PASS SQLRally

Microsoft PASS

Query Performance Tuning in SQL Server

Grant Fritchey

Product Evangelist – Redgate Software

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Get in touch

blog scarydba.com

email grant@scarydba.com

twitter @gfritchey

Why Tune Queries?

Most volatile aspect of a database system

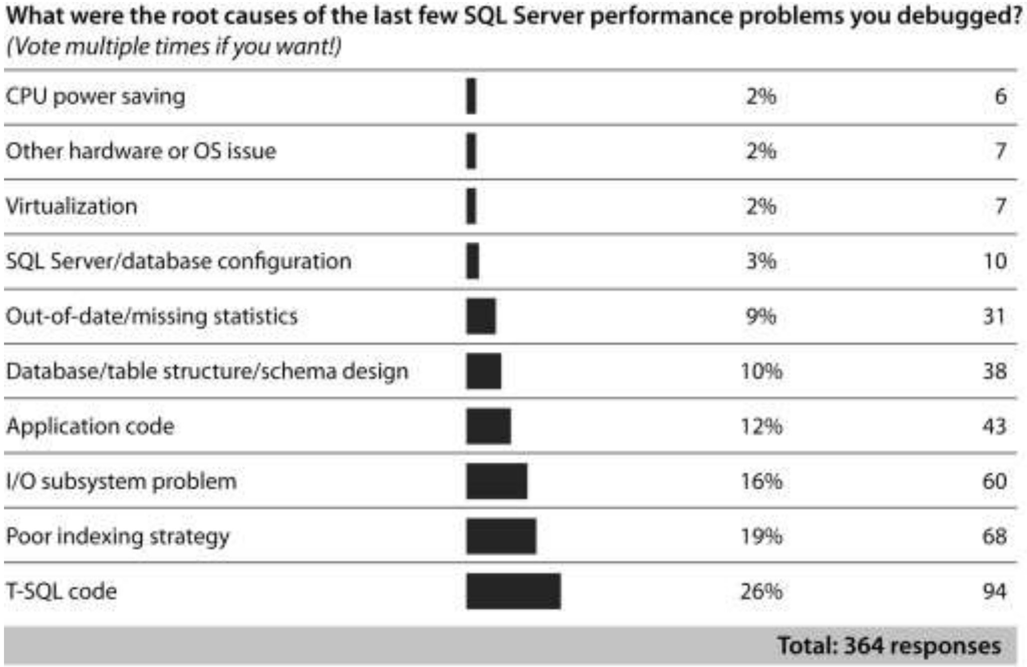
Subject to changes in data

Affected by changes in structure

Impacted by poor coding choices

Victim of resource contention

Why Tune Queries?



What were the root causes of the last few SQL Server performance problems you debugged?
(Vote multiple times if you want!)

CPU power saving	2%	6
Other hardware or OS issue	2%	7
Virtualization	2%	7
SQL Server/database configuration	3%	10
Out-of-date/missing statistics	9%	31
Database/table structure/schema design	10%	38
Application code	12%	43
I/O subsystem problem	16%	60
Poor indexing strategy	19%	68
T-SQL code	26%	94

Total: 364 responses

http://sqlskills.com/blogs/paul/post/survey-results-common-causes-of-performance-problems.aspx

Agenda

Capturing Query Performance

Optimizer, Statistics, Indexes, Constraints

Reading Execution Plans

Identifying and Fixing Common Problems

New Functionality

Query Performance Tuning in SQL Server

CAPTURING QUERY PERFORMANCE

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Where to Start Tuning?

Random

Pick a query?

Ask a user?

Alphabetically?

