Ministry of Education
Imam Abdulrahman bin Faisal University
Computer Science department

CIS 321: Databases Concept & Design Report

Audio Vibe

Students name	ID
Lojain Hafez Alibrahim	2220001991
Shatha Ahmed Alanazi	2220006290
Shahad Mohammed Alowfi	2220002692
Rahaf Adnan Alshammari	2220000977
Noura Saad Alqahtani	2220001667
Amal Ali Alqahtani	2220003347

Instructors:

Dr. Thowiba Awad - L. Sameerah Albalhareth - Dr. Mona Alghamdi

Academic year 1445 – 2023/2024

Introduction:

Introducing the Podcast (Audio Vibe) Management System, a comprehensive solution for navigating the expansive realm of podcasts. With meticulous cataloging of podcast details such as titles, hosts, publication dates, and episode counts, this system empowers users, from listeners to administrators, to effortlessly discover, organize, and enjoy their preferred audio content. Through personalized interest surveys, the system curates recommendations, ensuring tailored content discovery for each user. Additionally, it offers unique features like playlist creation, category modification, and user management, enhancing the overall podcasting experience for all users.

Constraints and keys:

```
CREATE TABLE `user_type` (
  `type_id` int NOT NULL,
  `type_desc` varchar(100) NOT NULL,
  PRIMARY KEY (`type_id`)
);
```

```
CREATE TABLE `users` (
  `User_id` int NOT NULL,
  `Username` varchar(100) NOT NULL,
  `Password` varchar(100) NOT NULL,
  `regdate` varchar(45) DEFAULT NULL,
  `phone` varchar(45) DEFAULT NULL,
  `Email` varchar(100) DEFAULT NULL,
  `loginstatus` int NOT NULL,
  `user_type` int NOT NULL,
  PRIMARY KEY (`User_id`),
  CONSTRAINT `user_types` FOREIGN KEY (`user_type`) REFERENCES `user_type` (`type_id`)
);
```

```
CREATE TABLE `categories` (
  `cat_id` int NOT NULL AUTO_INCREMENT,
  `cat_name` varchar(100) NOT NULL,
  PRIMARY KEY (`cat_id`)
);
```

```
CREATE TABLE `end_user` (
  `name` varchar(100) NOT NULL,
  `address` varchar(100) DEFAULT NULL,
  `DOB` varchar(45) DEFAULT NULL,
  `POB` varchar(100) DEFAULT NULL,
  `Gender` char(1) DEFAULT NULL,
  `Nat_id` int NOT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`user_id`),
  CONSTRAINT `nat_user` FOREIGN KEY (`Nat_id`) REFERENCES `nationality` (`nat_id`),
  CONSTRAINT `user_Users` FOREIGN KEY (`user_id`) REFERENCES `users` (`User_id`)
);
```

```
CREATE TABLE `randomqas` (
  `episodesno` int NOT NULL,
  `podcastprogramno` int DEFAULT NULL,
  `chooseintrest` varchar(100) DEFAULT NULL,
  `listenORwatch` varchar(100) DEFAULT NULL,
  `nonArabic` varchar(100) DEFAULT NULL,
  `User_id` int NOT NULL,
  PRIMARY KEY (`User_id`),
  CONSTRAINT `user_id_QAs` FOREIGN KEY (`User_id`) REFERENCES `end_user` (`user_id`
);
```

```
CREATE TABLE `playlist` (
  `PL_id` int NOT NULL,
  `PL_name` varchar(100) NOT NULL,
  `isdeleted` int NOT NULL DEFAULT '0',
  `cat_id` int NOT NULL,
  `creation_date` varchar(45) NOT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`PL_id`,`user_id`),
  CONSTRAINT `pl_cat` FOREIGN KEY (`cat_id`) REFERENCES `categories` (`cat_id`),
  CONSTRAINT `pl_users` FOREIGN KEY (`user_id`) REFERENCES `end_user` (`user_id`)
```

