

Oracle Database 11g: SQL 基础 II

学生指南第 2 册

D49994CN20

版本 2.0

2010 年 5 月

D66917

ORACLE®

作者

Chaitanya Koratamaddi
Brian Pottle
Tulika Srivastava

技术撰稿人和审稿人

Claire Bennett
Ken Cooper
Yanti Chang
Laszlo Czinkoczki
Burt Demchick
Gerdine Frenzen
Joel Goodman
Laura Garza
Richard Green
Nancy Greenberg
Akira Kinutani
Wendy Lo
Isabelle Marchand
Timothy Mcglue
Alan Paulson
Srinivas Putrevu
Bryan Roberts
Clinton Shaffer
Abhishek Singh
Jenny Tsai Smith
James Spiller
Lori Tritz
Lex van der Werff
Marcie Young

编辑

Amitha Narayan
Daniel Milne

制图员

Satish Bettegowda

出版商

Veena Narasimhan

版权所有 © 2010, Oracle。保留所有权利。

免责声明

本课程概要说明了在版本 11g 中计划提供的一些特性和增强功能。仅旨在帮助您评估升级到 11g 后对业务带来的好处，以及计划您的 IT 项目。

本课程任何形式的内容（包括课程练习及印刷材料）均属于 Oracle 拥有专利权的专有信息。未经 Oracle 预先书面许可，不得公开、复制和再版本课程及所包含的信息，也不能将其散布到 Oracle 以外的任何人员。本课程及其内容不属于许可证协议的一部分，也不能将其包括在与 Oracle 或其下属公司或子公司签署的任何合同中。

本课程仅用于参考目的，仅旨在帮助您计划实施和升级所描述的产品功能。本课程不承诺提供任何材料、代码或功能，不应将其作为制定采购决策时的依据。本文档中描述的所有特性或功能的开发、发行版本以及时间安排完全由 Oracle 自行决定。

本文档包含专有权信息，并受版权法和其它知识产权法的保护。您可以复制和打印本文档，但只能在 Oracle 培训课程中使用。不得以任何方式修改或变更本文档。除了在依照版权法中制定的“合理使用”范围内使用本文档外，在未经 Oracle 明确授权的情况下，您不得以全部或部分的形式使用、共享、下载、上载、复制、打印、显示、展示、再版、发布、许可、张贴、传播或散布本文档。

本文档中包含的信息如有更改，恕不另行通知。如果您在本文档中发现任何问题，请书面通知：Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA。Oracle 不保证本文档中没有错误。

有限权利声明

如果将本文档交付给美国政府或代表美国政府使用本文档的任何人，则适用以下通知中的规定：

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

商标声明

Oracle 是 Oracle 公司和（或）其分公司的注册商标。其它名称可能是其各自拥有者的商标。

目录

I 简介

课程目标	I-2
课程安排	I-3
课程目标	I-4
前导课程	I-5
课程安排	I-6
本课程中使用的表	I-8
本课程中使用的附录	I-9
开发环境	I-10
课程安排	I-11
复习对数据进行限制	I-12
复习对数据进行排序	I-13
复习 SQL 函数	I-14
复习单行函数	I-15
复习各类组函数	I-16
复习使用子查询	I-17
复习处理数据	I-19
课程安排	I-20
Oracle Database 11g 的 SQL 文档	I-21
其它资源	I-22
小结	I-23
练习 I: 概览	I-24

1 控制用户访问

课程目标	1-2
课程安排	1-3
控制用户访问	1-4
权限	1-5
系统权限	1-6
创建用户	1-7
用户系统权限	1-8

授予系统权限 1-10
课程安排 1-11
角色是什么 1-12
创建角色和为角色授予权限 1-13
更改口令 1-14
课程安排 1-15
对象权限 1-16
授予对象权限 1-18
传递权限 1-19
确认授予的权限 1-20
课程安排 1-21
撤消对象权限 1-22
小测验 1-24
小结 1-25
练习 1：概览 1-26

2 管理方案对象

课程目标 2-2
课程安排 2-3
ALTER TABLE 语句 2-4
添加列 2-6
修改列 2-7
删除列 2-8
SET UNUSED 选项 2-9
课程安排 2-11
添加约束条件语法 2-12
添加约束条件 2-13
ON DELETE 子句 2-14
延迟约束条件 2-15
INITIALLY DEFERRED 与 INITIALLY IMMEDIATE 之间的区别 2-16
删除约束条件 2-19
禁用约束条件 2-20
启用约束条件 2-21
级联约束条件 2-23
重命名表列和约束条件 2-25
课程安排 2-26

索引概览 2-27
CREATE INDEX 与 CREATE TABLE 语句配合使用 2-28
基于函数的索引 2-30
删除索引 2-31
DROP TABLE … PURGE 2-32
课程安排 2-33
FLASHBACK TABLE 语句 2-34
使用 FLASHBACK TABLE 语句 2-36
课程安排 2-37
临时表 2-38
创建临时表 2-39
课程安排 2-40
外部表 2-41
创建外部表目录 2-42
创建外部表 2-44
使用 ORACLE_LOADER 创建外部表 2-46
查询外部表 2-48
使用 ORACLE_DATAPUMP 创建外部表：示例 2-49
小测验 2-50
小结 2-52
练习 2：概览 2-53

3 使用数据字典视图管理对象

课程目标 3-2
课程安排 3-3
数据字典 3-4
数据字典结构 3-5
如何使用字典视图 3-7
USER_OBJECTS 和 ALL_OBJECTS 视图 3-8
USER_OBJECTS 视图 3-9
课程安排 3-10
表信息 3-11
列信息 3-12
约束条件信息 3-14
USER_CONSTRAINTS：示例 3-15
在 USER_CONS_COLUMNS 中进行查询 3-17

课程安排 3-18
视图信息 3-19
序列信息 3-20
确认序列 3-21
索引信息 3-22
USER_INDEXES: 示例 3-23
在 USER_IND_COLUMNS 中进行查询 3-24
同义词信息 3-25
课程安排 3-26
在表中添加注释 3-27
小测验 3-28
小结 3-29
练习 3: 概览 3-30

4 处理大型数据集

课程目标 4-2
课程安排 4-3
使用子查询处理数据 4-4
通过将子查询用作源来检索数据 4-5
通过将子查询用作目标来执行插入 4-7
在 DML 语句中使用 WITH CHECK OPTION 关键字 4-9
课程安排 4-11
显式默认值功能概览 4-12
使用显式默认值 4-13
从其它表中复制行 4-14
课程安排 4-15
多表 INSERT 语句概览 4-16
多表 INSERT 语句的类型 4-18
多表 INSERT 语句 4-19
无条件 INSERT ALL 4-21
条件 INSERT ALL: 示例 4-23
条件 INSERT ALL 4-24
条件 INSERT FIRST: 示例 4-26
条件 INSERT FIRST 4-27
转换 INSERT (Pivoting INSERT) 4-29

课程安排	4-32
MERGE 语句	4-33
MERGE 语句的语法	4-34
合并行：示例	4-35
课程安排	4-38
跟踪数据更改	4-39
闪回版本查询示例	4-40
VERSIONS BETWEEN 子句	4-42
小测验	4-43
小结	4-44
练习 4：概览	4-45
5 管理不同时区中的数据	
课程目标	5-2
课程安排	5-3
时区	5-4
TIME_ZONE 会话参数	5-5
CURRENT_DATE、CURRENT_TIMESTAMP 和 LOCALTIMESTAMP	5-6
对会话时区的日期和时间进行比较	5-7
DBTIMEZONE 和 SESSIONTIMEZONE	5-9
TIMESTAMP 数据类型	5-10
TIMESTAMP 字段	5-11
DATE 和 TIMESTAMP 之间的区别	5-12
TIMESTAMP 数据类型的比较	5-13
课程安排	5-14
INTERVAL 数据类型	5-15
INTERVAL 字段	5-17
INTERVAL YEAR TO MONTH: 示例	5-18
INTERVAL DAY TO SECOND 数据类型：示例	5-20
课程安排	5-21
EXTRACT	5-22
TZ_OFFSET	5-23
FROM_TZ	5-25
TO_TIMESTAMP	5-26
TO_YMINTERVAL	5-27
TO_DSINTERVAL	5-28

夏令时 5-29
小测验 5-31
小结 5-32
练习 5: 概览 5-33

6 使用子查询检索数据

课程目标 6-2
课程安排 6-3
多列子查询 6-4
列比较 6-5
成对比较子查询 6-6
不成对比较子查询 6-8
课程安排 6-10
标量子查询表达式 6-11
标量子查询: 示例 6-12
课程安排 6-14
相关子查询 6-15
使用相关子查询 6-17
课程安排 6-19
使用 EXISTS 运算符 6-20
查找没有任何雇员的所有部门 6-22
相关 UPDATE 6-23
使用相关 UPDATE 6-24
相关 DELETE 6-26
使用相关 DELETE 6-27
课程安排 6-28
WITH 子句 6-29
WITH 子句: 示例 6-30
递归 WITH 子句 6-32
递归 WITH 子句: 示例 6-33
小测验 6-34
小结 6-35
练习 6: 概览 6-37

7 正则表达式支持功能

- 课程目标 7-2
- 课程安排 7-3
- 正则表达式是什么 7-4
- 使用正则表达式的好处 7-5
- 在 SQL 和 PL/SQL 中使用正则表达式函数和条件 7-6
- 课程安排 7-7
- 元字符是什么 7-8
- 在正则表达式中使用元字符 7-9
- 课程安排 7-11
- 正则表达式函数和条件：语法 7-12
- 使用 REGEXP_LIKE 条件执行基本搜索 7-14
- 使用 REGEXP_REPLACE 函数替换模式 7-15
- 使用 REGEXP_INSTR 函数查找模式 7-16
- 使用 REGEXP_SUBSTR 函数提取子字符串 7-17
- 课程安排 7-18
- 子表达式 7-19
- 结合使用正则表达式支持与子表达式 7-20
- 为什么要访问第 n 个子表达式 7-21
- REGEXP_SUBSTR: 示例 7-22
- 课程安排 7-23
- 使用 REGEXP_COUNT 函数 7-24
- 正则表达式和检查约束条件：示例 7-25
- 小测验 7-26
- 小结 7-27
- 练习 7: 概览 7-28

附录 A: 练习和解答**附录 B: 表说明****附录 C: 使用 SQL Developer**

- 课程目标 C-2
- 什么是 Oracle SQL Developer C-3
- SQL Developer 说明 C-4
- SQL Developer 1.5 界面 C-5
- 创建数据库连接 C-7
- 浏览数据库对象 C-10
- 显示表结构 C-11
- 浏览文件 C-12
- 创建方案对象 C-13
- 创建新表: 示例 C-14
- 使用 SQL 工作表 C-15
- 执行 SQL 语句 C-18
- 保存 SQL 脚本 C-19
- 执行已保存的脚本文件: 方法 1 C-20
- 执行已保存的脚本文件: 方法 2 C-21
- 设置 SQL 代码的格式 C-22
- 使用片段 C-23
- 使用片段: 示例 C-24
- 调试过程和函数 C-25
- 数据库报表 C-26
- 创建用户定义报表 C-28
- 搜索引擎和外部工具 C-29
- 设置首选项 C-30
- 重置 SQL Developer 布局 C-31
- 小结 C-32

附录 D: 使用 SQL*Plus

- 课程目标 D-2
- SQL 和 SQL*Plus 交互 D-3
- SQL 语句和 SQL*Plus 命令 D-5
- SQL*Plus 概览 D-6
- 登录到 SQL*Plus D-7
- 显示表结构 D-8
- SQL*Plus 编辑命令 D-10
- 使用 LIST、n 和 APPEND D-12
- 使用 CHANGE 命令 D-13
- SQL*Plus 文件命令 D-14
- 使用 SAVE 和 START 命令 D-15
- SERVERTOURPUT 命令 D-16
- 使用 SQL*Plus 的 SPOOL 命令 D-17
- 使用 AUTOTRACE 命令 D-18
- 小结 D-19

附录 E: 使用 JDeveloper

- 课程目标 E-2
- Oracle JDeveloper E-3
- 数据库导航器 E-4
- 创建连接 E-5
- 浏览数据库对象 E-6
- 执行 SQL 语句 E-7
- 创建程序单元 E-8
- 编译 E-9
- 运行程序单元 E-10
- 删除程序单元 E-11
- 结构窗口 E-12
- 编辑器窗口 E-13
- 应用程序导航器 E-14
- 部署 Java 存储过程 E-15
- 将 Java 发布到 PL/SQL E-16
- 如何了解有关 JDeveloper 11g 的更多信息 E-17
- 小结 E-18

附录 F：通过将相关数据分组来生成报表

- 课程目标 F-2
- 复习组函数 F-3
- 复习 GROUP BY 子句 F-4
- 复习 HAVING 子句 F-5
- GROUP BY 与 ROLLUP 和 CUBE 运算符一起使用 F-6
- ROLLUP 运算符 F-7
- ROLLUP 运算符：示例 F-8
- CUBE 运算符 F-9
- CUBE 运算符：示例 F-10
- GROUPING 函数 F-11
- GROUPING 函数：示例 F-12
- GROUPING SETS F-13
- GROUPING SETS：示例 F-15
- 组合列 F-17
- 组合列：示例 F-19
- 级联分组 F-21
- 级联分组：示例 F-22
- 小结 F-23

附录 G：分层检索

- 课程目标 G-2
- EMPLOYEES 表中的示例数据 G-3
- 自然树结构 G-4
- 分层查询 G-5
- 遍历树 G-6
- 遍历树：自下而上 G-8
- 遍历树：自上而下 G-9
- 使用 LEVEL 伪列确定行的等级 G-10
- 使用 LEVEL 和 LPAD 设置分层报表的格式 G-11
- 修剪分支 G-13
- 小结 G-14

附录 H: 编写高级脚本

- 课程目标 H-2
- 使用 SQL 生成 SQL H-3
- 创建基本脚本 H-4
- 控制环境 H-5
- 完整脚本 H-6
- 将表内容转储到文件 H-7
- 生成动态谓词 H-9
- 小结 H-11

附录 I: Oracle DB 体系结构组件

- 课程目标 I-2
- Oracle DB 体系结构: 概览 I-3
- Oracle DB Server 结构 I-4
- 连接到数据库 I-5
- 与 Oracle DB 交互 I-6
- Oracle 内存体系结构 I-8
- 进程体系结构 I-10
- 数据库写进程 I-12
- 日志写进程 I-13
- 检查点进程 I-14
- 系统监视器进程 I-15
- 进程监视器进程 I-16
- Oracle DB 存储体系结构 I-17
- 逻辑和物理数据库结构 I-19
- 处理 SQL 语句 I-21
- 处理查询 I-22
- 共享池 I-23
- 数据库缓冲区高速缓存 I-25
- 程序全局区 (PGA) I-26
- 处理 DML 语句 I-27
- 重做日志缓冲区 I-29
- 回退段 I-30
- COMMIT 处理 I-31
- Oracle DB 体系结构小结 I-33

附加练习和解答

附录 A

练习和解答

目录

第 I 课的练习和解答	3
练习 I-1: 访问 SQL Developer 资源.....	4
练习 I-2: 使用 SQL Developer.....	5
练习解答 I-1: 访问 SQL Developer 资源.....	7
练习解答 I-2: 使用 SQL Developer.....	8
第 1 课的练习和解答.....	17
练习 1-1: 控制用户访问.....	17
练习解答 1-1: 控制用户访问.....	20
第 2 课的练习和解答.....	24
练习 2-1: 管理方案对象	24
练习解答 2-1: 管理方案对象	30
第 3 课的练习和解答.....	36
练习 3-1: 使用数据字典视图管理对象	36
练习解答 3-1: 使用数据字典视图管理对象	39
第 4 课的练习和解答.....	42
练习 4-1: 处理大型数据集	42
练习解答 4-1: 处理大型数据集	46
第 5 课的练习和解答.....	50
练习 5-1: 管理不同时区中的数据	50
练习解答 5-1: 管理不同时区中的数据	53
第 6 课的练习和解答.....	56
练习 6-1: 使用子查询检索数据	56
练习解答 6-1: 使用子查询检索数据	59
第 7 课的练习和解答.....	62
练习 7-1: 正则表达式支持功能	62
练习解答 7-1: 正则表达式支持功能	64

第 I 课的练习和解答

在本练习中，您要复习可用的 SQL Developer 资源。还要了解在本课程中使用的用户帐户。然后，启动 SQL Developer，创建一个新数据库连接，浏览您的 HR 表。还要设置一些 SQL Developer 首选项，执行 SQL 语句，然后使用 SQL 工作表执行一个匿名 PL/SQL 块。最后，要访问本课程中可使用的 Oracle Database 11g 文档和其它有用网站，并将其放入收藏夹。

练习 I-1：访问 SQL Developer 资源

在本练习中，您需要执行以下任务：

- 1) 访问 SQL Developer 主页。
 - a. 访问 SQL Developer 的联机主页：
http://www.oracle.com/technology/products/database/sql_developer/index.html
 - b. 给此页添加书签，以便于日后访问。
- 2) 访问 SQL Developer 教程，其网址为：
<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>。然后，复习以下部分及相关演示：
 - a) What to Do First (准备工作)
 - b) Working with Database Objects (处理数据库对象)
 - c) Accessing Data (访问数据)

练习 I-2：使用 SQL Developer

- 1) 使用桌面图标启动 SQL Developer。
- 2) 使用以下信息创建数据库连接：
 - a) Connection Name (连接名) : myconnection
 - b) Username (用户名) : oraxx, 其中 xx 为您的 PC 编号 (可以让教师在 ora21-ora40 帐户范围内为您分配一个 ora 帐户。)
 - c) Password (口令) : oraxx
 - d) Hostname (主机名) : localhost
 - e) Port (端口) : 1521
 - f) SID: ORCL (或者教师提供给您的值)
- 3) 测试新连接。如果状态为“Success (成功)”，则可使用这个新连接连接到数据库。
 - a) 在“New>Select Database Connection (新建/选择数据库连接)”窗口中单击“Test (测试)”按钮。
 - b) 如果状态为“成功”，则单击“Connect (连接)”按钮。
- 4) 浏览 EMPLOYEES 表的结构并显示其数据。
 - a) 单击 myconnection 连接旁边的加号将其展开。
 - b) 通过单击旁边的加号展开“Tables (表)”图标。
 - c) 显示 EMPLOYEES 表的结构。
 - d) 查看 DEPARTMENTS 表的数据。
- 5) 通过在 SQL 工作表区域中执行一些基本 SELECT 语句来查询 EMPLOYEES 表中的数据。使用“Execute Statement (执行语句)”图标 (或按 F9) 和“Run Script (运行脚本)”图标 (或按 F5) 执行 SELECT 语句。在相应选项卡页上复查用以上两种方法执行 SELECT 语句的结果。
 - a) 编写一个查询，选择其薪金少于或等于 \$3,000 的任何雇员的姓氏和薪金。
 - b) 编写一个查询，显示无资格获得佣金的所有雇员的姓氏、职务 ID 和佣金。

练习 I-2: 使用 SQL Developer (续)

- 6) 将您的脚本路径首选项设置为 /home/oracle/labs/sql2。
- 选择“Tools > Preferences > Database > Worksheet Parameters（工具 > 首选项 > 数据库 > 工作表参数）”。
 - 在“Select default path to look for scripts（选择用来查找脚本的默认路径）”字段中输入值。
- 7) 在“Enter SQL Statement（输入 SQL 语句）”框中输入以下内容:
- ```
SELECT employee_id, first_name, last_name,
 FROM employees;
```
- 8) 使用“File > Save As（文件 > 另存为）”菜单项，将该 SQL 语句保存在一个脚本文件。
- 选择“File > Save As（文件 > 另存为）”。
  - 指定文件名 intro\_test.sql。
  - 将文件置于 /home/oracle/labs/sql2/labs 文件夹下。
- 9) 在 /home/oracle/labs/sql2/labs 文件夹中打开并运行 confidence.sql，然后观察其输出。

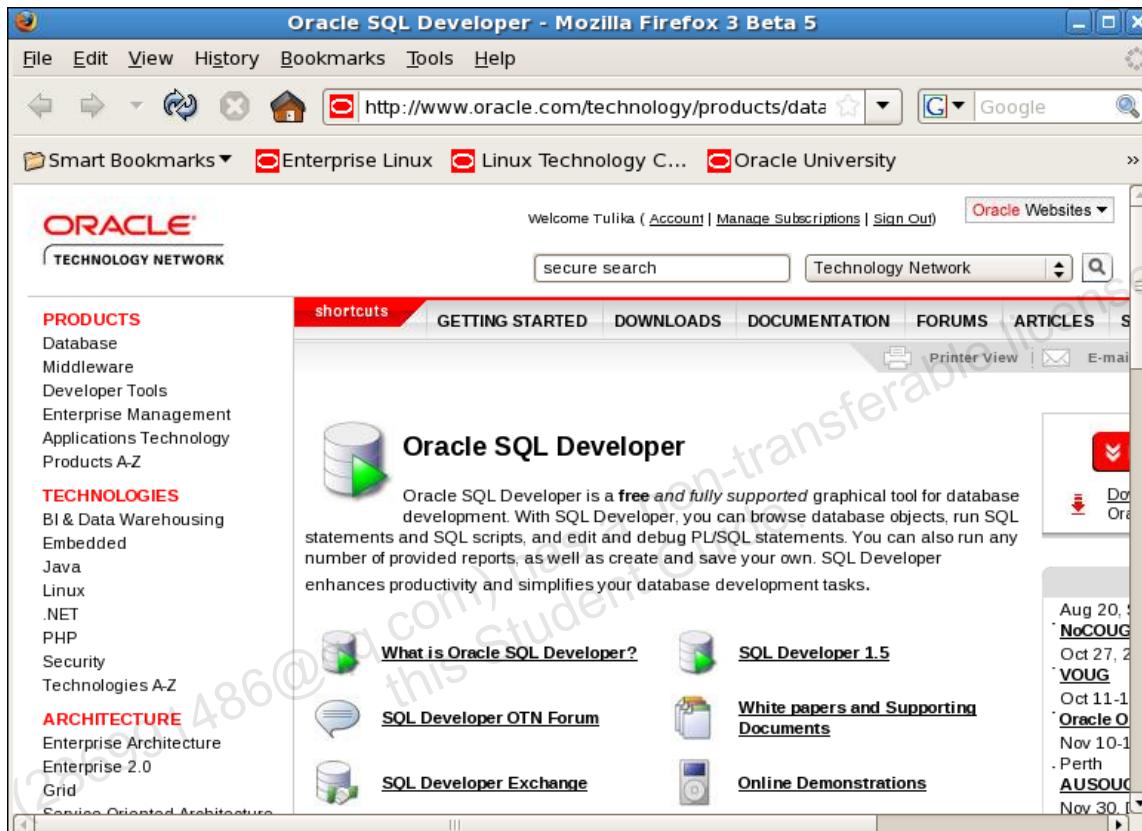
## 练习解答 I-1：访问 SQL Developer 资源

1) 访问 SQL Developer 主页。

a) 访问 SQL Developer 的联机主页：

[http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)

SQL Developer 的主页显示如下：



b) 将该网页放入收藏夹以便将来访问。

2) 访问 SQL Developer 教程，其网址为：

<http://st-curriculum.oracle.com/tutorial/SQLDeveloper/index.htm>

然后，复习以下部分及相关演示：

- a) What to Do First (准备工作)
- b) Working with Database Objects (处理数据库对象)
- c) Accessing Data (访问数据)

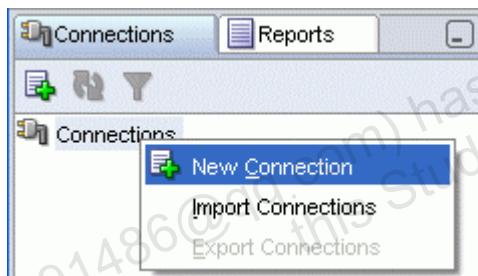
## 练习解答 I-2：使用 SQL Developer

- 1) 使用桌面图标启动 SQL Developer。

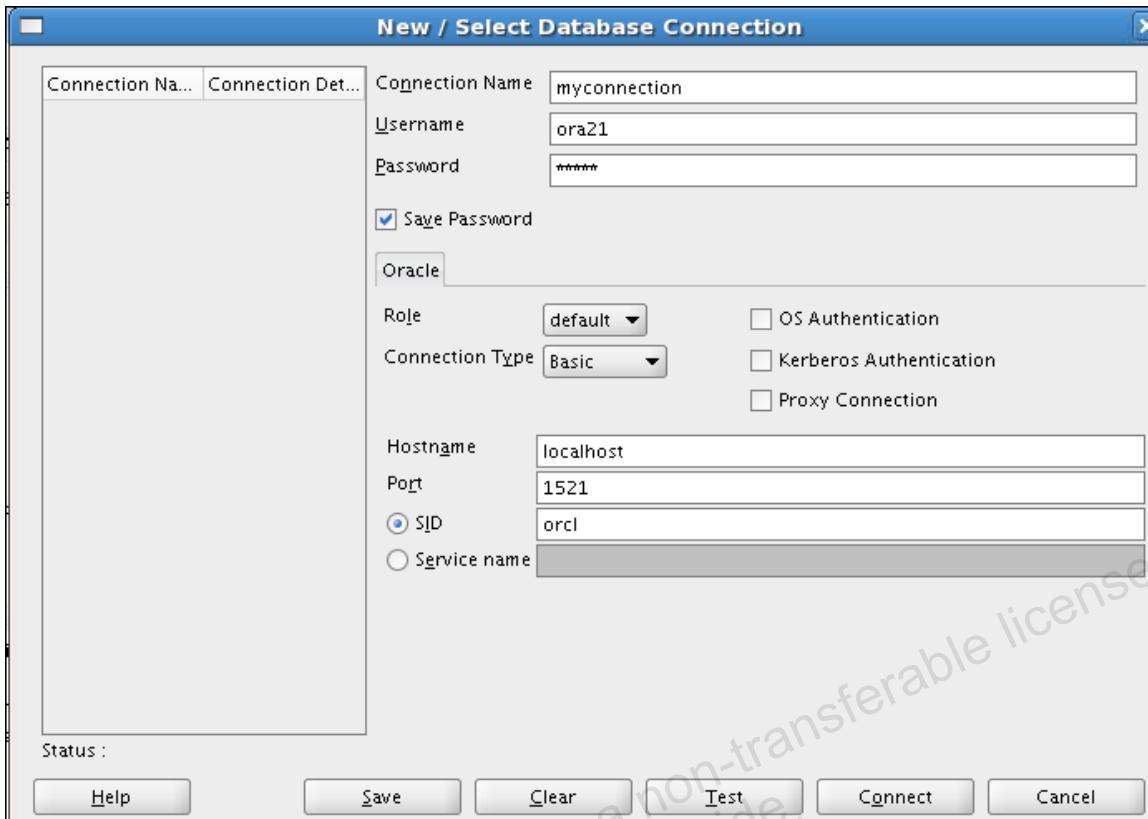


- 2) 使用以下信息创建数据库连接：

- a. Connection Name (连接名) : myconnection
- b. Username (用户名) : oraxx (可以让教师在 ora21-ora40 帐户范围内为您分配一个 ora 帐户。)
- c. Password (口令) : oraxx
- d. Hostname (主机名) : localhost
- e. Port (端口) : 1521
- f. SID: ORCL (或者教师提供给您的值)



## 练习解答 I-2: 使用 SQL Developer (续)



3) 测试新连接。如果状态为“Success (成功)”，则可使用这个新连接连接到数据库。

a) 单击“New/Select Database Connection (新建/选择数据库连接)”窗口中的“Test (测试)”按钮。



b) 如果状态为“Success (成功)”，则单击“Connect (连接)”按钮。



## 练习解答 I-2: 使用 SQL Developer (续)

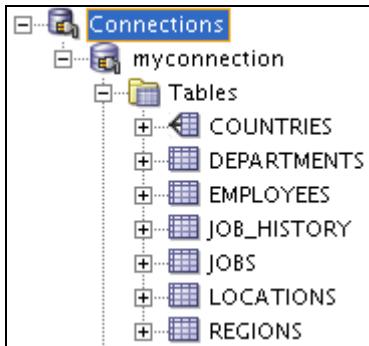
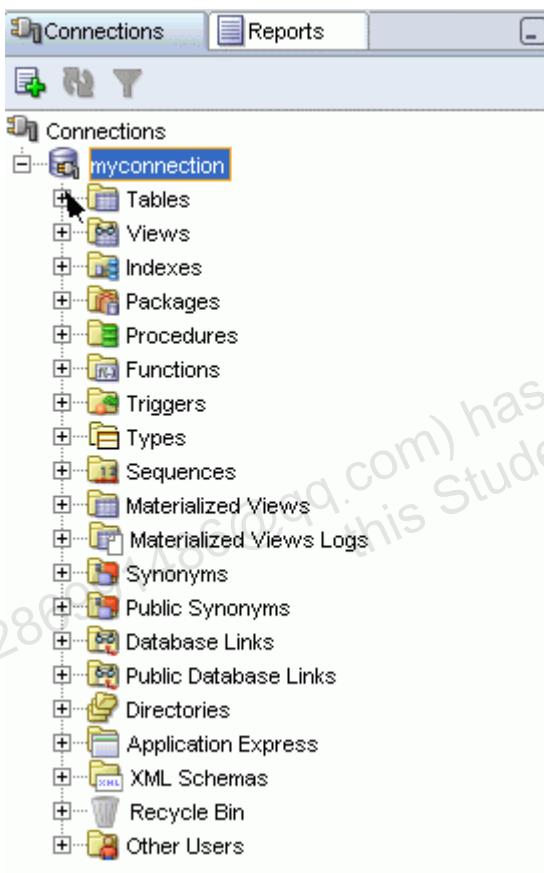
### 浏览表

4) 浏览 EMPLOYEES 表的结构并显示其数据。

a) 单击 myconnection 连接旁边的加号将其展开。



b) 单击 “Tables (表)” 图标旁边的加号将其展开。



## 练习解答 I-2: 使用 SQL Developer (续)

- c) 显示 EMPLOYEES 表的结构。

单击 **EMPLOYEES** 表。此时在“Columns (列)”选项卡上显示 EMPLOYEES 表中的列，如下所示：

| Column Name    | Data Type         | Nullable | Data Default | COLUMN ID | Primary Key | COMMENTS                               |
|----------------|-------------------|----------|--------------|-----------|-------------|----------------------------------------|
| EMPLOYEE_ID    | NUMBER(6,0)       | No       | (null)       | 1         | 1           | Primary key of employee                |
| FIRST_NAME     | VARCHAR2(20 BYTE) | Yes      | (null)       | 2         | (null)      | First name of the employee             |
| LAST_NAME      | VARCHAR2(25 BYTE) | No       | (null)       | 3         | (null)      | Last name of the employee              |
| EMAIL          | VARCHAR2(25 BYTE) | No       | (null)       | 4         | (null)      | Email id of the employee               |
| PHONE_NUMBER   | VARCHAR2(20 BYTE) | Yes      | (null)       | 5         | (null)      | Phone number of the employee           |
| HIRE_DATE      | DATE              | No       | (null)       | 6         | (null)      | Date when the employee was hired       |
| JOB_ID         | VARCHAR2(10 BYTE) | No       | (null)       | 7         | (null)      | Current job of the employee            |
| SALARY         | NUMBER(8,2)       | Yes      | (null)       | 8         | (null)      | Monthly salary of the employee         |
| COMMISSION_PCT | NUMBER(2,2)       | Yes      | (null)       | 9         | (null)      | Commission percent                     |
| MANAGER_ID     | NUMBER(6,0)       | Yes      | (null)       | 10        | (null)      | Manager id of the employee             |
| DEPARTMENT_ID  | NUMBER(4,0)       | Yes      | (null)       | 11        | (null)      | Department id where the employee works |

- d) 查看 DEPARTMENTS 表的数据。

在连接导航器中，右键单击“DEPARTMENTS”表。然后单击“Data (数据)”选项卡。

| DEPARTMENT_ID | DEPARTMENT_NAME  | MANAGER_ID | LOCATION_ID |
|---------------|------------------|------------|-------------|
| 1             | Administration   | 200        | 1700        |
| 2             | Marketing        | 201        | 1800        |
| 3             | Purchasing       | 114        | 1700        |
| 4             | Human Resources  | 203        | 2400        |
| 5             | Shipping         | 121        | 1500        |
| 6             | IT               | 103        | 1400        |
| 7             | Public Relations | 204        | 2700        |

## 练习解答 I-2: 使用 SQL Developer (续)

- 5) 通过在 SQL 工作表区域中执行一些基本 SELECT 语句来查询 EMPLOYEES 表中的数据。使用“Execute Statement (执行语句)”图标（或按 F9）和“Run Script (运行脚本)”图标（或按 F5）执行 SELECT 语句。在相应选项卡页上复查用以上两种方法执行 SELECT 语句的结果。

- a) 编写一个查询，选择其薪金少于或等于 \$3,000 的任何雇员的姓氏和薪金。

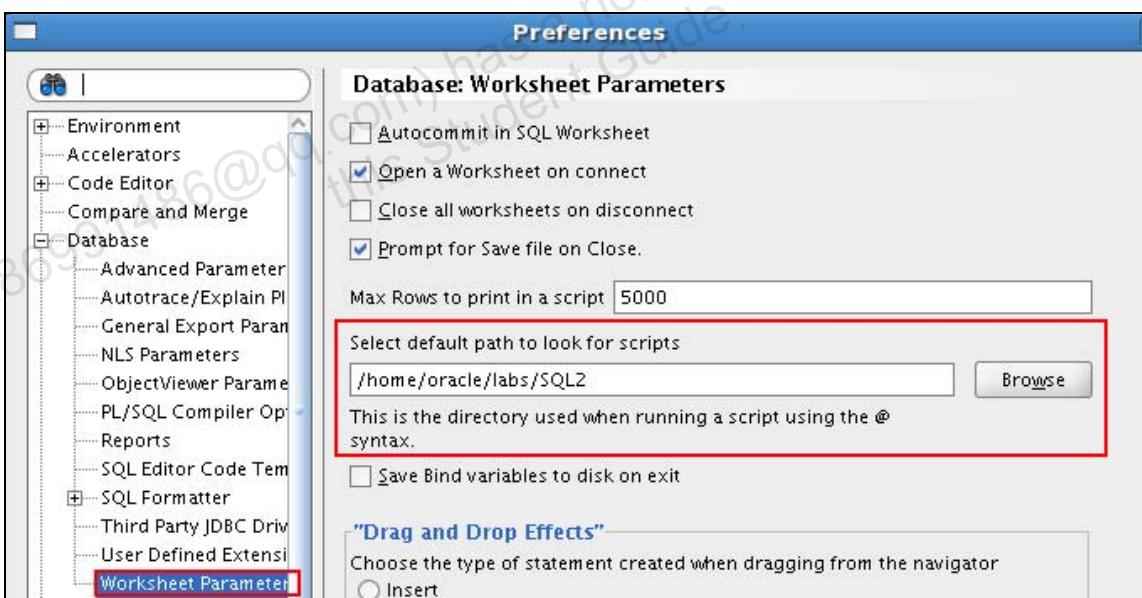
```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000;
```

- b) 编写一个查询，显示无资格获得佣金的所有雇员的姓氏、职务 ID 和佣金。

```
SELECT last_name, job_id, commission_pct
FROM employees
WHERE commission_pct IS NULL;
```

- 6) 将您的脚本路径首选项设置为 /home/oracle/labs/sql2。

- a) 选择“Tools > Preferences > Database > Worksheet Parameters (工具 > 首选项 > 数据库 > 工作表参数)”。  
b) 在“Select default path to look for scripts (选择用来查找脚本的默认路径)”字段中输入值。然后单击“OK (确定)”。



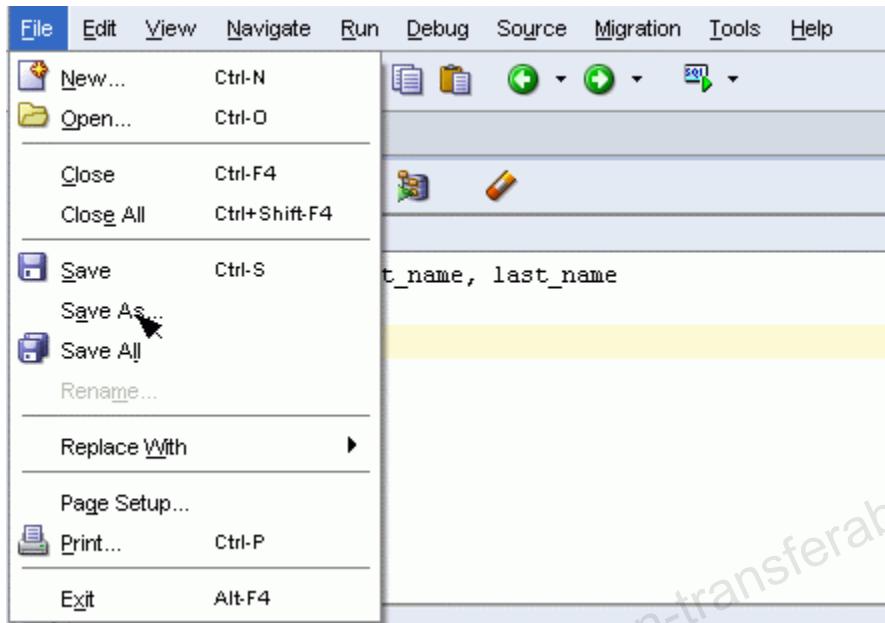
- 7) 输入以下 SQL 语句：

```
SELECT employee_id, first_name, last_name
FROM employees;
```

## 练习解答 I-2: 使用 SQL Developer (续)

8) 使用 “File > Save As (文件 > 另存为)” 菜单项，将该 SQL 语句保存在一个脚本文件。

a) 选择 “File > Save As (文件 > 另存为)”

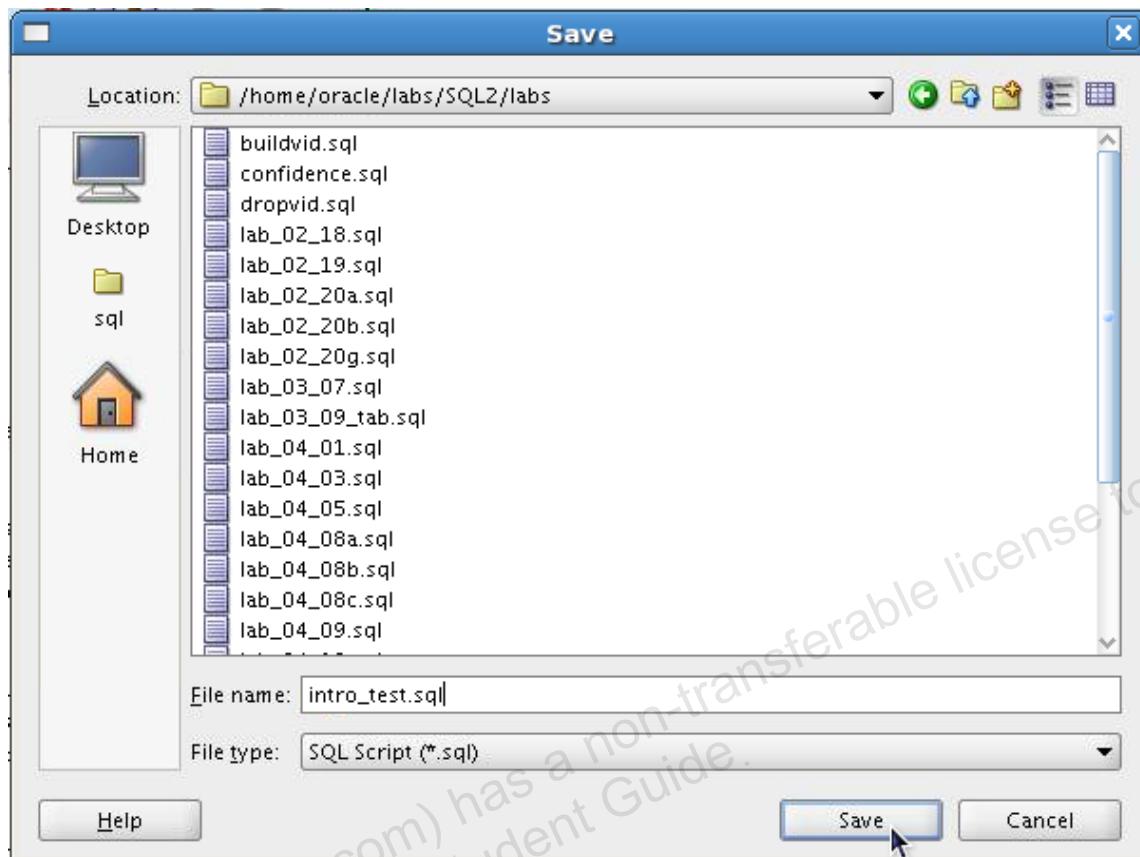


b) 指定文件名 **intro\_test.sql**

在 “File name (文件名)” 文本框中键入 `intro_test.sql`。

## 练习解答 I-2: 使用 SQL Developer (续)

- c) 将文件置于 /home/oracle/labs/SQL2/labs 文件夹下。

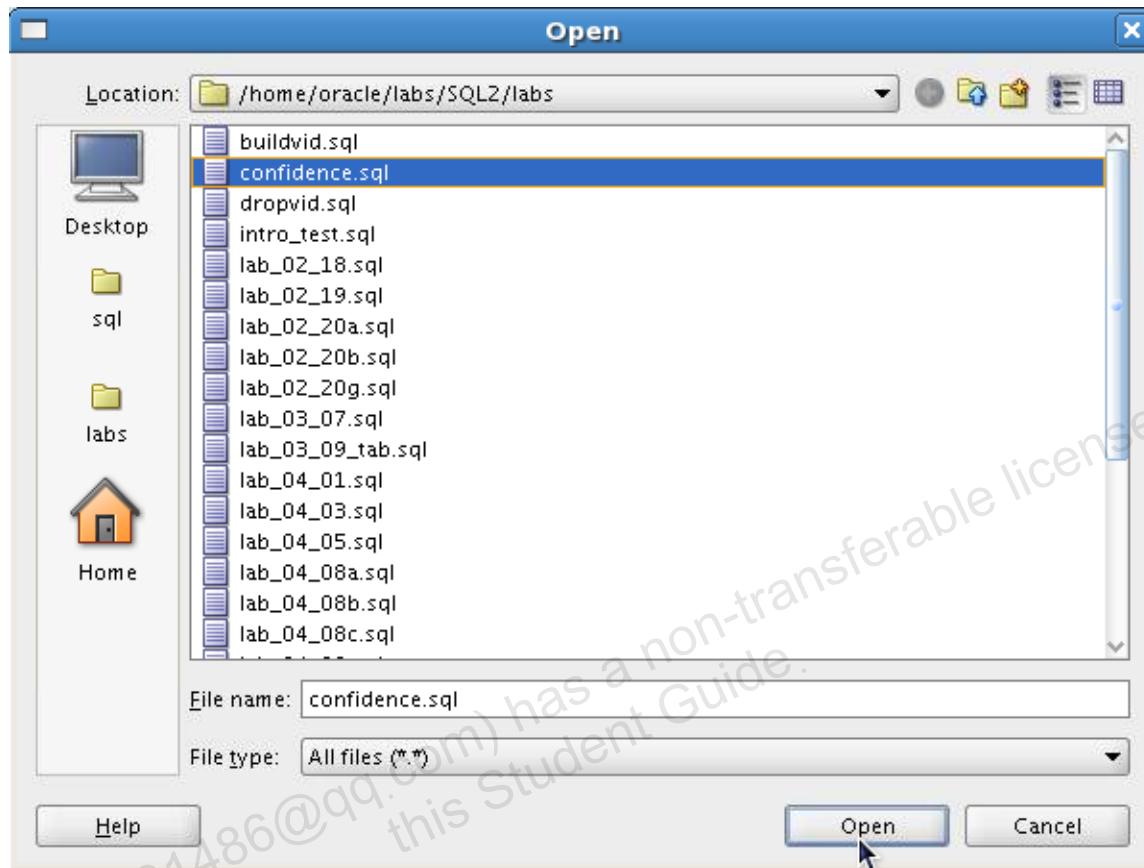


然后，单击“Save（保存）”。

## 练习解答 I-2: 使用 SQL Developer (续)

- 9) 在 /home/oracle/labs/SQL2/labs 文件夹中打开并运行 confidence.sql, 然后观察其输出。

使用“File > Open (文件 > 打开)”菜单项打开 confidence.sql 脚本文件。



然后, 按 F5 执行该脚本。

以下是预期的结果:

```
COUNT (*)

8
1 rows selected

COUNT (*)

107
1 rows selected
```

## 练习解答 I-2: 使用 SQL Developer (续)

```
COUNT(*)

25

1 rows selected

COUNT(*)

4

1 rows selected

COUNT(*)

23

1 rows selected

COUNT(*)

27

1 rows selected

COUNT(*)

19

1 rows selected

COUNT(*)

10

1 rows selected
```

## 第 1 课的练习和解答

### 练习 1-1：控制用户访问

1. 用户需要拥有什么权限才能登录到 Oracle Server？是系统权限还是对象权限？
2. 用户需要拥有什么权限才能创建表？
3. 如果您创建了一个表，谁可以将此表的权限传递给其他用户？
4. 您是 DBA。您创建的许多用户都需要拥有相同的系统权限。应使用什么方法使工作变得更容易？
5. 使用什么命令可更改您的口令？
6. User21 为 EMP 表的所有者，他使用 WITH GRANT OPTION 子句将 DELETE 权限授予了 User22。User22 又将 EMP 的 DELETE 权限授予了 User23。User21 现在发现 User23 拥有该权限，于是撤消了 User22 的该权限。现在哪个用户可在 EMP 表中执行删除操作？
7. 您要授予 SCOTT 在 DEPARTMENTS 表中更新数据的权限。您还希望 SCOTT 能将此权限授予其他用户。应使用什么命令？

要完成问题 8 及后续问题，需要使用 **SQL Developer** 连接到数据库。如果尚未建立连接，请执行以下步骤进行连接：

1. 单击 **SQL Developer** 桌面图标。
2. 在连接导航器中，使用教师提供的 **oraxxx** 帐户和相应口令登录到数据库。
8. 授予另一个用户对您的表的查询权限。然后验证该用户是否可以使用该权限。  
**注：**在本练习中请与另一小组合作。例如，如果您是用户 ora21，请与另一个用户 ora22 合作。
  - a. 授予另一个用户查看您的 REGIONS 表中的记录的权限。包含一个选项，让此用户再将此权限授予其他用户。
  - b. 让该用户查询您的 REGIONS 表。
  - c. 让该用户将查询权限传递给第三个用户（例如，ora23）。
  - d. 撤消执行步骤 b 的用户的权限。**注：**每个小组均可独立运行练习 9 和 10。

## 练习 1-1：控制用户访问（续）

9. 授予另一个用户在您的 COUNTRIES 表上执行查询和数据处理的权限。确保该用户不能将这些权限传递给其他用户。

10. 撤消在 COUNTRIES 表上已授予另一个用户的权限。

**注：**在练习 11 至 17 中，请与另一个小组合作。

11. 授予另一个用户访问您的 DEPARTMENTS 表的权限。让该用户授予您对其 DEPARTMENTS 表进行查询访问的权限。

12. 查询您的 DEPARTMENTS 表中的所有行。

|   | DEPARTMENT_ID | DEPARTMENT_NAME  | MANAGER_ID | LOCATION_ID |
|---|---------------|------------------|------------|-------------|
| 1 | 10            | Administration   | 200        | 1700        |
| 2 | 20            | Marketing        | 201        | 1800        |
| 3 | 30            | Purchasing       | 114        | 1700        |
| 4 | 40            | Human Resources  | 203        | 2400        |
| 5 | 50            | Shipping         | 121        | 1500        |
| 6 | 60            | IT               | 103        | 1400        |
| 7 | 70            | Public Relations | 204        | 2700        |
| 8 | 80            | Sales            | 145        | 2500        |

...

