|  |  |
| --- | --- |
|  |  |

70-464:

Developing Microsoft SQL Server Databases

The following tables show where changes to Exam 70-464 have been made to include updates that relate to database development and management-related tasks. These changes are effective as of February 18, 2016.

1. **Implement database objects (no change: 30-35%)**

|  |  |
| --- | --- |
| **Tasks currently measured** | **Tasks to be added/removed/changed in February 2016** |
| **Create and alter tables (complex statements)**  Develop an optimal strategy for using temporary objects, including table variables and temporary tables; define alternatives to triggers; define data version control and management; implement @Table and #table appropriately; create calculated columns; implement partitioned tables, schemas, and functions; implement column collation; implement online transaction processing (OLTP) | Revised task:  Create and alter tables  Additional subtask:  implement columnstore and sparse columns |
| **Design, implement, and troubleshoot security**  Implement data control language statements appropriately, troubleshoot connection issues, implement EXECUTE AS statements, implement certificate-based security, create loginless users, define appropriate database roles and permissions, implement contained users, implement cross db ownership chaining, implement schema security, implement server roles | No change |
| **Design the locking granularity level**  Choose the right lock mechanism for a given task; handle deadlocks; design index locking properties; fix locking and blocking issues; analyze a deadlock scenario; design appropriate isolation level, including Microsoft ActiveX data objects defaults; design for locks and lock escalation; design transactions that minimize locking; reduce locking contention; identify bottlenecks in data design; design appropriate concurrency control, such as pessimistic or optimistic | No change |
| **Maintain indexes**  Inspect physical characteristics of indexes and perform index maintenance, identify fragmented indexes, identify unused indexes, implement indexes, defrag/rebuild indexes, set up a maintenance strategy for indexes and statistics, optimize indexes (full, filter index), statistics (full, filter), force or fix queue, when to rebuild versus reorg and index, create a tuning and maintenance strategy for proactive operations | Revised task:  Implement indexes  Removed subtasks:  identify fragmented indexes; defrag/rebuild indexes; set up a maintenance strategy for indexes and statistics; when to rebuild versus reorg and index, create a tuning and maintenance strategy for proactive operations  Revised subtask:  optimize indexes, including full, filter, statistics, and force |
| **Implement data types**  Select appropriate data types, including BLOBs, GUIDs, and spatial data; develop a Common Language Runtime (CLR) data type; implement appropriate use of @Table and #table; implement columnstore and sparse columns; determine values based on implicit and explicit conversions | Revised subtask:  Select appropriate data types, including BLOBs, GUIDs, XML, and spatial data  Removed subtask:  implement columnstore and sparse columns |
| **Create and modify constraints using complex statements**  Create constraints on tables, define constraints, modify constraints according to performance implications, implement cascading deletes, configure constraints for bulk inserts | Revised task:  **Create and modify constraints** |
| **Work with XML data**  Implement XML, such as Query, Input, Output; transform XML data into relational data; retrieve relational data as XML; implement FOR XML; design a strategy to query and modify XML data; implement XML schemas and handling of XML data; import and export XML; return tables from XML data types using XQuery; navigate XML namespaces; implement XML selective indexes | Removed task |

1. **Implement programming objects (decreased: 15-20%)**

|  |  |
| --- | --- |
| **Tasks Currently Measured** | **Tasks to be added/removed/changed in February 2016** |
| **Write automation scripts**  Automate backup testing; automate shrink file; implement scripts that check and maintain indexes; implement scripts that archive data; run a SQL Server Integration Services (SSIS) job; write scripts that check disk space; write scripts that automate backups, including backup to Microsoft Azure Blob Storage Service | Removed task |
| **Design and implement stored procedures**  Create stored procedures and other programmatic objects; implement different types of stored procedure results; create a stored procedure for the data access layer; analyze and rewrite procedures and processes; program stored procedures by using T-SQL and CLR; implement parameters, including table valued, input, and output; implement encryption; implement error handling, including TRY…CATCH; configure appropriate connection settings, design appropriate query paging, including OFFSET and FETCH | Removed subtasks:  Implement encryption; design appropriate query paging, including OFFSET and FETCH |
| **Design T-SQL table-valued and scalar functions**  Modify scripts that use cursors and loops into a SET-based operation, design deterministic and non-deterministic functions | No change |
| **Create, use, and alter user-defined functions (UDFs)**  Implement deterministic or non-deterministic functions; implement CROSS APPLY by using UDFs; implement CLR functions | No change |
| **Create and alter views (complex statements)**  Set up and configure partitioned tables and partitioned views; create indexed views | Revised task:  **Create and alter views** |

