数据库系统课程实验报告

实验名称:	触发器
实验日期:	2022.12.9
实验地点:	四号楼
提交日期:	2022.12.10

学号:	22920202202877
姓名:	陈鑫蕾
专业年级:	数媒 2020 级
学年学期:	2022-2023 学年第一学期

1. 实验目的

理解 openGauss 触发器的作用和工作原理

■ AFTER/BEFORE 触发器

■ 行级(row)触发器和语句级(statement)触发器

熟练掌握 openGauss 触发器的设计方法

熟练掌握 openGauss 触发器的定义、查看、禁止、启用和删除操作

2. 实验内容和步骤

（1）连接到数据库

步骤如下:

1.在数据库主节点服务器上，切换至 omm 操作系统用户环境。

```
-bash: gsql: command not found
[root@ecs-7cda ~]# su - omm
Last login: Tue Oct 18 10:19:04 CST 2022 on pts/0
```

2.查看服务是否启动。

```
[omm@ecs-7cda ~]$  gs_om -t status
-----------------------------------------------------------------
--------

cluster_name     : dbCluster
cluster_state    : Normal
redistributing   : No

-----------------------------------------------------------------
--------
```

3.启动数据库服务

```
[omm@ecs-7cda ~]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] ecs-7cda:
[2022-10-18 12:26:33.753][14929][][gs_ctl]: gs_ctl started,data
dir is /gaussdb/data/db1
[2022-10-18 12:26:33.756][14929][][gs_ctl]:  another server mig
ht be running; Please use the restart command
=========================================
Successfully started.
```

4.连接数据库

```
[omm@ecs-7cda ~]$ gsql -d postgres -p 26000 -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 2
1:03:52 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requirin
g high-security)
Type "help" for help.
```

（2）触发器

1.创建部门表 dept(deptno, deptname)

```
postgres=# CREATE TABLE dept(
postgres(#     deptno char(2) PRIMARY KEY,
postgres(#     deptname varchar(20) NOT NULL
postgres(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"dept_pkey" for table "dept"
CREATE TABLE
```

2.创建 Teacher 表：Teacher(ID, job, Sal, deptno)

```
CREATE TABLE
postgres=# CREATE TABLE TEACHER(
postgres(#     id char(5) PRIMARY KEY,
postgres(#     job varchar(20) NOT NULL,
postgres(#     sal numeric(7,2),
postgres(#     deptno char(2),
postgres(#     FOREIGN KEY (deptno) REFERENCES dept(deptno)
postgres(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
"teacher_pkey" for table "teacher"
CREATE TABLE
```

3.为 dept 表增加实验数据：（‘01’,’CS’),(‘02’,’SW’),(‘03’,’MA’)；为 Teacher 表增加实验数据：(‘10001’, ‘教授’,3800,’01’),

('10002','教授',4100,'02'),('10003','副教授',3500,'01'),
('10004','助理教授',3000,'03')

```
postgres=# INSERT INTO dept
postgres-# VALUES('01','CS'),('02','SW'),('03','MA');
INSERT 0 3
```

```
postgres=# INSERT INTO TEACHER
postgres-# VALUES('10001','教授',3800,'01'), ('10002','教授',41
00,'02'), ('10003','副教授',3500,'01'), ('10004','助理教授',300
0,'03');
INSERT 0 4
```

4.在 Teacher 表上创建一个 BEFORE 行级触发器（名称：INSERT_OR_UPDATE_SAL）以实现如下完整性规则：**教授的工资不得低于 4000 元，如果低于 4000 元，自动改为 4000 元**。

```
postgres=# CREATE OR REPLACE FUNCTION UPDATE_SAL()
postgres-# RETURNS TRIGGER AS $$
postgres$# DECLARE
postgres$# BEGIN
postgres$# IF(NEW.SAL<4000)AND(NEW.JOB = '教授')
postgres$#
postgres$# THEN NEW.SAL := 4000;
postgres$# END IF;
postgres$# RETURN NEW;
postgres$# END
postgres$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
```

```
postgres=# CREATE TRIGGER INSERT_OR_UPDATE_SAL
postgres-# BEFORE INSERT OR UPDATE ON Teacher
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE UPDATE_SAL();
CREATE TRIGGER
```

5.验证触发器是否正常工作：分别执行以下 A，B 两种操作，验证 INSERT_OR_UPDATE_SAL 触发器是否被触发？工作是否正确？如果正确，请观察 Teacher 表中数据的变化是否与预期一致。

A. 插入两条新数据(‘10005’,‘教授’,3999,’02’),(‘10006’,‘教授’,4000,’03’);B. 更新数据:将 id 为 10002 的教授工资改为 3900。

①插入数据

```
postgres=# INSERT INTO TEACHER
postgres-# VALUES('10005','教授',3999,'02'), ('10006','教授',40
00,'03');
INSERT 0 2
```

②更新数据

```
postgres=# UPDATE TEACHER
postgres-# SET sal=3900
postgres-# WHERE id='10002';
UPDATE 1
```

③检验

查看表，发现教授的工资都大等于 4000，触发器正常工作。

```
INSERT 0 1
postgres=# SELECT *
postgres-# FROM TEACHER;
  id   |   job    |   sal   | deptno
-------+----------+---------+--------
 10001 | 教授     | 3800.00 | 01
 10003 | 副教授   | 3500.00 | 01
 10004 | 助理教授 | 3000.00 | 03
 10005 | 教授     | 4000.00 | 02
 10006 | 教授     | 4000.00 | 03
 10002 | 教授     | 4100.00 | 02
(6 rows)
```