13. 在 DEPARTMENTS 表中添加一行新行。小组 1 应添加 Education 部门，其部门编号为 500，小组 2 应添加 Human Resources 部门，其部门编号为 510。请查询对方小组的表。

14. 为对方小组的 DEPARTMENTS 表创建同义词。

15. 使用同义词查询对方小组的 DEPARTMENTS 表中的所有行。

小组 1 执行 SELECT 语句的结果：

|    | DEPARTMENT_ID | DEPARTMENT_NAME  | MANAGER_ID | LOCATION_ID |
|----|---------------|------------------|------------|-------------|
| 16 | 160           | Benefits         | (null)     | 1700        |
| 17 | 170           | Manufacturing    | (null)     | 1700        |
| 18 | 180           | Construction     | (null)     | 1700        |
| 19 | 190           | Contracting      | (null)     | 1700        |
| 20 | 200           | Operations       | (null)     | 1700        |
| 21 | 210           | IT Support       | (null)     | 1700        |
| 22 | 220           | NOC              | (null)     | 1700        |
| 23 | 230           | IT Helpdesk      | (null)     | 1700        |
| 24 | 240           | Government Sales | (null)     | 1700        |
| 25 | 250           | Retail Sales     | (null)     | 1700        |
| 26 | 260           | Recruiting       | (null)     | 1700        |
| 27 | 270           | Payroll          | (null)     | 1700        |
| 28 | 510           | Human Resources  | (null)     | (null)      |

## 练习 1-1：控制用户访问（续）

小组 2 执行 SELECT 语句的结果：

| DEPARTMENT_ID | DEPARTMENT_NAME      | MANAGER_ID | LOCATION_ID |
|---------------|----------------------|------------|-------------|
| 16            | 160 Benefits         | (null)     | 1700        |
| 17            | 170 Manufacturing    | (null)     | 1700        |
| 18            | 180 Construction     | (null)     | 1700        |
| 19            | 190 Contracting      | (null)     | 1700        |
| 20            | 200 Operations       | (null)     | 1700        |
| 21            | 210 IT Support       | (null)     | 1700        |
| 22            | 220 NOC              | (null)     | 1700        |
| 23            | 230 IT Helpdesk      | (null)     | 1700        |
| 24            | 240 Government Sales | (null)     | 1700        |
| 25            | 250 Retail Sales     | (null)     | 1700        |
| 26            | 260 Recruiting       | (null)     | 1700        |
| 27            | 270 Payroll          | (null)     | 1700        |
| 28            | 500 Education        | (null)     | (null)      |

16. 撤消对方小组的 SELECT 权限。
17. 删除执行步骤 13 时在 DEPARTMENTS 表中插入的行，然后保存更改。

## 练习解答 1-1：控制用户访问

要完成问题 8 及后续问题，需要使用 SQL Developer 连接到数据库。

1. 用户需要拥有什么权限才能登录到 Oracle Server？这是系统权限还是对象权限？

**CREATE SESSION** 系统权限

2. 用户需要拥有什么权限才能创建表？

**CREATE TABLE** 权限

3. 如果您创建了一个表，谁可以将此表的权限传递给其他用户？

您可以。如果您使用 **WITH GRANT OPTION** 将这些权限授予给了其他用户，则这些用户也可以。

4. 您是 DBA。您创建的许多用户都需要拥有相同的系统权限。

应使用什么方法使工作变得更容易？

创建拥有系统权限的角色，然后将此角色授予用户。

5. 使用什么命令可更改您的口令？

**ALTER USER** 语句

6. User21 是 EMP 表的所有者，他使用 **WITH GRANT OPTION** 子句将 **DELETE** 权限授予 User22。User22 又将 EMP 的 **DELETE** 权限授予 User23。

User21 现在发现 User23 拥有该权限，于是撤消了 User22 的该权限。现在哪个用户可从 EMP 表中删除数据？

只有 User21

7. 您要授予 SCOTT 在 DEPARTMENTS 表中更新数据的权限。您还希望 SCOTT 能将此权限授予其他用户。应使用什么命令？

**GRANT UPDATE ON departments TO scott WITH GRANT OPTION;**

## 练习解答 1-1：控制用户访问（续）

8. 授予另一个用户对您的表的查询权限。然后验证该用户是否可以使用该权限。

**注：**在本练习中请与另一小组合作。例如，如果您是用户 ora21，请与另一个用户 ora22 合作。

- a) 授予另一个用户查看您的 REGIONS 表中记录的权限。包含一个选项，让此用户再将此权限授予其他用户。

小组 1 执行以下语句：

```
GRANT select
ON regions
TO <team2_oraxx> WITH GRANT OPTION;
```

- b) 让该用户查询您的 REGIONS 表。

小组 2 执行以下语句：

```
SELECT * FROM <team1_oraxx>.regions;
```

- c) 允许该用户将查询权限传递给第三个用户（例如，ora23）。

小组 2 执行以下语句：

```
GRANT select
ON <team1_oraxx>.regions
TO <team3_oraxx>;
```

- d) 撤销执行步骤 b 的用户的权限。

小组 1 执行以下语句：

```
REVOKE select
ON regions
FROM <team2_oraxx>;
```

9. 授予另一个用户在您的 COUNTRIES 表上执行查询和数据处理的权限。确保该用户不能将这些权限传递给其他用户。

小组 1 执行以下语句：

```
GRANT select, update, insert
ON COUNTRIES
TO <team2_oraxx>;
```

10. 撤消在 COUNTRIES 表上已授予另一个用户的权限。

小组 1 执行以下语句：

```
REVOKE select, update, insert ON COUNTRIES FROM <team2_oraxx>;
```

**注：**在练习 11 至 17 中，请与另一个小组合作。

## 练习解答 1-1：控制用户访问（续）

11. 授予另一个用户访问您的 DEPARTMENTS 表的权限。让该用户授予您对其 DEPARTMENTS 表进行查询访问的权限。

小组 2 执行以下 GRANT 语句。

```
GRANT select
ON departments
TO <team1_oraxx>;
```

小组 1 执行以下 GRANT 语句。

```
GRANT select
ON departments
TO <team2_oraxx>;
```

其中，*<team1\_oraxx>* 是小组 1 的用户名，*<team2\_oraxx>* 是小组 2 的用户名。

12. 查询您的 DEPARTMENTS 表中的所有行。

```
SELECT *
FROM departments;
```

13. 在 DEPARTMENTS 表中添加一行。小组 1 应添加 Education 部门，其部门编号为 500，小组 2 应添加 Human Resources 部门，其部门编号为 510。  
请查询对方小组的表。

小组 1 执行以下 INSERT 语句。

```
INSERT INTO departments(department_id, department_name)
VALUES (500, 'Education');
COMMIT;
```

小组 2 执行以下 INSERT 语句。

```
INSERT INTO departments(department_id, department_name)
VALUES (510, 'Human Resources');
COMMIT;
```

14. 为对方小组的 DEPARTMENTS 表创建同义词。

小组 1 创建名为 team2 的同义词。

```
CREATE SYNONYM team2
FOR <team2_oraxx>.DEPARTMENTS;
```

小组 2 创建名为 team1 的同义词。

```
CREATE SYNONYM team1
FOR <team1_oraxx>.DEPARTMENTS;
```

## 练习解答 1-1：控制用户访问（续）

15. 使用同义词查询对方小组的 DEPARTMENTS 表中的所有行。

小组 1 执行以下 SELECT 语句。

```
SELECT *
 FROM team2;
```

小组 2 执行以下 SELECT 语句。

```
SELECT *
 FROM team1;
```

16. 撤消对方小组的 SELECT 权限。

小组 1 撤销权限。

```
REVOKE select
 ON departments
 FROM <team2_oraxx>;
```

小组 2 撤销权限。

```
REVOKE select
 ON departments
 FROM <team1_oraxx>;
```

17. 删除执行步骤 8 时在 DEPARTMENTS 表中插入的行，然后保存更改。

小组 1 执行以下 DELETE 语句。

```
DELETE FROM departments
 WHERE department_id = 500;
 COMMIT;
```

小组 2 执行以下 DELETE 语句。

```
DELETE FROM departments
 WHERE department_id = 510;
 COMMIT;
```

## 第 2 课的练习和解答

### 练习 2-1：管理方案对象

在本练习中，您将使用 ALTER TABLE 命令修改列和添加约束条件。CREATE INDEX 命令与 CREATE TABLE 命令一起使用的目的是为了在创建表时创建索引。您创建的是外部表。

- 根据下面的表实例图表创建 DEPT2 表。在 SQL 工作表中输入语法。然后，执行该语句来创建表。确认该表已创建。

| 列名    | ID     | NAME     |
|-------|--------|----------|
| 键类型   |        |          |
| 空值/唯一 |        |          |
| FK 表  |        |          |
| FK 列  |        |          |
| 数据类型  | NUMBER | VARCHAR2 |
| 长度    | 7      | 25       |

| Name            | Null | Type         |
|-----------------|------|--------------|
| ID              |      | NUMBER(7)    |
| NAME            |      | VARCHAR2(25) |
| 2 rows selected |      |              |

- 使用 DEPARTMENTS 表中的数据填充 DEPT2 表。仅包括所需的列。

## 练习 2-1：管理方案对象（续）

3. 根据下面的表实例图表创建 EMP2 表。在 SQL 工作表中输入语法。然后，执行该语句来创建表。确认该表已创建。

| 列名    | ID     | LAST_NAME | FIRST_NAME | DEPT_ID |
|-------|--------|-----------|------------|---------|
| 键类型   |        |           |            |         |
| 空值/唯一 |        |           |            |         |
| FK 表  |        |           |            |         |
| FK 列  |        |           |            |         |
| 数据类型  | NUMBER | VARCHAR2  | VARCHAR2   | NUMBER  |
| 长度    | 7      | 25        | 25         | 7       |

| Name       | Null | Type         |
|------------|------|--------------|
| ID         |      | NUMBER(7)    |
| LAST_NAME  |      | VARCHAR2(25) |
| FIRST_NAME |      | VARCHAR2(25) |
| DEPT_ID    |      | NUMBER(7)    |

4 rows selected

4. 修改 EMP2 表，允许输入更长的雇员姓氏。确认修改。

| Name       | Null | Type         |
|------------|------|--------------|
| ID         |      | NUMBER(7)    |
| LAST_NAME  |      | VARCHAR2(50) |
| FIRST_NAME |      | VARCHAR2(25) |
| DEPT_ID    |      | NUMBER(7)    |

4 rows selected

5. 根据 EMPLOYEES 表的结构创建 EMPLOYEES2 表。只包括 EMPLOYEE\_ID、FIRST\_NAME、LAST\_NAME、SALARY 和 DEPARTMENT\_ID 列。将新表中的各列分别命名为 ID、FIRST\_NAME、LAST\_NAME、SALARY 和 DEPT\_ID。  
 6. 删除 EMP2 表。  
 7. 在回收站中进行查询，查看该表是否存在。

| ORIGINAL_NAME  | OPERATION | DROPTIME            |
|----------------|-----------|---------------------|
| 17 EMP_NEW_SAL | DROP      | 2009-05-22:14:44:15 |
| 18 EMP2        | DROP      | 2009-05-22:14:57:57 |

## 练习 2-1：管理方案对象（续）

8. 将 EMP2 表还原到执行 DROP 语句之前的状态。

| Name                   | Null | Type         |
|------------------------|------|--------------|
| ID                     |      | NUMBER(7)    |
| LAST_NAME              |      | VARCHAR2(50) |
| FIRST_NAME             |      | VARCHAR2(25) |
| DEPT_ID                |      | NUMBER(7)    |
| <i>4 rows selected</i> |      |              |

9. 从 EMPLOYEES2 表中删除 FIRST\_NAME 列。通过检查该表的描述来确认修改。

| Name                   | Null     | Type         |
|------------------------|----------|--------------|
| ID                     |          | NUMBER(6)    |
| LAST_NAME              | NOT NULL | VARCHAR2(25) |
| SALARY                 |          | NUMBER(8,2)  |
| DEPT_ID                |          | NUMBER(4)    |
| <i>4 rows selected</i> |          |              |

10. 在 EMPLOYEES2 表中，将 DEPT\_ID 列标记为 UNUSED。通过检查该表的描述来确认修改。

| Name                   | Null     | Type         |
|------------------------|----------|--------------|
| ID                     |          | NUMBER(6)    |
| LAST_NAME              | NOT NULL | VARCHAR2(25) |
| SALARY                 |          | NUMBER(8,2)  |
| <i>3 rows selected</i> |          |              |

11. 从 EMPLOYEES2 表中删除所有 UNUSED 列。通过检查该表的描述来确认修改。

12. 在 EMP2 表的 ID 列上添加表级别 PRIMARY KEY 约束条件。应在创建时命名该约束条件。将约束条件命名为 my\_emp\_id\_pk。

13. 使用 ID 列为 DEPT2 表创建 PRIMARY KEY 约束条件。应在创建时命名该约束条件。将该约束条件命名为 my\_dept\_id\_pk。

14. 在 EMP2 表上添加外键引用，确保雇员不会被分配到不存在的部门。将该约束条件命名为 my\_emp\_dept\_id\_fk。

15. 修改 EMP2 表。添加数据类型为 NUMBER（精度为 2，小数位数为 2）的 COMMISSION 列。在 COMMISSION 列中添加约束条件，确保佣金值大于零。

16. 删除 EMP2 和 DEPT2 表，使其无法还原。检查回收站。

## 练习 2-1：管理方案对象（续）

17. 根据下面的表实例图表创建 DEPT\_NAMED\_INDEX 表。将 PRIMARY KEY 列的索引命名为 DEPT\_PK\_IDX。

| 列名   | Deptno | Dname    |
|------|--------|----------|
| 主键   | Yes    |          |
| 数据类型 | Number | VARCHAR2 |
| 长度   | 4      | 30       |

18. 创建外部表 library\_items\_ext。使用 ORACLE\_LOADER 访问驱动程序。

注：已经为此练习创建 emp\_dir 目录和 library\_items.dat 文件。

library\_items.dat 中记录的格式如下：

```
2354, 2264, 13.21, 150,
2355, 2289, 46.23, 200,
2355, 2264, 50.00, 100,
```

- 打开 lab\_02\_18.sql 文件。观察用于创建 library\_items\_ext 外部表的代码片段。将 <TODO1>、<TODO2>、<TODO3> 和 <TODO4> 替换相应值后将该文件保存为 lab\_02\_18\_soln.sql。然后运行该脚本创建外部表。
- 在 library\_items\_ext 表中进行查询。

|   | CATEGOR... | BOO... | BOOK_P... | QUAN... |
|---|------------|--------|-----------|---------|
| 1 | 2354       | 2264   | 13.21     | 150     |
| 2 | 2355       | 2289   | 46.23     | 200     |
| 3 | 2355       | 2264   | 50        | 100     |

19. HR 部门需要一个包含所有部门地址的报表。使用 ORACLE\_DATAPUMP 访问驱动程序来创建外部表 dept\_add\_ext。该报表的输出应显示位置 ID、街道地址、城市、州或省以及国家/地区。使用 NATURAL JOIN 生成结果。

注：已经为本练习创建 emp\_dir 目录。

- 打开 lab\_02\_19.sql 文件。观察用于创建 dept\_add\_ext 外部表的代码片段。然后，将 <TODO1>、<TODO2> 和 <TODO3> 替换为相应的代码。将 <oraxx\_emp4.exp> 和 <oraxx\_emp5.exp> 替换为相应的文件名称。例如，如果您是 ora21 用户，则您的文件名为 ora21\_emp4.exp 和 ora21\_emp5.exp。将该脚本另存为 lab\_02\_19\_soln.sql。
- 运行 lab\_02\_19\_soln.sql 脚本来创建外部表。

## 练习 2-1：管理方案对象（续）

- c. 在 dept\_add\_ext 表中进行查询。

| LOCAT... | STREET_ADDRESS          | CITY                | STATE_PROVINCE   | COUNTRY_NAME          |
|----------|-------------------------|---------------------|------------------|-----------------------|
| 1 1000   | 1297 Via Cola di Rie    | Roma                | (null)           | Italy                 |
| 2 1100   | 93091 Calle della Testa | Venice              | (null)           | Italy                 |
| 3 1200   | 2017 Shinjuku-ku        | Tokyo               | Tokyo Prefecture | Japan                 |
| 4 1300   | 9450 Kamiya-cho         | Hiroshima           | (null)           | Japan                 |
| 5 1400   | 2014 Jabberwocky Rd     | Southlake           | Texas            | United States of Amer |
| 6 1500   | 2011 Interiors Blvd     | South San Francisco | California       | United States of Amer |
| 7 1600   | 2007 Zagora St          | South Brunswick     | New Jersey       | United States of Amer |
| 8 1700   | 2004 Charade Rd         | Seattle             | Washington       | United States of Amer |

注：在执行前面的步骤时，在默认目录 emp\_dir 下创建了两个文件 oraxx\_emp4.exp 和 oraxx\_emp5.exp。

20. 创建 emp\_books 表并在其中填充数据。将主键设置为延迟，然后观察在事务处理结束时会发生什么情况。

- a. 运行 lab\_02\_20\_a.sql 文件来创建 emp\_books 表。此时观察到 emp\_books\_pk 主键在创建时没有指定为可延迟。

```
create table succeeded.
```

- b. 运行 lab\_02\_20\_b.sql 文件，在 emp\_books 表中填充数据。此时会看到什么结果？

```
1 rows inserted

Error starting at line 2 in command:
insert into emp_books values(300,'Change Management')
Error report:
SQL Error: ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated
00001. 00000 - "unique constraint (%s.%s) violated"
*Cause: An UPDATE or INSERT statement attempted to insert a duplicate key.
For Trusted Oracle configured in DBMS MAC mode, you may see
this message if a duplicate entry exists at a different level.
*Action: Either remove the unique restriction or do not insert the key.
```

- c. 将 emp\_books\_pk 约束条件设置为延迟。此时会看到什么结果？

```
Error starting at line 1 in command:
set constraint emp_books_pk deferred
Error report:
SQL Error: ORA-02447: cannot defer a constraint that is not deferrable
02447. 00000 - "cannot defer a constraint that is not deferrable"
*Cause: An attempt was made to defer a nondeferrable constraint
*Action: Drop the constraint and create a new one that is deferrable
```

## 练习 2-1：管理方案对象（续）

- d. 删除 emp\_books\_pk 约束条件。

```
alter table emp_books succeeded.
```

- e. 修改 emp\_books 表定义，添加 emp\_books\_pk 约束条件并将其指定为可延迟。

```
alter table emp_books succeeded.
```

- f. 将 emp\_books\_pk 约束条件设置为延迟。

```
set constraint succeeded.
```

- g. 运行 lab\_02\_20\_g.sql 文件，在 emp\_books 表中填充数据。此时会看到什么结果？

```
1 rows inserted
1 rows inserted
1 rows inserted
```

- h. 提交事务处理。此时会看到什么结果？

```
Error report:
SQL Error: ORA-02091: transaction rolled back
ORA-00001: unique constraint (ORA21.EMP_BOOKS_PK) violated
02091. 00000 - "transaction rolled back"
*Cause: Also see error 2092. If the transaction is aborted at a remote
site then you will only see 2091; if aborted at host then you will
see 2092 and 2091.
*Action: Add rollback segment and retry the transaction.
```

## 练习解答 2-1：管理方案对象

1. 根据下面的表实例图表创建 DEPT2 表。在 SQL 工作表中输入语法。然后，执行该语句来创建表。确认该表已创建。

|       |        |          |
|-------|--------|----------|
| 列名    | ID     | NAME     |
| 键类型   |        |          |
| 空值/唯一 |        |          |
| FK 表  |        |          |
| FK 列  |        |          |
| 数据类型  | NUMBER | VARCHAR2 |
| 长度    | 7      | 25       |

```
CREATE TABLE dept2
 (id NUMBER(7),
 name VARCHAR2(25));

DESCRIBE dept2
```

2. 使用 DEPARTMENTS 表中的数据填充 DEPT2 表。仅包括所需的列。

```
INSERT INTO dept2
SELECT department_id, department_name
FROM departments;
```

3. 根据下面的表实例图表创建 EMP2 表。在 SQL 工作表中输入语法。然后，执行该语句来创建表。确认该表已创建。

|       |        |           |            |         |
|-------|--------|-----------|------------|---------|
| 列名    | ID     | LAST_NAME | FIRST_NAME | DEPT_ID |
| 键类型   |        |           |            |         |
| 空值/唯一 |        |           |            |         |
| FK 表  |        |           |            |         |
| FK 列  |        |           |            |         |
| 数据类型  | NUMBER | VARCHAR2  | VARCHAR2   | NUMBER  |
| 长度    | 7      | 25        | 25         | 7       |

```
CREATE TABLE emp2
(id NUMBER(7),
 last_name VARCHAR2(25),
 first_name VARCHAR2(25),
 dept_id NUMBER(7));

DESCRIBE emp2
```

## 练习解答 2-1：管理方案对象（续）

4. 修改 EMP2 表，允许输入更长的雇员姓氏。确认修改。

```
ALTER TABLE emp2
MODIFY (last_name VARCHAR2(50));

DESCRIBE emp2
```

5. 根据 EMPLOYEES 表的结构创建 EMPLOYEES2 表。只包括 EMPLOYEE\_ID、FIRST\_NAME、LAST\_NAME、SALARY 和 DEPARTMENT\_ID 列。将新表中的各列分别命名为 ID、FIRST\_NAME、LAST\_NAME、SALARY 和 DEPT\_ID。

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
 department_id dept_id
 FROM employees;
```

6. 删除 EMP2 表。

```
DROP TABLE emp2;
```

7. 在回收站中进行查询，查看该表是否存在。

```
SELECT original_name, operation, droptime
 FROM recyclebin;
```

8. 将 EMP2 表还原到执行 DROP 语句之前的状态。

```
FLASHBACK TABLE emp2 TO BEFORE DROP;
DESC emp2;
```

9. 从 EMPLOYEES2 表中删除 FIRST\_NAME 列。通过检查该表的描述来确认修改。

```
ALTER TABLE employees2
DROP COLUMN first_name;

DESCRIBE employees2
```

10. 在 EMPLOYEES2 表中，将 DEPT\_ID 列标记为 UNUSED。通过检查该表的描述来确认修改。

```
ALTER TABLE employees2
SET UNUSED (dept_id);

DESCRIBE employees2
```

## 练习解答 2-1：管理方案对象（续）

11. 从 EMPLOYEES2 表中删除所有 UNUSED 列。通过检查该表的描述来确认修改。

```
ALTER TABLE employees2
 DROP UNUSED COLUMNS;

DESCRIBE employees2
```

12. 在 EMP2 表的 ID 列上添加表级别 PRIMARY KEY 约束条件。应在创建时命名该约束条件。将约束条件命名为 my\_emp\_id\_pk。

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

13. 使用 ID 列为 DEPT2 表创建 PRIMARY KEY 约束条件。应在创建时命名该约束条件。将该约束条件命名为 my\_dept\_id\_pk。

```
ALTER TABLE dept2
ADD CONSTRAINT my_dept_id_pk PRIMARY KEY(id);
```

14. 在 EMP2 表上添加外键引用，确保雇员不会被分配到不存在的部门。将该约束条件命名为 my\_emp\_dept\_id\_fk。

```
ALTER TABLE emp2
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept2(id);
```

15. 修改 EMP2 表。添加数据类型为 NUMBER (精度为 2, 小数位数为 2) 的 COMMISSION 列。在 COMMISSION 列中添加约束条件，确保佣金值大于零。

```
ALTER TABLE emp2
ADD commission NUMBER(2,2)
CONSTRAINT my_emp_comm_ck CHECK (commission > 0);
```

16. 删除 EMP2 和 DEPT2 表，使其无法还原。检查回收站。

```
DROP TABLE emp2 PURGE;
DROP TABLE dept2 PURGE;

SELECT original_name, operation, droptime
 FROM recyclebin;
```

## 练习解答 2-1：管理方案对象（续）

17. 根据下面的表实例图表创建 DEPT\_NAMED\_INDEX 表。将 PRIMARY KEY 列的索引命名为 DEPT\_PK\_IDX。

| 列名   | Deptno | Dname    |
|------|--------|----------|
| 主键   | Yes    |          |
| 数据类型 | Number | VARCHAR2 |
| 长度   | 4      | 30       |

```
CREATE TABLE DEPT_NAMED_INDEX
(deptno NUMBER(4)
PRIMARY KEY USING INDEX
(CREATE INDEX dept_pk_idx ON
DEPT_NAMED_INDEX(deptno)),
dname VARCHAR2(30));
```

18. 创建外部表 library\_items\_ext。使用 ORACLE\_LOADER 访问驱动程序。

注：已经为此练习创建 emp\_dir 目录和 library\_items.dat。

library\_items.dat 中记录的格式如下：

2354, 2264, 13.21, 150,

2355, 2289, 46.23, 200,

2355, 2264, 50.00, 100,

- a) 打开 lab\_02\_18.sql 文件。观察用于创建 library\_items\_ext 外部表的代码片段。将 <TODO1>、<TODO2>、<TODO3> 和 <TODO4> 替换相应值后将该文件保存为 lab\_02\_18\_soln.sql。然后运行该脚本创建外部表。

```
CREATE TABLE library_items_ext (category_id number(12)
 , book_id number(6)
 , book_price number(8,2)
 , quantity number(8)
)
ORGANIZATION EXTERNAL
 (TYPE ORACLE_LOADER
 DEFAULT DIRECTORY emp_dir
 ACCESS PARAMETERS (RECORDS DELIMITED BY NEWLINE
 FIELDS TERMINATED BY ',')
 LOCATION ('library_items.dat')
)
REJECT LIMIT UNLIMITED;
```

- b) 在 library\_items\_ext 表中进行查询。

```
SELECT * FROM library_items_ext;
```

## 练习解答 2-1：管理方案对象（续）

19. HR 部门需要一个包含所有部门地址的报表。使用 ORACLE\_DATAPUMP 访问驱动程序来创建外部表 dept\_add\_ext。该报表的输出应显示位置 ID、街道地址、城市、州或省以及国家/地区。使用 NATURAL JOIN 生成结果。

注：已经为本练习创建 emp\_dir 目录。

- a) 打开 lab\_02\_19.sql 文件。观察用于创建 dept\_add\_ext 外部表的代码片段。然后，将 <TODO1>、<TODO2> 和 <TODO3> 替换为相应的代码。将 <oraxx\_emp4.exp> 和 <oraxx\_emp5.exp> 替换为相应的文件名称。例如，如果您是用户 ora21，则您的文件名为 ora21\_emp4.exp 和 ora21\_emp5.exp。将该脚本另存为 lab\_02\_19\_soln.sql。

```
CREATE TABLE dept_add_ext (location_id,
 street_address, city,
 state_province,
 country_name)
ORGANIZATION EXTERNAL (
TYPE ORACLE_DATAPUMP
DEFAULT DIRECTORY emp_dir
LOCATION ('oraxx_emp4.exp', 'oraxx_emp5.exp'))
PARALLEL
AS
SELECT location_id, street_address, city, state_province,
country_name
FROM locations
NATURAL JOIN countries;
```

注：在执行前面的步骤时，在默认目录 emp\_dir 下创建了两个文件 **oraxx\_emp4.exp** 和 **oraxx\_emp5.exp**。

运行 lab\_02\_19\_soln.sql 脚本来创建外部表。

- b) 在 dept\_add\_ext 表中进行查询。

```
SELECT * FROM dept_add_ext;
```

20. 创建 emp\_books 表并在其中填充数据。将主键设置为延迟，然后观察在事务处理结束时会发生什么情况。

- a) 运行 lab\_02\_20a.sql 文件以创建 emp\_books 表。此时观察到 emp\_books\_pk 主键在创建时没有指定为可延迟。

```
CREATE TABLE emp_books (book_id number,
 title varchar2(20), CONSTRAINT
emp_books_pk PRIMARY KEY (book_id));
```

## 练习解答 2-1：管理方案对象（续）

b) 运行 lab\_02\_20b.sql 脚本将数据填充在 emp\_books 表中。

此时会看到什么结果？

```
INSERT INTO emp_books VALUES(300,'Organizations');
INSERT INTO emp_books VALUES(300,'Change Management');
```

插入了第一行。但是，插入第二行时，您看到了 ora-00001 错误。

c) 将 emp\_books\_pk 约束条件设置为延迟。此时会看到什么结果？

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

您将看到以下错误：“ORA-02447: Cannot defer a constraint that is not deferrable（不能延迟不可延迟的约束条件）”。

d) 删除 emp\_books\_pk 约束条件。

```
ALTER TABLE emp_books DROP CONSTRAINT emp_books_pk;
```

e) 修改 emp\_books 表定义；这次添加 emp\_books\_pk 约束条件并指定为可延迟。

```
ALTER TABLE emp_books ADD (CONSTRAINT emp_books_pk PRIMARY KEY
(book_id) DEFERRABLE);
```

f) 将 emp\_books\_pk 约束条件设置为延迟。

```
SET CONSTRAINT emp_books_pk DEFERRED;
```

g) 运行 lab\_02\_20g.sql 文件将数据填充在 emp\_books 表中。

此时会看到什么结果？

```
INSERT INTO emp_books VALUES (300,'Change Management');
INSERT INTO emp_books VALUES (300,'Personality');
INSERT INTO emp_books VALUES (350,'Creativity');
```

您看到插入了所有行。

h) 提交事务处理。此时会看到什么结果？

```
COMMIT;
```

您看到事务处理被回退。

## 第 3 课的练习和解答

### 练习 3-1：使用数据字典视图管理对象

在本练习中，将通过在字典视图中进行查询来查找关于方案对象的信息。

1. 在 USER\_TABLES 数据字典视图中进行查询，查看关于您所拥有的表的信息。

| TABLE_NAME         |
|--------------------|
| 1 REGIONS          |
| 2 LOCATIONS        |
| 3 DEPARTMENTS      |
| 4 JOBS             |
| 5 EMPLOYEES        |
| 6 JOB_HISTORY      |
| 7 EMP_NEW_SAL      |
| 8 EMPLOYEES2       |
| 9 DEPT_NAMED_INDEX |

...

2. 在 ALL\_TABLES 数据字典视图中进行查询，查看关于您可访问的所有表的信息。但不包括您自己的表。

注：您的列表可能与以下列表不完全一样：

| TABLE_NAME             | OWNER |
|------------------------|-------|
| 1 DUAL                 | SYS   |
| 2 SYSTEM_PRIVILEGE_MAP | SYS   |
| 3 TABLE_PRIVILEGE_MAP  | SYS   |

...

|                            |     |
|----------------------------|-----|
| 98 PLAN_TABLE\$            | SYS |
| 99 WRI\$_ADV_ASA_RECO_DATA | SYS |
| 100 PSTUBTBL               | SYS |

3. 创建一个脚本来报告指定表的列名、数据类型和数据类型长度以及是否允许为空。提示用户输入表名。为 DATA\_PRECISION 列和 DATA\_SCALE 列提供相应的别名。将此脚本保存在名为 lab\_03\_01.sql 的文件中。

例如，如果用户输入 DEPARTMENTS，则输出结果如下所示：

| COLUMN_NAME       | DATA_TYPE | DATA_LENGTH | PRECISION | SCALE  | NULLABLE |
|-------------------|-----------|-------------|-----------|--------|----------|
| 1 DEPARTMENT_ID   | NUMBER    | 22          | 4         | 0      | N        |
| 2 DEPARTMENT_NAME | VARCHAR2  | 30          | (null)    | (null) | N        |
| 3 MANAGER_ID      | NUMBER    | 22          | 6         | 0      | Y        |
| 4 LOCATION_ID     | NUMBER    | 22          | 4         | 0      | Y        |

### 练习 3-1：使用数据字典视图管理对象（续）

4. 创建一个脚本来报告指定表的列名、约束条件名、约束条件类型、搜索条件以及状态。必须将 USER\_CONSTRAINTS 表和 USER\_CONS\_COLUMNS 表联接在一起，才能获得以上所有信息。提示用户输入表名。将此脚本保存在名为 lab\_03\_04.sql 的文件中。

例如，如果用户输入 DEPARTMENTS，则输出结果如下所示：

| COLUMN_NAME       | CONSTRAINT_NAME | CONSTRA... | SEARCH_CONDITION             | STATUS  |
|-------------------|-----------------|------------|------------------------------|---------|
| 1 DEPARTMENT_NAME | DEPT_NAME_NN    | C          | "DEPARTMENT_NAME" IS NOT ... | ENABLED |
| 2 DEPARTMENT_ID   | DEPT_ID_PK      | P          | (null)                       | ENABLED |
| 3 LOCATION_ID     | DEPT_LOC_FK     | R          | (null)                       | ENABLED |
| 4 MANAGER_ID      | DEPT_MGR_FK     | R          | (null)                       | ENABLED |

5. 在 DEPARTMENTS 表中添加注释。然后在 USER\_TAB\_COMMENTS 视图中进行查询，验证注释是否存在。

| COMMENTS                                                             |
|----------------------------------------------------------------------|
| 1 Company department information including name, code, and location. |

6. 为 EMPLOYEES 表创建同义词。将其命名为 EMP。然后查找您方案中所有同义词的名称。

| SYNONYM_NAME | TABLE_OWNER | TABLE_NAME  | DB_LINK |
|--------------|-------------|-------------|---------|
| 1 TEAM2      | ORA22       | DEPARTMENTS | (null)  |
| 2 EMP        | ORA21       | EMPLOYEES   | (null)  |

7. 运行 lab\_03\_07.sql 来创建本练习使用的 dept50 视图。您需要确定方案中所有视图的名称和定义。创建一个报表来检索以下视图信息：USER\_VIEWS 数据字典视图中的视图名和文本。

注：创建的 EMP\_DETAILS\_VIEW 是方案中的一部分。

注：在 SQL Developer 中使用“Run Script（运行脚本）”（或按 F5）可查看视图的完整定义。在 SQL Developer 中使用“Execute Statement（执行语句）”（或按 F9）可在结果窗格中水平滚动。如果使用 SQL\*Plus，要查看 LONG 列的更多内容，请使用命令 SET LONG n，其中 n 是要查看的 LONG 列的字符数值。

| VIEW_NAME          | TEXT                                                                          |
|--------------------|-------------------------------------------------------------------------------|
| 1 DEPT50           | SELECT employee_id empno, last_name employee, department_id deptno            |
| 2 EMP_DETAILS_VIEW | SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, |

### 练习 3-1：使用数据字典视图管理对象（续）

8. 查找序列的名称。编写一个查询脚本来显示关于序列的以下信息：序列名、最大值、增量大小和最后一个编号。将该脚本命名为 lab\_03\_08.sql。运行该脚本中的语句。

| SEQUENCE_NAME     | MAX_VALUE                                | INCREMENT_BY | LAST_NUMBER |
|-------------------|------------------------------------------|--------------|-------------|
| 1 DEPARTMENTS_SEQ | 9990                                     | 10           | 280         |
| 2 EMPLOYEES_SEQ   | 9999999999999999999999999999999999999999 | 1            | 207         |
| 3 LOCATIONS_SEQ   | 9900                                     | 100          | 3300        |

开始练习 9 至 11 之前，请运行 lab\_03\_09\_tab.sql 脚本。或者，打开该脚本文件，复制代码并将其粘贴在 SQL 工作表中。然后执行该脚本。该脚本：

- 如果表 DEPT2 和 EMP2 已存在，则不执行脚本命令
- 创建 DEPT2 和 EMP2 两个表

注：在练习 2 中，您应该已经删除 DEPT2 和 EMP2 两个表，因此无法还原它们。

9. 确认 DEPT2 和 EMP2 表都存储在数据字典中。

| TABLE_NAME |
|------------|
| 1 DEPT2    |
| 2 EMP2     |

10. 在 USER\_CONSTRAINTS 视图中进行查询，确认已添加这些约束条件。注意约束条件的类型和名称。

| CONSTRAINT_NAME     | CONSTRAINT_TYPE |
|---------------------|-----------------|
| 1 MY_DEPT_ID_PK     | P               |
| 2 MY_EMP_ID_PK      | P               |
| 3 MY_EMP_DEPT_ID_FK | R               |

11. 显示 USER\_OBJECTS 数据字典视图中 EMP2 和 DEPT2 表的对象名称和类型。

12. 根据下面的实例图表创建 SALES\_DEPT 表。将 PRIMARY KEY 列的索引命名为 SALES\_PK\_IDX。然后在数据字典中进行查询，查找索引名、表名并确认索引是否唯一。

|      |         |          |
|------|---------|----------|
| 列名   | Team_Id | Location |
| 主键   | Yes     |          |
| 数据类型 | Number  | VARCHAR2 |
| 长度   | 3       | 30       |

| INDEX_NAME     | TABLE_NAME | UNIQUENESS |
|----------------|------------|------------|
| 1 SALES_PK_IDX | SALES_DEPT | NONUNIQUE  |

## 练习解答 3-1：使用数据字典视图管理对象

- 在数据字典中查询，查看有关您所拥有的表的信息。

```
SELECT table_name
 FROM user_tables;
```

- 在数据字典视图中查询，查看有关您可以访问的所有表的信息。但不包括您自己的表。

```
SELECT table_name, owner
 FROM all_tables
 WHERE owner <>'ORAXXX';
```

- 创建一个脚本来报告指定表的列名、数据类型和数据类型的长度以及是否允许空值。提示用户输入表名。为 DATA\_PRECISION 列和 DATA\_SCALE 列提供相应的别名。将此脚本保存在名为 lab\_03\_01.sql 的文件中。

```
SELECT column_name, data_type, data_length,
 data_precision PRECISION, data_scale SCALE, nullable
 FROM user_tab_columns
 WHERE table_name = UPPER('&tab_name');
```

要进行测试，请运行该脚本并输入 DEPARTMENTS 作为表名称。

- 创建一个脚本来报告指定表的列名、约束条件名、约束条件类型、搜索条件以及状态。必须将 USER\_CONSTRAINTS 表和 USER\_CONS\_COLUMNS 表联接在一起，才能获得以上所有信息。提示用户输入表名。将此脚本保存在名为 lab\_03\_04.sql 的文件中。

```
SELECT ucc.column_name, uc.constraint_name,
 uc.constraint_type,
 uc.search_condition, uc.status
 FROM user_constraints uc JOIN user_cons_columns ucc
 ON uc.table_name = ucc.table_name
 WHERE uc.constraint_name = ucc.constraint_name
 AND uc.table_name = UPPER('&tab_name');
```

要进行测试，请运行该脚本并输入 DEPARTMENTS 作为表名称。

- 在 DEPARTMENTS 表中添加注释。然后在 USER\_TAB\_COMMENTS 视图中进行查询，验证注释是否存在。

```
COMMENT ON TABLE departments IS
 'Company department information including name, code, and
location.';

SELECT COMMENTS
 FROM user_tab_comments
 WHERE table_name = 'DEPARTMENTS';
```

## 练习解答 3-1：使用数据字典视图管理对象（续）

6. 为 EMPLOYEES 表创建同义词。将其命名为 EMP。然后查找方案中所有同义词的名称。

```
CREATE SYNONYM emp FOR EMPLOYEES;
SELECT *
FROM user_synonyms;
```

7. 运行 lab\_03\_07.sql 来创建本练习使用的 dept50 视图。您需要确定方案中所有视图的名称和定义。创建一个报表来检索以下视图信息：USER\_VIEWS 数据字典视图中的视图名和文本。

**注：** 创建的 EMP\_DETAILS\_VIEW 是您方案中的一部分。

**注：** 在 SQL Developer 中使用“Run Script（运行脚本）”（或按 F5）可查看视图的完整定义。在 SQL Developer 中使用“Execute Statement（执行语句）”（或按 F9）可在结果窗格中水平滚动。如果使用 SQL\*Plus 查看 LONG 列的更多内容，请使用命令 SET LONG n，其中 n 是要查看的 LONG 列的字符数量值。

```
SELECT view_name, text
FROM user_views;
```

8. 查找序列的名称。编写一个查询脚本来显示关于序列的以下信息：序列名、最大值、增量大小和最后一个编号。将该脚本命名为 lab\_03\_08.sql。运行该脚本中的语句。

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

开始练习 9 至 11 之前，请运行 lab\_03\_09\_tab.sql 脚本。或者，打开该脚本文件，复制代码并将其粘贴在 SQL 工作表中。然后执行该脚本。该脚本：

- 删除 DEPT2 和 EMP2 两个表
- 创建 DEPT2 和 EMP2 两个表

**注：** 在练习 2 中，您应该已经删除 DEPT2 和 EMP2 两个表，因此无法还原它们。

9. 确认 DEPT2 和 EMP2 表都存储在数据字典中。

```
SELECT table_name
FROM user_tables
WHERE table_name IN ('DEPT2', 'EMP2');
```

10. 在数据字典中查询，查找这两个表的约束条件名称和类型。

```
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name IN ('EMP2', 'DEPT2');
```

### 练习解答 3-1：使用数据字典视图管理对象（续）

11. 在数据字典中查询，显示这两个表的对象名称和类型。

```
SELECT object_name, object_type
FROM user_objects
WHERE object_name LIKE 'EMP%'
OR object_name LIKE 'DEPT%';
```

12. 根据下面的实例图表创建 SALES\_DEPT 表。将 PRIMARY KEY 列的索引命名为 SALES\_PK\_IDX。然后在数据字典中进行查询，查找索引名、表名并确认索引是否唯一。

| 列名   | Team_Id | Location |
|------|---------|----------|
| 主键   | Yes     |          |
| 数据类型 | Number  | VARCHAR2 |
| 长度   | 3       | 30       |

```
CREATE TABLE SALES_DEPT
 (team_id NUMBER(3)
 PRIMARY KEY USING INDEX
 (CREATE INDEX sales_pk_idx ON
 SALES_DEPT(team_id)),
 location VARCHAR2(30));

SELECT INDEX_NAME, TABLE_NAME, UNIQUENESS
FROM USER_INDEXES
WHERE TABLE_NAME = 'SALES_DEPT';
```

**第 4 课的练习和解答****练习 4-1：处理大型数据集**

在本练习中，您将执行多表 INSERT 和 MERGE 操作，并跟踪行版本。

1. 运行练习文件夹中的 lab\_04\_01.sql 脚本来创建 SAL\_HISTORY 表。
2. 显示 SAL\_HISTORY 表的结构。

| Name            | Null | Type        |
|-----------------|------|-------------|
| EMPLOYEE_ID     |      | NUMBER(6)   |
| HIRE_DATE       |      | DATE        |
| SALARY          |      | NUMBER(8,2) |
| 3 rows selected |      |             |

3. 运行练习文件夹中的 lab\_04\_03.sql 脚本来创建 MGR\_HISTORY 表。
4. 显示 MGR\_HISTORY 表的结构。

| Name            | Null | Type        |
|-----------------|------|-------------|
| EMPLOYEE_ID     |      | NUMBER(6)   |
| MANAGER_ID      |      | NUMBER(6)   |
| SALARY          |      | NUMBER(8,2) |
| 3 rows selected |      |             |

5. 运行练习文件夹中的 lab\_04\_05.sql 脚本来创建 SPECIAL\_SAL 表。
6. 显示 SPECIAL\_SAL 表的结构。

| Name            | Null | Type        |
|-----------------|------|-------------|
| EMPLOYEE_ID     |      | NUMBER(6)   |
| SALARY          |      | NUMBER(8,2) |
| 2 rows selected |      |             |

7. a. 编写一个查询来执行下列任务：
  - 在 EMPLOYEES 表中检索雇员 ID 小于 125 的雇员的雇员 ID、受雇日期、薪金和经理 ID 之类的详细资料。
  - 当薪金大于 \$20,000 时，将雇员 ID 和薪金之类的详细资料插入在 SPECIAL\_SAL 表中。
  - 将雇员 ID、受雇日期和薪金之类的详细资料插入在 SAL\_HISTORY 表中。
  - 将雇员 ID、经理 ID 和薪金之类的详细资料插入在 MGR\_HISTORY 表中。

## 练习 4-1：处理大型数据集（续）

- b. 显示 SPECIAL\_SAL 表中的记录。

|   | EMPLOYEE_ID | SALARY |
|---|-------------|--------|
| 1 | 100         | 24000  |

- c. 显示 SAL\_HISTORY 表中的记录。

|   | EMPLOYEE_ID | HIRE_DATE | SALARY |
|---|-------------|-----------|--------|
| 1 | 101         | 21-SEP-89 | 17000  |
| 2 | 102         | 13-JAN-93 | 17000  |
| 3 | 103         | 03-JAN-90 | 9000   |
| 4 | 104         | 21-MAY-91 | 6000   |
| 5 | 105         | 25-JUN-97 | 4800   |
| 6 | 106         | 05-FEB-98 | 4800   |
| 7 | 107         | 07-FEB-99 | 4200   |

- d. 显示 MGR\_HISTORY 表中的记录。

|   | EMPLOYEE_ID | MANAGER_ID | SALARY |
|---|-------------|------------|--------|
| 1 | 101         | 100        | 17000  |
| 2 | 102         | 100        | 17000  |
| 3 | 103         | 102        | 9000   |
| 4 | 104         | 103        | 6000   |
| 5 | 105         | 103        | 4800   |
| 6 | 106         | 103        | 4800   |
| 7 | 107         | 103        | 4200   |

8.

- a. 运行练习文件夹中的 lab\_04\_08a.sql 脚本以创建 SALES\_WEEK\_DATA 表。
- b. 运行练习文件夹中的 lab\_04\_08b.sql 脚本将记录插入在 SALES\_WEEK\_DATA 表中。
- c. 显示 SALES\_WEEK\_DATA 表的结构。

| Name     | Null | Type         |
|----------|------|--------------|
| ID       |      | NUMBER(6)    |
| WEEK_ID  |      | NUMBER(2)    |
| QTY_MON  |      | NUMBER(8, 2) |
| QTY_TUE  |      | NUMBER(8, 2) |
| QTY_WED  |      | NUMBER(8, 2) |
| QTY_THUR |      | NUMBER(8, 2) |
| QTY_FRI  |      | NUMBER(8, 2) |

7 rows selected

## 练习 4-1：处理大型数据集（续）

- d. 显示 SALES\_WEEK\_DATA 表中的记录。

|   | ID  | WEEK_ID | QTY_MON | QTY_TUE | QTY_WED | QTY_THUR | QTY_FRI |
|---|-----|---------|---------|---------|---------|----------|---------|
| 1 | 200 | 6       | 2050    | 2200    | 1700    | 1200     | 3000    |

- e. 运行练习文件夹中的 lab\_04\_08\_e.sql 脚本来创建 EMP\_SALES\_INFO 表。  
f. 显示 EMP\_SALES\_INFO 表的结构。

| Name            | Null | Type        |
|-----------------|------|-------------|
| ID              |      | NUMBER(6)   |
| WEEK            |      | NUMBER(2)   |
| QTY_SALES       |      | NUMBER(8,2) |
| 3 rows selected |      |             |

- g. 编写一个查询来执行以下任务：
- 在 SALES\_WEEK\_DATA 表中检索雇员 ID、周 ID、周一销售额、周二销售额、周三销售额、周四销售额、周五销售额之类的详细资料。
  - 建立一种转换机制，将在 SALES\_WEEK\_DATA 表中检索到的每一条记录都转换为 EMP\_SALES\_INFO 表中的多条记录。  
**提示：** 使用转换 INSERT (pivoting INSERT) 语句。
- h. 显示 EMP\_SALES\_INFO 表中的记录。

|   | ID  | WEEK | QTY_SALES |
|---|-----|------|-----------|
| 1 | 200 | 6    | 2050      |
| 2 | 200 | 6    | 2200      |
| 3 | 200 | 6    | 1700      |
| 4 | 200 | 6    | 1200      |
| 5 | 200 | 6    | 3000      |

9. 您在名为 emp.data 的平面文件中存储了前任雇员的数据，现在希望将所有前任和现任雇员的姓名和电子邮件 ID 都存储在一个表中。为此，先使用 emp\_dir 目录中的 emp.dat 源文件创建名为 EMP\_DATA 的外部表。使用 lab\_04\_09.sql 脚本来完成此任务。

## 练习 4-1：处理大型数据集（续）

10. 接下来，运行 lab\_04\_10.sql 脚本来创建 EMP\_HIST 表。
- 将电子邮件列的大小增加至 45。
  - 将在上一个练习中创建的 EMP\_DATA 表中的数据与 EMP\_HIST 表中的数据合并在一起。假定外部 EMP\_DATA 表中的数据是最新的数据。如果 EMP\_DATA 表行与 EMP\_HIST 表相匹配，则更新 EMP\_HIST 表的电子邮件列以匹配 EMP\_DATA 表行。如果 EMP\_DATA 表行不匹配，则将其插入在 EMP\_HIST 表中。当雇员的姓和名都相同时，就认为这些行是匹配的。
  - 合并后在 EMP\_HIST 中检索行。

|    | FIRST_NAME | LAST_NAME | EMAIL    |
|----|------------|-----------|----------|
| 1  | Ellen      | Abel      | EABEL    |
| 2  | Sundar     | Ande      | SANDE    |
| 3  | Mozhe      | Atkinson  | MATKINSO |
| 4  | David      | Austin    | DAUSTIN  |
| 5  | Hermann    | Baer      | HBAER    |
| 6  | Shelli     | Baida     | SBAIDA   |
| 7  | Amit       | Banda     | ABANDA   |
| 8  | Elizabeth  | Bates     | EBATES   |
| 9  | Sarah      | Bell      | SBELL    |
| 10 | David      | Bernstein | DBERNSTE |
| 11 | Laura      | Bissot    | LBISSOT  |

11. 使用 lab\_04\_11.sql 脚本创建 EMP3 表。在 EMP3 表中，将 Kochhar 所在的部门更改为 60 并提交更改。接下来，将 Kochhar 所在的部门更改为 50 并提交更改。使用“行版本”功能跟踪 Kochhar 的更改。

|   | START_DATE                              | END_DATE                         | DEPARTMENT_ID |
|---|-----------------------------------------|----------------------------------|---------------|
| 1 | 18-JUN-09 06.04.26.0000000000 PM (null) |                                  | 50            |
| 2 | 18-JUN-09 06.04.26.0000000000 PM        | 18-JUN-09 06.04.26.0000000000 PM | 60            |
| 3 | (null)                                  | 18-JUN-09 06.04.26.0000000000 PM | 90            |

## 练习解答 4-1：处理大型数据集

1. 运行练习文件夹中的 lab\_04\_01.sql 脚本来创建 SAL\_HISTORY 表。
2. 显示 SAL\_HISTORY 表的结构。

DESC sal\_history

3. 运行练习文件夹中的 lab\_04\_03.sql 脚本来创建 MGR\_HISTORY 表。
4. 显示 MGR\_HISTORY 表的结构。

DESC mgr\_history

5. 运行练习文件夹中的 lab\_04\_05.sql 脚本来创建 SPECIAL\_SAL 表。
6. 显示 SPECIAL\_SAL 表的结构。

DESC special\_sal

7. a) 编写一个查询来完成下列任务：

- 在 EMPLOYEES 表中检索雇员 ID 小于 125 的雇员的雇员 ID、受雇日期、薪金和经理 ID 之类的详细资料。
- 当薪金大于 \$20,000 时，将雇员 ID 和薪金之类的详细资料插入在 SPECIAL\_SAL 表中。
- 将雇员 ID、受雇日期和薪金之类的详细资料插入在 SAL\_HISTORY 表中。
- 将雇员 ID、经理 ID 和薪金之类的详细资料插入在 MGR\_HISTORY 表中。

```

INSERT ALL
WHEN SAL > 20000 THEN
 INTO special_sal VALUES (EMPID, SAL)
ELSE
 INTO sal_history VALUES(EMPID, HIREDATE, SAL)
 INTO mgr_history VALUES(EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
 salary SAL, manager_id MGR
FROM employees
WHERE employee_id < 125;

```

- b) 显示 SPECIAL\_SAL 表中的记录。

SELECT \* FROM special\_sal;

- c) 显示 SAL\_HISTORY 表中的记录。

SELECT \* FROM sal\_history;

- d) 显示 MGR\_HISTORY 表中的记录。

SELECT \* FROM mgr\_history;

## 练习解答 4-1：处理大型数据集（续）

8. a) 运行练习文件夹中的 lab\_04\_08a.sql 脚本以创建 SALES\_WEEK\_DATA 表。
- b) 运行练习文件夹中的 lab\_04\_08b.sql 脚本将记录插入在 SALES\_WEEK\_DATA 表中。
- c) 显示 SALES\_WEEK\_DATA 表的结构。

```
DESC sales_week_data
```

- d) 显示 SALES\_WEEK\_DATA 表中的记录。

```
SELECT * FROM SALES_WEEK_DATA;
```

- e) 运行练习文件夹中的 lab\_04\_08\_e.sql 脚本来创建 EMP\_SALES\_INFO 表。
- f) 显示 EMP\_SALES\_INFO 表的结构。

```
DESC emp_sales_info
```

- g) 编写一个查询来执行下列任务：

- 在 SALES\_WEEK\_DATA 表中检索雇员 ID、周 ID、周一销售额、周二销售额、周三销售额、周四销售额、周五销售额之类的详细资料。
- 建立一种转换机制，将在 SALES\_WEEK\_DATA 表中检索到的每一条记录都转换为 EMP\_SALES\_INFO 表中的多条记录。

**提示：** 使用转换 INSERT (pivoting INSERT) 语句。

```
INSERT ALL
 INTO emp_sales_info VALUES (id, week_id, QTY_MON)
 INTO emp_sales_info VALUES (id, week_id, QTY_TUE)
 INTO emp_sales_info VALUES (id, week_id, QTY_WED)
 INTO emp_sales_info VALUES (id, week_id, QTY_THUR)
 INTO emp_sales_info VALUES (id, week_id, QTY_FRI)
 SELECT ID, week_id, QTY_MON, QTY_TUE, QTY_WED,
 QTY_THUR, QTY_FRI FROM sales_week_data;
```

- h) 显示 SALES\_INFO 表中的记录。

```
SELECT * FROM emp_sales_info;
```

## 练习解答 4-1：处理大型数据集（续）

9. 您在名为 emp.data 的平面文件中存储了前任雇员的数据，现在希望将所有前任和现任雇员的姓名和电子邮件 ID 都存储在一个表中。为此，先使用 emp\_dir 目录中的 emp.dat 源文件创建名为 EMP\_DATA 的外部表。可以使用 lab\_04\_09.sql 中的脚本来实现此操作。

```

CREATE TABLE emp_data
 (first_name VARCHAR2(20)
 ,last_name VARCHAR2(20)
 ,email VARCHAR2(30)
)
ORGANIZATION EXTERNAL
(
 TYPE oracle_loader
 DEFAULT DIRECTORY emp_dir
 ACCESS PARAMETERS
 (
 RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
 NOBADFILE
 NOLOGFILE
 FIELDS
 (first_name POSITION (1:20) CHAR
 , last_name POSITION (22:41) CHAR
 , email POSITION (43:72) CHAR
)
)
 LOCATION ('emp.dat')) ;

```

10. 接下来，运行 lab\_04\_10.sql 脚本来创建 EMP\_HIST 表。

- a) 将电子邮件列的大小增加至 45。

```
ALTER TABLE emp_hist MODIFY email varchar(45);
```

## 练习解答 4-1：处理大型数据集（续）

- b) 将在上一个练习中创建的 EMP\_DATA 表中的数据与 EMP\_HIST 表中的数据合并在一起。假定外部 EMP\_DATA 表中的数据是最新的数据。如果 EMP\_DATA 表行与 EMP\_HIST 表相匹配，则更新 EMP\_HIST 表的电子邮件列以匹配 EMP\_DATA 表行。如果 EMP\_DATA 表行不匹配，则将其插入在 EMP\_HIST 表中。当雇员的姓和名都相同时，就认为这些行是匹配的。

