Ease of Representing Data Through Graph Databases

A. Vijay Kumar, Dr. G. Anjan Babu

***Abstract*— Relational Database is the term which ruled the Database era for past 3 decades. The ruling portion of Relational Database was started shrinking from early 21st century due to some advancements taken place in the database realm. These advancements turned as drawbacks to the Relational Database, and introduced a new term called BigData to the database realm. The database designer's answer to deal with this new term is NoSql Databases. One of the popular NoSql database is Graph Database, whose base is nodes and edges. Natural way of representation is the main asset of Graph Databases which allowed to increase the performance over Relational databases. A vital focus was given to Neo4j, the popular Graph Database. This paper is going to compare the performances of present and past database solutions called Relational and Graph Databases.**

Keywords*— *Graph Database; NOSQL; performance.

I. INTRODUCTION

Relational Databases were came into picture in the 1980's with an unknown researcher at IBM, which solved lot of issues involved with the database field. Most of the data during that period was structured which makes the Relational Databases efficient to tackle. One more reason to their success is most of the data introduced by web and various sources was structured. Relational databases make use of such structured data to store in the form of tables very easily. It is going to use the concept of rows and columns to store and retrieve the data which was stored in the tables. For Relational Databases, it is difficult to deal with connected data, which has been introducing by most of sources now a days. This is due to JOIN operation which must be used to deal with connected data. Some limitations like this brought the concept called NoSql databases like Key Value stores, Column-Family stores, Document Databases and Graph Databases the topic of the paper. The major focus was cornered to Neo4j Graph Database from NoSql Databases.

II. BIG DATA MOVEMENT

Big Data was the result of so many activities done in the past years as shown in the below figure. Big Data refers to datasets whose size are beyond the ability of typical database software tools to capture, store, manage and analyses. There is no explicit definition of how big a dataset should be in order to be considered Big Data[1]. The inabilities of traditional databases and due to lack of features driven the concept of BigData that was even unable to visualize.

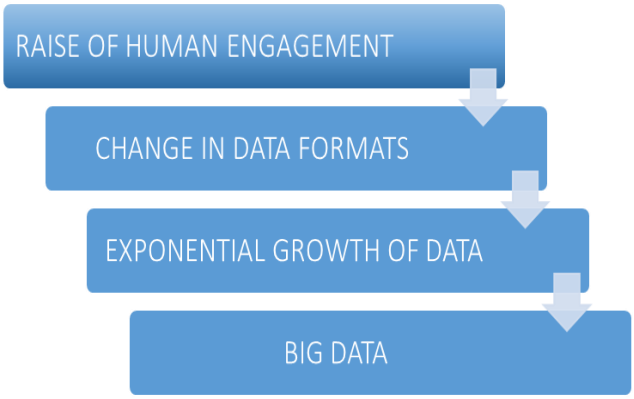

Fig: Evolvement of Big Data.

Due to increase of competitive pressure day by day human engagement was increasing. Let's take E-commerce applications as an example, they started their business with just text formats. To stand in the competitive field they started using unstructured formats like images and vides to advertise their products. Majority of the people in this world are using social networks like Facebook, Twitter, etc., as their communication medium. Initially they were communicating through text formats like chatting. Later advancements came in communication medium too by posting the images, videos as part of their communication. Like this in every technology the use of web was increased and data formats were changed from structured, to semi structured and semi structured to unstructured.

Structured data: This type describes data which is grouped into a relational scheme (e.g., rows and columns within a standard database). The data configuration and consistency allows it to respond to simple queries to arrive at usable information, based on an organization's parameters and operational needs.

Semi-structured data: This is a form of structured data that does not conform to an explicit and fixed schema. The data is inherently self-describing and contains tags or other markers to enforce hierarchies of records and fields within the data. Examples include weblogs and social media feeds.

Unstructured data: This type of data consists of formats which cannot easily be indexed into relational tables for analysis or querying. Examples include images, audio and video files.

