使用触发器维护数据的相关完整性

陆 鑫*
(电子科技大学计算机系 成都 610054)

【摘要】 描述了在数据库应用系统中利用触发器维护数据相关完整性的实际意义；介绍了在SYBASE 数据库应用系统中创建和删除触发器的基本语法，并对触发器涉及的临时表和全局变量进行了说明；给出了使用触发器维护数据相关完整性的应用方法和应用实例。

关 键 词 SYBASE 数据库； 触发器； 相关完整性； 应用系统
中图分类号 TP311.52

在数据库应用系统中，确保数据的相关完整性是系统的基本要求之一。所谓数据的相关完整性是指数据库相关表联系字段值的一致性。微机型关系数据库系统一般是把数据相关完整性的维护工作留给应用程序去处理。这样的应用程序虽然控制较直观，但系统的安全性容易失控，程序维护较困难，此外系统性能也难以提高。而大型关系数据库系统(如 SYBASE)提供了触发器机制来支持系统对数据的相关完整性进行维护处理。采用触发器维护数据的完整性具有如下优点：1) 触发器是定义到一个特定的数据库表上。在对该表进行相应数据操作时，触发器程序会被自动激发执行，这样可确保数据完整性维护的及时实施。2) 触发器的存储与执行均在数据库服务器上，它一方面能充分利用服务器的高计算性能，另一方面又不需要网络开销，因而能提高系统维护数据完整性的运行效率。3) 触发器可利用流控制语句完成复杂的判断和运算处理，大大增强了数据完整性的控制功能与灵活性。4) 当触发器程序的内部操作语句和流程改变后，它可在数据库服务器上重新创建生成，而不影响客户端应用程序的改变，系统的可维护性强并且简单。因此，在数据库应用系统中，充分利用触发器来完成数据的完整性维护处理，可保证数据完整性维护的及时性、高效性、灵活性和可维护性。本文以 SYBASE 大型数据库管理系统为背景，介绍触发器的创建方法和应用实例。

1 触发器的创建与删除语法

1.1 触发器的创建

在 SYBASE 中创建一个触发器的命令语法为[1]：

```
CREATE TRIGGER [owner.]trigger_name  ON [owner.]table_name
    FOR {INSERT,UPDATE,DELETE}
AS sql_statements
```

该命令中的选项含义如下：

- trigger_name —— 触发器名，它必须符合 SYBASE 命名约定；
- table_name —— 数据库表名，触发器所绑定的数据库表名；
- INSERT,UPDATE,DELETE —— 触发操作关键字，决定哪个操作(INSERT,UPDATE,DELETE)可激发触发器程序；
- sql_statements —— 触发器程序，它由 SQL 语句和流控制语句集合构成。

下面介绍一个触发器定义的简单例子。每次试图在 current_store 表中插入、删除或更新数据时，该触发器显示信息：

```
create trigger tr1 on current_store
for insert,update,delete
```

1998年11月4日收稿，1998年12月10日修改定稿
* 男 32岁 硕士 讲师

as print "现正在操作 current_store 表数据!"

1.2 触发器的删除

当不再使用触发器时，可使用删除命令删除该触发器，其命令语法为[2]:

DROP TRIGGER [owner.]trigger_name
[,[owner.]trigger_name]

例如，要删除上例所创建触发器，可执行命令 drop trigger tr1。另外，当一个数据库表被删除时，在其上面定义的触发器也将被自动删除。

2 触发器涉及的临时表与全局变量

触发器被激发后，在执行过程中涉及到两个临时表: inserted 插入表和 deleted 删除表。这两个临时表是在对触发器所定义的数据库表进行数据操作时，由系统在内存中自动建立的。inserted 插入表用来存储在 insert 插入语句或 update 更新语句中受影响的数据行的备份；在 insert 和 update 期间，新行同时被添加到 inserted 表和数据库表中，inserted 表中的数据行是数据库表中新行的备份。deleted 删除表用来存储在 delete 删除语句或 update 更新语句中受影响的数据行的备份；在 delete 和 update 期间，这些行从数据库表中删除，并被传递到 deleted 删除表，deleted 表和数据库表没有共同行。此外，触发器在执行时还涉及到一个全局变量@@rowcount。系统提供的这个全局变量用来存放本次数据库操作受影响的记录数，触发器程序可以根据需要引用它。

3 触发器维护数据相关完整性的应用方法与实例

数据的相关完整性是指数据库相关表联系字段值的一致性，即相关表的主键值与外键值的匹配。一般把主键表看作主细关系的主表，外键表看作主细关系的子表。在对主表或子表进行数据记录的插入、删除和修改操作过程中，需要对这些相关表的主键值或外键值进行操作维护，以保持它们的一致性。具体讲，在对主表进行记录删除或更新操作时，会影响到子表外键值的匹配，因此需要在主表上创建删除触发器或更新触发器来维护子表外键值的一致性；在对子表进行记录插入或更新操作时，同样要考虑与主表主键值的匹配，因此也需要在子表上创建插入触发器或更新触发器来保持与主表主键值的一致性。下面以具体应用实例说明如何利用触发器维护数据的相关完整性。

