SQL Server Optimization and Performance Tuning

As a database grows both in size and the number of transactions it supports, its performance can erode significantly unless it is actively managed. In this course, you will learn how to minimize the risk of performance problems when creating a new database, and to detect emerging performance problems before they create an impact for users. You will also learn effective troubleshooting techniques for resolving chronic performance problems. At the end of the course your skills will expand as you learn how to setup, monitor, tune, and maintain databases to ensure performance.

Who should take this course?

This course is for those with database and reporting experience who want to extend their skills with database tuning and performance. These skills will be immediately useful to those whose job involves database programming, system management, or database design, for example, software development engineers, test engineers, system administrators, program managers, and others.

Course Objectives

• Prevent performance problems caused by index page splits and fragmentation.
• Choose an indexing scheme most suited to optimize performance of reading and writing data.
• Configure automation of tools for maintaining and tuning databases.
• Configure indexes and the query optimizer to achieve the most efficient query execution plan.
• Monitor SQL Server performance using Management Views and SQL Profiler.
• Configure indexing to balance performance issues caused by locking and concurrency.
• Create a database maintenance plan to automate administrative tasks.

Course Details

• Length: 30 hours
• Format: Classroom
• Prerequisites: Beginning SQL Server and Structured Query Language: (T-SQL) Level 2 or equivalent

The above prerequisites are considered to be the basic skills and knowledge needed prior to taking this class. Instructors will assume your readiness for the class materials and will NOT use class time to discuss prerequisite materials.
Course Contents

Prevent performance problems caused by index page splits and fragmentation.

- Determine the extent to which page splits are causing a performance problem.
- Compare the costs and benefits of reducing page splits by means of configuring a fill factor versus using identity fields.
- Choose a fill factor that balances the space it reserves versus performance gained by minimizing page splits.
- Use identity fields on clustered indexes to minimize page splits.
- Identify situations that contraindicate the assignment of a clustered index to a primary key.
- Define the terms: selectivity, heap, covered query, and index padding.
- Use a script or Management Studio to create a clustered index on a new or existing table.
- Configure a fill factor to enable optimal index padding.
- Predict how using an identity field will affect an index.
- Compare the ideal use of a clustered index as a single field or part of a composite index.

Choose an indexing scheme most suited to optimize performance of reading and writing data.

- Compare the use of a clustered index versus a non-clustered index, in terms of: resource requirements, and performance.
- Create a non-clustered index.
- List the kinds of fields that must not contain indexes.
- Select a field performance association by using the INCLUDE keyword.
- Drop a non-clustered index.
- Identify where to put and where not to put clustered indexes for best performance.
Course Contents, continued

**Configure automation of tools for maintaining and tuning databases.**
- Compare the performance effects of fragmentation versus defragmentation.
- Identify situations that indicate using a seek versus a scan.
- Identify situations that indicate tuning versus optimizing.
- Create workload files.
- Create indexing schemes.
- Analyze workload files with Data Tuning Advisor (DTA).
- Run Dynamic Management views to evaluate the performance effect of index usage and index fragmentation.
- Rebuild and reorganize fragmented indexes.
- Maintain and tune indexes with database consistency checks Database Consistence Checker (DBCC’s).

**Configure indexes and the query optimizer to enable the most efficient query execution plan.**
- Create a Query Execution Plan.
- Recognize scenarios where the volume of seeks and scans could account for reduced performance.
- Increase query performance by using Query Optimizer.
- Assess index performance by auditing its statistics.
- Determine when to use SQL Query Analyzer to run a full table scan.
- Compare the affect on performance of running full table scans on smaller tables versus larger tables.
- Determine when to use the Update Statistics command.
- Use Dynamic Management views to monitor index usage statistics.
- Interpret the Result Set returned by a statistical index query.
Course Contents, continued

Monitor SQL Server performance using Management Views and SQL Profiler.

- Monitor performance with SQL Profiler.

Configure indexing to balance performance issues caused by locking and concurrency.

- Identify situations where running the following transactions is indicated: BEGIN TRAN, COMMIT TRAN, and ROLLBACK TRAN.
- Monitor Server Process ID’s (SPIDs).
- Configure monitoring of: Sp_Who / Sp_who2, Management Studio, and kill.
- Define the terms: shared lock, exclusive lock, and deadlocking.
- Change concurrency using locking hints.
- Monitor locks with SQL Profiler filters.

Create a database maintenance plan to automate administrative tasks.

- Create and schedule DBCC jobs.
- Create a maintenance plans to automate job creation.