

幻灯片 1

# 12

## 其它数据库对象

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
	20 分钟	讲演
	20 分钟	练习
	40 分钟	总共

## 目标

完成本课后，您应当能够执行下列操作：

- 创建、维护和使用序列
- 创建和维护索引
- 创建私有和公有同义词

### 课程目标

在本课中，你将学习怎样创建和维护一些其它用途的数据库对象，这些对象包括序列、索引和同义词。

## 数据库对象

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

### 数据库对象

许多应用程序要求使用唯一的数字作为主键的值。你即可以在应用程序中构建代码来处理这种需求，也可以用一个序列来产生唯一的数字。

如果你想要增进某些查询的性能，你应该考虑创建一个索引。你也可以用索引在列或列的集合上强制唯一性。

你可以用同义词为对象提供可替代的名字。

## 什么是序列?

序列:

- 是自动产生的唯一的数
- 是可共享的对象
- 典型的用途是创建一个主键值
- 可以代替应用程序编号
- 当使用高速缓存存储器时，访问序列值的效率提高

### 什么是序列?

序列是用户创建的数据库对象，序列可以被多个用户共享以产生唯一的整数。

序列的一个典型的用途是创建一个主键的值，它对于每一行必须是唯一的。序列由一个 Oracle 内部程序产生并增加或减少。

序列是一个节省时间的对象，因为它可以减少应用程序中产生序列程序的代码量。

序列号独立于表被存储和产生，因此，相同的序列可以被多个表使用。

## CREATE SEQUENCE 语句语法

定义一个序列来自动产生有顺序的数：

```
CREATE SEQUENCE sequence
  [ INCREMENT BY n ]
  [ START WITH n ]
  [ { MAXVALUE n | NOMAXVALUE } ]
  [ { MINVALUE n | NOMINVALUE } ]
  [ { CYCLE | NOCYCLE } ]
  [ { CACHE n | NOCACHE } ];
```

### 创建序列

用 CREATE SEQUENCE 语句自动产生序列数。

在语法中：

<i>sequence</i>	是序列发生器的名字
INCREMENT BY <i>n</i>	指定序列号之间的间隔，在这儿 <i>n</i> 是一个整数（如果该子句被省略，序列增量为 1）
START WITH <i>n</i>	指定要产生的第一个序列数（如果该子句被省略，序列从 1 开始）
MAXVALUE <i>n</i>	指定序列能产生的最大值
NOMAXVALUE	对于升序序列指定 $10^{27}$ 为最大值，对于降序序列指定 -1 为最大值（这是默认选项）
MINVALUE <i>n</i>	指定最小序列值
NOMINVALUE	对于升序序列指定 1 为最小值，对于降序序列指定 $-(10^{26})$ 为最小值（这是默认选项）
CYCLE   NOCYCLE	指定序列在达到它的最大或最小值之后，是否继续产生（NOCYCLE 是默认选项）
CACHE <i>n</i>   NOCACHE	指定 Oracle 服务器预先分配多少值，并且保持在内存中（默认情况下，Oracle 服务器缓冲 20 个值）

## 创建序列

- 创建一个序列，命名为 **DEPT\_DEPTID\_SEQ**，用于 **DEPARTMENTS** 表的主键
- 使用非 **CYCLE** 选项

```
CREATE SEQUENCE dept_deptid_seq  
    INCREMENT BY 10  
    START WITH 120  
    MAXVALUE 9999  
    NOCACHE  
    NOCYCLE;  
  
Sequence created.
```

### 创建序列 (续)

幻灯片中的例子创建一个序列，并被命名为 **DEPT\_DEPTID\_SEQ**，该序列用于 **DEPARTMENTS** 表的 **DEPARTMENT\_ID** 列，该序列从 120 开始，不允许高速缓冲的，不循环。

如果序列用于产生主键值，不使用 **CYCLE** 选项，除非你有一个可靠的机制比序列循环更快地清除旧的行。

更多信息，见 *Oracle9i SQL Reference*，“创建序列”

**注：**序列不依赖于一个表，通常，你应该命名序列；可是序列可以被用在任何地方，而不管它的名字。

### 教师注释

如果 **INCREMENT BY** 值是负数，序列是降序。另外，**ORDER | NOORDER** 选项可用，**ORDER** 选项保证序列值按顺序产生，如果你将序列用于产生主键值它是不重要的，该选项仅与 **Parallel Server** (并行服务) 选项有关。

