DataBase Management System
Final Project

Simulated Search Application Design

Team-1
Yashwant Avula- YashwantAvula
Yash Sanghani- Ysanghani521
Siddharth Yalamanchili- namaste-world
Anton Vernikov - AntonVernz
https://github.com/YashwantAvula/Team_1_694_2024

Index

1. Abstract

The development of a Twitter search application involves data management and search capabilities. This report describes the design and implementation of a search application that allows users to search for tweets based on various criteria, including user ID, partial tweet text, hashtags, and timeframe range.

To ensure efficiency, the application leverages a combination of PostgreSQL, Elasticsearch and caching mechanisms to manage tweet and user data. The data model and datastore structure are optimized for efficient retrieval.

This report discusses the design choices made during the development of the database and caching systems, and the pipeline of search queries into datastore operations. It also provides the results of test queries, focusing on response times for cached and non-cached data. The report concludes with observations on the performance of the application and the implications for further development.

2. Introduction

Twitter is a vast repository of real-time information, but finding specific tweets or tracking trends is potentially difficult. To address this, the search application was designed to simulate a Twitter environment, allowing users to search through a predefined dataset. The application provides functionality to find tweets based on user ID, twitter ID, specific text, hashtags, and timeframes.

The simulated environment created is a system that replicates the behavior of a live search application while working with static data (more information on page 5). To achieve this, PostgreSQL was used for structured data management, Elasticsearch for indexing and efficient searching as well as a caching system to improve response times. This setup was designed to handle high query volumes and provide rapid search results while maintaining data consistency and accuracy.

This report details the design and implementation of this simulated Twitter search application. It discusses the structure of the data model, the storage approach, caching mechanisms, and the types of search queries supported. Additionally, the trade-offs and optimizations that were necessary to maintain a smooth user experience were addressed. The report concludes with the results of test queries, offering insights into the application's performance in this simulated context.

3. Dataset Description

The dataset used for this project contains a total of 101,894 tweets, which includes a few duplicates. Each tweet in the dataset is contained in a JSON object and holds various types of metadata, such as tweet text,

timestamp, user information, hashtags, and media content. The size of the dataset, which includes all these elements, indicates a substantial volume of data for analysis.

Each tweet consists of several components. The core element of each tweet is the tweet text which is accompanied by a timestamp. In addition to the tweet content, the dataset includes user-related information such as the user's name, screen name, user ID, location, and other profile details. The dataset may also contain media content, such as images, videos, or GIFs, as well as hashtags used within the tweets.

Given the large size of the dataset, it exhibits notable characteristics. The tweets are in multiple languages, with common languages including English, Portuguese, and German. Content varies from plain text to media-rich tweets containing images or videos. The dataset also includes information on user profiles, allowing analysis of various user-related factors, such as follower count and user activity. The presence of duplicate tweets requires careful handling during data processing to ensure accurate results.

Overall, this dataset offers a wide range of analysis possibilities, from examining language-specific trends to exploring patterns in user engagement. The variety of content and associated metadata allows for diverse approaches to data analysis, while the temporal aspect, provided by the tweet timestamps, facilitates trend analysis over time. Given the dataset's complexity, proper data processing and deduplication techniques are essential for deriving meaningful insights.

4. Databases

The project's data architecture is designed to optimize storage and retrieval efficiency, supporting a variety of search functionalities crucial for the application. This section describes the data models and the corresponding datastores used, focusing on their roles in enabling the various search methods.

The data model is structured into 5 main entities, as reflected in the database schema:

1. User Meta Data: Holds detailed information about the users. Attributes include:
    - user_id: A unique identifier for the user.
    - screen_name: The Twitter handle of the user.
    - created_at: Timestamp indicating the account creation on Twitter.
    - followers: Number of followers the user has.

2. Original Tweet Metadata: Captures information about each tweet posted by users. Attributes include:
    - tweet_id: Unique identifier for the tweet.
    - user_id: References the user_id from the User Meta Data to establish a relationship between the tweet and the user.
    - created_at: Timestamp indicating when the tweet was posted.
    - language: The language of the tweet.
    - hashtags: A text field containing all hashtags included in the tweet.

- retweet_count: The number of times the tweet has been retweeted.

3. Retweet Metadata: Stores data specific to retweets. It shares the same structure with the Original Tweet MetaData to maintain consistency and ensure efficient queries related to retweets.
- tweet_id: Unique identifier for the tweet.
- user_id: References the user_id from the User Meta Data to establish a relationship between the tweet and the user.
- created_at: Timestamp indicating when the tweet was posted.
- language: The language of the tweet.
- hashtags: A text field containing all hashtags included in the tweet.
- retweet_count: The number of times the tweet has been retweeted.

4. Tweets_project : Stores data of both original tweets and retweets. This table in SQL was created to optimize search times for specific search cases such as time frame search and hashtags. This table is used to search for all tweets through one table as opposed to iterating over multiple tables within SQL. Also contains retweet count, user's followers count and favorite count that is used for ranking of search results.

Datastores

To fulfill requirements of the project, both relational and non-relational data stores are employed:

PostgreSQL: A relational database management system chosen for its robustness, ACID compliance, and extensive support for complex queries. It is used to store user data and tweet metadata, where relationships and integrity constraints are crucial. The relational nature of PostgreSQL supports complex queries involving joins that are essential for aggregating data across users and tweets.

