Volume 5, Issue 2, February 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Comparative Analysis of Graph Database Models using Classification and Clustering by using Weka Tool

Pradeep Jadhav, Ruhi Oberoi
Department of CSE, JNEC,
Dr. BAMU, Aurangabad, India

Abstract- The limitation of traditional databases, in particular relational model, to cover the requirements of current applications have leaded the development of new database technologies. The graph databases are calling the attention of the database community because in trendy projects where the database is needed the extraction of worthy information relies the processing of graph-like structure of the data. In this paper we present the systematic comparison of Neo4j and Dex graph database models. This paper includes general features (for data storing and querying), data modeling features (i.e. data structures, query languages and integrity constraints) and the support for graph essential queries with comparison of different graph databases such as Neo4j and Dex graph database models.

Keywords-DEX, Neo4j, NOSQL, Graph database, SQL

I. INTRODUCTION

The graph database queries are user-friendly domain-specific and can be thought of as an "SQL for graphs"[1], [3]. The similarity to SQL is intentional and makes the transition much easier for developers/consultants. When an SQL query on the RDBMS is as long as half a novel, the Cypher Query equivalent is usually much shorter and much more intuitive [3]. The traverser API in an RDBMS is highly resource intensive, since each step to a neighboring node has to be depicted with a JOIN. In contrast, the graph database property hypergraph concept allows direct access to neighboring nodes by eliminating the edge attribute.

Graph databases support a graph model which allows for a direct persistent storing of the particular objects in the database together with the relations between them. In addition, a GDB should provide an access to query methods that not only deal with the stored objects, but also with the graph structure itself. The best known example of such an operation is traversal, which in its most simple form can be used to obtain the neighbors of a specified object, that is, the objects that the specified object is directly related to.

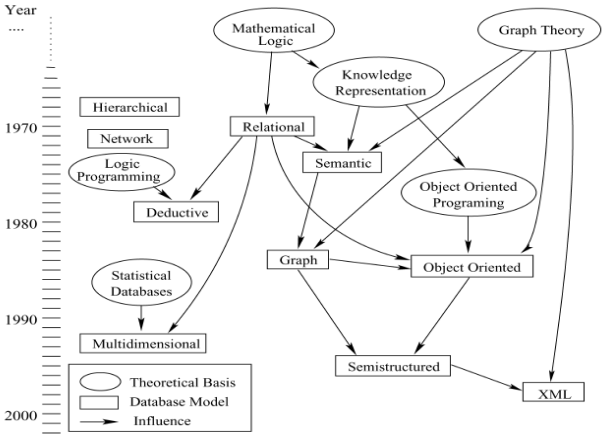

Fig.1. Evaluation of Graph Database Model

Fig.1 [1] shows evaluation of graph database model. Rectangles denote database models, arrows indicate influences, and circles denote theoretical developments. A time-line in years is shown on the left.

A. Objective and Contribution

- Define an application domain for graph database(Facebook Social Application)
- Select graph databases (Neo4j and Dex) and identify the modeling concepts and API of these systems.
- Evaluate existing graph database systems while building benchmark which comprised of graph-based tasks and a variety of graphs.
- An empirical study of the performance of graph databases while dining an representative high level queries mapping to system level.

B. Motivation

Graph database models can be characterized as those where data structures for the schema and instances are modeled as graphs or generalizations of them, and data manipulation is expressed by graph-oriented operations and type constructors.

One of the motivations towards this paper is to provide a benchmarking mechanism to measure the effectiveness of graph traversal operations. It also motivates us to measure the capabilities of graph databases to perform query like traversal where one searches for topologically related vertices for a given vertex. It also searches the graph analysis/mining operations that require the traversal of the whole graph.