如果序列值被高速缓冲，如果系统故障它们将被丢失。

## 确认序列

- 校验在 **USER\_SEQUENCES** 数据字典表中的序列值

```
SELECT  sequence_name, min_value, max_value,  
        increment_by, last_number  
FROM    user_sequences;
```

- 如果 **NOCACHE** 被指定, **LAST\_NUMBER** 列显示下一个可用序列数

### 确认序列

一旦创建了序列,它就被文本化在数据字典中。因为序列是一个数据库对象,你可以在 **USER\_OBJECTS** 数据字典表中识别它。

你也可以从 **USER\_SEQUENCES** 数据字典视图中用选择确认序列的设置。

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPARTMENTS_SEQ	1	9990	10	280
DEPT_DEPTID_SEQ	1	9999	10	120
EMPLOYEES_SEQ	1	1.0000E+27	1	207
LOCATIONS_SEQ	1	9900	100	3300

### 教师注释

演示: 12\_dd.sql

目的: 举例说明 **USER\_SEQUENCES** 数据字典视图和它的内容。

## NEXTVAL 和 CURRVAL 伪列

- **NEXTVAL** 返回下一个可用的序列值，它每次返回一个唯一的被引用值，即使对于不同的用户也是如此
- **CURRVAL** 获得当前的序列值
- 在 **CURRVAL** 获得一个值以前，**NEXTVAL** 对该序列必须发布

中国科学院西安网络中心 编译 2005

ORACLE

12-8

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用序列

在创建序列后，它产生连续的数给你在表中使用。用 **NEXTVAL** 和 **CURRVAL** 伪列引用序列值。

#### NEXTVAL 和 CURRVAL 伪列

**NEXTVAL** 伪列用于从指定的序列中取回连续的序列数的下一个值。你必须用序列名限定 **NEXTVAL**，当你引用 `sequence.NEXTVAL` 时，一个新的序列数被产生并且当前的序列数被放入 **CURRVAL**。

**CURRVAL** 伪列被用于查阅当前当前用户刚才产生的序列数，**NEXTVAL** 必须被在 **CURRVAL** 可以被引用之前用于在当前用户的会话中产生一个序列数，你必须用序列名限定 **CURRVAL**，当 `sequence.CURRVAL` 被引用时，最后返回给用户程序的值被显示。



中国科学院西安网络中心 编译 2005

ORACLE

12-9

Copyright © Oracle Corporation, 2001. All rights reserved.

### 使用 NEXTVAL 和 CURRVAL 的规则

你可以在下面的上下文中使用 NEXTVAL 和 CURRVAL:

- 一个不是子查询的一部分的 SELECT 语句的 SELECT 列表
- 在一个 INSERT 语句中子查询的 SELECT 列表
- 一个 INSERT 语句中的 VALUES 子句
- 一个 UPDATE 语句的 SET 子句

你不能在下面的上下文中使用 NEXTVAL 和 CURRVAL:

- 一个视图的 SELECT 列表
  - 一个带 DISTINCT 关键字的 SELECT 语句
  - 一个带 GROUP BY、HAVING 或 ORDER BY 子句的 SELECT 语句
  - 一个在 SELECT、DELETE 或 UPDATE 语句中的子句
  - 在 CREATE TABLE 或 ALTER TABLE 语句中的 DEFAULT 表达式
- 更多信息, 见 *Oracle9i SQL Reference*, “伪列” 部分和 “创建序列”。

### 教师注释

明确指出本页列出的规则。

## 使用序列

- 在 location ID 2500 中插入一个新部门名称 “Support”

```
INSERT INTO departments(department_id,
                        department_name, location_id)
VALUES      (dept_deptid_seq.NEXTVAL,
            'Support', 2500);
1 row created.
```

- 查看当前的 DEPT\_DEPTID\_SEQ 序列值

```
SELECT dept_deptid_seq.CURRVAL
FROM dual;
```

### 使用序列

幻灯片的例子在 DEPARTMENTS 表中插入一个新的部门。在该例子中使用 DEPT\_DEPTID\_SEQ 序列产生一个新的部门号：

你可以查看当前的序列值：

```
SELECT dept_deptid_seq.CURRVAL
FROM dual;
```

```
CURRVAL
```

```
-----
```

```
120
```

