VOL. 12, NO. 1, JANUARY 2017 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

EFFICIENT RETRIEVAL BY DYNAMIC SUBGRAPH MATCHING IN A HUGE GRAPH DATABASE

Anantha Priya V., Kalaivani K. and Ulagapriya K.
Department of Computer Science and Engineering, Vels University, Chennai, India
E-Mail: kalaivani_k@outlook.com

ABSTRACT

The objective of this paper is to reduce the retrieval time while processing a query graph. Graph is the powerful way to analyse a large database. Any given data can be easily visualized in graphical format. Sub graph matching is one of the most important factor when dealing with huge database. It retrieves structurally isomorphic sub graph when comparing query data with large graph data. The isomorphic problem has been overcome with the help of node Index. Neo4j is a powerful tool, which is used to visualize the query in graph method and to retrieve the matching sub graph in faster way. A pruning method is followed along with a Dynamic Sub Graph-Matching algorithm (DSMA) to solve the isomorphic problem. Rich information is available in every vertex of a social network graphs and it is effectively used while querying.

Keywords: dynamic sub graph matching, isomorphism, correlate, optimization.

1. INTRODUCTION

The easy way to analyse a tremendous data is the graphical visualisation of the actual data. It can be done by showing the patterns, highlighting chunk and their connections. Many experiments has used different types of graph queries, in which the basic graph query type is sub graph matching [1] [3][5]. A graph is a set of nodes which visualises the relationship and the connection between each node. Properties are assigned to each node inorder to represent a data. Neo4j server is based on built-in D3.js library which contains a powerful and customizable date visualization tool. Most of the social networking applications contain lot of structured, semi structured and unstructured data which are connected together. The graph database can store more kind of connected data. Neo4j does not need complex joins to recover the related data because of which the recovery is very faster. Neo4j uses Native GPE (Graph Processing Engine) to work with its Native graph storage format. Perfect matching state is achieved only when all the nodes and vertices are perfectly matched.

Many existing sub graph matching query is based on three different methods namely [7] [9] Indexing Structure, graph alignment method and Similarity based method [4] on a large graph database. But all the three methods can be clubbed together in the existing system [8] [13]. Ullmann Algorithm was described in 1976 which first solved the isomorphic problem in large graph database. It takes about a polynomial time for a fixed choice of data [10]. The generic sub graph isomorphic algorithm divides the complete graph data into small fragments. Then it probes index to identify similar fragments in large graph. The matches are joined together and sub graph is generated which involves a high retrieval time.

The data is described interms of pattern. Normally circle shape is used to represent a data as node. The arrows are represented as relationship between each node. In Neo4j multiple nodes and their relationship can be described using the patterns. An array of connected nodes and relationship is defined as a path. The path can be assigned using path identifier.

2. GENERIC SUB GRAPH ISOMORPHISM ALGORITHM

The sub graph Isomorphism algorithm is used to obtain the matching sub graphs [13] [10]. Sub graph distance method is used to identify the matching queries. An index is built on small fragments of graph data. While processing query, Query graph is divided into small fragments. It probes the index to identify the similar fragments in a large graph database. The match fragments are joined together to form a larger match as shown in Figure-1.

The work flow of an algorithm is as follows:
Input: Query graph (Q), data graph (G), vertex (u)
Output: (sub graph)

- Begins with the filter candidate to get the set of candidate vertices.
- The search will end if the candidate vertex search is null.
- Mapping pairs of query and data vertex is done by using the sub graph search.
- Query vertex is selected which is not yet matched.
- Pruning rules are used to refine the candidate vertex set.
- Joinable subroutine checks for the edges and already matched query vertex.
- Update the UPDATESTATE and proceed further checking if the matching is accurate.
- Modications are done by UPDATESTATE.
- Follow step 3 till 8 until the complete matching is done.

On applying this algorithm the following sub graph matching structure is obtained.

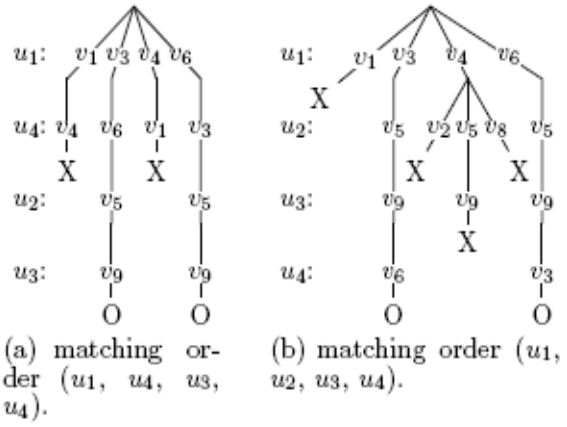

Figure-1. Subgraph matching.

