Oracle9*i* 数据库管理基础 I

第 2 册 • 学生指南

D11321CN11
产品版 1.1
2002 年 3 月
D34489

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

目录

前言

I 简介
课程目标 I-2
Oracle9*i* 企业版 I-3
数据库管理员的任务 I-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 Oracle9*i* 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 9*i* 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
Oracle9*i* Backup and Recovery Concepts	A90133-02
Oracle9*i* Database Administrator's Guide	A90117-01
Oracle9*i* Database Concepts	A88856-02
Oracle9*i* Database Error Messages	A90202-02
Oracle9*i* Database New Features	A90120-02
Oracle9*i* Database Reference	A90190-02
Oracle9*i* Database Utilities	A90192-01
Oracle9*i* Enterprise Manager Administrator’s Guide	A88767-02
Oracle9*i* Enterprise Manager Concepts Guide	A88770-01
Oracle9*i* Enterprise Manager Configuration Guide	A88769-01
Oracle9*i* Net Services Administrator’s Guide	A90154-01
Oracle9*i* Net Services Reference Guide	A90155-01
Oracle9*i* Recovery Manager Reference	A90136-02
Oracle9*i* Recovery Manager User’s Guide	A90135-01
Oracle9i Database Reference	A90190-02
Oracle9*i* SQL Reference	A90125-01
Oracle9*i* 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***算法***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：`ebug.set ('I",300);` 目录：`bin` (DOS)、`$FMHOME` (UNIX) 文件名：找到 `init.ora` 文件。 口令：User `tiger` 作为您的口令。 路径名：打开 `c:\my_docs\projects` URL：转到 `http://www.oracle.com` 用户输入：输入 `300` 用户名：以 `scott` 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（*但* Oracle Payables *除外*）
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 *Oracle7 Server SQL Language Reference Manual*。 输入 `user_id@us.oracle.com`，其中 *user_id* 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 SELECT 命令看存储在 EMP 表的 LAST_NAME 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` *`role`*
大写	SQL 命令和函数	`SELECT userid` `FROM emp;`

16

管理权限

目标

完成这一课的学习后，您应该能达到下列目标：

- 标识系统与对象权限
- 授予和撤消权限

ORACLE

管理权限

有两种 Oracle 用户权限：

- **系统权限：具备该权限的用户可在数据库中执行特定操作**
- **对象权限：具备该权限的用户可访问并操纵特定对象**

ORACLE

权限

权限是指执行特定类型的 SQL 语句或访问另一个用户的对象的权利。包括以下权利：

- 连接到数据库
- 创建表
- 从另一用户的表中选择行
- 执行另一用户的已存储过程

系统权限：

每一系统权限都允许用户执行某一特定的数据库操作或某类数据库操作，例如，创建表空间的权限就是一种系统权限。

对象权限：

每一对象权限都允许用户对特定对象（如表、视图、序列、过程、函数或程序包）执行特定的操作。

DBA 的权限控制包括：

- 为用户提供执行某种操作的权限
- 授予和撤消执行系统功能的权限
- 将权限直接授予用户或角色
- 将权限授予所有用户 (PUBLIC)

系统权限

- 有 100 多种不同的系统权限。
- 权限中的关键字 ANY 表示用户在任何方案中都具备这种权限。
- GRANT 命令为一个用户或一组用户添加一项权限。
- REVOKE 命令则用来删除权限。

ORACLE

系统权限

系统权限可分为以下几类：

- 允许执行系统范围操作的权限；如 CREATE SESSION，CREATE TABLESPACE
- 允许管理用户自己方案中的对象的权限；如 CREATE TABLE
- 允许管理任何方案中的对象的权限；如 CREATE ANY TABLE

可使用 DDL 命令 GRANT 和 REVOKE 控制权限，这两个命令为用户或角色添加和撤消系统权限。有关角色的详细信息，请参考 “管理角色” 一课。

系统权限：示例

类别	示例
索引(INDEX)	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
表(TABLE)	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
会话(SESSION)	CREATE SESSION ALTER SESSION RESTRICTED SESSION
表空间(TABLESPACE)	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

ORACLE

系统权限：示例

- 没有 CREATE INDEX 权限。
- CREATE TABLE 包括 CREATE INDEX 和 ANALYZE 命令。用户必须有表空间的限额，或必须被授予 UNLIMITED TABLESPACE 权限。
- 诸如 CREATE TABLE、CREATE PROCEDURE 或 CREATE CLUSTER 等权限包括删除这些对象的权限。
- 无法将 UNLIMITED TABLESPACE 授予角色。
- DROP ANY TABLE 权限是截断另一方案中的表所必需的。

授予系统权限

- 使用 GRANT 命令可授予系统权限。
- 被授予者可通过 ADMIN 选项进一步授予系统权限。

```
GRANT CREATE SESSION TO emi;
```

```
GRANT CREATE SESSION TO emi WITH ADMIN OPTION;
```

ORACLE

授予系统权限

使用 SQL 语句 GRANT 为用户授予系统权限。

被授予者可通过 ADMIN 选项进一步为其他用户授予系统权限。使用 ADMIN 选项授予系统权限时应小心。这样的权限通常只限于安全管理员使用，很少授予其他用户。

```
GRANT {system_privilege|role}
     [, {system_privilege|role} ]...
    TO {user|role|PUBLIC}
     [, {user|role|PUBLIC} ]...
    [WITH ADMIN OPTION]
```

其中：

system_privilege：指定要授予的系统权限

Role：指定要授予的角色名

PUBLIC：将系统权限授予所有用户

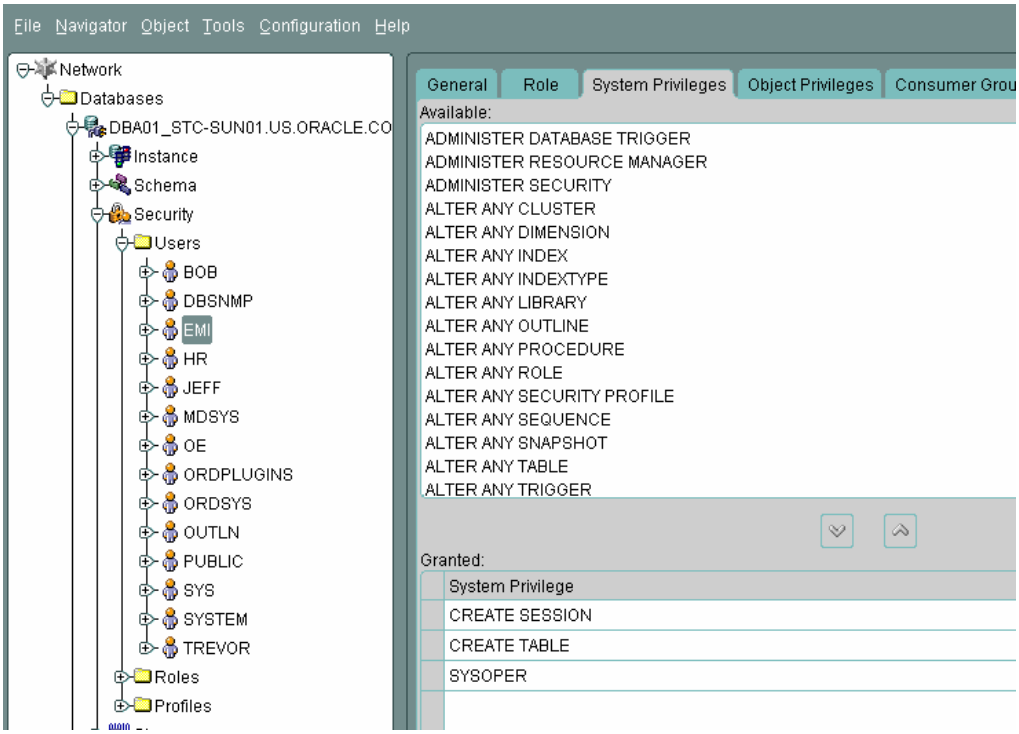
WITH ADMIN OPTION：允许被授予者进一步为其他用户或角色授予权限或角色

授予系统权限

使用 Oracle Enterprise Manger 授予系统权限

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 选择要为其授予权限的用户。
5. 单击控制台详细资料一侧上的 “系统权限” (System Privileges) 选项卡。
6. 选择要授予的系统权限。选中 “管理选项” (Admin Option) 复选框（可选操作）。
7. 单击 “应用” (Apply)。



SYSDBA 和 SYSOPER 权限

类别	示例
SYSOPER	STARTUP SHUTDOWN ALTER DATABASE OPEN | MOUNT ALTER DATABASE BACKUP CONTROLFILE TO RECOVER DATABASE ALTER DATABASE ARCHIVELOG RESTRICTED SESSION
SYSDBA	SYSOPER PRIVILEGES WITH ADMIN OPTION CREATE DATABASE ALTER TABLESPACE BEGIN/END BACKUP RESTRICTED SESSION RECOVER DATABASE UNTIL

ORACLE

SYSDBA 和 SYSOPER 权限

只有数据库管理员可以使用管理员权限与数据库连接。以 SYSDBA 身份连接可以授予用户不受限制的权限，以便对数据库或数据库中的对象执行任何操作。

系统权限限制

- **O7_DICTIONARY_ACCESSIBILITY 参数**
- **控制有关 SYSTEM 权限的限制**
- **如果设置为 TRUE，允许访问 SYS 方案中的对象**
- **缺省值为 FALSE：可确保允许访问任何方案的系统权限不允许访问 SYS 方案**

系统权限限制

Oracle9*i* 中的字典保护机制可防止未经授权的用户访问字典对象。

只有角色 SYSDBA 和 SYSOPER 可以访问字典对象。允许访问其他方案中的对象的系统权限并不授予您对字典对象的访问权限。例如，SELECT ANY TABLE 权限允许您访问其它方案中的视图和表，但不允许您选择字典对象（基表、视图、程序包和同义词）。

如果该参数设置为 TRUE, 则允许访问 SYS 方案中的对象（Oracle7 行为）。如果该参数设置为 FALSE，则允许访问其它方案中的对象的 SYSTEM 权限不允许访问字典方案中的对象。

例如，如果 O7_DICTIONARY_ACCESSIBILITY=FALSE，则 SELECT ANY TABLE 语句将允许访问除 SYS 方案外的任何方案中的视图或表（例如，不能访问字典）。系统权限 EXECUTE ANY PROCEDURE 将允许访问除 SYS 方案外的任何其它方案中的过程。

撤消系统权限

- 使用 **REVOKE** 命令可撤消用户的系统权限。
- 使用 **ADMIN OPTION** 授予系统权限的用户可以撤消系统权限。
- 只能撤消使用 **GRANT** 命令授予的权限。

```
REVOKE CREATE TABLE FROM emi;
```

ORACLE

撤消系统权限

可以使用 REVOKE SQL 语句撤消系统权限。使用 ADMIN OPTION 授予系统权限的用户可以撤消任何其他数据库用户的权限。撤消者不必是原先授予该权限的那个用户。

```
REVOKE {system_privilege|role}
 [, {system_privilege|role} ]...
FROM {user|role|PUBLIC}
 [, {user|role|PUBLIC} ]...
```

注：

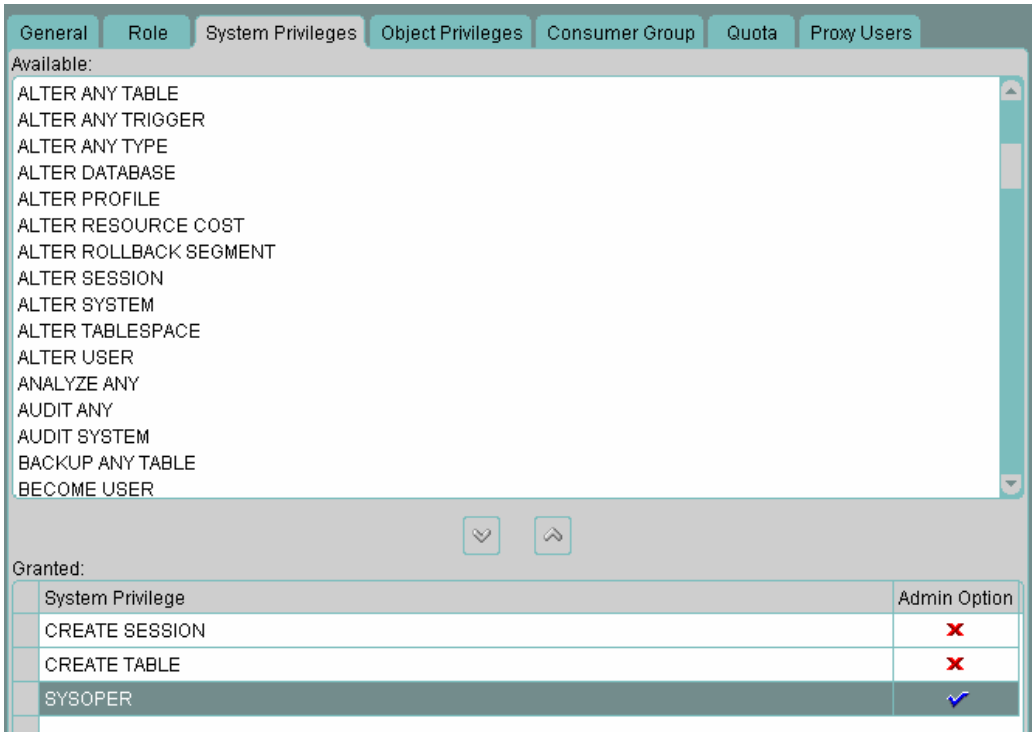
- REVOKE 命令只能撤消使用 GRANT 命令直接授予的权限。
- 撤消系统权限可能对一些相关对象有影响。例如，如果将 SELECT ANY TABLE 授予某用户，而该用户已创建了使用其它方案中的表的过程或视图，则撤消该权限将使这些过程或视图无效。

撤消系统权限

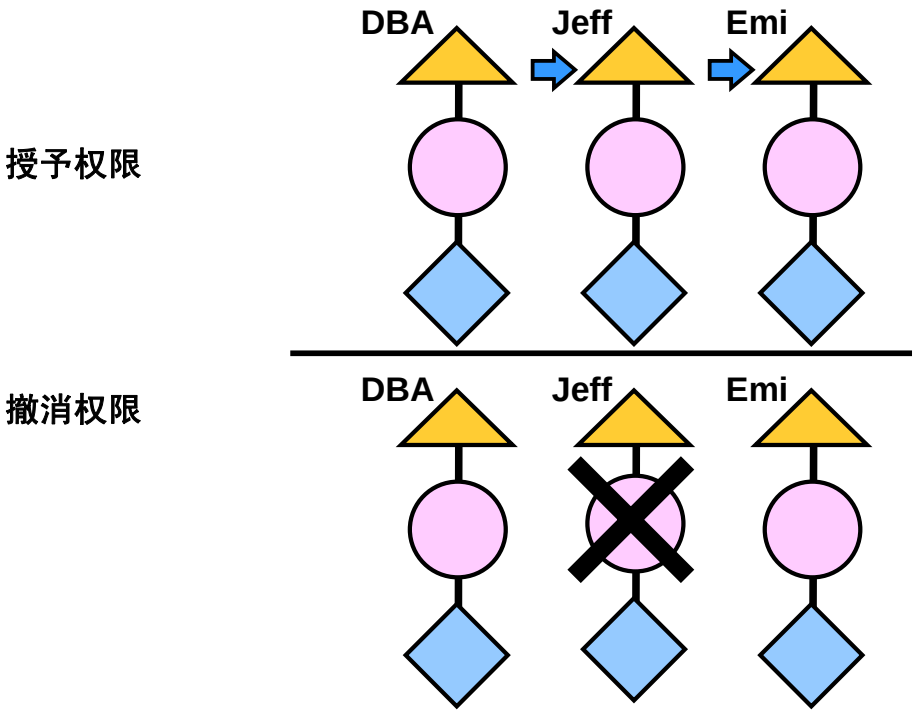
使用 Oracle Enterprise Manager 撤消系统权限

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 选择将撤消其权限的用户。
3. 单击控制台详细资料一侧上的 “系统权限” (System Privileges)页。
4. 选择要撤消的系统权限，然后单击向上箭头。
5. 单击 “应用” (Apply)。



撤消通过 ADMIN OPTION 授予的系统权限



撤消系统权限（续）

撤消系统权限时没有级联效果，这与该系统权限是否使用 ADMIN OPTION 授予无关。

请仔细阅读下列步骤，它们对这种特性进行了说明。

情况

1. DBA 使用 ADMIN OPTION 将系统权限 CREATE TABLE 授予 Jeff。
2. Jeff 创建一个表。
3. Jeff 将系统权限 CREATE TABLE 授予 Emi。
4. Emi 创建一个表。
5. DBA 撤消 Jeff 的 CREATE TABLE 系统权限。

结果

Jeff 的表依然存在，但是，无法创建新表。

Emi 的表依然存在，并且她仍然拥有 CREATE TABLE 系统权限。

对象权限

对象权限	表	视图	序列	过程
ALTER	√	√	√	√
DELETE	√	√		
EXECUTE				√
INDEX	√	√		
INSERT	√	√		
REFERENCES	√			
SELECT	√	√	√	
UPDATE	√	√		

对象权限

对象权限是一种对于特定的表、视图、序列、过程、函数或程序包执行特定操作的一种权限或权利。上表列出了各种对象的权限。需要注意的是适用于序列的权限只有 SELECT 和 ALTER。通过指定可更新列的子集可以对 UPDATE、REFERENCES 和 INSERT 权限加以限制。通过用列的子集创建视图并授予对于该视图的 SELECT 权限，则可对 SELECT 权限加以限制。对于同义词的授权会转换为对于该同义词所引用的基表的授权。

注：该幻灯片未提供有关对象权限的完整列表。

授予对象权限

- 使用 GRANT 命令授予对象权限。
- 授权必须在授予者方案中，或者授予者必须具有 GRANT OPTION 权限。

```
GRANT EXECUTE ON dbms_output TO jeff;
```

```
GRANT UPDATE ON emi.customers TO jeff WITH
GRANT OPTION;
```

ORACLE

授予对象权限

```
GRANT { object_privilege [(column_list)]
     [, object_privilege [(column_list)] ]...
     |ALL [PRIVILEGES]}
ON  [schema.]object
TO  {user|role|PUBLIC}[, {user|role|PUBLIC} ]...
    [WITH GRANT OPTION]
```

其中：

object_privilege：指定要授予的对象权限

column_list：指定表或视图列（只在授予 INSERT、REFERENCES 或 UPDATE 权限时才指定。）

ALL：将所有权限授予已被授予 WITH GRANT OPTION 的对象

ON object：标识将要被授予权限的对象

WITH GRANT OPTION：使被授予者能够将对象权限授予其他用户或角色

授予对象权限（续）

使用 GRANT 语句授予对象权限。

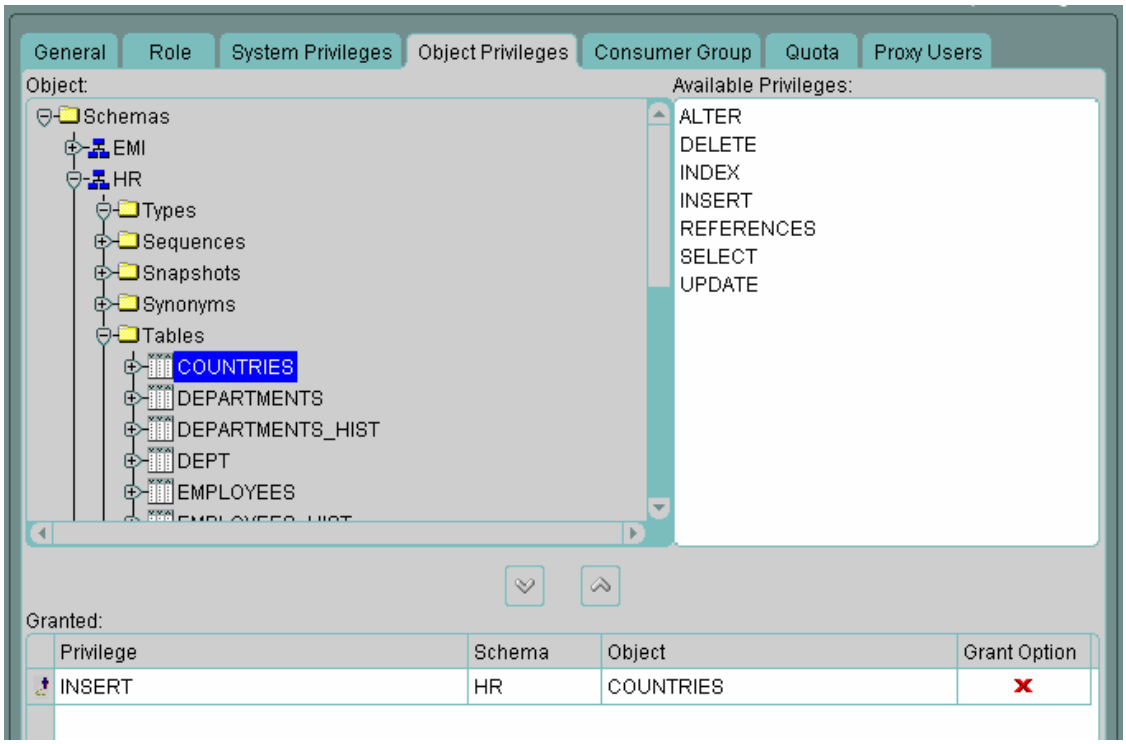
- 要授予权限，对象必须在您的方案中，或者您已通过 GRANT OPTION 被授予权限。
- 缺省情况下，如果拥有某个对象，则自动获得对该对象的所有权限。
- 若有安全方面的考虑，则将您的对象权限授予其他用户时应谨慎。

授予对象权限

使用 Oracle Enterprise Manager 授予对象权限

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 选择要为其授予权限的用户。
3. 单击控制台详细资料一侧上的 “对象权限” (Object Privileges) 选项卡。
4. 展开将为其授予对象权限的方案和对象文件夹。
5. 从 “可用权限” (Available Privileges) 字段选择要授予的权限，然后单击向下箭头。
6. 选中 “授予选项” (Grant Option) 复选框（可选操作）。
7. 单击 “应用” (Apply)。



撤消对象权限

- 使用 **REVOKE** 命令可撤消对象权限。
- 撤消权限的用户必须是将被撤消的对象权限的原始授予者。

```
REVOKE SELECT ON emi.orders FROM jeff;
```

ORACLE

撤消对象权限

REVOKE 语句用来撤消对象权限。要撤消对象权限，撤消者必须是将被撤消的对象权限的原始授予者。

使用下列命令撤消对象权限：

```
REVOKE { object_privilege
         [, object_privilege ]...
         | ALL [PRIVILEGES] }
ON  [schema.]object
FROM {user|role|PUBLIC}
         [, {user|role|PUBLIC} ]...
         [CASCADE CONSTRAINTS]
```

撤消对象权限（续）

其中：

object_privilege：指定将撤消的对象权限

ALL：撤消已授予用户的所有对象权限

ON：标识将撤消其对象权限的对象

FROM：标识将撤消其对象权限的用户或角色

CASCADE CONSTRAINTS：删除撤消使用 REFERENCES 或 ALL 权限定义的任何引用完整性约束

限制：

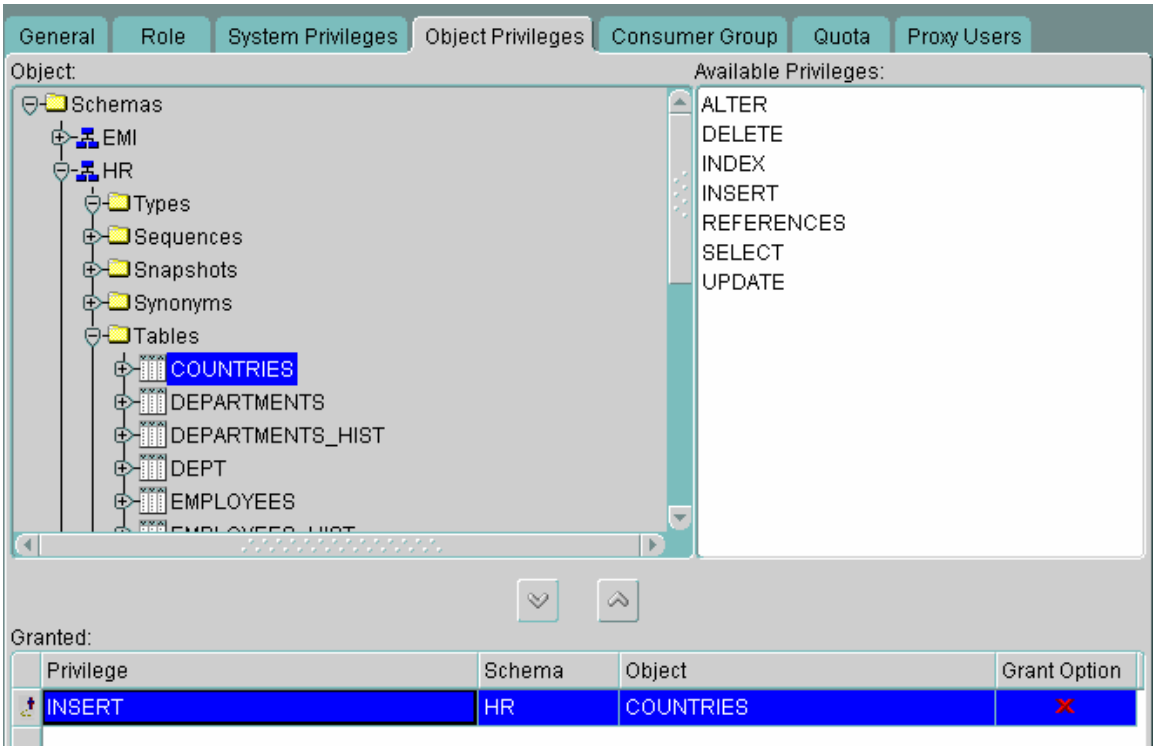
授予者只能对其已经授予权限的用户撤消对象权限。

撤消对象权限

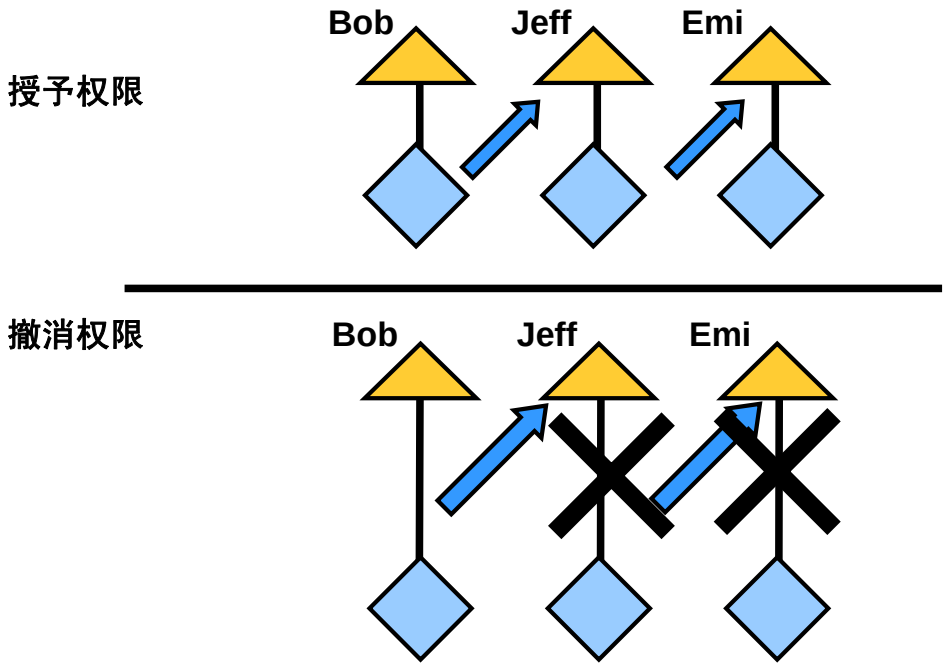
使用 Oracle Enterprise Manager 撤消对象权限

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 选择将撤消其权限的用户。
3. 单击控制台详细资料一侧上的 “对象权限” (Object Privileges) 选项卡。
4. 选择要撤消的对象权限，然后单击向上箭头。
5. 单击 “应用” (Apply)。



撤消对象权限
WITH GRANT OPTION



ORACLE

撤消对象权限（续）

撤消与 DML 操作有关的系统权限时，可看到级联效果。例如，如果为某用户授予 SELECT ANY TABLE 权限，并且该用户已经创建了使用某个表的过程，则必须对该用户的方案中包含的所有过程进行重新编译之后，才能使用这些过程。

如果对象权限是用 WITH GRANT OPTION 授予的，则撤消对象权限也将导致级联效果。

请仔细阅读下列步骤，它们对这种特性进行了说明。

情况：

- 通过 GRANT OPTION 授予 Jeff 对于 EMPLOYEES 的 SELECT 对象权限。
- Jeff 将对于 EMPLOYEES 的 SELECT 权限授予 Emi。
- 之后，撤消 Jeff 的 SELECT 权限。该撤消也对 Emi 产生级联影响。

获取权限信息

可以通过查询以下视图来获取有关权限的信息：

- **DBA_SYS_PRIVS**
- **SESSION_PRIVS**
- **DBA_TAB_PRIVS**
- **DBA_COL_PRIVS**

ORACLE

获取权限信息

DBA_SYS_PRIVS：列出授予用户和角色的系统权限

SESSION_PRIVS：列出用户当前可用的权限

DBA_TAB_PRIVS：列出对于数据库中所有对象的所有授权

DBA_COL_PRIVS：描述数据库中的所有对象-列授权

小结

在这一课中，您应该能够掌握：

- 标识系统与对象权限
- 授予和撤消权限

练习 16 概览

此练习涉及以下主题：

- 创建用户并授予系统权限
- 为用户授予对象权限

ORACLE

练习 16 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 16：管理特权

1 以用户 SYSTEM 的身份创建用户 Emi 并赋予她登录到数据库并在她的方案中创建对象的能力。将她分配到缺省表空间 DATA01 与临时表空间 TEMP。使她在 DATA01 上的限额为 1M。

2 **a** 运行脚本 lab16_02a.sql，以 Emi 的身份连接并创建表 CUSTOMERS1 和 ORDERS1。

b 以 SYSTEM 的身份连接，并将 SYSTEM.CUSTOMERS 中的数据复制到 Emi 的 CUSTOMERS1 表中。确认已插入记录。

c 以用户 SYSTEM 的身份赋予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的能力。结果如何？

3 以 Emi 的身份重新连接，并赋予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的能力。此外，使 Bob 能向其他用户赋予选择权限。以用户 SYSTEM 的身份检查记录这些操作的数据字典视图。

提示：使用 DBA_TAB_PRIVS 进行检查。

4 创建以 diamond1$ 标识的用户 Trevor，使其具有登录到数据库的能力。

5 **a** 以 Bob 的身份使 Trevor 能够访问 Emi 的 CUSTOMERS1 表。

注：由于练习 15 步骤 8 中的操作，将收到 “口令已过期” 消息。为 Bob 指定新口令 aaron$1。

b 以 Emi 的身份撤消 Bob 读取 Emi 的 CUSTOMERS1 表的权限。

c 以 Trevor 的身份查询 Emi 的 CUSTOMERS1 表。结果如何？

6 赋予 Emi 启动和关闭数据库的能力，但不赋予创建新数据库的能力。

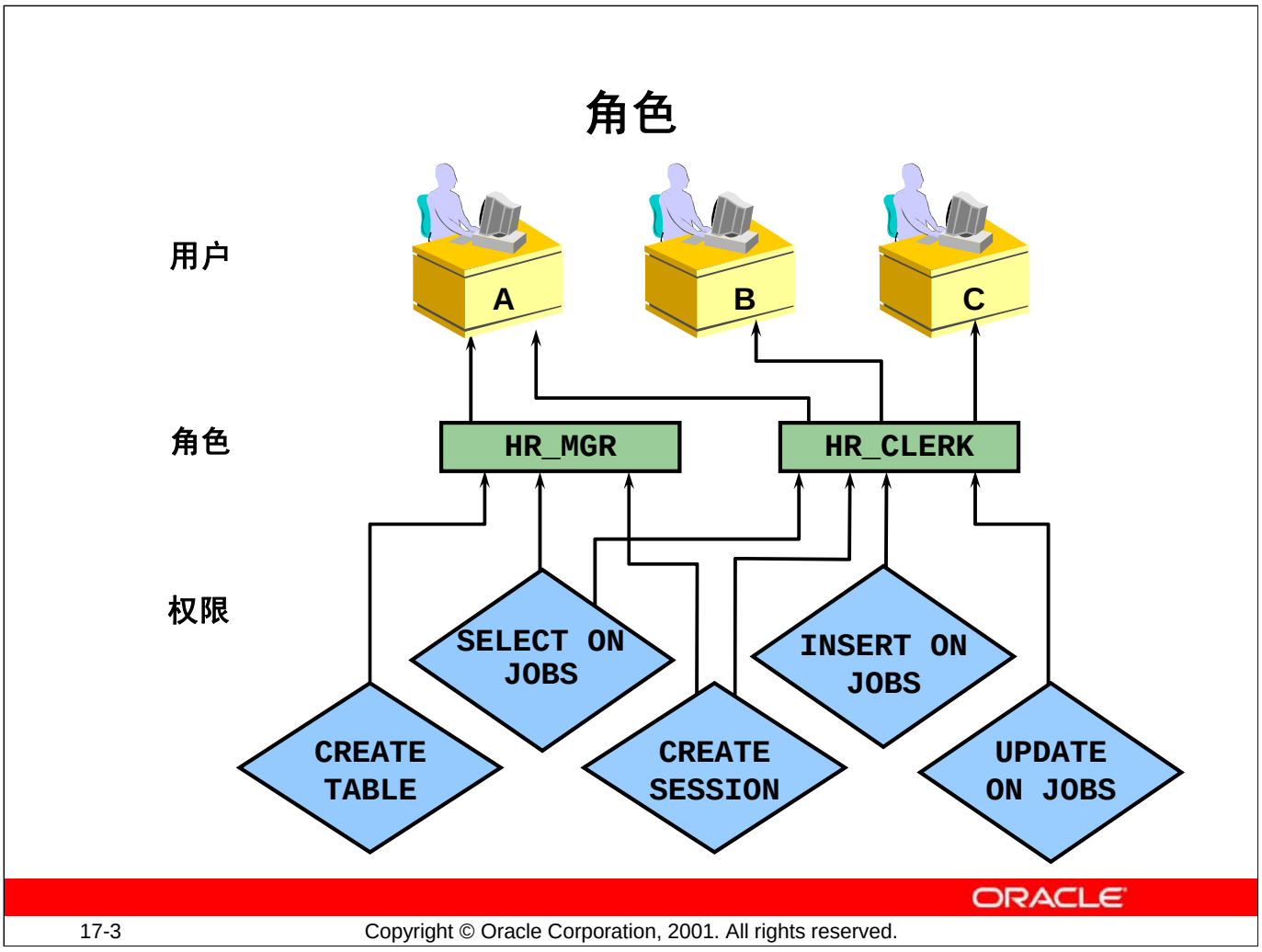
17

管理角色

目标

完成这一课的学习后，您应该能达到下列目标：

- 创建和修改角色
- 控制角色的可用性
- 删除角色
- 使用预定义角色
- 显示数据字典中的角色信息



什么是角色？

Oracle 通过角色提供简单且可控制的权限管理。角色是授予用户或其它角色的相关权限的指定组，旨在简化数据库中的权限管理。

角色的特点：

- 可以通过授予和撤消系统权限所用的命令来授予和撤消用户的角色。
- 可以将角色授予任何用户或角色。但是，不能将角色授予它本身，也不能循环授予。
- 角色可以由系统权限和对象权限组成。
- 对于被授予某种角色的每个用户来说，该角色可以启用，也可以禁用。
- 角色可要求通过口令启用。
- 在现有的用户名和角色名中，每个角色名必须唯一。
- 角色不属于任何人，也不存在于任何方案中。
- 在数据字典中存储了有关角色的说明。

角色的优点

- 轻松权限管理
- 动态权限管理
- 可选择权限可用性
- 可以通过操作系统授予

角色的优点

轻松权限管理：

使用角色可以简化权限管理。不是将同一组权限授予多个用户，而是将这些权限授予一个角色，然后将该角色授予每个用户。

动态权限管理：

如果修改了与角色关联的权限，被授予该角色的所有用户都将立即自动获得修改后的权限。

可选择权限可用性：

可启用和禁用角色以暂时打开和关闭权限。还可以通过启用角色验证用户是否已被授予该角色。

可以通过操作系统授予：

可以使用操作系统命令或实用程序向数据库中的用户分配角色。

创建角色

通过 ADMIN 选项授予的角色：

- **不验证：**

```
CREATE ROLE oe_clerk;
```

- **使用口令：**

```
CREATE ROLE hr_clerk
IDENTIFIED BY bonus;
```

- **外部验证：**

```
CREATE ROLE hr_manager
IDENTIFIED EXTERNALLY;
```


创建角色

使用 CREATE ROLE 语句可创建角色。必须具有 CREATE ROLE 系统权限才能创建角色。创建类型为 NOT IDENTIFIED、IDENTIFIED EXTERNALLY 或 BY password 的角色时，通过 ADMIN 选项为该角色授权。

使用下列命令创建角色：

```
CREATE ROLE role [NOT IDENTIFIED | IDENTIFIED
    {BY password | EXTERNALLY | GLOBALLY | USING package}]
```

其中：

role：是角色的名称

NOT IDENTIFIED：表明启用该角色时，不需要进行验证

IDENTIFIED：表明启用该角色时，需要进行验证

BY password：提供用户在启用角色时必须指定的口令

USING package：创建应用程序角色，该角色只能由使用授权的程序包的应用程序启用

EXTERNALLY：表明在启用该角色之前，用户必须由外部服务（例如操作系统或第三方服务）授权

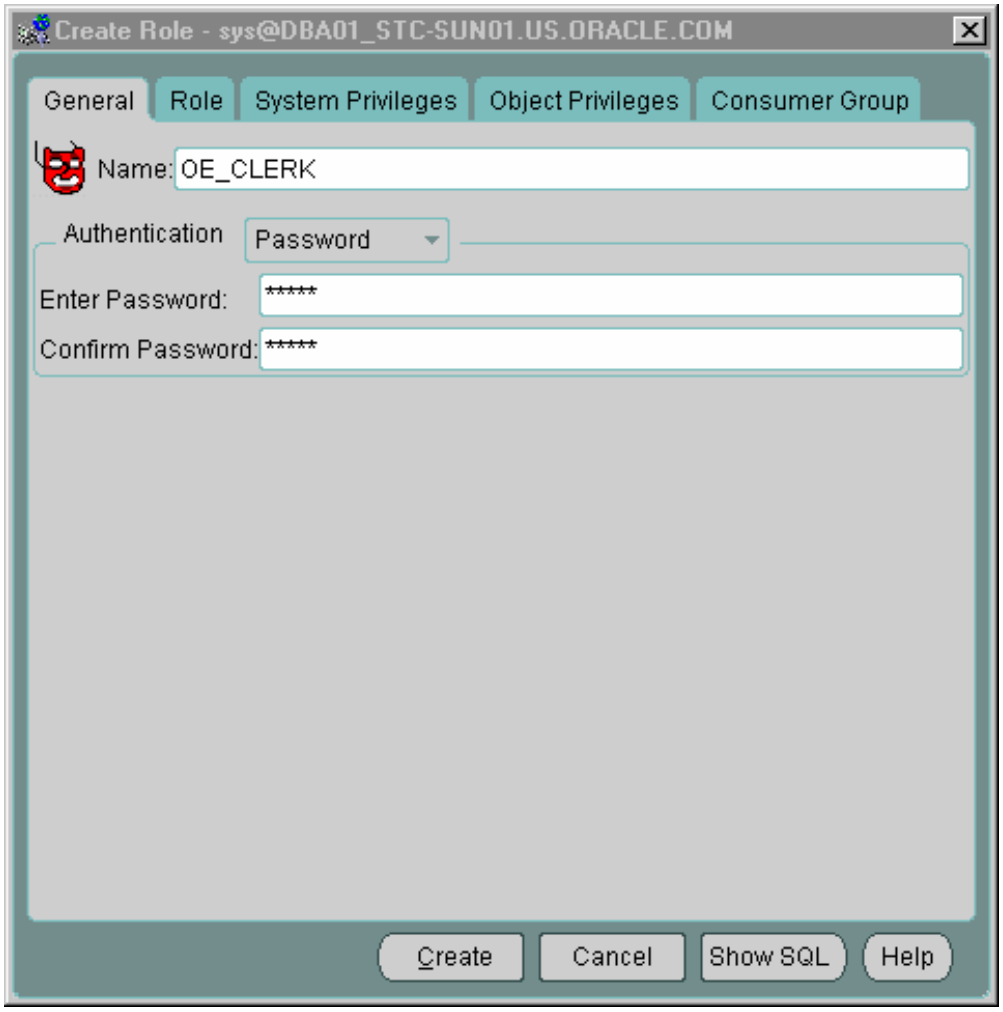
GLOBALLY：表明通过 SET ROLE 语句启用角色之前或登录时，必须由企业目录服务授权用户使用该角色

创建角色

使用 Oracle Enterprise Manager 创建角色

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “角色” (Roles)。
2. 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
5. 完成填写创建角色所需的信息。
6. 单击 “创建” (Create)。



预定义角色

角色名	说明
CONNECT, RESOURCE, DBA	提供这些角色的目的是为了向后兼容
EXP_FULL_DATABASE	导出数据库的权限
IMP_FULL_DATABASE	导入数据库的权限
DELETE_CATALOG_ROLE	对于数据字典表的 DELETE 权限
EXECUTE_CATALOG_ROLE	对于数据字典程序包的 EXECUTE 权限
SELECT_CATALOG_ROLE	对于数据字典表的 SELECT 权限

预定义角色

运行数据库创建脚本时，系统列出的角色是为 Oracle 数据库自动定义的角色。提供 CONNECT、 RESOURCE 和 DBA 角色的目的是为了向后与 Oracle 服务器的早期版本兼容。

提供了 EXP_FULL_DATABASE 和 IMP_FULL_DATABASE 角色以便于使用导入和导出实用程序。

提供 DELETE_CATALOG_ROLE、EXECUTE_CATALOG_ROLE 和 SELECT_CATALOG_ROLE 角色，用于访问数据字典视图和程序包。这些角色可以授予不具有 DBA 角色、但要求访问数据字典中的视图和表的用户。

其它特殊角色：

Oracle 服务器还创建授权您管理数据库的其它角色。在许多操作系统中，这些角色称为 OSOPER 和 OSDBA。它们的名称可能因操作系统而异。

其它角色是由数据库附带的 SQL 脚本定义的。例如，AQ_ADMINISTRATOR_ROLE 提供管理高级排队的权限。AQ_USER_ROLE 已废弃，继续使用的目的主要是为了与版本 8.0 兼容。

修改角色

- 使用 ALTER ROLE 可修改验证方法。
- 要求使用 ADMIN 选项或具有 ALTER ANY ROLE 权限。

```
ALTER ROLE oe_clerk
IDENTIFIED BY order;
```

```
ALTER ROLE hr_clerk
IDENTIFIED EXTERNALLY;
```

```
ALTER ROLE hr_manager
NOT IDENTIFIED;
```

ORACLE

修改角色

修改角色时，只能更改其验证方法。而且，您的角色必须通过 ADMIN 选项进行授予，或者您必须具有 ALTER ANY ROLE 系统权限。

使用下列命令修改角色：

```
ALTER ROLE role {NOT IDENTIFIED | IDENTIFIED
 {BY password |USING package| EXTERNALLY | GLOBALLY }};
```

其中：

role：是角色的名称

NOT IDENTIFIED：表明启用该角色时，不需要进行验证

IDENTIFIED：表明启用该角色时，需要进行验证

BY password：提供启用角色时所使用的口令

EXTERNALLY：表明在启用该角色之前，用户必须由外部服务（例如操作系统或第三方服务）授权

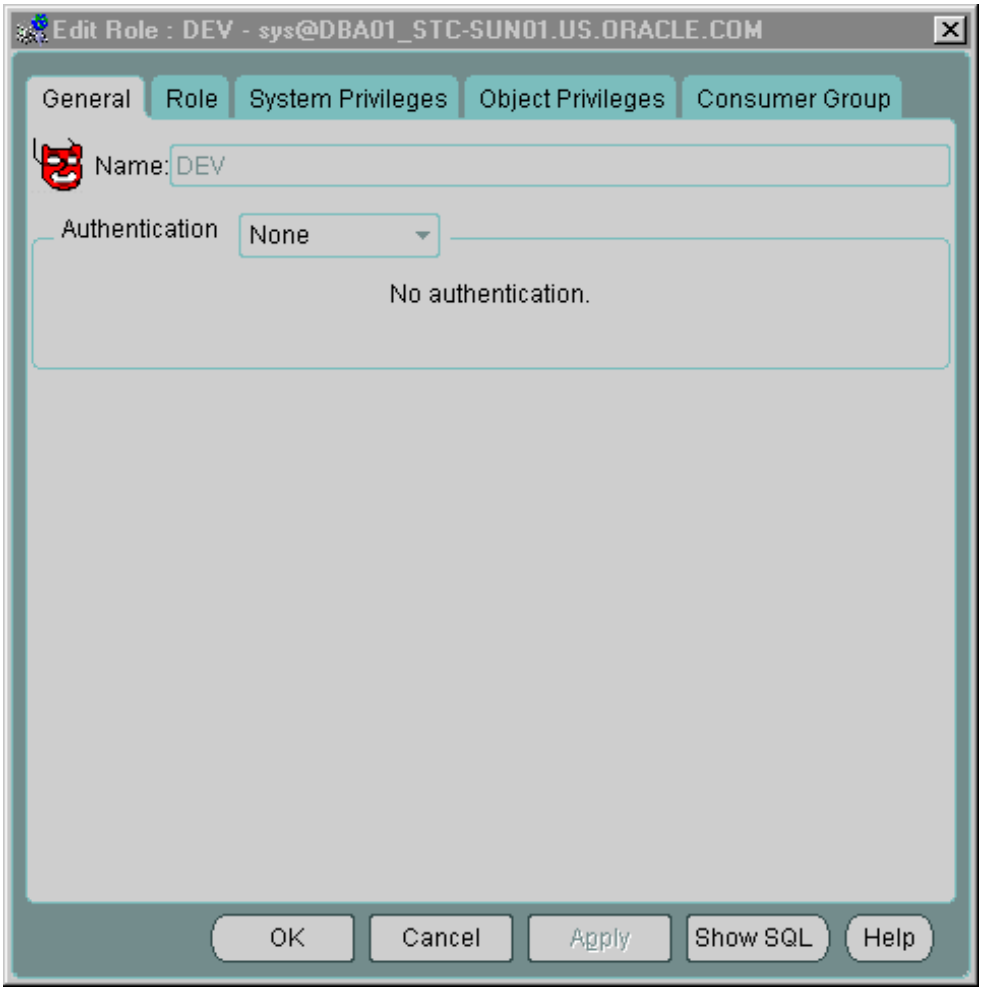
GLOBALLY：表明通过 SET ROLE 语句启用角色之前或登录时，必须由企业目录服务授权用户使用该角色