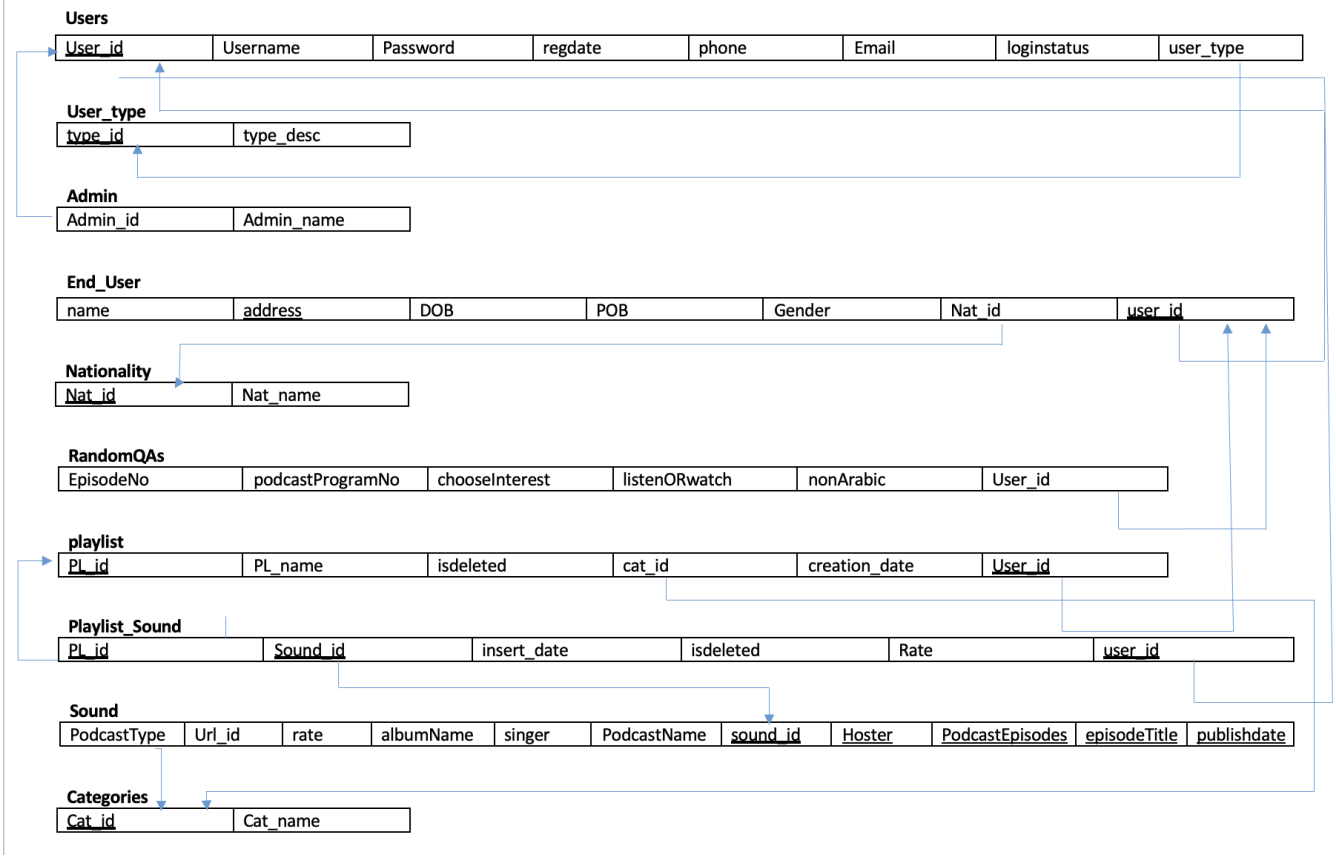
```
CREATE TABLE `sound` (
  `PodcastType` int NOT NULL,
  `Url_id` varchar(200) DEFAULT NULL,
  `rate` int DEFAULT NULL,
  `albumName` varchar(100) DEFAULT NULL,
  `singer` varchar(100) DEFAULT NULL,
  `PodcastName` varchar(100) DEFAULT NULL,
  `sound_id` int NOT NULL,
  `Hoster` varchar(100) DEFAULT NULL,
  `PodcastEpisodes` int DEFAULT NULL,
  `episodeTitle` varchar(100) DEFAULT NULL,
  `publishdate` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`sound_id`),
  CONSTRAINT `soundcat` FOREIGN KEY (`PodcastType`) REFERENCES `categories` (`cat_id`
);
```

```
CREATE TABLE `nationality` (
  `nat_id` int NOT NULL AUTO_INCREMENT,
  `nat_name` varchar(100) NOT NULL,
  PRIMARY KEY (`nat_id`)
);
```

