PROJECT REPORT - 1

WolfMedia Streaming Service

CSC 540 Database Management concepts and Systems

Created By:-

Team M

Dhruv Mukesh Patel (dpatel49)
Harshil Tushar Sanghavi (hsangha)
Mitul Patel (mpatel27)
Vishwa Hiren Gandhi (vgandhi)

Assumptions:

- Record Label name is unique
- Phone is unique for every user
- Artists can be under contract with only one record label.

1. Problem Statement:

In this project we are going to design and build a WolfMedia streaming system to maintain information regarding songs, artists, albums, podcasts, subscribers, royalties and payments. There are majorly four tasks that need to be accomplished: Information processing, maintaining metadata and records, generating royalties and payments, and making reports that contain data for calculating total royalties, total play counts, monthly listeners, ratings, subscribers , podcast episodes and other details related to the streaming service. The royalty rate and play count is used to calculate the royalties paid to the record label. In addition, listening count for podcasts and play count for songs is also maintained.

As the streaming services are used worldwide by multiple users at the same time it can be better to use a database system rather than simple files. Moreover by using the file system there can be chances of losing important data like payments, play count/subscribers if the system gets crashed. Apart from that, some complex tasks such as searching for a song, artist or a podcast can be easily done using a database instead of visiting multiple files. In addition to that, files may get outdated which may result in data skewing. Database provides more security than the file system and is easy to maintain. Having said that, it is more appropriate to use a database rather than the simple files to maintain the data.

2. Describe the intended classes of users of your database system:

Admin/Management: have access to all the information pertaining to the WolfMedia streaming services. The admin is responsible for adding, updating and deleting the data related to artist, song, album, podcast, podcast episode, podcast host and users. They have complete control of the media streaming services. Admin is also responsible for all payments.

Artists: They are responsible for creating the song. They have a contract with the record label.

Podcast Hosts: These are the people who host the podcast episodes.

Record Labels: They are responsible for managing the artists and also are the owners of the respective songs created by the particular artist.

Users: They are the people who listen to the songs and subscribe to podcasts. They are paying a monthly subscription fee for using this streaming service.

3. **Identify 5 main "things" about which you will need to keep information, and the information you will need to keep. Submit the names of the 5 "things" and the information you need to keep about them:**

    1. **Media:** mediaid, media_name, genre, language, and m_country.
    2. **Creators:** creatorsid, cf_name, and cl_name
    3. **Payment:** paymentid, pay_amount, pay_type, ps_date, pe_date, person_id, flat_fee, and bonus.
    4. **User :** uphone, start_date, end_date, uf_name, ul_name, u_email, u_status and sub_fee.
    5. **Album:** albumid, edition, release_year, album_name, and track_no.

4. **Tasks and Operations:**

    - **Situation 1**

      The month has ended so the admin makes payment to the record labels and podcast hosts according to the data collected. Then the balance is also updated for it. The management then generates the monthly report for it.

    - **Situation 2**

      The artist releases a new song so the information is processed in the database by the admin for that. As the play count for the song keeps on increasing during the period of time the admin updates the database for that.

5. **Application Program Interfaces:**

Information Processing:

enterSongMediaInfo(mediaid, media_name, genre, language, m_country, collaborators, duration, play_count, release_date, royalty_rate, royalty_paid)

return confirmation

updateSongMediaInfo(mediaid, media_name, genre, language, m_country, collaborators, duration, play_count, release_date, royalty_rate, royalty_paid)

return confirmation

* If NULL value for any of the fields, then they will not be updated

deleteSongMediaInfo(mediaid, media_name, genre, language, m_country, collaborators, duration, play_count, release_date, royalty_rate, royalty_paid)

return confirmation

* If NULL value for any of the fields, then they will not be updated

enterPodcastEpisodeMediaInfo(mediaid, media_name, genre, language, m_country, episode_count, rating, sponsors, total_subscribers, episodeno, title, p_duration, p_release_date,listening_count,ad_count)

return confirmation

updatePodcastEpisodeMediaInfo(mediaid, media_name, genre, language, m_country, episode_count, rating, sponsors, total_subscribers, episodeno, title, p_duration, p_release_date,listening_count,ad_count)

return confirmation

* If NULL value for any of the fields, then they will not be updated

deletePodcastEpisodeMediaInfo(mediaid, media_name, genre, language, m_country, episode_count, rating, sponsors, total_subscribers, episodeno, title, p_duration, p_release_date,listening_count,ad_count)

return confirmation

* If NULL value for any of the fields, then they will not be updated

enterArtistCreatorsInfo(creatorsid, cf_name, cl_name, a_status, type, primary_genre, monthly_listeners, a_country)

return confirmation

updateArtistCreatorsInfo(creatorsid, cf_name, cl_name, a_status, type, primary_genre, monthly_listeners, a_country)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

