

Getting the Most from Query Plan Reuse

Andrew J. Kelly

akelly@SolidQ.com

Who Am I?

- **Mentor with SolidQ**
- **Data Platform MVP since 2001**
- **Contributing editor & author for SQL Server Pro Magazine**
- **Over 20 years of development and database experience**
- **Specialize in performance tuning and maintenance of VLDB / High-volume SQL Server databases**



Agenda

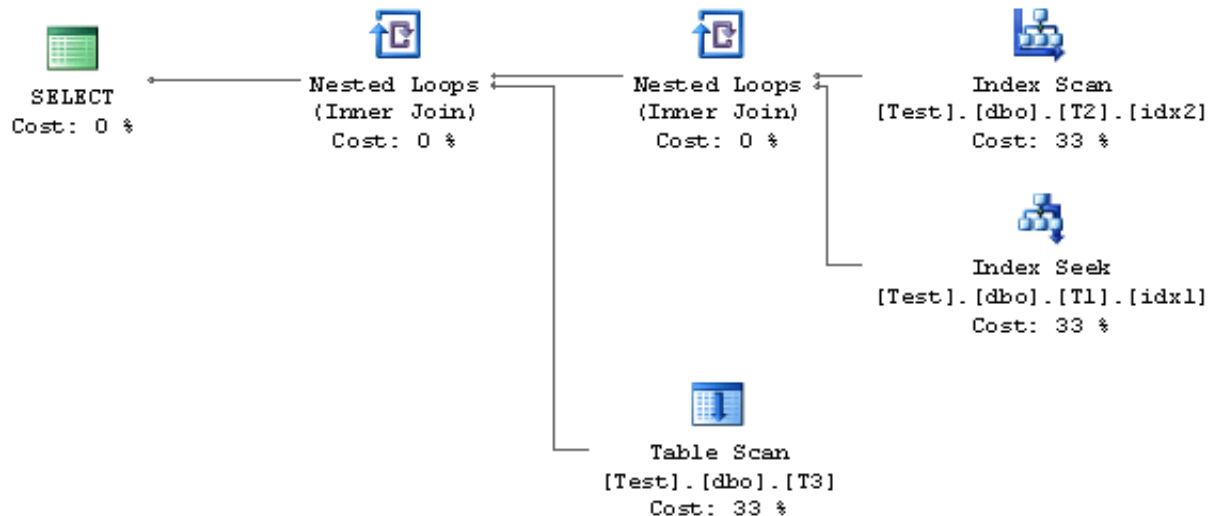
- **What is a query Plan?**
- **Who gets their own plan?**
- **Plan Cache**
- **Plan Compiles**
- **Plan Re-Use**

- **Questions & Answers**

What is a Query Plan?

- Set of instructions that tell the engine exactly how to perform the operations required to obtain the data requested

- Joins
- Seeks
- Scans
- Inserts
- Updates
- Deletes
- Etc.



- It can generate a Single and a parallel plan

Who gets their own plan

- **Batches**
- **Stored Procedures**
- **Triggers**
- **EXEC (@sql)**
- **sp_executeSql**
- **Some Individual statements**

What good are query plans?

