运行脚本 lab10\_03.sql 向表 DEPARTMENTS 插入行。不要提交、回退或退出该会话。
- 4 在以 SYS 身份进行连接的会话中，使用 ALTER SYSTEM 命令，将该例程的 UNDO 表空间从 UNDOTBS 切换到 UNDO2。
- 5 以 SYS 的身份删除表空间 UNDOTBS。结果如何？
- 6 列出表空间 UNDOTBS 中的还原段及其状态。将此列表与第 1 步中的列表进行比较。
- 7 在以 HR 身份进行连接的会话中，回退事务处理，然后退出会话。
- 8 在以 SYS 的身份连接的会话中，删除表空间 UNDOTBS。结果如何？
- 9 以 SYS 身份发出下列命令：

```
ALTER SYSTEM SET undo_retention=0 SCOPE=memory;
```

现在删除表空间 UNDOTBS。结果如何？

注：删除表空间之前，可能仍有延迟。



# 11

## 管理表

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

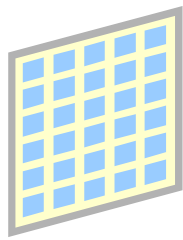
完成这一课的学习后，您应该能达到下列目标：

- 辨别各种存储数据的方法
- 概括各种 **Oracle** 数据类型
- 区分扩展的和受限的 **ROWID**
- 概括行的结构
- 创建规则表和临时表
- 管理表内的存储结构
- 重新组织、截断和删除表
- 删除表内的列

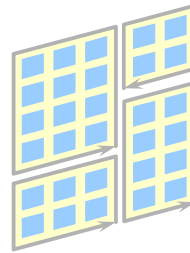
ORACLE



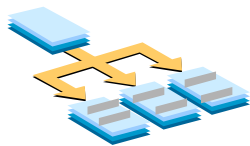
## 存储用户数据



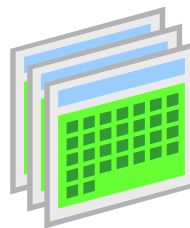
常规表



分区表



按索引组织的表



簇

ORACLE

11-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### 存储用户数据

在 Oracle 数据库中有几种存储用户数据的方法：

- 常规表
- 分区表
- 按索引组织的表
- 集簇表

注：分区表、按索引组织的表和集簇表在其它课程中进行阐述。

#### 常规表：

常规表（通常称为“表”）是存储用户数据最常用的形式。它是缺省表，并且是本课论述的重点。数据库管理员对表中行分布的控制很有限。行可能按任意顺序存储，具体顺序取决于在表中进行的操作。

## 存储用户数据（续）

### 分区表：

分区表使您可以生成可伸缩的应用程序。它具有以下特征：

- 每个分区表有一个或多个分区，每个分区存储已分区（使用范围分区、散列分区、组合分区或列表分区）的行。
- 分区表中的每个分区为一个段，可各自位于不同的表空间中。
- 对于能够同时使用几个进程进行查询或操作的大型表，分区非常有用。
- 有一些特殊的命令可用来管理一个表内的分区。

### 按索引组织的表：

按索引组织的表就像在一个或多个列中具有主键索引的堆表。但是，按索引组织的表并不为表和 B 树索引维护两个单独的存储空间，而是仅维护一个包含表主键和其它列值的 B 树。由于设置 PCTTHRESHOLD 值以及较长的行长度需要溢出区域，所以可能存在溢出段。

按索引组织的表为进行涉及精确匹配和范围搜索的查询，提供基于键的、对表数据的快速访问。

此外，存储要求也降低了，因为键列在表和索引中不重复。除非索引条目变得非常大，否则其余的非键列就存储在索引中；在此情况下，Oracle 服务器提供 OVERFLOW 子句来处理此问题。

## 存储用户数据（续）

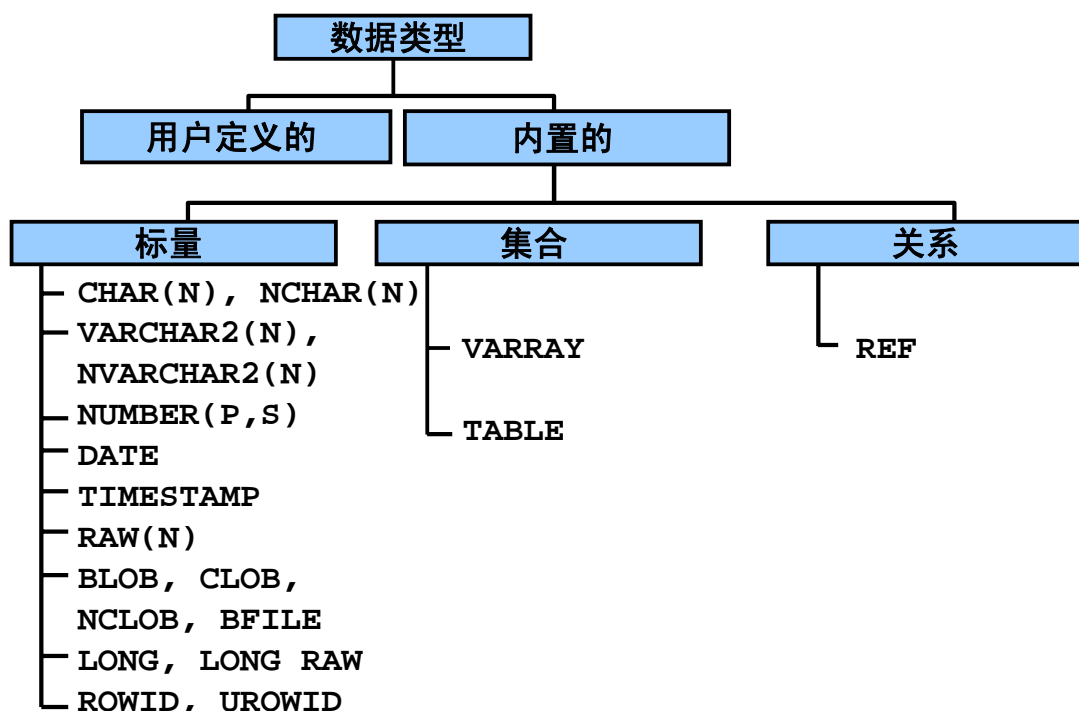
### 集簇表：

集簇表为存储表数据提供另一种可选的方法。簇由一个表或共享相同数据块的一组表构成，它们之所以被组织在一起，是因为它们共享共同的列并且经常一起使用。

簇具有以下特征：

- 簇有一个集簇键，用来标识需要存储在一起的多个行。
- 集簇键可由一个或多个列组成。
- 簇中的表具有与集簇键相对应的列。
- 集簇是一种对使用表的应用程序透明的机制。可以象操作存储在常规表中的数据那样操作集簇表中的数据。
- 更新集簇键中的一列可能需要移植该行。
- 集簇键独立于主键。簇中的表可有一个主键，它可以是集簇键，也可以是另一组列。
- 创建簇通常是为了改善性能。随机访问集簇数据更快，而对集簇表进行全表扫描通常较慢。
- 簇会重新规范表的物理存储，但不影响其逻辑结构。

# Oracle 内置数据类型



ORACLE

11-6

Copyright © Oracle Corporation, 2001. All rights reserved.

## Oracle 内置数据类型

Oracle 服务器提供几种内置数据类型来存储标量数据、集合和关系。

### 标量数据类型：

字符数据：字符数据可以作为长度固定或长度可变的字符串存储在数据库中。

长度固定的字符数据类型（如 CHAR 和 NCHAR）存储时带有填补空格。NCHAR 是“全球化支持”包含的一种数据类型，既可以存储宽度固定字符集，也可以存储宽度可变字符集。其最大大小取决于存储一个字符所需要的字节数，上限为每行 2,000 个字节。缺省值为1 个字符或 1 个字节，具体取决于字符集。

长度可变的字符数据类型仅使用存储实际列值所需要的字节数，并且每行的大小可以不同，最大可达 4,000 字节。VARCHAR2 和 NVARCHAR2 就是长度可变的字符数据类型的例子。

## Oracle 内置数据类型（续）

### 标量数据类型（续）：

数字数据类型：Oracle 数据库中的数字始终以长度可变的数据存储。最多可以存储 38 个有效数位。数字数据类型需要：

- 指数用 1 个字节
- 尾数中的每两个有效数位用 1 个字节
- 负数用 1 个字节（如果有效数位少于 38 个字节）

DATE 数据类型：Oracle 服务器将日期存储在包含七个字节的固定长度字段中。Oracle DATE 始终包括时间。

TIMESTAMP 数据类型：此数据类型存储日期和时间，包括零点几秒，最高可达 9 位小数。TIMESTAMP WITH TIME ZONE 和 TIMESTAMP WITH LOCAL TIME ZONE 可以使用时区设定时间，如夏时制。TIMESTAMP 和 TIMESTAMP WITH LOCAL TIME ZONE 可用于主键，而 TIMESTAMP WITH TIME ZONE 则不能。

RAW 数据类型：可以使用此数据类型存储小型二进制数据。在网络中的计算机之间传输 RAW 数据时，或者使用 Oracle 实用程序将 RAW 数据从一个数据库移到另一个数据库时，Oracle 服务器不执行字符集转换。存储实际列值所需要的字节数大小随每行大小而异，最多为 2,000 字节。

### LONG、LONG RAW 和大型对象 (LOB) 数据类型：

Oracle 为存储 LOB 提供六种数据类型：

- CLOB 和 LONG 用于存储大型的、宽度固定的字符数据
- NCLOB 用于存储大型的、宽度固定国家字符集数据
- BLOB 和 LONG RAW 用于存储非结构化数据
- BFILE 用于存储操作系统文件中的非结构化数据

LONG 和 LONG RAW 数据类型以前用于非结构化数据，如二进制图像、文档或地理信息，目前主要用于向后兼容。这两种数据类型已由 LOB 数据类型代替。LOB 数据类型与 LONG 和 LONG RAW 不同，不能互换。LOB 不支持 LONG 应用程序编程接口 (API)，反之亦然。

**Oracle 内置数据类型（续）**

**LONG、LONG RAW 和大型对象 (LOB) 数据类型（续）：**

最好与旧的数据类型（LONG 和 LONG RAW）相比较来讨论 LOB 功能。在下文中，LONG 指 LONG 和 LONG RAW 数据类型，而 LOB 指所有 LOB 数据类型。

| LONG, LONG RAW | LOB         |
|----------------|-------------|
| 每个表一列          | 每个表多列       |
| 最高 2 千兆字节      | 最高 4 千兆字节   |
| 数据存储在五行中       | 数据存储在五行中或行外 |
| 不支持对象类型        | 支持对象类型      |
| 顺序访问数据块        | 随机访问数据块     |

除非其大小小于 VARCHAR2 数据类型的最大大小（4,000 字节），否则，LOB 在表中存储一个定位器，而将数据存储在另一位置；LONG 则将所有数据存为五行。此外，LOB 允许将数据存储在单独的段和表空间中，或者存储在主机文件中。

LOB 支持对象类型属性（NCLOB 除外）和复制；而 LONG 不支持。

LONG 主要存储为五行串的行片段，每一块中有一个行片段指向存储在另一块中的五行片段。因此，需要按顺序访问五行片段。相反，LOB 通过类似文件的接口支持以片段方式随机访问数据。

**ROWID 和 UROWID 数据类型：**

ROWID 是一种可以和表中其它列一起查询的数据类型。它具有五行特征：

- ROWID 是数据库中每行的唯一标识符。
- ROWID 并不显式地作为一个列值存储。
- 虽然 ROWID 并不直接给出一行的物理地址，但它可以用来定位行。
- ROWID 为访问表五行行提供了最快的方法。
- ROWID 存储在索引中来指定具有一组给定的键值的行。

在 Oracle8.1 版中，Oracle 服务器提供一种称为通用 ROWID 或 UROWID 的数据类型。它支持外表（非 Oracle 表）的 ROWID，并且可存储各种类型的 ROWID。例如：要存储按索引组织的表 (IOT) 中存储的五行 ROWID，必须使用 UROWID 数据类型。要使用 UROWID，参数 COMPATIBLE 的值必须设置为 Oracle8.1 或更高。

## Oracle 内置数据类型（续）

### 集合数据类型：

有两种集合数据类型可用来为表中的一个给定行存储重复的数据。在 Oracle8i 以前，定义和使用集合需要“对象”(Objects) 选项。下面简要论述这些类型。

变化数组 (VARRAY)：变化数组对于存储包含少量组成元素的列表（如客户的电话号码）非常有用。

VARRAY 具有以下特征：

- 数组即一组有序的数据组成元素。
- 一个给定数组的所有组成元素的数据类型相同。
- 每个组成元素都有索引，即与数组中组成元素的位置相对应的编号。
- 数组中组成元素的数目决定了数组的大小。
- Oracle 服务器允许数组的大小可以变化，这就是它们被称为 VARRAY（意为变化数组）的原因，但在声明数组类型时必须指定最大大小。

嵌套表：嵌套表提供一种将一个表定义为另一个表内一列的方法。嵌套表可用来存储可能包含大量记录的集合（比如一个订单中的若干条目）。

嵌套表一般具有以下特征：

- 嵌套表是一组无次序的记录或行。
- 嵌套表中的各行结构相同。
- 嵌套表中的行与父表分别存储，并且父表中的对应行有一个指针。
- 嵌套表的存储特点可由数据库管理员来定义。
- 嵌套表没有预先确定的最大大小。

### 关系数据类型 (REF)：

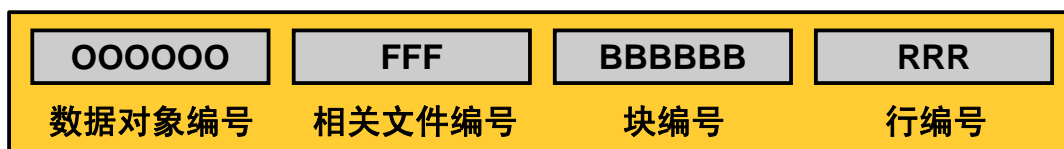
关系类型在数据库内用作指针。使用这些类型需要“对象”(Objects) 选项。这里给出一个例子：订购的每一项都可以指向或引用 PRODUCTS 表中的一行，而不必存储产品代码。

### Oracle 用户定义的数据类型：

Oracle 服务器允许用户定义抽象的数据类型并在应用程序内使用这些数据类型。

## ROWID 格式

- 扩展的 ROWID 格式



- 受限的 ROWID 格式



ORACLE

11-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### ROWID 格式

扩展的 ROWID 在磁盘上需要 10 个字节的存储空间，并使用 18 个字符来显示。它包含下列组成元素：

- 数据对象编号：每个数据对象（如表或索引）在创建时都分配有此编号，并且此编号在数据库中是唯一的
- 相关文件编号：此编号对于表空间中的每个文件是唯一的
- 块编号：表示包含此行的块在文件中的位置
- 行编号：标识块头中行目录位置的位置

在内部，数据对象编号需要 32 位、相关文件编号需要 10 位、块编号需要 22 位、行编号需要 16 位，加起来总共是 80 位或 10 个字节。

扩展的 ROWID 使用以 64 为基数的编码方案来显示，该方案将六个位置用于数据对象编号、三个位置用于相关文件编号、六个位置用于块编号、三个位置用于行编号。以 64 为基数的编码方案使用字符“A-Z”、“a-z”、“0-9”和“/”。共有 64 个字符，如下例所示：



## ROWID 格式（续）

```
SQL> SELECT department_id, rowid FROM hr.departments;
```

```
DEPARTMENT_ID ROWID
```

```

```

```
10 AAABQMAAFAAAAA6AAA
```

```
20 AAABQMAAFAAAAA6AAB
```

```
30 AAABQMAAFAAAAA6AAC
```

```
40 AAABQMAAFAAAAA6AAD
```

```
50 AAABQMAAFAAAAA6AAE
```

```
60 AAABQMAAFAAAAA6AAF
```

```
...
```

在本例中：

- AAABQM 是数据对象编号
- AAF 是相关文件编号
- AAAAA6 是块编号
- AAA 是 ID=10 的部分的行编号

### Oracle7 和更早版本中的受限 ROWID:

Oracle8 之前的 Oracle 数据库版本使用的是受限的 ROWID 格式。受限的 ROWID 在内部仅使用六个字节，不包含数据对象编号。此格式在 Oracle7 或更早的发行版中是可以接受的，因为文件编号在一个数据库内是唯一的。因此，较早的发行版不允许数据文件数超过 1,022 个。目前，对表空间有此限制。

即使 Oracle8 通过使用与表空间相关的文件编号摆脱了这种限制，但受限 ROWID 仍用在非分区表上的非分区索引之类的对象中，其中所有索引条目指的是同一段中的行。

### 使用 ROWID 定位行:

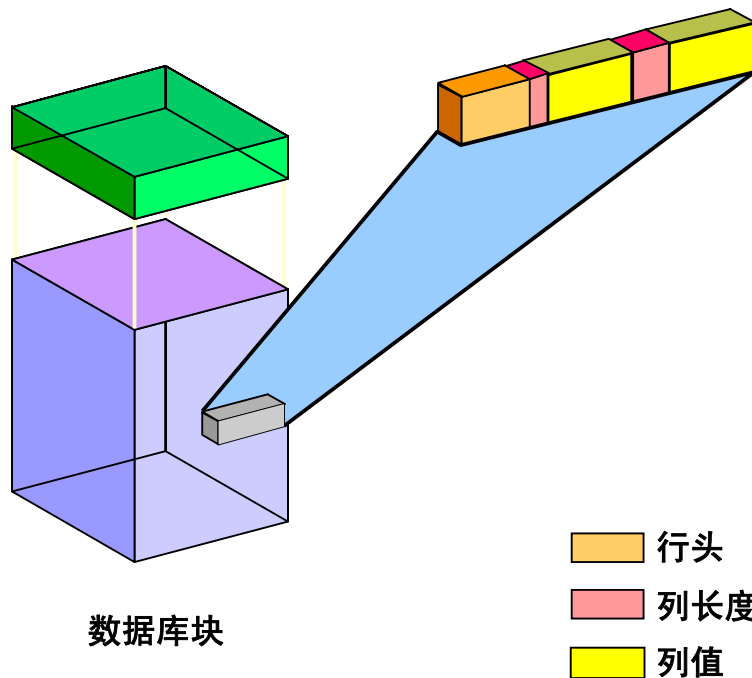
因为一个段只能驻留在一个表空间中，所以，Oracle 服务器可以使用数据对象编号来确定包含某一行的表空间。

表空间中的相关文件编号用来定位文件，块编号用来定位包含该行的块，行编号用来定位该行的行目录条目。

行目录条目可以用来定位行首。

这样，ROWID 就可以用来定位一个数据库中的任意行。

## 行的结构



ORACLE

11-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 行的结构

行数据作为长度可变的记录存储在数据库块中。通常，一个行的各列按其定义时的顺序存储，并且不存储尾随的 NULL 列。

注：对于非尾随的 NULL 列，列长度需要占用一个字节。表中的每行具有：

- 行头：用来存储行中的列数、链接信息和行锁定状态
- 行数据：对于每一列，Oracle 服务器存储列的长度和值（如果该列不超过 250 个字节，则需要一个字节来存储列长度；如果该列超过 250 个字节，则需要三个字节来存储列长度。列值在紧靠列长度字节后面存储。）

相邻的行之间不需要任何空格。块中的每一行在行目录中都有一个位置。目录位置指向行首。

# 创建表

```
CREATE TABLE hr.employees(
 employee_id NUMBER(6),
 first_name VARCHAR2(20),
 last_name VARCHAR2(25),
 email VARCHAR2(25),
 phone_number VARCHAR2(20),
 hire_date DATE DEFAULT SYSDATE,
 job_id VARCHAR2(10),
 salary NUMBER(8,2),
 commission_pct NUMBER(2,2),
 manager_id NUMBER(6),
 department_id NUMBER(4)
);
```

ORACLE

11-13

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建表

CREATE TABLE 命令用于创建关系表或对象表。

关系表：这是存储用户数据的基本结构。

对象表：是一种将对象类型用于列定义的表。对象表是一种显式定义的表，用来存储特定类型的对象例程。

注：本课不讨论对象表。

### 创建表的原理：

- 将各个表存放在单独的表空间中。
- 使用本地管理的表空间以避免产生存储碎片。

注：有关在使用 CREATE TABLE 命令时可以定义的各种子句和参数的详细信息，请参考 *Oracle9i SQL Reference* 文档。

要在您自己的方案中创建关系表，您必须具有 CREATE TABLE 系统权限。要在另一个用户的方案中创建表，您必须具有 CREATE ANY TABLE 系统权限。

注：有关授予权限的详细信息，请参考“管理权限”一课。

## 创建表（续）

下面的示例在数据字典管理的表空间中创建了一个 DEPARTMENTS 表。

```
SQL> CREATE TABLE hr.departments(
2 department_id NUMBER(4),
3 department_name VARCHAR2(30),
4 manager_id NUMBER(6),
5 location_id NUMBER(4))
6 STORAGE(INITIAL 200K NEXT 200K
7 PCTINCREASE 0 MINEXTENTS 1 MAXEXTENTS 5)
8 TABLESPACE data;
```

以上语法是 CREATE TABLE 子句的一部分。

### STORAGE 子句:

STORAGE 子句指定表的存储特性。给第一个区分配的存储空间为 200 KB。需要第二个区时，就会再创建一个 200 KB 的区（是由 NEXT 值定义的）。需要第三个区时，就会创建一个 200 KB 的区，这是因为已将 PCTINCREASE 设置为零。将可以使用的最大区数设置为 5，最小区数设置为 1。

- MINEXTENTS: 这是要分配的最小区数。
- MAXEXTENTS: 这是要分配的最大区数。如果将 MINEXTENTS 指定为一个大于 1 的值，而表空间包含多个数据文件，则这些区将分布在不同的数据文件中。
- PCTINCREASE: 这是 NEXT 区及以后的区有关区大小增长的百分比。

## 创建表（续）

还可以在 `physical_attributes_clause` 中指定表的块使用参数。

- **PCTFREE**: 指定表内每个数据块中空间的百分比。PCTFREE 的值必须介于 0 和 99 之间。如果值为零，表示可以通过插入新行来填充整个块。缺省值为 10。此值表示每个块中保留着 10% 的空间，用于更新现有的行以及插入新行，每个块最多可填充到 90%。
- **PCTUSED**: 指定为表内每个数据块维护的已用空间的最小百分比。如果一个块的已用空间低于 PCTUSED，则可在该块中插入行。PCTUSED 的值为介于 0 和 99 之间的整数，缺省值为 40。

结合 PCTFREE 和 PCTUSED 就可以确定将新行插入到现有数据块中，还是插入到新块中。这两个参数值的和必须小于或等于 100。使用这两个参数可以更有效地利用表内的空间。

