幻灯片 1

进度表：	时间	主题
	45 分钟	讲演
	25 分钟	练习
	70 分钟	总共

幻灯片 2

课程目标

在本课中，你将学习怎样用内置的完整性约束实现商业规则。

幻灯片 3

约束

Oracle 服务器用约束 *(constraints)* 来防止无效数据输入到表中。

你可以使用约束做下面的事：

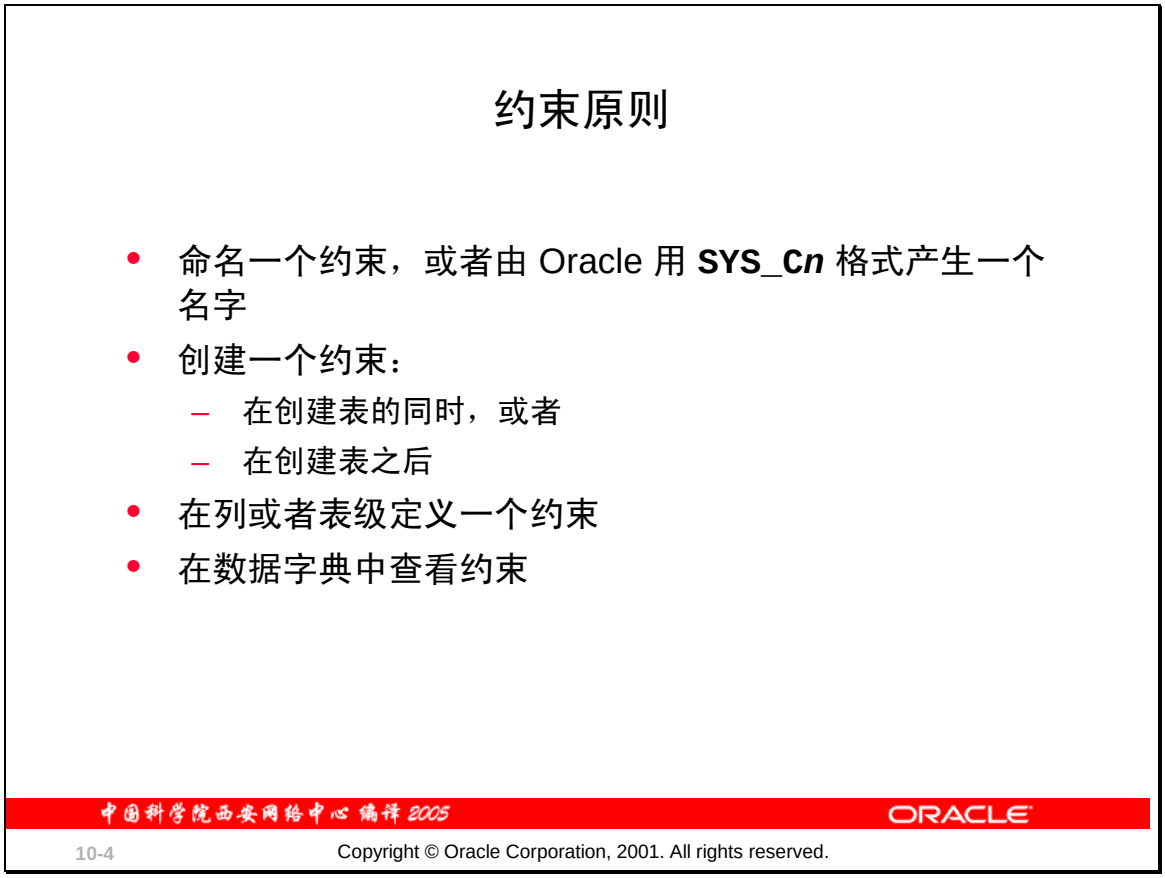
- 在插入、更新行或者从表中删除行的时候强制表中的数据遵循规则。对于成功的操作，约束必须被满足
- 如果表之间有依赖关系，防止表的删除
- 为 Oracle 工具提供规则，例如 Oracle Developer

数据一致性约束

约束	说明
NOT NULL	指定列不能包含空值
UNIQUE	指定列的值或者列的组合的值对于表中所有的行必须是唯一的
PRIMARY KEY	表的每行的唯一性标识
FOREIGN KEY	在列和引用表的一个列之间建立并且强制一个外键关系
CHECK	指定一个必须为真的条件

更多的信息，见 *Oracle9i SQL Reference*，“约束”。

幻灯片 4



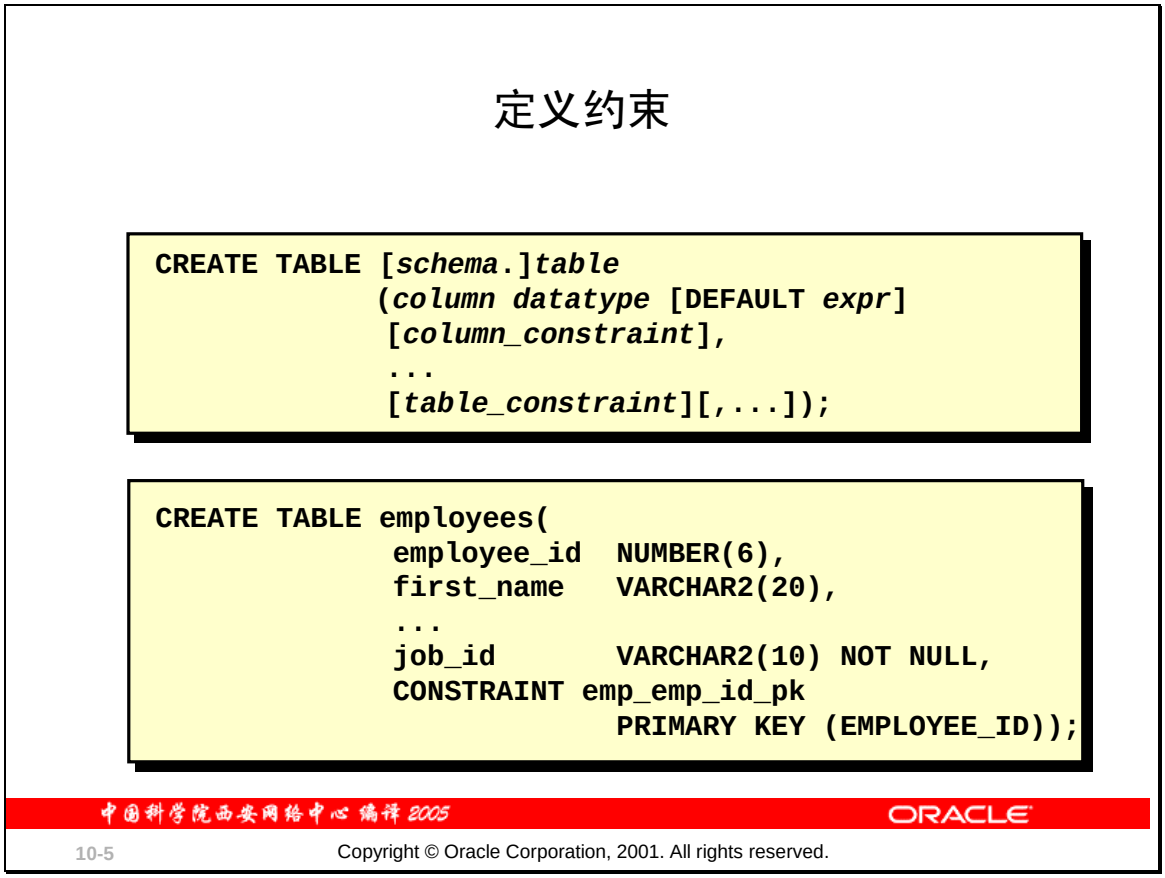
约束原则

所有的约束存储在数据字典中。如果给约束一个有意义的名字，约束易于引用，约束命名必须遵守标准的对象命名规则。如果你不命名你的约束，Oracle 服务器将用格式 SYS_C*n* 产生一个名字，这里 *n* 是一个唯一的整数，所以约束名是唯一的。

约束可以在创建表时定义，也可以在表创建之后定义。

你可以用 USER_CONSTRAINTS 数据字典表查看对一个表的约束的定义。

幻灯片 5



定义约束

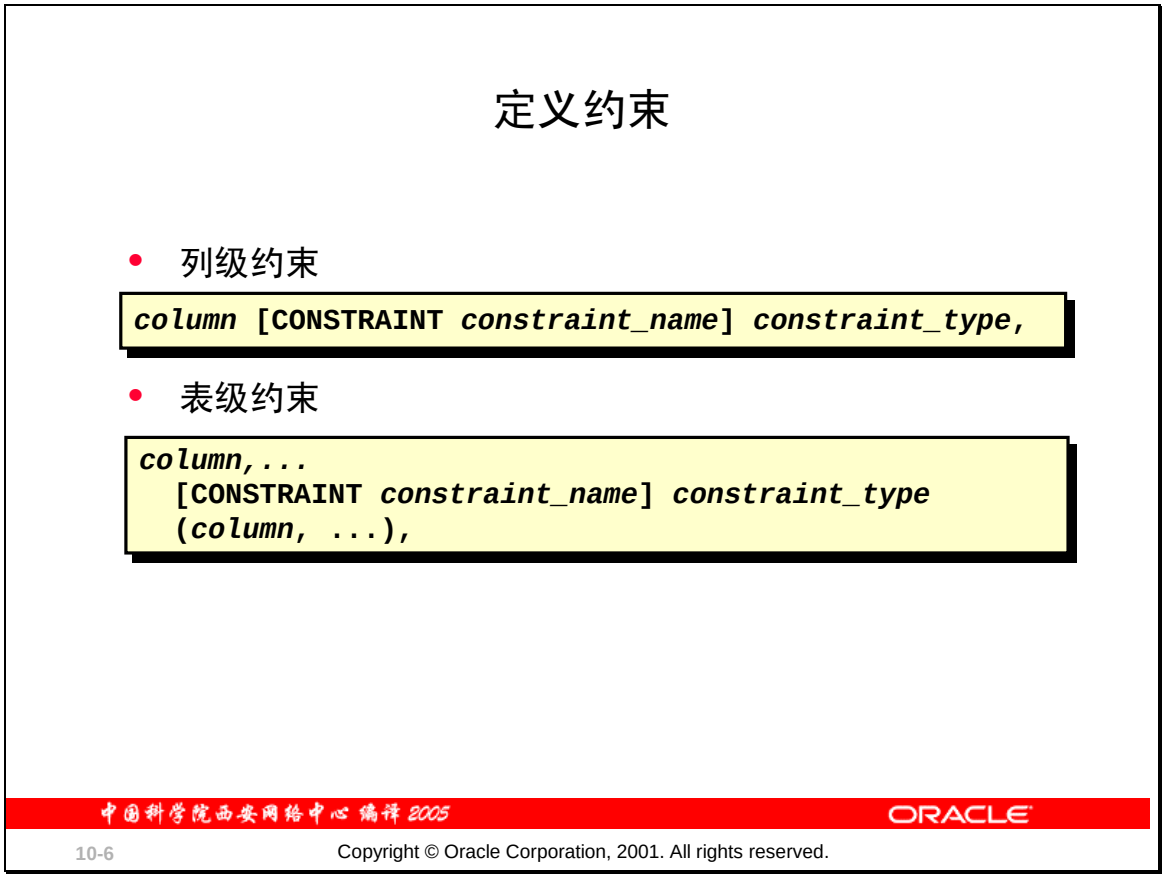
幻灯片给出了在创建表的同时定义约束的语法。

在语法中：

schema	与所有者同名
table	表的名字
DEFAULT *expr*	指定一个默认值。如果在插入语句中省略了一个值，在省略处使用该默认值
column	列的名字
datatype	列的数据类型和长度
column_constraint	是一个作为列定义一部分的完整性约束
table_constraint	是一个作为表定义一部分的完整性约束