6.查看触发器（名称和代码）；

```
  tgrelid |            tgname             | tgfoid | tgtype | tg
gargs | tgqual | tgowner
---------+-------------------------------+--------+--------+---
------+--------+---------
    16942 | insert_or_update_sal          |  16954 |     23 | O
x     |        |      10
```

```
   17040 | RI_ConstraintTrigger_c_17051 |   1644 |     5 | O       | t          |
x     |        |      10
   17040 | RI_ConstraintTrigger_c_17052 |   1645 |    17 | O       | t          |
x     |        |      10
   17025 | sc_after_update              |  17053 |    17 | O       | f          |
x     |        |      10
(22 rows)
```

7.设计触发器自动维持表间的外码约束：删除 dept 表中 deptno 为 03 的数据后，teacher 表上引用该数据的记录也被自动删除。

①创建触发器

```
postgres=# CREATE OR REPLACE FUNCTION DELETE_AFTER()
postgres-# RETURNS TRIGGER AS $$
postgres$# DECLARE
postgres$# BEGIN
postgres$#     IF(old.deptno='03')
postgres$#     THEN DELETE FROM TEACHER
postgres$#     WHERE TEACHER.deptno='03';
postgres$#     END IF;
postgres$#     RETURN NEW;
postgres$# END
postgres$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# CREATE TRIGGER DELETE_AFTER_tr
postgres-# BEFORE DELETE ON dept
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE DELETE_AFTER();
CREATE TRIGGER
```

②检验触发器

删除 dept 表中 deptno=‘03’的记录，再次查询 teacher 表，可以看到已经不存在 depteno=‘03’的记录了。

```
postgres=# delete from dept where deptno='03';
DELETE 0
postgres=# select * from teacher;
  id   |  job   |   sal   | deptno
-------+--------+---------+--------
 10001 | 教授   | 3800.00 | 01
 10003 | 副教授 | 3500.00 | 01
 10005 | 教授   | 4000.00 | 02
 10002 | 教授   | 4100.00 | 02
(4 rows)
```

8.设计触发器实现审计日志记录(教材例 5.21)：当对表 SC 的 Grade 属性进行修改时，若分数增加了 10%及其以上，则将此次操作记录到下面表中：SC_U(Sno, Cno, Oldgrade, Newgrade)，其中， Oldgrade 是修改前的分数，Newgrade 是修改后的分数

先切换到 test 数据库，数据库中已经有学生课程表。

①建表

```
test=# CREATE TABLE SC_U(
    Sno CHAR(9),
    Cno CHAR(4),
    Oldgrade SMALLINT,
    Newgrade SMALLINT,
    FOREIGN KEY(Sno) REFERENCES Student(Sno),
    FOREIGN KEY(Cno) REFERENCES Course(CNO)
);test(# test(# test(# test(# test(# test(#
CREATE TABLE
```

②创建触发器

```
test=# CREATE OR REPLACE FUNCTION INSERT_SC_U()
RETURNS TRIGGER AS $$
DECLARE
BEGIN
    IF(NEW.Grade >= 1.1*OLD.Grade)
    THEN INSERT INTO SC_U VALUES(OLD.Sno,OLD.Cno,OLD.Grade,NEW.Grade);
    END IF;
    RETURN NEW;
END
$$ LANGUAGE PLPGSQL;test-# test$# test$# test$# test$# test$# test$# te
CREATE FUNCTION
```

```
CREATE TRIGGER SC_AFTER_UPDATE
AFTER UPDATE ON SC
FOR EACH ROW
EXECUTE PROCEDURE INSERT_SC_U();tes
CREATE TRIGGER
```

③检验

```
test=# UPDATE SC SET Grade=100 WHERE Sno='201215122' AND Cno='2';
UPDATE 1
test=# SELECT * FROM SC_U;
    sno     | cno  | oldgrade | newgrade
------------+------+----------+----------
 201215122 | 2    |       90 |      100
(1 row)
```

```
test=# UPDATE SC SET Grade=90 WHERE Sno='201215121' AND Cno='2';
UPDATE 1
test=# SELECT * FROM SC_U;
    sno     | cno  | oldgrade | newgrade
------------+------+----------+----------
 201215122 | 2    |       90 |      100
(1 row)
```

第一条修改满足了条件，被记录下来，第二条 85->90 不满足 1.1，没被记录下来

9. 将触发器 sc_after_update 改名为 update_sc_tri

```
ALTER TRIGGER sc_after_update  RENAME TO update_sc_tri;
```

10.禁用触发器

①数据复原

```
test=# DELETE FROM SC_U WHERE Newgrade=100;
UPDATE SC SET Grade=90 WHERE Sno='201215122' AND Cno='2';
UPDATE SC SET Grade=85 WHERE Sno='201215121' AND Cno='2';DELETE 1
test=# UPDATE 1
```

②修改表

```
=> ALTER TABLE SC DISABLE TRIGGER SC_AFTER_UPDATE;
TER TABLE
```

重新修改数据，SC_U 中无内容

```
test=# SELECT * FROM SC_U;
 sno | cno | oldgrade | newgrade
-----+-----+----------+----------
```

11.删除触发器

```
test=# DROP TRIGGER SC_AFTER_UPDATE ON SC;
DROP TRIGGER
```

```
test=# DROP TRIGGER INSERT_OR_UPDATE_SAL ON Teacher;
DROP TRIGGER
```

思考：在对表的数据进行改动的时候，可以提前做出判断，避免错误的发生；在改动数据的时候，对该项改动进行记录；

3.实验总结

3.1 完成的工作

建表；

创建函数；

创建触发器；

对触发器进行测试；

3.2 对实验的认识

学会了触发器的使用；

3.3 遇到的困难及解决方法

无