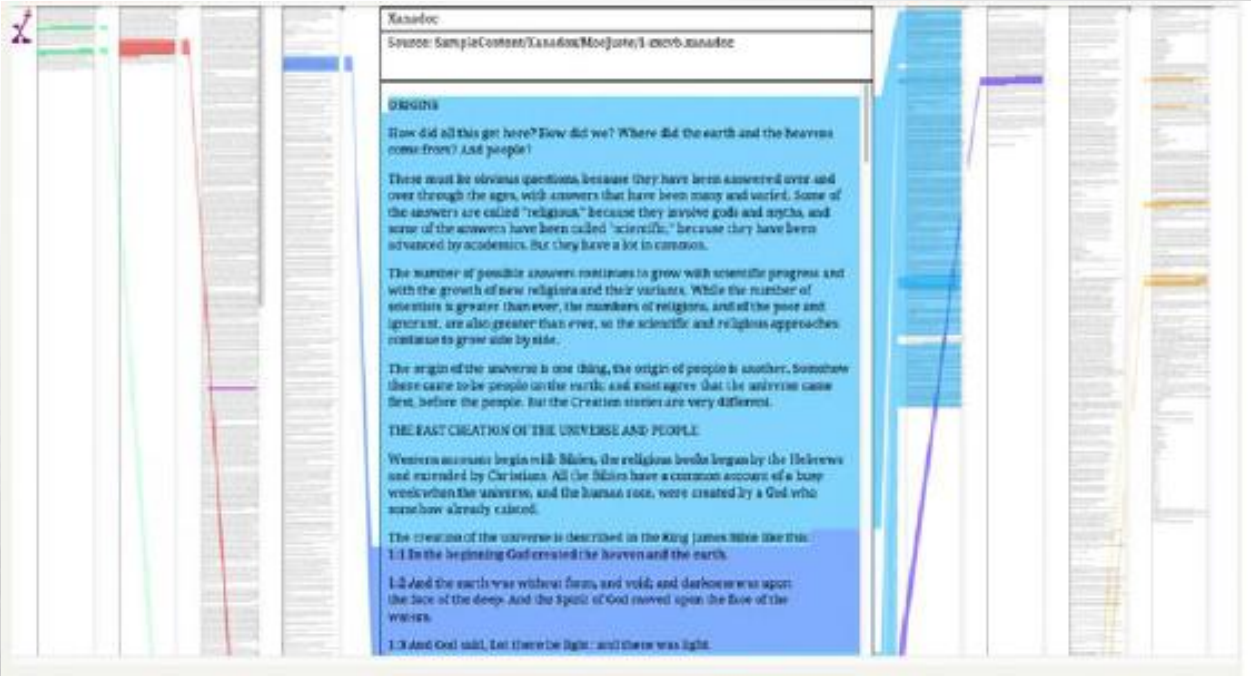
Novick Software

Writing Faster T-SQL Stored Procedures and Functions

Microsoft Most Valuable Professional

DID YOU EVER WORK ON A SOFTWARE PROJECT THAT WAS LATE?

Xanadu, The World's Most Delayed Software, Is Finally Released After 54 Years In The Making!

