

幻灯片 1

# 15

## 使用集合运算

中国科学院西安网络中心 编译 2005

ORACLE

Copyright © Oracle Corporation, 2001. All rights reserved.

进度表:	时间	主题
	30 分钟	讲演
	20 分钟	练习
	50 分钟	总共

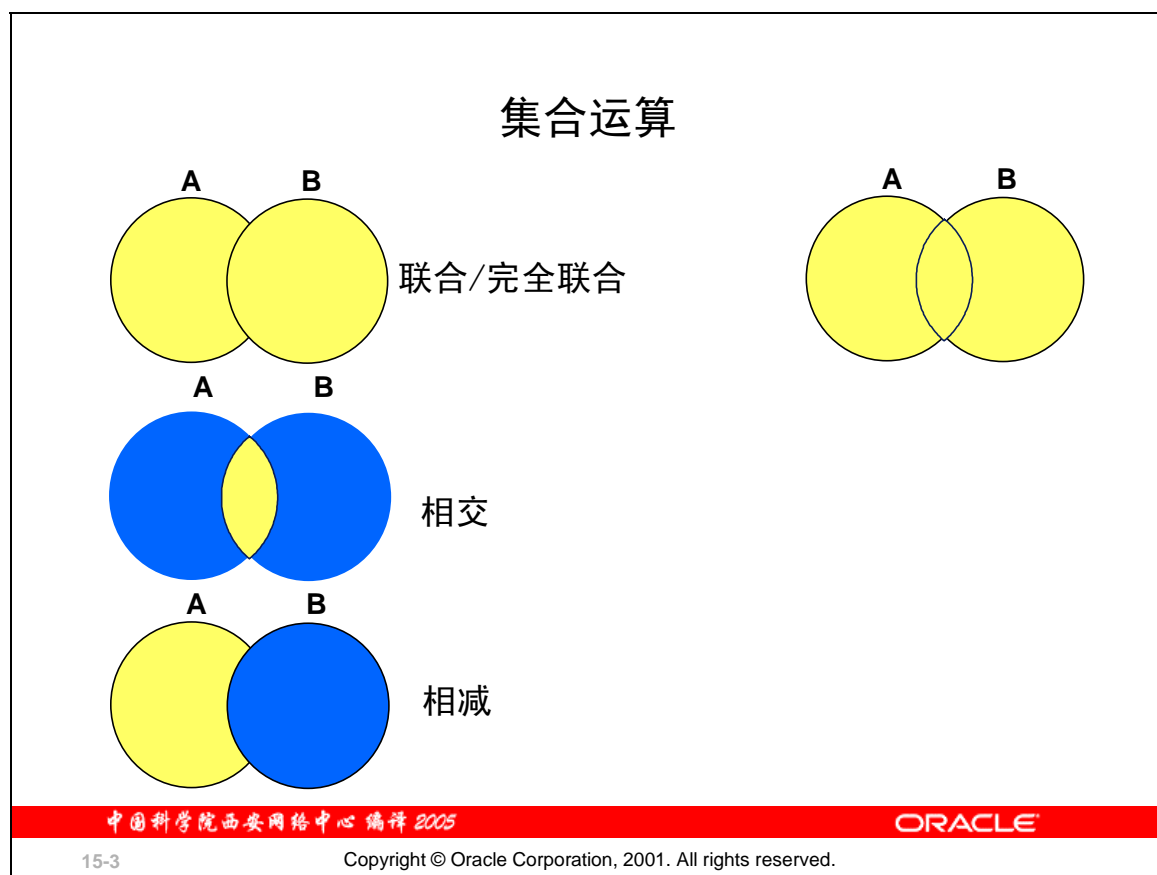
## 目标

完成本课后，您应当能够执行下列运算：

- 描述集合运算
- 用集合运算组合多个查询到一个单个的查询中
- 控制行返回的顺序

### 课程目标

在本课中，你将学习怎样用集合写查询。



### 集合运算

集合运算组合两个或多个部分查询的结果到一个结果中。包含集合运算的查询称为**复合查询**。

Operator	Returns
UNION	由每个查询选择的所有不同的行
UNION ALL	由每个查询选择的所有的行，包括所有重复的行
INTERSECT	由两个查询选择的所有不同的行
MINUS	由第一个查询选择的所有不同的行

