T-SQL: Dynamic SQL

\(^o^)/

September, 2019

Section 1

Introduction

Concept

- T-SQL code that constructs & executes T-SQL code
- Executed in a separate batch from the calling batch

Tools to Execute Stored Code

- {EXEC | EXECUTE} command
- sp_executesql procedure
  - Interface for parameters passing

Section 2

EXEC Command

An Example Using the EXEC Command

```
DECLARE @s AS NVARCHAR(200);
SET @s = N'Davis'; -- originates in user input

DECLARE @sql AS NVARCHAR(1000);
SET @sql = N'SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N''' + @s + N''';';

PRINT @sql; -- for debug purposes
EXEC (@sql);
```

Example (cont.)

Produced SQL statement:

```
SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N'Davis';
```

Potential Problems

- Performance issues
- Security issues
  User input directly concatenated into the code
  ==> vulnerable to SQL injection

SQL Injection Example

Probing schemas and tables in the database:

```
DECLARE @s AS NVARCHAR(200);
SET @s = N'abc'' UNION ALL
SELECT object_id, SCHEMA_NAME(schema_id), name, NULL
FROM sys.objects WHERE type IN (''U'', ''V'');--';

DECLARE @sql AS NVARCHAR(1000);
SET @sql = N'SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N''' + @s + N''';';

PRINT @sql; -- for debug purposes
EXEC (@sql);
```

SQL Injection Example (cont.)

Produced SQL statement:

```
SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N'abc' UNION ALL
SELECT object_id, SCHEMA_NAME(schema_id), name, NULL
FROM sys.objects WHERE type IN ('U', 'V'); --';
```

SQL Injection Example (cont.)

empid	firstname	lastname	hiredate
901578250	HR	Employees	NULL
965578478	Production	Suppliers	NULL
997578592	Production	Categories	NULL
1029578706	Production	Products	NULL
1141579105	Sales	Customers	NULL
1173579219	Sales	Shippers	NULL
1205579333	Sales	Orders	NULL
1301579675	Sales	OrderDetails	NULL
1461580245	Stats	Tests	NULL
1493580359	Stats	Scores	NULL

Figure 1: Query results show schemas and tables

SQL Injection Example (cont.)

Further probing on the Customers table:

```
DECLARE @s AS NVARCHAR(200);
SET @s = N'abc'' UNION ALL
SELECT NULL, name, NULL, NULL
FROM sys.columns WHERE object_id = 1141579105; --';

DECLARE @sql AS NVARCHAR(1000);
SET @sql = N'SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N''' + @s + N''';';

PRINT @sql; -- for debug purposes
EXEC (@sql);
```

SQL Injection Example (cont.)

- Produced SQL statement:

```
SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N'abc' UNION ALL
SELECT NULL, name, NULL, NULL
FROM sys.columns WHERE object_id = 1141579105; --';
```

SQL Injection Example (cont.)

empid	firstname	lastname	hiredate
NULL	custid	NULL	NULL
NULL	companyname	NULL	NULL
NULL	contactname	NULL	NULL
NULL	contacttitle	NULL	NULL
NULL	address	NULL	NULL
NULL	city	NULL	NULL
NULL	region	NULL	NULL
NULL	postalcode	NULL	NULL
NULL	country	NULL	NULL
NULL	phone	NULL	NULL
NULL	fax	NULL	NULL

Figure 2: Query results show Customers tables' structure

SQL Injection Example (cont.)

Collecting phone numbers from the customers:

```
DECLARE @s AS NVARCHAR(200);
SET @s = N'abc'' UNION ALL
SELECT NULL, companyname, phone, NULL
FROM Sales.Customers; --';

DECLARE @sql AS NVARCHAR(1000);
SET @sql = N'SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N''' + @s + N''';';

PRINT @sql; -- for debug purposes
EXEC (@sql);
```

SQL Injection Example (cont.)

