Socially Aware Tiered-Indexing for Efficient Social Query Processing on Big Social Data

Haesung Lee and Joonhee Kwon

Department of Computer Science, Kyonggi University
San 94-6, Yiui-dong, Yeongtong-ku, Suwon-si, Gyeonggi-do, Korea
fseastar0202, kwonjh }@kgu.ac.kr

Abstract. Efficient social query processing is needed for big social network data. However, the previous approaches do not consider characteristics of the social network. In this paper, we propose a novel indexing technique that constructs tiered index considering a social aware factor. In social network, important nodes are accessed more frequently as compared with other nodes. Given a social query, our method retrieves results promptly without traversing all data for important data. It makes query processing faster for important nodes. Our experiment shows that the social search system using proposed indexing technique is more efficient than using the existing graph index.

Keywords: Big social data, Tiered index, Social query processing.

1 Introduction

With the explosive increase of the social data, it has become more and more demanding to access big social data promptly [1][2]. However, despite increasing needs for efficient social query processing, there are few efficient indexing techniques for social query processing considering characteristics of the social data.

In this paper, we propose an efficient index structure for a large amount of social data. Not surprisingly, important nodes are accessed more frequently as compared with other nodes in social network. We focus on a few important nodes among many other nodes on social big data. In this paper, we adopt tiered index approach based on the observation that important data is needed to search faster than other data. Also, we use social network analysis for selecting important nodes. Social network analysis has been used to examine characteristics of the social data. This enables socially aware indexing.

The rest of the paper is organized as follows. Section 2 reviews related works. Section 3 and Section 4 describe preliminary concepts and our method. Section 5 presents our experimental results. Finally, Section 6 concludes this paper.

2 Related Works

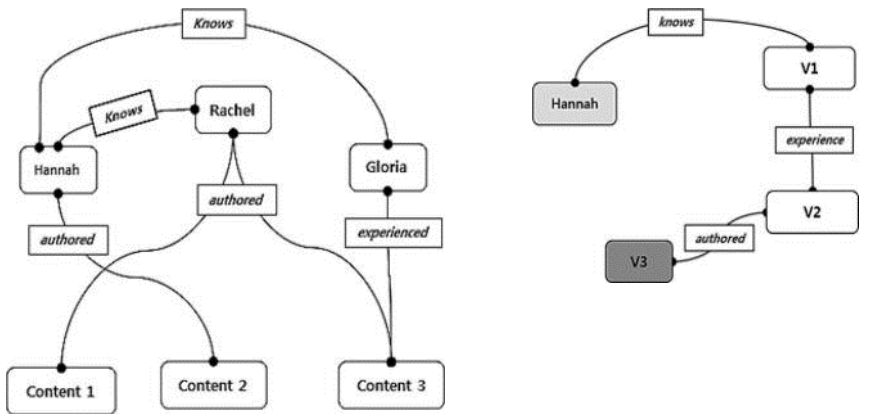
Because of explosively increasing user-generated social contents, many social service companies face the challenge of the big amount of social data. This need leads to the

consideration of NoSQL (Not only SQL) and created different types of new storage systems [3]. Nicolas et al. analyzed eight different storage system of NoSQL [4]. Among analyzed eight different storage systems, Neo4j is only graph based storage system [5].

In this paper, we consider Neo4j as the existing graph based index. Neo4j uses index mechanism provided from Lucene which is the most popular open source project for the Web search engine.

3 Preliminary Concepts

We consider a social graph $G=(V,E,O^{v},0^{E})$, where each node $v_i \in V$ is labeled. $EcVxV$ is set of directed edges connecting one node to another. O^{v} is a set of properties for each node and B^{E} is a set of properties for each edge. Different with query models used in traditional keyword based query model, the social query model can be represented by the graph. For this, we consider the given query as a query graph $QG = (v_{start} \in V, V_{end} \in V, V,E,8^{1},e9^{E})$. The QG has a start node v_{start} and an end node v_{end}, where the end node is a result node for the given query.



(a) The example of a social graph

(b) An social query graph in which an result node V3, belongs

Fig. 1. Social graph and social query graph

Figure 1 (a) shows an example of a social graph. "Who has authored a content experienced by someone who knows Hannah?" is an example of our social query. Figure 1(b) presents the example query for Figure 1(a), where the node V3 is a result node for the query.

4 Socially Aware Tiered-Index

4.1 Building Tiered-Index

Social network analysis is a key technique to examine characteristics of the social data. In it, the centrality of a node belonging to a social graph determines the relative importance of the node within the social graph [6]-[8]. Among various types of measuring the centrality of a node, the degree centrality is historically first and widely considered. The degree can be interpreted in terms of the possibility of a node for being accessed whatever is flowing through the network. The node with high centrality is likely to be more accessed than other nodes with low centrality [9][10].

We focus a few important data such as nodes with high centrality among many other nodes on social big data. In this paper, we adopt the tiered index approach in traditional method. The tiered indexing is considered as the fine pruning technique that enables the search operation to rapidly find data which are considered as results to a given query [11]. As compared with the existing tiered index approach, our index uses social network analysis technique to make tiers.

