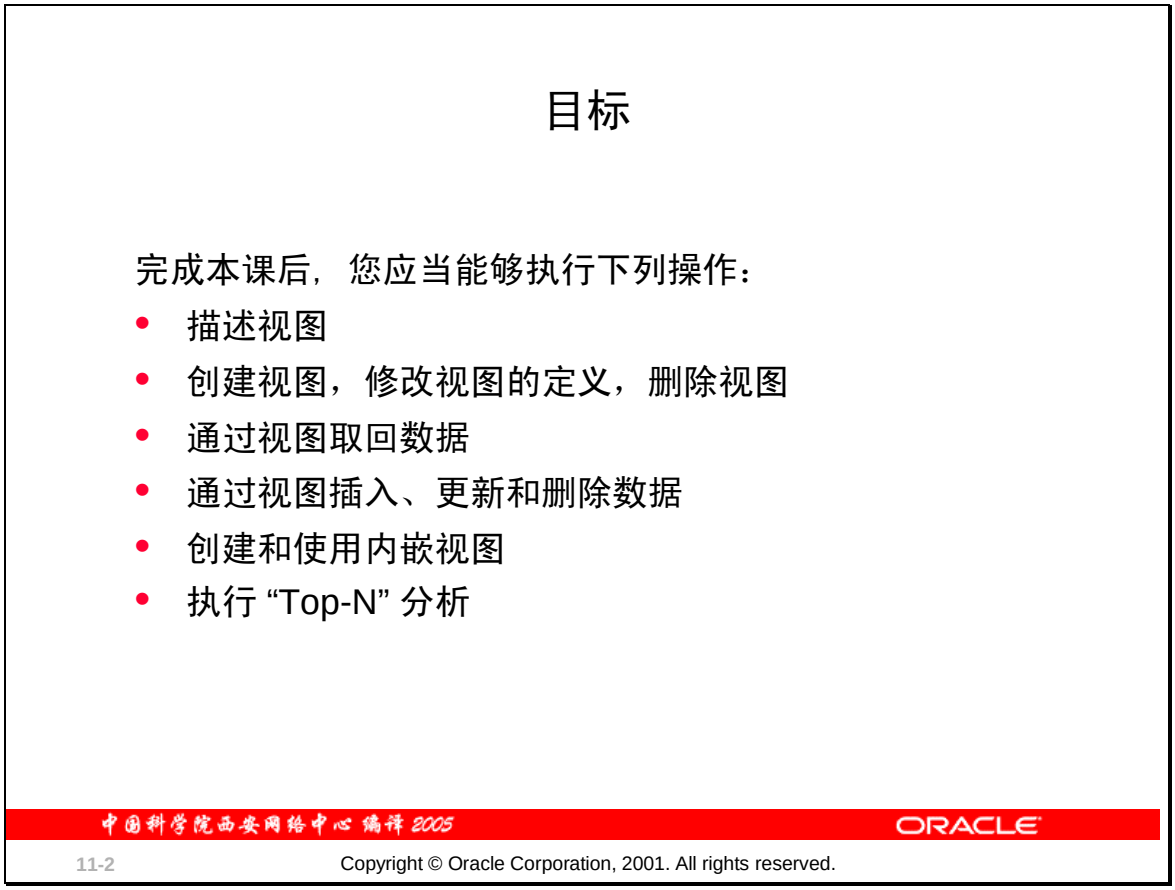
幻灯片 1

进度表：	时间	主题
	20 分钟	讲演
	20 分钟	练习
	40 分钟	总共

幻灯片 2



课程目标

在本课中，你将学习怎样创建和使用视图，你还将学习查询有关的数据字典对象来获得视图信息，最后，你将学习创建和使用内嵌视图，并且用内嵌视图进行 Top-N 分析。

幻灯片 3

数据库对象

对象	说明
Table (表)	基本存储单元；由行和列组成
View (视图)	数据来自一个或者多个表的数据子集的逻辑表示
Sequence (序列)	产生主键的值
Index (索引)	改善某些查询的性能
Synonym (同义词)	一个对象的替换名字

幻灯片 4

什么是视图?

你可以通过创建表的视图来表现数据的逻辑子集或数据的组合。视图是基于表或另一个视图的逻辑表，一个视图并不包含它自己的数据，它象一个窗口，通过该窗口可以查看或改变表中的数据。视图基于其上的表称为基表。视图在数据字典中作为一个 SELECT 语句存储。

教师注释

演示：11_easyvu.sql

目的：幻灯片上的视图显示由下面的语句创建：

```
CREATE OR REPLACE VIEW simple_vu
AS SELECT employee_id, last_name, salary
   FROM   employees;
```

幻灯片 5

为什么用视图?

- 限制数据访问
- 使得复杂的查询容易
- 提供数据的独立性
- 表现相同数据的不同观察

视图的优越性

- 视图限制数据的访问，因为视图能够选择性的显示表中的列。
- 视图可以用来构成简单的查询以取回复杂查询的结果。例如，视图能用于从多表中查询信息，而用户不必知道怎样写连接语句。
- 视图对特别的用户和应用程序提供数据独立性，一个视图可以从几个表中取回数据。
- 视图提供用户组按照他们的特殊标准访问数据。

更多信息见 *Oracle9i SQL Reference*，“创建视图”。

幻灯片 6

简单视图和复杂视图

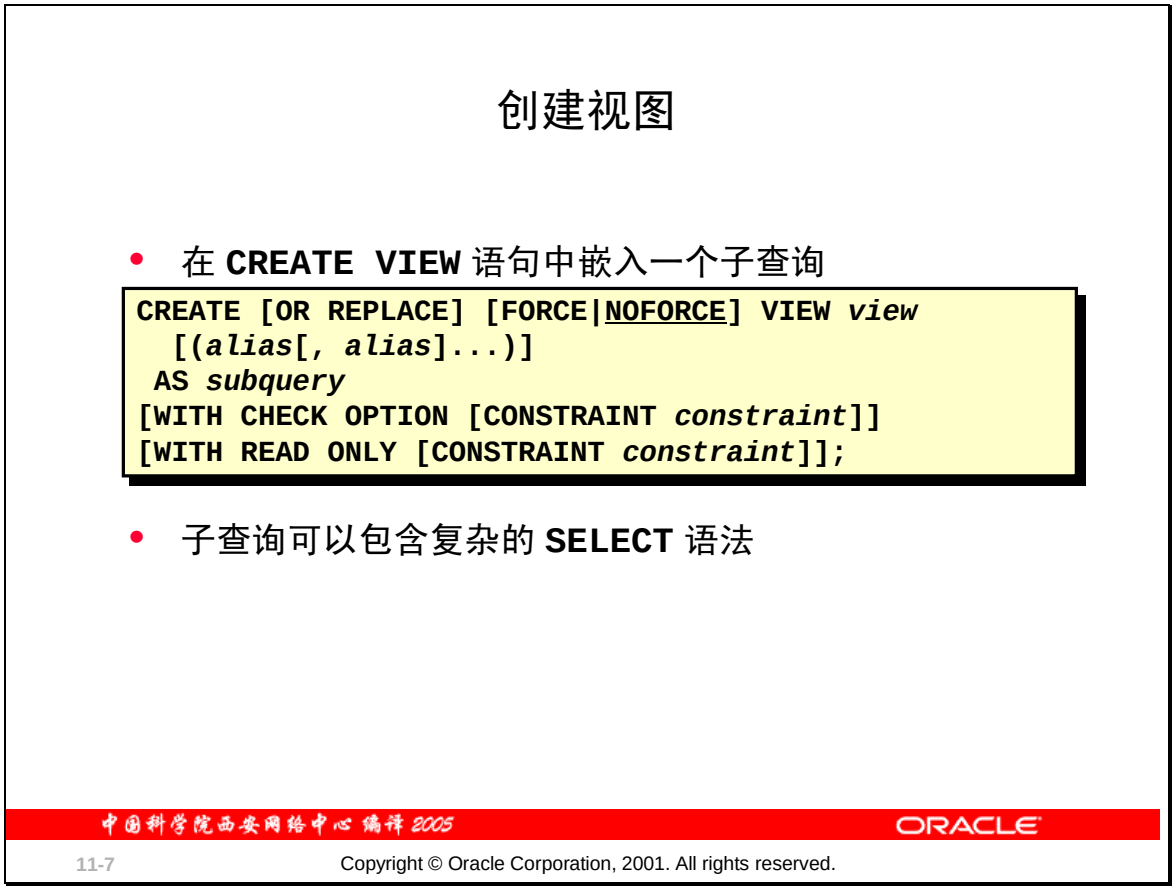
特性	简单视图	复杂视图
表的数目	一个	一个或多个
包含函数	无	有
包含数据分组	无	有
通过视图进行 DML 操作	是	不允许

