Normalization Status of Social Media Database Model

# Functional Dependencies

Here are the Functional Dependencies identified in each table:

**1. users:**

User\_ID ->Email\_ID, Phone\_No, Pass\_word, First\_name, Last\_name, City, PinCode, DOB, Gender

**2. posts:**

Post\_ID ->Posted\_User\_ID, Post\_Date, Post\_Content

**3. post\_comments:**

Comment\_ID -> textPost\_ID, Commented\_Date, Comment\_Content, Commented\_User\_ID

**4. post\_likes:**

(Post\_ID, Liked\_User\_ID) is the primary key; no other functional dependencies exist.

**5. post\_shares:**

(Post\_ID, Shared\_User\_ID) is the primary key; no other functional dependencies exist.

**6. pages:**

Page\_ID -> textPage\_Name, Page\_Content

**7. page\_likes:**

(Page\_ID, Page\_User\_ID) is the primary key; no other functional dependencies exist.

**8. friends:**

(User\_ID, Friend\_ID) is the primary key; no other functional dependencies exist.

**9. comments\_like:**

(Comment\_ID, Comment\_liked\_User\_ID) is the primary key; no other functional dependencies exist.

## Relationships

Below are the relationships between the tables, primarily defined through foreign keys:

1. post\_comments:

- Relationships:

- Commented\_User\_ID *references* User\_ID in users.

- Post\_ID *references* Post\_ID in posts.

2. post\_likes:

- Relationships:

- Liked\_User\_ID *references* User\_ID in users.

- Post\_ID *references* Post\_ID in posts.

3. post\_shares:

- Relationships:

- Shared\_User\_ID *references* User\_ID in users.

- Post\_ID *references* Post\_ID in posts.

4. page\_likes:

- Relationships:

- Page\_User\_ID *references* User\_ID in users.

- Page\_ID *references* Page\_ID in pages.

5. friends:

- Relationships:

- User\_ID *references* User\_ID in users.

- Friend\_ID *references* User\_ID in users.

6. comments\_like:

- Relationships:

- Comment\_ID *references* Comment\_ID in post\_comments.

- Comment\_liked\_User\_ID *references* User\_ID in users.

## Universal Relation (U)

The universal relation (U) can be defined as a set that contains all the attributes from all tables in the database. It can be represented as :

U =

(User\_ID, Email\_ID, Phone\_No, Pass\_word, First\_name, Last\_name, City, PinCode, DOB, Gender,

textPost\_ID, Posted\_User\_ID, Post\_Date, Post\_Content,

textComment\_ID, Commented\_Date, Comment\_Content, Commented\_User\_ID,

textLiked\_User\_ID,

textShared\_User\_ID,

textPage\_ID, Page\_Name, Page\_Content,

textPage\_User\_ID,

textFriend\_ID, textComment\_liked\_User\_ID)

### Check First Normal Form (1NF)

To be in 1NF, each table must meet the following criteria:

- There are no repeating groups or arrays.

- Each column contains atomic values.

- Each column must have a unique name.

- The order in which data is stored does not matter.

**Status for 1NF : All tables satisfy the 1NF requirements.**

Check Second Normal Form (2NF)

To be in 2NF, the table must:

- Be in 1NF.

- Have all non-key attributes fully functionally dependent on the primary key.

Since most tables have composite primary keys, we must check that each non-key column is fully dependent on the entirety of the primary key.

Example: post\_likes and post\_shares depend on both Post\_ID and Liked\_User\_ID/Shared\_User\_ID.

**Status for 2NF : All tables satisfy the 2NF requirements.**

Check Third Normal Form (3NF)

To be in 3NF, the table must:

- Be in 2NF.

- Have no transitive dependencies.

Example: users table contains City and PinCode. These should be separated into a locations table.

**Status for 3NF : All tables satisfy the 3NF requirements.**

Check Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form (BCNF) is a stronger version of the Third Normal Form (3NF). A table is in BCNF if:

1.It is in 3NF.

2.For every one of its non-trivial functional dependencies (X → Y), X is a superkey.

In simpler terms, a table is in BCNF if, for every functional dependency, the left-hand side is a superkey, meaning it can uniquely identify every row in the table.

**Status for BCNF : All tables satisfy the BCNF requirements.**

Final Normalized Schema

1. Users (user\_id, email\_id, phone\_no, password, first\_name, last\_name, city, pincode, dob, gender)

2. Posts (post\_id, posted\_user\_id, post\_date, post\_content)

3. Post\_Comments (comment\_id, post\_id, commented\_date, comment\_content, commented\_user\_id)

4. Post\_Likes (post\_id, liked\_user\_id)

5. Post\_Shares (post\_id, shared\_user\_id)

6. Pages (page\_id, page\_name, page\_content)

7. Page\_Likes (page\_id, page\_user\_id)

8. Friends (user\_id, friend\_id)

9. Comments\_Like (comment\_id, comment\_liked\_user\_id)

Advantages of Normalization

Eliminates Data Redundancy: No duplicate data across tables.

Improves Data Integrity: Updates occur in one place, avoiding inconsistencies.

Enhances Query Performance: Smaller tables improve query execution.

Prevents Anomalies: Avoids insertion, update, and deletion anomalies.

Conclusion

All tables are in Boyce-Codd Normal Form (BCNF). Each table complies with the requirements of BCNF, meaning that every functional dependency within the tables has a determinant that is a superkey.

The design of the database achieves a high degree of normalization, promoting data integrity and minimization of redundancy