

Oracle Database 11g: SQL 基础 I

学生指南第 1 册

D49996CN20

版本 2.0

2010 年 5 月

D67005

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
SERVEROUTPUT 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

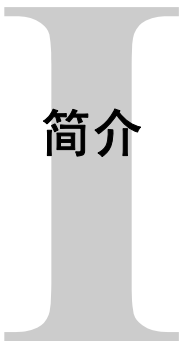
练习 F：概览 F-23

附录 AP：附加练习和解答

索引

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.



简介

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：

- 阐述本课程的目标
- 列出 Oracle Database 11g 的功能
- 从理论和实际实施两方面讨论关系数据库
- 描述 RDBMS 和对象关系数据库管理系统 (ORDBMS) 的 Oracle Server 实施
- 确定可用于本课程的开发环境
- 描述在本课程中使用的数据库和方案

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

本课程将对关系数据库管理系统 (RDBMS) 和对象关系数据库管理系统 (ORDBMS) 进行介绍。此外，本课程还将介绍用于执行 SQL 语句、进行格式设置和报表处理的 Oracle SQL Developer 和 SQL*Plus 开发环境。

课程安排

- 课程目标、课程安排以及本课程中使用的附录
- Oracle Database 11g 以及相关产品概览
- 关系数据库管理概念和术语概览
- SQL 及其开发环境简介
- 本课程中使用的 HR 方案和表
- Oracle Database 11g 的文档和其它资源

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

学完本课程后，应能完成以下工作：

- 识别 Oracle Database 11g 的主要组件
- 使用 SELECT 语句从表中检索行数据和列数据
- 创建一些报表，在其中对数据进行限制和排序
- 利用 SQL 函数生成和检索定制数据
- 运行复杂查询从多个表中检索数据
- 运行数据操纵语言 (DML) 语句来更新 Oracle Database 11g 中的数据
- 运行数据定义语言 (DDL) 语句来创建和管理方案对象

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

本课程将介绍 Oracle Database 11g 数据库技术。在本课程中，您将学到关系数据库的基本概念以及功能强大的 SQL 编程语言。本课程还介绍了一些基本 SQL 技能，利用这些技能您可以针对一个和多个表编写查询，处理表数据，创建数据库对象以及对元数据进行查询。

课程安排

- 第一天:
 - 简介
 - 使用 SQL SELECT 语句检索数据
 - 对数据进行限制和排序
 - 使用单行函数定制输出
 - 使用转换函数和条件表达式
- 第二天:
 - 使用组函数报告聚集数据
 - 使用联接显示多个表中的数据
 - 使用子查询来解决查询
 - 使用集合运算符

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程安排

- 第三天：
 - 处理数据
 - 使用 DDL 语句创建和管理表
 - 创建其它方案对象

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

本课程使用的附录

- 附录 A: 练习和解答
- 附录 B: 表说明
- 附录 C: 使用 SQL Developer
- 附录 D: 使用 SQL*Plus
- 附录 E: 使用 JDeveloper
- 附录 F: Oracle 联接语法
- 附录 AP: 附加练习和解答

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程安排

- 课程目标、课程安排以及本课程中使用的附录
- **Oracle Database 11g 以及相关产品概览**
- 关系数据库管理概念和术语概览
- SQL 及其开发环境简介
- 本课程中使用的 HR 方案和表
- Oracle Database 11g 的文档和其它资源

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Database 11g: 重点领域



基础结构网格

信息管理

应用程序开发

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Database 11g: 重点领域

Oracle Database 11g 在以下几个重点领域中提供了大量功能:

- **基础结构网格:** 借助 Oracle 的基础结构网格技术, 可以实现低成本服务器和存储的共享, 从而构建在易管理性、高可用性及性能方面都能提供最优质服务的系统。Oracle Database 11g 整合并扩展了网格计算的优点。除了充分利用网格计算外, Oracle Database 11g 还具有独一无二的更改保证功能, 可用一种可控制的、经济高效的方式管理更改。
- **信息管理:** Oracle Database 11g 扩展了内容管理、信息集成及信息生命周期管理等领域的现有信息管理功能。Oracle 可为高级数据类型提供内容管理, 这些数据类型包括扩展标记语言 (XML)、文本、空间、多媒体、医学图像和语义技术等等。
- **应用程序开发:** Oracle Database 11g 能够使用和管理所有主要的应用程序开发环境, 如 PL/SQL、Java/JDBC、.NET 和 Windows、PHP、SQL Developer 以及 Application Express。

Oracle Database 11g



易管理性

高可用性

性能

安全性

信息集成

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Database 11g

对于需要对业务应用程序进行全天候快速安全访问的用户，组织需要支持多个兆兆字节的信息。数据库系统必须可靠，而且无论发生何种故障，都能够快速恢复。Oracle Database 11g 在以下功能领域方面经过精心设计，有助于组织轻松管理基础结构网格并提供高质量的服务：

- **易管理性：**通过使用某些更改保证、管理自动化以及故障诊断功能，数据库管理员 (DBA) 可以提高工作效率、降低成本、最大限度地减少错误并提高服务质量。有助于改进管理的有用功能包括数据库重放工具、SQL 性能分析器以及自动 SQL 优化工具。
- **高可用性：**通过使用高可用性功能，可以降低宕机时间和数据丢失的风险。这些功能改进了联机操作，并可提高数据库升级的速度。

Oracle Database 11g (续)

- **性能:** 通过使用 SecureFiles、联机事务处理 (OLTP) 压缩、Real Application Clusters (RAC) 优化、结果高速缓存等功能, 可以大大提高数据库的性能。借助 Oracle Database 11g, 组织可以管理可伸缩的大型事务处理和数据仓库系统, 这些系统可使用低成本模块化存储提供快速数据访问。
- **安全性:** 借助 Oracle Database 11g, 组织可通过独一无二的安全配置、数据加密和掩码以及高级审计功能来保护其信息。通过使用行业标准接口, Oracle Database 11g 可提供安全、可伸缩的平台, 可用于对所有类型的信息进行可靠而快速的访问。
- **信息集成:** Oracle Database 11g 具有许多可以更好地在企业范围内集成数据的功能。它还支持高级信息生命周期管理功能。这有助于管理数据库中不断变化的数据。

Oracle Fusion Middleware

基于标准且经客户检验的先进软件产品的组合，其中涵盖了从 Java EE 和开发人员工具到集成服务、业务智能、协作和内容管理的一系列工具和服务



版权所有 © 2010, Oracle. 保留所有权利。

Oracle Fusion Middleware

Oracle Fusion Middleware 是一个有效集成的综合性产品系列，为开发、部署和管理面向服务的体系结构 (SOA) 提供全面支持。SOA 推动了模块化业务服务的开发，这类服务可以很容易地集成和重复使用，从而降低了开发和维护成本，并可提供更优质的服务。借助 Oracle Fusion Middleware 的可插入体系结构，可以充分利用在现有应用程序、系统或技术上进行的投资。其强大的核心技术可将计划断电或计划外断电所造成的业务中断降到最低。Oracle Fusion Middleware 系列的部分产品包括：

- 企业应用程序服务器：Oracle Application Server
- 集成和进程管理：BPEL Process Manager、Oracle Business Process Analysis Suite
- 开发工具：Oracle Application Development Framework、Jdeveloper、SOA Suite
- 业务智能：Oracle Business Activity Monitoring、Oracle Data Integrator
- 系统管理：Oracle Enterprise Manager
- 身份管理：Oracle Identity Management
- 内容管理：Oracle Content Database Suite
- 用户交互：Oracle Portal、WebCenter

Oracle Enterprise Manager Grid Control

- 高效的 Oracle Fusion Middleware 管理
- 简化应用程序和基础结构生命周期管理
- 改进数据库管理和应用程序管理功能



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Enterprise Manager Grid Control

从应用程序管理到中间件和数据库管理，Oracle Enterprise Manager Grid Control 为 Oracle 以及非 Oracle 系统提供了集成的企业管理。

针对业务应用程序所依赖的服务（包括 SOA、业务活动监控和身份管理），Oracle Enterprise Manager Grid Control 重点推出了高级 Oracle Fusion Middleware 管理功能。

- **广泛的管理功能：**应用程序可用的功能包括服务级别管理、应用程序性能管理、配置管理以及更改自动化。
- **内置网格自动化功能：**这表示信息技术会对不断变化的需求进行主动响应，并会更快地实施新服务以保持企业的稳定发展。
- **深层诊断和迅速可用的纠正功能：**适用于多种应用程序，包括定制应用程序、Oracle 电子商务套件、PeopleSoft、Siebel、Oracle Fusion Middleware、Oracle DB 以及底层基础结构。
- **广泛的生命周期管理功能：**扩展了网格计算，可为整个应用程序和基础结构生命周期（包括测试、暂存、生产直到运行）提供解决方案。通过同步打补丁、提供额外的操作系统支持和冲突检测功能，简化了对补丁程序的管理。

Oracle BI Publisher

- 为安全地创作、管理和传达多种格式的信息提供了一个中央体系结构
- 降低开发、测试和部署各种报表的复杂性并节省时间
 - 财务报表、发票、销售或采购订单、XML 和 EDI/EFT (eText 文档)
- 支持灵活定制
 - 例如，可将 Microsoft Word 文档报表生成为多种格式，如 PDF、HTML、Excel 和 RTF 等



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle BI Publisher

Oracle Database 11g 还包括 Oracle BI Publisher (Oracle 提供的企业报表解决方案)。

Oracle BI Publisher (以前称为 XML Publisher) 提供了极其高效、可伸缩的报表解决方案，适用于复杂分布式环境。

Oracle BI Publisher 在提高报表管理效率的同时，降低了业务文档的开发、定制和维护方面的高成本。通过使用一组熟悉的桌面工具，用户可以基于 IT 人员或开发人员创建的数据查询创建并维护自己的报表格式。

可使用大多数用户都熟悉的 Microsoft Word 或 Adobe Acrobat 工具，设计 Oracle BI Publisher 报表格式。使用 Oracle BI Publisher，还可以将多个数据源中的数据集中到一个输出文档中。可以通过打印机、电子邮件或传真传送报表。可以将您的报表发布到门户。甚至可以允许用户在基于 Web 的分布式创作和版本控制 (WebDav) Web 服务器上协作编辑和管理报表。

课程安排

- 课程目标、课程安排以及本课程中使用的附录
- Oracle Database 11g 以及相关产品概览
- 关系数据库管理概念和术语概览
- SQL 及其开发环境简介
- 本课程中使用的 HR 方案和表
- Oracle Database 11g 的文档和其它资源

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

关系和对象关系数据库管理系统

- 关系模型和对象关系模型
- 用户定义的数据类型和对象
- 与关系数据库完全兼容
- 支持多媒体和大对象
- 高质量的数据库服务器功能



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

关系和对象关系数据库管理系统

Oracle Server 同时支持关系数据库模型和对象关系数据库模型。

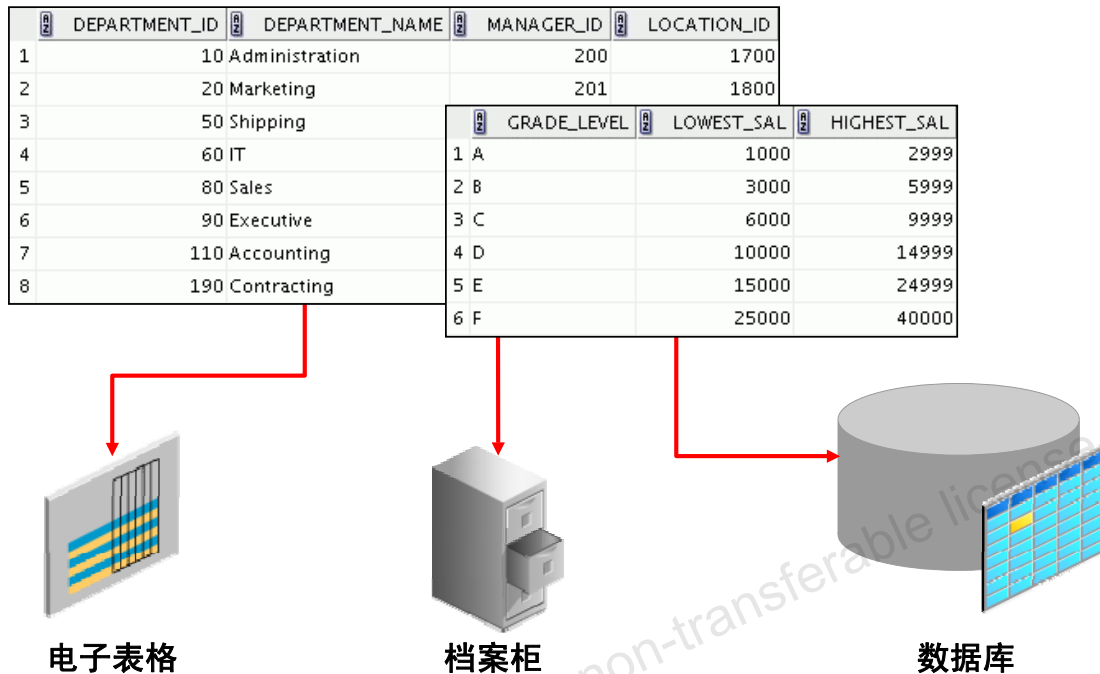
Oracle Server 扩展了数据建模功能以支持对象关系数据库模型，该模型提供面向对象的编程、复杂的数据类型、复杂的业务对象，并且与关系数据库完全兼容。

Oracle Server 包含多种可以改进 OLTP 应用程序性能和功能的特性，例如可更有效地共享运行时数据结构，获得更大的缓冲区高速缓存以及可延迟的约束条件。数据仓库应用程序可从下列增强功能中受益：插入、更新和删除操作的并行执行；分区以及识别并行的查询优化。

Oracle 模型支持客户机/服务器应用程序和基于 Web 的分布式多层应用程序。

有关关系模型和对象关系模型的详细信息，请参阅《Oracle 数据库概念 11g 发行版 1 (11.1)》。

在不同介质中存储数据



版权所有 © 2010, Oracle。保留所有权利。

在不同介质中存储数据

每个组织都有某种信息需求。图书馆需要保存成员、书籍、截止日期和罚款清单。公司需要保存有关其雇员、部门和薪金的信息。这些信息称为数据。

组织可以在各种介质中以不同格式存储数据，如档案柜中的硬拷贝文档，或在电子表格或数据库中存储的数据。

数据库是经过整理的信息集合。

要管理数据库，需要使用数据库管理系统 (DBMS)。DBMS 是一种按照要求在数据库中存储、检索和修改数据的程序。数据库有四种主要类型：分层数据库、网络数据库、关系数据库和（最新的）对象关系数据库。

关系数据库概念

- E. F. Codd 博士在 1970 年提出了数据库系统的关系模型。
- 它是关系数据库管理系统 (RDBMS) 的基础。
- 关系模型由以下各项组成：
 - 对象或关系的集合
 - 处理关系的一组运算符
 - 可保证精度和一致性的数据完整性

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

关系数据库概念

1970 年 6 月, E. F. Codd 博士在一篇题为 “A Relational Model of Data for Large Shared Data Banks” 的论文中第一次阐述了关系模型的原理。在该论文中, Codd 博士提出了数据库系统的关系模型。

当时常用的是分层模型和网络模型, 甚至还有简单的平面文件数据结构。由于关系数据库管理系统 (RDBMS) 的易用性以及结构上的灵活性, 它很快就普及开来。另外, 许多具有创新精神的供应商 (如 Oracle) 还对 RDBMS 进行了补充, 他们推出了一系列功能强大的应用程序开发产品和用户界面产品, 从而提供了一套完整的解决方案。

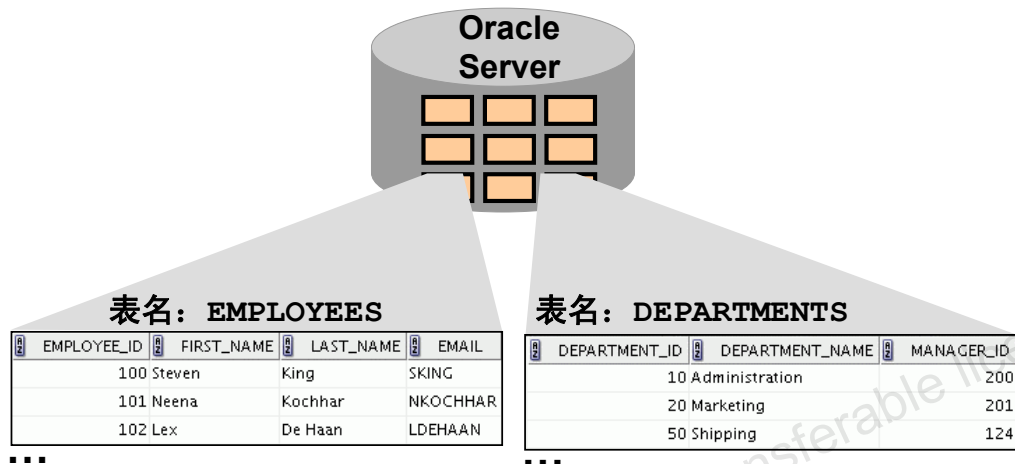
关系模型的组成部分

- 存储数据的对象或关系集合
- 一组运算符, 用于处理关系以生成其它关系
- 可保证精度和一致性的数据完整性

有关详细信息, 请参阅《An Introduction to Database Systems, Eighth Edition》(Addison-Wesley: 2004), Chris Date 著。

关系数据库的定义

关系数据库是关系或二维表的集合。



ORACLE

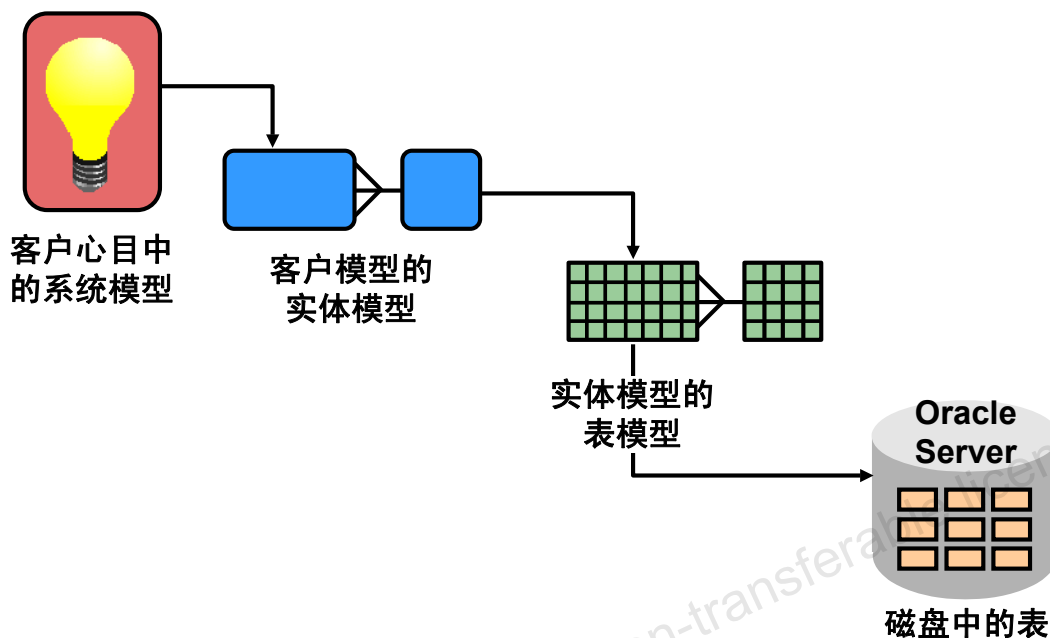
版权所有 © 2010, Oracle。保留所有权利。

关系数据库的定义

关系数据库使用关系或二维表存储信息。

例如，您可能需要存储公司中所有雇员的信息。在关系数据库中，可以创建多个表来存储雇员的各种信息，如雇员表、部门表和薪金表。

数据模型



版权所有 © 2010, Oracle。保留所有权利。

数据模型

模型是设计的基础。在汽车投入生产之前，工程师要制作汽车模型来确定所有细节。同样，系统设计人员也要构建模型，以研究数据库设计的概念并加深理解。

模型的用途

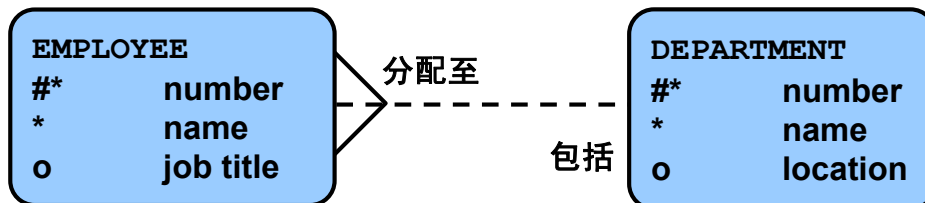
模型有助于相关人员交流各自心目中的概念。其用途包括：

- 交流
- 分类
- 描述
- 指定
- 研究
- 发展
- 分析
- 仿制

总之，目的是要建立一个能够满足以上诸多用途的模型，该模型不仅能够被最终用户所理解，还要包含足够的详细资料以便开发人员能够构建数据库系统。

实体关系模型

- 根据业务规范或陈述创建实体关系图：



- 方案：
 - “... 将一名或多名雇员分配到某个部门...”
 - “... 某些部门没有分配雇员...”

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

实体关系模型

在一个有效的系统中，数据分为不同的类别或实体。实体关系 (ER) 模型可以说明业务中的各种实体以及它们之间的关系。ER 模型来源于业务规范或陈述，且构建于系统开发周期的分析阶段。ER 模型将企业需要的信息与在企业内执行的活动相分离。尽管企业可以更改其活动，但信息类型往往保持不变。因此，数据结构也往往保持不变。

ER 建模的优点：

- 以清晰、精确的格式记录了组织的信息
- 明确表示出了信息需求的范围
- 提供了一个易于理解的数据库设计图解
- 为集成多种应用程序提供了一个有效的框架

主要组成部分

- 实体：**需要了解其信息的有意义的项。例如部门、雇员和订单。
- 属性：**描述或限定实体的项。例如，对于雇员实体，其属性为雇员编号、姓名、职务、雇用日期、部门编号等等。每种属性或者是必需的，或者是可选的。这种状态称作“optionality（可选性）”。
- 关系：**实体之间的指定关联，可以显示可选性和度。例如雇员与部门、订单与项目。

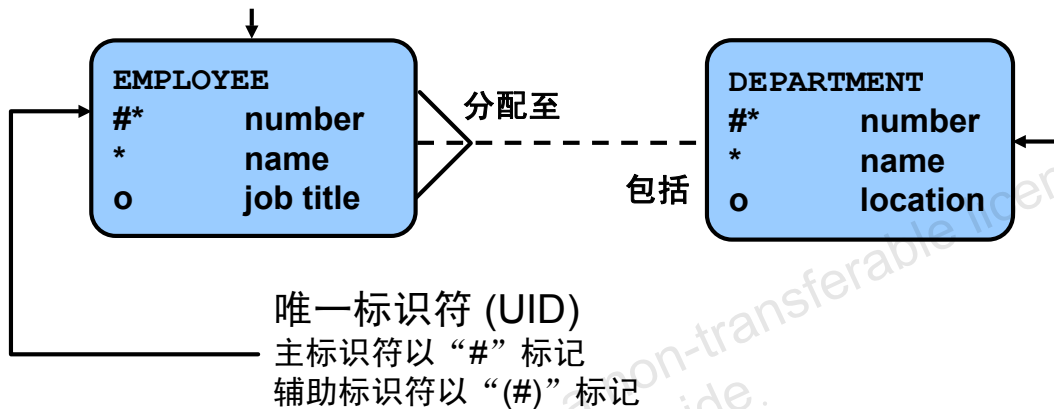
实体关系建模惯例

实体：

- 唯一名称，单数
- 大写
- 圆角方框
- 将同义词放在括号中

属性：

- 单数名称
- 小写
- 必需属性以 “*” 标记
- 可选属性以 “o” 标记



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

ER 建模惯例

实体

请使用以下惯例表示模型中的实体：

- 单数形式的唯一实体名称
- 实体名称要大写
- 使用圆角方框
- 可选同义词名称以大写形式放进括号中：()

属性

请使用以下惯例表示模型中的属性：

- 小写形式的单数名称
- 用星号 (*) 标记必需的属性（即其值必须是已知的）
- 用字母 “o” 标记可选属性（即其值可以是已知的）

ER 建模惯例（续）**关系**

符号	说明
短划线	可选元素，表示“可以”
实线	必需的元素，表示“必须”
分支线	度元素，表示“一个或更多”
单线	度元素，表示“一个且仅一个”

关系

关系的每个方向都包含以下内容：

- **标签：**例如，讲授人或分配至
- **可选性：**必须或可以
- **度：**一个且仅一个和一个或多个两者之一

注：术语“cardinality（基数）”是术语“degree（程度）”的同义词。

每个源实体 {可以 | 必须} 与 {一个且仅一个 | 一个或多个} 目标实体相关。

注：惯例是按顺时针方向进行阅读。

唯一标识符

唯一标识符 (UID) 是属性或关系（或两者）的任意组合，用于判别实体的具体实例。必须唯一地标识出实体的每个实例。

- 用井号“#”标记属于 UID 一部分的每个属性。
- 用括号内的井号 (#) 标记辅助 UID。

关联多个表

- 表中的每行数据都由主键唯一标识。
- 可以使用外键对多个表中的数据进行逻辑关联。

表名: EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DEPARTMENT_ID
100	Steven	King	90
101	Neena	Kochhar	90
102	Lex	De Haan	90
103	Alexander	Hunold	60
104	Bruce	Ernst	60
107	Diana	Lorentz	60
124	Kevin	Mourgos	50
141	Trenna	Rajs	50
142	Curtis	Davies	50

主键

外键

表名: DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

主键

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

关联多个表

每个表都包含用于准确描述一个实体的数据。例如，EMPLOYEES 表包含关于雇员的信息。数据的类别列在每个表的顶部，个别情况列在底部。使用表格式，可以轻松地显示、理解并使用信息。

由于不同实体的数据存储在不同的表中，因此可能需要将两个或多个表合并在一起，以解决某个特定问题。例如，可能需要知道某位雇员所在部门的位置。在此情况下，需要 EMPLOYEES 表（包含雇员的数据）和 DEPARTMENTS 表（包含部门的信息）中的信息。在 RDBMS 中，可以使用外键将一个表中的数据与另一个表中的数据关联起来。外键是一个列或一组列，用于引用同一个表或另一表中的主键。

可以使用此功能将一个表中的数据与另一表中的数据关联起来，以在单独的、易于管理的单元中对信息进行组织。通过将雇员数据存储在不同的表中，可以在逻辑上将其与部门数据区分开。

关联多个表（续）

主键和外键的准则

- 不能在主键中使用重复的值。
- 通常不能更改主键。
- 外键基于数据值，完全是逻辑（非物理）指针。
- 外键值必须与现有的某个主键值或唯一键值相匹配；否则，必须为空。
- 外键必须引用主键或唯一键列。

关系数据库术语

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	90
101	Neena	Kochhar	17000	(null)	90
102	Lex	De Haan	17000	(null)	90
103	Alexander	Hunold	9000	(null)	60
104	Bruce	Ernst	6000	(null)	60
107	Diana	Lorentz	4200	(null)	60
124	Kevin	Mourgos	5800	(null)	50
141	Trenna	Rajs	3500	(null)	50
142	Curtis	Davies	3100	(null)	50
143	Randall	Matos	2600	(null)	50
144	Peter	Vargas	2500	(null)	50
149	Eleni	Zlotkey	10500	0.2	80
174	Ellen	Abel	11000	0.3	80
176	Jonathan	Taylor	8600	0.2	80
178	Kimberely	Grant	7000	0.15	(null)
200	Jennifer	Whalen	4400	(null)	10
201	Michael	Hartstein	13000	(null)	20
202	Pat	Fay	6000	(null)	20
205	Shelley	Higgins	12000	(null)	110
206	William	Gietz	8300	(null)	110

Diagram annotations: 1 points to the first column (EMPLOYEE_ID), 2 points to the first row, 3 points to the SALARY column, 4 points to the DEPARTMENT_ID column, 5 points to the first row of the DEPARTMENT_ID column, and 6 points to the COMMISSION_PCT column.