简单视图 VS 复杂视图

视图有两种分类：简单和复杂，基本区别涉及 DML (INSERT、UPDATE 和 DELETE) 操作。

- 下面是简单视图：
  - 数据仅来自一个表
  - 不包含函数或数据分组
  - 能通过视图执行 DML 操作
- 下面是复杂视图：
  - 数据来自多个表
  - 包含函数或数据分组
  - 不总是允许通过视图进行 DML 操作

幻灯片 7



创建查询

你可以在 CREATE VIEW 语句中嵌入一个子查询来创建一个视图。

在语法中：

OR REPLACE	如果视图已经存在重新创建它
FORCE	创建视图，而不管基表是否存在
NOFORCE	只在基表存在的情况下创建视图(这是默认值)
view	视图的名字
alias	为由视图查询选择的表达式指定名字 (别名的个数必须与由视图选择的表达式的个数匹配)
subquery	是一个完整的 SELECT 语句(对于在 SELECT 列表中的字段你可以用别名)
WITH CHECK OPTION	指定只有可访问的行在视图中才能被插入或修改
constraint	为 CHECK OPTION 约束指定的名字
WITH READ ONLY	确保在该视图中没有 DML 操作被执行

幻灯片 8

创建视图

- 创建一个视图， **EMPVU80**，其中包含了在部门 80 中雇员的详细信息

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

- 用 *i*SQL*Plus **DESCRIBE** 命令查看视图的结构

```
DESCRIBE empvu80
```


创建视图 (续)

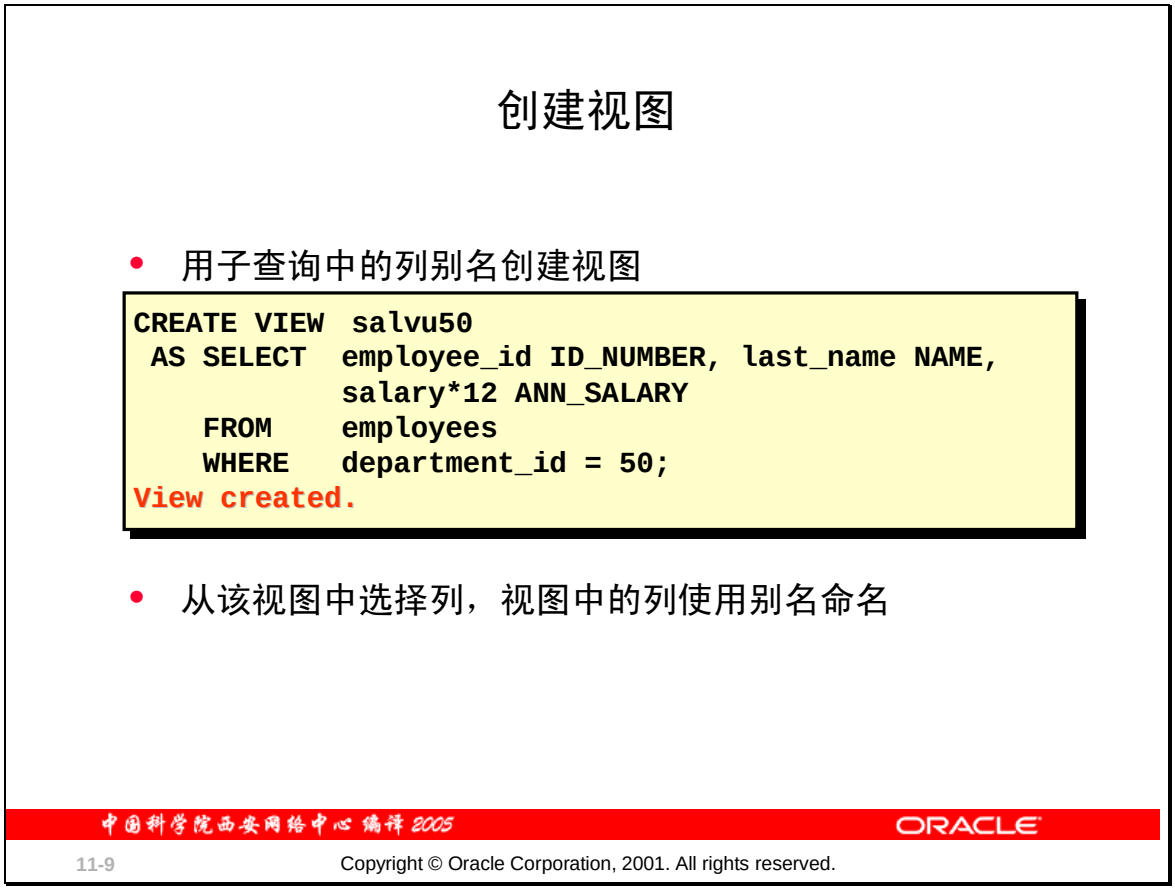
幻灯片上的例子创建一个包含在部门 80 中所有雇员的雇员号、名字和薪水视图。你可以用 *i*SQL*Plus DESCRIBE 命令显示视图的结构。

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

创建视图的原则：

- 定义一个视图的子查询可以包含复杂的 SELECT 语法，包括连分组和多个字查询。
- 定义视图的子查询不能包含 ORDER BY 子句，当你从视图取回数据时可以指定 ORDER BY 子句。
- 如果你没有为用 WITH CHECK OPTION 选项创建的视图指定一个约束名字，系统将以 SYS_C*n* 格式指定一个默认的名字。
- 你可以用 OR REPLACE 选项改变视图的定义而无须删除和重新创建它，或重新授予以前已经授予它的对象权限。

幻灯片 9



创建视图 (续)

你可以在子查询中包括列别名来控制列名。

幻灯片中的例子创建一个视图，包含部门 50 中的每个雇员的雇员号(EMPLOYEE_ID)，别名是 ID_NUMBER；名字(LAST_NAME)，别名是 NAME；年薪(SALARY)，别名是 ANN_SALARY。

