**Automated Trigger To Require a WHERE Clause**

**By** [**Rahul Kumar Ghosh**](http://www.sqlservercentral.com/Authors/Articles/Rahul_Kumar_Ghosh/1200971/)**, 2013/08/09 (first published: 2011/01/25)**

**FROM:** [**http://www.sqlservercentral.com/articles/delete/71468/**](http://www.sqlservercentral.com/articles/delete/71468/)

**The Problem**

I was working on an ERP project for an electrical company. As a DBA my main task was to maintain database security and data consistency. A database developer was appointed on the client side to maintain the database. By mistake the developer used an update query in an important monetary transaction table and updated all the rows instead of updating just a group of rows. This generated an incorrect payment status report from the front end ASP.NET application.

Now as a DBA, my task was to scrutinize the table, determine the error, and finally fix it. I used a SELECT query in the transaction table and saw that the payment status column data has become 'Paid' for all rows. But this was not expected; the status should be set to 'Paid' for only a few rows. The developer used the update query without a WHERE clause.

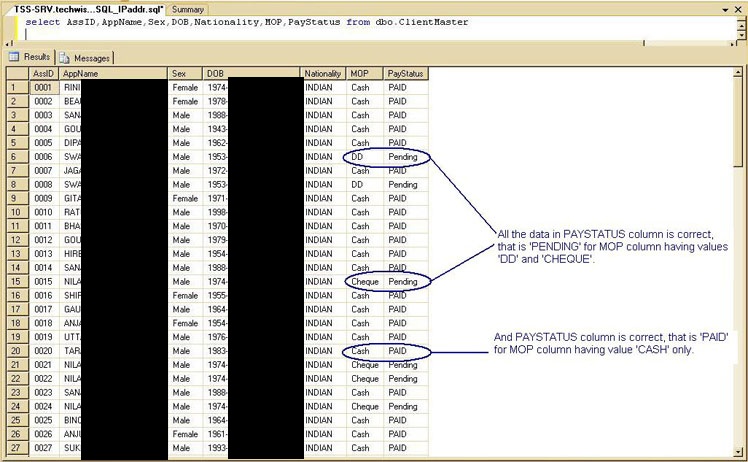
I determined the problem and its cause, and now it's time to implement some preventive methods so that this type of error will not occur in the future. By the end of this article one will be able to stop updates and deletes that are unexpectedly made on a table without WHERE conditions.

I have two figures of the transaction table below named Fig 1 and Fig 2. The two important columns are 'MOP' and 'PayStatus'. The columns are important because they are dependent on each other. In FIG 1 notice the circled items. The dependencies are explained below:

* In the MOP column where the value is 'Cash', the PayStatus column shows 'PAID' .
* In the MOP column where the value is 'DD' or 'Cheque', the PayStatus column is 'Pending'.

The transaction table should look like this after an update query.

**FIG 1.**

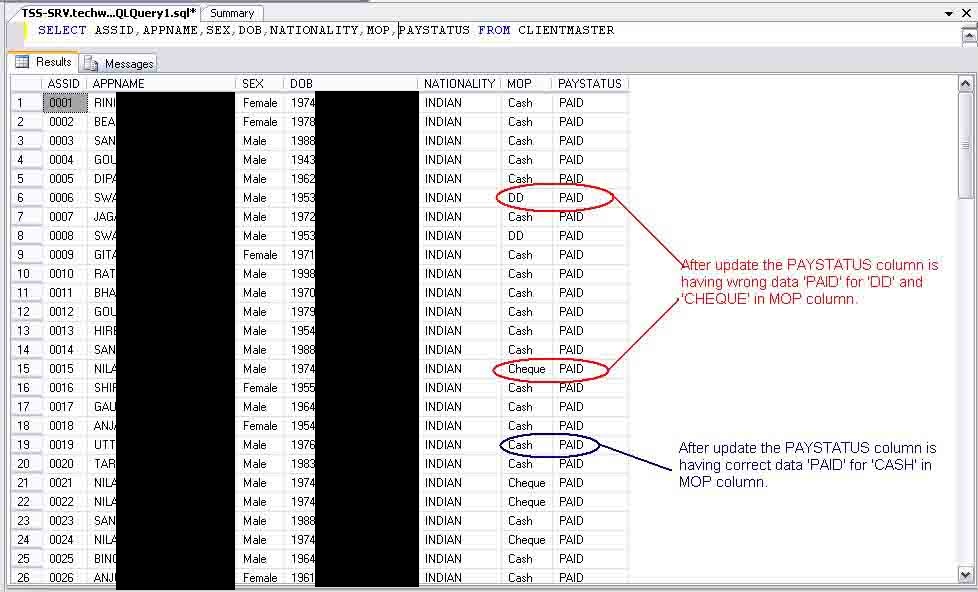


http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

The scenario changed when the database developer used an update query without a where clause. The transaction table after the developer's update is shown in FIG 2.