Knowledge based

Baseline

Metrics

Records

Server Metrics

Start query tuning at the server

Hardware

Operating system

SQL Server

Establish a baseline

Now is a good time

Save the data

Query Metrics

This is where you live

Too much information

Save the data, just not in its original form

Dynamic Management Objects

These are dependent on cache

No run-time information

Uses T-SQL

Mix & Match

DMOs

Sys.dm_exec_requests

Sys.dm_exec_query_stats

Sys.dm_exec_procedure_stats

QUERY METRICS: THE RIGHT WAY

Extended Events

Lightweight and low cost

XML output

Can be left on the server

Work through GUI or T-SQL

Can output to various locations

RML Utilities

Free

Huge time savings

Excellent resource

Still need long-term storage & reporting

QUERY METRICS: THE OLD WAY

The Server Side Trace

Profiler to generate the script

Files work best

Clean and store the data

Profiler GUI can be used to browse data

Works with Perfmon data

Schedule the start and stop

DO NOT USE PROFILER GUI ON PRODUCTION SYSTEMS

Metrics Resources

"SQL Server 2012 Query Performance Tuning"

Microsoft White Paper: Performance Tuning Waits and Queues.doc
http://technet.microsoft.com/en-us/library/cc966413.aspx

Microsoft White Paper: Troubleshooting Performance Problems in SQL Server 2008
http://msdn.microsoft.com/en-us/library/dd672789.aspx

Performance Tuning with SQL Server Dynamic Management Views, by Louis Davidson and Tim Ford

Questions?

How would you... ?

What happens when... ?

Why does...?

When do I... ?

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Query Performance Tuning in SQL Server

