Advanced Programming and Query Tuning in SQL Server

Course Syllabus

1. Description

This course takes programming and query tuning in SQL Server from the basic level to the advanced, providing students with tools and techniques that will allow them to write code for a broad range of requirements while optimizing performance, and on the other hand, optimize performance of existing SQL Server queries and processes. Throughout the course, the students will analyze different case studies while comparing several approaches for each one and choosing the best solutions.

The course is based on SQL Server 2014, but it is relevant also for previous versions of SQL Server (2005/2008/2008R2/2012).

2. Target Audience

The course is intended for SQL Server DBAs and developers who are responsible for either writing Transact-SQL queries or tuning already written Transact-SQL queries. Students should be familiar with basic programming and with the syntax of Transact-SQL.

3. Objectives

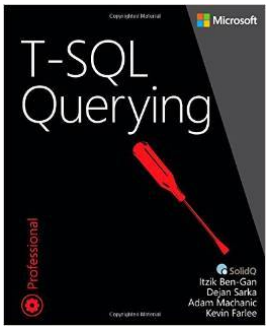
- Understand the various data structures and data types within SQL Server
- Be able to design and utilize indexes and statistics efficiently
- Learn best practices for using programming objects in SQL Server
- Understand how the query processor works
- Learn to write efficient Transact-SQL code using advanced techniques
- Understand how to read and analyze execution plans
- Acquire techniques for efficient query tuning

4. Schedule

- 5 days
- 40 academic hours

5. Materials

- Disk-on-key with all the course materials:
  - ✓ Presentations
  - ✓ Demonstrations
  - ✓ Class exercises & solutions
  - ✓ Homework
  - ✓ Additional resources
- DVD with SQL Server 2014 RTM Evaluation Edition
- Course Book: T-SQL QUERYING by Itzik Ben-Gan



6. Outline

a. Course Introduction

1) About the Instructor
2) About the Students
3) Course Objectives
4) Course Schedule
5) Course Materials
6) Course Outline

b. Data Structures

1) File and Filegroup Organization
   - Pages and Extents
   - The Transaction Log

- Design Best Practices

2) Table and Index Organization
   - Heap vs. Clustered Table
   - Non-Clustered Indexes
   - Partitions
   - Allocation Units
3) Data Types
   - System Data Types
   - Special System Data Types
   - Row Identifiers
   - Spatial Data Types
   - Sparse Columns
   - FILESTREAM
4) Tables
   - Table Variables and Table Types
   - Temporary Tables vs. Table Variables
   - Common Table Expressions
   - Partitioning
   - FileTables
5) Practice

c. Understanding Indexes

1) Index Types
   - Clustered vs. Non-Clustered
   - Unique vs. Non-Unique
   - Composite Indexes
   - Covering Indexes
   - Filtered Indexes
2) Missing Indexes
3) The Database Engine Tuning Advisor
4) Guidelines for Writing Efficient Queries
5) Index Maintenance
   - Fragmentation Types

- Page Splits
- Fill Factor
- Detecting Fragmentation
- Rebuild and Reorganize Operations
- Online Indexing Operations

6) Practice

d. Understanding Statistics

1) Statistics Types
   - Column vs. Index Statistics
   - Multiple Column Statistics
   - Statistics on Computed Columns
   - String Summary Statistics
2) Statistics Maintenance
   - Automatic vs. Manual
   - Synchronous vs. Asynchronous
   - Sample vs. Full Scan
3) Practice

e. Programming Objects Best Practices

1) Views
   - Updatable Views
   - Using Instead-Of Triggers with Views
   - Indexed Views
   - Partitioned Views
2) User-Defined Functions
   - Function Types
   - Joining with APPLY
   - Performance Considerations
3) Stored Procedures
   - Parameters, Returned Value and Result Sets
   - Table-Valued Parameters
   - Error Handling
4) Triggers

- DML vs. DDL Triggers
- After vs. Instead-Of Triggers
- Statement Types and Number of Rows Affected
- Trigger Order of Execution
- Nested and Recursive Triggers
- Usage Best Practices

5) Dynamic SQL
- EXECUTE vs. sys.sp_executesql
- SQL Injection
- Usage Best Practices

6) Practice

f. Query Processor Internals

1) The Plan Cache
2) Compilation-Execution Sequence
3) Recompilations
4) Query Logical Simplification
5) Parameterization
- Simple vs. Forced
- Skewed Data Distribution
- Parameters vs. Local Variables
- Changing Parameter Values

6) Practice

g. Understanding Execution Plans

1) Execution Plan Representation
- Graphical
- XML
- Text
- Using Profiler
- Using Extended Events

2) Logical and Physical Operators
- Scan vs. Seek Operations
- Join Physical Operations

- Aggregate Physical Operations
- Other Operators

3) Cost and Cardinality
   - Cost Measures
   - Operator Relative Cost
   - Statement Relative Cost
   - Number of Rows and Row Size
4) Estimated vs. Actual Execution Plan
5) Practice

h. Transactions and Locks

1) Transactions Overview
2) Lock Types
3) Concurrency Issues
4) Transaction Isolation Levels
5) Locking Hints
6) Deadlocks
7) Nesting Transactions
8) Practice

i. Advanced Programming Techniques

1) Ranking Functions and Window Functions
2) Implementing Query Paging
3) Grouping Sets
4) The MERGE Statement
5) Generating Random Values
6) COUNT (DISTINCT) vs. MIN/MAX
7) TOP vs. MIN/MAX
8) Practice

j. Advanced Query Tuning

1) <u>Case Study #1 – Analyzing Execution Plans</u>

This case study involves a relatively simple query, which performs a clustered index scan on one of the tables because an index is missing. The purpose of this case study is to learn how to read and analyze execution plans.

