

# 基于 SQL Server 2008 的 DML 触发器 设计实例分析

綦宝声

(山东劳动职业技术学院 信息工程与艺术设计系, 山东 济南 250022)

**摘 要:**SQL Server 数据库是当今信息管理系统中有代表性的大型网络数据库之一。文中讨论在 SQL Server 2008 中通过触发器技术实现数据完整性的机制,介绍触发器的概念、分类,重点论述 DML 触发器的实现过程,结合具体实例,设计 DML 触发器监督财务人员资金的各种业务操作,包括插入、删除和更新记录,分析了触发器设计的基本步骤,并改进触发器设计,使审计表能够记录全部变化的数据,从而反映了触发器的应用,在提高系统性能、维护数据库完整性、有效性等方面的强大功能。

**关键词:**数据库;完整性;DML 触发器;SQL Server

**中图分类号:**TP392

**文献标识码:**A

**文章编号:**1673-629X(2012)06-0229-05

## Analysis of DML Triggers Design Example Based on SQL Server 2008

QI Bao-sheng

(Department of Information Engineering and Art Design, Shandong Vocational College of Labor, Jinan 250022, China)

**Abstract:**SQL Server database is a representative large-scale network database of the information management system. It discusses, in SQL Server 2008, trigger technology to realize data integrity mechanism, introduces the concept, classification of the trigger, mainly deals with the DML trigger implementation process, combined with specific examples, the design DML triggers supervision of finance staff a variety of business operations including insert, delete and update records. Analyse the trigger design of the basic steps and optimize the design, to use the audit-table records all the change data. It reflects the trigger application in improving system performance, maintaining database integrity, validity of the powerful function.

**Key words:** database; integrity; DML triggers; SQL Server

### 1 知识准备

数据库完整性(Database Integrity)是指数据库中数据的正确性和相容性<sup>[1]</sup>。数据库完整性由各种各样的完整性约束来保证,因此可以说数据库完整性设计就是数据库完整性约束的设计。数据库完整性约束可以通过 DBMS 或应用程序来实现,基于 DBMS 的完整性约束作为模式的一部分存入数据库中,通过 DBMS 实现的数据库完整性按照数据库设计步骤进行设计,而由应用软件实现的数据库完整性则纳入应用软件设计<sup>[2]</sup>。文中主要讨论在 Microsoft SQL Server 2008 中通过触发器技术实现数据完整性的机制,通过一个实例

分析触发器设计的一般过程。

SQL Server 有效管理信息的能力源于它可以控制在系统应用中流过表及应用逻辑的数据<sup>[3]</sup>。SQL Server 将数据写入数据库之前先校验规则和默认值,类似于一种信息“预过滤器”,避免某些数据项会影响数据库完整性造成数据库中的数据冗余。

触发器是“后过滤器”,它在数据修改通过所有规则、默认值之后才执行,即它在对表进行插入、修改、删除操作后执行。因为触发器是在操作生效后执行的,因而它表示修改操作的最后一个步骤<sup>[4]</sup>。如果触发器请求失败,将拒绝修改信息,并返回错误信息。

触发器是一种特殊类型的存储过程,它不同于一般存储过程。一般存储过程通过存储过程名称被直接调用,而触发器主要是通过事件进行触发而被执行<sup>[5]</sup>。触发器是一个功能强大的工具,它与表格紧密相连,在表中数据发生变化时自动强制执行。触发器可以用于 SQL Server 约束、默认值和规则的完整性检查,还可以

收稿日期:2011-11-02;修回日期:2012-02-07

基金项目:国家自然科学基金项目(61070202);山东省高等学校优秀青年教师国内访问学者项目

作者简介:綦宝声(1969-),男,山东平度人,硕士,副教授,研究方向为程序设计、数据库。

完成难以用普通约束实现的复杂功能。

Microsoft SQL Server 提供两种主要机制来强制使用业务规则和数据完整性:约束和触发器<sup>[6]</sup>。触发器为特殊类型的存储过程,可在执行语言事件时自动生效。SQL Server 包括三种常规类型的触发器:DML 触发器、DDL 触发器和登录触发器<sup>[7]</sup>。

