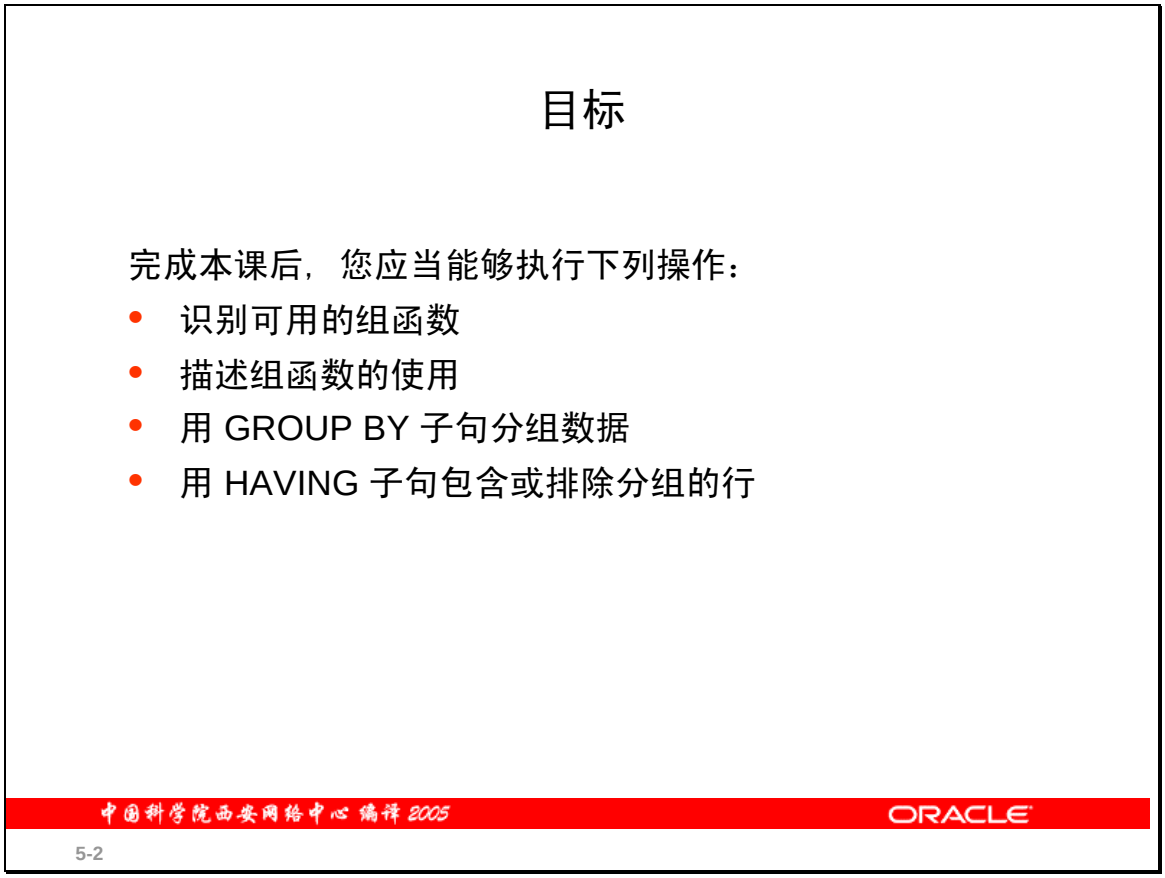
幻灯片 1

进度表：	时间	主题
	35 分钟	讲演
	40 分钟	练习
	75 分钟	总共

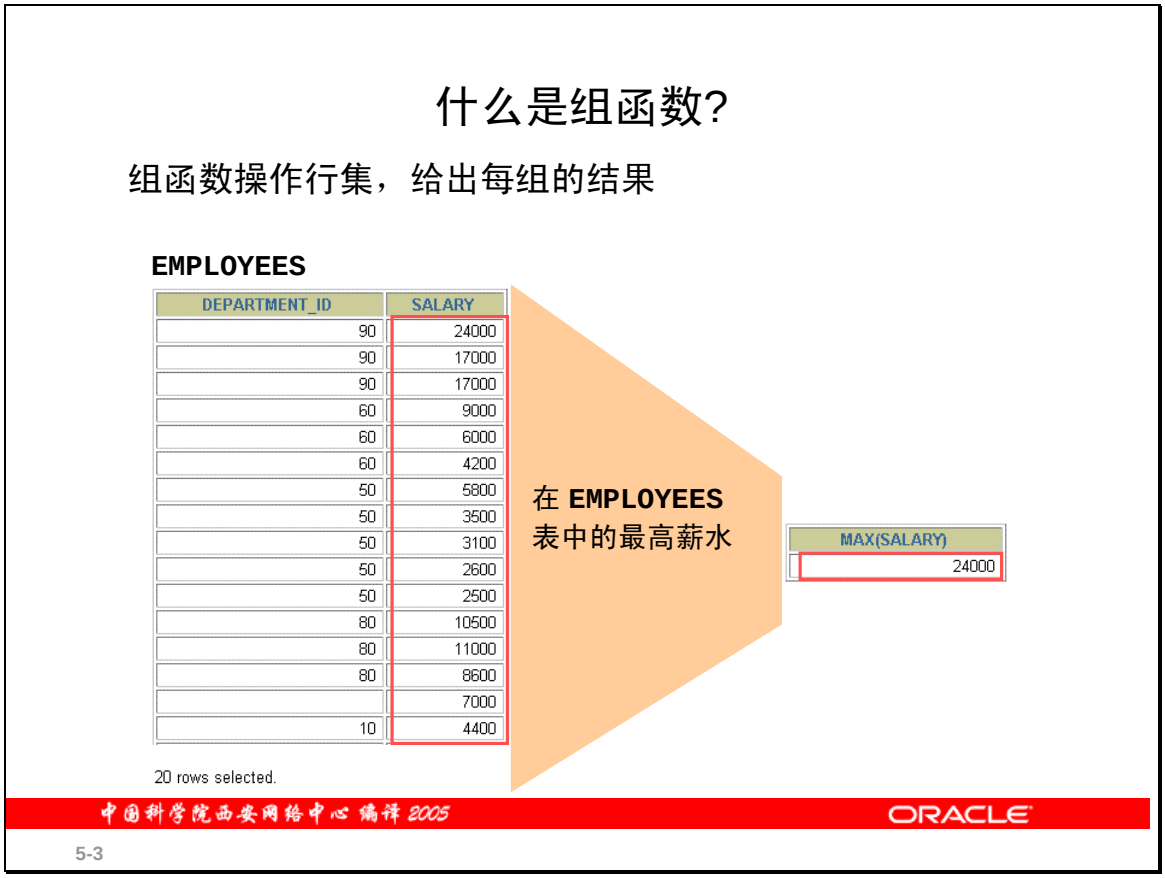
幻灯片 2



课程目标

本课进一步讲解函数。本课的重点是关于获得行聚集的概要信息，例如平均值。课程中将讨论怎样聚合表中的行到较小的集合中，怎样指定用于行聚合的搜索条件。

幻灯片 3



组函数

不象单行函数，组函数对行的集合进行操作，对每组给出一个结果。这些集合可能是整个表或者是表分成的组。

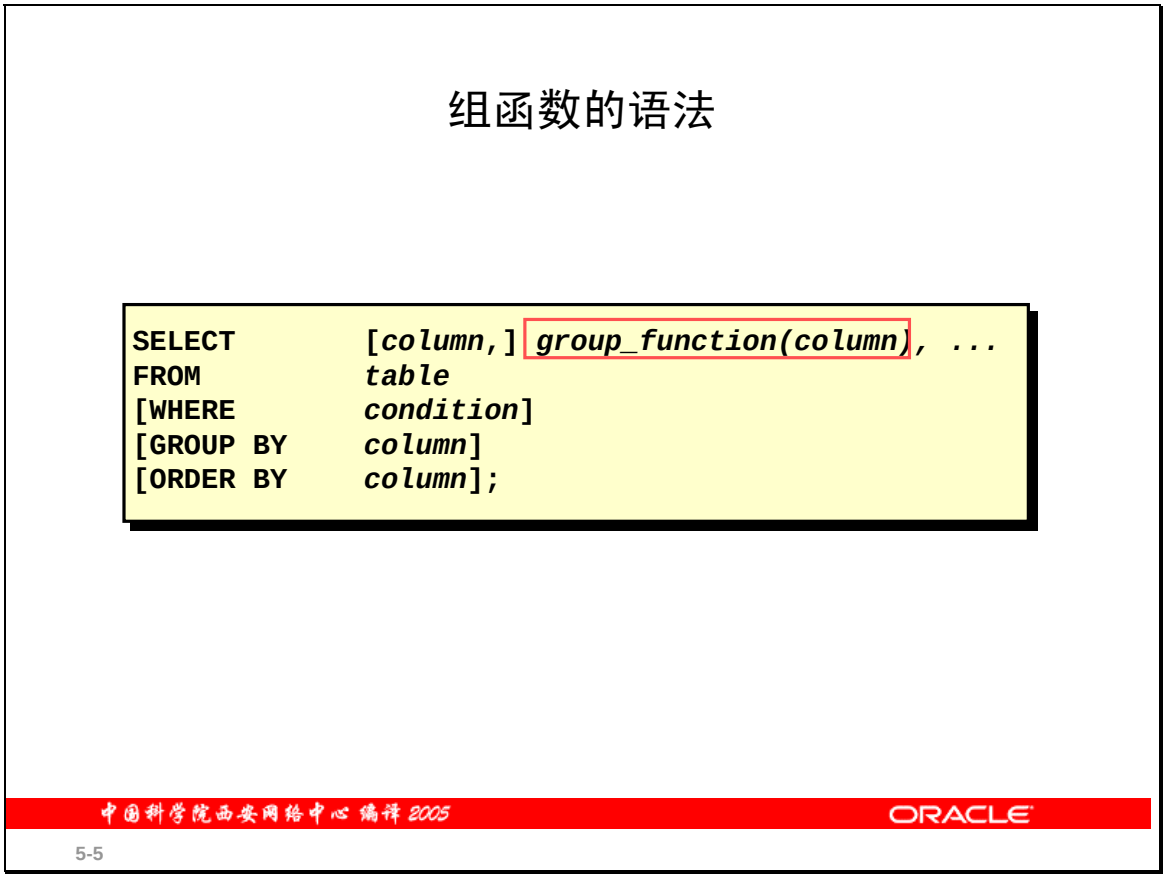
幻灯片 4

组函数 (续)