The traditional databases like Relational Database were failed to deal with these data formats due to complexities involved with their schema. All these led to increase the data exponentially as shown below [2].

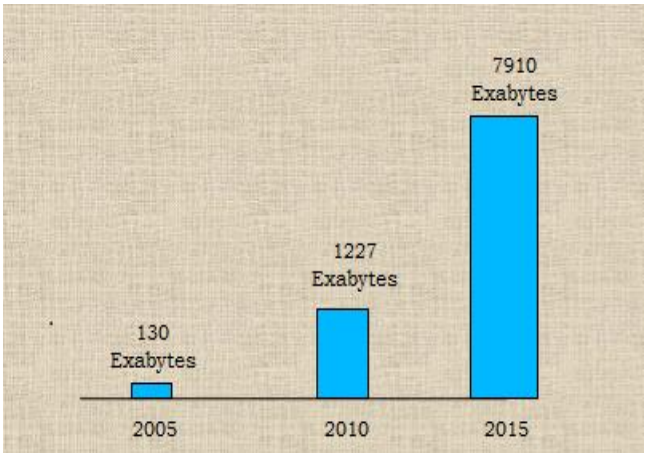

Fig: Exponential form of Data Explosion.

Such huge explosion which lead to large volume termed as BigData. This Big Data evolvement made Relational Databases inefficient to tackle.

A. *REASONS TO FAILURE OF RELATIONAL DATABASES*

- Relational Databases fails to accommodate new changes due to schema centric nature and sometimes leads to redesign the schema.
- Since the Join operation is expensive query performance deteriorates as the dataset gets bigger which is the key concept of this paper.
- Unstructured data can't be handled by Relational Databases.
- Their structure or schema must be known in advance.

In response to these changes, new ways of storing data like NoSql have emerged that allow data to be grouped together more naturally and logically without any restrictions on database schema. One of the most popular NoSql database is Graph Database.

III. NOSQL GRAPH DATABASES

More expressive method for storing data, thereby allowing much more complex questions to be asked of the database, and effectively demining the join bomb. Better and optimal solution for connected data which doesn't have performance variations as the dataset is getting bigger. There is no concept of fixed schema and it is having white board friendly nature, any data can be represented very naturally from the rough sketch. Accommodating new changes is the major asset of Graph Databases. This is one of the solution to the Big Data problem. Some of the use cases of Graph Databases are Bioinformatics, Geo Spatial networks, Recommendations, Authorization and Access control, Social media etc.

A. *PROPERTY GRAPH MODEL*

Data model used by Graph Databases for storing and retrieving data. It indicates that data will be managed through vertices and arcs with the help of labels and properties assigning to both vertices and arcs as shown below. Every arc in the property graph model shows the direction which traverses from one node to the other.

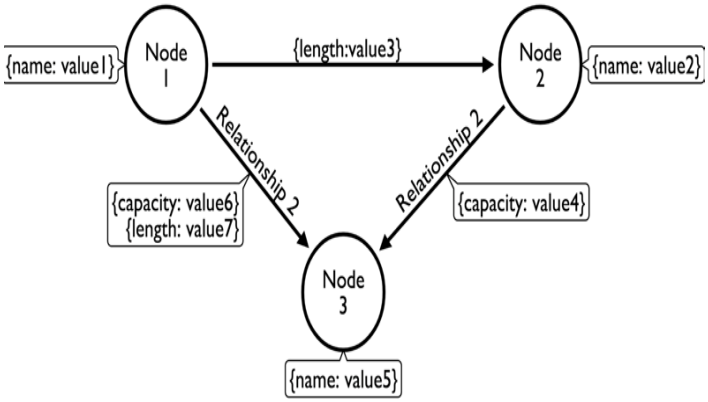

Fig: Property graph model.

B. *USE CASES*

There are so many use cases that were handled by Graph Databases [3]. The natural way of representation is the big asset of Graph Databases and it allows to handle the application areas which involved complexity.