修改角色（续）

使用 Oracle Enterprise Manager 修改角色

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “角色” (Roles)。
2. 选择要修改的角色。
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. 进行修改。
5. 单击 “确定” (OK)。



分配角色

使用 GRANT 命令分配角色

```
GRANT oe_clerk TO scott;
```

```
GRANT hr_clerk TO hr_manager;
```

```
GRANT hr_manager TO scott WITH ADMIN OPTION;
```


分配角色

可使用为用户授予系统权限所用的语法命令为用户授予角色：

```
GRANT role [, role ]...
TO {user|role|PUBLIC}
 [, {user|role|PUBLIC} ]...
[WITH ADMIN OPTION]
```

其中：

role：是要授予的角色集合

PUBLIC：将角色授予所有用户

WITH ADMIN OPTION：使被授予者能够为其他用户或角色授予角色。（如果使用该选项授予角色，被授予者将能够授予和撤消其他用户的角色，并可改变或删除角色。）

分配角色（续）

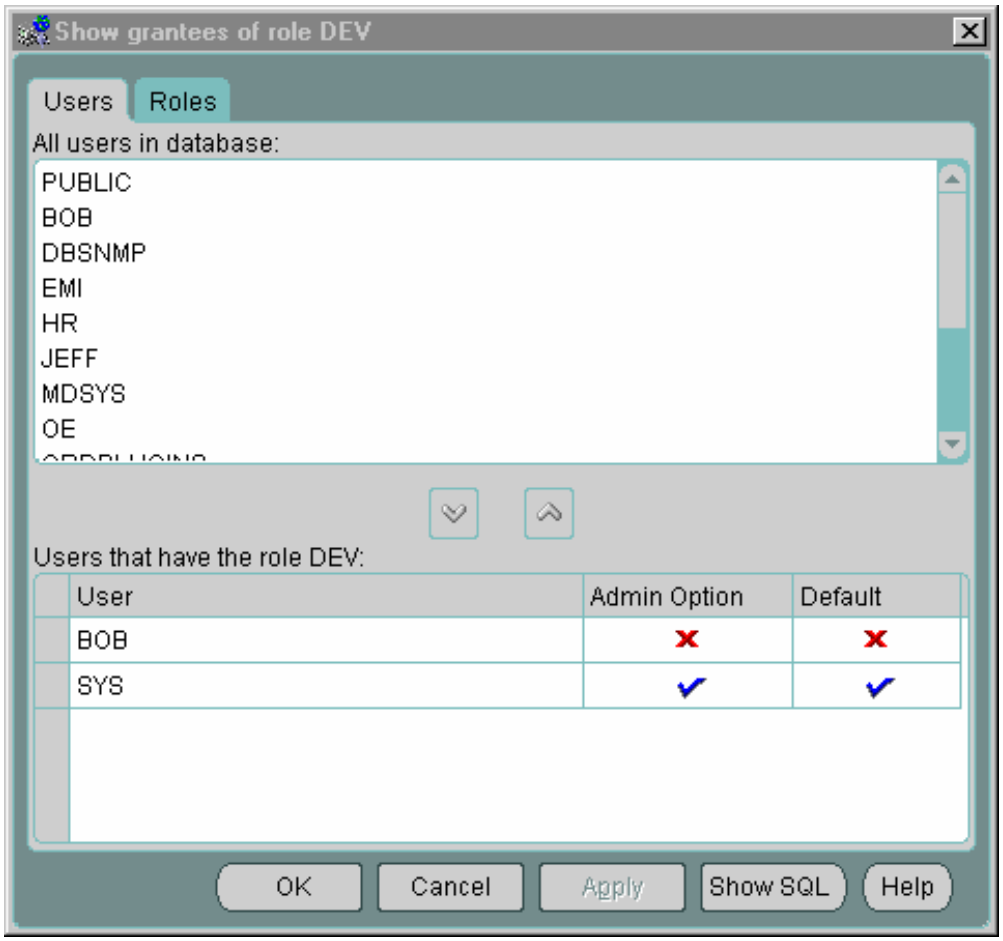
缺省情况下，对于创建角色的用户，系统将通过 ADMIN OPTION 为其分配角色。没有通过 ADMIN OPTION 被授予角色的用户需要具有 GRANT ANY ROLE 系统权限才能授予和撤消其它角色。

注：用户可启用的数据库角色的最多数目由初始化参数 MAX_ENABLED_ROLES 设置。

分配角色（续）

使用 Oracle Enterprise Manager 分配角色

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “角色” (Roles)。
2. 选择要分配的角色。
3. 单击鼠标右键，从弹出的菜单中选择 “被授予者” (Grantees)。
4. 从 “用户” (Users) 页选择要为其分配该角色的用户。
5. 单击向下箭头将该用户移入 “下列用户具有角色” (Users that have the role) 窗口中。
6. 单击 “确定” (OK)。



设置缺省角色

- 可以为一个用户分配多个角色。
- 可以为用户分配一个缺省角色。
- 限制用户的缺省角色的数目。

```
ALTER USER scott
     DEFAULT ROLE hr_clerk, oe_clerk;
```

```
ALTER USER scott DEFAULT ROLE ALL;
```

```
ALTER USER scott DEFAULT ROLE ALL EXCEPT
     hr_clerk;
```

```
ALTER USER scott DEFAULT ROLE NONE;
```


缺省角色

可以为一个用户分配多个角色。缺省角色是用户登录时自动启用的角色的子集。缺省情况下，登录时自动启用分配给用户的所有角色，而无需口令。使用 ALTER USER 命令可以限制用户的缺省角色的数目。

DEFAULT ROLE 子句只适用于已通过 GRANT 语句直接授予用户的角色。DEFAULT ROLE 子句不能用来启用下列角色：

- 未授予用户的角色
- 通过其它角色授予的角色
- 由外部服务（如操作系统）管理的角色

使用下列语法分配用户的缺省角色：

```
ALTER USER user DEFAULT ROLE
{role [,role]... | ALL [EXCEPT role [,role]... ] | NONE}
```

其中：

user：是被授予角色的用户的名称

role：是作为用户缺省角色的角色

缺省角色（续）

ALL：除 EXCEPT 子句中列出的角色外，将授予用户的所有角色都设置为缺省角色（这是缺省设置。）

EXCEPT：表明后随角色不应包括在缺省角色中

NONE：授予用户的任何角色都不作为缺省角色（用户登录时拥有的唯一权限是直接分配给该用户的权限。）

由于角色必须被授予后才有可能作为缺省角色，所以不能使用 CREATE USER 命令设置缺省角色。

应用程序角色

- 应用程序角色只能由授权的 PL/SQL 程序包启用。
- USING package 子句用来创建应用程序角色。

```
CREATE ROLE admin_role
IDENTIFIED USING hr.employee;
```


应用程序角色

CREATE ROLE 语句中的 USING package 子句用来创建应用程序角色。应用程序角色只能由使用授权的 PL/SQL 程序包的应用程序启用。应用程序开发人员无需在应用程序内嵌入口令来保护角色。他们可创建应用程序角色并指定授权哪个 PL/SQL 程序包可以启用该角色。

```
SQL> CREATE ROLE admin_role IDENTIFIED USING hr.employees;
```

本例中，admin_role 是一个应用程序角色，只有在 hr.employee PL/SQL 程序包内定义的模块才能启用该角色。

启用和禁用角色

- 禁用角色以暂时撤消用户拥有的该角色。
- 启用角色以暂时授予该角色。
- SET ROLE 命令可启用和禁用角色。
- 登录时启用用户的缺省角色。
- 启用角色可能需要口令。

启用和禁用角色

启用或禁用角色可暂时激活和不激活与角色关联的权限。必须首先为用户授予角色，然后才能启用该角色。

启用角色时，用户可以使用授予该角色的权限。如果禁用角色，用户将不能使用与该角色关联的权限，除非将该权限直接授予用户或授予为该用户启用的另一个角色。角色的启用针对的是一个会话。在下一个会话中，用户的活动角色将恢复为缺省角色。

指定要启用的角色：

SET ROLE 命令和 DBMS_SESSION.SET_ROLE 过程将启用包含在命令中的全部角色，并禁用所有其它角色。可以通过任何允许使用 PL/SQL 命令的工具或程序启用角色；但不能在存储过程中启用角色。

可以使用 ALTER USER...DEFAULT ROLE 命令指出用户登录时将启用的角色。所有其它角色都将被禁用。

启用角色时可能需要口令。SET ROLE 命令中必须包含口令才能启用角色。分配给用户的缺省角色不需要口令，这些角色同没有口令的角色一样在登录时启用。

启用和禁用角色（续）

限制：

不能在存储过程中启用角色，因为该操作可能会改变安全域（权限集），安全域允许首先调用存储过程。因此，在 PL/SQL 中，可以在匿名块和应用程序过程（如 Oracle Forms 过程）中启用和禁用角色，但不能在存储过程中执行该操作。

如果存储过程包含了 `SET ROLE` 命令，运行时会产生下列错误：

```
ORA-06565: cannot execute SET ROLE from within stored procedure
```

启用和禁用角色

```
SET ROLE hr_clerk;
```

```
SET ROLE oe_clerk IDENTIFIED BY order;
```

```
SET ROLE ALL EXCEPT oe_clerk;
```


启用和禁用角色

SET ROLE 命令关闭授予用户的任何其它角色。

```
SET ROLE {role [ IDENTIFIED BY password ]
         [, role [ IDENTIFIED BY password ]]...
         | ALL [ EXCEPT role [, role ] ...]
         | NONE }
```

其中：

role：是角色的名称

IDENTIFIED BY password：提供启用角色时所需的口令

ALL：除了 EXCEPT 子句中列出的角色外，启用授予当前用户的全部角色（不能使用该选项启用带口令的角色。）

EXCEPT role：不启用这些角色

NONE：禁用当前会话的全部角色（只有直接授予用户的权限是活动的。）

只有在启用的每个角色都没有口令时，不带 EXCEPT 子句的 ALL 选项才有效。

撤消用户角色

- 撤消用户角色需要有 ADMIN OPTION 或 GRANT ANY ROLE 权限。
- 使用以下命令撤消角色：

```
REVOKE oe_clerk FROM scott;
```

```
REVOKE hr_manager FROM PUBLIC;
```


撤消用户角色

使用 SQL 语句 REVOKE 可撤消用户角色。通过 ADMIN 选项获取角色的任何用户都可撤消任何其他数据库用户或角色的角色。此外，具有 GRANT ANY ROLE 权限的用户也可以撤消任何角色。

```
REVOKE role [, role ]
    FROM {user|role|PUBLIC}
     [, {user|role|PUBLIC} ]
```

其中

role：是要撤消的角色或从其撤消角色的角色

user：要撤消其系统权限或角色的用户

PUBLIC：撤消所有用户的权限或角色

撤消用户角色

使用 Oracle Enterprise Manager 撤消用户角色

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “安全性” (Security) > “用户” (Users)。
2. 选择将撤消其角色的用户。
3. 选择要撤消的角色。
4. 单击鼠标右键，从弹出的菜单中选择 “撤消” (Revoke)。
5. 选择 “是” (Yes) 确认撤消。

删除角色

- 删除角色：
  - 删除授予所有用户和角色的角色
  - 删除数据库角色
- 需要 ADMIN OPTION 或 DROP ANY ROLE 权限
- 使用以下命令删除角色：

```
DROP ROLE hr_manager;
```

ORACLE

删除角色

使用下列语法从数据库中删除角色：

```
DROP ROLE role
```

删除角色时，Oracle 服务器从所有用户和被授予该角色的角色中及数据库中撤消该角色。

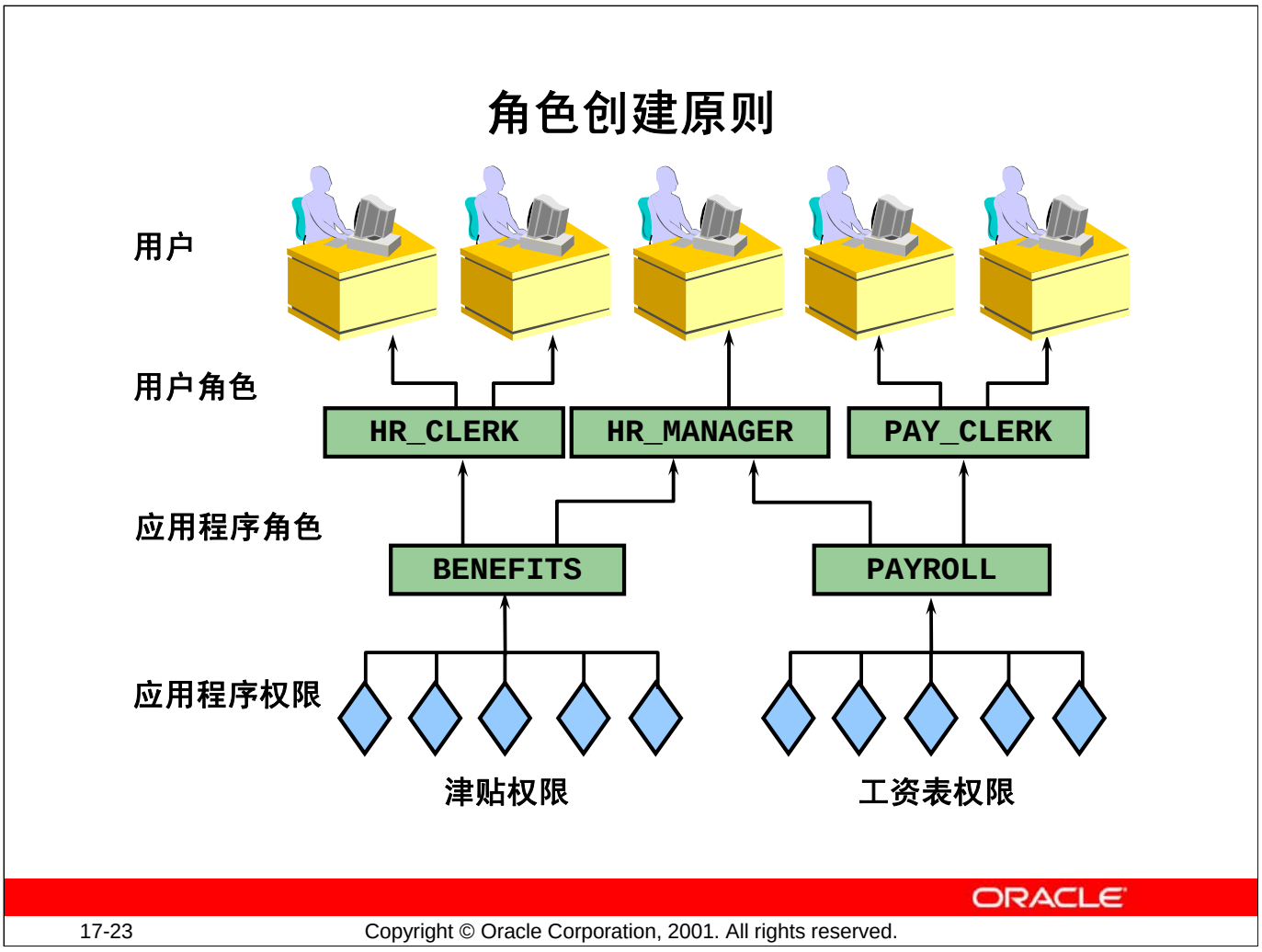
必须通过 ADMIN OPTION 被授予了角色或具有 DROP ANY ROLE 系统权限才能删除角色。

删除角色（续）

使用 Oracle Enterprise Manager 删除角色

从 “OEM 控制台” (OEM Console)：

1. 导航选择 “是” (Yes) 确认删除。到 “数据库” (Databases) > “安全性” (Security) > “角色” (Roles)。
2. 选择要删除的角色。
3. 单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。



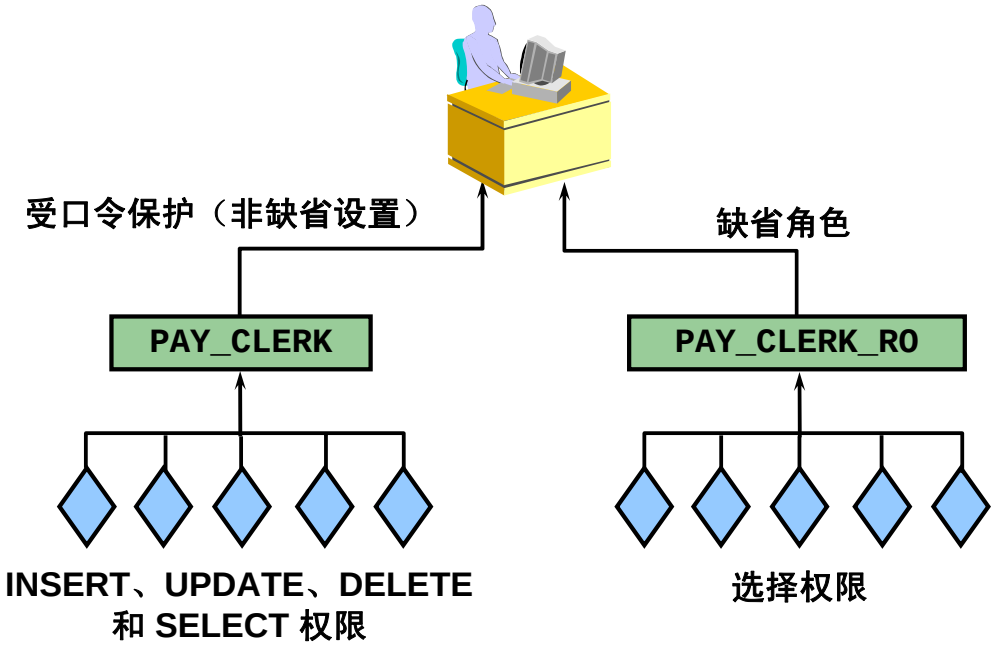
角色创建原则

因为角色包含执行任务所需的权限，因此，角色名通常就是应用程序任务或职称。幻灯片中的示例就使用应用程序任务与职称作为角色名。请按下列步骤创建、分配和授予用户角色：

1. 为每个应用程序任务创建一个角色。应用程序角色名对应于应用程序任务，如 PAYROLL。
2. 为应用程序角色分配执行任务所需的权限。
3. 为每种用户类型创建一个角色。用户角色名对应于职称，如 PAY_CLERK。
4. 将应用程序角色授予用户角色。
5. 将用户角色授予用户。

如果应用程序的修改要求使用新的权限执行工资表任务，则 DBA 只需向应用程序角色 PAYROLL 分配新的权限。当前正在执行该任务的所有用户都将接收到新的权限。

使用口令与缺省角色的原则



使用口令与缺省角色的原则

启用角色时口令提供了额外的安全级。例如，在启用 PAY_CLERK 角色时，应用程序可能要求用户输入口令，因为可以使用该角色发出支票。

通过使用口令，只能通过应用程序来启用角色。在幻灯片的示例中对这种技术做了说明。

- DBA 授予了用户两个角色：PAY_CLERK 和 PAY_CLERK_RO。
- PAY_CLERK 已被授予执行工资表职员功能所需的全部权限。
- PAY_CLERK_RO（RO 表示只读）仅被授予了在表上执行工资表职员功能所需的 SELECT 权限。
- 用户可以登录到 SQL* Plus 进行查询，但是不能修改任何数据，这是由于 PAY_CLERK 不是一个缺省角色，用户并不知道 PAY_CLERK 的口令。
- 当用户登录到工资表应用程序时，该程序提供口令以启用 PAY_CLERK。口令已编入程序中，系统不会提示用户输入口令。

获取角色信息

可以通过查询以下视图来获取有关角色的信息：

- **DBA_ROLES**：数据库中存在的所有角色
- **DBA_ROLES_PRIVS**：授予用户和角色的角色
- **ROLE_ROL_PRIVS**：授予角色的角色
- **DBA_SYS_PRIVS**：授予用户和角色的系统权限
- **ROLE_SYS_PRIVS**：授予角色的系统权限
- **ROLE_TAB_PRIVS**：授予角色的对象权限
- **SESSION_ROLES**：用户当前启用的角色

ORACLE

查询角色信息

许多数据字典视图不仅包含授予用户的权限信息，还包含有关角色是否需要口令的信息。

```
SQL> SELECT role, password_required
 2   FROM dba_roles;

ROLE                           PASSWORD
------------------------------ -----------
CONNECT                        NO
RESOURCE                       NO
DBA                            NO
SELECT_CATALOG_ROLE            NO
EXECUTE_CATALOG_ROLE           NO
DELETE_CATALOG_ROLE            NO
IMP_FULL_DATABASE              NO
EXP_FULL_DATABASE              NO
SALES_CLERK                    YES
HR_CLERK                       EXTERNAL
```

小结

在这一课中，您应该能够掌握：

- 创建角色
- 为角色分配权限
- 为用户或角色分配角色
- 设置缺省角色

ORACLE

练习 17 概览

此练习涉及以下主题：

- 列出角色的系统权限
- 创建、分配和删除角色
- 创建应用程序角色

ORACLE

练习 17 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 17：管理角色

1 检查数据字典视图，并列出 RESOURCE 角色的系统权限。

2 创建名为 DEV 的角色，该角色允许被授予该角色的用户能够创建表、视图并能够从 Emi 的 CUSTOMERS1 表进行选择。

3 **a** 将 RESOURCE 角色和 DEV 角色分配给 Bob，但只允许 RESOURCE 角色在 Bob 登录时自动启用。

b 为 Bob 授予读取所有数据字典信息的权限。

4 Bob 必须检查例程当前使用的还原段。

以 Bob 身份连接并列出所使用的还原段。
提示：使用 SET ROLE SELECT_CATALOG_ROLE

5 以 SYSTEM 用户身份创建一个有关 Emi 的 CUSTOMERS 表的视图 CUST_VIEW。结果如何？

6 以用户 Emi 的身份为 SYSTEM 授予在 CUSTOMERS1 上进行选择的权限。以 SYSTEM 用户身份创建有关 Emi 的 CUSTOMERS1 表的视图 CUST_VIEW。结果如何？

18

审计

目标

完成这一课的学习后，您应该能达到下列目标：

- 概述审计类别
- 对例程启用审计
- 概述审计选项

审计

- 审计即监视选定的用户数据库操作，可用于以下目的：
  - 调查可疑的数据库活动
  - 收集有关特定数据库活动的信息
- 可以按会话审计或按访问审计。

审计

如果任何未经授权的用户试图删除数据，DBA 可以决定对数据库的所有连接，以及对数据库中所有表的所有成功和未成功的删除操作进行审计。DBA 会收集下面这些统计信息，如哪些表正在更新、已执行的逻辑输入/输出 (I/O) 次数、峰期时的并发连接用户数等。

审计原则

- 确定要审计的内容：
  - 用户、语句或对象
  - 语句执行情况
  - 成功的语句执行情况与/或未成功的语句执行情况
- 管理审计线索：
  - 监视审计线索的增长。
  - 防止未经授权而访问审计线索。

审计原则

通过首先确定审计要求，然后设置可以满足这些要求的最少审计选项来限定审计。要尽可能使用对象审计来减少生成的条目。如果必须使用语句或权限审计，则下列设置可以让审计生成的内容减至最少：

- 指定要审计的用户
- 按会话审计，而不是按访问审计
- 审计成功或失败，但不要既成功又失败

注：审计记录可以写入 SYS.AUD$ 或操作系统的审计线索。能否使用操作系统的审计线索要取决于操作系统。

监视审计线索的增长：

如果审计线索已满，则不能再插入任何审计记录，并且审计过的语句不能成功执行。发布已审计语句的所有用户都将收到错误消息。您必须先释放审计线索中的一些空间，而后才能执行这些语句。

审计原则（续）

监视审计线索的增长（续）：

要确保审计线索不会增长太快，应注意以下要求：

- 只有需要时才启用审计。
- 选择特定的审计选项。
- 严格控制方案对象审计。用户可以对自己拥有的对象打开审计。
- 尽量避免授予 AUDIT ANY 权限，因为它还能让用户打开审计。

定期使用 DELETE 或 TRUNCATE 命令删除审计线索中的审计记录。审计文件位于 $ORACLE_HOME/rdbms/audit 目录中。

保护审计线索：

应保护审计线索，以防添加、修改或删除审计信息。发出以下命令：

```
SQL> AUDIT delete ON sys.aud$ BY ACCESS;
```

可防止审计线索未经授权即被删除；只有 DBA 才拥有 DELETE_CATALOG_ROLE 角色。

将审计线索移到系统表空间外：

随着新记录插入数据库审计线索，表 AUD$ 将无限增长。尽管不应丢弃 AUD$ 表，但是可以从中删除或截断信息，因为其中的行只是为了提供信息，对于运行 Oracle 例程并不是必需的。由于 AUD$ 表在增长后又会收缩，所以应将其存储在系统表空间之外。

若要将 AUD$ 移至 AUDIT_TAB 表空间，则：

- 确保审计当前是禁用的。
- 输入以下命令：

```
SQL> ALTER TABLE aud$ MOVE TABLESPACE AUDIT_TAB;
```

- 输入以下命令：

```
SQL> CREATE INDEX i_aud1 ON aud$(sessionid, ses$tid)
 2   TABLESPACE AUDIT_IDX;
```

- 对例程启用审计。

审计类别

- 缺省审计
  - 例程启动与例程关闭
  - 管理员权限
- 数据库审计
  - 由 DBA 启用
  - 不能记录列值
- 基于值的审计或应用程序审计
  - 通过代码实施
  - 能记录列值
  - 用来跟踪对表所做的更改

ORACLE

审计类别

不论是否启用数据库审计，Oracle 都始终将一些数据库操作记录到操作系统审计线索中。这些记录包括：

- 例程启动：审计记录会详述正启动例程的操作系统用户、用户终端标识符、日期和时间戳，以及已启用还是已禁用数据库审计。
- 例程关闭：详述正关闭例程的操作系统用户、用户的终端标识符以及日期和时间戳。
- 管理员权限：详述正以管理员权限连接 Oracle 的操作系统用户。

数据库审计：

数据库审计监视并记录选定的用户数据库操作。有关事件的信息存储在审计线索中。审计线索可用于调查可疑操作。例如，如果未授权的用户正在删除表中的数据，DBA 会决定审计数据库的所有连接，以及对数据库中表行的成功与不成功的删除操作。

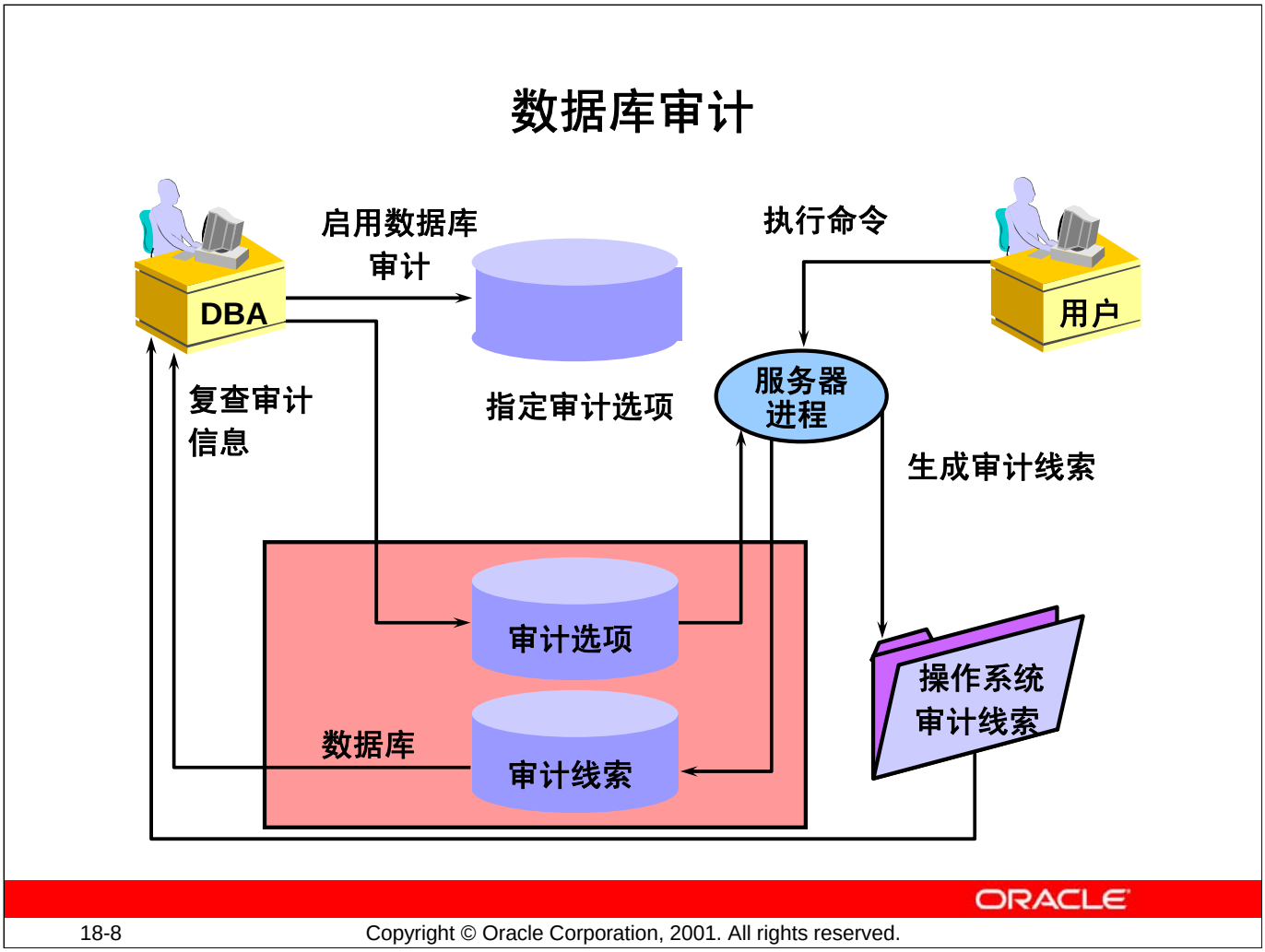
审计类别（续）

数据库审计（续）：

审计也可用于监视并收集关于特定数据库活动的数据。例如，DBA 可以收集有关哪些表正在更新、执行的逻辑 I/O 的操作次数以及峰期时的并发用户连接数等统计信息。

基于值的审计：

数据库审计无法记录列值。若需要跟踪数据库列的更改并存储每个更改的列值，则使用应用程序审计。可以通过客户代码、存储过程或数据库触发器进行应用程序审计。



数据库审计

启用和禁用数据库审计：

确定要审计的内容后，设置初始化参数 AUDIT_TRAIL 对例程启用审计。该参数指明是将审计线索写入数据库表还是操作系统审计线索。

```
AUDIT_TRAIL = value
```

其中，值可以是下列之一：

TRUE 或 DB：启用审计并将所有审计记录定向到数据库审计线索 (SYS.AUD$)

OS：启用审计并将所有审计记录定向到操作系统审计线索（如果操作系统上允许）

FALSE 或 NONE：禁用审计

注：没有缺省值。

数据库审计（续）

除非 DBA 已将参数 AUDIT_TRAIL 设置为 DB 或 OS，否则不将审计记录写入审计线索。尽管可随时使用 SQL 语句 AUDIT 和 NOAUDIT，但如果 DBA 已经在初始化文件中设置了参数 AUDIT_TRAIL，则只将记录写入审计线索。

注：操作系统的*安装和配置指南*提供了有关将审计记录写入操作系统审计线索的信息。

指定审计选项：

接下来，使用 AUDIT 命令设置特定的审计选项。通过 AUDIT 命令，可以指出要审计的命令、用户、对象或权限。也可以指出是针对每个事件生成一个审计记录还是针对每个会话生成一次审计记录。如果不再需要某个审计选项，可使用 NOAUDIT 命令关闭它。

语句的执行：

当用户执行 PL/SQL 和 SQL 语句时，服务器进程检查这些审计选项以确定正在执行的语句是否生成审计记录。如果执行的是 PL/SQL 程序单元，则会根据需要对其中的 SQL 语句分别进行审计。由于视图和过程可能引用其它数据库对象，所以执行单个语句可能会生成多个审计记录。

生成审计数据：

审计线索记录的生成和插入与用户的事务处理无关；因此，即使用户的事务处理回退，审计线索记录仍保持原样。由于审计记录在执行阶段生成，语法分析阶段出现的语法错误并不会导致生成审计线索记录。

复查审计信息：

通过从审计线索数据字典视图中选择，或使用操作系统实用程序查看操作系统审计线索，可以检查审计过程中生成的信息。该信息用于调查可疑操作并监视数据库操作。

审计选项

- 语句审计

```
AUDIT TABLE;
```

- 权限审计

```
AUDIT create any trigger;
```

- 方案对象审计

```
AUDIT SELECT ON emi.orders;
```


审计选项

语句审计：该种审计对 SQL 语句进行选择性审计，而并不审计语句针对的特定方案对象。例如，AUDIT TABLE 跟踪多个 DDL 语句，而与这些语句针对的表无关。可以设置语句审计，以便对数据库中的所选用户或每个用户进行审计。

权限审计：

该种审计对执行操作应具有的相应系统权限进行选择性审计，如 AUDIT CREATE ANY TRIGGER。可以设置权限审计对数据库中的所选用户或每个用户进行审计。

方案对象审计：

该种审计对特定方案对象上的特定语句进行选择性审计，如 AUDIT SELECT ON HR.EMPLOYEES。方案对象审计始终适用于所有数据库用户。

可以指定任何审计选项，并指定下列条件：

- WHENEVER SUCCESSFUL/WHENEVER NOT SUCCESSFUL
- BY SESSION/BY ACCESS

审计选项

小粒度审计

- **对基于内容的数据访问进行监视**
- **通过 DBMS_FGA 程序包实施**

ORACLE

审计选项

小粒度审计：对于基于内容的数据访问进行监视。可通过 PL/SQL 程序包 DBMS_FGA 来管理基于值的审计策略。使用 DBMS_FGA 时，DBA 会创建一个针对目标表的审计策略。如果从查询块返回的任何行符合审计条件，就会向审计线索插入一个审计事件条目，包括用户名、SQL 文本、绑定变量、策略名、会话 ID、时间戳与其它属性。

禁用审计：

使用 NOAUDIT 语句以停止由 AUDIT 命令选择的审计。

注：NOAUDIT 语句可以取消先前 AUDIT 语句的作用。要注意 NOAUDIT 语句必须与先前的 AUDIT 语句具有相同的语法，并且它只能取消那个特定语句的作用。因此，如果一个 AUDIT 语句（语句 A）对特定用户启用审计，另一个语句（语句 B）对所有用户启用审计，则对所有用户禁用审计的 NOAUDIT 语句只取消语句 B 的作用，而语句 A 仍然有效，可继续审计其指定的用户。

获取审计信息

可以通过查询以下视图来获取有关审计的信息：

- **ALL_DEF_AUDIT_OPTS**
- **DBA_STMT_AUDIT_OPTS**
- **DBA_PRIV_AUDIT_OPTS**
- **DBA_OBJ_AUDIT_OPTS**

ORACLE

获取审计信息

数据字典视图	说明
ALL_DEF_AUDIT_OPTS	缺省审计选项
DBA_STMT_AUDIT_OPTS	语句审计选项
DBA_PRIV_AUDIT_OPTS	权限审计选项
DBA_OBJ_AUDIT_OPTS	方案对象审计选项

获取审计记录信息

可以通过查询以下视图来获取有关审计记录的信息：

- **DBA_AUDIT_TRAIL**
- **DBA_AUDIT_EXISTS**
- **DBA_AUDIT_OBJECT**
- **DBA_AUDIT_SESSION**
- **DBA_AUDIT_STATEMENT**

ORACLE

获取审计记录信息

列出审计记录：

数据库审计线索 (SYS.AUD$) 是一个包含在每个 Oracle 数据库字典中的单独表，其中有若干预定义视图。幻灯片中列出了其中的一些视图，这些视图是 DBA 创建的。

数据字典视图	说明
DBA_AUDIT_TRAIL	所有审计线索条目
DBA_AUDIT_EXISTS	有关 AUDIT EXISTS/NOT EXISTS 的记录
DBA_AUDIT_OBJECT	有关方案对象的记录
DBA_AUDIT_SESSION	所有连接和断开连接条目
DBA_AUDIT_STATEMENT	语句审计记录

小结

在这一课中，您应该能够掌握：

- 概述审计需要
- 启用和禁用审计
- 确定和使用各种审计选项

练习 18 概览

本课没有练习。

19

将数据加载到数据库中

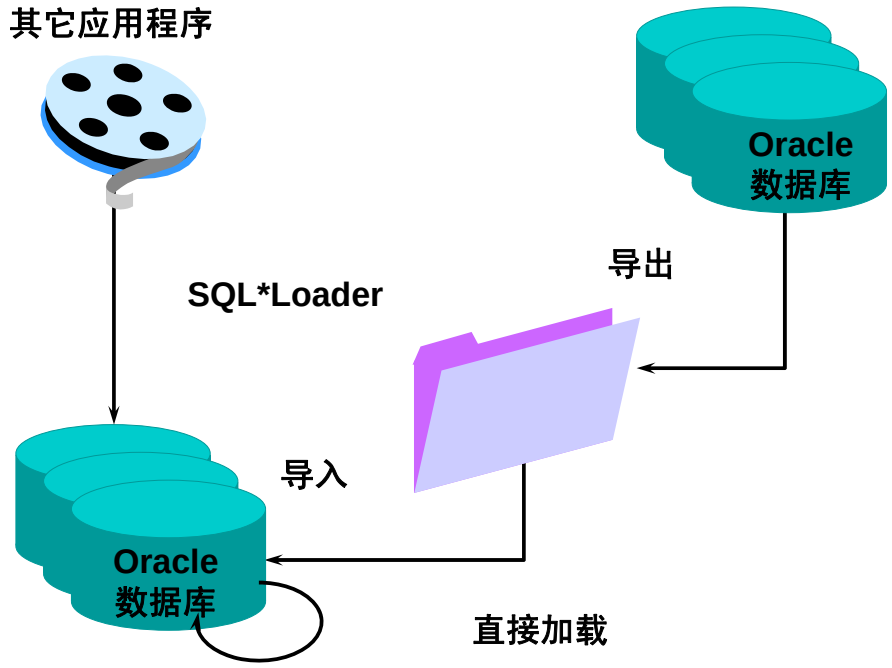
目标

完成这一课的学习后，您应该能达到下列目标：

- 演示直接加载操作的用法
- 说明 SQL*Loader 的用法
- 执行基本的 SQL*Loader 操作
- 列出 SQL*Loader 和直接加载的使用原则

ORACLE

数据加载方法



数据加载方法

有数种方法可用于向 Oracle 数据库的表中加载数据。此处讨论其中的直接加载插入和 SQL*Loader 这两种方法。导出和导入将在 *Oracle9i 数据库管理基础 II* 课程中加以讨论。

SQL*Loader:

SQL*Loader 将数据从外部文件加载到 Oracle 数据库表中。SQL*Loader 包含一个功能强大的数据分析引擎，该引擎对数据文件中数据的格式几乎没有限制。

直接加载：

直接加载插入可用于在同一数据库中从一个表向另一个表复制数据。此方法绕过数据库缓冲区高速缓存直接将数据写入数据文件，从而加快了插入操作的速度。

直接加载

可以使用以下方法执行直接加载插入：

- 正常（串行）或并行
- 加载到分区表、非分区表或表的单个分区中
- 记录或不记录重做数据

ORACLE

直接加载

直接加载插入（串行或并行）只能支持 INSERT 语句的 INSERT ... SELECT 语法而无法支持其 INSERT ... Values 语法。INSERT ... SELECT 的并行性是由并行提示或并行表定义决定的。Oracle9*i* 提供了语法扩展，扩大了 INSERT ... SELECT 语句的使用范围。结果，可以将行插入到多个表中（作为单个 DML 语句的一部分）。

可使用 APPEND 提示调用直接加载插入，如以下的命令所示：

```
INSERT /*+APPEND */ INTO [ schema. ] table
   [ [NO]LOGGING ]
sub-query;
```

其中：

schema：表的所有者

table：表名

sub-query：用于选择所要插入的列和行的子查询

直接加载插入（续）

LOGGING 模式：

如果使用 LOGGING 选项（缺省设置）进行插入，该操作将生成重做日志条目，从而使得从失败中完全恢复成为可能。如果使用 NOLOGGING 选项，则不会将对数据所做的更改记入重做日志缓冲区中。但对于更新数据字典的操作，仍进行最小限度的记录。如果已为表设置 NOLOGGING 属性，则将使用 NOLOGGING 模式。

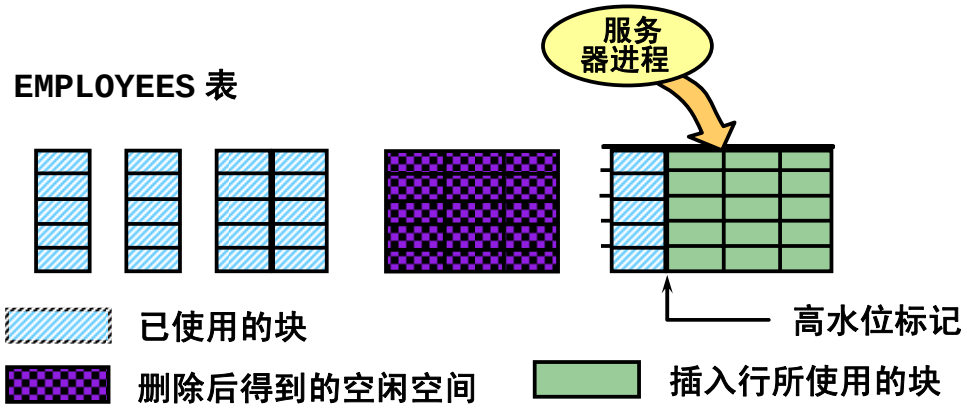
如果接下来可能对表数据进行多次联机修改，则最好在加载前设置 NOLOGGING 属性，并在完成加载后再将表的属性重置为 LOGGING。

其它注意事项：

通过直接加载插入而加载的所有数据都将加载在高水位标记之上。如果表中有许多块中的行已被删除，则可能会浪费空间并减慢全表扫描的速度。

串行直接加载

```
INSERT /*+ APPEND */ INTO emp
NOLOGGING
SELECT * FROM t_employees;
COMMIT;
```



ORACLE

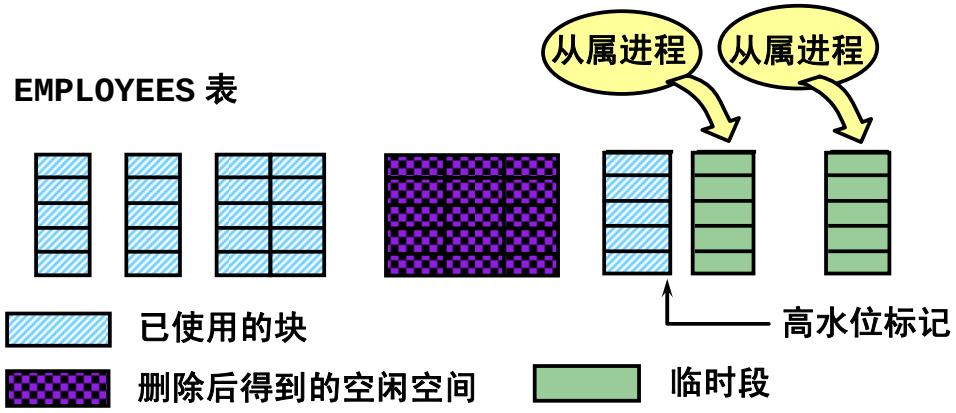
串行直接加载

到非分区表、分区表或子分区表中的串行直接加载插入：

数据将插入到表段或各分区段的当前高水位标记之上。高水位标记是一个分界线，分界线以上的块不会格式化，因而无法接收数据。执行一条语句后，就会将高水位标记更新为一个新值，以便使些数据可见。当加载分区表或子分区表时，SQL*Loader 将对行进行分区并维护索引（也可进行分区）。如果对分区表或子分区表进行直接路径加载，那可能会消耗大量的资源。

并行直接加载

```
ALTER SESSION ENABLE PARALLEL DML;
INSERT /*+PARALLEL(hr.employees,2) */
INTO hr.employees NOLOGGING
SELECT * FROM hr.old_employees;
```



ORACLE

并行直接加载

可以使用下列方法之一来并行执行直接加载插入：

- 在 INSERT 语句中使用 PARALLEL 提示，如上面的示例所示
- 创建表或改变表以指定 PARALLEL 子句

进行并行的直接加载插入时，Oracle 服务器会使用多个称为 “并行查询从属” 的进程将数据插入到表中。分配临时段以存储各个从属进程所插入的数据。提交事务处理时，这些段中的区就将成为用来插入记录的那个表的一部分。

注：

- 必须在事务处理一开始时执行 ALTER SESSION ENABLE PARALLEL DML 命令。
- 在同一事务处理中，不能再次查询或修改那些通过并行直接加载插入而被修改的对象。