所有的集合运算与等号的优先级相同，如果 SQL 语句包含多个集合运算并且没有圆括号明确地指定另一个顺序，Oracle 服务器将以从左到右的顺序计算。你应该使用圆括号来明确地指定带另外的集合运算的 INTERSECT (相交) 运算查询中的赋值顺序。

**注：**在幻灯片中，图中的亮色代表查询结果。

### 教师注释

INTERSECT (相交) 和 MINUS (相减) 运算不是 ANSI SQL-99 兼容的，他们是 Oracle 特定的。

## 本课所使用的表

在下面课程中要用到的表：

- **EMPLOYEES:** 提供所有在职雇员当前的详细资料
- **JOB\_HISTORY:** 当一个雇员改变工作时，记录他的以前的工作的开始日期和结束日期、department ID 和 job ID 的详细资料

### 本课所使用的表

本课要用到两个表，即 EMPLOYEES 表和 JOB\_HISTORY 表。

EMPLOYEES 表存储雇员的详细信息；对于人力资源记录，该表为每个用户存储一个唯一标识的数和 Email 地址，雇员工作的详细信息，标识号、薪水和经理也被存储。一些雇员在他们的薪水之外还有佣金；该信息也被记录下来。公司以工作中雇员的角色来组织，一些雇员来公司很长时间了，并且换过不同的工作，雇员改变工作的情况被记录在 JOB\_HISTORY 表中。当一个雇员改变工作时，以前工作的开始日期和结束日期、工作的标识号和部门等详细信息被记录在 JOB\_HISTORY 表中。

EMPLOYEES 表和 JOB\_HISTORY 表的结构和数据在下一页显示。

这儿有一些公司人员的实例，他们在公司期间曾有过不同的职位。例如，雇员 Taylor，24-MAR-1998 进入公司，24-MAR-98 到 31-DEC-98 期间他的工作是 SA\_REP，01-JAN-99 到 31-DEC-99 期间他的职位是 SA\_MAN，现在，Taylor 又回到 SA\_REP 职位。

类似地考虑雇员 Whalen，他 17-SEP-1987 进入公司，17-SEP-87 到 17-JUN-93 期间他的工作职位是 AD\_ASST，01-JUL-94 到 31-DEC-98 期间他的工作职别是 AC\_ACCOUNT，现在，Whalen 又回到 AD\_ASST 职位。

幻灯片 5

### 本课所使用的表 (续)

DESC employees

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

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	King	AD_PRES	17-JUN-87	90
101	Kochhar	AD_VP	21-SEP-89	90
102	De Haan	AD_VP	13-JAN-93	90
103	Hunold	IT_PROG	03-JAN-90	60
104	Ernst	IT_PROG	21-MAY-91	60
107	Lorentz	IT_PROG	07-FEB-99	60
124	Mourgos	ST_MAN	16-NOV-99	50
141	Rajs	ST_CLERK	17-OCT-95	50
142	Davies	ST_CLERK	29-JAN-97	50
143	Matos	ST_CLERK	15-MAR-98	50
144	Vargas	ST_CLERK	09-JUL-98	50
149	Zlotkey	SA_MAN	29-JAN-00	80
174	Abel	SA_REP	11-MAY-96	80
176	Taylor	SA_REP	24-MAR-98	80
178	Grant	SA_REP	24-MAY-99	
200	Whalen	AD_ASST	17-SEP-87	10
201	Hartstein	MK_MAN	17-FEB-96	20
202	Fay	MK_REP	17-AUG-97	20
205	Higgins	AC_MGR	07-JUN-94	110
206	Gietz	AC_ACCOUNT	07-JUN-94	110

20 rows selected.