现在设想你想要雇用雇员充当新部门的职员，对所有新雇员被执行的 INSERT 语句可以包含下面的代码：

```
INSERT INTO employees (employee_id, department_id, ...)
VALUES (employees_seq.NEXTVAL, dept_deptid_seq.CURRVAL, ...);
```

注：前面的例子假定一个称为 EMPLOYEE\_SEQ 序列已经被创建用来产生新的雇员号。

## 使用序列

- 可以更快地访问缓存在存储器中的序列值
- 序列值可能产生间隙，由于：
  - 一个回退发生
  - 系统崩溃
  - 一个序列被用于另一个表
- 如果带 **NOCACHE** 创建序列，查询 **USER\_SEQUENCES** 表，可以查看下一个可用值

中国科学院西安网络中心 编译 2005

ORACLE

12-11

Copyright © Oracle Corporation, 2001. All rights reserved.

### 缓存序列值

在内存中缓冲序列可以对序列值更快地存取，在你第一次使用到序列时被移入缓存。对于下一个序列值的每次请求从缓存的序列中找回。在最后一个序列值被使用后，对于序列的下一个请求拉出另一个在内存中的序列缓存。

### 序列中的间隙

尽管序列发生器发布无间隙的序列数，但因为这种行为不依赖于提交和回退，如果你回退包含一个序列的语句，该序列数将丢失。

另一个能在序列中导致间隙产生的事件是系统崩溃，如果序列在内存中缓冲值，那么，如果系统崩溃那些值将丢失。

因为序列不直接依赖于表，所以，相同的序列可以被用于多个表，如果你这样做了，每个表都将包含序列数的间隙。

### 查看下一个可用的序列值，而不增加它

如果序列用带 **NOCACHE** 的选项创建的，就可以在序列值不增加的情况下用查询 **USER\_SEQUENCES** 表的方法，查看下一个可用的序列值。

### 教师注释

经常使用的序列用带缓存创建将增进效率，对于缓存的序列，没有办法找出下一个可用的序列将是什么，该值不实际获得和使用，因此，建议用户不要查找下一个序列值，而相信每次一个序列被用于一个 **INSERT** 语句时系统会提供一个唯一值。

## 修改序列

创建增量值，最大值，最小值，循环选项和缓存选项

```
ALTER SEQUENCE dept_deptid_seq  
        INCREMENT BY 20  
        MAXVALUE 999999  
        NOCACHE  
        NOCYCLE;  
Sequence altered.
```

### 修改序列

如果序列达到 MAXVALUE 限制，将再无来自序列的新值产生，并且你将收到一个序列已经超过 MAXVALUE 的错误指示。为了继续使用序列，你可以用 ALTER SEQUENCE 语句修改该序列。

#### 语法

```
ALTER SEQUENCE sequence  
        [ INCREMENT BY n ]  
        [ { MAXVALUE n | NOMAXVALUE } ]  
        [ { MINVALUE n | NOMINVALUE } ]  
        [ { CYCLE | NOCYCLE } ]  
        [ { CACHE n | NOCACHE } ] ;
```

在语法中：

*sequence* 是序列发生器的名字

更多信息，见 *Oracle9i SQL Reference*，“修改序列”。

## 修改序列的原则

- 你必须是该序列的所有者，或者有 **ALTER** 该序列的权限
- 只有未来的序列数受影响
- 为了以不同的数字重新开始一个序列，该序列必须被删除并且被重新创建
- 一些确认被执行

中国科学院西安网络中心 编译 2005

ORACLE

12-13

Copyright © Oracle Corporation, 2001. All rights reserved.

