Gaming Console Database Project

by

Prateek Kumar (22CSB0B37)

Aayush Thakur (22CSB0F25)

Project Overview: Gaming Console Database

Introduction: The Gaming Console Database project aims to create a structured repository for organizing and managing data related to gaming consoles, games, players, developers, and other relevant information. Unlike a Database Management System (DBMS), which includes software for managing databases, this project focuses solely on designing and implementing the database schema.

Objectives: The primary objectives of the project are as follows:

- Design and implement a relational database schema to store gaming console-related data efficiently.
- Define tables, columns, primary and foreign keys, and establish relationships between entities.
- Populate the database with sample data to demonstrate its functionality.
- Provide documentation detailing the database schema, relationships, and sample queries.

Scope: The scope of the project includes the following key areas:

- Game Information: Storing details about individual games, including titles, genres, developers, release dates, ratings, and descriptions.
- Player Profiles: Managing player information, such as usernames, passwords, email addresses, and age.
- Activity records: Tracking activities of players, including playtime, game progress, and trophies.
- Console Details: Maintaining data related to gaming consoles.

Functionalities: The Gaming Console Database will offer the following functionalities:

- Creation of Tables: Defining tables for games, players, and consoles, along with their respective attributes.
- Establishing Relationships: Establishing relationships between tables to capture associations accurately, such as player-game relationships.
- Query Support: Demonstrating sample queries to retrieve information from the database, such as displaying a player's game records or listing all available games for a particular platform.

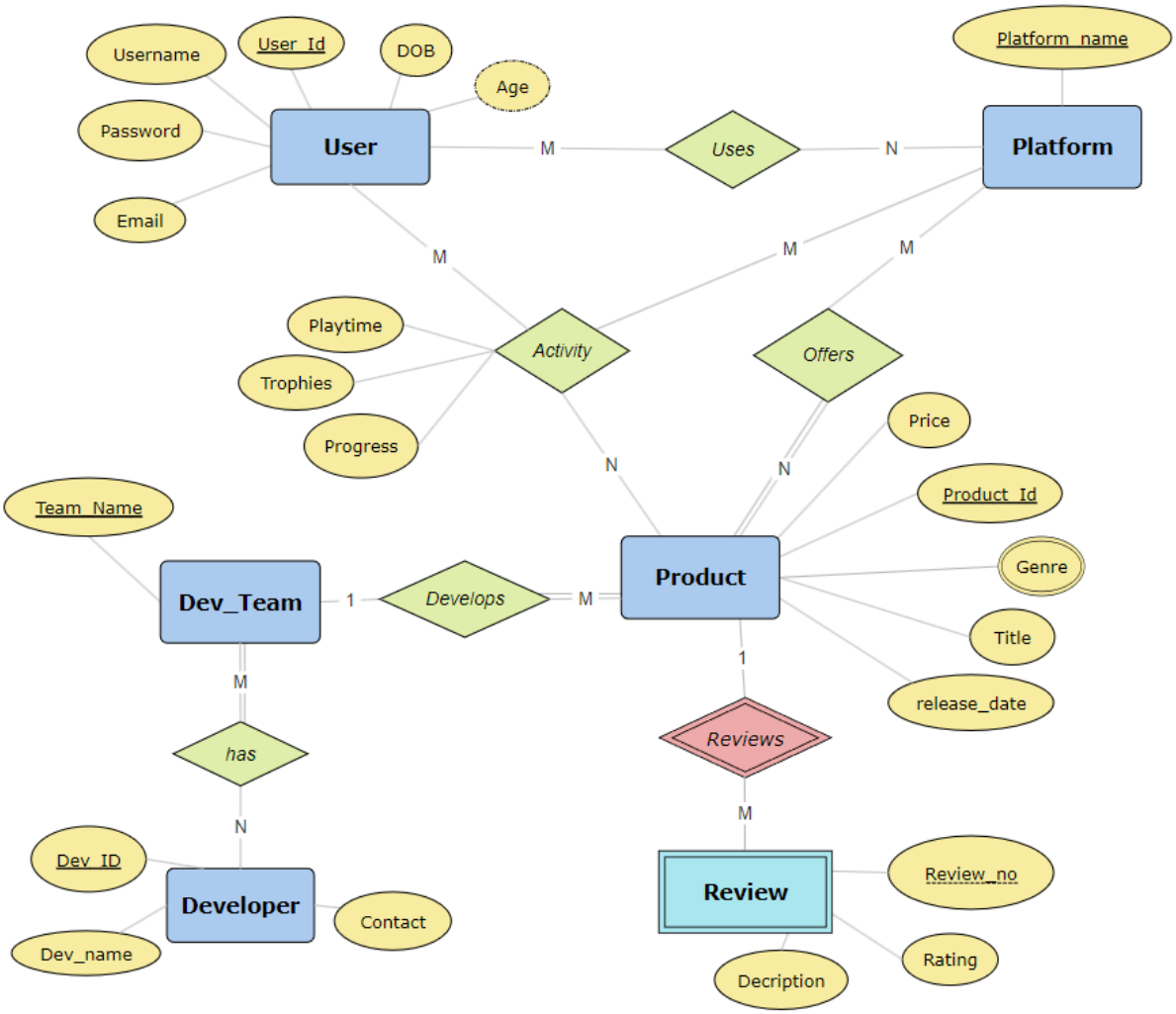
Design Considerations: Several design considerations guide the development of the database:

- Normalization: Ensuring the database schema is normalized to minimize redundancy and maintain data integrity.

- Entity-Relationship Modeling: Using ER diagrams to visualize the relationships between entities and refine the database schema.
- Data Integrity: Implementing constraints and validations to enforce data integrity rules, such as unique constraints and foreign key constraints.
- Documentation: Providing comprehensive documentation outlining the database schema, relationships, and sample queries to facilitate understanding and usage.

Conclusion: The Gaming Console Database project aims to create a structured repository for organizing and managing gaming console-related data. By focusing on database design and implementation, the project provides a foundation for efficiently storing and retrieving information related to games, players, purchases, and consoles. This overview outlines the project's objectives, scope, functionalities, and design considerations, laying the groundwork for successful development and implementation of the database.

ER Diagram:



Assumptions:

- **Multi-Platform Games**: Assuming that each game may be developed for a multiple gaming platform, but activity of same game on different platform will be taken different.
- **No Player Interactions**: Assuming that player interactions (e.g., friend requests, messages) are not within the scope of the system, focusing solely on player profiles and their relationships with games and purchases.
- **Transactions:** Assuming that transactions are managed externally and focusing solely on player profiles within the gaming console system.
- **Multi-Platforms Login:** Assuming that user is allowed to login to the same account on multiple platforms.
- **Account without Platform:** A user can hold an account without being on any of the platforms.
- **No Pre-defined Genres:** Assuming that there are no pre-defined genres and developers can give any genre names to the game they developed.

- **Anonymous Reviews:** Assuming that anonymity is provided to users for publishing reviews makes reviews a weak entity depending on product.
- **Playable restriction:** A user can play only those games offered by the platforms they use.
- **Team of Developers:** A game can be developed by one and only one team having one or more developers, whereas a team may develop multiple games.
- **No Regional Restrictions**: Assuming that games are available globally without regional restrictions or differences in content.

Relational Model:

The relational model for the ER model can be given as:

- **User** (User_Id, Username, Email, Password, DOB, Age)
- **Platform** (Platform_name)
- **Product** (Product_Id, Title, Price, Genre, Release_date, Team_name)
- **Uses** (User_Id, Platform_name)
- **Offers** (Platform_name, Product_Id)
- **Activity** (User_Id, Platform_name, Product_Id, Progress, Trophies, Playtime)
- **Dev_team** (Team_name)
- **Developer** (Dev_Id, Dev_name, Contact)
- **Has** (Team_name, Dev_Id)
- **Review** (Review_no, Product_Id, Rating, Description)

The FDs are primary key to all attributes of the relation and in **User** relation an extra FD is DOB -> Age.