```
MERGE INTO EMP_HIST f USING EMP_DATA h
ON (f.first_name = h.first_name
AND f.last_name = h.last_name)
WHEN MATCHED THEN
 UPDATE SET f.email = h.email
WHEN NOT MATCHED THEN
 INSERT (f.first_name
 , f.last_name
 , f.email)
VALUES (h.first_name
 , h.last_name
 , h.email);
```

- c) 合并后在 EMP\_HIST 中检索行。

```
SELECT * FROM emp_hist;
```

11. 使用 lab\_04\_11.sql 脚本创建 EMP3 表。在 EMP3 表中，将 Kochhar 所在的部门更改为 60 并提交更改。接下来，将 Kochhar 所在的部门更改为 50 并提交更改。使用“行版本”功能跟踪对 Kochhar 的更改。

```
UPDATE emp3 SET department_id = 60
WHERE last_name = 'Kochhar';
COMMIT;
UPDATE emp3 SET department_id = 50
WHERE last_name = 'Kochhar';
COMMIT;
```

```
SELECT VERSIONS_STARTTIME "START_DATE",
 VERSIONS_ENDTIME "END_DATE", DEPARTMENT_ID
 FROM EMP3
 WHERE VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
 AND LAST_NAME = 'Kochhar';
```

## 第 5 课的练习和解答

### 练习 5-1：管理不同时区中的数据

在本练习中，您将显示时区偏移量、CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。还要设置时区和使用 EXTRACT 函数。

1. 对会话进行更改，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY HH24:MI:SS。
2. a. 编写几个查询来显示下列时区的时区偏移量 (TZ\_OFFSET)。

- 美国/太平洋新时间

|   |                             |
|---|-----------------------------|
|   | TZ_OFFSET('US/PACIFIC-NEW') |
| 1 | -07:00                      |

- 新加坡

|   |                        |
|---|------------------------|
|   | TZ_OFFSET('SINGAPORE') |
| 1 | +08:00                 |

- 埃及

|   |                    |
|---|--------------------|
|   | TZ_OFFSET('EGYPT') |
| 1 | +03:00             |

- b. 更改会话，将 TIME\_ZONE 参数值设置为美国/太平洋新时间的时区偏移量。
- c. 显示此会话的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。
- d. 更改会话，将 TIME\_ZONE 参数值设置为新加坡的时区偏移量。
- e. 显示此会话的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

注：输出可能有所不同，这取决于执行命令的日期。

| CURRENT_DATE           | CURRENT_TIMESTAMP               | LOCALTIMESTAMP                         |
|------------------------|---------------------------------|----------------------------------------|
| 1 23-JUN-2009 15:12:08 | 23-JUN-09 03.12.08.000000000 PM | +08:00 23-JUN-09 03.12.08.000000000 PM |

注：在前面的练习中，您会发现 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP 都与会话时区相关。

3. 编写一个查询来显示 DBTIMEZONE 和 SESSIONTIMEZONE。

| DBTIMEZONE | SESSIONTIMEZONE |
|------------|-----------------|
| 1 +00:00   | +08:00          |

## 练习 5-1：管理不同时区中的数据（续）

4. 编写一个查询来在 EMPLOYEES 表的 HIRE\_DATE 列中提取在部门 80 中工作的那些雇员的 YEAR。

|   | LAST_NAME | EXTRACT(YEARFROMHIRE_DATE) |
|---|-----------|----------------------------|
| 1 | Russell   | 1996                       |
| 2 | Partners  | 1997                       |
| 3 | Errazuriz | 1997                       |
| 4 | Cambrault | 1999                       |
| 5 | Zlotkey   | 2000                       |
| 6 | Tucker    | 1997                       |
| 7 | Bernstein | 1997                       |

5. 更改会话，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY。  
 6. 检查并运行 lab\_05\_06.sql 脚本，以创建并填充 SAMPLE\_DATES 表。

- a. 从表中选择数据后进行查看。

|   | DATE_COL    |
|---|-------------|
| 1 | 23-JUN-2009 |

- b. 修改 DATE\_COL 列的数据类型，将其更改为 TIMESTAMP。从表中选择数据后进行查看。

|   | DATE_COL                        |
|---|---------------------------------|
| 1 | 23-JUN-09 02.14.52.000000000 PM |

- c. 尝试修改 DATE\_COL 列的数据类型，将其更改为 TIMESTAMP WITH TIME ZONE。此时会发生什么情况？

7. 创建一个查询，在 EMPLOYEES 表中检索姓氏并计算复查状态。如果受雇年份是 1998 年，则复查状态显示为“Needs Review（需要复查）”，否则，显示为“not this year（非本年度）”。将复查状态列命名为“Review（复查）”。按 HIRE\_DATE 列对结果进行排序。

提示：使用包含 EXTRACT 函数的 CASE 表达式来计算复查状态。

|   | LAST_NAME | Review         |
|---|-----------|----------------|
| 1 | King      | not this year! |
| 2 | Whalen    | not this year! |
| 3 | Kochhar   | not this year! |
| 4 | Hunold    | not this year! |
| 5 | Ernst     | not this year! |
| 6 | De Haan   | not this year! |
| 7 | Mavris    | not this year! |

...

## 练习 5-1：管理不同时区中的数据（续）

8. 创建一个查询来显示每位雇员的姓氏和任职年数。如果雇员的任职年数为 5 年或更久，则显示“5 years of service (已任职 5 年)”。如果雇员的任职年数为 10 年或更久，则显示“10 years of service (已任职 10 年)”。如果雇员的任职年数为 15 年或更久，则显示“15 years of service (已任职 15 年)”。如果与这些条件都不匹配，则显示“maybe next year (可能为下一年)”。按 HIRE\_DATE 列对结果进行排序。请使用 EMPLOYEES 表。

**提示：** 使用 CASE 表达式和 TO\_YMINTERVAL。

|   | LAST_NAME | HIRE_DATE   | SYSDATE     | Awards              |
|---|-----------|-------------|-------------|---------------------|
| 1 | OConnell  | 21-JUN-1999 | 23-JUN-2009 | 10 years of service |
| 2 | Grant     | 13-JAN-2000 | 23-JUN-2009 | 5 years of service  |
| 3 | Whalen    | 17-SEP-1987 | 23-JUN-2009 | 15 years of service |
| 4 | Hartstein | 17-FEB-1996 | 23-JUN-2009 | 10 years of service |
| 5 | Fay       | 17-AUG-1997 | 23-JUN-2009 | 10 years of service |
| 6 | Mavris    | 07-JUN-1994 | 23-JUN-2009 | 15 years of service |

...

## 练习解答 5-1：管理不同时区中的数据

1. 对会话进行更改，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY HH24:MI:SS。

```
ALTER SESSION SET NLS_DATE_FORMAT =
'DD-MON-YYYY HH24:MI:SS';
```

2. a. 编写几个查询来显示下列时区的时区偏移量 (TZ\_OFFSET)：美国/太平洋新时间、新加坡和埃及。

美国/太平洋新时间

```
SELECT TZ_OFFSET ('US/Pacific-New') from dual;
```

新加坡

```
SELECT TZ_OFFSET ('Singapore') from dual;
```

埃及

```
SELECT TZ_OFFSET ('Egypt') from dual;
```

- b. 更改会话，将 TIME\_ZONE 参数值设置为美国/太平洋新时间的时区偏移量。

```
ALTER SESSION SET TIME_ZONE = '-7:00';
```

- c. 显示此会话的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**输出可能有所不同，这取决于执行命令的日期。

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

- d. 更改会话，将 TIME\_ZONE 参数值设置为新加坡的时区偏移量。

```
ALTER SESSION SET TIME_ZONE = '+8:00';
```

- e. 显示此会话的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**根据执行命令的日期，输出可能会有所不同。

```
SELECT CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

**注：**在前面的练习中，您会发现 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP 都与会话时区相关。

## 练习解答 5-1：管理不同时区中的数据（续）

3. 编写一个查询来显示 DBTIMEZONE 和 SESSIONTIMEZONE。

```
SELECT DBTIMEZONE, SESSIONTIMEZONE
FROM DUAL;
```

4. 编写一个查询来在 EMPLOYEES 表的 HIRE\_DATE 列中提取在部门 80 中工作的那些雇员的 YEAR。

```
SELECT last_name, EXTRACT (YEAR FROM HIRE_DATE)
FROM employees
WHERE department_id = 80;
```

5. 更改会话，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY。

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

6. 检查并运行 lab\_05\_06.sql 脚本，以创建并填充 SAMPLE\_DATES 表。

- a. 从表中选择数据后进行查看。

```
SELECT * FROM sample_dates;
```

- b. 修改 DATE\_COL 列的数据类型，将其更改为 TIMESTAMP。从表中选择数据后进行查看。

```
ALTER TABLE sample_dates MODIFY date_col TIMESTAMP;
SELECT * FROM sample_dates;
```

- c. 尝试修改 DATE\_COL 列的数据类型，将其更改为 TIMESTAMP WITH TIME ZONE。此时会发生什么情况？

```
ALTER TABLE sample_dates MODIFY date_col
TIMESTAMP WITH TIME ZONE;
```

您无法更改 DATE\_COL 列的数据类型，因为 Oracle Server 不允许使用 ALTER 语句将 TIMESTAMP 转换成 TIMESTAMP WITH TIMEZONE。

7. 创建一个查询，在 EMPLOYEES 表中检索姓氏并计算复查状态。如果受雇年份是 1998 年，则复查状态显示为“Needs Review（需要复查）”，否则，显示为“not this year（非本年度）”。将复查状态列命名为“Review（复查）”。按 HIRE\_DATE 列对结果进行排序。

**提示：** 使用包含 EXTRACT 函数的 CASE 表达式来计算复查状态。

```
SELECT e.last_name
 ,(CASE extract(year from e.hire_date)
 WHEN 1998 THEN 'Needs Review'
 ELSE 'not this year!'
 END) AS "Review"
FROM employees e
ORDER BY e.hire_date;
```

## 练习解答 5-1：管理不同时区中的数据（续）

8. 创建一个查询来显示每位雇员的姓氏和任职年数。如果雇员的服务年数为 5 年或更久，则显示“5 years of service (已服务了 5 年)”。如果雇员的服务年数为 10 年或更久，则显示“10 years of service (已服务了 10 年)”。如果雇员的服务年数为 15 年或更久，则显示“15 years of service (已服务了 15 年)”。如果与这些条件都不匹配，则显示“maybe next year (可能为下一年)”。按 HIRE\_DATE 列对结果进行排序。请使用 EMPLOYEES 表。

**提示：** 使用 CASE 表达式和 TO\_YMINTERVAL。

```
SELECT e.last_name, hire_date, sysdate,
 (CASE
 WHEN (sysdate -TO_YMINTERVAL('15-0'))>=
 hire_date THEN '15 years of service'
 WHEN (sysdate -TO_YMINTERVAL('10-0'))>= hire_date
 THEN '10 years of service'
 WHEN (sysdate - TO_YMINTERVAL('5-0'))>= hire_date
 THEN '5 years of service'
 ELSE 'maybe next year!'
 END) AS "Awards"
 FROM employees e;
```

## 第 6 课的练习和解答

### 练习 6-1：使用子查询检索数据

在此练习中，您需要编写多列子查询、相关子查询和标量子查询。还要通过编写 WITH 子句解决问题。

1. 编写一个查询来显示符合以下条件的所有雇员的姓氏、部门编号和薪金：这些雇员的部门编号和薪金与领取佣金的任何雇员的部门编号和薪金相匹配。

|   | LAST_NAME | DEPARTMENT_ID | SALARY |
|---|-----------|---------------|--------|
| 1 | Russell   | 80            | 14000  |
| 2 | Partners  | 80            | 13500  |
| 3 | Errazuriz | 80            | 12000  |

2. 显示符合以下条件的所有雇员的姓氏、部门名称和薪金：这些雇员的薪金和佣金与位置 ID 为 1700 的任何雇员的薪金和佣金相匹配。

|   | LAST_NAME | DEPARTMENT_NAME | SALARY |
|---|-----------|-----------------|--------|
| 1 | Whalen    | Administration  | 4400   |
| 2 | Higgins   | Accounting      | 12000  |
| 3 | Greenberg | Finance         | 12000  |
| 4 | Gietz     | Accounting      | 8300   |

3. 创建一个查询来显示其薪金和佣金等于 Kochhar 的薪金和佣金的所有雇员的姓氏、受雇日期和薪金。

注：请勿在结果集中显示 Kochhar。

|   | LAST_NAME | HIRE_DATE   | SALARY |
|---|-----------|-------------|--------|
| 1 | De Haan   | 13-JAN-1993 | 17000  |

4. 创建一个查询来显示其薪金高于所有销售经理 (JOB\_ID = 'SA\_MAN') 的雇员。按从高到低的顺序对结果进行排序。

|   | LAST_NAME | JOB_ID  | SALARY |
|---|-----------|---------|--------|
| 1 | King      | AD_PRES | 24000  |
| 2 | De Haan   | AD_VP   | 17000  |
| 3 | Kochhar   | AD_VP   | 17000  |

## 练习 6-1：使用子查询检索数据（续）

5. 显示所在城市的城市名首字母为 T 的雇员的雇员 ID、姓氏和部门 ID 之类的详细资料。

|   | EMPLOYEE_ID | LAST_NAME     | DEPARTMENT_ID |
|---|-------------|---------------|---------------|
| 1 |             | 202 Fay       | 20            |
| 2 |             | 201 Hartstein | 20            |

6. 编写一个查询来查找其薪金高于所在部门平均薪金的所有雇员。显示姓氏、薪金、部门 ID 和部门的平均薪金。按平均薪金排序，舍入到 2 位小数。对查询中检索的那些列，请使用别名，如示例输出中所示。

7. 查找主管以外的所有雇员。

a. 首先使用 NOT EXISTS 运算符执行此任务。

|   | LAST_NAME |
|---|-----------|
| 1 | Abel      |
| 2 | Ande      |
| 3 | Atkinson  |
| 4 | Austin    |
| 5 | Baer      |
| 6 | Baida     |

b. 使用 NOT IN 运算符能否完成此任务？如何完成此任务？或者为什么不能？

8 编写一个查询来显示其薪金低于所在部门平均薪金的雇员的姓氏。

|   | A<br>Z | LAST_NAME |
|---|--------|-----------|
| 1 |        | Chen      |
| 2 |        | Sciarra   |
| 3 |        | Urman     |
| 4 |        | Popp      |
| 5 |        | Khoo      |
| 6 |        | Baida     |

## 练习 6-1：使用子查询检索数据（续）

9. 编写一个查询来显示符合以下条件的雇员的姓氏：所在部门中有一位或多位同事的受雇日期比他晚，但薪金却比他高。

|   | LAST_NAME |
|---|-----------|
| 1 | Vargas    |
| 2 | Patel     |
| 3 | Olson     |
| 4 | Marlow    |
| 5 | Landry    |
| 6 | Perkins   |

10. 编写一个查询来显示所有雇员的雇员 ID、姓氏和部门名称。

注：在 SELECT 语句中使用标量子查询来检索部门名称。

|   | EMPLOYEE_ID | LAST_NAME | DEPARTMENT     |
|---|-------------|-----------|----------------|
| 1 | 205         | Higgins   | Accounting     |
| 2 | 206         | Gietz     | Accounting     |
| 3 | 200         | Whalen    | Administration |
| 4 | 100         | King      | Executive      |
| 5 | 101         | Kochhar   | Executive      |

...

|     |     |        |          |
|-----|-----|--------|----------|
| 105 | 196 | Walsh  | Shipping |
| 106 | 197 | Feeney | Shipping |
| 107 | 178 | Grant  | (null)   |

11. 编写一个查询来显示其薪金总成本高于整个公司薪金总成本 1/8 的部门名称。

请使用 WITH 子句编写此查询。将此查询命名为“SUMMARY”。

|   | DEPARTMENT_NAME | DEPT_TOTAL |
|---|-----------------|------------|
| 1 | Sales           | 304500     |
| 2 | Shipping        | 156400     |

## 练习解答 6-1：使用子查询检索数据

1. 编写一个查询来显示符合以下条件的所有雇员的姓氏、部门编号和薪金：这些雇员的部门编号和薪金与领取佣金的任何雇员的部门编号和薪金相匹配。

```
SELECT last_name, department_id, salary
FROM employees
WHERE (salary, department_id) IN
 (SELECT salary, department_id
 FROM employees
 WHERE commission_pct IS NOT NULL);
```

2. 显示符合以下条件的所有雇员的姓氏、部门名称和薪金：这些雇员的薪金和佣金与位置 ID 为 1700 中的任意雇员的薪金和佣金相匹配。

```
SELECT e.last_name, d.department_name, e.salary
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND (salary, NVL(commission_pct,0)) IN
 (SELECT salary, NVL(commission_pct,0)
 FROM employees e, departments d
 WHERE e.department_id = d.department_id
 AND d.location_id = 1700);
```

3. 创建一个查询来显示其薪金和佣金等于 Kochhar 的薪金和佣金的所有雇员的姓氏、受雇日期和薪金。

**注：**请勿在结果集中显示 Kochhar。

```
SELECT last_name, hire_date, salary
FROM employees
WHERE (salary, NVL(commission_pct,0)) IN
 (SELECT salary, NVL(commission_pct,0)
 FROM employees
 WHERE last_name = 'Kochhar')
AND last_name != 'Kochhar';
```

4. 创建一个查询来显示其薪金高于所有销售经理 (JOB\_ID = 'SA\_MAN') 的雇员。按薪金从高到低的顺序对结果进行排序。

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary > ALL
 (SELECT salary
 FROM employees
 WHERE job_id = 'SA_MAN')
ORDER BY salary DESC;
```

## 练习解答 6-1：使用子查询检索数据（续）

5. 显示所在城市的城市名首字母为 T 的雇员的雇员 ID、姓氏和部门 ID 之类的详细资料。

```
SELECT employee_id, last_name, department_id
 FROM employees
 WHERE department_id IN (SELECT department_id
 FROM departments
 WHERE location_id IN
 (SELECT location_id
 FROM locations
 WHERE city LIKE 'T%'));
```

6. 编写一个查询来查找其薪金高于所在部门平均薪金的所有雇员。显示姓氏、薪金、部门 ID 和部门的平均薪金。按平均薪金进行排序。对查询中检索的那些列，请使用别名，如示例输出中所示。

```
SELECT e.last_name ename, e.salary salary,
 e.department_id deptno, AVG(a.salary) dept_avg
 FROM employees e, employees a
 WHERE e.department_id = a.department_id
 AND e.salary > (SELECT AVG(salary)
 FROM employees
 WHERE department_id = e.department_id)
 GROUP BY e.last_name, e.salary, e.department_id
 ORDER BY AVG(a.salary);
```

7. 查找主管以外的所有雇员。

- a. 首先，使用 NOT EXISTS 运算符执行此任务。

```
SELECT outer.last_name
 FROM employees outer
 WHERE NOT EXISTS (SELECT 'X'
 FROM employees inner
 WHERE inner.manager_id =
 outer.employee_id);
```

- b. 是否可以使用 NOT IN 运算符完成此任务？如何完成此任务？或者为什么不能？

```
SELECT outer.last_name
 FROM employees outer
 WHERE outer.employee_id
 NOT IN (SELECT inner.manager_id
 FROM employees inner);
```

后一种解决方法不适用。子查询得到了一个 NULL 值，所以整个查询不会返回任何行。原因是所有与 NULL 值进行比较的条件都会产生一个 NULL 值。只要 NULL 值可能是值集的一部分，就不要使用 NOT IN 代替 NOT EXISTS。

## 练习解答 6-1：使用子查询检索数据（续）

8. 编写一个查询来显示其薪金低于所在部门平均薪金的雇员的姓氏。

```
SELECT last_name
 FROM employees outer
 WHERE outer.salary < (SELECT AVG(inner.salary)
 FROM employees inner
 WHERE inner.department_id
 = outer.department_id);
```

9. 编写一个查询来显示某些雇员的姓氏，这些雇员所在部门中有一个或多个同事的薪金高于该雇员的薪金，但其受雇日期较晚。

```
SELECT last_name
 FROM employees outer
 WHERE EXISTS (SELECT 'X'
 FROM employees inner
 WHERE inner.department_id =
 outer.department_id
 AND inner.hire_date > outer.hire_date
 AND inner.salary > outer.salary);
```

10. 编写一个查询来显示所有雇员的雇员 ID、姓氏和部门名称。

注：在 SELECT 语句中使用标量子查询来检索部门名称。

```
SELECT employee_id, last_name,
 (SELECT department_name
 FROM departments d
 WHERE e.department_id =
 d.department_id) department
 FROM employees e
 ORDER BY department;
```

11. 编写一个查询来显示其薪金总成本高于整个公司薪金总成本 1/8 的部门名称。

请使用 WITH 子句编写此查询。将此查询命名为“SUMMARY”。

```
WITH
summary AS (
 SELECT d.department_name, SUM(e.salary) AS dept_total
 FROM employees e, departments d
 WHERE e.department_id = d.department_id
 GROUP BY d.department_name)
SELECT department_name, dept_total
 FROM summary
 WHERE dept_total > (SELECT SUM(dept_total) * 1/8
 FROM summary)
 ORDER BY dept_total DESC;
```

## 第 7 课的练习和解答

### 练习 7-1：正则表达式支持功能

在本练习中，您将使用正则表达式函数搜索、替换和处理数据。还将创建一个新的 CONTACTS 表，并在 p\_number 列上添加一个 CHECK 约束条件来确保在数据库中输入特定标准格式的电话号码。

- 编写一个查询来搜索 EMPLOYEES 表中名字以“Ki”或“Ko”开头的所有雇员。

|   | FIRST_NAME | LAST_NAME |
|---|------------|-----------|
| 1 | Janette    | King      |
| 2 | Steven     | King      |
| 3 | Neena      | Kochhar   |

- 创建一个查询来删除 LOCATIONS 表的 STREET\_ADDRESS 列中显示的空格。  
使用“Street Address”作为列标题。

|   | Street Address          |
|---|-------------------------|
| 1 | 1297 Via Cola di Rie    |
| 2 | 93091 Calle della Testa |
| 3 | 2017 Shinjuku-ku        |
| 4 | 9450 Kamiya-cho         |
| 5 | 2014 Jabberwocky Rd     |
| 6 | 2011 Interiors Blvd     |
| 7 | 2007 Zagora St          |

- 创建一个查询来显示 LOCATIONS 表的 STREET\_ADDRESS 列中“St”已替换为“Street”的行。注意，不要替换其中已包含“Street”的任何行。只显示那些已替换的行。

|   | REGEXP_REPLACE(STREET_ADDRESS,'ST\$','STREET') |
|---|------------------------------------------------|
| 1 | 2007 Zagora Street                             |
| 2 | 6092 Boxwood Street                            |
| 3 | 12-98 Victoria Street                          |
| 4 | 8204 Arthur Street                             |

- 创建一个 contacts 表，在 p\_number 列上添加一个检查约束条件来强制使用以下格式掩码，从而确保在数据库中输入以下标准格式的电话号码：  
(XXX) XXX-XXXX。该表应包含以下列：

- l\_name varchar2(30)
- p\_number varchar2 (30)

### 练习 7-1：正则表达式支持功能（续）

5. 运行 SQL 脚本 lab\_07\_05.sql，将以下七个电话号码插入 contacts 表中。添加了哪些号码？

| l_name 列值 | p_number 列值       |
|-----------|-------------------|
| NULL      | '(650) 555-5555'  |
| NULL      | '(215) 555-3427'  |
| NULL      | '650 555-5555'    |
| NULL      | '650 555 5555'    |
| NULL      | '650-555-5555'    |
| NULL      | '(650)555-5555'   |
| NULL      | ' (650) 555-5555' |

6. 编写一个查询来查找 DNA 模式 ctc 在字符串 gtctcgtctcgttctgtctgtcgttctg 中出现的次数。忽略大小写。

| COUNT_DNA |
|-----------|
| 1 2       |

## 练习解答 7-1：正则表达式支持功能

1. 编写一个查询来搜索 EMPLOYEES 表中名字以“Ki”或“Ko”开头的所有雇员。

```
SELECT first_name, last_name
FROM employees
WHERE REGEXP_LIKE (last_name, '^K(i|o)\.');
```

2. 创建一个查询来删除 LOCATIONS 表的 STREET\_ADDRESS 列中显示的空格。使用“Street Address”作为列标题。

```
SELECT regexp_replace (street_address, ' ', '') AS "Street
Address"
FROM locations;
```

3. 创建一个查询来显示 LOCATIONS 表的 STREET\_ADDRESS 列中“St”被替换为“Street”的行。注意，不要替换其中已包含“Street”的任何行。只显示那些更改的行。

```
SELECT regexp_replace (street_address, 'St$', 'Street')
FROM locations
WHERE regexp_like (street_address, 'St');
```

4. 创建一个 contacts 表，在 p\_number 列上添加一个检查约束条件来强制使用以下格式掩码，从而确保在数据库中输入以下标准格式的电话号码：

**(XXX) XXX-XXXX**。该表应包含以下列：

- l\_name varchar2(30)
- p\_number varchar2 (30)

```
CREATE TABLE contacts
(
 l_name VARCHAR2(30),
 p_number VARCHAR2(30)
 CONSTRAINT p_number_format
 CHECK (REGEXP_LIKE (p_number, '^(\d{3}) (\d{3}-
\d{4}$')));
```

5. 运行 SQL 脚本 lab\_07\_05.sql，将以下七个电话号码插入在合同表中。添加了哪些号码？

只有前两条 **INSERT** 语句使用的格式符合 **c\_contacts\_pnf** 约束条件；其余语句产生了 **CHECK** 约束条件错误。

## 练习解答 7-1：正则表达式支持功能（续）

6. 编写一个查询来查找 DNA 模式 ctc 在字符串

gtctcgtctcggtctgtctgtcggtctg 中出现的次数。使用别名 Count\_DNA。  
忽略大小写。

```
SELECT REGEXP_COUNT('gtctcgtctcggtctgtctgtcggtctg', 'ctc')
AS Count_DNA
FROM dual;
```



# JB

表说明

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 方案说明

### 总体说明

在 Oracle DB 示例方案中描述了一家示例公司，该公司在全球范围内运营，履行多种不同产品的订单。该公司有三个部门：

- **人力资源部门：**跟踪关于雇员和雇员资质的信息
- **订单录入部门：**跟踪产品库存情况和各种销售渠道的销售情况
- **销售历史记录部门：**跟踪有助于做出业务决策的业务统计信息

上述每个部门都用一个方案来表示。在本课程中，您可以访问所有这些方案中的对象。但是，在示例、演示和练习中主要使用的是人力资源 (HR) 方案。

创建示例方案所必需的所有脚本都在 \$ORACLE\_HOME/demo/schema/ 文件夹中。

### 人力资源 (HR)

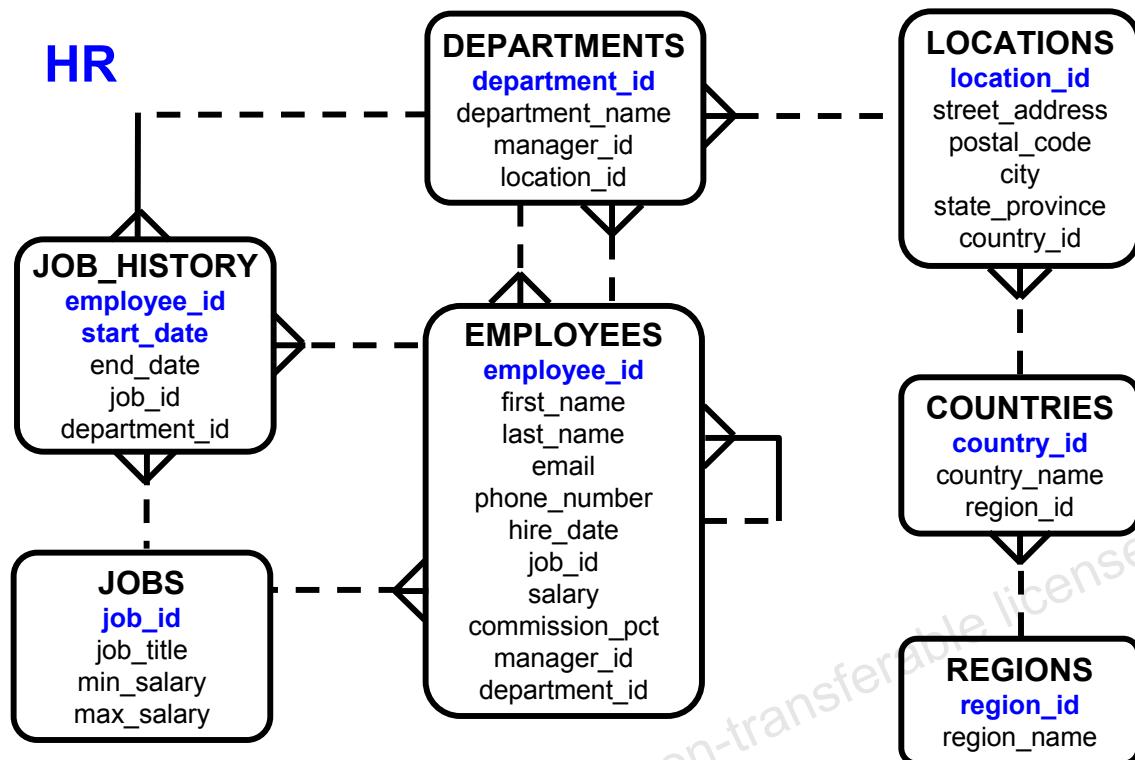
这是本课程中使用的方案。在人力资源 (HR) 记录中，每位雇员都有各自的标识号、电子邮件地址、职务标识代码、薪金和经理。某些雇员除了薪金还领取佣金。

公司还跟踪组织内职务信息。每个职务都有对应的标识代码、职位、最低和最高薪金范围。某些雇员在公司中已工作很长的时间，因此在公司内担任过不同的职位。当某一雇员辞职时，就会记录该雇员的工作期限、职务标识号和部门。

示例公司的经营地比较分散，所以需要跟踪公司仓库和公司部门的具体位置。每个雇员都分配到一个部门，每个部门不是用唯一部门编号来标识，就是用一个简称来标识。每个部门都与一个位置相关联，每个位置都有一个全称地址，其中包括街道名、邮政编码、城市、州/省，以及国家/地区代码。

公司会记录部门和仓库所在地的详细信息，如国家/地区名称、货币符号、货币名称以及国家/地区所在的地理区域。

## HR 实体关系图



## 人力资源 (HR) 表说明

DESCRIBE countries

| Name         | Null     | Type         |
|--------------|----------|--------------|
| COUNTRY_ID   | NOT NULL | CHAR(2)      |
| COUNTRY_NAME |          | VARCHAR2(40) |
| REGION_ID    |          | NUMBER       |

SELECT \* FROM countries;

|   | COUNTRY_ID | COUNTRY_NAME             | REGION_ID |
|---|------------|--------------------------|-----------|
| 1 | CA         | Canada                   | 2         |
| 2 | DE         | Germany                  | 1         |
| 3 | UK         | United Kingdom           | 1         |
| 4 | US         | United States of America | 2         |

## 人力资源 (HR) 表说明 (续)

DESCRIBE departments

| Name            | Null     | Type         |
|-----------------|----------|--------------|
| DEPARTMENT_ID   | NOT NULL | NUMBER(4)    |
| DEPARTMENT_NAME | NOT NULL | VARCHAR2(30) |
| MANAGER_ID      |          | NUMBER(6)    |
| LOCATION_ID     |          | NUMBER(4)    |

SELECT \* FROM departments;

| DEPARTMENT_ID | DEPARTMENT_NAME   | MANAGER_ID | LOCATION_ID |
|---------------|-------------------|------------|-------------|
| 1             | 10 Administration | 200        | 1700        |
| 2             | 20 Marketing      | 201        | 1800        |
| 3             | 50 Shipping       | 124        | 1500        |
| 4             | 60 IT             | 103        | 1400        |
| 5             | 80 Sales          | 149        | 2500        |
| 6             | 90 Executive      | 100        | 1700        |
| 7             | 110 Accounting    | 205        | 1700        |
| 8             | 190 Contracting   | (null)     | 1700        |

## 人力资源 (HR) 表说明 (续)

DESCRIBE employees

| Name           | Null     | Type         |
|----------------|----------|--------------|
| EMPLOYEE_ID    | NOT NULL | NUMBER(6)    |
| FIRST_NAME     |          | VARCHAR2(20) |
| LAST_NAME      | NOT NULL | VARCHAR2(25) |
| EMAIL          | NOT NULL | VARCHAR2(25) |
| PHONE_NUMBER   |          | VARCHAR2(20) |
| HIRE_DATE      | NOT NULL | DATE         |
| JOB_ID         | NOT NULL | VARCHAR2(10) |
| SALARY         |          | NUMBER(8,2)  |
| COMMISSION_PCT |          | NUMBER(2,2)  |
| MANAGER_ID     |          | NUMBER(6)    |
| DEPARTMENT_ID  |          | NUMBER(4)    |

SELECT \* FROM employees;

| #  | EMPLOYEE_ID | FIRST_N... | LAST_N... | EMAIL    | PHONE_NUMBER       | HIRE_DATE | JOB_ID    | SALARY | COMM... | MANAGER_ID | DEPARTMENT_ID |
|----|-------------|------------|-----------|----------|--------------------|-----------|-----------|--------|---------|------------|---------------|
| 1  | 100         | Steven     | King      | SKING    | 515.123.4567       | 17-JUN-87 | AD_PRES   | 24000  | (null)  | (null)     | 90            |
| 2  | 101         | Neena      | Kochhar   | NKOCHHAR | 515.123.4568       | 21-SEP-89 | AD_VP     | 17000  | (null)  | 100        | 90            |
| 3  | 102         | Lex        | De Haan   | LDEHAAN  | 515.123.4569       | 13-JAN-93 | AD_VP     | 17000  | (null)  | 100        | 90            |
| 4  | 103         | Alexander  | Hunold    | AHUNOLD  | 590.423.4567       | 03-JAN-90 | IT_PROG   | 9000   | (null)  | 102        | 60            |
| 5  | 104         | Bruce      | Ernst     | BERNST   | 590.423.4568       | 21-MAY-91 | IT_PROG   | 6000   | (null)  | 103        | 60            |
| 6  | 107         | Diana      | Lorentz   | DLORENTZ | 590.423.5567       | 07-FEB-99 | IT_PROG   | 4200   | (null)  | 103        | 60            |
| 7  | 124         | Kevin      | Mourgos   | KMOURGOS | 650.123.5234       | 16-NOV-99 | ST_MAN    | 5800   | (null)  | 100        | 50            |
| 8  | 141         | Trenna     | Rajs      | TRAJS    | 650.121.8009       | 17-OCT-95 | ST_CLERK  | 3500   | (null)  | 124        | 50            |
| 9  | 142         | Curtis     | Davies    | CDAVIES  | 650.121.2994       | 29-JAN-97 | ST_CLERK  | 3100   | (null)  | 124        | 50            |
| 10 | 143         | Randall    | Matos     | RMATOS   | 650.121.2874       | 15-MAR-98 | ST_CLERK  | 2600   | (null)  | 124        | 50            |
| 11 | 144         | Peter      | Vargas    | PVARGAS  | 650.121.2004       | 09-JUL-98 | ST_CLERK  | 2500   | (null)  | 124        | 50            |
| 12 | 149         | Eleni      | Zlotkey   | EZLOTKEY | 011.44.1344.429018 | 29-JAN-00 | SA_MAN    | 10500  | 0.2     | 100        | 80            |
| 13 | 174         | Ellen      | Abel      | EABEL    | 011.44.1644.429267 | 11-MAY-96 | SA REP    | 11000  | 0.3     | 149        | 80            |
| 14 | 176         | Jonathon   | Taylor    | JTAYLOR  | 011.44.1644.429265 | 24-MAR-98 | SA REP    | 8600   | 0.2     | 149        | 80            |
| 15 | 178         | Kimberely  | Grant     | KGRANT   | 011.44.1644.429263 | 24-MAY-99 | SA REP    | 7000   | 0.15    | 149        | (null)        |
| 16 | 200         | Jennifer   | Whalen    | JWHALEN  | 515.123.4444       | 17-SEP-87 | AD_ASST   | 4400   | (null)  | 101        | 10            |
| 17 | 201         | Michael    | Hartstein | MHARTSTE | 515.123.5555       | 17-FEB-96 | MK MAN    | 13000  | (null)  | 100        | 20            |
| 18 | 202         | Pat        | Fay       | PFAY     | 603.123.6666       | 17-AUG-97 | MK REP    | 6000   | (null)  | 201        | 20            |
| 19 | 205         | Shelley    | Higgins   | SHIGGINS | 515.123.8080       | 07-JUN-94 | AC_MGR    | 12000  | (null)  | 101        | 110           |
| 20 | 206         | William    | Gietz     | WGIEZ    | 515.123.8181       | 07-JUN-94 | AC ACC... | 8300   | (null)  | 205        | 110           |

## 人力资源 (HR) 表说明 (续)

DESCRIBE job\_history

| DESCRIBE job_history |          |              |
|----------------------|----------|--------------|
| Name                 | Null     | Type         |
| EMPLOYEE_ID          | NOT NULL | NUMBER(6)    |
| START_DATE           | NOT NULL | DATE         |
| END_DATE             | NOT NULL | DATE         |
| JOB_ID               | NOT NULL | VARCHAR2(10) |
| DEPARTMENT_ID        |          | NUMBER(4)    |

SELECT \* FROM job\_history

| #  | EMPLOYEE_ID | START_DATE | END_DATE  | #          | JOB_ID | # | DEPARTMENT_ID |
|----|-------------|------------|-----------|------------|--------|---|---------------|
| 1  | 102         | 13-JAN-93  | 24-JUL-98 | IT_PROG    |        |   | 60            |
| 2  | 101         | 21-SEP-89  | 27-OCT-93 | AC_ACCOUNT |        |   | 110           |
| 3  | 101         | 28-OCT-93  | 15-MAR-97 | AC_MGR     |        |   | 110           |
| 4  | 201         | 17-FEB-96  | 19-DEC-99 | MK_REP     |        |   | 20            |
| 5  | 114         | 24-MAR-98  | 31-DEC-99 | ST_CLERK   |        |   | 50            |
| 6  | 122         | 01-JAN-99  | 31-DEC-99 | ST_CLERK   |        |   | 50            |
| 7  | 200         | 17-SEP-87  | 17-JUN-93 | AD_ASST    |        |   | 90            |
| 8  | 176         | 24-MAR-98  | 31-DEC-98 | SA_REP     |        |   | 80            |
| 9  | 176         | 01-JAN-99  | 31-DEC-99 | SA_MAN     |        |   | 80            |
| 10 | 200         | 01-JUL-94  | 31-DEC-98 | AC_ACCOUNT |        |   | 90            |

## 人力资源 (HR) 表说明 (续)

DESCRIBE jobs

| Name       | Null     | Type         |
|------------|----------|--------------|
| JOB_ID     | NOT NULL | VARCHAR2(10) |
| JOB_TITLE  | NOT NULL | VARCHAR2(35) |
| MIN_SALARY |          | NUMBER(6)    |
| MAX_SALARY |          | NUMBER(6)    |

SELECT \* FROM jobs

| JOB_ID       | JOB_TITLE                     | MIN_SALARY | MAX_SALARY |
|--------------|-------------------------------|------------|------------|
| 1 AD_PRES    | President                     | 20000      | 40000      |
| 2 AD_VP      | Administration Vice President | 15000      | 30000      |
| 3 AD_ASST    | Administration Assistant      | 3000       | 6000       |
| 4 AC_MGR     | Accounting Manager            | 8200       | 16000      |
| 5 AC_ACCOUNT | Public Accountant             | 4200       | 9000       |
| 6 SA_MAN     | Sales Manager                 | 10000      | 20000      |
| 7 SA_REP     | Sales Representative          | 6000       | 12000      |
| 8 ST_MAN     | Stock Manager                 | 5500       | 8500       |
| 9 ST_CLERK   | Stock Clerk                   | 2000       | 5000       |
| 10 IT_PROG   | Programmer                    | 4000       | 10000      |
| 11 MK_MAN    | Marketing Manager             | 9000       | 15000      |
| 12 MK_REP    | Marketing Representative      | 4000       | 9000       |

## 人力资源 (HR) 表说明 (续)

DESCRIBE locations

| Name           | Null     | Type         |
|----------------|----------|--------------|
| LOCATION_ID    | NOT NULL | NUMBER(4)    |
| STREET_ADDRESS |          | VARCHAR2(40) |
| POSTAL_CODE    |          | VARCHAR2(12) |
| CITY           | NOT NULL | VARCHAR2(30) |
| STATE_PROVINCE |          | VARCHAR2(25) |
| COUNTRY_ID     |          | CHAR(2)      |

SELECT \* FROM locations

| # | LOCATION_ID | STREET_ADDRESS                           | POSTAL_CODE | CITY                | STATE_PROVINCE | COUNTRY_ID |
|---|-------------|------------------------------------------|-------------|---------------------|----------------|------------|
| 1 | 1400        | 2014 Jabberwocky Rd                      | 26192       | Southlake           | Texas          | US         |
| 2 | 1500        | 2011 Interiors Blvd                      | 99236       | South San Francisco | California     | US         |
| 3 | 1700        | 2004 Charade Rd                          | 98199       | Seattle             | Washington     | US         |
| 4 | 1800        | 460 Bloor St. W.                         | ON M5S 1X8  | Toronto             | Ontario        | CA         |
| 5 | 2500        | Magdalen Centre, The Oxford Science Park | OX9 9ZB     | Oxford              | Oxford         | UK         |

## 人力资源 (HR) 表说明 (续)

```
DESCRIBE regions
```

| Name        | Null     | Type         |
|-------------|----------|--------------|
| REGION_ID   | NOT NULL | NUMBER       |
| REGION_NAME |          | VARCHAR2(25) |