Elasticsearch: Used for indexing and searching tweets based on tokenized text content, hashtags, user_id, tweet_id and created_at. Elasticsearch is chosen for its excellent full-text search capabilities, scalability, and speed, particularly when dealing with large datasets such as Twitter data streams. It allows for quick retrieval of tweets based on various criteria without the performance penalties that might be incurred with traditional relational databases.

5. Processing Tweets

Data is loaded into the respective datastores using a line-by-line processing approach, simulating a streaming data ingestion scenario. This method emphasizes the pipeline's capability to handle real-time data efficiently, critical for maintaining up-to-date search results and analytics. The code provided details the SQL commands for creating tables, the logic for parsing and inserting data into PostgreSQL, and the indexing process in Elasticsearch. This comprehensive data handling strategy supports complex queries and provides a robust foundation for the search functionalities implemented in the project.

Relational Databases used : PostgreSQL
Non Relational Databases used : Elastic Search Indexing

Tweet Processing Workflow

1. Extraction:
Each tweet is represented as a JSON object containing various metadata fields such as id, text, user, entities (which include hashtags), and created_at. The extraction process involves parsing these JSON objects to retrieve relevant information.

2. Transformation:

- Date Parsing: The created_at field in tweets is formatted according to Twitter's standard, which needs conversion into a timestamp format compatible with PostgreSQL. A utility function, parse_twitter_date, converts this to an ISO8601 format.
- Text Tokenization: The tweet text is tokenized to facilitate effective search and analysis. This tokenization splits the text into individual words or tokens, which are then used for indexing in Elasticsearch.
- Hashtag Extraction: Hashtags are extracted from the entities field. They are critical for searches specific to topics and are stored both as a part of the tweet object in PostgreSQL and separately indexed in Elasticsearch.

3. Data Loading:

- User Data: Before inserting tweet data, the system checks if the user's information is already present in the user_meta_data table to avoid duplication. If not present, it inserts the new user data including user_id, screen_name, created_at, and followers.
- Tweet Data: Tweets are loaded into original_tweet_meta_data or retweet_meta_data depending on their type. Each tweet's metadata including tweet_id, user_id, created_at, language, hashtags, and retweet_count are stored. A foreign key relationship with the user_meta_data table ensures referential integrity.
- Elasticsearch Indexing: Concurrently with PostgreSQL data loading, tweet texts along with their metadata are indexed in Elasticsearch. This dual loading strategy ensures that while the relational data provides comprehensive analytical capabilities, the indexed data supports efficient full-text searches.

4. Batch Processing and Streaming Simulation

The tweets are processed one at a time to simulate a streaming data environment, which is typical for real-time Twitter data processing. This approach mimics the operational challenges and solutions required in a live-data scenario:

Batch Processing: Though processing is done one tweet at a time, transaction management in PostgreSQL ensures that changes are committed periodically, reducing the overhead of frequent disk writes.
Error Handling: Robust error handling mechanisms are in place to manage issues like JSON decoding errors or database insertion errors, ensuring the system's resilience and reliability.

6. Caching

For the caching system, a python list was used to store search results. Each entry in the list consists of a tuple of length 2, with the first entry containing a list of the tweet information and the second entry being the time last searched, which is used for the eviction of tweets. When doing a search, before searching the database, a check is done to see if any of the tweets stored in cache fits the search query, and if so, that tweet's information will be added to the list of the results eventually outputted when the rest of the search is complete and the tweet id will be inserted into a list of all unique tweet IDs used for said search. Then a first search is done on the Elasticsearch database, extracting the relevant tweet IDs, and a second followup search is done on the PostgreSQL database, extracting the results from said tweet ID, but only if that tweet ID is not already in the list of unique tweet IDs. This way if the result is already stored in the cache, the search for that specific tweet is skipped.

For the tweets stored in the cache, a cache size of 1000 was selected, as this represented around 1% of the tweets stored in the database. Roughly 50% of the cache was used for static tweet information, that will never get evicted, and the other 50% was for data that did get evicted. For the static tweet information, tweets that would likely to be searched frequently were decided as the tweets to be stored. The first set of tweets being the top 10 most popular tweets by the 10 most followed twitter accounts in the database. For the second set of tweets stored in cache, an analysis was done on the most popular hashtags, and 50 of the most commonly used hashtags were selected, with the 10 most popular tweets for each hashtag being stored in the database. For the tweets that would be evicted, these were the top 50 tweets from the last 10 (roughly) search queries

For eviction, to make sure the cache was limited to a certain capacity, the least recently used policy is used. This makes it so frequent searches would remain stored in the cache and those that have not been saved, will be replaced with a more recently stored cache. The cache stores up to 1000 search results, so when it's not full it will continue to store search results, and when it is full, for each search, the top 50 results (or less if the search query returns less than 50 results) not already in cache will be added to the cache, with their time of entry into cache being when they were the search was started. These entries will replace the 50 results that appear least frequently in the cache, which is extracted from the second entry of the tuple. To make sure that the static tweets that would remain in the cache at all times would not be dropped, a very large value(10 trillion) was used for the time of entry in the cache, so they would always remain as the most frequently searched

7. Search Application Design

1. Allowed searches and drill downs
In the Search Application Design, the implemented system facilitates diverse types of searches to efficiently navigate through the extensive dataset of tweets. The primary search capabilities include searching by Tweet text, hashtag, user ID, tweet ID, and within a specified time range. These searches leverage both exact match and partial match queries for tweet Text, catering to a wide range of user intentions.

