**第07章 序列、索引和访问控制**

**本章内容**

* 序列
* 索引
* 访问控制

# 序列

## 概述

* 序列是数据库对象，独立于表存储，可以为多个表使用
* 序列最主要的用途就是创建一个主键的值,序列能确保这个主键的唯一性

## 语法

CREATE SEQUENCE sequence --序列的名字;

[INCREMENT BY n] --序列号之间的间隔n是整数，默认为1，如果n为负值， --该序列为降序序列;

[START WITH n] --第一个序列数 ，n默认为1;

[{MAXVALUE n| NOMAXVALUE}]--序列最大值 | 无最大值，默认对于升序序列，最大 --值为10^27，降序序列为-1;

[{MINVALUE n| NOMINVALUE}]--序列最小值 | 无最小值，默认对于升序序列，最小 --值为1，降序序列为-10^26;

[{CYCLE | NOCYCLE}] --序列是否循环使用

[{CACHE n| NOCACHE}]; --是否使用缓存，使用可以提高性能

--简化：

create sequence seq\_id;

start with 1

increment by 1

## 创建序列

--需求：采用数据库序列的方式创建一个序列，命名为：seq\_dept\_deptno，用于给部门表设置主键

CREATE SEQUENCE seq\_dept\_deptno

INCREMENT BY 1

START WITH 1

MAXVALUE 99

NOCYCLE

NOCACHE;

## 查看序列

SELECT sequence\_name, last\_number, min\_value, max\_value, increment\_by

FROM user\_sequences;

## 使用序列

select seq\_dept\_deptno.nextval from dual ; --下一个序列值

select seq\_dept\_deptno.currval from dual ; --当前序列值

## 修改序列

修改序列的使用ALTER，且START WITH选项不可以修改，具体语法形式如下：

ALTER SEQUENCE sequence

[INCREMENT BY n]

[{MAXVALUE n| NOMAXVALUE}]

[{MINVALUE n| NOMINVALUE}]

[{CYCLE | NOCYCLE}]

[{CACHE n| NOCACHE}];

特别强调注意：

1.修改序列，必须是被修改序列的所有者或者用户具有ALTER权限

2.使用ALTER SEQUENCE修改序列，只有以后的序列值会受到影响，之前的不受影响

3.修改序列的部分选项不能使原有序列产生错误，例如不能将序列的最大值修改为小于当前的序列值

--示例

ALTER SEQUENCE seq\_dept\_deptno

INCREMENT BY 2;

## 删除序列

DROP SEQUENCE seq\_dept\_deptno;

# 索引

## 概述

索引是用于加速数据存取的数据对象。合理的使用索引可以大大降低i/o 次数，从而提高数据访问性能.

## 索引分类

* B树索引（B\*Tree Index） （默认索引）
* 位图索引（Bitmap Index）
* 反向索引（Reverse Key Index）
* 降序索引（Descending Index）
* 函数索引（Function-Based Index）

## 创建索引

语法：

CREATE [UNIQUE] INDEX index ON table(col1, col2…);

---UNIQUE表示建立的是唯一索引还是非唯一索引

## 删除索引

DROP INDEX IDX\_EMP\_JOB;

## 创建索引

### 创建单列非唯一键索引

---需求1：为了改善雇员表emp中职位JOB的查询速度，可以给雇员表的职位字段建立一个--索引，其SQL语句如下

CREATE INDEX IDX\_EMP\_JOB ON emp(job);

### 创建组合索引

---需求2：为了改善雇员表emp中职位JOB和上级经理MGR的查询速度，可以给雇员表的职--位字段和上级经理字段建立一个索引，其SQL语句如下

CREATE INDEX IDX\_EMP\_JOB\_MGR ON emp(job,mgr);

### 创建函数索引

---需求3：一条复杂SQL的WHERE条件中需要查询雇员表emp中名字ENAME的小写名

CREATE INDEX IDX\_EMP\_ENAME ON emp(LOWER(ename));

/\*老师可以对比有索引和没索引的情况，执行计划有什么不同\*/

## 查询所有索引

select \* from user\_indexes ;

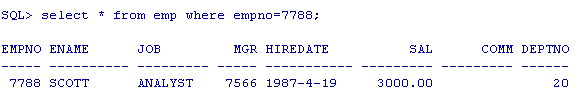
## 索引扫描方式

* 唯一扫描（UNIQUE SCAN）
* 范围扫描（RANGE SCAN）
* 全扫描（FULL SCAN）
* 快速扫描（FAST FULL SCAN）
* 跳跃式扫描(SKIP SCAN)

### 唯一扫描（UNIQUE SCAN）

**1、概述：**当谓词中包含使用unique或primary key索引的列作为条件的时候就会选用索引唯一扫描。这种类型的索引能够保证对于某个特定的值**只返回一行数据**

**2、示例：使用索引唯一扫描的例子**

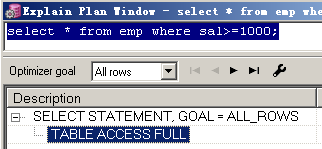


**3、使用UNIQUE SCAN 的3种情况**

* 必须是通过唯一索引【UNIQUE】来访问数据
* 通过唯一索引来访问数据时，每次返回的记录数必须是1条
* WHERE从句中必须要用等值【=】条件来过滤数据:

### 范围扫描（RANGE SCAN）

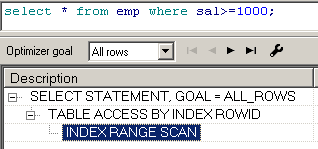
**概述：**当谓词中包含将会返回一定范围数据的条件时就会选用索引范围扫描。范围越大，就越有可能会选用全扫描运算来代替它。使用一个索引存取多行数据