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

幻灯片 6



定义约束 (续)

约束通常在创建表的同时被创建。在表创建后约束能够被添加，并且约束可以可以被临时禁用。

约束可以在两个级别上定义。

约束级别	说明
列	只涉及一个单个的列，对于该列用规范定义；能够定义完整性约束的任何类型
表	涉及一个或多个列，表中的列被分别定义；除了 NOT NULL，能够定义任意约束

在语法中：

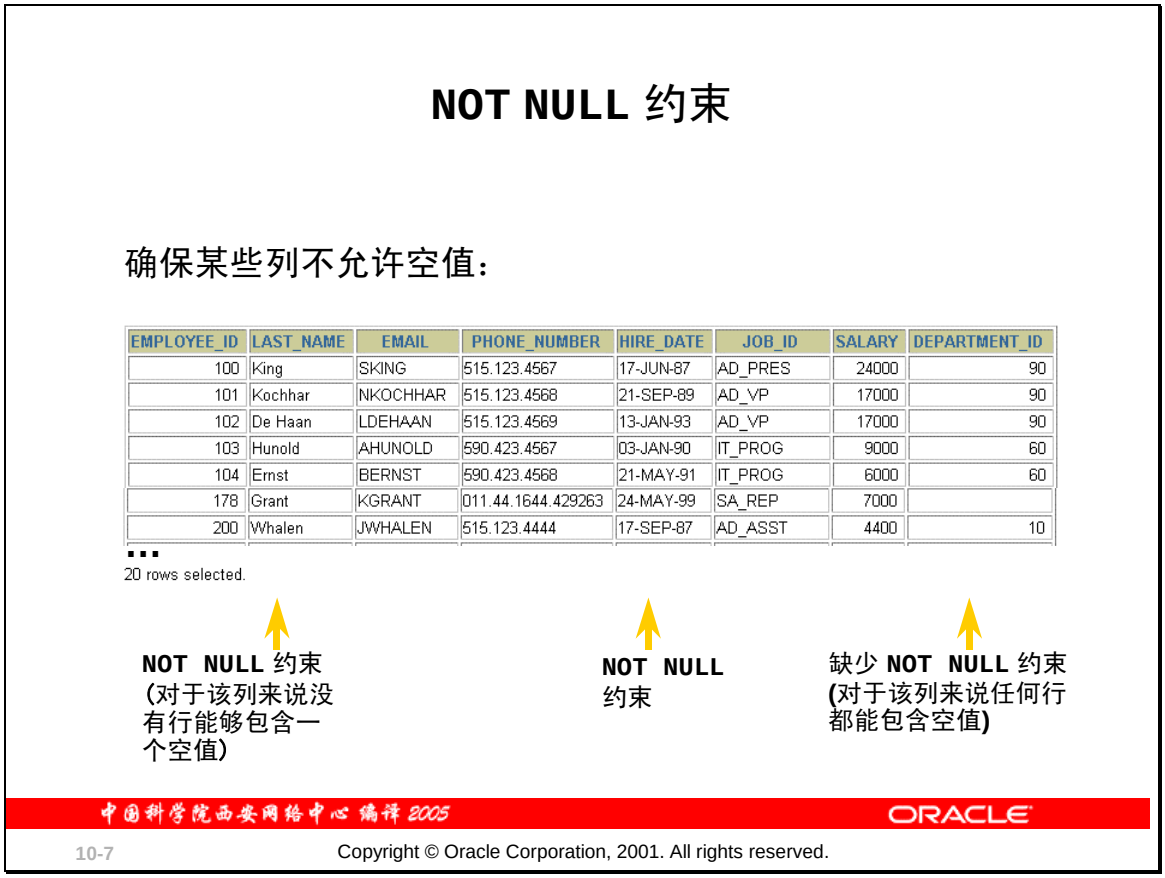
constraint_name 是约束的名字

constraint_type 是约束的类型

教师注释

解释在语法中的列级和表级。

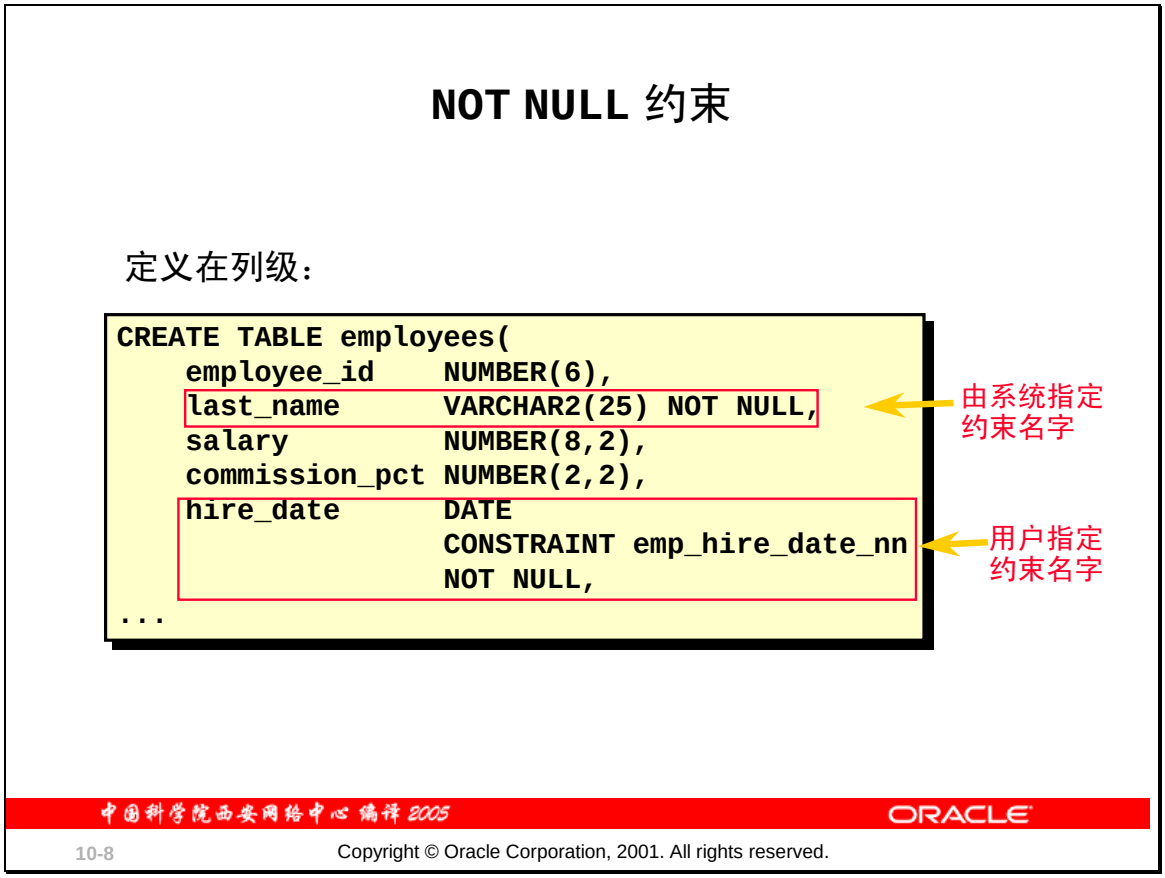
幻灯片 7



NOT NULL 约束

NOT NULL 约束确保列无不包含空值。在默认情况下，列没有 NOT NULL 约束，可以包含空值。

幻灯片 8



NOT NULL 约束 (续)

NOT NULL 约束只能在列级被指定，不能在表级。

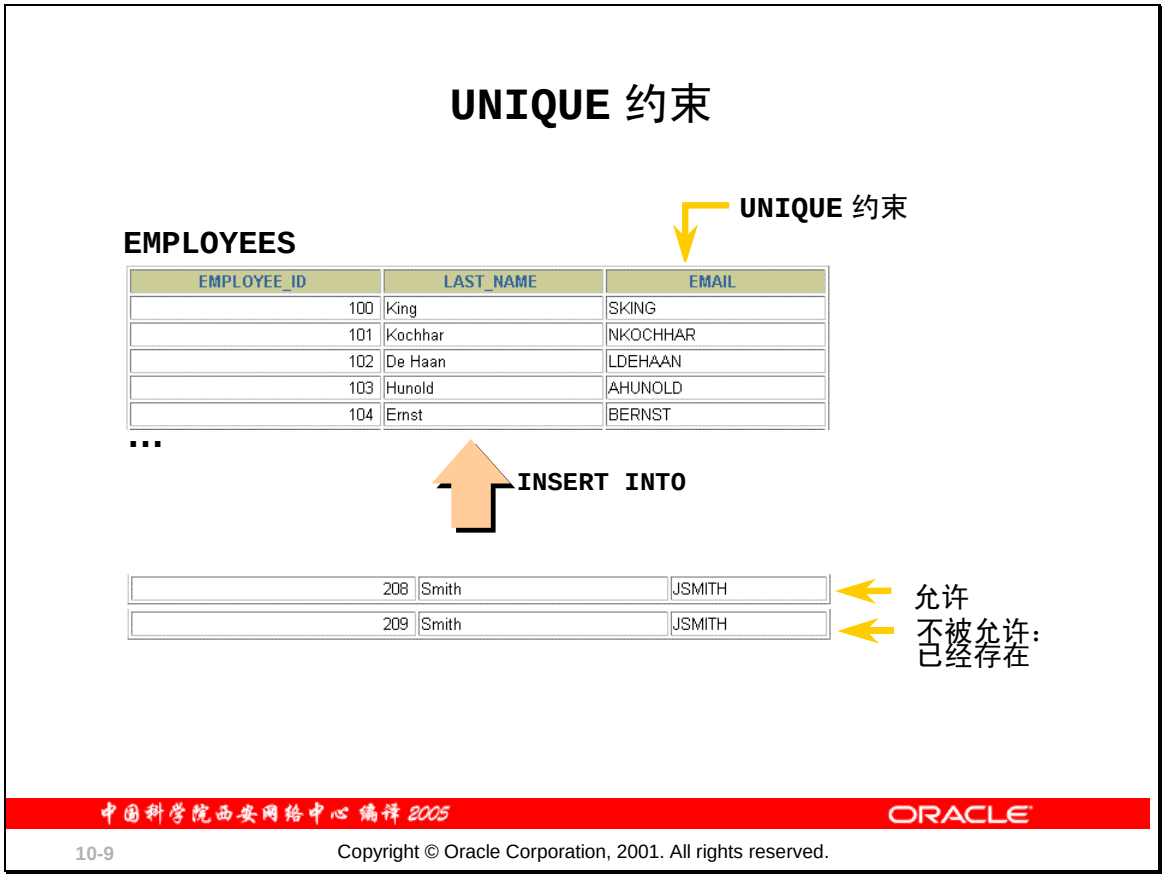
幻灯片的例子应用 NOT NULL 约束到 EMPLOYEES 表的 LAST_NAME 和 HIRE_DATE 列。因为对列 LAST_NAME 的约束未被命名，Oracle 服务器将为它创建名字。