The application supports drill-down features that enhance the user experience by allowing deeper investigation into the search results. Upon retrieving a list of tweets, the user can view the meta-data for each tweet, including the author's username, the date and time of posting, and the counts of retweets and favorites. This provides a thorough understanding of the tweet's impact and the author's presence on the platform.

The search application is designed with an emphasis on relevance, sorting the results by a combination of factors including the number of retweets, favorites, and the followers of the user. This ensures that the most impactful tweets, based on user engagement and influence, are prioritized in the search results. The incorporation of these features into the application ensures a robust and user-centric approach to navigating and understanding the dynamics of Twitter data.

2. Way in which search queries are handled and translated into queries for the datastores
Search queries within the application are translated into queries for the datastores by a combination of user input processing and system logic that converts the user's search intentions into executable database and Elasticsearch queries.

When a user initiates a search, the input is first sanitized to prevent any SQL injection or similar security issues. This sanitized input is then used to construct a structured query that the datastore understands. For example:

- Text searches: User inputs for text-based searches are tokenized and matched against tweet text data stored in Elasticsearch. The `match` query type is utilized to find tweets containing the user's input, taking advantage of Elasticsearch's full-text search capabilities.

- User and hashtag searches: When searching for user-related data or hashtags, if the search is initiated with an '@' symbol, the application strips these characters for accurate matching. The constructed query then uses the `term` or `match` queries for precise or partial matches in both PostgreSQL and Elasticsearch.

- Time range searches: For time-based searches, the application constructs a `range` query that filters results within the specified start and end dates. This query is executed in PostgreSQL using a `BETWEEN` clause, and in Elasticsearch using the `range` query functionality.

- ID searches: For searches by user ID or tweet ID, the application constructs queries that use the `term` query in Elasticsearch and an equality comparison in PostgreSQL to fetch the exact record.

In all cases, the search queries consider the indexing strategies used in the datastores. For example, frequently accessed fields like user IDs, hashtags, and tweet text are indexed to expedite query execution. PostgreSQL queries are optimized using `JOIN` operations where necessary, and the results are ordered by a combination of retweet count, favorite count, and user followers to reflect relevance.

The application's back end is responsible for converting these high-level strings into the specific query syntax required by each datastore, ensuring that the user's search translates into fast and accurate data retrieval.

3. Notion of relevance

In the search application, relevance is given together from a few key elements:

- Exactness: The closer a tweet's content matches the search term, the higher it ranks.
- Popularity: Tweets that have been liked, retweeted, or replied to more frequently are given priority.
- Authority: Users with more followers can indicate greater importance, so their tweets are likely pushed up in the search results.
- Timeliness: Fresh tweets often mean more relevant information, especially for trending topics.
- Hashtag Significance: Tweets that use the searched hashtags in a meaningful way come up first.
- Contextual Fit: Depending on the search, tweets that align well with the specific context or theme the user is interested in are considered more relevant.

Basically, the app tries to figure out what’s most useful and interesting based on what’s being searched and serves that up first.

8. Results

Figures 1, 2, and 3 and 4 below show the resulting tables available in the PostgreSQL database after the tweets are uploaded through the stream process

	tweet_id [PK] bigint	user_id bigint	created_at timestamp with time zone	language character varying (10)	hashtags text	retweet_count integer	tweet_text text
1	1254022770679320576	804046791348015107	2020-04-25 12:21:41-04	pt	[]	0	É isto, ou vo...
2	1254022770746372096	2242948745	2020-04-25 12:21:41-04	de	["sport", "...	0	Schöne Run...
3	1254022772575043586	2929344220	2020-04-25 12:21:42-04	de	[]	0	Was sollen 1...
4	1254022773598572544	1091660129894838272	2020-04-25 12:21:42-04	en	[]	0	@VinceMcM...
5	1254022776094105602	375777294	2020-04-25 12:21:43-04	en	[]	0	im making 1...
6	1254022776207429633	865733987561381888	2020-04-25 12:21:43-04	en	[]	0	@MichaelTo...
7	1254022776752615430	1132273796138905600	2020-04-25 12:21:43-04	en	[]	0	Oh brother a...
8	1254022778371571712	301470336	2020-04-25 12:21:43-04	en	[]	0	ahap , low cu...
9	1254022779608891393	1035107587006648320	2020-04-25 12:21:43-04	hi	[]	0	पश्चिम बंगाल स...
10	1254022780695252993	923169415054680064	2020-04-25 12:21:44-04	en	[]	0	Weekly mort...
11	1254022781710274566	1120761000561606656	2020-04-25 12:21:44-04	en	["kpop"]	0	tony montan...
12	1254022783249637376	847489262018641921	2020-04-25 12:21:44-04	en	[]	0	@HealthDep...
13	1254022784122003457	1151073083836502016	2020-04-25 12:21:45-04	en	[]	0	Germany co...
14	1254022784193363969	567891167	2020-04-25 12:21:45-04	en	[]	0	@realDonald...
15	1254022784646291456	1213865493951438849	2020-04-25 12:21:45-04	en	[]	0	Good mornin...
16	1254022784851861504	1215289012551741443	2020-04-25 12:21:45-04	in	[]	0	Mendamaika...
17	1254022785547923457	852883926	2020-04-25 12:21:45-04	in	[]	0	@PARTHIPA...
18	1254022786588131328	240950393	2020-04-25 12:21:45-04	in	[]	0	aduh ak bng...
19	1254022787729174528	2930195686	2020-04-25 12:21:45-04	de	[]	0	Eine App sie ...
20	1254022788370833408	620106850	2020-04-25 12:21:46-04	en	["michiga...	0	Nursing hom...
21	1254022790333771778	479687857	2020-04-25 12:21:46-04	en	[]	0	Very good q...
22	1254022793148141569	1162422360122777601	2020-04-25 12:21:47-04	en	[]	0	Paris challen...

