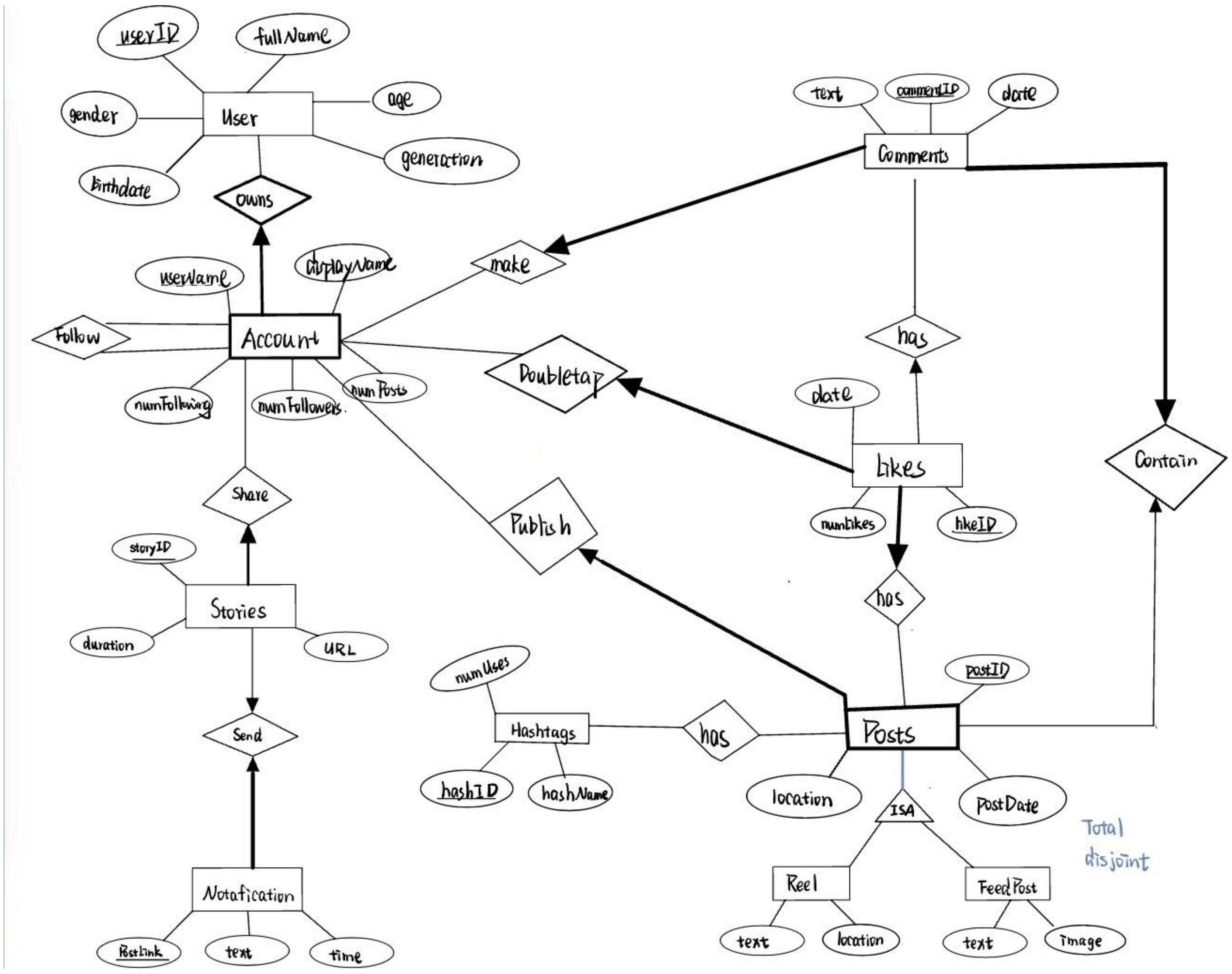
Our project is a database that will model social media users, accounts, the posts accounts can made and interactions made between these accounts. Our database models each individual user as an entity and linkthem with multiple accounts, we wish to store their profile information such as usernames, follower counts, following counts, as user-specific attributes.

ER diagram:



Schemas

NotificationsSend(<u>PostLink: </u>VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)

User(<u>userID:</u> INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date, generation: VARCHAR[]);

StoriesShare(storyID: VARCHAR[ ], duration: date NOT NULL, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER NOT NULL)

Reel(text: CHAR[ ], length: CHAR[ ], **postID:** INTEGER, postID: INTEGER, location: VARCHAR[ ], postDate: date)

FeedPost(text: CHAR[ ], image, **postID:** INTEGER, location: VARCHAR[ ], postDate: date)

Hashtags(hashID: INTEGER, hashName: VARCHAR[ ])

HashtagHasPost(**hashID:** CHAR[ ]**, postID:** INTEGER)

LikesDoubleTapHas(likeID: CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**, userName:** VARCHAR[ ]**, userID:**INTEGER**, commentID:** CHAR[ ])

