幻灯片 1

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

幻灯片 2

目标

完成本课后，您应当能够执行下列操作：

- 描述主要数据库对象
- 创建表
- 描述列定义时可用的数据类型
- 改变表的定义
- 删除、改名和截断表

课程目标

在本课中，你将学习关于表主要数据库对象，以及他们之间的关系。你还将学习怎样创建、修改和删除表。

幻灯片 3

数据库对象

对象	说明
表	基本存储单元；由行和列组成
视图	逻辑地从一个或多个表中表示数据子集
序列	数字值发生器
索引	改善一些查询的性能
同义词	给对象可选择的名字

数据库对象

Oracle 数据库能够包含多种数据结构。每一种结构应该在数据设计中描述，以使它能够在数据库开发阶段被创建。

- Table：存储数据
- View：来自一个或多个表的数据的子集
- Sequence：数字值发生器
- Index：改善一些查询的性能
- Synonym：给对象一个可替代的名字

Oracle9*i* 表结构

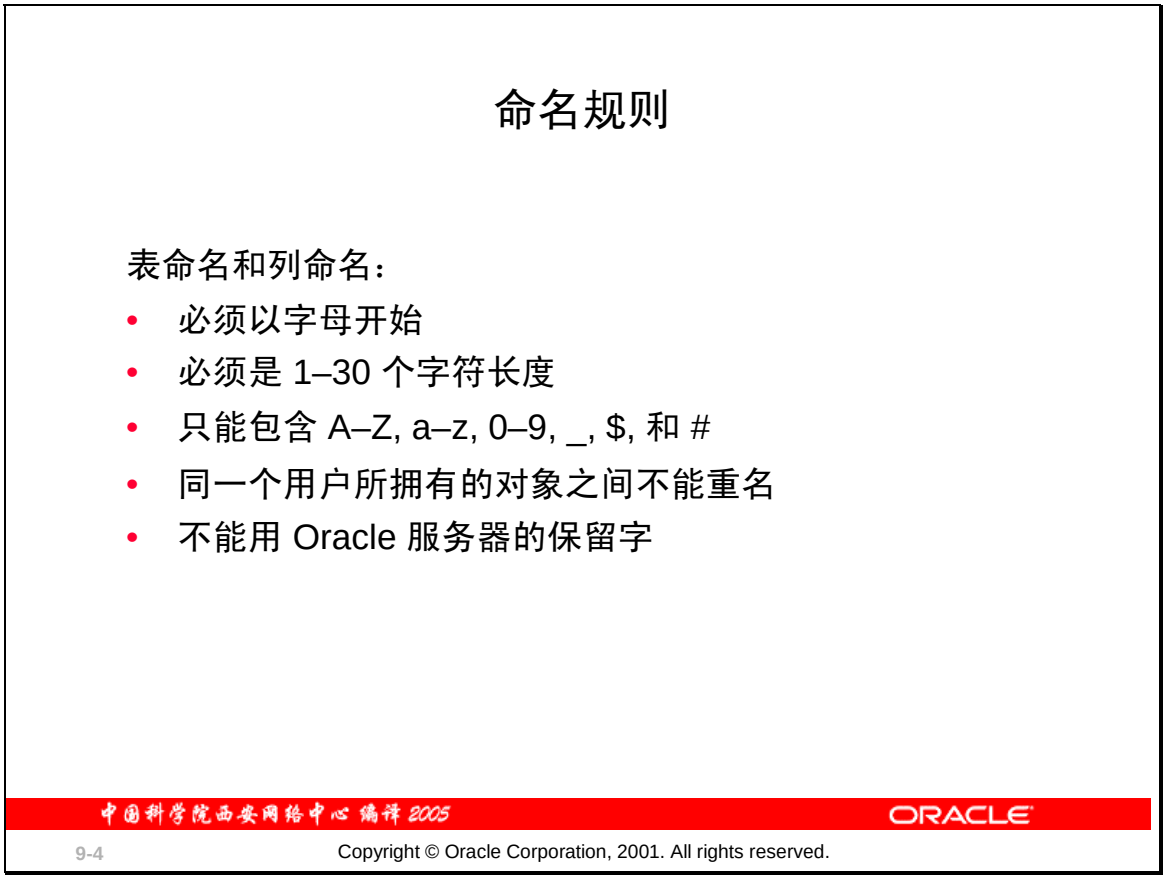
- 表可以在任何时间被创建，即使用户正在使用数据库
- 你不需要指定表的大小，表的大小最终由作为一个整体分配给数据库的空间的数量定义。但是随着时间的过去一个表将使用多少空间是重要的。
- 表结构能够被联机修改。

注：除了幻灯片中介绍的数据库对象，还有一些其他的数据库对象，但没有包括在本课程中。

教师注释

表能够有多达 1,000 个列，并且必须符合标准的数据库对象命名约定。当使用 AS 子查询子句时，列定义可以忽略。表在创建时没有数据，除非指定了一个查询。行通常用 INSERT 语句添加。

幻灯片 4



命名规则

依照命名 Oracle 数据库对象的标准规则来命名数据库表和列：

- 表名和列名必须由一个字母开始，长度在 1 - 30 之间。
- 名字只能包含 A - Z, a - z, 0 - 9, _ (下划线)，$ 和 # (合法字符，但建议不要使用)。
- 同一个 Oracle 服务器用户所拥有的对象名字不能重复。
- 名字不能用 Oracle 服务器的保留字。

命名原则

使用描述性的名字为表和其他数据库对象命名。

注：名字是大小写不敏感的，例如， `EMPLOYEES` 与 `eMPloyees` 或 `eMpLOYEES` 作为同一个名字来处理。

更多信息，见 *Oracle9i SQL Reference*，“对象名字与限定”。

幻灯片 5

CREATE TABLE 语句

- 必须有：
  - **CREATE TABLE** 权限
  - 一个存储区域

```
CREATE TABLE [schema.]table
            (column datatype [DEFAULT expr][, ...]);
```

- 指定：
  - 表名
  - 列名、列数据类型和列的大小

CREATE TABLE 语句

用 SQL 的 CREATE TABLE 语句创建表以存储数据，该语句是数据定义语言 (DDL) 语句之一，其它的 (DDL) 语句将在后面讲述。DDL 语句是 SQL 语句的一个子集，用来创建、修改或删除 Oracle9*i* 数据库的结构。这些语句会立即作用于数据库，并且他们还将信息记录在数据字典中。

为了创建表，用户必须有 CREATE TABLE 权限和用于创建对象的存储区域。数据库管理员用数据控制语言 (DCL) 语句，DCL 语句将在后面讲述，授予权限给用户。

在语法中：

schema	与所有者的名字一样
table	表的名字
DEFAULT *expr*	指定默认值，在 NSERT 语句省略值时使用
column	列的名字
datatype	列的数据类型和长度

教师注释

请读 9-37 页的教师注释

幻灯片 6

引用另一个用户的表

- 表属于另一个用户，不在该用户的方案中
- 在那些表名字的前面使用所有者的名字作为前缀

引用另一个用户的表

方案 (*schema*) 是对象的集合，方案对象直接反映数据在数据库中的逻辑结构，方案对象包括表、视图、同义词、序列、存储过程、索引、集群和数据库链接。

如果一个表不属于本用户，那么，其所有者的名字必须放在表名的前面，例如，如果一个方案命名为 USER_B，并且 USER_B 有一个表 EMPLOYEES，那么，其他用户用下面的语句从表中取回数据：

```
SELECT *
FROM   user_b.employees;
```

幻灯片 7

DEFAULT 选项

- 在插入时，为一个列指定一个默认值

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- 文字值、表达式或者 SQL 函数都是合法的值
- 另一个列名或者伪列是不合法的值
- 默认数据类型必须与列的数据类型匹配

DEFAULT 选项

一个列可以用 DEFAULT 选项给予一个默认值，列该选项防止插入时输入空值到列中。默认值可以是文字、表达式或 SQL 函数，例如用 SYSDATE 和 USER。 但默认值不能是另一个列的名字或伪列，例如 NEXTVAL 或 CURRVAL。默认表达式必须与列的数据类型相匹配。