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

关系数据库术语

关系数据库可以包含一个或多个表。表是 RDBMS 的基本存储结构。表包含现实生活中某些人或事物（如雇员、发票或客户）的所有必需数据。

本幻灯片显示了 EMPLOYEES 表或关系的内容。编号表示以下内容：

1. 单个行（又称元组），代表一位特定雇员的所有必需数据。表中的每一行均应由主键进行标识，以避免出现重复的行。行的顺序无关紧要，可以在检索数据时指定行的顺序。
2. 包含雇员编号的列或属性。在 EMPLOYEES 表中，雇员编号用于唯一地标识一个雇员。在本示例中，雇员编号列被指定为主键。主键必须包含值，而且此值必须是唯一的。
3. 不含键值的列。列代表表中的一类数据，本示例中的数据为所有雇员的薪金。存储数据时列的顺序无关紧要，可以在检索数据时指定列的顺序。
4. 包含部门编号的列，此列也是外键。外键是定义表之间如何关联的列。外键用于引用同一个表或另一表中的主键或唯一键。在本示例中，DEPARTMENT_ID 用于唯一地标识 DEPARTMENTS 表中的部门。
5. 行和列的交叉处是字段。字段中只能有一个值。
6. 字段中也可以没有值。此时称为空值。在 EMPLOYEES 表中，只有是销售代表的雇员在 COMMISSION_PCT（佣金）字段中才有值。

课程安排

- 课程目标、课程安排以及本课程中使用的附录
- Oracle Database 11g 以及相关产品概览
- 关系数据库管理概念和术语概览
- **SQL 及其开发环境简介**
- 本课程中使用的 HR 方案和表
- Oracle Database 11g 的文档和其它资源

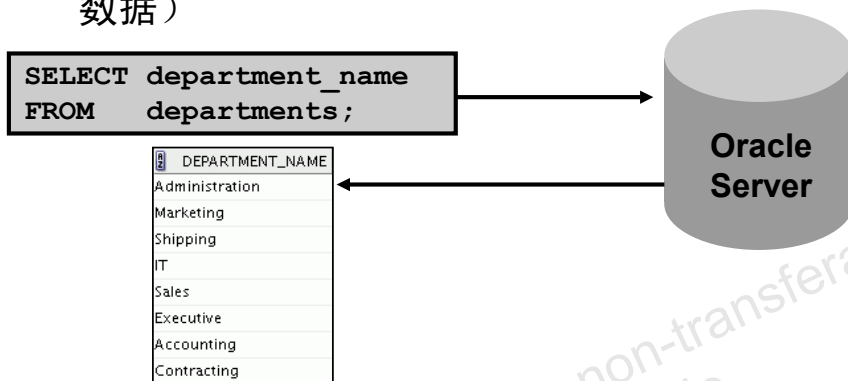
ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 SQL 查询数据库

结构化查询语言 (SQL) 具有以下特性:

- 是用于操作关系数据库的 ANSI 标准语言
- 效率高、易于学习和使用
- 功能全面 (使用 SQL, 可定义、检索和操作表中的数据)

**ORACLE**

版权所有 © 2010, Oracle。保留所有权利。

使用 SQL 查询数据库

在关系数据库中, 您不必指定表的访问路径, 也无需知道数据的实际排列方式。

要访问该数据库, 请执行结构化查询语言 (SQL) 语句, 该语言是美国国家标准协会 (ANSI) 制定的一种标准语言, 用于操作关系数据库。SQL 是所有程序和用户访问 Oracle DB 中的数据时使用的语句的集合。借助应用程序和 Oracle 工具, 用户通常不需要直接使用 SQL 访问数据库, 但这些应用程序在执行用户请求时必须使用 SQL。

SQL 为多种任务提供了语句, 其中包括:

- 查询数据
- 在表中插入、更新和删除行
- 创建、替换、更改和删除对象
- 控制对数据库及其对象的访问
- 保证数据库的一致性和完整性

SQL 将先前的所有任务统一为一种语言, 使您可以在一个逻辑层上处理数据。

SQL 语句

SELECT INSERT UPDATE DELETE MERGE	数据操纵语言 (DML)
CREATE ALTER DROP RENAME TRUNCATE COMMENT	数据定义语言 (DDL)
GRANT REVOKE	数据控制语言 (DCL)
COMMIT ROLLBACK SAVEPOINT	事务处理控制

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL 语句

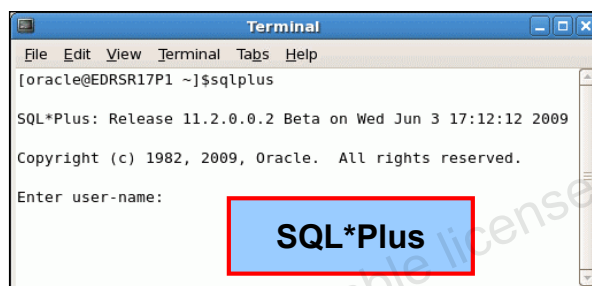
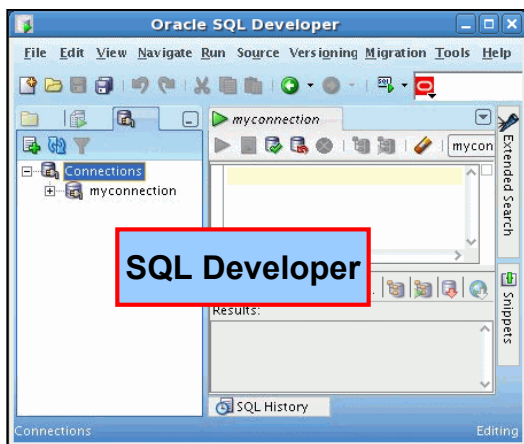
Oracle 支持的 SQL 语句遵循行业标准。Oracle Corporation 积极地与 SQL 标准委员会的重要职员保持联系，以确保始终遵从不断发展的标准。行业公认的委员会有 ANSI 和国际标准组织 (ISO)。ANSI 和 ISO 都已接受 SQL 作为关系数据库的标准语言。

语句	说明
SELECT INSERT UPDATE DELETE MERGE	分别用于从数据库中检索数据、在数据库的表中输入新行、更改现有行以及删除不需要的行。其通称为数据操纵语言 (DML)。
CREATE ALTER DROP RENAME TRUNCATE COMMENT	用于在表中设置、更改和删除数据结构。其通称为数据定义语言 (DDL)。
GRANT REVOKE	用于授予或撤消对 Oracle DB 及其中的结构的访问权限。
COMMIT ROLLBACK SAVEPOINT	用于管理由 DML 语句所做的更改。可以将对数据的更改组合到逻辑事务处理中。

SQL 的开发环境

本课程采用以下两个开发环境：

- 主要工具为 Oracle SQL Developer。
- 也可使用 SQL*Plus 命令行界面。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL 的开发环境

SQL Developer

本课程使用 Oracle SQL Developer 工具来运行课程示例和练习中讨论的 SQL 语句。Oracle Database 11g 中随附的 SQL Developer 版本 1.5.4 是本课程的默认工具。

SQL*Plus

也可使用 SQL*Plus 环境运行本课程中的所有 SQL 命令。

附注

- 请参阅附录 C 了解有关使用 SQL Developer 的信息，包括安装版本 1.5.4 的简要说明。
- 请参阅附录 D 了解有关使用 SQL*Plus 的信息。

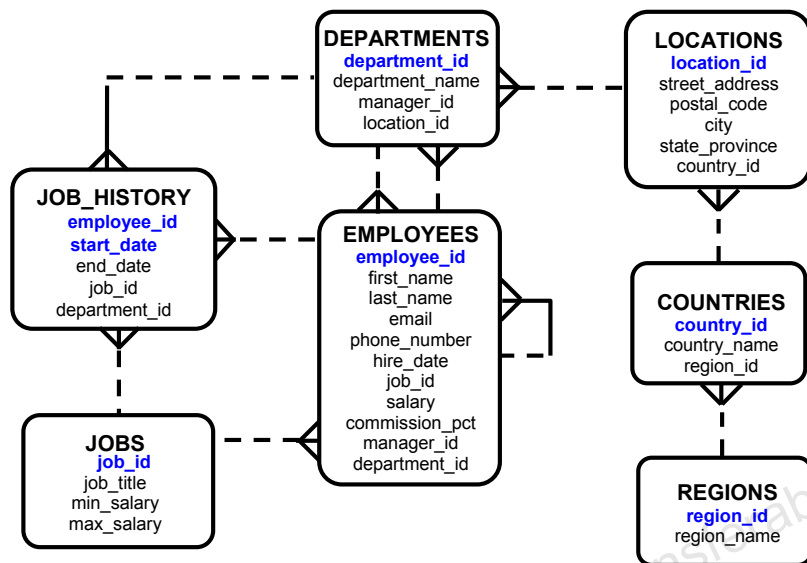
课程安排

- 课程目标、课程安排以及本课程中使用的附录
- Oracle Database 11g 以及相关产品概览
- 关系数据库管理概念和术语概览
- SQL 及其开发环境简介
- **本课程中使用的 HR 方案和表**
- Oracle Database 11g 的文档和其它资源

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

人力资源 (HR) 方案



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

人力资源 (HR) 方案描述

人力资源 (HR) 方案是 Oracle 示例方案的一部分，可以将其安装在 Oracle DB 中。本课程中的练习会话将使用 HR 方案中的数据。

表说明

- REGIONS 包含代表区域（例如美洲、亚洲等）的行。
- COUNTRIES 包含代表国家/地区的行，其中每个国家/地区均与一个区域相关联。
- LOCATIONS 包含特定国家/地区中某个公司的特定办公地点、仓库或生产地点的特定地址。
- DEPARTMENTS 显示雇员所在部门的详细资料。每个部门都可以有一个代表 EMPLOYEES 表中的部门经理的关系。
- EMPLOYEES 包含有关某个部门的每位雇员的详细资料。一些雇员可能未被分配到任何部门。
- JOBS 包含每位雇员可以担任的职务类型。
- JOB_HISTORY 包含雇员的职务历史记录。如果某个雇员更换部门，但职务保持不变，或更换职务但部门保持不变，则会在表中插入一个新行，其中包含该雇员原来的职务信息。

本课程使用的表

EMPLOYEES

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

GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

JOB_GRADES

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting	(null)	1700

DEPARTMENTS

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

本课程使用的表

本课程主要使用以下表：

- EMPLOYEES 表：给出所有雇员的详细资料
- DEPARTMENTS 表：给出所有部门的详细资料
- JOB_GRADES 表：给出各种级别的薪金的详细资料

除了上述表之外，还会用到上一张幻灯片中列出的其它表，如 LOCATIONS 和 JOB_HISTORY 表。

注：附录 B 提供所有表的结构和数据。

课程安排

- 课程目标、课程安排以及本课程中使用的附录
- Oracle Database 11g 以及相关产品概览
- 关系数据库管理概念和术语概览
- SQL 及其开发环境简介
- 本课程中使用的 HR 方案和表
- Oracle Database 11g 的文档和其它资源

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Database 11g 文档

- Oracle Database New Features Guide 11g, Release 1 (11.2)
- Oracle Database Reference 11g, Release 1 (11.2)
- Oracle Database SQL Language Reference 11g, Release 1 (11.2)
- Oracle Database Concepts 11g, Release 1 (11.2)
- Oracle Database SQL Developer User's Guide, Release 1.5

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Database 11g 文档

要访问 Oracle Database 11g 文档库，请访问 <http://www.oracle.com/pls/db112/homepage>。

其它资源

有关 Oracle Database 11g 的其它信息，请参阅以下项：

- Oracle Database 11g: New Features eStudies
- Oracle by Example 系列 (OBE): Oracle Database 11g
 - http://www.oracle.com/technology/obe/11gr1_db/index.htm

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：

- Oracle Database 11g 在以下方面进行了扩展：
 - 基础结构网络的优点
 - 现有信息管理功能
 - 使用主要应用程序开发环境（如 PL/SQL、Java/JDBC、.NET、XML 等）的能力
- 该数据库基于 ORDBMS
- 关系数据库由关系组成，并受数据完整性约束条件所控制，可以通过关系操作对其进行管理
- 通过 Oracle Server，可以使用 SQL 存储和管理信息

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

小结

关系数据库管理系统由对象或关系组成，并受数据完整性约束条件所控制，可以通过操作对其进行管理。

Oracle Corporation 提供的产品和服务可满足您的 RDBMS 需要。主要产品如下所示：

- Oracle Database 11g，可借助它使用 SQL 存储和管理信息
- Oracle Fusion Middleware，用于开发、部署和管理可集成和可重复使用的模块化业务服务
- Oracle Enterprise Manager Grid Control，用于在网格环境中跨系统集管理和自动执行管理任务

SQL

Oracle 服务器支持 ANSI 标准 SQL 并有所扩展。SQL 是用来与服务器进行通信以访问、处理和控制数据的语言。

练习 I：概览

本练习包含以下主题：

- 启动 Oracle SQL Developer
- 创建一个新的数据库连接
- 浏览 HR 表

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 I：概览

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“如果您有时间”或“如果您要接受更多的挑战”开头的练习。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。

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。

1

使用 SQL SELECT 语句检索数据

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

学完本课后，应能完成下列工作：

- 列举 SQL SELECT 语句的功能
- 执行基本 SELECT 语句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

要从数据库中提取数据，需要使用 SQL SELECT 语句。不过，您可能需要对显示的列进行限制。本课将介绍执行这些操作所需要的 SELECT 语句。此外，您可能要创建可以多次使用的 SELECT 语句。

课程安排

- 基本 SELECT 语句
- SELECT 语句中的算术表达式和空值
- 列别名
- 连接运算符、文字字符串、其它引号运算符和 DISTINCT 关键字的用法
- DESCRIBE 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL SELECT 语句的功能

映射

表 1

选择

表 1

表 1

联接

表 2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL SELECT 语句的功能

使用 SELECT 语句可从数据库中检索信息。使用 SELECT 语句可以执行下列操作：

- **映射：**选择由查询返回的表中列。可以根据需要选择任意数量的列。
- **选择：**选择由查询返回的表中行。可以使用不同的标准限制所检索的行。
- **联接：**通过指定不同表之间的链接来将存储在不同表中的数据集合在一起。SQL 联接将在“使用联接显示多个表中的数据”一课中进行详细介绍。

基本 SELECT 语句

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM      table;
```

- SELECT 标识要显示的列。
- FROM 标识包含上述各列的表。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

基本 SELECT 语句

形式最简单的 SELECT 语句必须包含以下内容：

- 一个 SELECT 子句，该子句指定要显示的列。
- 一个 FROM 子句，该子句标识包含 SELECT 子句中列出的各列的表。

在该语法中：

SELECT	包含一个列或多个列的列表
*	选择所有列
DISTINCT	隐藏重复项
column expression	选择指定的列或表达式
alias	为选定列赋予其它标题
FROM table	指定包含列的表

注：本课程中“关键字”、“子句”和“语句”的用法如下：

- 关键字指的是单个 SQL 元素，例如，SELECT 和 FROM 都是关键字。
- 子句是 SQL 语句的一部分，例如，SELECT employee_id、last_name 等。
- 语句是两个或更多个子句的组合，例如，SELECT * FROM employees。

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

选择所有列

```
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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

选择所有列

通过在 SELECT 关键字后加上一个星号 (*), 可以显示表中所有数据列。在幻灯片示例中, DEPARTMENTS 表包含以下四列: DEPARTMENT_ID、DEPARTMENT_NAME、MANAGER_ID 和 LOCATION_ID。该表包含八行, 每行代表一个部门。

也可以通过在 SELECT 关键字之后列出所有列来显示表中的所有列。例如, 下面的 SQL 语句 (与幻灯片示例类似) 将显示 DEPARTMENTS 表的所有列和所有行:

```
SELECT department_id, department_name, manager_id, location_id  
FROM departments;
```

注: 在 SQL Developer 中, 可在 SQL 工作表中输入一个 SQL 语句, 然后单击 “Execute Statement (执行语句)” 图标或按 [F9] 来执行该语句。在 “Results (结果)” 选项卡页上显示的输出如幻灯片所示。

选择特定列

```
SELECT department_id, location_id
FROM departments;
```

	DEPARTMENT_ID	LOCATION_ID
1	10	1700
2	20	1800
3	50	1500
4	60	1400
5	80	2500
6	90	1700
7	110	1700
8	190	1700

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

选择特定列

通过指定列名（以逗号分隔），可使用 SELECT 语句来显示表中的特定列。幻灯片示例中显示 DEPARTMENTS 表中的所有部门编号和位置编号。

在 SELECT 子句中，可以按所需输出顺序指定各列。例如，要让位置编号出现在部门编号的左边，应使用以下语句：

```
SELECT location_id, department_id
FROM departments;
```

	LOCATION_ID	DEPARTMENT_ID
1	1700	10
2	1800	20
3	1500	50
4	1400	60

...

编写 SQL 语句

- SQL 语句不区分大小写。
- SQL 语句可输入在一行或多行中。
- 关键字不能缩写，也不能跨行分开写。
- 子句通常放在单独的行中。
- 应使用缩进来提高可读性。
- 在 SQL Developer 中，可以根据需要使用分号 (;) 终止 SQL 语句。如果执行多条 SQL 语句，则需要使用分号来分隔这些语句。
- 在 SQL*Plus 中，必须使用分号 (;) 结束每条 SQL 语句。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

编写 SQL 语句

通过使用以下简单规则和准则，可以构建简单易懂易编辑的有效语句：

- SQL 语句不区分大小写（除非另行指明）。
- SQL 语句可输入在一行或多行中。
- 关键字不能缩写，也不能跨行分开写。
- 子句通常放在单独的行中，目的是为了提高可读性和方便进行编辑。
- 应使用缩进来提高代码的可读性。
- 关键字通常以大写字母形式输入，而其它所有词（如表名和列名）以小写字母形式输入。

编写 SQL 语句（续）

执行 SQL 语句

在 SQL Developer 中，单击“Run Script（运行脚本）”图标或按 [F5] 可运行 SQL 工作表中的一条或多条命令。此外，还可以单击“Execute Statement（执行语句）”图标或按 [F9] 来运行 SQL 工作表中的 SQL 语句。“Run Script（运行脚本）”图标执行“Enter SQL Statement（输入 SQL 语句）”框中的所有语句，而“Execute Statement（执行语句）”图标执行“Enter SQL Statement（输入 SQL 语句）”框中鼠标光标所指的语句。“Run Script（运行脚本）”图标模拟 SQL*Plus 显示并在“Script Output（脚本输出）”选项卡页上显示输出，而“Execute Statement（执行语句）”图标在“Results（结果）”选项卡页上显示查询输出。

在 SQL*Plus 中，使用分号终止 SQL 语句，然后按 [Enter] 可运行该命令。

列标题的默认设置

- SQL Developer:
 - 默认的标题对齐方式：左对齐
 - 默认的标题显示方式：大写
- SQL*Plus:
 - 字符和日期列标题的对齐方式：左对齐
 - 数字列标题的对齐方式：右对齐
 - 默认的标题显示方式：大写

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

列标题的默认设置

在 SQL Developer 中，列标题左对齐且以大写形式显示。

```
SELECT last_name, hire_date, salary
FROM employees;
```

	LAST_NAME	HIRE_DATE	SALARY
1	Whalen	17-SEP-87	4400
2	Hartstein	17-FEB-96	13000
3	Fay	17-AUG-97	6000
4	Higgins	07-JUN-94	12000
5	Gietz	07-JUN-94	8300
6	King	17-JUN-87	24000
7	Kochhar	21-SEP-89	17000
8	De Haan	13-JAN-93	17000
9	Hunold	03-JAN-90	9000

■ ■ ■

可以用别名改写所显示的列标题。本课稍后将对列别名进行介绍。

课程安排

- 基本 SELECT 语句
- SELECT 语句中的算术表达式和空值
- 列别名
- 连接运算符、文字字符串、其它引号运算符和 DISTINCT 关键字的用法
- DESCRIBE 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

算术表达式

使用算术运算符可以创建包含数字和日期数据的表达式。

运算符	说明
+	加
-	减
*	乘
/	除

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

算术表达式

您可能需要修改数据的显示方式，也可能要执行计算，或者要查看假设分析场景。这些任务都可以使用算术表达式来完成。算术表达式可以包含列名、数字常量值和算术运算符。

算术运算符

幻灯片中列出了可以在 SQL 中使用的算术运算符。可以在 SQL 语句的任何子句（FROM 子句除外）中使用算术运算符。

注：对于 DATE 和 TIMESTAMP 数据类型，只可以使用“加”和“减”运算符。

使用算术运算符

```
SELECT last_name, salary, salary + 300
FROM employees;
```

	LAST_NAME	SALARY	SALARY+300
1	Whalen	4400	4700
2	Hartstein	13000	13300
3	Fay	6000	6300
4	Higgins	12000	12300
5	Gietz	8300	8600
6	King	24000	24300
7	Kochhar	17000	17300
8	De Haan	17000	17300
9	Hunold	9000	9300
10	Ernst	6000	6300

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用算术运算符

幻灯片示例中使用“+”运算符计算所有雇员的薪金增加 \$300 后的结果。幻灯片输出中还显示一个 SALARY+300 列。

请注意，生成的计算列 SALARY+300 不是 EMPLOYEES 表中的一个新列，该列只是为了显示计算结果。默认情况下，新列的名称来自于生成该列的计算公式，在本例中，为 salary+300。

注：Oracle Server 会忽略算术运算符前后的空格。

运算符优先级

如果算术表达式包含多个运算符，则先执行乘和除运算。如果表达式中运算符的优先级相同，则从左到右依次进行计算。

可以使用括号强制先对括号中的表达式进行计算。

优先级规则：

- 乘除运算优先于加减运算。
- 相同优先级的运算符按从左到右的顺序进行计算。
- 括号可以更改默认优先级，也可以使语句变得更易于理解。

运算符优先级

```
SELECT last_name, salary, 12*salary+100
FROM employees;
```

1

	LAST_NAME	SALARY	12*SALARY+100
1	Whalen	4400	52900
2	Hartstein	13000	156100
3	Fay	6000	72100

...

```
SELECT last_name, salary, 12*(salary+100)
FROM employees;
```

2

	LAST_NAME	SALARY	12*(SALARY+100)
1	Whalen	4400	54000
2	Hartstein	13000	157200
3	Fay	6000	73200

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

运算符优先级（续）

幻灯片中的第一个示例显示雇员的姓氏、薪金和年度报酬。用月薪乘以 12 后再加上一次性奖金 \$100 就可以计算出年度报酬。注意，乘法运算在加法运算之前执行。

注：使用括号可以强制执行优先级标准顺序，使优先级顺序变得更清楚。例如，幻灯片中的表达式可写成 $(12 * salary) + 100$ ，计算结果是相同的。

使用括号

通过使用括号指定所需的运算符执行顺序，可以更改优先级规则。

幻灯片中的第二个示例也显示雇员的姓氏、薪金和年度报酬。该示例计算年度报酬的方法如下：将月薪与月奖金 \$100 相加，然后乘以 12。由于使用了括号，所以加法运算的优先级高于乘法运算的优先级。

定义空值

- 空值是不可用的、未分配的、未知的或不适用的值。
- 空值不同于零或空格。

```
SELECT last_name, job_id, salary, commission_pct
FROM employees;
```

	LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
1	Whalen	AD_ASST	4400	(null)
2	Hartstein	MK_MAN	13000	(null)
...				
17	Zlotkey	SA_MAN	10500	0.2
18	Abel	SA_REP	11000	0.3
19	Taylor	SA_REP	8600	0.2
20	Grant	SA_REP	7000	0.15

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

定义空值

如果某行的特定列缺少数据值，则称该值为空值或该列包含空值。

空值是不可用的、未分配的、未知的或不适用的值。空值不同于零或空格。零是一个数字，空格是一个字符。

任何数据类型的列都可以包含空值。但是，某些约束条件（NOT NULL 和 PRIMARY KEY）会禁止在有关列中使用空值。

您会注意到，在 EMPLOYEES 表的 COMMISSION_PCT 列中，只有销售经理或销售代表可以获得佣金。其他雇员无权获得佣金。空值就代表这种情况。

注：默认情况下，SQL Developer 使用文字值 (null) 来标识空值。不过，您可以将它设置为说明性更强的内容。为此，请从“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）”节点。单击右窗格中的“Advanced Parameters（高级参数）”，然后在“Display Null value As（将空值显示为）”字段中输入适当的值。

算术表达式中的空值

包含空值的算术表达式的计算结果也为空。

```
SELECT last_name, 12*salary*commission_pct
FROM employees;
```

	1 LAST_NAME	2 12*SALARY*COMMISSION_PCT
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)
...		
17	Zlotkey	25200
18	Abel	39600
19	Taylor	20640
20	Grant	12600

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

算术表达式中的空值

如果算术表达式中的任何一个列值为空，则其结果也为空。例如，如果试图执行除数为零的除法运算，则会出现错误。但是，如果用一个数除以空值，则结果为空值或未知。

在幻灯片示例中，雇员 Whalen 没有获得任何佣金。这是因为在算术表达式中 COMMISSION_PCT 列为空值，因此结果也为空值。

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

课程安排

- 基本 SELECT 语句
- SELECT 语句中的算术表达式和空值
- **列别名**
- 连接运算符、文字字符串、其它引号运算符和 DISTINCT 关键字的用法
- DESCRIBE 命令

ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

定义列别名

列别名具有以下特征和用途：

- 可重命名列标题
- 有助于计算
- 紧跟在列名后（列名和别名之间也可以加上可选关键字 `AS`）
- 如果别名包含空格或特殊字符，或者区分大小写，则需要使用双引号

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

定义列别名

在显示查询结果时，SQL Developer 通常使用选定列的名称作为列标题。此标题可能说明性较差，因此可能不容易理解。您可以使用列别名来更改列标题。

在 `SELECT` 列表中，可以使用空格作为分隔符，在列名之后指定别名。默认情况下，别名标题以大写形式显示。如果别名包含空格或特殊字符（如 `#` 或 `$`），或者区分大小写，则应将别名放在双引号 ("") 内。

使用列别名

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

...

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用列别名

第一个示例显示所有雇员的姓名和佣金百分比。注意，列别名前使用了可选关键字 AS。无论是否使用关键字 AS，查询结果都相同。另外请注意，SQL 语句中的列别名 name 和 comm 是小写形式，而查询结果中的列标题以大写形式显示。如先前的幻灯片所述，默认情况下列标题以大写形式显示。

第二个示例显示所有雇员的姓氏和年薪。因为 “Annual Salary” 包含一个空格，所以应将其放在双引号内。请注意，输出中的列标题与列别名是完全相同的。

课程安排

- 基本 SELECT 语句
- SELECT 语句中的算术表达式和空值
- 列别名
- 连接运算符、文字字符串、其它引号运算符和 DISTINCT 关键字的用法
- DESCRIBE 命令

ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

连接运算符

连接运算符具有以下特征和用途：

- 将列或字符串链接到其它列
- 由两条竖线 (||) 表示
- 创建一个由字符表达式生成的列

```
SELECT last_name||job_id AS "Employees"
FROM employees;
```

	Employees
1	AbelSA_REP
2	DaviesST_CLERK
3	De HaanAD_VP
4	ErnstIT_PROG
5	FayMK_REP
6	GietzAC_ACCOUNT

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

连接运算符

可以使用连接运算符 (||) 将某些列链接到其它列、算术表达式或常数值，从而创建一个字符表达式。运算符两边的列会组合形成一个输出列。

在本示例中，LAST_NAME 和 JOB_ID 连接在一起，而且具有一个别名 “Employees”。请注意，雇员姓氏和职务代码会组合形成一个输出列。

别名之前的关键字 AS 可提高 SELECT 子句的可读性。

将连接运算符用于空值

如果将一个空值连接到一个字符串，则结果为字符串。例如，LAST_NAME || NULL 结果为 LAST_NAME。

注：也可以将日期表达式链接到其它表达式或其它列。

文字字符串

- 文字是指 `SELECT` 语句中包含的字符、数字或日期。
- 日期和字符文字值必须放在单引号内。
- 每个字符串在每个返回行中输出一次。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

文字字符串

文字是指 `SELECT` 列表中包含的字符、数字或日期。列名或列别名除外。在每个返回行中都会打印文字字符串。自由格式文本的文字字符串可以包含在查询结果中，其处理方式与 `SELECT` 列表中的列相同。

日期和字符文字必须放在单引号 (' ') 内，数字文字则不需要。

使用文字字符串

```
SELECT last_name || ' is a ' || job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details	
1	Abel is a SA_REP
2	Davies is a ST_CLERK
3	De Haan is a AD_VP
4	Ernst is a IT_PROG
5	Fay is a MK_REP
6	Gietz is a AC_ACCOUNT
7	Grant is a SA_REP
8	Hartstein is a MK_MAN
9	Higgins is a AC_MGR
10	Hunold is a IT_PROG

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用文字字符串

幻灯片中的示例显示所有雇员的姓氏和职务代码。其列标题为“Employee Details”。注意 SELECT 语句中单引号之间的空格。空格提高了输出的可读性。

在下面的示例中，每位雇员的姓氏和薪金都通过一个文字连接在一起，使返回行更有意义：

```
SELECT last_name || ': 1 Month salary = ' || salary Monthly
FROM   employees;
```

MONTHLY	
1	Whalen: 1 Month salary = 4400
2	Hartstein: 1 Month salary = 13000
3	Fay: 1 Month salary = 6000
4	Higgins: 1 Month salary = 12000
5	Gietz: 1 Month salary = 8300
6	King: 1 Month salary = 24000
7	Kochhar: 1 Month salary = 17000
8	De Haan: 1 Month salary = 17000

...

其它引号 (q) 运算符

- 指定您自己的引号分隔符。
- 选择任一分隔符。
- 提高可读性和易用性。

```
SELECT department_name || q'[ Department's Manager Id: ]'
      || manager_id
      AS "Department and Manager"
FROM departments;
```

Department and Manager
1 Administration Department's Manager Id: 200
2 Marketing Department's Manager Id: 201
3 Shipping Department's Manager Id: 124
4 IT Department's Manager Id: 103
5 Sales Department's Manager Id: 149
6 Executive Department's Manager Id: 100
7 Accounting Department's Manager Id: 205
8 Contracting Department's Manager Id:

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

其它引号 (q) 运算符

许多 SQL 语句都在表达式或条件中使用字符文字。如果文字本身包含一个单引号，则可以使用引号 (q) 运算符并选择自己的引号分隔符。

可以选择所需要的任何分隔符，单字节或多字节分隔符，或者下列字符对中的任何一种：[]、{}、() 或 <>。

在显示的示例中，字符串包含一个单引号，该单引号通常被解释为字符串的分隔符。但是，通过使用 q 运算符，方括号 [] 被用作引号分隔符。方括号分隔符之间的字符串被解释为文字字符串。

重复行

默认情况下会显示查询返回的所有行，包括重复行。

1

```
SELECT department_id
FROM employees;
```

	DEPARTMENT_ID
1	10
2	20
3	20
4	110
5	110

...

2

```
SELECT DISTINCT department_id
FROM employees;
```

	DEPARTMENT_ID
1	(null)
2	20
3	90
4	110
5	50
6	80
7	10
8	60

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

重复行

除非另行说明，否则 SQL 将显示包括重复行在内的查询结果。幻灯片中的第一个示例显示 EMPLOYEES 表中的所有部门编号。请注意，有些部门编号是重复的。

要消除结果中的重复行，请紧接在 SELECT 子句中的 SELECT 关键字之后使用 DISTINCT 关键字。在幻灯片的第二个示例中，EMPLOYEES 表实际包含 20 行，但是该表中只有 7 个部门编号是唯一的。

您可以在 DISTINCT 限定词之后指定多个列。DISTINCT 限定词会影响所有选定的列，结果中只包括所有不同的列组合。

```
SELECT DISTINCT department_id, job_id
FROM employees;
```

	DEPARTMENT_ID	JOB_ID
1	110	AC_ACCOUNT
2	90	AD_VP
3	50	ST_CLERK

...

注：还可以指定关键字 UNIQUE，这是关键字 DISTINCT 的同义词。

课程安排

- 基本 SELECT 语句
- SELECT 语句中的算术表达式和空值
- 列别名
- 连接运算符、文字字符串、其它引号运算符和 DISTINCT 关键字的用法
- DESCRIBE 命令

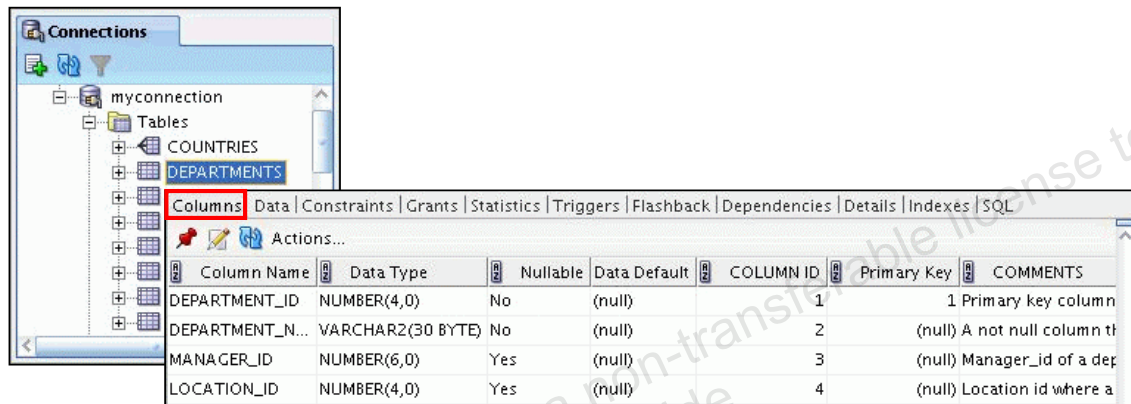
ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

显示表结构

- 使用 DESCRIBE 命令可显示表结构。
- 或者在“Connections（连接）”树中选择表，然后使用“Columns（列）”选项卡查看表结构。

```
DESC[RIBE] tablename
```



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

显示表结构

可以使用 DESCRIBE 命令显示表的结构。该命令显示列名和数据类型，还显示列是否必须包含数据（即，列是否有 NOT NULL 约束条件）。

在此语法中，*table name* 是用户可以访问的任何现有表、视图或同义词的名称。

使用 SQL Developer GUI 界面时，也可在“Connections（连接）”树中选择表，然后使用“Columns（列）”选项卡查看表结构。

注：SQL*Plus 和 SQL Developer 都支持 DESCRIBE 命令。

使用 DESCRIBE 命令

DESCRIBE employees

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)
11 rows selected		

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

使用 DESCRIBE 命令

幻灯片中的示例使用 DESCRIBE 命令显示关于 EMPLOYEES 表结构的信息。

在显示结果中, *Null* 表示此列的值可以是未知的。NOT NULL 表示列必须包含数据。Type 显示列的数据类型。

下表说明了数据类型:

数据类型	说明
NUMBER (<i>p</i> , <i>s</i>)	最大位数为 <i>p</i> , 小数点右侧的位数为 <i>s</i> 位
VARCHAR2 (<i>s</i>)	可变长度的字符值, 最大长度为 <i>s</i>
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

小测验

找出可成功执行的 SELECT 语句。

1.

