ISSN:2321-1156

International Journal of Innovative Research in Technology & Science(IJIRTS)

REDUCING TIME COMPLEXITY OF RELATIONAL DATABASES USING NEO4J

MS. MEENU DAGAR, AITM, Palwal, MR. MAHESH SINGH, AITM Palwal

Abstract

Relational databases have been around for many decades and are the database technology of choice for most traditional data-intensive storage and retrieval applications. Retrievals are usually accomplished using SQL, a declarative query language. Relational database systems are generally efficient unless the data contains many relationships requiring joins of large tables. Recently there has been much interest in data stores that do not use SQL exclusively, the so-called NoSQL movement. Examples are Google's BigTable and Facebook's Cassandra. Relational model has been dominating the computer industry since the 1980s mainly for storing and retrieving data. Lately, however, relational database is losing its importance due to its dependence on a rigid schema which makes it difficult to add new relationships between the objects. One of the proposed solutions is to shift to the Graph databases as they aspire to overcome such type of problems. This paper reports on a comparison of one such NoSQL graph database called Neo4j with a common relational database system, MySQL, for use as the underlying technology in the development of a software system to record and query data provenance information.

Keywords:- Flexibility, Maturity, Security, Retrieval.

Introduction

Relational Databases have been providing the storage support for many decades now with implementations like Oracle, MySQL, etc [1]. Way back people used database just for storing tabular data like purchase reports and finance records. Relational databases were perfect as one could associate a transaction in finance table with an item in purchase table. But in today's environment, these relational representations are not efficient in performing many operations for example the World Wide Web exhibits far more complicated networks of relationships than were expected when SQL was designed. The network of hyperlinks connecting all the pages on the World Wide Web is highly complex and almost impossible to model efficiently in a relational database [2]. Similar issues are involved in modeling

the social network like Twitter, Facebook, etc. Implementing such problems in relational databases involves large number of joins which is expensive to be calculated [3]. With the intense increase in usage of internet leading to need for storing large amounts of interconnected data, there was a clear desire for a data store tailored to the needs of graph data. Graph databases are optimized for these types of networks (social networking and website link structure), as graph is a natural way of storing connections between users. Relational databases are not useful when the data model evolves over time [4], which means relational databases depend on rigid schema and make it difficult to add new relationships between objects. All these limitations of relational databases led to the invention of graph databases.

A. Graph Databases

Graph Database is a database system where the relationships between objects or entities are equally as important as the objects themselves [5]. In a graph database, data is represented by nodes, edges and properties. Nodes are represented as objects and edges manifest the relationship between nodes. There are several implementations of graph database. Both nodes and edges can have properties that depict their specific charactersticts. Some of the best known graph databases are: Infogrid, HypergraphDB, Jena, DEx, FlockdB and Neo4j[6]-[8]. Out of these only Neo4j is discussed here.

B. Neo4j

Neo4j is a high performance, robust and scalable graph database solving queries with multiple relationships storing data in the nodes and relationships [9]. Neo4j is fully written in Java and can be deployed on multiple systems [10]. It is comprised of two parts(1)- A client that sends commands to the server via RMI.(2)- A Server that processes these com-

ands and sends back the processed result to the client.

The database is queried through Cypher Query Language. It is a new query language that has been recently added to the Neo4j. Unlike imperative languages like Java and scripting language like Gremlin and the Ruby Neo4j bindings, Cypher is a declarative language. Using Cypher, efficient querying of the graph is possible, without having to write traversers in the code.

Evaluation Parameters for Relational Databases and Graph Databases

The evaluation between MySQL and Neo4j is based upon different criteria [4]. These criteria are the pillars to decide which database should be adopted for implementation.

A. Level of Support/Maturity

Maturity refers to how well tested the system is. If a system has been tested, more number of times, it means it is more stable and more bugs have been found out. Maturity of a system is proportional to level of support.

Relational Databases have been providing storage support for decades now. So they are more stable and mature. Both Oracle and MySQL have been providing extensive support for their commercial products. Relational databases have a unified language SQL. As SQL does not differ much between implementations, support for one implementation is applicable to others as well. Since Neo4j version 1.1 was released in February 2010, it is less stable and less mature. It lacks a unified language to interact with the database as query languages supported by Neo4j (SPARQL, Gremlin and Cypher Query) differ much in implementation. Support for one implementation is not applicable to all others. Neo4j is still growing and maturing and has not undergone the same rigorous performance testing as relational databases. Most of the support comes from its parent company's website www.neo4j.org and is limited from outside of Neo4j site.