deleteArtistCreatorsInfo(creatorsid, cf_name, cl_name, a_status, type, primary_genre, monthly_listeners, a_country)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

enterPodcastHostCreatorsInfo(creatorsid, cf_name, cl_name, email, phone, city)
    return confirmation

updatePodcastHostCreatorsInfo(creatorsid, cf_name, cl_name, email, phone, city)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

deletePodcastHostCreatorsInfo(creatorsid, cf_name, cl_name, email, phone, city)
    return confirmation
    * If NULL value for any of the fields, then they will not be updated

assignSongArtist(mediaid, creatorsid , albumid)
    return confirmation

assignArtist(creatorsid, labelname)
    return confirmation

assignPodcast(mediaid, episode_no, creatorsid)
    return confirmation

Maintaining metadata and records:

updateSongMediaInfo(mediaid ,play count)
    returns true if updated and false otherwise.
    * If NULL value for any of the fields, then they will not be updated

updateArtistCreatorsInfo(creatorsid, monthly_listeners)
    returns true if updated and false otherwise.
    * If NULL value for any of the fields, then they will not be updated

updatePodcastMediaInfo(mediaid, total_subscribers, rating)
returns true if updated and false otherwise.
* If NULL value for any of the fields, then they will not be updated

updatePodcastEpisodeMediaInfo(mediaid, episodeno, listening_count)
returns true if updated and false otherwise.
* If NULL value for any of the fields, then they will not be updated

findSongMediaInfo(mediaid, creatorsid , albumid)
returns mediaid if the song exists and false otherwise.

findPodcastEpisodeMediaInfo(mediaid, episodeno)
returns episodeno if the podcast episode exists and false otherwise.

Maintaining payments:

generateRoyalties(mediaid, play_count, royalty_rate)
returns monthly_royalties

maintainPayments(paymentid, pay_amount, pay_type, ps_date, pe_date, person_id, flat_fee, bonus, monthly_royalties)
Returns true if generated or false otherwise

maintainArtistPayment(creatorsid, apaymentid, as_date, ae_date, a_amount, label_name)
Returns true if generated or false otherwise

getPodcasthostPayment(creatorsid, paymentid, pay_amount, ps_date, pe_date, flat_fee, bonus)
Returns pay_amount or false otherwise

getUserPayment(uphone, start_date,end_date, sub_fee, u_status, paymentid, pay_amount)
Returns pay_amount or false otherwise

getRecordLabelPayment(paymentid, pay_amount, ps_date, pe_date, labelname)
Returns pay_amount or false otherwise

getArtistPayment(creatorsid, labelname, a_amount, apaymentid, as_date, ae_date)
Returns a_amount or false otherwise

Reports:

getMonthlyPlayCountSong(mediaid, play_count)
Returns monthly play_count

getMonthlyPlayCountAlbum(mediaid, play_count, albumid)
Returns monthly play_count

getMonthlyPlayCountArtist(mediaid, play_count, creatorsid)
Returns monthly play_count

getHostTotalPayments(paymentid, creatorsid, pay_amount, ps_start, pe_end)
Returns host total_payments

getArtistTotalPayments(labelname, creatorsid, a_amount,apaymentid,as_date,ae_date)
Returns artist total_payments

getLabelTotalPayments(paymentid, labelname, pay_amount, ps_start, pe_end)
Returns label total_payments

getTotalRevenue(paymentid, pay_amount, ps_start, pe_end)
Returns total revenue pay_amount

getSongs(mediaid, creatorsid, albumid)
Returns mediaid which is the song id.

getPodcastEpisode(mediaid, episodeno)
Returns episodeno which is the podcast episode.

6. Description of Views

- **Admin** - Has access to all information in the database for types of media - songs and albums, types of creators - artists and podcast hosts, users , podcast episodes, and podcasts. Keeps track of all the information to generate payments to the record label and the podcast host. Maintains information of the payment received from the user. Has access to information like monthly play count and total subscribers for songs and podcasts respectively.