有关并行直接加载插入的更详细讨论，请参阅 *Oracle9i Database Reference* 中的 “Parallel Execution” 部分。

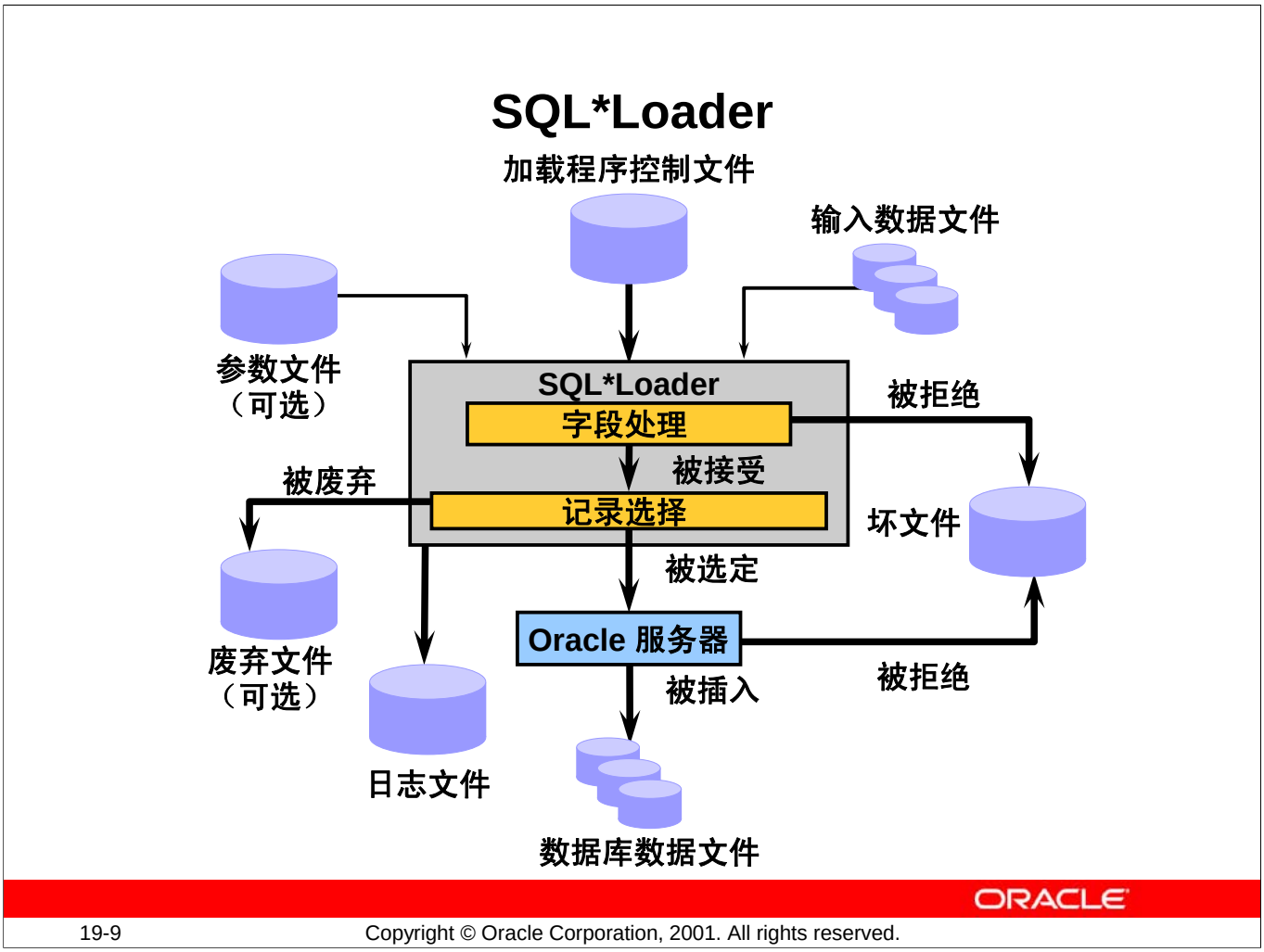
并行直接加载（续）

到非分区表中的并行直接加载插入：

每个并行执行服务器都将分配一个新的临时段，并将数据插入到该临时段中。执行一条语句后，并行执行协调程序就会将这些新临时段合并为主表段。

到分区表中的并行直接加载插入：

给每个并行执行服务器分配一个或多个分区，其中每个分区最多只能有一个进程在运行。并行执行服务器将数据插入到给它分配的分区段的当前高水位标记之上。执行一条语句后，并行执行协调程序就会将每个分区段的高水位标记更新为一个新值，以便使这些数据可见。



SQL*Loader 功能

SQL*Loader 将数据从外部文件加载到 Oracle 数据库的表中。SQL*Loader 具有下列功能：

- SQL*Loader 可以使用一个或多个输入文件。
- 可以将多个输入记录组合成一个逻辑记录进行加载。
- 输入字段的长度可以是固定的，也可以是不固定的。
- 输入数据可以采用任何格式：字符、二进制、压缩十进制、日期和分段十进制。
- 可以从磁盘、磁带或命名管道等不同类型的介质加载数据。
- 可以在一次运行中将数据加载到多个表中。
- 有用于替换或追加到表中现有数据的选项。
- 在行存储于数据库之前，可以将 SQL 函数应用于输入数据。
- 可基于规则自动生成列值。例如，可以生成顺序键值并将其存储在列中。
- 可以绕过数据库缓冲区高速缓存，将数据直接加载到表中。

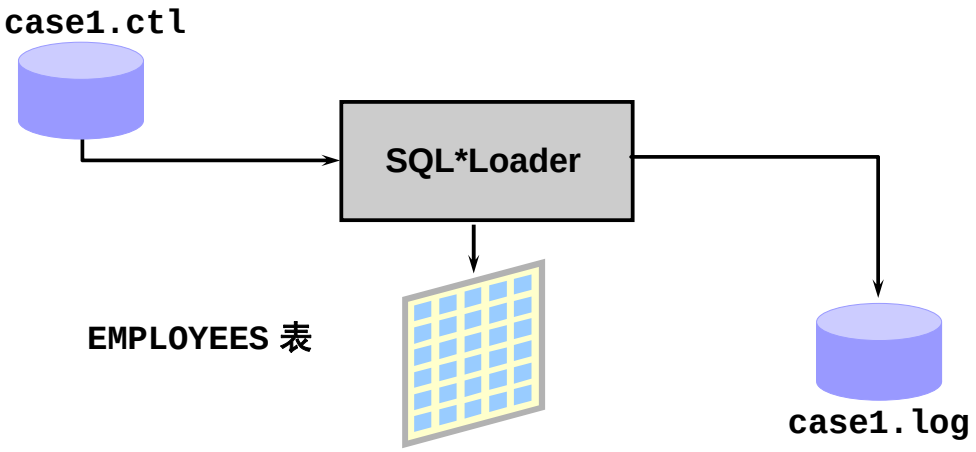
SQL*Loader 使用的文件

SQL*Loader 使用下列文件：

- 加载程序控制文件：指定输入格式、输出表以及可选条件（借助可选条件可指定仅加载输入数据文件中的部分记录）
- 输入数据文件：其中的数据采用控制文件中定义的格式
- 参数文件：是可选文件，可用于定义加载的命令行参数
- 日志文件：由 SQL*Loader 创建并包含加载记录
- 坏文件：由实用程序用于写入加载过程中被拒绝的记录（这会在实用程序验证输入记录期间或 Oracle 服务器插入记录期间发生。）
- 废弃文件：需要时可创建的文件，用于存储所有未满足选择准则的记录

使用 SQL*Loader

```
$sqlldr hr/hr \
> control=case1.ctl \
> log=case1.log direct=Y
```



使用 SQL*Loader

命令行：

在调用 SQL*Loader 时，可以指定一些参数以确定会话的特性。可以按任意顺序输入参数，并用逗号将它们隔开（可选）。可以指定参数值；在某些情况下，也可接受缺省值而不输入任何值。

如果调用 SQL*Loader 时未指定任何参数，则 SQL*Loader 将显示 “帮助” (Help) 屏幕，其中列出了可用的参数及其缺省值。

使用 SQL*Loader（续）

使用 Oracle Enterprise Manager 执行加载操作（使用“加载向导”）

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开要在其中加载数据的那个表。
3. 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
4. 此时，就会显示加载向导的 “简介” (Introduction) 页。
5. 选择 “下一步” (Next) 开始进行加载。
   注：本课中的其余幻灯片讨论了 “加载向导” 中所需的信息，如下所示：
6. 使用 “控制文件” (Control File) 页
7. 使用 “数据文件” (Data File) 页
8. 使用 “加载方法” (Load Method) 页可定义加载方法
9. 使用 “加载方法” (Load Method) 页可定义可选的文件
10. 使用 “调度” (Schedule) 页
11. 使用 “作业信息” (Job Information) 页
12. 使用 “概要” (Summary) 页

SQL*Loader 控制文件

加载程序控制文件通知 SQL*Loader 以下信息：

- **加载数据的所在位置**
- **数据格式**
- **配置方面的详细信息**
  - **内存管理**
  - **记录拒绝情况**
  - **与处理被中断的加载有关的详细信息**
- **如何操纵数据**

ORACLE

SQL*Loader 控制文件

SQL*Loader 控制文件是一个文本文件，其中包含一些数据定义语言 (DDL) 指令。DDL 用于控制 SQL*Loader 会话的以下方面：

- SQL*Loader 到何处查找所要加载的数据
- SQL*Loader 希望数据采用什么格式
- SQL*Loader 在加载数据时如何进行配置（内存管理、拒绝记录、处理被中断的加载等）
- SQL*Loader 如何操纵正在加载的数据

虽然并未明确定义，但是可以将加载程序控制文件划分为三个部分。

- 第一部分包含会话范围内的信息，例如：
  - 全局选项（如绑定大小、行、要跳过的记录等）
  - 指定输入数据所在位置的 `INFILE` 子句
  - 如何加载数据
- 第二部分包含一个或多个 `INTO TABLE` 块。其中每个块都包含一些与要在其中加载数据的那个表有关的信息，如该表的名称以及表中各列。
- 第三部分是可选的。如果该部分存在，其中将包含输入数据。

SQL*Loader 控制文件（续）

以下示例为一个典型的 SQL*Loader 控制文件。

```
1  -- This is a sample control file
2  LOAD DATA
3  INFILE 'SAMPLE.DAT'
4  BADFILE 'sample.bad'
5  DISCARDFILE 'sample.dsc'
6  APPEND
7  INTO TABLE emp
8  WHEN (57) = '.'
9  TRAILING NULLCOLS
10 (hiredate SYSDATE,
    deptno POSITION(1:2) INTEGER EXTERNAL(3)
   NULLIF deptno=BLANKS,
    job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
    NULLIF job=BLANKS "UPPER(:job)",
    mgr POSITION(28:31) INTEGER EXTERNAL
    TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
   ename POSITION(34:41) CHAR
    TERMINATED BY WHITESPACE "UPPER(:ename)",
    empno POSITION(45) INTEGER EXTERNAL
    TERMINATED BY WHITESPACE,
    sal POSITION(51) CHAR TERMINATED BY WHITESPACE
    "TO_NUMBER(:sal,'$99,999.99')",
    comm INTEGER EXTERNAL ENCLOSED BY '(' AND '%'
    ":comm * 100"
   )
```

对示例控制文件的解释如下：

1. 这就是在控制文件中输入注释的方法。注释可以出现在文件的命令部分中的任何地方，但不能出现在数据中。
2. LOAD DATA 语句通知 SQL*Loader 将要开始一个新的数据加载。如果要继续执行之前遭到中断的加载进程，则应该使用 CONTINUE LOAD DATA 语句。
3. INFILE 关键字指定其中包含所要加载的数据的数据文件的名称。

SQL*Loader 控制文件（续）

4. BADFILE 关键字指定用于保存遭拒绝记录的文件的名称。
5. DISCARDFILE 关键字指定用于放置废弃记录的文件的名称。
6. APPEND 关键字是那些可用于将数据加载到非空的表中的多种选项中的一种选项。若要将数据加载到空表中，则可以使用 INSERT 关键字。
7. INTO TABLE 关键字用于指定表、字段和数据类型。它定义了数据文件中的记录与数据库中的表之间的关系。
8. WHEN 子句指定了一个或多个字段条件。在 SQL*Loader 加载数据之前，每条记录都必须符合这些条件。在此示例中，仅当第 57 个字符是小数点时，SQL*Loader 才会加载这条记录。该小数点分隔字段中的元和分，而且在 SAL 没有值的情况下将导致该记录被拒绝。
9. TRAILING NULLCOLS 子句指示 SQL*Loader 将任何处于相对位置的列（即不在该记录中的列）作为空列处理。
10. 控制文件的剩余部分包含字段列表，该列表提供了所要加载的表中的列格式的有关信息。

SQL*Loader 控制文件（续）

使用 “OEM 控制台” (OEM Console) 执行加载操作（使用 “加载向导”）

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开要在其中加载数据的那个表。
3. 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
4. 此时，就会显示加载向导的 “简介” (Introduction)页。
5. 选择 “下一步” (Next)开始进行加载。
   注：本课中的其余幻灯片讨论了 “加载向导” 中所需的信息，如下所示。
6. **使用 “控制文件” (Control File) 页：指定控制文件在数据库服务器计算机上的完整路径和名称。**
7. 选择 “下一步” (Next)。
8. 使用 “数据文件” (Data File) 页
9. 使用 “加载方法” (Load Method) 页定义加载方法
10. 使用 “加载方法” (Load Method) 页定义可选的文件
11. 使用 “调度” (Schedule) 页
12. 使用 “作业信息” (Job Information) 页
13. 使用 “概要” (Summary) 页

与控制文件的语法有关的注意事项

- 控制文件的语法在格式方面没有任何限制。
- 语法不区分大小写。
- 注释从两个标记注释开始处的连字符 (--) 开始一直延续到行尾。
- 保留 CONSTANT 关键字。

ORACLE

与控制文件的语法有关的注意事项

- 控制文件的语法在格式方面没有任何限制（语句可以长达数行）。
- 它不区分大小写。但是，单引号或双引号内的字符串必须完全保留原样（包括大小写在内）。
- 在控制文件的语法中，注释从两个标记注释开始处的连字符 (--) 开始一直延续到行尾。控制文件的第三部分（可选）被认为是数据（而非控制文件的语法），因此，注释在此部分中不受支持。
- 对 SQL*Loader 来说，CONSTANT 关键字具有特殊含义，因此将该关键字加以保留。为避免出现潜在的冲突，不应将 CONSTANT 一词用作任何表或列的名称。

输入数据和数据文件

- SQL*Loader 从控制文件所指定的一个或多个文件中读取数据。
- 从 SQL*Loader 的角度看，数据文件中的数据是以记录的形式出现的。
- 数据文件可以采用以下三种格式之一：
  - 固定记录格式
  - 可变记录格式
  - 流式记录格式

ORACLE

输入数据和数据文件

固定记录格式：

如果数据文件中的所有记录具有相同的字节长度，则文件采用的是固定记录格式。虽然这种格式最为死板，但是它所带来的性能比可变记录格式或流格记录格式都好。此外，固定记录格式还很易于指定。例如：

```
INFILE <datafile_name> "fix n"
```

在此示例中，SQL*Loader 应该认为特定数据文件采用的是固定记录格式，其中每条记录都具有 *n* 个字节。

以下示例显示的是一个指定让数据文件采用固定记录格式的控制文件。该数据文件包含 4 条物理记录。第一条记录是[0001, abcd]，其中正好包括 9 个字节（使用单字节字符集），回车符是第 10 个字节。

```
load data
infile 'example.dat' "fix 10"
into table example
fields terminated by ','
(col1, col2)
example.dat:
0001,abcd
0002,fghi
0003,klmn
```

输入数据和数据文件（续）

可变记录格式：

如果数据文件中每条记录的开头都包含了该记录在字符字段中的长度，则文件所采用的可变记录格式。这种格式与固定记录格式相比具有更大的灵活性，与流式记录格式相比则能带来更高的性能。例如，可以按以下方法指定一个被认为是采用了可变记录格式的数据文件：

```
INFILE "datafile_name" "var n"
```

在此示例中，*n* 指定记录长度字段中的字节数。如果未指定 *n*，则 SQL*Loader 假定记录长度为 5 个字节。如果指定的 *n* 大于 40，就会出现错误。以下示例显示的是一个控制文件说明，它指示 SQL*Loader 在 example.dat 数据文件中查找数据，并且向 SQL*Loader 指明该文件采用的格式为可变记录格式，其中的记录长度字段包括 3 个字节。example.dat 数据文件包含 3 条物理记录。其中，第 1 条记录的长度被指定为 009（即 9）个字节，第 2 条记录的长度为 010 个字节（包括由一个字符组成的新行），第 3 条记录的长度为 012 个字节。此示例还假定该数据文件使用单字节字符集。

```
load data
infile ’example.dat’ "var 3“
into table example
fields terminated by ’,’ optionally enclosed by ’"’
(col1 char(5),col2 char(7))
example.dat:
009hello,cd,
010world,im,
012my,name is,
```

流式记录格式：

如果不是按大小来指定记录，而是让 SQL*Loader 通过扫描记录结束符来判断记录的起始和结束，则文件所采用的是流式记录格式。流式记录格式是最灵活的格式，但可能会使性能下降。要使数据文件的格式被认为是流式记录格式，则该数据文件的说明将类似以下语句：

```
INFILE <datafile_name> ["str terminator_string"]
```

将 terminator_string 指定为’char_string’或 X’hex_string’，其中：

’char_string’ 是用一个用单引号或双引号引起来的字符串

X’hex_string’ 是一个采用十六进制格式的字节字符串

输入数据和数据文件（续）

如果 terminator_string 中包含特殊字符（即不可打印的字符），应该将它指定为 X’hex_string’。但是，通过使用反斜杠，可以将某些不可打印的字符指定为 (’char_string’)。例如：

\n 换行符（新行）

\t 横向制表符

\f 换页符

\v 纵向制表符

\r 回车符

如果通过 NLS_LANG 参数给会话指定的字符集与数据文件的字符集不相同，则将字符串转换为数据文件的字符集。

十六进制字符串应该在数据文件的字符集范围之内，因此不会进行转换。如果未指定 terminator_string，则它将采用缺省值：新行（行尾）符。在基于 UNIX 的平台上，新行符为换行符，而在 Microsoft 平台上则为回车符加上换行符。新行符将放在数据文件的字符集后面。

以下示例说明了如何加载采用流式记录格式的数据，在此格式中将使用字符串 ’|\n’ 来指定结束符字符串。通过在该字符串中使用反斜杠字符，可指定不可打印的换行符。

```
load data
infile ’example.dat’ "str ’|\n’“
into table example
fields terminated by ’,’ optionally enclosed by ’"’
(col1 char(5),
col2 char(7))
example.dat:
hello,world,|
james,bond,|
```

输入数据和数据文件（续）

使用 Oracle Enterprise Manager 执行加载操作（使用“加载向导”）

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开要在其中加载数据的那个表。
3. 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
4. 此时，就会显示加载向导的 “简介” (Introduction) 页。
5. 选择 “下一步” (Next) 开始进行加载。注：本课中的其余幻灯片讨论了 “加载向导” 中所需的信息。
6. 使用 “控制文件” (Control File) 页：指定控制文件在数据库服务器计算机上的完整路径和名称。选择 “下一步” (Next)。
7. **使用 “数据文件” (Data File) 页：**
   此页可帮助您确定如何来指定包含数据的文件。可以选择以下方法之一：
   - **在控制文件（在上文中已确定）中指定数据。**
   - **或者提供数据文件所在的数据库服务器计算机的完整路径。如果使用此方法，请注意在 Unix 环境中是要区分大小写的。**
8. 使用 “加载方法” (Load Method) 页定义加载方法
9. 使用 “加载方法” (Load Method) 页定义可选的文件
10. 使用 “调度” (Schedule) 页
11. 使用 “作业信息” (Job Information) 页
12. 使用 “概要” (Summary) 页

逻辑记录

以下是两种用来形成逻辑记录的策略。可以指示 SQL*Loader 采用其中一种策略：

- **将固定数量的物理记录合并为一条逻辑记录。**
- **当某种条件为真时，将物理记录合并为逻辑记录。**

ORACLE

逻辑记录

SQL*Loader 按照指定的记录格式将输入数据以物理记录的形式出现。缺省情况下，一条物理记录就是一条逻辑记录。但是为了增大灵活性，可以指示 SQL*Loader 将大量的物理记录合并为一条逻辑记录。SQL*Loader 可以使用以下两种方法之一完成该操作：

- 将固定数量的物理记录合并为一条逻辑记录
- 当某种条件为真时，将物理记录合并为逻辑记录

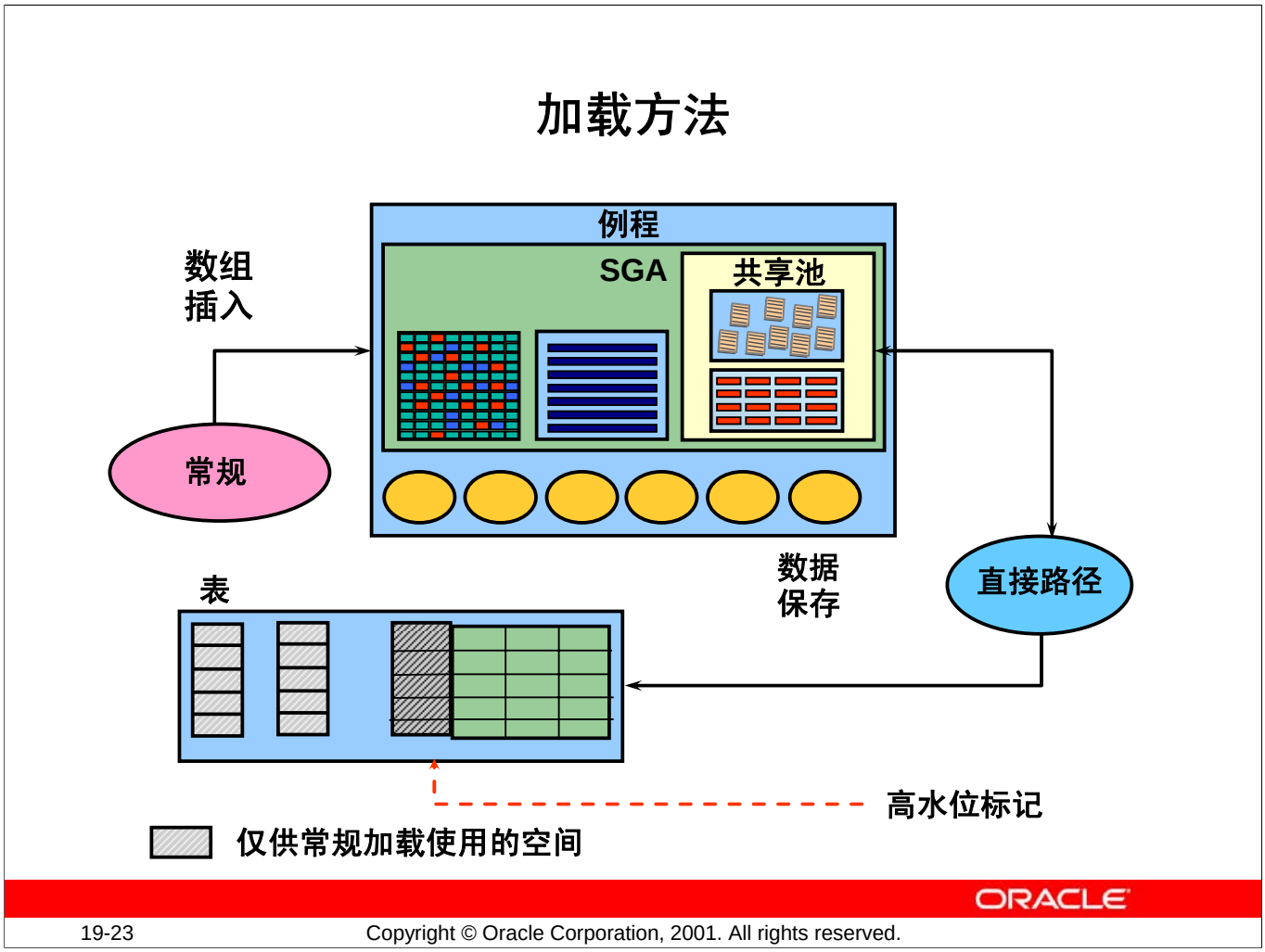
使用 CONCATENATE 来汇集逻辑记录：

如果 SQL*Loader 应该始终将相同数量的物理记录组合成一条逻辑记录，则使用 CONCATENATE。以下是 CONCATENATE 的一个使用示例。在该示例中，integer 用于指定所要合并的物理记录的数量：

```
CONCATENATE integer
```

使用 CONTINUEIF 来汇集逻辑记录：

如果所要合并的物理记录的数量是变化的，则必须使用 CONTINUEIF。在 CONTINUEIF 关键字后面跟有一个条件。读取每条物理记录时，都会对该条件进行求值。例如，如果第 1 条记录在字符位置 80 处有一个英镑符号 (#)，则有可能将两条记录合并。如果该字符位置上是任何其它字符，则不将第 2 条记录添加到第 1 条记录上。



加载方法

SQL*Loader 提供两种加载数据的方法：

- 常规路径
- 直接路径

常规路径加载：

常规路径加载将需要插入的行排成一个数组，并使用 SQL INSERT 语句加载数据。在常规路径加载期间，将基于字段说明来分析输入记录，而且将记录排成一个数组并将其插入到控制文件所指定的表中。不符合字段说明的记录将被拒绝，而不满足选择标准的记录则将被废弃。

可通过常规路径加载将数据加载到集簇表及非集簇表中。至于重做日志的生成情况，则受到所加载表的记录属性的控制。

加载方法（续）

直接路径加载：

直接路径加载在内存中建立数据块，并将这些块直接保存到为正在进行加载的表分配的区内。仅当数据库处于 ARCHIVELOG 模式时，才会生成重做日志条目。直接路径加载使用字段说明建立全部的 Oracle 数据块，并将这些块直接写入 Oracle 数据文件中。直接路径加载绕过数据库缓冲区高速缓存，而且仅在需要管理区和调整高水位标记时才会访问 SGA。

直接路径加载通常比常规路径加载速度快，但它不能应付所有情况。在下一部分中，将对常规路径加载与直接路径加载进行比较，同时还对每种加载方法的使用情况进行了举例。

注：Oracle 所提供的脚本 `catldr.sql` 可创建供直接路径加载使用的视图。该脚本将在运行 `catalog.sql` 脚本时自动调用。

加载方法（续）

使用 Oracle Enterprise Manager 执行加载操作（使用“加载向导”）

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开要在其中加载数据的那个表。
3. 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
4. 此时，就会显示加载向导的 “简介” (Introduction) 页。
5. 选择 “下一步” (Next) 开始进行加载。注：本课中的其余幻灯片讨论了 “加载向导” 中所需的信息。
6. 使用 “控制文件” (Control File) 页：指定控制文件在数据库服务器计算机上的完整路径和名称。选择 “下一步” (Next)。
7. 使用 “数据文件” (Data File) 页：此页可帮助您确定如何来指定包含数据的文件。可以选择以下方法之一：
   - 在控制文件（在上文中已确定）中指定数据。
   - 或者提供数据文件所在的数据库服务器计算机的完整路径。如果使用此方法，请注意在 Unix 环境中是要区分大小写的。
8. **使用 “加载方法” (Load Method) 页定义加载方法：**
   此页可定义您要使用的加载方法。可以选择以下方法之一：
   - **常规路径**
   - **直接加载**
   - **并行直接加载**
9. 使用 “加载方法” (Load Method) 页定义可选的文件
10. 使用 “调度” (Schedule) 页
11. 使用 “作业信息” (Job Information) 页
12. 使用 “概要” (Summary) 页

直接路径加载和常规路径加载的比较

常规路径加载	直接路径加载
使用 COMMIT 来使所做的更改能永久保留	使用数据保存
始终生成重做日志条目	仅在特定条件下才生成重做日志条目
执行所有的约束	仅执行主键约束、唯一性约束和 NOT NULL 约束
触发 INSERT 触发器	不触发 INSERT 触发器
可以加载到集簇表中	无法加载到集簇表中
其他用户可以对表进行更改	其他用户不能对表进行更改

直接路径加载和常规路径加载的比较

保存数据的方法：

常规路径加载使用 SQL 处理及数据库 COMMIT 来保存数据。在插入记录数组后将执行提交操作。每次数据加载都可能涉及多个事务处理。

直接路径加载使用数据保存来将数据块写入到 Oracle 数据文件中。数据保存与 COMMIT 之间存在以下区别：

- 数据保存期间，只将写满的数据库块写入数据库。
- 块写在表的高水位标记之后。
- 数据保存之后，高水位标记移动。
- 数据保存之后不释放内部资源。
- 数据保存不结束事务处理。
- 每次数据保存时不更新索引。

直接路径加载和常规路径加载的比较（续）

记录所进行的更改：

常规路径加载就像任何 DML 语句一样生成重做日志条目。如果使用直接路径加载，则在下列情况下将不生成重做日志条目：

- 数据库为 NOARCHIVELOG 模式
- 数据库为 ARCHIVELOG 模式，但禁用事件记录。通过为表设置 NOLOGGING 属性或在控制文件中使用 UNRECOVERABLE 子句，可禁用事件记录。

执行约束：

常规路径加载期间，与在任何 DML 操作期间一样，强制执行所有启用的约束。

在直接路径加载期间，对约束进行如下处理：

- 建立数组时检查 NOT NULL 约束。
- 禁用外键约束和 CHECK 约束，并可在运行结束时通过使用控制文件中的有关命令来启用它们。禁用外键约束的原因在于：这些约束引用其它行或表；禁用 CHECK 约束则是因为这些约束可能使用 SQL 函数。如果只要将少数几行插入到一个大表中，可使用常规加载。
- 运行期间和运行结束时检查主键约束和唯一性约束，如果违反，则可能禁用。

触发 INSERT 触发器：

虽然在常规加载期间会触发 INSERT 触发器，但它们在进行直接路径加载之前被禁用，并在运行结束时重新启用。如果某个引用对象在运行结束时无法访问，说明这些触发器可能仍被禁用。请考虑使用常规路径加载，以通过 INSERT 触发器将数据加载到表中。

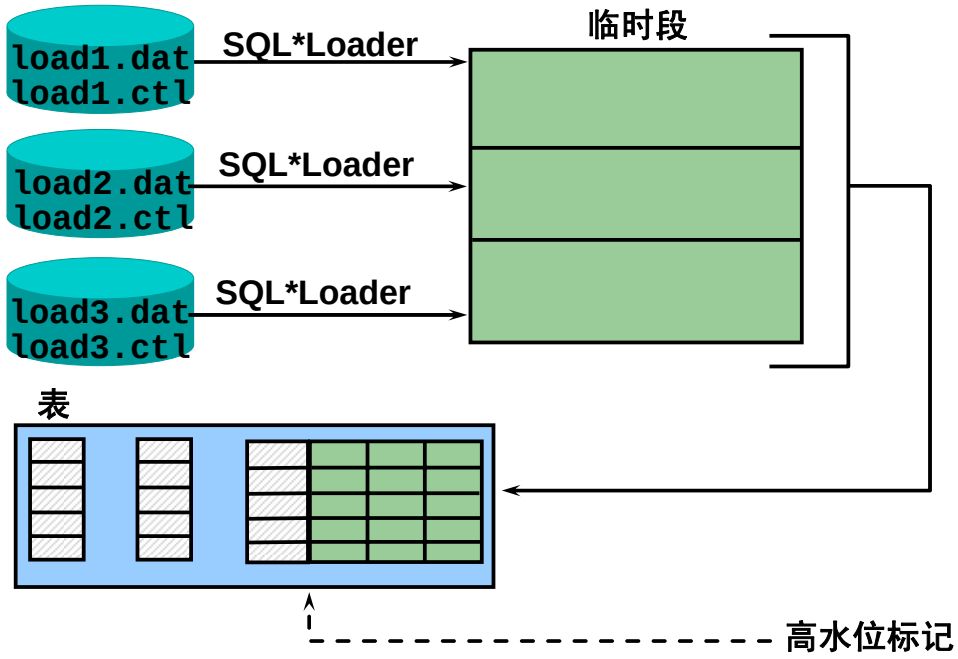
加载到集簇表中：

不能通过直接加载来将行加载到集簇表中。只能使用常规路径加载来加载集簇表。

锁定：

在直接加载期间，其它事务处理不能对正被加载的表进行更改。不过，该规则存在一种例外情况，即在并发使用多个并行直接加载会话时可以对正被加载的表进行更改。

并行直接路径加载



并行直接路径加载

使用多个 SQL*Loader 会话可提高直接路径加载的性能。有三种并发模式可用于最大限度地缩短数据加载所需的时间：

- 并行常规路径加载
- 在直接路径加载方法中使用段间并发
- 在直接路径加载方法中使用段内并发

并发常规路径：

如果触发器或完整性约束出现问题，但又希望获得更快的加载速度，则应该考虑使用多个并发的常规路径加载。请在具有多个 CPU 的系统上使用多个并发执行的加载会话。在逻辑记录边界上将输入数据文件拆分为多个单独的文件，然后使用常规路径加载会话分别加载拆分得到的输入数据文件。

段间并发：

段间并发可用于并发加载不同的对象。在直接路径加载中，这种技术可用于并发加载不同的表，或者并发加载同一表中的不同分区。

段内并发：

并行直接路径加载允许多个直接路径加载会话将数据并发加载到同一表中，或者加载到某个允许进行段内并发操作的分区表的同一分区中。

数据转换

在常规路径加载过程中，分两步将数据文件中的数据字段转换为数据库中的列：

- 控制文件中的字段说明用于解释数据文件的格式，并使用该数据将数据文件转换为 SQL INSERT 语句。
- Oracle 数据库服务器接受该数据，并执行 INSERT 语句将数据存储到数据库中。

ORACLE

被废弃或拒绝的记录

- 坏文件
  - 如果输入格式无效，SQL*Loader 将拒绝记录。
  - 如果 Oracle 数据库发现某行无效，则将拒绝该条记录，并且 SQL*Loader 将该记录放到坏文件中。
- 废弃文件
  - 该文件只有在启用之后方能使用。
  - 该文件包含被过滤出来的记录，因为这些记录与控制文件中指定的任何记录选择标准均不符合。

ORACLE

被废弃或拒绝的记录

坏文件：

坏文件包含被 SQL*Loader 或 Oracle 数据库拒绝的记录。

SQL*Loader 所拒绝的记录：

如果输入格式无效，SQL*Loader 将拒绝记录。例如，如果记录的右引号分隔符丢失或者其中某个受限制的字段超过其最大长度，则 SQL*Loader 将拒绝该记录。该遭拒绝的记录将被置入坏文件中。

Oracle 所拒绝的记录：

在 SQL*Loader 接受一条记录以进行处理后，就会向 Oracle 发送一行以进行插入。如果 Oracle 确定该行有效，就会将该行插入到数据库中。否则，将拒绝该记录，并且 SQL*Loader 将该记录放到坏文件中。如果出现以下情况，该行就可能被拒绝：某个键不是唯一的、某个所需的字段为空，或者字段中包含的数据对于 Oracle 来说属于无效的数据类型。

被废弃或拒绝的记录（续）

废弃文件：

执行 SQL*Loader 时，它可能会创建一个称为 “废弃文件” 的文件。仅在确实需要废弃文件、并且您已指定应该启用废弃文件时才创建该文件。废弃文件包含在加载过程中过滤出来的记录，因为这些记录与控制文件中指定的任何记录选择标准均不符合。因此，废弃文件包含那些未插入到数据库任何表中的记录。可以指定废弃文件所能接受的此类记录的最大数量。

被废弃或拒绝的记录（续）

使用 Oracle Enterprise Manager 执行加载操作（使用“加载向导”）

从 “OEM 控制台” (OEM Console)：

1. 导航到 “数据库” (Databases) > “方案” (Schema) > “表” (Table)。
2. 展开要在其中加载数据的那个表。
3. 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
4. 此时，就会显示加载向导的 “简介” (Introduction) 页。
5. 选择 “下一步” (Next) 开始进行加载。
   注：本课中的其余幻灯片讨论了 “加载向导” 中所需的信息。
6. 使用 “控制文件” (Control File) 页：指定控制文件在数据库服务器计算机上的完整路径和名称。选择 “下一步” (Next)。
7. 使用 “数据文件” (Data File) 页：此页可帮助您确定如何来指定包含数据的文件。可以选择以下方法之一：
   - 在控制文件（在上文中已确定）中指定数据。
   - 或者提供数据文件所在的数据库服务器计算机的完整路径。如果使用此方法，请注意在 Unix 环境中是要区分大小写的。
8. **使用 “加载方法” (Load Method) 页定义加载方法：**
   此页可定义您要使用的加载方法。可以选择以下方法之一：
   - **常规路径**
   - **直接加载**
   - **并行直接加载**

   在 “加载向导” 中提供了上述信息之后，就可开始将加载作为一项 OEM 作业来运行了。
9. 使用 “加载方法” (Load Method) 页定义可选的文件：还可以通过此页来定义其它的加载要求、优化选项和可选文件。若要为坏文件、废弃文件和日志文件定义路径，请执行以下操作：
   - 在此页上选择 “高级” (Advanced) 按钮
   - 选择 “可选文件” (Optional Files) 页
   - 选择生成以上文件中的任何一个文件，并为每个选定文件提供完整的路径。
   - 选择 “确定” (OK)。

     在 “加载向导” 中提供了上述信息之后，就可开始将加载作为一项 OEM 作业来运行了。

被废弃或拒绝的记录（续）

使用 Oracle Enterprise Manager 执行加载操作（使用“加载向导”）（续）

注：在 “加载向导” 中提供了上述信息之后，就可开始将加载作为一项 OEM 作业来运行了。

10. 使用 “调度” (Schedule) 页：此页指定何时运行该作业。
11. 使用 “作业信息” (Job Information) 页：此页指定作业的具体名称以及运行该作业的确切目的。此页还用于覆盖任何首选的身份证明。最好此时定义方案的所有者。
12. 使用 “概要” (Summary) 页：此页对您所提供的加载向导页信息进行总结。可以在此页上检查这些信息，如果需要进行任何更改，还可以回退到相应的页。如果所有信息均正确无误，请选择 “确定” (OK) 并提交该作业。
13. 导航并在树上选择 “作业” (Jobs)。
14. 检查 “历史记录” (History) 页以查看作业。

日志文件的内容

- 标头信息
- 全局信息
- 表信息
- 数据文件信息
- 表加载信息
- 小结统计信息
- 直接路径加载的其它统计信息和多线程信息

ORACLE

日志文件的内容

“标头信息” 部分包含以下条目：

- 运行日期
- 软件版本号

“全局信息” 部分包含以下条目：

- 所有输入/输出文件的名称
- 命令行参数的回显
- 续行符说明

“表信息” 部分为所加载的每个表提供以下条目：

- 表名
- 加载条件（如有）。也即，是加载所有的记录，还是仅加载那些符合 WHEN 子句标准的记录。
- INSERT、APPEND 或 REPLACE 说明
- 以下列信息：
  - 位置、长度、数据类型和分隔符（如果能在数据文件中找到）
  - RECNUM、EQUENCE、CONSTANT 或 EXPRESSION（如果指定）
  - DEFAULTIF 或 NULLIF（如果指定）

日志文件的内容（续）

如果 SQL*Loader 控制文件包含任何可用于加载日期时间或时间间隔等数据类型的指令，则日志文件将在数据类型标题下面包含 DATETIME 或 INTERVAL 关键字。如果适用的话，还将在 DATETIME 或 INTERVAL 关键字后面加上相应的标记。

仅当数据文件中的数据有错误时，才会出现 “数据文件信息” 部分。该部分提供以下条目：

- SQL*Loader 和 Oracle 数据记录错误
- 被废弃的记录

“表加载信息” 部分为所加载的每个表提供以下条目：

- 加载的行数
- 有资格加载但由于数据错误而被拒绝的行数
- 由于未通过 WHEN 子句测试而被废弃的行数
- 相关字段均为空的行数

“小结统计信息” 部分显示以下数据：

- 占用的空间量：
  - 用于绑定数组（实际使用量基于已指定的 BINDSIZE）
  - 用于其它开销（始终需要，而与 BINDSIZE 无关）
- 累计得到的加载统计信息；即对于所有的数据文件，所跳过、读取或拒绝的记录数加载表时，将记录以下统计信息：
- 如果对分区表进行直接路径加载，将报告每个分区的统计信息。
- 常规路径加载无法报告每个分区的统计信息。

如果没有启用介质恢复，则不对加载进行记录。即，如果禁用介质恢复，将忽略记录操作请求。

SQL*Loader 原则

- 使用参数文件来指定常用的命令行选项。
- 仅当运行的是小型、一次性加载时才将数据放在控制文件中。
- 提高性能的方法：
  - 分配充足的空间
  - 将数据按最大索引排序
  - （如果运行的是并行加载）为临时段指定不同的文件

ORACLE

SQL*Loader 原则

使用 SQL*Loader 时应遵循下列原则，以使错误减到最少并提高性能：

- 使用参数文件来指定常用的命令行选项。例如，如果每周都要将数据加载到数据仓库中，则除文件名外，其它所有选项都可以保持不变。
- 将控制文件与数据文件分开，这样可以在多个加载会话中反复使用控制文件。
- 基于预期的数据量预先分配好空间，以免在加载期间动态分配区，从而可提高加载速度。
- 对于直接加载，将使用临时段来生成新数据的索引。加载结束时，这些索引将与现有索引合并。通过按最大索引的关键字对输入数据进行排序，可以最大限度地减少排序所使用的空间。
- 在并行直接加载中，可指定用于插入数据的临时段的位置。对于每个加载会话，可指定不同的数据库文件以获得最佳性能。

小结

在这一课中，您应该能够掌握：

- 说明 SQL*Loader 的用法
- 执行基本的 SQL*Loader 操作
- 熟练使用直接加载操作
- 列出 SQL*Loader 操作和直接加载操作的使用原则

ORACLE

练习 19 概览

此练习涉及以下主题：

- 使用 SQL*Loader 恢复数据
  - 使用控制文件
  - 使用数据文件
- 使用直接加载来加载数据

ORACLE

练习 19 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 19：将数据加载到数据库中

1 a 查看加载时所要使用的数据文件，以便熟悉控制文件和数据文件的格式。

- 如果使用 SQL*Plus，请查看以下文件：

  `lcase1.ctl`、`lcase2.ctl` 和 `lcase2.dat`

- 如果使用 Oracle Enterprise Manager，请查看以下文件：

  `OEMsqlcase1.ctl`、`OEMsqlcase2.ctl` 和 `OEMsqlcase2`。

b 以用户 `HR` 的身份运行 `lab19_01.sql` 脚本，来创建 `DEPARTMENTS2` 表。

2 a 使用常规路径方法运行 `SQL*Loader`，以将数据加载到 `DEPARTMENTS2` 表中。请使用以下控制文件：

- `lcase1.ctl`（如果使用 SQL*Plus）
- `OEMsqlcase1.ctl`（如果使用 Oracle Enterprise Manager）

文件位于 LABS 目录中。

注：此次运行使用了包含有输入数据的控制文件。

b 使用操作系统命令查看日志文件。

c 查询 `DEPARTMENTS2` 表以检查数据。

3 删除 `DEPTARTMENTS2` 表中的所有记录。

4 a 采用直接路径模式运行 SQL*Loader，以将数据加载到 `DEPARTMENTS2` 表中。请使用以下控制文件：

- `lcase2.ctl`（如果使用 SQL*Plus）
- `OEMsqlcase2.ctl`（如果使用 Oracle Enterprise Manager）

文件位于 `LABS` 目录中。

注：此次运行通过使用输入数据文件来加载数据。

b 使用操作系统命令查看日志文件。

c 查询 `DEPARTMENTS2` 表以检查数据。

练习 19：将数据加载到数据库中

5 a 以用户 HR 的身份从 EMPLOYEES 表中选择，来创建 EMPLOYEES2 表。截断 EMPLOYEES2 表，然后从该表中选择，以验证该表中没有数据。

b 执行从 EMPLOYEES 表到 EMPLOYEES2 表的直接加载插入，并查询 EMPLOYEES2 以验证加载结果。

6 a 再次截断 EMPLOYEES2 表，然后从 EMPLOYEES2 表中选择，以验证该表中没有数据。

b 从 EMPLOYEES 表执行一次并行直接加载插入，以恢复数据。将并行度指定为 2. 验证数据。

20

使用“全球化支持”

ORACLE

目标

完成这一课的学习后，您应该能达到下列目标：

- 为数据库选择数据库字符集和国家字符集
- 使用初始化参数、环境变量以及 ALTER SESSION 命令，指定与语言有关的行为
- 使用不同类型的国家语言支持 (NLS) 参数
- 解释全球化支持对语言相关的应用程序行为的影响
- 获取有关 “全球化支持” 用法的信息

“全球化支持” 功能

- 语言支持
- 地域支持
- 字符集支持
- 文字排序
- 消息支持
- 日期格式和时间格式
- 数字格式
- 货币格式

ORACLE

“全球化支持” 功能

“全球化支持” 功能可确保数据库实用程序和错误消息、排序顺序、日期、时间、货币、数字和日历惯例能自动适合本国语言。

Oracle 目前支持 57 种语言、88 个地域、84 种文字排序（71 种单语言和 13 种多语言）以及 235 种编码字符集。

语言相关操作由客户端和服务器端的若干参数及环境变量控制。

服务器和客户机可能在相同位置或不同位置运行。当客户机和服务器使用不同的字符集时，Oracle 服务器会自动处理字符集的转换。

“全球化支持” 功能（续）

- 用户可以用他们本国的语言交互、存储、处理和检索数据，这些语言包括西欧、东欧、中东、东亚和东南亚语言。
- 不同的国家和地域有不同的文化习俗，直接影响到数据格式。
- 支持许多不同的字符编码方案，包括单字节、多字节和固定宽度的编码字符集。
- Oracle 服务器提供了多种在语言学上精确划分的不同语言类别。
- 数据库实用程序和错误消息以受支持的国家语言形式显示。Oracle 产品已经翻译成 30 种不同的语言。
- 根据国际标准化组织 (ISO) 的规定，日期和时间可以采用零点几秒、秒、分钟、小时、日、月和年这样的格式表示。可以使用时区以支持夏时制。
- 支持诸如公历、日本历、中国的农历和泰国的佛历等历法的本国日历。
- 数字数据以相应的本地格式表示。
- 货币符号体现本国经济制度和 ISO 的规定。借贷符号也因地域而异。

编码方案

Oracle 支持不同类别的字符编码方案：

- **单字节字符集**
  - **7 位**
  - **8 位**
- **宽度可变的多字节字符集**
- **宽度固定的多字节字符集**
- **Unicode (AL32UFT8, AL16UTF16, UTF8)**

ORACLE

编码方案

字符编码方案指定对应于计算机或终端可以显示和接收的字符的数字代码。

字符编码方案用于将数据解释为有意义的符号并从终端传递到主机。

Oracle 提供不同类别的编码方案：

- 单字节
- 宽度可变
- 宽度固定
- Unicode

单字节字符集：

在单字节字符集中，每个字符只占一个字节。单字节 7 位编码方案最多可以定义 128 (2^7) 个字符；单字节 8 位编码方案最多可以定义 256 (2^8) 个字符。

编码方案（续）

单字节方案示例：

7 位字符集：美国 7 位 ASCII 码 (US7ASCII)

8 位字符集：

- 西欧 ISO 8859-1 码 (WE8ISO8859P1)
- 西欧 8 位 EBCDIC 代码页 500 码 (WE8EBCDIC500)
- 西欧 8 位 DEC 码 (WE8DEC)

宽度可变的多字节字符集：

在宽度可变的多字节字符集中，每个字符以一个或多个字节表示。多字节字符集通常用于支持亚洲语言。有些多字节编码方案使用最有效的位值来表明，一个字节是表示单个字节还是作为代表一个字符的一系列字节中的一部分。然而，其它字符编码方案可以区分单字节和多字节字符。在碰到移入代码之前，由设备发出的移出控制代码表明后面的字节都是双字节字符。