**注:** Oracle9i “自动段空间管理” 功能可替代 PCTUSED、FREELISTS 和 FREELIST GROUPS。有关此功能的详细信息，请参考“存储结构和关系”一课。

- **INITTRANS**: 在分配给表的每个数据块内，指定分配的初始事务处理项数。此值的范围在 1 到 255 之间，缺省值为 1 个 INITTRANS: 确保最小数量的并发事务处理可以更新该块。通常，应该保留此值的缺省值。
- **MAXTRANS**: 指定可以更新分配给表的数据块的最大并发事务处理数量。此限制不适用于查询。值的范围在 1 到 255 之间，缺省值由数据块大小的函数确定。

### **TABLESPACE 子句:**

TABLESPACE 子句指定将要在其中创建表的表空间。示例中的表将驻留在数据表空间内。如果省略 TABLESPACE，则 Oracle 在包含该表的方案的所有者的缺省表空间中创建对象。

**注:** 有关表空间的详细信息，请参考“管理表空间”一课。

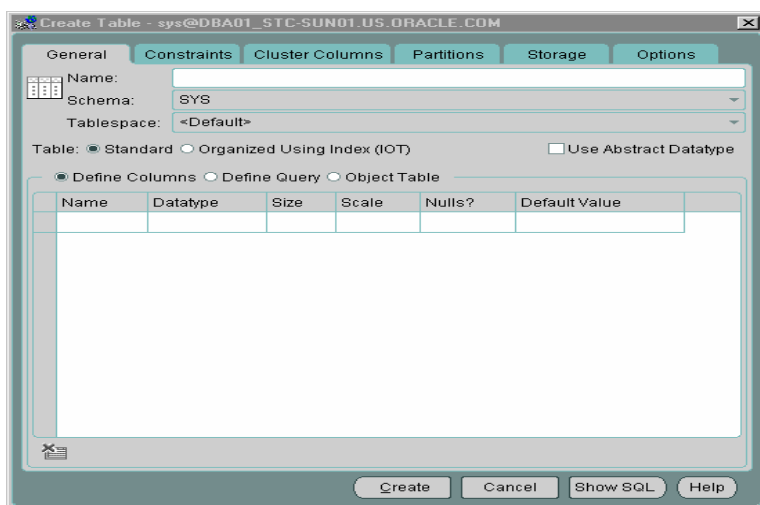
## 创建表

### 使用 Oracle Enterprise Manager 创建表

从“OEM 控制台”(OEM Console):

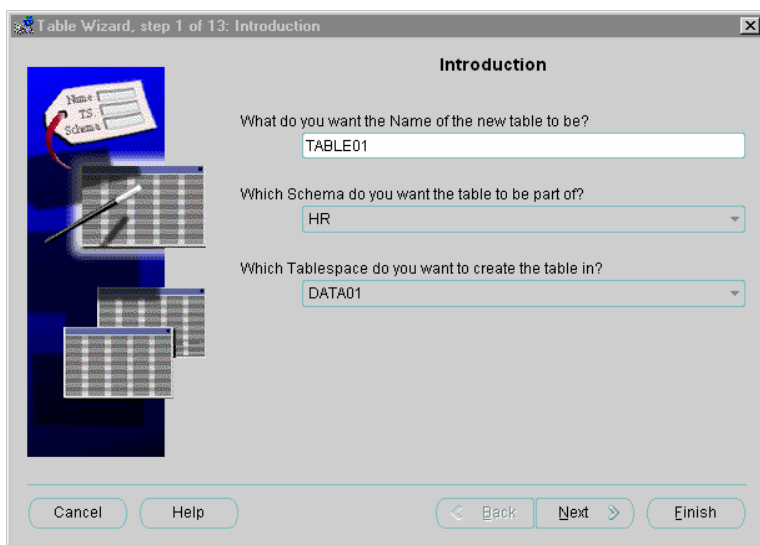
注: OEM 为创建表提供了很多选项。

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 单击鼠标右键, 从弹出的菜单中选择“创建”(Create)。
3. 输入表的信息, 如表名称、表空间、所有者、列、数据类型和大小。
4. 单击“创建”(Create)。



使用向导创建表:

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 选择“对象”(Object) > “使用向导创建”(Create Using Wizard)。
3. 输入表的信息, 如表名称、表空间、所有者、列、数据类型和大小。
4. 单击“完成”(Finish)。



## 创建表：原则

- 将各个表存放在单独的表空间中。
- 使用本地管理的表空间以避免产生存储碎片。
- 将表的标准区大小设置小一些，以减少表空间存储碎片。

ORACLE

11-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建表：原则

将各个表存放在单独的表空间内，而不是存放在有还原段、临时段和索引的表空间内。  
将表存放在本地管理的表空间内以避免产生存储碎片。

# 创建临时表

- 使用 GLOBAL TEMPORARY 子句创建

```
CREATE GLOBAL TEMPORARY TABLE
hr.employees_temp
AS SELECT * FROM hr.employees;
```

- 表仅为事务处理或会话期间保留数据
- 不能为数据获取 DML 锁
- 可以为临时表创建索引、视图和触发器

ORACLE

11-18

Copyright © Oracle Corporation, 2001. All rights reserved.

## 创建临时表

可以创建临时表，来保存仅在事务处理或会话期间存在的会话专用数据。

CREATE GLOBAL TEMPORARY TABLE 命令创建针对特定事务处理或针对特定会话的临时表。对于针对特定事务处理的临时表，数据仅在该事务处理期间存在；对于针对特定会话的临时表，数据仅在该会话期间存在。会话中的数据专用于该会话。每个会话只能查看和修改自己的数据。不能为临时表的数据获取 DML 锁。控制行持续时间的子句为：

- ON COMMIT DELETE ROWS：用来指定仅在该事务处理内可看到这些行
- ON COMMIT PRESERVE ROWS：用来指定在整个会话期间都能看到这些行

可以为临时表创建索引、视图和触发器，还可以使用导出和导入实用程序来导出和导入临时表的定义。但是，即使您使用 ROWS 选项，也不导出任何数据。所有会话都能看到临时表的定义。



## 设置 PCTFREE 和 PCTUSED

- 计算 PCTFREE

$$\frac{(\text{平均行大小} - \text{初始行大小}) * 100}{\text{平均行大小}}$$

- 计算 PCTUSED

$$100 - \text{PCTFREE} - \frac{\text{平均行大小} * 100}{\text{可用数据空间}}$$

ORACLE

11-19

Copyright © Oracle Corporation, 2001. All rights reserved.

## 设置 PCTFREE 和 PCTUSED

### 设置 PCTFREE

PCTFREE 值越高，可为数据库块内的更新提供的空间就越大。如果表存在下面两种情况，则应设置一个更高的值：

- 某些列最初为 NULL，后来更新为某个值
- 某些列由于更新，大小可能增加

PCTFREE 的值越高，块密度就越低，即每个块容纳的行数就越少。

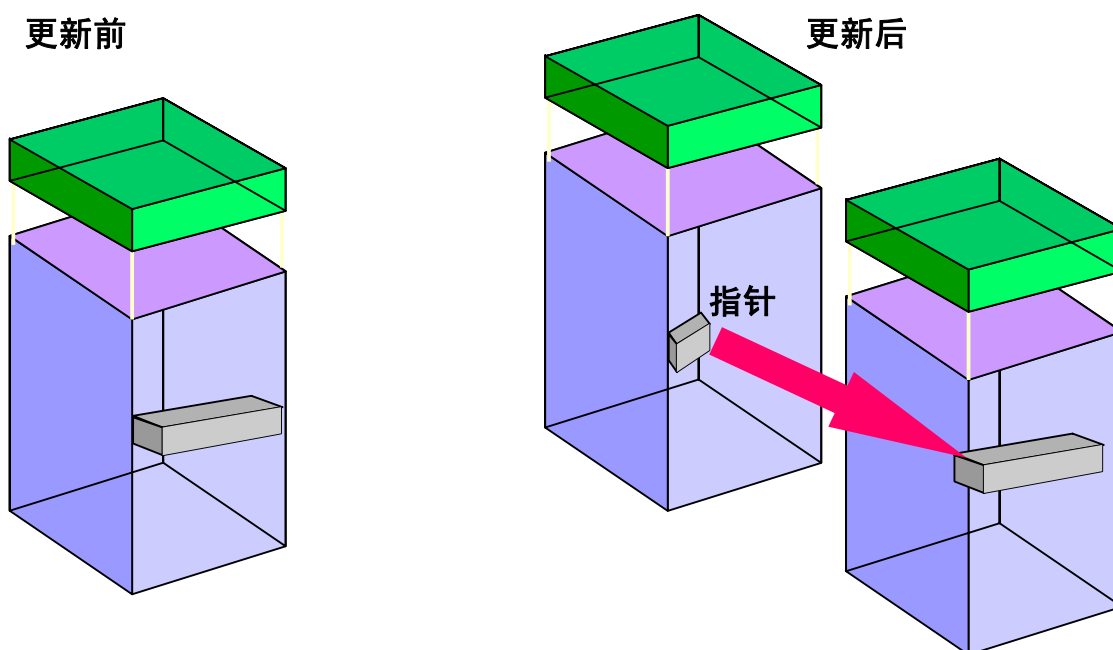
上面的公式确保块中有足够的空闲空间供行增长使用。

### 设置 PCTUSED

设置 PCTUSED 以确保只有在具备足够空间来容纳一个平均大小的行时才将块返回到空闲列表中。如果空闲列表中的某个块没有足够的空间来插入一行，Oracle 服务器将查找空闲列表中的下一个块。直到找到具备足够空间的块或者到达列表的末尾，这种线性扫描才会结束。使用给定的公式可以增加找到具有所需空闲空间的块的概率，从而缩短扫描空闲列表的时间。

注：可以使用 ANALYZE TABLE 命令估算平均行大小的值。

## 行移植和行链接



ORACLE

11-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### 行移植和行链接

#### 行移植

如果将 PCTFREE 设置为一个较低的值，则在一个块中可能没有足够的空间来容纳更新后增长的行。出现这种情况时，Oracle 服务器就会把整个行移到一个新块，并创建一个从原块指向新位置的指针。这个进程就称为“行移植”。在移植行时，与该行相关联的输入/输出 (I/O) 性能会降低，因为 Oracle 服务器必须扫描两个数据块才能检索该数据。

#### 行链接

如果一个行过大而任何一个块都容纳不下，就会发生行链接。如果行包含的列太长，就可能发生这种情况。在这种情况下，Oracle 服务器将该行拆分成更小的数据块，称为“行片段”。每个行片段存储在一个块中，并带有检索和组合整行所需要的指针。如果可能，可通过选择较大的块大小或将一个表拆分成包含更少列的多个表来最大限度地减少行链接。

**注：**在 *Oracle9i 数据库性能优化* 课程中将详细介绍行移植和行链接方面的信息。

## 更改存储和块使用参数

```
ALTER TABLE hr.employees
PCTFREE 30
PCTUSED 50
STORAGE(NEXT 500K
MINEXTENTS 2
MAXEXTENTS 100);
```

ORACLE

11-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### 更改存储和块使用参数

部分存储参数和所有块使用参数可通过 ALTER TABLE 命令加以修改。

语法：

```
ALTER TABLE [schema.]table
{[storage-clause]
[INITTRANS integer]
[MAXTRANS integer]}
```

更改存储参数的影响：

可修改的参数及其修改的影响如下所示：

- NEXT：当 Oracle 服务器为表分配另一个区时，就会使用新的值。此后的区大小将按 PCTINCREASE 增加。

## 更改存储和块使用参数（续）

### 更改存储参数的影响（续）：

- **PCTINCREASE**：对 **PCTINCREASE** 的更改将记录到数据字典中。Oracle 服务器分配下一个区时，将使用它来重新计算 **NEXT**。假定有这样的情况：一个表有两个区，其中 **NEXT**=10K，**PCTINCREASE**=0。如果将 **PCTINCREASE** 更改为 100，则要分配的第三个区为 10K，第四个区为 20K，第五个区为 40K，以此类推。
- **MINEXTENTS**：**MINEXTENTS** 的值可以更改为任何小于或等于表中的当前区数的值。它不会对表立即产生作用，但在截断表时将用到它。
- **MAXEXTENTS**：**MAXEXTENTS** 的值可以设置为任何大于或等于表的当前区数的值。也可以将该值设置为 **UNLIMITED**。

### 限制：

- 表的 **INITIAL** 值不能修改。
- 指定的 **NEXT** 的值将舍入为块大小的一个倍数，该值大于或等于指定的值。

## 使用 Oracle Enterprise Manager 更改存储参数

### 使用 Oracle Enterprise Manager 更改存储参数

从 “OEM 控制台” (OEM Console):

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开方案名称。
3. 选择该表。
4. 修改 “存储” (Storage) 页中的值。注意，不能用此方法修改最小区数和初始事务处理数。
5. 单击 “应用” (Apply)。

## 手动分配区

```
ALTER TABLE hr.employees
ALLOCATE EXTENT(SIZE 500K
DATAFILE '/DISK3/DATA01.DBF');
```

ORACLE

11-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 手动分配区

可能需要手动对区进行分配，以便：

- 控制一个表的区在文件之间的分配
- 在大量加载数据前避免表的动态扩展

#### 语法

使用下面的命令将一个区分配给一个表：

```
ALTER TABLE [schema.]table
ALLOCATE EXTENT [([SIZE integer [K|M]]
[DATAFILE 'filename'])]
```

如果忽略 SIZE，Oracle 服务器将使用 DBA\_TABLES 中的 NEXT\_EXTENT 大小来分配区。在 DATAFILE 子句中指定的文件必须属于该表所属的表空间。否则，该语句就会生成错误。如果未使用 DATAFILE 子句，则 Oracle 服务器将在包含该表的表空间中的一个文件中分配区。

**注：**手动分配区不会影响 DBA\_TABLES 中的 NEXT\_EXTENT 的值。执行此命令时，Oracle 服务器不会重新计算下一个区的大小。

## 重新组织非分区表

```
ALTER TABLE hr.employees
MOVE TABLESPACE data1;
```

- 重新组织非分区表时，将保留表的结构，但不保留表的内容。
- 用于将表移到另一个表空间中或者重新组织区。

ORACLE

11-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### 重新组织非分区表

不必运行导出或导入实用程序即可移动非分区表。此外，它允许更改存储参数。该特性在以下情况下很实用：

- 将表从一个表空间移到另一表空间
- 重新组织表以避免行移植

移动表后，必须重建索引以避免发生以下错误：

```
SQL> SELECT * FROM employees WHERE id=23;
```

```
select * from employees where id=23
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01502: 索引 'HR.EMPLOYEES_ID_PK' 或该索引的分区处于不可用状态
```

## 截断表

```
TRUNCATE TABLE hr.employees;
```

- 截断一个表将删除表中所有行，从而释放已使用的空间。
- 对应的索引将被截断。

ORACLE

11-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### 截断表

语法：

```
TRUNCATE TABLE [schema.] table
[{DROP | REUSE} STORAGE]
```

使用此命令的效果如下：

- 表中的所有行都被删除。
- 不会生成任何还原数据，且命令会隐式提交，因为 TRUNCATE TABLE 是一个 DDL 命令。
- 对应的索引也将截断。
- 不能截断某个外键正在引用的表。
- 使用此命令时不会触发删除触发器。



# 删除表

```
DROP TABLE hr.department
CASCADE CONSTRAINTS;
```

ORACLE

11-27

Copyright © Oracle Corporation, 2001. All rights reserved.

## 删除表

如果不再需要某个表，或者要对它进行重新组织，就可以将它删除。

### 语法：

使用下面的命令可删除表：

```
DROP TABLE [schema.] table
[CASCADE CONSTRAINTS]
```

删除一个表后，该表所使用的区将得以释放。如果这些区是相邻的，则可以在以后某个时间自动或手动将它们合并。

如果该表是外键关系中的父表，就必须使用 `CASCADE CONSTRAINTS` 选项。

**注：**`CASCADE CONSTRAINTS` 选项将在“维护数据完整性”一课中详细讨论。

## 删除表

### 使用 Oracle Enterprise Manager 删除表

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 展开包含要删除的表的方案。
3. 展开方案名称。
4. 选择该表。
5. 单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
6. 选择“是”(Yes) 确认删除。

# 删除列

从表中删除列：

```
ALTER TABLE hr.employees
DROP COLUMN comments
CASCADE CONSTRAINTS CHECKPOINT 1000;
```

- 从每行中删除列长度和数据，释放数据块中的空间。
- 删除大表中的一列可能需要相当长的时间。

ORACLE

11-29

Copyright © Oracle Corporation, 2001. All rights reserved.

## 删除列

可以使用 Oracle 服务器从表的行中删除列。删除列可清除未使用但可能占用大量空间的列，而不必导出或导入数据及重新创建索引和约束。

删除一列可能要用相当长的时间，因为该列的所有数据都将从表中删除。

在 Oracle8i 以前的发行版中，无法删除表中的列。

### 删除列时使用检查点：

删除列可能需要很长时间，并且需要大量的还原空间。从大型表中删除列时，可以指定检查点来尽量减少还原空间的使用。在幻灯片上的示例中，每 1,000 行出现一个检查点。在操作运行完成前，该表一直被标记为 INVALID。如果操作过程中例程失败，则该表在启动后仍将处于 INVALID 状态，因此该操作必须完成。

使用下面的语句可恢复中断的删除操作：

```
SQL> ALTER TABLE hr.employees DROP COLUMNS CONTINUE;
```

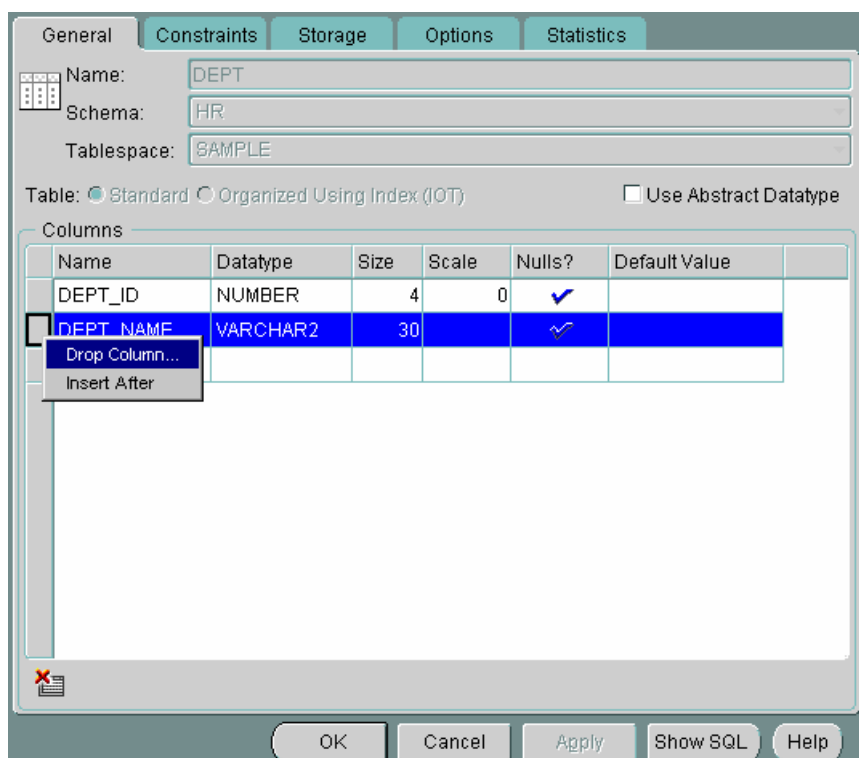
如果表处于 VALID 状态，则使用此语句将生成错误。

## 删除列

使用 **Oracle Enterprise Manager** 删除列：

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开包含要删除的表的方案。
3. 展开方案名称。
4. 选择该表。
5. 单击鼠标右键，从弹出的菜单中选择 “编辑/查看详细资料” (Edit/View Details)。
6. 选择要删除的列。
7. 单击鼠标右键，从弹出的菜单中选择 “删除列” (Drop Column)。



## 使用 UNUSED 选项

- 将列标记为未使用:

```
ALTER TABLE hr.employees
SET UNUSED COLUMN comments CASCADE
CONSTRAINTS;
```

- 删除未使用的列:

```
ALTER TABLE hr.employees
DROP UNUSED COLUMNS CHECKPOINT 1000;
```

- 继续执行删除列操作:

```
ALTER TABLE hr.employees
DROP COLUMNS CONTINUE CHECKPOINT 1000;
```

ORACLE

### 使用 UNUSED 选项

除将列从表中删除以外，还可以先将列标记为“未使用”，以后再删除。因为没有删除数据，所以此操作不回收磁盘空间，因而具有速度比较快的优点。被标为“未使用”的列可在以后系统活动较少时从表中删除。

未使用的列就像不属于表一样。查询时看不到未使用列中的数据。此外，在执行 DESCRIBE 命令时，也不会显示这些列的名称和数据类型。用户可以添加与未使用的列同名的新列。

如果想删除同一表中的两列，则可先将列设置为“未使用”然后再删除。在删除两列时，表中的所有行都会更新两次；但如果将这些列设置为“未使用”然后再删除，则所有的行仅更新一次。

## 使用 UNUSED 选项（续）

### 确定包含未使用列的表

要确定包含未使用列的表，可以查询视图 DBA\_UNUSED\_COL\_TABS。该查询可获取包含未使用列的表的名称及表中标记为未使用列的数目。下面的查询显示 HR 拥有的表 EMPLOYEES 含有一个未使用的列：

```
SQL > SELECT * FROM dba_unused_col_tabs;
```

```
OWNER TABLE_NAME COUNT

HR EMPLOYEES 1
```

要确定已完成一部分 DROP COLUMN 操作的表，可查询 DBA\_PARTIAL\_DROP\_TABS 视图。

```
SQL > SELECT * FROM dba_partial_drop_tabs;
```

```
OWNER TABLE_NAME COUNT

no rows selected
```

### 删除列的限制

不能执行下列操作：

- 从对象类型表中删除列
- 从嵌套表中删除列
- 删除一个表中的所有列
- 删除分区键列
- 从 SYS 拥有的表中删除列
- 从按索引组织的表中删除主键列
- 如果有未使用但未删除的 LONG 或 LONG RAW 列，将无法向表中添加 LONG 或 LONG RAW 列。（即使表的说明显示没有 LONG 或 LONG RAW 列也是如此。）

## 获取表信息

可以通过查询以下视图来获取有关表的信息：

- **DBA\_TABLES**
- **DBA\_OBJECTS**

ORACLE

11-33

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取表信息

有关表的信息可从数据字典中获取。要获取 HR 拥有的所有表的数据对象编号和表头位置，请使用下面的查询：

```
SQL > SELECT table_name FROM dba_tables WHERE owner = 'HR';
```

```
TABLE_NAME
```

```

```

```
COUNTRIES
```

```
DEPARTMENTS
```

```
DEPARTMENTS_HIST
```

```
EMPLOYEES
```

```
EMPLOYEES_HIST
```

```
JOBS
```

```
JOB_HISTORY
```

```
LOCATIONS
```

```
REGIONS
```

### 获取表信息（续）

```
SQL> SELECT object_name, created
 2 FROM DBA_OBJECTS
 3 WHERE object_name like 'EMPLOYEES'
 4 AND owner = 'HR';
```

```
OBJECT_NAME CREATED

EMPLOYEES 16-APR-01
```



## 小结

在这一课中，您应该能够掌握：

- 区分扩展的和受限的 ROWID
- 概括行的结构
- 创建规则表和临时表
- 管理表内的存储结构
- 重新组织、截断和删除表
- 删除表内的列

ORACLE

## 练习 11 概览

此练习涉及以下主题：

- 创建表
- 查看表中的列、将列标记为未使用，以及删除表中的列
- 手动分配区
- 截断表
- 获取表信息

ORACLE

11-36

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 11 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 11：管理表

- 1 以用户 SYSTEM 的身份，为目前正在使用的订单输入系统创建下列表。表和列显示如下：

| 表         | 列            | 数据类型和大小      |
|-----------|--------------|--------------|
| CUSTOMERS | CUST_CODE    | VARCHAR2(3)  |
|           | NAME         | VARCHAR2(50) |
|           | REGION       | VARCHAR2(5)  |
| ORDERS    | ORD_ID       | NUMBER(3)    |
|           | ORD_DATE     | DATE         |
|           | CUST_CODE    | VARCHAR2(3)  |
|           | DATE_OF_DELY | DATE         |

**注：**在使用 OEM 时，一定要将 DATE\_OF\_DELY 设置为 NULL。

请注意，在表 ORDERS 中插入的行并不包含 DATE\_OF\_DELY 的值，该信息将在履行订单后更新。请使用表空间 USERS。可以使用缺省的存储设置。

- 2 运行脚本 lab11\_02.sql，将行插入到表中。

- 3 查找哪些文件和块包含订单表的行。

**提示：**查询数据字典视图 DBA\_EXTENTS。

- 4 检查 ORDERS 表使用的区数。

- 5 使用缺省大小为 ORDERS 表手动分配一个区，并确认该区已按照指定添加。

- 6 再创建一个表 ORDERS2，作为 USERS 表空间中 ORDERS 表的副本，并且 MINEXTENTS 等于 10。检查是否已使用指定的区数创建该表。

- 7 截断 ORDERS 表而不释放空间，检查区数以验证区未被回收。

- 8 截断 ORDERS2 表并释放空间。现在该表有多少个区？

- 9 运行脚本 lab11\_09.sql，向 ORDERS2 表中插入一些行。

- 10 查看 ORDERS2 表的列。然后将 DATE\_OF\_DELY 列标记为 UNUSED。再次查看 ORDERS2 表的列。结果如何？

- 11 删除未使用的列 DATE\_OF\_DELY。

- 12 删除 ORDERS2 表。



# 12

## 管理索引

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 列出各种类型的索引及其用途
- 创建各种类型的索引
- 重新组织索引
- 维护索引
- 监视索引的使用

ORACLE

# 索引分类

- **逻辑**
  - 单列或串接
  - 唯一或非唯一
  - 基于函数
  - 域
- **物理**
  - 分区或非分区
  - **B 树**
    - 正常或反向键
  - 位图

ORACLE

12-3

Copyright © Oracle Corporation, 2001. All rights reserved.

