# Efficient Recommender System based on Graph Data for Multimedia Application

Haesung Lee and Joonhee Kwon

*Department of Computer Science, Kyonggi University*
*San 94-6, Yiui-dong, Suwon-si, Gyeonggi-do, Korea*

*{seastar0202, kwonjh}@kgu.ac.kr*

***Abstract***

*With the dawning ubiquitous computing age, increasing online-based multimedia data presents new challenges for storing and querying large amounts of data to online recommendation systems. Recent studies on recommendation systems show that graph data model is more efficient than relational data model for processing complex data. This paper proposes a new graph data storage model for the collaborative filtering-based recommendation system. Our proposed storage model efficiently filters out vertices which could not impact on calculating top-k recommended items in collaborative filtering algorithm. We present our structure, mechanisms and experimental results for improving the performance of recommender systems. For showing that proposed mechanisms are applicable in multimedia applications, we use real data set of the online site, MovieLense in the experiment. The result of the experiment shows that proposed approach is efficient storage model for recommendation system.*

***Keywords:*** *Graph database, graph data storage model, collaborative filtering, multimedia recommendation system*

## 1. Introduction

Information or multimedia overload on the Web has created enormous challenges to users for selecting multimedia contents and online businesses attempting to identity each user's preferences efficiently. Users frequently experience difficulty in searching for multimedia contents or products on the Web, while online businesses are often overwhelmed by the rich data they have collected and find it difficult to recommend multimedia contents or products to specific users. In addition, there are many specific types of recommender systems. For example, Julius T proposed disability-aware e-learning recommender systems for disabled s students [1]. So, there has been much work done both in the industry and academia on developing new approaches to recommender systems over the last decade. But there has been much few works about processing a lot of user's historical data in order to improve the performance of recommendation systems.

In CF (collaborative filtering)-based recommendation system, the rating/liking/preference behavior of users is very importantly correlated in order to recommend the favorites of one user to another, similar user [2]. Graph databases could be designed for lightning-fast access to correlating data used in recommendation systems as the graph makes it possible to intuitively represent relationship between any kinds of data [3]. However, it is recently known that graph data model has some limitations for processing large data [4]. Moreover, there are few works for resolving that limitation.

We propose an efficient graph data storage model for recommendation systems. In CF-based recommendation systems, the top-k recommendation algorithm identifies the k most similar users to an active user using similarity. After the k most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. Unfortunately, it is known that learning a user's preference and correlating them with a large database can be very time consuming and expensive [5]. Our method efficiently filters out data which could not impact on calculating top-k recommended items by using an efficient storage model based on graph data model.

## 2. Related Works and Backgrounds

Recommender systems assist users in choosing appropriate multimedia contents or products from a large set of alternatives. In such systems, personalized recommendations on items are generated by predicting preference of users. Among methods predicting preference of users, the collaborative filtering (CF) algorithm is the most known and widely used in recommender systems. Collaborative filtering (CF) methods produce user specific recommendations of items based on patterns of ratings or usage (*e.g.*, purchase). In order to establish recommendations, CF systems need to relate two fundamentally different entities; items and users [6].

CF has been very successful in both research and practice. However, Sarwar *et al*., argued that there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems in their work [7]. The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands neighbors in real-time. Further, existing algorithms have performance problems with individual users for whom the site has large amounts of information. The second challenge is to improve the quantity of the recommendations for the users. Users need recommendation they can trust to help them find products or multimedia contents they will like. In some ways, these two challenges are in conflict, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simultaneously so the solutions discovered are both useful and practical.

