SQL Services Ltd
www.sqlservices.com

SQL Services Ltd

- NZ based company, 4 years operation.
- Remote DBA Support core business, 24*7 support centre in Nelson, NZ.
- Focused entirely on SQL Server with highly experienced and certified staff.
- Strong process, systems & embedded IP.
- Strong track record with corporate customers.
- Premier provider SQL Server support & consulting.
- Delivery primarily through partners Computerland, HP, Microsoft.
- Partner and customer portals support business.

Overview of Customers

- University of Auckland, Sky City Casino, Beca Carter, Progressive Enterprises.
- IAG (NRMA), Meridian Energy, HP, Industry NZ, Dept of Conservation, Computerland NZ.
- NZ Dairy Foods, Origin Pacific Airways, Museum of NZ, Lincoln University, Scenic Circle Hotels,
- 7 City Councils nationwide.
- A range of Health providers.
- A number of international customers: Fiji, Australia and USA.

SQL Services in Australia

- Entering Australia second half 2003.
- SIG road show 6 cities.
- Presenting and exhibiting TechEd Sydney.
- Launching more formally in September.
- DBA Services delivered from NZ base.
- Consultants available on demand.
- Colin Andersen driving Business development initially.

The Query Execution Plan

www.sqlservices.com

SQL Services Ltd

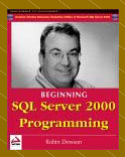
- Microsoft SQL Server only
- World leading remote SQL Server Administration and Support
- Highly skilled DBA and SQL Server Consultant teams.
- Since SQL Services began
  - 16 million database backups (est)
  - 15 billion transactions (est)
  - 0 clients unrecoverable

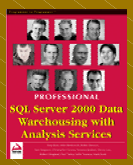
www.sqlservices.com

Who Am I?

www.tonybain.com

- Principal Consultant
- 7+ years experience with ALL areas of SQL Server
  - Database Administration / Performance Tuning
  - Development / Design
  - Data Warehousing / Business Intelligence / ETL
- SQL Server MVP, MCSE, MCDBA, MCSD
- Author:



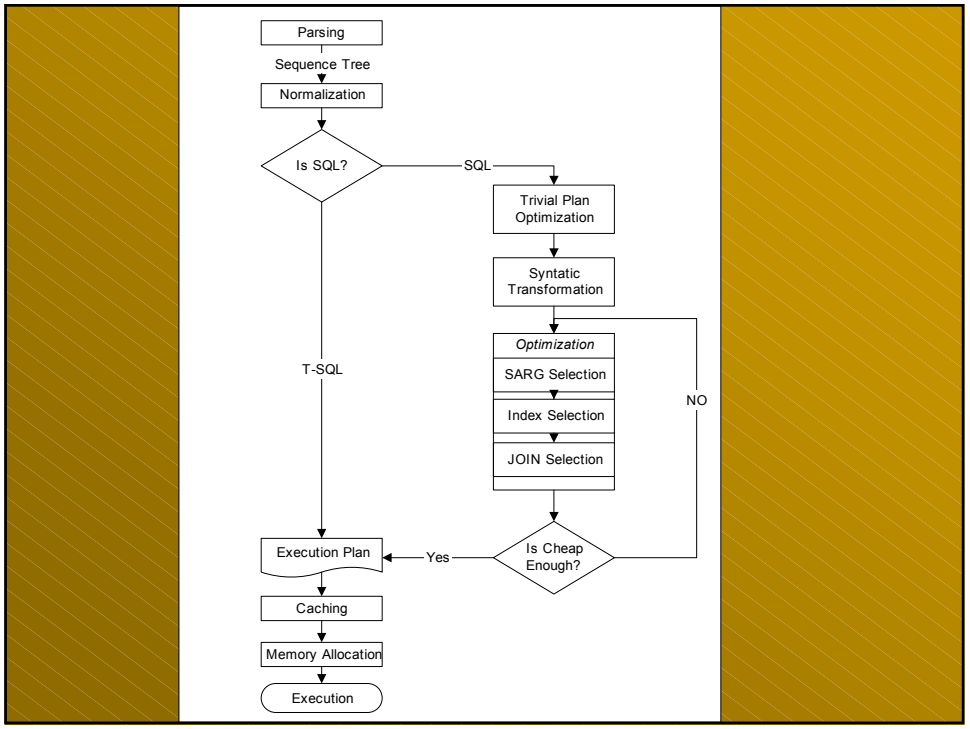


Coming Soon

Coming Soon

Agenda

- Execution Plan preparation steps
  - Compilation
    - Parsing
    - Normalization
  - Optimization
    - SARG determination
    - Index selection
    - JOIN strategy
  - Caching
  - Execution
    - Memory Allocation
    - Go!