```
SELECT * FROM regions
```

|   | REGION_ID | REGION_NAME            |
|---|-----------|------------------------|
| 1 | 1         | Europe                 |
| 2 | 2         | Americas               |
| 3 | 3         | Asia                   |
| 4 | 4         | Middle East and Africa |

# 使用 SQL Developer

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

华周 (286991486@qq.com) has a non-transferable license to use  
this Student Guide.

## 课程目标

学完本附录后，应能完成以下工作：

- 列举 Oracle SQL Developer 的主要功能
- 确定 Oracle SQL Developer 的菜单项
- 创建数据库连接
- 管理数据库对象
- 使用 SQL 工作表
- 保存和运行 SQL 脚本
- 创建和保存报表

ORACLE

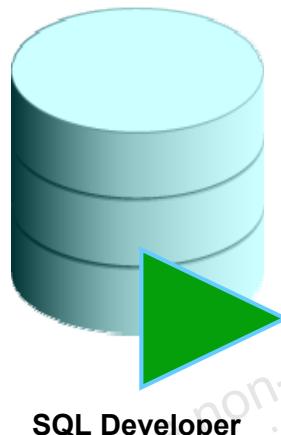
版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

本附录向您介绍一种称为 SQL Developer 的图形工具。您将学习如何使用 SQL Developer 执行数据库开发任务，还将学习如何使用 SQL 工作表执行 SQL 语句和 SQL 脚本。

## 什么是 Oracle SQL Developer

- Oracle SQL Developer 是一个图形工具，可以提高工作效率并简化数据库开发任务。
- 通过使用标准的 Oracle DB 验证可以连接到任何 Oracle DB 的目标方案。



SQL Developer

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 什么是 Oracle SQL Developer

Oracle SQL Developer 是一款免费的图形工具，旨在提高工作效率并简化日常数据库任务的开发过程。只需几次单击，就可以轻松地创建和调试存储过程、测试 SQL 语句以及查看优化程序计划。

SQL Developer 是一种用于开发数据库的可视化工具，可以简化以下任务：

- 浏览和管理数据库对象
- 执行 SQL 语句和脚本
- 编辑和调试 PL/SQL 语句
- 创建报表

通过使用标准的 Oracle DB 验证可以连接到任何 Oracle DB 的目标方案。连接后，就可以对数据库中的对象执行操作。

SQL Developer 1.2 发行版与 *Developer Migration Workbench* 紧密集成，让用户在一个位置就可以浏览第三方数据库中的数据库对象和数据并将这些数据库移植到 Oracle。您还可以连接到所选第三方（非 Oracle）数据库（如 MySQL、Microsoft SQL Server 和 Microsoft Access）的方案来查看这些数据库中的元数据和数据。

此外，SQL Developer 还支持 Oracle Application Express 3.0.1 (Oracle APEX)。

## SQL Developer 说明

- 随附在 Oracle Database 11g 发行版 2 中
- 以 Java 开发
- 支持 Windows、Linux 以及 Mac OS X 平台
- 默认情况下使用 Java 数据库连接 (JDBC) 瘦驱动程序进行连接
- 连接到 Oracle DB 版本 9.2.0.1 或更高版本
- 可从以下链接免费下载：
  - [http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)



版权所有 © 2010, Oracle。保留所有权利。

### SQL Developer 说明

Oracle SQL Developer 1.5 随附在 Oracle Database 11g 发行版 2 中。SQL Developer 是在 Oracle JDeveloper 集成开发环境 (IDE) 下采用 Java 开发的。因此，它是一种跨平台工具。此工具可以在 Windows、Linux 和 Mac 操作系统 (OS) X 平台上运行。

默认情况下，通过 JDBC 瘦驱动程序连接到数据库，因此不需要 Oracle 主目录。SQL Developer 不需要安装程序，您只需对已下载的文件进行解压缩即可。使用 SQL Developer 时，用户可以连接到 Oracle DB 9.2.0.1 和更高版本，以及包括 Express Edition 在内的所有 Oracle DB 版本。

#### 附注

对于 Oracle Database 11g 发行版 2 之前的 Oracle DB 版本，您必须下载并安装 SQL Developer。SQL Developer 1.5 可从以下链接免费下载：

[http://www.oracle.com/technology/products/database/sql\\_developer/index.html](http://www.oracle.com/technology/products/database/sql_developer/index.html)

有关 SQL Developer 的安装说明，您可以访问以下链接：

[http://download.oracle.com/docs/cd/E12151\\_01/index.htm](http://download.oracle.com/docs/cd/E12151_01/index.htm)



## SQL Developer 1.5 界面

SQL Developer 1.5 界面从左至右包含三个主要导航选项卡：

- “**Connections**（连接）” 选项卡：通过使用此选项卡，可以浏览对其有访问权限的数据库对象和用户。
- “**Files**（文件）” 选项卡：使用这一由“**Files**（文件）”文件夹图标表示的选项卡，您能够从本地计算机访问文件，无须使用“File > Open（文件 > 打开）”菜单项。
- “**Reports**（报表）” 选项卡：使用这一由“**Reports**（报表）”图标表示的选项卡，您能够运行预定义的报表或创建和添加自己的报表。

### 常规导航和使用

在 SQL Developer 导航页的左侧可查找和选择对象，在右侧可显示有关选定对象的信息。通过设置首选项可定制 SQL Developer 的各种不同的外观和行为。

**注：**需要至少定义一个连接，才能在连接到数据库方案后发出 SQL 查询或运行过程/函数。

## SQL Developer 1.5 界面（续）

### 菜单

以下菜单包含标准菜单项和代表 SQL Developer 特有功能的菜单项：

- **View (视图)**：包含影响 SQL Developer 界面中显示内容的选项
- **Navigate (导航)**：包含用于导航至各种窗格以及执行子程序的选项
- **Run (运行)**：包含选择函数或过程时要使用的选择项“Run File (运行文件)” 和“Execution Profile (执行概要文件)”，以及调试选项
- **Source (源)**：包含编辑函数和过程时要使用的选项
- **Versioning (版本化)**：为以下版本控制系统和源代码控制系统提供集成支持：并行版本系统 (CVS) 和子版本
- **Migration (移植)**：包含将第三方数据库移植到 Oracle 时要使用的选择项
- **Tools (工具)**：调用 SQL Developer 工具，如“SQL\*Plus”、“Preferences (首选项)” 和“SQL Worksheet (SQL 工作表)”

注：“Run (运行)” 菜单还包含为调试而选择函数或过程时要使用的选择项。这些选项与版本 1.2 中“Debug (调试)” 菜单中的选项一样。

## 创建数据库连接

- 必须至少建立一个数据库连接才能使用 SQL Developer。
- 可以为多个以下项创建和测试连接：
  - 数据库
  - 方案
- SQL Developer 将自动导入系统上 `tnsnames.ora` 文件中定义的所有连接。
- 可以将连接导出到可扩展标记语言 (XML) 文件。
- 创建的每个附加数据库连接都会在连接导航器层次结构中列出。



版权所有 © 2010, Oracle。保留所有权利。

### 创建数据库连接

连接是一个 SQL Developer 对象，指定了作为特定数据库的特定用户连接到该数据库时所需要的信息。要使用 SQL Developer，必须至少建立一个数据库连接，这个连接可以是现有连接，可以是已创建的连接，还可以是导入的连接。

可以为多个数据库和多个方案创建并测试连接。

默认情况下，`tnsnames.ora` 文件位于 `$ORACLE_HOME/network/admin` 目录中，但该文件也可位于 `TNS_ADMIN` 环境变量或注册表值指定的目录中。启动 SQL Developer 后显示“Database Connections（数据库连接）”对话框时，SQL Developer 会自动导入系统的 `tnsnames.ora` 文件中定义的所有连接。

**注：**在 Windows 上，如果 `tnsnames.ora` 文件存在，但 SQL Developer 没有使用其中的连接，请将 `TNS_ADMIN` 定义为系统环境变量。

可以将连接导出到 XML 文件，以供将来重复使用。

您可用其他用户的身份，另外创建到同一个数据库或到不同数据库的其它连接。



### 创建数据库连接（续）

要创建数据库连接，请执行以下步骤：

1. 在“Connections（连接）”选项卡页上，右键单击“Connections（连接）”，然后选择“New Connection（新建连接）”。
2. 在“New>Select Database Connection（新建/选择数据库连接）”窗口中，输入连接名。输入要连接到的方案的用户名和口令。
  - a) 在“Role（角色）”下拉框中，可以选择“default（默认值）”或“SYSDBA”（对系统用户或具有 DBA 权限的任何用户，应选择 SYSDBA）。
  - b) 可选择的连接类型包括：
    - **Basic（基本）**：选择此类型时，输入要连接数据库的主机名和 SID。端口已设置为 1521。或者，如果要使用远程数据库连接，也可以选择直接输入“Service name（服务名）”。
    - **TNS**：可以在从 tnsnames.ora 文件导入的数据库别名中任意选择一个。
    - **LDAP**：可以在 Oracle Internet Directory（Oracle Identity Management 的组件之一）中查找数据库服务。
    - **Advanced（高级）**：可以定义一个要连接到数据库的定制 JDBC URL。

## 创建数据库连接（续）

- c) 单击“Test（测试）”，以确保已正确地设置了该连接。
- d) 单击“Connect（连接）”。

如果选中了“Save Password（保存口令）”复选框，则会在 XML 文件保存口令。这样在关闭又再次打开 SQL Developer 连接后，就不会提示您输入口令了。

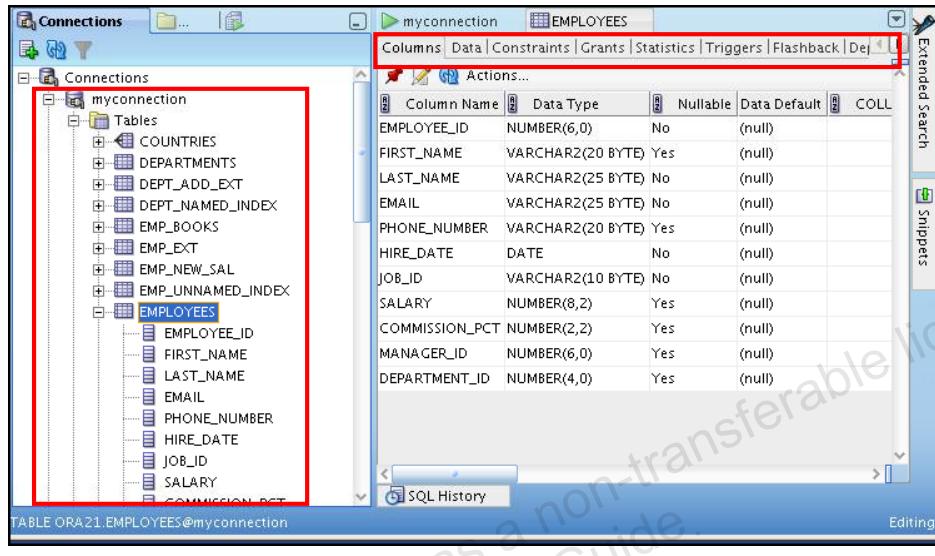
3. 此时该连接已添加在“Connections Navigator（连接导航器）”中。可以展开该连接，查看数据库对象和对象定义，例如，依赖性、详细资料、统计信息等。

**注：**在以上“New>Select Database Connection（新建/选择数据库连接）”窗口中，还可以使用 Access、MySQL 和 SQL Server 选项卡定义到非 Oracle 数据源的连接。不过，这些连接是只读连接，可用来浏览相应数据源中的对象和数据。

## 浏览数据库对象

使用连接导航器可以：

- 浏览数据库方案中的许多对象
- 快速查看对象定义



版权所有 © 2010, Oracle。保留所有权利。

ORACLE

## 浏览数据库对象

创建了数据库连接之后，可以使用连接导航器浏览数据库方案中的许多对象，包括表、视图、索引、程序包、过程、触发器和类型。

可以在不同的选项卡上查看各种对象的定义，从而了解从数据字典中提取的有关信息。

例如，如果在该导航器中选择一个表，则会在选项卡页上显示关于列、约束条件、权限、统计信息、触发器等的详细资料，非常易于查看。

如果要查看幻灯片中显示的 EMPLOYEES 表的定义，请执行以下步骤：

1. 在“Connections Navigator（连接导航器）”中展开“Connections（连接）”节点。
2. 展开“Tables（表）”。
3. 单击“EMPLOYEES”。默认情况下，“Columns（列）”选项卡处于选中状态。在该选项卡上会显示表的列说明。使用“Data（数据）”选项卡时，可以查看表数据，还可以输入新行、更新数据并将这些更改提交到数据库中。

## 显示表结构

使用 DESCRIBE 命令可显示表结构：

The screenshot shows the Oracle SQL Developer interface. The top bar displays the connection name "myconnection" and the execution time "1.69686902 seconds". The main area shows the command "DESC EMPLOYEES" and its output. The output table has three columns: Name, Null, and Type. The rows list the columns of the EMPLOYEES table, including EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NUMBER, HIRE\_DATE, JOB\_ID, SALARY, COMMISSION\_PCT, MANAGER\_ID, and DEPARTMENT\_ID. The "Null" column indicates whether the column can contain null values, and the "Type" column shows the data type for each column. A message at the bottom of the table says "11 rows selected".

| Name           | Null     | Type         |
|----------------|----------|--------------|
| EMPLOYEE_ID    | NOT NULL | NUMBER(6)    |
| FIRST_NAME     |          | VARCHAR2(20) |
| LAST_NAME      | NOT NULL | VARCHAR2(25) |
| EMAIL          | NOT NULL | VARCHAR2(25) |
| PHONE_NUMBER   |          | VARCHAR2(20) |
| HIRE_DATE      | NOT NULL | DATE         |
| JOB_ID         | NOT NULL | VARCHAR2(10) |
| SALARY         |          | NUMBER(8,2)  |
| COMMISSION_PCT |          | NUMBER(2,2)  |
| MANAGER_ID     |          | NUMBER(6)    |
| DEPARTMENT_ID  |          | NUMBER(4)    |

ORACLE

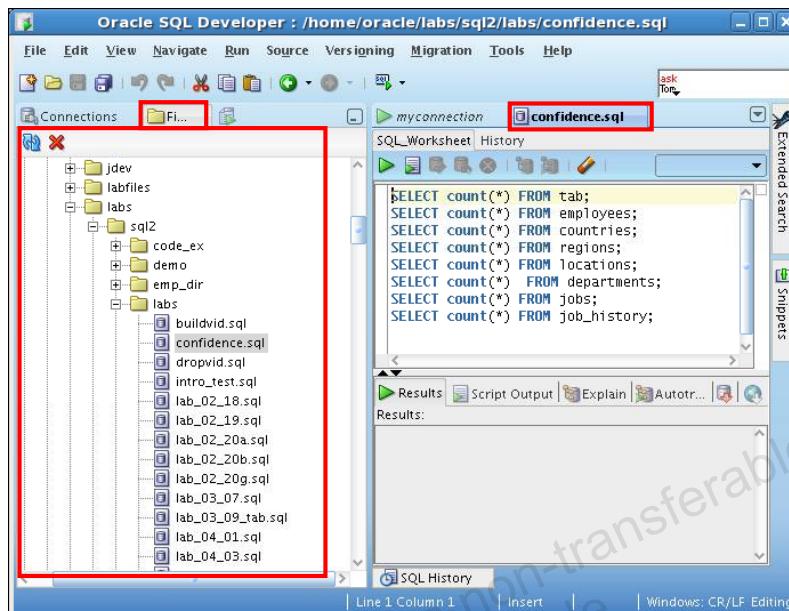
版权所有 © 2010, Oracle。保留所有权利。

## 显示表结构

在 SQL Developer 中，还可使用 DESCRIBE 命令显示表结构。执行该命令后会显示列名和数据类型，以及列中是否必须包含数据的指示信息。

## 浏览文件

使用文件导航器可以浏览文件系统并打开系统文件。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

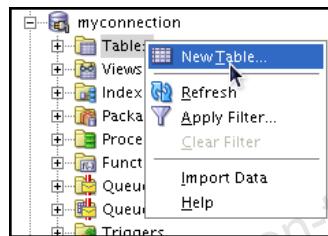
## 浏览数据库对象

可以使用文件导航器浏览并打开系统文件。

- 要查看文件导航器，请单击“Files（文件）”选项卡，或单击“View > Files（查看 > 文件）”。
- 要查看文件内容，请双击一个文件名以在 SQL 工作表区中显示其内容。

## 创建方案对象

- SQL Developer 支持通过以下方式创建任意方案对象：
  - 在 SQL 工作表中执行 SQL 语句
  - 使用上下文菜单
- 使用编辑对话框或多个上下文相关菜单之一编辑对象。
- 查看用于执行调整（如创建新对象或编辑现有方案对象）的数据定义语言 (DDL)。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

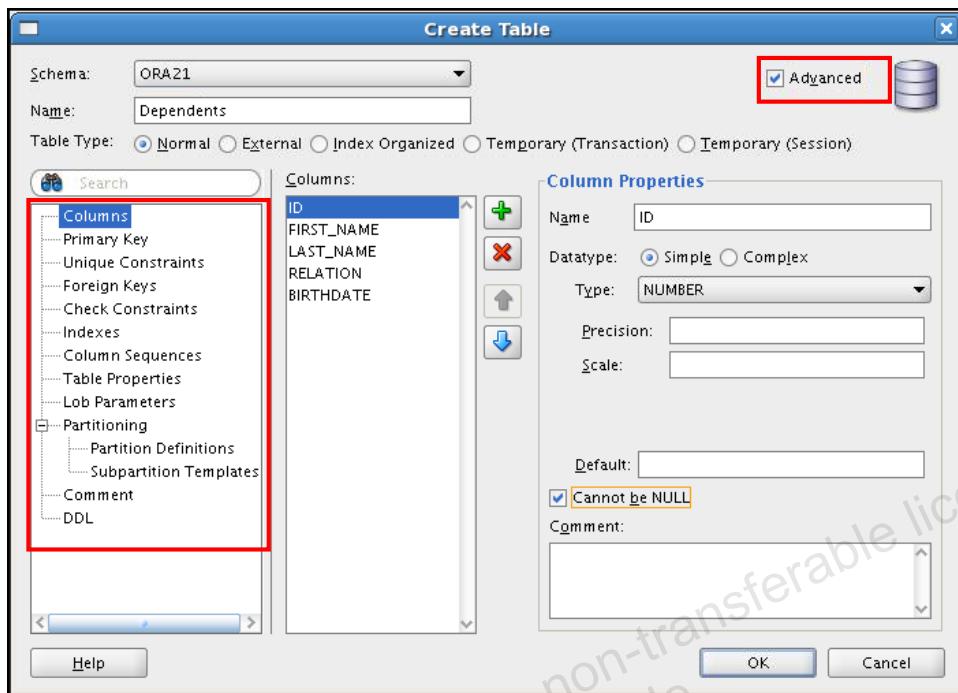
### 创建方案对象

SQL Developer 支持通过在 SQL 工作表中执行 SQL 语句来创建任意方案对象。此外，也可以使用上下文菜单创建对象。创建对象之后，可以立即使用编辑对话框或多个上下文相关菜单之一来编辑对象。

在创建新对象或编辑现有对象时，可以查看用于执行这些调整的 DDL。如果要查看方案中一个或多个对象的完整 DDL，则可以使用“Export DDL (导出 DDL)”选项。

该幻灯片中显示了如何使用上下文菜单创建表。要打开对话框来创建新表，请右键单击“Tables (表)”，然后选择“New Table (新建表)”。在创建和编辑数据库对象的对话框中有多个选项卡，在每个选项卡上会显示该类对象按逻辑分组的属性。

## 创建新表：示例



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 创建新表：示例

在“Create Table（创建表）”对话框中，如果没有选中“Advanced（高级）”复选框，则可以通过指定列和一些常用功能来快速创建一个表。

如果选中了“Advanced（高级）”复选框，则在“Create Table（创建表）”对话框中会显示多个选项，因此可以在创建表时指定一组扩展功能。

幻灯片示例显示了如何通过选中“Advanced（高级）”复选框创建 DEPENDENTS 表。

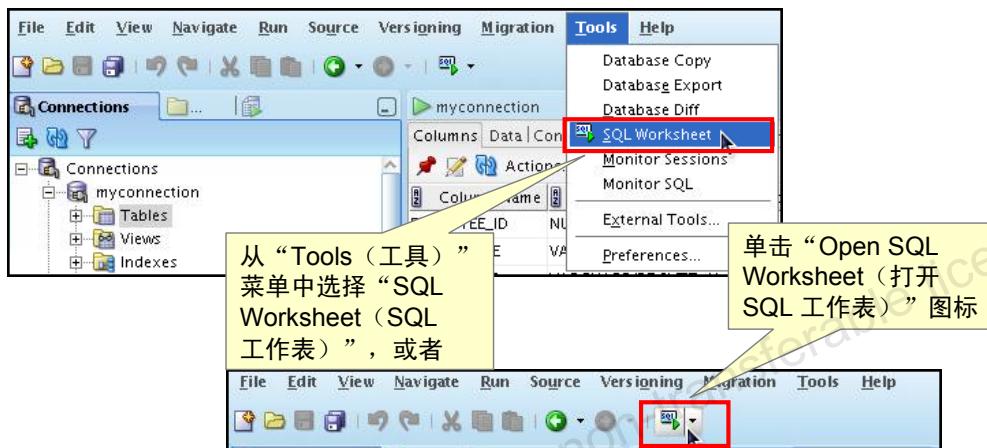
要创建一个新表，请执行以下步骤：

1. 在“Connections Navigator（连接导航器）”中，右键单击“Tables（表）”。
2. 选择“Create TABLE（创建表）”。
3. 在“Create Table（创建表）”对话框中，选中“Advanced（高级）”。
4. 指定列信息。
5. 单击“OK（确定）”。

尽管未作要求，但还是应当使用对话框中的“Primary Key（主键）”选项卡指定一个主键。有时，您可能需要编辑所创建的表，为此，请在“Connections Navigator（连接导航器）”中右键单击该表并选择“Edit（编辑）”。

## 使用 SQL 工作表

- 使用 SQL 工作表输入并执行 SQL、PL/SQL 和 SQL\*Plus 语句。
- 指定与工作表相关的数据库连接可处理的所有操作。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 SQL 工作表

连接到数据库后，用于该连接的“SQL Worksheet（SQL 工作表）”窗口就会自动打开。可以使用 SQL 工作表输入并执行 SQL、PL/SQL 和 SQL\*Plus 语句。SQL 工作表在某种程度上支持 SQL\*Plus 语句。SQL 工作表不支持的 SQL\*Plus 语句会被忽略，因而不会传递到数据库。

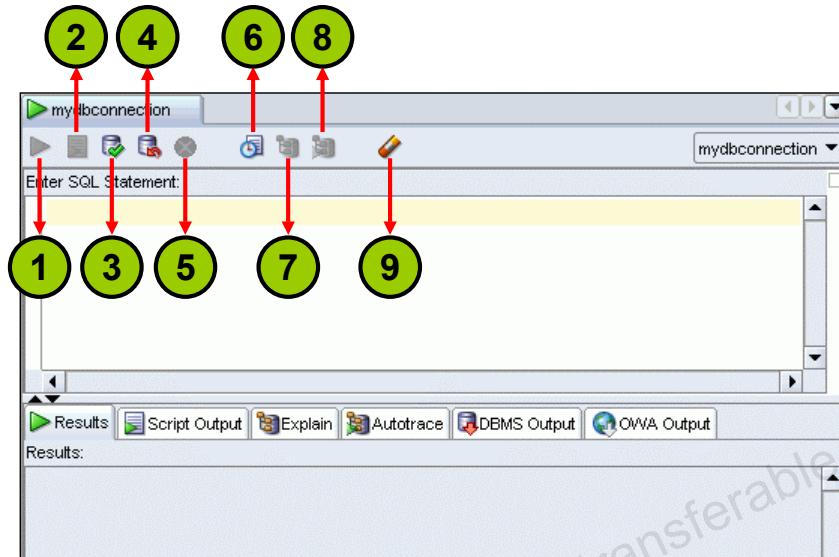
可以指定与工作表相关的数据库连接可处理的操作，例如：

- 创建表
- 插入数据
- 创建和编辑触发器
- 选择表中的数据
- 在文件中保存选定的数据

可以通过使用以下操作之一来显示 SQL 工作表：

- 选择“Tools > SQL Worksheet（工具 > SQL 工作表）”
- 单击“Open SQL Worksheet（打开 SQL 工作表）”图标

## 使用 SQL 工作表



### 使用 SQL 工作表（续）

您可能希望使用快捷键或图标来执行特定任务，例如执行 SQL 语句、运行脚本和查看已执行的 SQL 语句的历史记录。可以使用包含图标的 SQL 工作表工具栏执行以下任务：

1. **执行语句：** 执行“Enter SQL Statement（输入 SQL 语句）”框中光标处的语句。可以在 SQL 语句中使用绑定变量，但不能使用替代变量。
2. **运行脚本：** 通过使用脚本运行程序执行“Enter SQL Statement（输入 SQL 语句）”框中的所有语句。可以在 SQL 语句中使用替代变量，但不能使用赋值变量。
3. **提交：** 将所有更改写入数据库，然后结束事务处理。
4. **回退：** 放弃对数据库所做的所有更改，不将这些更改写入数据库，然后结束事务处理。
5. **取消：** 停止当前正在执行的任何语句。
6. **SQL 历史记录：** 显示一个对话框，其中包含有关已执行的 SQL 语句的信息。
7. **执行解释计划：** 生成执行计划，单击“Explain（解释）”选项卡可看到此计划。
8. **自动跟踪：** 为语句生成跟踪信息。
9. **清除：** 擦除“Enter SQL Statement（输入 SQL 语句）”框中的一条或多条语句。

## 使用 SQL 工作表

- 使用 SQL 工作表可以输入并执行 SQL、PL/SQL 和 SQL \*Plus 语句。
- 指定与工作表相关的数据库连接可处理的所有操作。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 SQL 工作表（续）

连接到数据库后，用于该连接的“SQL Worksheet（SQL 工作表）”窗口就会自动打开。可以使用 SQL 工作表输入并执行 SQL、PL/SQL 和 SQL\*Plus 语句。支持将所有的 SQL 和 PL/SQL 命令直接从 SQL 工作表传递到 Oracle 数据库。SQL Developer 中使用的 SQL\*Plus 命令必须先由 SQL 工作表进行解释，才能传递到数据库。

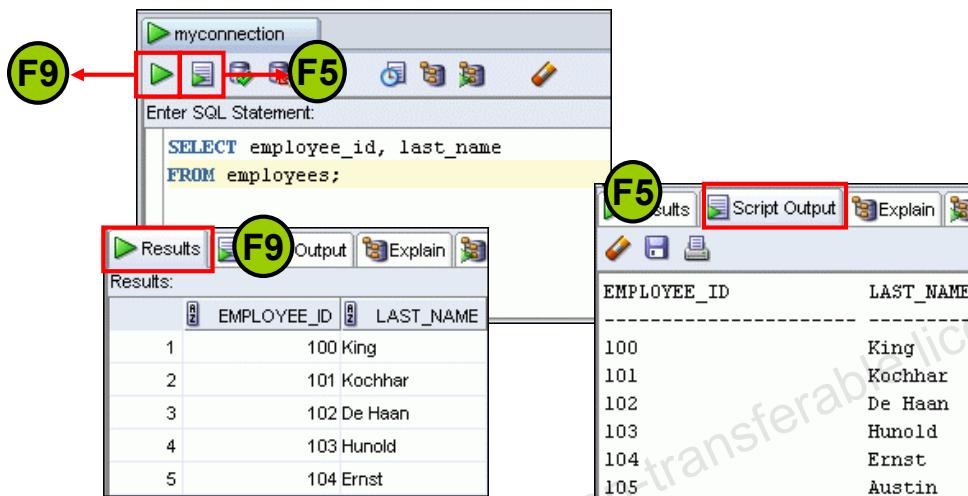
SQL 工作表当前支持许多 SQL\*Plus 命令。SQL 工作表不支持的命令将被忽略，不会发送到 Oracle 数据库。通过 SQL 工作表，可以执行 SQL 语句和一些 SQL\*Plus 命令。

可以通过使用以下两个选项中的任意一个来显示 SQL 工作表：

- 选择“Tools > SQL Worksheet（工具 > SQL 工作表）”。
- 单击“Open SQL Worksheet（打开 SQL 工作表）”图标。

## 执行 SQL 语句

使用“Enter SQL Statement（输入 SQL 语句）”框输入一条或多条 SQL 语句。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 执行 SQL 语句

幻灯片中的示例显示了按 F9 键或使用“Execute Statement（执行语句）”时与按 F5 或使用“Run Script（运行脚本）”时，同一查询的不同输出结果。



## 保存 SQL 脚本

您可以将 SQL 语句从 SQL 工作表保存到文本文件中。要保存“Enter SQL Statement（输入 SQL 语句）”框的内容，请执行以下步骤：

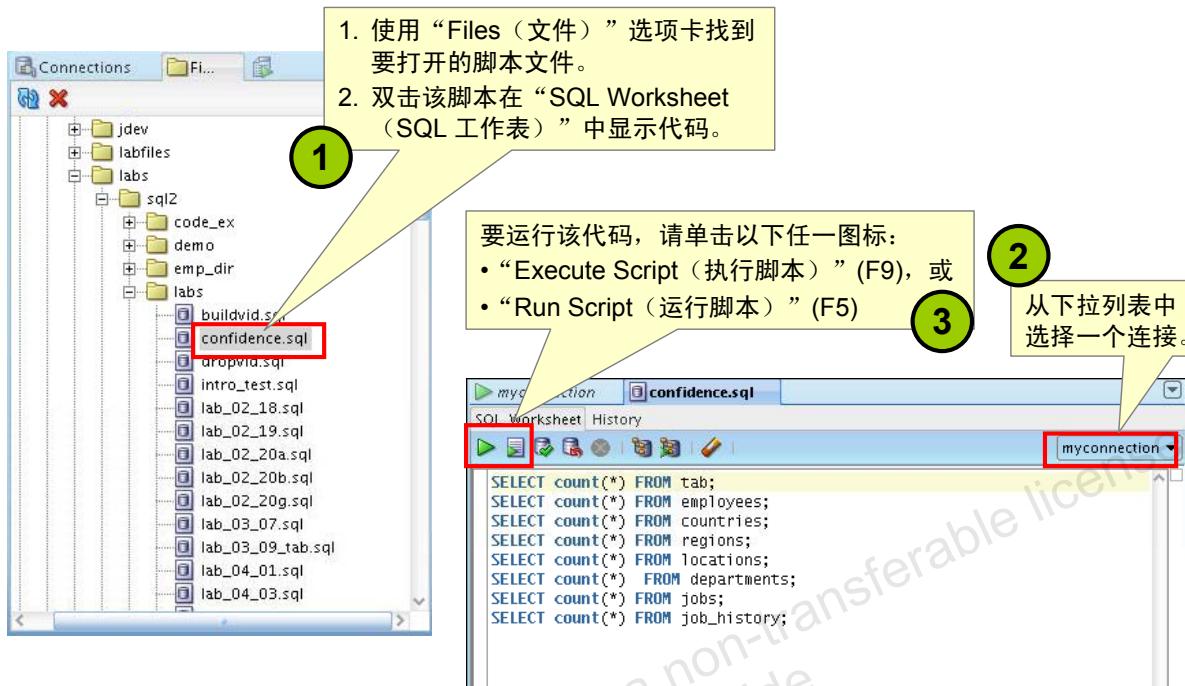
1. 单击“Save（保存）”图标或使用“File > Save（文件 > 保存）”菜单项。
2. 在 Windows “Save（保存）”对话框中，输入文件名和文件的保存位置。
3. 单击“Save（保存）”。

将内容保存到文件后，“Enter SQL Statement（输入 SQL 语句）”窗口将显示文件内容的选项卡页。一次可以打开多个文件。每个文件都显示为一个选项卡页。

### 脚本路径

可以选择一个路径作为查找和保存脚本时使用的默认路径。在“Tools > Preferences > Database > Worksheet Parameters（工具 > 首选项 > 数据库 > 工作表参数）”下，在“Select default path to look for scripts（选择用于查找脚本的默认路径）”字段中输入值。

## 执行已保存的脚本文件：方法 1



版权所有 © 2010, Oracle。保留所有权利。

ORACLE

### 执行已保存的脚本文件：方法 1

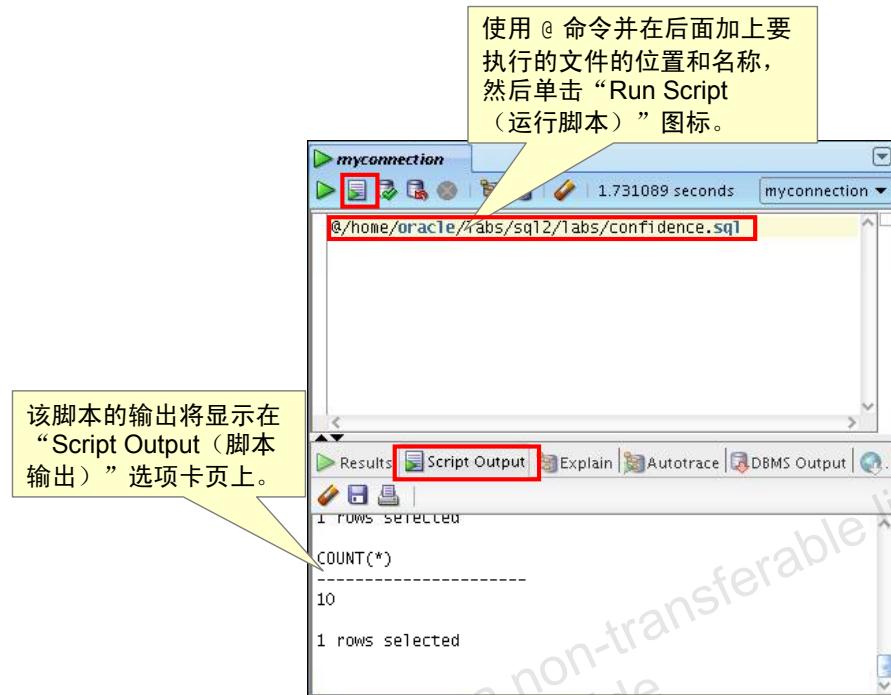
要打开脚本文件并在“SQL Worksheet（SQL 工作表）”区域中显示代码，请执行以下步骤：

1. 在文件导航器中选择（或导航至）要打开的脚本文件。
2. 双击将其打开。脚本文件的代码将显示在“SQL Worksheet（SQL 工作表）”区域中。
3. 从连接下拉列表中选择一个连接。
4. 要运行该代码，请在“SQL Worksheet（SQL 工作表）”工具栏中单击“Run Script（运行脚本）”图标 (F5)。如果未从连接下拉列表中选择连接，将显示连接对话框。选择要用于脚本执行的连接。

您也可以执行以下步骤：

1. 选择“File > Open（文件 > 打开）”。此时显示“Open（打开）”对话框。
2. 在“Open（打开）”对话框中，选择（或导航至）要打开的脚本文件。
3. 单击“Open（打开）”。脚本文件的代码将显示在“SQL Worksheet（SQL 工作表）”区域中。
4. 从连接下拉列表中选择一个连接。
5. 要运行该代码，请在“SQL Worksheet（SQL 工作表）”工具栏中单击“Run Script（运行脚本）”图标 (F5)。如果未从连接下拉列表中选择连接，将显示连接对话框。选择要用于脚本执行的连接。

## 执行已保存的脚本文件：方法 2



ORACLE

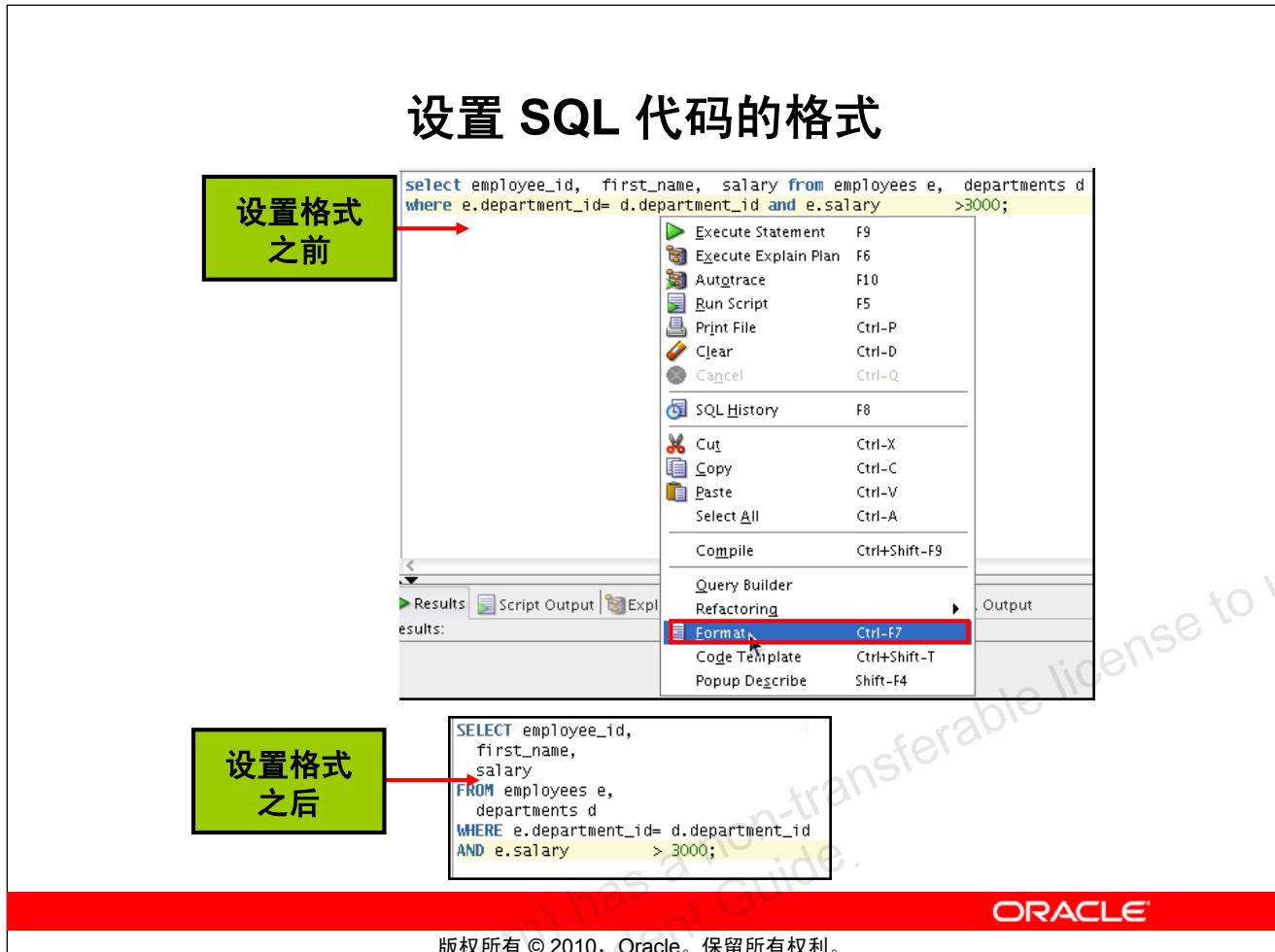
版权所有 © 2010, Oracle。保留所有权利。

### 执行已保存的脚本文件：方法 2

要运行已保存的 SQL 脚本，请执行以下步骤：

1. 在“Enter SQL Statement（输入 SQL 语句）”窗口中，使用 @ 命令并在后面加上要运行的文件的位置和名称。
2. 单击“Run Script（运行脚本）”图标。

此时将在“Script Output（脚本输出）”选项卡页上显示文件运行的结果。还可以通过单击“Script Output（脚本输出）”选项卡页上的“Save（保存）”图标保存脚本输出。此时将显示 Windows“保存”对话框，您可以为文件指定名称和位置。



## 设置 SQL 代码的格式

您可能希望通过缩进、间距、大小写和行间隔等方式来使 SQL 代码更加简洁易懂。SQL Developer 提供了设置 SQL 代码格式的功能。

要设置 SQL 代码的格式，请在语句区域内右键单击，然后选择“Format SQL（设置 SQL 格式）”。

在幻灯片示例中，在设置格式之前，SQL 代码中的关键字没有大写，语句也没有适当地缩进。在设置格式之后，关键字已大写，语句也已适当地缩进，因此 SQL 代码更加简洁易懂。



## 使用片段

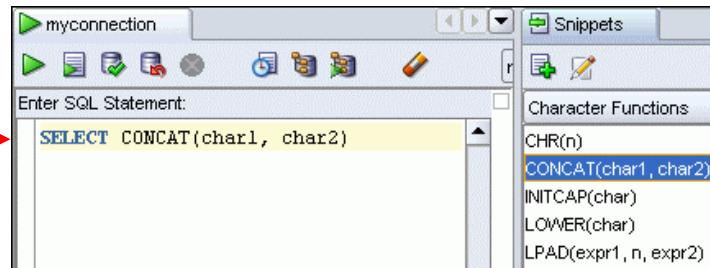
使用 SQL 工作表时，或者创建或编辑 PL/SQL 函数或过程时，可能需要使用特定的代码段。SQL Developer 提供了名为“片段”的功能。片段就是代码段，如 SQL 函数、优化程序提示和其它 PL/SQL 编程技术。可以将片段拖放到编辑器窗口中。

要显示“Snippets（片段）”，请选择“View > Snippets（视图 > 片段）”。

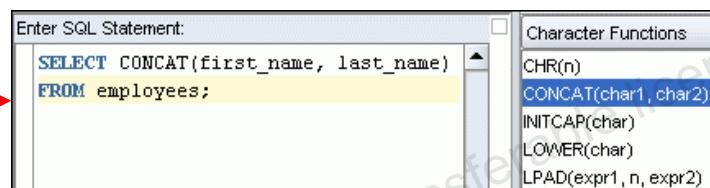
“Snippets（片段）”窗口显示在右侧。可以使用下拉列表选择一个组。“Snippets（片段）”按钮位于窗口的右边框上，在“Snippets（片段）”窗口隐藏起来时，单击该按钮可以使其显示出来。

## 使用片段：示例

**插入片段**



**编辑片段**



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用片段：示例

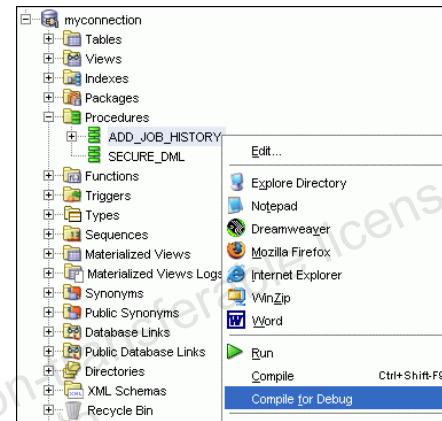
要将片段插入在 SQL 工作表或 PL/SQL 函数或过程的代码中，请将“Snippets（片段）”窗口中的片段拖放到代码的目标位置。然后可以编辑该语法，使 SQL 函数在当前上下文中生效。要在工具提示中查看 SQL 函数的简短说明，请将光标放在相应函数名的上方。

在幻灯片示例中，显示从“Snippets（片段）”窗口的字符函数组中拖动 CONCAT (char1, char2) 的过程。然后，编辑 CONCAT 函数语法，添加该语句的其余部分，如下所示：

```
SELECT CONCAT(first_name, last_name)
FROM employees;
```

## 调试过程和函数

- 使用 SQL Developer 调试 PL/SQL 函数和过程。
- 使用“Compile for Debug（为调试而编译）”选项执行 PL/SQL 编译，以便可对过程进行调试。
- 使用“Debug（调试）”菜单选项设置断点，并执行步入和步过任务。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 调试过程和函数

在 SQL Developer 中，您可以调试 PL/SQL 过程和函数。使用“Debug（调试）”菜单选项，可以执行以下调试任务：

- “Find Execution Point（查找执行点）”转至下一个执行点。
- “Resume（恢复）”继续执行。
- “Step Over（步过）”绕过下一个方法并转至该方法之后的下一条语句。
- “Step Into（步入）”转至下一个方法中的第一条语句。
- “Step Out（步出）”离开当前方法并转至下一条语句。
- “Step to End of Method（移到方法的末尾）”转至当前方法的最后一條语句。
- “Pause（暂停）”中断执行但不退出，因此允许恢复执行。
- “Terminate（终止）”中断并退出执行。您无法从此点恢复执行，只能通过单击“Source（源）”选项卡工具栏上的“Run（运行）”或“Debug（调试）”图标从函数或过程的开头开始运行或调试。
- “Garbage Collection（垃圾收集）”从高速缓存删除无效对象，以便高速缓存保存经常访问的对象和更有效的对象。

调试工具栏也以图标形式提供了这些选项。

## 数据库报表

SQL Developer 提供了很多有关数据库及其对象的预定义报表。

| Owner  | Name                     | Type    | Referenced Owner | Referenced Name          |
|--------|--------------------------|---------|------------------|--------------------------|
| CTXSYS | CTX_CLASSES              | VIEW    | CTXSYS           | DR\$CLASS                |
| CTXSYS | CTX_CLS                  | PACKAGE | SYS              | STANDARD                 |
| CTXSYS | CTX_DOC                  | PACKAGE | SYS              | STANDARD                 |
| CTXSYS | CTX_INDEX_SETS           | VIEW    | CTXSYS           | DR\$INDEX_SET            |
| CTXSYS | CTX_INDEX_SETS           | VIEW    | SYS              | USER\$                   |
| CTXSYS | CTX_INDEX_SET_INDEXES    | VIEW    | CTXSYS           | DR\$INDEX_SET            |
| CTXSYS | CTX_INDEX_SET_INDEXES    | VIEW    | CTXSYS           | DR\$INDEX_SET_INDEX      |
| CTXSYS | CTX_INDEX_SET_INDEXES    | VIEW    | SYS              | USER\$                   |
| CTXSYS | CTX_OBJECTS              | VIEW    | CTXSYS           | DR\$CLASS                |
| CTXSYS | CTX_OBJECTS              | VIEW    | CTXSYS           | DR\$OBJECT               |
| CTXSYS | CTX_OBJECT_ATTRIBUTES    | VIEW    | CTXSYS           | DR\$CLASS                |
| CTXSYS | CTX_OBJECT_ATTRIBUTES    | VIEW    | CTXSYS           | DR\$OBJECT               |
| CTXSYS | CTX_OBJECT_ATTRIBUTES    | VIEW    | CTXSYS           | DR\$OBJECT_ATTRIBUTE     |
| CTXSYS | CTX_OBJECT_ATTRIBUTE_LOV | VIEW    | CTXSYS           | DR\$CLASS                |
| CTXSYS | CTX_OBJECT_ATTRIBUTE_LOV | VIEW    | CTXSYS           | DR\$OBJECT               |
| CTXSYS | CTX_OBJECT_ATTRIBUTE_LOV | VIEW    | CTXSYS           | DR\$OBJECT_ATTRIBUTE     |
| CTXSYS | CTX_OBJECT_ATTRIBUTE_LOV | VIEW    | CTXSYS           | DR\$OBJECT_ATTRIBUTE_LOV |
| CTXSYS | CTX_PARAMETERS           | VIEW    | CTXSYS           | DR\$PARAMETER            |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 数据库报表

SQL Developer 提供了很多有关数据库及其对象的报表。这些报表可以分为以下几种类别：

- 关于数据库的报表
- 数据库管理报表
- 表报表
- PL/SQL 报表
- 安全性报表
- XML 报表
- 作业报表
- 流报表
- 所有对象报表
- 数据字典报表
- 用户定义报表

## 数据库报表（续）

要显示报表，请单击窗口左侧的“Reports（报表）”选项卡。单个报表显示在窗口右侧的选项卡窗格中。对于每个报表，您都可以（使用下拉列表）选择显示该报表要使用的数据库连接。在对象的报表中，显示的对象只是选定数据库连接所关联的数据库用户可见的那些对象，而且这些行通常按“Owner（所有者）”排序。还可以创建自己的用户定义报表。



## 创建用户定义报表

用户定义报表指的是由 SQL Developer 用户创建的报表。要创建用户定义报表，请执行以下步骤：

1. 右键单击“Reports（报表）”下的“User Defined Reports（用户定义报表）”节点，然后选择“Add Report（添加报表）”。
2. 在“Create Report（创建报表）”对话框中，指定报表名和用于检索报表信息的 SQL 查询。然后单击“Apply（应用）”。

在幻灯片示例中，报表名指定为 emp\_sal。在提供的可选说明中指出，此报表包含 salary >= 10000 的雇员的详细资料。在 SQL 框中指定了用来检索用户定义报表中显示的信息的完整 SQL 语句。还可以包含一个可选的工具提示，当光标短暂停留在报表导航器中显示的报表名的正上方时，就会显示该工具提示。

可以在文件夹中对用户定义报表进行组织，可以按层次结构创建文件夹及子文件夹。要创建一个文件夹来存放用户定义报表，请右键单击“User Defined Reports（用户定义报表）”节点或该节点下的任意文件夹名，然后选择“Add Folder（添加文件夹）”。有关用户定义报表的信息（包括存放这些报表的任何文件夹）都存储在一个名为 UserReports.xml 的文件中，该文件位于存放用户特定信息的目录下。



## 搜索引擎和外部工具

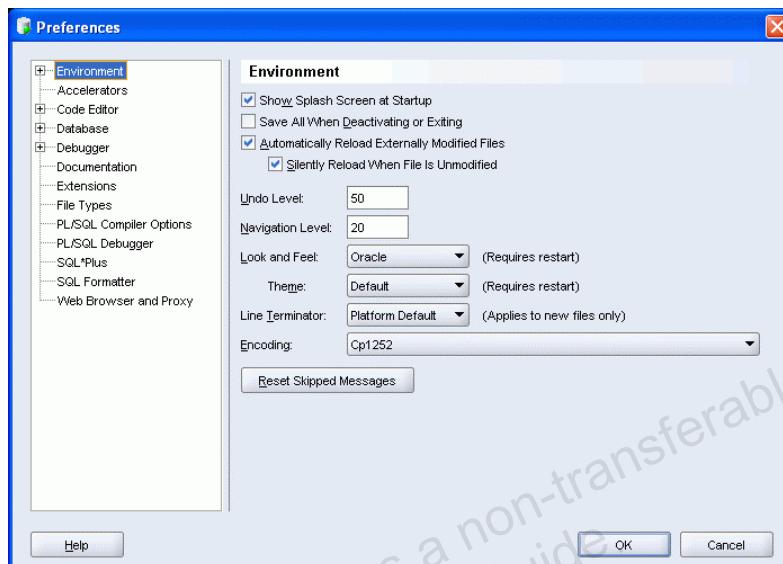
为了提高 SQL 开发人员的生产效率，SQL Developer 添加了常用搜索引擎和论坛（如 AskTom、Google 等）的快速链接。此外，还提供了一些常用工具（如 Notepad、Microsoft Word 和 Dreamweaver）的快捷方式图标。

您可以在现有列表中添加外部工具，甚至可以删除不常用工具的快捷方式。为此，请执行以下步骤：

1. 在“Tools（工具）”菜单中，选择“External Tools（外部工具）”。
2. 在“External Tools（外部工具）”对话框中，选择“New（新建）”可添加新工具。  
选择“Delete（删除）”可删除列表中的任何工具。

## 设置首选项

- 定制 SQL Developer 界面和环境。
- 在“Tools（工具）”菜单中，选择“Preferences（首选项）”。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 设置首选项

可根据个人爱好和需求修改 SQL Developer 首选项，从而定制各种不同的 SQL Developer 界面和环境。要修改 SQL Developer 的首选项，请选择“Tools（工具）”，然后选择“Preferences（首选项）”。

这些首选项可分为以下几类：

- Environment（环境）
- Accelerators（加速器）（键盘快捷方式）
- Code Editors（代码编辑器）
- Database（数据库）
- Debugger（调试器）
- Documentation（文档）
- Extensions（扩展）
- File Types（文件类型）
- Migration（移植）
- PL/SQL Compilers（PL/SQL 编译器）
- PL/SQL Debugger（PL/SQL 调试器）等

## 重置 SQL Developer 布局

```
Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$ locate windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.40/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
/home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48/windowinglayout.xml
[oracle@EDRSR5P1 ~]$ cd /home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ ls
Debugging.layout Editing.layout projects windowinglayout.xml
dtcache.xml preferences.xml settings.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$ rm windowinglayout.xml
[oracle@EDRSR5P1 o.ide.11.1.1.0.22.49.48]$
```



版权所有 © 2010, Oracle。保留所有权利。

### 重置 SQL Developer 布局

使用 SQL Developer 时，如果连接导航器消失或无法将“Log（日志）”窗口停靠在其原始位置，请执行以下步骤修复此问题：

1. 退出 SQL Developer。
2. 打开一个终端窗口并使用 `locate` 命令查找 `windowinglayout.xml` 的位置。
3. 转至包含 `windowinglayout.xml` 的目录并将其删除。
4. 重新启动 SQL Developer。

## 小结

在本附录中，您应该已经学会如何使用 SQL Developer 执行以下任务：

- 浏览、创建和编辑数据库对象
- 在 SQL 工作表中执行 SQL 语句和脚本
- 创建和保存定制报表

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

SQL Developer 是一款免费的图形工具，可用于简化数据库开发任务。使用 SQL Developer 时，可以浏览、创建和编辑数据库对象。可以使用 SQL 工作表运行 SQL 语句和脚本。通过使用 SQL Developer，您可以创建并保存自己的特殊报表集，以供将来反复使用。

# 使用 SQL\*Plus

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

- 登录到 SQL\*Plus
- 编辑 SQL 命令
- 使用 SQL\*Plus 命令设置输出格式
- 与脚本文件交互

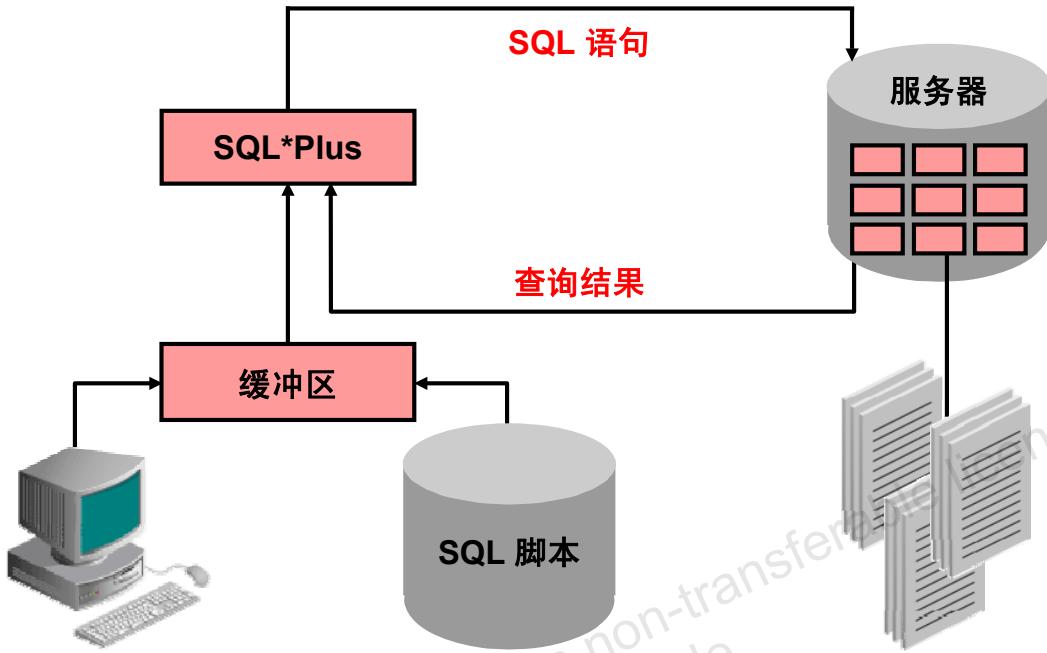
ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

您可能需要创建一些可反复使用的 SELECT 语句。本附录还介绍如何使用 SQL\*Plus 命令执行 SQL 语句。您将学习如何使用 SQL\*Plus 命令设置输出格式、编辑 SQL 命令以及如何在 SQL\*Plus 中保存脚本。

## SQL 和 SQL\*Plus 交互



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### SQL 和 SQL\*Plus

SQL 是一种命令语言，使用这种语言可通过任何工具或应用程序与 Oracle Server 进行通信。Oracle SQL 包含许多扩展。输入一条 SQL 语句后，该语句会被存储在名为 SQL 缓冲区的一部分内存中并一直保存在那里，直到输入新的 SQL 语句为止。SQL\*Plus 是一种 Oracle 工具，用于识别 SQL 语句并将 SQL 语句提交给 Oracle9i Server 来执行。它提供自己的命令语言。

### SQL 的特性

- 可供多种用户使用，包括具有很少编程经验或没有编程经验的用户
- 是非过程语言
- 可以减少创建和维护系统所需的时间
- 是类似英语的语言

## SQL 和 SQL\*Plus (续)

### SQL\*Plus 的特点

- 接受即席输入语句
- 接受来自文件的 SQL 输入
- 可提供一个行编辑器来修改 SQL 语句
- 可控制环境设置
- 可将查询结果的格式设置为基本报表格式
- 可访问本地和远程数据库

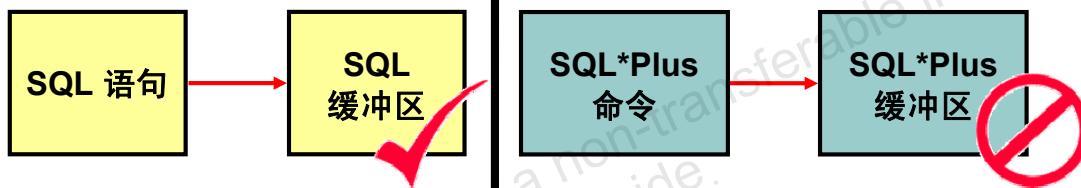
## SQL 语句和 SQL\*Plus 命令

### SQL

- 是一种语言
- 符合 ANSI 标准
- 关键字不能缩写
- 其语句用于处理数据库中的数据和表定义

### SQL\*Plus

- 是一种环境
- 是 Oracle 专用的
- 关键字可以缩写
- 不能通过其命令处理数据库中的值



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### SQL 和 SQL\*Plus (续)

下表对 SQL 和 SQL\*Plus 进行了比较：

| SQL                                 | SQL*Plus                    |
|-------------------------------------|-----------------------------|
| 是一种语言，用于通过与 Oracle Server 进行通信来访问数据 | 识别 SQL 语句并将 SQL 语句发送到服务器    |
| 是美国国家标准协会 (ANSI) 建立的标准 SQL          | 是用来执行 SQL 语句的 Oracle 专用界面   |
| 可处理数据库中的数据和表定义                      | 不允许处理数据库中的值                 |
| 以一行或多行的形式输入到 SQL 缓冲区中               | 一次输入一行，不会存储在 SQL 缓冲区中       |
| 没有续行符                               | 如果命令长度超过一行，则使用短划线 (-) 作为续行符 |
| 不能缩写                                | 可以缩写                        |
| 需要使用终止符来立即执行命令                      | 不需要终止符，就可以立即执行命令            |
| 使用函数来设置格式                           | 使用命令来设置数据格式                 |

## SQL\*Plus 概览

- 登录到 SQL\*Plus。
- 描述表结构。
- 编辑 SQL 语句。
- 在 SQL\*Plus 中执行 SQL。
- 将 SQL 语句保存到文件和将 SQL 语句附加到文件。
- 执行已保存的文件。
- 将命令从文件加载到缓冲区后进行编辑。

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### SQL\*Plus

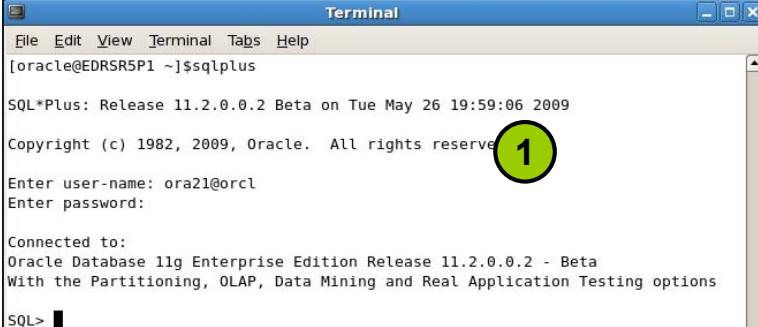
SQL\*Plus 是一种环境，在这个环境中可执行以下操作：

- 通过执行 SQL 语句，从数据库中检索、修改、添加和删除数据
- 设置查询结果格式，对查询结果进行计算，存储查询结果和以报表形式打印查询结果
- 通过创建脚本文件来存储 SQL 语句，以便将来重复使用

SQL\*Plus 命令可分为以下几个主要类别：

| 类别   | 用途                                  |
|------|-------------------------------------|
| 环境   | 影响会话中 SQL 语句的总体行为。                  |
| 设置格式 | 设置查询结果的格式。                          |
| 文件处理 | 保存、加载和运行脚本文件。                       |
| 执行   | 将 SQL 语句从 SQL 缓冲区发送到 Oracle Server。 |
| 编辑   | 修改缓冲区中的 SQL 语句。                     |
| 交互   | 创建变量并将其传递给 SQL 语句、打印变量值，以及在屏幕上显示消息。 |
| 其它   | 连接到数据库、处理 SQL*Plus 环境，以及显示列定义。      |

## 登录到 SQL\*Plus



```

Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$sqlplus

SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:59:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.