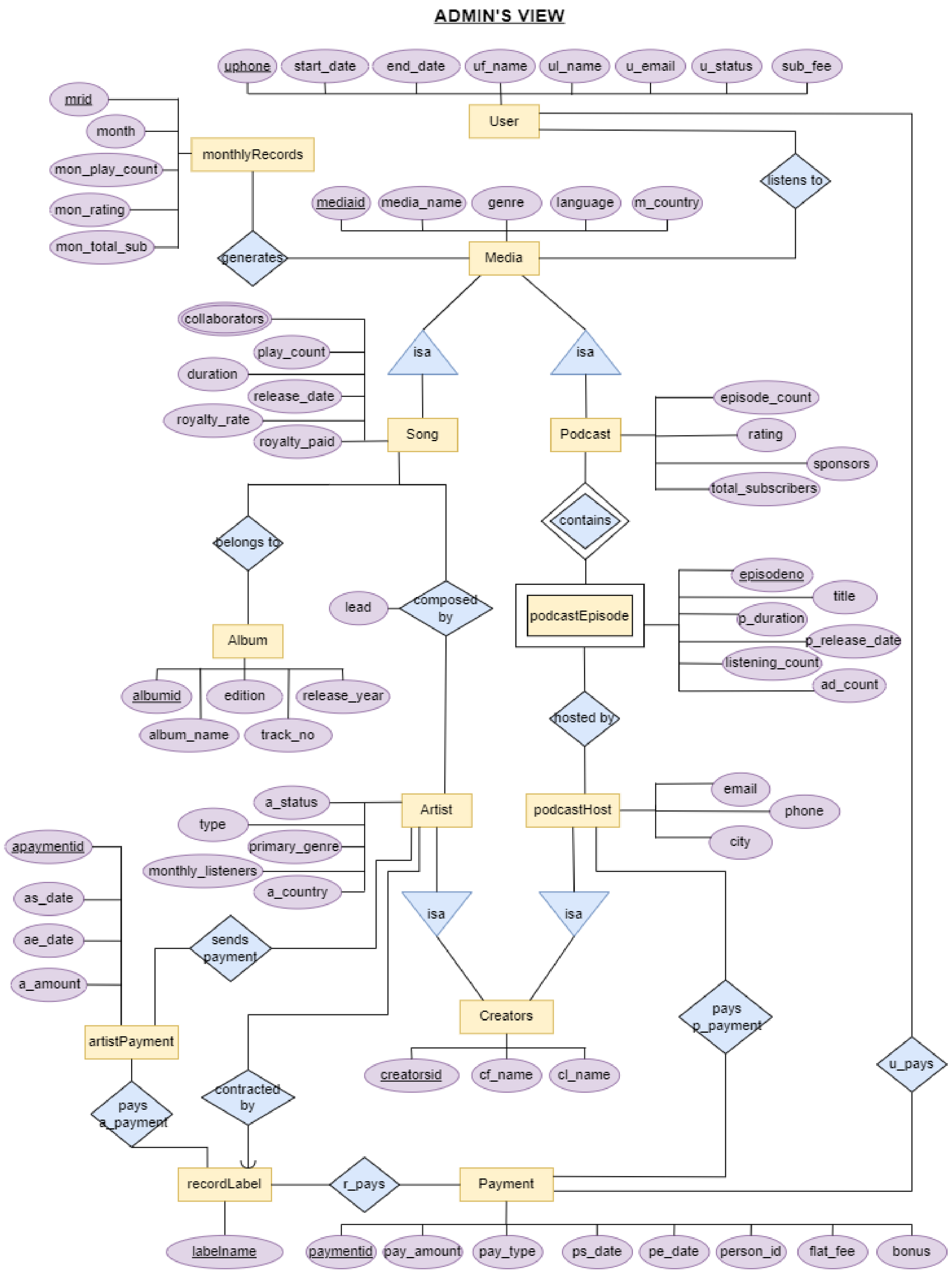
- **User** - Pays a monthly subscription fee to the media streaming service, subscribes to podcasts, and listens to songs which further influences the payments to be disbursed to the concerned - podcast hosts and artists(via record label). They do not have access to the payments being made to the creators.

- **Artist** - Get paid a part of royalties for the song they have composed by the record label. They create songs which feature in albums which are eventually owned by the record label. They need general information regarding the songs, artist, album and have no access to the podcast information.

