Keyword-based Search on Relational Data

Myint Myint Thein
University of Computer Studies
Mandalay, Myanmar
mmyintt@gmail.com

ABSTRACT

Keyword-based search in relational database is an easy and effective way for ordinary users or Web users to access relational databases. Even though the relational database management systems (RDBMs) have provided full-text search capabilities, they do not support keyword-based search model. The text databases and relational databases are different that is challenging task to apply the keyword search techniques in information retrieval (IR) to DB. Much research on keyword-search in relational databases has been developed. But existing researches are still problem which is a growing dissatisfaction among users searching for information that they are actually desirable. In this paper, we propose a new candidate network generation algorithm (Heuristic_CNGen) based on the Iterative Deepening A(IDA*) algorithm. We also propose a new ranking method by adapting existing IR scoring techniques based on the virtual document. We present the keyword-based search to retrieve relevant queries in relational databases by free-style keyword query that improves the efficiency and effectiveness.*

Keywords- *Keyword-based; Relational Database; Candidate Network; IR;*

1. INTRODUCTION

Keyword-based search in relational databases enables ordinary users, who do not understand the underneath schema and SQL, to find relevant results among the tuples stored in relations, with a given set of keywords. In traditional search model in relational databases, users need to have knowledge of the database schema and to use SQL. Even though RDBMs have provided full-text search capabilities, they do not support keyword search model [4, 5]. The database research community has recently recognized the benefits of keyword search and has been introducing keyword search capabilities into relational databases. The existing methods of keyword search in relational databases can be broadly classified into two categories that are schema based method and graph based method [1].

In schema based keyword search in relational database, there has two steps for retrieving answers by processing with the user typed keyword. First, generating all valid candidate networks are called connected tuples tree by joining tuples from multiple tables. A candidate network must satisfy the two conditions, total and minimal. Because it is meaningless if two tuples in a connected tuple tree are too far away from each other, the maximum numbers of tuples allowed in a connected tuple tree are needed to specify. Existing candidate network generation, CN's size is unbounded and the number of CNs grows very large for small CN's size. This fact brings large overhead for both computation and memory cost. Second, computing a single score for each CN and then combining them together to get the final score that the most relevant answers are ranked as high as possible. A ranking method is essential for getting user satisfaction. Some of existing ranking methods may even lead to search results contradictory to user perception.

Consider a DBLP [11] database maintains publication records in several relations in a relational database. Suppose a user wants to search papers written by "Jinlin Chen" with "Web and Content" on a publication database. A user has typed in a query "chen web and content", and the system will return the relevant answers possibly with multiple tuples from different tables that are joined together to form a meaningful result to the query.

In this paper, we study how to search structural information among tuples in a relational database using keyword-based. We develop the new methods to address these challenges. We fist propose a candidate network generation algorithm to find relevant answers on-the-fly by joining tuples in the database. We propose a new ranking method by adapting the IR ranking methods based on the virtual document that is answered the effective relevant documents from relational database for the user. The proposed methods support efficient and effective keyword-based search on large amounts of relational data.

The rest of the paper is organized as follows: Section 2 discusses the related works. Section 3 presents the basic concept of keyword query and architecture of our system. Section 4 and 5 present the Candidate Network Generation and

Query execution, respectively. Section 6 shows the experimental result. Section 7 concludes this paper.

2. RELATED WORK

The main goal of a keyword search system is to find a set of closely inter-connected tuples that collectively match the keywords. One type of methods is based on modeling data as a graph, and the results as subtrees or sub-graphs. Another type of methods is based on relational databases where structured data are stored.

Several researchers have been done on early keyword search systems for relational databases [2, 7, 8]. Yu et al. [4] surveyed the developments on finding structural information among tuples in an RDB using an l-keyword query. They discussed the keyword search systems by comparing between schema-based keyword search and graph-based keyword search in RDB. The former evaluated the sets of answers by defining all minimal total joining networks of tuples between CNs and the latter showed how to answer keyword queries using graph algorithms focused on weighted directed graph.