- **They allow for Plan Reuse**

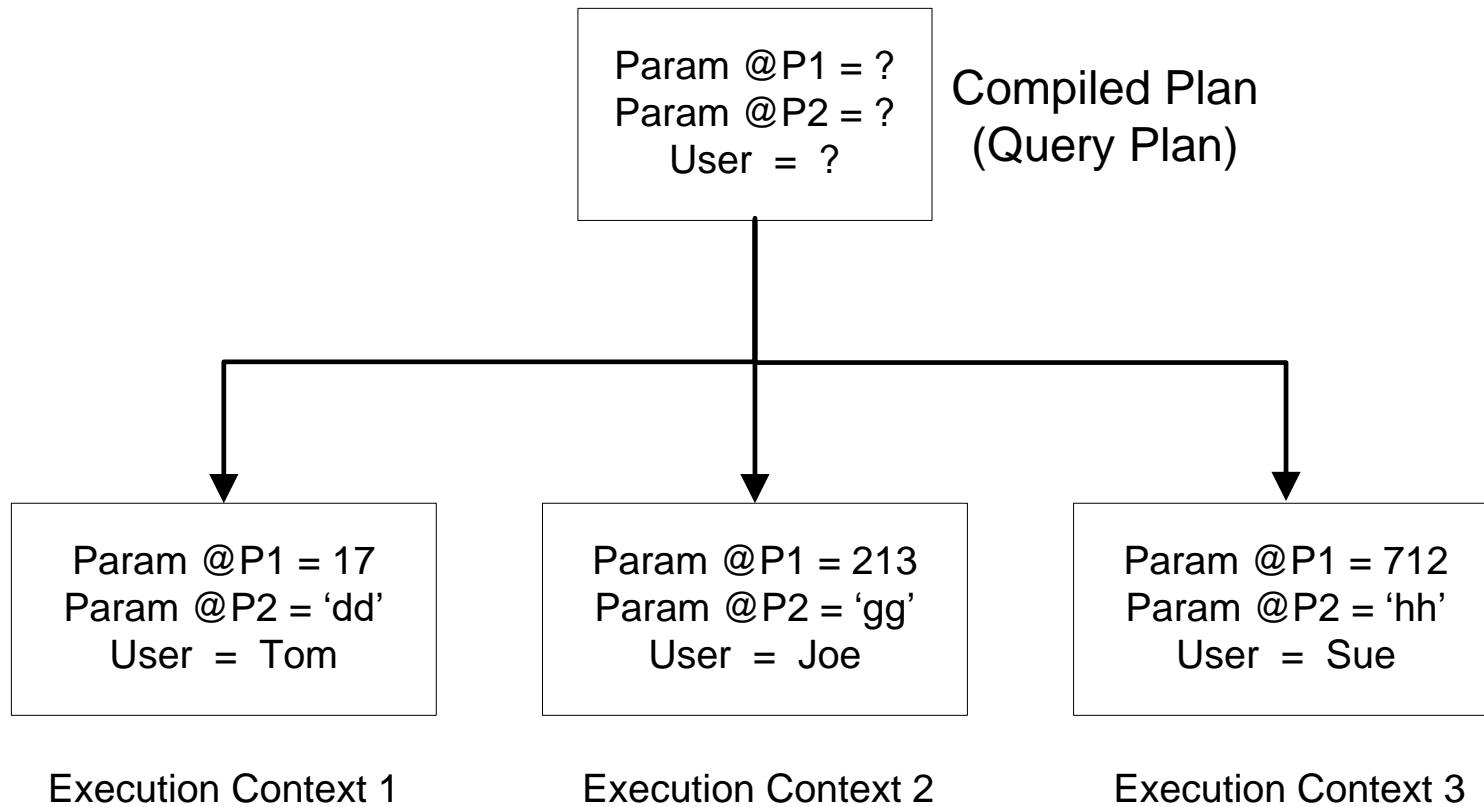


So why do we care about plan reuse?

- **It's all about Performance**

- Compiling a complex query plan is one of the most expensive things the engine does
- Too many compiles or recompiles can easily make you CPU bound on a busy system
- Additional time per query execution to compile the plan vs. doing the work just once
- The query lookup time can be much longer than you imagine
- Lack of plan reuse leads to excessive Memory usage and overhead to manage it

Beginnings of plan reuse



Where do the plans live?

