Database Design Guide

This guide will help the student to create a database on the Student Management System. It will help to manage the below functionalities.

- Student details
- Instructor details
- Course details
- Enrollment details
- Score
- Feedback

We will use MySQL as the DBMS to create the database and its related operations.

1. Introduction to MySQL

MySQL is an open-source relational database management system (RDBMS) that uses structured query language (SQL) to manage and manipulate data in a database. It is widely used for various applications, from small web applications to large enterprise systems.

MySQL's key features include:

- Scalability: Capable of handling large amounts of data and concurrent connections.
- Flexibility: Supports various data types and storage engines.
- Performance: Optimized for speed and efficiency.
- Reliability: Known for its stability and robustness.

2. Installation of MySQL

MySQL can be installed on various operating systems, including Windows, macOS, and Linux. Here are the general steps to install MySQL:

Windows:

- Download the MySQL installer from the official website. https://dev.mysql.com/downloads/installer/
- Run the installer and follow the on-screen instructions.
- Choose the installation type (Typical, Complete, or Custom). Recommended Custom.
- Set a root password for the MySQL server.

3. E-R Diagram (ERD)

An Entity-Relationship Diagram (ERD) is a visual representation of the data model that shows the entities, attributes, relationships between entities, and cardinality. ERDs are commonly used in database design to help developers and stakeholders understand the structure and relationships within a database.

Identify Entities

- Start by identifying the main entities in your system. These are the objects or concepts about which you want to store data.
- Each entity should correspond to a table in your database.

Define Attributes

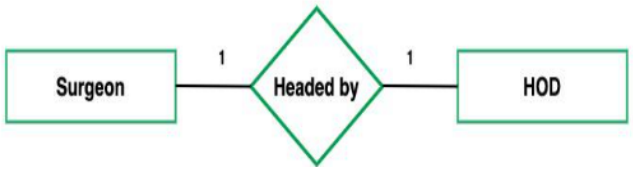
- For each entity, list the attributes (properties or fields) that describe it.
- These attributes will become columns in the corresponding database table.

Identify Relationships

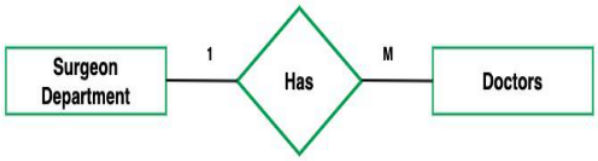
- Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
- Represent these relationships using lines connecting the entities.

Let's see a few examples of relationships:

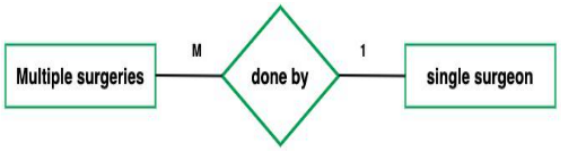
One to One



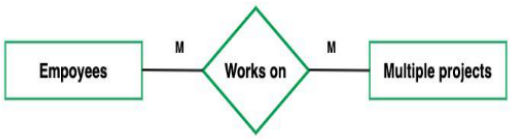
One to Many



Many to One



Many to Many



Cardinality Notation

Cardinality represents the number of times an entity of an entity set participates in a relationship set. Or we can say that the cardinality of a relationship is the number of tuples (rows) in a relationship.

- Use notation (such as Crow's Foot Notation or Chen Notation) to indicate the cardinality of each relationship.
- Cardinality describes how many instances of one entity are related to how many instances of another entity.
- Common notations include:
  - ★ One (1)
  - ★ Zero or one (0..1)
  - ★ Many (N)
  - ★ Zero or many (0..N)

Optional:
Add Attributes and Constraints

- Include additional information in your ERD, such as primary keys, foreign keys, and constraints (e.g., unique constraints).

Create the Diagram

- Use specialized diagramming software or tools (e.g., Lucidchart, draw.io, or even pen and paper) to create your ERD.

Refine and Review:

- Review your ERD with stakeholders and team members to ensure it accurately represents the data model and relationships. Make any necessary refinements.

Let's identify the entities of the Student management system

1. Student
2. Course
3. Instructor
4. Enrollment
5. Score
6. Feedback

*** Now let's identify the attributes and relationships of each entity for the Student Management System.

Student

- **Attributes:**
  StudentID (Primary Key)
  FirstName
  LastName
  DateOfBirth
  Gender
  Email
  Phone

- **Relationships:**
  One **Student** can enroll in more than one **Course** (One-to-Many)

Course

- **Attributes:**
  CourseID (Primary Key)
  CourseTitle
  Credits
- **Relationships:**
  Many **Course** is taught by one **Instructor** (**Many-to-One**)

Instructor

- **Attributes:**
  InstructorID (Primary Key)
  FirstName
  LastName
  Email
- **Relationships:**
  One **Instructor** teaches many **Courses** (**One-to-Many**)
  One **Instructor** has many **Students** (**One-to-Many**)

Enrollment

- **Attributes:**
  EnrollmentID (Primary Key)
  EnrollmentDate
  StudentID(Foreign key)
  CourseID(Foreign Key)
  InstructorID(Foreign key)
- **Relationships:**
  One **Student** maps to Many **Enrolment ids** (**One-to-Many**)
  Many **Enrolment ids** map to one **Course** (**Many-to-One**)

Score

- **Attributes**:
  ScoreID (Primary Key)
  CourseID (Foreign key)
  StudentID (Foreign Key)
  DateOfExam
  CreditObtained

- **Relationships:**
  Many **ScoreIDs** will map to one **Student (Many-to-one)**
  Many **ScoresIDs** will map to one **Course (Many-to-one)**

Feedback

- **Attributes**:
  FeedbackID (Primary Key)

StudentID (Foreign key)
Date
InstructorName
Feedback

- **Relationships**:
  One **Student** will maps to Many **Feedbacks** (**One-to-Many**)

Table Structure

1. Student

```
mysql> Desc Student;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| StudentID   | varchar(10) | NO   | PRI | NULL    |       |
| FirstName   | varchar(25) | NO   |     | NULL    |       |
| LastName    | varchar(25) | NO   |     | NULL    |       |
| DateOfBirth | datetime    | NO   |     | NULL    |       |
| Gender      | varchar(25) | NO   |     | NULL    |       |
| Email       | varchar(30) | NO   | UNI | NULL    |       |
| Phone       | varchar(25) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
7 rows in set (0.03 sec)
```

2. Course

```
mysql> desc Course;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| CourseID    | varchar(10) | NO   | PRI | NULL    |       |
| CourseTitle | varchar(30) | NO   |     | NULL    |       |
| Credits     | int         | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

3. Instructor

```
mysql> desc Instructor;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| InstructorID | varchar(10) | NO   | PRI | NULL    |       |
| Email        | varchar(30) | NO   | UNI | NULL    |       |
| FirstName    | varchar(30) | NO   |     | NULL    |       |
| LastName     | varchar(30) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

4. Enrollment

```
mysql> desc Enrollment;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| EnrollmentID | varchar(10) | NO   | PRI | NULL    |       |
| StudentID    | varchar(10) | NO   | MUL | NULL    |       |
| CourseID     | varchar(10) | NO   | MUL | NULL    |       |
| InstructorID | varchar(10) | NO   | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

5. Score

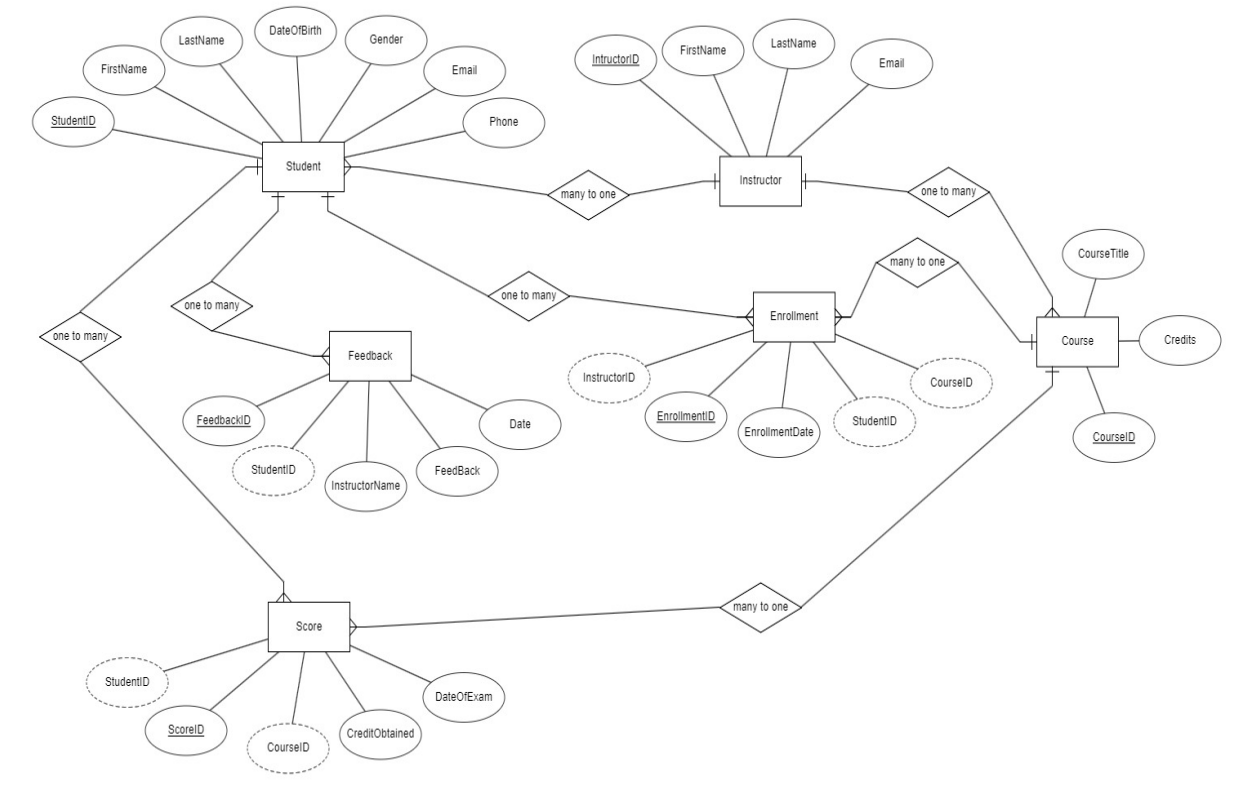
```
Query OK, 0 rows affected (0.03 sec)

mysql> desc score;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| ScoreID        | varchar(10) | NO   | PRI | NULL    |       |
| StudentID      | varchar(10) | NO   | MUL | NULL    |       |
| CourseID       | varchar(10) | NO   | MUL | NULL    |       |
| CreditObtained | varchar(10) | NO   |     | NULL    |       |
| DateOfExam     | datetime    | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

6. Feedback

```
mysql> desc Feedback;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| Feedback_ID    | int          | NO   | PRI | NULL    | auto_increment |
| StudentID      | varchar(10)  | NO   |     | NULL    |                |
| InstructorName | varchar(20)  | NO   |     | NULL    |                |
| Feedback       | varchar(100) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Now, let's create the ER diagram to visually represent the entities and relationships.

ERD Diagram



In this ERD:

- Students can enroll, and each course can have multiple students, creating a many-to-many relationship.

- The Enrollment entity serves as a bridge table between Student and Course entities to represent this relationship.
- Multiple courses can be taught by one Instructor (many-to-one relationship).
- Each Instructor can teach multiple courses (one-to-many relationship).
- A student can give multiple feedbacks
- Student may have scores of multiple courses

4. Creating a Database
Using MySQL server, create a new database for your student management system. You can do this with SQL commands or through the graphical interface.

```
CREATE DATABASE StudentManagementSystem;
```

5. Using a Database
Before performing any operations on a database, you need to select it using the USE statement:

```
USE StudentManagementSystem;
```

6. Creating the tables for each entity

```
USE StudentManagementSystem;

CREATE TABLE Student (
    StudentID VARCHAR(10) PRIMARY KEY,
    FirstName VARCHAR(25) NOT NULL,
   LastName  VARCHAR(25) NOT NULL,
   DateOfBirth DateTime NOT NULL,
   Gender VARCHAR(25) NOT NULL,
   Email VARCHAR(30) UNIQUE NOT NULL,
  Phone VARCHAR(25) NOT NULL
);

CREATE TABLE Course (
    CourseID VARCHAR(10) PRIMARY KEY,
    CourseTitle VARCHAR(30) NOT NULL,
   Credits INT NOT NULL
);

