SEARCH APPLICATION FOR TWITTER DATA

Jahnavi Shah

Keshvi Gupta

Amaan Vora

Atharva Sherekar

03 May, 2023

–

Data Management for Advanced Data Science Applications

–

Dr. Ajita John

INTRODUCTION

Twitter data is rich in content and structure, providing a unique opportunity to analyze user behavior, sentiment analysis, topic modeling, and network analysis. A comprehensive search application for Twitter data is presented in this report, leveraging the capabilities of MongoDB and PostgreSQL to enable efficient data storage and retrieval. The project involves the collection and analysis of Twitter data, which is then stored in both a relational and a non-relational datastore, along with a caching mechanism for frequently accessed data. The search application is designed to enable users to search for tweets based on hashtags, users, and time range, as well as providing drill-down search options. Indexing and caching techniques are utilized to ensure swift access to data. The report provides a detailed methodology, encompassing data analysis, database design, and search application implementation, with findings from a test set of representative queries demonstrating the effectiveness of the approach. The project is a valuable resource for researchers and analysts seeking to explore and analyze Twitter data in an efficient manner.

DECODING DATA

In this project, the data can be categorized into two types - structured and unstructured data.

Users Data

The user data model comprises several attributes such as `user_id`, `name`, `screen_name`, `date`, `twitter_join_date`, `location`, `description`, `verified`, `followers count`, `friends_count`, `listed_count`, `favorites count`, and `language`, which provide valuable information about individual Twitter users.

In order to efficiently store and manage this structured user data, a PostgreSQL database was selected, which enables the creation of a flexible schema and allows for the effective management of large datasets.

Additionally, the historical data is being persisted in the database, with a flag indicating the most recent entry for each user. This approach facilitates accurate analysis of the dataset and enables the derivation of meaningful insights from the data, leading to the achievement of research goals.

Columns

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
user_id	character varying	255		✓	✓
name	text			✓	
screen_name	text			✓	
date	timestamp with time zone			✓	
twitter_join_date	date			✓	
location	text				
description	text				
verified	boolean				
followers_count	integer				
friends_count	integer				
listed_count	integer				
favourites_count	integer				
language	text				

Fig1. USER DATA

This model enabled storage of all relevant user information, facilitating easy querying for specific user details and trend analysis.

Firstly, PostgreSQL provides a robust and reliable data management system, with a strong emphasis on data integrity and consistency. Secondly, PostgreSQL has a powerful and flexible data model that enables the creation of complex data structures, making it an ideal choice for managing large datasets. Python was used for JSON data parsing and insertion into the database, along with creation of an "id" field index to enhance query performance.

user_id character varying (25!	name text	screen_name text	date timestamp with time	twitter_join_da date	location text	description text	verified boolean	followers_cou integer	friends_count integer	listed_count integer	favourites_count integer
3917836273	aniel-ani	ani_royal007	2020-04-11 05:34...	2015-10-16	India	॥जय शिव-राम-कृष्ण-हनिहनु ...	false	15584	10763	109	50435
2986689026	RC Shukl	RC_Shukl	2020-04-12 03:18...	2015-01-17	New Delhi (क्षेत्र...	Editor (Output) @AajTak ...	false	8920	309	21	24414
85660287032147...	Ychizzy young	ChizzyMoney	2020-04-06 01:58...	2017-04-24	Nigeria	afro_highLife musician f...	false	31	57	0	17
11486458039065...	HakanDogu	HakanDogu1166	2020-04-12 02:11...	2019-07-09	Paris, France	Senior Vice President Pa...	false	581	173	0	673
340959711	Thendo From Twi...	MuloiwaThendo	2020-04-12 05:49...	2011-07-23	012 | 015	Eudaimonia is the end g...	false	7067	986	2	9985
332106651	Markku Peltonen	MarkkuPeltonen	2020-04-12 00:47...	2011-07-09	Helsinki	Epidemiologi, tilastotiete...	false	7475	899	52	2132
17128975	CNN Indonesia	CNNIndonesia	2020-04-12 04:55...	2008-11-03	[null]	News We Can Trust. red...	true	881578	28	1213	98