- **In several DMV's**

- `sys.dm_exec_cached_plans`
- `sys.dm_exec_plan_attributes`
- `sys.dm_exec_sql_text`

- **How important is this information?**

- When it comes to performance you should know this information as well as any other on your server
- However there is a lot of junk in here as well



Plan Cache Memory

- **The plan cache memory is comprised of 4 main cache stores:**
 - Object Plans
 - SQL Plans
 - Bound Trees
 - Extended Procs
- **For more details of the plan cache memory see here:**
 - <http://blogs.msdn.com/b/sqlprogrammability/archive/2007/01/09/1-0-structure-of-the-plan-cache-and-types-of-cached-objects.aspx>

SQL Server Version	Cache Pressure Limit
SQL Server 2005 SP2 and >	75% of visible target memory from 0-4GB + 10% of visible target memory from 4Gb-64GB + 5% of visible target memory > 64GB
SQL Server 2005 RTM and SQL Server 2005 SP1	75% of visible target memory from 0-8GB + 50% of visible target memory from 8Gb-64GB + 25% of visible target memory > 64GB

Plan Costs and Retention

- **Each plan and execution context gets a cost associated with it**
 - This is assigned at compile time with max of 31
 - Based on several factors gathered during compile:
 - I/O cost
 - Context switch cost
 - Memory cost
- **Plans that are more costly to produce will stay in the cache longer**
- **Adhoc plans get an initial cost set to 0**
- **Factors can force a decrement of the cost by 1 on a periodic basis**
 - Can be in a lazy thread fashion
 - Or a more aggressive dedicated thread fashion
- **When a plan is reused it's cost gets set back to the original value**

Demo

Peeking at the Plan Cache



When does a plan get compiled?

- **Not at creation time. This is a myth**
- **The first time the batch, stored procedure, trigger etc. is called and the plan does not already exist in cache**
- **This will generate a compiled plan that hopefully will get cached in the procedure cache**
 - Trivial or adhoc plans may not be cached
- **An Execution plan is created when the compiled plan is used to actually return the results**
- **Keep in mind that the plan is case and space sensitive**
- **So what is the goal in regards to Compiles?**

When does a plan get recompiled?

- **That depends. It may never recompile or it might recompile each time it is executed**
- **What causes a plan to recompile?**
 - Changes to DDL for relevant objects
 - Mixing DDL and DML in the batch
 - Changes to statistics on relevant objects
 - Adding or dropping an index
 - # of changes to a table or to a particular column
 - SET Options
 - sp_recompile or WITH RECOMPILE
 - Host of Database level options
- **So what is the goal in regards to Recompiles?**

SET options that affect re-use

Number	SET Option Name
1	ANSI_NULL_DFLT_OFF
2	ANSI_NULL_DFLT_ON
3	ANSI_NULLS
4	ANSI_PADDING
5	ANSI_WARNINGS
6	ARITHABORT
7	CONCAT_NULL_YIELDS_NULL
8	DATEFIRST
9	DATEFORMAT
10	FORCEPLAN
11	LANGUAGE
12	NO_BROWSETABLE
13	NUMERIC_ROUNDABORT
14	QUOTED_IDENTIFIER

Object Name Resolution

- **Best practice is to ALWAYS schema qualify objects**
- **Take the statement “SELECT * FROM Person”. Which schema does it use?**
- **If this is an adhoc or prepared statement it depends on the users default schema**
 - If it's HR then we get “SELECT * FROM HR.Person”
 - If it's Sales then we get “SELECT * FROM Sales.Person”
- **If name resolution needs to occur for this scenario you won't get plan reuse**
- **Stored procedures and triggers don't suffer this same fate**

Plan reuse (cont)

- **Main ways to issue Sql statements**
 - Batches: Adhoc sql Statements
 - Batches: Adhoc Parameterized Statements
 - Batches: Adhoc Stored Procedure calls
 - Batches: sp_executeSql
 - RPC: Remote Procedure Calls

Batches: adhoc sql statements

```
SqlCommand.CommandText = "SELECT TOP 5 a.[OrderID], a.[CustomerID] FROM  
Orders AS a WHERE [OrderID] = 11048"
```

```
SqlCommand.CommandType = CommandType.Text
```

```
SqlDataReader DR = SqlCommand.ExecuteReader()
```