幻灯片 6

本课所使用的表 (续)

DESC job\_history

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

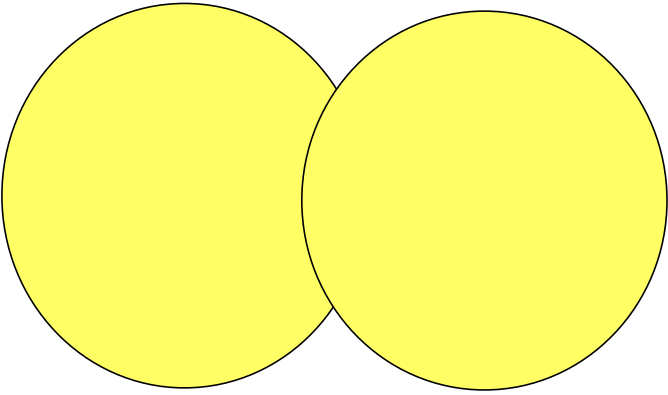
SELECT \* FROM job\_history;

EMPLOYEE_ID	START_DAT	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

**联合运算**

A                      B



联合 (UNION) 运算从两个查询返回除去重复值后的结果

中国科学院西安网络中心 编译 2005 ORACLE

15-7 Copyright © Oracle Corporation, 2001. All rights reserved.

### UNION (联合) 运算

UNION 运算返回所有由任一查询选择的行。用 UNION 运算从多表返回所有行，但除去任何重复的行。

#### 原则

- 被选择的列数和列的数据类型必须是与所有用在查询中的 SELECT 语句一致。列的名字不必相同。
- 联合运算在所有被选择的列上进行。
- 在做重复检查的时候不忽略空(NULL)值。
- IN 运算有比 UNION 运算高的优先级。
- 在默认情况下，输出以 SELECT 子句的第一列的升序排序。

#### 教师注释

为了举例说明联合集合(UNION SET)运算，运行脚本 demo\15\_union1.sql。指出输出以 SELECT 子句的第一列的升序排序。

## 使用联合集合运算

显示当前和以前所有雇员的工作岗位。每个雇员仅显示一次

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

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AC_ACCOUNT
...	
200	AC_ACCOUNT
200	AD_ASST
...	
205	AC_MGR
206	AC_ACCOUNT

## 使用联合集合 (UNION SET) 运算

联合运算消除重复记录，如果有相同的记录同时出现在 EMPLOYEES 和 JOB\_HISTORY 表中，该记录只显示一次，观察幻灯片的输出显示雇员号为 200 的雇员的雇员号虽然显示了两行，但每一行的 JOB\_ID 是不同的。

考虑下面的例子：

```
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
100	AD_PRES	90
101	AC_ACCOUNT	110
101	AC_MGR	110
101	AD_VP	90
102	AD_VP	90
200	AC_ACCOUNT	90
200	AD_ASST	10
200	AD_ASST	90
201	MK_MAN	20
201	MK_REP	20

29 rows selected.

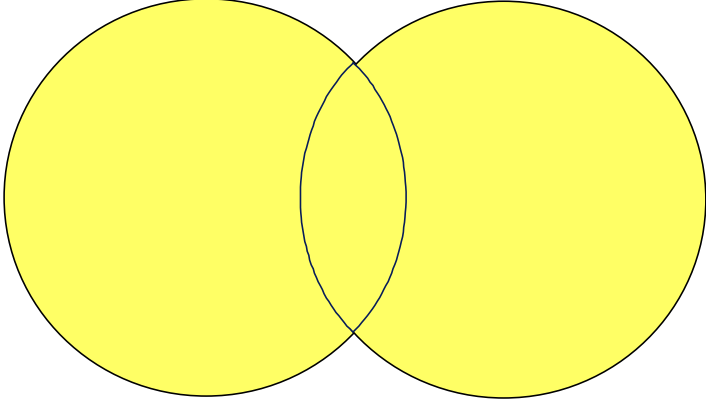


### 幻灯片 9

#### 使用联合集合(UNION SET)运算 (续)

在前面的输出中，雇员 200 出现了 3 次，为什么？注意雇员 200 的 DEPARTMENT\_ID 值，一行是 90，另一行是 10，第三行是 90，因为这些 job\_id 和 department\_id 的唯一组合，对于每行的雇员 200 是唯一的，因此他们是不重复的。观察输出以 SELECT 子句的第一列的升序排序，即以 EMPLOYEE\_ID 排序。

**全联合运算**



全联合 (**UNION ALL**) 运算从两个查询返回包括所有重复值的结果

中国科学院西安网络中心 编译 2005 ORACLE

15-10 Copyright © Oracle Corporation, 2001. All rights reserved.

### 全联合(UNION ALL)运算

用全联合运算从多个查询中返回所有行。

#### 原则

- 和联合不同，重复的行不被过滤，并且默认情况下输出不排序。
- 不能使用 DISTINCT 关键字。

**注：**除了上面的两点，UNION ALL 的原则与 UNION 相同。

## 使用全联合运算

显示当前和以前所有雇员所在的部门