Fig2. USER DATA

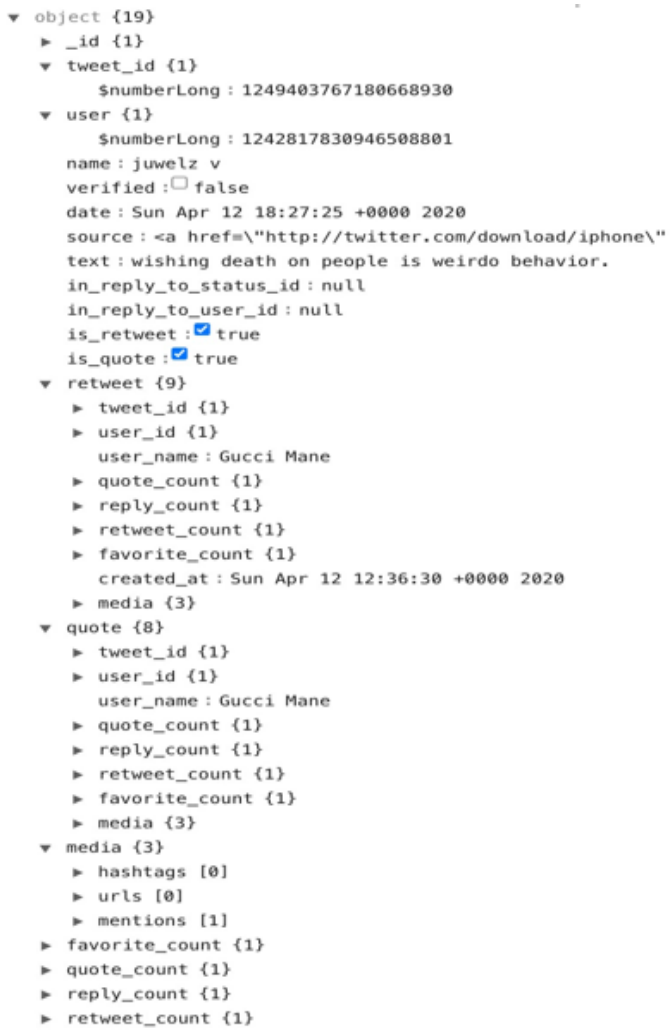
Tweets Data

In the tweet data, the user ID field represented a unique identifier assigned to each Twitter user account, which allowed distinguishing between different users. The user name field denoted the name associated with each user account, which provided further context about the individual. The text field

contained the actual content of the tweet posted by the user.

For convenience, two additional flags were introduced in the dataset, namely `is_retweet` and `is_quote`. These flags allowed easy identification for whether a tweet was a retweet or a quote tweet, aiding in further analysis. Additionally, if a tweet was a retweet or a quote tweet, the corresponding fields in the retweeted and quoted sections were also extracted, if present.

Fig 3&4. TWEET DATA



Since the tweet data was inherently unstructured and did not adhere to a rigid schema, a NoSQL database such as MongoDB was chosen for storage and management. MongoDB's flexible data model, scalability and performance capabilities enabled efficient processing and analysis of large volumes of tweet data, facilitating the extraction of valuable insights from the dataset.

Understanding the data

For the application, it was imperative that there was a significant understanding of the data being worked upon. A preliminary analysis was conducted to verify and rectify the data wherever needed. This preliminary analysis involved combing through the URLs, users, hashtags, retweets and latest data entry values for each user. Based on this, there was a flow established for the search application. Upon exploration, the copious amount of data values made computations seem technologically challenging, which required further intervention via indexing and partitioning.

INDEXING & PARTITIONING

To optimize the performance of data storage and querying, we utilized indexing techniques on both the structured user data and unstructured tweet data. For the structured user data, we employed a compound index on the name, follower count, and verified tag attributes, which allowed for faster data retrieval and reduced the need for full table scans or expensive filtering operations.