CREATE TABLE Instructor (
   InstructorID VARCHAR(10) PRIMARY KEY,
   Email VARCHAR(30) UNIQUE NOT NULL,
  FirstName VARCHAR(30) NOT NULL,
  LastName VARCHAR(30)
);
```

```
CREATE TABLE Enrollment (
   EnrollmentID VARCHAR(10) PRIMARY KEY,
   StudentID VARCHAR(10) NOT NULL,
   CourseID VARCHAR(10) NOT NULL,
   InstructorID VARCHAR(10) NOT NULL,
   FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
   FOREIGN KEY (CourseID) REFERENCES Course(CourseID),
   FOREIGN KEY (InstructorID) REFERENCES Instructor(InstructorID)
);


CREATE TABLE Score(
        ScoreID VARCHAR(10) PRIMARY KEY,
        StudentID VARCHAR(10) NOT NULL,
        CourseID VARCHAR(10) NOT NULL,
         FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
         FOREIGN KEY (CourseID) REFERENCES Course(CourseID),
        CreditObtained VARCHAR(10) NOT NULL,
        DateOfExam DateTime NOT NULL
);
```

Auto-increment primary key

```
CREATE TABLE Feedback(
        FeedbackID  INT  Auto_Increment PRIMARY KEY,
        StudentID VARCHAR(10) NOT NULL,
        InstructorName VARCHAR(10) NOT NULL,
        Feedback VARCHAR(100) NOT NULL
        );
```

7. Insert records

Add data to your tables to work with. This step helps you test your database.

-- Insert students

```
INSERT INTO Student (StudentID,FirstName,LastName,DateOfBirth,Gender,
Email,Phone) VALUES
   ('S101','John', 'Doe','2000-10-10','M', 'john@example.com','9878457945'),
   ('S102','Jane', 'Smith','2013-08-08','M', 'jane@example.com','9977457745'),
   ('S103','Alice', 'Johnson','2011-09-08','F', 'alice@example.com','9876457845'),
   ('S104','Jim', 'Doe','2011-07-08','F', 'jim.doe@india.com','9876457842'),
   ('S105','Peter', 'Parker','2011-06-05','F', 'p_parker@example.com','9276457843');
```

-- Insert courses

```
INSERT INTO Course (CourseID,CourseTitle,Credits) VALUES
  ('C101','Math101',12),
  ('C102','History101',13),
  ('C103','Computer Science101',11);
```

-- Insert instructor

```
INSERT INTO Instructor (InstructorID ,Email,FirstName,LastName) VALUES
  ('I101','sunil@example.com','Sunil','Rawat'),
  ('I102','nida@example.com','Nida','Fatima'),
  ('I103','shiv@example.com','Shiv','Kumar');
```

-- Enroll students in courses

```
INSERT INTO Enrollment (EnrollmentID,StudentID, CourseID,InstructorID) VALUES
  ('E1001','S101','C101','I101'),
  ('E1002','S102','C101', 'I101'),
  ('E1003','S103','C102','I102');
```

—--Insert Score

```
INSERT INTO Score
(ScoreID,StudentID,CourseID,CreditObtained,DateOfExam)VALUES
('SC101','S101','C101','12','2022-10-10'),
('SC102','S102','C101','10','2022-10-10');
```

— Example for Auto-increment primary key

```
INSERT INTO Feedback(StudentID,InstructorName,Feedback)VALUES
('S101','I101','Session was good'),
('S102','I101','Topic was well explained ');
```

8. Select records

Write SQL queries to retrieve and manage data.

For example:

Retrieve all student:

```
Select * FROM Student;
```

Retrieve a student's enrolled courses:

Select StudentID,StudentName from student where Gender='M';

Select Credits from Course where CourseTitle='History101' ;

***Now try similar Select queries with other tables**

9. Update records

Write SQL statements to update record(s) when needed. For example:

Update a student's email:

UPDATE Students

Update Student SET email = 'johndoe@example.com'
Where StudentName = 'John Doe';

10. Delete records

Write SQL statements to delete record(s) when needed.

Delete FROM Students
Where StudentName = 'John Doe';

PN: Ideally no data should be deleted from any tables. You can use an additional column to set the status of that record to 'Active/Inactive', etc. Or you can use an Archive table to move the unnecessary records out of the main table.