使用触发器提高信息系统性能和开发效率

杨莉萍 （济南山东财政学院信息系 250014）

摘要：本文介绍了触发器的运行机制及其创建语法，并通过实例详细讨论了触发器在维护数据的完整性和有效性方面的具体应用，阐明了在信息系统开发中合理使用触发器能够改善系统的性能、提高系统的开发效率，同时还提出了使用过程中的有关注意事项。

关键词：触发器 信息系统 参照完整性 有效性检测

1. 引言

触发器是大型数据库管理系统在保证数据的完整性和一致性方面提供的一项有效机制。所谓触发器，就是存在于数据库中的过程，它被绑定到一个指定的数据表上，在对该表中的数据进行指定操作后被自动激发，能够激发触发器的操作可以是插入、删除和数据更新三种。

触发器中可以使用 SQL 语句，还可以使用控制流语句，因此在触发器中可以进行判断和分析，根据不同的条件采取不同的行动。此外触发器通过检测认为不合理的操作，可以直接进行回滚，数据库管理系统将触发器和启动它的语句当作一个事物来处理，因此它回退的是整个事物，即事物在触发器中可以被回退。

正是由于触发器的上述特点，它在下述几个方面非常实用：第一维护数据的参照完整性。比如，在一些相互关联的数据表中，字段之间常会有数据依赖关系，也就是说某个字段是另一个字段的衍生，那么在基础字段的数据发生变化时，衍生字段也要跟着进行变化，否则就会破坏数据的相关完整性，象这种级联式的操作就可以使用触发器来完成。第二产生比规则更为复杂的限制，来保证数据的有效性和安全性。比如商务管理系统，商品的调价要严格遵循商业管理规定，调价幅度必须控制在原价格的某个百分比内，象这类审计工作就可以通过触发器来实现，碰到那些破坏这种规定的操作，触发器会自动拒绝。

在信息系统开发中，使用触发器实现数据的参照完整性维护和有效性检测，会使系统的性能更加完善。因为触发器的设定属于数据库定义的工作范畴，而这一部分工作在系统开发中是由系统总设计师来完成的，这样就把数据的完整性维护和有效性检测纳入到系统的总体规划中，克服了由前端应用开发人员完成这类任务时因受局部限制而出现的不完备性。再者由于触发器是被定义到数据表上的，不管最初的事务发生在什么地方，在对表进行相应的操作后，都会被自动引发，这就确保了数据完整性维护和有效性检测的及时实施。还有，触发器本身的开销很小，它的存储、引发和执行都是在服务器上，网络传送量几乎为零，不会因为网络拥挤而造成时间延迟，因而使用触发器改善了系统的执行效率。另外从系统开发的角度来看，合理地使用触发器会大大提高系统的开发效率。这是因为在前端应用中对数据操作有很多种情况，不管在那种情况下进行操作都必须保证数据的完整性和有效性，如果这些工作让前端应用来完成的化，那么各种情况都必须进行考虑，工作量无疑是很大的，采用触发器，数据的完整性维护和有效性检测工作会在每次数据操作后自动进行，前端应用根本无须考虑，大幅度地减轻前端开发人员的工作量，同时也减少了由于工作繁琐而带来的出错率。

触发器的创建语法随不同的数据库管理系统略有不同，下面就以 SQL SERVER 作为 DBMS 平台，介绍触发器的创建及其在应用中的使用。

2. 触发器的创建和删除语法

创建一个触发器的语法如下：

```
CREATE TRIGGER trigger-name
  ON table-name
  FOR INSERT/UPDATE/DELETE/NSERT, UPDATE
  AS
  SQL-STATEMENTS
  [ IF UPDATE(col-name1) [AND/OR UPDATE(col-name2)...]]
  SQL-STATEMENTS
```

语法中共有四个组成部分，它们分别是：

·触发器名 触发器的名字

·表名　　触发器所针对的表

·触发操作　启动触发器的操作，可以是插入操作、更新操作、删除操作或插入更新操作

·触发过程　指定触发条件和动作，由 SQL 语句和控制流语句构成，其中 UPDATE(col-name1)表示列

col-name1 的内容在操作中被改变了，此外过程中还可以包含 print 子句，用来向操作实施者返回信息。

触发器的创建必须在其所涉及到的表都已创建后才能进行，一个表最多能为其创建三个触发器，一个插入触发器、一个更新触发器及一个删除触发器。如果对应操作已有了一个触发器，再创建一个，它会自动将原先那个覆盖，也就是说只有最后创建的那个触发器才是有效的。

