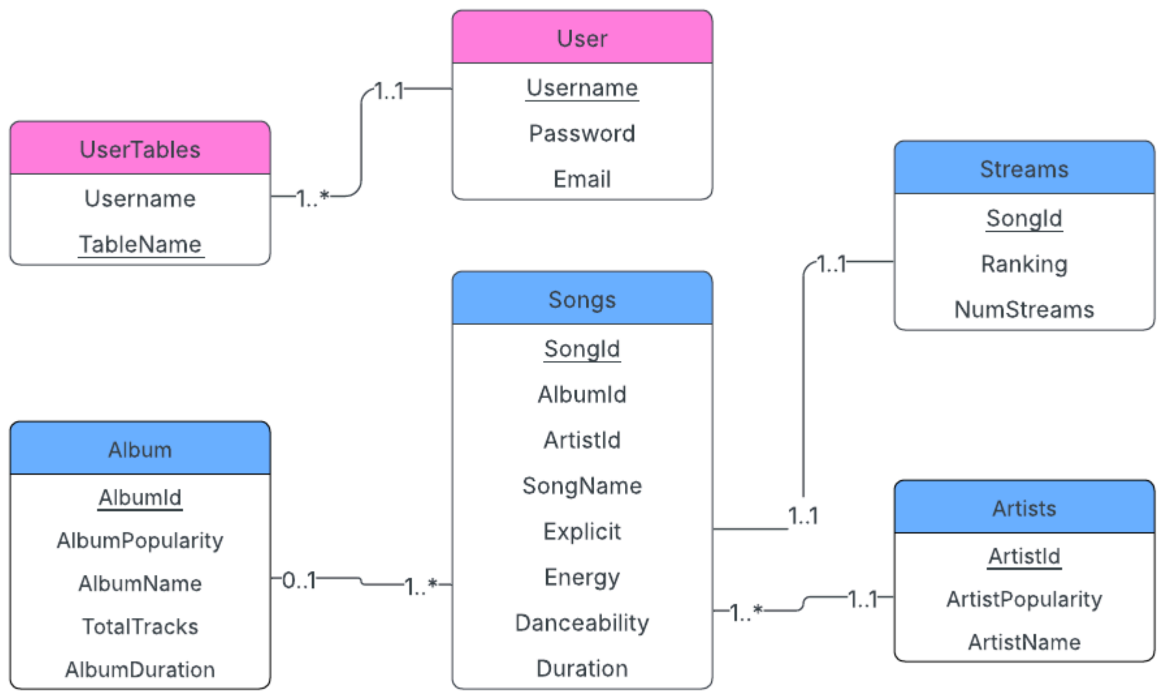
1 - UML Diagram



2 - Assumptions for entities and relationships

Users: Users represent unique accounts in the system. It would not make sense to store this information as an “attribute” of another entity since user-related data is not determined by other entities in the database.

UserTables: UserTables represents all the unique tables that the user may generate. Since each user can generate multiple tables, we want to store this information as a separate entity rather than an attribute of User

Songs: Songs represents unique songs stored in the database and stores attributes such as Energy, Danceability, and Duration. Thus, it makes sense for Songs to be its own entity rather than an attribute to another table.

Album: Album contains its own attributes that are not necessarily dependent on a single song. Therefore, Album must also be its own entity.

Artists: Similar to Album, each artist may have multiple songs and contain attributes that are not necessarily dependent on a particular album song. Hence, it is an entity, not an attribute.

Streams: We separate Streams as a distinct entity since this information is data that is updated frequently. So, it is easier and more organized to represent this data as a separate entity rather than just an attribute of Songs.

User -> UserTables: One-to-Many: Each user can generate multiple tables, however, each generated table can only have one user.

UserTables -> User: Many-to-One: Multiple tables can belong to one user.

Songs -> Album: One-to-One: Each Song belongs to either 0 or 1 album.

Album-> Songs: One-to-Many: Each album can have multiple songs.

Songs -> Streams: One-to-One: One song in Songs has unique statistics on streams data.

Streams -> Songs: One-to-One: Each row in streams uniquely represents the statistics for one song.

Songs -> Artist: One-to-One: Each song only has one artist.

Artist -> Songs: One-to-Many: Each artist can have as many songs as they want.

3 - Normalization

Given the attributes we are considering, these were the functional dependencies we determined:

- Username -> Password, Email
- TableName -> Username
- AlbumId -> AlbumPopularity, AlbumName, TotalTracks, AlbumDuration
- SongId -> Ranking, NumStreams
- SongId -> AlbumId, ArtistId, SongName, Explicit, Energy, Danceability, Duration
- ArtistId -> ArtistPopularity, ArtistName

Since all LHS are singletons, there are no unnecessary attributes in the LHS. Also, since all LHS attributes are unique we know no FDs can be inferred from the rest. Therefore, the schema for the new relation is:

- (Username, Password, Email)
- (TableName, Username)
- (AlbumId, AlbumPopularity, AlbumName, TotalTracks, AlbumDuration)
- (SongId, AlbumId, ArtistId, SongName, Explicit, Energy, Danceability, Duration, Ranking, NumStreams)
- (ArtistId, ArtistPopularity, ArtistName)

But since we made the design decision to separate the streams data into a new entity, we split

(SongId, AlbumId, ArtistId, SongName, Explicit, Energy, Danceability, Duration, Ranking, NumStreams)

into:

(SongId, Ranking, NumStreams) and (SongId, AlbumId, ArtistId, SongName, Explicit, Energy, Danceability, Duration)

Since SongId is a superkey for all relevant song, album, and artist data we do not need another relation whose schema is a key for this data.

(Username, TableName) is a superkey for all relevant user data. but we do not need to add another relation whose schema is a key for this data because we already have UserTables(Username, TableName) which matches the superkey exactly.

We do not require a superkey that identifies both song and user data because there is no meaningful relationship between the song and user data. Therefore, our schema is in 3NF.

<u>4 - Logical Design</u>

- Album(AlbumId:INT [PK], AlbumPopularity:INT, AlbumName:VARCHAR(1000), TotalTracks:INT, AlbumDuration:INT)
- Artists(ArtistId:INT [PK], ArtistPopularity:INT, ArtistName:VARCHAR(1000))
- Songs (SongId:INT [PK], AlbumId:INT [FK], ArtistId:INT [FK], SongId:VARCHAR(1000), Explicit:INT, Energy:FLOAT, Danceability:FLOAT, Duration:INT)
- Streams(SongId:INT [PK][FK], Ranking:INT, NumStreams:INT)
- User(Username:VARCHAR(1000) [PK], Password:VARCHAR(1000), Email:VARCHAR(1000))
- UserTables(TableName:VARCHAR(1000) [PK], Username:VARCHAR(1000) [FK])