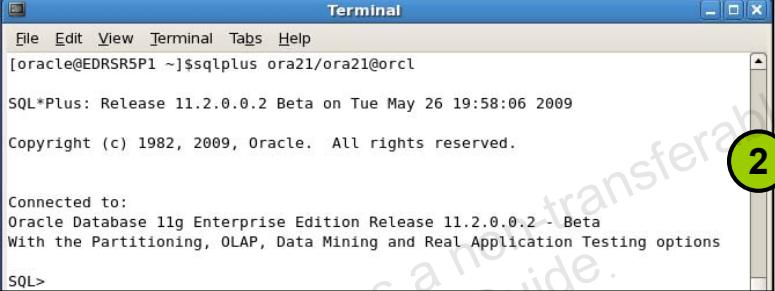
Enter user-name: ora21@orcl
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>

```

**sqlplus [username[/password[@database]]]**



```

Terminal
File Edit View Terminal Tabs Help
[oracle@EDRSR5P1 ~]$sqlplus ora21/ora21@orcl

SQL*Plus: Release 11.2.0.0.2 Beta on Tue May 26 19:58:06 2009
Copyright (c) 1982, 2009, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.0.2 - Beta
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL>

```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 登录到 SQL\*Plus

调用 SQL\*Plus 的方式取决于运行 Oracle DB 的操作系统的类型。

要在 Linux 环境中登录，请执行以下步骤：

1. 右键单击 Linux 桌面并选择终端。
2. 输入幻灯片中所显示的 sqlplus 命令。
3. 输入用户名、口令和数据库名。

在该语法中：

*username* 数据库的用户名

*password* 数据库口令（如果在此处输入口令，则口令可见）

*@database* 数据库连接字符串

**注：**为了确保口令的完整性，请不要在操作系统提示符下输入口令，而应仅输入用户名。  
请在口令提示符下输入口令。

## 显示表结构

使用 SQL\*Plus DESCRIBE 命令可显示表结构：

```
DESC[RIBE] tablename
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 显示表结构

在 SQL\*Plus 中，可以使用 DESCRIBE 命令显示表结构。执行该命令后会显示列名和数据类型，以及列中是否必须包含数据的指示信息。

在该语法中：

*tablename* 用户可以访问的任何现有表、视图或同义词的名称

要描述 DEPARTMENTS 表，请使用以下命令：

```
SQL> DESCRIBE DEPARTMENTS
 Name Null? Type
-----+
DEPARTMENT_ID NOT NULL NUMBER(4)
DEPARTMENT_NAME NOT NULL VARCHAR2(30)
MANAGER_ID NUMBER(6)
LOCATION_ID NUMBER(4)
```

## 显示表结构

```
DESCRIBE departments
```

| Name            | Null?    | Type         |
|-----------------|----------|--------------|
| DEPARTMENT_ID   | NOT NULL | NUMBER(4)    |
| DEPARTMENT_NAME | NOT NULL | VARCHAR2(30) |
| MANAGER_ID      |          | NUMBER(6)    |
| LOCATION_ID     |          | NUMBER(4)    |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 显示表结构（续）

幻灯片示例中显示了关于 DEPARTMENTS 表结构的信息。在结果中：

Null?: 指定列是否必须包含数据 (NOT NULL 指定列必须包含数据)

Type: 显示列的数据类型

## SQL\*Plus 编辑命令

- A [PPEND] *text*
- C [HANGE] / *old* / *new*
- C [HANGE] / *text* /
- CL [EAR] BUFF [ER]
- DEL
- DEL *n*
- DEL *m n*



版权所有 © 2010, Oracle。保留所有权利。

### SQL\*Plus 编辑命令

每次输入一行 SQL\*Plus 命令，而且该命令不会存储在 SQL 缓冲区中。

| 命令                                  | 说明                                                       |
|-------------------------------------|----------------------------------------------------------|
| A [PPEND] <i>text</i>               | 在当前行的末尾添加文本                                              |
| C [HANGE] / <i>old</i> / <i>new</i> | 在当前行上将 <i>old</i> 文本更改为 <i>new</i>                       |
| C [HANGE] / <i>text</i> /           | 从当前行中删除 <i>text</i>                                      |
| CL [EAR] BUFF [ER]                  | 删除 SQL 缓冲区中的所有行                                          |
| DEL                                 | 删除当前行                                                    |
| DEL <i>n</i>                        | 删除第 <i>n</i> 行                                           |
| DEL <i>m n</i>                      | 删除行 <i>m</i> 到 <i>n</i> 之间的行（包含 <i>m</i> 和 <i>n</i> 这两行） |

### 准则

- 如果还没有输入完命令就按了 Enter 键，SQL\*Plus 就会使用行号提示您。
- 您可以输入一个终止符（分号或斜杠）或按两次 Enter 键来终止 SQL 缓冲区输入。然后会出现 SQL 提示符。

## SQL\*Plus 编辑命令

- `I [NPUT]`
- `I [NPUT] text`
- `L [IST]`
- `L [IST] n`
- `L [IST] m n`
- `R [UN]`
- `n`
- `n text`
- `0 text`



版权所有 © 2010, Oracle。保留所有权利。

### SQL\*Plus 编辑命令（续）

| 命令                         | 说明                                                                                   |
|----------------------------|--------------------------------------------------------------------------------------|
| <code>I [NPUT]</code>      | 插入不限数量的行                                                                             |
| <code>I [NPUT] text</code> | 插入一个由 <code>text</code> 组成的行                                                         |
| <code>L [IST]</code>       | 列出 SQL 缓冲区中的所有行                                                                      |
| <code>L [IST] n</code>     | 列出一行（由 <code>n</code> 指定）                                                            |
| <code>L [IST] m n</code>   | 列出某一范围的行（从 <code>m</code> 到 <code>n</code> , 包括 <code>m</code> 和 <code>n</code> 这两行） |
| <code>R [UN]</code>        | 显示并运行缓冲区中的当前 SQL 语句                                                                  |
| <code>n</code>             | 指定该行成为当前行                                                                            |
| <code>n text</code>        | 用 <code>text</code> 替换第 <code>n</code> 行                                             |
| <code>0 text</code>        | 在第 1 行之前插入一行                                                                         |

注：在每个 SQL 提示符下，只能输入一条 SQL\*Plus 命令。SQL\*Plus 命令不会存储在缓冲区中。为了在下一行上继续输入 SQL\*Plus 命令，第一行应以连字符 (-) 结尾。

## 使用 LIST、n 和 APPEND

**LIST**

```
1 SELECT last_name
2* FROM employees
```

1

```
1* SELECT last_name
```

A , job\_id

```
1* SELECT last_name, job_id
```

**LIST**

```
1 SELECT last_name, job_id
2* FROM employees
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 LIST、n 和 APPEND

- 使用 L[IST] 命令可显示 SQL 缓冲区的内容。缓冲区中第 2 行旁边的星号 (\*) 表示第 2 行为当前行。您做的所有编辑操作都是对当前行进行的。
- 通过输入要编辑的行的编号 (n)，可以更改当前行的编号。然后会显示新的当前行。
- 使用 A[PPEND] 命令可在当前行上添加文本。然后会显示新编辑过的行。使用 LIST 命令可以验证缓冲区中的新内容。

**注：**许多 SQL\*Plus 命令（包括 LIST 和 APPEND）都可缩写成相应的首字母。例如，LIST 可缩写为 L，APPEND 可缩写为 A。

## 使用 CHANGE 命令

```
LIST
```

```
1* SELECT * from employees
```

```
c/employees/departments
```

```
1* SELECT * from departments
```

```
LIST
```

```
1* SELECT * from departments
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 CHANGE 命令

- 使用 L[IST] 可显示缓冲区的内容。
- 使用 C[HANGE] 命令可更改 SQL 缓冲区中当前行的内容。在本例中，可以用 departments 表替换 employees 表。然后会显示新的当前行。
- 使用 L[IST] 命令可验证缓冲区的新内容。

## SQL\*Plus 文件命令

- SAVE *filename*
- GET *filename*
- START *filename*
- @ *filename*
- EDIT *filename*
- SPOOL *filename*
- EXIT

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### SQL\*Plus 文件命令

SQL 语句可以与 Oracle Server 进行通信。SQL\*Plus 命令可以控制环境、设置查询结果的格式和管理文件。您可以使用下表中列出的命令：

| 命令                                                   | 说明                                                                            |
|------------------------------------------------------|-------------------------------------------------------------------------------|
| SAV[E] <i>filename</i> [.ext]<br>[REP[LACE]APP[END]] | 将 SQL 缓冲区中的当前内容保存到文件中。使用 APPEND 可以在现有文件中附加内容；使用 REPLACE 可以改写现有文件。默认扩展名为 .sql。 |
| GET <i>filename</i> [.ext]                           | 将先前保存文件的内容写入到 SQL 缓冲区中。文件的默认扩展名为 .sql。                                        |
| STA[RT] <i>filename</i> [.ext]                       | 运行先前保存的命令文件。                                                                  |
| @ <i>filename</i>                                    | 运行先前保存的命令文件（和 START 相同）。                                                      |
| ED[IT]                                               | 调用编辑器，将缓冲区内容保存到名为 afiedt.buf 的文件中。                                            |
| ED[IT] [ <i>filename</i> [.ext]]                     | 调用编辑器，编辑已保存文件的内容。                                                             |
| SPO[OL] [ <i>filename</i> [.ext]]   OFF OUT          | 将查询结果存储在文件中。OFF 将关闭假脱机文件。OUT 将关闭假脱机文件，并将文件结果发送到打印机。                           |
| EXIT                                                 | 退出 SQL*Plus。                                                                  |

## 使用 SAVE 和 START 命令

**LIST**

```
1 SELECT last_name, manager_id, department_id
2* FROM employees
```

**SAVE my\_query**

Created file my\_query

**START my\_query**

**LAST\_NAME**

**MANAGER\_ID DEPARTMENT\_ID**

King

90

Kochhar

100

90

...

107 rows selected.

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用 SAVE 和 START 命令

### SAVE

使用 SAVE 命令可以将缓冲区的当前内容存储在文件中。使用这种方法，可以将常用的脚本存储起来以备将来使用。

### START

使用 START 命令可以在 SQL\*Plus 中运行脚本。此外，还可以使用符号 @ 运行脚本。

@my\_query

## SERVEROUTPUT 命令

- 使用 SET SERVEROUT [PUT] 命令可控制是否在 SQL\*Plus 中显示存储过程或 PL/SQL 块的输出。
- DBMS\_OUTPUT 行的长度限制从 255 个字节增至 32767 个字节。
- 默认大小目前为无限制。
- 已设置 SERVEROUTPUT 时不能预分配资源。
- 因为对性能没有影响，所以请使用 UNLIMITED，除非要保留物理内存。

```
SET SERVEROUT[PUT] {ON | OFF} [SIZE {n | UNL[IMITED]}]
[FOR[MAT] {WRA[PPED] | WOR[D_WRAPPED] | TRU[NATED]}]
```



版权所有 © 2010, Oracle。保留所有权利。

### SERVEROUTPUT 命令

多数 PL/SQL 程序都通过 SQL 语句执行输入输出来在数据库表中存储数据或查询这些表。所有其它 PL/SQL 输入/输出都通过与其它程序交互的 API 来完成。例如，DBMS\_OUTPUT 程序包中包含诸如 PUT\_LINE 的过程。要在 PL/SQL 之外查看结果，需要另一个诸如 SQL\*Plus 的程序，才能读取和显示传递到 DBMS\_OUTPUT 的数据。

如果没有先发出 SQL\*Plus 命令 SET SERVEROUTPUT ON，则 SQL\*Plus 并不显示 DBMS\_OUTPUT 数据。如下所示：

```
SET SERVEROUTPUT ON
```

### 附注

- SIZE 设置了在 Oracle DB Server 中可以缓存的输出的字节数。默认值为 UNLIMITED。  
*n* 不能小于 2000 或大于 1,000,000。
- 有关 SERVEROUTPUT 的其它信息，请参阅《Oracle Database PL/SQL User's Guide and Reference 11g》。

## 使用 SQL\*Plus 的 SPOOL 命令

```
SPO[OL] [file_name[.ext]] [CRE[ATE] | REP[LACE] |
APP[END]] | OFF | OUT]
```

| 选项              | 说明                                 |
|-----------------|------------------------------------|
| file_name[.ext] | 将输出假脱机到指定的文件名                      |
| CRE[ATE]        | 创建具有指定名的新文件                        |
| REP[LACE]       | 替换现有文件的内容。如果该文件不存在， REPLACE 会创建该文件 |
| APP[END]        | 将缓冲区的内容添加到指定文件的末尾                  |
| OFF             | 停止假脱机                              |
| OUT             | 停止假脱机并将文件发送到计算机连接的标准（默认）打印机        |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用 SQL\*Plus 的 SPOOL 命令

SPOOL 命令用于将查询结果存储在一个文件中，或者根据需要将该文件发送到打印机。SPOOL 命令的功能已经增强。现在可以在现有文件中附加内容或替换现有文件，而以前只能使用 SPOOL 创建（和替换）文件。REPLACE 是默认设置。

要假脱机由脚本中的命令生成的输出而在屏幕上显示输出，请使用 SET TERMOUT OFF。SET TERMOUT OFF 并不影响交互运行命令的输出。

必须将包含白空格的文件名放在引号中。要使用 SPOOL APPEND 命令创建有效的 HTML 文件，必须使用 PROMPT 或类似命令创建 HTML 页的页眉和页脚。SPOOL APPEND 命令不会分析 HTML 标记。将 SQLPLUSCOMPAT [IBILITY] 设置为 9.2 或更低版本，可禁用 CREATE、APPEND 和 SAVE 参数。

## 使用 AUTOTRACE 命令

- 显示成功执行 SELECT、INSERT、UPDATE 或 DELETE 等 SQL 数据操纵语言 (DML) 语句后的报表。
- 此报表目前可包含执行统计信息和查询执行路径。

```
SET AUTOT[RACE] {ON | OFF | TRACE[ONLY]} [EXP[LAIN]]
[STATISTICS]
```

```
SET AUTOTRACE ON
-- The AUTOTRACE report includes both the optimizer
-- execution path and the SQL statement execution
-- statistics
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用 AUTOTRACE 命令

EXPLAIN 在执行 EXPLAIN PLAN 时会显示查询执行路径。STATISTICS 会显示 SQL 语句统计信息。AUTOTRACE 报表的格式可能有所不同，这取决于连接服务器的版本及其配置。DBMS\_XPLAN 程序包提供了几种预定义的格式，可以方便地显示 EXPLAIN PLAN 命令的输出。

### 附注

- 有关程序包和子程序的其它信息，请参阅《Oracle Database PL/SQL Packages and Types Reference 11g》指南。
- 有关 EXPLAIN PLAN 的其它信息，请参阅《Oracle Database SQL Reference 11g》。
- 有关执行计划和统计信息的其它信息，请参阅《Oracle Database Performance Tuning Guide 11g》。

## 小结

在本附录中，您应该已经学会如何将 SQL\*Plus 作为一种环境来执行以下任务：

- 执行 SQL 语句
- 编辑 SQL 语句
- 设置输出格式
- 与脚本文件交互

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

SQL\*Plus 是一种执行环境，可以使用它将 SQL 命令发送到数据库服务器，还可以使用它编辑和保存 SQL 命令。可以在 SQL 提示符下执行命令，也可以从脚本文件执行命令。



使用 JDeveloper

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

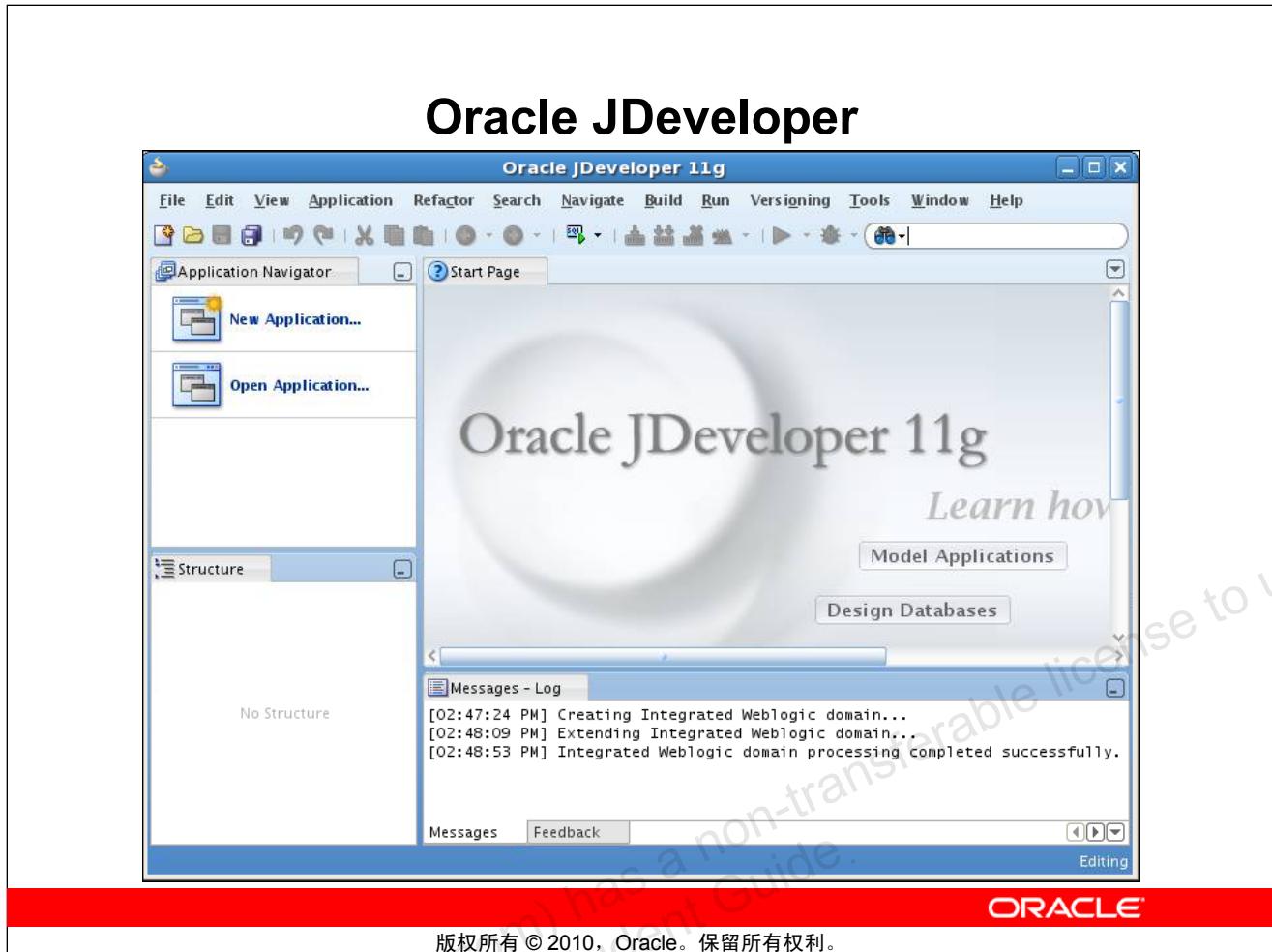
- 列举 Oracle JDeveloper 的主要功能
- 在 JDeveloper 中创建数据库连接
- 在 JDeveloper 中管理数据库对象
- 使用 JDeveloper 执行 SQL 命令
- 创建并运行 PL/SQL 程序单元

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

本附录将向您介绍一种工具 JDeveloper。您将学习如何使用 JDeveloper 执行数据库开发任务。

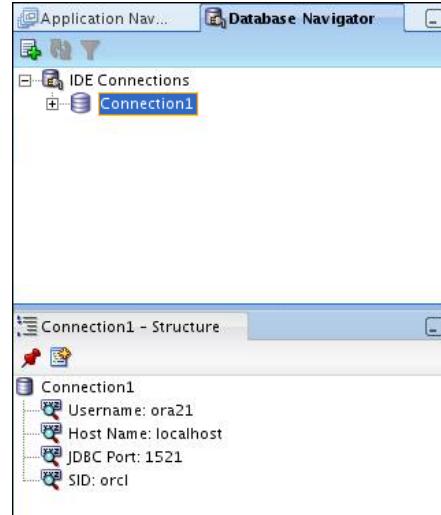


## Oracle JDeveloper

Oracle JDeveloper 是一个集成开发环境 (IDE)，用于开发和部署 Java 应用程序和 Web 服务。它支持软件开发周期 (SDLC) 从建模到部署的所有阶段。它的一大特色是，在开发应用程序时使用 Java、XML 和 SQL 的最新行业标准。

Oracle JDeveloper 11g 启用了一种新的 J2EE 开发方法，其特色是可以实现可视化的声明式开发。该创新方法使得 J2EE 开发更加简单而高效。

## 数据库导航器



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 数据库导航器

使用 Oracle JDeveloper，可以将连接数据库所需要的信息存储在名为“connection（连接）”的对象中。连接存储为 IDE 设置的一部分，可以导出和导入，因而可以方便地在用户组之间共享。连接有多种用途，从浏览数据库和构建应用程序一直到部署阶段的各个环节都会用到。



## 创建连接

连接是一个对象，用于指定作为特定数据库的特定用户连接到该数据库时所需要的信息。可以为多个数据库和多个方案创建并测试连接。

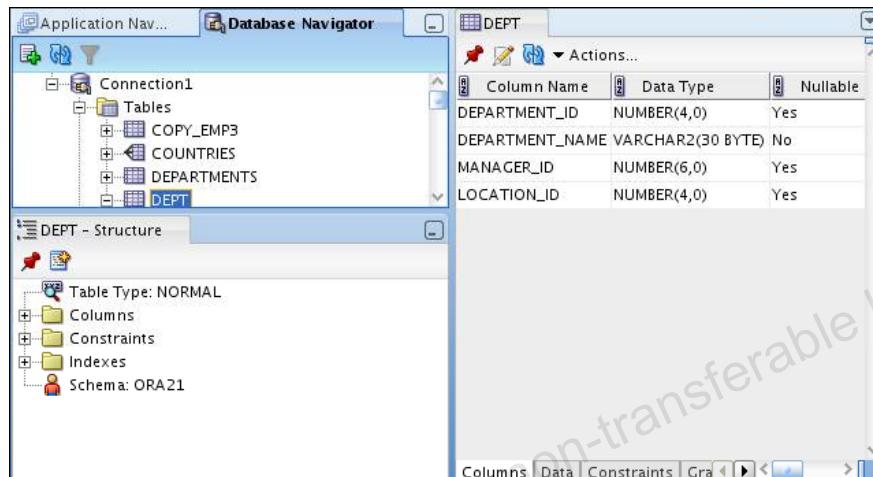
要创建数据库连接，请执行以下步骤：

1. 单击“Database Navigator（数据库导航器）”中的“New Connection（新建连接）”图标。
2. 在“Create Database Connection（创建数据库连接）”窗口中，输入连接名。输入要连接到的方案的用户名和口令。输入要连接到的数据库的SID。
3. 单击“Test Connection（测试连接）”，确保已正确地设置了该连接。
4. 单击“OK（确定）”。

## 浏览数据库对象

使用数据库导航器可以执行以下操作：

- 浏览数据库方案中的许多对象
- 快速查看对象定义



ORACLE

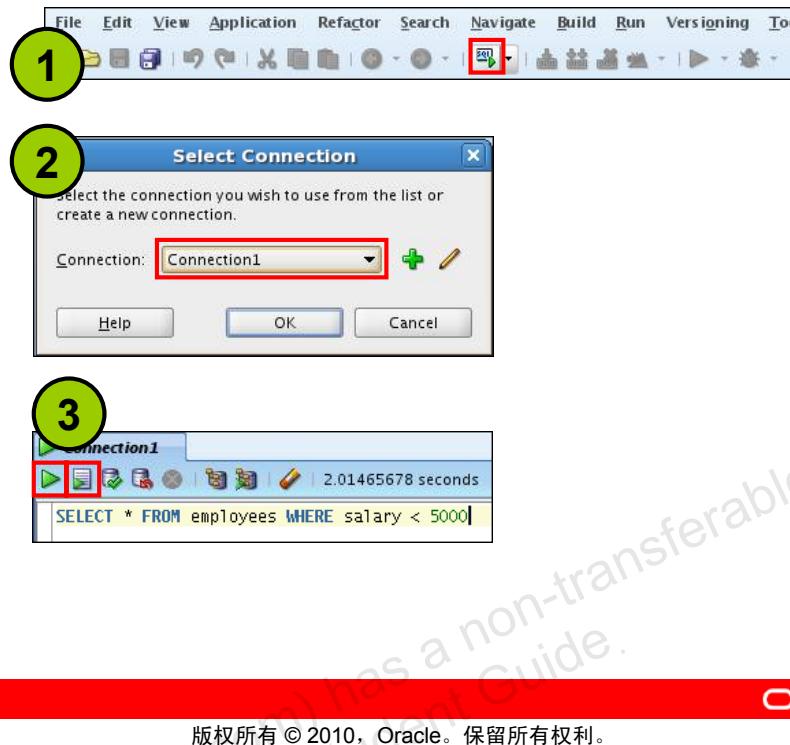
版权所有 © 2010, Oracle。保留所有权利。

### 浏览数据库对象

创建了数据库连接之后，可以使用数据库导航器浏览数据库方案中的许多对象，包括表、视图、索引、程序包、过程、触发器和类型。

可以在不同的选项卡上查看各种对象的定义，从而了解从数据字典中提取的有关信息。例如，如果在导航器中选择一个表，则会在选项卡页上显示关于列、约束条件、权限、统计信息、触发器等的详细资料，非常易于查看。

## 执行 SQL 语句



### 执行 SQL 语句

要执行 SQL 语句，请执行以下步骤：

1. 单击“Open SQL Worksheet（打开 SQL 工作表）”图标。
2. 选择连接。
3. 通过单击以下按钮之一执行 SQL 命令：
  - “Execute statement（执行语句）”按钮或按 F9。输出如下所示：

The screenshot shows the SQL Worksheet results for the query: SELECT \* FROM employees WHERE salary < 5000;. The results are displayed in a grid:

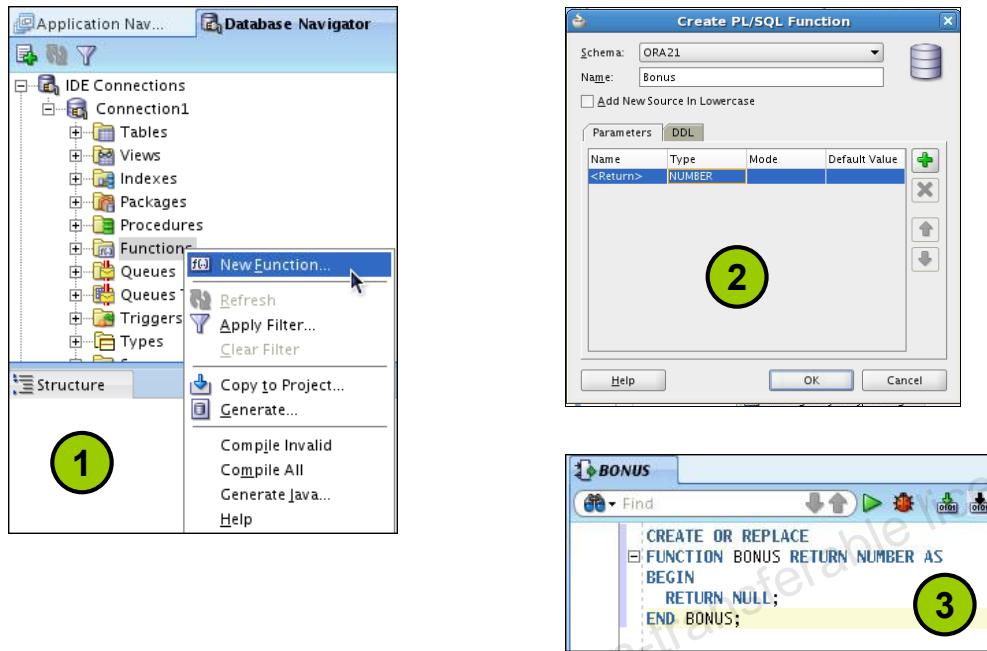
|   | EMPLOYEE_ID | FIRST_NAME | LAST_NAME |
|---|-------------|------------|-----------|
| 1 | 100         | Steven     | King      |
| 2 | 101         | Neena      | Kochhar   |

- “Run Script（运行脚本）”按钮或按 F5。输出如下所示：

The screenshot shows the SQL Worksheet results for the Run Script command, which executes the same SQL query. The results are displayed in a grid:

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME |
|-------------|------------|-----------|
| 100         | Steven     | King      |

## 创建程序单元



函数骨架

ORACLE

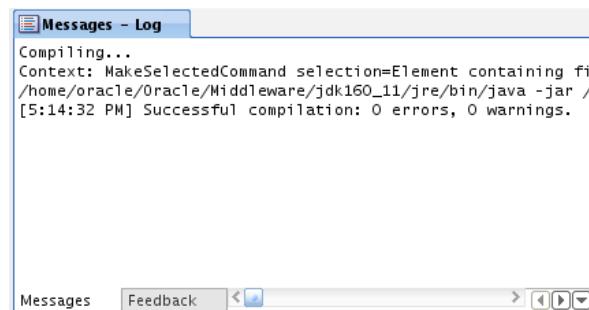
版权所有 © 2010, Oracle。保留所有权利。

### 创建程序单元

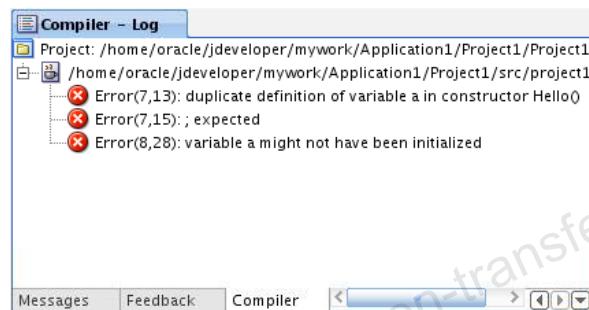
要创建 PL/SQL 程序单元，请执行以下步骤：

1. 选择“View > Database Navigator（视图 > 数据库导航器）”。选择并展开一个数据库连接。右键单击对应于对象类型（过程、程序包和函数）的文件夹。选择“New Procedures（新建过程）”、“New Packages（新建程序包）”或“New Functions（新建函数）”。
2. 输入函数、程序包或过程的有效名称，再单击“OK（确定）”。
3. 此时将在代码编辑器中创建并打开骨架定义。然后编辑子程序以适应需要。

## 编译



编译时出错



编译无错误

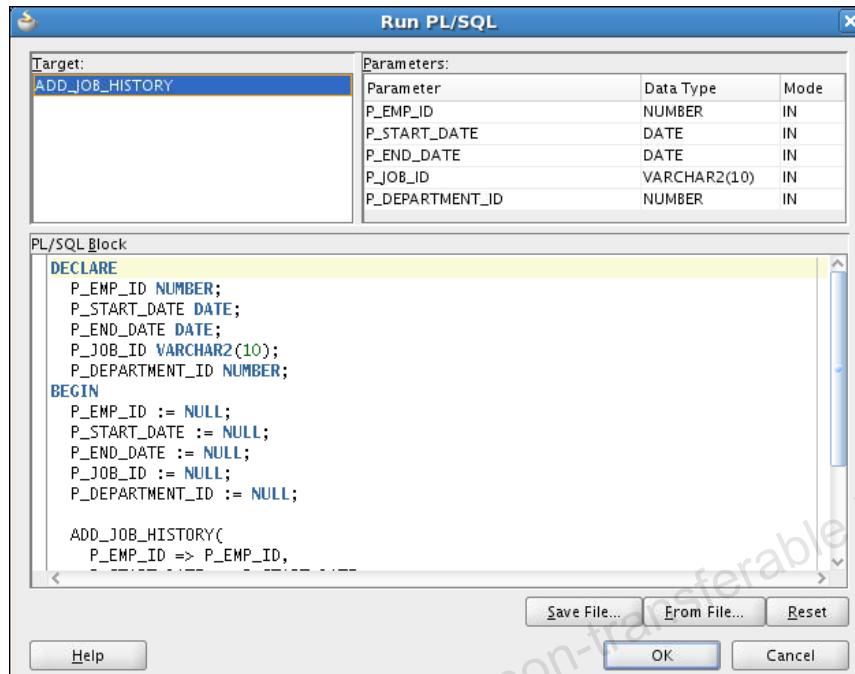
ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 编译

编辑骨架定义之后，需要编译程序单元。在“Connection Navigator（连接导航器）”中右键单击需要编译的PL/SQL对象，然后选择“Compile（编译）”。也可以按Ctrl + Shift + F9进行编译。

## 运行程序单元



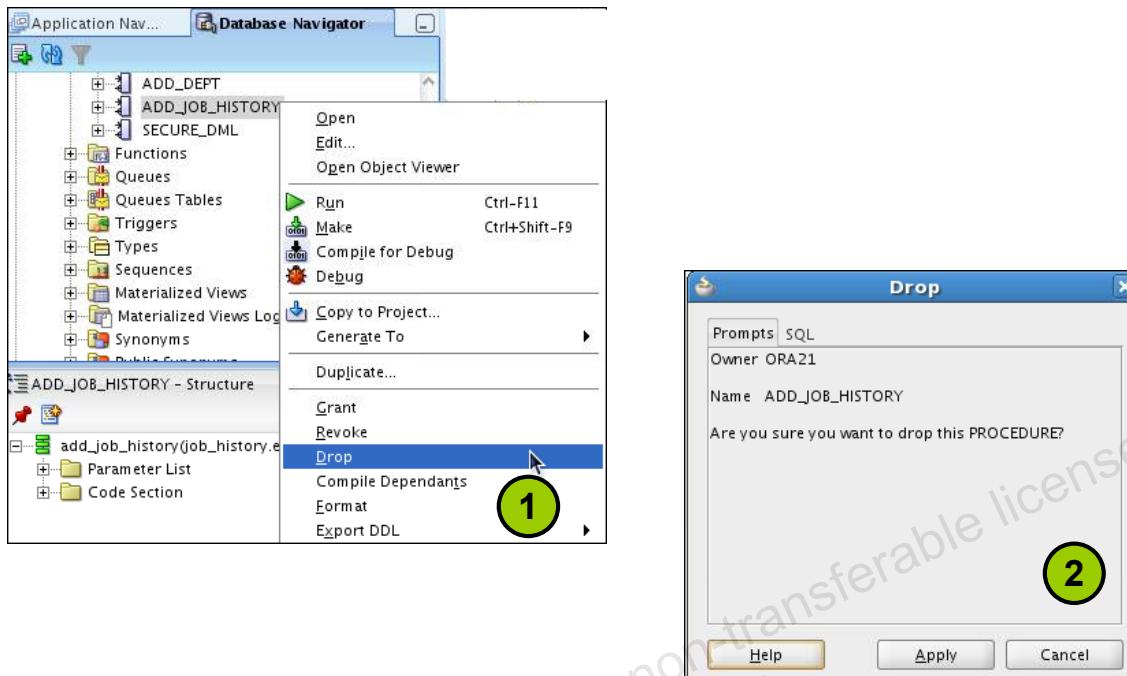
**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 运行程序单元

要执行程序单元，右键单击对象，再选择“Run（运行）”。此时将出现“Run PL/SQL（运行 PL/SQL）”对话框。可能需要将 NULL 值更改为可以传递到程序单元的合理值。更改这些值之后，单击“OK（确定）”。输出结果将显示在“Message-Log（消息日志）”窗口中。

# 删除程序单元



ORACLE

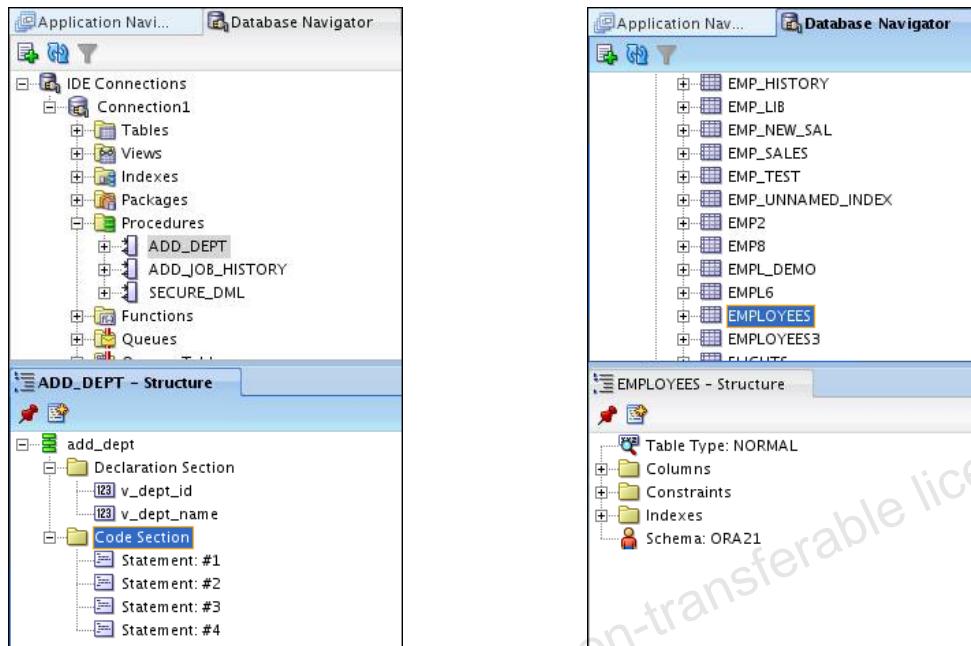
版权所有 © 2010, Oracle。保留所有权利。

## 删除程序单元

要删除程序单元，请执行以下步骤：

1. 右键单击对象并选择“Drop（删除）”。  
此时将出现“Drop Confirmation（删除确认）”对话框。
2. 单击“Apply（应用）”。  
该对象将从数据库中删除。

## 结构窗口



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

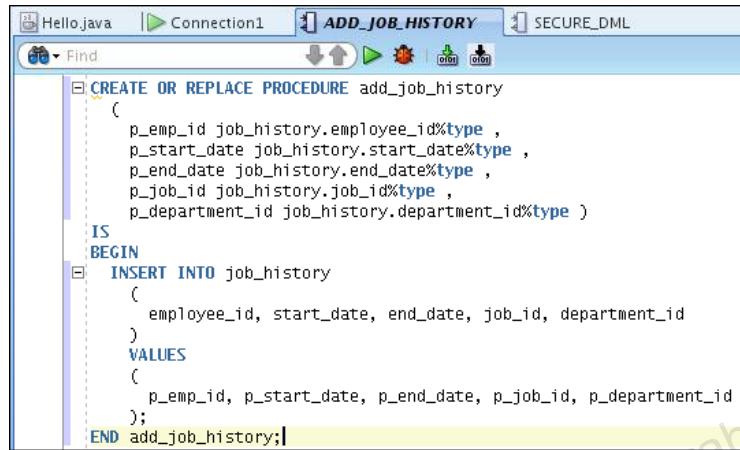
### 结构窗口

“Structure（结构）”窗口提供在活动窗口中当前选择的文档中数据的结构视图，该活动窗口可以是下列参与提供结构的窗口之一：导航器、编辑器、查看器和属性检查器。

在“Structure（结构）”窗口中，可以用多种方式查看文档数据。可显示的结构取决于文档类型。对于 Java 文件，可以查看代码结构、UI 结构或 UI 模型数据。对于 XML 文件，可以查看 XML 结构、设计结构或 UI 模型数据。

就当前活动编辑器而言，“Structure（结构）”窗口是动态的，始终跟踪活动窗口中的当前选项（除非在特定视图上冻结该窗口内容）。如果当前选择的对象是导航器中的节点，则假定为默认编辑器。要为当前选择更改结构视图，请选择不同的结构选项卡。

## 编辑器窗口



版权所有 © 2010, Oracle。保留所有权利。

ORACLE

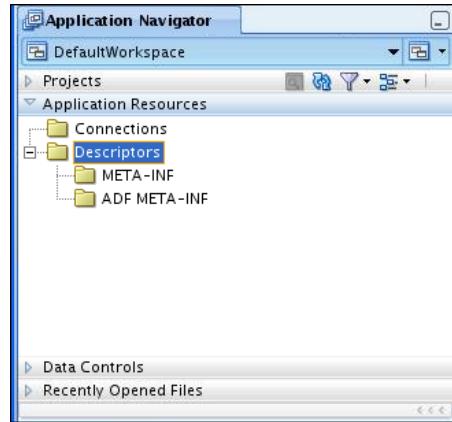
## 编辑器窗口

可以在一个单一编辑器窗口中查看所有项目文件，可以打开同一文件的多个视图，还可以打开不同文件的多个视图。

编辑器窗口顶部的选项卡是文档选项卡。选择一个文档选项卡将突出显示该文件，使其显示在当前编辑器中窗口的前景中。

对于给定文件，编辑器窗口底部的选项卡是各编辑器选项卡。选择一个编辑器选项卡将在该编辑器中打开文件。

## 应用程序导航器



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

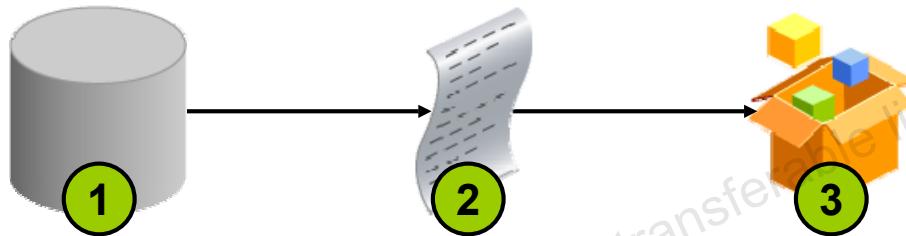
### 应用程序导航器

应用程序导航器提供了应用程序及其所包含数据的逻辑视图。应用程序导航器提供了一个基础结构，可以在该结构中插入各种扩展并使用它以一致而抽象的方式组织这些扩展的数据和菜单。尽管应用程序导航器可以包含单个文件（如 Java 源文件），但它是为合并复杂数据而设计的。在此导航器中，复杂数据类型（如实体对象、统一建模语言 (UML) 图表、Enterprise JavaBeans (EJB) 或 Web 服务）显示为单一节点。组成这些抽象节点的裸文件显示在“Structure (结构)”窗口中。

## 部署 Java 存储过程

在部署 Java 存储过程之前，请执行以下步骤：

1. 创建数据库连接。
2. 创建部署概要文件。
3. 部署对象。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 部署 Java 存储过程

创建 Java 存储过程的部署概要文件，然后使用概要文件中的设置在 JDeveloper 中部署类以及（可选）任何公共静态方法。

向数据库进行部署应使用部署概要文件向导提供的信息和以下两个 Oracle DB 实用程序：

- `loadjava` 将包含存储过程的 Java 类加载到 Oracle DB。
- `publish` 可为已加载的公共静态方法生成特定于 PL/SQL 调用的包装。发布功能可使 Java 方法作为 PL/SQL 函数或过程被调用。

## 将 Java 发布到 PL/SQL

The screenshot shows two windows of Oracle SQL Developer. The top window, titled 'TrimLob.java', displays Java code for a class named TrimLob with a main method. A green circle labeled '1' highlights the connection logic. The bottom window, titled 'TRIMLOBPROC', displays PL/SQL code for creating a procedure named TRIMLOBPROC that calls the Java class. A green circle labeled '2' highlights the procedure creation command. A watermark across the slide reads: 'This document has a non-transferable license to use' and '华周 (2002-2014) has a non-transferable license to use this Student Guide.'

```
public class TrimLob {
 public static void main (String args []) throws SQLException {
 Connection conn=null;
 if (System.getProperty("oracle.jserver.version") != null)
 {
 conn = DriverManager.getConnection("jdbc:default:connection:");
 }
 else
 {
 DriverManager.registerDriver(new oracle.jdbc.OracleDriver());
 conn = DriverManager.getConnection("jdbc:oracle:thin:scott/tiger");
 }
 }
}
```

```
CREATE OR REPLACE PROCEDURE TRIMLOBPROC
as Language java
name 'TrimLob.main(java.lang.String[])';
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 将 Java 发布到 PL/SQL

该幻灯片显示了 Java 代码并说明如何在 PL/SQL 过程中发布 Java 代码。

## 如何了解有关 JDeveloper 11g 的更多信息

| 主题                            | 网站                                                                                                                                      |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Oracle JDeveloper<br>产品页      | <a href="http://www.oracle.com/technology/products/jdev/index.html">http://www.oracle.com/technology/products/jdev/index.html</a>       |
| Oracle JDeveloper 11g<br>教程   | <a href="http://www.oracle.com/technology/obe/obe11jdev/11/index.html">http://www.oracle.com/technology/obe/obe11jdev/11/index.html</a> |
| Oracle JDeveloper 11g<br>产品文档 | <a href="http://www.oracle.com/technology/documentation/jdev.html">http://www.oracle.com/technology/documentation/jdev.html</a>         |
| Oracle JDeveloper 11g<br>论坛   | <a href="http://forums.oracle.com/forums/forum.jspa?forumID=83">http://forums.oracle.com/forums/forum.jspa?forumID=83</a>               |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 小结

在本附录中，您应该已经学会如何使用 JDeveloper 执行以下任务：

- 列举 Oracle JDeveloper 的主要功能
- 在 JDeveloper 中创建数据库连接
- 在 JDeveloper 中管理数据库对象
- 使用 JDeveloper 执行 SQL 命令
- 创建并运行 PL/SQL 程序单元



版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

本附录将向您介绍一种工具 JDeveloper。您将学习如何使用 JDeveloper 执行数据库开发任务。

通过将相关数据分组来生成报表

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

- 使用 ROLLUP 操作生成小计值
- 使用 CUBE 操作生成交叉报表值
- 使用 GROUPING 函数确定由 ROLLUP 或 CUBE 创建的行值
- 使用 GROUPING SETS 生成单个结果集

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

在本附录中，您将学习如何：

- 使用 ROLLUP 运算符分组数据以获取小计值
- 使用 CUBE 运算符分组数据以获取交叉报表值
- 使用 GROUPING 函数确定由 ROLLUP 或 CUBE 运算符生成的结果集中的汇总级别
- 使用等效于 UNION ALL 方法的 GROUPING SETS 来生成单个结果集

## 复习组函数

- 使用组函数可对行集进行计算，为每个组提供一个结果。

```
SELECT [column,] group_function(column) . . .
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

- 示例：

```
SELECT AVG(salary), STDDEV(salary),
 COUNT(commission_pct), MAX(hire_date)
 FROM employees
 WHERE job_id LIKE 'SA%';
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 组函数

可以使用 GROUP BY 子句将表中的行分成组。然后可以使用组函数返回每个组的汇总信息。组函数可以出现在选择列表以及 ORDER BY 和 HAVING 子句中。Oracle Server 会对每一组行应用组函数，然后为每一组返回单个结果行。

**组函数的类型：**每个组函数（AVG、SUM、MAX、MIN、COUNT、STDDEV 和 VARIANCE）都接受一个参数。AVG、SUM、STDDEV 和 VARIANCE 函数仅对数值进行操作。而 MAX 和 MIN 可以对数字、字符或日期数据值进行操作。COUNT 会为指定的表达式返回非空的行数。幻灯片示例中计算了 JOB\_ID 以 SA 开头的雇员的平均薪金、薪金标准偏差、领取佣金雇员的数量以及最晚受雇日期。

#### 使用组函数的准则

- 参数的数据类型可以为 CHAR、VARCHAR2、NUMBER 或 DATE。
- 除 COUNT(\*) 之外的所有组函数都忽略空值。要用某个值替代空值，请使用 NVL 函数。COUNT 会返回一个数字或零。
- 使用 GROUP BY 子句时，Oracle Server 会隐式按指定分组列的升序对结果集进行排序。要改写这种默认排序方式，可以在 ORDER BY 子句中使用 DESC。

## 复习 GROUP BY 子句

- 语法:

```
SELECT [column,] group_function(column) . . .
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

- 示例:

```
SELECT department_id, job_id, SUM(salary),
 COUNT(employee_id)
FROM employees
GROUP BY department_id, job_id;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 复习 GROUP BY 子句

在幻灯片示例中, Oracle Server 按如下方式进行计算:

- SELECT 子句指定检索下面的列:
  - EMPLOYEES 表中的部门 ID 列和职务 ID 列
  - 计算 GROUP BY 子句指定的每一组中雇员的数量及所有薪金总和
- GROUP BY 子句指定应如何对表行进行分组。计算每一部门内担任同一职务 ID 的雇员的数量及薪金总额。先按部门 ID 对行进行分组, 再在每一个部门内按职务对行进行分组。

## 复习 HAVING 子句

- 使用 HAVING 子句可指定要显示的组。
- 可根据限制条件进一步限制组。

```
SELECT [column,] group_function(column) ...
FROM table
[WHERE condition]
[GROUP BY group_by_expression]
[HAVING having_expression]
[ORDER BY column];
```



版权所有 © 2010, Oracle。保留所有权利。

### HAVING 子句

在对组应用 HAVING 子句之前，需要先分成组，还要使用组函数进行计算。HAVING 子句可放在 GROUP BY 子句之前，但建议将 GROUP BY 子句放在前面，因为这样更符合逻辑。使用 HAVING 子句时，Oracle Server 会执行以下步骤：

1. 对行进行分组。
2. 对这些组应用组函数，然后显示符合 HAVING 子句中的标准的组。

## GROUP BY 与 ROLLUP 和 CUBE 运算符一起使用

- 将 ROLLUP 或 CUBE 与 GROUP BY 一起使用可以通过交叉引用列来生成超级汇总行。
- 使用 ROLLUP 分组时会生成一个结果集，其中包含常规分组行和小计值。
- 使用 CUBE 分组时会生成一个结果集，其中包含来自 ROLLUP 的行和交叉报表行。



版权所有 © 2010, Oracle。保留所有权利。

### GROUP BY 与 ROLLUP 和 CUBE 运算符一起使用

可以在查询使用的 GROUP BY 子句中指定 ROLLUP 和 CUBE 运算符。使用 ROLLUP 分组时会生成一个结果集，其中包含常规分组行和小计行。ROLLUP 运算符还会计算合计。GROUP BY 子句中的 CUBE 操作会根据指定的表达式所有可能组合的值对选定行进行分组，然后为每一组返回一行来提供汇总信息。可以使用 CUBE 运算符生成交叉报表行。

**注：**使用 ROLLUP 和 CUBE 时，请确保跟在 GROUP BY 子句后的各个列有意义，而且彼此之间有实际关系，否则这两个运算符会返回不相干的信息。

## ROLLUP 运算符

- ROLLUP 是对 GROUP BY 子句的扩展。
- 执行 ROLLUP 操作后可生成累计汇总，如小计。

```

SELECT [column,] group_function(column) . . .
FROM table
[WHERE condition]
[GROUP BY [ROLLUP] group_by_expression]
[HAVING having_expression];
[ORDER BY column];

```



版权所有 © 2010, Oracle。保留所有权利。

### ROLLUP 运算符

ROLLUP 运算符为 GROUP BY 语句中的表达式提供了汇总和超级汇总。报表生成程序可以使用 ROLLUP 运算符从结果集中提取统计数据和汇总信息。累计汇总可用于报表、图表和图形中。

ROLLUP 运算符会沿着一个方向创建分组，也就是沿着 GROUP BY 子句中指定的列列表从右向左创建分组。然后对这些分组应用汇总函数。

#### 附注

- 在不使用 ROLLUP 运算符的情况下，要生成  $n$  维（即 GROUP BY 子句中的  $n$  个列）个小计，必须使用 UNION ALL 将  $n+1$  个 SELECT 语句链接起来。这会影响执行查询的效率，因为每个 SELECT 语句都会导致对表进行访问。ROLLUP 运算符仅对表进行一次访问即可收集其结果。如果在生成小计时涉及许多列，则 ROLLUP 运算符很有用。
- 使用 ROLLUP 可以生成小计与总计。CUBE 除生成总计外，还可在每一可能方向上有效地进行累计，从而生成交叉报表数据。

