A database of games

Cody Dance-Wilson, Alan Seed

Introduction

In our increasingly insular world, board games continue to be a means to spend a pleasant time interacting with a group of friends in a relaxed social setting. Not all board games are equal, and not all players enjoy the same types of games. The objective of this project is to build a database of board games that can be searched to find the games that meet a set of player requirements. Selection criteria include the type and mechanics of the game as well as the duration of the game, number of players, and suitability for a specific age group.

Extract

The Kaggle site includes a dataset of board games at https://www.kaggle.com/mshepherd/board-games.

These data have been scraped using scaping code that can be found at https://gitlab.com/recommend.games/board-game-scraper which sources data from

1. Board Game Atlas (`bga`)
2. BoardGameGeek (`bgg`)
3. DBpedia (`dbpedia`)
4. Luding.org (`luding`)
5. Spielen.de (`spielen`)
6. Wikidata (`wikidata`)

The following files were downloaded from the Kaggle site:

bga_GameItem.csv
bgg_Category.csv
bgg_GameFamily.csv
bgg_GameItem.csv
bgg_GameType.csv
bgg_Mechanic.csv
bgg_Person.csv
bgg_Publisher.csv
dbpedia_GameItem.csv
luding_GameItem.csv
spielen_GameItem.csv
wikidata_GameItem.csv

The data sets from Board Game Atlas, DBpedia, Luding, Spielen, and Wikidata did not have data for the ranking of the game or the recommended number of players for each game. Furthermore, the Lunding and Spielen data sets were mostly for non-English games and therefore it was decided to only use data from the BoardGameGeek web site.

Transform

The goal of the database is to assist with the selection of board games that meet several selection criteria. Following a discussion, the following criteria were selected as likely to be of interest

- Number of players
- Age of the players
- Duration of the game
- Complexity of the game
- Played by teams or individuals

The columns that were selected are shown in Table 1.

Table 1 Columns inserted into the database

Column	Description
Name	Name of the board game
Year	Year that the game was published
Game Type	Game type e.g., war game, children, party, strategy etc
Minimum number of players	Minimum number of players possible for the game
Minimum number of players (best)	Minimum number of players needed for an enjoyable experience
Maximum number of players	Maximum number of players possible for the game
Maximum number of players (best)	Maximum number of players for an enjoyable experience
Minimum age	Minimum age of a player in the game
Minimum time	Minimum duration of the game
Maximum time	Likely maximum duration of the game
Category	Theme of the game e.g., science fiction, political, trains, prehistoric, horror, etc.
Mechanic	How the game is played e.g., turn order, interrupts, scenario, player elimination, etc.
Cooperative	Play as individuals or in teams
Rank	The ranking of the game in a survey
Number of votes	Number of votes in a survey
Average Rating	Average rating from a survey
Complexity	Game complexity score from a survey

The selected CSV files were read into Pandas using the “read_csv” function to create Pandas data frames. This resulted in a data frame for each table in our ERD, board games, mechanics, categories and game types.

The board game data frame then had the unnecessary columns removed, this was done by listing all columns from the data frame, then creating a list of only the column names required for the final table. The data frame was then set back on itself to select only the columns required, and a copy created with “.copy()” notation to create a new object and not a slice of the original data frame.

All imported tables then had their columns renamed to suit the output for the final table. With “.rename()” the ‘bgg_id’ column in each of the ‘category,’ ‘mechanic’ and ‘game type’ data frames were renamed to be ‘category_id’, ‘mechanic_id’ and ‘type_id’ respectively for simplicity of reading the database. These tables also had their ‘name’ column amended to be ‘category_name’, ‘mechanic_name’ and ‘type_name’, again to make the database easier to read and to avoid

confusion as code was progressed. The board game data frame had the ‘bgg_id’ column renamed to ‘id’.

With the focus on year, category, mechanic and game type for the database, the decision was made to drop all games which had a null value in these columns. The “.dropna()” function was used in Pandas, with the subset of these columns, and the data frame was again set back on itself. This resulted in a reduction of the database items from 102,325 rows to 18,768 rows of data.

During inspection of the data, it was observed that the ‘mechanic’, ‘category’ and ‘game_type’ columns of the board games data frame contained a comma-separated list of values. It was determined, given time constraints on this project, to reduce this list down to the first four-digit number in each cell which would then be the value linked to each of the reference tables. This was done by setting each column value back on itself and making that value the first four items in the string value of each cell. This resulted in a clean column of four-digit numbers, which all corresponded to the reference tables.

Each data frame was inspected with Pandas using the “.info()” function to ensure that data types were correct for import to PostgreSQL. The smaller tables for categories, game types and mechanics needed no datatype modification as they were already in the correct format, but the board game table required adjustment.

The decision was made to transform the ‘cooperative’ column from a null value or a “1” to be Boolean as the first transformation. Using the “.astype()” function in Pandas, this was changed to Boolean, with null now being “False” and the “1” values being “True” in the transformed column.

