Contents

1 Entity Relationship Diagram 2

2 Erd Members 3

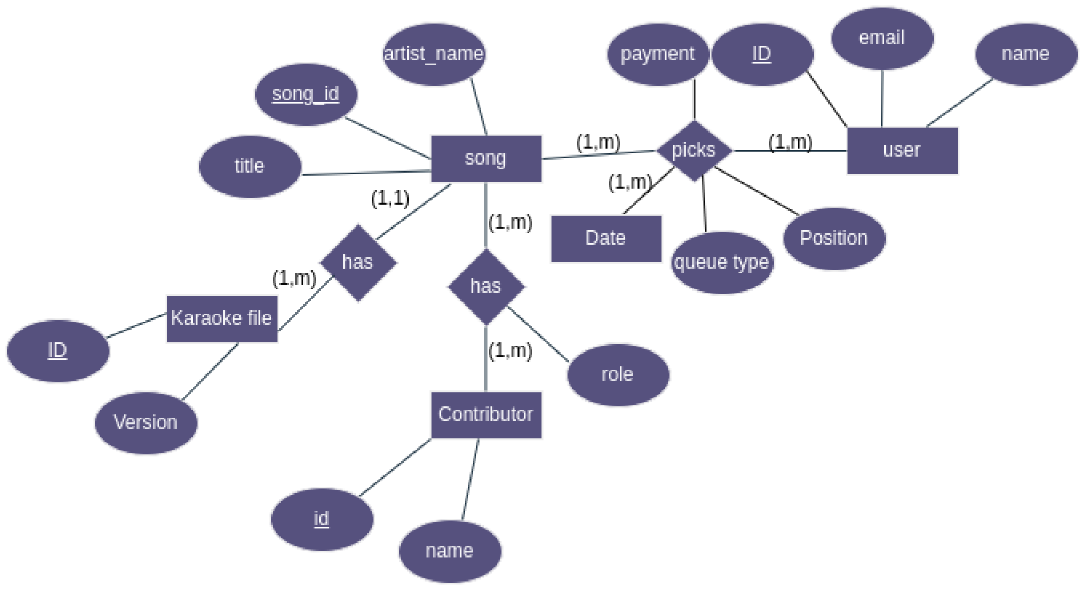
2.1 Entities . . . 3

2.2 Attributes . . . 3

2.3 Entity Relationships . . . 4

3 ERD to 3NF Relations 5

Entity Relationship Diagram



Erd Members

2.1 Entities

- **User**: The user entity is used to store information about the users, such as name, email, and the users ID. The identifier is the users ID.
- **Song**: The song entity is used to store information about the songs available for the user to select. Stores information about the song title, the name of the artist(s), and the songs ID. The identifier for this entity is the songs ID.
- **Contributor**: The contributor entity is used to store information about people who have contributed to the creation of songs. Stores information about the name of the contributor, and the contributors ID. Identifier is contributors ID.
- **Karaoke File**: The file entity is used to store information about the files associated with a specific song. Stores the files ID and the version. Identifier is the files ID.

2.2 Attributes

- **User**:
  1. **Email**: The users email
  2. **Name**: The users name
  3. **ID**: The ID given to the user by the database
- **Song**:
  1. **Artist_name**: The name of the artist or group associated with the song
  2. **Song_id**: The Id given to the song by the database
  3. **Title**: The title of the song
- **Contributor**:
  1. **Id**: The ID given to the contributor by the database
  2. **Name**: The name of the contributor
- **Karaoke File**:
  1. **ID**: The Id given to the file by the database
  2. **Version**: The name describing the version of the song

2.3 Entity Relationships

- **User, song, date**: Connected via a *picks* relationship. A user picks a song at a specific time. When this occurs, payment, position, queue type, and date (time) records are stored. When the user and song are known, there can be many dates. When the user and date are known, there can be many songs. When the song and date are known, there can be many users.
- **Song, contributor**: Connected via a *has* relationship. When this relationship occurs, a role record is stored. A song can have many contributors, a contributor may be involved with many songs.
- **Song, karaoke file**: Connected via a *has* relationship. A song may have many files, but each file is unique to a song.

ERD to 3NF Relations

The four strong non subtype entites are easily converted to relations.

user(user_ID, email, name)
song(song_ID, artist_name, title)
contributor(contrib_ID, name)
karaoke_file(file_ID, version, song_id†).

Remark. The *has* relationship between file and song is one-to-many. Thus, we add a foreign key *song_id* to the song relation in the file relation. When the file is known, there is only one song.

The remaining two relationships get converted to

user_song(user_ID, song_id, date, position, queue_type, payment)
song_contributor(song_id, contrib_id, role).

In the *user_song* relation, the relationship is many-to-many-to many, so the identifier in each relation become the primary key for the relation. The intersection data is added as non-prime attributes.

The *song_contributor* is also many-to-many, so the two identifiers become the primary key, and the intersection data *role* is added as a non-prime attribute.