Database Design: Project 1

Juan Sebastian Delgado

Victor Ramirez

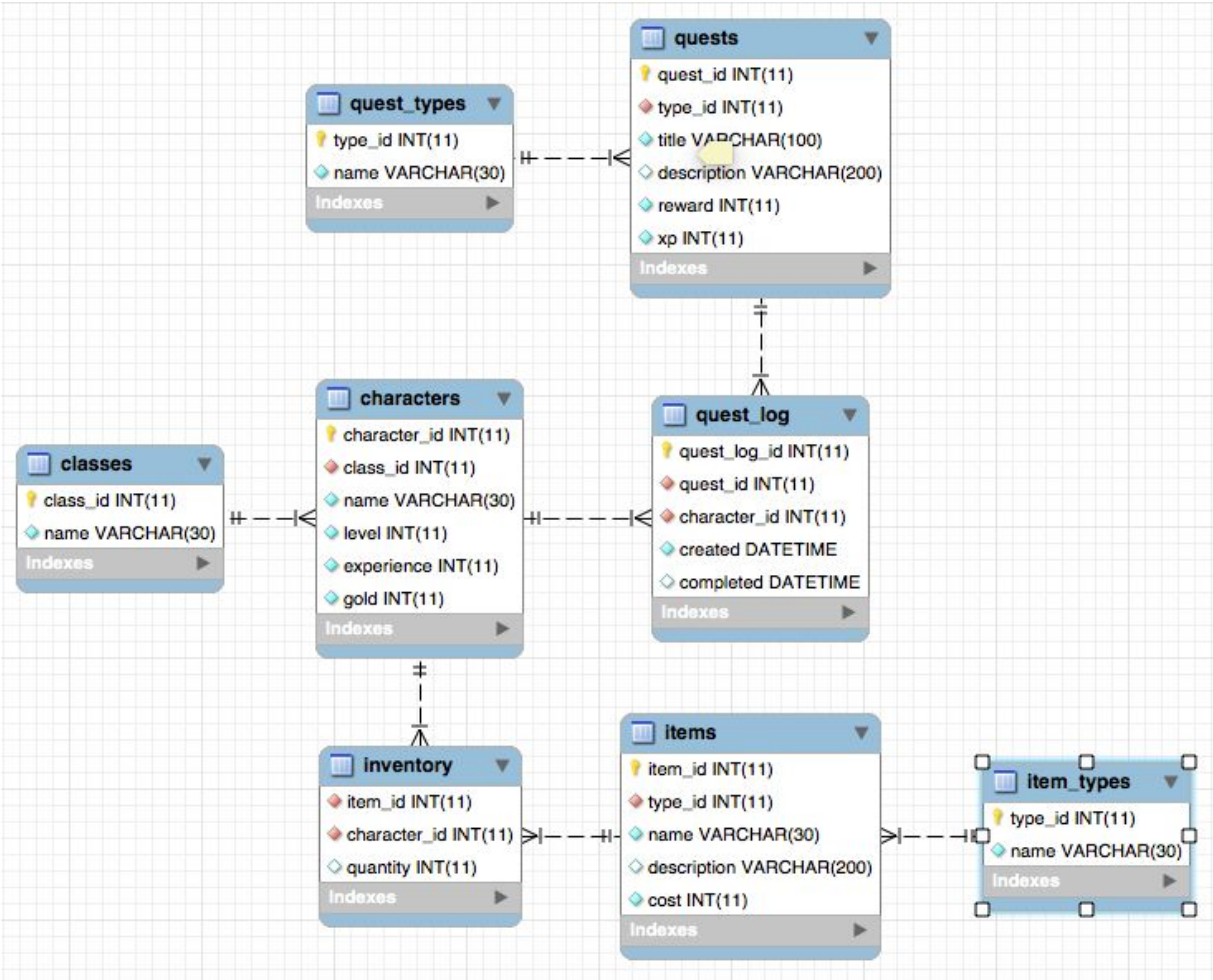
California State University Monterey Bay

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. The database design focus is around expandability, maintainability and high amount of writes. The player should be able to get large amount of items, and complete quests. The tables are designed to hold only the minimum amount of "variable" data required to perform and operation. Variable, and repeatable data is linked as foreign key to the appropriate table, allowing system administrators to add, modify, or remove such variables are required, with virtually no impact to the existing structure.