作为一个选择，你可以在 CREATE 语句后面在 SELECT 子查询前面用一个别名，被列出的别名的个数必须与在子查询中被选择的表达式相匹配。

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

视图创建。

教师注释

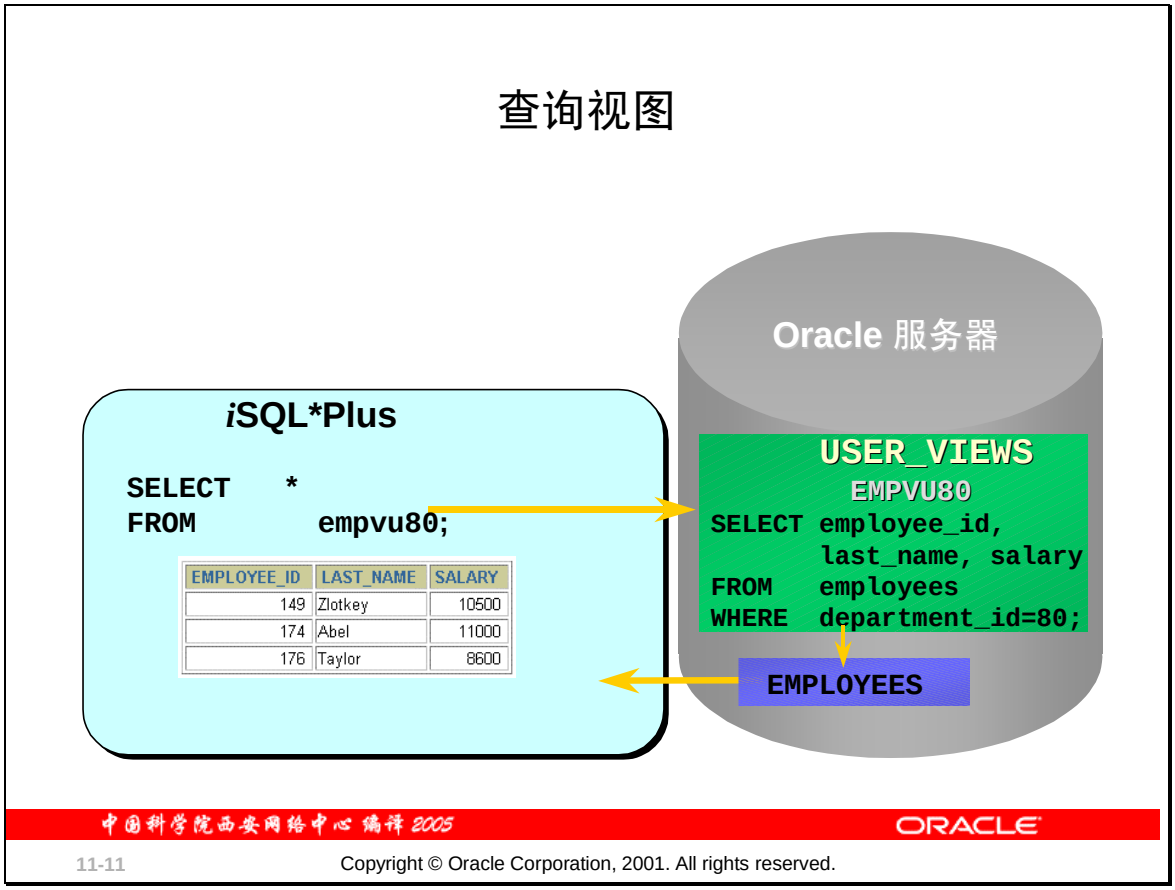
让学生知道关于被物化的视图或快照，术语快照 (*snapshot*) 和 物化视图 (*materialized view*) 是同义词。两者都涉及包含一个或多个表的查询结果的表，它们中的每一个都可能被定位于同一个数据库或者一个远程数据库上。在查询中的表被称为主表或细节表。包含主表的数据库称为主数据库。关于物化视图的更多信息可参考：*Oracle9i SQL Reference*，“创建物化视图/快照”。

幻灯片 10

从视图中取回数据

你可以象从任何表中取回数据一样从视图取回数据，你既可以显示整个视图的内容，也可以仅显示指定的行和列。

幻灯片 11



数据字典中的视图

一旦视图被创建，你就可以查询数据字典视图 USER_VIEWS 来看视图的名字和视图定义。构成视图的 SELECT 语句的文本被存储在一个 LONG 列中。

用视图存取数据

当你用视图存取数据时，Oracle 服务器执行下面的操作：

1. 从数据字典表 USER_VIEWS 中取回视图定义。
2. 检查对视图的基表的数据存取权限。
3. 转换视图查询为一个在基表或表上的等价操作，换句话说，数据从基表得到，或更新基表。

幻灯片 12

修改视图

- 用 **CREATE OR REPLACE VIEW** 子句，为每个列添加一个别名

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

- 在 **CREATE VIEW** 子句中的字段别名列表，按照与子查询中的字段相同的顺序排列

修改视图

用 OR REPLACE 选项，允许创建视图时同名的视图已经存在，这样旧版本的视图会被替换，这意味着视图可以在不被删除、重新创建和再次授予对象权限的情况下修改。

注：当在 CREATE VIEW 子句中指定列别名时，不要望了别名的列表顺序与子查询的列顺序一样。

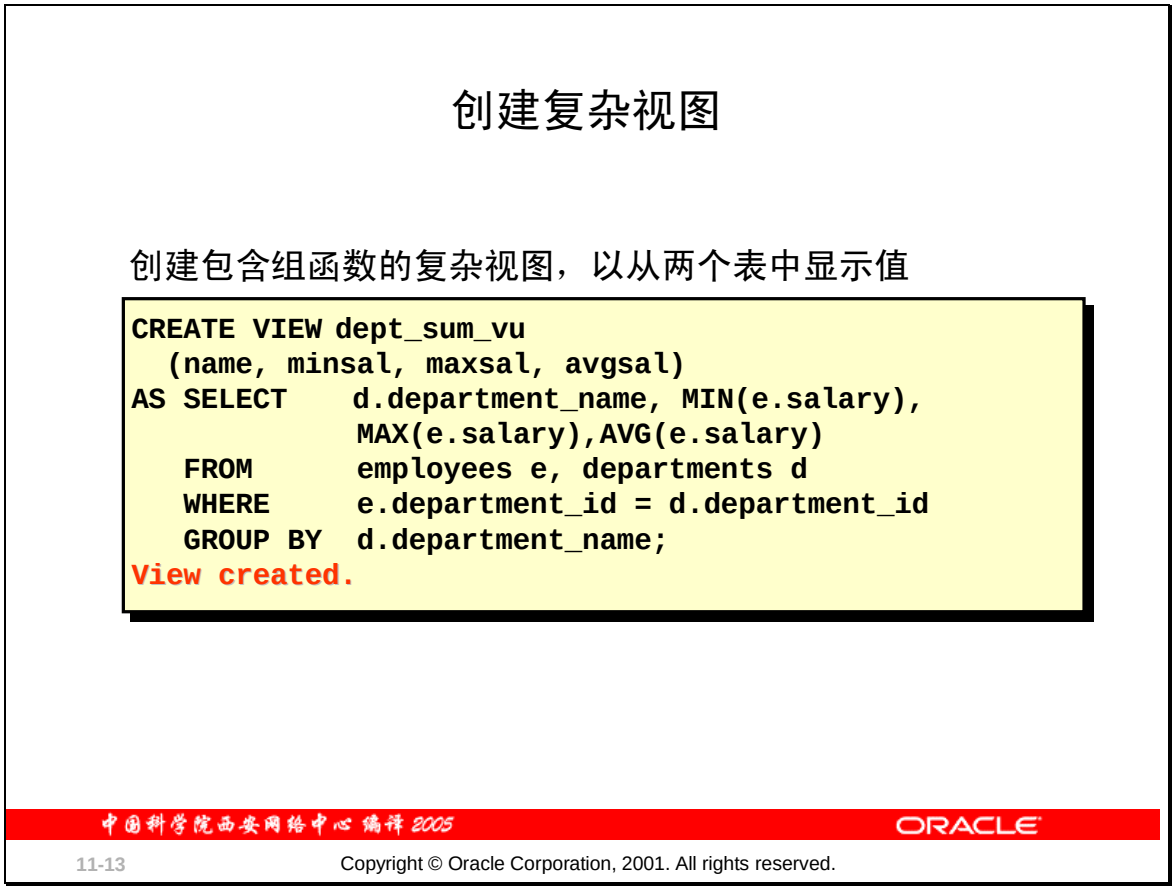
教师注释

OR REPLACE 选项是从 Oracle7 开始有的，用 Oracle 的早期版本，如果要改变视图，先要删除旧视图，再重新创建。

演示：11_emp.sql

目的：举例说明用别名创建视图

幻灯片 13



创建复杂视图

幻灯片的例子创建一个部门名称、最低薪水、最高薪水和部门平均薪水的复杂视图。注意，例子中指定了表别名。如果视图中的任何列源自函数或表达式，别名是必需的。你可以用 *i*SQL*Plus DESCRIBE 命令查看视图的结构，用 SELECT 语句显示视图的内容。

```
SELECT  *
FROM    dept_sum_vu;
```

NAME	MINSAL	MAXSAL	AVGSAL
Accounting	8300	12000	10150
Administration	4400	4400	4400
Executive	17000	24000	19333.3333
IT	4200	9000	6400
Marketing	6000	13000	9500
Sales	8600	11000	10033.3333
Shipping	2500	5800	3500

7 rows selected.