```
SELECT first_name, last_name, job_id, salary*12
   AS Yearly Sal
FROM   employees;
```
2.

```
SELECT first_name, last_name, job_id, salary*12
   "yearly sal"
FROM   employees;
```
3.

```
SELECT first_name, last_name, job_id, salary AS
   "yearly sal"
FROM   employees;
```
4.

```
SELECT first_name+last_name AS name, job_Id,
   salary*12 yearly sal
FROM   employees;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：2、3

小结

在本课中，您应该已经学会：

- 编写执行以下任务的 SELECT 语句：
 - 从表中返回所有行和列
 - 从表中返回指定列
 - 使用列别名显示说明性更强的列标题

```
SELECT *|{[DISTINCT] column|expression [alias],...}
FROM table;
```

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 SELECT 语句从数据库表中检索数据。

```
SELECT *|{[DISTINCT] column [alias],...}
FROM table;
```

在该语法中：

SELECT	包含一个列或多个列的列表
*	选择所有列
DISTINCT	隐藏重复项
column expression	选择指定的列或表达式
alias	为选定列赋予其它标题
FROM table	指定包含列的表

练习 1：概览

本练习包含以下主题：

- 从不同的表中选择所有数据
- 描述表结构
- 执行算术计算并指定列名

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 1：概览

在此练习中，您将编写几个简单的 SELECT 查询。这些查询涵盖您在本课中学习过的大部分 SELECT 子句和运算。

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.

2 对数据进行限制和排序

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：

- 对通过查询检索的行进行限制
- 对通过查询检索的行进行排序
- 在运行时使用 & 替代变量对输出进行限制和排序

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

在数据库中检索数据时，可能需要执行以下操作：

- 对显示的数据行进行限制
- 指定行的显示顺序

本课将介绍用来执行上述操作的 SQL 语句。

课程安排

- 使用以下项对行进行限制：
 - WHERE 子句
 - 使用 =、<=、BETWEEN、IN、LIKE 和 NULL 条件的比较条件
 - 使用 AND、OR 和 NOT 运算符的逻辑条件
- 表达式中运算符的优先级规则
- 使用 ORDER BY 子句对行进行排序
- 替代变量
- DEFINE 和 VERIFY 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

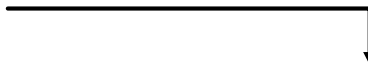
有选择地对行进行限制

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200 Whalen	AD_ASST	10
2	201 Hartstein	MK_MAN	20
3	202 Fay	MK_REP	20
4	205 Higgins	AC_MGR	110
5	206 Gietz	AC_ACCOUNT	110

...

“检索部门 90
中的所有雇员”



EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100 King	AD_PRES	90
2	101 Kochhar	AD_VP	90
3	102 De Haan	AD_VP	90

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

有选择地对行进行限制

在幻灯片示例中，假设您希望显示部门 90 中的所有雇员，则只会返回 DEPARTMENT_ID 列的值为 90 的那些行。此限制方法基于 SQL 中的 WHERE 子句。

对所选行进行限制

- 使用 WHERE 子句可以限制返回的行：

```
SELECT *|{ [DISTINCT] column|expression [alias],...}
FROM table
[WHERE condition(s)];
```

- WHERE 子句在 FROM 子句之后。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

对所选行进行限制

使用 WHERE 子句可限制由查询返回的行。WHERE 子句包含一个必须满足的条件，该子句直接跟在 FROM 子句之后。如果该条件为真，则返回满足条件的行。

在该语法中：

WHERE

限制查询只返回满足条件的行。

condition

由列名、表达式、常数和比较运算符组成。条件指定一个或多个表达式和逻辑（布尔型）运算符组合，并返回 TRUE、FALSE 或 UNKNOWN 值。

WHERE 子句可对列、文字、算术表达式或函数中的值进行比较。它包含三个元素：

- 列名
- 比较条件
- 列名、常数或值列表

使用 WHERE 子句

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 WHERE 子句

在本示例中，SELECT 语句将检索部门 90 中所有雇员的雇员 ID、姓氏、职务 ID 和部门编号。

注：不能在 WHERE 子句中使用列别名。

字符串和日期

- 字符串和日期值应放在单引号内。
- 字符值区分大小写，日期值区分格式。
- 默认日期显示格式为 DD-MON-RR。

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen' ;
```

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-FEB-96' ;
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

字符串和日期

WHERE 子句中的字符串和日期必须放在单引号 (') 内，但数字常数则不必。

所有字符搜索均区分大小写。在以下示例中，由于 EMPLOYEES 表以大小写混用形式存储所有姓氏，因此不会返回任何行：

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'WHALEN';
```

Oracle DB 以内部数字格式存储日期，以此代表世纪、年、月、日、小时、分钟和秒。默认日期显示格式为 DD-MON-RR。

注：有关 RR 格式及更改默认日期格式的详细信息，请参阅“使用单行函数定制输出”一课。在该课中，您还将学习如何使用 UPPER 和 LOWER 等单行函数来改写大小写。

比较运算符

运算符	含义
=	等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
<>	不等于
BETWEEN ...AND...	两值之间（包含这两个值）
IN(set)	与任一列表值相匹配
LIKE	与字符模式相匹配
IS NULL	为空值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

比较运算符

将一个表达式与另一个值或另一个表达式进行比较时需要使用比较运算符。在 WHERE 子句中按以下格式使用比较运算符：

语法

```
... WHERE expr operator value
```

示例

```
... WHERE hire_date = '01-JAN-95'
... WHERE salary >= 6000
... WHERE last_name = 'Smith'
```

请注意，不能在 WHERE 子句中使用别名。

注：符号 != 和 ^= 也可以表示“不等于”条件。

使用比较运算符

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

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用比较运算符

在本示例中，SELECT 语句从 EMPLOYEES 表检索其薪金少于或等于 \$3,000 的所有雇员的姓氏和薪金。请注意，在 WHERE 子句中提供了一个确切值。确切值 3000 用于与 EMPLOYEES 表的 SALARY 列中的薪金值进行比较。

使用 BETWEEN 运算符的范围条件

使用 BETWEEN 运算符可基于值范围显示行：

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

下限

上限

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 BETWEEN 运算符的范围条件

使用 BETWEEN 运算符可基于值范围显示行。指定的范围包含一个下限和一个上限。

幻灯片中的 SELECT 语句将返回 EMPLOYEES 表中薪金介于 \$2,500 和 \$3,500 之间的所有雇员对应的行。

BETWEEN 运算符指定的值范围包括两个端点值。不过，必须先指定下限。

此外，也可以对字符值使用 BETWEEN 运算符：

```
SELECT last_name
FROM employees
WHERE last_name BETWEEN 'King' AND 'Smith';
```

	LAST_NAME
1	King
2	Kochhar
3	Lorentz
4	Matos
5	Mourgos
6	Rajs

使用 IN 运算符的成员条件

使用 IN 运算符可测试列表中的值：

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	201	Hartstein	13000	100
2	101	Kochhar	17000	100
3	102	De Haan	17000	100
4	124	Mourgos	5800	100
5	149	Zlotkey	10500	100
6	200	Whalen	4400	101
7	205	Higgins	12000	101
8	202	Fay	6000	201

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 IN 运算符的成员条件

要测试指定值集中的值，请使用 IN 运算符。使用 IN 运算符定义的条件又称为“成员条件”。

幻灯片示例中显示雇员编号是 100、101 或 201 的经理属下所有雇员的雇员编号、姓氏、薪金及经理本人的雇员编号。

注：可以随机顺序指定一组值，例如，(201,100,101)。

IN 运算符可用于任何数据类型。以下示例将返回 EMPLOYEES 表中的若干行，这些行分别对应于其姓氏包含在 WHERE 子句的姓名列表中的任一雇员。

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  last_name IN ('Hartstein', 'Vargas');
```

如果要在列表中使用字符或日期，则必须将它们放在单引号 (') 内。

注：在 Oracle Server 内部将 IN 运算符作为一组 OR 条件（如 a=value1 or a=value2 or a=value3）进行求值。因此，使用 IN 运算符对改进性能并无益处，只能达到简化逻辑的目的。

使用 LIKE 运算符执行模式匹配

- 使用 LIKE 运算符可执行通配符搜索，查找有效搜索字符串值。
- 搜索条件可包含文字字符或数字：
 - % 表示零个或多个字符。
 - _ 表示一个字符。

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 LIKE 运算符执行模式匹配

有时您并不知道要搜索的准确值。此时可使用 LIKE 运算符来选择与字符模式相匹配的行。字符模式匹配操作被称为通配符搜索。可以使用两个符号来构建搜索字符串。

符号	说明
%	代表由零个或多个字符组成的任意序列
_	代表任意一个字符

幻灯片中的 SELECT 语句将返回 EMPLOYEES 表中其名字以字母 S 开头的所有雇员的名字。注意是大写的 S，因而不会返回以 s 开头的名字。

LIKE 运算符可用作执行某些 BETWEEN 比较的一种快捷方式。以下示例显示在 1995 年 1 月和 1995 年 12 月之间进入公司的所有雇员的姓氏和聘用日期：

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date LIKE '%95';
```

组合通配符字符

- 为执行模式匹配，可将两个通配符 (% , _) 与文字字符组合使用：

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

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

- 可使用 ESCAPE 标识符来搜索实际的 % 和 _ 符号。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

组合通配符字符

可将 % 和 _ 符号与文字字符以任意方式组合起来使用。幻灯片示例中显示其姓氏以字母“o”作为第二个字符的所有雇员的姓名。

ESCAPE 标识符

如果需要与实际的 % 和 _ 字符完全相匹配，请使用 ESCAPE 标识符。此选项指定转义符是什么。如果要搜索包含 SA_ 的字符串，则可使用下面的 SQL 语句：

```
SELECT employee_id, last_name, job_id
FROM   employees WHERE  job_id LIKE '%SA\_%' ESCAPE '\';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID
1	149	Zlotkey	SA_MAN
2	174	Abel	SA_REP
3	176	Taylor	SA_REP
4	178	Grant	SA_REP

ESCAPE 标识符将反斜杠 (\) 标识为转义符。在 SQL 语句中，转义符位于下划线 (_) 之前。这样 Oracle Server 将按字面意义解释下划线。

使用 NULL 条件

使用 IS NULL 运算符可测试空值。

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

	LAST_NAME	MANAGER_ID
1	King	(null)

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 NULL 条件

NULL 条件包括 IS NULL 条件和 IS NOT NULL 条件。

IS NULL 条件可测试空值。空值表示该值不可用、未分配、未知或不适用。因此，不能使用 = 来进行测试，因为空值即不会等于任何值，也不会不等于任何值。幻灯片示例中将检索没有经理的所有雇员的姓氏和经理。

下面是另一个示例：使用下面的 SQL 语句可显示无权获得佣金的所有雇员的姓氏、职务 ID 和佣金：

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

	LAST_NAME	JOB_ID	COMMISSION_PCT
1	Whalen	AD_ASST	(null)
2	Hartstein	MK_MAN	(null)
3	Fay	MK_REP	(null)
4	Higgins	AC_MGR	(null)
5	Gietz	AC_ACCOUNT	(null)

...

使用逻辑运算符定义条件

运算符	含义
AND	如果两个条件都为真，则返回 TRUE
OR	如果其中一个条件为真，则返回 TRUE
NOT	如果条件为假，则返回 TRUE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用逻辑运算符定义条件

逻辑条件组合了两个组合条件的结果并根据这两个条件生成一个结果，或者反转单个条件的结果。仅当条件的整体结果为真时才返回行。

在 SQL 中可以使用以下三种逻辑运算符：

- AND
- OR
- NOT

到目前为止，所有示例在 WHERE 子句中都仅指定一个条件。通过使用 AND 和 OR 运算符，可在一个 WHERE 子句中使用多个条件。

使用 AND 运算符

AND 要求两个条件都为真：

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
AND    job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	149	Zlotkey	SA_MAN	10500

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 AND 运算符

在本示例中，只会选择同时满足两个组合条件的所有记录。因此，只选择其职位包含字符串“MAN”且薪金为 \$10,000 或更高的那些雇员。

所有字符搜索都区分大小写，也就是说，不会返回“MAN”不是大写的任何行。另外，字符串必须放在引号内。

AND 真值表

下表显示用 AND 组合两个表达式的结果：

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

使用 OR 运算符

OR 要求其中一个条件为真：

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	201	Hartstein	MK_MAN	13000
2	205	Higgins	AC_MGR	12000
3	100	King	AD_PRES	24000
4	101	Kochhar	AD_VP	17000
5	102	De Haan	AD_VP	17000
6	124	Mourgos	ST_MAN	5800
7	149	Zlotkey	SA_MAN	10500
8	174	Abel	SA_REP	11000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 OR 运算符

在本示例中，将选择满足其中一个条件的所有记录。因此，将选择其职务 ID 包含字符串“MAN”或者薪金为 \$10,000 或更高的所有雇员。

OR 真值表

下表显示用 OR 组合两个表达式的结果：

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

使用 NOT 运算符

```
SELECT last_name, job_id
FROM employees
WHERE job_id
      NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

R	LAST_NAME	R	JOB_ID
1	De Haan	AD_VP	
2	Fay	MK_REP	
3	Gietz	AC_ACCOUNT	
4	Hartstein	MK_MAN	
5	Higgins	AC_MGR	
6	King	AD_PRES	
7	Kochhar	AD_VP	
8	Mourgos	ST_MAN	
9	Whalen	AD_ASST	
10	Zlotkey	SA_MAN	

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 NOT 运算符

幻灯片示例中显示其职务 ID 不是 IT_PROG、ST_CLERK 或 SA_REP 的所有雇员的姓氏和职务 ID。

NOT 真值表

下表显示对某个条件应用 NOT 运算符的结果：

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

注：NOT 运算符还可与其它 SQL 运算符（如 BETWEEN、LIKE 和 NULL）一起配合使用。

```
... WHERE job_id NOT IN ('AC_ACCOUNT', 'AD_VP')
... WHERE salary NOT BETWEEN 10000 AND 15000
... WHERE last_name NOT LIKE '%A%'
... WHERE commission_pct IS NOT NULL
```

课程安排

- 使用以下项对行进行限制：
 - WHERE 子句
 - 使用 =、<=、BETWEEN、IN、LIKE 和 NULL 运算符的比较条件
 - 使用 AND、OR 和 NOT 运算符的逻辑条件
- 表达式中运算符的优先级规则
- 使用 ORDER BY 子句对行进行排序
- 替代变量
- DEFINE 和 VERIFY 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

优先级规则

运算符	含义
1	算术运算符
2	连接运算符
3	比较条件
4	IS [NOT] NULL、LIKE、[NOT] IN
5	[NOT] BETWEEN
6	不等于
7	NOT 逻辑条件
8	AND 逻辑条件
9	OR 逻辑条件

可以使用括号更改优先级规则。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

优先级规则

优先级规则确定了表达式求值和计算的顺序。幻灯片表中列出了默认的优先级顺序。但是，通过对需要先计算的表达式使用括号，可以更改默认顺序。

优先级规则

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 'SA_REP'
OR job_id = 'AD_PRES'
AND salary > 15000;
```

1

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000
2	Abel	SA_REP	11000
3	Taylor	SA_REP	8600
4	Grant	SA_REP	7000

```
SELECT last_name, job_id, salary
FROM employees
WHERE (job_id = 'SA_REP'
OR job_id = 'AD_PRES')
AND salary > 15000;
```

2

	LAST_NAME	JOB_ID	SALARY
1	King	AD_PRES	24000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

优先级规则（续）

1. AND 运算符的优先级：示例

在本示例中，有两个条件：

- 第一个条件是职务 ID 为 AD_PRES 且薪金高于 \$15,000。
- 第二个条件是职务 ID 为 SA_REP。

因此，可将该 SELECT 语句解释为：

“如果某雇员是总裁且薪金高于 \$15,000，或者该雇员是销售代表，则选择相应的行。”

2. 使用括号：示例

在本示例中，有两个条件：

- 第一个条件是职务 ID 为 AD_PRES 或 SA_REP。
- 第二个条件是薪金高于 \$15,000。

因此，可将该 SELECT 语句解释为：

“如果某雇员是总裁或销售代表，而且薪金高于 \$15,000，则选择相应的行。”

课程安排

- 使用以下项对行进行限制：
 - WHERE 子句
 - 使用 =、<=、BETWEEN、IN、LIKE 和 NULL 运算符的比较条件
 - 使用 AND、OR 和 NOT 运算符的逻辑条件
- 表达式中运算符的优先级规则
- 使用 ORDER BY 子句对行进行排序
- 替代变量
- DEFINE 和 VERIFY 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ORDER BY 子句

- 使用 ORDER BY 子句可对检索行进行排序：
 - ASC：升序，默认顺序
 - DESC：降序
- ORDER BY 子句位于 SELECT 语句的最后：

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date ;
```

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	King	AD_PRES		90 17-JUN-87
2	Whalen	AD_ASST		10 17-SEP-87
3	Kochhar	AD_VP		90 21-SEP-89
4	Hunold	IT_PROG		60 03-JAN-90
5	Ernst	IT_PROG		60 21-MAY-91
6	De Haan	AD_VP		90 13-JAN-93

...

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

使用 ORDER BY 子句

查询结果中返回行的顺序是未定义的。可使用 ORDER BY 子句对这些行进行排序。但是，如果使用 ORDER BY 子句，该子句必须是 SQL 语句的最后一个子句。此外，可以指定表达式、别名或列位置作为排序条件。

语法

```
SELECT          expr
FROM            table
[WHERE          condition(s)]
[ORDER BY {column, expr, numeric_position} [ASC|DESC]];
```

在该语法中：

ORDER BY	指定检索行的显示顺序
ASC	按升序对行进行排序（此为默认顺序）
DESC	按降序对行进行排序

如果没有使用 ORDER BY 子句，则排序顺序未定义，因此如果同一查询执行两次，Oracle Server 每次提取行的顺序可能不同。使用 ORDER BY 子句可按特定顺序显示行。

注：使用关键字 NULLS FIRST 或 NULLS LAST 可指定包含空值的返回行是出现在排序序列的最前面还是最后。

排序

- 按降序排序:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC ;
```

1

- 按列别名排序:

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

排序

默认排序顺序是升序:

- 从小到大显示数值, 例如 1 到 999。
- 由远到近显示日期值, 例如, 01-JAN-92 显示在 01-JAN-95 之前。
- 按字母顺序显示字符值, 例如, 先显示 A, 最后显示 Z。
- 对于升序排序, 空值显示在最后, 对于降序排序, 则显示在最前面。
- 也可以按不在 SELECT 列表之内的列进行排序。

示例:


- 要使行的显示顺序反转, 请在 ORDER BY 子句中在列名之后指定 DESC 关键字。
幻灯片示例中按最近聘用的雇员对结果进行排序。
- 此外, 还可在 ORDER BY 子句中使用列别名。幻灯片示例中按年薪对数据进行排序。

注: 请勿将此处用于按降序排序的 DESC 关键字与用于描述表结构的 DESC 关键字混淆。

排序


- 使用列的数字位置进行排序：

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY 3;
```



- 按多个列进行排序：

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

排序（续）

示例：

- 通过在 SELECT 子句中指定列的数字位置可对查询结果排序。幻灯片示例中按 department_id 对结果进行排序，因为该列在 SELECT 子句中位于第三位。
- 可按多个列对查询结果进行排序。排序的列限制是给定表中的列数。在 ORDER BY 子句中，可以指定列并使用逗号分隔列名。如果希望反转列顺序，请在列名后指定 DESC。幻灯片中显示的查询示例结果按 department_id 的升序、salary 的降序排序。

课程安排

- 使用以下项对行进行限制：
 - WHERE 子句
 - 使用 =、<=、BETWEEN、IN、LIKE 和 NULL 运算符的比较条件
 - 使用 AND、OR 和 NOT 运算符的逻辑条件
- 表达式中运算符的优先级规则
- 使用 ORDER BY 子句对行进行排序
- **替代变量**
- DEFINE 和 VERIFY 命令

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

替代变量



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

替代变量

到目前为止，所有的 SQL 语句都使用预定义的列、条件及其值来执行。假定您需要创建一个查询，用于列出从事各种工作的员工，而不仅仅列出 job_ID 为 SA_REP 的那些员工。您可以对 WHERE 子句进行编辑，以便在每次运行该命令时都提供一个不同值，但是有一个更简便的方法。

通过使用替代变量来代替 WHERE 子句中的确切值，可以针对不同值运行同一查询。

通过使用替代变量，可以创建提示用户输入自己值的报表，以限制返回数据的范围。可将替代变量嵌入到命令文件或一条 SQL 语句中。变量可视为值的临时存储器。运行语句后，就会替代存储值。

替代变量

- 使用替代变量：
 - 使用单与号 (&) 及双与号 (&&) 替代变量可临时存储值
- 可将替代变量作为以下项的补充：
 - WHERE 条件
 - ORDER BY 子句
 - 列表表达式
 - 表名
 - 整个 SELECT 语句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

替代变量（续）

可使用单与号 (&) 替代变量来临时存储值。

还可使用 DEFINE 命令来预定义变量。DEFINE 用于创建变量并为其赋值。

限制数据范围：示例

- 仅报告当前季度或指定日期范围的数字
- 仅报告与请求报表的用户相关的数据
- 仅显示指定部门内的人员

其它交互影响

交互影响并不只限于影响 WHERE 子句的直接用户交互。还可使用相同的原理实现其它目标，例如：

- 从文件而不是从人员获取输入值
- 将一个 SQL 语句的值传递给另一个 SQL 语句

注：SQL Developer 和 SQL*Plus 都支持替代变量和 DEFINE/UNDEFINE 命令。但 SQL Developer 和 SQL*Plus 都不支持对用户输入进行验证检查（数据类型除外）。如果应用于部署至用户的脚本，则替代变量可能会被暗中破坏以实现 SQL 注入攻击。

使用单与号替代变量

使用以与号 (&) 为前缀的变量可提示用户输入一个值：

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```



版权所有 © 2010, Oracle。保留所有权利。

使用单与号替代变量

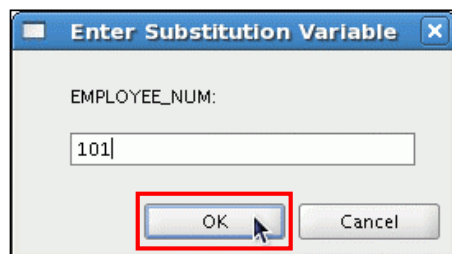
运行报表时，用户通常需要对动态返回的数据进行限制。SQL*Plus 或 SQL Developer 通过用户变量提供了这种灵活性。使用与号 (&) 可标识 SQL 语句中的每个变量。但是，您并不需要定义每个变量的值。

记号	说明
<i>&user_variable</i>	表示 SQL 语句中的一个变量，如果该变量不存在，则 SQL*Plus 或 SQL Developer 将提示用户输入一个值（新变量在用过后便被丢弃。）

幻灯片示例中为雇员编号创建了一个 SQL Developer 替代变量。执行语句后，SQL Developer 会提示用户输入一个雇员编号，然后会显示该雇员的雇员编号、姓氏、薪金以及部门编号。

通过使用单与号，每次执行命令时如果变量不存在，用户都会得到一个提示。

使用单与号替代变量



The dialog box titled "Enter Substitution Variable" has a close button (X) in the top right corner. It contains a label "EMPLOYEE_NUM:" followed by a text input field containing the value "101". Below the input field, the "OK" button is highlighted with a red rectangle, and a mouse cursor is pointing at it. A "Cancel" button is located to the right of the "OK" button.

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用单与号替代变量（续）

当 SQL Developer 检测到 SQL 语句中包含与号时，就会提示您为 SQL 语句中指定的替代变量输入一个值。

输入值之后，单击“OK（确定）”按钮，在 SQL Developer 会话的“Results（结果）”选项卡上就会显示有关结果。

使用替代变量指定字符值和日期值

对日期值和字符值使用单引号：

```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```

	LAST_NAME	DEPARTMENT_ID	SALARY*12
1	Hunold	60	108000
2	Ernst	60	72000
3	Lorentz	60	50400

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

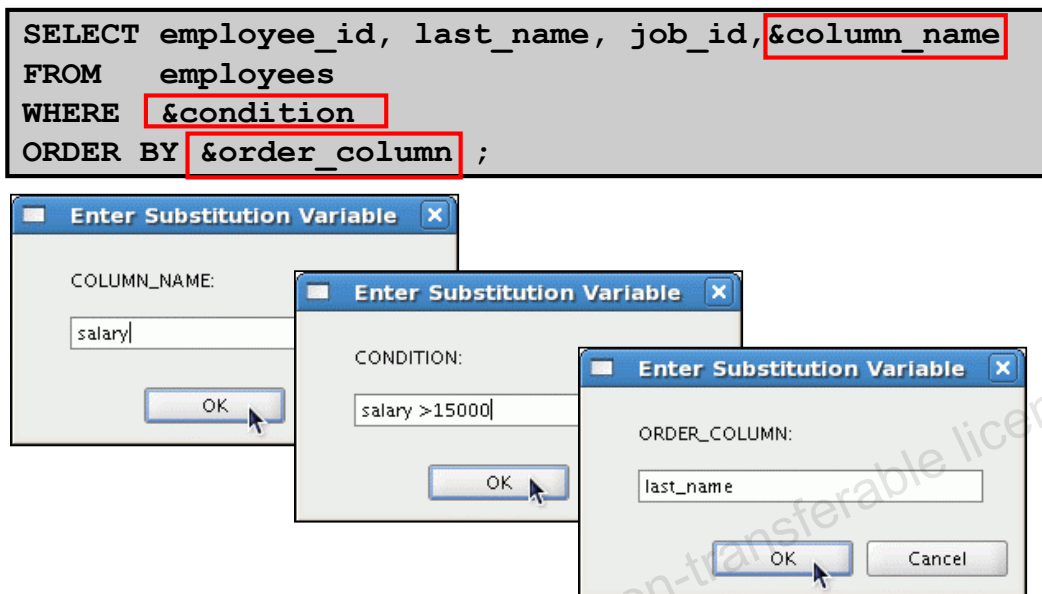
使用替代变量指定字符值和日期值

在 WHERE 子句中，日期值和字符值必须放在单引号中。这一规则也适用于替代变量。

将变量括在单引号中，放在 SQL 语句本身中。

本幻灯片显示一个查询，它根据 SQL Developer 替代变量的职位值来检索所有雇员的雇员姓名、部门编号和年薪。

指定列名、表达式和文本



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

指定列名、表达式和文本

不但可以在 SQL 语句的 WHERE 子句中使用替代变量，还可以对列名、表达式或文本使用替代变量。

示例：

幻灯片示例中显示用户在运行时指定 EMPLOYEES 表中的雇员编号、姓氏、职位以及任何其它列。对于 SELECT 语句中的每一个替代变量，系统都会提示用户输入一个值，然后单击“OK（确定）”继续进行操作。

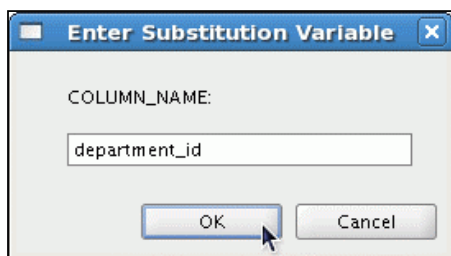
如果没有为替代变量输入值，则在执行上述语句时就会出现错误。

注：替代变量可用于 SELECT 语句中的任何位置，只是不能作为在命令提示符处输入的第一个字。

使用双与号替代变量

如果要重复使用变量值而不每次都显示提示，请使用双与号 (&&)：

```
SELECT  employee_id, last_name, job_id, &&column_name
FROM    employees
ORDER BY &column_name ;
```



Enter Substitution Variable

COLUMN_NAME:

department_id

OK Cancel

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用双与号替代变量

如果要重复使用变量值而不每次都显示提示，请使用双与号 (&&) 替代变量：用户只会看见一次输入值的提示。在幻灯片示例中，只要求用户为变量 column_name 提供一次值。由用户提供的值 (department_id) 既用于显示数据又用于对数据进行排序。如果再次运行查询，将不再提示用户输入变量值。

SQL Developer 将使用 DEFINE 命令提供的值存储起来，只要引用该变量名，SQL Developer 就会再次使用该值。用户变量被存储起来后，需要使用 UNDEFINE 命令将其删除：

```
UNDEFINE column_name
```

课程安排

- 使用以下项对行进行限制：
 - WHERE 子句
 - 使用 =、<=、BETWEEN、IN、LIKE 和 NULL 运算符的比较条件
 - 使用 AND、OR 和 NOT 运算符的逻辑条件
- 表达式中运算符的优先级规则
- 使用 ORDER BY 子句对行进行排序
- 替代变量
- **DEFINE 和 VERIFY 命令**


ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 DEFINE 命令

- 使用 DEFINE 命令可创建变量并为其赋值。
- 使用 UNDEFINE 命令可删除变量。

```
DEFINE employee_num = 200  
  
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num ;  
  
UNDEFINE employee_num
```



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 DEFINE 命令

该示例显示使用 DEFINE 命令为雇员编号创建一个替代变量。在运行时，它将显示该雇员的雇员编号、姓名、薪金和部门编号。

因为该变量是使用 SQL Developer DEFINE 命令创建的，因此系统不会提示用户输入雇员编号的值。而是自动替换 SELECT 语句中定义的变量值。

EMPLOYEE_NUM 替代变量将显示在会话中，直到用户取消其定义或者退出 SQL Developer 会话为止。

使用 VERIFY 命令

在 SQL Developer 使用值替换替代变量前后，使用 VERIFY 命令可切换替代变量的显示：

SET VERIFY ON

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```

EMPLOYEE_NUM:
200

OK Cancel

Results Script Output Explain Autotrace DBMS Output OWA Output