2) **Case Study #2 + #3 – Search Arguments**

This case studies introduce queries with search arguments that aren't optimized for index use. The goal of the case studies is to learn how to rewrite search arguments in order to make use of the appropriate indexes.

3) **Case Study #4 – Implicit Conversions**

This case study demonstrates the impact of implicit conversions on query performance. It introduces a query that performs a join between two tables based on incompatible data types. The goal of the case study is to understand the importance of using compatible data types and to learn how to solve this problem.

4) **Case Study #5 – Using Covering Indexes**

In this case study the execution plan includes a clustered index scan operation instead of an index seek operation, although an index exists for the relevant column. The goal of this case study is to understand the decision making process of the optimizer based on statistics and using covering indexes in order to eliminate the need to perform lookup operations.

5) **Case Study #6 – Temporary Tables vs. Table Variables**

This case study uses a table variable and demonstrates the main problem with table variables, which is the lack of statistics. The case study demonstrates the dramatic impact of using a temporary table instead of a table variable.

6) **Case Study #7 – Divide and Rule**

This case study involves a very complex query with a large amount of join operations. It demonstrates how the optimizer gets lost in such cases. The goal of this case study is to learn how to split the query into multiple smaller queries and store intermediate results in temporary tables.

7) **Case Study #8 – (Not) Using Cursors**

This case study demonstrates the use of a cursor and how the same logic can be rewritten without using a cursor. The goal of the case study is to demonstrate how a cursor can be rewritten, even when it doesn't seem possible at first sight, and how performance can be dramatically impacted by this change.

8) **Case Study #9 – (Not) Using Scalar Functions**

This case study involves the use and reuse of scalar functions inside a query. It demonstrates that although using functions is a good practice in terms of modular programming, it is a terrible practice in terms of performance. The goal of the case study is to learn how to optimize a query by merging the logic inside scalar functions into the query itself.

9) **Case Study #10 – Skewed Data Distribution**

This case study demonstrates the drawback of parameter sniffing when the data is not distributed evenly. The goal of this case study is to demonstrate several solutions to this problem, such as recompiling the query on each execution or using the OPTIMIZE FOR query hint.

10) **Case Study #11 – Local Variables**

This case study demonstrates the difference between local variables and parameters, and shows that there is no parameter sniffing associated with local variables. The purpose of this case study is to understand the problems of working with local variables instead of parameters and learn how to solve these problems.

11) **Case Study #12 – Ranking Functions**

This case study introduces a requirement and a poorly performing query to solve it. The goal of the case study is to learn about the ranking functions and to rewrite the query using them in order to improve the performance of the query.

12) **Case Study #13 – Transaction Log Overhead**

This case study demonstrates the overhead of writing to the transaction log and how this overhead can dramatically influence query performance in some cases. The goal of this case study is to learn how transaction log overhead can be manipulated by controlling transaction size and by utilizing minimal logging.

13) **Case Study #14 – LOB Data**

This case study involves querying a table with a high volume of LOB data. The goal of this case study is to learn how to manage and use LOB data appropriately in order to improve query performance.

14) **Case Study #15 – Manipulating Hierarchies**

This case study demonstrates a query on a hierarchical structure using a recursive approach in the traditional design of parent-child relationship. The goal of this case study is to introduce the

HIERARCHYID data type and to learn how to use it in order to manipulate hierarchies much more efficiently.

15) **Case Study #16 – Row Identifiers**

This case study presents the use of the UNIQUEIDENTIFIER data type as a row identifier and demonstrates the impact of this approach on index fragmentation and on query performance. The goal of this case study is to compare the various approaches to choosing row identifiers, including the new sequence object in SQL Server 2012.

16) **Case Study #17 + #18 – Manipulating Statistics**

This case study demonstrates a query that retrieves only recent data from a very large table, but due to outdated statistics, it suffers from poor performance. This case study serves as an opportunity to learn about statistics, including filtered statistics, which are used to solve the problem in this case.

17) **Case Study #19 – Querying XML**

This case study demonstrates how to query an XML document using the OPENXML function. The goal of this case study is to introduce the "nodes" method of the XML data type and to compare between the two methods for querying XML.

18) **Case Study #20 – Locking and Blocking**

This case study involves a query that uses the NOLOCK hint, and demonstrates the possible impact of this approach on data integrity. The case study serves as an opportunity to learn about the various isolation levels, including "Read-Committed Snapshot" and "Snapshot", and to discuss some locking and blocking best practices.

19) **Case Study #21 – Nested Transactions**

This case study introduces a problem associated with nested transactions. The goal of this case study is to understand the behavior of nested transactions and to learn about the proper method of using them.

20) **More Case Studies**

As time permits, we will analyze more case studies that involve any of the issues learned in the course or even a combination of multiple issues.

k. Summary

1) Course Summary
2) Additional Resources
3) Feedback