每个函数接收一个参数，下面的表确定你可以在语法中使用的选项：

函数	说明
AVG([DISTINCT|ALL]*n*)	*n* 的平均值，忽略空值
COUNT({*|[DISTINCT|ALL]*expr*})	行数，*expr* 求除了空计算(用 * 计数所有行，包括重复和带空值的行)
MAX([DISTINCT|ALL]*expr*)	*expr* 的最大值，忽略空值
MIN([DISTINCT|ALL]*expr*)	*expr* 的最小值，忽略空值
STDDEV([DISTINCT|ALL]*x*)	*n* 的标准差，忽略空值
SUM([DISTINCT|ALL]*n*)	合计 *n* 的值，忽略空值
VARIANCE([DISTINCT|ALL]*x*)	*n* 的方差，忽略空值

幻灯片 5



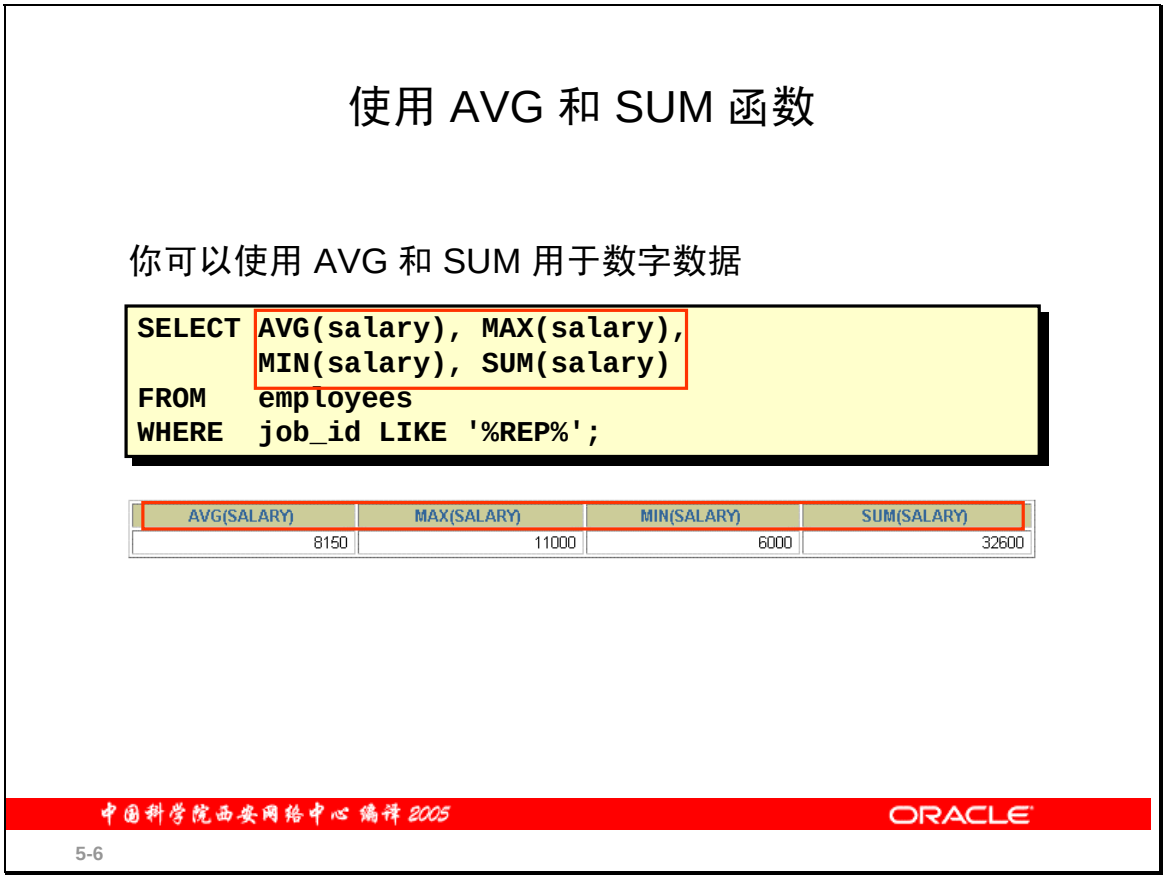
使用组函数的原则

- DISTINCT 使得函数只考虑不重复的值；ALL 使得函数考虑每个值，包括重复值。默认值是 ALL ，因此不需要指定。
- 用于函数的参数的数据类型可以是 CHAR、VARCHAR2、NUMBER 或 DATE。
- 所有组函数忽略空值。为了用一个值代替空值，用 NVL、NVL2 或 COALESCE 函数。
- 当使用 GROUP BY 子句时，Oracle 服务器隐式以升序排序结果集。为了覆盖该默认顺序，DESC 可以被用于 ORDER BY 子句。

教师注释

强调用 DISTINCT 和组函数忽略空值。ALL 是默认。

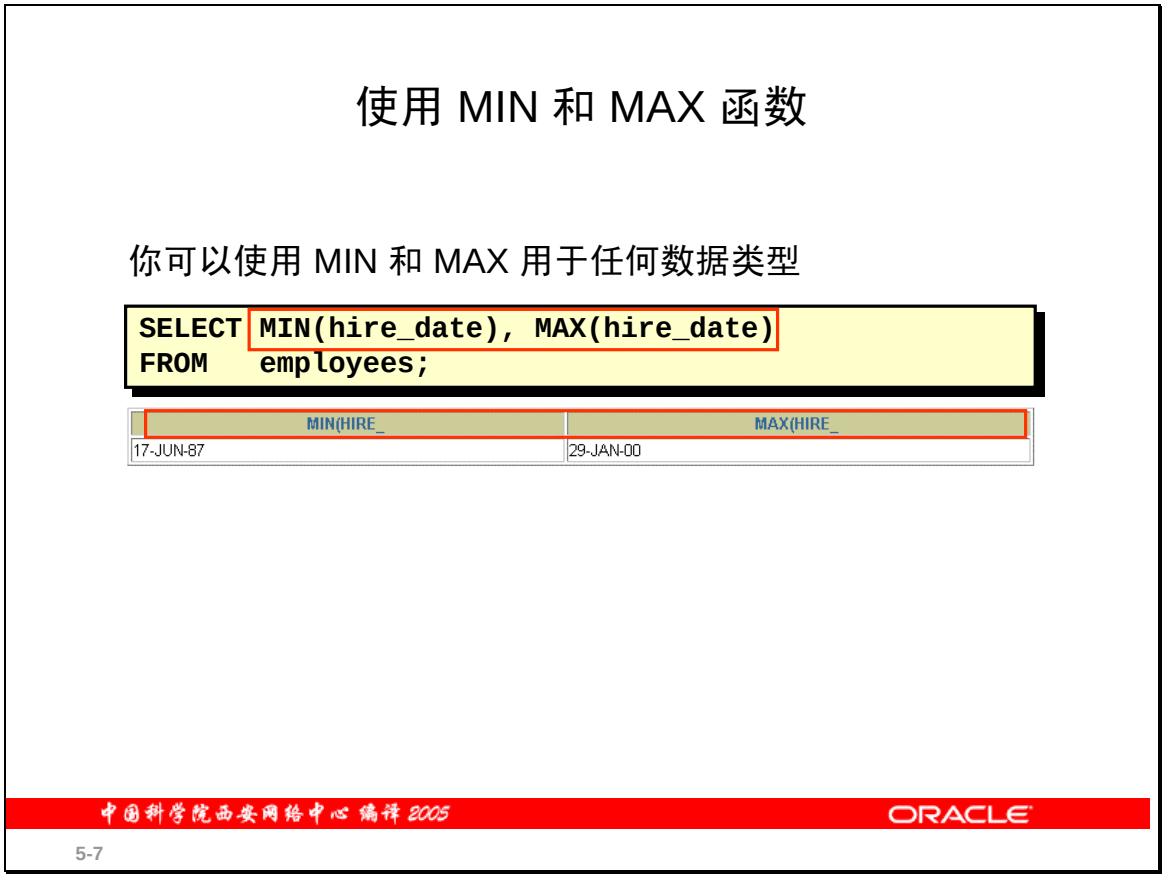
幻灯片 6



组函数

你可以使用 AVG、SUM、MIN 和 MAX 函数用于数字数据，幻灯片中例子显示平均、最高、最低和月薪水的和。

幻灯片 7



组函数 (续)