Produced SQL statement:

```
SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = N'abc' UNION ALL
SELECT NULL, companyname, phone, NULL
FROM Sales.Customers; --';
```

SQL Injection Example (cont.)

empid	firstname	lastname	hiredate
NULL	Customer NRZBB	030-3456789	NULL
NULL	Customer MLTDN	(5) 789-0123	NULL
NULL	Customer KBUDE	(5) 123-4567	NULL
NULL	Customer HFBZG	(171) 456-7890	NULL
NULL	Customer HGVLZ	0921-67 89 01	NULL
NULL	Customer XHXJV	0621-67890	NULL
NULL	Customer QXVLA	67.89.01.23	NULL
NULL	Customer QUHWH	(91) 345 67 89	NULL
NULL	Customer RTXGC	23.45.67.89	NULL
NULL	Customer EEALV	(604) 901-2345	NULL
NULL	Customer UBHAU	(171) 789-0123	NULL
NULL	Customer PSNMQ	(1) 890-1234	NULL

Figure 3: Query results show customers' phone numbers

Section 3

sp_executesql Procedure

Inputs of sp_executesql

- @stmt: input batch of code
- @params: parameters declaration
- parameter values

An Example Using sp_executesql

```
DECLARE @s AS NVARCHAR(200);
SET @s = N'Davis';

DECLARE @sql AS NVARCHAR(1000);
SET @sql = 'SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = @lastname;';

PRINT @sql; -- For debug purposes

EXEC sp_executesql
  @stmt = @sql,
  @params = N'@lastname AS NVARCHAR(200)',
  @lastname = @s;
```

Example (cont.)

Produced SQL statement:

```
SELECT empid, firstname, lastname, hiredate
FROM HR.Employees WHERE lastname = @lastname;
```

sp_executesql (cont.)

- Query optimized on first use & plan cached => eliminate performance issue
- User input always considered as value in the parameter => eliminate security issue

Section 4

Dynamic Search Conditions

Introduction

- Also called dynamic filtering
- Application provides an interface with various attributes
- Users choose the attribute to filter with each request
- Example: getOrders stored procedure
  - Optional inputs: @orderid, @custid, @empid, and @orderdate
  - Query and filter on non-NULL input values

Implementation with Static SQL

```
CREATE PROC dbo.GetOrders
  @orderid   AS INT  = NULL,
  @custid    AS INT  = NULL,
  @empid     AS INT  = NULL,
  @orderdate AS DATE = NULL
AS

SELECT orderid, custid, empid, orderdate, filler
FROM dbo.Orders
WHERE (orderid   = @orderid   OR @orderid   IS NULL)
  AND (custid    = @custid    OR @custid    IS NULL)
  AND (empid     = @empid     OR @empid     IS NULL)
  AND (orderdate = @orderdate OR @orderdate IS NULL);
```

Test Query

```
EXEC dbo.GetOrders @orderdate = '20140101';
```

Execution Plan

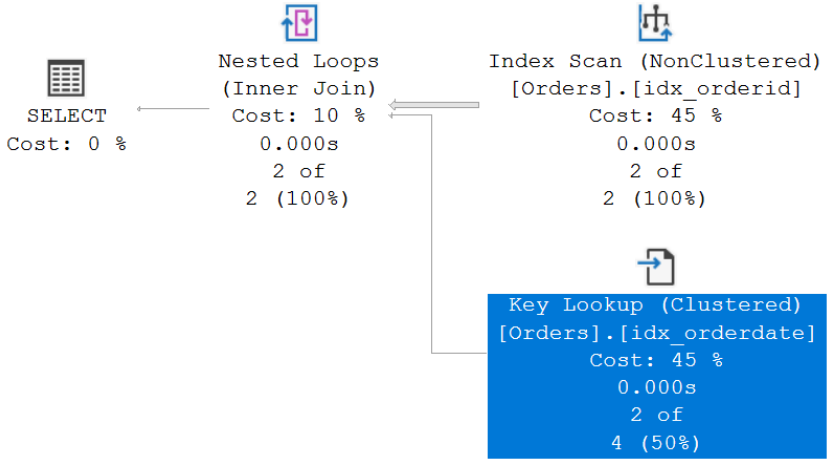

Figure 4: Plan with all predicates processed

Discussion

- All predicates processed => Inefficient query plan

Implementation with Static SQL and Recompile Option

```
CREATE PROC dbo.GetOrders
  @orderid   AS INT  = NULL,
  @custid    AS INT  = NULL,
  @empid     AS INT  = NULL,
  @orderdate AS DATE = NULL
AS
SELECT orderid, custid, empid, orderdate, filler
FROM dbo.Orders
WHERE (orderid   = @orderid   OR @orderid   IS NULL)
  AND (custid    = @custid    OR @custid    IS NULL)
  AND (empid     = @empid     OR @empid     IS NULL)
  AND (orderdate = @orderdate OR @orderdate IS NULL)
OPTION (RECOMPILE);
```

Test Queries

```
EXEC dbo.GetOrders @orderdate = '20140101';
EXEC dbo.GetOrders @orderid   = 10248;
```

Execution Plans

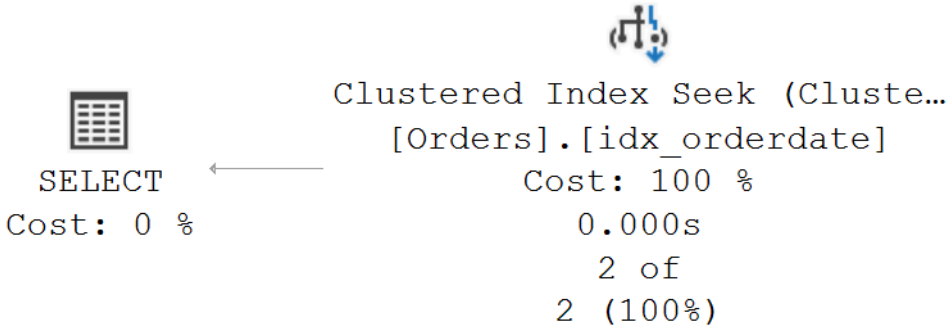

Figure 5: Plan with recompile filtering by orderdate

Execution Plans (cont.)

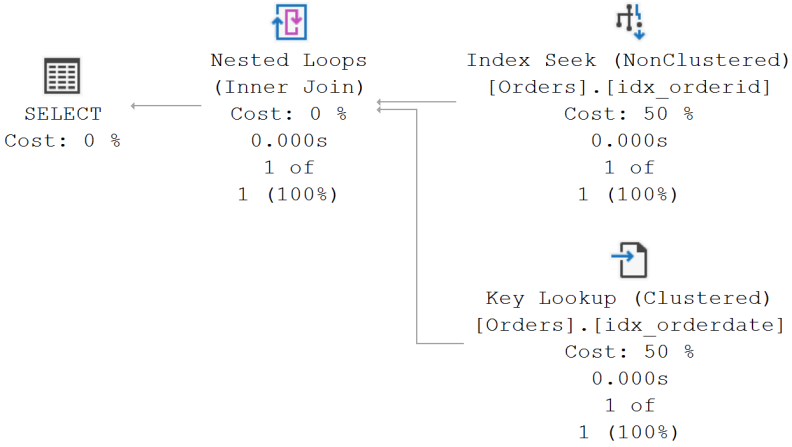

Figure 6: Plan with recompile filtering by orderid

Discussion

- Optimal plans
- Inefficient plan reuse
  Could be refined further but hard to maintain

Implementation with Dynamic SQL

```
CREATE PROC dbo.GetOrders
  @orderid   AS INT  = NULL,
  @custid    AS INT  = NULL,
  @empid     AS INT  = NULL,
  @orderdate AS DATE = NULL
AS
DECLARE @sql AS NVARCHAR(1000);
SET @sql =
    N'SELECT orderid, custid, empid, orderdate, filler'
  + N' /* 27702431-107C-478C-8157-6DFCECC148DD */'
  + N' FROM dbo.Orders'
  + N' WHERE 1 = 1'
  + CASE WHEN @orderid IS NOT NULL THEN
      N' AND orderid = @oid' ELSE N'' END
```