幻灯片 14

在视图中执行 DML 操作

你可以通过视图对数据进行执行 DML 操作，但那些操作必须符合下面的规则。如果视图中不包含下面的部分，你就可以从视图中删除数据：

- 组函数
- GROUP BY 子句
- DISTINCT 关键字
- 伪列 ROWNUM 关键字

教师注释

对于查询返回的每一个行，ROWNUM 伪列返回一个数，该数指示 Oracle 服务器从表或连接集合中选择的行的顺序。选择的首行 ROWNUM 是 1，第二行是 2，等等。

幻灯片 15

在视图中执行 DML 操作 (续)

如果视图中不包含幻灯片中提到的情况，或者不包含由表达式定义的列，例如，SALARY * 12，你就可以通过视图修改数据。

幻灯片 16

在视图中执行 DML 操作 (续)

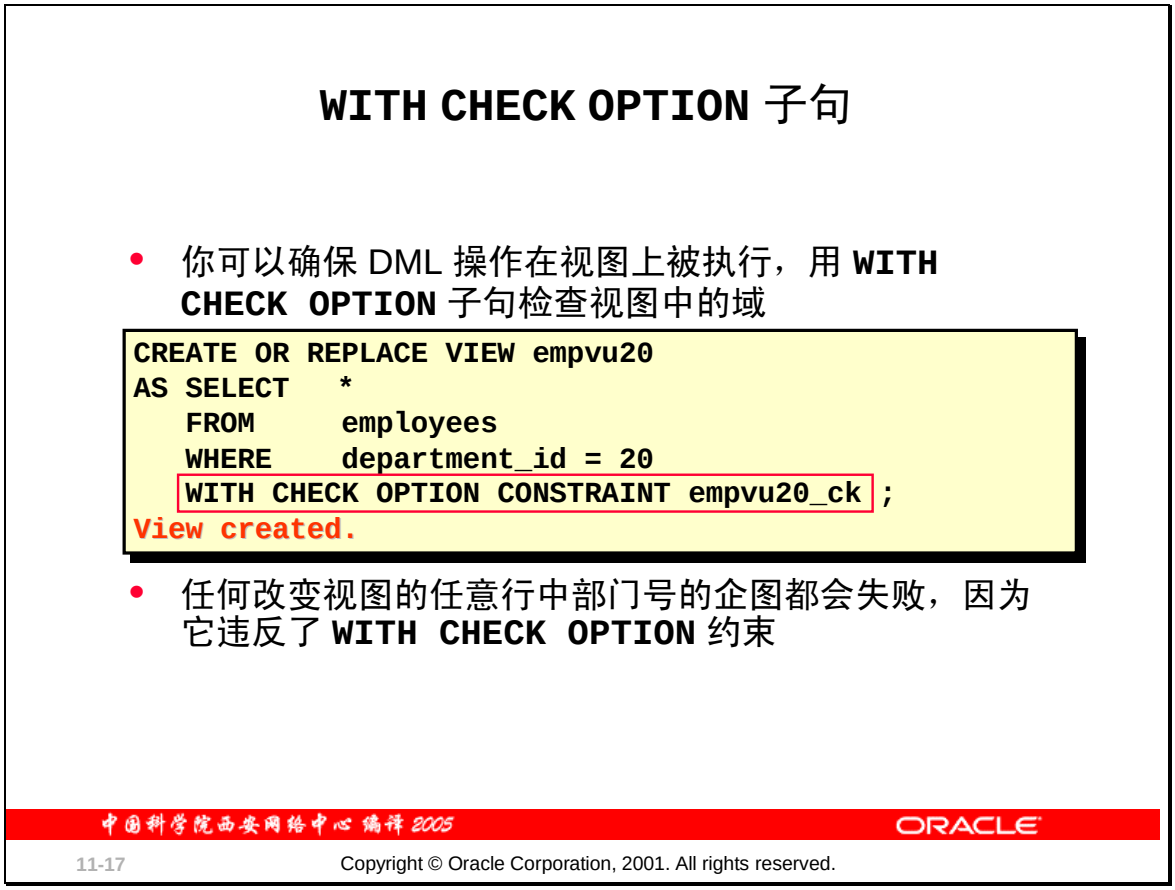
如果视图中不包含幻灯片中列出的项目，或者在没有被视图选择的基表中有没有默认值的 NOT NULL 列，你就可以通过视图添加数据。所有的值必须在视图中出现。不要忘记你正在通过视图直接添加值到基表中。

更多信息见 *0racle9i SQL Reference*，“创建视图”。

教师注释

用 Oracle7.3 和以后版本，你可以修改包括带一些连接的视图。在幻灯片中描述的对 DML 操作的约束也将应用于连接视图。在一个连接视图上，只有在一个基表上，任何 UPDATE、INSERT 或 DELETE 语句才能够执行。如果在子查询连接中至少有一行有一个唯一索引，那么在一个连接视图中修改基表是可能的。你可以查询 USER_UPDATABLE_COLUMNS 来看是否在连接视图中的列可以被更新。

幻灯片 17



使用 WITH CHECK OPTION 子句

可以通过视图执行引用完整性检查，你也可以在数据库级别强制约束。视图能用于保护数据的完整性，但用途非常有限。

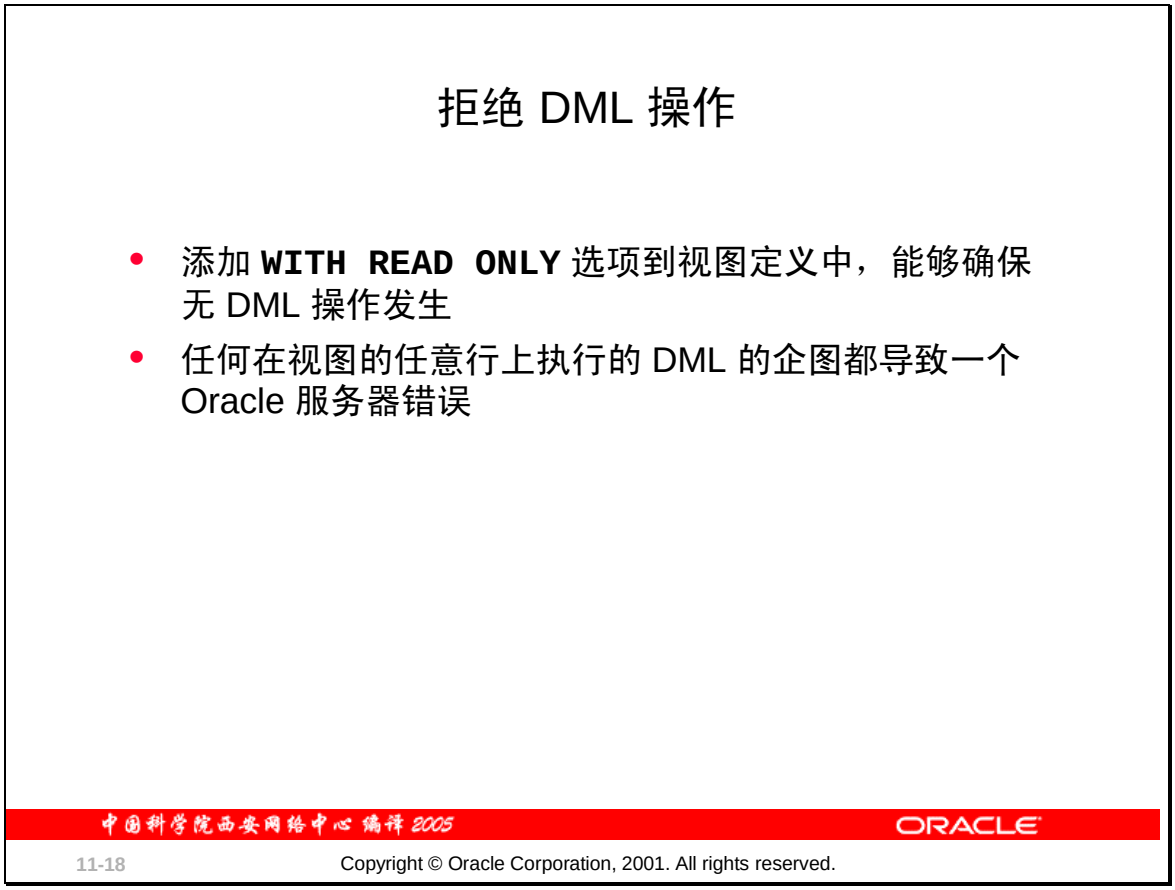
WITH CHECK OPTION 子句指出通过视图执行的 INSERTs 和 UPDATEs 不能创建视图不能选择的行，因此，该子句在数据开始插入或更新时允许完整性约束和数据验证检查。