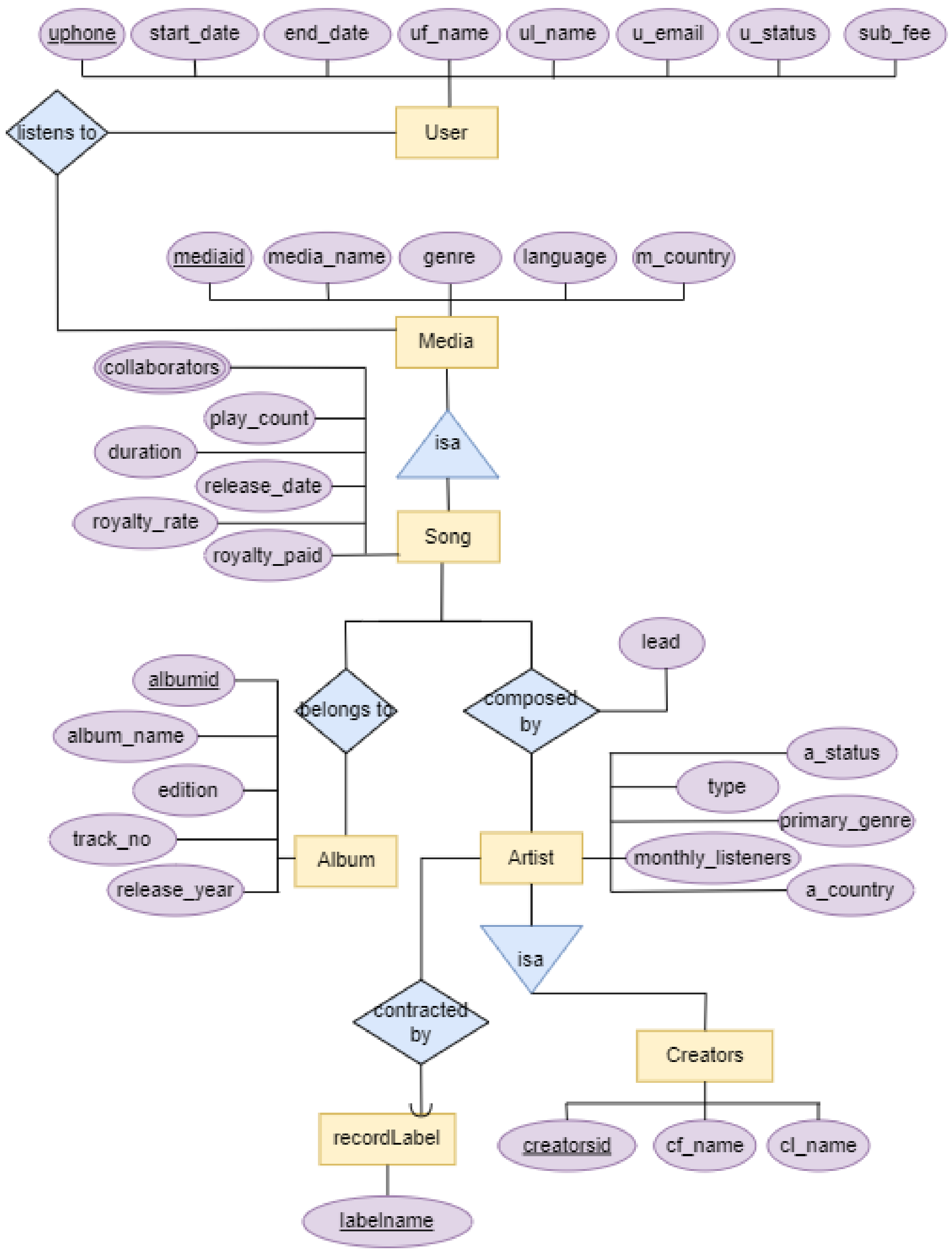
- **Podcast Host** - They receive fixed payments and additional bonus which will be based on the advertisements count for every episode they release from the media streaming service. A podcast can have special guests for any podcast episode. They do not need to access the information for songs, albums and artists which is handled by another set of users.

- **Record Label** - Generates the payment for the artist. Keep the tracks of its contracted artists and their songs. They do not have access to the information regarding the podcast.

