

## Oracle Database 11g: SQL 基础 I

学生指南第 2 册

D49996CN20

版本 2.0

2010 年 5 月

D67006

**ORACLE®**

## 作者

Salome Clement  
Brian Pottle  
Puja Singh

## 技术撰稿人和审稿人

Anjulaponni Azhagulekshmi  
Clair Bennett  
Zarko Cesljas  
Yanti Chang  
Gerlinde Frenzen  
Steve Friedberg  
Joel Goodman  
Nancy Greenberg  
Pedro Neves  
Surya Rekha  
Helen Robertson  
Lauran Serhal  
Tulika Srivastava

## 编辑

Aju Kumar  
Arijit Ghosh

## 制图员

Rajiv Chandrabhanu

## 出版商

Pavithran Adka  
Veena Narasimhan

版权所有 © 2010, Oracle。保留所有权利。

### 免责声明

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
- 课程安排 I-5
- 本课程使用的附录 I-7
- 课程安排 I-8
- Oracle Database 11g: 重点领域 I-9
- Oracle Database 11g I-10
- Oracle Fusion Middleware I-12
- Oracle Enterprise Manager Grid Control I-13
- Oracle BI Publisher I-14
- 课程安排 I-15
- 关系和对象关系数据库管理系统 I-16
- 在不同介质中存储数据 I-17
- 关系数据库概念 I-18
- 关系数据库的定义 I-19
- 数据模型 I-20
- 实体关系模型 I-21
- 实体关系建模惯例 I-22
- 关联多个表 I-24
- 关系数据库术语 I-26
- 课程安排 I-27
- 使用 SQL 查询数据库 I-28
- SQL 语句 I-29
- SQL 的开发环境 I-30
- 课程安排 I-31
- 人力资源 (HR) 方案 I-32
- 本课程使用的表 I-33
- 课程安排 I-34

Oracle Database 11g 文档 I-35

其它资源 I-36

小结 I-37

练习 I: 概览 I-38

## 1 使用 SQL SELECT 语句检索数据

课程目标 1-2

课程安排 1-3

SQL SELECT 语句的功能 1-4

基本 SELECT 语句 1-5

选择所有列 1-6

选择特定列 1-7

编写 SQL 语句 1-8

列标题的默认设置 1-10

课程安排 1-11

算术表达式 1-12

使用算术运算符 1-13

运算符优先级 1-14

定义空值 1-15

算术表达式中的空值 1-16

课程安排 1-17

定义列别名 1-18

使用列别名 1-19

课程安排 1-20

连接运算符 1-21

文字字符串 1-22

使用文字字符串 1-23

其它引号 (q) 运算符 1-24

重复行 1-25

课程安排 1-26

显示表结构 1-27

使用 DESCRIBE 命令 1-28

小测验 1-29

小结 1-30

练习 1: 概览 1-31

## 2 对数据进行限制和排序

- 课程目标 2-2
- 课程安排 2-3
- 有选择地对行进行限制 2-4
- 对所选行进行限制 2-5
- 使用 WHERE 子句 2-6
- 字符串和日期 2-7
- 比较运算符 2-8
- 使用比较运算符 2-9
- 使用 BETWEEN 运算符的范围条件 2-10
- 使用 IN 运算符的成员条件 2-11
- 使用 LIKE 运算符执行模式匹配 2-12
- 组合通配符字符 2-13
- 使用 NULL 条件 2-14
- 使用逻辑运算符定义条件 2-15
- 使用 AND 运算符 2-16
- 使用 OR 运算符 2-17
- 使用 NOT 运算符 2-18
- 课程安排 2-19
- 优先级规则 2-20
- 课程安排 2-22
- 使用 ORDER BY 子句 2-23
- 排序 2-24
- 课程安排 2-26
- 替代变量 2-27
- 使用单与号替代变量 2-29
- 使用替代变量指定字符值和日期值 2-31
- 指定列名、表达式和文本 2-32
- 使用双与号替代变量 2-33
- 课程安排 2-34
- 使用 DEFINE 命令 2-35
- 使用 VERIFY 命令 2-36
- 小测验 2-37
- 小结 2-38
- 练习 2: 概览 2-39

### 3 使用单行函数定制输出

- 课程目标 3-2
- 课程安排 3-3
- SQL 函数 3-4
- 两种类型的 SQL 函数 3-5
- 单行函数 3-7
- 课程安排 3-9
- 字符函数 3-10
- 大小写转换函数 3-12
- 使用大小写转换函数 3-13
- 字符处理函数 3-14
- 使用字符处理函数 3-15
- 课程安排 3-16
- 数字函数 3-17
- 使用 ROUND 函数 3-18
- 使用 TRUNC 函数 3-19
- 使用 MOD 函数 3-20
- 课程安排 3-21
- 处理日期 3-22
- RR 日期格式 3-23
- 使用 SYSDATE 函数 3-25
- 与日期有关的运算 3-26
- 使用算术运算符处理日期 3-27
- 课程安排 3-28
- 日期处理函数 3-29
- 使用日期函数 3-31
- 使用 ROUND 函数和 TRUNC 函数处理日期 3-32
- 小测验 3-33
- 小结 3-34
- 练习 3: 概览 3-35

## 4 使用转换函数和条件表达式

- 课程目标 4-2
- 课程安排 4-3
- 转换函数 4-4
  - 隐式数据类型转换 4-5
  - 显式数据类型转换 4-7
- 课程安排 4-10
  - 使用 TO\_CHAR 函数处理日期 4-11
  - 日期格式样式的元素 4-12
  - 使用 TO\_CHAR 函数处理日期 4-16
  - 使用 TO\_CHAR 函数处理数字 4-17
  - 使用 TO\_NUMBER 和 TO\_DATE 函数 4-20
  - 将 TO\_CHAR 和 TO\_DATE 函数与 RR 日期格式结合使用 4-22
- 课程安排 4-23
- 嵌套函数 4-24
  - 嵌套函数：示例 1 4-25
  - 嵌套函数：示例 2 4-26
- 课程安排 4-27
- 常规函数 4-28
- NVL 函数 4-29
  - 使用 NVL 函数 4-30
  - 使用 NVL2 函数 4-31
  - 使用 NULLIF 函数 4-32
  - 使用 COALESCE 函数 4-33
- 课程安排 4-36
- 条件表达式 4-37
- CASE 表达式 4-38
- 使用 CASE 表达式 4-39
- DECODE 函数 4-40
- 使用 DECODE 函数 4-41
- 小测验 4-43
- 小结 4-44
- 练习 4：概览 4-45

## 5 使用组函数报告聚集数据

- 课程目标 5-2
- 课程安排 5-3
- 何谓组函数 5-4
- 组函数的类型 5-5
- 组函数：语法 5-6
- 使用 AVG 和 SUM 函数 5-7
- 使用 MIN 和 MAX 函数 5-8
- 使用 COUNT 函数 5-9
- 使用 DISTINCT 关键字 5-10
- 组函数和空值 5-11
- 课程安排 5-12
- 创建数据组 5-13
- 创建数据组：GROUP BY 子句的语法 5-14
- 使用 GROUP BY 子句 5-15
- 按多个列进行分组 5-17
- 对多个列使用 GROUP BY 子句 5-18
- 使用组函数的非法查询 5-19
- 限定组结果 5-21
- 使用 HAVING 子句限定组结果 5-22
- 使用 HAVING 子句 5-23
- 课程安排 5-25
- 嵌套组函数 5-26
- 小测验 5-27
- 小结 5-28
- 练习 5：概览 5-29

## 6 使用联接显示多个表中的数据

- 课程目标 6-2
- 课程安排 6-3
- 获取多个表中的数据 6-4
- 联接类型 6-5
- 使用 SQL:1999 语法将表联接起来 6-6
- 限定不确定的列名 6-7
- 课程安排 6-8
- 创建自然联接 6-9

- 使用自然联接检索记录 6-10
- 使用 USING 子句创建联接 6-11
- 联接列名 6-12
  - 使用 USING 子句检索记录 6-13
  - 在 USING 子句中使用表别名 6-14
- 使用 ON 子句创建联接 6-16
- 使用 ON 子句检索记录 6-17
- 使用 ON 子句创建三向联接 6-18
- 对联接应用附加条件 6-19
- 课程安排 6-20
- 将表联接到自身 6-21
- 使用 ON 子句进行自联接 6-22
- 课程安排 6-23
- 非等值联接 6-24
- 使用非等值联接检索记录 6-25
- 课程安排 6-26
- 使用 OUTER 联接返回没有直接匹配的记录 6-27
- INNER 联接与 OUTER 联接 6-28
- LEFT OUTER JOIN 6-29
- RIGHT OUTER JOIN 6-30
- FULL OUTER JOIN 6-31
- 课程安排 6-32
- 笛卡尔积 6-33
- 生成笛卡尔积 6-34
- 创建交叉联接 6-35
- 小测验 6-36
- 小结 6-37
- 练习 6: 概览 6-39

## 7 使用子查询来解决查询

- 课程目标 7-2
- 课程安排 7-3
- 使用子查询解决问题 7-4
- 子查询语法 7-5
- 使用子查询 7-6
- 使用子查询的准则 7-7

- 子查询的类型 7-8
- 课程安排 7-9
- 单行子查询 7-10
- 执行单行子查询 7-11
- 在子查询中使用组函数 7-12
- 带有子查询的 HAVING 子句 7-13
- 此语句中有什么错误 7-14
- 内部查询没有返回任何行 7-15
- 课程安排 7-16
- 多行子查询 7-17
- 在多行子查询中使用 ANY 运算符 7-18
- 在多行子查询中使用 ALL 运算符 7-19
- 使用 EXISTS 运算符 7-20
- 课程安排 7-21
- 子查询中的空值 7-22
- 小测验 7-23
- 小结 7-24
- 练习 7: 概览 7-25

## 8 使用集合运算符

- 课程目标 8-2
- 课程安排 8-3
- 集合运算符 8-4
- 集合运算符准则 8-5
- Oracle Server 和集合运算符 8-6
- 课程安排 8-7
- 本课中使用的表 8-8
- 课程安排 8-12
- UNION 运算符 8-13
- 使用 UNION 运算符 8-14
- UNION ALL 运算符 8-16
- 使用 UNION ALL 运算符 8-17
- 课程安排 8-18
- INTERSECT 运算符 8-19
- 使用 INTERSECT 运算符 8-20
- 课程安排 8-21

MINUS 运算符 8-22  
使用 MINUS 运算符 8-23  
课程安排 8-24  
匹配 SELECT 语句 8-25  
匹配 SELECT 语句：示例 8-26  
课程安排 8-27  
在集合运算中使用 ORDER BY 子句 8-28  
小测验 8-29  
小结 8-30  
练习 8：概览 8-31

## 9 处理数据

课程目标 9-2  
课程安排 9-3  
数据操纵语言 9-4  
在表中添加新行 9-5  
INSERT 语句语法 9-6  
插入新行 9-7  
插入带有空值的行 9-8  
插入特殊值 9-10  
插入特定日期和时间值 9-11  
创建脚本 9-12  
从其它表中复制行 9-13  
课程安排 9-14  
更改表中的数据 9-15  
UPDATE 语句语法 9-16  
更新表中的行 9-17  
使用子查询更新两列 9-18  
根据另一个表更新行 9-19  
课程安排 9-20  
从表中删除行 9-21  
DELETE 语句 9-22  
从表中删除行 9-23  
根据另一个表删除行 9-24  
TRUNCATE 语句 9-25  
课程安排 9-26

数据库事务处理	9-27
数据库事务处理：开始和结束	9-28
COMMIT 和 ROLLBACK 语句的优点	9-29
显式事务处理控制语句	9-30
将更改回退到某个标记	9-31
隐式事务处理	9-32
执行 COMMIT 或 ROLLBACK 操作之前的数据状态	9-34
执行 COMMIT 操作之后的数据状态	9-35
提交数据	9-36
执行 ROLLBACK 操作之后的数据状态	9-37
执行 ROLLBACK 操作之后的数据状态：示例	9-38
语句级回退	9-39
课程安排	9-40
读一致性	9-41
实施读一致性	9-42
课程安排	9-43
SELECT 语句中的 FOR UPDATE 子句	9-44
FOR UPDATE 子句：示例	9-45
小测验	9-47
小结	9-48
练习 9：概览	9-49

## 10 使用 DDL 语句创建和管理表

课程目标	10-2
课程安排	10-3
数据库对象	10-4
命名规则	10-5
课程安排	10-6
CREATE TABLE 语句	10-7
引用另一个用户的表	10-8
DEFAULT 选项	10-9
创建表	10-10
课程安排	10-11
数据类型	10-12
日期时间数据类型	10-14
课程安排	10-15

包括约束条件 10-16  
约束条件准则 10-17  
定义约束条件 10-18  
NOT NULL 约束条件 10-20  
UNIQUE 约束条件 10-21  
PRIMARY KEY 约束条件 10-23  
FOREIGN KEY 约束条件 10-24  
FOREIGN KEY 约束条件: 关键字 10-26  
CHECK 约束条件 10-27  
CREATE TABLE: 示例 10-28  
违反约束条件 10-29  
课程安排 10-31  
使用子查询创建表 10-32  
课程安排 10-34  
ALTER TABLE 语句 10-35  
只读表 10-36  
课程安排 10-37  
删除表 10-38  
小测验 10-39  
小结 10-40  
练习 10: 概览 10-41

## 11 创建其它方案对象

课程目标 11-2  
课程安排 11-3  
数据库对象 11-4  
什么是视图 11-5  
视图的优点 11-6  
简单视图和复杂视图 11-7  
创建视图 11-8  
从视图中检索数据 11-11  
修改视图 11-12  
创建复杂视图 11-13  
对视图执行 DML 操作的规则 11-14  
使用 WITH CHECK OPTION 子句 11-17  
拒绝 DML 操作 11-18

删除视图 11-20  
练习 11: 第 1 部分概览 11-21  
课程安排 11-22  
序列 11-23  
CREATE SEQUENCE 语句: 语法 11-25  
创建序列 11-26  
NEXTVAL 和 CURRVAL 伪列 11-27  
使用序列 11-29  
高速缓存序列值 11-30  
修改序列 11-31  
修改序列的准则 11-32  
课程安排 11-33  
索引 11-34  
如何创建索引 11-36  
创建索引 11-37  
索引创建准则 11-38  
删除索引 11-39  
课程安排 11-40  
同义词 11-41  
创建对象的同义词 11-42  
创建和删除同义词 11-43  
小测验 11-44  
小结 11-45  
练习 11: 第 2 部分概览 11-46

## 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-22
设置 SQL 代码的格式	C-23
使用片段	C-24
使用片段：示例	C-25
调试过程和函数	C-26
数据库报表	C-27
创建用户定义报表	C-29
搜索引擎和外部工具	C-30
设置首选项	C-31
重置 SQL Developer 布局	C-32
小结	C-33

#### 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、START 命令	D-15
SERVERROUTPUT 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: Oracle 联接语法**

- 课程目标 F-2
- 获取多个表中的数据 F-3
- 笛卡尔积 F-4
- 生成笛卡尔积 F-5
- Oracle 专用联接的类型 F-6
- 使用 Oracle 语法联接表 F-7
- 限定不确定的列名 F-8
- 等值联接 F-9
- 使用等值联接检索记录 F-10
- 使用等值联接检索记录: 示例 F-11
- 使用 AND 运算符的附加搜索条件 F-12
- 联接两个以上的表 F-13
- 非等值联接 F-14
- 使用非等值联接检索记录 F-15
- 使用外部联接返回没有直接匹配的记录 F-16
- 外部联接语法 F-17

- 使用外部联接 F-18
- 外部联接：另一个示例 F-19
- 将表联接到自身 F-20
- 自联接：示例 F-21
- 小结 F-22
- 练习 F：概览 F-23

#### 附录 AP：附加练习和解答

#### 索引



# 9

## 处理数据

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本课后，应能完成下列工作：

- 描述各个数据操纵语言 (DML) 语句
- 在表中插入行
- 更新表中的行
- 从表中删除行
- 控制事务处理

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 目标

在本课中，您将学习如何使用数据操纵语言 (DML) 语句在表中插入行、更新表中的现有行以及从表中删除现有行。您还将学习如何使用 COMMIT、SAVEPOINT 和 ROLLBACK 语句控制事务处理。

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据操纵语言

- 在进行以下操作时可以执行 DML 语句：
  - 在表中添加新行
  - 修改表中的现有行
  - 从表中删除现有行
- 一个事务处理由构成一个逻辑工作单元的一组 DML 语句组成。



版权所有 © 2010, Oracle。保留所有权利。

### 数据操纵语言

数据操纵语言 (DML) 是 SQL 的核心部分。当您要在数据库中添加、更新或删除数据时，就需要执行 DML 语句。构成一个逻辑工作单元的一组 DML 语句被称为一个事务处理。

假定有一个银行数据库。当银行客户从储蓄帐户向支票帐户中划转资金时，该事务处理可能由三个单独的操作组成：减少储蓄帐户金额、增加支票帐户金额以及在事务处理日记帐中记录该事务处理。Oracle Server 必须确保所有这三条 SQL 语句都得以执行，才能使帐户得到正确的结算。如果由于某种原因未能执行事务处理中的某条语句，则必须取消事务处理的其它语句。

#### 附注

- 假设本课中的大多数 DML 语句都不违反表约束条件。本课稍后将对约束条件进行论述。
- 在 SQL Developer 中，单击“Run Script（运行脚本）”图标或按 [F5] 可运行 DML 语句。在“Script Output（脚本输出）”选项卡页上将显示反馈消息。

## 在表中添加新行

DEPARTMENTS

		70 Public Relations	100	1700	<b>新增行</b>
--	--	---------------------	-----	------	------------

	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	

在 DEPARTMENTS 表中  
插入新行



	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	70 Public Relations	100	1700	
2	10 Administration	200	1700	
3	20 Marketing	201	1800	
4	50 Shipping	124	1500	
5	60 IT	103	1400	
6	80 Sales	149	2500	
7	90 Executive	100	1700	
8	110 Accounting	205	1700	
9	190 Contracting	(null)	1700	

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 在表中添加新行

幻灯片图表中演示在 DEPARTMENTS 表中添加新部门的过程。

## INSERT 语句语法

- 使用 INSERT 语句可在表中添加新行：

```
INSERT INTO    table [(column [, column...])]  
VALUES        (value [, value...]);
```

- 使用此语法一次只能插入一行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### INSERT 语句语法

通过发出 INSERT 语句可以在表中添加新行。

在该语法中：

*table* 是表名称

*column* 是表中要填充的列的名称

*value* 是该列相应的值

注：这个带有 VALUES 子句的语句一次只能在表中添加一行。

## 插入新行

- 插入一个新行，此行的每一列都含有值。
- 按照表中列的默认顺序列出这些值。
- (可选) 在 INSERT 子句中列出这些列。

```
INSERT INTO departments(department_id,
    department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- 将字符和日期值放在单引号中。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 插入新行

由于可以插入的新行上的每一列都含有值，因此不需要在 INSERT 子句中使用列的列表。但是，如果不使用列的列表，则必须按照表中列的默认顺序列出值，而且必须为每一列都提供一个值。

```
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)

为清楚起见，请在 INSERT 子句中使用列的列表。

请将字符和日期值放在单引号中，但建议不要将数字值放在单引号中。

## 插入带有空值的行

- 隐式方法：在列的列表中省略该列。

```
INSERT INTO departments (department_id,
                        department_name)
VALUES          (30, 'Purchasing');
1 rows inserted
```

- 显式方法：在 VALUES 子句中指定 NULL 关键字。

```
INSERT INTO departments
VALUES          (100, 'Finance', NULL, NULL);
1 rows inserted
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 插入带有空值的行

方法	说明
隐式	在列的列表中省略该列。
显式	在 VALUES 列表中指定 NULL 关键字，在 VALUES 列表中指定代表字符串和日期的空字符串 ('')。

通过使用 DESCRIBE 命令验证 Null 状态来确保可以在目标列中使用空值。

Oracle Server 会自动强制实施所有数据类型、数据范围和数据完整性约束条件。对于没有显式列出的所有列，在新行中都包含一个空值。

可以按以下顺序对用户输入过程中可能发生的常见错误进行检查：

- NOT NULL 列缺少必需值
- 违反唯一性约束条件或主键约束条件的重复值
- 违反 CHECK 约束条件的任何值
- 为外键维护的引用完整性约束条件
- 数据类型不匹配或者值太长而无法放入列中

## 插入带有空值的行（续）

注：建议使用列的列表，因为这样可以增加 INSERT 语句的可读性和可靠性，也可以减少错误的发生。

## 插入特殊值

SYSDATE 函数用于记录当前日期和时间。

```
INSERT INTO employees (employee_id,
                      first_name, last_name,
                      email, phone_number,
                      hire_date, job_id, salary,
                      commission_pct, manager_id,
                      department_id)
VALUES
      (113,
       'Louis', 'Popp',
       'LPOPP', '515.124.4567',
       SYSDATE, 'AC_ACCOUNT', 6900,
       NULL, 205, 110);
```

1 rows inserted

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 插入特殊值

可以使用函数在表中输入特殊值。

幻灯片示例在 EMPLOYEES 表中记录了雇员 Popp 的信息，其中在 HIRE\_DATE 列中提供的是当前日期和时间。示例中使用 SYSDATE 函数返回数据库服务器的当前日期和时间。此外，也可以使用 CURRENT\_DATE 函数获取会话时区的当前日期。在向表中插入行时，也可以使用 USER 函数。USER 函数会记录当前用户名。

### 确认添加到表的内容

```
SELECT employee_id, last_name, job_id, hire_date, commission_pct
FROM   employees
WHERE  employee_id = 113;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	COMMISSION_PCT
1	113	Popp	AC_ACCOUNT	10-JUL-09	(null)

## 插入特定日期和时间值

- 添加新雇员。

```
INSERT INTO employees
VALUES      (114,
              'Den', 'Raphealy',
              'DRAPHEAL', '515.127.4561',
              TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
              'SA REP', 11000, 0.2, 100, 60);
1 rows inserted
```

- 确认添加的内容。

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT
1	114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	SA REP	11000	0.2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 插入特定日期和时间值

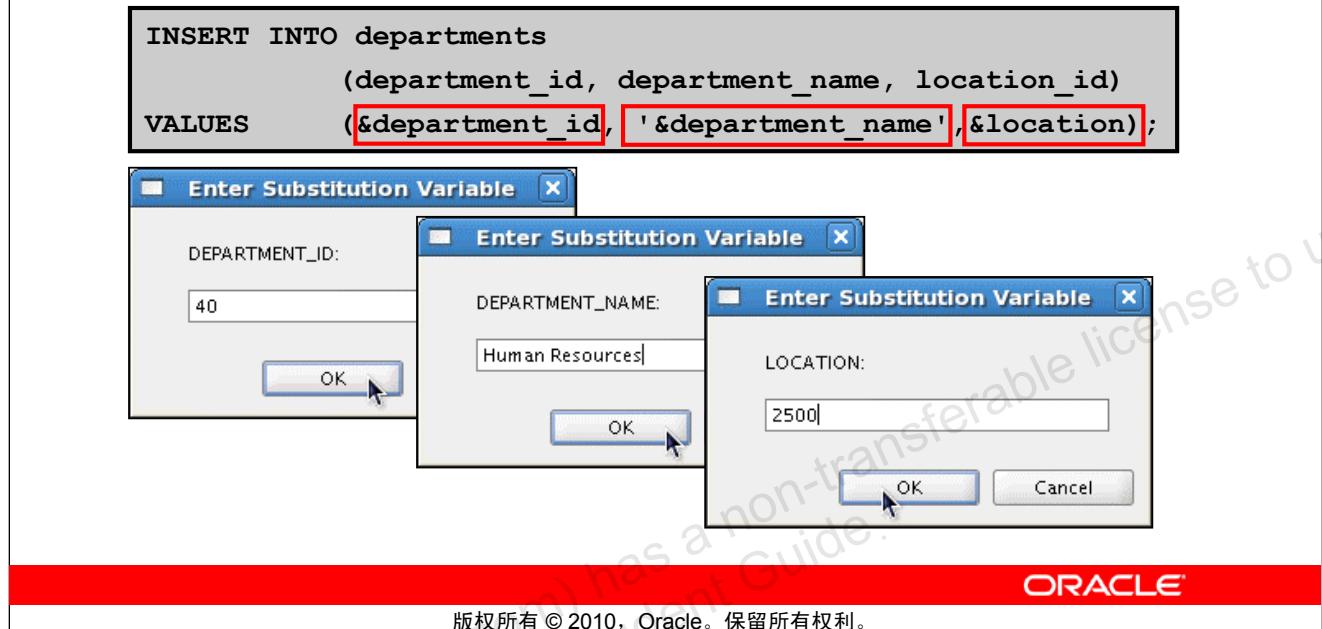
通常使用 DD-MON-RR 格式插入日期值。使用 RR 格式时，系统会自动提供正确的世纪。还可以使用 DD-MON-YYYY 格式提供日期值。建议使用这种格式，因为这样可以明确地指定世纪，而不必依赖于内部 RR 格式逻辑来指定正确的世纪。

如果必须以不同于默认格式的格式输入日期，例如输入另一个世纪或特定时间，则必须使用 TO\_DATE 函数。

幻灯片示例在 EMPLOYEES 表中记录了雇员 Raphealy 的信息，并且将 HIRE\_DATE 列设置为 “February 3, 1999”。

## 创建脚本

- 在 SQL 语句中使用 & 替代来提示用户输入值。
- & 是变量值的占位符。



### 创建脚本

可以将含有替代变量的命令保存到文件中，然后在文件中执行该命令。幻灯片示例在 DEPARTMENTS 表中记录了一个部门的信息。

运行该脚本文件时，系统会提示您为每个“与(&)”替代变量输入值。为替代变量输入值后，单击“OK(确定)”按钮。您输入的值就会被替换到语句中。这样您可以反复运行同一脚本文件，但每次运行该脚本文件时应提供一组不同的值。

## 从其它表中复制行

- 编写带有子查询的 INSERT 语句：

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
SELECT employee_id, last_name, salary, commission_pct
FROM   employees
WHERE  job_id LIKE '%REP%';
```

4 rows inserted

- 请勿使用 VALUES 子句。
- 使 INSERT 子句中的列数与子查询中的列数匹配。
- 将子查询返回的所有行插入到表 sales\_reps 中。

版权所有 © 2010, Oracle。保留所有权利。

### 从其它表中复制行

可以使用 INSERT 语句在一个表中添加一些行，该表的值来自现有表。在幻灯片示例中，若要执行 INSERT INTO 语句，必须首先使用 CREATE TABLE 语句创建 sales\_reps 表。将在“使用 DDL 语句创建和管理表”一课中讨论有关 CREATE TABLE 语句的内容。可以使用子查询来代替 VALUES 子句。

#### 语法

```
INSERT INTO table [ column (, column) ] subquery;
```

在该语法中：

table 是表名称

column 是表中要填充的列的名称

subquery 是向表中返回行的子查询

INSERT 子句中列列表的列数及其数据类型必须与子查询中值的个数及其数据类型相匹配。根据子查询返回的行数，可能会添加一些行，也可能不添加任何行。要为表中的行创建一个副本，请在子查询中使用 SELECT \*:

```
INSERT INTO copy_emp
SELECT *
FROM   employees;
```

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 更改表中的数据

**EMPLOYEES**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

更新 EMPLOYEES 表中的行:

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 更改表中的数据

幻灯片演示了将部门 60 中雇员的部门编号更改为部门 80 的过程。

## UPDATE 语句语法

- 使用 UPDATE 语句修改表中的现有值：

```
UPDATE          table
SET            column = value [, column = value, ...]
[WHERE]        condition;
```

- 如果需要，可以一次更新多行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### UPDATE 语句语法

可以使用 UPDATE 语句修改表中的现有值。

在该语法中：

*table* 是表名称

*column* 是表中要填充的列的名称

*value* 是该列相应的值或子查询

*condition* 标识要更新的行，由列名、表达式、常数、子查询和比较运算符组成

通过查询表来显示更新的行就可以确认更新操作。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“UPDATE”一节。

**注：**通常情况下，请使用 WHERE 子句中的主键列来指定要更新的一行。使用其它列时可能会意外地更新其它的行。例如，用姓名标识 EMPLOYEES 表中的一行是危险的，因为可能有多名雇员具有相同的姓名。

## 更新表中的行

- 如果指定 WHERE 子句，则可以修改特定一行或多行的值：

```
UPDATE employees
SET department_id = 50
WHERE employee_id = 113;
1 rows updated
```

- 如果省略 WHERE 子句，则可以修改表中所有行的值：

```
UPDATE copy_emp
SET department_id = 110;
22 rows updated
```

- 指定 SET column\_name= NULL 可将列值更新为 NULL。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 更新表中的行

如果指定 WHERE 子句，则 UPDATE 语句可以修改特定一行或多行的值。幻灯片示例显示如何将雇员 113 (Popp) 调到部门 50。

如果省略 WHERE 子句，则会修改该表中所有行的值。检查 COPY\_EMP 表中的更新行。

```
SELECT last_name, department_id
FROM copy_emp;
```

	LAST_NAME	DEPARTMENT_ID
1	Whalen	110
2	Hartstein	110
3	Fay	110

...

例如，曾经担任 SA\_REP 职务的某个雇员现在的职务为 IT\_PROG。因此，需更新其 JOB\_ID 并将其佣金字段设置为 NULL。

```
UPDATE employees
SET job_id = 'IT_PROG', commission_pct = NULL
WHERE employee_id = 114;
```

注：COPY\_EMP 表与 EMPLOYEES 表具有相同的数据。

## 使用子查询更新两列

更新雇员 113 的职务和薪金，使其与雇员 205 的职务和薪金一样。

```
UPDATE employees
SET job_id = (SELECT job_id
               FROM employees
               WHERE employee_id = 205),
    salary = (SELECT salary
               FROM employees
               WHERE employee_id = 205)
WHERE employee_id = 113;
1 rows updated
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用子查询更新两列

通过编写多个子查询，可以在 UPDATE 语句的 SET 子句中更新多个列。语法如下：

```
UPDATE table
SET column =
          (SELECT column
           FROM table
           WHERE condition)
      [ ,
        column =
          (SELECT column
           FROM table
           WHERE condition) ]
[WHERE condition] ;
```

幻灯片示例还可以编写如下：

```
UPDATE employees
SET (job_id, salary) = (SELECT job_id, salary
                         FROM employees
                         WHERE employee_id = 205)
WHERE employee_id = 113;
```

## 根据另一个表更新行

在 UPDATE 语句中使用子查询可根据另一个表中的值来更新某一个表中的行值：

```
UPDATE copy_emp
SET department_id = (SELECT department_id
                      FROM employees
                      WHERE employee_id = 100)
WHERE job_id          = (SELECT job_id
                         FROM employees
                         WHERE employee_id = 200);
1 rows updated
```



版权所有 © 2010, Oracle。保留所有权利。

### 根据另一个表更新行

可以在 UPDATE 语句中使用子查询来更新表中的值。幻灯片示例根据 EMPLOYEES 表中的值更新 COPY\_EMP 表。它将与雇员 200 具有相同职务 ID 的所有雇员的部门编号都更改为雇员 100 的当前部门编号。

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 从表中删除行

### 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

### 从 DEPARTMENTS 表中删除一行：

	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

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 从表中删除行

如幻灯片图表所示，此时已从 DEPARTMENTS 表中删除 Contracting 部门（假定没有违反 DEPARTMENTS 表的约束条件）。

## DELETE 语句

使用 DELETE 语句可以从表中删除现有行：

```
DELETE [FROM]    table
[WHERE]          condition;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### DELETE 语句语法

使用 DELETE 语句可以从表中删除现有行。

在该语法中：

*table* 是表名称

*condition* 标识要删除的行，由列名、表达式、常数、子查询和比较运算符组成

**注：**如果没有删除任何行，则会在 SQL Developer 的“Script Output（脚本输出）”选项卡上返回消息“0 rows deleted”。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“DELETE”一节。

## 从表中删除行

- 如果指定 WHERE 子句，则会删除特定行：

```
DELETE FROM departments
WHERE department_name = 'Finance';
1 rows deleted
```

- 如果省略 WHERE 子句，则会删除表中的所有行：

```
DELETE FROM copy_emp;
22 rows deleted
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 从表中删除行

通过在 DELETE 语句中指定 WHERE 子句，可以删除特定行。幻灯片第一个示例从 DEPARTMENTS 表中删除了 Accounting 部门。通过使用 SELECT 语句显示被删除的行，可以确认删除操作。

```
SELECT *
FROM departments
WHERE department_name = 'Finance';
0 rows selected
```

但是，如果省略 WHERE 子句，则会删除该表中的所有行。幻灯片第二个示例从 COPY\_EMP 表中删除了所有行，因为没有指定 WHERE 子句。

#### 示例：

删除 WHERE 子句中标识的行。

```
DELETE FROM employees WHERE employee_id = 114;
1 rows deleted
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
2 rows deleted
```

## 根据另一个表删除行

在 DELETE 语句中使用子查询可根据另一个表中的值来删除某一个表中的行：

```
DELETE FROM employees
WHERE department_id =
    (SELECT department_id
     FROM departments
     WHERE department_name
       LIKE '%Public%');

1 rows deleted
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 根据另一个表删除行

使用子查询可根据另一个表中的值来删除某一个表中的行。幻灯片示例中删除了部门名称包含字符串“Public”的部门的所有雇员。

子查询将根据包含字符串“Public”的部门名称来搜索 DEPARTMENTS 表以找到部门编号。然后，子查询将该部门编号反馈给主查询，主查询将根据该部门编号从 EMPLOYEES 表中删除有关数据行。

## TRUNCATE 语句

- 从表中删除所有行，使表为空并保留表结构不变
- 是数据定义语言 (DDL) 语句而不是 DML 语句，无法轻易将其取消
- 语法：

```
TRUNCATE TABLE table_name;
```

- 示例：

```
TRUNCATE TABLE copy_emp;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### TRUNCATE 语句

清空表的一个更有效的方法就是使用 TRUNCATE 语句。

可以使用 TRUNCATE 语句从表或集群中快速删除所有行。使用 TRUNCATE 语句删除行比使用 DELETE 语句要快，原因如下：

- TRUNCATE 语句是数据定义语言 (DDL) 语句，它不会生成回退信息。本课稍后将对回退信息进行介绍。
- 截断一个表不会触发该表的删除触发器。

如果表是具有引用完整性约束条件的父表，则不能截断表。需要在发出 TRUNCATE 语句之前禁用该约束条件。将在“使用 DDL 语句创建和管理表”一课中讨论有关禁用约束条件的内容。

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

# 数据库事务处理

数据库事务处理由以下语句之一组成：

- 用于对数据进行一次一致更改的 DML 语句
- 一条 DDL 语句
- 一条数据控制语言 (DCL) 语句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据库事务处理

Oracle Server 在事务处理的基础上确保数据的一致性。事务处理令您在更改数据时具有更高的灵活性和控制力，还可以确保在用户进程失败或系统出现故障时保持数据的一致性。

事务处理由一些 DML 语句组成，这些 DML 语句用于对数据进行一次一致更改。例如，在两个帐户之间转移资金时应涉及两个帐户，一个是借方帐户，另一个是贷方帐户，这两个帐户的金额相同。这两个操作必须都失败或都成功，没有借方也就没有贷方。

### 事务处理类型

类型	说明
数据操纵语言 (DML)	由任意数量的 DML 语句组成，Oracle Server 将这些语句视为单个实体或一个逻辑工作单元
数据定义语言 (DDL)	仅由一个 DDL 语句组成
数据控制语言 (DCL)	仅由一个 DCL 语句组成

## 数据库事务处理：开始和结束

- 在执行第一条 DML SQL 语句时开始。
- 在发生下列事件之一时结束：
  - 发出 COMMIT 或 ROLLBACK 语句。
  - 执行 DDL 或 DCL 语句（自动提交）。
  - 用户退出 SQL Developer 或 SQL\*Plus。
  - 系统崩溃。



版权所有 © 2010, Oracle。保留所有权利。

### 数据库事务处理：开始和结束

数据库事务处理何时开始和结束？

事务处理在遇到第一条 DML 语句时开始，在发生以下事件之一时结束：

- 发出 COMMIT 或 ROLLBACK 语句。
- 发出 DDL 语句，例如 CREATE。
- 发出 DCL 语句。
- 用户退出 SQL Developer 或 SQL\*Plus。
- 计算机出现故障或系统崩溃。

一个事务处理结束后，下一个可执行的 SQL 语句会自动启动下一个事务处理。

DDL 语句或 DCL 语句是自动提交的，因此会隐式结束一个事务处理。

## COMMIT 和 ROLLBACK 语句的优点

使用 COMMIT 和 ROLLBACK 语句，您可以：

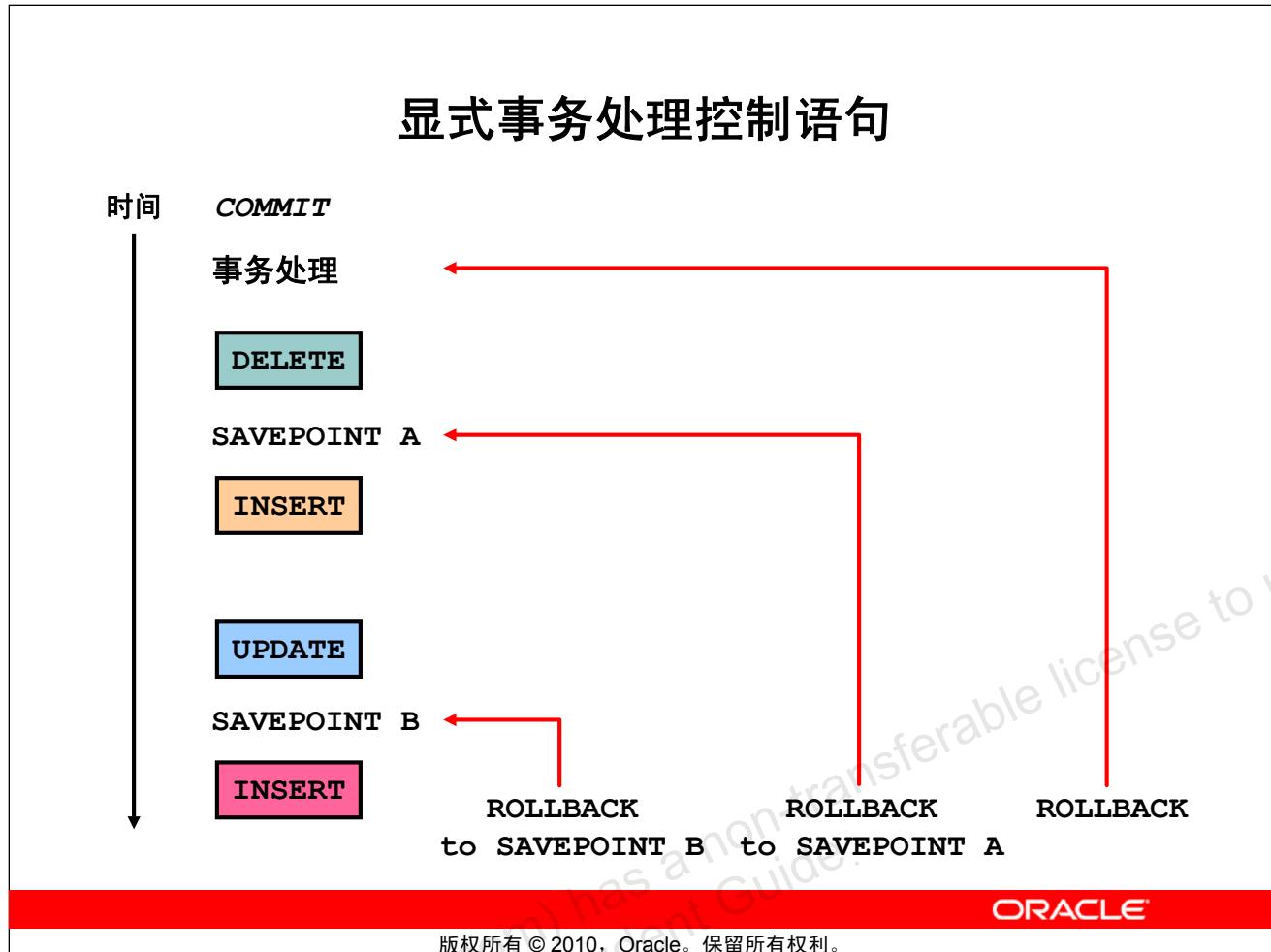
- 确保数据的一致性
- 在使更改变成永久性更改之前预览数据更改
- 按逻辑关系对相关操作进行分组

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### COMMIT 和 ROLLBACK 语句的优点

使用 COMMIT 和 ROLLBACK 语句，您可以控制将数据更改变成永久性更改。



## 显式事务处理控制语句

使用 COMMIT、SAVEPOINT 和 ROLLBACK 语句可以控制事务处理的逻辑。

语句	说明
COMMIT	COMMIT 通过将所有待定的数据更改变成永久性更改来结束当前的事务处理。
SAVEPOINT <i>name</i>	SAVEPOINT <i>name</i> 在当前事务处理中标记一个保存点。
ROLLBACK	ROLLBACK 通过放弃所有待定的数据更改来结束当前的事务处理。
ROLLBACK TO SAVEPOINT <i>name</i>	ROLLBACK TO SAVEPOINT 将当前事务处理回退到指定的保存点，从而放弃在回退到的保存点之后所做的所有更改和/或创建的保存点。如果省略了 TO SAVEPOINT 子句，则 ROLLBACK 语句会回退整个事务处理。因为保存点是逻辑上的保存点，所以无法列出已创建的保存点。

注：不能对 SAVEPOINT 使用 COMMIT 语句。SAVEPOINT 不是符合 ANSI 标准的 SQL。

## 将更改回退到某个标记

- 使用 SAVEPOINT 语句可在当前事务处理中创建一个标记。
- 使用 ROLLBACK TO SAVEPOINT 语句可回退到该标记。

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.

INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```



版权所有 © 2010, Oracle。保留所有权利。

### 将更改回退到某个标记

使用 SAVEPOINT 语句可以在当前事务处理中创建一个标记，该标记会将事务处理分成更小的部分。然后，使用 ROLLBACK TO SAVEPOINT 语句可以放弃该标记之后的待定更改。

请注意，如果创建的第二个保存点的名称与较早保存点的名称相同，则会删除较早保存点。

## 隐式事务处理

- 在下列情况下会发生自动提交：
  - 发出 DDL 语句
  - 发出 DCL 语句
  - 从 SQL Developer 或 SQL\*Plus 正常退出，而没有显式发出 COMMIT 或 ROLLBACK 语句
- 当异常终止 SQL Developer 或 SQL\*Plus 或者系统出现故障时会发生自动回退。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 隐式事务处理

状态	情况
自动提交	DDL 语句或 DCL 语句已发出正常退出 SQL Developer 或 SQL*Plus，而没有显式发出 COMMIT 或 ROLLBACK 命令
自动回退	异常终止 SQL Developer 或 SQL*Plus 或者系统出现故障

注：在 SQL\*Plus 中，AUTOCOMMIT 命令可以在 ON 或 OFF 之间切换。如果设置为 ON，则在执行单个 DML 语句后立即提交该语句。您不能回退更改。如果设置为 OFF，则仍可以显式发出 COMMIT 语句。此外，也可以在发出 DDL 语句或退出 SQL\*Plus 时发出 COMMIT 语句。在 SQL Developer 中会跳过 SET AUTOCOMMIT ON/OFF 命令。只有启用“Autocommit（自动提交）”首选项，才会在从 SQL Developer 正常退出时提交 DML 语句。要启用“Autocommit（自动提交）”，请执行以下步骤：

- 在“Tools（工具）”菜单中，选择“Preferences（首选项）”。在“Preferences（首选项）”对话框中，展开“Database（数据库）”并选择“Worksheet Parameters（工作表参数）”。
- 在右侧窗格中，选择“Autocommit in SQL Worksheet（以 SQL 工作表的形式自动提交）”选项。单击“OK（确定）”。

## 隐式事务处理（续）

### 系统故障

当系统出现故障而中断事务处理时，就会自动回退整个事务处理。这样可以防止因为错误而导致数据发生不必要的更改，还可以使表返回到上一次提交时的状态。Oracle Server 通过这种方式保护表的完整性。

在 SQL Developer 中，通过在“File（文件）”菜单中选择“Exit（退出）”，可以正常退出会话。在 SQL\*Plus 中，在提示符下键入 EXIT 命令可以正常退出。关闭窗口则被解释为异常退出。

## 执行 COMMIT 或 ROLLBACK 操作之前的数据状态

- 可以将数据还原到以前的状态。
- 当前用户可以使用 SELECT 语句查看 DML 操作的结果。
- 其他用户不能查看当前用户发出的 DML 语句的结果。
- 受影响行已锁定，其他用户不能更改受影响行中的数据。



版权所有 © 2010, Oracle。保留所有权利。

## 执行 COMMIT 或 ROLLBACK 操作之前的数据状态

在提交事务处理之前，其间进行的所有数据更改都是临时的。

发出 COMMIT 或 ROLLBACK 语句之前数据的状态如下所述：

- 数据处理操作主要影响数据库缓冲区，因此可以将数据还原到以前的状态。
- 当前用户可以通过对表进行查询来查看数据处理操作的结果。
- 其他用户不能查看当前用户执行的数据处理操作的结果。Oracle Server 会建立读一致性，从而确保每个用户看到的数据都是上次提交时的数据。
- 受影响行已锁定，其他用户不能更改受影响行中的数据。

## 执行 COMMIT 操作之后的数据状态

- 数据更改已保存在数据库中。
- 已改写以前的数据状态。
- 所有用户都可以查看结果。
- 受影响行上的锁已被释放，其他用户可以对这些行进行处理。
- 所有保存点都已被清除。



版权所有 © 2010, Oracle。保留所有权利。

## 执行 COMMIT 操作之后的数据状态

使用 COMMIT 语句可以使所有待定更改变成永久性更改。执行 COMMIT 语句之后会发生以下情况：

- 数据更改已写入数据库中。
- 以前的数据状态不再适用于正常的 SQL 查询。
- 所有用户都可以查看事务处理的结果。
- 受影响行上的锁已被释放，其他用户可以对这些行执行新的数据更改。
- 所有保存点都已被清除。

## 提交数据

- 进行更改:

```
DELETE FROM employees
WHERE employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- 提交更改:

```
COMMIT;
```

```
COMMIT succeeded.
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 提交数据

在幻灯片示例中，从 EMPLOYEES 表中删除了一行并向 DEPARTMENTS 表中添加了新行。通过发出 COMMIT 语句已保存所做的更改。

### 示例:

删除 DEPARTMENTS 表中的部门 290 和 300 并更新 EMPLOYEES 表中的一行。保存数据更改。

```
DELETE FROM departments
WHERE department_id IN (290, 300);

UPDATE employees
SET department_id = 80
WHERE employee_id = 206;

COMMIT;
```

## 执行 ROLLBACK 操作之后的数据状态

使用 ROLLBACK 语句放弃所有待定更改之后会有如下结果：

- 数据更改已取消。
- 数据已还原到以前的状态。
- 受影响行上的锁已被释放。

```
DELETE FROM copy_emp;  
ROLLBACK;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 执行 ROLLBACK 操作之后的数据状态

使用 ROLLBACK 语句放弃所有待定更改之后会有如下结果：

- 数据更改已取消。
- 数据已还原到以前的状态。
- 受影响行上的锁已被释放。

## 执行 ROLLBACK 操作之后的数据状态：示例

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE id = 100;
1 row deleted.

SELECT * FROM test WHERE id = 100;
No rows selected.

COMMIT;
Commit complete.
```



版权所有 © 2010, Oracle。保留所有权利。

## 执行 ROLLBACK 操作之后的数据状态：示例

在试图从 TEST 表中删除一条记录时，可能会意外地清空该表。但是，您可以更正该错误，然后重新发出适当的语句，使数据更改变成永久性更改。

## 语句级回退

- 如果在执行期间单个 DML 语句失败，则只会回退该语句。
- Oracle Server 会实施一个隐式保存点。
- 保留其它所有更改。
- 用户应通过执行 COMMIT 或 ROLLBACK 语句显式终止事务处理。



版权所有 © 2010, Oracle。保留所有权利。

### 语句级回退

如果检测到语句执行错误，则可以通过隐式回退来放弃事务处理的一部分。如果执行事务处理期间单个 DML 语句失败，则可使用语句级回退来取消其影响，但不放弃事务处理中前面的 DML 语句所做的更改。用户可以显式地提交或回退这些 DML 语句。

Oracle Server 会在任意 DDL 语句之前和之后发出隐式提交。因此，即使 DDL 语句未成功执行，您也不能回退前面的语句，因为服务器已发出提交。

通过执行 COMMIT 或 ROLLBACK 语句可以显式终止事务处理。

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 读一致性

- 读一致性可以确保用户所看到的数据始终是一致的。
- 一个用户进行的更改不会与另一个用户进行的更改相冲突。
- 读一致性可以确保对于同一数据：
  - 读取者不必等待写入者完成操作即可读取
  - 写入者不必等待读取者完成操作即可写入
  - 写入者必须等待其他写入者完成操作才可写入



版权所有 © 2010, Oracle。保留所有权利。

### 读一致性

数据库用户可通过以下两种方式访问数据库：

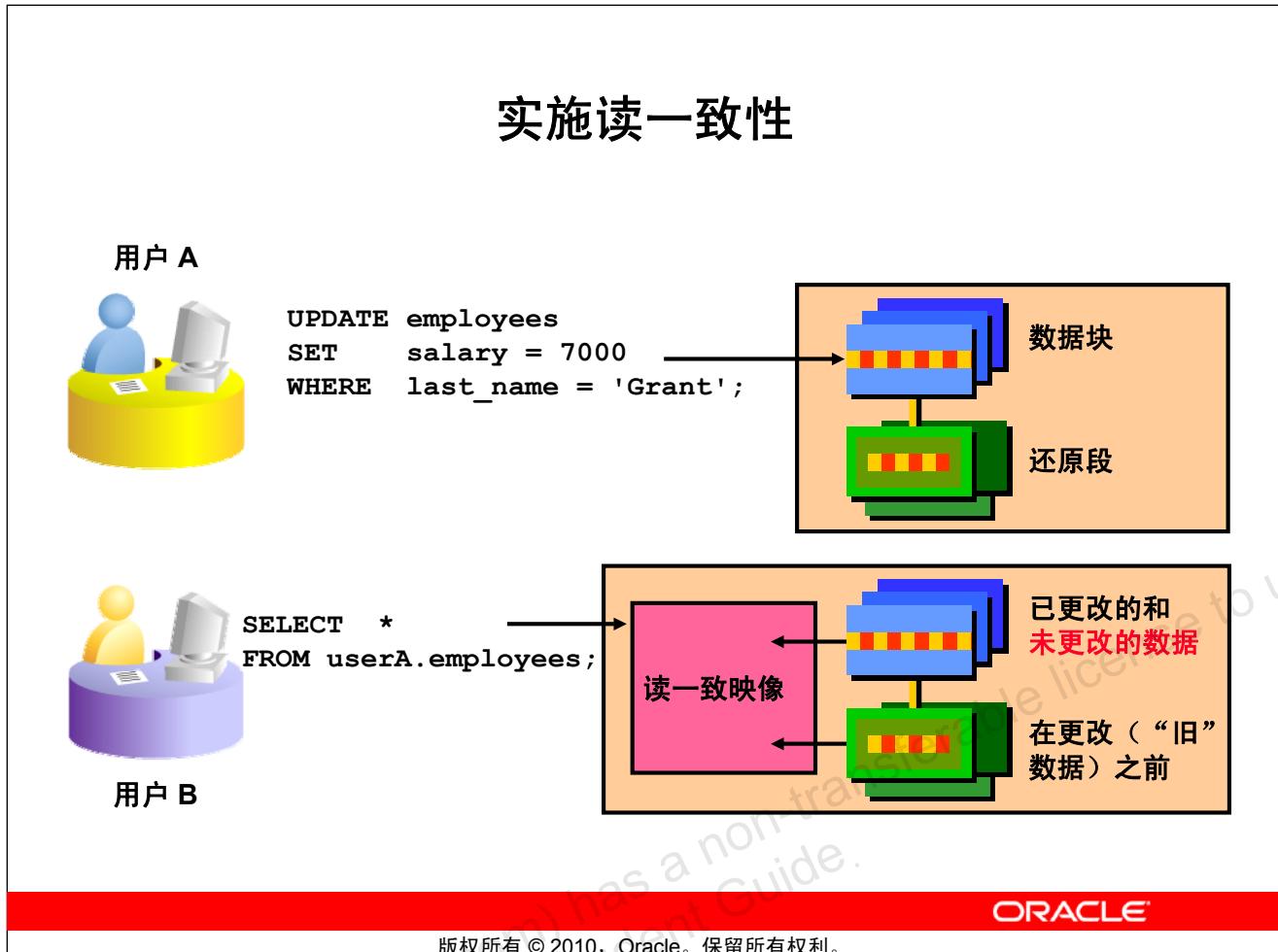
- 读取操作（SELECT 语句）。
- 写入操作（INSERT、UPDATE、DELETE 语句）。

