Experiment No.2
Mapping ER/EER to Relational schema model.
Name:-Bhagyashri Kaleni Sutar
Roll No:-75
Date of Performance:
Date of Submission:

Aim :- Prepare the schema for Relational Model with the ER/ERR diagram, drawn for the identified case study in experiment no.1.

Objective :- To map the Entity Relationship (ER) / Extended Entity-Relationship (EER) Diagram to Relational Model schema and learn to incorporate various schema-based constraints.

Theory:

Mapping an Entity-Relationship (ER) model to a relational database schema involves translating the conceptual model represented in the ER diagram into tables and relationships in a relational database management system (DBMS). Here are the general rules for mapping ER to a schema in a DBMS:

1. Entities to Tables:
    a. Each entity in the ER diagram corresponds to a table in the relational schema.
    b. The attributes of the entity become the columns of the table.
    c. The primary key of the entity becomes the primary key of the table.

2. Relationships to Tables:
    a. Many-to-Many Relationships:
        i. Convert each many-to-many relationship into a new table.
        ii. Include foreign key columns in this table to reference the participating entities.
        iii. The primary key of this table may consist of a combination of the foreign keys from the participating entities.
    b. One-to-Many and One-to-One Relationships:
        i. Represented by foreign key columns in one of the participating tables.
        ii. The table on the "many" side of the relationship includes the foreign key column referencing the table on the "one" side.
        iii. The foreign key column typically references the primary key of the related table.

3. Attributes to Columns:
    a. Each attribute of an entity becomes a column in the corresponding table.
    b. Choose appropriate data types for each attribute based on its domain and constraints.
    c. Ensure that attributes participating in relationships are represented as foreign keys when needed.

4. Primary and Foreign Keys:
    a. Identify the primary key(s) of each table based on the primary key(s) of the corresponding entity.
    b. Ensure referential integrity by defining foreign keys in tables to establish relationships between them.

c. Foreign keys should reference the primary key(s) of related tables.

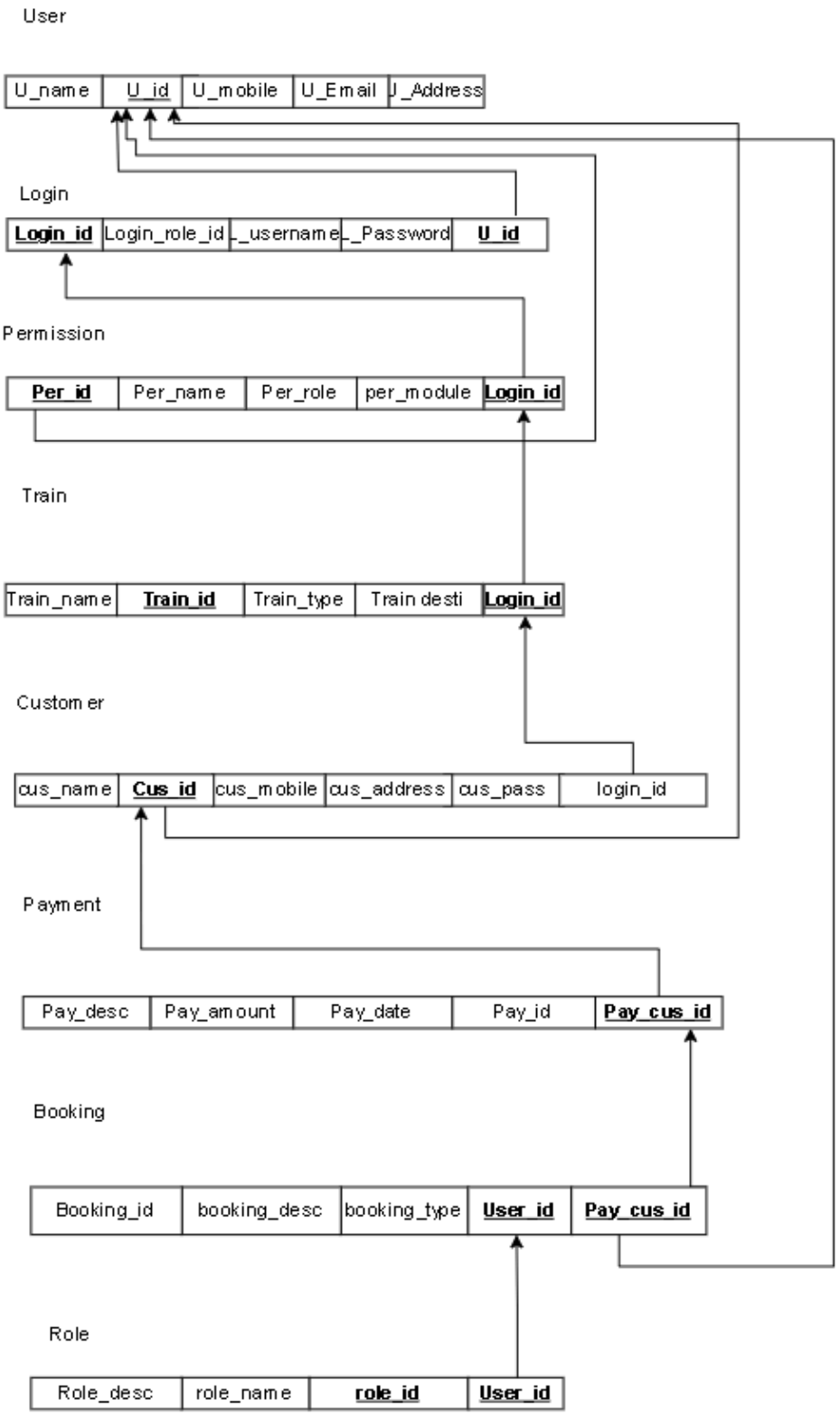
d. Ensure that foreign keys have appropriate constraints, such as ON DELETE CASCADE or ON UPDATE CASCADE, to maintain data integrity.