B. Security

MySQL has extensive multi user support. However Neo4j does not have any built in mechanisms for managing security restrictions and multiple users [11]. It presumes a trusted environment. Although there is Access Control List security mechanisms but even Access Control List management is handled at application layer. On the other hand, there is extensive support for ACL based security in MySQL.

C. Flexibility

Although relational databases are more mature and secure as compared to graph databases, but its schema is fixed, which makes it difficult to extend these databases [12] and less suitable to manage ad-hoc schemas that evolve over time. This can be comprehended with the help of an example, suppose we have following information.

1) Marko is a human and Fluffy is a dog.
2) Marko and fluffy are good friends.
3) Human and dog are subclass of mammal.

In Relational Databases this information can be expressed as given here(table1, table2 and table3):

Table 1 Object Table

ID	NAME	TYPE
001	Marko	Human
002	Fluffy	Dog

Table 2 Friendship Table

ID1	ID2
001	002
002	001

TYPE1	TYPE2
Human	Mammal
Dog	Mammal

Table 3 Subclass

In graph databases, the same information is represented as given in Fig1:

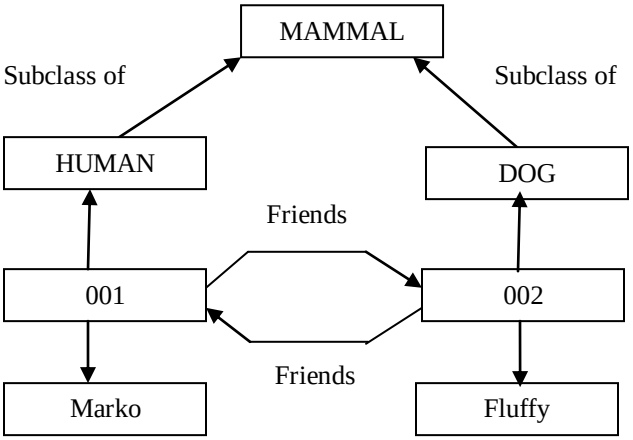


Fig 1 Representation in graph database

Now if we want to add some new information for example:

- Marko and Fluffy are both mammals.

Storing this information in the relational database needs to restructure the entire relational database schema (table4, table5, table6 and table7).

Table 4Object Table

ID	NAME	TYPE
001	Marko	Human
002	Fluffy	Dog

Table 5 Friendship Table

ID1	ID2
001	002
002	001

Table 6 Subclass Table

TYPE1	TYPE2
Human	Mammal
Dog	Mammal

Table 7 Type Table

ID	TYPE
001	Human
002	Dog
001	Mammal
002	Mammal

However in graph databases, there is no need to restructure the entire schema every time a new relationship is added; only a few edges and nodes are added to the graph. For example the same information can be easily added in the graph databases without any restructuring of the original graph (Fig 2).

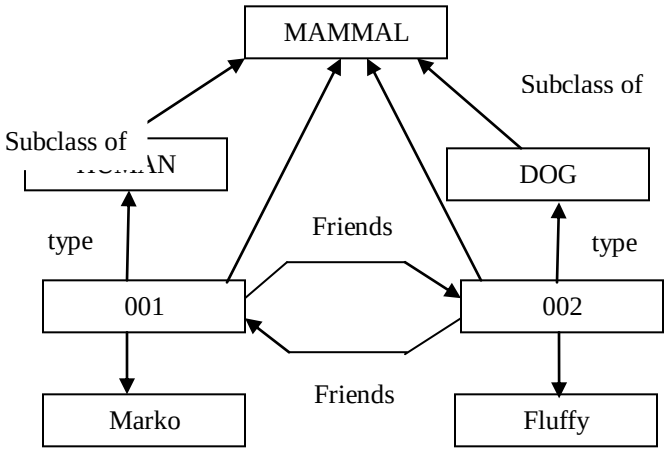


Fig 2 Modified Representation in Graph Database

Hence, Neo4j has an easily mutable schema while Relational databases are less mutable.

It has been theoretically said that relational model works best when there are a relatively small and static number of relationships between objects. When the data sets become larger, they require expensive join operations because they search all of the data to find the data that meets the search criteria. The larger the data set, the longer it takes to find matches. Conversely, a graph database does not scan the entire graph to find the nodes that meet the search criteria. It looks only at records that are directly connected to other records, increasing the number of nodes which does not increased the retrieval time. To prove this an experiment has been conducted and results have been tabulated.

