Personal Data Extraction for STEAM

CS 541 – DBMS Spring-2014

Rahul Kadukar,
Kalidas Nalla,
Siddharth Modala.

1. Introduction

Steam is an entertainment platform to play and connect with users of the gaming community developed by Valve Corporation which provides users with installation and automatic management of games across multiple platforms apart from other features like cloud saving, in-game voice and chat functionality. Most of the games offered support Steam Play; a cross-platform multi-player platform which allows users to buy games once and access the games from different operating systems. The application provides a freely available application programming interface (API) called SteamWorks, that we took advantage to integrate many of the Steam's functions within our project.

It has been estimated that over 75% of the purchased PC games are downloaded from Steam and by early January 2014, Steam has over 3000 games available and 75 million active users.

2. Motivation

Though Steam is the biggest gaming platform on the internet, we were surprised that users were not provided with good recommendations that would enhance their gaming experience. Steam has a recommendation system that is based on the overall popularity of the game and doesn't take into consideration individual preferences. We wanted to help the users with this shortcoming by providing additional intricate details and providing suggestions on the games which they should buy based on the games that they own and the ones that their friends own in order to compete with a bigger and closer community.

3. Implementation

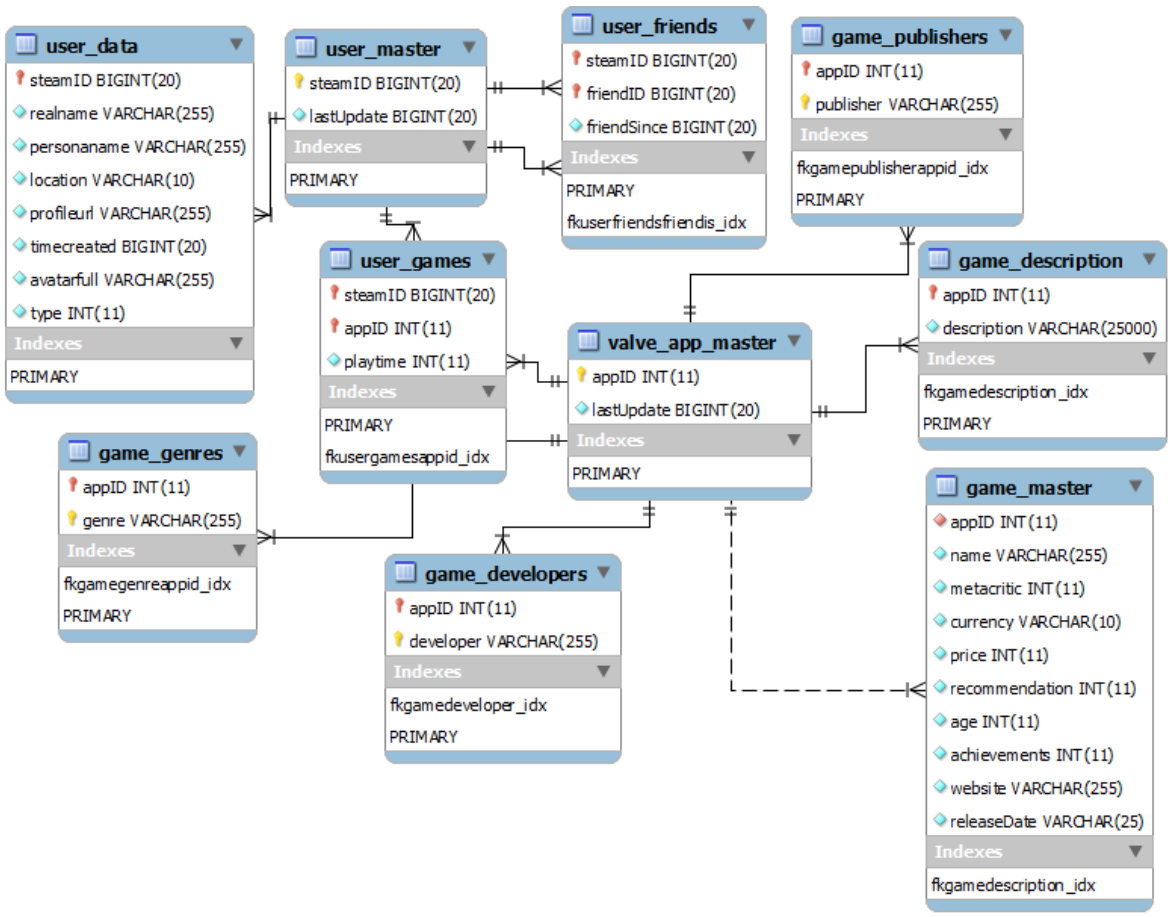
3.1.Data Gathering and storage

Using the steam works API we extracted the information of individual users. We used a recursive approach to gather data. We started with a seed user who had the highest number of friends on steam. We extracted the following information about each user.

1) Personal Information like name, location, steam ID, profile URL, steam alias etc.
2) Friends steam ID's.
3) Games that he owns along with playtime of each game.

Then we would recursively go on to each of his friends and extract their information. Using this approach we collected approximately 35000 user's information. Apart from the users information we also gathers information regarding a game.

Shown below is the image of the database schema that we have used to store user information.



3.2.Back end

We used PHP to write our backend logic to connect to MySQL database. We used some complex SQL queries to extract summarized data. Sometimes when the queries became too complex and took lot of time we had split the processing of the data into two parts. One in SQL queries where we would get a high level summarization and then in Java Script where we summarized it further to improve performance.

3.3.Front end

We used HTML, CSS, Java Script, JQuery to build our front end. We used a charting API called HighCharts to visualize the data.

4. Features

4.1. Popularity of an app based on genre.

