A Performance Analysis of Graph Databases over Social Network Datasets

Akul Gupta, Harshith Gundappa, Pramod Shenoy

Abstract

Relational databases are most popular due to their performance and scalability characteristics. Lately, due to the rapid influx of large networked datasets, graph databases have become a very popular tool to store and analyze them. The latest research on social media usage pattern (7) suggests that almost 58.4% of the world uses social media with average time spent per day around 2.5 hours. With such a staggering usage pattern, Facebook, which is the most popular of these social media sites, generates 4 Petabytes of data per day. To target this use case, we analysed the performance of graph databases and relational databases on Facebook and Twitter data. We run different types of queries that might be commonly used on user data as well as social circles. We also run aggregate queries such as top/least 'K' in popularity or size to see how graph databases handle queries that require to traverse the entire graph. Finally, we also show the relation of dataset size and query performance for the two and compare the disk space occupied by the systems. We show that Graph databases are indeed better than Relational databases for most social network analysis queries. Code can be found here.

1 Introduction

Relational databases have been very popular and ubiquitous in most scenarios, however, they do not really show relationships between data elements (10) (20). Due to having a predetermined fixed schema, they do not adapt well to changes. Any changes, if needed, take a significant amount of time. Graph databases are becoming a popular and natural way to model graph-like data. Their structure allows high levels of expressivity that can work for a large number of domains (17). When trying to store a graph model in a relational database, the following factors (15) become important:

- Modeling data with a large number of relationships
- Ease of expanding the model to add new data/relationships
- Speed and ease of querying the data

Figure 1: A sample Neo4j graph representation(18)

Over the past decade, social media sites and providers have taken the world by storm. As of January 2022, over half the world is active on social media and spends over 2 hours per day on it. Social media has enabled companies to form closer relations with current and prospective customers. From a purely marketing and business standpoint, companies are embracing the trend and need to actively collect and analyze social media data(7). They need sophisticated storage solutions that are specifically designed for such applications and graph databases might be the way forward.

In this project, we intend to evaluate the performance of Neo4j, a popular graph database and compare it with PostgreSQL, a popular RDBMS on social network graph data. We use Neo4j because it has been shown to be the most powerful (14) and the supported query language **Cypher** provides an intuitive and visual way to match patterns using an ASCII-art style syntax.

Our experiments are performed on SNAP's ego-Facebook (11) and ego-Twitter (12) datasets. We designed queries that could be used in a variety of applications, including aggregate and social queries. By running the same set of queries on the two database systems, we observe the performance difference and also the ease and intuitiveness of writing queries for the two platforms.

2 Related Work

Anita Brigit and Madhu Kumar (14) compare four types NoSQL databases – Wide Column Stores, Document Stores, Key-Value Stores, and Graph Databases based on a variety of factors. Graph databases leverage graph structures such as nodes, edges, and properties of relationships between entities to represent and store data in a graph-like data structure. Every element in the storage structure contains a pointer to its adjacent element(s). Graph databases typically feature optimized traversal over the structure without requiring expensive indexes, and can leverage algorithms such as shortest-path, A* to further accelerate traversal.

They evaluated several popular graph databases used in social networks - Neo4j, FlockDB, InfoGrid, OrientDB, and AllegroGraph. Their experiments concluded that Neo4j was the fastest at inserting nodes, relations, and edge-properties. Further, Neo4j and FlockDB exhibited the best overall read performance amongst compared databases. Therefore, Neo4j had the best overall performance characteristics. This in-part influenced our choice of Neo4j as the Graph DB used in our experiment.

The work of Mary Femy P.F, Reshma K.R, Surekha Mariam Varghese (16) compares the relative efficacy of MySQL, a relational database system, to Neo4j, a graph database, for the development of a provenance system. They provide a benchmarking mechanism to measure graph traversal operations. In their evaluations, they used queries that would simulate those used in provenance systems. These included traversals to determine nodes affected by some other starting or descendant node, searching for specific values within the payload, etc. They collected 100 data points for different dataset sizes and averaged the results. Using a query to find the number of students whose age is within a range, their experiment showed that Neo4j is almost 10x slower than MySQL. Their work concluded that graph databases are better when modeling complex networks and scale incredibly well for such datasets.