There are two main customers for this database: The game designers, who create quests, and items, and are ultimately the managers for the system. This users will be directly interacting with the tables mentioned above creating content for the game. The customers for this data are the players who have the ability to create a character, buy and sell items and, get and complete quests. Characters should not have the ability to change quests or items. Instead they should read the data from the table, and interact with the quest_log, and inventory which maintain the relationships between characters and content. Although the system does not support this distinct roles right now, the existing infrastructure allows for the roles to be included easily. The EER Diagram for the table relationships is described in Figure 1.

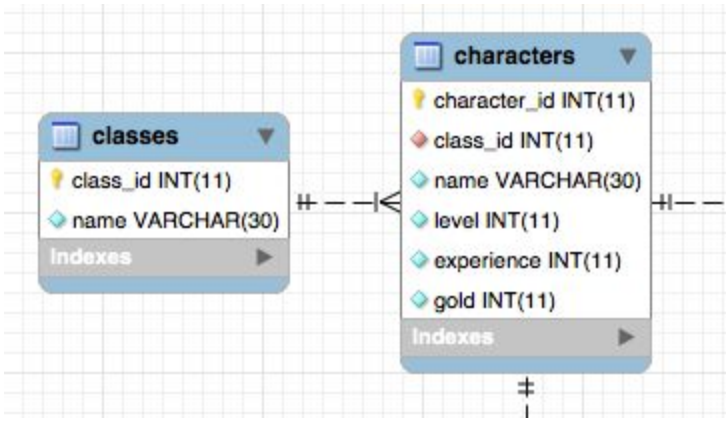
Figure1



Data Interactions

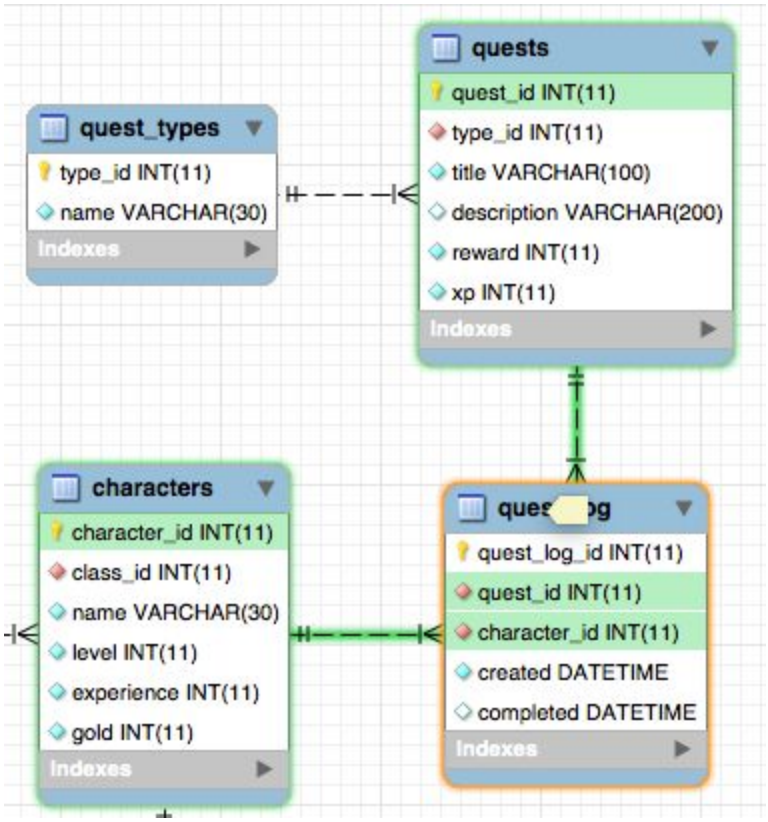
The *character* table describes and basic information about the character including their current gold and experience. The character must be a predetermined class as defined in the *classes* table which will determine the characteristics or abilities of the character (Figure 2). Equally, the *quests* 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*.

Figure 2



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 (Figure 3). 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.

Figure 3



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.