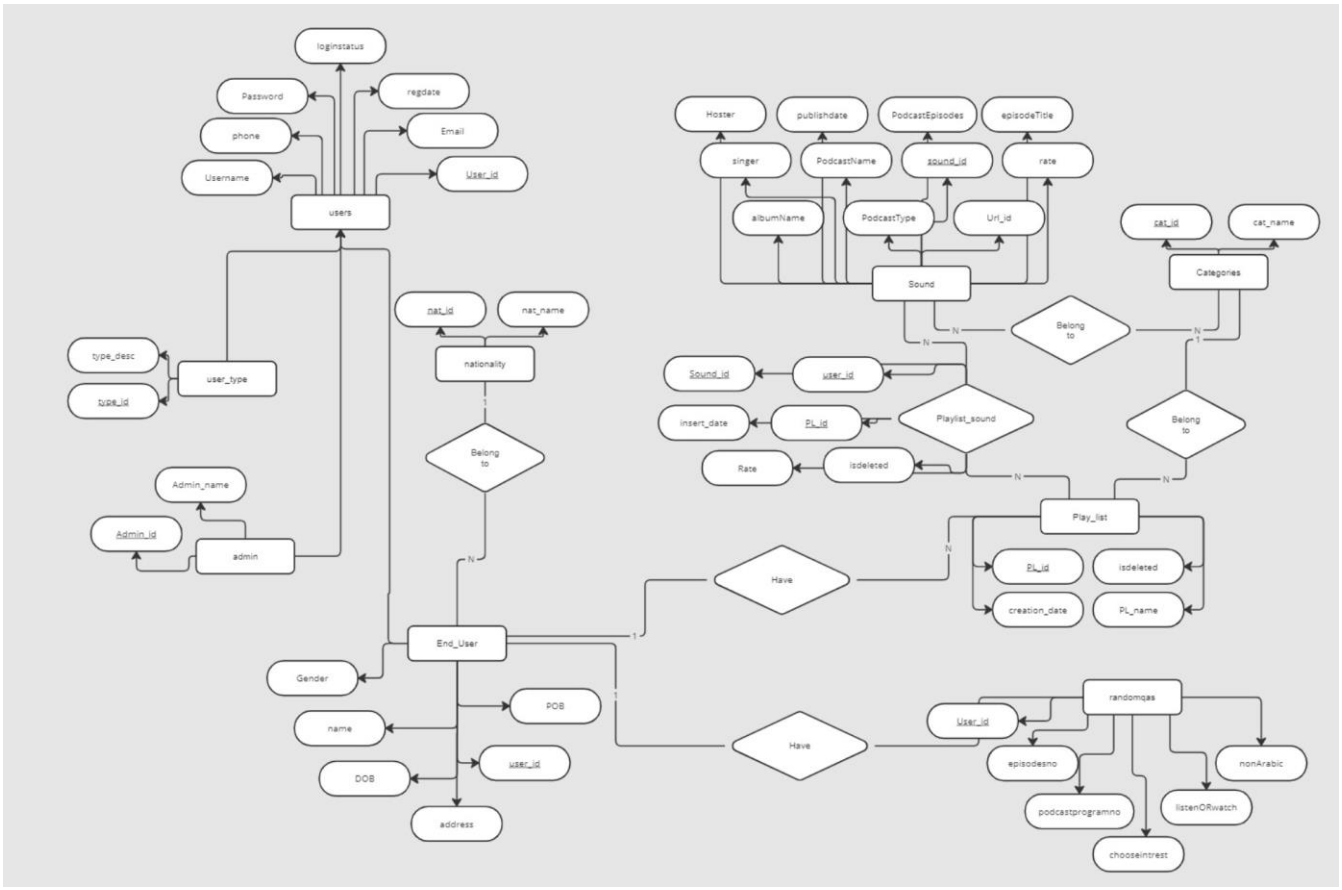
```
CREATE TABLE `playlist_sound` (
  `PL_id` int NOT NULL,
  `sound_id` int NOT NULL,
  `insert_date` varchar(45) NOT NULL,
  `isdeleted` int DEFAULT '0',
  `Rate` int DEFAULT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`PL_id`,`sound_id`,`user_id`),
  CONSTRAINT `pl_sound` FOREIGN KEY (`PL_id`) REFERENCES `playlist` (`PL_id`),
  CONSTRAINT `sound_pl` FOREIGN KEY (`sound_id`) REFERENCES `sound` (`sound_id`),
  CONSTRAINT `user_plylst` FOREIGN KEY (`user_id`) REFERENCES `users` (`User_id`)
);
```

```
CREATE TABLE `admin` (
  `Admin_id` int NOT NULL,
  `Admin_name` varchar(100) NOT NULL,
  PRIMARY KEY (`Admin_id`),
  CONSTRAINT `Admin_users` FOREIGN KEY (`Admin_id`) REFERENCES `users` (`User_id`
);
```

ERD:


Users
User_id | Username | Password | regdate | phone | Email | loginstatus | user_type
User_type
type_id | type_desc
Admin
Admin_id | Admin_name
End_User
name | address | DOB | POB | Gender | Nat_id | user_id
Nationality
Nat_id | Nat_name
RandomQAs
EpisodeNo | podcastProgramNo | chooseInterest | listenORwatch | nonArabic | User_id
playlist
PL_id | PL_name | isdeleted | cat_id | creation_date | User_id
Playlist_Sound
PL_id | Sound_id | insert_date | isdeleted | Rate | user_id
Sound
PodcastType | Url_id | rate | albumName | singer | PodcastName | sound_id | Hoster | PodcastEpisodes | episodeTitle | publishdate
Categories
Cat_id | Cat_name

Mapping:


loginstatus
Password
regdate
phone
Email
Username
User_id
users
Hoster
publishdate
PodcastEpisodes
episodeTitle
singer
PodcastName
sound_id
rate
albumName
PodcastType
Url_id
Sound
cat_id
cat_name
Categories
nat_id
nat_name
nationality
Belong to
type_desc
user_type
type_id
Sound_id
user_id
insert_date
PL_id
Playlist_sound
Rate
isdeleted
Admin_name
Admin_id
admin
Play_list
Have
End_User
Gender
POB
name
user_id
DOB
address
randomqas
episodesno
nonArabic
podcastprogramno
listenORwatch
chooseintrest

Description:
The Database Podcast Management System maintains track of each detail of podcasts, including their name, host, publication date, and number of episodes. Based on a brief interest based survey, the method makes it easier to look for and explore the most popular podcast. Whether a user is a listener or an administrator, this system is meant to assist them with unique features including playlist organization, category modification, and user creation.

Scenario:
People can learn through various means, and one particularly effective avenue is through listening, encompassing both boring and fascinating subjects alike. The world of podcasts opens up a seamless opportunity to explore and engage with captivating episodes on virtually any topic. In our podcast application, we strive to set ourselves apart by offering innovative features that go beyond what's commonly found in well-known platforms. From an administrative standpoint, we've set specific goals to enhance the user experience. This includes content management, ensuring a constant influx of diverse and captivating podcast content. We also recognize the importance of efficient deletion, giving administrators the power to permanently remove unwanted content from the application. Visual appeal and freshness are crucial, and we commit to regularly updating the appearance and contents of podcast lists. User feedback integration is a key aspect, with a system in place for users to suggest their most and least liked episodes, contributing to continuous improvement. Lastly, we focus on organized categorization to streamline content discovery by organizing podcasts into user-friendly categories. Addressing user-specific needs, we aim to provide personalized lists, allowing users to evaluate their favorite podcasts, comments, and friends' accounts for a tailored and social listening experience. Account management features empower users to delete specific podcast episodes from their lists and modify or delete their comments. User profile updates are facilitated, enabling users to refine their personal information and interests for a more personalized experience. A robust search function is implemented to facilitate seamless discovery, ensuring users can find the best episodes across all podcasts. To enhance user engagement, we incorporate dynamic playlists with a shuffle feature, providing a dynamic and varied listening experience.

Functionalities:
1. User: It contains the shared attributes and methods between the Administrators and End Users such as: login status.

2. Admin: To maintain the administrator personal information and the specialized privileges and methods to keep the system working, and contains the name, phone number, email address.

3. End User: To record the users' information, interests, and to play the episodes or share opinions with other users, and it contains the user Id, name, email address and survey results.

4. Personal Information: To save both admins and users information in the same attributes, and contains name, age, gender, email address, phone number, username, password and birth date.

5. Sound: To keep the podcasts details up to date and record any new ones, and it contains podcast name, number of episodes, hosts, categories and publish date.

6. Play List: To organize the users favorite podcasts, and it contains play list Id and category.

7. Categories: To organize the podcast in different categories to facilitate searching podcast in certain topics, and it contains Category Id and name.

8. Random QAs: A short survey for new users to analyze their favourites and interests to recommend the best podcast for them.

DIFFICULTIES & HOW PROBLEMS WERE SOLVED:

Problem: Not knowing the number of people in each type
Solution:

```
SELECT user_type, COUNT(*) AS total_users
FROM users GROUP BY user_type;
```

Problem: User registration dates are not arranged
Solution:

```
SELECT * FROM users
 ORDER BY regdate DESC;
```

Problem: Can't retrieve all playlists belonging to a specific category.
Solution:

```
SELECT * FROM playlist
WHERE cat_id = 1;
```

Problem: Can't retrieve random questions that have been listened to by all users.
Solution:

```
SELECT * FROM randomqas
WHERE listenORwatch = 'Listen';
```

Problem: The retrieval of random questions for a specific user is not feasible.
Solution:

```
SELECT * FROM randomqas
WHERE User_id = 3;
```

Problem: The retrieval of all podcasts of a specific type is not possible.