## 索引分类

索引是一种允许对表中的行进行直接访问的树型结构。可以根据索引的逻辑设计或物理实现对索引进行分类。逻辑分类从应用程序的角度对索引进行分组，而物理分类则是基于索引的存储方式。

### 单列索引和串接索引：

*单列索引* 在索引键中仅有一列，例如，雇员表中雇员编号列上的索引。

*串接索引* 是在表内多个列上创建的，也称为“组合索引”。串接索引中的列不必与表中的列顺序一致，也不必相互邻接，例如，雇员表中部门和职务列上的索引。

组合键索引最多包含 32 列。不过，所有列大小的总和不能超过数据块中可用数据空间的大约二分之一（减去某些开销）。

## 索引分类（续）

### 唯一索引和非唯一索引：

索引可以是唯一的或非唯一的。唯一索引保证表中没有两行在键列（或多个键列）中具有重复的值。非唯一索引对列值没有此限制。

### 基于函数的索引：

如果在表中要建立索引的一列或多列上使用了函数或表达式，则创建的是基于函数的索引。基于函数的索引预先计算函数或表达式的值，并将结果存储在索引中。可以将基于函数的索引创建为 B 树或位图索引。

### 域索引：

域索引是特定于应用程序 (Text, Spatial) 的索引，由索引类型提供的例程序创建、管理和访问。它之所以称为“域索引”，是因为它对特定于应用程序的域中的数据建立索引。

仅支持单列域索引。可以在具有标量、对象或 LOB 数据类型的列上建立单列域索引。

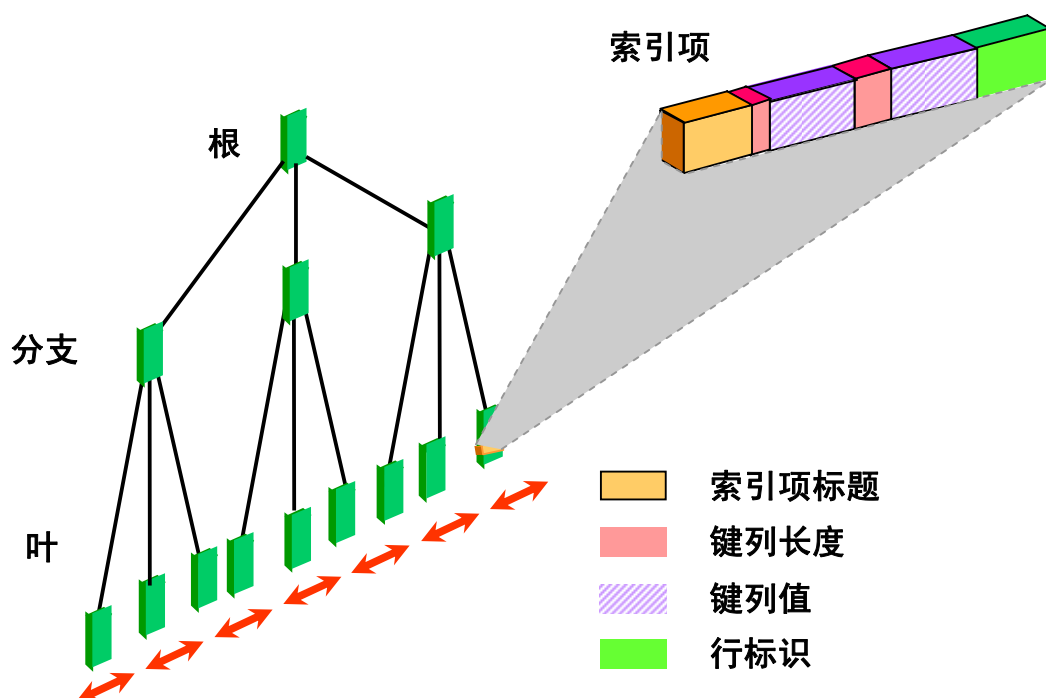
### 分区索引和非分区索引：

分区索引用于大型表，将与某个索引对应的索引项存储在多个段中。分区使索引可以在多个表空间中展开，从而降低索引查找争用并提高可管理性。分区索引通常用于分区表以提高可伸缩性和可管理性。可为每个表分区创建一个索引分区。

本课讨论非分区 B 树索引和位图索引的创建和维护。



## B 树索引



12-5

Copyright © Oracle Corporation, 2001. All rights reserved.

ORACLE

### B 树索引

虽然所有索引都使用 B 树结构，但术语“B 树索引”通常与存储每个关键字的行标识列表的索引关联。

#### B 树索引的结构：

索引的顶部为根，其中包含指向索引中下一级的项。下一级为分支块，分支块又指向索引中下一级的块。最低一级为叶节点，其中包含指向表行的索引项。叶块为双重链接，有助于按键值的升序和降序扫描索引。

#### 索引叶项的格式：

索引项由以下部分组成：

- 项标题，存储列数和锁定信息
- 键列的“长度 - 值”对，用于定义键列的大小及该列的值（该值对的数目即索引中的最大列数。）
- 行的行标识，包含键值

## B 树索引（续）

### 索引叶项的特征：

在非分区表上的 B 树索引中：

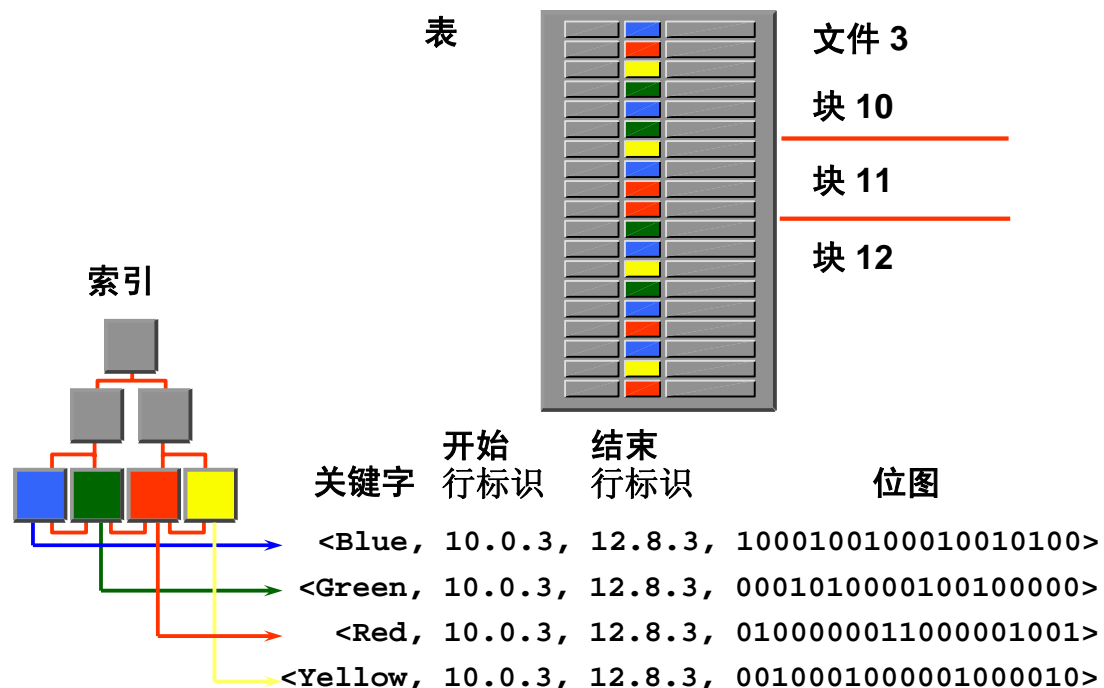
- 如果多行具有相同的键值，除非对索引进行了压缩，否则键值重复。
- 对于所有键列都为 NULL 的行，没有对应的索引项。因此，指定 NULL 的 WHERE 子句始终进行全表扫描。
- 因为所有行都属于同一段，所以使用受限行标识指向表中的行。

### DML 操作对索引的影响：

在表上执行 DML 操作时，Oracle 服务器将维护所有的索引。下面解释 DML 命令对索引的影响：

- 插入操作导致在适当的块中插入索引项。
- 删除行只导致逻辑删除索引项。删除的行所用的空间仍不能用于新项，直到删除块中的所有项。
- 更新键列将导致逻辑删除和向索引插入项。除了创建时以外，PCTFREE 设置在其它任何时候都对索引没有影响。即使索引块空间少于 PCTFREE 指定的空间，仍可以向索引块添加新项。

# 位图索引



ORACLE

12-7

Copyright © Oracle Corporation, 2001. All rights reserved.

## 位图索引

在下列情况中，位图索引比 B 树索引更有利：

- 当表包含数百万行且键列的基数很低（即，该列中重复的值很多）时。例如，对于包含护照记录的表的性别列和婚姻状况列而言，位图索引比 B 树索引更适合
- 当查询经常使用涉及 OR 运算符的多个 WHERE 条件组合时
- 当键列上存在只读或很少的更新操作时

### 位图索引的结构：

也可以将位图索引组织为 B 树，但叶节点存储每个键值的位图而非行标识列表。位图中的每一位对应一个可能的行标识，如果设置了位，则意味着具有相应行标识的行包含键值。

如图所示，位图索引的叶节点包含下列几项：

- 项标题，包含列数和锁定信息
- 键值由每个键列的“长度 - 值”双值组成（本例中，关键字仅包含一列，第一项的键值为 Blue）。

## 位图索引（续）

### 位图索引的结构（续）：

- 开始行标识，本例中的开始行标识包含文件号 3、块号 10 和行号 0
- 结束行标识，本例中的结束行标识包含块号 12 和行号 8
- 位图段，由位串组成（对应的行包含键值时设置位；不包含键值时不设置位。Oracle 服务器使用专利压缩技术存储位图段。）

开始行标识是位图的位图段指向的第一行的行标识，即，位图的第一位对应此行标识，位图的第二位对应块中的下一行，而结束行标识是位图段所包含的表中最后一行的指针。位图索引使用受限行标识。

### 使用位图索引：

B 树用于定位包含给定键值的位图段的叶节点。开始行标识和位图段用于定位包含键值的行。

更改表中的键列时，必须修改位图。这将导致锁定相关的位图段。由于锁是在整个位图段上获取的，所以直到第一个事务处理结束后，才能由其它事务处理更新位图包含的行。

## 比较 B 树索引和 位图索引

| B 树               | 位图                |
|-------------------|-------------------|
| 适用于高基数列           | 适用于低基数列           |
| 更新关键字的成本相对较低      | 更新键列的成本非常高        |
| 使用 OR 谓词进行查询时效率较低 | 使用 OR 谓词进行查询时效率较高 |
| 对 OLTP 很有用        | 对数据仓库很有用          |

ORACLE

12-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 比较 B 树索引和位图索引

用于低基数列时，位图索引比 B 树索引更紧凑。

由于位图使用位图段级锁定，所以位图索引中的键列的更新成本较高；而在 B 树索引中，锁位于与表中单个行相对应的项上。

位图索引可用于执行位图布尔等操作。Oracle 服务器可以使用两个位图段执行逐位布尔操作并得到一个结果位图。这将允许在使用布尔谓词的查询中更有效地使用位图。

总之，B 树索引更适合索引动态表的 OLTP 环境，而位图索引更适合在大型静态表上使用复杂查询的数据仓库环境。

## 创建正常的 B 树索引

```
CREATE INDEX hr.employees_last_name_idx
ON hr.employees(last_name)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE indx;
```

ORACLE

12-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建正常的 B 树索引

索引可在表所有者的帐户下或其它帐户下创建，但索引通常在表所在的同一帐户下创建。上面的语句在 EMPLOYEES 表上创建一个索引（使用 LAST\_NAME 列）。

## 创建正常的 B 树索引（续）

### 语法选项

**UNIQUE**: 用于指定唯一的索引（缺省为 **Nonunique**。）

**Schema**: 索引/表的所有者

**Index**: 索引名

**Table**: 表名

**Column**: 列名

**ASC/DESC**: 指示是按升序还是按降序创建索引

**TABLESPACE**: 指定要在其中创建索引的表空间

**PCTFREE**: 创建索引时为容纳新的索引项而在每块中保留的空间大小（以总空间量减去块头后的百分比表示）

**INITTRANS**: 指定每块中预先分配的事务处理项的数目（缺省值和最小值为 2。）

**MAXTRANS**: 限制可以为每个块分配的事务处理项数（缺省值为 255。）

**STORAGE** 子句: 标识确定如何为索引分配区的存储子句

**LOGGING**: 指定在重做日志文件中记录索引创建操作和在索引上执行的后续操作（这是缺省值。）

**NOLOGGING**: 指定在重做日志文件中不记录创建操作和某些类型的数据加载操作

**NOSORT**: 指定将行按升序存储在数据库中，这样，Oracle 服务器在创建索引时不必对行进行排序

### 注

- 如果已为表空间定义了 **MINIMUM EXTENT**，则索引的区大小将向上舍入为下一个更高的 **MINIMUM EXTENT** 值的倍数。
- 如果省略了 **[NO]LOGGING** 子句，索引的事件记录属性将缺省为表所驻留的表空间的事件记录属性。
- 不能为索引指定 **PCTUSED**。由于索引项必须按正确的顺序存储，所以用户无法控制何时在某一索引块中插入。
- 如果在数据未按关键字排序的情况下使用 **NOSORT** 关键字，语句将终止并显示错误。如果表上已经有多个 **DML** 操作，则该选项很可能无效。
- 如果可能，Oracle 服务器使用现有索引创建新的索引。当新索引的关键字与现有索引键的主要部分对应时，就会发生这种情况。

## 创建正常的 B 树索引（续）

### 使用 Oracle Enterprise Manager 创建索引

从 “OEM 控制台” (OEM Console):

1. 导航到 “数据库” (Databases) > “方案” (Schema) 文件夹 > “索引” (Index)。
2. 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
3. 在 “常规” (General)、 “存储” (Storage) 和 “选项” (Options) 页中输入相应的信息。
4. 单击 “创建” (Create)。

Create Index - sys@DBA01\_STC-SUN01.US.ORACLE.COM

General Partitions Storage Options

Name: EMPLOYEES\_LAST\_NAME\_IDX

Schema: HR

Tablespace: INDX

Index On: ☒ Table ☐ Cluster

Schema: HR

Table: EMPLOYEES

| Table Columns  | Datatype | Order |
|----------------|----------|-------|
| COMMISSION_PCT | NUMBER   |       |
| DEPARTMENT_ID  | NUMBER   |       |
| EMAIL          | VARCHAR2 |       |
| EMPLOYEE_ID    | NUMBER   |       |
| FIRST_NAME     | VARCHAR2 |       |
| HIRE_DATE      | DATE     |       |
| JOB_ID         | VARCHAR2 |       |
| LAST_NAME      | VARCHAR2 |       |

Options

☐ Unique ☐ Bitmap ☐ Unsorted ☐ Reverse

Create Cancel Show SQL Help



## 创建索引：原则

- 平衡查询和 DML
- 存放在单独的表空间中
- 使用统一的区大小：块数是 5 的倍数或对表空间使用 MINIMUM EXTENT 大小
- 对于大型索引，请考虑使用 NOLOGGING
- 通常，INITRANS 在索引中比在对应的表中高。

ORACLE

12-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建索引：原则

创建索引时应考虑：

- 索引能够提高查询性能并降低 DML 操作速度。始终使易失表所需的索引数保持最少。
- 将索引放在一个单独的表空间中，不要放在有还原段、临时段和表的表空间中。
- 对大型索引而言，避免生成重做日志可显著提高性能。请考虑使用 NOLOGGING 子句创建大型索引。
- 由于索引项比索引行小，所以索引块趋向于在每块中包含更多的项。因此，INITRANS 在索引中通常比在对应的表中高。

#### 索引和 PCTFREE:

索引的 PCTFREE 参数与表的 PCTFREE 参数工作方式不同。前者仅在创建索引时用来为需要插入到同一索引块的索引项保留空间。而不更新索引项。更新键列时，这将涉及逻辑删除索引项和插入。

## 创建索引：原则（续）

### 索引和 PCTFREE（续）

在单调递增（如系统生成的发票号）列的索引上使用较低的 PCTFREE 值。在这些情况下，新的索引项总是追加到现有项上，没有必要在两个现有索引项间插入一个新项。

如果插入行的索引列值可采用任何值（即新值在当前的值范围内），则应该提供较高的 PCTFREE。发票表的客户代码列上的索引就是一个要求高 PCTFREE 值的索引。在这种情况下，将 PCTFREE 值指定为由下列等式所表示的值是非常有用的：

$$\frac{\text{最大行数} - \text{初始行数}}{\text{最大行数}} \times 100$$

最大值可用于特定的时间周期，如一年。

## 创建位图索引

```
CREATE BITMAP INDEX orders_region_id_idx
ON orders(region_id)
PCTFREE 30
STORAGE(INITIAL 200K NEXT 200K
PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE indx;
```

ORACLE

12-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建位图索引

语法:

使用下列命令创建位图索引:

```
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ASC | DESC] [, column [ASC | DESC]] ...)
[TABLESPACE tablespace]
 [PCTFREE integer]
 [INITTRANS integer]
 [MAXTRANS integer]
 [storage-clause]
 [LOGGING | NOLOGGING]
 [NOSORT]
```

注意, 位图索引不能是唯一的。

## 创建位图索引（续）

### **CREATE\_BITMAP\_AREA\_SIZE 参数：**

初始化参数 `CREATE_BITMAP_AREA_SIZE` 决定了内存中用于存储位图段的空间量。缺省值为 8 MB。使用较大的值，可提高索引创建的速度。如果基数很小，可将该值设置为一个较小值。例如，如果基数仅为 2，则该值可以为千字节数量级而非兆字节数量级。一般来讲，基数越大，则获取最佳性能所需的内存越多。

## 创建位图索引

### 使用 Oracle Enterprise Manager 创建位图索引

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “索引”(Index)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General)、“存储”(Storage)和“选项”(Options)页中输入相应的信息。
4. 从“常规”(General)页中选择“位图”(Bitmap)。
5. 单击“创建”(Create)。

Create Index - sys@DBA01\_STC-SUN01.US.ORACLE.COM

General Partitions Storage Options

Name: ORDERS\_REGION\_ID\_IDX

Schema: OE

Tablespace: INDX

Index On: ☒ Table ☐ Cluster

Schema: OE

Table: ORDERS

| Table Columns | Datatype                | Order |
|---------------|-------------------------|-------|
| CUSTOMER_ID   | NUMBER                  |       |
| ORDER_DATE    | TIMESTAMP(6) WITH LO... |       |
| ORDER_ID      | NUMBER                  |       |
| ORDER_MODE    | VARCHAR2                |       |
| ORDER_STATUS  | NUMBER                  |       |
| ORDER_TOTAL   | NUMBER                  |       |
| PROMOTION_ID  | NUMBER                  |       |
| SALES_REP_ID  | NUMBER                  |       |

Options

☐ Unique ☒ Bitmap ☐ Unsorted ☐ Reverse

Create Cancel Show SQL Help

## 更改索引的存储参数

```
ALTER INDEX employees_last_name_idx
STORAGE(NEXT 400K
MAXEXTENTS 100);
```

ORACLE

12-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### 更改索引的存储参数

某些存储参数和块使用参数可通过 ALTER INDEX 命令进行修改。

语法：

```
ALTER INDEX [schema.]index
[storage-clause]
[INITRANS integer]
[MAXTRANS integer]
```

