SOEN 363 Project Phase 2

Marwa Khalid (40155098) – khalidmarwa786@gmail.com
William Wells (40111253) - williamwells2013@hotmail.com
Lucas Catchlove (27145640) - lucascatchlove@gmail.com
Shawn Gorman (40157925) - gorman.shawn23@hotmail.com

Analyzing Big Data With SQL and RDBMS


PostgreSQL

Dataset

About Dataset:

- contains information about 17, 562 animes and the preference from over 300 000 different users.
- Provides the anime list per user, their ratings, and more.

Dataset Information:

- Size: 2.87GB
- Format: CSV
- Number of files: 3
  - **anime_list.csv:** contains the list of unique animes per user with rating and other information.
  - **ratings_completed.csv**: contains the list of ratings given by the user to animes with watching status = 2 (complete)
  - **anime.csv**: contains information of anime scrapped of main page and stats page.

Link:https://www.kaggle.com/datasets/hernan4444/anime-recommendation-database-2020?select=anime.csv

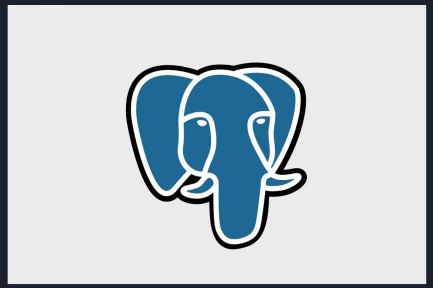
Context

Once upon a time, there was a person named Alice who had always been a huge fan of anime. She grew up watching her favorite shows and collecting all sorts of merchandise, from plushies and figurines to keychains and posters.

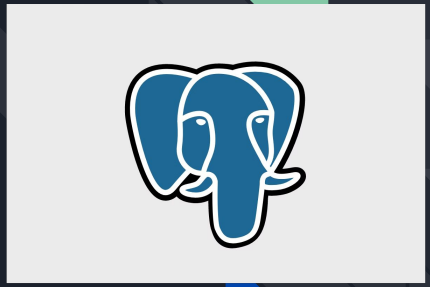
Context - contd

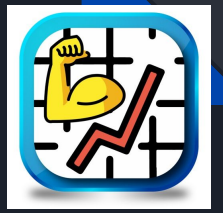
As she got older, Alice decided that she wanted to share her love of anime with others and opened up a small shop that sold anime merchandise.

She learned how to use SQL to organize and analyze her sales data, which helped her to see which items were popular and which ones were not. With this information, she was able to make more strategic decisions about which items to stock in her shop.