Implementation with Dynamic SQL (cont.)

```
  + CASE WHEN @custid IS NOT NULL THEN
      N' AND custid = @cid' ELSE N'' END
  + CASE WHEN @empid IS NOT NULL THEN
      N' AND empid = @eid' ELSE N'' END
  + CASE WHEN @orderdate IS NOT NULL THEN
      N' AND orderdate = @dt' ELSE N'' END;
EXEC sp_executesql
  @stmt = @sql,
  @params = N'@oid AS INT, @cid AS INT, @eid AS INT, @dt AS DATE',
  @oid = @orderid,
  @cid = @custid,
  @eid = @empid,
  @dt  = @orderdate;
```

Test Queries

```
EXEC dbo.GetOrders @orderdate = '20140101';
EXEC dbo.GetOrders @orderdate = '20140102';
EXEC dbo.GetOrders @orderid   = 10248;
```

Execution Plans

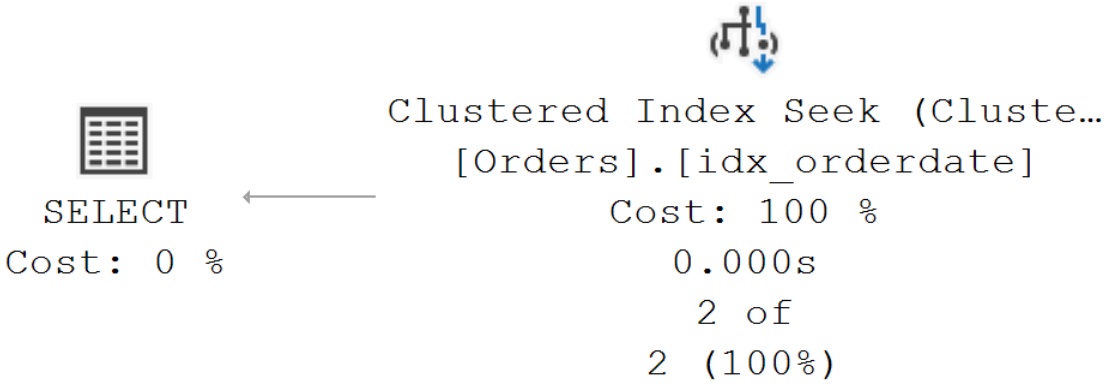

Figure 7: Plan for a dynamic query filtering by orderdate

Execution Plans (cont.)

