MyAnimeList User and Anime data

SOEN 363 - Section S

Project Phase II - Final Presentation

Submitted to: Dr. Essam Mansour

Team: Bits Please

Auvigoo Ahmed - 40128901

Bicher Chammaa - 40096200

Radley Carpio - 40074888

Radjabi Siar - 40136258

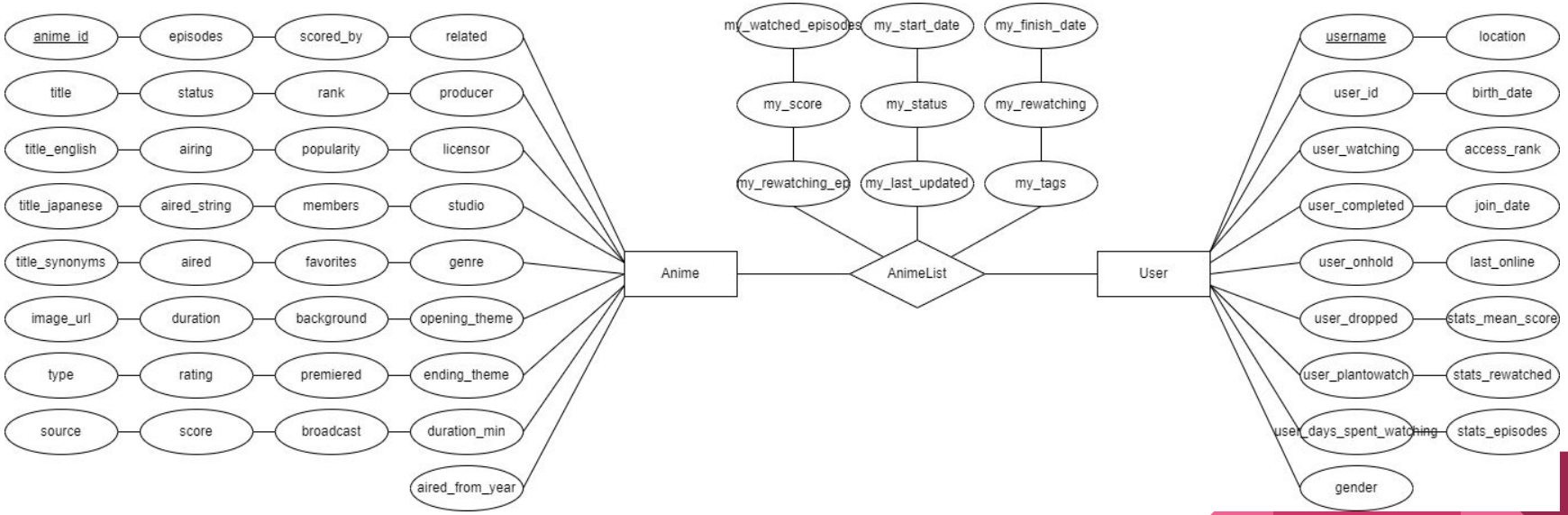
PostgreSQL

What is the dataset? how big is it? the original format? how many files?

- Data set:
  - Source: Kaggle (https://www.kaggle.com/azathoth42/myanimelist)
  - Topic: MyAnimeList users and anime data
- Files:
  - anime_cleaned.csv: 6.03 MB
  - users_cleaned.csv: 15 MB → Total size: 1.041 GB
  - animelists_cleaned.csv: 1.02 GB
    - “cleaned” version of the .csv files have truncated users with impossible numbers of episodes watched or with corrupt data, such as anime finish date = year 1900

PostgreSQL (continued)