What is the Execution Plan?

- The steps necessary to execute each batch or stored procedure
- Includes
  - Which indexes are used
  - What JOIN strategies are used
  - What grouping and ordering strategies are used

Execution Plan Performance Tuning Tip

"Performance tuning SQL Statements involves doing things to allow the optimizer make better decisions"

Viewing The Execution Plan

- Query Analyzer
  - Show Graphical Execution Plan
  - Show Estimated Execution Plan
  - SET SHOWPLAN_TEXT
  - SET SHOWPLAN_ALL
- SQL Profiler
  - Performance: Show Plan Text
  - Performance: Show Plan All
  - Performance: Show Plan Statistics

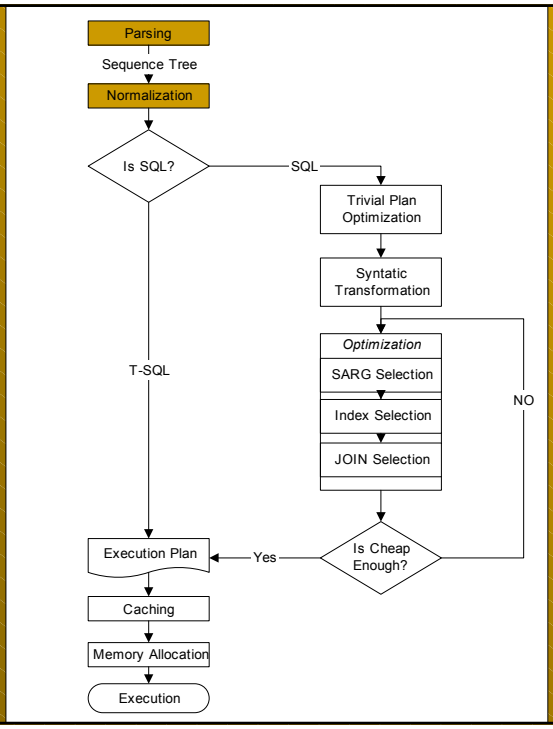
Execution Plan Performance Tuning Tip

The Execution Plan for a Batch or Stored Procedure shows:

- **Cost of Query Relative to Batch**
- **Cost of Operator Relative To Query**

Use these measures to quickly narrow down what is causing performance problem.

