Users(UserID, Name)

Songs(SongID, Title, BandName)

SongVersions(VersionID, SongID†, FileName, Description)

(SongID -> Songs.SongID)

Contributors(ContributerID, Name)

RequestQueue(RequestID, VersionID†, UserID†, Time, AmountPaid, Played, QueueType)

(VersionID -> SongVersions.VersionID, UserID -> Users.UserID)

SongContributors(SongID†, ContributerID†, Role)

(SongID -> Songs.SongID, ContributerID -> Contributors.ContributerID)

† means foreign

The User entity represents the person or account that makes requests. We split up the ID and name attributes because we using the name as a primary key is not the best choice because repeat names are possible and simply using an ID is not descriptive enough.

The Song entity simply represents any useful information the song has. With the exception of the Contributors, which have been split off due to the special nature, the other attributes consisting of an ID, Title, and BandName give a general overview of the object.

Because each Song can have multiple Files, these Files needed to be stored as entities too. By doing this we can give it attributes which hold the FileID, a FileName to locate the file, and a Description to explain what is unique about this version.

Lastly are the Contributor entities. We chose to create this one due to how we can expect to reuse contributors for multiple songs and even multiple times per song. Simply an ID and Name should be enough information to distinguish between Contributors.

We match up Contributors with the songs the help with via the SongContributors relation. To account for the fact that a Contributor can do multiple things for a song, we added a Role attribute to the primary key. This clarifies what each Contributor has done.

In order to connect the Song with their corresponding files, we added a relation but due to the nature of it, we didn’t need to create a new table. We could simply attach the SongID to the SongVersions entity.

Lastly we have the queue. This queue connects a User with a Song and gives the request an ID for uniqueness and ordering. In addition to this we added an attribute for the QueueType to distinguish what queue the entity is in. With this attribute we can find out whether the request is in the free queue, priority queue, now playing queue or history queue. The AmountPaid and Time attributes are used when the request is in the priority queue. The played attribute is self explanatory.