WolfMedia Streaming Service

CSC 540 - Project Report 1
Team E
Zach Hadgraft (zhadgra), Corey Capooci (cvcapooc), Ethan Purnell (efpurne2), Andrew Shon (ashon)
February 13, 2023

Assumptions:

- Content creators include artists and podcasters. They interact with the database by uploading and removing their content.
- Every song belongs to an album.
- Every album belongs to a single primary artist.
- Every artist contracts with one record label.
    - Every artist is given a unique ID.
- The genre(s) of a song consists of the collection of primary genres for the artist(s) on the song.
- Every record label has a unique name to serve as the key
- Management tracks ratings externally and manually updates the attribute.
- Every podcast has a unique name which can serve as the key.
- Artist name is entered in the first name attribute of Content Creator.
- Additional operations were included beyond those explicitly listed in the project narrative to aid in testing of our database as it is built.
- Counted attributes such as subscribers and monthly listeners can be determined by aggregate values from relationship-based relations.

1. Problem Statement

We will design a WolfMedia Database System for the staff at WolfMedia to maintain artist/podcaster information, artist/podcaster content, royalty payment information, and subscriber/usage information. There are four major tasks which will be performed: 1) information processing, 2) maintaining metadata and records, 3) generating royalties and payment, and 4) creating reports summarizing monthly/yearly/total payment summaries and subscribers/play counts. Listeners of the service will “sign up,” at which time they provide basic user info such as name and email address. The database system will require a unique email address for subscribers, which will aid in tracking unique active listeners/subscribers.

Implementing a database system offers advantages over a file system for this service as it is substantially more scalable and provides more real-time tracking capabilities. Should the WolfMedia service become very popular, it is feasible that many thousands of users could be interacting with the system at once. In this case, it may be impossible for an administrator to accurately track, input, and store the artist or user information when the system is being used, especially considering that a user may only be interacting with the system for a few minutes to listen to one song. Implementing a database system for this service substantially increases efficiency and ensures that there is a mechanism to correctly collect and store the required information to pay out correct royalties to record labels and artists. A mistake made in a file system being manually updated could result in very costly lawsuits if royalty payments are inaccurate.

2. Intended Users

Administrators

Administrators have complete control over the WolfMedia Streaming Service.

Management

Importantly, they collect, analyze, and generate reports on various aspects of the streamed items and associated parties including but not limited to song play count, podcast subscribers, ratings, sponsors, and all payments made.

Podcasters

Podcasters publish content on the platform.

Artists

Artists publish content on the platform.

3. Five Main Entities

1. Song: song title, artist(s), duration, genre(s), album, play count, release date, release country, language, royalty rate, collaborators (guest artist)
2. Artist: name, collaboration(s), duration, genre(s), album, play count, release date, release country, language, royalty rate, collaborators
3. Album: album name, artist(s), song(s), track number(s), release year, edition (special, limited, collector's edition)
4. Podcast: podcast host, podcast name, language, country, episode count, genre(s), rating, sponsor(s), total subscribers
5. Podcast episode: episode title, duration, release date, listening count, special guest(s), advertisement count

4. Situation

Situation: A record label is about to release an album for their most valuable client, singer-songwriter Charlie Brown. Upon receiving the tracks for the album, Brown's representative at the label, Alice, enters basic information about the newly released songs and assigns the songs and Charlie Brown to the new album.

5. Application Program Interfaces

Information Processing:

1. enterSongInfo(songTitle, creatorId, duration, albumName, playCount, releaseDate, releaseCountry, language, royaltyRate, royaltyPaid)
   return confirmation

2. updateSongInfo(songTitle, creatorId, duration, albumName, playCount, releaseDate, releaseCountry, language, royaltyRate)
   return confirmation
   * if NULL value provided for any inputs, those existing values will not be updated

3. deleteSongInfo(songTitle, albumName, creatorId)
   return confirmation
   * delete operations delete the full tuple, same for below operations as well

4. enterArtistInfo(name, status, type, country, primaryGenre)
   return confirmation