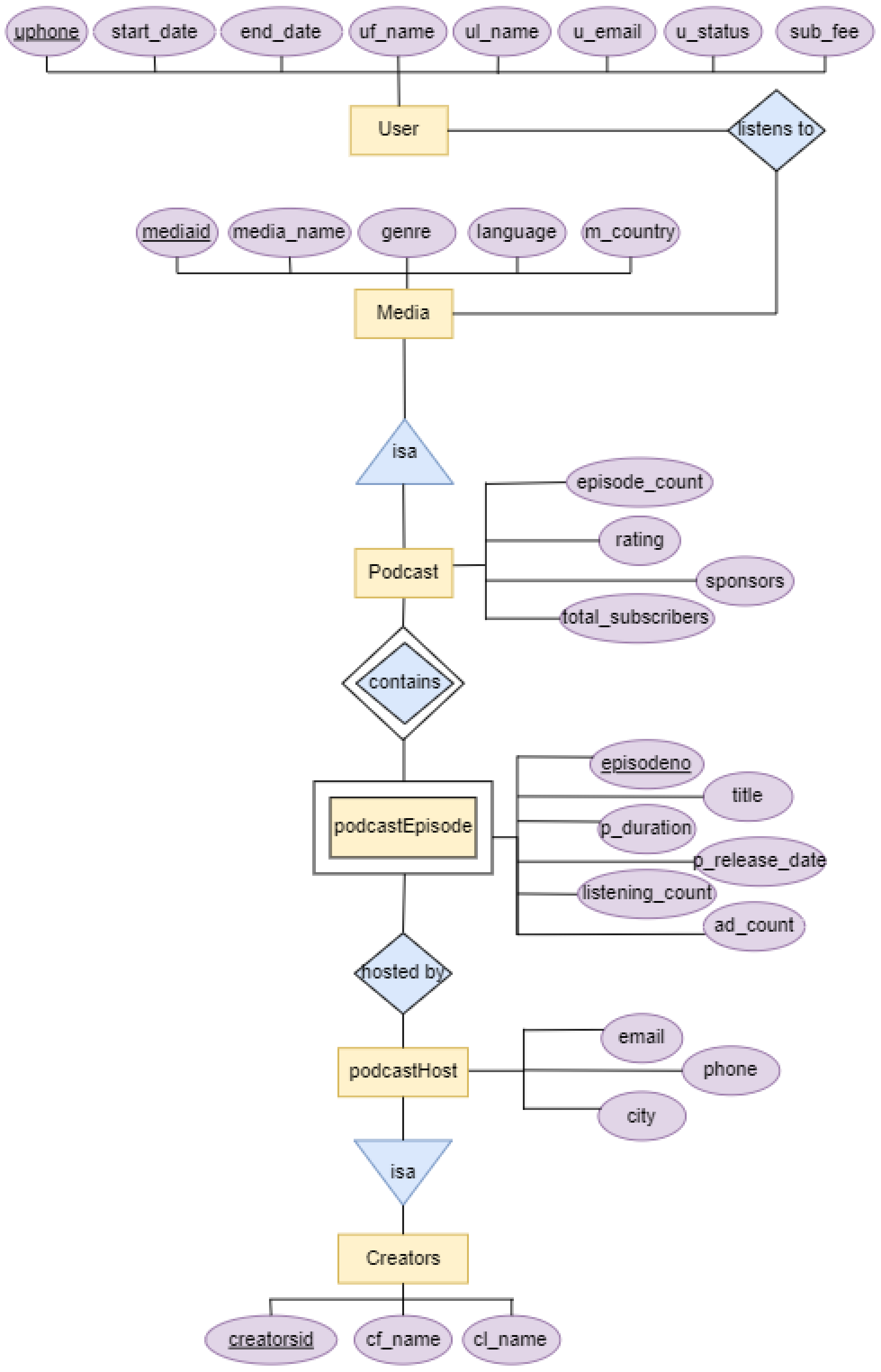
7. Local E/R Diagrams



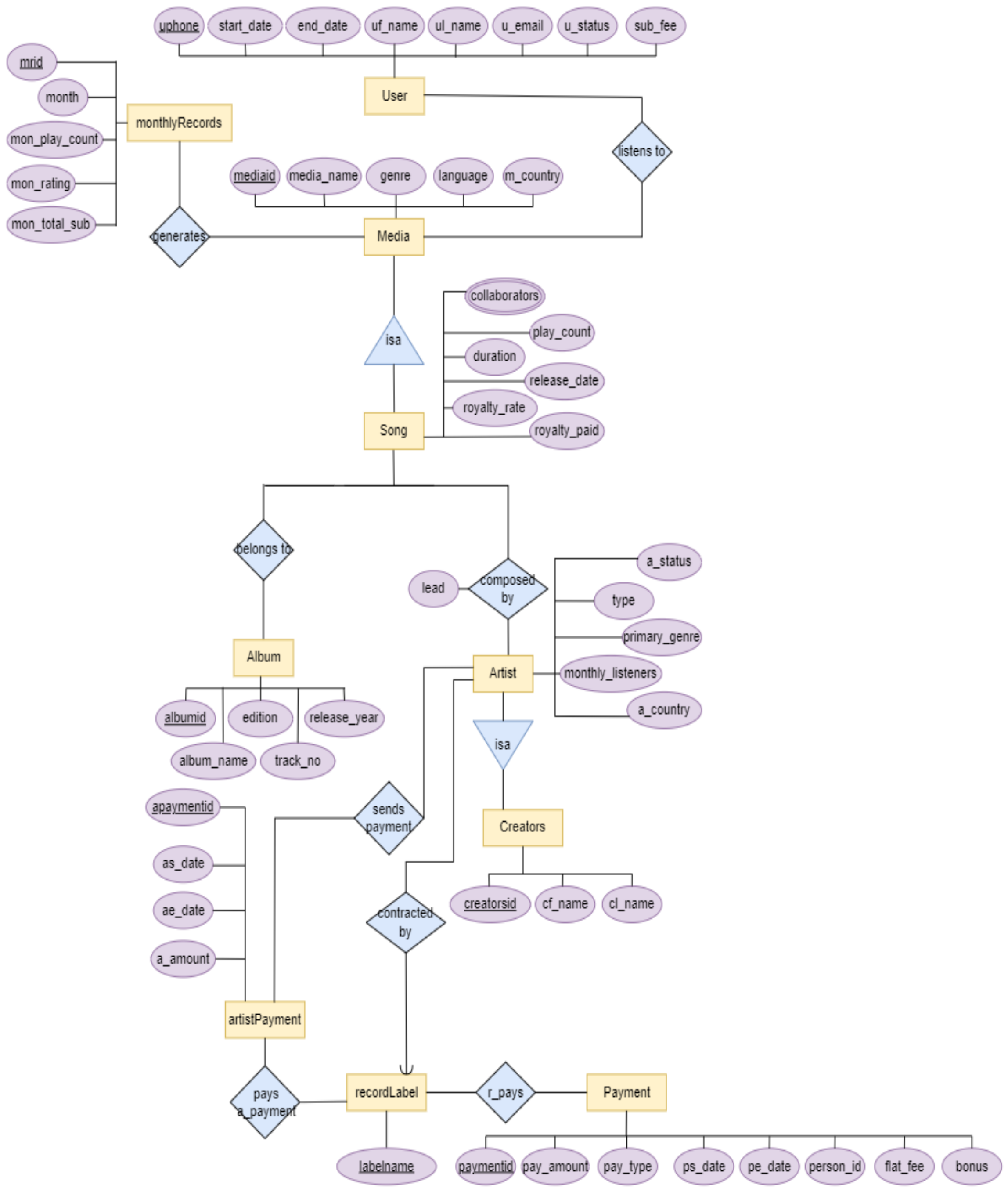
ARTIST'S VIEW



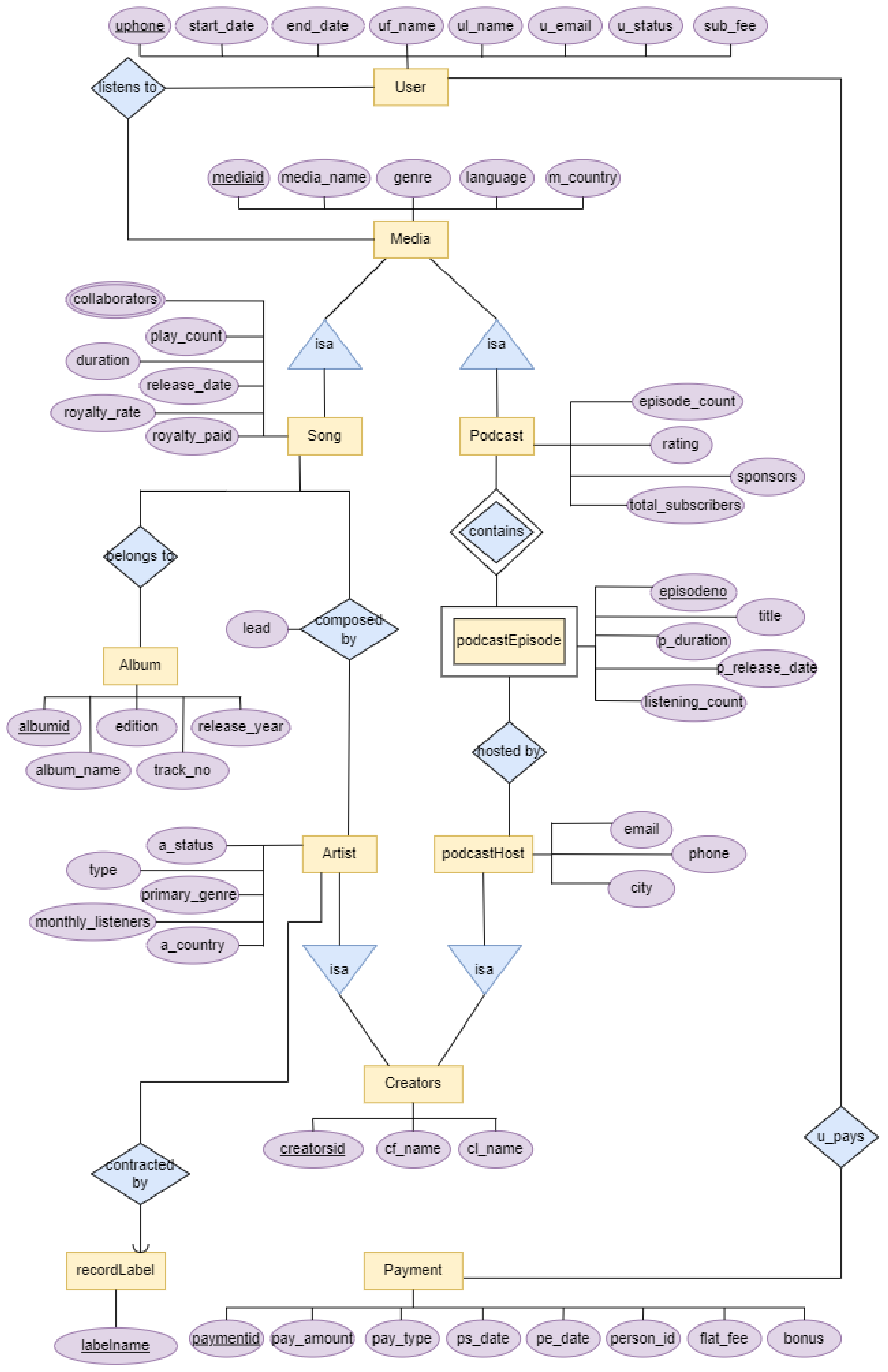
PODCAST HOST'S VIEW



RECORD LABEL'S VIEW



USER'S VIEW



uphone
start_date
end_date
uf_name
ul_name
u_email
u_status
sub_fee
listens to
User
mediaid
media_name
genre
language
m_country
Media
collaborators
play_count
duration
release_date
royalty_rate
royalty_paid
isa
isa
Song
Podcast
episode_count
rating
sponsors
total_subscribers
contains
belongs to
lead
composed by
podcastEpisode
episodeno
title
p_duration
p_release_date
listening_count
Album
albumid
edition
release_year
album_name
track_no
hosted by
a_status
type
primary_genre
monthly_listeners
a_country
Artist
podcastHost
email
phone
city
isa
isa
Creators
creatorsid
cf_name
cl_name
u_pays
contracted by
recordLabel
labelname
Payment
paymentid
pay_amount
pay_type
ps_date
pe_date
person_id
flat_fee
bonus

8. Description of Local E/R Diagrams:

- The media streaming service consists of artists, songs, albums, podcasts and listeners (User).
- The streaming service will have two types of Media - Song and Podcast.
- We have two types of Creators in the Streaming Service, namely - Artist and Podcast Host.
- A Song , Album and an Artist is uniquely identified by a mediaid, albumid and creatorsid respectively, since one can have similar names but will have different characteristics.
- We have assumed that every record label has its own unique name - labelname.
- Phone number is mandatory for a User, since it is a unique identifier. Even though user_email can be uniquely recognized, they aren't an accurate unique identifier.
