Video Games

Team Name
Isaac Kim
Will Poirier
Gabe Rippel
Alex Denny

November 2, 2024

1 Introduction (Phase 1)

The video games database project will be a command line tool from which users with will be able to execute commands that allow access, modification, or analysis of the data via certain privileges. Data from the Steam database will be predominantly used so any analysis will be valid as if you were asking questions about Steam video games.

We are planning to use Python with the psycopg2 PostgreSQL adapter. The sample data that will be used for analysis will be aggregated from multiple datasets including the Steam database and free datasets on Kaggle. The team members will for the most part, concurrently work on every phase and section of the project together. Code writing and report writing have all been divided up between members who communicate their plans and intentions before beginning to work so that there is no confusion or technical debt accumulated later on by decisions made early on. All the project code and documents are hosted on GitHub where members have access.

2 Design (Phase 1)

2.1 Conceptual Model

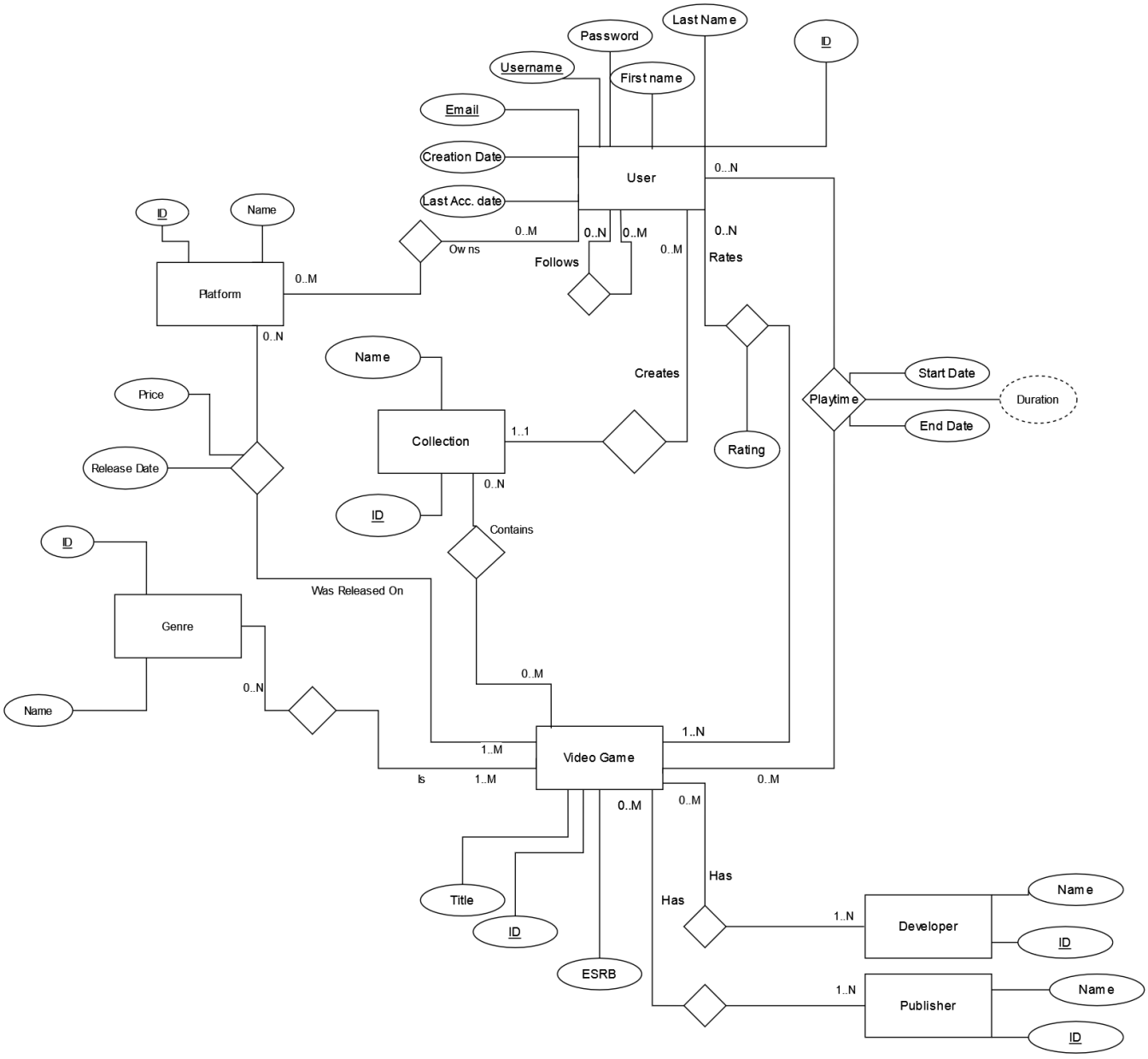

Figure 1: EER Diagram (PNG)

We originally had Collection as a relation, but determined that it has to be its own entity, which is instead related to both User and Game. Similarly, since Genres can be the same across multiple games, we chose to make it its own entity (as opposed to a string field of Game). We also chose to make Platform its own entity, so we could more easily identify which games a user can play, given the platforms they own. We determined that a Publisher and/or Developer need not have any games (for example, a new studio may not have released any games yet), and that Collections need not have any games either (for example, a new collection made by a User will start empty). We had a back-and-forth about whether to store Start Date/End Date or Start Date/Duration for Playtime; in the end, we determined that Start Date/End Date was better, to help with data entry (most people don't keep track of how long they play – just when they start and stop playing).