你可以用 MAX 和 MIN 函数于任意数据类型。幻灯片中的例子显示最年少的和做年老的雇员。

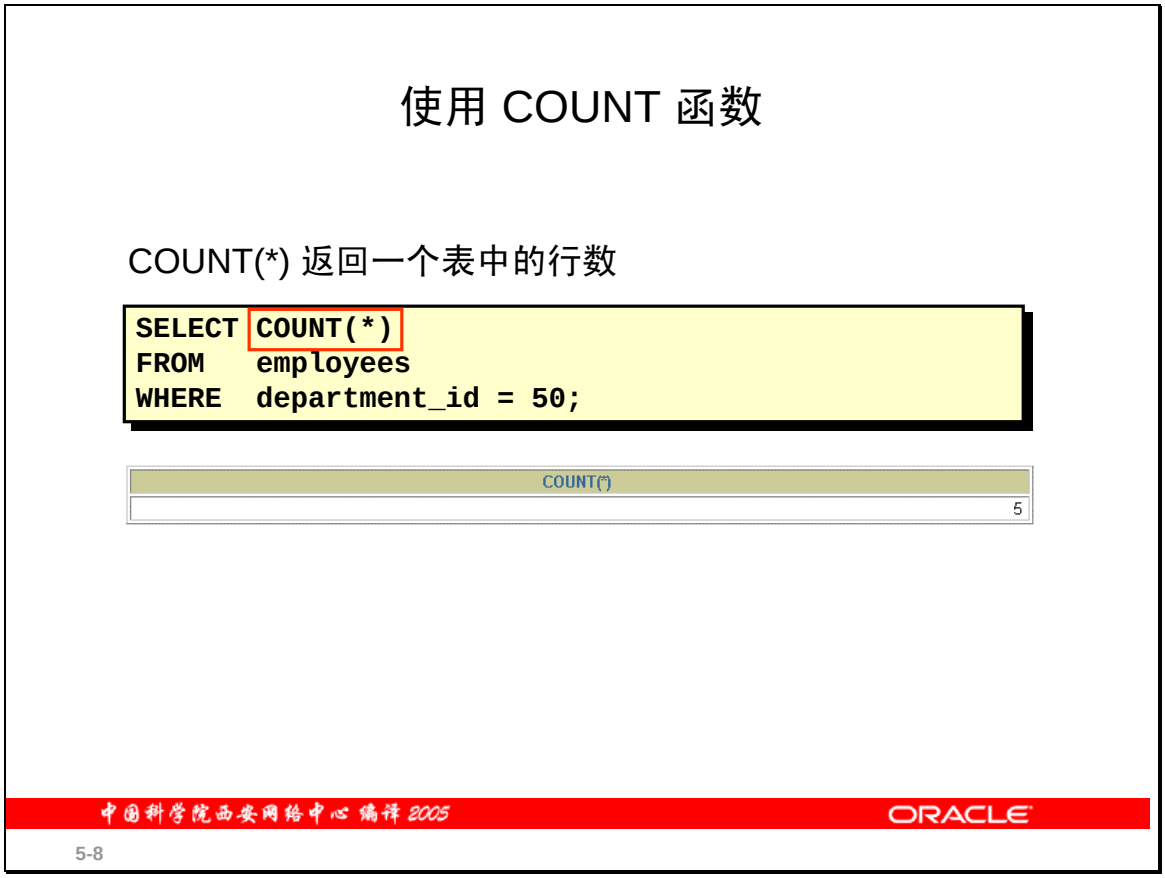
下面的例子显示按字母排序，雇员的排在最前面和最后面的名字。

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

MIN(LAST_NAME)	MAX(LAST_NAME)
Abel	Zlotkey

注：AVG、SUM、VARIANCE 和 STDDEV 函数只能被用于数字数据类型。

幻灯片 8



COUNT 函数

COUNT 函数有三中格式：

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

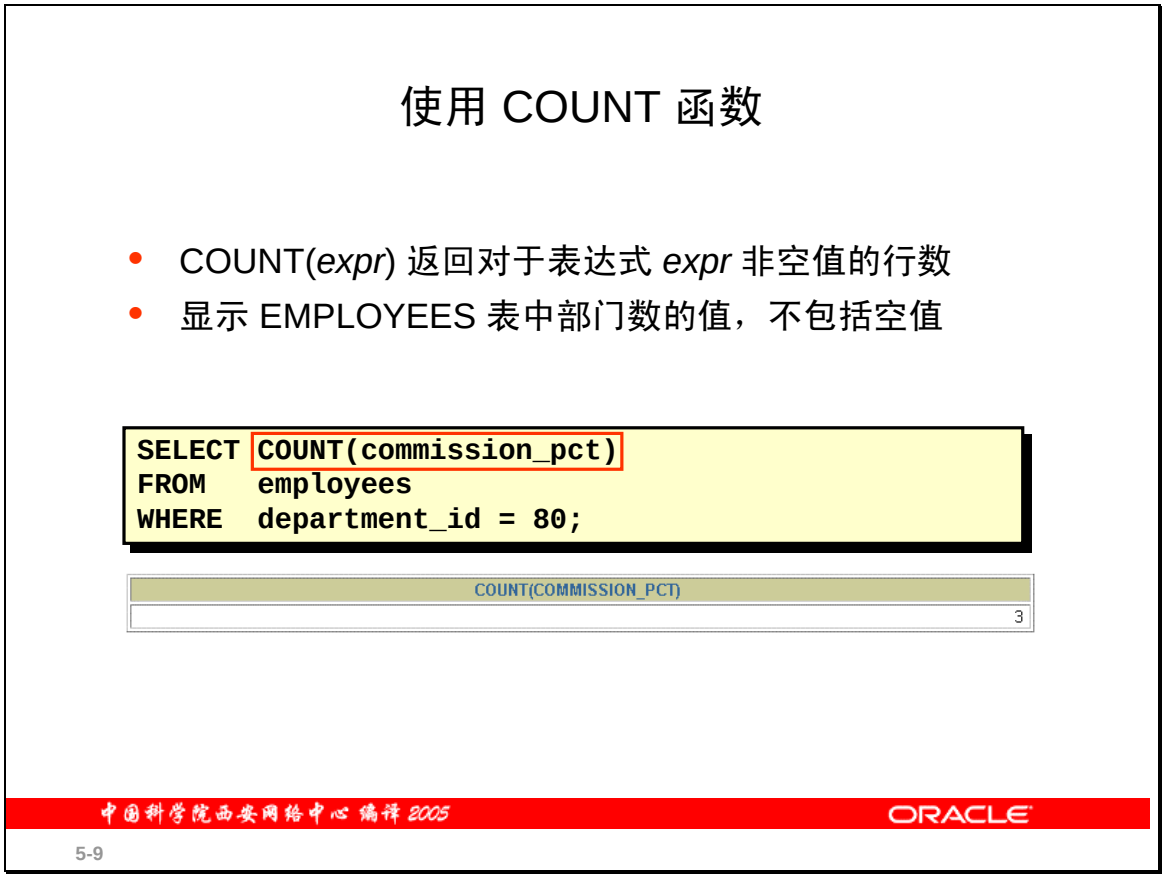
COUNT(*) 返回表中满足 SELECT 语句标准的行数，包括重复行，包括有空值列的行。如果 WHERE 子句包括在 SELECT 语句中，COUNT(*) 返回满足 WHERE 子句条件的行数。

对比，COUNT(*expr*) 返回在列中的由 *expr* 指定的非空值的数。

COUNT(DISTINCT *expr*) 返回在列中的由 *expr* 指定的唯一的非空值的数。

幻灯片的例子显示部门 50 中的雇员人数。

幻灯片 9



COUNT 函数 (续)

