Problem 2: University Examination System

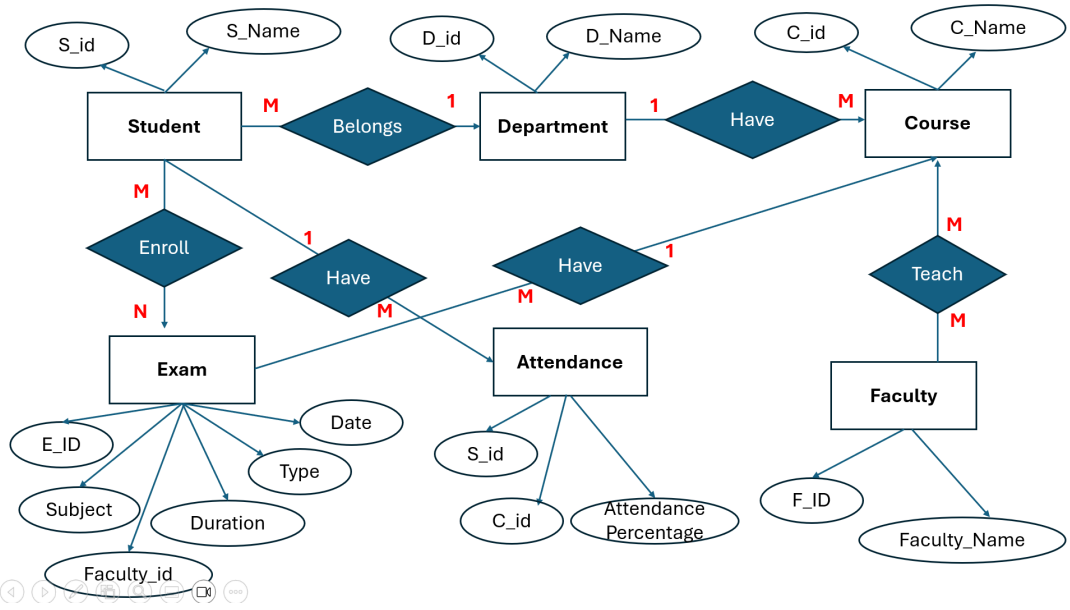
Design an Entity-Relationship schema for a university examination system that manages data about **exams**, **students**, **faculty members**, **courses**, and **departments**.

Each **department** has a unique name and is headed by a **faculty member**. A department can offer multiple **courses**, and each course has a unique course code, title, and is coordinated by a faculty member.**Faculty members** have an employee ID, name, and designation. They can teach multiple courses, coordinate specific courses, and also serve as heads of departments. A faculty member may handle multiple roles at once.

Students have a roll number and name, and each student belongs to one department. A student can enroll in multiple courses offered by that department. For each enrolled course, a student has an **attendance percentage** recorded.

Exams are created by faculty members .Each exam has a title, subject name (which is assumed to be the same as the course name), duration, date, type (internal or external), and is always linked to a specific course. Students may appear in multiple exams related to their courses, and for each exam, a student may have multiple attempts, with marks and attempt dates recorded for each.

All relationships between students, courses, faculty, and exams must reflect these associations clearly — such as student-course enrollment, faculty-course teaching, course-department mapping, and exam-course ownership.



🎓 Student

- Each student (`S_id`, `S_Name`) belongs to **one department**
- A student **enrolls in courses**
- A student **gives exams** and **has attendance** in each course

---

🏢 Department

- Each department (`D_id`, `D_Name`) offers multiple **courses**
- Examples:
    - D_id = 101, D_Name = "CSE"
    - D_id = 102, D_Name = "Electrical"

📚 Course

- Each course (`C_id`, `C_Name`) belongs to **one department**
- A course is **taught by faculty**
- Students **attend** courses and give **exams** in them

👨‍🏫 Faculty

- Each faculty (`F_ID`, `Faculty_Name`) can:
    - **Teach courses**
    - **Create exams** (examiner)

📝 Exam

- Each exam has:
    - `E_ID` (Exam ID)
    - `Subject` (e.g., Java, ML, DBMS)
    - `Date`, `Duration`, `Type` (internal/external)
    - Created by a faculty (`Faculty_id`)
- Students attempt exams for courses they're enrolled in.