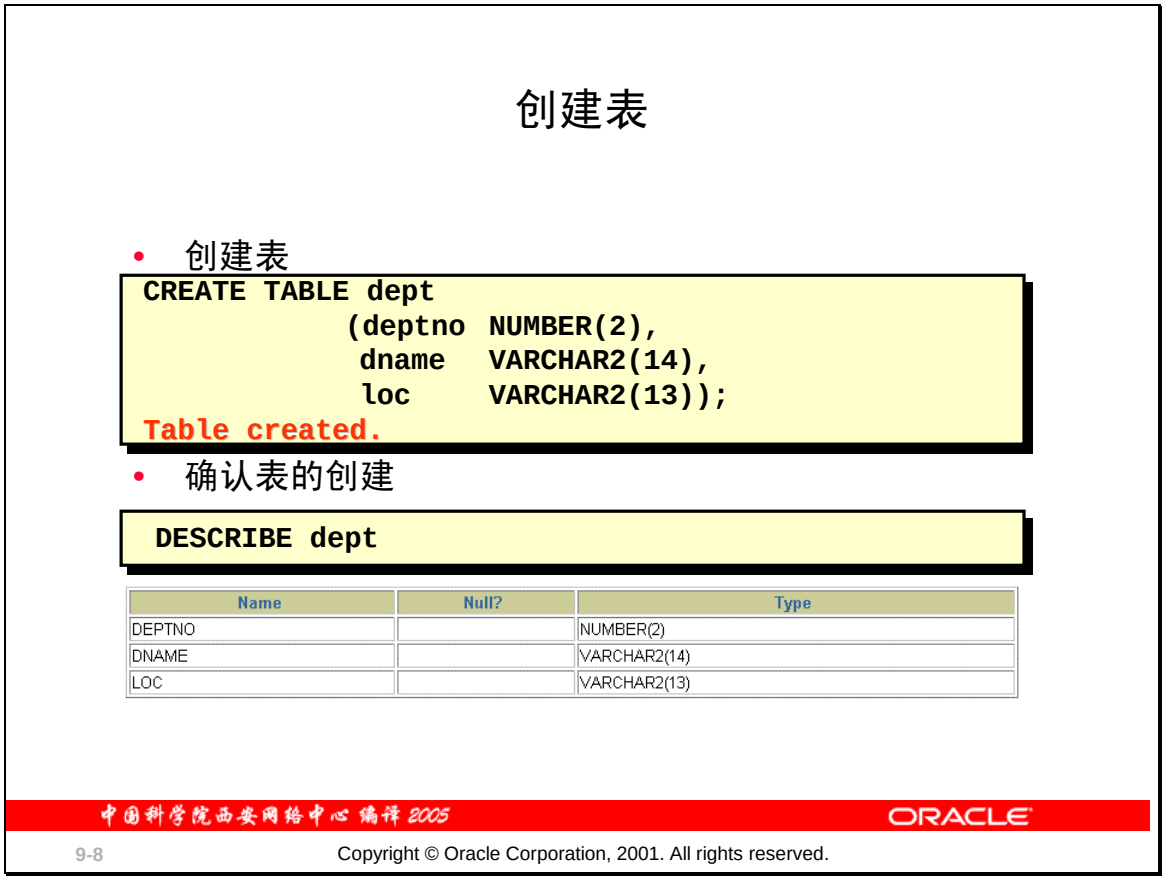
注：CURRVAL 和 NEXTVAL 在后面的章节中说明。

教师注释

这里是一个伪列的例子，对于一个查询返回的每一行，ROWNUM 伪列返回一个数来指示 Oracle 服务器从一个表中选择的行顺序。被选择的第一行的 ROWNUM 为 1，第二行为 2，等等。

对于带有 DEFAULT 关键字的 INSERT 和 UPDATE 语句，其默认值的处理方式将在“操纵数据” 一课中讲述。

幻灯片 8



创建表

幻灯片内中的例子创建 DEPT 表，该表有三个列：DEPTNO、DNAME 和 LOC。用 DESCRIBE 命令来确认表的创建。

因为创建表是一个 DDL 语句，当该语句执行时将会发生一个自动提交。

教师注释

解释 CREATE TABLE 的附加语法可以包括约束等。对于 CREATE TABLE 语法的更多信息，请参考：*Oracle9i SQL Reference*，“创建表”。

幻灯片 9

Oracle 数据库中的表

- 用户表：
  - 由用户创建和维护的表的集合
  - 包含用户信息
- 数据字典：
  - 由 Oracle 服务器创建和维护的表的集合
  - 包含数据库信息

Oracle 数据库中的表

用户表由用户创建，例如 EMPLOYEES。在 Oracle 数据库中有另一个表和视图的集合称为数据字典 (*data dictionary*)，该集合由 Oracle 服务器创建和维护，其中包含有关数据库的信息。

全部数据字典表的所有者是用户 SYS。数据字典表的基表很少被用户访问，因为其中的信息不容易理解，因此，用户一般是访问数据字典视图，因为视图中的信息是以容易理解的格式表示的。存储在数据字典中的信息包括 Oracle 服务器用户的名字，被授予用户的权限，数据库对象名，表结构和审计信息。

有四种数据字典视图，每一种有一个特定的前缀来反映其不同的目的。

前缀	说明
USER_	这些视图包含关于用户所拥有的对象的信息。
ALL_	这些视图包含所有用户可访问的表 (对象表和相关的表) 的信息。
DBA_	这些视图是受限制的视图，它们只能被分配有 DBA 角色的用户所访问。
V$	这些视图是动态执行的视图，包含数据库服务器的性能、存储器和锁的信息。

幻灯片 10

查询数据字典

- 查看本用户所拥有的表的名称

```
SELECT table_name
FROM   user_tables ;
```

- 查看本用户所拥有的不同的对象类型

```
SELECT DISTINCT object_type
FROM   user_objects ;
```

- 查看本用户所拥有的表、视图、同义词和序列

```
SELECT *
FROM   user_catalog ;
```


查询数据字典

你可以查询数据字典表来查看你所拥有的各种数据库对象。常用的数据字典表有：

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG

注：USER_CATALOG 有一个称为 CAT 的同义词，你可以在 SQL 语句中用该同义词代替 USER_CATALOG。

```
SELECT  *
FROM    CAT;
```

幻灯片 11

数据类型

数据类型	说明
VARCHAR2(*size*)	可变长度的字符数据
CHAR(*size*)	固定长度的字符数据
NUMBER(*p,s*)	可变长度的数字数据
DATE	日期和时间值
LONG	最大2G的可变长度字符数据
CLOB	最大4G的字符数据
RAW and LONG RAW	原始二进制数据
BLOB	最大4G的二进制数据
BFILE	最大4G的，存储在外部文件中的二进制数据
ROWID	一个64进制的数制系统，表示表中一行的唯一地址

数据类型

数据类型	说明
VARCHAR2(*size*)	可变长度字符数据（必须指定最大字符数：最小字符数是 1；最大字符数是 4000）
CHAR [(*size*)]	固定长度字符数据，长度的大小以字节为单位(默认和最小字符数为 1；最大字符数为 2000)
NUMBER [(*p*,*s*)]	数字，精度为 *p*，小数为 *s* (*p* 是小数数字的总长度，*s* 是小数点右边的数字长度；*p* 的范围从 1 到 38，*s* 的范围从-84 到 127)
DATE	日期和时间值，从公元前 4712. 1. 1到公元 9999. 12. 31
LONG	最大 2G 的可变长度字符数据
CLOB	最大 4G 的字符数据
RAW(*size*)	原始二进制数据（必须指定最大长度，最大长度为 2000）
LONG RAW	可变长度原始二进制数据，最大 2G
BLOB	二进制数据，最大 4G
BFILE	二进制数据存储在一个外部文件中；最大到 4G
ROWID	十六进制串，表示行在所在的表中唯一的行地址。该数据类型主要用于返回 ROWID 伪列

幻灯片 12

数据类型 (续)

- 在用子查询创建表时，LONG 列不会被复制。
- LONG 列不能包括在 GROUP BY 或 ORDER BY 子句中。
- 在每个表中只能有一个 LONG 列。
- 在 LONG 列上不能定义约束。
- 通常用情况下使用 CLOB 列而不是 LONG 列。

教师注释

Oracle8 引入了大对象 (LOB) 数据类型，它可以存储大的和非结构化的数据，例如文本、图象、视频和空间数据，最大 4G。LONG 列可以容易地移动到 LOB 列。学生可参考 *Oracle9i Migration Release 9.0.1 Guide*。

教师注释 (9-13 页)