For the unstructured tweet data, we indexed the text, name, and date attributes to improve query performance. By indexing on the text attribute, we can quickly search for tweets containing specific keywords or phrases. Indexing on the name and date attributes enables fast retrieval of tweets posted by specific users or during specific time periods, which is often a crucial factor in social media analysis.

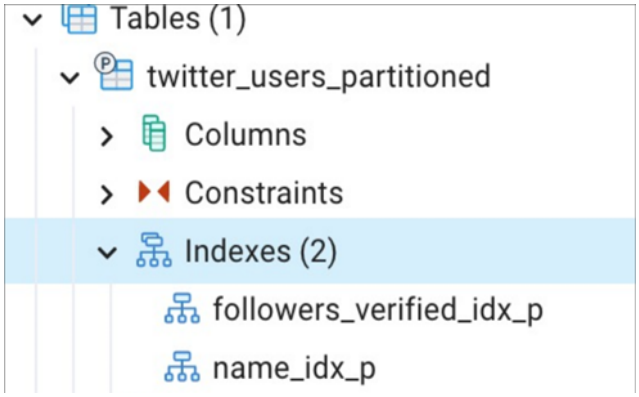


Fig5. INDEXING ON USER DATA

Fig6. INDEXING ON TWEET DATA

For the unstructured data, partitioning was implemented. Partitioning allowed the two large datasets to be divided into smaller, more manageable pieces or partitions, allowing for more efficient use of system resources. We partitioned our data into groups of hours, specifically 6 hours. This allowed for data to be distributed across 4 partitions, smaller datasets, making search queries easier. Partitioning also made it easier to manage the large dataset by allowing data to be logically grouped and isolated. If there existed a search query, depending upon the time of tweet to be searched, the system would know which dataset to access, thereby making the algorithm cost-effective and quick.

```
CREATE TABLE public.twitter_users_partitioned_1 PARTITION OF public.twitter_users_partitioned FOR VALUES FROM (0) TO (6);
CREATE TABLE public.twitter_users_partitioned_2 PARTITION OF public.twitter_users_partitioned FOR VALUES FROM (6) TO (12);
CREATE TABLE public.twitter_users_partitioned_3 PARTITION OF public.twitter_users_partitioned FOR VALUES FROM (12) TO (18);
CREATE TABLE public.twitter_users_partitioned_4 PARTITION OF public.twitter_users_partitioned FOR VALUES FROM (18) TO (24);
```

Fig7. PARTITIONING

By implementing these indexing and partitioning techniques, there was a significant improvement in the query performance of our Twitter data storage and retrieval, enabling efficient processing and analysis of the Twitter datasets. Additionally, caching mechanisms were also implemented to further optimize performance by reducing the frequency of queries to the database.

SEARCH APPLICATION

This particular search app allowed users to search for tweets based on various options such as `users`, `text`, `hashtags`, and `time range`. Users can also view top users and top hashtags based on certain criteria. Relevance of tweets has been demonstrated for showcasing rank of tweets in search results, and its methodology is attained with engagement.

The relevance index was calculated by developing the weighting scheme of 30% reply_count, 30% quote_count, 20% favorite_count, and 20% retweet_count to compute a score. This score, coupled with the number of followers, number of likes and number of retweets formed the basis of ranking for search results.