Our index is an inverted-index with a dictionary and an entry. The number of tier can be adjusted by the user according to data characteristics. The tier value means the importance of social data. The tier with low tier value has more important nodes than tiers with high tier value. We divide social data into each tier according the degree centrality of each node. The value of the centrality for each tier is determined by an adjustable threshold. Figure 2 shows the overall structure of our socially aware tiered index. In this example, we assume the number of tiers as 3.

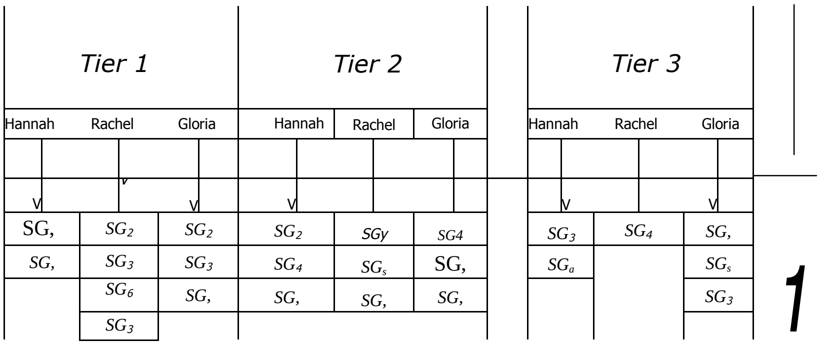


Fig. 2. The overall socially aware tiered-index structure

Our index building process is comprised of three steps. Firstly, for each tier, we make a dictionary called as person node dictionary in which the set of all person nodes *PV* = pv,} is composed of.

Secondly, we extract the set of sub social graph $SG=$ ‘ (*vstart* e V'^{V}_{end} $V,V,E,O^v,O^E)$ for each person node. A *SG* is a graph which is composed of all nodes and relationships for a specific person node. The *SG* has a start node and an end node. A

start node is a person node in the dictionary and an end node is a result node for a given query. Figure 3 shows *SG* s for person node, 'Hannah' in Figure **1.** The person node 'Hannah' has 9 *SGs.*

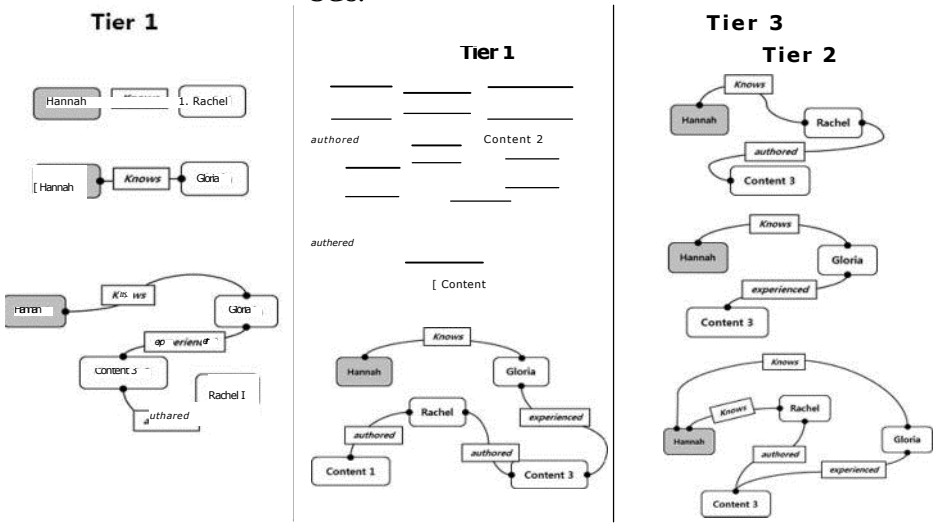


Fig. 3. The tiered entries of a person node, 'Hannah'

Thirdly, we compute the degree centrality of the end node in each *SG*. Then, we divide *SGs* of a person node into the certain tier according to the value of the degree centrality of their end nodes.

4.2 Hashing *SG* for Social Query Processing

In graph database such as social database, it takes time to traverse over the entire network to find result nodes. Our method use hashing technique to find a result node more quickly.

Hannah

Key	Value
2311	[Rachel] [Gloria]
19419	[Content 2]
34606	[Rachel]

Fig. 4. The hashed entries of a person node 'Hannah'

To hash each *SG*, we generate unique key of each *SG* based on the properties of relationships. In this method, all *SGs* which have same properties of relationships generates same hash key. The SG_i is mapped to a certain integer value as a hash key by a hash function. Figure 4 presents an example of the hash table which stores *SGs* of a person node, Hannah, in *Tier 1* as depicted in Figure 3.