宽度可变的多字节方案示例：

- 日文扩展 UNIX 代码 (JEUC)
- 中文 GB2312-80 (CGB2312-80)
- AL32UTF8 (UTF-8)

宽度固定的多字节字符集：

除了每个字符采用字节数固定的格式外，宽度固定的多字节字符集同多字节字符集提供的支持类似。

这提供了每个字符具有统一字节长度表示法的好处。

Oracle 仅支持一个宽度固定的多字节字符集，且该字符集只位于国家字符集 AL16UTF16 中。

宽度固定的多字节字符集示例：

AL16UTF16、16 位 Unicode（宽度固定的双字节 Unicode）

编码方案（续）

Unicode 字符集：

Unicode 是一种全球字符编码标准，可以表示计算机中使用的所有字符，包括技术符号和出版用的字符。Unicode 标准 3.0 版包含 49,149 个字符，容量超过一百多万个字符。

Unicode 全套字符可以用不同的编码格式表示。UTF-16（通用字符集转换格式）是一种宽度固定的双字节格式；而 UTF-8 是一种宽度可变的多字节格式。

Oracle 提供 AL32UTF8、UTF8 和 UTFE 作为数据库字符集，同时提供 AL16UTF16 和 UTF8 作为国家字符集。基于 UTF-8 的字符集的优点是它们包括使用相同单字节编码的 ASCII。UTF8 是 ASCII 的超集，因此，从基于 ASCII 的字符集升级到 Unicode 时，数据库字符集的移植会变得更加简单。

注：有关上述带有连字符的 UTF-16 和 UTF-8 的信息，请参考 Unicode 标准编码；有关不带连字符的 UTF8、AL32UTF8 和 AL16UTF16 的信息，请参考基于 Unicode 标准的 Oracle 字符集。

数据库字符集和国家字符集

数据库字符集	国家字符集
在创建时定义	在创建时定义
除非重新创建，否则无法更改	除非重新创建，否则无法更改，例外情况很少
存储类型为 CHAR、 VARCHAR2、CLOB、 LONG 的数据列	存储类型为 NCHAR、NVARCHAR2、NCLOB 的数据列
可以存储宽度可变的字符集	可以采用 AL16UTF16 或 UTF8 格式存储 Unicode

数据库字符集和国家字符集

字符集类型：

CREATE DATABASE 语句包含 CHARACTER SET 子句和附加可选子句 NATIONAL CHARACTER SET，用于声明要用作数据库字符集和国家字符集的字符集。如果未设置 NATIONAL CHARACTER SET 子句，则国家字符集默认为 AL16UTF16。

由于数据库字符集用于识别并保存 SQL 和 PL/SQL 源代码，它必须将 EBCDIC 码或 7 位 ASCII 码作为子集，判断方法是看哪一个是所用平台的本机码。因此，不可能使用宽度固定的多字节字符集作为数据库字符集，而只能将其作为国家字符集。

在 Oracle9*i* 中，国家字符集只能存储 Unicode，SQL NCHAR 数据类型（NCHAR、NVARCHAR2 和 NCLOB）为 Unicode 数据类型。

选择 Oracle 数据库字符集的原则

考虑事项

- **数据库必须支持哪些语言？**
- **有哪些互操作与系统资源和应用程序相关？**
- **对性能的要求如何？**
- **有哪些限制条件？**

ORACLE

选择 Oracle 数据库字符集的原则

数据库必须支持哪些语言？

可能有几种字符集都可以满足您当前的语言要求，但是，您还应该考虑到将来的需求。如果您知道将来需要扩展对不同语言的支持，现在就要选择一个使用范围较广的字符集，以免日后再进行移植。

有哪些互操作与系统资源和应用程序相关？

尽管数据库可以维护和处理实际字符数据，但是您还必须依赖操作系统中的其它资源。例如，操作系统提供与所选字符集相对应的字体。另外，支持所需语言和应用程序软件的输入法也必须与特定字符集保持一致。

如果所选的字符集与操作系统中的可用字符集不同，Oracle 会将操作系统字符集转换为数据库字符集。但是，这样做会产生一些字符集转换开销，同时您应确保操作系统字符集中包含等价的全套字符，以避免可能出现的数据丢失情况。

选择 Oracle 数据库字符集的原则（续）

对性能的要求如何？

在处理不同的编码方案时，性能开销可能会有所不同，这取决于所选的字符集。为了获得最佳性能，您应该尝试选择一个无需进行转换的字符集，并将对所需语言使用最有效的编码方案。单字节字符集在性能方面要优于多字节字符集，此外，在空间需求方面，该字符集也是最有效的。但是，单字节字符集对所能使用的语言数目有所限制。

有哪些限制条件？

您不能选择宽度固定的多字节数据库字符集。

在 EBCDIC 平台上，不能将基于 ASCII 的字符集设置为数据库字符集；同样，在 ASCII 平台上也不能具有基于 EBCDIC 的数据库字符集。

选择 Oracle 国家字符集的原则

- 两种选择
  - AL16UTF16
  - UTF8
- 是否需要考虑空间问题？
- 是否需要考虑性能问题？

ORACLE

选择 Oracle 国家字符集的原则

AL16UTF16 和 UTF8 这两种选择都适用于国家字符集。AL16UFT16 是宽度固定的双字节 Unicode 字符集。而 UTF8 是宽度可变的、一至三个字节的 Unicode 字符集。欧洲字符在 UTF8 中按一至两个字节存储，而在 AL16UTF16 中按两个字节存储，相比之下，前一种选择可以节省空间。亚洲字符在 UTF8 中按三个字节存储，这样，所需的空间比在 AL16UTF16 中要多。

由于 AL16UTF16 是宽度固定的编码，因此在执行速度上要比宽度可变的 UTF8 快。

在 Oracle9*i* 中，AL16UTF16 支持最新的 3.0 版的 Unicode 标准。

在 Oracle9*i* 中，UTF8 支持 Unicode 3.0 版，并将在今后的版本中继续支持 Unicode 3.0 版。

选择 Unicode 解决方案：Unicode 数据库

应在何时使用 Unicode 数据库？

- **易于移植 Java 或 PL/SQL 代码**
- **易于移植 ASCII 编码的数据**
- **多语言数据均匀分布**
- **InterMedia 文本搜索**

ORACLE

选择 Unicode 解决方案：Unicode 数据库

易于移植 Java 或 PL/SQL 代码：

在实现多语言支持时，如果是使用现有的 SQL CHAR 数据类型（CHAR、VARCHAR2、CLOB 和 LONG）来存储多语言数据，则 Unicode 数据库可以最大程度地减少代码更改。对于 SQL NCHAR 数据类型，不需要重新编码。

易于移植 ASCII 编码的数据：

如果当前数据库字符集和数据严格符合 US7ASCII 格式，即可以使用简单的 ALTER DATABASE 语句移植数据库。

多语言数据均匀分布：

如果多语言数据分布在整个数据库中，则可选择 Unicode 数据库解决方案，因为您无需了解哪些列存储有多语言数据。

Oracle 文本：

要将多语言 BLOB 和 Oracle 文本一起使用，则必须使用 Unicode 数据库解决方案。

选择 Unicode 解决方案：Unicode 数据类型

何时应使用 Unicode 数据类型？

- 逐渐添加多语言支持时
- 打包应用程序
- 性能
  - 单字节数据库字符集和宽度固定的国家字符集
- 更好地支持 windows 客户端的 UTF-16

ORACLE

选择 Unicode 解决方案：Unicode 数据类型

逐渐添加多语言支持：

要添加 Unicode 支持而不移植数据库，可以在新表和现有表中添加 SQL NCHAR 数据类型。

程序包应用程序：

对打包应用程序使用 SQL NCHAR 数据类型，因为这种数据类型是一种可靠的 Unicode 数据类型，在 Unicode 中，数据始终以这种数据类型存储，并且数据的长度始终按 UTF-16 代码单元指定。因此，您只需要对该应用程序测试一次，它即可在包含任意数据库字符集的客户数据库上运行。

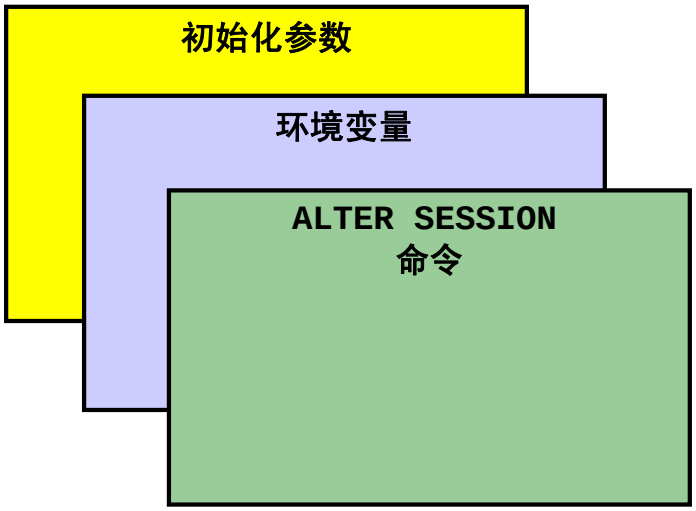
性能：

对于性能问题，请考虑将单字节字符集用于数据库字符集，而将使用 AL16UTF16 的 SQL NCHAR 数据类型用于多语言数据。使用宽度可变的 UTF-8 编码格式时，会产生性能开销。宽度固定的单字节和多字节字符集的执行效率更高。

更好地支持 windows 客户端的 UTF-16：

如果您的应用程序是用运行在 Windows 上的 Visual C/C++ 或 Visual Basic 编写的，则可能希望使用 SQL NCHAR 数据类型，因为以这些数据类型存储 UTF-16 数据的方法，与在 Visual C/C++ 中使用 `wchar_t` 缓冲区以及在 Visual Basic 使用字符串缓冲区存储此类数据的方法相同。由于 `wchar_t` 和字符串数据类型的长度与数据库中的 SQL NCHAR 数据类型的长度匹配，因此可以避免客户端应用程序中的缓冲区溢出。

指定语言相关行为



指定语言相关行为

有三种指定国家语言支持 (NLS) 参数的方法：

- 服务器端的初始化参数，用于指定缺省服务器环境（这些缺省设置对客户端无效。）
- 客户端的环境变量，指定覆盖服务器缺省设置的语言相关行为
- ALTER SESSION 参数，用于覆盖客户机或服务器的缺省设置

指定服务器的语言相关行为

- **NLS_LANGUAGE 指定：**
  - **消息的语言**
  - **日期和月份名称**
  - **用于 A.D.、B.C.、a.m. 和 p.m. 的符号**
  - **缺省排序机制**
- **NLS_TERRITORY 指定：**
  - **天数和周数**
  - **缺省日期格式、十进制字符、组分隔符以及缺省的 ISO 和本地货币符号**

ORACLE

指定服务器的语言相关行为

NLS 初始化参数：

初始化参数 NLS_LANGUAGE 定义语言相关惯例值，例如：

- Oracle 消息所使用的语言
- 日期和月份名称及其缩写所使用的语言
- a.m.、p.m.、A.D. 和 B.C. 的等价语言所使用的符号
- 缺省的字符数据排序顺序

初始化参数 NLS_LANGUAGE 定义地域相关惯例值，其中包括：

- 缺省日期格式
- 十进制字符和组分隔符
- 本地货币符号
- ISO 货币符号
- ISO 周数计算
- 一周起始日

注：当地域名包含空格时，如 The Netherlands，应该用双引号将地域名引起来，如“The Netherlands”。

相关语言和地域缺省值

参数	值
NLS_LANGUAGE NLS_DATE_LANGUAGE NLS_SORT	AMERICAN AMERICAN BINARY
NLS_TERRITORY NLS_CURRENCY NLS_ISO_CURRENCY NLS_DATE_FORMAT NLS_NUMERIC_CHARACTERS	AMERICA $ AMERICA DD-MON-RR ,.

相关语言和地域缺省值

NLS 初始化参数：

初始化参数 NLS_LANGUAGE 确定下列参数的缺省值：

列	说明
NLS_DATE_LANGUAGE	显式更改日期、月份名及其缩写和其它日期格式元素的拼写值所使用的语言。
NLS_SORT	更改 Oracle 服务器用于排序字符值的文字排序顺序（排序值必须是是文字排序顺序名。）

相关语言和地域缺省值（续）

NLS 初始化参数（续）：

NLS_TERRITORY 确定下列参数的缺省值：

列	说明
NLS_CURRENCY	显式指定新的本地货币符号
NLS_ISO_CURRENCY	显式指定应使用 ISO 货币符号的地域
NLS_DATE_FORMAT	显式指定新的缺省日期格式（值必须是日期格式模型。）
NLS_NUMERIC_CHARACTERS	显式指定新的十进制字符和组分隔符

对欧元的双重货币支持：

1999 年 1 月 1 日，欧共体新货币－欧元首次亮相。为了支持这个新的欧共体货币，新增了对特定国家的双重货币支持。初始化参数 NLS_DUAL_CURRENCY 为用户会话设置一个代用货币符号。

对以下国家增加了欧元符号作为双重货币支持：

爱尔兰　　卢森堡
奥地利　　瑞典
比利时　　葡萄牙
丹麦　　西班牙
德国　　希腊
法国　　意大利
芬兰　　英国
荷兰

WE8ISO8859P15 和 MS 代码页 WE8MSWIN1252 等 ISO 字符集已指定了欧元符号的代码点。

指定会话的语言相关行为

- 环境变量：
  NLS_LANG=French_France.UTF8
- 其它环境变量：
  - **NLS_DATE_FORMAT**
  - **NLS_DATE_LANGUAGE**
  - **NLS_SORT**
  - **NLS_NUMERIC_CHARACTERS**
  - **NLS_CURRENCY**
  - **NLS_ISO_CURRENCY**
  - **NLS_CALENDAR**

ORACLE

指定会话的语言相关行为

环境变量 NLS_LANG:

使用环境变量 NLS_LANG 为个别用户指定所需的文化习俗。NLS_LANG 值将覆盖 NLS 初始化参数的所有值。

每一部分控制 NLS 功能的一个子集：

NLS_LANG=<language>_<territory>.<charset>

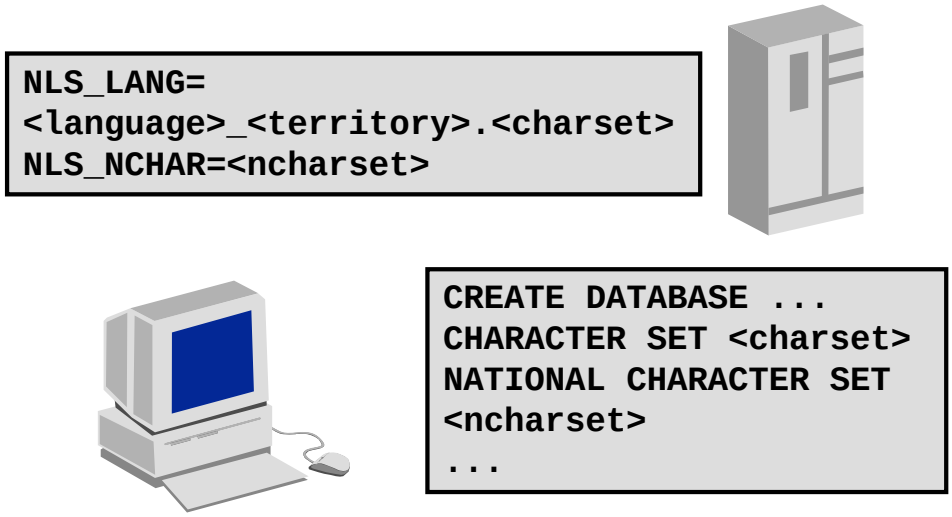
其中：

Language：覆盖 NLS_LANGUAGE 的值并控制与 NLS_LANGUAGE 相同的功能

Territory：覆盖 NLS_TERRITORY 的值并控制与 NLS_TERRITORY 相同的功能

Charset：指定客户端应用程序（通常是用户终端使用的客户端应用程序）使用的字符编码方案

客户机–服务器体系结构中的字符集



指定会话的语言相关行为（续）

环境变量 NLS_LANG（续）：

NLS_LANG 定义客户机终端的字符编码方案。不同客户机可以使用不同的编码方案。在客户机和服务器之间传递的数据将在两套编码方案间自动转换。数据库编码方案应是所有客户机编码方案的超集或与之等价。转换对客户机应用程序是透明的。

如果数据库字符集和客户机字符集相同，Oracle 将假定正在发送或接收的数据属于同一字符集，因此不进行验证或转换。该方案的优点在于能优化性能，但是，如果使用不当，可能会导致数据不一致，如另一字符集中存储的数据不同于数据库字符集中存储的数据。

例如，数据库字符集是 US7ASCII，客户机终端使用的是简体中文 Windows。将 NLS_LANG 设置为 SIMPLIFIED CHINESE_HONGKONG.US7ASCII 以指定客户机字符集，这样，就有可能在单字节数据库中存储多字节简体中文字符。这意味着 Oracle 可以把这些字符当作单字节 US7ASCII 字符使用，因此，所有的 SQL 字符串操作函数（如 SUBSTR 或 LENGTH）将基于字节而不是字符。在导出然后导入到其它数据库后，可能会丢失所有的非 ASCII 字符。

指定会话的语言相关行为（续）

其它环境变量：

所有 NLS 初始化参数都可作为环境变量，从而可以为每个客户机分别指定 NLS 特性。

此外，还可以使用 NLS_CALENDAR 指定 Oracle 服务器使用的日历系统，如公历、波斯历或泰国佛历。

只有在客户环境中才能设置下列变量：

- NLS_CREDIT
- NLS_DEBIT
- NLS_DISPLAY
- NLS_LANG
- NLS_LIST_SEPARATOR
- NLS_MONETARY

注：有关这些参数的说明，请参考 *Oracle9i Globalization Support Manual*。

如果将环境变量 ORA_NLS33 设置为无效目录，则有可能仅能使用缺省字符集 US7ASCII 创建数据库。在 UNIX 上，ORA_NLS33 应该设置为：

```
$ORACLE_HOME/ocommon/nls/admin/data
```

如果未设置 ORA_NLS33，这即是缺省设置。

指定会话的语言相关行为

```
ALTER SESSION SET
NLS_DATE_FORMAT='DD.MM.YYYY';
```

```
DBMS_SESSION.SET_NLS('NLS_DATE_FORMAT',
'''DD.MM.YYYY''') ;
```

ORACLE

指定会话的语言相关行为

更改 NLS 参数：

使用 ALTER SESSION 命令更改会话的个别 NLS 特性。通过发出 ALTER SESSION 命令，可同时更改可在客户端和服务器端设置的所有环境变量。

在上面的示例中，更改了会话的日期格式。

此外，SQL*Plus 等工具还可以读取环境变量并发出相应的 ALTER SESSION 命令。

除了显式发出 ALTER SESSION 命令外，还有一个数据包 DEMS_SESSION.SET_NLS 可获得参数的名称和值。

文字排序

有三种排序类型：

- 二进制排序，即根据编码字符的二进制值进行排序
- 单语言排序
  - 执行两遍排序
  - 基于分配给字符的主值和次值
- 多语言排序
  - 基于新的 ISO 14651 和
  - 支持多语言排序的 Unicode 3.0 标准

ORACLE

文字排序

二进制排序是一种传统排序机制，它依据字符编码所用的二进制值对字母进行排序。对于不同的语言，字符在字母表里的位置可能有所不同。

对于单语言排序，Oracle 在比较按单语言顺序排序的字符串时要执行两遍排序。第一遍排序是比较主表中整个字符串的主值，第二遍排序是比较次表中的次值。通常，具有相同外观的字母的主值也相同。Oracle 根据音调符号和大小写的不同来区分主值相同而次值不同的字母。单语言排序要优于二进制排序，但仍存在限制。

对于多语言排序，Oracle 提供了一种基于 ISO 标准 (ISO14651) 和 Unicode 3.0 标准的排序机制。这样，就可以对每种语言的每个编码字符进行正确地排序。

Oracle 目前支持 84 种文字排序顺序（68 种单语言排序顺序和 13 种多语言排序顺序）。有关完整列表，请参考 *Oracle9i Globalization Support Manual*。

NLS 排序

- NLS_SORT 用于指定字符数据的排序类型
  - 由环境变量 NLS_LANG 定义
  - 可在会话级别覆盖
- NLSSORT 函数
  - 用于指定字符数据的排序类型
  - 允许在查询级别定义排序顺序

ORACLE

NLS 排序

例如，在法语中，ä 排在 b 的前面，但是，如果使用的是二进制排序，则 ä 排在 z 之后。为了克服二进制排序的局限，Oracle 服务器通过设置 NLS_SORT 参数提供文字排序服务。

NLS 如何影响排序：

下例演示了所有三种排序类型：

- 二进制
- 单语言，使用 French
- 多语言，使用 French_M

下例根据表列表创建了一个包含四个法语单词的列表。

```
SQL> CREATE TABLE list ( num NUMBER(1),
 2   word VARCHAR2(5),
 3   def VARCHAR2(7)
 4   );
Table created.
```

NLS 排序（续）

NLS 如何影响排序（续）：

```
SQL> INSERT INTO list VALUES (1, 'gelée', 'frost');
1 row created.
SQL> INSERT INTO list VALUES (2, 'gelé', 'frozen');
1 row created.
SQL> INSERT INTO list VALUES (3, 'gèle', 'freezes');
1 row created.
SQL> INSERT INTO list VALUES (4, 'gelez', 'freeze');
1 row created.
```

第一个 NLS_SORT 设置为 BINARY。注意，使用 BINARY 排序类型时，e 排在 è 的前面。这是因为，e 包含的二进制值小于以该字符编码格式表示 è 时的二进制值

```
SQL> ALTER SESSION SET NLS_SORT = BINARY;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 4  gelez freeze
 2  gelé  frozen
 1  gelée frost
 3  gèle  freezes
```

下一个 NLS_SORT 设置为法语。法语采用的是单语言排序方法。由于单语言排序限制为两遍排序，因此不能涵盖法语语言的所有细微差别。例如，法语的字母排序顺序是从左到右，而音调符的排序顺序却是从右到左。在多语言排序中将会看到这些。

```
SQL> ALTER SESSION SET NLS_SORT = FRENCH;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 2  gelé  frozen
 3  gèle  freezes
 1  gelée frost
 4  gelez freeze
```

NLS 排序（续）

NLS 如何影响排序（续）：

最后是多语言排序 French_M。请注意这种排序与上述排序的区别。

```
SQL> ALTER SESSION SET NLS_SORT = FRENCH_M;
Session altered.
SQL> SELECT   num, word, def
 2   FROM     list
 3   ORDER BY word;
NUM WORD  DEF
--- ----- -------
 3  gèle  freezes
 2  gelé  frozen
 1  gelée frost
 4  gelez freeze
```

NLSSORT 允许在查询级定义排序。如下例所示，在会话级将 NLS_SORT 设置为 BINARY，但在查询级对其进行更改。请注意所得结果与上例相同。

```
SQL> ALTER SESSION SET NLS_SORT=BINARY;
Session altered.
SQL> SELECT     num, word, def
 2   FROM       list
 3   ORDER BY NLSSORT(word,'NLS_SORT=FRENCH_M');

NUM WORD  DEF
--- ----- -------
 3  gèle  freezes
 2  gelé  frozen
 1  gelée frost
 4  gelez freeze
```

在 SQL 函数中使用 NLS 参数

```
SELECT TO_CHAR(hire_date,'DD.Mon.YYYY',
 'NLS_DATE_LANGUAGE=FRENCH')
FROM employees;
```

```
SELECT ename, TO_CHAR(sal,'9G999D99',
 'NLS_NUMERIC_CHARACTERS=''.,''')
FROM emp;
```

ORACLE

在 SQL 函数中使用 NLS 参数

SQL 字符函数支持单字节和多字节字符。

有些 SQL 函数需要将 NLS 参数显式指定为其参数列表的一部分。因此，SQL 函数可以覆盖环境所指定的行为。

在 SQL 函数中使用 NLS 参数的示例（续）

```
SQL> SELECT TO_CHAR(hire_date, 'DD.Mon.YYYY',
2 'NLS_DATE_LANGUAGE=FRENCH') AS "Hire Date"
3 FROM employees;

Hire Date
-----------
15.Dec.1997
03.Nov.1998
11.Nov.1997
19.Mar.1999
24.Jan.2000
23.Fev.2000
24.Mar.2000
21.Avr.2000
11.Mar.1997
23.Mar.1998
24.Jan.1998
23.Fev.1999
24.Mar.1999
21.Avr.2000
11.Mai.1996
19.Mar.1997
24.Mar.1998
23.Avr.1998
24.Mai.1999
04.Jan.2000
```

在 SQL 函数中使用 NLS 参数的示例（续）

```
SQL> SELECT last_name,
  2   TO_CHAR(salary,'99G999D99','NLS_NUMERIC_CHARACTERS='',.''')
  3   FROM employees;

LAST_NAME                 TO_CHAR(SA
------------------------- ----------
Doran                     7.500,00
Sewall                    7.000,00
Vishney                   10.500,00
Greene                    9.500,00
Marvins                   7.200,00
Lee                       6.800,00
Ande                      6.400,00
Banda                     6.200,00
Ozer                      11.500,00
Bloom                     10.000,00
Fox                       9.600,00
Smith                     7.400,00
Bates                     7.300,00
Kumar                     6.100,00
Abel                      11.000,00
Hutton                    8.800,00
Taylor                    8.600,00
Livingston                8.400,00
Grant                     7.000,00
```

在 SQL 函数中使用 NLS 参数的示例（续）

以下 SQL 函数使用 NLS 参数：

函数	NLS 参数
TO_DATE	NLS_DATE_LANGUAGE NLS_CALENDAR
TO_NUMBER	NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY
TO_CHAR	NLS_DATE_LANGUAGE NLS_NUMERIC_CHARACTERS NLS_CURRENCY NLS_ISO_CURRENCY NLS_CALENDAR
NLS_UPPER, NLS_LOWER, NLS_INITCAP, NLSSORT	NLS_SORT

对于 TO_CHAR、TO_DATE 和 TO_NUMBER 这类函数，还定义了多个格式掩码元素。

数字格式掩码元素：

- “D” 代表十进制分隔符
- “G” 代表组（千位）分隔符
- “L” 代表本地货币符号
- “C” 代表本地 ISO 货币符号
- “U” 代表欧元的双重货币符号

日期格式掩码元素：

- “RM, rm” 代表罗马月份数字
- “IW” 代表 ISO 星期号
- “IYYY, IYY, IY,” 和 “I” 代表 ISO 年份

文字索引支持

- 文字索引
- 通过本地排序获取高性能

```
CREATE INDEX list_word ON
 list (NLSSORT(word, 'NLS_SORT =
French_M'));
```

- NLS_COMP 参数，用于文字比较

文字索引支持

函数型索引专门用于创建按文字排序的索引。SQL 函数 NLSSORT 以给定的文字排序顺序返回用于对第一个参数排序的字节串。在此例中，索引在按 French_M 排序顺序排序的 WORD 上创建。这样，您就可以对依据每种语言规则排序的数据执行基于索引的查询。

比较运算符的文字行为：

NLS_COMP 是一个动态初始化参数，它用于控制 <、> 和 = 等比较运算符处理文字顺序的方式。当这个参数设置为 BINARY（缺省设置）时，将基于字符串的二进制值进行比较。当设置为 ANSI 时，比较运算符通过文字排序顺序来确定依据 NLS_SORT 会话参数执行的操作结果。

使用 NLS 导入和加载数据

- 在导入过程中，数据将从导出文件字符集转换成数据库字符集。
- SQL*Loader:
  - 常规路径：将数据转换为 NLS_LANG 指定的会话字符集。
  - 直接路径：数据直接转换为数据库字符集。

ORACLE

使用 NLS 导入和加载数据

导出文件是通过源数据库字符集导出的。在导入过程中，数据可以从导出文件字符集自动转换为目标数据库字符集。

SQL*Loader 也具有将数据从数据文件字符集转换为数据库字符集的能力。

使用常规路径时，数据可以转换为该会话的 NLS_LANG 参数指定的会话字符集。

使用直接路径时，数据直接转换为数据库字符集。

控制文件会指导 SQL*Loader 如何解释数据文件。

参数字符集说明在每个数据文件中使用什么字符集。

示例：

```
$sqlldr control=utl1case.ctl
characterset=WE8ISO9959P1
```

获取字符集信息

NLS_DATABASE_PARAMETERS:

- **PARAMETER**
  (NLS_CHARACTERSET,
  NLS_NCHAR_CHARACTERSET)
- **VALUE**

获取字符集信息

使用以下查询查看数据库和国家字符集：

```
SQL> SELECT parameter, value
  2   FROM nls_database_parameters
  3   WHERE parameter LIKE '%CHARACTERSET%';

PARAMETER                      VALUE
------------------------       --------------------------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         AL16UTF16
2 rows selected.
```

获取 NLS 设置信息

- **NLS_INSTANCE_PARAMETERS:**
  - **PARAMETER（已显式设置的初始化参数）**
  - **VALUE**
- **NLS_SESSION_PARAMETERS:**
  - **PARAMETER（会话参数）**
  - **VALUE**

获取 NLS 设置信息

以下视图仅显示出已在 init<*SID*>.ora 文件中显式设置的参数的值。

```
SQL> SELECT * FROM nls_instance_parameters;
PARAMETER                      VALUE
------------------------------ ------------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_SORT
NLS_DATE_LANGUAGE
NLS_DATE_FORMAT
NLS_CURRENCY
NLS_NUMERIC_CHARACTERS
NLS_ISO_CURRENCYNLS_CALENDAR
NLS_TIME_FORMAT
NLS_TIMESTAMP_FORMAT
```

获取 NLS 设置信息（续）

```
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
NLS_DUAL_CURRENCY
NLS_COMP
NLS_LENGTH_SEMANTICS            BYTE
NLS_NCHAR_CONV_EXCP             FALSE

17 rows selected.
```

以下视图显示会话参数。

```
SQL> SELECT * FROM nls_session_parameters;
PARAMETER                       VALUE
------------------------------ ------------------------------
NLS_LANGUAGE                    AMERICAN
NLS_TERRITORY                   AMERICA
NLS_CURRENCY                    $
NLS_ISO_CURRENCY                AMERICA
NLS_NUMERIC_CHARACTERS          .,
NLS_CALENDAR                    GREGORIAN
NLS_DATE_FORMAT                 DD-MON-RR
NLS_DATE_LANGUAGE               AMERICAN
NLS_SORT                        BINARY
NLS_TIME_FORMAT                 HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT            DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT              HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT         DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY               $
NLS_COMP                        BINARY
NLS_LENGTH_SEMANTICS            BYTE
NLS_NCHAR_CONV_EXCP             FALSE

17 rows selected.
```

获取 NLS 设置信息

- **V$NLS_VALID_VALUES:**
  - **PARAMETER (LANGUAGE, SORT, TERRITORY, CHARACTERSET)**
  - **VALUE**
- **V$NLS_PARAMETERS:**
  - **PARAMETER （NLS 会话参数，NLS_CHARACTERSET）**
  - **VALUE**

获取 NLS 设置信息

列出 NLS 参数的所有有效值。

```
SQL>  SELECT * FROM v$nls_valid_values
 2    WHERE parameter='LANGUAGE';

PARAMETER          VALUE
---------          --------------
LANGUAGE           AMERICAN
LANGUAGE           GERMAN
LANGUAGE           FRENCH
LANGUAGE           CANADIAN FRENCH
LANGUAGE           SPANISH
LANGUAGE           ITALIAN
LANGUAGE           DUTCH
LANGUAGE           SWEDISH
LANGUAGE           NORWEGIAN
...
```

获取 NLS 设置信息（续）

注：V$NLS_VALID_VALUES 视图显示出 NLS 数据引导文件的内容。它返回一个随给定数据库版本一起提供的列表，该列表包含所有字符集、语言、文字排序以及地域定义。该列表中可能还包含不受支持或内部使用的定义。

显示 NLS 参数的当前值。

```
SQL> SELECT * FROM v$nls_parameters;
PARAMETER                      VALUE
------------------------------ ----------------------------
NLS_LANGUAGE                   AMERICAN
NLS_TERRITORY                  AMERICA
NLS_CURRENCY                   $
NLS_ISO_CURRENCY               AMERICA
NLS_NUMERIC_CHARACTERS         .,
NLS_CALENDAR                   GREGORIAN
NLS_DATE_FORMAT                DD-MON-RR
NLS_DATE_LANGUAGE              AMERICAN
NLS_CHARACTERSET               WE8ISO8859P1
NLS_SORT                       BINARY
NLS_TIME_FORMAT                HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT           DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT             HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT        DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY              $
NLS_NCHAR_CHARACTERSET         AL16UTF16
NLS_COMP                       BINARY
NLS_LENGTH_SEMANTICS           BYTE
NLS_NCHAR_CONV_EXCP            FALSE

19 rows selected.
```

注：其它视图可能包含有一个新列 CHARACTER_SET_NAME，用于显示字符集的名称：CHAR_CS 是数据库字符集的名称，NCHAR_CS 是国家字符集的名称。

例如，DBA_TAB_COLUMNS 从 COL$ 建立此列。

在 SQL 函数中使用 NLS 参数

- 字符集扫描程序
  - 扫描数据库，以确定字符集是否可以进行更改
  - 提供详细说明问题及解决方法的报表
- Oracle 区域设置构建器 (Oracle Locale Builder)
  - 易于使用图形界面
  - 用于查看、修改和创建区域设置定义

ORACLE

在 SQL 函数中使用 NLS 参数

字符集扫描程序：

字符集扫描程序是命令行实用程序，用于辅助字符集转换。扫描程序可以标识字符集转换和数据截断可能发生的区域、所需的工作量以及应该扩展的列的宽度。它提供可行性评估、报告潜在的移植问题、检查所有字符数据并生成数据库扫描摘要。在转换任意字符集前，都应该使用字符集扫描程序。

Oracle 区域设置构建器 (Oracle Locale Builder)：

Oracle9*i* 服务器提供了一组完整的区域设置定义，包括语言、地域、字符集和文字排序。如果需要对任何现有的区域设置定义进行自定义，或者创建新的区域设置定义，则可以使用新增的 Oracle 区域设置构建器 (Oracle Locale Builder)，它提供了一种简单易用的图形用户界面，使您可以轻松地查看、自定义和定义各种区域设置。

小结

在这一课中，您应该能够掌握：

- 为数据库选择数据库字符集和国家字符集
- 使用服务器或会话的各种类型的国家语言支持参数

ORACLE

练习 20 概览

此练习涉及以下主题：

- 检查数据库和国家字符集
- 标识有效的 NLS 值
- 设置 NLS 参数

ORACLE

练习 20 概览

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

练习 20：使用 “全球化支持”

1. 检查数据库和国家字符集。
2. 哪些是数据库字符集的有效值？
3. 检查该会话的所有日期中的年份是否以四位数字的格式显示。将 NLS_LANGUAGE 更改为 FRENCH。从 DUAL 选择 sysdate。

如何在 UNIX 环境中创建 Oracle9*i* 数据库

简介

创建数据库的步骤

创建一个可使用的数据库共需要六步来完成：其中三步用于创建数据库，其余三步用于设置数据库，让它可以使用。

- 设置操作系统环境变量 ORACLE_HOME、ORACLE_SID、PATH 和 LD_LIBRARY_PATH。
- 编辑/创建 initsid.ora 参数文件。
- 在 SQL*Plus 中执行 CREATE DATABASE 命令。
- 运行所需的 catalog.sql 和 catproc.sql 脚本。
- 运行 pupbld.sql 脚本。
- 创建用户数据的表空间以及数据库所需的任何其它表空间

注：编写本文档的内容时，假定已在 ORACLE_HOME 中安装了 Oracle9*i* Server。本附录并不讨论 Oracle9*i* Server 的安装。

设置环境

在创建数据库之前，必须配置 UNIX 环境，并且必须已安装了 Oracle9*i* Server。

必须设置四个环境变量：ORACLE_HOME、ORACLE_SID、PATH、LD_LIBRARY_PATH。

ORACLE_HOME 是安装 Oracle9*i* Server 的顶级目录的完整路径。ORACLE_HOME 目录应该由安装 Oracle9*i* Server 的人员提供，通常是 UNIX 管理员或 DBA。

ORACLE_SID 是给数据库例程分配的、可由用户定义的名称。操作系统使用 ORACLE_SID（系统标识符）来区分在同一台计算机上运行的各个数据库例程。

PATH 指定操作系统查找可执行程序（如 SQL*Plus）时要搜索的路径。Oracle9*i* 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 定义所需库文件的存储目录。

示例

Bourne 或 Korn shell：

```
$ ORACLE_HOME=/u01/oracle9i/product/9.0.1; export ORACLE_HOME
$ ORACLE_SID=db01; export ORACLE_SID
$ PATH=/usr/bin:/usr/ccs/bin:$ORACLE_HOME/bin; export PATH
$ LD_LIBRARY_PATH=/usr/lib:$ORACLE_HOME/lib; export LD_LIBRARY_PATH
```

C shell：

```
% setenv ORACLE_HOME /u01/oracle9i/product/9.0.1
% setenv ORACLE_SID db01
% setenv PATH $PATH:$ORACLE_HOME/bin
% setenv LD_LIBRARY_PATH /usr/lib:$ORACLE_HOME/lib
```

编辑 initsid.ora

编辑/创建 initsid.ora

每次数据库启动时，都会读取 initsid.ora 文件（可由用户配置的文本文件）。该文件中的参数将对数据库设置进行初始化。initsid.ora 文件中的参数设置不仅在启动时影响数据库，而且还影响数据库的创建方式。在创建数据库之前，必须先配置 initsid.ora 文件。

在安装 Oracle9*i* Server 时，就会将一个示例 init.ora 文件存放在 $ORACLE_HOME/dbs 中。将该文件作为备份文件保存，不要修改该文件；创建该文件的一个副本，并包含 ORACLE_SID 的名称。

示例

```
$ cd $ORACLE_HOME/dbs
$ cp init.ora initdb01.ora
```

示例 init.ora 文件中有很多注释，这些注释包含有关参数设置的建议。

initsid.ora 中的参数不需要按顺序列出，如果多次列出某个参数，则使用该参数的最后一个设置。Oracle9*i* Reference 中建议将参数按字母顺序列出以免重复。

应该对某些参数进行配置，其中包括 db_name、control_files、background_dump_dest、user_dump_dest、core_dump_dest 和 undo_management。

将参数 background_dump_dest、user_dump_dest 和 core_dump_dest 设置为存放跟踪文件的完整路径位置：

- core_dump_dest　　包含数据库生成的核心转储
- user_dump_dest　　包含用户跟踪文件
- background_dump_dest　　包含后台进程的跟踪文件和 alert.log。

db_name 是数据库的名称，其用途与 ORACLE_SID 不同。ORACLE_SID 是用于指定数据库例程的名称。db_name 和 ORACLE_SID 在多数情况下是相同的，但这并不是必需的。initsid.ora 中的 db_name 必须与创建数据库时 CREATE DATABASE 命令中使用的数据库名称相同（区分大小写）。

control_files 初始化参数为数据库指定每个控制文件的完整路径和文件名。在创建数据库时，它指定必须创建的控制文件。

undo_management 初始化参数确定是由 Oracle 服务器自动处理还原数据，还是由 DBA 手动处理还原数据。在初始化文件中将 undo_management 设置为 AUTO。

附录 A 结尾附有一个示例 initdb01.ora 文件。

创建数据库

在设置环境并配置 initsid.ora 后，就可以创建数据库了。Oracle 数据库是通过执行 CREATE DATABASE 命令创建的。CREATE DATABASE 命令指定了日志文件的数量和位置、SYSTEM 表空间、UNDO 表空间和 TEMP 表空间的位置和大小，以及数据库使用的字符集（这并不是一个详尽的列表）。

可使用 Oracle 实用程序 SQL*Plus 创建数据库。UNIX 的 SQL*Plus 可执行程序是 sqlplus。

在创建数据库时，Oracle9*i* 服务器仅识别 SYS 用户和 SYSDBA 角色。要创建数据库，必须以用户 SYS 和角色 SYSDBA 的身份连接到 SQL*Plus。可以使用以下两种方法之一完成该操作。

1. 如果连接 SQL*Plus 的 UNIX 用户是管理员组的成员（在 Oracle9*i* Server 安装期间定义），则使用以下语法：

   ```
   $ sqlplus  '/ as sysdba '
   ```

   或者

   ```
   $ sqlplus /nolog
   SQL> connect / as sysdba
   ```

2. 如果连接 SQL*Plus 的 UNIX 用户不是管理员组的成员，则管理员必须创建数据库的口令文件，并且必须使用在口令文件中给 SYS 分配的口令。以下语法假定在口令文件中给 SYS 分配的口令是 oracle。

   ```
   $ sqlplus 'sys/oracle as sysdba'
   ```

   或者

   ```
   $ sqlplus /nolog
   SQL> connect sys/oracle as sysbda
   ```

创建和执行 SQL 语句的步骤是：

1. 创建一个包含 CREATE DATABASE 命令的 SQL 脚本。（附录 A 的结尾附有一个创建数据库的示例脚本。）
2. 使用上面所示的两种方法之一，以 SYS AS SYSDBA 的身份连接到 SQL*Plus。
3. 在 NOMOUNT 模式下启动数据库。
4. 执行 SQL 脚本。

创建数据库（续）

示例

```
% sqlplus 'sys/oracle as sysdba'
SQL> startup nomount
ORACLE instance started.
Total System Global Area   21790532 bytes
Fixed Size                   278340 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
SQL> @crdbdb01.sql
SQL> CREATE DATABASE db01
  2     LOGFILE
  3       GROUP 1 ('$HOME/ORADATA/u03/log_01_01_db01.rdo') SIZE 10M,
  4       GROUP 2 ('$HOME/ORADATA/u03/log_02_01_db01.rdo') SIZE 10M
  5     DATAFILE '$HOME/ORADATA/u01/system_01_db01.dbf' SIZE 100M
  6        AUTOEXTEND ON NEXT 5M MAXSIZE 150M
  7     DEFAULT TEMPORARY TABLESPACE temp
  8        TEMPFILE '$HOME/ORADATA/u02/temp_01_db01.dbf' SIZE 15M
  9        AUTOEXTEND ON NEXT 5M MAXSIZE 30M
 10     CHARACTER SET WE8ISO8859P1
 11     NATIONAL CHARACTER SET AL16UTF16
 12  ;
Statement processed.
```

运行脚本

必须在创建数据库之后运行脚本 catalog.sql 和 catproc.sql，这两个脚本位于 $ORACLE_HOME/rdbms/admin 目录中。catalog.sql 脚本用于创建数据字典视图；catproc.sql 用于创建使用 PL/SQL 所需的程序包和过程。

必须以 SYS 的身份运行这两个脚本。在创建脚本之前，确保数据库已打开。

示例

```
% sqlplus /nolog
SQL> CONNECT / AS SYSDBA
SQL> @$ORACLE_HOME/rdbms/admin/catalog.sql
SQL> @$ORACLE_HOME/rdbms/admin/catproc.sql
```

在系统不忙的时候，运行这两个脚本总共需要 35 到 65 分钟。

在运行这些脚本后，验证对象是否有效。以下查询可返回所有无效的对象。

```
SQL> SELECT   owner,object_name,object_type
  2  FROM     dba_objects
  3  WHERE    status = 'INVALID'
  4  ORDER BY owner,object_type,object_name;
```

运行 pupbld.sql

pupbld.sql 脚本用于创建 “产品用户配置文件” (Product User Profile) 表以及相关的过程，该脚本位于 $ORACLE_HOME/sqlplus/admin 目录中。运行此脚本还可在每次用户连接到 SQL*Plus 时防止生成警告消息。必须以用户 SYSTEM 的身份运行此脚本。

```
$ sqlplus system/manager
SQL> @$ORACLE_HOME/sqlplus/admin/pupbld.sql
```

创建表空间

创建安装所需的其它表空间。

在数据库的安装过程中，通常创建以下表空间：

- users　　用户数据
- tools　　用户 SYSTEM 创建的对象（可选）

应该创建这两个表空间。

示例

```
SQL> create tablespace USERS
  2  datafile '$HOME/ORADATA/u03/users_01_db01.dbf' SIZE 25M
  3  PERMANENT
  4  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
  5  SEGMENT SPACE MANAGEMENT auto;
```

小结

- 设置 ORACLE_HOME、ORACLE_SID、PATH 和 LD_LIBRARY_PATH。
- 编辑 initsid.ora。
- 执行 CREATE DATABASE 命令。
- 运行 catalog.sql 和 catproc.sql 脚本。
- 运行 pupbld.sql 脚本。
- 创建数据库所需的表空间。

示例 initdb01.ora

```
background_dump_dest=$HOME/ADMIN/BDUMP
compatible=9.0.0
control_files=$HOME/ORADATA/u01/ctrl_01_sid.ctl
core_dump_dest=$HOME/ADMIN/CDUMP
db_block_size=4096
db_cache_size=4M
db_domain=world
db_name=db01
global_names=TRUE
instance_name=db01
max_dump_file_size=10240
remote_login_passwordfile=exclusive
service_names=db01
shared_pool_size=8M
undo_management=AUTO
user_dump_dest=$HOME/ADMIN/UDUMP
```

创建数据库的示例脚本

```
CREATE DATABASE db01
   LOGFILE
      GROUP 1 ('$HOME/ORADATA/u03/log_01_01_db01.rdo') SIZE 10M,
      GROUP 2 ('$HOME/ORADATA/u03/log_02_01_db01.rdo') SIZE 10M
   DATAFILE '$HOME/ORADATA/u01/system_01_db01.dbf' SIZE 100M
      AUTOEXTEND ON NEXT 5M MAXSIZE 150M
   DEFAULT TEMPORARY TABLESPACE temp
      TEMPFILE '$HOME/ORADATA/u02/temp_01_db01.dbf' SIZE 15M
      AUTOEXTEND ON NEXT 5M MAXSIZE 30M
   CHARACTER SET WE8ISO8859P1
   NATIONAL CHARACTER SET AL16UTF16
;
```

B

手动管理还原数据（回退段）

ORACLE

目标

完成这一课的学习后，您应该能达到下列目标：

- 使用相应的存储设置创建回退段
- 维护回退段
- 计划回退段的个数和大小
- 解决常见的回退段问题

ORACLE

创建回退段

```
CREATE ROLLBACK SEGMENT rbs01
  TABLESPACE rbs
  STORAGE (
    INITIAL    100K
    NEXT      100K
    MINEXTENTS   20
    MAXEXTENTS   100
    OPTIMAL    2000K );
```