PublishPosts(PostID, location, post_date, **userName, userID**)

MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date, **userName:** VARCHAR[ ] NOT NULL**, userID**:INTEGER NOT NULL) // we need to include userName, userID since both of them are our primary key

Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)

AccountOwns(userName: VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **userID:** INTEGER)

Function Dependencies

1. NotificationsSend(PostLink: VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)
   FD:
   PostLink -> text, time, storyID
   (time, storyID) -> PostLink

2. User(userID: INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date, generation: CHAR[]);
   FD:
   userID-> gender, fullName, age, birthdate
   birthdate-> generation

3. StoriesShare(storyID: CHAR[ ], duration: date, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER)
   FD:
   storyID -> duration, URL, userName, userID
   URL -> duration, storyID, userName, userID

(userID, duration) -> (storyID, URL, name) //// not in BNF

4. Reel(text: CHAR[ ], length: CHAR[ ], postID: INTEGER, location: VARCHAR[ ], postDate: date)
FD:
postID -> length, text, location, postDate
(length, text, location, postDate) => hashID

5. FeedPost(text: CHAR[ ], numImage: INTEGER, postID: INTEGER, location: VARCHAR[ ], postDate: date)
FD:
postID->text, numImage, location, postDate
(text, numImage, location, postDate) => hashID

6. Hashtags(hashID: INTEGER, hashName: VARCHAR[ ], numUses: INTEGER)
FD:
hashID -> hashName, numUses
hashName -> numUses

7. HashtagHasPost(**hashID:** INTEGER**, postID:** INTEGER)
FD:
hashID -> postID
postID -> length, text, location, postDate

8. LikesDoubletapHas(likeID: CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**, userName:** VARCHAR[ ]**, userID:**INTEGER**, commentID:** CHAR[ ])
FD:
likeID -> numLikes, date, postID, userName, userID, commentID
(postID, userID) -> likeID
userID->userName
postID-> userID, userName, numLikes, commentID
date, postID -> numLikes

9. MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date, **userName:** VARCHAR[ ] NOT NULL**, userID:** INTEGER NOT NULL)
FD:
commentID -> text, date, userName, userID
userID->userName
userID, text, date-> commentID

10. Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)

FD:
commentID-> postId
postID -> commentID

11. AccountOwns(<u>userName: </u>VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **<u>userID: </u>**CHAR[ ])
FD:
userName, userID -> displayName, numFollowing, numFollowers, numPosts
displayName, numFollowing, numFollowers, numPosts -> username, userID

12. PublishPosts(<u>PostID, </u>location, postDate, **userName, userID**)
FD:
PostID -> location, postDate, userName, userID
userID->userName

Normalization

1. Both PostLink and (time, storyID) are superkeys, so no decomposition is required
   NotificationsSend(<u>PostLink: </u>VARCHAR[ ], text: VARCHAR[ ], time: time, **storyID:** NOT NULL)

2. Birthdate is not superkey, decompose the table into two tables. One contains birthdate and generation, and the other one contains all attributes except generation.
BirthdateGen(<u>birthdate</u>: DATE, generation: CHAR[])
User(<u>userID:</u> INTEGER, gender: VARCHAR[ ], fullName: VARCHAR[ ], age: INTEGER, birthdate: Date)

3. Already in BCNF.
StoriesShare<u>(storyID</u>: VARCHAR[ ], duration: date, URL: VARCHAR[ ] UNIQUE, **userName:** VARCHAR[ ]**, userID:** INTEGER)

4. Reel
Already in BCNF
Reel(text: CHAR[ ], length: CHAR[ ], <u>postID: </u>INTEGER, location: VARCHAR[ ], postDate: date)

5. FeedPost
Already in BCNF
FeedPost(text: CHAR[ ], numImage: INTEGER, <u>postID: </u>INTEGER, location: VARCHAR[ ], postDate: date)

6. Hashtags

hashname is not superkey, so it is not in BCNF, decomposes into two tables. One contains hashName and numUses, and the other one contains hashID and hashName.
Hashtags(hashID: INTEGER, hashName: VARCHAR[ ])
HashName(hashName: VARCHAR[ ], numUses: INTEGER)

7. HashtagHasPost
Already in BCNF.
HashtagHasPost(**hashID:** INTEGER**, postID:** INTEGER])

8. LikesDoubletapHas
UserID is not superkey, so it is not in BCNF. PostID is not superkey either but it is part of a minimal key, so it is in 3NF. We only need to decompose the FD userID->userName.
UserIdentity(**userID:**INTEGER,userName**:** VARCHAR[ ])
LikesDoubletapHas(likeID: CHAR[ ], numLikes: INTEGER, date: date, **postID:** INTEGER NOT NULL**,userID:**INTEGER**, commentID:** CHAR[ ])