### 修改序列的原则

你必须是修改序列的所有者，或者有 ALTER 权限。

用 ALTER SEQUENCE 语句，只有以后的序列数会受影响。

用 ALTER SEQUENCE 语句，START WITH 选项不能被改变。为了以不同的数重新开始一个序列，该序列必须被删除和重新创建。

一些验证被执行，例如，一个新 MAXVALUE 如果小于当前的序列值就不能用。

```
ALTER SEQUENCE dept_deptid_seq
      INCREMENT BY 20
      MAXVALUE 90
      NOCACHE
      NOCYCLE;
```

```
ALTER SEQUENCE dept_deptid_seq
*
```

```
ERROR at line 1:
```

```
ORA-04009: MAXVALUE cannot be made to be less than the current
value
```

## 删除序列

- 用 **DROP SEQUENCE** 语句从数据字典中删除序列
- 序列一旦不再被引用可以被删除

```
DROP SEQUENCE dept_deptid_seq;  
Sequence dropped.
```

### 删除序列

使用 **DROP SEQUENCE** 语句从数据字典中删除一个序列。你必须是被删除序列的所有者或者有 **DROP ANY SEQUENCE** 权限来删除它。

### 语法

```
DROP SEQUENCE sequence;
```

在语法中：

*sequence* 是序列发生器的名

更多信息，见 *Oracle9i SQL Reference*，“删除序列”。

## 什么是索引?

### 索引:

- 是一个方案对象
- 由 Oracle 服务器使用, 索引用一个指针来加速行的取回
- 用快速路径访问方法来快速定位数据, 减小磁盘 I/O
- 表和它的索引是无关的
- 被 Oracle 服务器自动地使用和维护

### 索引

Oracle 服务器索引是一个方案对象, 索引能用指针加速行的取回, 索引可以被显式创建, 也可以被自动创建, 如果你在列上没有索引, 那么将发生全表扫描。

索引提供对表中行的直接和快速访问, 它的目的是用已索引的路径快速定位数据以减少磁盘 I/O。索引由 Oracle 服务器自动使用和维护, 一旦一个索引被创建, 它就不再需要用户直接管理。

索引逻辑地和物理地独立于他们索引的表, 这意味者索引可以在任何时候被创建或删除, 并且不影响基表或其它的索引。

**注:** 当你删除表时, 相应的索引也被删除。

更多信息, 见 *Oracle9i Concepts*, “方案对象” 部分, “索引” 主题。

### 教师注释

创建索引的决定是是一个全局的、高级的决定, 对于数据库管理员, 索引的创建与维护创建是一个经常性的工作。

在谓词 **WHERE** 字句中引用有索引的列, 如果没有修改带函数或表达式的列的索引。**ROWID** 是一个十六进制的串, 表示包含块定义的行地址, 行的位置在块中, 并且有数据库文件标识符, 访问任何指定行的最快的方法是引用它的 **ROWID**。

## 索引怎样被创建？

- 自动：在一个表的定义中，当定义一个 **PRIMARY KEY** 或 **UNIQUE** 约束时，一个唯一索引被自动创建
- 手动：用户能够在列上创建非唯一的索引来加速对行的访问

### 索引的类型

可以创建两种类型的索引，一种是唯一性索引：当你在一个表中定义一个列为主键，或者定义一个唯一键约束时 Oracle 服务器自动创建该索引，索引的名字习惯上是约束的名字。另一种索引类型是非唯一索引，它可以由用户创建，例如，你可以创建一个 FOREIGN KEY 列索引用于一个查询中的连接来改进数据取回的速度。

**注：**你可以手工创建唯一索引，但建议你创建一个唯一约束，这样会隐式创建一个唯一索引。



## 创建索引

- 在一个或多个列上创建索引

```
CREATE INDEX index  
ON table (column[, column]...);
```

- 改善 EMPLOYEES 表中 LAST\_NAME 列的查询访问速度

```
CREATE INDEX emp_last_name_idx  
ON          employees(last_name);  
Index created.
```

### 创建索引

用 CREATE INDEX 语句在一个或多个列上创建一个索引。

在语法中：

<i>index</i>	是索引的名字
<i>table</i>	是表的名字
<i>column</i>	是表中被索引的列的名字

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

### 教师注释

为了在你自己的方案中创建索引，你必须有 CREATE TABLE 权限。为了在任何方案中创建索引，你需要 CREATE ANY INDEX 权限或者在正在创建索引的表上的 CREATE TABLE 权限。

在语法中的另一个选项是 UNIQUE 关键字，强调你应该在表上显式地定义唯一索引。在表中可用约束代替定义唯一性索引，Oracle 服务器用自动在唯一键上定义一个唯一索引强制唯一完整性约束。

## 什么时候创建索引

你应该创建索引，如果：

- 一个列包含一个大范围的值
- 一个列包含很多的空值
- 一个或多个列经常同时在一个 **WHERE** 子句中或一个连接条件中被使用
- 表很大，并且经常的查询期望取回少于百分之 2 到 4 的行

中国科学院西安网络中心 编译 2005

ORACLE

12-18

Copyright © Oracle Corporation, 2001. All rights reserved.

