广州大学学生实验报告

开课实验室： 电子楼 418　　2019 年 5 月 23 日

学院	计算机科学与网络工程学院	年级、专业、班	软件 171	姓名	谢金宏	学号	1706300001
实验课程名称	数据库原理					成绩	
实验项目名称	实验五 Oracle 数据库对象					指导老师	张少宏

教师评语：

一、实验目的

1. Oracle 数据库包含许多数据库对象，例如表、视图、索引、序列、存储过程、触发器等。表、视图、索引的操作在前面的实验中已经做了相应的练习，本实验将介绍如何使用序列、触发器和存储过程。同学们可以通过本实验掌握如何定义、使用删除这些数据库对象。

二、实验环境

安装有 Oracle 11g 数据库软件的远程计算机和安装有 SQL Developer 软件的本地计算机。

三、实验内容

1. 序列
   a) 创建序列
   b) 查看创建的序列对象
   c) 使用序列
   d) 修改序列
   e) 删除序列
2. 存储过程
   a) 创建三个数据表
   b) 插入数据创建存储过程
   c) 创建存储过程，更新表中的数据
   d) 执行存储过程，并比较存储过程执行前后的数据变化情况
   e) 删除存储过程
   f) 创建存储过程
   g) 运行存储过程
3. 触发器
   a) 创建触发器
   b) 创建触发器 credit_id
   c) 查看刚创建的触发器对象
   d) 激活刚创建的触发器

四、实验步骤

本次实验中使用的 SQL 语句：

代码.sql

1. 以 SYSTEM 连接数据库 ORCL，执行以下语句查看对象：

```
SELECT COUNT(object_name) FROM all_objects WHERE OWNER = 'SYSTEM';
```

查询出共有 502 个数据库对象。

2. 创建新用户 Light 并授予其 Resource, Connect 和 DBA 权限。
3. 以新用户的身份连接，并执行以下操作：

```
SELECT COUNT(*) FROM all_objects WHERE OWNER = 'SYSTEM';
-- 显示 502 行。

SELECT object_name, OWNER FROM all_objects WHERE OWNER = 'light';
-- 显示 6 行，均为此用户在过往的实验中创建的表和约束。

CREATE SEQUENCE my_seq_01 INCREMENT BY 1 START WITH 1 NOMAXVALUE NOCYCLE;
CREATE SEQUENCE my_seq_02 INCREMENT BY 2 START WITH 1;

SELECT object_name, OWNER FROM all_objects WHERE OWNER = 'light';
-- 显示 8 行，新增 2 个序列对象。

SELECT object_name,object_type, OWNER FROM all_objects WHERE OWNER = 'SYSTEM' AND
object_type='SEQUENCE';
-- 共有 20 行。

SELECT object_name, object_type, OWNER FROM all_objects WHERE OWNER = 'light' AND
object_type='SEQUENCE';
-- 显示 2 行。
```

```
SELECT my_seq_01.NEXTVAL FROM dual;
-- 重复执行上述语句得到序列：1, 2, 3, 4, ...

ALTER SEQUENCE my_seq_01 INCREMENT BY 10;

SELECT my_seq_01.NEXTVAL FROM dual;
-- 重复执行上述语句得到序列：14（从上一个值继续）, 24, 34, 44, ...

SELECT my_seq_02.NEXTVAL FROM dual;
-- 重复执行上述语句得到序列：1, 3, 5, 7, ...

DROP SEQUENCE my_seq_02;

SELECT my_seq_02.NEXTVAL FROM dual;
-- ORA-02289: 序列不存在

CREATE SEQUENCE my_seq_02 INCREMENT BY 3 START WITH 100;

SELECT my_seq_02.NEXTVAL FROM dual;
-- 重复执行上述语句得到序列：100, 103,106, 109, ...
```

4. 执行储存过程操作：

```
-- 删除旧表和旧过程。
DECLARE
tmp INTEGER DEFAULT 0;
BEGIN
SELECT COUNT(*) INTO tmp FROM user_tables WHERE table_name='SC';
IF(tmp>0) THEN
EXECUTE IMMEDIATE 'drop table SC ';
END IF;
SELECT COUNT(*) INTO tmp FROM user_tables WHERE table_name='STUDENT';
IF(tmp>0) THEN
EXECUTE IMMEDIATE 'drop table STUDENT ';
END IF;
SELECT COUNT(*) INTO tmp FROM user_tables WHERE table_name='COURSE';
IF(tmp>0) THEN
EXECUTE IMMEDIATE 'drop table COURSE ';
END IF;
SELECT COUNT(*) INTO tmp FROM all_objects WHERE object_name='SC_INS' AND
object_type='PROCEDURE';
IF(tmp>0) THEN
EXECUTE IMMEDIATE 'drop PROCEDURE SC_INS ';
END IF;
SELECT COUNT(*) INTO tmp FROM all_objects WHERE object_name='STUDENT_NO' AND
object_type='SEQUENCE';
IF(tmp>0) THEN
EXECUTE IMMEDIATE 'drop SEQUENCE STUDENT_NO ';
END IF;
END;

-- 重新创建数据表。
CREATE TABLE student(sno INT PRIMARY KEY, sname VARCHAR2(8 CHAR));
CREATE TABLE course(cno INT PRIMARY KEY, cname VARCHAR2(10 CHAR));
CREATE TABLE sc(sno INT, cno INT, grade INT,
      PRIMARY KEY(sno,cno),
      FOREIGN KEY (sno) REFERENCES student(sno), FOREIGN KEY (cno) REFERENCES course(cno)
);


SELECT object_name, object_type, OWNER FROM all_objects WHERE OWNER = 'light';
-- 显示 8 行。
```

