Database Administration Security

Cybersecurity Department

Course Code: ***CBS 214***

Practical Lecture 3: Database Design (Task-Based Lecture)

Halal Abdulrahman Ahmed

Lecture Outlines

- SQL Mini Project (University System)
- Create Database and Tables
- Primary Key & Foreign Key Relationships
- Apply Constraints (IDENTITY, UNIQUE, CHECK)
- Insert Sample Data
- Data Retrieval using SELECT
- JOIN Types (INNER, LEFT, Multi-table)
- GROUP BY with JOIN

Learning Outcomes

By the end of this lecture, students will be able to:

- Create a database and relational tables in SQL Server
- Apply keys and constraints correctly
- Insert and manage data records
- Use JOIN queries to combine tables
- Perform simple data analysis using GROUP BY
- Execute a complete SQL mini project

Practical Task - SQL Table Design & Relationships

Design and implement a small database using **Microsoft SQL Server** by completing the following requirements:

Part 1 - Create Database

- Create a new database called **UniversityDB**.

```
CREATE DATABASE UniversityDB;
GO

USE UniversityDB;
GO
```

Command Explanation

CREATE DATABASE

- Creates a new database.

GO

- Batch separator.
- Executes all commands written before it.
- Used to separate blocks of code.

USE

- Selects the database to work inside.

Part 2 - Create Tables

Create Table 1 - Department

```
CREATE TABLE Department(
    DepartmentID INT IDENTITY PRIMARY KEY,
    DepartmentName NVARCHAR(100) NOT NULL
);
```

Create Table 2 - Instructor

```
CREATE TABLE Instructor(
    InstructorID INT IDENTITY PRIMARY KEY,
    FullName NVARCHAR(100) NOT NULL,
    Email VARCHAR(120) UNIQUE NOT NULL,
    Rank VARCHAR(30) CHECK
        (Rank IN ('Assistant Lecturer','Lecturer','Professor')),
    DepartmentID INT NOT NULL,
    FOREIGN KEY (DepartmentID)
        REFERENCES Department(DepartmentID)
);
```

Part 2 - Create Tables

Create Table 3 - Student

```
CREATE TABLE Student(
    StudentID INT IDENTITY PRIMARY KEY,
    FullName NVARCHAR(100) NOT NULL,
    Email VARCHAR(120) UNIQUE NOT NULL,
    DepartmentID INT NOT NULL,
    FOREIGN KEY (DepartmentID)
        REFERENCES Department(DepartmentID)
);
```

Create Table 4 - Course

```
CREATE TABLE Course(
    CourseID INT IDENTITY PRIMARY KEY,
    CourseName NVARCHAR(100) NOT NULL,
    CreditHours INT NOT NULL,
    InstructorID INT NOT NULL,
    FOREIGN KEY (InstructorID)
        REFERENCES Instructor(InstructorID)
);
```

Part 3 - Insert Sample Data

```sql
INSERT INTO Department (DepartmentName)
VALUES ('Cybersecurity'),
       ('IT'),
       ('Software Engineering');

INSERT INTO Instructor (FullName, Email, Rank, DepartmentID)
VALUES ('Ali Ahmed','ali@tiu.edu.iq','Lecturer',1),
       ('Sara Karim','sara@tiu.edu.iq','Assistant Lecturer',2);

INSERT INTO Student (FullName, Email, DepartmentID)
VALUES ('Hana Aziz','hana@tiu.edu.iq',1),
       ('Kozhen Taher','kozhen@tiu.edu.iq',2);

INSERT INTO Course (CourseName, CreditHours, InstructorID)
VALUES ('Database Fundamentals',3,1),
       ('Programming Basics',3,2);
```

Part 4 - Display All Data

```sql
SELECT * FROM Department;
SELECT * FROM Instructor;
SELECT * FROM Student;
SELECT * FROM Course;
```

Part 5 - Inner JOIN - Students with Department

```sql
SELECT s.StudentID, s.FullName, s.GradeLevel, d.DepartmentName
FROM Student s
INNER JOIN Department d ON s.DepartmentID = d.DepartmentID;
```

Purpose

- Shows students together with their department.
- Basic JOIN (beginner level).