5. updateArtistInfo(name, status, type, country, primaryGenre, monthlyListeners,)
   return confirmation
   * if NULL value provided for any inputs, those existing values will not be updated

6. deleteArtistInfo(creatorId)
   return confirmation

7. enterPodcastHostInfo(firstName, lastName, phone, email, city)
   return confirmation

8. updatePodcastHostInfo(firstName, lastName, phone, email, city)
   return confirmation
   * if NULL value provided for any inputs, those existing values will not be updated

9. deletePodcastHostInfo(email)
   return confirmation

10. enterPodcastEpisodeInfo(podcastName, episodeTitle, duration, releaseDate, advertisementCount)
    return confirmation

11. updatePodcastEpisodeInfo(podcastName, episodeTitle, duration, releaseDate, advertisementCount)
    return confirmation
    * if NULL value provided for any inputs, those existing values will not be updated

12. deletePodcastEpisodeInfo(podcastName, episodeTitle)
    return confirmation

13. enterAlbumInfo(albumName, creatorId, releaseYear, edition)
    return confirmation

14. updateAlbumInfo(albumName, creatorId, releaseYear, edition)
    return confirmation
    * if NULL value provided for any inputs, those existing values will not be updated

15. deleteAlbumInfo(albumName, creatorId, edition)
    return confirmation

16. enterUserInfo(firstName, lastName, phone, email, registrationDate, monthlyFee)
    return confirmation
    * subscription status initializes to a default value when a user is created

17. updateUserInfo(firstName, lastName, phone, email, registrationDate, subscriptionStatus, monthlyFee)
    return confirmation
    * if NULL value provided for any inputs, those existing values will not be updated

18. deleteUserInfo(email)
    return confirmation

19. assignToAlbum(song(s), creatorId(s), albumName, trackNumber(s), releaseYear, edition)
    return confirmation

20. assignToRecordLabel(creatorId(s), recordLabel)
    return confirmation

21. assignToPodcast(creatorId(s), podcastEpisode(s), podcast)
    return confirmation

Maintaining Metadata and Records:

1. updateSongPlayCount(songTitle, creatorId, albumName, edition, playCount)
    if update successful:
        return True
    else:
        return False
    * all inputs required to ensure accurate update. Also acts as the enter operation, as a play count is initialized at 0 when a song is entered.

2. updateArtistMonthlyListeners(creatorId, monthlyListeners)
    if update successful:
        return True
    else:
        return False
    * all inputs required to ensure accurate update. Also acts as the enter operation, see above.

3. updatePodcastMetadata(podcast, subscribers, ratings)
    if update successful:
        return True
    else:
        return False
    * if NULL value provided for subscribers/ratings, those existing values will not be updated. Also acts as the enter operation, see above.

4. updatePodcastEpisodeListenerCount(listeningCount, podcast)
    if update successful:
        return True
    else:
        return False
    * Also acts as the enter operation, see above.

5. updateContractedArtists(creatorId(s), recordLabel)
    if update successful:
        return True
    else:
        return False

6. updateArtistStatus(creatorId, status)
    if update successful:
        return True
    else:
        return False

7. findSongs(creatorId, albumName)
    return list of all songs on given album (songTitle, artist(s), playCount, duration, track number).
    * If NULL passed for album, returns list of all songs associated with given artist. creatorId is a required input.

8. findPodcastEpisodes(creatorId, podcastName)
    return list of episodes associated with the podcast (episodeTitle, duration, releaseDate)
    * if NULL passed for podcastName, returns list of all podcast episodes associated with given podcast host, and vice versa

Maintaining Payments:

1. makeSongPayment(songTitle, albumName, edition, creatorId, month, year)
    return confirmation, amount paid to each party
    *if NULL passed for attributes other than month and year, payment generated for all songs

2. payPodcastHosts(podcastEpisode, podcastName, month, year)
    return confirmation, amount paid to each host per episode
    *if NULL passed for podcastEpisode and podcastName, payment generated for all podcastEpisodes