Implementation Details

The evaluation between MySQL and Neo4j is based upon a set of predefined queries.

To implement relational databases, MySQL version 5.1.41 was used. The database was queried using PHP scripting language. Graph databases are implemented using Neo4j Community version 1.6. The database is queried with Cypher Query Language.

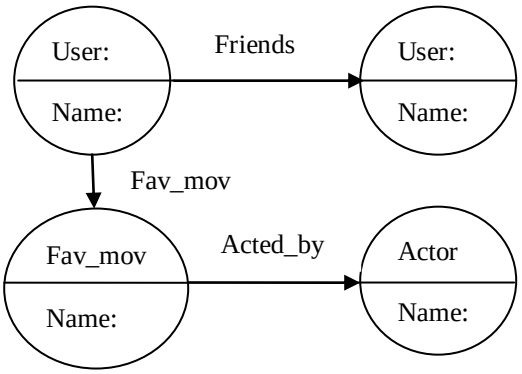
Queries were designed to analyze the performance difference between a relational database and a graph database.

Schema for relational database included the following tables:

1) User: user_id, user_name
2) Friends: user_id, friend_id
3) Fav_movies: user_id , movie_name
4) Actors: movie_name, actor_name

Schema for graph databases is represented in Fig 3:

Fig 3 Graph Database Schema



The three queries defined were:
S0: Find all friends of Esha.
S1: Find the favorite movies of Esha's friends.
S3: Find the lead actors of Esha's friends favorite movies.

Experimental Results

Execution times were collected after executing the queries and noted in milliseconds. The results have been tabulated in table 8. It can be easily observed from the values retrieved (Table 8.) that the retrieval times of graph databases is less than relational databases. This is because relational databases search all of the data to find the data that meets the search criteria. The larger the data set, the longer it takes to find matches, so when number of users get increased from one hundred to five hundred, the retrieval time gets increased manifold. On the other hand graph database looks only at records that are directly connected to other records; it does not scan the entire graph to find the nodes that meet the search criteria. So, increasing number of nodes from one hundred to five hundred does not increase the retrieval time much as can be visualized from the graphs (Fig 4 and Fig 5).

Table 8 Query Results In Milliseconds

No_of_objects	MySQL:S0	Neo4j:S0	MySQL:S1	Neo4j:S1	MySQL:S2	Neo4j:S2
100	19.56	8	33	12.65	111.334	19.57
500	281.38	10	333.96	17	620.56	21

Fig 4: Retrieval times of queries by neo4j and mysql (100 objects)

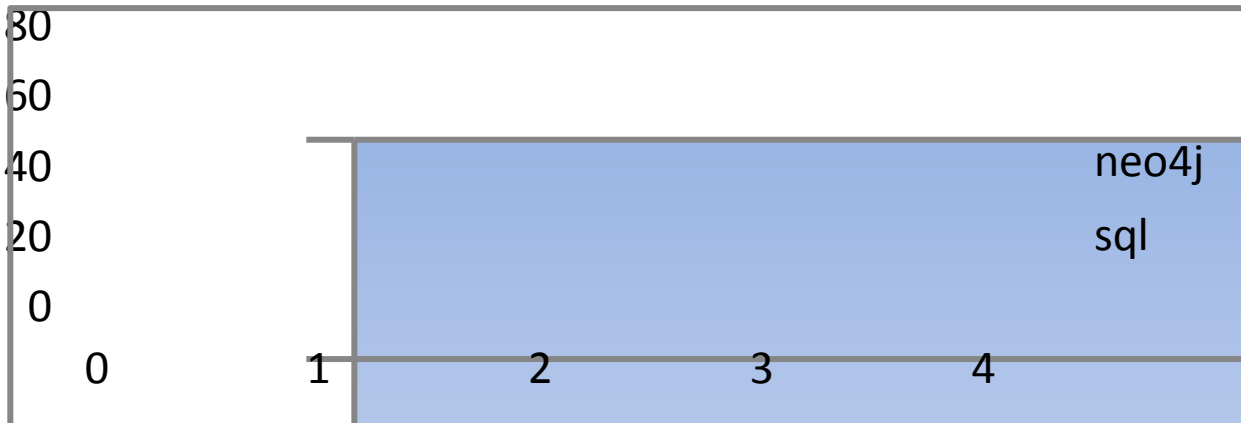
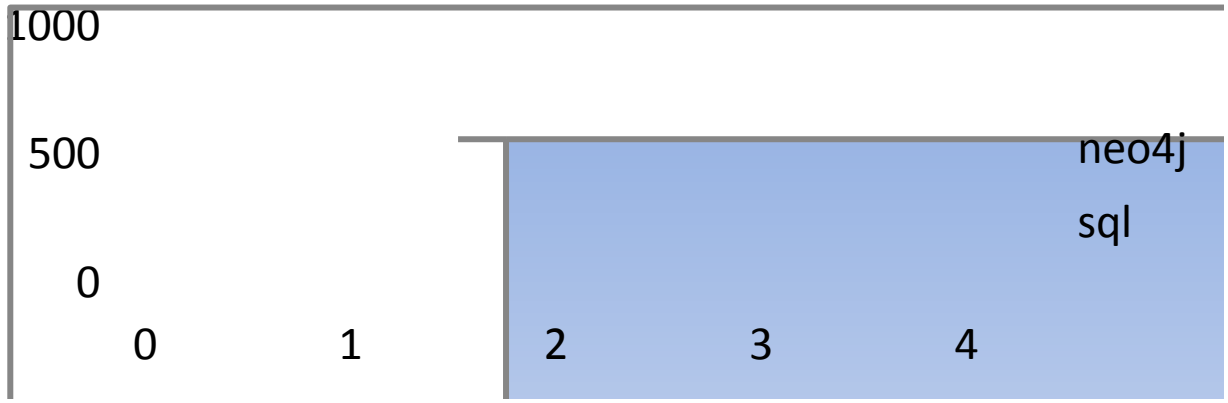


