Project Team: Eric Studley; Jake Dec; Tom Diaz
Team Name: Gold Team
CS5200 DBMS Project - Game Ranker
PM6: Conclusion and Final Presentation; Group Submission
Github (all code, data, workspaces): https://github.com/JakeDec/CS5200_Project

Did you fulfill your value proposition?

"Game Ranker is a database of video games with recommendations, reviews, average time played, sales data, and availability per console. This database offers recommendations of new games for players who want to play "good" games of a specific genre; games of over/under a specific time played; and games with certain keywords in reviews who are unhappy with the massive selection of games with questionable quality or replay value. Game Ranker also allows developers to investigate if the length of a particular game or genre has an effect on game sales."

The result of our team's work was exactly as proposed. We used three primary sources of data for this project; a database of video game information from vgchartz, critic and user text reviews and number scores from Metacritic, and time played per game by users on Steam. Video game sales data and general GDP data was added as a data warehouse in P5. The Steamworks API was accessed to generate more information about users and generate Steam user names from the provided Steam ID surrogate key. Critic scores ranged from 1-100 and user scores from 1-10 so user scores were scaled accordingly on input into the database.

Goals vs Accomplished

Players who want to play "good" games of a specific genre.
PM3 Q5 - What are the top three games of a given genre from both critic and user scores?

Players who want to find certain lengths of games.
PM3 Q9 - What's the average playtime for the most popular genres?

Identify games with questionable quality or replay value.
PM3 Q2 - What games have a large number of purchases with little/no playtime?

Identify if game length, genre, and effect on game sales.
PM5 ETL/Chart 2 - Game Platform Sale Trends

Recommend games of a specific genre available on platforms the user owns.
The database could be queried by a list of PlatformNames to get PlatformIds and find the highest rated games that contain results in the GameOnPlatform table.

These goals can be displayed via web browser in formatted tables using our created JSPs. Queries related to sales data were demonstrated through the data warehousing assignment and queries from the ETL.

What went well?

Accessing the Steamworks API was much easier than originally thought and it allowed us to get the user names for all of the users in the Steam data source which was indexed by Steam ID. This allowed us to cross-reference users who had both a Metacritic account and Steam account with the same user name.

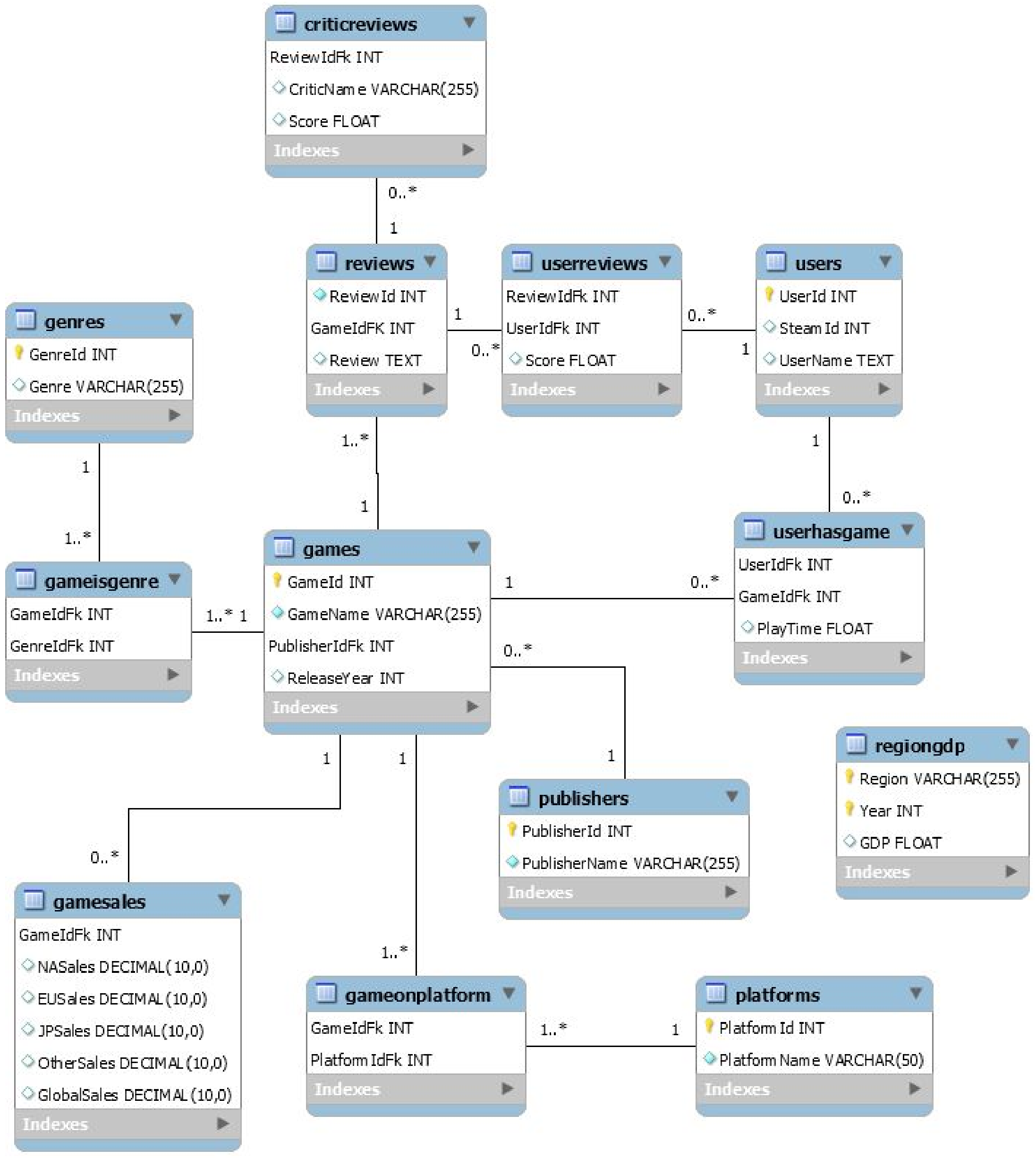
What would you do differently?

Some of the raw data was difficult to work with initially and required some light python scripting to correct. Genre information came packed as a single field separated by semicolons. Some of the developer names in our data set were not formatted correctly. If possible it might have been better to scrape some data ourselves; the free Steamworks API has a 100K requests a day limit which is fairly high.

What do you plan to do next?

Continue to utilize external APIs, such as Steamworks API, to pull an input user's information from Steam such as games owned, friends, and other information to personalize and automatically populate data for a user.

Final UML



Description of Changes to UML

No updates were made to the initial UML during the assignment. One multi-value data source was normalized in PM3 when it was noticed while doing the queries.
For PM5 to warehouse game sales and global economic data, the GameSales and RegionGdp tables were added to store the additional data from these two sources.