Total rows: 1000 of 40793 Query complete 00:00:00.267

Fig.1 Original Tweet Metadata

	tweet_id [PK] bigint	user_id bigint	created_at timestamp with time zone	language character varying (10)	hashtags text	retweet_count integer	tweet_text text
1	1254022772558368768	908326492718764034	2020-04-25 12:21:42-04	en	[]	0	RT @BJP...
2	1254022772877131777	1206650133976408064	2020-04-25 12:21:42-04	tr	[]	0	RT @schr...
3	1254022773149589510	1248123252	2020-04-25 12:21:42-04	en	[]	0	RT @Mon...
4	1254022773858545665	50993809	2020-04-25 12:21:42-04	it	[]	0	RT @gust...
5	1254022774521081856	792325679354417152	2020-04-25 12:21:42-04	en	[]	0	RT @Paw...
6	1254022776232394752	469083492	2020-04-25 12:21:43-04	en	["kpop"]	0	RT @clou...
7	1254022777268568064	1044295463267119104	2020-04-25 12:21:43-04	de	["Digitalisierun...	0	RT @evan...
8	1254022778363265024	109877154	2020-04-25 12:21:43-04	it	["25Aprile", "Lib...	0	RT @Quiri...
9	1254022780980527105	2459721182	2020-04-25 12:21:44-04	de	["Corona"]	0	RT @fr: Di...
10	1254022781093601280	2493849410	2020-04-25 12:21:44-04	in	[]	0	RT @Part...
11	1254022781722923009	21400571	2020-04-25 12:21:44-04	de	["Pfarrerstocht...	0	RT @DerL...
12	1254022781873856512	1220425350540382209	2020-04-25 12:21:44-04	nl	[]	0	RT @dvhn...
13	1254022782188261376	1228183815576682497	2020-04-25 12:21:44-04	in	[]	0	RT @The...
14	1254022782561718272	2870459221	2020-04-25 12:21:44-04	en	[]	0	RT @view...
15	1254022783480332288	94687002	2020-04-25 12:21:44-04	de	[]	0	RT @denc...
16	1254022784323252225	1080798295	2020-04-25 12:21:45-04	in	[]	0	RT @oxfa...
17	1254022786449891328	834386400665595904	2020-04-25 12:21:45-04	tr	[]	0	RT @gaze...
18	1254022786454077440	776181606004424706	2020-04-25 12:21:45-04	en	[]	0	RT @otch...
19	1254022786508623872	1184860086008762369	2020-04-25 12:21:45-04	it	[]	0	RT @gieip...
20	1254022786789466112	1178130261302636545	2020-04-25 12:21:45-04	hi	[]	0	RT @bhu...
21	1254022787196387329	814316767665418248	2020-04-25 12:21:45-04	en	[]	0	RT @Pas...
22	1254022788337348608	1103945573353377792	2020-04-25 12:21:46-04	en	[]	0	RT @jjgne...

Total rows: 1000 of 61101 Query complete 00:00:00.274

Fig.2 Retweet Metadata

	tweet_id [PK] bigint	created_at timestamp with time zone	user_id bigint	user_name text	user_screen_name text	tweet_text text	retweet_count integer	favorite_count integer	user_followers integer
1	1254022770(	2020-04-25 12:21:41-04	8040467913	Bi Sex Uau	B_King69	É isto, ou …	0	0	89
2	1254022770	2020-04-25 12:21:41-04	2242948745	Thomas …	tho1965	Schöne Ru…	0	0	173
3	1254022772	2020-04-25 12:21:42-04	9083264927	शचीन्द्र पाण्डेय	im_S_pandey	RT @BJP4…	340	1870	2362
4	1254022772	2020-04-25 12:21:42-04	2929344220	Ralf Sch…	RusticusArat	Was solle…	0	0	778
5	1254022772	2020-04-25 12:21:42-04	1206650133	Büşra Özt…	schrodingerk42	RT @schr…	72	83	318
6	1254022773	2020-04-25 12:21:42-04	1248123252	minhyuk.	mizhyuklee	RT @Mon…	41	108	1194
7	1254022773	2020-04-25 12:21:42-04	1091660129	Milli tekn…	milliteknoloj	@VinceM…	0	0	2294
8	1254022773	2020-04-25 12:21:42-04	50993809	Henry …	Enrico_Bianchi	RT @gusti…	20	29	1165
9	1254022774	2020-04-25 12:21:42-04	7923256793	Balu Pspk	Balu54368353	RT @Paw…	2181	7049	221
10	1254022776	2020-04-25 12:21:43-04	375777294	TeéLanee…	TWD40_	im making…	0	0	637
11	1254022776	2020-04-25 12:21:43-04	8657339875	ANH	BritishKatieKim	@Michael…	0	0	142
12	1254022776	2020-04-25 12:21:43-04	469083492	J.	menggyeuxxxx…	RT @clou…	1	0	916
13	1254022776	2020-04-25 12:21:43-04	1132273796	Terri Kamp	RampTheresa	Oh brother…	0	0	1522
14	1254022777	2020-04-25 12:21:43-04	1044295463	Peter No…	nosskolbe	RT @evan…	4	6	311
15	1254022778	2020-04-25 12:21:43-04	109877154	MrBelzoni	MrBelzoni	RT @Quiri…	1434	4062	339
16	1254022778	2020-04-25 12:21:43-04	301470336	BABA ALI	_FreshAA	ahap , low …	0	0	9406
17	1254022779	2020-04-25 12:21:43-04	1035107587	Satya Hindi	SatyaHindi	पश्चिम बंगा…	0	0	11576
18	1254022780	2020-04-25 12:21:44-04	9231694150	Quantum	QuantumAspect	Weekly m…	0	0	3196
19	1254022780	2020-04-25 12:21:44-04	2459721182	M'kh Pirra…	Alfred996	RT @fr: Di…	1	1	2022
20	1254022781	2020-04-25 12:21:44-04	2493849410	iwan phitoy	nirwansyah122	RT @Parta…	86	259	239
21	1254022781	2020-04-25 12:21:44-04	1120761000	🌸	cloudseokjinie	tony mont…	0	0	0
22	1254022781	2020-04-25 12:21:44-04	21400571	Siegfried …	S_Kraemer	RT @DerL…	9	11	327