2.2 Reduction to tables

```
User     (ID, Username, Email, Password, First_name, Last_name,
            Creation_date, Last_access_date)
Following (Follower_ID, Followee_ID)
Genre (ID, Name)
Videogame (ID, Title, ESRB)
Developer (ID, name)
Publisher (ID, name)
Playtime (User_ID, Game_ID, Start_datetime, end_datetime)
Release     (Platform_ID, Game_ID, Price, Release_date)
Publishing (Game_ID, Publisher_ID)
Developing (Game_ID, Developer_ID)
Platform (ID, Name)
GameGenre (Genre_ID, Game_ID)
Collection (ID, Name, User_ID)
Rating (User_ID, Game_ID, rating)
PlatformOwnedByUser (User_ID, Platform_ID)
GameInCollection (Game_ID, Collection_ID)
```

After we finalized our EER Diagram, creating the reduction to tables was fairly straightforward. We made all of the Many-to-Many relations their own tables (Following, Playtime, Releases, PlatformOwnedByUser, Publishing, Developing); as well as some One-to-Many relations (GameInCollection). Other One-to-Many relations appeared as foreign keys in entity tables (Collection having User ID, GameGenre having Game ID). Otherwise, we copied the fields from the EER Diagram entities to similarly-names tables in the reduction to tables, with the notable exception of Playtime not having a field for Duration.

2.3 Data Requirements/Constraints

Game ESRB rating must be one of (E, E10+, T, M, Ao, RP, RPm). Ratings must be between 1 and 5 (inclusive). User Creation Date and Last Access Date must be valid dates, as do Playtime Start Date, End Date and a Releases' Release Date. Every attribute is required except for a User's First Name and Last Name. Users do not need to follow or be followed, nor do they need to have any collections, ratings, or logged play time. User

Passwords should be at least 7 characters, and contain at least one number and one special (non-alphanumeric) character. Playtime Duration must be greater than 0.

2.4 Sample instance data

```
User (ID, Username, Email, Password, First_name, Last_name,
            Creation_date, Last_access_date):

1. (001, jdoe123, jdoe123@example.com, Passw0rd!, John, Doe,
        2022-05-15, 2024-09-25)

2. (002, amartin89, amartin89@example.com, Secret1234, Alice, Martin,
        2023-01-08, 2024-09-22)

3. (003, bking55, bking55@example.com, King\$45pass, Ben, King,
        2021-12-19, 2024-09-23)

4. (004, snguyen22, snguyen22@example.com, Pa55wordNg, Sarah, Nguyen,
        2023-06-11, 2024-09-26)

5. (005, rwilson77, rwilson77@example.com, Wils0nStrong!, Robert, Wilson,
        2022-09-29, 2024-09-24)


Following (Follower_ID, Followee_ID):

1. (001, 002)
2. (001, 003)
3. (003, 001)
4. (002, 001)
5. (005, 002)


Genre (ID, Name)

1. (001, Action)
2. (002, Platformer)
```

3. (003, Rhythm)
4. (004, Roguelike)
5. (005, Shooter)

Videogame (ID, Title, ESRB)

1. (001, The Last Quest, T)
2. (002, Battle Arena X, E10+)
3. (003, Pixel Adventures, E)
4. (004, Mystery of the Lost City, T)
5. (005, Galactic Wars, M)

Developer (ID, Name)

1. (001, David the Developer)
2. (002, Toby Fox from Undertale)
3. (003, John the Keyboarder)
4. (004, Skate 3 Devs)
5. (005, Lucasfilms LLC)

Publisher (ID, Name)

1. (001, Devolver Digital)
2. (002, 30 monkeys on typewriters)
3. (003, Penguin Random House)
4. (004, Lockheed Martin)
5. (005, Wizards with Guns)

Playtime (User_ID, Game_Id, Start_datetime, end_datetime)

1. (001, 003, 2023-03-03 4:11, 2024-03-03 5:30)
2. (002, 001, 2022-01-01 9:20 2022-01-01 9:25)
3. (001, 003, 2024-03-03 12:30, 2024-03-05 1:25)
4. (004, 001, 1999-11-20 1:35, 199-11-20 4:40)
5. (002, 001, 2022-01-01 9:30, 2022-01-01 9:34)

Publishing (Game_ID, Publisher_ID)

1. (001, 003)
2. (002, 003)
3. (003, 003)
4. (003, 001)
5. (005, 004)

Developing (Game_ID, Developer_ID)

1. (001, 001)
2. (001, 002)
3. (002, 001)
4. (005, 005)
5. (004, 003)

Platform (ID, Name)

1. (001, PlayBox)
2. (002, GameSphere)
3. (003, X-Cube)
4. (004, NeoStation)
5. (005, HandiConsole)