在你指定约束时你可以指定约束的名字：

```
... last_name VARCHAR2(25)
     CONSTRAINT emp_last_name_nn NOT NULL...
```

注：在本课中讲述的约束的例子可能没有出现在随课程提供的表中，如果需要，这些约束可以被加到表中。

幻灯片 9



UNIQUE 约束

UNIQUE 键完整性约束，要求列或者列的组合中（键）的每个值是唯一的，既，在表中指定的列或列组合中不能有两行有相同的值。定义 UNIQUE 键约束的列（或列组合）被称为唯一键（*unique key*）。

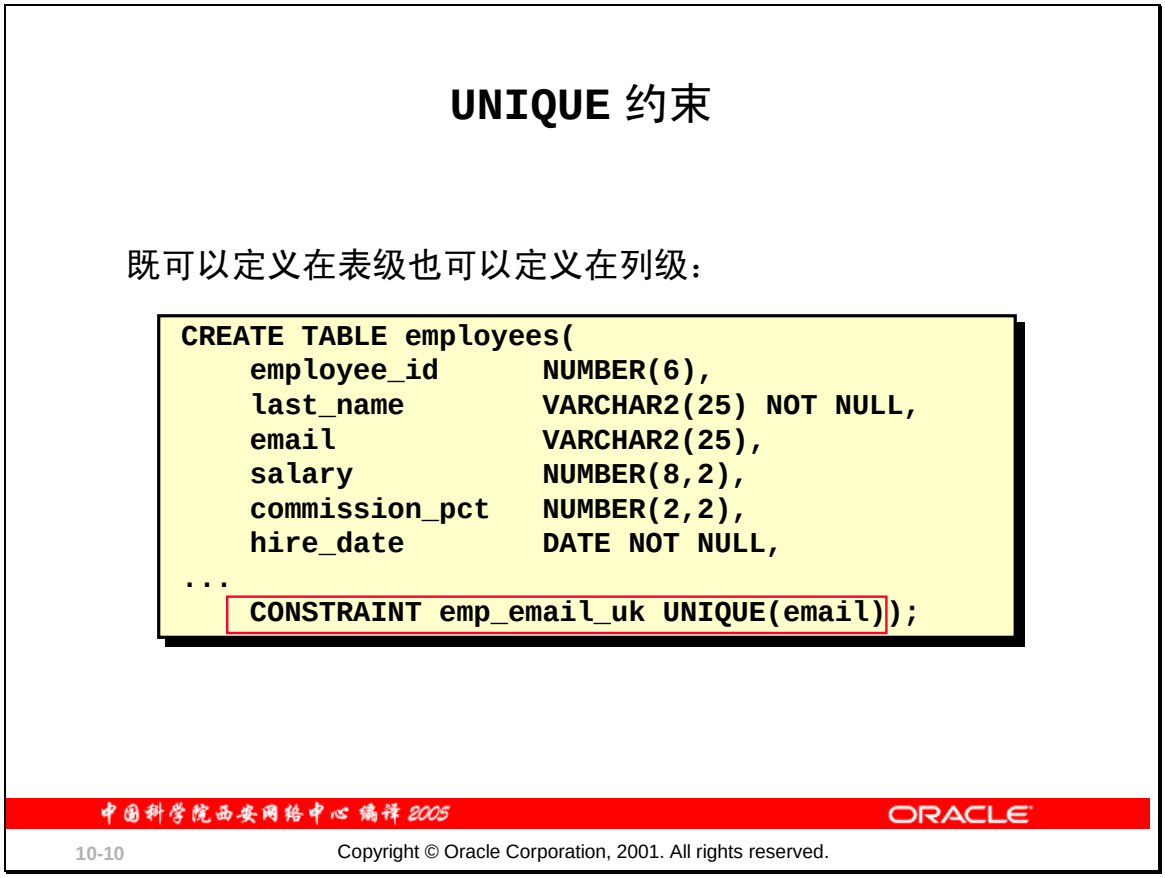
除非你对相同的列也定义了 NOT NULL 约束，UNIQUE 约束允许输入空值，事实上，对于无 NOT NULL 约束的列，能包含空值的行可以是任意数目，因为空不等于任何事。在一个列（或者在一个复合 UNIQUE 键中的所有列）中的空总是满足 UNIQUE 约束。

注：因为在多于一列上的 UNIQUE 约束的搜索机制原因，在一个部分为空的组合 UNIQUE 键约束的非空列中你不能有相同的值。

教师注释

给学生解释由于 JSMITH e-mail ID 在第一次插入后已经存在，不允许第二次输入。

幻灯片 10



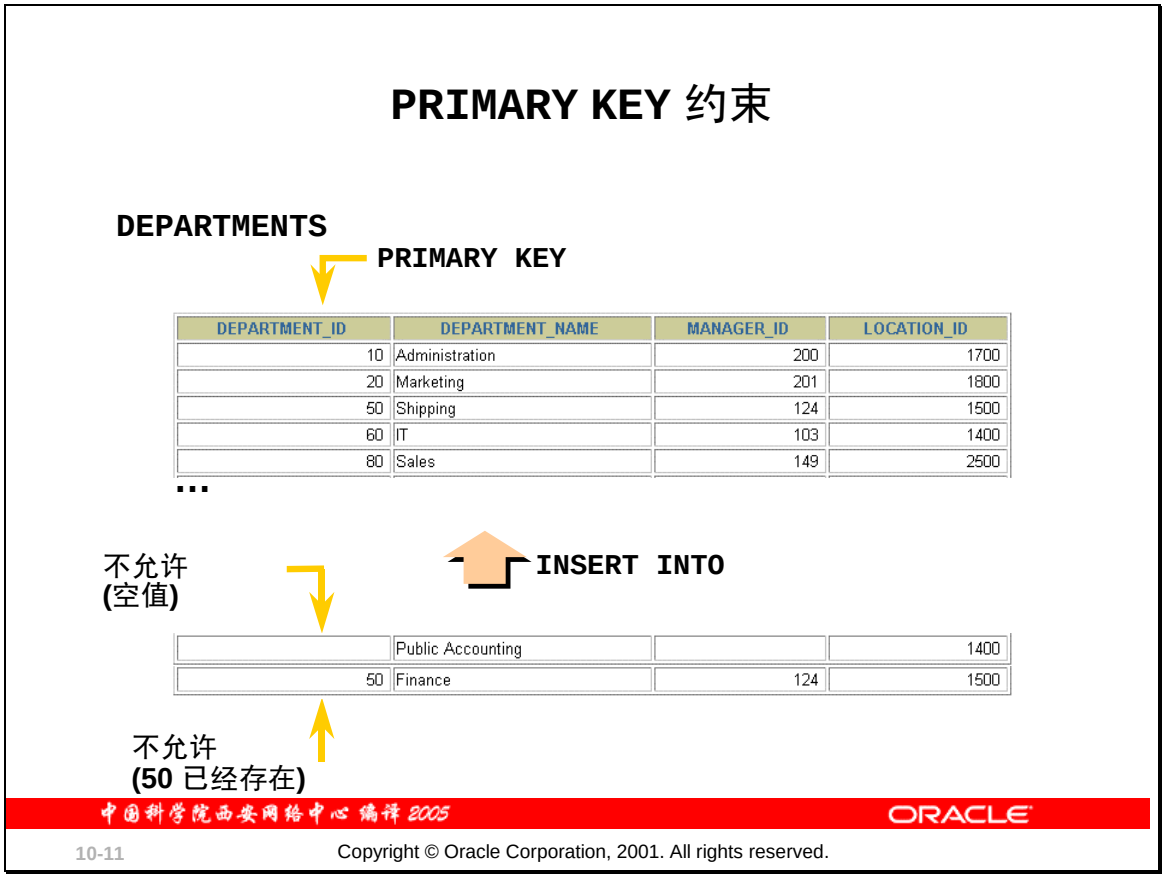
UNIQUE 约束 (续)