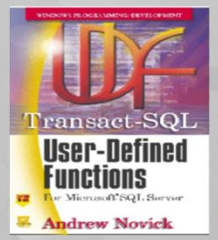


Novick Software

Andy Novick

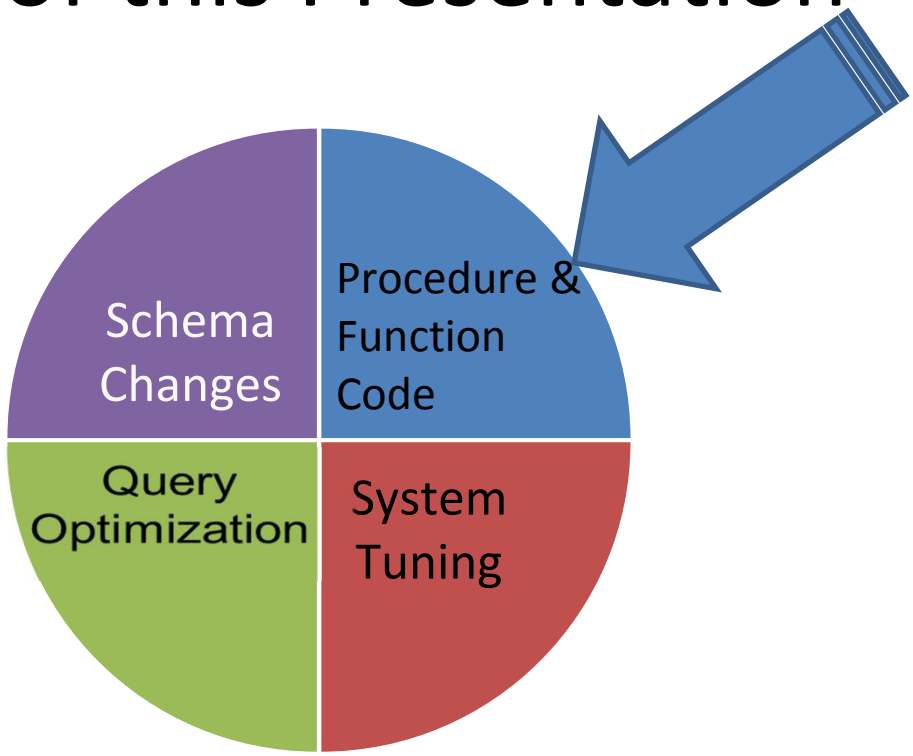
Novick Software

- SQL Server Consultant
- SQL Server MVP since 2010
- Author of 2 books on SQL Server
- anovick@NovickSoftware.com
- www.NovickSoftware.com



Microsoft® Most Valuable Professional

Focus of this Presentation



Agenda

- Know where the time is going!
- Functions
  - Why they can slow you down
  - Which ones slow you down
  - What to do about the problem functions
- Stored Procedures
  - Working with TempDB
  - Other Techniques

Know where the time going?

Figuring out where time is going

- Profiler/Server Traces
- DMV's
  - sys.dm_exec_procedure_stats
  - sys.dm_exec_query_stats
- Extended Events
- Write your own logging code

Profiler/Server Trace

SQL Profiler

- High overhead
- Too much detail
- Defaults aren't always best
- Best for development

Server Traces

- Moderate Overhead
  - Lower than Profiler
- Just as much detail
- Can be analyzed offline.
- Usable in production (with care)

Tracing – 3 Level Approach

- High Level – What sprocs are being executed
  - Might use sys.dm_exec_procedure_stats
  - Trace Template Minimal RPC and Batch
  - Has to come from production
- Medium Level – sproc call tree
  - Ns_trace_call_tree (dev only)
- Detailed
  - Template –Novick_Software_detailed**(DEV ONLY!)**

DEMO – BETTER TRACING DMVS

Better Tracing – User Templates

- Standard Templates
  - Difficult to find where the code is coming from

- User templates are in

  C:\Users\<yourusername>\AppData\Roming\Microsoft\SQL Profiler\10.0\Templates\Microsoft SQL Server\1050

  C:\Users\<yourusername>\AppData\Roming\Microsoft\SQL Profiler\12.0\Templates\Microsoft SQL Server\

Better Tracing – Loading a Trace File

```
SELECT * into mytracetable
FROM fn_trace_gettable('c:\temp\mytrc', 999)
```

- Join to get event names from

  sys.trace_events

DMV's

- sys.dm_exec_procedure_stats
- sys.dm_exec_query_stats

Extended Events

- The New Tool
- Configuration and use can be difficult
- Very low overhead
- Can be used “With Care” in production
- SQL 2012 includes the events in SQL Trace

Write Logging Code

- It's a lot of work!
- Can slow procedures
- Getting around transactions is tricky
- The only way to get the procedure parameters!
- If done well, can be used production
- It's usually worth the effort.

