IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Volume-3, Issue-7 E-ISSN: 2347-2693

Web based ETL Approach to Transform Relational Database to Graph Database

Sonali D. Chaure[1*], M. U. Kulkarni[2] and Pankaj M. Jadhav[3]

[1,2,3]*Computer Engineering and Information Technology, VJTI, Mumbai, India*

www.ijcseonline.org

Received: Jun/03/2015 Revised: Jul/10/2015 Accepted: July/24/2015 Published: July/30/ 2015

***Abstract*—** Data size is growing exponentially, it is coming in more and more connected form. Graph Database Management System (GDBMS) provides efficient solution to data storage in current scenarios. Nowadays, many companies rely on cloud services where you pay as per your need basis and most of the cloud platforms supports non-relational database to avoid scalability issues. Graph Databases have applications in many domains such as social network, organization management, banking, insurance, fraud detection, etc. Therefore there is need to migrate data from relational to non-relational database. Also many companies shifting from traditional relational database to NoSQL databases to avoid scalability issues. In this paper a web based ETL approach has been suggested to convert a Relational Database to Graph Database. Experimental results have been presented to show feasibility of the proposed methodology. Also query execution comparison is done on source and target databases.

***Keywords*—**GDBMS, NoSQL, TRDB

I. Introduction

ETL is most important component of Data warehousing. Using ETL we can extract, transform and load data [1]. Relational databases are in existence from since 1970s [2]. Because of prolonged existence of relational databases, they are matured well but they do not support scalability and also not well designed to handle sparse, large data [3]. Nowadays data is coming in more and more connected form, so while increasing data size there is a need for scalability of database [4]. Most of the popular applications requires huge database to store and retrieve data. We can achieve scalability in two ways- vertical and horizontal scalability. Vertical scalability means to scale up and it is achieved by increasing resources to a single machine. Horizontal scalability means to scale out and it is achieved by adding commodity servers to the existing node. Therefore vertical scalability is expensive as compared to horizontal scalability. With increasing data size vertical scalability is not an efficient option. Relational database does not support horizontal scalability and recently evolved non-relational database i.e. NoSQL databases supports horizontal scalability and it is achieved by increasing commodity servers or by increasing cloud instances [5]. There are several advantages of NoSQL databases over relational databases such as NoSQL databases are schema-less, easy for fast traversal and highly scalable etc. NoSQL databases are classified into four classes - Key Value pair, Document, Graph and Column-oriented, each class has its own features to suit their data, thus different storage and retrieval requirements as per class. NoSQL means ”Not only SQL” to emphasize that they may also support SQL-like query languages [6]. Graph Databases are use for fast traversal of nodes that are connected to each other by number of relationships [7]. There are many social networking websites like twitter, facebook produces huge amount of data and nowadays this data is coming in very connected form. To traverse such a highly scalable and connected data, graph databases are used. In every graph model every node is connected to adjacent node by relationship, this helps to get faster querying results on huge amount data. big retailers like eBay and Wal-Mart are using Neo4j for their critical business projects [8]. Also it helps to eliminate the need of any index lookup for searching or table joins as in RDBMS. There are many graph databases exists like Neo4j [9], OrientDB [10] etc. Nowadays, many companies rely on cloud services where you pay as per your need basis and most of the cloud platforms supports non-relational database to avoid scalability issues and many companies shifting from traditional relational database to NoSQL databases to avoid scalability issues. Therefore there is need to migrate data from relational database to non-relational database.

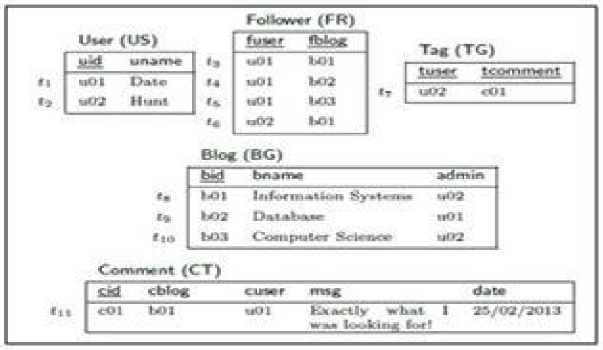
The ultimate objective of this paper is to present a Web based ETL approach to transform Relational Database to Graph Database. This takes input as a Relational Database R (MySql) and converts it into Graph database G (Neo4j). Also comparative analysis of migrated Graph Database creation time, SQL and CQL results is done. MySql (open source Relational Database) and Neo4j (open source Graph Database) is used for evaluating the outcomes of the research. Finally visualization of migrated Graph Database is also shown.

