

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： | 2023-5-26 |
| 实验地点： | 文宣楼B311 |
| 提交日期： | 2023-5-26 |
|  | |
| 学号： | 33920212204567 |
| 姓名： | 任宇 |
| 专业年级： | 软工2021级 |
| 学年学期： | 2022-2023学年第二学期 |

1. 实验环境

* 华为 ECS+openGauss 数据库服务器平台

前提：openGauss 数据库服务器正常运行

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

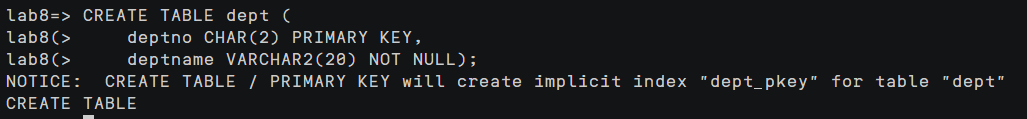
* 熟练掌握 openGauss 触发器的设计方法
* 熟练掌握 openGauss 触发器的定义、查看、禁止、启用和删除

1. 实验内容和步骤
2. 创建部门表 dept(deptno, deptname)：

**CREATE TABLE dept (**

**deptno CHAR(2) PRIMARY KEY,**

**deptname VARCHAR2(20) NOT NULL);**



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

**CREATE TABLE Teacher (**

**ID CHAR(5) PRIMARY KEY,**

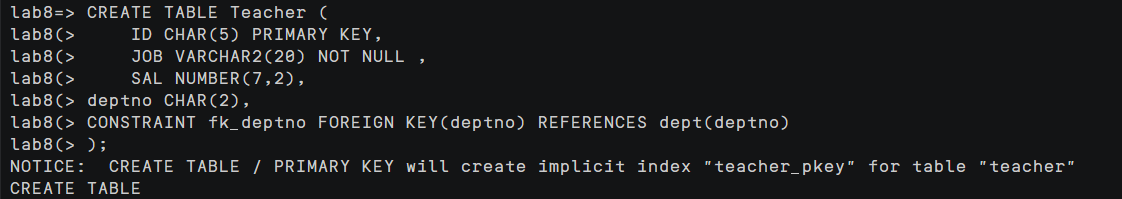
**JOB VARCHAR2(20) NOT NULL ,**

**SAL NUMBER(7,2),**

**deptno CHAR(2),**

**CONSTRAINT fk\_deptno FOREIGN KEY(deptno) REFERENCES dept(deptno)**

**);**

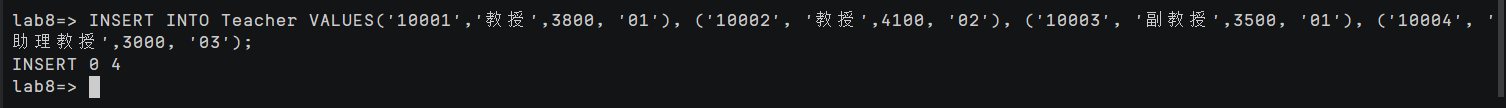


1. 为dept表增加实验数据：('01', 'CS'), ('02', 'SW'), ('03', 'MA')；为Teacher表增加实验数据：('10001','教授',3800, '01'), ('10002', '教授',4100, '02'), ('10003', '副教授',3500, '01'), ('10004', '助理教授',3000, '03'):

**INSERT INTO dept VALUES ('01', 'CS'), ('02', 'SW'), ('03', 'MA');**

**INSERT INTO Teacher VALUES('10001','教授',3800, '01'), ('10002', '教授',4100, '02'), ('10003', '副教授',3500, '01'), ('10004', '助理教授',3000, '03');**





1. 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:

**CREATE FUNCTION func\_Teacher\_INSERT\_OR\_UPDATE\_SAL ()**

**RETURNS TRIGGER AS $Teacher\_INSERT\_OR\_UPDATE\_SAL$**

**BEGIN**

**IF (new.JOB='教授') AND (new.SAL < 4000)**

**THEN new.SAL := 4000;**

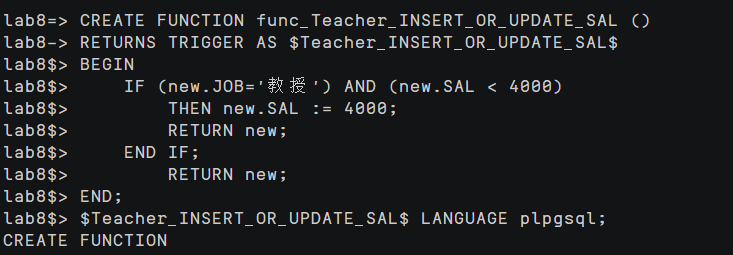
**RETURN new;**

**END IF;**

**RETURN new;**

**END;**

**$Teacher\_INSERT\_OR\_UPDATE\_SAL$ LANGUAGE plpgsql;**

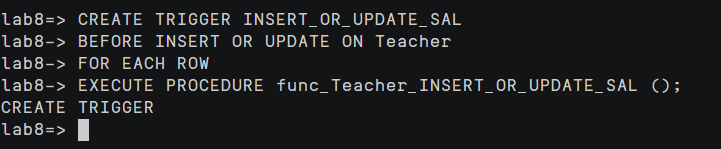


**CREATE TRIGGER INSERT\_OR\_UPDATE\_SAL**

**BEFORE INSERT OR UPDATE ON Teacher**

**FOR EACH ROW**

**EXECUTE PROCEDURE func\_Teacher\_INSERT\_OR\_UPDATE\_SAL ();**

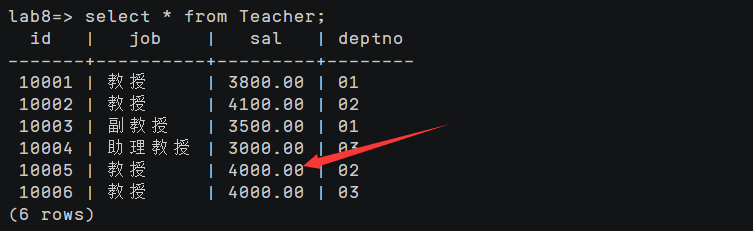


1. 验证触发器是否正常工作：分别执行以下 A，B 两种操作，验证 INSERT\_OR\_UPDATE\_SAL 触发器是否被触发？工作是否正确？如果正确，请观察 Teacher 表中数据的变化是否与预期一致:

A. 插入两条新数据('10005', '教授',3999, '02'), ('10006', '教授',4000, '03');

**INSERT INTO Teacher VALUES('10005', '教授',3999, '02'), ('10006', '教授',4000, '03');**

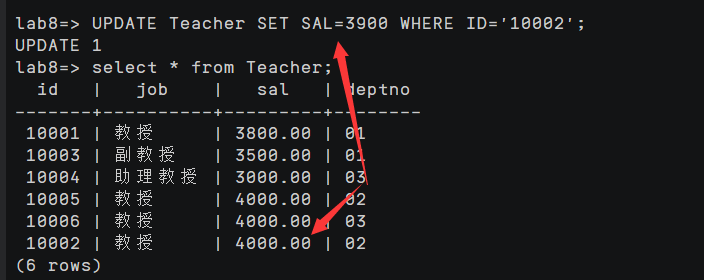




可见触发器被触发且正确工作，将3999改为4000。

B. 更新数据:将 id 为 10002 的教授工资改为 3900。

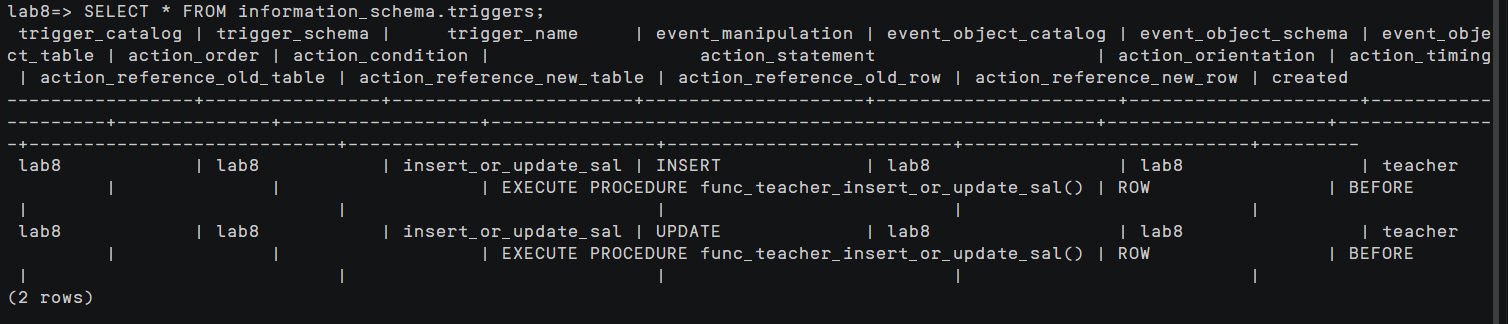
**UPDATE Teacher SET SAL=3900 WHERE ID='10002';**