C. Applications of Graph Database

Several areas have witnessed the emergence of huge data networks called complex networks. So graph databases are the best database to implement such complex network of relationships having millions of nodes and relationships. The main application areas of graph databases are:

1) Social networks: In social networks, nodes are people or groups, while links show relationships or flows among nodes. Some examples are friendships, business relationships, research networks (collaboration, coauthorship), communication records (mail, telephone calls, email), computer networks, and national security There is growing activity in the area of social network analysis and also in visualization and data processing techniques for these networks.

2) Information networks: Information networks model relations representing information flow, such as citations among academic papers, World Wide Web (hypertext, hypermedia), peer-to-peer networks, relations among word classes in a thesaurus, and preference networks.

D. Advantages

The benefits of using a graph data model are given by: the introduction of a level of abstraction which allows a more natural modeling of graph data; query languages and operators for querying directly the graph structure; and ad-hoc structures and algorithms for storing and querying graphs.

Graph databases are also somewhat similar to object databases in case where objects and relationships between them are all represented as objects with their own respective sets of attributes.

Graph database consists of several advantages:

- It enables very fast queries when the value of the data is the relationships between people/items.
- Use Graph Databases to identify relationships between people/items, even when there are many degrees of separation.
- Where the relationships represent costs, identify the optimal combination of groups of people/items.

II. LITERATURE SURVEY

The limitations of traditional databases, in particular the relational model, to cover the requirements of current application domains, has lead the development of new technologies called NOSQL databases [1]. According to its data model, these databases can be categorized as: Wide-column stores, which follow the BigTable model of Google (e.g., Cassandra); Document stores, which are oriented to store semi-structured data (e.g., MongoDB); Key-value stores, which implement a key to value persistent map for data indexing and retrieval (e.g. BerkeleyDB); and Graph Databases, which are oriented to store graph-like data.

Activity around graph databases flourished in the first half of the nineties and then the topic almost disappeared [2]. Recently the area is gaining attention because in trendy projects where a database is needed, the importance of the information relies on the relations more or equal than on the entities (a basic principle of every graph database). Moreover, the continued emergence and increase of massive and complex graph-like data makes a graph database a crucial requirement. This renascence is showed by the availability of several graph databases systems.

One of the most important elements conforming a database is its database model (or simply data model). In the most general sense a data model is a collection of conceptual tools used to model representations of real-world entities and the relations among these entities. From a database point of view, a data model consists of three components: a set of data structure types, a set of operators or inference rules, and a set of integrity rules.

Graph database models can be defined as those in which data structures for the schema and instances are modeled as graphs or generalizations of them, and data manipulation is expressed by graph-oriented operations and type constructors. These models took off in the eighties and early nineties alongside object oriented models. Their influence gradually died out with the emergence of other database models, in particular geographical, spatial, semi structured, and XML. Recently, the need to manage information with graph-like nature has reestablished the relevance of this area. The main objective of this survey is to present the work that has been conducted in the area of graph database modeling, concentrating on data structures, query languages, and integrity constraints.

Renzo Angles and Claudio Gutierrez [1] introduce Survey of Graph Database Models. They give the information about graph database models with evaluation of graph database. They provide a historical data which provides very broad and depth analysis of literature on the graph data models and query languages graph. The authors compare the proposals of notion of a graph database model of available at the moment. Renzo Angles [2] describes comparison of current graph database model. This paper consists of current graph databases and their support for querying graphs. Domingo De Abreu introducedgraph databases and RDF engines for consuming and mining linked data [6].

With respect to the recent developments in the area. Pere Burton reviewed six graph databases (Neo4j, Hyper-GraphDB, DEX, InfoGrid, Sones and VertexDB) and published a comparison-matrix that included information like software features (e.g., license), schema features (e.g., types of nodes and edges), query features (e.g., language and traversals), general database features (e.g., transactions, indexing), database operation utilities (e.g., protocols), language bindings and operating systems. This work summarizes the features but does not include major discussion nor analysis.