Related work of our approach is presented in section 2, system design and implementation of proposed system is presented in section 3, experimental results and visualization of migrated Graph Database are shown in section 4 and concludes the paper by analysing experimentation results.

II. Related Work

Data analysis is playing a significant role in science and engineering. Data warehouses provide online analytical processing tools for the interactive analysis of multidimensional data of various granularities. ETL is one of most important components of data warehousing. ETL tool can extract data from multiple heterogeneous sources, combine them together, apply various transformations and load the transformed data into target database [11].

Relational Database exists from 1970s, organize data into tables i.e. it stores data in 2-dimensional tables. A table is a two-dimensional structure made up of rows (tuple, records) and columns (attributes, fields). SQL is use to retrieve data from relational database. It does not support scalability, also not good to handle huge and more connected database as it takes too many join operations [12]. Example: RDBMS (open source) MySql. In relational database relational schema is denoted as Ri(Ai), where Ri is ith relation and Ai is set of attributes of ith relation. The set of such relational schema R1(A1)..Rn(An) is called as Relational Database schema. The Relational Database R is set of Tuples over all the Relational schema R1(A1)..Rn(An). Figure 1 shows the example of Relational Database R. Figure 2 shows the schema details of Relational Database R. Attributes that belongs to primary key of that relation are underlined.



User (US)

	uid	uname
t_1	u01	Date
t_2	u02	Hunt

Follower (FR)

	fuser	fblog
t_3	u01	b01
t_4	u01	b02
t_5	u01	b03
t_6	u02	b01

Tag (TG)

	tuser	tcomment
t_7	u02	c01

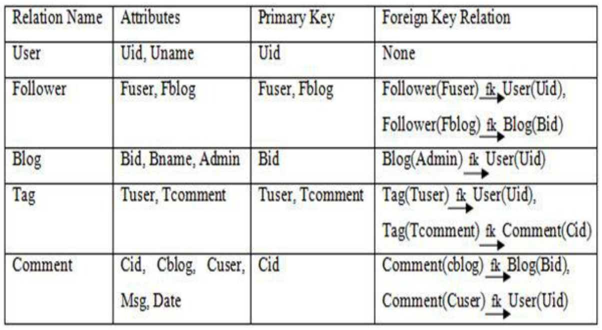
Blog (BG)

	bid	bname	admin
t_8	b01	Information Systems	u02
t_9	b02	Database	u01
t_{10}	b03	Computer Science	u02

Comment (CT)

	cid	cblog	cuser	msg	date
t_{11}	c01	b01	u01	Exactly what I was looking for!	25/02/2013

Fig. 1. Example of Relational Database R



Relation Name	Attributes	Primary Key	Foreign Key Relation
User	Uid, Uname	Uid	None
Follower	Fuser, Fblog	Fuser, Fblog	Follower(Fuser) $\xrightarrow{fk}$ User(Uid), Follower(Fblog) $\xrightarrow{fk}$ Blog(Bid)
Blog	Bid, Bname, Admin	Bid	Blog(Admin) $\xrightarrow{fk}$ User(Uid)
Tag	Tuser, Tcomment	Tuser, Tcomment	Tag(Tuser) $\xrightarrow{fk}$ User(Uid), Tag(Tcomment) $\xrightarrow{fk}$ Comment(Cid)
Comment	Cid, Cblog, Cuser, Msg, Date	Cid	Comment(cblog) $\xrightarrow{fk}$ Blog(Bid), Comment(Cuser) $\xrightarrow{fk}$ User(Uid)

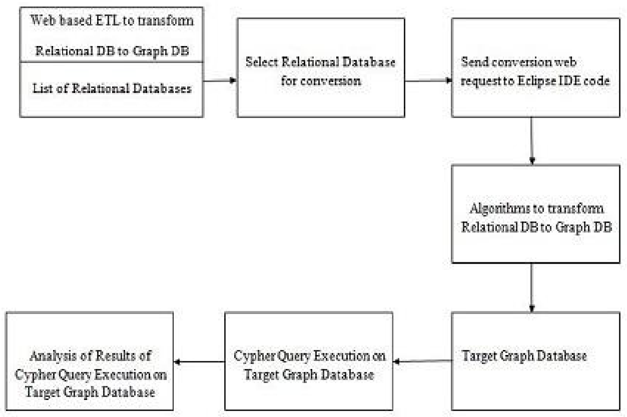
Fig. 2. Schema Details of Example Relational Database R

