**DATABASE MANGAEMNET INTERN**

as a part of

presenting TASK 2

Submitted By:

Shafak

Date: 14 Jan. 25

**Develop a database for entity movie rental. This project involves more queries and database design. Write SQL queries to handle customer orders.**

Designing and implementing a relational database for a **Movie Rental System** involves several steps, including defining the requirements, designing the schema, and implementing the database. Below is a comprehensive guide to help you through this process:

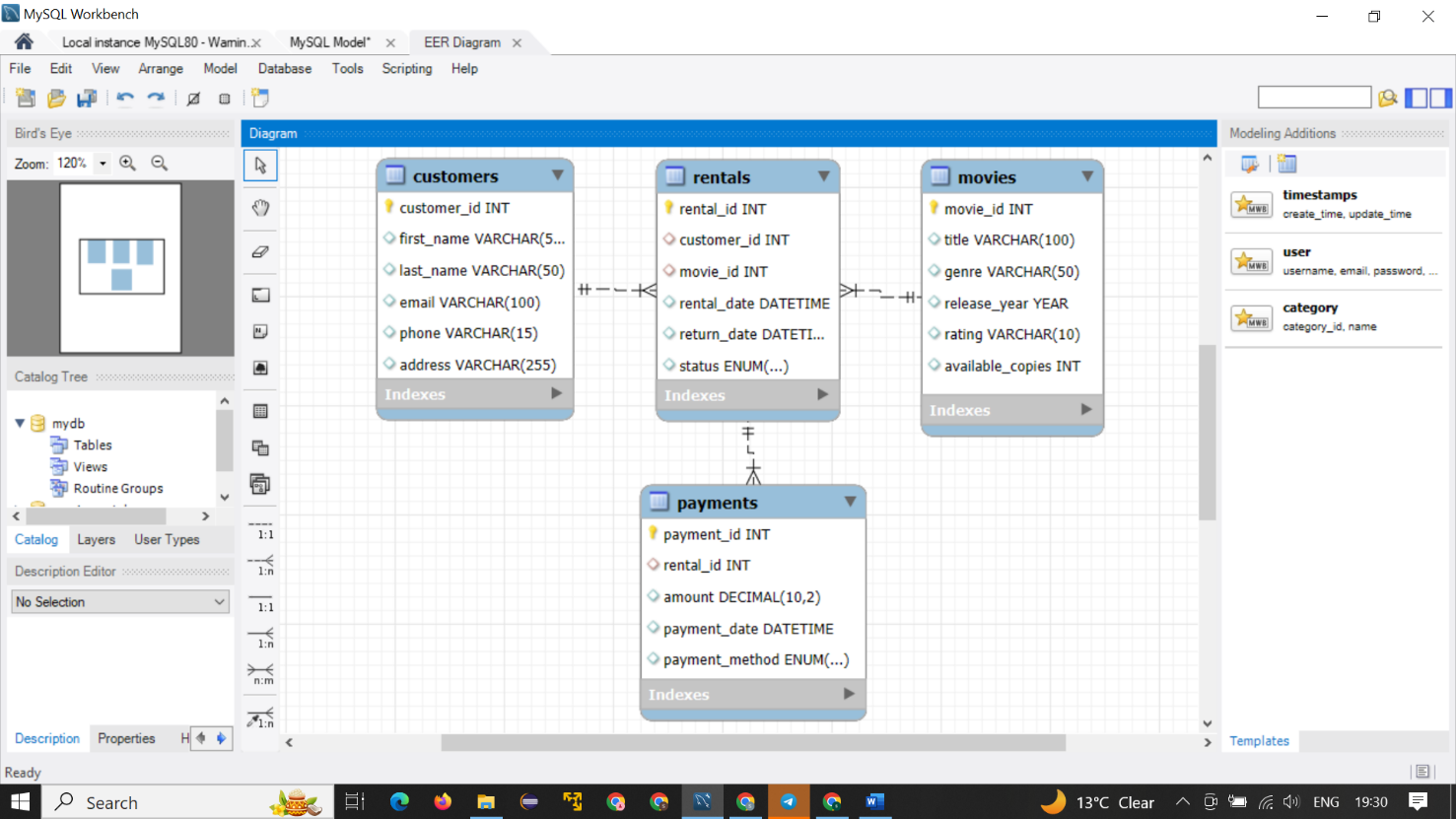
**Step 1: Requirements Gathering**

Before designing the database, we need to identify the key entities, their attributes, and the relationships between them. Here are the common requirements for a Movie Rental System:

**Key Entities and Their Attributes**

1. **Movies**
   * **Attributes:**
     + **movie\_id:** Unique identifier for each movie (Primary Key)
     + **title:** Title of the movie
     + **genre:** Genre of the movie (e.g., Action, Comedy, Drama)
     + **release\_year:** Year the movie was released
     + **rating:** Rating of the movie (e.g., PG, PG-13, R)
     + **available\_copies:** Number of copies available for rent
     + **duration:** Duration of the movie (in minutes)
     + **description:** Brief description or synopsis of the movie
2. **Customers**
   * **Attributes:**
     + **customer\_id:** Unique identifier for each customer (Primary Key)
     + **first\_name:** First name of the customer
     + **last\_name**: Last name of the customer
     + **email:** Email address of the customer (Unique)
     + **phone:** Contact number of the customer
     + **address:** Residential address of the customer
     + **membership\_status:** Status of the customer's membership (e.g., Active, Inactive)
3. **Rentals**
   * **Attributes:**
     + **rental\_id:** Unique identifier for each rental transaction (Primary Key)
     + **customer\_id:** Identifier for the customer who rented the movie (Foreign Key)
     + **movie\_id:** Identifier for the rented movie (Foreign Key)
     + **rental\_date:** Date and time when the movie was rented
     + **return\_date:** Date and time when the movie was returned
     + **due\_date:** Date by which the movie should be returned
     + **status:** Status of the rental (e.g., Rented, Returned, Overdue)
4. **Payments**
   * **Attributes:**
     + **payment\_id:** Unique identifier for each payment transaction (Primary Key)
     + **rental\_id:** Identifier for the rental associated with the payment (Foreign Key)
     + **amount:** Amount paid for the rental
     + **payment\_date:** Date and time when the payment was made
     + **payment\_method:** Method of payment (e.g., Credit Card, PayPal, Cash)
5. **Categories**
   * **Attributes:**
     + **category\_id:** Unique identifier for each category (Primary Key)
     + **category\_name:** Name of the category (e.g., Action, Comedy, Drama)
6. **Movie\_Categories (Join Table)**
   * **Attributes:**
     + **movie\_id:** Identifier for the movie (Foreign Key)
     + **category\_id:** Identifier for the category (Foreign Key)

**Relationships**

1. **Movies and Categories**
   * A movie can belong to multiple categories, and a category can have multiple movies. This is a many-to-many relationship, which will be implemented using a join table called Movie\_Categories.
2. **Customers and Rentals**
   * A customer can rent multiple movies, but each rental is associated with only one customer. This is a one-to-many relationship.
3. **Movies and Rentals**
   * A movie can be rented multiple times, but each rental is associated with only one movie. This is a one-to-many relationship.
4. **Rentals and Payments**
   * Each rental can have one payment associated with it, but a payment is linked to only one rental. This is a one-to-one relationship.
5. **Movies and Actors (Optional)**
   * If we want to include actors, we can add an Actors table with a many-to-many relationship to Movies.

**Step 2: Database Schema Design**

Based on the requirements gathered, we can design the database schema. Below is the SQL code to create the necessary tables:

1. **Create the Database**

CREATE DATABASE movie\_rental;

USE movie\_rental;

1. **Create the Tables**

CREATE TABLE Customers (

customer\_id INT AUTO\_INCREMENT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone VARCHAR(15),

address VARCHAR(255)

membership\_status ENUM('Active', 'Inactive')

);

CREATE TABLE Movies (

movie\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(100),

genre VARCHAR(50),

release\_year YEAR,

rating VARCHAR(10),

available\_copies INT

);

CREATE TABLE Rentals (

rental\_id INT AUTO\_INCREMENT PRIMARY KEY,

customer\_id INT,

movie\_id INT,

rental\_date DATETIME,

return\_date DATETIME,

status ENUM('rented', 'returned'),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id),

FOREIGN KEY (movie\_id) REFERENCES Movies(movie\_id)

);

CREATE TABLE Payments (

payment\_id INT AUTO\_INCREMENT PRIMARY KEY,

rental\_id INT,

amount DECIMAL(10, 2),

payment\_date DATETIME,

payment\_method ENUM('credit\_card', 'paypal', 'cash'),

FOREIGN KEY (rental\_id) REFERENCES Rentals(rental\_id)

);

1. **Insert Sample Data**

**Insert Customers**

INSERT INTO Customers (first\_name, last\_name, email, phone, address) VALUES

('John', 'Doe', 'john.doe@example.com', '1234567890', '123 Elm St'),

('Jane', 'Smith', 'jane.smith@example.com', '0987654321', '456 Oak St');

**Insert Movies**

INSERT INTO Movies (title, genre, release\_year, rating, available\_copies) VALUES

('Inception', 'Sci-Fi', 2010, 'PG-13', 5),

('The Godfather', 'Crime', 1972, 'R', 3),

('The Dark Knight', 'Action', 2008, 'PG-13', 4);

**Insert Rentals**

INSERT INTO Rentals (customer\_id, movie\_id, rental\_date, status) VALUES

(1, 1, NOW(), 'rented'),

(2, 2, NOW(), 'rented');

**Insert Payments**

INSERT INTO Payments (rental\_id, amount, payment\_date, payment\_method) VALUES

(1, 4.99, NOW(), 'credit\_card'),

(2, 3.99, NOW(), 'paypal');

**Step 3. Queries to Handle Customer Orders**

1. **Get All Movies Available for Rent**

SELECT \* FROM Movies WHERE available\_copies > 0;

1. **Rent a Movie**

Assuming customer\_id and movie\_id are provided

SET @customer\_id = 1;

SET @movie\_id = 1;

Check if the movie is available

SELECT available\_copies FROM Movies WHERE movie\_id = @movie\_id;

If available, insert rental and update available copies

INSERT INTO Rentals (customer\_id, movie\_id, rental\_date, status) VALUES

(@customer\_id, @movie\_id, NOW(), 'rented');

UPDATE Movies SET available\_copies = available\_copies - 1 WHERE movie\_id = @movie\_id;

1. **Update Customer Information**

UPDATE Customers

SET phone = '555-5678', address = '101 Maple St'

WHERE customer\_id = 1; **-- Assuming customer\_id 1 is the customer to update**

1. **Retrieve All Customers**

SELECT \* FROM Customers;

1. **Retrieve a Specific Customer by ID**

SELECT \* FROM Customers

WHERE customer\_id = 1; -- Replace with the desired customer\_id

1. **Retrieve Customers with Active Membership**

SELECT \* FROM Customers

WHERE membership\_status = 'Active';

1. **Retrieve Customer Count by Membership Status**

SELECT membership\_status, COUNT(\*) AS count

FROM Customers

GROUP BY membership\_status;