3. receivePayment(userEmail, month, year)
   return confirmation, amount paid by each user
   *if NULL passed for userEmail, payment received from all users

Reports:

1. getPlayCounts(songTitle, albumName, edition, creatorId, startDate, endDate, period)
   return summary of play counts by month
   * period input specifies whether it is a monthly, yearly, or total report. If no start or end date is provided, range is unbounded on the respective end. Same for all API functions below

2. getTotalHostPayment(creatorId, startDate, endDate, period)
   return amount paid to host in given period

3. getTotalArtistPayment(creatorId, startDate, endDate, period)
   return amount paid to artist in given period

4. getTotalRecordLabelPayment(recordLabel, startDate, endDate, period)
   return amount paid to record label in given period

5. getTotalPayout(startDate, endDate, period)
   return total amount paid out to artists/labels/podcast hosts in given time period

6. totalRevenue(startDate, endDate, period)
   return total payments received from subscribers on specified period basis since inception

7. reportSongs(creatorId, albumName)
   return list of all songs on given album (songTitle, artist(s), playCount, duration, track number). If NULL passed for album, returns list of all songs associated with given artist

8. reportPodcastEpisodes(podcastName)
   return list of episodes associated with the podcast (episodeTitle, duration, releaseDate, rating)

9. getPodcastSubscribers(startDate, endDate, period, podcastName)
   return aggregate subscribers for specified periodicity between startDate and endDate

10. getPodcastRatings(startDate, endDate, period, podcastName)
    return podcast rating

6. Description of Views

Management

Management in their capacities will view only artist, song, podcast episode, podcast, podcast host, and payment data.

Administrators

Since administrators have full control they will be able to view data related to songs, artists, record labels, albums, podcast, podcast hosts, and podcast episodes.

Artists

Artists are given a limited view of artists, albums, and songs data that is associated with them.

Podcast Hosts

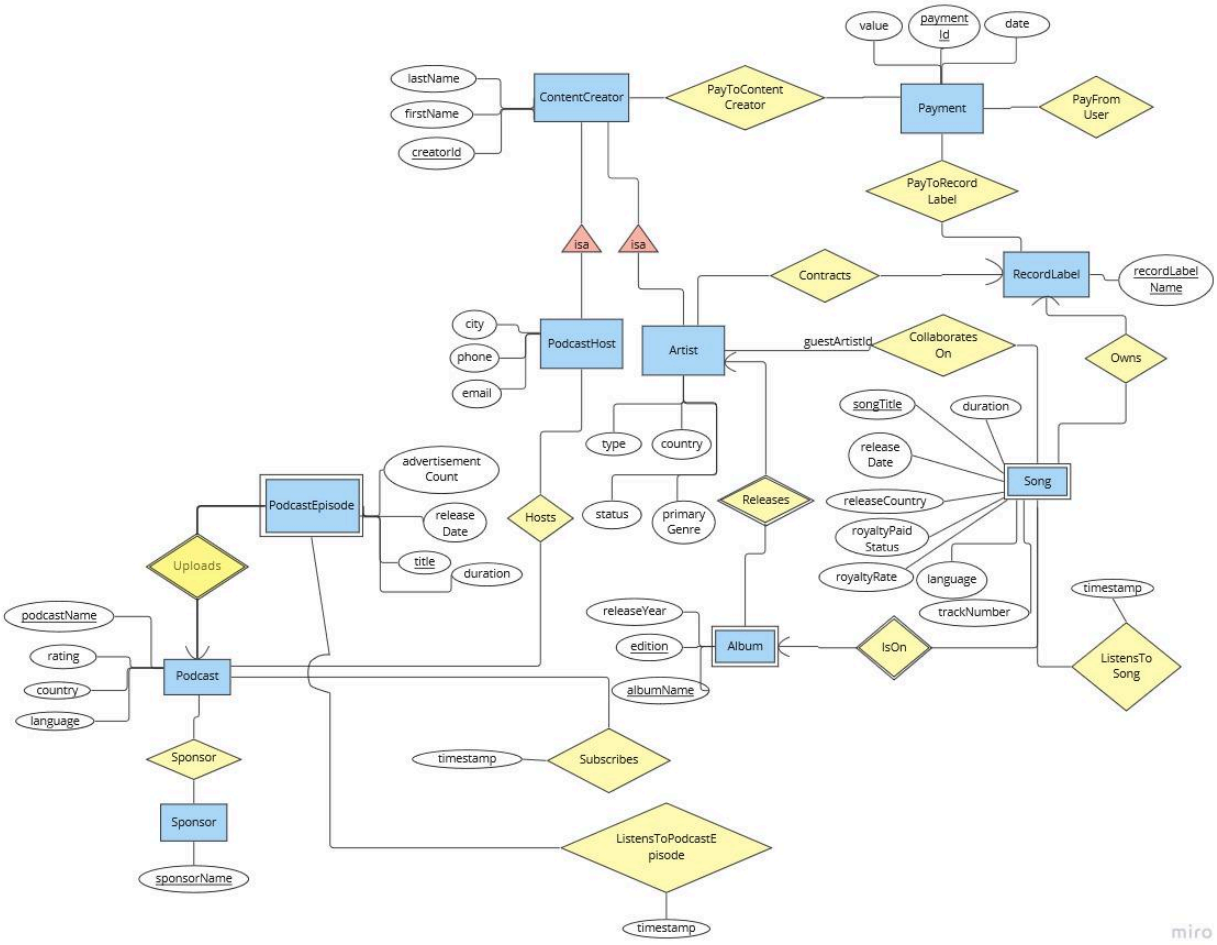
Podcast Hosts are given a limited view of podcast, podcast hosts, and podcast episodes data that is associated with them.