1. **Design database objects (increased: 25-30%)**

|  |  |
| --- | --- |
| **Tasks Currently Measured** | **Tasks to be added/removed/changed in February 2016** |
| **Design tables**  Apply data design patterns; develop appropriately normalized and de-normalized SQL tables; design transactions; design views; implement GUID as a clustered index appropriately; design temp tables appropriately, including # vs. @; implement set-based logic; design an encryption strategy; design table partitioning; design a BLOB storage strategy, including filestream and filetable; design tables for In-Memory OLTP | Removed subtask:  implement set-based logic |
| **Design for currency**  Develop a strategy to maximize concurrency; define a locking and concurrency strategy; design a transaction isolation strategy, including server database and session; design triggers for concurrency | No change |
| **Create and alter indexes**  Create indexes and data structures; create filtered indexes; create an indexing strategy, including column store, semantic indexes, and INCLUDE; design indexes and statistics; assess which indexes on a table are likely to be used, given different search arguments (SARG); create indexes that contain included columns; create spatial indexes | Revised task: Design indexes  Revised subtasks:  design indexes and data structures; design filtered indexes; design an indexing strategy, including column store, semantic indexes, and INCLUDE; design statistics; design spatial and XML indexes  Removed subtask:  create indexes that contain included columns |
| **Design data integrity**  Design a table data integrity policy, including checks, primary key, foreign key, uniqueness, XML schema, and nullability; select a primary key | No change |
| **Design for implicit and explicit transactions**  Manage transactions; ensure data integrity by using transactions; manage distributed transaction escalations; design savepoints; design error handling for transactions, including TRY, CATCH, and THROW | No change |

1. **Optimize and troubleshoot queries (increased: 25-30-%)**

|  |  |
| --- | --- |
| **Tasks Currently Measured** | **Tasks to be added/removed/changed in February 2016** |
| **Optimize and tune queries**  Tune a poorly performing query, including avoiding unnecessary data type conversions; identify long-running queries; review and optimize code; analyze execution plans to optimize queries; tune queries using execution plans and Microsoft Database Tuning Advisor (DTA); optimize queries using pivots and common table expressions (CTE); design database layout to optimize queries; implement query hints; tune query workloads; implement recursive CTE; implement full text and semantic search; analyze execution plans; implement plan guides | No change |
| **Troubleshoot and resolve performance problems**  Interpret performance monitor data; integrate performance monitor data with SQL Traces; design an appropriate recovery model; optimize data files; identify and fix transactional replication problems; detect and resolve server failures; identify and troubleshoot data access problems; manage tempdb contention and auto growth; implement Resource Governor; monitor and resolve In-Memory OLTP issues, including merge and garbage collection | Removed subtasks:  design an appropriate recovery model; optimize data files; identify and fix transactional replication problems; detect and resolve server failures; identify and troubleshoot data access problems; manage tempdb contention and auto growth; implement Resource Governor; monitor and resolve In-Memory OLTP issues, including merge and garbage collection |
| **Optimize indexing strategies**  Develop an optimal strategy for clustered indexes; analyze index usage; optimize indexes for workload, including data warehousing and OLTP; generate appropriate indexes and statistics by using INCLUDE columns; create filtered indexes; implement full-text indexing; implement columnstore indexes; optimize online index maintenance | Revised task:  **Optimize indexes** |
| **Capture and analyze execution plans**  Collect and read execution plans, create an index based on an execution plan, batch or split implicit transactions, split large queries, consolidate smaller queries, review and optimize parallel plans | No change |
| **Collect performance and system information**  Monitor performance using Dynamic Management Views, collect output from the Database Engine Tuning Advisor, design Extended Events Sessions, review and interpret Extended Event logs; optimize Extended Event session settings, use Activity Monitor to minimize server impact and determine IO bottlenecks, monitor In-Memory OLTP resources | No change |