更改索引存储参数与更改表存储参数的含义一样。通常，进行此类更改是为了增加索引的 MAXEXTENTS。

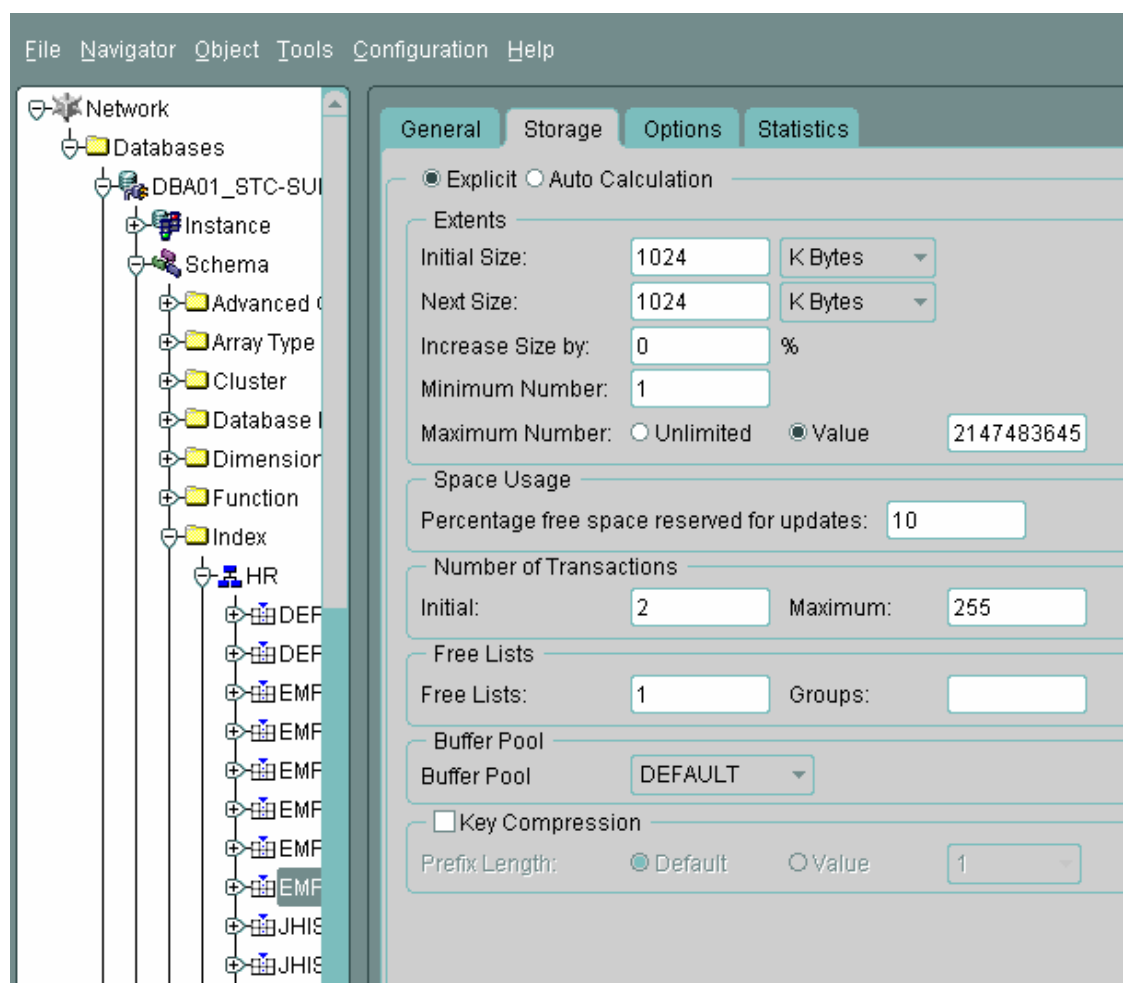
可更改块使用参数以保证在索引块上实现更高级别的并发性。

## 更改索引的存储参数

### 使用 Oracle Enterprise Manager 更改索引的存储参数

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “索引”(Index)。
2. 展开该索引所属的方案名称。
3. 选择索引。
4. 修改“存储”(Storage) 页中的值。
5. 单击“应用”(Apply)。



## 分配和回收索引空间

```
ALTER INDEX orders_region_id_idx
ALLOCATE EXTENT (SIZE 200K
DATAFILE '/DISK6/indx01.dbf');
```

```
ALTER INDEX orders_id_idx
DEALLOCATE UNUSED;
```

ORACLE

12-20

Copyright © Oracle Corporation, 2001. All rights reserved.

### 分配和回收索引空间

#### 手动分配索引空间：

在表上进行频繁的插入操作前，可能需要向索引添加区。添加区可防止索引动态扩展并导致性能降低。

#### 从索引中手动回收空间：

使用 ALTER INDEX 命令的 DEALLOCATE 子句释放索引中超过高水位标记的未用空间。

#### 语法：

使用下列命令分配或回收索引空间：

```
ALTER INDEX [schema.]index
{ALLOCATE EXTENT ([SIZE integer [K|M]]
[DATAFILE 'filename'])
| DEALLOCATE UNUSED [KEEP integer [K|M]] }
```

手动分配和手动回收索引空间所遵循的规则与在表上使用这些命令时相同。

**注：**当建立索引的表被截断时，回收索引空间。截断表将导致截断关联的索引。



# 重建索引

使用 ALTER INDEX 命令执行以下操作：

- 将索引移到另一个表空间中
- 通过移除已删除的项，提高空间的使用率

```
ALTER INDEX orders_region_id_idx REBUILD
TABLESPACE indx02;
```

ORACLE

12-21

Copyright © Oracle Corporation, 2001. All rights reserved.

## 重建索引

索引重建具有以下特点：

- 将现有索引作为数据源建立新索引。
- 使用现有索引建立索引时无需排序，从而使性能更佳。
- 在建立新索引后，删除旧索引。在重建期间，各自的表空间内需要有足够的空间以容纳新旧索引。
- 结果索引不包括任何已删除的项。因此，该索引可以更有效地使用空间。
- 在建立新索引的过程中，查询可继续使用现有索引。

可能的重建情况：

在下列情况下应重建索引：

- 需要将现有索引移到另外的表空间中。如果索引和表在同一表空间中或者需要跨磁盘重新分布对象时，可能需要执行此操作。

## 重建索引（续）

### 可能的重建情况（续）：

- 索引中包含很多已删除的项。这是滑动索引（如订单表中的订单号上的索引）存在的典型问题，完成的订单已被删除，并将具有更高订单号的新订单添加到表中。如果有几个旧订单未完成，则可能有若干个索引叶块包含除几个已删除项之外的全部项。
- 需要将现有正常索引转换成反向键索引。在从 Oracle 服务器的早期发行版移植应用程序时，可能会出现这种情况。
- 已通过 ALTER TABLE...MOVE TABLESPACE 命令将索引表移至其它表空间。

### 语法：

使用下列命令重建索引：

```
ALTER INDEX [schema.] index REBUILD
[TABLESPACE tablespace]
[PCTFREE integer]
[INITRANS integer]
[MAXTRANS integer]
[storage-clause]
[LOGGING | NOLOGGING]
[REVERSE | NOREVERSE]
```

ALTER INDEX ...REBUILD 命令不能用于将位图索引更改为 B 树索引，反之亦然。只能为 B 树索引指定 REVERSE 或 NOREVERSE 关键字。

# 联机重建索引

- 可以使用最小限度的表锁定来重建索引。

```
ALTER INDEX orders_id_idx REBUILD ONLINE;
```

- 某些限制仍然适用。

ORACLE

12-23

Copyright © Oracle Corporation, 2001. All rights reserved.

## 联机重建索引

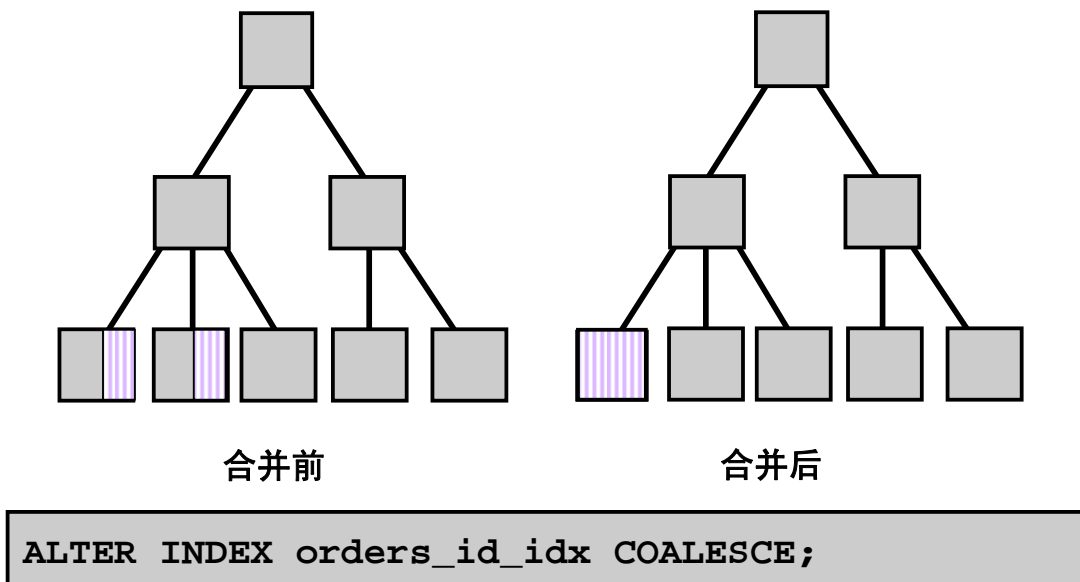
建立或重建索引是一项费时的任务，尤其当表非常大时更是如此。在 Oracle8i 之前，建立或重建索引都需要锁定表，并要防止并发的 DML 操作。Oracle9i 允许在基表上进行并发操作的同时建立或重建索引，但不建议在此过程中执行大量的 DML 操作。

注：仍存在 DML 锁，这意味着在联机索引建立期间不能执行其它 DDL 操作。

限制：

- 不能在临时表中重建索引
- 不能重建整个分区索引。必须分别重建每个分区或子分区。
- 也不能回收未用空间。
- 不能整个更改索引的 PCTFREE 参数值。

## 合并索引



ORACLE

12-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 合并索引

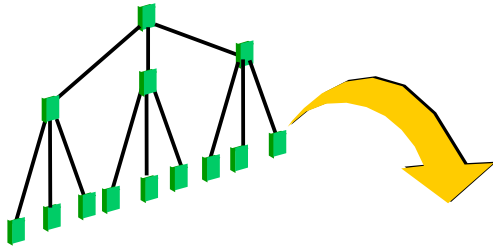
遇到索引碎片时，可以重建或合并索引。执行上述任务前，应考虑每种选择的成本和好处，然后选择最适合自己的方案。索引合并是联机完成的块重建过程。

如果有可以释放以供重用的 B 树索引叶块，则可使用下列 SQL 语句合并这些叶块：

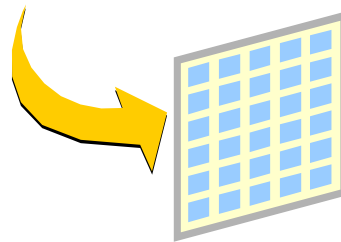
```
SQL> ALTER INDEX hr.employees_idx COALESCE;
```

上图显示了 ALTER INDEX ... COALESCE 语句对索引 hr.employees\_idx 的影响。在执行 COALESCE 操作之前，前两个叶块为 50% 填满。这意味着，索引包含碎片，可以对索引进行合并以完全填充第一个块，从而减少了碎片。

## 检查索引及其有效性



```
ANALYZE INDEX orders_region_id_idx
VALIDATE STRUCTURE;
```



**INDEX\_STATS**

ORACLE

12-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### 检查索引及其有效性

分析索引以执行以下操作：

- 检查所有的索引块是否存在损坏。注意，此命令并不验证索引项是否与表中的数据对应。
- 使用索引的有关信息填充 INDEX\_STATS 视图。

语法：

```
ANALYZE INDEX [schema.]index VALIDATE STRUCTURE
```

运行此命令后，查询 INDEX\_STATS 以获取索引的有关信息，如下例所示：

### 检查索引及其有效性（续）

```
SQL> SELECT blocks, pct_used, distinct_keys
2 lf_rows, del_lf_rows
3 FROM index_stats;
```

| BLOCKS | PCT_USED | LF_ROWS | DEL_LF_ROWS |
|--------|----------|---------|-------------|
| -----  | -----    | -----   | -----       |
| 25     | 11       | 14      | 0           |

1 row selected.

如果索引中已删除行的比例很高，请重新组织该索引。例如：当 DEL\_LF\_ROWS 占 LF\_ROWS 的比率超过 30% 时。

# 删除索引

- 在批量加载前，请删除并重新创建索引。
- 删除不常用的索引，并在需要时创建这些索引。
- 删除并重新创建无效的索引。

```
DROP INDEX hr.departments_name_idx;
```

ORACLE

12-27

Copyright © Oracle Corporation, 2001. All rights reserved.

## 删除索引

下列情况应删除索引：

- 应用程序不再需要索引时，可将索引删除。
- 执行批量加载前，索引可能已删除。在大量加载数据前，先删除索引，加载后再重新创建索引，这样做的好处有：
  - 提高加载性能
  - 更有效地使用索引空间
- 仅定期使用的索引无需不必要的维护，尤其在基于易失表时更是如此。这是 OLTP 系统中的通常情况，在该系统中，年末或季度末会生成特殊的查询，以收集在总结会上使用的信息。
- 当在某种类型的操作（如加载）期间出现例程失败时，可能会将索引标记为 `INVALID`。在这种情况下，需要删除并重建索引。
- 索引已损坏。

不能删除约束所需的索引，因此，必须先禁用或删除相关的约束。

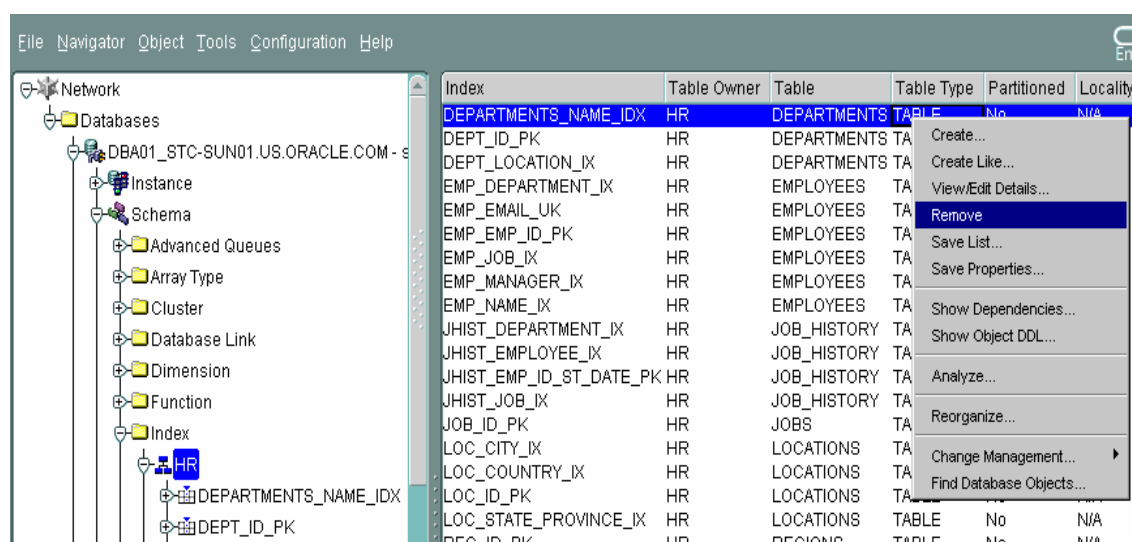
## 删除索引

### 使用 Oracle Enterprise Manager 删除索引

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “索引”(Index)。
2. 展开该索引所属的方案名称。
3. 选择该索引所属的方案名称。
4. 选择索引。
5. 单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
6. 选择相应选项以确认删除。

**注：**如果索引用于实现已启用的完整性约束，则不能删除该索引。约束将在“维护数据完整性”一课中介绍。





## 标识未用索引

- 要开始监视索引的使用，请执行以下语句：

```
ALTER INDEX hr.dept_id_idx
MONITORING USAGE
```

- 要停止监视索引的使用，请执行以下语句：

```
ALTER INDEX hr.dept_id_idx
NOMONITORING USAGE
```

ORACLE

12-29

Copyright © Oracle Corporation, 2001. All rights reserved.

### 标识未用索引

从 Oracle9i 开始，可以在 V\$OBJECT\_USAGE 中收集和显示有关索引使用的统计信息。如果收集的信息表明索引从未使用过，则删除该索引。此外，删除未用索引还可减少 Oracle 服务器用于 DML 操作的开销，从而改善了性能。每次指定 MONITORING USAGE 子句时，将对指定的索引重置 V\$OBJECT\_USAGE。以前的信息被清除或重置，并记录新的开始时间。

#### V\$OBJECT\_USAGE 列

INDEX\_NAME: 索引名

TABLE\_NAME: 对应的表

MONITORING: 指示监视是 ON 还是 OFF

USED: 指示 YES 或 NO，即在监视时间内是否使用了索引

START\_MONITORING: 索引监视的开始时间

END\_MONITORING: 索引监视的结束时间

## 获取索引信息

可以通过查询以下视图来获取有关索引的信息：

- **DBA\_INDEXES**：提供有关索引的信息
- **DBA\_IND\_COLUMNS**：提供有关索引列的信息
- **V\$OBJECT\_USAGE**：提供有关索引使用情况的信息

ORACLE

## 小结

在这一课中，您应该能够掌握：

- 创建各种类型的索引
- 重新组织索引
- 删除索引
- 从数据字典获取索引信息
- 启动和结束对索引使用情况的监视

ORACLE

## 启动和结束对索引使用情况的监视

此练习涉及以下主题：

- 创建表列的索引
- 将索引移到另一个表空间
- 删除索引
- 获取索引信息

ORACLE

12-32

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 12

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 12

- 1 您正考虑在 CUSTOMERS 表的 NAME 列和 REGION 列上创建索引。什么类型的索引适合这两列？创建两个索引，将它们分别命名为 CUST\_NAME\_IDX 和 CUST\_REGION\_IDX，并将它们放在 INDX01 表空间中。

**提示：**B 树索引适用于具有很多唯一值的列，而位图索引适用于具有很多重复值的列。CUSTOMERS 表位于 SYSTEM 方案中。

- 2 将 CUST\_REGION\_IDX 索引移动到另一个表空间。

**提示：**可通过指定不同的表空间来重建索引。

- 3 记下通过 CUST\_REGION\_IDX 索引找到的这些区所使用的文件和块。

**提示：**使用视图 DBA\_EXTENTS 获得此信息。

- 4 使用先删除然后重新创建以外的方法重新创建 CUST\_REGION\_IDX 索引，并将其保留在与以前相同的表空间中。新索引使用的块与以前使用的块相同吗？

**提示：**重建索引。

**注：**重建后的新索引不会重新使用从该区位置处看到的同一空间。这是因为 Oracle 服务器建立临时索引，删除旧索引然后再重命名临时索引。

- 5 **a** 以用户 SYSTEM 的身份，运行脚本 lab12\_05a.sql 以创建和填充 NUMBERS 表。

- b** 查询 NUMBERS 表以查找表的两列中非重复值的数目。

- c** 使用统一的区大小 4KB，分别在 NUMBERS 表的 ODD\_EVEN 列和 NO 列上创建两个 B 树索引 NUMB\_OE\_IDX 和 NUMB\_NO\_IDX。将索引存放在 INDX01 表空间中。检查总的索引大小，并将块数填写到下面的框中。

**提示：**将 PCTINCREASE 设置为 0 可创建大小相同的区。通过 DBA\_SEGMENTS 可检查分配给这些区的总块数。

| 索引          | 列        | 块 |
|-------------|----------|---|
| NUMB_OE_IDX | ODD_EVEN |   |
| NUMB_NO_IDX | NO       |   |



# 13

## 维护数据完整性

ORACLE<sup>®</sup>

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

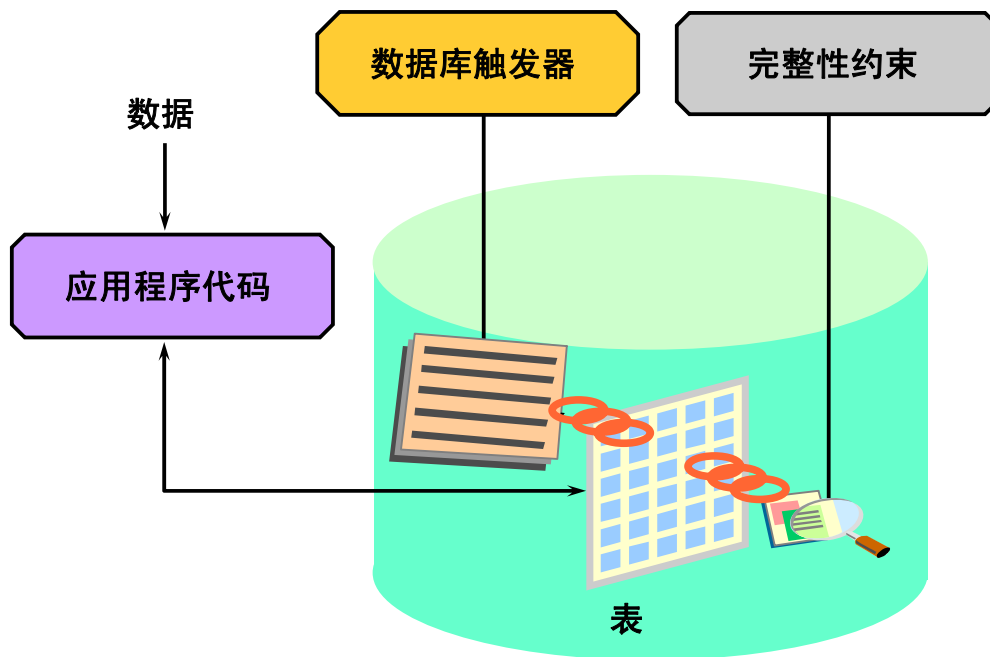
完成这一课的学习后，您应该能达到下列目标：

- 实施数据完整性约束
- 维护完整性约束
- 从数据字典获取约束信息

ORACLE



# 数据完整性



ORACLE

13-3

Copyright © Oracle Corporation, 2001. All rights reserved.