在下一页显示的日期和时间数据类型是 Oracle9*i* 新发布的。

幻灯片 13

日期时间数据类型

Oracle9*i* 对日期时间的增强：

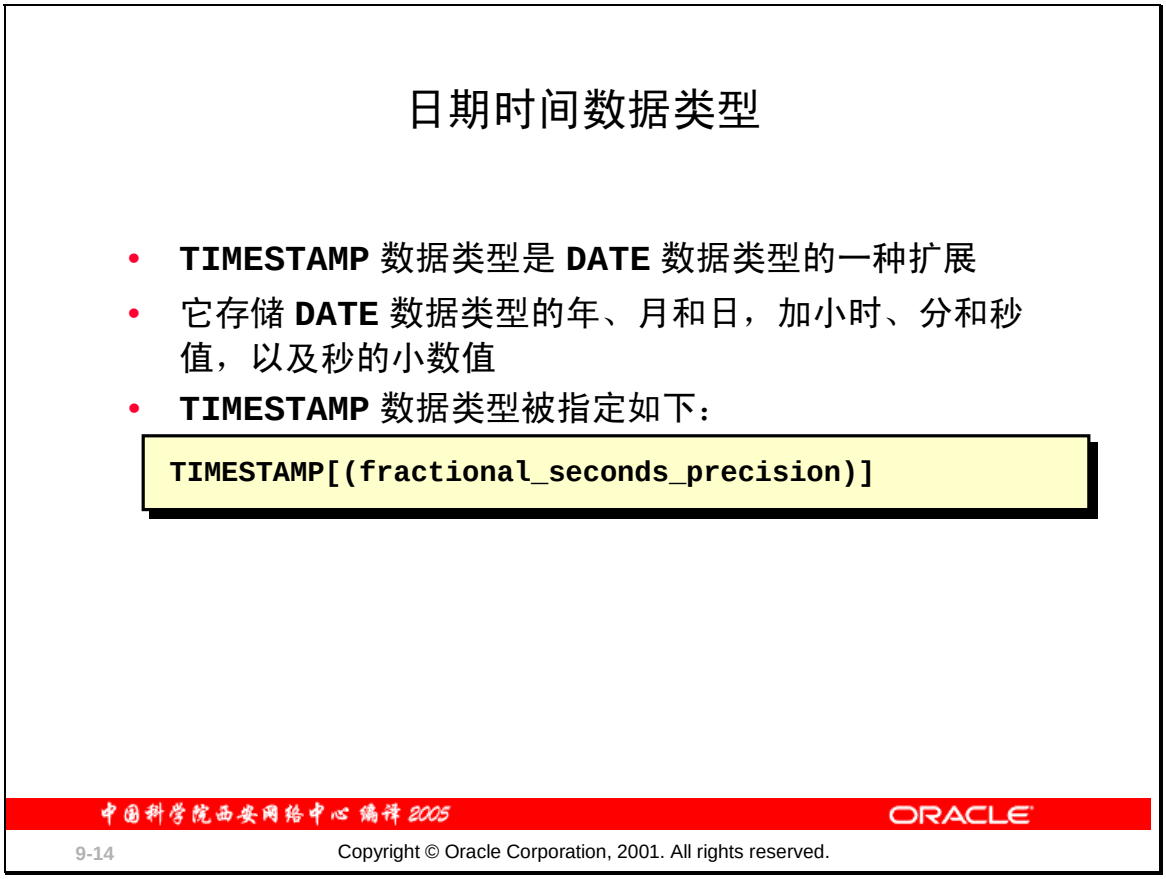
- 引入了新的日期时间数据类型
- 可用新数据类型存储
- 对时区和本地时区的增强

数据类型	说明
TIMESTAMP	带小数秒的日期
INTERVAL YEAR TO MONTH	作为年和月的时间间隔存储
INTERVAL DAY TO SECOND	作为天、小时、分和秒的时间间隔存储

其他日期时间数据类型

数据类型	说明
TIMESTAMP	允许带小数秒的时间被作为日期存储。有一些变异的数据类型。
INTERVAL YEAR TO MONTH	允许时间作为年和月的间隔被存储
INTERVAL DAY TO SECOND	允许时间作为天、小时、分和秒的间隔被存储

幻灯片 14



日期时间数据类型

fractional_seconds_precision 在秒日期时间域的小数部分随意地指定 0 到 9 个数字，默认是 6。

例

```
CREATE TABLE new_employees
(employee_id NUMBER,
 first_name VARCHAR2(15),
 last_name VARCHAR2(15),
 ...
 start_date TIMESTAMP(7),
 ...);
```

在上面的例子中，我们创建了一个表 NEW_EMPLOYEES，其中字段 start_date 的数据类型是 TIMESTAMP，精度 7 指示小数秒的精度，如果不指定，小数秒的默认精度是 6。

假定插入两行到 NEW_EMPLOYEES 表中，输出展示了显示的差异。(DATE 数据类型以 DD-MON-RR 格式显示)：

```
SELECT start_date  FROM   new_employees;
17-JUN-87 12.00.00.000000 AM
21-SEP-89 12.00.00.000000 AM
```

幻灯片 15

TIMESTAMP WITH TIME ZONE 数据类型

- **TIMESTAMP WITH TIME ZONE** 是 **TIMESTAMP** 的一个变量，它对 **TIMESTAMP** 值进行一个时区转换
- 在本地时间和 UTC 之间，小时和分钟的时区转换是不同的

```
TIMESTAMP[(fractional_seconds_precision)]
WITH TIME ZONE
```


日期时间数据类型

UTC 代表协调世界时—以前的格林尼治标准时间。如果两个 TIMESTAMP WITH TIME ZONE 在 UTC 中代表同一时刻，它们的值被认为是相同的，而不管存储在数据中的 TIME ZONE 偏移。

因为 TIMESTAMP WITH TIME ZONE 也可以存储时区信息，它特别适合记录那些必须组合或协调地理区域的日期信息。

例如，

TIMESTAMP '1999-04-15 8:00:00 -8:00'

与

TIMESTAMP '1999-04-15 11:00:00 -5:00'

是相同的。

美国西部标准时间 8:00 a.m. 和东部标准时间 11:00 a.m. 是相同的。

该时间也可以被指定为：

TIMESTAMP '1999-04-15 8:00:00 US/Pacific'

注：小数秒精度指定 SECOND 日期时间字段的小数部分数字的数目，其范围是 0 到 9，默认是 6。

幻灯片 16

TIMESTAMP WITH LOCAL TIME 数据类型

- **TIMESTAMP WITH LOCAL TIME ZONE**是 **TIMESTAMP** 的另一个变量，它对 **TIMESTAMP** 值进行一个时区转换
- 存储在数据库中的数据被格式化为数据库时区
- 时区的转换不被作为列数据的一部分存储；Oracle 以本地会话时区返回数据
- **TIMESTAMP WITH LOCAL TIME ZONE** 数据类型被如下指定：

```
TIMESTAMP[(fractional_seconds_precision)]
WITH LOCAL TIME ZONE
```


日期时间数据类型

不像 TIMESTAMP WITH TIME ZONE，你可以指定 TIMESTAMP WITH LOCAL TIME ZONE 类型作为一个主键或唯一键的一部分。在本地时间和 UTC 之间的时区转换(小时或分钟) 是不同的，对于 TIMESTAMP WITH LOCAL TIME ZONE 是非文字的。

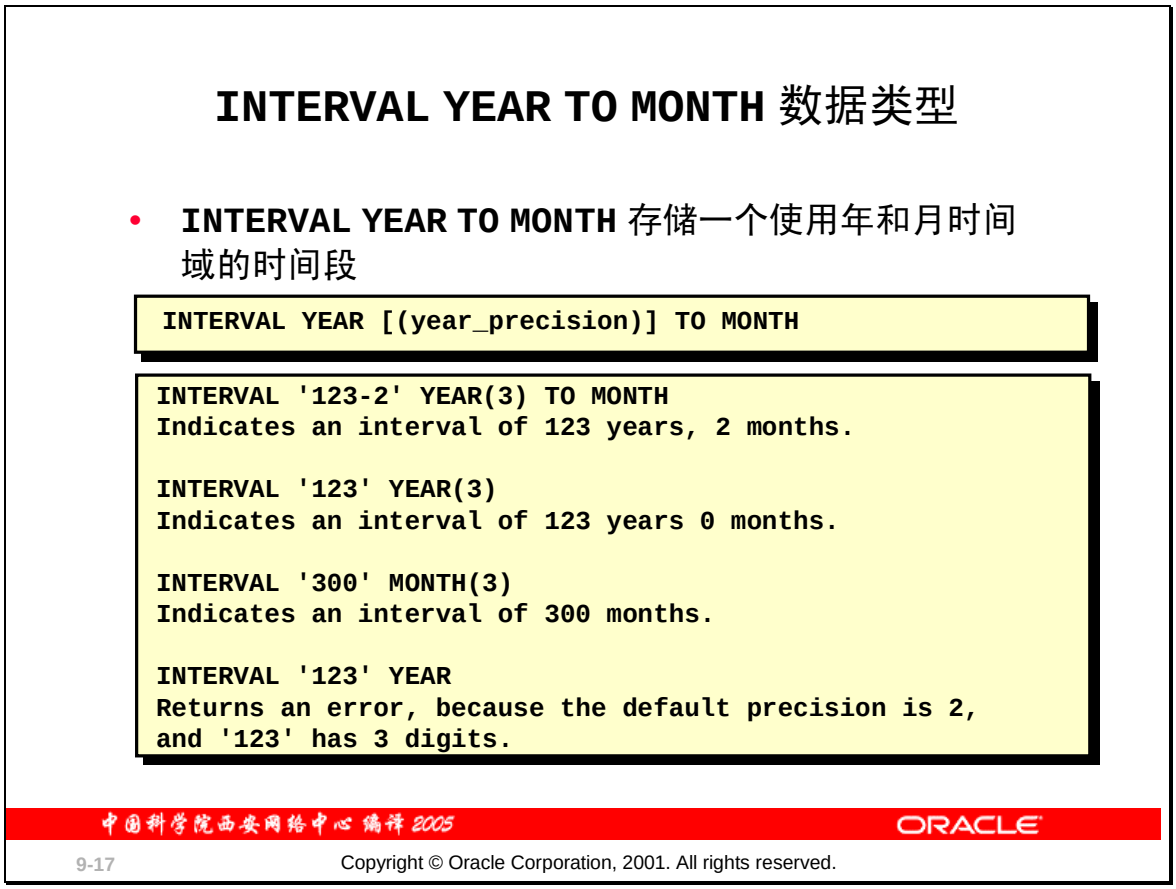
注：小数秒精度指定 SECOND 日期时间字段的小数部分数字的数目，其范围是 0 到 9，默认是 6。

