ISSN No. 0976-5697

Volume 5, No. 7, September-October 2014

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Creating Graph Databases From Relational Databases: An Implementation

Mr. Deep Kumar Das, Ms. Sabiha Hussain Barlaskar
Student, MCA 6th Semester
Dept. Of Computer Applications,
Assam Engg. College, Guwahati-13, Assam, India

Mr. S. Bordoloi, Dr. B. Kalita
Associate Professor
Dept. Of Computer Applications,
Assam Engg. College, Guwahati-13, Assam, India

Abstract: In this paper, a system to convert relational databases to graph databases is presented. In this method, the graph database (Neo4j) is automatically created from the data read from the specified relational database

Keywords: Graph Database, Domain Relationship Diagram, E-R Diagram, Domain Oriented Dependency, Tuple Oriented Dependency, Composite Keys.

I. INTRODUCTION

Graph database is now becoming the popular area of research in database technology. Graph databases are mainly used when there are complex relationships among data [2][4][5][11]. Few examples of such applications with complex relationships among the data include social networks[10], genomic analysis, etc. In graph databases, data are organized as graphs- nodes represent the data item and the edges represent the interactions among the data items. The edges can also store information [4]. When there are many relationships between two entities, relational database requires many tables to store these relationships depending on their types and cardinality. If the relationships are more complex, it is impossible to represent them in relational database.

A good number of graph databases have been developed in the recent years and they have their own merits and demerits depending on the graph models used, functionality, usability in heterogeneous application domains [3]. Mike Buerli [5], discussed about the application of different graph database models and categorized them into different types. Ravinsingh Jain, et. Al.[7], discussed about few nosql-like databases including Google's "Big Table", Facebook's Cassandra, Amazon's dynamodb and Linkedin's Project Voldemort. There are several other graph databases like Neo4j[10], infinitegraph, infogrid, DEX, allegrograph etc.[3]. Leonid Libkin, et. Al.[6], concluded that xpath-like languages can also be applied to graph databases in addition to dedicated graph database query languages like Cypher Query[10] etc. Yuanyuan Tian presented in his Ph.D. Thesis about different graph matching algorithms like SAGA, TALE and two aggregation operations– SNAP and k-SNAP[9]. Abdurashid Mamadolimov, in his paper [8], has presented few search algorithms for conceptual graph databases.

II. THE PROPOSED SYSTEM

The data are extracted from the relational database (here MySQL) using JDBC. The user gets a list of all the databases present and can choose which relational database to be converted into graph database.

```
List of Databases:
Database1: information_schema
Database2: mysql
Database3: neo4j
Database4: sup_part
Database5: test
Please enter name of the relational database to be converted into graph database:
```

Figure: 1

When a database is chosen, the conversion is done table wise selecting data from the table field wise and then performing necessary operation based on the algorithm given below

This is done for each field in all the tables in the selected database. After this, the graph gets created in Neo4j using the data. The graph database model is then viewed using a browser.

```
Please enter name of the relational database to be converted into graph database: sup_part
Connecting to database...
List of Tables in Database 'sup_part':
Table1: part
Table2: shipment
Table3: supplier
Selecting table [supplier] to convert into graph database.
No of col= 3
Column1 - sup_no
Column2 - sup_name
Column3 - city
Primary Key is : sup_no
Table [supplier] converted to graph database.
Selecting table [part] to convert into graph database.
No of col= 3
Column1 - part_no
Column2 - part_name
Column3 - color
Primary Key is : part_no
Table [supplier] converted to graph database.
Selecting table [shipment] to convert into graph database.
No of col= 4
Column1 - supplier
Column2 - parts
Column3 - date
Column4 - qty
Table [shipment] converted to graph database.
```