Data Warehouse Description

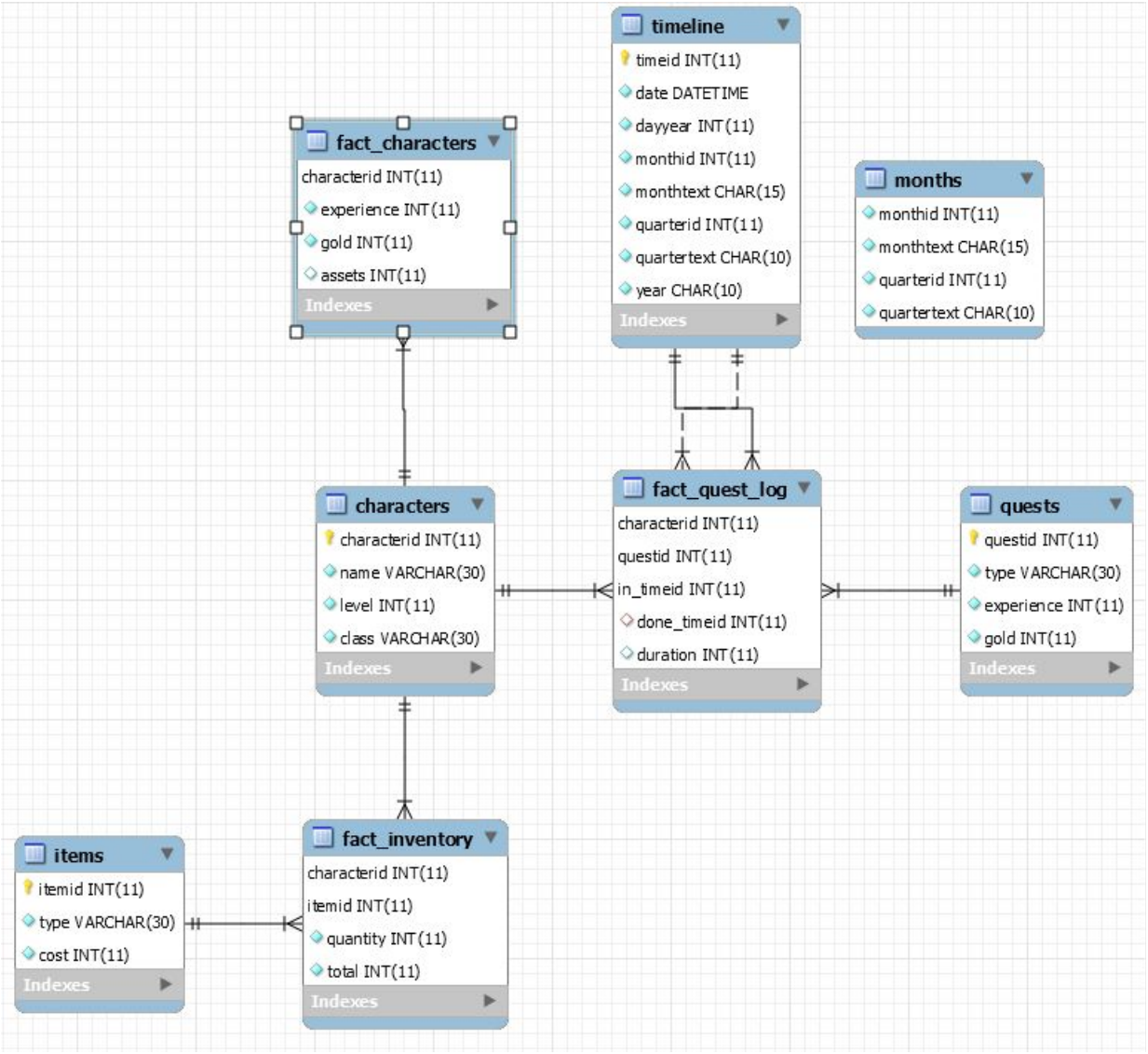
The database warehouse follows a star schema design. The main goal of this database is to provide the live operators of the game with the information required to run the game properly, which should include information about the game economy, as well as patterns of behavior for players. To get insight into the data, we have split the the information in to the following tables:

Table Name	**Type**	
timeline	Dimension Table	Describes information regarding time.
characters	Dimension Table	Describes the characters
fact_character	Fact Table	Contains basic statistics about the character such as amount of gold they have.
quests	Dimension Table	Describes the basic information for each quest.
fact_quest_log	Fact Table	Contains the relation between a character, and the historical data about the quests it has acquired.
items	Dimension Table	Describes the basic information for each item.
fact_inventory	Fact Table	Contains the statistical data of the relation between a character and the items it owns.