例

```
CREATE TABLE time_example
 (order_date TIMESTAMP WITH LOCAL TIME ZONE);

INSERT INTO time_example VALUES('15-NOV-00 09:34:28 AM');

SELECT *
FROM   time_example;

order_date
---------------------------
15-NOV-00 09.34.28.000000 AM
```

TIMESTAMP WITH LOCAL TIME ZONE 类型适合于两层应用程序，在其中你可以用客户系统的时区显示日期和时间。

幻灯片 17



INTERVAL YEAR TO MONTH 数据类型

INTERVAL YEAR TO MONTH 用年和月日期时间字段存储一段时间。

用 INTERVAL YEAR TO MONTH 表示两个日期时间值的差，该差值只有年和月的部分。例如，你可能用该值设置一个往后 120 个月的提醒日期，或检查是否 从某个特定的日期后 6 月已过去。

指定 INTERVAL YEAR TO MONTH 如下：

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

在语法中：

year_precision 是在 YEAR 日期时间字段中数字的数目,年精度的默认值是 2。

例

```
CREATE TABLE time_example2
(loan_duration INTERVAL YEAR (3) TO MONTH);

INSERT INTO time_example2 (loan_duration)
VALUES (INTERVAL '120' MONTH(3));

SELECT TO_CHAR( sysdate+loan_duration, 'dd-mon-yyyy')
FROM   time_example2;          --today's date is 26-Sep-2001
```

限制

前面的部分要大于后面的部分，例如：INTERVAL '0-1' **MONTH** TO **YEAR** 是无效的，必须写成：INTERVAL '0-1' **YEAR** TO **MONTH**。

幻灯片 18

INTERVAL DAY TO SECOND 数据类型

- **INTERVAL DAY TO SECOND** 按照天、小时、分和秒存储一段时间

```
INTERVAL DAY [(day_precision)]
   TO SECOND [(fractional_seconds_precision)]
```

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)
Indicates 4 days, 5 hours, 12 minutes, 10 seconds,
and 222 thousandths of a second.INTERVAL '123' YEAR(3).

INTERVAL '7' DAY
Indicates 7 days.

INTERVAL '180' DAY(3)
Indicates 180 days.
```


INTERVAL DAY TO SECOND 数据类型

INTERVAL DAY TO SECOND 根据天、小时、分和秒存储一段时间。

用 INTERVAL DAY TO SECOND 来表示两个日期时间值精确的差。例如，你可能用该值设置一个往后 36 个小时的提醒，或记录一个赛跑的开始和结束之间的时间。为了表示很长的时间跨度，包括很多年，用很高的精度，你可以用一个很大的值表示天的一部分。

指定 INTERVAL DAY TO SECOND 如下：

```
INTERVAL DAY [(day_precision)]
TO SECOND [(fractional_seconds_precision)]
```

在语法中：

day_precision 是在 DAY 日期时间字段中数字的数目，可接受的值的范围是 0 到 9，默认是 2。

fractional_seconds_precision 是在 SECOND 日期时间字段中数字的数目，可接受的值的范围是 0 到 9，默认是 6。

幻灯片 19

INTERVAL DAY TO SECOND 数据类型

- **INTERVAL DAY TO SECOND** 按照天、小时、分和秒存储一段时间

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)
Indicates 4 days, 5 hours, 12 minutes, 10 seconds,
and 222 thousandths of a second.

INTERVAL '4 5:12' DAY TO MINUTE
Indicates 4 days, 5 hours and 12 minutes.

INTERVAL '400 5' DAY(3) TO HOUR
Indicates 400 days 5 hours.

INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)
indicates 11 hours, 12 minutes, and 10.2222222 seconds.
```


INTERVAL DAY TO SECOND 数据类型

例

```
CREATE TABLE time_example3
(day_duration INTERVAL DAY (3) TO SECOND);

INSERT INTO time_example3 (day_duration)
VALUES (INTERVAL '180' DAY(3));

SELECT sysdate + day_duration "Half Year"
FROM   time_example3;       --今天的日期是 26-Sep-2001
```

幻灯片 20

用子查询语法创建表

- 用子查询选项组合 **CREATE TABLE** 语句创建表并插入行

```
CREATE TABLE table
       [(column, column...)]
AS subquery;
```

匹配表中指定的列数和子查询的列数

- 用列名和默认值定义列

用另一个表的行创建表

创建表的第二种方法是用 AS *subquery* 子句，该方法既可以创建表还可以将从子查询返回的行插入新创建的表中。

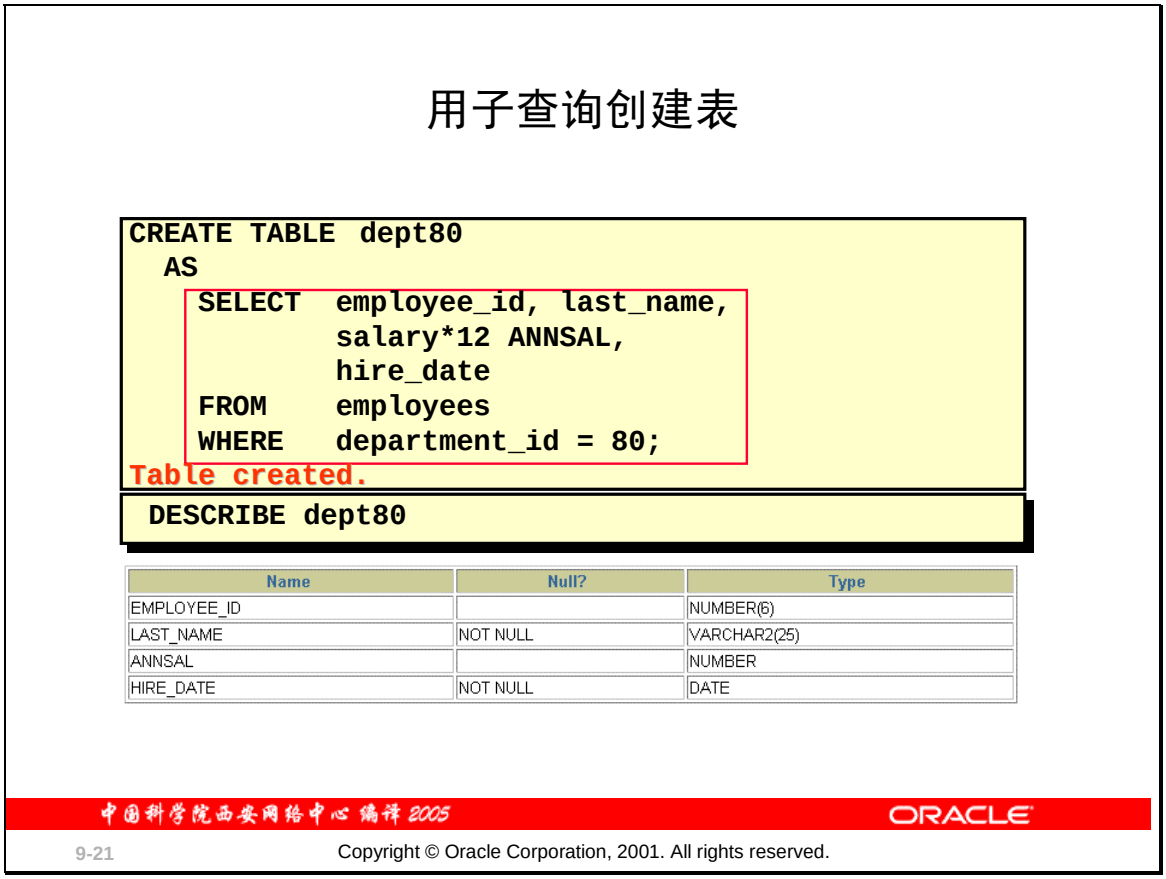
在语法中：

table	是表的名字
column	是列的名字，默认值和完整性约束
subquery	是 SELECT 语句，用来定义将要被插入到新表中的行集

原则

- 被创建的表要带指定的列名，并且由 SELECT 语句返回的行被插入到新表中。
- 字段的定义只能包括列名和默认值。
- 如果给出了指定的列，列的数目必须等于子查询的 SELECT 列表的列数目。
- 如果没有给出了指定的列，表的列名应和子查询中的列名是相同的。
- 完整性规则不会被传递到新表中，仅列的数据类型被定义。