A NoSQL database provides a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in relational databases. Motivations to use NoSQL Database are simplicity of design, horizontal scaling, and finer control over availability. NoSQL databases are increasingly used in big data and real-time web applications. NoSQL means "Not only SQL" to emphasize that they may also support SQL-like query languages [13]. NoSQL databases are classified into four classes - Key Value pair, Document, Graph and Column-oriented, each class have its own features to suit their data, thus different storage and retrieval requirements as per class. Graph Databases has been invented for fast traversal of millions of nodes that are interconnected via number of relations [14]. Social network websites like facebook and twitter produce several terabytes of data within a month. They contain highly interconnected data. To traverse such data Graph Databases are used. Graph Databases provide flexible model to distribute data over many servers. This property provides high scalability for growing data. In graph model every node has its related node directly connected to it. This eliminates the need of any index lookup or table joins as in RDBMS. Graph model based data stores are backbone of social networking websites. Graph databases allow you to tell a story. They allow you to connect the dots. When you use this powerful type of database, true meaning is one query away. Graph databases are based on graph theory. Graph databases employ nodes, properties, and edges. Neo4j is an open source NoSQL Graph Database.

There are different approaches are present to convert relational database into Graph Database, it uses schema paths and source constraints [15]. Initially before conversion lots of paper work involves in their approach, therefore their approach is not fully automatic [16]. In this paper, presented a web based ETL approach to transform Relational Database to Graph Database. Our approach will take input as a relational database R (MySql) and finally it converts into graph database G (Neo4j).

III. Proposed Work

A. System Architecture


Web based ETL to transform Relational DB to Graph DB
List of Relational Databases
Select Relational Database for conversion
Send conversion web request to Eclipse IDE code
Algorithms to transform Relational DB to Graph DB
Target Graph Database
Cypher Query Execution on Target Graph Database
Analysis of Results of Cypher Query Execution on Target Graph Database

Block diagram of our system web based ETL to transform Relational Database to Graph Database consists of seven modules as below:

1. Web Page: Web based ETL to transform Relational DB to Graph DB
This module consists of web page of our system containing List of Relational Databases (MySql database from our wamp server).
2. Selection of Relational Database for conversion
In this module user can select one Relational Database from List for conversion.
3. Send conversion web request to Eclipse IDE code
In this module user can send web request for conversion to Eclipse IDE.
4. Algorithms to transform Relational DB to Graph DB
In this module different algorithms used to convert Relational Database to Graph Database. By using these algorithms, I have migrated foreign key relations from Relational Database to Graph Database without any data loss.
5. Target Graph Database
In this module Target (Migrated) Graph Database will be created with nodes and its respective relations.
6. Cypher Query Execution on Target Graph Database
In this module, Cypher query execution will be done on Target Graph Database created in pervious module.
7. Analysis of Results of Cypher Query Execution on Target Graph Database
In this module, Analysis of results of Cypher query execution will be done. From this analysis user can conclude that time complexity of migrated Graph Database creation time depends on factors like number of nodes and number of relations. Finally comparative analysis of SQL (on Source Database) and CQL (on Migrated Database) results will be performed based on average time required to retrieve all data from database.
Finally visualization of migrated Graph Database is shown.

B. Algorithms to convert Relational Database to Graph Database

In this section, initially created all nodes for each tuple belongs to all tables. For creating relations between nodes proposed below algorithms and finally target Graph Database is shown in figure 8.