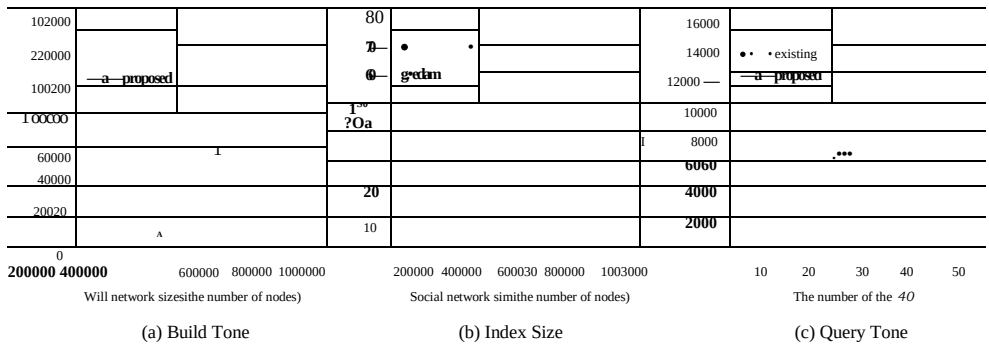
For example query in Figure 1(b), the given query graph has a start node, 'Hannah' and three properties of relationship, {knows, experienced, authored} . Our method firstly searches a person node in dictionary, where it searches data in only *Tier 1*. In the query, properties of relationship are mapped as a hash key, '34606'. We search the entry with the hash key, and then return promptly the value 'Rachel' as a result. Using tiered structure and hashed entry, our method retrieves results promptly for social query with important nodes.

5 Experiments

For the experiment, we implemented both the proposed method and existing method. The existing method was implemented by index API provided by Lucene and Neo4j. All programs were written in Java. The two methods were experimented in Intel® CoreTM Quad CPU at 2.4 GHz, and 4 GB RAM on Window 7 Enterprise K.

For the experiment data set, we randomly generated five set of social networks with the different number of nodes and relationship. Among generated networks, the social network with the largest data size has about 1 million nodes and about 4 million relationships. The number of tier was set as 3. For social query graph, the data ratio was tierl with 50%, tier 2 with 30%, and tier 3 with 20%. The total number of the data set with the social networks and the social query was 50.

We evaluated the average index build time and average index size. For comparison we varied the number of nodes of social network by 200,000 from 200,000 nodes to 1 million nodes. They show the existing method is slightly better off than our method. We found that our method shows slow build time in large social network. However, the difference is slightly low.



(a) Build Tone (b) Index Size (c) Query Tone

Fig. 5. Experiment results

We measured the average query time by varying the number of the social query. Figure 5(c) shows the result. In this experiment, we used the largest social network with 1 million nodes and 4 million relationships. The number of social query was generated from 10 to 50. The experimental result shows our proposed method outperforms consistently the existing method. Moreover, the difference between the existing method and our proposed method increases.

6 Conclusions

Despite increasing needs for efficient social query processing of the social search, there are rarely indexing techniques for the fast and proper social query processing with a large amount of social data.

In this paper, we proposed an efficient index structure of the social graph for an ideal social search operation conducted over a large amount of social data. We tried tiered index approach by the observation that important data is needed to search faster than other data. With the fact that important nodes are accessed more frequently as compared with other nodes, we used with social network analysis for selecting important nodes. Then, with the importance of nodes, we could build socially aware tiered index. We performed some experiments on our socially aware index. It showed that proposed socially aware tiered index mechanism makes it possible to improve social query processing over big social data.

Acknowledgement. This work was supported by the Gyonggi Regional Research Center (GRRC) and Contents Convergence Software (CCS) research center in Korea.

References

1. Nicholas A. Christakis and James H. Fowler, The Spread of Obesity in a Large Social Network over 32 Years, N Engl J Med 357, pp. 370-379 (2007)
2. Ravi Kumar, Jasmine Novak and Andrew Tomkins, Structure and Evolution of Online Social Networks, Link Mining: Models, Algorithms, and Applications 2010, Part 4, pp.337-357 (2010)
3. Nicolas Ruflin, Helmar Burkhart and Sven Rizzotti, Social data storage systems, Proceeding DBSocial '11 Databases and Social Networks, pp. 7-12, Athens, Greece (2011)
4. Neo4j, http://neo4j.org/
5. Lucene, http://lucene.apache.org/core/
6. Stephen **P.** Borgatti, Centrality and networkflow, Social Networks, Volume 27, Issue 1, January 2005, pp.55-71(2005)
7. Linton C. Freeman, Centrality in social networks conceptual clarification, Social Networks, Volume 1, Issue 3, pp. 215-239 (1979)
8. Scott D. Gest, Sandra A. Graham-Bermann and Willard W. Hartup, Peer Experience: Common and Unique Features of Number of Friendships, Social Network Centrality, and Sociometric Status, Social Development Volume 10, Issue 1, pp. 23-40 (2001)
9. Tore Opsahla, Filip Agneessens and John Skvoretz, Node centrality in weighted networks: Generalizing degree and shortest paths, Social Networks, Vol. 32, No. 3, pp. 245-251(2010)
10. Elizabeth Costenbader and Thomas W Valente, The stability of centrality measures when networks are sampled, Social Networks Volume 25, Issue 4, pp.283-307 (2003)
11. David M. Choy and Chandrasekaran, Multi-tiered indexing method for partitioned data, U.S. patent 5551027. Aug 27(1996)