Total rows: 101894 of 101894 | Query complete 00:00:00.347

Fig.3 Actual data

	user_id [PK] bigint	screen_name text	created_at timestamp with time zone	followers integer
1	767	xeni	2006-07-14 05:49:14-04	152608
2	5039	jrome	2006-08-30 13:21:20-04	642
3	5242	ahmed	2006-09-04 00:18:12-04	149844
4	11364	Kits	2006-11-03 08:43:59-05	922
5	12094	Nicholai	2006-11-11 12:21:11-05	277
6	12301	absinthe	2006-11-13 22:14:36-05	856
7	18703	Kanikoski	2006-11-23 19:25:46-05	79
8	30833	nahnu	2006-11-29 18:49:03-05	1424
9	45353	thekillustrator	2006-12-06 14:23:06-05	121
10	45763	sasakifujika	2006-12-06 17:06:48-05	4412
11	106053	SPORT1	2006-12-21 11:14:55-05	588715
12	386513	profmusgrave	2006-12-31 07:18:57-05	33832
13	608583	Zeinobia	2007-01-06 21:21:50-05	248597
14	755504	AlexandraErin	2007-02-06 23:18:36-05	40642
15	759066	Galloway	2007-02-08 23:18:36-05	204
16	797975	thijs	2007-02-27 12:26:55-05	2460
17	804650	secretlyrobotic	2007-03-01 23:02:48-05	1791
18	808231	piemme	2007-03-03 11:00:57-05	191
19	810911	smangala	2007-03-04 22:21:30-05	268
20	812825	KitSeeborg	2007-03-05 18:38:19-05	3504
21	819522	vangeest	2007-03-07 21:12:06-05	21162
22	851211	benwikler	2007-03-09 22:36:12-05	132398

Total rows: 1000 of 80943 | Query complete 00:00:00.472

Fig. 4 User Metadata

Figure 5 and 6 show the resulting table available in the elasticsearch database after the tweets are uploaded through the stream process

Fig. 5 Tweets Data

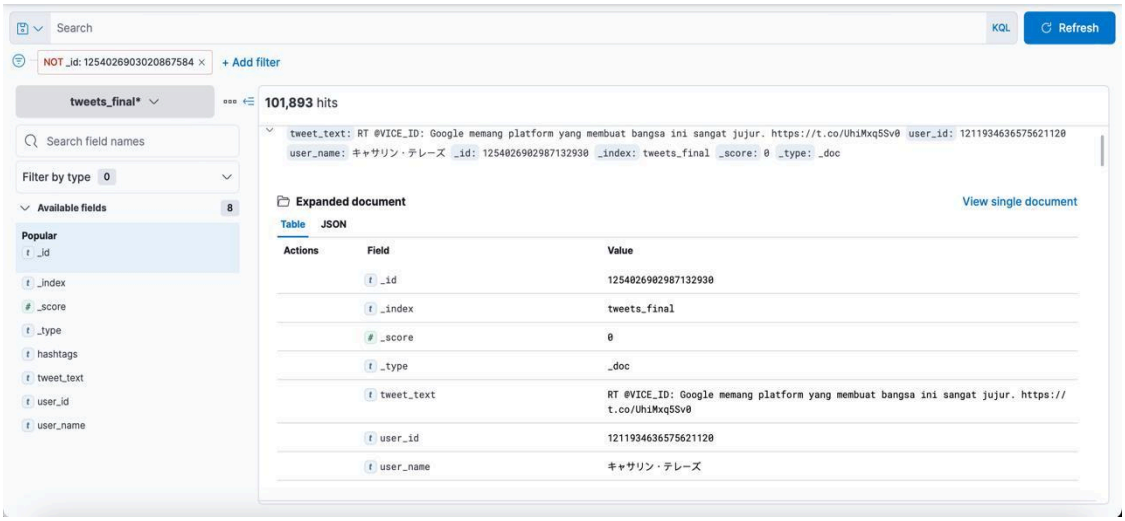

Fig. 6 Expanded Tweet Document

Figure 7 shows the GUI for the search application, and how you can choose how you want to filter the data. Figure 8, 9, and 10 then show the output when searching by User ID, outputting tweets made by the user, and allowing users to drill down to see more information about the User of the tweet.