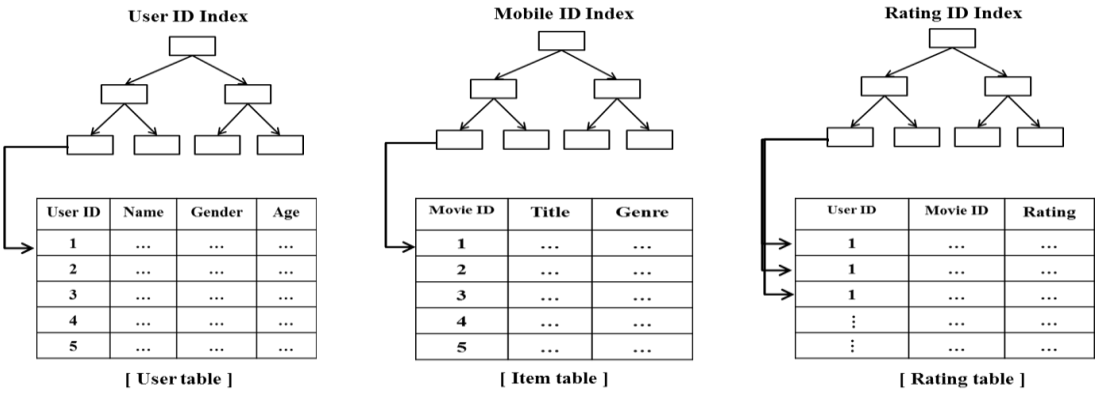


**Figure 1. A Rating Database Example including Three Tables and Tree Indexes**

Relational databases have many limitations for storing and processing complex data used in biological system, social network or recommendation system because relational databases have to represent all law data into tabular. On the contrary, graph databases represent data as

"things" (or vertices) and relationships between things. This comes much closer to these complex systems.

Most works related with the graph data model only consider the domain such as chemical informatics, bioinformatics, and pattern recognition [8-9]. Graph databases are starting to emerge as a newly way of dealing with Web data delivered in a non-relational format, such as rating data used in recommendation systems. However, there are few works about the storing and processing data for recommendation system.

From the research of the recommendation systems with the graph data model, Batul J. *et al.*, proposes benefit of recommendation systems with the graph data model [10]. However, their study did not consider any pruning techniques for reducing the search space which makes the study not suitable for a large size of the graph. Figure 1 shows a simple example where there are three tables, a user table, an item table and a rating table for CF-based recommendation, where each table has their own index. Table 1 describes computing times for the CF-based recommendation system over the example of the database shown in Figure 1 [11]. Like shown in Table 1, the highlighted operation (step 7) related with computing similarity between a user and other users is the most time-consumed factor among other operations of CF-based recommendation systems.

**Table 1. Computing Times for CF-based Recommendation System**

| Operations | Times |
|---|---|
| 1. Query the *rating* index to find items which are rated by the user $u$ in the rating table. | $O(\log_2 n)$ |
| 2. Given rows returned by the *rating* index, get *k* item IDs for these rows. | $O(k)$ |
| 3. Query the *rating* index to find other users who valuate item $k_i$ with some rating value in the rating table. | $O(k \log_2 n)$ |
| 4. Given rows returned by the rating index, get *m* user IDs for these rows. | $O(m)$ |
| 5. Query the rating index to find items which are rated by the user $m_i$. | $O(m \log_2 n)$ |
| 6. Given rows returned by the rating index, get *l* item IDs and each rating value. | $O(ml)$ |
| **7. Computing similarity between the user $u$ and $m_i$.** | $O(l^m)$ |
| 8. Pick up top *r* user IDs who have higher similarities than other users. | $O(r)$ |
| 9. Query the *rating* index to find items which are rated by the user $r_i$. | $O(r \log_2 n)$ |
| 10. Given rows returned by the rating index, get *p* item IDs for these rows. | $O(p)$ |
| 11. Query the item ID index to find all item rows $p$ in item table. | $O(p \log_2 n)$ |
| 12. Compose the recommendation list by getting $p$ titles for these rows. | $O(p)$ |

## 3. Graph Data Storage Model for CF-based Recommendation

We are mainly concerned with the rating graph which describes entities and relationships between entities. In this graph, entities are represented by vertices and relationships by edges. And all entities of the graph have their attributes. For the CF-based recommendation systems, the considered graph has two types of vertices, user and item. The graph includes a type of the edge, rated. Figure 1 shows an item-rating graph which contains two types of nodes, user and item and a type of the edge, rated.

