Database Design Guide

This guide will help the student to create a database for a eCommerce Application. It will help to manage the below functionalities: -

- Products
- Customers
- Login_Details
- Orders
- Payments
- Suppliers

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

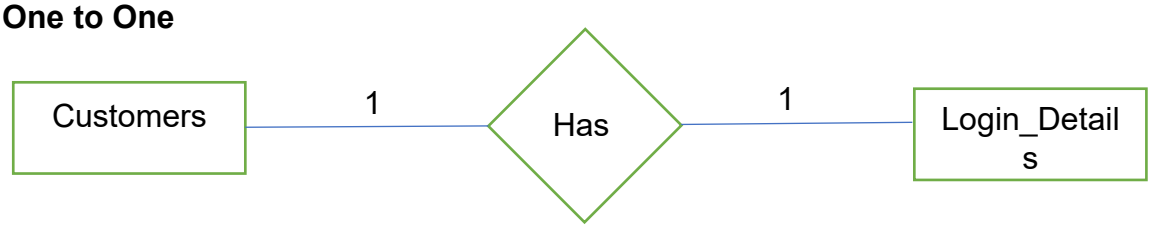
- For each entity, list the attributes (properties or fields) that describe it.
- These attributes will become columns in the corresponding database table.

Identify Relationships

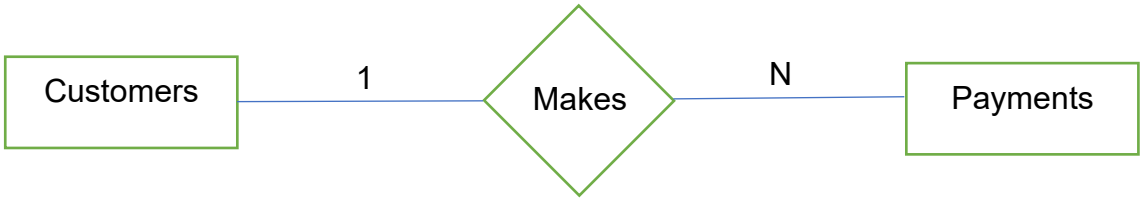
- Determine how entities are related to each other. There are three types of relationships: one-to-one (1:1), one-to-many (1:N), and many-to-many (N:M).
- Represent these relationships using lines connecting the entities.

Let's see a few examples of relationships:

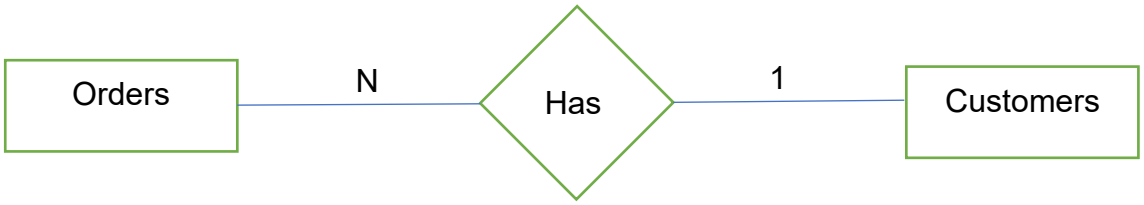
One to One



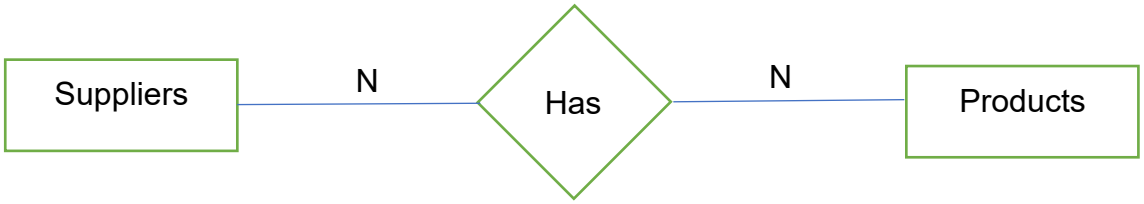
One to Many



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 eCommerce Application:

- Products
- Customers
- Login_Details
- Orders
- Payments
- Suppliers

*** Now let's identify the attributes and relationships of each entity for the eCommerce Application.

<u>Suppliers</u>

- **Attributes:**
  - Supplier_Id(Primary Key)
  - Supplier_Name

<u>Products</u>

- **Attributes:**
  - Product_Id(Primary Key)
  - Product_Name
  - Product_Cost
  - Supplier_Id(Foreign Key)
- **Relationships:**
  - One **Product** has One **Supplier** (**One-to-One**) (based on "Product_Id")
  - One **Product** has One **Payment** (**One-to-One**)
  - One **Product** has Many **Customers** (**One-to-Many**)
  - One **Product** has Many **Orders** (**One-to-Many**)

Customers

- **Attributes:**
  - Customer_Id(Primary Key)
  - Customer_Name
  - Customer_DOB
  - Customer_Address

Login_Details

- **Attributes:**
  - Username(Primary Key)
  - Password
  - Customer_Id(Foreign Key)
- **Relationships:**
  - One **Login_Detail** has One **Customer** (**One-to-One**)

Payments

- **Attributes:**
  - Payment_Id(Primary Key)
  - Product_Id(Foreign Key)
  - Mode_of_Payment

  Relationships:
  - One **Payment** has One **Product** (**One-to-One**)

Orders

- **Attributes:**
  - Order_Id(Primary Key)
  - Customer_Id(Foreign Key)
  - Payment_Id(Foreign Key)
- **Relationships:**
  - One **Order** has One **Customer** (**One-to-One**)
  - One **Order** has One **Payment** (**One-to-One**)

Table Structure

1. Suppliers

```
mysql> desc suppliers;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Supplier_Id   | int         | NO   | PRI | NULL    |       |
| Supplier_Name | varchar(60) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.06 sec)
```

2. Products

```
mysql> desc products;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Product_Id   | int         | NO   | PRI | NULL    |       |
| Product_Name | varchar(60) | YES  |     | NULL    |       |
| Product_Cost | int         | YES  |     | NULL    |       |
| Supplier_Id  | int         | YES  | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3. Customers

```
mysql> desc Customers;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| Customer_Id      | int          | NO   | PRI | NULL    |       |
| Customer_Name    | varchar(60)  | YES  |     | NULL    |       |
| Customer_DOB     | date         | YES  |     | NULL    |       |
| Customer_Address | varchar(100) | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