It was also required to transform ‘category’, ‘game_type’, ‘mechanic’ and ‘year’ columns from the object datatype into integers for the final output. To avoid Python errors with attempting to set a null value to an integer, the datatype change was achieved using a “.loc()” function which targeted only the values in each cell which were not null and then changed their datatype to “int32” using the “.astype()” function. This function was run on each column which needed to be set to an integer, and it was determined that all other columns would be left with their original datatype for import into the database.

Before exporting to PostgreSQL for the final database, the ‘id’ column from the board game data frame was set as the index of the data frame. To ensure there were no duplicate ‘id’ values and clean-up the final appearance of the dataframe, the index column was then reset using “reset_index(drop=True)” to ensure it began at zero and counted upwards sequentially.

Load

The data from the BoardGameGeek site were arranged into a set of CSV files, with separate tables for the game category, mechanic, and game type. This could have been simplified since these tables consisted of two columns, the ID and the name. These data are more suited to a SQL style data base since they are already arranged into a schema and the flexibility of a less structured database like Mongo would not be an advantage. Therefore, the data were loaded into a PostgreSQL data base.

The schema for the PostgreSQL database was created using SQL and laid out the following tables, primary keys and foreign keys:

- type
    - Primary Key: ‘type_id’
- categories
    - Primary Key: ‘category_id’
- mechanics

    - Primary Key: ‘mechanic_id’
- board_games
    - Primary Key: ‘id’
    - Foreign Keys:
        - ‘game_type’ referencing ‘type_id’
        - ‘category’ referencing ‘category_id’
        - ‘mechanic’ referencing ‘mechanic_id’

The columns in the PostgreSQL database had the datatypes of each column set out following the ERD seen in Figure 1.

Each of the data frames created in Pandas were loaded into PostgreSQL using SQLAlchemy to create a connection with the PostgreSQL local host. The data was appended to the tables created using the SQL schema, setting the index column of the board_games table to align with the ‘id’ column in the PostgreSQL table.

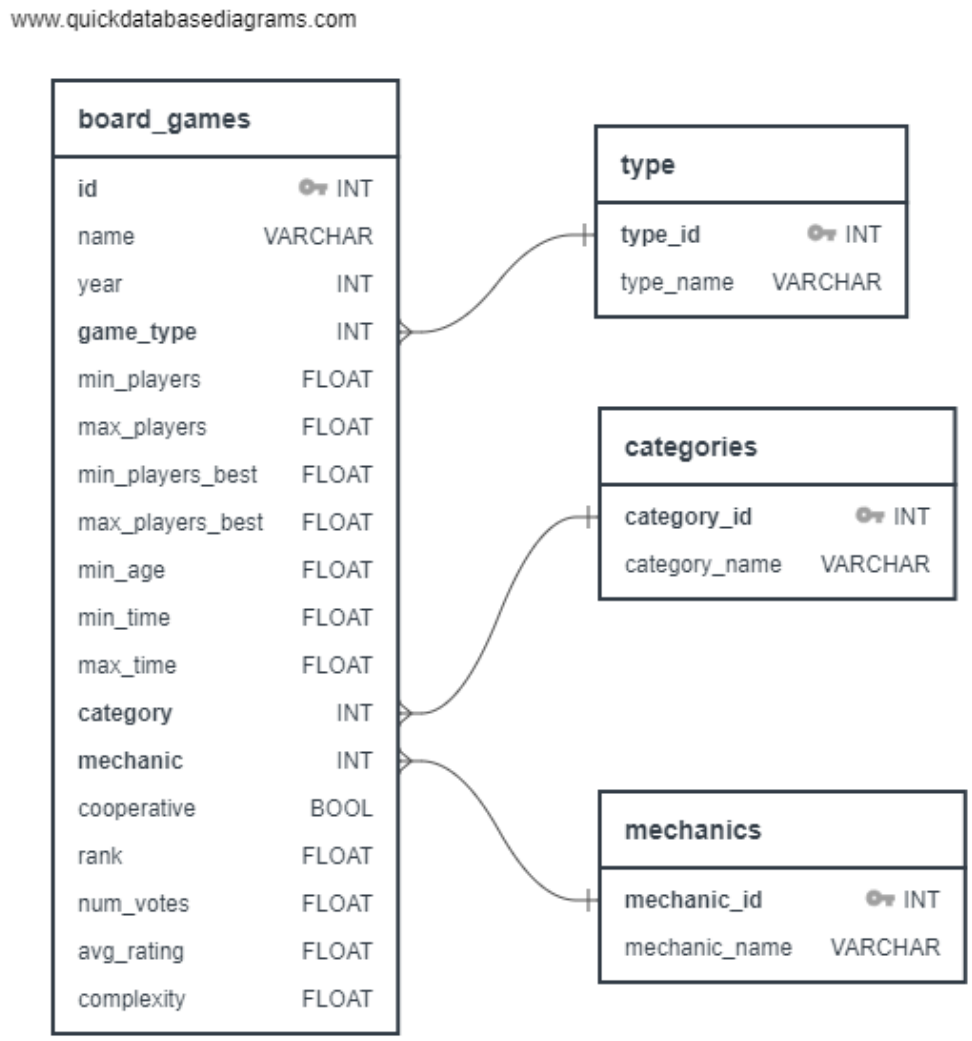

Figure 1 Design of the database