Group Project Name: GigaGamer
Group Member Names: Esther Tan, Owen Foster, Brendan Van Diest, Ryan Owsley

Content

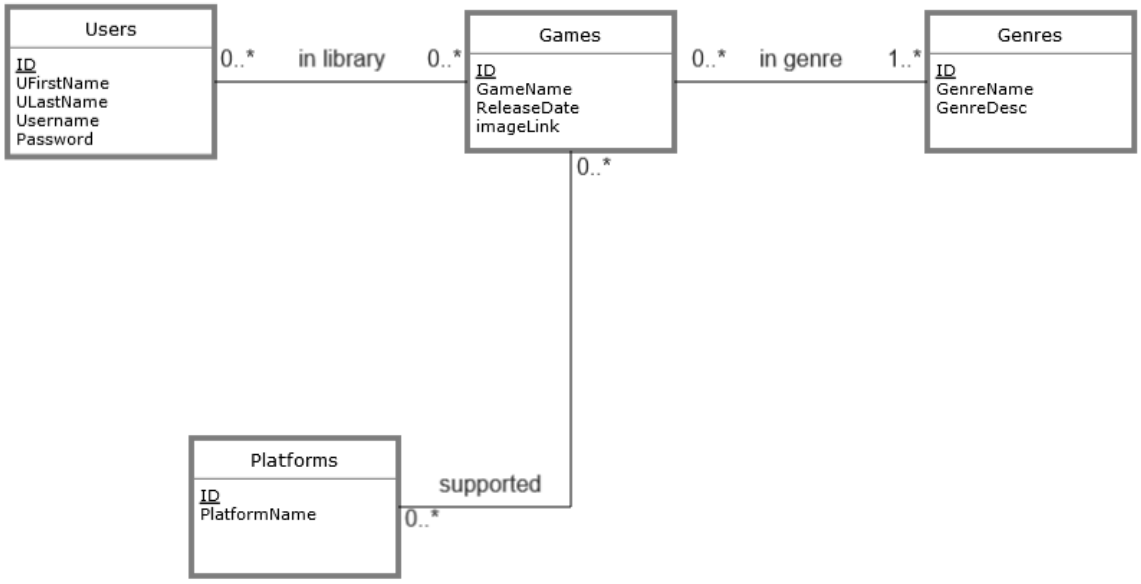
Application Description

GigaGamer is a program meant for people who regularly game or people who are new to games to help keep track of the games they play while also helping them find more of the games they might like to play. The program allows you to create an account, in which you can search for titles and save them under your account.

Functions:

- Account creation
- Searching for a game and providing information on it
- Saving games
- Find similar games based on platform
- Find similar games based on genre

Database Design



Keys

1. Users
    a. ID
2. Games
    a. ID
3. Genres
    a. ID
4. Platforms
    a. ID

Foreign Keys

1. Users
    a. ID from Games
2. Games
    a. ID from User
    b. ID from Genres
    c. ID from Platforms

3. Genres
    a. ID from Games
4. Platforms
    a. ID from Games

Database Redundancies

The redundancy problem in the database arises when there is duplicated platform information across multiple game records in the "games" table. This redundancy can lead to data inconsistency and inefficiency. To address this issue, we create a view called "game_platform_view" that joins the "games" table with the "platform" table. By doing so, we consolidate the relevant data from both tables into a single view, eliminating the need for duplicating platform details for each game. This solution helps us maintain data integrity, simplify queries, and present a consistent representation of games and their associated platforms.

Queries

a. Queries in the application
    1. "SELECT * FROM " + this.PrimaryView + " WHERE Username = '" + username + "'";

    2. "INSERT INTO User_Games (gameID, userID) VALUES (@gameID, @userID)"

    3. "DELETE FROM user_games WHERE userID = @userId AND gameID IN ({gameIdsString})"

    4. "SELECT * FROM Games WHERE ID IN (SELECT gameID FROM User_Games WHERE userID = '" + this.ID + "')"

    5. "INSERT INTO " + this.TableName + " DEFAULT VALUES"
       "SELECT LAST_INSERT_ID();"

    6. "UPDATE " + this.TableName + " " +
       "SET UFirstName = @firstName, ULastName = @lastName, Username = @username, Password = @password " +
       "WHERE ID = @ID"

7. "SELECT * FROM games WHERE ID IN (SELECT DISTINCT ID FROM game_platform_view WHERE " + sql + ")";
   return sql

8. if (Criteria.title != "")
   {

   sql = sql + " GameName LIKE '%" + Criteria.title + "%'";
   }

   if (Criteria.platformID != 0)
   {
   if (sql != "")
   {
   sql += " AND ";
   }
   sql = sql + "PlatformID = " + Criteria.platformID;

   }

   sql = "SELECT * FROM games WHERE ID IN (SELECT DISTINCT ID FROM game_platform_view WHERE " + sql + ")";

9. "SELECT * FROM platforms"

10. "SELECT * FROM genres"

b. Queries in english

1. Retrieve all columns from the table specified in "this.PrimaryView" where the Username matches the provided username.

2. Insert a new record into the "User_Games" table with the given gameID and userID values.

3. Delete records from the "user_games" table where the userID matches the provided userId and the gameID is within the list of game IDs specified in "gameIdsString".

4. Retrieve all columns from the "Games" table where the ID matches any gameID from the "User_Games" table for the given userID.

5. Insert a new record into the table specified in "this.TableName" with default values and retrieve the last inserted ID.

6. Update the table specified in "this.TableName" by setting the UFirstName, ULastName, Username, and Password columns to the provided values where the ID matches the provided ID.

7. Retrieve all columns from the "games" table where the ID matches any distinct ID from the "game_platform_view" table that satisfies the given condition.

8. Construct a SQL query based on the given criteria: If the title in the criteria is not empty, retrieve all columns from the "games" table where the GameName contains the provided title. If the platformID in the criteria is not zero, add an additional condition to the query to retrieve records where the PlatformID matches the provided platformID. Retrieve all columns from the "games" table where the ID matches any distinct ID from the "game_platform_view" table that satisfies the constructed SQL query.

9. Retrieve all records from the "platforms" table.

10. Retrieve all records from the "genres" table.

Application:

Find our source code here:
https://whitgit.whitworth.edu/2023/spring/CS-374-1/Group_Projects/gamerecommender

Please see the ReadMe.md file in our repository for steps to run the program.