ORACLE

创建回退段

使用下列命令创建回退段：

```
CREATE [PUBLIC] ROLLBACK SEGMENT rollback_segment
    [TABLESPACE tablespace]
    [STORAGE (        [INITIAL     integer[K|M]]
                                   [NEXT        integer[K|M]]
                                   [MINEXTENTS  integer]
                                   [MAXEXTENTS {integer|UNLIMITED}]
                                   [OPTIMAL    {integer[K|M]|NULL}]
                      )
]
```

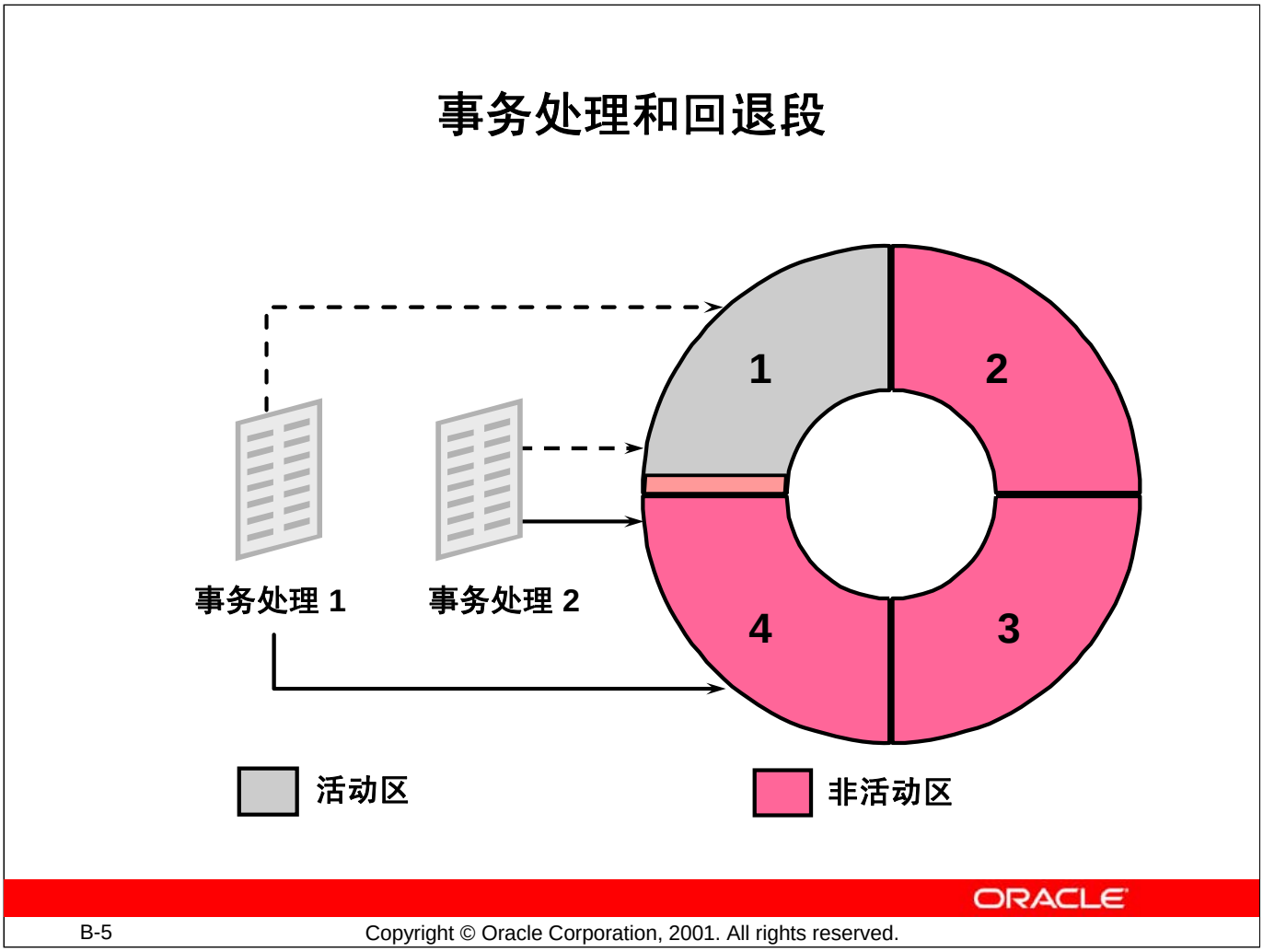
创建回退段（续）

限制

- 创建回退段时可指定为公用回退段或专用回退段（缺省值），但无法对其进行更改。
- 对于回退段，MINEXENTS 必须至少为两个。
- 无法为回退段指定 PCTINCREASE，它始终设置为 0。
- 如果指定了 OPTIMAL，则它必须至少等于回退段的初始大小，该大小是 INEXTENTS 定义的区数所使用的空间。

原则

- 始终为回退段使用 INITIAL = NEXT，以确保所有区的大小相同。
- 设置 OPTIMAL 值以最小化回退段区的分配和回收。
- 避免将 MAXEXTENTS 设置为 UNLIMITED。否则，可能会因程序错误对回退段和数据文件进行不必要的扩展。
- 始终将回退段放置在单独、排它的表空间中，以使争用和碎片减到最小。



回退段分配

当事务处理开始时，需要将一个回退段分配给这个事务处理。事务处理可以使用下列命令请求特定回退段：

```
SET TRANSACTION USE ROLLBACK SEGMENT rollback_segment
```

如果没有进行这样的请求，则 Oracle 服务器将选择事务处理最少的回退段，并将它分配给事务处理。

使用区

事务处理以连续、循环的方式使用回退段中的区，即在当前区已满后从一个区移动到下一个区。事务处理将一个条目写入其在回退段中的当前位置，并使当前指针向前移动，前进量为该条目的大小。

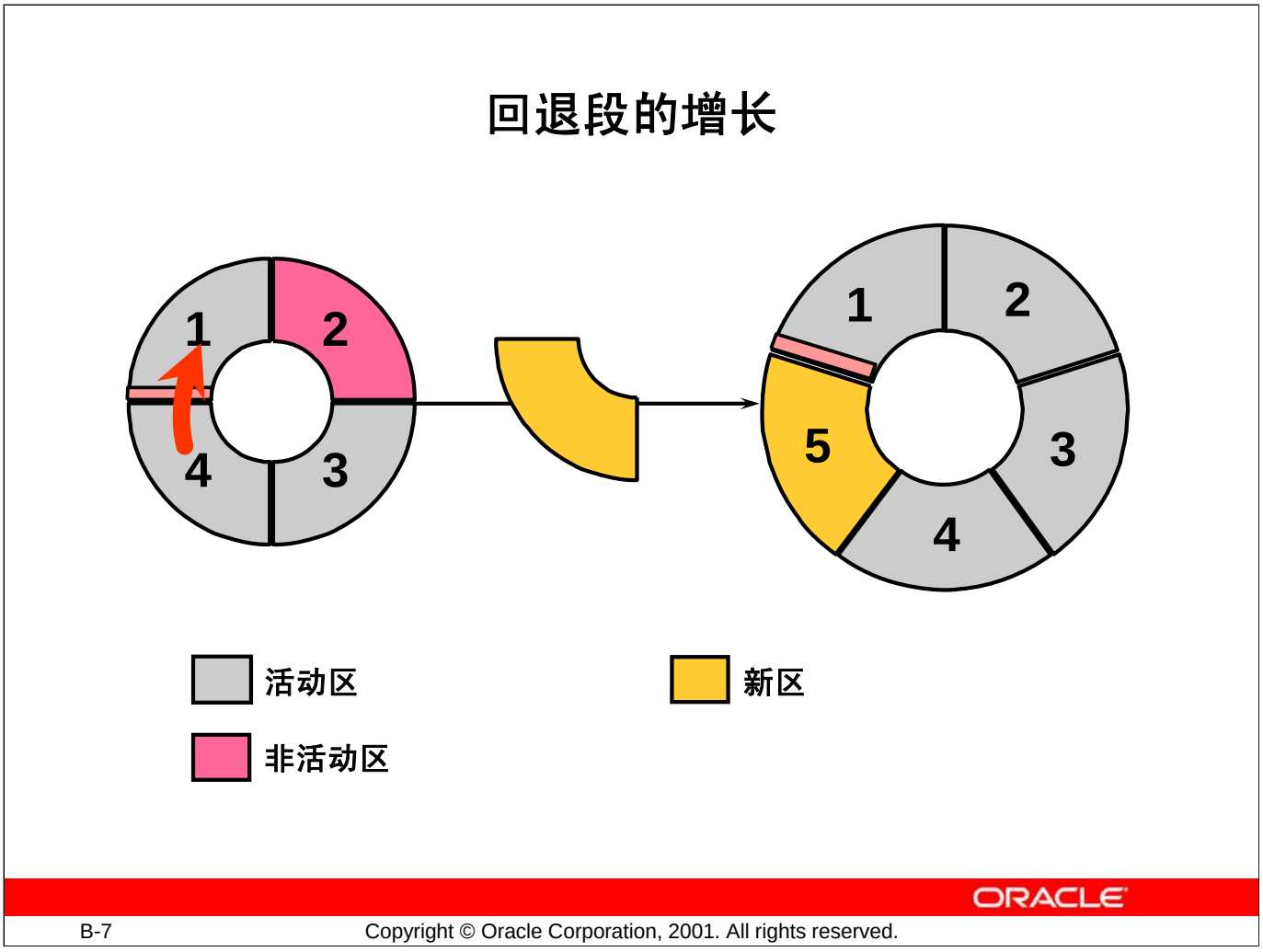
可以向回退段的同一个区写入不止一个事务处理；但是每个回退段块都仅包含来自一个事务处理的信息。

回退段分配（续）

示例

在幻灯片的示例中，已将两个事务处理分配到一个具有四个区的回退段中。

1. 当事务处理开始时，它们开始写入回退段的第 3 个区中。
2. 随着这两个事务处理生成更多的回退信息，它们继续写入第 3 个区中。
3. 当第 3 个区已满时，事务处理写入环中的下一个区（即第 4 个区）。当事务处理开始写入新区时（如本步骤），称为绕行。
4. 当回退段的最后一个区（第 4 个区）已满时，事务处理可以使用环中的第一个区（第 1 个区），前提是该区空闲或处于非活动状态。只有当前不存在使用某区的活动事务处理（即所有写入该区的事务处理都已完成）时，这个区才是空闲或非活动的。



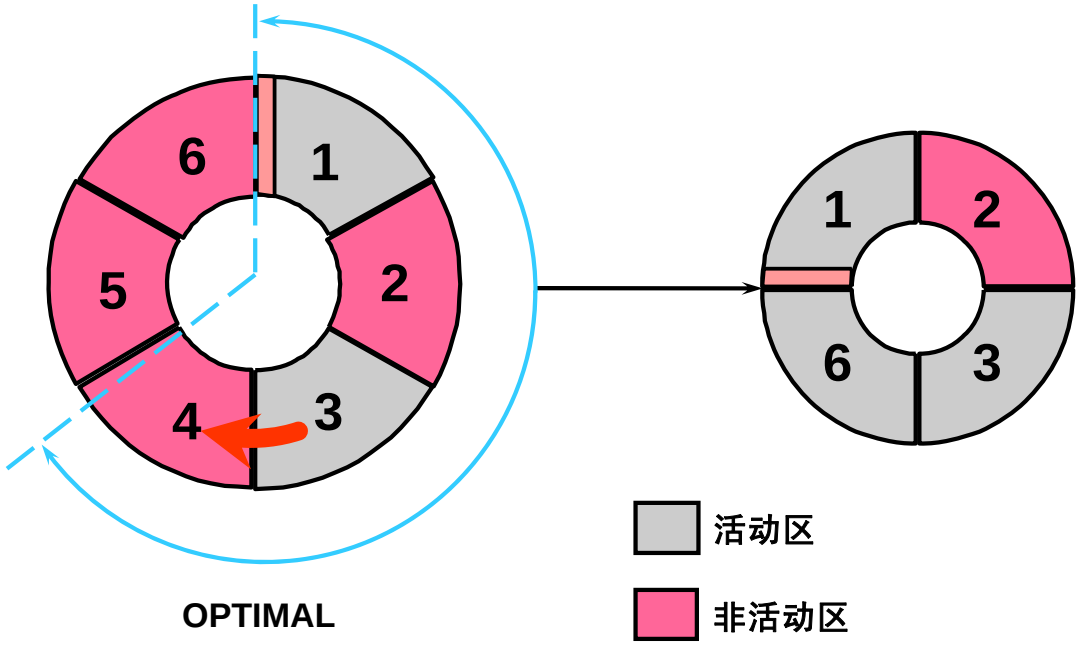
回退段的增长

在当前区的所有块都已被占用而事务处理需要其它块以获得更多空间时，回退段的指针或头部会移动到下一个区。当最后一个区已满时，指针会移动到第一个区的前面。

只有当指针所在的区没有活动事务处理时，指针才能移动到下一个区。指针不能跳过某个区。如果下一个区正在使用中，则事务处理将为回退段分配其它区。这个过程称作扩展。

回退段将以这种方式增长，直到达到由 MAXEXTENTS 参数指定的区的最大个数。

回退段的收缩



OPTIMAL 参数

如果可能，可以使用 OPTIMAL 参数来指定回退段必须收缩到的大小（单位为字节）。指定 OPTIMAL 将使回退段中空间的浪费降至最低。如果指定了 OPTIMAL 参数，则回退段将在导致其增长的事务处理完成时释放空间。

事务处理完成后，将不会对区进行回收。只有在头部从一个区移动到下一个区时，才会对区进行回收。要对区进行回收，必须满足以下两个条件：

- 回退段的当前大小超过 OPTIMAL 的设置值。
- 存在连续的非活动区。

Oracle 服务器尝试回收回退段的大小，直到它等于 OPTIMAL，但当要回收的下一个区正在使用时将不得不突然停止。

Oracle 服务器总是回收最旧的非活动区，因为这些活动区最不可能用于读一致性。

使回退段联机

- 使用下列命令使回退段可用：

```
ALTER ROLLBACK SEGMENT rbs01 ONLINE;
```

- 指定下列初始化参数，确保启动时回退段处于联机状态：

```
ROLLBACK_SEGMENTS=(rbs01, rbs02)
```

ORACLE

ALTER ROLLBACK SEGMENT 命令

创建回退段时，该回退段处于脱机状态而无法使用。要使回退段可供事务处理使用，请使用 ALTER ROLLBACK SEGMENT 命令使它联机。

语法

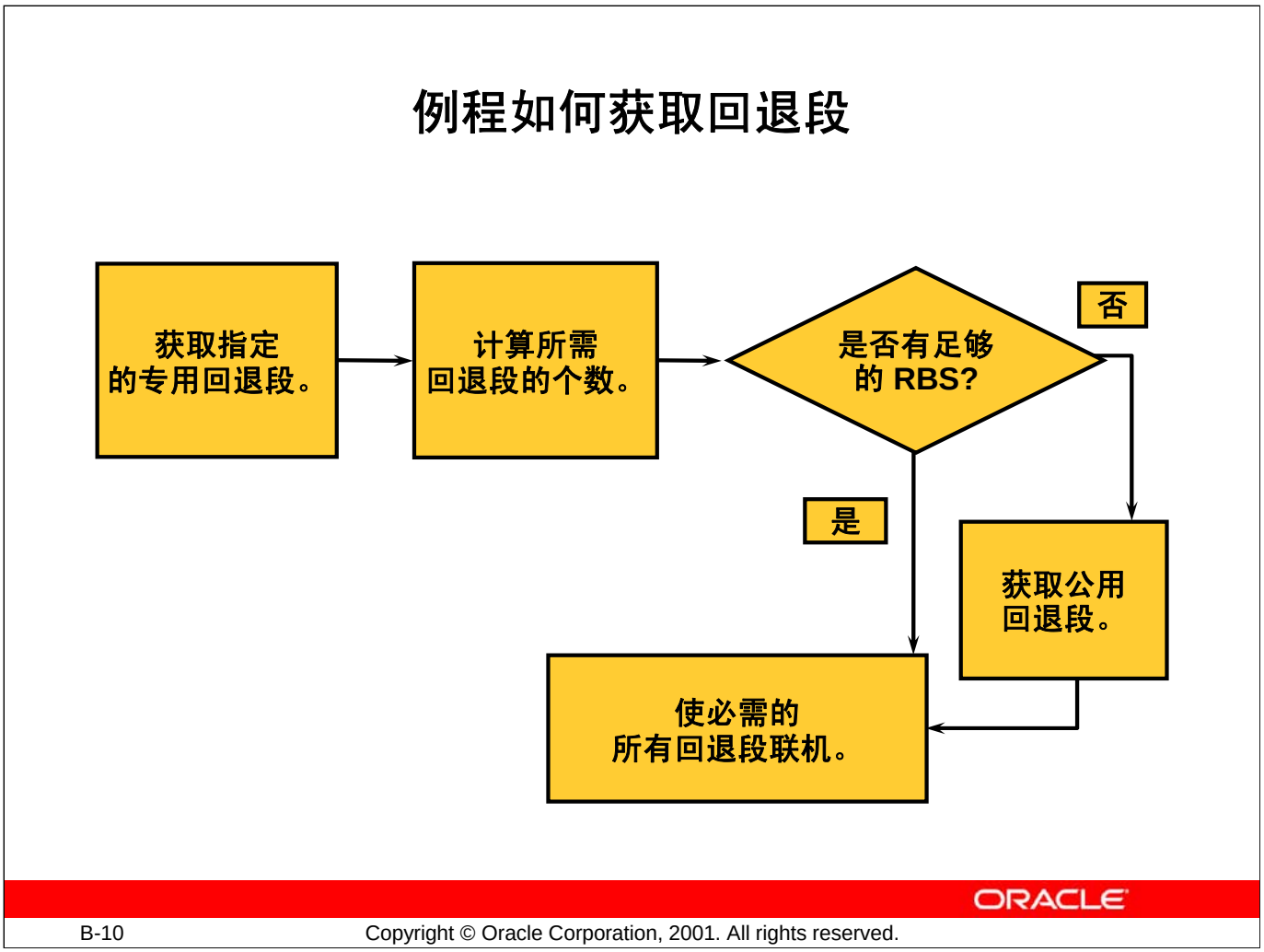
使用下列命令使回退段可用：

```
ALTER ROLLBACK SEGMENT rollback_segment ONLINE;
```

可以通过某个例程联机的回退段的个数由 MAX_ROLLBACK_SEGMENTS 参数限制。请对该值进行设置，使其大于所用例程需要的非 SYSTEM 回退段的个数。

只有关闭例程后，回退段才处于联机状态。要确保回退段始终可以通过例程进行联机，请在参数文件中指定回退段的名称，如下所示：

```
ROLLBACK_SEGMENTS=(rbs01, rbs02)
```



例程如何获取回退段

下列步骤解释当例程打开数据库时如何获取回退段：

- 例程可以获取初始化参数 ROLLBACK_SEGMENTS 中指定的所有回退段。
- TRANSACTIONS init.ora 参数除以 TRANSACTIONS_PER_ROLLBACK_SEGMENT init.ora 参数，所得结果即是例程所需回退段的个数。如果该值大于已通过该例程联机的非 SYSTEM 回退段的个数，则该例程将获取其它公用回退段以弥补不足的部分。如果公用回退段不足，将打开数据库以供用户使用。在此过程中，不会生成任何错误。

更改回退段存储设置

- 使用 ALTER ROLLBACK SEGMENT 命令。
- 您可以更改 OPTIMAL 或 MAXEXTENTS。

```
ALTER ROLLBACK SEGMENT rbs01
  STORAGE( MAXEXTENTS 200 );
```

ORACLE

更改回退段存储

回退段的存储参数可以通过 ALTER ROLLBACK SEGMENT 命令进行更改：

```
ALTER ROLLBACK SEGMENT rollback_segment
[STORAGE ( [NEXT         integer[K|M]]
           [MINEXTENTS   integer]
           [MAXEXTENTS {integer|UNLIMITED}]
           [OPTIMAL    {integer[K|M]|NULL}]
         )
]
```

使用该命令重新定义 OPTIMAL 或 MAXEXTENTS 参数。

从回退段回收

- 使用 **ALTER ROLLBACK SEGMENT** 命令。
- 如果区处于活动状态，则这些区可能不会收缩到指定的大小。

```
ALTER ROLLBACK SEGMENT rbs01
  SHRINK TO 4M;
```


从回退段回收空间

如果已为回退段指定了 OPTIMAL，则 Oracle 服务器将尝试回收区以释放大于最优大小的空间。

手动回收

若要从回退段手动回收空间，请使用下列命令：

```
ALTER ROLLBACK SEGMENT rollback_segment
      SHRINK [ TO integer [ K|M ]];
```

该命令尝试将回退段的大小减小到指定大小，但当某个区处于活动状态而无法回收时将突然停止。

如果未指定整数值，则 *integer* 服务器尝试回收区，直到回退段的大小等于 OPTIMAL 指定的值。

如果所指定的整数值大于回退段的当前大小，将忽略该命令。

使回退段脱机

- 使回退段脱机，以使其不可用。
- 如果有事务处理正在使用回退段，则其状态将暂时更改为 PENDING OFFLINE。

```
ALTER ROLLBACK SEGMENT rbs01
  OFFLINE;
```

ORACLE

使回退段脱机

使回退段脱机：

- 以防止新的事务处理使用回退段
- 如果需要删除该回退段

语法

使用下列命令使回退段脱机：

```
ALTER ROLLBACK SEGMENT rollback_segment OFFLINE;
```

如果执行这条语句时有事条处理正在使用该回退段，则回退段的状态将被设置为 PENDING OFFLINE，如同在动态性能视图 V$ROLLSTAT 中看到的那样。只有当现有的所有事务处理都完成后，该段才能进入脱机状态。

删除回退段

- 回退段必须脱机之后才能删除。
- 使用以下命令删除回退段：

```
DROP ROLLBACK SEGMENT rbs01;
```

ORACLE

删除回退段

使用下列命令删除回退段：

```
DROP ROLLBACK SEGMENT rollback_segment;
```

当不再需要某回退段或需要使用不同的 INITIAL、NEXT 或 MINEXTENTS 存储设置重新创建某回退段时，可能需要删除该回退段。

回退段必须脱机之后才能删除。

计划回退段：个数

- OLTP
  - 许多小回退段
  - 每个回退段对应四个事务处理
  - 每个回退段最多对应十个事务处理
- 批处理
  - 少量大回退段
  - 每个事务处理对应一个回退段

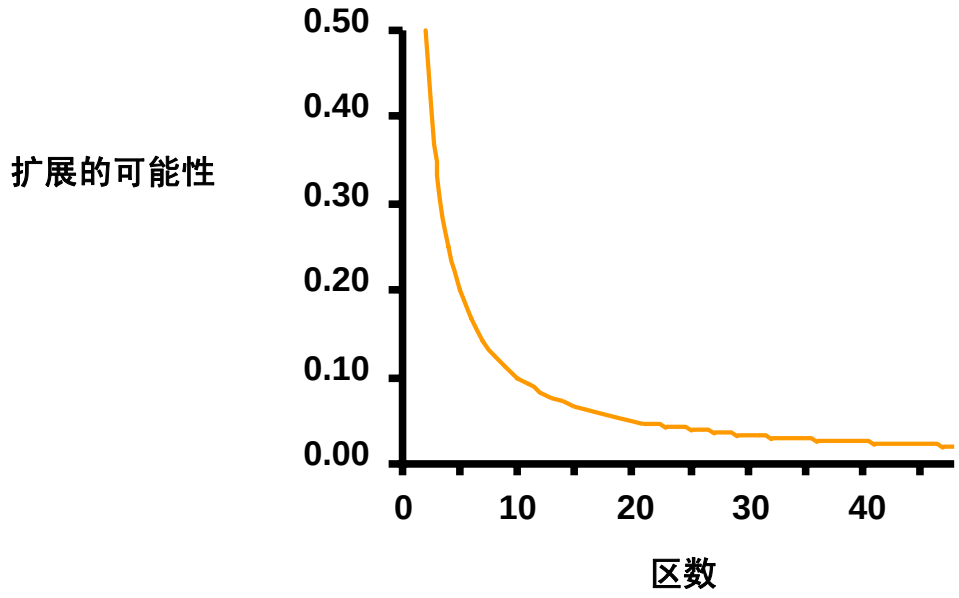
ORACLE

回退段个数

回退段的标题块包含定义每个事务处理状态的事务处理表条目。使用回退段的每个事务处理必须经常更新事务处理表。但是这样做会导致对标题块的争用，尤其是在联机事务处理 (OLTP) 环境中。因为在 OLTP 环境中通常使用的是短事务处理，所以建议在该环境中使用许多小型回退段。如果可能，请为每四个并发事务处理创建一个回退段。

在批处理环境中，通常运行的是数量较少且可能需要执行多次更改的作业。这些作业需要使用大型的回退段。因此，在批处理环境中，可以通过在大型表空间中创建回退段来实现回退段的增长。

计划回退段：区数



ORACLE

回退段的大小

存储回退时的必要信息所需的字节数取决于以下两项：

- 正在执行的事务处理的类型（插入、更新、删除等等）
- 正在处理的实际数据

总的来说，在表中插入给定记录要比删除同一记录生成的回退要少。插入时只需在回退中存储行标识，而删除时必须存储实际的行本身。

估计回退段的大小时，可以通过运行预期的最长事务处理并检查回退段的大小或生成的回退段的数量来进行。 *Oracle8i：性能优化* 和 *Oracle8i：SQL 语句优化* 这两个课程讨论了如何监视统计信息。

区数

通过创建具有大量区的回退段，可以将回退段的动态扩展降至最低。建议使用 `MINEXTENTS=20` 创建回退段以减少可能的扩展。

回退段问题

- 用于事务处理的空间不足
- 读一致性错误
- 阻塞会话
- 将表空间置于脱机状态时出错

ORACLE

用于事务处理的空间不足

- 表空间中没有空间：
  - 扩展数据文件
  - 启用数据文件自动扩展
  - 添加数据文件
- 达到段的 MAXEXTENTS 设置值
  - 增加 MAXEXTENTS 值
  - 重新创建具有最大区大小的段

ORACLE

可能的原因

如果事务处理使用单个回退段，当回退段中空间不足 (ORA-01562) 时事务处理可能失败。这可能由以下原因之一引起：

- 表空间中没有足够的空间供回退段扩展 (ORA-01560)。
- 回退段中的区数已达到 MAXEXTENTS 的设置值，并且无法分配额外的区 (ORA-01628)。

解决方案

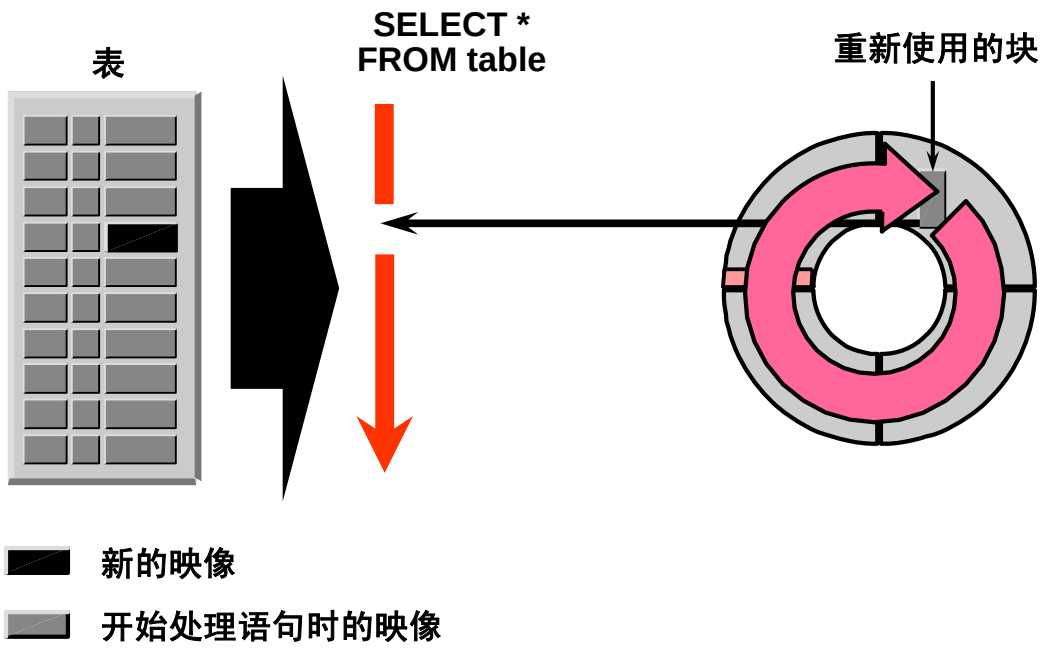
如果表空间没有空闲空间，则使用以下方法增加可用空间：

- 设置 OPTIMAL 以确保单个回退段不会使用表空间中的所有空闲空间
- 将回退段收缩到其最优大小
- 增加表空间的大小

如果回退段由于已达到 MAXEXTENTS 所强加的限制而无法分配更多的区，则：

- 增加回退段的 MAXEXTENTS 值。
- 删除旧的回退段，重新创建新的包含更大区的回退段，以避免再次出现问题。

读一致性错误



可能的原因

为了提供读一致性，Oracle 服务器保证语句不会看到其他用户在语句开始时进行的未提交的更改或在语句开始执行之后进行的更改。如果 Oracle 服务器无法构造数据的读一致性映像，用户将会收到 “ORA-01555 快照已过期” 错误。这个错误可能发生的条件是，进行更改的事务处理已提交并且：

- 回退标题中的事务处理位置已被重新使用。
- 回退段中的前像已被另一个事务处理覆盖。

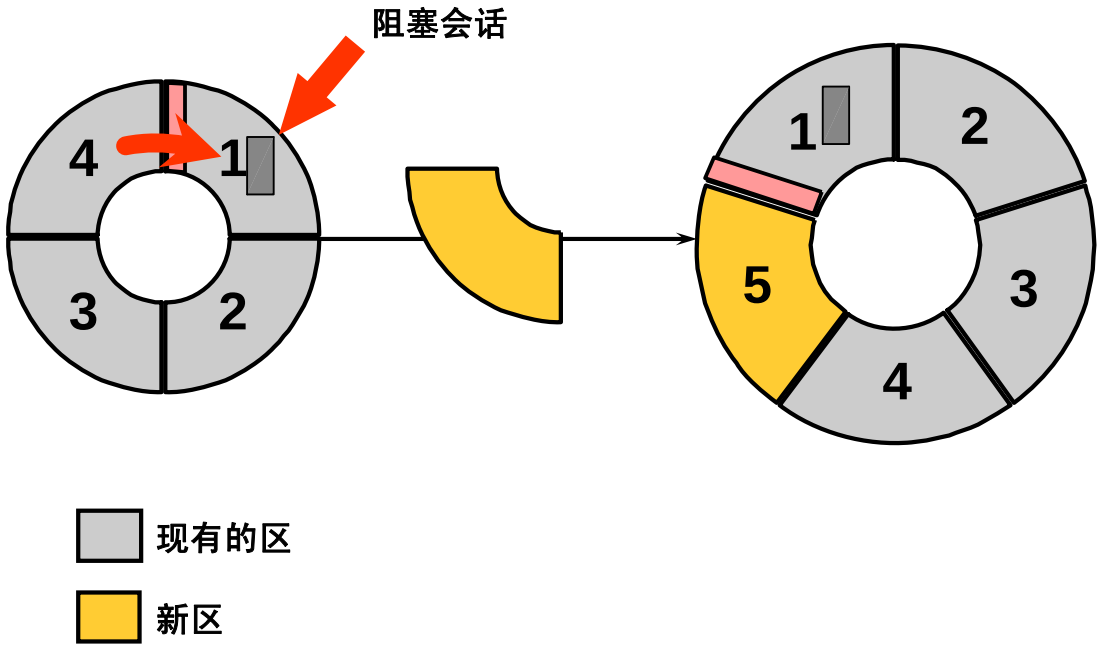
解决方案

通过确保创建具有下列特点的回退段，可以将读一致性错误减到最小：

- 较高的 MINEXTENTS 值
- 较大的区大小
- 较高的 OPTIMAL 值

注：无法通过增加 MAXEXTENTS 值来避免这些错误。

阻塞会话



可能的原因

当回退段中的某个区已满时，Oracle 服务器将尝试重新使用该段中的下一个区。如果这个新区包含一个活动条目（即由活动事务处理写入的条目），则无法使用该区。在这些情况下，回退段将分配其它区。事务处理无法跳过环中的某个区而继续写入后续的区。尽管存在许多空闲区，但只进行了少量更改、却已空闲了较长时间的事务处理可能会导致回退段的增长。在这种情况下，将浪费大量空间，因此，数据库管理员可能需要进行干预，以避免回退段增长过度。

可能的原因（续）

解决方案

查询 V$ROLLSTAT、V$SESSION 和 V$TRANSACTION 视图以找到所有阻塞的事务处理。

示例

```
SQL> SELECT s.sid, s.serial#, t.start_time, t.xidusn, s.username
     2 FROM v$session s, v$transaction t, v$rollstat r
     3 WHERE s.saddr = t.ses_addr
     4 AND t.xidusn = r.usn
     5 AND ((r.curext = t.start_uext-1) OR
     6 ((r.curext = r.extents-1) AND t.start_uext=0));
    SID   SERIAL#     START_TIME             XIDUSN   USERNAME
    ---   -------     ------------------      ------- ------
    9     27          10/30/97 21:10:41       2       SYSTEM
    1 row selected.
```

检查事务处理是否可以由用户结束。如果不能，可能需要终止会话。

将表空间置于脱机状态时出错

您不能将包含有活动回退段的表空间置于脱机状态。

1. **确定表空间中有哪些回退段。**
2. **将所有这些回退段置于脱机状态。**
3. **查找使用这些回退段的活动事务处理。**
4. **查找会话 ID 和序列号。**
5. **如有必要，终止会话。**
6. **使表空间脱机。**

ORACLE

问题诊断和解决

如果某表空间包含一个或多个活动的回退段，则无法使之脱机。执行语句的会话将收到 ORA-01546 错误消息。

解决方案

执行下列步骤：

1. 查询 DBA_ROLLBACK_SEGS 以找到该表空间中有哪些回退段。
2. 令表空间中的所有回退段脱机。
3. 检查 V$TRANSACTION 以找到哪些事务处理当前正在使用这些回退段。
4. 使用 V$SESSION 以获取用户名和会话信息。
5. 终止会话或令用户结束事务处理。
6. 使表空间脱机。

SQL*Plus 练习解答

练习解答约定

以下解答是为那些对使用 SQL*Plus 完成各项练习感兴趣的学生而准备的。

运行 SQL Lab 脚本

Lab 脚本已包含在执行特定任务的 lab 练习中。此练习要完成的任务您目前可能还没有学到，或者它只是为了便于您练习。要求您运行 `.sql` 脚本时，一定要先仔细查看该脚本，然后再运行。这将有助于全面了解实际的创建结果，以及该脚本如何影响您要在练习中完成的任务。

使用操作系统命令行

有时候需要使用操作系统命令行来执行某些任务。要退出 SQL*Plus 并进入操作系统命令行模式，就需要在 SQL*Plus 命令行中使用 !。在操作系统命令行中完成操作后，使用 `Exit` 即可返回到 SQL*Plus 命令行。例如：删除位于 `$HOME/ADMIN/PFILE` 中的当前口令文件。

```
SQL > !rm $HOME/ADMIN/PFILE/orapw$ORACLE_SID
$ > exit
```

作为 SYSDBA 连接

如果要求作为用户 SYS 进行连接，将始终使用 SYSDBA 权限。

```
CONNECT / AS SYSDBA
```

练习 1：解答

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。

答案：D

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

答案：B

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。

答案：B, D

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

答案：D

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

答案：对

6 PGA 是一个内存区，其中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

答案：错，PGA 是一个内存区，其中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。

练习 1：解答（续）

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
答案：C

8 列出五个必备的后台进程。

_____
_____
_____
_____
_____

答案：DBWR, LGWR, PMON, SMON, CKPT

9 请将下列进程与其相应的任务相连。

a 数据库写入器	E 帮助写入数据文件头
b 日志写入器	C 负责例程恢复
c 系统监视器	D 进程失败后进行清理
d 进程监视器	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

答案：E, C, D, B, A

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
答案：对

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
答案：D, A, C, B, E

12 列出 Oracle 服务器的组件。

_____
_____

答案：Oracle 进程、 Oracle 数据库

练习 1：解答（续）

13 列出 Oracle 例程的组件。

_____

_____

答案：SGA、后台进程

14 列出组成 Oracle 数据库的三种文件类型。

_____

_____

_____

答案：数据文件、控制文件、重做日志文件

练习 2：解答

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)。

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

     输入您的班级所使用的 Unix 服务器的名称。教师将提供该名称。请完全按原样输入。此名称是区分大小写的。

  2. 打开一个 MSDOS 窗口。
  3. 在命令提示符处输入：`oemctl start oms`。等待出现以下消息：

     ```
     "The Oracleoracle901_homeManagementServer service
     was started successfully."
     ```

  4. 关闭该 MSDOS 窗口。

- 启动 “OEM 控制台” (OEM Console) 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” (OEM Console) 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。
- 双击教师提供的指定数据库。

（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)：sys

  口令 (Password)：secure

  身份 (As)：SYSDBA
- 接着设置用于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对应条目。
- 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。

练习 3：解答

1 以 SYS AS SYSDBA 用户身份连接至数据库并关闭数据库。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

2 关闭数据库后，从 PFILE 中创建 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。

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> CREATE SPFILE='$HOME/ADMIN/PFILE/spfile$ORACLE_SID.ora'
  2  FROM PFILE='$HOME/ADMIN/PFILE/init$ORACLE_SID.ora';
File created.
```

练习 3：解答（续）

3 从操作系统查看 SPFILE。

```
SQL> !more $HOME/ADMIN/PFILE/spfile$ORACLE_SID.ora

*.background_dump_dest='/u01/home/dba01/ADMIN/BDUMP'
*.compatible='9.0.0'
*.control_files='/u01/home/dba01/ORADATA/u01/ctrl01.ctl'
*.core_dump_dest='/u01/home/dba01/ADMIN/CDUMP'
*.db_block_size=4096
*.db_cache_size=4M
*.db_domain='world'
*.db_name='dba01'
*.global_names=TRUE
*.instance_name='dba01'
*.java_pool_size='0'
*.max_dump_file_size='10240'
*.remote_login_passwordfile='exclusive'
*.service_names='dba01'
*.shared_pool_size=8M
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS'
*.user_dump_dest='/u01/home/dba01/ADMIN/UDUMP'
```

4 以 SYS AS SYSDBA 用户身份连接，然后使用 SPFILE 启动数据库。

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.

Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
Database mounted.
Database opened.
```

练习 3：解答（续）

5 a 关闭数据库然后以只读模式打开。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
Database mounted.
SQL> ALTER DATABASE OPEN READ ONLY;
Database altered.
```

b 以 HR 用户身份和 HR 口令进行连接，然后在 REGIONS 表中插入一行，如下所示：

```
INSERT INTO regions VALUES (5, ‘Mars’);
```

结果如何？

```
SQL> INSERT INTO regions VALUES ( 5, 'Mars');
INSERT INTO regions VALUES ( 5, 'Mars')
            *
ERROR at line 1:
ORA-01552: cannot use system rollback segment for non-system
tablespace
‘SAMPLE'
```

练习 3：解答（续）

5 c 将数据库重置回读写模式。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
```

6 a 以 HR 用户身份和 HR 口令进行连接，然后在 REGIONS 表中插入以下行；请勿提交或退出。

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

HR 会话

```
SQL> CONNECT hr/hr
Connected.
SQL> INSERT INTO regions VALUES (5, 'Mars');
1 row created.
```

b 在一个新的 Telnet 会话中启动 SQL*Plus。以 SYS AS SYSDBA 用户身份连接并执行。关闭事务处理。

SYS 会话

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN TRANSACTIONAL
```

练习 3：解答（续）

6 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？

HR 会话

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
ERROR:
ORA-01089: immediate shutdown in progress - no operations are
permitted
Disconnected from Oracle9i Enterprise Edition Release
9.0.0.0.0 - Beta
With the Partitioning option
JServer Release 9.0.0.0.0 - Beta (with complications)
```

注：SYS 会话现已结束，而数据库将关闭。

SYS 会话

```
Database closed.
Database dismounted.
```

练习 3：解答（续）

7 a 以 SYS 用户身份启动数据库。

SYS SESSION

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.
```

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。

HR 会话

```
SQL> CONNECT hr/hr
Connected.
```

c 以 SYS 用户身份启用受限会话。

SYS 会话

```
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
System altered.
```

练习 3：解答（续）

7 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？

HR 会话

```
SQL> SELECT *
  2  FROM   regions;
REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL> EXIT
SQL> CONNECT hr/hr
ERROR:
ORA-01035: ORACLE only available to users with RESTRICTED
SESSION privilege
Warning: You are no longer connected to ORACLE.
```

e 以 SYS 用户身份禁用受限会话。

SYS 会话

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
System altered.
```

练习 5：解答

1 下面哪些关于数据字典的描述是正确的？

a 数据字典说明了数据库及其对象。

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。

答案：以上全部选项

2 基表是使用 `catalog.sql` 脚本创建的。

a 对

b 错

答案：错，基表是使用 `sql.bsq` 脚本创建的

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。

答案：B, C, D

4 数据字典视图是静态视图。

a 对

b 错

答案：对

5 动态性能视图的信息是从控制文件中收集的。

a 对

b 错

答案：对

6 动态性能视图可帮助用户了解以下哪些方面的信息？

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？

答案：A, B, E

练习 5：解答（续）

7 请查找数据字典视图的列表。

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT table_name
  2  FROM   dictionary;

TABLE_NAME
------------------------------
ALL_ALL_TABLES
ALL_ARGUMENTS
ALL_ASSOCIATIONS
ALL_AUDIT_POLICIES
ALL_BASE_TABLE_MVIEWS
ALL_CATALOG
ALL_CLUSTERS
ALL_CLUSTER_HASH_EXPRESSIONS
ALL_COLL_TYPES
.
.
.
1063 rows selected.
```

练习 5：解答（续）

8 识别数据库名称、例程名和数据库块的大小。

提示：查询以下动态性能视图：V$DATABASE、V$THREAD 和 V$PARAMETER。

```
SQL> SELECT name
  2  FROM   v$database;
NAME
---------
DBA01
SQL> SELECT instance
  2  FROM   v$thread;
INSTANCE
----------------
dba01
SQL> SELECT value
  2  FROM   v$parameter
  3  WHERE  name ='db_block_size';
VALUE
--------------------------------
4096
```

9 列出数据文件的名称。

提示：查询动态性能视图 V$DATAFILE。

```
SQL> SELECT name
  2  FROM   v$datafile;
NAME
------------------------------------------------------------
/u01/home/dba01/ORADATA/u01/system01.dbf
/u01/home/dba01/ORADATA/u02/undotbs.dbf
/u01/home/dba01/ORADATA/u03/users01.dbf
/u01/home/dba01/ORADATA/u03/index01.dbf
/u01/home/dba01/ORADATA/u02/sample01.dbf
/u01/home/dba01/ORADATA/u01/querydata01.dbf

6 rows selected.
```

练习 5：解答（续）

10 确定组成 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。

```
SQL> SELECT file_name
  2  FROM   dba_data_files
  3  WHERE  tablespace_name = 'SYSTEM';

FILE_NAME
----------------------------------------
/u01/home/dba01/ORADATA/u01/system01.dbf
```

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 以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。

```
SQL> SELECT sum(bytes)/1024 "free space in KB"
  2  FROM   dba_free_space;
free space in KB
----------------
           42700
SQL> SELECT sum(bytes)/1024 "used space in KB"
  2  FROM   dba_segments;
used space in KB
----------------
          131992
```

练习 5：解答（续）

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，以列出数据库用户的姓名和创建日期。

```
SQL> SELECT username, created
  2  FROM   dba_users;
USERNAME                       CREATED
------------------------------ ---------
SYS                            16-APR-01
SYSTEM                         16-APR-01
OUTLN                          16-APR-01
DBSNMP                         16-APR-01
ORDSYS                         16-APR-01
ORDPLUGINS                     16-APR-01
MDSYS                          16-APR-01
HR                             16-APR-01
OE                             16-APR-01
9 rows selected.0
```

练习 6：解答

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。

```
SQL> COL name FORMAT a50
SQL> SELECT *
  2  FROM   v$controlfile;
STATUS  NAME
------- --------------------------------------
        /u01/home/dba01/ORADATA/u01/ctrl01.ctl
```

练习 6：解答

2 a 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直接更改控制文件名，也可实现此目的。
结果如何？

提示

- 以 SYS 用户身份进行连接。
- 使用 IMMEDIATE 选项关闭数据库。
- 使用操作系统命令行复制控制文件 .ctl，并使其扩展名为 .bak。
- 删除控制文件 .ctl。
- 启动数据库。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
       $HOME/ORADATA/u01/ctrl01.bak
SQL> !rm $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
ORA-00205: error in identifying controlfile, check alert
log for more info
```

b 要解决此问题，可关闭数据库、将复制的控制文件重命名为合适的名称，然后再启动数据库。

```
SQL> SHUTDOWN IMMEDIATE
ORA-01507: database not mounted
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.bak
       $HOME/ORADATA/u01/ctrl01.ctl
SQL> STARTUP
ORACLE instance started.
```

练习 6：解答（续）

3 使用目录 u02 对现有的控制文件进行多元备份，并将新控制文件命名为 ctrl02.ctl 份。确保 Oracle 服务器可以写入该新控制文件。例如，在 Unix 上可使用 chmod 660 命令。确认这两个控制文件都在使用。

提示

- 关闭数据库前，改变 SPFILE (SCOPE=SPILE) 以将该新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，该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。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET control_files =
       '$HOME/ORADATA/u01/ctrl01.ctl',
       '$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
System altered.
SQL> SHUTDOWN IMMEDIATE;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> !cp $HOME/ORADATA/u01/ctrl01.ctl
        $HOME/ORADATA/u02/ctrl02.ctl
SQL> !chmod 660 $HOME/ORADATA/u02/ctrl02.ctl
SQL> STARTUP
ORACLE instance started.
Database mounted.
Database opened.
SQL> SELECT name
  2  FROM v$controlfile;
NAME
-------------------------------------
/u01/home/dba01/ORADATA/u01/ctrl01.ctl
/u01/home/dba01/ORADATA/u02/ctrl02.ctl
```

练习 6：解答（续）

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。

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

RECORDS_TOTAL
-------------
           40
```

练习 7：解答

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

- 查询动态性能视图 V$LOGFILE。
- 使用动态性能视图 V$LOG。

```
SQL> SELECT member
  2  FROM   v$logfile;
MEMBER
------------------------------------
/u01/home/dba01/ORADATA/u03/log02a.rdo
/u01/home/dba01/ORADATA/u03/log01a.rdo
SQL> SELECT group#, members
  2  FROM   v$log;
    GROUP#    MEMBERS
---------- ----------
         1          1
         2          1
```