您需要使用读一致性来确保达到如下效果：

- 保证数据库读取者和写入者看到的数据是一致的。
- 读取者看不到正在更改的数据。
- 保证写入者对数据库的更改是在一致方式下完成的。
- 一个写入者进行的更改不会中断另一个写入者正在进行的更改，也不会与之相冲突。

建立读一致性的目的是为了确保每个用户看到的都是在 DML 操作开始之前上次提交的数据。

**注：**同一用户可以登录不同的会话。每个会话均以上述方式维护读一致性，即使这些会话是同一用户的会话也如此。



### 实施读一致性

读一致性是自动实现的。它在还原段中保留一个部分数据库的副本。读一致性映像由表中的已提交数据和还原段中正在更改且尚未提交的旧数据组成。

在对数据库执行插入、更新或删除操作时，Oracle Server 会在数据发生更改之前创建它的副本，并将其写入到还原段中。

除进行更改的人员之外，所有读取者看到的仍是更改开始之前的数据库，他们看到的是还原段的数据“快照”。

在将更改提交到数据库之前，只有正在修改数据的用户可以看到数据库中的更改内容。其他所有人看到的都是还原段中的快照。这样可以保证数据读取者能读取到一致的数据，而不是当前正在进行更改的数据。

提交 DML 语句之后，任何发出 SELECT 语句的用户都可以看到对数据库所做的更改。还原段文件中旧数据占用的空间已被释放出来，可供重新使用。

如果回退事务处理，则会取消这些更改：

- 还原段中原始的旧版本数据已写回到表中。
- 所有用户看到的都是事务处理开始之前的数据库。

## 课程安排

- 在表中添加新行
  - INSERT 语句
- 更改表中的数据
  - UPDATE 语句
- 从表中删除行:
  - DELETE 语句
  - TRUNCATE 语句
- 使用 COMMIT、ROLLBACK 和 SAVEPOINT 执行数据库事务处理控制
- 读一致性
- SELECT 语句中的 FOR UPDATE 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## SELECT 语句中的 FOR UPDATE 子句

- 锁定 EMPLOYEES 表中 job\_id 为 SA REP 的行。

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA REP'
FOR UPDATE
ORDER BY employee_id;
```

- 仅当发出 ROLLBACK 或 COMMIT 语句时才释放锁。
- 如果 SELECT 语句要锁定另一个用户已锁定的某一行，数据库就会一直等待到该行可用为止，然后返回 SELECT 语句的结果。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### SELECT 语句中的 FOR UPDATE 子句

當您针对数据库发出 SELECT 语句来查询某些记录时并不会锁定所选行。通常情况下这样做是有必要的，因为默认情况下任何指定时间锁定的记录数量保持在绝对最低水平：也就是只锁定那些已更改但尚未提交的记录。只有那样，其他用户才能够读取更改前的记录（数据的“前像”）。但是，有时需要在程序中对某组记录进行更改之前将其锁定。使用 Oracle 提供的 SELECT 语句的 FOR UPDATE 子句即可执行该锁定。

发出 SELECT...FOR UPDATE 语句时，关系数据库管理系统 (RDBMS) 会自动获得对由 SELECT 语句标识的所有行的行级互斥锁，因此可暂挂这些记录“仅供您进行更改”。其他人将无法更改这些记录，直至您执行 ROLLBACK 或 COMMIT 语句为止。

您可以将可选的关键字 NOWAIT 附加到 FOR UPDATE 子句中，告知 Oracle Server 在另一用户锁定该表时不要等待。在这种情况下，可以立即将控制权返回给您的程序或 SQL Developer 环境，以便您可以执行其它工作或者仅仅等待一段时间后重试。如果没有 NOWAIT 子句，则会暂时终止您的进程，直至其他用户通过发出 COMMIT 或 ROLLBACK 命名释放锁而使该表可用为止。

## FOR UPDATE 子句：示例

- 可以针对多个表在 SELECT 语句中使用 FOR UPDATE 子句。

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- 已锁定 EMPLOYEES 和 DEPARTMENTS 表中的所有行。
- 使用 FOR UPDATE OF *column\_name* 来限定要更改的列，此时只会锁定特定表中的行。



版权所有 © 2010, Oracle。保留所有权利。

### FOR UPDATE 子句：示例

在幻灯片示例中，该语句锁定了 EMPLOYEES 表中 JOB\_ID 设置为 ST\_CLERK 且 LOCATION\_ID 设置为 1500 的那些行，同时还锁定了 DEPARTMENTS 表中 LOCATION\_ID 的部门设置为 1500 的那些行。

您可使用 FOR UPDATE OF *column\_name* 来限定要更改的列。FOR UPDATE 子句中的 OF 列表并没有限制您只更改选定行的那些列。所有行仍处于锁定状态；如果仅仅在查询中列出了 FOR UPDATE 子句而在 OF 关键字后没有添加任何列，那么，数据库将会锁定 FROM 子句列出的所有表中的所有指定行。

以下语句只锁定了 EMPLOYEES 表中 ST\_CLERK 位于 LOCATION\_ID 1500 的那些行，而没有锁定 DEPARTMENTS 表中的任何行：

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK' AND location_id = 1500
FOR UPDATE OF e.salary
ORDER BY e.employee_id;
```

**FOR UPDATE 子句：示例（续）**

在以下示例中，数据库得知要等待五秒钟该行才可用，因而此时会将控制权返回给用户。

```
SELECT employee_id, salary, commission_pct, job_id  
  FROM employees  
 WHERE job_id = 'SA_REP'  
 FOR UPDATE WAIT 5  
 ORDER BY employee_id;
```

## 小测验

下面的语句可生成相同的结果：

`DELETE FROM copy_emp;`

`TRUNCATE TABLE copy_emp;`

1. 正确
2. 错误

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：2

## 小结

在本课中，您应该已经学会如何使用以下语句：

函数	说明
INSERT	在表中添加新行
UPDATE	修改表中的现有行
DELETE	从表中删除现有行
TRUNCATE	从表中删除所有行
COMMIT	使所有待定更改变成永久性更改
SAVEPOINT	用于回退到保存点标记
ROLLBACK	放弃所有待定的数据更改
SELECT 中的 FOR UPDATE 子句	锁定由 SELECT 查询标识的行

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

在本课中，您应该已经学会如何使用 INSERT、UPDATE、DELETE 和 TRUNCATE 语句在 Oracle DB 中处理数据，以及如何使用 COMMIT、SAVEPOINT 和 ROLLBACK 语句控制数据更改。还应该已学会如何使用 SELECT 语句的 FOR UPDATE 子句来锁定只能由自己进行更改的那些行。

请记住，Oracle Server 可以确保用户在任何时候看到的数据都是一致的。

## 练习 9：概览

本练习包含以下主题：

- 在表中插入行
- 更新和删除表中的行
- 控制事务处理

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 练习 9：概览

在本练习中，您将向 MY\_EMPLOYEE 表中添加行，更新和删除该表中的数据，以及控制您的事务处理。运行脚本来创建 MY\_EMPLOYEE 表。



# 10

## 使用 DDL 语句创建和管理表

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本课后，应能完成下列工作：

- 对主要的数据库对象进行分类
- 查看表结构
- 列举列可以使用的数据类型
- 创建简单的表
- 说明创建表时如何创建约束条件
- 描述方案对象如何工作

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

在本课中，您将学习数据定义语言 (DDL) 语句。还会学到创建、更改和删除简单表的基础知识。本课将介绍 DDL 中可以使用的数据类型以及方案概念。本课还会讨论约束条件，同时还会说明由于在执行 DML 操作期间违反约束条件而生成的异常错误消息。

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



版权所有 © 2010, Oracle。保留所有权利。

# 数据库对象

对象	说明
表	基本存储单元，由行组成
视图	逻辑上代表一个或多个表中数据的子集
序列	用于生成数字值
索引	提高某些查询的性能
同义词	给出对象的替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据库对象

Oracle DB 可以包含多种数据结构。在数据库设计中应对每种结构加以概述，以便可在数据库开发的构建阶段创建数据库结构。

- 表：用于存储数据
- 视图：一个或多个表中数据的子集
- 序列：用于生成数字值
- 索引：提高某些查询的性能
- 同义词：给出对象的替代名称

### Oracle 表结构

- 在任何时候都可以创建表，即使用户正在使用数据库时也是如此。
- 无需指定表的大小。表的大小最终由全部分配给数据库的空间量确定。但是，需要估计一个表将要使用的空间大小，这一点非常重要。
- 可以联机修改表结构。

注：可用的数据库对象很多，但是本课只介绍其中一部分。

## 命名规则

表名和列名必须满足以下条件:

- 以字母开头
- 长度为 1-30 个字符
- 只包含 A-Z、a-z、0-9、\_、\$ 和 #
- 不与同一用户拥有的其它对象重名
- 不是 Oracle Server 的保留字

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 命名规则

应根据命名任意 Oracle DB 对象的标准规则来命名数据库表和列:

- 表名和列名必须以字母开头，长度必须为 1-30 个字符。
- 名称中只能包含字符 A-Z、a-z、0-9、\_（下划线）、\$ 和 #（这两个字符是合法字符，但建议不要使用它们）。
- 不能与同一 Oracle Server 用户拥有的其它对象重名。
- 不能是 Oracle Server 的保留字。
  - 还可以使用加引号的标识符来表示对象名称。加引号的标识符以双引号 ("") 开始和结束。如果使用加双引号的标识符为方案命名，那么，只要引用该对象，就必须使用双引号。加引号的标识符可以是保留字，不过建议不要这样做。

### 命名准则

对于表和其它数据库对象，应使用描述性名称。

**注：**名称不区分大小写，例如，EMPLOYEES 与 eMPloyees 或 eMpLOYEES 被认为是同一名称。但是，加引号的标识符区分大小写。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Schema Object Names and Qualifiers”一节。

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



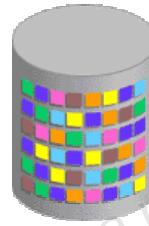
版权所有 © 2010, Oracle。保留所有权利。

## CREATE TABLE 语句

- 您必须具有以下项才能使用此语句：
  - CREATE TABLE 权限
  - 一个存储区

```
CREATE TABLE [schema.]table
(column datatype [DEFAULT expr] [, . . .]);
```

- 可以指定：
  - 表名称
  - 列名、列数据类型和列大小



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### CREATE TABLE 语句

通过执行 SQL CREATE TABLE 语句可以创建用于存储数据的表。此语句是一条 DDL 语句，DDL 语句是 SQL 语句的子集，用于创建、修改或删除 Oracle DB 结构。这些语句会对数据库产生直接的影响，它们还会在数据字典中记录信息。

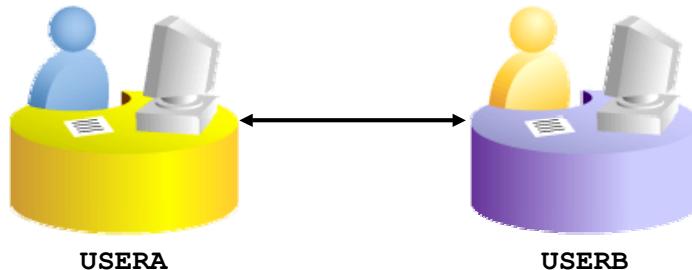
要创建一个表，用户必须具有 CREATE TABLE 权限和一个用于在其中创建对象的存储区。数据库管理员 (DBA) 可以使用数据控制语言 (DCL) 语句为用户授权。

在该语法中：

<i>schema</i>	与所有者的姓名相同
<i>table</i>	是表名称
<i>DEFAULT expr</i>	指定当 INSERT 语句中省略了值时所使用的默认值语句
<i>column</i>	是列名称
<i>datatype</i>	是列的数据类型和长度

## 引用另一个用户的表

- 在用户方案中没有属于其他用户的表。
- 应使用所有者姓名作为那些表的前缀。



```
SELECT *          SELECT *
FROM userB.employees;   FROM userA.employees;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 引用另一个用户的表

方案是由数据或方案对象构成的一组逻辑结构。方案由数据库用户拥有，而且与该用户具有相同的名称。每个用户都拥有一个方案。

方案对象可使用 SQL 来创建和操作；方案对象包括表、视图、同义词、序列、存储过程、索引、集群和数据库链接。

如果某个表不属于该用户，则必须将所有者的姓名作为该表的前缀。例如，假设存在名为 USERA 和 USERB 的两个方案，每个方案都有一个 EMPLOYEES 表，如果 USERA 要访问属于 USERB 的 EMPLOYEES 表，USERA 就必须将 USERB 方案名作为该表名的前缀：

```
SELECT *
FROM userb.employees;
```

如果 USERB 要访问属于 USERA 的 EMPLOYEES 表，USERB 就必须将 USERA 的方案名作为该表名的前缀：

```
SELECT *
FROM usera.employees;
```

## DEFAULT 选项

- 指定插入过程中列的默认值。

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- 文字值、表达式或 SQL 函数都是合法值。
- 其它列的名称或假列是非法值。
- 默认数据类型必须与列的数据类型相匹配。

```
CREATE TABLE hire_dates
  (id          NUMBER(8),
   hire_date DATE DEFAULT SYSDATE);
```

CREATE TABLE succeeded.

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### DEFAULT 选项

在定义表时，可以使用 DEFAULT 选项指定列的默认值。当插入的行中没有某列的相应值时，使用此选项可以防止将空值输入到列中。默认值可以是文字值、表达式或 SQL 函数（例如 SYSDATE 或 USER），但是该值不能是其它列或假列的名称（例如 NEXTVAL 或 CURRVAL）。默认表达式必须与列的数据类型相匹配。

请看如下示例：

```
INSERT INTO hire_dates values(45, NULL);
```

以上语句将插入空值而非默认值。

```
INSERT INTO hire_dates(id) values(35);
```

以上语句将在 HIRE\_DATE 列中插入 SYSDATE。

**注：**在 SQL Developer 中，单击“Run Script（运行脚本）”图标或按 [F5] 可运行 DDL 语句。在“Script Output（脚本输出）”选项卡页上将显示反馈消息。

# 创建表

- 创建表:

```
CREATE TABLE dept
  (deptno      NUMBER(2),
   dname       VARCHAR2(14),
   loc         VARCHAR2(13),
   create_date DATE DEFAULT SYSDATE);
CREATE TABLE succeeded.
```

- 确认表创建:

```
DESCRIBE dept
```

Name	Null	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)
CREATE_DATE		DATE
4 rows selected		

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 创建表

幻灯片示例中创建的 DEPT 表包含以下四列: DEPTNO、DNAME、LOC 和 CREATE\_DATE。CREATE\_DATE 列具有默认值。如果没有为 INSERT 语句提供值, 则会自动插入系统日期。要确认该表是否已创建, 请运行 DESCRIBE 命令。

因为创建表的命令是一条 DDL 语句, 所以在执行该语句后会自动提交。

注: 可以通过查询数据字典来查看您拥有的表列表。例如:

```
select table_name from user_tables
```

使用数据字典视图, 您还可以查找有关其它数据库对象 (例如视图、索引等) 的信息。

有关数据字典的详细信息, 请参阅《Oracle Database 11g: SQL 基础 II》课程。

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据类型

数据类型	说明
VARCHAR2 ( <i>size</i> )	长度可变的字符数据
CHAR ( <i>size</i> )	固定长度的字符数据
NUMBER ( <i>p, s</i> )	长度可变的数字数据
DATE	日期和时间值
LONG	长度可变的字符数据（最多 2 GB）
CLOB	字符数据（最多 4 GB）
RAW 和 LONG RAW	原始二进制数据
BLOB	二进制数据（最多 4 GB）
BFILE	存储在外部文件中的二进制数据（最多 4 GB）
ROWID	64 位基本编号系统，表示行在表中的唯一地址

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据类型

在指定表的列时，需要提供列的数据类型。下面是几种可用的数据类型：

数据类型	说明
VARCHAR2 ( <i>size</i> )	长度可变的字符数据（必须指定最大 <i>size</i> : 最小 <i>size</i> 为 1; 最大 <i>size</i> 为 4,000）
CHAR [( <i>size</i> )]	固定长度的字符数据，长度为 <i>size</i> 字节（默认和最小 <i>size</i> 为 1; 最大 <i>size</i> 为 2,000）
NUMBER [( <i>p, s</i> )]	精度为 <i>p</i> 和小数位数为 <i>s</i> 的数字（精度是十进制数字的总位数，而小数位数是小数点右侧的位数，精度范围在 1 至 38 之间，而该小数位数范围在 -84 至 127 之间）
DATE	公元前 4712 年 1 月 1 日到公元 9999 年 12 月 31 日之间的日期和时间值，精确到最接近的秒
LONG	长度可变的字符数据（最多 2 GB）
CLOB	字符数据（最多 4 GB）

## 数据类型（续）

数据类型	说明
RAW ( <i>size</i> )	长度为 <i>size</i> 的原始二进制数据的（必须指定最大 <i>size</i> : 最大 <i>size</i> 为 2,000）
LONG RAW	长度可变的原始二进制数据（最多 2 GB）
BLOB	二进制数据（最多 4 GB）
BFILE	存储在外部文件中的二进制数据（最多 4 GB）
ROWID	64 位基本编号系统，表示行在表中的唯一地址

### 准则

- 在使用子查询创建表时不复制 LONG 列。
- 不能在 GROUP BY 或 ORDER BY 子句中包括 LONG 列。
- 每个表只能使用一个 LONG 列。
- 不能对 LONG 列定义约束条件。
- 您可以要求使用 CLOB 列，而不是 LONG 列。

## 日期时间数据类型

可以使用以下几种日期时间数据类型：

数据类型	说明
TIMESTAMP	精确到零点几秒的日期
INTERVAL YEAR TO MONTH	存储为时间间隔，以年和月表示
INTERVAL DAY TO SECOND	存储为时间间隔，以天、小时、分钟和秒表示



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 日期时间数据类型

数据类型	说明
TIMESTAMP	将时间存储为精确到零点几秒的日期。该数据类型可以存储 DATE 数据类型的年、月、日、小时、分钟和秒值，可以精确到零点几秒的值。 该数据类型存在好几种变体，如 WITH TIMEZONE、WITH LOCALTIMEZONE。
INTERVAL YEAR TO MONTH	可以将时间存储为以年和月表示的间隔。用于表示两个日期时间值之间的差，其中最重要的部分是年和月。
INTERVAL DAY TO SECOND	可以将时间存储为以天数、时数、分钟数和秒数表示的间隔。用于表示两个日期时间值之间的精确差。

注：可以在 Oracle9i 和更高版本中使用这些日期时间数据类型。将在《Oracle Database 11g: SQL 基础 II》课程的“管理不同时区中的数据”一课中详细讨论有关日期时间数据类型的内容。

此外，有关日期时间数据类型的详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“TIMESTAMP Datatype”、“INTERVAL YEAR TO MONTH Datatype”和“INTERVAL DAY TO SECOND Datatype”三节。

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



版权所有 © 2010, Oracle。保留所有权利。

## 包括约束条件

- 约束条件用于在表级别强制执行各种规则。
- 约束条件用于防止在存在相关性时删除表。
- 下列约束条件类型有效：
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 约束条件

Oracle Server 使用约束条件来防止将无效的数据输入到表中。

可以使用约束条件完成以下任务：

- 在表中插入、更新或删除某一行时，对表中的数据强制执行各种规则。必须满足约束条件，操作才会成功。
- 防止当某个表与其它表存在相关性时删除该表。
- 为 Oracle 工具（例如 Oracle Developer）提供规则。

### 数据完整性约束条件

约束条件	说明
NOT NULL	指定该列不能包含空值
UNIQUE	指定一个列或列组合的值对于表中的所有行必须是唯一的
PRIMARY KEY	唯一地标识表中的每一行
FOREIGN KEY	在该列和所引用表的列之间建立联系后强制实施引用完整性，这样其中一个表的值与另一个表中的值相匹配
CHECK	指定必须为真的条件

## 约束条件准则

- 可以为约束条件命名，也可以由 Oracle Server 使用 SYS\_Cn 格式生成一个名称。
- 可采用以下任何一种方式创建约束条件：
  - 创建表的同时创建约束条件
  - 创建表以后
- 可以在列或表级别定义约束条件。
- 可以在数据字典中查看约束条件。



版权所有 © 2010, Oracle。保留所有权利。

### 约束条件准则

所有约束条件都存储在数据字典中。如果为约束条件指定了一个有意义的名称，则引用时较为容易。约束条件名称必须遵循标准对象命名规则，但是该名称不能与同一用户的另一对象名称相同。如果您没有对约束条件命名，Oracle Server 就会按照 SYS\_Cn 格式生成一个名称，其中 n 是一个整数，这样约束条件名称是唯一的。

既可以在创建表的同时定义约束条件，也可以在创建表之后定义约束条件。您可以在列级别或表级别定义约束条件。从功能上来说，表级别约束条件与列级别约束条件的作用是相同的。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Constraints”一节。

# 定义约束条件

- 语法：

```
CREATE TABLE [schema.]table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint][,...]);
```

- 列级别约束条件语法：

```
column [CONSTRAINT constraint_name] constraint_type,
```

- 表级别约束条件语法：

```
column, ...
[CONSTRAINT constraint_name] constraint_type
(column, ...),
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 定义约束条件

幻灯片给出了在创建表时定义约束条件的语法。您可以在列级别或表级别创建约束条件。定义列时会包括在列级别定义的约束条件。在表定义结束时定义表级别约束条件，必须在一组括号中引用应用了约束条件的列或列组合。这两者主要在语法上有所不同；此外从功能上来说，列级别约束条件和表级别约束条件的作用是相同的。

必须在列级别定义 NOT NULL 约束条件。

必须在表级别定义适用于多个列的约束条件。

在该语法中：

schema	与所有者的姓名相同
table	是表名称
DEFAULT expr	指定当 INSERT 语句中省略了值时所使用的默认值
column	是列名称
datatype	是列的数据类型和长度
column_constraint	是作为列定义一部分的完整性约束条件
table_constraint	是作为表定义一部分的完整性约束条件

## 定义约束条件

- 列级别约束条件示例:

```
CREATE TABLE employees(
    employee_id NUMBER(6)
        CONSTRAINT emp_emp_id_pk PRIMARY KEY,
    first_name VARCHAR2(20),
    ...);
```

1

- 表级别约束条件示例:

```
CREATE TABLE employees(
    employee_id NUMBER(6),
    first_name VARCHAR2(20),
    ...
    job_id      VARCHAR2(10) NOT NULL,
    CONSTRAINT emp_emp_id_pk
        PRIMARY KEY (EMPLOYEE_ID));
```

2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 定义约束条件（续）

通常会在创建表的同时创建约束条件。可以在创建表之后将约束条件添加到表，也可以临时禁用约束条件。

幻灯片两个示例中都对 EMPLOYEES 表的 EMPLOYEE\_ID 列创建了主键约束条件。

- 第一个示例使用列级别语法定义约束条件。
- 第二个示例使用表级别语法定义约束条件。

本课稍后将提供有关主键约束条件的更多详细资料。

## NOT NULL 约束条件

确保列不会有空值：

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

↑  
NOT NULL 约束条件  
(使用主键强制实施  
NOT NULL 约束条件)

↑  
NOT NULL  
约束条件

没有 NOT NULL 约束条件  
(任何行的此列都可以包含空值)

ORACLE

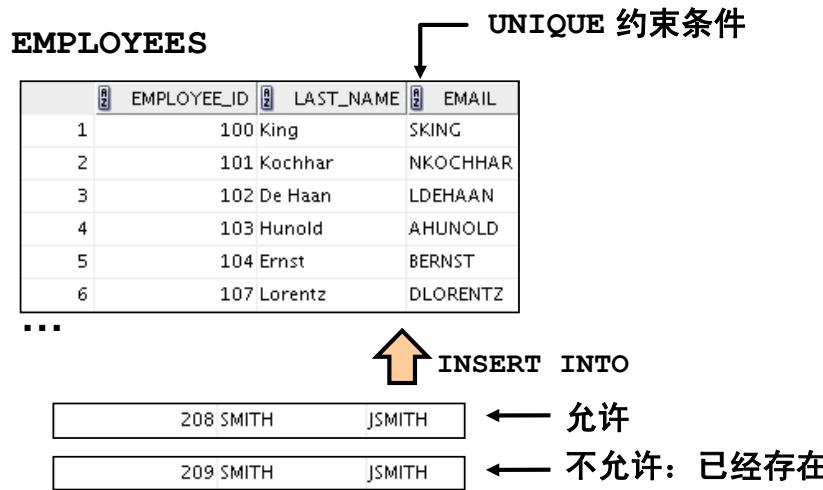
版权所有 © 2010, Oracle。保留所有权利。

### NOT NULL 约束条件

NOT NULL 约束条件可以确保某列不包含空值。默认情况下，没有 NOT NULL 约束条件的列可以包含空值。必须在列级别定义 NOT NULL 约束条件。在 EMPLOYEES 表中，EMPLOYEE\_ID 列继承了 NOT NULL 约束条件，因为该列已定义为主键。否则，在 LAST\_NAME、EMAIL、HIRE\_DATE 和 JOB\_ID 列上强制实施 NOT NULL 约束条件。

注：本课稍后将详细论述主键约束条件。

## UNIQUE 约束条件



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### UNIQUE 约束条件

UNIQUE 关键字完整性约束条件要求一列或一组列（键）中的每个值必须是唯一的，即在指定的列或一组列中，表的任意两行无重复值。UNIQUE 关键字约束条件的定义中包括的列（或一组列）被称为唯一关键字。如果 UNIQUE 约束条件由多个列组成，则该组列被称为组合唯一关键字。

UNIQUE 约束条件允许输入空值，除非您还为同一列定义了 NOT NULL 约束条件。实际上，因为空值被认为不等于任何值，所以任意数量的行都可以在没有 NOT NULL 约束条件的列中包括空值。一个列（或组合 UNIQUE 关键字的所有列）中的空值总是满足 UNIQUE 约束条件。

**注：**由于多个列上 UNIQUE 约束条件的搜索机制所致，在部分空值组合 UNIQUE 关键字约束条件的非空列中不能有相同的值。

## UNIQUE 约束条件

可以在表级别或列级别定义：

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    CONSTRAINT emp_email_uk UNIQUE(email));
```

ORACLE

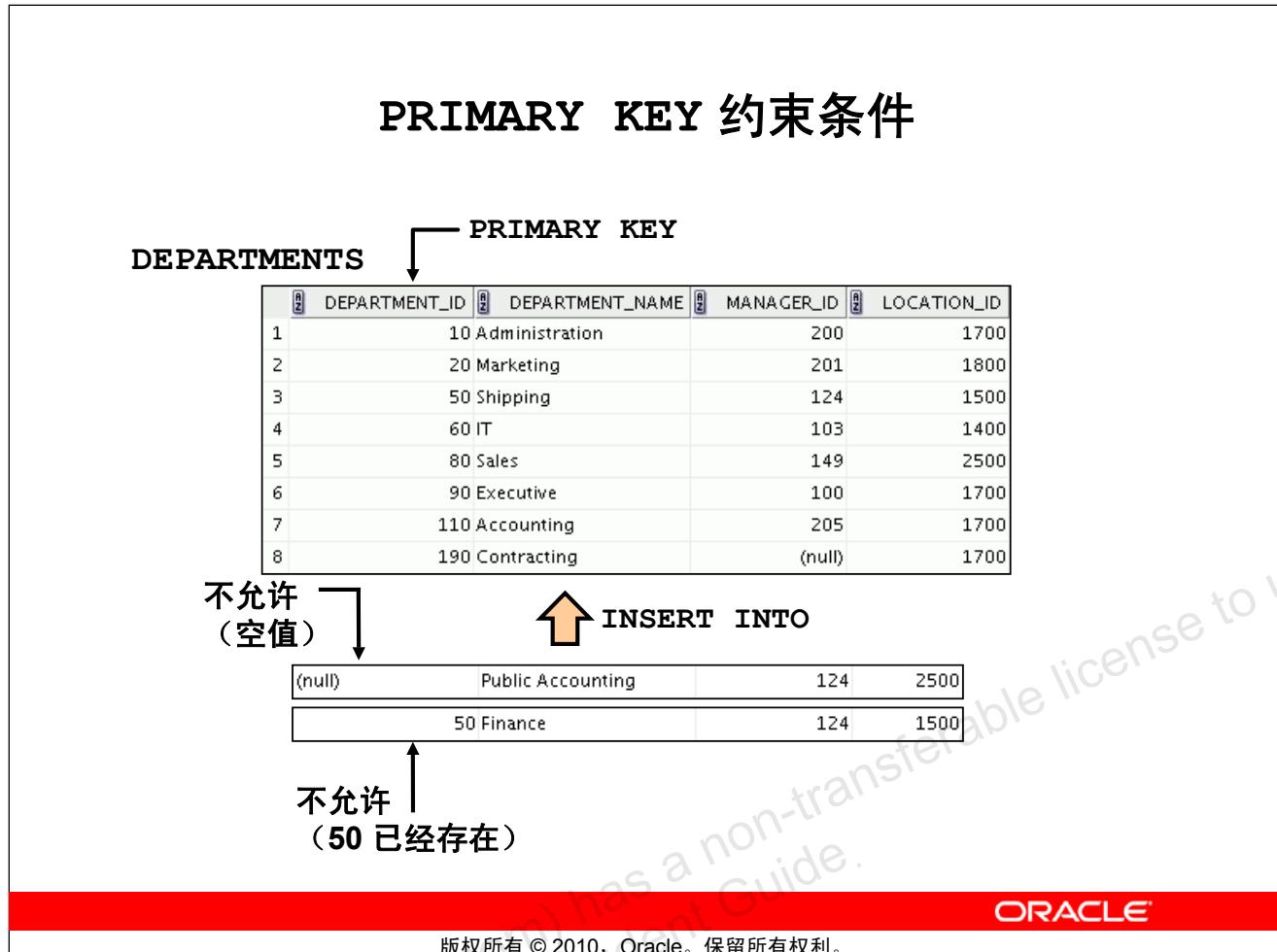
版权所有 © 2010, Oracle。保留所有权利。

### UNIQUE 约束条件（续）

可以在列级别或表级别定义 UNIQUE 约束条件。如果要创建一个组合唯一关键字，则可在表级别定义该约束条件。如果不能使用单个属性来唯一地标识某一行，则需定义组合关键字。在这种情况下，可以创建由两个或两个以上列组成的唯一关键字，其组合值总是唯一的，可用于标识行。

幻灯片示例中将 UNIQUE 约束条件应用于 EMPLOYEES 表的 EMAIL 列。该约束条件的名称为 EMP\_EMAIL\_UK。

注：Oracle Server 通过对一个或多个唯一关键字列隐式创建一个唯一索引来强制实现 UNIQUE 约束条件。

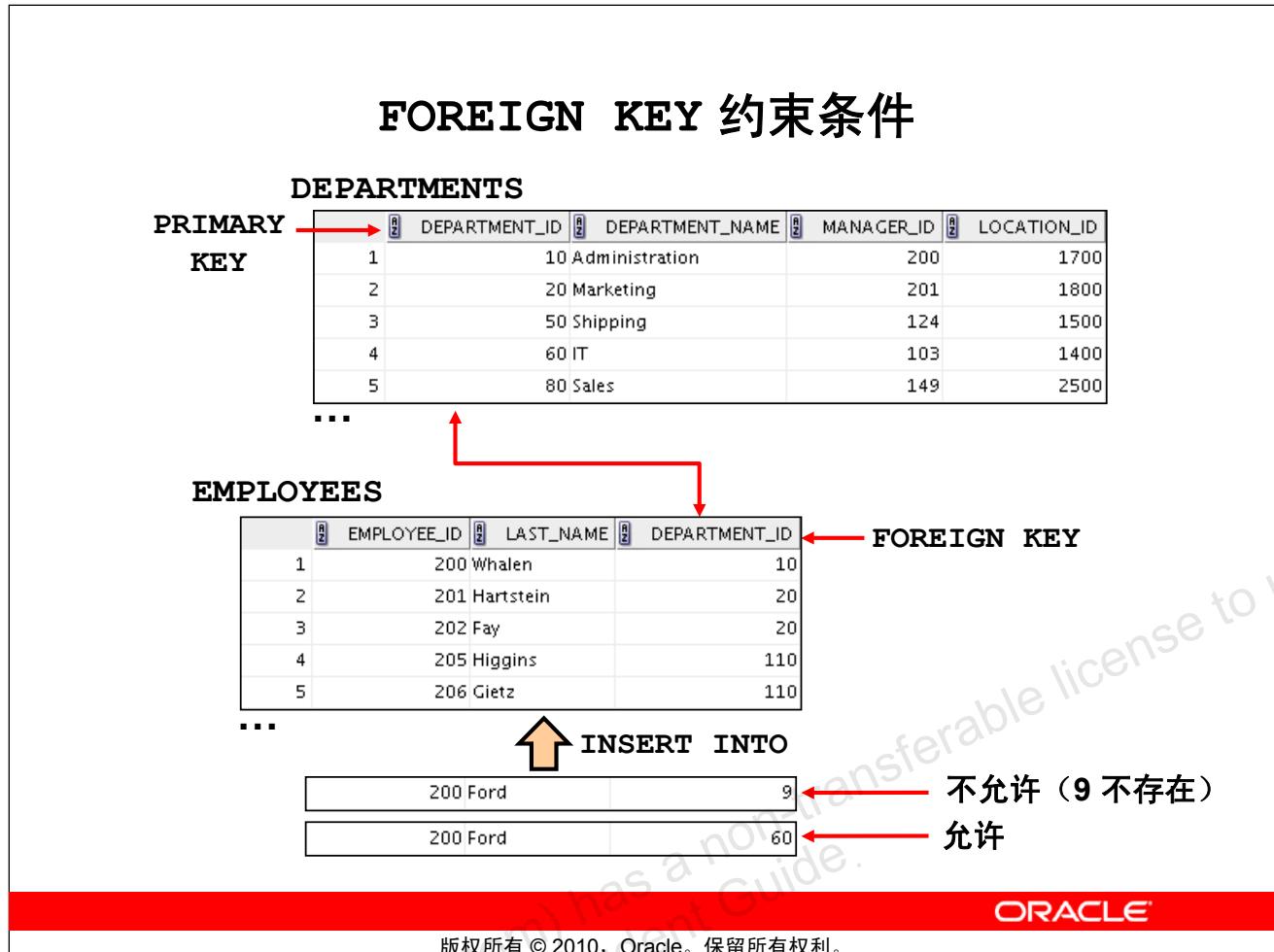


### PRIMARY KEY 约束条件

PRIMARY KEY 约束条件用于创建表的主键。只能为每一个表创建一个主键。

PRIMARY KEY 约束条件是唯一标识表中每一行的一个列或一组列。此约束条件可以强制一个列或列组合是唯一的，还可以确保作为主键一部分的列不包含空值。

**注：**因为唯一性是主键约束条件定义的一部分，所以 Oracle Server 通过对一个或多个主键列隐式创建一个唯一索引来强制实现唯一性。



### FOREIGN KEY 约束条件

FOREIGN KEY (或引用完整性) 约束条件指定一个列或列组合作为外键，并建立与同一表或不同表中主键或唯一关键字的关系。

在幻灯片示例中，DEPARTMENT\_ID 已被定义为 EMPLOYEES 表（相关表或子表）中的外键，它引用 DEPARTMENTS 表（被引用表或父表）的 DEPARTMENT\_ID 列。

#### 准则

- 外键值必须与父表中的现有值相匹配，或为 NULL。
- 外键取决于数据值，外键是纯逻辑指针，而不是物理指针。

## FOREIGN KEY 约束条件

可以在表级别或列级别定义：

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
    ...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
        REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));

```

版权所有 © 2010, Oracle。保留所有权利。

### FOREIGN KEY 约束条件（续）

可以在列级别或表级别定义 FOREIGN KEY 约束条件。必须使用表级别定义来创建组合外键。

幻灯片示例中使用表级别语法，对 EMPLOYEES 表的 DEPARTMENT\_ID 列定义一个 FOREIGN KEY 约束条件。该约束条件的名称为 EMP\_DEPT\_FK。

如果约束条件只是针对单个列，则也可以在列级别定义外键。语法上的不同之处在于没有出现关键字 FOREIGN KEY。例如：

```
CREATE TABLE employees
(
...
    department_id NUMBER(4) CONSTRAINT emp_deptid_fk
        REFERENCES departments(department_id),
    ...
)
```

## FOREIGN KEY 约束条件：关键字

- FOREIGN KEY：在表约束条件级别定义子表中的列
- REFERENCES：标识父表中的表和列
- ON DELETE CASCADE：删除父表中的行时还删除子表中的相关行
- ON DELETE SET NULL：将相关外键值转换为空值



版权所有 © 2010, Oracle。保留所有权利。

### FOREIGN KEY 约束条件：关键字

外键是在子表中定义的，而包含被引用列的表是父表。外键是使用以下关键字的组合定义的：

- FOREIGN KEY 用于在表约束条件级别定义子表中的列。
- REFERENCES 用于标识父表中的表和列。
- ON DELETE CASCADE 指出在删除父表中的行时，还删除子表中的相关行。
- ON DELETE SET NULL 指出在删除父表中的行时，将外键值设为空值。

默认行为被称为限制规则，该规则可禁止更新或禁止删除被引用的数据。

在没有 ON DELETE CASCADE 或 ON DELETE SET NULL 选项时，如果在子表中引用父表中的一行，则不能删除该行。

## CHECK 约束条件

- 定义每行都必须满足的一个条件
- 以下表达式是不允许的：
  - 引用 CURRVAL、NEXTVAL、LEVEL 和 ROWNUM 假列的表达式
  - 调用 SYSDATE、UID、USER 和 USERENV 函数的表达式
  - 引用其它行中的其它值的查询

```
..., salary NUMBER(2)
CONSTRAINT emp_salary_min
    CHECK (salary > 0), ...
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### CHECK 约束条件

CHECK 约束条件用于定义每行都必须满足的一个条件。该条件可以使用与查询条件相同的结构，但是以下项除外：

- 引用 CURRVAL、NEXTVAL、LEVEL 和 ROWNUM 假列的表达式
- 调用 SYSDATE、UID、USER 和 USERENV 函数的表达式
- 引用其它行中的其它值的查询

一个列可以有多个 CHECK 约束条件，这些约束条件将在其定义中引用该列。可以按需要，对一个列定义任意数量的 CHECK 约束条件。

可以在列级别或表级别定义 CHECK 约束条件。

```
CREATE TABLE employees
(
    ...
    salary NUMBER(8,2) CONSTRAINT emp_salary_min
        CHECK (salary > 0),
    ...
)
```

## CREATE TABLE: 示例

```
CREATE TABLE employees
  ( employee_id      NUMBER(6)
    CONSTRAINT emp_employee_id PRIMARY KEY
  , first_name        VARCHAR2(20)
  , last_name         VARCHAR2(25)
    CONSTRAINT emp_last_name_nn NOT NULL
  , email             VARCHAR2(25)
    CONSTRAINT emp_email_nn    NOT NULL
    CONSTRAINT emp_email_uk    UNIQUE
  , phone_number      VARCHAR2(20)
  , hire_date         DATE
    CONSTRAINT emp_hire_date_nn NOT NULL
  , job_id            VARCHAR2(10)
    CONSTRAINT emp_job_nn      NOT NULL
  , salary             NUMBER(8,2)
    CONSTRAINT emp_salary_ck    CHECK (salary>0)
  , commission_pct    NUMBER(2,2)
  , manager_id        NUMBER(6)
    CONSTRAINT emp_manager_fk REFERENCES
      employees (employee_id)
  , department_id     NUMBER(4)
    CONSTRAINT emp_dept_fk REFERENCES
      departments (department_id));
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## CREATE TABLE: 示例

幻灯片示例中显示用于创建 HR 方案中 EMPLOYEES 表的语句。

## 违反约束条件

```
UPDATE employees  
SET department_id = 55  
WHERE department_id = 110;
```

```
Error starting at line 1 in command:  
UPDATE employees  
SET department_id = 55  
WHERE department_id = 110  
Error report:  
SQL Error: ORA-02291: integrity constraint (ORA1.EMP_DEPT_FK) violated - parent key not found  
02291. 00000 - "integrity constraint (%s.%s) violated - parent key not found"  
*Cause: A foreign key value has no matching primary key value.
```

部门 55 不存在。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 违反约束条件

当对列设置了约束条件后，如果试图违反约束条件规则，则会返回一条错误。例如，如果尝试更新一条记录，但该记录中的值受完整性约束条件所约束，则会返回一条错误。

在幻灯片示例中，因为父表 DEPARTMENTS 中不存在部门 55，所以您会收到违例 ORA-02291 “parent key not found（未找到父关键字）”。

## 违反约束条件

如果某行中包含用作其它表中的外键的主键，则不能删除该行。

```
DELETE FROM departments
WHERE department_id = 60;
```

```
Error starting at line 1 in command:
DELETE FROM departments
WHERE department_id = 60
Error report:
SQL Error: ORA-02292: integrity constraint (ORA1.JHIST_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
          dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

版权所有 © 2010, Oracle。保留所有权利。

### 违反约束条件（续）

如果试图删除一条记录，但该记录中的值受完整性约束条件所约束，则会返回一条错误。幻灯片示例中试图从 DEPARTMENTS 表中删除部门 60，但此操作导致一个错误，因为该部门编号已用作 EMPLOYEES 表中的外键。如果您试图删除具有子记录的父记录，则会收到违例 ORA-02292 “child record found（存在子记录）”。

由于部门 70 中没有任何雇员，因此下面的语句有效：

```
DELETE FROM departments
WHERE department_id = 70;
```

**1 rows deleted**

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



版权所有 © 2010, Oracle。保留所有权利。

## 使用子查询创建表

- 通过组合 CREATE TABLE 语句和 AS subquery 选项可以创建表并插入行。

```
CREATE TABLE table
    [(column, column...)]
AS subquery;
```

- 使指定列的数量与子查询列的数量相匹配。
- 定义具有列名和默认值的列。



版权所有 © 2010, Oracle。保留所有权利。

### 使用子查询创建表

创建表的另一种方法是应用 AS subquery 子句，该方法既可以创建表，又可以将子查询返回的行插入表中。

在该语法中：

*table* 是表名称

*column* 是列的名称、默认值和完整性约束条件

*subquery* 是一条 SELECT 语句，用于定义要插入到新表中的一组行

#### 准则

- 使用指定的列名创建一个表，然后将 SELECT 语句检索到的那些行插入到该表中。
- 列定义只能包含列名和默认值。
- 如果已经给出列的规格，则列数必须等于子查询 SELECT 列表中的列数。
- 如果没有给出列的规格，则表的列名与子查询中的列名相同。
- 列数据类型定义和 NOT NULL 约束条件会传递到新表中，请注意，只会继承显式 NOT NULL 约束条件。PRIMARY KEY 列不会将 NOT NULL 特性传递给新列。任何其它约束条件规则也不会传递到新表中。但是，您可以在列定义中添加约束条件。

## 使用子查询创建表

```
CREATE TABLE dept80
AS
SELECT employee_id, last_name,
       salary*12 ANNSAL,
       hire_date
FROM employees
WHERE department_id = 80;
```

CREATE TABLE succeeded.

DESCRIBE dept80

Name	Null	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用子查询创建表（续）

幻灯片示例中创建了一个名为 DEPT80 的表，该表包含在部门 80 中工作的所有雇员的详细资料。请注意，DEPT80 表中的数据来自 EMPLOYEES 表。

可以使用 DESCRIBE 命令验证数据库表的存在，并检查列定义。

但是，在选择表达式时请务必提供列别名。表达式 SALARY\*12 被赋予了别名 ANNSAL。如果没有别名，系统就会生成以下错误：

```
Error starting at line 1 in command:
CREATE TABLE dept80
AS      SELECT employee_id, last_name,
           salary*12 ,
           hire_date FROM employees WHERE department_id = 80
Error at Command Line:3 Column:18
Error report:
SQL Error: ORA-00998: must name this expression with a column alias
00998. 00000 - "must name this expression with a column alias"
*Cause:
*Action:
```

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



版权所有 © 2010, Oracle。保留所有权利。

## ALTER TABLE 语句

使用 ALTER TABLE 语句可：

- 添加新列
- 修改现有列定义
- 定义新列的默认值
- 删除列
- 重命名列
- 将表的状态更改为“只读”

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### ALTER TABLE 语句

在创建表之后，您可能会因以下原因而需要更改表结构：

- 省略了某列。
- 列定义或列名需要更改。
- 需要删除列。
- 要将表设置为“只读”模式。

可以使用 ALTER TABLE 语句完成此任务。

## 只读表

可以使用 ALTER TABLE 语法执行下列操作：

- 将表置于只读模式，从而阻止在维护表过程中进行 DDL 或 DML 更改
- 将表重新置于读/写模式

```
ALTER TABLE employees READ ONLY;  
-- 执行表维护，然后再  
-- 将表改回读/写模式  
  
ALTER TABLE employees READ WRITE;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 只读表

在 Oracle Database 11g 中，可以通过指定 READ ONLY 将表设置为“只读”模式。当表处于 READ-ONLY 模式时，用户不能发出可对表造成影响的任何 DML 语句或任何 SELECT ... FOR UPDATE 语句。可以发出 DDL 语句，但前提条件是不对表中的任何数据进行修改。当表处于 READ ONLY 模式时，可以对与表关联的索引执行操作。

指定 READ/WRITE，可令处于“只读”模式的表返回“读/写”模式。

**注：**可以删除处于 READ ONLY 模式的表。DROP 命令只能在数据字典中执行，因此无需访问表内容。在表空间返回“读/写”状态之前，系统不会回收表使用的空间，此时可对块段头进行必要的更改。

有关 ALTER TABLE 语句的信息，请参阅《Oracle Database 10g: SQL 基础 II》课程。

## 课程安排

- 数据库对象
  - 命名规则
- CREATE TABLE 语句:
  - 访问另一个用户的表
  - DEFAULT 选项
- 数据类型
- 约束条件概览: NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 约束条件
- 使用子查询创建表
- ALTER TABLE
  - 只读表
- DROP TABLE 语句



版权所有 © 2010, Oracle。保留所有权利。

## 删除表

- 将表移至回收站
- 如果指定了 PURGE 子句，则可将表及其所有数据全部删除
- 使从属对象失效并删除表的对象权限

```
DROP TABLE dept80;
```

```
DROP TABLE dept80 succeeded.
```



版权所有 © 2010, Oracle。保留所有权利。

### 删除表

使用 DROP TABLE 语句可以将表移至回收站或从数据库中全部删除该表及其所有数据。如果不指定 PURGE 子句，DROP TABLE 语句就不会将使用的空间重新释放到表空间中供其它对象使用，而且该空间会继续视为用户的空间限额。删除表会使从属对象失效并删除表的对象权限。

删除表后，在数据库中会失去表的所有数据以及与表关联的所有索引。

#### 语法

```
DROP TABLE table [PURGE]
```

在该语法中，table 是表的名称。

#### 准则

- 删除表中的所有数据。
- 保留视图和同义词，但不再有效。
- 提交所有待定的事务处理。
- 只有表的创建者或具有 DROP ANY TABLE 权限的用户才能删除表。

**注：** 使用 FLASHBACK TABLE 语句可从回收站中还原已删除的表。这将在《Oracle Database 11g: SQL 基础 II》课程中进行详细讨论。

## 小测验

可以使用约束条件完成以下任务：

1. 插入、更新或删除某一行时，对表中的数据强制执行各种规则。
2. 阻止删除表。
3. 阻止创建表。
4. 阻止在表中创建数据。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、2、4

## 小结

在本课中，您应该已经学会如何使用 CREATE TABLE 语句创建表和包括约束条件：

- 对主要的数据库对象进行分类
- 查看表结构
- 列举列可以使用的数据类型
- 创建简单的表
- 说明创建表时如何创建约束条件
- 描述方案对象如何工作

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

在本课中，您应该已经学会如何使用以下语句：

#### **CREATE TABLE**

- 使用 CREATE TABLE 语句创建表和包括约束条件。
- 使用子查询根据其它表创建表。

#### **DROP TABLE**

- 删除行和表结构。
- 一旦执行，就无法回退该语句。

## 练习 10：概览

本练习包含以下主题：

- 创建新表
- 使用 CREATE TABLE AS 语法创建新表
- 验证表的存在
- 将表的状态设置为“只读”
- 删除表

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 练习 10：概览

使用 CREATE TABLE 语句创建新表。确认新表已添加到数据库。您还会学习如何将表的状态设置为 READ ONLY，然后再还原为 READ/WRITE。

**注：**对于所有的 DDL 和 DML 语句，单击“Run Script（运行脚本）”图标（或按 [F5]）即可在 SQL Developer 中执行查询。您可以在“Script Output（脚本输出）”选项卡页上查看反馈消息。对于 SELECT 查询，可以继续单击“Execute Statement（执行语句）”图标或按 [F9]，在“Results（结果）”选项卡页上获得格式化输出。



# 11

## 创建其它方案对象

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

学完本课后，应能完成下列工作：

- 创建简单视图和复杂视图
- 从视图中检索数据
- 创建、维护和使用序列
- 创建和维护索引
- 创建私用和公用同义词

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

在本课中，您将学习视图、序列、同义词和索引对象。还将学习创建和使用视图、序列以及索引的基础知识。

## 课程安排

- 视图概览:
  - 创建、修改和检索视图中的数据
  - 对视图执行数据操纵语言 (DML) 操作
  - 删除视图
- 序列概览:
  - 创建、使用和修改序列
  - 高速缓存序列值
  - NEXTVAL 和 CURRVAL 伪列
- 索引概览
  - 创建、删除索引
- 同义词概览
  - 创建、删除同义词

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

# 数据库对象

对象	说明
表	基本存储单元，由行组成
视图	逻辑上代表一个或多个表中数据的子集
序列	用于生成数字值
索引	提高数据检索查询的性能
同义词	给出对象的替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 数据库对象

数据库中除表之外还存在几种其它对象。

使用视图，可以显示和隐藏表中的数据。

许多应用程序都要求将唯一编号用作主键值。可以将代码编入到应用程序中来处理这种要求，也可以使用序列生成唯一编号。

如果要提高数据检索查询的性能，则应考虑创建一个索引。还可以使用索引对一个列或列集合强制实现唯一性。

可以使用同义词为对象提供替代名称。

# 什么是视图

**EMPLOYEES 表**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000
104	Bruce	Ernst	BERNSTEIN	590.423.4568	23-JAN-90	IT_PROG	6000
105	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	12000
106	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	8300