Apps often have multiple genres and it impacts other stats. For example, say we want to calculate the fraction of time we spend playing different genres, like Action games vs. Adventure games. What if there is a game which has both Action and Adventure in its genres? Here are the two simplest approaches that we are considering:

1) Split evenly: Split the playtime into equal-sized chunks for each genre that an app belongs.
2) Over count: Count the playtime fully (100%) in each genre that an app belongs to.

We are going to provide a toggle whereby the user will be able to view the data in one of the two views and hence can see the data as they see fit. Analytically they represent the same information and the decision to view it is left to the user.

4.2. Value of an app by playtime

Say we want to figure out which game we should purchase. If we look at the games our friends own we can predict the dollars/hours based on the current prices and the playtime of our friends. However, maybe we have some friends who play Steam games 100 hours a week, and some who only play ten. If we calculate the value by raw playtime, the players who use Steam 100 hours a week can skew results. If we calculate the value by fractional playtime (aka "indexed"), the players who use Steam 10 hours a week can skew results.

The data can be shown in 2 ways as shown below and the user can then view the data as per the requirement, and analytically the two views are equivalent.

- **Raw**
  Formula:

$$\frac{\text{Current price of game}}{\text{Friend's total playtime for this game}}$$

Example: Of all your friends, the total play time on Game A is 250 hours. The current price is 25.00 USD.

The raw score is $25/250hr = $0.10 per hour

- **Indexed**
  Formula:

$$Price\ of\ game\ *\ (\frac{Friends\ total\ playtime\ for\ this\ game}{total\ Steam\ playtime})$$

Example: Of all your friends, the total play time on Game A is 250 hours. However, their total lifetime Steam play time adds up to 2500 hours. So, their fraction of time played on Game A, out of all Steam play time is 10 percent. The current price is 25.00 USD.

The indexed score is $25 * (250/2500) = $2.5/% playtime.

The above two formulas performed poorly with the data we had as for most of the games we dint had the actual price of the game. i.e the price which it had at the time of release of the game. Over a period of time most of the games have a decrease in their price which leads to uneven results.

4.3. Recommended games based on his friends games.

We provide game recommendation based on the most played games among his friends. This recommendation is calculated based on the total play time of the games owned by his friends. If a game is owned by multiple friends then the play time of the game is the individual playtime added together. Then this final set of games is subtracted from the games owned by the user. The top 10 games in the resultant set are shown as recommendations.

4.4. Recommended games based on his region.

Similar to the above recommendation we summarize the data based on the users location and using the same approach as above we recommend the top 10 games that are most played in his region.