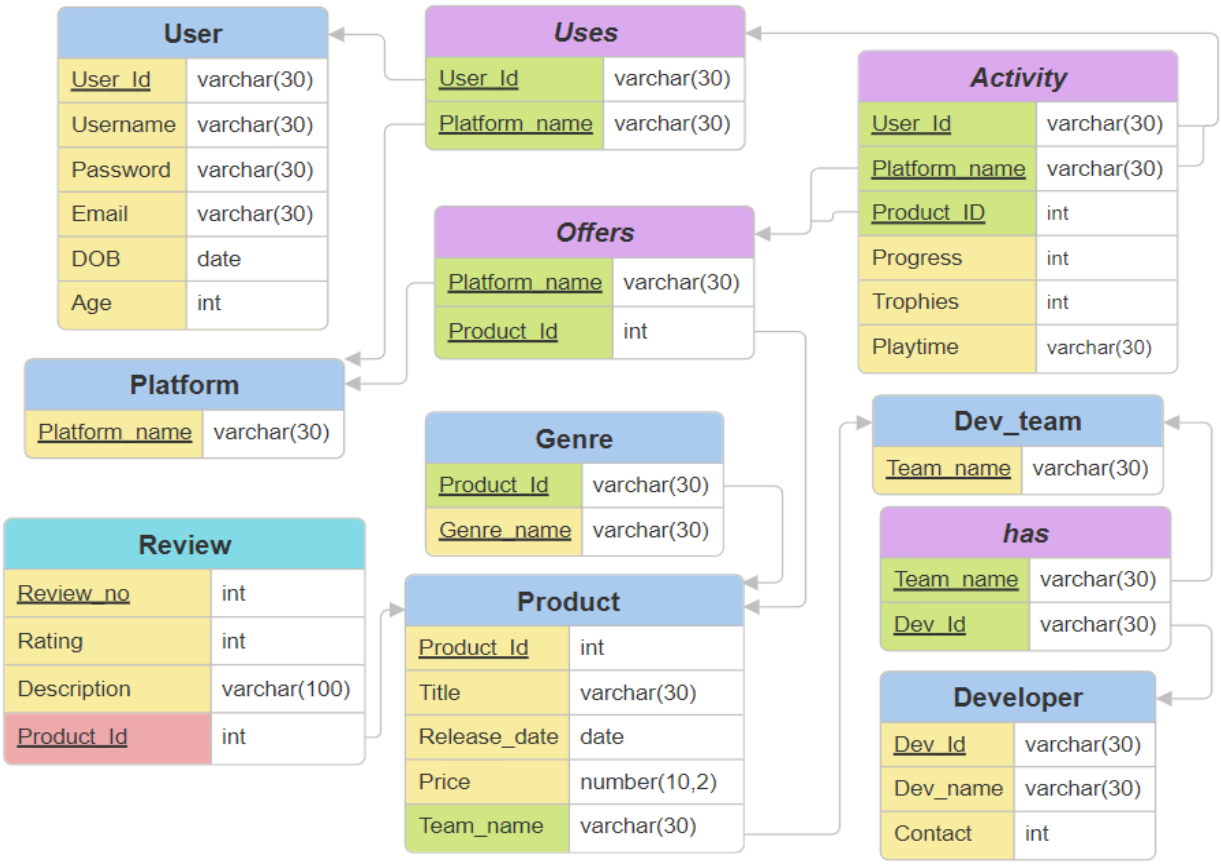
Normalization:

1st Normal Form: In 1NF the model should have atomic attributes, so we add a new relation **Genre** (Product_Id, Genre_name) to remove the multivalued attribute (genre) from **Product** relation.

2nd Normal Form: In 2NF the model all non-key attributes should have a fully functional dependency on the primary keys which is true for this relational model.

3rd Normal Form: In 3NF there should be no transitive dependency, i.e. no non-key attribute should derive another non-key attribute. So, we remove the attribute Age from the **User** relation as it is a derived attribute and need not be stored.

Finally, the model is in 3rd Normal Form and can be represented graphically as:



The arrows in the diagram represent **foreign key** constraint and underline represent **primary key** constraint.