This approach was useful as it ensured that the search results were more relevant and useful to the user. Prioritizing engagement metrics over popularity ensured that tweets that were more likely to be of interest to the user were ranked higher in the search results. Sorting users based on followers count and verified status also helped identify reputable accounts that were more likely to have reliable and relevant information.

```
New entry, retrieving 'most popular users' from database!
Checkpoint saved!
Query took 0.5209 seconds
```

	user_id	name	twitter_join_date	location	verified	followers_count	friends_count	favourites_count
0	813286	Barack Obama	2007-03-05	Washington, DC	True	115603427	607612	11
1	18839785	Narendra Modi	2009-01-10	India	True	55786179	2364	0
2	807095	The New York Times	2007-03-02	New York City	True	46361159	904	18483
3	145125358	Amitabh Bachchan	2010-05-18	Mumbai, India	True	41596464	1833	75
4	101311381	Shah Rukh Khan	2010-01-02	None	True	40028019	77	32
5	471741741	PMO India	2012-01-23	India	True	34461808	486	0

Fig8. SEARCH RESULT OUTPUT

CACHING

Caching significantly improved the performance of this search application by storing frequently accessed data, such as the most engaging tweets, popular users, and hashtags. A cache size of 1000 items was used, and the least recently used items were removed using a Least Recently Used (LRU) algorithm to ensure that the most relevant data was always available.

To maintain the freshness of the cached data, a Time to Live (TTL) of 3600 seconds was set. This meant that after 3600 seconds, the cached data was considered invalid and must be refreshed from the source. To ensure that the search results were always up to date, a checkpoint was set for every 2 minutes, which involved refreshing the cache with the latest data based on the query.

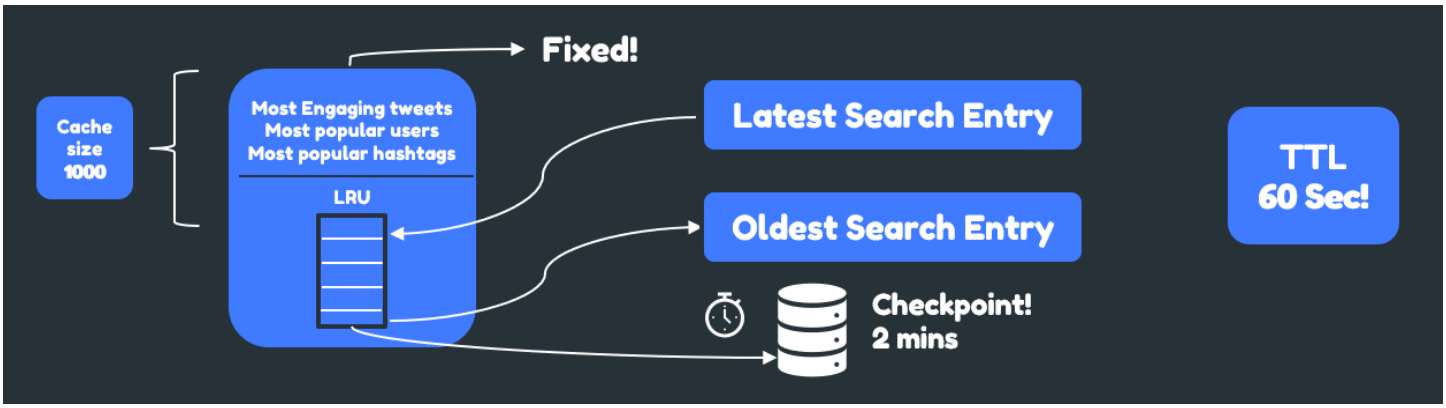


Fig9. CACHE ARCHITECTURE

By using caching in this way, the Twitter search application can provide fast and responsive search results, improving the overall user experience. Additionally, this approach helps reduce the load on Twitter's servers by reducing the number of requests for the same data, resulting in more efficient use of resources.

```
New entry, retrieving 'most popular users' from database! Retrieving 'most popular users' from cache!
Checkpoint saved!                                         Query took 0.0005 seconds
Query took 0.5209 seconds

New entry, retrieving 'Sözcü' from database! Retrieving 'Sözcü' from cache!
Checkpoint saved!                            Query took 0.0008 seconds
Query took 0.0536 seconds

New entry, retrieving 'most popular hashtags' from database! Retrieving 'most popular hashtags' from cache!
Checkpoint saved!                                            Query took 0.0000 seconds
Query took 0.6222 seconds
```