- Podcast and Podcast Host have mediaid and creatorsid respectively as their unique key.
- Podcast Episode contain episodeno and mediaid as the keys because it is a weak entity set.
- Songs can be composed by more than one artist and it belongs to the album.
- Each Artist and the Podcast host is recognized uniquely with help of the creatorsid.
- Every Artist has a contract with only one record label. Each song is owned by the record label of its main artist.
- Each Podcast can contain multiple Podcast Episodes. Every Podcast Episode features a Podcast Host.
- Users will pay a monthly subscription fee(mon_sub_fee) to the Media Streaming Service.
- Media Streaming Service will generate the payment for the Record Label and Podcast Host.
- The Record Label will retain 30% of the received payment and distribute the remaining amongst the artists involved.
- Now, let's say there is more than one artist for a particular song, then the record label who contracts the main artist of the song shall receive this payment, since the song is owned by that record label.
- We recognize the main artist by the 'lead' attribute associated with the 'composed by' relation. Every artist will have a boolean value - 'lead'. If the value is 1 then that artist is the main artist and if the value is 0 then otherwise.
- Management collects all data related to Podcasts and Songs like play count, subscribers, ratings.
- They also keep record of all 3 types of payments made - received from users, payment made to record label and podcast host, in addition to payment from label to various artists.