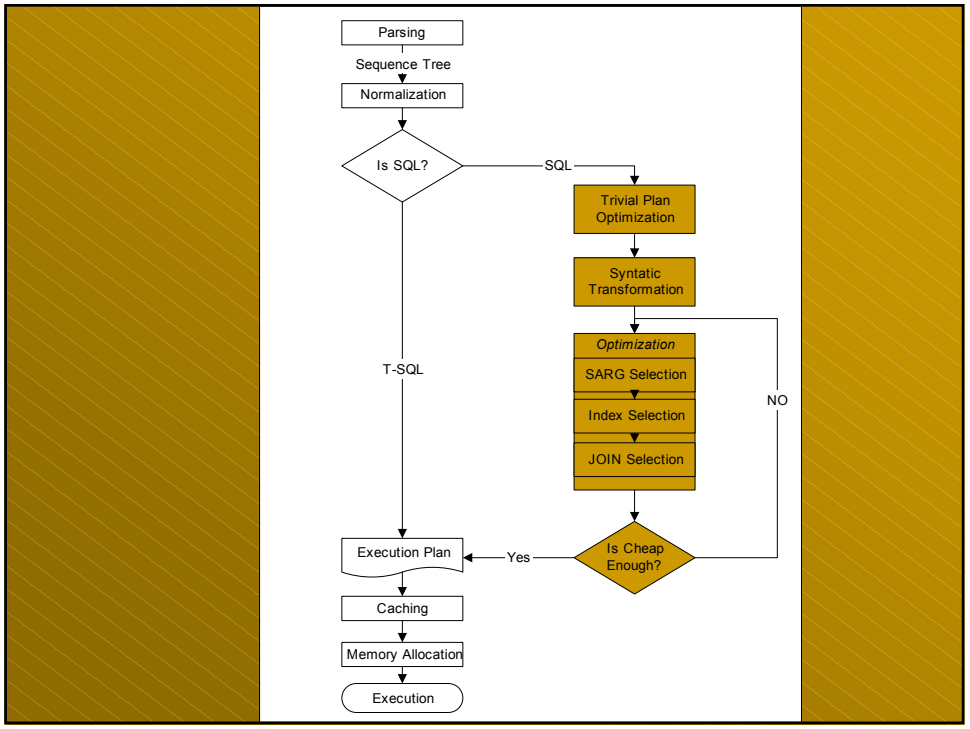
www.sqlservices.com



Compilation

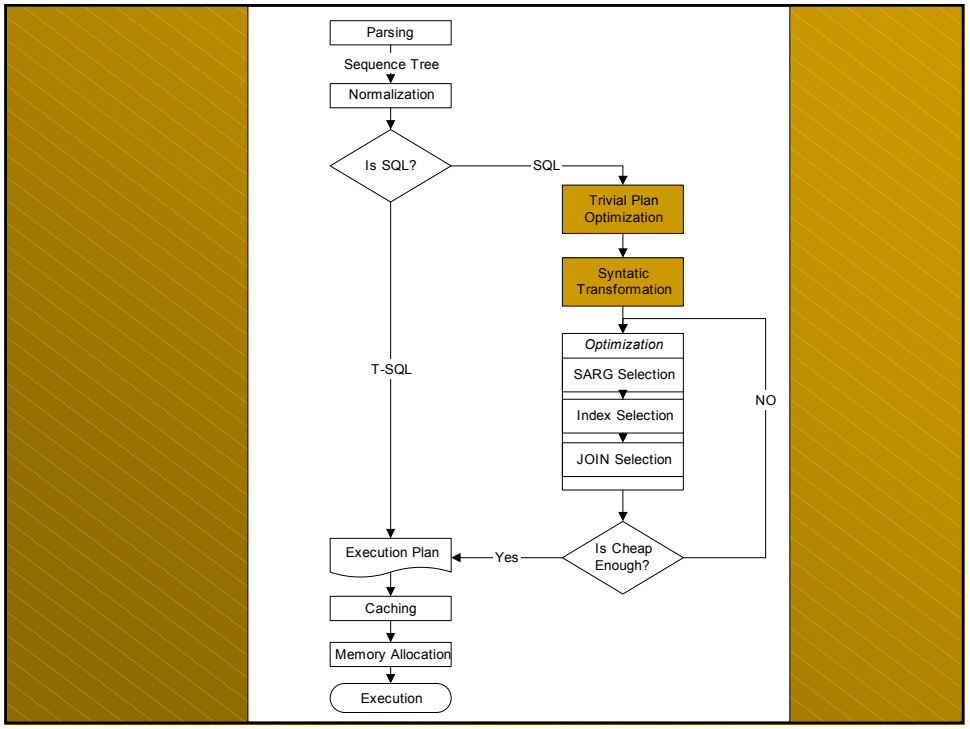
- Parsing
  - Parsing and validating syntax
  - Turning statements into compiler ready structures
  - Production of sequence tree
- Normalization
  - Checks characteristics of objects to ensure they make sense
  - Object binding
  - Implicit conversions
  - Definitions are replaced

EXAMPLE



Optimization

- Only SELECT, INSERT, UPDATE and DELETE Statements can be optimized
- Optimizer is a set of rules that tries out index and join strategies
- Uses *Cost Based* optimization
- Query optimization is probability based which means conclusions can be wrong
- Because cost based optimization is expensive, may try a few things first