Figure: 2

A. Software Requirements:

Operating System: Windows 7

Software Tool: MySQL, Neo4j, Eclipse IDE, Mozilla Firefox

Language used: Java, Structured Query Language, Cypher Query

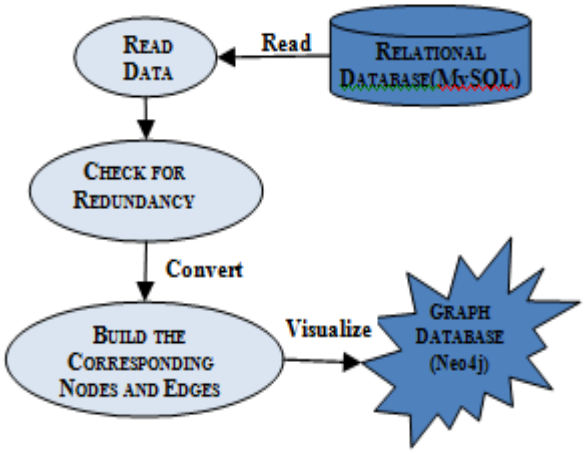

Figure 3: The Proposed System

B. The Algorithms.

a. Algorithm For Creating Domain Relationship Model:

Step1: Develop an E-R model by using reverse engineering approach. [12].

Identify the unique domains for the attributes.

Step 2: Develop the Domain Relationship Model

a) For the entities create relationships (1: n) from the key attribute to all describing attributes. For entities having composite key we get an n-ary relationship.

b) For n: 1 relationship in the E-R model, create an 1: n relationship from the key of n-side entity to the key of the 1-side entity. For m:n relationship in the E-R model, create m:n n-ary relationship from the keys of the participating entities in the m:n relationship and other distinguishing attributes in the m:n relationship to the describing attributes of the m:n relationship.

b. Algorithm For Creating Graph Database Model From The Table Data:

Step1: Read the Table data and find the distinct domain values in the database.

Step2: Create nodes for these distinct domain values.

Step3: Create edges for the relationship following the Domain Relationship Diagram. For each row in a Table create edges from the domain value for the key in the table to other non-key domain values and label the edge with the attribute names. If the Table has composite key then create a hyper edge to connect the key domain values to other non-key domain values.

III. IMPLEMENTATION

At first, the Relational Database (MySQL) is connected with Eclipse with the help of MySQL JDBC connector and the connection strings are shown in *fig. 4*.

```
Class.forName("com.mysql.jdbc.Driver");
StringDB_URL= jdbc:mysql://localhost:3306/";
```

Figure 4: JDBC Connection String

Next, the list of all the MySQL databases is displayed to the user using JDBC code (*See fig.5*).

```
conn=DriverManager.getConnection(URL,USER,PASS );
DatabaseMetaData dbmd = conn.getMetaData();
ResultSet ctlgs = dbmd.getCatalogs();
int dbcount=0;
System.out.println("List of Databases: ");
 while (ctlgs.next())
  {
  dbcount++;
  System.out.println("Database"+dbcount+":
              "+ctlgs.getString(1));
 }
```

Figure 5: Sample Code to List the MySQL Databases

Now the user inputs the database which is to be converted into the corresponding GraphDB.

Data are extracted table wise using JDBC ResultSet and each field is read and checked to see if a node with same field value is present in graph database created. If not present then it is checked whether it is a primary key. If it is a primary key then a node is created with the field value and the above process is continued for other fields, else if it is not a primary key then a node is created using the field value and then a relationship is established between the node and its corresponding primary key node. If a node with the field name is already present in the graph database created then a relationship is established between the node and its corresponding primary key node.

```
stmt = conn.createStatement();
String sql;
System.out.println("Selecting table ["+tab1+"] to convert into graph
database.");
sql = "SELECT * FROM "+tab1;
ResultSet  rs = stmt.executeQuery(sql);
node1= graphDb.createNode();
node1.setProperty("name", var);
node2= graphDb.createNode();
node2.setProperty("name", var);
node2.createRelationshipTo(node1,relType);
```

Figure 6: Sample Code to create Nodes and Relationships

The graph database system (Neo4j) is connected with eclipse by adding the libraries in Neo4j community package.

In this way, all the fields in each row of all the tables in the relational database are converted into nodes and edges of the graph database.