UNIQUE 约束既可以在列级也可以在表级定义。使用表级定义时一个复合唯一键被创建。

幻灯片的例子应用 UNIQUE 约束到 EMPLOYEES 表的 EMAIL 列。约束的名字是 EMP_EMAIL_UK。

注：Oracle 服务器在唯一键列或组合列上隐式地创建一个唯一索引强制 UNIQUE 约束。

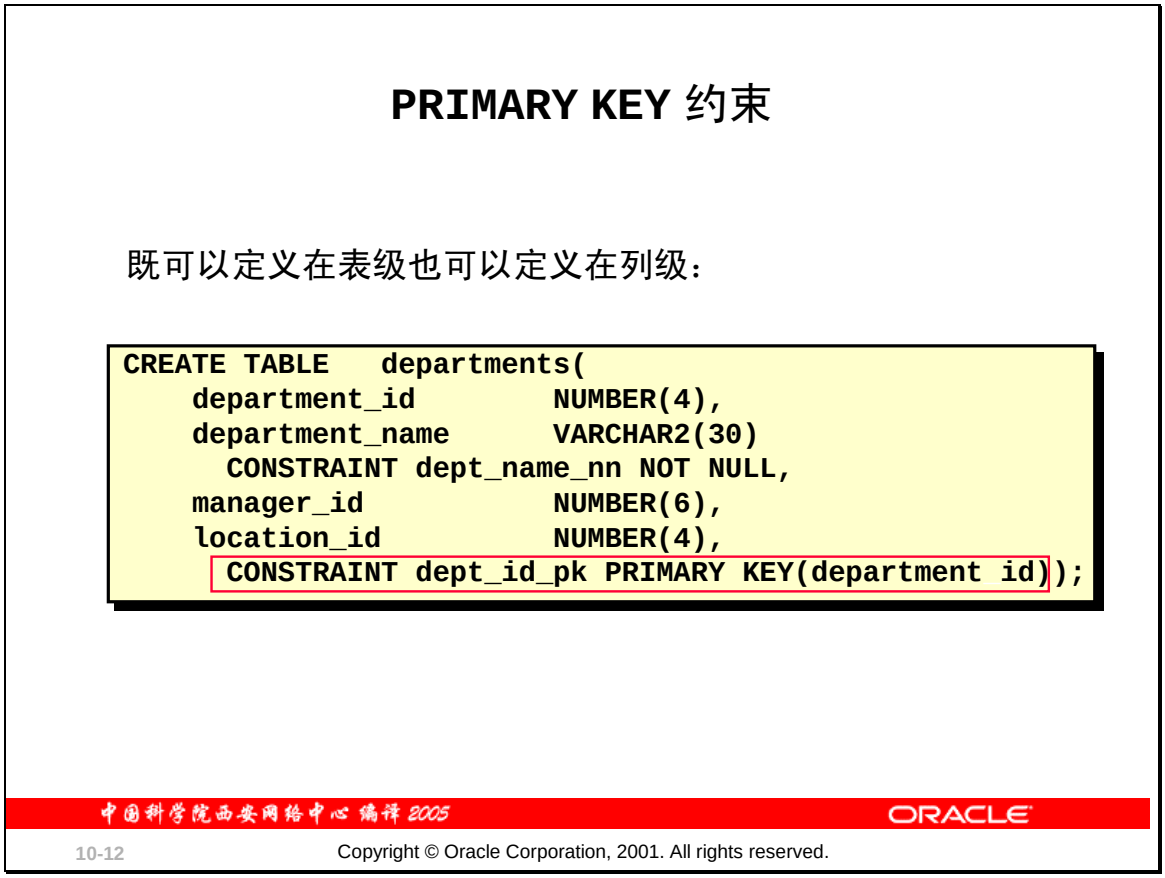
幻灯片 11



PRIMARY KEY 约束

PRIMARY KEY 约束为表创建一个主键。每个表只能创建一个主键。PRIMARY KEY 约束是表中的对行唯一标识的一个列或者列组合，该约束强制列或列组合的唯一性，并且确保作为主键一部分的列不能包含空值。

幻灯片 12



PRIMARY KEY 约束 (续)

PRIMARY KEY 约束既可以定义在列级也可以定义在表级。用表级定义创建一个组合 PRIMARY KEY。

一个表只能有一个 PRIMARY KEY 约束，但可以有多个 UNIQUE 约束。

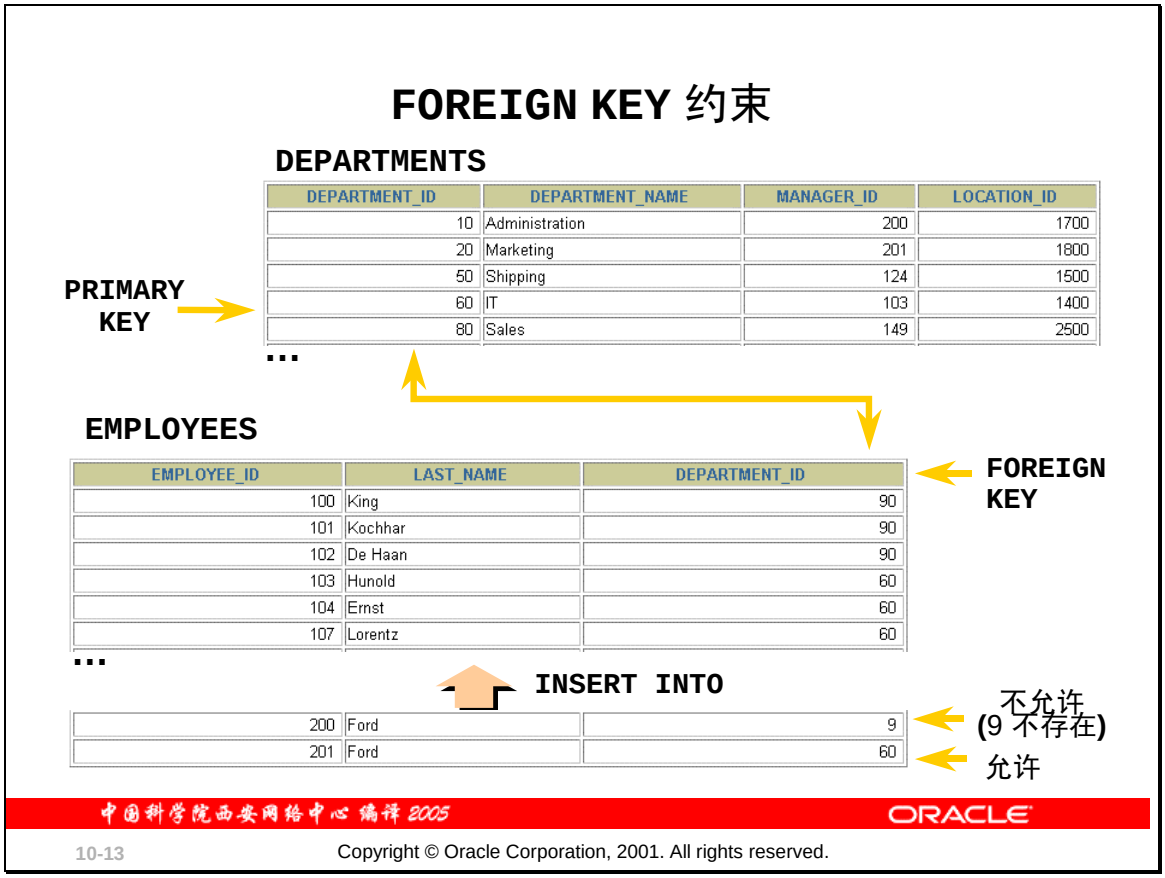
幻灯片的例子在 DEPARTMENTS 表的 DEPARTMENT_ID 列上定义了一个 PRIMARY KEY 约束。约束的名字是 DEPT_ID_PK。

注：对于一个 PRIMARY KEY 列 UNIQUE 索引被自动创建。

教师注释

上面显示的例子在你的方案中将不工作，因为 DEPARTMENTS 表不存在。为了示范该代码，修改脚本中的表的名字，然后再运行脚本。

幻灯片 13



FOREIGN KEY 约束

FOREIGN KEY，引用完整性约束，指明一个列或者列的组合作为一个外键，并且在相同表或者不同表的主键或者唯一键和外键之间建立一个关系。在幻灯片的例子中，DEPARTMENT_ID 已经在 EMPLOYEES 表（依赖表或子表）中被定义为外键；它引用 DEPARTMENTS 表（引用表或父表）的 DEPARTMENT_ID 列。

一个外键值必须匹配一个在父表中存在的值或者空值。

外键基于数据值，并且纯粹是逻辑的，不是物理的，指针。

教师注释

向学生解释，你不能创建不存在的主键值的外键。

幻灯片 14

FOREIGN KEY 约束

既可以定义在表级也可以定义在列级：

```
CREATE TABLE employees(
    employee_id      NUMBER(6),
    last_name        VARCHAR2(25) NOT NULL,
    email            VARCHAR2(25),
    salary           NUMBER(8,2),
    commission_pct   NUMBER(2,2),
    hire_date        DATE NOT NULL,
...
    department_id    NUMBER(4),
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)
      REFERENCES departments(department_id),
    CONSTRAINT emp_email_uk UNIQUE(email));
```


FOREIGN KEY 约束（续）

FOREIGN KEY 约束能够被定义在列或者表约束级。一个组合外键必须用表级定义创建。

幻灯片上的例子中用表级语法定义了一个 EMPLOYEES 表的关于 DEPARTMENT_ID 列的 FOREIGN KEY 约束，约束的名字是 EMP_DEPTID_FK。倘若约束是基于单个列的，外键也能够被定义在列级，其语法不同之处在于 FOREIGN KEY 不出现，例如：

```
CREATE TABLE employees
(...
department_id NUMBER(4) CONSTRAINT emp_deptid_fk
   REFERENCES departments(department_id),
...
)
```

幻灯片 15

FOREIGN KEY 约束 (续)

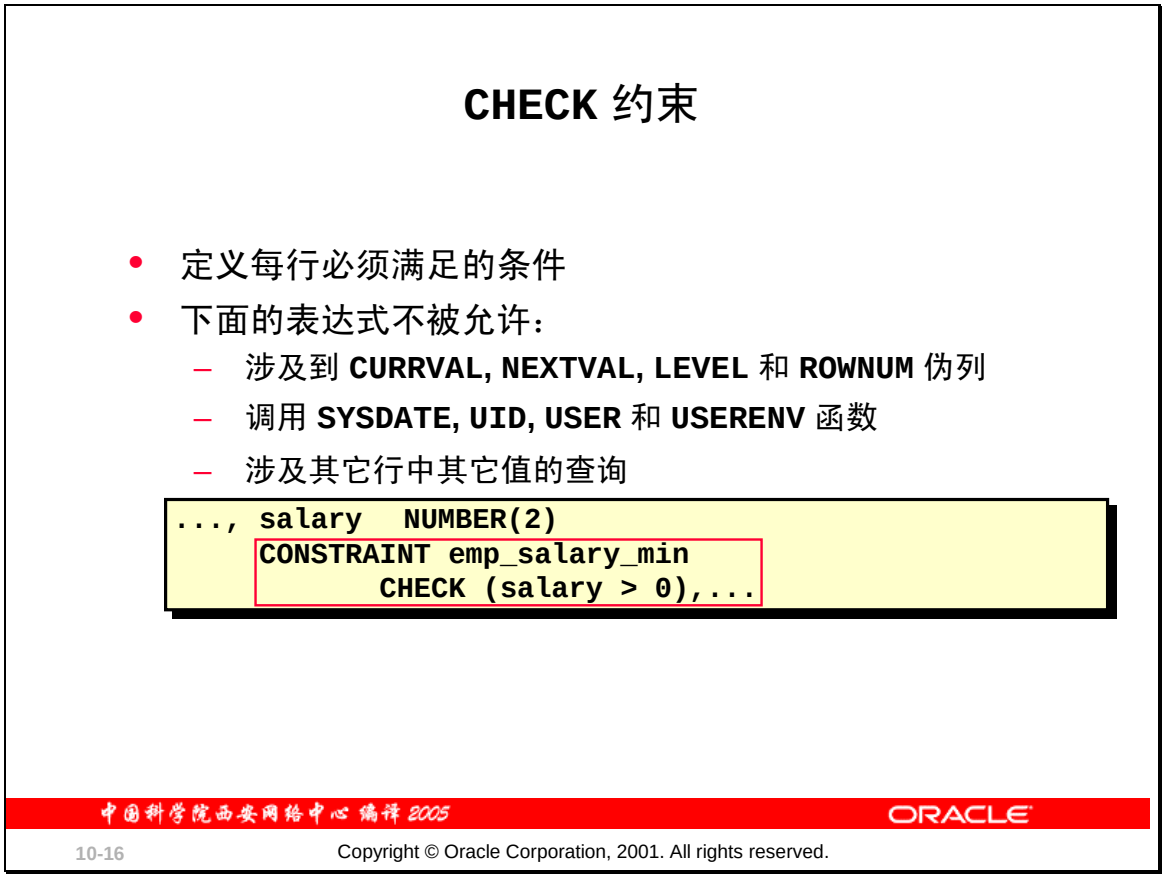
外键被定义在子表中，包含引用列的表是父表。外键用下面关键字的组合定义：

- FOREIGN KEY 被用于在表约束级定义子表中的列。
- REFERENCES 确定父表中的表和列。
- ON DELETE CASCADE 指出当父表中的行被删除时，子表中相依赖的行也将被级联删除。
- ON DELETE SET NULL 当父表的值被删除时，转换外键值为空。