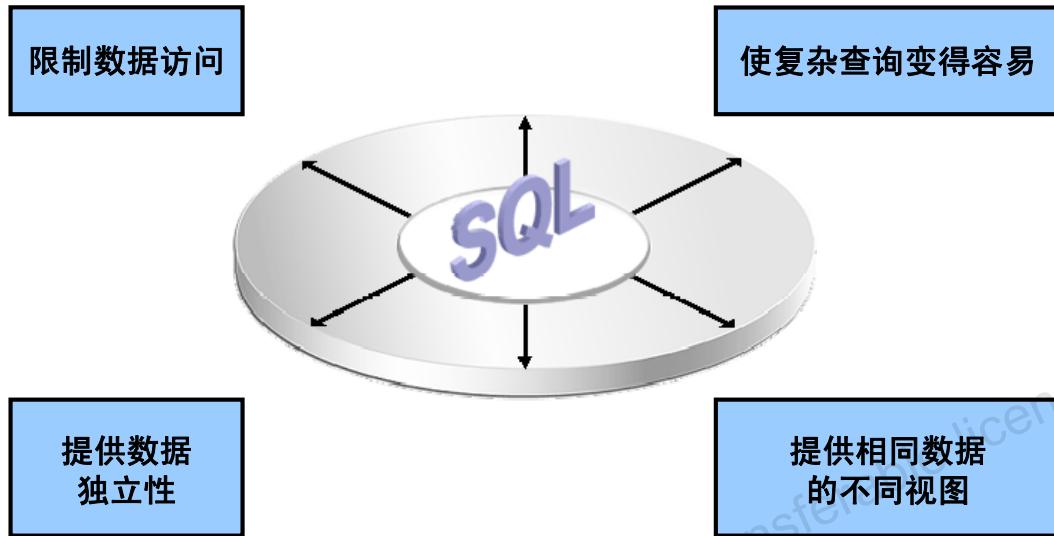
EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000
205	Shelley	Higgins	6666
206	William	Gietz	515.123.8181

版权所有 © 2010, Oracle。保留所有权利。

## 什么是视图

通过创建表的视图可以显示数据的逻辑子集或组合。视图是一种基于表或其它视图的逻辑表。视图没有自己的数据，但它如同一个窗口，通过它可以查看或更改表中的数据。视图所基于的表被称为基表。视图以 SELECT 语句的形式存储在数据字典中。

## 视图的优点



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 视图的优点

- 由于视图可以显示表中的选定列，因而可以限制对数据的访问。
- 视图可用来通过进行简单查询来检索复杂查询的结果。例如，用户在不了解如何编写联接语句时，使用视图就可以查询多个表中的信息。
- 视图为特定用户和特定应用程序提供了数据独立性。一个视图可用来检索多个表中的数据。
- 通过视图，用户组可根据各自的特定标准访问数据。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“CREATE VIEW”一节。

## 简单视图和复杂视图

特点	简单视图	复杂视图
表的数量	一个	一个或多个
是否包含函数	否	是
是否包含数据组	否	是
是否通过视图执行 DML 操作	是	不一定

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 简单视图和复杂视图

视图有两种分类：简单视图和复杂视图。这两类视图的基本差别与 DML（INSERT、UPDATE 和 DELETE）操作有关。

- 简单视图有如下特点：
  - 只从一个表中获得数据
  - 不包含函数或数据组
  - 可以通过视图执行 DML 操作
- 复杂视图有如下特点：
  - 从多个表中获得数据
  - 包含函数或数据组
  - 不一定允许通过视图执行 DML 操作

## 创建视图

- 在 CREATE VIEW 语句中嵌入一个子查询：

```
CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW view
[ (alias[, alias]... ) ]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- 子查询可以包含复杂的 SELECT 语法。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 创建视图

可以通过在 CREATE VIEW 语句中嵌入子查询来创建视图。

在该语法中：

OR REPLACE	如果该视图已存在则重新创建
FORCE	不管基表是否存在都创建视图
NOFORCE	仅当基表存在时才创建视图（默认设置）
<i>view</i>	是视图的名称
<i>alias</i>	指定由视图查询选定的表达式的名称（别名的数量必须与视图选择的表达式数量匹配）
<i>subquery</i>	是一个完整的 SELECT 语句（可以在 SELECT 列表中使用列的别名）
WITH CHECK OPTION	指定只插入或只更新视图中可以访问的那些行
<i>constraint</i>	是为 CHECK OPTION 约束条件指定的名称
WITH READ ONLY	确保不对此视图执行 DML 操作

注：在 SQL Developer 中，单击“Run Script（运行脚本）”图标或按[F5]可运行数据定义语言 (DDL) 语句。在“Script Output（脚本输出）”选项卡页上将显示反馈消息。

## 创建视图

- 创建视图 EMPVU80，其中包括部门 80 中雇员的详细资料：

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
CREATE VIEW succeeded.
```

- 使用 SQL\*Plus DESCRIBE 命令描述视图的结构：

```
DESCRIBE empvu80
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 创建视图（续）

幻灯片示例中创建了一个视图，该视图包含部门 80 中每位雇员的雇员编号、姓氏和薪金。可以使用 DESCRIBE 命令显示该视图的结构。

Name	Null	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
SALARY		NUMBER(8,2)

### 准则

- 定义视图的子查询可以包含复杂的 SELECT 语法，其中包括联接、分组和子查询。
- 如果您没有为使用 WITH CHECK OPTION 创建的视图指定约束条件名称，系统则会以 SYS\_Cn 格式指定默认名称。
- 可以使用 OR REPLACE 选项更改视图的定义，而不必在删除该视图后再重新创建，也不必对其重新授予权限。

## 创建视图

- 在子查询中使用列别名创建视图：

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
           salary*12 ANN_SALARY
      FROM employees
     WHERE department_id = 50;
CREATE VIEW succeeded.
```

- 按给定的别名从此视图中选择列。

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 创建视图（续）

可以通过将列别名包含在子查询中来控制列名。

幻灯片示例中创建了一个视图，其中包含部门 50 中每位雇员的以下信息：别名为 ID\_NUMBER 的雇员编号 (EMPLOYEE\_ID)、别名为 NAME 的姓名 (LAST\_NAME) 以及别名为 ANN\_SALARY 的年薪 (SALARY)。

您也可以在 CREATE 语句之后和 SELECT 子查询之前使用别名。列出的别名数量必须与在子查询中选定的表达式数量相匹配。

```
CREATE OR REPLACE VIEW salvu50 (ID_NUMBER, NAME, ANN_SALARY)
AS SELECT employee_id, last_name, salary*12
      FROM employees
     WHERE department_id = 50;
CREATE VIEW succeeded.
```

## 从视图中检索数据

```
SELECT *
FROM salvu50;
```

	ID_NUMBER	NAME	ANN_SALARY
1	124 Mourgos		69600
2	141 Rajs		42000
3	142 Davies		37200
4	143 Matos		31200
5	144 Vargas		30000



版权所有 © 2010, Oracle。保留所有权利。

### 从视图中检索数据

您可以从视图中检索数据，就像从任何表中检索数据一样。可以显示整个视图的内容，也可以只显示特定的行或列。

## 修改视图

- 可以使用 CREATE OR REPLACE VIEW 子句修改视图 EMPVU80。为每个列名添加一个别名：

```
CREATE OR REPLACE VIEW empvu80
(id_number, name, sal, department_id)
AS SELECT employee_id, first_name || ' '
    || last_name, salary, department_id
  FROM employees
 WHERE department_id = 80;
CREATE OR REPLACE VIEW succeeded.
```

- CREATE OR REPLACE VIEW 子句中列出的列别名与子查询中的列具有相同的顺序。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 修改视图

使用 OR REPLACE 选项，可以创建一个视图，甚至可以创建一个与已存在的视图同名的视图，以便替换旧版本的视图。这意味着可以更改视图，而不必经过删除、重新创建对象和重新授予对象权限的过程。

**注：**在 CREATE OR REPLACE VIEW 子句中指定列别名时，请注意别名的列出顺序应与子查询中列的顺序相同。

## 创建复杂视图

创建包含组函数的复杂视图以显示两个表中的值：

```
CREATE OR REPLACE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT    d.department_name, MIN(e.salary),
              MAX(e.salary), AVG(e.salary)
      FROM employees e JOIN departments d
     ON      (e.department_id = d.department_id)
    GROUP BY d.department_name;
CREATE OR REPLACE VIEW succeeded.
```

版权所有 © 2010, Oracle。保留所有权利。

### 创建复杂视图

幻灯片示例中创建了一个复杂视图，其中包含按部门列出的部门名称、最低薪金、最高薪金和平均薪金。请注意，已为该视图指定了替代名称。如果视图中有任何列来自于函数或表达式，则需要使用替代名称。

可以使用 DESCRIBE 命令查看视图的结构。通过发出 SELECT 语句可显示视图的内容。

```
SELECT *
  FROM dept_sum_vu;
```

	NAME	MIN SAL	MAX SAL	Avg SAL
1	Administration	4400	4400	4400
2	Accounting	8300	12000	10150
3	IT	4200	9000	6400
4	Executive	17000	24000	19333.333333333...
5	Shipping	2500	5800	3500
6	Sales	8600	11000	10033.333333333...
7	Marketing	6000	13000	9500

## 对视图执行 DML 操作的规则

- 通常可以对简单视图执行 DML 操作。
- 如果视图包含以下内容，则不能删除行：
  - 组函数
  - GROUP BY 子句
  - DISTINCT 关键字
  - 伪列 ROWNUM 关键字



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 对视图执行 DML 操作的规则

- 可以对整个视图中的数据执行 DML 操作，但这些操作必须符合特定的规则。
- 如果视图不包含以下任何内容，则可以从视图中删除行：
  - 组函数
  - GROUP BY 子句
  - DISTINCT 关键字
  - 伪列 ROWNUM 关键字

## 对视图执行 DML 操作的规则

如果视图包含以下内容，则不能修改视图中的数据：

- 组函数
- GROUP BY 子句
- DISTINCT 关键字
- 伪列 ROWNUM 关键字
- 由表达式定义的列

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 对视图执行 DML 操作的规则（续）

可以修改视图中的数据，除非该视图包含前一幻灯片中提到的任何条件或由表达式定义的列（例如，SALARY \* 12）。

## 对视图执行 DML 操作的规则

如果视图包括以下内容，则不能向视图添加数据：

- 组函数
- GROUP BY 子句
- DISTINCT 关键字
- 伪列 ROWNUM 关键字
- 由表达式定义的列
- 基表中未被视图选中的 NOT NULL 列

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 对视图执行 DML 操作的规则（续）

您可以向视图添加数据，除非视图包含此幻灯片中列出的任何项。如果视图包含的 NOT NULL 列在基表中没有指定默认值，则不能向视图添加数据。在视图中必须显示所有需要的值。请记住，您要通过视图将值直接添加到基表中。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“CREATE VIEW”一节。

## 使用 WITH CHECK OPTION 子句

- 使用 WITH CHECK OPTION 子句可确保对视图执行的 DML 操作只在视图范围内起作用：

```
CREATE OR REPLACE VIEW empvu20
AS SELECT      *
   FROM employees
  WHERE department_id = 20
    WITH CHECK OPTION CONSTRAINT empvu20_ck ;
CREATE OR REPLACE VIEW succeeded.
```

- 如果尝试使用 INSERT 语句插入 department\_id 不为 20 的一行，或者使用 UPDATE 语句更新视图中任何行的部门编号，则操作会失败，因为这违反 WITH CHECK OPTION 约束条件。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用 WITH CHECK OPTION 子句

可以在视图中执行引用完整性检查。还可以在数据库级别强制实现约束条件。视图可以用来保护数据完整性，但此用途很有限。

WITH CHECK OPTION 子句指定通过视图执行的 INSERT 和 UPDATE 不能创建该视图无法选择的行。因此，能够对要插入或更新的数据强制执行完整性约束条件和数据验证检查。如果尝试对视图未选中的行执行 DML 操作，则会显示一条错误，还会显示约束条件名称（如果已指定）。

```
UPDATE empvu20
SET department_id = 10
WHERE employee_id = 201;
```

原因：

```
Error report:
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
```

**注：**如果部门编号已更改为 10，则该视图将无法看到此雇员，因此不会更新任何行。所以，使用 WITH CHECK OPTION 子句时，该视图只能看到部门 20 中的雇员，并且不允许通过该视图更改这些雇员的部门编号。

## 拒绝 DML 操作

- 通过在视图定义中添加 WITH READ ONLY 选项可以确保不会执行 DML 操作。
- 尝试对视图中的任何行执行 DML 操作都会导致产生 Oracle Server 错误。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 拒绝 DML 操作

通过在创建视图时使用 WITH READ ONLY 选项，可以确保不会对视图执行 DML 操作。  
下一张幻灯片中的示例修改了 EMPVU10 视图，可防止对该视图执行任何 DML 操作。

## 拒绝 DML 操作

```
CREATE OR REPLACE VIEW empvu10
  (employee_number, employee_name, job_title)
AS SELECT      employee_id, last_name, job_id
   FROM        employees
  WHERE      department_id = 10
    WITH READ ONLY ;
CREATE OR REPLACE VIEW succeeded.
```



版权所有 © 2010, Oracle。保留所有权利。

### 拒绝 DML 操作（续）

尝试从具有只读约束条件的视图中删除任何行会导致产生错误：

```
DELETE FROM empvu10
WHERE employee_number = 200;
```

与此类似，尝试使用具有只读约束条件的视图插入行或修改行会导致同样的错误。

```
Error report:
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view
```

## 删除视图

因为视图是基于数据库中的基表建立的，所以删除视图不会导致丢失数据。

```
DROP VIEW view;
```

```
DROP VIEW empvu80;
```

```
DROP VIEW empvu80 succeeded.
```



版权所有 © 2010, Oracle。保留所有权利。

### 删除视图

使用 `DROP VIEW` 语句可删除视图。该语句会从数据库中删除视图定义。但是，删除视图不会影响视图的基表。此外，基于已删除视图的视图或其它应用程序会变得无效。只有创建者或具有 `DROP ANY VIEW` 权限的用户才能删除视图。

在该语法中，`view` 是视图的名称。

## 练习 11：第 1 部分概览

本练习包含以下主题：

- 创建简单视图
- 创建复杂视图
- 创建具有检查约束条件的视图
- 尝试修改视图中的数据
- 删除视图

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 练习 11：第 1 部分概览

本课练习的第 1 部分为您提供了有关创建、使用和删除视图的各种练习。学完本课后请完成问题 1-6。

## 课程安排

- 视图概览:
  - 创建、修改和检索视图中的数据
  - 对视图执行 DML 操作
  - 删除视图
- 序列概览:
  - 创建、使用和修改序列
  - 高速缓存序列值
  - NEXTVAL 和 CURRVAL 伪列
- 索引概览
  - 创建、删除索引
- 同义词概览
  - 创建、删除同义词



版权所有 © 2010, Oracle。保留所有权利。

# 序列

对象	说明
表	基本存储单元，由行组成
视图	逻辑上代表一个或多个表中数据的子集
序列	用于生成数字值
索引	提高某些查询的性能
同义词	给出对象的替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

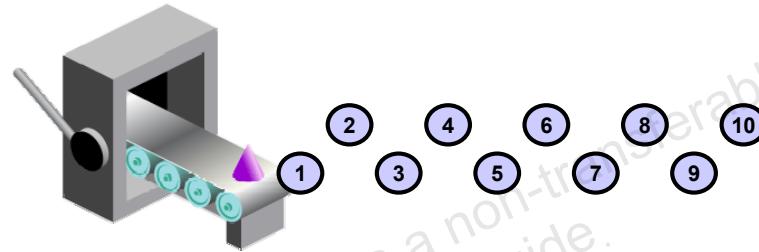
## 序列

序列是一个用于创建整数值的数据库对象。可以创建序列，然后再用其生成编号。

# 序列

序列具有如下特点：

- 可以自动生成唯一编号
- 是一个可共享的对象
- 可用于创建主键值
- 替换应用程序代码
- 如果将序列高速缓存到内存中，则访问序列值的效率会有所提高



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 序列（续）

序列是用户创建的数据库对象，可由多个用户共享来生成整数。

可以通过定义一个序列来生成唯一值，或者收回编号后重新使用相同的编号。

序列的常见用途是创建主键值，每行的主键值必须是唯一的。序列由内部 Oracle 例行程序按递增（或递减）方式生成。由于可以减少编写生成序列的例行程序所需的应用程序代码量，因此使用该对象可以节省一些时间。

序列号的存储和生成与表无关。因此，同一序列可以用于多个表。

## CREATE SEQUENCE 语句：语法

定义一个可以自动生成序号的序列：

```
CREATE SEQUENCE sequence
[INCREMENT BY n]
[START WITH n]
[{MAXVALUE n | NOMAXVALUE}]
[{MINVALUE n | NOMINVALUE}]
[{CYCLE | NOCYCLE}]
[{CACHE n | NOCACHE}];
```

版权所有 © 2010, Oracle。保留所有权利。

### CREATE SEQUENCE 语句：语法

通过使用 CREATE SEQUENCE 语句可以自动生成序号。

在该语法中：

sequence	是序列生成器的名称
INCREMENT BY n	指定序列号之间的间隔，其中 n 是一个整数（如果省略此子句，则序列按 1 递增）
START WITH n	指定要生成的第一个序列号（如果省略此子句，则序列从 1 开始）
MAXVALUE n	指定序列可以生成的最大值
NOMAXVALUE	指定 $10^{27}$ 作为递增序列的最大值，而 -1 作为递减序列的最大值（这是默认选项）
MINVALUE n	指定最小的序列值
NOMINVALUE	指定 1 作为递增序列的最小值，而 $-(10^{26})$ 作为递减序列的最小值（这是默认选项）

## 创建序列

- 创建一个名为 DEPT\_DEPTID\_SEQ 的序列，将其用作 DEPARTMENTS 表的主键。
- 不使用 CYCLE 选项。

```
CREATE SEQUENCE dept_deptid_seq
    INCREMENT BY 10
    START WITH 120
    MAXVALUE 9999
    NOCACHE
    NOCYCLE;
```

CREATE SEQUENCE succeeded.

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 创建序列（续）

CYCLE | NOCYCLE

指定在达到最大值或最小值之后，序列是否继续生成值  
(NOCYCLE 是默认选项)

CACHE n | NOCACHE

指定 Oracle Server 预先分配并保留在内存中的值的数量  
(默认情况下，Oracle Server 会高速缓存 20 个值)

幻灯片示例中创建了一个名为 DEPT\_DEPTID\_SEQ 的序列，用作 DEPARTMENTS 表的 DEPARTMENT\_ID 列。该序列从 120 开始，不允许高速缓存，也不进行循环。

请勿在序列用于生成主键值时使用 CYCLE 选项，除非您有一个可靠的机制与序列循环相比可以更快地清除旧行。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“CREATE SEQUENCE”一节。

**注：**序列与表没有关系。通常，您应按序列的预期用途来命名序列。但是，序列可以在任何地方使用，而与其名称无关。

## NEXTVAL 和 CURRVAL 伪列

- NEXTVAL 会返回下一个可用的序列值。每次被引用时 NEXTVAL 都会返回一个唯一值，即使对于不同用户也是如此。
- CURRVAL 会获得当前序列值。
- 只有对序列发出 NEXTVAL 之后，CURRVAL 才能包含值。



版权所有 © 2010, Oracle。保留所有权利。

### NEXTVAL 和 CURRVAL 伪列

在创建一个序列之后，该序列会生成可以在表中使用的序号。通过使用 NEXTVAL 和 CURRVAL 伪列可以引用序列值。

NEXTVAL 伪列用于从指定的序列中提取连续的序列号。必须用序列名来限定 NEXTVAL。在引用 `sequence.NEXTVAL` 时，就会生成新的序列号，还会将当前的序列号放在 CURRVAL 中。

CURRVAL 伪列用于引用当前用户刚刚生成的序列号。但是，必须先用 NEXTVAL 在当前用户的会话中生成一个序列号，然后才能引用 CURRVAL。必须用序列名来限定 CURRVAL。在引用 `sequence.CURRVAL` 时，会显示返回给用户进程的最后一个值。

#### 使用 NEXTVAL 和 CURRVAL 的规则

可以在下列上下文中使用 NEXTVAL 和 CURRVAL:

- 不是子查询一部分的 SELECT 语句的 SELECT 列表
- INSERT 语句中子查询的 SELECT 列表
- INSERT 语句的 VALUES 子句
- UPDATE 语句的 SET 子句

## NEXTVAL 和 CURRVAL 伪列 (续)

不能在下列上下文中使用 NEXTVAL 和 CURRVAL:

- 视图的 SELECT 列表
- 带有 DISTINCT 关键字的 SELECT 语句
- 带有 GROUP BY、HAVING 或 ORDER BY 子句的 SELECT 语句
- SELECT、DELETE 或 UPDATE 语句中的子查询
- CREATE TABLE 或 ALTER TABLE 语句中的 DEFAULT 表达式

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Pseudocolumns”和“CREATE SEQUENCE”两节。

## 使用序列

- 在位置 ID 2500 中插入一个名为“Support”的新部门：

```
INSERT INTO departments(department_id,
                       department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
               'Support', 2500);
```

- 查看 DEPT\_DEPTID\_SEQ 序列的当前值：

```
SELECT dept_deptid_seq.CURRVAL
FROM dual;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用序列

幻灯片示例中将一个新部门插入到 DEPARTMENTS 表中。它使用 DEPT\_DEPTID\_SEQ 序列生成如下所示的新部门编号。

可以使用 *sequence\_name.CURRVAL* 查看序列的当前值，如第二个幻灯片示例所示。查询输出如下所示：

	CURRVAL
1	120

假定您现在要聘用某些雇员作为新部门的工作人员。在要对所有新雇员执行的 INSERT 语句中可以包含以下代码：

```
INSERT INTO employees (employee_id, department_id, ...)
VALUES (employees_seq.NEXTVAL, dept_deptid_seq.CURRVAL, ...);
```

注：上面的示例假设已经创建一个名为 EMPLOYEE\_SEQ 的序列来生成新雇员编号。

## 高速缓存序列值

- 将序列值高速缓存在内存中，这样可以更快地对这些值进行访问。
- 在发生以下情况时，序列值会出现间断：
  - 发生回退
  - 系统崩溃
  - 序列已用于其它表中



版权所有 © 2010, Oracle。保留所有权利。

### 高速缓存序列值

您可以将序列高速缓存在内存中，以便于更快地对这些值进行访问。当您首次引用序列时，序列值会被填充到高速缓存中。因此会从高速缓存序列中检索每次请求的下一个新序列值。用完最后一个序列值之后，就会在下一次请求序列时将序列的另一个高速缓存拖入到内存中。

#### 序列中的间断

虽然序列生成器会发出没有间断的序号，但是此操作的发生与提交或回退有关。因此，如果您回退一条包含序列的语句，则会丢失相应的序号。

另一个可能导致序列出现间断的事件是系统崩溃。如果序列值已高速缓存在内存中，那么在系统崩溃时就会丢失一些值。

因为序列不直接与表相关联，所以同一序列可用于多个表。但是，如果同一序列用于多个表，则每个表的序号可能会有间断。

## 修改序列

更改增量值、最大值、最小值、循环选项或高速缓存选项：

```
ALTER SEQUENCE dept_deptid_seq
  INCREMENT BY 20
  MAXVALUE 999999
  NOCACHE
  NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq succeeded.
```

版权所有 © 2010, Oracle。保留所有权利。

### 修改序列

如果序列达到 MAXVALUE 限制，则序列不会再分配额外的值，此时您会收到一条错误消息，指明序列超出了 MAXVALUE。要继续使用该序列，可以使用 ALTER SEQUENCE 语句修改该序列。

#### 语法

```
ALTER SEQUENCE sequence
  [INCREMENT BY n]
  [{MAXVALUE n | NOMAXVALUE}]
  [{MINVALUE n | NOMINVALUE}]
  [{CYCLE | NOCYCLE}]
  [{CACHE n | NOCACHE}];
```

在此语法中，*sequence* 是序列生成器的名称。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“ALTER SEQUENCE”一节。

## 修改序列的准则

- 您必须是序列的所有者或拥有序列的 ALTER 权限。
- 修改只会影响以后生成的序列号。
- 如果要从另一编号处重新开始，则必须删除原有序列后重新创建。
- 在修改过程中会执行一些验证操作。
- 要删除序列，请使用 DROP 语句：

```
DROP SEQUENCE dept_deptid_seq;
DROP SEQUENCE dept_deptid_seq succeeded.
```

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

### 修改序列的准则

- 要修改一个序列，您必须是此序列的所有者或拥有序列的 ALTER 权限。要删除一个序列，您必须是此序列的所有者或拥有此序列的 DROP ANY SEQUENCE 权限。
- ALTER SEQUENCE 语句只会影响以后生成的序列号。
- 使用 ALTER SEQUENCE 语句不能更改 START WITH 选项。如果要从另一编号处重新开始，则必须删除原有序列后重新创建。
- 在修改过程中会执行一些验证操作。例如，不能强制实施一个小于当前序列号的新 MAXVALUE。

```
ALTER SEQUENCE dept_deptid_seq
    INCREMENT BY 20
    MAXVALUE 90
    NOCACHE
    NOCYCLE;
```

- 错误：

Error report:

SQL Error: ORA-04009: MAXVALUE cannot be made to be less than the current value  
 04009. 00000 - "MAXVALUE cannot be made to be less than the current value"  
 \*Cause: the current value exceeds the given MAXVALUE  
 \*Action: make sure that the new MAXVALUE is larger than the current value

## 课程安排

- 视图概览:
  - 创建、修改和检索视图中的数据
  - 对视图执行 DML 操作
  - 删除视图
- 序列概览:
  - 创建、使用和修改序列
  - 高速缓存序列值
  - NEXTVAL 和 CURRVAL 伪列
- 索引概览
  - 创建、删除索引
- 同义词概览
  - 创建、删除同义词



版权所有 © 2010, Oracle。保留所有权利。

# 索引

对象	说明
表	基本存储单元，由行组成
视图	逻辑上代表一个或多个表中数据的子集
序列	用于生成数字值
索引	提高某些查询的性能
同义词	给出对象的替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

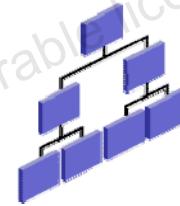
## 索引

索引是数据库对象，可以通过创建索引来提高一些查询的性能。在您创建主键或唯一约束条件时，服务器会同时自动创建索引。

# 索引

索引具有以下特点：

- 是一个方案对象
- Oracle Server 可用来通过指针加快行检索速度
- 可通过使用快速路径访问方法迅速找到数据来减少磁盘的输入/输出 (I/O)
- 与建立索引的表无关
- 由 Oracle Server 自动使用和维护



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 索引（续）

Oracle Server 索引是一个方案对象，可以通过指针加快行检索速度。可以显式创建索引，也可以自动创建索引。如果没有在列上建立索引，则会对整个表进行扫描。

使用索引可以直接而快速的访问表中的行。其作用是通过使用索引路径快速找到数据来减少磁盘的 I/O。索引由 Oracle Server 自动使用和维护。创建索引之后，就不需要用户直接执行任何操作了。

索引在逻辑上和实际上都独立于建立索引的表。这意味着可以在任何时候创建或删除索引，而不会对基表或其它索引产生任何影响。

注：在删除表时，会另外删除相应的索引。

有关详细信息，请参阅《Oracle Database Concepts 11g, Release 1 (11.1)》中的“Schema Objects: Indexes”一节。

## 如何创建索引

- 自动创建：如果在表定义中定义了 PRIMARY KEY 或 UNIQUE 约束条件，则会自动创建一个唯一的索引。



- 手动创建：用户可以通过对列创建非唯一的索引来加快行访问速度。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 如何创建索引

可以创建两种类型的索引。

- 唯一索引：**如果您在表中定义的列具有 PRIMARY KEY 或 UNIQUE 约束条件，则 Oracle Server 会自动创建此类型的索引。索引的名称和约束条件的名称相同。
- 非唯一索引：**这种类型的索引可以由用户创建。例如，可以通过在查询中为联接创建一个 FOREIGN KEY 列索引来提高检索速度。

**注：**可以手动创建唯一索引，但是建议您创建唯一约束条件，这样可隐式创建唯一索引。

## 创建索引

- 对一个或多个列创建索引：

```
CREATE [UNIQUE] [BITMAP] INDEX index
ON table (column[, column]...) ;
```

- 提高对 EMPLOYEES 表中 LAST\_NAME 列的查询访问速度：

```
CREATE INDEX emp_last_name_idx
ON employees(last_name) ;
CREATE INDEX succeeded.
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 创建索引

通过发出 CREATE INDEX 语句可以对一个或多个列创建索引。

在该语法中：

- index 是索引的名称
- table 是表名称
- column 是表中要建立索引的列的名称

指定 UNIQUE 时指出单个或多个列的列值（索引所基于的）必须为唯一。指定 BITMAP 时指出会使用位图为每个不同的键创建索引，而不用是分别为每一行建立索引。建立位图索引时会将与键值关联的 rowids 存储为位图。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“CREATE INDEX”一节。

## 索引创建准则

### 在以下情况下应创建索引：

列包含值的范围很广
列包含大量空值
在 WHERE 子句或联接条件中频繁使用一个或多个列
表很大，但是预计大多数查询要检索的行小于表中行数的 2% 至 4%

### 请勿在以下情况下创建索引：

这些列没有频繁用做查询中的条件
表比较小，或者预计大多数查询要检索的行超过表中行数的 2% 至 4%
表更新频繁
在表达式中已引用索引列

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 索引创建准则

### 索引多未必会更好

并不是表索引越多查询速度越快。在具有索引的表上提交每项 DML 操作后，都意味着必须更新相应的索引。与表关联的索引越多，在执行 DML 操作之后，Oracle Server 为更新全部索引所做的工作就越多。

### 什么情况下要创建索引

因此，仅在以下情况下才应创建索引：

- 列包含值的范围很广
- 列包含大量的空值
- 在 WHERE 子句或联接条件中频繁使用一个或多个列
- 表很大，但是预计大多数查询要检索的行小于行数的 2% 至 4%

请注意，如果要强制实现唯一性，则应在表定义中定义一个唯一约束条件。此时可以自动创建一个唯一索引。

## 删除索引

- 使用 `DROP INDEX` 命令可从数据字典中删除索引：

```
DROP INDEX index;
```

- 从数据字典中删除 `emp_last_name_idx` 索引：

```
DROP INDEX emp_last_name_idx;
```

```
DROP INDEX emp_last_name_idx succeeded.
```

- 要删除索引，您必须是索引的所有者或拥有 `DROP ANY INDEX` 权限。



版权所有 © 2010, Oracle。保留所有权利。

### 删除索引

您不能修改索引。要更改索引，必须先删除它，然后重新创建。

发出 `DROP INDEX` 语句后，可从数据字典中删除索引定义。要删除索引，您必须是索引的所有者或拥有 `DROP ANY INDEX` 权限。

在此语法中，`index` 是索引的名称。

**注：**如果删除了一个表，则会自动删除索引和约束条件，但会保留视图和序列。

## 课程安排

- 视图概览:
  - 创建、修改和检索视图中的数据
  - 对视图执行 DML 操作
  - 删除视图
- 序列概览:
  - 创建、使用和修改序列
  - 高速缓存序列值
  - NEXTVAL 和 CURRVAL 伪列
- 索引概览
  - 创建、删除索引
- 同义词概览
  - 创建、删除同义词

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

# 同义词

对象	说明
表	基本存储单元，由行组成
视图	逻辑上代表一个或多个表中数据的子集
序列	用于生成数字值
索引	提高某些查询的性能
同义词	给出对象的替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 同义词

同义词是数据库对象，利用同义词可以通过另一个名称来调用表。可以通过创建同义词来给出表的替代名称。

## 创建对象的同义词

通过创建同义词（对象的另一个名称）可以简化对对象的访问。使用同义词，您可以：

- 更方便地引用其他用户拥有的表
- 缩短冗长的对象名

```
CREATE [PUBLIC] SYNONYM synonym
FOR object;
```



版权所有 © 2010, Oracle。保留所有权利。

### 创建对象的同义词

要引用其他用户拥有的表，需要为表名加上一个前缀，即该表的创建者姓名，后跟一个句点。创建同义词后，就不必使用方案来限定对象名了，同时还为您提供提供了表、视图、序列、过程或其它对象的替代名称。此方法在对象名（例如视图）冗长时特别有用。

在该语法中：

PUBLIC	创建所有用户都可以访问的同义词
<i>synonym</i>	是要创建的同义词的名称
<i>object</i>	标识要为其创建同义词的对象

#### 准则

- 该对象不能包含在程序包中。
- 私用同义词名一定不能与同一用户拥有的其它所有对象的名称相同。

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“CREATE SYNONYM”一节。

## 创建和删除同义词

- 为 DEPT\_SUM\_VU 视图创建一个简短的名称:

```
CREATE SYNONYM d_sum
FOR dept_sum_vu;
CREATE SYNONYM succeeded.
```

- 删除同义词:

```
DROP SYNONYM d_sum;
DROP SYNONYM d_sum succeeded.
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 创建和删除同义词

### 创建同义词

幻灯片示例中为 DEPT\_SUM\_VU 视图创建了一个同义词，目的是为了便于更迅速地进行引用。

数据库管理员可以创建所有用户都可以访问的公用同义词。下面的示例为 Alice 的 DEPARTMENTS 表创建一个名为 DEPT 的公用同义词:

```
CREATE PUBLIC SYNONYM dept
CREATE SYNONYM succeeded.
```

### 删除同义词

要删除同义词，请使用 DROP SYNONYM 语句。只有数据库管理员可以删除公用同义词。

```
DROP PUBLIC SYNONYM dept;
```

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“DROP SYNONYM”一节。

## 小测验

索引必须手动创建并用于加速对表中行的访问。

- 1. 正确**
- 2. 错误**

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

**答案: 2**

注: 索引设计用于提高查询性能。但是, 并不是所有索引都必须手动创建。如果您在表中定义的列具有 PRIMARY KEY 或 UNIQUE 约束条件, 则 Oracle Server 会自动创建索引。

## 小结

在本课中，您应该已经学会：

- 创建、使用和删除视图
- 使用序列生成器自动生成序列号
- 通过创建索引来提高查询检索速度
- 使用同义词为对象提供替代名称

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 小结

在本课中，您应该已经掌握有关数据库对象（如视图、序列、索引和同义词）的知识。

## 练习 11：第 2 部分概览

本练习包含以下主题：

- 创建序列
- 使用序列
- 创建非唯一索引
- 创建同义词

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 练习 11：第 2 部分概览

本课练习的第 2 部分为您提供创建和使用序列、索引和同义词的各种练习。

学完本课后请完成问题 7-10。

---

## 附录 A

### 练习和解答

---

## 目录

第 I 课的练习 .....	3
练习 I-1: 简介 .....	4
练习解答 I-1: 简介 .....	5
第 1 课的练习 .....	11
练习 1-1: 使用 SQL SELECT 语句检索数据 .....	12
练习解答 1-1: 使用 SQL SELECT 语句检索数据 .....	17
第 2 课的练习 .....	20
练习 2-1: 对数据进行限制和排序 .....	21
练习解答 2-1: 对数据进行限制和排序 .....	25
第 3 课的练习 .....	28
练习 3-1: 使用单行函数定制输出 .....	29
练习解答 3-1: 使用单行函数定制输出 .....	33
第 4 课的练习 .....	36
练习 4-1: 使用转换函数和条件表达式 .....	37
练习解答 4-1: 使用转换函数和条件表达式 .....	40
第 5 课的练习 .....	42
练习 5-1: 使用组函数报告聚集数据 .....	43
练习解答 5-1: 使用组函数报告聚集数据 .....	46
第 6 课的练习 .....	48
练习 6-1: 使用联接显示多个表中的数据 .....	49
练习解答 6-1: 使用联接显示多个表中的数据 .....	53
第 7 课的练习 .....	55
练习 7-1: 使用子查询来解决查询 .....	56
练习解答 7-1: 使用子查询来解决查询 .....	58
第 8 课的练习 .....	60
练习 8-1: 使用集合运算符 .....	61
练习解答 8-1: 使用集合运算符 .....	63
第 9 课的练习 .....	65
练习 9-1: 处理数据 .....	66
练习解答 9-1: 处理数据 .....	70
第 10 课的练习 .....	74
练习 10-1: 使用 DDL 语句创建和管理表 .....	75
练习解答 10-1: 使用 DDL 语句创建和管理表 .....	77
第 11 课的练习 .....	80
练习 11-1: 创建其它方案对象 .....	81
练习解答 11-1: 创建其它方案对象 .....	83
附录 F 的练习 .....	85
练习 F-1: Oracle 联接语法 .....	86
练习解答 F-1: Oracle 联接语法 .....	90

## 第 I 课的练习

在本练习中，您将执行以下任务：

- 启动 Oracle SQL Developer 并创建一个到 ora1 帐户的新连接。
- 使用 Oracle SQL Developer 查看 ora1 帐户中的数据对象。ora1 帐户包含 HR 方案表。

请注意，练习文件的位置如下：

\home\oracle\labs\sql1\labs

如果系统要求您保存任何练习文件，请将其保存到上述位置。

在任何一个练习中，都可能存在以词语“如果您有时间”或“如果您要接受更多的挑战”为开始的练习。仅当您在指定时间内完成其它所有练习之后，并希望进一步挑战您的技能时，再做这些练习。

请认真并准确地完成这些练习。您可以练习保存和运行命令文件。只要您有问题，可随时咨询您的教师。

### 附注

- 所有书面练习都使用 Oracle SQL Developer 作为开发环境。尽管建议使用 Oracle SQL Developer，但也可以使用本课程中提供的 SQL\*Plus。
- 对于任何查询，从数据库检索的行的序列可能与所示屏幕快照不同。

## 练习 I-1：简介

这是本课程中众多练习中的第一个练习。本练习结尾处提供了解答（如果需要）。这些练习包括了相应课程中提供的大部分主题。

### 启动 Oracle SQL Developer

- 1) 使用 SQL Developer 桌面图标启动 Oracle SQL Developer。

### 创建新的 Oracle SQL Developer 数据库连接

- 2) 要创建新的数据库连接，请在连接导航器中，右键单击“Connections（连接）”。从菜单中选择“New Connection（新建连接）”。此时出现“New/Select Database Connection（新建/选择数据库连接）”对话框。
- 3) 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）: ora1
- c) Password（口令）: ora1
- d) Hostname（主机名）: localhost
- e) Port（端口）: 1521
- f) SID: ORCL

请确保选中“Save Password（保存口令）”复选框。

### 使用 Oracle SQL Developer Database Connection 进行测试和连接

- 4) 测试此新连接。
- 5) 如果状态为“Success（成功）”，则可使用这个新连接连接到数据库。

### 在连接导航器中浏览表

- 6) 在连接导航器中，查看在“Tables（表）”节点中可用的对象。验证是否存在以下表：

COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS  
LOCATIONS  
REGIONS

- 7) 浏览 EMPLOYEES 表的结构。
- 8) 查看 DEPARTMENTS 表的数据。

## 练习解答 I-1：简介

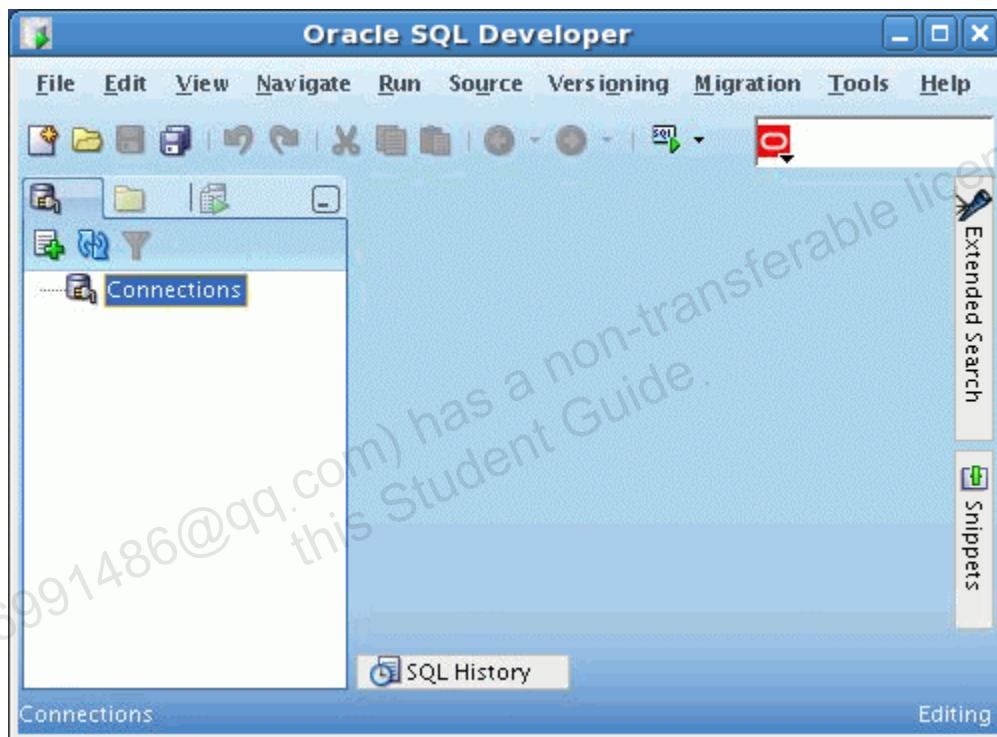
### 启动 Oracle SQL Developer

1) 使用 SQL Developer 桌面图标启动 Oracle SQL Developer。

a) 双击 SQL Developer 桌面图标。



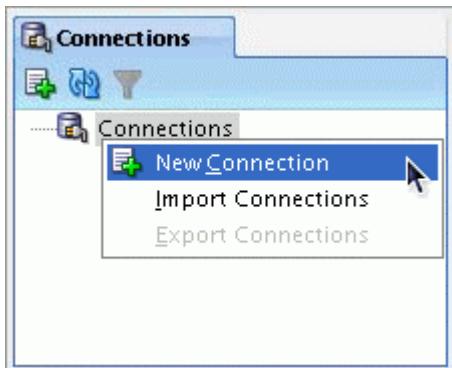
SQL Developer 界面显示。



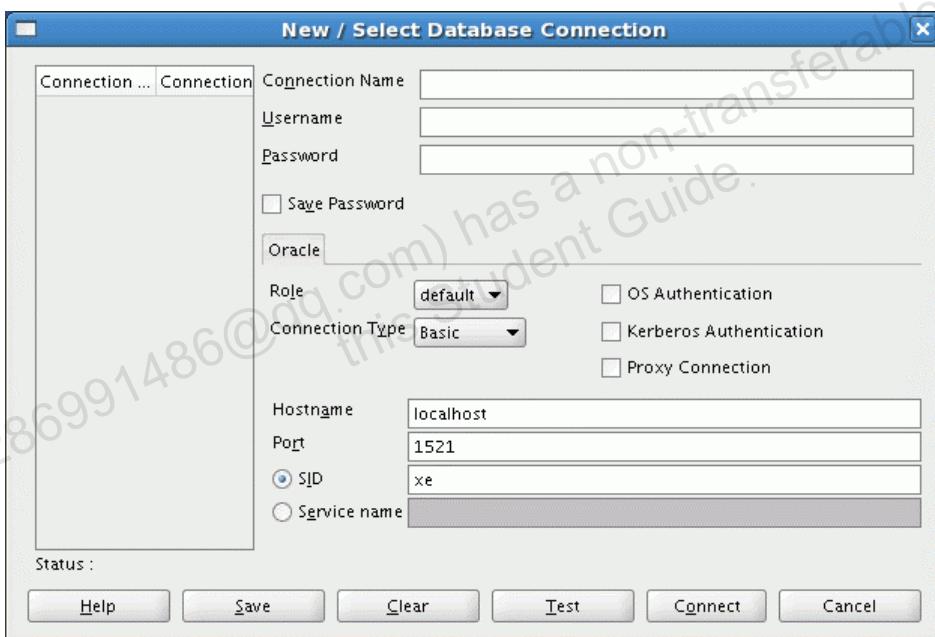
## 练习解答 I-1：简介（续）

### 创建新的 Oracle SQL Developer 数据库连接

- 2) 要创建新的数据库连接，请在“Connections Navigator（连接导航器）”中，右键单击“Connections（连接）”并从菜单中选择“New Connection（新建连接）”。



此时出现“New/Select Database Connection（新建/选择数据库连接）”对话框。

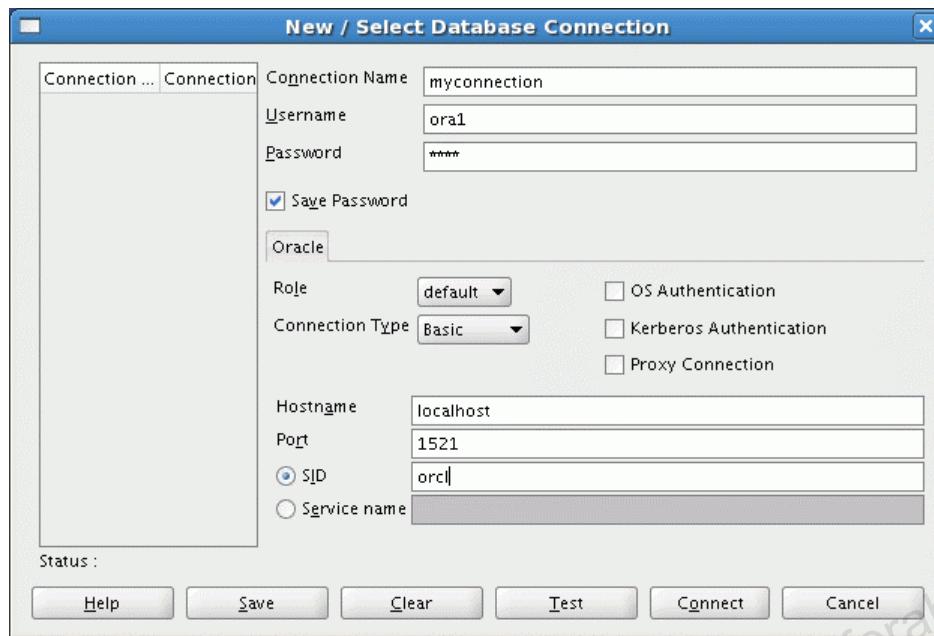


- 3) 使用以下信息创建数据库连接：

- Connection Name（连接名）: myconnection
- Username（用户名）: ora1
- Password（口令）: ora1
- Hostname（主机名）: localhost
- Port（端口）: 1521
- SID: ORCL

## 练习解答 I-1：简介（续）

请确保选中“Save Password（保存口令）”复选框。



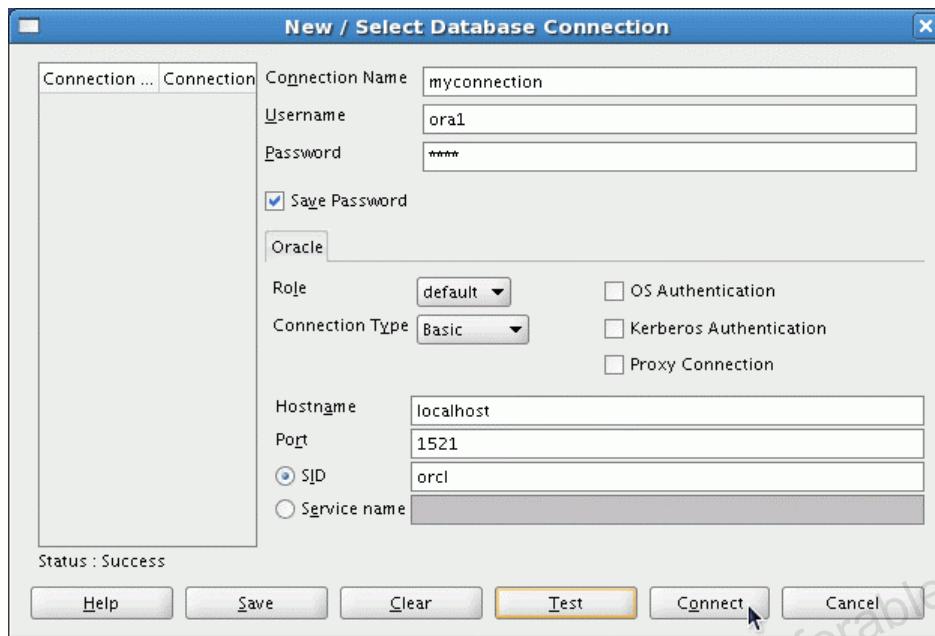
使用 Oracle SQL Developer Database Connection 进行测试和连接

- 4) 测试此新连接。

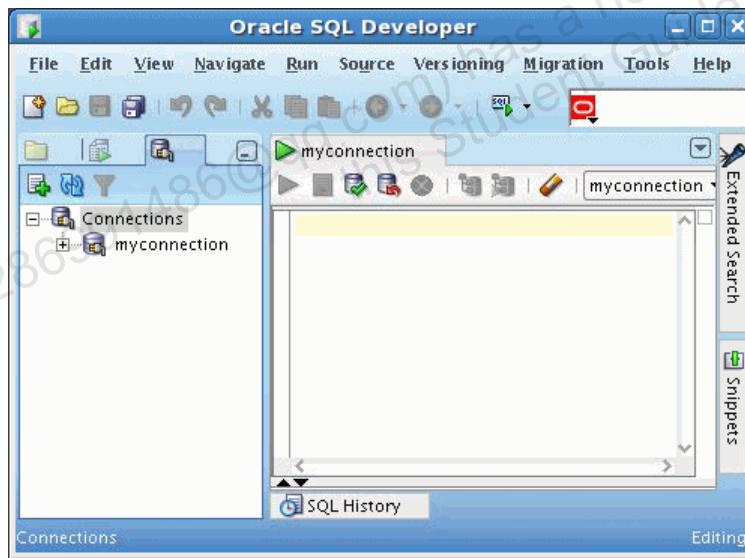


## 练习解答 I-1：简介（续）

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



在创建了一个连接后，用于此连接的“SQL Worksheet（SQL 工作表）”将自动打开。

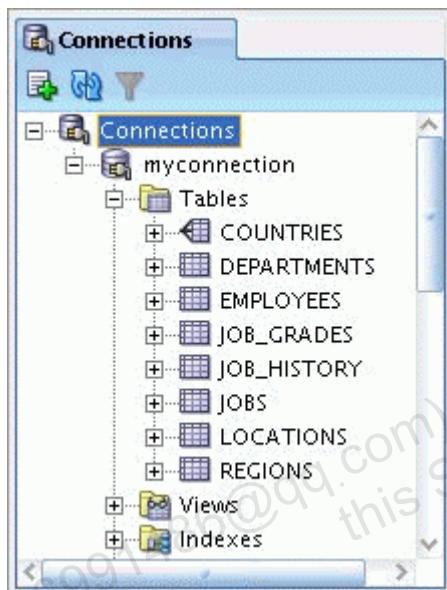


## 练习解答 I-1：简介（续）

在连接导航器中浏览表

- 6) 在连接导航器中，查看在“Tables（表）”节点中可用的对象。验证是否存在以下表：

COUNTRIES  
DEPARTMENTS  
EMPLOYEES  
JOB\_GRADES  
JOB\_HISTORY  
JOBS  
LOCATIONS  
REGIONS



## 练习解答 I-1：简介（续）

7) 浏览 EMPLOYEES 表的结构。

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Tables' tree view shows the 'EMPLOYEES' table selected. The main panel displays the 'EMPLOYEES' table structure with the following columns:

Column Name	Data Type	Nullable	Default Value
EMPLOYEE_ID	NUMBER(6,0)	No	(null)
FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)
LAST_NAME	VARCHAR2(25 BYTE)	No	(null)
EMAIL	VARCHAR2(25 BYTE)	No	(null)
PHONE_NUMBER	VARCHAR2(20 BYTE)	Yes	(null)
HIRE_DATE	DATE	No	(null)
JOB_ID	VARCHAR2(10 BYTE)	No	(null)
SALARY	NUMBER(8,2)	Yes	(null)
COMMISSION_PCT	NUMBER(2,2)	No	(null)
MANAGER_ID	NUMBER(6,0)	No	(null)
DEPARTMENT_ID	NUMBER(3,0)	No	(null)

8) 查看 DEPARTMENTS 表的数据。

The screenshot shows the Oracle SQL Developer interface. On the left, the 'Tables' tree view shows the 'DEPARTMENTS' table selected. The main panel displays the 'DEPARTMENTS' table data with the following rows:

DEPARTMENT_ID	DEPARTMENT_NAME
1	10 Administration
2	20 Marketing
3	30 Shipping
4	40 IT
5	50 Sales
6	60 Executive
7	70 Accounting
8	80 Contracting

## 第 1 课的练习

在此练习中，您将编写几个简单的 SELECT 查询。这些查询应运用您在本课中学习过的大多数 SELECT 子句和运算。

## 练习 1-1：使用 SQL SELECT 语句检索数据

### 第 1 部分

测试您的知识：

- 1) 下面的 SELECT 语句能够成功执行：