IV. OUTPUTS

A. Displaying Complete Database:

SQL Query:

```
SELECT * FROM SUPPLIER;
SELECT * FROM PART;
SELECT * FROM SHIPMENT;
```

Cypher Query:

```
START N=NODE (*) WHERE (N.NAME <> "" AND
N.NAME<> NULL) RETURN N.NAME;
```

```
mysql> select * from supplier;
+--------+----------+---------+
| sup_no | sup_name | city    |
+--------+----------+---------+
| s1     | Ram      | Kolkata |
| s2     | Hari     | Mumbai  |
| s3     | Bob      | Mumbai  |
+--------+----------+---------+
3 rows in set (0.00 sec)

mysql> select * from part;
+---------+-----------+--------+
| part_no | part_name | color  |
+---------+-----------+--------+
| p1      | Nut       | Red    |
| p2      | Bolt      | Green  |
| p3      | Screw     | Blue   |
| p4      | Bolt      | Red    |
+---------+-----------+--------+
4 rows in set (0.00 sec)

mysql> select * from shipment;
+----------+--------+----------+------+
| supplier | parts  | date     | qty  |
+----------+--------+----------+------+
| s1       | p1     | 1/1/13   | 100  |
| s1       | p2     | 1/1/13   | 200  |
| s2       | p1     | 1/1/13   | 200  |
| s2       | p3     | 2/1/13   | 300  |
| s3       | p1     | 2/1/13   | 300  |
| s3       | p3     | 2/1/13   | 200  |
+----------+--------+----------+------+
6 rows in set (0.00 sec)
```

Figure 7: SQL Query for Displaying Complete Database.

```
neo4j-sh (0)$ start n=node (*) where (n.name <>"" and n.name<>null) return n.name;
==> +-----------+
==> | n.name    |
==> +-----------+
==> | "s1"      |
==> | "Ram"     |
==> | "Kolkata" |
==> | "s2"      |
==> | "Hari"    |
==> | "Mumbai"  |
==> | "s3"      |
==> | "Bob"     |
==> | "p1"      |
==> | "Nut"     |
==> | "Red"     |
==> | "p2"      |
==> | "Bolt"    |
==> | "Green"   |
==> | "p3"      |
==> | "Screw"   |
==> | "Blue"    |
==> | "p4"      |
==> | "1/1/13"  |
==> | "100"     |
==> | "200"     |
==> | "2/1/13"  |
==> | "300"     |
==> +-----------+
==> 23 rows
==> 384 ms
```

Figure 8: Cypher Query to Display the Complete Database

The corresponding Graph is as shown in *fig. 9*.

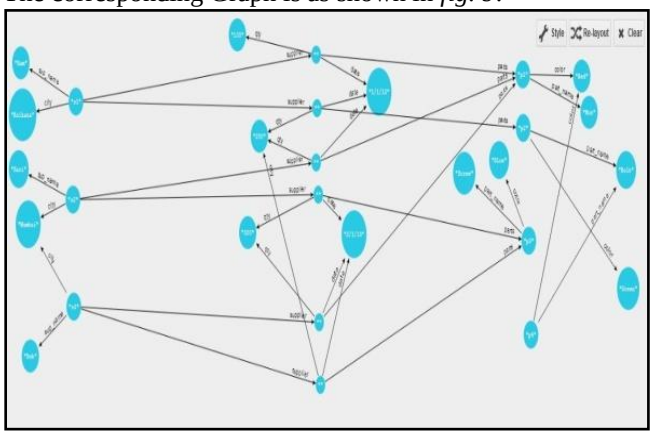

Figure 9: Graph Obtained For the Cypher Query In Fig.8

B. Display the Details of Supplier with Supplier No. 'S1':

SQL Query:

SELECT * FROM SUPPLIER WHERE SUP_NO='S1';

Cypher Query:

Start n=node (*) match n-[r]->b where n.name='s1'and b.name<> null return n.name,r,b.name;

```
mysql> select * from supplier where sup_no='s1';
+--------+----------+---------+
| sup_no | sup_name | city    |
+--------+----------+---------+
| s1     | Ram      | Kolkata |
+--------+----------+---------+
1 row in set (0.00 sec)
```

Figure 10: SQL Query for Displaying Details of Supplier 'S1'

```
neo4j-sh (0)$ start n = node(*) match n-[r]->b where n.name = 's1' and b.name <>"" return n.name,r,b.name;
==> +-------------------------------------------+
==> | n.name | r                  | b.name      |
==> +-------------------------------------------+
==> | "s1"   | :sup_name[46816]{} | "Ram"       |
==> | "s1"   | :city[46817]{}     | "Kolkata"   |
==> +-------------------------------------------+
==> 2 rows
==> 390 ms
```

Figure 11: Cypher Query for Displaying Details of Supplier 'S1'

The corresponding graph is shown in *fig. 12*.