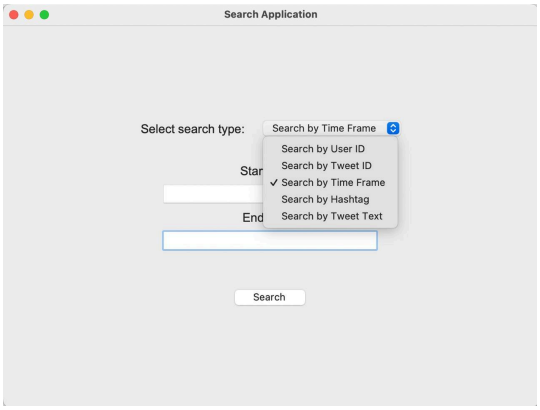

Fig. 7 User Search Interface

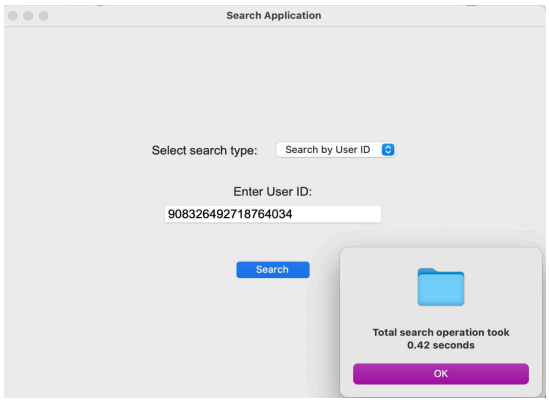

Fig. 8 Time Taken by search operation by User ID (Cached)

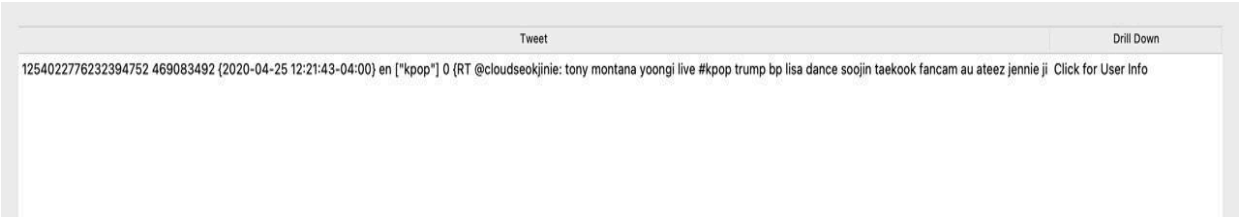

Fig. 9 Output of the above search operation

Fig. 10 Drill down Feature for getting the user information

Figure 11, 12, and 13 show a search by time-frame for tweets in a 13 minute time interval on April 25th 2020. The tweets in this time frame are not cached, so it takes longer to do the complete search.

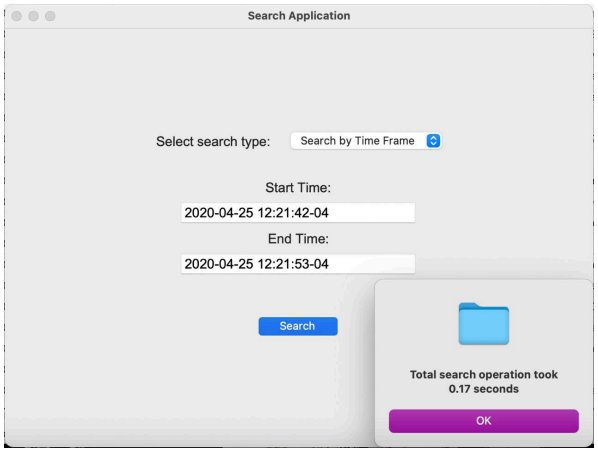

Fig. 11 Search Interface for search by Time Frame and Time Taken (non-cached)