4. Login_Details

```
mysql> desc Login_Details;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Username    | varchar(30) | YES  |     | NULL    |       |
| Password    | varchar(30) | YES  |     | NULL    |       |
| Customer_Id | int         | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

5. Payments

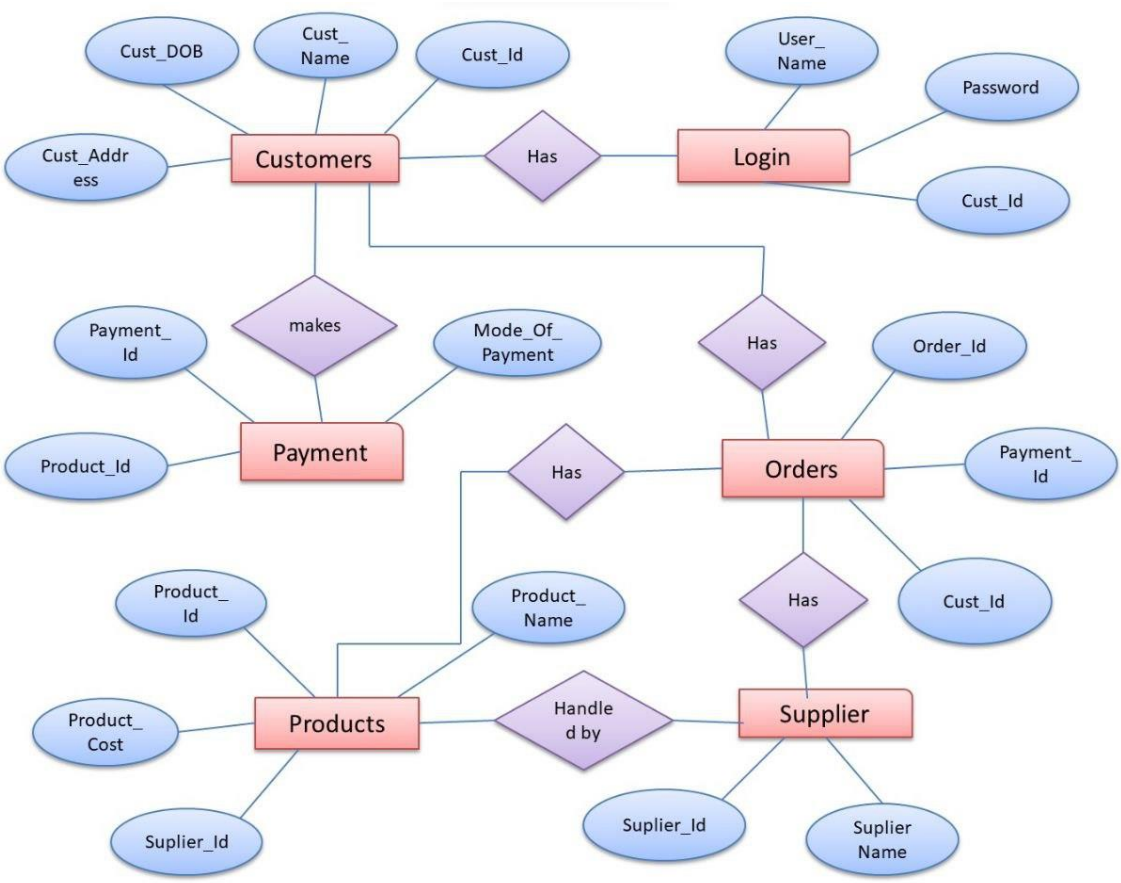
```
mysql> desc Payments;
+-----------------+------------+------+-----+---------+-------+
| Field           | Type       | Null | Key | Default | Extra |
+-----------------+------------+------+-----+---------+-------+
| Payment_Id      | int        | NO   | PRI | NULL    |       |
| Product_Id      | int        | YES  | MUL | NULL    |       |
| Mode_of_Payment | varchar(6) | YES  |     | NULL    |       |
+-----------------+------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

6. Orders

```
mysql> desc Orders;
+-------------+------+------+-----+---------+-------+
| Field       | Type | Null | Key | Default | Extra |
+-------------+------+------+-----+---------+-------+
| Order_Id    | int  | NO   | PRI | NULL    |       |
| Customer_Id | int  | YES  | MUL | NULL    |       |
| Payment_Id  | int  | YES  | MUL | NULL    |       |
+-------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

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

ERD Diagram



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 eCommerceBiz;

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

USE eCommerceBiz;

6. Creating the tables for each entity

```
create table suppliers
   -> (Supplier_Id int primary key,
   -> Supplier_Name varchar(60));

create table Products
   -> (Product_Id int primary key,
   -> Product_Name varchar(60),
   -> Product_Cost int,
   -> Supplier_Id int,
   -> foreign key(Supplier_Id) references Suppliers(Supplier_Id));

create table Customers
   -> (Customer_Id int primary key,
   -> Customer_Name varchar(60),
   -> Customer_DOB date,
   -> Customer_Address varchar(100));

create table Login_Details
   -> (Username varchar(30),
   -> Password varchar(30),
   -> Customer_Id int,
   -> foreign key (Customer_Id) references Customers(Customer_Id));

create table Payments
   -> (Payment_Id int primary key,
   -> Product_Id int,
   -> Mode_of_Payment varchar(6),
   -> foreign key (Product_Id) references Products(Product_Id));

create table Orders
   -> (Order_Id int primary key,
   -> Customer_Id int,
   -> Payment_Id int,
   -> foreign key (Customer_Id) references Customers(Customer_Id),
   -> foreign key (Payment_Id) references Payments(Payment_Id));
```

7. Insert records

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

