CSC 436 Assignment

Semester DB Project: Phase I

Overview

Your group will analyze suggestions and critiques, document specific revisions, address any identified weaknesses, and implement revisions of your database design. Through this process, you will enhance the quality and functionality of your project.

Part I: Review Feedback & Revisions

Carefully review the feedback provided during the Phase I presentations and peer reviews. Take note of any suggestions, critiques, or areas for improvement identified for your project

1) . Introduced a “Character” parent entity and moved Player/NPC/Creature under it.
   - Why: Feedback pointed out that Player/NPC/Creature being subtypes of “Prop” was confusing because “prop” usually means objects/environment assets, not characters.
   - How it improves the project: Makes the model semantically correct and easier to understand. Now Players, NPCs, and Creatures share common character-like attributes while keeping role-specific attributes in their subtypes.

Added clear identifiers (primary keys) to entities and corrected subtype key usage.
   - Why: Feedback noted that entities like NPC/Player/Creature did not have attributes that uniquely identify a record.
   - How it improves the project: Adds uniqueness and prevents ambiguity (e.g., two NPCs with the same race). In the specialization, subtypes inherit the parent key (Character uses characterID, Prop uses propID).

Added Character location tracking using IsAt relationship.
   - Why: In gameplay, characters exist in locations (players, NPCs, and creatures are somewhere).
   - How it improves the project: Supports note-taking and world consistency by letting the database answer “who/what is at this location?”

2) Work closely with your group members to discuss the feedback received and determine the best approach for making revisions.

Concern 1: “Player is not really a Prop.”

- Fix: Created Character as the parent entity and placed Player, NPC, Creature as subtypes under Character.

Concern 2: Missing unique identifiers for entity instances.

- Fix: Added explicit IDs for parent entities (EX: characterID, propID) and ensured subtypes inherit these keys.

Concern 3: ER specialization rules are stricter than OOP inheritance.

- Fix: Adjusted specialization so subtype meaning is precise: Player/NPC/Creature are true subsets of Character, and Item is a true subset of Prop.

Concern 4: Unclear data sources / how data will be collected.

- Fix: Data for the database will be manually entered by the Dungeon Master (DM) and participating players. The DM will primarily input world building data such as locations, NPCs, creatures, and props. Players may contribute character specific data, such as player character details, inventory updates, and notes from previous campaigns. Additionally, existing campaign materials and prior gameplay notes can be used as reference sources when creating new records. For the purpose of testing app functionality, we will provide some initial dummy information for the database. This information will be from a mix of our personal tabletop games and extra information generated by an LLM.

Part II: Implementation Of Revisions

Implement the identified revisions and improvements into your project. Ensure that all changes are accurately reflected in your project materials. Your group must submit the following revised project materials based on the feedback you received:

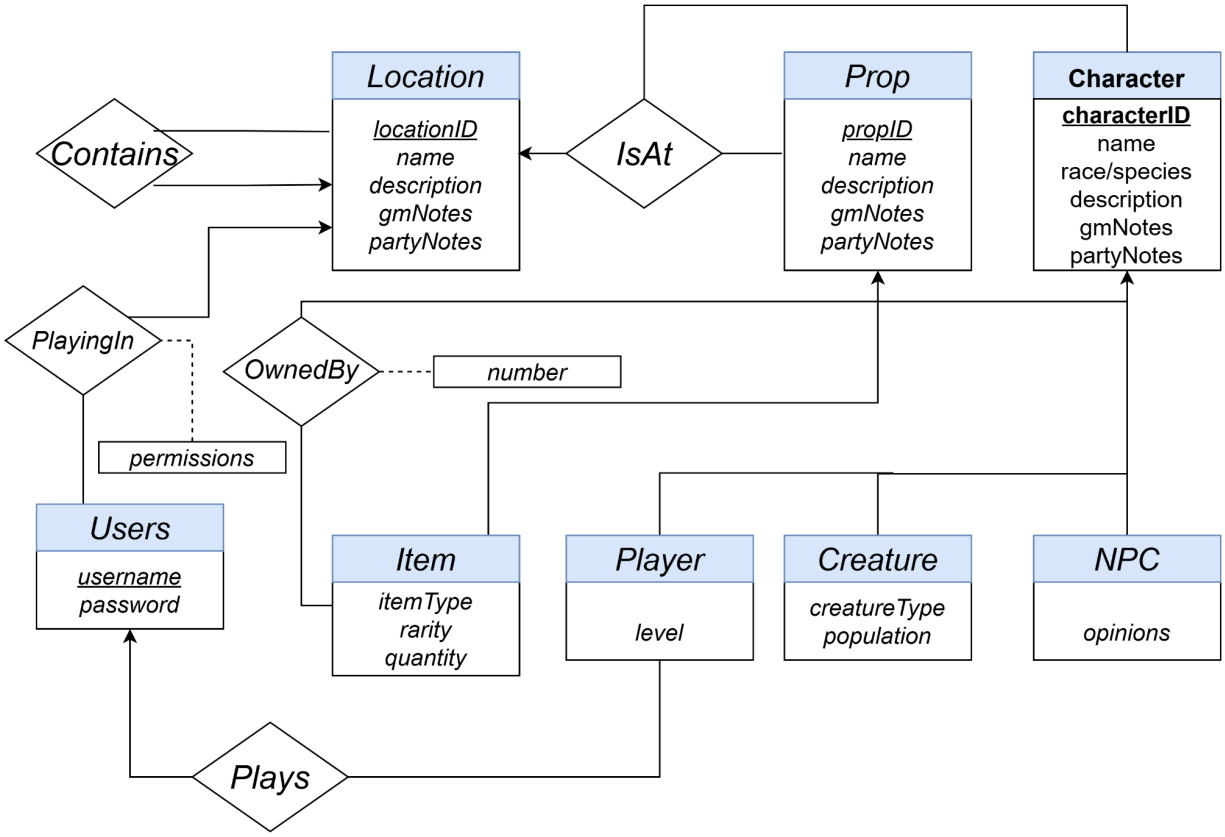
1. Updated entity sets and their attributes with primary keys underlined.

