Data definition Language (DDL)

Here's a comprehensive overview of the key data definition concepts and commands in SQL Server, covering how to manage databases, schemas, and tables:

List of Commands

1. Database Management
    - CREATE DATABASE
    - DROP DATABASE
2. Schema Management
    - CREATE SCHEMA
    - ALTER SCHEMA
    - DROP SCHEMA
3. Table Management
    - CREATE TABLE
    - Identity Column
    - Sequence
    - ALTER TABLE ADD Column
    - ALTER TABLE ALTER COLUMN
    - ALTER TABLE DROP COLUMN
    - Computed Columns
    - DROP TABLE
    - TRUNCATE TABLE

Additional Commands
- RENAME

Advanced

4. Index Management
    - CREATE INDEX
    - DROP INDEX
    - ALTER INDEX
5. View Management
    - CREATE VIEW
    - DROP VIEW
6. Stored Procedure Management
    - CREATE PROCEDURE
    - DROP PROCEDURE
7. User-Defined Function Management
    - CREATE FUNCTION
    - DROP FUNCTION
8. Trigger Management
    - CREATE TRIGGER
    - DROP TRIGGER

1. Database Management

1. CREATE DATABASE

Purpose: To create a new database in an SQL Server instance.

Syntax:

```
  CREATE DATABASE DatabaseName;
```

Example:

```
  CREATE DATABASE CompanyDB;
```

2. DROP DATABASE

Purpose: To delete an existing database.

Syntax:

```
  DROP DATABASE DatabaseName;
```

Example:

```
  DROP DATABASE CompanyDB;
```

2. Schema Management

3. CREATE SCHEMA

Purpose: To create a new schema within a database, which can group related database objects.

Syntax:

```
  CREATE SCHEMA SchemaName;
```

Example:

```
  CREATE SCHEMA HR;
```

4. ALTER SCHEMA

Purpose: To transfer a securable (such as a table) from one schema to another within the same database.

Syntax:

```
  ALTER SCHEMA TargetSchemaName TRANSFER
SourceSchemaName.ObjectName;
```

Example:

```
  ALTER SCHEMA HR TRANSFER dbo.Employees;
```

5. DROP SCHEMA

Purpose: To delete a schema from a database.

Syntax:

```
  DROP SCHEMA SchemaName;
```

Note: You must first drop all objects within the schema.

Example:

```
  DROP SCHEMA HR;
```

3. Table Management

6. CREATE TABLE

Purpose: To create a new table in a specific schema of a database.
Syntax:
```
CREATE TABLE SchemaName.TableName (
    Column1 DataType [Constraints],
    Column2 DataType [Constraints],
    ...
);
```
Example:
```
CREATE TABLE HR.Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    HireDate DATETIME DEFAULT GETDATE()
);
```

7. Identity Column

Purpose: To create a column that automatically generates numeric values, usually used for primary keys.
Syntax:
```
ColumnName INT IDENTITY(Seed, Increment) [Constraints]
```
Example:
```
CREATE TABLE HR.Employees (
    EmployeeID INT IDENTITY(1,1) PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50)
);
```

8. Sequence

Purpose: To generate a sequence of numeric values based on a specification, which can be used for various purposes.
Syntax:
```
CREATE SEQUENCE SequenceName
START WITH StartValue
INCREMENT BY IncrementValue;
```
Example:
```
CREATE SEQUENCE EmployeeSeq
START WITH 1
INCREMENT BY 1;
```

9. ALTER TABLE ADD Column

Purpose: To add one or more columns to an existing table.
Syntax:
```
ALTER TABLE TableName
ADD ColumnName DataType [Constraints];
```
Example:
```
ALTER TABLE HR.Employees
ADD Email VARCHAR(100);
```

10. ALTER TABLE ALTER COLUMN
Purpose: To change the definition of existing columns in a table.
Syntax:
```
ALTER TABLE TableName
ALTER COLUMN ColumnName NewDataType [Constraints];
```
Example:
```
ALTER TABLE HR.Employees
ALTER COLUMN LastName VARCHAR(100);
```

11. ALTER TABLE DROP COLUMN
Purpose: To drop one or more columns from a table.
Syntax:
```
ALTER TABLE TableName
DROP COLUMN ColumnName;
```
Example:
```
ALTER TABLE HR.Employees
DROP COLUMN Email;
```

12. Computed Columns
Purpose: To create a column that is calculated from other columns, allowing you to reuse calculation logic in multiple queries.
Syntax:
```
ColumnName AS (Expression)
```
Example:
```
CREATE TABLE HR.Employees (
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    FullName AS (FirstName + ' ' + LastName)
);
```

13. DROP TABLE
Purpose: To delete tables from the database.
Syntax:
```
DROP TABLE TableName;
```
Example:
```
DROP TABLE HR.Employees;
```

14. TRUNCATE TABLE
Purpose: To remove all rows from a table quickly without logging individual row deletions, while keeping the table structure.
Syntax:
```
TRUNCATE TABLE TableName;
```
Example:
```
TRUNCATE TABLE HR.Employees;
```

Additional Commands

RENAME

Purpose: To rename an existing database object (e.g., table, column).