索引因创建表的主键约束而产生。例如 SC 表的索引由 SNO, CNO 产生。

```
CREATE SEQUENCE student_no INCREMENT BY 1 START WITH 2012001;

INSERT INTO student VALUES(student_no.NEXTVAL, 'aaaaaa');
INSERT INTO student VALUES(student_no.NEXTVAL, 'bbbbbbb');
INSERT INTO student VALUES(student_no.NEXTVAL, 'ccccccc');
INSERT INTO student VALUES(student_no.NEXTVAL, 'ddddddd');
COMMIT;
SELECT * FROM student;

INSERT INTO course VALUES (105, '程序设计');
INSERT INTO course VALUES (908, '大学英语');
INSERT INTO course VALUES (433, '数据结构');
COMMIT;
SELECT * FROM course;

CREATE PROCEDURE sc_ins(ino INT,cno INT,grade INT) IS
BEGIN
  IF(grade>=0) THEN INSERT INTO sc VALUES (ino,cno,grade);
  ELSE INSERT INTO sc VALUES (ino,cno,NULL);
END IF;
END;
-- Procedure SC_INS 已编译。

SELECT object_name, object_type, OWNER FROM all_objects WHERE OWNER = 'light' AND
object_type='PROCEDURE';
-- 显示 1 行：SC_INS。

EXEC sc_ins (2012001,105,60);
EXEC sc_ins (2012001,908,0);
EXEC sc_ins (2012001,433,98);
EXEC sc_ins (2012002, 105,75);
EXEC sc_ins (2012002, 433,-1);
EXEC sc_ins (2012003, 105,64);
EXEC sc_ins (2012003, 908,90);
EXEC sc_ins (2012003, 433,-100);
-- PL/SQL 过程已成功完成。

SELECT student.sno,sname,cname,grade
  FROM student,course,sc
  WHERE student.sno=sc.sno AND course.cno=sc.cno;
```

如果成绩为负数的话，在数据表中为 NULL 值。

成绩为 0 在表中的储存结果为 0，成绩为负数在表中的储存结果为 NULL。
使用储存过程有简化操作、降低失误的可能性的好处。

5. 执行触发器的操作。

```
ALTER TABLE sc ADD (gradelevel CHAR(1 CHAR));

UPDATE sc SET gradelevel='A' WHERE grade>=85;
UPDATE sc SET gradelevel='B' WHERE grade>=75 AND grade<85;
UPDATE sc SET gradelevel='C' WHERE grade>=60 AND grade<75;
UPDATE sc SET gradelevel='D' WHERE grade<60;

SELECT student.sno, sname, cname, grade, gradelevel
  FROM student,course,sc
  WHERE student.sno=sc.sno AND course.cno=sc.cno;
-- 新增了四级评价的 GradeLevel 列。

CREATE OR REPLACE TRIGGER sc_ins BEFORE INSERT OR UPDATE ON sc
FOR EACH ROW
BEGIN
  IF :NEW.grade>=85 THEN :NEW.gradelevel:='A';
  ELSE IF :NEW.grade>=75 THEN :NEW.gradelevel:='B';
      ELSE IF :NEW.grade>=60 THEN :NEW.gradelevel:='C';
          ELSE IF :NEW.grade>=60 THEN :NEW.gradelevel:='D';
 END IF;
  END IF;
   END IF;
    END IF;
 END;
-- Trigger SC_INS 已编译。

SELECT * FROM sc WHERE sno=2012002;
INSERT INTO sc(sno, cno, grade) VALUES (2012002, 908, 80);
SELECT * FROM sc WHERE sno=2012002;
-- 对于新插入的行，GradeLevel 列由触发器计算和生成。

ALTER TABLE course ADD (maxgrade INT);

UPDATE course SET maxgrade=0;
SELECT * FROM course;
-- 成功修改了 COURSE 表的结构，并将 MaxGrade 设置为 0。
```

```
CREATE OR REPLACE TRIGGER course_ins BEFORE INSERT OR UPDATE ON sc
FOR EACH ROW
DECLARE oldg INT;
BEGIN
 SELECT maxgrade INTO oldg FROM course WHERE cno=:NEW.cno;
 IF oldg<:NEW.grade THEN UPDATE course SET maxgrade=:NEW.grade WHERE
cno=:NEW.cno;
 END IF;
 END course_ins;
-- Trigger COURSE_INS 已编译。
-- 目前共有 2 个用户创建的触发器。

SELECT * FROM course;
-- 当前各科的最高分数均为 0。

INSERT INTO sc(sno,cno,grade) VALUES (2012004,908,99);
INSERT INTO sc(sno,cno,grade) VALUES (2012004,433,88);
INSERT INTO sc(sno,cno,grade) VALUES (2012004,105,59);
SELECT * FROM sc;
SELECT * FROM course;
-- 插入成功。各科最高分也得到更新。

SELECT * FROM sc  WHERE sno=2012003 AND cno=105;
-- 2012003 号同学的 105 号课程的分数为 64，级别为 C。

UPDATE sc SET grade=100 WHERE sno=2012003 AND cno=105;
SELECT * FROM sc WHERE sno=2012003 AND cno=105;
SELECT * FROM course;
-- 2012003 号同学的 105 号课程的分数修改成功。
-- GradeLevel 变化为 A，各科的最高分(course.maxgrade)得到更新。
-- 单个修改语句，可以同时触发多个触发器。
```

五、分析总结

完成了全部实验内容，基本掌握了创建简单序列、储存过程和触发器的方法。

对数值型主键的自增，MariaDB 等数据库软件还有 AUTO INCREAMENT 属性的实现思路。