Video Game Library Database

Purpose of this Document

This document provides a **clear, structured, and detailed explanation** of the entire database design for the *Video Game Library Management System*. It is intended to be used during the **oral presentation**, where three speakers will explain different parts of the database.

The explanation is divided into: - A **table-by-table breakdown** (what each table stores and why) - A **relationship explanation** (how tables depend on each other) - A **speaker division guide** (who explains what)

---

1. Core Entity Tables

These tables represent **real-world objects** and do not depend on other tables to exist.

1.1 `developer` Table

What it represents: Companies or studios that *develop* video games.

Why it exists: Storing developer data separately avoids duplication and follows normalization principles.

Main columns: - `developer_id` – Primary Key, unique identifier - `name` – Developer name - `country` – Country of origin

Relationships: - One developer can develop **many games** - Referenced by `game.developer_id`

1.2 `publisher` Table

What it represents: Companies that *publish* video games.

Why it exists: Developers and publishers are often different entities. Separating them keeps the model realistic and normalized.

Main columns: - `publisher_id` – Primary Key - `name` – Publisher name - `country` – Country of origin

Relationships: - One publisher can publish **many games** - Referenced by `game.publisher_id`

1.3 `genre` Table

What it represents: Game categories such as RPG, Action, Adventure.

Why it exists: Genres are reused across many games and must not be duplicated.

Main columns: - `genre_id` – Primary Key - `name` – Genre name (unique)

Relationships: - Many games can belong to many genres - Connected through `game_genre`

1.4 `platform` Table

What it represents: Hardware or systems where games run (PC, PlayStation, Xbox).

Why it exists: A game can be available on multiple platforms.

Main columns: - `platform_id` – Primary Key - `name` – Platform name (unique)

Relationships: - Many-to-many relationship with games - Connected through `game_platform`

---

2. Central Table

2.1 `game` Table

What it represents: The main entity of the database, storing core information about video games.

Why it is central: Almost every other table connects to `game`.

Main columns: - `game_id` – Primary Key - `title` – Game title - `release_year` – Year of release - `developer_id` – Foreign Key → developer - `publisher_id` – Foreign Key → publisher

Relationships: - Each game has **one developer** - Each game has **one publisher** - Each game can have **many genres** - Each game can run on **many platforms** - Each game can be borrowed **many times**

---

3. Associative (Bridge) Tables

These tables resolve **many-to-many relationships**.

3.1 `game_genre` Table

Purpose: Links games and genres.

Main columns: - `game_id` – Foreign Key → game - `genre_id` – Foreign Key → genre

Why it exists: A game can belong to multiple genres, and a genre can describe multiple games.

3.2 `game_platform` Table

Purpose: Links games and platforms.

Main columns: - `game_id` – Foreign Key → game - `platform_id` – Foreign Key → platform

Why it exists: Ensures correct modeling of platform availability.

---

4. User & Transaction Tables

4.1 `user` Table

What it represents: People who can borrow games from the library.

Main columns: - `user_id` – Primary Key - `username` – Unique username - `email` – Unique email

Relationships: - One user can have **many borrowings**

4.2 `borrowing` Table

What it represents: Borrowing activity between users and games.

Why it is important: This table tracks real-world usage and is the most dynamic table.

Main columns: - `borrowing_id` – Primary Key - `user_id` – Foreign Key → user - `game_id` – Foreign Key → game - `borrow_date` – Date of borrowing - `return_date` – NULL if not returned

Business logic: - `return_date IS NULL` → game currently borrowed

---

5. Speaker Distribution for Presentation

Speaker 1 – Core Entities

Explains: - developer - publisher - genre - platform

Focus: - Normalization - Why these tables are independent

Speaker 2 – Central Logic

Explains: - game - game_genre - game_platform

Focus: - Central role of game - Many-to-many relationships - Referential integrity

---

Speaker 3 – Transactions & Performance

Explains: - user - borrowing - Indexes

Focus: - Real-world usage - Borrowing logic - Performance and scalability

---

Final Note

This structure ensures: - Data integrity - No redundancy - Efficient querying - Easy scalability

The database fully satisfies academic and practical database design requirements.