幻灯片 1

进度表：

时间	**主题**
55 分钟	讲演
55 分钟	练习
110 分钟	总共

幻灯片 2

目标

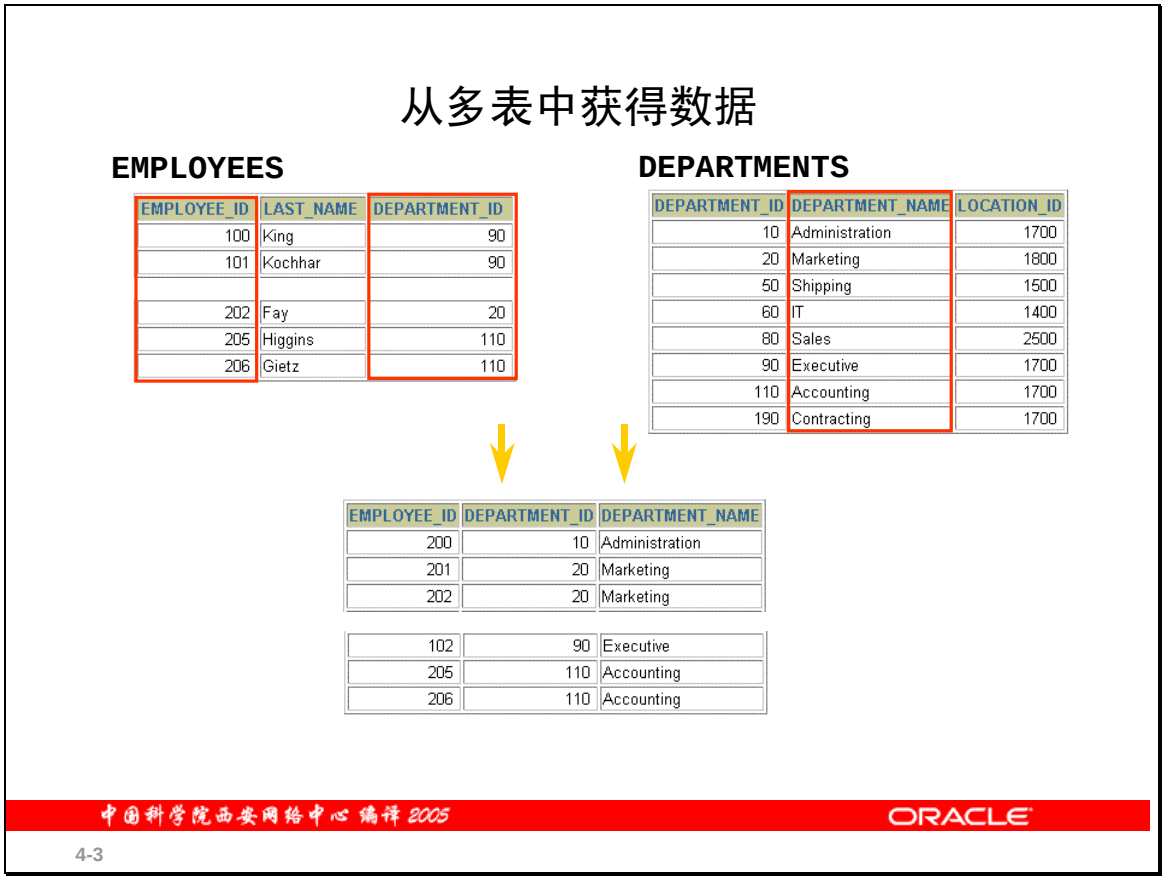
完成本课后，您应当能够执行下列操作：

- 写 SELECT 语句使用等值和非等值连接从多个表中访问数据
- 使用外连接查看不满足连接条件的数据
- 使用一个自连接，连接一个表到它自己

课程目标

本课学习怎样从多个表中获得数据。

幻灯片 3



来自多表的数据

有时你需要使用来自多表的数据。在幻灯片中，报告显示了来自单独的两个表的数据。

- Employee ID 在 EMPLOYEES 表中。
- Department ID 在 EMPLOYEES 和 DEPARTMENTS 表中都有。
- Location IDs 在 DEPARTMENTS 表中。

为了生成报告，你需要连接 EMPLOYEES 和 DEPARTMENTS 表，并从两个表中访问数据。

幻灯片 4

笛卡尔乘积

- 笛卡尔乘积的形成，当：
  - 一个连接条件被遗漏时
  - 一个连接条件不正确时
  - 在第一个表中的所有行被连接到第二个表的所有行时
- 为了避免笛卡尔乘积的形成，在 WHERE 子句中应当总是包含正确的连接条件

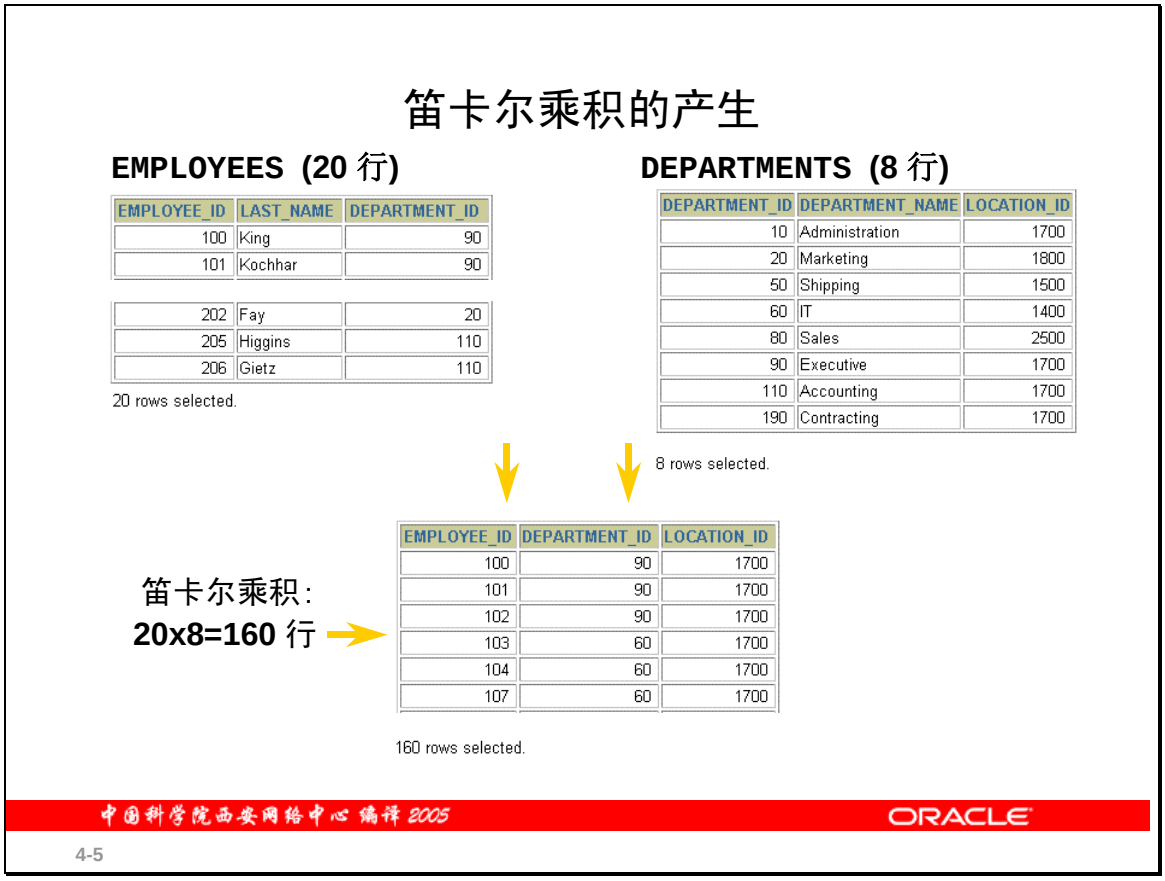
笛卡尔乘积

当一个连接条件无效或被遗漏时，其结果是一个笛卡尔乘积 (*Cartesian product*)，其中所有行的组合都被显示。第一个表中的所有行连接到第二个表中的所有行。