Consider example given in Figure 1,

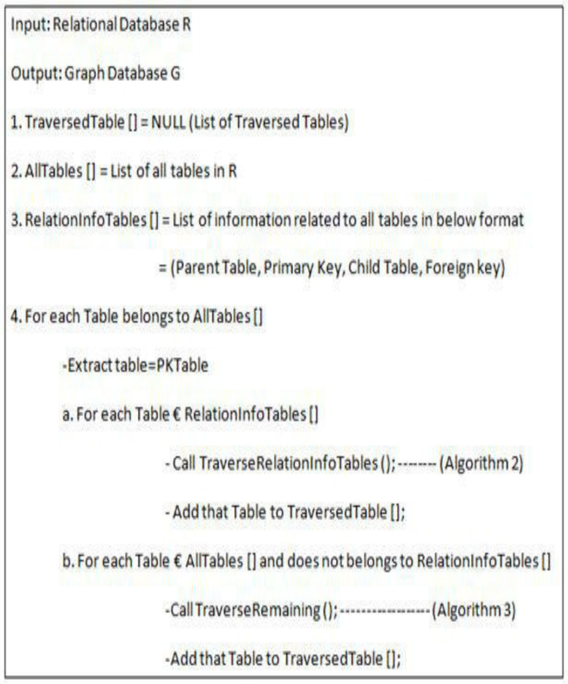
TraversedTables[] = Initially list is empty. After traversal of table, table name will be added to list.

AllTables[] = User, Follower, Blog, Tag, Comment

RelationInfoTables[] = List of information related to all tables in below format (Parent Table, Primary Key, Child Table, Foreign key). Traverse each table belongs to RelationInfoTables[] and after traversing add it to TraversedTable[] Different methods called from algorithm 1 are explained as follows:

1. TraverseRelationInfoTables() = This method is called for each table that belongs to RelationInfoTables[] list.

2. TraverseRemaining() = This method is called for each table that belongs to AllTables[] and does not belongs to RelationInfoTables[].



```
Input: Relational Database R
Output: Graph Database G
1. TraversedTable [] = NULL (List of Traversed Tables)
2. AllTables [] = List of all tables in R
3. RelationInfoTables [] = List of information related to all tables in below format
                    = (Parent Table, Primary Key, Child Table, Foreign key)
4. For each Table belongs to AllTables []
      -Extract table=PKTable
      a. For each Table € RelationInfoTables []
            - Call TraverseRelationInfoTables (); ------- (Algorithm 2)
            - Add that Table to TraversedTable [];
      b. For each Table € AllTables [] and does not belongs to RelationInfoTables []
            -Call TraverseRemaining (); ---------------- (Algorithm 3)
            -Add that Table to TraversedTable [];
```

Fig. 4. Algorithm to Convert Relational Database to Graph Database

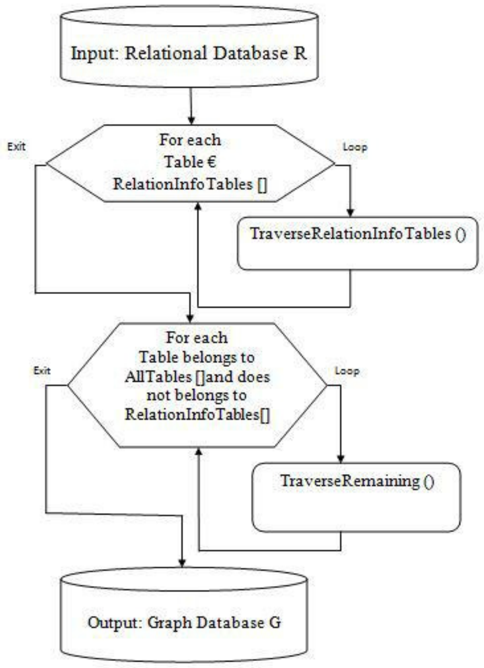

Input: Relational Database R
Exit
For each Table € RelationInfoTables []
Loop
TraverseRelationInfoTables ()
Exit
For each Table belongs to AllTables []and does not belongs to RelationInfoTables[]
Loop
TraverseRemaining ()
Output: Graph Database G

Fig. 5. Flow Chart

Figure 5 shows the flowchart for conversion of Relational

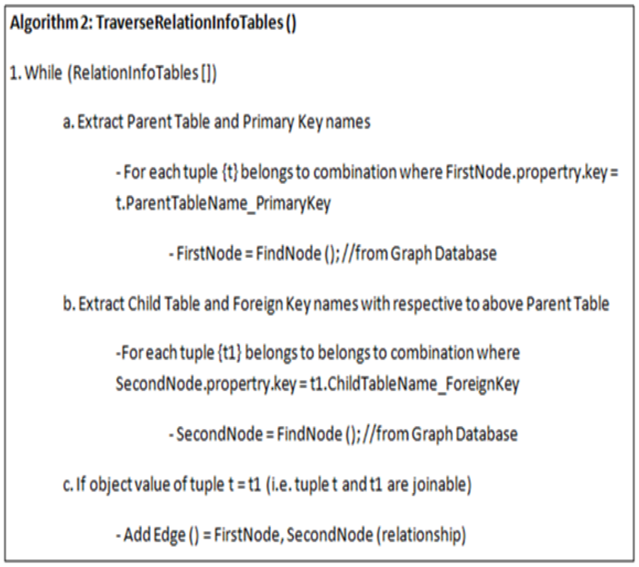
Database R to Graph Database G. It shows the flow of Algorithm defined above.