III. PROPOSED SYSTEM

Proposed System consists of research and comparison of two databases such as Neo4j and Dex graph databases. A graph database stores data in a graph, the most generic of data structures, capable of elegantly representing any kind of data in a highly accessible way.

A. Types of Graph Database Models

*1) Neo4j Graph Database:*As a robust, scalable and high-performance database, Neo4j is suitable for full enterprise deployment or a subset of the full server can be used in lightweight projects.

It features:

- true ACID transactions
- high availability
- scales to billions of nodes and relationships
- high speed querying through traversals

Proper ACID behavior is the foundation of data reliability. Neo4j enforces that all operations that modify data occur within a transaction, guaranteeing consistent data. This robustness extends from single instance embedded graphs to multi-server high availability installations.

Neo4j is a commercially supported open-source graph database. It was designed and built from the ground-up to be a reliable database, optimized for graph structures instead of tables. Neo4j is based on the data model of a directed multigraph with edge labels and optional node and edge properties. Node and links can be changed but have identity maintained by DBMS. Labels and property keys are strings, property values can be primitive java data types and strings or arrays of both.

The fundamental units that form a graph are nodes and relationships. In Neo4j, both nodes and relationships can contain properties. Nodes are often used to represent entities, but depending on the domain relationships may be used for that purpose as well.

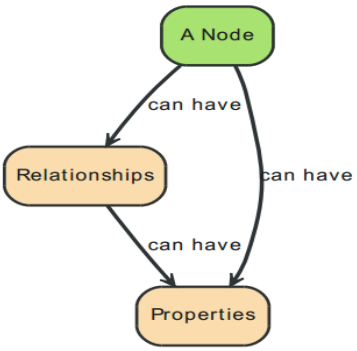

Fig.2. Neo4j Graph Database Nodes and relationships

Various Operations perform by Neo4j Graph Database:

a) Add Neo4j to the build path

- Get the Neo4j libraries from one of these sources:

1. Extract a Neo4j download<http://neo4j.org/download/>zip/tarball, and use the jarfiles found in the lib/directory.
2. Use the jarfiles available from Maven Central Repository<http://search.maven.org/#search|ga|1|g%3A%22org.neo4j%22>

- Add the jar files to your project:

JDK tools
Append to -classpath
Eclipse

1. Right-click on the project and then go Build Path → Configure Build Path. In the dialog, choose Add External JARs, browse to the Neo4j lib/directory and select all of the jar files.
2. Another option is to use User Libraries<http://help.eclipse.org/indigo/index.jsp?topic=/org.eclipse.jdt.doc.user/reference/preferences/java/buildpath/ref-preferences-userlibraries.htm>.

NetBeans

1. Right-click on the Librariesnode of the project, choose Add JAR/Folder, browse to the Neo4j lib/directory and select all of the jar files.

2. You can also handle libraries from the project node, see Managing a Project's Classpath<http://netbeans.org/kb/docs/java/project-setup.html#projects-classpath>.

b) Add Neo4j as a dependency

Syntax:

<project>
...
<dependencies>
<dependency>
<groupId>org.neo4j</groupId>
<artifactId>neo4j</artifactId>
<version>1.9.M04</version>
</dependency>
...
Using Neo4j embedded in Java applications:
</dependencies>
...
</project>

c) Starting and stopping

Syntax to create a new database or ópen an existing one:

graphDb = new GraphDatabaseFactory().newEmbeddedDatabase(DB_PATH);

2) Dex Graph Dtabase

Here, we evaluate DEX, a high performance graph database querying system that allows for the integration of multiple data sources. DEX makes graph querying possible in different flavors, including link analysis, social network analysis, and pattern recognition and keyword search.

DEX [2], [3] provides a Java library for management of persistent and temporary graphs. Its implementation, based on bitmaps and other secondary structures, is oriented to ensure a good performance in the management of very large graphs.

DEX queries are implemented as a combination of low level graph-oriented operations, which are highly optimized to get the maximum from the data structures. DEX aims at maintaining the list of operations as small as possible and to leave the implementation of more complex algorithms to a higher level. Fig.3 shows Dex architecture.