OPTIMIZER, STATISTICS, INDEXES & CONSTRAINTS

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Optimizer

Simply an amazing piece of software

Cost-based

Not perfect

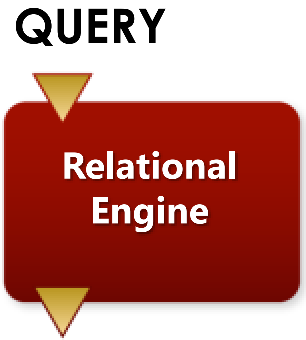
Plan on helping the Optimizer

Relational Engine

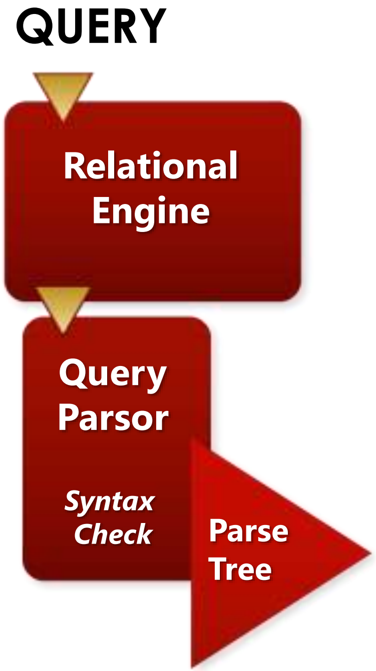
QUERY

▼

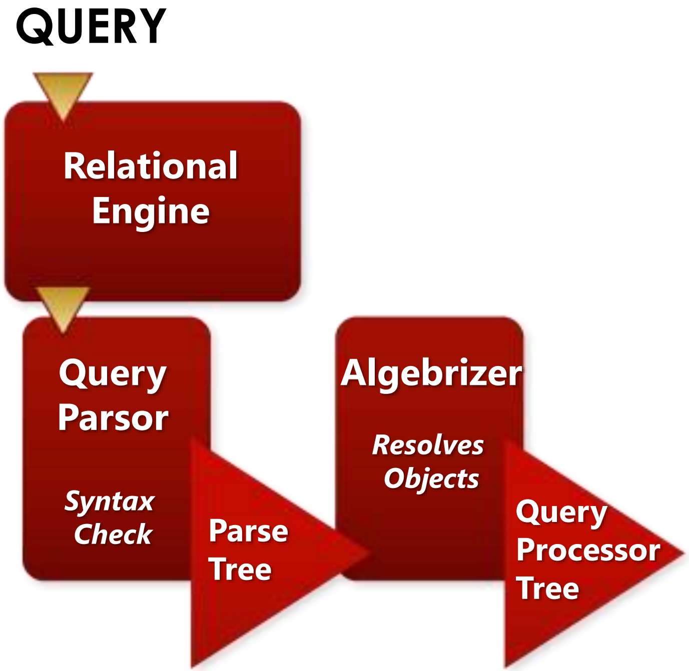
Relational Engine



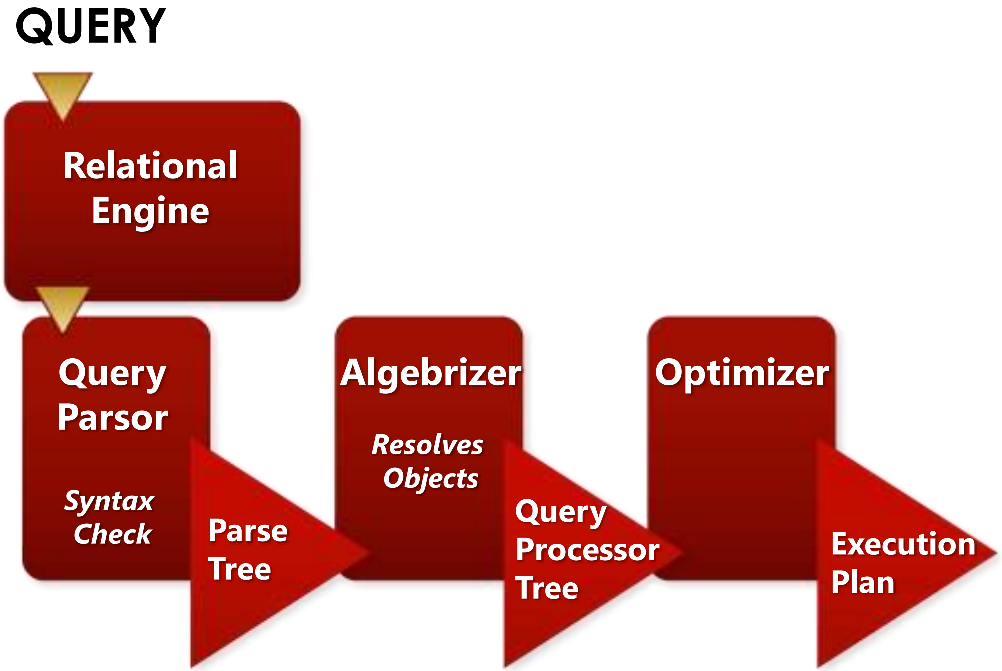
Relational Engine



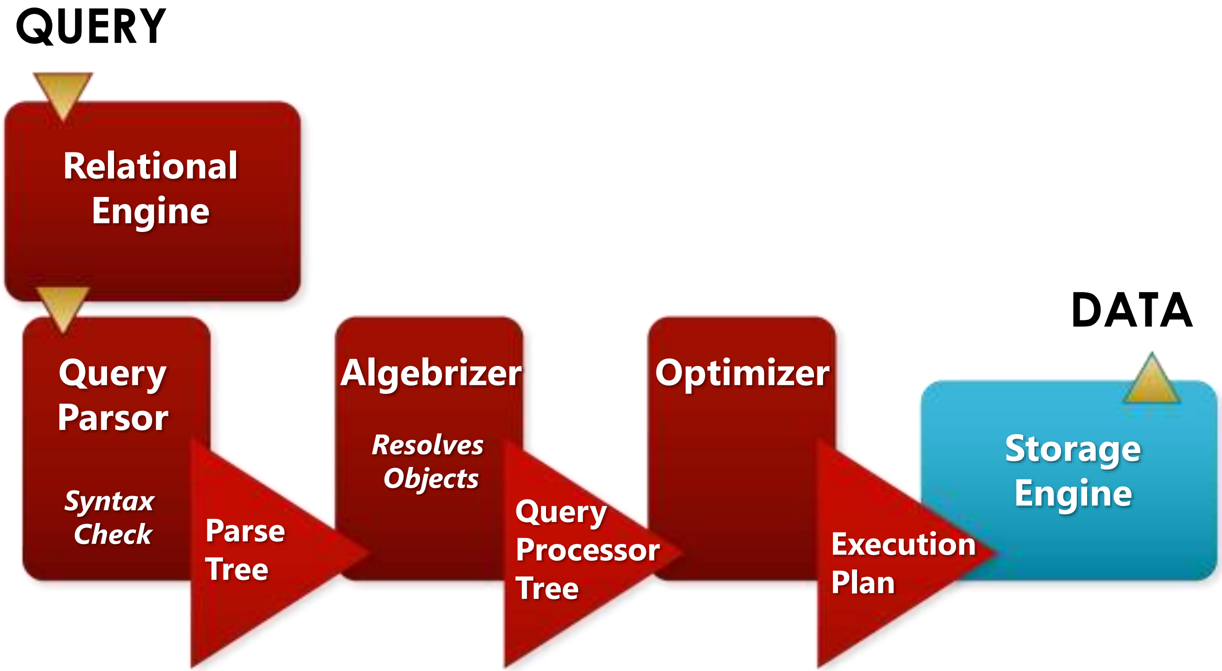
Relational Engine



Relational Engine



Relational Engine



QUERY

Relational Engine

Query Parsor

Syntax Check

Parse Tree

Algebrizer

Resolves Objects

Query Processor Tree

Optimizer

Execution Plan

Storage Engine

DATA

Observing the Optimizer

Sys.dm_exec_query_optimizer_info

Execution plans

Statistics

Information about the distribution of the data

Created on index keys

Created on columns