## 数据完整性

数据完整性是指数据库中的数据符合业务规则。维护数据完整性共有三种主要方法：

- 应用程序代码
- 数据库触发器
- 声明完整性约束

具体使用上述哪种方法映射业务规则是设计时应考虑的问题。而数据库管理员主要关心的是实施设计人员选择的方法，并在完整性需求和性能要求之间取得平衡。

应用程序代码既可作为数据库中的存储过程实现，也可作为在客户端上运行的应用程序实现。本课着重讲述完整性约束的使用。

## 保证数据完整性的方法（续）

### 数据库触发器：

数据库触发器是 PL/SQL 程序，在表上发生事件（如插入或更新列）时执行。可以启用或禁用触发器，即可以设置触发器在事件发生时执行，或者将触发器设置为不执行（即使已定义）。通常情况下，创建数据库触发器只是为了强制应用不能定义为完整性约束的复杂业务规则。

注：数据库触发器在其它 Oracle 课程中讲述。

### 完整性约束：

完整性约束是执行业务规则的首选机制，这是因为它可以：

- 改善性能
- 易于声明和修改，不需要进行大量编码
- 集中管理规则
- 使用灵活（启用或禁用）
- 在数据字典中完全文档化

以下部分解释完整性约束的行为，并论述 Oracle 服务器如何执行这些完整性约束。

## 约束的类型

| 约束          | 说明                      |
|-------------|-------------------------|
| NOT NULL    | 指示出列不能包含空值              |
| UNIQUE      | 指示一个列或列的组合是唯一的          |
| PRIMARY KEY | 指示一个列或列的组合作为表的主键        |
| FOREIGN KEY | 指示一个列或列的组合在引用完整性约束中作为外键 |
| CHECK       | 指定表中的每一行必须满足的条件         |

ORACLE

13-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### 约束的类型

缺省情况下，表中的所有列均可以为空。空意味着没有值。NOT NULL 约束要求表列必须包含值。

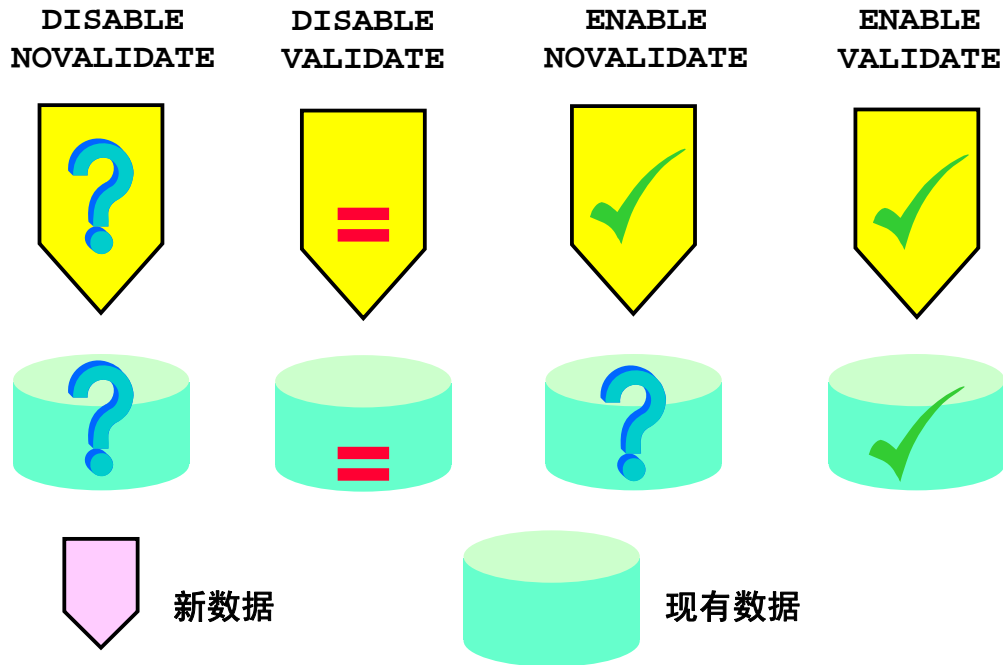
UNIQUE 关键字约束要求某列或一组列（关键字）中的值必须是唯一的。表中的任何两行在指定的一列或一组列中不能有重复的值。

数据库中的每个表至多有一个 PRIMARY KEY 约束。PRIMARY KEY 约束确保以下两种情况均为真：

- 表中的任何两行在指定列中没有重复的值。
- 主键列不包含 NULL 值
- 某列或一组列上的 CHECK 完整性约束要求，对于表的每一行，指定的条件必须为真或未知。

虽然 NOT NULL 和 CHECK 约束并不直接要求 DBA 关注，但 PRIMARY KEY、UNIQUE 和 FOREIGN KEY 约束仍须进行管理，以确保高可用性和性能水平可接受。

## 约束的状态



ORACLE

13-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 约束的状态

可以启用 (ENABLE) 或禁用 (DISABLE) 完整性约束。如果启用某个约束，则在数据库中输入或更新数据时，就会对数据进行检查。禁止输入不符合约束规则的数据。如果禁用某个约束，则可以在数据库中输入不符合约束规则的数据。完整性约束可处于以下状态之一：

- DISABLE NOVALIDATE
- DISABLE VALIDATE
- ENABLE NOVALIDATE
- ENABLE VALIDATE

**DISABLE NOVALIDATE:** 不检查处于 DISABLE NOVALIDATE 状态的约束。表中的数据（包括输入或更新的新数据）可以不符合约束所定义的规则。

**DISABLE VALIDATE:** 当约束处于此状态时，不允许对受约束的列进行任何修改。另外，约束上的索引将被删除并且禁用约束。**注：**如果约束可延迟，则不删除索引。

## 约束的状态（续）

**ENABLE NOVALIDATE:** 如果约束处于此状态，则不能输入违反约束的新数据。但是，表可能包含无效的数据，即数据违反约束。启用处于 NOVALIDATE 状态的约束对正在上传有效 OLTP 数据的数据仓库配置是非常有用的。

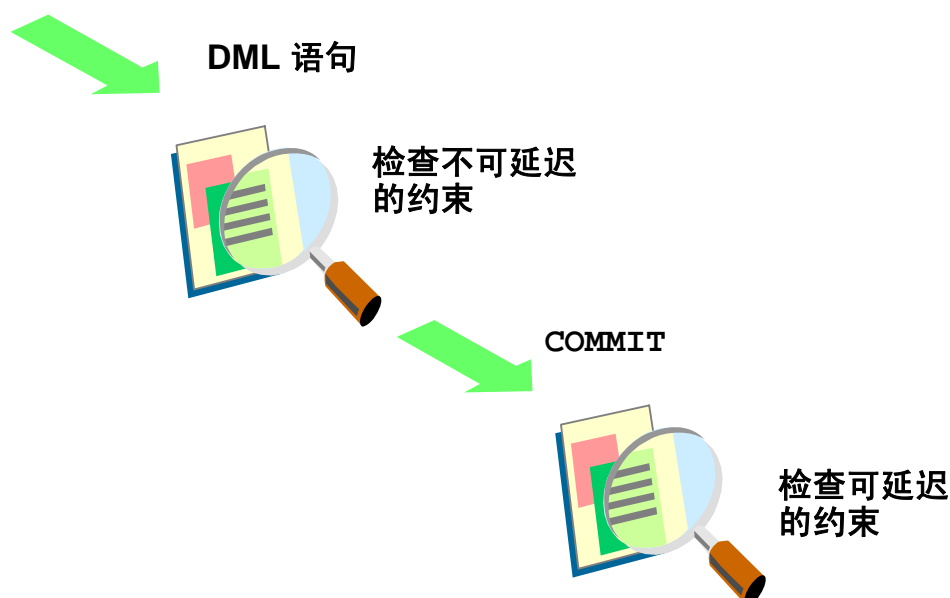
**ENABLE VALIDATE:** 如果约束处于此状态，则不能将违反约束的行插入到表中。但是，禁用该约束时，可以插入此类行。此类行称为该约束的例外。如果约束处于 ENABLE NOVALIDATE 状态，则在禁用约束时输入的数据所引起的违反情况仍然存在。要将约束置于已验证状态，必须更新或删除违反约束的行。

当某一约束由禁用状态更改为 ENABLE VALIDATE 时，将锁定表并对表中的所有数据进行一致性检查。这可能会引起 DML 操作（如等待数据加载），因此，建议先从禁用状态转为 ENABLE NOVALIDATE，然后再转为 ENABLE VALIDATE。

这些状态之间的转换须符合以下规则：

- 除非指定 NOVALIDATE，否则 ENABLE 表示 VALIDATE。
- 除非指定 VALIDATE，否则 DISABLE 表示 NOVALIDATE。
- VALIDATE 和 NOVALIDATE 没有缺省的 ENABLE 和 DISABLE 状态。
- 当唯一键或主键从 DISABLE 状态转为 ENABLE 状态且没有现有索引时，将自动创建唯一索引。（如果索引可延迟，则将存在异常。）与此类似，当唯一键或主键从 ENABLE 转为 DISABLE 且是使用唯一索引启用时，则删除该唯一索引。
- 当任何约束从 NOVALIDATE 状态转为 VALIDATE 状态时，必须检查所有的数据。但是，从 VALIDATE 转为 NOVALIDATE 时，将忽略数据已经过检查这一事实。
- 将单个约束从 ENABLE NOVALIDATE 状态转为 ENABLE VALIDATE 状态时，并不禁止使用读取、写入或其它 DDL 语句。

# 约束检查



ORACLE

13-8

Copyright © Oracle Corporation, 2001. All rights reserved.

## 约束检查

可以将约束有效性检查延迟到事务处理完成时进行。

### 不可延迟的约束或立即约束：

不可延迟的约束（也称立即约束）在每个 DML 语句结束时执行。违反约束将导致语句回退。如果约束导致删除级联等操作，则将该操作作为引起该操作的语句的一部分。不能将定义为不可延迟的约束修改为在事务处理结束时执行。

### 可延迟的约束：

可延迟的约束是仅在提交事务处理时才检查的约束。如果提交时检测到违反约束的行，则回退整个事务处理。当同时输入外键关系中的父行和子行时（如在订单输入系统中同时输入订单和订单所包含的项目），这些约束非常有用。

可以将定义为可延迟的约束指定为下列模式之一：

- “最初为立即”表示，除非另外明确设定，否则在缺省情况下作为立即约束使用。
- “最初为延迟”表示，缺省情况下只在事务处理结束时执行约束。

## 将约束定义为立即或延迟

- 使用 **SET CONSTRAINTS**
- **ALTER SESSION** 语句还包含将约束设置为 **DEFERRED** 或 **IMMEDIATE** 的子句 **SET CONSTRAINTS**。

ORACLE

13-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 将约束定义为立即或延迟

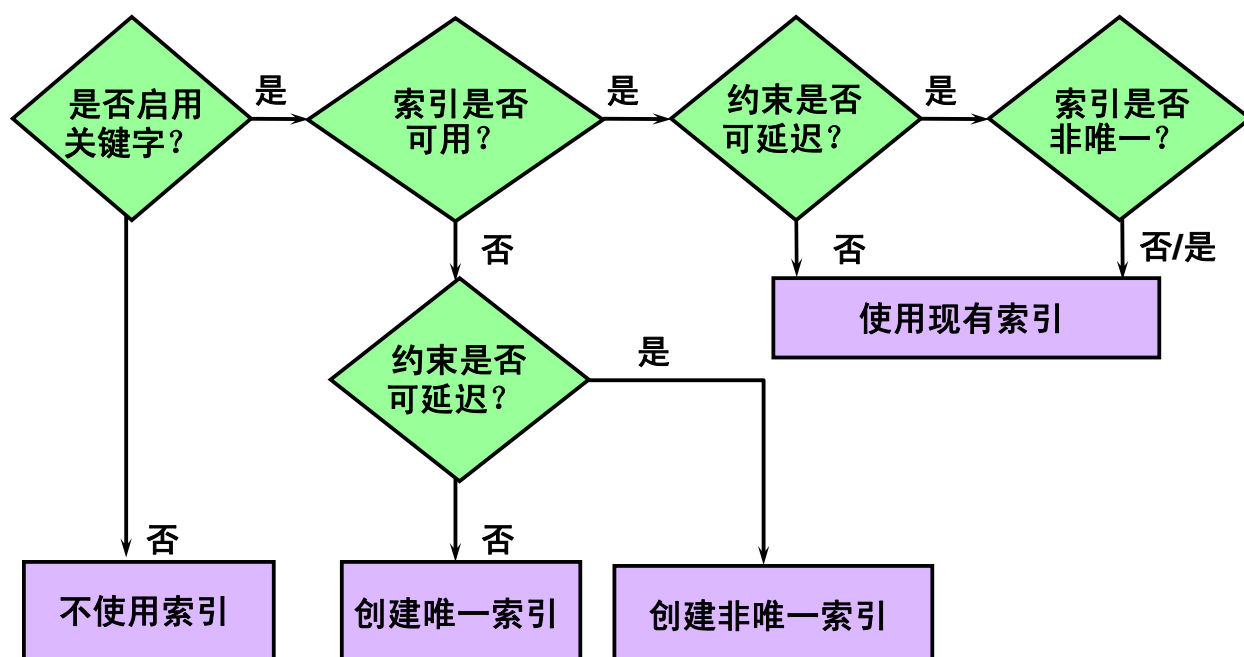
**SET CONSTRAINTS** 语句用于将特定事务处理的约束设置为 **DEFERRED** 或 **IMMEDIATE**。可以使用此语句设置约束名称列表或约束的模式。**SET CONSTRAINTS** 模式将一直持续到事务处理完成或者另一个 **SET CONSTRAINTS** 语句重置模式。**SET CONSTRAINTS** 语句不允许在触发器内部使用。

**ALTER SESSION** 语句还包含将约束设置为 **IMMEDIATE** 或 **DEFERRED** 的子句 **SET CONSTRAINTS**。此命令缺省为设置所有 (**ALL**) 可延迟的约束（不能指定约束名称列表）。**ALTER SESSION SET CONSTRAINTS** 语句仅适用于当前的会话。

```
ALTER SESSION
 SET CONSTRAINT[S] =
 {IMMEDIATE|DEFERRED|DEFAULT}
```

```
SET CONSTRAINT | CONSTRAINTS
 {constraint |ALL }
 {IMMEDIATE|DEFERRED}
```

## 执行主键和唯一键约束



ORACLE

13-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 执行主键和唯一键约束

主键和唯一键通过索引执行。可控制用来执行这些约束的索引的位置和类型。

Oracle 服务器按下列步骤实现唯一键和主键约束：

- 如果约束被禁用，则不需要索引。
- 如果启用约束且约束中的列构成索引的主要部分，则无论是否将索引本身创建为唯一还是非唯一索引，都可以使用该索引执行约束。
- 如果启用约束且没有任何索引将约束列用作索引的主要部分，则按照下列规则创建一个名称与约束相同的索引：
  - 如果关键字为可延迟，则在关键字列上创建一个非唯一索引。
  - 如果关键字为不可延迟，则将创建一个唯一索引。
- 如果可以使用某个索引，并且约束是不可延迟的，则使用现有索引。如果约束是可延迟的，并且索引是非唯一的，则使用现有索引。



## 外键注意事项

| 目标操作          | 相应解决方法                                 |
|---------------|----------------------------------------|
| 删除父表          | 级联约束                                   |
| 截断父表          | 禁用或删除外键                                |
| 删除包含父表的表空间    | 使用 <code>CASCADE CONSTRAINTS</code> 子句 |
| 在子表上执行 DML 操作 | 确保包含父键的表空间联机                           |

ORACLE

13-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### 外键注意事项

维护外键关系中的表时，应该考虑几个因素。

#### 涉及父表的 DDL：

在删除父表之前，必须先删除外键。可以使用以下一条语句同时执行这两个操作：

```
DROP TABLE table CASCADE CONSTRAINTS
```

在未删除或禁用外键之前无法截断父表。

在删除包含父表的表空间之前，必须先删除外键。可使用下列命令完成该操作：

```
DROP TABLESPACE tablespace INCLUDING CONTENTS
CASCADE CONSTRAINTS
```

## 外键注意事项（续）

如果从父表中删除行时没有使用 `DELETE CASCADE` 选项，Oracle 服务器必须确保子表中的行不包含相应的外键。同样，仅当子行中不包含旧键值时，才允许更新父键。如果子表的外键上没有索引，则 Oracle 服务器锁定子表并禁止更改以确保引用完整性。如果表上有索引，则可通过锁定索引项并避免子表上有更具限制性的锁来维护引用完整性。如果必须从不同的事务处理同时更新两个表，则在外键列上创建索引。

当在子表中插入数据或更新子表中的外键列时，Oracle 服务器检查父表上用来执行引用关键字的索引。因此，仅当包含索引的表空间联机时，该操作才能成功。注意，包含父表的表空间在子表上执行 DML 操作时不需要联机。

Oracle9i 在主键上执行更新或删除操作时，不再要求在未建索引的外键上获取共享锁定。它仍然获取表级共享锁定，但在获取后立即释放该锁定。如果更新或删除多个主键，则每行获取和释放一次锁定。

## 创建表时定义约束

```
CREATE TABLE hr.employee(
 id NUMBER(7)
 CONSTRAINT employee_id_pk PRIMARY KEY
 DEFERRABLE
 USING INDEX
 STORAGE(INITIAL 100K NEXT 100K)
 TABLESPACE indx,
 last_name VARCHAR2(25)
 CONSTRAINT employee_last_name_nn NOT NULL,
 dept_id NUMBER(7))
 TABLESPACE users;
```

ORACLE

13-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建表时定义约束

可以在创建或改变表时定义约束。请使用 CREATE TABLE 或 ALTER TABLE 语句中的 constraint\_clause 子句来定义约束。要定义完整性约束，必须具有所需的权限。要创建引用完整性约束，父表必须位于您自己的方案中，或者您必须对父表中的引用键列拥有 REFERENCES 权限。

## 创建表时定义约束（续）

column\_constraint 语法是表定义的一部分。在创建表时，可以使用以下语法定义约束：

```
column datatype [CONSTRAINT constraint]
 {[NOT] NULL
 | UNIQUE [USING INDEX index_clause]
 | PRIMARY KEY [USING INDEX index_clause]
 | REFERENCES [schema.]table [(column)]
 [ON DELETE CASCADE]
 | CHECK (condition)
 }
 constraint_state ::=
 [NOT DEFERRABLE | DEFERRABLE [INITIALLY
 { IMMEDIATE | DEFERRED }]
]
 [DISABLE | ENABLE [VALIDATE | NOVALIDATE]]
```

其中：

**CONSTRAINT:** 使用存储在数据字典中的名称 `constraint` 来标识完整性约束

**USING INDEX:** 指定将 `index_clause` 中定义的参数用于 Oracle 服务器使用的索引，以执行唯一键约束或主键约束（索引的名称与约束的名称相同。）

**DEFERRABLE:** 表示可使用 `SET CONSTRAINT` 命令将约束检查延迟到事务处理结束时

**NOT DEFERRABLE:** 表示在每一 DML 语句结束时检查该约束（会话或事务处理不能延迟 NOT DEFERRABLE 约束。NOT DEFERRABLE 是缺省值。）

**INITIALLY IMMEDIATE:** 表示在每一事务处理开始时，缺省为在每一 DML 语句结束时检查该约束（如果没有指定子句 `INITIALLY`，则缺省情况下为 `INITIALLY IMMEDIATE`。）

**INITIALLY DEFERRED:** 表示该约束为 DEFERRABLE，并指定缺省时只在每一事务处理结束时检查该约束

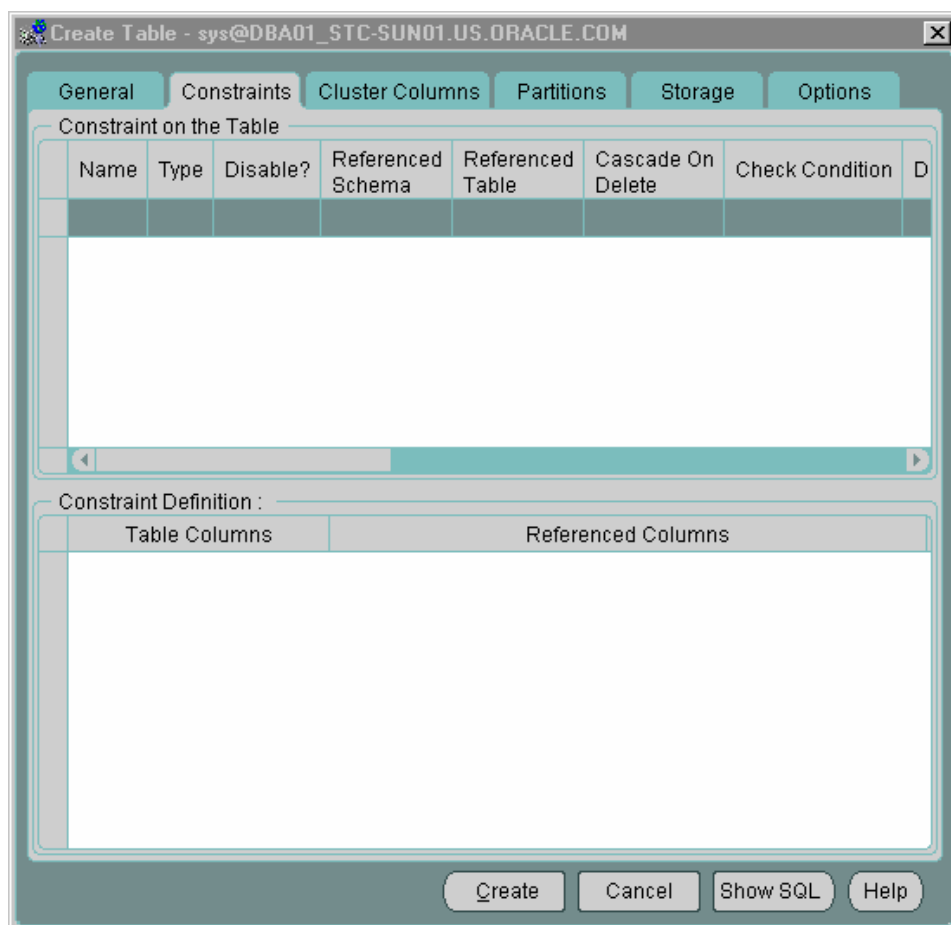
**DISABLE:** 禁用完整性约束（如果禁用完整性约束，则 Oracle 服务器不执行该约束。）

## 创建表时定义约束（续）

### 使用 Oracle Enterprise Manager 定义约束

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 在“常规”(General) 页中完成填写或选择相应信息。
4. 选择“约束”(Constraints) 页，并定义完整性约束。
5. 单击“创建”(Create)。



## 创建表时定义约束（续）

### 表约束：

表约束是表定义的一部分。它可以定义除 NOT NULL 约束以外的任何约束。表约束是使用以下语法定义的：

```
[CONSTRAINT constraint]
{PRIMARY KEY (column [, column]...)
 [USING INDEX index_clause]
|UNIQUE (column [, column]...)
 [USING INDEX index_clause]
|FOREIGN KEY (column [, column]...)
 REFERENCES [schema.]table [(column [, column]...)]
 [ON DELETE CASCADE]
|CHECK (condition)
}
[constraint_state]
```

### 注

- 采用约束的标准命名约定是一个好习惯。这对 CHECK 约束更是如此，因为可使用不同的名称多次创建同一约束。
- 下列情形需要使用表约束：
  - 当约束命名两列或更多列时
  - 改变表以添加除 NOT NULL 约束外的约束时
- 要在创建表后从类型 NOT NULL 定义约束，只能使用以下语句：

```
ALTER TABLE table MODIFY column CONSTRAINT constraint NOT
NULL;
```

### 创建表后定义约束：示例

```
SQL> ALTER TABLE hr.employee
2 ADD(CONSTRAINT employee_dept_id_fk FOREIGN KEY(dept_id)
3 REFERENCES hr.department(id)
4 DEFERRABLE INITIALLY DEFERRED);
```

**注：**本课后面部分的“启用约束”中讲述了 EXCEPTIONS 子句，该子句可用来找出违反约束（通过 ALTER TABLE 命令添加）的行。

## 约束定义原则

- 主约束和唯一性约束：
  - 将索引放在单独的表空间中。
  - 如果经常使用批量加载，请使用非唯一索引。
- 自引用外键：
  - 在初始加载后定义或启用外键。
  - 延迟约束检查。

ORACLE

13-17

Copyright © Oracle Corporation, 2001. All rights reserved.

