CS 340 Project: Game Library
Nathan Perkins
Heather Godsell

Game Library Web App

Project Outline

At Nathan's work, there is an extensive library of video games that employees can check out for free and return whenever convenient. Currently, the library is managed by one person, manually on an Excel spreadsheet. We will work on a game library web app that can be used by employees to examine the current library inventory and to make requests. The heart of the web app will be a database that contains all the information about the users, games, game metadata, platforms and requests.

The project will be written in node.js, HTML and CSS. We will use hand-written MySQL queries to manage all database access and modifications.

Project Feedback and Changes

Note on Peer Feedback

The feedback we received was for our initial project idea based on a database for the video game, *Stardew Valley*. We decided to change our project so this feedback no longer applies. Below, we have included our feedback from the last project and our responses to that feedback, just in case it is required for grading. Then, we provide the justification for our project change.

Feedback by the peer reviewer

1.
Are there sufficient entities as required?
Yes. Heather and Nathan have 6 entities (required: 4).
2.
Are there sufficient number and types of relationships as required ?
Yes. The Villagers <-> Gems relationship satisfies the many-to-many requirement.
3.
Is there a good reason to have the things described as entities to be entities?

Or can they be just attributes of other entities?
Locations is an entity that was a bit confusing to me. For Villagers, the home attribute links to the Locations entity. However, for Players, the farm attribute (which seems to also represent a location) does not link to the Locations entity. I wasn't able to discern the difference. You may want to consider making location an attribute linking to the Locations entity for both (or neither and remove the Locations entity).
4.
Do the various relationships make sense to you ?
"Buildings & Animals are owned by one Player, and a Player can own many animals (one-to-many)"
You may want to break Buildings and Animals up into separate points or include a description of the Player to Buildings relationship like you did for Player and Animals in the latter half of the description.
5.
Is there sufficient information about each attribute to justify it's presence ?
Yes. I feel like I have a good picture based on the description of each attribute.
6.
Are the data types mentioned for attributes ? It need not be an actual keyword from MySQL/MariaDB like "varchar" but is it sufficiently described as a string, number, date, etc?
Yes, data types are specified.
7.
Do the data types make sense for the attributes?
Yes.
8.
Are constraints described for attributes? Do they make sense ? The implementation details of the constraints do not matter right now.
Yes, constraints are described. The max char for name (15) seemed a bit low to me but not being familiar with the game, I can't say for certain.
9.
Is there anything else that you wish the student did to make their outline better and more suitable for a CS340 Project?
The outline was well thought out and suitable for a CS340 Project.

Actions based on the feedback

Although we changed our project entirely, here are our responses to the actions that were suggested to our original project. We didn't feel that any action was necessary but will offer clarification.

1. Regarding the comment on the max char for a name being 15 characters, it is only intended to hold a first name and it needs to be easy enough to display frequently.
2. Regarding the locations entity, a location is intended to describe an area outside the user's farm. The user's farm is special in that it is customizable by the player.
3. Regarding the relationship between animals, buildings, and players, players own buildings. A building is located on the player's farm, and can only be owned by one player, however there can be many buildings owned by one player. Animals simply live inside those buildings. This eliminates the direct relationship between players and animals.

Upgrades to the Draft version

We decided to change our project from a database representing the game *Stardew Valley* to this game library project because we felt that we wanted to complete a project that would demonstrate higher level skills and be more impressive on our resumes. The game library project is an improvement because:

1. It will be used by people in the real world, requiring us to make real decisions about which features that people will find useful. We will also be externally motivated to maintain it and fix bugs as necessary.
2. It has the potential to pull in data via real-world APIs (MobyGames) or through screen/DOM scraping (game review sites), demonstrating skills that are desirable in the job market.

Database Outline

Entities

- **All Entities:**
    - All entities will have these fields:
    - id: an auto-incremented integer identifying the user; cannot be NULL; used as the primary key
- **Users:**
    - Describes a user of the web app and their role.
    - first_name: a varchar string containing the user's first name; cannot be NULL
    - last_name: a varchar string containing the user's last name; cannot be NULL
    - email: a varchar string containing the user's email; cannot be NULL. Is Unique.
    - password: a varchar string containing the user's hashed password; cannot be NULL
    - password_salt: a varchar string containing the user's unique password salt; cannot be NULL
    - role: an enum describing the user's role. Cannot be NULL
        - 'User', 'admin', 'root'
- **Game_Copies:**
    - Describes an actual physical copy of a video game that is in the library.
    - status: an enum describing the status of this copy; cannot be NULL
        - Available, checked_out, lost
    - release_id: a foreign key linking a copy of a game to the metadata about a release of that game; cannot be NULL.
    - library_tag: a varchar containing the library tag that is stuck to the game box; cannot be NULL
    - dt_procured: a date field containing the date this copy was purchased; can be NULL.
- **Game_Titles:**
    - Describes metadata related to a game title, but not a specific platform.
    - name: a varchar string containing the game's name; cannot be NULL
    - description: a TEXT field containing a description of the game; can be NULL
    - genre: a varchar string containing the game's genre; can be NULL
    - developer: a varchar string containing the game's developer; can be NULL
    - producer: a varchar string containing the game's producer; can be NULL
- **Game_Releases:**
    - A relationship table linking a GameTitle to a Platform, and any additional information associated with that game title on that platform.
    - title_ID: A foreign key linking a game release to GameTitle about that release; cannot be NULL

    - platform_ID: A foreign key linking a game release to the platform of that release; cannot be NULL
    - release_date: A date field representing the release date; cannot be NULL
    - rating: an integer field representing the review score of this release
    - boxart_url: A varchar string with a URL link to boxart for this release.
- **Game_Platforms:**
    - Describes a game platform, like a specific console family or a PC.
    - Name: A varchar string with the name of this platform; cannot be NULL. Is Unique.
    - Manufacturer: A varchar string with the manufacturer of this platform; can be NULL
    - Release date: A date field with the release date of this platform; can be NULL
- **Game_Requests:**
    - Describes a request made by a user to borrow a game, and the status of that request until completion.
    - user_ID: foreign key linking a request to a specific user. Cannot be NULL
    - release_ID: foreign key linking a request to a release of a game. Used for the request initially, before an actual copy of the game is delivered. Cannot be NULL
    - game_ID: foreign key linking a request to a specific copy of a game, once it has been delivered. Can be NULL
    - dt_requested: datetime field for when the game was requested by the user.
    - dt_delivered: datetime field for when the game was delivered to the user.
    - dt_completed: datetime field for when the game was returned to the library or the request cancelled.
    - status: an enum describing the status of the request. Cannot be null
        - 'pending', 'checked_out', 'completed'

Relationships

- **Game_Releases - game_titles, and game_platforms**
    - Many-to-many relationship involving game_titles, and game_platforms.
    - A Game_Title can have many game_platforms through game_releases.
    - A game_platform can have many game_titles through game_releases.
    - Contains additional data about the release, described in the entities section.
- **Game_Requests - users, game_releases**
    - Many-to-Many relationship involving users, game_releases.
    - A user can make many requests for a game_release.
    - A game_release can have many requests by many users.
    - Contains additional data about the request, described in the entities section.
- **Adding a game to the library - game_copies and game_releases**
    - One to many relationship involving a game game_release and a game copy.
    - A game_copy must be a purchased physical copy (instance) of a game_release.
    - A game_release can have many purchased copies.

- **Loaning a game out - game_requests and game_copies**
    - One-to-many relationship involving a game_request and a game_copy.
    - A game_request can be fulfilled by loaning out one game_copy.
    - A game_copy can fulfill many requests (one at a time).