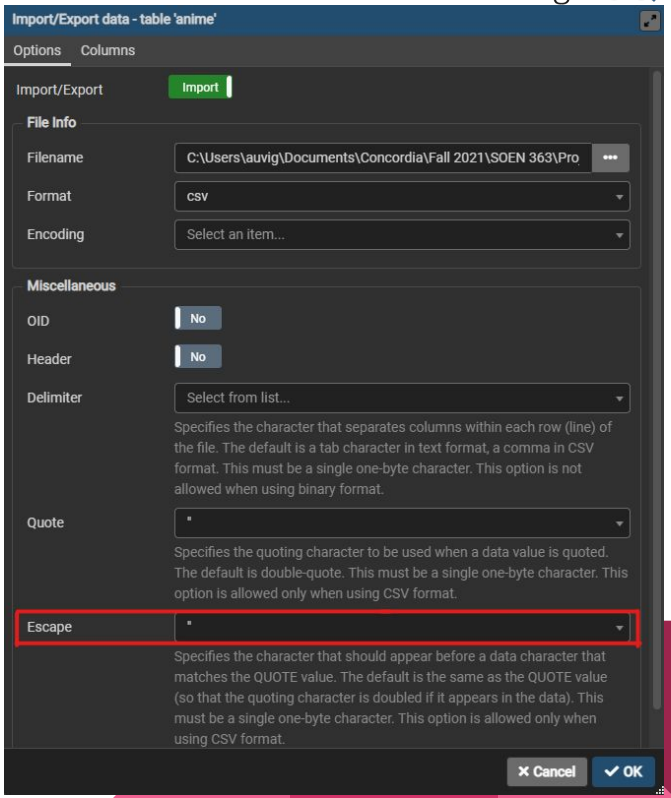
The ER diagram of the dataset



PostgreSQL (continued)

Loading the dataset into the database

- Download the dataset
- Run the DDL scripts to create the *Anime*, *Users* and *AnimeList* tables
- In pgAdmin, right-click on the table schema and select Import/Export data
- The order of importing should be:
  - Anime table
  - Users table
  - AnimeList table
    - The animelists_cleaned.csv file is very large (1.02GB), causing importing issues with pgAdmin
    - Solution was to split the file into three separate .csv files and import them consecutively
    - Escape character is specified to be " because of data that contains text with "



PostgreSQL (continued)

4 Queries to get insight on anime, user demographics:

- Get the top 100 users who watched the highest total number of episodes and their count and location (Q4)
- Get the top anime genre per gender (popularity percentage + appreciation percentage/2) (Q7)
- The percentage of time the top 50 users with the highest watch time have spent watching anime since creating their accounts (Q8)
- Find out whether or not users who have watched the greatest number of animes also have the most watch time (Q10)

Neo4j

What is the dataset? how big is it? the original format?

- Data set:
  - Same MyAnimeList data set used in PostgreSQL
- Files:
  - anime_cleaned.csv: 6.03 MB
  - users_cleaned.csv: 15 MB
  - animelists_cleaned.csv: 1.02 GB

Total size: 1.041 GB

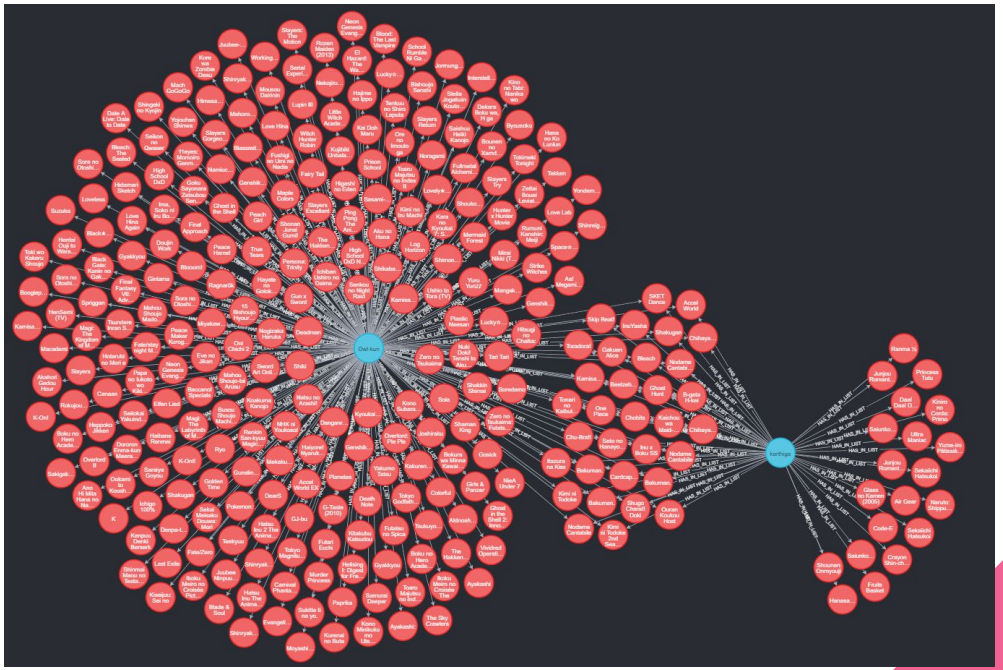
  - PRE-PROCESSING:
    - Issues with importing the split animelists_cleaned.csv files into Neo4j
    - Inability to specify escape character to be “ like when importing from pgAdmin
    - Manually clean the split animelists_cleaned.csv to escape the “ character properly