- Social networks:
  Social networks is one of the popular area where Neo4j is the most suitable source. Since all social network applications build on the base of connectivity, Neo4j is the appropriate option for dealing such connected data. One beautiful example is finding the friends of friends in the network.
- Bioinformatics:
  Very powerful area where there is a need of Graph Databases involved like storing the names of proteins and their associated interactions. It also helpful to show the sequence of transformations involved very naturally. Bio4j is the specific Graph Database which is designed to deal the bioinformatics data.
- Recommendations:
  Recommending people by analyzing the purchasing behavior of a specific user. This purchasing behavior will be attained through preferences and attitudes as expressed in ratings and reviews.
- Geo:
  Geospatial applications of graph databases are particularly relevant in the areas of telecommunications, logistics, travel, timetabling, and route planning.

IV. NEO4J

Schema free database especially designed for scalability and high availability in order to deal with OLTP. Neo4j's clustering provides [4],

- Horizontal Scalability: Adding more machines to the cluster and distributing the load.

- Vertical Scalability: Adding more horsepower to the machines.

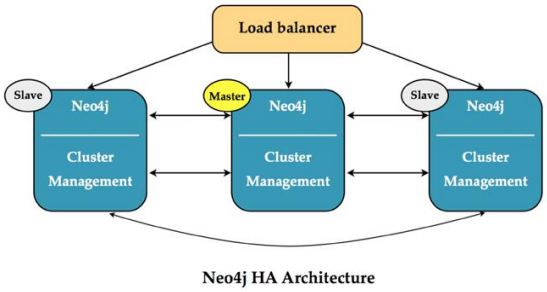

Fig: Neo4j High Availability architecture.

By holding the entire dataset of the database, all servers hold the same data and therefore can respond to all query requests. Comply with a master-slave consistency scheme. This means that, in case of potential conflicting data in the database, the Master server instance will decide what would be right data to keep and persist. If at some point, the cluster would lose its master, the remaining cluster member instances would run a master election algorithm that allows them to quickly choose a new master[5].

A. *CYPHER*

Query language specific to Neo4j Graph Database. Pattern matching nature of the Cypher allows the people even with less technical knowledge to tackle database. It's a Declarative query language which allows which allows greater readability of the queries which is important as people tend to read their database queries. Better and optimal one for path finding queries to find out whether there are useful paths between different nodes on the network.

There is, as of today, no agreed-upon standard for graph query languages, as exists in the relational database management systems (RDBMS) world with SQL[6]. Cypher was chosen in part because of the authors' fluency with the language, but also because it is easy to learn and understand, and is widely used. Part of the reason why cypher is such an important part of Neo4j is that we know that declarative languages, especially in the database management systems world, are critical to mass adoption.

V. IMPLEMENTATION

As an example Movie Database which includes movies, actors, co-actors, directors etc. was taken to make comparison with example query using Neo4j and MySql as shown below:

"Finding the list of directors a person with name alice acted in"

MySql:

```
SELECT dir.name, count(*)
FROM Person Alice
   JOIN Actor on
      Person.person_id = Actor.person_id
   JOIN Director on
      Actor.movie_id=Director.movie_id
  JOIN Person dir on
      Director.person.id = dir.person_id
WHERE alice.name ="Alice"
GROUP BY dir.name.
```

VS

Neo4j:

```
START alice=node:Person(name="Alice")
MATCH alice-[:ACTED_IN]->movie,
        director-[:DIRECTED]-> movie,
RETURN director.name, count(*)
ORDER BY count(*) desc
```

A starting point is a relationship or a node where a pattern is anchored. You can either introduce starting points by legacy index lookups or by id. MySQL query as shown above depends on Join operation which is expensive and eats lot of time as the size of the dataset is getting increased as shown below[7]. Whereas the Cypher query uses the START command for locating the starting node then it uses the traversing technique node by node which matches through MATCH command.

