WolfMedia Database Management System

Team S

Ameya Girish Vaichalkar (agvaicha)

Kartik Hemendra Rawool (khrawool)

Dhrumil Jignesh Shah (dshah6)

Subodh Suryakant Gujar (sgujar)

Contents

Assumptions

1. Artist can have a contract with exactly one record label at a time.
2. Podcast is deleted then all the episodes within that podcast will also be deleted.
3. Album is deleted then all the album tracks within that album will also be deleted.
4. All artists, record labels, and podcast hosts are paid on first day of the month.
5. Users pays subscription fees on the first day of the month.
6. Monthly active listeners include only the users who have subscribed to the artist or the podcast host.
7. Phone numbers entered by users are unique.
8. Song will always be part of at most one album.
9. Each track number is unique within an album. For eg., multiple albums can have same track number.
10. Each episode number is unique within a podcast. For eg., multiple podcast episode can have the same episode number.
11. Different tables are maintained for keeping the records of song history, artist history, and podcast history.
12. Data will be added in History tables only at the last day of the month.
13. Each song has exactly one primary artist.
14. Royalties will be paid for the songs to the record label of the primary artist.
15. Service account represents all the transactions done by and to WolfMedia Streaming service.
16. Podcast ratings will be in the range of 0-5.
17. There are multiple plans for subscription to WolfMedia. Each plan has different subscription fees.

1 Problem Statement

The media streaming service needs to manage a large amount of data related to songs, podcasts, artists, record labels, hosts, and episodes. The service generates monthly payments based on royalties collected from song play counts and podcast advertisements. The royalties are paid out to record labels and artists, while podcast hosts are paid a flat fee and bonus. The service needs to keep track of all payment information and generate reports on various aspects of the streamed items and associated parties, including monthly and yearly summaries of payments, subscribers, play count, and ratings. The challenge is to design a database system that can efficiently store and manage all this information, provide easy access for analysis, and ensure data integrity and security.

Why Database ?
A database is a good idea in this challenge for organizing and storing data because it provides several advantages over simple files. Some of these advantages include:

- *Structured Data Management*
  A database allows us to store data in a structured manner, meaning we can define relationships between different data elements, enforce constraints, and ensure data consistency and integrity.

- *Query Ability*
  Databases have built-in query languages like SQL that allow us to easily retrieve and manipulate the data stored in the database. This makes it much easier to find specific pieces of information and perform complex data analysis.

- *Scalability*
  Databases are designed to handle large amounts of data and many concurrent users. They can be easily scaled to accommodate growing data needs and increasing numbers of users.

- *Data Security*
  Databases have built-in security features like user authentication, access control, and data encryption, which can help protect sensitive data from unauthorized access and ensure data privacy and confidentiality.

- *Concurrent Access*
  Databases allow multiple users to access and modify the data stored in the database at the same time. This makes it easier to collaborate and share information with others.

2 Intended Users

We have identified the following users for our system:

Admin: Admins will have full access to the database system, allowing them to maintain, update, and analyze information related to the media streaming service. Admins will have the ability to create new records, edit existing ones, and delete records as necessary.

Artist: Artist User is designed to store and maintain information on individual artists, including bands, solo musicians, and composers, who contribute to the audio and media content on the platform. Artists can view songs and albums he has made. As an artist, you will have access to information about your contributions to the platform, including the number of monthly listeners and the total amount of royalties earned.

Podcast Host: Podcast Hosts are individuals who create and host podcasts for the streaming service. They will be able to view, add and edit information about their episodes, such as episode title, duration, release date, and listening count. The Podcast Hosts will also have access to monthly, yearly, and total reports on their podcast subscribers, episode listening count, and sponsor information.

Listener: The End User, or Listener, is the primary target audience for the Wolf-Media media streaming service. This class of users will have access to the database system to listen to a wide variety of songs and podcasts from various artists and podcast hosts.

Management: The Management user class is intended for those individuals within the media streaming service who need to view and analyze all data stored in the database system. They do not have the ability to modify or update any information in the database, but they have full access to view all data and use it to generate reports and gain insights. This user class is similar to the Admin class, but with limited access rights to maintain data integrity.

Record Label: This user will have access to the information related to their contracted artists, albums, and songs within the database system. They will also be able to see the royalties generated from the songs and the payments made to the artists, including themselves.

3 Five Main Things

1. Song:
songID, song title, artist(s), duration, genre(s), album, play count, release date, release country, language, royalty rate, collaborators (guest artist), royalty paid(status of whether the current royalty has been paid out)

2. Artist:
artistID, name, collaboration(s), status (active/retired), type (Band/musician/composer), country, primary genre, monthly listeners, album(s), record label

3. Podcast:
podcastID, podcast host, podcast name, podcast episode(s), language, country, episode count, genre(s), rating, sponsor(s), total subscribers

4. Payment Service:
Record Label payment(s), Podcast host payment(s), User subscription(s)

5. User:
userID, first name, last name, phone, email, registration date, status of subscription, podcast(s) subscribed, artist(s) subscribed, status of subscription, monthly subscription fee

4 Tasks and Operations-Realistic Situations

Situation 1

New podcast episode is released:
When a new podcast episode is released on the WolfMedia streaming service, the information of the podcast episode is entered, and the episode is assigned to the respective podcast it belongs to and updates the episode count of podcast.

Situation 2

Monthly Payments:
Every month, WolfMedia generates royalty payments for the songs to the record label based on royalty rate and for the podcast to the podcast host as a flat fee. The transactions are stored in the database.