### 多不总是更好

在表上建立更多的索引并不意味着更快地查询，在带索引的表上被提交的每个 DML 操作意味着索引必须更新；与表联系的索引越多，对 Oracle 服务器的影响越大，Oracle 服务器在每次 DML 操作之后必须更新所有的索引。

#### 什么时候创建索引

因此，你应该只在下面的情况下创建索引：

- 列包含一个大范围的值
- 列包含大量的空值
- 一个或多个列在 **WHERE** 子句或连接条件中被频繁使用
- 表很大并且大多数查询所期望的返回行数少于总行数的 2 - 4%

不要忘记，如果你想要强制非唯一，你应该在表中定义一个唯一的约束，然后唯一索引被自动创建。

#### 教师注释

复合索引（也称为连接索引）是在一个表中的多个列上创建的索引，在复合索引中的列可以任何顺序出现，并且不需要与在表中的列相一致。

复合索引可以加速 **SELECT** 语句的数据取回速度，在 **SELECT** 语句中 **WHERE** 子句引用复合索引的所有列或部分主要的列。

## 什么时候不创建索引

下面的情况通常不值得创建索引，如果：

- 表很小
- 不经常在查询中作为条件被使用的列
- 大多数查询期望取回多于表中百分之 2 到 4 的行
- 表经常被更新
- 被索引的列作为表达式的的一部分被引用

### 教师注释

空值不能包含在索引中。

为了优化连接，你可以在 FOREIGN KEY 列上创建一个索引，它将加速搜索匹配的行到 PRIMARY KEY 列。

如果 WHERE 子句包含 IS NULL 表达式，优化不使用索引。

## 确认索引

- **USER\_INDEXES** 数据字典视图包含索引和它唯一的名字
- **USER\_IND\_COLUMNS** 视图包含索引名、表名和列名

```
SELECT  ic.index_name, ic.column_name,
        ic.column_position col_pos, ix.uniqueness
FROM    user_indexes ix, user_ind_columns ic
WHERE   ic.index_name = ix.index_name
AND     ic.table_name = 'EMPLOYEES';
```

### 确认索引

从 **USER\_INDEXES** 数据字典视图可以确认索引的存在。你也可以查询 **USER\_IND\_COLUMNS** 视图，检查与索引有关的列。

幻灯片的例子显示在 **EMPLOYEES** 表上所有已创建的索引，包括受影响的列的名字和索引的唯一性。

INDEX_NAME	COLUMN_NAME	COL_POS	UNIQUENES
EMP_EMAIL_UK	EMAIL	1	UNIQUE
EMP_EMP_ID_PK	EMPLOYEE_ID	1	UNIQUE
EMP_DEPARTMENT_IX	DEPARTMENT_ID	1	NONUNIQUE
EMP_JOB_IX	JOB_ID	1	NONUNIQUE
EMP_MANAGER_IX	MANAGER_ID	1	NONUNIQUE
EMP_NAME_IX	LAST_NAME	1	NONUNIQUE
EMP_NAME_IX	FIRST_NAME	2	NONUNIQUE
EMP_LAST_NAME_IDX	LAST_NAME	1	NONUNIQUE

8 rows selected.