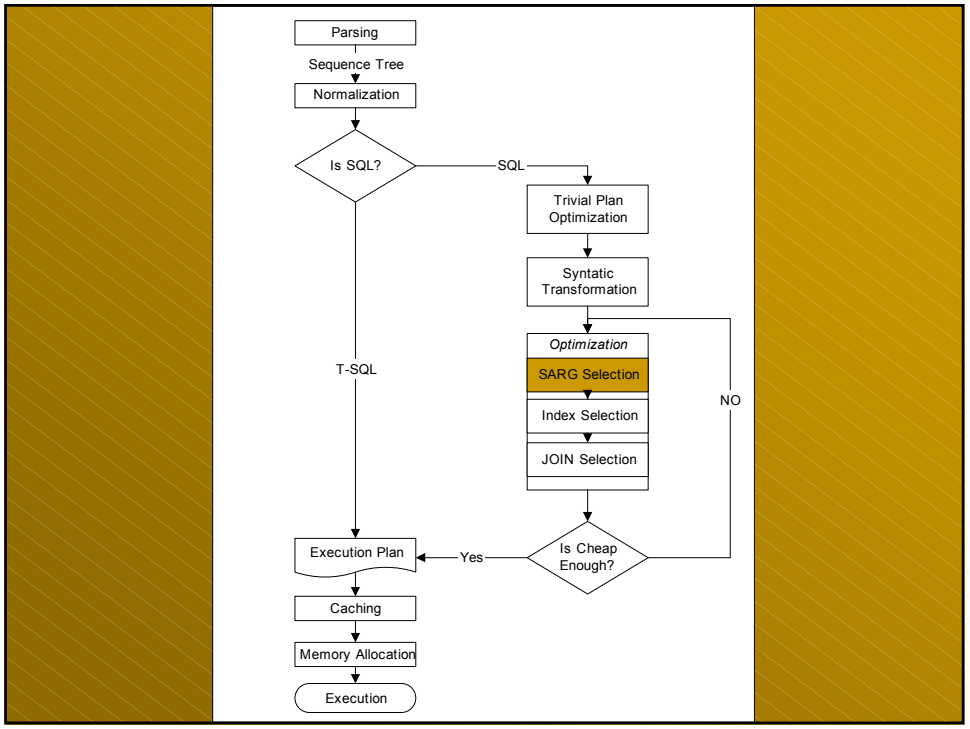
Optimization
Pre Optimization Steps

- Trivial Plan Optimization
  - If SQL Server knows that there is only one possible plan it can avoid a lot of work.
- Syntactic Transformation
  - Looks for properties and operations that can be rearranged
  - SQL Server can rearrange things that don't require looking at cost or indexes

EXAMPLE

Optimization How It Works

- Phased Approach
  - Each phase is a set of rules
  - After each phase checks if the plan is “cheap enough”
  - If not then uses the next optimization phase
- SQL Server usually finds a “cheap enough” plan in an early phase
- Optimizer chooses plan that will return results to the user the quickest with a reasonable resource cost.
- If all phases have completed checks results of all phases found so far. If above threshold then will try an produce a parallel plan.



Optimization SARGS

- Each clause (predicate) is checked to see if it is useful in reducing the number of rows
- SARG's limit the number of rows using either: = BETWEEN, >, <, LIKE'x%'
- Not SARAGABLE
  - LIKE '%x', NOT LIKE, FUNCTION(*column*)
- AND creates a single SARG
- OR creates multiple SARG's

SARG Selection

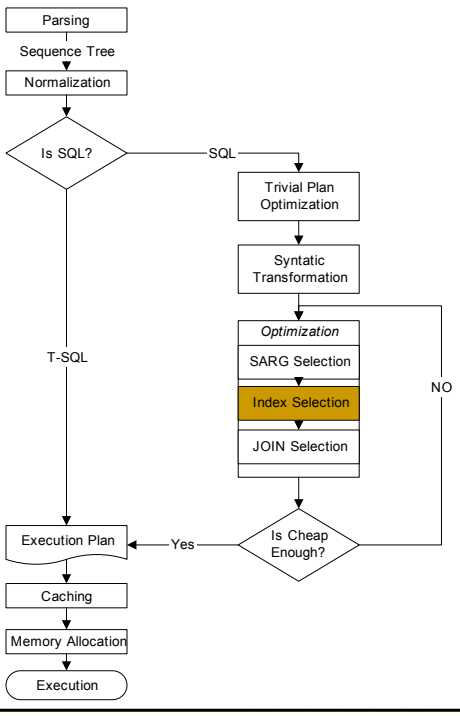
```
SELECT  *
FROM dbo.Products p
INNER JOIN dbo.[Order Details] od ON p.ProductID = od.ProductID
WHERE p.ProductID BETWEEN 10 AND 20
AND p.CategoryID = 4
OR p.UnitPrice=10
AND od.Quantity>10
AND p.ProductName LIKE '%co%'
OR p.ProductName NOT LIKE 'Chocolade'
```


SARG Selection

```
SELECT  *
FROM dbo.Products p
INNER JOIN dbo.[Order Details] od ON p.ProductID = od.ProductID
WHERE p.ProductID BETWEEN 10 AND 20
AND p.CategoryID = 4
OR p.UnitPrice=10
AND od.Quantity>10
AND p.ProductName LIKE '%co%'
OR p.ProductName NOT LIKE 'Chocolade'
```