幻灯片的离子显示部门 80 中有佣金的雇员人数。

例

显示 EMPLOYEES 表中的部门数。

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

COUNT(DEPARTMENT_ID)
19

幻灯片 10

使用 DISTINCT 关键字

- COUNT(DISTINCT expr) 返回对于表达式 *expr* 非空并且值不相同的行数
- 显示 EMPLOYEES 表中不同部门数的值

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

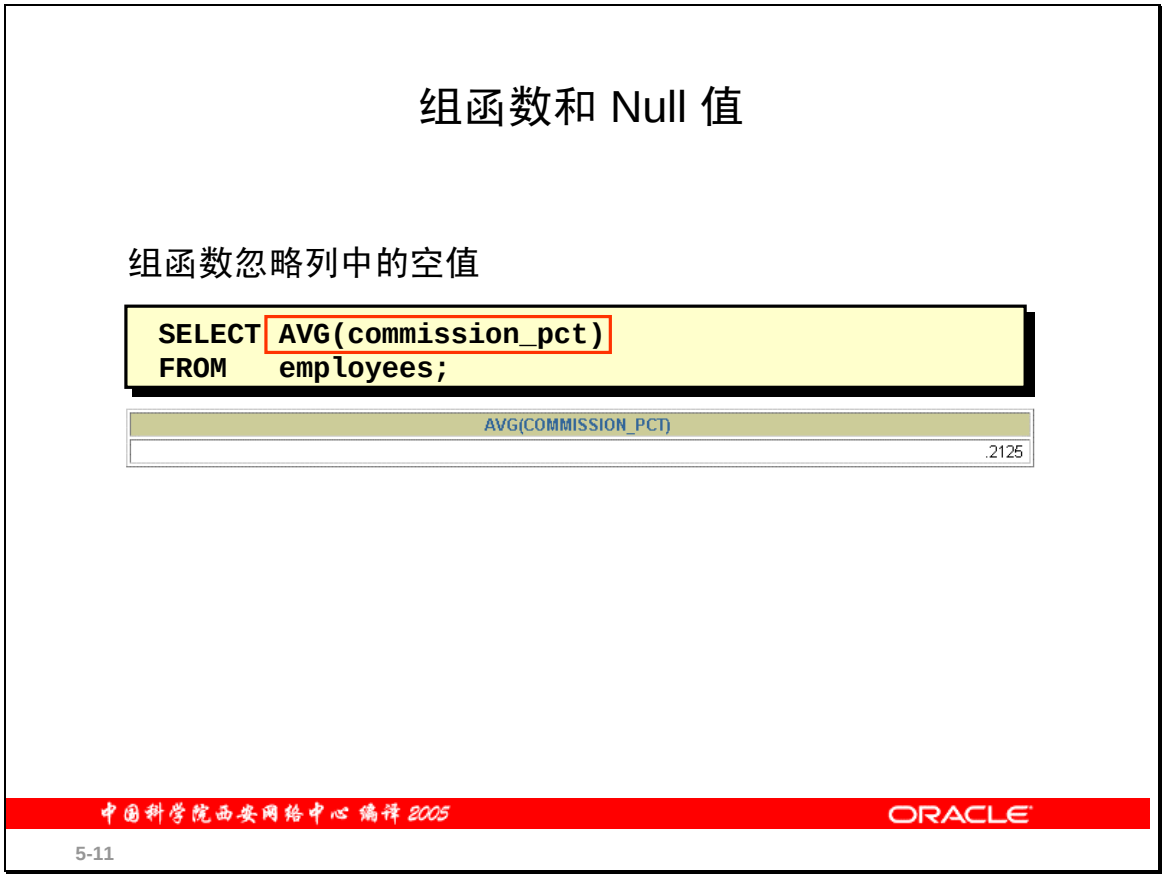
COUNT(DISTINCTDEPARTMENT_ID)
7

DISTINCT 关键字

使用 DISTINCT 关键字禁止计算在一列中的重复值。

幻灯片中的例子显示 EMPLOYEES 表中不重复的部门数。

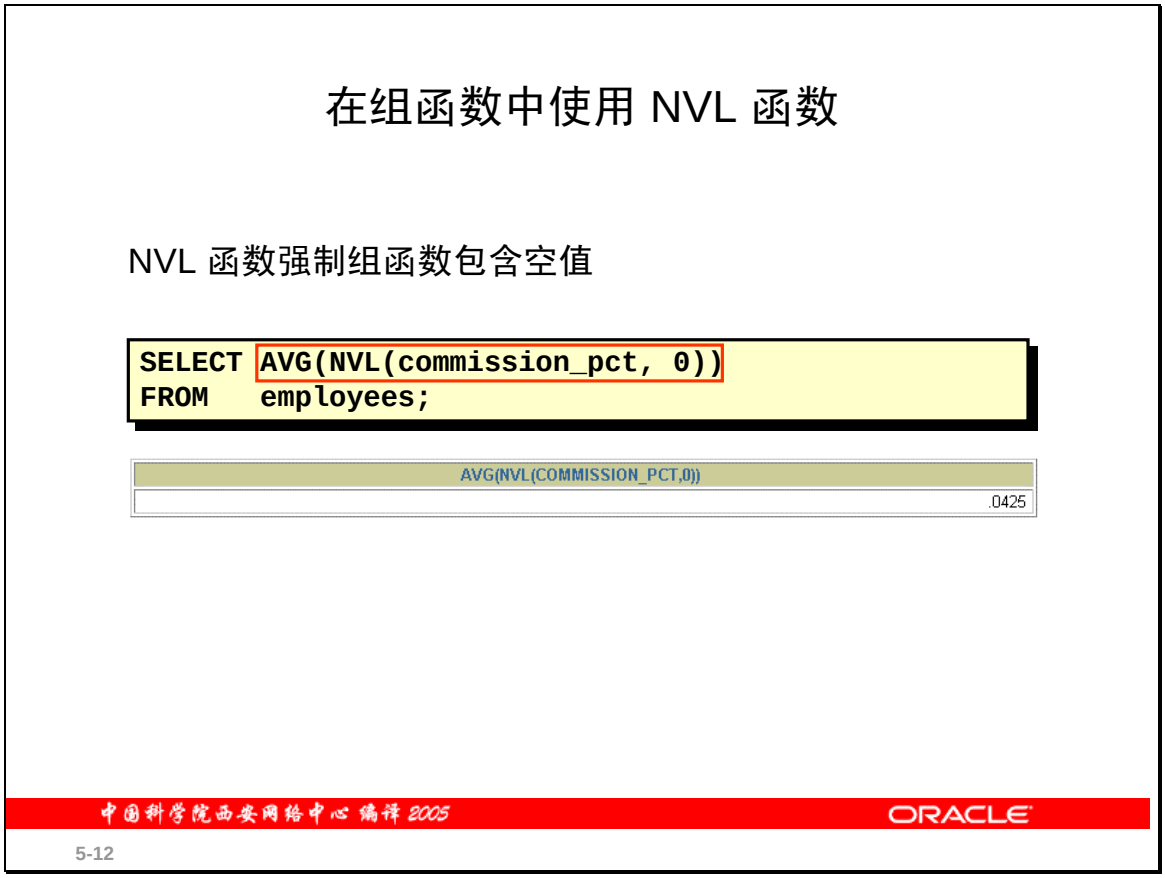
幻灯片 11



组函数和空值

所有组函数忽略列中的空值。在幻灯片的例子中，平均值只基于表中的那些 COMMISSION_PCT 列的值有效的行的计算。平均值计算是用付给所有雇员的总佣金除以接受佣金的雇员数 (4)。

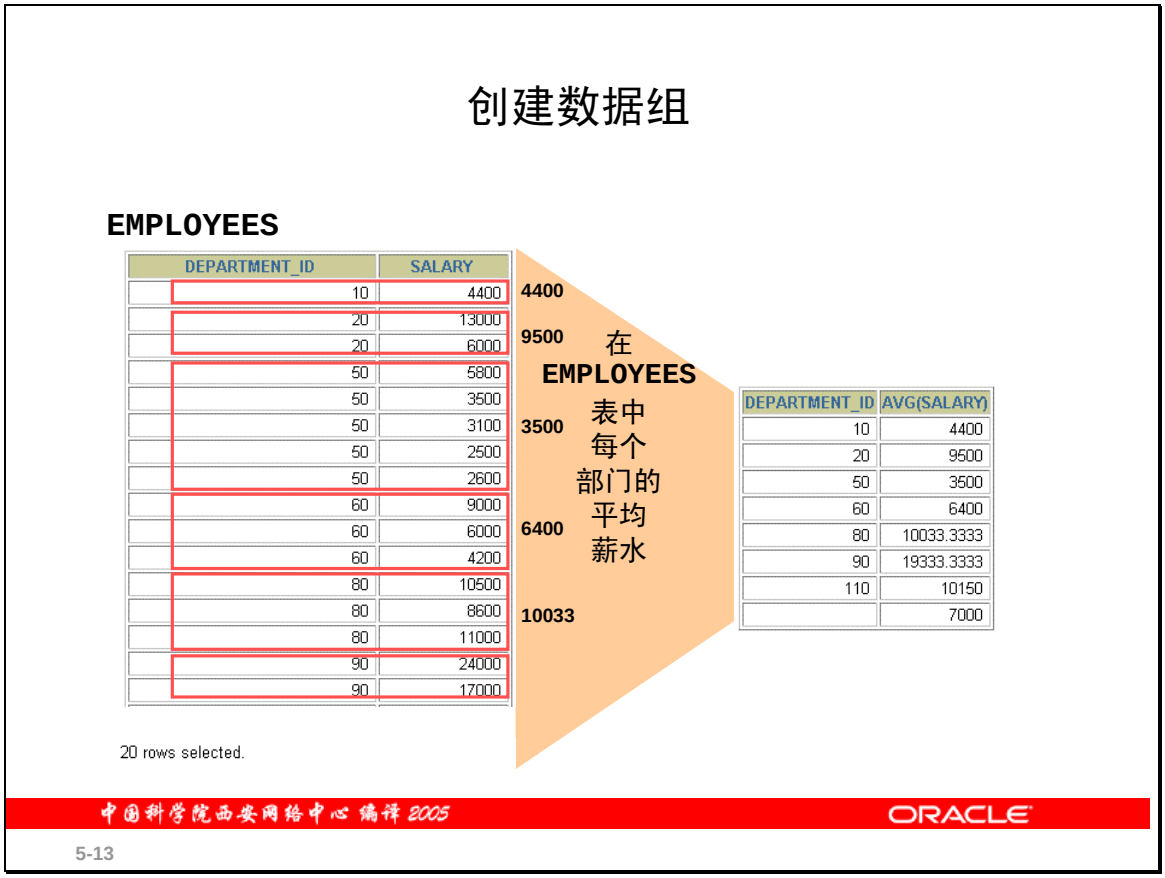
幻灯片 12



组函数和空值 (续)

NVL 函数强制组函数包括空值。在幻灯片的例子中，平均值被基于所有表中的行来计算，不管 COMMISSION_PCT 列是否为空。平均值的计算是用付给所有雇员的总佣金除以公司的雇员总数 (20)。