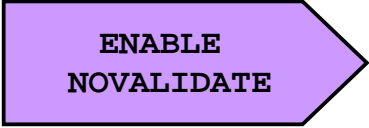
### 约束定义原则

定义约束时遵循下列原则十分有用：

- 将用于执行主键约束和唯一性约束的索引与表放在不同的表空间中。这可通过指定 `USING INDEX` 子句或通过创建表、创建索引并改变表以添加或启用约束来实现。
- 如果经常向表中批量加载数据，则最好先禁用约束，执行完加载后再启用约束。如果唯一索引用于执行主键约束或唯一性约束，则在禁用约束时必须删除该索引。在这种情况下，可以使用非唯一索引执行主键约束或唯一性约束来改善性能：创建可延迟的键，或者在定义或启用键之前创建索引。
- 如果表中包含自引用外键，请使用下列方法之一加载数据：
  - 在初始加载后定义或启用外键。
  - 将约束定义为可延迟的约束。

在频繁加载数据的情况下，第二种方法非常有用。

## 启用约束



ENABLE  
NOVALIDATE

- 没有表锁定
- 主键和唯一键必须使用非唯一索引

```
ALTER TABLE hr.departments
ENABLE NOVALIDATE CONSTRAINT dept_pk;
```

ORACLE

13-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用约束

可以采用下列两种方法之一来启用当前禁用的约束：ENABLE NOVALIDATE 或 ENABLE VALIDATE

#### 启用 NOVALIDATE:

对于当前已有索引的 PRIMARY KEY 和 UNIQUE 约束，启用 NOVALIDATE 约束比启用 VALIDATE 约束要快得多，这是因为，如果约束是可延迟的，则不检查现有数据是否违反约束。如果使用该选项启用约束，则不要求锁定表。这种方法适合表上有许多 DML 活动的情况，如在 OLTP 环境中。

下列命令可用于启用 ENABLE NOVALIDATE 约束：

```
ALTER TABLE [schema.] table
ENABLE NOVALIDATE {CONSTRAINT constraint
 | PRIMARY KEY
 | UNIQUE (column [, column] ...) }
[USING INDEX index_clause]
```



## 启用约束（续）

### 限制：

USING INDEX 子句仅适用于创建为可延迟的主键约束或唯一性约束，并且下列条件之一为真的情况：

- 约束被创建为禁用。
- 约束被禁用且索引已删除。

但是，如果需要创建索引，使用这种启用约束的方法并不能比 ENABLE VALIDATE 带来更多的好处，因为 Oracle 服务器在建立索引时锁定表。

注：有关禁用约束的讨论，请参见 *SQL 和 PL/SQL 简介* 课程。

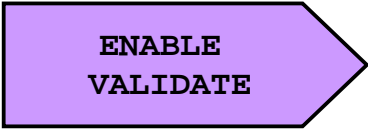
## 启用约束（续）

### 使用 Oracle Enterprise Manager 定义约束

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “方案”(Schema) > “表”(Table)。
2. 选择要在其中修改约束的表。
3. 单击“约束”(Constraints) 选项卡并进行相应的修改。
4. 单击“应用”(Apply)。

## 启用约束



ENABLE  
VALIDATE

- 锁定表
- 可以使用唯一或非唯一的索引
- 需要有效的表数据

```
ALTER TABLE hr.employees
ENABLE VALIDATE CONSTRAINT emp_dept_fk;
```

ORACLE

13-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用约束

启用 VALIDATE 约束后将检查现有数据中是否违反约束。这是启用约束时的缺省操作。若在禁用约束时执行，则会产生下列影响：

- 锁定表，以防在验证完现有数据前对表进行更改。
- 如果索引列上不存在索引，Oracle 服务器就会创建一个索引。当启用不可延迟的主键约束或唯一性约束时，Oracle 服务器将创建一个唯一索引。对于可延迟的主键约束或唯一性约束，将建立一个非唯一索引。

如果在执行约束时执行此命令，则不要求在验证过程中锁定任何表。执行的约束将保证在验证期间不会出现违反约束的情况。这有如下好处：

- 所有约束并发启用。
- 每一约束在内部保持并行。
- 允许表上存在并发操作。

## 启用约束（续）

使用以下命令启用约束 `ENABLE VALIDATE`:

```
ALTER TABLE [schema.] table
ENABLE [VALIDATE] {CONSTRAINT constraint
 | PRIMARY KEY
 | UNIQUE (column [, column] ...) }
[USING INDEX index_clause]
[EXCEPTIONS INTO [schema.] table]
```

注:

- `VALIDATE` 选项为缺省设置，不需要在启用被禁用约束时指定。
- 如果表中的数据违反约束，则语句回退，约束仍被禁用。
- 下面部分将讨论 `EXCEPTIONS` 子句的使用。

## 使用 EXCEPTIONS 表

- 通过运行 `utlexcpt1.sql` 脚本来创建 **EXCEPTIONS** 表。
- 执行带有 **EXCEPTIONS** 选项的 **ALTER TABLE** 语句。
- 使用 **EXCEPTIONS** 上的子查询定位包含无效数据的行。
- 纠正错误。
- 重新执行 **ALTER TABLE** 以启用约束。

ORACLE

13-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### 如何识别行违反

**EXCEPTIONS** 子句标识出任何违反已启用约束的行。使用下列步骤检测违反约束的行为，纠正它们并重新启用约束：

1. 如果尚未创建 **EXCEPTIONS**，则运行 `utlexcpt1.sql` 脚本：

```
SQL> @?/rdbms/admin/utlexcpt1
```

```
Statement processed.
```

```
SQL> DESCRIBE exceptions
```

| Name       | Null? | Type         |
|------------|-------|--------------|
| -----      | ----- | -----        |
| ROW_ID     | ROWID |              |
| OWNER      |       | VARCHAR2(30) |
| TABLE_NAME |       | VARCHAR2(30) |
| CONSTRAINT |       | VARCHAR2(30) |

**注：**`utlexcpt1.sql` 脚本的确切名称和位置视操作系统而定。有关详细信息，请参阅专用于该操作系统的 Oracle 文档。

## 如何识别行违反（续）

2. 执行使用 EXCEPTIONS 子句的 ALTER TABLE 命令：

```
SQL> ALTER TABLE hr.employee
 2 ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
 3 EXCEPTIONS INTO system.exceptions;
ALTER TABLE hr.employee
*
ORA-02298: cannot enable (HR.EMPLOYEE_DEPT_ID_FK) - parent keys
not found
```

如果未使用所有者姓名限定 EXCEPTIONS 表，则它必须属于被修改的表的所有者。

将行插入到 EXCEPTIONS 表中。如果重新运行该命令，将截断 EXCEPTIONS 表以删除全部现有的行。

3. 使用 EXCEPTIONS 表上的子查询标识无效数据：

```
SQL> SELECT rowid, id, last_name, dept_id
 2 FROM hr.employee
 3 WHERE ROWID in (SELECT row_id
 4 FROM exceptions)
 5 FOR UPDATE;

ROWID ID LAST_NAME DEPT_ID

AAAAeyAADA AAAA1AAA 1003 Pirie 50

1 row selected.
```

4. 更正数据中的错误：

```
SQL> UPDATE hr.employee
 2 SET dept_id=10
 3 WHERE rowid='AAAAeyAADA AAAA1AAA';

1 row processed.

SQL> COMMIT;

Statement processed.
```

## 如何识别行违反（续）

5. 截断 EXCEPTIONS 表并重新启用约束：

```
SQL> TRUNCATE TABLE exceptions;
```

```
Statement processed.
```

```
SQL> ALTER TABLE hr.employee
```

```
2 ENABLE VALIDATE CONSTRAINT employee_dept_id_fk
```

```
3 EXCEPTIONS INTO system.exceptions;
```

```
Statement processed.
```

## 获取约束信息

通过查询以下视图获取有关约束的信息：

- **DBA\_CONSTRAINTS**
- **DBA\_CONS\_COLUMNS**

ORACLE

13-26

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取约束信息

使用下列查询获得 HR 的 EMPLOYEE 表上所有约束的名称、类型和状态：

```
SQL> SELECT constraint_name, constraint_type, deferrable,
2 deferred, validated
3 FROM dba_constraints
4 WHERE owner='HR'
5 AND table_name='EMPLOYEES';
```

| CONSTRAINT_NAME C | DEFERRABLE DEFERRED VALIDATED      |
|-------------------|------------------------------------|
| -----             | -----                              |
| EMPLOYEE_DEPT.. R | DEFERRABLE DEFERRED VALIDATED      |
| EMPLOYEE_ID_PK P  | DEFERRABLE IMMEDIATE VALIDATED     |
| SYS_C00565 C      | NOT DEFERRABLE IMMEDIATE VALIDATED |

3 rows selected.



获取约束信息（续）

下表列出 DBA\_CONSTRAINTS 视图中根据名称不易确定用途的列：

| 名称                           | 说明                                                                           |
|------------------------------|------------------------------------------------------------------------------|
| CONSTRAINT_TYPE              | 如果为主键约束，则约束类型为 P；如果为唯一性约束，则为 U；如果为外键约束，则为 R；如果为检查约束，则为 C。NOT NULL 约束存储为检查约束。 |
| SEARCH_CONDITION             | 显示为检查约束指定的条件                                                                 |
| R_OWNER<br>R_CONSTRAINT_NAME | 为外键定义引用约束的所有者和名称                                                             |
| GENERATED                    | 指示约束名是否由系统生成（有效值为 USERNAME 和 GENERATED NAME。）                                |
| BAD                          | 指示将重写约束以避免千年虫等问题                                                             |
| RELY                         | 如果设置此标志，则此标志用于优化程序                                                           |
| LAST_CHANGE                  | 显示上次启用或禁用约束的日期                                                               |

约束中的列：

要获得 HR 的 EMPLOYEE 表内约束中的列，请使用下列查询：

```
SQL> SELECT c.constraint_name, c.constraint_type,
2 cc.column_name
3 FROM dba_constraints c, dba_cons_columns cc
4 WHERE c.owner='HR'
5 AND c.table_name='EMPLOYEE'
6 AND c.owner = cc.owner
7 AND c.constraint_name = cc.constraint_name
8 ORDER BY cc.position;

CONSTRAINT_NAME C COLUMN_NAME

EMPLOYEE_DEPT... R DEPT_ID
EMPLOYEE_ID_PK P ID
SYS_C00565 C LAST_NAME

3 rows selected.
```

## 获取约束信息（续）

### 查找主键-外键关系：

要查找 HR 的 EMPLOYEE 表上的外键和父约束，请使用下列查询：

```
SQL> SELECT c.constraint_name AS "Foreign Key",
 2 p.constraint_name AS "Referenced Key",
 3 p.constraint_type,
 4 p.owner,
 5 p.table_name
 6 FROM dba_constraints c, dba_constraints p
 7 WHERE c.owner='HR'
 8 AND c.table_name='EMPLOYEE'
 9 AND c.constraint_type='R'
 10 AND c.r_owner=p.owner
 11 AND c.r_constraint_name = p.constraint_name;
```

```
Foreign Key Referenced Key C OWNER TABLE_NAME
```

```

EMPLOYEES_DEPT.. DEPT_PK P HR DEPARTMENT
```

```
1 row selected.
```

## 小结

在这一课中，您应该能够掌握：

- 实现数据完整性
- 使用相应的策略创建和维护约束
- 从数据字典获取信息

ORACLE

## 练习 13

此练习涉及以下主题：

- 创建约束
- 启用唯一性约束
- 创建 EXCEPTIONS 表
- 找出表中现有的约束违反情况，纠正错误，并重新启用约束

ORACLE

## 练习 13

- 1 检查 lab13\_01.sql 脚本。运行该脚本以创建约束。
- 2 查询数据字典以：
  - a 检查约束，确定它们是否可延迟以及它们的状态  
提示：使用 DBA\_CONSTRAINTS 视图获取此信息。
  - b 检查所创建的索引名称及类型以验证约束  
提示：只能为主键约束和唯一性约束创建索引，且索引与约束应具有相同的名称。
- 3 以用户 SYSTEM 的身份运行 lab13\_03.sql 脚本，将两个记录插入到 PRODUCTS 表中。
- 4 启用 PRODUCT 表上的唯一性约束。能否成功？
- 5 a 确保添加到表中的新行不违反 PRODUCT 表上的约束。  
提示：这可以通过启用 NOVALIDATE 约束来完成。
  - b 查询数据字典以验证更改结果。
  - c 通过添加包含下列值的行，测试约束是否禁用违反更改的插入：

| PRODUCT_ID | PRODUCT_DESCRIPTION | LIST_PRICE |
|------------|---------------------|------------|
| 4000       | Monitor             | 3000       |

- 6 采取必要的步骤找出 PRODUCTS 表中现有的违反约束的行，根据需要修改产品代码，  
确保所有现有数据和新数据均不违反约束。（假设表有上千行，手动验证每行太浪费时间。）  
提示：使用下列步骤：
  - a 创建 EXCEPTIONS 表。
  - b 运行该命令以启用约束并捕获异常错误。
- 7 c 使用 EXCEPTIONS 表中的行标识列出 PRODUCTS 表中违反约束的行。不要列出大型对象 (LOB) 列。
  - d 纠正错误。
  - e 启用约束。
- 8 运行 lab13\_07.sql 脚本将这些行插入到表中。插入能否成功？回退更改。



# 14

## 管理口令安全性和资源

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

# 目标

完成这一课的学习后，您应该能达到下列目标：

- 使用配置文件管理口令
- 管理配置文件
- 使用配置文件控制资源的使用
- 获取有关配置文件、口令管理和资源的信息

ORACLE



## 配置文件

- 配置文件是口令和资源限制的指定集合。
- 通过 **CREATE USER** 或 **ALTER USER** 命令可以将配置文件分配给用户。
- 既可启用也可禁用配置文件。
- 配置文件可以与 **DEFAULT** 配置文件相关。

ORACLE

14-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### 配置文件

配置文件是下列口令和资源限制的指定集合：

- 口令过期和失效
- 口令历史记录
- 口令复杂性校验
- 帐户锁定
- CPU 时间
- 输入/输出 (I/O) 操作
- 空闲时间
- 连接时间
- 内存空间（仅用于共享服务器的 SQL 专用区）
- 并发会话

创建配置文件后，数据库管理员可以将它分配给各个用户。如果启用了资源限制，则 Oracle 服务器将数据库的使用和资源限制在定义的用户配置文件所允许的范围内。

## 配置文件（续）

创建数据库时，Oracle 服务器自动创建 DEFAULT 配置文件。

没有被显式分配特定配置文件的用户遵从 DEFAULT 配置文件的所有限制。DEFAULT 配置文件的所有限制最初为无限制。不过，数据库管理员可以更改这些值，让这些限制在缺省情况下适用于所有用户。

### 配置文件的用途：

- 限制用户执行某些需要大量资源的操作。
- 确保在用户会话空闲一段时间后，将用户从数据库注销。
- 对相似的用户启用组资源限制。
- 便于为用户分配资源限制。
- 管理大型、复杂的多用户数据库系统中的资源使用。
- 控制口令的使用

### 配置文件的特点：

- 配置文件的分配不影响当前会话。
- 配置文件只能分配给用户，而不能分配给角色或其它配置文件。
- 如果创建用户时未分配配置文件，则系统将 DEFAULT 配置文件自动分配给用户。

## 口令管理



ORACLE

14-5

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令管理

为了更严格地控制数据库的安全，数据库管理员使用配置文件对 Oracle 口令管理进行控制。

本课讲述可用的口令管理功能：

- 帐户锁定：如果用户在指定的次数内未能登录到系统中，则启用自动帐户锁定
- 口令过期和失效：指定口令的生存期，此生存期过后口令将失效，必须更改
- 口令历史记录：检查新口令，确保在指定的时间长度内或指定的口令更改次数内不重新使用此口令
- 口令复杂性校验：检查口令的复杂性，验证其复杂性能否阻止通过猜测口令试图闯入系统的入侵者

## 启用口令管理

- 使用配置文件并将它们分配给用户，由此设立口令管理。
- 使用 `CREATE USER` 或 `ALTER USER` 命令可以对帐户进行锁定、解除锁定，或使帐户失效。
- 始终执行口令限制。
- 要启用口令管理，请以用户 `sys` 的身份运行 `utlpwdmg.sql` 脚本。

ORACLE

14-6

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用口令管理

创建配置文件以限制口令设置，然后使用 `CREATE USER` 或 `ALTER USER` 命令将配置文件分配给用户。

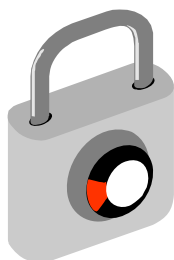
始终执行配置文件中的口令限制设置。

如果启用了口令管理，则可以使用 `CREATE USER` 或 `ALTER USER` 命令锁定或解除锁定用户帐户。

**注：**有关 `CREATE USER` 命令的详细信息，请参考“管理用户”一课。

## 口令帐户锁定

| 参数                    | 说明                    |
|-----------------------|-----------------------|
| FAILED_LOGIN_ATTEMPTS | 锁定帐户前登录失败的次数          |
| PASSWORD_LOCK_TIME    | 达到指定的登录失败次数后，要锁定帐户的天数 |



ORACLE

14-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令帐户锁定

达到 FAILED\_LOGIN\_ATTEMPTS 值后，Oracle 服务器会自动锁定帐户。帐户应该在 PASSWORD\_LOCK\_TIME 参数定义的指定时间后自动解除锁定，如果不是这样，则必须由数据库管理员使用 ALTER USER 命令解除锁定。

可以使用 ALTER USER 命令显式锁定数据库帐户。但在这种情况下，帐户不能自行解除锁定。

**注：**本课后续部分中将演示如何执行 ALTER USER 命令。

## 口令失效和过期

| 参数                               | 参数                            |
|----------------------------------|-------------------------------|
| <code>PASSWORD_LIFE_TIME</code>  | 口令在失效前的生存期，单位是天               |
| <code>PASSWORD_GRACE_TIME</code> | 口令失效后从第一次成功登录算起的更改口令的宽限期，单位是天 |



ORACLE

14-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令失效和过期

`PASSWORD_LIFE_TIME` 参数设置口令的最长生存期，此生存期过后必须更改口令。

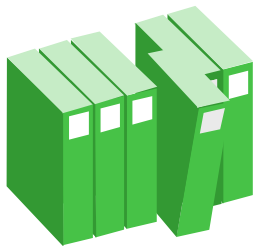
数据库管理员可以指定一个宽限期 (`PASSWORD_GRACE_TIME`)，此宽限期从口令失效后第一次试图登录到数据库开始算起。宽限期内，用户每次试图登录到数据库时都生成一条警告消息。用户应在宽限期内更改口令。

如未更改口令，将锁定帐户。

如果通过显式设置让口令失效，用户帐户状态便更改为 `EXPIRED`。

## 口令历史记录

| 参数                  | 说明            |
|---------------------|---------------|
| PASSWORD_REUSE_TIME | 可以重新使用口令前的天数  |
| PASSWORD_REUSE_MAX  | 可以重新使用口令的最多次数 |



ORACLE

14-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令历史记录

口令历史记录检查可确保用户在指定的时间间隔内不能重新使用口令。可使用下列参数之一执行检查：

- PASSWORD\_REUSE\_TIME：指定用户在规定天数内不能重新使用口令
- PASSWORD\_REUSE\_MAX：强制用户定义与前一口令不同的口令

如果一个参数设置成 DEFAULT 或 UNLIMITED 以外的值，则另一个参数必须设置为 UNLIMITED。

## 口令校验

| 参数                       | 说明                        |
|--------------------------|---------------------------|
| PASSWORD_VERIFY_FUNCTION | PL/SQL 函数，可在分配口令前检查口令的复杂性 |

ORACLE

14-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令校验

向用户分配新口令前，可以调用 PL/SQL 函数校验口令的有效性。

Oracle 服务器提供缺省的校验例行程序，数据库管理员也可以编写 PL/SQL 函数。



## 用户提供的口令函数

这个函数必须在 `sys` 方案中创建，并且必须按如下方式指定：

```
function_name(
 userid_parameter IN VARCHAR2(30),
 password_parameter IN VARCHAR2(30),
 old_password_parameter IN VARCHAR2(30))
RETURN BOOLEAN
```

ORACLE

14-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### 用户提供的口令函数

添加新的口令校验函数时，数据库管理员必须考虑下列限制：

- 此过程必须使用幻灯片中显示的说明。
- 成功时此过程返回 `TRUE` 值，失败时返回 `FALSE` 值。
- 如果口令函数产生异常错误，将返回错误消息并终止 `ALTER USER` 或 `CREATE USER` 命令。
- 口令函数由 `SYS` 所有。
- 如果口令函数无效，则返回错误消息并终止 `ALTER USER` 或 `CREATE USER` 命令。

注：有关 `CREATE USER` 的详细信息，请参考“管理用户”一课。

## 口令校验函数 VERIFY\_FUNCTION

- 最短为四个字符。
- 口令不应与用户名相同。
- 口令至少应该包含一个字母、一个数字和一个特殊字符。
- 新口令与旧口令相比，应至少有三个字母不同。

ORACLE

14-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 口令校验函数

Oracle 服务器以缺省的 PL/SQL 函数的形式（在脚本 utlpwdmg.sql 中称为 VERIFY\_FUNCTION）提供了复杂性校验函数，它必须在 SYS 方案中运行。

执行脚本 utlpwdmg.sql 的过程中，Oracle 服务器使用下面的 ALTER PROFILE 命令创建 VERIFY\_FUNCTION 并更改 DEFAULT 配置文件：

```
SQL> ALTER PROFILE DEFAULT LIMIT
2 PASSWORD_LIFE_TIME 60
3 PASSWORD_GRACE_TIME 10
4 PASSWORD_REUSE_TIME 1800
5 PASSWORD_REUSE_MAX UNLIMITED
6 FAILED_LOGIN_ATTEMPTS 3
7 PASSWORD_LOCK_TIME 1/1440
8 PASSWORD_VERIFY_FUNCTION verify_function;
```

## 创建配置文件： 口令设置

```
CREATE PROFILE grace_5 LIMIT
 FAILED_LOGIN_ATTEMPTS 3
 PASSWORD_LOCK_TIME UNLIMITED
 PASSWORD_LIFE_TIME 30
 PASSWORD_REUSE_TIME 30
 PASSWORD_VERIFY_FUNCTION verify_function
 PASSWORD_GRACE_TIME 5;
```