Data Normalization

The schema follows the pattern of abstracting the dimensions, from the facts. Because of this, the data is denormalized for the dimensions to keep it the aggregations of such data much simpler. The facts tables, are then only dependent on a few foreign keys against the dimensions that are important. This allows us to gather better statistic information by joining the facts against the dimensions. The final result for this is described in the EER model in Figure 4.

Figure 4

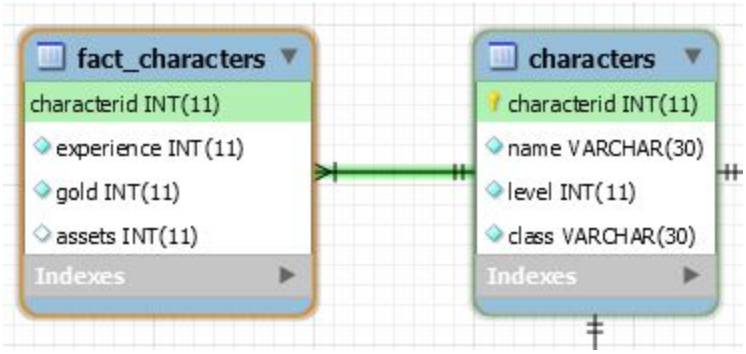


Data Interactions

There are three distinct star schemas that can produce interesting data for the game operators. The main schema is the relation between the character and his facts. The character table describes the basic data for a character, while the facts table contains the statistics about the character that can be useful (Figure 5). The second schema has a

relation between a character, and the quests. The facts table in this case has multiple dependencies between the timeline of when a quest was acquired, and completed, as well the character the quest themselves. We can run multiple aggregations describing when users are active, which quests are most popular at different levels, or how much a given quest has paid out across the character dimension (Figure 6).

Figure 5



Finally, the inventory fact table describes the relationship between the character dimension, and the items dimension. Because we do not keep track of when the item was acquired, we don't have to worry about the time. However, it still produces interesting data regarding patterns of behavior for characters. In this case, we can get insight in how much gold is available in the economy, which items are most popular at different levels, or the type of items a different from a given class is likely to purchase (Figure 7).

Figure 6

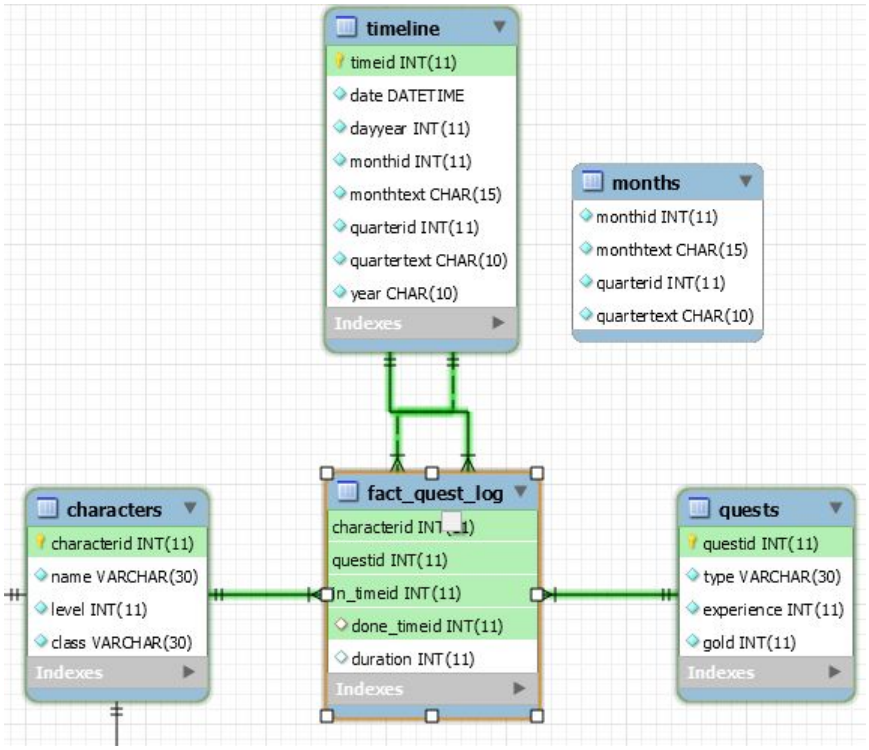


Figure 7