幻灯片 21



用另一个表的行创建表 (续)

幻灯片的例子创建一个名为 DEPT80 的表，该表包含所有工作在部门 80 的雇员的详细资料，注意 DEPT80 表的数据来自 EMPLOYEES 表。

你可以检验一个数据库表的存在，并且用 *i*SQL*Plus DESCRIBE 命令检查列的定义对于在 SELECT 列表中的表达式给出别名。在幻灯片中表达式 SALARY*12 被给予别名 ANNSAL。如果没有别名会产生错误：

```
ERROR at line 3:
ORA-00998: must name this expression with a column alias
```

教师注释

为了用一个已存在的表的相同结构创建一个新表，而不用旧表的数据，用带 WHERE 子句的子查询，该子查询的永远是假，例如：

```
CREATE TABLE COPY_TABLE AS
  (SELECT *
   FROM   employees
   WHERE  1 = 2);
```

幻灯片 22

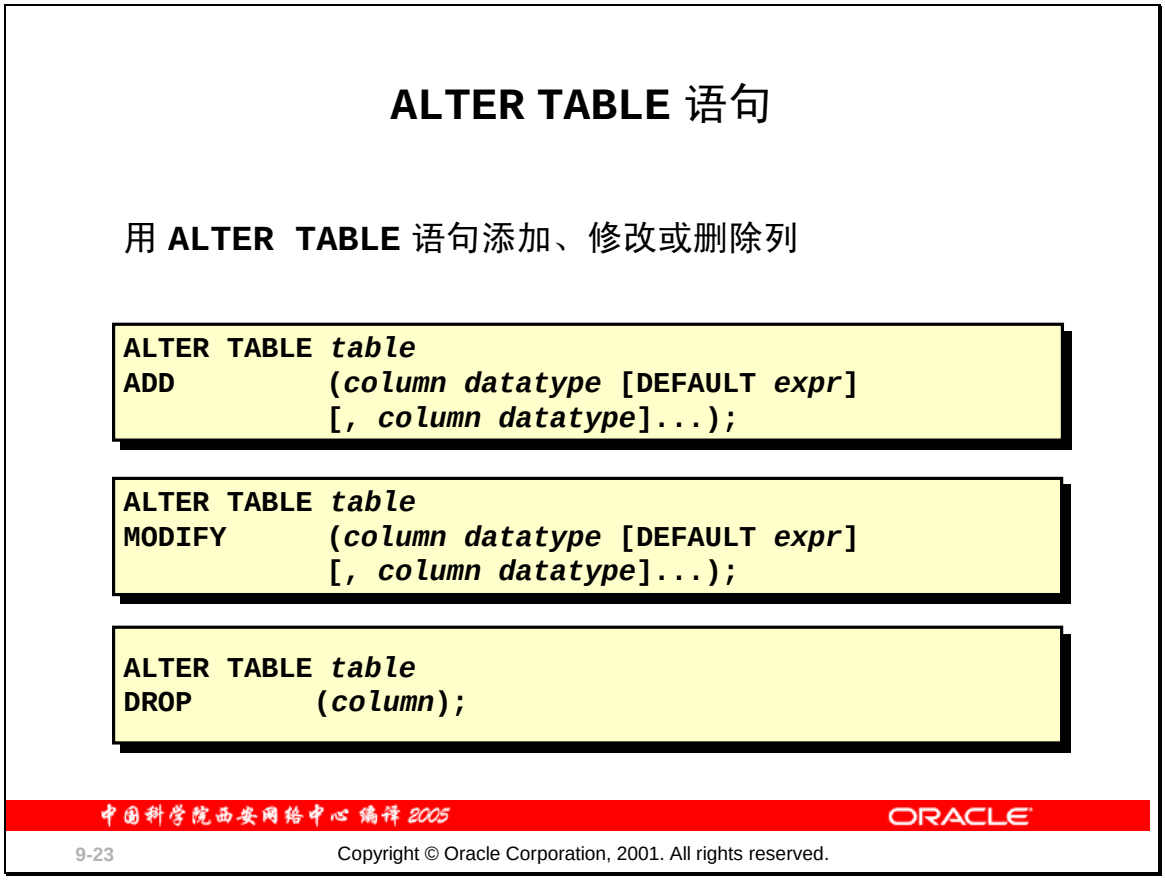
ALTER TABLE 语句

用 **ALTER TABLE** 语句来：

- 添加一个新列
- 修改一个已存在的列
- 为新列定义一个默认值
- 删除一个列

ALTER TABLE 语句

在你创建一个表后，你可能需要改变表的结构，因为，你可能遗漏了一个列，或者列的定义需要改变，或者某些列需要删除，你可以用 ALTER TABLE 语句来做这些改变。



ALTER TABLE 语句 (续)

你可以用 ALTER TABLE 语句添加、修改和删除表中的列。

在语法中：

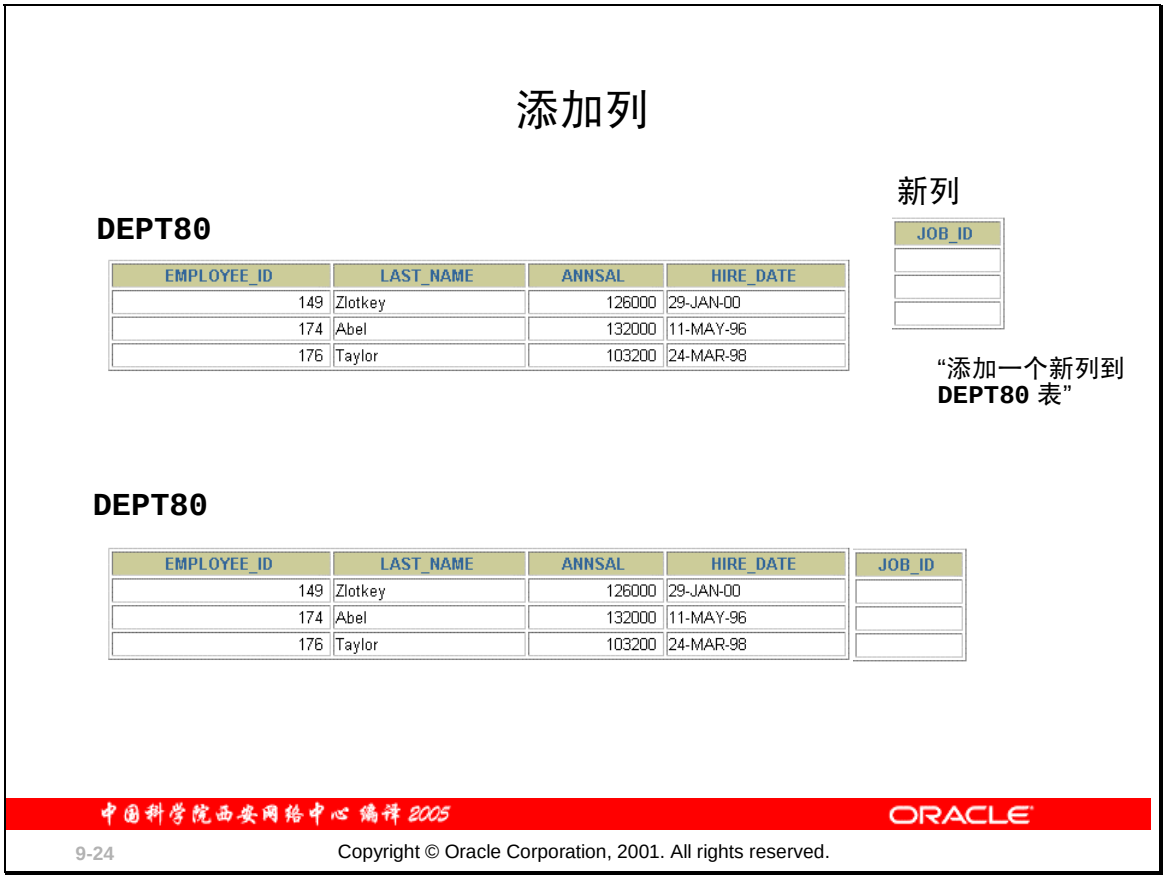
table	是表的名字
ADD|MODIFY|DROP	是修改类型
column	是新列的名字
datatype	是新列的数据类型和长度
DEFAULT *expr*	为一个新行指定默认值