Relational Database R is given as input.

There are three for loops. Each loop calls a specific module until the condition holds. Both modules are called in a specific order.

After processing through all the modules the output generated is the Graph Database G.

Different modules are called from the main module. Functionality of all algorithm modules is explained below.

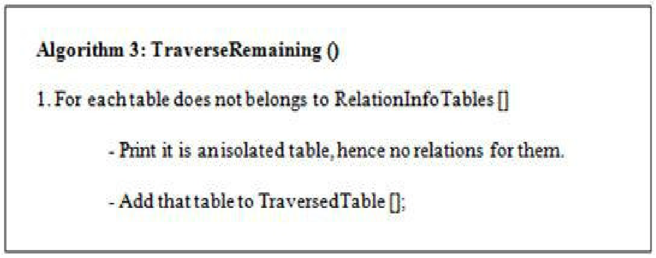


```
Algorithm 2: TraverseRelationInfoTables ()
1. While (RelationInfoTables [])
    a. Extract Parent Table and Primary Key names
        - For each tuple {t} belongs to combination where FirstNode.propertry.key =
        t.ParentTableName_PrimaryKey
            - FirstNode = FindNode (); //from Graph Database
    b. Extract Child Table and Foreign Key names with respective to above Parent Table
        -For each tuple {t1} belongs to belongs to combination where
        SecondNode.property.key = t1.ChildTableName_ForeignKey
            - SecondNode = FindNode (); //from Graph Database
    c. If object value of tuple t = t1 (i.e. tuple t and t1 are joinable)
        - Add Edge () = FirstNode, SecondNode (relationship)
```

Fig. 6. Algorithm to TraverseRelationInfoTables()

TraverseRelationInfoTables () method is called for each table that belongs to RelationInfoTables[] list, Consider example given in Figure 1. Initially consider Parent Table= User and Primary Key= Uid. Find out node from graph database which has combination like Node.propertry.key = t.ParentTable-PrimaryKey from this will get first node (User-Uid), then find second reference node from graph database like combination Node.propertry.key = t.ChildTable-ForeignKey (i.e. Blog-Admin) from this will get second joinable node with first node and then check whether there object values are equal, if equal then add edge between them because they are joinable Tuples. So finally add edge between first node (User-Uid=u01, User-Uname=Date) and second node(Blog-Bid=b02, Blog-Bname=Database, Blog-Admin=u01). Repeat this procedure for all Tuples of table belongs to TraverseRelationInfoTables[].

TraverseRemaining () method is called for each table that belongs to AllTables[] and does not belongs to RelationInfoTables[], All this tables does not have relation with other tables, So their nodes are isolated, So after visiting add that table to TraversedTables[].



```
Algorithm 3: TraverseRemaining ()
1. For each table does not belongs to RelationInfoTables []
        - Print it is an isolated table, hence no relations for them.
        - Add that table to TraversedTable [];
```

Fig. 7. Algorithm to TraverseRemaining()

Figure 8 shows the Graph Database G which is obtained from the Relational Database R shown in Figure 1 after the execution above proposed algorithms to convert Relational to Graph Database.