可见触发器被触发且正确工作，将3900改为4000

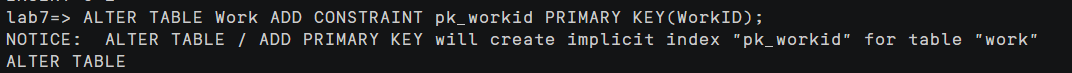
1. 查看触发器（名称和代码）:

**SELECT \* FROM information\_schema.triggers;**



1. 设置工作表的 WorkID 为主码:

**ALTER TABLE Work ADD CONSTRAINT pk\_workid PRIMARY KEY(WorkID);**



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

**CREATE TABLE SC\_U(**

**Sno CHAR(9),**

**Cno CHAR(4),**

**Oldgrade NUMBER(3,0),**

**Newgrade NUMBER(3,0),**

**CONSTRAINT fk\_Sno FOREIGN KEY (Sno) REFERENCES Student(Sno)**

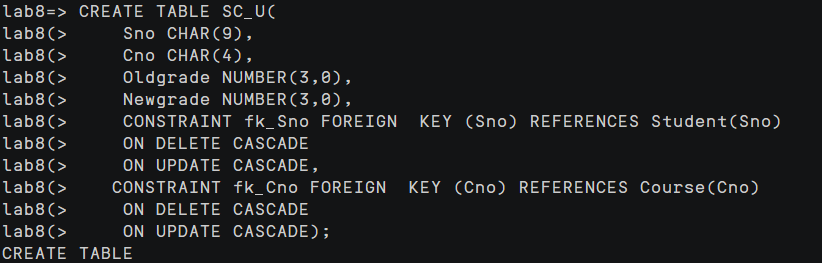
**ON DELETE CASCADE**

**ON UPDATE CASCADE,**

**CONSTRAINT fk\_Cno FOREIGN KEY (Cno) REFERENCES Course(Cno)**

**ON DELETE CASCADE**

**ON UPDATE CASCADE);**



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

**CREATE OR REPLACE FUNCTION func\_tri\_update\_sc()**

**RETURNS TRIGGER AS $update\_sc$**

**BEGIN**

**IF(NEW.Grade >= 1.1 \* OLD.Grade) THEN**

**INSERT INTO SC\_U**

**VALUES(OLD.Sno, OLD.Cno, OLD.Grade, NEW.Grade);**

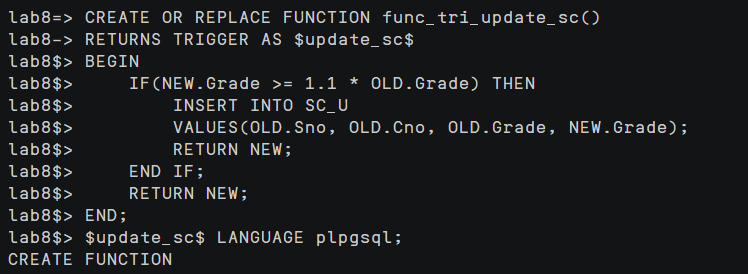
**RETURN NEW;**

**END IF;**

**RETURN NEW;**

**END;**

**$update\_sc$ LANGUAGE plpgsql;**

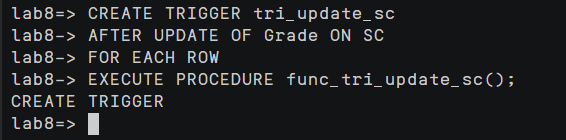


**CREATE TRIGGER tri\_update\_sc**

**AFTER UPDATE OF Grade ON SC**

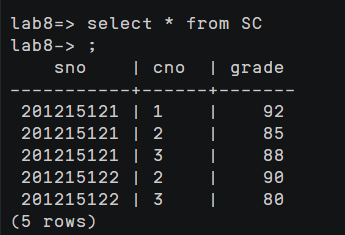
**FOR EACH ROW**

**EXECUTE PROCEDURE func\_tri\_update\_sc();**

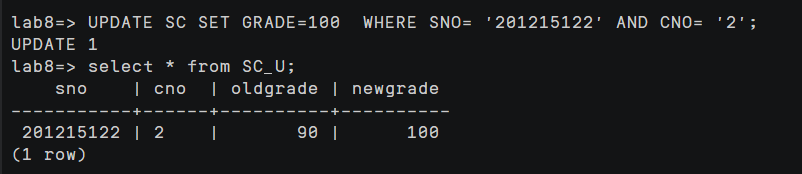


1. 验证 tri\_update\_sc 触发器是否正常工作（测试数据同教材）

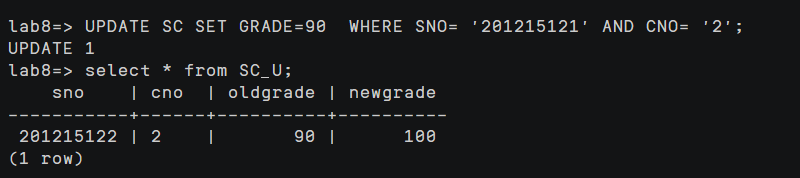
SC表初始值为



**UPDATE SC SET GRADE=100 WHERE SNO=** **'201215122' AND CNO=** **'2';**



**UPDATE SC SET GRADE=90 WHERE SNO=** **'201215121' AND CNO=** **'2';**



可见触发器正常工作