SQL Code:

Table Creation:

```
-- Table creation statements
create table USER(
    USERNAME varchar(30),
    PASSWORD varchar(30),
    EMAIL varchar(30),
    USER_ID varchar(30),
    DATE_OF_BIRTH date,
    primary key(USER_ID)
);
```

```
create table PLATFORM(
    PLATFORM_NAME varchar(30) primary key
);
```

```
create table USES(
    USER_ID varchar(30),
    PLATFORM_NAME varchar(30),
    primary key(USER_ID,PLATFORM_NAME),
    foreign key(USER_ID) references USER(USER_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    foreign key(PLATFORM_NAME) references PLATFORM(PLATFORM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table DEV_TEAM(
    TEAM_NAME varchar(30) primary key
);
```

```
create table PRODUCT(
    PRICE int,
    PRODUCT_ID int primary key,
    TITLE varchar(30),
    RELEASE_DATE date,
    TEAM_NAME varchar(30) NOT NULL,
    foreign key(TEAM_NAME) references DEV_TEAM(TEAM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table GENRE(
```

```
    PRODUCT_ID int ,
    GENRE1 varchar(30),
    primary key(PRODUCT_ID,GENRE1)
    foreign key(PRODUCT_ID) references PRODUCT(PRODUCT_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table OFFERS(
    PLATFORM_NAME varchar(30),
    PRODUCT_ID int,
    primary key(PLATFORM_NAME,PRODUCT_ID),
    foreign key(PLATFORM_NAME) references PLATFORM(PLATFORM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    foreign key(PRODUCT_ID) references PRODUCT(PRODUCT_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table ACTIVITY(
    PLAYTIME varchar(30),
    TROPHIES int,
    PROGRESS int,
    USER_ID varchar(30),
    PRODUCT_ID int,
    PLATFORM_NAME varchar(30),
    primary key(USER_ID,PRODUCT_ID,PLATFORM_NAME),
    foreign key(USER_ID,PLATFORM_NAME) references USES(USER_ID,PLATFORM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    foreign key(PRODUCT_ID,PLATFORM_NAME)references OFFERS
(PRODUCT_ID,PLATFORM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table DEVELOPER(
    DEV_ID varchar(30) primary key,
    DEV_NAME varchar(30),
    CONTACT_INFO int
);
```

```
create table HAS(
    DEV_ID varchar(30),
    TEAM_NAME varchar(30) NOT NULL,
    primary key(DEV_ID,TEAM_NAME),
    foreign key(DEV_ID) references DEVELOPER(DEV_ID)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    foreign key(TEAM_NAME) references DEV_TEAM(TEAM_NAME)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
create table REVIEW(
    DESCRIPTION varchar(100),
    RATING int,
    REVIEW_NO int,
    PRODUCT_ID int NOT NULL,
    primary key(REVIEW_NO,PRODUCT_ID),
    foreign key(PRODUCT_ID) references PRODUCT(PRODUCT_ID)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Data Insertion:

```sql
-- Inserting data into USER table
INSERT INTO USER (USERNAME, PASSWORD, EMAIL, USER_ID, DATE_OF_BIRTH)
VALUES ('user1', 'password1', 'user1@example.com', '1', '1990-01-01'),
       ('user2', 'password2', 'user2@example.com', '2', '1995-05-15'),
       ('user3', 'password3', 'user3@example.com', '3', '1988-11-30'),
       ('user4', 'password4', 'user4@example.com', '4', '1992-03-12'),
       ('user5', 'password5', 'user5@example.com', '5', '1997-08-25');

-- Inserting data into PLATFORM table
INSERT INTO PLATFORM (PLATFORM_NAME)
VALUES ('PlayStation'), ('Xbox'), ('Nintendo Switch');

-- Inserting data into USES table
INSERT INTO USES (USER_ID, PLATFORM_NAME)
VALUES ('1', 'PlayStation'),
       ('2', 'Xbox'),
       ('3', 'Nintendo Switch'),
       ('3', 'PlayStation'),
       ('4', 'Xbox'),
       ('5', 'Nintendo Switch');

-- Inserting data into DEV_TEAM table
INSERT INTO DEV_TEAM (TEAM_NAME)
VALUES ('Team A'), ('Team B'), ('Team C');

-- Inserting data into PRODUCT table
INSERT INTO PRODUCT (PRICE, PRODUCT_ID, TITLE, RELEASE_DATE, TEAM_NAME)
VALUES (59, 1, 'Game1', '2023-05-01', 'Team A'),
       (49, 2, 'Game2', '2024-01-15', 'Team B'),
       (39, 3, 'Game3', '2023-10-20', 'Team C'),
       (49, 4, 'Game4', '2024-02-10', 'Team B'),
       (39, 5, 'Game5', '2023-12-05', 'Team C');