* The PayStatus column shown in the red circle contains the wrong data. Is is wrong for the rows with an MOP column set to 'DD' / 'Cheque', the PayStatus should be 'Pending'.
* The PayStatus column shown in the blue circle contains the correct data. For the MOP column rows with a value 'Cash', the PayStatus should be 'Paid'.

**FIG 2.**



http://www.sqlservercentral.com/Resources/Images/zoom.gif[Zoom in](javascript:;)  |  [Open in new window](javascript:;)

**The Solution**

We have to make the use of a Wherecondition mandatory for all update and delete queries. If it is not included, an error message 'Cannot update all rows. Use WHERE CONDITION' or 'Cannot delete all rows. Use WHERE CONDITION' will be shown.

**Code Description**

I have written the stored procedure to generate a trigger for a table to stop the execution of update and delete statements without a WHERE condition. The explanation of the stored procedure is shown below along with instructions on how to use it.

The parameters used in this stored procedure are:

| **Store Procedure Parameter Declaration** | | | |
| --- | --- | --- | --- |
| **Sl.No.** | **Parameter** | **Explanation** | **Example** |
| **1** | @tblname | The name of the table on which trigger needs to be created. | @tblname = 'TableName' |
| **2** | @Type | The type of trigger to be generated like update trigger or delete trigger or update and delete triggers (both). | **1)** For delete @Type = 'delete',  **2)** For update @Type = 'update',  **3)** For update and delete (both) =@Type = 'both' |

The variables used in this stored procedure are:

| **Store Procedure Variable Declaration** | | | |
| --- | --- | --- | --- |
| **Sl.No.** | **Parameter** | **Explanation** | **Example** |
| **1** | @trgname | To set up trigger name, table name is required and it's appended to the pre-fixed text '([dbo].[trg\_])'depending on the type of trigger. | 1) For delete trigger Set@trgname = [dbo].[trg\_del\_tblname],  2) For update trigger Set@trgname =[dbo].[trg\_upd\_tblname],  3) For update and delete trigger Set@trgname =[dbo].[trg\_DelUpd\_tblname] |
| **2** | @strTRGText | To hold the trigger body text. |  |
| **3** | @errupd | To hold the error message for update trigger. | Set @errupd = 'Cannotupdate all rows. Use WHERE CONDITION' |
| **4** | @errdel | To hold the error message for delete trigger. | Set @errdel = 'Cannotdelete all rows. Use WHERE CONDITION' |
| **5** | @errboth | To hold the error message for update and delete trigger (both). | Set @errboth = 'Cannot update or delete all rows. Use WHERE CONDITION' |
| **6** | @severity | It is the user-defined severity level associated with the message, note that the value of severity is fixed to 16 because severity levels from 0 to 18 can be specified by any user. Severity levels from 19 through 25 can only be specified by members of the sysadmin fixed server role and the WITH LOG option is required. Severity levels from 20 to 25 are considered fatal. Now it can be created by any user | Set @severity = '16' |
| **7** | @state | It is an integer and should be within 0 to 255. It is needed because if the same user-defined error is raised at multiple locations, using a unique state number for each location can help to find which section of code is raising the errors. | Set @state = '1' |

First the stored procedure will assign the trigger name then it will check whether that trigger exists in dbo.sysobjects or not. If yes, it will print the message 'Sorry!! [dbo].[trg\_upd\_aa] Already exists in the database...'. Otherwise it will create the trigger.

When the stored procedure when gets executed with all its needed parameters, an auto-generated trigger gets created. The only variable that has been declared in the auto-generated trigger code is @Count and it is set to @@ROWCOUNT, which returns the number of rows affected by the last statement.

The main code snippet which performs this action is this:

IF @Count >= (SELECT SUM(row\_count)

FROM sys.dm\_db\_partition\_stats

WHERE OBJECT\_ID = OBJECT\_ID('tablename')

)

This code snippet above compute the sum of the row\_count column from the sys.dm\_db\_partition\_stats table for those rows where the OBJECT\_ID column is equals the tablename parameter. The selected value will be used by the auto-generated trigger to stop update and delete statements that have no WHERE conditions.

