Team 109 - Conceptual and Logical Database Design

Team members: Sally Xue, Pradyumann Singhal, Lily Zhang, Joseph Schanne

1. Assumptions

a. ALBUMS
    i. We assume that an album is only under one artist for sake of simplicity so we have - Artists and Albums will be **one to many** relationship since one artist can have multiple Albums.

2. Description of Relationship

- USERS
    - We have a **Users Entity Set** to store information about Users and to deal with the authentication part with the primary key as User ID that will be used to identify the user.
- SONGS
    - Songs table is connected to Artists and Albums using Artist ID and album ID respectively.
    - Songs to Artists have **many to one** relationship as multiple songs are released by one artist. (We don't consider case where 2 artists release song as mentioned in assumption)
    - Songs to Albums have **many to one** relationship as there can 1 or more than 1 song under 1 artist
- ALBUMS
    - Albums is a primary key with Album ID.
    - Songs and albums have a **many to one** relationship as many songs exist in 1 album
- ARTISTS
    - Artists is a simple table with Artist ID and Name so we use this table to get values.
    - Songs and Artists have **many to one** relationship.
- COMMENTS
    - We have **Comments** as the last entity Set which we are using to store all comments.
    - We have Song ID Column as a way to filter for each song along with ResponseTo field that will store the value of comment ID field so we can code to create a nested comment Loop.

3. Normalize Database (Apply 3NF)

Functional Dependencies:
UserID → Username, Password, Email
SongID → Song Name, ArtistID, AlbumID, Release Date
ArtistID → Artist Name, AlbumID
AlbumID → Album Name, SongID, ArtistsID, Long Description
CommentId -> UserID, SongID, CommentInfo, Rating, Created On, ResponseTo

Left	**Middle**	**Right**	**None**
CommentID	ArtistID	Username	
	AlbumID	Password	
	SongID	Email	
	UserID	SongName	
		ReleaseDate	
		ArtistName	
		AlbumName	
		LongDescription	
		CommentInfo	
		Rating	
		CreatedOn	
		ResponseTo	

Candidate Key = CommentID
Key = UserID, SongID, ArtistID, AlbumID, CommentID

Compute the minimum basis for FD (Making sure RHS of every FD is singleton):
UserID → Username
UserID → Password

UserID → Email
SongID → Song Name,
SongID → ArtistID,
SongID → AlbumID,
SongID → Release Date
ArtistID → Artist Name
ArtistID → AlbumID
AlbumID → Album Name,
AlbumID → SongID
AlbumID → ArtistsID
AlbumID → Long Description
CommentID -> UserID,
CommentID -> SongID
CommentID -> CommentInfo
CommentID -> Rating
CommentID -> Created On
CommentID -> ResponseTo

Removing redundant from the LHS:
We only need to consider cases where the left-hand side has more than one attribute. However, none of our functional dependencies meet this condition, so nothing needs to be done for this step.