如果在没有选择的行上有一个执行 DML 操作的尝试，将会显示错误，如果指定了约束名，会带在错误提示中。

```
UPDATE empvu20
  SET    department_id = 10
  WHERE  employee_id = 201;
UPDATE empvu20
       *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

注：没有行被更新，因为如果部门号变成 10，该视图将不再能够看到雇员，因此，带 WITH CHECK OPTION 子句，视图只能看到部门 20 中的雇员，并且不允许通过视图改变那些看不到的部门中的雇员。

幻灯片 18



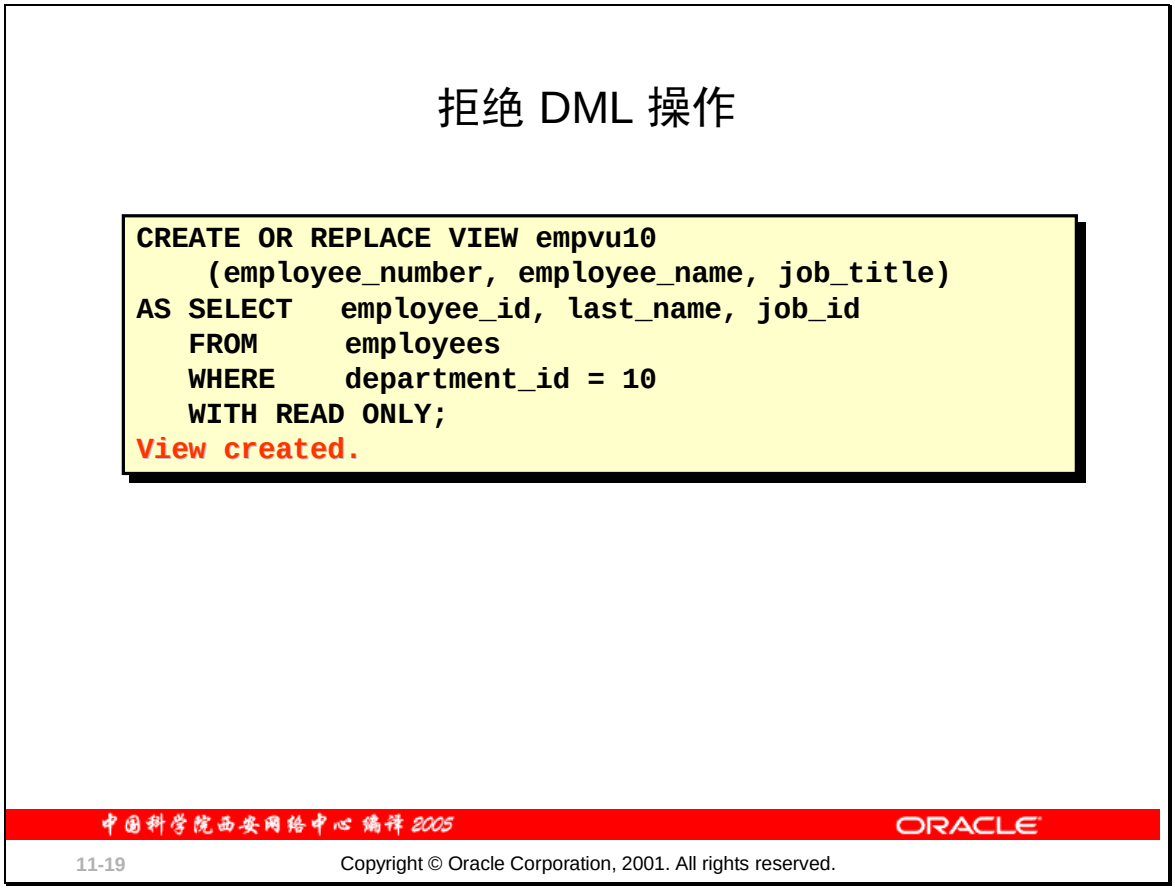
拒绝 DML 操作

你可在创建视图时带上 WITH READ ONLY 选项，以确保无 DML 操作发生。幻灯片中的例子修改 EMPVU10 视图，以防止任何在视图上进行的 DML 操作。

教师注释 (11-17 页)

如果用户没有提供约束名，系统将以 SYS_C*n* 格式指定一个名字，这里 *n* 是一个整数，在系统中约束名是唯一的。

幻灯片 19



拒绝 DML 操作

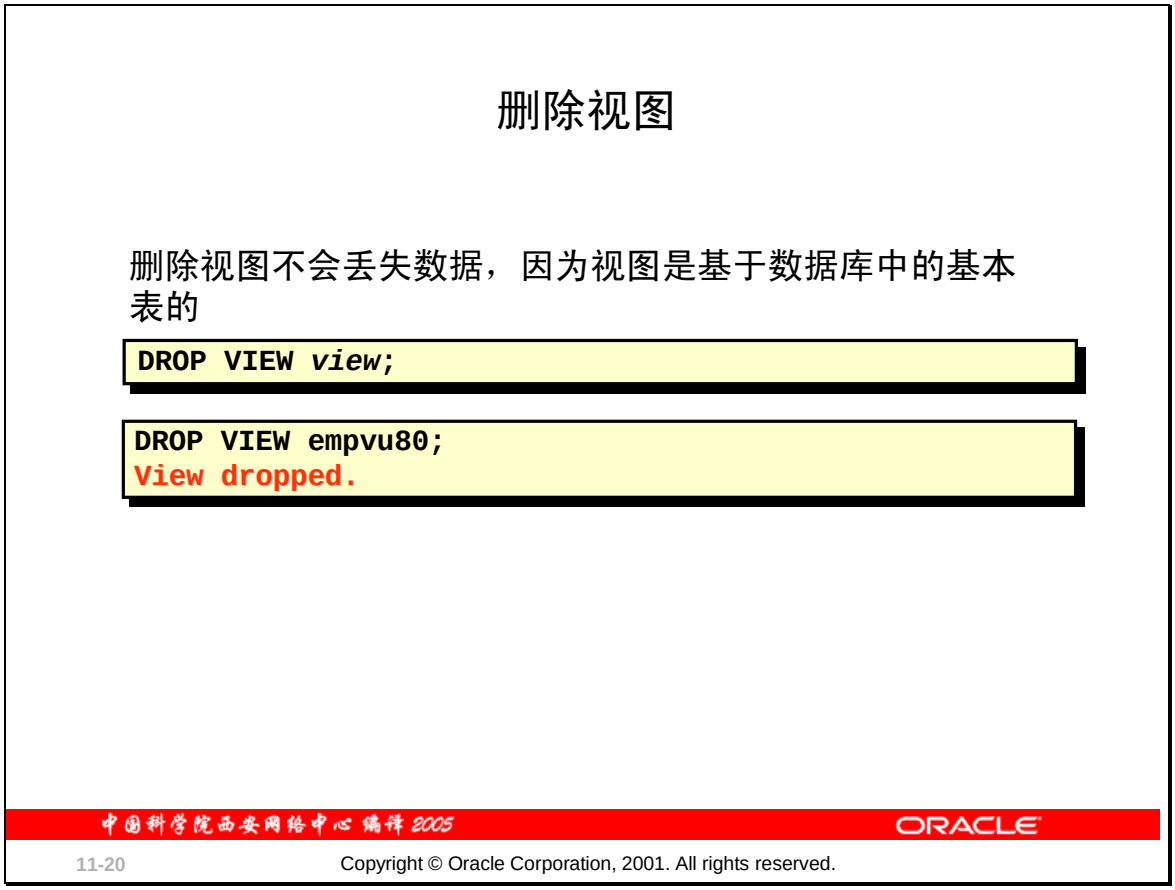
从带只读约束的视图中做任何删除行的尝试只会得到错误的结果。