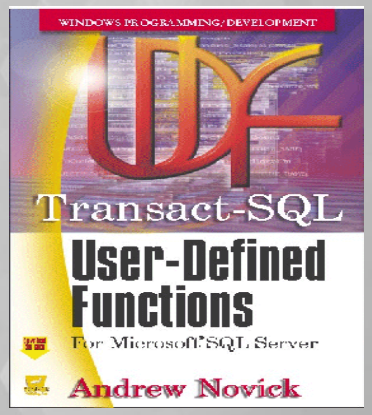
Logging code – What you get

Results | Messages

	time	W	sec	dth	event	volume	wrkr_sec	lgcl_read	lgcl_writ	phys_re...	query_plan	statement
1	17:10:08.418	S	413.140	1.0	dbo.ETL @Src=100, @DataFormat='D', @exp...	7,359,959	749.509	49,078,227	606,336	252,743	NULL	
2	17:10:08.448		0.004	1.5	.ETL_internal Load attributes	231	0.004	1,461	3	0	<ShowPlanXML x...	<?x INSERT INTO
3	17:10:08.481			1.5	.ETL_internal Add constraint to #unpivot						NULL	<?x ALTER TABL...
4	17:10:08.520			1.5	.ETL_internal add columns to #imported_data						NULL	<?x alter table #etl
5	17:10:08.563			1.5	.ETL_internal CREATE STATISTICS to suppr...						NULL	<?x CREATE STA.
6	17:10:11.062		2.432	1.5	.ETL_internal express copy	46,736	2.431	13,091	8,743	0	<ShowPlanXML x...	<?x insert into #etl
7	17:10:59.827		48.584	1.5	.ETL_internal unpivot #input_data	7,796,016	119.173	201,852	111,287	0	<ShowPlanXML x...	<?x INSERT into #
8	17:11:00.781		0.608	1.5	.ETL_internal treatment_1		12.733	129,988	0	0	<ShowPlanXML x...	<?x UPDATE #unp
9	17:11:13.252		12.459	1.5	.ETL_internal Update unpivot eliminate NULLs	3,440,739	12.458	128,411	116,705	0	<ShowPlanXML x...	<?x UPDATE td
10	17:11:19.637		6.369	1.5	.ETL_internal Update unpivot remove non-n...		6.369	126,089	0	0	<ShowPlanXML x...	<?x UPDATE tp
11	17:11:19.818		0.166	1.5	.ETL_internal Update unpivot remove non-d...		0.166	1,329	0	0	<ShowPlanXML x...	<?x UPDATE tp
12	17:11:26.519		6.420	1.5	.ETL_internal Find Duplicates into #records_...		6.420	127,177	3	0	<ShowPlanXML x...	<?x SELECT COU.
13	17:16:39.724		312.544	1.5	.ETL_internal MERGE eav_data from unpivot	7,355,277	520.697	45,898,925	358,007	232,129	<ShowPlanXML x...	<?x WITH ActiveR.
14	17:16:40.374		0.565	1.5	.ETL_internal INSERT eav_data from merge...	4,682	1.404	137,530	10,604	38	<ShowPlanXML x...	<?x INSERT INTO
15	17:16:41.088		0.429	1.5	.ETL_internal INSERT #eav_data_log	231	8.657	69,156	0	0	<ShowPlanXML x...	<?x INSERT INTO
16	17:16:45.977		4.873	1.5	.ETL_internal MERGE #eav_data_log to_get...	231	4.871	129,335	0	0	<ShowPlanXML x...	<?x MERGE INTO
17	17:16:46.932		0.938	1.5	.ETL_internal INSERT eav_data_log	231	0.003	467	231	823	<ShowPlanXML x...	<?x INSERT INTO
18	17:16:50.178		3.229	1.5	.ETL_internal INSERT @nodes	231	3.229	128,873	0	0	<ShowPlanXML x...	<?x INSERT INTO
19	17:16:50.181	S	11.259	2.0	.ETLAttribSummary @AsofDatetime='2014-...	231	38.785	480,373	296	1,229	NULL	
20	17:17:00.416		1.681	2.5	.ETLAttribSummary Calculate #nodes_wit...	231	31.202	234,808	1	640	<ShowPlanXML x...	<?x WITH cte AS
21	17:17:01.018		0.543	2.5	.ETLAttribSummary INSERT #existing_no...	231	7.474	234,056	9	9	<ShowPlanXML x...	<?x INSERT INTO
22	17:17:01.439		0.363	2.5	.ETLAttribSummary INSERT DL_attribute_...	231	0.004	964	231	580	<ShowPlanXML x...	<?x INSERT INTO
23	17:17:01.440	E	11.259	2.0	.ETLAttribSummary @AsofDatetime='2014-...	231	38.785	480,373	296	1,229	NULL	
24	17:17:01.501			1.5	.ETL_internal Table inserts done at 2014-08...						NULL	
25	17:17:01.558	E	413.140	1.0	dbo.ETL @Src=100, @DataFormat='D', @exp...	7,359,959	749.509	49,078,227	606,336	252,743	NULL	