## ROLLUP 运算符：示例

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY ROLLUP(department_id, job_id);
```

|    | DEPARTMENT_ID | JOB_ID      | SUM(SALARY) |
|----|---------------|-------------|-------------|
| 1  |               | 10 AD_ASST  | 4400        |
| 2  |               | 10 (null)   | 4400        |
| 3  |               | 20 MK_MAN   | 13000       |
| 4  |               | 20 MK_REP   | 6000        |
| 5  |               | 20 (null)   | 19000       |
| 6  |               | 30 PU_MAN   | 11000       |
| 7  |               | 30 PU_CLERK | 13900       |
| 8  |               | 30 (null)   | 24900       |
| 9  |               | 40 HR REP   | 6500        |
| 10 |               | 40 (null)   | 6500        |
| 11 |               | 50 ST_MAN   | 36400       |
| 12 |               | 50 SH_CLERK | 64300       |
| 13 |               | 50 ST_CLERK | 55700       |
| 14 |               | 50 (null)   | 156400      |
| 15 | (null) (null) |             | 211200      |

- 1 表示按 DEPARTMENT\_ID 和 JOB\_ID 进行合计的一组。
- 2 表示只按 DEPARTMENT\_ID 进行合计的一组。
- 3 表示合计。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### ROLLUP 运算符示例

在幻灯片示例中：

- 要计算在 GROUP BY 子句中显示的每个部门（这些部门的部门 ID 小于 60）内每个职务 ID 的薪金总额。
- ROLLUP 运算符会显示：
  - 计算部门 ID 小于 60 的每个部门的薪金总额
  - 部门 ID 小于 60 的所有部门的薪金总额，而不考虑职务 ID

在此示例中，1 表示按 DEPARTMENT\_ID 和 JOB\_ID 进行合计的一组，2 表示只按 DEPARTMENT\_ID 进行合计的一组，而 3 表示合计。

当后面跟着 GROUP BY 子句中指定的分组列表时，ROLLUP 运算符会创建从最详细级别到合计级别的累计小计。首先，计算 GROUP BY 子句中指定组的标准汇总值（本示例中，计算某个部门内每一职务组的薪金的总和）。然后，在分组列的列表中从右向左移动，逐一计算更高级别的小计。（本示例中，先计算每个部门的薪金总额，然后计算所有部门的薪金总额。）

- 如果 GROUP BY 子句的 ROLLUP 运算符中有  $n$  个表达式，则运算会产生  $n + 1$ （在本例中， $2 + 1 = 3$ ）个分组。
- 根据前  $n$  个表达式值分组的那些行称作行或常规行，其它行称作超级汇总行。

## CUBE 运算符

- CUBE 是对 GROUP BY 子句的扩展。
- 可以在一条 SELECT 语句中使用 CUBE 运算符来生成交叉报表值。

```

SELECT [column,] group_function(column)...
FROM table
[WHERE condition]
[GROUP BY [CUBE] group_by_expression]
[HAVING having_expression]
[ORDER BY column];

```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### CUBE 运算符

CUBE 运算符是 SELECT 语句的 GROUP BY 子句中的另一个开关。CUBE 运算符可以应用于所有的汇总函数，包括 AVG、SUM、MAX、MIN 和 COUNT。该运算符可用来生成通常在交叉报表中使用的结果集。ROLLUP 只能生成一部分可能组合的小计，而 CUBE 可生成 GROUP BY 子句中指定的所有可能组合的分组的小计以及合计。

CUBE 运算符与汇总函数一起使用时会在结果集中生成附加行。对 GROUP BY 子句中包含的列进行交叉引用可生成组的超集。对这些组应用选择列表中指定的汇总函数，就会生成附加超级汇总行的汇总值。结果集中附加组的数量取决于 GROUP BY 子句中包含的列的数量。

实际上，GROUP BY 子句中列或表达式的每种可能组合都会生成超级汇总。如果 GROUP BY 子句中有  $n$  个列或表达式，就可能有  $2^n$  个超级汇总组合。从数学角度看，这些组合形成了一个  $n$  维立方体 (cube)，此运算符之名也正是因此而得。

然后通过使用应用程序或编程工具，可将这些超级汇总值输入在图表和图形中，从而在图表和图形上直观有效地呈现这些结果及其关系。

## CUBE 运算符：示例

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id < 60
GROUP BY CUBE (department_id, job_id) ;
```

|    | DEPARTMENT_ID   | JOB_ID | SUM(SALARY) |
|----|-----------------|--------|-------------|
| 1  | (null) (null)   |        | 211200      |
| 2  | (null) HR_REP   |        | 6500        |
| 3  | (null) MK_MAN   |        | 13000       |
| 4  | (null) MK_REP   |        | 6000        |
| 5  | (null) PU_MAN   |        | 11000       |
| 6  | (null) ST_MAN   |        | 36400       |
| 7  | (null) AD_ASST  |        | 4400        |
| 8  | (null) PU_CLERK |        | 13900       |
| 9  | (null) SH_CLERK |        | 64300       |
| 10 | (null) ST_CLERK |        | 55700       |
| 11 | 10 (null)       |        | 4400        |
| 12 | 10 AD_ASST      |        | 4400        |
| 13 | 20 (null)       |        | 19000       |
| 14 | 20 MK_MAN       |        | 13000       |
| 15 | 20 MK_REP       |        | 6000        |
| 16 | 30 (null)       |        | 24900       |

- 1
- 2
- 3
- 4

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### CUBE 运算符示例

示例中 SELECT 语句的输出可如下解释：

- 计算部门 ID 小于 60 的某个部门内每种职务的薪金总额
- 计算部门 ID 小于 60 的每个部门的薪金总额
- 计算每一职务的薪金总额，不考虑所在的部门
- 计算部门 ID 小于 60 的部门的薪金总额，不考虑职务

在本示例中，1 表示合计，2 表示仅按 JOB\_ID 进行合计的那些行，3 表示按 DEPARTMENT\_ID 和 JOB\_ID 进行合计的部分行，而 4 表示仅按 DEPARTMENT\_ID 进行合计的部分行。

CUBE 运算符也会执行 ROLLUP 操作，显示部门 ID 小于 60 的那些部门的小计，以及部门 ID 小于 60 的那些部门的薪金总额，而不考虑职务。此外，CUBE 运算符还会显示每种职务的薪金总额，而不考虑部门。

注：与 ROLLUP 运算符类似，要不使用 CUBE 运算符而生成  $n$  维（即，GROUP BY 子句中有  $n$  个列）个小计，需要通过 UNION ALL 将  $2^n$  个 SELECT 语句链接起来。因此，一个三维报表需要通过 UNION ALL 将  $2^3 = 8$  个 SELECT 语句链接起来。

## GROUPING 函数

GROUPING 函数:

- 与 CUBE 或 ROLLUP 运算符一起使用
- 用于查找在某一行产生小计的组
- 用于区分存储的 NULL 值与 ROLLUP 或 CUBE 创建的 NULL 值
- 返回 0 或 1

```
SELECT [column,] group_function(column) . . ,
 GROUPING(expr)
 FROM table
 [WHERE condition]
 [GROUP BY [ROLLUP][CUBE] group_by_expression]
 [HAVING having_expression]
 [ORDER BY column];
```

版权所有 © 2010, Oracle。保留所有权利。

### GROUPING 函数

GROUPING 函数可与 CUBE 或 ROLLUP 运算符一起使用，有助于了解汇总值是如何获得的。

GROUPING 函数使用一列作为它的参数。GROUPING 函数中的 *expr* 必须与 GROUP BY 子句中的一个表达式相匹配。该函数的返回值是 0 或 1。

GROUPING 函数返回的值可用于:

- 确定指定小计的汇总级别，即，产生小计的一个或多个组。
- 确定结果集行上表达式列中的 NULL 值是:
  - 基表中的 NULL 值（存储的 NULL 值）
  - 由 ROLLUP 或 CUBE 创建的 NULL 值（对该表达式应用组函数的结果）

如果对表达式应用 GROUPING 函数后返回的值为 0，则表示:

- 此表达式已用于计算汇总值。
- 表达式列中的 NULL 值是存储的 NULL 值。

如果对表达式应用 GROUPING 函数后返回的值为 1，则表示:

- 此表达式尚未用于计算汇总值。
- 表达式列中的 NULL 值是由 ROLLUP 或 CUBE 创建的分组结果。

## GROUPING 函数：示例

```

SELECT department_id DEPTID, job_id JOB,
 SUM(salary),
 GROUPING(department_id) GRP_DEPT,
 GROUPING(job_id) GRP_JOB
FROM employees
WHERE department_id < 50
GROUP BY ROLLUP(department_id, job_id);

```

|    | DEPTID        | JOB | SUM(SALARY) | GRP_DEPT | GRP_JOB |
|----|---------------|-----|-------------|----------|---------|
| 1  | 10 AD_ASST    |     | 4400        | 0        | 0       |
| 2  | 10 (null)     |     | 4400        | 0        | 1       |
| 3  | 20 MK_MAN     |     | 13000       | 0        | 0       |
| 4  | 20 MK_REP     |     | 6000        | 0        | 0       |
| 5  | 20 (null)     |     | 19000       | 0        | 1       |
| 6  | 30 PU_MAN     |     | 11000       | 0        | 0       |
| 7  | 30 PU_CLERK   |     | 13900       | 0        | 0       |
| 8  | 30 (null)     |     | 24900       | 0        | 1       |
| 9  | 40 HR_REP     |     | 6500        | 0        | 0       |
| 10 | 40 (null)     |     | 6500        | 0        | 1       |
| 11 | (null) (null) |     | 54800       | 1        | 1       |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### GROUPING 函数示例

在幻灯片示例中，请看一下第一行中的汇总值 4400（标记为 1）。此汇总值是部门 10 内职务 ID AD\_ASST 的薪金总额。为了计算此汇总值，DEPARTMENT\_ID 和 JOB\_ID 列都在考虑范围内。因此，GROUPING(department\_id) 和 GROUPING(job\_id) 两个表达式返回的值都是 0。

请看一下第二行中的汇总值 4400（标记为 2）。此值是部门 10 的薪金总额，在计算时已将 DEPARTMENT\_ID 列考虑在内，因此，GROUPING(department\_id) 返回的值为 0。因为计算此值时没有考虑 JOB\_ID 列，所以 GROUPING(job\_id) 返回的值为 1。您可以在第 5 行中看到类似的输出。

在最后一行中，请看一下汇总值 54800（标记为 3）。这是部门 ID 小于 50 的那些部门中所有职务的薪金总额。要计算此汇总值，不要考虑 DEPARTMENT\_ID 和 JOB\_ID 两列。因此，GROUPING(department\_id) 和 GROUPING(job\_id) 两个表达式返回的值都是 1。

## GROUPING SETS

- GROUPING SETS 语法用于定义同一查询中的多个分组。
- 对 GROUPING SETS 子句中指定的所有分组进行计算并使用 UNION ALL 操作合并各个分组的结果。
- GROUPING SETS 的功能：
  - 只需要对基表执行一次操作即可完成分组计算。
  - 无需编写复杂的 UNION 语句。
  - GROUPING SETS 子句包含的元素越多，性能就越好。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### GROUPING SETS

GROUPING SETS 是对 GROUP BY 子句的进一步扩展，可以用来指定多个数据分组。这样做有利于有效地进行汇总，从而有利于对多维数据进行分析。

现在使用 GROUPING SETS 可以编写一条 SELECT 语句来指定各种分组（也可以包含 ROLLUP 或 CUBE 运算符），而不必通过 UNION ALL 运算符组合多条 SELECT 语句。例如：

```
SELECT department_id, job_id, manager_id, AVG(salary)
 FROM employees
 GROUP BY
 GROUPING SETS
 ((department_id, job_id, manager_id),
 (department_id, manager_id), (job_id, manager_id));
```

此语句计算三个分组的汇总：

```
(department_id, job_id, manager_id), (department_id,
manager_id) and (job_id, manager_id)
```

如果没有使用此功能，则需要使用 UNION ALL 将多个查询组合在一起，才能获得上述 SELECT 语句的输出。多查询方法因需要对同一数据进行多次扫描而导致效率较低。

## GROUPING SETS (续)

将前面的示例与以下语句进行比较：

```
SELECT department_id, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY CUBE(department_id, job_id, manager_id);
```

此语句计算了所有  $8 (2 * 2 * 2)$  个分组，尽管您只关心 (department\_id, job\_id, manager\_id)、(department\_id, manager\_id) 和 (job\_id, manager\_id) 组。

下面的语句是另一种可选方法：

```
SELECT department_id, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY department_id, job_id, manager_id
UNION ALL
SELECT department_id, NULL, manager_id, AVG(salary)
FROM employees
GROUP BY department_id, manager_id
UNION ALL
SELECT NULL, job_id, manager_id, AVG(salary)
FROM employees
GROUP BY job_id, manager_id;
```

此语句需要对基表进行三次扫描，因而效率较低。

CUBE 和 ROLLUP 可视为具有特定语义和结果的分组集。下面的等效语句说明了这一点：

|                        |                                                                            |
|------------------------|----------------------------------------------------------------------------|
| CUBE(a, b, c)<br>等同于   | GROUPING SETS<br>((a, b, c), (a, b), (a, c), (b, c),<br>(a), (b), (c), ()) |
| ROLLUP(a, b, c)<br>等同于 | GROUPING SETS ((a, b, c), (a, b), (a), ())                                 |

## GROUPING SETS: 示例

```
SELECT department_id, job_id,
 manager_id, AVG(salary)
FROM employees
GROUP BY GROUPING SETS
 ((department_id,job_id), (job_id,manager_id));
```

|   | DEPARTMENT_ID | JOB_ID   | MANAGER_ID | Avg(SALARY) |
|---|---------------|----------|------------|-------------|
| 1 | (null)        | SH_CLERK | 122        | 3200        |
| 2 | (null)        | AC_MGR   | 101        | 12000       |
| 3 | (null)        | ST_MAN   | 100        | 7280        |
| 4 | ...           | ST_CLERK | 121        | 2675        |

1

|    | DEPARTMENT_ID | JOB_ID  | MANAGER_ID | Avg(SALARY) |
|----|---------------|---------|------------|-------------|
| 39 | 110           | AC_MGR  | (null)     | 12000       |
| 40 | 90            | AD_PRES | (null)     | 24000       |
| 41 | 60            | IT_PROG | (null)     | 5760        |
| 42 | 100           | FL_MGR  | (null)     | 12000       |

2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### GROUPING SETS: 示例

在幻灯片的查询中计算了两个分组的汇总。此表已分为以下组：

- 部门 ID、职务 ID。
- 职务 ID、经理 ID。

计算这些组中每一组的平均薪金。结果集中显示两个组中每一组的平均薪金。

在输出中，标记为 1 的组可以解释为：

- 职务 ID 为 SH\_CLERK 且经理为 122 的所有雇员的平均薪金为 3,200。
- 职务 ID 为 AC\_MGR 且经理为 101 的所有雇员的平均薪金为 12,000，依次类推。

在输出中，标记为 2 的组可以解释为：

- 部门 110 中职务 ID 为 AC\_MGR 的所有雇员的平均薪金为 12,000。
- 部门 90 中职务 ID 为 AD\_PRES 的所有雇员的平均薪金为 24,000，依此类推。

### GROUPING SETS: 示例 (续)

幻灯片示例还可以编写为：

```
SELECT department_id, job_id, NULL as manager_id,
 AVG(salary) as AVGSAL
 FROM employees
 GROUP BY department_id, job_id
UNION ALL
SELECT NULL, job_id, manager_id, avg(salary) as AVGSAL
 FROM employees
 GROUP BY job_id, manager_id;
```

如果优化程序没有检查查询块来生成执行计划，上述查询需要对基表 EMPLOYEES 进行两次扫描。这导致效率非常低。因此，建议使用 GROUPING SETS 语句。

## 组合列

- 组合列是可视为一个单元的多个列的集合。  
ROLLUP (a, (b, c), d)
- 在 GROUP BY 子句内使用括号对某些列进行分组，以便在执行 ROLLUP 或 CUBE 操作时将这些列视为一个单元。
- 在与 ROLLUP 或 CUBE 一起使用时，组合列需要跳过某些级别的汇总。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 组合列

组合列是进行分组计算时可视为一个单元的某些列的集合。可以在括号中指定列，如以下语句所示：ROLLUP (a, (b, c), d)

此处，(b, c) 形成一个组合列且可视为一个单元。通常，组合列在 ROLLUP、CUBE 和 GROUPING SETS 中很有用。例如，在 CUBE 或 ROLLUP 中，组合列需要跳过某些级别的汇总。

也就是说，GROUP BY ROLLUP(a, (b, c)) 等效于：

```
GROUP BY a, b, c UNION ALL
GROUP BY a UNION ALL
GROUP BY ()
```

此处，(b, c) 被视为一个单元，因此不会对 (b, c) 应用 ROLLUP。就好像 (b, c) 有一个别名 z 一样，GROUP BY 表达式可简化为：GROUP BY ROLLUP(a, z)。

**注：**GROUP BY() 通常是一条 SELECT 语句，其中 a 和 b 两列为 NULL 值，只是一个汇总函数。它一般用于生成合计。

```
SELECT NULL, NULL, aggregate_col
FROM <table_name>
GROUP BY ();
```

## 组合列（续）

将其与常规 ROLLUP 进行比较：

```
GROUP BY ROLLUP(a, b, c)
```

此语句等同于：

```
GROUP BY a, b, c UNION ALL
GROUP BY a, b UNION ALL
GROUP BY a UNION ALL
GROUP BY ()
```

类似地：

```
GROUP BY CUBE((a, b), c)
```

此语句等同于：

```
GROUP BY a, b, c UNION ALL
GROUP BY a, b UNION ALL
GROUP BY c UNION ALL
GROUP BY ()
```

下表显示 GROUPING SETS 规范以及等效的 GROUP BY 规范。

| GROUPING SETS 语句                                                           | 等效的 GROUP BY 语句                                               |
|----------------------------------------------------------------------------|---------------------------------------------------------------|
| GROUP BY GROUPING SETS(a, b, c)                                            | GROUP BY a UNION ALL<br>GROUP BY b UNION ALL<br>GROUP BY c    |
| GROUP BY GROUPING SETS(a, b, (b, c))<br>(GROUPING SETS 表达式含有一个组合列。)        | GROUP BY a UNION ALL<br>GROUP BY b UNION ALL<br>GROUP BY b, c |
| GROUP BY GROUPING SETS((a, b, c))                                          | GROUP BY a, b, c                                              |
| GROUP BY GROUPING SETS(a, (b), ())                                         | GROUP BY a UNION ALL<br>GROUP BY b UNION ALL<br>GROUP BY ()   |
| GROUP BY GROUPING SETS<br>(a, ROLLUP(b, c))<br>(GROUPING SETS 表达式含有一个组合列。) | GROUP BY a UNION ALL<br>GROUP BY ROLLUP(b, c)                 |

## 组合列：示例

```
SELECT department_id, job_id, manager_id,
 SUM(salary)
 FROM employees
 GROUP BY ROLLUP(department_id, (job_id, manager_id));
```

The diagram illustrates the execution of a ROLLUP query on the employees table. It shows two result sets. The first result set has 7 rows, and the second has 6 rows. Green circles numbered 1 through 4 point to specific rows in each set, indicating the levels of grouping:

|   | DEPARTMENT_ID | JOB_ID        | MANAGER_ID | SUM(SALARY) |
|---|---------------|---------------|------------|-------------|
| 1 | 1             | (null) SA_REP | 149        | 7000        |
| 2 | 2             | (null) (null) | (null)     | 7000        |
| 3 | 3             | 10 AD_ASST    | 101        | 4400        |
| 4 | 4             | 10 (null)     | (null)     | 4400        |
| 5 | 5             | 20 MK_MAN     | 100        | 13000       |
| 6 | 6             | 20 MK_REP     | 201        | 6000        |
| 7 | 7             | 20 (null)     | (null)     | 19000       |

|    | DEPARTMENT_ID | JOB_ID         | MANAGER_ID | SUM(SALARY) |
|----|---------------|----------------|------------|-------------|
| 40 | 40            | 100 FI_MGR     | 101        | 12000       |
| 41 | 41            | 100 FI_ACCOUNT | 108        | 39600       |
| 42 | 42            | 100 (null)     | (null)     | 51600       |
| 43 | 43            | 110 AC_MGR     | 101        | 12000       |
| 44 | 44            | 110 AC_ACCOUNT | 205        | 8300        |
| 45 | 45            | 110 (null)     | (null)     | 20300       |
| 46 | 46            | (null) (null)  | (null)     | 691400      |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 组合列：示例

请考虑以下示例：

```
SELECT department_id, job_id, manager_id, SUM(salary)
 FROM employees
 GROUP BY ROLLUP(department_id, job_id, manager_id);
```

此查询导致 Oracle Server 对以下分组进行计算：

- (job\_id, manager\_id)
- (department\_id, job\_id, manager\_id)
- (department\_id)
- 合计

如果只对特定组感兴趣，则必须使用组合列，才能限制只对这些分组进行计算。使用组合列时，可在累计时将 JOB\_ID 和 MANAGER\_ID 列视为一个单元。计算 ROLLUP 和 CUBE 时，括号中的多个列被视为一个单元。幻灯片示例中说明了这种情况。将 JOB\_ID 和 MANAGER\_ID 列放在括号中后，就会指示 Oracle Server 将 JOB\_ID 和 MANAGER\_ID 视为一个单元，即一个组合列。

## 组合列：示例（续）

幻灯片示例中对以下分组进行了计算：

- (department\_id, job\_id, manager\_id)
- (department\_id)
- ()

幻灯片示例显示了下列内容：

- 职务和经理的每个组合的薪金总额（标记为 1）
- 部门、职务和经理的每个组合的薪金总额（标记为 2）
- 每个部门的薪金总额（标记为 3）
- 合计（标记为 4）

幻灯片示例还可以编写为：

```
SELECT department_id, job_id, manager_id, SUM(salary)
 FROM employees
 GROUP BY department_id, job_id, manager_id
UNION ALL
SELECT department_id, TO_CHAR(NULL), TO_NUMBER(NULL),
 SUM(salary)
 FROM employees
 GROUP BY department_id
UNION ALL
SELECT TO_NUMBER(NULL), TO_CHAR(NULL), TO_NUMBER(NULL),
 SUM(salary)
 FROM employees
 GROUP BY ();
```

如果优化程序没有检查查询块来生成执行计划，上述查询需要对基表 EMPLOYEES 进行三次扫描。这导致效率非常低。因此，建议使用组合列。

## 级联分组

- 级联分组提供了一种简单方法来生成有用的组合分组。
- 要指定级联分组集，可以用逗号分隔多个分组集、ROLLUP 和 CUBE 操作，以便 Oracle Server 将这些操作组合在一条 GROUP BY 子句中。
- 结果是每个 GROUPING SET 中的交叉单元分组。

```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```



版权所有 © 2010, Oracle。保留所有权利。

### 级联分组

级联分组提供了一种简单方法来生成有用的组合分组。通过列出多个分组集、CUBE 和 ROLLUP 并用逗号进行分隔，即可指定级联分组。下面是一个级联分组集示例：

```
GROUP BY GROUPING SETS(a, b), GROUPING SETS(c, d)
```

此 SQL 示例定义了下列分组：

(a, c)、(a, d)、(b, c)、(b, d)

级联分组集极为有用，原因如下：

- **便于开发查询：**无需手动列举所有分组。
- **可由应用程序使用：**由联机分析处理 (OLAP) 应用程序生成的 SQL 通常包括级联分组集，每个 GROUPING SET 都定义一个维所需要的分组。

## 级联分组：示例

```
SELECT department_id, job_id, manager_id,
 SUM(salary)
FROM employees
GROUP BY department_id,
 ROLLUP(job_id),
 CUBE(manager_id);
```

|   | DEPARTMENT_ID | JOB_ID | MANAGER_ID | SUM(SALARY) |
|---|---------------|--------|------------|-------------|
| 1 | (null) SA_REP | 149    | 7000       |             |
| 2 | 10 AD_ASST    | 101    | 4400       |             |
| 3 | 20 MK_MAN     | 100    | 13000      |             |
| 4 | 20 MK_REP     | 201    | 6000       |             |
|   | ...           |        |            |             |
| 1 | 90 AD_VP      | 100    | 34000      |             |
|   | 90 AD_PRES    | (null) | 24000      |             |
|   | ...           |        |            |             |
|   | (null) SA_REP | (null) | 7000       |             |
|   | 10 AD_ASST    | (null) | 4400       |             |
|   | ...           |        |            |             |
| 2 | 110 (null)    | 101    | 12000      |             |
|   | 110 (null)    | 205    | 8300       |             |
|   | 110 (null)    | (null) | 20300      |             |
| 3 |               |        |            |             |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 级联分组：示例

幻灯片示例中生成了以下分组：

- (department\_id, job\_id,) (1)
- (department\_id, manager\_id) (2)
- (department\_id) (3)

计算了这些组中每一组的薪金总额。

以下为级联分组的另一个示例。

```
SELECT department_id, job_id, manager_id, SUM(salary) totsal
FROM employees
WHERE department_id<60
GROUP BY GROUPING SETS(department_id),
GROUPING SETS (job_id, manager_id);
```

## 小结

在本附录中，您应该已经学会如何：

- 使用 ROLLUP 操作生成小计值
- 使用 CUBE 操作生成交叉报表值
- 使用 GROUPING 函数确定由 ROLLUP 或 CUBE 创建的行值
- 使用 GROUPING SETS 语法定义同一查询中的多个分组
- 使用 GROUP BY 子句以不同的方式组合表达式：
  - 组合列
  - 级联分组集

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

- ROLLUP 和 CUBE 是对 GROUP BY 子句的扩展。
- ROLLUP 用于显示小计与合计值。
- CUBE 用于显示交叉报表值。
- 使用 GROUPING 函数可以确定是否由 CUBE 或 ROLLUP 运算符生成汇总行。
- 使用 GROUPING SETS 语法时，可以在同一查询中定义多个分组。GROUP BY 会计算所有指定分组并使用 UNION ALL 对其进行组合。
- 在 GROUP BY 子句中，可以用不同的方式组合表达式：
  - 要指定组合列，可以将某些列作为一组放在括号中，这样 Oracle Server 在执行 ROLLUP 或 CUBE 操作时会将这些列视为一个单元。
  - 要指定级联分组集，可以用逗号分隔多个分组集、ROLLUP 和 CUBE 操作，以便 Oracle Server 将这些操作组合在一条 GROUP BY 子句中。结果是每个分组集中的交叉单元分组。



分层检索

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

- 解释分层查询的概念
- 创建树状结构报表
- 设置分层数据的格式
- 去除树状结构中的分支

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

在本附录中，您将学习如何使用分层查询创建树状结构的报表。

## EMPLOYEES 表中的示例数据

|   | EMPLOYEE_ID | LAST_NAME | JOB_ID  | MANAGER_ID |
|---|-------------|-----------|---------|------------|
| 1 | 100         | King      | AD_PRES | (null)     |
| 2 | 101         | Kochhar   | AD_VP   | 100        |
| 3 | 102         | De Haan   | AD_VP   | 100        |
| 4 | 103         | Hunold    | IT_PROG | 102        |
| 5 | 104         | Ernst     | IT_PROG | 103        |
| 6 | 107         | Lorentz   | IT_PROG | 103        |

...

|    |     |           |            |     |
|----|-----|-----------|------------|-----|
| 16 | 200 | VWhalen   | AD_ASST    | 101 |
| 17 | 201 | Hartstein | MK_MAN     | 100 |
| 18 | 202 | Fay       | MK_REP     | 201 |
| 19 | 205 | Higgins   | AC_MGR     | 101 |
| 20 | 206 | Gietz     | AC_ACCOUNT | 205 |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

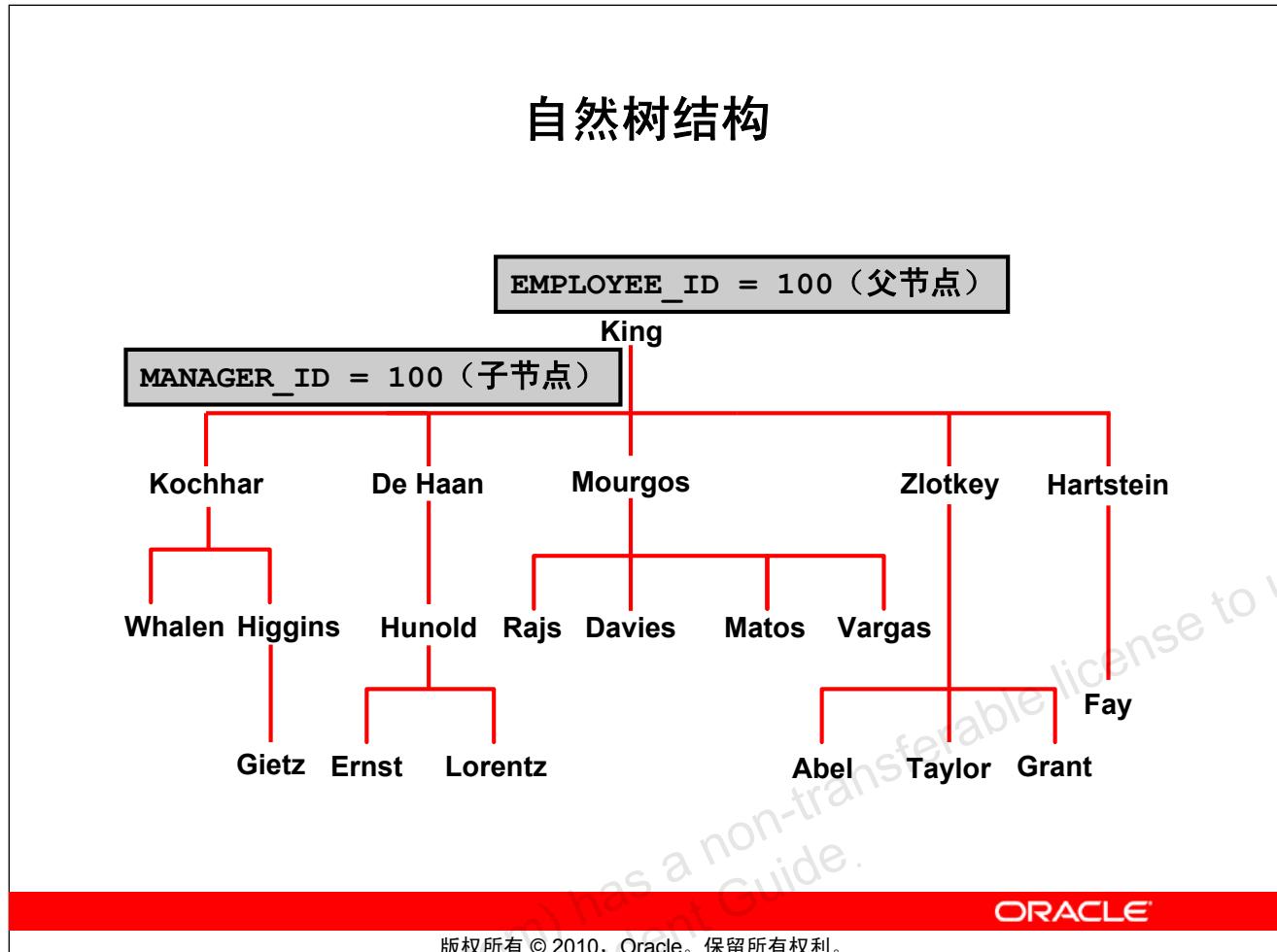
## EMPLOYEES 表中的示例数据

使用分层查询时，可以根据表行之间的自然层次关系检索数据。关系数据库并不采用分层方式存储记录。但是，如果一个表中各行之间存在层次关系，则可以使用名为树遍历 (*tree walking*) 的进程构建层次结构。分层查询是一种以特定顺序显示树分支的报告方法。

例如一个家族树，最年长的族成员在树基或树干附近，而最年轻的成员在树枝上。树枝还可以有自己的分支，依此类推。

当表中各行之间存在某种关系时，就可以进行分层查询。例如，在幻灯片中，您看到 Kochhar、De Haan 和 Hartstein 向 MANAGER\_ID 100（这是 King 的 EMPLOYEE\_ID）报告。

**注：**层次树会用于各种领域，如家谱（家族树）、家畜（出于繁殖目的）、公司管理（管理层次结构）、制造业（产品装配）、进化研究（物种繁衍）和科学的研究。



### 自然树结构

EMPLOYEES 表有一个树状结构，用来呈现管理报告路线。通过寻找 EMPLOYEE\_ID 和 MANAGER\_ID 两列中相等值之间存在的关系，就可以创建层次结构。通过在该表中建立连接，就可以显示这种关系。MANAGER\_ID 列提供了雇员经理的雇员编号。

利用树结构中的父子关系，可以控制：

- 遍历层次结构的方向
- 层次结构中的起始点

注：本幻灯片中显示 EMPLOYEES 表中雇员的倒置树状管理层次结构。

# 分层查询

```
SELECT [LEVEL], column, expr...
FROM table
[WHERE condition(s)]
[START WITH condition(s)]
[CONNECT BY PRIOR condition(s)];
```

条件:

```
expr comparison_operator expr
```

版权所有 © 2010, Oracle。保留所有权利。

## 关键字和子句

存在 CONNECT BY 和 START WITH 子句时确定了要进行分层查询。

在该语法中:

|            |                                                     |
|------------|-----------------------------------------------------|
| SELECT     | 是标准的 SELECT 子句。                                     |
| LEVEL      | 如果分层查询返回的每一行是根行，则 LEVEL 伪列返回 1，如果是根行的子行，则返回 2，依此类推。 |
| FROM table | 指定包含列的表、视图或快照。可以只从一个表中进行选择。                         |
| WHERE      | 限制查询返回的行，而不影响层次结构中的其它行。                             |
| condition  | 是表达式中的比较条件。                                         |
| START WITH | 指定层次结构的根行（起始位置）。需要有此子句才能执行正确的分层查询。                  |
| CONNECT BY | 指定父行和子行之间存在 PRIOR 关系的列。需要有此子句才能执行分层查询。              |

# 遍历树

## 起始点

- 指定必须满足的条件
- 接受任何有效的条件

```
START WITH column1 = value
```

使用 EMPLOYEES 表，从姓氏为 Kochhar 的雇员开始。

```
...START WITH last_name = 'Kochhar'
```



版权所有 © 2010, Oracle。保留所有权利。

## 遍历树

START WITH 子句确定了作为树根的行。START WITH 子句可以包含任何有效条件。

### 示例

使用 EMPLOYEES 表，从公司总裁 King 开始。

```
... START WITH manager_id IS NULL
```

使用 EMPLOYEES 表，从雇员 Kochhar 开始。START WITH 条件可以包含子查询。

```
... START WITH employee_id = (SELECT employee_id
 FROM employees
 WHERE last_name = 'Kochhar')
```

如果省略了 START WITH 子句，则树遍历从表中所有根行开始。

**注：**CONNECT BY 和 START WITH 子句不是符合美国国家标准协会 (ANSI) SQL 标准的子句。

# 遍历树

```
CONNECT BY PRIOR column1 = column2
```

使用 EMPLOYEES 表，自上而下进行遍历。

```
... CONNECT BY PRIOR employee_id = manager_id
```

## 方向

自上而下 → Column1 = 父关键字  
 Column2 = 子关键字

自下而上 → Column1 = 子关键字  
 Column2 = 父关键字

版权所有 © 2010, Oracle。保留所有权利。

## 遍历树（续）

查询的方向由 CONNECT BY PRIOR 列的位置确定。对于自上而下，PRIOR 运算符会引用父行。对于自下而上，PRIOR 运算符会引用子行。要找到父行的子行，Oracle Server 会对父行执行 PRIOR 表达式计算，并对表中的每一行执行其它表达式计算。条件为真的那些行就是父行的子行。Oracle Server 始终通过对当前父行执行 CONNECT BY 条件计算来选择子行。

### 示例

使用 EMPLOYEES 表自上而下进行遍历。定义父行的 EMPLOYEE\_ID 值与子行的 MANAGER\_ID 值相等的层次结构关系。

```
... CONNECT BY PRIOR employee_id = manager_id
```

使用 EMPLOYEES 表自下而上进行遍历：

```
... CONNECT BY PRIOR manager_id = employee_id
```

PRIOR 运算符不必紧跟在 CONNECT BY 之后。因此，下列 CONNECT BY PRIOR 子句给出的结果与上一个示例的结果相同。

```
... CONNECT BY employee_id = PRIOR manager_id
```

注：CONNECT BY 子句不能包含子查询。

## 遍历树：自下而上

```
SELECT employee_id, last_name, job_id, manager_id
FROM employees
START WITH employee_id = 101
CONNECT BY PRIOR manager_id = employee_id ;
```

|   | EMPLOYEE_ID | LAST_NAME | JOB_ID  | MANAGER_ID |
|---|-------------|-----------|---------|------------|
| 1 | 101         | Kochhar   | AD_VP   | 100        |
| 2 | 100         | King      | AD_PRES | (null)     |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 遍历树：自下而上

幻灯片示例中显示一个从雇员 ID 为 101 的雇员开始的经理列表。

## 遍历树：自上而下

```
SELECT last_name||' reports to '||
PRIOR last_name "Walk Top Down"
FROM employees
START WITH last_name = 'King'
CONNECT BY PRIOR employee_id = manager_id ;
```

| Walk Top Down                  |
|--------------------------------|
| 1 King reports to              |
| 2 King reports to              |
| 3 Kochhar reports to King      |
| 4 Greenberg reports to Kochhar |
| 5 Faviet reports to Greenberg  |

|                                |
|--------------------------------|
| ...                            |
| 105 Grant reports to Zlotkey   |
| 106 Johnson reports to Zlotkey |
| 107 Hartstein reports to King  |
| 108 Fay reports to Hartstein   |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 遍历树：自上而下

自上而下进行遍历时，显示雇员及其经理的姓名。使用雇员 King 作为起始点。只打印一列。

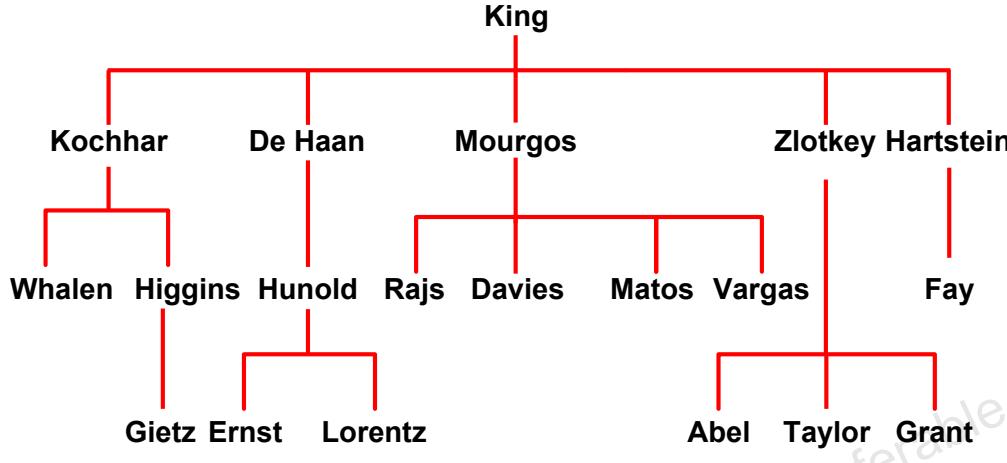
### 示例

在下面的示例中，会计算父行的 EMPLOYEE\_ID 值，计算子行的 MANAGER\_ID 和 SALARY 值。PRIOR 运算符只应用于 EMPLOYEE\_ID 值。

```
... CONNECT BY PRIOR employee_id = manager_id
 AND salary > 15000;
```

要是子行的话，该行的 MANAGER\_ID 值必须等于父行的 EMPLOYEE\_ID 值且 SALARY 值必须大于 \$15,000。

## 使用 LEVEL 伪列确定行的等级



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 LEVEL 伪列确定行的等级

通过使用 LEVEL 伪列，可以在层次结构中明确地显示行的等级或级别。这提高了报表的可读性。从较大分支分出一个或多个分支的叉处被称为节点，每一个分支的末端被称为叶或叶节点。幻灯片图中显示倒置树的节点及其 LEVEL 值。例如，雇员 Higgins 既是父又是子，而雇员 Davies 既是子又是叶。

#### LEVEL 伪列

| 值 | 自上而下的级别      | 自下而上的级别      |
|---|--------------|--------------|
| 1 | 根节点          | 根节点          |
| 2 | 根节点的子节点      | 根节点的父节点      |
| 3 | 子节点的子节点，依此类推 | 父节点的父节点，依此类推 |

在幻灯片中，King 是根或父 (LEVEL = 1)。Kochhar、De Haan、Mourgos、Zlotkey、Hartstein、Higgins 和 Hunold 既是子又是父 (LEVEL = 2)。Whalen、Rajs、Davies、Matos、Vargas、Gietz、Ernst、Lorentz、Abel、Taylor、Grant 和 Fay 是子和叶 (LEVEL = 3 和 LEVEL = 4)。

**注：**在倒置树中根节点是最高节点。子节点是任意非根节点。父节点是拥有子节点的任意节点。叶节点是没有子节点的任意节点。分层查询返回的级别数量可能受用户可用内存大小的限制。

## 使用 LEVEL 和 LPAD 设置分层报表的格式

创建一个显示公司管理级别的报表，从最高级别开始，之下每一个级别依次缩进。

```
COLUMN org_chart FORMAT A12
SELECT LPAD(last_name, LENGTH(last_name)+(LEVEL*2)-2, '_')
 AS org_chart
 FROM employees
START WITH first_name='Steven' AND last_name='King'
CONNECT BY PRIOR employee_id=manager_id
```

版权所有 © 2010, Oracle。保留所有权利。

### 使用 LEVEL 和 LPAD 设置分层报表的格式

树中的各个节点从根节点起都有分配的级别编号。LPAD 函数与 LEVEL 伪列一起使用时，可以显示缩进格式的树状分层报表。

在幻灯片示例中：

- LPAD(*char1, n [, char2]*) 返回 *char1*，而且用 *char2* 中的字符序列从左边填充到长度 *n*。参数 *n* 是显示在终端屏幕上的返回值的总长度。
- LPAD(*last\_name, LENGTH(last\_name)+(LEVEL\*2)-2, '\_'*) 定义显示格式。
- *char1* 是 LAST\_NAME，*n* 是返回值的总长度，也就是 LAST\_NAME+(LEVEL\*2)-2 的长度，而 *char2* 是 '\_'。

换句话说，这会告知 SQL 使用 LAST\_NAME 并在左边填充 '\_' 字符，直至结果字符串的长度等于由 LENGTH(last\_name)+(LEVEL\*2)-2 确定的值为止。

由于 King 的 LEVEL = 1，所以， $(2 * 1) - 2 = 2 - 2 = 0$ 。这样 King 不用填充任何 '\_' 字符且显示在列 1 中。

由于 Kochhar 的 LEVEL = 2，所以， $(2 * 2) - 2 = 4 - 2 = 2$ 。这样 Kochhar 需要填充 2 个 '\_' 字符且缩进显示。

EMPLOYEES 表中其余记录的显示方式都类似。

## 使用 LEVEL 和 LPAD 设置分层报表的格式（续）

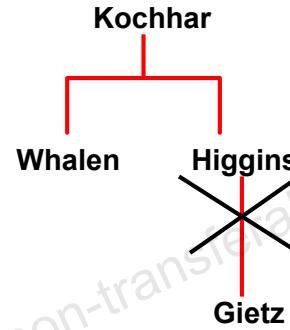
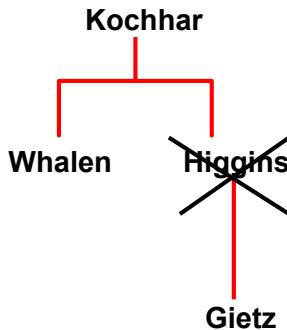
| ORG_CHART |           |
|-----------|-----------|
| 1         | King      |
| 2         | Kochhar   |
| 3         | Greenberg |
| 4         | Faviet    |
| 5         | Chen      |
| 6         | Sciarra   |
| 7         | Urman     |
| 8         | Popp      |
| 9         | Whalen    |
| 10        | Mavris    |
| 11        | Baer      |
| 12        | Higgins   |
| 13        | Gietz     |
| 14        | De Haan   |
| 15        | Hunold    |
| 16        | Ernst     |
| 17        | Austin    |

## 修剪分支

使用 WHERE 子句  
删除一个节点。

使用 CONNECT BY 子句  
删除一个分支。

```
WHERE last_name != 'Higgins' CONNECT BY PRIOR
 employee_id = manager_id
 AND last_name != 'Higgins'
```



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 修剪分支

可以使用 WHERE 和 CONNECT BY 子句来修剪树，也就是控制显示的节点或行。使用的谓词用作一个布尔条件。

#### 示例

从根节点开始，自上而下进行遍历，在结果中删除雇员 Higgins，但是保留其子行。

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
WHERE last_name != 'Higgins'
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id;
```

从根节点开始，自上而下进行遍历，在结果中删除雇员 Higgins 及其所有子行。

```
SELECT department_id, employee_id, last_name, job_id, salary
FROM employees
START WITH manager_id IS NULL
CONNECT BY PRIOR employee_id = manager_id
AND last_name != 'Higgins';
```

## 小结

在本附录中，您应该已经学会：

- 使用分层查询查看表中各行之间的层次关系
- 指定查询的方向和起始点
- 通过修剪来删除节点或分支

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

可以使用分层查询来根据表中各行之间的自然层次关系检索数据。LEVEL 伪列会计算已遍历层次树的级别数。可以使用 CONNECT BY PRIOR 子句指定查询方向。可以使用 START WITH 子句指定起始点。可以使用 WHERE 和 CONNECT BY 子句修剪树分支。

# H

## 编写高级脚本

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

- 说明哪类问题可以用 SQL 生成 SQL 的方法来解决
- 编写一个脚本来生成一个包含 DROP TABLE 语句的脚本
- 编写一个脚本来生成一个包含 INSERT INTO 语句的脚本

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

在本附录中，您将学习如何编写一个 SQL 脚本来生成另一个 SQL 脚本。

## 使用 SQL 生成 SQL

- SQL 可用来生成 SQL 脚本。
- 数据字典：
  - 是包含数据库信息的表和视图的集合
  - 由 Oracle Server 创建和维护



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用 SQL 生成 SQL

SQL 是一种功能强大的工具，可用于生成其它 SQL 语句。在大多数情况下，这涉及编写一个脚本文件。使用 SQL 生成 SQL 可：

- 避免重复编码
- 从数据字典中访问信息
- 删 除或重新创建数据库对象
- 生成包含运行时参数的动态谓词

本附录中使用的示例涉及从数据字典中选择信息。数据字典是包含关于数据库信息的表和视图的集合。这个集合由 Oracle Server 创建和维护。所有数据字典表都归 SYS 用户所有。存储在数据字典中的信息包括 Oracle Server 用户的名称、授予用户的权限、数据库对象名称、表约束条件和审核信息。数据字典视图有四种类别。每种类别都用不同的前缀来表明它的特定用途。

| 前缀    | 说明                              |
|-------|---------------------------------|
| USER_ | 包含用户拥有的对象的详细资料                  |
| ALL_  | 除用户拥有的对象外，还包含用户有访问权限的对象的详细资料    |
| DBA_  | 包含拥有 DBA 权限的用户可访问的数据库中任何对象的详细资料 |
| V\$ _ | 存储有关数据库服务器性能和锁定的信息，只有 DBA 才能使用  |

## 创建基本脚本

```
SELECT 'CREATE TABLE ' || table_name ||
 '_test ' || 'AS SELECT * FROM '
 || table_name ||' WHERE 1=2;'
 AS "Create Table Script"
FROM user_tables;
```

| >Create Table Script                                                    |
|-------------------------------------------------------------------------|
| 1 CREATE TABLE REGIONS_test AS SELECT * FROM REGIONS WHERE 1=2;         |
| 2 CREATE TABLE LOCATIONS_test AS SELECT * FROM LOCATIONS WHERE 1=2;     |
| 3 CREATE TABLE DEPARTMENTS_test AS SELECT * FROM DEPARTMENTS WHERE 1=2; |
| 4 CREATE TABLE JOBS_test AS SELECT * FROM JOBS WHERE 1=2;               |
| 5 CREATE TABLE EMPLOYEES_test AS SELECT * FROM EMPLOYEES WHERE 1=2;     |
| 6 CREATE TABLE JOB_HISTORY_test AS SELECT * FROM JOB_HISTORY WHERE 1=2; |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 基本脚本

幻灯片示例中生成了一个由您所拥有的每个表中的 CREATE TABLE 语句组成的报表。

报表中生成的每一条 CREATE TABLE 语句提供的语法都是使用一个表名加上后缀 \_test 创建该表，而且该表的结构只与相应现有表的结构相同。原有表名是从数据字典视图 USER\_TABLES 的 TABLE\_NAME 列获得的。

下一步需要做的是通过增强报表功能来自动执行该过程。

**注：**可以通过在数据字典表中进行查询可查看所拥有的各种数据库对象。常用的数据字典视图包括：

- USER\_TABLES: 显示用户所拥有表的说明
- USER\_OBJECTS: 显示用户所拥有的所有对象
- USER\_TAB\_PRIVS\_MADE: 显示在用户所拥有对象上授予的所有权限
- USER\_COL\_PRIVS\_MADE: 显示在用户所拥有对象的列上授予的所有权限

## 控制环境

```
SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SQL 语句

SET FEEDBACK ON
SET PAGESIZE 24
SET ECHO ON
```

将系统变量设置为适当的值。

将系统变量重新设置为默认值。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 控制环境

要执行生成的 SQL 语句，必须在一个以后可以运行的文件中保存这些语句。还必须制定计划来清除生成的输出，确保隐藏标题、反馈消息、顶级标题之类的元素。在 SQL Developer 中，可以将这些语句保存在一个脚本中。要保存“Enter SQL Statement（输入 SQL 语句）”框中输入的内容，请单击“Save（保存）”图标或使用“File（文件）> Save（保存）”菜单项。此外，还可以右键单击“Enter SQL Statement（输入 SQL 语句）”框，然后从下拉菜单中选择“Save File（保存文件）”选项。

**注：**SQL 工作表不支持某些 SQL\*Plus 语句。有关 SQL 工作表支持或不支持 SQL\*Plus 语句的完整列表，请参阅 SQL Developer 联机帮助主题“SQL\*Plus Statements Supported and Not Supported in SQL Worksheet（SQL 工作表支持的 SQL\*Plus 语句和不支持的 SQL\*Plus 语句）”。

## 完整脚本

```

SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SELECT 'DROP TABLE ' || object_name || ';'
FROM user_objects
WHERE object_type = 'TABLE'
/

SET FEEDBACK ON
SET PAGESIZE 24
SET ECHO ON

```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 完整脚本

幻灯片中命令的输出被保存在 SQL Developer 中的名为 dropem.sql 的文件中。要在 SQL Developer 中将输出保存在一个文件中，请使用“Script Output（脚本输出）”窗格下的“Save File（保存文件）”选项。此 dropem.sql 文件包含下面的数据。在 SQL Developer 中找到该脚本文件后加载它并执行它，就可以启动该文件。

|   |                               |
|---|-------------------------------|
|   | 'DROPTABLE'  OBJECT_NAME  ';' |
| 1 | DROP TABLE REGIONS;           |
| 2 | DROP TABLE COUNTRIES;         |
| 3 | DROP TABLE LOCATIONS;         |
| 4 | DROP TABLE DEPARTMENTS;       |
| 5 | DROP TABLE JOBS;              |
| 6 | DROP TABLE EMPLOYEES;         |
| 7 | DROP TABLE JOB_HISTORY;       |
| 8 | DROP TABLE JOB_GRADES;        |

## 将表内容转储到文件

```

SET HEADING OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SELECT
 'INSERT INTO departments_test VALUES
 (' || department_id || ', "' || department_name || '
 '", "' || location_id || '")'
 AS "Insert Statements Script"
FROM departments
/

SET PAGESIZE 24
SET HEADING ON ECHO ON FEEDBACK ON

```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 将表内容转储到文件

有时，需要采用 `INSERT INTO VALUES` 语句的形式将一个表中各行上的值保存在一个文本文件中，这会提供帮助。万一不小心删除了表，可以运行此脚本来填写表。

幻灯片示例中生成了针对 `DEPARTMENTS_TEST` 表的一些 `INSERT` 语句，这些语句已保存在 `data.sql` 文件中（在 SQL Developer 中使用“Save File（保存文件）”选项）。

`data.sql` 脚本文件的内容如下：

```

INSERT INTO departments_test VALUES
 (10, 'Administration', 1700);
INSERT INTO departments_test VALUES
 (20, 'Marketing', 1800);
INSERT INTO departments_test VALUES
 (50, 'Shipping', 1500);
INSERT INTO departments_test VALUES
 (60, 'IT', 1400);
...