练习 7：解答

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。

```
SQL> SELECT log_mode
  2  FROM   v$database;
LOG_MODE
------------
NOARCHIVELOG

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

练习 7：解答（续）

3 使用以下命名约定，将重做日志成员添加到位于 u04 上的数据库内的每个组中：

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。

```
SQL> ALTER DATABASE ADD LOGFILE MEMBER
  2  '$HOME/ORADATA/u04/log01b.rdo' to Group 1,
  3  '$HOME/ORADATA/u04/log02b.rdo' to Group 2;
Database altered.
SQL> COLUMN GROUP# FORMAT 99
SQL> COLUMN MEMBER FORMAT a40
SQL> SELECT *
  2  FROM   v$logfile;

GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ----------------------------------
     2         ONLINE u01/home/dba01/ORADATA/u03/log02a.rdo
     1 STALE   ONLINE u01/home/dba01/ORADATA/u03/log01a.rdo
     1 INVALID ONLINE u01/home/dba01/ORADATA/u04/log01b.rdo
     2 INVALID ONLINE u01/home/dba01/ORADATA/u04/log02b.rdo
```

练习 7：解答（续）

4 使用以下命名约定添加一个重做日志组并验证结果。该日志组应包含两个成员，分别位于 u03 和 u04 上。

添加第 3 组： log03a.rdo 和 log03b.rdo

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。

```
SQL> ALTER DATABASE ADD
  2     LOGFILE GROUP 3('$HOME/ORADATA/u03/log03a.rdo',
  3     '$HOME/ORADATA/u04/log03b.rdo') SIZE 1024K;
Database altered.
SQL> COLUMN GROUP# FORMAT 99
SQL> COLUMN MEMBER FORMAT a40
SQL> SELECT * FROM v$logfile;
GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ----------------------------------------
    2          ONLINE /u01/home/dba01/ORADATA/u03/log02a.rdo
    1  STALE   ONLINE /u01/home/dba01/ORADATA/u03/log01a.rdo
    1  INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log01b.rdo
    2  INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log02b.rdo
    3          ONLINE  /u01/home/dba01/ORADATA/u03/log03a.rdo
    3          ONLINE  /u01/home/dba01/ORADATA/u04/log03b.rdo
6 rows selected.
SQL> SELECT group#, members FROM v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2
```

练习 7：解答（续）

5 删除第 4 步创建的重做日志组。

提示

- 如果日志文件是活动的，则使用 ALTER SYSTEM SWITCH LOGFILE。需要进行的日志切换的次数将发生变化。**注：**查询数据库以查看哪个日志文件是活动的，然后决定需要执行 ALTER SYSTEM SWITCH LOGFILE 命令的次数。
- 执行 ALTER DATABASE DROP LOGFILE GROUP 命令删除日志组。
- 查询动态性能视图 V$LOG 以对结果进行验证。
- 删除该组的操作系统文件。

```
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER DATABASE DROP LOGFILE GROUP 3;
Database altered.
SQL> SELECT group#, members
  2  FROM   v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2

SQL> !rm $HOME/ORADATA/u03/log03a.rdo
SQL> !rm $HOME/ORADATA/u04/log03b.rdo
```

练习 7：解答（续）

6 将所有联机重做日志文件的大小调整到 1024 KB。

提示

- 无法调整日志文件的大小。因此，请添加新日志并删除旧日志。
- 执行 ALTER DATABASE ADD LOGFILE GROUP 命令添加大小为 1024 KB 的新组。
- 查询动态性能视图 V$LOG 以检查活动组。
- 执行 ALTER SYSTEM SWITCH LOGFILE 命令强制执行日志切换，并将组状态更改为非活动状态。需要进行的日志切换的次数将发生变化。**注：**查询数据库以查看哪个日志文件是活动的，然后决定需要执行 ALTER SYSTEM SWITCH LOGFILE 命令的次数。
- 执行 ALTER DATABASE DROP LOGFILE 删除非活动状态组。
- 查询动态性能视图 V$LOG 以对结果进行验证。

```
SQL> ALTER DATABASE ADD LOGFILE
  2     GROUP 3( '$HOME/ORADATA/u03/log03a.rdo',
  3              '$HOME/ORADATA/u04/log03b.rdo‘ )
  4               SIZE 1024K,
  5     GROUP 4( '$HOME/ORADATA/u03/log04a.rdo',
  6              '$HOME/ORADATA/u04/log04b.rdo‘ )
  7               SIZE 1024K;
Database altered.
SQL> SELECT group#, status
  2  FROM   v$log;

GROUP# STATUS
------ ----------------
     1 ACTIVE
     2 CURRENT
     3 UNUSED
     4 UNUSED

SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
SQL> ALTER SYSTEM SWITCH LOGFILE;
System altered.
                      - 续 -
```

练习 7：解答（续）

```
                              - 续 -

SQL> ALTER DATABASE DROP LOGFILE GROUP 1, GROUP 2;
Database altered.
SQL> SELECT group#, bytes
  2  FROM   v$log;

GROUP#      BYTES
------ ----------
     3    1048576
     4    1048576
```

练习 8：解答

1 创建具有下列名称和存储的永久表空间：

a DATA01 是数据字典管理的表空间。

```
SQL> CREATE TABLESPACE data01
  2  DATAFILE '$HOME/ORADATA/u04/data01.dbf' SIZE 2M
  3  EXTENT MANAGEMENT DICTIONARY;
Tablespace created.
```

b DATA02 是具有统一区大小且在本地进行管理的表空间。确保该表空间内使用的每个区大小都是 100 KB 的倍数。

```
SQL> CREATE TABLESPACE data02
  2  DATAFILE '$HOME/ORADATA/u03/data02.dbf' SIZE 1M
  3  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;
Tablespace created.
```

c INDEX01 是具有统一区大小 4K 且在本地进行管理的表空间。当需要更多区时，允许自动扩展 500 KB。区的最大大小为 2 MB。

```
SQL> CREATE TABLESPACE index01
  2  DATAFILE '$HOME/ORADATA/u02/index01.dbf' SIZE 1M
  3  AUTOEXTEND ON NEXT 500K MAXSIZE 2M
  4  EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4K;
Tablespace created.
```

d RONLY 用于具有缺省存储的只读表。请勿在此时将表空间设为只读。

```
SQL> CREATE TABLESPACE ronly
  2  DATAFILE '$HOME/ORADATA/u01/ronly01.dbf' SIZE 1M;
Tablespace created.
```

练习 8：解答

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

```
SQL> SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
------------------------------
SYSTEM
TEMP
USERS
INDX
SAMPLE
DATA01
DATA02
DATA03
UNDO2
INDEX01
10 rows selected.
```

练习 8：解答（续）

2 为表空间 DATA02 再分配 500K 的磁盘空间。验证结果。

```
SQL> ALTER DATABASE
  2  DATAFILE '$HOME/ORADATA/u03/data02.dbf' RESIZE 1500K;
Database altered.
SQL> COLUMN name FORMAT a40
SQL> SELECT name, bytes, create_bytes
  2  FROM   v$datafile
  3  WHERE  name LIKE '%data02%‘;
NAME                                     BYTES   CREATE_BYTES
---------------------------------------- ------- ------------
/u01/home/db02/ORADATA/u03/data02.dbf 1536000 1048576
```

练习 8：解答（续）

3 将表空间 INDEX01 重定位到子目录 u06。验证 INDEX01 的重定位和状态。

提示

- 使 INDEX01 表空间脱机。
- 使用 V$DATAFILE 验证状态。
- 使用操作系统的移动命令将表空间移到 u06。
- 使用 ALTER TABLESPACE 命令重定位表空间。
- 使 INDEX01 表空间联机。
- 使用 V$DATAFILE 验证状态。

```
SQL> ALTER TABLESPACE index01 OFFLINE;
Tablespace altered.
SQL> SELECT name, status
  2  FROM v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/dba01/ORADATA/u01/system01.dbf            SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf               ONLINE
/u01/home/dba01/ORADATA/u02/sample01.dbf           ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf        ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf             ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf             ONLINE
/u01/home/dba01/ORADATA/u02/index01.dbf            OFFLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf            ONLINE
10 rows selected.
SQL> !mv $HOME/ORADATA/u02/index01.dbf
       $HOME/ORADATA/u06/index01.dbf
SQL> ALTER TABLESPACE index01
  2     RENAME DATAFILE
  3     '$HOME/ORADATA/u02/index01.dbf' TO
  4     '$HOME/ORADATA/u06/index01.dbf';
Tablespace altered.
SQL> ALTER TABLESPACE index01 ONLINE;
Tablespace altered.
                    - 续 -
```

练习 8：解答（续）

```
                        - 续 -

SQL> SELECT name, status
  2  FROM v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/dba01/ORADATA/u01/system01.dbf            SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf               ONLINE
/u01/home/dba01/ORADATA/u02/sample01.dbf           ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf        ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf             ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf             ONLINE
/u01/home/dba01/ORADATA/u06/index01.dbf            ONLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf            ONLINE
10 rows selected.
```

练习 8：管理表空间和数据文件

4 a 在表空间 RONLY 中创建一个表。将表空间 RONLY 设为只读。运行查询来验证它。

```
SQL> CREATE TABLE table1 ( x CHAR(1))
  2  TABLESPACE ronly;
Table created.
SQL> ALTER TABLESPACE ronly READ ONLY;
Tablespace altered.
SQL> SELECT name, enabled, status
  2  FROM v$datafile;
NAME                                       ENABLED    STATUS
------------------------------------------ ---------- ------
/u01/home/dba01/ORADATA/u01/system01.dbf     READ WRITE SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf      READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf      READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf         READ WRITE ONLINE
/u01/home/dba01/ORADATA/u02/example01.dbf    READ WRITE ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf  READ ONLY  ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf       READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf       READ WRITE ONLINE
/u01/home/dba01/ORADATA/u06/index01.dbf      READ WRITE ONLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf      READ ONLY  ONLINE
10 rows selected.
```

练习 8：管理表空间和数据文件

4 b 尝试创建另一个表 TABLE2。删除所创建的第一个表 TABLE1。结果如何？

```
SQL> CREATE TABLE table2 ( y CHAR(1))
  2  TABLESPACE ronly;
CREATE TABLE table2 ( y CHAR(1))
*
ERROR at line 1:
ORA-01647: tablespace 'RONLY' is read only, cannot allocate
space in it
SQL> DROP TABLE table1;
Table dropped.
```

练习 8：解答（续）

5 删除表空间 RONLY 及关联的数据文件。验证结果。

```
SQL> DROP TABLESPACE ronly INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
SQL> SELECT *
  2  FROM   v$tablespace;
    TS# NAME           INC
------- -------------- ---
      0 SYSTEM         YES
      1 UNDOTBS        YES
      3 USERS          YES
      4 INDX           YES
      5 SAMPLE         YES
      2 TEMP           YES
      6 QUERY_DATA     YES
      7 DATA01         YES
      8 DATA02         YES
      9 INDEX01        YES
10 rows selected.
SQL> !ls $HOME/ORADATA/u01/*
/u01/home/dba01/ORADATA/u01/ctrl01.bak
/u01/home/dba01/ORADATA/u01/querydata01.dbf
/u01/home/dba01/ORADATA/u01/ctrl01.ctl
/u01/home/dba01/ORADATA/u01/system01.dbf
```

练习 8：解答（续）

6 仅在内存中将 DB_CREATE_FILE_DEST 设置为 $HOME/ORADATA/u05。创建大小为 5M 的表空间 DATA03。不要指定文件位置。验证该数据文件的创建情况。

```
SQL> ALTER SYSTEM SET
       DB_CREATE_FILE_DEST='$HOME/ORADATA/u05‘
       SCOPE=MEMORY;
System altered.
SQL> CREATE TABLESPACE data03
  2    DATAFILE SIZE 5M;
Tablespace created.
SQL> SELECT *
  2  FROM   v$tablespace;
    TS# NAME           INC
------- -------------- ---
      0 SYSTEM         YES
      1 UNDOTBS        YES
      3 USERS          YES
      4 INDX           YES
      5 SAMPLE         YES
      2 TEMP           YES
      6 QUERY_DATA     YES
      7 DATA01         YES
      8 DATA02         YES
      9 INDEX01        YES
     11 DATA03         YES
11 rows selected.
SQL> !ls $HOME/ORADATA/u05
ora_data03_xg17n9nd.dbf
```

练习 9：解答

1 以 SYSTEM 用户身份运行 lab09_01.sql 脚本以创建表和索引。

```
SQL> @$HOME/STUDENT/LABS/lab09_01.sql
```

2 标识数据库中不同类型的段。

```
SQL> SELECT DISTINCT segment_type
  2  FROM   dba_segments;
SEGMENT_TYPE
------------------
CACHE
CLUSTER
INDEX
INDEX PARTITION
LOBINDEX
LOBSEGMENT
NESTED TABLE
ROLLBACK
TABLE
TABLE PARTITION
TYPE2 UNDO
11 rows selected.
```

练习 9：解答（续）

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。

```
SQL> COLUMN segment_name FORMAT a20
SQL> COLUMN segment_type FORMAT a15
SQL> SELECT segment_name,segment_type,
  2         max_extents, extents
  3  FROM   dba_segments
  4  WHERE  extents+5 > max_extents
  5  AND    segment_type<>'CACHE';

SEGMENT_NAME         SEGMENT_TYPE    MAX_EXTENTS    EXTENTS
-------------------- --------------- ----------- ----------
EMP                  TABLE                    10          8
```

4 哪些文件已为 EMP 表分配了空间？

```
SQL> SELECT DISTINCT f.file_name
  2  FROM   dba_extents e,dba_data_files f
  3  WHERE  e.segment_name='EMP'
  4  AND    e.file_id=f.file_id;

FILE_NAME
-------------------------------------
/u01/home/dba01/ORADATA/u04/data01.dbf
```

练习 9：解答（续）

5 运行 lab09_05.sql 脚本。

```
SQL> @$HOME/STUDENT/LABS/lab09_05.sql
```

6 按表空间来列出可用的空闲空间。该查询应显示各个表空间中的碎片数、总空闲空间及最大空闲区。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT tablespace_name,COUNT(*) AS fragments,
  2    SUM(bytes) AS total,
  3    MAX(bytes) AS largest
  4  FROM     dba_free_space
  5  GROUP BY tablespace_name;
TABLESPACE_NAME          FRAGMENTS      TOTAL   LARGEST
------------------------ ----------     ------  ---------
DATA01                            3     147456     126976
DATA02                            1    1433600    1433600
DATA03                            1    5177344    5177344
SAMPLE                            1    2555904    2555904
INDX                              1    5120000    5120000
INDEX01                           1     917504     917504
QUERY_DATA                        1     983040     983040
SYSTEM                            1    4943872    4943872
UNDOTBS                          15   24903680    6750208
USERS                             1    5177344    5177344
10 rows selected.
```

练习 9：解答（续）

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。

```
SQL> SELECT s.segment_name, s.segment_type, s.tablespace_name,
  2    s.next_extent
  3  FROM   dba_segments s
  4  WHERE  NOT EXISTS
  5    ( SELECT 1
  6       FROM dba_free_space f
  7       WHERE s.tablespace_name=f.tablespace_name
  8       HAVING  max(f.bytes) > s.next_extent ) ;

SEGMENT_NAME SEGMENT_TYPE TABLESPACE_NAME NEXT_EXTENT
------------ ------------ --------------- -----------
BIG_EMP      TABLE        DATA01              1048576
```

练习 10：解答

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。

```
SQL> CONNECT SYSTEM/MANAGER
Connected.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
8 rows selected.
```

练习 10：解答

2 在 $HOME/oradata/u03 中创建还原表空间 UNDO2，大小为 15M。列出表空间 UNDO2 中的还原段。

```
SQL> CREATE UNDO TABLESPACE undo2
  2  DATAFILE '$HOME/ORADATA/u03/undo2.dbf' size 15M;
Tablespace created.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDO2';
SEGMENT_NAME
--------------------
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
8 rows selected.
```

练习 10：解答（续）

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。

```
SQL> @$HOME/STUDENT/LABS/lab10_03.sql
SQL> CONNECT hr/hr
Connected.
SQL> INSERT INTO departments
  2    (department_id,department_name)
  3  VALUES (9999,'x');
1 row created.
```

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。

```
SQL> ALTER SYSTEM SET undo_tablespace='UNDO2' SCOPE=BOTH;
System altered.
```

5 以 SYS 身份删除表空间 UNDOTBS。结果如何？

```
SQL> DROP TABLESPACE undotbs
  2    INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE undotbs INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

练习 10：解答（续）

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。

```
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
8 rows selected.
SQL> SELECT a.usn,a.name,b.status
  2  FROM v$rollname a, v$rollstat b
  3  WHERE a.name
  4  IN ( SELECT segment_name
  5       FROM   dba_segments
  6       WHERE  tablespace_name = 'UNDOTBS' )
  7  AND a.usn = b.usn;

  USN NAME         STATUS
----- ------------ ---------------
    2 _SYSSMU2$    PENDING OFFLINE
```

练习 10：解答（续）

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。

```
SQL> ROLLBACK;
Rollback complete.
SQL> EXIT;
```

8 在以 SYS 的身份连接的会话中，删除表空间 UNDOTBS。结果如何？

```
SQL> DROP TABLESPACE undotbs;
DROP TABLESPACE undotbs
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

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。

```
SQL> ALTER SYSTEM SET undo_retention=0 SCOPE=MEMORY;
System altered.
SQL> DROP TABLESPACE undotbs
  2  INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
```

练习 11：解答

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。**注：**在使用 OEM 时，一定要将 DATE_OF_DELY 设置为 NULL。

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

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。

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE TABLE customers
  2    ( cust_code VARCHAR2(3),
  3      name VARCHAR2(50),
  4      region VARCHAR2(5) )
  5  TABLESPACE users;
Table created.
SQL> CREATE TABLE orders
  2    ( ord_id NUMBER(3),
  2      ord_date DATE,
  3      cust_code VARCHAR2(3),
  4      date_of_dely DATE )
  5  TABLESPACE users;
Table created.
```

练习 11：解答（续）

2 运行脚本 lab11_02.sql，将行插入到表中。

```
SQL> @$HOME/STUDENT/LABS/lab11_02.sql
```

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。

```
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  owner = 'SYSTEM'
  4  AND    segment_name = 'ORDERS'
  5  AND    segment_type = 'TABLE';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         3         25          8
```

4 检查 ORDERS 表使用的区数。

```
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         1
```

练习 11：解答（续）

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。

```
SQL> ALTER TABLE orders ALLOCATE EXTENT;
Table altered.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

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。验证是否已按照指定的区数创建该表。

```
SQL> CREATE TABLE orders2
  2  TABLESPACE users
  3  STORAGE(MINEXTENTS 10)
  4  AS SELECT * FROM orders;
Table created.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS2'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
        10
```

练习 11：解答（续）

7 截断 ORDERS 表而不释放空间，检查区数以验证区未被回收。

```
SQL> TRUNCATE TABLE orders REUSE STORAGE;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

8 截断 ORDERS2 表并释放空间。现在该表有多少个区？

```
SQL> TRUNCATE TABLE orders2;
Table truncated.
SQL> SELECT count(*)
  2  FROM   dba_extents
  3  WHERE  segment_name='ORDERS2'
  4  AND    owner='SYSTEM';

  COUNT(*)
----------
        10
```

9 运行脚本 lab11_09.sql，向 ORDERS2 表中插入一些行。

```
SQL> @$HOME/STUDENT/LABS/lab11_09.sql
```

练习 11：解答（续）

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？

```
SQL> DESCRIBE orders2;
 Name                          Null?    Type
 ----------------------------- -------- -----------------
 ORD_ID                                 NUMBER(3)
 ORD_DATE                               DATE
 CUST_CODE                              VARCHAR2(3)
 DATE_OF_DELY                           DATE
SQL> ALTER TABLE orders2
  2  SET UNUSED COLUMN date_of_dely
  3  CASCADE CONSTRAINTS;
Table altered.
SQL> DESCRIBE orders2;
 Name                                       Null?    Type
 ----------------------------- -------- -----------------
 ORD_ID                                 NUMBER(3)
 ORD_DATE                               DATE
 CUST_CODE                              VARCHAR2(3)
```

11 删除未使用的列 DATE_OF_DELY。

```
SQL> ALTER TABLE orders2
  2  DROP UNUSED COLUMNS;
Table altered.
```

12 删除 ORDERS2 表。

```
SQL> DROP TABLE orders2;
Table dropped.
```

练习 12：解答

1 您将考虑为 CUSTOMERS 表的 NAME 列和 REGION 列创建索引。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 INDEX01 表空间中。

提示：A B 树索引适合于重复值很少的列，而位图索引则适合于重复值很多的列。

CUSTOMERS 表位于 SYSTEM 方案中。

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE INDEX cust_name_idx
  2  ON customers(name)
  3  TABLESPACE index01;
Index created.
SQL> CREATE BITMAP INDEX cust_region_idx
  2  ON system.customers(region)
  3  TABLESPACE index01;
Index created.
```

2 将 CUST_REGION_IDX 索引移动到另一个表空间。

提示：可通过指定另外一个表空间来重建该索引。

```
SQL> ALTER INDEX cust_region_idx REBUILD
  2  TABLESPACE indx;
Index altered.
```

练习 12：解答

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 可获得此信息。

```
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';
   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4         17        125
```

4 重新创建 CUST_REGION_IDX 索引（而不用将其删除后再重新创建），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：重建该索引。

注：新索引所使用的空间与该索引重建后从该区所在位置中看到的空间并不相同。这是因为 Oracle 服务器会建立一个临时索引并删除旧索引，然后再重命名该临时索引。

```
SQL> ALTER INDEX cust_region_idx REBUILD;
Index altered.
SQL> SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4        142        125
```

练习 12：解答（续）

5 a 以用户 SYSTEM 的身份，运行脚本 lab12_05a.sql 以创建和填充 NUMBERS 表。

```
SQL> @$HOME/STUDENT/LABS/lab12_05a.sql
```

b 查询 NUMBERS 表以查找该表的两列中非重复值的数目。

```
SQL> SELECT count(DISTINCT no) NO,
  2    count(DISTINCT odd_even) OE
  3  FROM   numbers;

        NO         OE
---------- ----------
     10000          2
```

c 使用统一的区大小 4KB，为 NUMBERS 表的 ODD_EVEN 列和 NO 列分别创建两个 B 树索引：NUMB_OE_IDX 和 NUMB_NO_IDX。将这些索引存放在 INDEX01 表空间中。检查总的索引大小，并将块数写入到下面的框中。

提示： 使用 PCTINCREASE = 0 创建大小相等的区。从 DBA_SEGMENTS 检查为区分配的总块数。

索引	列	块
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

练习 12：解答（续）

5 c （续）

```
SQL> CREATE INDEX numb_oe_idx
  2  ON numbers(odd_even)
  3  TABLESPACE index01;
Index created.
SQL> CREATE INDEX numb_no_idx
  2  ON numbers(no)
  3  TABLESPACE index01;
Index created.
SQL> COLUMN segment_name FORMAT a15
SQL> SELECT segment_name, blocks
  2  FROM   dba_segments
  3  WHERE  segment_name LIKE 'NUMB%'
  4  AND    segment_type='INDEX';

SEGMENT_NAME        BLOCKS
--------------- ----------
NUMB_OE_IDX             40
NUMB_NO_IDX             46
```

练习 12：解答（续）

5 d 记录上面的块之后，删除索引 NUMB_OE_IDX 和 NUMB_NO_IDX。使用统一的区大小 4KB，为 NUMBERS 表的 ODD_EVEN 和 NO 列分别创建位图索引：NUMB_OE_IDX 和 NUMB_NO_IDX。将这些索引存放在 INDEX01 表空间中。重新执行查询，以从 DBA_SEGMENTS 检查给区分配的总块数。检查总的索引大小，并写入到下面的框中。

索引	列	块
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

```
SQL> DROP INDEX numb_oe_idx;
Index dropped.
SQL> DROP INDEX numb_no_idx;
Index dropped.
SQL> CREATE BITMAP INDEX numb_oe_idx
  2  ON numbers(odd_even)
  3  TABLESPACE index01;
Index created.
SQL> CREATE BITMAP INDEX numb_no_idx
  2  ON numbers(no)
  3  TABLESPACE index01;
Index created.
SQL> SELECT segment_name, blocks
  2  FROM   dba_segments
  3  WHERE  segment_name LIKE 'NUMB%'
  4  AND    segment_type='INDEX';

SEGMENT_NAME         BLOCKS
---------------  ----------
NUMB_OE_IDX               2
NUMB_NO_IDX              72
```

可以得出两类索引的基数和大小之间是什么关系吗？

答案：从结果中可以看出：位图索引是低基数列的紧凑形式，而 B 树索引则是高基数列的紧凑形式。

练习 13：解答

1 检查 lab13_01.sql 脚本。运行该脚本以创建约束。

```
SQL> @$HOME/STUDENT/LABS/lab13_01.sql
```

2 查询数据字典以：

a 检查约束，了解它们是否可以延迟以及它们的状态。

提示：使用 DBA_CONSTRAINTS 视图获取此信息。

```
SQL> COLUMN constraint_name FORMAT a25
SQL> COLUMN table_name      FORMAT a10
SQL> COLUMN constraint_type FORMAT a1
SQL> COLUMN deferrable      FORMAT a15
SQL> COLUMN status          FORMAT a10
SQL> SELECT constraint_name, table_name,
  2    constraint_type, deferrable, status
  3  FROM   dba_constraints
  4  WHERE  table_name IN
  5    ('PRODUCTS','ORDERS','CUSTOMERS')
  6  AND owner='SYSTEM';

CONSTRAINT_NAME            TABLE_NAME C DEFERRABLE      STATUS
-------------------------- ---------- - --------------- -------
-
CUSTOMERS_REGION_CK        CUSTOMERS  C NOT DEFERRABLE  ENABLED
CUSTOMERS_CUST_CODE_PK     CUSTOMERS  P DEFERRABLE      ENABLED
ORDERS_CUST_CODE_FK        ORDERS     R DEFERRABLE      ENABLED
ORDERS_DATE_OF_DELY_CK     ORDERS     C NOT DEFERRABLE  ENABLED
ORDERS_ORD_ID_PK           ORDERS     P NOT DEFERRABLE  ENABLED
PRODUCTS_PROD_CODE_UK      PRODUCTS   U DEFERRABLE
DISABLED

6 rows selected.
```

练习 13：解答（续）

2 b 检查所创建的索引名称及类型，对约束进行验证。

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。

```
SQL> SELECT index_name,table_name,uniqueness
  2  FROM   dba_indexes
  3  WHERE  index_name in
  4    ( SELECT constraint_name
  5      FROM   dba_constraints
  6      WHERE  table_name IN ('PRODUCTS', 'ORDERS',
  7                            'CUSTOMERS')
  8      AND    owner='SYSTEM'
  9      AND    constraint_type in ('P','U')
  10     ) ;

INDEX_NAME                     TABLE_NAME UNIQUENES
------------------------------ ---------- ---------
CUSTOMERS_CUST_CODE_PK         CUSTOMERS  NONUNIQUE
ORDERS_ORD_ID_PK               ORDERS     UNIQUE
```

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。

```
SQL> @$HOME/STUDENT/LABS/lab13_03.sql
```

练习 13：解答（续）

4 对 PRODUCT 表启用唯一性约束。是否成功？

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

5 a 确保添加到表中的新行不违反对 PRODUCT 表的约束。

提示：这可以通过启用 NOVALIDATE 约束来完成。

```
SQL> ALTER TABLE system.products
  2  ENABLE NOVALIDATE CONSTRAINT products_prod_code_uk;
Table altered.
```

b 查询数据字典以验证更改结果。

```
SQL> SELECT constraint_name, table_name,
  2    constraint_type, validated, status
  3  FROM   dba_constraints
  4  WHERE  table_name = 'PRODUCTS'
  5  AND    owner='SYSTEM';

CONSTRAINT_NAME           TABLE_NAME C VALIDATED     STATUS
------------------------  ---------- - ------------- ------
PRODUCTS_PROD_CODE_UK     PRODUCTS   U NOT VALIDATED ENABLED
```

练习 13：解答（续）

5 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

```
SQL> INSERT INTO system.products
  2  VALUES(4000,'Monitor',3000);
INSERT INTO system.products
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.PRODUCTS_PROD_CODE_UK)
violated
```

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。

```
SQL> CONNECT system/manager
Connected.
SQL> @?/rdbms/admin/utlexcpt
```

b 运行该命令以启用约束并捕获异常情况。

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

练习 13：解答（续）

6 c 使用 EXCEPTIONS 表中的 ROWID 列出 PRODUCTS 表中违反约束的行。不要列出 LOB 列。

```
SQL> SELECT rowid, prod_code, description
  2  FROM   system.products
  3  WHERE  rowid IN
  4    ( SELECT row_id
  5      FROM   exceptions
  6      WHERE  table_name='PRODUCTS‘ ) ;
ROWID              PROD_CODE DESCRIPTION
------------------ ---------- ------------------------------
AAABSgAAIAAAAASAAA       4000 UNIX Monitor
AAABSgAAIAAAAASAAB       4000 NT Monitor
```

d 纠正错误。

```
SQL> UPDATE   system.products
  2  SET      prod_code='4001'
  3  WHERE    rowid = ( SELECT max(row_id)
  4                     FROM   exceptions
  5                     WHERE  table_name='PRODUCTS‘ ) ;
1 row updated.
```

e 启用约束。

```
SQL> ALTER TABLE system.products
  2  ENABLE CONSTRAINT products_prod_code_uk
  3  EXCEPTIONS INTO system.exceptions;
Table altered.
```

练习 13：解答（续）

7 运行 lab13_07.sql 脚本将这些行插入到表中。插入是否成功？回退更改。

```
SQL> @$HOME/STUDENT/LABS/lab13_07.sql
SQL> INSERT INTO system.orders
  2  VALUES (800,'01-JAN-98','J01',NULL);
INSERT INTO system.orders
*
ERROR at line 1:
ORA-02291: integrity constraint
(SYSTEM.ORDERS_CUST_CODE_FK) violated - parent key not
found
SQL> INSERT INTO system.customers
  2  VALUES ('J01','Sports Store', 'East');
1 row created.
SQL> ROLLBACK;
Rollback complete.
```

8 现在检查 lab13_08 脚本。请注意，此脚本也按相同的顺序执行插入操作。运行此脚本并检查它是否已成功执行。

```
SQL> @$HOME/STUDENT/LABS/lab13_08.sql
SQL> ALTER SESSION SET CONSTRAINTS=deferred;
Session altered.
SQL> INSERT INTO system.orders
  2  VALUES (800,'01-JAN-98','J01',NULL);
1 row created.
SQL> INSERT INTO system.customers
  2  VALUES ('J01','Sports Store', 'East');
1 row created.
SQL> COMMIT;
Commit complete.
```

9 截断 CUSTOMERS 表。是否成功？

```
SQL> TRUNCATE TABLE system.customers;
TRUNCATE TABLE system.customers
                     *
ERROR at line 1:
ORA-02266: unique/primary keys in table referenced by
enabled foreign keys
```

练习 14：解答

1 运行 lab14_01.sql 脚本来创建用户 Jeff。通过运行 @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql 脚本来启用口令管理。

```
SQL> @$HOME/STUDENT/LABS/lab14_01.sql
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql
```

2 尝试将用户 Jeff 的口令更改为 Jeff。结果如何？

```
SQL> ALTER USER jeff IDENTIFIED BY jeff;
ALTER USER jeff IDENTIFIED BY jeff
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20001: Password same as or similar to user
```

3 尝试更改 Jeff 的口令，以符合口令管理格式。

提示：口令中至少应当包含一个数字、一个字符和一个标点符号。

```
SQL> ALTER USER jeff
  2  IDENTIFIED BY super1$;
User altered.
```

练习 14：解答（续）

4 更改 DEFAULT 配置文件，确保以下情况适用于分配了 DEFAULT 配置文件的用户：

- 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。
- 确保已执行给定的要求。

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。

```
SQL> ALTER PROFILE default LIMIT
  2     FAILED_LOGIN_ATTEMPTS 2
  3     PASSWORD_LIFE_TIME 30
  4     PASSWORD_REUSE_TIME 1/1440
  5     PASSWORD_GRACE_TIME 5;
Profile altered.
SQL> SELECT resource_name, limit
  2  FROM   dba_profiles
  3  WHERE  profile='DEFAULT'
  4  AND    resource_type='PASSWORD';
RESOURCE_NAME                    LIMIT
-------------------------------- ----------------
FAILED_LOGIN_ATTEMPTS            2
PASSWORD_LIFE_TIME               30
PASSWORD_REUSE_TIME              .0006
PASSWORD_REUSE_MAX               UNLIMITED
PASSWORD_VERIFY_FUNCTION         VERIFY_FUNCTION
PASSWORD_LOCK_TIME               .0006
PASSWORD_GRACE_TIME              5

7 rows selected.
```

练习 14：解答（续）

5 以用户 Jeff 的身份使用无效口令进行登录。进行两次这样的登录，然后再使用正确的口令进行登录。结果如何？

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
ERROR:
ORA-28000: the account is locked
```

练习 14：解答（续）

6 使用数据字典视图 DBA_USERS 验证用户 Jeff 是否已锁定。解除对用户 Jeff 帐户的锁定。解除对用户 Jeff 的锁定之后，再以 Jeff 的身份进行连接。

提示：执行 ALTER USER 命令解除对帐户的锁定。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SELECT username, account_status
  2  FROM   dba_users;
USERNAME        ACCOUNT_STATUS
--------------- ---------------
SYS             OPEN
SYSTEM          OPEN
OUTLN           OPEN
ORDSYS          OPEN
MDSYS           OPEN
OE              OPEN
HR              OPEN
ORDPLUGINS      OPEN
DBSNMP          OPEN
JEFF            LOCKED(TIMED)
10 rows selected.

SQL> ALTER USER jeff
  2  ACCOUNT UNLOCK;
User altered.
SQL> CONNECT jeff/super1$
Connected.
```

练习 14：解答（续）

7 对 DEFAULT 配置文件禁用口令检查。

提示：执行 ALTER PROFILE 命令禁用口令检查。

```
SQL> ALTER PROFILE default LIMIT
  2     FAILED_LOGIN_ATTEMPTS     UNLIMITED
  3     PASSWORD_LIFE_TIME        UNLIMITED
  4     PASSWORD_REUSE_TIME       UNLIMITED
  5     PASSWORD_REUSE_MAX        UNLIMITED
  6     PASSWORD_VERIFY_FUNCTION  NULL
  7     PASSWORD_LOCK_TIME        UNLIMITED
  8     PASSWORD_GRACE_TIME       UNLIMITED;

Profile altered.
```

8 以用户 Jeff 的身份使用无效口令进行登录。进行两次这样的登录，然后再使用正确的口令进行登录。结果如何？为什么？

```
SQL> CONNECT jeff/superman
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.
SQL> CONNECT jeff/super
ERROR:
ORA-01017: invalid username/password; logon denied
SQL> CONNECT jeff/super1$
Connected.
```

答案：当使用正确口令进行第三次登录时（象第 5 步中那样），帐户没有锁定。这是因为在第 7 步中禁用了 PASSWORD_VERIFY_FUNCTION（设置为 NULL）。

练习 15：解答

1 创建用户 Bob，口令为 CRUSADER。确保 Bob 所创建的任何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 能够登录，并可以在 USERS 和 INDX 这两个表空间中创建最大大小可达 1 兆字节的对象。使用 lab15_01.sql 脚本授予 Bob 创建会话的权限。

提示：请为 Bob 分配缺省表空间 USERS 和临时表空间 TEMP。

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER bob
  2     IDENTIFIED BY crusader
  3     DEFAULT TABLESPACE USERS
  4     TEMPORARY TABLESPACE temp
  5     QUOTA 1M ON USERS
  6     QUOTA 1M ON INDX;
User created.
SQL> @$HOME/STUDENT/LABS/lab15_01.sql
SQL> GRANT CREATE SESSION TO bob;
Grant succeeded.
```

练习 15：解答

2 创建用户 Emi，口令为 Mary。确保由 Emi 创建的任何对象和排序段都不是在系统表空间中创建的。

```
SQL> CREATE USER emi
  2     IDENTIFIED BY mary
  3     DEFAULT TABLESPACE users
  4     TEMPORARY TABLESPACE temp;
User created.
```

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。

```
SQL> SELECT username, default_tablespace,
  2     temporary_tablespace
  3  FROM   dba_users
  4  WHERE  username IN ('BOB', 'EMI');

USERNAME   DEFAULT_TABLESPACE   TEMPORARY_TABLE
---------- -------------------- ---------------
EMI        USERS                TEMP
BOB        USERS                TEMP
```

练习 15：解答（续）

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。

```
SQL> COLUMN tablespace_name FORMAT a15
SQL> COLUMN user            FORMAT a10
SQL> SELECT *
  2  FROM   dba_ts_quotas
  3  WHERE username = 'BOB';
TABLESPACE_NAME USERNAME BYTES  MAX_BYTES BLOCKS MAX_BLOCKS
--------------- -------- ----- ---------- ------ ----------
INDX            BOB          0    1048576      0        256
USERS           BOB          0    1048576      0        256
```

5 a 以用户 Bob 的身份更改其临时表空间。结果如何？

```
SQL> connect bob/crusader
Connected.
SQL> ALTER USER bob
  2  TEMPORARY TABLESPACE users;
ALTER USER bob
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

练习 15：解答（续）

5 b 以用户 Bob 的身份将其口令更改为 Sam。

```
SQL> CONNECT bob/crusader
Connected.
SQL> ALTER USER bob
  2  IDENTIFIED BY sam;
User altered.
```

6 以用户 SYSTEM 的身份删除 Bob 的缺省表空间限额。

```
SQL> CONNECT system/manager
Connected.
SQL> ALTER USER bob QUOTA 0 ON users;
User altered.
```

练习 15：解答（续）

7 从数据库中删除 Emi 的帐户。

```
SQL> DROP USER emi;
User dropped.
```

8 Bob 忘记了他的口令。为他指定一个口令 OLINK，并要求他在下次登录时更改其口令。

```
SQL> ALTER USER bob
  2  IDENTIFIED BY olink
  3  PASSWORD EXPIRE;
User altered.
```

练习 16：解答

1 以用户 SYSTEM 的身份创建用户 Emi，并授予她登录到数据库以及在她的方案中创建对象的权限。为她分配缺省表空间 DATA01 和临时表空间 TEMP。将她在 DATA01 上的限额设为 1M。

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER emi
  2  IDENTIFIED BY "abcd12"
  3  DEFAULT TABLESPACE data01
  4  TEMPORARY TABLESPACE temp
  5  QUOTA 1M ON data01;
User created.
SQL> GRANT create session, create table TO emi;
Grant succeeded.
```

2 a 运行脚本 lab16_02a.sql，以便以 Emi 身份进行连接并创建表 CUSTOMERS1 和 ORDERS1。

```
SQL> @$HOME/STUDENT/LABS/lab16_02a.sql;
SQL> CONNECT emi/abcd12
Connected.
CREATE TABLE customers1 (cust_code VARCHAR2(3),
                         name      VARCHAR2(50),
                         region    VARCHAR2(5) )
TABLESPACE data01;

CREATE TABLE orders1 (ord_id       NUMBER(3),
                      ord_date     DATE,
                      cust_code    VARCHAR2(3),
                      date_of_dely DATE )
TABLESPACE data01;
```

练习 16：解答（续）

2 b 以 SYSTEM 身份进行连接，并将 SYSTEM.CUSTOMERS 中的数据复制到 Emi 的 CUSTOMERS1 表中。确认已插入这些记录。

```
SQL> CONNECT system/manager
Connected.
SQL> INSERT INTO emi.customers1
  2  SELECT *
  3  FROM system.customers;
9 rows created.
SQL> SELECT * FROM emi.customers1;

CUS NAME                                               REGIO
--- -------------------------------------------------- -----
A01 TKB SPORT SHOP                                     West
A02 VOLLYRITE                                          North
A03 JUST TENNIS                                        North
A04 EVERY MOUNTAIN                                     South
A05 SHAPE UP                                           South
A06 SHAPE UP                                           West
A07 WOMENS SPORTS                                      South
A08 NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER       East
J01 Sports Store                                       East

9 rows selected.
```

c 以用户 SYSTEM 的身份授予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的权限。结果如何？

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT select ON emi.customers1 TO bob;
GRANT select ON emi.customers1 TO bob
                    *
ERROR at line 1:
ORA-01031: insufficient privileges
```

练习 16：解答（续）

3 以 Emi 的身份重新进行连接，并授予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的权限。此外，使 Bob 能向其他用户赋予选择权限。以用户 SYSTEM 的身份检查记录这些操作的数据字典视图。

提示：使用 DBA_TAB_PRIVS 进行检查。

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1
  2  TO bob WITH GRANT OPTION;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> COLUMN grantee    FORMAT a8
SQL> COLUMN owner      FORMAT a8
SQL> COLUMN table_name FORMAT a10
SQL> COLUMN grantor    FORMAT a8
SQL> COLUMN privilege  FORMAT a10
SQL> COLUMN grantable  FORMAT a3
SQL> COLUMN hierarchy   FORMAT a3
SQL> SELECT *
  2  FROM   dba_tab_privs
  3  WHERE  grantee='BOB';
GRANTEE OWNER TABLE_NAME GRANTOR  PRIVILEGE  GRANT HIERARCHY
------- ----- ---------- -------- ---------- ----- ---------
BOB      EMI  CUSTOMERS1 EMI      SELECT     YES   NO
```

练习 16：解答（续）

4 创建以 diamond1$ 标识的用户 Trevor，使其具有登录到数据库的能力。

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE USER trevor IDENTIFIED BY “diamond1$";
User created.
SQL> GRANT create session TO trevor;
Grant succeeded.
```

5 a 以 Bob 的身份使 Trevor 能够访问 Emi 的 CUSTOMERS1 表。
注：由于第 15 课第 8 步中的操作，将收到 “口令已到期” 消息。为 Bob 指定新口令 aaron$1。

```
SQL> CONNECT bob/olink
ERROR:
ORA-28001: the password has expired
Changing password for bob
New password: aaron1$
Retype new password: aaron1$
Password changed
Connected.
SQL> GRANT select ON emi.customers1 TO trevor;
Grant succeeded.
```

练习 16：解答（续）

5 b 以 Emi 的身份撤消 Bob 读取 Emi 的 CUSTOMERS1 表的权限。

```
SQL> CONNECT emi/abcd12
Connected.
SQL> REVOKE select ON customers1 FROM bob;
Revoke succeeded.
```

c 以 Trevor 的身份查询 Emi 的 CUSTOMERS1 表。结果如何？

```
SQL> CONNECT trevor/diamond1$
Connected.
SQL> SELECT *
  2  FROM emi.customers1;
FROM emi.customers1
         *
ERROR at line 2:
ORA-00942: table or view does not exist
```

练习 16：解答（续）

6 赋予 Emi 启动和关闭数据库的能力，但不赋予创建新数据库的能力。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> GRANT sysoper TO emi;
Grant succeeded.
```

练习 17：解答

1 检查数据字典视图，并列出 RESOURCE 角色的系统权限。

```
SQL> CONNECT system/manager
Connected.
SQL> COLUMN privilege FORMAT a20
SQL> COLUMN grantee   FORMAT a10
SQL> SELECT  *
  2  FROM    dba_sys_privs
  3  WHERE   grantee = 'RESOURCE';

GRANTEE    PRIVILEGE            ADM
---------- -------------------- ---
RESOURCE   CREATE CLUSTER       NO
RESOURCE   CREATE INDEXTYPE     NO
RESOURCE   CREATE OPERATOR      NO
RESOURCE   CREATE PROCEDURE     NO
RESOURCE   CREATE SEQUENCE      NO
RESOURCE   CREATE TABLE         NO
RESOURCE   CREATE TRIGGER       NO
RESOURCE   CREATE TYPE          NO
8 rows selected.
```

练习 17：解答

2 创建名为 DEV 的角色，该角色允许被授予该角色的用户能够创建表、视图并能够从 Emi 的 CUSTOMERS1 表进行选择。

```
SQL> CREATE ROLE dev;
Role created.
SQL> GRANT create table, create view TO dev;
Grant succeeded.
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers1 TO dev;
Grant succeeded.
```

练习 17：解答（续）

3 a 将 RESOURCE 角色和 DEV 角色分配给 Bob，但只允许 RESOURCE 角色在 Bob 登录时自动启用。

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT dev, resource TO bob;
Grant succeeded.
SQL> ALTER USER bob
  2     DEFAULT ROLE resource;
User altered.
```

b 为 Bob 授予读取所有数据字典信息的权限。

```
SQL> CONNECT system/manager
Connected.
SQL> GRANT select_catalog_role TO bob;
Grant succeeded.
```

练习 17：解答（续）

4 Bob 需要检查例程当前使用的还原段。以 Bob 的身份进行连接，并列出所使用的还原段。

提示：使用 SET ROLE SELECT_CATALOG_ROLE

```
SQL> CONNECT bob/aaron1$
Connected.
SQL> SET ROLE select_catalog_role;
Role set.
SQL> SELECT segment_name
  2  FROM   dba_rollback_segs
  3  WHERE  status='ONLINE';
SEGMENT_NAME
---------------
SYSTEM
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
9 rows selected.
```

5 以 SYSTEM 用户身份创建一个有关 Emi 的 CUSTOMERS 表的视图 CUST_VIEW。结果如何？

```
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2    SELECT *
  3    FROM   emi.customers;
   FROM   emi.customers
               *
ERROR at line 3:
ORA-01031: insufficient privileges
```

练习 17：解答（续）

6 以用户 Emi 的身份为 SYSTEM 授予在 CUSTOMERS1 上进行选择的权限。以 SYSTEM 用户身份创建有关 Emi 的 CUSTOMERS1 表的视图 CUST_VIEW。结果如何？

```
SQL> CONNECT emi/abcd12
Connected.
SQL> GRANT select ON customers TO system;
Grant succeeded.
SQL> CONNECT system/manager
Connected.
SQL> CREATE VIEW cust_view AS
  2     SELECT *
  3     FROM   emi.customers;

View created.
```

练习 19：解答

1 a 查看加载时所要使用的数据文件，以便熟悉控制文件和数据文件的格式。

- 如果使用 SQL*Plus，请查看以下文件：
  `lcase1.ctl`、`lcase2.ctl` 和 `lcase2.dat`
- 如果使用 Oracle Enterprise Manager，请查看以下文件：
  `OEMsqlcase1.ctl`、`OEMsqlcase2.ctl` 和 `OEMsqlcase2`。

b 以用户 HR 的身份运行 `lab19_01.sql` 脚本，创建 DEPARTMENTS2 表。

```
SQL> CONNECT hr/hr
Connected.
SQL> @HOME/STUDENT/LABS/lab19_01.sql
Table created.
```