```

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
100	AD_PRES	90
101	AD_VP	90
...		
200	AD_ASST	10
200	AD_ASST	90
200	AC_ACCOUNT	90
...		
205	AC_MGR	110
206	AC_ACCOUNT	110

30 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

15-11

Copyright © Oracle Corporation, 2001. All rights reserved.

## 全联合(UNION ALL)运算 (续)

在例子中，有 30 行被选择，两个表组合到共 30 行中，全联合运算不会消除重复的行，在幻灯片中重复的行被突出显示，联合返回任一查询所选择的所有不重复的行，而全联合返回任一查询所选择的所有行，包括所有重复。考虑幻灯片中的查询，现在用联合子句来写：

```

SELECT   employee_id, job_id, department_id
FROM     employees
UNION
SELECT   employee_id, job_id, department_id
FROM     job_history
ORDER BY employee_id;

```

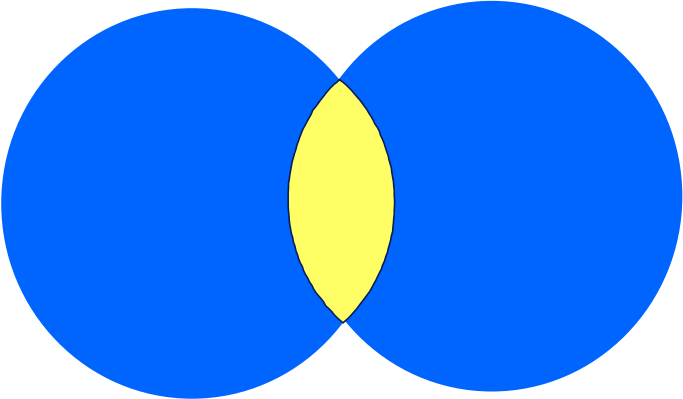
上面的查询返回 29 行，这是因为下面的行被除去了(因为它是一个重复行)：

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

## 教师注释

注意，该例子来自 15-8 页。

## 相交运算



相交 (**INTERSECT**) 运算返回多个查询中所有相同的行

中国科学院西安网络中心 编译 2005

ORACLE

15-12

Copyright © Oracle Corporation, 2001. All rights reserved.

### 相交运算

用相交运算返回多个查询中所有的公共行。

#### 原则

- 在查询中被 **SELECT** 语句选择的列数和数据类型必须与在查询中所使用的所有的 **SELECT** 语句中的一样，但列的名字不必一样。
- 相交的表的顺序排序不改变结果。
- 相交不忽略空值。

#### 教师注释

为了举例说明相交集合(**INTERSECT SET**)运算，运行脚本 `demo\15_inters.sql`。

## 使用相交运算

显示雇员表的 employee\_ID 和 job\_ID，这些雇员当前所做的工作是以前他们做过一端时间，后来有变化，现在又在做的工作。

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

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

中国科学院西安网络中心 编译 2005

ORACLE

15-13

Copyright © Oracle Corporation, 2001. All rights reserved.

## 相交运算 (续)

在幻灯片的例子中，查询仅返回在两个表的被选择的列中有相同值的记录。

如果你从 EMPLOYEES 表添加 DEPARTMENT\_ID 列到 SELECT 语句中，并且从 JOB\_HISTORY 表添加 DEPARTMENT\_ID 列到 SELECT 语句中，然后运行该查询，将返回什么？其结果可能是不同的，因为在加入的另一个列中，其值可能是重复的，也可能不重复。