DISCOVER [7] proposed the CN generation algorithm based on a breadth-first traversal in the search space. This proposed algorithm expanded the partial CNs generated to larger partial CNs until all CNs are generated. The problem with this algorithm is that the cost of generating the set of CNs is high and kept in memory for further extension. IR-Style [8] proposed IR-style ranking method to rank tuple trees. This method had not considered the effectiveness of the query results. S-KWS [2] developed an algorithm that reduces the number of partial results generated by expanding from part of the nodes in a partial tree and avoid isomorphism testing by assigning a proper expansion order. Although it reduced the generated partial results, it existed overhead for generating minimal CNs to the query. In contrast, we propose the CN generation algorithm to apply heuristic value for generating minimal CNs.

Liu et al. [3] described the answer graph generation algorithm to generate tuple trees and ranking formula by adapting four normalizations to address the retrieval effectiveness issue. Although they produced duplication-free CNs by assigning the different alias, they had not considered the efficiency of answer generation. SPARK2 [9] developed the duplication-free algorithm by canonical form but it did not solve the number of CNs grows very large for small CN size. They modified the IR ranking method based on the virtual document. Their method produced repeated information which concerns overlapping among the top-k JTTs. In this paper, we propose a new ranking method to reduce the meaningless results which are disappointed for user.

3. PRELIMINARIES

3.1. Query Representation

A relational database can be viewed as a graph which represents a relational model such as schema graph G_s (V, E) [4, 6, 10]. A relational database is a collection of relations. Each relation in the database corresponds to a vertex in G_s, denoted as the set of relation schemas {R1,R2, …}. Edges represent the foreign key to primary key relationships between pairs of relation schemas, R_i and R_j, denoted $R_i \rightarrow R_j$. A relation on relation schema R_i is an instance of the relation schema, such as a set of tuples, conforming to the relation schema.

We use directed schema graph that shows in Figure 1. as the schema graph of publication database. It consists of six relation schemas: Person, Inproceeding, Proceeding, Publisher, Series and Relation-Person-Inproceeding. Each relation has a primary key (PK). Inproceeding has one foreign key that refers to the primary key defined on Proceeding. Proceeding has two foreign key that refer to the primary key defined on Publisher and Series. For simplicity, we assume all primary key and foreign key attributes are made of

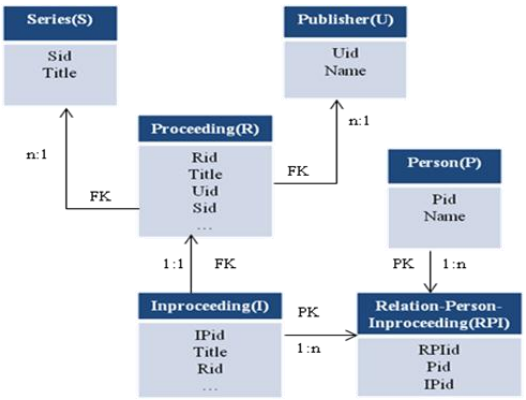

Figure 1. Publication Database Schema Graph

same attribute with attribute of related relation. There are no self loops and at most one foreign key to primary key relationship between any two relations.

A keyword query (Q) consists of a list of keywords $\{k_1,k_{2,\ldots,}\ k_q\}$, and searches interconnected tuples that contain the given keywords. For a given query Q, a result is the set of all possible joining networks of tuples. A joining network of tuple is a connected tuple tree (T). Each node t_i is a tuple in the database, and each pair of adjacent tuples in T is connected via a foreign key to primary key relationship. Suppose (R_i,R_j) is an edge in the schema graph. Let $t_i \in R_i$, $t_j \in R_j$, and $(t_i \text{ join } t_j) \in (R_i \text{ join } R_j)$. Then (t_i,t_j) is an edge in the connected tuple tree T. The size of a connected tuple tree is the number of tuples involved. Note that a single tuple is the simplest tuple tree with size 1. For simplicity, we use I, R, U, P, S and RPI to denote the relations Inproceeding, Proceeding, Publisher, Person, Series and Relation-Person-Inproceeding respectively.