A. *Relational Database model:*

The schema in advance nature of Relational Database as shown below is the main reason why they are unable to accommodate the new changes at intermediate stage.

Actors: actors (actor_id, actor_name, actor_age)
Movies: movies (movie_id, movie_name, release_date, actor_id, director_id)
Direcors: directors (director_id, director_name, director_age)

Actors:

Actor_id	Acor_name	Actor_age

Movies:

movie_id	movie_name	Acor_id	Direcor_id

Directors:

director_id	director_name	director_age

the above three tables are used recursively through JOIN operation which consumes lot of time.

B. *Neo4j database model:*

The representation of sample movie database is shown below, which includes list of actors, movies, and directors. As depicted below actor act as a starting node in the in the process of finding the list of directors of an actor through cypher query. Just it uses the traversing technique from node to node and reaches the destination.

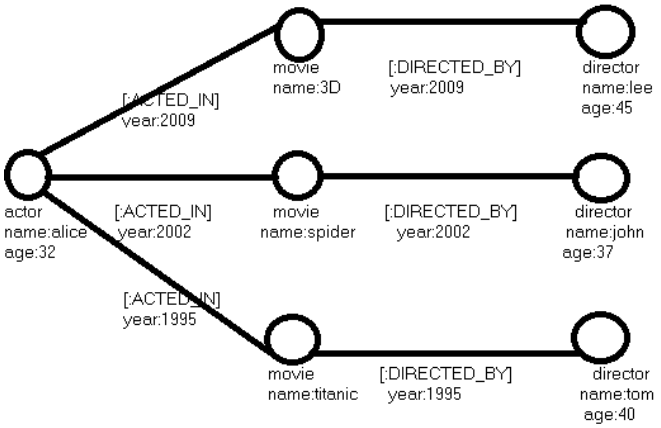

Fig: sample movie databases using Neo4j.

Those edges which are having ACTED_IN relationship from the source which was selected by START command will be filtered for traversing from the query. This is accomplished by the MATCH command in cypher. From the intermediate node movie those nodes which are having DIRECTED_BY relationship will be selected to produce as an output.

C. IMPLEMENTATION RESULTS

The following table illustrates the performance times consumed by Relational Database and Graph Database. As the data set getting increased the time for generating the output was increased exponentially in case of traditional database which was shown using MySql, whereas Neo4j Graph Database is having just linear increment as the number of records getting increased.

DATA SET SIZE	MYSQL	NEO4J
100	27.1	13
200	98.47	17
300	285.13	22

Fig: Performance table in milliseconds.

Therefore advancement is taken place in both ends as the data advanced from structured to unstructured the technology too advanced from traditional to Graph Databases.

VI. CONCLUSIONS

The table itself presented above is suffice to conclude that Graph Database will stand in the top with Relational Database with respect to the performance. Various reasons are presented in the paper to prove that and some of the BigData issues too handled by Graph Databases. To handle the present generation data which was purely unstructured, selection of Graph Databases is an optimal solution.

VII. FUTURE WORK

Entire database field in the future is going to be subset of BigData which is the present trend. Therefore the demand for techniques used for handling BigData too increases. So, BigData solutions like Graph Databases will place a major role in the coming future. In coming days, we are going to delve into the particulars of Neo4j and how it is going to tackle the crucial unstructured data diplomatically by taking an application area of it as a sample.

REFERENCES

[1] http://ovum.com/research/what-is-big-data-the-end-game/
[2] http://www.emc.com/collateral/analyst-reports/idc-extracting-value-from-chaos-ar.pdf.
[3] Mukul sharma, Pradeep soni “quantitative analysis and implementation of relational and graph databases.”
[4] www.it-ebooks.ingo/ Learning Neo4j.
[5] http://ovum.com/research/what-is-big-data-the-end-game/
[6] http://www.tutorialspoint.com/neo4j/neo4j_overview.htm
[7] Ian Robinson, Jim Webber & Emil Eifrem O’reilly Complements of Neo Technology “Graph Databases”