Fig 5: Retrieval times of queries by neo4j and mysql (500 objects)



CONCLUSION

Both systems performed well on the objective and subjective benchmarks. In general, graph databases performed better when objective tests were performed. This implies that graph databases retrieve the results of the set of predefines query faster than relational databases. Not only this, graph databases are more flexible than relational databases as new relationships can be added to graph databases without the need to restructure the schema again. With such a difference in the query retrieval time of MySQL and Neo4j, Neo4j can be used for commercial purposes like website link structures and social networking.

References

[1] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, Dawn Wilkins "A Comparison of a Graph Database and a Relational Database", ACM Southeast Regional Conference,2010.

[2] Database Trends And Applications. Available http://www.dbta.com/Articles/Columns/Notes-on-NoSQL/Graph-Dat abases-and-the-Value-They-Provide-74544.aspx,2012

[3] M. Kleppmann. Should you go beyond relational databases? Available http://carsonified.com/blog/dev/should-you-go-beyond-relational-dat abadat.

[4] Neo4j Blog, Available http://blog.neo4j.org/2009/04/current-database-debate-and-graph.html.

[5] M. I. Jordan (ed). (1998) ."*Learning in Graphical Models*". MIT Press. [6] HypergraphDB website, Available http://www.kobrix.com/hgdb.jsp

[7] Jena documentation, Available http://jena.sourceforge.net/documentation.html

[8] Infogrid. Blog, Available

http://infogrid.org/blog/2010/03/operations-on-a-graph-databae-papa -4.
[9] Neo4j. Home. http://neo4j.org, 2012.
[10] D. Dominguez-Sal, P. Urb´on-Bayes, A. Gim´enez-Va˜n´o, S. G´omez-Villamor, N. Mart´ınez-Baz´an, and J.L. Larriba-Pey, "Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark", Proceeding WAIM'10 Proceedings of the 2010 international conference on Web-age information management.
[11] T. Ivarsson. [neo] security, Available http://lists.neo4j.org/pipermail/user/2009-November/001955.html, 2011.
[12] R. Angles and C. Gutierrez," Survey of graph database moDels",. ACM Comput. Surv., 40(1):1–39, 2008. [13] Neo4j. The neo database (2006), Available http://dist.neo4j.org/neo-technology-introduction.pdf

Biographies

AUTHORS 1)

Meenu Dagar

Presently pursuing her Master of Technology in Computer Science and Engineering from Maharshi Dayanand University. She has done Master of Computer science from Banasthali University.
Email id: **dagar.meenu22@gmail.com**

Mahesh Singh

Presently working as an Assistant Professor in AITM (Advanced Institute of Technology and Management)Palwal. He has done his Master degree from Delhi University.His teaching and research area is Database.
Email id: **mahesh100nucs@gmail.com**