3.2. Keyword-based Search Architecture

In this section, we demonstrate the architecture of keyword-based search on relational data that is shown in Figure 2. The system supports free-style keyword search by computing answers to keyword queries with user typed keywords. The query cleaning phase filters out as potential index terms

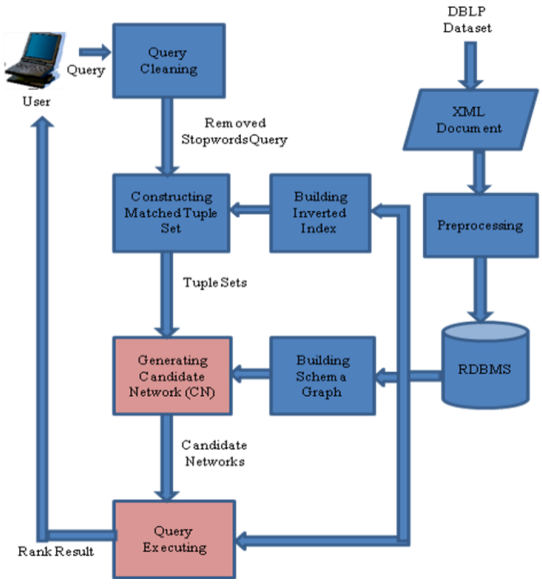

Figure 2. Architecture of Keyword-based Search on Relational Data

as removed stopwords query. This process reduces the size of the indexing structure considerably. The indexing unit in a relational document can be a field, attribute, tuple, table, or any combination of these. After the system has built the inverted index files as posting table for each relation, the indexer produces the matched tuple sets by using the filtered input query. The system generates a set of CNs by traversing on the schema graph in order to the tuple sets. Query executing phase executes queries for each CNs and ranks the executed queries on the virtual document. Finally, the system returns the ranking relevant results to the user for a given query.

4. CANDIDATE NETWORK GENERATION

In this section, we introduce the existing candidate network strategies by motivating our work. Then, we propose a new CN generation algorithm for schema-based keyword search in relational database.

4.2. Identifying Connected Tuple Tree As Result

For a given query Q, the connected tuple tree is generated according to a CN that is some tuples coming from different relations. For each pair of adjacent tuple sets R_i, R_j in connected tuple tree, there is an edge (R_i, R_j) in the directed graph G. Each connected tuple tree that defined satisfaction as follow:

- Property 1: If a node in connected tuple tree is one of tuples in relation, it contains at least one keyword in query Q (completeness).
- Property 2: there is no duplicate tuple with each other in the connected tuple tree (duplication-free).

Note that if a node has two or more edges that may or may not contain any keyword. This fact implies that (1) if a result contains multiple tuples, they must be joined together as a tree, and (2) if we remove any node in the connected tuple tree that has a tuple with no keywords, there is no redundancy.

The Connected Tuple Tree 1 for Query 1 and Connected Tuple Tree 2 for Query 2 show in Figure 3., such as examples. In Connected Tuple Tree 1, a node P1 contains the keyword "Chen", and I1 contains two keywords "Web" and "Content", and U1 contains the keyword "Springer" in Query 1. In Connected Tuple Tree 2, the nodes P2 and I3 contain the keywords "Yui" and "Web", and U3 contains the two keywords "Erlbaum" and "Lawrence" in Query 2 respectively. Except primary-foreign relation nodes, all remaining nodes contain the keywords in given query, and there are no duplicate nodes. In this paper, we consider a connected tuple tree as a result as long as it fulfills the properties.

