Exploring Dynamic SQL

Madhivanan

SQL Server MVP

- A SQL whose statements are known only during runtime
- EXEC or EXECUTE is used to execute such queries

- Static SQL

  Select * from customers

- Dynamic SQL

  EXEC(@sql)

- Examples

```
Declare @sql varchar(100)
Set @sql='select * from customers'
Exec(@sql)
```

```
Declare @sql varchar(1000)
Declare @table_name varchar(100)
Select @sql='', @table_name='customer'
Set @sql='select * from '+@table_name
Exec(@sql)
```

When should I use Dynamic SQL?

- When an object name is passed as a parameter

Example

Select @column_name from table
Select * from @table_name
Select * from table @where

- You can execute Dynamic SQL using EXEC and sp_executesql

- EXEC (‘select 1’)
- exec sp_executesql N'select 1'

- The parameter should be nvarchar datatype

exec sp_executesql 'select 1‘ throws an error

Msg 214, Level 16, State 2, Procedure sp_executesql, Line 1

Procedure expects parameter '@statement' of type 'ntext/nchar/nvarchar'.

- Understanding single quotes in dynamic sql

- Static SQL

  select ''

- Dynamic sql

  exec('select ''''') -- totally 6 single quotes

- Result is empty string

- Static SQL

  select ‘’’'

- Dynamic sql

  exec('select '''''''') -- totally 10 single quotes

  Result is ‘ (single quote)

- Static SQL

  select ‘’’‘’’

- Dynamic sql

  exec('select '‘’’’’''''''') – totally 14 single quotes

  Result is ‘’ (Two single quotes)

- Formula 6+4*n (where n denotes number of single quotes as a output)

- When you use a static sql and express a value in a single quote then first and last sigle quotes specify that the value is a string. Then within those single quotes every double single quotes represent a single single quote

- When you use a Dynamic sql then first and last sigle quotes specify that it is a dynamic sql. Then within those single quotes first and last double single quotes specify that it is a string.Then within those single quotes every four single quotes represent a single single quote

Dynamic SQL and table access

- A dynamic SQL written in a stored procedure requires direct Read permissions on the tables used rather than just EXECUTE permission over the stored procedures

Variable scopes in dynamic sql

```
Declare @i int
Set @i=1
Exec('select @i')
```

Throws an error

```
Msg 137, Level 15, State 2, Line 1
Must declare the scalar variable "@i".
```

```
Declare @i int
Set @i=1
declare @sql varchar(100)
set @sql='select '+cast(@i as char(10))
exec(@sql)
```

Result is 1

```
exec('

declare @i int
set @i=1
select @i

')
```

```
exec('

declare @t table(id int, names varchar(100))
insert into @t(id, names)
select 1,''''''test''''''
select * from @t
')
```

Temporary tables and Dynamic SQL

You can access temporary tables in Dynamic SQL

```
create table #t (i int)
insert into #t(i)
select 1

exec
(
‘select i from #t'
)
Returns 1
```

A temporary table created inside Dynamic SQL will be in a different scope

```
create table #t (i int)
insert into #t(i)
select 1
exec
(
‘create table #t(i int)
select i from #t'
)
```

Returns empty result

```
create table #t (i int)
insert into #t(i)
select 1

exec
(
'create table #t(i int)
select i from #t'
)

select i from #t
```

Returns two resultsets (empty and 1)

Session scope
create table #t (i int)
insert into #t(i)
select 1

Execution scope
exec
(
‘create table #t(i int)
select i from #t'
)

Session level Table variables can not be accessed inside dynamic SQL

```
declare @t table (i int)
insert into @t(i)
select 1
exec
(
'select i from @t'
)
Msg 1087, Level 15, State 2, Line 1
Must declare the table variable "@t".
```

EXEC Vs SP_executesql

- EXEC is not parameterised

EXEC(‘select 1’)

- SP_executesql is parametrised

EXEC sp_executesql N'select 1'

- EXEC is subject to SQL Injection
- With proper parameterization, you can avoid SQL Injection in sp_executesql

Dynamic SQL and SQL Injection

A method that will gain access to your objects and execute any queries

```
create table product_master
(
product_name varchar(100)
)

insert into product_master (product_name)
select 'Television' union all
select 'Mobile phone' union all
select 'landline phone' union all
select 'computer'
```

```
Create procedure get_products
(
@product_name varchar(100)
)
as
declare @sql varchar(8000)
set @sql='select product_name from product_master
    where product_name like '''+@product_name+'%'''
exec(@sql)
GO
```

- exec get_products 'mobile'

Result is

Mobile phone

Actual query is

select product_name from product_master where product_name like 'mobile%'

exec get_products 'mobile'' or 1=1; --'

