Class Notes for Session 3 (SQL–TL3): Transact-SQL with Expanded Focus on DDL and DML

Module

Data Management with SQL Server
Session Title: SQL–TL3
Source: SQL Server–The Definitive Guide, Session 4
Duration: 2 hours
Objective: By the end of this session, students will be able to:

- Understand Transact-SQL (T-SQL) and its role in SQL Server, with a deep focus on DDL (`CREATE` , `ALTER` , `DROP`) and DML statements.
- Categorize and apply T-SQL statements, emphasizing DDL and DML.
- Describe T-SQL data types, language elements, sets, predicate logic, and the logical order of SELECT statements.
- Write and execute T-SQL queries in SSMS using the AdventureWorks 2022 database.

Sub-Session 1: Introduction to Transact-SQL

Definition

Transact-SQL (T-SQL) is Microsoft’s proprietary extension to SQL, used for interacting with SQL Server databases. It includes standard SQL for querying and managing data, plus advanced features like procedural programming, error handling, and functions.

Explanation

- **Purpose**: T-SQL enables creating, querying, modifying, and managing databases.
- **Components**: Includes DDL (defining structures), DML (manipulating data), DCL (controlling access), and procedural constructs (e.g., stored procedures).

- **Execution**: Run via SSMS, command-line tools, or application code.

Example

```
USE AdventureWorks2022;
SELECT FirstName, LastName
FROM Person.Person
WHERE Title = 'Ms.';
```

This query retrieves names of individuals with the title "Ms." from the `Person.Person` table.

Class Work

1. **Discussion**: Students discuss how T-SQL's procedural capabilities (e.g., variables, loops) extend standard SQL, using an example from AdventureWorks.
2. **Lab Exercise**: Write a T-SQL query to retrieve the top 5 records from `Production.Product`. Save as `IntroTSQL.sql` and execute in SSMS.

Sub-Session 2: Categories of Transact-SQL Statements with Expanded Focus on DDL and DML

Definition

T-SQL statements are categorized into:

- **Data Definition Language (DDL)**: Defines or modifies database structures (e.g., tables, schemas).
- **Data Manipulation Language (DML)**: Manipulates data within tables (e.g., query, insert, update).
- **Data Control Language (DCL)**: Manages permissions (e.g., `GRANT`, `REVOKE`).

Expanded Focus on DDL: CREATE, ALTER, DROP

CREATE

- **Definition**: Creates new database objects like tables, views, or schemas.
- **Syntax**:

```
CREATE TABLE TableName (
    Column1 DataType [Constraints],
    Column2 DataType [Constraints],
    ...
);
```

- **Key Features**:
  - Defines columns, data types, and constraints (e.g., `PRIMARY KEY`, `FOREIGN KEY`, `NOT NULL`).
  - Supports creating indexes, triggers, or stored procedures.
  - Ensures objects are uniquely named within a schema.
- **Use Cases**: Creating tables for new data, such as customer records or logs.
- **Best Practices**:
  - Use meaningful column names and appropriate data types.
  - Include constraints to enforce data integrity (e.g., `CHECK` for valid ranges).
  - Specify schemas to avoid naming conflicts (e.g., `dbo.TableName`).

Example:

```
USE AdventureWorks2022;
CREATE TABLE CustomerFeedback (
    FeedbackID INT PRIMARY KEY,
    CustomerID INT NOT NULL FOREIGN KEY REFERENCES Sales.Customer(CustomerID),
    FeedbackDate DATE NOT NULL,
    Comments NVARCHAR(500),
    Rating INT CHECK (Rating BETWEEN 1 AND 5)
);
```

This creates a table to store customer feedback, linked to the `Sales.Customer` table, with constraints for data integrity.

ALTER

- **Definition**: Modifies existing database objects, such as adding, modifying, or dropping columns or constraints.
- **Syntax**:

```
ALTER TABLE TableName
ADD ColumnName DataType [Constraints] |
ALTER COLUMN ColumnName DataType [Constraints] |
DROP COLUMN ColumnName;
```

- **Key Features**:
  - Adds new columns or constraints (e.g., `ADD` for new fields).
  - Modifies column properties (e.g., changing data type or constraints).
  - Drops columns or constraints (with caution to avoid data loss).
- **Use Cases**: Updating a table to include new fields (e.g., adding a `Status` column) or changing data types for better storage.
- **Best Practices**:
  - Verify data compatibility before altering (e.g., ensure existing data fits new data type).
  - Backup data before dropping columns or constraints.
  - Test alterations in a development environment first.

Example:

```
USE AdventureWorks2022;
ALTER TABLE CustomerFeedback
ADD FeedbackStatus NVARCHAR(20) DEFAULT 'Pending';
ALTER TABLE CustomerFeedback
ALTER COLUMN Comments NVARCHAR(1000); -- Increase comment length
ALTER TABLE CustomerFeedback
DROP COLUMN Rating; -- Remove rating column
```

This modifies the `CustomerFeedback` table by adding a status column, increasing the comment length, and dropping the rating column.