📊 Attendance

- Attendance is recorded per **student per course**
- Contains: `S_id`, `C_id`, and `Attendance_Percentage`

📋 3. SQL TABLE CREATION STATEMENTS

🧑‍🎓 Student Table

```
CREATE TABLE Student (
    S_id INT PRIMARY KEY,
    S_Name VARCHAR(100),
    D_id INT,
    FOREIGN KEY (D_id) REFERENCES Department(D_id)
);
```

🏢 Department Table

```
CREATE TABLE Department (
    D_id INT PRIMARY KEY,
    D_Name VARCHAR(100)
);
```

📚 Course Table

```
CREATE TABLE Course (
    C_id INT PRIMARY KEY,
    C_Name VARCHAR(100),
    D_id INT,
    FOREIGN KEY (D_id) REFERENCES Department(D_id)
);
```

👨‍🏫 Faculty Table

```
CREATE TABLE Faculty (
    F_ID INT PRIMARY KEY,
    Faculty_Name VARCHAR(100)
);
```

🔁 Faculty Teaches Course

```
CREATE TABLE Teaches (
    F_ID INT,
```

```
    C_id INT,
    FOREIGN KEY (F_ID) REFERENCES Faculty(F_ID),
    FOREIGN KEY (C_id) REFERENCES Course(C_id)
);
```

📝 Exam Table

```
CREATE TABLE Exam (
    Exam_ID INT PRIMARY KEY,
    Subject VARCHAR(100),
    Type VARCHAR(50), -- 'Internal' or 'External'
    Date DATE,
    Duration INT, -- In minutes
    Faculty_id INT,
    C_id INT,
    FOREIGN KEY (Faculty_id) REFERENCES Faculty(F_ID),
    FOREIGN KEY (C_id) REFERENCES Course(C_id)
);
```

📊 Attendance Table

```
CREATE TABLE Attendance (
    S_id INT,
    C_id INT,
    Attendance_Percentage DECIMAL(5,2),
    PRIMARY KEY (S_id, C_id),
    FOREIGN KEY (S_id) REFERENCES Student(S_id),
    FOREIGN KEY (C_id) REFERENCES Course(C_id)
);
```

📌 OPTIONAL: Enrollment Table (Recommended)

```
CREATE TABLE Enrollment (
    S_id INT,
```

```
    C_id INT,
    PRIMARY KEY (S_id, C_id),
    FOREIGN KEY (S_id) REFERENCES Student(S_id),
    FOREIGN KEY (C_id) REFERENCES Course(C_id)
);
```

/* ========== DATA INSERTION ========== */

-- Department
INSERT INTO Department (D_id, D_Name) VALUES
(1, 'Computer Science'),
(2, 'Mechanical Engineering'),
(3, 'Electrical Engineering');

-- Student
INSERT INTO Student (S_id, S_Name, D_id) VALUES
(101, 'Ravi Kumar', 1),
(102, 'Anita Sharma', 2),
(103, 'Mohit Verma', 3);

-- Course
INSERT INTO Course (C_id, C_Name, D_id) VALUES
(201, 'Data Structures', 1),
(202, 'Thermodynamics', 2),
(203, 'Circuit Theory', 3);

-- Faculty
INSERT INTO Faculty (F_ID, Faculty_Name) VALUES
(301, 'Dr. Ramesh'),
(302, 'Prof. Neha'),
(303, 'Dr. Singh');