Created manually

Cardinality

By default, created automatically

By default, maintained automatically

Automatic maintenance is not enough

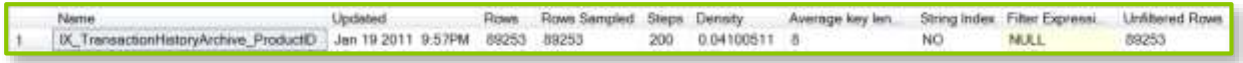
Investigating Statistics

DBCC SHOW_STATISTICS(*table, target*)

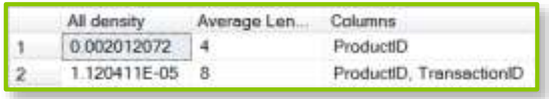
Header

Density

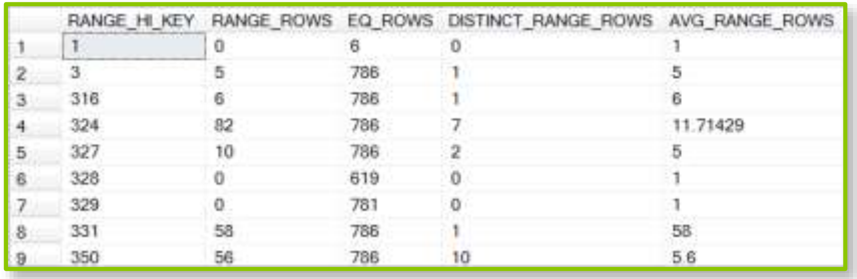
Histogram



	Name	Updated	Rows	Rows Sampled	Steps	Density	Average key len	String Index	Filter Expressi	Unfiltered Rows
1	IX_TransactionHistoryArchive_ProductID	Jan 19 2011 9:57PM	89253	89253	200	0.04100511	8	NO	NULL	89253



	All density	Average Len...	Columns
1	0.002012072	4	ProductID
2	1.120411E-05	8	ProductID, TransactionID



	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	1	0	6	0	1
2	3	5	786	1	5
3	316	6	786	1	6
4	324	82	786	7	11.71429
5	327	10	786	2	5
6	328	0	619	0	1
7	329	0	781	0	1
8	331	58	786	1	58
9	350	56	786	10	5.6

