MYSQL触发器

1）实验目的

掌握数据库触发器的设计和使用方法

（2）实验内容和要求

定义BEFORE触发器和AFTER触发器。能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。

（3）实验重点和难点

实验重点：触发器的定义。

实验难点：利用触发器实现较为复杂的用户自定义完整性。

**(1).    AFTER触发器**

i.  在takes表上定义一个UPDATE触发器，当takes表中元组的属性grade被更新时，意味着学生上的这么课的课程成绩被更新，需要用触发器维护student里元组的tot\_cred属性，使其获得的总学分保持实时更新。只有当属性grade从空值或者’F’被更新为代表课程已经完成的具体分数时，触发器才会被激发。

[复制代码](javascript:void(0);)

delimiter

create trigger credit\_get after update

on takes for each row

begin

    if (NEW.grade <> 'F' and NEW.grade is not null

              and (OLD.grade = 'F' or OLD.grade is null))

       then

    update student

    set tot\_cred = tot\_cred +

              (select credits

        from course

        where course.course\_id = NEW.course\_id)

        where student.ID = NEW.ID;

 end if;

 end;

[复制代码](javascript:void(0);)

ii.  在section表上定义一个INSERT触发器，当对课程section表执行任何插入操作后触发器被启动，以确保插入元组的time\_slot\_id属性即上课时间段是合法的，受time\_slot表的time\_slot\_id约束，检查插入时的参照完整性。

[复制代码](javascript:void(0);)

delimiter

create trigger timeslot\_check1 after insert

on section for each row

begin

       if(NEW.time\_slot\_id not in

              (select time\_slot\_id

         from time\_slot))

       then

       delete from section

    where time\_slot\_id = NEW.time\_slot\_id;

end if;

end

[复制代码](javascript:void(0);)

iii.  在time\_slot表上定义一个DELETE触发器，当删除的元组的time\_slot\_id不在删除后的time\_slot表中时，且section中存在包含这个time\_slot\_id 值的元组时，触发器被启动，以确保time\_slot中发生删除时，section的参照完整性。

[复制代码](javascript:void(0);)

delimiter

create trigger timeslot\_check2 after delete

on time\_slot for each row

begin

       if(OLD.time\_slot\_id not in

              (select time\_slot\_id

         from time\_slot)

         and OLD.time\_slot\_id in

              (select time\_slot\_id

         from section))

       then

       insert into time\_slot

              values(OLD);

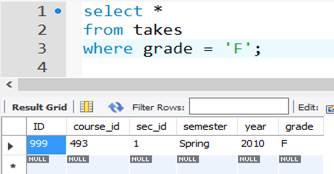
end if;

end

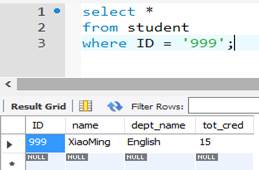
[复制代码](javascript:void(0);)

iv.   验证定义在takes表上的AFTER UPDATE触发器

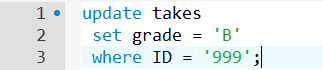
  初始时编号999的学生在493这门课成绩为F，即不通过，无法获得学分：



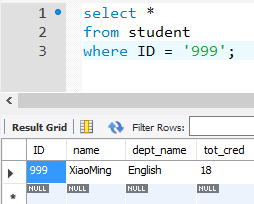
  且其已获学分为15：



  假设补考过后其成绩为B，需更新takes：



  定义在takes表上的UPDATE触发器触发，此时学生的总学分应该增加：



从上图可以看到UPDATE触发器触发成功。

**(2).    BEFORE触发器**

Before触发器的update，insert，delete语句语法和after触发器基本类似，故此下面不再赘述全部三种语句，只拿BEFORE INSERT触发器做为例子。

i.                     在takes表上定义一个before insert触发器，插入一条记录时，假设所插入的分数的

值为空白则表明该分数发生缺失，所以定义此触发器在分数值条件满足时触发用null值代替空白。

[复制代码](javascript:void(0);)

delimiter

create trigger setnull before insert

on takes for each row

begin

       if(NEW.grade = ' ')

    then

    set NEW.grade = null;

end if;

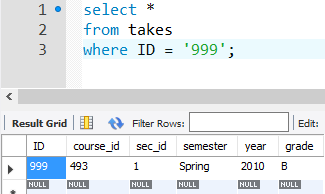
end

[复制代码](javascript:void(0);)

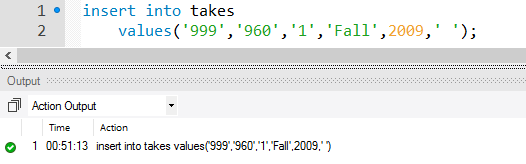
ii．  验证定义在takes表上的BEFORE INSERT触发器

 初始时：

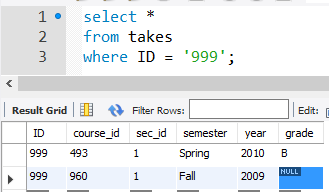
 编号为999的学生只有一门课



 假设该学生现在上了另一门课，插入元组，且该课程成绩缺失还没给出：



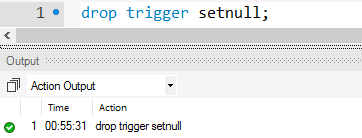
 查看该学生所上课程的信息：



 从上表可以看出takes表上的BEFORE INSERT触发器触发成功。

iii．  删除触发器

 drop trigger setnull;



总结：

after触发器—是在记录操纵之后触发，是先完成数据的增删改，再触发，触发的语句晚于监视的增删改操作，无法影响前面的增删改动作  
before触发器—是在记录操纵之前触发，是先完成触发，再增删改，触发的语句先于监视的增删改，我们就有机会判断，修改即将发生的操作，如：我们在触发之前需要判断new值和old值的大小或关系，如果满足要求就触发，不通过就修改再触发；如：表之间定义的有外键，在删除主键时，必须要先删除外键表，这时就有先后之分，这里before相当于设置了断点，我们可以处理删除外键。

对于INSERT语句, 只有NEW是合法的；

对于DELETE语句，只有OLD才合法；

对于UPDATE语句，NEW、OLD可以同时使用。