```
DELETE FROM empvu10
WHERE  employee_number = 200;
DELETE FROM empvu10
        *
ERROR at line 1:
ORA-01752: cannot delete from view without exactly one key-
preserved table
```

任何对带只读约束的视图进行的插入或修改行的尝试，在 Oracle 服务器中都显示下面的错误结果：

01733: virtual column not allowed here.

幻灯片 20



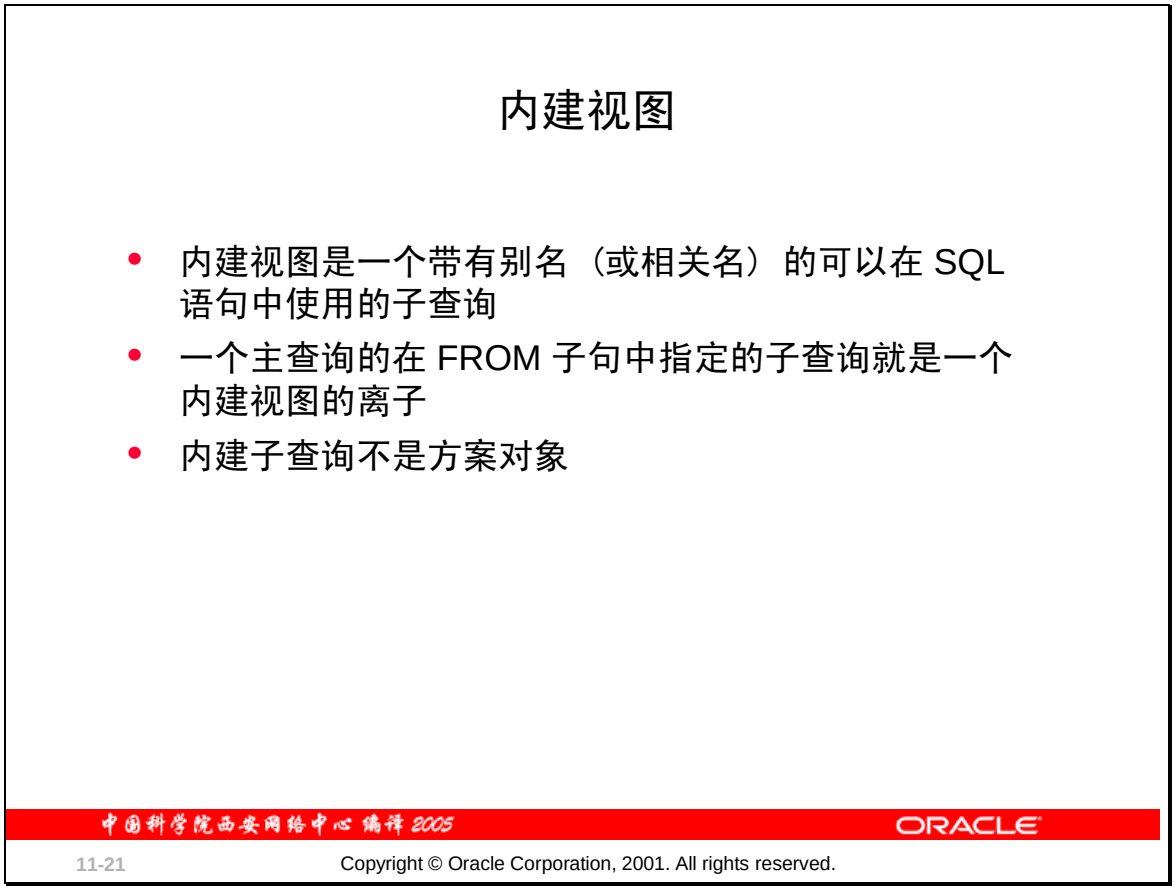
删除视图

你可以用 DROP VIEW 语句来删除视图。该语句从数据库中删除视图定义。删除视图不影响用于建立视图的基表。基于已删除视图上的其它视图或应用程序将无效。只有创建者或具有 DROP ANY VIEW 权限的用户才能删除视图。

在语法中：

view 是视图的名字

幻灯片 21



内建视图

内建视图由位于 FROM 子句中命名了别名的字查询创建。该字查询定义一个可以在主查询中引用数据源。在下面的例子中，内建视图 b 从 EMPLOYEES 表中返回每一个部门的部门号和最高薪水。主查询显示雇员的名字、薪水、部门号和该部门的最高薪水，WHERE a.department_id = b.department_id AND a.salary < b.maxsal 子句取回所有收入少于该部门最高薪水的雇员。

