Entities

1. **`Achievment`**: Stores achievements for games, with images and descriptions, linked to `Game`.
2. **`Character`**: Contains character details (e.g., full name).
3. **`CharacterRole`**: Junction table between `Game` and `Character`, defining roles using the role-level enum.
4. **`CompanyManager`**: Stores manager details for game studios, including email and position.
5. **`Game`**: Central entity representing a game, with attributes like title, price, and studio reference.
6. **`GameAward`**: Stores awards received by games, with unique award names and years.
7. **`Genre`**: Represents game genres.
8. **`Guide`**: User-created guides for games, with images, creation timestamps, and updates.
9. **`Inventory`**: Tracks items owned by users in the game, linking users to items.
10. **`Item`**: Represents in-game items with attributes like title, description, price, and tradability.
11. **`Language`**: Stores available languages for localization.
12. **`Library`**: Tracks user game ownership, including purchase date and playtime.
13. **`Localization`**: Defines game language support (audio, interface, subtitles).
14. **`Review`**: Stores user reviews for games, including rating and timestamps.
15. **`Studio`**: Represents game studios with information like name, location, and founding year.
16. **`SupportRequest`**: Logs user support requests, linked to a company manager.
17. **`User`**: Represents users of the platform, including username, birth date, and email.
18. **`User_Achievment`**: Junction table linking users to achievements.
19. **`User_Guide`**: Junction table linking users to guides.
20. **`User_Review`**: Junction table linking users to reviews with a view mode (enum).
21. **`User_User`**: Represents user friendships on the platform.
22. **`Genre_Game`**: Junction table linking games to genres.

Total Entities: 18

DBMS: MariaDB

`https://dbdiagram.io/d/GameServiceDB-66dc0ef0eef7e08f0efa18ba`

Functions

get_inventory_price(_user_id)

get_played_hours(_user_id, _game_id)

Procedures

make_purchase(_user_id, _game_id)

//adds game to user library

//if game was refunded new record is not created

make_refund(_user_id, _game_id)

//soft delete game from user library

//this way we still store spent hours in game

soft_delete_user(_user_id)

//soft delete user

//created view table to easily identify deleted profiles

recover_user(_user_id)

//recovering user

View Tables

GameCharacterView

//demonstrated role of character in game

//similar as wiki page for specific game character

GameLocalizationView

//demonstrates game localization details

UserInventoryView

//demonstrates item details and owner

UserLibraryView

//demonstrates game details and owner

Triggers

creation_info_*

//sets creator and time when record was created

//before insert

update_info

//updates fields updated_at, updated_by

//before update

Both triggers used for 2 tables. 4 in total.