2 a 使用常规路径方法运行 SQL*Loader，以将数据加载到 DEPARTMENTS2 表中。请使用以下控制文件：

- `lcase1.ctl`（如果使用 SQL*Plus）
- `OEMsqlcase1.ctl`（如果使用 Oracle Enterprise Manager）

文件位于 LABS 目录中。

注：此次运行使用了包含有输入数据的控制文件。

```
$ cd $HOME/STUDENT/LABS
$ sqlldr hr/hr control=lcase1.ctl log=$HOME/lcase1.log
  SQL*Loader: Release 9.0.1.0.0 - Production on Fri Oct 5
  23:18:29 2001
  (c) Copyright 2001 Oracle Corporation.  All rights
  reserved.
  Commit point reached - logical record count 27
  ...
```

练习 19：解答

2 b 使用操作系统命令查看日志文件。

```
$ more lcase1.log
   -- Note:  Path used:  Conventional
```

c 查询 DEPARTMENTS2 表以检查数据。

```
SQL> SELECT * FROM DEPTARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- --------------------
         10 Administration
         20 Marketing
         30 Purchasing
         40 Human Resources
         50 Shipping
         60 IT
         70 Public Relations
         …
         27 rows selected
```

3 删除 DEPTARTMENTS2 表中的所有记录。

```
SQL> TRUNCATE TABLE departments2;
Table truncated.
```

练习 19：解答

4 a 采用直接路径模式运行 SQL*Loader，以将数据加载到 DEPARTMENTS2 表中。请使用以下控制文件：

- `lcase2.ctl`（如果使用 SQL*Plus）
- `OE Msqlcase2.ctl`（如果使用 Oracle Enterprise Manager）

文件位于 LABS 目录中。

注：此次运行通过使用输入数据文件来加载数据。

```
$ cd $HOME/STUDENT/LABS
$ sqlldr hr/hr control=lcase2.ctl direct=true
     log=$HOME/STUDENT/LABS/lcase2.log
  SQL*Loader: Release 9.0.1.0.0 - Production on Fri Oct 5
  10:40:10 2001
  (c) Copyright 2001 Oracle Corporation.  All rights
  reserved.
  Load completed - logical record count 27.
  ...
```

b 使用操作系统命令查看日志文件。

```
$ more lcase2.log
  -- Note:  Path used:  Direct
```

c 查询 DEPARTMENTS2 表以检查数据。

```
SQL> SELECT * FROM DEPARTMENTS2;
  DEPT_ID DEPT_NAME
  ------- ------------------
        10 Administration
        20 Marketing
        30 Purchasing
        40 Human Resources
        50 Shipping
        60 IT
        70 Public Relations
        80 Sales
        90 Executive
        …
        27 rows selected.
```

练习 19：解答

5 a 以用户 HR 的身份通过从 EMPLOYEES 表中选择来创建 EMPLOYEES2 表。截断 EMPLOYEES2 表，然后从该表中选择，以验证表中没有数据。

```
SQL> CONNECT hr/hr
SQL> CREATE TABLE employees2
  2  AS SELECT * FROM employees;
Table created.
SQL> TRUNCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b 执行从 EMPLOYEES 表到 EMPLOYEES2 表的直接加载插入，并查询 EMPLOYEES2 以验证加载结果。

```
SQL> INSERT /*+ append */ INTO employees2
  2  NOLOGGING
  3  SELECT * from employees;
107 rows created.
SQL> COMMIT;
SQL> SELECT employee_id, first_name, last_name
  2  FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ------------------ -------------------------
139   John                     Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

练习 19：解答

6 a 再次截断 EMPLOYEES2 表，然后从 EMPLOYEES2 表中选择，以验证该表中没有数据。

```
SQL> TRUHCATE TABLE employees2;
Table truncated.
SQL> SELECT * FROM employees2;
no rows selected
```

b 从 EMPLOYEES 表执行一次并行直接加载插入，以恢复数据。将并行度指定为 2。验证数据。

```
SQL> ALTER SESSION ENABLE PARALLEL DML;
Session altered
SQL> INSERT /*+ parallel(employees2,2) */
  2  INTO employees2 NOLOGGING
  3  SELECT * FROM employees;
107 rows created.
SQL> COMMIT;
Commit complete.
SQL> SELECT employee_id, first_name, last_name
  2 FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ----------------- -------------------------
139   John                   Seo
140   Joshua               Patel
141   Trenna               Rajs
142   Curtis               Davies
…
107 rows selected.
```

练习 20：解答

1 检查数据库和国家字符集。

```
SQL> CONNECT sys/oracle AS SYSDBA
Connected.
SQL> SELECT parameter, value
  2  FROM   nls_database_parameters
  3  WHERE  parameter LIKE '%CHARACTERSET%';

PARAMETER                      VALUE
------------------------------ --------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         AL16UTF16
```

2 哪些是数据库字符集的有效值。

```
SQL> SELECT   value
  2  FROM     v$nls_valid_values
  3  WHERE    parameter = 'CHARACTERSET'
  4  ORDER BY value;

VALUE
-------------------
AL16UTF16
AL24UTFFSS
AL32UTF8
AR8ADOS710
AR8ADOS710T
AR8ADOS720
AR8ADOS720T
.
.
.
```

练习 20：解答（续）

3 确保在该会话中所有日期的年份都采用 4 位数。将 NLS_LANGUAGE 更改为 FRENCH。从 DUAL 中选择 sysdate。

```
SQL> ALTER SESSION SET nls_date_format = 'DD-MON-YYYY';
Session altered.
SQL> ALTER SESSION SET nls_language = FRENCH;
Session modifiée.
SQL> SELECT sysdate
  2  FROM   dual;
SYSDATE
-----------
07-OCT-2001
```

D

练习解答
Oracle Enterprise Manager

练习解答约定

下列解答是专门为使用 Oracle Enterprise Manager 和 SQL*Plus Worksheet 完成练习的学生准备的。

使用 SQL*Plus Worksheet

使用 SQL*Plus Worksheet 和 SQL*Plus 的命令（在*附录 C：SQL*Plus 练习解答* 内使用）是相同的。例外情况已在 “使用 SQL*Plus Worksheet” 标题下做了注释。

使用 “OEM 控制台” (OEM Console)

如果有特定的 “OEM 控制台” 解决方法可以代替 SQL*Plus Worksheet 使用，则 “使用 OEM 控制台” 标题下会标有相应的注释。

使用操作系统命令行

有时候需要使用操作系统命令行来执行某些任务。要退出 SQL*Plus 并进入操作系统命令行模式，就需要在 SQL*Plus 命令行中使用 !。在操作系统命令行中完成操作后，使用 Exit 即可返回到 SQL*Plus 命令行。例如：删除位于 `$HOME/ADMIN/PFILE` 中的当前口令文件。

```
SQL > !rm $HOME/ADMIN/PFILE/orapw$ORACLE_SID
$ > exit
```

运行 SQL Lab 脚本

Lab 脚本已包含在执行特定任务的 lab 练习中。此练习要完成的任务您目前可能还没有学到，或者它只是为了便于您练习。要求您运行 `.sql` 脚本时，一定要先仔细查看该脚本，然后再运行。这将有助于全面了解实际的创建结果，以及该脚本如何影响您要在练习中完成的任务。

以 SYS 身份连接

如果要求作为用户 SYS 进行连接，将始终使用 SYSDBA 权限。

```
CONNECT / AS SYSDBA
```

启动和关闭数据库

尽管 “OEM 控制台” (OEM Console) 可用来启动和关闭数据库，但是，考虑到在 Oracle classroom 中学习本课程的目的，您将使用 SQL*Plus 启动和关闭数据库。

练习 1：解答

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。

答案：D

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

答案：B

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。

答案：B, D

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

答案：D

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

答案：对

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

答案：错。PGA 是一个内存区，它包含有关单一服务器进程或单一后台进程的数据与控制信息。

练习 1：解答（续）

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

答案：C

8 列出五个必备的后台进程。

_____
_____
_____
_____
_____

答案：DBWR, LGWR, PMON, SMON, CKPT

9 请将下列进程与它们所执行的任务对号入座。

a	数据库写入器	E	协助写入数据文件的标头
b	日志写入器	C	负责例程恢复
c	系统监视器	D	进程失败后进行清理
d	进程监视器	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

答案：E, C, D, B, A

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

答案：对

11 以数据库开头，按层次顺序排列下面的结构。

a 表空间
b 区
c 段
d 数据库
e 块

答案：D, A, C, B, E

12 列出 Oracle 服务器的组件。

_____
_____

答案：Oracle 例程、Oracle 数据库

练习 1：解答（续）

13 列出 Oracle 例程的组件。

_____

_____

答案：SGA、后台进程

14 列出组成 Oracle 数据库的三种文件类型。

_____

_____

_____

答案：数据文件、控制文件、重做日志文件

练习 2：解答

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至您的帐户。
- 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 Service 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
     was started successfully."
     ```

  4. 关闭 MSDOS 窗口。

- 启动 “OEM 控制台” (OEM Console) 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” (OEM Console) 后，进入主菜单并导航到以下菜单项：

  “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**：如果该操作未成功，请向教师报告。
- 展开导航树中的 “数据库” (`Database`) 文件夹。
- 双击教师提供的指定数据库。

（转至下页）

4 使用 Oracle Management Service 启动 Oracle Enterprise Manager。（续）

- 提供以下连接信息：

  用户 (User)：`sys`

  口令 (Password)：`secure`

  身份 (As)：`SYSDBA`
- 接着设置运行作业所需的节点身份证明
- 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对应的条目。
- 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

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

- 导航到 “工具” (`Tools`) > “数据库应用程序” (`Database Applications`) > SQL*Plus Worksheet 还可通过执行以下操作从 Windows NT 菜单启动 SQL*Plus Worksheet：
- 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：每次以不同用户身份登录时（在 SQL*Plus Worksheet 内），必须在连接字符串中包括服务名。

练习 3：解答

1 以 SYS 用户身份连接至数据库并关闭数据库。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL>
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

注：尽管 “OEM 控制台” (OEM Console) 可用来启动和关闭数据库，但考虑到本课程的目的，您将使用 SQL*Plus 启动和关闭数据库。以下说明仅供您参考。

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “常规” (General) 页。
- 在 “例程状态” (Instance State) 下，选择 “关闭” (Shutdown) 选项。
- 选择 “应用” (Apply)。

2 数据库关闭后，从 PFILE 创建一个 SPFILE。将 SPFILE 置入目录 $HOME/ADMIN/PFILE，文件名格式为 spfileSID.ora。使用具体的例程名来代替 SID。从位于 $HOME/ADMIN/PFILE 的 PFILE 中创建 SPFILE。

使用 “OEM 控制台” (OEM Console)

- 选择 “对象” (Object) > “创建 spfile” (Create spfile)
- 随即显示 “从 pfile 创建 spfile” (Create an spfile from a pfile) 对话框。
- 确定要使用的 PFILE。
- 提供要创建的 SPFILE 的名称。
  示例：spfile$ORACLE_SID.ora
- 选择 “确定” (OK)

练习 3：解答（续）

3 从操作系统查看 SPFILE。

使用操作系统命令行

```
$ more $HOME/ADMIN/PFILE/spfile$ORACLE_SID.ora

*.background_dump_dest='/u01/home/dba01/ADMIN/BDUMP'
*.compatible='9.0.0'
*.control_files='/u01/home/dba01/ORADATA/u01/ctrl01.ctl'
*.core_dump_dest='/u01/home/dba01/ADMIN/CDUMP'
*.db_block_size=4096
*.db_cache_size=4M
*.db_domain='world'
*.db_name='dba01'
*.global_names=TRUE
*.instance_name='dba01'
*.java_pool_size='0'
*.max_dump_file_size='10240'
*.remote_login_passwordfile='exclusive'
*.service_names='dba01'
*.shared_pool_size=8M
*.undo_management='AUTO'
*.undo_tablespace='UNDOTBS'
*.user_dump_dest='/u01/home/dba01/ADMIN/UDUMP'
```

练习 3：解答（续）

4 以 SYS 用户身份连接，然后使用 SPFILE 启动数据库。

```
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.

Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
Database mounted.
Database opened.
```

注：尽管 “OEM 控制台” (OEM Console) 可用来启动和关闭数据库，但考虑到本课程的目的，您将使用 SQL*Plus 启动和关闭数据库。以下说明仅供您参考。

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “常规” (General) 页。
- 在 “例程状态” (Instance State) 下，选择 “打开” (Open) 选项
- 选择 “应用” (Apply)。

练习 3：解答（续）

5 a 关闭数据库然后以只读模式打开。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP MOUNT
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
Database mounted.
SQL> ALTER DATABASE OPEN READ ONLY;
Database altered.
```

注：尽管 “OEM 控制台” (OEM Console) 可用来启动和关闭数据库，但考虑到本课程的目的，您将使用 SQL*Plus 启动和关闭数据库。以下说明仅供您参考。

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “常规” (General) 页。
- 在 “例程状态” (Instance State) 下，选择 “关闭” (Shutdown) 选项。
- 选择 “应用” (Apply)。
- 从 “关闭选项” (Shutdown Options) 对话框选择 “立即” (Immediate)。
- 选择 “确定” (OK)。
- 处理完成后，单击 “关闭” (Close)。
- 在 “例程状态” (Instance State) 下，选择 “显示所有状态” (Show All States) 选项。
- 选择 “打开” (Open)。
- 选择 “应用” (Apply)。
- 从 “启动选项” (Startup Options) 对话框选择 “只读模式” (Read Only Mode)。
- 选择 “确定” (OK)。随即显示 “启动数据库” (Starting Up Database) 对话框。
- 处理完成后，选择 “关闭” (Close)。

练习 3：解答（续）

5 b 以用户 HR 口令 HR 连接，并按如下方式向 REGIONS 表插入一行：

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

结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
  示例：CONNECT HR/HR@[service name]
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
INSERT INTO regions VALUES ( 5, 'Mars');
INSERT INTO regions VALUES ( 5, 'Mars')
            *
ERROR at line 1:
ORA-01552: cannot use system rollback segment for non-system
tablespace
'SAMPLE'
```

5 c 将数据库重置回读写模式。

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> STARTUP
ORACLE instance started.
```

注：尽管 “OEM 控制台” (OEM Console) 可用来启动和关闭数据库，但考虑到本课程的目的，您将使用 SQL*Plus 启动和关闭数据库。以下说明仅供您参考。

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “常规” (General)页。
- 在 “例程状态” (Instance State) 下，选择 “关闭” (Shutdown) 选项。
- 选择 “应用” (Apply)。
- 在 “例程状态” (Instance State) 下，选择 “显示所有状态” (Show All States) 选项。
- 选择 “打开” (Open)。
- 选择 “确定” (OK)。随即显示 “启动数据库” (Starting Up Database) 对话框。
- 选择 “关闭” (Close)。

练习 3：解答（续）

6 a 以用户 HR 口令 HR 连接，并将下面的行插入 REGIONS 表；

不要提交或退出。

```
INSERT INTO regions VALUES ( 5, ‘Mars’ );
```

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
HR SESSION
CONNECT hr/hr@[service name]
Connected.
INSERT INTO regions VALUES (5, 'Mars');
1 row created.
```

b 在一个新的 telnet 会话中启动 SQL*Plus。以用户 SYS 身份连接并执行关闭事务处理。

```
SYS SESSION
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN TRANSACTIONAL
```

练习 3：解答（续）

6 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？

HR SESSION

```
ROLLBACK;
Rollback complete.
EXIT;
ERROR:
ORA-01089: immediate shutdown in progress - no operations are
permitted
Disconnected
```

注：此时 SYS 会话将结束，数据库也随之关闭。

SYS SESSION

```
Database closed.
Database dismounted.
```

练习 3：解答（续）

7 **注**：对于下列练习，请使用 SQL*Plus 会话。

a 以用户 SYS 身份启动数据库。

```
SYS SESSION
SQL> CONNECT / AS SYSDBA
Connected to an idle instance.
SQL> STARTUP
ORACLE instance started.
```

b 以用户 HR 身份启动另一个会话。

注：使这两个 SQL*Plus 会话保持打开状态，一个会话以用户 SYS 身份打开，另一个会话以用户 HR 身份打开。

```
HR SESSION
SQL> CONNECT hr/hr
Connected.
```

c 以 SYS 用户身份启用受限会话。

```
SYS SESSION
SQL> ALTER SYSTEM ENABLE RESTRICTED SESSION;
System altered.
```

练习 3：解答（续）

7 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？

```
HR SESSION
SQL> SELECT *
  2  FROM   regions;
REGION_ID REGION_NAME
---------- -------------------------
         1 Europe
         2 Americas
         3 Asia
         4 Middle East and Africa
SQL> EXIT
SQL> CONNECT hr/hr
ERROR:
ORA-01035: ORACLE only available to users with RESTRICTED
SESSION privilege
Warning: You are no longer connected to ORACLE.
```

e 以 SYS 用户身份禁用受限会话。

```
SYS SESSION
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
System altered.
```

注：尽管可使用 “OEM 控制台” (OEM Console) 将数据库置于受限模式，但考虑到本课程的目的，您将使用 SQL*Plus。以下说明仅供您参考。

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “常规” (General) 页。
- 在 “数据库和例程信息” (Database and Instance Information) 下。为 “受限模式” (Restricted Mode) 选项选择 “是” (Yes)。
- 选择 “应用” (Apply)。

练习 5：解答

1 下面哪些关于数据字典的描述是正确的？

a 数据字典说明了数据库及其对象。

b 数据字典包括两种类型的对象：基表和数据字典视图。

c 数据字典是一组表。

d 数据字典记录和验证了它所关联的数据库的有关信息。

答案：全部正确

2 基表是使用 `catalog.sql` 脚本创建的。

a 对

b 错

答案：错，应该是使用 `sql.bsq` 脚本创建的

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。

答案：B, C, D

4 数据字典视图是静态视图。

a 对

b 错

答案：对

5 动态性能视图的信息是从控制文件中收集的。

a 对

b 错

答案：对

6 动态性能视图可帮助用户了解以下哪些方面的信息？

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？

答案：A, B, E

练习 5：解答（续）

7 请查找数据字典视图的列表。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYSTEM/MANAGER@[service name]
Connected.
SELECT table_name FROM dictionary;

TABLE_NAME
------------------------------
ALL_ALL_TABLES
ALL_ARGUMENTS
ALL_ASSOCIATIONS
ALL_AUDIT_POLICIES
ALL_BASE_TABLE_MVIEWS
ALL_CATALOG
ALL_CLUSTERS
ALL_CLUSTER_HASH_EXPRESSIONS
ALL_COLL_TYPES
.
.
.
1063 rows selected.
```

练习 5：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 输入每条命令后，选择 “执行” (Execute)。

```
SELECT name FROM v$database;
NAME
---------
DBA01

SELECT instance FROM v$thread;
INSTANCE
---------
dba01

SELECT value FROM v$parameter WHERE name='db_block_size';
VALUE
---------
4096
```

9 列出数据文件的名称。

提示：查询动态性能视图 V$DATAFILE。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT name FROM v$datafile;
NAME
----------------------------------------------
/u01/home/dba01/ORADATA/u01/system01.dbf
/u01/home/dba01/ORADATA/u02/undotbs.dbf
/u01/home/dba01/ORADATA/u03/users01.dbf
/u01/home/dba01/ORADATA/u03/index01.dbf
/u01/home/dba01/ORADATA/u02/sample01.dbf
/u01/home/dba01/ORADATA/u01/querydata01.dbf

6 rows selected.
```

练习 5：解答（续）

10 确定组成 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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT file_name FROM dba_data_files
WHERE  tablespace_name = 'SYSTEM';

FILE_NAME
----------------------------------------
/u01/home/dba01/ORADATA/u01/system01.dbf
```

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 以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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 输入每条命令后，选择 “执行” (Execute)。

```
SELECT sum(bytes)/1024 "free space in KB"
FROM dba_free_space;
free space in KB
----------------
           42700

SELECT sum(bytes)/1024 "used space in KB“
FROM   dba_segments;
used space in KB
----------------
          131992
```

练习 5：解答（续）

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，以列出数据库用户的姓名和创建日期。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT username, created FROM dba_users;
USERNAME                       CREATED
------------------------------ ---------
SYS                            07-JUN-01
SYSTEM                         07-JUN-01
ORDPLUGINS                     07-JUN-01
OUTLN                          07-JUN-01
DBSNMP                         07-JUN-01
ORDSYS                         07-JUN-01
MDSYS                          07-JUN-01
HR                             07-JUN-01
OE                             07-JUN-01

9 rows selected.
```

练习 6：解答

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
COL name FORMAT a50
SELECT * FROM   v$controlfile;
STATUS  NAME
------- ----------------------------------------
        /u01/home/dba01/ORADATA/u01/ctrl01.ctl
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “控制文件” (Controlfile)。
- 控制文件将显示在右窗口中

练习 6：解答

2 a 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直接更改控制文件名，也可实现此目的。结果如何？

提示

- 以 SYS 用户身份进行连接。
- 使用 IMMEDIATE 选项关闭数据库。
- 使用操作系统命令行复制控制文件 .ctl，并使其扩展名为 .bak。
- 删除控制文件 .ctl。
- 启动数据库。

注：考虑到教室环境中该 lab 的目的，需使用 SQL*Plus 关闭数据库。

使用 SQL*Plus

```
SQL> CONNECT / AS SYSDBA
Connected.
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

使用操作系统命令行

```
$ cp $HOME/ORADATA/u01/ctrl01.ctl
     $HOME/ORADATA/u01/ctrl01.bak
$ rm $HOME/ORADATA/u01/ctrl01.ctl
```

使用 SQL*Plus

```
SQL> STARTUP
ORACLE instance started.
Total System Global Area   21790412 bytes
Fixed Size                   278220 bytes
Variable Size              16777216 bytes
Database Buffers            4194304 bytes
Redo Buffers                 540672 bytes
ORA-00205: error in identifying controlfile, check alert
log for more info
```

练习 6：解答

2 b 要解决该问题，需关闭数据库，将复制的控制文件重命名为适当的名称，然后再启动数据库。

注：考虑到教室环境的需要，需使用 SQL*Plus 关闭数据库。

使用 **SQL*Plus**

```
SQL> SHUTDOWN IMMEDIATE
ORA-01507: database not mounted
ORACLE instance shut down.
```

使用操作系统命令行

```
cp $HOME/ORADATA/u01/ctrl01.bak
   $HOME/ORADATA/u01/ctrl01.ctl
```

使用 **SQL*Plus**

```
SQL> STARTUP
ORACLE instance started.
```

练习 6：解答（续）

3 使用目录 u02 对现有的控制文件进行多元备份，并将新的控制文件命名为 ctrl02.ctl。确保 Oracle 服务器可以写入新的控制文件。例如，在 UNIX 上可使用 chmod 660 命令。确认这两个控制文件都在使用。

提示

- 关闭数据库前，改变 SPFILE (SCOPE = SPILE) 以将这个新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，该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。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYS/ORACLE[service name] AS SYSDBA
Connected.
ALTER SYSTEM SET control_files =
'$HOME/ORADATA/u01/ctrl01.ctl',
'$HOME/ORADATA/u02/ctrl02.ctl' SCOPE=SPFILE;
System altered.
```

使用 SQL*Plus

注：考虑到教室环境的需要，需使用 SQL*Plus 关闭数据库。

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
```

练习 6：解答（续）

3 （续）

使用操作系统命令行

```
$ cp $HOME/ORADATA/u01/ctrl01.ctl
     $HOME/ORADATA/u02/ctrl02.ctl
$ chmod 660 $HOME/ORADATA/u02/ctrl02.ctl
```

使用 **SQL*Plus**

注：考虑到教室环境的需要，需使用 SQL*Plus 关闭数据库。

```
SQL> STARTUP
ORACLE instance started.
Database mounted.
Database opened.
```

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT name FROM v$controlfile;
NAME
---------------------------------------
/u01/home/dba01/ORADATA/u01/ctrl01.ctl
/u01/home/dba01/ORADATA/u02/ctrl02.ctl
```

练习 6：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT records_total
FROM   v$controlfile_record_section
WHERE  type = 'DATAFILE';

RECORDS_TOTAL
-------------
          40
```

练习 6：解答

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

- 查询动态性能视图 V$LOGFILE。
- 使用动态性能视图 V$LOG。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT member FROM v$logfile;
MEMBER
-------------------------------------
/u01/home/dba01/ORADATA/u03/log02a.rdo
/u01/home/dba01/ORADATA/u03/log01a.rdo

SELECT group#, members FROM v$log;
    GROUP#    MEMBERS
---------- ----------
         1          1
         2          1
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “重做日志组” (Redo Log Groups)。
- 展开 “重做日志组” (Redo Log Groups) 文件夹查看组的数目。
- 选择一个组以查看组中的成员数及其位置。

练习 7：解答

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT log_mode FROM v$database;
LOG_MODE
------------
NOARCHIVELOG

SELECT archiver FROM v$instance;
ARCHIVE
-------
STOPPED
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “活动日志模式：NOARCHIVELOG” (Archive Log Mode: NOARCHIVELOG)
- 选择 “全部初始化参数” (ALL INITIALIZATION PARAMETERS)，参数 LOG_ARCHIVE_START 将指出是否启用了归档。

练习 7：解答（续）

3 使用以下命名约定向数据库中位于 u04 的每个组

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 命令将重做日志成员添加到每个组。
- 查询动态性能视图 V$LOGFILE 以对结果进行验证。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER DATABASE ADD LOGFILE MEMBER
'$HOME/ORADATA/u04/log01b.rdo' to Group 1,
'$HOME/ORADATA/u04/log02b.rdo' to Group 2;
Database altered.

COLUMN GROUP# FORMAT 99
COLUMN MEMBER FORMAT a40

SELECT * FROM v$logfile;

GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ----------------------------------------
     2         ONLINE  /u01/home/dba01/ORADATA/u03/log02a.rdo
     1 STALE   ONLINE  /u01/home/dba01/ORADATA/u03/log01a.rdo
     1 INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log01b.rdo
     2 INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log02b.rdo
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “重做日志组” (Redo Log Groups)。
- 选择重做日志组并在右窗口的电子表格中输入新的日志文件：“文件名” (File Name) 和 “文件目录” (File Directory)。
- 单击 “应用” (Apply)。

练习 7：解答（续）

4 使用以下命名约定添加一个重做日志组并验证结果。该日志组应包含两个成员，分别位于 u03 和 u04 上。

添加第 3 组：log03a.rdo 和 log03b.rdo

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER DATABASE ADD
LOGFILE GROUP 3('$HOME/ORADATA/u03/log03a.rdo',
'$HOME/ORADATA/u04/log03b.rdo') SIZE 1024K;
Database altered.

COLUMN GROUP# FORMAT 99
COLUMN MEMBER FORMAT a40

SELECT * FROM v$logfile;
GROUP# STATUS  TYPE    MEMBER
------ ------- ------- ------------------------------------
    2          ONLINE  /u01/home/dba01/ORADATA/u03/log02a.rdo
    1  STALE   ONLINE  /u01/home/dba01/ORADATA/u03/log01a.rdo
    1  INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log01b.rdo
    2  INVALID ONLINE  /u01/home/dba01/ORADATA/u04/log02b.rdo
    3          ONLINE  /u01/home/dba01/ORADATA/u03/log03a.rdo
    3          ONLINE  /u01/home/dba01/ORADATA/u04/log03b.rdo
6 rows selected.

SELECT group#, members FROM v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “重做日志组” (Redo Log Groups)。
- 单击鼠标右键，从弹出的菜单中使用 “创建” (Create) 输入重做日志成员的名称和位置。
- 单击 “创建” (Create)。

练习 7：解答（续）

5 删除第 4 步创建的重做日志组。

提示

- 如果日志文件是活动的，则使用 ALTER SYSTEM SWITCH LOGFILE。需要进行的日志切换的次数将发生变化。注：查询数据库以查看哪个日志文件是活动的，然后决定需要执行 ALTER SYSTEM SWITCH LOGFILE 命令的次数。
- 执行 ALTER DATABASE DROP LOGFILE GROUP 命令删除日志组。
- 查询动态性能视图 V$LOG 以对结果进行验证。
- 删除该组的操作系统文件。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER SYSTEM SWITCH LOGFILE;
System altered.
ALTER SYSTEM SWITCH LOGFILE;
System altered.
ALTER SYSTEM SWITCH LOGFILE;
System altered.
ALTER DATABASE DROP LOGFILE GROUP 3;
Database altered.
SELECT group#, members FROM v$log;
GROUP#    MEMBERS
------ ----------
     1          2
     2          2
     3          2
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “重做日志组” (Redo Log Groups)。
- 如果 Group 3 是活动的，则选择该组，然后单击鼠标右键，从弹出的菜单中选择 “切换日志文件” (Switch Logfile)。
- 选择 Group 3，然后单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。
- 删除该组的操作系统文件。见下文。

使用操作系统命令行

```
$ rm $HOME/ORADATA/u03/log03a.rdo
$ rm $HOME/ORADATA/u04/log03b.rdo
```

练习 7：解答（续）

6 将所有联机重做日志文件的大小调整到 1024 KB。

提示

- 无法调整日志文件的大小。因此，请添加新日志并删除旧日志。
- 执行 ALTER DATABASE ADD LOGFILE GROUP 命令添加大小为 1024 KB 的新组。
- 查询动态性能视图 V$LOG 以检查活动组。
- 执行 ALTER SYSTEM SWITCH LOGFILE 命令强制执行日志切换，并将组状态更改为非活动状态。需要进行的日志切换的次数将发生变化。注：查询数据库以查看哪个日志文件是活动的，然后决定需要执行 ALTER SYSTEM SWITCH LOGFILE 命令的次数。
- 执行 ALTER DATABASE DROP LOGFILE 删除非活动状态组。
- 查询动态性能视图 V$LOG 以对结果进行验证。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER DATABASE ADD LOGFILE
GROUP 3( '$HOME/ORADATA/u03/log03a.rdo',
'$HOME/ORADATA/u04/log03b.rdo‘ )
SIZE 1024K,
GROUP 4( '$HOME/ORADATA/u03/log04a.rdo',
'$HOME/ORADATA/u04/log04b.rdo‘ )
SIZE 1024K;
Database altered.

SELECT group#, status FROM v$log;

GROUP# STATUS
------ ----------------
     1 ACTIVE
     2 CURRENT
     3 UNUSED
     4 UNUSED


- 续 -
```

练习 7：解答（续）

6 （续）

```
- 续 -

ALTER SYSTEM SWITCH LOGFILE;
System altered.
ALTER SYSTEM SWITCH LOGFILE;
System altered.
ALTER SYSTEM SWITCH LOGFILE;
System altered.

ALTER DATABASE DROP LOGFILE GROUP 1, GROUP 2;
Database altered.

SELECT group#, bytes FROM v$log;

GROUP#      BYTES
------ ----------
     3    1048576
     4    1048576
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “重做日志组” (Redo Log Groups)。
- 单击鼠标右键，从弹出的菜单中使用 “创建” (Create) 添加两个日志组。
- 单击鼠标右键，从弹出的菜单中选择 “切换日志文件” (Switch Logfile)，直到第 1 组和第 2 组变为非活动状态。
- 单击鼠标右键，从弹出的菜单中选择 “删除” (Remove) 以删除第 1 组和第 2 组。
- 删除该组的操作系统文件。见下文。

练习 8：解答

1 创建具有下列名称和存储的永久表空间：

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “表空间” (Tablespace)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create) 以添加表空间。

a DATA01 是数据字典管理的表空间。

```
CREATE TABLESPACE data01
DATAFILE '$HOME/ORADATA/u04/data01.dbf' SIZE 2M
EXTENT MANAGEMENT DICTIONARY;
Tablespace created.
```

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。

```
CREATE TABLESPACE data02
DATAFILE '$HOME/ORADATA/u03/data02.dbf' SIZE 1M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 100K;
Tablespace created.
```

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。

```
CREATE TABLESPACE index01
DATAFILE '$HOME/ORADATA/u02/index01.dbf' SIZE 1M
AUTOEXTEND ON NEXT 500K MAXSIZE 2M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 4K;
Tablespace created.
```

d RONLY 用于具有缺省存储的只读表。请勿在此时将表空间设为只读。

```
CREATE TABLESPACE ronly
DATAFILE '$HOME/ORADATA/u01/ronly01.dbf' SIZE 1M;
Tablespace created.
```

练习 8：解答

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

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT tablespace_name FROM dba_tablespaces;

TABLESPACE_NAME
------------------------
SYSTEM
TEMP
USERS
INDX
SAMPLE
DATA01
DATA02
DATA03
UNDO2
INDEX01
10 rows selected.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “表空间” (Tablespace)。
- 单击 “表空间” (Tablespaces)，右窗口中将显示所有表空间。

练习 8：解答（续）

2 为表空间 DATA02 再分配 500K 的磁盘空间。验证结果。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER DATABASE
DATAFILE '$HOME/ORADATA/u03/data02.dbf' RESIZE 1500K;
Database altered.

COLUMN name FORMAT a40

SELECT name, bytes, create_bytes
FROM   v$datafile
WHERE  name LIKE '%data02%';
NAME                                     BYTES   CREATE_BYTES
---------------------------------------- ------- ------------
/u01/home/db02/ORADATA/u03/data02.dbf 1536000 1048576
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “表空间” (Tablespaces) > DATA02 > “数据文件” (Datafiles)
- 在 “常规” (General) 选项卡中，将文件的大小指定为 1500 KB。
- 单击 “应用” (Apply)。

练习 8：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLESPACE index01 OFFLINE;
Tablespace altered.
SELECT name, status FROM v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/dba01/ORADATA/u01/system01.dbf             SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf              ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf              ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf                 ONLINE
/u01/home/dba01/ORADATA/u02/sample01.dbf             ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf          ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf               ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf               ONLINE
/u01/home/dba01/ORADATA/u02/index01.dbf              OFFLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf              ONLINE
10 rows selected.
```

练习 8：解答（续）

3 （续）

使用操作系统命令行

```
$ mv $HOME/ORADATA/u02/index01.dbf
  $HOME/ORADATA/u06/index01.dbf
```

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLESPACE index01 RENAME DATAFILE
'$HOME/ORADATA/u02/index01.dbf' TO
'$HOME/ORADATA/u06/index01.dbf';
Tablespace altered.

ALTER TABLESPACE index01 ONLINE;
Tablespace altered.

SELECT name, status FROM v$datafile;
NAME                                               STATUS
-------------------------------------------------- -------
/u01/home/dba01/ORADATA/u01/system01.dbf            SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf            ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf               ONLINE
/u01/home/dba01/ORADATA/u02/sample01.dbf           ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf        ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf             ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf             ONLINE
/u01/home/dba01/ORADATA/u06/index01.dbf            ONLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf            ONLINE
10 rows selected.
```

练习 8：解答（续）

3 （续）

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “表空间” (Tablespaces) > INDEX01。
- 从 “常规” (General) 选项卡选择 “脱机” (Offline) 按钮。
- 单击 “应用” (Apply)，或者单击鼠标右键，从弹出的菜单中选择 “脱机” (Offline) > “正常” (Normal)。
- 单击此对话框中的 “是” (Yes)。
- 登录到服务器并移动数据文件。见上文。
- 返回到 INDEX01 的 “常规” (General) 选项卡，并将 “文件目录” (File Directory) 更新为 u06。
- 单击 “应用” (Apply)。
- 从 “常规” (General) 选项卡选择 “联机” (Online) 按钮。
- 单击 “应用” (Apply)。

注：也可以通过下列方式进行更名操作：

- 导航到 “存储” (Storage) > “数据文件” (Datafiles)，然后选择属于 INDEX01 表空间的文件。

练习 8：管理表空间和数据文件

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE TABLE table1 ( x CHAR(1)) TABLESPACE ronly;
Table created.
ALTER TABLESPACE ronly READ ONLY;
Tablespace altered.
SELECT name, enabled, status FROM v$datafile;
NAME                                          ENABLED    STATUS
--------------------------------------------- ---------- ------
/u01/home/dba01/ORADATA/u01/system01.dbf       READ WRITE SYSTEM
/u01/home/dba01/ORADATA/u02/undotbs.dbf        READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/users01.dbf        READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/indx.dbf           READ WRITE ONLINE
/u01/home/dba01/ORADATA/u02/example01.dbf      READ WRITE ONLINE
/u01/home/dba01/ORADATA/u01/querydata01.dbf    READ ONLY  ONLINE
/u01/home/dba01/ORADATA/u04/data01.dbf         READ WRITE ONLINE
/u01/home/dba01/ORADATA/u03/data02.dbf         READ WRITE ONLINE
/u01/home/dba01/ORADATA/u06/index01.dbf        READ WRITE ONLINE
/u01/home/dba01/ORADATA/u01/ronly01.dbf        READ ONLY  ONLINE
10 rows selected.
```

使用 “OEM 控制台” (OEM Console)

- 通过 “方案” (Schema) > “表” (Table) 创建表。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” 选项卡中填写创建表所需的相应信息。
- 单击 “创建” (Create)。
- 通过 “存储” (Storage) > “表空间” (Tablespace) > RONLY 将表空间设为只读。
- 单击鼠标右键，从弹出的菜单中选择 “只读模式” (Make Read-Only)。
- 在此对话框中选择 “是” (Yes)。
- 从 “常规” (General) 页中验证表空间是否为只读。

练习 8：管理表空间和数据文件

4 b 尝试创建另一个表 TABLE2。删除创建的第一个表 TABLE1。结果如何？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE TABLE table2 ( y CHAR(1)) TABLESPACE ronly;
CREATE TABLE table2 ( y CHAR(1))
*
ERROR at line 1:
ORA-01647: tablespace 'RONLY' is read only, cannot allocate
space in it
DROP TABLE table1;
Table dropped.
```

使用 “OEM 控制台” (OEM Console)

- 通过 “方案” (Schema) > “表” (Table) 创建另一个表。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” 页中填写创建表所需的相应信息。
- 单击 “创建” (Create)。
- 在 “表” (Table) 目录中选择表，然后单击鼠标右键，从弹出的菜单中选择 “删除” (Remove) 以删除该表。

练习 8：解答（续）

5 删除表空间 RONLY 及关联的数据文件。验证结果。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DROP TABLESPACE ronly INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
SELECT * FROM v$tablespace;
    TS# NAME           INC
------- -------------- ---
      0 SYSTEM         YES
      1 UNDOTBS        YES
      3 USERS          YES
      4 INDX           YES
      5 SAMPLE         YES
      2 TEMP           YES
      6 QUERY_DATA     YES
      7 DATA01         YES
      8 DATA02         YES
      9 INDEX01        YES
10 rows selected.
```

使用操作系统命令行

```
ls $HOME/ORADATA/u01/*
/u01/home/dba01/ORADATA/u01/ctrl01.bak
/u01/home/dba01/ORADATA/u01/querydata01.dbf
/u01/home/dba01/ORADATA/u01/ctrl01.ctl
/u01/home/dba01/ORADATA/u01/system01.dbf
```

使用 “OEM 控制台” (OEM Console)

- “存储” (Storage) > “表空间” (Tablespaces)。
- 选择 RONLY 表空间，然后单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。
- 通过从树中选择 “表空间” (Tablespace) 验证是否已经删除该表空间。
- 删除操作系统文件：$rm $HOME/ORADATA/u01/RONLY.dbf

练习 8：解答（续）

6 仅在内存中将 DB_CREATE_FILE_DEST 设置为 $HOME/ORADATA/u05。创建大小为 5M 的表空间 DATA03。不要指定文件位置。验证该数据文件的创建情况。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER SYSTEM SET
DB_CREATE_FILE_DEST='$HOME/ORADATA/u05‘ SCOPE=MEMORY;
System altered.
CREATE TABLESPACE data03 DATAFILE SIZE 5M;
Tablespace created.
SELECT * FROM v$tablespace;
    TS# NAME           INC
------- -------------- ---
      0 SYSTEM         YES
      1 UNDOTBS        YES
      3 USERS          YES
      4 INDX           YES
      5 SAMPLE         YES
      2 TEMP           YES
      6 QUERY_DATA     YES
      7 DATA01         YES
      8 DATA02         YES
      9 INDEX01        YES
     11 DATA03         YES
11 rows selected.
```

使用操作系统命令行

```
ls $HOME/ORADATA/u05
ora_data03_xg17n9nd.dbf
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “例程” (Instance) > “配置” (Configuration)。
- 选择 “全部初始化参数” (ALL INITIALIZATION PARAMETERS)。
- 验证是否已选择 “运行” (Running) 单选按钮。
- 向下移至 DB_CREAT_FILE_DEST，然后将位置改为 u05。
- 单击 “确定” (OK)。
- 将出现一个对话框，指出参数已被更改。单击 “确定” (OK)。
- 用 SQL*Plus Worksheet 代替 “OEM 控制台” (OEM Console) 来创建表空间。这样 Oracle 就可以提供位置以及 “Oracle 管理文件” (Oracle Managed File) 的名称。

练习 9：解答

1 以 SYSTEM 用户身份运行 lab09_01.sql 脚本以创建表和索引。

使用 SQL*Plus Worksheet

- 主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab09_01.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

2 标识数据库中不同类型的段。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT DISTINCT segment_type FROM dba_segments;
SEGMENT_TYPE
------------------
CACHE
CLUSTER
INDEX
INDEX PARTITION
LOBINDEX
LOBSEGMENT
NESTED TABLE
ROLLBACK
TABLE
TABLE PARTITION
TYPE2 UNDO
11 rows selected.
```

练习 9：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
COLUMN segment_name FORMAT a20
COLUMN segment_type FORMAT a15

SELECT segment_name,segment_type, max_extents, extents
FROM dba_segments
WHERE  extents+5 > max_extents
AND    segment_type<>'CACHE';

SEGMENT_NAME         SEGMENT_TYPE    MAX_EXTENTS    EXTENTS
-------------------- --------------- ----------- ----------
EMP                  TABLE                    10          8
```

4 哪些文件已为 EMP 表分配了空间？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT DISTINCT f.file_name
FROM   dba_extents e,dba_data_files f
WHERE  e.segment_name='EMP'
AND    e.file_id=f.file_id;

FILE_NAME
-----------------------------------
/u01/home/dba01/ORADATA/u04/data01.dbf
```

练习 9：解答（续）

5 运行 lab09_05.sql 脚本。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab09_05.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。

6 按表空间来列出可用的空闲空间。该查询应显示各个表空间中的碎片数、总空闲空间及最大空闲区。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYS/ORACLE@[service name] AS SYSDBA;
Connected.

SELECT tablespace_name,COUNT(*) AS fragments,
SUM(bytes) AS total, MAX(bytes) AS largest
FROM dba_free_space GROUP BY tablespace_name;
TABLESPACE_NAME          FRAGMENTS      TOTAL   LARGEST
------------------------ ----------     ------  ---------
DATA01                            3     147456     126976
DATA02                            1    1433600    1433600
DATA03                            1    5177344    5177344
SAMPLE                            1    2555904    2555904
INDX                              1    5120000    5120000
INDEX01                           1     917504     917504
QUERY_DATA                        1     983040     983040
SYSTEM                            1    4943872    4943872
UNDOTBS                          15   24903680    6750208
USERS                             1    5177344    5177344
10 rows selected.
```

练习 9：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT s.segment_name, s.segment_type, s.tablespace_name,
s.next_extent
FROM   dba_segments s
WHERE  NOT EXISTS
( SELECT 1 FROM dba_free_space f
WHERE s.tablespace_name=f.tablespace_name
HAVING  max(f.bytes) > s.next_extent ) ;

SEGMENT_NAME SEGMENT_TYPE TABLESPACE_NAME NEXT_EXTENT
------------ ------------ --------------- -----------
BIG_EMP      TABLE        DATA01              1048576
```

练习 10：解答

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。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYSTEM/MANAGER@[service name]
Connected.
SELECT segment_name FROM   dba_rollback_segs
WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
8 rows selected.
```

练习 10：解答

2 在 $HOME/oradata/u03 中创建还原表空间 UNDO2，大小为 15M。列出表空间 UNDO2 中的还原段。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE UNDO TABLESPACE undo2
DATAFILE '$HOME/ORADATA/u03/undo2.dbf' size 15M;
Tablespace created.

SELECT segment_name
FROM   dba_rollback_segs
WHERE  tablespace_name = 'UNDO2';
SEGMENT_NAME
--------------------
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
8 rows selected.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “存储” (Storage) > “表空间” (Tablespaces) 。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页中创建表空间 UNDO2。
- 选择 “还原” (Undo) 作为表空间的类型，然后单击 “创建” (Create)。

练习 10：解答（续）

3 在新的 telnet 会话中启动 SQL*Plus，然后以用户 HR 的身份连接并运行脚本 lab10_03.sql，以便向表 DEPARTMENTS 插入一行。不要提交、回退或退出会话。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab10_03.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER SYSTEM SET undo_tablespace='UNDO2' SCOPE=BOTH;
System altered.
```

5 以 SYS 身份删除表空间 UNDOTBS。结果如何？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DROP TABLESPACE undotbs INCLUDING CONTENTS AND DATAFILES;
DROP TABLESPACE undotbs INCLUDING CONTENTS AND DATAFILES
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

练习 10：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT segment_name
FROM   dba_rollback_segs
WHERE  tablespace_name = 'UNDOTBS';
SEGMENT_NAME
--------------------
_SYSSMU1$
_SYSSMU2$
_SYSSMU3$
_SYSSMU4$
_SYSSMU5$
_SYSSMU6$
_SYSSMU7$
_SYSSMU8$
8 rows selected.

SELECT a.usn,a.name,b.status
FROM v$rollname a, v$rollstat b
WHERE a.name
IN ( SELECT segment_name
FROM   dba_segments
WHERE  tablespace_name = 'UNDOTBS‘ )
AND a.usn = b.usn;

  USN NAME         STATUS
----- ------------ ---------------
    2 _SYSSMU2$    PENDING OFFLINE
```

练习 10：解答（续）

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。

```
ROLLBACK;
Rollback complete.
EXIT;
```

8 在以 SYS 的身份连接的会话中，删除表空间 UNDOTBS。结果如何？

```
DROP TABLESPACE undotbs;
DROP TABLESPACE undotbs
*
ERROR at line 1:
ORA-30013: undo tablespace 'UNDOTBS' is currently in use
```

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。

```
ALTER SYSTEM SET undo_retention=0 SCOPE=MEMORY;
System altered.

DROP TABLESPACE undotbs INCLUDING CONTENTS AND DATAFILES;
Tablespace dropped.
```

练习 11：解答

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。**注：**在使用 OEM 时，一定要将 DATE_OF_DELY 设置为 NULL。

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

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。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

CREATE TABLE customers
( cust_code VARCHAR2(3),
name VARCHAR2(50),
region VARCHAR2(5) )
TABLESPACE users;
Table created.