```
SELECT last_name, job_id, salary AS Sal
FROM   employees;
```

对/错

- 2) 下面的 SELECT 语句能够成功执行：

```
SELECT *
FROM   job_grades;
```

对/错

- 3) 下面的语句中有四处编码错误。是否能将其标识出来？

```
SELECT      employee_id, last_name
sal x 12  ANNUAL SALARY
FROM       employees;
```

### 第 2 部分

在开始练习前，请注意以下几点：

- 将所有练习文件保存到以下位置：/home/oracle/labs/sql1/labs。
- 在 SQL 工作表中输入 SQL 语句。要在 SQL Developer 中保存脚本，请确保需要使用的 SQL 工作表处于活动状态，然后，从“File（文件）”菜单中选择“Save As（另存为）”，将 SQL 语句另存为 lab\_<lessonno>\_<stepno>.sql 脚本。在修改现有脚本时，请一定要使用“Save As（另存为）”并使用其它文件名来进行保存。
- 要运行查询，请单击 SQL 工作表中的“Execute Statement（执行语句）”图标。或者，也可以按 [F9]。对于 DML 和 DDL 语句，请使用“Run Script（运行脚本）”图标或按 [F5]。
- 在执行查询后，请确保未在同一工作表中输入下一个查询。请打开一个新工作表。

您是 Acme Corporation 的一名 SQL 程序员。您的第一个任务是根据人力资源表中的数据创建一些报表。

## 练习 1-1：使用 SQL SELECT 语句检索数据（续）

- 4) 您的第一个任务是确定 DEPARTMENTS 表的结构及其内容。

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

4 rows selected

	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

- 5) 确定 EMPLOYEES 表的结构。

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)

11 rows selected

HR 部门需要一个查询，用于显示每位员工的姓氏、职务 ID、聘用日期和员工 ID，而且希望首先显示员工 ID。请为 HIRE\_DATE 列提供一个别名 STARTDATE。将您的 SQL 语句保存到名为 lab\_01\_05.sql 的文件中，以便可以将该文件发送到 HR 部门。

## 练习 1-1：使用 SQL SELECT 语句检索数据（续）

- 6) 测试 lab\_01\_05.sql 文件中的查询，确保其运行是正确的。

注：在执行查询后，请确保未在同一工作表中输入下一个查询。请打开一个新工作表。

	EMPLOYEE_ID	LAST_NAME	JOB_ID	STARTDATE
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

19	176	Taylor	SA_REP	24-MAR-98
20	178	Grant	SA_REP	24-MAY-99

- 7) HR 部门需要一个查询，用于显示 EMPLOYEES 表中所有唯一的职务 ID。

JOB_ID
1 AC_ACCOUNT
2 AC_MGR
3 AD_ASST
4 AD_PRES
5 AD_VP
6 IT_PROG
7 MK_MAN
8 MK_REP
9 SA_MAN
10 SA_REP
11 ST_CLERK
12 ST_MAN

## 练习 1-1：使用 SQL SELECT 语句检索数据（续）

### 第 3 部分

如果您有时间，请完成以下练习：

- 8) HR 部门希望员工报表的列标题更具描述性。将该语句从 lab\_01\_05.sql 复制到一个新 SQL 工作表中。将列标题分别命名为“Emp #”、“Employee”、“Job”和“Hire Date”。然后，再次运行查询。

	Emp #	Employee	Job	Hire Date
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

19	176	Taylor	SA_REP	24-MAR-98
20	178	Grant	SA_REP	24-MAY-99

- 9) HR 部门需要一个包含所有员工及其职务 ID 的报表。以级联方式显示姓氏和职务 ID（以逗号或空格分隔），并将该列命名为“Employee and Title”。

Employee and Title
1 Abel, SA_REP
2 Davies, ST_CLERK
3 De Haan, AD_VP
4 Ernst, IT_PROG
5 Fay, MK_REP

...

19 Whalen, AD_ASST
20 Zlotkey, SA_MAN

## 练习 1-1：使用 SQL SELECT 语句检索数据（续）

如果您要接受更多的挑战，请完成下面的练习：

- 10) 为了了解 EMPLOYEES 表中的数据，请创建一个查询来显示该表中的所有数据。  
用逗号分隔输出中的每一列。将该列标题命名为“THE\_OUTPUT”。

	THE_OUTPUT
1	200,Jennifer,Whalen,JWHALEN,515.123.4444,AD_ASST,101,17-SEP-87,4400,,10
2	201,Michael,Hartstein,MHARTSTE,515.123.5555,MK_MAN,100,17-FEB-96,13000,,20
3	202,Pat,Fay,PFAY,603.123.6666,MK_REP,201,17-AUG-97,6000,,20
4	205,Shelley,Higgins,SHIGGINS,515.123.8080,AC_MGR,101,07-JUN-94,12000,,110
5	206,William,Cietz,WGIETZ,515.123.8181,AC_ACCOUNT,205,07-JUN-94,8300,,110
...	
19	176,Jonathon,Taylor,JTAYLOR,011.44.1644.429265,SA REP,149,24-MAR-98,8600,,280
20	178,Kimberely,Grant,KGRANT,011.44.1644.429263,SA REP,149,24-MAY-99,7000,,15,

## 练习解答 1-1：使用 SQL SELECT 语句检索数据

### 第 1 部分

测试您的知识：

- 1) 下面的 SELECT 语句能够成功执行：

```
SELECT last_name, job_id, salary AS Sal  
FROM   employees;
```

对/错

- 2) 下面的 SELECT 语句能够成功执行：

```
SELECT *  
FROM   job_grades;
```

对/错

- 3) 下面的语句中有四处编码错误。是否能将其标识出来？

```
SELECT      employee_id, last_name  
sal x 12  ANNUAL SALARY  
FROM       employees;
```

- **EMPLOYEES** 表并不包含名为 **sal** 的列。该列名为 **SALARY**。
- 乘法运算符为 **\***，而不是第 2 行显示的 **x**。
- **ANNUAL SALARY** 别名不能包括空格。该别名应为 **ANNUAL\_SALARY** 或者放在双引号中。
- **LAST\_NAME** 列后缺少一个逗号。

## 练习解答 1-1：使用 SQL SELECT 语句检索数据（续）

### 第 2 部分

您是 Acme Corporation 的一名 SQL 程序员。您的第一个任务是根据人力资源表中的数据创建一些报表。

- 4) 您的第一个任务是确定 DEPARTMENTS 表的结构及其内容。

- a. 要确定 DEPARTMENTS 表结构，请使用以下查询：

```
DESCRIBE departments
```

- b. 要查看包含在 DEPARTMENTS 表中的数据，请使用以下查询：

```
SELECT *
FROM   departments;
```

- 5) 确定 EMPLOYEES 表的结构。

```
DESCRIBE employees
```

HR 部门需要一个查询，用于显示每位员工的姓氏、职务 ID、聘用日期和员工 ID，而且希望首先显示员工 ID。请为 HIRE\_DATE 列提供一个别名 STARTDATE。将您的 SQL 语句保存到名为 lab\_01\_05.sql 的文件中，以便可以将该文件发送到 HR 部门。

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

- 6) 测试 lab\_01\_05.sql 文件中的查询，确保其运行是正确的。

```
SELECT employee_id, last_name, job_id, hire_date StartDate
FROM   employees;
```

- 7) HR 部门需要一个查询，用于显示 EMPLOYEES 表中所有唯一的职务 ID。

```
SELECT DISTINCT job_id
FROM   employees;
```

## 练习解答 1-1：使用 SQL SELECT 语句检索数据（续）

### 第 3 部分

如果您有时间，请完成以下练习：

- 8) HR 部门希望员工报表的列标题更具描述性。将该语句从 lab\_01\_05.sql 复制到一个新 SQL 工作表中。将列标题分别命名为“Emp #”、“Employee”、“Job”和“Hire Date”。然后，再次运行查询。

```
SELECT employee_id "Emp #", last_name "Employee",
       job_id "Job", hire_date "Hire Date"
  FROM employees;
```

- 9) HR 部门需要一个包含所有员工及其职务 ID 的报表。以级联方式显示姓氏和职务 ID（以逗号或空格分隔），并将该列命名为“Employee and Title”。

```
SELECT last_name||', '||job_id "Employee and Title"
  FROM employees;
```

如果您要接受更多的挑战，请完成下面的练习：

- 10) 为了了解 EMPLOYEES 表中的数据，请创建一个查询来显示该表中的所有数据。用逗号分隔输出中的每一列。将该列标题命名为“THE\_OUTPUT”。

```
SELECT employee_id || ',' || first_name || ',' || last_name
      || ',' || email || ',' || phone_number || ',' || job_id
      || ',' || manager_id || ',' || hire_date || ',' ||
      || salary || ',' || commission_pct || ',' ||
department_id
      THE_OUTPUT
  FROM employees;
```

## 第 2 课的练习

在本练习中，您将创建很多报表，其中包括使用 WHERE 子句和 ORDER BY 子句的对帐表。通过使用 & 替代变量，SQL 语句更易于重复使用，而且具有更高的通用性。

## 练习 2-1：对数据进行限制和排序

HR 部门需要您协助创建一些查询。

- 因为预算问题，HR 部门需要一个报表，用于显示薪金高于 \$12,000 的员工的姓氏和薪金。将您的 SQL 语句保存到名为 lab\_02\_01.sql 的文件中。运行您的查询。

	LAST_NAME	SALARY
1	Hartstein	13000
2	King	24000
3	Kochhar	17000
4	De Haan	17000

- 打开一个新 SQL 工作表。创建一个报表，用于显示编号为 176 的员工的姓氏和部门编号。运行该查询。

	LAST_NAME	DEPARTMENT_ID
1	Taylor	80

- HR 部门需要查找高薪与低薪员工。修改 lab\_02\_01.sql，使其显示其薪金在 \$5,000 到 \$12,000 范围以外的所有员工的姓氏和薪金。将您的 SQL 语句保存到名为 lab\_02\_03.sql 的文件中。

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	King	24000
4	Kochhar	17000
5	De Haan	17000
6	Lorentz	4200
7	Rajs	3500
8	Davies	3100
9	Matos	2600
10	Vargas	2500

- 创建一个报表，用于显示姓氏为 Matos 和 Taylor 的员工的姓氏、职务 ID 和聘用日期。按聘用日期升序顺序对查询进行排序。

	LAST_NAME	JOB_ID	HIRE_DATE
1	Matos	ST_CLERK	15-MAR-98
2	Taylor	SA_REP	24-MAR-98

## 练习 2-1：对数据进行限制和排序（续）

- 5) 按姓名的字母顺序显示部门 20 或 50 中所有员工的姓氏和部门 ID。

LAST_NAME	DEPARTMENT_ID
1 Davies	50
2 Fay	20
3 Hartstein	20
4 Matos	50
5 Mourgos	50
6 Rajs	50
7 Vargas	50

- 6) 修改 lab\_02\_03.sql，使其显示其薪金在 \$5,000 和 \$12,000 之间且部门为 20 或 50 的员工的姓氏和薪金。分别标记列 Employee 和 Monthly Salary。再次将 lab\_02\_03.sql 保存为 lab\_02\_06.sql。运行 lab\_02\_06.sql 中的语句。

Employee	Monthly Salary
1 Fay	6000
2 Mourgos	5800

- 7) HR 部门需要一个报表，用于显示 1994 年聘用的所有员工的姓氏和聘用日期。

LAST_NAME	HIRE_DATE
1 Higgins	07-JUN-94
2 Gietz	07-JUN-94

- 8) 创建一个报表，用于显示没有经理的所有员工的姓氏和职位。

LAST_NAME	JOB_ID
1 King	AD_PRES

- 9) 创建一个报表，用于显示领取佣金的所有员工的姓氏、薪金和佣金。按薪金和佣金的降序顺序对数据进行排序。

在 ORDER BY 子句中使用列的数字位置。

LAST_NAME	SALARY	COMMISSION_PCT
1 Abel	11000	0.3
2 Zlotkey	10500	0.2
3 Taylor	8600	0.2
4 Grant	7000	0.15

## 练习 2-1：对数据进行限制和排序（续）

- 10) HR 部门的成员希望在使用您所编写的查询时拥有更多的灵活性。他们希望报表能够显示一些员工的姓氏和薪金，这些员工的薪金高于用户在系统提示下指定的金额。将此查询保存到名为 lab\_02\_10.sql 的文件中。如果您在收到提示后输入 12000，则报表会显示以下结果：

	LAST_NAME	SALARY
1	Hartstein	13000
2	King	24000
3	Kochhar	17000
4	De Haan	17000

- 11) HR 部门需要根据经理来运行报表。创建一个查询来提示用户输入一个经理 ID 并生成该经理的员工的员工 ID、姓氏、薪金和部门。HR 部门需要根据选定列对报表进行排序。您可以使用下列值测试数据：

manager\_id = 103, 按 last\_name 排序：

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	104	Ernst	6000	60
2	107	Lorentz	4200	60

manager\_id = 201, 按 salary 排序：

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	202	Fay	6000	20

manager\_id = 124, 按 employee\_id 排序：

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	141	Rajs	3500	50
2	142	Davies	3100	50
3	143	Matos	2600	50
4	144	Vargas	2500	50

## 练习 2-1：对数据进行限制和排序（续）

如果您有时间，请完成以下练习：

- 12) 显示姓名中第三个字母为“a”的所有员工的姓氏。

LAST_NAME
Grant
Whalen

- 13) 显示姓氏中有“a”和“e”的所有员工的姓氏。

LAST_NAME
Davies
De Haan
Hartstein
Whalen

如果您要接受更多的挑战，请完成下面的练习：

- 14) 显示职务为销售代表或仓储职员且薪金不等于\$2,500、\$3,500或\$7,000的所有员工的姓氏、职务和薪金。

LAST_NAME	JOB_ID	SALARY
Abel	SA_REP	11000
Taylor	SA_REP	8600
Davies	ST_CLERK	3100
Matos	ST_CLERK	2600

- 15) 修改 lab\_02\_06.sql，显示佣金率为20%的所有员工的姓名、薪金和佣金。

再次将 lab\_02\_06.sql 保存为 lab\_02\_15.sql。运行 lab\_02\_15.sql 中的语句。

Employee	Monthly Salary	Commission_Pct
Zlotkey	10500	0.2
Taylor	8600	0.2

## 练习解答 2-1：对数据进行限制和排序

HR 部门需要您协助创建一些查询。

- 由于预算问题，HR 部门需要一个报表，用于显示薪金超过 \$12,000 的员工的姓氏和薪金。将您的 SQL 语句另存为名为 lab\_02\_01.sql 的文件。运行您的查询。

```
SELECT last_name, salary
FROM employees
WHERE salary > 12000;
```

- 打开一个新 SQL 工作表。创建一个报表，用于显示编号为 176 的员工的姓氏和部门编号。

```
SELECT last_name, department_id
FROM employees
WHERE employee_id = 176;
```

- HR 部门需要查找高薪与低薪员工。修改 lab\_02\_01.sql，显示薪金不在 \$5,000 至 \$12,000 范围内的所有员工的姓氏和薪金。将您的 SQL 语句另存为 lab\_02\_03.sql。

```
SELECT last_name, salary
FROM employees
WHERE salary NOT BETWEEN 5000 AND 12000;
```

- 创建一个报表，用于显示姓氏为 Matos 和 Taylor 的员工的姓氏、职务 ID 和聘用日期。按聘用日期的升序对查询进行排序。

```
SELECT last_name, job_id, hire_date
FROM employees
WHERE last_name IN ('Matos', 'Taylor')
ORDER BY hire_date;
```

- 按姓名的字母顺序显示部门 20 或 50 中所有员工的姓氏和部门 ID。

```
SELECT last_name, department_id
FROM employees
WHERE department_id IN (20, 50)
ORDER BY last_name ASC;
```

- 修改 lab\_02\_03.sql，列出薪金在 \$5,000 到 \$12,000 之间且部门编号为 20 或 50 的员工的姓氏和薪金。将这两个列分别标记为 Employee 和 Monthly Salary。再次将 lab\_02\_03.sql 保存为 lab\_02\_06.sql。运行 lab\_02\_06.sql 中的语句。

```
SELECT last_name "Employee", salary "Monthly Salary"
FROM employees
WHERE salary BETWEEN 5000 AND 12000
AND department_id IN (20, 50);
```

## 练习解答 2-1：对数据进行限制和排序（续）

- 7) HR 部门需要一个报表，用于显示 1994 年聘用的所有员工的姓氏和聘用日期。

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%94';
```

- 8) 创建一个报表，用于显示没有经理的所有员工的姓氏和职位。

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

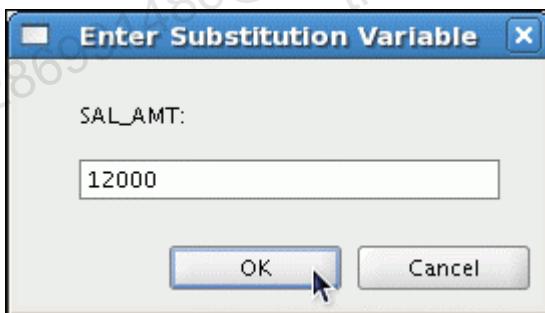
- 9) 创建一个报表，用于显示领取佣金的所有员工的姓氏、薪金和佣金。按薪金和佣金的降序顺序对数据进行排序。在 ORDER BY 子句中使用列的数字位置。

```
SELECT last_name, salary, commission_pct
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY 2 DESC, 3 DESC;
```

- 10) HR 部门的成员希望在使用您所编写的查询时拥有更多的灵活性。他们希望报表能够显示一些员工的姓氏和薪金，这些员工的薪金高于用户在系统提示下指定的金额。您可以使用在练习 1 中创建的查询并进行相应的修改。将此查询保存到名为 lab\_02\_10.sql 的文件中。

```
SELECT last_name, salary
FROM employees
WHERE salary > &sal_amt;
```

提示输入值时在对话框中输入“12000”。单击“OK（确定）”。



## 练习解答 2-1：对数据进行限制和排序（续）

- 11) HR 部门需要根据经理来运行报表。创建一个查询来提示用户输入一个经理 ID 并生成该经理的员工的员工 ID、姓氏、薪金和部门。HR 部门需要根据选定列对报表进行排序。您可以使用下列值测试数据：

manager\_id = 103, 按姓氏排序  
 manager\_id = 201, 按薪金排序  
 manager\_id = 124, 按 employee\_id 排序

```
SELECT employee_id, last_name, salary, department_id
FROM employees
WHERE manager_id = &mgr_num
ORDER BY &order_col;
```

如果您有时间，请完成以下练习：

- 12) 显示姓名中第三个字母为“a”的所有员工的姓氏。

```
SELECT last_name
FROM employees
WHERE last_name LIKE '__a%';
```

- 13) 显示姓氏中有“a”和“e”的所有员工的姓氏。

```
SELECT last_name
FROM employees
WHERE last_name LIKE '%a%'
AND last_name LIKE '%e%';
```

如果您要接受更多的挑战，请完成下面的练习：

- 14) 显示职务为销售代表或仓储职员且薪金不等于\$2,500、\$3,500 或 \$7,000 的所有员工的姓氏、职务和薪金。

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id IN ('SA_REP', 'ST_CLERK')
AND salary NOT IN (2500, 3500, 7000);
```

- 15) 修改 lab\_02\_06.sql，显示佣金百分比为 20% 的所有员工的姓名、薪金和佣金。再次将 lab\_02\_06.sql 保存为 lab\_02\_15.sql。重新运行 lab\_02\_15.sql 中的语句。

```
SELECT last_name "Employee", salary "Monthly Salary",
commission_pct
FROM employees
WHERE commission_pct = .20;
```

## 第 3 课的练习

本练习各部分提供了各种练习，使您有机会使用各种函数来处理字符、数字和日期数据类型。

### 练习 3-1：使用单行函数定制输出

- 1) 编写一个查询来显示系统日期。将该列标记为 Date。

注：如果您的数据库位于远程的另一个时区，则输出日期将是数据库所在操作系统的日期。

Date
1 10-JUN-09

- 2) HR 部门需要一个报表，用于显示每位员工的员工编号、姓氏、薪金和增加 15.5% 之后的薪金（用整数表示）。将该列标记为 New\_Salary。将您的 SQL 语句保存在名为 lab\_03\_02.sql 的文件中。
- 3) 运行 lab\_03\_02.sql 文件中的查询。

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
1	Whalen	4400	5082
2	Hartstein	13000	15015
3	Fay	6000	6930
4	Higgins	12000	13860
5	Gietz	8300	9587

...

19	Taylor	8600	9933
20	Grant	7000	8085

- 4) 修改查询 lab\_03\_02.sql，使其增加一列，用于显示用新的薪金减去原来的薪金之后的结果。将该列标记为 Increase。将该文件的内容另存为 lab\_03\_04.sql。运行修订后的查询。

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
1	Whalen	4400	5082	682
2	Hartstein	13000	15015	2015
3	Fay	6000	6930	930
4	Higgins	12000	13860	1860
5	Gietz	8300	9587	1287

...

19	Taylor	8600	9933	1333
20	Grant	7000	8085	1085

### 练习 3-1：使用单行函数定制输出（续）

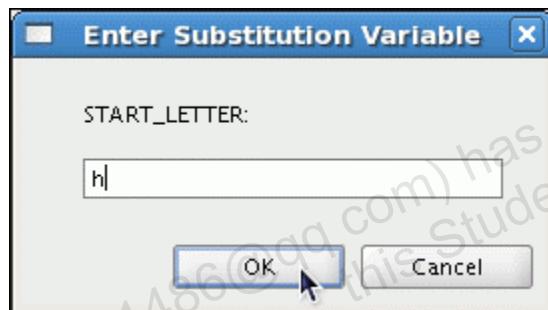
- 5) 编写一个查询，用于显示姓名以字母“J”、“A”或“M”开头的所有员工的姓氏（首字母大写，其它所有字母小写）和姓氏长度。为每列指定一个合适的标签。按照员工的姓氏对结果排序。

	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

重新编写查询，以便提示用户输入姓氏的开头字母。例如，如果用户在看到提示后，输入字母“H”，则输出将显示姓氏以字母“H”开头的所有员工。

	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

对该查询进行修改，以便输入字母的大小写不会影响输出。在使用 SELECT 查询进行处理之前，所输入字母必须为大写。



	Name	Length
1	Hartstein	9
2	Higgins	7
3	Hunold	6

### 练习 3-1：使用单行函数定制输出（续）

- 6) HR 部门希望得到每位员工的聘用持续时间。显示每位员工的姓氏并计算该员工的聘用日期与当前日期之间的月数。将该列标记为 MONTHS\_WORKED。根据聘用的月数对结果排序。将该月数舍入到最接近的整数。

注：因为该查询与执行查询的日期相关，因此 MONTHS\_WORKED 列中的值会发生变化。

	LAST_NAME	MONTHS_WORKED
1	Zlotkey	112
2	Mourgos	115
3	Grant	121
4	Lorentz	124
5	Vargas	131

...

19	Whalen	261
20	King	264

如果您有时间，请完成以下练习：

- 7) 创建一个查询，用于显示所有员工的姓氏和薪金。将薪金的格式设置为 15 个字符长，在左边填充 \$ 符号。将该列标记为 SALARY。

	LAST_NAME	SALARY
1	Whalen	\$\$\$\$\$\$\$\$\$\$\$\$\$\$4400
2	Hartstein	\$\$\$\$\$\$\$\$\$\$\$\$\$\$13000
3	Fay	\$\$\$\$\$\$\$\$\$\$\$\$\$\$6000
4	Higgins	\$\$\$\$\$\$\$\$\$\$\$\$\$\$12000
5	Gietz	\$\$\$\$\$\$\$\$\$\$\$\$\$\$8300

...

19	Taylor	\$\$\$\$\$\$\$\$\$\$\$\$\$\$8600
20	Grant	\$\$\$\$\$\$\$\$\$\$\$\$\$\$7000

### 练习 3-1：使用单行函数定制输出（续）

- 8) 创建一个查询，用于显示员工姓氏的前 8 个字符，并用星号表明他们的薪金额。每个星号代表一千美元。按薪金的降序对数据排序。将该列标记为 EMPLOYEES \_ AND \_ THEIR \_ SALARIES。

EMPLOYEES _ AND _ THEIR _ SALARIES	
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****

...

19	Matos **
20	Vargas **

- 9) 创建一个查询，用于显示部门 90 中所有员工的姓氏和聘用周数。将周数列标记为 TENURE。将周数值截断到 0 位小数位。按员工聘用期长短的降序显示记录。

注：因 TENURE 值与您运行查询的日期相关，因此查询的聘用期会发生变化。

LAST_NAME	TENURE
King	1147
Kochhar	1028
De Haan	856

## 练习解答 3-1：使用单行函数定制输出

- 1) 编写一个查询来显示系统日期。将该列标记为 Date。

注：如果您的数据库位于远程的另一个时区，则输出日期将是数据库所在操作系统的日期。

```
SELECT sysdate "Date"
FROM dual;
```

- 2) HR 部门需要一个报表，用于显示每位员工的员工编号、姓氏、薪金和增加 15.5% 之后的薪金（用整数表示）。将该列标记为 New Salary。将您的 SQL 语句保存在名为 lab\_03\_02.sql 的文件中。

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

- 3) 运行 lab\_03\_02.sql 文件中的查询。

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.155, 0) "New Salary"
FROM employees;
```

- 4) 修改查询 lab\_03\_02.sql，使其增加一列，用于显示用新的薪金减去原来的薪金之后的结果。将该列标记为 Increase。将该文件的内容另存为 lab\_03\_04.sql。运行修订后的查询。

```
SELECT employee_id, last_name, salary,
ROUND(salary * 1.155, 0) "New Salary",
ROUND(salary * 1.155, 0) - salary "Increase"
FROM employees;
```

### 练习解答 3-1：使用单行函数定制输出（续）

- 5) 编写一个查询，用于显示姓名以字母“J”、“A”或“M”开头的所有员工的姓氏（首字母大写，其它所有字母小写）和姓氏长度。为每列指定一个合适的标签。按照员工的姓氏对结果排序。

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE 'J%'
 OR    last_name LIKE 'M%'
 OR    last_name LIKE 'A%'
ORDER BY last_name ;
```

重新编写查询，以便提示用户输入姓氏的开头字母。例如，如果用户在看到提示后，输入字母“H”，则在输出中显示姓氏以字母“H”开头的所有员工。

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE '&start_letter%'
ORDER BY last_name;
```

对该查询进行修改，以便输入字母的大小写不会影响输出。在使用 SELECT 查询进行处理之前，所输入字母必须为大写。

```
SELECT INITCAP(last_name) "Name",
       LENGTH(last_name) "Length"
  FROM employees
 WHERE last_name LIKE UPPER('&start_letter%')
ORDER BY last_name;
```

- 6) HR 部门希望得到每位员工的聘用持续时间。显示每位员工的姓氏并计算该员工的聘用日期与当前日期之间的月数。将该列标记为 MONTHS\_WORKED。根据聘用的月数对结果排序。将该月数舍入到最接近的整数。

**注：**因为该查询与执行查询的日期相关，因此 MONTHS\_WORKED 列中的值会发生变化。

```
SELECT last_name, ROUND(MONTHS_BETWEEN(
      SYSDATE, hire_date)) MONTHS_WORKED
  FROM employees
 ORDER BY months_worked;
```

## 练习解答 3-1：使用单行函数定制输出（续）

如果您有时间，请完成以下练习：

- 7) 创建一个查询，用于显示所有员工的姓氏和薪金。将薪金的格式设置为 15 个字符长，在左边填充 \$ 符号。将该列标记为 SALARY。

```
SELECT last_name,
       LPAD(salary, 15, '$') SALARY
  FROM employees;
```

- 8) 创建一个查询，用于显示员工姓氏的前 8 个字符，并用星号表明他们的薪金额。每个星号代表一千美元。按薪金的降序对数据排序。将该列标记为 EMPLOYEES\_AND\_THEIR\_SALARIES。

```
SELECT rpad(last_name, 8) || ' ' ||
       rpad(' ', salary/1000+1, '*')
          EMPLOYEES_AND_THEIR_SALARIES
     FROM employees
    ORDER BY salary DESC;
```

- 9) 创建一个查询，用于显示部门 90 中所有员工的姓氏和聘用周数。将周数列标记为 TENURE。将周数值截断到 0 位小数位。按员工聘用期长短的降序显示记录。

注：因 TENURE 值与运行查询的日期有关，因此聘用期会发生变化。

```
SELECT last_name, trunc((SYSDATE-hire_date)/7) AS TENURE
  FROM employees
 WHERE department_id = 90
 ORDER BY TENURE DESC;
```

## 第 4 课的练习

本练习提供使用 TO\_CHAR 和 TO\_DATE 函数以及条件表达式（如 DECODE 和 CASE）的各种练习。请记住，对于嵌套函数，将从最内层函数到最外层函数计算结果。

## 练习 4-1：使用转换函数和条件表达式

- 1) 创建一个报表来生成每位员工的以下内容：

<employee last name> 的月薪为 <salary>, 但是要求 <3 times salary.>。将该列标记为 Dream Salaries。

	Dream Salaries
1	Whalen earns \$4,400.00 monthly but wants \$13,200.00.
2	Hartstein earns \$13,000.00 monthly but wants \$39,000.00.
3	Fay earns \$6,000.00 monthly but wants \$18,000.00.
4	Higgins earns \$12,000.00 monthly but wants \$36,000.00.
5	Gietz earns \$8,300.00 monthly but wants \$24,900.00.

...

19	Taylor earns \$8,600.00 monthly but wants \$25,800.00.
20	Grant earns \$7,000.00 monthly but wants \$21,000.00.

- 2) 显示每位员工的姓氏、聘用日期和薪金复核日期，其中薪金复核日期是服务 6 个月之后的第一个星期一。将该列标记为 REVIEW。设置日期的格式，使其显示格式类似于“Monday, the Thirty-First of July, 2000”。

	LAST_NAME	HIRE_DATE	REVIEW
1	Whalen	17-SEP-87	Monday, the Twenty-First of March, 1988
2	Hartstein	17-FEB-96	Monday, the Nineteenth of August, 1996
3	Fay	17-AUC-97	Monday, the Twenty-Third of February, 1998
4	Higgins	07-JUN-94	Monday, the Twelfth of December, 1994
5	Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

...

19	Taylor	24-MAR-98	Monday, the Twenty-Eighth of September, 1998
20	Grant	24-MAY-99	Monday, the Twenty-Ninth of November, 1999

- 3) 显示员工的姓氏、聘用日期和该员工是在星期几开始工作的。将该列标记为 DAY。按一星期中各日的顺序（从星期一开始）对结果排序。

	LAST_NAME	HIRE_DATE	DAY
1	Grant	24-MAY-99	MONDAY
2	Ernst	21-MAY-91	TUESDAY
3	Taylor	24-MAR-98	TUESDAY
4	Rajs	17-OCT-95	TUESDAY
5	Mourgos	16-NOV-99	TUESDAY

...

19	Matos	15-MAR-98	SUNDAY
20	Fay	17-AUG-97	SUNDAY

## 练习 4-1：使用转换函数和条件表达式（续）

- 4) 创建一个查询，用于显示员工的姓氏和佣金额。如果某位员工没有佣金，则显示“No Commission”。将该列标记为 COMM。

	LAST_NAME	COMM
1	Whalen	No Commission
2	Hartstein	No Commission
3	Fay	No Commission
4	Higgins	No Commission
5	Gietz	No Commission

...

16	Vargas	No Commission
17	Zlotkey	.2
18	Abel	.3
19	Taylor	.2
20	Grant	.15

如果您有时间，请完成以下练习：

- 5) 使用 DECODE 函数编写一个查询，以便使用以下数据根据 JOB\_ID 列的值显示所有员工的等级：

作业	等级
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA REP	D
ST_CLERK	E
上述各项都不是	0

JOB_ID	GRADE
1 AC_ACCOUNT	0
2 AC_MGR	0
3 AD_ASST	0
4 AD_PRES	A
5 AD_VP	0
6 AD_VP	0
7 IT_PROG	C

...

14	SA REP	D
15	SA REP	D

...

19	ST_CLERK	E
20	ST_MAN	B

**练习 4-1：使用转换函数和条件表达式（续）**

- 6) 使用 CASE 语法重新编写上一个练习中的语句。

JOB_ID	GRADE
1 AC_ACCOUNT	O
2 AC_MGR	O
3 AD_ASST	O
4 AD_PRES	A
5 AD_VP	O
6 AD_VP	O
7 IT_PROG	C
...	
14 SA_REP	D
15 SA_REP	D
...	
19 ST_CLERK	E
20 ST_MAN	B

## 练习解答 4-1：使用转换函数和条件表达式

- 1) 创建一个报表来生成每位员工的以下内容：

*<employee last name>* 的月薪为 *<salary>*，但是要求 *<3 times salary.>*。将该列标记为 Dream Salaries。

```
SELECT last_name || ' earns '
  || TO_CHAR(salary, 'fm$99,999.00')
  || ' monthly but wants '
  || TO_CHAR(salary * 3, 'fm$99,999.00')
  || '.' "Dream Salaries"
FROM employees;
```

- 2) 显示每位员工的姓氏、聘用日期和薪金复核日期，其中薪金复核日期是服务 6 个月之后的第一个星期一。将该列标记为 REVIEW。设置日期的格式，使其显示格式类似于“Monday, the Thirty-First of July, 2000”。

```
SELECT last_name, hire_date,
       TO_CHAR(NEXT_DAY(ADD_MONTHS(hire_date, 6), 'MONDAY'),
               'fmDay, "the" Ddspth "of" Month, YYYY') REVIEW
  FROM employees;
```

- 3) 显示员工的姓氏、聘用日期和该员工是在星期几开始工作的。将该列标记为 DAY。按一星期中各日的顺序（从星期一开始）对结果排序。

```
SELECT last_name, hire_date,
       TO_CHAR(hire_date, 'DAY') DAY
  FROM employees
 ORDER BY TO_CHAR(hire_date - 1, 'd');
```

- 4) 创建一个查询，用于显示员工的姓氏和佣金额。如果某位员工没有佣金，则显示“No Commission”。将该列标记为 COMM。

```
SELECT last_name,
       NVL(TO_CHAR(commission_pct), 'No Commission') COMM
  FROM employees;
```

## 练习解答 4-1：使用转换函数和条件表达式（续）

- 5) 使用 DECODE 函数编写一个查询，用于根据 JOB\_ID 列的值（即使用以下数据）显示所有员工的等级：

作业	等级
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA REP	D
ST_CLERK	E
上述各项都不是	0

```
SELECT job_id, decode (job_id,
                      'ST_CLERK',      'E',
                      'SA REP',        'D',
                      'IT_PROG',       'C',
                      'ST_MAN',         'B',
                      'AD_PRES',       'A',
                      '0') GRADE
FROM employees;
```

- 6) 使用 CASE 语法重新编写上一个练习中的语句。

```
SELECT job_id, CASE job_id
    WHEN 'ST_CLERK' THEN 'E'
    WHEN 'SA REP' THEN 'D'
    WHEN 'IT_PROG' THEN 'C'
    WHEN 'ST_MAN' THEN 'B'
    WHEN 'AD_PRES' THEN 'A'
    ELSE '0' END GRADE
FROM employees;
```

## 第 5 课的练习

在本练习结束时，您应该已熟悉如何使用组函数以及如何选择数据组。

## 练习 5-1：使用组函数报告聚集数据

判断下面三句话是否正确：选择“对”或“错”。

- 1) 组函数通过对多行进行处理为每个组生成一个结果。

对/错

- 2) 组函数在计算中可以包括空值。

对/错

- 3) WHERE 子句用于对要在分组计算中包含的行进行限定。

对/错

HR 部门需要以下报表：

- 4) 查找所有员工的最高薪金、最低薪金、薪金总额以及平均薪金。将这些列分别标记为 Maximum、Minimum、Sum 和 Average。将结果舍入到最接近的整数。将 SQL 语句另存为 lab\_05\_04.sql。运行该查询。

	Maximum	Minimum	Sum	Average
1	24000	2500	175500	8775

- 5) 修改 lab\_05\_04.sql 中的查询，显示每个职务类型的最低薪金、最高薪金、薪金总额以及平均薪金。再次将 lab\_05\_04.sql 保存为 lab\_05\_05.sql。运行 lab\_05\_05.sql 中的语句。

	JOB_ID	Maximum	Minimum	Sum	Average
1	AC_MGR	12000	12000	12000	12000
2	AC_ACCOUNT	8300	8300	8300	8300
3	IT_PROG	9000	4200	19200	6400
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	MK_MAN	13000	13000	13000	13000
8	SA_MAN	10500	10500	10500	10500
9	MK_REP	6000	6000	6000	6000
10	AD_PRES	24000	24000	24000	24000
11	SA_REP	11000	7000	26600	8867
12	ST_CLERK	3500	2500	11700	2925

## 练习 5-1：使用组函数报告聚集数据（续）

- 6) 编写一个查询，用于显示担任相同职务的人员的数量。

JOB_ID	COUNT(*)
1 AC_ACCOUNT	1
2 AC_MGR	1
3 AD_ASST	1
4 AD_PRES	1
5 AD_VP	2
6 IT_PROG	3
7 MK_MAN	1
8 MK_REP	1
9 SA_MAN	1
10 SA_REP	3
11 ST_CLERK	4
12 ST_MAN	1

使该查询变成通用查询，提示 HR 部门中的用户输入一个职务。将该脚本保存到名为 lab\_05\_06.sql 的文件中。运行该查询。出现提示时输入 IT\_PROG。

JOB_ID	COUNT(*)
1 IT_PROG	3

- 7) 确定经理的人数但不列出经理。标记列 Number of Managers。

**提示：** 使用 MANAGER\_ID 列确定经理的人数。

Number of Managers
1 8

- 8) 找出最高与最低薪金之间的差额。将该列标记为 DIFFERENCE。

DIFFERENCE
1 21500

如果您有时间，请完成以下练习：

- 9) 创建一个报表，用于显示经理编号及该经理属下员工的最低薪金。将经理不知道的任何员工排除在外。排除最低薪金不超过 \$6,000 的所有组。按薪金降序对输出进行排序。

MANAGER_ID	MIN(SALARY)
1 102	9000
2 205	8300
3 149	7000

## 练习 5-1：使用组函数报告聚集数据（续）

如果您要接受更多的挑战，请完成下面的练习：

- 10) 创建一个查询，用于显示员工总数以及其中在 1995、1996、1997 和 1998 年聘用的员工数。请创建相应的列标题。

	TOTAL	1995	1996	1997	1998
1	20	1	2	2	3

- 11) 创建一个矩阵查询，以显示职务、基于部门编号的该职务的薪金以及部门 20、50、80 和 90 中该职务的薪金总额，并为每一列指定相应的标题。

	Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
1	AC_MGR	(null)	(null)	(null)	(null)	12000
2	AC_ACCOUNT	(null)	(null)	(null)	(null)	8300
3	IT_PROG	(null)	(null)	(null)	(null)	19200
4	ST_MAN	(null)	5800	(null)	(null)	5800
5	AD_ASST	(null)	(null)	(null)	(null)	4400
6	AD_VP	(null)	(null)	(null)	34000	34000
7	MK_MAN	13000	(null)	(null)	(null)	13000
8	SA_MAN	(null)	(null)	10500	(null)	10500
9	MK_REP	6000	(null)	(null)	(null)	6000
10	AD_PRES	(null)	(null)	(null)	24000	24000
11	SA_REP	(null)	(null)	19600	(null)	26600
12	ST_CLERK	(null)	11700	(null)	(null)	11700

## 练习解答 5-1：使用组函数报告聚集数据

判断下面三句话是否正确：选择“对”或“错”。

- 1) 组函数通过对多行进行处理为每个组生成一个结果。

对/错

- 2) 组函数在计算中可以包括空值。

对/错

- 3) WHERE 子句用于对要在分组计算中包含的行进行限定。

对/错

HR 部门需要以下报表：

- 4) 查找所有员工的最高薪金、最低薪金、薪金总额以及平均薪金。将这些列分别标记为 Maximum、Minimum、Sum 和 Average。将结果舍入到最接近的整数。将 SQL 语句另存为 lab\_05\_04.sql。运行该查询。

```
SELECT ROUND(MAX(salary),0) "Maximum",
       ROUND(MIN(salary),0) "Minimum",
       ROUND(SUM(salary),0) "Sum",
       ROUND(AVG(salary),0) "Average"
  FROM employees;
```

- 5) 修改 lab\_05\_04.sql 中的查询，显示每个职务类型的最低薪金、最高薪金、薪金总额以及平均薪金。再次将 lab\_05\_04.sql 保存为 lab\_05\_05.sql。运行 lab\_05\_05.sql 中的语句。

```
SELECT job_id, ROUND(MAX(salary),0) "Maximum",
       ROUND(MIN(salary),0) "Minimum",
       ROUND(SUM(salary),0) "Sum",
       ROUND(AVG(salary),0) "Average"
  FROM employees
 GROUP BY job_id;
```

- 6) 编写一个查询，用于显示担任相同职务的人员的数量。

```
SELECT job_id, COUNT(*)
  FROM employees
 GROUP BY job_id;
```

使该查询变成通用查询，提示 HR 部门中的用户输入一个职务。将该脚本保存到名为 lab\_05\_06.sql 的文件中。运行该查询。提示时输入 IT\_PROG，然后单击“OK（确定）”。

```
SELECT job_id, COUNT(*)
  FROM employees
 WHERE job_id = '&job_title'
 GROUP BY job_id;
```

## 练习解答 5-1：使用组函数报告聚集数据（续）

- 7) 确定经理的人数但不列出经理。标记列 Number of Managers。

**提示：** 使用 MANAGER\_ID 列确定经理的人数。

```
SELECT COUNT(DISTINCT manager_id) "Number of Managers"
FROM employees;
```

- 8) 找出最高与最低薪金之间的差额。将该列标记为 DIFFERENCE。

```
SELECT MAX(salary) - MIN(salary) DIFFERENCE
FROM employees;
```

如果您有时间，请完成以下练习：

- 9) 创建一个报表，用于显示经理编号及该经理属下员工的最低薪金。将经理不知道的任何员工排除在外。排除最低薪金不超过 \$6,000 的所有组。按薪金降序对输出进行排序。

```
SELECT manager_id, MIN(salary)
FROM employees
WHERE manager_id IS NOT NULL
GROUP BY manager_id
HAVING MIN(salary) > 6000
ORDER BY MIN(salary) DESC;
```

如果您要接受更多的挑战，请完成下面的练习：

- 10) 创建一个查询，用于显示员工总数以及在 1995、1996、1997 和 1998 年聘用员工的总数。创建合适的列标题。

```
SELECT COUNT(*) total,
SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1995, 1, 0)) "1995",
SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1996, 1, 0)) "1996",
SUM(DECODE(TO_CHAR(hire_date,
'YYYY'), 1997, 1, 0)) "1997",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'), 1998, 1, 0)) "1998"
FROM employees;
```

- 11) 创建一个矩阵查询，显示职务、基于部门编号的该职务的薪金以及部门 20、50、80 和 90 中该职务的薪金总额，并为每一列指定相应的标题。

```
SELECT job_id "Job",
SUM(DECODE(department_id, 20, salary)) "Dept 20",
SUM(DECODE(department_id, 50, salary)) "Dept 50",
SUM(DECODE(department_id, 80, salary)) "Dept 80",
SUM(DECODE(department_id, 90, salary)) "Dept 90",
SUM(salary) "Total"
FROM employees
GROUP BY job_id;
```

## 第 6 课的练习

本练习旨在让您熟悉使用符合 SQL:1999 的联接从多个表中提取数据的过程。

## 练习 6-1：使用联接显示多个表中的数据

- 1) 为 HR 部门编写一个查询来生成所有部门的地址。请使用 LOCATIONS 和 COUNTRIES 表。在输出中显示位置 ID、街道地址、城市、州或省以及国家/地区。使用 NATURAL JOIN 生成结果。

	LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400	2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500	2011 Interior Blvd	South San Francisco	California	United States of America
3	1700	2004 Charade Rd	Seattle	Washington	United States of America
4	1800	460 Bloor St. W.	Toronto	Ontario	Canada
5	2500	Magdalene Centre, The Oxford Science Park	Oxford		United Kingdom

- 2) HR 部门需要一个其中只提供相应部门的员工的报表。编写一个查询，显示所有员工的姓氏、部门编号和部门名称。

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping

...

18 Higgins	110 Accounting
19 Gietz	110 Accounting

- 3) HR 部门需要一个在多伦多工作的员工的报表，显示在多伦多工作的所有员工的姓氏、职务、部门编号和部门名称。

	LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	Hartstein	MK_MAN	20	Marketing
2	Fay	MK_REP	20	Marketing

## 练习 6-1：使用联接显示多个表中的数据（续）

- 4) 创建一个报表，用于显示员工的姓氏和员工编号及其经理的姓氏和经理编号。将这些列分别标记为 Employee、EMP#、Manager 和 Mgr#。将 SQL 语句另存为 lab\_06\_04.sql。运行该查询。

Employee	EMP#	Manager	Mgr#
1 Hunold	103 De Haan	102	
2 Fay	202 Hartstein	201	
3 Gietz	206 Higgins	205	
4 Lorentz	107 Hunold	103	
5 Ernst	104 Hunold	103	

...

18 Taylor	176 Zlotkey	149
19 Abel	174 Zlotkey	149

- 5) 修改 lab\_06\_04.sql，用于显示所有员工，包括没有经理的员工 King。按员工编号对结果进行排序。将 SQL 语句另存为 lab\_06\_05.sql。运行 lab\_06\_05.sql 中的查询。

Employee	EMP#	Manager	Mgr#
1 King	100 (null)	(null)	
2 Kochhar	101 King	100	
3 De Haan	102 King	100	
4 Hunold	103 De Haan	102	
5 Ernst	104 Hunold	103	

...

19 Higgins	205 Kochhar	101
20 Gietz	206 Higgins	205

- 6) 为 HR 部门创建一个报表，用于显示员工的姓氏、部门编号以及与该员工在同一部门中工作的所有员工。为每个列指定一个合适的标签。将脚本保存到名为 lab\_06\_06.sql 的文件中。

DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20 Fay	Hartstein
2	20 Hartstein	Fay
3	50 Davies	Matos
4	50 Davies	Mourgos
5	50 Davies	Rajs

...

41	110 Gietz	Higgins
42	110 Higgins	Gietz

## 练习 6-1：使用联接显示多个表中的数据（续）

- 7) HR 部门需要一个关于职务等级和薪金的报表。为了熟悉 JOB\_GRADES 表，应先显示 JOB\_GRADES 表的结构。然后创建一个查询，用于显示所有员工的姓名、职务、部门名称、薪金和等级。

```
DESC JOB_GRADES
Name          Null    Type
-----
GRADE_LEVEL      VARCHAR2(3)
LOWEST_SAL      NUMBER
HIGHEST_SAL     NUMBER

3 rows selected
```

	LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1	King	AD_PRES	Executive	24000	E
2	Kochhar	AD_VP	Executive	17000	E
3	De Haan	AD_VP	Executive	17000	E
4	Hartstein	MK_MAN	Marketing	13000	D
5	Higgins	AC_MGR	Accounting	12000	D

...

