

数据库系统课程实验报告

|  |  |
| --- | --- |
| 实验名称： | 实验八：触发器 |
| 实验日期： | 2024.5.24—5.25 |
| 实验地点： | 文宣楼B313 |
| 提交日期： | 2024.5.25 |
|  | |
| 学号： | 37220222203791 |
| 姓名： | 魏一诺 |
| 专业年级： | 软工2022级 |
| 学年学期： | 2023-2024学年第二学期 |

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

1. 实验内容和步骤

（0）连接到数据库

（1）创建部门表 dept(deptno, deptname)，其中，

 deptno 为部门号，定长为 2 的字符型，主码

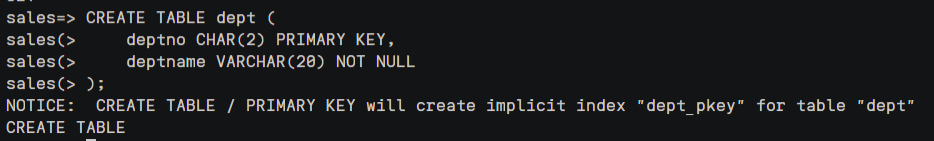
 deptname 为部门名，最大长度为 20 的变长字符型，非空

CREATE TABLE dept (

deptno CHAR(2) PRIMARY KEY,

deptname VARCHAR(20) NOT NULL

);



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

 ID 为教工号，定长为 5 的字符型，主码

 JOB 为职称，最大长度为 20 的变长字符型，非空

 SAL 为工资，长度为 7 的数字型，其中保留两位小数

 deptno 为部门号，定长为 2 的字符型，外码，引用 dept 表中的主码 deptno

CREATE TABLE Teacher (

ID CHAR(5) PRIMARY KEY,

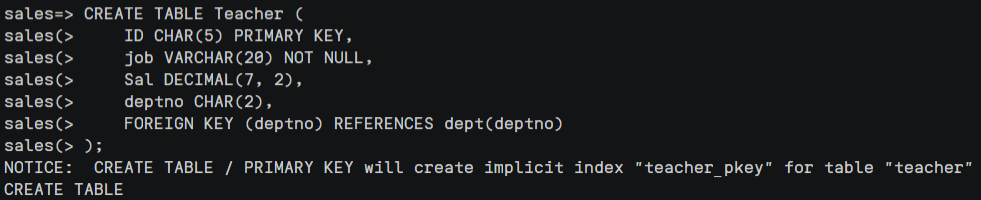
job VARCHAR(20) NOT NULL,

Sal DECIMAL(7, 2),

deptno CHAR(2),

FOREIGN KEY (deptno) REFERENCES dept(deptno)

);



（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’)

INSERT INTO dept (deptno, deptname) VALUES

('01', 'CS'),

('02', 'SW'),

('03', 'MA');

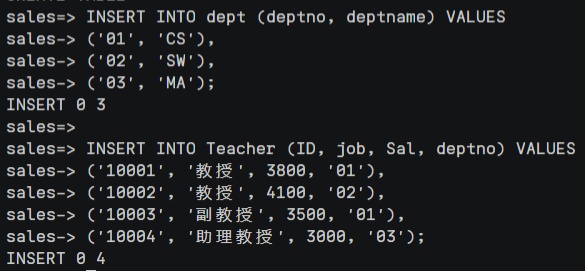
INSERT INTO Teacher (ID, job, Sal, deptno) VALUES

('10001', '教授', 3800, '01'),

('10002', '教授', 4100, '02'),

('10003', '副教授', 3500, '01'),

('10004', '助理教授', 3000, '03');