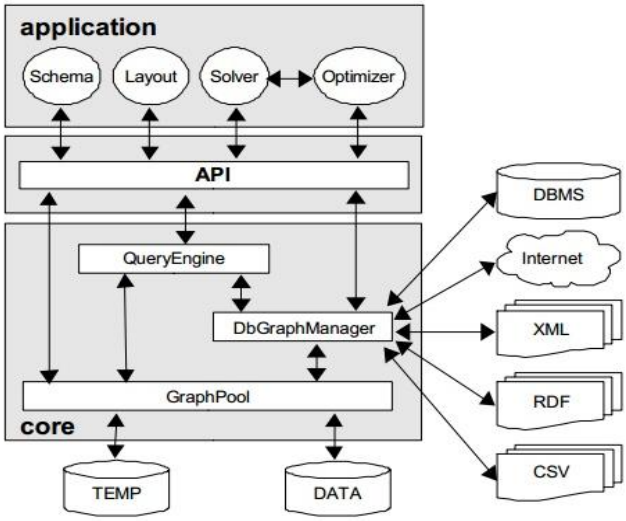

Fig.3. Dex Architecture

B. Graph Databases and Their Support for Querying Graphs

1) *Adjacency Queries:* In this type of queries the primary notion is node/edge adjacency. Two nodes are adjacent when there is edge between them[2].
2) *Reachability Queries:* These queries are characterized by path or traversal problem. The problem causes in reachability test whenever two given nodes are connected to path[2].
3) *Pattern Matching Queries:* Pattern matching queries find all sub-graphs of data graph that are isomorphic to pattern graph [2].
4) *Summarization Queries:* Summarized queries are not related to consult the graph structure [2]. They are based on special functions that allow summarizing or operating on the query results, normally returning a single value.

IV. EXPERIMENTAL EVALUATION

A. Setup: Computer Configurations and Datasets

We used eclipse 3.7.2, running at 2.4GHz, core i3 processor, 3 GB of RAM and 320 GB hard-disk for implementing Neo4j graph database and Dex database. Here we used large synthetic datasets for comparison is synthetic dataset which

is generated by facebook generator. A variety of synthetic graphs are analyzed graph with different size and densities are generated using facebook generator. To examine the impact on the performance of graph systems under different glowing scale and size using facebook graph generator is an important aspect in this paper.

B. Data Loading

Here, we have configured transaction during the creation of corresponding graph representations. We have imported data and increased the scale for graph creation in order to build benchmark in Neo4j and DEX graph for comparing Neo4j and DEX graph databases. We have considered datasets based up on SocNet data model 2.2. The numbers of nodes created are users and numbers of edges generated are posts, likes and comments, respectively.

C. Query Workload and Evaluation

The workload of queries has been assembled by selecting common and well-known graph algorithms as well as used metrics from the domain of SocNet data model 2.2. We perform a selection of domain specific queries for the graph database benchmarks. Our approach is based on the user interaction with SocNet data model that are mapped to the queries of the benchmark. Such interaction includes data analysis to identify friend-of-friend requests based on mutual likes, posts and comments of users.

Our graph databases benchmark implementation is used to evaluate the individual performance of atomic operations (such as joins and aggregations), rather than more complex queries. When considering graphs, we need several micro-queries which may be atomic and we group them into selection, adjacency and pattern matching.

The comparison of Neo4j graph database and Dex database model shown in table 1 where it shows time require executing query for one user and measuring results for graph creation is shown fig.4. Same as a result of one user in table 2 and table 3 shown comparison of database models for 10 and 100 users respectively. Figure 5 and figure 6 shows comparison of database models for 10 and 100 users respectively.