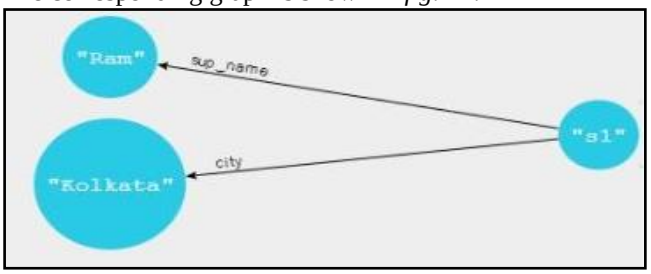

Figure 12: Graph Obtained for the Cypher Query in Fig.11

C. Display the Details of Part with Part No. 'P1':

SQL query:

SELECT * FROM PART WHERE PART_NO= 'P1';

Cypher Query:

Start n=node (*) match n-[r]->b where n.name='p1'and b.name<>null return n.name, r, b.name;

```
mysql> select * from part where part_no='p1';
+---------+-----------+-------+
| part_no | part_name | color |
+---------+-----------+-------+
| p1      | Nut       | Red   |
+---------+-----------+-------+
1 row in set (0.00 sec)
```

Figure 13: SQL Query for Displaying Details of Part 'P1'

```
neo4j-sh (0)$ start n = node(*) match n-[r]->b where n.name = 'p1' and b.name <>"" return n.name,r,b.name;
==> +-------------------------------------------+
==> | n.name | r                  | b.name      |
==> +-------------------------------------------+
==> | "p1"   | :part_name[46822]{} | "Nut"      |
==> | "p1"   | :color[46823]{}     | "Red"      |
==> +-------------------------------------------+
==> 2 rows
==> 419 ms
```

Figure 14: SQL Query for Displaying Details of Part 'P1'

The corresponding Graph is as shown in *fig. 15*.

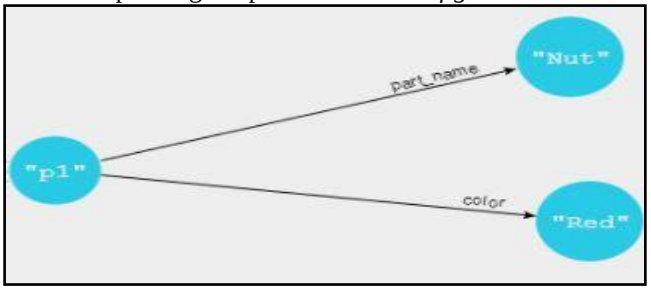

Figure 15: Graph Obtained for the Cypher Query In Fig.14

D. Display The Shipment Details Of Supplier 'S1':

SQL Query:

SELECT * FROM SHIPMENT WHERE SUPPLIER= 'S1';

Cypher Query:

Start a=node (*) match a-[r1]->b-[r2]->c return a.name,r2,c.name;

```
mysql> select * from shipment where supplier='s1';
+----------+-------+---------+-------+
| supplier | parts | date    | qty   |
+----------+-------+---------+-------+
| s1       | p1    | 1/1/13  | 100   |
| s1       | p2    | 1/1/13  | 200   |
+----------+-------+---------+-------+
2 rows in set (0.00 sec)
```

Figure 16: SQL Query to Display the Shipment Details of Supplier 'S1'

```
neo4j-sh (0)$ start a = node(*) match a-[r1]->b-[r2]->c where a.name = 's1' return a.name,r2,c.name;
==> +-----------------------------------+
==> | a.name | r2             | c.name  |
==> +-----------------------------------+
==> | "s1"   | :parts[46835]{} | "p2"    |
==> | "s1"   | :qty[46837]{}   | "200"   |
==> | "s1"   | :date[46836]{}  | "1/1/13" |
==> | "s1"   | :parts[46831]{} | "p1"    |
==> | "s1"   | :qty[46833]{}   | "100"   |
==> | "s1"   | :date[46832]{}  | "1/1/13" |
==> +-----------------------------------+
==> 6 rows
==> 203 ms
```

Figure 17: Cypher Query to Display the Shipment Details of Supplier 'S1'