18	Matos	ST_CLERK	Shipping	2600	A
19	Vargas	ST_CLERK	Shipping	2500	A

如果您要接受更多的挑战，请完成下面的练习：

- 8) HR 部门希望确定在 Davies 之后聘用的所有员工的姓名。创建一个查询，用于显示在员工 Davies 之后聘用的任何员工的姓名和聘用日期。

	LAST_NAME	HIRE_DATE
1	Fay	17-AUG-97
2	Lorentz	07-FEB-99
3	Mourgos	16-NOV-99
4	Matos	15-MAR-98
5	Vargas	09-JUL-98
6	Zlotkey	29-JAN-00
7	Taylor	24-MAR-98
8	Grant	24-MAY-99

**练习 6-1：使用联接显示多个表中的数据（续）**

- 9) HR 部门需要查找在聘用其经理之前已聘用的所有员工的姓名和聘用日期，及其经理的姓名和聘用日期。将脚本保存到名为 lab\_06\_09.sql 的文件中。

	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

## 练习解答 6-1：使用联接显示多个表中的数据

- 1) 为 HR 部门编写一个查询来生成所有部门的地址。请使用 LOCATIONS 和 COUNTRIES 表。在输出中显示位置 ID、街道地址、城市、州或省以及国家/地区。使用 NATURAL JOIN 生成结果。

```
SELECT location_id, street_address, city, state_province,
       country_name
  FROM   locations
NATURAL JOIN countries;
```

- 2) HR 部门需要一个包含所有员工的报表。编写一个查询，用于显示所有员工的姓氏、部门编号和部门名称。

```
SELECT last_name, department_id, department_name
  FROM   employees
  JOIN   departments
  USING (department_id);
```

- 3) HR 部门需要一个在多伦多工作的员工的报表，显示在多伦多工作的所有员工的姓氏、职务、部门编号和部门名称。

```
SELECT e.last_name, e.job_id, e.department_id,
       d.department_name
  FROM   employees e JOIN departments d
  ON     (e.department_id = d.department_id)
  JOIN   locations l
  ON     (d.location_id = l.location_id)
 WHERE  LOWER(l.city) = 'toronto';
```

- 4) 创建一个报表，用于显示员工的姓氏和员工编号及其经理的姓氏和经理编号。将这些列分别标记为 Employee、Emp#、Manager 和 Mgr#。将 SQL 语句另存为 lab\_06\_04.sql。运行该查询。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w join employees m
  ON     (w.manager_id = m.employee_id);
```

- 5) 修改 lab\_06\_04.sql，用于显示所有员工，包括没有经理的员工 King。按员工编号对结果进行排序。将 SQL 语句另存为 lab\_06\_05.sql。运行 lab\_06\_05.sql 中的查询。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w
 LEFT   OUTER JOIN employees m
  ON     (w.manager_id = m.employee_id)
 ORDER BY 2;
```

## 练习解答 6-1：使用联接显示多个表中的数据（续）

- 6) 为 HR 部门创建一个报表，用于显示员工的姓氏、部门编号以及与该员工在同一部门中工作的所有员工。为每个列指定一个合适的标签。将脚本保存到名为 lab\_06\_06.sql 的文件中。运行该查询。

```
SELECT e.department_id department, e.last_name employee,
       c.last_name colleague
  FROM employees e JOIN employees c
 WHERE (e.department_id = c.department_id)
 ORDER BY e.department_id, e.last_name, c.last_name;
```

- 7) HR 部门需要一个关于职务等级和薪金的报表。为了熟悉 JOB\_GRADES 表，应先显示 JOB\_GRADES 表的结构。然后创建一个查询，用于显示所有员工的姓名、职务、部门名称、薪金和等级。

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
  FROM employees e JOIN departments d
 WHERE (e.department_id = d.department_id)
 JOIN job_grades j
 WHERE (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

如果您要接受更多的挑战，请完成下面的练习：

- 8) HR 部门希望确定在 Davies 之后聘用的所有员工的姓名。创建一个查询，用于显示在员工 Davies 之后聘用的任何员工的姓名和聘用日期。

```
SELECT e.last_name, e.hire_date
  FROM employees e JOIN employees davies
 WHERE (davies.last_name = 'Davies')
 ORDER BY davies.hire_date < e.hire_date;
```

- 9) HR 部门需要查找在其经理之前聘用的所有员工的姓名和聘用日期，及其经理的姓名和聘用日期。将脚本保存到名为 lab\_06\_09.sql 的文件中。

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
  FROM employees w JOIN employees m
 WHERE (w.manager_id = m.employee_id)
 ORDER BY w.hire_date < m.hire_date;
```

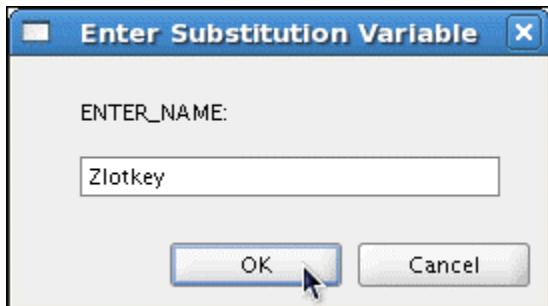
## 第 7 课的练习

在此练习中，您将使用嵌套的 SELECT 语句编写复杂的查询。

可能需要针对练习中的问题先创建内部查询。在编写外部查询代码之前，应先确保内部查询可以运行，并且可以生成所需的数据。

## 练习 7-1：使用子查询来解决查询

- 1) HR 部门需要一个查询，用于提示用户输入员工的姓氏。该查询将显示与提供姓名的员工在同一部门工作的所有员工（不包括此员工）的姓氏和聘用日期。例如，如果用户输入 Zlotkey，则会查找与 Zlotkey 一起工作的所有员工（不包括 Zlotkey）。



	LAST_NAME	HIRE_DATE
1	Abel	11-MAY-96
2	Taylor	24-MAR-98

- 2) 创建一个报表，用于显示薪金高于平均薪金的所有员工的员工编号、姓氏和薪金。按薪金升序对结果排序。

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000
2	149	Zlotkey	10500
3	174	Abel	11000
4	205	Higgins	12000
5	201	Hartstein	13000
6	102	De Haan	17000
7	101	Kochhar	17000
8	100	King	24000

- 3) 编写一个查询，用于显示满足以下条件的所有员工的员工编号和姓氏：其所在部门的任意一位员工的姓氏中包含字母“u”。将 SQL 语句另存为 lab\_07\_03.sql。运行您的查询。

	EMPLOYEE_ID	LAST_NAME
1	124	Mourgos
2	141	Rajs
3	142	Davies
4	143	Matos
5	144	Vargas
6	103	Hunold
7	104	Ernst
8	107	Lorentz

## 练习 7-1：使用子查询来解决查询（续）

- 4) HR 部门需要一个报表，用于显示部门位置 ID 为 1700 的所有员工的姓氏、部门编号和职务 ID。

	LAST_NAME	DEPARTMENT_ID	JOB_ID
1	Whalen	10	AD_ASST
2	King	90	AD_PRES
3	Kochhar	90	AD_VP
4	De Haan	90	AD_VP
5	Higgins	110	AC_MGR
6	Gietz	110	AC_ACCOUNT

修改此查询，以便提示用户输入位置 ID。将此查询保存到名为 lab\_07\_04.sql 的文件中。

- 5) 为 HR 创建一个报表，用于显示属于 King 的直属下级的每位员工的姓氏和薪金。

	LAST_NAME	SALARY
1	Hartstein	13000
2	Kochhar	17000
3	De Haan	17000
4	Mourgos	5800
5	Zlotkey	10500

- 6) 为 HR 创建一个报表，用于显示行政部门中每位员工的部门编号、姓氏和职务 ID。

	DEPARTMENT_ID	LAST_NAME	JOB_ID
1	90	King	AD_PRES
2	90	Kochhar	AD_VP
3	90	De Haan	AD_VP

- 7) 创建一个报表，用于列出其薪金高于部门 60 中任意一名员工薪水的所有员工。

如果您有时间，请完成以下练习：

- 8) 通过修改 lab\_07\_03.sql 中的查询来显示满足以下条件的所有员工的员工编号、姓氏和薪金：其薪金高于平均薪金且所在部门的任意一位员工的姓氏中包含“u”。再次将 lab\_07\_03.sql 保存为 lab\_07\_08.sql。运行 lab\_07\_08.sql 中的语句。

	EMPLOYEE_ID	LAST_NAME	SALARY
1	103	Hunold	9000

## 练习解答 7-1：使用子查询来解决查询

- 1) HR 部门需要一个查询，用于提示用户输入员工的姓氏。该查询将显示与提供姓名的员工在同一部门工作的所有员工（不包括此员工）的姓氏和聘用日期。例如，如果用户输入 Zlotkey，则会查找与 Zlotkey 一起工作的所有员工（不包括 Zlotkey）。

```
UNDEFINE Enter_name

SELECT last_name, hire_date
FROM employees
WHERE department_id = (SELECT department_id
                        FROM employees
                        WHERE last_name = '&&Enter_name')
AND last_name <> '&Enter_name';
```

- 2) 创建一个报表，用于显示薪金高于平均薪金的所有员工的员工编号、姓氏和薪金。按薪金升序对结果排序。

```
SELECT employee_id, last_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary)
                  FROM employees)
ORDER BY salary;
```

- 3) 编写一个查询，用于显示满足以下条件的所有员工的员工编号和姓氏：其所在部门的任意一名员工的姓氏里包含“u”。将您的 SQL 语句另存为 lab\_07\_03.sql。运行您的查询。

```
SELECT employee_id, last_name
FROM employees
WHERE department_id IN (SELECT department_id
                        FROM employees
                        WHERE last_name like '%u%');
```

- 4) HR 部门需要一个报表，用于显示部门位置 ID 为 1700 的所有员工的姓氏、部门编号和职务 ID。

```
SELECT last_name, department_id, job_id
FROM employees
WHERE department_id IN (SELECT department_id
                        FROM departments
                        WHERE location_id = 1700);
```

## 练习解答 7-1：使用子查询来解决查询（续）

修改此查询，以便提示用户输入位置 ID。将此查询保存到名为 lab\_07\_04.sql 的文件中。

```
SELECT last_name, department_id, job_id
FROM employees
WHERE department_id IN (SELECT department_id
                           FROM departments
                           WHERE location_id =
&Enter_location);
```

- 5) 为 HR 创建一个报表，用于显示属于 King 的直属下级的每位员工的姓氏和薪金。

```
SELECT last_name, salary
FROM employees
WHERE manager_id = (SELECT employee_id
                      FROM employees
                      WHERE last_name = 'King');
```

- 6) 为 HR 创建一个报表，用于显示行政部门中每位员工的部门编号、姓氏和职务 ID。

```
SELECT department_id, last_name, job_id
FROM employees
WHERE department_id IN (SELECT department_id
                           FROM departments
                           WHERE department_name =
'Executive');
```

- 7) 创建一个报表，用于列出其薪金高于部门 60 中任意一名员工薪水的所有员工。

```
SELECT last_name FROM employees
WHERE salary > ANY (SELECT salary
                      FROM employees
                      WHERE department_id=60);
```

如果您有时间，请完成以下练习：

- 8) 通过修改 lab\_07\_03.sql 中的查询来显示满足以下条件的所有员工的员工编号、姓氏和薪金：其薪金高于平均薪金且所在部门的任意一位员工的姓氏中包含“u”。再次将 lab\_07\_03.sql 保存到 lab\_07\_08.sql。运行 lab\_07\_08.sql 中的语句。

```
SELECT employee_id, last_name, salary
FROM employees
WHERE department_id IN (SELECT department_id
                           FROM employees
                           WHERE last_name like '%u%')
AND salary > (SELECT AVG(salary)
                  FROM employees);
```

## 第 8 课的练习

在本练习中，您将使用集合运算符编写查询。

## 练习 8-1：使用集合运算符

- 1) HR 部门需要不包含职务 ID ST\_CLERK 的各个部门的部门 ID 列表。使用集合运算符可创建此报表。

	DEPARTMENT_ID
1	10
2	20
3	60
4	80
5	90
6	110
7	190

- 2) HR 部门需要一个尚未设立任何部门的国家/地区列表。显示国家/地区 ID 和国家/地区名称。使用集合运算符可创建此报表。

COUNTRY_ID	COUNTRY_NAME
DE	Germany

- 3) 以部门 10、50 和 20 为顺序为这些部门生成一个职务列表。使用集合运算符显示职务 ID 和部门 ID。

JOB_ID	DEPARTMENT_ID
AD_ASST	10
ST_MAN	50
ST_CLERK	50
MK_MAN	20
MK_REP	20

- 4) 创建一个报表，列出符合以下条件的员工的员工 ID 和职务 ID：这些员工的当前职务与公司最初聘用他们时的职务相同，就是说这些员工曾担任过别的职务，但现在又再次担任最初的职务。

EMPLOYEE_ID	JOB_ID
1	176 SA_REP
2	200 AD_ASST

## 练习 8-1：使用集合运算符（续）

5) HR 部门需要一个包含以下内容的报表：

- EMPLOYEES 表中所有员工的姓氏和部门 ID，不管这些员工是否属于某个部门
- DEPARTMENTS 表中所有部门的部门 ID 和部门名称，不管这些部门中是否有在职员工

编写一个复合查询可完成此任务。

	LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
1	Abel		80 (null)
2	Davies		50 (null)
3	De Haan		90 (null)
4	Ernst		60 (null)
5	Fay		20 (null)
6	Gietz		110 (null)
7	Grant		(null) (null)
8	Hartstein		20 (null)
9	Higgins		110 (null)
10	Hunold		60 (null)
11	King		90 (null)
12	Kochhar		90 (null)
13	Lorentz		60 (null)
14	Matos		50 (null)
15	Mourgos		50 (null)
16	Rajs		50 (null)
17	Taylor		80 (null)
18	Vargas		50 (null)
19	Whalen		10 (null)
20	Zlotkey		80 (null)
21	(null)		10 Administration
22	(null)		20 Marketing
23	(null)		50 Shipping
24	(null)		60 IT
25	(null)		80 Sales
26	(null)		90 Executive
27	(null)		110 Accounting
28	(null)		190 Contracting

## 练习解答 8-1：使用集合运算符

- 1) HR 部门需要不包含职务 ID ST\_CLERK 的各个部门的部门 ID 列表。使用集合运算符可创建此报表。

```
SELECT department_id
FROM departments
MINUS
SELECT department_id
FROM employees
WHERE job_id = 'ST_CLERK';
```

- 2) HR 部门需要一个尚未设立任何部门的国家/地区列表。显示国家/地区 ID 和国家/地区名称。使用集合运算符可创建此报表。

```
SELECT country_id, country_name
FROM countries
MINUS
SELECT l.country_id, c.country_name
FROM locations l JOIN countries c
ON (l.country_id = c.country_id)
JOIN departments d
ON d.location_id=l.location_id;
```

- 3) 以部门 10、50 和 20 为顺序为这些部门生成一个职务列表。使用集合运算符显示职务 ID 和部门 ID。

```
SELECT distinct job_id, department_id
FROM employees
WHERE department_id = 10
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 50
UNION ALL
SELECT DISTINCT job_id, department_id
FROM employees
WHERE department_id = 20
```

- 4) 创建一个报表，列出符合以下条件的员工的员工 ID 和职务 ID：这些员工的当前职务与公司最初聘用他们时的职务相同，就是说这些员工曾担任过别的职务，但现在又再次担任最初的职务。

```
SELECT employee_id, job_id
FROM employees
INTERSECT
SELECT employee_id, job_id
FROM job_history;
```

## 练习解答 8-1：使用集合运算符（续）

5) HR 部门需要一个包含以下内容的报表：

- EMPLOYEES 表中所有员工的姓氏和部门 ID，不管这些员工是否属于某个部门
- DEPARTMENTS 表中所有部门的部门 ID 和部门名称，不管这些部门中是否包含正在工作的员工

编写一个复合查询可完成此任务。

```
SELECT last_name, department_id, TO_CHAR(null)
FROM   employees
UNION
SELECT TO_CHAR(null), department_id, department_name
FROM   departments;
```

## 第 9 课的练习

在本练习中，您将向 MY\_EMPLOYEE 表中添加行，更新和删除该表中的数据，以及控制您的事务处理。运行脚本来创建 MY\_EMPLOYEE 表。

## 练习 9-1：处理数据

HR 部门要求您创建一些 SQL 语句来插入、更新和删除员工数据。将这些语句提供给 HR 部门之前，可以将 MY\_EMPLOYEE 表用作原型。

**注：**对于所有 DML 语句，均可使用“Run Script（运行脚本）”图标（或按 [F5]）执行查询。您可以在“Script Output（脚本输出）”选项卡页上查看反馈消息。对于 SELECT 查询，可以继续使用“Execute Statement（执行语句）”图标或按 [F9]，在“Results（结果）”选项卡页上获得格式化输出。

将数据插入到 MY\_EMPLOYEE 表中。

- 1) 运行 lab\_09\_01.sql 脚本中的语句，以构建在本练习中使用的 MY\_EMPLOYEE 表。
- 2) 描述 MY\_EMPLOYEE 表的结构以确定列名。

DESCRIBE my_employee		
Name	Null	Type
ID	NOT NULL	NUMBER(4)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
USERID		VARCHAR2(8)
SALARY		NUMBER(9,2)
5 rows selected		

- 3) 创建一个 INSERT 语句，将下面示例数据中的第一行数据添加到 MY\_EMPLOYEE 表中。不要在 INSERT 子句中列出这些列。也不要输入任何行。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

## 练习 9-1：处理数据（续）

- 4) 使用上面列表中的第二行示例数据填充 MY\_EMPLOYEE 表。这次请在 INSERT 子句中显式列出这些列。
- 5) 确认添加到表中的内容。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

- 6) 在动态可重用脚本文件中编写一条 INSERT 语句，以便将剩余行加载到 MY\_EMPLOYEE 表中。该脚本应提示输入所有列 (ID、LAST\_NAME、FIRST\_NAME、USERID 和 SALARY)。将此脚本保存到 lab\_09\_06.sql 文件。
- 7) 运行您所创建的脚本中的 INSERT 语句，使用步骤 3 列出的示例数据中接下来的两行填充表。
- 8) 确认添加到表中的内容。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

- 9) 使数据添加变成永久性添加。

更新和删除 MY\_EMPLOYEE 表中的数据。

- 10) 将员工 3 的姓氏更改为 Drexler。
- 11) 将薪金低于 \$900 的所有员工的薪金更改为 \$1,000。
- 12) 验证对表进行的更改。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Dancs	Betty	bdancs	1000
3	Drexler	Ben	bbiri	1100
4	Newman	Chad	cnewman	1000

- 13) 从 MY\_EMPLOYEE 表中删除 Betty Dancs。

## 练习 9-1：处理数据（续）

14) 确认对表所做的更改。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000

15) 提交所有待定更改。

控制 MY\_EMPLOYEE 表的数据事务处理。

16) 利用您在步骤 6 创建的脚本中的语句，使用步骤 3 列出的示例数据中的最后一行填充表。运行该脚本中的语句。

17) 确认添加到表中的内容。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000
4	Ropeburn	Audrey	aropetur	1550

18) 标记事务处理过程中的中间点。

19) 删除 MY\_EMPLOYEE 表中的所有行。

20) 确认表为空。

21) 放弃最近一次的 DELETE 操作，但不放弃较早的 INSERT 操作。

22) 确认新行仍为原样。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	1000
2	Drexler	Ben	bbiri	1100
3	Newman	Chad	cnewman	1000
4	Ropeburn	Audrey	aropetur	1550

23) 使数据添加变成永久性添加。

## 练习 9-1：处理数据（续）

如果您有时间，请完成以下练习：

- 24) 修改 lab\_09\_06.sql 脚本，以便通过连接名字的首字母和姓氏的头七个字符，自动生成 USERID。生成的 USERID 必须为小写。因此，该脚本不应提示输入 USERID。将此脚本保存到名为 lab\_09\_24.sql 的文件中。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

- 25) 运行脚本 lab\_09\_24.sql，插入以下记录：

- 26) 确认添加的新行具有正确的 USERID。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	6 Anthony	Mark	manthony	1230

## 练习解答 9-1：处理数据

将数据插入到 **MY\_EMPLOYEE** 表中。

- 1) 运行 `lab_09_01.sql` 脚本中的语句，以构建在本练习中使用的 **MY\_EMPLOYEE** 表。
  - a) 在“File（文件）”菜单中，选择“Open（打开）”。在“Open（打开）”对话框中，导航至 `/home/oracle/labs/sql11/labs` 文件夹，然后双击 `lab_09_01.sql`。
  - b) 在 SQL 工作表中显示该语句以后，单击“Run Script（运行脚本）”图标即可运行该脚本。此时在“Script Output（脚本输出）”选项卡页上会显示“Create Table succeeded（表创建成功）”消息。
- 2) 描述 **MY\_EMPLOYEE** 表的结构以确定列名。

```
DESCRIBE my_employee
```

- 3) 创建一个 `INSERT` 语句，将下面示例数据中的第一行数据添加到 **MY\_EMPLOYEE** 表中。不要在 `INSERT` 子句中列出这些列。

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750
5	Ropeburn	Audrey	aropebur	1550

```
INSERT INTO my_employee
VALUES (1, 'Patel', 'Ralph', 'rpatel', 895);
```

- 4) 使用上面列表中的第二行示例数据填充 **MY\_EMPLOYEE** 表。这次请在 `INSERT` 子句中显式列出这些列。

```
INSERT INTO my_employee (id, last_name, first_name,
                        userid, salary)
VALUES (2, 'Dancs', 'Betty', 'bdancs', 860);
```

## 练习解答 9-1：处理数据（续）

- 5) 确认添加到表中的内容。

```
SELECT *
FROM my_employee;
```

- 6) 在动态可重用脚本文件中编写一条 INSERT 语句，以便将剩余行加载到 MY\_EMPLOYEE 表中。该脚本应提示输入所有列 (ID、LAST\_NAME、FIRST\_NAME、USERID 和 SALARY)。将此脚本保存到名为 lab\_09\_06.sql 的文件中。

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 7) 运行所创建脚本中的 INSERT 语句，使用步骤 3 列出的示例数据中的接下来两行填充表。

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
       '&p_userid', &p_salary);
```

- 8) 确认添加到表中的内容。

```
SELECT *
FROM my_employee;
```

- 9) 使数据添加变成永久性添加。

```
COMMIT;
```

**更新和删除 MY\_EMPLOYEE 表中的数据。**

- 10) 将员工 3 的姓氏更改为 Drexler。

```
UPDATE my_employee
SET last_name = 'Drexler'
WHERE id = 3;
```

- 11) 将薪金低于 \$900 的所有员工的薪金更改为 \$1,000。

```
UPDATE my_employee
SET salary = 1000
WHERE salary < 900;
```

- 12) 验证对表进行的更改。

```
SELECT *
FROM my_employee;
```

## 练习解答 9-1：处理数据（续）

13) 从 MY\_EMPLOYEE 表中删除 Betty Dancs。

```
DELETE
FROM my_employee
WHERE last_name = 'Dancs';
```

14) 确认对表所做的更改。

```
SELECT *
FROM my_employee;
```

15) 提交所有待定更改。

```
COMMIT;
```

控制 MY\_EMPLOYEE 表的数据事务处理。

16) 利用您在步骤 6 创建的脚本中的语句，使用步骤 3 列出的示例数据中的最后一行填充表。运行该脚本中的语句。

```
INSERT INTO my_employee
VALUES (&p_id, '&p_last_name', '&p_first_name',
'&p_userid', &p_salary);
```

17) 确认添加到表中的内容。

```
SELECT *
FROM my_employee;
```

18) 标记事务处理过程中的中间点。

```
SAVEPOINT step_17;
```

19) 删除 MY\_EMPLOYEE 表中的所有行。

```
DELETE
FROM my_employee;
```

20) 确认表为空。

```
SELECT *
FROM my_employee;
```

21) 放弃最近一次的 DELETE 操作，但不放弃较早的 INSERT 操作。

```
ROLLBACK TO step_17;
```

## 练习解答 9-1：处理数据（续）

22) 确认新行仍为原样。

```
SELECT *
FROM my_employee;
```

23) 使数据添加变成永久性添加。

```
COMMIT;
```

如果您有时间，请完成以下练习：

24) 修改 lab\_09\_06.sql 脚本，以便通过连接名字的首字母和姓氏的头七个字符，自动生成 USERID。生成的 USERID 必须为小写。因此，该脚本不应提示输入 USERID。将此脚本保存到名为 lab\_09\_24.sql 的文件中。

```
SET ECHO OFF
SET VERIFY OFF
INSERT INTO my_employee
VALUES ('&p_id', '&&p_last_name', '&&p_first_name',
        lower(substr('&p_first_name', 1, 1)) ||
        substr('&p_last_name', 1, 7)), &p_salary);
SET VERIFY ON
SET ECHO ON
UNDEFINE p_first_name
UNDEFINE p_last_name
```

25) 运行脚本 lab\_09\_24.sql，插入以下记录：

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
6	Anthony	Mark	manthony	1230

26) 确认添加的新行具有正确的 USERID。

```
SELECT *
FROM my_employee
WHERE ID='6';
```

## 第 10 课的练习

使用 CREATE TABLE 语句创建新表。确认新表已添加到数据库。您还会学习如何将表的状态设置为 READ ONLY，然后再还原为 READ/WRITE。

**注：**对于所有的 DDL 和 DML 语句，单击“Run Script（运行脚本）”图标（或按 [F5]）即可在 SQL Developer 中执行查询。您可以在“Script Output（脚本输出）”选项卡页上查看反馈消息。对于 SELECT 查询，可以继续单击“Execute Statement（执行语句）”图标或按 [F9]，在“Results（结果）”选项卡页上获得格式化输出。

## 练习 10-1：使用 DDL 语句创建和管理表

- 1) 根据下面的表实例图表创建 DEPT 表。将语句保存在名为 lab\_10\_01.sql 的脚本中，然后通过执行该脚本中的语句来创建表。确认该表已创建。

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

Name	Null	Type
ID	NOT NULL	NUMBER(7)
NAME		VARCHAR2(25)

- 2) 使用 DEPARTMENTS 表中的数据填充 DEPT 表。仅包括所需的列。
- 3) 根据下面的表实例图表创建 EMP 表。将语句保存在名为 lab\_10\_03.sql 的脚本中，然后通过执行该脚本中的语句来创建表。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				DEPT
FK 列				ID
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) 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。
- 5) 将 EMPLOYEES2 的表状态更改为“只读”。

## 练习 10-1：使用 DDL 语句创建和管理表（续）

- 6) 尝试将以下行插入到 EMPLOYEES2 表中：

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

您会收到以下错误消息：

```
Error starting at line 1 in command:
INSERT INTO employees2
VALUES (34, 'Grant','Marcie',5678,10)
Error at Command Line:1 Column:12
Error report:
SQL Error: ORA-12081: update operation not allowed on table "ORA1"."EMPLOYEES2"
12081. 00000 -  "update operation not allowed on table \"%s\".\"%s\""
*Cause: An attempt was made to update a read-only materialized view.
*Action: No action required. Only Oracle is allowed to update a
read-only materialized view.
```

- 7) 将 EMPLOYEES2 表还原为“读/写”状态。现在可以尝试再次插入同一行。

您应收到以下消息：

```
ALTER TABLE employees2 succeeded.
1 rows inserted
```

- 8) 删除 EMPLOYEES2 表。

## 练习解答 10-1：使用 DDL 语句创建和管理表

- 1) 根据下面的表实例图表创建 DEPT 表。将语句保存在名为 lab\_10\_01.sql 的脚本中，然后通过执行该脚本中的语句来创建表。确认该表已创建。

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

```
CREATE TABLE dept
(id NUMBER(7) CONSTRAINT department_id_pk PRIMARY KEY,
name VARCHAR2(25));
```

要确认已创建表并查看其结构，请使用以下命令：

```
DESCRIBE dept
```

- 2) 使用 DEPARTMENTS 表中的数据填充 DEPT 表。仅包括那些所需的列。

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

## 练习解答 10-1：使用 DDL 语句创建和管理表（续）

- 3) 根据下面的表实例图表创建 EMP 表。将语句保存在名为 lab\_10\_03.sql 的脚本中，然后通过执行该脚本中的语句来创建表。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				DEPT
FK 列				ID
数据类型	NUMBER	VARCHAR2	VARCHAR2	NUMBER
长度	7	25	25	7

```
CREATE TABLE emp
(id          NUMBER(7),
last_name    VARCHAR2(25),
first_name   VARCHAR2(25),
dept_id      NUMBER(7)
CONSTRAINT emp_dept_id_FK REFERENCES dept (id)
);
```

要确认已创建表并查看其结构，请使用以下命令：

```
DESCRIBE emp
```

- 4) 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;
```

- 5) 将 EMPLOYEES2 的表状态更改为“只读”。

```
ALTER TABLE employees2 READ ONLY
```

## 练习解答 10-1：使用 DDL 语句创建和管理表（续）

ID	FIRST_NAME	LAST_NAME	SALARY	DEPT_ID
34	Grant	Marcie	5678	10

- 6) 尝试将下一行插入到 EMPLOYEES2 表中。请注意，此时将显示“Update operation not allowed on table (不允许对表执行更新操作)”错误消息。因此无法在该表中插入任何行，因为表状态为“只读”。

```
INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

- 7) 将 EMPLOYEES2 表还原为“读/写”状态。现在可以尝试再次插入同一行。

现在，表状态为 READ WRITE，因此可以在表中插入行。

```
ALTER TABLE employees2 READ WRITE

INSERT INTO employees2
VALUES (34, 'Grant', 'Marcie', 5678, 10)
```

- 8) 删除 EMPLOYEES2 表。

注：甚至可以删除处于 READ ONLY 模式的表。要进行此测试，请将表再次更改为 READ ONLY 状态，然后发布 DROP TABLE 命令。此时将删除表 EMPLOYEES2。

```
DROP TABLE employees2;
```

## 第 11 课的练习

本课练习的第 1 部分为您提供了有关创建、使用和删除视图的各种练习。请完成本课的问题 1-6。

本课练习的第 2 部分为您提供创建和使用序列、索引和同义词的各种练习。请完成本课的问题 7-10。

## 练习 11-1：创建其它方案对象

### 第 1 部分

- 1) HR 部门中的工作人员希望隐藏 EMPLOYEES 表中的一些数据。需要基于 EMPLOYEES 表中的员工编号、员工姓名和部门编号，为他们创建一个名为 EMPLOYEES\_VU 的视图。员工姓名的标题应为 EMPLOYEE。
- 2) 确认视图可以正常工作。显示 EMPLOYEES\_VU 视图的内容。

	EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
5	206	Gietz	110

...

19	205 Higgins	110
20	206 Gietz	110

- 3) 使用 EMPLOYEES\_VU 视图，为 HR 部门编写一个查询，用来显示所有员工的姓名和部门编号。

	EMPLOYEE	DEPARTMENT_ID
1	King	90
2	Kochhar	90
3	De Haan	90
4	Hunold	60
5	Ernst	60

...

19	Higgins	110
20	Gietz	110

- 4) 部门 50 需要访问其员工数据。创建一个名为 DEPT50 的视图，该视图包含部门 50 中所有员工的员工编号、员工姓氏和部门编号。您需要将视图列分别标记为 EMPNO、EMPLOYEE 和 DEPTNO。为了安全起见，不允许通过视图将员工重新分配到另一个部门。

## 练习 11-1：创建其它方案对象（续）

- 5) 显示 DEPT50 视图的结构和内容。

DESCRIBE dept50		
Name	Null	Type
EMPNO	NOT NULL	NUMBER(6)
EMPLOYEE	NOT NULL	VARCHAR2(25)
DEPTNO		NUMBER(4)

EMPNO	EMPLOYEE	DEPTNO
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50
144	Vargas	50

- 6) 测试视图。尝试将 Matos 重新分配到部门 80。

### 第 2 部分

- 7) 您需要一个用于 DEPT 表的 PRIMARY KEY 列的序列。序列应从 200 开始，最大值为 1,000。序列增量为 10。将该序列命名为 DEPT\_ID\_SEQ。
- 8) 要测试您的序列，请编写一个脚本，用于在 DEPT 表中插入两个行。将您的脚本命名为 lab\_11\_08.sql。请务必使用为 ID 列创建的序列。添加以下两个部门：Education 和 Administration。确认所添加的内容。运行脚本中的命令。
- 9) 对 DEPT 表的 NAME 列创建非唯一索引。
- 10) 为 EMPLOYEES 表创建同义词。将其命名为 EMP。

## 练习解答 11-1：创建其它方案对象

### 第 1 部分

- 1) HR 部门中的工作人员希望隐藏 EMPLOYEES 表中的一些数据。需要基于 EMPLOYEES 表中的员工编号、员工姓名和部门编号，为他们创建一个名为 EMPLOYEES\_VU 的视图。员工姓名的标题应为 EMPLOYEE。

```
CREATE OR REPLACE VIEW employees_vu AS
    SELECT employee_id, last_name employee, department_id
    FROM employees;
```

- 2) 确认视图可以正常工作。显示 EMPLOYEES\_VU 视图的内容。

```
SELECT *
FROM     employees_vu;
```

- 3) 使用 EMPLOYEES\_VU 视图，为 HR 部门编写一个查询，用来显示所有员工的姓名和部门编号。

```
SELECT      employee, department_id
FROM        employees_vu;
```

- 4) 部门 50 需要访问其员工数据。创建一个名为 DEPT50 的视图，该视图包含部门 50 中所有员工的员工编号、员工姓氏和部门编号。您需要将视图列分别标记为 EMPNO、EMPLOYEE 和 DEPTNO。为了安全起见，不允许通过视图将员工重新分配到另一个部门。

```
CREATE VIEW dept50 AS
    SELECT      employee_id empno, last_name employee,
                department_id deptno
    FROM        employees
    WHERE       department_id = 50
    WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

- 5) 显示 DEPT50 视图的结构和内容。

```
DESCRIBE dept50
SELECT      *
FROM        dept50;
```

- 6) 测试视图。尝试将 Matos 重新分配到部门 80。

```
UPDATE      dept50
SET         deptno = 80
WHERE      employee = 'Matos';
```

由于 DEPT50 视图是使用 WITH CHECK OPTION 约束条件创建的，因此会出现错误。这样可以确保视图中的 DEPTNO 列免遭更改。

## 练习解答 11-1：创建其它方案对象（续）

### 第 2 部分

- 7) 您需要一个用于 DEPT 表的主键列的序列。序列应从 200 开始，最大值为 1,000。序列增量为 10。将该序列命名为 DEPT\_ID\_SEQ。

```
CREATE SEQUENCE dept_id_seq
    START WITH 200
    INCREMENT BY 10
    MAXVALUE 1000;
```

- 8) 要测试您的序列，请编写一个脚本，用于在 DEPT 表中插入两个行。将您的脚本命名为 lab\_11\_08.sql。请务必使用为 ID 列创建的序列。添加以下两个部门：Education 和 Administration。确认所添加的内容。运行脚本中的命令。

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

- 9) 对 DEPT 表的 NAME 列创建非唯一索引。

```
CREATE INDEX dept_name_idx ON dept (name);
```

- 10) 为 EMPLOYEES 表创建同义词。将其命名为 EMP。

```
CREATE SYNONYM emp FOR EMPLOYEES;
```

## 附录 F 的练习

本练习旨在为您提供实际经验，帮助您使用 Oracle 联接语法从多个表中提取数据。

## 练习 F-1: Oracle 联接语法

- 1) 为 HR 部门编写一个查询来生成所有部门的地址。请使用 LOCATIONS 和 COUNTRIES 表。在输出中显示位置 ID、街道地址、城市、州或省以及国家/地区。运行该查询。

LOCATION_ID	STREET_ADDRESS	CITY	STATE_PROVINCE	COUNTRY_NAME
1	1400 2014 Jabberwocky Rd	Southlake	Texas	United States of America
2	1500 2011 Interiors Blvd	South San Francisco	California	United States of America
3	1700 2004 Charade Rd	Seattle	Washington	United States of America
4	1800 460 Bloor St. W.	Toronto	Ontario	Canada
5	2500 Magdalene Centre, The Oxford Science Park	Oxford		United Kingdom

- 2) HR 部门需要一个包含所有员工的报表。编写一个查询，显示所有员工的姓氏、部门编号和部门名称。运行该查询。

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1 Whalen	10	Administration
2 Hartstein	20	Marketing
3 Fay	20	Marketing
4 Davies	50	Shipping
5 Vargas	50	Shipping

...

18 Higgins	110	Accounting
19 Gietz	110	Accounting

- 3) HR 部门需要一个在多伦多工作的员工的报表，显示在多伦多工作的所有员工的姓氏、职务、部门编号和部门名称。

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1 Hartstein	MK_MAN	20	Marketing
2 Fay	MK_REP	20	Marketing

## 练习 F-1: Oracle 联接语法 (续)

- 4) 创建一个报表，显示员工的姓氏和员工编号及其经理的姓氏和经理编号。将这些列分别标记为 Employee、Emp#、Manager 和 Mgr#。将 SQL 语句另存为 lab\_f\_04.sql。

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

18	Taylor	176	Zlotkey	149
19	Abel	174	Zlotkey	149

- 5) 修改 lab\_f\_04.sql，用于显示所有员工，包括没有经理的员工 King。按员工编号对结果进行排序。将 SQL 语句另存为 lab\_f\_05.sql。运行 lab\_f\_05.sql 中的查询。

	Employee	EMP#	Manager	Mgr#
1	Hunold	103	De Haan	102
2	Fay	202	Hartstein	201
3	Gietz	206	Higgins	205
4	Lorentz	107	Hunold	103
5	Ernst	104	Hunold	103

...

19	Abel	174	Zlotkey	149
20	King	100	(null)	(null)

## 练习 F-1: Oracle 联接语法 (续)

- 6) 为 HR 部门创建一个报表，用于显示员工的姓氏、部门编号以及与该员工在同一部门中工作的所有员工。为每个列指定一个合适的标签。将脚本保存到名为 lab\_f\_06.sql 的文件中。

	DEPARTMENT	EMPLOYEE	COLLEAGUE
1	20	Fay	Hartstein
2	20	Hartstein	Fay
3	50	Davies	Matos
4	50	Davies	Mourgos
5	50	Davies	Rajs

...

39	90	Kochhar	De Haan
40	90	Kochhar	King
41	110	Gietz	Higgins
42	110	Higgins	Gietz

- 7) HR 部门需要一个关于职务等级和薪金的报表。为了熟悉 JOB\_GRADES 表，应先显示 JOB\_GRADES 表的结构。然后创建一个查询，用于显示所有员工的姓名、职务、部门名称、薪金和等级。

Name	Null	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRADE_LEVEL
1 King	AD_PRES	Executive	24000	E
2 De Haan	AD_VP	Executive	17000	E
3 Kochhar	AD_VP	Executive	17000	E
4 Hartstein	MK_MAN	Marketing	13000	D
5 Higgins	AC_MGR	Accounting	12000	D

...

18 Matos	ST_CLERK	Shipping	2600	A
19 Vargas	ST_CLERK	Shipping	2500	A

## 练习 F-1: Oracle 联接语法 (续)

如果您要接受更多的挑战, 请完成下面的练习:

- 8) HR 部门希望确定在 Davies 之后聘用的所有员工的姓名。创建一个查询, 用于显示在员工 Davies 之后聘用的任何员工的姓名和聘用日期。

	LAST_NAME	HIRE_DATE
1	Lorentz	07-FEB-99
2	Mourgos	16-NOV-99
3	Matos	15-MAR-98
4	Vargas	09-JUL-98
5	Zlotkey	29-JAN-00
6	Taylor	24-MAR-98
7	Grant	24-MAY-99
8	Fay	17-AUG-97

- 9) HR 部门需要查找在其经理之前聘用的所有员工的姓名和聘用日期, 及其经理的姓名和聘用日期。将脚本保存到名为 lab\_f\_09.sql 的文件中。

	LAST_NAME	HIRE_DATE	LAST_NAME_1	HIRE_DATE_1
1	Whalen	17-SEP-87	Kochhar	21-SEP-89
2	Hunold	03-JAN-90	De Haan	13-JAN-93
3	Vargas	09-JUL-98	Mourgos	16-NOV-99
4	Matos	15-MAR-98	Mourgos	16-NOV-99
5	Davies	29-JAN-97	Mourgos	16-NOV-99
6	Rajs	17-OCT-95	Mourgos	16-NOV-99
7	Grant	24-MAY-99	Zlotkey	29-JAN-00
8	Taylor	24-MAR-98	Zlotkey	29-JAN-00
9	Abel	11-MAY-96	Zlotkey	29-JAN-00

## 练习解答 F-1: Oracle 联接语法

- 1) 为 HR 部门编写一个查询来生成所有部门的地址。请使用 LOCATIONS 和 COUNTRIES 表。在输出中显示位置 ID、街道地址、城市、州或省以及国家/地区。运行该查询。

```
SELECT location_id, street_address, city, state_province,
       country_name
  FROM   locations, countries
 WHERE  locations.country_id = countries.country_id;
```

- 2) HR 部门需要一个包含所有员工的报表。编写一个查询，显示所有员工的姓氏、部门编号和部门名称。运行该查询。

```
SELECT e.last_name, e.department_id, d.department_name
  FROM   employees e, departments d
 WHERE  e.department_id = d.department_id;
```

- 3) HR 部门需要一个在多伦多工作的员工的报表，显示在多伦多工作的所有员工的姓氏、职务、部门编号和部门名称。

```
SELECT e.last_name, e.job_id, e.department_id,
       d.department_name
  FROM   employees e, departments d , locations l
 WHERE  e.department_id = d.department_id
    AND  d.location_id = l.location_id
    AND  LOWER(l.city) = 'toronto';
```

- 4) 创建一个报表，用于显示员工的姓氏和员工编号及其经理的姓氏和经理编号。将这些列分别标记为 Employee、Emp#、Manager 和 Mgr#。将 SQL 语句另存为 lab\_f\_04.sql。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w, employees m
 WHERE  w.manager_id = m.employee_id;
```

- 5) 修改 lab\_f\_04.sql，用于显示所有员工，包括没有经理的员工 King。按员工编号对结果进行排序。将 SQL 语句另存为 lab\_f\_05.sql。运行 lab\_f\_05.sql 中的查询。

```
SELECT w.last_name "Employee", w.employee_id "EMP#",
       m.last_name "Manager", m.employee_id "Mgr#"
  FROM   employees w, employees m
 WHERE  w.manager_id = m.employee_id (+);
```

## 练习解答 F-1: Oracle 联接语法 (续)

- 6) 为 HR 部门创建一个报表，用于显示员工的姓氏、部门编号以及与该员工在同一部门中工作的所有员工。为每个列指定一个合适的标签。将脚本保存到名为 lab\_f\_06.sql 的文件中。

```
SELECT e1.department_id department, e1.last_name employee,
       e2.last_name colleague
  FROM employees e1, employees e2
 WHERE e1.department_id = e2.department_id
   AND e1.employee_id <> e2.employee_id
ORDER BY e1.department_id, e1.last_name, e2.last_name;
```

- 7) HR 部门需要一个关于职务等级和薪金的报表。为了熟悉 JOB\_GRADES 表，应先显示 JOB\_GRADES 表的结构。然后创建一个查询，用于显示所有员工的姓名、职务、部门名称、薪金和等级。

```
DESC JOB_GRADES

SELECT e.last_name, e.job_id, d.department_name,
       e.salary, j.grade_level
  FROM employees e, departments d, job_grades j
 WHERE e.department_id = d.department_id
   AND e.salary BETWEEN j.lowest_sal AND j.highest_sal;
```

如果您要接受更多的挑战，请完成下面的练习：

- 8) HR 部门希望确定在 Davies 之后聘用的所有员工的姓名。创建一个查询，用于显示在 Davies 之后聘用的任何员工的姓名和聘用日期。

```
SELECT e.last_name, e.hire_date
  FROM employees e, employees davies
 WHERE davies.last_name = 'Davies'
   AND davies.hire_date < e.hire_date;
```

- 9) HR 部门需要查找在其经理之前聘用的所有员工的姓名和聘用日期，及其经理的姓名和聘用日期。将这些列分别标记为 Employee、Emp Hired、Manager 和 Mgr Hired。将脚本保存到名为 lab\_f\_09.sql 的文件中。

```
SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
  FROM employees w, employees m
 WHERE w.manager_id = m.employee_id
   AND w.hire_date < m.hire_date;
```



# 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)

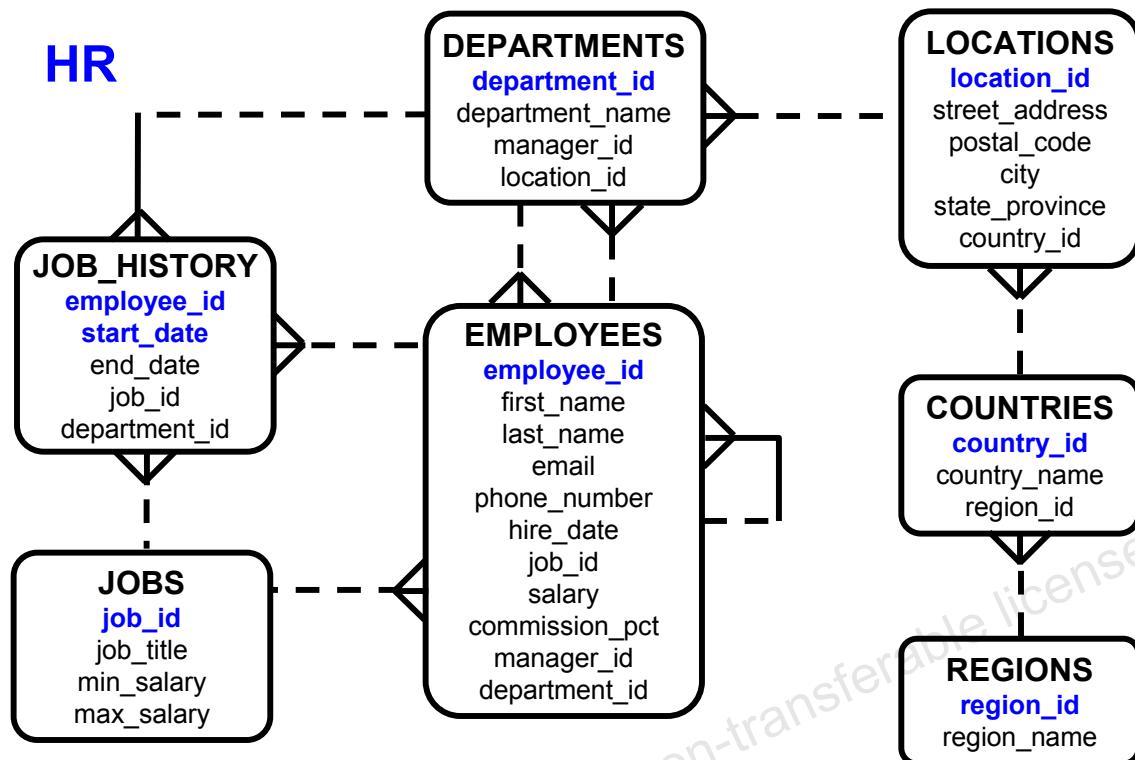
这是本课程中使用的方案。在人力资源 (HR) 记录中，每位雇员都有各自的标识号、电子邮件地址、职务标识代码、薪金和经理。某些雇员除了薪金还领取佣金。

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	Europe
2	Americas
3	Asia
4	Middle East and Africa

# 使用 SQL Developer

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 SQL Developer 的主要功能
- 识别 Oracle SQL Developer 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

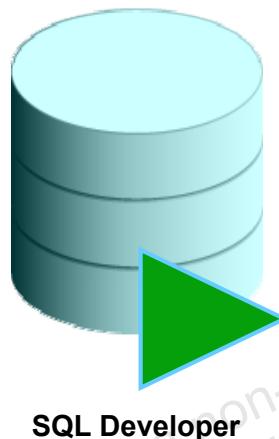
版权所有 © 2010, Oracle。保留所有权利。

### 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。

作为一种用于数据库开发的可视化工具，SQL Developer 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
- 支持使用 JDBC 瘦驱动程序建立默认连接
- 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 Oracle Database 11g 发行版 2 一起提供。SQL Developer 是利用 Oracle JDeveloper 集成开发环境 (IDE) 使用 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，会通过 Java 数据库连接 (JDBC) 瘦驱动程序连接到数据库，因此不需要使用 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 Database 版本，您需要自行下载并安装 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)

有关如何安装 SQL Developer 的说明，请访问以下网站：

[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)**：包含用于导航至窗格和执行子程序的选项。
- **Run (运行)**：包含选择函数或过程时要使用的相关选项“Run File (运行文件)” 和“Execution Profile (执行概要文件)”，此外还包括调试选项。
- **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, Concurrent Versions System) 和 Subversion。
- **Migration (移植)**：包含将第三方数据库移植到 Oracle 时要使用的相关选项。
- **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相关选项。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。
- 可以针对下面两种情况创建和测试连接：
  - 多个数据库
  - 多个方案
- 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。

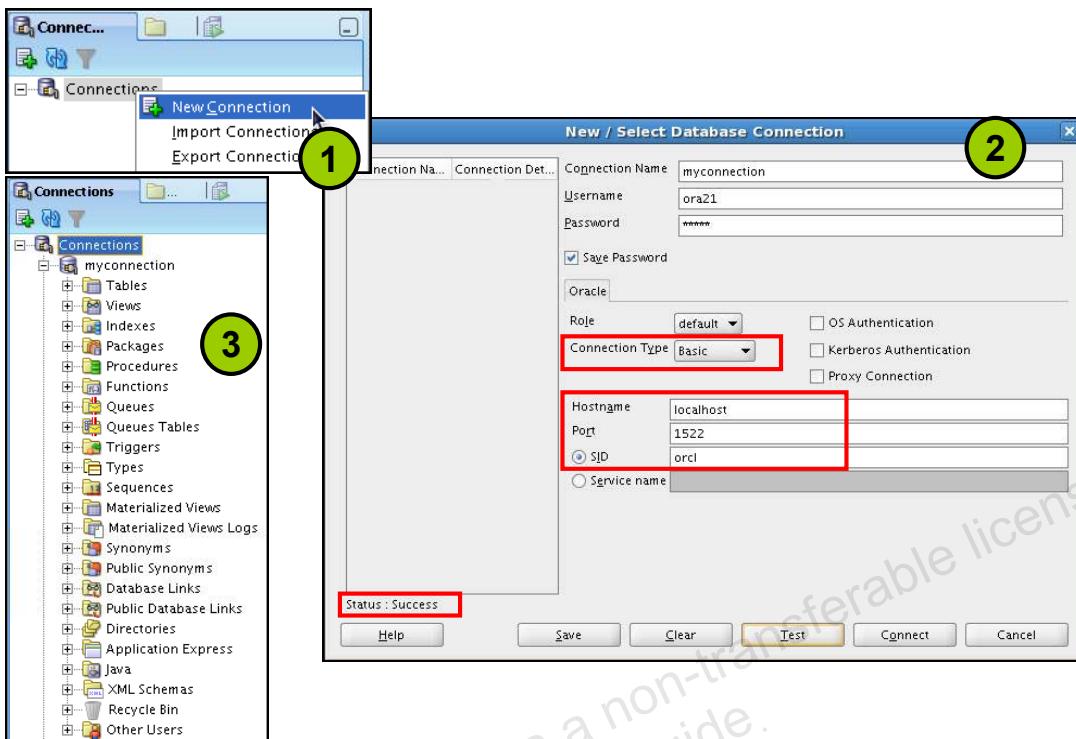
默认情况下，`tnsnames.ora` 文件位于 `$ORACLE_HOME/network/admin` 目录中，但该文件也可位于 `TNS_ADMIN` 环境变量或注册表值指定的目录中。启动 SQL Developer 后显示“Database Connections（数据库连接）”对话框时，SQL Developer 会自动导入系统的 `tnsnames.ora` 文件中定义的所有连接。

**注：**在 Windows 系统上，如果 `tnsnames.ora` 文件存在，但 SQL Developer 未使用其连接，请将 `TNS_ADMIN` 定义为系统环境变量。

可以将连接导出为 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



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 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) 从“Role（角色）”下拉列表中，可选择“Default（默认）”或“SYSDBA”。  
(对于 sys 用户或任何具有数据库管理员权限的用户，选择 SYSDBA。)
  - b) 可选择的连接类型包括：
    - Basic（基本）**：在此类型中，需输入要连接到的数据库的主机名和 SID。  
“Port（端口）”已设置为 1521。此外，如果使用远程数据库连接，也可以选择直接输入“Service（服务）”的名称。
    - 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 的一个组件）中查找数据库服务。
    - Advanced（高级）**：可以定义一个用以连接到数据库的定制 Java 数据库连接 (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。

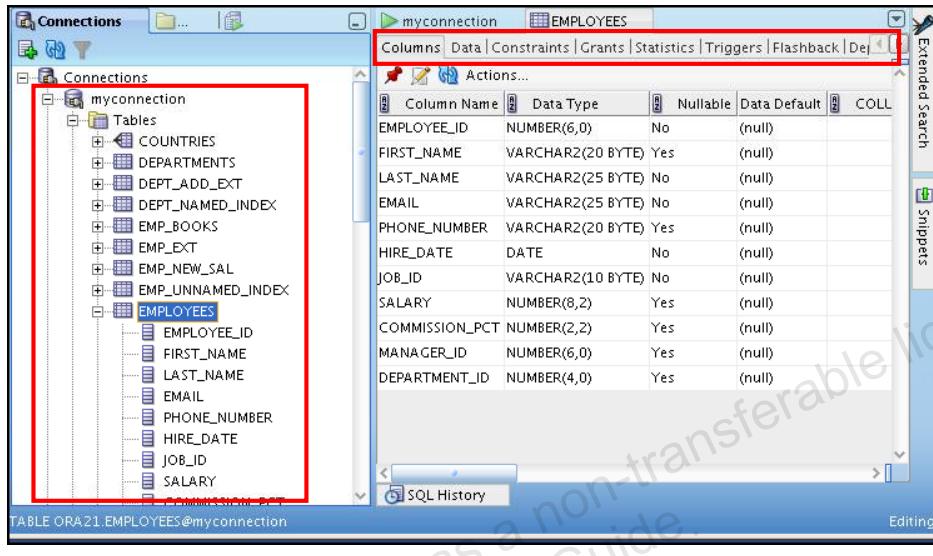
- 3. 此时该连接已添加在“Connections Navigator（连接导航器）”中。您可以展开该连接查看数据库对象和对象定义，例如，依赖关系、详细资料和统计信息等。

**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



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。

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。通过设置首选项可以定制 SQL Developer 的各种不同的外观。

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。

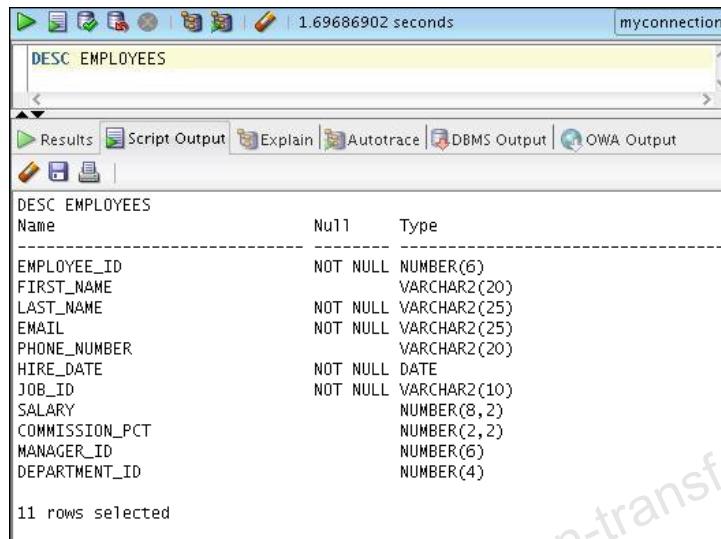
例如，如果在该导航器中选择一个表，则会在选项卡页上显示关于列、约束条件、权限、统计信息、触发器等的详细资料，非常易于查看。

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 SQL Developer interface with a connection named 'myconnection'. In the top-left query editor, the command 'DESC EMPLOYEES' is entered. Below the command, the results are displayed in a table format:

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)

At the bottom of the results pane, it says '11 rows selected'.