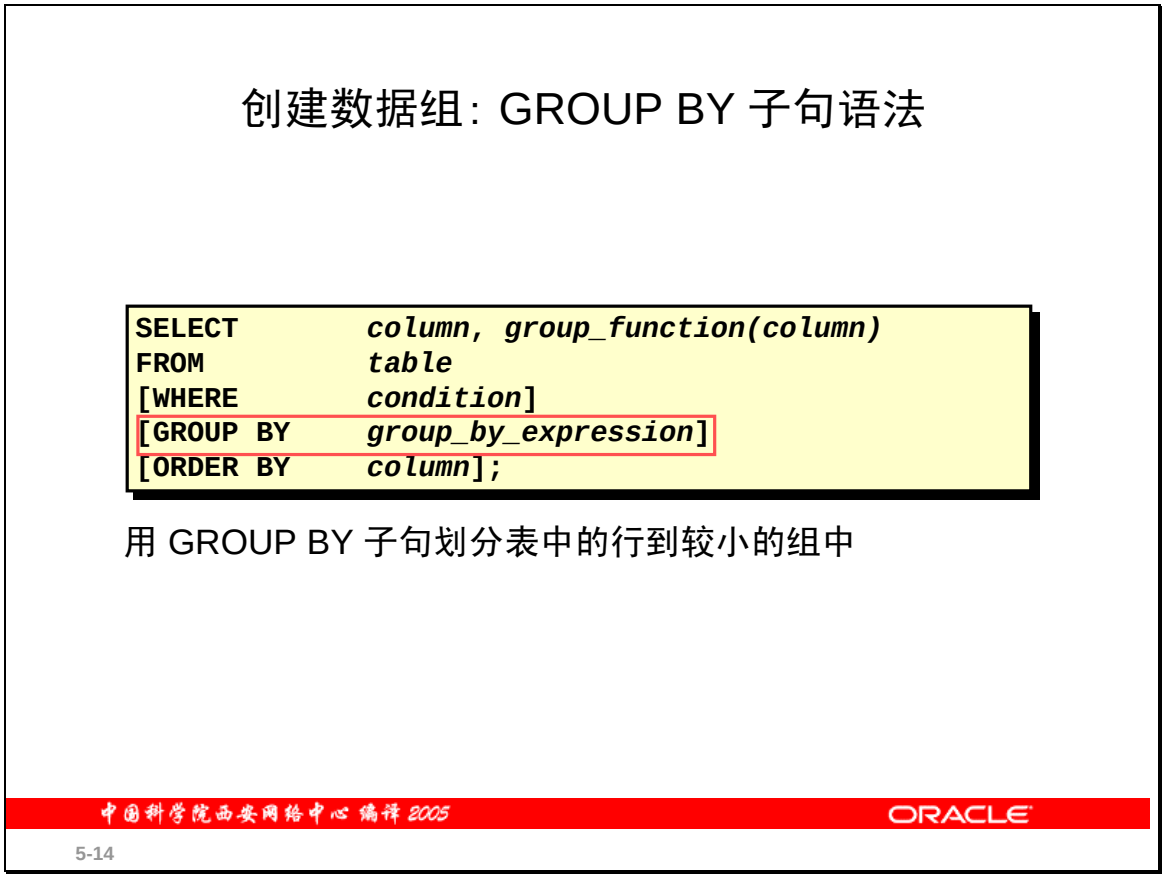
幻灯片 13



创建数据组

直到现在，所有组函数是将表作为一个大的信息组进行处理，有时，你需要将表的信息划分为较小的组，可以用 GROUP BY 子句实现。

幻灯片 14



GROUP BY 子句

你可以用 GROUP BY 子句把表中的行划分为组。然后你可以用组函数返回每一组的摘要信息。

在语法中：

group_by_expression 指定那些用于将行分组的列，这些列的值作为行分组的依据。

原则

- 如果在 SELECT 子句中包含了组函数，就不能选择单独的结果，除非单独的列出现在 GROUP BY 子句中。如果你未能在 GROUP BY 子句中包含一个字段列表，你会收到一个错误信息。
- 使用 WHERE 子句，你可以在划分行成组以前过滤行。
- 在 GROUP BY 子句中必须包含列。
- 在 GROUP BY 子句中你不能用列别名。
- 默认情况下，行以包含在 GROUP BY 列表中的字段的升序排序。你可以用 ORDER BY 子句覆盖这个默认值。

幻灯片 15

使用 GROUP BY 子句

在 SELECT 列表中的不在组函数中的所有列必须在 GROUP BY 子句中

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

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

GROUP BY 子句 (续)

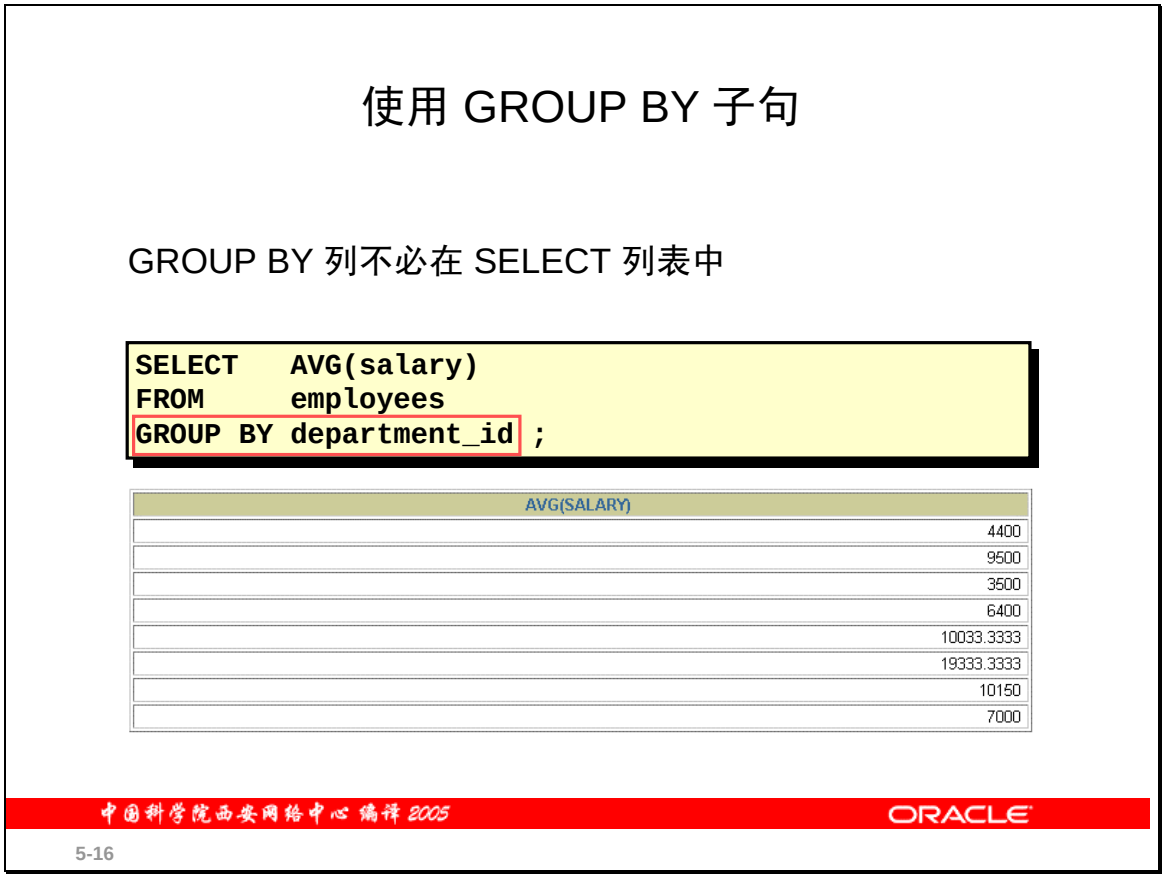
当使用 GROUP BY 子句时，确保在 SELECT 列表中的所有没有包括在组函数中的列必须在 GROUP BY 子句中。幻灯片中的例子显示每个部门的部门号和薪水的平均值。下面是包含一个 GROUP BY 子句 SELECT 语句的求值过程：

- SELECT 子句指定要返回的列：
- 在 EMPLOYEES 表中的部门号
  - 你在 GROUP BY 子句中指定分组的所有薪水的平均值
  - FROM 子句指定数据库必须访问的表：EMPLOYEES 表。
- WHERE 子句指定被返回的行。因为无 WHERE 子句默认情况下所有行被返回。
- GROUP BY 子句指定行怎样被分组。行用部门号分组，所以 AVG 函数被应用于薪水列，以计算每个部门的平均薪水。

教师注释

分组结果被以分组列隐式排序，可以用 ORDER BY 指定不同的排序顺序，但只能用组函数或分组列。

幻灯片 16



GROUP BY 子句 (续)

GROUP BY 列不必在 SELECT 子句中，例如，幻灯片中的 SELECT 语句显示每个部门的平均薪水，而没有显示各部门的部门号，但是结果却是以部门号分组的。