注：幻灯片给出的是删减的 ALTER TABLE 语法。更多有关 ALTER TABLE 的内容在后面的课程中。

教师注释

在 Oracle8*i* 和以后的版本中，有一些 ALTER TABLE 命令的新选项，包括从表中删除列的能力，这些内容包含在本课程的后面。

幻灯片 24



添加列

上图中添加 JOB_ID 列到 DEPT80 表中，注意新的列已成为表中的最后一列。

幻灯片 25

添加新列

- 用 **ADD** 字句添加列

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
Table altered.
```

- 新列成为最后的列

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

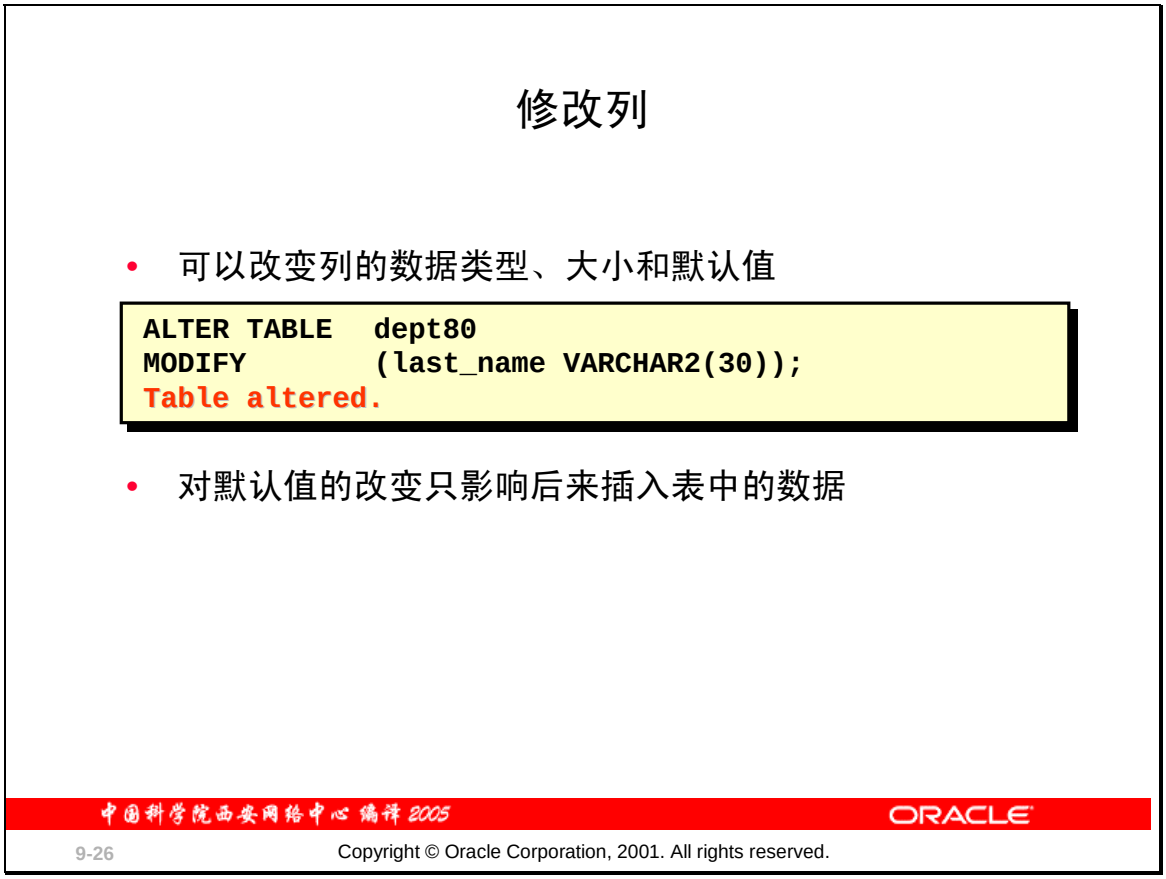
添加新列的原则

- 你可以添加或修改列。
- 你不能指定新添加的列出的位置，新列将成为最后一列。

幻灯片中例子添加一个名为 JOB_ID 的新列到 DEPT80 表中，JOB_ID 列成为表中的最后一列。

注：如果一个表在添加新列时已包含有行，那么，所有行的新列被初始化为空。

幻灯片 26



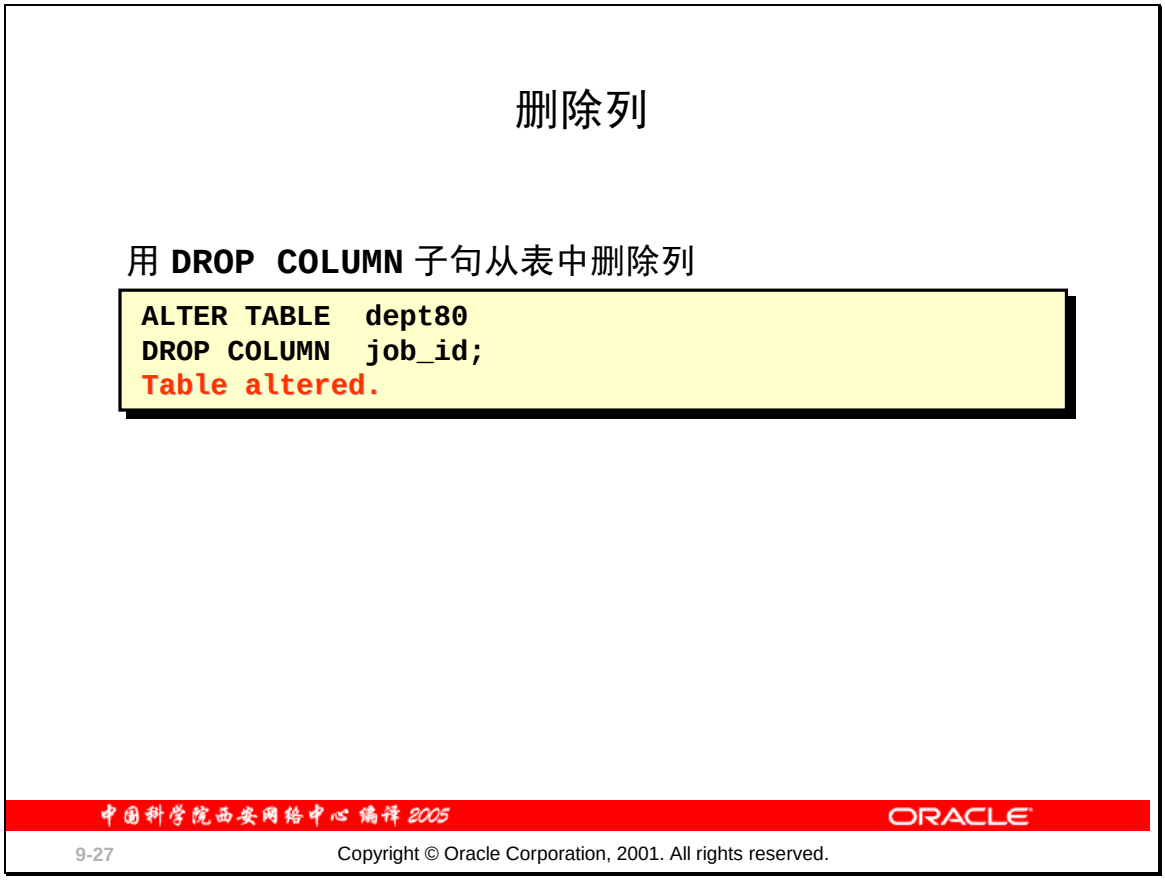
修改列

你可以用带 MODIFY 子句的 ALTER TABLE 语句修改一个列，列的修改包括修改列的数据类型，大小和默认值。

原则

- 你可以增加宽度或一个数字列的精度。
- 你可以增加数字列或字符列的宽度。
- 你可以减少一个列的宽度，但仅在列中只包含空值或表中没有行时。
- 你可以改变数据类型，但仅在列中只包含空值时。
- 你可以转换一个 CHAR 列到 VARCHAR2 数据类型或转换一个 VARCHAR2 列到 CHAR 数据类型仅当列中只包含空值时，或者你不改变列的大小时。
- 对默认值的改变仅影响以后插入的列。

幻灯片 27



删除列

你可以用带 DROP COLUMN 子句的 ALTER TABLE 语句从表中删除列，该特性在 Oracle8*i* 及以后的版本中可用。

原则

- 列可以有也可以没有数据。
- 用 ALTER TABLE 语句，一次只能有一列被删除。
- 表被修改后必须至少保留一列。
- 一旦一列被删除，它不能再恢复。

教师注释

当一列从表中被删除时，该表中任何其他的被用 SET UNUSED 选项标记列也被删除。

幻灯片 28

SET UNUSED 选项

- 用 **SET UNUSED** 选项标记一个或多个未使用的列
- 用 **DROP UNUSED COLUMNS** 选项删除被标记为未使用的列