EXAMPLE

www.sqlservices.com



Optimization
Index Analysis

- Determines if index(s) exists for a SARG
- Index is only useful if the first column of an index is a SARG.
- Assesses the selectivity of the Index using statistics
- Estimates the cost of finding rows within the index
- Checks upstream index column usage to determine if covering

SARG Selection

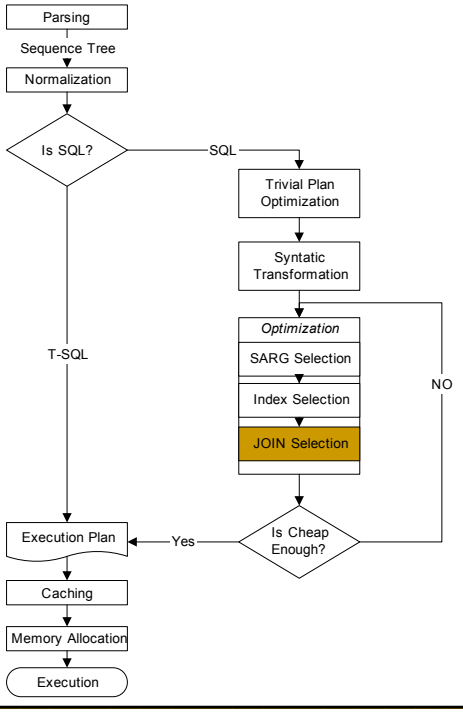
```
SELECT  *
FROM dbo.Products p
INNER JOIN dbo.[Order Details] od ON p.ProductID = od.ProductID
WHERE p.ProductID BETWEEN 10 AND 20
AND p.CategoryID = 4
OR p.UnitPrice=10
AND od.Quantity>10
AND p.ProductName LIKE '%co%'
OR p.ProductName NOT LIKE 'Chocolade'
```

Execution Plan Performance Tuning Tip

Indexes are used to reduce the number of rows the optimizer deals with as early as possible in the query plan.

Indexes offer little benefit if you are not reducing the number of rows significantly with your query SARGS.

www.sqlservices.com



Optimization
JOIN Selection

- Selecting the type of JOIN's is a major step in optimization
- Evaluates
  - Expected number of logical I/O;s (reads)
  - Amount of memory required by JOIN
- JOIN Types
  - **NESTED LOOP**
  - **MERGE**
  - **HASH**
- Join Order is also important
- Evaluates the combination of JOIN strategies, JOIN order and SARG Indexes and selects strategy with lowest cost

Optimization
JOIN Selection
Nested Loop

- A set of loops that take a row from the first JOIN input and compare it with every row from the second JOIN input.
- Usually considered when you have one smaller table and one larger table
- Smaller JOIN inputs are always in the OUTER loop.

Optimization
JOIN Selection
Merge

- Takes two pre-sorted lists and merges them together
- Usually selected when the JOIN inputs are already sorted
- Use little I/O if already sorted
- Builds an in memory structure

Optimization
JOIN Selection
Hash

- Hashing is basically taking one JOIN input and dividing it into buckets based on some property
- The other JOIN input is divided up using the same property and then JOIN criteria comparison only need take place in the relevant bucket.
- Used when NO USEFUL index exists on either JOIN input and dealing with a large number of rows

Execution Plan Performance Tuning Tip

HASH Joins are used when no useful index exists on one or both of the JOIN inputs.

These can be converted to MERGE or LOOP joins through effective indexing.