DROP

- **Definition**: Deletes database objects like tables, views, or indexes.
- **Syntax**:

  ```
  DROP TABLE TableName;
  ```

- **Key Features**:
  - Permanently removes objects and their data.
  - Cannot be undone unless a transaction is used or data is backed up.
  - Checks for dependencies (e.g., foreign keys) before dropping.
- **Use Cases**: Removing obsolete tables or cleaning up test data.
- **Best Practices**:
  - Confirm the object is no longer needed.
  - Check for dependencies (e.g., foreign key constraints).
  - Use `IF EXISTS` to avoid errors if the object doesn't exist.

Example:

```
USE AdventureWorks2022;
DROP TABLE IF EXISTS CustomerFeedback;
```

This safely deletes the `CustomerFeedback` table if it exists.

Expanded Focus on DML

DML Statements: SELECT, INSERT, UPDATE, DELETE

- **Definition**: DML statements manipulate data within tables, allowing retrieval, addition, modification, or deletion of records.
- **Key Statements**:
  - **SELECT**: Retrieves data from one or more tables.
  - **INSERT**: Adds new rows to a table.
  - **UPDATE**: Modifies existing rows.
  - **DELETE**: Removes rows from a table.
- **Key Features**:
  - Operate on sets of data (rows) rather than individual records.
  - Support conditions (e.g., `WHERE`) for precise data manipulation.
  - Can be combined with joins, subqueries, or set operations.
- **Use Cases**:
  - Querying sales data for reports (`SELECT`).
  - Adding new customer records (`INSERT`).
  - Updating product prices (`UPDATE`).
  - Removing outdated orders (`DELETE`).
- **Best Practices**:
  - Use `WHERE` clauses to avoid unintended changes.
  - Test DML statements in a transaction to allow rollback if needed.
  - Optimize queries with indexes for large datasets.

Examples:

- **SELECT**:

```
USE AdventureWorks2022;
SELECT TOP 5 ProductID, Name, ListPrice
FROM Production.Product
WHERE ListPrice > 500
ORDER BY ListPrice DESC;
```

Retrieves the top 5 expensive products.

- **INSERT**:

```
INSERT INTO CustomerFeedback (FeedbackID, CustomerID, FeedbackDate, Comments)
VALUES (1, 11001, '2025-05-26', 'Excellent service!');
```

Adds a feedback record for a customer.

- **UPDATE**:

```
UPDATE CustomerFeedback
SET Comments = 'Updated: Excellent service!'
WHERE FeedbackID = 1;
```

Modifies the comment for a specific feedback record.

- **DELETE**:

```
DELETE FROM CustomerFeedback
WHERE FeedbackDate < '2025-01-01';
```

Removes outdated feedback records.

Class Work

1. **DDL Lab Exercise**: Write a T-SQL script that:
   - Creates a table `EmployeeReviews` with columns: `ReviewID` (INT, PK), `EmployeeID` (INT, FK to `HumanResources.Employee`), `ReviewDate` (DATE), `Feedback` (NVARCHAR).
   - Alters the table to add a `Score` (INT) column with a default value of 0.
   - Drops the table safely using `IF EXISTS`.
     Save as `DDLOperations.sql` and execute in SSMS.
2. **DML Lab Exercise**: Write a T-SQL script that:
   - Inserts 2 records into `EmployeeReviews`.
   - Updates one record to change the `Feedback` text.
   - Deletes one record based on a condition (e.g., `Score < 3`).

- Queries the table to display all records.
  Save as `DMLOperations.sql` and execute in SSMS.

3. **Discussion**: In groups, discuss the impact of running `DROP TABLE` without `IF EXISTS` or a `WHERE` clause in `DELETE`. Share potential risks and solutions.

Sub-Session 3: Data Types Supported by Transact-SQL

Definition

T-SQL data types define the type of data a column or variable can store, such as numbers, text, or dates, critical for DDL and DML operations.

Explanation

- **Key Data Types** (used in DDL/DML):
  - **Numeric**: `INT` (whole numbers, e.g., IDs), `DECIMAL` (precise decimals, e.g., prices), `FLOAT` (approximate decimals).
  - **Character**: `CHAR` (fixed-length), `VARCHAR` (variable-length), `NVARCHAR` (Unicode variable-length, e.g., names).
  - **Date and Time**: `DATE` (dates), `DATETIME` (date and time), `TIME` (time only).
  - **Other**: `XML`, `JSON`, `UNIQUEIDENTIFIER` (GUIDs).
- **Use in DDL**: Define column types in `CREATE TABLE` or `ALTER TABLE`.
- **Use in DML**: Ensure data matches column types in `INSERT` or `UPDATE`.

