Volume 3, Issue 11, November 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

An Effective Approach to Understand the Graph Database with Shortest path Algorithm

Nikhlesh Patidar*, Digvijay Singh
M.C.A
SITE,VIT University Vellore, India

Varun Kumar.M, Deepa.P
Assistant Professor
SITE,VIT University Vellore, India

Abstract: A graph database is a database that uses graph structure with nodes, edges, and properties to represent and store data. By definition, a graph database is any storage system that provides index-free adjacency. This means that every element contains a direct pointer to its adjacent element and no index[2] lookup are necessary. General graph databases that can store any graph are distinct from specialized graph databases such as triple stores and network database. Graph databases are based on graph theory. Graph databases employ nodes, properties, and edges. Nodes are very similar in nature to the objects that object-oriented programmers will be familiar with Nodes represent entities such as people, businesses, accounts, or any other item you might want to keep track of. Properties are pertinent information that relate to nodes.

Keywords: Edges, Nodes, Graph theory, Neo4jTool[1], Shortest Path Algorithm.

I. INTRODUCTION

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, generally a graph database having various nodes and each nodes having some relationship[3] among the other's with index positions also generally graph database follow the acid properties as well as relationship among the nodes with indexing position also , a graph database is more faster than the other relational databases as well as for more fast result , there are lots of other shortest path algorithm's, which can be used to process and fetch the result more faster.

II. GRAPH CONTAINS NODES AND RELATIONSHIPS

A Graph[1] – [:RECORDS_DATA_IN]–> Nodes – [:WHICH_HAVE]–> Properties.
The simplest possible graph is a single Node, a record that has named values referred to as Properties. A Node could start with a single Property[4] and grow to a few million, though that can get a little awkward. At some point it makes sense to distribute the data into multiple nodes, organized with explicit Relationships[5].

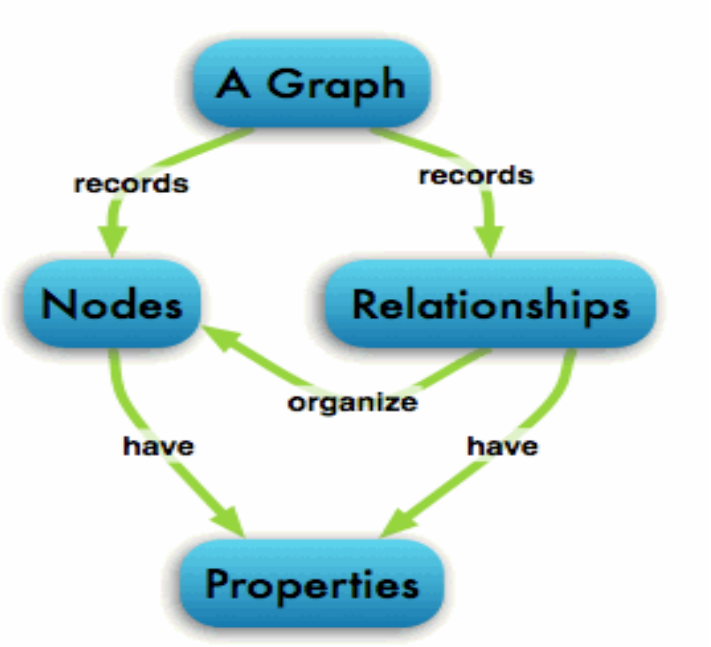

Fig.:1 Nodes and Relationships

III. QUERY A GRAPH WITH A TRAVERSAL[6]

A Traversal –navigates–> a Graph; it –identifies–> Paths –which order–> Nodes.

A Traversal is how you query a Graph, navigating from starting Nodes to related Nodes according to an algorithm, finding answers to questions like "what music do my friends like that I don't yet own," or "if this power supply goes down, what web services are affected?