```
ALTER TABLE   table
SET   UNUSED (column);
OR
ALTER TABLE  table
SET   UNUSED COLUMN column;
```

```
ALTER TABLE table
DROP  UNUSED COLUMNS;
```


SET UNUSED 选项

SET UNUSED 选项标记一个或多个列作为不使用的，所以，当需求的系统资源较低时他们可以被删除，该特性在 Oracle8*i* 和以后的版本中有效。指定该子句不会真的从表的每一行中删除目标列 (即，它不会恢复这些列所使用的磁盘空间)，因此，SET UNUSED 选项标记的执行响应时间会比执行 DROP 子句快一些。不使用的列就好象它被删除了一样的被处理，即使他们的列数据还保留在表的行中。在一列已经被标记为不使用后，你就不能访问该列了。一个 SELECT *查询不会从标记为不使用的列返回数据。另外，在使用 DESCRIBE 命令时，被标记为不使用的列的名字和类型将不再显示，并且你可以用一个与不使用列相同的名字添加一个新列到表中。SET UNUSED 信息被存储在 USER_UNUSED_COL_TABS 字典视图中。

DROP UNUSED COLUMNS 选项

DROP UNUSED COLUMNS 从表中删除当前所有被标记为不使用的列，当你想要从表中的不使用列回收额外的磁盘空间时你可以用该语句，如果表中不包含不使用列，该语句不返回错误。

```
ALTER TABLE  dept80
SET   UNUSED (last_name);
Table altered.
ALTER TABLE  dept80
DROP  UNUSED COLUMNS;
Table altered.
```

幻灯片 29

删除表

- 在表中的所有数据和结构都被删除
- 任何未决的事务都被提交
- 所有的索引被删除
- 你 *不能* 回退 **DROP TABLE** 语句

```
DROP TABLE dept80;
Table dropped.
```


删除表

DROP TABLE 语句删除 Oracle 表定义，当你删除一个表时，数据库丢失表中所有的数据，并且所有与其相关的索引也被删除。

语法

```
DROP TABLE table
```

在语法中：

table 是表的名字

原则

- 所有的数据从表中删除。
- 任何视图和同义词被保留但无效。
- 任何未决的事务被提交。
- 只有表的创建者或具有 DROP ANY TABLE 权限的用户才能 删除表。

注：DROP TABLE 语句，一旦被执行，就不能撤回。当你发布 DROP TABLE 语句时，Oracle 服务器不询问其行为，如果你拥有该表或有一个高级权限，那么，该表立即被删除。当使用所有 DDL 语句时，DROP TABLE 被自动提交。

幻灯片 30

改变一个对象的名字

- 执行 **RENAME** 语句，改变一个表、视图、序列或同义词

```
RENAME dept TO detail_dept;
Table renamed.
```

- 你必须是对象的所有者

重命名表

另外的 DDL 语句包括 RENAME 语句，该语句被用于改变表、视图序列或同义词的名字。

语法

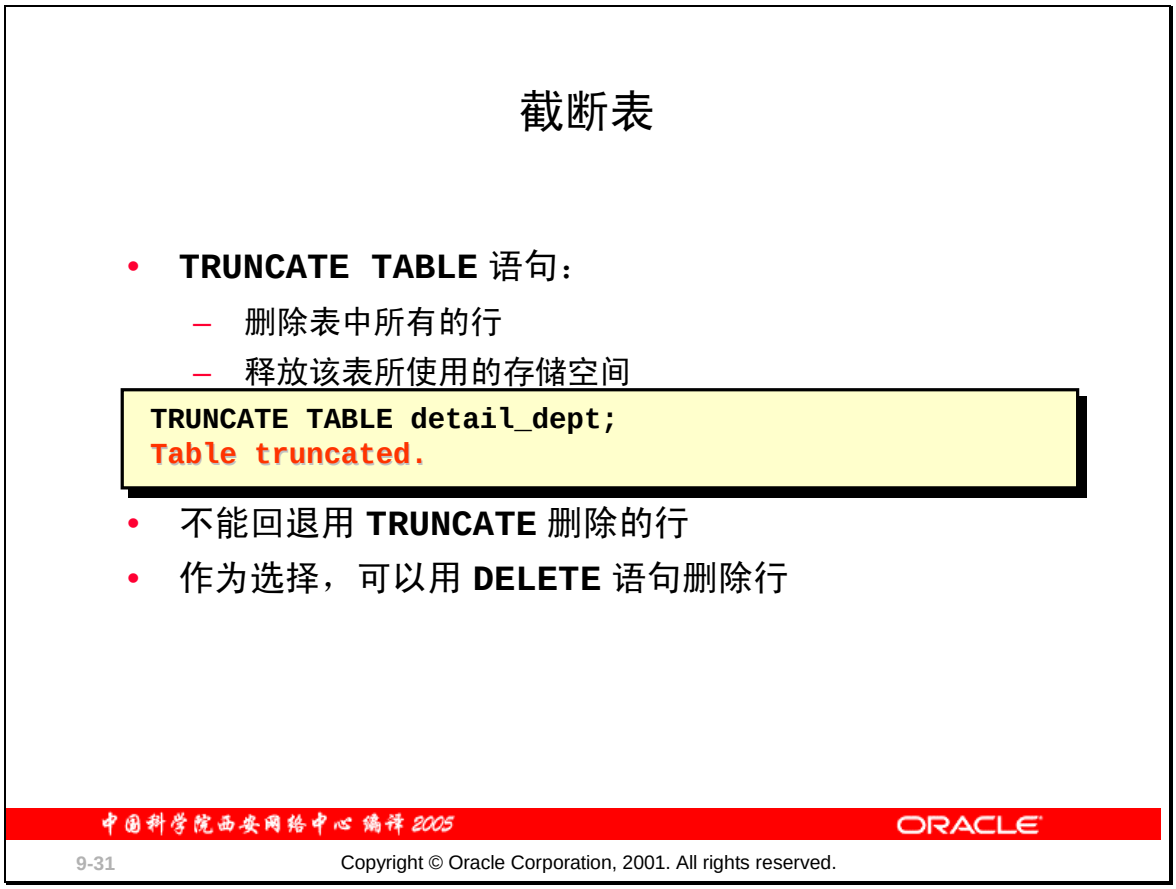
```
RENAME   old_name  TO  new_name;
```

在语法中：

old_name	是表、视图序列或同义词的旧名字。
new_name	是表、视图序列或同义词的新名字。

你必须是你重命名的对象的所有者。

幻灯片 31



截断表

另一个 DDL 语句是 TRUNCATE TABLE 语句，该语句被用于从表中删除所有的行，并且释放该表所使用的存储空间。在使用 TRUNCATE TABLE 语句时，你不能回退已删除的行。

语法

```
TRUNCATE  TABLE   table;
```

在语法中：

table 是表的名字

你必须是表的所有者，或者有 DELETE TABLE 系统权限来截断表。

DELETE 语句也可以从表中删除所有的行，但它不能释放存储空间。TRUNCATE 命令更快一些，用 TRUNCATE 语句删除行比用 DELETE 语句删除同样的行快一些，原因如下：

- TRUNCATE 语句是数据定义 (DDL) 语句。并且不产生回滚信息。
- 截断一个表不触发表的删除触发器。
- 如果表是一个引用完整性约束的父表，你不能截断该表，在发布 TRUNCATE 语句之前禁用约束。

幻灯片 32

添加注释到表中

- 用 **COMMENT** 语句添加注释到一个表或列中

```
COMMENT ON TABLE employees
IS 'Employee Information';
Comment created.
```

- 注释能够通过数据字典视图查看：
  - **ALL_COL_COMMENTS**
  - **USER_COL_COMMENTS**
  - **ALL_TAB_COMMENTS**
  - **USER_TAB_COMMENTS**

添加注释到表中

你可以用 COMMENT 语句给一个列、表、视图或快照添加一个最多 2K 字节的注释。注释被存储在数据字典中，并且可以通过下面的数据字典视图查看 COMMENTS 列：

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

语法