当服务器或数据库中发生数据定义语言 (DDL) 事件时将调用 DDL 触发器。登录触发器将为响应 LOGON 事件而激发存储过程,与 SQL Server 实例建立用户会话时将引发此事件。

当数据库中发生数据操作语言 (DML) 事件时将调用 DML 触发器。DML 事件包括在指定表或视图中修改数据的 INSERT 语句、UPDATE 语句或 DELETE 语句<sup>[8]</sup>。DML 触发器可以查询其他表,还可以包含复杂的 Transact-SQL 语句。将触发器和触发它的语句作为可在触发器内回滚的单个事务对待。如果检测到错误(例如,磁盘空间不足),则整个事务即自动回滚。

可以设计三种类型的 DML 触发器:AFTER 触发器、INSTEAD OF 触发器、CLR 触发器<sup>[9]</sup>。AFTER 触发器在 INSERT、UPDATE 或 DELETE 语句操作之后执行;INSTEAD OF 触发器代替通常的触发动作;CLR 触发器将执行在托管代码中编写的方法,而不用执行 Transact-SQL 存储过程。这里讨论的主要是前两种类型的触发器。

任何事物都不是尽善尽美的,触发器也不例外,由于触发器需要大量的代码来实现,因此运行触发器也会花费大量的时间,有时会影响到系统的运行速度。

## 2 设计触发器的过程

一般地,设计触发器的过程包括用户需求分析、确定触发器的逻辑结构、编写触发器代码和测试触发器<sup>[10]</sup>,下面结合具体实例,介绍设计触发器的一般步骤,最后,提供了该例子进一步完善的方法。

### 2.1 用户需求分析

假设某个公司的数据库中有两个表:accountData 表和 auditAccountData 表。accountData 表记录了公司重要的资金信息,且只能由公司指定的财务人员使用,财务人员可以根据业务需要修改表中的数据。为了加强公司的财务管理,auditAccountData 表监督财务人员对资金的各种业务操作,确保资金运转安全高效。其中 accountData、auditAccountData 表的模式分别为:

```
accountData ( accounted, accountType, accountAmount)
```

```
auditAccountData ( audit_log_id, audit_log_loginname, audit_log_username, audit_log_actionType, audit_log_amount, audit_log_actionTime)
```

### 2.2 确定触发器的逻辑结构

触发器由时间条件、触发事件和动作组成。确定触发器的逻辑结构,就是确定触发器的时间条件、触发事件和动作以及触发器的选项<sup>[11]</sup>。由于这是一个审计触发器,那么只有在财务人员执行操作之后,才对这些操作进行记录。因此,可以确定该触发器的时间条件为 AFTER。确定了时间条件之后,接着开始研究触发器的事件类型,审计的对象是财务人员对帐户的所有操作,这些操作包括插入数据、删除数据和更新数据,可以确定该触发器的触发事件是 INSERT、DELETE 和 UPDATE。所以,可以为该表创建 INSERT、DELETE、UPDATE 类型的 3 个触发器。

触发器的条件和事件确定之后,就需要确定触发器本身的动作。这些将要确定的动作就是编写 Transact-SQL 语句来执行审计的记录工作。虽然在 auditAccountData 表中包含了 6 个列,但是只有 audit\_log\_actionType 列和 audit\_log\_amount 列需要填写财务人员执行操作类型值和金额,而其他 4 个列的值都可以由系统自动插入。

### 2.3 编写触发器代码

创建 INSERT 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_insert
ON accountData
WITH ENCRYPTION
FOR INSERT
AS
DECLARE @insertActionAmount MONEY
SELECT @insertActionAmount = accountAmount
FROM inserted
```

```
INSERT INTO auditAccountData ( audit_log_actionType, audit_log_amount)
```

```
VALUES ( 'INSERT', @insertActionAmount)
```

```
RETURN
```

创建 DELETE 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_delete
ON accountData
WITH ENCRYPTION
FOR DELETE
AS
DECLARE @deleteActionAmount MONEY
SELECT @deleteActionAmount = accountAmount
FROM deleted
```

```
INSERT INTO auditAccountData ( audit_log_actionType, audit_log_amount)
```

```
VALUES ( 'DELETE', @deleteActionAmount)
```

```
RETURN
```