7. Local E/R Diagrams

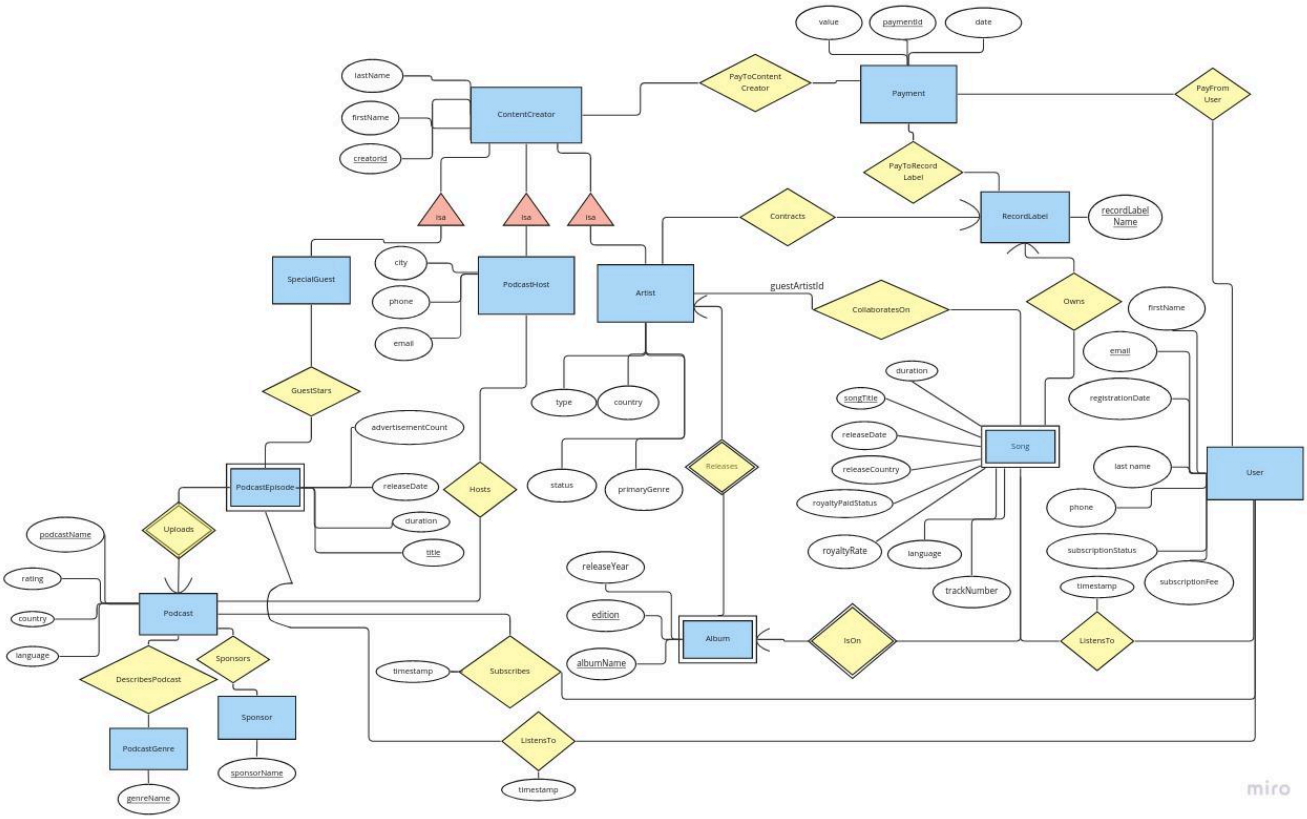
If the diagrams appear unclear, then use this link to find the Miro board that was used.
https://miro.com/app/board/uXjVPp03Sxc=/?share_link_id=808226277617

