Experiment No.2
Mapping ER/EER to Relational schema model.
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.

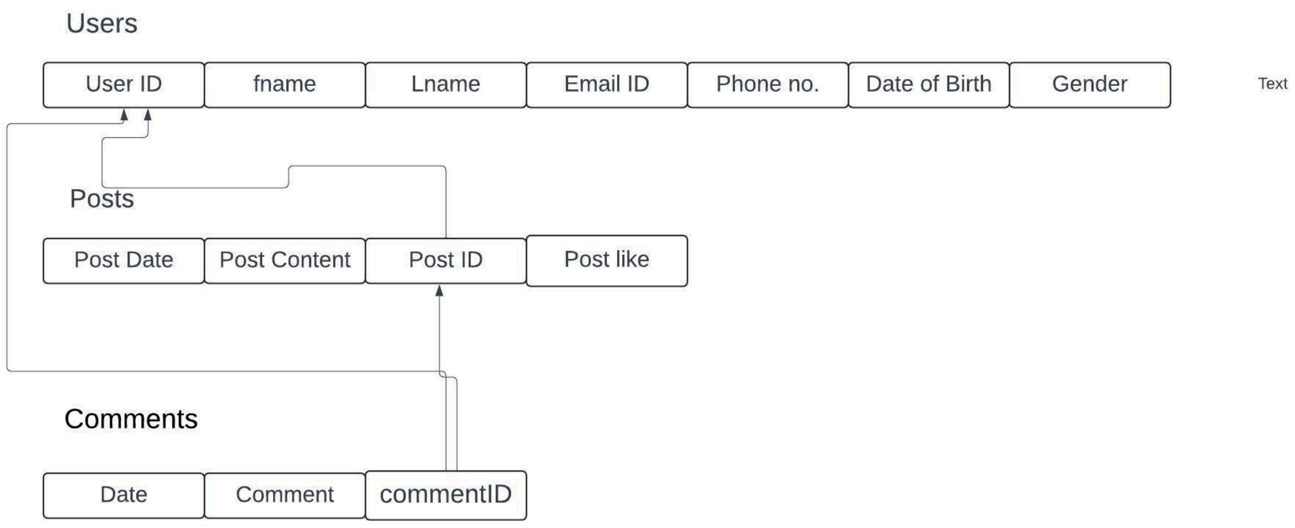
   d. Ensure that foreign keys have appropriate constraints, such as ON DELETE CASCADE or ON UPDATE CASCADE, to maintain data integrity.

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:



Conclusion:

1. **write definition of relational schema and notations.**

a) A relational schema is a formal blueprint or description of the structure of a relational database. It defines the logical organization and layout of the database, including tables, attributes, relationships, keys, and constraints. Here's a breakdown of the components typically included in a relational schema:

b) Tables: Represented by rectangles, each table in a relational schema corresponds to an entity in the database. Tables consist of rows and columns, where each row represents a record or tuple, and each column represents an attribute or field.

c) Attributes: Represented by ovals or ellipses within the table rectangles, attributes define the properties or characteristics of the entities. Each attribute has a name and a data type specifying the kind of data it can hold (e.g., integer, varchar, date).

d) Primary Key: Indicated by underlining one or more attributes within a table, the primary key uniquely identifies each record in the table. It ensures that no two records in the table have the same values for the primary key attributes.

e) Foreign Key: Represented by a dashed underline or a separate line connecting attributes in different tables, foreign keys establish relationships between tables. A foreign key in one table refers to the primary key in another table, defining a constraint that enforces referential integrity.

f) Relationships: Represented by lines connecting tables, relationships describe how tables are related to each other. Common types of relationships include one-to-one, one-to-many, and many-to-many.

g) Cardinality: Often indicated near the ends of the relationship lines, cardinality specifies the number of instances of one entity that can be associated with instances of another entity. Common notations include "1" for one, "M" for many, and "0" for optional.

h) Constraints: Additional constraints, such as unique constraints, check constraints, and default values, may be specified within the schema to enforce data integrity rules and business logic.

2. **write various schema-based constraints.**

Schema-based constraints are rules and conditions that are applied to the structure of a database schema to enforce data integrity and maintain consistency. These constraints ensure that the data stored in the database adheres to certain conditions and requirements. Here are various types of schema-based constraints commonly used in relational databases:

a) Primary Key Constraint: This constraint ensures that each record in a table has a unique identifier, which cannot be null. It is defined on one or more attributes of a table, and it uniquely identifies each record within that table.

b) Foreign Key Constraint: A foreign key constraint establishes a relationship between two tables. It ensures referential integrity by enforcing that the values in a column (or set of columns) in one table must match the values in another table's primary key.
c) Unique Constraint: This constraint ensures that the values in a column (or set of columns) are unique across the table. Unlike primary keys, unique constraints allow null values but enforce uniqueness for non-null values.
d) Check Constraint: A check constraint defines a condition that all data in a column must satisfy. It restricts the range of values that can be inserted into a column, ensuring that only valid data is stored.
e) Default Constraint: This constraint specifies a default value for a column when no explicit value is provided during insertion. It ensures that the column always has a valid value, even if one is not explicitly provided.

Not Null Constraint: The not null constraint ensures that a column cannot contain null values. It forces the column to always have a value, preventing the insertion of nulls.
Identity Constraint: An identity constraint (auto-increment in some databases) is used to automatically generate unique values for a column. It is commonly used for primary key columns to ensure that each new row is assigned a unique identifier automatically.