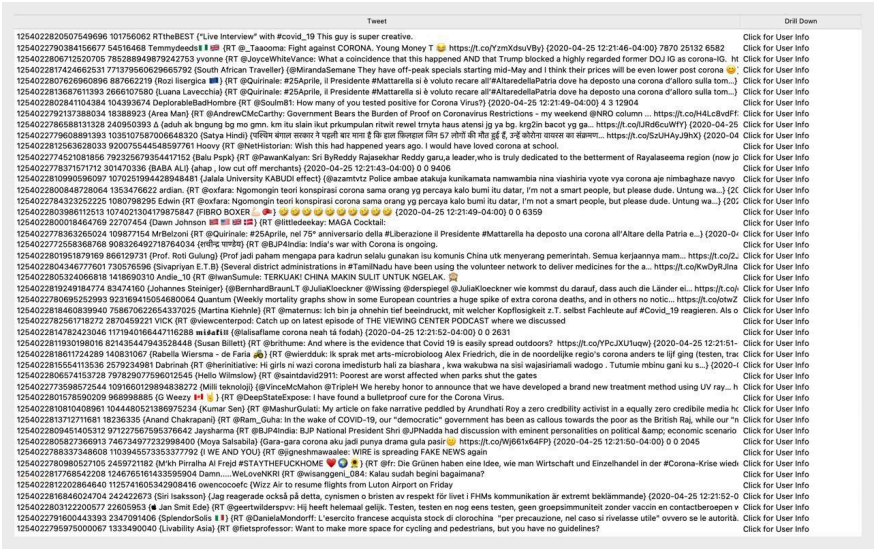

Fig. 12 Output of the above search operation

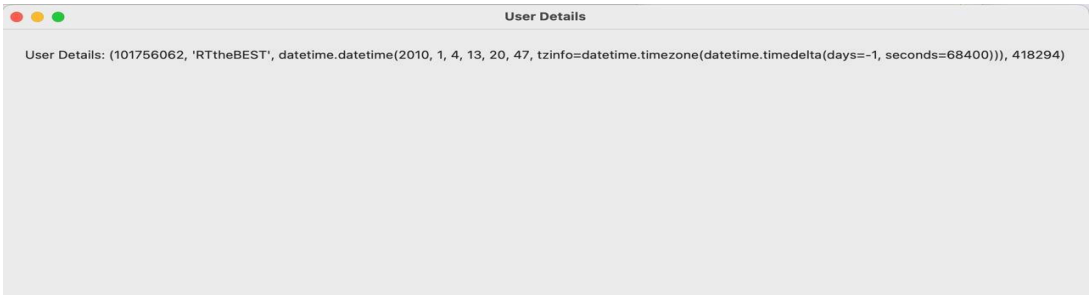

Fig. 13 Drill down Feature for getting the user information

Figure 14, 15, and 16 show a search by tweet id. Since you are searching by tweet id, only a single result will be returned if the tweet does exist.

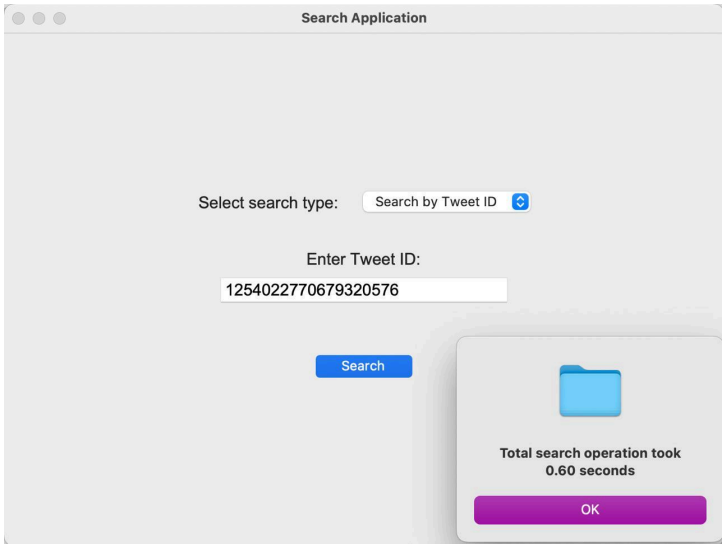

Fig. 14 Search Interface for search by Tweet ID and Time Taken (cached)

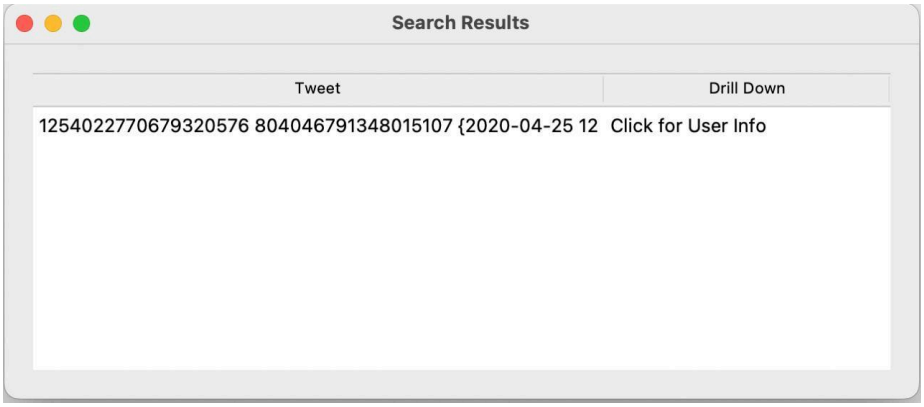

Fig. 15 Output of the above search operation

Fig. 16 Drill down Feature for getting the user information

Figure 17, 18, and 19 show a search by hashtag for the term kpop. Putting the term in quotes and in between brackets makes it so it only finds tweets with a single hashtag, in this instance being kpop. If a search is done without the quotations and brackets, tweets containing multiple hashtags can appear(but still must have kpop as one of them")