一个笛卡尔乘积会产生大量的行，其结果没有什么用。你应该在 WHERE 子句中始终包含一个有效的连接条件，除非你有特殊的需求，需要从所有表中组合所有的行。

对于一些测试笛卡尔乘积是有用的，例如你需要产生大量的行来模拟一个相当大的数据量。

幻灯片 5



笛卡尔乘积（续）

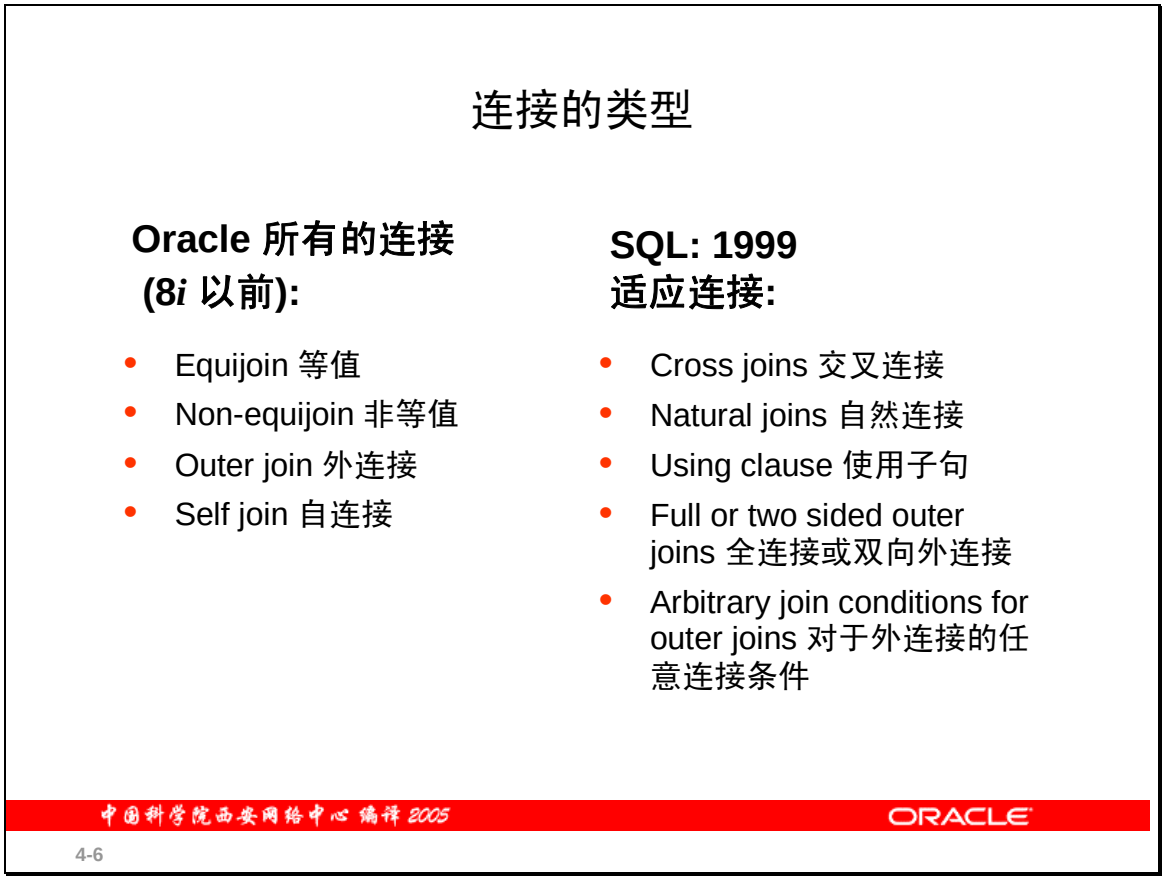
如果连接条件被遗漏，就会产生笛卡尔乘积。幻灯片中的例子从 EMPLOYEES 和 DEPARTMENTS 表中显示雇员的名字和部门名字。因为无 WHERE 子句被指定，EMPLOYEES 表中所有的行 (20 行) 被与 DEPARTMENTS 表中的所有行 (8 行) 连接，因此产生 160 行的输出。

```
SELECT last_name, department_name dept_name
FROM   employees, departments;
```

LAST_NAME	DEPT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Higgins	Contracting
Gietz	Contracting

160 rows selected.

幻灯片 6



连接的类型

Oracle9*i* 数据库提供 SQL: 1999 兼容的连接语法。在 9*i* 发布以前，连接语法不同于 ANSI 标准。新的 SQL: 1999 兼容连接语法不提供任何对 Oracle 以前发布的版本中私有连接语法性能的改进。