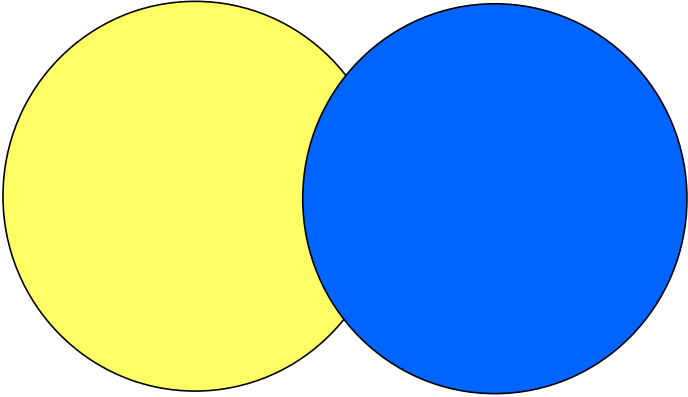
## 例子

```
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
176	SA_REP	80

雇员 200 不再是结果的一部分，因为 EMPLOYEES.DEPARTMENT\_ID 值不同于 JOB\_HISTORY.DEPARTMENT\_ID 值。

**相减运算**



相减 (**MINUS**) 运算返回在第一个查询中而不在第二个查询中的行

中国科学院西安网络中心 编译 2005 ORACLE

15-14 Copyright © Oracle Corporation, 2001. All rights reserved.

### 相减运算

用相减运算返回由第一个查询返回的行，那些行不出现在第二个查询中（第一个 SELECT 语句减第二个 SELECT 语句）。

### 原则

- 在查询中被 SELECT 语句选择的列数和数据类型必须与在查询中所使用的所有的 SELECT 语句中的一样，但列的名字不必一样。
- 对于 MINUS 运算，在 WHERE 子句中所有的列都必须在 SELECT 子句中。

### 教师注释

为了举例说明相减运算，运行脚本 demo\15\_minus.sql。

## 相减运算

显示那些从来没有改变过他们的工作的雇员 ID

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

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_VP
103	IT_PROG
...	
201	MK_MAN
202	MK_REP
205	AC_MGR
206	AC_ACCOUNT

18 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

15-15

Copyright © Oracle Corporation, 2001. All rights reserved.

### 相减运算 (续)

在幻灯片的例子中，JOB\_HISTORY 表的 employee\_id 列和 job\_id 列被从 EMPLOYEES 表的那些列中减去。结果集显示相减后剩余的雇员，他们由存在于 EMPLOYEES 表中但不存在于 JOB\_HISTORY 表中的行所表示。这些行是那些从未改变过工作的雇员的记录。

雇员 200 未出现，因为他换过工作。

## 集合运算的原则

- 在两个 **SELECT** 列表中的表达式必须在数目上和数据类型上相匹配
- 可以用圆括号改变执行的顺序
- **ORDER BY** 子句：
  - 只能出现在语句的最后
  - 从第一个 **SELECT** 语句接收列名、别名，或者位置记号

### 集合运算的原则

- 在两个查询的选择列表中的表达式在数目上和数据类型上必须匹配。使用 **UNION**、**UNION ALL**、**INTERSECT** 和 **MINUS** 运算的查询，在它们的 **WHERE** 子句中必须有与它们的 **SELECT** 列表相同的列数和列数据类型。例如：

```
SELECT employee_id, department_id
FROM   employees
WHERE  (employee_id, department_id)
       IN (SELECT employee_id, department_id
           FROM   employees
           UNION
           SELECT employee_id, department_id
           FROM   job_history);
```
- **ORDER BY** 子句：
  - 只可以出现在每个语句的末尾
  - 将接受列、别名或位置记号
- 列名或别名，如果用在 **ORDER BY** 子句中，必须来自第一个 **SELECT** 列表。
- 集合运算可以用在子查询中。

### 教师注释

你可以说，如果来自两个查询中的该列有相同的名字，**ORDER BY** 子句只接受列名。



## Oracle 服务器和集合运算

- 除了 **UNION ALL**，重复行自动被清除
- 在结果中的列名是第一个查询中出现的列名
- 除了 **UNION ALL**，默认情况下按升序顺序输出

中国科学院西安网络中心 编译 2005

ORACLE

15-17

Copyright © Oracle Corporation, 2001. All rights reserved.

### Oracle 服务器和集合运算

当一个查询使用集合运算时，除了 **UNION ALL** 运算，Oracle 服务器会自动消除重复的行。输出中的列名由第一个 **SELECT** 语句的列表确定。默认情况下，输出以 **SELECT** 子句的第一列的升序排序。

在一个复合查询的各查询组成部分的选择列表中相应的表达式必须在数目和类型上匹配。如果查询的组成部分选择字符数据，返回值的数据类型被如下决定：

- 如果查询选择的数据类型的值为 **CHAR**，那么，返回值的数据类型也为 **CHAR**。
- 如果查询选择的两者之一或两者的数据类型值为 **VARCHAR2**，那么，返回值的数据类型也是 **VARCHAR2**。

### 教师注释

你可能需要提及，输出以第一个 **SELECT** 子句的第一列的升序被排序，然后是第二列，等等。

## 匹配 SELECT 语句

使用 UNION 运算，显示所有雇员的 department ID、location、和受雇日期