3. DYNAMIC SUB-GRAPH MATCHING

Dynamic sub graph matching algorithm (DSMA) is proposed to find the exact match between the query graph and data graph. The processing time can be reduced while implementing in neo4j graph database since it has its own inbuilt matching component. Neo4j has powerful, customizable data visualization tool based on built in D3.js library. Target JSON data structure toolkit has a JSON structure of node objects with their Ids, labels and properties, along with the list of relationship. A graph model is derived where it converts the rows into node and table name into label name without any null values. DSMA retrieves the exact matching data with very minimum response query processing time. Many bench mark problems were examined to demonstrate the effectiveness of this sub graph matching technique and pruning techniques. The proposed model (Figure-2) is effective in solving the isomorphic problem and can be implemented in real world situations. The Sub graph retrieval time and the cost involved make the project more successful and reliable.

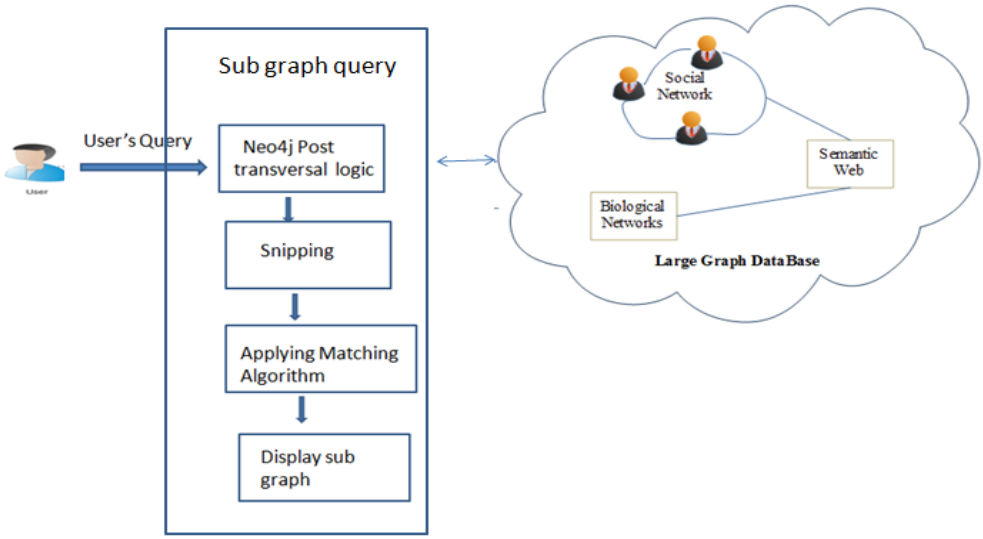

Figure-2. Architecture diagram.

3.1 Graph visualization in Neo4j

Neo4j uses two major guidelines while deriving a graph model from a relational model.

- The row is considered as a Node.
- A table name is considered as label Name.

The Relationship between the nodes is explained briefly with no nulls identified. Exporting the data to CSV is done by CSV command. It executes a SQL query and writes the result to CSV file. After exporting the data from PostgreSQL, cipher's load CSV command transforms the content of CSV into a graph structure. First step is to create the nodes. Second step is to create the relationship between the neighbouring nodes. We create indexes on the next generated nodes to map their lookup and relate them accordingly. The relationship is created considering their different categories and along with the data and indices. Finally the graph structure is represented for the given data by creating a 'REPORTS_TO' relationship between nodes. The Neo4j scripting is used to completely represent the relationship in a table format. For achieving an optimal speed, a unique constraint is created.

4. SNIPPING

Snipping is technique that minimizes the size of a tree. It does not reduce the predictive accuracy as measured by cross validation set.

4.1. Upward snipping

The prefix filtering logic is used in the upward pruning. If two sets are same, then overlapping of prefixes is done between those sets. The arrangement of the data is done descending order according to their weights. The maximum prefix size is identified. The sets are pruned

when the overlapping is null in prefix. A similarity threshold is set. The prefix is obtained by removing the elements with largest weight from the given set. It also checks whether the remaining sets meet the threshold value with the actual data set. After finding the prefix we can access all the descent nodes in vertical manner.

4.2. Uniform snipping

The size of the frequent pattern of big data cannot be same as that of small set. For a given inverted pattern every frequent pattern is subset of element set in inverted order. The upper bound length of the query set is identified by adding nodes in increasing order according to their weights. For every new insertion a new set is added. The similarity value is matched and if it passes, the elements are added further. After obtaining the upper bond length, the horizontal pruning is applied and the patterns are successfully pruned. Both the pruning techniques are clubbed and the final pruned result is retrieved. After using upward and uniform pruning techniques the duplicate patterns are removed and other nodes are connected together the breadth first search is used to traverse them.