ORACLE

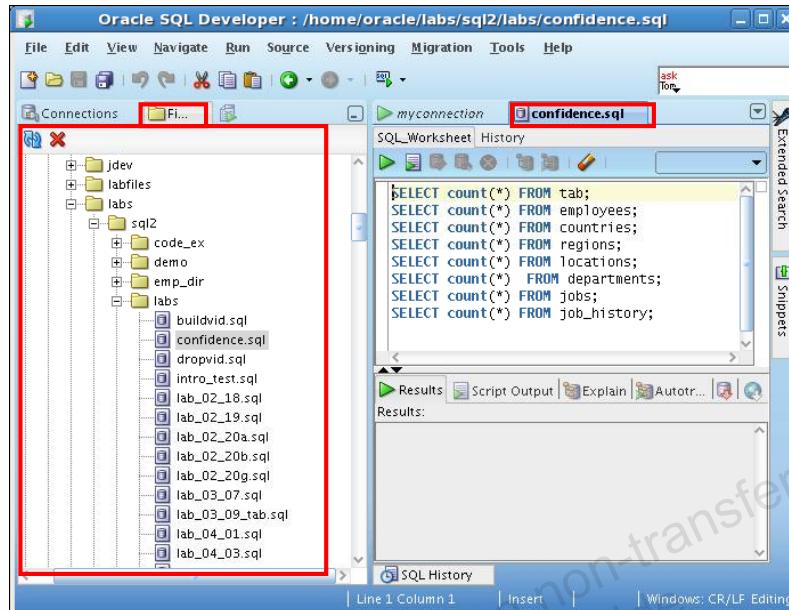
版权所有 © 2010, Oracle。保留所有权利。

### 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。

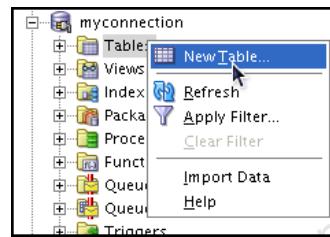
## 浏览数据库对象

可以使用文件导航器浏览并打开系统文件。

- 要查看文件浏览器，请单击“Files（文件）”选项卡，或选择“View > Files（视图 > 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。

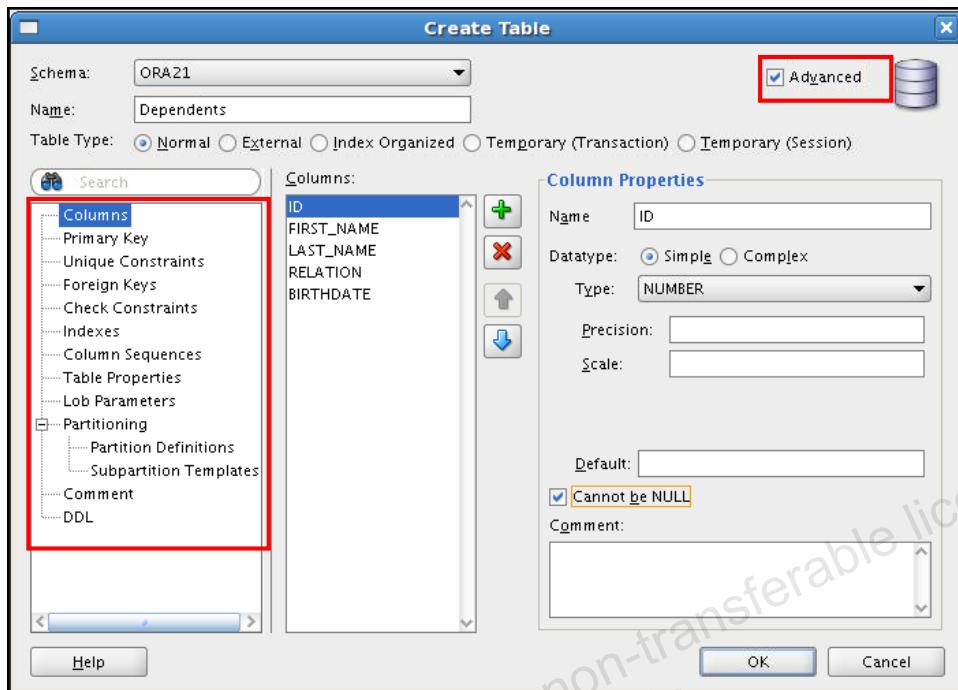
### 创建方案对象

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。

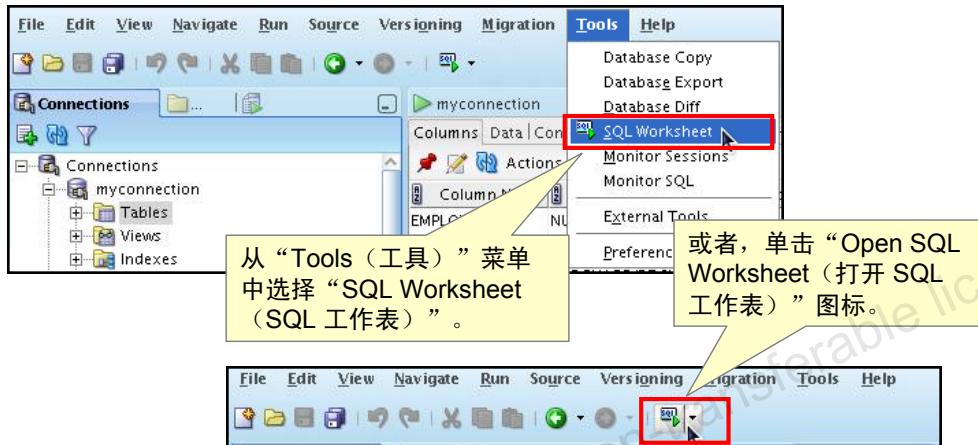
要创建一个新表，请执行以下步骤：

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。

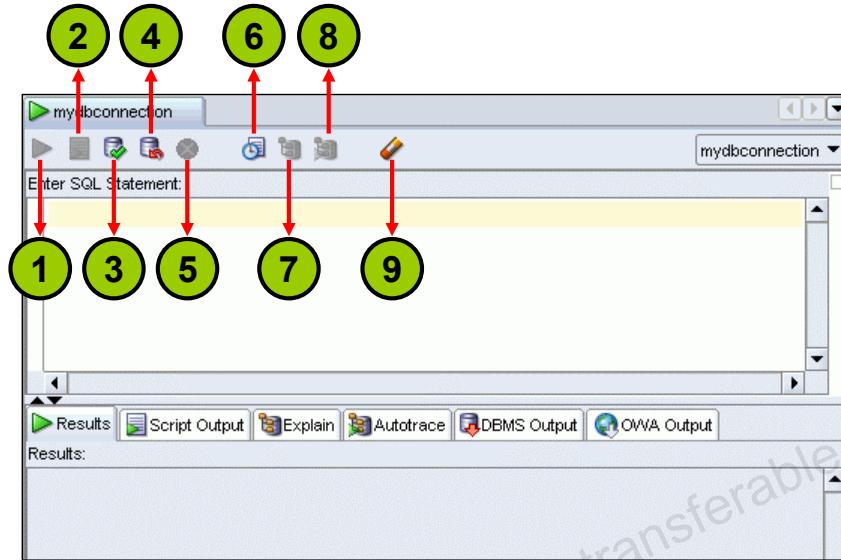
可以指定与工作表相关的数据库连接可处理的操作，例如：

- 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



ORACLE

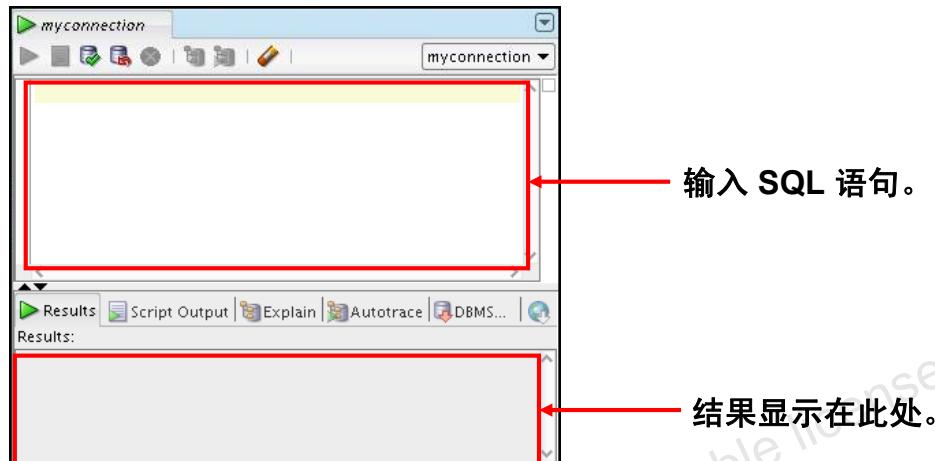
版权所有 © 2010, Oracle。保留所有权利。

### 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，但不能使用绑定变量。
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：**停止目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



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。

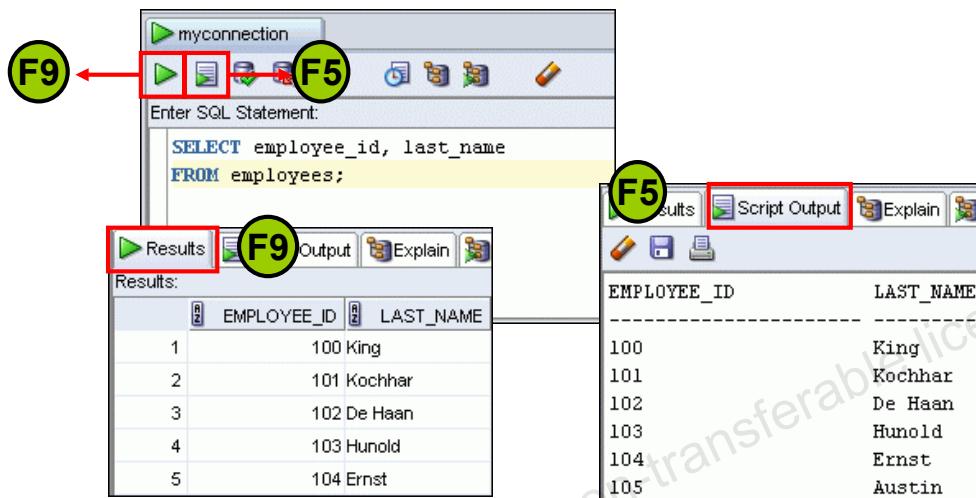
SQL 工作表当前支持许多 SQL\*Plus 命令。SQL 工作表不支持的命令将被忽略，不会发送到 Oracle 数据库。通过 SQL 工作表，可以执行 SQL 语句和一些 SQL\*Plus 命令。

可以使用以下任意选项来显示 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使用两种不同方式执行同一查询时的输出差异：一种方式是按 [F9] 键或使用“Execute Statement（执行语句）”；另一种方式是按 [F5] 键或使用“Run Script（运行脚本）。”



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 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）”框中的内容，请执行以下步骤：

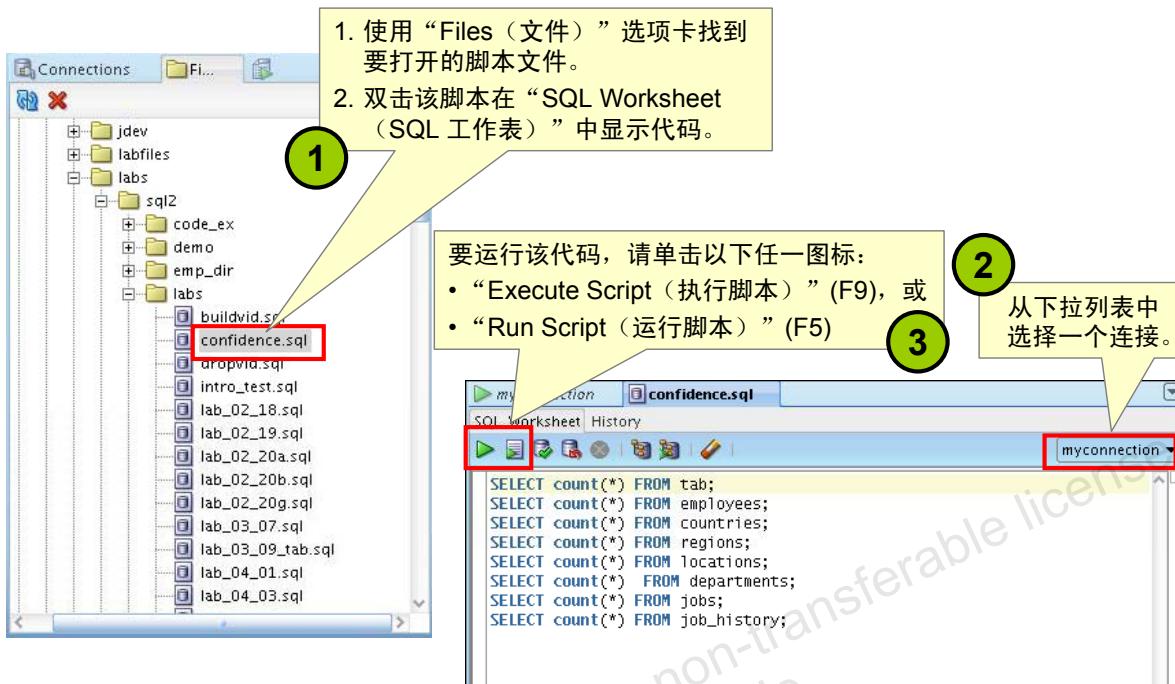
1. 单击“Save（保存）”图标或使用“File > Save（文件 > 保存）”菜单项。
2. 在“Save（保存）”对话框中，输入文件名和要保存文件的位置。
3. 单击“Save（保存）”。

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



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 1

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

1. 在文件导航器中，选择（或导航到）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。

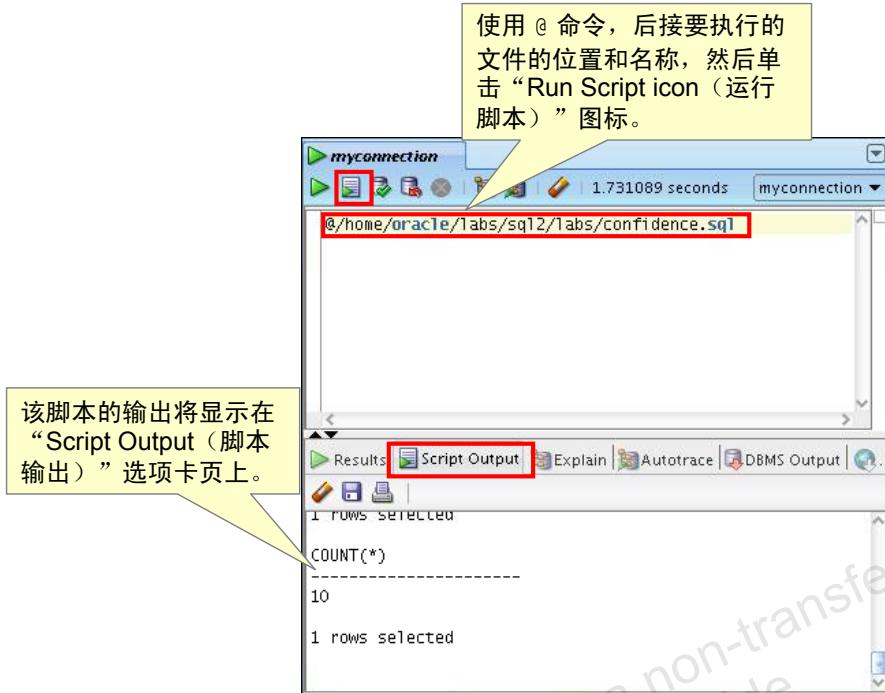
或者，也可以执行下列操作：

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。

### 执行已保存的脚本文件：方法 1（续）

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。请选择要用于执行脚本的连接。

## 执行已保存的脚本文件：方法 2



版权所有 © 2010, Oracle。保留所有权利。

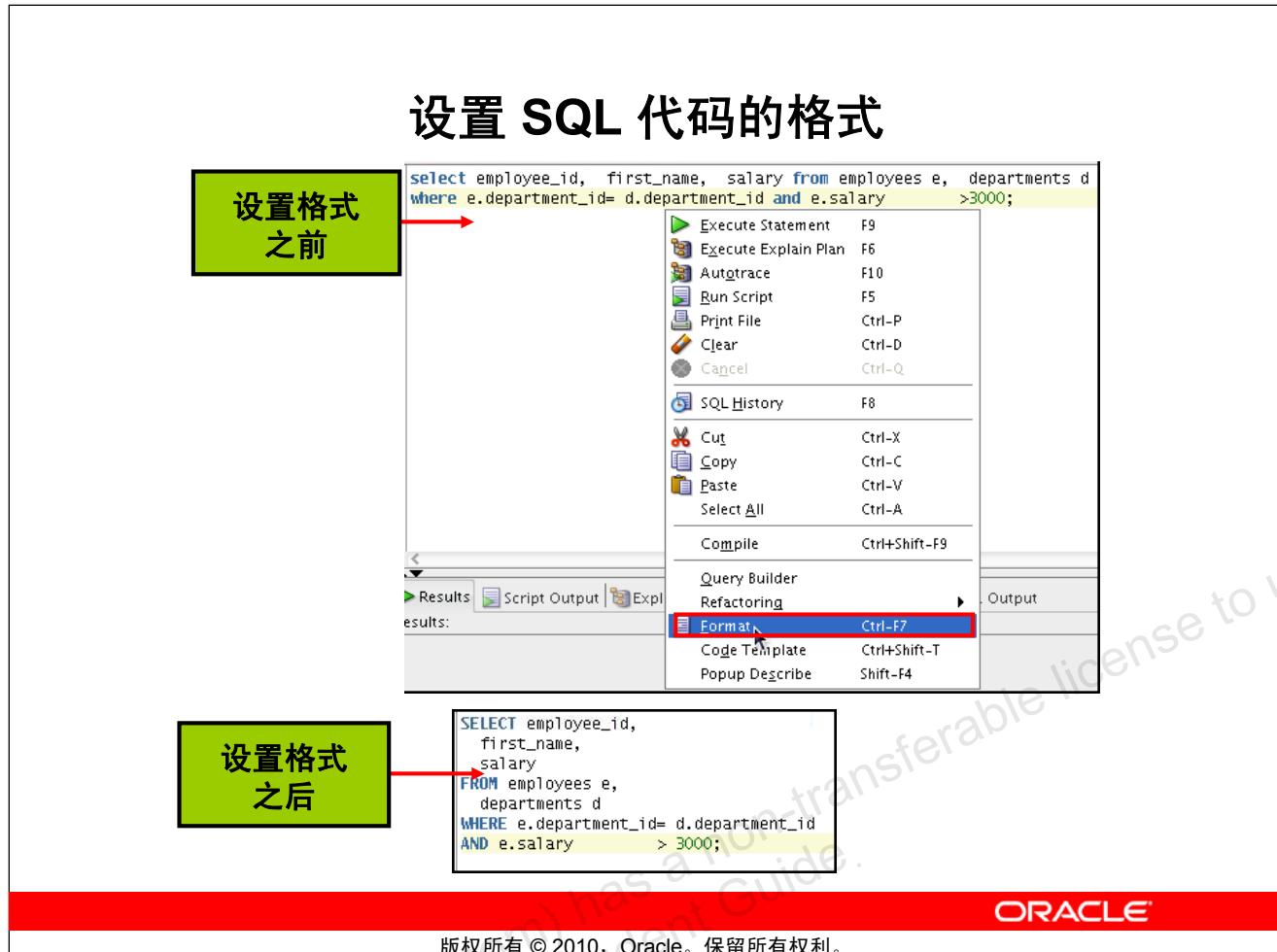
ORACLE

### 执行已保存的脚本文件：方法 2

要运行已保存的 SQL 脚本，请执行以下步骤：

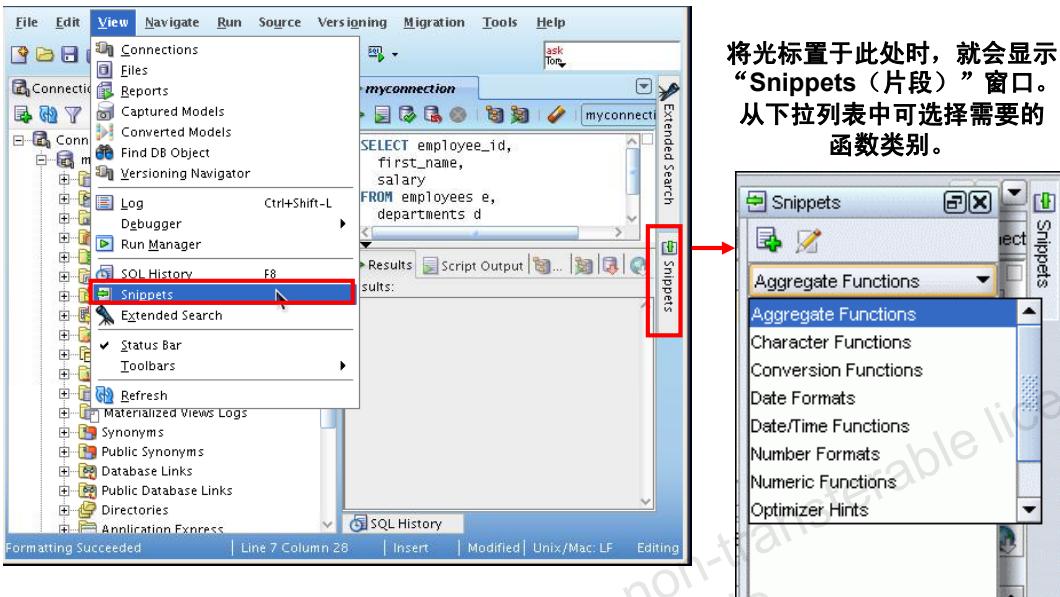
1. 在“Enter SQL Statement（输入 SQL 语句）”窗口中，使用 @ 命令并在后面加上要运行的文件的位置和名称。
2. 单击“Run Script（运行脚本）”图标。

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。在随后显示的“File Save（保存文件）”对话框中，您可以为文件指定名称和位置。



## 使用片段

片段是代码段，可能只是语法或示例。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

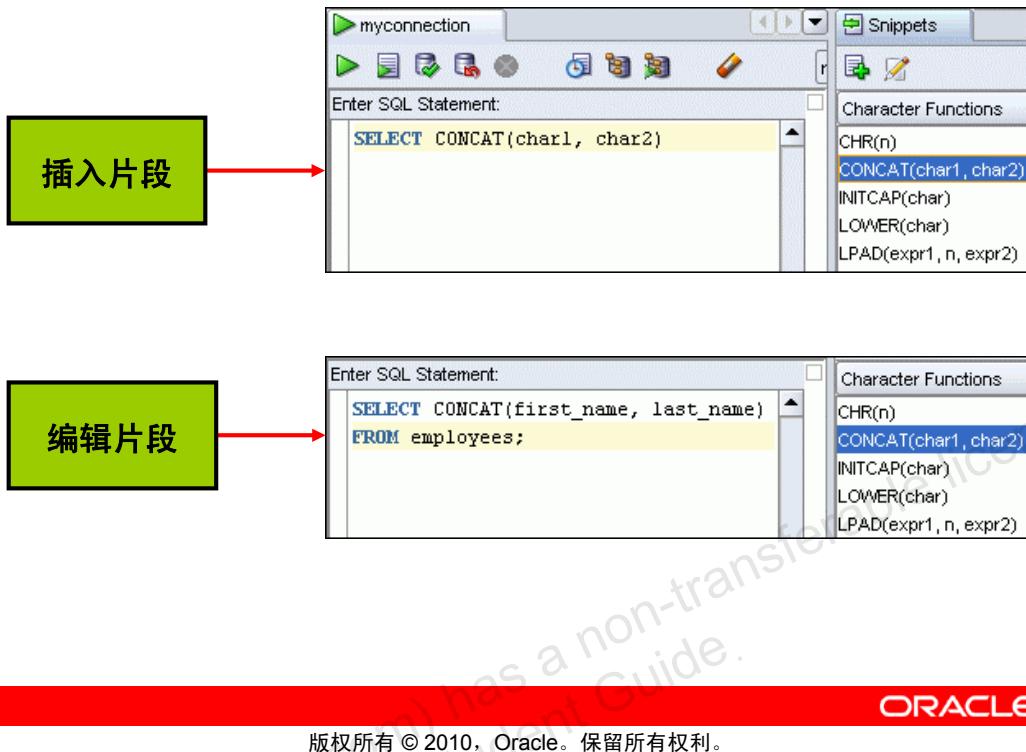
### 使用片段

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）”窗口将显示在右侧。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



### 使用片段：示例

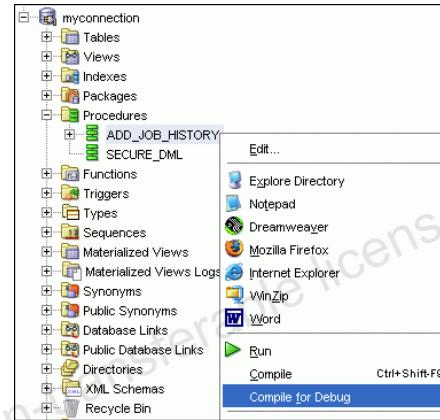
要将片段插入 SQL 工作表或 PL/SQL 函数或过程的代码中，请将片段从“Snippets（片段）”窗口拖至代码中的目标位置。然后可以编辑该语法，使 SQL 函数在当前上下文中生效。要在工具提示中查看 SQL 函数的简短说明，请将光标放在相应函数名的上方。

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）选项可执行 PL/SQL 编译，以便对过程进行调试。
- 使用“Debug（调试）”菜单选项可设置断点以及执行步入、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，这样可以恢复执行。
- “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	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：

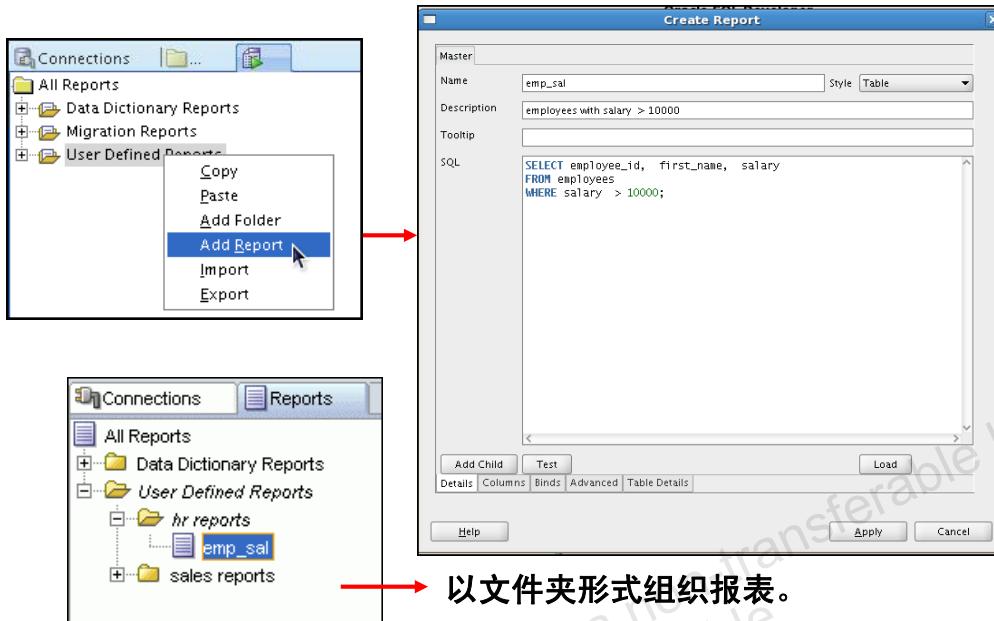
- 关于数据库的报表
- 数据库管理报表
- 表报表
- PL/SQL 报表
- 安全性报表
- XML 报表
- 作业报表
- 流报表
- 所有对象报表
- 数据字典报表
- 用户定义报表

## 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。单个报表都显示在窗口右侧的选项卡窗格中。对于每种报表，您都可以（使用下拉列表）选择要为其显示报表的数据连接。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

创建并保存用户定义报表以供反复使用。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 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



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 搜索引擎和外部工具

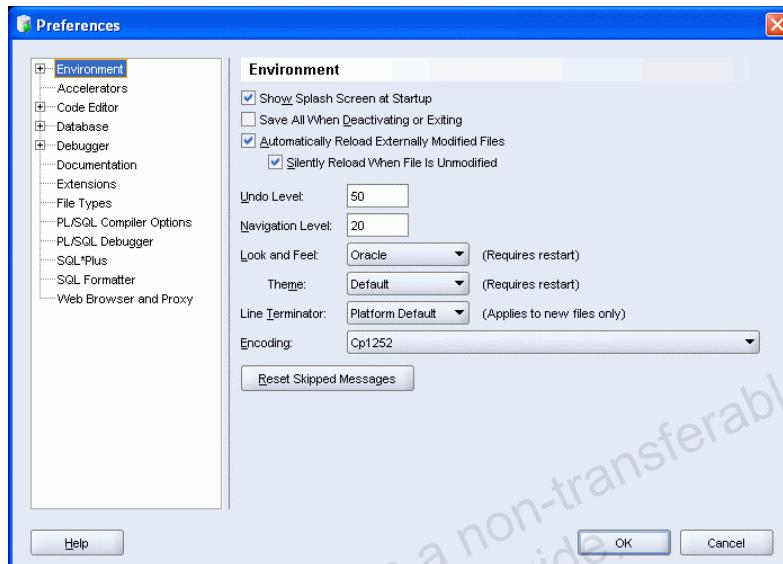
为了提高 SQL 开发人员的生产效率，SQL Developer 添加了常用搜索引擎和论坛（如 AskTom、Google 等）的快速链接。此外，还提供了一些常用工具（如 Notepad、Microsoft Word 和 Dreamweaver）的快捷方式图标。

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：

- 环境
- Accelerators（加速键，即键盘快捷键）
- Code Editors（代码编辑器）
- 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）

## 重置 SQL Developer 布局

The screenshot shows a terminal window titled "Terminal". The user has run the command "locate windowinglayout.xml" which finds two files: "/home/oracle/.sqldeveloper/system1.5.4.59.40/o.ide.11.1.1.0.22.49.48/windowinglayout.xml" and "/home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48/windowinglayout.xml". The user then runs "cd /home/oracle/.sqldeveloper/system1.5.4.59.41/o.ide.11.1.1.0.22.49.48" followed by "ls" to list the contents of the directory, which include "Debugging.layout", "Editing.layout", "projects", "windowinglayout.xml", "dtcache.xml", "preferences.xml", and "settings.xml". Finally, the user runs "rm windowinglayout.xml" to delete the file. The terminal window has a red border at the bottom.

ORACLE

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. 转到 windowinglayout.xml 所在的目录并删除该文件。
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。

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：

- 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

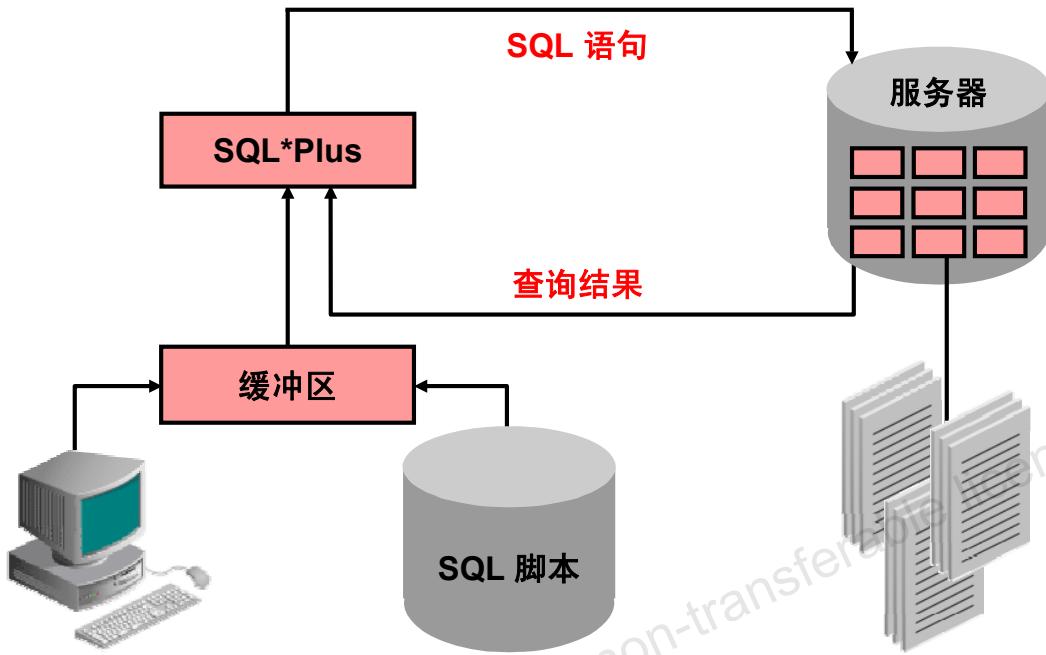
ORACLE

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



### 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
- 设置查询结果的格式，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

```
[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]]]**

```
[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)



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。此时将出现 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] [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、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.		

版权所有 © 2010, Oracle。保留所有权利。

## 使用 SAVE、START 和 EDIT 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 | UNLIMTED}]
[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 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
```



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》。
- 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：

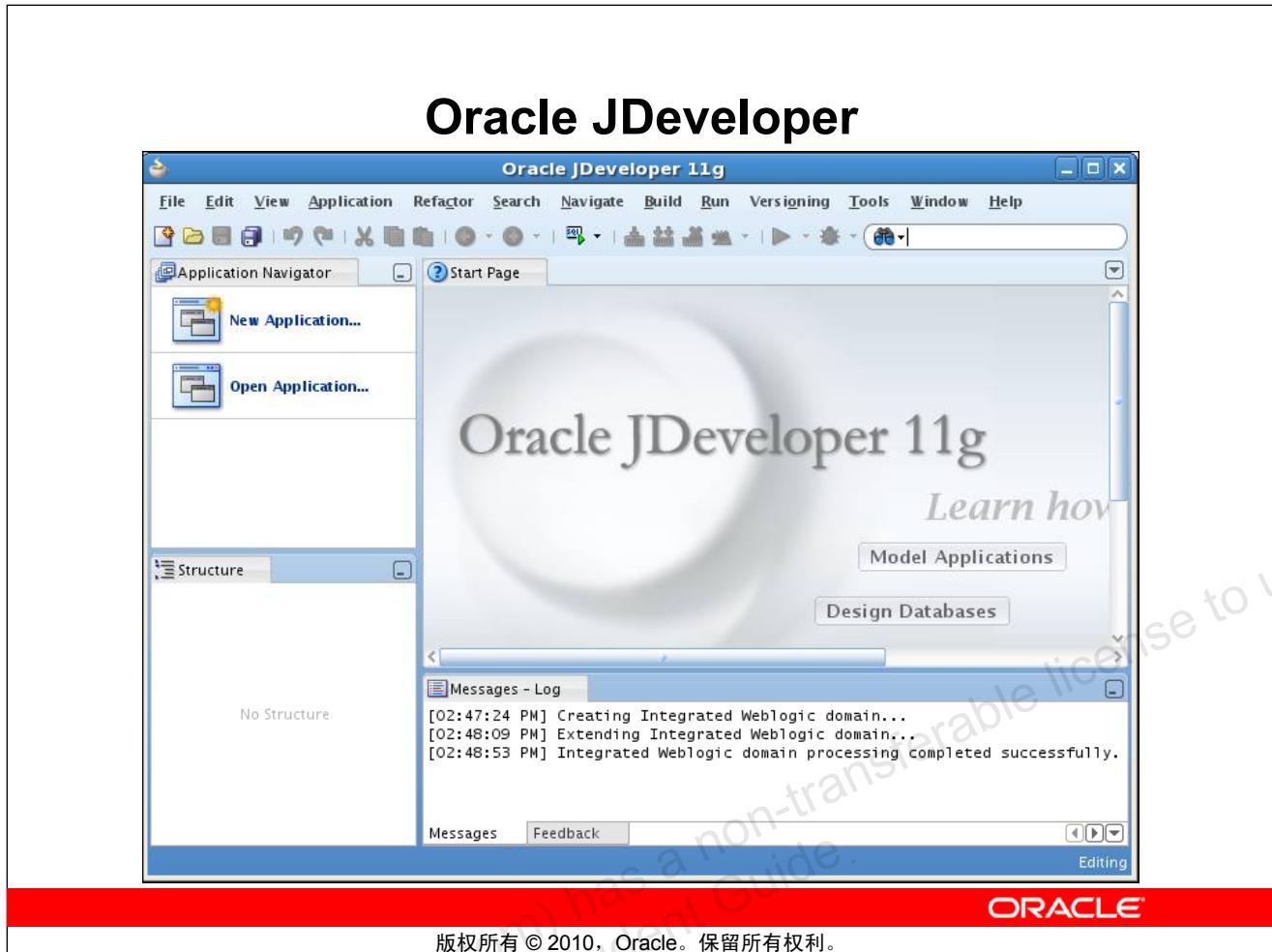
- 列举 Oracle JDeveloper 的主要功能
- 在 JDeveloper 中创建数据库连接
- 在 JDeveloper 中管理数据库对象
- 使用 JDeveloper 执行 SQL 命令
- 创建并运行 PL/SQL 程序单元

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 课程目标

本附录向您介绍 JDeveloper。您将学习如何使用 JDeveloper 执行数据库开发任务。

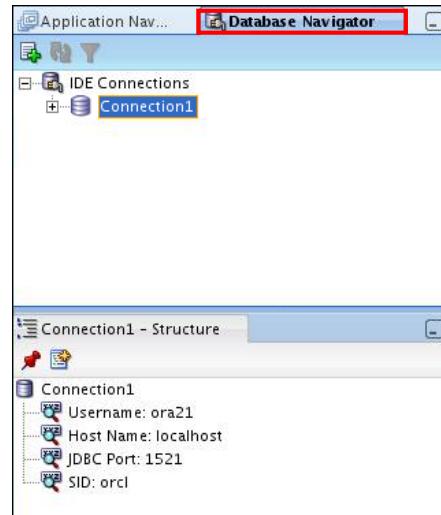


## Oracle JDeveloper

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

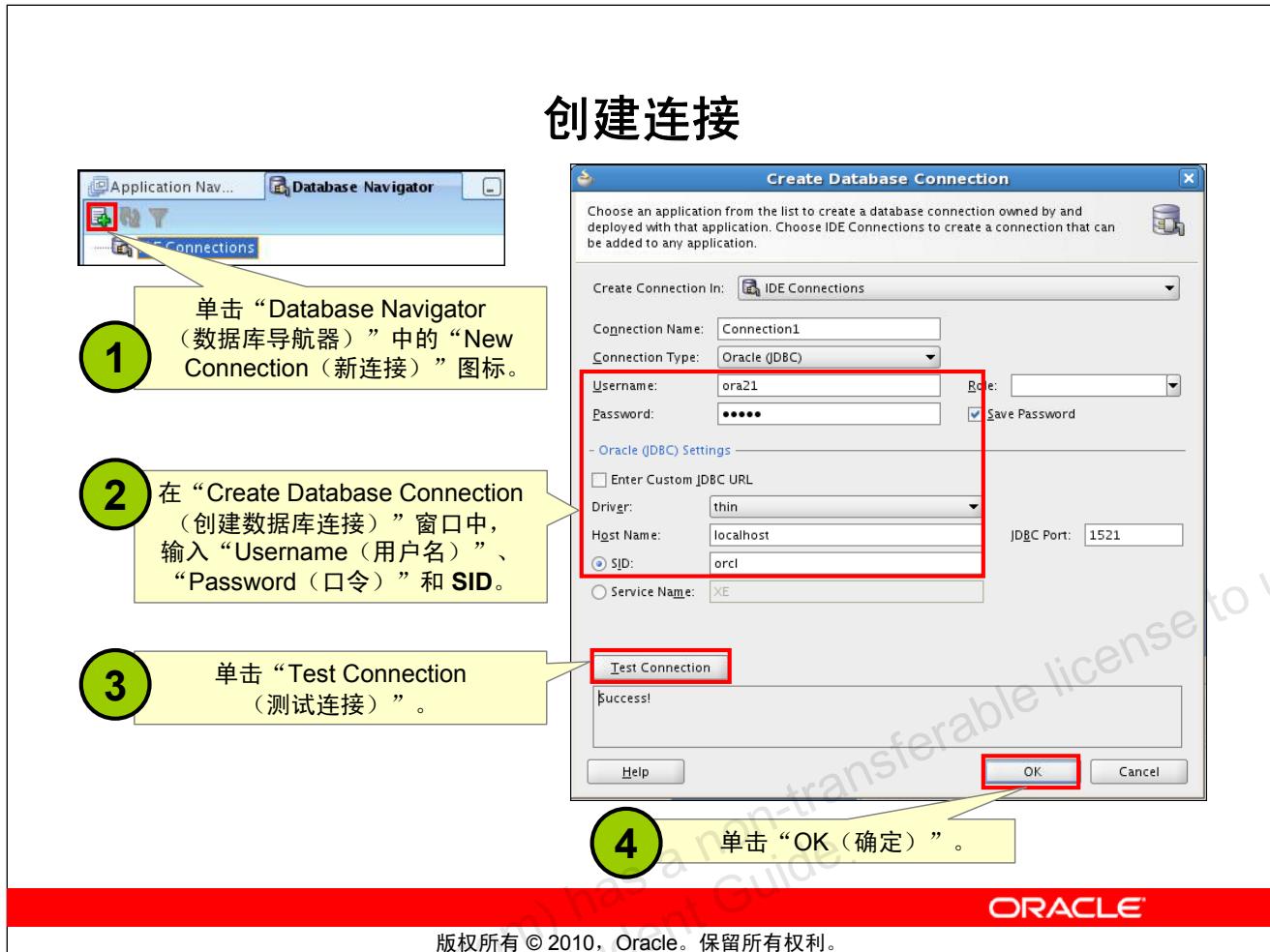


ORACLE

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。

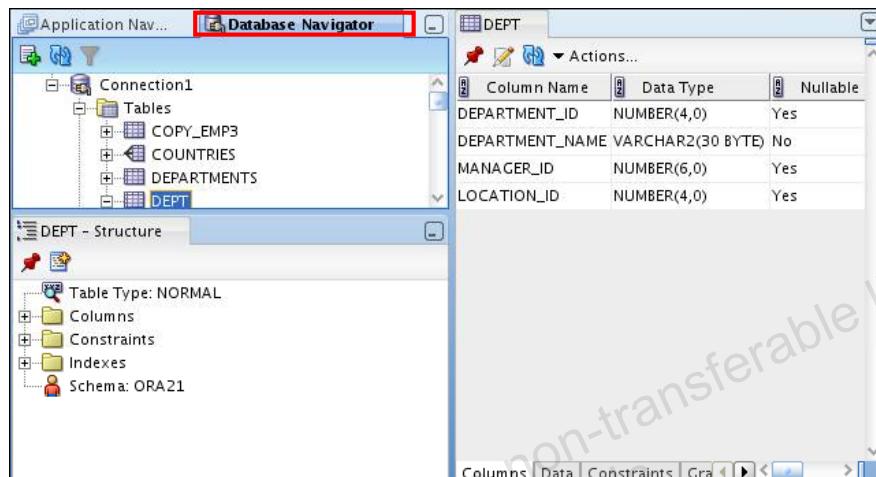
要创建数据库连接，请执行以下步骤：

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。输入要连接数据库的 SID。
3. 单击“Test（测试）”，以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



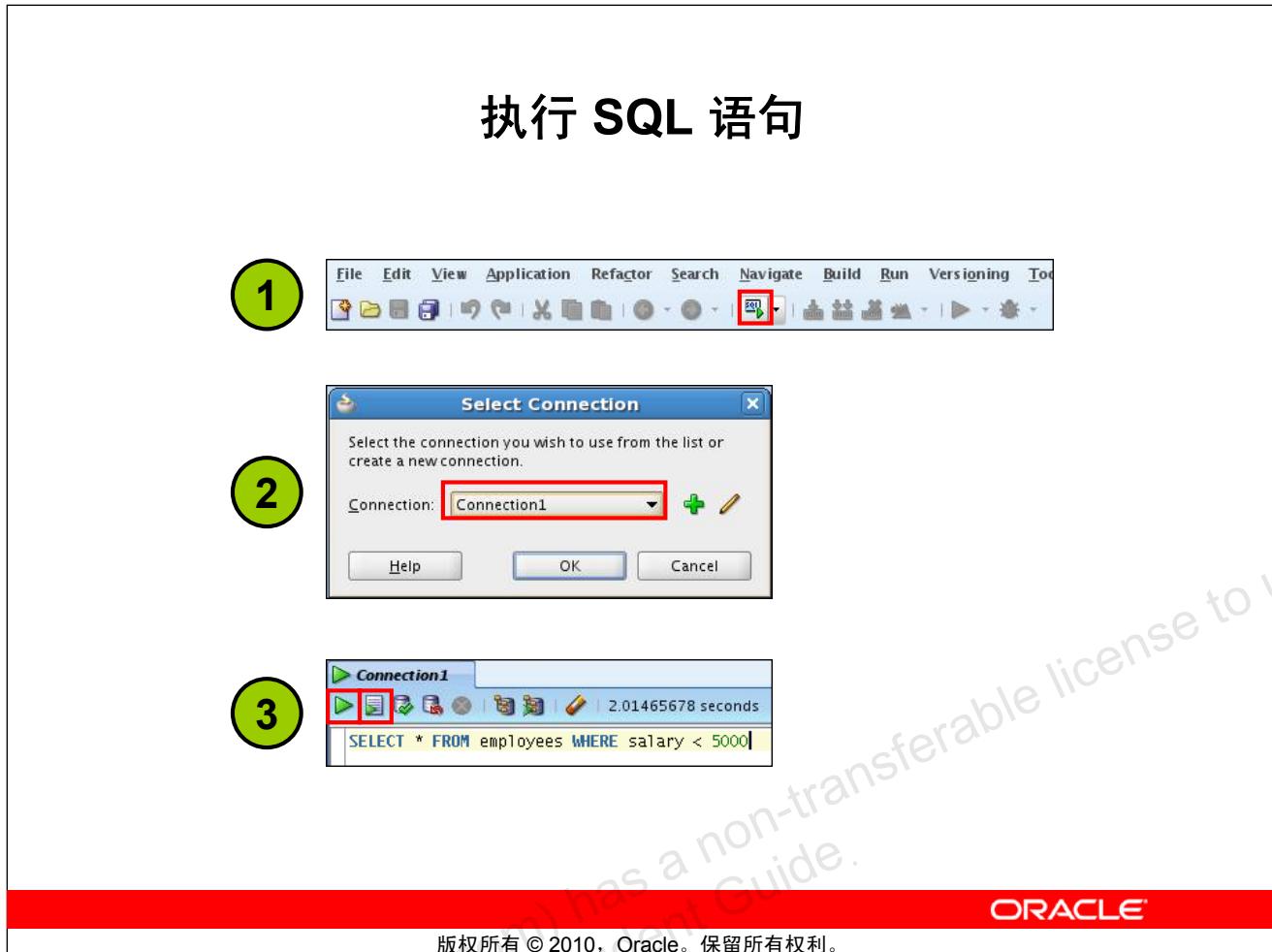
ORACLE

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“Database Navigator（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、类型等等。

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“Database Navigator（数据库导航器）”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

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：
  1. “Execute statement（执行语句）”按钮或按 F9。输出如下所示：

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME
1	100	Steven	King
2	101	Neena	Kochhar

2. “Run Script（运行脚本）”按钮或按 F5。输出如下所示：

EMPLOYEE_ID	FIRST_NAME	LAST_NAME
100	Steven	King

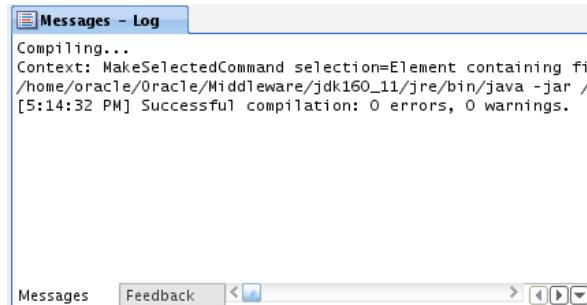


## 创建程序单元

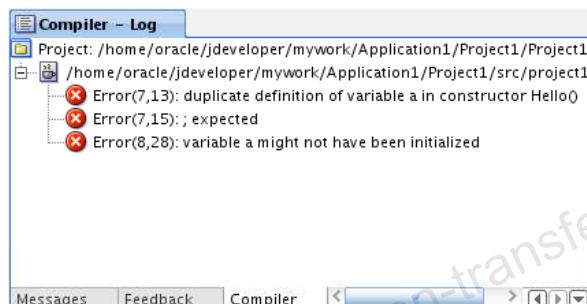
要创建 PL/SQL 程序单元，请执行以下步骤：

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

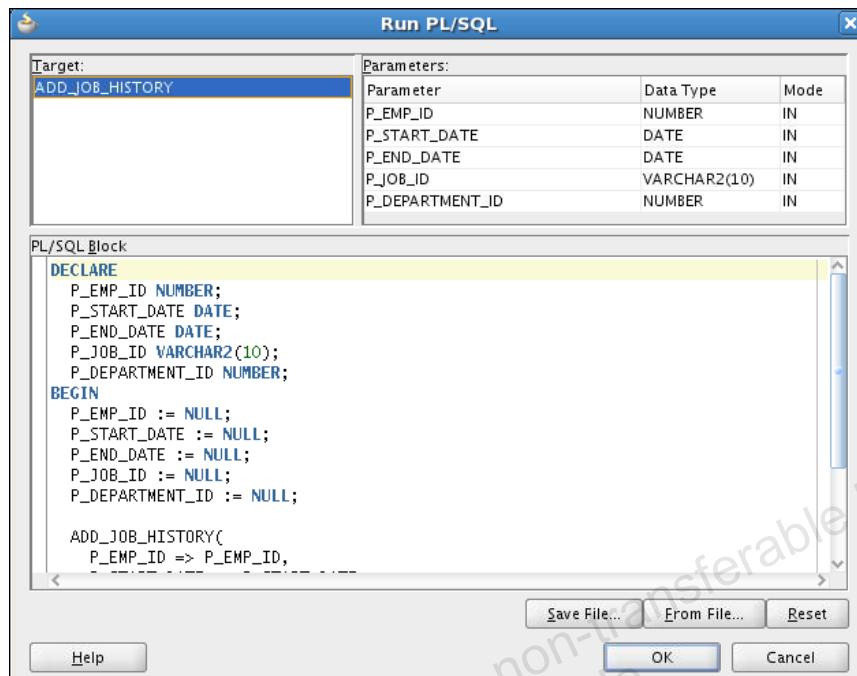
ORACLE

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）”。或者，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

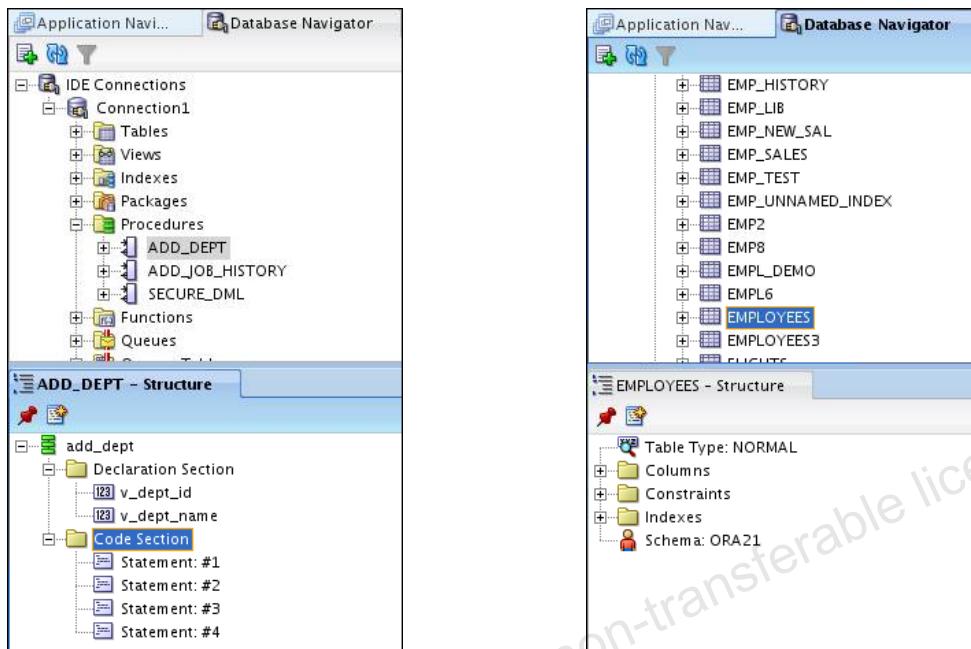
要执行程序单元，右键单击对象，再单击“Run（运行）”。此时将出现“Run PL/SQL（运行 PL/SQL）”对话框。可能需要将 NULL 值更改为可以传递到程序单元的合理值。更改这些值之后，单击“OK（确定）”。输出结果将显示在“Message-Log（消息日志）”窗口中。



## 删除程序单元

要删除程序单元，右键单击对象，再选择“Drop（删除）”。此时将出现“Drop Confirmation（删除确认）”对话框，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

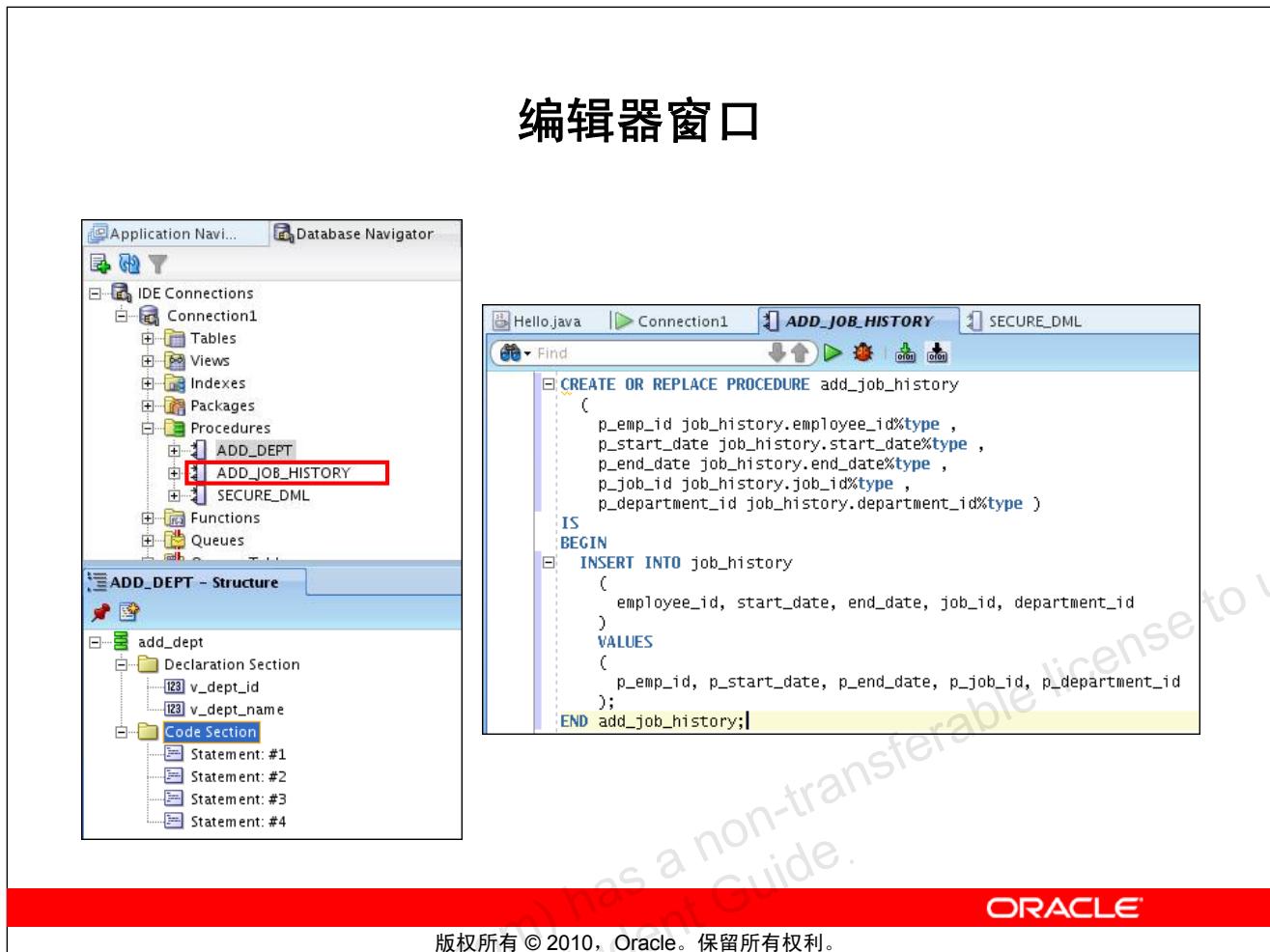
“Structure (结构) ” 窗口提供在活动窗口中当前选择的文档中数据的结构视图，该活动窗口可以是下列参与提供结构的窗口之一：导航器、编辑器、查看器和属性检查器。

单击 “View > Structure (查看 > 结构) ” 窗口，以查看 “Structure (结构) ” 窗口。

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。要更改当前选择对象的结构视图，请单击不同的结构选项卡。

## 编辑器窗口



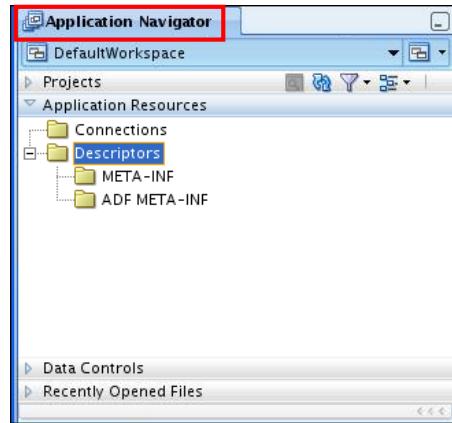
### 编辑器窗口

双击某程序单元的名称可将其在“Editor（编辑器）”窗口中打开。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。单击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。

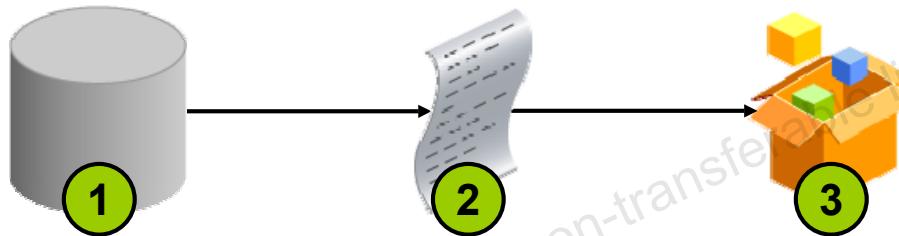
### 应用程序导航器

“Application - Navigator（应用程序 - 导航器）”提供了应用程序及其所含数据的逻辑视图。它提供了一个基础结构，可以在该基础结构中插入各种扩展并使用该基础结构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（例如 Java 源文件），但它是为合并复杂数据而设计的。复杂数据类型（例如实体对象、UML 图表、EJB 或 Web 服务）作为单一节点显示在此导航器中。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 side-by-side. The left window is titled 'TrimLob.java' and contains Java code for a main method that connects to an Oracle database using JDBC. The right window is titled 'TRIMLOBPROC' and contains the PL/SQL code for creating a procedure named TRIMLOBPROC that calls the Java main method. A green circle with the number '1' is positioned over the Java code, and a green circle with the number '2' is positioned over the PL/SQL code.