## 基于函数的索引

- 一个基于函数的索引就是一个基于表达式的索引
- 索引表达式用表中的列、常数 SQL 函数和自定义函数来构建

```
CREATE INDEX upper_dept_name_idx
ON departments(UPPER(department_name));

Index created.

SELECT *
FROM   departments
WHERE  UPPER(department_name) = 'SALES';
```

### 基于函数的索引

基于函数的索引用 **UPPER** (列名)或 **LOWER** (列名)关键字定义，它允许大小写敏感的查询，例如，下面的索引：

```
CREATE INDEX upper_last_name_idx ON employees (UPPER(last_name));
```

使得处理查询容易，例如：

```
SELECT * FROM employees WHERE UPPER(last_name) = 'KING';
```

确保 Oracle 服务器使用索引而不是执行一个全表扫描，注意，在子查询中的函数值不能为空，例如，下面的语句保证使用索引，但如果没有 **WHERE** 子句，Oracle 服务器还可能执行一个全表扫描：

```
SELECT *
FROM   employees
WHERE  UPPER (last_name) IS NOT NULL
ORDER BY UPPER (last_name);
```

## 基于函数的索引

- 一个基于函数的索引就是一个基于表达式的索引
- 索引表达式用表中的列、常数 SQL 函数和自定义函数来构建

```
CREATE INDEX upper_dept_name_idx
ON departments(UPPER(department_name));

Index created.

SELECT *
FROM   departments
WHERE  UPPER(department_name) = 'SALES';
```

### 基于函数的索引 (续)

Oracle 服务器将带有 DESC 标记的列作为基于函数的索引处理，带 DESC 标记的列被以降序排序。

### 教师注释

让学生知道在用户方案中的表上创建一个基于函数的索引，你必须有 CREATE INDEX 和 QUERY REWRITE 系统权限。为了在另一个方案或另一个方案的表上创建索引，你必须有 CREATE ANY INDEX 和 GLOBAL QUERY REWRITE 权限。在基于函数的索引中，表的所有者也必须有关于函数的使用 EXECUTE 对象的权限。

## 删除索引

- 用 **DROP INDEX** 命令从数据字典中删除索引

```
DROP INDEX index;
```

- 从数据字典中删除 **UPPER\_LAST\_NAME\_IDX** 索引

```
DROP INDEX upper_last_name_idx;  
Index dropped.
```

- 为了删除索引，你必须是索引的所有者，或者有 **DROP ANY INDEX** 权限

### 删除索引

你不能修改索引，为了改变索引，你必须先删除它，然后重新创建它。用 **DROP INDEX** 语句从数据字典中删除索引，为了删除索引，你必须是索引的所有者，或者有 **DROP ANY INDEX** 权限。

在语法中：

*index*          是索引的名字

**注：**如果你删除一个表，相关的索引和约束将被自动删除，但视图和序列将保留。

## 同义词

创建同义词可以简化对象访问 (对象的另一个名字)。同义词能够:

- 另一个用户易于查阅表的所有者
- 使对象名字变短

```
CREATE [PUBLIC] SYNONYM synonym
FOR    object;
```

中国科学院西安网络中心 编译 2005

ORACLE

12-24

Copyright © Oracle Corporation, 2001. All rights reserved.

### 为一个对象创建同义词

为了查阅另一个用户所拥有的表,你需要将创建该表的用户名加句点作为前缀加在表名前面。创建一个同义词可以除去对象名必须带的方案限制,并提供给你一个可替换表名、视图名、序列名和过程名或其它对象名。该方法对具有特别长的对象的名字很有用。

在语法中:

<code>PUBLIC</code>	创建一个可以被所有用户访问的同义词
<code><i>synonym</i></code>	是要被创建的同义词的名字
<code><i>object</i></code>	指出要创建同义词的对象

### 原则

对象不能包含包。

一个私有同义词名字对于同一个用必须与所有其它的对象不同。

更多信息, 见 *Oracle9i SQL Reference*, “创建同义词”。



## 创建和删除同义词

- 为 DEPT\_SUM\_VU 视图创建一个短名字

```
CREATE SYNONYM d_sum  
FOR dept_sum_vu;  
Synonym Created.
```

- 删除同义词

```
DROP SYNONYM d_sum;  
Synonym dropped.
```

### 为一个对象创建同义词 (续)

幻灯片中的例子创建了一个 DEPT\_SUM\_VU 视图的用于快速引用的同义词。数据库管理员可以创建一个可为所有用户访问的公共同义词。下面的例子为 Alice 的 DEPARTMENTS 表创建一个公共同义词：

```
CREATE PUBLIC SYNONYM dept  
FOR alice.departments;  
Synonym created.
```