如果不想使用触发器了，可以删除，删除一个触发器的语法为

DROP TRIGGER trigger-name

另外，当一个表被删除时，在其上面定义的触发器将被自动删除。

3.触发器执行过程中涉及到的临时表和全局变量

触发器被触发后，在执行期间涉及到两个临时表，一个删除表 deleted、一个插入表 inserted，这两个表是在对触发表进行操作时由系统在内存中自动建立的，两个表的结构同触发表完全一样。其中表 deleted 记载了对触发表进行操作期间被删除的那些记录；而表 inserted 则记载了对触发表操作期间新增加的那些记录。更新操作在这里被认为是先将老数据记录删除、再增加新数据记录，因此更新操作涉及的记录在两个表中都有记载。触发器被启动后，可以从这两个表中获取数据，进行操作前后的数据对比，对不合规则的操作实施回滚，或使用当前最新数据完成相关数据的更新等。

触发器是在完成指定的操作时被触发的，一条 SQL 语句仅能触发触发器一次，而一条 SQL 语句可能会影响多条记录，为此系统专门设置了一个全局变量@@rowcount，用它来记录本次操作受影响的记录数，以便触发器在需要时引用。

4.触发器的应用实例

例 1.本例针对的是库存管理系统，涉及到两个数据表，一个是库存表 kcb，记录了库中所有物品的当前库存量；一个是出库记录表 ckjlb，它记录每次物品出库的数量。由于每次物品出库都会直接影响当前的库存量，为此每次向出库记录表增加记录时，要同时修改库存表，以便保持数据的完整性。这个任务就可以使用触发器来完成。下面就是它的创建语法，创建后，每次对出库记录表增加记录时它会自动触发

```
create trigger ck
        on ckjlb
        for insert
        as
/*判断本次出库物品是否都在当前库中，如都在，则更新库存表*/
    if (select count(*) from inserted, kcb where inserted.bh = kcb.bh) = @@rowcount
    begin
    /*在库存表中减掉出库物品的数量*/
    UPDATE kcb
    SET sl = sl - (select sum(sl) from inserted
    group by inserted.bh having kcb.bh = inserted.bh)
    where kcb.bh in (select bh from inserted)
    /*判断有无出库物品，其出库数量超出了现有库存数量，如有，则拒绝本次出库操作、回滚，并给出返回信息*/
        if (select count(*) from kcb where kcb.bh in (select bh from inserted)
    and kcb.sl<0) > 0
      begin
      rollback
      print '出库数量超过了现有库存'
    end
  end
else
    /*出库物品不都在当前库中，拒绝出库操作、并给出提示信息*/
    begin
    rollback
    print '出库物品在现有库中不存在'
    end ;
```

例 2.本例针对的是物价管理系统，涉及到一个数据表，物价表 wjb。商品的物价会经常进行调整，为了避免失控，有关部门制定了相应的物价管理规定，通常将其调整幅度限定在原价格的 10% 内。因此在对物价表中的价格进行改动时，要同时进行检测，超过这个范围的价格变动，系统将拒绝接受。这个任务也可以由触发器来完成，语法如下：

（下转第 58 页）

(上接第 37 页)

```
create trigger wjtz
        on wjb
        for update
        as
/*判断是否修改了商品价格,若是,则要对本次操作进行检测*/
if update(jg)
/*检测本次物价改动,是否有调价物品违反了物价管理规定,如有,则拒绝本次操作,并给出提示信息*/
if (select count(*) from deleted, inserted where (deleted.sphh = inserted.spbh) and
abs(inserted.jg - deleted.jg) > deleted.jg * 0.1) > 0
  begin
  rollback
  print '本次物价改动是否违反了物价管理规定'
end;
```

5. 使用触发器时应注意的事项

在信息系统开发过程中,使用触发器一定要在对整个数据库进行了全面的分析、搞清各表之间的相互关系后方可进行,因为只有这样才能明确什么地方可以使用,什么地方不能使用,否则盲目使用很容易造成触发器的循环触发,即源触发器又被由它引发的触发器所出发,从而无穷循环下去,这样不仅达不到使用触发器增强数据完整性的目的,反而破坏了数据库的基本功能。另外,在使用触发器时还要注意的一个问题就是触发器的开销,触发器在每次进行规定的操作时都自动触发,如果触发器中包含相当数量的代码,那么所有与它相关的事务都可能是很耗时的,在一个存在大量事物的系统中,这显然会大大降低系统的处理速度,此时可考虑放弃使用触发器,采用其他的数据库设计方法来提高系统的执行效率。

(来稿时间:1998 年 3 月)