创建 UPDATE 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_update
ON accountData
WITH ENCRYPTION
FOR UPDATE
AS
DECLARE @oldValue MONEY
SELECT @oldValue = accountAmount FROM deleted
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
VALUES ( 'update_old_value ', @oldValue)
DECLARE @newValue MONEY
SELECT @newValue = accountAmount FROM in-
serted
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
VALUES ( 'update_new_value ', @newValue)
RETURN
```

## 2.4 测试触发器

触发器创建之后,在正式使用之前,应该对触发器进行测试。测试的目的是保证建立了正确的触发器,能够正常工作。首先测试插入数据的操作。下面是一组插入数据的例子,把这些数据插入到 accountData 表中。这些数据插入之后,会触发 t\_accountData\_insert 触发器的执行。

一组插入数据的操作:

```
INSERT INTO accountData VALUES( '存款', 2500)
INSERT INTO accountData VALUES( '支票', 2337)
INSERT INTO accountData VALUES( '存款', 199)
INSERT INTO accountData VALUES ( '存款',
2476300)
```

然后查看审计表 auditAccountData 中插入的数据,可以看到,触发器的操作是正确的,命令如下:

```
SELECT * FROM auditAccountData
```

下面是一组删除数据的操作,这些删除操作完成之后,会触发 t\_accountData\_delete 触发器的执行,应该在 auditAccountData 中记录下来。删除记录的命令如下:

```
DELETE FROM accountData WHERE accountID = 1
DELETE FROM accountData WHERE accountID = 2
DELETE FROM accountData WHERE accountID = 3
```

然后查看审计表 auditAccountData 中插入的数据,可以看到,触发器的操作是正确的,命令如下:

```
SELECT * FROM auditAccountData
```

对于触发器 t\_accountData\_update 可以进行类似

的测试,这里不再赘述。

## 2.5 优化触发器设计

上面的测试仅仅是一次插入、删除一条记录,但是,如果一次插入、删除多条记录,触发器的设计仅考虑记录一行数据,其他插入、删除或更新的记录信息仍然丢掉了,为了验证出现的结果,需要事先删除表 accountData、auditAccountData,再重建这两个表,然后创建插入、删除、更新触发器,在 accountData 表中插入记录,执行下面的命令,一次删除满足 accountID ≤ 3 的多条记录时,就会出现这个问题,命令如下:

```
DELETE FROM accountData WHERE accountID ≤ 3
SELECT * FROM auditAccountData
```

上面的命令一次删除三行记录,结果在 auditAccountData 表中只记录了删除的第一行记录的信息,第二、三两行记录信息丢掉了,导致记录数据的不完整。对于插入、更新触发器,也存在记录数据不完整的情况。

由于存在上面的问题,需要对触发器进行改进,使触发器能够记录插入、更新、删除多条记录的所有行的数据。

改进触发器设计之前,先补充学习触发器的背景知识。在触发器执行的时候,会产生两个临时表:inserted 表和 deleted 表<sup>[12]</sup>。它们的结构和触发器所在的表的结构相同,SQL Server 2008 自动创建和管理这些表。当一条记录插入表中时,相应的插入触发器创建一个 inserted 表,该表映射了与该触发器相连接的表的列结构,相应的 inserted 表中也包含了插入表中的数据行;当执行一条 delete 语句时,从表中删除的每一行都包含在删除触发器内的 deleted 表中。执行 update 语句时,修改之前的原始的值存入 deleted 表中,修改后的新值存入 inserted 表中。可以使用这两个临时的驻留内存的表测试某些数据修改的效果及设置触发器操作的条件,然而,不能直接对表中的数据进行更改。

根据触发器中的语句所执行的操作类型的不同,执行触发器过程中可以创建一个或者两个临时表(inserted 表和 deleted 表),表 1 说明了在进行何种操作时,触发器创建哪些表。

表 1 触发器创建的表

触发器类型	创建 inserted 表	创建 deleted 表
INSERT	是	否
UPDATE	是	是
DELETE	否	是

向表中插入数据时,INSERT 触发器触发执行。当 INSERT 触发器触发时,新的记录增加到触发器表中和 inserted 表中。inserted 表是一个逻辑表,保存了所插

入记录的拷贝,允许用户参考 INSERT 语句中数据。触发器可以检查 inserted 表,来确定该触发器的操作是否应该执行和如何执行。在 inserted 表中的记录,总是触发器表中一行或多行记录的冗余。

当触发一个 DELETE 触发器时,被删除的记录放在一个特殊的 deleted 表中。deleted 表是一个逻辑表,用来保存已经从表中删除的记录。该 deleted 表允许参考原来的 DELETE 语句删除的已经记录在日志中的数据。

修改一条记录就等于插入一条新记录同时删除一条旧记录。同样,UPDATE 语句也可以看成是由删除记录的 DELETE 语句和增加记录的 INSERT 语句组成,当在某一个有 UPDATE 触发器表的上面修改记录时,表中原来的记录移动到 deleted 表中,修改过的记录插入到了 inserted 表中。触发器可以检查 deleted 表和 inserted 表,以便确定是否修改了多个行和应该如何执行触发器的操作。

下面创建 INSERT、DELETE、UPDATE 触发器时,访问 inserted 表和 deleted 表的数据,将其插入 auditAccountData 表,包括可能的多行数据。

创建 INSERT 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_insert
ON accountData
WITH ENCRYPTION
FOR INSERT
AS
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'insert' , accountAmount FROM inserted
RETURN
```