你可以在 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);
```

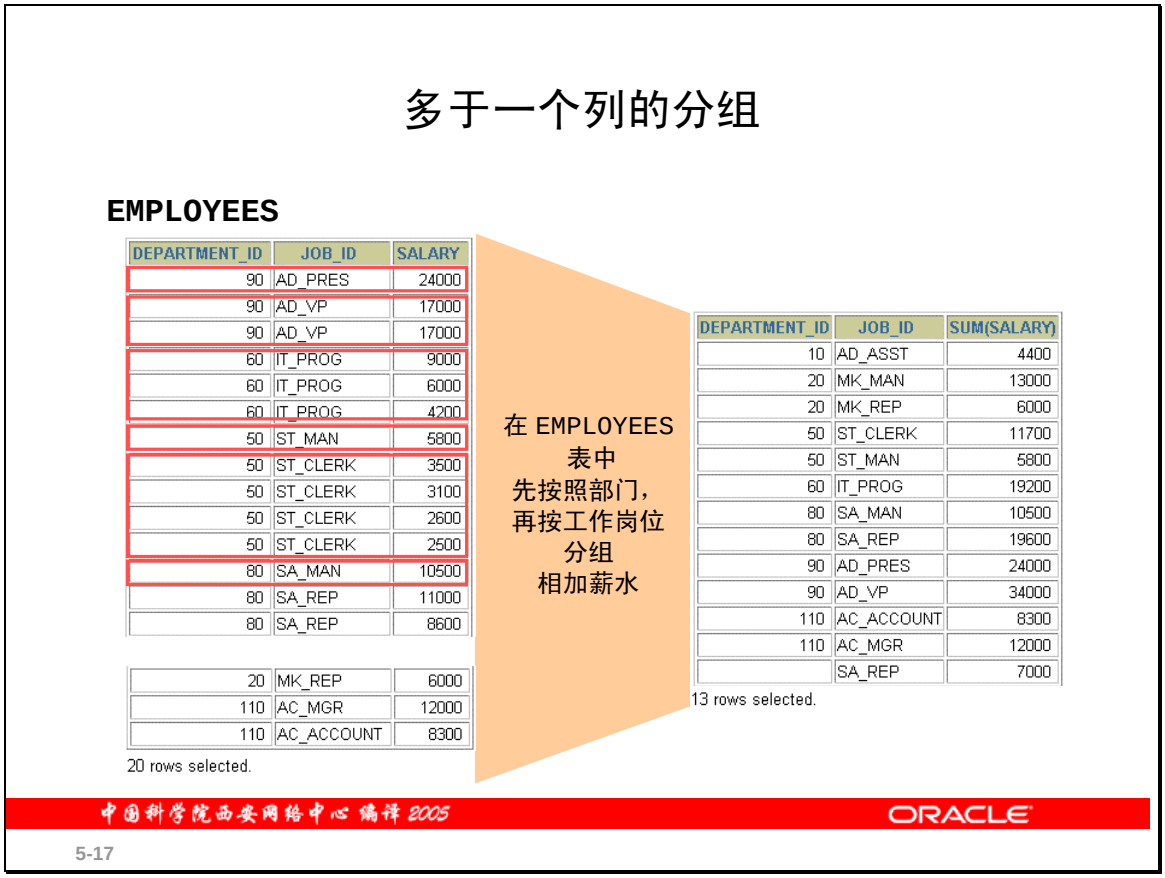
DEPARTMENT_ID	AVG(SALARY)
50	3500
10	4400
60	6400
90	19333.3333

8 rows selected.

教师注释

示范在 SELECT 语句中用和不用 DEPARTMENT_ID 列查询。本页和上页对比。

幻灯片 17



在组中分组

有时你需要看在组中分组的结果。幻灯片展示一个显示在每个部门中付给每个工作岗位的合计薪水的报告。

EMPLOYEES 表先用部门号分组，在分组中再用工作岗位分组。例如，在部门 50 中，4 个 stock clerks 被分组在一起，并且对在分组中的所有 stock clerks 产生一个单个的结果 (合计薪水)。

幻灯片 18

在多列上使用 GROUP BY 子句

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

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

在组中分组 (续)

你可以列出多个 GROUP BY 列返回组和子组的摘要结果。你可以用 GROUP BY 子句中的列的顺序确定结果的默认排序顺序。下面是幻灯片的 SELECT 语句中包含一个 GROUP BY 子句时的求值过程：

- SELECT 子句指定被返回的列：
  - 部门号在 EMPLOYEES 表中
  - Job ID 在 EMPLOYEES 表中
  - 你在 GROUP BY 子句中指定的组中所有薪水的合计
- FROM 子句指定数据库必须访问的表：EMPLOYEES 表。
- GROUP BY 子句指定你怎样分组行：
  - 首先，用部门号分组行。
  - 第二，在部门号的分组中再用 job ID 分组行。

如此 SUM 函数被用于每个部门号分组中的所有 job ID 的 salary 列。

幻灯片 19

非法使用 Group 函数的查询

- 在 SELECT 列表中的任何列或表达式（非计算列）必须在 GROUP BY 子句中
- 在 GROUP BY 子句中的列或表达式不必在 SELECT 列表中

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

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

列未包含在 GROUP BY 子句中

违反规定使用 Group 函数的查询

无论何时，你在同一个 SELECT 语句中使用单独的列 (DEPARTMENT_ID) 和组函数 (COUNT) 的混合时，你必须包括一个 GROUP BY 子句来指定单独的列 (在本例中，DEPARTMENT_ID)。如果缺少 GROUP BY，将会出现错误信息 “not a single-group group function”，并且一个星号 (*) 位于有问题的列的下面。你可以添加 GROUP BY 子句来纠正幻灯片中的这个错误。

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

DEPARTMENT_ID	COUNT(LAST_NAME)
10	1
20	2
	1

8 rows selected.

在 SELECT 列表中的任何没有使用聚集函数的列或表达式必须放在 GROUP BY 子句中。

幻灯片 20

非法使用 Group 函数的查询

- 不能使用 WHERE 子句来约束分组
- 可以使用 HAVING 子句来约束分组
- 在 WHERE 子句中不能使用组函数作为条件，只能用非计算列

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

```
WHERE  AVG(salary) > 8000
       *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

不能使用 **WHERE** 子句约束分组

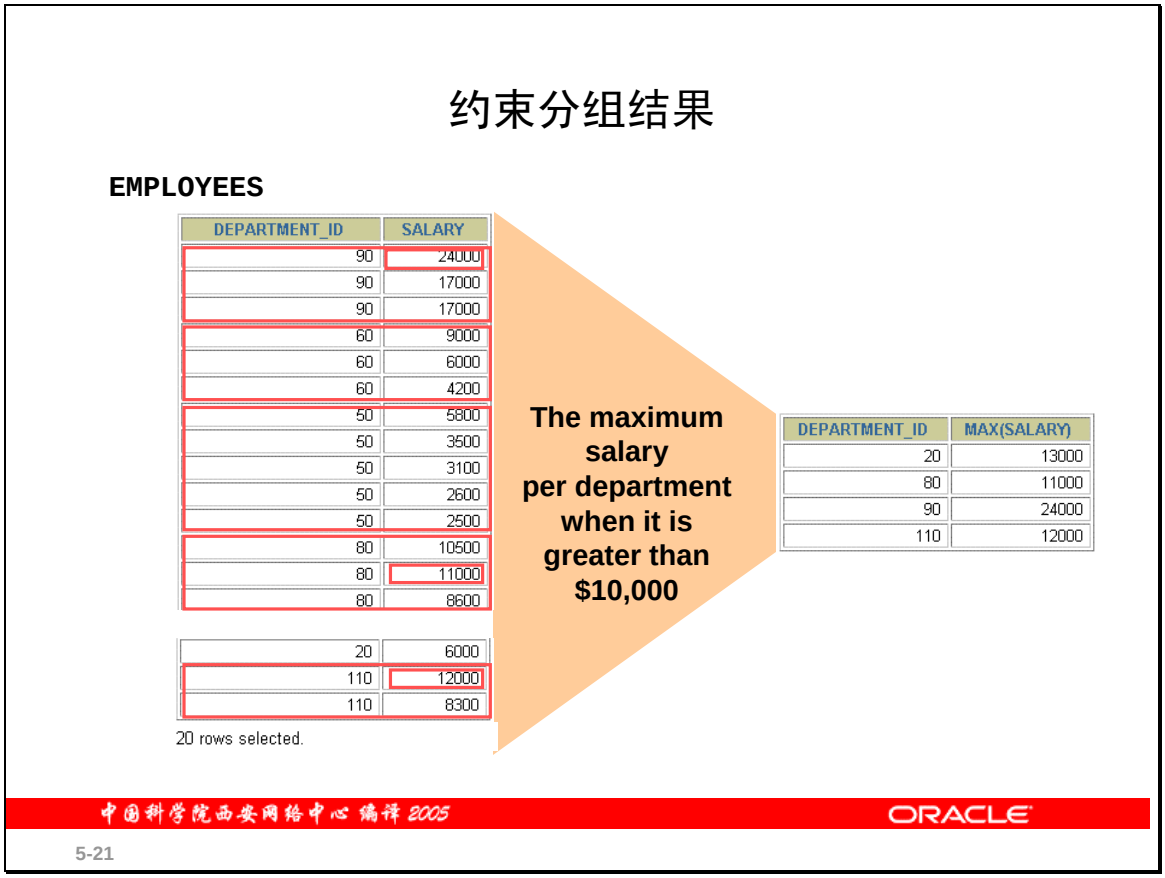
违反规定使用组函数查询 (续)

WHERE 子句不能用于限制组。幻灯片中的 SELECT 子句结果出现错误。因为使用 WHERE 子句限制了那些只有部门的平均薪水大于 $8,000 的分组才能显示。你可以用 HAVING 约束组来纠正幻灯片中的错误。

