

# ASSIGNMENT NO. 1

## TITLE

Draw E-R diagram and convert entities and relationships to relation table for a given scenario. (eg. bank, college, Employee, Hotel etc)

## PROBLEM STATEMENT

Analyze and design a College Database by drawing its Entity-Relationship (E-R) diagram using standard notations. Convert the identified entities and relationships into relational tables with clear schema. This exercise will help in understanding data modeling and relational schema design for real-world scenarios.

## OBJECTIVE

- Identify entities and their attributes for a college scenario.
- Draw an E-R diagram using standard notations.
- Identify and represent relationships among entities with cardinality.
- Convert entities and relationships into relational tables.
- Understand how to map E-R diagrams to relational models effectively.

## PLATFORM REQUIRED

- **Operating System:** Windows or Linux
- **Software/Tools:** Draw.io, Lucidchart, or manual drawing on paper; SQL environments like MySQL or Oracle for table implementation.

## THEORY

### Entity-Relationship (E-R) Diagram

An **E-R diagram** visually represents entities, their attributes, and the relationships among them in a database system.

### E-R Diagram Components:

- **Entity:** Object that can be uniquely identified (rectangle).
- **Attribute:** Property of an entity (oval).
- **Relationship:** Association between entities (diamond).
- **Primary Key:** Underlined attribute in E-R diagram.

- **Cardinality:** Specifies the number of relationships (1:1, 1:M, M:N) between entities.

### Conversion of E-R Diagram to Relation Tables

The process of transforming E-R diagrams into relational tables involves:

- Creating tables for each entity with primary keys.
- Adding foreign keys for relationships.
- Handling M:N relationships with a separate relation table.
- Maintaining referential integrity.

### SCENARIO: College Database

#### Entities identified:

1. **Student** (Student\_ID, Name, Address, Phone)
2. **Course** (Course\_ID, Course\_Name, Credits)
3. **Enrollment** (Student\_ID, Course\_ID, Enrollment\_Date)

#### Relationships identified:

- **Enrolls:** Students enroll in Courses (M:N).

### E-R DIAGRAM (NOTATIONS)

- **Rectangles:** This Entity Relationship Diagram symbol represents entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes



**Figure 1.** ER Diagram Symbols

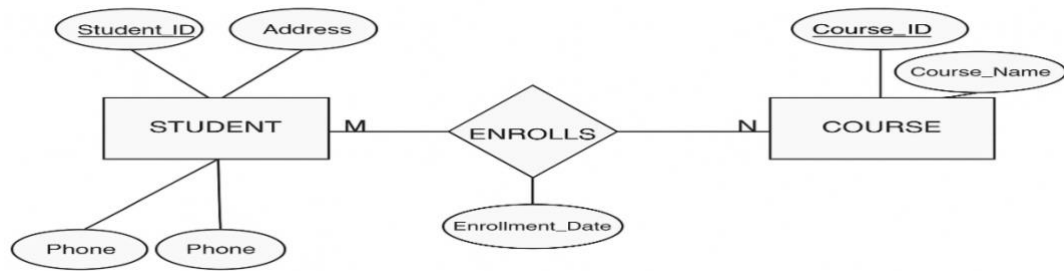


Figure 2. College Database ER Diagram

## RELATION TABLES

### 1. Student

Column Name	Data Type	Constraint
Student_ID	INT	PRIMARY KEY
Name	VARCHAR	NOT NULL
Address	VARCHAR	
Phone	VARCHAR	

### 2. Course

Column Name	Data Type	Constraint
Course_ID	INT	PRIMARY KEY
Course_Name	VARCHAR	NOT NULL
Credits	INT	

### 3. Enrollment

(To handle M:N relationship)

Column Name	Data Type	Constraint
Student_ID	INT	FOREIGN KEY REFERENCES Student
Course_ID	INT	FOREIGN KEY REFERENCES Course
Enrollment_Date	DATE	

**Primary Key** (Student\_ID, Course\_ID) Composite PK

## STEP BY STEP ALGORITHM

1. Identify entities and their attributes from the scenario.
2. Determine primary keys for each entity.
3. Identify relationships and their cardinalities.
4. Draw the E-R diagram using standard notations.
5. Convert entities into relational tables with appropriate data types.
6. Convert relationships into foreign keys or relation tables for M:N.
7. Check referential integrity between foreign keys and primary keys.

## QUESTIONS

1. What is an E-R diagram, and why is it used?
2. What are the components of an E-R diagram with examples?
3. Explain how cardinality is represented in E-R diagrams.
4. How do you handle M:N relationships while converting to relation tables?
5. What is the importance of primary keys and foreign keys in relation tables?
6. Convert the “college” E-R diagram into relational tables and explain each table.
7. What is referential integrity, and why is it important in relational databases?