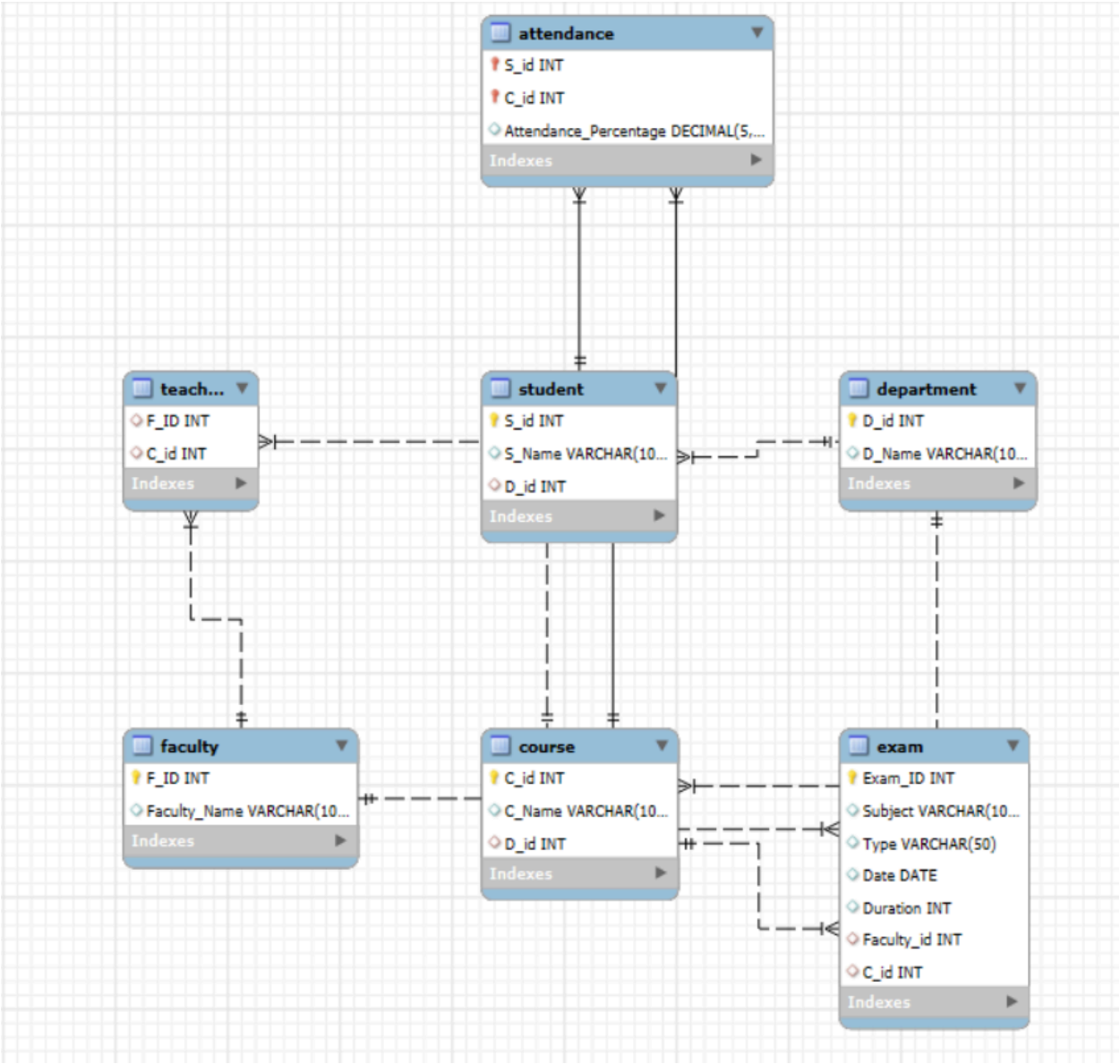
-- Teaches
INSERT INTO Teaches (F_ID, C_id) VALUES
(301, 201),
(302, 202),
(303, 203);

-- Exam
INSERT INTO Exam (Exam_ID, Subject, Type, Date, Duration, Faculty_id, C_id) VALUES
(401, 'Data Structures', 'Internal', '2025-06-20', 90, 301, 201),
(402, 'Thermodynamics', 'External', '2025-06-22', 120, 302, 202),

```
(403, 'Circuit Theory', 'Internal', '2025-06-24', 60, 303, 203);

-- Attendance
INSERT INTO Attendance (S_id, C_id, Attendance_Percentage) VALUES
(101, 201, 85.50),
(102, 202, 78.25),
(103, 203, 92.00);

-- Enrollment
INSERT INTO Enrollment (S_id, C_id) VALUES
(101, 201),
(102, 202),
(103, 203);
```


attendance
S_id INT
C_id INT
Attendance_Percentage DECIMAL(5,...
Indexes
teach...
F_ID INT
C_id INT
Indexes
student
S_id INT
S_Name VARCHAR(10...
D_id INT
Indexes
department
D_id INT
D_Name VARCHAR(10...
Indexes
faculty
F_ID INT
Faculty_Name VARCHAR(10...
Indexes
course
C_id INT
C_Name VARCHAR(10...
D_id INT
Indexes
exam
Exam_ID INT
Subject VARCHAR(10...
Type VARCHAR(50)
Date DATE
Duration INT
Faculty_id INT
C_id INT
Indexes