- Media will generate monthlyRecords, which will store information like the month, monthly play count, monthly rating and monthly total subscribers for a particular song or podcast.

- Every Song and Podcast can be uniquely recognized by its mediaid.
- Podcast Episode is a weak entity, since each episode will be dependent on its podcast.

9. Local Relational Schemas

Admin's View:

User(uphone, start_date, end_date, uf_name, ul_name, u_email, u_status, sub_fee)
Media(mediaid, media_name, genre, language, m_country)
Song(mediaid, collaborators, play_count, duration, release_date, royalty_rate, royalty_paid)
Album(albumid, album_name, edition, track_no, release_year)
Podcast(mediaid, episode_count, rating, sponsors, total_subscribers)
podcastEpisode(mediaid, episodeno, title, p_duration, p_release_date, listening_count, ad_count)
Composedby(mediaid, creatorsid, lead)
Creators(creatorsid, cf_name, cl_name)
podcastHost(creatorsid, email, phone, city)
Artist(creatorsid, labelname, a_status, type, primary_genre, monthly_listeners, a_country)
monthlyRecords(mrid, month, mon_play_count, mon_rating, mon_total_sub)
artistPayment(apaymentid, as_date, ae_date, a_amount)
recordLabel(labelname)
Payment(paymentid, pay_amount, pay_type, ps_date, pe_date, person_id, flat_fee, bonus)
generates(mediaid, mrid)
belongsto(mediaid, albumid)
hostedby(mediaid, episodeno, creatorsid)
u_pays(uphone, paymentid)
paysp_payment(paymentid, creatorsid)
r_pays(paymentid, labelname)
paysa_payment(apaymentid, labelname)
sendspayment(apaymentid, creatorsid)

Record Label View:

