System Requirements Document for Podcast Hosting Service

Source of research:

https://www.geeksforgeeks.org/dbms/database-design-in-dbms/

Critical Points for design:

1. Data consistency and integrity
2. Low redundancy
3. Faster searching through indices
4. Security measures
5. Data should be stored in fragmented bits of information in the most atomic format possible

Steps for design:

1. Determine the goal of the DB – The goal of this database is to provide a design for a podcast hosting service in which podcasters can publish episodes of their podcasts to listeners who can then subscribe to the podcast.
2. List all entities that will be present in the DB (This is further down the document)
3. Organize the info in different tables so that there is little to no redundancy
4. After all tables are structured, use normalization to further reduce redundancy and ensure consistency.

Usual cycle for database design:

1. Req analysis – Understand requirements
2. Logical and physical design – The phase that contains actually designing the database, identifying entities, attributes, data types and relationships. Split into the logical data model design and physical design phases.
3. Implementation & testing

Final Report:

Section 1: Summary

This database project serves the purpose of providing a design as well as some sample data for a database that can be used for a podcast hosting service. The purpose of this database is to efficiently store data regarding listeners, hosts, and the episodes of their podcasts.

Section 2: Intro

The main problem with storing data for a podcasting service is that there is a lot of data to be processed. While this database design does not account for the actual data types regarding audio files, it does account for important information about the podcast and their episodes, including season and episode numbers, as well as duration and even the language of each podcast.

Section 3: Requirements Analysis

Users are going to either be a host or a listener. As a listener, they just need to access the episodes of podcasts created by hosts; however, they are not allowed to edit these episodes. As a host, they get to access the same things as the listeners but can edit the episodes as well as different things about the podcast like the name or maybe the description.

Section 4: Database design

Now that we know what we need to design our database and what kinds of considerations to make when doing so, I will begin to identify the aspects of my database.

Entities (attributes):

Hosts (hostID(PK), hostName, bio)

Listeners (listenerID(PK), username, email, dateJoined)

Podcasts (podcastID(PK), title, desc., genre, language)

Episodes (episodeID(PK), title, episodeNum, seasonNum, dateCreated, duration)

SUBSCRIBE (listenerID(FK), podcastID(FK), dateSubscribed)

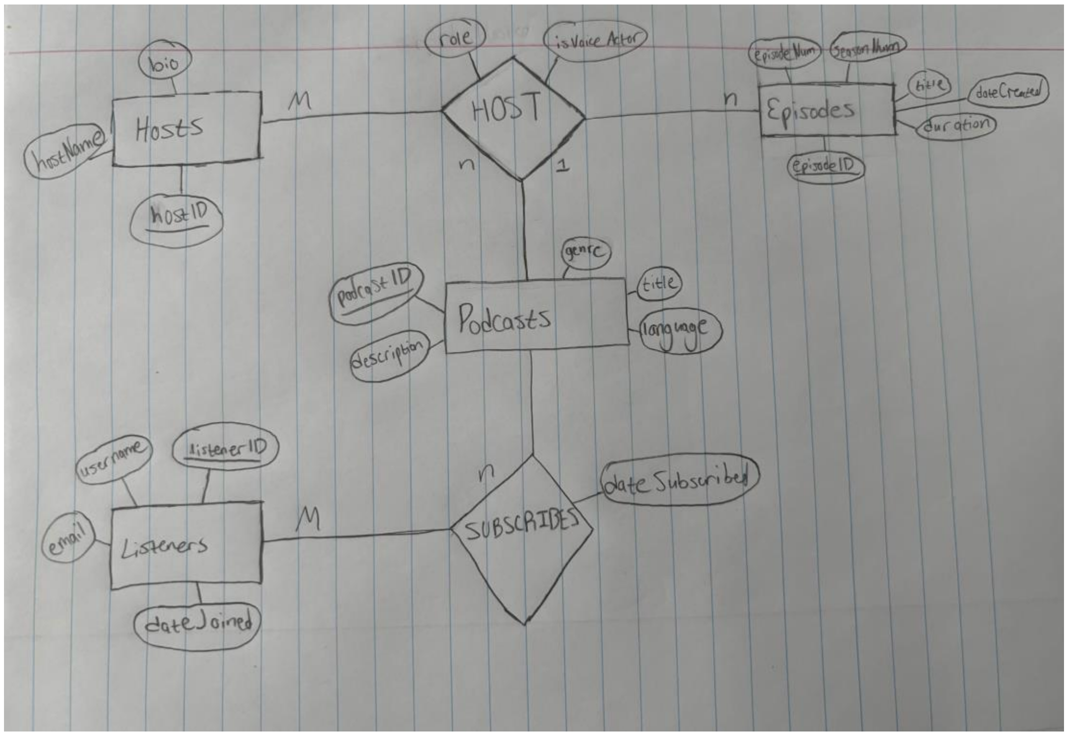
HOST (hostID(FK), episodeID(FK), podcastID(FK), role, isVoiceActor)