1. 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;**

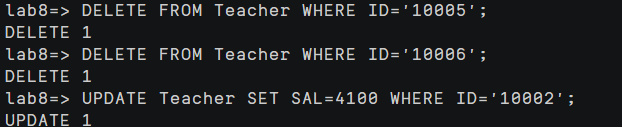


1. 验证触发器禁用后效果:
2. 将数据还原到步骤(5)之前，即触发器工作前的原数据

**DELETE FROM Teacher WHERE ID='10005';**

**DELETE FROM Teacher WHERE ID='10006';**

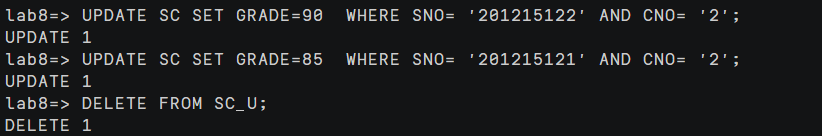
**UPDATE Teacher SET SAL=4100 WHERE ID='10002';**



**UPDATE SC SET GRADE=90 WHERE SNO=** **'201215122' AND CNO=** **'2';**

**UPDATE SC SET GRADE=85 WHERE SNO=** **'201215121' AND CNO=** **'2';**

**DELETE FROM SC\_U;**



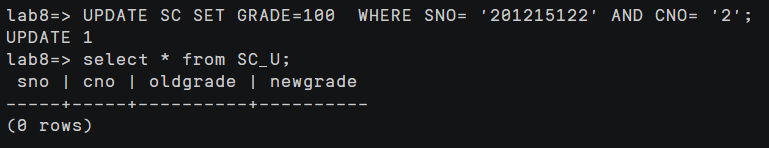
1. 修改 SC 表使 AFTER\_UPDATE\_SC 触发器失效

**ALTER TABLE SC DISABLE TRIGGER update\_sc\_tri;**

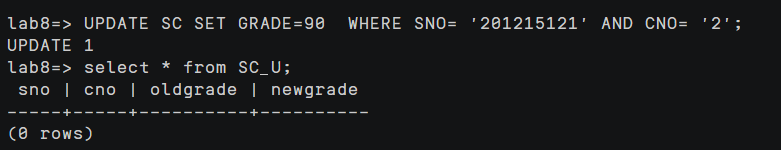


1. 再次执行上面的步骤③，验证触发器被禁用后是否还能正常工作

**UPDATE SC SET GRADE=100 WHERE SNO=** **'201215122' AND CNO=** **'2';**



**UPDATE SC SET GRADE=90 WHERE SNO=** **'201215121' AND CNO=** **'2';**

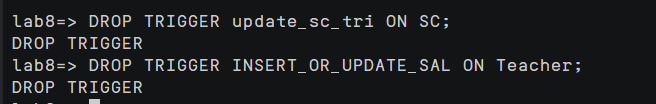


从SC\_U表可以看出，在禁用触发器后，触发器不再工作

1. 删除所创建的触发器:

**DROP TRIGGER update\_sc\_tri ON SC;**

**DROP TRIGGER INSERT\_OR\_UPDATE\_SAL ON Teacher;**



1. 实验总结

4.1 完成的工作

1. 创建表以及在表上进行触发器的操作，如添加触发器，修改触发器，删除触发器等，同时通过UPDATE，INSERT语句验证触发器是否正常工作。

4.2 对实验的认识

通过本次实验，学习到了openGuass中先定义触发动作体再定义触发器的方法，对触发器的使用有了更深入地掌握。

4.3 遇到的困难及解决方法

无。

1. 实验思考

 简述 openGauss 触发器的作用及适用场景。

答：触发器是用户定义在基本表上的一类由事件驱动的特殊过程。任何用户对表的增、删、改操作均由服务器自动激活相应的触发器，在关系数据库管理系统核心层进行集中的完整性控制。触发器类似于约束，但是比约束更加灵活，可以实施更为复杂的操作和检查，具有更精细和更强大的数据控制能力。

触发器不仅可以用于数据库完整性检查，也可以实现数据库系统的其他功能，包括数据库安全性，以及更加广泛的应用系统的一些业务流程和控制流程、基于规则的数据和业务控制功能等。