3 Methodology

3.1 Experimental Setup

We ran all our benchmarks on two identical *c6525-25g* Cloudlab(2) nodes with the following hardware specifications:

CPU	AMD 7302P
Cores	16
Clock Speed	3.0 GHz
Memory	128 GB ECC (8x 16 GB)

Table 1: Hardware specifications

We used the following Software configuration:

OS	Ubuntu 20.04 LTS
Kernel	5.4.0-107
Neo4j	community-4.4.5
PostgreSQL	14.2

Table 2: Software specifications

We ran our experiments on two different datasets:

	ego-facebook(11)	**ego-twitter(12)**
Nature	Undirected	Directed
Nodes	4,039	81,306
Edges	88,234	1,768,149

Table 3: Dataset details

4 Evaluation

In this section, we describe the different queries we executed, along with our observations for the two systems and datasets. In the context of Facebook, each user is connected to a 'Friend' and the term 'Follower' is used for Twitter but in a directed sense. A 'Follows' B does not imply B 'Follows' A.

4.1 Edge-connection Queries

These include some general queries which try to obtain information about a particular user's friend or follower list. Figures [4] and [5] show the queries and their running time in milliseconds. As expected, we see that Neo4j outperforms PostgreSQL by significant margins. Few of the edge-connection queries that we ran are as follows:

- People followed by users A and B
- People with higher than average number of followers
- 2nd circle friends of A
- Given K users, check if a person who is mutual friends with everyone, exists

Figures [2] and [3] show a couple of queries written in Cypher.

```
WITH '0' AS A, '59' AS B
MATCH (p:Person{id:A})-[:IS_FRIENDS_WITH]-(c:Person)
WHERE NOT exists ((c)-[:IS_FRIENDS_WITH]-(:Person{id:B}))
RETURN c;
```

Figure 2: Friends of A who are not friends with B

```
MATCH (p:Person)-[:IS_FRIENDS_WITH]-(c:Person)
RETURN p.id,count(c)
ORDER BY count(c) desc LIMIT 10
```

Figure 3: 10 Most Popular People

Figures [6] and [7] show some aggregate queries on the graph data. Here, we observe that Neo4j takes a lot more time to find the 10 most/least popular people. We theorize that this is because it has

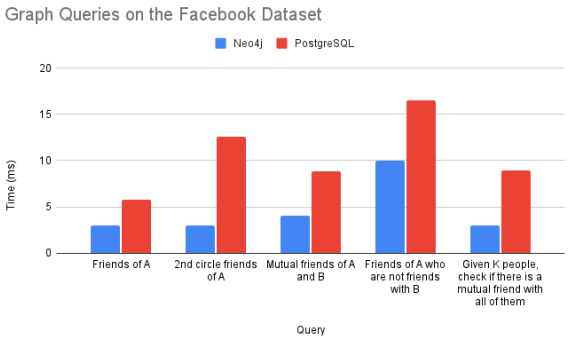

Figure 4

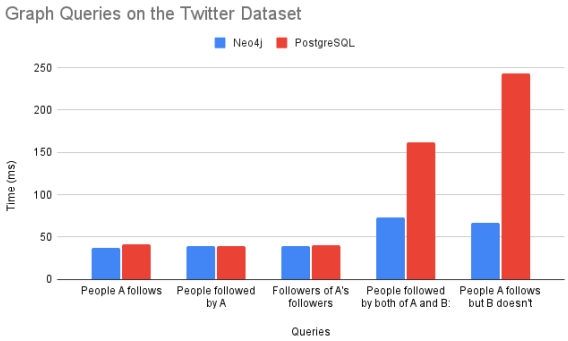

Figure 5

to traverse the entire graph in order to do so while PostgreSQL is simply designed to perform such queries efficiently. In a later experiment, we actually see that for queries such as this one, the size of the dataset becomes an important factor for Neo4j while its effects are negligible for PostgreSQL.

