stackoverflow.com/questions/15012886/how-to-rollback-or-commit-a-transaction-in-sql-server

Code Magician 17.5k 5 46 73

54

In my stored procedure, I have three insert statements.

On duplicate key value insertion first two queries generate the error

> *Violation of PRIMARY KEY constraint*

and third query runs as usual.

Now I want that if any query generates any exception, everything should get rolled back.

If there isn't any exception generate by any query, it should get committed.

```
declare @QuantitySelected as char
    set @QuantitySelected = 2

    declare @sqlHeader as varchar(1000)
    declare @sqlTotals as varchar(1000)
    declare @sqlLine as varchar(1000)

    select @sqlHeader = 'Insert into tblKP_EstimateHeader '
    select @sqlHeader = @sqlHeader +
'(CompanyID,CompanyName,ProjectName,EstimateID,EstimateHeader,QuoteDate,ValidUntil,RFQNum,Revision,Contact,Status,NumConfigurations) '
    select @sqlHeader = @sqlHeader + ' select
CompanyID,CompanyName,ProjectName,EstimateID,EstimateHeader,QuoteDate,ValidUntil,RFQNum,Revision,Contact,Status,NumConfigurations '
    select @sqlHeader = @sqlHeader +  'from V_EW_Estimate_Header where EstimateID = 2203'



    select @sqlTotals = 'Insert into tblKP_Estimate_Configuration_Totals '
    select @sqlTotals = @sqlTotals + '(ConfigRecId,RecId,SellQty,ConfigNum,ConfigDesc,SortOrder,OptionsInMainPrice,MarkupPctQty,'
    select @sqlTotals = @sqlTotals + '
SellPriceQty,RubberStamp,OptPriceQty,StatusRecid,LastUpdate_Date,LastUpdate_User,TotalCost,QuantityBracketSelected)'
    select @sqlTotals = @sqlTotals + ' select ConfigRecId,RecId,SellQty' + @QuantitySelected +
',ConfigNum,ConfigDesc,SortOrder,OptionsInMainPrice'
    select @sqlTotals = @sqlTotals + ' ,MarkupPctQty' + @QuantitySelected + ',SellPriceQty' + @QuantitySelected + ',RubberStamp,OptPriceQty'
+ @QuantitySelected + ',StatusRecid,LastUpdate_Date,LastUpdate_User,TotalCost' + @QuantitySelected + ',' + @QuantitySelected
    select @sqlTotals = @sqlTotals + ' from v_EW_Estimate_Configuration_Totals where ConfigRecId = -3'


    select @sqlLine = 'Insert into tblKP_Estimate_Configuration_Lines'
    select @sqlLine = @sqlLine + '(MstrRfqRecId,RfqRecId,RfqLineRecId,CompanyId,VendorQuoteNum,LineGrp,LineNum,StatusRecId,'
    select @sqlLine = @sqlLine + ' LineDesc,LineSize,LineMatl,LineDeco,LineFinish,CopyFromRecId,PerPieceCost,IsOptional,'
    select @sqlLine = @sqlLine + '
CopyToNewRev,RecId,UnitPrice,LineQty,LinePrice,CustOrVend,SellQty1,RfqNum,ConfigLineIsOptional,ConfigLinePerPieceCost,ConfigLineRecid,SellPrice,SaleQty)'

    select @sqlLine = @sqlLine + ' select distinct MstrRfqRecId,RfqRecId,RfqLineRecId,CompanyId,VendorQuoteNum,LineGrp,LineNum,'
    select @sqlLine = @sqlLine + ' StatusRecId,LineDesc,LineSize,LineMatl,LineDeco,LineFinish,CopyFromRecId,PerPieceCost,IsOptional,'
    select @sqlLine = @sqlLine + ' CopyToNewRev,RecId,UnitPrice' + @QuantitySelected + ',LineQty' + @QuantitySelected + ', isnull(LinePrice'
+ @QuantitySelected + ', 0.0000),CustOrVend,SellQty' + @QuantitySelected +
',RfqNum,ConfigLineIsOptional,ConfigLinePerPieceCost,ConfigLineRecid,SellPrice' + @QuantitySelected + ',SaleQty' + @QuantitySelected
    select @sqlLine = @sqlLine + ' from v_EW_EstimateLine  where rfqlinerecid in (select RfqLineRecID from kp_tblVendorRfqConfigLine where
ConfigRecID = -3) '

    exec( @sqlHeader)
    exec(@sqlTotals)
    exec(@sqlLine)
```

158

✔

The good news is a transaction in SQL Server can span multiple batches (each `exec` is treated as a separate batch.)

You can wrap your `EXEC` statements in a `BEGIN TRANSACTION` and `COMMIT` but you'll need to go a step further and rollback if any errors occur.

Ideally you'd want something like this:

```
BEGIN TRY
    BEGIN TRANSACTION
        exec( @sqlHeader)
        exec(@sqlTotals)
        exec(@sqlLine)
    COMMIT
END TRY
BEGIN CATCH

    IF @@TRANCOUNT > 0
        ROLLBACK
END CATCH
```

The `BEGIN TRANSACTION` and `COMMIT` I believe you are already familiar with. The `BEGIN TRY` and `BEGIN CATCH` blocks are basically there to catch and handle any errors that occur. If any of your `EXEC` statements raise an error, the code execution will jump to the `CATCH` block.

Your existing SQL building code should be outside the transaction (above) as you always want to keep your transactions as short as possible.

Your Answer

Post as a guest

Not the answer you're looking for? Browse other questions tagged sql sql-server sql-server-2008 sql-server-2005 or ask your own question.