默认行为被称为约束规则，该规则不允许引用数据的更新或删除。

无 ON DELETE CASCADE 或 ON DELETE SET NULL 选项，如果父表中的行在子表中引用，则它不能被删除。

幻灯片 16



CHECK 约束

CHECK 约束定义一个每行都必须满足的条件，该条件可以用和查询条件一样的结构，下面的情况例外：

- 引用 CURRVAL, NEXTVAL, LEVEL 和 ROWNUM 伪列
- 调用 SYSDATE, UID, USER 和 USERENV 函数
- 查询涉及其它行中的其它值

一个单个列在它的定义中可以有多个 CHECK 约束，在一个列上能够定义的 CHECK 约束的数目无限制。

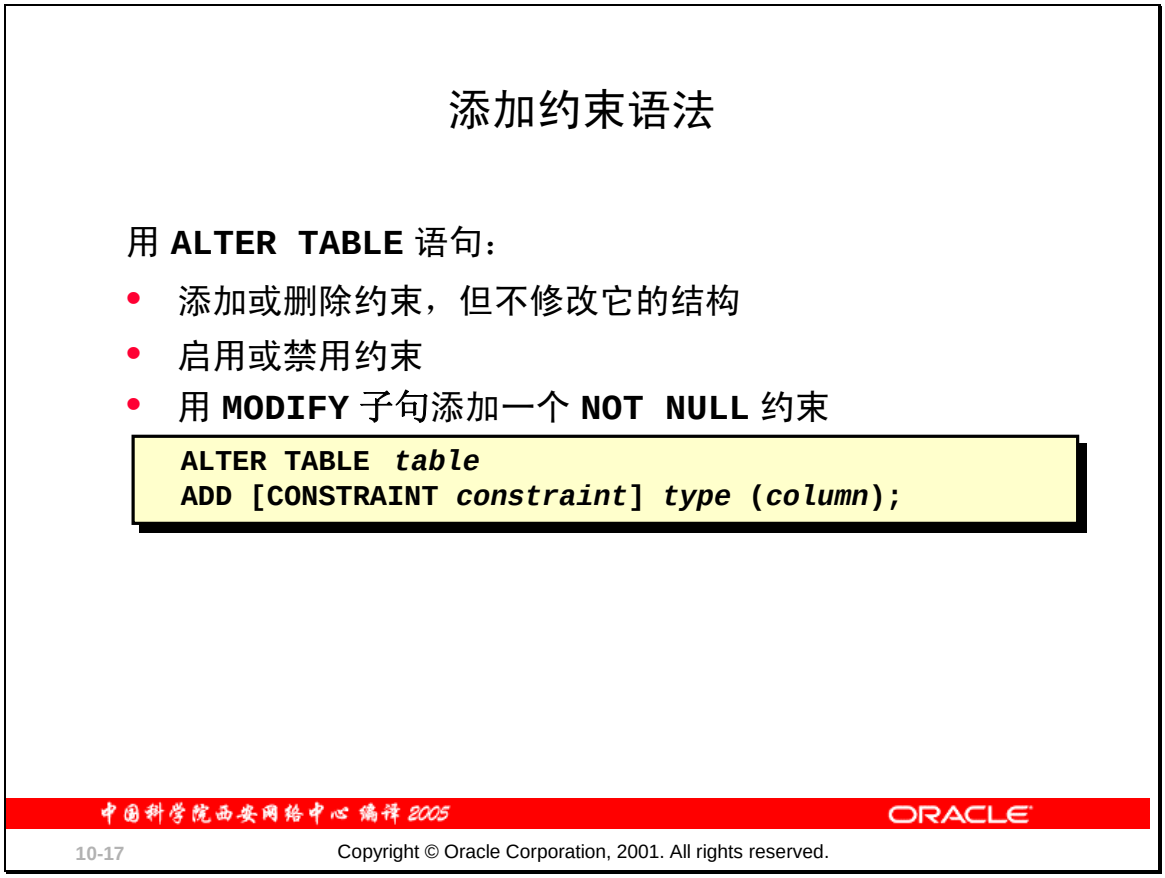
CHECK 约束能够被定义在列级或表级。

```
CREATE TABLE employees
      (...
       salary NUMBER(8,2) CONSTRAINT emp_salary_min
                                CHECK (salary > 0),
      ...
```

教师注释

解释什么是伪列。伪列不是表中实际的列，但它们的行为象列一样，例如，你能从伪列中选择够选择值，可是，你不能插入、更新或从伪列中删除。伪列能被用于 SQL 语句中。

幻灯片 17



添加约束

你可以用带 ADD 子句的 ALTER TABLE 语句为已经存在的表添加一个约束。

在语法中：

table	是表的名字
constraint	是约束的名字
type	是约束的类型
column	是受约束影响的列的名字

尽管建议命名约束，但约束名在语法中是个选项。如果你不命名约束，系统将产生约束名。

原则

- 你可以添加、删除或禁用一个约束，但你不能修改它的结构。
- 你可以用 ALTER TABLE 语句的 MODIFY 子句添加一个 NOT NULL 约束到一个已经存在的列。

注：只有在表是空的或者每个行的该列都有非空值的情况下，你才可以定义一个 NOT NULL 列。

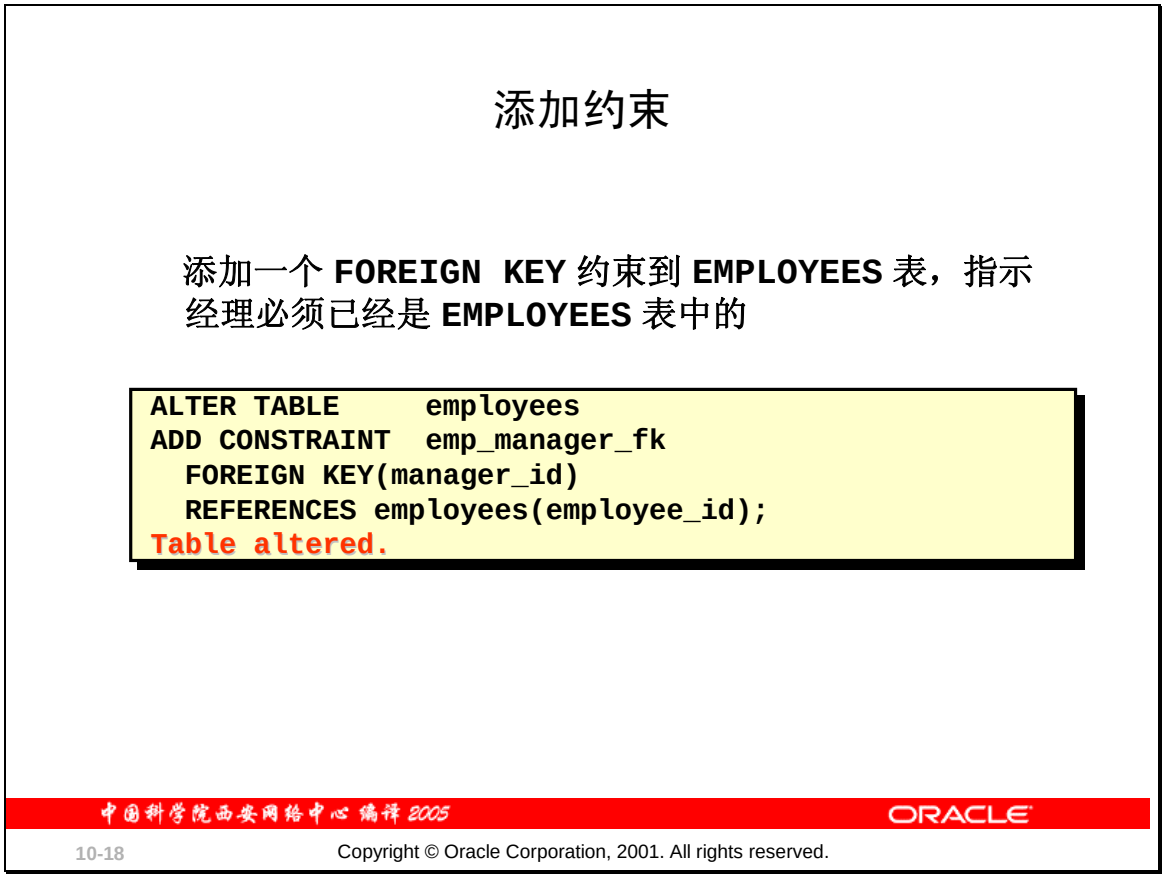
教师注释

你可以延迟检查约束的有效，直到事务结束。

如果仅在提交的时候进行系统检查，约束将被延迟 (*deferred*)。如果一个延迟的约束被违反，则该提交导致事务回退。

如果在每条语句结束时进行检查，约束是立即的 (*immediate*)。如果该约束被违反，则该语句被立即回退。

幻灯片 18



添加约束 (续)