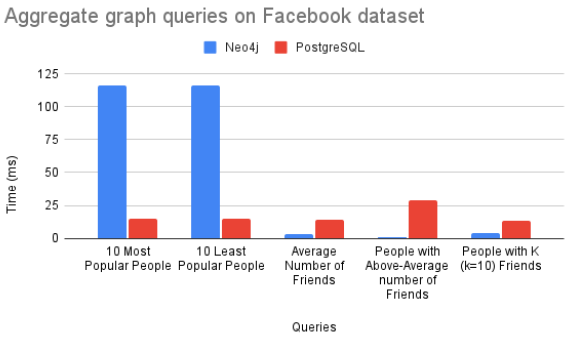

Figure 6

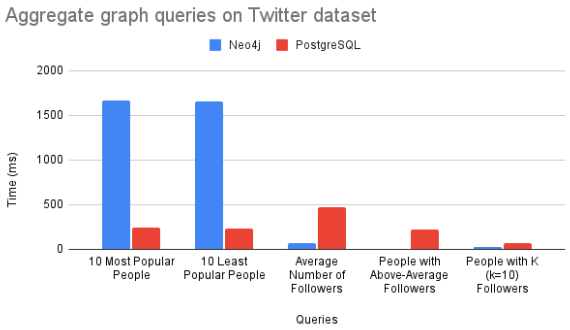

Figure 7

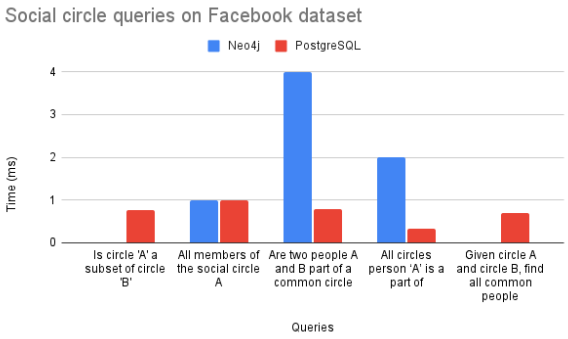

Figure 10

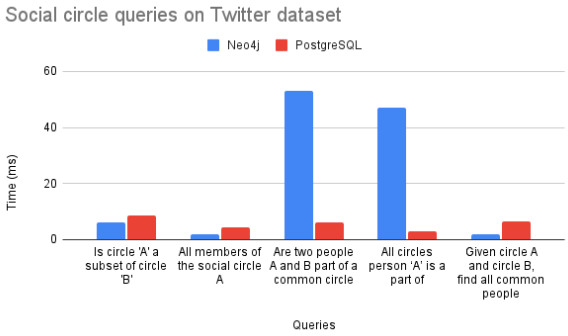

Figure 11

4.2 Social Circle Queries

Social Circles refers to 'Facebook Groups' in the ego-facebook dataset and 'Follower List' in the context of the ego-twitter dataset. There are multiple social circles in the dataset and the same user can belong to multiple groups. A few of the queries that we ran with respect to social circles are :

- All members of a social circle A
- Is circle 'A' a subset of circle 'B'?
- Are two people A and B part of a common circle?
- K largest social circles
- Given circle A and circle B, find all the common members

```
WITH 'circle25_1912' AS A, 'circle10_3437' AS B
MATCH (p:Person)-[:IS_PART_OF_CIRCLE]->(c:Circle{id:A}),
(r:Person)-[:IS_PART_OF_CIRCLE]->(d:Circle{id:B})
WITH collect(p.id) AS plist, collect(r.id) AS rlist
RETURN
CASE size([label IN plist WHERE label IN rlist])
WHEN size(plist) THEN "YES"
ELSE "NO"
END
```

Figure 8: Is Circle A a subset of Circle B?

```
WITH 'circle0_0' AS A, 'circle1_0' AS B
MATCH (p:Person)-[:IS_PART_OF_CIRCLE]->(:Circle{id:A}),
(p)-[:IS_PART_OF_CIRCLE]->(:Circle{id:B})
RETURN (p.id);
```

Figure 9: Given Circle A and B, find all the common members

Figures [8] and [9] show a couple of Cypher queries that were run for social circles.

As shown in Figure[10] and Figure[11], Neo4j outperforms PostgreSQL for most of the queries. Similar to edge-connection queries, we added aggregate queries for social circles too. The plots in figures[12] and [13] capture our observations. Like in the case of edge-connection aggregate queries, Neo4j took considerably more time for K-largest circles and K-smallest circles than PostgreSQL.The difference was more pronounced in the Twitter dataset. The massive scale of the ego-twitter dataset compared to ego-facebook seems to be playing a role here.

4.3 Query Planning

The first run of the query on Neo4j will be slower than subsequent runs, as the query will have to be planned, and the part of the graph touched may not be in the page-cache. To evaluate this, we ran eleven social circle queries (shown in Fig [15]) on