幻灯片 7

用 Oracle 语法连接表

使用一个连接从多个表中查询数据

```
SELECT   table1.column, table2.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

- 在 WHERE 子句中写连接条件
- 当多个表中有相同的列名时，将表名作为列名的前缀

定义连接

当数据从多表中查询时，要使用连接 (*join*) 条件。一个表中的行按照存在于相应列中的公值被连接到另一个表中的行，即，通常所说的主键和外键列。

从多个表中显示数据，在 WHERE 子句中写一个简单的连接条件。

在语法中：

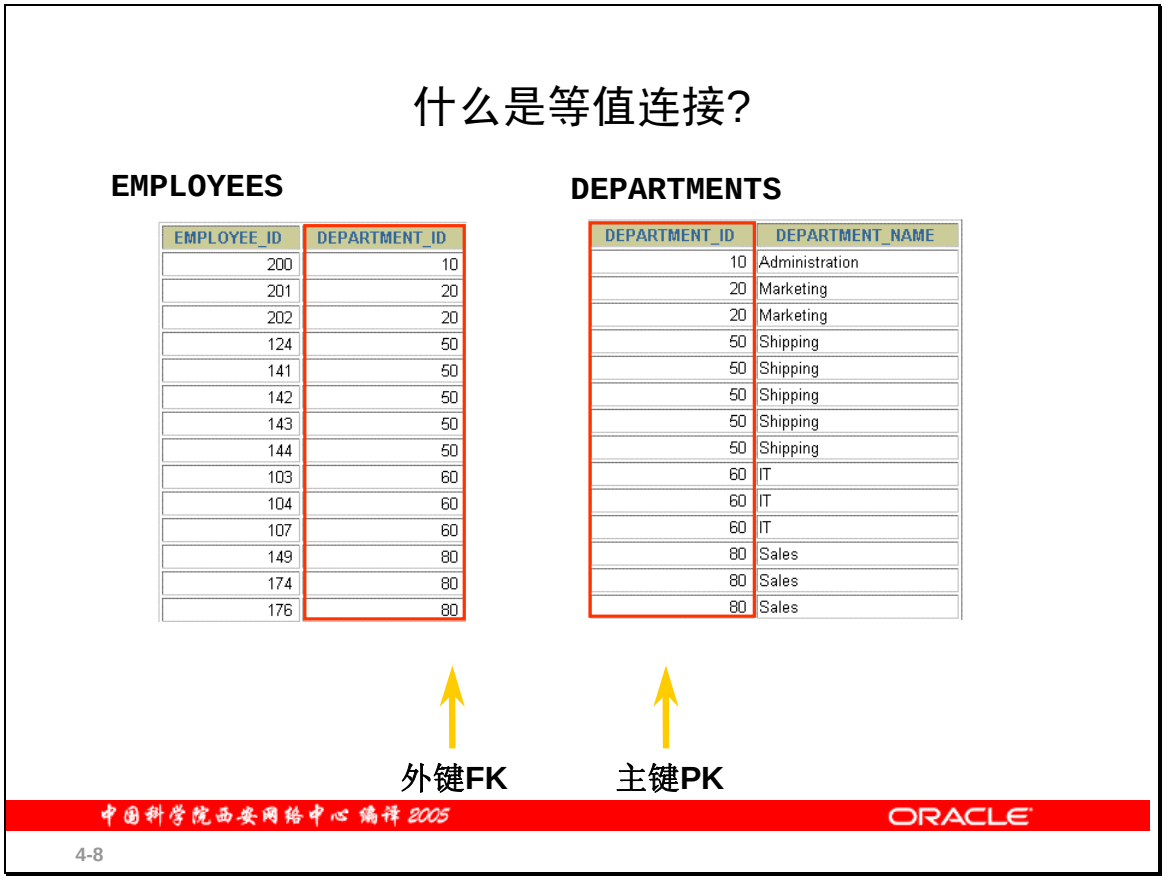
table1.column	指示获取数据的表和列
table1.column1 = table2.column2	是连接表的条件

原则

- 在写一个连接表的 SELECT 语句时，在列名前面用表名可以使语义清楚，并且加快数据库访问。
- 如果相同的列名出现在多个表中，列名必须前缀表名。
- 为了连接 *n* 个表在一起，你最少需要 n-1 个连接条件。例如，为了连接 4 个表，最少需要 3 个连接条件。如果表中有一个连接主键，该规则可能不适用，其中可能有多行用来唯一地标识每一行。

更多信息，见 *Oracle9i SQL Reference*, “SELECT”。

幻灯片 8



等值连接

为了确定一个雇员的部门名，需要比较 EMPLOYEES 表中的 DEPARTMENT_ID 列与 DEPARTMENTS 表中的 DEPARTMENT_ID 列的值。在 EMPLOYEES 和 DEPARTMENTS 表之间的关系是一个相等 (*equijoin*) 关系，即，两个表中 DEPARTMENT_ID 列的值必须相等。通常，这种连接类型包括主键和外键。

注：等值连接也被称为简单连接 (*simple joins*) 或内连接 (*inner joins*)。

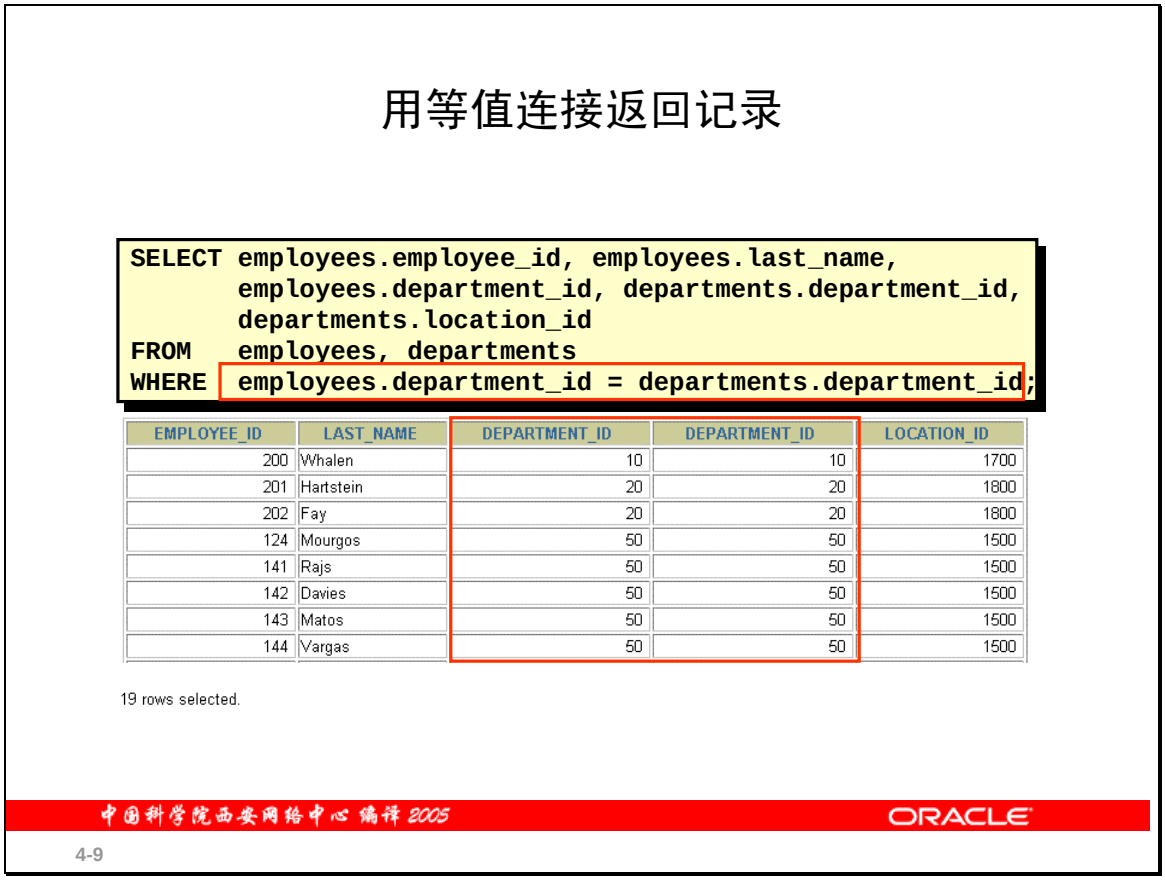
教师注释

解释抉择矩阵 (decision matrix) 用于简化写连接的使用，例如，如果你想显示同一个部门中所有姓 Goyal 的雇员的名字和部门号，可以写出下面的决策矩阵：

显示列	*源表*	*条件*
`last_name`	`employees`	`last_name='Goyal'`
`department_name`	`departments`	`employees.department_id = departments.department_id`

现在看着上面的抉择矩阵，SQL 语句可以容易地写出。第一列给出 SELECT 语句的字段列表，第二列给出 FROM 子句，第三列给出 WHERE 子句的条件。

幻灯片 9



用等值连接返回记录

在幻灯片的例子中：

- SELECT 子句指定要返回的列名：
  - employee last name、employee number 和 department number，这些是 EMPLOYEES 表中的列
  - department number、department name 和 location ID，这些是 DEPARTMENTS 表中的列
- FROM 子句指定数据库必须访问的两个表：
  - EMPLOYEES 表
  - DEPARTMENTS 表
- WHERE 子句指定表怎样被连接：
  EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID

因为 DEPARTMENT_ID 列是两个表的公共列，它必须用表名做前缀以避免混淆。

幻灯片 10

使用 AND 操作符附加搜索条件

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT

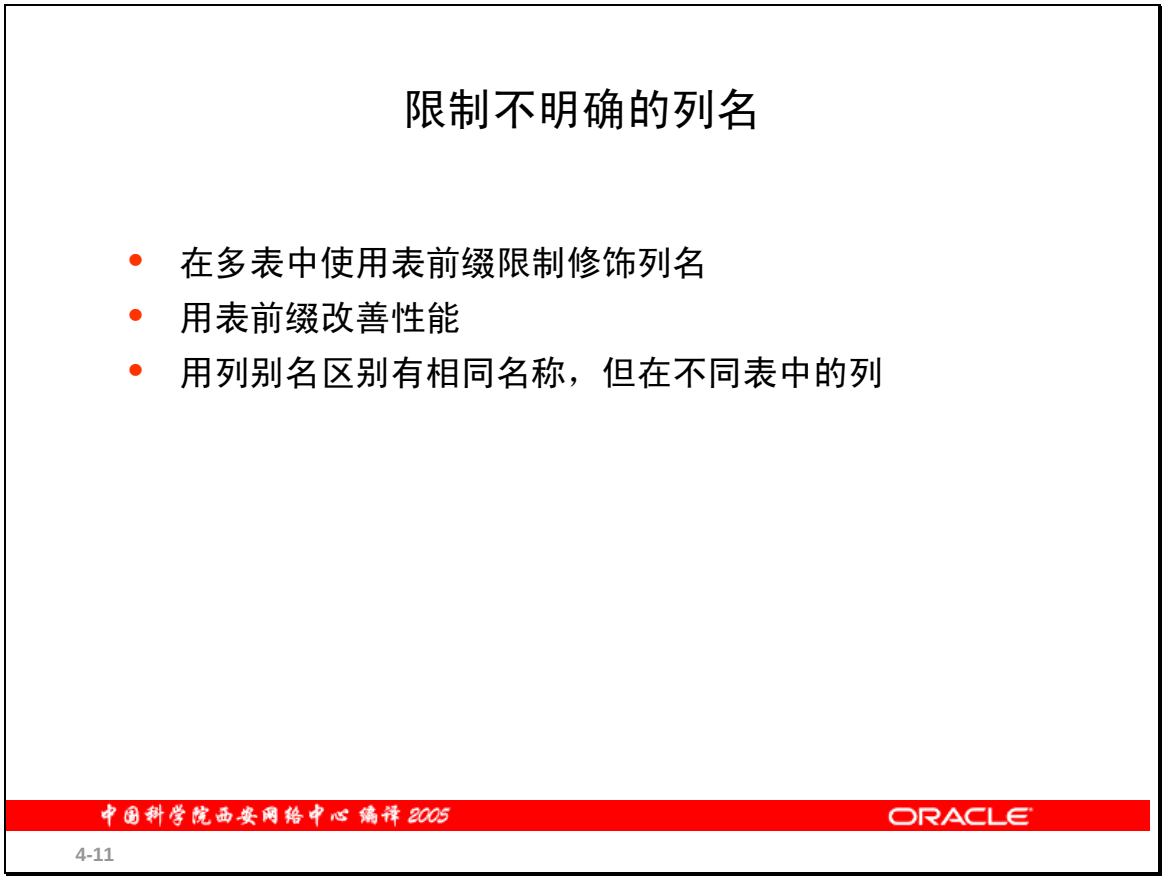
添加查询条件

除连接之外，你可能还要求用 WHERE 子句在连接中限制一个或多个表中的行。例如，为了显示雇员 Matos 的部门号和部门名，你需要添加条件到 WHERE 子句中。

```
SELECT last_name, employees.department_id,
       department_name
FROM   employees, departments
WHERE  employees.department_id = departments.department_id
AND    last_name = 'Matos';
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Matos	50	Shipping

幻灯片 11



限制不明确的列名

你需要在 WHERE 子句中用表的名字限制列的名字以避免含糊不清。没有表前缀，DEPARTMENT_ID 列可能来自 DEPARTMENTS 表，也可能来自 EMPLOYEES 表，这种情况下需要添加表前缀来执行查询。