ORACLE

14-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建配置文件

使用下面的 CREATE PROFILE 命令可以管理口令：

```
CREATE PROFILE profile LIMIT
 [FAILED_LOGIN_ATTEMPTS max_value]
 [PASSWORD_LIFE_TIME max_value]
 [{PASSWORD_REUSE_TIME
 |PASSWORD_REUSE_MAX} max_value]
 [PASSWORD_LOCK_TIME max_value]
 [PASSWORD_GRACE_TIME max_value]
 [PASSWORD_VERIFY_FUNCTION
 {function|NULL|DEFAULT}]
```

## 创建配置文件（续）

其中

PROFILE: 是要创建的配置文件的名称

FAILED\_LOGIN\_ATTEMPTS: 指定在锁定帐户之前，试图登录用户帐户的失败次数

PASSWORD\_LIFE\_TIME: 限制使用同一口令进行验证的天数。如果在此期限内不更改口令，则口令将过期，而此后的连接将被拒绝

PASSWORD\_REUSE\_TIME: 指定天数，此天数过后才能重新使用口令。如果将 PASSWORD\_REUSE\_TIME 设置为整数值，则必须将 PASSWORD\_REUSE\_MAX 设置为 UNLIMITED。

PASSWORD\_REUSE\_MAX: 指定在能够重新使用当前口令之前，需要对口令进行更改的次数。如果将 PASSWORD\_REUSE\_MAX 设置为整数值，则必须将 PASSWORD\_REUSE\_TIME 设置为 UNLIMITED。

PASSWORD\_LOCK\_TIME: 在连续的登录尝试失败达到指定的数量后，指定帐户将被锁定的天数。

PASSWORD\_GRACE\_TIME: 指定宽限期开始后延续的天数，在此期间将不断发出警告，但允许登录。如果在宽限期内未更改口令，口令将会失效。

PASSWORD\_VERIFY\_FUNCTION: 允许将 PL/SQL 口令复杂性校验函数作为参数传递给 CREATE PROFILE 语句。

## 创建配置文件

### 使用 Oracle Enterprise Manager 创建配置文件

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “安全性”(Security) > “配置文件”(Profiles)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 输入配置文件的名称，并完成填写其它字段或接受缺省值。
4. 选择“口令”(Password) 选项卡，然后输入帐户口令参数。
5. 单击“创建”(Create)。

Create Profile - sys@DBA01\_STC-SUN01.US.ORACLE.COM

General Password

Name:

Details

CPU/Session:  Sec./100

CPU/Call:  Sec./100

Connect Time:  Minutes

Idle Time:  Minutes

Database Services

Concurrent Sessions:  Per User

Reads/Session:  Blocks

Reads/Call:  Blocks

Private SGA:  KBytes

Composite Limit:  Service Units

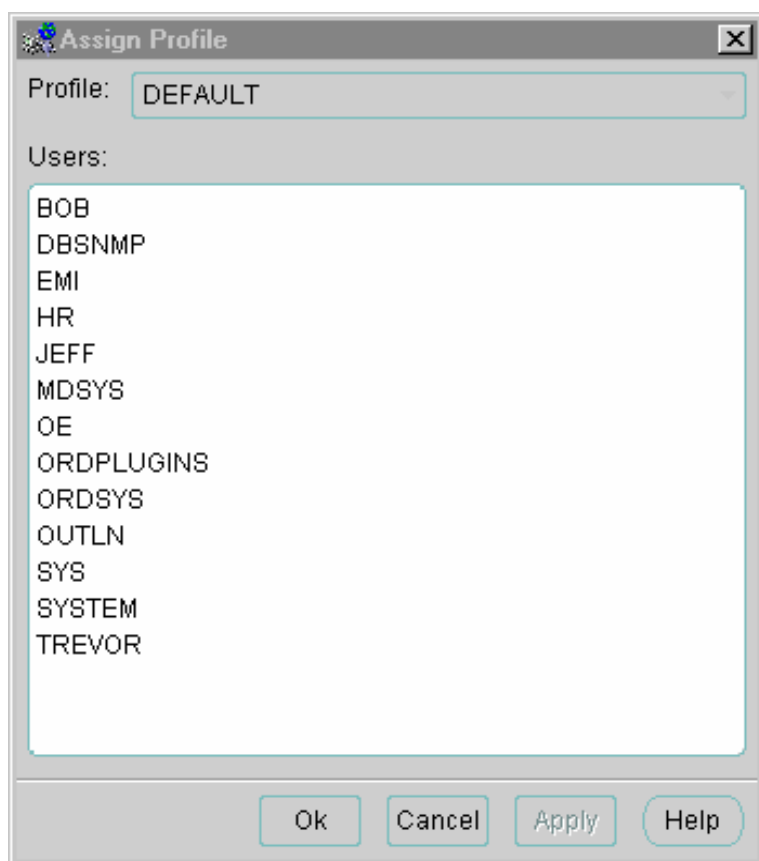
Create Cancel Show SQL Help

## 创建配置文件

### 使用 Oracle Enterprise Manager 创建配置文件

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “安全性”(Security) > “配置文件”(Profiles)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 单击鼠标右键，从弹出的菜单中选择“对象”(Object) > “为用户分配配置文件”(Assign a Profile to User(s))。
4. 选择用户。
5. 单击“确定”(OK)。



# 改变配置文件：口令设置

## 使用 ALTER PROFILE 可更改口令限制

```
ALTER PROFILE default LIMIT
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME 60
PASSWORD_GRACE_TIME 10;
```

ORACLE

14-17

Copyright © Oracle Corporation, 2001. All rights reserved.

## 改变配置文件

使用 ALTER PROFILE 命令可更改对配置文件指定的口令限制：

```
ALTER PROFILE profile LIMIT
[FAILED_LOGIN_ATTEMPTS max_value]
[PASSWORD_LIFE_TIME max_value]
[{PASSWORD_REUSE_TIME
 |PASSWORD_REUSE_MAX} max_value]
[PASSWORD_LOCK_TIME max_value]
[PASSWORD_GRACE_TIME max_value]
[PASSWORD_VERIFY_FUNCTION
 {function|NULL|DEFAULT}]
```

若要将口令参数设置为不超过一天：

1 小时： PASSWORD\_LOCK\_TIME = 1/24

10 分钟： PASSWORD\_LOCK\_TIME = 10/1400

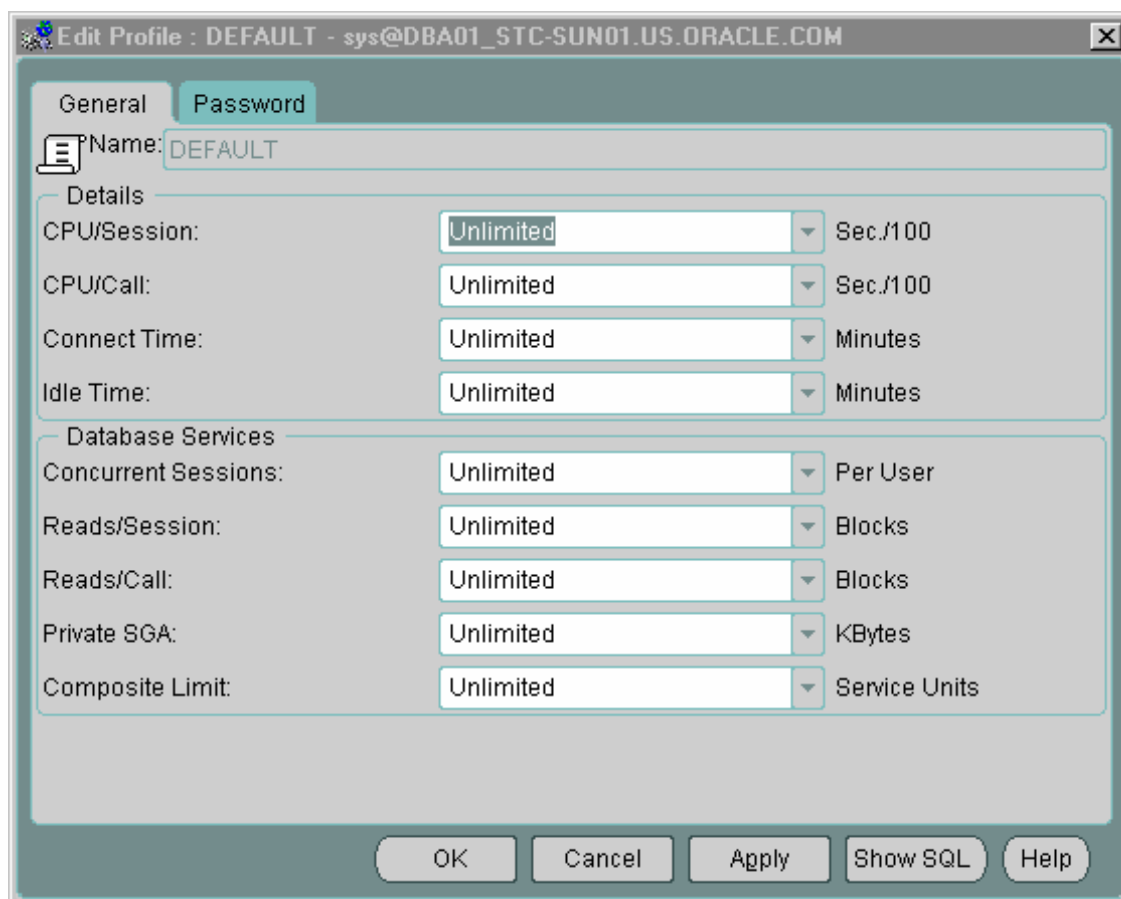
5 分钟： PASSWORD\_LOCK\_TIME = 5/1440

## 如何使用 Oracle Enterprise Manager 改变配置文件

### 使用 Oracle Enterprise Manager 改变配置文件

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “安全性”(Security) > “配置文件”(Profiles)。
2. 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
3. 选择配置文件。
4. 单击鼠标右键，从弹出的菜单中选择“查看/编辑详细资料”(View/Edit Details)。
5. 选择“口令”(Password) 页并改变配置文件。
6. 单击“应用”(Apply)。





## 删除配置文件：口令设置

- 使用 DROP PROFILE 命令删除配置文件。
- 不能删除 DEFAULT 配置文件。
- CASCADE 从分配有配置文件的用户处撤消配置文件。

```
DROP PROFILE developer_prof;
```

```
DROP PROFILE developer_prof CASCADE;
```

ORACLE

14-19

Copyright © Oracle Corporation, 2001. All rights reserved.

### 删除配置文件：口令设置

使用 DROP PROFILE 命令删除配置文件：

```
DROP PROFILE profile [CASCADE]
```

其中：

profile：是要删除的配置文件的名称

CASCADE：从分配有配置文件的用户处撤消配置文件（Oracle 服务器自动将 DEFAULT 配置文件分配给这些用户。指定此选项可以删除当前分配给用户的配置文件。）

原则：

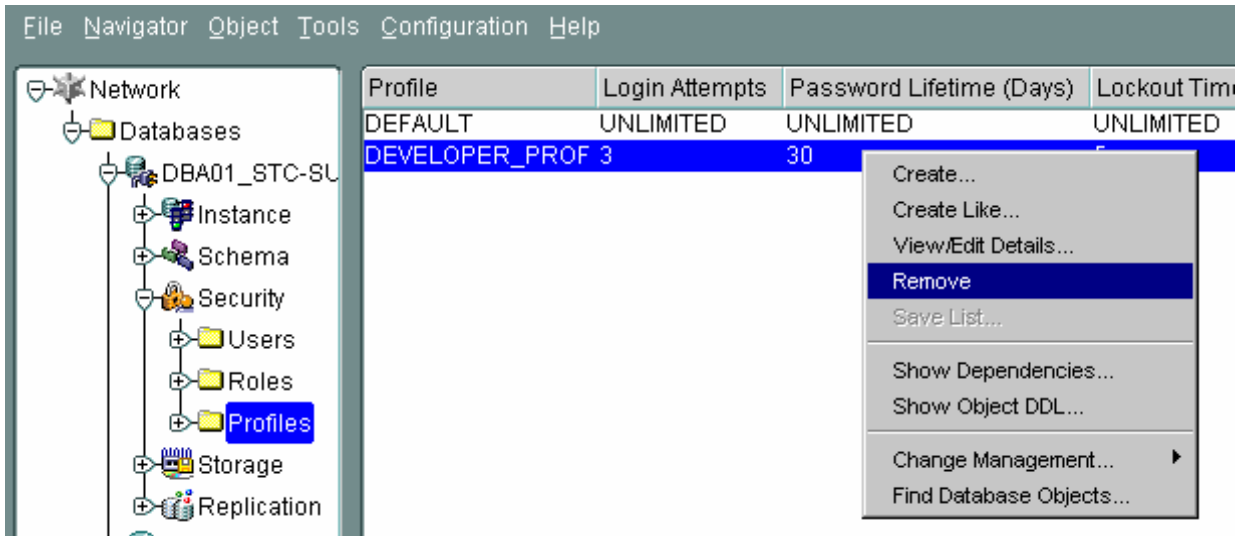
- 不能删除 DEFAULT 配置文件。
- 删除配置文件后，此更改仅适用于此后创建的会话，而不适用于当前会话。

# 删除配置文件

## 使用 Oracle Enterprise Manager 删除配置文件

从 “OEM 控制台” (OEM Console):

- 1. 导航到 “数据库” (Databases) > “安全性” (Security) > “配置文件” (Profiles)。
- 2. 选择配置文件。
- 3. 单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。
- 4. 选择 “是” (Yes) 确认删除。



## 资源管理

- 可以在会话级、调用级或同时在这两个级别上执行资源管理限制。
- 通过使用 `CREATE PROFILE` 命令创建的配置文件可以定义限制。
- 通过以下方法可启用资源限制：
  - `RESOURCE_LIMIT` 初始化参数
  - `ALTER SYSTEM` 命令

ORACLE

14-21

Copyright © Oracle Corporation, 2001. All rights reserved.

### 资源管理

使用下列步骤，通过配置文件控制资源的使用：

1. 使用 `CREATE PROFILE` 命令创建配置文件，以确定资源和口令限制。
2. 使用 `CREATE USER` 或 `ALTER USER` 命令分配配置文件。
3. 通过使用 `ALTER SYSTEM` 命令或编辑初始化参数文件（并停止和重新启动例程），执行资源限制。

这些步骤将在后面章节详细讨论。

注：启用 Oracle 口令管理时无需执行资源限制。

## 启用资源限制

- 将初始化参数 `RESOURCE_LIMIT` 设置为 `TRUE`。
- 使用 `ALTER SYSTEM` 命令启用资源限制参数也可以执行该限制。

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE;
```

ORACLE

14-22

Copyright © Oracle Corporation, 2001. All rights reserved.

### 启用资源限制

通过改变 `RESOURCE_LIMIT` 初始化参数或使用 `ALTER SYSTEM` 命令，启用或禁用资源限制的执行。

#### **RESOURCE\_LIMIT 初始化参数：**

- 若要启用或禁用资源限制的执行，请改变初始化文件中的此参数并重新启动例程。
- `TRUE` 值启用执行。
- `FALSE` 值禁用执行（缺省）。
- 使用这个参数可以执行对整个体系结构的限制。

#### **ALTER SYSTEM 命令：**

- 若要对某一例程启用或禁用资源限制的执行，请使用 `ALTER SYSTEM` 命令。
- 在改动使用 `ALTER SYSTEM` 命令指定的设置或关闭数据库之前，此设置一直有效。
- 无法关闭数据库时，可使用此命令启用或禁用执行。

## 在会话级设置资源限制

| 资源                        | 说明                             |
|---------------------------|--------------------------------|
| CPU_PER_SESSION           | 以百分之一秒衡量的 CPU 总时间              |
| SESSIONS_PER_USER         | 允许每个用户名打开的并发会话数                |
| CONNECT_TIME              | 以分钟衡量的连接持续的时间                  |
| IDLE_TIME                 | 以分钟衡量的不活动时间段                   |
| LOGICAL_READS_PER_SESSION | 数据块数（物理读取数和逻辑读取数）              |
| PRIVATE_SGA               | 以字节衡量的 SGA 中的专用空间<br>（仅限共享服务器） |

ORACLE

14-23

Copyright © Oracle Corporation, 2001. All rights reserved.

### 在会话级设置资源限制

#### 原则：

可以在会话级、调用级或同时在这两个级别上执行配置文件限制。每一个连接都要执行会话级限制。

如果超出了会话级限制，则：

- 返回错误消息；例如：  
ORA-02391：超过了 SESSION\_PER\_USER 的并发限制
- Oracle 服务器断开与用户的连接。

#### 原则：

- 仅计算服务器进程的 IDLE\_TIME。不考虑应用程序活动。长时间运行的查询和其它操作不影响 IDLE\_TIME 限制。
- LOGICAL\_READS\_PER\_SESSION 是对从内存和磁盘的读取总数的限制。它可以确保 I/O 密集型语句不会占据内存及冻结磁盘。
- 只有运行共享服务器体系结构时，PRIVATE\_SGA 才适用，可以用 MB 或 KB 为单位指定该参数。

注：Oracle9i 数据库管理基础 II 课程中详细介绍了共享服务器体系结构。

## 在调用级设置资源限制

| 资源                     | 说明                      |
|------------------------|-------------------------|
| CPU_PER_CALL           | 每次调用占用的 CPU 时间，单位为百分之一秒 |
| LOGICAL_READS_PER_CALL | 每次调用可读取的数据块数            |

ORACLE

14-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 在调用级设置资源限制

对于执行 SQL 语句时执行的每次调用，均适用调用级限制。

如果超出了调用级限制，则：

- 暂停处理语句
- 语句被退回
- 前面所有的语句保持原样
- 用户会话仍保持连接

## 创建配置文件： 资源限制

```
CREATE PROFILE developer_prof LIMIT
SESSIONS_PER_USER 2
CPU_PER_SESSION 10000
IDLE_TIME 60
CONNECT_TIME 480;
```

ORACLE

14-25

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建配置文件：资源限制

使用下列 CREATE PROFILE 命令创建配置文件：

```
CREATE PROFILE profile LIMIT
[SESSIONS_PER_USER max_value]
[CPU_PER_SESSION max_value]
[CPU_PER_CALL max_value]
[CONNECT_TIME max_value]
[IDLE_TIME max_value]
[LOGICAL_READS_PER_SESSION max_value]
[LOGICAL_READS_PER_CALL max_value]
[COMPOSITE_LIMIT max_value]
[PRIVATE_SGA max_bytes]
```

其中：

profile: 是配置文件的名称

max\_value: 是一个整数、UNLIMITED 或 DEFAULT

max\_bytes: 是一个整数，后面可以跟 KB 或 MB、UNLIMITED 或者 DEFAULT

## 创建配置文件：资源限制（续）

UNLIMITED：表示分配了此配置文件的用户可以不受限制地使用此资源

DEFAULT：表示此配置文件要遵从 DEFAULT 配置文件中对此资源的限制

COMPOSITE\_LIMIT：对一个以服务单元表示的会话限制其资源总成本；

Oracle 计算的资源成本是以下各项的总和：

- CPU\_PER\_SESSION
- CONNECT\_TIME
- LOGICAL\_READS\_PER\_SESSION
- PRIVATE\_SGA

数据字典视图 RESOURCE\_COST 可以提供为不同资源指定的资源限制。

**注：**有关如何为每个会话资源 ALTER RESOURCE COST 命令指定加权值的信息，请参阅 *Oracle9i SQL Reference* 文档。



## 创建配置文件：资源限制

使用 **Oracle Enterprise Management** 设置资源限制

从“OEM 控制台”(OEM Console):

- 导航到“安全性”(Security) > “配置文件”(Profiles)。
- 单击鼠标右键，从弹出的菜单中选择“创建”(Create)。
- 在“常规”(General) 页中输入资源参数。
- 单击“创建”(Create)。

Create Profile - sys@DBA01\_STC-SUN01.US.ORACLE.COM

General Password

Name: DEVELOPER\_PROF

Details

|               |      |          |
|---------------|------|----------|
| CPU/Session:  | 1000 | Sec./100 |
| CPU/Call:     | 1000 | Sec./100 |
| Connect Time: | 120  | Minutes  |
| Idle Time:    | 60   | Minutes  |

Database Services

|                      |         |               |
|----------------------|---------|---------------|
| Concurrent Sessions: | 2       | Per User      |
| Reads/Session:       | 5000    | Blocks        |
| Reads/Call:          | 5000    | Blocks        |
| Private SGA:         | 16      | KBytes        |
| Composite Limit:     | 5000000 | Service Units |

Create Cancel Show SQL Help

## 使用“数据库资源管理器” 管理资源

- 加强 Oracle 服务器对资源管理决策的控制
- “数据库资源管理器”的元素
  - 资源使用者组
  - 资源计划
  - 资源分配方法
  - 资源计划指令
- 使用 DBMS\_RESOURCE\_MANAGER 程序包可创建和维护元素
- 需要 ADMINISTER\_RESOURCE\_MANAGER 权限

ORACLE

14-28

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用“数据库资源管理器”管理资源

“数据库资源管理器”旨在加强 Oracle 服务器对资源管理决策的控制，进而解决因操作系统管理不当而引发的各种问题。

#### “数据库资源管理器”的元素

资源使用者组：用户或会话组，根据资源处理要求进行分组。

资源计划：包含指定如何向资源使用者组分配资源的指令

资源分配方法：供“数据库资源管理器”在分配特定资源时使用

资源计划指令：管理员可以借助它使资源使用者组与特定计划关联，并在资源使用者组中分配资源

#### 管理“数据库资源管理器”

只有拥有系统权限 ADMINISTER\_RESOURCE\_MANAGER 时，才能管理“数据库资源管理器”(DBMS\_RESOURCE\_MANAGER)。通常，具有这种权限的 DBA 还拥有 ADMIN 选项，作为 DBA 职责的一部分。

## 使用“数据库资源管理器” 管理资源

- 资源计划指定了属于这个计划的资源使用者组。
- 资源计划包含有关资源将在使用者组中如何分配的指令。

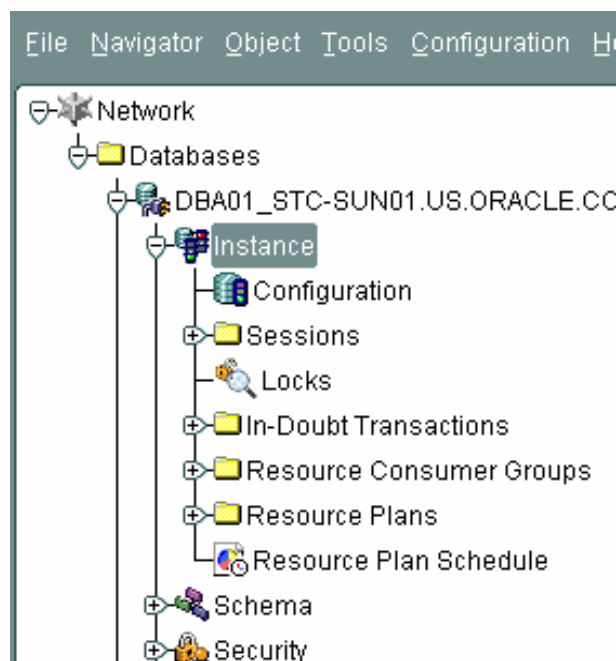
ORACLE

## 使用“数据库资源管理器”管理资源

### 使用 Oracle Enterprise Manager 设置资源管理器

从“OEM 控制台”(OEM Console):

- 导航到“数据库”(Databases) > “例程”(Instance)。
- 选择要创建或修改的“资源使用者组”(Resource Consumer Groups)。
- 选择要创建或修改的“资源计划”(Resource Plans)。
- 选择要创建或修改的“资源计划调度”(Resource Plans Schedule)。



# 资源计划指令

“数据库资源管理器”提供了以下几种分配资源的方法：

- CPU 方法
- 活动会话池和排队
- 并行度限制
- 自动切换使用者组
- 估算的最长执行时间
- 还原限额

ORACLE

14-31

Copyright © Oracle Corporation, 2001. All rights reserved.

## 资源计划指令

**CPU 方法：**用于指定 CPU 资源在使用者组中将如何分配

**活动会话池和排队：**您可以控制使用者组内所允许的并发活动会话的最大数目。这个最大数目同时也指定了活动会话池的大小。如果有会话因这个池已满而无法启动，则该会话将排入某个队列。当某个活动会话完成后，下一个要执行的就是这个队列中的第一个会话。另外，可以定义超时时段，让队列中的作业超时，从而中止该作业，同时显示错误消息。

**并行度限制：**为使用者组中的任一操作指定并行度限制

**自动切换使用者组：**允许您通过特定标准控制资源。如果不符合标准，就会使会话自动切换到另一使用者组。用于确定切换的标准如下：

切换组：正要切换到的组

切换时间：以秒为单位的切换时间

切换估计时间：估计完成操作所需的时间，用于在切换操作开始前决定是否进行切换

## 资源计划指令（续）

**最长估计执行时间：**预先估计操作的执行时间。通过设置资源计划指令参数 `MAX_ESTIMATED_EXEC_TIME`，DBA 可以定义在任一时间执行任一操作所需的最长估计执行时间。如果操作的估计执行时间大于定义的 `MAX_ESTIMATED_EXEC_TIME`，将不会执行操作，从而避免处理占用过多系统资源的特大型作业。

**还原池：**可以为每个使用者组指定一个还原池，用来控制使用者组生成的还原数据总量。当使用者组超过这个限制时，便会终止当前正在生成重做日志的 **DML** 语句。通过名为 `UNDO_POOL` 的资源计划指令参数可以定义还原池。

**注：***Oracle9i 性能优化* 课程对“数据库资源管理器”进行了详细介绍。

## 获取口令和资源 限制信息

可以通过查询以下视图来获取有关口令和资源限制的信息：

- **DBA\_USERS**
- **DBA\_PROFILES**

ORACLE

14-33

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取口令和资源限制信息

使用 DBA\_USERS 可获得有关帐户状态的信息。