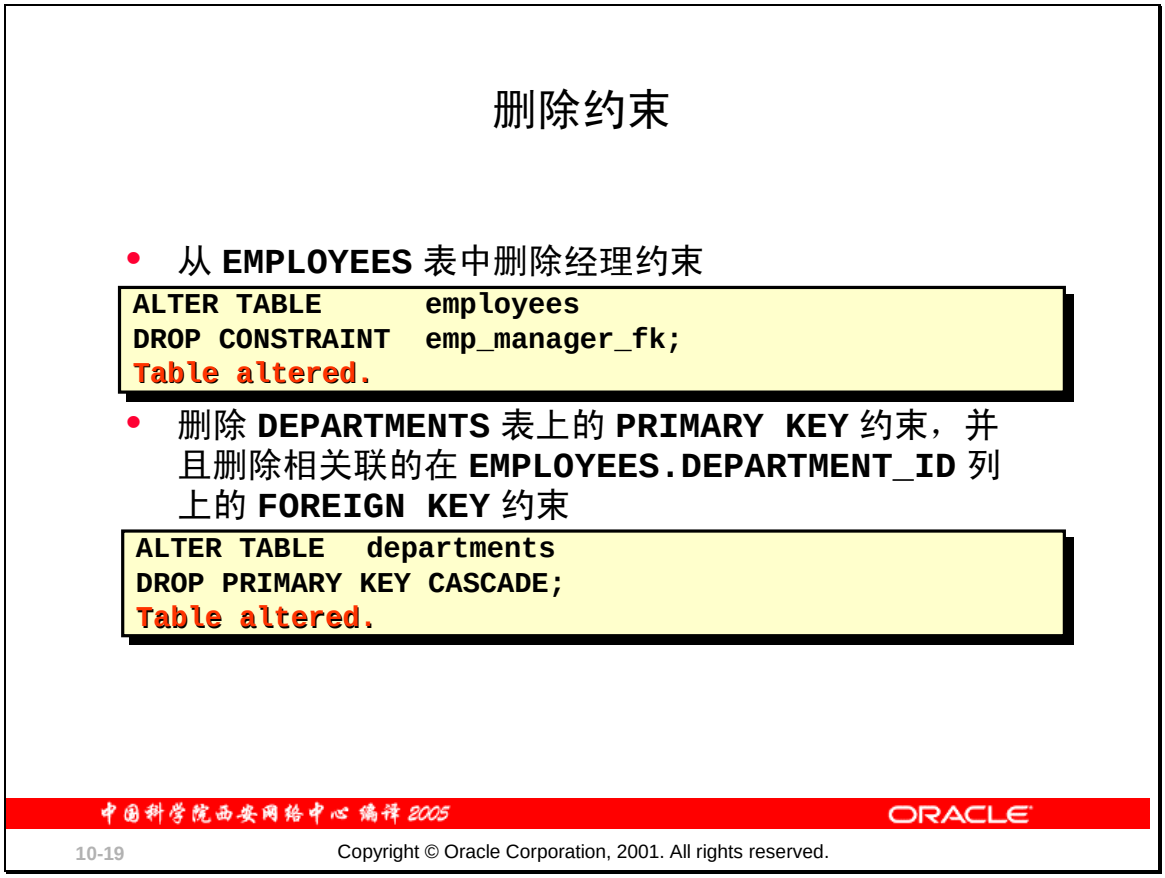
幻灯片中的例子在 EMPLOYEES 表上创建 FOREIGN KEY 约束，该约束确保在 EMPLOYEES 表中经理作为一个有效的雇员而存在。

教师注释

用 ALTER TABLE MODIFY 语法添加一个 NOT NULL 约束：

```
ALTER TABLE employees
MODIFY (salary CONSTRAINT emp_salary_nn NOT NULL);
```

幻灯片 19



删除约束

为了删除约束，你可以先从 USER_CONSTRAINTS 和 USER_CONS_COLUMNS 数据字典视图中确定约束的名字，然后使用带 DROP 子句的 ALTER TABLE 语句。DROP 子句的 CASCADE 选项导致任何与其相依赖的约束也被删除。

语法

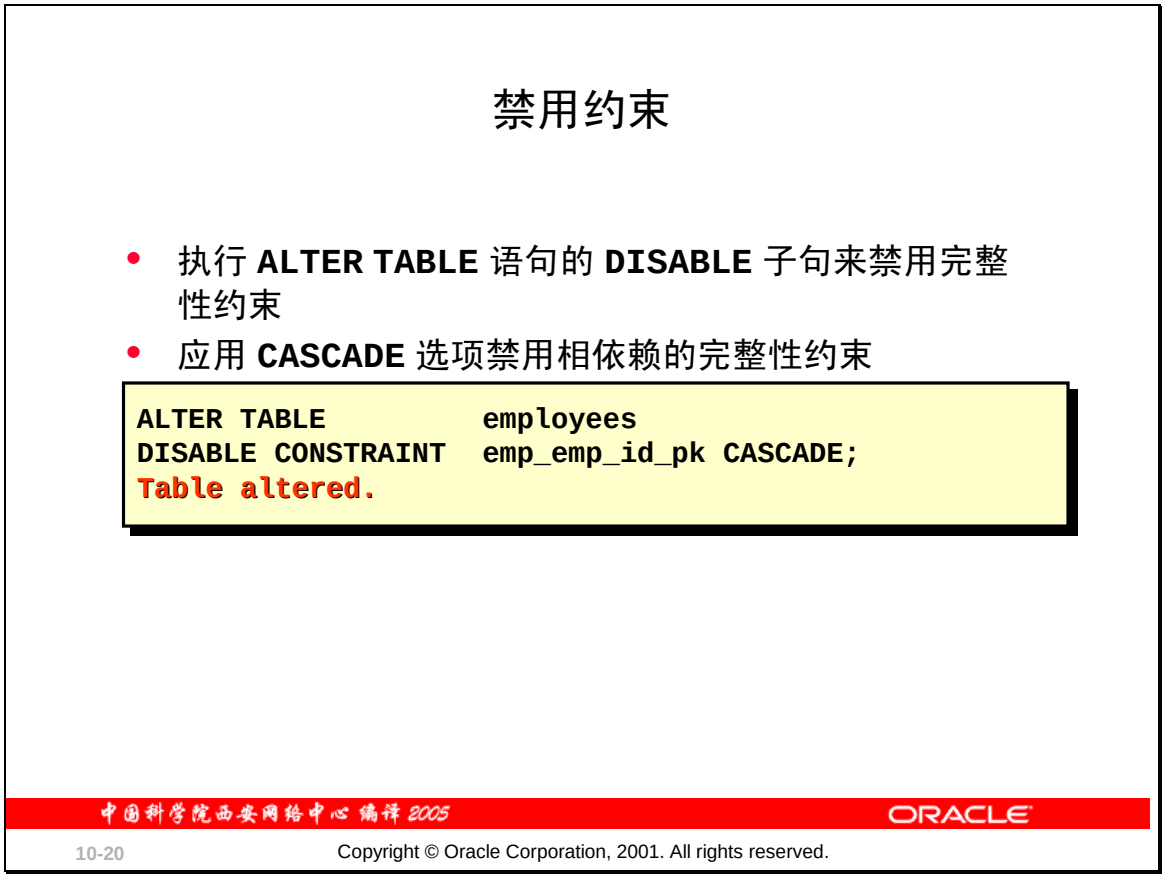
```
ALTER TABLE  table
DROP  PRIMARY KEY | UNIQUE (column) |
 CONSTRAINT   constraint  [CASCADE];
```

在语法中：

table	是表的名字
column	是受约束影响的列的名字
constraint	是约束的名字

当你删除一个完整性约束时，约束不再由 Oracle 服务器强制，并且在数据字典中不再可用。

幻灯片 20



禁用约束

你可以禁用一个约束而不删除它，或者用带 DISABLE 子句的 ALTER TABLE 语句重新创建它。

语法

```
ALTER   TABLE   table
DISABLE CONSTRAINT constraint [CASCADE];
```

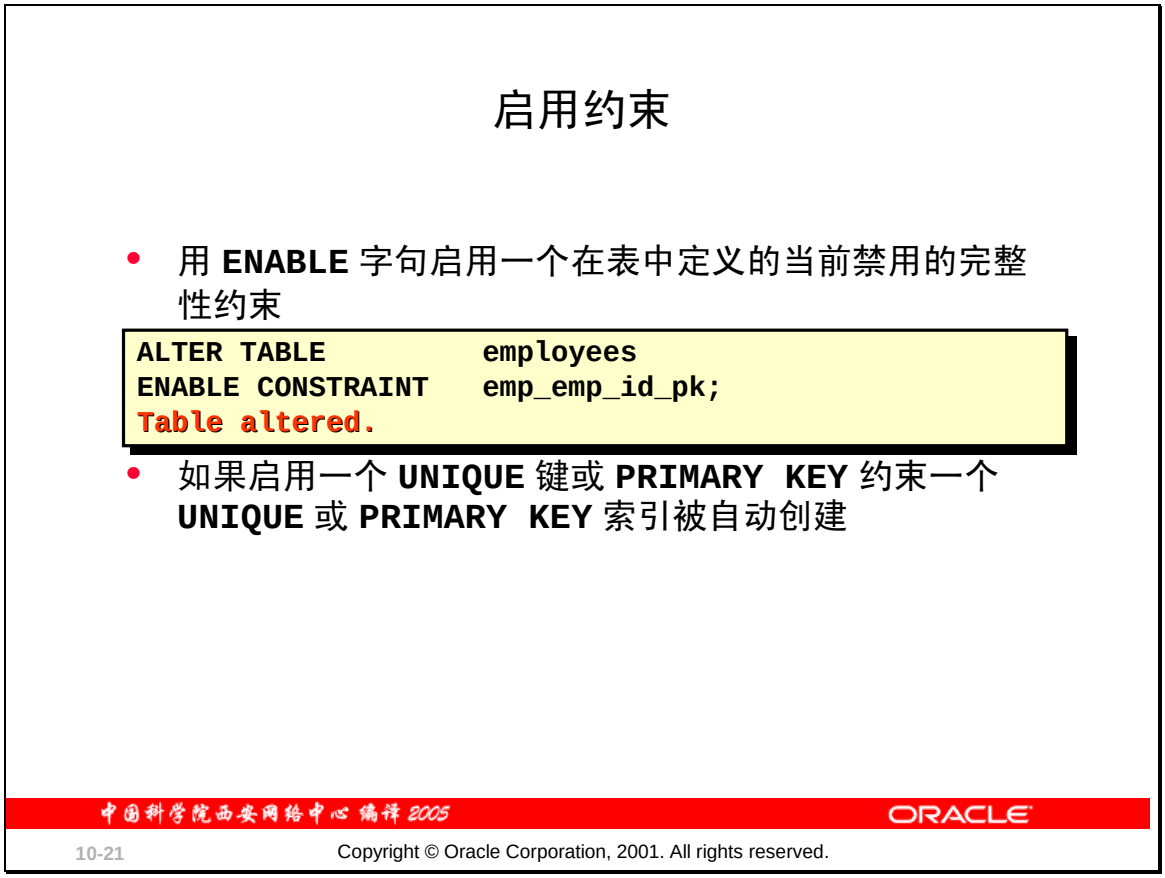
在语法中：

table	是表的名字
constraint	是约束的名字

原则

- 你即可以在 CREATE TABLE 语句也可以在 ALTER TABLE 语句中使用 DISABLE 子句。
- CASCADE 子句禁用相依赖的完整性约束。
- 禁用唯一或主键约束会移除唯一性索引。