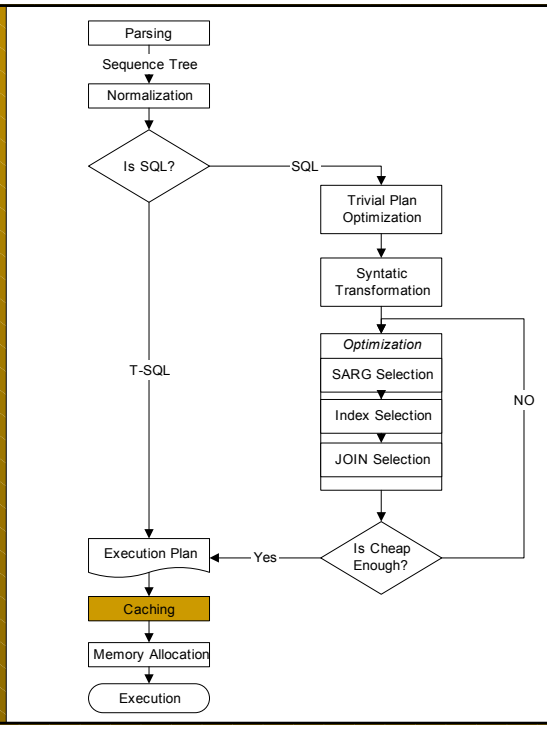
EXAMPLE

Execution Plan Performance Tuning Tip

Nested Loops are the most efficient JOIN type but require a small number of rows on one of the JOIN inputs.

If you are finding that number of rows being reduced after the JOIN operator you may have the opportunity to improve performance through indexing

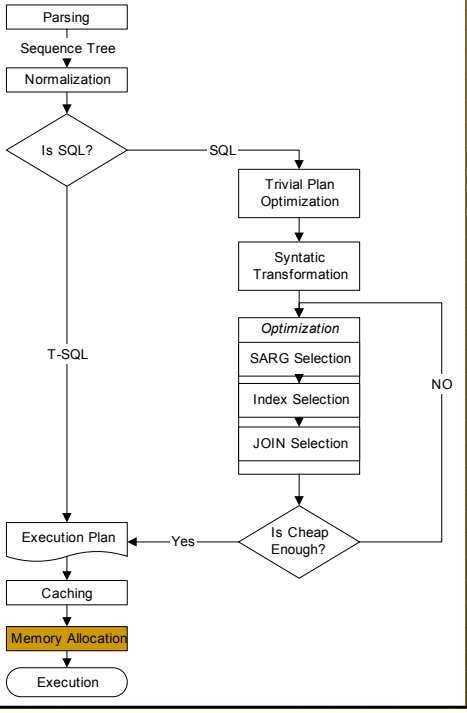
www.sqlservices.com



Caching

- Once query plan has been produced goes into the procedure cache
- Every time batch/sp is execute SQL Server checks to see if plan is in cache and if plan needs to be recompiled
- Plan aging is handled by the Lazywriter
  - Cost is de-incremented, reset when executed
- Can view what is in cache but examining master.dbo.syscacheobjects

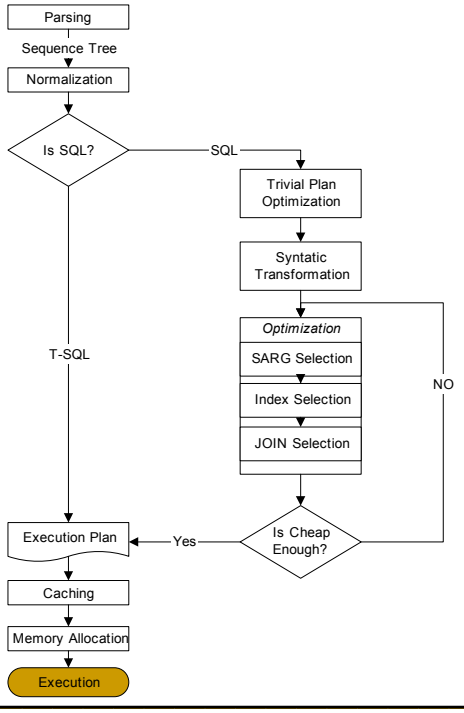
www.sqlservices.com



Execution Memory Allocation

- Plan passed to memory grant scheduler
- This picks the minimum & maximum amount of memory need to execute the plan (stored in plan)
- If no hash,merge or sort required then query is passed through for execution
- If at least 50% of the minimum memory required is available the query will be passed through for execution
- Otherwise it will wait up to 25 seconds (error 8645)

www.sqlservices.com



Execution

- Plan is opened
- Plan runs to completion
- Everybody is happy

Summary

- Execution Plan preparation steps
  - Compilation
  - Optimization
    - SARG determination
    - Index selection
    - JOIN strategy
  - Caching
  - Execution
    - Memory Allocation

Performance Tuning Tip Summary

- At a minimum use the execution plan to isolate what is costing the most
- Your options for performance tuning are rewriting, indexing
- Understand what SARGS you are targeting with an index
- HASH JOIN's are used when no useful indexes exist.

Thanks For Coming
Please Ask Questions

For More Info:
www.sqlservices.com