Fig10. CACHE vs NON-CACHE TIME COMPARISON

RESULTS

As seen from the images below, there are varied implementations of the search application. These implementations detail two separate categories - cache implementations and non-cache implementations (pulled straight from the database). There are implementations from the users database, stored in PostgreSQL and tweets database, stored in MongoDB. Additionally, the images also include cache loss for a particular search, which happens when the cache data is dumped after the TTL expires.

```
New entry, retrieving 'most engaging tweets' from database!
Query took 1.2433 seconds

 [{'tweet_id': 1254051230822944770,
   'user': 1039346340449452033,
   'name': 'Grace',
   'date': 'Sat Apr 25 14:14:47 +0000 2020',
   'text': 'But Joe, what if I WANT to drink bleach? What if I wanted to do that even before the orange man said to inject Lysol into our veins to stop
 corona? What if?',
   'retweet': None,
   'quote': {'tweet_id': 1253751812194070529,
    'user_id': 939091,
    'user_name': 'Joe Biden',
    'quote_count': 32237,
    'reply_count': 46159,
    'retweet_count': 263475,
    'favorite_count': 1280593,
    'media': {'hashtags': [], 'urls': [], 'mentions': []}},
   'retweet_count': 263475,
   'reply_count': 46159,
   'favorite_count': 1280593,
   'quote_count': 32237,
   'engagement': 332332.4},
```

Fig11. MOST ENGAGING TWEETS SEARCH

```
New entry, retrieving 'most popular hashtags' from database!
Checkpoint saved!
Query took 0.6222 seconds

[{'Corona': 9800},
 {'Mattarella': 3406},
 {'25Aprile': 3371},
 {'corona': 3263},
 {'AltaredellaPatria': 1829},
 {'COVID19': 1722},
 {'PideAlmayaDiyeÇıkıp': 1599},
 {'Liberazione': 1573},
 {'Covid_19': 1545},
 {'coronavirus': 1160}]
```

Fig12. MOST POPULAR HASHTAGS

```
Retrieving 'most popular hashtags' from cache!
Query took 0.0000 seconds

[{'Corona': 9800},
 {'Mattarella': 3406},
 {'25Aprile': 3371},
 {'corona': 3263},
 {'AltaredellaPatria': 1829},
 {'COVID19': 1722},
 {'PideAlmayaDiyeÇıkıp': 1599},
 {'Liberazione': 1573},
 {'Covid_19': 1545},
 {'coronavirus': 1160}]
```

Fig13. MOST POPULAR HASHTAGS-CACHE

```
New entry, retrieving 'Sözcü' from database!
Checkpoint saved!
Query took 0.0536 seconds
```

	user_id	name	twitter_join_date	location	verified	followers_count	friends_count	favourites_count
0	218078497	Sözcü	2010-11-21	İstanbul	True	2398838	28	0
1	3142919739	Sözcü Dünya	2015-04-07	İstanbul, Türkiye	False	39909	11	0
2	3501964461	Sözcü Ekonomi	2015-08-31	None	False	23393	13	0

Fig14. SEARCH BY NAME

```
New entry, retrieving '#corona' from database!
Checkpoint saved!
Query took 0.3121 seconds
```


```
[{'tweet_id': 1254052339301978112,
  'user': 304065893,
  'name': 'Paul Bocken (@ indien mogelijk)',
  'date': 'Sat Apr 25 14:19:11 +0000 2020',
  'text': "Wie maakt een filmpje om ook alle BN'ers te bedanken?\n#SARSCoV2 #Covid19 #corona https://t.co/DKtz9eMU3n"
  'retweet': {'tweet_id': 1253626273395507200,
   'user_id': 18078366,
   'user_name': 'Zara-Blue Exotic ',
   'quote_count': 2704,
   'reply_count': 816,
   'retweet_count': 14185,
   'favorite_count': 26702,
   'created_at': 'Fri Apr 24 10:06:09 +0000 2020',
   'media': {'hashtags': [], 'urls': [], 'mentions': []}},
  'quote': {'tweet_id': 1253626273395507200,
   'user_id': 18078366,
   'user_name': 'Zara-Blue Exotic ',
   'quote_count': 2704,
   'reply_count': 816,
   'retweet_count': 14185,
   'favorite_count': 26702,
   'media': {'hashtags': [], 'urls': [], 'mentions': []}},
  'hashtags': 'SARSCoV2',
  'retweet_count': 14185,
  'reply_count': 816,
  'favorite_count': 26702,
  'quote_count': 2704,
  'engagement': 9233},
```