```
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = 200
```

EMPLOYEE_ID	LAST_NAME	SALARY
200	Whalen	4400

1 rows selected

使用 VERIFY 命令

要确认 SQL 语句中的更改，请使用 VERIFY 命令。设置 SET VERIFY ON 会强制 SQL Developer 在使用值替换替代变量之后显示命令的文本。要查看 VERIFY 输出，应使用 SQL 工作表中的“Run Script（运行脚本）” (F5) 图标。SQL Developer 使用值替换替代变量之后，将在“Script Output（脚本输出）”选项卡上显示命令文本，如幻灯片所示。

幻灯片示例在 SQL 语句中显示了 EMPLOYEE_ID 列的新值，然后显示输出。

SQL*Plus 系统变量

SQL*Plus 使用不同的系统变量来控制工作环境。其中一个变量就是 VERIFY。要获得所有系统变量的完整列表，请在 SQL*Plus 命令提示符下输入 SHOW ALL 命令。

小测验

以下哪些运算符在 WHERE 子句中是有效的？

1. >=
2. IS NULL
3. !=
4. IS LIKE
5. IN BETWEEN
6. <>

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、2、3、6

小结

在本课中，您应该已经学会：

- 使用 WHERE 子句来限制输出行：
 - 使用比较条件
 - 使用 BETWEEN、IN、LIKE 和 NULL 运算符
 - 应用 AND、OR 和 NOT 逻辑运算符
- 使用 ORDER BY 子句对输出行进行排序：

```
SELECT *|{[DISTINCT] column|expression [alias],...}  
FROM table  
[WHERE condition(s)]  
[ORDER BY {column, expr, alias} [ASC|DESC]];
```

- 在运行时使用 & 替代变量对输出进行限制和排序

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对 SELECT 语句返回的行进行限制和排序。还应该学会了如何运用各种运算符和条件。

通过使用替代变量，可以增加 SQL 语句的灵活性。这样可以在查询运行时提示用户输入某些行的过滤条件。

练习 2：概览

本练习包含以下主题：

- 选择数据并更改行的显示顺序
- 使用 WHERE 子句对行进行限制
- 使用 ORDER BY 子句对行进行排序
- 使用替代变量增加 SQL SELECT 语句的灵活性

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 2：概览

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。

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.

使用单行函数定制输出

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 SQL 提供的各类函数
- 在 SELECT 语句中使用字符、数字和日期函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

函数可使基本查询块的功能更强大，还可用于处理数据值。介绍函数的课程共有两课，这是第一课。主要介绍单行字符、数字和日期函数。

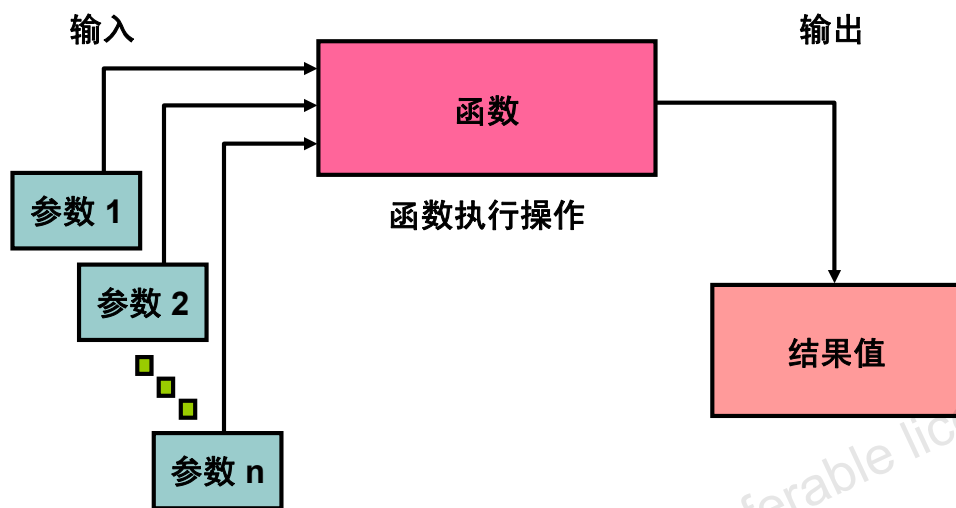
课程安排

- 单行 SQL 函数
- 字符函数
- 数字函数
- 处理日期
- 日期函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL 函数



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

SQL 函数

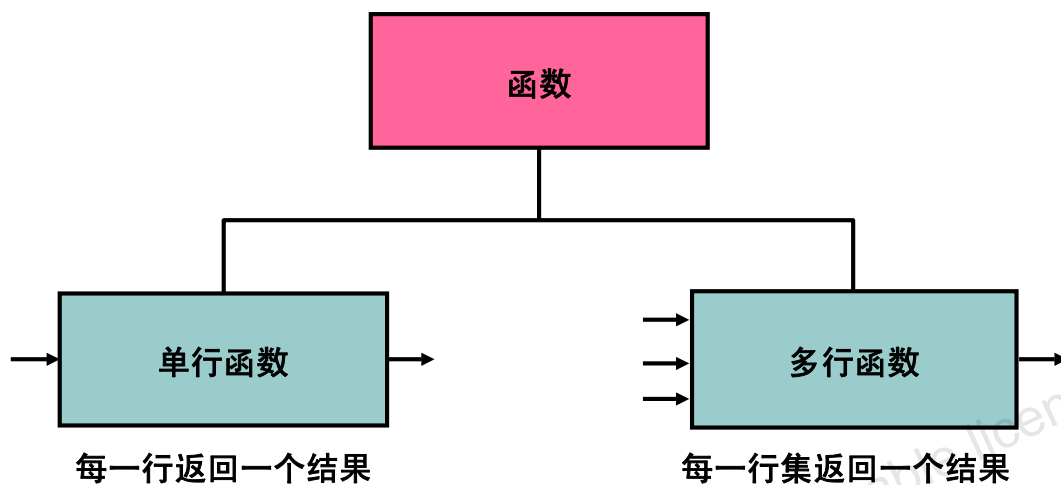
函数是 SQL 的一项非常强大的功能，可用于执行以下操作：

- 执行数据计算
- 修改单个数据项
- 处理成组行的输出
- 设置日期和数字的显示格式
- 转换列数据类型

SQL 函数有时接受多个参数，但始终返回一个值。

注：如果要判断一个函数是不是符合 SQL:2003 的函数，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Oracle Compliance To Core SQL:2003”一节。

两种类型的 SQL 函数



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

两种类型的 SQL 函数

有两种类型的函数：

- 单行函数
- 多行函数

单行函数

这些函数仅对单行进行处理，为每行返回一个结果。单行函数具有多种不同类型。本课将介绍以下几种函数：

- 字符
- 数字
- 日期
- 转换
- 常规

两种类型的 SQL 函数（续）

多行函数

这些函数可以处理成组的行，为每组行返回一个结果。这些函数又称为组函数（将在“使用组函数报告聚集数据”一课中予以介绍）。

注：有关详细信息以及可用函数及其语法的完整列表，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Functions”一节。

单行函数

单行函数：

- 处理数据项
- 接受参数并返回一个值
- 对每个返回行进行处理
- 为每行返回一个结果
- 可能会修改数据类型
- 可以嵌套
- 接受参数，这些参数可以是列或表达式

```
function_name [(arg1, arg2,...)]
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

单行函数

单行函数用于处理数据项。其接受一个或多个参数，并对查询返回的每一行返回一个值。参数可以是下列对象之一：

- 用户提供的常量
- 变量值
- 列名
- 表达式

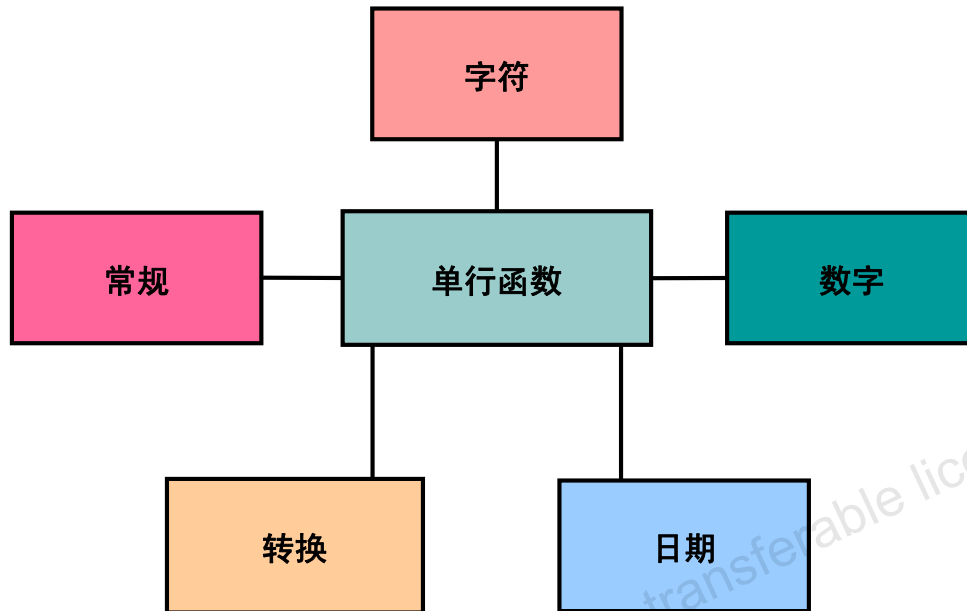
单行函数的特点包括：

- 对查询中返回的每一行进行处理
- 为每行返回一个结果
- 可能会返回一个与所引用类型不同的数据值
- 可能需要一个或多个参数
- 可用于 SELECT、WHERE 和 ORDER BY 子句中；也可以嵌套。

在该语法中：

function_name 是函数的名称
arg1, arg2 是函数使用的任意参数，可以是列名称或表达式。

单行函数



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

单行函数（续）

本课将介绍以下单行函数：

- **字符函数：**接受字符输入，可以返回字符值和数字值
- **数字函数：**接受数字输入，可以返回数字值
- **日期函数：**对 DATE 数据类型的值进行处理（所有日期函数都会返回一个 DATE 数据类型的值，只有 MONTHS_BETWEEN 函数返回一个数字。）

以下单行函数将在“使用转换函数和条件表达式”一课中予以介绍：

- **转换函数：**将值从一种数据类型转换为另一种数据类型
- **常规函数：**
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
 - CASE
 - DECODE

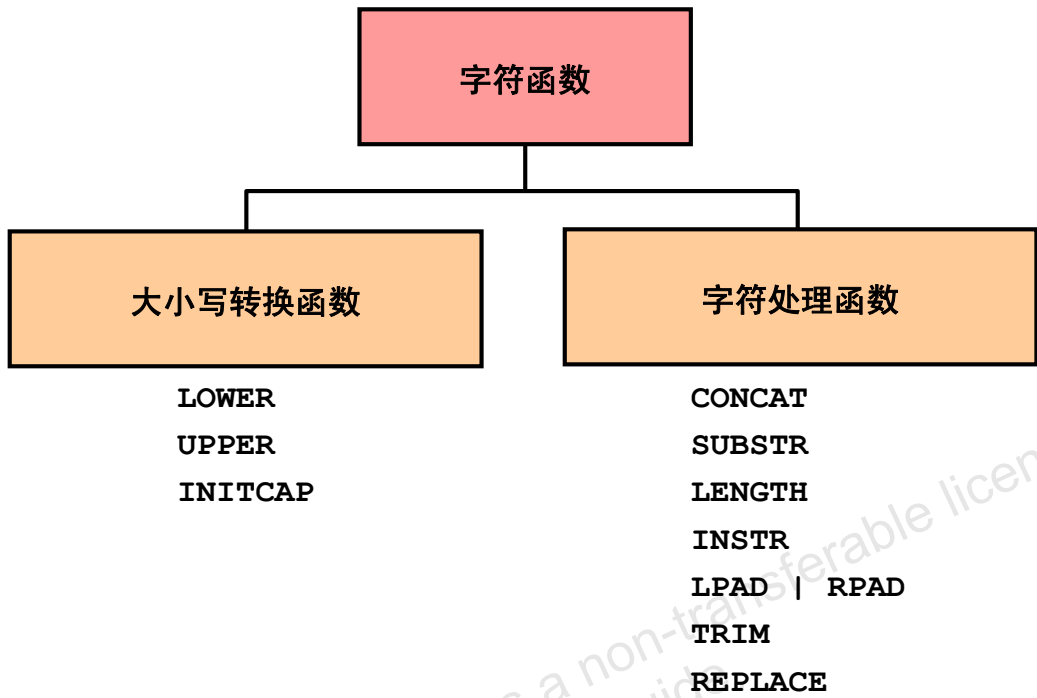
课程安排

- 单行 SQL 函数
- 字符函数
- 数字函数
- 处理日期
- 日期函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

字符函数



ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

字符函数

单行字符函数接受的输入是字符数据，可以返回字符值和数字值。字符函数可以分为以下几类：

- 大小写转换函数
- 字符处理函数

函数	用途
LOWER(<i>column expression</i>)	将字母字符值转换为小写。
UPPER(<i>column expression</i>)	将字母字符值转换为大写。
INITCAP(<i>column expression</i>)	将每个单词首字母的字母字符值转换为为大写，其它所有字母均为小写。
CONCAT(<i>column1 expression1, column2 expression2</i>)	将第一个字符值连接到第二个字符值，与连接运算符 () 等效。
SUBSTR(<i>column expression, m[,n]</i>)	从字符值中第 <i>m</i> 个字符开始返回指定的字符，长度为 <i>n</i> 个字符（如果 <i>m</i> 为负数，则从字符值的末尾开始计算。如果省略 <i>n</i> ，则返回一直到字符串末尾的所有字符。）。

注：本课中讨论的函数只是可用函数的一部分。

字符函数（续）

函数	用途
LENGTH(<i>column expression</i>)	返回表达式中的字符数。
INSTR(<i>column expression</i> , ' <i>string</i> ', [, <i>m</i>], [<i>n</i>])	返回指定字符串的数字位置。还可以提供一个开始搜索的位置 <i>m</i> 和该字符串的出现次数 <i>n</i> 。 <i>m</i> 和 <i>n</i> 默认为 1，这表示从字符串的起始位置开始搜索并报告该字符串的第一次出现。
LPAD(<i>column expression</i> , <i>n</i> , ' <i>string</i> ') RPAD(<i>column expression</i> , <i>n</i> , ' <i>string</i> ')	返回一个表达式，左边使用一个字符表达式填充到 <i>n</i> 个字符的长度。 返回一个表达式，右边使用一个字符表达式填充到 <i>n</i> 个字符的长度。
TRIM(<i>leading trailing both</i> , <i>trim_character</i> FROM <i>trim_source</i>)	使您可以截去字符串的首字符或尾字符（或者两者都截去）。如果 <i>trim_character</i> 或 <i>trim_source</i> 是一个字符文字，则必须将其放在单引号内。 这是 Oracle8i 和更高版本中提供的一项功能。
REPLACE(<i>text</i> , <i>search_string</i> , <i>replacement_string</i>)	搜索字符串的文本表达式，如果找到，则使用指定的替代字符串替换它。

注：以下是一些完全或部分符合 SQL:2003 的函数：

UPPER
LOWER
TRIM
LENGTH
SUBSTR
INSTR

有关详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中的“Oracle Compliance To Core SQL:2003”一节。

大小写转换函数

以下函数用于转换字符串的大小写：

函数	结果
LOWER('SQL Course')	sql course
UPPER('SQL Course')	SQL COURSE
INITCAP('SQL Course')	Sql Course

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

大小写转换函数

LOWER、UPPER 和 INITCAP 是三个大小写转换函数。

- LOWER：将大小写混合或大写的字符串转换为小写
- UPPER：将大小写混合或小写的字符串转换为大写
- INITCAP：将每个单词的首字母转换为大写，其余字母保留为小写

```
SELECT 'The job id for '||UPPER(last_name)||' is '
       ||LOWER(job_id) AS "EMPLOYEE DETAILS"
FROM   employees;
```

	EMPLOYEE DETAILS
1	The job id for ABEL is sa_rep
2	The job id for DAVIES is st_clerk
3	The job id for DE HAAN is ad_vp
4	The job id for ERNST is it_prog
5	The job id for FAY is mk_rep
6	The job id for GIETZ is ac_account

...

使用大小写转换函数

使用以下语句可显示雇员 Higgins 的雇员编号、姓名和部门编号：

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  last_name = 'higgins';
```

0 rows selected

```
SELECT employee_id, last_name, department_id
FROM   employees
WHERE  LOWER(last_name) = 'higgins';
```

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	205	Higgins	110

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用大小写转换函数

幻灯片示例中显示雇员 Higgins 的雇员编号、姓名和部门编号。

第一个 SQL 语句的 WHERE 子句将雇员姓名指定为 higgins。因为 EMPLOYEES 表中的所有数据都是以正常大小写形式存储的，所以姓名 higgins 在表中没有找到匹配项，因此不会选择任何行。

第二个 SQL 语句中的 WHERE 子句指定将 EMPLOYEES 表中的雇员姓名与 higgins 进行比较，但是先将 LAST_NAME 列转换为小写，然后再进行比较。因为两个姓名都是小写的，因此找到了匹配项，从而选择了一行。可以按以下方式重写 WHERE 子句，产生的结果相同：

```
...WHERE last_name = 'Higgins'
```

输出中姓名的显示格式与其存储在数据库中的一样。要以大写形式显示姓名，可以在 SELECT 语句中使用 UPPER 函数。

```
SELECT employee_id, UPPER(last_name), department_id
FROM   employees
WHERE  INITCAP(last_name) = 'Higgins';
```

字符处理函数

下面的函数用于处理字符串：

函数	结果
CONCAT('Hello', 'World')	HelloWorld
SUBSTR('HelloWorld',1,5)	Hello
LENGTH('HelloWorld')	10
INSTR('HelloWorld', 'W')	6
LPAD(salary,10,'*')	*****24000
RPAD(salary, 10, '*')	24000*****
REPLACE ('JACK and JUE', 'J', 'BL')	BLACK and BLUE
TRIM('H' FROM 'HelloWorld')	elloWorld

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

字符处理函数

CONCAT、SUBSTR、LENGTH、INSTR、LPAD、RPAD 和 TRIM 是本课中要介绍的字符处理函数。

- CONCAT：将值联接在一起（CONCAT 函数中只能使用两个参数）
- SUBSTR：提取确定长度的字符串
- LENGTH：以数字值的形式显示字符串的长度
- INSTR：查找指定字符串的数字位置
- LPAD：返回一个表达式，左边使用一个字符表达式填充到 n 个字符的长度
- RPAD：返回一个表达式，右边使用一个字符表达式填充到 n 个字符的长度
- TRIM：截去字符串首字符或尾字符（或者两者都截去）（如果 *trim_character* 或 *trim_source* 是一个字符文字，则必须将其放在单引号内）

注：可以将 UPPER 和 LOWER 等函数与 & 替代变量组合使用。例如，如果使用 UPPER('&job_title')，用户就不必以特定的大小写形式输入职务了。

使用字符处理函数

The diagram illustrates the use of character processing functions in SQL. It shows a query and its results with annotations 1, 2, and 3.

Query:

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       job_id, LENGTH (last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
```

Results Table:

	EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
1	202	PatFay	MK_REP	3	2
2	174	EllenAbel	SA_REP	4	0
3	176	JonathonTaylor	SA_REP	6	2
4	178	KimberelyGrant	SA_REP	5	3

Annotations:

- 1: Points to the `CONCAT(first_name, last_name) NAME` expression in the query.
- 2: Points to the `LENGTH (last_name)` expression in the query.
- 3: Points to the `INSTR(last_name, 'a') "Contains 'a'?"` expression in the query.

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用字符处理函数

幻灯片示例中显示从其职务 ID 的第四个位置开始包含字符串 REP 的所有雇员的以下信息：联接在一起的名字和姓氏、雇员姓氏的长度，以及字母 a 在雇员姓氏中的数字位置。

示例：

修改幻灯片中的 SQL 语句，以显示姓氏以字母 “n” 为结尾的雇员的数据。

```
SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH (last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
```

	EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
1	102	LexDe Haan	7	5
2	200	JenniferWhalen	6	3
3	201	MichaelHartstein	9	2

课程安排

- 单行 SQL 函数
- 字符函数
- **数字函数**
- 处理日期
- 日期函数

ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

数字函数

- ROUND: 将值舍入到指定的小数位
- TRUNC: 将值截断到指定的小数位
- MOD: 返回除法运算的余数

函数	结果
ROUND (45.926, 2)	45.93
TRUNC (45.926, 2)	45.92
MOD (1600, 300)	100

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

数字函数

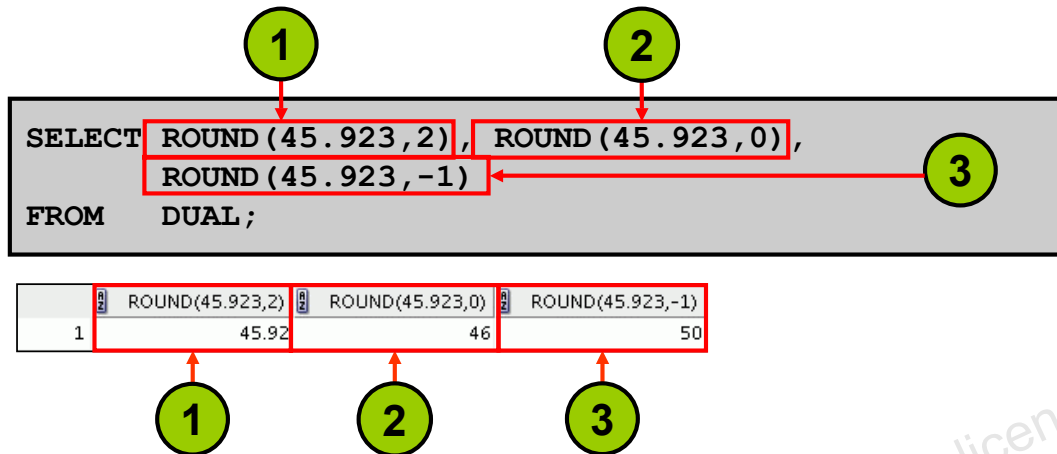
数字函数接受数字形式的输入并返回数字值。本节将介绍一些数字函数。

函数	用途
ROUND (column expression, n)	将列、表达式或值舍入到 n 位小数位，如果省略了 n ，则不保留小数位（如果 n 为负数，则会舍入小数点左边的数字）
TRUNC (column expression, n)	将列、表达式或值截断到 n 位小数位，如果省略了 n ，则 n 默认为零
MOD (m, n)	返回 m 除以 n 之后的余数

注：该列表仅列出了部分可用数字函数。

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

使用 ROUND 函数



DUAL 是可用于查看函数和计算结果的公用表。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ROUND 函数

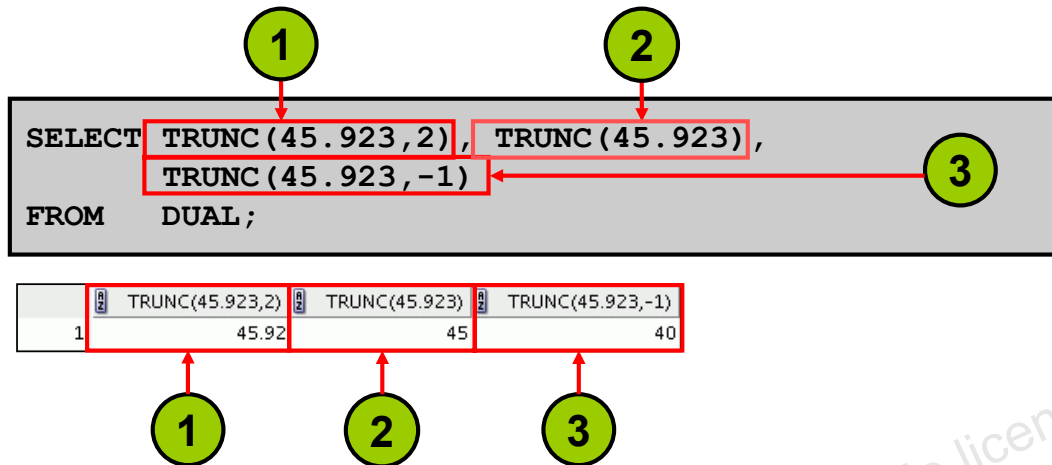
ROUND 函数用于将列、表达式或值舍入到 n 位小数位。如果第二个参数为 0 或者缺失，则会将值舍入为整数。如果第二个参数为 2，则会将值舍入到 2 位小数位。相反，如果第二个参数为 -2，则会将值舍入到小数点左边 2 位（即舍入到最近的百位）。

ROUND 函数还可与日期函数一起使用。在本课稍后您将看到相应的示例。

DUAL 表

DUAL 表属于 SYS 用户，但所有用户都可以访问它。它包含一个 DUMMY 列和一个值为 X 的行。如果某个值只需返回一次，例如，不是从含有用户数据的表中导出的常量值、伪列值或表达式值，则 DUAL 表非常有用。DUAL 表通常用于保持 SELECT 子句语法的完整性，因为 SELECT 和 FROM 子句都是必需的，而有些计算并不需要从实际表中进行选择。

使用 TRUNC 函数



ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

使用 TRUNC 函数

TRUNC 函数用于将列、表达式或值截断到 n 位小数位。

TRUNC 函数使用参数的方式与 ROUND 函数非常类似。如果第二个参数为 0 或者缺失，则会将值截断为整数。如果第二个参数为 2，则会将值截断到 2 位小数位。相反，如果第二个参数为 -2，则会将值截断到小数点左边 2 位。如果第二个参数为 -1，则会将值截断到小数点左边 1 位。

与 ROUND 函数一样，TRUNC 函数也可与日期函数一起使用。

使用 MOD 函数

针对职务为销售代表的所有雇员，计算薪金除以 5,000 后的余数。

```
SELECT last_name, salary, MOD(salary, 5000)
FROM   employees
WHERE  job_id = 'SA_REP';
```

	LAST_NAME	SALARY	MOD(SALARY,5000)
1	Abel	11000	1000
2	Taylor	8600	3600
3	Grant	7000	2000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 MOD 函数

MOD 函数将返回第一个参数除以第二个参数之后的余数。在幻灯片示例中，针对职务 ID 为 SA_REP 的所有雇员计算薪金除以 5,000 后的余数。

注：MOD 函数经常用于确定一个值是奇数还是偶数。MOD 函数也是 Oracle 散列函数。

课程安排

- 单行 SQL 函数
- 字符函数
- 数字函数
- **处理日期**
- 日期函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

处理日期

- Oracle DB 以内部数字格式存储日期：世纪、年、月、日、小时、分钟和秒。
- 默认的日子显示格式为 DD-MON-RR。
 - 通过仅指定年份的后两位，可以在 20 世纪存储 21 世纪的日期
 - 同样，也可以在 21 世纪存储 20 世纪的日期

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date < '01-FEB-88';
```

	LAST_NAME	HIRE_DATE
1	Whalen	17-SEP-87
2	King	17-JUN-87

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

处理日期

Oracle DB 以内部数字格式存储日期，以此代表世纪、年、月、日、小时、分钟和秒。

任何日期的默认显示格式和输出格式均为 DD-MON-RR。有效的 Oracle 日期介于公元前 4712 年 1 月 1 日和公元 9999 年 12 月 31 日之间。

在幻灯片示例中，HIRE_DATE 列输出以默认格式 DD-MON-RR 显示。但是，在数据库中并不以这种格式存储日期，而是存储了日期和时间的所有组成部分。所以，尽管 17-JUN-87 之类的 HIRE_DATE 只显示出日、月和年，但是数据库中还存在与该日期相关联的时间和世纪信息。完整数据可能是 June 17, 1987, 5:10:43 PM。

RR 日期格式

当前年份	指定的日期	RR 格式	YY 格式
1995	27-OCT-95	1995	1995
1995	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2001	27-OCT-95	1995	2095

		如果指定的两位数年份为:	
		0-49	50-99
如果当前年份的两位数为:	0-49	返回日期在当前世纪中	返回日期在当前世纪的前一个世纪中
	50-99	返回日期在当前世纪的下一个世纪中	返回日期在当前世纪中

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

RR 日期格式

RR 日期格式与 YY 元素类似，但是可以使用它指定不同的世纪。请使用 RR 日期格式元素代替 YY，这样返回值的世纪会根据指定的两位数年份和当前年份的后两位数而变化。幻灯片表中汇总了 RR 元素的行为。

当前年份	给定的日期	解释结果 (RR)	解释结果 (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017
2048	27-OCT-52	1952	2052
2051	27-OCT-47	2147	2047

请注意显示在上表倒数两行中的值。当接近世纪中期时，RR 的行为大概不是您所希望看到的。

此数据在内部按以下格式进行存储：

CENTURY	YEAR	MONTH	DAY	HOUR	MINUTE	SECOND
19	87	06	17	17	10	43

RR 日期格式（续）**世纪和 2000 年**

将包含日期列的记录插入到表中时，系统会从 SYSDATE 函数中获取世纪信息。但是，在屏幕上显示日期列时，默认情况下却不显示世纪的相应部分。

DATE 数据类型用 2 个字节存储年份信息：1 个字节存储世纪，1 个字节存储年。无论是否指定或显示世纪值，其始终会包含在内。这种情况下，RR 决定 INSERT 中世纪的默认值。

使用 SYSDATE 函数

SYSDATE 是返回以下对象的函数：

- 日期
- 时间

```
SELECT sysdate  
FROM dual;
```

SYSDATE
1 10-JUN-09

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 SYSDATE 函数

SYSDATE 是一个日期函数，它返回当前数据库服务器的日期和时间。可以像使用任何其它列名称一样使用 SYSDATE。例如，通过从表中选择 SYSDATE，可以显示当前日期。通常从称为 DUAL 的公用表中选择 SYSDATE。

注：SYSDATE 将返回为数据库所在操作系统设置的当前日期和时间。因此，如果您本人位于澳大利亚，但连接到位于美国 (U.S.) 某地的远程数据库，则 sysdate 函数将返回美国的日期和时间。在这种情况下，可以使用 CURRENT_DATE 函数返回会话时区的当前日期。

将在“Oracle Database 11g: SQL 基础 II”课程中讨论有关 CURRENT_DATE 函数和其它相关时区函数的内容。

与日期有关的运算

- 对日期加上或减去一个数字可以获得一个新的日期值。
- 将两个日期相减可以得出它们之间的天数。
- 将小时数除以 24 可以将小时添加到日期中。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

与日期有关的运算

因为数据库将日期作为数字进行存储，因此可以使用算术运算符（如加和减）执行计算。可以加减数字常量和日期。

可以执行下列操作：

操作	结果	说明
日期 + 数字	日期	为日期添加若干天
日期 - 数字	日期	从日期中减去若干天
日期 - 日期	天数	从一个日期中减去另一个日期
日期 + 数字/24	日期	为日期添加若干小时

使用算术运算符处理日期

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM employees  
WHERE department_id = 90;
```

	LAST_NAME	WEEKS
1	King	1147.102432208994708994708994708995
2	Kochhar	1028.959575066137566137566137566138
3	De Haan	856.102432208994708994708994708995

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用算术运算符处理日期

幻灯片示例中显示部门 90 中所有雇员的姓氏和聘用周数。它用当前日期 (SYSDATE) 减去聘用雇员的日期，然后用所得的结果除以 7 计算出雇员的聘用周数。

注：SYSDATE 是一个 SQL 函数，它用于返回当前日期和时间。在运行 SQL 查询时，返回的结果将随为本地数据库所在操作系统设置的日期和时间而变化。

如果从较早的日期中减去较晚的日期，差值将是一个负数。

课程安排

- 单行 SQL 函数
- 字符函数
- 数字函数
- 处理日期
- 日期函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

日期处理函数

函数	结果
MONTHS_BETWEEN	两个日期之间的月数
ADD_MONTHS	将日历月添加到日期
NEXT_DAY	指定日期之后的下一个日期
LAST_DAY	当月最后一天
ROUND	舍入日期
TRUNC	截断日期