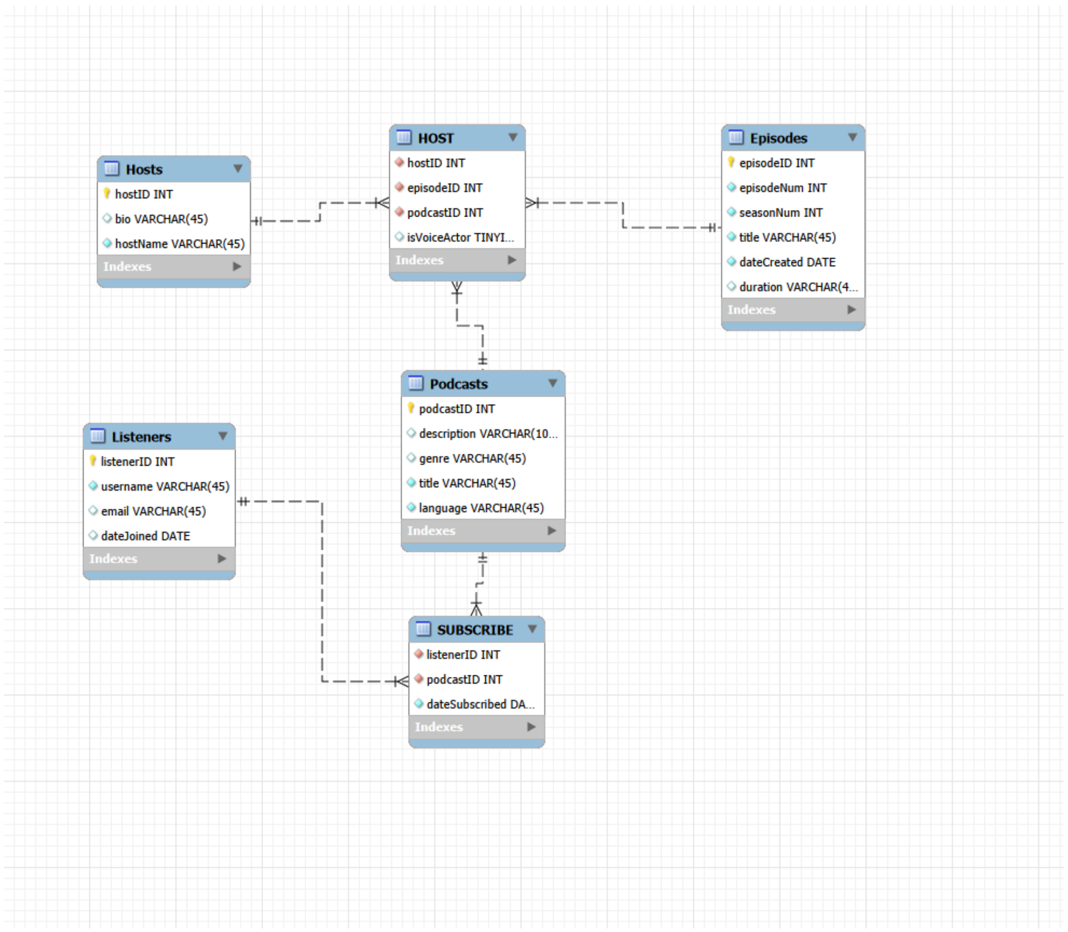
Relationships:

Hosts HOST episodes for podcasts (hosts can have more than one podcast, so this relationship will be many to many, same for episodes)

Listeners SUBSCRIBE to podcasts (Many listeners can subscribe to many podcasts, so this will be a M:N relationship as well)

ER Diagrams (Chen and UML format)





Section 5: Implementation

The SQL script files are provided in the github repository (project1.sql), the script contains the forwarding script used to create the tables from the initial model (as shown in the UML diagram) in addition to the queries run to populate the tables.

Section 6: Testing and results

After running tests with various select queries, we can see the data is stored properly and can be retrieved with no problem. These queries are included in the script as well. If you would like to test it yourself, feel free to use them as I did in testing.

Section 7: Conclusion

This was a fun project to take on and design, and I am glad to have learned a lot about the nitty gritty when it comes to this sort of thing. While there are things that I could have included (audio files, entities and stricter constraints), I feel it was a bit out of reach given the scope of the project and serves just fine as a foundation that could be built upon to further improve efficiency and include data that could be useful from a business perspective.