```

## 将表内容转储到文件

| 源                           | 结果               |
|-----------------------------|------------------|
| ' ''x'''                    | 'x'              |
| ''''                        | '                |
| ''''  department_name  '''' | 'Administration' |
| ''' , '''                   | ', '             |
| ''' ); ''                   | ') ;             |

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 将表内容转储到文件（续）

您可能已经注意到上面的幻灯片中有许多单引号。四个一组的单引号会在最后一条语句中生成一个单引号。另外请记住，字符和日期值必须放在引号中。

在字符串中，要显示一个引号，需要在其前面加上另一个单引号。例如，在幻灯片第 5 个示例中，最外面的两个引号是针对整个字符串的。第 2 个引号充当显示第 3 个引号的前缀。因此，结果是一个单引号，后面跟着括号，再跟一个分号。

## 生成动态谓词

```
COLUMN my_col NEW_VALUE dyn_where_clause

SELECT DECODE(''||deptno', null,
DECODE (''||hiredate', null, ' ',
'WHERE hire_date=TO_DATE(''||'||'||&hiredate'||',''DD-MON-YYYY'))',
DECODE (''||hiredate', null,
'WHERE department_id = ' || '||deptno',
'WHERE department_id = ' || '||deptno' ||
' AND hire_date = TO_DATE(''||'||'||&hiredate'||',''DD-MON-YYYY'))'
AS my_col FROM dual;
```

```
SELECT last_name FROM employees &dyn_where_clause;
```

版权所有 © 2010, Oracle。保留所有权利。

### 生成动态谓词

幻灯片示例中生成了一个 SELECT 语句，用于检索一个部门中受雇日期为特定日期的所有雇员的数据。此脚本会动态生成 WHERE 子句。

**注：**在输入用户变量后，必须使用 UNDEFINE 命令才能删除它。

第一个 SELECT 语句提示您输入部门编号。如果没有输入任何部门编号，则 DECODE 函数会将部门编号当作空值对待，然后提示用户输入受雇日期。如果没有输入任何受雇日期，则 DECODE 函数会将聘用日期当作空值对待，此时生成的动态 WHERE 子句也是空值，这会导致第二条 SELECT 语句检索 EMPLOYEES 表中的所有行。

**注：**NEW\_V[ALUE] 变量会指定一个变量来保存列值。可以在 TTITLE 命令中引用该变量。使用 NEW\_VALUE 可显示列值或顶级标题中的日期。必须在执行 SKIP PAGE 操作的 BREAK 命令中包括该列。变量名不能包含英镑字符 (#)。NEW\_VALUE 对主/从报表都很有用，因为这种报表每一页都有一条新的主记录。

## 生成动态谓词（续）

注：此处输入的受雇日期必须采用 DD-MON-YYYY 格式。

对幻灯片中的 SELECT 语句可以做如下解释：

```

IF (<<deptno>> 未输入) THEN
 IF (<<hiredate>> 未输入) THEN
 返回空字符串
 ELSE
 返回字符串 'WHERE hire_date = TO_DATE('<<hiredate>>',
'DD-MON-YYYY')'
 ELSE
 IF (<<hiredate>> 未输入) THEN
 返回字符串 'WHERE department_id = <<deptno>> 已输入'
 ELSE
 返回字符串 'WHERE department_id = <<deptno>> 已输入
 AND hire_date =
 TO_DATE(' <<hiredate>>', 'DD-MON-YYYY'))'
 END IF
 END IF

```

返回字符串会成为 DYN\_WHERE\_CLAUSE 变量的值，此值将用在第二个 SELECT 语句中。

注：这些示例应使用 SQL\*Plus。

执行幻灯片中的第一个示例时，会提示用户输入 DEPTNO 和 HIREDATE 的值：

|                                   |
|-----------------------------------|
| <b>Enter value for deptno: 10</b> |
|-----------------------------------|

|                                              |
|----------------------------------------------|
| <b>Enter value for hiredate: 17-SEP-1987</b> |
|----------------------------------------------|

生成 MY\_COL 的以下值：

|                                                                                       |
|---------------------------------------------------------------------------------------|
| <b>MY_COL</b>                                                                         |
| <b>WHERE department_id = 10 AND hire_date = TO_DATE('27-SEP-1987', 'DD-MON-YYYY')</b> |

执行幻灯片中的第二个示例时，会生成以下输出：

|                  |
|------------------|
| <b>LAST_NAME</b> |
| -----            |
| <b>Whalen</b>    |

## 小结

在本附录中，您应该已经了解到：

- 可以编写一个 SQL 脚本来生成另一个 SQL 脚本
- 脚本文件常常使用数据字典
- 可以在文件中保存输出

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

SQL 可用来生成 SQL 脚本。这些脚本可用来避免重复编码、删除或重新创建对象、从数据字典中获得帮助，以及生成包含运行时参数的动态谓词。



# Oracle DB 体系结构组件

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本附录后，应能完成以下工作：

- 列出主要的数据库体系结构组件
- 描述后台进程
- 说明内存结构
- 将逻辑存储结构与物理存储结构关联起来



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

本附录将概括说明 Oracle DB 体系结构。您将了解有关 Oracle DB 的物理结构和逻辑结构以及各种组件及其作用。

## Oracle DB 体系结构：概览

Oracle 关系数据库管理系统 (RDBMS) 是一种数据库管理系统，它提供一种开放、全面、集成的信息管理方式。



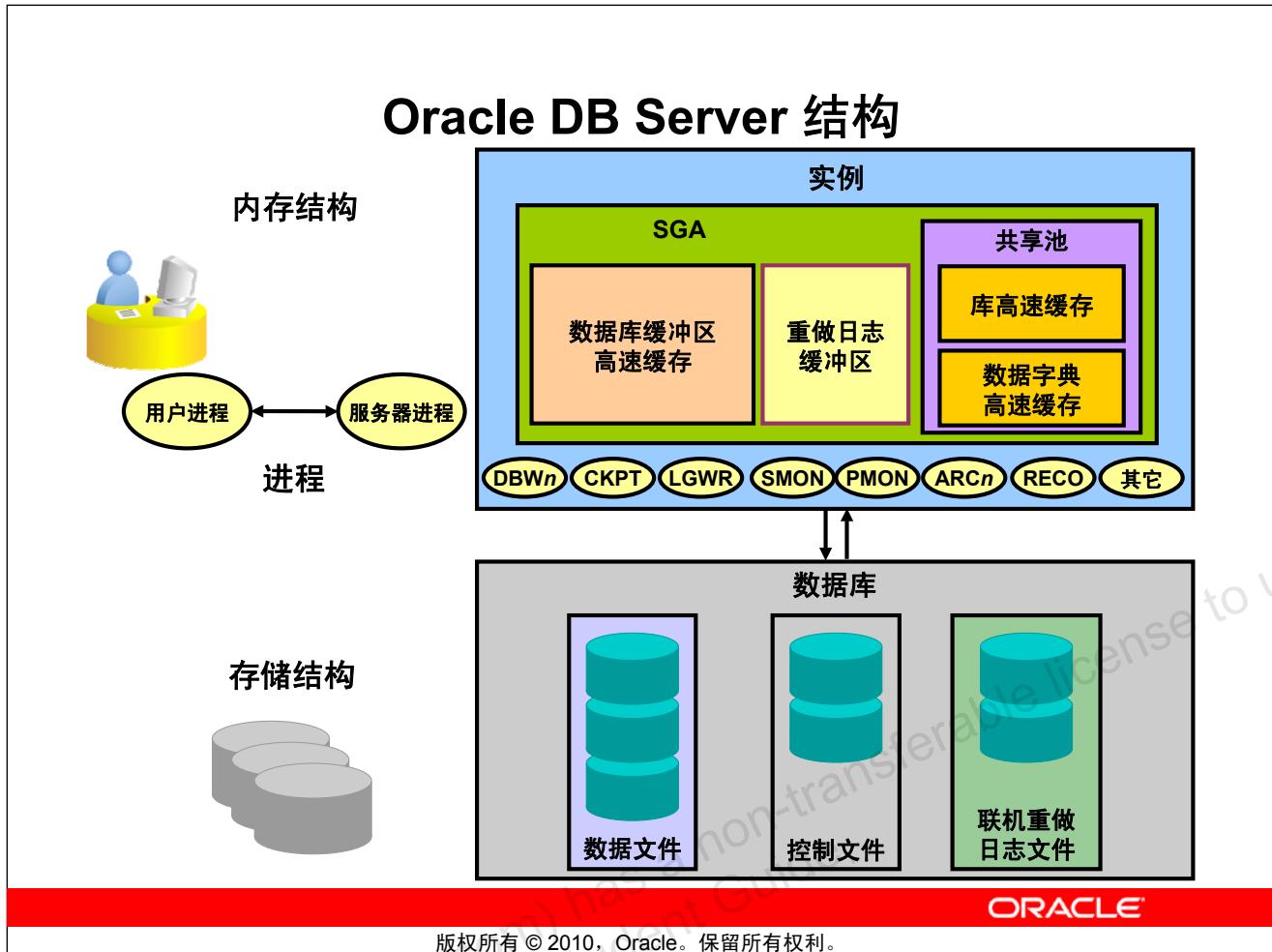
ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### Oracle DB 体系结构：概览

数据库是被视为单元的数据的集合，其用途是存储和检索相关信息。

Oracle DB 的可靠性非常高，可以管理多用户环境下的大量数据，这样很多用户可以并行访问同一数据。实现此功能的同时还可以保持较高的性能。同时，既可以阻止未授权的访问，又为故障恢复提供了有效解决方案。



## Oracle DB Server 结构

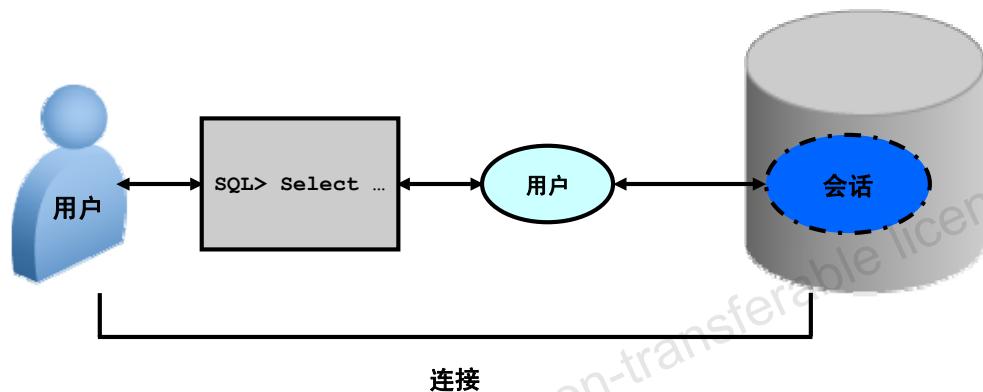
Oracle DB 由两个主要组件组成，即实例和数据库本身。

- 实例由系统全局区 (SGA, 内存结构中的一个集合) 和在数据库中执行任务的后台进程组成。每次启动实例时，都会分配 SGA 并启动后台进程。
- 数据库包括物理结构和逻辑结构。由于物理结构和逻辑结构是分开的，因此管理数据的物理存储结构时不会影响对逻辑存储结构的访问。物理存储结构包括：
  - 存储数据库配置的控制文件
  - 包含数据库恢复所需信息的重做日志文件
  - 存储所有数据的数据文件

Oracle 实例使用内存结构和进程来管理和访问数据库存储结构。所有内存结构都存在在构成数据库服务器的那些计算机的主内存中。进程是在这些计算机的内存中运行的作业。一个进程被定义为操作系统中可运行一系列步骤的一个“控制线程”或一种机制。

## 连接到数据库

- 连接：用户进程与数据库实例之间的通信路径
- 会话：用户通过用户进程到数据库实例的特定连接



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 连接到数据库

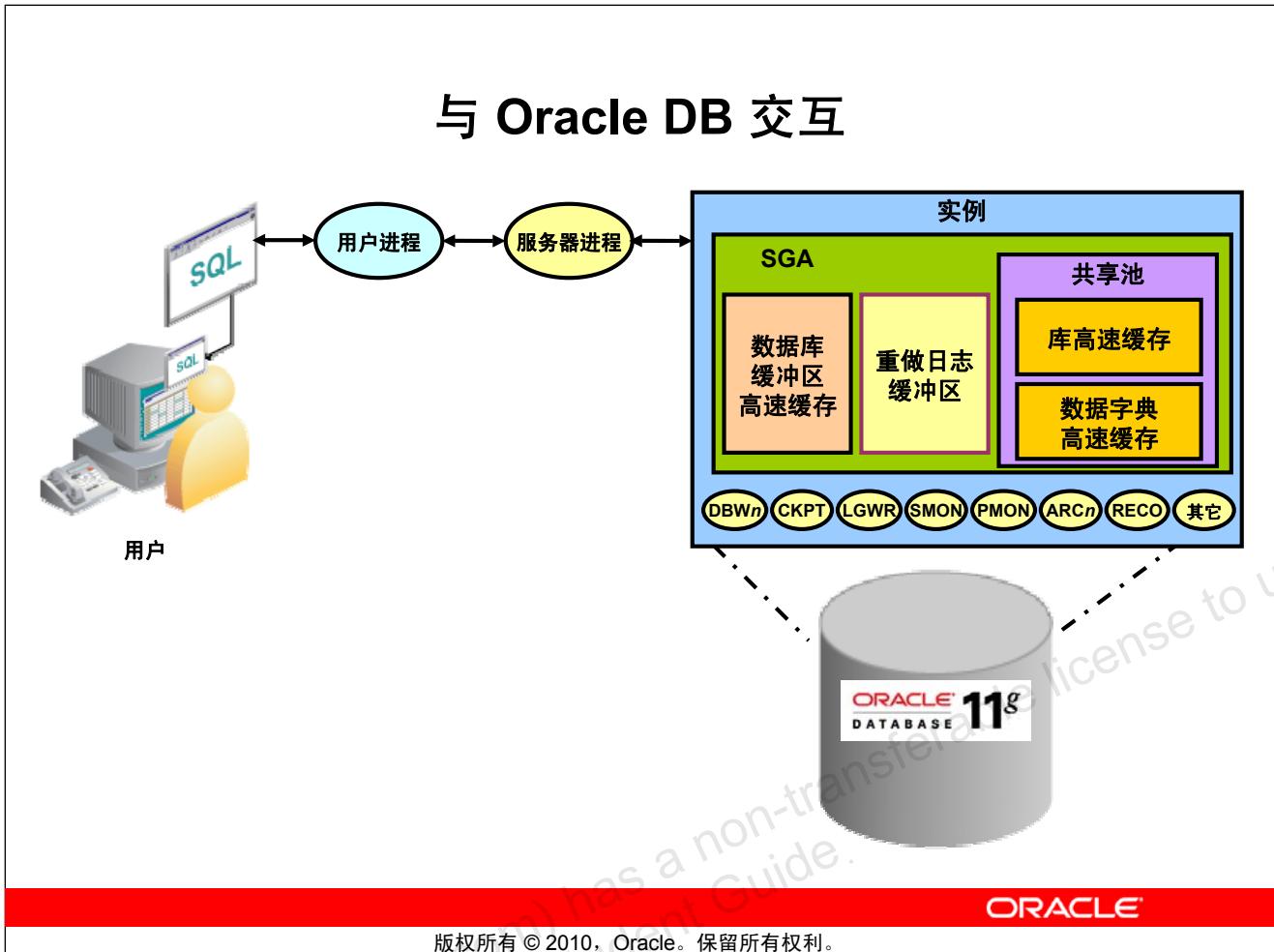
要访问数据库中的信息，用户需要使用工具（如 SQL\*Plus）连接到数据库。用户建立连接之后，就会为用户创建一个会话。连接和会话都与用户进程密切相关，但意义却大不相同。

连接是用户进程和 Oracle DB 实例之间的通信路径。通信路径是使用提供的进程间通信机制或网络软件建立的（当不同的计算机运行数据库应用程序和 Oracle DB 并通过网络进行通信时）。

会话代表当前用户登录数据库实例的状态。例如，当某个用户启动 SQL\*Plus 时，该用户必须提供有效的用户名和口令，然后系统才会为该用户建立一个会话。会话从用户建立连接时开始，一直持续到用户断开连接或退出数据库应用程序时为止。

如果使用的是专用连接，则会话由永久专用进程来处理。在使用共享连接的情况下，会话由中间层或 Oracle 共享服务器体系结构从进程池中选择的可用服务器进程提供服务。

可以为使用同一用户名的单个 Oracle DB 用户创建并行存在的多个会话，但这些会话是通过不同应用程序或同一应用程序的多次调用创建的。



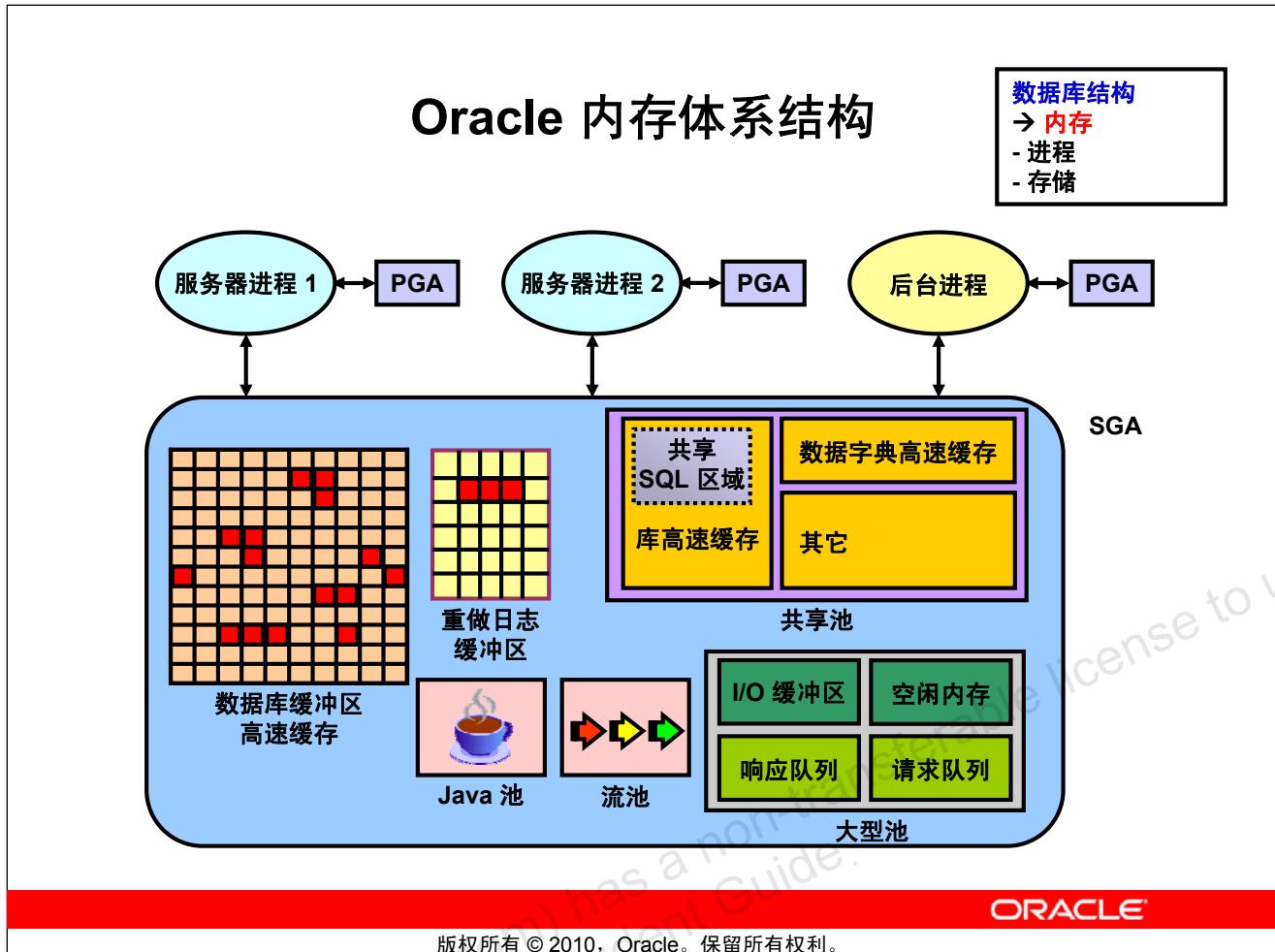
### 与 Oracle DB 交互

以下示例描述了一些最基本的 Oracle DB 操作。该示例中举例说明了 Oracle DB 的配置，其中用户进程和关联服务器进程分别位于不同的计算机上，而且已通过网络建立连接。

1. 在安装了 Oracle DB 的节点（通常称为主机或数据库服务器）上启动了一个实例。
2. 用户启动一个应用程序，从而衍生了一个用户进程。该应用程序尝试建立一个与服务器的连接（连接可以是本地连接、客户机服务器连接或者是通过中间层建立的三层连接。）
3. 服务器运行具有适当 Oracle Net Services 处理程序的监听程序。服务器检测到应用程序发出的连接请求，并创建一个代表用户进程的专用服务器进程。
4. 用户运行一条 DML 类型的 SQL 语句并提交事务处理。例如，用户更改一个表中客户的地址并提交更改。
5. 服务器进程接收该语句，并检查共享池（一个 SGA 组件）中是否有包含类似 SQL 语句的共享 SQL 区域。如果找到共享 SQL 区域，服务器进程就会检查用户对请求数据的访问权限，然后使用现有共享 SQL 区域处理该语句。如果未找到，会为该语句分配新的共享 SQL 区域，因此可以解析和处理该语句。

## 与 Oracle DB 交互（续）

6. 服务器进程检索所有需要的数据值，可以从存储表的实际数据文件中进行检索，也可以从 SGA 中高速缓存的数据值中进行检索。
7. 服务器进程修改 SGA 中的数据。因为已提交事务处理，所以逻辑写进程 (LGWR) 会立即在重做日志文件中记录事务处理。如果有必要，数据库写进程 (DBWn) 会将修改块永久写入磁盘。
8. 如果事务处理成功，服务器进程通过网络向应用程序发送一条消息。如果未成功，则发送一条错误消息。
9. 在整个过程中，其它后台进程也在运行，监视是否存在需要干预的情况。此外，数据库服务器还管理其他用户的事务处理，防止在请求同一数据的事务处理之间发生争用。



## Oracle 内存结构

Oracle DB 会创建和使用各种用途的内存结构。例如，在内存中存储正在运行的程序代码、用户间共享数据以及每个连接用户的专用数据区域。

一个实例有两个关联的基本内存结构：

- 一种是系统全局区 (SGA)，这是一组共享内存结构，被称为 SGA 组件，其中包含一个 Oracle DB 实例的数据和控制信息。SGA 由所有服务器进程和后台进程共享。SGA 中存储的数据示例包括高速缓存的数据块和共享的 SQL 区域。
- 另一种是程序全局区 (PGA)，这是包含一个服务器进程或后台进程的数据及控制信息的内存区。PGA 是启动服务器进程或后台进程启动时由 Oracle DB 创建的非共享内存。服务器进程对 PGA 的访问是互斥的。每个服务器进程和后台进程都有自己的 PGA。

## Oracle 内存结构（续）

SGA 是包含实例的数据和控制信息的内存区。SGA 包含以下数据结构：

- **数据库缓冲区高速缓存：**用来缓存从数据库中检索到的数据块
- **重做日志缓冲区：**用来缓存用于实例恢复的重做信息（可以写入磁盘上存储的物理重做日志文件之前的）
- **共享池：**用来缓存在用户间可共享的各种结构
- **大型池：**可选区域，用来为某些大型进程（例如 Oracle 备份和恢复操作）和输入/输出（I/O）服务器进程提供较大的内存分配空间
- **Java 池：**用来存储 Java 虚拟机（JVM）中特定会话的所有 Java 代码和数据
- **流池：**Oracle Streams 使用它存储捕获和应用所需的信息

使用 Oracle Enterprise Manager 或 SQL\*Plus 启动实例时，将会显示为 SGA 分配的内存量。

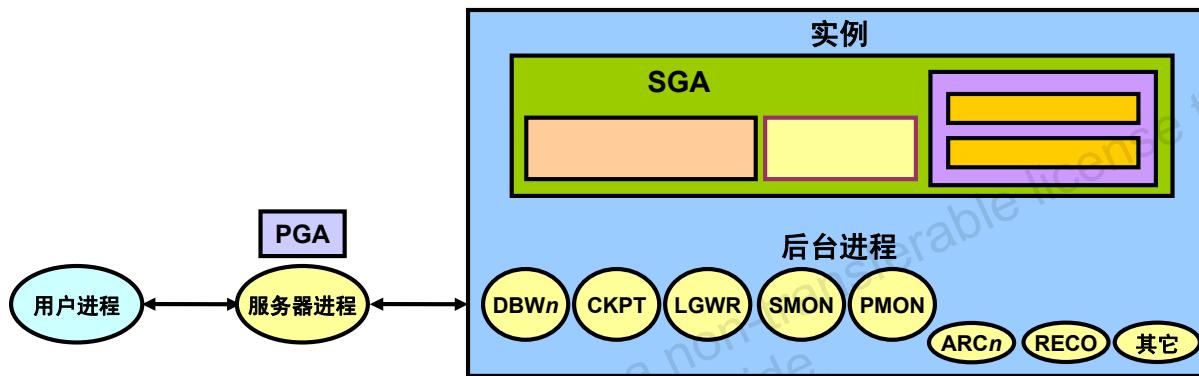
使用动态 SGA 基础结构时，可以在不关闭实例的情况下更改数据库缓冲区高速缓存、共享池、大型池、Java 池和流池的大小。

Oracle DB 使用初始化参数创建和配置内存结构。例如，SGA\_TARGET 参数指定 SGA 组件的总大小。如果将 SGA\_TARGET 设置为 0，则会禁用“Automatic Shared Memory Management（自动共享内存管理）”功能。

# 进程体系结构

**数据库结构**  
 - 内存  
 → **进程**  
 - 存储

- **用户进程:**
  - 在一个数据库用户或一个批处理连接到 Oracle DB 时启动
- **数据库进程:**
  - 服务器进程: 连接到 Oracle 实例, 在用户建立会话时启动
  - 后台进程: 在启动 Oracle 实例时启动



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 进程体系结构

Oracle DB Server 中的进程主要分为两组:

- 运行应用程序或 Oracle 工具代码的用户进程。
- 运行 Oracle DB Server 代码的 Oracle DB 进程。数据库进程包括服务器进程和后台进程。

当用户运行应用程序或 Oracle 工具（例如 SQL\*Plus）时，Oracle DB 会创建一个用户进程来运行该用户的应用程序。Oracle DB 还会创建一个服务器进程来执行该用户进程发布的命令。此外，Oracle Server 还为一个实例提供一组后台进程。这些进程不仅彼此交互，还与操作系统交互，目的是为了管理内存结构，通过异步执行 I/O 操作将数据写入磁盘，以及执行其它需要完成的任务。

不同的 Oracle DB 配置采用不同的进程结构，这取决于操作系统和选择的 Oracle DB 选件。连接用户的代码可以配置为专用服务器或共享服务器。

- 配置为专用服务器时，对于每位用户，数据库应用程序由一个用户进程运行，该用户进程由执行 Oracle DB Server 代码的专用服务器进程提供服务。
- 配置为共享服务器时，不需要由专用服务器进程为每个连接提供服务。分派程序将多个传入网络会话请求指引到共享服务器进程池。共享服务器进程为所有客户机请求提供服务。

## 进程体系结构（续）

### 服务器进程

Oracle DB 会创建服务器进程来处理连接到实例的用户进程的请求。某些情况下，当应用程序和 Oracle DB 在同一计算机上运行时，可以将用户进程与对应的服务器进程组合为单个进程来减少系统开销。但是，当应用程序和 Oracle DB 在不同计算机上运行时，用户进程总是通过一个另外的服务器进程与 Oracle DB 通信。

创建的服务器进程代表每个用户的应用程序，这些进程可以执行以下一项或多项操作：

- 分析和运行通过应用程序发布的 SQL 语句。
- 如果数据块目前尚未在 SGA 中，则从磁盘数据文件中将必要的数据块读取到 SGA 的共享数据库缓冲区中。
- 以某种方式返回结果，以便应用程序可以处理有关信息。

### 后台进程

为了最大限度地提高性能并满足多个用户的需要，多进程 Oracle DB 系统使用一些称为后台进程的附加 Oracle DB 进程。一个 Oracle DB 实例可以有多个后台进程。

要成功启动数据库实例，需要以下后台进程：

- 数据库写进程 (DBW $n$ )
- 日志写进程 (LGWR)
- 检查点 (CKPT)
- 系统监视器 (SMON)
- 进程监视器 (PMON)

以下后台进程是根据需要可启动的一些后台进程示例：

- 恢复器 (RECO)
- 作业队列
- 归档 (ARC $n$ )
- 队列监视器 (QMN $n$ )

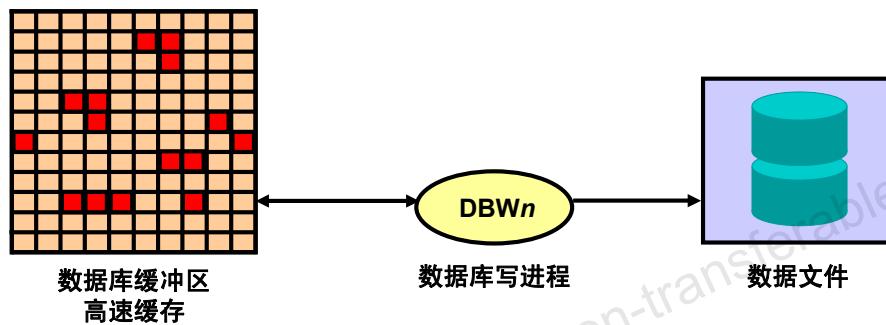
在更高级配置（如 Real Application Clusters (RAC)）中可能还有其它后台进程。有关后台进程的详细信息，请参见 V\$BGPROCESS 视图。

在许多操作系统中，启动实例时会自动创建后台进程。

## 数据库写进程

将数据库缓冲区高速缓存中经过修改的缓冲区（灰数据缓冲区）写入磁盘的两种方式：

- 在执行其它处理时异步执行
- 定期执行以推进检查点



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

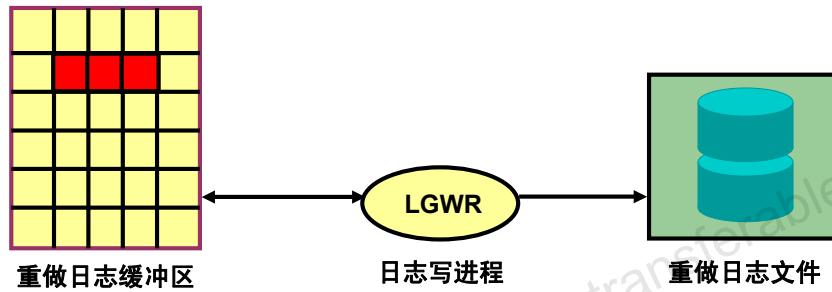
### 数据库写进程

数据库写进程 (DBWn) 会将缓冲区的内容写入数据文件。DBWn 进程负责将数据库缓冲区高速缓存中经过修改的缓冲区（灰数据缓冲区）写入磁盘。虽然在多数系统中一个数据库写进程 (DBW0) 足以满足需要，但是，如果系统中修改数据的任务非常繁重，则可通过配置附加进程 (DBW1 到 DBW9 及 DBWa 到 DBWj) 来提高写入性能。这些附加 DBWn 进程在单处理器系统中没有什么用。

如果数据库缓冲区高速缓存中的某个缓冲区已修改，该缓冲区会被标记为“灰”缓冲区，并添加在按系统更改号 (SCN) 顺序保存的灰缓冲区 LRUW 列表中，因此与在重做日志中写入的这些更改缓冲区的对应重做顺序相匹配。如果缓冲区高速缓存中可用缓冲区的数目低于内部阈值，使服务器进程难以获得可用缓冲区，则 DBWn 会将灰缓冲区写入数据文件，写入顺序是按 LRUW 列表顺序修改缓冲区的顺序。

## 日志写进程

- 将重做日志缓冲区写入磁盘上的重做日志文件
- LGWR 在以下情况下执行写操作：
  - 进程提交事务处理时
  - 重做日志缓冲区的三分之一已满时
  - DBW $n$  进程将经过修改的缓冲区写入磁盘之前



**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 日志写进程

日志写进程 (LGWR) 负责通过将重做日志缓冲区条目写入磁盘重做日志文件来管理重做日志缓冲区。LGWR 会写入自上次写入以来复制到缓冲区中的所有重做条目。

重做日志缓冲区是循环缓冲区。当 LGWR 将重做日志缓冲区中的重做条目写入重做日志文件后，服务器进程就可以将新条目复制到重做日志缓冲区中那些已写入磁盘的条目之上。LGWR 的写入速度通常足够快，可以确保缓冲区中始终有空间可供新条目使用，即使对重做日志的访问量很大时也是如此。

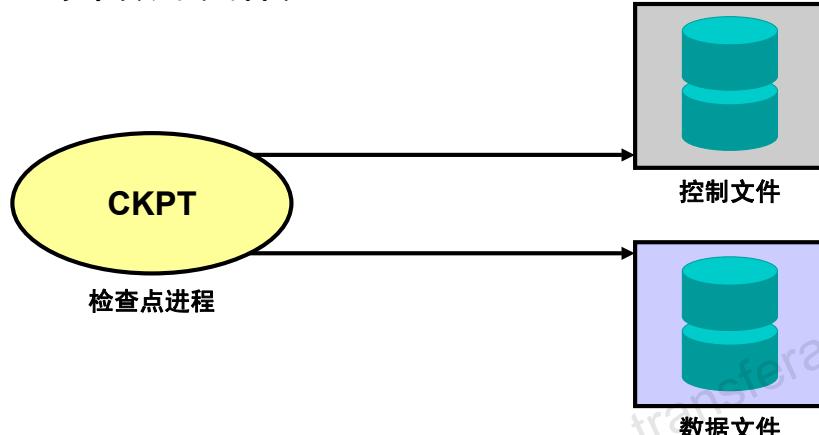
LGWR 将缓冲区的一个连续部分写入磁盘。LGWR 在以下情况下执行写操作：

- 用户进程提交事务处理时
- 重做日志缓冲区的三分之一已满时
- DBW $n$  进程将修改缓冲区写入磁盘之前（如有必要）

## 检查点进程

记录以下项目中的检查点信息：

- 控制文件
- 每个数据文件头



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

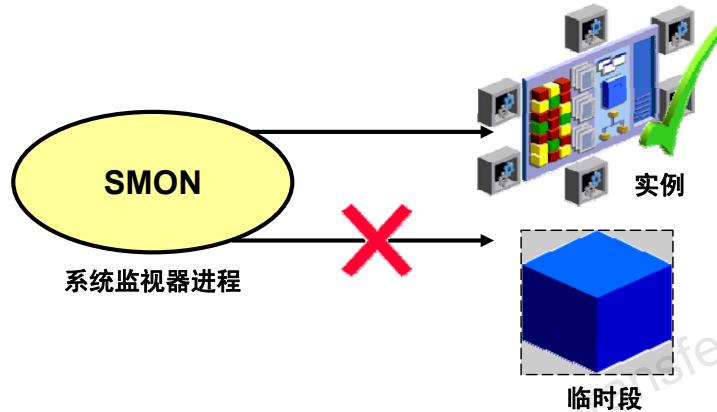
### 检查点进程

检查点是一种数据结构，用于定义数据库重做线程中的 SCN。检查点记录在控制文件和每个数据文件头中，对进行恢复至关重要。

出现检查点时，Oracle DB 必须更新所有数据文件的头来记录检查点详细信息。这个任务由 CKPT 进程完成。CKPT 进程不会将块写入磁盘；该工作总是由 DBWn 执行。在文件头中记录的 SCN 保证了对数据库块进行的所有更改都发生在将相应 SCN 写入磁盘之前。由 Oracle Enterprise Manager 中的 SYSTEM\_STATISTICS 监视器显示的统计 DBWR 检查点数指出了完成的检查点请求数。

## 系统监视器进程

- 在实例启动时执行恢复
- 清除不使用的临时段



ORACLE

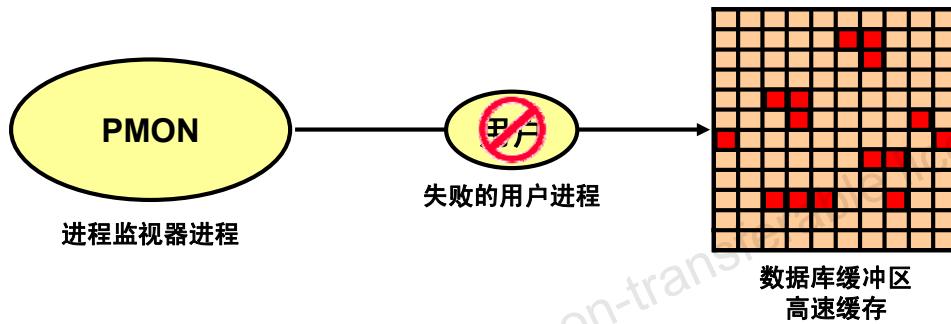
版权所有 © 2010, Oracle。保留所有权利。

### 系统监视器进程

如果需要，系统监视器 (SMON) 会在实例启动时执行恢复。SMON 还负责清除不再使用的临时段。如果在实例恢复过程中由于文件读取或脱机错误跳过任何已终止的事务处理，则 SMON 将在表空间或文件重新联机时恢复这些事务处理。SMON 会定期检查来查看是否需要运行。其它进程在检测到需要 SMON 时也可以调用它。

## 进程监视器进程

- 在用户进程失败时执行进程恢复：
  - 清除数据库缓冲区高速缓存
  - 释放用户进程占用的资源
- 监视会话是否发生空闲会话超时
- 在监听程序中动态注册数据库服务



**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 进程监视器进程

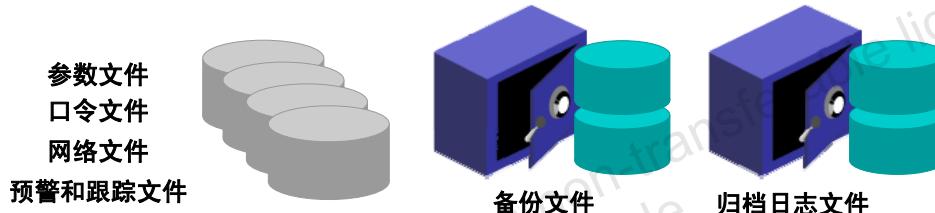
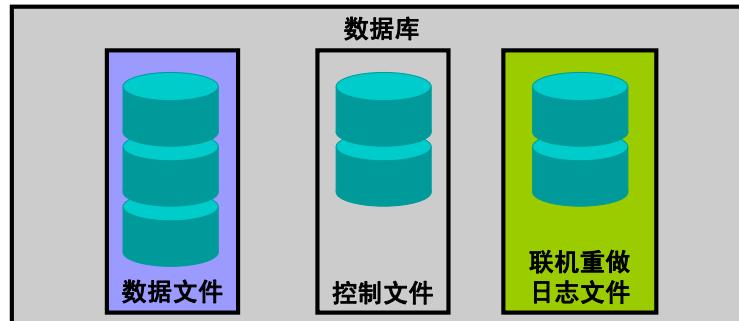
进程监视器 (PMON) 会在用户进程失败时执行进程恢复。PMON 负责清除数据库缓冲区高速缓存，还负责释放用户进程占用的资源。例如，PMON 会重置活动事务处理表的状态，释放锁，以及从活动进程列表中删除有关进程 ID。

PMON 会定期检查分派程序和服务器进程的状态，重新启动任何已停止运行的分派程序和服务器进程（但不会启动 Oracle DB 故意终止的分派程序和服务器进程）。PMON 还会在网络监听程序中注册有关实例和分派程序进程的信息。

与 SMON 一样，PMON 也会定期检查来查看是否需要运行。当其它进程检测到需要 PMON 时可以调用它。

# Oracle DB 存储体系结构

**数据库结构**  
 - 内存  
 - 进程  
 → **存储**



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## Oracle DB 存储体系结构

构成 Oracle DB 的文件可分为以下类别：

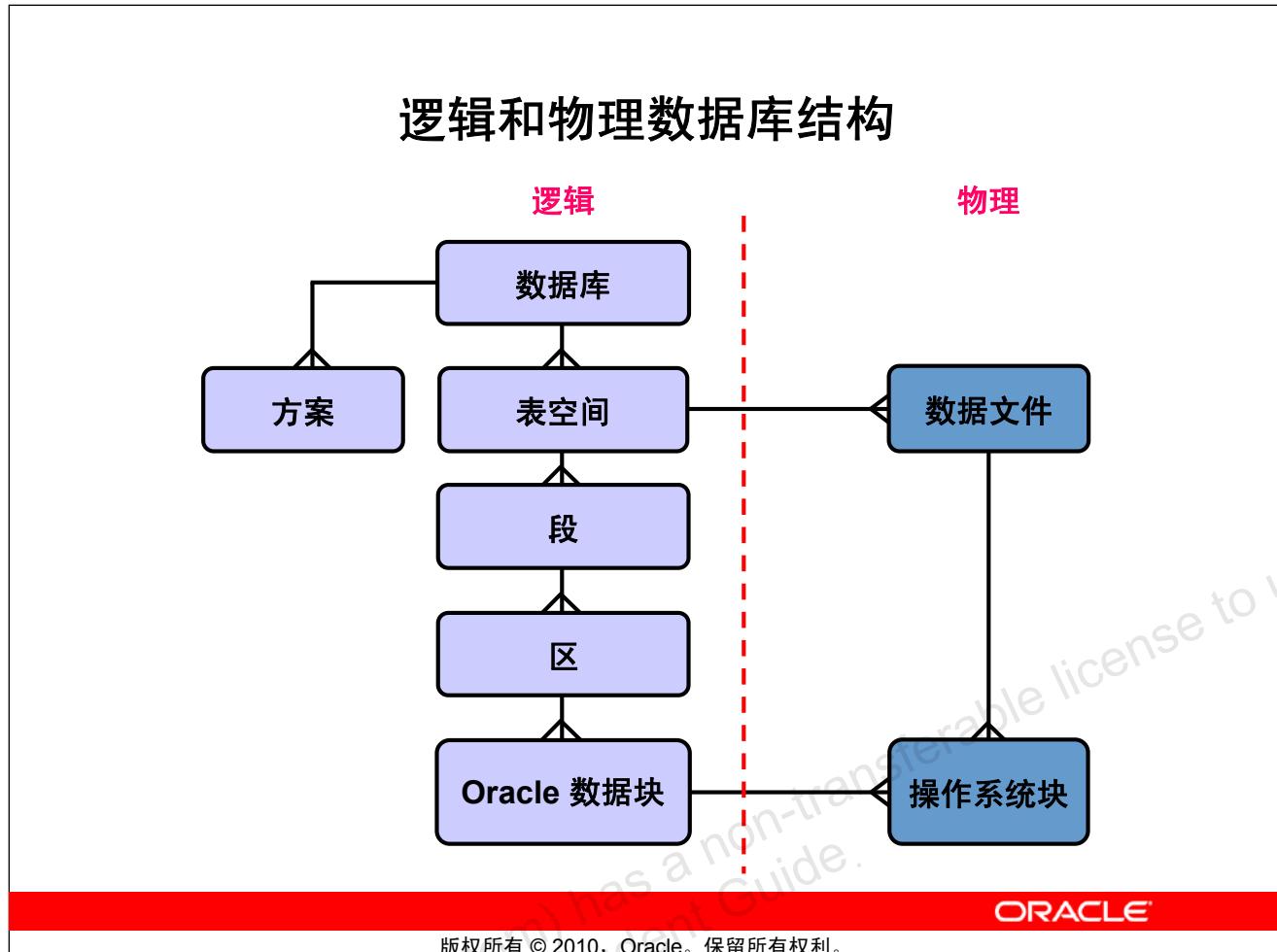
- **控制文件**：包含与数据库本身相关的数据，即物理数据库结构信息。这些文件对数据库至关重要。没有这些文件，就无法打开数据文件来访问数据库中的数据。
- **数据文件**：包含数据库的用户数据或应用程序数据，以及元数据和数据字典。
- **联机重做日志文件**：用于数据库实例恢复。如果数据库服务器崩溃，但未丢失任何数据文件，那么在实例中可使用这些文件中的信息恢复数据库。

下列附加文件对成功运行数据库非常重要：

- **备份文件**：用于数据库恢复。如果原始文件在发生介质故障或用户错误时被损坏或删除，通常要还原备份文件。
- **归档日志文件**：包含实例发生的数据更改（重做）的实时历史记录。使用这些文件和数据库备份，可以恢复丢失的数据文件。也就是说，归档日志能够恢复还原的数据文件。
- **参数文件**：用于定义实例启动时的配置。
- **口令文件**：允许 sysdba/sysoper/sysasm 远程连接到数据库执行管理任务。

## Oracle DB 存储体系结构（续）

- **网络文件：**用于启动数据库监听程序和存储用户连接所需的信息。
- **跟踪文件：**每个服务器和后台进程都可以写入一个关联的跟踪文件。当进程检测到内部错误时，进程会将有关错误的信息转储到相应的跟踪文件中。跟踪文件中写入的某些信息是为数据库管理员提供的，其它信息是为 Oracle Support Services 提供的。
- **预警日志文件：**这些文件包含特殊的跟踪条目。数据库预警日志是按时间顺序列出的消息日志和错误日志。每个实例都有一个预警日志文件。Oracle 建议定期复查此预警日志。



## 逻辑和物理数据库结构

在 Oracle DB 中有逻辑和物理两种存储结构。

### 表空间

数据库分为多个逻辑存储单元，这些单元称为表空间，用于对相关逻辑结构进行分组。

例如，表空间一般会将应用程序的所有对象分成一组，以简化一些管理操作。您可以使用一个表空间存放应用程序数据，使用另一个存放应用程序索引。

### 数据库、表空间和数据文件

本幻灯片对数据库、表空间和数据文件之间的关系进行了说明。每个数据库在逻辑上都分为一个或多个表空间。对每个表空间均显式创建一个或多个数据文件，用于在物理上存储表空间中所有逻辑结构的数据。如果表空间是一个 TEMPORARY 表空间，则这个表空间有一个临时文件，而不是数据文件。

## 逻辑和物理数据库结构（续）

### 方案

方案是数据库用户所拥有的数据库对象的集合。方案对象是直接引用数据库数据的逻辑结构。方案对象包括诸如表、视图、序列、存储过程、同义词、索引、集群和数据库链接等结构。通常，方案对象包括应用程序在数据库中创建的所有内容。

### 数据块

从最细的层面来讲，Oracle DB 的数据存储在数据块中。一个数据块与磁盘中特定字节数的物理数据库空间相对应。在创建表空间时指定了每个表空间的数据块大小。数据库会使用和分配 Oracle 数据块中的空闲数据库空间。

### 区

逻辑数据库空间的下一层称为区。区是特定数量的相邻数据块（通过一次分配获得的），用于存储特定类型的信息。

### 段

逻辑数据库存储中区的上一层称为段。一个段是为某个逻辑结构分配的一组区。例如，不同类型的段包括：

- **数据段：**每一个以非集群、非索引方式组织的表都有一个数据段，但外部表、全局临时表和分区表除外，这些表每一个表都有一个或多个段。表中的所有数据都存储在相应数据段的区中。对于分区表，每个分区都有一个数据段。每个集群也都有一个数据段。集群中每个表的数据都存储在集群的数据段中。
- **索引段：**每个索引都有一个索引段，用于存储索引的所有数据。对于分区索引，每个分区都有一个索引段。
- **还原段：**为每个数据库实例创建了一个 UNDO 表空间，该表空间包含多个还原段，用于临时存储还原信息。还原段中的信息用于生成读一致性数据库信息，以便在数据库恢复过程中回退用户未提交的事务处理。
- **临时段：**当 SQL 语句需要临时工作区来完成执行时，Oracle DB 就会创建临时段。语句完成执行后，临时段的区将返回到实例以备将来使用。可以为每个用户指定一个默认临时表空间，也可以指定一个在数据库范围内使用的默认临时表空间。

在 Oracle DB 中会动态分配空间。如果段中的现有区都已满，则会再增加一些区。因为区是根据需要分配的，因此段中的区在磁盘中可能是相邻的，也可能是不相邻的。

## 处理 SQL 语句

- 使用以下进程连接到实例：
  - 用户进程
  - 服务器进程
- 使用的 Oracle Server 组件取决于 SQL 语句的类型：
  - 查询会返回一些行
  - 数据操纵语言 (DML) 语句会记录更改
  - 提交会确保执行事务处理恢复
- 某些 Oracle Server 组件不参与 SQL 语句处理。



版权所有 © 2010, Oracle。保留所有权利。

### 处理 SQL 语句

并不是 Oracle 实例的所有组件都用来处理 SQL 语句。用户进程和服务器进程用来将用户连接到 Oracle 实例。这些进程不是 Oracle 实例的一部分，但它们是处理 SQL 语句所必需的。

有些后台进程、SGA 结构和数据库文件用来处理 SQL 语句。根据 SQL 语句的类型可使用不同的组件：

- 查询需要执行其它处理才能将行返回给用户。
- 数据操纵语言 (DML) 语句需要执行其它处理才能记录数据更改。
- 提交处理确保可以恢复事务处理中的修改数据。

某些需要的后台进程并不直接参与处理 SQL 语句，但可用来提高性能和恢复数据库。

例如，归档程序后台进程 ARCn 是一个可选进程，可用来确保恢复生产数据库。

## 处理查询

- 分析：
  - 搜索相同的语句。
  - 检查语法、对象名和权限。
  - 锁定分析期间使用的对象。
  - 创建并存储执行计划。
- 执行：确定所执行。
- 提取：将行返回到用户进程。



版权所有 © 2010, Oracle。保留所有权利。

### 处理查询

查询不同于其它类型的 SQL 语句，因为如果查询执行成功，则会返回数据结果。其它语句只是返回成功或失败，而查询却可以返回一行或数千行。

处理查询分为三个主要阶段：

- 分析
- 执行
- 提取

在分析阶段，SQL 语句从用户进程传递到服务器进程，分析后的 SQL 语句被加载到共享 SQL 区。

在分析期间，服务器进程会执行以下活动：

- 在共享池中搜索 SQL 语句的现有副本
- 通过检查 SQL 语句的语法对其进行验证
- 在数据字典中进行查找来验证表和列定义

执行阶段会使用最佳优化程序方法来执行语句，并将检索到的行返回给用户。

# 共享池

- 库高速缓存包含 SQL 语句文本、经过分析的代码和执行计划。
- 数据字典高速缓存包含表、列及其它对象定义和权限。
- 共享池的大小由 SHARED\_POOL\_SIZE 确定。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 共享池

