Data and Applications

Group Project Phase - 1

Team AGDP

Harshita Gupta 2020101078 | Parth Maradia 2020111006 | Prerak Srivastava 2020111013

Miniworld - **SYMPHONY** (A music streaming platform)

Introduction

Symphony is a music streaming media application that allows users to access and listen to their favorite songs and albums from singers and artists from all over the world.

Purpose

Allows users to search and listen to music that can be browsed through or searched for using parameters such as artist, album, or playlist.

Users

The users of the Symphony's database would be:

- Artists and singers who have their songs accessible on Symphony
- Listeners/Audience/Average user who can use the Symphony database to listen to their favorite songs
- Software developers that can gather data about a particular song or an artist.

Applications

The applications of the Symphony database would be:

- The Symphony database would store millions of songs across different genres and styles as well as user data.
- It would allow the application to offer categories for users to find the music that they want to listen to quickly and efficiently.

Database Requirements

Entities

1. **User**
    a. User_Id
        i. [Primary Key]
        ii. [Integer > 0]
        iii. [Not NULL]
    b. User_Name
        i. [Varchar(30)]
        ii. [Not NULL]
    c. User_Email
        i. [Candidate Key]
        ii. [Varchar(30)] [Not NULL]
    d. User_Password
        i. Varchar
        ii. Not NULL]
        iii. Restriction - minimum 15 characters long
    e. User_Registration_Date
        i. Datetime
    f. User_State
        i. Varchar(15)
    g. User_Country
        i. Varchar(15)
    h. User_Location
        i. Varchar(30)
        ii. composite - User_State + User_Country
    i. User_Country_Code
        i. Varchar (4)
        ii. derived - User_Country
    j. User_Mobile_Number
        i. Varchar(10)
    k. User_Contact
        i. Varchar (15)
        ii. composite - User_Country_Code + User_Mobile_Number
    l. User_Img
        i. Varchar

Subclasses (Disjoint)

1. Premium Users
    a. Additional attributes
        i. Plan_Id [int] [not NULL] [Foreign Key]
2. Normal Users

2. Song

a. Song_Id
    i. Primary Key
    ii. Integer>0
    iii. Not NULL
b. Song_Title
    i. Varchar
c. Song_Length
    i. Datetime
d. Song_Lyrics
    i. Varchar
e. Song_Genre
    i. Varchar
    ii. Multivalued
f. Is_Explicit
    i. Boolean
    ii. Derived
g. Date_Added
    i. Datetime
h. Song_Path
    i. Varchar
i. Song_Cover_Img
    i. Varchar

3. Artist

a. Artist_Id
    i. Primary Key
    ii. Integer>0
    iii. Not NULL
b. Artist_Name
    i. Varchar
c. Artist_Followers
    i. int
d. Artist_Category

        i. Varchar
    e. Artist_Img
        i. Varchar

4. Album

- a. Album_Id
    - i. Primary Key
    - ii. Integer>0
    - iii. Not NULL
- b. Album_Name
    - i. Varchar
- c. Album_Release_Date
    - i. Datetime
- d. Album_Cover_Img
    - i. Varchar

5. Podcast

- a. Podcast_ID
    - i. Primary Key
    - ii. Integer>0
    - iii. Not NULL
- b. Podcast_Title
    - i. Varchar
- c. Podcast_Category
    - i. Varchar
- d. Podcast_Number_of_Episodes
    - i. Int
    - ii. Not NULL
- e. Podcast_Total_Duration
    - i. Datetime
- f. Podcast_Language
    - i. Varchar
    - ii. Multi-valued
- g. Podcast_Cover_Img
    - i. varchar

6. Plans

- a. Plan_Id

i. int > 0
ii. Primary key
iii. not NULL

b. Plan_Name
i. varchar(20)

c. Plan_Cost
i. per month
ii. int

Weak Entities

1. **Playlist**
    a. Playlist_Title [Varchar]
    b. Playlist_Description [Varchar]
    c. Playlist_Saves [int]
    d. Is_Secret [boolean]

2. **Episode**
    a. Episode_Title [Varchar]
    b. Episode_Number [Int]
    c. Episode_Duration [Datetime]
    d. Episode_Release_Date [Datetime]
    e. Episode_Description [Varchar]

Relationships

1. “Album” **has** “Song” **by** “Artist”
    a. Attributes
        i. Album_Id
        ii. Song_Id
        iii. Artist_Id
    b. Cardinality Constraint:
        i. An album can have multiple songs but a song is there in a single album.
        ii. An album should have at least 2 songs.
        iii. An album can be created by multiple artists. (many to many)
        iv. An artist can have multiple albums.
    c. Ratio:
        i. Album : Song - One to many

    ii. Album : Artist - Many to many
    iii. Song : Artist - Many to one
d. Participation Constraints:
    i. Total Album
    ii. Partial Artist
    iii. Partial Song
e. Degree = 3