幻灯片 21



启用约束

你可以用带 ENABLE 子句的 ALTER TABLE 语句启用一个禁用的约束，而不需要重新创建它。

语法

```
ALTER   TABLE      table
ENABLE  CONSTRAINT constraint;
```

在语法中：

table	是表的名字
constraint	是约束的名字

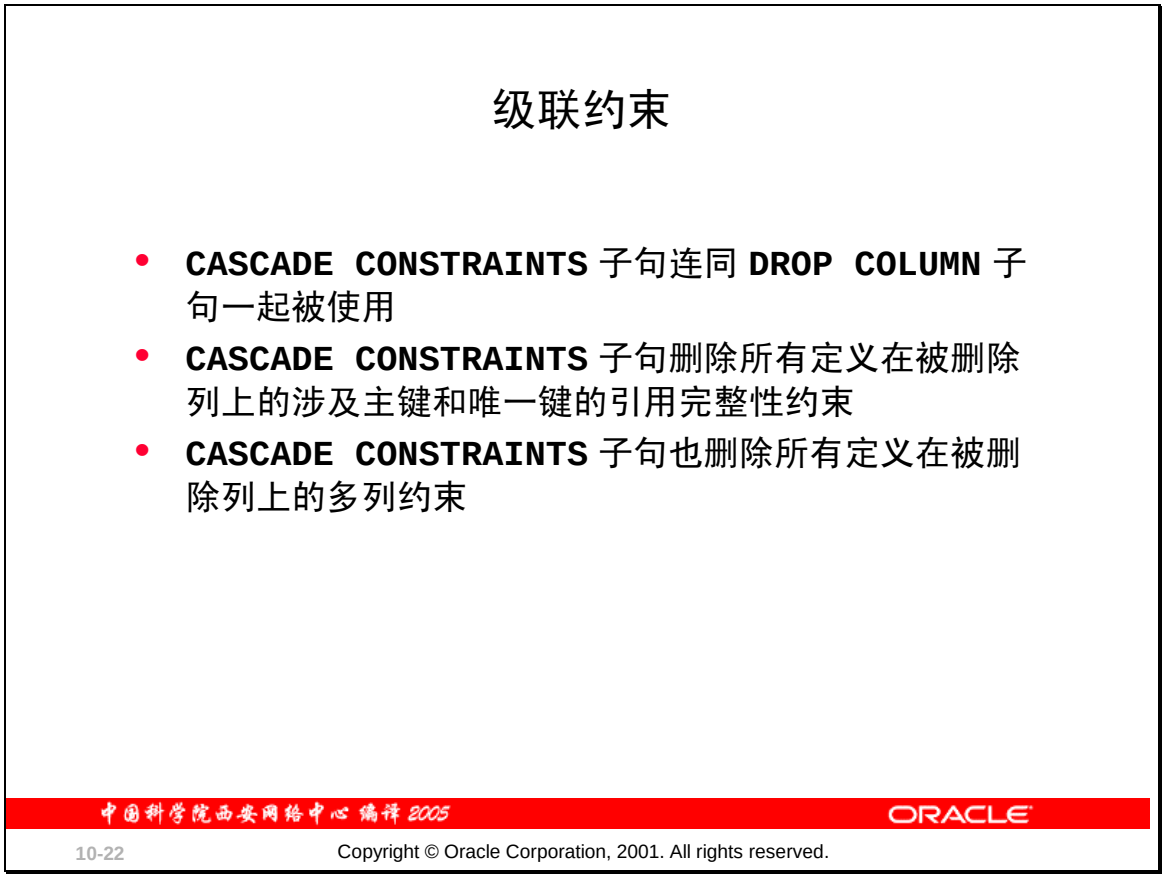
原则

- 如果启用一个约束，约束将应用于表中所有的数据，所有在表中的数据都必须适合该约束。
- 如果你启用一个 UNIQUE 键或者 PRIMARY KEY 约束，一个 UNIQUE 或 PRIMARY KEY 索引将被自动地创建。
- 你即可以 CREATE TABLE 语句也可以在 ALTER TABLE 语句中使用 ENABLE 子句。
- 启用一个带 CASCADE 选项的被禁用的主键约束不会起用任何依赖于该主键的外键。

教师注释

关于 VALIDATE 和 NOVALIDATE 选项的更多信息，请看 10-29 页的教师注释。

幻灯片 22



级联约束

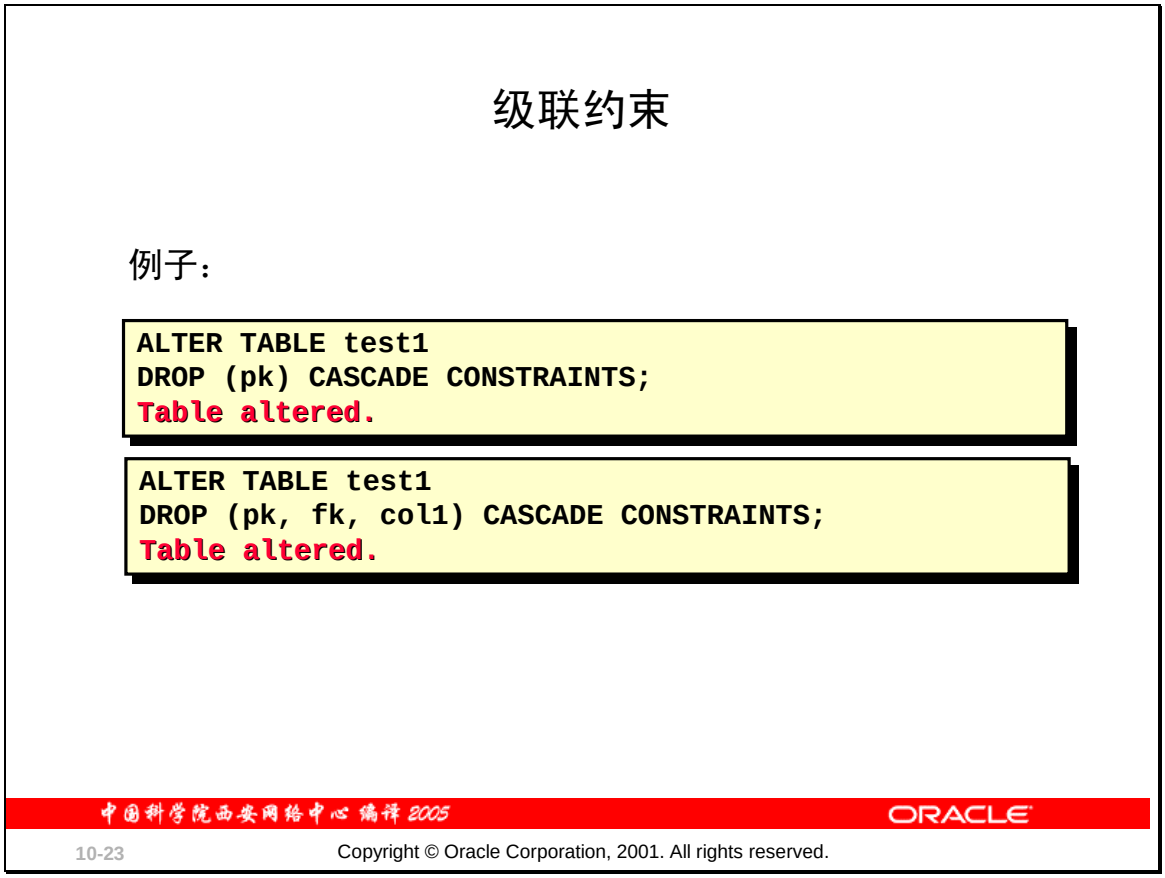
该语句举例说明 CASCADE CONSTRAINTS 子句的用法。假设表 TEST1 被如下创建：

```
CREATE TABLE test1 (
  pk NUMBER PRIMARY KEY,
  fk NUMBER,
  col1 NUMBER,
  col2 NUMBER,
  CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES test1,
  CONSTRAINT ck1 CHECK (pk > 0 and col1 > 0),
  CONSTRAINT ck2 CHECK (col2 > 0));
```

对于下面的语句返回一个错误：

```
ALTER TABLE test1 DROP (pk);  -- pk 是父键
ALTER TABLE test1 DROP (col1); -- col1 被多列约束 ck1 引用
```

幻灯片 23



级联约束 (续)

提交下面的语句删除行 PK，主键约束，fk_constraint 外键约束和检查约束 CK1：

```
ALTER TABLE test1 DROP (pk) CASCADE CONSTRAINTS;
```

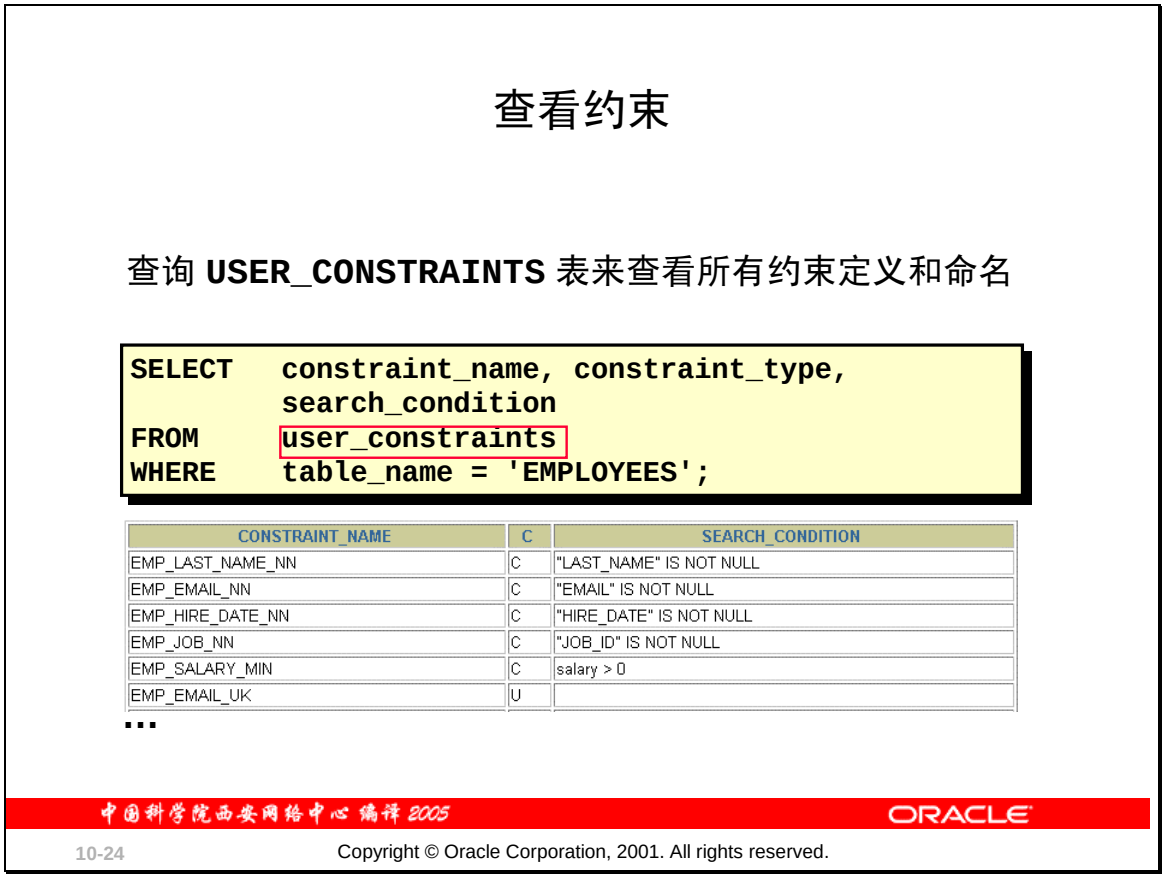
如果所有由定义在已删除列上的约束引用的列也被删除，那么就不需要 CASCADE CONSTRAINTS，例如，假设没有其它表引用 PK，提交下面的不带 CASCADE CONSTRAINTS 子句的语句是恰当的：

```
ALTER TABLE test1 DROP (pk, fk, col1);
```