Part 6 - Multi-table JOIN - Enrollment Report

```
FROM Enrollment e
INNER JOIN Student s ON e.StudentID = s.StudentID
INNER JOIN Course c ON e.CourseID = c.CourseID
```

Purpose

- Combines **3 tables** together.
- Used to build reports.

Shows:

- Student name
- Course
- Marks
- Average score

Part 7 - LEFT JOIN (Show All Students)

```
LEFT JOIN Enrollment e ON s.StudentID = e.StudentID
LEFT JOIN Course c ON e.CourseID = c.CourseID;
```

Purpose

- Shows ALL students.
- Even if they are not enrolled.

Difference:
INNER JOIN → only matching rows
LEFT JOIN → keeps everything from left table

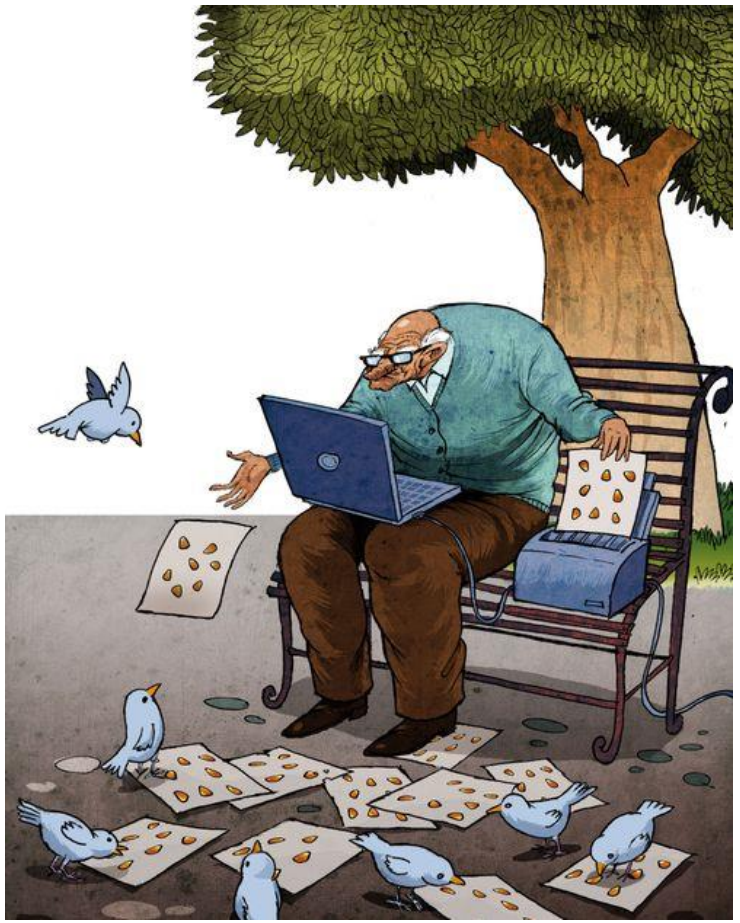
Part 8 - JOIN + GROUP BY (Count Students)

```sql
SELECT c.CourseCode, COUNT(e.StudentID)
FROM Course c
LEFT JOIN Enrollment e ON c.CourseID = e.CourseID
GROUP BY c.CourseCode;
```

Purpose

- Counts how many students are in each course.
- Introduces aggregation with JOIN.