如果列名在两个表之间不相同，就不需要限定列。但是，使用表前缀可以改善性能，因为你确切地告诉 Oracle 服务器在那里找到列。

必须限定不明确的列名也适用于在其它子句中可能引起混淆的那些列，例如 SELECT 子句或 ORDER BY 子句。

幻灯片 12

使用表别名

- 使用表别名简化查询
- 使用表别名改善性能

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e , departments d
WHERE  e.department_id = d.department_id;
```


表别名

用表名限制列名可能是非常耗时的，特别是当表名字很长时，你可以使用表别名代替表名。就象列别名给列另一个名字一样，表别名给表另一个名字。表别名有助于保持 SQL 代码较小，因此使用的存储器也少。

注意在例子中表的 FROM 子句中怎样定义表别名。表名完全指定，然后跟着别名。EMPLOYEES 表被给予别名 e，DEPARTMENTS 表被给予别名 d。

原则

- 表别名最多可以有 30 个字符，但短一些更好。
- 如果在 FROM 子句中表别名被用于指定的表，那么在整个 SELECT 语句中都要使用表别名。
- 表别名应该是有意义的。
- 表别名只对当前的 SELECT 语句有效。

幻灯片 13

多于两个表的连接

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Zlotkey	80
Abel	80
Taylor	80

20 rows selected.

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

- 为了连接 *n* 个表，你最少需要 n-1 个连接条件。例如，为了连接 3 个表，最少需要两个连接

添加查询条件

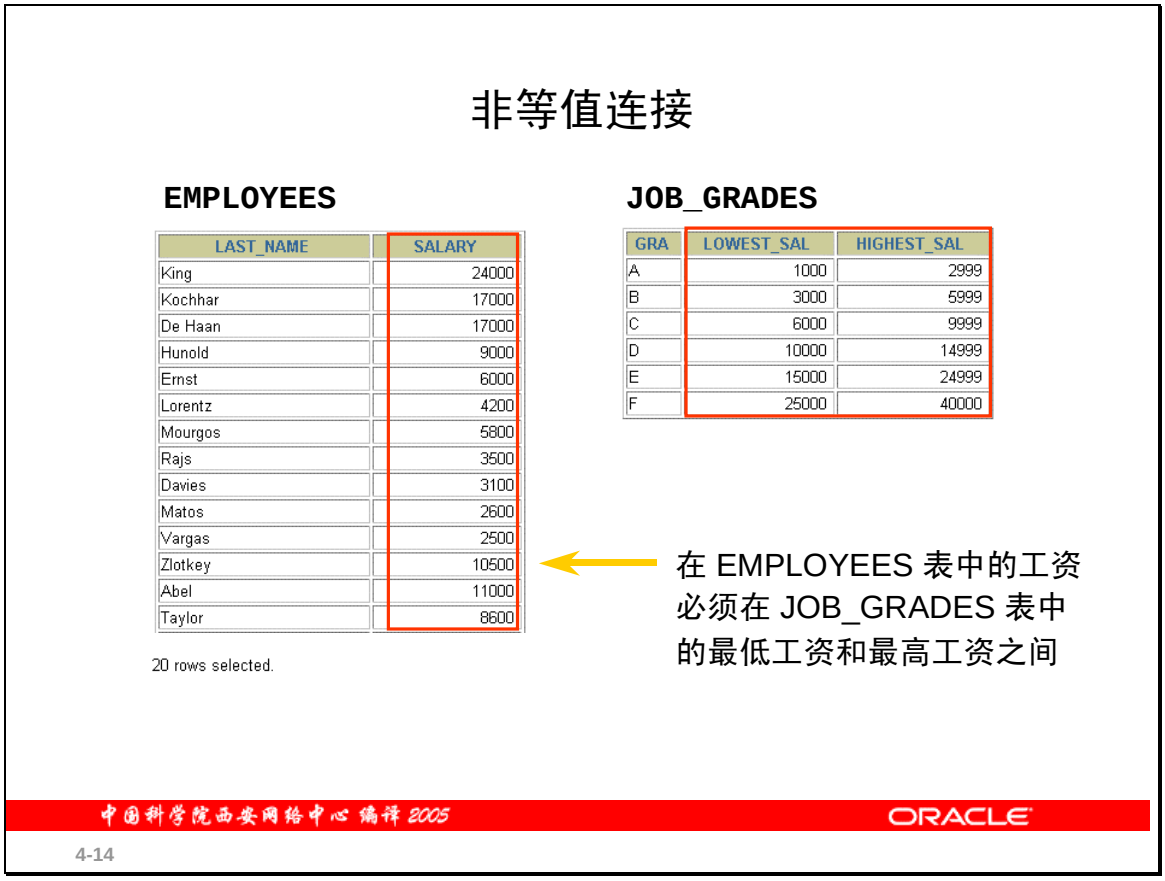
有时你可能需要连接两个以上的表。例如，为了显示每个雇员的 last name、department name 和 city，你必须连接 EMPLOYEES、DEPARTMENTS 和 LOCATIONS 表。

```
SELECT e.last_name, d.department_name, l.city
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
Rajs	Shipping	South San Francisco
...	...	...
Taylor	Sales	Oxford

19 rows selected.

幻灯片 14



非等值连接

一个非等值连接是一种不同于等值操作的连接条件。 EMPLOYEES 表和 JOB_GRADES A 表之间的关系有一个非等值连接例子。在两个表之间的关系是 EMPLOYEES 表中的 SALARY 列必须是 JOB_GRADES 表的 LOWEST_SALARY 和 HIGHEST_SALARY 列之间的值。使用不同于等于 (=) 的操作符获得关系。

幻灯片 15

用非等值连接返回记录

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

20 rows selected.

非等值连接（续）

幻灯片的例子创建一个非等值连接来求一个雇员的薪水级别。薪水必须在任何一对最低和最高薪水范围内。

要注意的是当查询被执行时，所有雇员只出现一次是重要的。没有雇员在列表中重复。对此有两个理由：

- 在工作等级表中，没有行是交迭的，即，一个雇员的薪水值只能位于薪水级别表的最低和最高薪水值之间。
- 所有雇员的薪水位于由工作级别表提供的限制中。即，没有雇员的收入少于 LOWEST_SAL 列所包含的最低值，或高于 HIGHEST_SAL 列所包含的最高值。

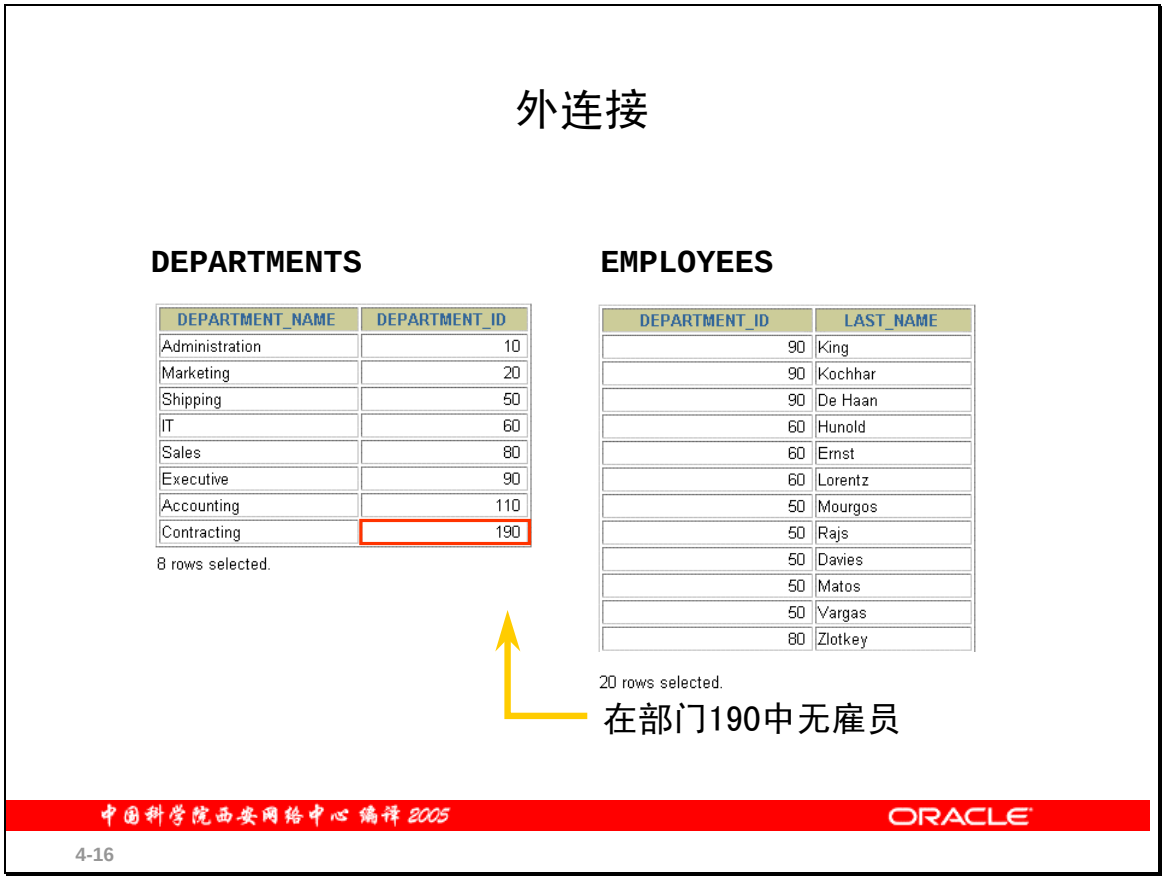
注：其它条件，例如 <= 和 >= 可以被使用，但 BETWEEN 是最简单的。在使用 BETWEEN 时先指定最低值后指定最高值。

在幻灯片的例子中指定表别名是因为性能的原因，而不是因为可能产生含糊。

教师注释

解释 BETWEEN … AND … 实际上被 Oeacle 服务器转换为一对 AND 条件 (a >= 最小值) and (a <= 最大值) ，IN (…) 被 Oracle 服务器转换为一组 OR 条件 (a = value1 OR a = value2 OR a = value3)。所以用 BETWEEN … AND … 、IN(…) 并没有性能上的提高；好处是逻辑上简单。

幻灯片 16



用外连接返回不直接匹配的记录

如果一个行不满足连接条件，该行将不出现在查询结果中。例如，在 EMPLOYEES 和 DEPARTMENTS 表的等值连接条件中，雇员 Grant 不出现，因为在 EMPLOYEES 表中没有她的 department ID 记录。在结果集中有 20 个雇员，你只看得见 19 个记录。

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Higgins	110	Accounting
Gietz	110	Accounting

19 rows selected.