Transact-SQL User Defined Functions

- This SQL 2000 book is still valid.
- PDF available for free
  - HTTP://www.NovickSoftware.com/



Functions

- A great way to encapsulate reusable logic
- Scalar Functions are slow
  - Row-by-Row cursor-like processing
  - They inhibit query parallelism
- Table Valued Functions (TVF) are slow
  - Row-by-Row cursor-like processing
  - Inhibit query parallelism
  - Use of a @Table Variable to return data
- Inline Functions are fast
  - They're Views with parameters

Which Functions should be Rewritten?

- Hard to find: functions are not in sys.dm_exec_procedure_stats
- Traces can record functions
  - Event SP:Procedure Complete
  - Filter on ObjectType = 20038
  - Lots of overhead!
- Extended Events is the low overhead way to measure function use

How about a SQLCLR function?

- Write Scalar or TVF in C# or VB.Net
- Great for complex algorithms that don't do much data access
  - Analytics or Statistics
  - String manipulations
  - When the code has loops/cursors
  - When the code needs arrays
- Aggregates

DEMO – FUNCTION OVERHEAD AND PARALLELISM AND EXTENDED EVENTS

Making Stored Procedures Faster

- Better Algorithms
- Query Tuning
- Working with TempDB
- Updates:
  Divide & Conquer with Service Broker

Faster Procedures For OLTP

- Fully qualify names dbo.load_table_value
- Never name a procedure sp_.....
  Use usp_... or just the … part
- Limit the number of temp tables and limit schema changes to temp tables
- Any CURSORs should be LOCAL
- Consider SQL 2014 In-Memory tables and Natively Compiled procedures

What is necessary is not the net but the fish.

Example: A better algorithm

The Problem: compare 2 tables

Table_1

ColA_PK	ColB	ColC	ColD
ABCD	1.24456	3.21e-16	Andy
EFGH	1.9123	10002	Eric

Table 2

ColA_PK	ColB	ColC	ColD
ABCD	2.2456	2.11e-15	Andy
EFGH	1.9123	NULL	Tom

Differences

PK_Value	Col_name	Value_b	Value_a	Msg
ABCD	ColB	1.24456	2.2456	Not equal
EFGH	COLC	10002	NULL	Only Table_1
EFGH	ColD	Eric	Tom	Not equal

The Old Way - UNPIVOT

Table_1

ColA_PK	ColB	ColC	ColD
ABCD	1.24456	3.21e-16	Andy
EFGH	1.9123	10002	Eric

Table_2

ColA_PK	ColB	ColC	ColD
ABCD	2.2456	2.11e-15	Andy
EFGH	1.9123	NULL	Tom

#Unpivot_1

PK_value	Col_name	Value_str
ABCD	ColB	1.24456
ABCD	ColC	3.21e-16
ABCD	ColD	Andy
EFGH	ColB	1.9123
EFGH	ColC	10002
EFGH	ColD	Eric

#unpivot_2

PK_value	Col_name	Value_str
ABCD	ColB	2.2456
ABCD	ColC	2.11e-15
ABCD	ColD	Andy
EFGH	ColB	1.9123
EFGH	ColC	NULL
EFGH	ColD	Tom

Compare UNPIVOTed Tables

```
SELECT Coalesce(a.ColA_PK, b.ColA_PK) ColA_PK
     , coalesce(a.column_name, b.column_name) col_name
     , CASE WHEN b.column_name is null or b.value is NULL
            THEN ‘only After'
            WHEN a.column_name is NULL or a.value is NULL
            THEN ‘only Before'
            WHEN a.value is NOT NULL and b.value IS NOT NULL
                 and a.value!=b.value
            THEN 'not equal'
            ELSE 'unknown issue‘ END
FROM #unpivot_1 b FULL outer join #unpivot_2 a
                        on b.ColA_PK = a.ColA_PK
WHERE 1=CASE WHEN b.column_name is null or b.value is NULL then 1
             WHEN a.column_name is NULL or a.value is NULL then 1
             WHEN a.value is NOT NULL and b.value IS NOT NULL
                  and a.value!=b.value THEN 1
             ELSE 0
             END
```

New Way – Compare in Place

Table_1

ColA_PK	ColB	ColC	ColD
ABCD	1.24456	3.21e-16	Andy
EFGH	1.9123	10002	Eric

Table_2

ColA_PK	ColB	ColC	ColD
ABCD	2.2456	2.11e-15	Andy
EFGH	1.9123	NULL	Tom

```sql
WITH issue_lines_cte AS(
SELECT COALESCE(b.[ColA_PK], a.[ColA_PK]) ColA_PK
  ,CASE WHEN b.[ColB] IS NOT NULL AND a.[ColB] IS NULL
        THEN '|ColB^only before^C^'+ b.[ColB]+'^'
        WHEN b.[ColB] IS NULL AND a.[ColB] IS NOT NULL
        THEN '|ColB^only after^C^^'+ a.[ColB]
        WHEN b.[ColB] != a.[ColB]
        THEN '|ColB^not equal^'+ b.[ColB] + '^' + a.[ColB]   ELSE '' END
 + CASE WHEN b.[ColC] IS NOT NULL AND a.[ColC] IS NULL
        THEN '|ColC^only before^C^'+ b.[ColC]+'^'
        WHEN b.[ColC] IS NULL AND a.[ColC] IS NOT NULL
        THEN '|ColC^only after^C^^'+ a.[ColC]
        WHEN b.[ColC] != a.[ColC]
        THEN '|ColC^not equal^C^'+ b.[ColC] + '^' + a.[ColC] ELSE '' END
  . . .
     issue_line
   FROM Table_1 b FULL OUTER JOIN Table_2 a ON b.[PK_ColA] = a.[PK_ColA]
)
INSERT INTO #issue_lines (PK_value, issue_line)
SELECT ColA_PK, issue_line
FROM issue_lines_cte
WHERE issue_line != ''
```

When looking for better logic:

- Don’t do what does not have to be done
- Don’t move data round unnecessarily
- Use T-SQL for what it’s good at and no more

TempDB – Why is it Different?

- Review the ACID properties
- The ACID property TempDB doesn't need?
- What TempDB does differently

ACID Properties

Atomicity	Each transaction is “all or nothing”
Consistency	Database moves from one valid state to another
Isolation	Ensures that concurrent execution results in the same state as serial transactions
Durability	Once a transaction is committed it remains so even if there are crashes, errors, power loss

What TempDB does Differently

- It is not recovered!
  - New tempdb at each startup
- Minimal logging, usually just allocations
- Checkpoint doesn't write the pages
- Fewer Log Records written
  - i.e. Updates – no after image rows needed

@Table Variables

- Are objects in tempdb
- Can be flushed to disk
- Can have Primary Key and Unique Indexes
- INSERT … EXEC (sql) allowed since 2005
- NOT part of transactions other than the statement
- Use only unnamed CHECK or UNIQUE constraints

- In SQL 2014, can be memory optimized!

@Table vs #Table

@Table

- Scope: Current batch/proc
- Collation: current DB
- Name: 128 Characters
- Indexes: PK, Unique when created
- Statistics: NONE!
- DROP: automatically at the end of the proc/batch
- Txn: Only for statement
- Rollback: Unaffected

#Table

- Scope: current Session
- Collation: tempdb
- Name: 116 Characters
- Indexes: Any at any time
- Statistics: YES!
- DROP: Automatically when session ends or out of scope
- Txn: Length of transaction
- Rollback: part of rollback

Query Optimization of @Table

- Optimizer cannot create column statistics
- Always treated it as if it has 1 row
- Recommendation:
  - Use when required - Functions
  - Use when there are few rows in the table
  - Use to jump around ROLLBACK TRAN

  - Otherwise, use #Tables

Temp Table Caching

- Since SQL Server 2005 SP3, SQL Server caches a few pages of temp table definitions by renaming them.
- Saves around 2 milliseconds/table/proc call
- Important with very high activity
- Restrictions
  - No DDL Allowed on the table except DROP TABLE
  - No named constraints
  - Less than 8 MB

CHECK @@ROWCOUNT

```
DECLARE @myRowCount INT
SELECT into #driver_table (attribute_id)
    FROM metadata_table_A a
        inner join dbo.fnMetaDataFunctionB(@arg1, @arg2) b
           on a.attribute_id = b.attribute_id
SELECT @myRowCount = @@ROWCOUNT

IF @myRowCount > 0 BEGIN
    UPDATE d
       SET d.…
        FROM data_table d
        INNER JOIN #driver_table  dr
        on d.attribute_id = dr.attribute_id
END
```

Limit Schema Changes to #Temp

```
CREATE TABLE #myTempTab (ColA int identity (1,1) NOT NULL
                                 primary key clustered
                        ,ColB varchar(255) NULL
                        ,ColC FLOAT NULL
                        ,ColD DATE NULL
)
...
If @Var1 = 'N'
    ALTER TABLE #myTempTab ADD ColC float NOT NULL
ELSE
    ALTER TABLE #myTempTab ADD ColD date NOT NULL
```

Query Tuning – Add Temp Tables

- Simplify queries by SELECTing driver rows to a temp #table

- Create a query with:
  - a INNER JOIN or
  - CROSS APPLY
    from the driver to the data.
  - You usually get a better query plan

```sql
SELECT d.*, a.attribute_name, f.another_attribute_value
    FROM dbo.data_table d
    inner join dbo.Metadata_table_A a
        on d.attribute_id = a.attribute_id
    inner join dbo.fn_metadata_function (@var1, @var2) f
        on a.another_attribute_id =  f.another_attribute_id
```

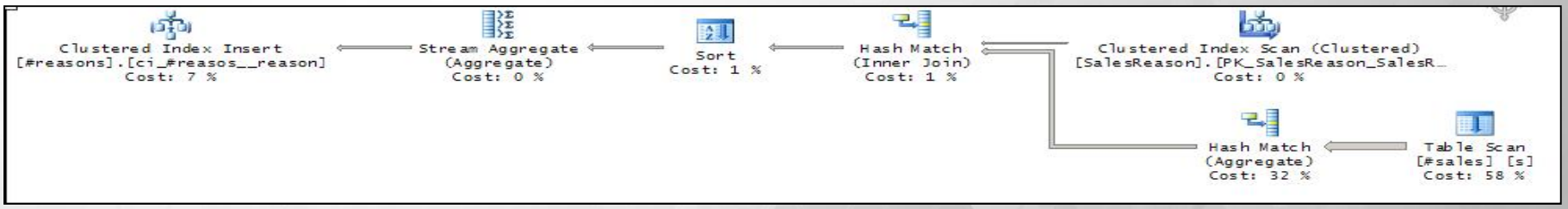
```sql
CREATE TABLE #metadata (attribute_id int PRIMARY KEY CLUSTERED
                       ,attribute_name varchar(255)
                       ,another_attribute_value float  )

INSERT #metadata (attribute_id, attribute_name, another_attribute_value)
    SELECT a.attribute_id, a.attribute_name, f.attribute_value
        FROM dbo.Metadata_table_A a
        inner join dbo.fn_metadata_function (@var1, @var2) f
              on a.another_attribute_id = f.another_attribute_id

SELECT d.*, m.attribute_name ,m.another_attribute_value
    FROM #metadata m
    CROSS APPLY(SELECT *
              FROM dbo.data_table d
              WHERE d.attribute_id = M.attribute_id )
```

Index in TempDB

- Great if they actually get used!
- If the index is used only once, consider a heap

- If the table has multiple inserts
  - Create the index after the inserts
  - Before any use
- If the table has only one insert
  - Try creating the index when the table is created

Examine the Query Plan



Optimizing Indexes on TempDB

- Try it without the indexes
- Try creating the indexes after inserts
- Try creating non-unique indexes on fewer columns
- Try
- Try
- Try
- Look at the query plans that you get!

Avoid Deletes from a #TempTable

```
CREATE TABLE #myTemp (ColA varchar(64) NOT NULL
                    , ColB float NULL
                    ,is_deleted BIT Default (0))

INSERT INTO #myTemp …

DELETE #myTemp  WHERE <some_predicate> ColB < 3.14159

UPDATE #myTemp  SET is_deleted = 1
                WHERE <some_predicate> ColB < 3.14159

SELECT *
    FROM #myTemp
    WHERE is_deleted=0
```

Novick Software

SELECT … INTO instead of UPDATE

SELECT INTO a new table is often faster then UPDATE and DELETE

```
CREATE TABLE #myTemp (ColA varchar(64) NOT NULL
                    , ColB float NULL       )
UPDATE #myTemp SET …    WHERE <some_predicate>
UPDATE #myTemp SET …    WHERE <some_other_predicate>
DELETE #myTemp          WHERE <delete_predicate>

SELECT  …CASE WHEN <some_predicate> THEN … ELSE … END
        …CASE WHEN <some_other_predicate> THEN … ELSE … END
    INTO #myNewTemp
    FROM #myTemp
    WHERE NOT ( <delete_predicate> )
```

Suppress Statistics on #Tables

Only do this on #TempTables

```
CREATE STATISTICS stat_#tmp_col9
    on #tmp(col9)with NORECOMPUTE;
```

What SQL Server knows could hurt you!

Novick Software

Parameter Sniffing!

- SQL Serve optimizes as soon as it can!
- It doesn’t wait for the statement to be run.
- It uses stored procedures values if they’re available.

Stopping Parameter Sniffing

- Copy @parameters to @local_variables
- HINT: OPTIMZE FOR <specific value>
- HINT: OPTIMIZE FOR UNKNOWN
- Dynamic SQL and OPTION RECOMPILE

Updates: Divide & Conquer with Service Broker

- Break the problem down based on some column that has 2-10,000 distinct entries
- Instead of updating all at once, add each entry to a queue.
- The the SB process does updating
- Use Partitioning to eliminate contention

```
ALTER TABLE myTable SET (LOCK_ESCALATION=AUTO)
```

Divide & Conquer Before

Driver_Table

Attr_id
1
2
3
…

Data_Table

Attr_id	Entity_id	Value
1	ABCD	2
1	DEFG	3
3	HIJK	1
3	LMNO	2
3	PQRS	2.34

Update

Updated_Data_Table

Attr_id	Entity_id	Value	Other
1	ABCD	2	…
2	ABCD	5	…
3	ABCD	1	…
3	LMNO	2	…
3	PQRS	2.34	…
4	ABCD	34.53	…