User(uphone, start_date, end_date, uf_name, ul_name, u_email, u_status, sub_fee)
Media(mediaid, media_name, genre, language, m_country)
monthlyRecords(mrid, month, mon_play_count, mon_rating, mon_total_sub)

generates(mediaid, mrid)
Song(mediaid, collaborators, play_count, duration, release_date, royalty_rate, royalty_paid)
belongsto(mediaid, albumid)
Album(albumid, album_name, edition, track_no, release_year)
Composedby(mediaid, creatorsid, lead)
Artist(creatorsid, labelname, a_status, type, primary_genre, monthly_listeners, a_country)
Creators(creatorsid, cf_name, cl_name)
sendspayment(apaymentid, creatorsid)
artistPayment(apaymentid, as_date, ae_date, a_amount)
paysa_payment(apaymentid, labelname)
recordLabel(labelname)
r_pays(paymentid, labelname)
Payment(paymentid, pay_amount, pay_type, ps_date, pe_date, person_id, flat_fee, bonus)

Podcast Host View:

User(uphone, start_date, end_date, uf_name, ul_name, u_email, u_status, sub_fee)
Media(mediaid, media_name, genre, language, m_country)
Podcast(mediaid, episode_count, rating, sponsors, total_subscribers)
podcastEpisode(mediaid, episodeno, title, p_duration, p_release_date, listening_count, ad_count)
hostedby(mediaid, episodeno, creatorsid)
podcastHost(creatorsid, email, phone, city)
Creators(creatorsid, cf_name, cl_name)

Users view:

User(uphone, start_date, end_date, uf_name, ul_name, u_email, u_status, sub_fee)
Media(mediaid, media_name, genre, language, m_country)
Song(mediaid, collaborators, play_count, duration, release_date, royalty_rate, royalty_paid)
Podcast(mediaid, episode_count, rating, sponsors, total_subscribers)
podcastEpisode(mediaid, episodeno, title, p_duration, p_release_date, listening_count, ad_count)
Composedby(mediaid, creatorsid, lead)
belongsto(mediaid, albumid)

Album(albumid, album_name, edition, track_no, release_year)
Artist(creatorsid, labelname, a_status, type, primary_genre, monthly_listeners, a_country)
podcastHost(creatorsid, email, phone, city)
Creators(creatorsid, cf_name, cl_name)
recordLabel(labelname)
Payment(paymentid, pay_amount, pay_type, ps_date, pe_date, person_id, flat_fee, bonus)
u_pays(uphone, paymentid)

Artist View:

User(uphone, start_date, end_date, uf_name, ul_name, u_email, u_status, sub_fee)
Media(mediaid, media_name, genre, language, m_country)
Song(mediaid, collaborators, play_count, duration, release_date, royalty_rate, royalty_paid)
belongsto(mediaid, albumid)
Album(albumid, album_name, edition, track_no, release_year)
Composedby(mediaid, creatorsid, lead)
Artist(creatorsid, labelname, a_status, type, primary_genre, monthly_listeners, a_country)
Creators(creatorsid, cf_name, cl_name)
recordLabel(labelname)

10. Local Schema Documentation

Entity Sets to Relations:

- The entity sets in our diagram were made into relations, with the attributes the same for User, media, creators, artistPayment, recordLabel and Payment.
- The Entity sets that are subsets of media and creators were made into relations based on the isa hierarchy. This helped to avoid the problem of redundancy and thus was useful in saving a lot of space in the table.
- The podcastEpisode is taken as a weak entity as it cannot be uniquely identified and it is dependent on Podcast.
- In order to store payment information for users, record labels, and podcast hosts, a single Payment entity has been created which is eventually converted into a relation.
- By doing so, a considerable amount of space was saved as a single Payment table was created instead of multiple tables for different payments.

Combining Many-One Relationships:

- While combining many-one relationships to relational schema some of the relationships were not converted into tables which made the queries easier and avoided the problem of attributes being repeated in various tables.
- Weak entity set podcastEpisode was made into a schema with all of its attributes plus the uniquely identified attributes inherited from the strong entity set Podcast which was connected by the relation 'contains'.
- PodcastEpisode will have mediaid as a foreign key in addition to all its attributes to uniquely identify itself as episode number can be the same in every podcast.
- A supporting (double - diamond) relationship 'contains' is redundant, and thus, yields no relation.
- Entity set 'Artist' with the relation 'contracted by' for a many-to-one relationship from artist to record label is combined, so instead of making Artist (<u>creatorsid</u>, a_status, type, primary_genre, monthly_listeners, a_country) and contractedby (<u>creatorsid</u>, <u>labelname</u>) are combined to make Artist (<u>creatorsid</u>, a_status, type, primary_genre, monthly_listeners, a_country, <u>labelname</u>).

Relationships to Relations:

- Relationships generates, composedby, hostedby, sendspayment, r_pays and u_pays from the ER Diagrams are turned into relations in our schema. Their attributes in the schema consist of key attributes of the entities they are connected with.
- Relationship 'listensto' is only used for visualization purposes and does not have a direct impact on data. Thus, there is no need to include it in the schema.
- Relationship 'contactedby' is not converted into the table because all its attributes are present in the Artist.
- Relationship 'contractedby' contains the attribute lead which is a boolean value. So while converting it into relational schema the value will be True only for its main artist and the other artist will have its value as False.