Result is

Television

Mobile phone

landline phone

Computer

Autual query is

select product_name from product_master where product_name like 'mobile' or 1=1; --%

(and the value becomes two different conditions)

- exec get_products 'mobile'' select * from information_schema.tables; --'

If your application handles multiple resultsets, the results are

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
test	dbo	product_master	BASE TABLE
test	dbo	test	BASE TABLE
test	dbo	testing	BASE TABLE

exec get_products 'mobile'' truncate table product_master; --'

Actual query is

select product_name from product_master where product_name like 'mobile' truncate table product_master; --%

The table product_master will be truncated

exec get_products 'mobile'' drop table product_master; --‘

Actual query is

select product_name from product_master where product_name like 'mobile' drop table product_master; --%

The table product_master will be truncated

How to avoid SQL Injection?

If you use EXEC, use derived table

```
alter procedure get_products
(
@product_name varchar(100)
)
as
declare @sql varchar(8000)
set @sql='select * from (select product_name from product_master
    where product_name like '''+@product_name+'%'') as t‘
exec(@sql)
GO
```

exec get_products 'mobile'

Result is

Mobile phone

Actual query is

```
select * from (select product_name from product_master
    where product_name like 'mobile%') as t
```

exec get_products 'mobile'' or 1=1; --'

There is an error
Msg 102, Level 15, State 1, Line 1
Incorrect syntax near ';'.

Autual query is

```
select * from (select product_name from
    product_master where product_name like
    'mobile' or 1=1; --%') as t
```

```
Use sp_executesql

alter procedure get_products
(
@product_name varchar(100)
)
as
declare @sql nvarchar(4000)
set @product_name =@product_name +'%'
set @sql='select product_name from product_master where product_name
    like @product_name‘
exec sp_executesql @sql,N'@product_name varchar(100)',@product_name
GO
```

- exec get_products 'mobile'

Result is

Mobile phone

Actual query is

select product_name from product_master where product_name like 'mobile%'

- exec get_products 'mobile'' or 1=1; --'

The result is empty because there is no product named mobile’ or 1=1; --

Actual query is

select product_name from product_master where product_name like 'mobile'' or 1=1; --‘

(and the value does not become two conditions)

Always use static SQL when object names are not passed as parameter

```
alter procedure get_products
(
@product_name varchar(100)
)
as
select product_name from product_master where product_name like @product_name+'%'
```

- exec get_products 'mobile'

Result is

Mobile phone

Actual query is

select product_name from product_master where product_name like 'mobile%'

exec get_products 'mobile'' or 1=1; --'

The result is empty because there is no product named mobile’ or 1=1; --

Actual query is

select product_name from product_master where product_name like 'mobile'' or 1=1; --'

- What can you execute in Static SQL that can not be executed in Dynamic SQL?

- What can you execute in Static SQL that can not be executed in Dynamic SQL?

- Batch seperator

```
Select 1 as number
GO
Select 2 as number
```

Two result sets

1

And

2

```
Exec
('
Select 1 as number
GO

Select 2 as number
'
)
```

Result is

```
Msg 102, Level 15, State 1, Line 3
Incorrect syntax near 'GO'.
```

- What can you execute in Dynamic SQL that can not be executed in Static SQL?

- What can you execute in Dynamic SQL that can not be executed in Static SQL?

- Conditional Object creation/alteration

```
if exists(select * from INFORMATION_SCHEMA.ROUTINES where
    ROUTINE_NAME='testing')
alter procedure testing
(
@i int
)
as
select @i as number
else
create procedure testing
(
@j int
)
as
select @j as number
```

Msg 156, Level 15, State 1, Line 5

Incorrect syntax near the keyword 'procedure'.

Msg 137, Level 15, State 2, Line 10

Must declare the scalar variable "@i".

Msg 111, Level 15, State 1, Line 17

'CREATE/ALTER PROCEDURE' must be the first statement in a query batch.

```
if exists(select * from INFORMATION_SCHEMA.ROUTINES where
    ROUTINE_NAME='testing')
exec(
'alter procedure testing
(@i int)
as
select @i as number
')
else
Exec('create procedure testing
(@j int)
as
select @j as number')
```

Dbcc commands

Insert into table(…)
DBCC useroptions

Insert into table(…)
EXEC('DBCC useroptions')

- Use Database will not change Database for the current session

Exec (‘use test’)

Select * from customers

- The object will be checked not in test but in the database of the current session

Thank you

Contact me
Email madhivanan2001@gmail.com
Blog :
http://beyondrelational.com/blogs/madhivanan
Facebook :
https://www.facebook.com/madhivanan2001

Twitter : madhivanan2001