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 Fixes

Project Change

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

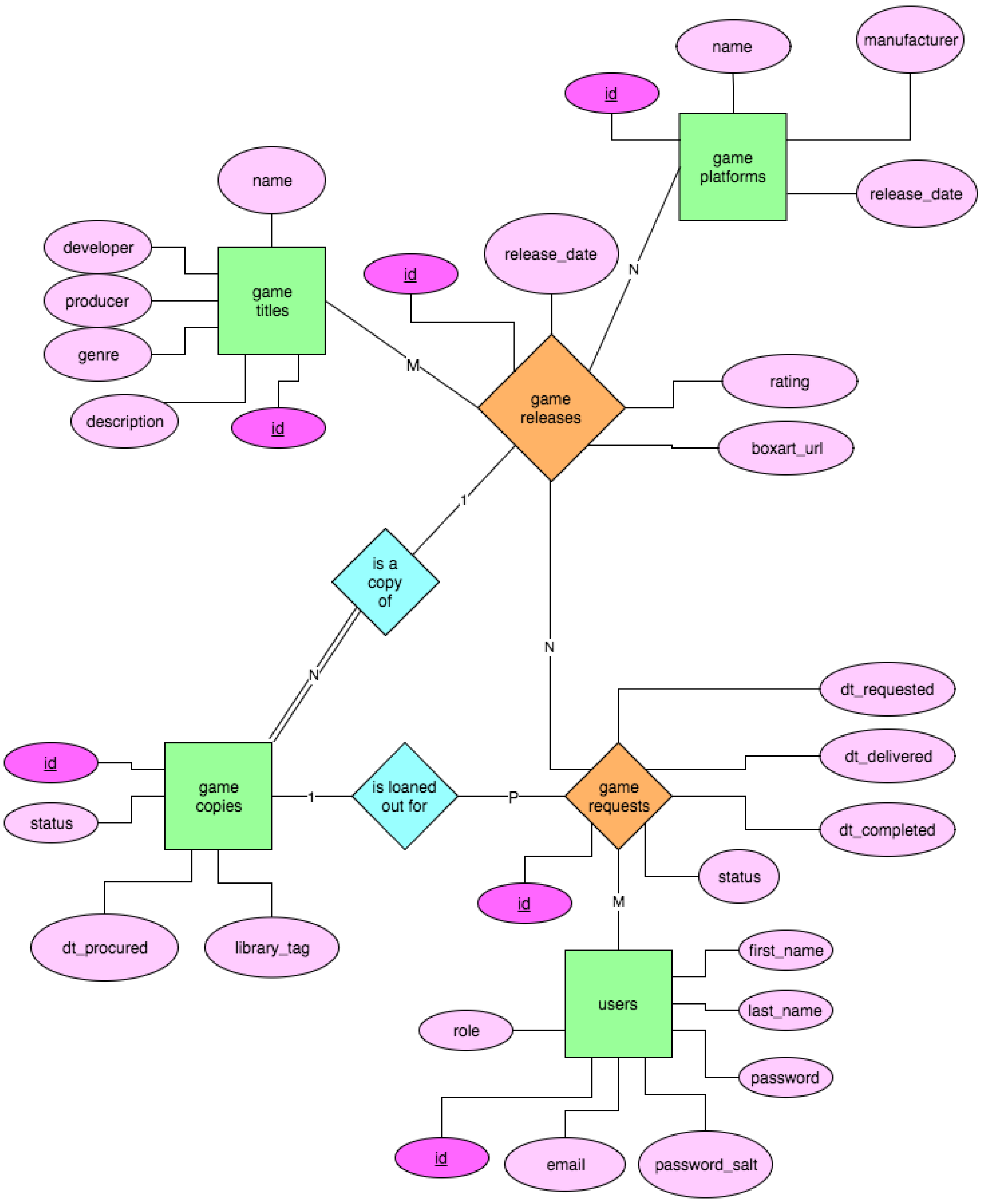
    - platform_ID: A foreign key linking a game release to the platform of that release; cannot be NULL
    - release_date: A date field representing the release date; cannot be NULL
    - rating: an integer field representing the review score of this release
    - boxart_url: A varchar string with a URL 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

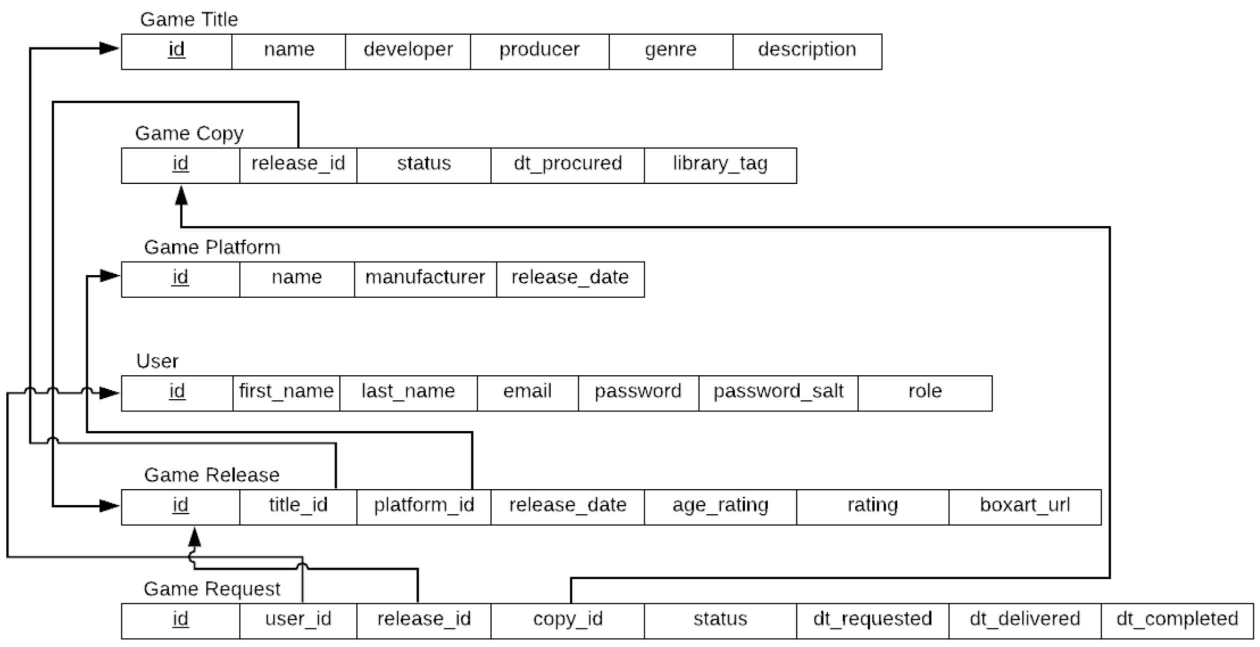
- **Game_Releases - game_titles, and game_platforms**
    - Many-to-many relationship involving game_titles, and game_platforms.
    - A Game_Title can have many game_platforms through game_releases.
    - A game_platform can have many game_titles through game_releases.
    - Contains additional data 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).

Entity-Relationship Diagram



Schema


Game Title
id
name
developer
producer
genre
description
Game Copy
id
release_id
status
dt_procured
library_tag
Game Platform
id
name
manufacturer
release_date
User
id
first_name
last_name
email
password
password_salt
role
Game Release
id
title_id
platform_id
release_date
age_rating
rating
boxart_url
Game Request
id
user_id
release_id
copy_id
status
dt_requested
dt_delivered
dt_completed