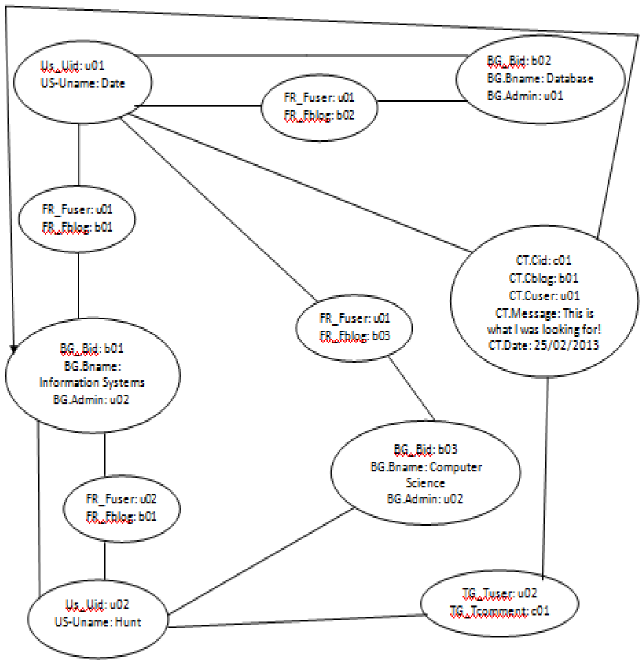
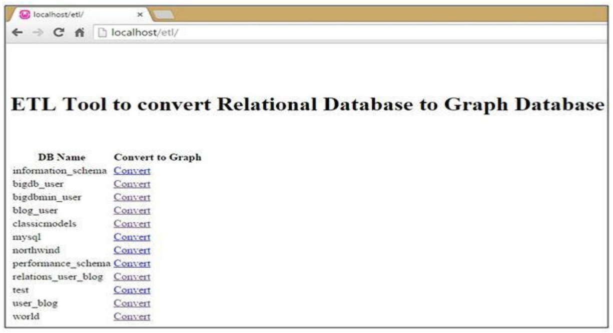


Fig. 8. Graph Database G

IV. EXPERIMENTAL RESULTS



ETL Tool to convert Relational Database to Graph Database

DB Name	Convert to Graph
information_schema	Convert
bigdb_user	Convert
bigdbmin_user	Convert
blog_user	Convert
classicmodels	Convert
mysql	Convert
northwind	Convert
performance_schema	Convert
relations_user_blog	Convert
test	Convert
user_blog	Convert
world	Convert

Fig. 9. Web page for ETL Tool to convert Relational Database to Graph Database

In this project, created one web page which is having list of Relational Database (MySql) from wampserver as shown in figure 9. From this list user needs to select one Relational Database for Graph Database conversion and click on Convert, then conversion web request will be send to Eclipse IDE for selected Relational Database and output will be displayed on Eclipse IDE output console.

This Web based approach has been tested on seven Relational Database (MySql) as shown in figure 10.

Sr. No.	Database (Database Name)	Graph Database creation time (ms)	Number of rows	SQL time to retrieve all rows from Relational Database (ms)	CQL time to retrieve all nodes from Graph Database (ms)	Number of relations	Time to retrieve all relations from Graph Database (ms)
1	Bigdb_user	4573	902	229	31	438	203
2	Bigdbmin_user	6104	902	166	31	895	369
3	Blog_user	2013	307	107	16	148	140
4	Classicmodels	9179	3865	1733	81	3728	875
5	Relations_user_blog	1228	75	71	0	22	47
6	User_blog	1415	75	149	0	68	78
7	World	2108	5303	1389	156	0	0

Fig. 10. Comparison of Graph Database creation time based on its total number of nodes and total number of relations

per figure 10 seven Relational Databases are taken for testing. Graph Database creation time is the time taken to migrate Relational Database to Graph Database. Bigdb-user database takes 4573 ms for migration, which has 902 nodes, 438 relations. Bigdbmin-user database takes 6104 ms for migration, which has 902 nodes, 895 relations, both Bigdb-user and Bigdbmin-user databases has 902 rows still Bigdbmin-user takes more time for migration than Bigdbmin-user because it has more number of relations than total number of relations present in Bigdb-user. Blog-user database takes 2013 ms for migration, which has 307 nodes, 148 relations. Classicmodels database takes 9179 ms for migration, which has 3865 nodes and 3728 relations. Relations-user-blog database takes 1228 ms for migration, which has 75 nodes and 22 relations. User-blog database takes 1415 ms for migration, which has 75 modes and 68 relations. Relations-user-blog and User-blog has same number of nodes still User-blog database takes more time for migration than Relations-user-blog because it has more number of relations than total number of relations present in Relations-user-blog. World database takes 2108 ms for migration, which has 5303 nodes and 0 relations. World database has more number of nodes than Classicmodels, still Classicmodels takes more time for migration than World database because it has comparatively too many relations than World database.

Representations of above results are in the form of bar chart as shown in figure 11. As per bar chart, Classicmodels database takes more time than other databases since it has 3865 nodes and 3728 relations. Classicmodels database has less number of nodes than World database but it has comparatively too many numbers of relations than World database.