4.5. Most played games by the user.

This is just a visualization of the most played game by the user sorted by the playtime.

4.6. Most games played by user's friends

This is a visualization of the most games played by his friends summarized based on the game's playtime and shown as a bar graph.

4.7. Most played game in his region.

This is a visualization of the most played games in his region summarized based on the game's playtime and shown as bar graph.

5. Challenges

- Initially each member of the team tried to make API calls to get user data from the Steam database and then merge the dataset to get a consolidated and comprehensive set of user profiles which was to be used for the personal data analysis project. While integrating the datasets we found a huge chunk of our dataset to be inconsistent; filled with redundant values. We later realized that, this was due to the reason that if any two of us have extracted the same user profile, we amass a large chunk of our data to be inconsistent as the API works in such a way that a user is fetched and then the profiles of the users friends are fetched next. Hence if we get the same user by any chance, their friend's data and so forth the process repeats. We overcame this problem by fetching user profiles and storing it in our database which was handled by a single person of the group rather than each of us collecting data and merging them.

- Since we make repeated calls to the API and were able to get user profiles of around 35,000 people on Steam and our database has accumulated around million records, the queries took a lot of time to execute. We overcame this problem by making use of a dedicated RAM of 8GB for our project. RAMDISK software was used to partition and allocate space.

- Some of the queries took lot of time hence we had to split the processing of the data into two stages. One where we extracted only a high level summary of the data then another where we further summarized the data using JavaScript code.