```
SELECT  a.last_name,a.salary,a.department_id,b.maxsal
FROM    employees a,(SELECT department_id,max(salary) maxsal
                     FROM    employees
                     GROUP BY department_id) b
WHERE   a.department_id = b.department_id
AND     a.salary < b.maxsal;
```

LAST_NAME	SALARY	DEPARTMENT_ID	MAXSAL
Fay	6000	20	13000
Rajs	3500	50	5800
Davies	3100	50	5800
Matos	2600	50	5800
Gietz	8300	110	12000

12 rows selected.

幻灯片 22

Top-N 分析

- Top-N 查询寻找一列的 *n* 个最大或最小值，例如：
  - 销售最好的前 10 位产品是什么?
  - 销售最差的前 10 位产品是什么?
- 最大值和最小值在 Top-N 查询中设置

“Top-N”分析

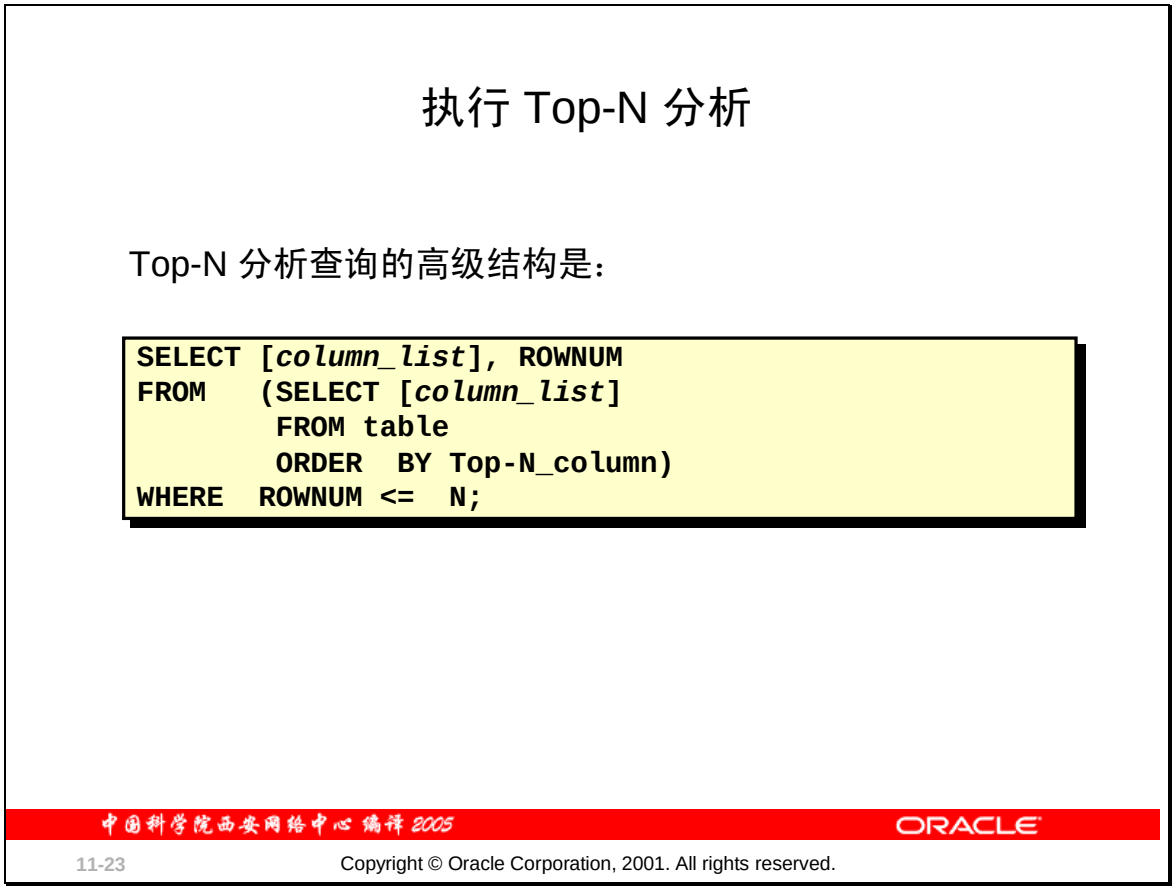
Top-N 查询在需要基于一个条件，从表中显示最前面的 n 条记录或最后面的 n 条记录时是有用的。该结果可以用于进一步分析，例如，用 Top-N 分析你可以执行下面的查询类型：

- 在中挣钱最多的三个人
- 公司中最新的四个成员
- 销售产品最多的两个销售代表
- 过去 6 个月中销售最好的 3 种产品

教师注释

在子查询中包含 ORDER BY 子句也可以进行 Top-N 分析。

幻灯片 23

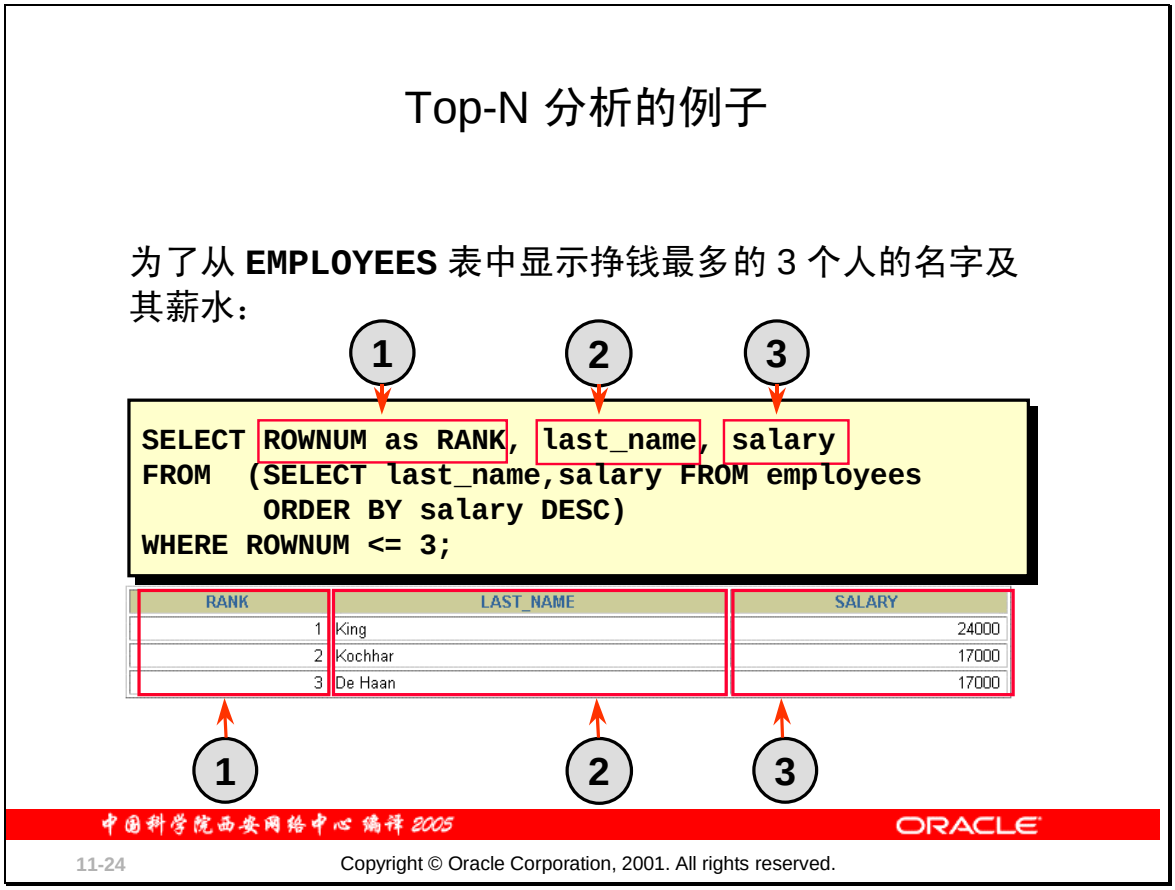


执行“Top-N”分析

Top-N 查询使用一个带有下面描述的元素的一致的嵌套查询结构：

- 子查询或者内建视图产生数据的排序列表，该子查询或者内建视图包含 ORDER BY 子句来确保排序以想要的顺序排列。为了取回最大值，需要用 DESC 参数。
- 在最后的结果集中用外查询限制行数。外查询包括下面的组成部分：
  - ROWNUM 伪列，它为从子查询返回的每一行指定一个从 1 开始的连续的值
  - 一个 WHERE 子句，它指定被返回的 *n* 行，外 WHERE 子句必须用一个<或者<=操作。

幻灯片 24



“Top-N”分析的例子

幻灯片中的例子举例说明怎样从 EMPLOYEES 表中显示挣钱最多的 3 个人的名字和薪水。子查询从 EMPLOYEES 表中返回所有所有雇员的名字和薪水的详细资料，以逆序排序薪水。主查询的 WHERE ROWNUM < 3 子句确保只有在结果集中的前 3 条记录被显示。

下面是用内建视图进行 Top-N 分析的另一个例子，该例子用内建视图 E 显示公司中 4 个资格最老的雇员。