Therefore Classicmodels database takes more time for migration than other databases. Relations-user-blog and User-blog has same number of nodes still User-blog takes more time for migration than Relations-user-blog because it has more number of relations than Relations-user-blog.

From above results, it is observed that time complexity of Graph Database creation is not only depends on number of nodes but also depends on number of relations.

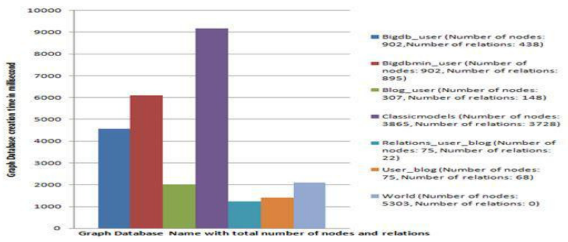


Fig. 11. Bar Chart for Time Comparison of Graph Database

To compare SQL and CQL results based on their execution time referred figure 7 observations. For each database calculated both SQL and CQL time (in millisecond) to retrieve all data from database. For comparison purpose, taken average of all SQL time required to retrieve all data from all Relational Databases and also taken average of all CQL time required to retrieve all data from migrated all Graph Databases.

Comparison of SQL and CQL execution time based on average time taken to retrieve all data from databases is shown in figure 12.

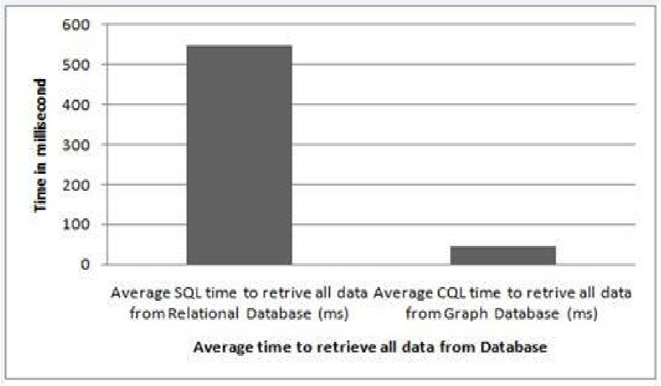


Fig. 12. Comparison of SQL and CQL based on average time to retrieve all data from database

Figure 12, shows that Cypher queries are much faster as compared to the SQL queries because Joins in SQL queries are slower than the pattern matching in the Cypher queries.

Migrated Graph Database Relations-user-blog has 75 nodes and 22 relations selected for visualization.

Neo4j community version 2.2.3 is downloaded for visualization purpose. Visualization for displaying all nodes and relations from Relations-user-blog is shown in figure 13.

After getting results user can export output in different format like PNG, JSON and CSV etc.

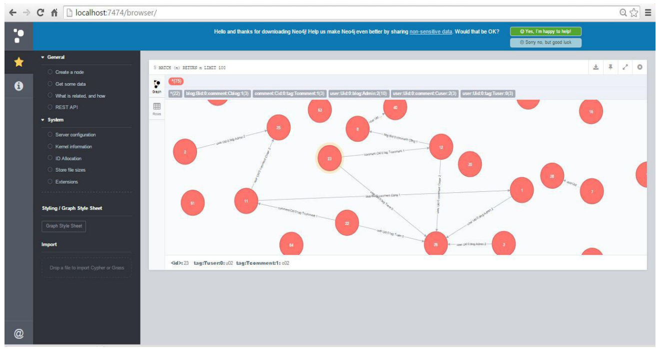


Fig. 13. Visualization of all nodes and relations from Relations-user-blog

Visualization for displaying all nodes in the form of rows from Relations-user-blog is shown in figure 14. In this all nodes from Relations-user-blog are represented in the form of rows.

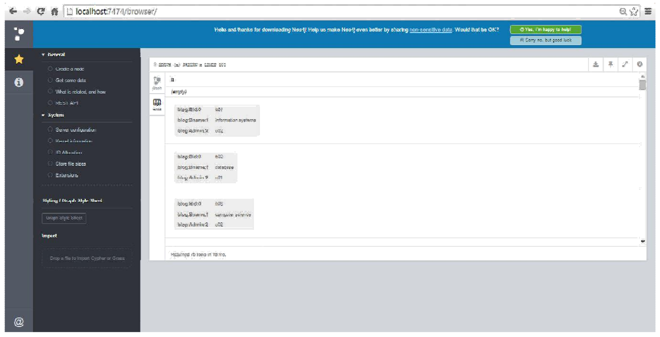


Fig. 14. Visualization of all nodes in the form of rows from Relations-user-blog

Visualization of all relations from Relations-user-blog is shown in figure 15.