创建 DELETE 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_delete
ON accountData
WITH ENCRYPTION
FOR DELETE
AS
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'delete' , accountAmount FROM deleted
RETURN
```

在数据更新时,将更新前的旧值表示为 update\_old\_value,将更新后的新值表示为 update\_new\_value,创建 UPDATE 触发器脚本命令如下:

```
CREATE TRIGGER t_accountData_update
ON accountData
WITH ENCRYPTION
```

```
FOR UPDATE
AS
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'update_old_value' , accountAmount FROM
deleted
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'update_new_value' , accountAmount
FROM inserted
RETURN
```

再次执行一次删除多条记录的命令,可以看到,它一次在 auditAccountData 表中记录了三条删除记录的信息,结果是正确的,达到了设计的目标,命令如下:

```
DELETE FROM accountData WHERE accountID<=3
SELECT * FROM auditAccountData
```

经过上面的改进,所有插入、更新、删除的记录都可以写入 auditAccountData 审计表,丰富了记录的内容,关键是没有发生操作数据丢失的现象。

进一步,可以通过编程控制,将 INSERT、DELETE 和 UPDATE 三个触发器合并到一起,经过判断 inserted 表和 deleted 表中记录数来区别不同的情况,程序代码如下:

```
CREATE TRIGGER t_accountData_idu
ON accountData
FOR INSERT,DELETE,UPDATE
AS
DECLARE @ins_count INTEGER
DECLARE @del_count INTEGER
SELECT @ins_count=count( *) FROM inserted
SELECT @del_count=count( *) FROM deleted
--insert row
IF @ins_count>0 AND @del_count=0
BEGIN
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'insert' , accountAmount FROM inserted
END
--delete row
IF @ins_count=0 AND @del_count>0
BEGIN
INSERT INTO auditAccountData ( audit_log_action-
Type, audit_log_amount)
SELECT 'delete' , accountAmount FROM deleted
END
--update row
```