```
COMMENT ON TABLE table | COLUMN table.column
   IS 'text';
```

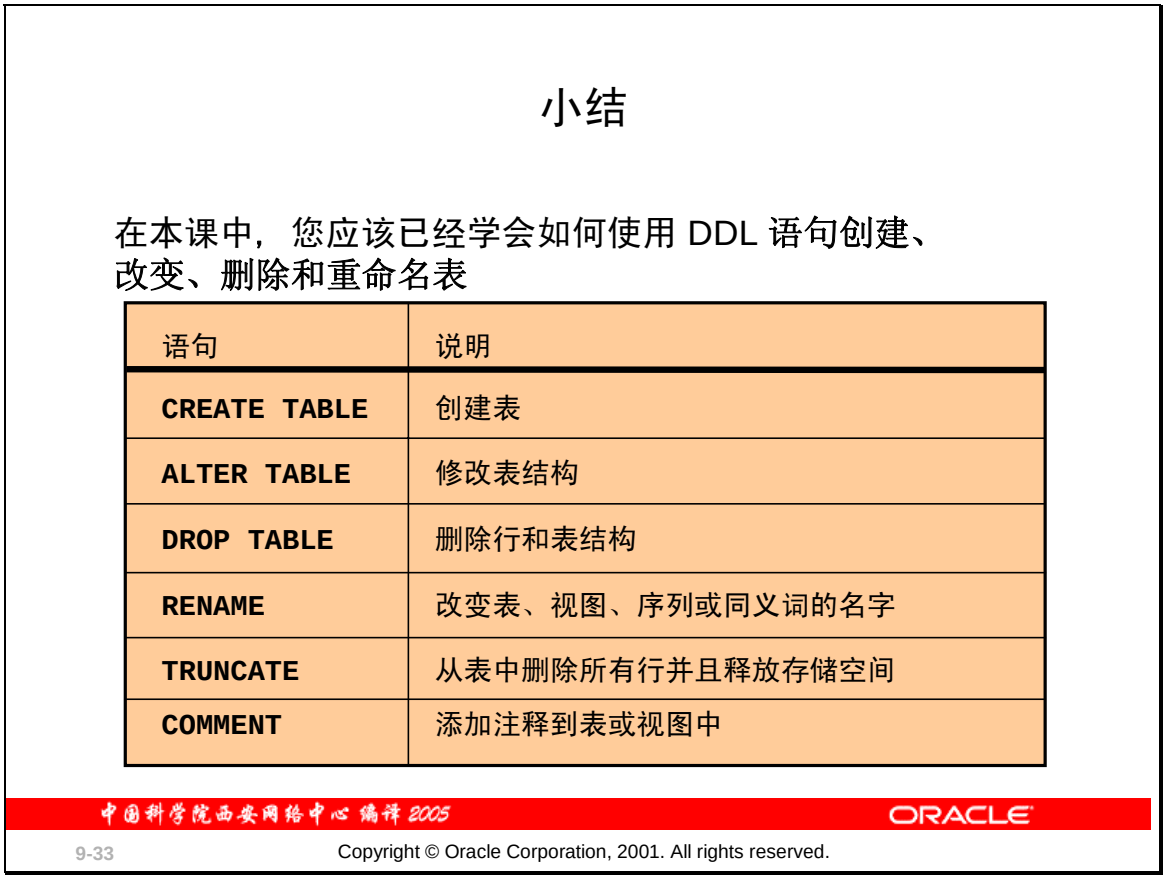
在语法中：

table	是表的名字
column	是表中列的名字
text	是注释的文本

你可以用设置注释为空串 ('') 的办法从数据库中删除一个注释：

```
COMMENT ON TABLE  employees IS '';
```

幻灯片 33



小结

在本课中，您应该已经学会如何使用 DDL 命令来创建、修改、删除和重命名表。你也学会了怎样截断表和添加注释到一个表。

创建表

- 创建表。
- 用子查询基于另一个表创建表。

修改表

- 修改表结构。
- 修改列宽，改变列数据类型和添加列。

删除表

- 删除行和表结构。
- 一旦执行，该语句不能回滚。

重命名

- 重命名一个表、视图、序列或同义词。

截断

- 从表中删除所有行，并且释放该表已使用的存储空间。
- DELETE 语句只删除行。

注释

- 添加注释到表或列。
- 查询数据字典来查看注释。

幻灯片 34

练习 9 概览

本章练习包括下面的主题：

- 创建新表
- 用 **CREATE TABLE AS** 语法创建新表
- 修改列定义
- 验证已经存在的表
- 添加注释到表中
- 删除表
- 改变表

练习 9 概览

用 CREATE TABLE 语句创建新表，确认新表被添加到数据库中。在命令文件中的创建语句，然后执行命令文件来创建表。

教师注释

解释什么是表的实例图表，告诉学生怎样解释一个表实例图表，说明他们需要查看的条目有列名、数据类型和域的长度。其他的条目是可选的，并且如果这些条目存在，他们需要作为表定义的一部分被强制为一体。

对学生指出，对于本课程，练习是基于专门创建的表的。他们需要小心，不要修改方案中的其他表。

幻灯片 35

练习 9

1. 按下面的要求创建 DEPT 表，将创建命令存入脚本文件 lab9_1.sql，然后执行该脚本中的语句来创建表，确认表已被创建。

列名	ID	NAME
键类型		
空/唯一性		
FK 表		
FK 列		
数据类型	Number	VARCHAR2
长度	7	25

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)

```
CREATE TABLE dept
(id NUMBER(7),
name VARCHAR2(25));

DESCRIBE dept
```

2. 从 DEPARTMENTS 取数据填充的 DEPT 表，只包括你需要的行。

```
INSERT INTO dept
SELECT department_id, department_name
FROM departments;
```

3. 根据下面的表创建 EMP 表，将语句保存到脚本文件 lab9_3.sql 中，然后执行脚本中的语句创建表，确认表已被创建。

列名	ID	LAST_NAME	FIRST_NAME	DEPT_ID
键类型				
空/唯一性				
FK 表				
FK 列				
数据类型	Number	VARCHAR2	VARCHAR2	Number
长度	7	25	25	7

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

```
CREATE TABLE emp
(id NUMBER(7),
last_name VARCHAR2(25),
first_name VARCHAR2(25),
dept_id NUMBER(7));

DESCRIBE emp
```

4. 修改 EMP 表允许更长的雇员名字 (last names)，确认你的修改。

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

```
ALTER TABLE emp
MODIFY (last_name VARCHAR2(50));
DESCRIBE emp
```

5. 确认 DEPT 表和 EMP 表被存储在数据字典中。(*提示：* USER_TABLES)

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

```
SELECT table_name
FROM user_tables
WHERE table_name IN ('DEPT', 'EMP');
```

6. 基于 EMPLOYEES 表的结构创建 EMPLOYEES2 表，只包括 EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY 和 DEPARTMENT_ID 列，分别将新表中的列命名为 ID, FIRST_NAME, LAST_NAME, SALARY 和 DEPT_ID。

```
CREATE TABLE employees2 AS
SELECT employee_id id, first_name, last_name, salary,
department_id dept_id
FROM employees;
```

7. 删除 EMP 表。

```
DROP TABLE emp;
```

8. 重命名 EMPLOYEES2 表为 EMP。

```
RENAME employees2 TO emp;
```

9. 添加一个列到 DEPT 和 EMP 表定义，用来描述表。确认你添加的列在数据字典中。

```
COMMENT ON TABLE emp IS 'Employee Information';
COMMENT ON TABLE dept IS 'Department Information';

SELECT *
FROM user_tab_comments
WHERE table_name = 'DEPT'
OR table_name = 'EMP';
```

10. 从 EMP 表中删除 FIRST_NAME 列，检查表的定义来确认你的修改。

```
ALTER TABLE emp
DROP COLUMN FIRST_NAME;

DESCRIBE emp
```

11. 在 EMP 表中，将 DEPT_ID column 列标记为 UNUSED，检查表定义以确认你的修改。

```
ALTER TABLE emp
SET UNUSED (dept_id);

DESCRIBE emp
```

12. 从 EMP 表中删除所有 UNUSED 列，检查表的定义以确认你的修改。

```
ALTER TABLE emp
DROP UNUSED COLUMNS;

DESCRIBE emp
```

幻灯片 37

教师注释（9-5 页）

有一个选项，CREATE GLOBAL TEMPORARY TABLE，定义一个表为临时表，并且对所有的会话可见，在临时表中的数据仅对插入数据到表中的会话是可见。

临时表可以象常规表那样有一个持久化的定义，而且它既可以包含会话指定也可以包含事务指定的数据，用 ON COMMIT 关键字确定数据是会话指定的还是事务指定的。临时表使用临时段，不象永久表，临时表和及其索引在它们被创建时不会自动分配段，而是在执行第一个 INSERT 语句或 CREATE TABLE AS SELECT 语句时段被分配，这意味者在第一个 INSERT 之前，如果有一个 SELECT，UPDATE 或 DELETE 语句被执行，那么，表看起来是空的。只有当没有会话被绑定到一个临时表上的时候，你可以在该临时表上执行 DDL 命令 (ALTER TABLE，DROP TABLE，CREATE INDEX，等等) 。当一个 INSERT 在临时表上被执行时，一个会话开始绑定到一个临时表，该会话被一个 TRUNCATE，在会话终止时，或对一个事务指定的临时表做 COMMIT 或 ABORT 操作时解除绑定。对于事务指定的临时表，临时段在事务结束时被释放；对于会话指定的临时表，临时段在会话结束时被释放。

关于临时表和 CREATE TABLE 的更多信息，查阅 *Oracle9i Concepts*，“临时表”。