4.3. Generating Candidate Networks As Result

In this section, we describe generating the connected tuple trees as result in detail. Given a keyword query Q, the system first receives all the query tuple set R^Q for all relations R as input. Then it focus on generating all the valid CNs which are joined expressions to be used to create connected trees of tuples that will be considered as potential results to the query. For example, the non-free query tuple set R^N of relation Person for Query 1and Query 2 are P^{Q1}={P1,P2,P3,P4} and P^{Q2}={P2} respectively. The free query tuple set R^F of relation Person for Query 2 is P^{Q2}={P1,P3,P4}.

Query 1: "Chen Web Content Springer"

Connected
Tuple Tree 1: P1→RPI1←I1→R1→U1

CN1: $P^N \bowtie RPI^F \bowtie I^N \bowtie R^F \bowtie U^N$

Query 2: "Yui Web Erlbaum Lawrence"

Connected
Tuple Tree 2: P2→RPI2←I3→R2→U3

CN2: $P^N \bowtie RPI^F \bowtie I^N \bowtie R^F \bowtie U^N$

Figure 3. Queries, Connected Tuples Trees and Candidate Networks

We use R^{NorQ} to define a tuple set, if CN is a result then each node belongs to the non-free query tuple set R^N and the free query tuple set R^F of each relation R for a given query. Note that the free query tuple set in CN cannot contain the query keyword, but they support to the non-free query tuple set as primary-foreign keys relationship. We identify a network graph as a joined expression of the query tuple sets that produces connected tuple trees as result. We define the size of a network graph as the number of nodes the same as the generated connected tuple tree's size. We apply the candidate network generation algorithm based on IDA* to generate all network graphs for a given query Q and schema graph SG. In the proposed algorithm, we set up three parameters: MAXN, f_limit and f_new. First, the maximum number of tuple sets, denote MAXN, in a network graph to reduce generating meaningless results. Second, the node the cheapest solution through node R_j^N is less than given f_limit value. If the estimated cost of node R_j^N add in front of queue E, if the estimated cost of R_j^N is more than f_limit

value, the node R_i^N that is adjacent node R_j^N in SG add in front of E. Third, f_new assign heuristic value of a new node that is adjacent by the existing node in schema graph. Finally, the number of CNs is only data bounded by the query and database. And it produces connected tuple trees as results by evaluating the corresponding joined expressions.

The following Properties 3 and 4 prove the completeness and duplication-free on the results of the algorithm, if we do not violate any constraints.

- Property 3: The set contains all CNs with no more than MAXN (completeness).
- Property 4: Every two CNs are not isomorphic to each other. (duplication-free).

5. QUERY EXECUTION

In this section, we propose a new ranking strategy for effective keyword search in relational database. Then, we discuss how to use it to improve the ranking strategy.

5.1. Modified IR Ranking Score

To rank documents, IR systems assign a score for each document as an estimation of the document relevance to the given query. In IR, a document is a basic information unit stored in a text database. It is also the basic unit of answers needed by users. A similarity value between a given query and a document is computed to rank documents. In relational keyword search, the basic text information unit stored in a relational database is a text column value. The basic unit of answers needed by users is a connected tuple tree which is assembled by joining multiple tuples, each of which may contain zero, one or multiple text column values. A similarity value between a given query and a connected tuple tree needs to be computed to rank connected tuple trees. We propose a solution based on the idea of modeling a connected tuple tree as a virtual document. Consequently, the entire results produced by a CN will be modeled as a document collection. By adopting such a model, we assign an IR ranking score to a connected tuple tree as:

$$score_a(k, D) = \frac{ntf}{ndl} * \ln(idf) \quad (1)$$

$$ndl = (1 - s) + s * \frac{dl_{CN}}{avgdl(CN)} \quad (1.1)$$

$$ntf = 1 + ln(1 + \ln(tf_k(CN))) \quad (1.2)$$

$$tf_k(CN) = \sum_{k \in D} \frac{kf_i}{\max\{kf_1, \dots, kf_i\}} \quad (1.2.1)$$

$$idf = \sum_{k \in D} \frac{N(CN) + 1}{df_k(CN)} \quad (1.3)$$

, where ntf indicates the normalized trem frequency, ndl is the normalized document length, idf is the inverse document frequency and tf_k (CN) denotes the number of occurrences of the CN which belongs to the connected tuple tree such as document.