```
SELECT department_id, TO_NUMBER(null)
       location, hire_date
FROM   employees
UNION
SELECT department_id, location_id,  TO_DATE(null)
FROM   departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96
...		
110	1700	
110		07-JUN-94
190	1700	
		24-MAY-99

27 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

15-18

Copyright © Oracle Corporation, 2001. All rights reserved.

## 匹配 SELECT 语句

由于在两个查询的 SELECT 列表中的表达式必须在数量上匹配，你可以使用虚拟列和转换函数数据类型来满足该规则。在幻灯片中使用了虚拟列，在第一个查询中的 TO\_NUMBER 函数被用以匹配第二个查询中返回的 LOCATION\_ID 列的数字数据类型，同样地，第二个查询中的 TO\_DATE 函数被用于匹配 第一个查询返回的日期数据类型。

## 教师注释

示范：demo\15\_union3.sql, demo\15\_dummy.sql

目的：示范 15\_union3.sql 举例说明使用转换函数同时匹配在两个选择列表中的列。示范 15\_dummy.sql 为了匹配选择列表使用了虚拟列。对于 15\_dummy.sql，运行脚本，然后，反注释 REMARKS，添加 ORDER BY 2，再运行。

你可能需要说明，如果在幻灯片的代码中的转换函数不被强制，即使没有转换函数代码也会工作的很好，但为了性能更好建议显式地转换值。

## 匹配 SELECT 语句

- 使用 UNION 运算，显示所有雇员的 employee ID、job ID 和 salary

```
SELECT employee_id, job_id, salary
FROM   employees
UNION
SELECT employee_id, job_id, 0
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
...		
205	AC_MGR	12000
206	AC_ACCOUNT	8300

30 rows selected.

中国科学院西安网络中心 编译 2005

ORACLE

15-19

Copyright © Oracle Corporation, 2001. All rights reserved.

## 匹配 SELECT 语句：例

EMPLOYEES 和 JOB\_HISTORY 表中有一些列是公有的；例如，EMPLOYEE\_ID, JOB\_ID 和 DEPARTMENT\_ID，但如果你想要用 UNION 运算查询他们以显示 EMPLOYEE\_ID、JOB\_ID 和 SALARY 时，知道薪水仅存在于 EMPLOYEES 表中吗？

在幻灯片中的代码例子匹配 EMPLOYEES 表和 JOB\_HISTORY 表中的 EMPLOYEE\_ID 和 JOB\_ID 列。一个 0 值被添加到 JOB\_HISTORY 的 SELECT 语句中以匹配在 EMPLOYEES 的 SELECT 语句中的数字的 SALARY 列。

在前面的结果中，在输出中的每一行相对应于 JOB\_HISTORY 表的一条记录，包括 SALARY 列值为 0 的列。

## 控制行顺序

用两个 UNION 运算产生一个英语句子