5 Application Program Interfaces

5.1 Information Processing

- addSongDetail (songTitle, artists, duration, genre, album, playCount, releaseDate, releaseCountry, language, royaltyRate, collaborators, royaltyPaid)
  - return confirmation and songId
- addArtistDetail (name, collaboration, status, type, country, primaryGenre, monthlyListeners, albums, recordLabel)
  - return confirmation and artistId
- addPodcastHostDetail (firstName, lastName, phone, email, city, podcasts)
  - return confirmation and podcastHostId
- addPodcastEpDetail (episodeTitle, duration, releaseDate, listeningCount, specialGuests, adCount)
  - return confirmation and podcastEpId
- updateSongDetail (songId, songTitle, artists, duration, genre, album, playCount, releaseDate, releaseCountry, language, royaltyRate, collaborators, royaltyPaid)
  - return confirmation
  - If the attributes songTitle, artists, duration, genre, album, playCount, releaseDate, releaseCountry, language, royaltyRate, collaborators, royaltyPaid are not given, then the update will not happen.
- updateArtistDetail (artistId, name, collaboration, status, type, country, primaryGenre, monthlyListeners, albums, recordLabel)
  - return confirmation
  - If the attributes name, collaboration, status, type, country, primaryGenre, monthlyListeners, albums, recordLabel are not given, then the update will not happen.
- updatePodcastHostDetail (podcastHostId, firstName, lastName, phone, email, city, podcasts)
  - return confirmation
  - If the attributes firstName, lastName, phone, email, city, podcasts are not given, then the update will not happen.
- updatePodcastEpDetail (podcastEpId, episodeTitle, duration, releaseDate, listeningCount, specialGuests, adCount)
  - return confirmation

  - If the attributes episodeTitle, duration, releaseDate, listeningCount, specialGuests, adCount are not given, then the update will not happen.

- deleteSongDetail (songId)
  - return confirmation if exist
- deleteArtistDetail (artistId)
  - return confirmation if exist
- deletePodcastHostDetail (podcastHostId)
  - return confirmation if exist
- deletePodcastEpDetail (podcastEpId)
  - return confirmation if exist
- assignSongToAlbum (songId, albumId)
  - return confirmation if both attributes exist
- assignArtistToAlbum (artistId, albumId)
  - return confirmation if both attributes exist
- assignArtistToRecordLabel (artistId, recordLabelId)
  - return confirmation if both attributes exist
- assignPodcastEpToPodcast (podcastEpId, podcastId)
  - return confirmation if both attributes exist
- assignPodcastHostToPodcast (podcastHostId, podcastId)
  - return confirmation if both attributes exist

s

5.2 Maintaining metadata and records:

- addPlayCountSong(songId, playCount)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- updatePlayCountSong(songId, playCount)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- addMonthlyListeners(artistId, songId, monthlyListeners)

  - returns confirmation
  - NULL error if any of the attributes is not provided
- updateMonthlyListeners(artistId, songId, monthlyListeners)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- addTotalSubscribers(podcastId, totalSubscribers)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- updateTotalSubscribers(podcastId, totalSubscribers)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- addPodcastRatings(podcastId, rating)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- UpdatePodcastRatings(podcastId, rating)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- addPodcastEpisodeListenerCount(podcastId, episodeId, listenerCount)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- updatePodcastEpisodeListenerCount(podcastId, episodeId, listenerCount)
  - returns confirmation
  - NULL error if any of the attributes is not provided
- findSongsByArtist(artistId)
  - returns all songs of a artist if artistId present else return false
- findSongsByAlbum(albumId)
  - returns all songs in an album if albumId present else return false
- findPodcastEpisodes(podcastId)
  - returns episodes in a podcast if podcastId present else return false
  - NULL error if any of the attributes is not provided

5.3 Maintaining Payments

- makeRoyaltyPayment(songID)
  - returns confirmation if the royalty payment for the given song was successful
- generateMonthlyRoyalties()
  - returns calculated monthly royalties for each song
- makePaymentToPodcastHostForBonus()
  - returns confirmation if payment to the podcast hosts was successful
- makePaymentToPodcastHosts()
  - returns confirmation if payment to the podcast hosts was successful
- receivePaymentFromSubscribers()
  - returns confirmation if the payment was successful

5.4 Reports

- getPlayCountPerSongPerMonth()
  - returns play count per month for all songs mentioned
- getPlayCountPerAlbumPerMonth()
  - returns play count of total Songs in each Album per month
- getPlayCountPerArtistPerMonth()
  - returns play count of total Songs associated with each artist per month
- calculateHostPayments(listOfHostIds, startDate, endDate)
  - returns Payments made out to the given Podcast Hosts in the given time period
- calculateArtistPayments(listOfArtistIds, startDate, endDate)
  - returns Payments made out to Artists in the given time period
- calculateRecordLabelPayments(listOfRecordLabelIds, startDate, endDate)
  - returns Payments made out to RecordLabels in the given time period
- calculateMonthlyRevenue()
  - returns total revenue generated per month

- calculateYearlyRevenue()
  - returns total revenue generated per year
- getSongsByArtist(artistID)
  - returns the songs that belong to the artist
- getSongsByAlbum(albumID)
  - returns the songs that belong to the album
- getPodcastEpisodesByPodcast(podcastID)
  - returns the Podcast Episodes that belong to the podcast

6 Description of Views

1. **Admin:**
   Admin have access to all the entities in the database. Admin can access all the information related to Songs, Albums, Artist, Podcast, Podcast Host, Record Label and the payment information.

2. **Artist:**
   Artist has access to the information relating to itself, Record Label, songs artist has created and the details of the album associated with the songs. It also has the details of the royalty payment paid to him/her by record label.

3. **Podcast Host:**
   Podcast Host will be able to view information about the podcasts and their episodes, such as episode title, duration, release date, and listening count. Podcast Host can view the details of payments from the Service Account.

4. **User(Listener):**
   This class of users will have access to the database system to browse, search and listen to a wide variety of songs and podcasts from various artists and podcast hosts. Listeners can see the details of its subscription fee payments to the Service Account.

5. **Management:**
   The Management can access all the tables in the database with view only access. It would also view the history tables for song, artist and podcast.

6. **Record Label:**
   The Record Label view the information related to artist, song and album. It can also view the payment information of royalties from the Service Account and the transaction history of royalty payments made to the artists.

7 Local E/R Diagrams

7.1 Admin (Read/Write Access)

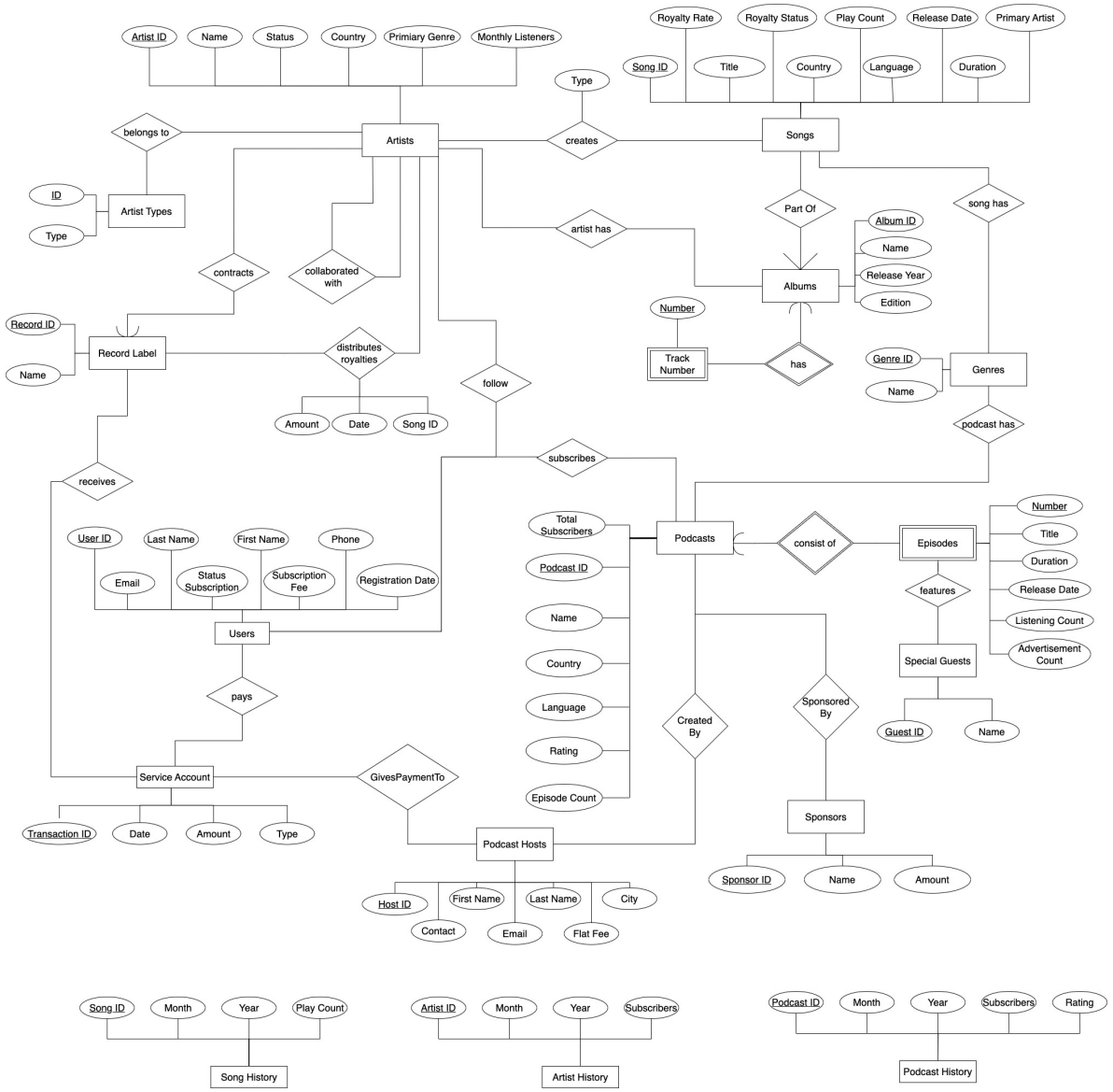

Figure 1: Admin Local ER Diagram

7.2 User(Listener)

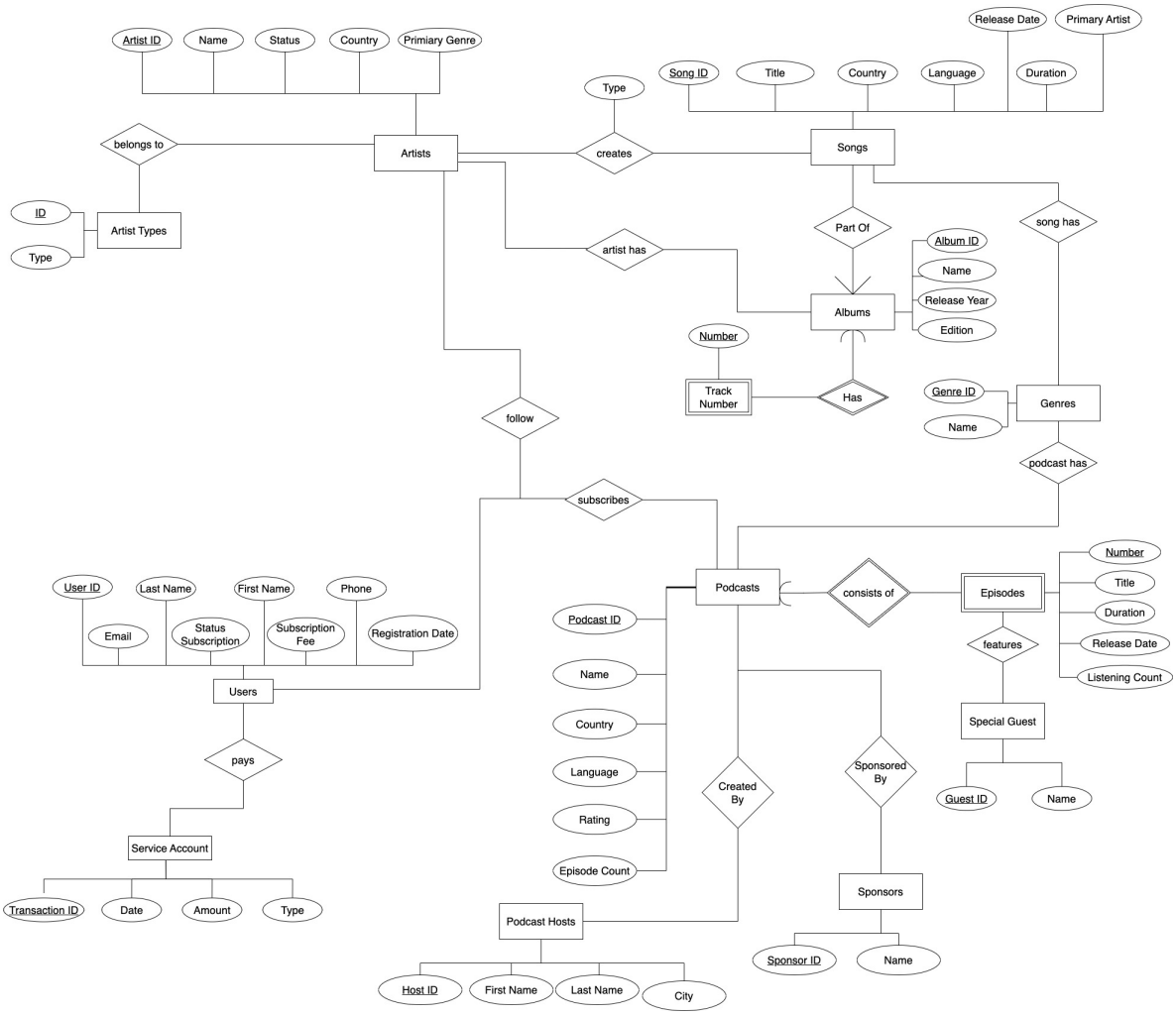

Figure 2: Listener Local ER Diagram

7.3 Artist

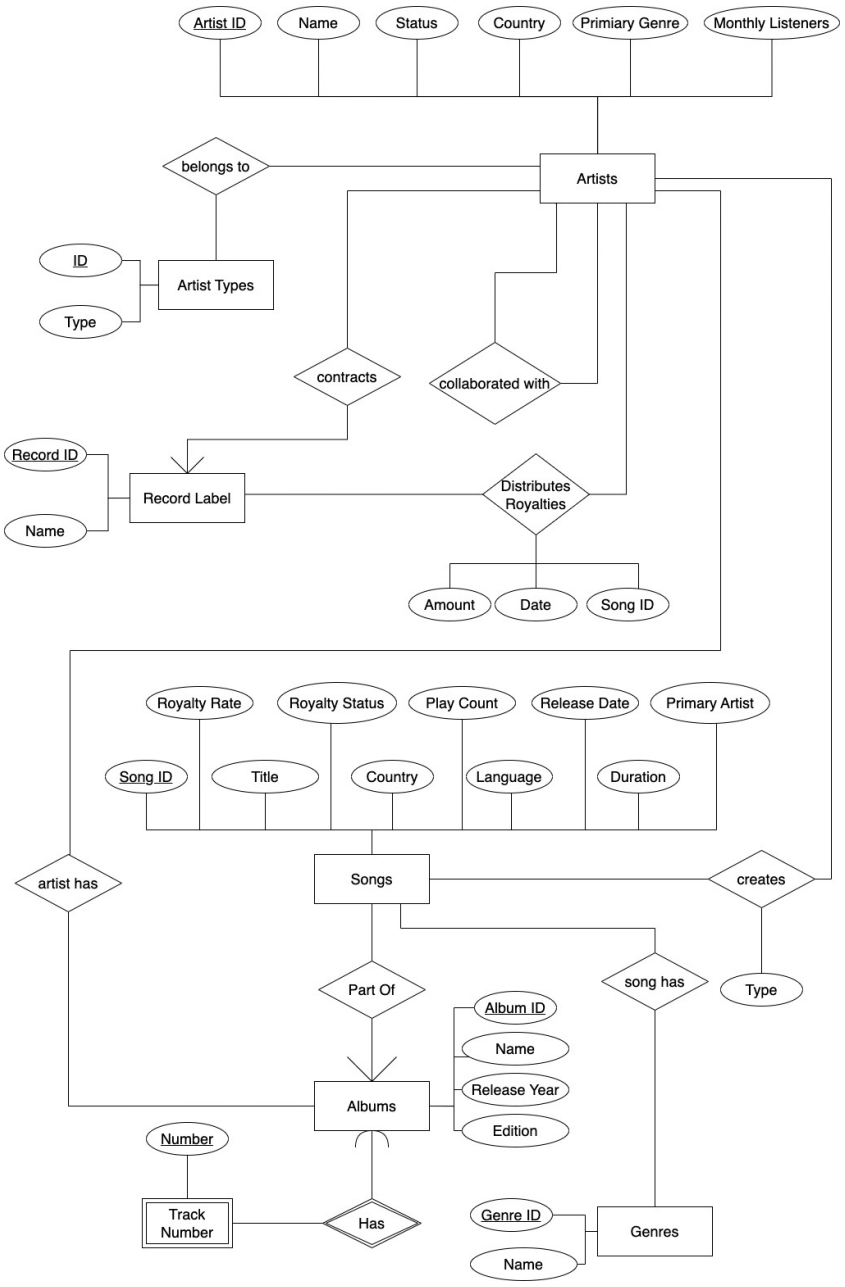

Figure 3: Artist Local ER Diagram

7.4 Podcast Host

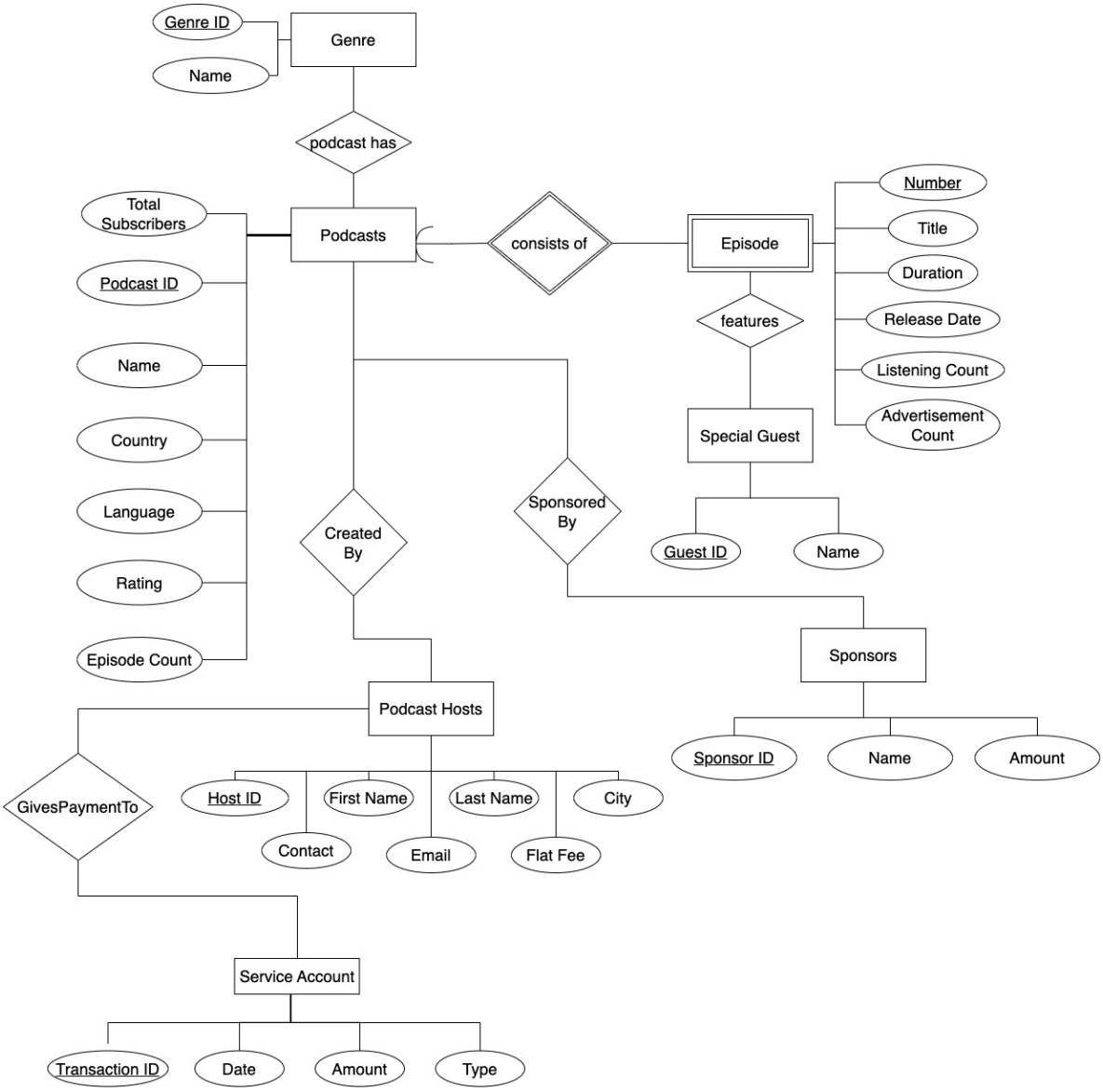

Figure 4: Podcast Host Local ER Diagram

7.5 Management Staff (Read Only Access)

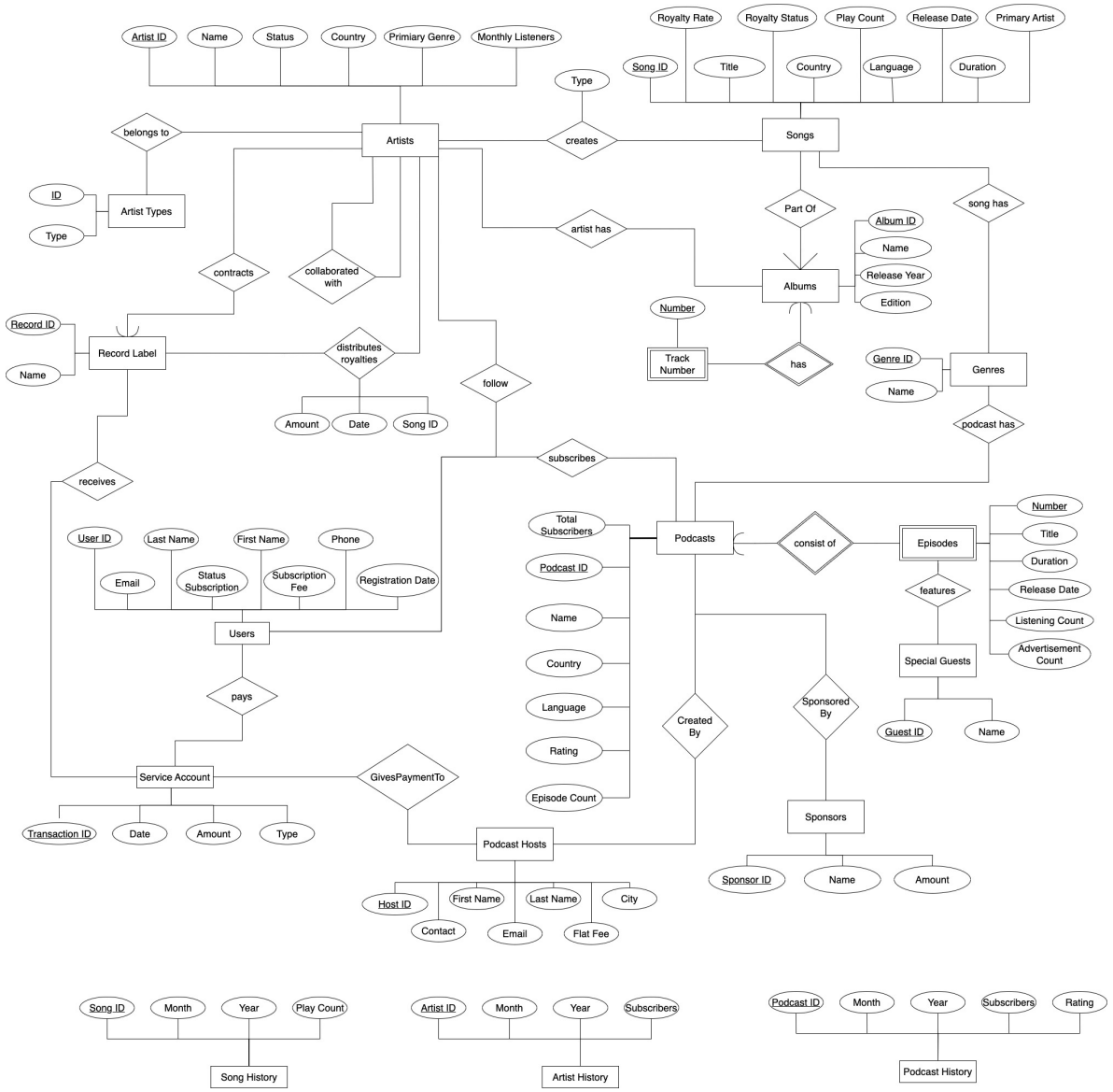

Figure 5: Management Local ER Diagram

7.6 Record Label

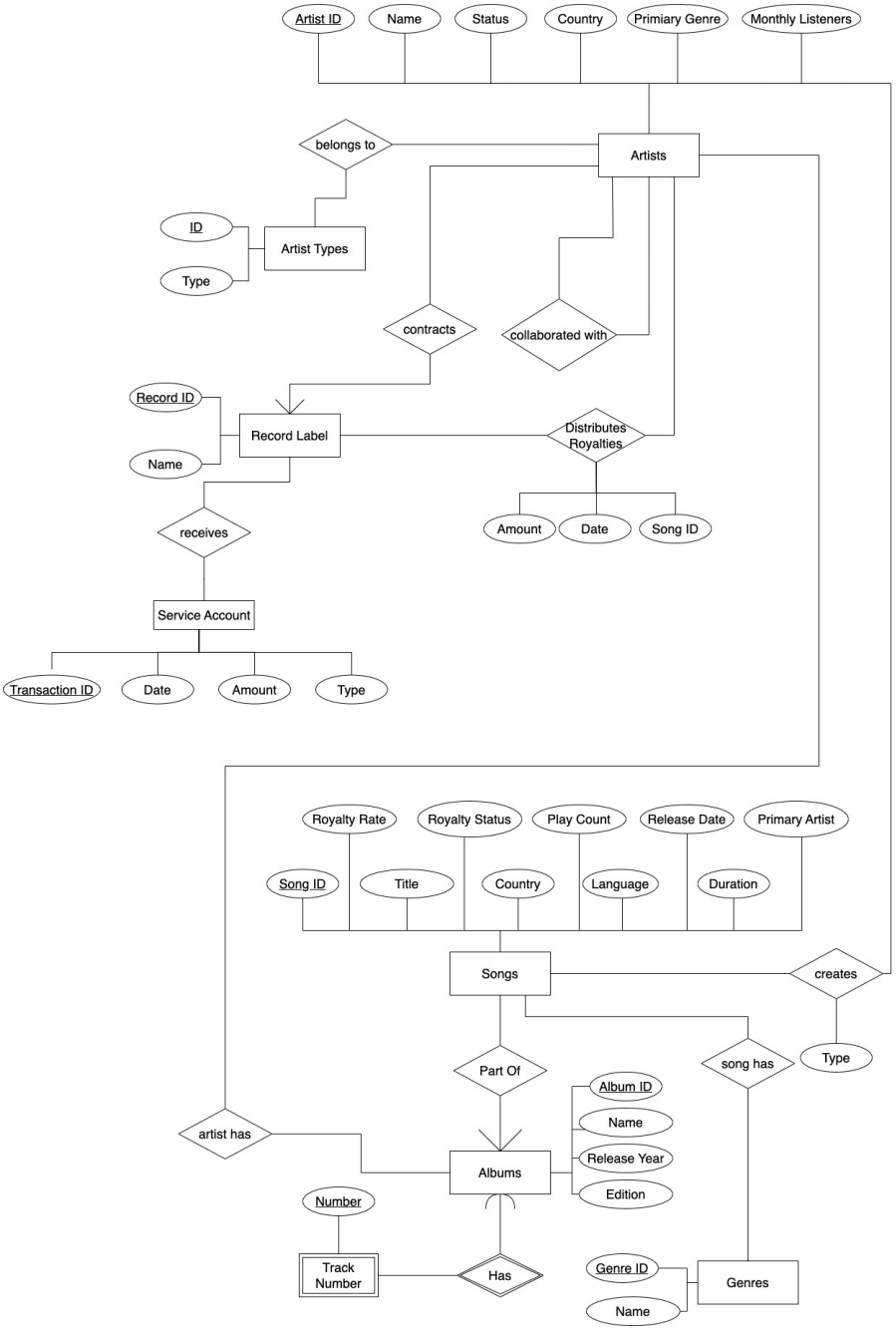

Figure 6: Record Label Local ER Diagram

8 Description of Local E/R Diagrams

- The primary artist is stored as an attribute since it will never be null and have only a single value.
- The collaborations of a songs are represented through a relationship between song and artist.
- The Artist type is created as a separate entity since one artist can be of multiple type, to maintain the 1NF property.
- The relationship between album and song is maintained through a relationship entity, allowing for multiple songs to be associated with one album.
- Track Number is identified as a weak entity as track number can be uniquely identified within a given Album.
- Record label is maintained as an entity with a relationship to contracted artists to keep track of which artists are under contract with the record label.
- Artist has a exactly one contracts relationship with Record Label since Artist can be only be contracted by one Record Label.
- Podcast host is maintained as an entity with a relationship to the podcasts they host to keep track of all the podcasts a host is associated with.
- The relationship between podcast episode and podcast is maintained through a relationship entity, allowing for multiple episodes to be associated with one podcast and vice versa.
- Podcast Episode is identified as weak entity as it cannot exist without Podcast and it can be identified uniquely within a Podcast.
- The User(Listener) is maintained as an entity with an attribute of subscription status, allowing for easy tracking of active and inactive subscribers.
- The relationship between artist and record label is maintained to keep track of the royalties paid to each artist.
- The relationship between podcast host and episode count is maintained to keep track of the total number of episodes hosted by each host.
- Service Account represents the Account of WolfMedia Streaming Service.
- Sponsors for the Podcast are created as separate entity as a there can be multiple sponsors for a single podcast.

9 Local Relational Schema

Admin:

1. songs(songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)
2. album(albumId, name, releaseYear, edition)
3. trackNumber(albumId,number)
4. genre(genreId, name)
5. SongHas(songId,genreId)
6. artistHas(artistId, albumId)
7. creates(artistId, songId, type)
8. Podcast(podcastId, name, country, language, rating, episodeCount, totalSubscribers)
9. episode(podcastId, number, title, duration, releaseDate, ListeningCount, AdvertisementCount)
10. episodeFeaturesGuest(podcastId, number, guestId)
11. ~~consistOf(podcastId, number)~~
12. podcastHas(podcastId, genreId)
13. podcastHistory(podcastId, month, year, subscribers, rating)
14. specialGuests(guestId, name)
15. createdBy(podcastId, hostId)
16. sponsors(sponsorId, name, amount)
17. sponseredBy(sponsorId, podcastId)
18. artist(artistId, name, status, country, primaryGenre, monthlyListeners, recordId)
19. belongsTo(id, artistId)
20. artistType(id, type)
21. collaboratedWith(artistId1, artistId2)
22. recordLabel(recordId, name)
23. receives(transactionId, recordId)
24. user(id, email, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)