If the selected value exceeds or is equal to @@ROWCOUNT, then the RAISERROR statement will get fired and the update or delete statement without the WHERE condition is stopped.

When the auto-generated trigger gets created, it will print a message according to the @Type:

* If the @Type = 'Update' then it will print 'Trigger done (update) Trigger [dbo].[trg\_upd\_tblname] Created Successfully'.
* If @Type = 'delete' then it will print 'Trigger done (delete) Trigger [dbo].[trg\_del\_tblname] Created Successfully'.
* If @Type = 'both' then it will print 'Trigger done (update & delete) Trigger [dbo].[trg\_upd\_tblname] created Successfully'.

The stored procedure code is in the resource section below. You can get all the trigger code when the stored procedure gets executed with the required parameters.

**Conclusion**

The accidental update and delete of data in a table can be stopped with this trigger created in the database. This will help you in ensuring data integrity, consistency, reliability and security.

**Resources:**

[SP\_Restriction.sql](http://www.sqlservercentral.com/Files/SP_Restriction.sql/8101.sql)

NOTE NOTE NOTE NOTE NOTE NOTE NOTE

SP\_Restriction.sql is shown below and as a wjb-sql-scripts\sp\_restrictiion.sql

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

/\*-- ============================================-- Author : Rahul Kr. Ghosh : MCITP-DBA 2009-- Create date : Sep 20 2010 1:34PM-- Description : STOPPING ALL ROWS UPDATE, DELETE AT ONCE-- @Type : 'Update' if to create only update trigger : 'Delete' if to create only delete trigger : 'Both' if to create both (combine) delete & update trigger-- ============================================= \*/

CREATE PROC [dbo].[SP\_Restriction]

--getting table name & type of trigger

@tblname varchar(50),

@Type varchar(20)

as

begin

--getting my tools

declare @trgname nvarchar(50)

Declare @strTRGText Varchar(max)

declare @errdel varchar(50)

declare @errupd varchar(50)

declare @errboth varchar(60)

declare @severity nvarchar(5)

declare @state nvarchar (5)

--setting my tools

Set @tblname = SubString(@tblName,CharIndex('.',@tblName)+1, Len(@tblName))

Set @strTRGText = '' ;

set @errupd = 'Cannot update all rows. Use WHERE CONDITION'; --- UPDATE TRIGGER ERROR MSG

set @errdel = 'Cannot delete all rows. Use WHERE CONDITION'; --- DELETE TRIGGER ERROR MSG

set @errboth = 'Cannot update or delete all rows. Use WHERE CONDITION'; --- UPDATE & DELETE TRIGGER ERROR MSG

set @severity = '16'

set @state = '1'

--if update trigger

if @Type = 'Update'

begin

Set @trgname = '[dbo].[trg\_upd\_'+ @tblName +']';

IF Not EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = OBJECT\_ID(@trgname))

begin

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + '/\*-- ============================================='

Set @strTRGText = @strTRGText + CHAR(13) + '-- Author : Rahul Kr. Ghosh'

Set @strTRGText = @strTRGText + CHAR(13) + ' : MCITP-DBA 2009'

Set @strTRGText = @strTRGText + CHAR(13) + '-- Create date : ' + Convert(varchar(20),Getdate())

Set @strTRGText = @strTRGText + CHAR(13) + '-- Description : STOPPING THE UPDATE OF ALL ROWS AT A STRESS IN TABLE ' + @tblName

Set @strTRGText = @strTRGText + CHAR(13) + '-- ============================================= \*/'

-- creating the update trigger code

Set @strTRGText = @strTRGText + CHAR(13) + 'CREATE TRIGGER ' + @trgname

Set @strTRGText = @strTRGText + CHAR(13) + 'ON ' + @tblname

Set @strTRGText = @strTRGText + CHAR(13) + 'FOR UPDATE AS'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'DECLARE @Count int'

Set @strTRGText = @strTRGText + CHAR(13) + 'SET @Count = @@ROWCOUNT;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'IF @Count >= (SELECT SUM(row\_count)'

Set @strTRGText = @strTRGText + CHAR(13) + 'FROM sys.dm\_db\_partition\_stats'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'WHERE OBJECT\_ID = OBJECT\_ID(''' + @tblname + ''')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RAISERROR('''+ @errupd + ''',' + @severity +',' + @state +')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'ROLLBACK TRANSACTION'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RETURN;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ''

Exec(@strTRGText);

print 'Trigger done (update)'

if (@@ERROR=0)