### 删除同义词

用 DROP SYNONYM 语句删除一个同义词，只有数据库管理员可以删除一个公共同义词。

```
DROP PUBLIC SYNONYM dept;  
Synonym dropped.
```

更多信息，见 *Oracle9i SQL Reference*，“删除同义词”

### 教师注释

在 Oracle 服务器中，DBA 可以明确授予 CREATE PUBLIC SYNONYM 权限给任何用户，这样普通用户就可以创建公共同义词。

## 小结

在本课中，您应该已经学会如何：

- 用序列发生器自动产生序列数
- 在 **USER\_SEQUENCES** 数据字典表中查看序列信息
- 创建索引改进查询取回速度
- 在 **USER\_INDEXES** 字典表中查看索引信息
- 使用同义词为对象提供另一个名字

中国科学院西安网络中心 编译 2005

ORACLE

12-26

Copyright © Oracle Corporation, 2001. All rights reserved.

## 小结

在本课中你应该已经学习了关于另外的一些数据库对象，包括序列、索引和同义词。

### 序列

序列发生器可以用来为表中的行自动产生序列数，这样可以节省时间并且减少应用程序的代码量。

序列是可以与其它用户共享的数据库对象，关于序列的信息可以在数据字典的 **USER\_SEQUENCES** 表中找到。

为了使用一个序列，用 **NEXTVAL** 或 **CURRVAL** 伪列引用它。

- 引用 `sequence.NEXTVAL` 可以取回序列的下一个数。
- 引用 `sequence.CURRVAL` 可以返回当前可用序列数。

### 索引

索引被用于改进查询的速度，用户可以在 **USER\_INDEXES** 数据字典视图中查看索引的定义。索引可以被创建者或者其它有 **DROP ANY INDEX** 权限的用户用 **DROP INDEX** 语句删除。

### 同义词

为了方便，使用 **CREATE SYNONYM** 语句，数据库管理员可以创建公共同义词，普通用户也可以创建私有同义词。同义词允许短名字或者用于对象的替换名。用 **DROP SYNONYM** 语句删除同义词。

## 练习 12 概览

本章练习包括下面的主题：

- 创建序列
- 使用序列
- 创建非唯一索引
- 显示关于序列和索引的数据字典信息
- 删除索引

### 练习 12 概览

在本章的练习中，你将创建一个序列并将它组装到你的表中使用，你还将创建隐式和显式索引。

幻灯片 28

## 练习 12

创建一个序列用于 DEPT 表的主键列，该序列从 200 开始，并且有最大值 1000，序列的增量是 10，序列的名字是 DEPT\_ID\_SEQ。

```
CREATE SEQUENCE dept_id_seq
START WITH 200
INCREMENT BY 10
MAXVALUE 1000;
```

2. 在脚本中写一个查询，显示下面关于序列的信息：序列名、最大值、增量大小和最后的值，命名脚本为 script lab12\_2.sql。在脚本中运行语句。

SEQUENCE_NAME	MAX_VALUE	INCREMENT_BY	LAST_NUMBER
DEPARTMENTS_SEQ	9990	10	280
DEPT_ID_SEQ	1000	10	200
EMPLOYEES_SEQ	1.0000E+27	1	207
LOCATIONS_SEQ	9900	100	3300

```
SELECT sequence_name, max_value, increment_by, last_number
FROM user_sequences;
```

3. 写一个脚本插入两行到 DEPT 表中，命名该脚本为 lab12\_3.sql，确信将前面创建的序列用于 ID 列，添加两个部门名字 Education 和 Administration。确认你的添加。运行脚本中的命令。

```
INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Education');

INSERT INTO dept
VALUES (dept_id_seq.nextval, 'Administration');
```

4. 在 EMP 表中的外键列 DEPT\_ID 上创建一个非唯一性索引。

```
CREATE INDEX emp_dept_id_idx ON emp (dept_id);
```

5. 显示存在于数据字典中对于 EMP 表的索引和唯一性。保存语句到一个命名为 lab12\_5.sql 的脚本中。

INDEX_NAME	TABLE_NAME	UNIQUENES
EMP_DEPT_ID_IDX	EMP	NONUNIQUE
EMP_ID_PK	EMP	UNIQUE

```
SELECT index_name, table_name, uniqueness
FROM user_indexes
WHERE table_name = 'EMP';
```