CREATE TABLE orders
( ord_id NUMBER(3),
ord_date DATE,
cust_code VARCHAR2(3),
date_of_dely DATE )
TABLESPACE users;
Table created.
```

练习 11：解答

1 （续）

使用 “OEM 控制台” (OEM Console)

- 通过 “方案” (Schema) > “表” (Table) > SYSTEM 创建表。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页中填写创建表 CUSTOMERS 所需的相应信息。
- 单击 “创建” (Create)。
- 重复上述步骤创建表 ORDERS。

2 运行脚本 lab11_02.sql，将行插入到表中。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab10_02.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

3 查找哪些文件和块包含订单表中的行。

提示：查询数据字典视图 DBA_EXTENTS。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT file_id, block_id, blocks
FROM   dba_extents
WHERE  owner = 'SYSTEM'
AND    segment_name = 'ORDERS'
AND    segment_type = 'TABLE';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         3         25          8
```

练习 11：解答（续）

4 检查 ORDERS 表使用的区数。

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT count(*)
FROM   dba_extents
WHERE  segment_name='ORDERS'
AND    owner='SYSTEM';

  COUNT(*)
----------
         1
```

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。

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE orders ALLOCATE EXTENT;
Table altered.

SELECT count(*)
FROM   dba_extents
WHERE  segment_name='ORDERS'
AND    owner='SYSTEM';

  COUNT(*)
----------
         2
```

练习 11：解答（续）

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。验证是否已按照指定的区数创建该表。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE TABLE orders2
TABLESPACE users
STORAGE(MINEXTENTS 10)
AS SELECT * FROM orders;
Table created.

SELECT count(*)
FROM   dba_extents
WHERE  segment_name='ORDERS2'
AND    owner='SYSTEM';
  COUNT(*)
----------
        10
```

使用 “OEM 控制台” (OEM Console)

- 通过 “方案” (Schema) > “表” (Table) > SYSTEM 创建表。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页中填写 “名称” (NAME)、“方案” (SCHEMA) 和 “表空间” (TABLESPACE)。
- 选择 “定义查询” (Define Query) 单选按钮，在框中写入查询 SELECT * FROM ORDERS;
- 在 “存储” (Storage) 页上将 “最小数目” (Minimum Number) 下的 “最小区数” (MINEXTENTS) 设置为 10。
- 单击 “创建” (Create)。

注：您必须以 SYSTEM 身份登录到 “OEM 控制台” (OEM Console)，否则，会收到错误消息。

练习 11：解答（续）

7 截断 ORDERS 表而不释放空间，检查区数以验证区未被回收。

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
TRUNCATE TABLE orders REUSE STORAGE;
Table truncated.

SELECT count(*)
FROM   dba_extents
WHERE  segment_name='ORDERS'
AND    owner='SYSTEM';
  COUNT(*)
----------
         2
```

8 截断 ORDERS2 表并释放空间。现在该表有多少个区？

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
TRUNCATE TABLE orders2;
Table truncated.

SELECT count(*)
FROM   dba_extents
WHERE  segment_name='ORDERS2'
AND    owner='SYSTEM';
  COUNT(*)
----------
        10
```

练习 11：解答（续）

9 运行脚本 lab11_09.sql，向 ORDERS2 表中插入一些行。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab11_09.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

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？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DESCRIBE orders2;
 Name                          Null?    Type
 ----------------------------- -------- ------------------
 ORD_ID                                 NUMBER(3)
 ORD_DATE                               DATE
 CUST_CODE                              VARCHAR2(3)
 DATE_OF_DELY                           DATE

ALTER TABLE orders2 SET UNUSED COLUMN date_of_dely
CASCADE CONSTRAINTS;
Table altered.

DESCRIBE orders2;
 Name                                       Null?    Type
 ------------------------- -------- ------------------
 ORD_ID                                 NUMBER(3)
 ORD_DATE                               DATE
 CUST_CODE                              VARCHAR2(3)
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “方案” (Schema) > “表” (Table) > SYSTEM。
- 在右侧的列中双击 ORDERS2 表。该操作将显示各列状况。
- 在 “常规” (General) 页上，选择 DATE_OF_DELY 列，然后单击位于 “常规” (General) 页左下侧的 “删除列” (Drop Column) 图标。
- 在 “操作” (Operations) 下的 “删除列” (Drop Columns) 对话框中，选择 “将所选列标记为未使用” (Mark selected columns as unused)。

练习 11：解答（续）

11 删除未使用的列 DATE_OF_DELY。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE orders2 DROP UNUSED COLUMNS;
Table altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “方案” (Schema) > “表” (Table) > SYSTEM > ORDERS2。
- 从 “常规” (General) 页选择 DATE_OF_DELY 列。
- 单击 “常规” (General) 页左下侧的 “删除列” (Drop Columns) 图标。
- 将出现 “删除列” (Drop Columns) 对话框。
- 选择删除未使用的列。

12 删除表 ORDERS2。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DROP TABLE orders2;
Table dropped.
```

使用 “OEM 控制台” (OEM Console)

- 导航到“方案” (Schema) > “表” (Table) > SYSTEM > ORDERS2。
- 单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。
- 选择 “是” (Yes) 确认删除表。

练习 12：解答

1 您将考虑为 CUSTOMERS 表的 NAME 列和 REGION 列创建索引。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 INDEX01 表空间中。

提示：B 树索引适合于重复值很少的列，而位图索引则适合于重复值很多的列。CUSTOMERS 表位于 SYSTEM 方案中。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

CREATE INDEX cust_name_idx
ON customers(name)
TABLESPACE index01;
Index created.

CREATE BITMAP INDEX cust_region_idx
ON system.customers(region)
TABLESPACE index01;
Index created.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “方案” (Schema) > INDEX。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页上填写创建索引所需的信息。
- 突出显示要对其创建索引的列。该操作将在 “订单” (Order) 列中放入一个 1。

注：要创建位图索引，应单击 “位图” (Bitmap) 单选按钮，否则，缺省情况下将创建 B*树索引。

练习 12：解答

2 将 CUST_REGION_IDX 索引移动到另一个表空间。

提示：可通过指定另外一个表空间来重建该索引。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER INDEX cust_region_idx REBUILD TABLESPACE indx;
Index altered.
```

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 可获得此信息。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT file_id, block_id, blocks
FROM   dba_extents
WHERE  segment_name='CUST_REGION_IDX'
AND    owner='SYSTEM';
   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4         17        125
```

练习 12：解答

4 重新创建 CUST_REGION_IDX 索引（而不用将其删除后再重新创建），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： 重建该索引。

注：新索引所使用的空间与该索引重建后从该区所在位置中看到的空间并不相同。这是因为 Oracle 服务器会建立一个临时索引并删除旧索引，然后再重命名该临时索引。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER INDEX cust_region_idx REBUILD;
Index altered.

SELECT file_id, block_id, blocks
  2  FROM   dba_extents
  3  WHERE  segment_name='CUST_REGION_IDX'
  4  AND    owner='SYSTEM';

   FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ----------
         4        142        125
```

5 a 以用户 SYSTEM 的身份，运行脚本 lab12_05a.sql 以创建和填充 NUMBERS 表。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab12_05a.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

练习 12：解答（续）

5 b 查询 NUMBERS 表以查找该表的两列中非重复值的数目。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT count(DISTINCT no) NO,
count(DISTINCT odd_even) OE
FROM   numbers;

        NO         OE
---------- ----------
     10000          2
```

c 使用统一的区大小 4KB，为 NUMBERS 表的 ODD_EVEN 列和 NO 列分别创建两个 B 树索引：NUMB_OE_IDX 和 NUMB_NO_IDX。将这些索引存放在 INDEX01 表空间中。检查总的索引大小，并将块数写入到下面的框中。

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	

练习 12：解答（续）

5 c（续）

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE INDEX numb_oe_idx
ON numbers(odd_even)
TABLESPACE index01;
Index created.

CREATE INDEX numb_no_idx
ON numbers(no)
TABLESPACE index01;
Index created.

COLUMN segment_name FORMAT a15
SELECT segment_name, blocks
FROM   dba_segments
WHERE  segment_name LIKE 'NUMB%'
AND    segment_type='INDEX';
SEGMENT_NAME        BLOCKS
--------------- ----------
NUMB_OE_IDX             40
NUMB_NO_IDX             46
```

使用 “OEM 控制台” (OEM Console)

要创建 B*树索引

- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页和 “存储” (Storage) 页上填写创建索引所需的信息。
- 突出显示要对其创建索引的列。该操作将在 “订单” (Order) 列中放入一个 1。
- 重复上述步骤创建下一个索引。

练习 12：解答（续）

5 d 记录上面的块之后，删除索引 NUMB_OE_IDX 和 NUMB_NO_IDX。使用统一的区大小 4KB，为 NUMBERS 表的 ODD_EVEN 和 NO 列分别创建位图索引：NUMB_OE_IDX 和 NUMB_NO_IDX。将这些索引存放在 INDEX01 表空间中。重新执行查询，从 DBA_SEGMENTS 检查给区分配的总块数。检查总的索引大小，并写入到下面的框中。

索引	列	块
NUMB_OE_IDX	ODD_EVEN	
NUMB_NO_IDX	NO	

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DROP INDEX numb_oe_idx;
Index dropped.

DROP INDEX numb_no_idx;
Index dropped.

CREATE BITMAP INDEX numb_oe_idx
ON numbers(odd_even) TABLESPACE index01;
Index created.

CREATE BITMAP INDEX numb_no_idx
ON numbers(no) TABLESPACE index01;
Index created.

SELECT segment_name, blocks
FROM   dba_segments
WHERE  segment_name LIKE 'NUMB%'
AND    segment_type='INDEX';
SEGMENT_NAME         BLOCKS
--------------- ----------
NUMB_OE_IDX               2
NUMB_NO_IDX              72
```

练习 12：解答（续）

5 d （续）

可以得出两类索引的基数和大小之间是什么关系吗？

答案：从结果中可以看出：位图索引是低基数列的紧凑形式，而 B 树索引则是高基数列的紧凑形式。

使用 “OEM 控制台” (OEM Console)

要删除索引

- 导航到 “方案” (Schema) > INDEX。
- 突出显示右侧列中的索引，单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)，以删除该索引。重复上述步骤删除下一个索引。
- “方案” (Schema) > INDEX

要创建位图索引

- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 在 “常规” (General) 页和 “存储” (Storage) 页上填写创建索引所需的信息。
- 突出显示要对其创建索引的列。该操作将在 “订单” (Order) 列中放入一个 1。
- 重复上述步骤创建下一个索引。

注： NUMBERS 表位于 SYSTEM 方案中。

练习 13：解答

1 检查脚本 `lab13_01.sql`。运行该脚本以创建约束。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 `lab13_01.sql` 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

2 查询数据字典以：

a 检查约束，检查它们是否可以被延迟以及它们的状态。
提示：使用 `DBA_CONSTRAINTS` 视图可获得此信息。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
COLUMN constraint_name FORMAT a25
COLUMN table_name      FORMAT a10
COLUMN constraint_type FORMAT a1
COLUMN deferrable      FORMAT a15
COLUMN status          FORMAT a10

SELECT constraint_name, table_name, constraint_type,
deferrable, status
FROM   dba_constraints
WHERE  table_name IN ('PRODUCTS','ORDERS','CUSTOMERS')
AND owner='SYSTEM';
CONSTRAINT_NAME           TABLE_NAME C DEFERRABLE      STATUS
------------------------- ---------- - --------------- --------
CUSTOMERS_REGION_CK       CUSTOMERS  C NOT DEFERRABLE  ENABLED
CUSTOMERS_CUST_CODE_PK    CUSTOMERS  P DEFERRABLE      ENABLED
ORDERS_CUST_CODE_FK       ORDERS     R DEFERRABLE      ENABLED
ORDERS_DATE_OF_DELY_CK    ORDERS     C NOT DEFERRABLE  ENABLED
ORDERS_ORD_ID_PK          ORDERS     P NOT DEFERRABLE  ENABLED
PRODUCTS_PROD_CODE_UK     PRODUCTS   U DEFERRABLE      DISABLED

6 rows selected.
```

练习 13：解答（续）

2 b 检查所创建的索引名称及类型，对约束进行验证。

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT index_name,table_name,uniqueness
FROM   dba_indexes
WHERE  index_name in
 ( SELECT constraint_name
 FROM   dba_constraints
 WHERE  table_name IN ('PRODUCTS', 'ORDERS', 'CUSTOMERS')
 AND    owner='SYSTEM'
 AND    constraint_type in ('P','U')  ) ;
INDEX_NAME                     TABLE_NAME UNIQUENES
------------------------------ ---------- ---------
CUSTOMERS_CUST_CODE_PK         CUSTOMERS  NONUNIQUE
ORDERS_ORD_ID_PK               ORDERS     UNIQUE
```

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。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab13_03.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

练习 13：解答（续）

4 在表 PRODUCT 上启用唯一性约束。是否已成功？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE system.products
ENABLE CONSTRAINT products_prod_code_uk;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE system.products
ENABLE NOVALIDATE CONSTRAINT products_prod_code_uk;
Table altered.
```

b 查询数据字典以验证更改结果。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT constraint_name, table_name, constraint_type,
validated, status FROM dba_constraints
WHERE table_name = 'PRODUCTS‘ AND owner='SYSTEM';
CONSTRAINT_NAME           TABLE_NAME C VALIDATED     STATUS
----------------------    ---------- - ------------- ------
PRODUCTS_PROD_CODE_UK     PRODUCTS   U NOT VALIDATED ENABLED
```

练习 13：解答（续）

5 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

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
INSERT INTO system.products VALUES(4000,'Monitor',3000);
INSERT INTO system.products
*
ERROR at line 1:
ORA-00001: unique constraint (SYSTEM.PRODUCTS_PROD_CODE_UK)
violated
```

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。

注：必须使用 SQL*Plus 或操作系统命令行运行该任务。

使用 SQL*Plus

```
SQL> CONNECT system/manager
Connected.
SQL> @?/rdbms/admin/utlexcpt
```

练习 13：解答（续）

6 b 运行该命令以启用约束并捕获异常情况。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE system.products
ENABLE CONSTRAINT products_prod_code_uk
EXCEPTIONS INTO system.exceptions;
ALTER TABLE system.products
*
ERROR at line 1:
ORA-02299: cannot validate (SYSTEM.PRODUCTS_PROD_CODE_UK) -
duplicate keys found
```

c 使用 EXCEPTIONS 表中的 ROWID 列出 PRODUCTS 表中违反约束的行。不要列出 LOB 列。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT rowid, prod_code, description FROM   system.products
WHERE rowid IN
 ( SELECT row_id
   FROM   exceptions
   WHERE  table_name='PRODUCTS‘ ) ;
ROWID              PROD_CODE DESCRIPTION
------------------ ---------- ------------------------------
AAABSgAAIAAAAASAAA       4000 UNIX Monitor
AAABSgAAIAAAAASAAB       4000 NT Monitor
```

练习 13：解答（续）

6 d 纠正错误。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
UPDATE system.products SET prod_code='4001'
WHERE rowid =
 ( SELECT max(row_id)
 FROM exceptions
 WHERE table_name='PRODUCTS‘ ) ;
1 row updated.
```

e 启用约束。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER TABLE system.products
ENABLE CONSTRAINT products_prod_code_uk
EXCEPTIONS INTO system.exceptions;
Table altered.
```

7 运行 lab13_07.sql 脚本将这些行插入到表中。插入是否成功？回退更改。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab13_07.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

练习 13：解答（续）

8 现在检查脚本 lab13_08。请注意：该脚本也按同样的顺序执行插入操作。运行该脚本并检查它是否执行成功。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab13_08.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

9 截断 CUSTOMERS 表。是否成功？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
TRUNCATE TABLE system.customers;
TRUNCATE TABLE system.customers
                     *
ERROR at line 1:
ORA-02266: unique/primary keys in table referenced by
enabled foreign keys
```

练习 14：解答

1 运行 lab14_01.sql 脚本来创建用户 Jeff。通过运行 @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql 脚本来启用口令管理。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab14_01.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

使用 SQL*Plus

```
SQL> @$ORACLE_HOME/rdbms/admin/utlpwdmg.sql
```

2 尝试将用户 Jeff 的口令更改为 Jeff。结果如何？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER USER jeff IDENTIFIED BY jeff;
ALTER USER jeff IDENTIFIED BY jeff
*
ERROR at line 1:
ORA-28003: password verification for the specified password
failed
ORA-20001: Password same as or similar to user
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Jeff。
- 在 “常规” (General) 页上使用 “输入口令” (Enter Password) 与 “确认口令” (Confirm Password)。
- 单击 “应用” (Apply)。

练习 14：解答

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER USER jeff IDENTIFIED BY super1$;
User altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Jeff。
- 在 “常规” (General) 页上使用 “输入口令” (Enter Password) 与 “确认口令” (Confirm Password)。
- 单击 “应用” (Apply)。

4 更改 DEFAULT 配置文件，确保以下情况适用于分配了 DEFAULT 配置文件的用户：

- 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。
- 确保已执行给定的要求。

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。

练习 14：解答（续）

4 （续）

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER PROFILE default LIMIT
 FAILED_LOGIN_ATTEMPTS 2
 PASSWORD_LIFE_TIME 30
 PASSWORD_REUSE_TIME 1/1440
 PASSWORD_GRACE_TIME 5;
Profile altered.

SELECT resource_name, limit FROM   dba_profiles
WHERE profile='DEFAULT‘ AND resource_type='PASSWORD';
RESOURCE_NAME                    LIMIT
-------------------------------- --------------------
FAILED_LOGIN_ATTEMPTS            2
PASSWORD_LIFE_TIME               30
PASSWORD_REUSE_TIME              .0006
PASSWORD_REUSE_MAX               UNLIMITED
PASSWORD_VERIFY_FUNCTION         VERIFY_FUNCTION
PASSWORD_LOCK_TIME               .0006
PASSWORD_GRACE_TIME              5

7 rows selected.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “配置文件” (Profiles) > “缺省” (Default)。
- 在 “口令” (Password) 页上，修改该配置文件。
- 单击 “应用” (Apply)。

练习 14：解答（续）

5 以用户 Jeff 的身份使用无效口令进行登录。进行两次这样的登录，然后再使用正确的口令进行登录。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT jeff/superman@[service name]
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.

CONNECT jeff/super@[service name]
ERROR:
ORA-01017: invalid username/password; logon denied

CONNECT jeff/super1$@[service name]
ERROR:
ORA-28000: the account is locked
```

练习 14：解答（续）

6 使用数据字典视图 DBA_USERS 验证是否已锁定用户 Jeff。如果已锁定，则为用户 Jeff 解除帐户锁定。为用户 Jeff 解除锁定之后，再以 Jeff 身份连接。

提示：执行 ALTER USER 命令可解除对帐户的锁定。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYS/ORACLE@[service name] AS SYSDBA
Connected.

SELECT username, account_status FROM dba_users;
USERNAME        ACCOUNT_STATUS
--------------- ---------------
SYS             OPEN
SYSTEM          OPEN
OUTLN           OPEN
ORDSYS          OPEN
MDSYS           OPEN
OE              OPEN
HR              OPEN
ORDPLUGINS      OPEN
DBSNMP          OPEN
JEFF            LOCKED(TIMED)
10 rows selected.

ALTER USER jeff ACCOUNT UNLOCK;
User altered.

CONNECT jeff/super1$
Connected.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Jeff。
- 可以从 “常规” (General) 页检查用户帐户的状态。
- 要解除锁定，应选择 “解除锁定” (Unlocked) 单选按钮，再单击 “应用” (Apply)。

练习 14：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER PROFILE default LIMIT
 FAILED_LOGIN_ATTEMPTS     UNLIMITED
 PASSWORD_LIFE_TIME        UNLIMITED
 PASSWORD_REUSE_TIME       UNLIMITED
 PASSWORD_REUSE_MAX        UNLIMITED
 PASSWORD_VERIFY_FUNCTION  NULL
 PASSWORD_LOCK_TIME        UNLIMITED
 PASSWORD_GRACE_TIME       UNLIMITED;

Profile altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “配置文件” (Profiles) > “缺省” (Default)。
- 在 “口令” (Password) 页上，修改该配置文件。
- 单击 “应用” (Apply)。

练习 14：解答（续）

8 以用户 Jeff 的身份使用无效口令进行登录。进行两次这样的登录，然后再使用正确的口令进行登录。结果如何？为什么？

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT jeff/superman@[service name]
ERROR:
ORA-01017: invalid username/password; logon denied
Warning: You are no longer connected to ORACLE.

CONNECT jeff/super@[service name]
ERROR:
ORA-01017: invalid username/password; logon denied

CONNECT jeff/super1$@[service name]
Connected.
```

答案：当使用正确口令进行第三次登录时（象第 5 步中那样），帐户没有锁定。这是因为在第 7 步中禁用了 PASSWORD_VERIFY_FUNCTION（设置为 NULL）。

练习 15：解答

1 创建用户 Bob，口令为 CRUSADER。确保 Bob 所创建的任何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 能够登录，并可以在 USERS 和 INDX 这两个表空间中创建最大大小可达 1 兆字节的对象。使用 lab15_01.sql 脚本授予 Bob 创建会话的权限。

提示：请为 Bob 分配缺省表空间 USERS 和临时表空间 TEMP。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

CREATE USER bob IDENTIFIED BY crusader
DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE temp
QUOTA 1M ON USERS QUOTA 1M ON INDX;
User created.
```

使用 SQL*Plus Worksheet

要打开 lab 脚本：

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab15_01.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。将出现 “创建用户” (Create User) 对话框。
- 使用 “常规” (General) 页标识用户并分配缺省与临时表空间。
- 使用 “限额” (Quota) 页定义 USERS 和 INDX 表空间的限额。
- 单击 “创建” (Create)。

练习 15：解答

2 创建用户 Emi，口令为 Mary。确保由 Emi 创建的任何对象和排序段都不是在系统表空间中创建的。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE USER emi IDENTIFIED BY mary
DEFAULT TABLESPACE users TEMPORARY TABLESPACE temp;
User created.
```

使用 “OEM 控制台” (OEM Console)

- 导航到“安全性” (Security) > “用户” (Users)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。将出现 “创建用户” (Create User) 对话框。
- 使用 “常规” (General) 页标识用户并分配缺省与临时表空间。
- 单击 “创建” (Create)。

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT username, default_tablespace, temporary_tablespace
FROM   dba_users WHERE  username IN ('BOB', 'EMI');
USERNAME   DEFAULT_TABLESPACE   TEMPORARY_TABLE
---------- -------------------- ---------------
EMI        USERS                TEMP
BOB        USERS                TEMP
```

练习 15：解答（续）

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。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
COLUMN tablespace_name FORMAT a15
COLUMN user FORMAT a10
SELECT * FROM dba_ts_quotas WHERE username = 'BOB';
TABLESPACE_NAME USERNAME BYTES  MAX_BYTES BLOCKS MAX_BLOCKS
--------------- -------- ----- ---------- ------ ----------
INDX            BOB          0    1048576      0        256
USERS           BOB          0    1048576      0        256
```

5 a 以用户 Bob 的身份更改其临时表空间。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
connect bob/crusade@[service name]
Connected.

ALTER USER bob TEMPORARY TABLESPACE users;
ALTER USER bob
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

使用 “OEM 控制台” (OEM Console)

尽管 “OEM 控制台” (OEM Console) 可用来更改 Bob 的临时表空间，但就此步骤来说，应使用 SQL*Plus Worksheet，以接收在以 Bob 身份连接时收到的错误消息。

练习 15：解答（续）

5 b 以用户 Bob 的身份将其口令更改为 Sam。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT bob/crusader@[service name]
Connected.

ALTER USER bob IDENTIFIED BY sam;
User altered.
```

使用 “OEM 控制台” (OEM Console)

尽管 “OEM 控制台” (OEM Console) 可用来更改 Bob 的口令，但就此步骤来说，应使用 SQL*Plus Worksheet，以验证因 Bob 拥有更改其口令的权限而不会收到错误消息。

6 以用户 SYSTEM 的身份删除 Bob 的缺省表空间限额。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

ALTER USER bob QUOTA 0 ON users;
User altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 使用 “限额” (Quota) 页删除对 Bob 的缺省表空间设置的限额。
- 单击 “应用” (Apply)。

练习 15：解答（续）

7 从数据库中删除 Emi 的帐户。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
DROP USER emi;
User dropped.
```

使用 “OEM 控制台” (OEM Console)

- 导航到“安全性” (Security) > “用户” (Users) > Emi。
- 单击鼠标右键，从弹出的菜单中选择 “删除” (Remove)。
- 选择 “是” (Yes) 确认删除用户 Emi。

8 Bob 忘了他的口令。给他分配一个口令 OLINK 并要求 Bob 在下次登录时更改自己的口令。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER USER bob IDENTIFIED BY olink PASSWORD EXPIRE;
User altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 使用 “常规” (General) 页更改口令。
- 选择 “立即使口令失效” (Expire Password Now)。
- 单击 “应用” (Apply)。

练习 16：解答

1 以用户 SYSTEM 的身份创建用户 Emi，并授予她登录到数据库以及在她的方案中创建对象的权限。为她分配缺省表空间 DATA01 和临时表空间 TEMP。将她在 DATA01 上的限额设为 1M。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

CREATE USER emi IDENTIFIED BY "abcd12"
DEFAULT TABLESPACE data01 TEMPORARY TABLESPACE temp
QUOTA 1M ON data01;
User created.

GRANT create session, create table TO emi;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security)> “用户” (Users)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。将出现 “创建用户” (Create User) 对话框。
- 使用 “常规” (General) 页标识用户并分配缺省与临时表空间。
- 使用 “限额” (Quota) 页将 DATA01 的限额设为 1M。
- 单击 “创建” (Create)。

2 a 运行脚本 lab16_02a.sql，以用户 Emi 身份连接并创建表 CUSTOMERS1 和 ORDERS1。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (File) > “打开” (Open)。
- 选择 lab16_02a.sql 脚本。
- 选择 “打开” (Open)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (Execute)。

练习 16：解答（续）

2 b 以 SYSTEM 身份进行连接，并将 SYSTEM.CUSTOMERS 中的数据复制到 Emi 的 CUSTOMERS1 表中。确认已插入这些记录。

使用 **SQL*Plus Worksheet**

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

INSERT INTO emi.customers1 SELECT * FROM system.customers;
9 rows created.

SELECT * FROM emi.customers1;
CUS_CODE NAME                                          REGION
-------- ----------------------------------------    -----
A01    TKB SPORT SHOP                                  West
A02    VOLLYRITE                                       North
A03    JUST TENNIS                                     North
A04    EVERY MOUNTAIN                                  South
A05    SHAPE UP                                        South
A06    SHAPE UP                                        West
A07    WOMENS SPORTS                                   South
A08    NORTH WOODS HEALTH                              East
J01    Sports Store                                    East

9 rows selected.
```

练习 16：解答（续）

2 c 以用户 SYSTEM 的身份授予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的权限。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

GRANT select ON emi.customers1 TO bob;
GRANT select ON emi.customers1 TO bob
                     *
ERROR at line 1:
ORA-01031: insufficient privileges
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 选择 “对象权限” (Object Privileges)页。
- 导航到 Emi > “表”(Tables) > CUSTOMERS1。
- 通过从 “对象权限” (Object Privileges) 页的 “可用权限” (Available Privileges) 窗口双击 SELECT，可向 Bob 授予 SELECT 权限。该操作将该权限放入 “已授予” (Granted) 窗口。
- 选择 “应用” (Apply)。

注：您将收到错误消息，指明权限不足。

练习 16：解答（续）

3 以 Emi 的身份重新进行连接，并授予 Bob 从 Emi 的 CUSTOMERS1 表中进行选择的权限。此外，使 Bob 能向其他用户赋予选择权限。以用户 SYSTEM 的身份检查记录这些操作的数据字典视图。

提示：使用 DBA_TAB_PRIVS 进行检查。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT emi/abcd12@[service name]
Connected.

GRANT select ON customers1 TO bob WITH GRANT OPTION;
Grant succeeded.

CONNECT system/manager@[service name]
Connected.

COLUMN grantee    FORMAT a8
COLUMN owner      FORMAT a8
COLUMN table_name FORMAT a10
COLUMN grantor    FORMAT a8
COLUMN privilege  FORMAT a10
COLUMN grantable  FORMAT a3
COLUMN hierarchy  FORMAT a3
SELECT * FROM dba_tab_privs WHERE grantee='BOB';
GRANTEE OWNER TABLE_NAME GRANTOR  PRIVILEGE  GRANT HIERARCHY
------- ----- ---------- -------- ---------- ----- ---------
BOB      EMI  CUSTOMERS1 EMI      SELECT     YES   NO
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 选择 “对象权限” (Object Privileges) 页。
- 导航到 Emi > “表”(Tables) > CUSTOMERS1。
- 通过从“对象权限” (Object Privileges) 页的 “可用权限” (AvailablePrivileges) 窗口双击 SELECT，可向 Bob 授予 SELECT 权限。该操作将该权限放入 “已授予” (Granted) 窗口。
- 在 “已授予” (Granted) 窗口单击 “授予选项” (Grant Option)。
- 选择 “应用” (Apply)。

练习 16：解答（续）

4 创建以 diamond1$ 标识的用户 Trevor，使其具有登录到数据库的能力。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager
Connected.

CREATE USER trevor IDENTIFIED BY “diamond1$";
User created.

GRANT create session TO trevor;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

1. 导航到 “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)。将出现 “创建用户” (Create User) 对话框。
3. 使用 “常规” (General) 页创建用户。
4. 使用 “系统权限” (System Privileges) 页选择 CREATE SESSION 权限。
5. 单击 “创建” (Create)。

练习 16：解答（续）

5 a 以 Bob 的身份使 Trevor 能够访问 Emi 的 CUSTOMERS1 表。

注：由于第 15 课第 8 步中的操作，将收到 “口令已到期” 消息。为 Bob 指定新口令 aaron$1。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT bob/olink@[service name]
ERROR:
ORA-28001: the password has expired
Changing password for bob

New password: aaron1$
Retype new password: aaron1$
Password changed
Connected.

GRANT select ON emi.customers1 TO trevor;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Trevor。
- 选择 “对象权限” (Object Privileges) 页。
- 导航到 Emi > “表” (Tables) > CUSTOMERS1。
- 通过从 “对象权限” (Object Privileges) 页的 “可用权限” (Available Privileges) 窗口双击 SELECT，可向 Trevor 授予 SELECT 权限。该操作将该权限放入 “已授予” (Granted) 窗口。
- 选择 “应用” (Apply)。

练习 16：解答（续）

5 b 以 Emi 的身份撤消 Bob 读取 Emi 的 CUSTOMERS1 表的权限。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT emi/abcd12@[service name]
Connected.

REVOKE select ON customers1 FROM bob;
Revoke succeeded.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 选择 “对象权限” (Object Privileges)页。
- 导航到 Emi > “表” (Tables) > CUSTOMERS1。
- 突出显示 “已授予” (Granted) 窗口中的 SELECT 权限。
- 单击位于 “已授予” 窗口上方的向上箭头。该操作将撤消该权限。
- 选择 “应用” (Apply)。

c 以 Trevor 的身份查询 Emi 的 CUSTOMERS1 表。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT trevor/diamond1$@[service name]
Connected.

SELECT * FROM emi.customers1;
FROM emi.customers1
         *
ERROR at line 2:
ORA-00942: table or view does not exist
```

练习 16：解答（续）

6 赋予 Emi 启动和关闭数据库的能力，但不赋予创建新数据库的能力。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT SYS/oracle@[service name] AS SYSDBA
Connected.
GRANT sysoper TO emi;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Emi。
- 选择 “系统权限” (System Privileges) 页。
- 从 “可用” (Available) 窗口双击 SYSOPER 权限。该操作将该权限放入 “已授予” (Granted) 窗口。
- 选择 “应用” (Apply)。

练习 17：解答

1 检查数据字典视图，并列出 RESOURCE 角色的系统权限。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

COLUMN privilege FORMAT a20
COLUMN grantee   FORMAT a10
SELECT * FROM dba_sys_privs WHERE grantee = 'RESOURCE';
GRANTEE    PRIVILEGE            ADM
---------- -------------------- ---
RESOURCE   CREATE CLUSTER       NO
RESOURCE   CREATE INDEXTYPE     NO
RESOURCE   CREATE OPERATOR      NO
RESOURCE   CREATE PROCEDURE     NO
RESOURCE   CREATE SEQUENCE      NO
RESOURCE   CREATE TABLE         NO
RESOURCE   CREATE TRIGGER       NO
RESOURCE   CREATE TYPE          NO
8 rows selected.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “角色” (Roles) > Resource。
- 使用 “系统权限” (System Privileges) 页查看分配给 Resource 角色的系统权限列表。

练习 17：解答

2 创建名为 DEV 的角色，该角色允许被授予该角色的用户能够创建表、视图并能够从 Emi 的 CUSTOMERS1 表进行选择。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CREATE ROLE dev;
Role created.

GRANT create table, create view TO dev;
Grant succeeded.

CONNECT emi/abcd12@[service name]
Connected.

GRANT select ON customers1 TO dev;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

创建 DEV 角色：

- 导航到 “安全性” (Security) > “角色” (Roles)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。
- 使用 “常规” (General) 页标识角色。
- 使用 “系统权限” (System Privileges) 页授予 CREATE TABLE 和 CREATE VIEW 权限。
- 选择 “创建” (Create)。

向 DEV 授予对于 CUSTOMERS1 的 Select 权限：

- 选择 “对象权限” (Object Privileges) 页。
- 导航到 Emi > “表” (Tables) > CUSTOMERS1。
- 突出显示 “已授予” (Granted) 窗口中的 SELECT 权限。
- 选择 “应用” (Apply)。

注：使用 “OEM 控制台” (OEM Console) 时，不要以 Emi 身份登录。

练习 17：解答（续）

3 a 将 RESOURCE 角色和 DEV 角色分配给 Bob，但只允许 RESOURCE 角色在 Bob 登录时自动启用。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

GRANT dev, resource TO bob;
Grant succeeded.

ALTER USER bob DEFAULT ROLE resource;
User altered.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 使用 “角色” (Role) 页分配 RESOURCE 和 DEV 角色。
- 在 “已授予” 窗口为 RESOURCE 角色选择 Default 权限。执行该操作后会选中 Default 框。
- 单击 “应用”(Apply)。

b 为 Bob 授予读取所有数据字典信息的权限。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

GRANT select_catalog_role TO bob;
Grant succeeded.
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “安全性” (Security) > “用户” (Users) > Bob。
- 使用 “角色”(Role)页授予 SELECT_CATALOG_ROLE 角色。
- 单击 “应用”(Apply)。

练习 17：解答（续）

4 Bob 需要检查例程当前使用的还原段。
以 Bob 的身份进行连接，并列出所使用的还原段。
提示：使用 SET ROLE SELECT_CATALOG_ROLE

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT bob/aaron1$@[service name]
Connected.

SET ROLE select_catalog_role;
Role set.

SELECT segment_name FROM   dba_rollback_segs
WHERE  status='ONLINE';
SEGMENT_NAME
---------------
SYSTEM
_SYSSMU9$
_SYSSMU10$
_SYSSMU11$
_SYSSMU12$
_SYSSMU13$
_SYSSMU14$
_SYSSMU15$
_SYSSMU16$
9 rows selected.
```

练习 17：解答（续）

5 以 SYSTEM 用户身份创建一个有关 Emi 的 CUSTOMERS 表的视图 CUST_VIEW。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT system/manager@[service name]
Connected.

CREATE VIEW cust_view AS SELECT * FROM emi.customers;
   FROM   emi.customers
               *
ERROR at line 3:
ORA-01031: insufficient privileges
```

使用 “OEM 控制台” (OEM Console)

- 导航到 “方案” (Schema) > “视图” (View)。
- 单击鼠标右键，从弹出的菜单中选择 “创建” (Create)。将出现 “创建视图” (Create View) 对话框。
- 使用 “常规” (General) 页标识视图名称、方案 SYSTEM 和 AS SELECT 查询文本。
- 选择 “创建” (Create)。

注：由于以 SYSTEM 身份连接时您没有执行该操作的足够权限，您会收到错误代码为 ORA-01031 的消息。

练习 17：解答（续）

6 以用户 Emi 的身份向用户 SYSTEM 授予对于 CUSTOMERS1 的 select 权限。以用户 SYSTEM 的身份对 Emi 的 CUSTOMERS1 表创建视图 CUST_VIEW。结果如何？

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT emi/abcd12@[service name]
Connected.

GRANT select ON customers TO system;
Grant succeeded.

CONNECT system/manager@[service name]
Connected.

CREATE VIEW cust_view AS SELECT * FROM emi.customers;
View created.
```

练习 19：OEM 解答

1 a 查看加载时所要使用的数据文件，以便熟悉控制文件和数据文件的格式。

- 如果使用 SQL*Plus，请查看以下文件：

  `lcase1.ctl`、`lcase2.ctl` 和 `lcase2.dat`

- 如果使用 Oracle Enterprise Manager，请查看以下文件：

  `OEMsqlcase1.ctl`、`OEMsqlcase2.ctl` 和 `OEMsqlcase2`

b 以用户 `HR` 的身份运行 `lab19_01.sql` 脚本，创建 `DEPARTMENTS2` 表。

使用 SQL*Plus Worksheet

- 从主菜单选择 “文件” (`File`) > “打开” (`Open`)。
- 选择 `lab19_01.sql` 脚本。
- 选择 “打开” (`Open`)。该操作会使该脚本的文本显示在 SQL*Plus Worksheet 窗口中。
- 更新第一行，在连接字符串中包含服务名。
- 选择 “执行” (`Execute`)。

练习 19：OEM 解答

2 a 使用常规路径方法运行 SQL*Loader，以将数据加载到 DEPARTMENTS2 表中。请使用以下控制文件：

- lcase1.ctl（如果使用 SQL*Plus）
- OEMsqlcase1.ctl（如果使用 Oracle Enterprise Manager）

文件位于 LABS 目录中。

注：此次运行使用了包含有输入数据的控制文件。

使用 “OEM 控制台” (OEM Console)

- 在数据库树中向下导航到 “方案” (Schema) > “表” (Table) > HR > DEPARTMENTS2。
- 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
- 从 “加载向导简介” (Load Wizard Introduction) 页选择 “下一步” (Next)。
- 在 “控制文件” (Control File) 页中，指定 OEMsqlcase1.ctl 文件的完整路径和名称，然后选择 “下一步” (Next)。

  （LABS 目录由教师提供）

  注：输入路径和名称时需区分大小写。
- 在 “数据文件” (Data File) 页中，选择 “在控制文件中指定数据文件” (The data file is specified in the control file) 单选按钮。
- 在 “加载方法” (Load Method)页中，选择 “常规路径” (Conventional Path) 单选按钮，然后选择 “下一步” (Next)。
- 在 “调度” (Schedule) 页中，选择 “立即” (Immediately) 单选按钮立即运行该作业，然后选择 “下一步” (Next)。
- 在 “作业信息” (Job Information) 页中指定作业的名称：LoadOEMsqlcase1，然后选择 “现在提交作业” (Submit the job now) 单选按钮，接着选中下面的框：“覆盖首选身份证明：用户名：HR 口令：HR 连接身份：SYSDBA” (Override Preferred Credentials: Username: HR Password: HR Connect As: SYSDBA)，最后选择 “完成” (Finish)。
- 随即显示概要页，列出有关加载的所有详细资料。检查这些信息，确保无误后选择 “确定” (OK) 继续操作。注：系统将创建一个日志文件列出加载涉及到的各个操作。

  注：系统将创建一个日志文件列出加载过程中出现的所有问题。
- 会有一个对话框指明已成功提交了作业，选择 “确定” (OK) 继续操作。
- 在导航树中找到 “作业” (Jobs)。在右窗口中选择 “历史记录” (History) 选项卡。随即显示一个作业列表。检查 LOAD OEMsqlcase1 的 “状态” (Status) 字段。它将显示 “已完成” (Completed)，表明作业已成功完成。

练习 19：OEM 解答

2 b 使用操作系统命令查看日志文件。

使用操作系统命令行

```
$ more lcase1.log
   -- Note:  Path used:  Conventional
```

c 查询 DEPARTMENTS2 表以检查数据。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT * FROM DEPTARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- --------------------
         10 Administration
         20 Marketing
         30 Purchasing
         40 Human Resources
         50 Shipping
         60 IT
         70 Public Relations
         …
         27 rows selected
```

3 删除 DEPTARTMENTS2 表中的所有记录。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
TRUNCATE TABLE departments2;
Table truncated.
```

练习 19：OEM 解答

4 a 采用直接路径模式运行 SQL*Loader，以将数据加载到 DEPTARTMENTS2 表中。请使用以下控制文件：

- lcase2.ctl（如果使用 SQL*Plus）
- OEMsqlcase2.ctl（如果使用 Oracle Enterprise Manager）

文件位于 LABS 目录中。

注：此次运行通过使用输入数据文件来加载数据。

使用 “OEM 控制台” (OEM Console)

- 在数据库树中向下导航到 “方案” (Schema) > “表” (Table) > HR > DEPARTMENTS2。
- 单击鼠标右键，从弹出的菜单中选择 “数据管理” (Data Management) > “加载” (Load)。
- 从 “加载向导简介” (Load Wizard Introduction) 页选择 “下一步” (Next)。
- 在 “控制文件” (Control File) 页中指定 OEMsqlcase2.ctl 文件的完整路径和名称（LABS 目录由教师提供），然后选择 “下一步” (Next)。
  注：输入路径和名称时需区分大小写。
- 在 “数据文件” (Data File) 页中，选择 “提供在数据库服务器上的完整路径和名称” (Provide the full path and name on the database server machine) 单选按钮。
- 输入 OEMsqlcase2.dat 数据文件所在的完整路径，然后选择 “下一步” (Next)。
  注：输入路径和名称时需区分大小写。
- 在 “加载方法” (Load Method) 页中，选择 “直接路径” (Direct Path) 单选按钮，然后选择 “下一步” (Next)。
- 在 “调度” (Schedule) 页中，选择 “立即” (Immediately) 单选按钮立即运行该作业，然后选择 “下一步” (Next)。
- 在 “作业信息” (Job Information) 页中指定作业的名称：LoadOEMsqlcase2，然后选择 “现在提交作业” (Submit the job now) 单选按钮，接着选中下面的框：“覆盖首选身份证明：用户名：HR 口令：HR 连接身份：SYSDBA” (Override Preferred Credentials: Username: HR Password: HR Connect As: SYSDBA)，最后选择 “完成” (Finish)。
- 随即显示概要页，列出有关加载的所有详细资料。检查这些信息，确保无误后选择 “确定” (OK) 继续操作。注：系统将创建一个日志文件列出加载涉及到的各个操作。
- 会有一个对话框指明已成功提交了作业，选择 “确定” (OK) 继续操作。
- 在导航树中找到 “作业” (Jobs)。在右窗口中选择 “历史记录” (History) 选项卡。随即显示一个作业列表。检查 Load OEMsqlcase2 的 “状态” (Status) 字段。它将显示 “已完成” (Completed)，表明作业已成功完成。

练习 19：解答

4b 使用操作系统命令查看日志文件。

使用操作系统命令行

```
$ more lcase2.log
   -- Note:  Path used:  Direct
```

c 查询 DEPARTMENTS2 表以检查数据。

使用 **SQL*Plus Worksheet**

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT * FROM DEPARTMENTS2;
   DEPT_ID DEPT_NAME
   ------- ------------------
         10 Administration
         20 Marketing
         30 Purchasing
         40 Human Resources
         50 Shipping
         60 IT
         70 Public Relations
         80 Sales
         90 Executive
         …
         27 rows selected.
```

练习 19：解答

5 a 以用户 HR 的身份通过从 EMPLOYEES 表中选择来创建 EMPLOYEES2 表。截断 EMPLOYEES2 表，然后从该表中选择，以验证表中没有数据。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT hr/hr@[service name]

CREATE TABLE employees2 AS SELECT * FROM employees;
Table created.

TRUNCATE TABLE employees2;
Table truncated.

SELECT * FROM employees2;
no rows selected
```

b 执行从 EMPLOYEES 表到 EMPLOYEES2 表的直接加载插入，并查询 EMPLOYEES2 以验证加载结果。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
INSERT /*+ append */ INTO employees2
NOLOGGING  SELECT * from employees;
107 rows created.

COMMIT;

SELECT employee_id, first_name, last_name FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ----------------- -------------------------
139   John                   Seo
140   Joshua              Patel
141   Trenna              Rajs
142   Curtis              Davies
…
107 rows selected.
```

练习 19：解答

6 a 再次截断 EMPLOYEES2 表，然后从 EMPLOYEES2 表中选择，以验证该表中没有数据。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
TRUHCATE TABLE employees2;
Table truncated.

SELECT * FROM employees2;
no rows selected
```

b 从 EMPLOYEES 表执行一次并行直接加载插入，以恢复数据。将并行度指定为 2。检验数据。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER SESSION ENABLE PARALLEL DML;
Session altered

INSERT /*+ parallel(employees2,2) */ INTO employees2 NOLOGGING
SELECT * FROM employees;
107 rows created.

COMMIT;
Commit complete.

SELECT employee_id, first_name, last_name FROM employees2;
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- ------------------ -------------------------
139   John                     Seo
140   Joshua                Patel
141   Trenna                Rajs
142   Curtis                Davies
…
107 rows selected.
```

练习 20：解答

1 检查数据库和国家字符集。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
CONNECT sys/oracle@[service name] AS SYSDBA
Connected.

SELECT parameter, value FROM nls_database_parameters
WHERE parameter LIKE '%CHARACTERSET%';
PARAMETER                      VALUE
-----------------------        ------------
NLS_CHARACTERSET               WE8ISO8859P1
NLS_NCHAR_CHARACTERSET         AL16UTF16
```

2 哪些是数据库字符集的有效值。

使用 SQL*Plus Worksheet

- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
SELECT value FROM v$nls_valid_values
WHERE parameter = 'CHARACTERSET‘ ORDER BY value;
VALUE
--------------------
AL16UTF16
AL24UTFFSS
AL32UTF8
AR8ADOS710
AR8ADOS710T
AR8ADOS720
AR8ADOS720T
.
.
.
```

练习 20：解答（续）

3 确保该会话的所有日期中的年份均以 4 位数表示。将 NLS_LANGUAGE 更改为 FRENCH。从 DUAL 中选择 sysdate。

使用 SQL*Plus Worksheet

- 登录时，必须在连接字符串中包含服务名。
- 输入 SQL 命令。
- 选择 “执行” (Execute)。

```
ALTER SESSION SET nls_date_format = 'DD-MON-YYYY';
Session altered.

ALTER SESSION SET nls_language = FRENCH;
Session modifiée.

SELECT sysdate FROM dual;
SYSDATE
-----------
07-OCT-2001
```