Example

```
USE AdventureWorks2022;
CREATE TABLE SalesLog (
    LogID INT PRIMARY KEY,
    SaleDate DATE NOT NULL,
    Amount DECIMAL(10,2),
    Description NVARCHAR(200)
);
INSERT INTO SalesLog (LogID, SaleDate, Amount, Description)
VALUES (1, '2025-05-26', 999.99, 'Bike Sale');
SELECT * FROM SalesLog;
```

Class Work

1. **Exercise**: Create a table `OrderTracking` with columns using `INT`, `NVARCHAR`, `DATE`, and `DECIMAL`. Insert a record and query it. Save as `DataTypes.sql`.
2. **Identification Task**: List 3 data types used in `Production.Product` and explain their purpose (e.g., `NVARCHAR` for `Name`).

Sub-Session 4: Transact-SQL Language Elements

Definition

T-SQL language elements include variables, operators, predicates, expressions, and comments, used in both DDL and DML.

Explanation

- **Variables**: Store temporary data (e.g., `DECLARE @Total DECIMAL;`).
- **Operators**: Arithmetic (`+`), comparison (`=`), logical (`AND`).
- **Predicates**: Conditions in `WHERE` or `HAVING` (e.g., `Amount > 100`).
- **Expressions**: Calculations or concatenations (e.g., `ListPrice * 1.1`).
- **Comments**: `--` or `/* */` for documentation.

Example

```
USE AdventureWorks2022;
DECLARE @MinPrice DECIMAL(10,2) = 1000.00;
SELECT ProductID, Name, ListPrice
FROM Production.Product
WHERE ListPrice > @MinPrice; -- Filter expensive products
/* Displays products above the variable price */
```

Class Work

1. **Lab Exercise**: Write a script using a variable, predicate, and expression to query `Sales.SalesOrderHeader` for orders above a threshold. Save as `LanguageElements.sql`.
2. **Activity**: Identify language elements in a provided T-SQL script.

Sub-Session 5: Sets and Predicate Logic

Definition

- **Sets**: Collections of rows manipulated with operations like `UNION`.
- **Predicate Logic**: Logical conditions (e.g., `WHERE`) using `AND`, `OR`, `NOT`.

Explanation

- **Sets**: Used in DML for combining query results (e.g., `UNION`).
- **Predicate Logic**: Filters data in DML `SELECT`, `UPDATE`, or `DELETE`.

Example

```
USE AdventureWorks2022;
SELECT ProductID, Name
FROM Production.Product
WHERE Color = 'Black'
UNION
SELECT ProductID, Name
WHERE Color = 'Silver';
```

Class Work

1. **Lab Exercise**: Write a query using `UNION` to combine `Production.Product` records for two conditions (e.g., `Color = 'Red'` and `ListPrice > 500`). Save as `SetLogic.sql`.
2. **Exercise**: Write a query with `AND` / `OR` predicates to filter `Sales.SalesOrderHeader`. Save as `PredicateLogic.sql`.

Sub-Session 6: Logical Order of Operators in the SELECT Statement

Definition

The logical order of a `SELECT` statement defines how SQL Server processes clauses, critical for DML queries.

Explanation

- **Order**: FROM, WHERE, GROUP BY, HAVING, SELECT, ORDER BY.
- **Impact on DML**: Affects how `SELECT` retrieves data or how `UPDATE` / `DELETE` targets rows.

Example

```
USE AdventureWorks2022;
SELECT CustomerID, SUM(TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader
WHERE OrderDate >= '2020-01-01'
GROUP BY CustomerID
HAVING SUM(TotalDue) > 5000
ORDER BY TotalSales DESC;
```

Class Work

1. **Exercise**: Write a `SELECT` query with `GROUP BY`, `HAVING`, and `ORDER BY` for `Sales.SalesOrderHeader`. Explain the logical order. Save as `SelectOrder.sql`.
2. **Diagram Activity**: Draw the logical order of a `SELECT` statement.

Additional Class Work Summary

1. **Group Project**: Create a presentation on DDL (`CREATE`, `ALTER`, `DROP`) and DML (`SELECT`, `INSERT`, `UPDATE`, `DELETE`) with AdventureWorks examples.
2. **Practical Assignment**: Write scripts for:
   - DDL: Create and modify a table.

- DML: Insert, update, delete, and query data.

  Save in a `Session3_Exercises` folder. Submit scripts and results.

3. **Quiz**:
   - What does `ALTER TABLE` do? (Answer: Modifies table structure.)
   - Name a DML statement. (Answer: `INSERT` .)
   - First step in SELECT logical order? (Answer: FROM.)

Deliverables Mapping

- **Student Guide (SG)**: Session 4
- **Exercise Package (XP)**: Session 4
- **Trainer Guide (TG)**: Session 4

Resources

- **OnlineVarsity**: Download eBook, access Glossary, use Practice 4 Me.
- **Library References**:
  - *Murach's SQL Server 2022 for Developers* by Bryan Syverson.
  - *T-SQL Fundamentals, 4th Edition* by Itzik Ben-Gan.