Database Design: Project 1

Juan Sebastian Delgado

Victor Ramirez

California State University Monterey Bay

Database Description

The database is a design to be the backend of a basic game that allows the player to create a character, complete quests, and accumulate items. The system tracks the items the character acquires and their quantity, as well as the quests the character is currently on, or has completed along with when those quests were acquired and completed.

Data Interactions

The *character* table describes and basic information about the character including their current gold and experience. The character must be a predetermined class as defined in the *class_type* table which will determine the characteristics or abilities of the character. Equally, the *quests* table has a dependency on the *quest_type* which determines the type of quest for both tracking and allocation purpose. The *items* tables has a dependency on the *item_types*.

Because multiple character can acquire multiple items or quests, there are two auxiliary tables to help track of an individual character collection of items and quests. The *quest_log* table has a dependency on both the *quest* table, and the *character* table as a many-to-many relationship of the possible quests the user may acquire. It's important to note, this table allows for multiple of the same quests per character to allow for repeatable quests. The *inventory* table tracks what the user owns and creates a many-to-many relationship between character and items. Because when the user acquires an item is not necessarily important and the user may acquire many of a single item, the table tracks a third column of quantity and restricts the relationship between item-character as a unique key.

Data Normalization

The system is as 3NF which aims to reduce all redundant data. The *classes*, *item_types* and *quest_types* tables remove otherwise repeated, predictable, standardized data from their corresponding tables. For example, if the `mage` class was to be renamed to `wizard`, the change is made to a single row in the *classes* table, and the characters playing the game would not need to update their information. Expanding to the system to include other classes, would only mean adding a new row. And additional benefit to normalizing both *_*types* and *classes* data, is that it allows to validate only allowed data can be inserted into the main tables, and classes cannot be deleted if a character is currently allocated to the class.

Because such data is normalize, we can have additional information about classes or items that is global to the corresponding category. For instance, *classes* may determine a main attribute that favors each class such as charisma or intelligence, a restriction on weapons such as swords or staves. Because those changes affect every character belonging the class, the normalization of such data means updating the attributes will not incur in penalties or risks by updating additional tables.

The *quest_log* and *inventory* tables also separate otherwise the duplication of items and quests as they need to be allocated to a character. If any of the information from a quest changes, the character currently owning the quest does not need to be updated.