## CONCLUSION

This assignment demonstrates the practical application of designing E-R diagrams and converting them into relational tables using a college database scenario. The exercise reinforces understanding of **entities, relationships, cardinality, and mapping to SQL-based relational structures**, preparing you for **database design and normalization in practical environments**. Observations may vary depending on attribute selection, but the principles of correct mapping and referential integrity remain constant.

# ASSIGNMENT NO. 2

## TITLE

To Perform SQL Activities:

- a) Creating a Database
- b) Creating Tables (With and Without Constraints)
- c) Inserting Records into Tables

## PROBLEM STATEMENT

Perform practical SQL activities to understand database creation, table creation with and without constraints, and data insertion. This assignment will help you understand the fundamental concepts of database structure and data manipulation using SQL.

## OBJECTIVE

- Learn to create a database using SQL commands.
- Create tables without constraints for basic structure understanding.
- Create tables with constraints (PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK, DEFAULT) to enforce data integrity.
- Insert records using INSERT INTO statements.
- Understand the difference between table creation with and without constraints.
- Retrieve and verify records using SELECT queries.

## PLATFORM REQUIRED

- Operating System: Windows/Linux
- Software/Tools: MySQL, Oracle, PostgreSQL, SQL Server
- GUI Tools: MySQL Workbench, Oracle SQL Developer, pgAdmin

## THEORY

### SQL

Structured Query Language (SQL) is used to communicate with relational databases to create, manage, and manipulate data using:

- DDL (Data Definition Language): For creating and modifying structures (CREATE, ALTER, DROP).

- DML (Data Manipulation Language): For data handling (INSERT, UPDATE, DELETE, SELECT).

## Constraints in SQL

Constraints are rules applied to columns in a table to maintain the accuracy and integrity of the data:

1. PRIMARY KEY: Uniquely identifies each record.
2. FOREIGN KEY: Maintains referential integrity by linking two tables.
3. NOT NULL: Ensures a column cannot store NULL.
4. UNIQUE: Ensures all values in a column are distinct.
5. CHECK: Ensures values meet a specific condition.
6. DEFAULT: Assigns a default value if no value is provided during insertion.

## STEP BY STEP PROCEDURE

### a) Creating a Database

```
CREATE DATABASE CollegeDB;
```

```
USE CollegeDB;
```

### b) Creating Tables

Without Constraints:

```
CREATE TABLE Department (  
    DeptID INT,  
    DeptName VARCHAR(50)  
);
```

With Constraints:

```
CREATE TABLE Student (  
    RollNo INT PRIMARY KEY,           -- PRIMARY KEY  
    Name VARCHAR(50) NOT NULL,        -- NOT NULL  
    Age INT CHECK (Age >= 17 AND Age <= 30), -- CHECK  
    Email VARCHAR(100) UNIQUE,         -- UNIQUE  
    DeptID INT,  
    Fees DECIMAL(10,2) DEFAULT 50000.00, -- DEFAULT
```

```
FOREIGN KEY (DeptID) REFERENCES Department(DeptID) -- FOREIGN KEY
);
```

#### c) Inserting Records into Tables

Insert into Department:

```
INSERT INTO Department (DeptID, DeptName) VALUES (1, 'Computer Science');
```

```
INSERT INTO Department (DeptID, DeptName) VALUES (2, 'Information Technology');
```

Insert into Student:

```
INSERT INTO Student (RollNo, Name, Age, Email, DeptID, Fees)
```

```
VALUES (101, 'Amit Sharma', 20, 'amit@gmail.com', 1, 55000.00);
```

```
INSERT INTO Student (RollNo, Name, Age, Email, DeptID)
```

```
VALUES (102, 'Pooja Singh', 19, 'pooja@gmail.com', 2);
```

-- Fees will take the DEFAULT value of 50000.00

Verifying Inserted Records

```
SELECT * FROM Department;
```

```
SELECT * FROM Student;
```

### QUESTIONS

1. What command is used to create a new database in SQL?
2. Differentiate between creating tables with and without constraints.
3. Write the SQL command to insert a record into a table.
4. Explain the purpose of the CHECK constraint with an example.
5. What will happen if you try to insert a duplicate value in a column with a UNIQUE constraint?
6. How does the FOREIGN KEY constraint ensure referential integrity between two tables?
7. Why is the DEFAULT constraint useful during insertion?

## **CONCLUSION**

In this assignment, we practiced creating a database and designing tables using SQL. We created tables without constraints to learn the basic structure and with constraints to ensure data integrity. Using constraints like PRIMARY KEY, FOREIGN KEY, NOT NULL, UNIQUE, CHECK, and DEFAULT helped maintain consistency and accuracy. We inserted records into these tables and verified them using SELECT queries. This activity improved our understanding of SQL commands and their practical application. It also highlighted the importance of constraints in managing structured data within databases.