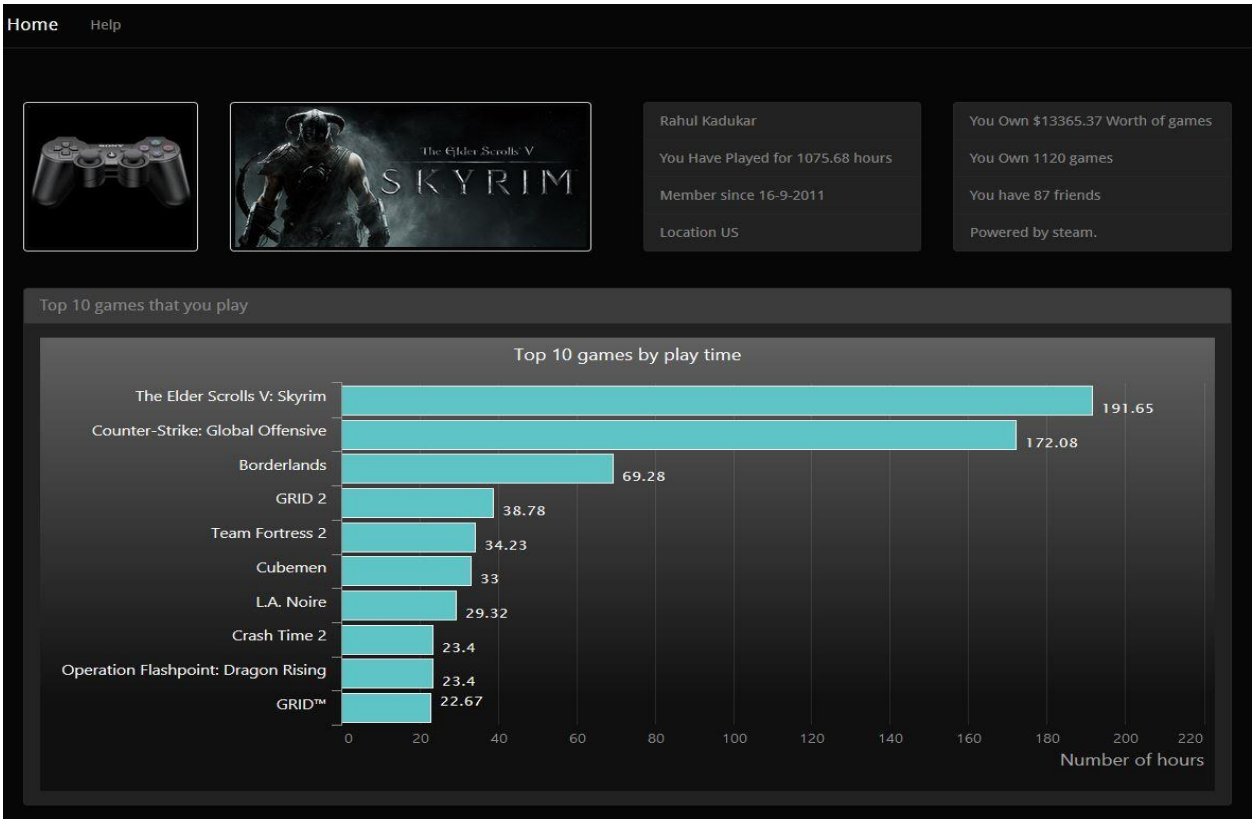
6. Conclusion and Future Work

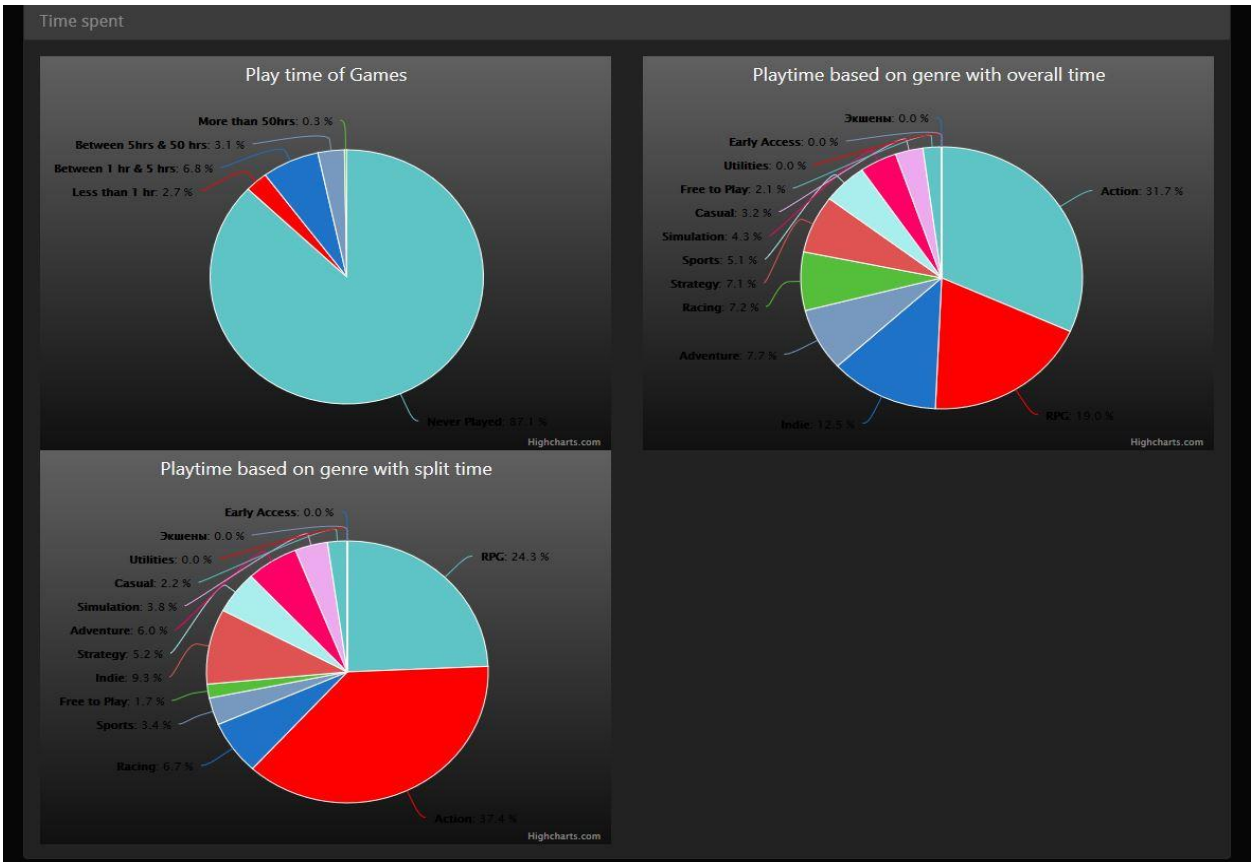
We believe our project has a very high scope, we have proposed the game recommendation system for a user by analyzing the playtimes of a user's friends and the most popular games in the user's country. We plan to extend our recommendation system by allowing the user to have more flexibility in choosing the games by including better recommendations.