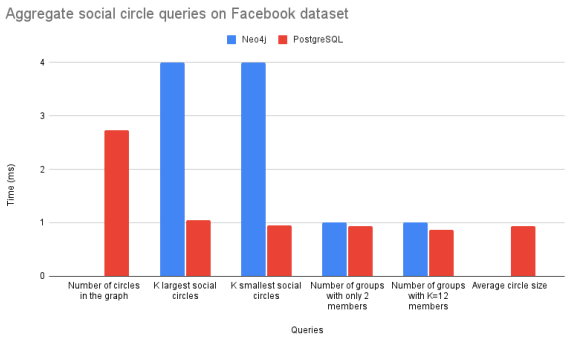

Figure 12

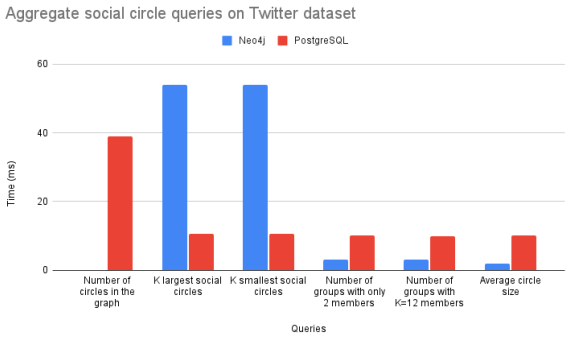

Figure 13

the ego-twitter dataset and captured the first and subsequent run times. Figure[14] shows our results. Blue bars denote the first execution time and the red bar represents subsequent runs. On average, we noticed a 10% difference in query execution time between the first and subsequent runs of a query.

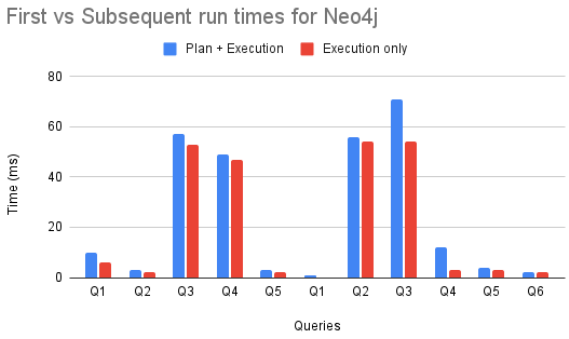

Figure 14

Q1	Is circle 'A' a subset of circle 'B'
Q2	All members of social circle 'A'
Q3	Are two people A and B part of a common circle
Q4	All circles person 'A' is a part of
Q5	Given circle A and circle B, find all common people
Q6	Number of circles in the graph
Q7	K largest social circles
Q8	K smallest social circles
Q9	Number of groups with only 2 members
Q10	Number of groups with K=12 members
Q11	Average circle size

Figure 15

4.4 Dataset Size

We conducted the following experiment to measure the effect of the dataset size on the execution time of the queries. We ran several iterations of '10 Most Popular People' query. In each of the iterations, we deleted 100K edge connections from the ego-twitter dataset and measured the runtime of the query. Figure[16] shows our results. On both Neo4j and PostgreSQL, the query-time reduced as the edges were deleted. The change in the runtime was significantly more pronounced with Neo4j. A 50% reduction in the number of edges led to a 45% drop on Neo4j while on PostgreSQL, we only noticed a fall of 15%.

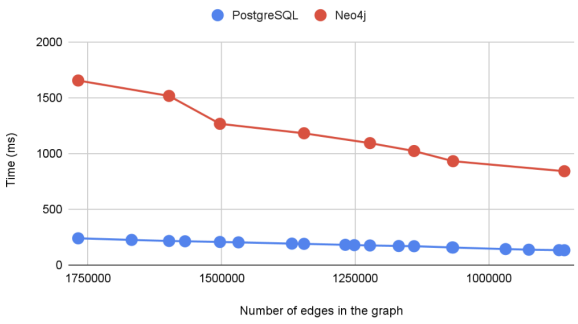

Figure 16: Query Run:10 most popular people

4.5 Storage Comparison

Another aspect in which we measured the difference between Graph and Relational databases was storage.Table [4] shows the on-disk memory usage of both Twitter and Facebook datasets for Neo4j and PostgreSQL. The bi-directional nature of ego-facebook's connections means that PostgreSQL needs to store both (user, friend) and (friend, user), while a single undirected edge (user, friend) suffices for Neo4j. We suspect this to be the reason behind 50% difference between Neo4j and PostgreSQL for the Facebook dataset. Directional graphs such as Twitter, do not get this advantage, and therefore, we see similar storage spaces on both the databases.

Dataset	Neo4j	PostgreSQL
ego-facebook	6.6	12
ego-twitter	76	72

Table 4: Disk Storage(in MB)

4.6 Ease of Writing the Queries

The ease with which queries can be written is an important factor for any query language. Unlike SQL which was designed for traditional relational database applications, Cypher(Neo4j's query language) was built specifically for graph traversals. Therefore, compared to SQL, Cypher graph queries are intuitive, graph-aware, and easy-to-write. Cypher provides a visual way of matching patterns and relationships. It uses rounded brackets(NODES) to denote nodes and -[:ARROWS] for relationships. Writing a query in Cypher is like drawing a graph pattern through the data. For example, query :- (nodes)-[:ARE_CONNECTED_TO]->(otherNodes) returns all the nodes that are connected to any other node in the graph. Note that the relationships in Cypher are directional. This makes it easy to handle directed graphs as well.