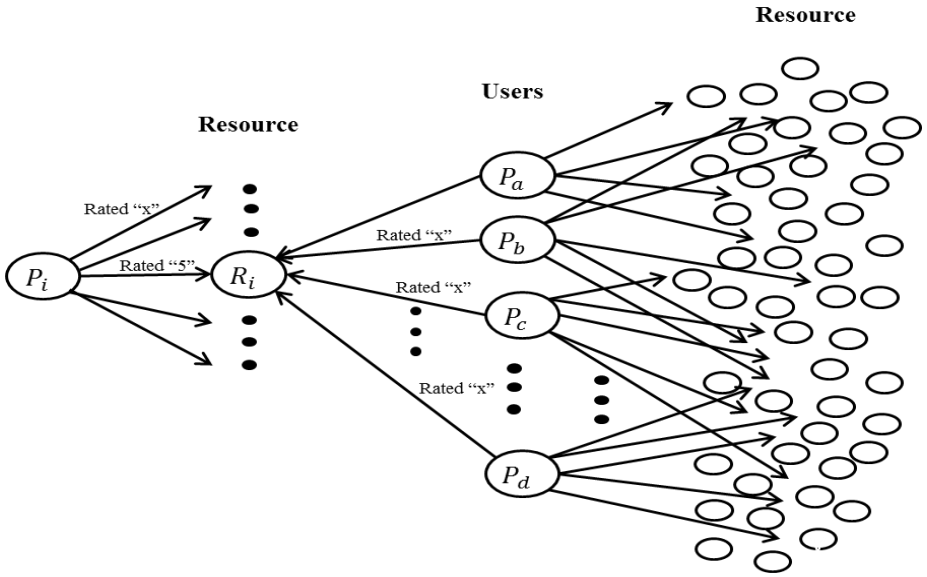


**Figure 1. Item-rating Graph**

Commonly there are two functions for founding out recommending resources with a graph database [12]. Function f traverse to all people vertices that like the same resources as person vertices i. Function g traverses to all the resources rated by person vertices j. In composition, determines all those resources that are rated by those people that have similar tastes to the person vertex i.

However, this approach has fatal limitation. If the size of the graph database is increasingly large, the cost of traversing nodes is more and more growing. For resolving the limitation of recommendation systems with graph database, we construct a candidate user index which store user vertices that have the higher possibility of being chosen as top k similar user vertices for a specific user vertex than other user vertices.

Figure 2 shows the structure of the candidate user index. The candidate user index is represented as a B-tree with candidate lists. Candidate similar user vertices for a specific user vertex are stored in a leaf node of the user index. We take note of that the high similarity between user is mainly influenced with user's common rating behavior for a resource. Based on this fact, we select the candidate similar user vertexes, for a specific user vertex, if the vertex has relatively many frequencies of valuating same items with similar values. In addition, Figure 2 shows the benefit of using the candidate user index for the process of searching candidate user vertices which have relatively high possibility of having higher similarities than other user vertices in the rating graph database.