In future we would like to make the following changes to the application for better results.

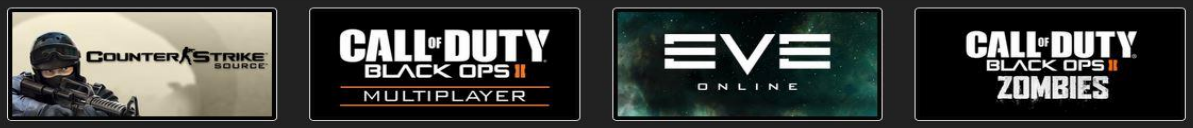
- We would like to make use of a NOSQL database like 'Mongo DB' for further enhancements as it provides us with additional flexibility of working directly with the JSON data.
- Our experience with gaming has led to the understanding that the popularity of the games among the user's friends is more important than the popularity considering the internet community as it gives more options for the user to connect with his friends and improve his gaming experience.
  We plan to extend our recommendation system by allowing the user to have more flexibility in choosing the games by including recommendations based on
  - Playtime of the games by the male and female community.
  - game genres
  - free/paid games

7. Screenshots

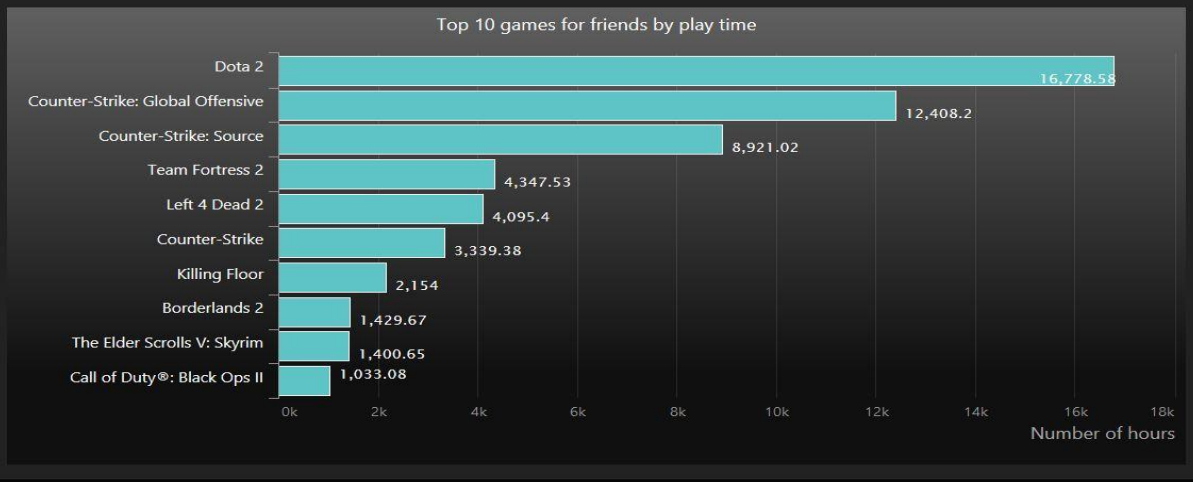




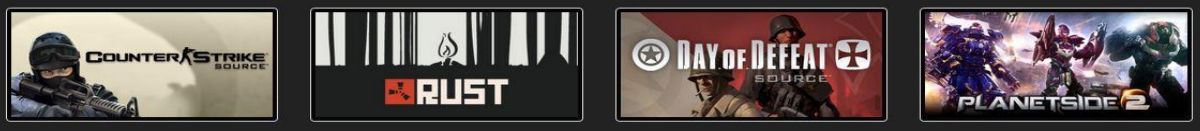
Recommended games to buy(Suggestions from friends).



Top 10 most popular games among you friends



Recommended games to buy based on country



Top 10 most popular games in your country