（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。

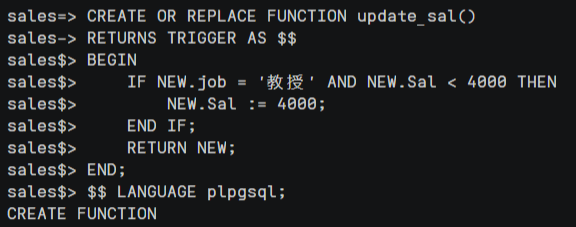
sales=> CREATE TRIGGER insert\_or\_update\_sal

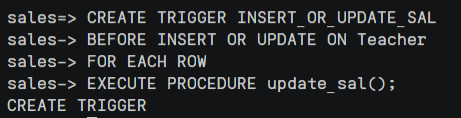
sales-> BEFORE INSERT OR UPDATE ON Teacher

sales-> FOR EACH ROW

sales-> EXECUTE PROCEDURE

sales-> enforce\_minimum\_salary();





（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

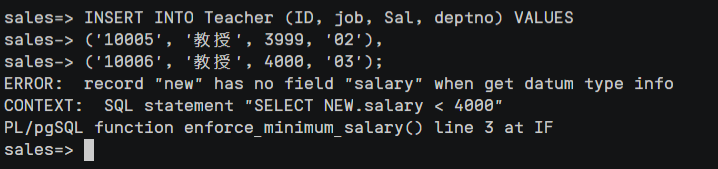
是否被触发？工作是否正确？如果正确，请观察 Teacher 表中数据的变化是否与预期一致。

1. 插入两条新数据(‘10005’,‘教授’,3999,’02’), (‘10006’,‘教授’,4000,’03’);

INSERT INTO Teacher (ID, job, Sal, deptno) VALUES

('10005', '教授', 3999, '02'),

('10006', '教授', 4000, '03');

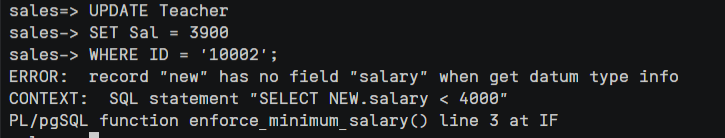


B. 更新数据:将 id 为 10002 的教授工资改为 3900。

UPDATE Teacher

SET Sal = 3900

WHERE ID = '10002';

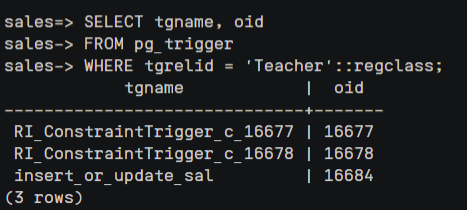


（6）查看触发器（名称和代码）；

SELECT tgname, oid

FROM pg\_trigger

WHERE tgrelid = 'Teacher'::regclass;



（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。

CREATE OR REPLACE FUNCTION delete\_teacher\_on\_dept\_delete\_function()

RETURNS TRIGGER AS $$

BEGIN

DELETE FROM Teacher WHERE deptno = OLD.deptno;

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_teacher\_on\_dept\_delete

BEFORE DELETE ON dept

FOR EACH ROW

EXECUTE PROCEDURE delete\_teacher\_on\_dept\_delete\_function();

思考并验证：如何实现在删除 dept 表中 deptno 为 01 的数据后自动将 teacher 表中引用了该值（即 01）的 deptno 字段值设置为 null。

CREATE OR REPLACE FUNCTION set\_teacher\_deptno\_to\_null()

RETURNS TRIGGER AS $$

BEGIN

UPDATE Teacher

SET deptno = NULL

WHERE deptno = OLD.deptno;

RETURN OLD;

END;

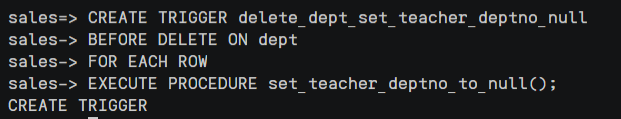
$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_dept\_set\_teacher\_deptno\_null