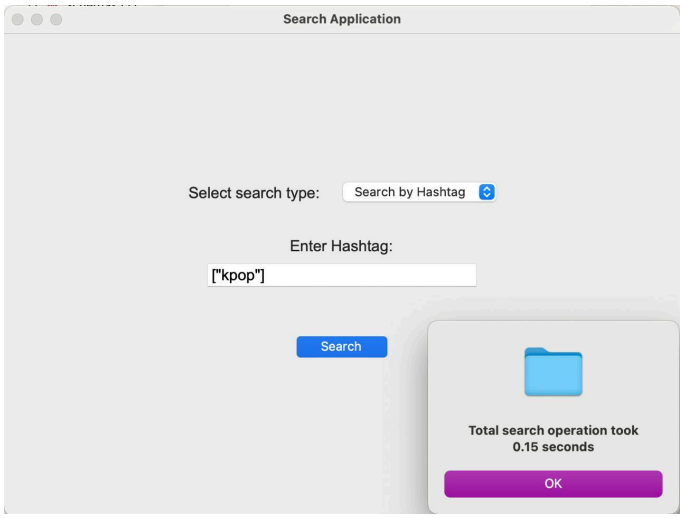

Fig. 17 Search Interface for search by Hashtags and Time Taken (cached)

Fig. 18 Output of the above search operation

Fig. 19 Drill down Feature for getting the user information

Figure 20, 21, and 22 show the last search type, searching by text, for the term “BJP”. The results of the search include tweets that just include the term “BJP”, rather than the whole tweet text being just “BJP”

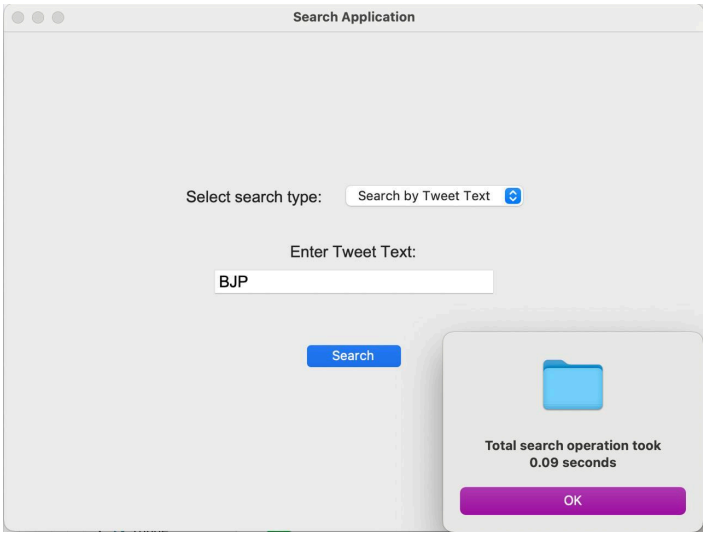

Fig. 20 Search Interface for search by Tweet Text and Time Taken (cached)

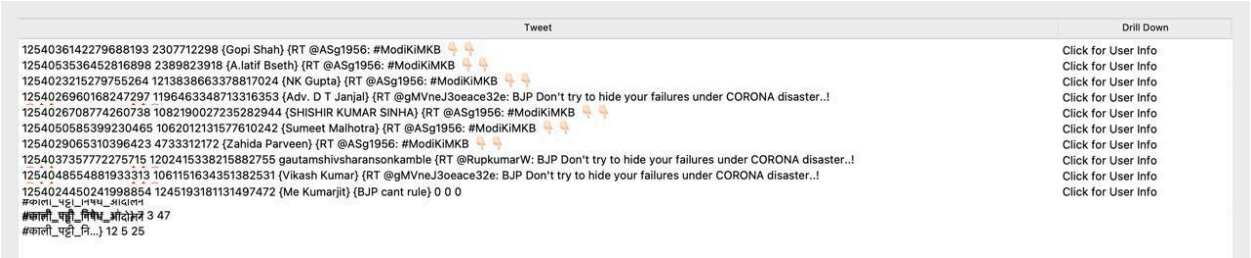

Fig. 21 Output of the above search operation

Fig. 22 Drill down Feature for getting the user information

8. Conclusion

The project closed with the development of a robust search application, effectively harnessing the Twitter dataset. The application facilitates diverse search capabilities alongside drill-downs, enabling an in-depth exploration of tweet connections. A dual-database approach ensured an optimal balance between structured data integrity and retrieval efficiency.

The implementation of a caching mechanism significantly reduced query latency, streamlining access to frequently requested data. This strategy functionally equates to caching frequent computational results, thus expediting retrieval times.

Insights into the role of initial design choices were gained, particularly their impact on the performance scalability of the application and the thoughtfulness that's needed to make it compact while maintaining low latency. Timing tests revealed opportunities for further optimization, underscoring the importance of efficiency in data retrieval processes.

Finally, the project showcased the effectiveness of meticulous planning in the creation of adaptable, high-performance applications. It highlighted the criticality of detail-oriented design in achieving an optimized user experience.

9. Contribution

- Yashwant Avula: Designed the front-end search interface and implemented drill-down functionality.
- Yash Sanghani: Developed the data curation application.
- Siddharth Yalamanchili: Integrated data from two databases for the search application and managed data ranking.
- Anton Vernikov: Conducted research on caching techniques and led the implementation of the caching system.

10. References

- For Elasticsearch- https://youtu.be/bcsII2dGnDU?si=7QY99qc8V4qNGPM4
- For Postgre- https://youtu.be/WxBfnGH3FsU?si=qcE_zOFq3y9NTP5h
- For Caching- Caching in Python Using the LRU Cache Strategy – Real Python
- For TThinker for Frontend Design- https://youtube.com/playlist?list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&si=ZcRvHRTxVJCxo5TQ