```
SQL> SELECT username, password, account_status,
2 FROM dba_users;
```

```
USERNAME PASSWORD ACCOUNT_STATUS
```

```

SYS 8A8F025737A9097A OPEN
SYSTEM D4DF7931AB130E37 OPEN
OUTLN 4A3BA55E08595C81 OPEN
DBSNMP E066D214D5421CCC OPEN
HR BB69FBB77CFA6B9A OPEN
OE 957C7EF29CC223FC LOCKED
```

## 查看配置文件信息

查询 DBA\_PROFILES 视图以显示口令配置文件信息：

```
SQL> SELECT * FROM dba_profiles
 2 WHERE resource_type='PASSWORD'
 3 AND profile='GRACE_5';
```

| PROFILE | RESOURCE_NAM             | RESOURCE | LIMIT     |
|---------|--------------------------|----------|-----------|
| GRACE_5 | FAILED_LOGIN_ATTEMPTS    | PASSWORD | 3         |
| GRACE_5 | PASSWORD_LIFE_TIME       | PASSWORD | 30        |
| GRACE_5 | PASSWORD_REUSE_TIME      | PASSWORD | 30        |
| GRACE_5 | PASSWORD_REUSE_MAX       | PASSWORD | UNLIMITED |
| GRACE_5 | PASSWORD_VERIFY_FUNCTION | PASSWORD | DEFAULT   |
| GRACE_5 | PASSWORD_LOCK_TIME       | PASSWORD | UNLIMITED |
| GRACE_5 | PASSWORD_GRACE_TIME      | PASSWORD | 5         |



## 小结

在这一课中，您应该能够掌握：

- 管理口令
- 管理配置文件

ORACLE

## 练习 14

此练习涉及以下主题：

- 启用口令管理
- 定义配置文件并将其分配给用户
- 禁用口令管理

ORACLE

14-36

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 14

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成练习。

## 练习 14：管理口令安全性和资源

1 运行 lab14\_01.sql 脚本来创建用户 Jeff。运行 @\$ORACLE\_HOME/rdbms/admin/utlpwdmg.sql 脚本来启用口令管理。

2 尝试将用户 Jeff 的口令更改为 Jeff。结果如何？

3 尝试按照下面的口令管理格式更改 Jeff 的口令。

**提示：**口令至少应当包含一个数字、一个字符和一个标点符号。

4 改变 DEFAULT 配置文件，确保以下设置适用于分配有 DEFAULT 配置文件的所有用户：

- 经过两次登录尝试后，应锁定帐户。
- 口令应在 30 天后失效。
- 至少在一分钟内不应重新使用同一口令。
- 帐户应有 5 天的宽限期来更改失效的口令。
- 确保已按照列出的要求执行了操作。

**提示：**

使用 ALTER PROFILE 命令更改缺省配置文件限制。

查询数据字典视图 DBA\_PROFILES 以验证结果。

5 以用户 Jeff 的身份用无效口令登录。试着这样做两次，然后再使用正确的口令登录。结果如何？

6 使用 DBA\_USERS 数据字典视图验证用户 Jeff 是否已锁定。解除对用户 Jeff 的帐户锁定。对用户 Jeff 解除锁定后，以 Jeff 身份连接。

**提示：**执行 ALTER USER 命令可解除对帐户的锁定。

7 对 DEFAULT 配置文件禁用口令检查。

**提示：**执行 ALTER PROFILE 命令可禁用口令检查。

8 使用无效口令登录到用户 Jeff 的帐户。试着这样做两次，然后再使用正确的口令登录。结果如何？



# 15

管理用户

ORACLE®

Copyright © Oracle Corporation, 2001. All rights reserved.

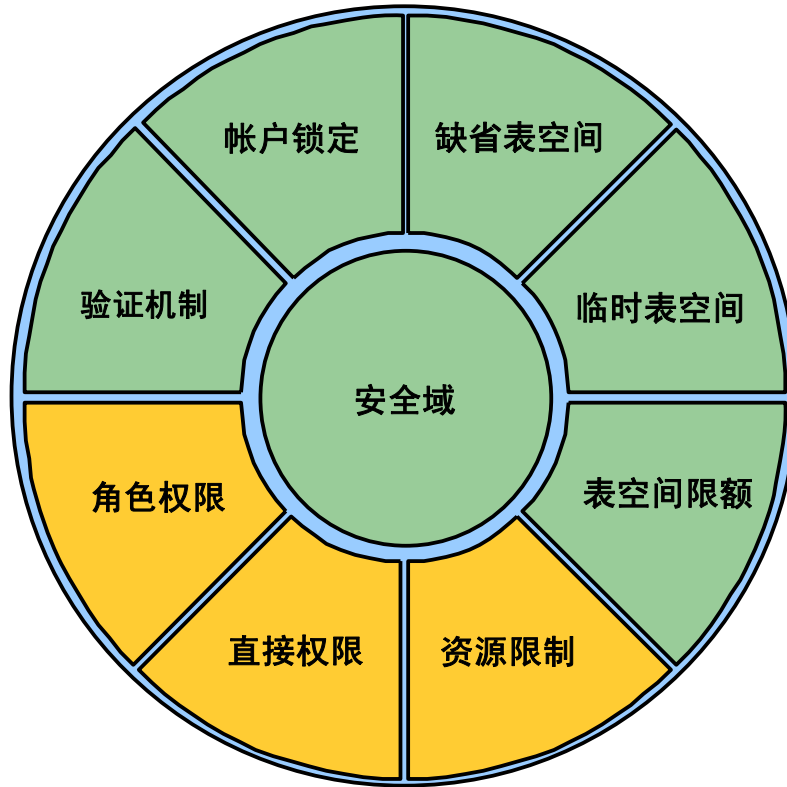
# 目标

完成这一课的学习后，您应该能达到下列目标：

- 创建新的数据库用户
- 改变和删除现有数据库用户
- 监视有关现有用户的信息

ORACLE<sup>®</sup>

## 用户与安全性



ORACLE

15-3

Copyright © Oracle Corporation, 2001. All rights reserved.

### 用户与安全性

#### 安全域：

数据库管理员负责定义可以访问数据库的用户名。安全域定义应用于用户的设置。

#### 验证机制：

可通过以下方式之一对需要访问数据库的用户进行验证：

- 数据字典
- 操作系统
- 网络

验证方法是在数据库中定义用户时指定的，之后可随时改变。本课仅讨论通过数据库和操作系统进行验证的方式。

**注：**有关通过角色进行操作系统验证的详细信息，请参考“Oracle 服务器入门”一课。

有关通过网络验证的方式，请参考 *Oracle9i 数据库管理基础 II* 这部分的课程。

## 用户与安全性（续）

### 表空间限额：

表空间限额控制分配给数据库中表空间用户的物理存储空间的大小。

### 缺省表空间：

如果用户在创建段时不明确指定表空间，那么缺省表空间会定义用户所创建段的存储位置。

### 临时表空间：

如果用户执行的操作要求将排序数据写入磁盘，那么临时表空间会定义 Oracle 服务器分配哪些区。

### 帐户锁定：

可锁定帐户以防止用户登录数据库。可以设置自动锁定，也可由数据库管理员手动锁定帐户或解除锁定。

### 资源限制：

可以对某些资源的使用加以限制，如：CPU 时间、逻辑输入/输出 (I/O) 以及用户打开的会话数目。

### 直接权限：

权限用于控制用户能在数据库中执行的操作。

### 角色权限：

可通过使用角色间接授予用户权限。

**注：**有关角色权限的信息，请参考“管理权限”与“管理角色”这两课。

本课讨论通过适当的验证机制定义用户、限制系统中用户对空间的使用和手动控制帐户锁定。



# 数据库方案

- 方案是一个特定的对象集合。
- 创建用户时，就会创建一个对应的方案。
- 一个用户只能与一个方案关联。
- 用户名和方案经常交换使用。

## 方案对象

表

触发器

约束

索引

视图

序列

存储程序单元

同义词

用户定义的数据类型

数据库链接

## 数据库方案

方案是与某个用户关联的表、视图、簇、过程和程序包等对象的特定集合。创建数据库用户时，就会相应地为该用户创建同名的方案。用户只能与同名的方案关联，因此用户名和方案经常互用。

该幻灯片显示用户能在 Oracle 数据库中拥有的某些对象。

## 创建用户操作的核对清单

- 确定用户需要在其中存储对象的表空间。
- 确定每个表空间的限额。
- 指定一个缺省表空间与临时表空间。
- 创建用户。
- 向用户授予权限与角色。

ORACLE

## 创建新用户： 数据库验证

设置初始口令：

```
CREATE USER aaron
IDENTIFIED BY soccer
DEFAULT TABLESPACE data
DEFAULT TEMPORARY TABLESPACE temp
QUOTA 15M ON data
QUOTA 10M ON users
PASSWORD EXPIRE;
```

ORACLE

15-7

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建新用户：数据库验证

语法：

使用以下命令创建新用户：

```
CREATE USER user
IDENTIFIED {BY password | EXTERNALLY}
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {integer [K | M] | UNLIMITED } ON tablespace
[QUOTA {integer [K | M] | UNLIMITED } ON tablespace
...]
[PASSWORD EXPIRE]
[ACCOUNT { LOCK | UNLOCK }]
[PROFILE { profile | DEFAULT }]
```

## 创建新用户：数据库验证（续）

语法（续）：

其中：

User： 是用户名

BY password： 指定用户在登录时需通过数据库验证，还必须提供口令

EXTERNALLY： 指定用户需通过操作系统验证

GLOBALLY AS： 指定对用户进行全局验证

DEFAULT 或 TEMPORARY TABLESPACE： 为用户标识缺省或临时表空间

QUOTA： 定义表空间中允许用户拥有对象所具有的最大空间（可将限额定义为整数字节或千字节/兆字节。关键字 UNLIMITED 用于指定用户拥有的对象可使用表空间内的全部可用空间。缺省情况下，用户在任何表空间上都没有限额。）

PASSWORD EXPIRE： 强制用户在使用 SQL\*Plus 登录到数据库时重置口令（该选项仅在用户通过数据库进行验证时有效）。

ACCOUNT LOCK/UNLOCK： 可用于显式锁定或解除锁定用户帐户（UNLOCK 为缺省设置）

PROFILE： 用于控制资源使用和指定用户的口令控制机制。

注：有关创建配置文件的信息，请参考“管理配置文件”一课。

口令验证方法是必需的。如果指定了口令，则 Oracle 服务器将在数据字典中对其进行维护。用户通过服务器进行验证时，可使用 Oracle 服务器提供的口令控制机制。

设置了口令之后，当用户使用 SQL\*Plus 登录时将接收到下列登录消息，同时系统提示用户输入新口令：

ERROR:

ORA-28001: 该帐户已过期

更改 PETER 的口令

旧口令:

新口令:

重新键入新口令:

口令已更改

## 创建新用户：数据库验证

### 使用 Oracle Enterprise Manager 创建新用户

从“OEM 控制台” (OEM Console):

1. 导航到“数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 单击鼠标右键，从弹出的菜单中选择“创建” (Create)。
3. 输入创建用户所需的信息。
4. 单击“创建” (Create)。

Oracle Security Manager 自动将 CONNECT 角色授予使用该工具创建的任何用户。

注：有关 CONNECT 角色的信息，请参考“管理角色”一课。

The screenshot shows the 'Create User' dialog box in Oracle Enterprise Manager. The title bar reads 'Create User - sys@DBA01\_STC-SUN01.US.ORACLE.COM'. The dialog has several tabs: 'General', 'Role', 'System Privileges', 'Object Privileges', 'Consumer Group', 'Quota', and 'Proxy Users'. The 'General' tab is selected. It contains the following fields and options:

- Name:** AARON
- Profile:** DEFAULT
- Authentication:** Password
- Enter Password:** masked with asterisks
- Confirm Password:** masked with asterisks
- ☐ Expire Password Now
- Tablespaces:**
  - Default:** USERS
  - Temporary:** DEFAULTTEMPORARY
- Status:**
  - ☐ Locked
  - ☒ Unlocked

At the bottom of the dialog are four buttons: 'Create', 'Cancel', 'Show SQL', and 'Help'.

## 创建新用户： 操作系统验证

**OS\_AUTHENT\_PREFIX** 初始化参数用于指定用户名的格式。  
缺省为 OPS\$。

```
CREATE USER aaron
IDENTIFIED EXTERNALLY
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE temp
QUOTA 15m ON data
PASSWORD EXPIRE;
```

ORACLE

15-10

Copyright © Oracle Corporation, 2001. All rights reserved.

### 创建新用户：操作系统验证

#### 操作系统验证：

用 CREATE USER 命令的 IDENTIFIED EXTERNALLY 子句指定用户必须通过操作系统进行验证。当用户直接登录到运行 Oracle 服务器的计算机上时，该选项通常很有用。

#### 操作系统验证的用户名：

OS\_AUTHENT\_PREFIX 初始化参数用来指定操作系统验证的用户名的格式。该参数的缺省值为 OPS\$，以便与 Oracle 服务器的早期版本向后兼容。要将前缀设置为 NULL 值，请将该初始化参数指定为：

```
OS_AUTHENT_PREFIX = ''
```

幻灯片中的示例显示如何在数据库内定义用户 aaron。此处指定允许操作系统用户 aaron 无需经过 Oracle 服务器验证即可访问数据库。因此，若要使用 SQL\*Plus 登录到系统，则 UNIX 用户 aaron 必须从该操作系统输入下列命令：

```
$ sqlplus /
```

## 创建新用户：操作系统验证（续）

操作系统验证的用户名：

注：

- 使用 OS\_AUTHENT\_PREFIX=OPS\$ 时：提供了灵活的用户验证方式，既可通过操作系统进行验证，也可通过 Oracle 服务器进行验证。在这种情况下，DBA 可通过输入下列格式的命令创建用户：

```
CREATE USER ops$user
IDENTIFIED BY password ...
```

- 登录到运行 Oracle 服务器的计算机上的用户无需提供口令。如果用户从远程客户机连接，则可提供口令以实现连接。
- 设置另一个初始化参数 REMOTE\_OS\_AUTHENT=TRUE，指定用户可通过远程操作系统进行验证。缺省值 FALSE 表示用户只能通过运行 Oracle 服务器的计算机进行验证。应小心使用该参数，因为可能存在安全隐患。
- 如果数据库中的某些用户通过操作系统进行验证，则更改 OS\_AUTHENT\_PREFIX 可防止这些用户登录到数据库。

## 更改用户的表空间限额

- 在下列情况下需要修改用户的表空间限额：
  - 当用户所拥有表的增长速度异常快时。
  - 当应用程序得到增强而要求额外的表或索引时。
  - 当重新安排对象并将其置入不同的表空间时。
- 修改用户的表空间限额：

```
ALTER USER aaron
QUOTA 0 ON USERS;
```

ORACLE

15-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 更改用户的表空间限额

使用下列命令修改表空间限额或重新分配表空间：

```
ALTER USER user
[DEFAULT TABLESPACE tablespace]
[TEMPORARY TABLESPACE tablespace]
[QUOTA {integer [K | M] | UNLIMITED } ON tablespace
[QUOTA {integer [K | M] | UNLIMITED } ON tablespace]
...]
```

如果分配的限额为 0，用户拥有的对象仍保留在撤消的表空间内，但不能给它们分配新的空间。例如，如果表空间 USERS 内有一个 10 MB 的表，若将该表空间的限额改为 0，则不能再为该表分配新区。

任何未更改的选项保持不变。

**注：**对于 UNLIMITED TABLESPACE 权限应格外注意，因为它优先于限额设置。



## 更改用户的表空间限额

### 使用 Oracle Enterprise Manager 修改用户的表空间限额

1. 导航到“数据库”(Databases) > “安全性”(Security) > “用户”(Users)。
2. 选择用户。
3. 单击鼠标右键，从弹出的菜单中选择“查看/编辑详细资料”(View/Edit Details)。
4. 在“限额”(Quota) 页中输入限额的大小。
5. 单击“应用”(Apply)。

| Tablespace       | Quota Size |
|------------------|------------|
| DATA01           | 5 K        |
| DATA02           | 5 K        |
| DATA03           | 5 K        |
| DEFAULTTEMPORARY | <none>     |
| INDEX01          | <none>     |
| INDX             | <none>     |
| MARIETEMP        | <none>     |
| SAMPLE           | <none>     |
| SYSTEM           | <none>     |
| TEMP             | <none>     |
| TESTING_TBS      | <none>     |
| UNDO02           | <none>     |
| USERS            | <none>     |

☒ None ☐ Unlimited ☐ Value  K Bytes

Create Cancel Show SQL Help

## 删除用户

- 如果方案中含有对象，请使用 CASCADE 子句删除该方案中的所有对象。

```
DROP USER aaron;
```

- 不能删除当前与 Oracle 服务器连接的用户。

```
DROP USER aaron CASCADE;
```

ORACLE

15-14

Copyright © Oracle Corporation, 2001. All rights reserved.

### 删除用户

```
DROP USER user [CASCADE]
```

#### 原则：

- 在删除用户前，CASCADE 选项将删除方案中的所有对象。如果方案中包含任何对象，则必须指定该选项。
- 不能删除当前与 Oracle 服务器连接的用户。

## 删除用户

### 使用 Oracle Enterprise Manager 删除用户

从“OEM 控制台”(OEM Console):

1. 导航到“数据库”(Databases) > “安全性”(Security) > “用户”(Users)。
2. 选择用户。
3. 单击鼠标右键，从弹出的菜单中选择“删除”(Remove)。
4. 选择“是”(Yes) 确认删除。
5. 单击“应用”(Apply)。

## 获取用户信息

可以通过查询以下视图来获取有关用户的信息：

- **DBA\_USERS**
- **DBA\_TS\_QUOTAS**

ORACLE

15-16

Copyright © Oracle Corporation, 2001. All rights reserved.

### 获取用户信息

使用以下查询查找用于所有用户的 default\_tablespace。

```
SQL> SELECT username, default_tablespace
2 FROM dba_users;
```

| USERNAME | DEFAULT_TABLESPACE |
|----------|--------------------|
| -----    | -----              |
| SYS      | SYSTEM             |
| SYSTEM   | SYSTEM             |
| OUTLN    | SYSTEM             |
| DBSNMP   | SYSTEM             |
| HR       | SAMPLE             |
| OE       | SAMPLE             |

## 小结

在这一课中，您应该能够掌握：

- 通过指定适当的口令机制来创建用户
- 控制用户对空间的使用

ORACLE

## 练习 15 概览

此练习涉及以下主题：

- 创建用户
- 显示有关用户的数据字典信息
- 删除用户限额

ORACLE

15-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### 练习 15 概览

注：可以使用 SQL\*Plus 或使用 Oracle Enterprise Manager 和 SQL\*Plus Worksheet 完成此处的练习。

## 练习 15：管理用户

- 1 创建用户 Bob，口令为 CRUSADER。确保 Bob 创建的所有对象和临时段都不是在系统表空间中创建的。此外，还应确保 Bob 可以登录，并可以在 USERS 和 INDX 表空间中创建最多占用 1 兆字节空间的对象。使用 lab15\_01.sql 脚本授予 Bob 创建会话的能力。

**提示：** 请为 Bob 分配缺省表空间 USERS 和临时表空间 TEMP。

- 2 创建用户 Emi，口令为 MARY。确保由 Emi 创建的所有对象和排序段都不是在系统表空间中创建的。
- 3 从数据字典中显示有关 Bob 和 Emi 的信息。

**提示：** 可以通过查询 DBA\_USERS 获得相关信息。

- 4 从数据字典中显示 Bob 可以在表空间中使用的空间量信息。

**提示：** 可以通过查询 DBA\_TS\_QUOTAS 获得相关信息。

- 5 **a** 以用户 Bob 的身份更改其临时表空间。结果如何？

**b** 以用户 Bob 的身份将其口令更改为 SAM。

- 6 以用户 SYSTEM 的身份删除 Bob 的缺省表空间限额。

- 7 从数据库中删除 Emi 的帐户。

- 8 假设 Bob 忘记了他的口令。为他指定一个口令 OLINK，并要求他下次登录时对口令做出相应更改。