After selecting the links, if you cannot see the boards, then scroll out until you see the diagrams. The link is for anyone, so it should work without a Miro account.

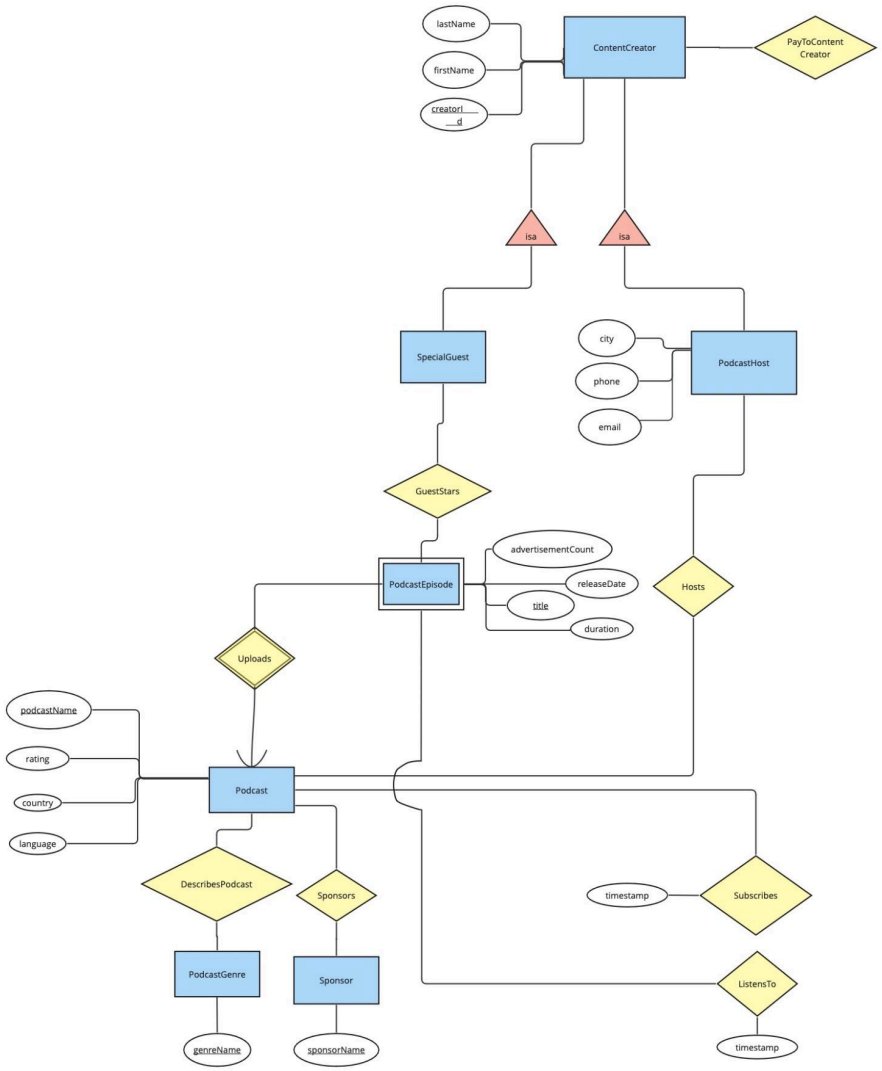
Management



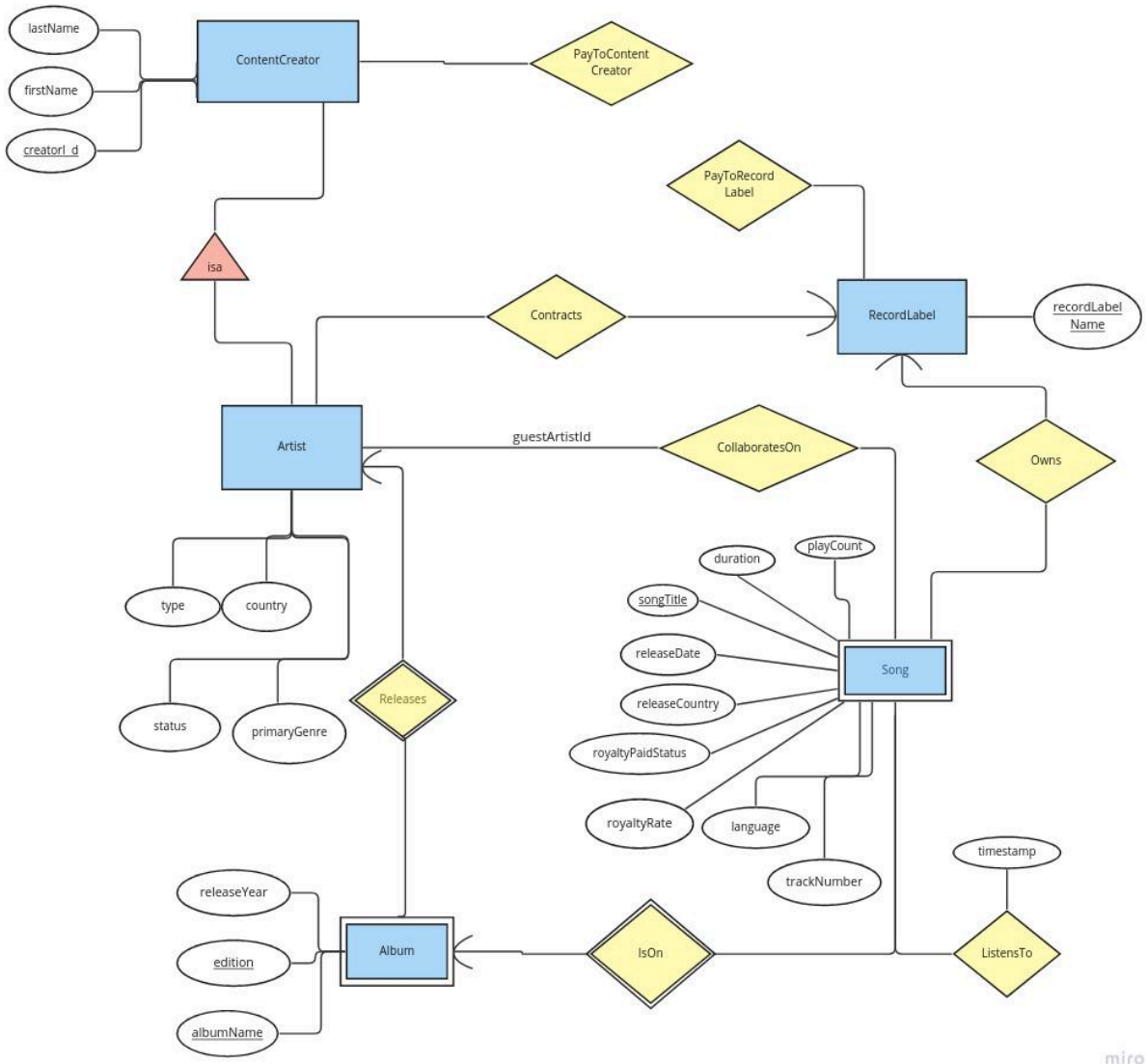
Administrators



PodcastHost



Artist



8. Local E/R Diagram Documentation

- Payments to Content Creators and Record Labels as well as those from Users are recorded in separate relationships that associate the paymentId (the key from Payments) with the appropriate key: creatorId, recordLabel, and userEmail, respectively.

- The Releases supporting relationship associates Artists with the weak entity set Albums with the creatorId (key) from artist, which are associated with the weak entity set Song through the IsOn supporting relationship using the albumName and edition attributes (the key). Artists can work on songs in the role of guest artist in the collaboratesOn relationship.
- Every artist is contracted by exactly one record label, which owns the songs the artist produces while under that contract.
- When a content creator joins the platform, they are given a unique id to differentiate them to help avoid confusion if two creators happen to have very similar names.