```
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 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 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 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 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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

# Oracle 联接语法

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：

- 编写 SELECT 语句，以使用等值联接和非等值联接访问多个表中的数据
- 使用自联接将表联接到自身
- 使用外部联接查看通常不满足联接条件的数据
- 生成两个或多个表中所有行的笛卡尔积

ORACLE

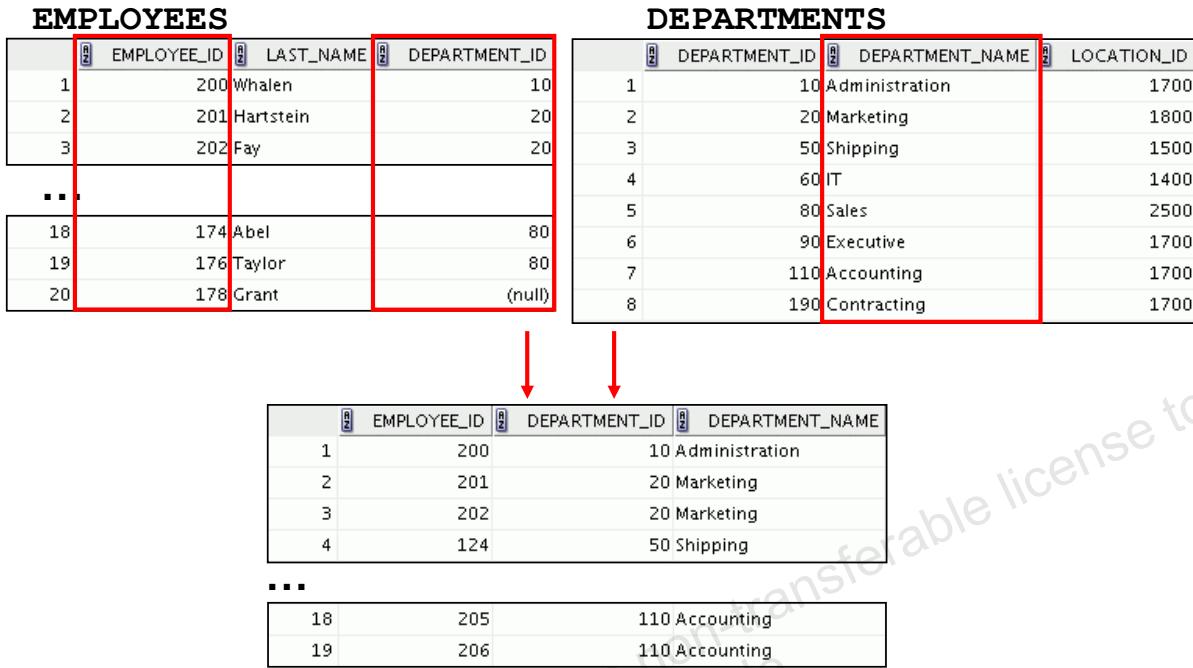
版权所有 © 2010, Oracle。保留所有权利。

### 课程目标

本课将介绍如何从多个表中获取数据。可以使用联接来查看多个表中的信息。因此，可以将表联接在一起以查看多个表中的信息。

注：有关联接的信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中“SQL Queries and Subqueries: Joins”一节。

## 获取多个表中的数据



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 获取多个表中的数据

有时需要使用多个表中的数据。在幻灯片示例中，报表中显示了两个独立表中的数据：

- 雇员 ID 在 EMPLOYEES 表中。
- 部门 ID 在 EMPLOYEES 和 DEPARTMENTS 两个表中。
- 部门名称在 DEPARTMENTS 表中。

要生成该报表，需要将 EMPLOYEES 表和 DEPARTMENTS 表链接起来，然后访问这两个表中的数据。

## 笛卡尔积

- 出现以下情况时将形成笛卡尔积：
  - 联接条件被忽略
  - 联接条件无效
  - 第一个表中的所有行被联接到第二个表中的所有行
- 为了避免生成笛卡尔积，请始终在 WHERE 子句中包括有效的联接条件。



版权所有 © 2010, Oracle。保留所有权利。

### 笛卡尔积

当一个联接条件无效或被完全忽略时，就会生成笛卡尔积。此时会显示行的所有组合。换句话说，第一个表中的所有行会被联接到第二个表中的所有行。

笛卡尔积往往会产生大量的行，这种结果几乎没有任何用处。因此，应始终包括有效联接条件，除非您有特别需要，需要组合所有表中的所有行。

不过，如果某些测试需要生成大量的行来模拟合理的数据量，则笛卡尔积非常有用。

## 生成笛卡尔积

**EMPLOYEES (20 行)**

#	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
19	176	Taylor	80
20	178	Grant	(null)

**DEPARTMENTS (8 行)**

#	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

**笛卡尔积:**  
**20 × 8 = 160 行**

#	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
...			
21	200	10	1800
22	201	20	1800
...			
159	176	80	1700
160	178	(null)	1700

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

## 生成笛卡尔积

当联接条件被省略时，就会生成笛卡尔积。幻灯片示例中分别显示 EMPLOYEES 表和 DEPARTMENTS 表中的雇员姓氏和部门名称。由于没有指定联接条件，EMPLOYEES 表中的所有行（20 行）与 DEPARTMENTS 表中的所有行（8 行）联接在一起，因此在输出中生成了 160 行。

```
SELECT last_name, department_name dept_name
FROM employees, departments;
```

#	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration
...		
158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

## Oracle 专用联接的类型

- 等值联接
- 非等值联接
- 外部联接
- 自联接



版权所有 © 2010, Oracle。保留所有权利。

### 联接类型

要联接表，可以使用 Oracle 的联接语法。

**注：**在 Oracle9i 之前的版本中，该联接语法是专用语法。与 Oracle 专用联接语法相比，符合 SQL:1999 的联接语法并不提供任何性能优势。

Oracle 没有提供等效语法来支持符合 SQL:1999 的联接语法中的 FULL OUTER JOIN。

## 使用 Oracle 语法联接表

使用联接可查询多个表中的数据：

```
SELECT      table1.column, table2.column
FROM        table1, table2
WHERE       table1.column1 = table2.column2;
```

- 在 WHERE 子句中编写联接条件。
- 如果在多个表中出现了相同的列名，则在列名的前面加上表名作为前缀。



版权所有 © 2010, Oracle。保留所有权利。

### 使用 Oracle 语法联接表

如果需要使用数据库的多个表中的数据，则可以使用联接条件。可以根据相应列（通常为主键列和外键列）中的共同值，将一个表中的行与另一个表中的行联接在一起。

要显示两个或更多个相关表中的数据，请在 WHERE 子句中编写一个简单的联接条件。

在该语法中：

`table1.column` 表示从中检索数据的表和列

`table1.column1 = table2.column2` 将表联接（或关联）在一起的条件

#### 准则

- 在编写用于联接表的 SELECT 语句时，请将表名作为前缀添加在列名前，这样可以避免混淆，而且还可以增强数据库的访问能力。
- 如果多个表中出现相同的列名，则必须将表名作为前缀添加在列名前。
- 要将  $n$  个表联接在一起，最少需要  $n-1$  个联接条件。例如，要联接四个表，最少需要三个联接。如果表具有级联主键（这种情况下需要使用多个列才能唯一地标识每个行），则该规则可能不适用。

## 限定不确定的列名

- 使用表前缀可以限定多个表中的列名。
- 使用表前缀可以提高性能。
- 可以使用表别名来代替完整表名前缀。
- 表别名是表的短名称。
  - 使 SQL 代码变得更短，因而占用更少的内存
- 使用列别名可区分具有相同名称但位于不同表中的列。



版权所有 © 2010, Oracle。保留所有权利。

### 限定不确定的列名

联接两个或更多表时，需要使用表名来限定列的名称，以避免混淆。如果不使用表前缀，则 SELECT 列表中的 DEPARTMENT\_ID 列可能来自 DEPARTMENTS 表，也可能来自 EMPLOYEES 表。因此，需要添加表前缀来执行查询。如果两个表中没有相同的列名，则无需限定列。但是，使用表前缀可以提高性能，因为这等于告知 Oracle Server 查找这些列的确切位置。

用表名限定列名可能非常耗时，特别是当表名较长时。因此，可以使用表别名代替表名。就像列别名是列的另一个名称一样，表别名也是表的另一个名称。表别名有助于使 SQL 代码变得更短，因而占用更少的内存。

先指定表全名，然后是一个空格，再后面是表别名。例如，EMPLOYEES 表的别名可以是 e，而 DEPARTMENTS 表的别名可以是 d。

#### 准则

- 表别名的长度最多为 30 个字符，但越短越好。
- 如果在 FROM 子句中使用了某个特定表名的表别名，则必须在整个 SELECT 语句中使用该表别名代替该表名。
- 表别名应是有意义的名称。
- 表别名仅对当前的 SELECT 语句有效。

# 等值联接

**EMPLOYEES**

	EMPLOYEE_ID	DEPARTMENT_ID
1	200	10
2	201	20
3	202	20
4	205	110
5	206	110
6	100	90
7	101	90
8	102	90
9	103	60
10	104	60

**DEPARTMENTS**

	DEPARTMENT_ID	DEPARTMENT_NAME
1	10	Administration
2	20	Marketing
3	50	Shipping
4	60	IT
5	80	Sales
6	90	Executive
7	110	Accounting
8	190	Contracting

外键

主键

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 等值联接

要确定雇员的部门名称，应将 EMPLOYEES 表中 DEPARTMENT\_ID 列的值与 DEPARTMENTS 表中 DEPARTMENT\_ID 列的值进行比较。EMPLOYEES 表和 DEPARTMENTS 表之间的关系就是等值联接关系，即这两个表中 DEPARTMENT\_ID 列的值必须相等。通常，这种类型的联接涉及到主键和外键的补码。

注：等值联接也称为简单联接或内部联接。

## 使用等值联接检索记录

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
  FROM employees e, departments d
 WHERE e.department_id = d.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	200 Whalen	10	10	1700
2	201 Hartstein	20	20	1800
3	202 Fay	20	20	1800
4	144 Vargas	50	50	1500
5	143 Matos	50	50	1500
6	142 Davies	50	50	1500
7	141 Rajs	50	50	1500
8	124 Mourgos	50	50	1500
9	103 Hunold	60	60	1400
10	104 Ernst	60	60	1400
11	107 Lorentz	60	60	1400
...				

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用等值联接检索记录

在幻灯片示例中：

- **SELECT 子句指定要检索的列名：**
  - EMPLOYEES 表中的雇员姓氏列、雇员编号列和部门编号列
  - DEPARTMENTS 表中的部门编号列、部门名称列和位置 ID 列
- **FROM 子句指定数据库必须访问的两个表：**
  - EMPLOYEES 表
  - DEPARTMENTS 表
- **WHERE 子句指定如何将表联接起来：**

$$e.department_id = d.department_id$$

由于 DEPARTMENT\_ID 列是两个表的共有列，所以必须将表名作为前缀添加在列名前，以避免混淆。没有同时出现在两个表中的其它列不需要使用表别名来加以限定，但是建议这样做以提高性能。

注：使用“Execute Statement（执行语句）”图标运行查询时，SQL Developer 会通过添加后缀“\_1”来区分两个 DEPARTMENT\_ID。

## 使用等值联接检索记录：示例

```
SELECT d.department_id, d.department_name,
       d.location_id, l.city
  FROM departments d, locations l
 WHERE d.location_id = l.location_id;
```

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
1	60	IT	1400	Southlake
2	50	Shipping	1500	South San Francisco
3	10	Administration	1700	Seattle
4	90	Executive	1700	Seattle
5	110	Accounting	1700	Seattle
6	190	Contracting	1700	Seattle
7	20	Marketing	1800	Toronto
8	80	Sales	2500	Oxford



版权所有 © 2010, Oracle。保留所有权利。

## 使用等值联接检索记录：示例

在幻灯片示例中，通过 LOCATION\_ID 列将 LOCATIONS 表和 DEPARTMENTS 表联接起来，LOCATION\_ID 列是唯一在两个表中具有相同名称的列。已使用表别名限定该列，因而可避免混淆。

## 使用 AND 运算符的附加搜索条件

```
SELECT d.department_id, d.department_name, l.city
FROM departments d, locations l
WHERE d.location_id = l.location_id
AND d.department_id IN (20, 50);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	CITY
1	20	Marketing	Toronto
2	50	Shipping	South San Francisco

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 使用 AND 运算符的附加搜索条件

除了联接外，可能还需要在 WHERE 子句中添加一些条件，以限制联接中一个或多个表中的相应行。幻灯片示例将输出行限制为部门 ID 等于 20 或 50 的那些行：

例如，要显示雇员 Matos 的部门编号和部门名称，需要在 WHERE 子句中添加一个附加条件。

```
SELECT e.last_name, e.department_id,
       d.department_name
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
   AND last_name = 'Matos';
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1 Matos		50 Shipping

## 联接两个以上的表

**EMPLOYEES**

LAST_NAME	DEPARTMENT_ID
1 King	90
2 Kochhar	90
3 De Haan	90
4 Hunold	60
5 Ernst	60
6 Lorentz	60
7 Mourgos	50
8 Rajs	50
9 Davies	50
10 Matos	50

...

**DEPARTMENTS**

DEPARTMENT_ID	LOCATION_ID
1	10
2	20
3	50
4	60
5	80
6	90
7	110
8	190

**LOCATIONS**

LOCATION_ID	CITY
1	Southlake
2	South San Francisco
3	Seattle
4	Toronto
5	Oxford

要将  $n$  个表联接在一起，最少需要  $n-1$  个联接条件。  
例如，要联接三个表，最少需要两个联接。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 联接两个以上的表

有时可能需要联接两个以上的表。例如，要显示每位雇员的姓氏、部门名称和城市，必须将 EMPLOYEES、DEPARTMENTS 和 LOCATIONS 表联接起来。

```
SELECT e.last_name, d.department_name, l.city
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
1 Abel	Sales	Oxford
2 Davies	Shipping	South San Francisco
3 De Haan	Executive	Seattle
4 Ernst	IT	Southlake
5 Fay	Marketing	Toronto

...

# 非等值联接

**EMPLOYEES**

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	Fay	6000
4	Higgins	12000
5	Gietz	8300
6	King	24000
7	Kochhar	17000
8	De Haan	17000
9	Hunold	9000
10	Ernst	6000
...		
19	Taylor	8600
20	Grant	7000

**JOB\_GRADES**

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

JOB\_GRADES 表定义了每个 GRADE\_LEVEL 的 LOWEST\_SAL 值到 HIGHEST\_SAL 值的范围。因此，GRADE\_LEVEL 列可用于为每位雇员指定等级。



版权所有 © 2010, Oracle。保留所有权利。

## 非等值联接

非等值联接是一个包含非等号运算符的联接条件。

EMPLOYEES 表和 JOB\_GRADES 表之间的关系就是一个非等值联接的示例。EMPLOYEES 表中的 SALARY 列的范围介于 JOB\_GRADES 表的 LOWEST\_SAL 和 HIGHEST\_SAL 列中的值之间。因此，可以根据每位雇员的薪金划分其等级。通过使用等号 (=) 以外的运算符可以实现这一关系。

## 使用非等值联接检索记录

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C
...			

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用非等值联接检索记录

幻灯片示例中创建了一个非等值联接来评估雇员的薪金等级。薪金必须介于任何一对最低薪金和最高薪金之间。

值得注意的是，在执行这一查询时，所有雇员只能出现一次。雇员不能在列表中重复出现。这有以下两个原因：

- 职务等级表中的任何一行都不包含重叠的等级。也就是说，每位雇员的薪金值只能介于薪金等级表中某一行的最低薪金值和最高薪金值之间。
- 所有雇员的薪金都在职务等级表提供的限度之内。也就是说，任何雇员的薪金都不低于 LOWEST\_SAL 列中的最低值，也不高于 HIGHEST\_SAL 列中的最高值。

**注：**可以使用其它条件，如  $\leq$  和  $\geq$ ，但最简单的方法是使用 BETWEEN。请记住，在使用 BETWEEN 条件时，应先指定最低值，后指定最高值。Oracle Server 将 BETWEEN 条件解释为一对 AND 条件。因此，使用 BETWEEN 没有性能优势，只是为了简化逻辑才使用它。

幻灯片示例中指定了表别名，这是出于性能考虑，而不是因为可能出现混淆。

## 使用外部联接返回没有直接匹配的记录

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

**EMPLOYEES**

DEPARTMENT_ID	LAST_NAME
1	10 Whalen
2	20 Hartstein
3	20 Fay
4	110 Higgins
5	110 Gietz
6	90 King
7	90 Kochhar
8	90 De Haan
9	60 Hunold
10	60 Ernst
...	
18	80 Abel
19	80 Taylor

部门 190 中没有雇员。

版权所有 © 2010, Oracle。保留所有权利。

## 使用外部联接返回没有直接匹配的记录

如果某一行不满足联接条件，则查询结果中就不会出现该行。例如，在 EMPLOYEES 表和 DEPARTMENTS 表的等值联接条件下，部门 ID 190 没有出现，这时因为 EMPLOYEES 表中没有具备该部门 ID 的雇员记录。同样，如果有一个雇员的 DEPARTMENT\_ID 设置为 NULL，那么此行也不会出现在等值联接的查询结果中。要返回没有任何雇员的部门记录，或者要返回不属于任何部门的雇员记录，可以使用外部联接。

## 外部联接语法

- 使用外部联接可以查看不符合联接条件的行。
- 外部联接运算符是加号 (+)。

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column (+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column (+);
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 外部联接：语法

如果在联接条件中使用外部联接运算符，则可以返回缺少的行。该运算符是一个用括号括起来的加号 (+)，被放在联接条件中缺少信息的“一侧”。该运算符的作用是创建一个或多个空行，这些行可以与不缺少信息的表中的一行或多行联接在一起。

在该语法中：

`table1.column =` 将表联接（或关联）在一起的条件。

`table2.column (+)` 是外部联接符号，该符号可以放在 WHERE 子句条件的任意一侧，但不能同时放在两侧（请将外部联接符号放在表中没有匹配行的列名的后边）。

## 使用外部联接

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id(+) = d.department_id ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping
9	Hunold	60	IT
10	Ernst	60	IT
<b>...</b>			
19	Gietz	110	Accounting
20	(null)	(null)	Contracting

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 使用外部联接

幻灯片示例中显示了雇员姓氏、部门 ID 和部门名称。Contracting 部门不包含任何雇员。所以在输出中显示为空值。

#### 外部联接限制

- 外部联接运算符只能出现在表达式的一侧，即缺少信息的那一侧。它会返回一个表在另一个表中没有直接匹配项的那些行。
- 包含外部联接的条件不能使用 IN 运算符，也不能通过 OR 运算符链接到另一个条件。

注：Oracle 没有等效的联接语法来对应符合 SQL:1999 的联接语法中的 FULL OUTER JOIN。

## 外部联接：另一个示例

```
SELECT e.last_name, e.department_id, d.department_name  
FROM   employees e, departments d  
WHERE  e.department_id = d.department_id(+);
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

...

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 外部联接：另一个示例

幻灯片示例中的查询会检索 EMPLOYEES 表中的所有行，即使 DEPARTMENTS 表中没有匹配项也是如此。

## 将表联接到自身

**EMPLOYEES (WORKER)**

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

**EMPLOYEES (MANAGER)**

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...

**WORKER 表中的 MANAGER\_ID 等于  
MANAGER 表中的 EMPLOYEE\_ID。**

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 将表联接到自身

有时需要将表联接到自身。要查找每位雇员的经理的姓名，需要将 EMPLOYEES 表联接到自身，即执行自联接。例如，要查找 Lorentz 的经理的姓名，需要执行以下操作：

- 在 EMPLOYEES 表中通过搜索 LAST\_NAME 列来查找 Lorentz。
- 通过搜索 MANAGER\_ID 列查找 Lorentz 的经理编号。Lorentz 的经理编号为 103。
- 通过搜索 LAST\_NAME 列查找 EMPLOYEE\_ID 为 103 的经理的姓名。Hunold 的雇员编号为 103，因此 Hunold 是 Lorentz 的经理。

在这一过程中，您对表进行了两次搜索。第一次是在表中 LAST\_NAME 列中查找 Lorentz，得知其 MANAGER\_ID 值为 103。第二次是搜索 EMPLOYEE\_ID 列以查找 103，然后在 LAST\_NAME 列中找到了 Hunold。

## 自联接：示例

```
SELECT worker.last_name || ' works for '
    || manager.last_name
  FROM employees worker, employees manager
 WHERE worker.manager_id = manager.employee_id ;
```

	WORKER.LAST_NAME  'WORKSFOR'  MANAGER.LAST_NAME
1	Hunold works for De Haan
2	Fay works for Hartstein
3	Gietz works for Higgins
4	Lorentz works for Hunold
5	Ernst works for Hunold
6	Zlotkey works for King
7	Mourgos works for King
8	Kochhar works for King
9	Hartstein works for King
10	De Haan works for King
...	

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

## 自联接：示例

幻灯片示例中将 EMPLOYEES 表联接到自身。为了模拟 FROM 子句中的两个表，我们为 EMPLOYEES 表起了两个别名，分别为 worker 和 manager。

在此示例中，WHERE 子句包含一个联接，该联接表示“雇员的经理编号与经理的雇员编号匹配”。

## 小结

在本附录中，您应该已经学会如何使用 Oracle 专用联接语法来显示多个表中的数据。



版权所有 © 2010, Oracle。保留所有权利。

### 小结

联接表的方式有多种。

### 联接类型

- 等值联接
- 非等值联接
- 外部联接
- 自联接

### 笛卡尔积

笛卡尔积会导致显示行的所有组合。这种情况是因为省略 WHERE 子句造成的。

### 表别名

- 使用表别名可加快对数据库的访问速度
- 表别名有助于缩短 SQL 代码，因而可以节省内存

## 练习 F: 概览

本练习包含以下主题:

- 使用等值联接将表联接起来
- 执行外部联接和自联接
- 添加条件

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

### 练习 F: 概览

本练习旨在为您提供实践经验，帮助您使用 Oracle 联接语法从多个表中提取数据。



---

## 附录 AP

### 附加练习和解答

---

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

## 目录

附加练习 .....	3
练习 1-1 .....	4
练习解答 1-1 .....	12
案例研究 .....	17
练习 2-1 .....	18
练习解答 2-1 .....	26

## 附加练习

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

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

## 练习 1-1

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令和 SQL 函数。

- 1) HR 部门需要查找在 1997 年以后雇用的所有职员的数据。

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	2600
2	144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	2500

- 2) HR 部门需要一个包括领取佣金的雇员的报表。此报表上显示这些雇员的姓氏、职务、薪金和佣金。按薪金的降序对这些数据进行排序。

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Abel	SA_REP	11000	0.3
2	Zlotkey	SA_MAN	10500	0.2
3	Taylor	SA_REP	8600	0.2
4	Grant	SA_REP	7000	0.15

- 3) 为了进行预算，HR 部门需要一个报表，其中包含有关加薪的数据。该报表应显示不领取佣金但薪金增涨 10%（舍入薪金）的雇员。

	New salary
1	The salary of King after a 10% raise is 26400
2	The salary of Kochhar after a 10% raise is 18700
3	The salary of De Haan after a 10% raise is 18700
4	The salary of Hunold after a 10% raise is 9900
5	The salary of Ernst after a 10% raise is 6600
6	The salary of Lorentz after a 10% raise is 4620
7	The salary of Mourgos after a 10% raise is 6380
8	The salary of Rajs after a 10% raise is 3850
9	The salary of Davies after a 10% raise is 3410
10	The salary of Matos after a 10% raise is 2860
11	The salary of Vargas after a 10% raise is 2750
12	The salary of Whalen after a 10% raise is 4840
13	The salary of Hartstein after a 10% raise is 14300
14	The salary of Fay after a 10% raise is 6600
15	The salary of Higgins after a 10% raise is 13200
16	The salary of Gietz after a 10% raise is 9130

## 练习 1-1 (续)

- 4) 创建一个包括雇员及其任职期的报表。此报表上显示所有雇员的姓氏，及其任职的年数和完整月数。按照雇员任职时间长短对报表进行排序。任职时间最长的雇员显示在列表的最顶部。

	LAST_NAME	YEARS	MONTHS
1	King	22	0
2	Whalen	21	9
3	Kochhar	19	9
4	Hunold	19	6
5	Ernst	18	1
6	De Haan	16	6
7	Higgins	15	1
8	Gietz	15	1
9	Rajs	13	8
10	Hartstein	13	4
11	Abel	13	2
12	Davies	12	5
13	Fay	11	10
14	Matos	11	4
15	Taylor	11	3
16	Vargas	11	0
17	Lorentz	10	5
18	Grant	10	1
19	Mourgos	9	7
20	Zlotkey	9	5

- 5) 显示姓氏以字母“J”、“K”、“L”或“M”为开头的雇员。

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos

## 练习 1-1（续）

- 6) 创建一个显示所有雇员的报表，并用“Yes（是）”或“No（否）”指示他们是否领取佣金。在查询中使用 DECODE 表达式。

	LAST_NAME	SALARY	COMMISSION
1	King	24000	No
2	Kochhar	17000	No
3	De Haan	17000	No
4	Hunold	9000	No
5	Ernst	6000	No
6	Lorentz	4200	No
7	Mourgos	5800	No
8	Rajs	3500	No
9	Davies	3100	No
10	Matos	2600	No
11	Vargas	2500	No
12	Zlotkey	10500	Yes
13	Abel	11000	Yes
14	Taylor	8600	Yes
15	Grant	7000	Yes
16	Whalen	4400	No
17	Hartstein	13000	No
18	Fay	6000	No
19	Higgins	12000	No
20	Gietz	8300	No

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令、SQL 函数、联接和组函数。

- 7) 创建一个报表，用于显示在特定位置工作的雇员的部门名称、位置 ID、姓氏、职务和薪金。提示用户输入位置。例如，如果用户输入 1800，则结果如下所示：

	DEPARTMENT_NAME	LOCATION_ID	LAST_NAME	JOB_ID	SALARY
1	Marketing	1800	Hartstein	MK_MAN	13000
2	Marketing	1800	Fay	MK_REP	6000

- 8) 查找姓氏以字母“n”结尾的雇员的数量。请提供两种可能的解答。

	COUNT(*)
1	3

## 练习 1-1（续）

- 9) 创建一个报表，用于显示每个部门的名称、位置和雇员数。确保该报表还包括没有任何雇员的部门。

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	COUNT(E.EMPLOYEE_ID)
1	80	Sales	2500	3
2	110	Accounting	1700	2
3	10	Administration	1700	1
4	60	IT	1400	3
5	20	Marketing	1800	2
6	90	Executive	1700	3
7	50	Shipping	1500	5
8	190	Contracting	1700	0

- 10) HR 部门需要查找部门 10 和部门 20 中的职务。创建一个报表来显示这些部门的职务 ID。

JOB_ID
AD_ASST
MK_MAN
MK_REP

- 11) 创建一个报表，用于显示 Administration 部门和 Executive 部门中的职务。同时显示担任这些职务的雇员数。首先显示雇员数最多的职务。

JOB_ID	FREQUENCY
AD_VP	2
AD_PRES	1
AD_ASST	1

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令、SQL 函数、联接、组函数和子查询。

- 12) 显示在前半月（当月 16 日之前）雇用的所有雇员。

	LAST_NAME	HIRE_DATE
1	De Haan	13-JAN-93
2	Hunold	03-JAN-90
3	Lorentz	07-FEB-99
4	Matos	15-MAR-98
5	Vargas	09-JUL-98
6	Abel	11-MAY-96
7	Higgins	07-JUN-94
8	Gietz	07-JUN-94

## 练习 1-1 (续)

- 13) 创建一个报表，用于显示所有雇员的以下信息：姓氏、薪金和用千美元表示的薪金。

	LAST_NAME	SALARY	THOUSANDS
1	King	24000	24
2	Kochhar	17000	17
3	De Haan	17000	17
4	Hunold	9000	9
5	Ernst	6000	6
6	Lorentz	4200	4
7	Mourgos	5800	5
8	Rajs	3500	3
9	Davies	3100	3
10	Matos	2600	2
11	Vargas	2500	2
12	Zlotkey	10500	10
13	Abel	11000	11
14	Taylor	8600	8
15	Grant	7000	7
16	Whalen	4400	4
17	Hartstein	13000	13
18	Fay	6000	6
19	Higgins	12000	12
20	Gietz	8300	8

- 14) 显示其经理的薪金高于 \$15,000 的所有雇员。显示以下数据：雇员姓名、经理姓名、经理薪金和经理的薪金等级。

	LAST_NAME	MANAGER	SALARY	GRADE_LEVEL
1	De Haan	King	24000 E	
2	Hartstein	King	24000 E	
3	Higgins	Kochhar	17000 E	
4	Hunold	De Haan	17000 E	
5	Kochhar	King	24000 E	
6	Mourgos	King	24000 E	
7	Whalen	Kochhar	17000 E	
8	Zlotkey	King	24000 E	

## 练习 1-1 (续)

- 15) 显示所有部门的部门编号、名称、雇员数量和平均薪金以及在每个部门中工作的雇员的姓名、薪金和职务。

	DEPARTMENT_ID	DEPARTMENT_NAME	EMPLOYEES	Avg_Sal	LAST_NAME	SALARY	JOB_ID
1	10	Administration	1	4400.00	Whalen	4400	AD_ASST
2	20	Marketing	2	9500.00	Hartstein	13000	MK_MAN
3	20	Marketing	2	9500.00	Fay	6000	MK_REP
4	50	Shipping	5	3500.00	Davies	3100	ST_CLERK
5	50	Shipping	5	3500.00	Matos	2600	ST_CLERK
6	50	Shipping	5	3500.00	Rajs	3500	ST_CLERK
7	50	Shipping	5	3500.00	Mourgos	5800	ST_MAN
8	50	Shipping	5	3500.00	Vargas	2500	ST_CLERK
9	60	IT	3	6400.00	Hunold	9000	IT_PROG
10	60	IT	3	6400.00	Lorentz	4200	IT_PROG
11	60	IT	3	6400.00	Ernst	6000	IT_PROG
12	80	Sales	3	10033.33	Zlotkey	10500	SA_MAN
13	80	Sales	3	10033.33	Taylor	8600	SA_REP
14	80	Sales	3	10033.33	Abel	11000	SA_REP
15	90	Executive	3	19333.33	Kochhar	17000	AD_VP
16	90	Executive	3	19333.33	De Haan	17000	AD_VP
17	90	Executive	3	19333.33	King	24000	AD_PRES
18	110	Accounting	2	10150.00	Gietz	8300	AC_ACCOUNT
19	110	Accounting	2	10150.00	Higgins	12000	AC_MGR
20	(null)	(null)	0	No average	Grant	7000	SA_REP

- 16) 创建一个报表，用于显示平均薪金最高的部门编号和该部门的最低薪金。

DEPARTMENT_ID	MIN(SALARY)
1	90
	17000

- 17) 创建一个报表，用于显示没有销售代表的部门。在输出中包括部门编号、部门名称、经理 ID 和位置。

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	90 Executive	100	1700
6	110 Accounting	205	1700
7	190 Contracting	(null)	1700

## 练习 1-1 (续)

18) 为 HR 部门创建以下统计报表：其中包括符合以下条件的每个部门的部门编号、部门名称和雇员数量：

a) 雇员数少于 3 人的部门：

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1		10 Administration	1
2		110 Accounting	2
3		20 Marketing	2

b) 雇员数最多的部门：

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1		50 Shipping	5

c) 雇员数最少的部门：

	DEPARTMENT_ID	DEPARTMENT_NAME	COUNT(*)
1		10 Administration	1

19) 创建一个报表，用于显示所有雇员在各自部门中的雇员编号、姓氏、薪金、部门编号和平均薪金。

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	SALARY	AVG(S.SALARY)
1	149	Zlotkey	80	10500	10033.3333333333...
2	174	Abel	80	11000	10033.3333333333...
3	144	Vargas	50	2500	3500
4	101	Kochhar	90	17000	19333.3333333333...
5	100	King	90	24000	19333.3333333333...
6	103	Hunold	60	9000	6400
7	142	Davies	50	3100	3500
8	205	Higgins	110	12000	10150
9	104	Ernst	60	6000	6400
10	143	Matos	50	2600	3500
11	102	De Haan	90	17000	19333.3333333333...
12	107	Lorentz	60	4200	6400
13	141	Rajs	50	3500	3500
14	200	Whalen	10	4400	4400
15	202	Fay	20	6000	9500
16	176	Taylor	80	8600	10033.3333333333...
17	201	Hartstein	20	13000	9500
18	206	Gietz	110	8300	10150
19	124	Mourgos	50	5800	3500

## 练习 1-1 (续)

20) 显示在一周中雇用雇员数最多那一天所雇用的所有雇员。

	LAST_NAME	DAY
1	Ernst	TUESDAY
2	Mourgos	TUESDAY
3	Rajs	TUESDAY
4	Taylor	TUESDAY
5	Higgins	TUESDAY
6	Gietz	TUESDAY

21) 根据雇员的雇用日期，创建一个周年纪念日概览表。按升序对周年纪念日进行排序。

	LAST_NAME	BIRTHDAY
1	Hunold	January 03
2	De Haan	January 13
3	Davies	January 29
4	Zlotkey	January 29
5	Lorentz	February 07
6	Hartstein	February 17
7	Matos	March 15
8	Taylor	March 24
9	Abel	May 11
10	Ernst	May 21
11	Grant	May 24
12	Higgins	June 07
13	Gietz	June 07
14	King	June 17
15	Vargas	July 09
16	Fay	August 17
17	Whalen	September 17
18	Kochhar	September 21
19	Rajs	October 17
20	Mourgos	November 16

## 练习解答 1-1

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令和 SQL 函数。

- 1) HR 部门需要查找在 1997 年以后雇用的所有职员的数据。

```
SELECT *
FROM   employees
WHERE  job_id = 'ST_CLERK'
AND    hire_date > '31-DEC-1997';
```

- 2) HR 部门需要一个包括领取佣金的雇员的报表。此报表上显示这些雇员的姓氏、职务、薪金和佣金。按薪金的降序对这些数据进行排序。

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees
WHERE  commission_pct IS NOT NULL
ORDER BY salary DESC;
```

- 3) 为了进行预算，HR 部门需要一个报表，其中包含有关加薪的数据。该报表应显示不领取佣金但薪金增涨 10%（舍入薪金）的雇员。

```
SELECT 'The salary of '||last_name||' after a 10% raise is '
      || ROUND(salary*1.10) "New salary"
FROM   employees
WHERE  commission_pct IS NULL;
```

- 4) 创建一个包括雇员及其任职期的报表。此报表上显示所有雇员的姓氏，及其任职的年数和完整月数。按照雇员任职期对报表进行排序。任职时间最长的雇员显示在列表的最顶部。

```
SELECT last_name,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date) / 12) YEARS,
       TRUNC(MOD(MONTHS_BETWEEN(SYSDATE, hire_date), 12))
             MONTHS
  FROM employees
 ORDER BY years DESC, months desc;
```

- 5) 显示姓氏以字母“J”、“K”、“L”或“M”为开头的雇员。

```
SELECT last_name
  FROM employees
 WHERE SUBSTR(last_name, 1, 1) IN ('J', 'K', 'L', 'M');
```

- 6) 创建一个显示所有雇员的报表，并用“Yes（是）”或“No（否）”指示他们是否领取佣金。在查询中使用 DECODE 表达式。

```
SELECT last_name, salary,
       decode(commission_pct, NULL, 'No', 'Yes') commission
  FROM employees;
```

## 练习解答 1-1（续）

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令、SQL 函数、联接和组函数。

- 7) 创建一个报表，用于显示在特定位置工作的雇员的部门名称、位置 ID、姓名、职务和薪金。提示用户输入位置。

- a) 提示时输入 1800 作为 location\_id。

```
SELECT d.department_name, d.location_id, e.last_name,
e.job_id, e.salary
FROM employees e, departments d
WHERE e.department_id = d.department_id
AND d.location_id = &location_id;
```

- 8) 查找姓氏以字母“n”结尾的雇员的数量。请提供两种可能的解答。

```
SELECT COUNT(*)
FROM employees
WHERE last_name LIKE '%n';
--or
SELECT COUNT(*)
FROM employees
WHERE SUBSTR(last_name, -1) = 'n';
```

- 9) 创建一个报表，用于显示每个部门的名称、位置和雇员数。确保该报表还包括没有任何雇员的部门。

```
SELECT d.department_id, d.department_name,
d.location_id, COUNT(e.employee_id)
FROM employees e RIGHT OUTER JOIN departments d
ON e.department_id = d.department_id
GROUP BY d.department_id, d.department_name, d.location_id;
```

- 10) HR 部门需要查找部门 10 和部门 20 中的职务。创建一个报表来显示这些部门的职务 ID。

```
SELECT DISTINCT job_id
FROM employees
WHERE department_id IN (10, 20);
```

- 11) 创建一个报表，用于显示 Administration 部门和 Executive 部门中的职务。同时还显示担任这些职务的雇员数。首先显示雇员数最多的职务。

```
SELECT e.job_id, count(e.job_id) FREQUENCY
FROM employees e JOIN departments d
ON e.department_id = d.department_id
WHERE d.department_name IN ('Administration', 'Executive')
GROUP BY e.job_id
ORDER BY FREQUENCY DESC;
```

在讨论完以下主题之后，就可以做这些补充练习了：基本 SQL SELECT 语句、基本 SQL Developer 命令、SQL 函数、联接、组函数和子查询。

## 练习解答 1-1 (续)

12) 显示在前半月（当月 16 日之前）雇用的所有雇员。

```
SELECT last_name, hire_date
FROM   employees
WHERE  TO_CHAR(hire_date, 'DD') < 16;
```

13) 创建一个报表，用于显示所有雇员的以下信息：姓氏、薪金和用千美元表示的薪金。

```
SELECT last_name, salary, TRUNC(salary, -3)/1000  Thousands
FROM   employees;
```

14) 显示其经理的薪金高于 \$15,000 的所有雇员。显示以下数据：雇员姓名、经理姓名、经理薪金和经理的薪金等级。

```
SELECT e.last_name, m.last_name manager, m.salary,
j.grade_level
FROM   employees e JOIN employees m
ON     e.manager_id = m.employee_id
JOIN   job_grades j
ON     m.salary BETWEEN j.lowest_sal AND j.highest_sal
AND    m.salary > 15000;
```

15) 显示所有部门的部门编号、名称、雇员数量和平均薪金以及在每个部门中工作的雇员的姓名、薪金和职务。

```
SELECT d.department_id, d.department_name,
       count(e1.employee_id) employees,
       NVL(TO_CHAR(AVG(e1.salary), '99999.99'), 'No
average') avg_sal,
       e2.last_name, e2.salary, e2.job_id
  FROM departments d RIGHT OUTER JOIN employees e1
  ON      d.department_id = e1.department_id
  RIGHT OUTER JOIN employees e2
  ON      d.department_id = e2.department_id
 GROUP BY d.department_id, d.department_name, e2.last_name,
e2.salary,
       e2.job_id
 ORDER BY d.department_id, employees;
```

16) 创建一个报表，用于显示平均薪金最高的部门编号和该部门的最低薪金。

```
SELECT department_id, MIN(salary)
  FROM employees
 GROUP BY department_id
 HAVING AVG(salary) = (SELECT MAX(AVG(salary))
                        FROM   employees
                        GROUP BY department_id);
```

## 练习解答 1-1 (续)

- 17) 创建一个报表，用于显示没有销售代表的部门。在输出中包括部门编号、部门名称和位置。

```
SELECT *
FROM   departments
WHERE  department_id NOT IN(SELECT department_id
                             FROM employees
                             WHERE job_id = 'SA_REP'
                             AND department_id IS NOT NULL);
```

- 18) 为 HR 部门创建以下统计报表：其中包括符合以下条件的每个部门的部门编号、部门名称和雇员数量：

a) 雇员数少于 3 人的部门：

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) < 3;
```

b) 雇员数最多的部门：

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                     FROM employees
                     GROUP BY department_id);
```

c) 雇员数最少的部门：

```
SELECT d.department_id, d.department_name, COUNT(*)
FROM   departments d JOIN employees e
ON     d.department_id = e.department_id
GROUP BY d.department_id, d.department_name
HAVING COUNT(*) = (SELECT MIN(COUNT(*))
                     FROM employees
                     GROUP BY department_id);
```

## 练习解答 1-1 (续)

- 19) 创建一个报表，用于显示所有雇员在各自部门中的雇员编号、姓氏、薪金、部门编号和平均薪金。

```
SELECT e.employee_id, e.last_name, e.department_id, e.salary,
AVG(s.salary)
FROM employees e JOIN employees s
ON e.department_id = s.department_id
GROUP BY e.employee_id, e.last_name, e.department_id,
e.salary;
```

- 20) 显示在一周中雇用雇员数最多那一天所雇用的所有雇员。

```
SELECT last_name, TO_CHAR(hire_date, 'DAY') day
FROM employees
WHERE TO_CHAR(hire_date, 'Day') =
(SELECT TO_CHAR(hire_date, 'Day')
FROM employees
GROUP BY TO_CHAR(hire_date, 'Day')
HAVING COUNT(*) = (SELECT MAX(COUNT(*))
FROM employees
GROUP BY TO_CHAR(hire_date,
'Day')));
```

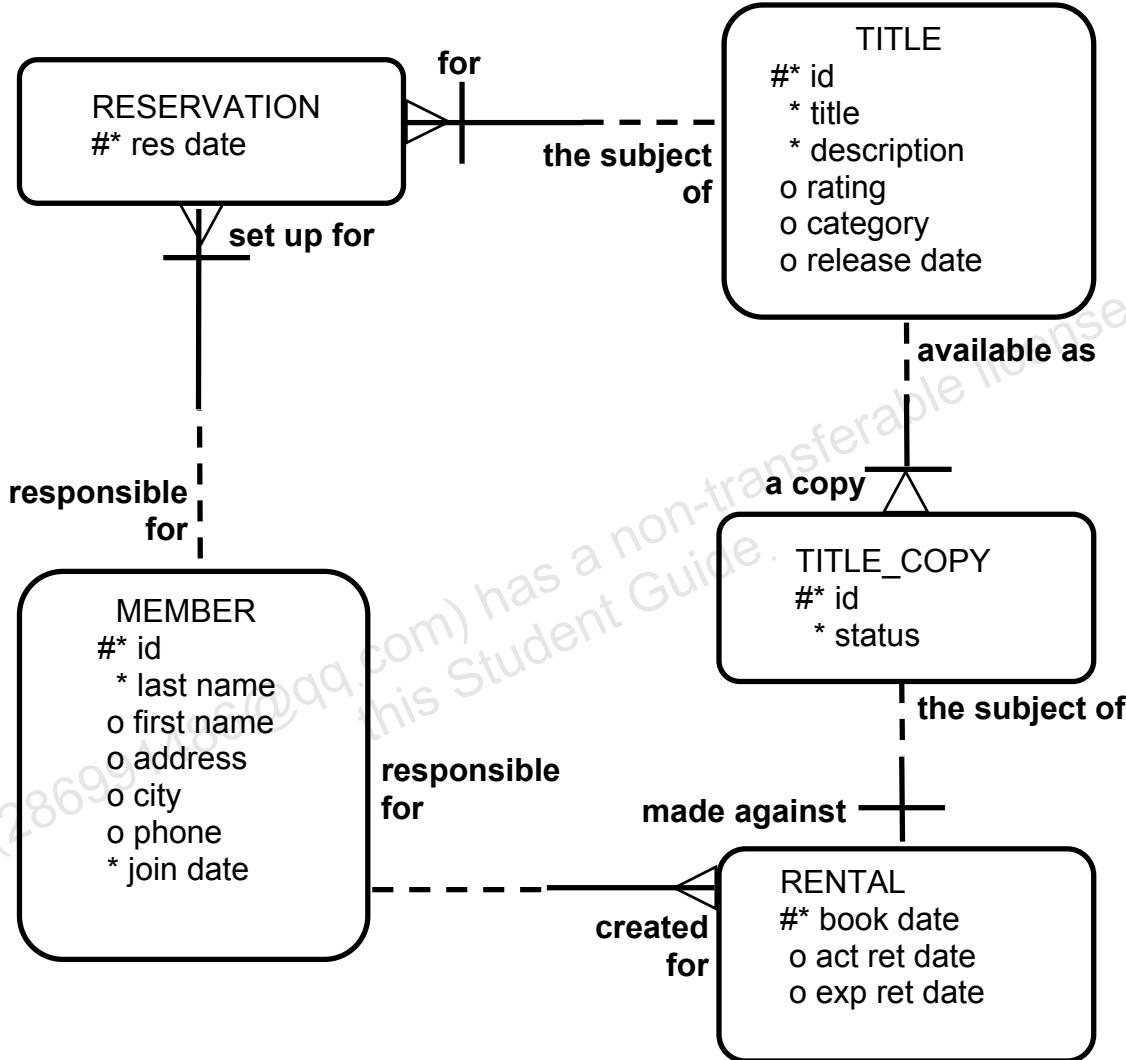
- 21) 根据雇员的雇用日期，创建一个周年纪念日概览表。按升序对周年纪念日进行排序。