**1、不加索引前**  


**结果显示：全扫描**

**2、添加索引后**

create index idx\_sal on emp(sal);



**结果显示：范围扫描**

使用index rang scan的3种情况：  
1、在唯一索引列上使用了range操作符(> < <> >= <= between)。   
2、在组合索引上(unique index)，只使用部分列进行查询，导致查询出多行。   
3、对非唯一索引列上进行的任何查询。

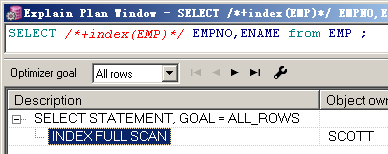
### 全扫描（FULL SCAN）

**1、概述**：SQL请求的全部列必须包含在索引中；也就是说，SELECT和WHERE字句中的所有数据列必须存在于索引中，索引全扫描是根据叶节点链来进行的。进行索引全扫描首先要从根开始，找到叶节点链上的第一个数据块，然后沿着叶节点链进行扫描，由于叶节点链是根据索引键值排序的，因此这样扫描出来的数据本身就是排序的，数据读出后不需要再次排序。这种扫描方式，首先要找到索引的根，然后通过枝节点找到第一个叶节点，然后再顺着叶节点链扫描整个索引

**2、示例：全索引扫描的例子**

CREATE INDEX EMP\_EMPNO\_ENAME\_IDX ON EMP(EMPNO,ENAME) ;

SELECT */\*+index(EMP\_EMPNO\_ENAME\_IDX)\*/* EMPNO,ENAME from EMP ;



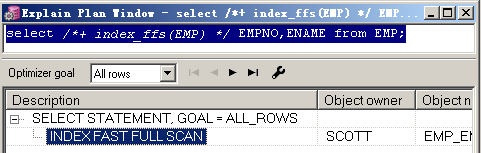
### 快速扫描（FAST FULL SCAN）

1、概述：查询列表中所有字段都包含在索引中并且索引中至少有一列具有非空约束时替代全表扫描的。扫描索引中的所有的数据块，数据是通过无序的多块读取来的，它不对查询出的数据进行排序，即数据不是以排序顺序被返回，因为。这一点带来的好处最终抵消任何排序导致的成本，索引快速全扫描运算依赖于非空约束的，如果没有这样的约束，将会选择全表扫描

**2、示例：索引快速扫描的例子**

*--其中ename 为not null 字段*

CREATE INDEX EMP\_EMPNO\_ENAME\_IDX ON EMP(EMPNO,ENAME) ;



## 索引优缺点

**优点：**

* 通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性
* 可以大大加快数据的检索速度
* 可以加速表和表之间的连接，特别是在实现数据的引用完整性方面特别有意义

**缺点：**

* 创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加
* 索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间
* 当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度

# 访问控制

## Oracle 用户的管理

### 创建用户

语法:create user 用户名 identified by 密码

### 修改别人的密码

语法:alter user 用户名 identified by 新密码

注意:(需要dba或alter user权限)

### 删除用户:

语法:drop user 用户名 [cascade]

注意:1.如果被删除用户已经创建了表,需要带cascade,否则不需要

2.(需要拥有dba或alter user权限)

### 授予表权限命令()

语法:grant 权限名 on 表名 to 用户名

注意:权限:select(查) update(增删改) all(所有)

### 授予系统权限

语法:grant 权限名 to 用户名

角色权限: 允许用户登录：CREATE SESSION

允许创建表：CREATE TABLE

允许创建视图：CREATE VIEW

允许创建序列：CREATE SEQUENCE

允许创建过程：CREATE PROCEDURE

### 角色权限:

创建角色: CREATE ROLE developer;

给角色赋予权限: GRANT create table, create view, create sequence, create procedure TO developer;

把角色赋予用户: GRANT developer TO jacky;

### 收回表权限命令

语法:revoke 权限名 on 表名 from 用户名

### 授予用户"授予权限"

语法:grant 权限名 on 表名 to 用户名 with grant option

注意:1.要授予用户"授予权限",自己必须具备授予权限

2.(\*株连,通杀)如果A授予B查询权限,此时回收A的查询权限,E的查询权限也 被回收

## 锁定和解锁

### 帐户锁定,密码策略(三天内必须登录,或只能尝试登录三次)

创建策略语法:create profile 策略名 limit 系统策略 单位

如:create profile MyProfile limit failed\_login\_attempts 3 password\_lock\_time 3

注意:系统策略名如下:

1.failed\_login\_attempts:登录次数

2.password\_lock\_time:登录时间 单位天

3.password\_life\_time:每隔多长时间修改一次密码 单位天

4.password\_grace\_time:宽限修改密码 时间

5.password\_reuse\_time:密码多长时间内不可重复

分配策略语法:alter user 用户名 profile 策略名

删除策略语法:drop profile 策略名 [cascade]

### 给帐号解锁 scott - system - sys

语法:alter user 用户名 account unlock;

## 创建表空间

/\*分为四步 \*/

/\*第1步：创建临时表空间 \*/

create temporary tablespace user\_temp

tempfile 'D:\oracle\oradata\Oracle9i\user\_temp.dbf'

size 50m

autoextend on

next 50m maxsize 20480m

extent management local;

/\*第2步：创建数据表空间 \*/

create tablespace user\_data

logging

datafile 'D:\oracle\oradata\Oracle9i\user\_data.dbf'

size 50m

autoextend on

next 50m maxsize 20480m

extent management local;

/\*第3步：创建用户并指定表空间 \*/

create user username identified by password

default tablespace user\_data

temporary tablespace user\_temp;

/\*第4步：给用户授予权限 \*/

grant connect,resource,dba to username;