DP-800 Exam Prep

Overview

As a candidate for this Microsoft Certification, you should have subject matter expertise in designing and developing AI-enabled database solutions across Microsoft SQL platforms, including Microsoft SQL Server, Azure SQL, and SQL databases in Microsoft Fabric.

You should also have experience writing T-SQL code and developing databases in Microsoft SQL platforms. Plus, you need to be familiar with continuous integration and continuous deployment (CI/CD) practices in GitHub, AI-assisted development tools, and AI concepts, such as embeddings, vectors, and models.

Your responsibilities include:

- Designing and developing database solutions that include both structured and semi-structured data.
- Integrating AI features into modern and highly scalable enterprise applications.
- Securing, optimizing, and deploying database solutions.
- Implementing AI capabilities in database solutions.

You work closely with application developers; database administrators (DBAs); architects; AI engineers; development, security, operations (DevSecOps) engineers; security and compliance administrators; and other stakeholders to deliver robust, high-performance database solutions that power modern applications and AI-driven experiences.

Skills at a glance

- Design and develop database solutions (35–40%)
- Secure, optimize, and deploy database solutions (35–40%)
- Implement AI capabilities in database solutions (25–30%)

Design and develop database solutions (35–40%)

- **Design and implement database objects**
  - Design and implement tables, including data types, size, columns, indexes, and column store indexes
  - Design and implement specialized tables, including in-memory, temporal, external, ledger, and graph
  - Design and implement JSON columns and indexes
  - Design and implement database constraints, including PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, and DEFAULT
  - Design and implement SEQUENCES
  - Design and implement partitioning for tables and indexes
- **Implement programmability objects**
  - Create views
  - Create scalar functions
  - Create table-valued functions
  - Create stored procedures
  - Create triggers
- **Write advanced T-SQL code**
  - Write common table expressions (CTEs)
  - Write queries that include window functions
  - Write queries that include JSON functions, such as JSON_OBJECT, JSON_ARRAY, JSON_ARRAYAGG, JSON_CONTAINS, OPENJSON, and JSON_VALUE
  - Write queries that include regular expressions, such as REGEXP_LIKE, REGEXP_REPLACE, REGEXP_SUBSTR, REGEXP_INSTR, REGEXP_COUNT, REGEXP_MATCHES, and REGEXP_SPLIT_TO_TABLE
  - Write queries that include fuzzy string matching functions, such as EDIT_DISTANCE, EDIT_DISTANCE_SIMILARITY, and JARO_WINKLER_DISTANCE

- Write graph queries that use the MATCH operator
- Write correlated queries
- Implement error handling

- **Design and implement SQL solutions by using AI-assisted tools**
  - Interpret security impact of using AI-assisted tools
  - Enable GitHub Copilot and Microsoft Copilot in Fabric
  - Configure model and Model Context Protocol (MCP) tool options in a GitHub Copilot or Copilot in Fabric chat session
  - Create and configure GitHub Copilot instruction files
  - Connect to MCP server endpoints, including Microsoft SQL Server and Fabric lakehouse

Secure, optimize, and deploy database solutions (35–40%)

- **Implement data security and compliance**
  - Design and implement data encryption, including Always Encrypted and column-level encryption
  - Design and implement Dynamic Data Masking
  - Design and implement Row-Level Security (RLS)
  - Design and implement object-level permissions
  - Implement secure database access, including passwordless
  - Implement auditing
  - Secure model endpoints, including Managed Identity
  - Secure GraphQL, REST, and MCP endpoints
- **Optimize database performance**
  - Recommend database configurations
  - Preserve data integrity and consistency by using transaction isolation levels and concurrency controls
  - Evaluate query performance by using query execution plans, dynamic management views (DMVs), Query Store, and Query Performance Insight

- Identify and resolve query performance issues, including blocking and deadlocks

- **Implement CI/CD by using SQL Database Projects**
  - Design and implement a testing strategy, including unit tests and integration tests
  - Create and manage reference/static data in source control
  - Create, build, and validate database models by using SQL Database Projects, including SDK-style models
  - Configure source control for SQL Database Projects
  - Manage branching, pull requests, and conflict resolution
  - Implement secrets management
  - Detect schema drift by using SQL Database Projects
  - Update an SQL database project and deploy changes
  - Design and implement controls for deployment pipelines, including branching policies, triggers in approvals, authentication tables, and code owners

- **Integrate SQL solutions with Azure services**
  - Create configuration files for Data API builder (DAB)
  - Configure entities for REST and GraphQL, including data caching, pagination, searching, and filtering
  - Configure REST or GraphQL endpoints
  - Expose database objects, stored procedures, and views, including GraphQL relationships
  - Configure and implement DAB deployment
  - Recommend Azure Monitor configurations, including Application Insights and Log Analytics
  - Handle changes by using change event streaming (CES), change data capture (CDC), Change Tracking, Azure Functions with SQL trigger binding, or Azure Logic Apps

Implement AI capabilities in database solutions (25–30%)

- **Design and implement models and embeddings**
  - Evaluate external models, including multimodal, multilanguage, sizes, and structured output
  - Create and manage external models
  - Choose an embedding maintenance method, including table triggers, Change Tracking, Azure Functions with SQL trigger binding, Azure Logic Apps, CDC, CES, and Microsoft Foundry
  - Identify which columns to include in embeddings
  - Design and implement chunks for embeddings
  - Generate embeddings
- **Design and implement intelligent search**
  - Choose from full-text, semantic vector, and hybrid search
  - Implement full-text search
  - Design for vector data, including vector data type, vector indexes, and size
  - Identify when to use vector-related types and functions for semantic searching, including VECTOR_NORMALIZE, VECTOR_DISTANCE, VECTORPROPERTY, and VECTOR_SEARCH
  - Choose between using ANN and ENN for vector search
  - Evaluate vector index types and metrics
  - Implement vector search
  - Implement hybrid search
  - Implement reciprocal rank fusion (RRF)
  - Evaluate performance of vector and hybrid search
- **Design and implement retrieval-augmented generation (RAG)**
  - Identify use cases for RAG

- Create a prompt by using the sp_invoke_external_rest_endpoint stored procedure
- Convert structured data to JSON for language model processing
- Send results to language model
- Extract language model responses