在分析阶段中，服务器进程使用 SGA 中称为共享池的区域来编译 SQL 语句。共享池有两个主要组件：

- 库高速缓存
- 数据字典高速缓存

### 库高速缓存

库高速缓存在称为共享 SQL 区域的内存结构中存储关于最近使用的 SQL 语句的信息。

共享 SQL 区域包含：

- SQL 语句的文本
- 分析树，这是编译版本的语句
- 执行计划，其中包含执行语句时要采取的步骤

优化程序是 Oracle Server 中的功能，用来确定最佳执行计划。

当重复执行一条 SQL 语句时，因为共享 SQL 区域已经包含该语句的执行计划，所以服务器进程不需要再分析该语句。库高速缓存通过降低分析时间和内存需求提高了重复使用 SQL 语句的应用程序的性能。如果不再重复使用 SQL 语句，最终到期时会从库高速缓存中清除该语句。

## 共享池（续）

### 数据字典高速缓存

数据字典高速缓存也称为字典高速缓存或行高速缓存，它是数据库中最近使用定义的集合，其中包含关于数据库文件、表、索引、列、用户、权限和其它数据库对象的信息。

在分析阶段中，服务器进程会在字典高速缓存中查找信息，以便对 SQL 语句中指定的对象名进行解析并验证访问权限。如果有必要，服务器进程会从数据文件中加载此信息。

### 调整共享池大小

共享池的大小由初始化参数 SHARED\_POOL\_SIZE 指定。

## 数据库缓冲区高速缓存

- 在数据库缓冲区高速缓存中存储了最近使用的块。
- 缓冲区的大小由 DB\_BLOCK\_SIZE 确定。
- 缓冲区的数量由 DB\_BLOCK\_BUFFERS 定义。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 数据库缓冲区高速缓存

处理某个查询时，服务器进程会在数据库缓冲区高速缓存中查找它所需的所有块。如果没有在数据库缓冲区高速缓存中找到相应的块，服务器进程就会从数据文件中读取块并在缓冲区高速缓存中添加一个副本。由于对同一块的后续请求可能在内存中找到相应的块，因此，这些请求可能不需要物理读取。Oracle Server 使用一种“least recently used（近来最不常用）”的算法来定期释放最近未访问的缓冲区，以便在缓冲区高速缓存中为新块腾出空间。

#### 调整数据库缓冲区高速缓存的大小

缓冲区高速缓存中每一个缓冲区的大小都等于 Oracle 块的大小，它是由 DB\_BLOCK\_SIZE 参数指定的。缓冲区的数量等于 DB\_BLOCK\_BUFFERS 参数的值。

## 程序全局区 (PGA)

- 不共享
- 只有服务器进程可以写入
- 包含：
  - 排序区
  - 会话信息
  - 游标状态
  - 堆栈空间



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

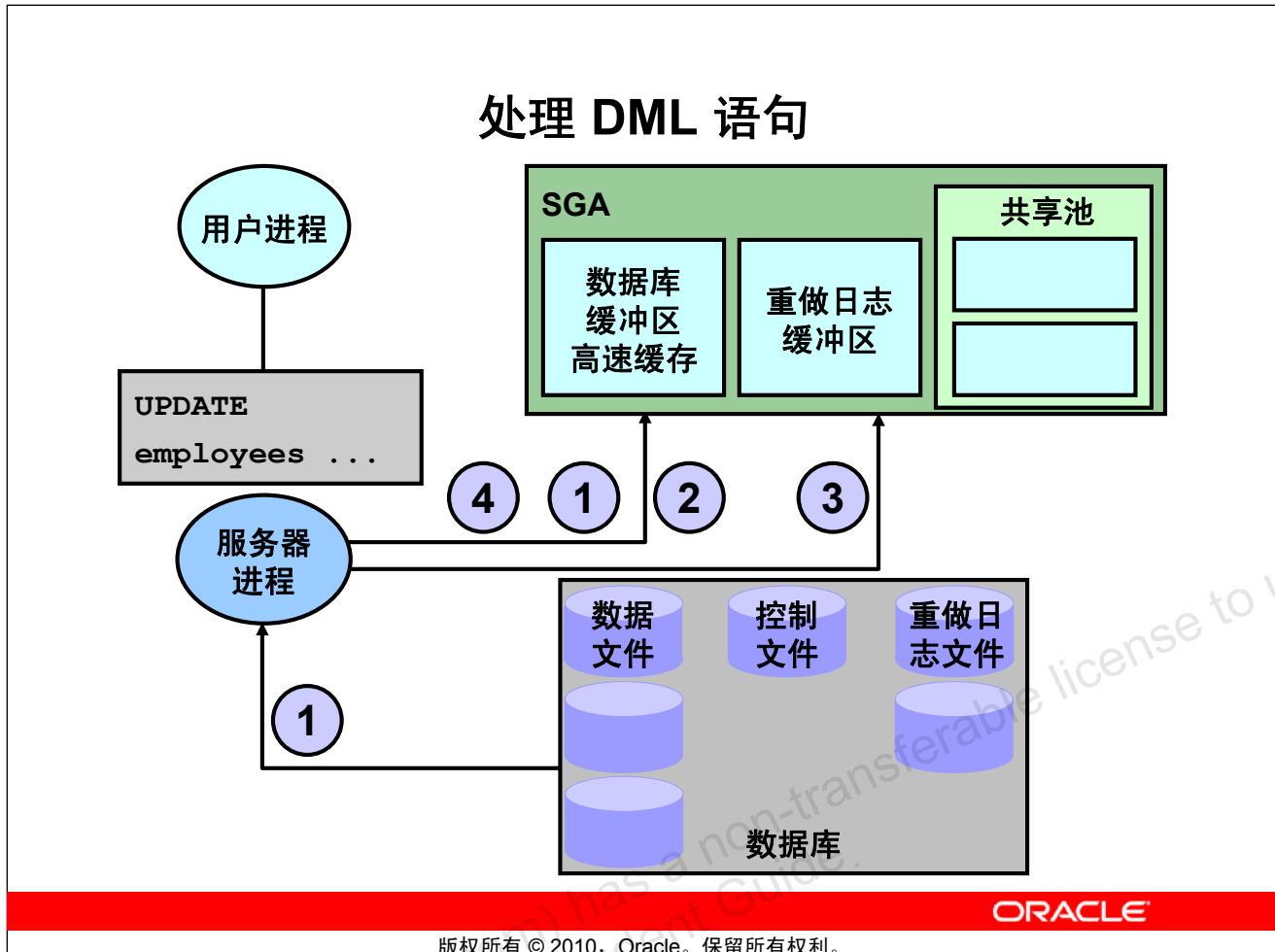
### 程序全局区 (PGA)

程序全局区 (PGA) 是一个内存区，其中包含服务器进程的数据及控制信息。它是启动某个服务器进程时由 Oracle Server 创建的非共享内存。只有该服务器进程可以对 PGA 进行访问，只有代表该服务器进程的 Oracle Server 代码可以对 PGA 进行读写。由 Oracle 实例附加的每一个服务器进程分配的 PGA 内存被称为实例分配的聚集 PGA 内存。

在专用服务器配置中，服务器的 PGA 包括以下组件：

- **排序区：**在处理 SQL 语句可能需要的任何排序过程中使用
- **会话信息：**包含会话的用户权限和性能统计数据
- **游标状态：**表示会话当前使用的 SQL 语句所处的处理阶段
- **堆栈空间：**包含其它会话变量

在创建进程时会分配 PGA，在终止进程时会释放它。



## 处理 DML 语句

数据操纵语言 (DML) 语句只需要两个处理阶段：

- 分析阶段与处理查询使用的分析阶段相同。
- 执行阶段需要执行其它处理才能进行数据更改。

### DML 执行阶段

执行 DML 语句时：

- 如果缓冲区高速缓存中没有数据块和回退块，服务器进程会将其从数据文件读取到缓冲区高速缓存。
- 服务器进程会在要修改的行上加锁。
- 在重做日志缓冲区中，服务器进程会记录对回退块和数据块所做的更改。
- 在回退块更改中记录了修改前的数据值。回退块用来存储数据的“前期映象”，这样在需要时可以回退 DML 语句。
- 在数据块更改中记录了新的数据值。

## 处理 DML 语句（续）

服务器进程会将“前期映象”记录在回退块中后更新数据块。这两项更改都是在数据库缓冲区高速缓存中进行的。缓冲区高速缓存中的任何更改块都被标记为灰缓冲区，即不同于磁盘中相应块的缓冲区。

DELETE 命令或 INSERT 命令的处理过程使用的步骤相似。DELETE 的“前期映象”包含已删除行中的列值，而 INSERT 的“前期映象”包含行位置信息。

因为对块进行的更改只记录在内存结构中，而没有立即写入磁盘，所以如果计算机故障导致 SGA 丢失，还会导致这些更改丢失。

## 重做日志缓冲区

- 它的大小由 LOG\_BUFFER 确定
- 记录在实例中所做的更改
- 可连续使用
- 是一种循环缓冲区



ORACLE

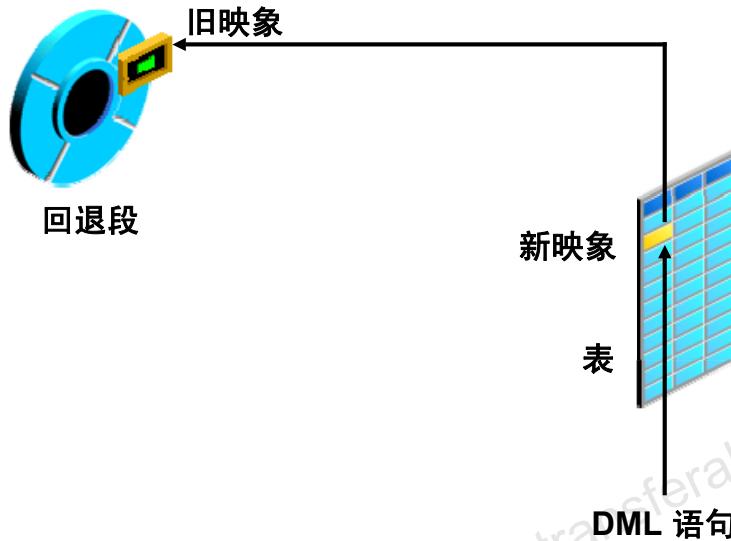
版权所有 © 2010, Oracle。保留所有权利。

### 重做日志缓冲区

服务器进程会在重做日志缓冲区中记录对数据文件块所做的大部分更改，重做日志缓冲区是 SGA 的一部分。重做日志缓冲区有以下特性：

- 它的大小（以字节计）由参数 LOG\_BUFFER 确定。
- 它在重做条目中记录更改块、更改位置和新值。在重做条目中不会区分更改块的类型，只会记录块中已更改的字节。
- 重做日志缓冲区可连续使用，一个事务处理所做的更改可与其它事务处理所做的更改交叉存放。
- 它是一种循环缓冲区，可以在填满后重复使用，但只有在重做日志文件中记录了所有旧的重做条目之后才可以。

## 回退段



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

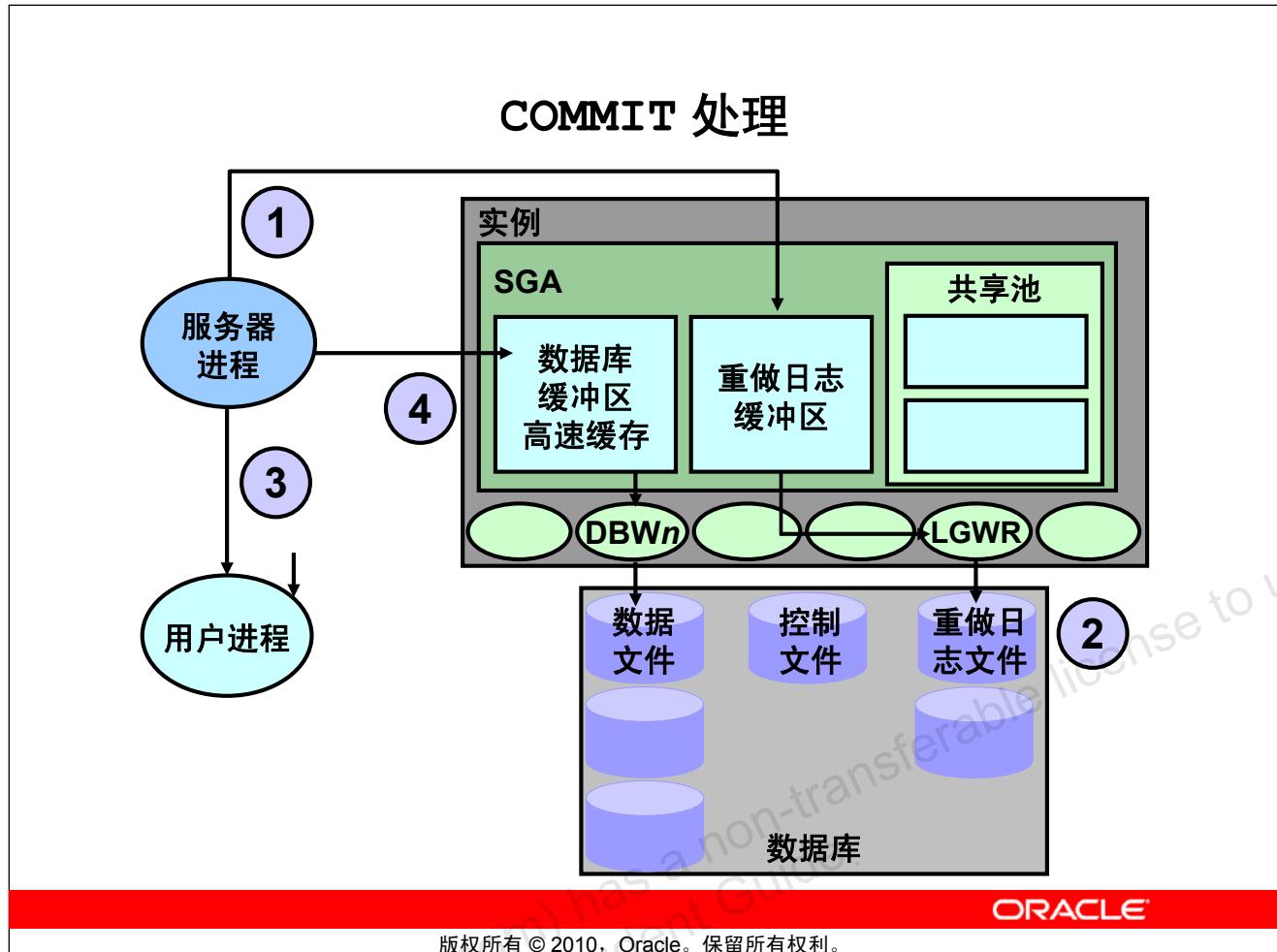
### 回退段

进行更改之前，服务器进程会将旧数据值保存在回退段中。此“前期映象”用来：

- 在回退事务处理时撤消更改。
- 提供读一致性，确保其它事务处理不会看到 DML 语句执行的未提交更改。
- 在出现故障时将数据库恢复到一致状态。

诸如表和索引之类的回退段存在于数据文件中，需要时回退块会被转入数据库缓冲区高速缓存中。回退段是由 DBA 创建的。

对回退段的更改会记录在重做日志缓冲区中。



## COMMIT 处理

Oracle Server 使用一种快速的 COMMIT 机制来保证在实例出现故障时可以恢复提交的更改。

### 系统更改号

无论何时提交事务处理时，Oracle Server 都会给事务处理分配一个提交 SCN。SCN 有规律地递增，而且在数据库中是唯一的。Oracle Server 将 SCN 用作内部时间戳来保持数据同步，这样从数据文件中检索数据时提供了读一致性。使用 SCN 时，Oracle Server 可以执行一致性检查，而不必依赖操作系统的日期和时间。

### 处理 COMMIT 的步骤

发出 COMMIT 命令后会执行以下步骤：

1. 服务器进程将提交记录连同 SCN 一起保存在重做日志缓冲区中。
2. LGWR 将提交前的所有重做日志缓冲区条目（包括提交记录）连续写入重做日志文件。此后，Oracle Server 就可以保证即使在出现实例故障的情况下也不会丢失更改。

## COMMIT 处理（续）

3. 通知用户 COMMIT 已完成。
4. 服务器进程会记录有关信息，表明事务处理已完成，可以解除资源锁了。

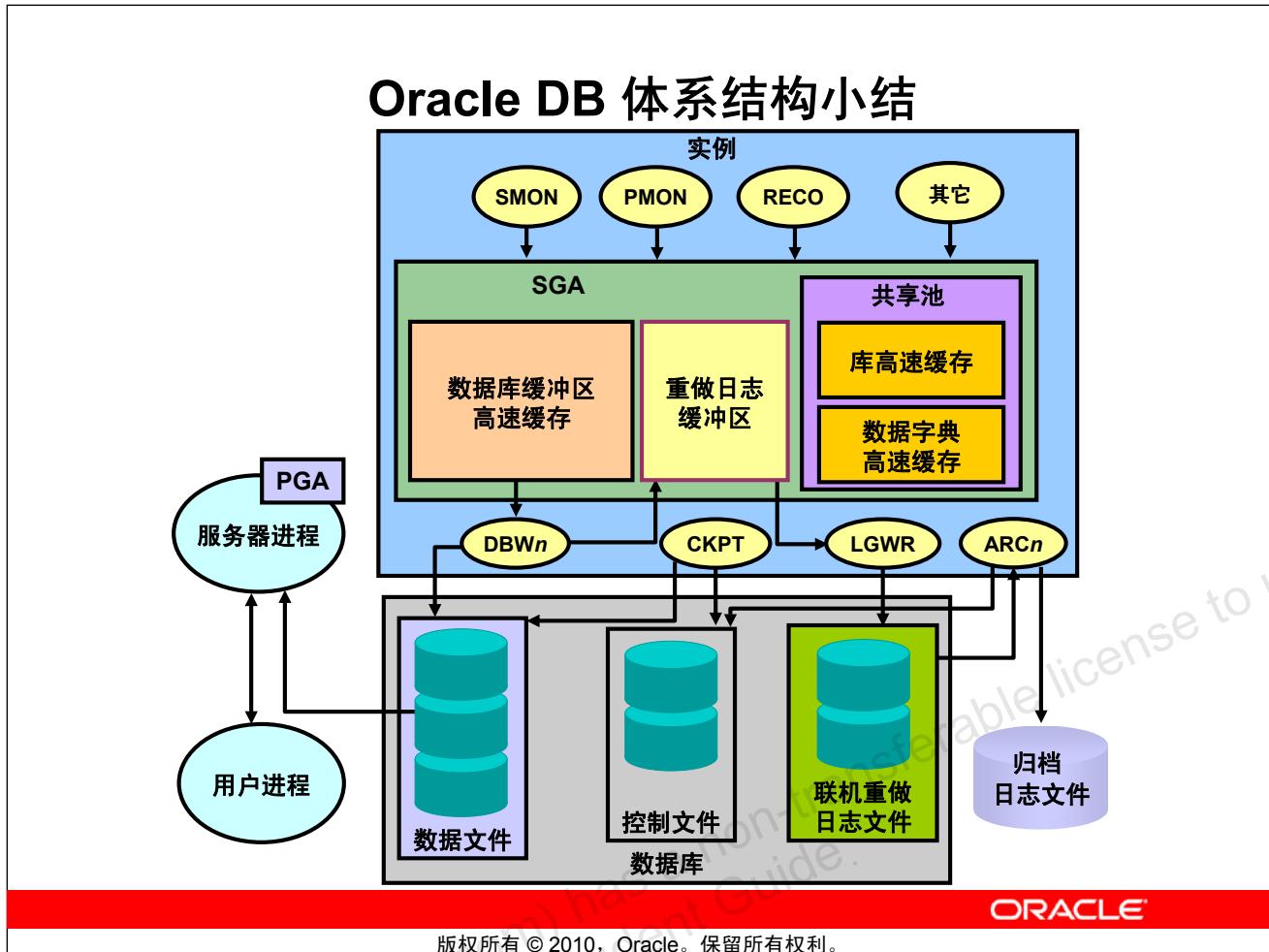
将灰缓冲区刷新到数据文件是由 DBW0 独立执行的，可以在提交前执行，也可以在提交后执行。

### 快速 COMMIT 的优势

快速 COMMIT 机制通过将更改写入重做日志缓冲区而不是写入数据文件来确保数据得以恢复。它具有以下优势：

- 与写入数据文件的不同块相比，连续写入日志文件速度要更快。
- 在日志文件只需要写入记录更改的少量信息，而在数据文件中写入时需要写入整块数据。
- 如果多个事务处理同时请求提交，实例会将重做日志记录合并在一个写入操作中。
- 如果重做日志缓冲区不是特别满，则每个事务处理只需要一个同步写入。出现合并这种情况时，每个事务处理需要的同步写入可能少于一个。
- 因为重做日志缓冲区可能在 COMMIT 之前已刷新，所以事务处理的大小不会影响实际 COMMIT 操作所需的时间量。

注：回退事务处理并不会触发 LGWR 写入磁盘。Oracle Server 总是在进行故障恢复时回退未提交的更改。如果在执行回退后且在磁盘上记录回退条目记录之前发生了故障，由于缺少提交记录，所以完全能确保回退事务处理所做的更改。



## Oracle DB 体系结构小结

Oracle DB 由实例及其关联的数据库组成:

- 实例由 SGA 和后台进程组成
  - **SGA:** 数据库缓冲区高速缓存、重做日志缓冲区、共享池等
  - **后台进程:** SMON、PMON、DBW $n$ 、CKPT、LGWR 等
- 数据库由下存储结构组成:
  - **逻辑:** 表空间、方案、段、区和 Oracle 块
  - **物理:** 数据文件、控制文件、重做日志文件

用户通过应用程序访问 Oracle DB 时，服务器进程会代表用户进程与实例通信。



---

## 附加练习和解答

---

## 目录

|                  |    |
|------------------|----|
| 附加练习.....        | 3  |
| 附加练习.....        | 4  |
| 附加练习：案例学习.....   | 10 |
| 附加练习解答.....      | 13 |
| 附加练习解答.....      | 14 |
| 附加练习：案例学习解答..... | 20 |

## 附加练习

在“管理方案对象”和“处理大型数据集”两课中学习数据操纵语言(DML)和数据定义语言(DDL)语句之后，就可以做下面的补充练习了。

注：请运行练习文件夹中的 `lab_ap_cre_special_sal.sql`、  
`lab_ap_cre_sal_history.sql` 和 `lab_ap_cre_mgr_history.sql` 脚本  
来创建 `SPECIAL_SAL`、`SAL_HISTORY` 和 `MGR_HISTORY` 表。

## 附加练习

- 人力资源部门希望在进行行业薪金调查之后得到一份某些工资过低的雇员的名单，其中应包含雇员薪金历史记录和经理薪金历史记录。所以，您需要完成下列任务：  
编写一条语句来完成以下工作：
  - 在 EMPLOYEES 表中检索雇员 ID 大于或等于 200 的那些雇员的雇员 ID、受雇日期、薪金和其经理 ID 之类的详细资料。
  - 如果薪金少于 \$5,000，则将雇员 ID 和薪金之类的详细资料插入在 SPECIAL\_SAL 表中。
  - 将雇员 ID、受雇日期和薪金之类的详细资料插入在 SAL\_HISTORY 表中。
  - 将雇员 ID、经理 ID 和薪金之类的详细资料插入在 MGR\_HISTORY 表中。
- 在 SPECIAL\_SAL、SAL\_HISTORY 和 MGR\_HISTORY 表中进行查询，查看插入的记录。

SPECIAL\_SAL

|   | EMPLOYEE_ID | SALARY |
|---|-------------|--------|
| 1 | 200         | 4400   |

SAL\_HISTORY

|   | EMPLOYEE_ID | HIRE_DATE   | SALARY |
|---|-------------|-------------|--------|
| 1 | 201         | 17-FEB-1996 | 13000  |
| 2 | 202         | 17-AUG-1997 | 6000   |
| 3 | 203         | 07-JUN-1994 | 6500   |
| 4 | 204         | 07-JUN-1994 | 10000  |
| 5 | 205         | 07-JUN-1994 | 12000  |
| 6 | 206         | 07-JUN-1994 | 8300   |

## 附加练习（续）

MGR\_HISTORY

|  | EMPLOYEE_ID | MANAGER_ID | SALARY |
|--|-------------|------------|--------|
|  | 201         | 100        | 13000  |
|  | 202         | 201        | 6000   |
|  | 203         | 101        | 6500   |
|  | 204         | 101        | 10000  |
|  | 205         | 101        | 12000  |
|  | 206         | 205        | 8300   |

3. Nita 是 DBA，她要求您创建一个表，该表上有主键约束条件，但是她希望将索引命名为不同于约束条件的名称。根据下面的实例图表创建 LOCATIONS\_NAMED\_INDEX 表。将 PRIMARY KEY 列的索引命名为 LOCATIONS\_PK\_IDX。

| 列名   | Deptno | Dname    |
|------|--------|----------|
| 主键   | Yes    |          |
| 数据类型 | Number | VARCHAR2 |
| 长度   | 4      | 30       |

4. 在 USER\_INDEXES 表中进行查询，显示 LOCATIONS\_NAMED\_INDEX 表的 INDEX\_NAME。

INDEX\_NAME TABLE\_NAME  
1 LOCATIONS\_PK\_IDX LOCATIONS\_NAMED\_INDEX

## 附加练习（续）

学习日期时间函数之后，就可以做下面的补充练习了。

您受雇于一家跨国公司，新任运营副总裁希望知道所有分公司所在的不同时区。  
新任副总裁要求您完成以下工作：

5. 更改会话，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY HH24:MI:SS。

6.

- a. 编写几个查询来显示下列时区的时区偏移量 (TZ\_OFFSET):
  - 澳大利亚/悉尼

|                                            |
|--------------------------------------------|
| <code>TZ_OFFSET('AUSTRALIA/SYDNEY')</code> |
| 1 +10:00                                   |

- 智利/复活节岛

|                                              |
|----------------------------------------------|
| <code>TZ_OFFSET('CHILE/EASTERISLAND')</code> |
| 1 -06:00                                     |

- b. 更改会话，将 TIME\_ZONE 参数值设置为澳大利亚/悉尼的时区偏移量。

- c. 显示此会话的 SYSDATE、CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**输出可能有所不同，这取决于执行命令的日期。

| SYSDATE              | CURRENT_DATE         | CURRENT_TIMESTAMP                      | LOCALTIMESTAMP                  |
|----------------------|----------------------|----------------------------------------|---------------------------------|
| 02-JUL-2009 17:11:46 | 02-JUL-2009 20:11:46 | 02-JUL-09 08.11.46.000000000 PM +10:00 | 02-JUL-09 08.11.46.000000000 PM |

- d. 更改会话，将 TIME\_ZONE 参数值设置为智利/复活节岛的时区偏移量。

**注：**上面这个问题的结果因日期不同而有所不同，因此某些情况下，可能与学员得到的实际结果不一致。而且，各个国家（地区）的时区偏移量也因夏令时而有所不同。

- e. 显示此会话的 SYSDATE、CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**输出可能有所不同，这取决于执行命令的日期。

| SYSDATE              | CURRENT_DATE         | CURRENT_TIMESTAMP                      | LOCALTIMESTAMP                  |
|----------------------|----------------------|----------------------------------------|---------------------------------|
| 02-JUL-2009 17:12:33 | 02-JUL-2009 04:12:33 | 02-JUL-09 04.12.33.000000000 AM -06:00 | 02-JUL-09 04.12.33.000000000 AM |

## 附加练习（续）

- f. 更改会话，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY。

### 附注

- 注意上面这个问题中的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP 都与会话时区相关。但 SYSDATE 与会话时区无关。
- 上面这个问题的结果因日期不同而有所不同，因此某些情况下，可能与学员得到的实际结果不一致。而且，各个国家（地区）的时区偏移也因夏令时而有所不同。

7. 人力资源部门需要一月份要进行复查的雇员的名单，所以，要求您完成以下工作：

编写一个查询，显示一月份进入公司的雇员的姓氏、受雇日期的月份和受雇日期，而不考虑受雇年份。

|    | LAST_NAME | EXTRACT(MONTHFROMHIRE_DATE) | HIRE_DATE     |
|----|-----------|-----------------------------|---------------|
| 1  | Grant     |                             | 1 13-JAN-2000 |
| 2  | De Haan   |                             | 1 13-JAN-1993 |
| 3  | Hunold    |                             | 1 03-JAN-1990 |
| 4  | Landry    |                             | 1 14-JAN-1999 |
| 5  | Davies    |                             | 1 29-JAN-1997 |
| 6  | Partners  |                             | 1 05-JAN-1997 |
| 7  | Zlotkey   |                             | 1 29-JAN-2000 |
| 8  | Tucker    |                             | 1 30-JAN-1997 |
| 9  | King      |                             | 1 30-JAN-1996 |
| 10 | Marvins   |                             | 1 24-JAN-2000 |
| 11 | Fox       |                             | 1 24-JAN-1998 |
| 12 | Johnson   |                             | 1 04-JAN-2000 |
| 13 | Taylor    |                             | 1 24-JAN-1998 |
| 14 | Sarchand  |                             | 1 27-JAN-1996 |

## 附加练习（续）

学习高级子查询之后，就可以做下面的补充练习了。

8. CEO 需要一份报表，其中应列出在公司利润分红中收入占前三名的人员。您负责向 CEO 提供一份名单。编写一个查询，显示 EMPLOYEES 表中薪金占前三名的人员。显示他们的姓氏和薪金。

|   | LAST_NAME | SALARY |
|---|-----------|--------|
| 1 | King      | 24000  |
| 2 | Kochhar   | 17000  |
| 3 | De Haan   | 17000  |

9. 根据当地法规的规定，加利福尼亚州的福利已发生了变化。所以福利代表要求您提供一份受影响人员的名单。

编写一个查询，显示在加利福尼亚州工作的雇员的雇员 ID 和姓氏。

**提示：** 使用标量子查询。

|    | EMPLOYEE_ID | LAST_NAME   |
|----|-------------|-------------|
| 1  | 198         | O'Connell   |
| 2  | 199         | Grant       |
| 3  | 120         | Weiss       |
| 4  | 121         | Fripp       |
| 5  | 122         | Kaufling    |
| 6  | 123         | Vollman     |
| 7  | 124         | Mourgos     |
| 8  | 125         | Nayer       |
| 9  | 126         | Mikkilineni |
| 10 | 127         | Landry      |
| 11 | 128         | Markle      |

10. Nita (DBA) 要求删除数据库中的过时信息。她认为过时雇员记录是多余的记录。您需要完成如下工作：

编写一个查询，通过在 JOB\_HISTORY 表中查找某雇员的 MIN(START\_DATE) 来删除该雇员最早的 JOB\_HISTORY 行。只删除至少调换过两次职务的那些雇员的记录。

**提示：** 使用相关的 DELETE 命令。

## 附加练习（续）

11. 人力资源副总裁需要一份完整的雇佣记录，用来准备年度雇员表彰大会的演讲。这位副总裁打来一个电话，让您不要执行 DBA 的命令。

此时应回退该事务处理。

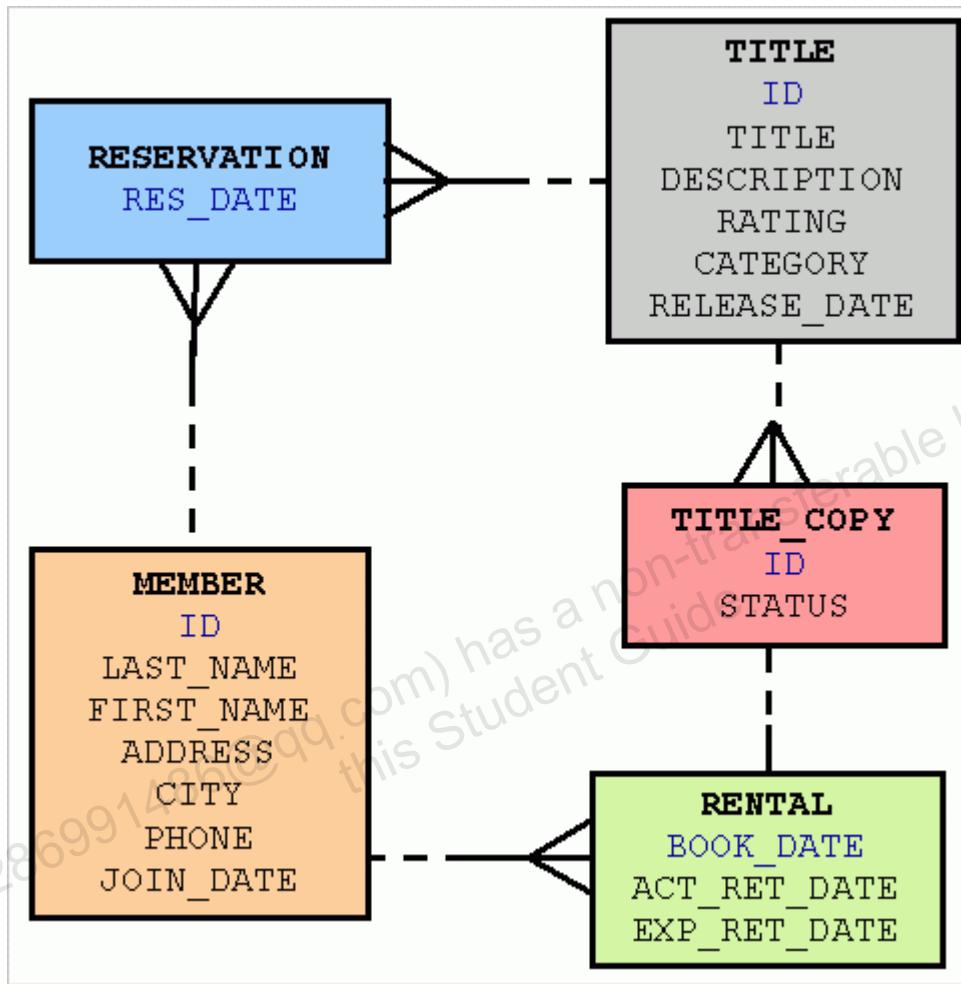
12. 经济不景气迫使管理层采取各种措施来削减成本。CEO 希望复查公司中薪金最高的职务。您负责向 CEO 提供一份名单，名单中的人员取决于以下规定：编写一个查询，显示某些职务的职务 ID，这些职务的最高薪金高于全公司最高薪金的一半以上。请使用 WITH 子句编写此查询。将该查询命名为 MAX\_SAL\_CALC。

|   | JOB_TITLE                     | JOB_TOTAL |
|---|-------------------------------|-----------|
| 1 | President                     | 24000     |
| 2 | Administration Vice President | 17000     |
| 3 | Sales Manager                 | 14000     |
| 4 | Marketing Manager             | 13000     |

## 附加练习：案例学习

在“SQL 基础 I”课程的案例学习中，您为录像带应用程序创建了一组数据库表。此外，您在音像店数据库中插入、更新和删除了某些记录，并且生成了一份报表。

以下是您为录像带应用程序创建的表和列的图表：



注：首先运行 labs 文件夹中的 `dropvid.sql` 脚本来删除现有表。然后运行 labs 文件夹中的 `buildvid.sql` 脚本来创建、填充这些表。

## 附加练习：案例学习（续）

1. 通过运行报表来显示表及其列定义的列表，验证是否已正确创建表。

| TABLE_NAME     | COLUMN_NAME  | DATA_TYPE | NULLABLE |
|----------------|--------------|-----------|----------|
| 1 MEMBER       | MEMBER_ID    | NUMBER    | N        |
| 2 MEMBER       | LAST_NAME    | VARCHAR2  | N        |
| 3 MEMBER       | FIRST_NAME   | VARCHAR2  | Y        |
| 4 MEMBER       | ADDRESS      | VARCHAR2  | Y        |
| 5 MEMBER       | CITY         | VARCHAR2  | Y        |
| 6 MEMBER       | PHONE        | VARCHAR2  | Y        |
| 7 MEMBER       | JOIN_DATE    | DATE      | N        |
| 8 RENTAL       | BOOK_DATE    | DATE      | N        |
| 9 RENTAL       | COPY_ID      | NUMBER    | N        |
| 10 RENTAL      | MEMBER_ID    | NUMBER    | N        |
| 11 RENTAL      | TITLE_ID     | NUMBER    | N        |
| 12 RENTAL      | ACT_RET_DATE | DATE      | Y        |
| 13 RENTAL      | EXP_RET_DATE | DATE      | Y        |
| 14 RESERVATION | RES_DATE     | DATE      | N        |
| 15 RESERVATION | MEMBER_ID    | NUMBER    | N        |
| 16 RESERVATION | TITLE_ID     | NUMBER    | N        |

2. 验证数据字典中是否存在 MEMBER\_ID\_SEQ 和 TITLE\_ID\_SEQ 序列。

| SEQUENCE_NAME     |
|-------------------|
| 1 DEPARTMENTS_SEQ |
| 2 EMPLOYEES_SEQ   |
| 3 LOCATIONS_SEQ   |
| 4 MEMBER_ID_SEQ   |
| 5 TITLE_ID_SEQ    |

3. 创建只能访问各自租借影片的用户。创建名为 Carmen 的用户，授予他在 RENTAL 表中的选择权限。

**注：**确保用户名的前缀以数据库帐户开头。例如，如果您是用户 oraxx，则创建名为 oraxx\_Carmen 的用户。

4. 在 TITLE 表中添加价格列 (number 4,2)，用来存储租录像带的价格。
5. 添加 CATEGORY 表，用来存储 CATEGORY\_ID 和 CATEGORY\_DESCRIPTION。该表包含 TITLE 表中 CATEGORY 列的外键。
6. 在数据字典中选择所有表。
7. 现实中不再需要存储预约信息了。您可以删除该表。

## 附加练习：案例学习（续）

8. 创建 RENTAL\_HISTORY 表，用来存储会员最近 6 个月的详细租借信息。  
(提示：可以复制 RENTAL 表。)
9. 显示上月每类租借最多的 10 部录像带的清单。

|   | CATEGORY | TITLE                    |
|---|----------|--------------------------|
| 1 | ACTION   | Soda Gang                |
| 2 | CHILD    | Willie and Christmas Too |
| 3 | COMEDY   | My Day Off               |
| 4 | SCIFI    | Alien Again              |
| 5 | SCIFI    | Interstellar Wars        |

10. 您要在会员退录像带时间晚了 6 天时计算补交费用（录像带价格/天）。

|   | TITLE             | MEMBER_ID | PRICE  | LATEFEE |
|---|-------------------|-----------|--------|---------|
| 1 | Alien Again       | 101       | (null) | (null)  |
| 2 | My Day Off        | 102       | (null) | (null)  |
| 3 | Interstellar Wars | 101       | (null) | (null)  |

11. 显示租借了 2 次或 2 次以上的会员名单。

|   | MEMBER_ID | LAST_NAME | FIRST_NAME |
|---|-----------|-----------|------------|
| 1 | 101       | Velasquez | Carmen     |

12. 显示状态为“已租借”的录像带清单。

|   | TITLE             |
|---|-------------------|
| 1 | Alien Again       |
| 2 | My Day Off        |
| 3 | Interstellar Wars |

13. 显示电话号码中包含“99”的会员的名单。

|   | POSITION | MEMBER_ID | LAST_NAME | FIRST_NAME |
|---|----------|-----------|-----------|------------|
| 1 | 1        | 101       | Velasquez | Carmen     |
| 2 | 1        | 106       | Urguhart  | Molly      |
| 3 | 1        | 109       | Catchpole | Antoinette |

## 附加练习解答

在“管理方案对象”和“处理大型数据集”两课中学习数据操纵语言(DML)和数据定义语言(DDL)语句之后，就可以做下面的补充练习了。

注：请运行练习文件夹中的 lab\_ap\_cre\_special\_sal.sql、  
lab\_ap\_cre\_sal\_history.sql 和 lab\_ap\_cre\_mgr\_history.sql 脚本  
来创建 SPECIAL\_SAL、SAL\_HISTORY 和 MGR\_HISTORY 表。

## 附加练习解答

- 人力资源部门希望在进行行业薪金调查之后得到一份某些工资过低的雇员的名单，其中应包含雇员薪金历史记录和经理薪金历史记录。所以，您需要完成下列任务：

编写一条语句来完成以下工作：

- 在 EMPLOYEES 表中检索雇员 ID 大于或等于 200 的那些雇员的雇员 ID、受雇日期、薪金和其经理 ID 之类的详细资料。
- 如果薪金少于 \$5,000，则将雇员 ID 和薪金之类的详细资料插入在 SPECIAL\_SAL 表中。
- 将雇员 ID、受雇日期和薪金之类的详细资料插入在 SAL\_HISTORY 表中。
- 将雇员 ID、经理 ID 和薪金之类的详细资料插入在 MGR\_HISTORY 表中。

```
INSERT ALL
WHEN SAL < 5000 THEN
 INTO special_sal VALUES (EMPID, SAL)
ELSE
 INTO sal_history VALUES(EMPID, HIREDATE, SAL)
 INTO mgr_history VALUES(EMPID, MGR, SAL)
SELECT employee_id EMPID, hire_date HIREDATE,
 salary SAL, manager_id MGR
FROM employees
WHERE employee_id >=200;
```

- 在 SPECIAL\_SAL、SAL\_HISTORY 和 MGR\_HISTORY 表中进行查询，查看插入的记录。

```
SELECT * FROM special_sal;
SELECT * FROM sal_history;
SELECT * FROM mgr_history;
```

## 附加练习解答（续）

3. Nita 是 DBA，她要求您创建一个表，该表上有主键约束条件，但是她希望将索引命名为不同于约束条件的名称。根据下面的实例图表创建 LOCATIONS\_NAMED\_INDEX 表。将 PRIMARY KEY 列的索引命名为 LOCATIONS\_PK\_IDX。

| 列名   | Deptno | Dname    |
|------|--------|----------|
| 主键   | Yes    |          |
| 数据类型 | Number | VARCHAR2 |
| 长度   | 4      | 30       |

```
CREATE TABLE LOCATIONS_NAMED_INDEX
(location_id NUMBER(4) PRIMARY KEY USING INDEX
(CREATE INDEX locations_pk_idx ON
LOCATIONS_NAMED_INDEX(location_id)),
location_name VARCHAR2(20));
```

4. 在 USER\_INDEXES 表中进行查询，显示 LOCATIONS\_NAMED\_INDEX 表的 INDEX\_NAME。

```
SELECT INDEX_NAME, TABLE_NAME
FROM USER_INDEXES
WHERE TABLE_NAME = 'LOCATIONS_NAMED_INDEX';
```

## 附加练习解答（续）

学习日期时间函数之后，就可以做下面的补充练习了。

您受雇于一家跨国公司，新任运营副总裁希望知道所有分公司所在的不同时区。新任副总裁要求您完成以下工作：

5. 对会话进行更改，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY  
HH24:MI:SS。

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

6.
  - a. 编写几个查询来显示下列时区的时区偏移量 (TZ\_OFFSET):
    - 澳大利亚/悉尼

```
SELECT TZ_OFFSET ('Australia/Sydney') from dual;
```

- 智利/复活节岛

```
SELECT TZ_OFFSET ('Chile/EasterIsland') from dual;
```

- b. 更改会话，将 TIME\_ZONE 参数值设置为澳大利亚/悉尼的时区偏移量。

```
ALTER SESSION SET TIME_ZONE = '+10:00';
```

- c. 显示此会话的 SYSDATE、CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**输出可能有所不同，这取决于执行命令的日期。

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

- d. 更改会话，将 TIME\_ZONE 参数值设置为智利/复活节岛的时区偏移量。

**注：**上面这个问题的结果因日期不同而有所不同，因此某些情况下，可能与学员得到的实际结果不一致。而且，各个国家（地区）的时区偏移量也因夏令时而有所不同。

```
ALTER SESSION SET TIME_ZONE = '-06:00';
```

## 附加练习解答（续）

- e. 显示此会话的 SYSDATE、CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP。

**注：**输出可能有所不同，这取决于执行命令的日期。

```
SELECT SYSDATE, CURRENT_DATE, CURRENT_TIMESTAMP,
LOCALTIMESTAMP FROM DUAL;
```

- f. 更改会话，将 NLS\_DATE\_FORMAT 设置为 DD-MON-YYYY。

```
ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY';
```

### 附注

- 注意上面这个问题中的 CURRENT\_DATE、CURRENT\_TIMESTAMP 和 LOCALTIMESTAMP 都与会话时区相关。但 SYSDATE 与会话时区无关。
- 上面这个问题的结果因日期不同而有所不同，因此某些情况下，可能与学员得到的实际结果不一致。而且，各个国家（地区）的时区偏移量也因夏令时而有所不同。

7. 人力资源部门需要一月份要进行复查的雇员的名单，所以，要求您完成以下工作：

编写一个查询，显示一月份进入公司的雇员的姓氏、受雇日期的月份和受雇日期，而不考虑受雇年份。

```
SELECT last_name, EXTRACT (MONTH FROM HIRE_DATE), HIRE_DATE
FROM employees
WHERE EXTRACT (MONTH FROM HIRE_DATE) = 1;
```

学习高级子查询之后，就可以做下面的补充练习了。

**注：**如果已使用 code\_05\_12\_sb.sql 将 HIRE\_DATE 列转换为 TIMESTAMP，则 HIRE\_DATE 列的显示可能是 13-JAN-00 12.00.00.000000000 AM

8. CEO 需要一份报表，其中应列出在公司利润分红中收入占前三名的人员。您负责向 CEO 提供一份名单。

编写一个查询，显示 EMPLOYEES 表中薪金占前三名的人员。显示他们的姓氏和薪金。

```
SELECT last_name, salary
 FROM employees e
 WHERE 3 > (SELECT COUNT (*)
 FROM employees
 WHERE e.salary < salary);
```

## 附加练习解答（续）

9. 根据当地法规的规定，加利福尼亚州的福利已发生了变化。所以福利代表要求您提供一份受影响人员的名单。编写一个查询，显示在加利福尼亚州工作的雇员的雇员 ID 和姓氏。

**提示：** 使用标量子查询。

```
SELECT employee_id, last_name
 FROM employees e
 WHERE ((SELECT location_id
 FROM departments d
 WHERE e.department_id = department_id)
 IN (SELECT location_id
 FROM locations l
 WHERE state_province = 'California'));
```

10. Nita (DBA) 要求删除数据库中的过时信息。她认为过时雇员记录是多余的记录。您需要完成如下工作：

编写一个查询，通过在 JOB\_HISTORY 表中查找某雇员的 MIN(START\_DATE) 来删除该雇员最早的 JOB\_HISTORY 行。只删除至少调换过两次职务的那些雇员的记录。

**提示：** 使用相关的 DELETE 命令。

```
DELETE FROM job_history JH
 WHERE employee_id =
 (SELECT employee_id
 FROM employees E
 WHERE JH.employee_id = E.employee_id
 AND START_DATE = (SELECT MIN(start_date)
 FROM job_history JH
 WHERE JH.employee_id =
 E.employee_id)
 AND 3 > (SELECT COUNT(*)
 FROM job_history JH
 WHERE JH.employee_id =
 E.employee_id
 GROUP BY EMPLOYEE_ID
 HAVING COUNT(*) >= 2));
```

## 附加练习解答（续）

11. 人力资源副总裁需要一份完整的雇佣记录，用来准备年度雇员表彰大会的演讲。这位副总裁打来一个电话，让您不要执行 DBA 的命令。

此时应回退该事务处理。

```
ROLLBACK;
```

12. 经济不景气迫使管理层采取各种措施来削减成本。CEO 希望复查公司中薪金最高的职务。您负责向 CEO 提供一份名单，名单中的人员取决于以下规定：编写一个查询，显示某些职务的职务 ID，这些职务的最高薪金高于全公司最高薪金的一半以上。请使用 WITH 子句编写此查询。将该查询命名为 MAX\_SAL\_CALC。

```
WITH
MAX_SAL_CALC AS (SELECT job_title, MAX(salary) AS
job_total
FROM employees, jobs
WHERE employees.job_id = jobs.job_id
GROUP BY job_title)
SELECT job_title, job_total
FROM MAX_SAL_CALC
WHERE job_total > (SELECT MAX(job_total) * 1/2
FROM MAX_SAL_CALC)
ORDER BY job_total DESC;
```

## 附加练习：案例学习解答

首先运行 labs 文件夹中的 dropvid.sql 脚本来删除现有表。然后运行 labs 文件夹中的 buildvid.sql 脚本来创建、填充这些表。

- 通过运行报表来显示表及其列定义的列表，验证是否已正确创建表。

```
SELECT table_name,column_name,data_type,nullable
FROM user_tab_columns
WHERE table_name
IN ('MEMBER','TITLE','TITLE_COPY','RENTAL','RESERVATION');
```

- 验证数据字典中是否存在 MEMBER\_ID\_SEQ 和 TITLE\_ID\_SEQ 序列。

```
SELECT sequence_name FROM user_sequences;
```

- 创建只能访问各自租借影片的用户。创建名为 Carmen 的用户，授予他在 RENTAL 表中的选择权限。

**注：**确保用户名的前缀以数据库帐户开头。例如，如果您是用户 oraxx，则创建名为 oraxx\_Carmen 的用户。

```
CREATE USER oraxx_carmen IDENTIFIED BY oracle ;
GRANT select ON rental TO oraxx_carmen;
```

- 在 TITLE 表中添加价格列 (number 4,2)，用来存储租录像带的价格。

```
ALTER TABLE title ADD (price NUMBER (6))
```

- 添加 CATEGORY 表，用来存储 CATEGORY\_ID 和 CATEGORY\_DESCRIPTION。该表包含 TITLE 表中 CATEGORY 列的外键。

```
CREATE TABLE CATEGORY
(
 "CATEGORY_ID" NUMBER(6,0) NOT NULL ENABLE,
 "CATEGORY_DESCRIPTION" VARCHAR2(4000 BYTE),
 CONSTRAINT "CATEGORY_PK" PRIMARY KEY ("CATEGORY_ID"))
```

- 在数据字典中选择所有表。

```
SELECT table_name FROM user_tables order by table_name;
```

- 现实中不再需要存储预约信息了。您可以删除该表。

```
DROP TABLE reservation cascade constraints;
```

## 附加练习：案例学习解答（续）

8. 创建 RENTAL\_HISTORY 表，用来存储会员最近 6 个月的详细租借信息。  
 (提示：可以复制 RENTAL 表。)

```
CREATE TABLE rental_history AS SELECT * FROM rental WHERE '1' = '1'
```

9. 显示上月每类租借最多的 10 部录像带的清单。

```
SELECT t.CATEGORY, t.TITLE
FROM TITLE t, RENTAL r
WHERE t.TITLE_ID = r.TITLE_ID AND
 r.BOOK_DATE > (SYSDATE - 30) AND
 rownum < 10
ORDER BY category;
```

10. 您要在会员退录像带时间晚了 6 天时计算补交费用（录像带价格/天）。

```
SELECT t.title, m.member_id, t.price, (t.price*6) latefee
FROM title t, member m, rental r
WHERE t.title_id = r.title_id AND
 m.member_id = r.member_id AND
 r.act_ret_date IS NULL;
```

11. 显示租借了 2 次或 2 次以上的会员名单。

```
SELECT member_id, last_name, first_name FROM member m
WHERE 2 <= (SELECT COUNT(*) FROM rental_history WHERE
member_id = m.member_id);
```

12. 显示状态为“已租借”的录像带清单。

```
SELECT t.title
FROM title t
JOIN (SELECT title_id, status FROM title_copy) b
ON t.title_id = b.title_id AND b.status = 'RENTED';
```

13. 显示电话号码中包含“99”的会员的名单。

```
SELECT REGEXP_COUNT(phone, '99', 1, 'i') position, member_id,
last_name, first_name
FROM member
WHERE REGEXP_COUNT(phone, '99', 1, 'i') > 0;
```