Fig15. SEARCH BY HASHTAG

```
search_engine.cache.get_items()
```


```
[('most_popular_users',
  ('[{"user_id":"813286","name":"Barack Obama","twitter_join_date":1173052800000,"location":"Washington, DC","verified":true,"followers_count":1156034
27,"friends_count":607612,"favourites_count":11},{"user_id":"18839785","name":"Narendra Modi","twitter_join_date":1231545600000,"location":"India","ve
rified":true,"followers_count":55786179,"friends_count":2364,"favourites_count":0},{"user_id":"807095","name":"The New York Times","twitter_join_dat
e":1172793600000,"location":"New York City","verified":true,"followers_count":46361159,"friends_count":904,"favourites_count":18483},{"user_id":"14512
5358","name":"Amitabh Bachchan","twitter_join_date":1274140800000,"location":"Mumbai, India","verified":true,"followers_count":41596464,"friends_coun
t":1833,"favourites_count":75},{"user_id":"101311381","name":"Shah Rukh Khan","twitter_join_date":1262390400000,"location":null,"verified":true,"follo
wers_count":40028019,"friends_count":77,"favourites_count":32},{"user_id":"471741741","name":"PMO India","twitter_join_date":1327276800000,"locatio
n":"India","verified":true,"followers_count":34461808,"friends_count":486,"favourites_count":0},{"user_id":"113419517","name":"Hrithik Roshan","twitte
r_join_date":1265846400000,"location":null,"verified":true,"followers_count":28170371,"friends_count":90,"favourites_count":172},{"user_id":"9272467
7","name":"Virender Sehwag","twitter_join_date":1259193600000,"location":"India","verified":true,"followers_count":20571543,"friends_count":143,"favou
rites_count":4627},{"user_id":"405427035","name":"Arvind Kejriwal","twitter_join_date":1320451200000,"location":"India","verified":true,"followers_cou
nt":18339248,"friends_count":221,"favourites_count":618},{"user_id":"14293310","name":"TIME","twitter_join_date":1207180800000,"location":null,"verifi
ed":true,"followers_count":17057740,"friends_count":494,"favourites_count":536}]',
   1682757277.867035)),
```


Fig16. CACHE LOGS

CONCLUSION

The purpose of this report is to present a comprehensive search application for Twitter data using MongoDB and PostgreSQL for efficient data storage and retrieval. The report covers the collection and analysis of Twitter data, its storage in both relational and non-relational databases, and caching mechanisms for frequently accessed data. It also discusses indexing and partitioning techniques and their implementation to improve query performance. Additionally, the project provides a detailed methodology for implementing the search application and demonstrates its effectiveness through a test set of representative queries. This project provides an opportunity to work with both relational (PostgreSQL) and non-relational (MongoDB) databases, and explores indexing, partitioning, and caching strategies to optimize search application efficiency.

CONTRIBUTIONS

The whole team participated in initial brainstorming sessions where the team discussed about the structure of the data, which data should be kept in relation database and which should be kept in non-relational database. Team also came up with the ranking strategy based on engagement metric.

Amaan & Jahnvi : EDA, Report, Analytics, Logic development for segregating dtaa, implementing timing the queries, power point presenation
Atharva : Relational and non-relational data upload, partitioning, indexing, documentation
Keshvi : Searching, Caching

REPOSITORY:
https://github.com/Rutgers-Network/twitter-analysis2023-dbms694-team10