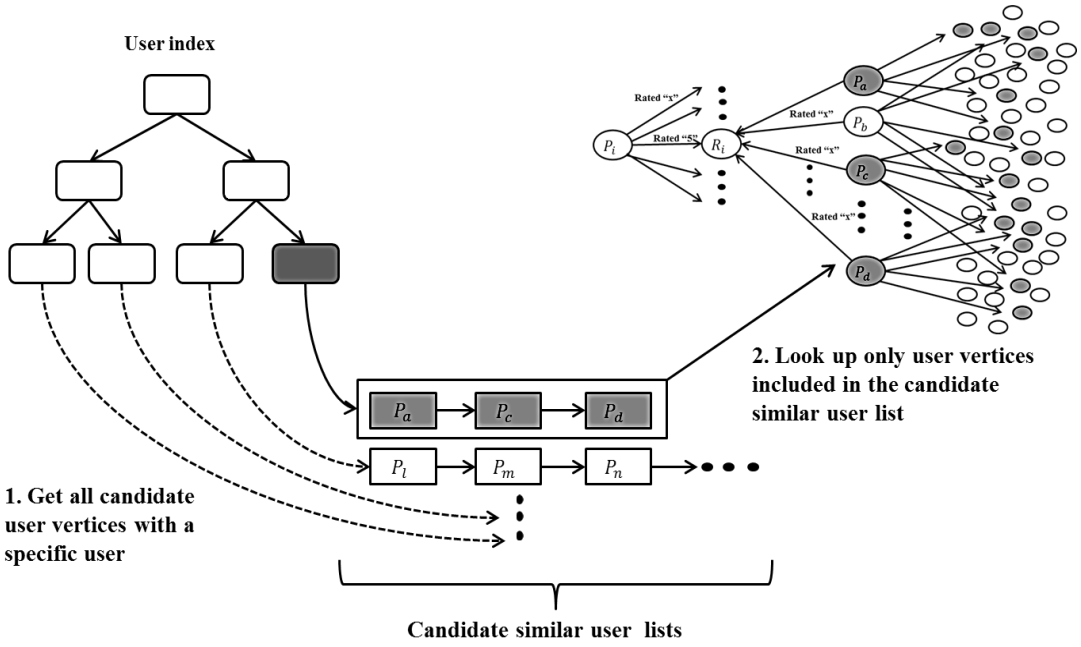


**Figure 2. The Structure of Candidate Similar user Index**

As shown in Figure 2, the recommender system with the candidate similar user index efficiently prunes the non-important vertices that have low influence on computing similarity between a specific user and other users. That is, we efficiently reduce the number of targeted users (m in step 7 described in Table 1) and the number of considered items (l in step 7 of Table 1), which are main reasons that computing similarity is time-consumed operation. Therefore, the recommender system with the candidate similar user index can more quickly find the similar users and compose the recommendation list.

**Algorithm 1.** Constructing the candidate similar user index

**Input**: Rating graph database *G*, rating threshold $\delta_r$, rating frequency threshold $\delta_f$.

**Output**: the candidate similar user vertices index *I*.

1. **for each** *user* vertex $u \in G$ **do**
2. $UR = UR \cup$ (all out *rated* edges from $u$ and values of attributes of these rated edges $V$);
3. $R = R \cup$ (tail vertices from edges $\in UR$);
4. **end do**
5. **for each** *item* vertex $r \in R$ in *G* **do**
6. $ER = ER \cup$ **(** all *rated* edges head to $r$ $\cup$ values of attributes of these *rated* edges);
7. **end do**
8. **for each** *rated* edge $e_i \in ER$ **do**
9. **if** The value of subtracting ratingvalue$v_i \in UR$ from $e_i$'s ratingvalue$v$ $\leq \delta_r$

   **then** Discard $e_i$ from *ER*;

10. **else** $FC = FC \cup$ tail vertex from $e_i$ ;
11. **end do**
12. **for each** user vertex $c_i \in FC$
13. **if** the number of $c_i$ 's appearance frequency $\geq \delta_f$
    **then** $I = I \cup c_i$ ;
14. **end do**
15. **return** *I*;

Algorithm 1 describes the construction of the candidate similar user index. It is not needed to traverse over the whole graph database to find similar user vertices for a specific user vertex. Our method only accesses a few candidate similar user vertices index.

Algorithm 2 describes the search of items which are composed in a recommendation list R with our proposed index I. The recommender system with our proposed storage model just queries the candidate similar user index I to find similar user vertices without looking up all user vertices in the graph database G. Then the recommender system computes similarity between an active user and the other users who are included in the candidate user vertices index I with the collaborative filtering function CF.

**Algorithm 2.** Searching recommended items

**Input:** graph database *G*, the candidate similar user index *I*, an active user $u_{id}$

**Output:** the recommended items *R*.

1.Get a node *n* with $u_{id}$ in user index
2.Get candidate similar user list, *CR* pointed by node *n*.
3.**for each** *user* vertex $c_i \in CR$ **do**
4. $R = R \cup CF(u_{id}, c_i)$ ;
5.**end do**
6. **return** *R*

## 4. Experiments

For the experiment, we implemented different types of recommender systems which have their own databases, relational database, RDB, general graph database, GraphDB, and the proposed graph database with the candidate similar user index respectively. The recommender system with RDBMS was implemented by MySQL, a widely used RDBMS [13]. Both recommender systems operated over the graph database implemented with Neo4j which is the most widely known graph database [14]. All recommender systems were experimented in Intel® Core™ is CPU at 3.4 GHz, and 8 GB RAM on Window 7 Enterprise K.