The corresponding Graph is as shown in *fig. 18.*

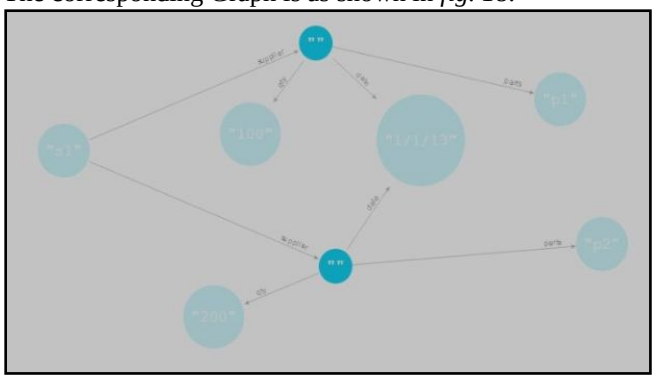

Figure 18: Graph Obtained for the Cypher Query in Fig.17

VII. CONCLUSION

This paper presents an implementation of the concepts discussed by Subhrajyoti Bordoloi, et.al.[1]. For this implementation the Domain Oriented Dependency approach is considered. Now, the method is implemented only to convert MySQL databases to graph database (Neo4j) models. But this approach can also be implemented to convert any kind of relational databases to graph database model. Implementation of Tuple Oriented Dependency model is also possible with minor changes in the algorithms used.

VIII. REFERENCES

[1] Subhrajyoti Bordoloi, Bichitra Kalita, "Designing Graph Database Models from existing relational databases", International Journal of Computer Applications (0975–8887) Volume 74 No.1, July, 2013, pp. 25-31.

[2] Philippe Cudré-Mauroux, Sameh Elniketyt, "Graph Data Management Systems for New Application Domains", Proceedings of the VLDB Endowment, Vol. 4, No. 12, 2011, pp-1510-1511.

[3] Darshana Shimpi ,Sangita Chaudhari "An overview of Graph Databases**",** International Conference in Recent Trends in Information Technology and Computer Science(ICRTITCS-2012) Proceedings published in International Journal of Computer Applications® (IJCA) (0975 – 8887), pp. 16-22, 2012.

[4] Shalini Batra, Charu Tyagi ,"Comparative Analysis of Relational And Graph Databases" International Journal of Soft Computing and Engineering (IJSCE) Volume-2, Issue-2, May 2012 ,pp-509-512.

[5] Mike Buerli,"The Current State of Graph Databases" Department of Computer Science, Cal Poly San Luis Obispo, mbuerli@calpoly.edu, pp.1-7, Dec. 2012

[6] Leonid Libkin, Wim Martens, Domagoj Vrgoc, "Querying Graph Databases with XPATH", ICDT '13 Proceedings of the 16th International Conference on Database Theory, pp. 129-140, ACM (2013).

[7] Ravinsingh Jain, Srikant Iyengar, Ananyaa Arora, "Overview of Popular Graph Databases", Proceedings of the Fourth IEEE International Conference on Computing, Communication and Networking Technologies, July 4-6, 2013, pp. 18-23.

[8] Abdurashid Mamadolimov, "SEARCH ALGORITHMS FOR CONCEPTUAL GRAPH DATABASES", International Journal of Database Management Systems (IJDMS) Vol.5, No.1, February 2013, pp. 35-44.

[9] Yuanyuan Tian, "QUERYING GRAPH DATABASES", PhD Thesis, Computer Science and Engineering, The University of Michigan, 2008.

[10] Justin J. Miller, "Graph Database Applications and Concepts with Neo4j", Proceedings of the Southern Association for Information Systems Conference, Atlanta, GA, USA, pp.141-147, March 23rd-24th, 2013.

[11] Ian Robinson, Jim Webber, and Emil Eifrem, "GRAPH DATABASES", Copyright (c) 2013 Neo Technology, Inc. , Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472., June 2013: First Edition.

[12] Subhrajyoti Bordoloi, Bichitra Kalita, "E-R Model to an Abstract Mathematical Model for Database Schema using Reference Graph", International Journal of Engineering Research And Development, e-ISSN:2278-067X, p-ISSN:2278-800X, Vol. 6, Issue 4, March 2013, pp. 51-60.