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

日期处理函数

日期函数用于处理 Oracle 日期。所有日期函数都返回一个 DATE 数据类型的值，只有 MONTHS_BETWEEN 函数返回一个数字值。

- MONTHS_BETWEEN(date1, date2): 计算 date1 和 date2 之间的月数。结果可以是正数，也可以是负数。如果 date1 晚于 date2，则结果为正数；如果 date1 早于 date2，则结果为负数。结果中的非整数部分代表月份的一部分。
- ADD_MONTHS(date, n): 将 n 个日历月添加到 date。n 的值必须为整数，但可以为负数。
- NEXT_DAY(date, 'char'): 计算 date 之后一周内下一个指定日 ('char') 的日期。char 的值可以是代表某一天的一个数字或者是一个字符串。
- LAST_DAY(date): 计算包含 date 的月份的最后一天的日期。

日期处理函数（续）

此列表只列出了部分可用日期函数。ROUND 和 TRUNC 数字函数也可用来处理日期值，如下所示：

- `ROUND(date[, 'fmt'])`：返回舍入到由格式样式 *fmt* 指定的单位的 *date*。如果省略格式样式 *fmt*，则 *date* 将舍入到最近的一天。
- `TRUNC(date[, 'fmt'])`：返回包含时间部分的日期 *date*，该日期已截断到由格式样式 *fmt* 指定的单位。如果省略格式样式 *fmt*，则 *date* 将截断到最近的一天。

将在“使用转换函数和条件表达式”一课中详细介绍有关格式样式的内容。

使用日期函数

函数	结果
MONTHS_BETWEEN ('01-SEP-95' , '11-JAN-94')	19.6774194
ADD_MONTHS ('31-JAN-96' , 1)	'29-FEB-96'
NEXT_DAY ('01-SEP-95' , 'FRIDAY')	'08-SEP-95'
LAST_DAY ('01-FEB-95')	'28-FEB-95'

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用日期函数

在幻灯片示例中，ADD_MONTHS 函数在提供的日期值“31-JAN-96”上添加了一个月，因而返回“29-FEB-96”。该函数将 1996 年识别为闰年，因此返回二月份的最后一天。如果将输入日期值更改为“31-JAN-95”，该函数将返回“28-FEB-95”。

例如，显示聘用时间不足 150 个月的所有雇员的雇员编号、聘用日期、聘用月数、六个月复核日期、聘用日期之后的第一个星期五和聘用月份的最后一天。

```
SELECT employee_id, hire_date, MONTHS_BETWEEN (SYSDATE, hire_date)
TENURE, ADD_MONTHS (hire_date, 6) REVIEW, NEXT_DAY (hire_date,
'FRIDAY'), LAST_DAY(hire_date)
FROM employees WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 150;
```

	EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DA...	LAST_DAY...
1	202	17-AUG-97	141.79757989...	17-FEB-98	22-AUG-97	31-AUG-97
2	107	07-FEB-99	124.12016054...	07-AUG-99	12-FEB-99	28-FEB-99
3	124	16-NOV-99	114.82983796...	16-MAY-00	19-NOV-99	30-NOV-99
4	142	29-JAN-97	148.41048312...	29-JUL-97	31-JAN-97	31-JAN-97
5	143	15-MAR-98	134.86209602...	15-SEP-98	20-MAR-98	31-MAR-98
6	144	09-JUL-98	131.05564441...	09-JAN-99	10-JUL-98	31-JUL-98
7	149	29-JAN-00	112.41048312...	29-JUL-00	04-FEB-00	31-JAN-00
8	176	24-MAR-98	134.57177344...	24-SEP-98	27-MAR-98	31-MAR-98
9	178	24-MAY-99	120.57177344...	24-NOV-99	28-MAY-99	31-MAY-99

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

使用 ROUND 函数和 TRUNC 函数处理日期

假设 SYSDATE = '25-JUL-03':

函数	结果
ROUND (SYSDATE , 'MONTH')	01-AUG-03
ROUND (SYSDATE , 'YEAR')	01-JAN-04
TRUNC (SYSDATE , 'MONTH')	01-JUL-03
TRUNC (SYSDATE , 'YEAR')	01-JAN-03

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ROUND 函数和 TRUNC 函数处理日期

可以使用 ROUND 和 TRUNC 函数处理数字和日期值。在处理日期时，这些函数会将日期舍入或截断到指定的格式样式。因此，可以将日期舍入到最近的年份或月份。如果格式样式为 Month，则日期在 1-15 时，返回当前月份的第一天。日期在 16-31 时，返回下一月份的第一天。如果格式样式为 Year，则月份在 1-6 时，返回当前年份的第一天。月份在 7-12 时，返回下一年份的第一天。

示例：

将在 1997 年进入公司的所有雇员的聘用日期进行比较。使用 ROUND 和 TRUNC 函数显示雇员编号、聘用日期和起始月份。

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
FROM   employees
WHERE  hire_date LIKE '%97';
```

	EMPLOYEE_ID	HIRE_DATE	ROUND(HIRE_DATE,'MONTH')	TRUNC(HIRE_DATE,'MONTH')
1	202	17-AUG-97	01-SEP-97	01-AUG-97
2	142	29-JAN-97	01-FEB-97	01-JAN-97

小测验

在以下关于单行函数的陈述中，哪项是正确的？

1. 处理数据项
2. 接受参数并且每个参数返回一个值
3. 对每个返回行进行处理
4. 每组行返回一个结果
5. 不会修改数据类型
6. 可以嵌套
7. 接受参数，这些参数可以是列或表达式

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、3、6、7

小结

在本课中，您应该已经学会：

- 使用函数执行数据计算
- 使用函数修改单个数据项

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

小结

单行函数可以嵌套到任意层。单行函数可以处理以下内容：

- **字符数据：** LOWER、UPPER、INITCAP、CONCAT、SUBSTR、INSTR、LENGTH
- **数字数据：** ROUND、TRUNC、MOD
- **日期值：** SYSDATE、MONTHS_BETWEEN、ADD_MONTHS、NEXT_DAY、LAST_DAY

请牢记以下内容：

- 日期值也可以使用算术运算符
- ROUND 函数和 TRUNC 函数也可用来处理日期值

SYSDATE 和 DUAL

SYSDATE 是一个日期函数，它会返回当前日期和时间。通常从称为 DUAL 的单行公用表中选择 SYSDATE。

练习 3：概览

本练习包含以下主题：

- 编写一个显示当前日期的查询
- 创建要求使用数字、字符和日期函数的查询
- 计算一个雇员的服务年数和月数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 3：概览

本练习分为几个部分，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。

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.

4

使用转换函数和条件表达式

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 SQL 提供的各类转换函数
- 使用 TO_CHAR、TO_NUMBER 和 TO_DATE 转换函数
- 在 SELECT 语句中应用条件表达式

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

本课重点介绍用于将数据从一种类型转换为另一种类型（例如，从字符数据转换为数字数据）的函数，还讨论 SQL SELECT 语句中的条件表达式。

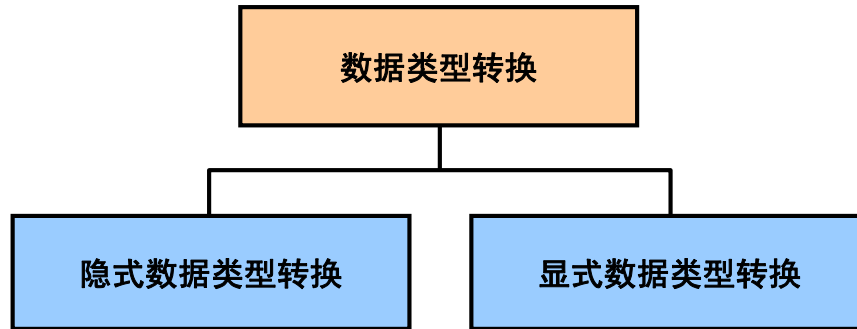
课程安排

- 隐式和显式数据类型转换
- TO_CHAR、TO_DATE、TO_NUMBER 函数
- 嵌套函数
- 常规函数：
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件表达式：
 - CASE
 - DECODE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

转换函数



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

转换函数

除了 Oracle 数据类型，还可以使用美国国家标准协会 (ANSI)、DB2 和 SQL/DS 数据类型定义 Oracle DB 中表的列。但是，Oracle Server 会在内部将这些数据类型转换为 Oracle 数据类型。

在某些情况下，Oracle Server 会收到数据类型与预期的数据类型不同的数据。发生这种情况时，Oracle Server 可自动将该数据转换为预期的数据类型。这种数据类型转换可以由 Oracle Server 隐式完成，也可以由用户显式完成。

隐式数据类型转换将按照下面幻灯片中说明的规则进行。

显式数据类型转换可通过使用转换函数来执行。转换函数用于将一种数据类型的值转换为另一种数据类型的值。通常，函数名称的格式遵循 *data type TO data type* 惯例。第一个数据类型是输入数据类型，第二个数据类型是输出数据类型。

注：尽管可以使用隐式数据类型转换，但仍建议您执行显式数据类型转换以确保 SQL 语句的可靠性。

隐式数据类型转换

在表达式中，Oracle Server 可以自动执行以下转换：

从	至
VARCHAR2 或 CHAR	NUMBER
VARCHAR2 或 CHAR	DATE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

隐式数据类型转换

Oracle Server 可以在表达式中自动执行数据类型转换。例如，表达式 `hire_date > '01-JAN-90'` 将导致字符串 `'01-JAN-90'` 隐式转换为一个日期。因此，表达式中的 VARCHAR2 或 CHAR 值可以隐式转换为数字或日期数据类型。

隐式数据类型转换

对于表达式计算，Oracle Server 可以自动执行以下转换：

从	至
NUMBER	VARCHAR2 或 CHAR
DATE	VARCHAR2 或 CHAR

ORACLE

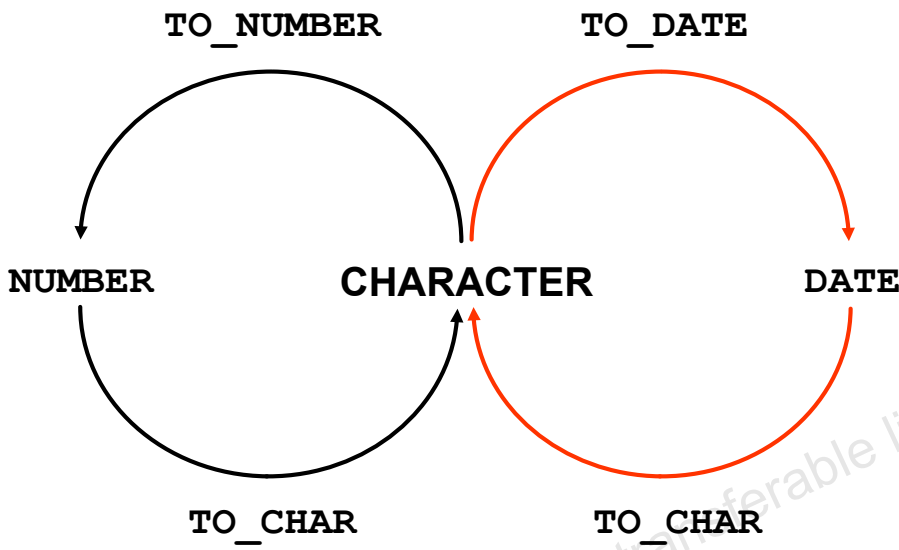
版权所有 © 2010, Oracle。保留所有权利。

隐式数据类型转换（续）

通常，Oracle Server 在需要进行数据类型转换时使用适用于表达式的规则。例如，表达式 `grade = 2` 将导致数字 2 隐式转换为字符串“2”，因为 `grade` 为 `CHAR(2)` 列。

注：只有字符串代表一个有效数字时，CHAR 到 NUMBER 的转换才能成功。

显式数据类型转换



ORACLE

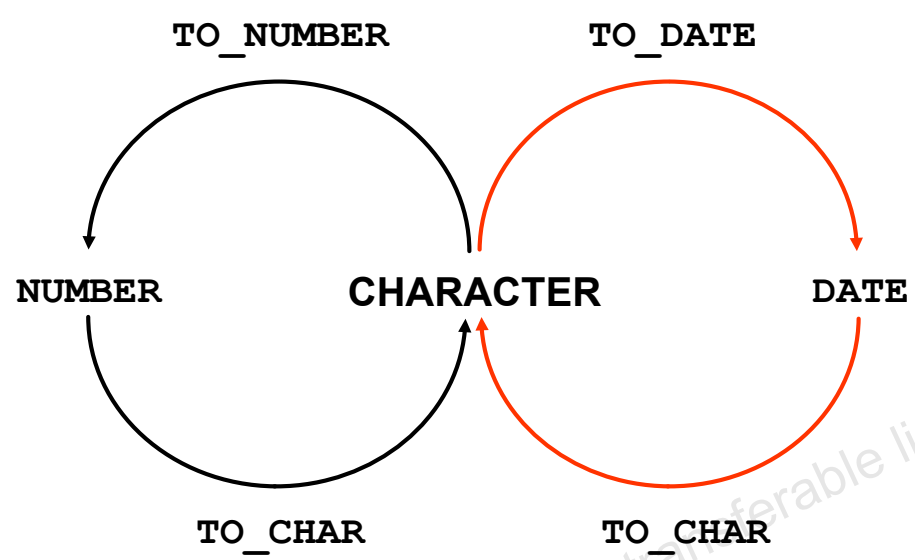
版权所有 © 2010, Oracle。保留所有权利。

显式数据类型转换

SQL 提供了以下三种函数，用于将值从一种数据类型转换为另一种数据类型：

函数	用途
<code>TO_CHAR(<i>number date</i>, [<i>fmt</i>], [<i>nlsparams</i>])</code>	<p>按照格式样式 <i>fmt</i>，将数字或日期值转换为 VARCHAR2 字符串。</p> <p>数字转换： <i>nlsparams</i> 参数指定以下由数字格式元素返回的字符：</p> <ul style="list-style-type: none">• 小数点字符• 组分隔符• 本地货币符号• 国际货币符号 <p>如果省略 <i>nlsparams</i> 或者任何其它参数，则该函数将在会话中使用默认参数值。</p>

显式数据类型转换



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

显式数据类型转换（续）

函数	用途
<code>TO_CHAR(<i>number date</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	日期转换： <i>nlsparms</i> 参数指定返回的月和日名称以及缩写所用的语言。如果省略此参数，则该函数将在会话中使用默认的语言。
<code>TO_NUMBER(<i>char</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	按照可选格式样式 <i>fmt</i> 指定的格式，将包含数字的字符串转换为数字。 对于数字转换， <i>nlsparms</i> 参数在此函数中的作用与它在 <code>TO_CHAR</code> 函数中的作用相同。
<code>TO_DATE(<i>char</i>, [<i>fmt</i>], [<i>nlsparms</i>])</code>	按照指定的 <i>fmt</i> ，将代表日期的字符串转换为日期值。如果省略 <i>fmt</i> ，则格式为 DD-MON-YY。 对于日期转换， <i>nlsparms</i> 参数在此函数中的作用与它在 <code>TO_CHAR</code> 函数中的作用相同。

显式数据类型转换（续）

注：本课中列出的函数列表只是部分可用转换函数。

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

课程安排

- 隐式和显式数据类型转换
- TO_CHAR、TO_DATE、TO_NUMBER 函数
- 嵌套函数
- 常规函数：
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件表达式：
 - CASE
 - DECODE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 TO_CHAR 函数处理日期

```
TO_CHAR(date, 'format_model')
```

格式样式具有以下特点：

- 必须放在单引号内
- 区分大小写
- 可以包含任何有效的日期格式元素
- 具有一个 `fm` 元素，用于删除填充的空格或隐藏前导零
- 与日期值之间用逗号分隔

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 TO_CHAR 函数处理日期

TO_CHAR 可以按照由 *format_model* 指定的格式，将日期时间数据类型转换为数据类型为 VARCHAR2 的值。格式样式是一种字符文字，用于描述字符串中存储的日期时间的格式。例如，字符串 '11-Nov-1999' 的日期时间格式样式为 'DD-Mon-YYYY'。可以使用 TO_CHAR 函数将日期从默认格式转换为指定的格式。

准则

- 格式样式是区分大小写的，而且必须放在单引号内。
- 格式样式可以包含任何有效的日期格式元素。但一定要使用逗号将日期值与格式样式分隔开。
- 输出中的日和月名称会自动用空格填充。
- 要删除填充的空格或隐藏前导零，请使用填充模式 *fm* 元素。

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired
FROM   employees
WHERE  last_name = 'Higgins';
```

	EMPLOYEE_ID	MONTH_HIRED
1	205	06/94

日期格式样式的元素

元素	结果
YYYY	用数字表示的完整年份
YEAR	拼写出的年份（用英文表示）
MM	月份的两数值
MONTH	月份的完整名称
MON	月份的三个字母缩写
DY	一周中某日的三个字母缩写
DAY	一周中某日的完整名称
DD	用数字表示的月份中某日

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

有效日期格式的格式元素示例

元素	说明
SCC 或 CC	世纪；对于公元前的日期，服务器会加上前缀 -
以 YYYY 或 SYYYY 日期表示的年份	年份；对于公元前的日期，服务器会加上前缀 -
YYY 或 YY 或 Y	年份的最后 3 位、2 位或 1 位数字
Y,YYY	在此位置具有逗号的年份
IYYY、IYY、IY、I	基于 ISO 标准的 4 位、3 位、2 位或 1 位数字的年份
SYEAR 或 YEAR	拼写出的年份；对于公元前的日期，服务器会加上前缀 -
BC 或 AD	表示公元前或公元年份
B.C. 或 A.D.	表示带有句点的公元前或公元年份
Q	年份中的季度
MM	月份：两位数的值
MONTH	使用空格填充到 9 个字符长度的月名称
MON	3 个字母缩写形式的月份名称
RM	用罗马数字表示的月份
WW 或 W	年份或月份中的周
DDD 或 DD 或 D	年份、月份或周中的日
DAY	使用空格填充到 9 个字符长度的日名称
DY	3 个字母缩写形式的日名称
J	儒略日，自公元前 4713 年 12 月 31 日以来的天数
IW	符合 ISO 标准的年份中的周（1 至 53）

日期格式样式的元素

- 日期中时间部分的时间元素格式：

HH24:MI:SS AM	15:45:32 PM
---------------	-------------

- 通过将字符串放在双引号内来添加字符串：

DD "of" MONTH	12 of OCTOBER
---------------	---------------

- 为拼写数字指定后缀：

ddspth	fourteenth
--------	------------

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

日期格式样式的元素

使用下表中列出的格式可显示时间信息和文字，并将数字更改为拼写数字。

元素	说明
AM 或 PM	上午或下午标志
A.M. 或 P.M.	带有句点的上午或下午标志
HH 或 HH12 或 HH24	一日中的小时，或为 12 小时制 (1-12)，或为 24 小时制 (0-23)
MI	分钟 (0-59)
SS	秒 (0-59)
SSSSS	午夜之后的秒数 (0-86399)

日期格式样式的元素（续）

其它格式

元素	说明
/ . ,	结果中将出现标点符号。
“of the”	引号中的字符串将出现在结果中。

指定后缀以影响数字的显示

元素	说明
TH	序数（例如，用 DDTH 表示 4TH）
SP	拼写出的数字（例如，用 DDSP 表示 FOUR）
SPTH 或 THSP	拼写出的序数（例如，用 DDSPTH 表示 FOURTH）

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.

使用 TO_CHAR 函数处理日期

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
6	King	17 June 1987
7	Kochhar	21 September 1989
8	De Haan	13 January 1993
9	Hunold	3 January 1990
10	Ernst	21 May 1991

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 TO_CHAR 函数处理日期

幻灯片中的 SQL 语句显示所有雇员的姓氏和聘用日期。聘用日期显示为 “17 June 1987”。

示例：

修改幻灯片中的示例，使其以 “Seventeenth of June 1987 12:00:00 AM” 格式显示日期。

```
SELECT last_name,
       TO_CHAR(hire_date,
               'fmDdspth "of" Month YYYY fmHH:MI:SS AM')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	Whalen	Seventeenth of September 1987 12:00:00 AM
2	Hartstein	Seventeenth of February 1996 12:00:00 AM

...

请注意，月份将遵循指定的格式样式；换句话说，首字母为大写，其它字母为小写。

使用 TO_CHAR 函数处理数字

```
TO_CHAR(number, 'format_model')  

```

下面列出了一些格式元素，可以将其与 TO_CHAR 函数配合使用，以便将数字值显示为字符：

元素	结果
9	代表一个数字
0	强制显示零
\$	放置一个浮动的美元符号
L	使用浮动的本地货币符号
.	显示小数点
,	显示作为千位指示符的逗号



版权所有 © 2010, Oracle。保留所有权利。

使用 TO_CHAR 函数处理数字

如果将数字值作为字符串进行处理，应使用 TO_CHAR 函数将那些数字转换为字符串数据类型，该函数会将 NUMBER 数据类型的值转换为 VARCHAR2 数据类型的值。此方法在进行连接时尤其有用。

使用 TO_CHAR 函数处理数字（续）

数字格式元素

如果要将数字转换为字符数据类型，则可以使用下列格式元素：

元素	说明	示例	结果
9	数字位置（9 的个数确定显示宽度）	999999	1234
0	显示前导零	099999	001234
\$	浮动的美元符号	\$999999	\$1234
L	浮动的本地货币符号	L999999	FF1234
D	在指定的位置返回小数点字符。默认值为句点 (.)	99D99	99.99
.	小数点位于指定的位置	999999.99	1234.00
G	在指定的位置返回组分隔符。可以指定多个以数字格式样式表示的组分隔符	9,999	9G999
,	逗号位于指定的位置	999,999	1,234
MI	减号位于右边（负值）	999999MI	1234-
PR	将负数用括号括起来	999999PR	<1234>
EEEE	科学记数法（格式必须指定 4 个 E）	99.999EEEE	1.234E+03
U	在指定的位置返回“欧元”（或其它）双重币种	U9999	€1234
V	乘以 10 的 n 次方（ $n = V$ 后面 9 的个数）	9999V99	123400
S	返回负值或正值	S9999	-1234 或 +1234
B	将零值显示为空白，而不是 0	B9999.99	1234.00

使用 TO_CHAR 函数处理数字

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM employees  
WHERE last_name = 'Ernst';
```

	SALARY
1	\$6,000.00

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 TO_CHAR 函数处理数字（续）

- 如果数字的位数超过了格式样式中提供的位数，则 Oracle Server 就会用由数字符号 (#) 组成的字符串来替代整个数字。
- Oracle Server 会将存储的小数值舍入到格式样式中指定的小数位数。

使用 TO_NUMBER 和 TO_DATE 函数

- 使用 TO_NUMBER 函数可将字符串转换为数字格式:

```
TO_NUMBER(char[, 'format_model'])
```

- 使用 TO_DATE 函数可将字符串转换为日期格式:

```
TO_DATE(char[, 'format_model'])
```

- 这两个函数都有一个 `fx` 限定符。此限定符指定必须与 TO_DATE 函数的字符参数和日期格式样式完全匹配。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 TO_NUMBER 和 TO_DATE 函数

您可能需要将字符串转换为数字或日期。要完成此任务, 请使用 TO_NUMBER 或 TO_DATE 函数。可根据前面演示的格式元素来选择格式样式。

`fx` 限定符指定必须与 TO_DATE 函数的字符参数和日期格式样式完全匹配:

- 字符参数中的标点和引号中的文本必须与格式样式的相应部分完全匹配(大小写除外)。
- 字符参数不能有额外的空格。如果不使用 `fx`, Oracle Server 就会忽略额外的空格。
- 字符参数中的数字数据必须与格式样式中的相应元素具有相同的位数。如果不使用 `fx`, 字符参数中的数字就会省略前导零。

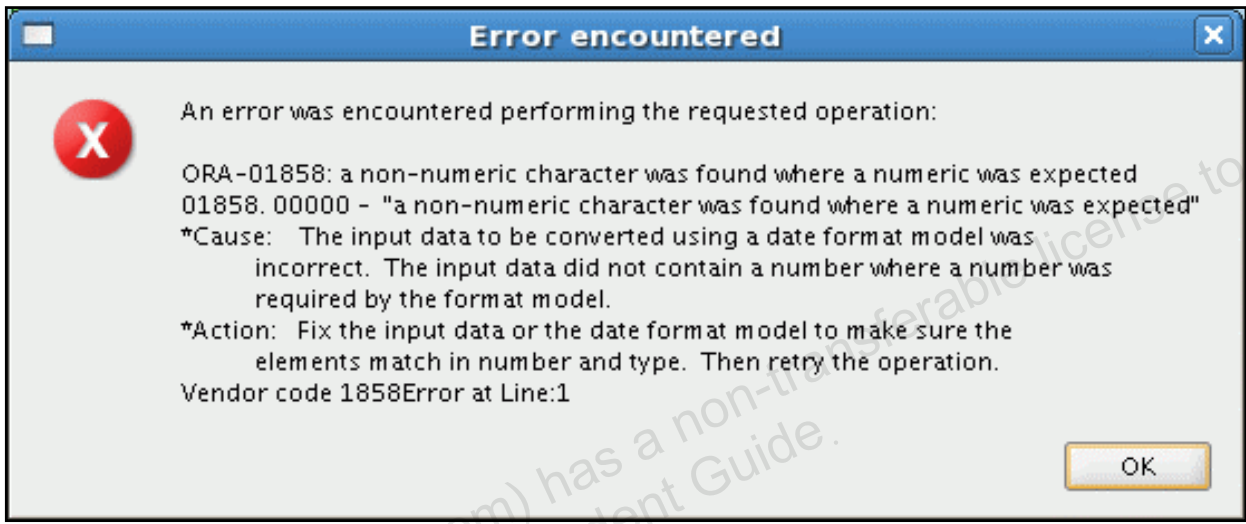
使用 TO_NUMBER 和 TO_DATE 函数 (续)

示例:

显示在 1999 年 5 月 24 日进入公司的所有雇员的姓名和聘用日期。在以下示例中, 月份 May 后和数字 24 前有两个空格。因为使用了 fx 限定符, 要求完全匹配, 所以系统无法识别单词 May 之后的空格。

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May  24, 1999', 'fxMonth DD, YYYY');
```

得出的错误输出如下所示:



要看到输出, 请删除 “May” 和 “24” 之间的额外空格来更正查询。

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');
```

	LAST_NAME	HIRE_DATE
1	Grant	24-MAY-99

将 TO_CHAR 和 TO_DATE 函数 与 RR 日期格式结合使用

查找在 1990 年之前聘用的雇员时，如果使用 RR 日期格式，则无论命令是在 1999 年运行还是在现在运行，此格式都会产生相同的结果：

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

将 TO_CHAR 和 TO_DATE 函数与 RR 日期格式结合使用

查找 1990 年之前聘用的雇员时，可以使用 RR 格式。由于当前年份晚于 1999，因此 RR 格式将该日期的年份部分解释为在 1950 到 1999 年之间。

而如果使用下面的命令，则不会选中任何行，因为 YY 格式将该日期的年份部分解释为当前世纪 (2090)。

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM employees
WHERE TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';
```

```
0 rows selected
```

课程安排

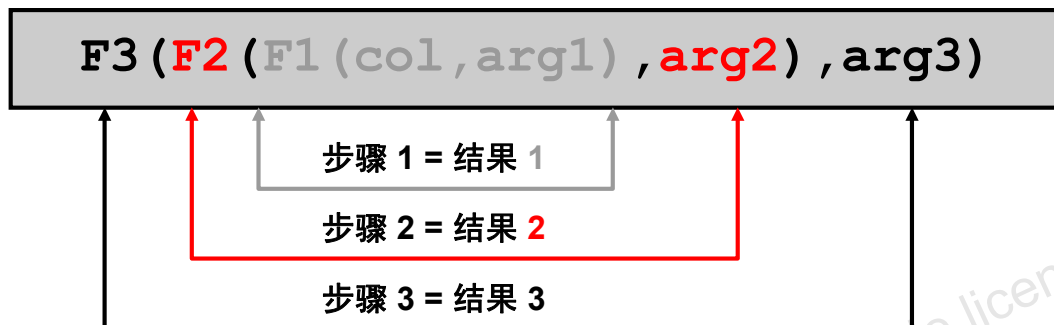
- 隐式和显式数据类型转换
- TO_CHAR、TO_DATE、TO_NUMBER 函数
- **嵌套函数**
- 常规函数：
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件表达式：
 - CASE
 - DECODE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

嵌套函数

- 单行函数可以嵌套到任意层。
- 嵌套函数的计算顺序是从最内层到最外层。



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

嵌套函数

单行函数可以嵌套到任意深度。嵌套函数的计算顺序是从最内层到最外层。下面的一些示例显示这类函数的灵活性。

嵌套函数：示例 1

```
SELECT last_name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US



版权所有 © 2010, Oracle。保留所有权利。

嵌套函数（续）

幻灯片中的示例显示部门 60 中的雇员的姓氏。该 SQL 语句的计算涉及以下三个步骤：

- 1. 内部函数检索姓氏的前 8 个字符。
Result1 = SUBSTR (LAST_NAME, 1, 8)
- 2. 外层函数将结果与 _US 连接在一起。
Result2 = CONCAT (Result1, '_US')
- 3. 最外层函数将结果转换为大写。

因为没有给出列别名，所以整个表达式就成为列标题。

示例：

显示距聘用日期六个月后的下一个星期五的日期。得到的日期应为 “Friday, August 13th, 1999”。按聘用日期对结果排列。

```
SELECT      TO_CHAR(NEXT_DAY(ADD_MONTHS  
                        (hire_date, 6), 'FRIDAY'),  
                        'fmDay, Month ddth, YYYY')  
            "Next 6 Month Review"  
FROM        employees  
ORDER BY    hire_date;
```

嵌套函数：示例 2

```
SELECT      TO_CHAR(ROUND((salary/7), 2), '99G999D99',
              'NLS_NUMERIC_CHARACTERS = ','.')
            "Formatted Salary"
FROM employees;
```

	Formatted Salary
1	628,57
2	1.857,14
3	857,14
4	1.714,29
5	1.185,71
6	3.428,57

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

嵌套函数（续）

幻灯片中的示例显示雇员薪金除以 7 并舍入到 2 位小数的结果。然后，该结果按照丹麦表示法显示。即逗号用作小数点，句点用作千位分隔符。

首先，执行内部 ROUND 函数将薪金除以 7 之后的值舍入到 2 位小数。然后，使用 TO_CHAR 函数设置 ROUND 函数结果的格式。

注：在 TO_CHAR 函数参数中指定的 D 和 G 是数字格式元素。D 在指定的位置返回小数点字符。G 用作组分隔符。

课程安排

- 隐式和显式数据类型转换
- TO_CHAR、TO_DATE、TO_NUMBER 函数
- 嵌套函数
- 常规函数：
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件表达式：
 - CASE
 - DECODE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

常规函数

下列函数可用于任何数据类型，且适合使用空值的场合：

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

常规函数

下列函数可用于任何数据类型，且适合表达式列表中使用空值的场合。

函数	说明
NVL	将空值转换为实际值
NVL2	如果 expr1 为非空值，则 NVL2 返回 expr2。如果 expr1 为空值，则 NVL2 返回 expr3。参数 expr1 可以是任意数据类型
NULLIF	将两个表达式进行比较，如果它们相等，则返回空值；如果不相等，则返回第一个表达式
COALESCE	返回表达式列表中的第一个非空表达式

注：有关数百个可用函数的详细信息，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》中“Functions”一节。

NVL 函数

将空值转换为实际值：

- 可以使用的数据类型为日期、字符和数字。
- 数据类型必须匹配：
 - NVL(commission_pct, 0)
 - NVL(hire_date, '01-JAN-97')
 - NVL(job_id, 'No Job Yet')

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

NVL 函数

要将空值转换为实际值，请使用 NVL 函数。

语法

NVL (expr1, expr2)

在该语法中：

- expr1 是可能包含空值的源值或表达式
- expr2 是用于转换空值的目标值

可以使用 NVL 函数转换任意数据类型，但是返回值始终与 expr1 具有相同的数据类型。

各种数据类型的 NVL 转换

数据类型	转换示例
NUMBER	NVL(number_column, 9)
DATE	NVL(date_column, '01-JAN-95')
CHAR or VARCHAR2	NVL(character_column, 'Unavailable')

使用 NVL 函数

```
SELECT last name, salary, NVL(commission_pct, 0),
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000
9	Hunold	9000	0	108000
10	Ernst	6000	0	72000

...

1

2

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 NVL 函数

要计算所有雇员的年度报酬，需要将月薪乘以 12，然后再加上佣金百分比：

```
SELECT last_name, salary, commission_pct,
       (salary*12) + (salary*12*commission_pct) AN_SAL
FROM employees;
```

	LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
1	Whalen	4400	(null)	(null)
...				
16	Vargas	2500	(null)	(null)
17	Zlotkey	10500	0.2	151200
18	Abel	11000	0.3	171600
19	Taylor	8600	0.2	123840
20	Grant	7000	0.15	96600

请注意，仅为那些领取佣金的雇员计算了年度报酬。如果表达式中的任何列值为空，则结果也为空。要计算所有雇员的年度报酬，必须将空值转换为数字，然后才能应用算术运算符。在幻灯片中的示例中，NVL 函数用于将空值转换为零。

使用 NVL2 函数

```
SELECT last name, salary, commission_pct
      NVL2 (commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 NVL2 函数

NVL2 函数首先会检查第一个表达式。如果第一个表达式不为空，则 NVL2 函数会返回第二个表达式。如果第一个表达式为空，则返回第三个表达式。

语法

```
NVL2(expr1, expr2, expr3)
```

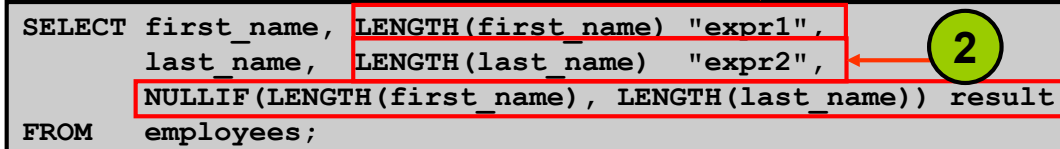
在该语法中：

- *expr1* 是可能包含空值的源值或表达式
- *expr2* 是 *expr1* 不为空时返回的值
- *expr3* 是 *expr1* 为空时返回的值

在幻灯片所示的示例中，检查的是 COMMISSION_PCT 列。如果检测到值，则返回文本值 SAL+COMM。如果 COMMISSION_PCT 列包含空值，则返回文本值 SAL。

注：参数 *expr1* 可以是任意数据类型。参数 *expr2* 和 *expr3* 可以是除 LONG 之外的任意数据类型。

使用 NULLIF 函数



```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name, LENGTH(last_name) "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM employees;
```

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

...