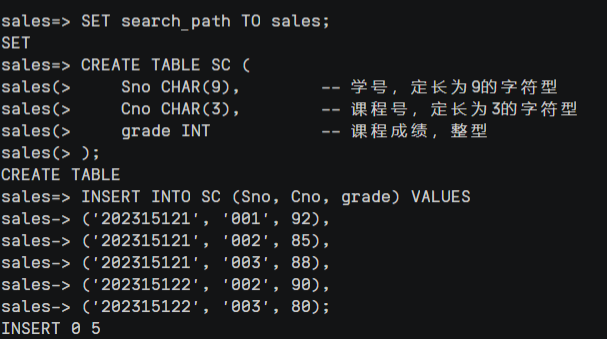
BEFORE DELETE ON dept

FOR EACH ROW

EXECUTE PROCEDURE set\_teacher\_deptno\_to\_null();



（8）现有选修表 SC(Sno,Cno,grade),各字段含义分别为学号（数据类型：定长为 9 的字符型），课程号（数据类型：定长为 3 的字符型）和课程成绩（数据类型：长度为 3 的整型），样本数据集为{(202315121,1,92),(202315121,2,85),(202315121,3,88),(202315122,2,90),(202315122,3,80)} 。



要求设计一个触发器实现审计日志记录：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)，其中各字段含义分别为学号(同 SC.SNO)，课程号(同 SC.CNO)，修改前的分数和修改后的分数(数据类型同 sc.grade)。

1. 创建 SC\_U 表：SC\_U(Sno, Cno, Oldgrade, Newgrade)

CREATE TABLE SC\_U (

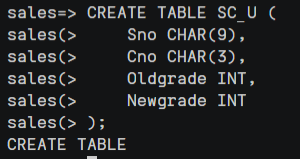
Sno CHAR(9),

Cno CHAR(3),

Oldgrade INT,

Newgrade INT

);



1. 创建 SC 表上的 AFTER 行级触发器，触发器名为 tri\_update\_sc

-- 创建触发器函数

CREATE OR REPLACE FUNCTION update\_sc\_trigger\_function()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.grade >= OLD.grade \* 1.1 THEN

INSERT INTO SC\_U (Sno, Cno, Oldgrade, Newgrade)

VALUES (NEW.Sno, NEW.Cno, OLD.grade, NEW.grade);

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

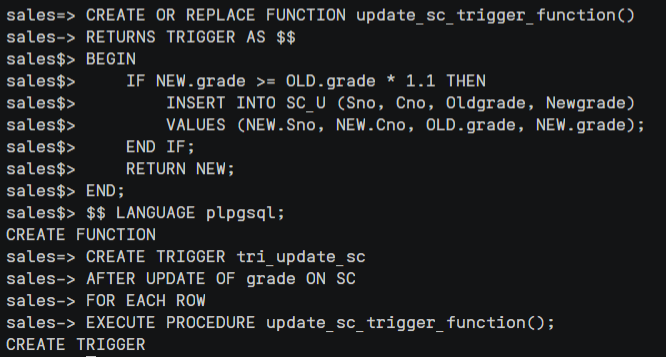
-- 创建触发器

CREATE TRIGGER tri\_update\_sc

AFTER UPDATE OF grade ON SC

FOR EACH ROW

EXECUTE PROCEDURE update\_sc\_trigger\_function();



1. 验证 tri\_update\_sc 触发器是否正常工作。

-- 第一次 UPDATE 操作

UPDATE SC

SET grade = 100

WHERE sno = '202315122' AND cno = '002';