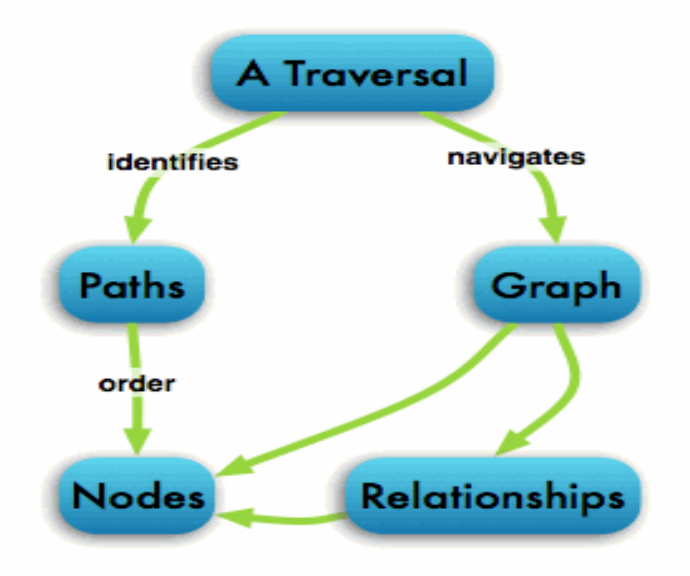

Fig.: 2 Graph with Traversal

IV. INDEXES LOOK-UP NODES OR RELATIONSHIPS[7]

An Index –maps from–> Properties –to either–> Nodes or Relationships .It –is a special–> Traversal .Often, you want to find a specific Node or Relationship according to a Property it has. This special case of Traversal is so common that it is optimized into an Index look-up, for questions like "find the Account for username master-of-graphs[5].

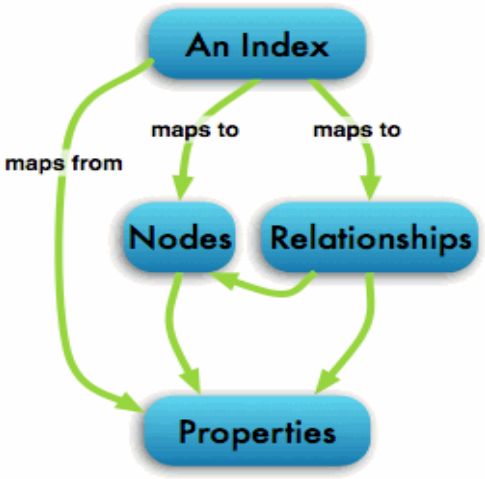

Fig.: 3 Index Lookup Nodes or Relationships

V. PROPOSED ALGORITHM

Dijkstra algorithm :

Initialize single-source (node) (G, s)
S ← { } // S will ultimately contains vertices of final shortest-path weights from s
Initialize priority queue Q i.e., Q ← V[G]
while priority queue Q is not empty do
  u ← EXTRACT_MIN(Q) // Pull out new vertex
  S ← S È {*u*}
  // Perform relaxation for each vertex *v* adjacent to u
  for each vertex v in Adj[u] do
    Relax (*u*, *v*, *w*)

Implementation: Execution of a Dijkstra algorithm[4] with weights on relationship.