2. “User” **follows** “Artist”
    a. Attributes
        i. User_Id
        ii. Artist_Id
    b. Cardinality Constraint: A user can follow multiple artists and an artist can have multiple followers.
    c. Ratio: many to many
    d. Participation Constraint
        i. Partial User
        ii. Partial Artist
    e. Degree = 2

3. “User” **likes** “Song”
    a. Attributes
        i. Song_Id
        ii. User_Id
    b. Cardinality Constraint: A user can like multiply songs and a song can be liked by multiple people.
    c. Ratio: many to many
    d. Participation Constraint
        i. Partial User
        ii. Partial Song
    e. Degree = 2

4. “User” **creates** “Playlist”
    a. Attributes
        i. User_Id
    b. Cardinality Constraint: A user can create multiple playlists but a playlist has a single creator.
    c. Ratio: One to many
    d. Participation Constraints
        i. Partial User

ii. Total Playlist

e. Degree = 2

5. "User" **saves** "Playlist"
    a. Attributes
        i. User_Id
    b. Cardinality Constraint: A user can save multiple playlists and a playlist can be saved by multiple users.
    c. Ratio: Many to Many.
    d. Participation Constraints
        i. Partial User
        ii. Partial Playlist
    e. Degree = 2

6. "Playlist" **has** "Song" **from** "Album" **by** "Artist"
    a. Attributes
        i. Album_Id
        ii. Song_Id
        iii. Artist_Id
    b. Cardinality Constraint:
        i. Playlist should have atleast one song.
        ii. An album can have multiple songs but a song is there in a single album.
        iii. An album should have at least 2 songs.
        iv. An album can be created by multiple artists. (many to many)
        v. An artist can have multiple albums.

    c. Ratio:
        i. Playlist : Song - Many to many
        ii. Playlist : Album - Many to many
        iii. Playlist : Artist - Many to many
        iv. Album : Song - One to many
        v. Album : Artist - Many to many
        vi. Song : Artist - Many to one

    d. Degree = 4

7. "Artist" **compose** "Song" (For singles - not in any album)
    a. Attributes
        i. Song_Id

    ii. Artist_Id
b. Cardinality Constraint: An artist can compose multiple songs.
c. Ratio: One to many.
d. Participation Constraints
    i. Partial Artist
    ii. Total Song
e. Degree = 2

8. “Artist” **creates** “Podcast”
    a. Attributes
        i. Artist_Id
        ii. Podcast_Id
    b. Cardinality Constraint: An artist can create multiple podcasts but a podcast has a single creator.
    c. Ratio: One to many
    d. Participation Constraints
        i. Partial Artist
        ii. Total Podcast
    e. Degree = 2

9. “Episode” **of** “Podcast”
    a. Attributes
        i. Podcast_Id
    b. Cardinality Constraint: A podcast can have multiple episodes but an episode can be a part of only one podcast.
    c. Ratio: Many to one.
    d. Participation Constraints
        i. Total Episode
        ii. Total Podcast
    e. Degree = 2

Functional Requirements

Modifications

1. Insert:
    a. Insert a new user record in the User entity. (restrict user from making multiple accounts with the same email (primary key in User entity))
    b. Insert a new song in the Song entity.

2. Delete:
    a. When a user deletes his/her account, delete his/her record from the User entity. (relationships from that user record to other entities(playlists, songs etc.) would also be deleted).

3. Update:
    a. When a user follows a new artist - update Artist_followers attribute in Artist entity.
    b. When a new episode for a podcast releases - update Podcast_Total_Duration and Podcast_Number_Of_Episodes.
    c. When a user saves/likes a new playlist - update Playlist_Saves in Playlist Entity.

Retrievals

1. Selection:
    a. Select all podcast records with Podcast_Duration > 60 mins.
    b. Select all artist records with more than 1 lakh followers.

2. Projection:
    a. Select all song names with genre “Folk” .
    b. Select all album Ids and names by a particular artist.

3. Aggregate:
    a. Artist name with maximum number of followers.
    b. Details of playlist with maximum saves.
    c. Details of podcast with minimum duration.

4. Search:
    a. Search songs by a given keyword (artist name, song name, album name).
    b. Search podcasts in a given language.

    c. Search premium users with a given plan.

5. Analysis:
    a. List of popular albums based on average number of likes per song.
    b. Top genres for a particular user based on his/her liked songs.