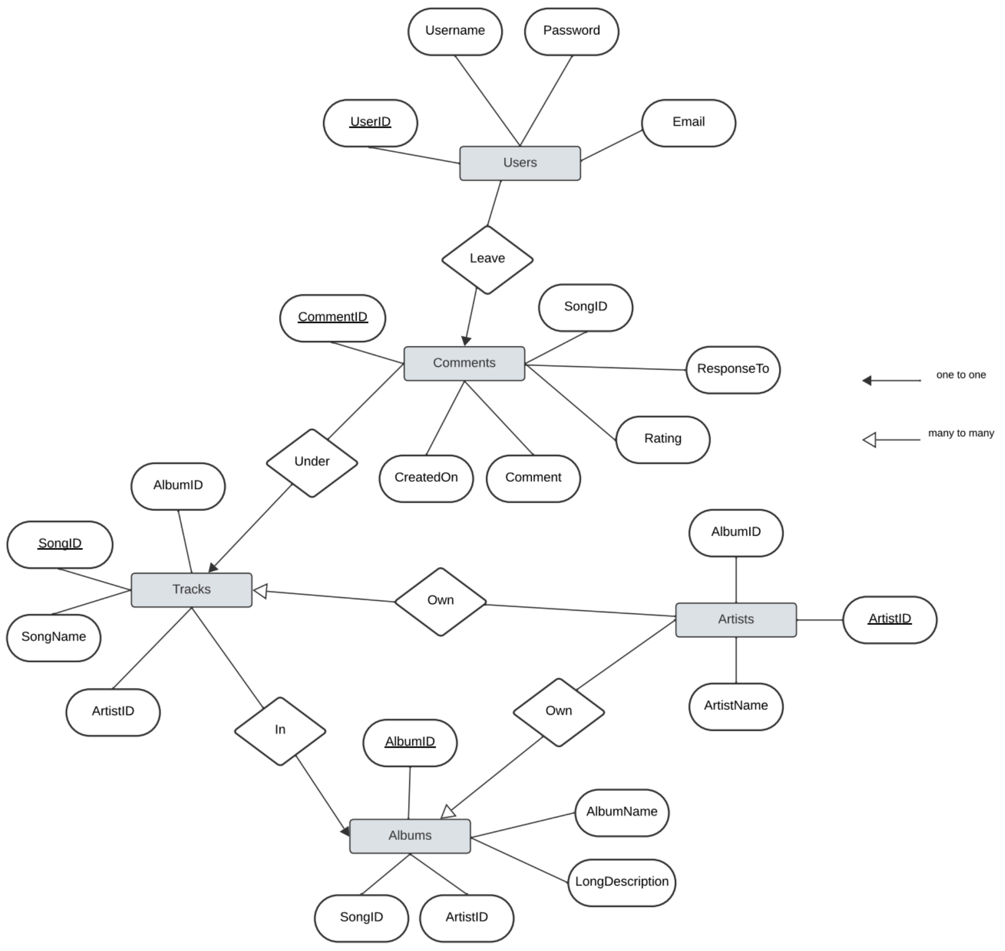
Remove unnecessary FD: (if A->B and B->A, then we remove one of them)

UserID → Username
UserID → Password
UserID → Email
SongID → Song Name,
SongID → ArtistID,
SongID → AlbumID,
SongID → Release Date
ArtistID → Artist Name
AlbumID → ArtistID
AlbumID → Album Name,
AlbumID → Long Description
CommentID -> UserID,
CommentID -> SongID
CommentID -> CommentInfo

CommentID -> Rating
CommentID -> Created On
CommentID -> ResponseTo

Relations:
A (UserID [PK], Username, Password, Email)
B(SongID [PK] , Song Name, ArtistID [FK to ARTISTS.ArtistID], AlbumsID [FK to ALBUMS.AlbumID], Release Date)
C (ArtistID [PK], Artist Name)
D (AlbumID [PK], ArtistID [FK to ARTISTS.ArtistID], Album Name, Long Description)
E(CommentID [PK], UserID [FK to USERS.UserID], SongID [FK to SONGS.SongID], CommentInfo, Rating, Created On, ResponseTo)

4. ER Diagram



5. Relational Schema

USERS(
UserID: VARCHAR(255) [PK],
Username: VARCHAR(255),
Password: VARCHAR(255),
Email: VARCHAR(255)
)

SONGS(
SongID: VARCHAR(255) [PK],
Song Name: VARCHAR(255),
ArtistID: VARCHAR(255) [FK to ARTISTS.ArtistID],
AlbumsID: VARCHAR(255) [FK to ALBUMS.AlbumID],
Release Date: DATE
)

ARTISTS(
ArtistID : VARCHAR(255) [PK]
Artist Name: VARCHAR(255)
)

ALBUMS(
AlbumID : VARCHAR(255) [PK]
ArtistID: VARCHAR (255) [FK to ARTISTS.ArtistID]
Album Name : VARCHAR (255)
Description: VARCHAR (255)
)

COMMENTS(
CommentID : VARCHAR(255) [PK]
UserID: VARCHAR(255) [FK to USERS.UserID]
SongID: VARCHAR(255) [FK to SONGS.SongID]
CommentInfo: VARCHAR(255)
Rating: INTEGER
Created On: DATE
ResponseTo: VARCHAR(255)
)