MSSQL & T-SQL Course Notes

Module 2: Microsoft SQL Management Studio (SSMS)

1. Why SSMS?

Microsoft SQL Server Management Studio (SSMS) is an integrated environment used for managing SQL Server databases. It provides a Graphical User Interface (GUI) alongside a powerful query editor to interact with databases using T-SQL.

Advantages of SSMS:

- **Comprehensive Management Tool**: Suitable for writing T-SQL queries, managing database objects, configuring settings, handling security, and monitoring SQL servers.
- **Ease of Administration**: Provides a visual interface for database administration tasks.
- **Industry Standard**: Used widely in enterprise environments for database management.
- **Integration with Cloud and BI Tools**: Supports Microsoft Azure, Power BI, indexing, performance tuning, and workload monitoring.
- **Support for T-SQL**: Unlike MySQL, MariaDB, or PostgreSQL, MSSQL includes **Transact-SQL (T-SQL)**, which extends SQL with procedural programming, error handling, and advanced query capabilities.

---

2. Degree of Relationship in Databases

Degree of relationship represents the number of entity types involved in an association. There are four main types:

Unary Relationship (Degree = 1)

A relationship between entities of the same type.

Example: Employees have managers who are also employees.

```
CREATE TABLE Employee (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(100),
    ManagerID INT FOREIGN KEY REFERENCES Employee(EmployeeID)
);
```

Binary Relationship (Degree = 2)

A relationship between two different entities.

Example: A student enrolls in a course.

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
```

```
    Name VARCHAR(100)
);

CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    Title VARCHAR(100)
);

CREATE TABLE Enrollment (
    StudentID INT FOREIGN KEY REFERENCES Student(StudentID),
    CourseID INT FOREIGN KEY REFERENCES Course(CourseID),
    PRIMARY KEY (StudentID, CourseID)
);
```

Ternary Relationship (Degree = 3)

A relationship between three different entities.

Example: A doctor treats a patient in a specific hospital.

```
CREATE TABLE Doctor (
    DoctorID INT PRIMARY KEY,
    Name VARCHAR(100)
);

CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,
    Name VARCHAR(100)
);

CREATE TABLE Hospital (
    HospitalID INT PRIMARY KEY,
    Name VARCHAR(100)
);

CREATE TABLE Treatment (
    DoctorID INT FOREIGN KEY REFERENCES Doctor(DoctorID),
    PatientID INT FOREIGN KEY REFERENCES Patient(PatientID),
    HospitalID INT FOREIGN KEY REFERENCES Hospital(HospitalID),
    PRIMARY KEY (DoctorID, PatientID, HospitalID)
);
```

N-ary Relationship (Degree > 3)

A relationship involving more than three entities.

---

3. Client-Server Architecture

Types of Database Architectures

1. **1-Tier Architecture**: Everything is on a single system (database, application, and user interface).
2. **2-Tier Architecture**: The database and application are separated; a client communicates

directly with the database.
3. **3-Tier Architecture**: The database, application logic, and user interface are separated for better scalability.

Common Database APIs

- **ODBC (Open Database Connectivity)**: Standard API for accessing database management systems.
- **JDBC (Java Database Connectivity)**: API used in Java applications for database interaction.

---

4. Normalization

Normalization is the process of organizing data to **minimize redundancy** and **improve integrity** by dividing large tables into smaller related tables.

Common Normal Forms (NF):

1NF (First Normal Form) – Atomicity

- Each column must contain atomic (indivisible) values.
- Each row must be unique.

Example (Before 1NF):

StudentID	Name	Subjects
1	Alice	Math, Science

Example (After 1NF):

StudentID	Name	Subject
1	Alice	Math
1	Alice	Science

2NF (Second Normal Form) – No Partial Dependency

- The table must be in 1NF.
- Every non-key attribute should be fully functionally dependent on the **entire** primary key.

Example (Before 2NF):

StudentID	CourseID	StudentName	CourseName
1	C101	Alice	Math

Here, **StudentName** depends only on **StudentID**, and **CourseName** depends only on **CourseID**, violating 2NF.

After 2NF (Splitting into two tables):

```
CREATE TABLE Student (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(100)
);
```

```
CREATE TABLE Course (
    CourseID INT PRIMARY KEY,
    Title VARCHAR(100)
);
```

3NF (Third Normal Form) – No Transitive Dependency

- The table must be in 2NF.
- No non-key attribute should depend on another non-key attribute.

Example (Before 3NF):

StudentID	Name	ZipCode	City
1	Alice	12345	NY

Here, **City** depends on **ZipCode**, which is not a primary key.

After 3NF (Splitting into two tables):

```
CREATE TABLE ZipCode (
    ZipCode INT PRIMARY KEY,
    City VARCHAR(100)
);
```

BCNF (Boyce-Codd Normal Form)

- A stronger version of 3NF.
- Every determinant must be a **candidate key**.

4NF (Fourth Normal Form)

- Removes multi-valued dependencies.
- Avoids redundancy due to independent relationships.

5NF (Fifth Normal Form)

- Ensures no join dependency exists in the table.

Advantages of Normalization:

✓Minimizes redundancy ✓Ensures data consistency ✓Improves flexibility ✓Enforces relational integrity

Disadvantages of Normalization:

✗More complex queries due to multiple tables ✗Performance overhead due to joins ✗Requires a thorough understanding of data dependencies before designing the database

---

Conclusion

MSSQL and T-SQL provide robust database management capabilities. SSMS enhances efficiency through its GUI and query editor, while normalization ensures efficient database design. Understanding these concepts is essential for data science and enterprise database management.