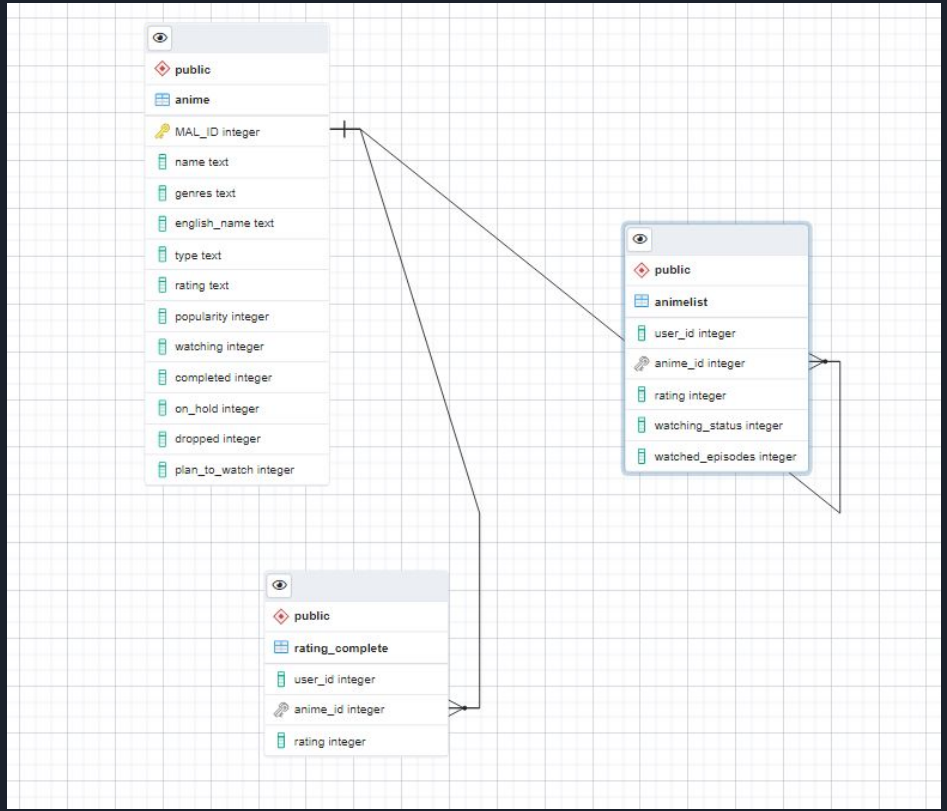


In the end, Alice's love of anime and her use of SQL helped her to create a successful and thriving business!





ER Diagram


public
anime
MAL_ID integer
name text
genres text
english_name text
type text
rating text
popularity integer
watching integer
completed integer
on_hold integer
dropped integer
plan_to_watch integer
public
animelist
user_id integer
anime_id integer
rating integer
watching_status integer
watched_episodes integer
public
rating_complete
user_id integer
anime_id integer
rating integer

Creating ‘anime’ Table

```
-- Table: public.anime

-- DROP TABLE IF EXISTS public.anime;

CREATE TABLE IF NOT EXISTS public.anime
(
    mal_id integer NOT NULL,
    name text COLLATE pg_catalog."default" NOT NULL,
    genres text COLLATE pg_catalog."default" NOT NULL,
    english_name text COLLATE pg_catalog."default" NOT NULL,
    type text COLLATE pg_catalog."default" NOT NULL,
    rating text COLLATE pg_catalog."default" NOT NULL,
    popularity integer NOT NULL,
    watching integer NOT NULL,
    completed integer NOT NULL,
    on_hold integer NOT NULL,
    dropped integer NOT NULL,
    plan_to_watch integer NOT NULL,
    CONSTRAINT anime_pkey PRIMARY KEY (mal_id)
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.anime
    OWNER to postgres;
```

Creating ‘animelist’ Table

```
CREATE TABLE IF NOT EXISTS public.animelist
(
    user_id integer NOT NULL,
    anime_id integer,
    rating integer,
    watching_status integer,
    watched_episodes integer,
    CONSTRAINT anime_id FOREIGN KEY (anime_id)
        REFERENCES public.anime ("mal_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.animelist
    OWNER to postgres;
-- Index: fki_N

-- DROP INDEX IF EXISTS public."fki_N";

CREATE INDEX IF NOT EXISTS "fki_N"
    ON public.animelist USING btree
    (anime_id ASC NULLS LAST)
    TABLESPACE pg_default;
```

Creating ‘ratingcomplete’ Table

```
CREATE TABLE IF NOT EXISTS public.rating_complete
(
    user_id integer NOT NULL,
    anime_id integer NOT NULL,
    rating integer NOT NULL,
    CONSTRAINT anime_id FOREIGN KEY (anime_id)
        REFERENCES public.anime ("mal_id") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.rating_complete
    OWNER to postgres;
-- Index: fki_anime_id

-- DROP INDEX IF EXISTS public.fki_anime_id;

CREATE INDEX IF NOT EXISTS fki_anime_id
    ON public.rating_complete USING btree
    (anime_id ASC NULLS LAST)
    TABLESPACE pg_default;
```

Live demo SQL

Analyzing Big Data Using NoSQL Document Database

Context

- The Chicago Police Department wants to identify patterns and trends in crime over time, and to better understand the underlying factors that may be contributing to those trends.

- This could include analyzing data on the types of crimes being committed, the locations where crimes are occurring,

Context

By using a document-oriented NoSQL database, the police department could quickly and easily retrieve and analyze data from a variety of sources.

This could help the department to make more informed decisions about how to allocate resources.