Solution:

```
SELECT * FROM sound
WHERE publishdate = '2022-01-01';
```

Problem: The inability to delete the sound.
Solution:

```
DELETE FROM sound
WHERE PodcastName = 'YsierKheir';
```

Problem: The inability to query and retrieve user information with birthdates between specific dates.
Solution:

```
SELECT * FROM end_user
WHERE DOB BETWEEN '2000-01-01' AND '2005-12-31';
```

Problem: Can't retrieve the sound associated with this specific hoster.
Solution:

```
select * FROM sound
WHERE Hoster LIKE '%MohamedAlbeez%';
```

Problem: The inability to update the name of a playlist.
Solution:

```
UPDATE playlist
SET PL_name = 'Favorites'
WHERE PL_id = 6001 AND user_id = 3;
```

Problem: The index cannot accelerate search operations using the 'User_id' column in the 'users' table.
Solution:

```
CREATE INDEX idx_user_id ON users (User_id);
```

Problem: Without data union, the integration process fails, resulting in incomplete user profiles and potentially compromising data integrity and analytical accuracy.
Solution:

```
SELECT User_id, Username, Email, NULL AS address, NULL AS DOB
FROM users
UNION
SELECT user_id, NULL AS Username, NULL AS Email, address, DOB
FROM end_user;
```

Problem: The duplicate cannot be found.
Solution:

```
SELECT User_id FROM users
INTERSECT
SELECT user_id FROM end_user;
```

Problem: The sound files rated above 4.5 cannot be found.
Solution:

```
SELECT s.PodcastName, AVG(ps.Rate) AS avg_rate
FROM sound s
JOIN playlist_sound ps ON s.sound_id = ps.sound_id
GROUP BY s.PodcastName
HAVING AVG(ps.Rate) > 4.5;
```

Problem: The categories that are not of the specified type cannot be retrieved.
Solution:

```
SELECT *
FROM categories
WHERE NOT cat_id = 1;
```

Problem: The unique (non-duplicate) names of users cannot be retrieved.
Solution:

```
SELECT DISTINCT Username
FROM users;
```

Problem: The retrieval of sound belonging to specific types is not possible.
Solution:

```
SELECT *
FROM sound
WHERE PodcastType IN (1, 2, 3);
```

Problem: The count of sound and their ratings cannot be calculated.
Solution:

```sql
SELECT
    COUNT(sound_id) AS total_sounds,
    AVG(rate) AS average_rate
FROM sound;
```

Problem: The retrieval of sound without a value for a specific field is not possible.
Solution:

```sql
SELECT *
FROM sound
WHERE albumName IS NULL;
```

Problem: The retrieval of rows from the 'end_user' table where the 'user_id' in any row is greater than any 'user_id' value in the 'users' table is not possible.
Solution:

```sql
SELECT *
FROM end_user
WHERE user_id > ANY (SELECT user_id FROM users);
```

Problem: need to retrieve all records from the users table in our database. To achieve this, we want to create a procedure named GetAllUsers() that, when called, will return all columns and rows present in the users table.

Solution:

```sql
DELIMITER $$
CREATE PROCEDURE GetAllUsers()
BEGIN
    SELECT * FROM users;
END$$
DELIMITER ;

CALL GetAllUsers();
```

Problem: Retrieve the playlist names and creation dates of playlists created by users from Saudi Arabia

Solution:

```
SELECT PL_name, creation_date
FROM playlist
WHERE user_id IN (
SELECT user_id
FROM end_user
WHERE Nat_id = 1
);
```

CHECKLIST OF REQUIREMENTS FULFILLED IN THE PROJECT:

REQUIREMENTS	Do it
Union/Intersect/Difference	All done
Distinct, aggregate functions, In, Between, ISNULL, NOT, LIKE	All done
Nested queries, comparison queries (ALL/ANY), Group By, Order By, Having, INTO, Case expression, and compute clause.	All done
Use DML (Data Manipulation Language) for (insert/update/delete).	All done
Create a database objects, such as: Index, Procedure or Function, Trigger and Assertion.	All done
DDL (Data Definition Language)	All done

WERE TASKS HAVE BEEN DISTRIBUTED EQUALLY BETWEEN MEMBERS?
Yes, the tasks were distributed equally.

References:
Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2014). Data structures and algorithms in Java. John wiley & sons.

Lafore, R. (2017). Data structures and algorithms in Java. Sams publishing.

Carrano, F. M., & Savitch, W. J. (2003). Data structures and abstractions with Java. Upper Saddle River, NJ, USA: Prentice Hall.