25. serviceAccount(transactionId, date, amount, type)
26. songHistory(songId, month, year, playCount)
27. pays(userId, transactionId)
28. podcastHost(podcastId, Contact, firstName, lastName, contact, email, flatFee, city)
29. follow(artistId, userId)
30. subscribes(podcastId, userId)
31. listensTo(userId, podcastId)
32. givesPaymentTo(transactionId, HostId)
33. artistHistory(artistId, Month, Year, Subscribers)
34. distributesroyalties(artistId, recordId, songId, date, amount)

User(Listener):

1. songs(songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)
2. album(albumId, name, releaseYear, edition)
3. trackNumber(albumId,number)
4. genre(genreId, name)
5. SongHas(songId, genreId)
6. artistHas(artistId, albumId)
7. creates(artistId, songId, type)
8. podcast(podcastId, name, country, language, rating, episodeCount)
9. episode(podcastId,number, title, duration, releaseDate, ListeningCount)
10. episodeFeaturesGuest(podcastId, number, guestId)
11. ~~consistOf(podcastId, number)~~
12. podcastHas(podcastId, genreId)
13. specialGuests(guestId, name)
14. createdBy(podcastId, HostId)
15. sponsors(sponsorId, name)
16. sponseredBy(sponsorId, podcastId)

17. artist(artistId, name, status, country, primaryGenre, monthlyListeners)
18. belongsTo(id, artistId)
19. artistType(id, type)
20. user(id, email, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)
21. serviceAccount(transactionId, date, amount, type)
22. pays(userId, transactionId)
23. podcastHost(podcastId, Contact, firstName, lastName, contact, email, flatFee, city)
24. subscribes(artistId, userId)
25. listensTo(userId, podcastId)

Artist:

1. songs(songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)
2. album(albumId, name, releaseYear, edition)
3. trackNumber(albumId,number)
4. genre(genreId, name)
5. SongHas(songId, genreId)
6. artistAlbum(artistId, albumId)
7. creates(artistId, songId, type)
8. artist(artistId, name, status, country, primaryGenre, monthlyListeners, recordId)
9. belongsTo(id, artistId)
10. artistType(id, type)
11. collaboratedWith(artistId1, artistId2)
12. recordLabel(recordId, name)
13. receives(transactionId, recordId)
14. user(id, email, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)
15. serviceAccount(transactionId, date, amount, type)
16. songHistory(songId, month, year, playCount)

17. distributesroyalties(artistId, recordId, songId, date, amount)

Podcast Host:

1. Podcast(podcastId, name, country, language, rating, episodeCount, totalSubscribers)
2. episode(podcastId, number, title, duration, releaseDate, ListeningCount, AdvertisementCount)
3. episodeFeaturesGuest(podcastId, number, guestId)
4. ~~consistOf(podcastId, number)~~
5. podcastHas(podcastId, genreId)
6. specialGuest(guestId, name)
7. createdBy(podcastId, podcastHostId)
8. sponsors(sponsorId, name, amount)
9. sponseredBy(sponsorId, podcastId)
10. receives(transactionId, recordId)
11. user(id, email, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)
12. serviceAccount(transactionId, date, amount, type)
13. pays(userId, transactionId)
14. podcastHost(podcastId, Contact, firstName, lastName, contact, email, flatFee, city)
15. listensTo(userId, podcastId)
16. givesPaymentTo(transactionId, podcastHostId)

Record Label:

1. songs(songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)
2. album(albumId, name, releaseYear, edition)
3. trackNumber(albumId,number)
4. genre(genreId, name)
5. SongHas(songId, genreId)
6. artistHas(artistId, albumId)

7. creates(artistId, songId, type)
8. sponsors(sponsorId, name, amount)
9. sponseredBy(sponsorId, podcastId)
10. artist(artistId, name, status, country, primaryGenre, monthlyListeners, recordId)
11. belongsTo(id, artistId)
12. artistType(id, type)
13. collaboratedWith(artistId1, artistId2)
14. recordLabel(recordId, name)
15. receives(transactionId, recordId)
16. serviceAccount(transactionId, date, amount, type)
17. songHistory(songId, month, year, playCount)
18. subscribes(artistId, userId)
19. artistHistory(artistId, month, year, subscribers)
20. distributesroyalties(artistId, recordId, songId, date, amount)

10 Local Schema Documentation

Entity Sets to Relations

- All the entity sets in our E/R diagram were made into relation schemas with the same attributes.

- The conversion to relations from E/R for the different views was done as per the point of view of the user. For example, a listener cannot view the royalty rate of the song.

Combining Many-One Relationships

- In the following part, for many to one relations in the E/R, an attribute was added to the Entity Set on the many side. Adding an attribute will eliminate redundancy.

  - Entity set Artists has all its attributes plus an attribute record Record ID that comes from combining its many-one relationship to Record Label. Thus we do not have a separate relationship for the "contracts" relation from the E/R diagram.
  - Entity set Songs has all its attributes plus an attribute Album ID that comes from combining its many-one relationship to Albums. Thus we do not have a separate relationship for the "Part Of" relation from the E/R diagram.

Relationships to Relations

- Relationships for many to many associations(example subscribes, follow), relations with attributes(distributes royalties), and relations involving weak entity sets(example features) have been converted into relations.