For the experiment data set, we use the date set publically provided by MovieLense, a movie recommendation website [15]. Based on this real dataset, we generated five rating graphs and five rating relational databases with the different number of vertices and rating edges like shown Table 2. In each rating graph, we conduct 10 experiments with 10 user IDs which are randomly selected. And we take the result by averaging 10 results of each rating

graph. All recommender systems were equally implemented with a CF-based recommendation algorithm.

**Table 2. Data Set**

| The kind of data / The number of total data | Users | Items | Ratings |
|---|---|---|---|
| 2,000 | 1,500 | 500 | 54,032 |
| 4,000 | 3000 | 1,000 | 84,020 |
| 6,000 | 4000 | 2,000 | 354,003 |
| 8,000 | 5000 | 3,000 | 754,004 |
| 9,940 | 6040 | 3,900 | 1,000,209 |

We evaluated the index sizes of each databases. Also, we respectively measure the time of each recommender system for searching and composing the recommendation list in order to evaluate the performance of query time over each type of databases. Figure 4(a) and Figure 4(b) show these experimental results.

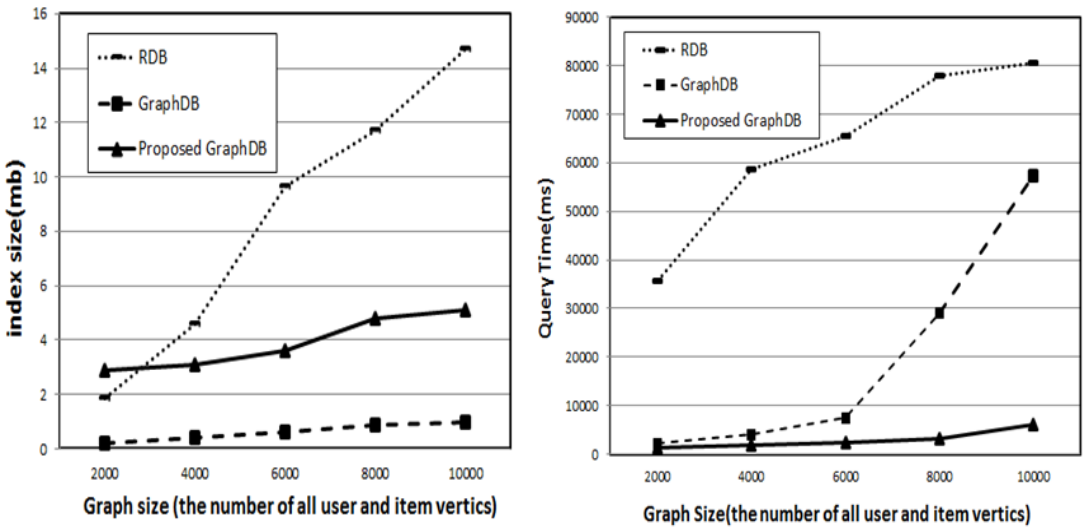


**Figure 4. Experimental Results for the Index Size (a) and Query Time (b)**

Through the result shown in Figure 4(a), we found three factors. Firstly rating databases, GraphDB and proposed GraphDB are better in storing rating data than the relational database. Secondly, GraphDB is slightly better than proposed GraphDB with the perspective of index size because proposed GraphDB additionally includes candidate user index. Finally, the number of rating data has main influence on the increment of the index size.

Actually, the index size of RDB was significantly increased by increasing the graph size. In case of the performance of querying recommended items, we found that proposed GraphDB model significantly outperforms two recommender systems with RDB and GraphDB respectively trough the result shown in Figure 4(b). With the result shown in Figure 4(b), it is very noticeable that the recommender system with proposed GraphDB was almost not influenced from the increment of the graph size while the recommender system with Graph was mainly influenced the increment of the graph size, especially in the increment of the rating edges. The main reason of this result is that the number of targeted users and

considered items in the computation of the similarity is efficiently reduced by the candidate similar user index.