- Each record label can contract many artists, and every record label has a unique name (recordLabelName) as registered business entities. An artist can only be contracted by one record label at a time.
- Podcast hosts are content creators and content creators are uniquely identified by the creatorId. Podcast hosts may host more than one podcast and podcasts may have many podcast hosts.
- Sponsors use name as the key and may support many podcasts. Podcasts could also have many sponsors.
- Podcast genre use the name of the genre as the key. A podcast could have multiple genres and multiple podcasts could have the same genre.
- Podcast episodes is a weak entity set that uses the title of the episode and the name of the podcast it is associated with as a key. Podcast episodes may only be associated with one podcast in order to uniquely identify the episode. Users listen to podcasts and the timestamp of the user's action of listening is recorded.

- Users can listen to many podcast episodes and podcast episodes may have many listeners. Users can subscribe to many podcasts and podcasts can have many subscribers. The timestamp of the subscription is tracked for purposes of calculating monthly subscribers.
- Podcast uses the name of the podcast as the key, since there should not be more than one podcast with the same name.
- Special guests are content creators with their own unique creatorId as the key. Special guests may guest star on many podcast episodes and podcast episodes can have many special guests.

9. Local Relational Schemas

Management

ContentCreator(<u>creatorId</u>, firstName, lastName)

Payment(<u>paymentId</u>, date, value)
PayToContentCreator(<u>creatorId</u>, <u>paymentId</u>)

PayToRecordLabel(paymentId, recordLabelName)
PayFromUser(paymentId, email)

Song(creatorId, albumName, edition, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyPaidStatus, royaltyRate)
CollaboratesOn(creatorId, guestArtistId, songTitle, albumName, edition)
Album(albumName, edition, releaseYear)

Artist(creatorId, name, status, type, country, primaryGenre)
Contracts(creatorId, recordLabelName)
RecordLabel(labelName)
Owns(creatorId, recordLabelName, songTitle, albumName, edition)

PodcastHost(creatorId, email, phone, city)
Hosts(podcastName, creatorId)
Podcast(podcastName, language, country, rating)
Sponsors(podcastName, sponsorName)
Sponsor(sponsorName)
PodcastEpisode(title, podcastName, duration, releaseDate, advertisementCount)

ListensToSong(email, songTitle, id, albumName, edition, timestamp)
SubscribesToPodcast(userEmail, creatorId, timestamp)
ListensToPodcastEpisode(userEmail, podcastEpisodeTitle, timestamp)

Admin

ContentCreator(creatorId, firstName, lastName)