1

2

3

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 NULLIF 函数

NULLIF 函数用于对两个表达式进行比较。

语法

```
NULLIF (expr1, expr2)
```

在该语法中：

- NULLIF 用于对 *expr1* 和 *expr2* 进行比较。如果它们相等，则函数返回空值。如果不相等，则函数返回 *expr1*。但不能为 *expr1* 指定文字值 NULL。

在幻灯片所示的示例中，对 EMPLOYEES 表中名字的长度和 EMPLOYEES 表中姓氏的长度进行了比较。如果姓氏和名字的长度相等，则显示空值。如果姓氏和名字的长度不相等，则显示名字的长度。

注：NULLIF 函数在逻辑上等效于下面的 CASE 表达式。CASE 表达式将在后续页进行介绍：

```
CASE WHEN expr1 = expr2 THEN NULL ELSE expr1 END
```

使用 COALESCE 函数

- COALESCE 函数优于 NVL 函数之处在于 COALESCE 函数可以接受多个备选值。
- 如果第一个表达式不为空，则 COALESCE 函数返回该表达式；否则，它将对其余的表达式执行 COALESCE 运算。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 COALESCE 函数

COALESCE 函数用于返回列表中的第一个非空表达式。

语法

```
COALESCE (expr1, expr2, ... exprn)
```

在该语法中：

- *expr1* 返回此表达式（如果它不为空）
- *expr2* 返回此表达式（如果第一个表达式为空，而此表达式不为空）
- *exprn* 返回此表达式（如果前面的表达式都为空）

请注意，所有表达式都必须具有相同的数据类型。

使用 COALESCE 函数

```
SELECT last name, employee id,
COALESCE(TO_CHAR(commission_pct), TO_CHAR(manager_id),
'No commission and no manager')
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMMISSION_PCT), TO_CHAR(MANAGER_ID), 'No commission and no manager')
1	Whalen	200	101
2	Hartstein	201	100
3	Fay	202	201
4	Higgins	205	101
5	Gietz	206	205
6	King	100	No commission and no manager

...

17	Zlotkey	149	.2
18	Abel	174	.3
19	Taylor	176	.2
20	Grant	178	.15

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 COALESCE 函数（续）

在幻灯片所示的示例中，如果 `manager_id` 值不为空，则会显示出来。如果 `manager_id` 值为空，则会显示 `commission_pct`。如果 `manager_id` 和 `commission_pct` 值都为空，则会显示 “No commission and no manager”。请注意，已应用 `TO_CHAR` 函数，所以所有表达式都具有相同的数据类型。

示例：

对于不领取佣金的雇员，您的组织想要将其薪金增加 \$2,000；对于领取佣金的雇员，在查询中应计算新的薪金，该薪金等于现有薪金加上佣金额。

```
SELECT last_name, salary, commission_pct,
COALESCE((salary+(commission_pct*salary)), salary+2000, salary)
"New Salary"
FROM employees;
```

使用 COALESCE 函数（续）

注：请检查输出。对于不领取佣金的雇员，New Salary 列显示增加了 \$2,000 后的薪金；对于领取佣金的雇员，New Salary 列显示计算的佣金额加上薪金。

	LAST_NAME	SALARY	COMMISSION_PCT	New Salary
1	Whalen	4400	(null)	6400
2	Hartstein	13000	(null)	15000
3	Fay	6000	(null)	8000
4	Higgins	12000	(null)	14000
5	Gietz	8300	(null)	10300
6	King	24000	(null)	26000

...

17	Zlotkey	10500	0.2	12600
18	Abel	11000	0.3	14300
19	Taylor	8600	0.2	10320
20	Grant	7000	0.15	8050

课程安排

- 隐式和显式数据类型转换
- TO_CHAR、TO_DATE、TO_NUMBER 函数
- 嵌套函数
- 常规函数：
 - NVL
 - NVL2
 - NULLIF
 - COALESCE
- 条件表达式：
 - CASE
 - DECODE

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

条件表达式

- 使您可以在 SQL 语句中使用 IF-THEN-ELSE 逻辑。
- 使用下面两种方法：
 - CASE 表达式
 - DECODE 函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

条件表达式

在 SQL 语句中，用于执行条件处理（IF-THEN-ELSE 逻辑）的两种方法是 CASE 表达式和 DECODE 函数。

注：CASE 表达式符合 ANSI SQL。而 DECODE 函数是 Oracle 专用的语法。

CASE 表达式

具有与 IF-THEN-ELSE 语句相同的功效，可简化条件查询：

```
CASE expr WHEN comparison_expr1 THEN return_expr1
      [WHEN comparison_expr2 THEN return_expr2
      WHEN comparison_exprn THEN return_exprn
      ELSE else_expr]
END
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

CASE 表达式

CASE 表达式使您可以在 SQL 语句中使用 IF-THEN-ELSE 逻辑，而无需调用任何过程。

在简单的 CASE 表达式中，Oracle Server 会搜索其 *expr* 等于 *comparison_expr* 的第一对 WHEN ... THEN，然后返回 *return_expr*。如果没有满足此条件的 WHEN ... THEN 对，并且存在一个 ELSE 子句，则 Oracle Server 将返回 *else_expr*。否则，Oracle Server 会返回空值。您不能为所有的 *return_expr* 和 *else_expr* 指定文字值 NULL。

表达式 *expr* 和 *comparison_expr* 必须具有相同的数据类型，可以是 CHAR、VARCHAR2、NCHAR 或 NVARCHAR2。所有返回值 (*return_expr*) 必须具有相同的数据类型。

使用 CASE 表达式

具有与 IF-THEN-ELSE 语句相同的功效，可简化条件查询：

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG' THEN 1.10*salary
                   WHEN 'ST_CLERK' THEN 1.15*salary
                   WHEN 'SA_REP' THEN 1.20*salary
                   ELSE salary END "REVISED_SALARY"
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
1	Whalen	AD_ASST	4400	4400
...				
9	Hunold	IT_PROG	9000	9900
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
14	Davies	ST_CLERK	3100	3565
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

使用 CASE 表达式

幻灯片中的 SQL 语句用于解码 JOB_ID 的值。如果 JOB_ID 为 IT_PROG，则薪金增加 10%；如果 JOB_ID 为 ST_CLERK，则薪金增加 15%；如果 JOB_ID 为 SA_REP，则薪金增加 20%。对于所有其它职务角色，不增加薪金。

可以使用 DECODE 函数编写同样的语句。

以下代码是一个搜索 CASE 表达式示例。在搜索 CASE 表达式中，搜索从左到右进行，直到找到列出的条件，然后它会返回相应的返回表达式。如果找不到满足的条件，但存在一个 ELSE 子句，则返回 ELSE 子句中的返回表达式，其它情况则返回 NULL。

```
SELECT last_name, salary,
       (CASE WHEN salary < 5000 THEN 'Low'
             WHEN salary < 10000 THEN 'Medium'
             WHEN salary < 20000 THEN 'Good'
             ELSE 'Excellent'
          END) qualified_salary
FROM   employees;
```

DECODE 函数

具有与 CASE 表达式或 IF-THEN-ELSE 语句相同的功效，
可简化条件查询：

```
DECODE(col|expression, search1, result1  
      [, search2, result2, ..., ]  
      [, default])
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

DECODE 函数

DECODE 函数用于对表达式进行解码，其方式与在各种语言中使用的 IF-THEN-ELSE 逻辑类似。DECODE 函数在将表达式与每个搜索值进行比较之后对表达式进行解码。如果表达式与搜索值相同，则返回结果。

如果省略了默认值，并且搜索值不与任何结果值相匹配，则会返回一个空值。

使用 DECODE 函数

```
SELECT last name, job id, salary,
       DECODE(job_id, 'IT_PROG', 1.10*salary,
               'ST_CLERK', 1.15*salary,
               'SA_REP', 1.20*salary,
               salary)
       REVISED_SALARY
FROM   employees;
```

	LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...				
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 DECODE 函数

幻灯片中的 SQL 语句用于测试 JOB_ID 的值。如果 JOB_ID 为 IT_PROG，则薪金增加 10%；如果 JOB_ID 为 ST_CLERK，则薪金增加 15%；如果 JOB_ID 为 SA_REP，则薪金增加 20%。对于所有其它职务角色，不增加薪金。

如果使用伪代码，则可将上述语句表示为以下 IF-THEN-ELSE 语句：

```
IF job_id = 'IT_PROG'      THEN salary = salary*1.10
IF job_id = 'ST_CLERK'    THEN salary = salary*1.15
IF job_id = 'SA_REP'      THEN salary = salary*1.20
ELSE salary = salary
```

使用 DECODE 函数

显示部门 80 中每位雇员的适用税率：

```
SELECT last name, salary,
       DECODE (TRUNC(salary/2000, 0),
               0, 0.00,
               1, 0.09,
               2, 0.20,
               3, 0.30,
               4, 0.40,
               5, 0.42,
               6, 0.44,
               0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```



版权所有 © 2010, Oracle。保留所有权利。

使用 DECODE 函数（续）

此幻灯片显示使用 DECODE 函数的另一个示例。在此示例中，根据月薪确定部门 80 中每位雇员的税率。税率如下所示：

月薪范围	税率
\$0.00-1,999.99	00%
\$2,000.00-3,999.99	09%
\$4,000.00-5,999.99	20%
\$6,000.00-7,999.99	30%
\$8,000.00-9,999.99	40%
\$10,000.00-11,999.99	42%
\$12,200.00-13,999.99	44%
\$14,000.00 或更高	45%

	LAST_NAME	SALARY	TAX_RATE
1	Zlotkey	10500	0.42
2	Abel	11000	0.42
3	Taylor	8600	0.4

小测验

TO_NUMBER 函数用于按照可选格式样式指定的格式将字符串或日期值转换为数字。

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：

- 使用函数更改显示的日期格式
- 使用函数转换列数据类型
- 使用 NVL 函数
- 在 SELECT 语句中使用 IF-THEN-ELSE 逻辑和其它条件表达式

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

小结

请牢记以下内容：

- 转换函数可以转换字符、日期和数字值：TO_CHAR、TO_DATE、TO_NUMBER。
- 还有几个适合处理空值的函数，其中包括 NVL、NVL2、NULLIF 和 COALESCE。
- 通过使用 CASE 表达式或 DECODE 函数，可以在 SQL 语句中应用 IF-THEN-ELSE 逻辑。

练习 4：概览

本练习包含以下主题：

- 创建使用 TO_CHAR、TO_DATE 和其它 DATE 函数的查询
- 创建使用 DECODE 和 CASE 等条件表达式的查询

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 4：概览

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。

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.

5 使用组函数报告聚集数据

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：

- 确定可用的组函数
- 描述组函数的使用方法
- 通过使用 GROUP BY 子句对数据进行分组
- 通过使用 HAVING 子句包括或排除分组的行

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

本课将进一步介绍函数。重点介绍如何获取行组的汇总信息（如平均值）。还讨论如何将表中的行分组为较小的集合，以及如何指定行组的搜索标准。

课程安排

- 组函数：
 - 类型和语法
 - 使用 AVG、SUM、MIN、MAX、COUNT
 - 在组函数中使用 DISTINCT 关键字
 - 组函数中的 NULL 值
- 对行进行分组：
 - GROUP BY 子句
 - HAVING 子句
- 嵌套组函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

何谓组函数

组函数会对行集进行计算，为每个组提供一个结果。

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	110	12000
5	110	8300
6	90	24000
7	90	17000
8	90	17000
9	60	9000
10	60	6000
...		
18	80	11000
19	80	8600
20	(null)	7000

**EMPLOYEES 表中的
最高薪金**

	MAX(SALARY)
	24000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

何谓组函数

与单行函数不同，组函数用于对行集进行计算，从而为每个组提供一个结果。这些集合可以包含整个表，也可以包含表分割成的组。

组函数的类型

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

组函数的类型

每个函数都接受一个参数。下表列出了在语法中可使用的选项：

函数	说明
AVG ([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> 的平均值，忽略空值
COUNT ({ * [DISTINCT <u>ALL</u>] <i>expr</i> })	行数，其中 <i>expr</i> 的求值为非空值（使用 * 统计所有选定行的行数，包括重复行和有空值的行）
MAX ([DISTINCT <u>ALL</u>] <i>expr</i>)	<i>expr</i> 的最大值，忽略空值
MIN ([DISTINCT <u>ALL</u>] <i>expr</i>)	<i>expr</i> 的最小值，忽略空值
STDDEV ([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> 的标准差，忽略空值
SUM ([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> 的总计值，忽略空值
VARIANCE ([DISTINCT <u>ALL</u>] <i>n</i>)	<i>n</i> 的方差，忽略空值

组函数：语法

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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

组函数：语法

组函数应放在 SELECT 关键字之后。可以使用逗号分隔多个组函数。

使用组函数的准则：

- DISTINCT 使函数仅考虑非重复值；ALL 使函数考虑每个值（包括重复值）。默认值为 ALL，因此无需指定。
- 使用 expr 参数的函数的数据类型可以是 CHAR、VARCHAR2、NUMBER 或 DATE。
- 所有组函数都忽略空值。要用一个值替代空值，请使用 NVL、NVL2、COALESCE、CASE 或 DECODE 函数。

使用 AVG 和 SUM 函数

您可以对数字数据使用 AVG 和 SUM 函数。

```
SELECT AVG(salary), MAX(salary),
       MIN(salary), SUM(salary)
FROM   employees
WHERE  job_id LIKE '%REP%';
```

	AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
1	8150	11000	6000	32600

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 AVG 和 SUM 函数

可以对能够存储数字数据的列使用 AVG、SUM、MIN 和 MAX 函数。幻灯片中的示例显示所有销售代表的月薪平均值、最高值、最低值与总和。

使用 MIN 和 MAX 函数

您可以对数字、字符和日期数据类型使用 MIN 和 MAX 函数。

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

	MIN(HIRE_DATE)	MAX(HIRE_DATE)
1	17-JUN-87	29-JAN-00

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 MIN 和 MAX 函数

可以对数字、字符和日期数据类型使用 MAX 和 MIN 函数。幻灯片中的示例显示任职时间最短和最长的雇员。

下面的示例显示在包含所有雇员的列表中，按字母顺序排列姓氏时位于首位及位于末位的雇员姓氏：

```
SELECT MIN(last_name), MAX(last_name)
FROM   employees;
```

	MIN(LAST_NAME)	MAX(LAST_NAME)
1	Abel	Zlotkey

注：AVG、SUM、VARIANCE 和 STDDEV 函数仅可用于处理数字数据类型。MAX 和 MIN 函数不能用于处理 LOB 或 LONG 数据类型。

使用 COUNT 函数

COUNT (*) 将返回表中的行数:

1

```
SELECT COUNT (*)
FROM   employees
WHERE  department_id = 50;
```

	COUNT(*)
1	5

COUNT(expr) 将返回 expr 为非空值的行的数量:

2

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

	COUNT(COMMISSION_PCT)
1	3

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 COUNT 函数

COUNT 函数有以下三种格式:

- COUNT (*)
- COUNT(expr)
- COUNT(DISTINCT expr)

COUNT (*) 用于返回表中符合 SELECT 语句标准的行数, 包括重复行和在任何列中含有空值的行。如果 SELECT 语句中包含 WHERE 子句, 则 COUNT (*) 会返回符合 WHERE 子句中条件的行数。

相反, COUNT(expr) 返回由 expr 标识的列中非空值的数量。

COUNT(DISTINCT expr) 返回由 expr 标识的列中不同非空值的数量。

示例:

1. 幻灯片中的示例显示部门 50 中雇员的数量。
2. 幻灯片中的示例显示部门 80 中可以获得佣金的雇员的数量。

使用 DISTINCT 关键字

- COUNT(DISTINCT expr) 将返回 *expr* 的不同非空值的数量。
- 要显示 EMPLOYEES 表中不同部门值的数量，可使用：

```
SELECT COUNT(DISTINCT department_id)
FROM employees;
```

COUNT(DISTINCT DEPARTMENT_ID)	
1	7

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 DISTINCT 关键字

使用 DISTINCT 关键字可以避免对某一列中的任何重复值进行计数。

幻灯片中的示例显示 EMPLOYEES 表中不同部门值的数量。

组函数和空值

组函数将忽略列中的空值：

1

```
SELECT AVG(commission_pct)
FROM employees;
```

AVG(COMMISSION_PCT)	
1	0.2125

NVL 函数会强制组函数包括空值：

2

```
SELECT AVG(NVL(commission_pct, 0))
FROM employees;
```

AVG(NVL(COMMISSION_PCT,0))	
1	0.0425

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

组函数和空值

所有组函数都忽略列中的空值。

但是，NVL 函数会强制组函数包括空值。

示例：

1. 仅根据在表的 COMMISSION_PCT 列中存储了有效值的行计算平均值。平均值的计算方法是用付给所有雇员的佣金总和除以获得佣金的雇员数 (4)。
2. 根据表中的所有行计算平均值，不考虑 COMMISSION_PCT 列中是否存储空值。平均值的计算方法是用付给所有雇员的佣金总和除以公司中的雇员总数 (20)。

课程安排

- 组函数：
 - 类型和语法
 - 使用 AVG、SUM、MIN、MAX、COUNT
 - 在组函数中使用 DISTINCT 关键字
 - 组函数中的 NULL 值
- 对行进行分组：
 - GROUP BY 子句
 - HAVING 子句
- 嵌套组函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

创建数据组

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

4400

9500

3500

6400

10033

**EMPLOYEES 表中
每个部门的平均薪金**

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.333333333333...
4	110	10150
5	50	3500
6	80	10033.333333333333...
7	10	4400
8	60	6400

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

创建数据组

在介绍该知识点之前，所有组函数都将表当作一个大型的信息组。但是，有时您需要将此信息表分成几个较小的组。可以通过使用 GROUP BY 子句完成此任务。

创建数据组：GROUP BY 子句的语法

可以通过使用 GROUP BY 子句将表中的行分成较小的组。

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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

创建数据组：GROUP BY 子句的语法

可以使用 GROUP BY 子句将表中的行分成组。然后，可以使用组函数返回每个组的汇总信息。

在该语法中：

group_by_expression 指定某些列，这些列的值确定对行进行分组的基准

准则

- 除非在 GROUP BY 子句中指定了单个列，否则即使在 SELECT 子句中包括组函数，也不能选择单个结果。如果未在 GROUP BY 子句中包括列的列表，则会收到一条错误消息。
- 通过使用 WHERE 子句，您可以在将行分成多个组之前先排除某些行。
- 必须将列包括在 GROUP BY 子句中。
- 不能在 GROUP BY 子句中使用列别名。

使用 GROUP BY 子句

SELECT 列表中未出现在组函数中的所有列都必须包含在 GROUP BY 子句中。

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

	DEPARTMENT_ID	AVG(SALARY)
1	(null)	7000
2	20	9500
3	90	19333.33333333333...
4	110	10150
5	50	3500
6	80	10033.33333333333...
7	10	4400
8	60	6400

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 GROUP BY 子句

使用 GROUP BY 子句时，应确保将 SELECT 列表中未出现在组函数中的所有列都包含在 GROUP BY 子句中。幻灯片中的示例显示每个部门的部门编号和平均薪金。下面介绍含有 GROUP BY 子句的 SELECT 语句是如何进行求值的：

- SELECT 子句指定要检索的列，如下所示：
 - EMPLOYEES 表中的部门编号列
 - GROUP BY 子句指定的组中所有薪金的平均值
- FROM 子句指定数据库必须访问的表：EMPLOYEES 表。
- WHERE 子句指定要检索的行。由于没有 WHERE 子句，默认情况下会检索所有行。
- GROUP BY 子句指定如何对行进行分组。由于是按部门编号对行进行分组，因此应用于薪金列的 AVG 函数会计算每个部门的平均薪金。

注：要按升序或降序对查询结果进行排序，请在查询中包含 ORDER BY 子句。

使用 GROUP BY 子句

GROUP BY 列不一定要出现在 SELECT 列表中。

```
SELECT    AVG(salary)
FROM      employees
GROUP BY  department_id ;
```

	AVG(SALARY)
1	7000
2	9500
3	19333.33333333333333333333...
4	10150
5	3500
6	10033.33333333333333333333...
7	4400
8	6400

ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

使用 GROUP BY 子句 (续)

GROUP BY 列不一定要出现在 SELECT 子句中。例如，幻灯片中的 SELECT 语句显示每个部门的平均薪金，但没有显示相应的部门编号。但是如果没有部门编号，结果看起来毫无意义。

也可以在 ORDER BY 子句中使用组函数:

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id
ORDER BY  AVG(salary);
```

	DEPARTMENT_ID	AVG(SALARY)
1	50	3500
2	10	4400
3	60	6400

■ ■ ■

[illegible]

按多个列进行分组

EMPLOYEES

	DEPARTMENT_ID	JOB_ID	SALARY
1	10	AD_ASST	4400
2	20	MK_MAN	13000
3	20	MK_REP	6000
4	50	ST_CLERK	2500
5	50	ST_CLERK	2600
6	50	ST_CLERK	3100
7	50	ST_CLERK	3500
8	50	ST_MAN	5800
9	60	IT_PROG	9000
10	60	IT_PROG	6000
11	60	IT_PROG	4200
12	80	SA_REP	11000
13	80	SA_REP	8600
14	80	SA_MAN	10500
...			
19	110	AC_MGR	12000
20	(null)	SA_REP	7000

将 EMPLOYEES 表中每种职务的薪金相加，并按部门进行分组。

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	110	AC_ACCOUNT	8300
2	110	AC_MGR	12000
3	10	AD_ASST	4400
4	90	AD_PRES	24000
5	90	AD_VP	34000
6	60	IT_PROG	19200
7	20	MK_MAN	13000
8	20	MK_REP	6000
9	80	SA_MAN	10500
10	80	SA_REP	19600
11	(null)	SA_REP	7000
12	50	ST_CLERK	11700
13	50	ST_MAN	5800

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

按多个列进行分组

有时，您需要查看组内的各个组的结果。此幻灯片显示一个报表，其中显示要付给各个部门中每种职务的薪金总和。

EMPLOYEES 表首先按部门编号进行分组，然后在各个组中又按职务进行分组。例如，将部门 50 中的四个仓储职员分成一个组，并为该组中的所有仓储职员生成一个结果（薪金总和）。

以下 SELECT 语句返回幻灯片中显示的结果：

```
SELECT department_id, job_id, sum(salary)
FROM employees
GROUP BY department_id, job_id
ORDER BY job_id;
```

对多个列使用 GROUP BY 子句

```
SELECT department_id, job_id, SUM(salary)
FROM employees
WHERE department_id > 40
GROUP BY department_id, job_id
ORDER BY department_id;
```

	DEPARTMENT_ID	JOB_ID	SUM(SALARY)
1	50	ST_CLERK	11700
2	50	ST_MAN	5800
3	60	IT_PROG	19200
4	80	SA_MAN	10500
5	80	SA_REP	19600
6	90	AD_PRES	24000
7	90	AD_VP	34000
8	110	AC_ACCOUNT	8300
9	110	AC_MGR	12000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

对多个列使用 GROUP BY 子句

通过列出多个 GROUP BY 列，可以返回组和子组的汇总结果。GROUP BY 子句对行进行分组，但不保证结果集的顺序。要对组进行排序，请使用 ORDER BY 子句。

在幻灯片中的示例中，包含 GROUP BY 子句的 SELECT 语句按如下方式进行求值：

- SELECT 子句指定要检索的列：
 - EMPLOYEES 表中的部门 ID
 - EMPLOYEES 表中的职务 ID
 - GROUP BY 子句指定的组中所有薪金的总和
- FROM 子句指定数据库必须访问的表：EMPLOYEES 表。
- WHERE 子句将结果集限定为部门 ID 大于 40 的行。
- GROUP BY 子句指定应如何对结果行进行分组：
 - 首先，按部门 ID 对行进行分组
 - 其次，在部门 ID 组中按职务 ID 对行进行分组
- ORDER BY 子句按部门 ID 对结果进行排序。

注：SUM 函数将应用于每个部门 ID 组的结果集中所有职务 ID 的薪金列。另外，请注意，不返回 SA_REP 行。此行的部门 ID 为 NULL，因此不满足 WHERE 条件。

使用组函数的非法查询

SELECT 列表中不在聚集函数中的任何列或表达式都必须出现在 GROUP BY 子句中：

```
SELECT department_id, COUNT(last_name)
FROM employees;
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

必须添加 GROUP BY 子句，才能对每个 department_id 对应的姓氏进行计数。

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

ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

要么在 GROUP BY 中添加 job_id，要么从 SELECT 列表中删除 job_id 列。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用组函数的非法查询

只要在一个 SELECT 语句中混合使用单个项 (DEPARTMENT_ID) 和组函数 (COUNT)，就必须包括一个指定这些单个项（本例中为 DEPARTMENT_ID）的 GROUP BY 子句。如果缺少 GROUP BY 子句，则会出现错误消息 “not a single-group group function（不是一个组的组函数）”，而且显示一个指向错误列的星号 (*)。可通过添加 GROUP BY 子句更正幻灯片中的第一个示例中的错误：

```
SELECT department_id, count(last_name)
FROM employees
GROUP BY department_id;
```

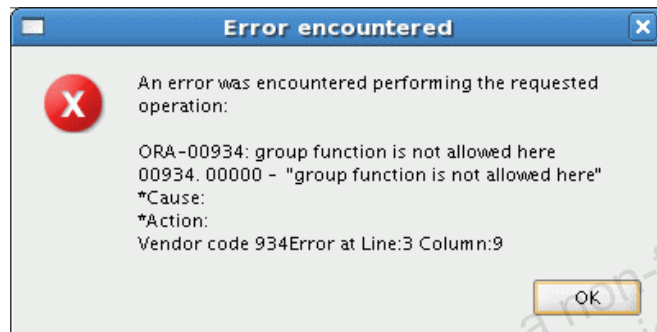
SELECT 列表中不在聚集函数中的任何列或表达式都必须出现在 GROUP BY 子句中。在幻灯片中的第二个示例中，job_id 既不在 GROUP BY 子句中也不在组函数中，因此将出现 “not a GROUP BY expression（不是 GROUP BY 表达式）” 错误。可通过在 GROUP BY 子句中添加 job_id 更正幻灯片中的第二个示例中的错误。

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

使用组函数的非法查询

- 不能使用 WHERE 子句限定组。
- 可以使用 HAVING 子句限定组。
- 不能在 WHERE 子句中使用组函数。

```
SELECT    department_id, AVG(salary)
FROM      employees
WHERE     AVG(salary) > 8000
GROUP BY  department_id;
```



不能使用 WHERE
子句限定组

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用组函数的非法查询（续）

不能使用 WHERE 子句限定组。幻灯片示例中的 SELECT 语句产生了一个错误，因为该语句使用 WHERE 子句限定显示平均薪金大于 \$8,000 的那些部门的平均薪金。

但是，通过使用 HAVING 子句限定组，可以更正该示例中的错误：

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id
HAVING    AVG(salary) > 8000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	20	9500
2	90	19333.3333333333...
3	110	10150
4	80	10033.3333333333...