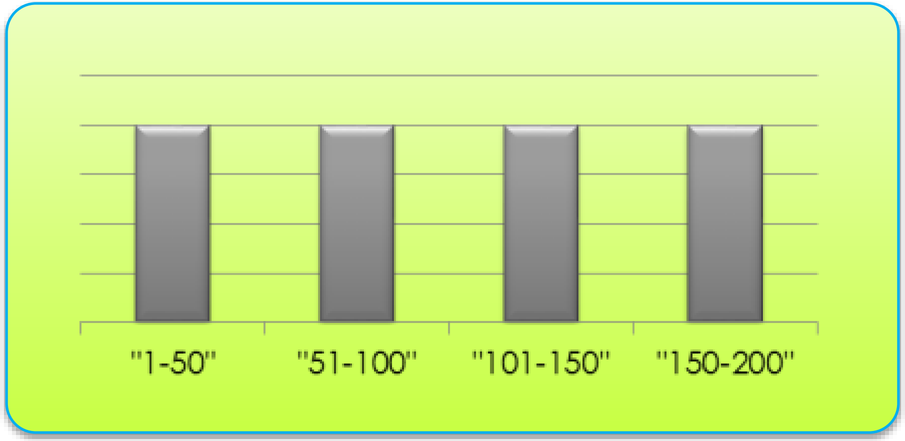
Histogram

200 steps across the data

An equal distribution of rows

Leads to best possible sampling of data

But it’s not perfect



Updating Statistics

sp_updatestats

Can resample

Won’t run everywhere

UPDATE STATISTICS X

WITH FULLSCAN

AUTO_UPDATE_STATISTICS_ASYNC (2014)

INCREMENTAL (2014)

Indexes

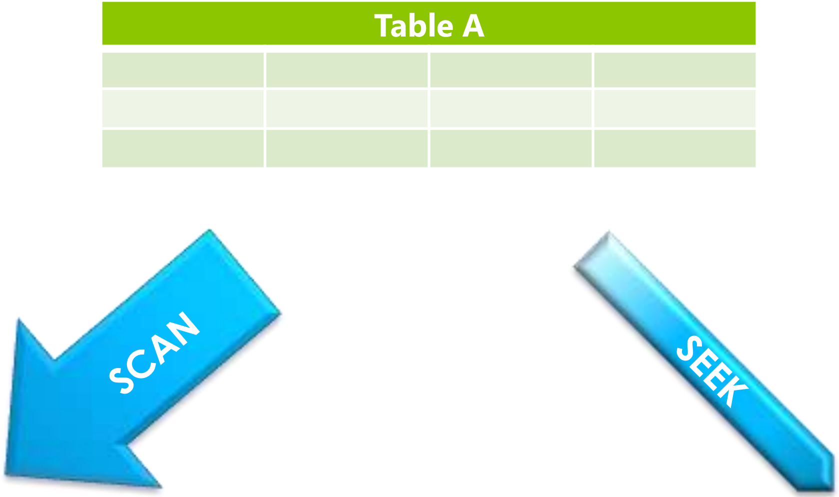
- Clustered Index
  - Primary key default (but not necessary)
  - Data is stored at the leaf level
  - Data is ordered by the key
- Non-clustered Index
  - Uses cluster key or RID of a heap
  - INCLUDE stored at leaf
- And the rest – outside the scope of this session

Constraints

- Primary Key
  - Cluster by default, but doesn’t have to be
  - Always an index
- Foreign Key
  - No indexes are created with this constraint
  - Be sure you use WITH CHECK
- Unique Constraint
  - This constraint is an index

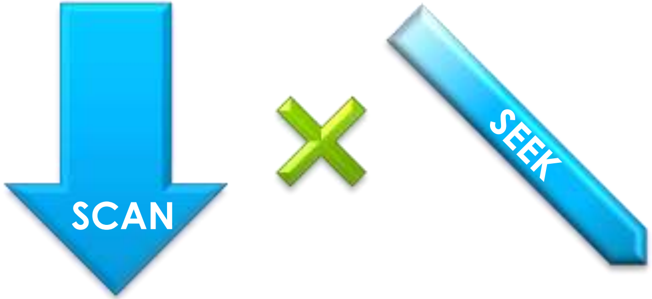
What's All This Mean?