5. Cardinality Constraints:
   a. Use the cardinality constraints from the ER diagram to determine the multiplicity of relationships in the relational schema.
   b. Ensure that the constraints are enforced through the appropriate use of primary and foreign keys.

6. Normalization:
   a. Normalize the schema to minimize redundancy and dependency.
   b. Follow normalization rules such as First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), etc., to ensure data integrity and minimize anomalies.

7. Indexing and Optimization:
   a. Consider indexing frequently queried columns to improve query performance.
   b. Evaluate the schema design for optimization opportunities based on query patterns and performance requirements.

Implementation


User
U_name
U_id
U_mobile
U_Email
U_Address
Login
Login_id
Login_role_id
L_username
L_Password
U_id
Permission
Per_id
Per_name
Per_role
per_module
Login_id
Train
Train_name
Train_id
Train_type
Train desti
Login_id
Customer
cus_name
Cus_id
cus_mobile
cus_address
cus_pass
login_id
Payment
Pay_desc
Pay_amount
Pay_date
Pay_id
Pay_cus_id
Booking
Booking_id
booking_desc
booking_type
User_id
Pay_cus_id
Role
Role_desc
role_name
role_id
User_id

Conclusion

a. write definition of relational schema and notations

Ans:-

A **relational schema** is a structural definition that outlines how data is organized in a relational database. It describes the tables (also known as relations), the attributes (columns) within those tables, the data types for each attribute, and the relationships between tables. The relational schema ensures that the data is stored in an efficient, organized manner while maintaining integrity through constraints like primary keys, foreign keys, and other rules.

In simple terms, it is the blueprint for how the database is structured, specifying what tables will exist, what each table's columns will represent, and how these tables relate to one another. It does not contain actual data but defines the organization of the data.

Notation refers to a system of symbols or signs used to represent concepts, objects, or operations in a particular field of study. It is a formalized way of writing down information to make it easier to communicate and understand complex ideas, processes, or structures.

b. write various schema-based constraints

Ans:-

1.Data Integrity constraint:-

A **data integrity constraint** is a rule or condition applied to a database to ensure the accuracy, consistency, and validity of the data stored in it. These constraints are enforced by the database management system (DBMS) and are crucial for maintaining the reliability and correctness of the data as it is inserted, updated, or deleted in the database.

There are several types of data integrity constraints, each addressing different aspects of data integrity:

a. Entity Integrity Constraints

- **Definition**: Ensures that each row (or record) in a table can be uniquely identified by a primary key. The primary key must be unique and cannot contain `NULL` values.
- **Purpose**: This constraint guarantees that each entity in a table is distinct and identifiable by a unique key.
- **Example**: In a `Student` table, the `StudentID` serves as the primary key and must have a unique, non-null value for each student.
- **Example in SQL**:

```
CREATE TABLE Student (

StudentID INT PRIMARY KEY,

Name VARCHAR(100),

  Age INT

);
```

2. Domain Integrity Constraints

- **Definition**: Ensures that the data entered into a database column falls within a valid range or set of values. It restricts the values that an attribute can take by defining a domain.
- **Purpose**: This ensures that each attribute in a table adheres to the correct type, range, or predefined list of acceptable values.
- **Example**: In a `Student` table, the `Age` attribute must be an integer between 18 and 100.

Example in SQL:

```
CREATE TABLE Student (
StudentID INT PRIMARY KEY,
Name VARCHAR(100),
Age INT CHECK (Age BETWEEN 18 AND 100)
);
```

a. Null Integrity Constraints

- **Definition**: Ensures whether a column allows `NULL` values or not. A column can either have `NULL` values (meaning no data) or not, depending on the constraints defined.
- **Purpose**: This guarantees that a column can either accept `NULL` values (when there is no value for that attribute) or require non-null values (when the attribute is mandatory).
- **Example**: In a `Student` table, if the `Name` column is defined with a `NOT NULL` constraint, every student must have a name.

Example in SQL:

```
CREATE TABLE Student (
StudentID INT PRIMARY KEY,
Name VARCHAR(100) NOT NULL
);
```

b.Check Constraints

- **Definition**: Ensures that data entered into a column satisfies a specific condition or boolean expression. If the condition is not met, the operation is rejected.

- **Purpose**: This constraint validates the data before it is inserted or updated, ensuring that it meets specific conditions (e.g., value range, format, etc.).
- **Example**: In a `Employee` table, the `Salary` attribute must be greater than zero.

Example in SQL:

```
CREATE TABLE Employee (
 EmployeeID INT PRIMARY KEY,
 Name VARCHAR(100),
 Salary DECIMAL(10, 2),
 CHECK (Salary > 0)
 );
```

c.Default Constraints

- **Definition**: Specifies a default value for a column when no value is provided during an insert operation.
- **Purpose**: This ensures that a column automatically gets a default value if no explicit value is provided by the user during insertion.
- **Example**: If no value is provided for the `HireDate` attribute in an `Employee` table, it will default to the current date.

Example in SQL:

```
CREATE TABLE Employee (
EmployeeID INT PRIMARY KEY,
Name VARCHAR(100),
HireDate DATE DEFAULT CURRENT_DATE
 );
```