

Developing High-Performance SQL Server® Databases: Hands-On - 5 Days

Course 535 Overview

- You Will Learn How To**
- Design and implement high-performance databases for SQL Server
 - Create indexes that optimize different types of queries
 - Design transactions that maximize concurrency and minimize contention
 - Interpret the data access plans produced by the query optimizer
 - Minimize I/O by designing efficient physical data structures
 - Analyze and cure performance problems using SQL Server's tools
- Course Benefits** High-availability database systems offer timely access to business-critical data. Microsoft SQL Server offers powerful features to maintain these vital systems. In this course, you acquire an in-depth knowledge of SQL Server's core components—the storage engine, lock manager and query optimizer. Through hands-on exercises, you gain the skills to implement a high-performance SQL Server database solution.
- Who Should Attend** Those responsible for increasing the performance and efficiency of SQL Server databases. Knowledge of SQL at the level of Course 532, "SQL Server Transact-SQL Programming," and a familiarity with logical database design are assumed.
- Hands-On Training** Hands-on exercises provide experience developing high-performance SQL Server databases. Exercises include:
- Monitoring and analyzing performance
 - Developing a performance baseline
 - Setting up a server-side Profiler trace
 - Eliminating extent fragmentation
 - Inspecting procedures in the procedure cache
 - Creating indexes for different query types
 - Improving performance with indexed views
 - Implementing partitioning solutions

Developing High-Performance SQL Server® Databases: Hands-On - 5 Days

Course 535 Outline

Fundamental Concepts

Analyzing performance

- Selecting an appropriate monitoring tool
- Investigating plans with SHOWPLAN_ALL
- Interpreting STATISTICS IO output
- Pinpointing performance problems with aggregated Profiler data

Developing a monitoring plan

- Establishing a performance baseline
- Tracking changes over time
- Creating server-side Profiler traces
- Monitoring SQL Server and the operating system with System Monitor

Managing Storage

Database architecture

- Page and extent allocation
- Controlling data placement with file groups

Defining tables

- Selecting the correct data types
- Specifying text and image locations
- Examining internal page structures

Creating and managing indexes

- Clustered vs. nonclustered
- Defining indexed views
- Analyzing and repairing fragmentation

Memory and Locking

Managing memory

- Buffer pool
- Buffer manager
- Lazywriter
- Checkpoint
- Log writer

Designing transactions

- Consistency vs. concurrency
- Investigating lock types and their compatibility
- Choosing isolation levels
- Designing transactions to limit lock duration
- Resolving contention problems
- Handling deadlock
- Implementing row versioning

Optimizing Queries

Query optimizer architecture

- Phases

- Strategies
- Data access plans
- Auto-parameterization
- Avoiding recompilation of queries

Maintaining up-to-date statistics

- Index vs. column
- Automatic vs. manual
- Full-scan vs. sample

Distinguishing among query types

- Point
- Multipoint
- Range
- Prefix match
- Extremal
- Ordering
- Grouping
- Join

Designing effective indexes

- Relating indexes to query types
- Providing alternate access paths
- Improving join performance
- Increasing sort efficiency
- Reducing I/O with covering indexes
- Implementing sparse indexes
- Getting design advice from built-in tuning tools

Designing a Physical Data Model

Storing summarized data for faster retrieval

- Defining roll-up tables
- Materializing aggregated data with indexed views

Minimizing response time by introducing redundant data

- Maintaining redundant data with triggers
- Enhancing performance with surrogate keys
- Creating a read-only query database

Solving performance problems with partitioning strategies

- Horizontal vs. vertical partitioning
- Partitioning tables
- Defining partitioned views

Managing diverse workloads

- Creating resource pools and workload groups
- Developing a classifier function