Users(<u>username</u>, password)
Location(<u>locationID</u>, name, description, gmNotes, partyNotes)
Prop(<u>propID</u>, name, description, gmNotes, partyNotes)
Character(<u>characterID</u>, name, race/species, description, gmNotes, partyNotes)
Player(<u>characterID</u>, level) (subtype of Character)
NPC(<u>characterID</u>, opinions) (subtype of Character)
Creature(<u>characterID</u>, creatureType, population) (subtype of Character)
Item(<u>propID</u>, itemType, rarity, quantity) (subtype of Prop)

2. Updated relationships and constraints.

Contains(Location, Location): one-to-many (a location can contain sublocations)
IsAt(Prop, Location): many-to-one (many props can be at one location)
IsAt(Character, Location): many-to-one (many characters can be at one location)
Plays(Users, Player): one-to-many (one user can play multiple player characters)
PlayingIn(Users, Location): many-to-one with relationship attribute permissions (controls what a user can view)
OwnedBy(Character, Item): many-to-many with relationship attribute number (how many of an item a character owns)

3. Updated E-R Diagram reflecting the changes made to the project structure.



4. Descriptions of the sources from which you intend to collect data for your database, possible high-level data modification operations, and the types of questions your stored data will be able to answer.

Data Sources

The data stored in the database will primarily be generated through manual entry by users of the system. The Dungeon Master (DM) will be responsible for entering and maintaining world building data, including locations, NPCs, creatures, and props. Players will contribute character specific information such as player character attributes, notes, and inventory updates, depending on their assigned permissions. For the purpose of testing, we will manually input example information from our personal tabletop games into the initial database. If even more information is needed for testing, we will use ChatGPT or another LLM to generate extra data for testing app functionality.

High-Level Modification Operations (CRUD)
The system will support standard create, read, update, and delete operations for all major entities. These include:

- Creating, editing, and deleting
  Locations and nested sublocations,
  NPCs, including updating their opinions over time,
  Creatures associated with specific locations,
  Props and Items, and assigning them to locations
- Assigning or removing Items owned by Characters and updating the owned quantity through the OwnedByrelationship

These operations allow the database to evolve as the campaign progresses.

Example Questions the Database Can Answer
The database design supports queries that assist both the DM and players, such as:

- What NPCs are currently located at a specific location?
- What creatures can be found at a particular location?
- Where is a specific character currently located?
- What items does a given player own, and how many of each?
- What props exist within a location?
- Which sublocations are contained within a larger location?
- What information is visible to a specific user based on their assigned permissions?

Submission

When you're finished, complete the following steps to submit your work:

- ☐ Export your document with responses as a **PDF file AND save** it **inside** your **documentation** folder. Refer to the following for documentation on how to do this:
  - Google Docs (*File → Download → PDF Document*)
  - Microsoft Word (*File → Save As / Export → PDF*)
  - Pages (*File → Export To → PDF*)
- ☐ Export your updated **E-R diagram** as an **image file** and save it **inside** your **documentation** folder. Be sure your file is clearly named (i.e., **er_diagram.png**).
- ☐ Upload all your changes to GitHub.

- ☐ **If you're using GitHub Desktop (GUI)**, complete the Uploading Changes (GitHub Desktop) section to upload your changes from your local device to GitHub.

- ☐ **If you're using Git (CLI)**, complete the Uploading Changes (GitHub CLI) section to upload your changes from your local device to GitHub.

ONE group member must paste the URL of your GitHub repository in the provided textbox in Brightspace. Click the blue *Submit* button to successfully submit your work for this assignment.

Grading Rubric

You can refer to the **Semester DB Project: Phase I grading rubric** given in Brightspace for this assignment to find details on how your submission will be graded.