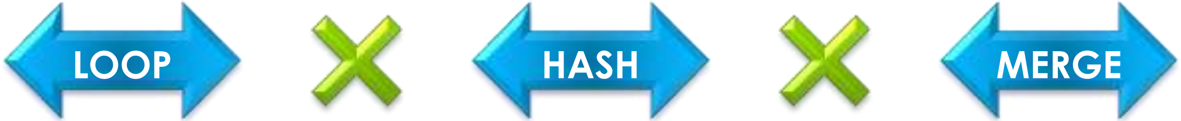
SELECT ID FROM TableA WHERE ID = 42



What's All This Mean?

```
SELECT a.ID, b.Name, c.Value
FROM TableA as a
        JOIN TableB as b
            On a.ID = B.ID
        JOIN TableC as c
            ON b.OtherID = c.OtherID
WHERE a.ID = 42
```





324 Possible Plans

Questions?

How would you… ?

What happens when… ?

Why does…?

When do I… ?

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Optimizer Resources

Dr. Dewitt's Key Note, PASS Summit 2010
http://www.facebook.com/l.php?u=http%3A%2F%2Fwww.slideshare.net%2FGraySystemsLab%2Fpass-summit-2010-keynote-david-dewitt&h=306f5

"Inside SQL Server 2008 T-SQL Querying" Itzik Ben-Gan

"SQL Server 2012 Internals" Kalen Delaney

"Inside the SQL Server Optimizer" Benjamin Nevarez

Query Performance Tuning in SQL Server

READING EXECUTION PLANS

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Why Execution Plans

What will be accessed

What indexes were used

What kind of joins were used

How much did all these operations cost

Tuning

Troubleshooting

Concepts and Architecture

- Relational Engine
  - Estimated Execution Plan
- Storage Engine
  - Actual Execution Plan
- Optimizer
  - Cost-based
    - Just an estimate
    - Not based on your computer
- Cache
  - Most queries go to cache

What To Look For

First Operator

Warnings

Most Costly Operations

Fat Pipes

Extra Operations

Scans

Graphical Plans

Basic Execution

Join

Update

Delete

Insert

Sub-select

Views

XML Plans

Every Graphical Plan is XML

All cached plans are XML

Text plans show less information

Execution Plans LIVE

Sys.dm_exec_query_profiles (2014)

Execution Plan Resources

SQL Server Execution Plans

Microsoft Whitepapers and Web Sites

Statistics used by the Query Optimizer
http://www.microsoft.com/technet/prodtechnol/sql/2005/qrystats.mspx

Compilation and Caching
http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.mspx

Showplan Security
http://technet.microsoft.com/en-us/library/ms189602.aspx

Understanding Joins
http://technet.microsoft.com/en-us/library/ms191426.aspx

Analyzing a Query
http://technet.microsoft.com/en-us/library/ms191227.aspx

Database Engine Developer Info Center
http://technet.microsoft.com/en-us/library/ms191267.aspx

Database Engine Architect Info Center
http://technet.microsoft.com/en-us/library/ms175560.aspx

Forcing Query Plans
http://download.microsoft.com/download/4/7/a/47a548b9-249e-484c-abd7-29f31282b04d/Forcing_Query_Plans.doc

PASS Top 10 Execution Plan Web Sites

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Questions?

How would you… ?

What happens when… ?

Why does…?

When do I… ?

Query Performance Tuning in SQL Server

IDENTIFYING AND FIXING COMMON PROBLEMS

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Query Tuning Methods

Identify the query to be tuned

Configure the server

Design the database

Maintenance

Design the T-SQL

Configure the Server

Memory Configuration

Cost Threshold for Parallelism

Max Degree of Parallelism

Optimize for Ad Hoc Workloads

File layout

Compression

Design the Database

Balance under and over-normalization

Use entity-integrity constraints

Use domain and referential constraints

Adopt indexing best practices

Minimize the use of triggers

Partitioning as necessary (primarily for data management)

Maintenance

Keep statistics up to date

Minimize Index fragmentation

Design the T-SQL

Define the owners of objects explicitly

Don’t use nonsargable search conditions

Try not to use operations and functions on WHERE & JOIN columns

Avoid optimizer hints

Stay away from nesting views

Ensure there are no implicit data type conversions

Minimize logging overhead

Adopt best practices for reusing execution plans

Eliminate or reduce the overhead of cursors

Adopt best practices for database transactions

Tune the Query

Small to medium, look at the query first

Medium to large, go straight to the execution plan

Very large and insane, query the execution plan

Watch for low-hanging fruit

Fix syntax over stats

Stats over indexing

- Indexing over restructuring
  - Restructuring

Read the execution plan

Understand the business needs

Common Problems

Slow Running Query

Key Lookup

Parameter Sniffing

Index Use

Table Valued User Defined Functions

Triggers

Other Ways to Get Them

Slow Running Query

Description

- Slow running query
- Expensive to run query
- The query the boss notices

Indications

- The query is slow

Solutions

- Fix it

Key Lookup

Description

AKA Bookmark Lookup

Not necessarily a problem

Indications

Key Lookup Operator and a Join

Solutions

Change Query

Change the index

INCLUDE

Bad Parameter Sniffing

Description

In general, parameter sniffing is a good thing

Depends on the data distribution and parameters used

Indications

Intermittent poor performance

Disparity on estimated & actual rows

Different execution plans at different times

Solutions

OPTIMIZE FOR query hint

Use local variables

Last resort – RECOMPILE query hint

Last last resort – Plan Guides

Seriously don’t go there last resort – turn parameter sniffing off

Index Use

Descriptions

Just because you see the index name, doesn’t mean it’s getting used properly

Scans are not necessarily bad

Seeks are not necessarily good

Indications

Table Scan

Index Scan

Extra operators like table spool or sort

Solutions

Create an index

Modify an index

Modify the query

Multi-Statement Table Valued User Defined Functions

Description

Yes, I see it. It says 0%. It’s a lie.

“One row is a tragedy; one million is a statistic. ” Joseph Stalin (sort of)

Indications

Table Scan with a cost of 0%

Or Table Valued Function with a cost of 0%

Solutions

When working with more than a few rows… don’t use them

Triggers

Description

Triggers are not immediately visible

Estimated plan won't display

Slow performance from query that shouldn't be

Querying from optimizer…TEST TEST TEST this

Indications

Second plan with the actual plan

No hint of it in the estimated plan

Solutions

Be sure the trigger is optimized

Avoid where possible

Individual Statement is Slow

Large queries or lots of queries

The exact execution plan you want may be hard to find

SHOWPLAN_XML - Estimated

STATISTICS XML - Actual

Query is Sometimes Slow

Intermittent behavior is hard to catch

Profiler

Not the gui

Server-side trace

Even with a server-side trace, capturing execution plans is more expensive (primarily disk space), exercise restraint

Data size increase from 2k to 64k for an XML Plan per statement

Added overhead for storage and processing

Query Was Slow Earlier Today

Knowing that the query is in cache is the key

Once it’s in cache, DMV’s are your friend

- sys.dm_exec_cached_plans
- sys.dm_exec_query_plan
  - Really large plans won’t be stored here
- sys.dm_exec_query_stats
- sys.dm_exec_plan_attributes
- sys.dm_exec_sql_text
- sys.dm_exec_text_query_plan
  - Used for really large plans

Identifying Similar Queries

Ad hoc systems need hugs/tuning too

Identifying similar queries can suggest needed indexes

Similar queries could be candidates for procedures

Multiple stored procedures may have same query

Query Hash to see similarities in query

Query Plan Hash to see similarities in query plan

Working With Large Plans

Really large plans are hard to read

Large plans in text

Large plans in XML

In XML, XQuery opens up the plan

Using XML has other benefits

Hints

Are you smarter than these guys?

Have you spent more time working on SQL Server internals than these guys?

Then why do you think you should take control of the optimizer?

