Experiment No.2
Mapping ER/EER to Relational schema model.
Date of Performance:
Date of Submission:

Experiment No.2

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.

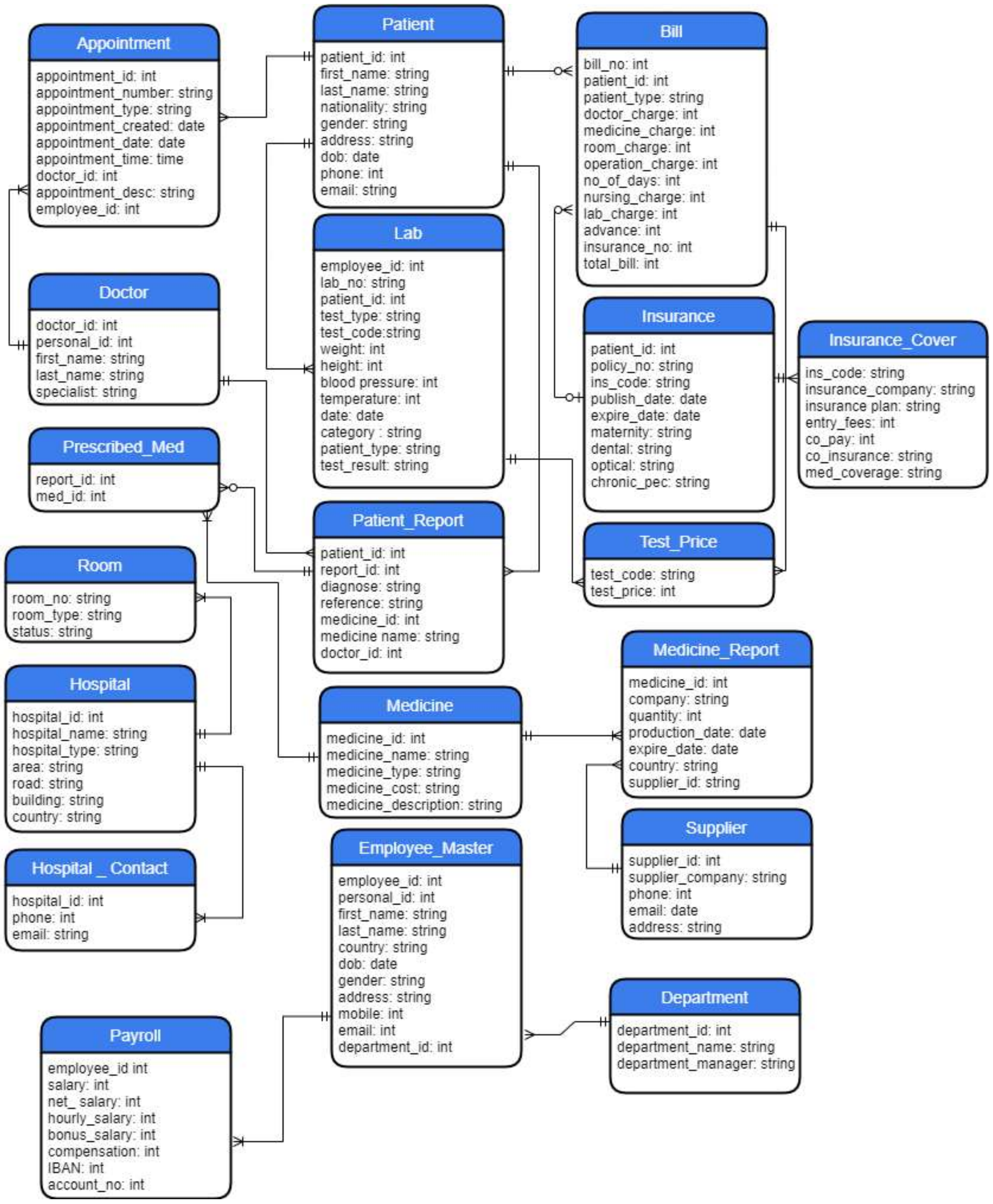
3. Attributes to Columns:
   a. Each attribute of an entity becomes a column in the corresponding table.
   b. Choose appropriate data types for each attribute based on its domain and constraints.
   c. Ensure that attributes participating in relationships are represented as foreign keys when needed.

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:

a. Write definition of relational schema and notations.

Answer: A relational schema is a logical blueprint that represents the structure of a relational database. It defines the organization of data into tables (relations) and the relationships among those tables. A relational schema typically includes the following components:

Table Names: These are the names of the tables in the database.

Attributes/Columns: Attributes represent the properties or characteristics of the entities being modeled. Each attribute is associated with a data type that defines the kind of data it can store (e.g., integer, string, date).

Primary Keys: Primary keys are unique identifiers for each record in a table. They ensure that each row in a table is uniquely identifiable.

Foreign Keys: Foreign keys establish relationships between tables. They are attributes in one table that refer to the primary key in another table.

Constraints: Constraints define rules that restrict the values that can be stored in certain columns. Common constraints include uniqueness constraints (ensuring no duplicate values in a column) and referential integrity constraints (ensuring that foreign keys reference existing primary keys).

Indexes: Indexes are optional structures that improve the speed of data retrieval operations by providing quick access to rows based on the values of certain columns.

Notations commonly used to represent a relational schema include:

Entity-Relationship Diagram (ERD): ERDs use graphical representations to depict the entities, attributes, relationships, and constraints in a database schema. Entities are represented as rectangles, attributes as ovals, relationships as lines connecting entities, and constraints as additional symbols or annotations.

Relational Schema Diagram: This is a simplified version of an ERD that focuses primarily on tables (relations) and their attributes. Tables are represented as rectangles, with their attributes listed inside. Lines connecting tables indicate relationships, and additional symbols may be used to represent keys and constraints.

Relational Schema Notation: This notation uses a textual format to represent the structure of a relational schema. Tables are represented by their names, followed by a list of their attributes and their data types. Keys and constraints are typically specified using additional annotations or comments within the schema definition.

b. Write various schema-based constraints.
Answer: Schema-based constraints are rules that are defined at the schema level to enforce data integrity and maintain consistency in a relational database. Here are various types of schema-based constraints commonly used in relational databases:

Primary Key Constraint: Ensures that each record in a table is uniquely identifiable by specifying one or more columns as the primary key. It prevents duplicate or null values in the primary key column(s).

Example:
sql
Copy code
```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50),
    ...
);
```

Unique Constraint: Ensures that the values in one or more columns are unique across all rows in a table, except for null values.

Example:
sql
Copy code
```
CREATE TABLE Students (
    StudentID INT,
    Email VARCHAR(100) UNIQUE,
    ...
);
```

Foreign Key Constraint: Establishes a relationship between two tables by enforcing referential integrity. It ensures that values in a column (foreign key) of one table match values in another table (primary key or unique key).

Example:
sql
Copy code
```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    ...
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

Check Constraint: Specifies a condition that must be satisfied for each row in a table. It restricts the values that can be inserted or updated in specific columns based on a logical expression.

Example:
sql
Copy code
```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Quantity INT,
    ...
    CHECK (Quantity >= 0)
);
```

Not Null Constraint: Ensures that a column does not contain null values. It requires every row to have a value for the specified column.

Example:
sql
Copy code

```
CREATE TABLE Employees (
    EmployeeID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    ...
);
```

Default Constraint: Specifies a default value for a column if no value is provided during an insert operation.

Example:
sql
Copy code

```
CREATE TABLE Tasks (
    TaskID INT PRIMARY KEY,
    Status VARCHAR(20) DEFAULT 'Pending',
    ...
);
```

These schema-based constraints play a crucial role in maintaining data integrity and enforcing business rules within a relational database.