Full Task



```sql
/* MINI PROJECT – University System (No Inheritance) */

-- 1) Create Database
IF DB_ID('MiniUniversityDB') IS NOT NULL
    DROP DATABASE MiniUniversityDB;
GO

CREATE DATABASE MiniUniversityDB;
GO

USE MiniUniversityDB;
GO
```

```sql
IF DB_ID('UniversityDB') IS NOT NULL
    DROP DATABASE UniversityDB;
GO

CREATE DATABASE UniversityDB;
GO

USE UniversityDB;
GO

/* TABLE 1: Department */
IF OBJECT_ID('Department') IS NOT NULL
    DROP TABLE Department;

CREATE TABLE Department(
    DeptID INT IDENTITY PRIMARY KEY,
    DeptName NVARCHAR(50) NOT NULL UNIQUE
);
```

```sql
/* TABLE 2: Student */
IF OBJECT_ID('Student') IS NOT NULL
    DROP TABLE Student;

CREATE TABLE Student(
    SID INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Grade TINYINT CHECK(Grade BETWEEN 1 AND 5),
    DeptID INT REFERENCES Department(DeptID)
);
```

```sql
/* TABLE 3: Instructor */
IF OBJECT_ID('Instructor') IS NOT NULL
    DROP TABLE Instructor;

CREATE TABLE Instructor(
    IID INT IDENTITY PRIMARY KEY,
    Name NVARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Rank VARCHAR(30)
        CHECK (
            Rank IN (
                'Assistant Lecturer',
                'Lecturer',
                'Professor'
            )
        ),
    DeptID INT REFERENCES Department(DeptID)
);
```

```sql
/* TABLE 4: Course */
IF OBJECT_ID('Course') IS NOT NULL
    DROP TABLE Course;

CREATE TABLE Course(
    CID INT IDENTITY PRIMARY KEY,
    Code VARCHAR(10) UNIQUE,
    Title NVARCHAR(100),
    Credits TINYINT CHECK(Credits BETWEEN 1 AND 6),
    DeptID INT REFERENCES Department(DeptID)
);
```

```sql
/* TABLE 5: Enrollment */
IF OBJECT_ID('Enrollment') IS NOT NULL
    DROP TABLE Enrollment;

CREATE TABLE Enrollment(
    EID INT IDENTITY PRIMARY KEY,
    SID INT REFERENCES Student(SID),
    CID INT REFERENCES Course(CID),
    Mid DECIMAL(5,2) CHECK(Mid BETWEEN 0 AND 100),
    Fin DECIMAL(5,2) CHECK(Fin BETWEEN 0 AND 100),
    UNIQUE(SID, CID)
);
GO
```

```sql
/* INSERT DATA */
INSERT INTO Department
VALUES ('Cybersecurity'), ('IT');

INSERT INTO Student
VALUES ('Rawan','rawan@tiu.edu',1,1),
       ('Shanaz','shanaz@tiu.edu',1,1),
       ('Emer','emer@tiu.edu',1,1);

INSERT INTO Instructor
VALUES ('Halal','halal@tiu.edu','Lecturer',1);

INSERT INTO Course
VALUES ('DB101','Database',3,1),
       ('CS201','Networks',3,1);

INSERT INTO Enrollment
VALUES (1,1,78,82),
       (2,1,88,90);
```

```sql
/* PART 5: INNER JOIN */
SELECT s.Name, d.DeptName
FROM Student s
INNER JOIN Department d
ON s.DeptID = d.DeptID;

/* PART 6: MULTI-TABLE JOIN */
SELECT s.Name,
       c.Title,
       e.Mid,
       e.Fin,
       (e.Mid + e.Fin)/2.0 AS Average
FROM Enrollment e
JOIN Student s ON e.SID = s.SID
JOIN Course c ON e.CID = c.CID;
```

```sql
/* PART 7: LEFT JOIN */
SELECT s.Name, e.CID
FROM Student s
LEFT JOIN Enrollment e
ON s.SID = e.SID;

/* PART 8: JOIN + GROUP BY */
SELECT c.Code,
       COUNT(e.SID) AS Total
FROM Course c
LEFT JOIN Enrollment e
ON c.CID = e.CID
GROUP BY c.Code;
```

References

Microsoft. (n.d.). *SQL sample databases*. Microsoft Learn. https://learn.microsoft.com/en-us/sql/samples/sql-samples-where-are?view=sql-server-ver17

Ben-Gan, I. (2016). *T-SQL fundamentals* (3rd ed.). Microsoft Press.


Any
Question