Query Hints

Unions

Joins

FORCE ORDER

MAXDOP

OPTIMIZE FOR

ROBUST PLAN

KEEPFIXED PLAN

Join Hints

Loop

Merge

Hash

Table Hints

NOEXPAND

INDEX()

FAST N

Plan Guides

- For Use When You Can’t Modify Code
- Three Kinds
  - Object
  - SQL
  - Template
- Applies Hints

Plan Forcing

USE PLAN

As close as you can get to direct control of the Optimizer

Still can’t actually control it

Absolute Last Ditch Efforts

Limits:

- Must be a valid plan
- No INSERT, UPDATE, DELETE
- No distributed or full text queries
- Cursors can only be static or fast_forward

Azure & Virtual Machines

The same

Except where it’s different

Configure the Server

Memory Configuration

Cost Threshold for Parallelism

Max Degree of Parallelism

Optimize for Ad Hoc Workloads

File layout

Compression

Design the Database

Balance under and over-normalization

Use entity-integrity constraints

Use domain and referential constraints

Adopt indexing best practices

Minimize the use of triggers

Maintenance

Keep statistics up to date

Minimize Index fragmentation

Design the T-SQL

Define the owners of objects explicitly

Don’t use nonsargable search conditions

Try not to use operations and functions on WHERE & JOIN columns

Avoid optimizer hints

Stay away from nesting views

Ensure there are no implicit data type conversions

Minimize logging overhead

Adopt best practices for reusing execution plans

Eliminate or reduce the overhead of cursors

Adopt best practices for database transactions

Query Tuning Methods

Identify the query to be tuned

Configure the server

Design the database

Maintenance

Design the T-SQL

Questions?

How would you… ?

What happens when… ?

Why does…?

When do I… ?

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Query Performance Tuning in SQL Server

NEW FUNCTIONALITY

Performance Functionality

Columnstore Indexes

In-Memory Tables

Compiled stored procedures

Columnstore Index

Specific Uses

Aggregation

Pivots

Warehouse style storage

Restrictions

No LOB

No CLR

No sparse columns

Clustered column store only one on the table

No constraints on clustered column store

Nonclustered is not updateable

Columnstore Index

Clustered is updateable in 2014

Two modes

- Row
- Batch

Execution plans are useful

No order required

In-Memory Tables

Specific Uses

OLTP

To reduce latches

Improve data collection

Restrictions

No LOB

No CLR

No user defined types

No VARIANT

No ROWVERSION

No foreign keys

Must have index

Durable tables must have a primary key

In-Memory Tables

Queries can be combined with standard tables

No cross-database queries

Generate execution plans

Up to 8 indexes at the same time

Durability

- Schema only
- Schema and data

Still persists to disk

In-Memory Indexes - Hash

Hash

No B-tree

Must define hash buckets

- Not too large
- Not too small
- Err on too large

Point lookups are VERY fast

Scans are VERY not

Hash collisions

No more than five values recommended

In-Memory Indexes – Nonclustered

B-tree

Pointers to data store

No reverse order

In-Memory Indexes - Maintenance

UPDATE STATISTICS

- Must use FULLSCAN and RESAMPLE

No DBCC SHOW_STATISTICS

Compiled Stored Procedures

Compiles to DLL

Runs within SQL Server executable

In-memory tables only

Must be an Atomic operation

All succeed or all rollback

No NULL parameters

Must have schema binding

Estimated plans only

Questions?

How would you... ?

What happens when… ?

Why does…?

When do I… ?

Goals

An understanding of how the optimizer works in support of writing better TSQL code as well as troubleshooting poorly performing queries

The ability to generate, read, and understand execution plans from multiple sources in support of troubleshooting poorly performing queries

Knowledge enabling you to identify and address common query performance problems

Get in touch

blog scarydba.com

email grant@scarydba.com

twitter @gfritchey

Explore Everything PASS Has to Offer

Free SQL Server and BI Web Events

Free 1-day Training Events

Regional Event

This is Community

Business Analytics Training

Local User Groups Around the World

Session Recordings

PASS Newsletter

Free Online Technical Training