幻灯片 17

外连接语法

- 你可以用一个外连接查看那些不满足连接条件的行
- 外连接运算符是加号 (+)

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```


用外连接返回不直接匹配的记录

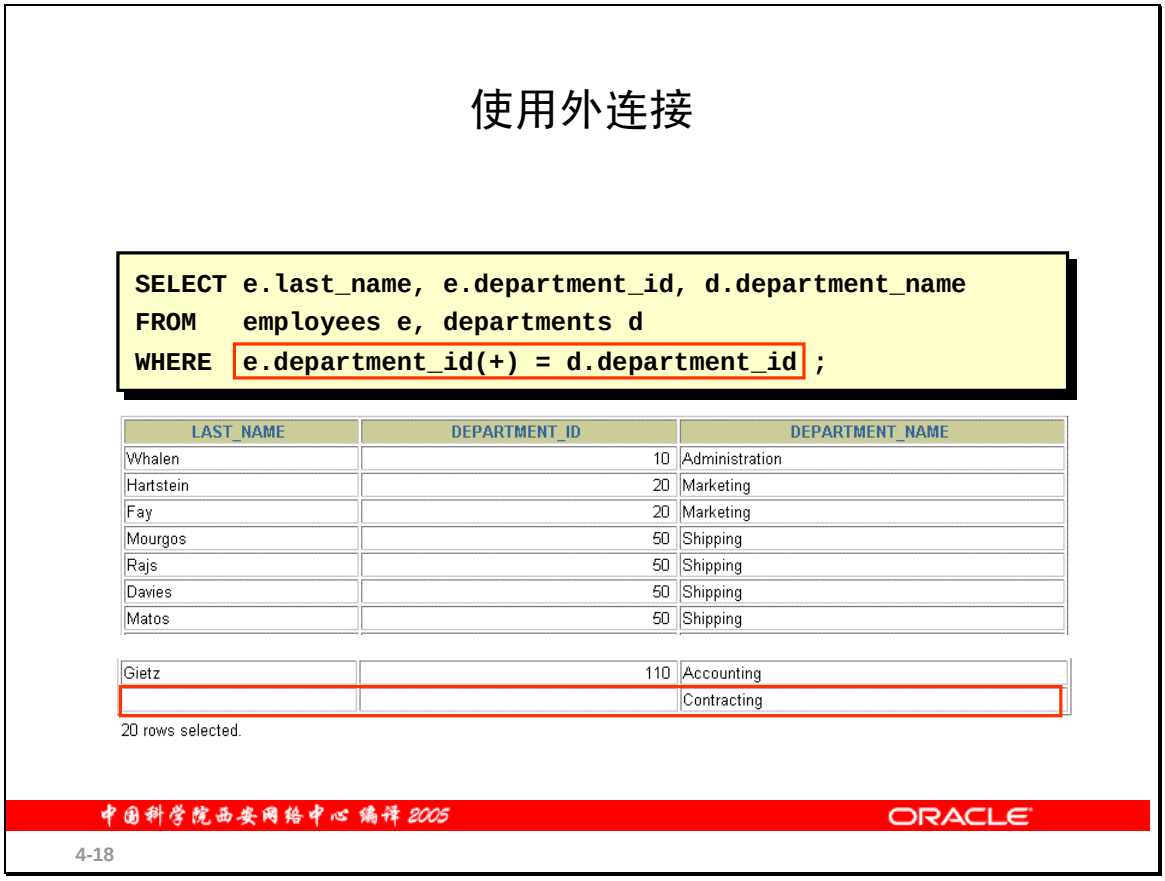
如果在连接条件中使用外连接操作，缺少的行就可以被返回。操作符是一个在圆括号中的加号 (+)，它被放置在连接的缺少信息的一侧。为了使来自不完善表的一行或多行能够被连接，该操作符有产生一个或多个空行的作用。

在语法中：

table1.column = 是连接表在一起的条件。

table2.column (+) 是外连接符号，它可以放在 WHERE 子句的条件的任一边，但不能两边都放。(跟着没有匹配行的表中列的名字放置外连接符号)。

幻灯片 18



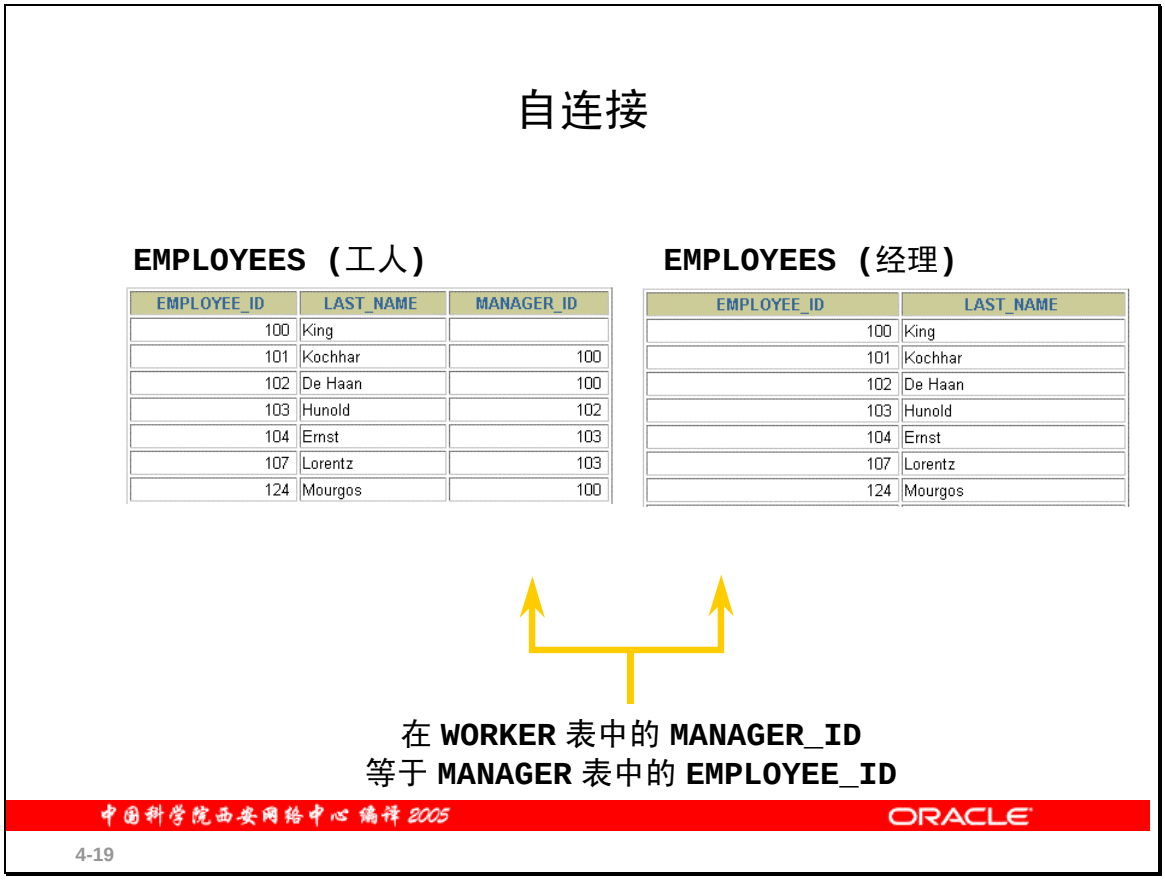
使用外连接返回不直接匹配的记录 (续)

幻灯片的例子显示雇员的 last name、department ID 和 department names。Contracting 部门还没有雇员。空值被显示在输出中。

外连接约束

- 外连接操作符只能出现在表达式一侧—缺少信息的一侧。它从一个表中返回那些在另一个表中没有直接匹配的行。
- 包括一个外连接的条件不能用 IN 操作符或连接到另一个用 OR 操作符的条件。

幻灯片 19



连接一个表到它自己

有时你需要连接一个表到它自己。为了找到每个雇员的经理的名字，你需要连接 EMPLOYEES 表到它自己，或执行一个自连接。例如，为了找到 Whalen 的经理的名字，你需要：

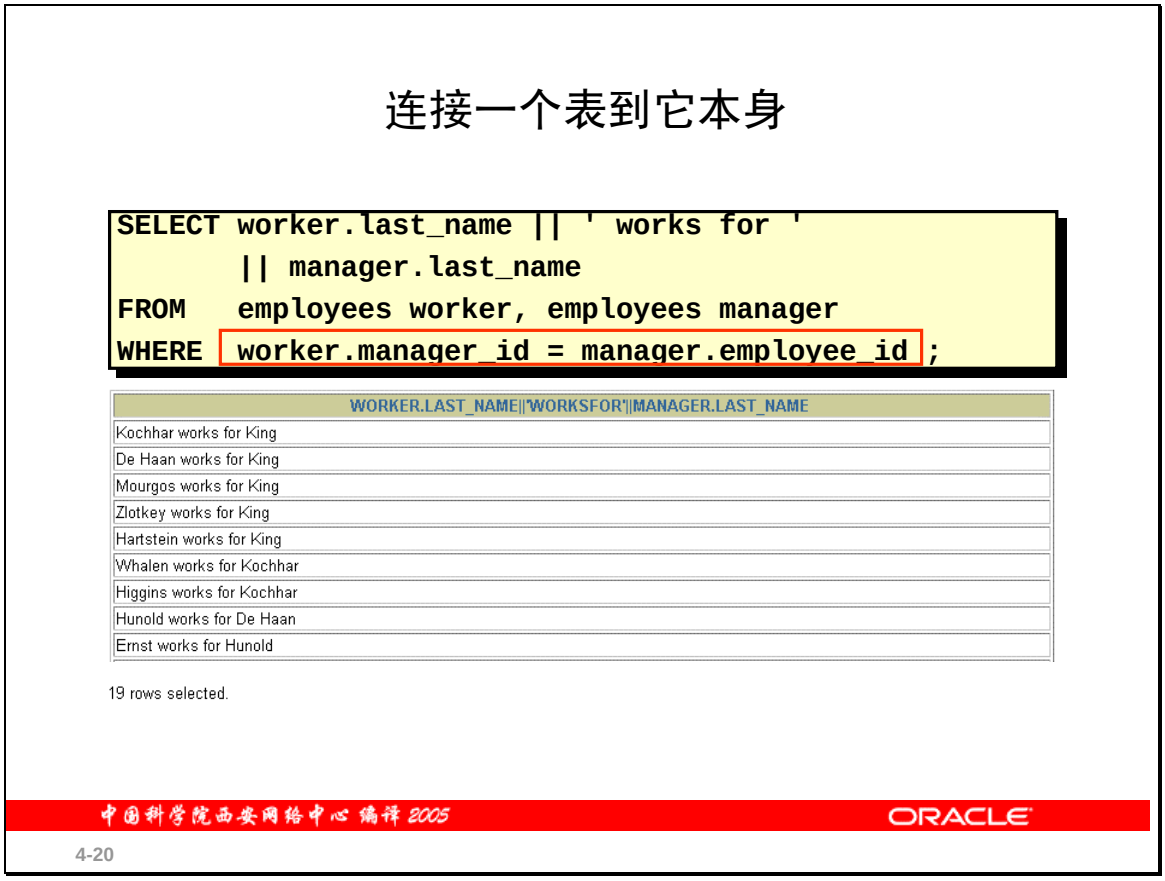
- 在 EMPLOYEES 的 LAST_NAME 列找到 Whalen。
- 在 MANAGER_ID 列找到 Whalen 的经理号。Whalen 的经理号是 101。
- 用 EMPLOYEE_ID 101 在 LAST_NAME 列找到经理的名字。Kochhar 的雇员号是 101，所以 Kochhar 是 Whalen 的经理。