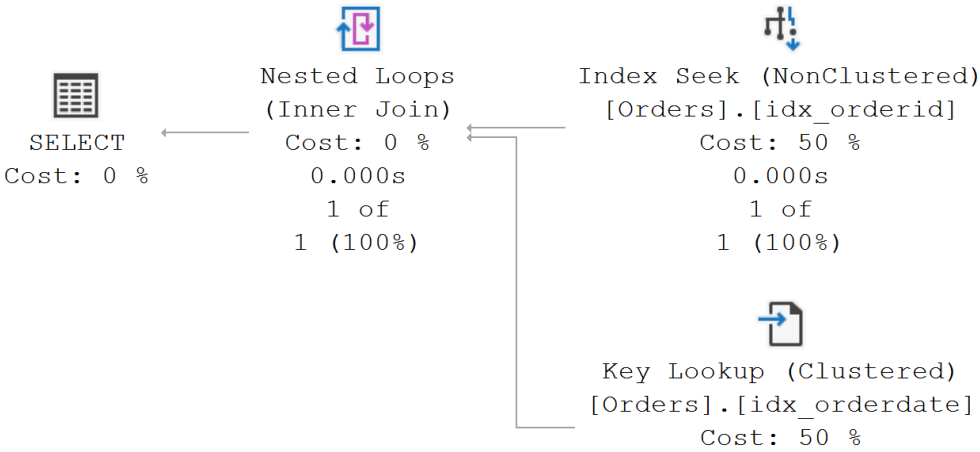

Figure 8: Plan for a dynamic query filtering by orderid

Cached Plans and Reuse Behavior

```sql
SELECT usecounts, text
FROM sys.dm_exec_cached_plans AS CP
  CROSS APPLY sys.dm_exec_sql_text(cp.plan_handle) AS ST
WHERE ST.text LIKE '%27702431-107C-478C-8157-6DFCECC148DD%'
  AND ST.text NOT LIKE '%sys.dm_exec_cached_plans%'
  AND CP.objtype = 'Prepared';
```

Cached Plans and Reuse Behavior (cont.)

usecounts text

1 (@oid AS INT, @cid AS INT, @eid AS INT, @dt AS DATE)SELECT orderid, custid, empid, orderdate, filler /* 27702431-107C-478C-8157-6DFCECC148DD */ FROM dbo.Orders WHERE 1 = 1 AND orderid = @oid

2 (@oid AS INT, @cid AS INT, @eid AS INT, @dt AS DATE)SELECT orderid, custid, empid, orderdate, filler /* 27702431-107C-478C-8157-6DFCECC148DD */ FROM dbo.Orders WHERE 1 = 1 AND orderdate = @dt

Discussion

- Optimal plans
- Efficient plan reuse

Section 5

Dynamic Sorting

Example

- GetSortedShippers stored procedure
  - Inputs: @colname
  - Returns Sales.Shippers sorted by @colname

Implementation with Static SQL and Recompile Option

```
CREATE PROC dbo.GetSortedShippers
  @colname AS sysname, @sortdir AS CHAR(1) = 'A'
AS

SELECT shipperid, companyname, phone
FROM Sales.Shippers
```

Implementation with Static SQL and Recompile Option (cont.)

```
ORDER BY
  CASE WHEN @colname = N'shipperid'   AND @sortdir = 'A'
       THEN shipperid   END,
  CASE WHEN @colname = N'companyname' AND @sortdir = 'A'
       THEN companyname END,
  CASE WHEN @colname = N'phone'       AND @sortdir = 'A'
       THEN phone       END,
  CASE WHEN @colname = N'shipperid'   AND @sortdir = 'D'
       THEN shipperid   END DESC,
  CASE WHEN @colname = N'companyname' AND @sortdir = 'D'
       THEN companyname END DESC,
  CASE WHEN @colname = N'phone'       AND @sortdir = 'D'
       THEN phone       END DESC
OPTION (RECOMPILE);
```