- Typically results in a SQL:BatchCompleted event
- This may or may not be cached
- This call will not reuse the plan from above even if it was cached:
 - SqlCommand.CommandText = "SELECT TOP 5 a.[OrderID], a.[CustomerID] FROM Orders AS a
WHERE [OrderID] = 11049"

Auto parameterization

```
SqlCommand.CommandText = "SELECT a.[OrderID], a.[CustomerID]  
FROM Orders AS a WHERE [OrderID] = 11048"
```

■ Auto parameterization

- The engine will turn the query plan into one with parameters in place of the OrderID literal
- SELECT a.[OrderID], a.[CustomerID] FROM Orders AS a WHERE [OrderID] = @p1 , @p1 INT , 11048

Batches parameterized queries

```
SqlCommand.CommandText = "SELECT a.[OrderID], a.[CustomerID] FROM Orders AS a  
WHERE [OrderID] = ?"
```

```
SqlCommand.Parameters.Add("@OrderID", SqlDbType. Char,5,"OrderID") ;
```

```
SqlCommand.Parameters["@OrderID"].Value = "11048" ;
```

```
SqlCommand.CommandType = CommandType.Text
```

```
SqlDataReader DR = SqlCommand.ExecuteReader()
```

- Typically results in a SQL:BatchCompleted & SP:Completed event as a sp_executeSql call
- This may or may not be cached
- However this call will reuse the plan from above if it was cached:
 - SqlCommand.CommandText = "SELECT a.[OrderID], a.[CustomerID] FROM Orders AS a
WHERE [OrderID] = ?"
 - SqlCommand.Parameters["@OrderID"].Value = "11049"

Batches: adhoc stored procs

```
SqlCommand.CommandText = "EXEC [dbo].[test_cache_1] @CustomerID = 'MAISD'"
```

```
SqlCommand.CommandType = CommandType.Text
```

```
SqlDataReader DR = SqlCommand.ExecuteReader()
```

- Typically results in a SQL:BatchCompleted event
- This may or may not be cached. This will generate two plans, one for the batch and one for the sp execution
- This call will not reuse the batch plan from above but may reuse the sp plan:
 - SqlCommand.CommandText = "EXEC [dbo].[test_cache_1] @CustomerID = 'DESF'"

Rpc's

```
SqlCommand.CommandText = "[dbo].[Your_sp] "  
SqlCommand.Parameters.Add("@OrderID", SqlDbType. Char,5,"OrderID") ;  
SqlCommand.Parameters["@OrderID"].Value = "11048" ;
```

```
SqlCommand.CommandType = CommandType.StoredProcedure
```

```
SqlDataReader DR = SqlCommand.ExecuteReader()
```

- **Results in a RPC:Completed**
- **This will be cached (unless a hint prevents it)**
- **This call will reuse the plan from above:**
 - `SqlCommand.CommandText = "[dbo].[Your_sp]`
 - `SqlCommand.Parameters["@OrderID"].Value = "11049"`

Query hints & forced parameterization

- **Parameterization**

- ALTER DATABASE SET PARAMETERIZATION { SIMPLE | FORCED }

- **Instance Level**

- Optimize for Adhoc Workloads

- **RECOMPILE**

- **KEEP PLAN**

- **KEEP_FIXED_PLAN**

- **OPTIMIZE FOR**

- **USE PLAN**

Query Plan Documents

Plan Caching in SQL Server 2012

<http://msdn.microsoft.com/en-us/library/dn148262.aspx>

Statistics used by the Optimizer in SQL2008

<http://technet.microsoft.com/en-us/library/dd535534.aspx>

How to troubleshoot the performance of Ad-Hoc queries

<http://support.microsoft.com/default.aspx?scid=kb;en-us;243588>

SP's & Caching (oldie but goodie)

<http://www.sqlservercentral.com/columnists/bkelley/procedurecache.asp>

Optimizing sp Recompiles

http://www.sql-server-performance.com/rd_optimizing_sp_recompiles.asp

Demo

Query Plan Reuse



Questions?