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

幻灯片 21

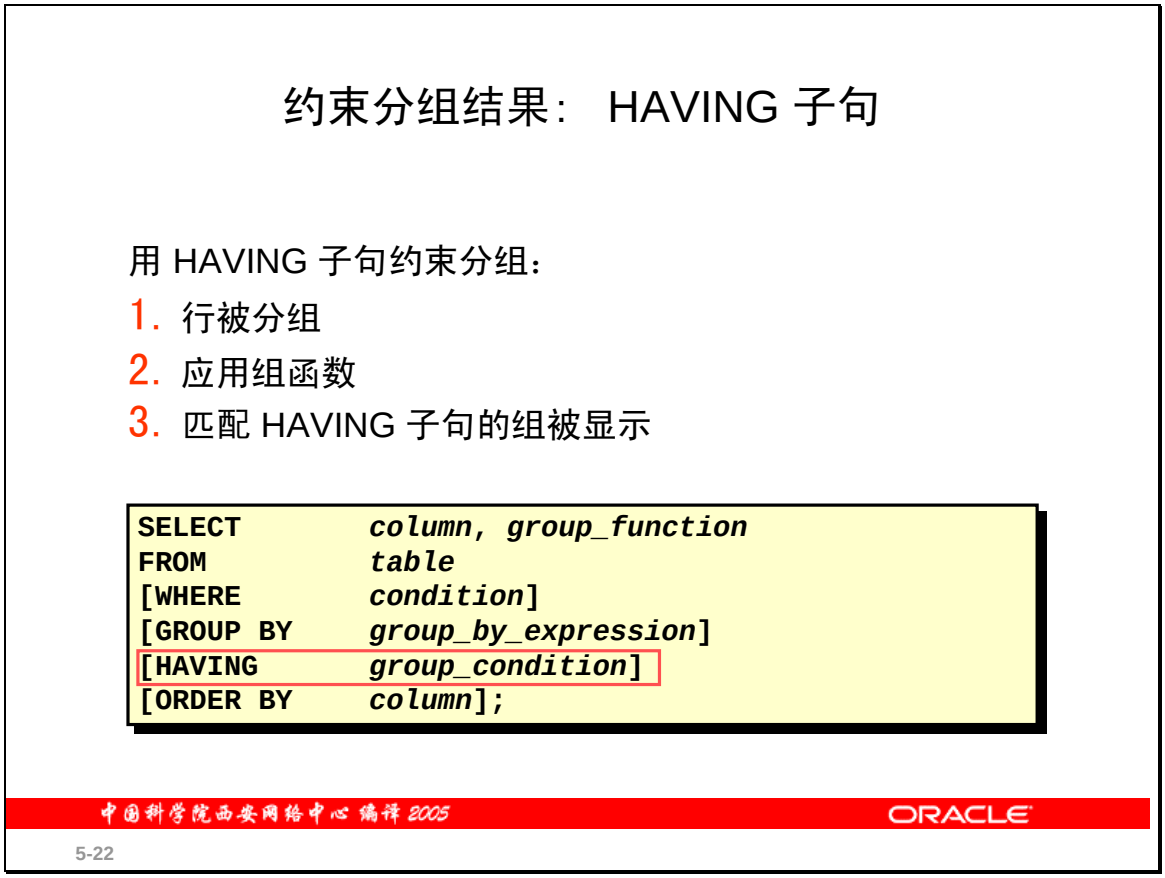


约束分组结果

用 WHERE 子句约束选择的行，用 HAVING 子句约束组。为了找到每个部门中的最高薪水，而且只显示最高薪水大于 $10,000 的那些部门，你可以象下面这样做：

1. 用部门号分组，在每个部门中找最大薪水。
2. 返回那些有最高薪水大于 $10,000 的雇员的部门。

幻灯片 22



HAVING 子句

使用 HAVING 子句指定哪个组将被显示，并且进一步基于聚集信息约束分组。

在语法中：

group_condition 限制行分组，返回那些指定条件为 `true` 的组

当你使用 HAVING 子句时，Oracle 服务器执行下面的步骤：

1. 行被分组。
2. 组函数被用于分组。
3. 匹配 `HAVING` 子句的标准的组被显示。

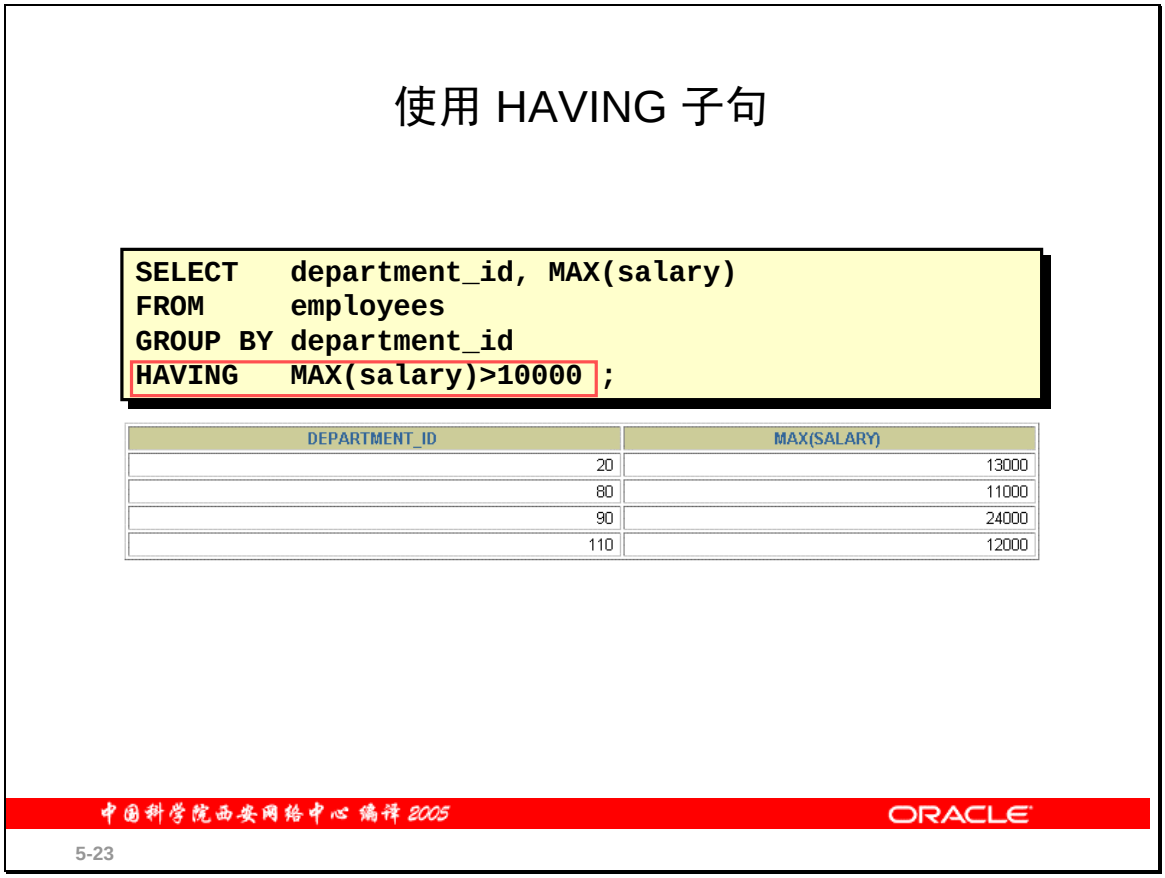
HAVING 子句可以优先于 GROUP BY 子句，但建议你先用 GROUP BY 子句，因为这样更合逻辑。在 HAVING 子句被用于 SELECT 列表的组之前，分组已形成，并且组函数已被计算。

教师注释

Oracle 服务器以下面的顺序求子句的值：

- 如果语句包含一个 WHERE 子句，服务器建立候选行。
- 服务器确定在 GROUP BY 子句中指定的分组。
- HAVING 子句进一步约束结果那些在 HAVING 子句中不满足分组标准的组。

幻灯片 23



HAVING 子句 (续)

幻灯片的例子显示那些最高薪水大于 $10,000 的部门的部门号和最高薪水。你可以在 SELECT 列表中不使用组函数而使用 GROUP BY 子句。

如果你基于组函数的结果约束行，你必须有一个 GROUP BY 子句以及 HAVING 子句。

下面的例子显示那些最高薪水大于 $10,000 的部门的部门号和平均薪水。

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

幻灯片 24

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
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

HAVING 子句 (续)