```
COLUMN a_dummy NOPRINT
SELECT 'sing' AS "My dream", 3 a_dummy
FROM dual
UNION
SELECT 'I'd like to teach', 1
FROM dual
UNION
SELECT 'the world to', 2
FROM dual
ORDER BY 2;
```

My dream
I'd like to teach
the world to
sing

### 控制行顺序

在默认情况下，输出以第一列的升序排序，你可以用 ORDER BY 子句来改变顺序。

#### 使用 ORDER BY 来改变行的顺序

ORDER BY 子句在复合查询中只能用一次。如果使用了，ORDER BY 子句必须放在最后面的查询中。ORDER BY 子句接受列名、别名或位置记号。如果没有 ORDER BY 子句，幻灯片中的代码例子将按第一列的字母顺序产生下面的输出：

My dream
I'd like to teach
sing
the world to

**注：**考虑一个多次使用 UNION SET 运算的复合查询，在这种情况下，ORDER BY 子句中只能用位置而不能显式表示。在幻灯片的例子中 3 个 SELECT 列表的第二个位置分别是 3、1、2。另外，因为输出按照第一个 SELECT 列表，所以别名 My dream 定义在第一行；指示第二列不输出的 a\_dummy 也在第一列。

#### 教师注释

为了举例说明集合运算中的行顺序，运行脚本 demo\15\_setord.sql。简要地解释带 NOPRINT 选项的 COLUMN 命令。强调单引号在第二个 SELECT 语句的 'I'd like to teach' 文字中的用法。你可能需要提及，ORDER BY 只能用列、别名或第一个查询的列位置。

## 小结

在本课中，您应该已经学会如何：

- 用联合 (**UNION**) 返回所有不重复的行
- 用全联合 (**UNION ALL**) 返回所有行，包括重复行
- 用相交 (**INTERSECT**) 返回被两个查询共享的所有行
- 用相减 (**MINUS**) 返回由第一个查询选择但不被第二个查询选择的所有不重复的行
- **ORDER BY** 只能用在语句的最后

## 小结

- 用联合(**UNION**)运算返回所有查询的所有行，用联合运算从多个表中返回所有行，并且除去任何重复的行。
- 用全联合(**UNION ALL**)运算从多个表中返回所有行，不象 **UNION** 运算，重复的行不被除去，并且默认情况下输出不排序。
- 用相交(**INTERSECT**)运算返回多个表中所有的公共行。
- 用相减(**MINUS**)运算返回在第一个查询中返回的不在第二个查询中出现的行。
- **ORDER BY** 子句只能用在复合语句的最后。
- 确定在多个 **SELECT** 列表中的相应的表达式在数目和数据类型上匹配。

## 练习 15 概览

本章练习包括下面的主题：

- 用集合运算写查询
- 找可替换的连接方法

### 练习 15 概览

在本可的练习中，你将用集合(SET)运算写查询。

幻灯片 23

### 练习 15

1. 用集合运算，列出不包含 job ID ST\_CLERK 的部门的部门号。

DEPARTMENT_ID	
	10
	20
	60
	80
	90
	110
	190

7 rows selected.

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

2. 用集合运算，显示没有设立部门的国家的国家标识号和国家的名字。

CO	COUNTRY_NAME
DE	Germany

```
SELECT country_id,country_name
FROM countries
MINUS
SELECT l.country_id,c.country_name
FROM locations l, countries c
WHERE l.country_id = c.country_id;
```

3. 用集合运算，生成一个部门 10、50 和 20 为顺序的工作岗位列表，显示工作标识和部门标识。

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

```

COLUMN dummy PRINT
SELECT job_id, department_id, 'x' dummy
FROM employees
WHERE department_id = 10
UNION
SELECT job_id, department_id, 'y'
FROM employees
WHERE department_id = 50
UNION
SELECT job_id, department_id, 'z'
FROM employees
WHERE department_id = 20
ORDER BY 3;
COLUMN dummy NOPRINT

```

4. 列出雇员的雇员标识和工作标识,那些雇员现在的工作和他们上个任期的工作相同。

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

```

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

```



5. 写一个复合查询，列出下面的信息：

EMPLOYEES 表中所有雇员的名字和部门标识，不管是否他们属于任何部门，DEPARTMENTS 表中的所有部门标识和部门名称，不管这些部门有无雇员。

LAST_NAME	DEPARTMENT_ID	TO_CHAR(NULL)
Abel	80	
Davies	50	
De Haan	90	
Ernst	60	
Fay	20	
Gietz	110	
Grant		
Hartstein	20	
Higgins	110	
Hunold	60	
King	90	
Kochhar	90	
Lorentz	60	
Matos	50	
Mourgos	50	
Rajs	50	
Taylor	80	
Vargas	50	
Whalen	10	
Zlotkey	80	
	10	Administration
	20	Marketing
	50	Shipping
	60	IT
	80	Sales
	90	Executive
	110	Accounting
	190	Contracting

28 rows selected.

```

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

```