Test Query

```
EXEC dbo.GetSortedShippers N'shipperid', N'D';
```

Execution Plans

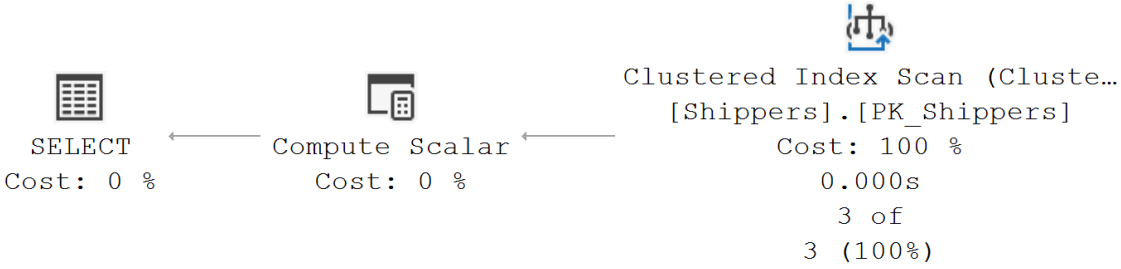

Figure 9: Plan for a static query with multiple case and recompile

Discussion

- Optimal plans
- No plan reuse
- Hard to extend

Implementation with Dynamic SQL

```
CREATE PROC dbo.GetSortedShippers
  @colname AS sysname, @sortdir AS CHAR(1) = 'A'
AS
IF @colname NOT IN(N'shipperid', N'companyname', N'phone')
  THROW 50001,
        'Column name not supported. Possibly a SQL injection attempt.', 1;

DECLARE @sql AS NVARCHAR(1000);
SET @sql = N'SELECT shipperid, companyname, phone
FROM Sales.Shippers
ORDER BY '
  + QUOTENAME(@colname) +
        CASE @sortdir WHEN 'D' THEN N' DESC' ELSE '' END + ';';

EXEC sys.sp_executesql @stmt = @sql;
```

Test Query

```
EXEC dbo.GetSortedShippers N'shipperid', N'D';
```

Execution Plans

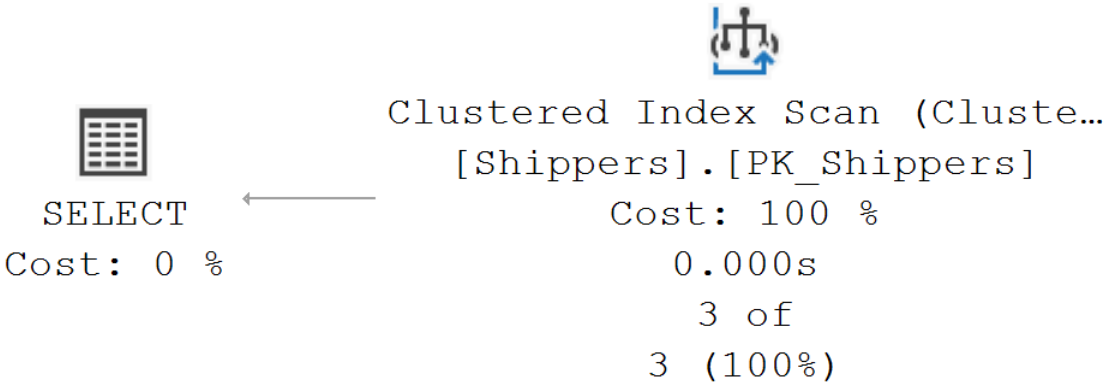

Figure 10: Plan for a dynamic query

Extend for Multi-column Sorting

```
CREATE PROC dbo.GetSortedShippers
  @colname1 AS sysname, @sortdir1 AS CHAR(1) = 'A',
  @colname2 AS sysname = NULL, @sortdir2 AS CHAR(1) = 'A',
  @colname3 AS sysname = NULL, @sortdir3 AS CHAR(1) = 'A'
AS
IF @colname1 NOT IN(N'shipperid', N'companyname', N'phone')
   OR @colname2 IS NOT NULL
        AND @colname2 NOT IN(N'shipperid', N'companyname', N'phone')
   OR @colname3 IS NOT NULL
        AND @colname3 NOT IN(N'shipperid', N'companyname', N'phone')
  THROW 50001,
        'Column name not supported. Possibly a SQL injection attempt.', 1;
```

Extend for Multi-column Sorting (cont.)

```
DECLARE @sql AS NVARCHAR(1000);

SET @sql = N'SELECT shipperid, companyname, phone
FROM Sales.Shippers
ORDER BY '
  + QUOTENAME(@colname1) +
        CASE @sortdir1 WHEN 'D' THEN N' DESC' ELSE '' END
  + ISNULL(N',' + QUOTENAME(@colname2) +
        CASE @sortdir2 WHEN 'D' THEN N' DESC' ELSE '' END, N'')
  + ISNULL(N',' + QUOTENAME(@colname3) +
        CASE @sortdir3 WHEN 'D' THEN N' DESC' ELSE '' END, N'')
  + ';';

EXEC sys.sp_executesql @stmt = @sql;
```