```
SELECT ROWNUM as SENIOR,E.last_name, E.hire_date
FROM  (SELECT last_name,hire_date FROM employees
       ORDER BY hire_date)E
WHERE rownum <= 4;
```

SENIOR	LAST_NAME	HIRE_DATE
1	King	17-JUN-87
2	Whalen	17-SEP-87
3	Kochhar	21-SEP-89
4	Hunold	03-JAN-90

幻灯片 25

小结

在本课中，您应该已经学会视图来自其它表或视图中的数据，视图有下面的好处：

- 限制数据库访问 Restricts database access
- 简化查询
- 提供数据的独立性
- 提供相同数据的多种视图
- 能够被删除而不破坏底层数据
- 内建视图是一个带有别名的子查询
- 用子查询和外查询能够进行 Top-N 分析

什么是视图?

一个视图基于一个表或者另一个视图，其作用好象是一个窗口，通过该窗口表中的数据能够被观察或修改。视图不包含数据，视图的定义被存储在数据字典中，你可以在 USER_VIEWS 数据字典表中查看视图的定义。

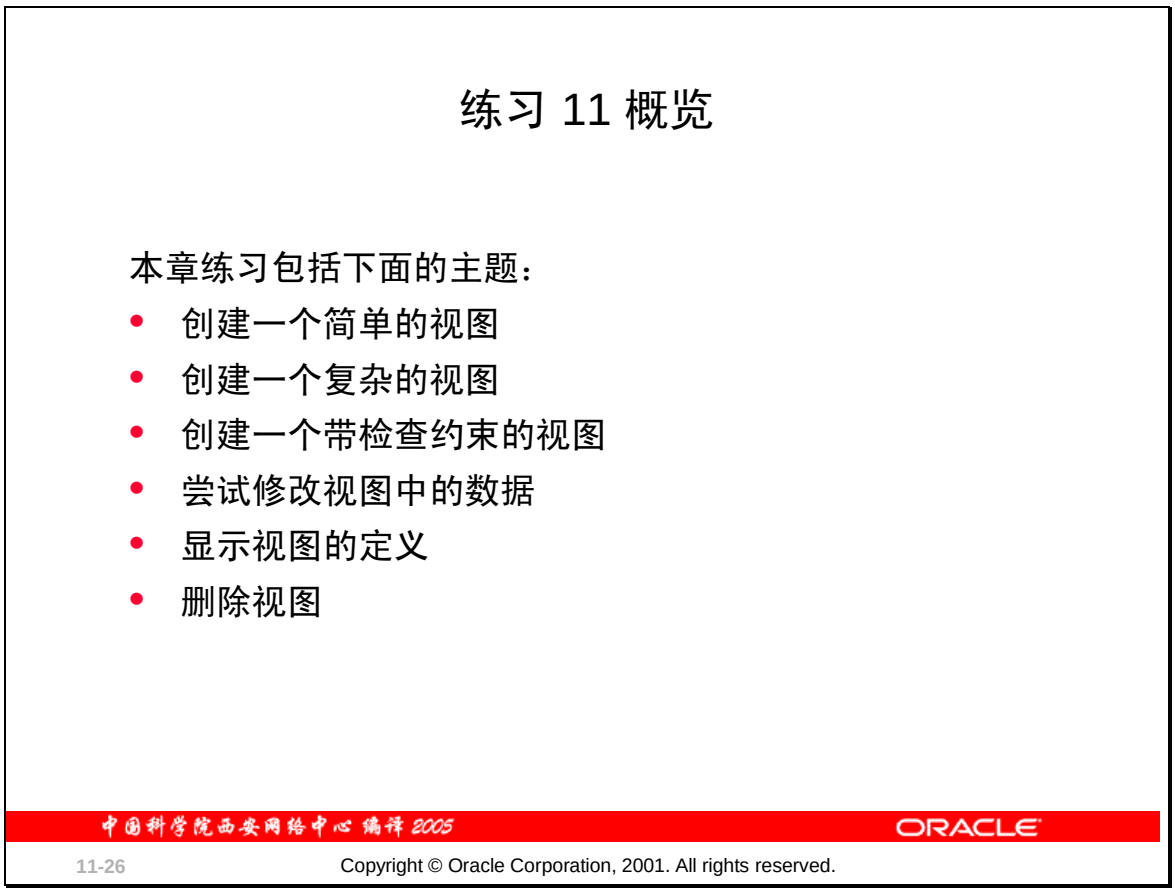
视图的优点

- 限制数据访问
- 简化查询
- 提供数据独立性
- 提供相同数据的多种视图
- 能够在不影响底层数据的情况下被删除

视图选项

- 可以是基于一个表的简单视图
- 可以是基于多表或包含函数组的复杂视图
- 可以用相同的名字替换其它视图
- 可以包含检查约束
- 可以是只读的

幻灯片 26



练习 11 概览

在该练习中，你将创建简单的和复杂的视图，并且在视图中尝试执行 DML 语句。

幻灯片 27

练习 11

1. 创建一个称为 EMPLOYEES_VU 的视图，它基于 EMPLOYEES 表中的雇员号、雇员名和部门号。将雇员名的列标题改为 EMPLOYEE。

```
CREATE OR REPLACE VIEW employees_vu AS
SELECT employee_id, last_name employee, department_id
FROM employees;
```

2. 显示 EMPLOYEES_VU 视图的内容。

EMPLOYEE_ID	EMPLOYEE	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60
206	Gietz	110

20 rows selected.

```
SELECT *
FROM employees_vu;
```

3. 从 USER_VIEWS 数据字典视图中选择视图名字和文本。

注：另一个视图已经存在，EMP_DETAILS_VIEW 作为你的方案被创建。

注：为了看见一个 LONG 列的更多的内容，用 *i*SQL*Plus 命令 SET LONG *n*，在这儿 *n* 是你想要看到的 LONG 列的字符的个数。

VIEW_NAME	TEXT
EMPLOYEES_VU	SELECT employee_id, last_name employee, department_id FROM employees
EMP_DETAILS_VIEW	SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.country_id, e.first_name, e.last_name, e.salary, e.commission_pct, d.department_name, j.job_title, l.city, l.state_province, c.country_name, r.region_name FROM employees e, departments d, jobs j, locations l, countries c, regions r WHERE e.department_id = d.department_id AND d.location_id = l.location_id AND l.country_id = c.country_id AND c.region_id = r.region_id AND j.job_id = e.job_id WITH READ ONLY

```
SET LONG 600
SELECT view_name, text
FROM user_views;
```

4. 使用 EMPLOYEES_VU 视图，输入一个查询来显示所有的雇员名和部门号。

EMPLOYEE	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Gietz	110

20 rows selected.

```
SELECT employee, department_id
FROM employees_vu;
```

5. 创建一个名为 DEPT50 视图，其中包含部门 50 中的所有雇员的雇员号、雇员名和部门号，视图的列标签为 EMPNO、EMPLOYEE 和 DEPTNO。不允许通过视图将一个雇员重新分配到另一个部门。

```
CREATE VIEW dept50 AS
SELECT employee_id empno, last_name employee,
department_id deptno
FROM employees
WHERE department_id = 50
WITH CHECK OPTION CONSTRAINT emp_dept_50;
```

6. 显示 DEPT50 视图的结构和内容。

Name	Null?	Type
EMPNO	NOT NULL	NUMBER(6)
EMPLOYEE	NOT NULL	VARCHAR2(25)
DEPTNO		NUMBER(4)

EMPNO	EMPLOYEE	DEPTNO
124	Mourgos	50
141	Rajs	50
142	Davies	50
143	Matos	50
144	Vargas	50

```
DESCRIBE dept50
SELECT *
FROM dept50;
```

7. 试图重新指定 Matos 到部门 80。

```
UPDATE dept50
SET deptno = 80
```

```
WHERE employee = 'Matos';
```

如果你还有时间，完成下面的练习：

8. 创建一个名为 SALARY_VU 的视图，该视图基于所有雇员的名字、部门名、薪水和薪水级别。用 EMPLOYEES、DEPARTMENTS 和 JOB_GRADES 表，分别命名列标签为 Employee、Department、Salary 和 Grade。

```
CREATE OR REPLACE VIEW salary_vu
AS
SELECT e.last_name "Employee",
d.department_name "Department",
e.salary "Salary",
j.grade_level "Grades"
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;
```