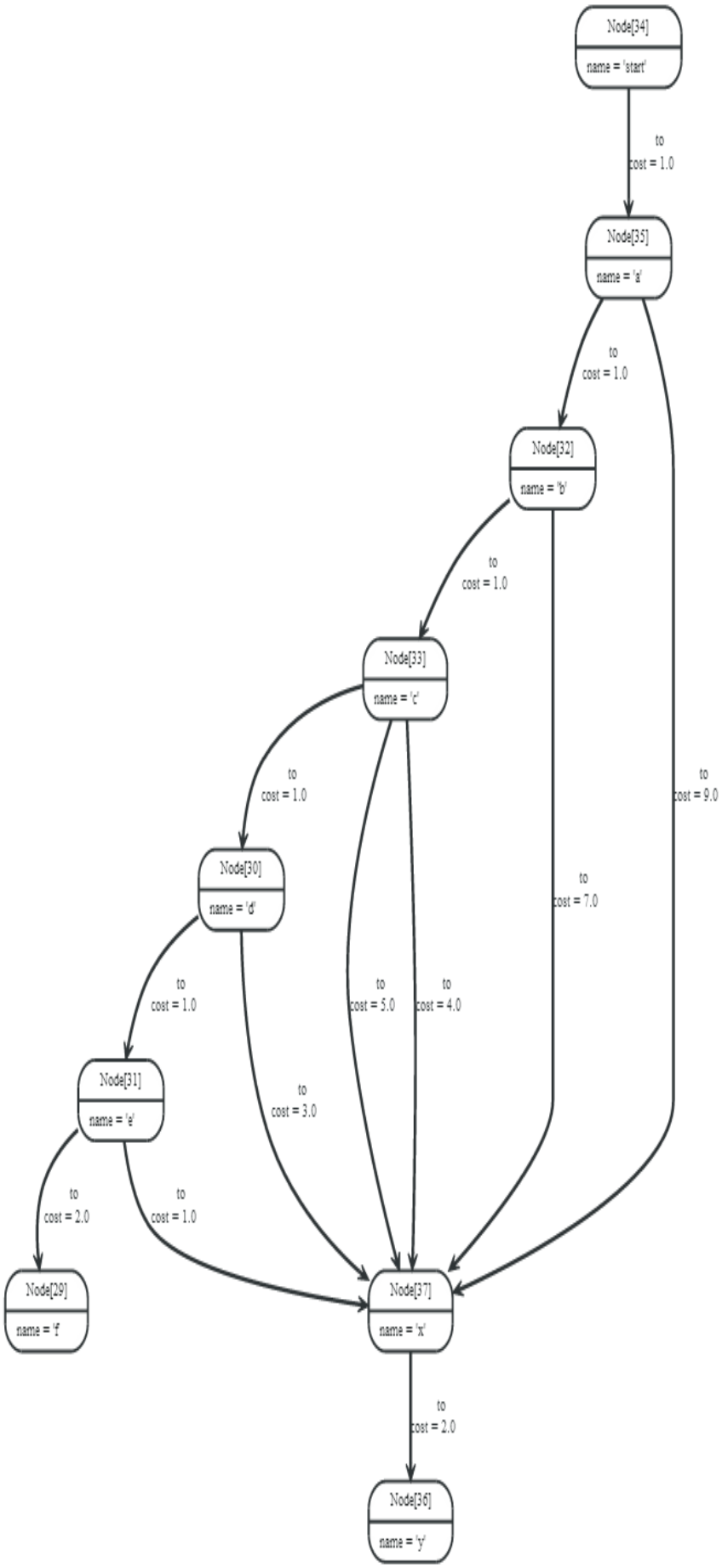

Fig.: 4 Implementation

Example request:

POST: Full Path of Graph Database with Post Request Method.
Accept: Application; charset=UTF-8
Content-Type: Application
{
"to" : "node37 (Node For Searching)" ,
"cost_property" : "cost",
"relationships" : {
"type" : "to",
"direction" : "out"
},
"algorithm" : "dijkstra"
}
Example response

Content-Type: application; charset=UTF-8
{
"weight" : 6.0,
"start" : " node34",
"nodes" : ["node34","node35", "node32", "node33", "node30", "node31","node37"],
"length" : 6,
"relationships" : ["relationship38","relationship40","relationship42","relationship45", "relationship47", "relationship48"],
"end" : "node37"
}

VI. CONCLUSION:

The graph data base is more faster than the normal database , we used neo4j tool for graph database, Based on our measure, the database that obtained the best results with traversal workloads is definitely Neo4j[1]: it outperforms all the other candidates, regardless the workload or the parameters used . it is more efficient than the normal database , As future work we plan to develop an empirical evaluation of current graph databases; this oriented to make a quantitative and qualitative analysis of their support for storing and querying graph data.

REFERENCES:

[1.] D. J. Cook and L. B. Holder (2000) *Graph-Based Data Mining*, IEEE Intelligent Systems, 15(2).
[2.] L. Getoor, Friedman, N., D. Koller, B. Taskar (2001). *Probabilistic Models of Relational Structure.* International Conference on Machine Learning, Williamstown, MA.
[3.] F. V. Jensen (2001). *"Bayesian Networks and Decision Diagrams".* Springer.
[4.] Renzo Angles and Claudio Gutierrez. Survey of graph database models. ACM Comput Surv., 40(1):1:1{1:39, February 2008.
[5.] J. R. Ullman, "An Algorithm for Subgraph Isomorphism", Journal of the Association for Computing Machinery, 1976, Vol. 23, pp. 31-42.
[6] P. Willett, J. Barnard, and G. Downs, "Chemical similarity searching", J. Chem. Inf. Comput. Sci., 1998, Vol. 38, No. 6, pp. 983-996.