在这个过程中，你要查找表两次，第一次你在 LAST_NAME 列中查找 Whalen 并且在找到对应的 MANAGER_ID 列的值 101。第二次你在 EMPLOYEE_ID 列查找 101 并且在 LAST_NAME 列找到 Kochhar。

教师注释

从 EMPLOYEES 表显示数据并且指出每个经理同时也是雇员。

幻灯片 20



连接一个表到它自己 (续)

幻灯片的例子连接 EMPLOYEES 表到它自己。为了在 FROM 子句中模拟两个表，对于相同的表 EMPLOYEES，用两个别名，分别为 w 和 m。

在该例中，WHERE 子句包含的连接意味着 “一个工人的经理号匹配该经理的雇员号”。

教师注释

给学生指出：

在幻灯片中，查询结果中的列标题似乎不重要。应该用一个有意义的列别名。

在输出中只有 19 行，但在 EMPLOYEES 表中有 20 行。产生这个结果是因为雇员 King，他是总经理，他上面没有经理。

幻灯片 21

练习 4, 第一部分: 概览

这部分练习包括用 Oracle 语法写将表连接在一起的查询

练习 4, 第一部分

该练习设计了多种到目前为止在课程中学过的 Oracle 的将连接表在一起的语法的习题。

完成本课后面的练习 1- 4。

幻灯片 22

用 SQL 连接表：1999 语法

用一个连接从多个表中查询数据

```
SELECT   table1.column, table2.column
FROM     table1
[CROSS JOIN table2] |
[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)];
```


定义连接

用 SQL: 1999 语法，你可以获得与用前面讲述的 Oracle 语法同样的结果。在语法中：

table1.column	表示要从其中返回数据的表和列
CROSS JOIN	从两个表中返回笛卡尔乘积。
NATURAL JOIN	基于相同的列名连接两个表
JOIN *table*	
USING *column_name*	执行一个基于列名的等值连接
JOIN *table ON*	
table1.column_name = *table2.column_name*	执行一个基于在 ON 子句中的条件的等值连接
LEFT/RIGHT/FULL OUTER	

更多信息，见 *Oracle9i SQL Reference*, “SELECT”。

幻灯片 23

创建交叉连接

- **CROSS JOIN** 子句导致两个表的交叉乘积
- 该连接和两个表之间的笛卡尔乘积是一样的

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

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

160 rows selected.

创建交叉连接

幻灯片中例子给出与下面语句相同的结果：

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

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Higgins	Contracting
Gietz	Contracting

160 rows selected.

幻灯片 24

创建自然连接

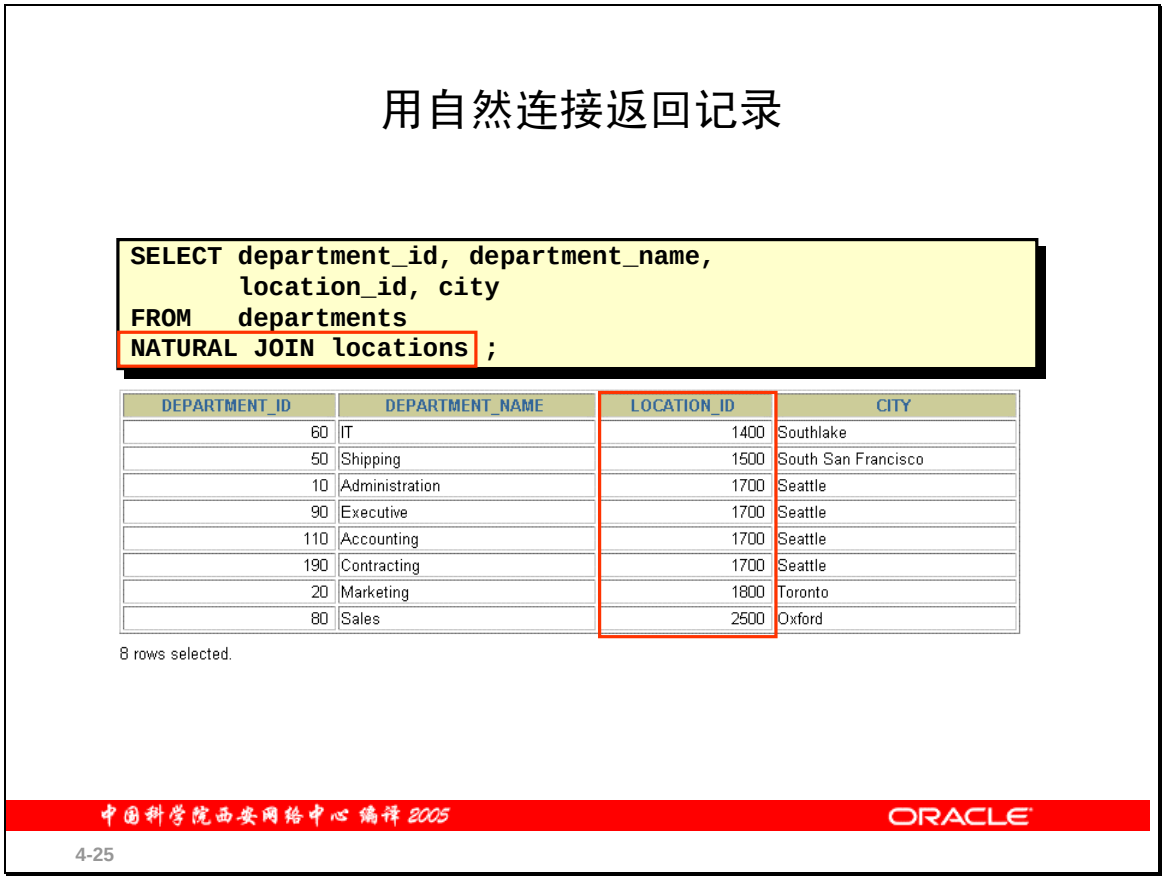
- **NATURAL JOIN** 子句基于两个表之间有相同名字的所有列
- 它从两个表中选择在所有的匹配列中有相等值的行
- 如果有相同名字的列的数据类型不同，返回一个错误

创建自然连接

在以前发布的 Oracle 中做一个在相应表中无明确指定的列的连接是不可能的。在 Oracle9*i* 中，让连接完全自动基于有匹配数据类型和名字的两个表中的列是可能的，使用 NATURAL JOIN 关键字。

注：连接只能发生在两个表中有相同名字和数据类型的列上。如果列有相同的名字，但数据类型不同，NATURAL JOIN 语法会引起错误。

幻灯片 25



用自然连接返回记录

在幻灯片的例子中，LOCATIONS 表被用 LOCATION_ID 列连接到 DEPARTMENT 表，这是在两个表中唯一名字相同的列。如果存在其它的公共列，连接会全部使用他们。

等值连接

自然连接也可以被写为等值连接：

```
SELECT    department_id, department_name,
          departments.location_id, city
FROM      departments, locations
WHERE     departments.location_id = locations.location_id;
```

带 WHERE 子句的自然连接

可以用 WHERE 子句实现在一个自然连接中添加约束。下面的例子限制部门号 department ID 等于 20 或 50 那些输出的行。

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

幻灯片 26

用 USING 子句创建连接

- 如果一些列有相同的名字，但数据类型不匹配，**NATURAL JOIN** 子句能够用 USING 子句修改以指定将被用于一个等值连接的列
- 当有多个列匹配时，用 USING 子句匹配唯一的列
- 在引用列不要使用表名或者别名
- NATURAL JOIN 和 **USING** 子句是相互排斥的

