Dynamic Query and Incorrect Syntax Near 'GO' Error

pAtuAry

Rate me! ★★★★★ 4.80 (3 votes)

24 Jan 2013 CPOL

Here, I have tried to figure out some basic concept about dynamic query.

Introduction

Sometimes we are experiencing a common Microsoft SQL Server error: Incorrect syntax near 'GO'. What is this? And in which scenario do we get this error? One most common scenario is when we write a dynamic query. Here, I am trying to discuss some basic concepts about dynamic queries.

What is Dynamic Query?

1. Dynamic query is a query which evaluates at runtime.
2. Dynamic query is used to write inside double quote ("...") as a `string`.
3. `EXEC` command or `sp_executesql` statement is used to execute dynamic query.
4. We cannot use 'GO' statement inside a dynamic query. If we try, we will get "`Incorrect syntax near 'GO'`" exception.

What Does 'GO' Statement Do?

1. 'GO' statement forces SQL Server to execute code batch or code group immediately.
2. 'GO' signals the end of batch or code group.
3. Each code batch separated by 'GO' is compiled like one execution plan.
4. Any error in one batch will not affect the next batch.

For example: Consider the following two lines. In the first statement, I am trying to convert a character into an integer. That's why we get exception. As the first statement gets exception, that's why the second statement will also not work.

Hide Copy Code

```
exec('select convert(int,''b'')')
exec ('select getdate()')
```

Now, consider the following two lines. Though the first statement will throw an exception, the second statement still executes as the second statement is differented in different batch by GO statement.

Hide Copy Code

```
exec('select convert(int,''b'')')
go
exec ('select getdate()')
go
```

5. Any local variable used in one batch is not accessible by another batch.

Try the following example:

Hide Copy Code

```
DECLARE @MyMsg VARCHAR(50)
SELECT @MyMsg = 'Hello, World.'
GO -- @MyMsg is not valid after this GO ends the batch.
-- Yields an error because @MyMsg is not declared in this batch.
PRINT @MyMsg
GO
```

Dynamic Query Does Not Support 'GO'

Dynamic SQL executer does not support multiple batch. That means dynamic SQL query is treated as a single batch. You cannot differentiate query into different batch by 'GO' statement.

Another reason for not supporting 'GO' statement by SQL parser of dynamic query is that 'GO' is not a TSQL statement 😉, so funny right? It is only recognized by SQLCMD and OSQL utilities and SQL Server Management Code Editor. I do not know how many other statements we are using that are not actually TSQL...

What is the Main Problem of Dynamic SQL?

The main problem is that you cannot guess any syntax error of a dynamic query at compile time.

For debugging, I use `Print` command instead of `Exec` command and copy the printed query from result pane and execute separately to see where the problem is.

When Should You Use A Dynamic Query?

When the query is not `static`. That's mean when any part of your query will be evaluated at runtime or when query parameter does not have fixed value. That means, Parameter value will be changed at runtime.

Let's consider a simple example.

Let's do a task for each user created database in our SQL Server instance.

The pseudo-code for this task is:

- Step 1: Loop each user created database
- Step 2: Check if the database is online
- Step 3: If database is online
- Step 4: Do specific task for that database
- Step 5: End loop

The solution code is as below:

Hide Shrink ▲ Copy Code

```
DECLARE @EndLine nchar(2); SET @EndLine = NCHAR(13) + NCHAR(10);
DECLARE @sql nvarchar(4000);
DECLARE @BigSQL nvarchar(4000);
DECLARE @dbName varchar(100);

DECLARE MY_CURSOR Cursor --Loop through all user created database

FOR
SELECT NAME FROM sys.databases
WHERE name NOT IN ('master', 'tempdb', 'model', 'msdb') AND state = 0
--here state = 0 means online
--'master', 'tempdb', 'model', 'msdb' - these 4 databases are called system database
--all user defined databases are stored in there and they are also responsible
--for some other specific task like, temporary tables used in sql query
--are stored in 'tempdb' database

Open My_Cursor

Fetch NEXT FROM MY_Cursor INTO @dbName
While (@@FETCH_STATUS = 0)
BEGIN

    -------------->>>>>>>>>>
    --Create a table to each database
    --Task#1: at first check is the table is exist
    SET @sql =
        N'IF OBJECT_ID(''''dbo.A_TmpTable'''', ''''U'''') is not null' + @EndLine +
        N'    DROP TABLE dbo.A_TmpTable'
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC sp_executesql N''' + @sql + '''';
    EXEC (@BigSQL)

    SET @sql = N' Create Table A_TmpTable' + @EndLine
    SET @sql += N'(' + @EndLine
    SET @sql += N'[DB_Name] Varchar(100),' + @EndLine
    SET @sql += N'[Current_TIme] datetime' + @EndLine
    SET @sql += N')'
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC sp_executesql N''' + @sql + '''';
    EXEC (@BigSQL)

    --Then insert some data to the table.
    SET @sql=N' Insert Into A_TmpTable([DB_Name], [Current_TIme])
            values( DB_Name(), getdate() );'
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC sp_executesql N''' + @sql + '''';
    EXEC (@BigSQL)
    --------------<<<<<<<<<<


    -------------->>>>>>>>>>
    --Create a stored procedure
    --The task of procedure is to display table data that is DB_Name and Current_Time
    --at first check if the procedure exists
    SET @sql =
        N'IF OBJECT_ID(''''dbo.A_TmpProcedure'''') is not null' + @EndLine +
        N'    DROP PROCEDURE dbo.A_TmpProcedure'
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC sp_executesql N''' + @sql + '''';
    EXEC (@BigSQL)

    --Then create the procedure
    SET @sql =
        N'CREATE PROCEDURE dbo.A_TmpProcedure' + @EndLine +
        N'AS' + @EndLine +
        N'BEGIN' + @EndLine +
        N' Select * From A_TmpTable;' + @EndLine +
        N'END'
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC sp_executesql N''' + @sql + '''';
    EXEC (@BigSQL)

    --finally run the stored procedure
    SET @BigSQL = 'USE [' + @dbName + ']; EXEC dbo.A_TmpProcedure';
    EXEC (@BigSQL)
    --------------<<<<<<<<<<
    FETCH NEXT FROM MY_CURSOR INTO @dbName
END
CLOSE MY_CURSOR
DEALLOCATE MY_CURSOR
GO
```

In the above code, an important thing is database name. As I do not want to write hard-coded query for each database, I use cursor. Cursor in MSSQL database acts as a loop. Loop on table record. I pick all currently online database names from the `sys.database` table and loop through on each database.

Now for each database, I check whether the specific table exists or not. Here the database name is a variable which is fetched by cursor.

Then I create table, then insert a row in that table, then create a stored procedure that will show the data of newly created table. And all these things are done for each database.

Point of Interest

As a database developer, sometimes we need to write dynamic query like, change status or change recovery model or backup transaction log of databases. Also, sometimes you need to write dynamic query in stored procedure in which the value of a variable which is evaluated at runtime in "`where`" clause. Here, I have tried to discuss some basic things about dynamic query.

Here, some terms are used but not discussed briefly as they are beyond the scope of this topic. like SQLCMD, OSQL, CURSOR. Some concepts like table construction, procedure construction are also not discussed here.

In the End

Today morning, my boss asked me for writing a dynamic query that was very cool stuff. While writing that query, I have struggled much and thought of sharing my knowledge with you. Thank you very much. Any suggestions from you are highly appreciated.

License

Share

About the Author

pAtuAry
Software Developer
Bangladesh

Follow this Member

Nothing more...
.

Comments and Discussions

Add a Comment or Question Email Alerts Search Comments

First Prev Next

My vote of 3
ghlewis 10-Oct-14 21:41

Re: My vote of 3
pAtuAry 10-Oct-14 22:08

Refresh 1

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.