Payment(paymentId, date, value)
PayToContentCreator(creatorId, paymentId)
PayToRecordLabel(paymentId, recordLabelName)
PayFromUser(paymentId, email)

Song(creatorId, albumName, edition, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyPaidStatus, royaltyRate)
CollaboratesOn(creatorId, guestArtistId, songTitle, albumName, edition)
Album(albumName, edition, releaseYear)

Artist(creatorId, name, status, type, country, primaryGenre)
Contracts(creatorId, recordLabelName)
RecordLabel(labelName)
Owns(creatorId, recordLabelName, songTitle, albumName, edition)

PodcastHost(creatorId, email, phone, city)

Hosts(podcastName, creatorId)
Podcast(podcastName, language, country, rating)
Sponsors(podcastName, sponsorName)
Sponsor(sponsorName)
PodcastGenre(genreName)
DescribesPodcast(podcastName, genreName)
PodcastEpisode(title, podcastName, duration, releaseDate, advertisementCount)
GuestStars(creatorId, title, podcastName)
SpecialGuest(creatorId)
User(firstName, lastName, email, subscriptionFee, statusOfSubscription, phone, registrationDate)
ListensToSong(email, songTitle, id, albumName, edition, timestamp)
SubscribesToPodcast(userEmail, creatorId, timestamp)
Pays(paymentId, userEmail)
ListensToPodcastEpisode(userEmail, podcastEpisodeTitle, timestamp)

Podcast Host

ContentCreator(creatorId, firstName, lastName)
PayToContentCreator(creatorId, paymentId)

PodcastHost(creatorId, email, phone, city)
Hosts(podcastName, creatorId)
Podcast(podcastName, language, country, rating)
Sponsors(podcastName, sponsorName)
Sponsor(sponsorName)
PodcastGenre(genreName)
DescribesPodcast(podcastName, genreName)
PodcastEpisode(title, podcastName, duration, releaseDate, advertisementCount)
GuestStars(creatorId, title, podcastName)
SpecialGuest(creatorId)

ListensToPodcastEpisode(userEmail, podcastEpisodeTitle, timestamp)
SubscribesToPodcast(userEmail, creatorId, timestamp)

Artist

ContentCreator(creatorId, firstName, lastName)
PayToContentCreator(creatorId, paymentId)

Song(creatorId, albumName, edition, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyPaidStatus, royaltyRate)
CollaboratesOn(creatorId, guestArtistId, songTitle, albumName, edition)
Album(albumName, edition, releaseYear)

Artist(creatorId, name, status, type, country, primaryGenre)
Contracts(creatorId, recordLabelName)
RecordLabel(labelName)
Owns(creatorId, recordLabelName, songTitle, albumName, edition)

ListensToSong(email, songTitle, id, albumName, edition, timestamp)

10. Local Relational Schema Documentation

We used AexemOne.pdf from Moodle's Sample past project report 1 as a model.

Entity Sets to Relations

- The entity sets in the E/R diagrams were made into relations. The keys in the entity sets were translated into keys in the relations and the attributes in entity sets were translated into attributes in the relations. Examples include Podcast and User.
- Weak entity sets were translated into relations by using their own keys coupled with keys from the entity sets connected by supporting relations. Attributes for weak entity sets became attributes for the weak entity relations.
- The entity sets that are subsets of Content Creators were made into relations based on the E/R viewpoint to facilitate better querying through explicitness and partitioning.

Combining Many-One Relationships:

- The many to one relationships were converted to reduce anomalies within the database schema. For example, the weak entity set podcast episode relies on its relationship with Podcast to uniquely identify itself. For this relationship, the relationship, Uploads, was not converted into a relation. Instead, the Podcast Episode relation includes the key from the Podcast.

Relationships to Relations:

- The relationships were converted to relations. The keys for these relations are the keys from the entity sets that are included in the relationship. Any attributes attached to the relationships are included in the relation as well. Examples include Hosts which is a relation that includes the keys for the podcaster and the content creatorId. This relation describes which podcast hosts host which podcasts.