```
-- INSERT Suppliers
INSERT INTO Suppliers(Supplier_Id,Supplier_Name) VALUES
   -> (1,'Pramod Shinde'),
   -> (2,'Abhishek Gaikwad'),
   -> (3,'Pranjali Ingale'),
   -> (4,'Govind Patil'),
   -> (5,'Pravin Dhatrak');

-- INSERT Products
INSERT INTO Products(Product_Id, Product_Name, Product_Cost,Supplier_Id) VALUES
   -> (1,'Vortex Bluetooth Earphones',499,1),
   -> (2,'USHA Table Fan',8500,5),
   -> (3,'Learn Java (Book)',675,3),
   -> (4,'Pillow Cover Set',1200,2),
   -> (5,'HP Pavilion Laptop',63470,5),
   -> (6,'CASIO Calculator',600,4);

-- INSERT Customers
INSERT INTO Customers (Customer_Id, Customer_Name, Customer_DOB,
Customer_Address) VALUES
   -> (1, 'Yash Palde','2001-09-22','At Post. Gangawadi, Nashik'),
   -> (2,'Mayur Patil','2001-12-22','At Post. Chehadi Pumping, near Sinnar Phata, Nashik'),
   -> (3,'Harshal Kumawat','2001-08-17','Plot B3, Nandur Naka, Nashik'),
   -> (4,'Sandip Sarukte','1997-11-27','Flat No. 8, Sai Apartments, Amrutdham, Nashik'),
   -> (5,'Nikhil Chaudhari','1999-11-26','Hari Krupa Bungalow, Mumbai Naka, Nashik');

-- INSERT Login_Details
INSERT INTO Login_Details (Username, Password, Customer_Id) VALUES
   -> ('yashpalde3691','yash@123',1),
   -> ('mayur_201_patil','patil#201',2),
   -> ('harshal6531','harshk@#1',3),
   -> ('sandipSarukte','sarukte_Sandip1997',4),
   -> ('nikChaudhari21','hiddenNik@21',5);

-- INSERT Payments
INSERT INTO Payments (Payment_Id,Product_Id,Mode_of_Payment) VALUES
   -> (1101,4,'Credit'),
   -> (974,3,'UPI'),
   -> (1363,6,'Credit'),
   -> (2263,1,'Cash'),
   -> (5611,5,'Cash'),
   -> (9144,2,'UPI');
```

-- INSERT Orders

```
INSERT INTO Orders (Order_Id, Customer_Id, Payment_Id) VALUES
   -> (143,1,9144),
   -> (38,5,5611),
   -> (467,4,974),
   -> (220,3,2263),
   -> (89,3,1363),
   -> (127,2,1101);
```

8. Select records

Write SQL queries to retrieve and manage data.

For example:

Retrieve all Customers:

```
SELECT * FROM Customers;
```

Retrieve name of Customer along with, Product ordered, Cost of product and mode of payment:

```
SELECT cust.Customer_Name,pro.Product_Name,pro.Product_Cost,pay.Mode_of_Payment
   -> FROM Customers cust, Products pro, Payments pay, Orders o
   -> WHERE (o.Payment_Id = pay.Payment_Id) AND (o.Customer_Id = cust.Customer_Id)
AND (pay.Product_Id = pro.Product_Id);
```

9. Update records

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

Update a Customer's address:

UPDATE Customers

```
UPDATE Customers SET Customer_Address = 'At Post. Mohu, Sinnar, Nashik'
WHERE Customer_Id = 1;
```

10. Delete records

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

```
DELETE FROM Customers
WHERE Customer_Id = 1;
```

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.