Azure SQL

Azure SQL

- Azure SQL, part of Microsoft's Azure cloud computing platform, is a fully managed relational database service designed to provide a scalable, high-performance, and secure environment for hosting SQL Server databases in the cloud.
- It's built on the foundation of Microsoft SQL Server and offers several deployment options to meet different application needs.

Features

- Scalability: Azure SQL can automatically scale resources up or down based on demand, helping you handle varying workloads without manual intervention.
- High Availability: Azure SQL provides built-in high availability options, including automatic backups, geo-replication, and failover capabilities to ensure minimal downtime.
- Security: It offers robust security features such as data encryption, firewall rules, virtual network integration, and advanced threat detection.
- Intelligence: Azure SQL Database has built-in intelligence to optimize query performance, provide insights into database performance, and recommend improvements.
- Elastic Pools: In Azure SQL Database, you can use elastic pools to manage and share resources among a group of databases, optimizing resource utilization and cost efficiency.

Schemas

- A schema in a database is a logical container that holds objects like tables, views, procedures, functions, and more.
- It acts as a way to organize and group related database objects together.
- Schemas are particularly useful in large databases with multiple users or applications, as they provide a way to manage object naming conflicts and access control.

DDL (Data Definition Language)

- DDL consists of SQL commands used to define and manage the structure or schema of a database.
- DDL commands are responsible for creating, altering, and deleting database objects.
- Some common DDL commands include:
  - CREATE: Used to create database objects like tables, views, indexes, and more.
  - ALTER: Used to modify the structure of existing database objects.
  - DROP: Used to delete or remove database objects.
  - RENAME: Used to change the name of a database object.
  - TRUNCATE: Used to quickly remove all rows from a table.

DML (Data Manipulation Language)

- DML consists of SQL commands used to manipulate data stored within the database.
- DML commands are responsible for inserting, updating, and deleting data in database tables.
- Some common DML commands include:
    - SELECT: Used to retrieve data from one or more tables.
    - INSERT: Used to add new rows of data into a table.
    - UPDATE: Used to modify existing data in a table.
    - DELETE: Used to remove rows of data from a table.

DCL (Data Control Language)

- DCL consists of SQL commands used to control access to data within the database.
- DCL commands are responsible for granting or revoking permissions and managing user access.
- Two primary DCL commands are:
  - GRANT: Used to give specific privileges or permissions to users or roles.
  - REVOKE: Used to take away or revoke previously granted privileges.

Uses

- Schemas: Schemas help organize and manage database objects by grouping them together, allowing for better organization and access control.
- DDL: DDL commands are used to create, modify, and delete database objects, helping to define the structure and schema of the database.
- DML: DML commands are used to manipulate data within the database, allowing for data insertion, modification, and deletion.
- DCL: DCL commands are used to control access to data by granting or revoking permissions to users or roles, ensuring data security and access control.

Create Statement

- The CREATE statement is used to create database objects, such as tables, views, indexes, and more.
- Syntax:

```
CREATE TABLE table_name (
    column1 datatype constraints,
    column2 datatype constraints,
    ...
);
```

Alter Statement

- The ALTER statement is used to modify the structure of an existing database object.
- Syntax (Adding a Column):

```
ALTER TABLE table_name
ADD column_name datatype constraints;
```

Drop Statement

- The DROP statement is used to delete database objects.
- Syntax:

```
DROP TABLE table_name;
```

Primary Key (PK) Constraint

- The primary key constraint ensures that a column (or set of columns) uniquely identifies each row in a table.
- Syntax:

```
CREATE TABLE table_name (
    column_name datatype PRIMARY KEY,
    ...
);
```

Foreign Key (FK) Constraint

- The foreign key constraint establishes a relationship between two tables, ensuring data integrity and referential integrity.
- Syntax:

```
CREATE TABLE table_name1 (
    column_name datatype PRIMARY KEY,
    ...
);

CREATE TABLE table_name2 (
    column_name datatype,
    ...
    FOREIGN KEY (column_name) REFERENCES table_name1(column_name)
);
```

Unique Constraint

- The unique constraint ensures that values in a column (or set of columns) are unique across rows in a table.
- Syntax:

```
CREATE TABLE table_name (
    column_name datatype UNIQUE,
    ...
);
```

NULL Constraint

- The NULL constraint specifies whether a column can contain NULL values or not.
- Syntax (Allowing NULL):

```
CREATE TABLE table_name (
    column_name datatype NULL,
    ...
);
```

Check Constraint

- The check constraint enforces a condition on the values that can be inserted or updated in a column.
- Syntax:

```
CREATE TABLE table_name (
    column_name datatype,
    ...
    CHECK (condition)
);
```

CRUD Operations

Create (INSERT) Operation

- The INSERT statement is used to add new rows of data into a table.
- Syntax:

```
INSERT INTO table_name (column1, column2, ...)
VALUES (value1, value2, ...);
```

Read (SELECT) Operation

- The SELECT statement is used to retrieve data from one or more tables.
- Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Update Operation

- The UPDATE statement is used to modify existing data in a table.
- Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Delete Operation

- The DELETE statement is used to remove rows of data from a table.
- Syntax:

```
DELETE FROM table_name
WHERE condition;
```

GRANT

- The GRANT command is used to give specific privileges or permissions to users or roles.
- Privileges can include the ability to perform certain actions, such as reading data from a table, modifying data, creating objects, and more.
- Syntax for Granting Permissions:

  ```
  GRANT privilege_name ON object_name TO user_or_role;
  ```

- privilege_name: The specific privilege you want to grant, such as SELECT, INSERT, UPDATE, DELETE, EXECUTE, etc.
- object_name: The name of the database object (table, view, procedure, etc.) to which you're granting the privilege.
- user_or_role: The user or role to whom you're granting the privilege.

REVOKE

- The REVOKE command is used to take away or revoke previously granted privileges from users or roles.
- Syntax for Revoking Permissions:

```
REVOKE privilege_name ON object_name FROM user_or_role;
```

- privilege_name: The specific privilege you want to revoke.
- object_name: The name of the database object from which you're revoking the privilege.
- user_or_role: The user or role from whom you're revoking the privilege.

Filtering with WHERE Clause

- The WHERE clause is used to filter rows based on a specified condition.
- It allows you to retrieve only the rows that satisfy the given condition.
- Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

GROUP BY

- The GROUP BY clause is used to group rows that have the same values in specified columns.
- Aggregation functions (like SUM, COUNT, AVG) can be used with GROUP BY to perform calculations on grouped data.
- Syntax:

```
SELECT column1, column2, aggregate_function(column), ...
FROM table_name
GROUP BY column1, column2, ...;
```

HAVING Clause

- The HAVING clause is used with GROUP BY to filter grouped data based on aggregate function results.
- It allows you to select groups that satisfy a given condition.
- Syntax:

```
SELECT column1, column2, aggregate_function(column), ...
FROM table_name
GROUP BY column1, column2, ...
HAVING condition;
```

Sorting Data

- The ORDER BY clause is used to sort the result set in ascending or descending order based on one or more columns.
- By default, sorting is done in ascending order.
- Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1 [ASC | DESC], column2 [ASC | DESC], ...;
```