Print 'Trigger ' + @trgname + ' Created Successfully '

end

--trigger already there

else

begin

Print 'Sorry!! ' + @trgname + ' Already exists in the database... '

end

end

--if delete trigger

else

if @Type = 'Delete'

begin

Set @trgname = '[dbo].[trg\_del\_'+ @tblName +']';

IF Not EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = OBJECT\_ID(@trgname))

begin

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + '/\*-- ============================================='

Set @strTRGText = @strTRGText + CHAR(13) + '-- Author : Rahul Kr. Ghosh'

Set @strTRGText = @strTRGText + CHAR(13) + ' : MCITP-DBA 2009'

Set @strTRGText = @strTRGText + CHAR(13) + '-- Create date : ' + Convert(varchar(20),Getdate())

Set @strTRGText = @strTRGText + CHAR(13) + '-- Description : STOPPING THE DELETE OF ALL ROWS AT A STRESS IN TABLE ' + @tblName

Set @strTRGText = @strTRGText + CHAR(13) + '-- ============================================= \*/'

-- creating the delete trigger code

Set @strTRGText = @strTRGText + CHAR(13) + 'CREATE TRIGGER ' + @trgname

Set @strTRGText = @strTRGText + CHAR(13) + 'ON ' + @tblname

Set @strTRGText = @strTRGText + CHAR(13) + 'FOR DELETE AS'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'DECLARE @Count int'

Set @strTRGText = @strTRGText + CHAR(13) + 'SET @Count = @@ROWCOUNT;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'IF @Count >= (SELECT SUM(row\_count)'

Set @strTRGText = @strTRGText + CHAR(13) + 'FROM sys.dm\_db\_partition\_stats'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'WHERE OBJECT\_ID = OBJECT\_ID(''' + @tblname + ''')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RAISERROR('''+ @errdel + ''',' + @severity +',' + @state +')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'ROLLBACK TRANSACTION'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RETURN;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ''

Exec(@strTRGText);

print 'Trigger done (delete)'

if (@@ERROR=0)

Print 'Trigger ' + @trgname + ' Created Successfully '

end

--trigger already there

else

begin

Print 'Sorry!! ' + @trgname + ' Already exists in the database... '

end

end

--- BOTH THE TRIGGER DELETE & UPDATE

else

if @Type = 'Both'

begin

Set @trgname = '[dbo].[trg\_DelUpd\_'+ @tblName +']';

IF Not EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = OBJECT\_ID(@trgname))

begin

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + '/\*-- ============================================='

Set @strTRGText = @strTRGText + CHAR(13) + '-- Author : Rahul Kr. Ghosh'

Set @strTRGText = @strTRGText + CHAR(13) + ' : MCITP-DBA 2009'

Set @strTRGText = @strTRGText + CHAR(13) + '-- Create date : ' + Convert(varchar(20),Getdate())

Set @strTRGText = @strTRGText + CHAR(13) + '-- Description : STOPPING THE UPDATE AND DELETE OF ALL ROWS AT A STRESS IN TABLE ' + @tblName

Set @strTRGText = @strTRGText + CHAR(13) + '-- ============================================= \*/'

-- creating the delete & UPDATE trigger code

Set @strTRGText = @strTRGText + CHAR(13) + 'CREATE TRIGGER ' + @trgname

Set @strTRGText = @strTRGText + CHAR(13) + 'ON ' + @tblname

Set @strTRGText = @strTRGText + CHAR(13) + 'FOR UPDATE , DELETE AS'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'DECLARE @Count int'

Set @strTRGText = @strTRGText + CHAR(13) + 'SET @Count = @@ROWCOUNT;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'IF @Count >= (SELECT SUM(row\_count)'

Set @strTRGText = @strTRGText + CHAR(13) + 'FROM sys.dm\_db\_partition\_stats'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'WHERE OBJECT\_ID = OBJECT\_ID(''' + @tblname + ''')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'BEGIN'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RAISERROR('''+ @errboth + ''',' + @severity +',' + @state +')'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'ROLLBACK TRANSACTION'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'RETURN;'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + 'END'

Set @strTRGText = @strTRGText + CHAR(13) + ''

Set @strTRGText = @strTRGText + CHAR(13) + ''

Exec(@strTRGText);

print 'Trigger done (update & delete)'

if (@@ERROR=0)

Print 'Trigger ' + @trgname + ' Created Successfully '

end

--trigger already there

else

begin

Print 'Sorry!! ' + @trgname + ' Already exists in the database... '

end

end

end