例如在商品资料管理子系统中，有如下的商品资料表 COMMODITY(见表 1)和供应商资料表 SUPPLIERS(见表 2)。它们通过 SUPPLIERS 的主键(CODE 供应商编码)与 COMMODITY 表的外键(SUP_CODE供应商代码)建立相关关系。当对供应商资料表SUPPLIERS或商品资料表COMMODITY的数据记录进行操作处理时，需考虑维护它们的相关数据的一致性。这可通过在 SUPPLIERS 表和 COMMODITY 表上建立相应的触发器来维护数据的完整性。完成该任务的触发器如下：

表 1 商品资料表 COMMODITY

编码	列名
CODE	商品编号
NAME	商品名称
SPEC	商品规格
COST	进价
PRICE	零售价
SUP_CODE	供应商代码

表 2 供应商资料表 SUPPLIERS

编码	列名
CODE	供应商编码
NAME	供应商名称
ADDRESS	地址
TEL	电话
BANK	开户行
ACCOUNT	帐号

```
/* 创建主表 SUPPLIERS 的更新触发器 tu_suppliers */
create trigger tu_suppliers on SUPPLIERS for update as
begin
```

```
    if  @@rowcount = 0
        return
    /* 若主键码 CODE 修改，则维护子表 COMMODITY 外键码 SUP_CODE 以保持一致 */
            if update(CODE)
            begin
                update COMMODITY   set   SUP_CODE = i1.CODE
                from   COMMODITY t2, inserted i1, deleted d1
                where   t2.SUP_CODE = d1.CODE   and   (i1.CODE != d1.CODE)
            end
end
go
/* 创建主表 SUPPLIERS 的删除触发器 td_suppliers   */
create trigger td_suppliers on SUPPLIERS for delete as
begin
          if   @@rowcount = 0
             return
       /* 若一个主键码 CODE 被删除，则将子表 COMMODITY 外键码 SUP_CODE 对应的值置为
空以保持一致 */
          update COMMODITY   set    SUP_CODE = NULL   from    COMMODITY t2, deleted t1
          where   t2.SUP_CODE = t1.CODE
end
go
/* 创建子表 COMMODITY 的插入触发器 ti_commodity */
create trigger ti_commodity on COMMODITY for insert as
begin
        declare    @numrows   int, @numnull   int, @errno   int, @errmsg   varchar(255)
        select   @numrows = @@rowcount
        if @numrows = 0
          return
        /* 插入子表的外键码必须是主表存在的主键码 */
if (select count(*) from SUPPLIERS t1, inserted t2
     where   t1.CODE = t2.SUP_CODE) != @numrows
     begin
       select @errno   = 30002,
       @errmsg = '在主表"SUPPLIERS"中没有对应编码. 不能插入子表"COMMODITY"记录.'
       raiserror @errno @errmsg
            rollback   transaction
       end
end
go
/* 创建子表 COMMODITY 的更新触发器 tu_commodity   */
create trigger tu_commodity on COMMODITY for update as
begin
```

```
declare @numrows int, @numnull int, @errno int, @errmsg varchar(255)
select @numrows = @@rowcount
if @numrows = 0
    return
/* 更新子表的外键码必须是主表存在的主键码 */
    if update(SUP_CODE)
    begin
        select @numnull = (select count(*) from inserted where SUP_CODE is null)
        if @numnull != @numrows
            if (select count(*) from SUPPLIERS t1, inserted t2
                where t1.CODE = t2.SUP_CODE) != @numrows - @numnull
            begin
                select @errno = 30003, @errmsg = '主表"SUPPLIERS" 不存在对应编码. 子表"COMMODITY"的外键码不能修改.'
                raiserror @errno @errmsg
                rollback transaction
            end
        end
end
go
```

4 结 束 语

触发器是数据库服务器中的一种特殊存储过程，它被定义到一个指定的数据库表上。当对该数据库表进行数据记录的插入、删除或更新操作时，触发器程序可被自动激发执行。充分利用触发器来进行数据的完整性维护处理，可保证数据完整性维护的及时性、高效性、灵活性和可维护性。本文给出了创建触发器的基本语法、应用触发器维护数据完整性的方法和实例。

参 考 文 献

1 陶浦洲. Sybase 数据库技术大全.北京：科学出版社，1995

2 Peter Hazlehurst 著，周保力译. Sybase System XI 实用大全. 北京：清华大学出版社，1997

Using Trigger to Maintain Data's Referential Integrity

Lu Xin

(Dept.of Computer , UEST of China Chendu 610054)

Abstract This paper describes the realistic significance of using trigger to maintain data's referential integrity in database application system. The basic grammar of creating trigger and deleteing trigger in sybase database application system are introduced, and the temporary table and global variable refered are discussed. At last, the application method and example of using trigger to maintain data's referential integrity are also presented.

Key words SYBASE database; trigger; referential integrity; application system