幻灯片中的例子显示那些合计薪水册超过 $13,000 的每个工作岗位的 job ID 和合计薪水。该例子排除了销售代表，并且用合计月薪排序列表。

幻灯片 25

嵌套组函数

显示最大平均薪水

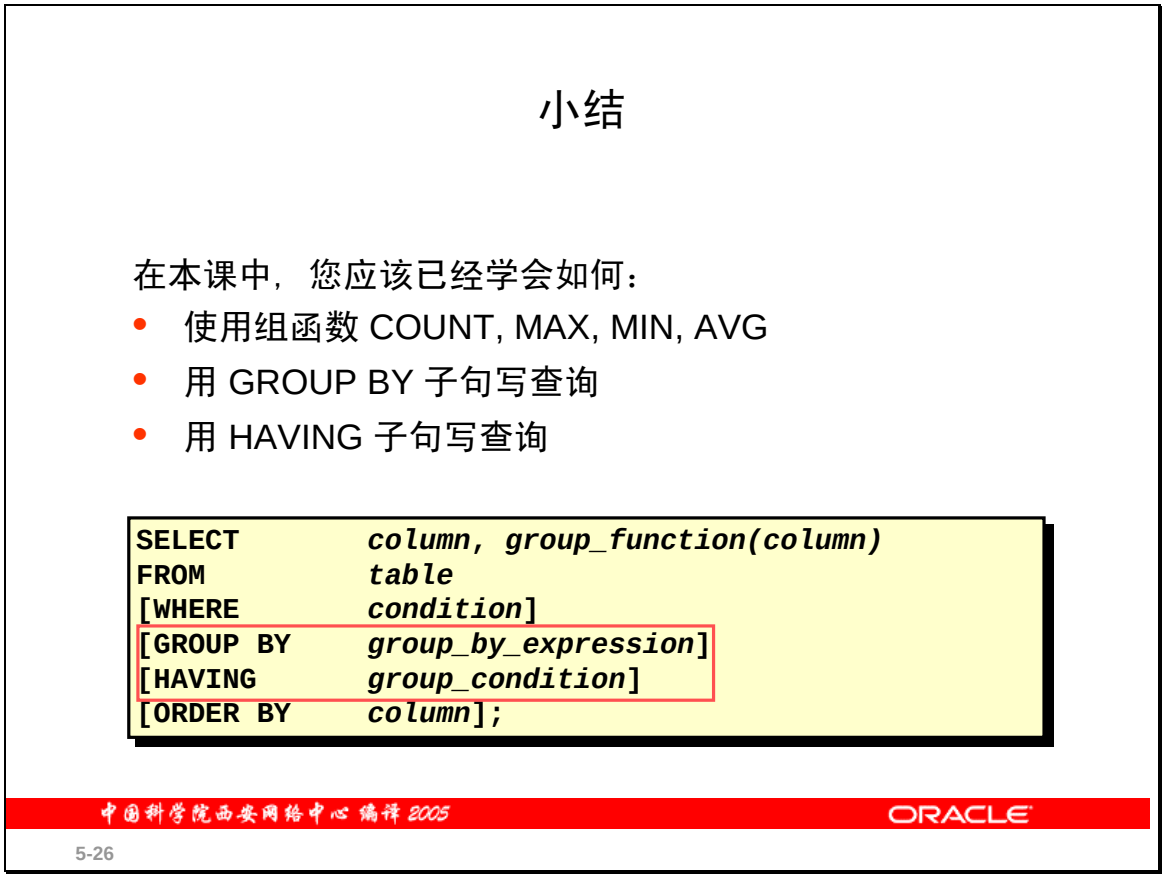
```
SELECT   MAX(AVG(salary))
FROM     employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

嵌套组函数

组函数可以被嵌套两层深度，幻灯片中的例子显示最高平均薪水。

幻灯片 26



小结

在 SQL 中有 7 个组函数可用：

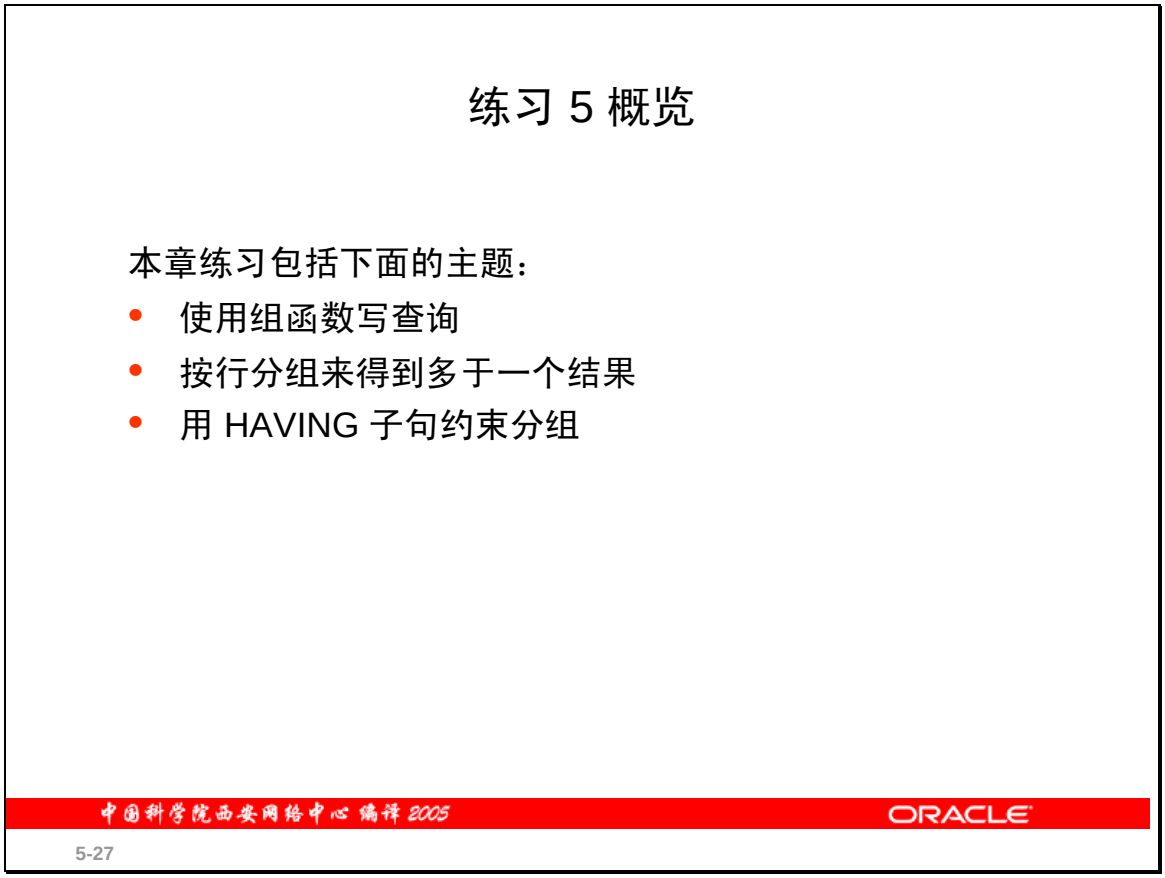
- AVG 平均值
- COUNT 计数
- MAX 最大值
- MIN 最小值
- SUM 合计
- STDDEV 标准差
- VARIANCE 方差

你可以用 GROUP BY 子句创建子分组。分组可以用 HAVING 子句约束。在语句中将 HAVING 和 GROUP BY 子句放在 WHERE 子句的后面。将 ORDER BY 子句放在最后。

Oracle 服务器按下面的顺序求子句的值：

1. 如果语句包含一个 WHERE 子句，服务器建立侯选行。
2. 服务器确定在 GROUP BY 子句中指定的组。
3. HAVING 子句进一步约束不满足在 HAVING 子句中分组标准的结果分组。

幻灯片 27



练习 5 概览

在该练习结束后，你应该熟悉使用组函数和选择数据分组。

基于卷面的问题

问题 1-3，圈 True 或 False。

注：列别名被用于问题中。

教师注释

提示问题 #7：告诉学生思考关于 EMPLOYEES 表中的 MANAGER_ID 列，什么时候决定经理号，而不是工作岗位列。

练习 5

决定下面三个语句的合法性，圈 True 或 False。

1. 组函数在多行上计算，对每每个组产生一个结果。
   True/False

   True

2. 组函数在计算中包含空值。
   True/False

   False. 组函数忽略空值。如果你想要包含空值，使用 NVL 函数。

3. 在分组计算中，WHERE 子句对行的限制在计算的前面。
   `True/False`

   True

4. 显示所有雇员的最高、最低、合计和平均薪水，列标签分别为：Maximum、Minimum、Sum 和 Average。四舍五入结果为最近的整数，竟语句存在文本文件 lab5_4.sql 中。

Maximum	Minimum	Sum	Average
24000	2500	175500	8775

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

5. 修改 lab5_4.sql 中的问题显示每中工作类型的最低、最高、合计和平均薪水。再保存 `lab5_4.sql` 到 `lab5_5.sql`。运行 lab5_5.sql 中的语句。

JOB_ID	Maximum	Minimum	Sum	Average
AC_ACCOUNT	8300	8300	8300	8300
AC_MGR	12000	12000	12000	12000
AD_ASST	4400	4400	4400	4400
AD_PRES	24000	24000	24000	24000
AD_VP	17000	17000	34000	17000
IT_PROG	9000	4200	19200	6400
MK_MAN	13000	13000	13000	13000
MK_REP	6000	6000	6000	6000
SA_MAN	10500	10500	10500	10500
SA_REP	11000	7000	26600	8867
ST_CLERK	3500	2500	11700	2925
ST_MAN	5800	5800	5800	5800

12 rows selected.

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

6. 写一个查询显示每一工作岗位的人数。

JOB_ID	COUNT(*)
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD_PRES	1
AD_VP	2
IT_PROG	3
MK_MAN	1
MK_REP	1
SA_MAN	1
SA_REP	3
ST_CLERK	4
ST_MAN	1

12 rows selected.

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

7. 确定经理人数，不需要列出他们，列标签是 Number of Managers。***提示：****用 MANAGER_ID 列决定经理号。*

Number of Managers
8

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

8. 写一个查询显示最高和最低薪水之间的差。列标签是 DIFFERENCE。

DIFFERENCE
21500

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

如果你后时间，完成下面的习题：

9. 显示经理号和经理付给雇员的最低薪水。排除那些经理未知的人。排除最低薪水小于等于 $6,000 的组。按薪水降序排序输出。

MANAGER_ID	MIN(SALARY)
102	9000
205	8300
149	7000

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

10. 写一个查询显示每个部门的名字、地点、人数和部门中所有雇员的平均薪水。四舍五入薪水到两位小数。

Name	Location	Number of People	Salary
Accounting	1700	2	10150
Administration	1700	1	4400
Executive	1700	3	19333.33
IT	1400	3	6400
Marketing	1800	2	9500
Sales	2500	3	10033.33
Shipping	1500	5	3500

7 rows selected.

```
SELECT d.department_name "Name", d.location_id "Location",
COUNT(*) "Number of People",
ROUND(AVG(salary),2) "Salary"
FROM employees e, departments d
WHERE e.department_id = d.department_id
GROUP BY d.department_name, d.location_id;
```

如果你想要接受额外的挑战，完成下面的习题：

11. 创建一个查询显示雇员总数，和在 1995、1996、1997 和 1998 受雇的雇员人数。创建适当的列标题。

TOTAL	1995	1996	1997	1998
20	1	2	2	3

```
SELECT COUNT(*) total,
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1995,1,0))"1 995",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1996,1,0))"1 996",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1997,1,0))"1 997",
SUM(DECODE(TO_CHAR(hire_date, 'YYYY'),1998,1,0))"1 998"
FROM employees;
```

12. 创建一个混合查询显示工作岗位和工作岗位的薪水合计，并且合计部门 20、50、80 和 90 的工作岗位的薪水。给每列一个恰当的列标题。

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
AC_ACCOUNT					8300
AC_MGR					12000
AD_ASST					4400
AD_PRES				24000	24000
AD_VP				34000	34000
IT_PROG					19200
MK_MAN	13000				13000
MK_REP	6000				6000
SA_MAN			10500		10500
SA_REP			19600		26600
ST_CLERK		11700			11700
ST_MAN		5800			5800

12 rows selected.

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