```
SELECT last_name, TO_CHAR(hire_date, 'Month DD') BIRTHDAY
FROM employees
ORDER BY TO_CHAR(hire_date, 'DDD');
```

## 案例研究

在本案例研究中，您将为录像带应用程序创建一组数据库表。创建这些表后，可以在录像带存储数据库中插入、更新和删除记录并生成一个报表。该数据库只包含重要的表。

下图提供了录像带应用程序的实体和属性：



**注：**如果要创建表，可以在 SQL Developer 中执行 `buildtab.sql` 脚本中的命令。如果要删除表，可以在 SQL Developer 中执行 `dropvid.sql` 脚本中的命令。然后，可以通过在 SQL Developer 中执行 `buildvid.sql` 脚本中的命令来创建和填充这些表。

所有这三个 SQL 脚本都位于 `/home/oracle/labs/sql1/labs` 文件夹中。

- 如果使用 `buildtab.sql` 脚本创建表，应从步骤 4 开始。
- 如果使用 `dropvid.sql` 脚本删除录像带表，应从步骤 1 开始。
- 如果使用 `buildvid.sql` 脚本创建和填充表，应从步骤 6(b) 开始。

## 练习 2-1

- 1) 根据以下表实例图表创建表。选择适当的数据类型，而且一定要添加完整性约束条件。

a) 表名: MEMBER

列名	MEMBER_ID	LAST_NAME	FIRST_NAME	ADDRESS	CITY	PHONE	JOIN_DATE
键类型	PK						
空值/唯一	NN、U	NN					NN
默认值							系统日期
数据类型	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
长度	10	25	25	100	30	15	

b) 表名: TITLE

列名	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
键类型	PK					
空值/唯一	NN、U	NN	NN			
检查				G、PG、R、NC17、NR	DRAMA、COMEDY、ACTION、CHILD、SCIFI、DOCUMENTARY	
数据类型	NUMBER	VARCHAR2	VARCHAR2	VARCHAR2	VARCHAR2	DATE
长度	10	60	400	4	20	

## 练习 2-1 (续)

c) 表名: TITLE\_COPY

列名	COPY_ID	TITLE_ID	STATUS
键类型	PK	PK、FK	
空值/唯一	NN、U	NN、U	NN
检查			AVAILABLE、 DESTROYED、 RENTED、 RESERVED
FK 引用表		TITLE	
FK 引用列		TITLE_ID	
数据类型	NUMBER	NUMBER	VARCHAR2
长度	10	10	15

d) 表名: RENTAL

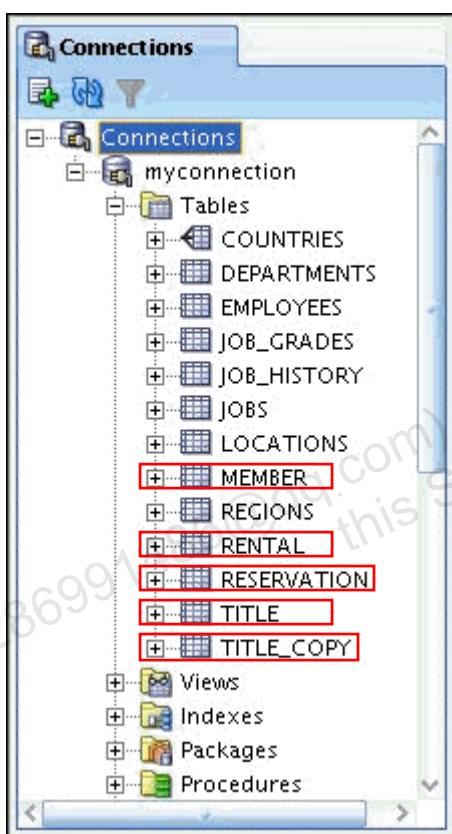
列名	BOOK_DATE	MEMBER_ID	COPY_ID	ACT_RET_DATE	EXP_RET_DATE	TITLE_ID
键类型	PK	PK、FK1	PK、FK2			PK、FK2
默认值	系统日期				系统日期 + 2 天	
FK 引用表		MEMBER	TITLE_COPY			TITLE_COPY
FK 引用列		MEMBER_ID	COPY_ID			TITLE_ID
数据类型	DATE	NUMBER	NUMBER	DATE	DATE	NUMBER
长度		10	10			10

## 练习 2-1 (续)

e) 表名: RESERVATION

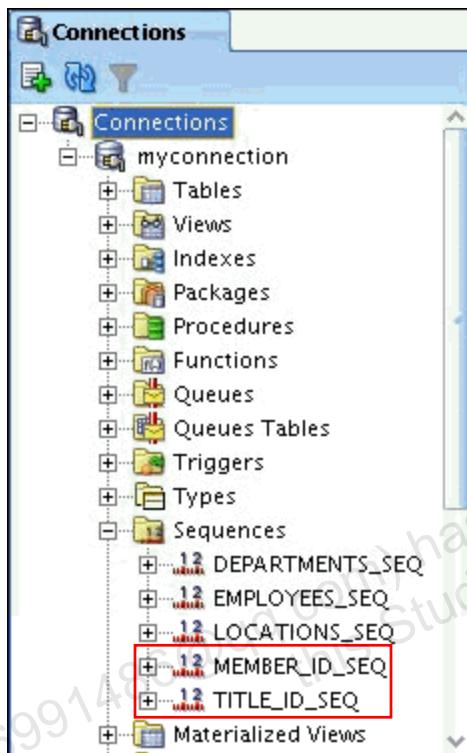
列名	RES_DATE	MEMBER_ID	TITLE_ID
键类型	PK	PK、FK1	PK、FK2
空值/唯一	NN、U	NN、U	NN
FK 引用表		MEMBER	TITLE
FK 引用列		MEMBER_ID	TITLE_ID
数据类型	DATE	NUMBER	NUMBER
长度		10	10

- 2) 通过在 SQL Developer 的连接导航器中进行检查来验证是否已正确创建表。



## 练习 2-1（续）

- 3) 通过创建序列来唯一地标识 MEMBER 表和 TITLE 表中的每一行。
- MEMBER 表的成员编号：从 101 开始，不允许对这些值进行高速缓存。将该序列命名为 MEMBER\_ID\_SEQ。
  - TITLE 表的标题编号：从 92 开始，不允许对这些值进行高速缓存。将此序列命名为 TITLE\_ID\_SEQ。
  - 验证在 SQL Developer 的连接导航器中是否存在序列。



- 4) 在这些表中添加数据。为要添加的每组数据创建一个脚本。

- 向 TITLE 表中添加电影标题。通过编写脚本来输入电影信息。将这些语句保存在名为 lab\_apcs\_4a.sql 的脚本中。使用序列来唯一地标识每个标题。采用 DD-MON-YYYY 格式输入公映日期。请记住，必须对字符字段中的单引号进行特殊处理。验证您添加的信息。

	AZ	TITLE
1		'Willie and Christmas Too'
2		'Alien Again'
3		'The Glob'
4		'My Day Off'
5		'Miracles on Ice'
6		'Soda Gang'

## 练习 2-1 (续)

Title	Description	Rating	Category	Release date
Willie and Christmas Too	All of Willie's friends make a Christmas list for Santa, but Willie has yet to add his own wish list.	G	CHILD	05-OCT-1995
Alien Again	Yet another installation of science fiction history. Can the heroine save the planet from the alien life form?	R	SCIFI	19-MAY-1995
The Glob	A meteor crashes near a small American town and unleashes carnivorous goo in this classic.	NR	SCIFI	12-AUG-1995
My Day Off	With a little luck and a lot of ingenuity, a teenager skips school for a day in New York.	PG	COMEDY	12-JUL-1995
Miracles on Ice	A six-year-old has doubts about Santa Claus, but she discovers that miracles really do exist.	PG	DRAMA	12-SEP-1995
Soda Gang	After discovering a cache of drugs, a young couple find themselves pitted against a vicious gang.	NR	ACTION	01-JUN-1995

b) 在 MEMBER 表中添加数据。将插入语句保存在名为 lab\_apcs\_4b.sql 的脚本中。执行该脚本中的命令。务必使用序列添加成员编号。

First_Name	Last_Name	Address	City	Phone	Join_Date
Carmen	Velasquez	283 King Street	Seattle	206-899-6666	08-MAR-1990
LaDoris	Ngao	5 Modrany	Bratislava	586-355-8882	08-MAR-1990
Midori	Nagayama	68 Via Centrale	Sao Paolo	254-852-5764	17-JUN-1991
Mark	Quick-to-See	6921 King Way	Lagos	63-559-7777	07-APR-1990
Audry	Ropeburn	86 Chu Street	Hong Kong	41-559-87	18-JAN-1991
Molly	Urguhart	3035 Laurier	Quebec	418-542-9988	18-JAN-1991

## 练习 2-1 (续)

c) 在 TITLE\_COPY 表中添加下列电影副本:

注: 此练习可以使用 TITLE\_ID 编号。

Title	Copy_Id	Status	Title	Copy_Id
Willie and Christmas Too	1	AVAILABLE	Willie and Christmas Too	1
Alien Again	1	AVAILABLE	Alien Again	1
	2	RENTED		2
The Glob	1	AVAILABLE	The Glob	1
My Day Off	1	AVAILABLE	My Day Off	1
	2	AVAILABLE		2
	3	RENTED		3
Miracles on Ice	1	AVAILABLE	Miracles on Ice	1
Soda Gang	1	AVAILABLE	Soda Gang	1

d) 在 RENTAL 表中添加下列租赁信息:

注: 标题编号可能因序列号不同而有所不同。

Title_Id	Copy_Id	Member_Id	Book_date	Exp_Ret_Date
92	1	101	三天前	一天前
93	2	101	一天前	一天之后
95	3	102	两天前	当天
97	1	106	四天前	两天前

## 练习 2-1（续）

- 5) 创建一个名为 TITLE\_AVAIL 的视图来显示电影标题、每个副本的可用性以及预计的归还日期（如果已租出）。从该视图查询所有行。按标题对结果进行排序。

注：结果可能会有所不同。

TITLE	COPY_ID	STATUS	EXP_RET_DATE
1 Alien Again	1	AVAILABLE	(null)
2 Alien Again	2	RENTED	15-JUL-09
3 Miracles on Ice	1	AVAILABLE	(null)
4 My Day Off	1	AVAILABLE	(null)
5 My Day Off	2	AVAILABLE	(null)
6 My Day Off	3	RENTED	16-JUL-09
7 Soda Gang	1	AVAILABLE	14-JUL-09
8 The Glob	1	AVAILABLE	(null)
9 Willie and Christmas Too	1	AVAILABLE	15-JUL-09

- 6) 更改这些表中的数据。

- a) 添加新标题。电影是“Interstellar Wars”，等级为 PG，类别为科幻电影。公映日期为 07-JUL-77。说明为“Futuristic interstellar action movie. Can the rebels save the humans from the evil empire（未来的星际动作电影。反抗者能将人类从罪恶帝国中拯救出来吗）？”务必为两个副本添加标题副本记录。
- b) 输入两个预订。一个是 Carmen Velasquez 的预订，他希望租借“Interstellar Wars”。另一个是 Mark Quick-to-See 的预订，他希望租借“Soda Gang”。

## 练习 2-1 (续)

7) 修改以下表之一。

- a) 运行位于 /home/oracle/labs/sql1/labs 文件夹中的脚本 lab\_apcs\_7a.sql, 将 PRICE 列添加到 TITLE 表以便记录录像带的采购价格。验证您所做的修改。

DESCRIBE title		
Name	Null	Type
TITLE_ID	NOT NULL	NUMBER(10)
TITLE	NOT NULL	VARCHAR2(60)
DESCRIPTION	NOT NULL	VARCHAR2(400)
RATING		VARCHAR2(4)
CATEGORY		VARCHAR2(20)
RELEASE_DATE		DATE
PRICE		NUMBER(8, 2)

Title	Price
Willie and Christmas Too	25
Alien Again	35
The Glob	35
My Day Off	35
Miracles on Ice	30
Soda Gang	35
Interstellar Wars	29

- b) 创建名为 lab\_apcs\_7b.sql 的脚本, 其中包含更新语句, 这些更新语句会根据上面的列表更新每部录像带的价格。运行脚本中的命令。

注: 此练习可以使用 TITLE\_ID 编号。

- 8) 创建一个报表, 其中包含每个客户租赁录像带的历史记录。请务必包括客户名称、租赁的电影、租赁日期和租赁期限。计算该报告期间所有客户的租赁数量总和。将生成该报表的命令保存在名为 lab\_apcs\_8.sql 的脚本文件中。

注: 结果可能会有所不同。

MEMBER	TITLE	BOOK_DATE	DURATION
1 Carmen Velasquez	Willie and Christmas Too	13-JUL-09	1
2 Carmen Velasquez	Alien Again	15-JUL-09	(null)
3 LaDoris Ngao	My Day Off	14-JUL-09	(null)
4 Molly Urguhart	Soda Gang	12-JUL-09	2

## 练习解答 2-1

- 1) 根据以下表实例图表创建表。选择适当的数据类型，而且一定要添加完整性约束条件。

a) 表名: MEMBER

```
CREATE TABLE member
  (member_id          NUMBER(10)
    CONSTRAINT member_member_id_pk PRIMARY KEY,
   last_name          VARCHAR2(25)
    CONSTRAINT member_last_name_nn NOT NULL,
   first_name         VARCHAR2(25),
   address            VARCHAR2(100),
   city               VARCHAR2(30),
   phone              VARCHAR2(15),
   join_date          DATE DEFAULT SYSDATE
    CONSTRAINT member_join_date_nn NOT NULL);
```

b) 表名: TITLE

```
CREATE TABLE title
  (title_id           NUMBER(10)
    CONSTRAINT title_title_id_pk PRIMARY KEY,
   title              VARCHAR2(60)
    CONSTRAINT title_title_nn NOT NULL,
   description        VARCHAR2(400)
    CONSTRAINT title_description_nn NOT NULL,
   rating             VARCHAR2(4)
    CONSTRAINT title_rating_ck CHECK
      (rating IN ('G', 'PG', 'R', 'NC17', 'NR')),
   category           VARCHAR2(20)
    CONSTRAINT title_category_ck CHECK
      (category IN ('DRAMA', 'COMEDY', 'ACTION',
                    'CHILD', 'SCIFI', 'DOCUMENTARY')),
   release_date        DATE);
```

c) 表名: TITLE\_COPY

```
CREATE TABLE title_copy
  (copy_id            NUMBER(10),
   title_id           NUMBER(10)
    CONSTRAINT title_copy_title_if_fk REFERENCES
   title(title_id),
   status              VARCHAR2(15)
    CONSTRAINT title_copy_status_nn NOT NULL
    CONSTRAINT title_copy_status_ck CHECK (status IN
      ('AVAILABLE', 'DESTROYED', 'RENTED', 'RESERVED')),
   CONSTRAINT title_copy_copy_id_title_id_pk
   PRIMARY KEY (copy_id, title_id));
```

## 练习解答 2-1 (续)

d) 表名: RENTAL

```
CREATE TABLE rental
    (book_date      DATE DEFAULT SYSDATE,
     member_id      NUMBER(10)
        CONSTRAINT rental_member_id_fk REFERENCES
member(member_id),
     copy_id        NUMBER(10),
     act_ret_date   DATE,
     exp_ret_date   DATE DEFAULT SYSDATE + 2,
     title_id       NUMBER(10),
        CONSTRAINT rental_book_date_copy_title_pk
        PRIMARY KEY (book_date, member_id, copy_id, title_id),
        CONSTRAINT rental_copy_id_title_id_fk
        FOREIGN KEY (copy_id, title_id)
        REFERENCES title_copy(copy_id, title_id));
```

e) 表名: RESERVATION

```
CREATE TABLE reservation
    (res_date        DATE,
     member_id       NUMBER(10)
        CONSTRAINT reservation_member_id REFERENCES
member(member_id),
     title_id        NUMBER(10)
        CONSTRAINT reservation_title_id REFERENCES
title(title_id),
        CONSTRAINT reservation_resdate_mem_tit_pk PRIMARY KEY
        (res_date, member_id, title_id));
```

- 2) 通过在 SQL Developer 的连接导航器中进行检查来验证是否已正确创建表。
  - a) 在连接导航器中, 展开 “Connections (连接) > myconnection > Tables (表) ”。
- 3) 通过创建序列来唯一地标识 MEMBER 表和 TITLE 表中的每一行。
  - a) MEMBER 表的成员编号: 从 101 开始, 不允许对这些值进行高速缓存。将该序列命名为 MEMBER\_ID\_SEQ。

```
CREATE SEQUENCE member_id_seq
START WITH 101
NOCACHE;
```
  - b) TITLE 表的标题编号: 从 92 开始, 不允许对这些值进行高速缓存。将此序列命名为 TITLE\_ID\_SEQ。

```
CREATE SEQUENCE title_id_seq
START WITH 92
NOCACHE;
```

## 练习解答 2-1 (续)

- c) 验证在 SQL Developer 的连接导航器中是否存在序列。
- i) 在连接导航器中（假定 myconnection 节点已展开）展开“Sequences (序列)”。
- 4) 在这些表中添加数据。为要添加的每组数据创建一个脚本。
- a) 向 TITLE 表中添加电影标题。通过编写脚本来输入电影信息。将这些语句保存在名为 lab\_apcs\_4a.sql 的脚本中。使用序列来唯一地标识每个标题。采用 DD-MON-YYYY 格式输入公映日期。请记住，必须对字符字段中的单引号进行特殊处理。验证您添加的信息。

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Willie and Christmas Too',
        'All of Willie''s friends make a Christmas list for
        Santa, but Willie has yet to add his own wish list.',
        'G', 'CHILD', TO_DATE('05-OCT-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id , title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Alien Again', 'Yet another
        installment of science fiction history. Can the
        heroine save the planet from the alien life form?',
        'R', 'SCIFI', TO_DATE( '19-MAY-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'The Glob', 'A meteor crashes
        near a small American town and unleashes carnivorous
        goo in this classic.', 'NR', 'SCIFI',
        TO_DATE( '12-AUG-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'My Day Off', 'With a little
        luck and a lot ingenuity, a teenager skips school
        for
        a day in New York.', 'PG', 'COMEDY',
        TO_DATE( '12-JUL-1995','DD-MON-YYYY'))
/
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Miracles on Ice', 'A six-
        year-old has      doubts about Santa Claus, but she discovers
        that miracles really do exist.', 'PG', 'DRAMA',
        TO_DATE('12-SEP-1995','DD-MON-YYYY'))
/

```

## 练习解答 2-1 (续)

```

INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES      (title_id_seq.NEXTVAL, 'Soda Gang', 'After
discovering a cache of drugs, a young couple find themselves
pitted against a vicious gang.', 'NR', 'ACTION', TO_DATE('01-
JUN-1995','DD-MON-YYYY'))
/
COMMIT
/
SELECT  title
FROM    title;

```

- b) 在 MEMBER 表中添加数据。将插入语句保存在名为 lab\_apcs\_4b.sql 的脚本中。执行该脚本中的命令。务必使用序列添加成员编号。

```

SET VERIFY OFF
INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Carmen', 'Velasquez',
        '283 King Street', 'Seattle', '206-899-6666',
        TO_DATE('08-MAR-1990',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'LaDoris', 'Ngao',
        '5 Modrany', 'Bratislava', '586-355-8882',
        TO_DATE('08-MAR-1990',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Midori', 'Nagayama',
        '68 Via Centrale', 'Sao Paolo', '254-852-5764',
        TO_DATE('17-JUN-1991',
                'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Mark', 'Quick-to-See',
        '6921 King Way', 'Lagos', '63-559-7777', TO_DATE('07-
APR-1990',
                'DD-MM-YYYY'))
/

```

## 练习解答 2-1 (续)

```

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Audry', 'Ropeburn',
        '86 Chu Street', 'Hong Kong', '41-559-87',
        TO_DATE('18-JAN-1991',
                 'DD-MM-YYYY'))
/

INSERT INTO member(member_id, first_name, last_name,
                   address, city, phone, join_date)
VALUES (member_id_seq.NEXTVAL, 'Molly', 'Urguhart',
        '3035 Laurier', 'Quebec', '418-542-9988', TO_DATE('18-
JAN-1991',
                 'DD-MM-YYYY'));
/

COMMIT
SET VERIFY ON

```

c) 在 TITLE\_COPY 表中添加下列电影副本:

注: 此练习可以使用 TITLE\_ID 编号。

```

INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 92, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 93, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 93, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 94, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (2, 95, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (3, 95, 'RENTED')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 96, 'AVAILABLE')
/
INSERT INTO title_copy(copy_id, title_id, status)
VALUES (1, 97, 'AVAILABLE')
/

```

## 练习解答 2-1 (续)

- d) 在 RENTAL 表中添加下列租赁信息：

注：标题编号可能因序列号不同而有所不同。

```
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (92, 1, 101, sysdate-3, sysdate-1, sysdate-2)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (93, 2, 101, sysdate-1, sysdate-1, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (95, 3, 102, sysdate-2, sysdate, NULL)
/
INSERT INTO rental(title_id, copy_id, member_id,
                   book_date, exp_ret_date, act_ret_date)
VALUES (97, 1, 106, sysdate-4, sysdate-2, sysdate-2)
/
COMMIT
/
```

- 5) 创建一个名为 TITLE\_AVAIL 的视图来显示电影标题、每个副本的可用性以及预计的归还日期（如果已租出）。从该视图查询所有行。按标题对结果进行排序。

注：结果可能会有所不同。

```
CREATE VIEW title_avail AS
SELECT t.title, c.copy_id, c.status, r.exp_ret_date
FROM title t JOIN title_copy c
ON t.title_id = c.title_id
FULL OUTER JOIN rental r
ON c.copy_id = r.copy_id
AND c.title_id = r.title_id;

SELECT *
FROM title_avail
ORDER BY title, copy_id;
```

## 练习解答 2-1 (续)

6) 更改这些表中的数据。

- a) 添加新标题。电影是“Interstellar Wars”，等级为 PG，类别为科幻电影。公映日期为 07-JUL-77。说明为“Futuristic interstellar action movie. Can the rebels save the humans from the evil empire (未来的星际动作电影。反抗者能将人类从罪恶帝国中拯救出来吗)？”务必为两个副本添加标题副本记录。

```
INSERT INTO title(title_id, title, description, rating,
                  category, release_date)
VALUES (title_id_seq.NEXTVAL, 'Interstellar Wars',
        'Futuristic interstellar action movie. Can the
        rebels save the humans from the evil empire?',
        'PG', 'SCIFI', '07-JUL-77')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (1, 98, 'AVAILABLE')
/
INSERT INTO title_copy (copy_id, title_id, status)
VALUES (2, 98, 'AVAILABLE')
```

- b) 输入两个预订。一个是 Carmen Velasquez 的预订，他希望租借“Interstellar Wars”。另一个是 Mark Quick-to-See 的预订，他希望租借“Soda Gang”。

```
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 101, 98)
/
INSERT INTO reservation (res_date, member_id, title_id)
VALUES (SYSDATE, 104, 97)
```

7) 修改以下表之一。

- a) 运行位于 /home/oracle/labs/sql1/labs 文件夹中的脚本 lab\_apcs\_7a.sql，将 PRICE 列添加到 TITLE 表以便记录录像带的采购价格。验证您所做的修改。

```
ALTER TABLE title
ADD (price NUMBER(8,2));
DESCRIBE title
```

## 练习解答 2-1 (续)

- b) 创建名为 lab\_apcs\_7b.sql 的脚本，其中包含更新语句，这些更新语句会根据上面的列表更新每部录像带的价格。运行脚本中的命令。

注：此练习可以使用 TITLE\_ID 编号。

```
SET ECHO OFF
SET VERIFY OFF
UPDATE title
SET    price = &price
WHERE  title_id = &title_id;
SET VERIFY OFF
SET ECHO OFF
```

- 8) 创建一个报表，其中包含每个客户租赁录像带的历史记录。请务必包括客户名称、租赁的电影、租赁日期和租赁期限。计算该报告期间所有客户的租赁数量总和。将生成该报表的命令保存在名为 lab\_apcs\_8.sql 的脚本文件中。

注：结果可能会有所不同。

```
SELECT  m.first_name||' '||m.last_name MEMBER, t.title,
        r.book_date, r.act_ret_date - r.book_date DURATION
FROM    member m
JOIN   rental r
ON     r.member_id = m.member_id
JOIN   title t
ON     r.title_id = t.title_id
ORDER BY member;
```



# 第5章

**A**

ALL 运算符 7-19, 8-3, 8-6, 8-7, 8-12, 8-16, 8-17, 8-18, 8-21, 8-24, 8-27, 8-29, 8-30  
 ALTER TABLE 语句 10-35, 10-36  
 AND 运算符 2-16, 2-21, F-12  
 ANY 运算符 7-3, 7-9, 7-18, 7-21  
 AVG 5-3, 5-5, 5-7, 5-8, 5-11, 5-12, 5-15, 5-16, 5-20, 5-23, 5-25, 5-26, 5-28, 7-13

**B**

BETWEEN 运算符 2-10  
 BI Publisher I-14  
 比较运算符 2-5, 2-8, 2-9, 7-10, 7-17, 9-16, 9-22  
 别名 1-3, 1-10, 1-11, 1-17, 1-18, 1-19, 1-20, 1-21, 1-22, 1-26, 1-30, 2-6, 2-8, 2-23,  
 2-24, 4-25, 5-14, 5-27, 6-7, 6-14, 6-15, 6-17, 6-25, 6-38, 8-28, 10-33, C-8, F-8,  
 F-10, F-11, F-15, F-21, F-22

**C**

CASE 表达式 4-32, 4-37, 4-38, 4-39, 4-40, 4-44  
 CHECK 约束条件 9-8, 10-3, 10-6, 10-11, 10-15, 10-27, 10-31, 10-34, 10-37  
 COALESCE 函数 4-33, 4-34, 4-35  
 COUNT 函数 5-9  
 CREATE TABLE 语句 9-13, 10-3, 10-6, 10-7, 10-11, 10-15, 10-31, 10-32, 10-34,  
 10-37, 10-40, 10-41  
 CURRENT\_DATE 3-25, 9-10  
 CURRVAL 10-9, 10-27  
 查询 1-4, 1-9, 1-18, 1-19, 1-22, 1-25, 1-31, 2-2, 2-5, 2-23, 2-25, 2-27, 2-31, 2-33, 2-38,  
 3-2, 3-7, 3-27, 3-35, 4-21, 4-34, 4-38, 4-39, 4-40, 4-45, 5-15, 5-19, 5-20, 5-28, 5-29,  
 6-6, 6-7, 6-14, 6-17, 6-25, 6-27, 6-29, 6-30, 6-31, 7-1, 7-2, 7-3, 7-4, 7-5, 7-6, 7-7,  
 7-8, 7-9, 7-10, 7-11, 7-12, 7-13, 7-14, 7-15, 7-16, 7-17, 7-18, 7-19, 7-20, 7-21, 7-22,  
 7-23, 7-24, 7-25, 8-2, 8-4, 8-5, 8-6, 8-13, 8-16, 8-17, 8-19, 8-20, 8-22, 8-25, 8-26,  
 8-28, 8-29, 8-30, 8-31, 9-13, 9-16, 9-18, 9-19, 9-22, 9-24, 9-34, 9-44, 9-45, 9-48,  
 10-3, 10-4, 10-6, 10-10, 10-11, 10-13, 10-15, 10-27, 10-31, 10-32, 10-33, 10-34,  
 10-37, 10-40, 10-41, C-5, C-18, C-29, D-3, D-4, D-6, D-14, D-16, D-17, D-18, F-7,  
 F-8, F-10, F-15, F-16, F-19, I-4, I-5, I-14, I-16, I-28  
 创建数据库连接 C-7, C-8, C-9

**D**

DBMS 9-44, D-16, D-18, I-2, I-17, I-18, I-24, I-26, I-37  
 DECODE 函数 4-37, 4-39, 4-40, 4-41, 4-42, 4-44, 5-6  
 DEFAULT 选项 10-3, 10-6, 10-9, 10-11, 10-15, 10-31, 10-34, 10-37  
 DELETE 语句 9-3, 9-14, 9-20, 9-22, 9-23, 9-24, 9-25, 9-26, 9-40, 9-41, 9-43  
 DESCRIBE 命令 1-3, 1-11, 1-17, 1-20, 1-26, 1-27, 1-28, 9-8, 10-10, 10-33, C-11, D-8  
 DISTINCT 关键字 1-3, 1-11, 1-17, 1-20, 1-25, 1-26, 5-3, 5-10, 5-12, 5-25  
 DUAL 表 3-18  
 等值联接 6-2, 6-3, 6-8, 6-12, 6-20, 6-23, 6-24, 6-25, 6-36, F-2, F-9, F-10, F-11, F-14, F-15, F-22  
 笛卡尔积 6-2, 6-3, 6-6, 6-8, 6-20, 6-23, 6-26, 6-32, 6-33, 6-34, 6-35, 6-36, F-2, F-4, F-5, F-22  
 读一致性 9-3, 9-14, 9-20, 9-26, 9-40, 9-41, 9-42, 9-43  
 对象关系 I-2, I-16, I-17

**F**

FOR UPDATE 子句 9-3, 9-14, 9-20, 9-26, 9-40, 9-43, 9-44, 9-45, 9-46, 9-48  
 方案 10-2, 10-5, 10-8, 10-28, 10-40, B-2, C-3, C-5, C-7, C-8, C-10, C-13, E-5, E-6, I-2, I-3, I-4, I-6, I-8, I-15, I-27, I-31, I-32, I-34, I-38  
 非等值联接 6-2, 6-3, 6-8, 6-20, 6-23, 6-24, 6-25, 6-26, 6-32, 6-36, 6-37, F-2, F-6, F-14, F-15, F-22

**G**

GROUP BY 子句 5-2, 5-3, 5-12, 5-13, 5-14, 5-15, 5-16, 5-18, 5-19, 5-22, 5-23, 5-25, 5-26, 5-27, 5-28, 7-14  
 格式样式 3-30, 3-32, 4-11, 4-12, 4-14, 4-15, 4-16, 4-19, 4-20, 4-43  
 关键字 1-5, 1-8, 1-21, 6-9, 10-25, C-23, D-5  
 关系数据库 9-44, I-2, I-3, I-4, I-8, I-15, I-16, I-17, I-18, I-19, I-26, I-27, I-28, I-29, I-31, I-34, I-37  
 国际标准组织 I-29

**H**

HAVING 子句 5-2, 5-3, 5-12, 5-20, 5-21, 5-22, 5-23, 5-24, 5-25, 5-28, 5-29, 7-3, 7-5, 7-9, 7-13, 7-16, 7-21, 7-24  
 函数 2-5, 2-7, 3-1, 3-2, 3-3, 3-4, 3-5, 3-6, 3-7, 3-8, 3-9, 3-10, 3-11, 3-12, 3-13, 3-14, 3-15, 3-16, 3-17, 3-18, 3-19, 3-20, 3-21, 3-24, 3-25, 3-27, 3-28, 3-29, 3-30, 3-31, 3-32, 3-33, 4-1, 4-2, 4-3, 4-4, 4-7, 4-9, 4-10, 4-20, 4-21, 4-22, 4-23, 4-24, 4-25, 4-26, 4-27, 4-28, 4-36, 4-44, 4-45, 5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-7, 5-8, 5-9, 5-11, 5-12, 5-13, 5-14, 5-15, 5-19, 5-20, 5-22, 5-25, 5-26, 5-27, 5-28, 5-29, 7-3, 7-9, 7-12, 7-16, 7-21, 7-24, 8-25, 9-10, 9-11, 10-9, 10-27, C-5, C-6, C-24, C-25, C-26, E-8, E-15, I-4, I-5

**I**

IN 运算符 2-11, 7-22, F-18

INSERT 语句 9-3, 9-6, 9-9, 9-13, 9-14, 9-20, 9-26, 9-40, 9-43, 10-7, 10-10, 10-18

INTERSECT 运算符 8-3, 8-4, 8-5, 8-7, 8-12, 8-18, 8-19, 8-20, 8-21, 8-24, 8-27, 8-30, 8-31

INTERVAL YEAR TO MONTH 10-14

**J**

Java C-4, C-8, E-3, E-12, E-14, E-15, E-16, I-9, I-12, I-37

集合运算符 8-1, 8-2, 8-3, 8-4, 8-5, 8-6, 8-7, 8-12, 8-18, 8-21, 8-24, 8-27, 8-29, 8-31, I-5

交叉联接 6-5, 6-35, 6-37

**K**

空值 1-3, 1-11, 1-15, 1-16, 1-17, 1-20, 1-21, 1-26, 2-8, 2-14, 2-23, 2-24, 4-28, 4-29, 4-30, 4-31, 4-32, 4-38, 4-40, 4-44, 5-6, 5-9, 5-10, 5-11, 7-3, 7-9, 7-15, 7-16, 7-21, 7-22, 9-8, 9-9, 10-9, 10-20, 10-21, 10-23, 10-26, F-18, I-26

**L**

LENGTH 3-10, 3-11, 3-14, 3-15, 3-34, 4-32

LIKE 运算符 2-12

LPAD 3-10, 3-14

连接运算符 1-3, 1-11, 1-17, 1-20, 1-21, 1-26, 2-20

联接表 6-11, 6-37, F-6, F-7, F-21, F-22

列别名 1-3, 1-10, 1-11, 1-17, 1-18, 1-19, 1-20, 1-22, 1-26, 1-30, 2-6, 2-24, 4-25, 5-14, 5-27, 6-7, 10-33, F-8

**M**

MAX 5-3, 5-5, 5-7, 5-8, 5-12, 5-23, 5-25, 5-26, 5-28, B-3, I-32

MIN 3-23, 5-3, 5-5, 5-7, 5-8, 5-12, 5-25, 5-28, 7-12, 7-13, 7-14, 7-17, 8-3, 8-4, 8-5, 8-7, 8-12, 8-18, 8-21, 8-22, 8-23, 8-24, 8-27, 8-30, 8-31, 10-27, 11-13, 11-25, 11-31, B-3, C-7, C-26, I-32

MINUS 运算符 8-3, 8-5, 8-7, 8-12, 8-18, 8-21, 8-22, 8-23, 8-24, 8-27, 8-30, 8-31

MOD 函数 3-20

美国国家标准协会 4-4, 6-5, I-28

命名 10-3, 10-5, 10-6, 10-11, 10-15, 10-17, 10-31, 10-34, 10-35, 10-37

**N**

NEXTVAL 10-9, 10-27  
 NOT NULL 约束条件 1-27, 10-18, 10-20, 10-21, 10-32  
 NOT 运算符 2-3, 2-18, 2-19, 2-22, 2-26, 2-34, 7-19  
 NULL 条件 2-3, 2-14  
 NULLIF 函数 4-32  
 NVL 函数 4-29, 4-30, 4-33, 4-44, 5-11  
 NVL2 函数 4-31

**O**

OLTP I-11, I-16  
 ON DELETE CASCADE 10-26  
 ON DELETE SET NULL 10-26  
 ON 子句 6-3, 6-5, 6-6, 6-8, 6-16, 6-17, 6-18, 6-19, 6-20, 6-22, 6-23, 6-26, 6-28, 6-32, 8-17  
 OR 运算符 2-15, 2-17, F-18  
 Oracle Database 11g 3-25, 7-8, 10-10, 10-14, 10-36, 10-38, C-4, I-2, I-3, I-4, I-8, I-9, I-10, I-11, I-14, I-15, I-27, I-30, I-31, I-34, I-35, I-36, I-37  
 Oracle Enterprise Manager Grid Control I-13, I-37  
 Oracle Fusion Middleware I-12, I-13, I-37  
 Oracle Server 1-13, 2-11, 2-13, 2-23, 4-4, 4-5, 4-6, 4-19, 4-20, 4-38, 5-22, 5-28, 6-7, 6-25, 7-13, 7-17, 7-24, 8-4, 8-6, 9-4, 9-8, 9-27, 9-33, 9-34, 9-39, 9-42, 9-44, 9-48, 10-5, 10-16, 10-17, 10-22, 10-23, D-3, D-14, F-8, F-15, I-2, I-16, I-37  
 Oracle SQL Developer C-2, C-3, C-4, I-2, I-30, I-38  
 ORDBMS I-2, I-37  
 ORDER BY 子句 2-3, 2-19, 2-22, 2-23, 2-24, 2-25, 2-26, 2-28, 2-34, 2-38, 2-39, 3-7, 5-15, 5-16, 5-18, 5-28, 8-3, 8-5, 8-7, 8-12, 8-18, 8-21, 8-24, 8-27, 8-28, 8-29, 8-30, 10-13

**P**

PRIMARY KEY 约束条件 10-23  
 排序 2-1, 2-2, 2-3, 2-19, 2-22, 2-23, 2-24, 2-25, 2-26, 2-34, 2-38, 2-39, I-5

**Q**

q 运算符 1-24  
 其它引号 (q) 运算符 1-24

**R**

RDBMS 9-44, I-2, I-18, I-24, I-26, I-37  
 REFERENCES 1-15, 9-32, 10-25, 10-26, 10-28, C-6, C-19, C-31  
 REPLACE 3-10, 3-14, D-17  
 ROUND 函数 3-18, 3-19, 4-26  
 ROUND 和 TRUNC 函数 3-32  
 RPAD 3-10, 3-14  
 RR 日期格式 3-23, 3-24, 4-22  
 日期 1-10, 1-12, 1-21, 1-22, 2-7, 2-11, 2-12, 2-24, 2-28, 2-31, 3-2, 3-3, 3-4, 3-5, 3-8, 3-9, 3-16, 3-18, 3-19, 3-21, 3-22, 3-23, 3-24, 3-25, 3-26, 3-27, 3-28, 3-29, 3-30, 3-31, 3-32, 3-34, 3-35, 4-5, 4-11, 4-12, 4-13, 4-14, 4-15, 4-16, 4-20, 4-21, 4-22, 4-25, 4-29, 4-43, 4-44, 5-8, 7-8, 8-8, 8-9, 9-7, 9-10, 9-11, 10-10, 10-12, 10-14, I-17, I-21  
 日期时间数据类型 10-14

**S**

SELECT 语句 1-1, 1-2, 1-3, 1-4, 1-5, 1-7, 1-11, 1-17, 1-20, 1-22, 1-23, 1-26, 1-29, 1-30, 2-6, 2-9, 2-10, 2-12, 2-21, 2-23, 2-28, 2-32, 2-35, 2-38, 2-39, 3-2, 3-13, 4-2, 4-44, 5-9, 5-15, 5-16, 5-17, 5-18, 5-19, 5-20, 6-2, 6-7, 7-2, 7-5, 7-8, 7-10, 7-11, 7-24, 7-25, 8-3, 8-6, 8-7, 8-12, 8-18, 8-19, 8-20, 8-21, 8-22, 8-24, 8-25, 8-26, 8-27, 9-3, 9-14, 9-20, 9-23, 9-26, 9-34, 9-40, 9-41, 9-42, 9-43, 9-44, 9-45, 9-48, 10-32, D-2, F-2, F-7, F-8, I-4, I-5  
 SET VERIFY ON 2-36  
 SQL Developer 1-6, 1-8, 1-9, 1-10, 1-15, 1-18, 1-27, 2-28, 2-29, 2-30, 2-31, 2-33, 2-35, 2-36, 6-17, 9-4, 9-22, 9-28, 9-32, 9-33, 9-44, 10-9, 10-41, C-1, C-2, C-3, C-4, C-5, C-6, C-7, C-9, C-10, C-11, C-13, C-17, C-23, C-24, C-26, C-27, C-29, C-30, C-31, F-10, I-2, I-7, I-9, I-30, I-35, I-38  
 SUBSTR 3-10, 3-11, 3-14, 3-15, 3-34, 4-25  
 SYSDATE 函数 3-24, 3-25, 9-10  
 实体关系 B-3, I-21, I-22  
 使用片段 C-24, C-25  
 事务处理 9-2, 9-3, 9-4, 9-14, 9-20, 9-26, 9-27, 9-28, 9-30, 9-31, 9-32, 9-33, 9-34, 9-35, 9-39, 9-40, 9-42, 9-43, 9-49, 10-38, C-16, I-11, I-29  
 视图 10-8, 10-10, 10-38, C-10, E-6, E-13

**S (续)**

**数据库** 1-2, 1-4, 1-15, 1-30, 2-2, 3-13, 3-22, 3-25, 3-26, 3-27, 5-15, 5-18, 6-38, 9-3, 9-4, 9-10, 9-14, 9-20, 9-26, 9-27, 9-28, 9-32, 9-34, 9-35, 9-40, 9-41, 9-42, 9-43, 9-44, 9-45, 9-46, 10-2, 10-3, 10-4, 10-5, 10-6, 10-7, 10-8, 10-10, 10-11, 10-15, 10-31, 10-33, 10-34, 10-37, 10-38, 10-40, 10-41, C-2, C-3, C-4, C-5, C-6, C-7, C-8, C-9, C-10, C-12, C-13, C-15, C-16, C-17, C-19, C-27, C-28, C-31, D-4, D-5, D-6, D-7, D-16, D-19, E-2, E-4, E-5, E-6, E-8, E-11, E-15, E-18, F-7, F-10, F-22, I-2, I-3, I-4, I-8, I-10, I-11, I-13, I-15, I-16, I-17, I-18, I-19, I-20, I-21, I-26, I-27, I-28, I-29, I-31, I-34, I-37, I-38

**数据库事务处理** 9-3, 9-14, 9-20, 9-26, 9-27, 9-28, 9-40, 9-43

**数据类型** 1-12, 1-27, 1-28, 3-4, 3-33, 4-4, 4-29, 4-31, 4-44, 5-6, 5-8, 6-9, 6-11, 8-13, 8-19, 8-22, 9-8, 9-13, 10-2, 10-3, 10-6, 10-11, 10-12, 10-13, 10-14, 10-15, 10-31, 10-34, 10-37, 10-40, C-11, D-8, D-9, E-14, I-9, I-16

**数字函数** 3-3, 3-8, 3-9, 3-16, 3-17, 3-21, 3-28, 3-30

**顺序** 1-7, 1-13, 1-14, 2-2, 2-11, 2-20, 2-23, 2-24, 2-25, 2-39, 4-24, 5-8, 5-18, 5-27, 5-28, 8-2, 8-4, 8-5, 8-29, 9-7, 9-8, I-26

**算术表达式** 1-3, 1-11, 1-12, 1-13, 1-16, 1-17, 1-20, 1-21, 1-26, 2-5

**算术运算符** 1-12, 1-13, 2-20, 3-26, 3-27, 3-34

**索引** 10-4, 10-8, 10-10, 10-22, 10-23, 10-36, 10-38, C-10, C-30, E-6

**T**

**TO\_CHAR** 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-11, 4-16, 4-17, 4-18, 4-19, 4-22, 4-23, 4-25, 4-26, 4-27, 4-34, 4-36, 4-44, 4-45, 8-25

**TO\_DATE** 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-20, 4-21, 4-22, 4-23, 4-27, 4-36, 4-44, 4-45, 9-11

**TO\_NUMBER** 4-2, 4-3, 4-7, 4-8, 4-9, 4-10, 4-20, 4-21, 4-23, 4-27, 4-36, 4-43, 4-44

**TRIM** 3-10, 3-11, 3-14

**TRUNC** 3-17, 3-19, 3-29, 3-30, 3-32, 3-34, 4-42, 9-3, 9-14, 9-20, 9-25, 9-26, 9-40, 9-43, 9-47, 9-48, I-29

**替代变量** 2-3, 2-19, 2-22, 2-26, 2-27, 2-28, 2-31, 2-32, 2-34, 2-36, 2-38, 2-39, 9-12, C-16

**& 替代变量** 2-2, 2-38, 2-39, 3-14

**同义词** 1-25, 1-27, 7-18, 10-4, 10-8, 10-38, D-8, I-22, I-23

**U**

**UNION ALL** 8-3, 8-4, 8-5, 8-6, 8-7, 8-12, 8-16, 8-17, 8-18, 8-21, 8-24, 8-27, 8-29, 8-30

**UNION 运算符** 8-13, 8-14, 8-15, 8-25, 8-26, 8-30, 8-31

**UNIQUE 约束条件** 10-21, 10-22

**UPDATE 语句** 9-3, 9-14, 9-16, 9-17, 9-18, 9-19, 9-20, 9-26, 9-40, 9-43, 9-44, 10-36

**USING 子句** 6-3, 6-5, 6-8, 6-11, 6-13, 6-14, 6-15, 6-18, 6-20, 6-23, 6-26, 6-32

**V**

VARIANCE 5-5, 5-8, 5-28  
 VERIFY 命令 2-3, 2-19, 2-22, 2-26, 2-34, 2-36

**W**

WHERE 子句 2-3, 2-4, 2-5, 2-6, 2-7, 2-8, 2-9, 2-11, 2-15, 2-19, 2-22, 2-26, 2-27, 2-28, 2-31, 2-32, 2-34, 2-37, 2-38, 2-39, 3-13, 5-9, 5-14, 5-15, 5-18, 5-20, 5-21, 5-22, 5-27, 5-28, 6-10, 6-14, 6-16, 6-19, 6-37, 7-2, 7-5, 7-13, 7-14, 7-15, 7-22, 8-5, 9-16, 9-17, 9-23, 11-38, F-4, F-7, F-10, F-12, F-17, F-21, F-22

唯一标识符 I-22, I-23

伪列 3-18, 8-25, 11-3, 11-14, 11-15, 11-16, 11-22, 11-27, 11-28, 11-33, 11-40

文字 1-3, 1-11, 1-15, 1-17, 1-20, 1-22, 1-23, 1-24, 1-26, 2-5, 2-12, 2-13, 3-14, 4-11, 4-14, 4-31, 4-32, 4-38, 8-26, 10-9

**X**

XML C-7, C-9, C-27, C-29, C-32, E-3, E-12, I-9, I-14, I-37

显式数据类型转换 4-3, 4-4, 4-7, 4-8, 4-9, 4-10, 4-23, 4-27, 4-36

序列 2-23, 9-7, 10-4, 10-8, 11-2, 11-3, 11-4, 11-22, 11-23, 11-24, 11-25, 11-26, 11-27, 11-29, 11-30, 11-31, 11-32, 11-33, 11-34, 11-39, 11-40, 11-41, 11-42, 11-45, 11-46

选择 1-4, 1-5, 1-6, 1-7, 1-15, 1-24, 1-27, 1-30, 1-31, 2-4, 2-12, 2-16, 2-17, 2-21, 2-39, 3-13, 3-18, 3-25, 3-34, 4-20, 5-14, 5-27, 5-29, 6-9, 7-5, 7-15, 7-20, 8-6, 8-17, 9-32, 9-33, 10-33, 11-8, 11-10, 11-17, C-5, C-6, C-8, C-9, C-10, C-12, C-13, C-14, C-15, C-17, C-19, C-20, C-21, C-23, C-24, C-28, C-29, C-30, C-31, D-7, E-6, E-7, E-8, E-9, E-11, E-12, E-13

**Y**

隐式数据类型转换 4-4, 4-5, 4-6

映射 1-4

优先级规则 1-13, 1-14, 2-3, 2-19, 2-20, 2-21, 2-22, 2-26, 2-34

约束条件 1-15, 9-4, 9-8, 9-21, 9-25, 10-2, 10-3, 10-6, 10-11, 10-13, 10-15, 10-16, 10-17, 10-18, 10-19, 10-20, 10-21, 10-22, 10-23, 10-24, 10-25, 10-26, 10-27, 10-29, 10-30, 10-31, 10-32, 10-34, 10-37, 10-39, 10-40, 11-8, 11-9, 11-17, 11-19, 11-21, 11-34, 11-36, 11-38, 11-39, 11-44, C-10, E-6, I-16, I-37

**Z**

执行 SQL C-2, C-3, C-13, C-15, C-16, C-17, C-18, C-33, D-2, D-6, D-19, E-2, E-7, E-18

执行语句 1-6, 1-9, 2-29, 6-17, 10-41, E-7, F-10

只读表 10-3, 10-6, 10-11, 10-15, 10-31, 10-34, 10-36, 10-37

重复行 5-9, 8-6, 8-13, 8-15, 8-16, 8-17, 8-30, 1-25

属性 I-21, I-22, I-23, I-26

## Z (续)

转换函数 3-8, 3-10, 3-12, 3-13, 3-30, 4-1, 4-2, 4-4, 4-9, 4-44, 8-25, I-5

子查询 7-3, 7-4, 7-5, 7-6, 7-7, 7-8, 7-9, 7-10, 7-11, 7-12, 7-13, 7-14, 7-15, 7-16, 7-17, 7-18, 7-19, 7-20, 7-21, 7-22, 7-23, 7-24, 7-25, 8-5, 9-13, 9-16, 9-18, 9-19, 9-22, 9-24, 10-3, 10-6, 10-11, 10-13, 10-15, 10-31, 10-32, 10-33, 10-34, 10-37, 10-40, 11-8, 11-9, 11-10, 11-12, 11-27, 11-28

子查询中的组函数 7-3, 7-9, 7-16, 7-21

字符串 1-3, 1-11, 1-17, 1-20, 1-21, 1-22, 1-23, 1-24, 1-26, 2-7, 2-16, 2-17, 3-12, 3-14, 4-14, 4-17, 4-19, 4-20, 4-43

自联接 6-2, 6-3, 6-8, 6-20, 6-21, 6-22, 6-23, 6-26, 6-32, 6-36, 6-37, 6-39, F-2, F-6, F-20, F-21, F-22, F-23

组函数 3-6, 5-1, 5-2, 5-3, 5-4, 5-5, 5-6, 5-11, 5-12, 5-13, 5-14, 5-15, 5-16, 5-19, 5-20, 5-22, 5-23, 5-25, 5-26, 5-27, 5-28, 5-29, 7-3, 7-9, 7-12, 7-16, 7-21, 7-24, 11-13, 11-14, 11-15, 11-16, I-5