Release (Platform_ID, Game_ID, Price, Release_date)

1. (001, 001, 59.99, 2023-11-15)
2. (002, 002, 49.99, 2024-01-10)
3. (003, 003, 39.99, 2022-06-20)
4. (004, 004, 59.99, 2024-03-05)
5. (005, 005, 69.99, 2023-09-30)

GameGenre (Genre_ID, Game_ID)

1. (001, 001)
2. (001, 002)
3. (001, 003)
4. (003, 001)
5. (002, 001)

Collection (ID, Name, User_ID)

1. (001, Adventure Games, 001)
2. (002, Action Packed, 002)
3. (003, Retro Classics, 003)
4. (004, Mystery & Puzzle, 004)
5. (005, Sci-Fi Legends, 005)

Rating (User_ID, Game_ID, rating)

1. (001, 001, 5)
2. (002, 001, 1)
3. (003, 002, 3)
4. (004, 003, 4)
5. (003, 001, 3)

PlatformOwnedByUser (User_ID, Platform_ID)

1. (001, 001)
2. (002, 002)
3. (003, 003)
4. (004, 004)
5. (005, 005)

GameInCollection (Game_ID, Collection_ID)

1. (001, 001)
2. (002, 001)
3. (003, 002)
4. (004, 003)
5. (005, 004)

3 Implementation (Phase 2)

It was during the design of the database that we learned something that should've been obvious - many to many relationships don't need to have IDs if they don't have dependencies. They're always referenced by one of the two components, so unless they're in relationships, there's no need to give them an ID.

We populated the user data by generating some random content - random first name, last name, username and email as some combination of first last name with a little bit of extra content. Passwords were generated by taking the first last half of the most common passwords in the world and mashing them together randomly.

We got most of our game data from steam using JSON scripts, such as the title of games, their developer and publisher, as well as the release date. Some data we had to input manually because steam doesn't keep good track of them internally, for example the games' ESRB ratings and different platforms were not well kept.

Example Table Creation Statements:

```
create table p320_23.user(id serial PRIMARY KEY, first_name varchar(16),
    last_name varchar(16), password varchar(16),
    email varchar(32), creation_date time, last_accessed time);
```

Example SQL Statements:

```
insert into p320_23.user(username, email, password, first_name,
    last_name, creation_date, last_access_time) values
        ('{username}', '{email}', '{password}',
        '{f_n}', '{l_n}', '{now}', '{now}')
```

4 Data Analysis (Phase 3)

4.1 Hypothesis

Use this section to state the objectives of your data analysis; what are the observations you are expecting to find. Note that your final observations may end up differing from your proposal, that is also a valid result.

4.2 Data Preprocessing

Use this section to describe the preprocessing steps you have performed to prepare the data for the analytics. Preprocessing steps may include: data cleaning (e.g., filling missing values, fixing outliers), formatting the data (e.g., resolving issues like inconsistent abbreviations, multiples date format in the data), combining or splitting fields, add new information (data enrichness).

Explain how the data was extracted from the database for the analysis; if you used complex queries or views, or both.

4.3 Data Analytics & Visualization

Use this section to explain the process/techniques used to analyze the data, use data visualization to present the results, and explain them.

4.4 Indexing

Use this section to explain the indexes that you added to speed up your queries. If you did not add any indexing, you should justify why no indexes were added. This section should have indexing/justification for 5 indexes. Finally, add an appendix of all the SQL statements created in your application during Phase 3 and a description of the indexes created to boost the performance of your application.

4.5 Conclusions

Use this section to explain the conclusions drawn from your data analysis.

Table 1: Feelings about Issues

Flavor	Percentage	Comments
Issue 1	10%	Loved it a lot
Issue 2	20%	Disliked it immensely
Issue 3	30%	Didn't care one bit
Issue 4	40%	Duh?

5 Lessons Learned

Use this section to describe the issues you faced during the project and how you overcame them. Also, describe what you learned during this effort; this section, like the others, plays a critical component in determining your final grade.

The next subsection is meant to provide you with some help in dealing with figures, tables and references, as these are sometimes hard for folks new to LaTeX. Your figures and tables may be distributed all over your paper (not just here), as appropriate for your paper.

Please delete the following subsection before you make any submissions!

5.1 Tables, Figures, and Citations/References

Tables, figures, and references in technical documents need to be presented correctly. As many students are not familiar with using these objects, here is a quick guide extracted from the ACM style guide.

First, note that figures in the report must be original, that is, created by the student: please do not cut-and-paste figures from any other paper or report you have read or website. Second, if you do need to include figures, they should be handled as demonstrated here. State that Figure 2 is a simple illustration used in the ACM Style sample document. Never refer to the figure below (or above) because figures may be placed by LaTeX at any appropriate location that can change when you recompile your source *.tex* file. Incidentally, in proper technical writing (for reasons beyond the scope of this discussion), table captions are above the table and figure captions are below the figure. So the truly junk information about flavors is shown in

Table 1.

Figure 2: A sample black & white graphic (JPG).

6 Resources

Include in this section the resources you have used in your project beyond the normal code development such as data sets or data analytic tools (i.e. Weka, R).

7 Appendix