Dataset & Database Management Software

- Dataset is on Chicago crime from 2001 to present (dataset is ~1.93gb in CSV format): https://catalog.data.gov/dataset/crimes-2001-to-present
  - Hard drive usage of database with dataset loaded: ~5.27 GIB

- Couchbase was chosen due to its simplicity and that it is for document databases

- Only required single simple command to import dataset:
  - cbimport csv --infer-types -c http://localhost:8091/ -u username -p password -b "chicago-crime" --scope-collection-exp "_default._default" -g "#UUID#" -d file://path here

Data model of dataset

Each crime is a document with the following:

- Arrest (boolean)
- Beat (number)
- Block (string)
- Case Number (string)
- Community Area (number,string)
- Date (string)
- Description (string)
- District (number)
- Domestic (boolean)
- FBI Code (number,string)
- ID (number)
- IUCR (number,string)
- Latitude (number,string)
- Location (string)
- Location Description (string)
- Longitude (number,string)
- Primary Type (string)
- Updated On (string)
- Ward (number,string)
- X Coordinate (number,string)
- Y Coordinate (number,string)
- Year (number)

Example Of Interesting Queries For Useful Info

Top Ten Districts With Highest Number Of Domestic Assaults

```
SELECT District,
    COUNT(*) AS domestic_assault_incidents
FROM `chicago-crime`
WHERE `Primary Type` = "ASSAULT"
  AND Domestic=TRUE
GROUP BY District
ORDER BY domestic_assault_incidents DESC
LIMIT 10
```

Example Of Interesting Queries For Useful Info

Years With Highest Number Of Burglaries

```
SELECT Year,
    COUNT(*) AS number_of_burglaries
FROM `chicago-crime`
WHERE `Primary Type`="BURGLARY"
GROUP BY Year
ORDER BY Year DESC
```

Example Of Interesting Queries For Useful Info

Total Arrest Rate

```
SELECT (a.arrests / ta.total_arrests)*100 AS arrest_rate_percentage
FROM (
   SELECT COUNT(*) AS arrests
   FROM `chicago-crime`
   WHERE Arrest=TRUE) AS a,
(
   SELECT COUNT(*) AS total_arrests
   FROM `chicago-crime`) ta
```

Example Of Interesting Queries For Useful Info

Cannabis Possession Offenses Count By Community Area

```
SELECT `Community Area`,
    COUNT(Description) cannabis_possession_offenses
FROM `chicago-crime`
WHERE Description LIKE "%POSS: CANNABIS%"
GROUP BY `Community Area`
```

Example Of Interesting Queries For Useful Info

Crimes Involving A Firearm At Schools

```
SELECT `Case Number`,
    Description
FROM `chicago-crime`
WHERE Description LIKE "%FIREARM%"
  AND `Location Description` LIKE "%SCHOOL%"
```

Example Of Interesting Queries For Useful Info

Primary Type Of Crime And The Number Of Crimes For Each District

```
SELECT `Primary Type`, `District`, COUNT(*) AS `NumberOfCrimes`

FROM `chicago-crime`

GROUP BY `Primary Type`, `District`
```

Example Of Interesting Queries For Useful Info

Top 10 Most Common Crime Descriptions

```
SELECT `Description`, COUNT(*) AS `NumberOfOccurrences`
FROM `chicago-crime`
GROUP BY `Description`
ORDER BY `NumberOfOccurrences` DESC
LIMIT 10
```

Example Of Interesting Queries For Useful Info

All Crimes That Occurred On A Specific Date

```
SELECT *

FROM `chicago-crime`

WHERE `Date` LIKE '04/19/2004%'
```

Example Of Interesting Queries For Useful Info

All Crimes That Occurred Within A Specific Location Or Block

```
SELECT *

FROM `chicago-crime`

WHERE `Block` = '004XX E 63RD ST'
```

Example Of Interesting Queries For Useful Info

Number Of Crimes That Occurred In Each Community Area

```
SELECT `Community Area`, COUNT(*) AS `NumberOfCrimes`

FROM `chicago-crime`

GROUP BY `Community Area`
```

Balance Between Consistency and Availability

- Couchbase allows you to configure the level of consistency and availability for your data using a number of different options
  - Consistency mode, the replication factor, and the durability level

- This puts the balancing consistency and availability in the database completely into the hands of the database administrator and maintainer

Indexes Techniques

- Couchbase supports a variety of indexing techniques
  - Primary indexes, secondary indexes, and functional indexes.
    - Primary indexes are the default indexes that are automatically created for every bucket in Couchbase.
    - Secondary indexes allow you to index any attribute or combination of attributes in your documents, allowing you to perform efficient lookups and queries based on those attributes.
    - Functional indexes are special types of indexes that store the results of a particular function or expression, rather than the raw values of the indexed attributes.