## 5. Conclusions

Graph databases represent data as "things" (or vertices) and relationships between things. This comes much closer to these complex systems. However, graph databases have the fatal limitation of storing and processing a large amount of data. For storing and querying a large amount of user's online commerce data used in recommendation system, we proposed new graph data storage model in this paper. For improving the recommendation query time over large graph database, we index vertices which are considered as more important than other vertices in graph database. Through experimental results, we knew that our proposed graph data storage model is more efficient than RDBMS and existing general graph database. That is, the proposed graph data model is efficiently suitable with and specialized in storing and querying data treated by CF-based recommender systems.

## Acknowledgements

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

## References

[1] J. T. Nganji, "Designing Disability-Aware E-Learning Systems", Disabled Students' Recommendations, International Journal of Advanced Science and Technology (IJAST), vol. 48, **(2012)**, pp. 61-70.
[2] B. Sarwar, G. Karypis, J. Konstan and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms", Proc. 10th Int'l WWW Conf., **(2001)**, pp. 285-295.
[3] R. Angles and C. Gutierrez, "Survey of graph database models", ACM Comput. Surv., vol. 40, no. 1, **(2008)**, pp. 1-39.
[4] J. Han, "Survey on NoSQL database, Pervasive Computing and Applications (ICPCA)", 6th International Conference, **(2011)**, pp. 364-366.
[5] J. Ben Schafer, J. A. Konstan and J. Riedl, "E-Commerce Recommendation Applications", Data Mining and Knowledge Discovery, vol. 5, **(2001)**, pp. 115-153.
[6] Collaborative filtering, Wikipedia, http://en.wikipedia.org/wiki/Collaborative_filtering.
[7] B. M. Sarwar, G. Karypis, J. A. Konstan and J. Riedl, "Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering", Proceeding of The Fifth International Conference on Computer and Information Technology (ICCIT), **(2002)**.
[8] B. T. Messmer and H. Bunke, "A Decision Tree Approach to Graph and Subgraph Isomorphism Detection", Pattern Recognition, vol. 32, **(1999)**, pp. 1979-1998.
[9] R. Giugno and D. Shasha, "GraphGrep: A fast and universal method for querying graphs", Pattern Recognition, 2002. Proceedings 16th International Conference, **(2002)**, pp. 112-115.
[10] B. J. Mirza, B. J. Keller and N. Ramakrishnan, "Studying Recommendation Algorithms by Graph Analysis", Journal of Intelligent Information Systems, vol. 20, no. 2, **(2003)**, pp. 131-160.
[11] M. A. Rodriguez and P. Neubauer, "The Graph Traversal Pattern", chapter in Graph Data Management: Techniques and Applications, **(2011)**, pp. 1-18.
[12] Z. Huang, W. Chung, T.-H. Ong and H. Chen, "A graph-based recommender system for digital library", Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries, **(2002)**, pp. 65-73.
[13] MySQL, http://www.mysql.com/.
[14] Neo4j, http://www.neo4j.org/.
[15] MovieLens, http://movielens.umn.edu/login.

# Authors

**Haesung Lee** is a Ph.D. candidate of Computer Science at Kyonggi University, Korea. Her research areas include Context-aware Computing, Social Network, Information Retrieval and Mobile Computing. She works for software development on areas of data search in ubiquitous environment. She received her B.S., M.S. in Computer Science from Kyonggi University,Korea. Contact her at seastar0202@ kyonggi.ac.kr.

**Joon Hee Kwon** is an associate professor of Computer Science at Kyonggi University, Korea. She was a visiting research professor at the Computer Science Department at New Jersey Institute of Technology. Her research areas include Context-aware Computing, Information Retrieval, Social Network, Web 2.0 and Mobile Database. Her research projects focus on areas of d1ata search using social network and Web 2.0 in ubiquitous environment. She received her B.S., M.S. and Ph.D. in Computer Science from Sookmyung Women's University, Korea. Contact her at kwonjh@kyonggi.ac.kr.