4.7 Security

PostgreSQL has extensive multi-user support. Neo4j, on the other hand, does not have any built in mechanisms for managing security restrictions and multiple users(9). It presumes a trusted environment. Although Access Control List based security mechanisms are present, the management is handled at the application layer.

4.8 Schema Flexibility

Although relational databases are more secure compared to graph databases, a limitation of RDBMS systems is it's fixed schema. This makes it difficult to extend the database and less suitable to manage adhoc requirements that evolve over time. On the contrary, adding new relationships to Neo4j graphs is extremely simple.

5 Future Work

While our current experiments only compared Graph databases against Relational Databases, it would be useful to also compare the performance against other types of NoSQL databases such as Wide-Column stores like Apache Cassandra (1), and Document stores like MongoDB (4).
We would also like to consider different storage schema designs on Relational databases to consider how they impact performance with respect to different query types.

6 Conclusion

We sought to evaluate the potential performance benefits that Graph Databases might offer over traditional Relational Databases systems. Our work evaluated and compared the performance, storage, and other characteristics between Neo4j (18) and PostgreSQL (21) across two different social-network datasets (11) (12). We observed that Neo4j outperformed PostgreSQL at most queries with the exception of certain aggregate queries such as top 'K' which required traversing all nodes and edges. Overall, transitioning to Graph Databases can offer significant performance benefits if the data is graph-like and the workload predominantly consists of queries that do not require traversing the entirety of the graph. The move to a Graph DB can also

offer sizable storage space benefits if the edges are undirected.

References

[1] Apache cassandra. URL: https://cassandra.apache.org/_/index.html.

[2] Cloudlab. URL: https://cloudlab.us.

[3] Fb data generation. URL: https://techjury.net/blog/how-much-data-is-created-every-day/.

[4] Mongodb. URL: https://www.mongodb.com.

[5] Mysql database. URL: https://www.mysql.com.

[6] Oracle database. URL: https://www.oracle.com/database/technologies/.

[7] Social media stats. URL: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/.

[8] Soad Almabdy. Comparative analysis of relational and graph databases for social networks. In *2018 1st International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–4, 2018. doi:10.1109/CAIS.2018.8441982.

[9] Shalini Batra and Charu Tyagi. Comparative analysis of relational and graph databases. 2012.

[10] Neo4j Staff Editor. 5 sure signs it's time to give up your relational database. URL: https://neo4j.com/blog/five-signs-to-give-up-relational-database/.

[11] Stanford University Jure Leskovec. Social circles: Facebook. URL: https://snap.stanford.edu/data/ego-Facebook.html.

[12] Stanford University Jure Leskovec. Social circles: Twitter+. URL: https://snap.stanford.edu/data/ego-Twitter.html.

[13] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/7a614fd06c325499f1680b9896beedeb-Paper.pdf.

[14] Anita Brigit Mathew and S. D. Madhu Kumar. Analysis of data management and query handling in social networks using nosql databases. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 800–806, 2015. doi:10.1109/ICACCI.2015.7275708.

[15] Dave Packer. How using graph technology improves performance and business agility. URL: https://neo4j.com/whitepapers/overcoming-sql-strain-graph-databases/?ref=blog.

[16] Mary P.F, Reshma K.R, and Surekha mariam varghese. Outcome analysis using neo4j graph database. *International Journal on Cybernetics Informatics*, 5:229–236, 04 2016. doi:10.5121/ijci.2016.5225.

[17] Haikal Pribadi. The challenges of working with a graph database. URL: https://blog.vaticle.com/the-challenges-of-working-with-a-graph-database-2a5f9a7c903b.

[18] Walker Rowe. Introduction to the neo4j graph database. URL: https://www.bmc.com/blogs/neo4j-graph-database/.

[19] Bryce Merkl Sasaki. Graph databases for beginners: Why a database query language

matters (more than you think). URL: https://neo4j.com/blog/why-database-query-language-matters/.

[20] Bryce Merkl Sasaki. Graph databases for beginners: Why connected data matters. URL: https://neo4j.com/blog/why-graph-data-relationships-matter/?ref=pdf-white-paper-sql-strain.

[21] M. Stonebraker, L.A. Rowe, and M. Hirohama. The implementation of postgres. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):125–142, 1990. doi:10.1109/69.50912.