USING 子句

自然连接 (Natural joins) 用具有相匹配的名字和数据类型的所有列来连接表。USING 子句可以被用来指定那些将被用语一个等值连接的列中的唯一列。在 USING 子句中引用的列不应该在 SQL 语句的任何地方用表名或表别名限制 (前缀)。

例如，该语句是有效的：

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

该语句是无效的，因为 LOCATION_ID 在 WHERE 子句中被限制了：

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE d.location_id = 1400;
ORA-25154: column part of USING clause cannot have qualifier
```

同样的限制也用于 NATURAL 连接。因此，那些在两个表中有相同名字的列不能没有任何限定词。

幻灯片 27

用 USING 子句返回记录

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

EMPLOYEE_ID	LAST_NAME	LOCATION_ID
200	Whalen	1700
201	Hartstein	1800
202	Fay	1800
124	Mourgos	1500
141	Rajs	1500
142	Davies	1500
143	Matos	1500
144	Vargas	1500
103	Hunold	1400

19 rows selected.

USING 子句 (续)

该例子示范连接 EMPLOYEES 和 DEPARTMENTS 表中的 DEPARTMENT_ID 列，并以此显示雇员工作的场所编号。

该例子也写成一个等值连接：

```
SELECT employee_id, last_name,
       employees.department_id, location_id
FROM   employees, departments
WHERE  employees.department_id = departments.department_id;
```

幻灯片 28

用 ON 子句创建连接

- 对于自然连接的连接条件，基本上是带有相同名字的所有列的等值连接
- 为了指定任意条件，或者指定要连接的列，可以使用 ON 子句
- 连接条件从另一个搜索条件中被分开
- ON 子句使得代码易懂

ON 条件

用 ON 子句指定一个连接条件。这让你从在 WHERE 子句中的查找或过滤条件中分离指定的连接条件。

幻灯片 29

用 ON 子句返回记录

```
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	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

19 rows selected.

创建带 ON 子句的连接

ON 子句也可以象下面一样被用于有不同名字的连接列：

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

EMP	MGR
Kochhar	King
De Haan	King
Mourgos	King
Zlotkey	King
Gietz	Higgins

19 rows selected.

前面的例子是 EMPLOYEE 表基于 EMPLOYEE_ID 和 MANAGER_ID 列的到它自己的自连接。

幻灯片 30

用 ON 子句创建三向连接

```
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
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

19 rows selected.

三向连接

三向连接是三个表的连接。在 SQL: 1999 兼容语法中，连接被从左到右执行，所以第一个连接执行 EMPLOYEES JOIN DEPARTMENTS。第一个连接条件可以引用在 EMPLOYEES 和 DEPARTMENTS 中的列，但不能引用在 LOCATIONS 中的列。第二个条件可以引用所有三个表中的列。

这也可以被写为三个等值连接：

```
SELECT employee_id, city, department_name
FROM   employees, departments, locations
WHERE  employees.department_id = departments.department_id
AND    departments.location_id = locations.location_id;
```

教师注释

下面的例子显示也可以用 USING 子句完成同样的连接：

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

幻灯片 31

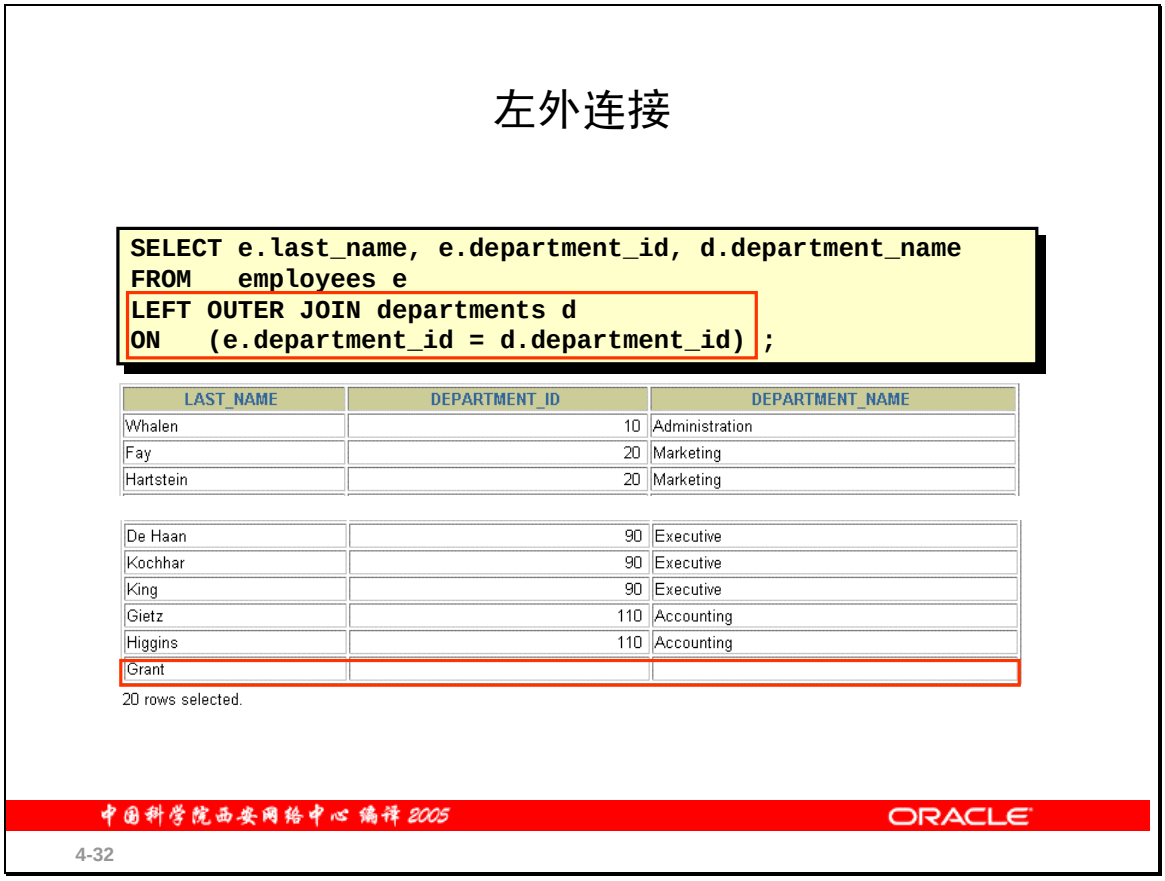
内与外连接

- 在 SQL: 1999 中，连接两个表，仅返回匹配的行的连接，称为内连接
- 在两个表之间的连接，返回内连接的结果，同时还返回不匹配行的左（或右）表的连接，称为左（或右）连接
- 在两个表之间的连接，返回内连接的结果， 同时还返回左和右连接，称为全连接

连接 – 比较 SQL: 1999 和 Oracle 语法

Oracle	SQL: 1999
Equijoin	Natural or Inner Join
Outerjoin	Left Outer Join
Selfjoin	Join `ON`
Nonequijoin	Join `USING`
Cartesian	Product Cross Join

幻灯片 32



左外连接的例子

左边的表 (EMPLOYEES) 中即使没有与 DEPARTMENTS 表中匹配的行，该查询也会取回 EMPLOYEES 表中所有的行。

该查询可以用如下的更容易的语句完成：

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  d.department_id (+) = e.department_id;
```

幻灯片 33

右外连接

```
SELECT e.last_name, e.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
King	90	Executive
Kochhar	90	Executive
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Higgins	110	Accounting
Gietz	110	Accounting
		Contracting

20 rows selected.

右外连接

右边的表 (DEPARTMENTS) 中即使没有与 EMPLOYEES 表中匹配的行，该查询也会取回 DEPARTMENTS 表中所有的行。

该查询可以用如下的更容易的语句完成：

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  d.department_id = e.department_id  (+);
```

幻灯片 34

全外连接

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

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
		Contracting

21 rows selected.

全外连接的例子

该查询取回 EMPLOYEES 表中所有的行，即使在 DEPARTMENTS 表中没有相匹配的行。它也取回 DEPARTMENTS 表中所有的行，即使 EMPLOYEES 表中没有相匹配的行。