9. MakeComments
UserID is not superkey, but it is being decomposed in the above relationship, still decompose with this FD but get rid of the duplicated (UserID,UserName) table.
MakeComments(text: CHAR[ ], commentID: CHAR[ ], date: date**, userID:** INTEGER NOT NULL)

10. Contain
Already in BCNF.
Contain(**commentID:** CHAR[ ]**, postID:** INTEGER)

11. AccountOwns
Already in BCNF.
AccountOwns(userName: VARCHAR[ ], displayName: VARCHAR[ ], numFollowing: INTEGER, numFollowers: INTEGER, numPosts: INTEGER, **userID:** INTEGER)

12. PublishPosts
UserID is not superkey, but it is being decomposed in the above relationship, still decompose with this FD but get rid of the duplicated (UserID,UserName) table.
PublishPosts(PostID: INTEGER, location: VARCHAR[ ], postDate: DATE, **userID**: INTEGER)

SQL DDL statements

1. CREATE TABLE NotificationsSend (
PostLink VARCHAR[255],
text VARCHAR[500],
time TIME,

```
        storyID VARCHAR[255] NOT NULL,
        PRIMARY KEY (PostLink),
        FOREIGN KEY (storyID) REFERENCES StoriesShare,
        ON UPDATE CASCADE
        ON DELETE CASCADE
        )
2.      CREATE TABLE User(
        userID INTEGER,
        gender VARCHAR[3],
        fullName VARCHAR[30],
        age INTEGER,
        birthdate DATE,
        PRIMARY KEY (userID)
        )

        CREATE TABLE BirthdateGen(
        birthdate DATE,
        generation CHAR[4],
        PRIMARY KEY (birthdate)
        )


3.      CREATE TABLE StoriesShare(
        storyID VARCHAR[255],
        duration DATE,
        URL VARCHAR[255] UNIQUE,
        userName VARCHAR[30],
        userID INTEGER,
        PRIMARY KEY (storyID),
        FOREIGN KEY (userName) REFERENCES AccountOwns,
        ON UPDATE CASCADE
        ON DELETE CASCADE
        FOREIGN KEY (userID) REFERENCES UserIdentity,
        ON UPDATE CASCADE
        ON DELETE CASCADE
        )


4.      CREATE TABLE Reel(
        text CHAR[255],
        length CHAR[255],
        postID INTEGER,
        location VARCHAR[255],
        postDate DATE,
        PRIMARY KEY (postID)
        )


5.      CREATE TABLE FeedPost(
        text CHAR[255],
        numImage INTEGER,
        postID INTEGER,
        location VARCHAR[255],
        postDate DATE,
        PRIMARY KEY (postID)
        )
```

6.
```
CREATE TABLE Hashtags(
hashID INTEGER,
hashName VARCHAR[100],
PRIMARY KEY (hashID)
)

CREATE TABLE HashName(
hashName: VARCHAR[100],
numUses: INTEGER,
PRIMARY KEY (hashName)
)
```

7.
```
CREATE TABLE HashtagHasPost(
hashID INTEGER,
postID INTEGER,
PRIMARY KEY (hashID,postID),
FOREIGN KEY (hashID) REFERENCES Hashtags,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (postID) REFERENCES FeedPost,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

8.
```
CREATE TABLE UserIdentity(
userID INTEGER,
userName VARCHAR[100]
PRIMARY KEY (userID),
FOREIGN KEY (userID) REFERENCES User,
ON UPDATE CASCADE
ON DELETE CASCADE

CREATE TABLE LikesDoubletapHas(
likeID CHAR[11],
numLikes INTEGER,
date DATE,
postID INTEGER NOT NULL,
userID INTEGER,
commentID CHAR[11]
PRIMARY KEY (likeID),
FOREIGN KEY (postID) REFERENCES FeedPost,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (commentID) REFERENCES MakeComments,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

9.

```
CREATE TABLE MakeComments(
text CHAR[100],
commentID CHAR[11],
date DATE,
userID INTEGER NOT NULL,
PRIMARY KEY (commentID),
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

10.

```
CREATE TABLE Contain(
commentID CHAR[11],
postID INTEGER,
PRIMARY KEY (commentID,postID),
FOREIGN KEY (commentID) REFERENCES MakeComments,
ON UPDATE CASCADE
ON DELETE CASCADE
FOREIGN KEY (postID) REFERENCES FeedPost,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

11.

```
 CREATE TABLE AccountOwns(
userName VARCHAR[100],
displayName VARCHAR[100],
numFollowing INTEGER,
numFollowers INTEGER,
numPosts INTEGER,
userID INTEGER,
PRIMARY KEY (userName,userID),
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```

12.

```
CREATE TABLE PublishPosts(
PostID INTEGER,
location VARCHAR[100],
postDate DATE,
userID INTEGER,
PRIMARY KEY (PostID),
FOREIGN KEY (userID) REFERENCES UserIdentity,
ON UPDATE CASCADE
ON DELETE CASCADE
)
```