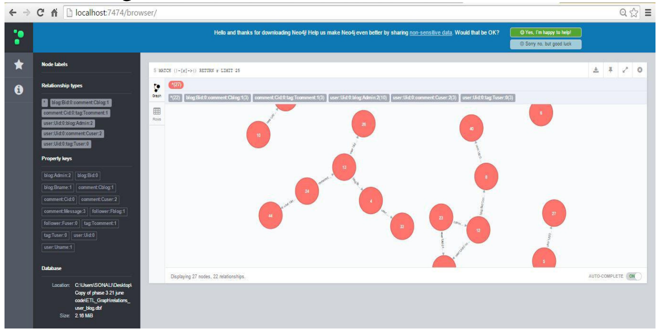


Fig. 15. Visualization of all relations from Relations-user-blog

Visualization for displaying all relations of particular type from Relations-user-blog is shown in figure 16.

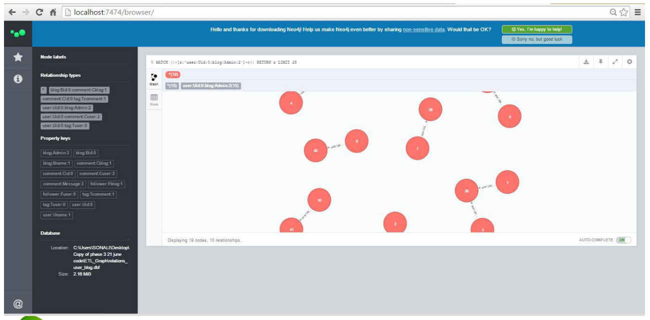


Fig. 16. Visualization of all relations of particular type from Relations-user-blog

V. Conclusion and Future Work

In this paper, presented a web based ETL approach to convert relational database to graph database. This approach uses integrity constraints defined over source and Tuples of Relational Database to transform into target Graph Database without any data loss. Also no prior paper work needed to this web based approach. While querying joins are used in relational database are converted to a traversal path in graph database. From analysis of implementation results, it is observed that time complexity of migrated Graph Database creation is depends on both parameters i.e. Number of nodes and Number of relations. Also from comparative analysis of SQL and CQL on the basis of their execution time to retrieve all data from database, Cypher queries are much faster as compared to the SQL queries because Joins in SQL queries are slower than the pattern matching in the Cypher queries. Finally visualization of migrated Graph Database is shown in Neo4j community.

This web based Relational Database to Graph Database migration can be further explored for transforming data from Relational Database to various other NoSQL databases and for these migrations new methodologies can be use.

References

[1] Kimball R, Caserta J, "The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data", Wiley.

[2] E.F. Codd,"A Relational Model of Data for Large Shared Data Banks, Communications of the ACM.

[3] Relational-databases-sandbox-handout (doc.gold.ac.uk).

[4] K. Kaur and R. Rani, "Modeling and Querying Data in NoSQL Databases",IEEE International Conference on Big Data, pp. 7, 6-9, Oct. 2013.

[5] C. Strauch, "NoSQL Databases", ACM.

[6] https://en.wikipedia.org/wiki/NoSQL.

[7] http://highscalability.com/neo4j-graph-database-kicks-Buttox.

[8] Neotechnology. (http://www.neotechnology.com/ebay-walmart-adopt-neo4j-graph-transforming retail/)

[9] Rob McColl, David Ediger, "A Brief Study of Open Source Graph Databases".

[10] A Brief Study of Open Source Graph Databases.

[11] Mohammed Shafeeq Ahmed, "Data Warehousing Applications: An Analytical Tool for Decision Support System", International Journal of Computer Science and Informatics.

[12] Relational-databases-sandbox-handout.

[13] https://en.wikipedia.org/wiki/NoSQL.

[14] Database (http://neo4j.com/developer/graph-database).

[15] Arora and R.R. Aggarwal, "An Algorithm for Transformation of Data from MySQL to NoSQL (MongoDB)", International Journal of Advanced Studies in Computer Science and Engineering (IJASCSE).

[16] Roberto De Virgilio, Antonio Maccioni Riccardo Torlone, "Converting Relational to Graph Databases", 2013 ACM.