Neo4j (continued)

A data model for the dataset

Neo4j (continued)

Example of data model visualization after running a simple query:



Neo4j (continued)

Loading the dataset into the database

- Download the CLEANED, PRE-PROCESSED data set
- Import the dataset into Neo4j desktop for quick access
- Run the DDL scripts to create the indexes for the Anime and User nodes
- Set DBMS settings to have max heap size of 4GB for faster import
- Run the LOAD scripts for Anime and User to create the nodes
- Run the LOAD script for each split and cleaned csv file for animelists
- :auto USING PERIODIC COMMIT 100000
    - Each split csv file has ~5 million rows

```
:auto USING PERIODIC COMMIT 100000
LOAD CSV WITH HEADERS FROM 'file:///animelists_cleaned_split/animelists_cleaned_2.csv' AS row
WITH
row.username AS username,
toInteger(row.anime_id) AS anime_id,
toInteger(row.my_watched_episodes) AS my_watched_episodes,
row.my_start_date AS my_start_date,
row.my_finish_date AS my_finish_date,
toInteger(row.my_score) AS my_score,
toInteger(row.my_status) AS my_status,
row.my_rewatching AS my_rewatching,
toInteger(row.my_rewatching_ep) AS my_rewatching_ep,
datetime(replace(row.my_last_updated, ' ', 'T')) AS my_last_updated,
row.my_tags AS my_tags

MATCH (a:Anime {anime_id:anime_id})
MATCH (u:User {username:username})

MERGE (u)-[rel:HAS_IN_LIST {my_watched_episodes: my_watched_episodes}]→(a)
SET rel.my_start_date = my_start_date,
    rel.my_finish_date = my_finish_date,
    rel.my_score = my_score,
    rel.my_status = my_status,
    rel.my_rewatching = my_rewatching,
    rel.my_rewatching_ep = my_rewatching_ep,
    rel.my_last_updated = my_last_updated,
    rel.my_tags = my_tags

RETURN count(rel)
```

Neo4j (continued)

Discuss the consistency and availability of the NoSQL used in the project

- Neo4j is fully ACID compliant (Atomicity, Consistency, Isolation, Durability)
- Prioritizes consistency:
  - All nodes in the system will always see the same database values
  - Consequently, requires complex locking mechanisms
  - Data is safely stored
- Availability:
  - Neo4j high availability (HA) offered in enterprise edition
  - The graph is not sharded; complete graph is replicated across clusters
  - Master-slave architecture:
    - Master manages the locking mechanisms, propagates updates to slave clusters
  - Consequently, no partition tolerance

Neo4j (continued)

Discuss the indexing techniques available in the NoSQL used in the project

- Indexes can be added on nodes and relationships
- Single-property index: index created on a property of a label (node or relationship)
- Composite index: index created on two or more properties of a label (node or relationship)
- Types of indexes:
  - B-tree: good for equality lookups and range scans for any type of value
  - Text: good for lookups that use CONTAINS for only string values
  - Token lookup: good for looking up nodes or relationships with a specific label
- Cypher's query planner decided the best index to use depending on the situation
- Sample script to create index on node:
- ```
  myanimelist$ CREATE INDEX FOR (a:Anime) ON (a.anime_id);
  ```
- Adding unique constraint on node properties AUTOMATICALLY creates the index on that property
  ```

Neo4j (continued)

4 Queries to get insight on anime, user demographics:

- Get users who added “One Piece” in their list (Q1)
- How much anime do Montrealers consume (Q4)
- Average age of anime consumer (Q5)
- How many times more likely is a user going to complete the Naruto series rather than dropping it? (Q10)