Table I. Comparison of Neo4j and Dex database for one user

For 1 user		
No of Nodes	*N*	**DEX**
8	*686*	*155*
16	*680*	*121*
32	*682*	*126*
64	*696*	*232*
128	*669*	*276*
256	*715*	*286*
512	*723*	*325*
1.024	*789*	*354*
2.048	*858*	*366*
4.096	*991*	*389*
8.192	*1146*	*403*
16.384	*1329*	*421*
32.768	*1395*	*443*
65.536	*1509*	*459*
131.072	*1686*	***648***

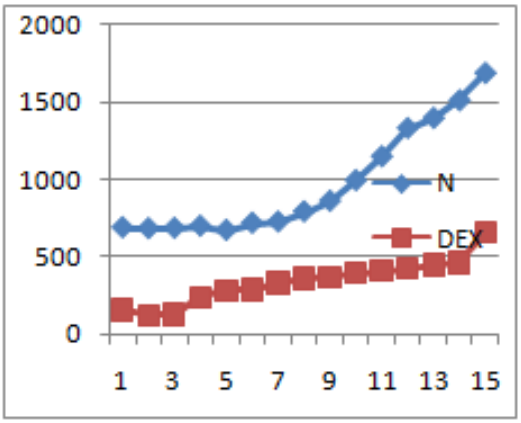

Fig. 4. Comparison of Neo4j and Dex database for one user

Table II Comparison of Neo4j and Dex database for ten users

For 10 User		
No of Nodes	N	DEX
8	*672*	*16*
16	*696*	*13*
32	*635*	*13*
64	*687*	*24*
128	*691*	*28*
256	*717*	*34*
512	*702*	*36*
1.024	*768*	*37*
2.048	*945*	*39*
4.096	*984*	*41*
8.192	*1153*	*43*
16.384	*1248*	*46*
32.768	*1383*	*50*
65.536	*1506*	*59*
131.072	*1782*	***67***

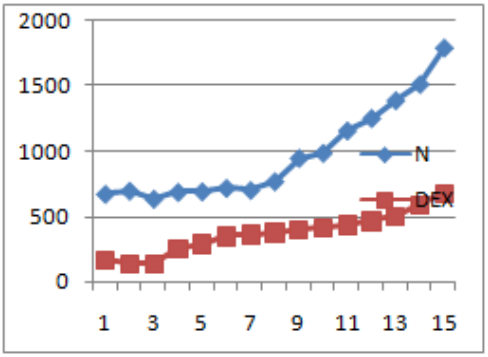

Fig. 5. Comparison of Neo4j and Dex database for ten users

Table III. Comparison of Neo4j and Dex database for hundred users

For 100 User		
No of Nodes	N	DEX
8	675	167
16	696	155
32	702	154
64	705	261
128	728	294
256	759	356
512	795	375
1.024	888	382
2.048	1068	415
4.096	1152	429
8.192	1158	454
16.384	1357	478
32.768	1383	526
65.536	1557	633
131.072	1731	**781**

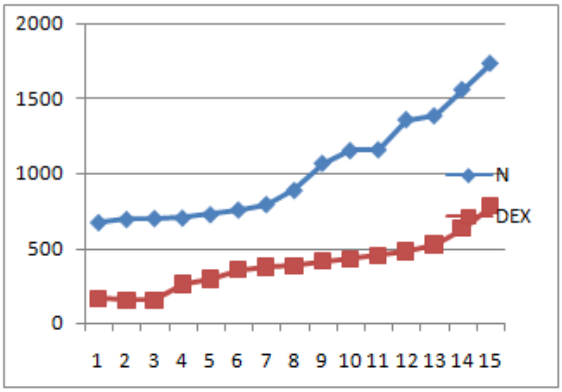

Fig. 6. Comparison of Neo4j and Dex database for 100 users

D. Comparing Graph Database Models using classification

1) Using Weka Tool result of Classification :

When we classify Neo4j and Dex database using Decision tree classifier for 4 instances of table then it gives result as below:

Decision Table:

Number of training instances: 4
Number of Rules: 4
Non matches covered by Majority class.
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 1
For Neo4j database model Merit of best subset found: 21.448
For Dex database model Merit of best subset found: 23.195.