限定组结果

EMPLOYEES

	DEPARTMENT_ID	SALARY
1	10	4400
2	20	13000
3	20	6000
4	50	2500
5	50	2600
6	50	3100
7	50	3500
8	50	5800
9	60	9000
10	60	6000
11	60	4200
12	80	11000
13	80	8600
...		
18	110	8300
19	110	12000
20	(null)	7000

最高薪金大于 \$10,000 的
每个部门的最高薪金

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

限定组结果

与使用 WHERE 子句限定所选行的方式相同，可以使用 HAVING 子句限定组。要在最高薪金大于 \$10,000 的每个部门中查找最高薪金，需要执行以下操作：

1. 通过按部门编号进行分组，查找每个部门的最高薪金。
2. 将组限定为最高薪金大于 \$10,000 的部门。

使用 HAVING 子句限定组结果

使用 HAVING 子句时，Oracle Server 将按以下方式对组进行限定：

1. 对行进行分组。
2. 应用组函数。
3. 显示符合 HAVING 子句的组。

```
SELECT    column, group_function
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[HAVING   group_condition]
[ORDER BY column];
```

ORACLE

版权所有 © 2010, Oracle. 保留所有权利。

使用 HAVING 子句限定组结果

可使用 HAVING 子句指定要显示的组，该子句基于汇总信息进一步限定组。

在上述语法中，*group_condition* 用于限定满足指定条件的组的返回行组。

使用 HAVING 子句时，Oracle Server 会执行以下步骤：

1. 对行进行分组。
2. 对组应用组函数。
3. 显示符合 HAVING 子句中的标准的组。

HAVING 子句可放在 GROUP BY 子句之前，但建议将 GROUP BY 子句放在前面，因为这样更符合逻辑。应先形成组并计算组函数，然后再对 SELECT 列表中的组应用 HAVING 子句。

注：WHERE 子句限定行，而 HAVING 子句限定组。

使用 HAVING 子句

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY  department_id
HAVING    MAX(salary)>10000 ;
```

	DEPARTMENT_ID	MAX(SALARY)
1	20	13000
2	90	24000
3	110	12000
4	80	11000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 HAVING 子句

幻灯片中的示例显示最高薪金大于 \$10,000 的部门的部门编号和最高薪金。

可以在 SELECT 列表中使用 GROUP BY 子句，而不使用组函数。如果根据组函数的结果来限定行，则必须采用 GROUP BY 子句和 HAVING 子句。

下面的示例显示最高薪金大于 \$10,000 的部门的部门编号和平均薪金：

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY  department_id
HAVING    max(salary)>10000;
```

	DEPARTMENT_ID	AVG(SALARY)
1	20	9500
2	90	19333.333333333...
3	110	10150
4	80	10033.333333333...

使用 HAVING 子句

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY  job_id
HAVING    SUM(salary) > 13000
ORDER BY  SUM(salary);
```

	JOB_ID	PAYROLL
1	IT_PROG	19200
2	AD_PRES	24000
3	AD_VP	34000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 HAVING 子句（续）

幻灯片中的示例显示工资合计超过 \$13,000 的每个职务的职务 ID 和月薪总额。该示例将销售代表排除在外，而且按月薪总额对列表进行排序。

课程安排

- 组函数：
 - 类型和语法
 - 使用 AVG、SUM、MIN、MAX、COUNT
 - 在组函数中使用 DISTINCT 关键字
 - 组函数中的 NULL 值
- 对行进行分组：
 - GROUP BY 子句
 - HAVING 子句
- 嵌套组函数

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

下列语句显示最高平均薪金:

[illegible]

版权所有 © 2010, Oracle。保留所有权利。

组函数可以嵌套两层。幻灯片中的示例计算每个 department_id 对应的平均薪金，然后显示最高平均薪金。

请注意，嵌套组函数时，必须使用 GROUP BY 子句。

小测验

找出组函数和 GROUP BY 子句的准则。

1. 不能在 GROUP BY 子句中使用列别名。
2. GROUP BY 列必须出现在 SELECT 子句中。
3. 通过使用 WHERE 子句，您可以在将行分成多个组之前先排除某些行。
4. GROUP BY 子句对行进行分组并确保结果集的顺序。
5. 即使在 SELECT 子句中包括一个组函数，也不能选择单个结果。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、3

小结

在本课中，您应该已经学会：

- 使用组函数 COUNT、MAX、MIN、SUM 和 AVG
- 编写包含 GROUP BY 子句的查询
- 编写包含 HAVING 子句的查询

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[HAVING     group_condition]
[ORDER BY  column];
```

ORACLE®

版权所有 © 2010, Oracle。保留所有权利。

小结

SQL 中提供了多个组函数，例如 AVG、COUNT、MAX、MIN、SUM、STDDEV 和 VARIANCE。

可以使用 GROUP BY 子句创建子组。此外，可以使用 HAVING 子句限定组。

在一个语句中，应将 HAVING 和 GROUP BY 子句放在 WHERE 子句之后。WHERE 子句后面的 GROUP BY 和 HAVING 子句的顺序并不重要。请将 ORDER BY 子句放在末尾。

Oracle Server 按下面的顺序对子句进行求值：

1. 如果语句包含 WHERE 子句，则服务器会确定候选行。
2. 服务器识别 GROUP BY 子句中指定的组。
3. HAVING 子句进一步限定结果组，排除不符合 HAVING 子句中组标准的组。

注：有关完整的组函数列表，请参阅《Oracle Database SQL Language Reference 11g, Release 1 (11.1)》。

练习 5：概览

本练习包含以下主题：

- 编写包含组函数的查询
- 按行进行分组以获得多个结果
- 使用 HAVING 子句对组进行限定

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 5：概览

在本练习中，您将学习使用组函数和选择数据组。

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.

使用联接显示多个表中的数据

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：

- 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 OUTER 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。

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- 非等值联接
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

获取多个表中的数据

EMPLOYEES				DEPARTMENTS			
	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID		DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	200	Whalen	10	1	10	Administration	1700
2	201	Hartstein	20	2	20	Marketing	1800
3	202	Fay	20	3	50	Shipping	1500
...				4	60	IT	1400
18	174	Abel	80	5	80	Sales	2500
19	176	Taylor	80	6	90	Executive	1700
20	178	Grant	(null)	7	110	Accounting	1700
				8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

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。

联接类型

符合 SQL:1999 标准的联接包括：

- 自然联接：
 - NATURAL JOIN 子句
 - USING 子句
 - ON 子句
- OUTER 联接：
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN
- 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

联接类型

要联接各个表，可以使用符合 SQL:1999 标准的联接语法。

附注

- 在 Oracle9i 之前的发行版中，该联接语法与美国国家标准协会 (ANSI) 的标准不同。与以前发行版中的 Oracle 专用联接语法相比，符合 SQL:1999 的联接语法没有任何性能优势。有关专用联接语法的详细信息，请参阅附录 F：Oracle 联接语法。
- 以下幻灯片讨论符合 SQL:1999 的联接语法。

使用 SQL:1999 语法将表联接起来

使用联接可查询多个表中的数据：

```
SELECT    table1.column, table2.column
FROM      table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)] |
[CROSS JOIN table2];
```

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 SQL:1999 语法将表联接起来

在该语法中：

- table1.column 表示从中检索数据的表和列
- NATURAL JOIN 根据相同的列名联接两个表
- JOIN table2 USING column_name 根据列名执行等值联接
- JOIN table2 ON table1.column_name = table2.column_name 根据 ON 子句中的条件执行等值联接
- LEFT/RIGHT/FULL OUTER 用于执行 OUTER 联接
- CROSS JOIN 用于返回两个表的笛卡尔积

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

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。

ORACLE

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。

但是，使用表名限定列名可能非常耗时，特别是当表名较长时。可改为使用表别名。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。

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- 非等值联接
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

创建自然联接

- NATURAL JOIN 子句以两个表中具有相同名称的所有列为基础。
- 它从两个表中选择在所有相匹配列中具有相同值的那些行。
- 如果名称相同的列具有不同的数据类型，则返回一个错误。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

创建自然联接

可以根据两个表中具有相匹配的数据类型和名称的那些列，对表执行自动联接。使用 NATURAL JOIN 关键字可以完成此操作。

注：只能对两个表中具有相同名称和数据类型的那些列执行联接。如果列的名称相同但数据类型不同，那么 NATURAL JOIN 语法将导致产生一个错误。

使用自然联接检索记录

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations ;
```

	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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用自然联接检索记录

在幻灯片中的示例中，通过 LOCATION_ID 列将 LOCATIONS 表和 DEPARTMENT 表联接起来，LOCATION_ID 列是两个表中具有相同名称的唯一列。如果存在其它通用列，联接也会使用所有这些列。

使用 WHERE 子句的自然联接

可以使用 WHERE 子句对自然联接施加其它限制。以下示例将输出行限制为部门 ID 等于 20 或 50 的那些行：

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations
WHERE  department_id IN (20, 50);
```

使用 USING 子句创建联接

- 如果多个列具有相同的名称，但数据类型不匹配，请使用 USING 子句指定等值联接的列。
- 当有多个列相匹配时，使用 USING 子句可仅与一列相匹配。
- NATURAL JOIN 和 USING 语句是互相排斥的。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 USING 子句创建联接

自然联接使用具有匹配的名称和数据类型的所有列来联接表。可以使用 USING 子句仅指定等值联接应使用的那些列。

联接列名

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

...

外键

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

主键

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

联接列名

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补码。

注：等值联接也称为简单联接或内部联接。

使用 USING 子句检索记录

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

	EMPLOYEE_ID	LAST_NAME	LOCATION_ID	DEPARTMENT_ID
1	200	Whalen	1700	10
2	201	Hartstein	1800	20
3	202	Fay	1800	20
4	144	Vargas	1500	50
5	143	Matos	1500	50
6	142	Davies	1500	50
7	141	Rajs	1500	50
8	124	Mourgos	1500	50

...

18	206	Gietz	1700	110
19	205	Higgins	1700	110

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

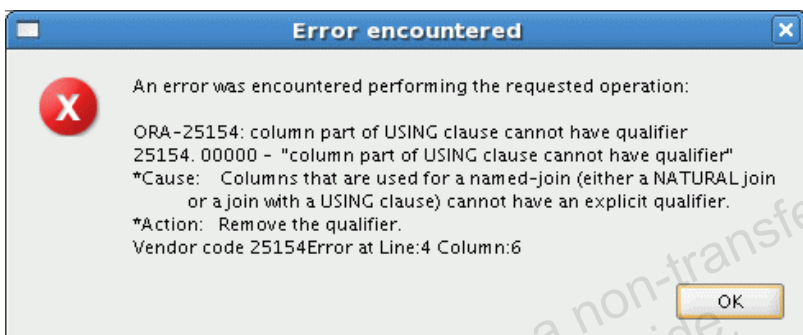
使用 USING 子句检索记录

在幻灯片中的示例中，由于 EMPLOYEES 表和 DEPARTMENTS 表中的 DEPARTMENT_ID 列已联接起来，因此会显示雇员所在部门的 LOCATION_ID。

在 USING 子句中使用表别名

- 不要对 USING 子句中使用的列加以限定。
- 如果在 SQL 语句的另一个位置使用了同一列，则不要对其设置别名。

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d
USING (location_id)
WHERE  d.location_id = 1400;
```



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在 USING 子句中使用表别名

使用 USING 子句进行联接时，不能对 USING 子句自身中使用的列加以限定。此外，如果在 SQL 语句的任何位置使用了该列，则不能对其设置别名。例如，在幻灯片提及的查询中，不能对 WHERE 子句中的 location_id 列设置别名，因为在 USING 子句中已使用该列。

USING 子句中引用的那些列不能在 SQL 语句的任何位置使用限定词（表名或别名）。

例如，下面的语句是有效的：

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  location_id = 1400;
```

这两个表中通用但未在 USING 子句中使用的列必须以表别名为前缀，否则会出现“column ambiguously defined（定义的列含糊不清）”错误。

在下面的语句中，manager_id 既在 employees 表中又在 departments 表中，如果 manager_id 不以表别名为前缀，则会收到“column ambiguously defined（定义的列含糊不清）”错误。

在 USING 子句中使用表别名（续）

下面的语句是有效的：

```
SELECT first_name, d.department_name, d.manager_id
FROM   employees e JOIN departments d USING (department_id)
WHERE  department_id = 50;
```

使用 ON 子句创建联接

- 自然联接的基本联接条件是对具有相同名称的所有列进行等值联接。
- 使用 ON 子句可指定任意条件或指定要联接的列。
- 联接条件独立于其它搜索条件。
- 使用 ON 子句可使代码易于理解。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ON 子句创建联接

使用 ON 子句可指定联接条件。这样，便可以在 WHERE 子句中指定独立于任何搜索条件或过滤条件的联接条件。

使用 ON 子句检索记录

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON      (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 ON 子句检索记录

在此示例中，EMPLOYEES 表和 DEPARTMENTS 表中的 DEPARTMENT_ID 列已使用 ON 子句联接起来。只要 EMPLOYEES 表中的部门 ID 等于 DEPARTMENTS 表中的部门 ID，就返回相应的行。要限定相匹配的 column_names，表别名是必需的。

还可以使用 ON 子句来联接名称不相同的列。幻灯片中的示例所示的联接列 (e.department_id = d.department_id) 两边的圆括号是可选的。因此，即便是 ON e.department_id = d.department_id 效果也一样。

注：当您使用“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 ON 子句创建三向联接

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

	EMPLOYEE_ID	CITY	DEPARTMENT_NAME
1	100	Seattle	Executive
2	101	Seattle	Executive
3	102	Seattle	Executive
4	103	Southlake	IT
5	104	Southlake	IT
6	107	Southlake	IT
7	124	South San Francisco	Shipping
8	141	South San Francisco	Shipping
9	142	South San Francisco	Shipping

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ON 子句创建三向联接

三向联接是指三个表的联接。在符合 SQL:1999 的语法中，从左至右执行联接。这样，要执行的第一个联接是 EMPLOYEES JOIN DEPARTMENTS。第一个联接条件可以引用 EMPLOYEES 和 DEPARTMENTS 中的列，但不能引用 LOCATIONS 中的列。第二个联接条件可以引用全部三个表中的列。

注：幻灯片中的代码示例也可以使用 USING 子句实现：

```
SELECT e.employee_id, l.city, d.department_name
FROM employees e
JOIN departments d
USING (department_id)
JOIN locations l
USING (location_id)
```

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.

对联接应用附加条件

使用 AND 子句或 WHERE 子句可应用附加条件：

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

或

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
WHERE  e.manager_id = 149 ;
```



版权所有 © 2010, Oracle。保留所有权利。

对联接应用附加条件

可以对联接应用附加条件。

所示示例对 EMPLOYEES 和 DEPARTMENTS 表执行联接，另外，还仅显示经理 ID 为 149 的雇员。要对 ON 子句添加附加条件，可以添加 AND 子句。或者，可以使用 WHERE 子句应用附加条件。

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID_1	LOCATION_ID
1	174	Abel	80	80	2500
2	176	Taylor	80	80	2500

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- 非等值联接
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

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。

使用 ON 子句进行自联接

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King

...

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 ON 子句进行自联接

ON 子句还可以用于联接同一表内或者不同表中具有不同名称的列。

所示示例为 EMPLOYEES 表基于 EMPLOYEE_ID 和 MANAGER_ID 列进行自联接。

注：幻灯片中的示例所示的联接列 (e.manager_id = m.employee_id) 两边的圆括号是可选的。因此，即便是 ON e.manager_id = m.employee_id 效果也一样。

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- **非等值联接**
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

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 的值的 LOWEST_SAL 和 HIGHEST_SAL 范围。因此，GRADE_LEVEL 列可用于为每位雇员指定等级。

ORACLE

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 JOIN job_grades j
ON     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：

- JOB_GRADES 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额之内。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，这是考虑到性能原因，而不是因为可能出现的混淆。

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- 非等值联接
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 OUTER 联接返回没有直接匹配的记录

DEPARTMENTS

	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

部门 190 中没有雇员。

没有为雇员 “Grant”
分配部门 ID。

EMPLOYEES 中的等值联接

	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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 OUTER 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。

在幻灯片示例中，针对 EMPLOYEES 和 DEPARTMENTS 表使用简单等值联接来返回右侧的结果。结果集不包含以下内容：

- 部门 ID 190，因为 EMPLOYEES 表中不存在具有该部门 ID 的雇员
- 姓氏为 Grant 的雇员，因为没有为该雇员分配部门 ID

要返回没有任何雇员部门记录，或者返回没有分配部门的雇员，您可以使用 OUTER 联接。

INNER 联接与 OUTER 联接

- 在 SQL:1999 中，如果两个表的联接只返回相匹配的行，则称该联接为 INNER 联接。
- 如果两个表之间的联接不仅返回 INNER 联接的结果，还返回左（或右）表中不匹配的行，则称该联接为左（或右）OUTER 联接。
- 如果两个表之间的联接不仅返回 INNER 联接的结果，还返回左和右联接的结果，则称该联接为完全 OUTER 联接。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

INNER 联接与 OUTER 联接

使用 NATURAL JOIN、USING 或 ON 子句对表进行联接时会生成 INNER 联接。输出中不会显示任何不匹配的行。要返回不匹配的行，可以使用 OUTER 联接。OUTER 联接将返回满足联接条件的所有行，还会返回一个表在另一表中没有满足联接条件的对应行的部分或全部行。

有三种 OUTER 联接类型：

- LEFT OUTER
- RIGHT OUTER
- FULL OUTER

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON     (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。

LEFT OUTER JOIN

此查询将检索 EMPLOYEES 表（它是左表）中的所有行，即使 DEPARTMENTS 表中没有匹配项也是如此。

RIGHT OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (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

...			
18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

RIGHT OUTER JOIN

此查询将检索 DEPARTMENTS 表（它是右表）中的所有行，即使 EMPLOYEES 表中没有匹配项也是如此。

FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

FULL OUTER JOIN

此查询将检索 EMPLOYEES 表中的所有行，即使 DEPARTMENTS 表中没有匹配项也是如此。它还检索 DEPARTMENTS 表中的所有行，即使 EMPLOYEES 表中没有匹配项也是如此。

课程安排

- JOIN 的类型及其语法
- 自然联接：
 - USING 子句
 - ON 子句
- 自联接
- 非等值联接
- OUTER 联接：
 - LEFT OUTER 联接
 - RIGHT OUTER 联接
 - FULL OUTER 联接
- 笛卡尔积
 - 交叉联接

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

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
- 如果要避免生成笛卡尔积，请始终包括有效的联接条件。

ORACLE

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。第一个表中的所有行会被联接到第二个表中的所有行。

笛卡尔积往往会生成大量的行，这种结果几乎没有任何用处。因此，应始终包括有效的联接条件，除非有特定需求需要组合所有表中的所有行。

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 x 8 = 160 行

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10
2	201	20
...		
21	200	10
22	201	20
...		
159	176	80
160	178	(null)

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

生成笛卡尔积

如果某个联接条件被忽略，则会生成笛卡尔积。幻灯片中的示例显示 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。

创建交叉联接

- CROSS JOIN 子句可生成两个表的叉积。
- 这也称为两个表间的笛卡尔积。

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

	1	LAST_NAME	2	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

版权所有 © 2010, Oracle。保留所有权利。

创建交叉联接

幻灯片中的示例生成 EMPLOYEES 表和 DEPARTMENTS 表的笛卡尔积。

CROSS JOIN 方法非常有用，可以应用于许多情况。例如，要按月按办公室返回总人工成本，即使 X 月没有人工成本，您也可以执行所有月的表与办公室表的交叉联接。

如果要创建笛卡尔积，最佳做法是在 SELECT 中显式说明 CROSS JOIN。因此，可以非常清楚表明您要生成笛卡尔积，而不是缺少联接的结果。

小测验

SQL:1999 标准联接语法支持下列类型的联接。Oracle 联接语法支持下列哪些联接类型？

1. 等值联接
2. 非等值联接
3. LEFT OUTER 联接
4. RIGHT OUTER 联接
5. FULL OUTER 联接
6. 自联接
7. 自然联接
8. 笛卡尔积

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、2、3、4、6、8

小结

在本课中，您应该已经学会如何通过以下方式使用联接来显示多个表中的数据：

- 等值联接
- 非等值联接
- OUTER 联接
- 自联接
- 交叉联接
- 自然联接
- FULL（或双边）OUTER 联接

ORACLE

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
- OUTER 联接
- 自联接
- 交叉联接
- 自然联接
- FULL（或双边）OUTER 联接

笛卡尔积

笛卡尔积会导致显示所有组合的行。通过忽略 WHERE 子句或指定 CROSS JOIN 子句可实现此操作。

小结（续）

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。
- 表别名有时是强制性的，目的是为了避免列混淆。

练习 6：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 6：概览

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。

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.

使用子查询来解决查询

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：

- 定义子查询
- 描述子查询可以解决的问题类型
- 列出子查询的类型
- 编写单行和多行子查询

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

在本课中，您将学习有关 SELECT 语句的更多高级功能。您可以在另一个 SQL 语句的 WHERE 子句中编写子查询，从而根据未知条件值获得值。本课还介绍单行子查询和多行子查询。

课程安排

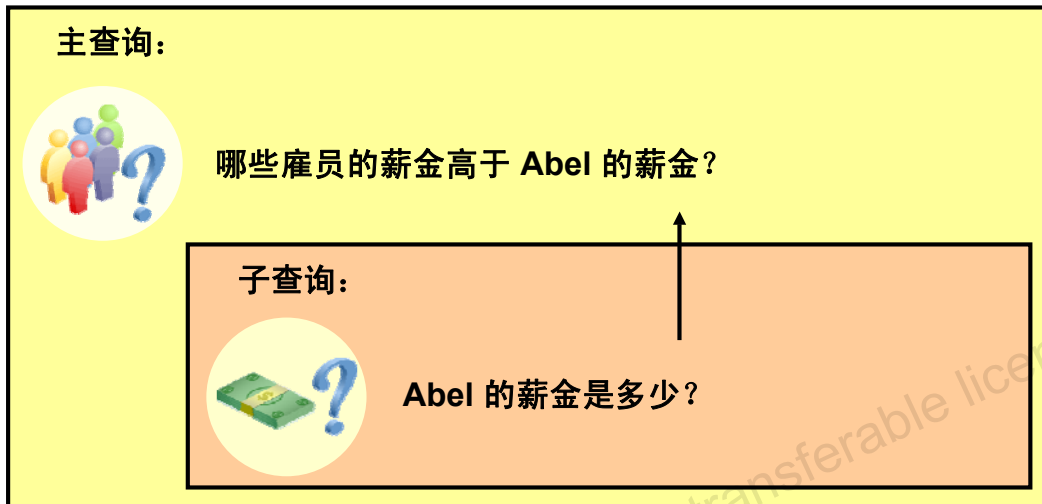
- 子查询：类型、语法和准则
- 单行子查询：
 - 子查询中的组函数
 - 带有子查询的 HAVING 子句
- 多行子查询
 - 使用 ALL 或 ANY 运算符
- 使用 EXISTS 运算符
- 子查询中的空值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用子查询解决问题

谁的薪金高于 Abel 的薪金？



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用子查询解决问题

假设您要编写一个查询来找出谁的薪金高于 Abel 的薪金。

要解决此问题，需要使用两个查询：一个查询用于查找 Abel 的薪金，另一个查询用于查找薪金超过该金额的人员。

通过组合这两个查询，即将一个查询放在另一个查询中，可以解决此问题。

内部查询（即子查询）会返回一个外部查询（即主查询）要使用的值。使用子查询等同于执行两个连续的查询，而且将第一个查询的结果用作第二个查询中的搜索值。

子查询语法

```

SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT      select_list
         FROM         table);

```

- 先执行子查询（内部查询），再执行主查询（外部查询）。
- 主查询会使用子查询的结果。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

子查询语法

子查询是一个 SELECT 语句，它嵌入到另一个 SELECT 语句的子句中。通过使用子查询，可以用简单的语句构建功能强大的语句。当您需要从表中选择行，而选择条件却取决于该表自身中的数据时，子查询非常有用。

可以在许多 SQL 子句中使用子查询，其中包括以下子句：

- WHERE 子句
- HAVING 子句
- FROM 子句

在该语法中：

operator 包括比较条件，例如 >、= 或 IN

注：比较条件分为以下两类：单行运算符（>、=、>=、<、<>、<=）和多行运算符（IN、ANY、ALL、EXISTS）。

子查询通常被称为嵌套 SELECT 语句、子 SELECT 语句或内部 SELECT 语句。通常先执行子查询，然后使用其输出来完善主查询（即外部查询）的查询条件。

使用子查询

```
SELECT last_name, salary
FROM   employees
WHERE  salary > 11000
      (SELECT salary
       FROM   employees
       WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	Hartstein	13000
2	Higgins	12000
3	King	24000
4	Kochhar	17000
5	De Haan	17000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用子查询

在幻灯片中，内部查询确定了雇员 Abel 的薪金。外部查询会采用内部查询的结果并根据此结果显示薪金超过雇员 Abel 的所有雇员。

使用子查询的准则

- 将子查询放在括号中。
- 子查询放在比较条件的右侧可增加可读性。（但是，子查询可出现在比较运算符的任意一侧。）
- 对单行子查询使用单行运算符，对多行子查询使用多行运算符。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用子查询的准则

- 子查询必须放在括号中。
- 子查询放在比较条件的右侧可增加可读性。但是，子查询可出现在比较运算符的任意一侧。
- 在子查询中可以使用两类比较条件：单行运算符和多行运算符。

子查询的类型

- 单行子查询



- 多行子查询



ORACLE

版权所有 © 2010, Oracle。保留所有权利。

子查询的类型

- 单行子查询：从内部 SELECT 语句中仅返回一行的查询
- 多行子查询：从内部 SELECT 语句中返回多行的查询

注：此外，还有多列子查询，此类查询从内部 SELECT 语句中返回多个列。将在《Oracle Database 11g: SQL 基础 II》课程中详细讨论有关日期时间数据类型的内容。

课程安排

- 子查询：类型、语法和准则
- 单行子查询：
 - 子查询中的组函数
 - 带有子查询的 HAVING 子句
- 多行子查询
 - 使用 ALL 或 ANY 运算符
- 使用 EXISTS 运算符
- 子查询中的空值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

单行子查询

- 仅返回一行
- 使用单行比较运算符

运算符	含义
=	等于
>	大于
>=	大于或等于
<	小于
<=	小于或等于
<>	不等于

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

单行子查询

单行子查询是从内部 SELECT 语句中返回一行的一种查询。此类子查询使用单行运算符。幻灯片给出了单行运算符的列表。

示例：

显示其职务 ID 与雇员 141 的职务 ID 相同的雇员：

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
        (SELECT job_id
         FROM   employees
         WHERE  employee_id = 141);
```

	LAST_NAME	JOB_ID
1	Rajs	ST_CLERK
2	Davies	ST_CLERK
3	Matos	ST_CLERK
4	Vargas	ST_CLERK

执行单行子查询

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = (SELECT job_id
                FROM employees
                WHERE last_name = 'Taylor')
AND salary > (SELECT salary
              FROM employees
              WHERE last_name = 'Taylor');
```

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

执行单行子查询

可以将 SELECT 语句看作一个查询块。幻灯片中的示例显示其职务与 Taylor 相同但薪金高于 Taylor 的雇员。

该示例由三个查询块组成：一个外部查询和两个内部查询。先执行内部查询块，生成的查询结果分别为 SA_REP 和 8600。然后可以处理外部查询块，使用内部查询返回的值来完善其搜索条件。

两个内部查询都返回单个值（分别为 SA_REP 和 8600），因此将此 SQL 语句称为单行子查询。

注：外部查询和内部查询可以从不同的表中获得数据。

在子查询中使用组函数

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary = (SELECT MIN(salary)
                FROM employees);
```

	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在子查询中使用组函数

通过在子查询中使用组函数来返回单行，可以从主查询中显示数据。子查询包含在括号中，位于比较条件之后。

幻灯片中的示例显示其薪金等于最低薪金的所有雇员的姓氏、职务 ID 和薪金。MIN 组函数将单个值 (2500) 返回给外部查询。

带有子查询的 HAVING 子句

- Oracle Server 会先执行子查询。
- Oracle Server 会将结果返回到主查询的 HAVING 子句中。

<pre> SELECT department_id, MIN(salary) FROM employees GROUP BY department_id HAVING MIN(salary) > (SELECT MIN(salary) FROM employees WHERE department_id = 50); </pre>		
		2500
		(SELECT MIN(salary)
		FROM employees
		WHERE department_id = 50);

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	20	6000
3	90	17000
4	110	8300
5	80	8600
6	10	4400
7	60	4200

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

带有子查询的 HAVING 子句

不仅可以在 WHERE 子句中使用子查询，还可以在 HAVING 子句中使用子查询。Oracle 服务器会执行子查询，并将结果返回到主查询的 HAVING 子句中。

幻灯片中的 SQL 语句显示最低薪金高于部门 50 的最低薪金的所有部门。

示例：

查找具有最低平均薪金的职务。

```

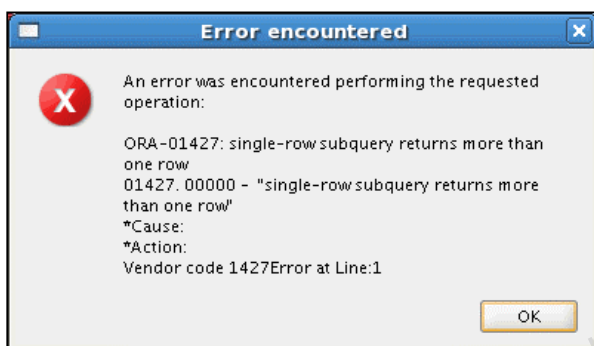
SELECT    job_id, AVG(salary)
FROM      employees
GROUP BY  job_id
HAVING    AVG(salary) = (SELECT    MIN(AVG(salary))
                        FROM      employees
                        GROUP BY  job_id);

```

	JOB_ID	AVG(SALARY)
1	ST_CLERK	2925

此语句中有什么错误

```
SELECT employee_id, last_name
FROM employees
WHERE salary =
      (SELECT MIN(salary)
       FROM employees
       GROUP BY department_id);
```



对多行子查询
使用了单行运算符

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

此语句中有什么错误

子查询的常见错误是单行子查询返回了多行。

在幻灯片的 SQL 语句中，子查询包含一个 GROUP BY 子句，这意味着该子查询将返回多行，每行都对应于一个它找到的组。这种情况下，子查询的结果为 4400、6000、2500、4200、7000、17000 和 8300。

外部查询将采用这些结果，并在其 WHERE 子句中使用它们。该 WHERE 子句包含一个等于 (=) 运算符，该运算符是一个只需要一个值的单行比较运算符。= 运算符无法接受子查询中的多个值，因此产生了错误。

要更正此错误，请将 = 运算符更改为 IN。

内部查询没有返回任何行

```
SELECT last_name, job_id
FROM employees
WHERE job_id =
      (SELECT job_id
       FROM employees
       WHERE last_name = 'Haas');
```

0 rows selected

因为没有名为“Haas”的雇员，所以子查询没有返回任何行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

内部查询没有返回任何行

子查询的另外常见问题是内部查询没有返回任何行。

在幻灯片的 SQL 语句中，子查询包含一个 WHERE 子句。据推测，该语句的目的是要查找名为 Haas 的雇员。该语句正确，但是没有名为“Haas”的雇员，所以执行时没有选择任何行。因此，子查询不会返回任何行。

外部查询采用子查询的结果（空值），并在其 WHERE 子句中使用这些结果。外部查询没有找到职务 ID 等于空值的雇员，因此不会返回任何行。即使存在值为空的职务，也不会返回行，因为两个空值的比较会产生一个空值，从而使 WHERE 条件不为“真”。

课程安排

- 子查询：类型、语法和准则
- 单行子查询：
 - 子查询中的组函数
 - 带有子查询的 HAVING 子句
- 多行子查询
 - 使用 IN、ALL 或 ANY
- 使用 EXISTS 运算符
- 子查询中的空值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

多行子查询

- 返回多个行
- 使用多行比较运算符

运算符	含义
IN	等于列表中的任意一个成员。
ANY	前面必须是 =、!=、>、<、<=、>=。将某个值与列表中的每个值或查询返回的每个值进行比较。如果查询没有返回任何行，则求值结果为 FALSE。
ALL	前面必须是 =、!=、>、<、<=、>=。将某个值与列表中的每个值或查询返回的每个值进行比较。如果查询没有返回任何行，则求值结果为 TRUE。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

多行子查询

返回多行的子查询被称为多行子查询。应对多行子查询使用多行运算符，而不是使用单行运算符。多行运算符需要一个或多个值：

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MIN(salary)
                  FROM employees
                  GROUP BY department_id);
```