5. DYNAMIC SUBGRAPH MATCHING ALGORITHM (DSMA)

A matching algorithm is used to retrieve the perfect match. A threshold value is set for a given query Q. The algorithm works as follows:

- Begin with root node (R) with index (I).
- Prune the entries in the root that does not match query vertex
- A new node is inserted if the entry cannot be pruned.
- The tree index is traversed in breadth first manner.
- Pruning methods are now applied to the child node for all leaf or non leaf node.
- A sub graph is added to leaf node, if the child node cannot be pruned.
- The child node is inserted in the candidate set for further filtering.
- New candidate set is obtained after index traversal.
- Then the nodes are connected together removing the duplicates.

The sub-graph formed in Neo-4j is shown in Figure-3.

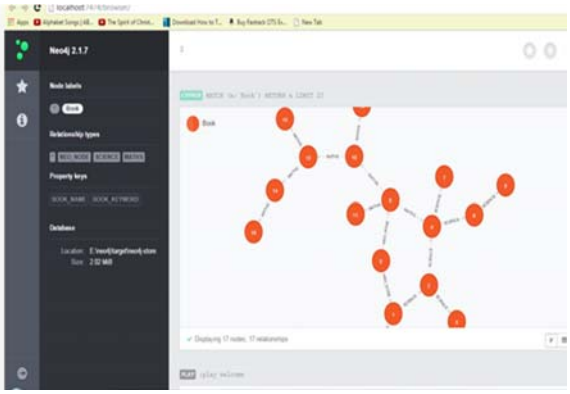

Figure-3. Sub graph in Neo4j.

The query response time of both generic sub-graph matching and dynamic sub-graph matching is analyzed. The performance of DSMA is better while querying the large graph database. The performance and sample subgragh generation is shown.

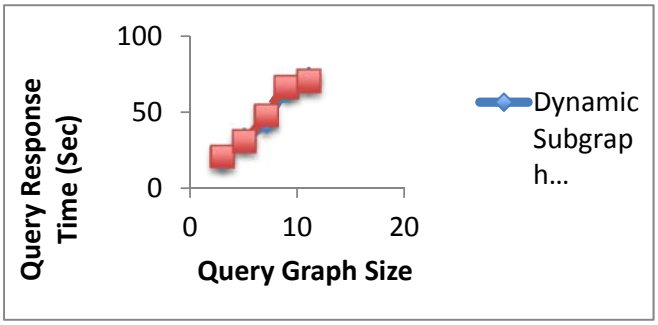

Figure-4. Query response time.

6. CONCLUSIONS

An efficient sub graph matching algorithm is proposed which is essential to address the set similarity and isomorphic issues in a wide range of applications. Both the graph topology and vertex data are considered to derive an efficient dynamic pruning technique. The proposed technique is good at retrieving the perfect matches from the rich data available in the vertex. The matching quality score and query response time is comparatively high, which results in faster sub graph extraction using Neo4j.

REFERENCES

[1] L. Zou, L. Chen and M. T. Ozsu. 2009. Distance-join: Pattern match query in a large graph database. PVLDB. 2(1).

[2] P. Zhao and J. Han. 2010. On graph query optimization in large networks. PVLDB. 3(1-2).

[3] Z. Sun, H. Wang, H. Wang, B. Shao and J. Li. 2012. Efficient sub graph matching on billion node graphs. PVLDB. 5(9).

[4] W. Lu, J. Janssen, E. Milios, N. Japkowicz and Y. Zhang. 2006. Node similarity in the citation graph. Knowledge and Information Systems. 11(1): 105-129.

[5] L. P. Cordella, P. Foggia, C. Sansone and M. Vento. 2004. A (sub) graph isomorphism algorithm for matching large graphs. PAMI. 26(10).

[6] W.-S. Han, J. Lee and J.-H. Lee. 2013. Turboiso: Towards ultrafast and robust sub graph isomorphism search in large graph databases. in SIGMOD.

[7] H. He and A. K. Singh. 2006. Closure-tree: An index structure for graph queries. in ICDE.

[8] J. R. Ullmann. 1976. An algorithm for sub graph isomorphism. Journal of the ACM. 23(1).

[9] S. Zhang, M. Hu and J. Yang. 2007. Treepi: A novel graph indexing method. in ICDE. 7: 966975.

[10] J. Lee, W.-S. Han, R. Kasperovics and J.-H. Lee. 2012. An indepth comparison of subgraph isomorphism algorithms in graph databases. Proceedings of the VLDB Endowment. 6(2): 133-144.

[11] L. Zou, L. Chen and Y. Lu. 2007. Top-k sub graph matching query in a large graph. in PIKM, 2007.

[12] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. 2008. Fast indexes and algorithms for set similarity selection queries. in ICDE.

[13] Y. Tian, R. C. McEachin, C. Santos, D. J. States and J. M. Patel. 2007. Saga: a subgraph matching tool for biological graphs. Bioinformatics. 23(2): 232-239.

[14] Y. Yuan, G. Wang, H. Wang and L. Chen. 2011. Efficient subgraph searchover large uncertain graph. In PVLDB.

[15] X. Yan, P. S. Yu and J. Han. 2005. Substructure similarity search in graph databases. In SIGMOD.