-- Inserting data into GENRE table with multiple genres for each game
INSERT INTO GENRE (PRODUCT_ID, GENRE1)
VALUES (1, 'Action'),
       (1, 'Adventure'),
       (2, 'Adventure'),
       (3, 'Puzzle'),
       (4, 'Racing'),
       (5, 'Role-Playing');

-- Inserting data into OFFERS table
INSERT INTO OFFERS (PLATFORM_NAME, PRODUCT_ID)
VALUES ('PlayStation', 1),
       ('Xbox', 2),
```

```
       ('Nintendo Switch', 3),
       ('PlayStation', 2),
       ('Xbox', 4),
       ('Nintendo Switch', 5);

-- Inserting data into ACTIVITY table
INSERT INTO ACTIVITY (PLAYTIME, TROPHIES, PROGRESS, USER_ID, PRODUCT_ID,
PLATFORM_NAME)
VALUES ('50 hours', 20, 80, '1', 1, 'PlayStation'),
       ('30 hours', 15, 70, '2', 2, 'Xbox'),
       ('20 hours', 10, 50, '3', 3, 'Nintendo Switch'),
       ('40 hours', 18, 60, '3', 2, 'PlayStation'),
       ('25 hours', 12, 40, '4', 4, 'Xbox'),
       ('35 hours', 18, 60, '5', 5, 'Nintendo Switch');

-- Inserting data into DEVELOPER table
INSERT INTO DEVELOPER (DEV_ID, DEV_NAME, CONTACT_INFO)
VALUES ('DEV1', 'Developer A', 1234567890),
       ('DEV2', 'Developer B', 9876543210),
       ('DEV3', 'Developer C', 999888777),
       ('DEV4', 'Developer D', 5551234567),
       ('DEV5', 'Developer E', 9871234567);

-- Inserting data into HAS table
INSERT INTO HAS (DEV_ID, TEAM_NAME)
VALUES ('DEV1', 'Team A'),
       ('DEV2', 'Team B'),
       ('DEV3', 'Team C'),
       ('DEV4', 'Team B'),
       ('DEV5', 'Team C');

-- Inserting data into REVIEW table
INSERT INTO REVIEW (DESCRIPTION, RATING, REVIEW_NO, PRODUCT_ID)
VALUES ('Great game!', 5, 1, 1),
       ('Good storyline.', 4, 2, 2),
       ('Challenging puzzles.', 3, 3, 3),
       ('Exciting multiplayer!', 4, 4, 4),
       ('Immersive storyline.', 5, 5, 5),
       ('Great graphics and gameplay!', 4, 6, 1),
       ('Enjoyable multiplayer experience.', 4, 7, 2),
       ('Clever level design.', 5, 8, 3),
       ('Fast-paced racing action!', 4, 9, 4),
       ('Captivating story with memorable characters.', 5, 10, 5);
```

Some Select Queries:

1. Retrieve the games in descending average rating:

```
SELECT P.TITLE, AVG(R.RATING) AS AVERAGE_RATING
FROM PRODUCT P
NATURAL JOIN REVIEW R
GROUP BY P.TITLE
ORDER BY AVERAGE_RATING DESC;
```

```
Game2                                    4
```

2. Find the total number of games offered on each platform:

```
SELECT PL.PLATFORM_NAME, COUNT(O.PRODUCT_ID) AS TOTAL_GAMES_OFFERED
FROM PLATFORM PL
natural JOIN OFFERS O
GROUP BY PL.PLATFORM_NAME;
```

OUTPUT:

3. Identify users who have played games for more than 30 hours on any platform:

```
SELECT DISTINCT U.USERNAME
FROM USER U
Natural JOIN ACTIVITY A
WHERE A.PLAYTIME > '30 hours';
```

OUTPUT:

4. Retrieve the average price of each genres:

```
SELECT G.GENRE1, AVG(price) as AVERAGE_PRICE
FROM PRODUCT P
NATURAL JOIN GENRE G
GROUP BY G.GENRE1;
```

OUTPUT:

5. Retrieve the total number of trophies earned by each user:

```
SELECT U.USERNAME, SUM(A.TROPHIES) AS TOTAL_TROPHIES
FROM USER U
NATURAL JOIN ACTIVITY A
GROUP BY U.USERNAME;
```

OUTPUT: