Experiment 2

Name: Sunil Bhagat
Roll no:20
Batch:C12

ER/EER Mapping rules :

1. ER to Relational Mapping

Step 1:

- Figure out all the regular/strong entity from the diagram and then create a corresponding relation(table) that includes all the simple attributes.
- Choose one of the attributes as a primary key. If composite, the simple attributes together form the primary key.

Step 2:

- Figure out the weak entity types from the diagram and create a corresponding relation(table) that includes all its simple attributes.
- Add as foreign key all of the primary key attributes in the entity corresponding to the owner entity.
- The primary key is a combination of all the primary key attributes from the owner and the primary key of the weak entity.

Step 3:

- Now we need to figure out the entities from ER diagram for which there exists a 1-to-1 relationship.
- The entities for which there exists a 1-to-1 relationship, choose one relation(table) as S, the other as T.
  Better if S has total participation (reduces the number of NULL values).
- Then we need to add to S all the simple attributes of the relationship if there exists any.
- After that, we add as a foreign key in S the primary key attributes of T.

Step 4:

- Now we need to figure out the entities from ER diagram for which there exists a 1-to-N relationship.
- The entities for which there exists a 1-to-N relationship, choose a relation as S as the type at N-side of relationship and other as T.
- Then we add as a foreign key to S all of the primary key attributes of T.

Step 5:

- Now we need to figure out the entities from ER diagram for which there exists an M-to-N relationship.
- Create a new relation(table) S.
- The primary keys of relations(tables) between which M-to-N relationship exists, are added to the new relation S created, that acts as a foreign key.
- Then we, add any simple attributes of the M-to-N relationship to S.

Step 6:

- Now identify the relations(tables) that contain multi-valued attributes.
- Then we need to create a new relation S
- In the new relation S we add as foreign keys the primary keys of the corresponding relation.
- Then we add the multi-valued attribute to S; the combination of all attributes in S forms the primary key.

Step 7:

Mapping of N-ary Relationship Types :

For each n-ary relationship type R

- Create a new relation S to represent R
- Include primary keys of participating entity types as foreign keys
- Include any simple attributes as attributes

2. EER to Relational Mapping

Step 8:

Generalization and Specialization occur when a superclass (general entity) has subclasses (specific entities).

- If we have a generalized superclass **C** (with attributes $\mathbf{k, a_1, a_2, ..., a_n}$, where **k** is the primary key) and **m** specialized subclasses ($\mathbf{S_1, S_2, ..., S_m}$), we can map them into relational schemas using four different methods (8A – 8D). These options define how data is structured in tables while preserving relationships between the superclass and subclasses.
- These mapping techniques help in efficiently organizing and storing data while ensuring consistency in a relational database.

1. Option 8A

Mapping Specialization Using Multiple Relations

- In this approach, we create separate tables for both the superclass and each subclass in the database.
- Superclass Table (C):
- A table L is created for the superclass C with attributes $\{k, a_1, ..., a_n\}$, where k is the primary key.
- Subclass Tables ($S_1, S_2, ..., S_m$):
- For each subclass S_i (where $1 \leq i \leq m$), we create a separate table L_i.
- Each table L_i contains:
- Primary key (k) – inherited from the superclass.
- Attributes specific to the subclass S_i.

2. Option 8B
Mapping Specialization Using Subclass Relations Only

- In this approach, we do not create a separate table for the superclass. Instead, we create tables only for the subclasses, and each subclass table includes the attributes of the superclass.
- Subclass Tables ($S_1, S_2, ..., S_m$):
- A table L_i is created for each subclass S_i (where $1 \leq i \leq m$).
- Each table contains:
- Attributes of the subclass S_i.
- All attributes of the superclass $\{k, a_1, ..., a_n\}$.
- Primary key (k), inherited from the superclass.

This method is useful when specialization ensures that every entity fits into a subclass, avoiding unnecessary superclass tables.

3. Option 8C
Mapping Specialization Using a Single Relation with a Type Attribute

- In this approach, we store all entities (superclass and subclasses) in a single table, adding an extra attribute to indicate the type of each entity.

Single Table (L):

- A single relation L is created with attributes:
- Superclass attributes: $\{k, a_1, ..., a_n\}$
- Subclass-specific attributes: $\{\text{attributes of } S_1\} \cup \{\text{attributes of } S_2\} \cup ... \cup \{\text{attributes of } S_m\}$
- A type attribute (t): Identifies which subclass an entity belongs to.
- Primary key (k)

- When to Use This Approach?
- Only works for disjoint specialization (each entity belongs to only one subclass).
- Can lead to many NULL values if subclasses have many unique attributes, as some entities may not need certain subclass attributes.

This method is useful when subclass attributes are minimal, and storing everything in one table improves query performance by avoiding joins. However, if there are many subclass-specific attributes, a large number of NULL values may make the table inefficient.

4. **Option 8D**

Mapping Specialization Using a Single Relation with Multiple Type Attributes

In this approach, we store all entities (superclass and subclasses) in a single table, but instead of a single type attribute, we use multiple Boolean attributes to indicate subclass membership.

1. **Single Table (L):**

A single relation L is created with attributes:

- Superclass attributes: $\{k, a_1, ..., a_n\}$
- Subclass-specific attributes: $\{\text{attributes of } S_1\} \cup \{\text{attributes of } S_2\} \cup ... \cup \{\text{attributes of } S_m\}$
- Boolean type attributes ($t_1, t_2, ..., t_m$): Each t_i ($1 \le i \le m$) indicates whether an entity belongs **to subclass S_i (1 for yes, 0 for no).**