示例：

查找其薪金等于各个部门最低薪金的雇员。

先执行内部查询，生成一个查询结果。然后可以处理主查询块，使用内部查询返回的值完善其搜索条件。事实上，主查询将以下面的形式出现在 Oracle Server 上：

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);
```

在多行子查询中使用 ANY 运算符

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在多行子查询中使用 ANY 运算符

ANY 运算符（及其同义词 SOME 运算符）用于将某个值与子查询返回的每个值进行比较。幻灯片中的示例显示不是 IT 程序员且薪金低于任一 IT 程序员的雇员。程序员的最高薪金为 \$9,000。

- <ANY 表示低于最高值。
- >ANY 表示高于最低值。
- =ANY 等同于 IN。

在多行子查询中使用 ALL 运算符

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

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

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在多行子查询中使用 ALL 运算符

ALL 运算符用于将某个值与子查询返回的每个值进行比较。幻灯片中的示例显示薪金低于职务 ID 为 IT_PROG 的任何雇员的薪金且职务不是 IT_PROG 的雇员。

>ALL 表示大于最高值，而 <ALL 表示小于最低值。

NOT 运算符可以与 IN、ANY 和 ALL 运算符一起使用。

使用 EXISTS 运算符

```
SELECT * FROM departments
WHERE NOT EXISTS
(SELECT * FROM employees
WHERE employees.department_id=departments.department_id);
```

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	190	Contracting	(null)	1700

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 EXISTS 运算符

在查询中使用 EXISTS 运算符，查询结果取决于某些行是否在表中存在。如果子查询至少返回一行，则求值结果为 TRUE。

幻灯片中的示例显示没有雇员的部门。对于 DEPARTMENTS 表中每一行，检查条件，看一看在 EMPLOYEES 表中是否存在具有相同部门 ID 的行。如果不存在这样的行，则相应行满足条件，从而选择该行。如果 EMPLOYEES 表中存在相应行，则不选择该行。

课程安排

- 子查询：类型、语法和准则
- 单行子查询：
 - 子查询中的组函数
 - 带有子查询的 HAVING 子句
- 多行子查询
 - 使用 ALL 或 ANY 运算符
- 使用 EXISTS 运算符
- 子查询中的空值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

子查询中的空值

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

0 rows selected

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

子查询中的空值

幻灯片中的 SQL 语句尝试显示没有任何下属的所有雇员。逻辑上，此 SQL 语句应该返回 12 行。但是，该 SQL 语句没有返回任何行。因为内部查询返回的值中有一个为空值，所以整个查询不会返回任何行。

原因是所有条件与空值进行比较后都会产生一个空值。因此，只要子查询的结果集中可能包含空值，就不要使用 NOT IN 运算符。NOT IN 运算符等同于 <> ALL。

请注意，如果使用的是 IN 运算符，则子查询的结果集中存在空值就不会成为问题。IN 运算符等同于 =ANY。例如，要显示具有下属的雇员，可以使用下面的 SQL 语句：

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
      (SELECT mgr.manager_id
       FROM   employees mgr);
```

另外，可以在子查询中包括 WHERE 子句，用来显示没有下属的所有雇员：

```
SELECT last_name FROM employees
WHERE  employee_id NOT IN
      (SELECT manager_id
       FROM   employees
       WHERE  manager_id IS NOT NULL);
```

小测验

使用子查询等同于执行两个连续的查询，而且将第一个查询的结果用作第二个查询中的搜索值。

1. 正确
2. 错误

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1

小结

在本课中，您应该已经学会：

- 确定在什么情况下子查询可以帮助解决问题
- 在查询基于未知值时编写子查询

```
SELECT    select_list
FROM      table
WHERE     expr operator
          (SELECT select_list
           FROM     table);
```

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子查询。子查询是一个 SELECT 语句，它嵌入到另一个 SQL 语句的子句中。当查询基于带有未知中间值的搜索标准时，使用子查询非常有帮助。

子查询具有以下特性：

- 可以将一行数据传递给包含单行运算符（如 =、<>、>、>=、< 或 <=）的主语句
- 可以将多行数据传递给包含多行运算符（如 IN）的主语句
- Oracle Server 先处理子查询，随后 WHERE 或 HAVING 子句会使用生成的结果
- 可以包含组函数

练习 7：概览

本练习包含以下主题：

- 创建查询值基于未知标准的子查询
- 使用子查询找出仅存在于一个数据集而不存在于其它数据集集中的值

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 7：概览

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。

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.

8 使用集合运算符

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：

- 描述集合运算符
- 使用集合运算符将多个查询组成一个查询
- 控制返回行的顺序

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

课程目标

在本课中，您将学习如何使用集合运算符编写查询。

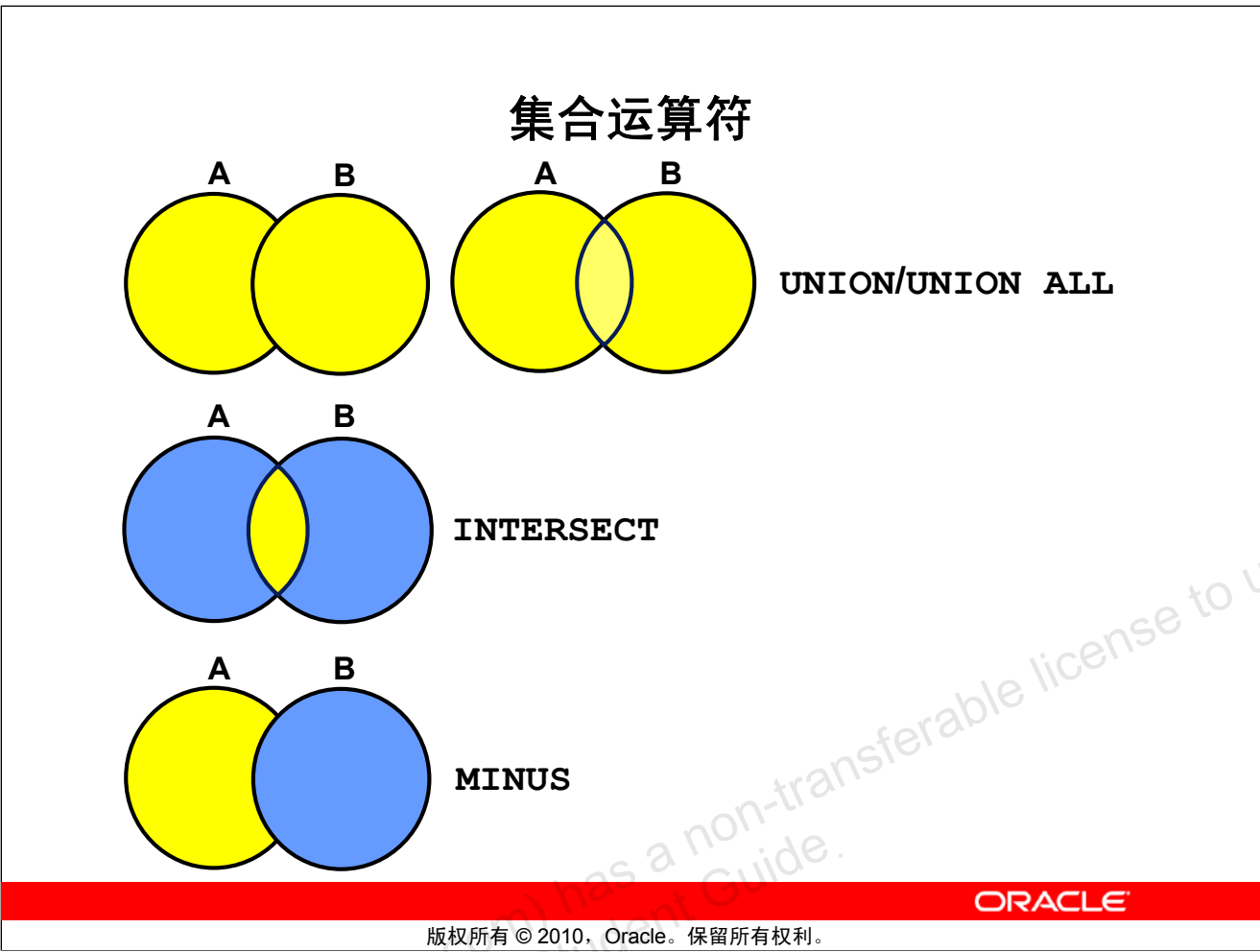
课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Unauthorized reproduction or distribution prohibited. Copyright© 2014, Oracle and/or its affiliates.



集合运算符

集合运算符用于将两个或更多查询的结果合并成一个结果。包含集合运算符的查询被称为复合查询。

运算符	返回
UNION	从两个查询中返回不包括重复行的那些行
UNION ALL	从两个查询中返回行，其中包括所有重复行
INTERSECT	返回两个查询的共同行
MINUS	返回第一个查询中有但第二个查询中没有的行

所有集合运算符都具有相同的优先级。如果 SQL 语句包含多个集合运算符，在没有使用括号明确地指定其它顺序时，Oracle Server 会从左（上）到右（下）对这些运算符进行计算。在将 INTERSECT 运算符与其它集合运算符配合使用的查询中，您应使用括号明确地指定计算的顺序。

集合运算符准则

- SELECT 列表中的表达式在数量上必须匹配。
- 第二个查询中每一列的数据类型必须与第一个查询中对应列的数据类型相匹配。
- 可以使用括号更改执行顺序。
- ORDER BY 子句只能出现在语句的末尾。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

集合运算符准则

- 查询的 SELECT 列表中的表达式在数量和数据类型上必须匹配。在 WHERE 子句中使用 UNION、UNION ALL、INTERSECT 和 MINUS 运算符的查询，其 SELECT 列表中的列必须具有相同的数量和数据类型。对于复合查询中包含的各个查询，其 SELECT 列表中各个列的数据类型可能不完全相同。第二个查询中的列必须与第一个查询中的对应列属于相同的数据类型组（如数字或字符）。
- 可在子查询中使用集合运算符。
- 在将 INTERSECT 运算符与其它集合运算符配合使用的查询中，您应使用括号指定计算的顺序。这可确保符合新出台的 SQL 标准，从而为 INTERSECT 运算符赋予比其它集合运算符更高的优先级。

Oracle Server 和集合运算符

- 除非使用 UNION ALL 运算符，否则会自动删除重复行。
- 第一个查询中的列名将显示在结果中。
- 除非使用 UNION ALL 运算符，否则默认情况下输出按升序进行排序。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

Oracle Server 和集合运算符

如果查询使用了集合运算符，则 Oracle Server 会自动删除重复行，除非使用的是 UNION ALL 运算符。输出中的列名由第一个 SELECT 语句中列的列表确定。默认情况下，输出按 SELECT 子句第一列的升序进行排序。

对于复合查询包含的各个查询，其 SELECT 列表中的相应表达式在数量和数据类型上必须是相匹配的。如果这些查询选择了字符数据，则会按如下方式确定返回值的数据类型：

- 如果两个查询选择了数据类型为 CHAR 且长度相等的值，则返回值的数据类型也为 CHAR 且长度保持不变。如果两个查询选择了数据类型为 CHAR 但长度不同的值，则返回值的数据类型为 VARCHAR2，长度为较大的 CHAR 值。
- 如果其中一个或两个查询选择了数据类型为 VARCHAR2 的值，则返回值的数据类型也为 VARCHAR2。

如果这些查询选择了数字数据，则按数字优先级确定返回值的数据类型。如果所有查询选择了数据类型为 NUMBER 的值，则返回值的数据类型也为 NUMBER。在使用集合运算符的查询中，Oracle Server 不会执行跨数据类型组的隐式转换。因此，如果这些查询的相应表达式解析为字符数据和数字数据，Oracle Server 则会返回错误。

课程安排

- 集合运算符：类型和准则
- **本课中使用的表**
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

本课中使用的表

本课中使用的表包括：

- EMPLOYEES：提供有关所有当前雇员的详细信息
- JOB_HISTORY：记录雇员更换职务时先前职务的开始日期和结束日期、职务标识号及部门的详细信息

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

本课中使用的表

本课中使用了两个表：EMPLOYEES 表和 JOB_HISTORY 表。

您已经熟悉 EMPLOYEES 表，该表中存储了雇员的详细信息，如唯一标识号、电子邮件地址、职务标识号（如 ST_CLERK、SA_REP 等等）、薪金和经理等等。

一些雇员已经在公司工作了很长时间，已更换为其它职务。因此可以使用 JOB_HISTORY 表对该信息进行监视。在雇员更换职务时，先前职务的开始日期和结束日期、job_id（如 ST_CLERK、SA_REP 等等）以及部门的详细信息都记录在 JOB_HISTORY 表中。

后面几页中将显示 EMPLOYEES 表和 JOB_HISTORY 表的结构和数据。

本课中使用的表（续）

很多情况下，公司中的某些人员会在公司任期内多次担任相同的职务。例如，假定雇员 Taylor 在 1998 年 3 月 24 日加入公司。Taylor 在 1998 年 3 月 24 日至 1998 年 12 月 31 日期间担任职务 SA_REP，在 1999 年 1 月 1 日至 1999 年 12 月 31 日期间担任职务 SA_MAN。此后 Taylor 又再次担任职务 SA_REP 一直到现在。

```
DESCRIBE employees
```

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)
11 rows selected		

本课中使用的表（续）

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

	EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
1	200	Whalen	AD_ASST	17-SEP-87	10
2	201	Hartstein	MK_MAN	17-FEB-96	20
3	202	Fay	MK_REP	17-AUG-97	20
4	205	Higgins	AC_MGR	07-JUN-94	110
5	206	Gietz	AC_ACCOUNT	07-JUN-94	110
6	100	King	AD_PRES	17-JUN-87	90
7	101	Kochhar	AD_VP	21-SEP-89	90
8	102	De Haan	AD_VP	13-JAN-93	90
9	103	Hunold	IT_PROG	03-JAN-90	60
10	104	Ernst	IT_PROG	21-MAY-91	60
11	107	Lorentz	IT_PROG	07-FEB-99	60
12	124	Mourgos	ST_MAN	16-NOV-99	50
13	141	Rajs	ST_CLERK	17-OCT-95	50
14	142	Davies	ST_CLERK	29-JAN-97	50
15	143	Matos	ST_CLERK	15-MAR-98	50
16	144	Vargas	ST_CLERK	09-JUL-98	50
17	149	Zlotkey	SA_MAN	29-JAN-00	80
18	174	Abel	SA_REP	11-MAY-96	80
19	176	Taylor	SA_REP	24-MAR-98	80
20	178	Grant	SA_REP	24-MAY-99	(null)

```
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)
5 rows selected		

本课中使用的表（续）

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

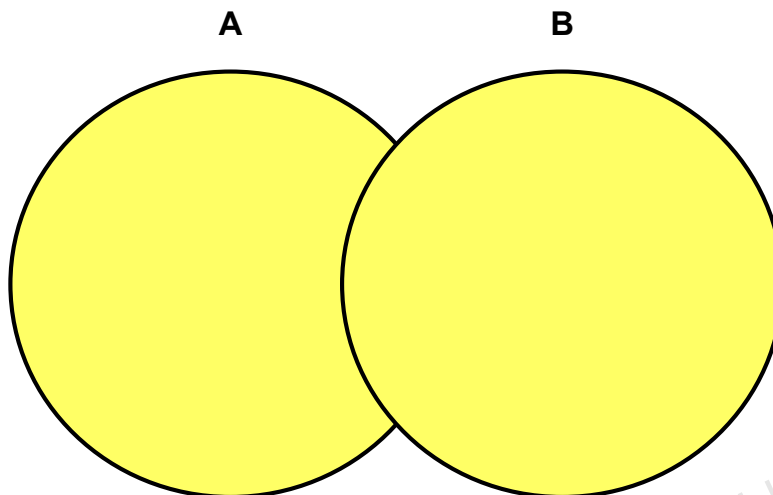
课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

UNION 运算符



UNION 运算符从两个查询中返回不包括重复行的那些行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

UNION 运算符

UNION 运算符用于返回由任一查询选定的所有行。使用 UNION 运算符可以返回多个表中的所有行，但不包括重复行。

准则

- 所选列的数量必须相同。
- 所选列的数据类型必须属于相同的数据类型组（如数字或字符）。
- 列名不必相同。
- UNION 将对所有选定的列执行操作。
- 在重复项检查过程中不会忽略 NULL 值。
- 默认情况下，输出按 SELECT 子句中列的升序进行排序。

使用 UNION 运算符

显示所有雇员的当前职务和先前职务的详细信息。每位雇员仅显示一次。

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

EMPLOYEE_ID	JOB_ID
1	100 AD_PRES
2	101 AC_ACCOUNT
...	
22	200 AC_ACCOUNT
23	200 AD_ASST
...	
27	205 AC_MGR
28	206 AC_ACCOUNT

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 UNION 运算符

UNION 运算符会删除所有重复记录。如果出现在 EMPLOYEES 表和 JOB_HISTORY 表中的记录是相同的，则这些记录只显示一次。在幻灯片显示的输出中，您会看到由于每行中的 JOB_ID 不同，EMPLOYEE_ID 为 200 的雇员的记录出现了两次。

请看如下示例：

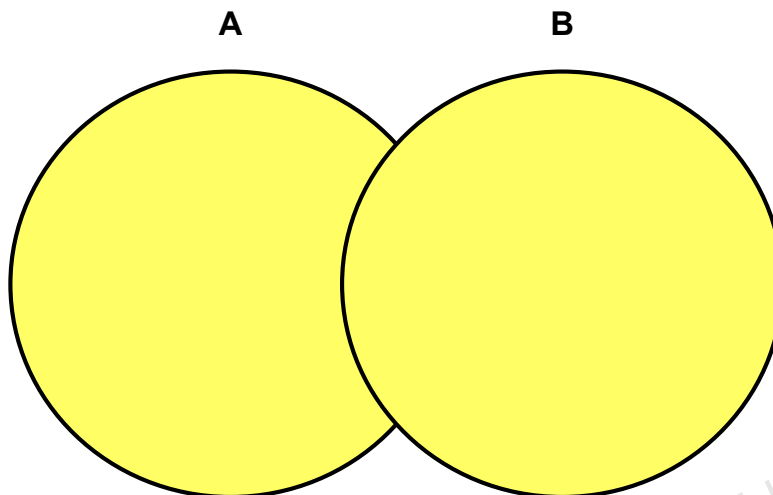
```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
...		
22	200 AC_ACCOUNT	90
23	200 AD_ASST	10
24	200 AD_ASST	90
...		
29	206 AC_ACCOUNT	110

使用 UNION 运算符（续）

在前面的输出中，雇员 200 出现了三次。为什么？请注意雇员 200 的 DEPARTMENT_ID 值。一行的 DEPARTMENT_ID 为 90，另一行为 10，第三行为 90。由于职务 ID 和部门 ID 的组合是唯一的，所以雇员 200 的每行都是唯一的，因此没有被认为是重复行。请注意，输出按 SELECT 子句的第一列（在此例中，为 EMPLOYEE_ID）的升序进行排序。

UNION ALL 运算符



UNION ALL 运算符从两个查询中返回行，包括所有重复行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

UNION ALL 运算符

使用 UNION ALL 运算符可以返回多个查询中的所有行。

准则

适用于 UNION 和 UNION ALL 的准则是相同的，除了下面适用于 UNION ALL 的两个例外：与 UNION 不同，默认情况下 UNION ALL 不会删除重复行，而且不对输出进行排序。

使用 UNION ALL 运算符

显示所有雇员的当前部门和先前部门。

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100 AD_PRES	90
...		
17	149 SA_MAN	80
18	174 SA_REP	80
19	176 SA_REP	80
20	176 SA_MAN	80
21	176 SA_REP	80
22	178 SA_REP	(null)
23	200 AD_ASST	10
...		
30	206 AC_ACCOUNT	110

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 UNION ALL 运算符

在此示例中，选择了 30 行。在两个表中共选择了 30 行。UNION ALL 运算符不会删除重复行。UNION 将返回由任一查询选定的所有不同行。而 UNION ALL 将返回由任一查询选定的所有行，包括所有重复行。请看幻灯片中的查询，现在使用 UNION 子句编写该查询：

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

上面的查询会返回 29 行。这是因为此查询删除了后面一行（因为该行是重复行）：

176 SA_REP	80
------------	----

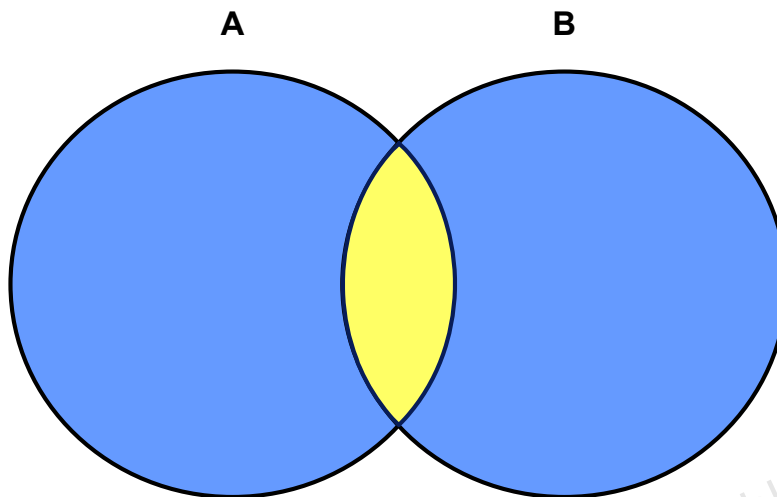
课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- **INTERSECT 运算符**
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

INTERSECT 运算符



INTERSECT 运算符将返回两个查询的共同行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

INTERSECT 运算符

使用 INTERSECT 运算符可以返回多个查询的所有共同行。

准则

- 在查询中使用的所有 SELECT 语句中，由查询中的 SELECT 语句选定的列数和列的数据类型必须相同。不过，列名不必相同。
- 使相交的表按反方向排序不会更改结果。
- INTERSECT 不会忽略 NULL 值。

使用 INTERSECT 运算符

显示符合以下条件的雇员的雇员 ID 和职务 ID：这些雇员的当前职务与以前的职务相同，也就是说这些雇员曾担任过别的职务，但现在又重新担任了以前的同一职务。

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

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 INTERSECT 运算符

在幻灯片中的示例中，查询只返回在两个表的选定列中具有相同值的那些记录。

如果将 DEPARTMENT_ID 列添加到 EMPLOYEES 表的 SELECT 语句，将 DEPARTMENT_ID 列添加到 JOB_HISTORY 表的 SELECT 语句，然后运行此查询，会产生什么结果？由于引入的其它列的值可能重复，也可能不重复，因此结果可能不同。

示例：

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

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	176	SA_REP	80

由于 EMPLOYEES.DEPARTMENT_ID 值与 JOB_HISTORY.DEPARTMENT_ID 值不相同，所以结果中不再包含雇员 200。

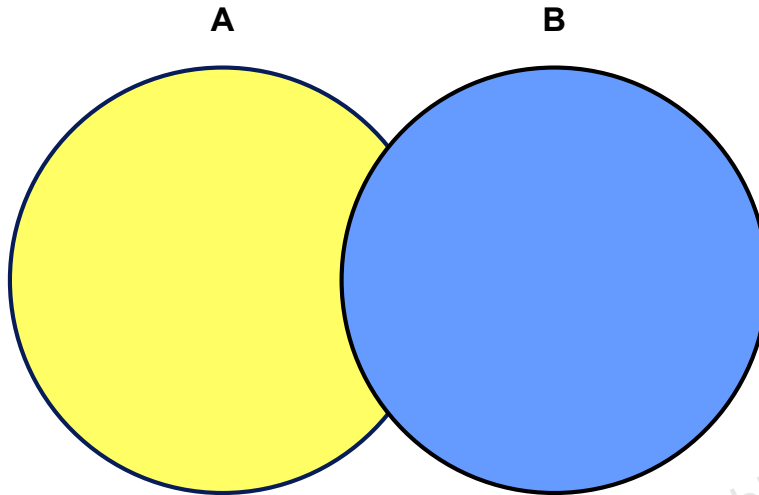
课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

MINUS 运算符



MINUS 运算符将返回由第一个查询选定的但没有出现在第二个查询结果集中的所有不同行。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

MINUS 运算符

使用 MINUS 运算符可以返回由第一个查询选定的但没有出现在第二个查询结果集中的所有不同行（第一个 SELECT 语句减去第二个 SELECT 语句）。

注：在查询中使用的所有 SELECT 语句中，由查询中的 SELECT 语句选定的列数必须相同，而且列的数据类型必须属于相同的数据类型组。不过，列名不必相同。

使用 MINUS 运算符

显示从未更换过职务的雇员的雇员 ID。

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
...	
13	202
14	205
15	206

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

使用 MINUS 运算符

在幻灯片中的示例中，用 EMPLOYEES 表中的雇员 ID 减去 JOB_HISTORY 表中的雇员 ID。结果集中显示运行减法运算后剩下的雇员，这些雇员由存在于 EMPLOYEES 表中但并不存在于 JOB_HISTORY 表中的那些行代表。这些行记录了从未更换过职务的雇员。

课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- **匹配 SELECT 语句**
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

匹配 SELECT 语句

- 使用 UNION 运算符可显示位置 ID、部门名称和所在的省/市/自治区。
- 如果某些列不存在于一个表或另一个表中，则必须使用 TO_CHAR 函数或任何其它转换函数匹配数据类型。

```
SELECT location_id, department_name "Department",  
       TO_CHAR(NULL) "Warehouse location"  
FROM departments  
UNION  
SELECT location_id, TO_CHAR(NULL) "Department",  
       state_province  
FROM locations;
```



版权所有 © 2010, Oracle。保留所有权利。

匹配 SELECT 语句

由于查询的 SELECT 列表中的表达式必须在数量上相匹配，因此可以使用伪列和数据类型转换函数来遵循此规则。在幻灯片中，已将名称 Warehouse location 指定为伪列标题。在第一个查询中使用了 TO_CHAR 函数，以匹配由第二个查询检索的 state_province 列的 VARCHAR2 数据类型。同样，第二个查询中的 TO_CHAR 函数用于匹配由第一个查询检索的 department_name 列的 VARCHAR2 数据类型。

该查询的输出如下所示：

	LOCATION_ID	Department	Warehouse location
1	1400	IT	(null)
2	1400	(null)	Texas
3	1500	Shipping	(null)
4	1500	(null)	California
5	1700	Accounting	(null)
6	1700	Administration	(null)
7	1700	Contracting	(null)
8	1700	Executive	(null)

...

匹配 SELECT 语句：示例

使用 UNION 运算符可显示所有雇员的雇员 ID、职务 ID 和薪金。

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000
...			
29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

匹配 SELECT 语句：示例

EMPLOYEES 和 JOB_HISTORY 表具有多个共同的列（例如，EMPLOYEE_ID、JOB_ID 和 DEPARTMENT_ID）。但是，如果您知道薪金仅存在于 EMPLOYEES 表中，而且要在查询中使用 UNION 运算符来显示雇员 ID、职务 ID 和薪金，这时又该如何操作？

幻灯片代码示例中将 EMPLOYEES 表和 JOB_HISTORY 表中的 EMPLOYEE_ID 列和 JOB_ID 列相匹配。将文字值 0 添加到 JOB_HISTORY SELECT 语句中，可匹配 EMPLOYEES SELECT 语句中 SALARY 列的数值。

在幻灯片显示结果中，对应于 JOB_HISTORY 表中的一条记录的每个输出行在 SALARY 列中都包含一个 0。

课程安排

- 集合运算符：类型和准则
- 本课中使用的表
- UNION 和 UNION ALL 运算符
- INTERSECT 运算符
- MINUS 运算符
- 匹配 SELECT 语句
- 在集合运算中使用 ORDER BY 子句

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在集合运算中使用 ORDER BY 子句

- ORDER BY 子句只能在复合查询的末尾出现一次。
- 这些查询中不能包含多个 ORDER BY 子句。
- ORDER BY 子句仅识别第一个 SELECT 查询中的列。
- 默认情况下，使用第一个 SELECT 查询中的第一列按升序对输出进行排序。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

在集合运算中使用 ORDER BY 子句

ORDER BY 子句只能在复合查询中使用一次。如果使用 ORDER BY 子句，则必须将其放在查询的末尾。ORDER BY 子句接受列名或别名。默认情况下，输出按第一个 SELECT 查询的第一列的升序进行排序。

注：ORDER BY 子句不能识别第二个 SELECT 查询中的列名。为避免列名混淆，通常的做法是对列位置执行 ORDER BY。

例如，在下面的语句中，输出将按 job_id 的升序显示。

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history
ORDER BY 2;
```

如果忽略 ORDER BY，则默认情况下输出将按 employee_id 的升序进行排序。您不能使用第二个查询中的列对输出进行排序。

小测验

找出集合运算符准则。

1. SELECT 列表中的表达式在数量上必须匹配。
2. 不可以使用括号更改执行顺序。
3. 第二个查询中每一列的数据类型必须与第一个查询中对应列的数据类型相匹配。
4. 如果不使用 UNION ALL 运算符，则 ORDER BY 子句只能在复合查询中使用一次。

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

答案：1、3

小结

在本课中，您应该已经学会：

- 使用 UNION 返回所有不同的行
- 使用 UNION ALL 返回所有行，包括重复行
- 使用 INTERSECT 返回两个查询共有的所有行
- 使用 MINUS 返回由第一个查询选定的但没有出现在第二个查询中的所有不同行
- 仅在语句的末尾使用 ORDER BY

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

小结

- UNION 运算符用于返回由复合查询中的每个查询选定的所有不同行。使用 UNION 运算符可以返回多个表中的所有行，但不包括重复行。
- 使用 UNION ALL 运算符可以返回多个查询中的所有行。与使用 UNION 运算符的情况不同，默认情况下 UNION ALL 不会删除重复行，也不对输出进行排序。
- 使用 INTERSECT 运算符可以返回多个查询的所有共同行。
- 使用 MINUS 运算符可以返回由第一个查询返回的且没有出现在第二个查询中的行。
- 请记住，只能在复合语句的末尾使用 ORDER BY 子句。
- 确保 SELECT 列表中的相应表达式在数量和数据类型上是相匹配的。

练习 8：概览

在本练习中，需要使用以下运算符创建报表：

- UNION 运算符
- INTERSECT 运算符
- MINUS 运算符

ORACLE

版权所有 © 2010, Oracle。保留所有权利。

练习 8：概览

在本练习中，您将使用集合运算符编写查询。

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.