教师注释

让学生知道如果任何约束被来自其它表中的列或在目的表中的保留列所引用，那么必须指定 CASCADE CONSTRAINTS，否则，该语句异常中断，并且返回错误提示 ORA-12991：在多列约束中的列被引用。

幻灯片 24



查看约束

在创建表之后，你可以用 DESCRIBE 命令来确认它的存在。你唯一能够校验的约束是 NOT NULL 约束。为了查看表上所有的约束，查询 USER_CONSTRAINTS 表。幻灯片的例子显示了 EMPLOYEES 表上的约束。

注：那些没有被表的所有者命名的约束将收到系统指定的约束名。在约束类型中，C 代表 CHECK，P 代表 PRIMARY KEY，R 代表引用完整性， U 代表 UNIQUE 键。注意 NOT NULL 约束实际上是一个 CHECK 约束。

教师注释

给学生指出 NOT NULL 约束在数据字典中被作为 CHECK 约束存储。将他们的注意力引到约束类型，对于幻灯片中的 NOT NULL 约束，在 constraint_type 域输入 C（表示 CHECK）。

幻灯片 25

查看约束关联的列

观察在 **USER_CONS_COLUMNS** 视图中与约束名关联的列

```
SELECT   constraint_name, column_name
FROM     user_cons_columns
WHERE    table_name = 'EMPLOYEES';
```

CONSTRAINT_NAME	COLUMN_NAME
EMP_DEPT_FK	DEPARTMENT_ID
EMP_EMAIL_NN	EMAIL
EMP_EMAIL_UK	EMAIL
EMP_EMP_ID_PK	EMPLOYEE_ID
EMP_HIRE_DATE_NN	HIRE_DATE
EMP_JOB_FK	JOB_ID
EMP_JOB_NN	JOB_ID

...

查看约束 (续)

你可以用 USER_CONS_COLUMNS 数据字典视图查看与约束相关的列名。该视图对于那些由系统指定名字的约束特别有用。

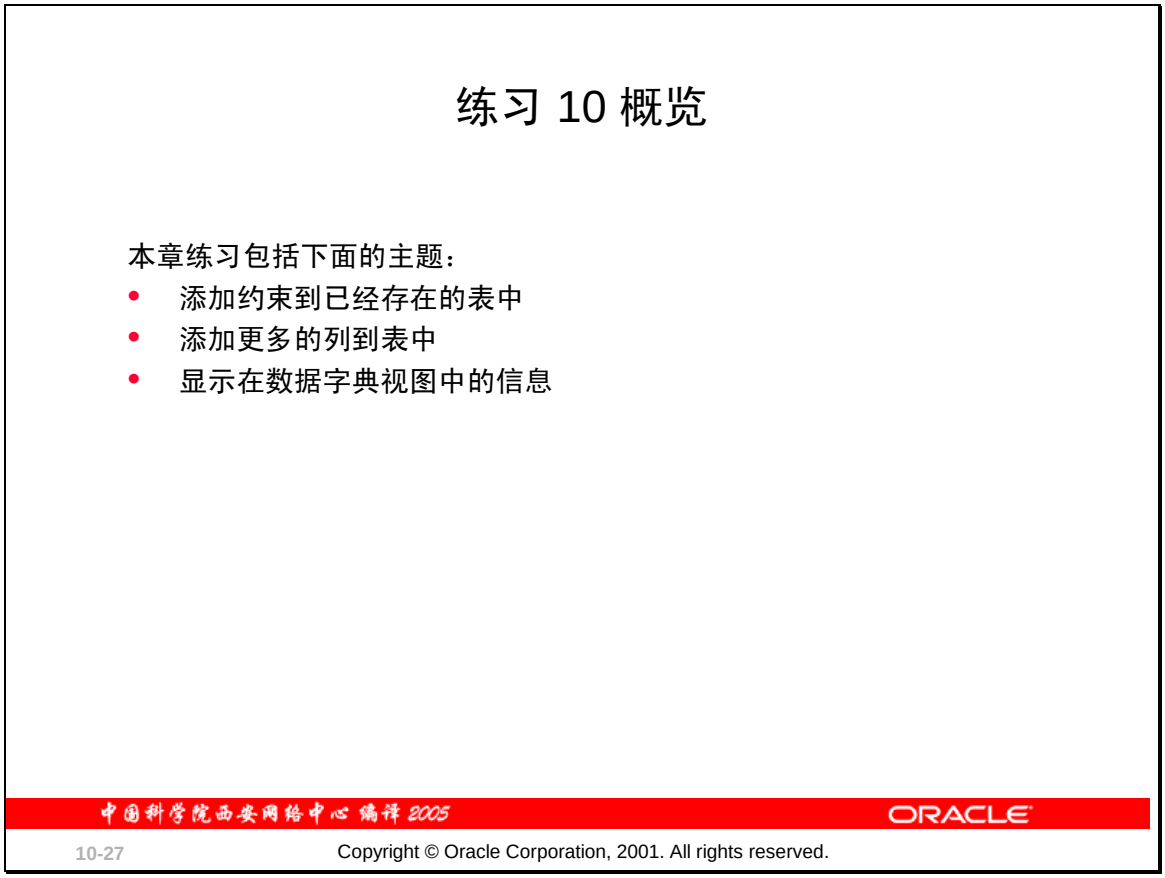
幻灯片 26

小结

在本课中，您应该已经学会如何创建约束：

- 约束类型：
  - **NOT NULL**
  - **UNIQUE**
  - **PRIMARY KEY**
  - **FOREIGN KEY**
  - **CHECK**
- 你能够通过查询 **USER_CONSTRAINTS** 表来观察所有约束定义和命名

幻灯片 27



练习 10 概览

在本章的练习中，你将用本课中学过的语句添加约束和更多的列到表中。

注：建议你命名你在练习中所定义的约束。

练习 10

1. 添加一个表级 PRIMARY KEY 约束到 EMP 表的 ID 上，约束在创建时应该被命名，将该约束命名为 my_emp_id_pk。

提示：约束在 ALTER TABLE 命令执行成功后立即生效。

```
ALTER TABLE emp
ADD CONSTRAINT my_emp_id_pk PRIMARY KEY (id);
```

2. 创建一个 PRIMARY KEY 约束到 DEPT 表的 ID 列上，约束在创建时应该被命名，将该约束命名为 my_dept_id_pk。

提示：约束在 ALTER TABLE 命令执行成功后立即生效。

```
ALTER TABLE dept
ADD CONSTRAINT my_deptid_pk PRIMARY KEY(id);
```

3. 添加一个列 DEPT_ID 到 EMP 表中。添加一个外键到 EMP 表的 DEPT_ID 列上，确保雇员不被指定到一个不存在的部门，将该约束命名为 my_emp_dept_id_fk。

```
ALTER TABLE emp
ADD (dept_id NUMBER(7));
ALTER TABLE emp
ADD CONSTRAINT my_emp_dept_id_fk
FOREIGN KEY (dept_id) REFERENCES dept(id);
```

4. 查询 USER_CONSTRAINTS 视图，确认约束已被添加，注意约束的类型和名字。将 语句文本保存到文件 lab10_4.sql 中。

```
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name IN ('EMP', 'DEPT');
```

5. 从 USER_OBJECTS 数据字典视图中显示 EMP 和 DEPT 表的对象名和类型。注意新表和新的索引被创建。

```
SELECT object_name, object_type
FROM user_objects
WHERE object_name LIKE 'EMP%'
OR object_name LIKE 'DEPT%';
```

如果有时间，你可以完成下面的：

6. 修改 EMP 表，添加 NUMBER 数据类型的列 COMMISSION，精度 2，数值范围 2。添加一个约束到 commission 列，确保佣金值大于零。

```
ALTER TABLE EMP
ADD commission NUMBER(2,2)
CONSTRAINT my_emp_comm_ck CHECK (commission >= 0;
```

幻灯片 29

教师注释 (10-21 页)

你可以用带ENABLE或DISABLE的任意组合设置约束到VALIDATE或NOVALIDATE。

- VALIDATE 确保现有的数据符合约束。
- NOVALIDATE 意思是有些现有的数据可能不符合约束。

另外：

- ENABLE VALIDATE 与 ENABLE 相同。检查约束并且保证对所有行有效。
- ENABLE NOVALIDATE 意思是约束被检查，但结果不必对所有行都为真。该设置允许现有的行行违反约束，但确保所有新行的或修改的行是正确的。
- 在一个 ALTER TABLE 语句中，ENABLE NOVALIDATE 在不确认表中现有数据的情况下，恢复禁用的约束。
- DISABLE NOVALIDATE 与 DISABLE 是相同的。不检查约束，并且结果不必为真。
- DISABLE VALIDATE 显示约束，删除约束上的索引，并且不允许修改任何被约束的行。

在这些状态之间的事务受下面的规则支配：

- 除非 NOVALIDATE 被指定，ENABLE 暗示 VALIDATE。
- 除非 VALIDATE 被指定，DISABLE 暗示 NOVALIDATE。
- VALIDATE 和 NOVALIDATE 对于 ENABLE 和 DISABLE 状态没有任何默认的暗示。
- 当一个唯一或主键从 DISABLE 状态移动到 ENABLE 状态，并且没有现有的索引，一个唯一索引被自动创建。

类似地，当一个唯一或主键从 ENABLE 移动到 DISABLE，并且一个唯一索引是启用的，该唯一索引被删除。

当任何约束从 NOVALIDATE 状态移动到 VALIDATE 状态时，所有的数据都必须被检查 (这个过程可能很慢)。但从 VALIDATE 状态移动到 NOVALIDATE 状态只是简单地忽略曾经被检查过的数据。

从 ENABLE NOVALIDATE 状态移动一个单个的约束到 ENABLE VALIDATE 状态不妨碍读、写或者其它 DDL 语句，它可以以并行方式执行。

下面的语句的启用但不验证禁用的完整性约束：

```
ALTER TABLE employees
   ENABLE NOVALIDATE CONSTRAINT EMP_EMAIL_UK;
ALTER TABLE employees
   ENABLE NOVALIDATE PRIMARY KEY
   ENABLE NOVALIDATE UNIQUE (employee_id, last_name);
```

下面的语句启用或验证禁用的完整性约束：

```
ALTER TABLE employees
   MODIFY CONSTRAINT emp_email_uk VALIDATE;
ALTER TABLE employees
   MODIFY PRIMARY KEY ENABLE NOVALIDATE;
```