As above result we can say that merit of best subset found is less when we use Neo4j database model.

When we classify Neo4j and Dex database using ZeroR classifier then it gives result as:

Instances: 4
Attributes: 3
Nodes
N
Dex
Test mode:user supplied test set:
=== Classifier model (full training set) ===
ZeroR predicts class value: 30.0
Time taken to build model: 0 seconds
=== Evaluation on test set ===
=== Summary ===
Correlation coefficient 0
Mean absolute error 18
Root mean squared error 21.4476
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 4

2)Using Weka Tool result of Clustering:

When we classify Neo4j and Dex database using simple k-mean clustering on 4 instances:

Instances: 4
Attributes: 3
Nodes
N
Dex
Test mode:evaluate on training data
=== Model and evaluation on training set ===
kMeans
======
Number of iterations: 3
Within cluster sum of squared errors: 0.22285811390541116
Missing values globally replaced with mean/mode

Cluster centroids:

Attribute	*Full Data (4)*	*Cluster# 0 (3)*	*1 (1)*
Nodes	*30*	*18.6667*	*64*
N	*686*	*682.6667*	*696*
Dex	*158.5*	*134*	*232*

Time taken to build model (full training data) : 0.02 seconds

=== Model and evaluation on training set ===

Clustered Instances

0 3 (75%)

1 1 (25%)

V. CONCLUSIONS

In paper consists of comparison of two different graph databases such as Neo4j and Dex graph database models. This shows that some aspect of different graph database models which deserve more development.

According to the result we can conclude that, Dex graph database model required more time as compare to the Neo4j graph database model. Therefore Neo4j graph database is more suitable to use than Dex graph database model.

ACKNOWLEDGMENT

We are thankful to the Department of Computer Science and Information Technology for providing the PG Lab at JNEC, Aurangabad (M.S.) India.

REFERENCES

[1] Renzo Angles And Claudio Gutierrez." Survey of Graph Database Models"ACM Computing Surveys, Vol. 40, No. 1, Article 1, Publication date: February 2008.

[2] Renzo Angles. "Comparison of Current Graph Database Models", Department of Computer Science, Talca..

[3] "NOSQL Databases", http://nosql-database .org/

[4] "DEX," http://www.sparsity-technologies.com/dex.

[5] "Neo4j," http://neo4j.org/.

[6] Jena- RD F.Jena documentation.Internet: http://jena.sourceforge.net/documentation.html , 2010.

[7] Sesame. Open RDF website, Internet: http://www.openrdf.org, 2010.

[8] Neo4j B log, Internet: http://bl og. neo4j.org/2009/04/current-database-debate-andgraph.html ,201 0.

[9] Neo4jmanual, Internet: http://docs. neo4j.org/chunked/stabl e/graphdb- neo4jnodes. html ,2010

[10] J. Paredaens and B. Kuijpers, "Data Models and Query Languages for Spatial Databases," Data & Knowledge Engineering (DKE), vol. 25, no. 1-2, pp. 29–53, 1998.

[11] P. Urb ´ on, "Nosql graph database matrix," http://nosql.mypopescu.com/post/619181345 /nosql-graph-databasematrix, May 2010.

[12] "Short overview on the emerging world of graph databases,"http://www.graph-database.org/overview.html.

[13] D. Dominguez-Sal, P. Urb ´ on-Bayes, A. Gim´ enez-Va ˜ n ´ o, S. G ´ omezVillamor, N. Mart´ ınez-Baz ´ an, and J. L. Larriba-Pey, "Survey of graph database performance on the hpc scalable graph analysis benchmark," in Proc. of the 2010 international conference on Web-age information management (WAIM). Springer-Verlag, 2010, pp. 37–48.

