DBMS - Experiment 2

Name: Kartik Jolapara SapID: 60004200107

Name: Meet Patel SapID: 60004200104

Name: Ayush Jain SapID: 60004200132

Div/Batch: B / B1

AIM:

Mapping ER/EER to Relational schema model.

Theory:

After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages like Oracle SQL, MySQL etc. So, we will see what Relational Model is.

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).

Step 1: Mapping of Regular Entity Types.

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- Choose one of the key attributes of E as the primary key for R. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

Step 2: Mapping of Weak Entity Types

- For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or

simple components of composite attributes) of W as attributes of R.

- In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

Step 3: Mapping of Binary 1:1 Relation Types

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R. There are three possible approaches:
    - Foreign Key approach: Choose one of the relations-S, say- and include a foreign key in S the primary key of T. It is better to choose an entity type with *total participation* in R in the role of S.
    - Merged relation option: An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when *both participations are total.*
    - Cross-reference or relationship relation option: The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

Step 4: Mapping of Binary 1:N Relationship Types.

- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- Include any simple attributes of the 1:N relation type as attributes of S.

Step 5: Mapping of Binary M:N Relationship Types.

- For each regular binary M:N relationship type R, *create a new relation* S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key* of S.
- Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

Step 6: Mapping of Multivalued attributes.

- For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

Step 7: Mapping Specialization or Generalization.

Convert each specialization with m subclasses $\{S_1, S_2,\ldots,S_m\}$ and generalized superclass C, where the attributes of C are $\{k,a_1,\ldots a_n\}$ and k is the (primary) key, into relational schemas using one of the four following options:

- **Option 8A: Multiple relations-Superclass and subclasses.**
  Create a relation L for C with attributes Attrs(L) = $\{k,a_1,\ldots a_n\}$ and PK(L) = k. Create a relation L_i for each subclass S_i, $1 < i < m$, with the attributesAttrs(L_i) = {k} U {attributes of S_i} and PK(L_i)=k. This option works **for any specialization** (total or partial, disjoint of over-lapping).

- **Option 8B: Multiple relations-Subclass relations only**
  Create a relation L_i for each subclass S_i, $1 < i < m$, with the attributes Attr(L_i) = {attributes of S_i} U $\{k,a_1\ldots,a_n\}$ and PK(L_i) = k. This option only works for a specialization whose subclasses are

total (every entity in the superclass must belong to (at least) one of the subclasses).

- **Option 8C: Single relation with one type attribute.**
  Create a single relation L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k. The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs(**Disjoint**)

- **Option 8D: Single relation with multiple type attributes.**
  Create a single relation schema L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t_1, t_2,…,t_m} and PK(L) = k. Each t_i, 1 < I < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i.(**Overlap)**

Relation Schema Diagram:

Subscription:

sub_id	startDate	expiryDate

User:

user_id	userName	password	mobileNo	email	sub_id(FK)

Plays Album:

user_id(FK)	album_id(FK)

Plays Podcast:

user_id(FK)	podcast_id(FK)

Podcast:

podcast_id	podcast_title	podcast_desc	podcast_type	release_date	duration	artist_id(FK)

Album:

album_id	album_title	release_date	duration

Song:

song_id	song_title	album_id(FK)	genre(FK)	artist_id(FK)

Genre:

genre_name	genre_description

Artist:

artist_id	artist_name	rating

Playlist:

playlist_id	playlist_title	user_id(FK)

Playlist contains:

playlist_id(FK)	song_id(FK)

Conclusion:

By applying conversion process, we converted **Entity Relationship/EER Modeling** to relational schema.