Divide & Conquer After

Driver_Queue

Attr_id
1
2
3
…

Data_Table

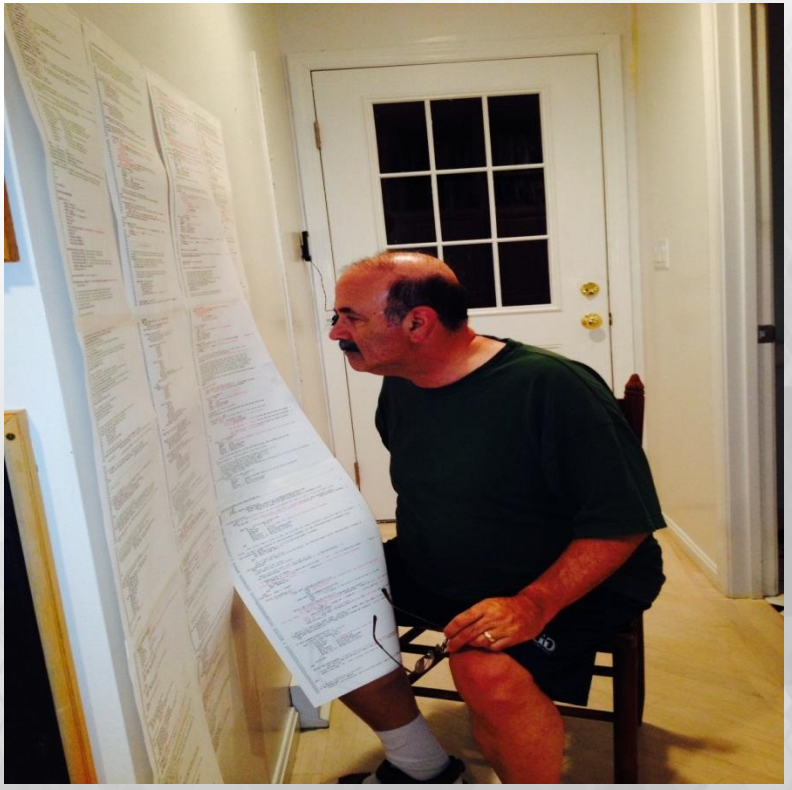
Attr_id	Entity_id	Value
1	ABCD	2
1	DEFG	3
3	HIJK	1
3	LMNO	2
3	PQRS	2.34

SB Task
SB Task
SB Task
SB Task

Updated_Data_Table

Attr_id	Entity_id	Value	Other
1	ABCD	2	…
2	ABCD	5	…
3	ABCD	1	…
3	LMNO	2	…
3	PQRS	2.34	…
4	ABCD	34.53	…

Live with your code!



References

- Adventureworks Databases for 2008 to 2014
  http://msftdbprodsamples.codeplex.com/releases/view/93587
- Proposed solution to the performance problem with SQL Server Scalar UDFs – Vote on Connect
  http://173.254.94.169/proposed-solution-to-the-performance-problem-with-sql-server-scalar-udfs/
- Mastering SQL Server Profiler by Brad M. McGehee
  http://download.red-gate.com/ebooks/SQL/Mastering_Profiler_eBook.pdf

References

- Novick Software web site: http://www.NovickSoftware.com

- Simple process to track and log SQL Server stored procedure use
  http://www.mssqltips.com/sqlservertip/2003/simple-process-to-track-and-log-sql-server-stored-procedure-use/
- Methods to collect SQL Server Stored Procedure Execution History
  http://www.mssqltips.com/sqlservertip/3259/several-methods-to-collect-sql-server-stored-procedure-execution-history/
- SQL Server Performance Statistics Using a Server Side Trace
  http://www.mssqltips.com/sqlservertip/1035/sql-server-performance-statistics-using-a-server-side-trace/

- Articles that written by Andy Novick on MSSQQLTIPS
  http://www.mssqltips.com/sqlserverauthor/10/andy-novick/

New England Microsoft Developers UG

- 1st Thursday of the Month
- Meeting in Burlington, MA
  - Foliage, 20 North Avenue
- http://www.meetup.com/NE-MSFT-Devs/
- May 7: Prof Pito Salas: University Computing
- June 4: Kevin Ford, Magenic – Phone Apps

Novick Software

Thank you for coming

Andy Novick
anovick@novicksoftware.com

www.NovickSoftware.com

Microsoft Most Valuable Professional