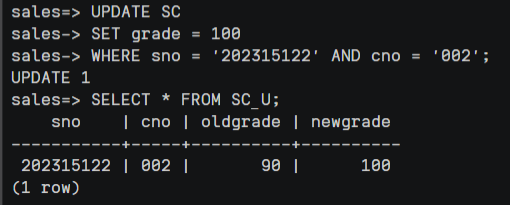
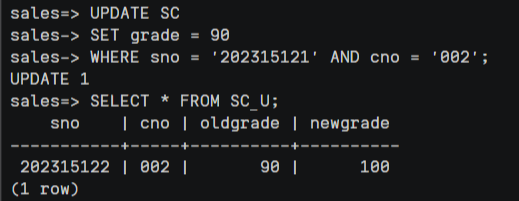
-- 第二次 UPDATE 操作

UPDATE SC

SET grade = 90

WHERE sno = '202315121' AND cno = '002';

要求：执行上述两种操作，如果触发器正确工作，请观察 SC\_U 表中数据的变化。

（9）将触发器 tri\_update\_sc 改名为 update\_sc\_tri;

ALTER TRIGGER tri\_update\_sc ON SC RENAME TO update\_sc\_tri;



（10）验证触发器禁用后效果

① 将数据还原到步骤(5)之前，即触发器工作前的原数据；

DROP TRIGGER IF EXISTS INSERT\_OR\_UPDATE\_SAL ON Teacher;

TRUNCATE TABLE Teacher;

INSERT INTO Teacher (ID, job, Sal, deptno) VALUES

('10001', '教授', 3800, '01'),

('10002', '教授', 4100, '02'),

('10003', '副教授', 3500, '01'),

('10004', '助理教授', 3000, '03');

CREATE OR REPLACE FUNCTION enforce\_minimum\_salary()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.Sal < 4000 THEN

NEW.Sal := 4000;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER INSERT\_OR\_UPDATE\_SAL

BEFORE INSERT OR UPDATE ON Teacher

FOR EACH ROW

EXECUTE PROCEDURE enforce\_minimum\_salary();

1. 修改 SC 表使 AFTER\_UPDATE\_SC 触发器失效；

ALTER TABLE SC DISABLE TRIGGER update\_sc\_tri;



③ 再次执行上面的步骤(8)之③，验证触发器被禁用后是否还能正常工作？

-- 第一次 UPDATE 操作

UPDATE SC

SET grade = 100

WHERE sno = '202315122' AND cno = '002';

-- 第二次 UPDATE 操作

UPDATE SC

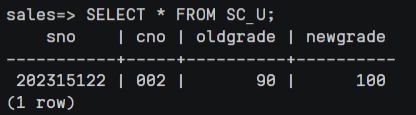
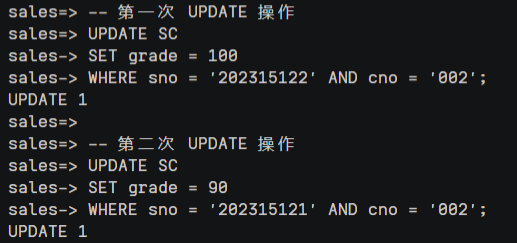
SET grade = 90

WHERE sno = '202315121' AND cno = '002';

UPDATE SC

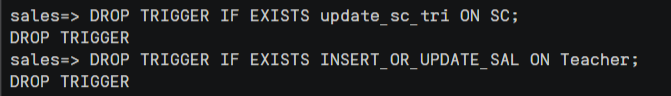
SET grade = 85

WHERE Sno = '202315121' AND Cno = '2';



（11）删除所创建的触发器

DROP TRIGGER IF EXISTS update\_sc\_tri ON SC;



1. 实验总结

3.1 完成的工作

创建了触发器 insert\_or\_update\_sal 和update\_sc\_tri

练习了禁用和启用触发器以及删除触发器。

3.2 对实验的认识

通过实验我掌握了如何编写和使用触发器函数。如何创建触发器，并指定触发器函数及触发时机。如何禁用和启用触发器。以及如何删除触发器。

3.3 遇到的困难及解决方法

有很多触发器相关操作不太熟悉，出现了创建失败、疑似创建成功但不能成功被触发等等问题，但通过上网查询资料最终解决了