Syntax:

```
EXEC sp_rename 'OldName', 'NewName';
```

Example (Renaming a Table):

```
EXEC sp_rename 'HR.Employees', 'HR.Staff';
```

Example (Renaming a Column):

```
EXEC sp_rename 'HR.Staff.FirstName', 'First_Name', 'COLUMN';
```

Here are additional commands related to data definition in SQL Server that may have been overlooked in the previous overview:

4. Index Management

15. CREATE INDEX

Purpose: To create an index on a table to improve query performance.

Syntax:
```
  CREATE INDEX IndexName
  ON TableName (Column1, Column2);
```
Example:
```
  CREATE INDEX IX_Employees_LastName
  ON HR.Employees (LastName);
```

16. DROP INDEX

Purpose: To remove an index from a table.

Syntax:
```
  DROP INDEX IndexName ON TableName;
```
Example:
```
  DROP INDEX IX_Employees_LastName ON HR.Employees;
```

17. ALTER INDEX

Purpose: To rebuild or reorganize an existing index to improve performance.

Syntax:
```
  ALTER INDEX IndexName ON TableName REBUILD; -or REORGANIZE
```
Example:
```
  ALTER INDEX IX_Employees_LastName ON HR.Employees REBUILD;
```

5. View Management

18. CREATE VIEW

Purpose: To create a virtual table based on the result set of a SELECT query.

Syntax:
```
  CREATE VIEW ViewName AS
  SELECT Column1, Column2
  FROM TableName
  WHERE Condition;
```
Example:
```
  CREATE VIEW vw_EmployeeNames AS
  SELECT FirstName, LastName
  FROM HR.Employees;
```

19. DROP VIEW

Purpose: To remove a view from the database.

Syntax:
```
  DROP VIEW ViewName;
```
Example:
```
  DROP VIEW vw_EmployeeNames;
```

6. Stored Procedure Management

20. CREATE PROCEDURE

Purpose: To create a stored procedure, which is a set of SQL statements that can be executed as a single unit.

Syntax:

```
CREATE PROCEDURE ProcedureName
AS
BEGIN
   -SQL statements
END;
```

Example:

```
CREATE PROCEDURE GetAllEmployees
AS
BEGIN
   SELECT * FROM HR.Employees;
END;
```

21. DROP PROCEDURE

Purpose: To remove a stored procedure from the database.

Syntax:

```
DROP PROCEDURE ProcedureName;
```

Example:

```
DROP PROCEDURE GetAllEmployees;
```

7. User-Defined Function Management

22. CREATE FUNCTION

Purpose: To create a user-defined function that can return a single value or a table.

Syntax:

```
CREATE FUNCTION FunctionName (@Parameter DataType)
RETURNS ReturnType
AS
BEGIN
   -SQL statements
   RETURN Value;
END;
```

Example:

```
CREATE FUNCTION GetEmployeeFullName (@EmployeeID INT)
RETURNS VARCHAR(100)
AS
BEGIN
   DECLARE @FullName VARCHAR(100);
   SELECT @FullName = FirstName + ' ' + LastName
   FROM HR.Employees
   WHERE EmployeeID = @EmployeeID;
   RETURN @FullName;
END;
```

23. DROP FUNCTION

Purpose: To remove a user-defined function from the database.

Syntax:

```
DROP FUNCTION FunctionName;
```
Example:
```
DROP FUNCTION GetEmployeeFullName;
```

8. Trigger Management

24. CREATE TRIGGER

Purpose: To create a trigger, which is a special type of stored procedure that automatically runs when a specific event occurs in the database (e.g., INSERT, UPDATE, DELETE).

Syntax:
```
CREATE TRIGGER TriggerName
ON TableName
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
   -SQL statements
END;
```
Example:
```
CREATE TRIGGER trg_AfterInsert_Employees
ON HR.Employees
AFTER INSERT
AS
BEGIN
   PRINT 'New employee added.';
END;
```

25. DROP TRIGGER

Purpose: To remove a trigger from the database.

Syntax:
```
DROP TRIGGER TriggerName;
```
Example:
```
DROP TRIGGER trg_AfterInsert_Employees;
```