教师注释

可以更容易的完成全外连接，你可以用 UNION 操作符得到相同的结果。

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id (+) = d.department_id
UNION
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id = d.department_id (+);
```

幻灯片 35

附加条件

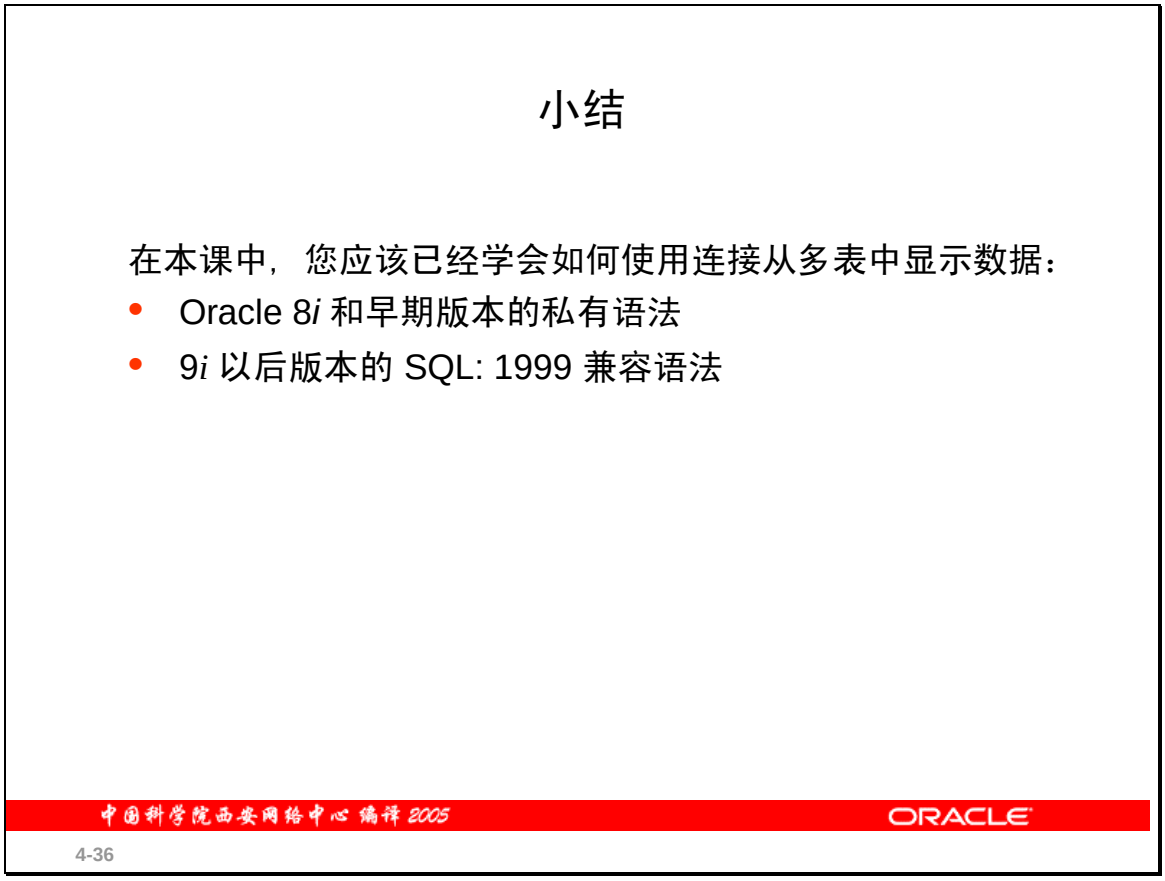
```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

应用附加条件

你可以在 WHERE 子句中应用附加条件。该例子显示在 EMPLOYEES 和 DEPARTMENTS 表上执行一个连接，并且附加条件，只显示经理 ID 等于 149 的雇员。

幻灯片 36



小结

有多种方法连接表。

连接类型

- 等值 Equijoins
- 非等值 Non-equijoins
- 外连接 Outer joins
- 自连接 Self joins
- 交叉连接 Cross joins
- 自然连接 Natural joins
- 全外连接 Full or outer joins

笛卡尔乘积

一个笛卡尔乘积导致所有行的组合被显示。其原因可能是忽略了 WHERE 子句或指定了 `CROSS JOIN` 子句。

表别名

- 使用表别名加速数据库的访问。
- 表别名有助于保持 SQL 代码较小，并节省存储器。

幻灯片 37

练习 4, 第二部分 Part Two: 概览

本练习包括下面的主题：

- 用等值连接来连接表
- 演示外连接和自连接
- 附加条件

练习 4, 第二部分

该练习有意给你从多表中提取数据的实际的经验。试用 Oracle 专用语法和 SQL: 1999 兼容语法。

在第二部分中，问题 5-8，试写出使用 ANSI 语法的连接语句。

在第二部分中，问题 9-11，用 Oracle 语法和 ANSI 语法试写出连接语句。

练习 4 – 第一部分

1. 写一个查询显示所有雇员的 last name、department number、and department name。

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
Vargas	50	Shipping
Hunold	60	IT
Higgins	110	Accounting
Gietz	110	Accounting

19 rows selected.

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id;
```

2. 创建一个在部门 80 中的所有工作岗位的唯一列表，在输出中包括部门的地点。

JOB_ID	LOCATION_ID
SA_MAN	2500
SA_REP	2500

```
SELECT DISTINCT job_id, location_id
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND employees.department_id = 80;
```

3. 写一个查询显示所有有佣金的雇员的 last name、department name、location ID 和城市。

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Zlotkey	Sales	2500	Oxford
Abel	Sales	2500	Oxford
Taylor	Sales	2500	Oxford

```
SELECT e.last_name, d.department_name, d.location_id, l.c ity
FROM employees e, departments d, locations l
```

```
WHERE e.department_id = d.department_id
AND
d.location_id = l.location_id
AND e.commission_pct IS NOT NULL;
```

4. 显示所有在其 last names 中有一个小写 *a* 的雇员的 last name 和 department name。请将你的 SQL 语句用文件名 lab4_4.sql 存为脚本文件。

LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing
Fay	Marketing
Rajs	Shipping
Davies	Shipping
Matos	Shipping
Vargas	Shipping
Taylor	Sales
Kochhar	Executive
De Haan	Executive

10 rows selected.

```
SELECT last_name, department_name
FROM employees, departments
WHERE employees.department_id = departments.department_id
AND last_name LIKE '%a%';
```

练习 4 – 第二部分

5. 写一个查询显示那些工作在 Toronto 的所有雇员的 last name、job、department number 和 department name。

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

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

6. 显示雇员的 last name 和 employee number 连同他们的经理的 last name 和

manager number。列标签分别为 Employee、Emp#、Manager 和 Mgr#。将你的 SQL 语句存放在名为 lab4_6.sql 的文本文件中。

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100
Abel	174	Zlotkey	149
Taylor	176	Zlotkey	149
Grant	178	Zlotkey	149
Fay	202	Hartstein	201
Gietz	206	Higgins	205

19 rows selected.

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

7. 修改 lab4_6.sql 显示所有雇员包括 King，他没有经理。用雇员号排序结果。将你的 SQL 语句存放在名为 lab4_7.sql 的文本文件中。运行 lab4_7.sql 中的查询。

Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100
Rajs	141	Mourgos	124
Higgins	205	Kochhar	101
Gietz	206	Higgins	205

20 rows selected.

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

如果你有时间，完成下面的习题：

8. 创建一个查询显示所有与被指定雇员工作在同一部门的雇员 (同事) 的 last names、department numbers。给每列一个适当的标签。

DEPARTMENT	EMPLOYEE	COLLEAGUE
20	Fay	Hartstein
20	Hartstein	Fay
50	Davies	Matos
50	Davies	Mourgos
50	Davies	Rajs
50	Davies	Vargas
50	Matos	Davies
50	Matos	Mourgos
50	Matos	Rajs
50	Matos	Vargas
50	Mourgos	Davies
110	Gietz	Higgins
110	Higgins	Gietz

42 rows selected.

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

9. 显示 JOB_GRADES 表的结构。创建一个查询显示所有雇员的 name、job、department name、salary 和 grade。

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRA
Matos	ST_CLERK	Shipping	2600	A
Vargas	ST_CLERK	Shipping	2500	A
Lorentz	IT_PROG	IT	4200	B
Mourgos	ST_MAN	Shipping	5800	B
Rajs	ST_CLERK	Shipping	3500	B
Davies	ST_CLERK	Shipping	3100	B
Whalen	AD_ASST	Administration	4400	B
Hunold	IT_PROG	IT	9000	C

De Haan	AD_VP	Executive	17000	E

19 rows selected.

```
   DESC JOB_GRADES

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

如果你想要接受额外的挑战，完成下面的习题：

10. 创建一个查询显示那些在雇员 Davies 之后入本公司工作的雇员的 name 和 hire date。

LAST_NAME	HIRE_DATE
Lorentz	07-FEB-99
Mourgos	16-NOV-99
Matos	15-MAR-98
Vargas	09-JUL-98
Zlotkey	29-JAN-00
Taylor	24-MAR-98
Grant	24-MAY-99
Fay	17-AUG-97

8 rows selected.

```
   SELECT e.last_name, e.hire_date
   FROM employees e, employees davies
   WHERE davies.last_name = 'Davies'
   AND davies.hire_date < e.hire_date
-- OR
   SELECT e.last_name, e.hire_date
   FROM employees e JOIN employees davies
   ON (davies.last_name = 'Davies')
   WHERE davies.hire_date < e.hire_date;
```

11. 显示所有雇员的 names 和 hire dates，他们在他们的经理之前进入本公司，连同他们的经理的名字和受雇日期一起显示。列标签分别为 Employee、Emp Hired、Manager 和 Mgr Hired。

Employee	Emp Hired	Manager	Mgr Hired
Whalen	17-SEP-87	Kochhar	21-SEP-89
Hunold	03-JAN-90	De Haan	13-JAN-93
Rajs	17-OCT-95	Mourgos	16-NOV-99
Davies	29-JAN-97	Mourgos	16-NOV-99
Matos	15-MAR-98	Mourgos	16-NOV-99
Vargas	09-JUL-98	Mourgos	16-NOV-99
Abel	11-MAY-96	Zlotkey	29-JAN-00
Taylor	24-MAR-98	Zlotkey	29-JAN-00
Grant	24-MAY-99	Zlotkey	29-JAN-00

9 rows selected.

```
   SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
   FROM employees w, employees m
   WHERE w.manager_id = m.employee_id
   AND w.hire_date < m.hire_date;
-- OR
   SELECT w.last_name, w.hire_date, m.last_name, m.hire_date
   FROM employees w JOIN employees m
   ON (w.manager_id = m.employee_id)
   WHERE w.hire_date < m.hire_date;
```