```

IF @ins_count=@del_count AND @ins_count>0
BEGIN
INSERT INTO auditAccountData (audit_log_action-
Type, audit_log_amount)
SELECT 'update_old_value', accountAmount FROM
deleted
INSERT INTO auditAccountData (audit_log_action-
Type, audit_log_amount)
SELECT 'update_new_value', accountAmount
FROM inserted
END
RETURN

```

在上面的代码中,使用变量@ins\_count统计inserted表中的记录数,@del\_count统计deleted表中的记录数,如果@ins\_count>0并且@del\_count=0,则说明当前在accountData表上进行的是插入记录的操作;如果@ins\_count=0并且@del\_count>0,则说明当前在accountData表上进行的是删除记录的操作;如果@ins\_count=@del\_count并且@ins\_count>0,则说明当前在accountData表上进行的是更新记录的操作,由此可以通过编程将三种触发器区分开来。

### 3 结束语

文中介绍了SQL Server中触发器的工作机制和分类,结合具体实例,重点论述了DML触发器设计的一般步骤,改进了最初的设计,使触发器能够记录发生在表上的所有操作,进一步,将INSERT、DELETE和UPDATE三个触发器合并到一起,提高了系统性能。

#### 参考文献:

- [1] 张峰,张莉莉.触发器在数据处理过程中的应用研究

(上接第228页)

- [4] 李兰英,杨晨.基于S3C44BOX的智能家居终端控制系统的设计与实现[J].哈尔滨理工大学学报,2007,12(3):85-86.
- [5] 莫满春.射频路由算法的研究及智能家居无线控制系统的实现[D].广州:中山大学,2008.
- [6] 臧大进,刘增良.基于物联网的智能家居系统设计与实现[J].襄樊学院学报,2010,31(11):38-39.
- [7] Yu M C, Shin D, Shin D K, et al. Fundamentals and design of smart home middleware[C]//CSO 2009: International Joint Conference on Computational Sciences and Optimization. Washington, DC: IEEE, 2009: 647-650.
- [8] Bluetooth SIG. Specification of the Bluetooth System, Volume

[J]. 计算机工程与科学, 2008, 30(5): 156-158.

- [2] Behrend A, Dorau C, Manthey R. SQL Triggers Reacting on Time Events: An Extension Proposal[C]//Proceedings of the 13th East European Conference on Advances in Databases and Information Systems. [s. l.]: [s. n.], 2009.
- [3] 黄金敢. 基于B/S结构的教学设备管理系统研究实现[J]. 计算机技术与发展, 2010, 20(11): 170-173.
- [4] 祝红涛, 李玺. SQL Server 2008 数据库应用简明教程[M]. 北京: 清华大学出版社, 2010.
- [5] 韦晨艳, 杨键鸣, 姚斯立. SQL数据库中存储过程、触发器的应用研究[J]. 中国信息界, 2011(6): 59-60.
- [6] 陈然, 黄劭, 曾力, 等. 基于.NET的订单驱动进销存管理系统[J]. 计算机技术与发展, 2011, 21(4): 202-205.
- [7] 钟亚妹. 触发器在SQL Server数据库开发中的应用与研究[J]. 电脑知识与技术, 2011, 7(11): 2492-2494.
- [8] 杨忠, 郭俊, 李思莉. 基于ASP.NET的手机图书馆的设计与实现[J]. 计算机技术与发展, 2011, 21(1): 202-205.
- [9] 徐友武. SQL Server 2005 触发器应用研究[J]. 计算机与信息技术, 2009(9): 105-106.
- [10] Wu Dasheng, Wu Shengyu. Dynamically maintain the teaching examples of triggers and stored procedures about the course of database application[C]//2010 2nd International Conference on Education Technology and Computer (ICETC). [s. l.]: [s. n.], 2010: 525-527.
- [11] Fakas G J, Cawley B, Cai Zhi. Automated Generation of Personal Data Reports from Relational Databases[J]. Journal of Information & Knowledge Management (JIKM), 2011, 10(2): 193-208.
- [12] Kocakoç I D, Erdem S. Business Intelligence Applications in Retail Business: OLAP, Data Mining & Reporting Services[J]. Journal of Information & Knowledge Management (JIKM), 2010, 9(2): 171-181.
1. Core. Version 1.1[EB/OL]. [2005-07-15]. <http://WWW.bluetooth.org/spec/>.
- [9] Yeo L K, Weon C J. Remote-controlled Home Automation System Via Bluetooth Home Network[C]//Proc of SICE 2003 Annum Conference. Tokyo: Soc of Instrum and Control Eng, 2008: 2824-2829.
- [10] 陈桥云, 贾金玲. 基于智能手机与PC机的智能家居系统设计[J]. 电子设计工程, 2009, 17(9): 25-27.
- [11] 郭宏志. Android应用开发详解[M]. 北京: 电子工业出版社, 2010.
- [12] 殷华英, 杨红梅. 使用Java编写基于C/S模式的网络通信程序[J]. 计算机信息与技术, 2006(6): 23-24.