5.2. Tuple Size Normalization Factor

We evaluate a score value for the size of CN and the size of the given query, especially for a complex query whose relevant results are connected tuple tree involving multiple tuples, each of which contains a subset of the keywords query. We believe that the users usually prefer documents matching many keywords query to those matching only few keywords. To approximately the user perception, we define the tuple size normalization factor for a query.

$$score_b(k, D) = \frac{\ln(size(CN))}{size(Q)} \quad (2)$$

Finally, the relevance score of a connected tuple tree to a keyword query is computed as:

$$score(T, Q) = score_a(k, D) * score_b(k, D) \quad (3)$$

6. EXPERIMENTAL RESULTS

We evaluate our proposed algorithm on DBLP dataset. It consists of a set of XML entries with each entry representing a single publication. We decomposed into relations from a downloaded

XML file according to the schema that is shown in Figure 1. For evaluation, we manually picked a large number of queries on this data. We focus on 10 queries with query length ranging from 2 to 6. All queries generating algorithm was implemented in Java, and JDBC was used to connect to the database.

In this paper, we do not evaluate the performance of the proposed ranking method. We focused on the performance of the proposed Heuristic_CNGen algorithm. We compare the evaluation results of the native algorithm and the proposed algorithm by using the same DBLP dataset that is shown in Figure 4. We observe that proposed algorithm achieve better search performance than the existing native methods.

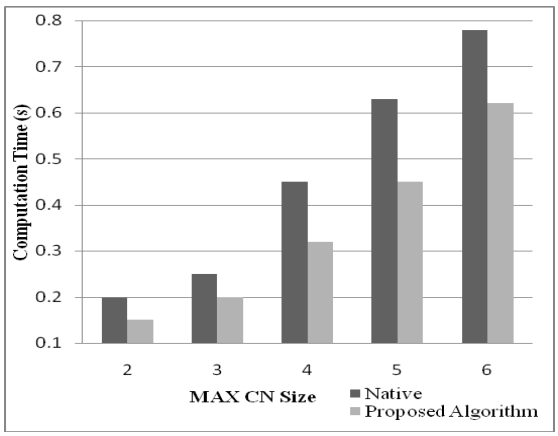

Figure 4. Comparison of Native Algorithm and Proposed Algorithm

7. CONCLUSION

Keyword search allows ordinary users to find text information in relational databases with much higher flexibility. In this paper, we present the keyword-based search to retrieve relevant queries in relational databases by free-style keyword query. A keyword query in the system is a list of keywords and does not need to specify any relation or attributes names. The result to such a query consists of the minimal connected tuple trees, which potentially include tuples from multiple relations by joining tuples in database. To produce connected tuple trees, we presented a new candidate network generation algorithm (Heuristic_CNGen) by generating all the valid CNs which are joined expressions. The proposed method can solve the growing CNs for small CN size with the generating candidate network methods in previous works. We also proposed a new ranking method by adapting the IR ranking techniques based on the virtual document to rank the connected tuple trees. The proposed methods retrieve the relevant results approximately for a given query.

REFERENCES

[1] A.Baid, I.Rae, J.Li, A.Doan, J.Naughton, "Toward Scalable Keyword Search over Relational Data," Proc. VLDB End., Vol. 3, 2010.

[2] A.Markowetz, Y.Yang, D.Papadias, "Keyword Search on Relational Data Streams," Proc. 2007 ACM SIGMOD Int. conference on Management of data, 2007, pp. 605-616.

[3] F.Liu, C.Yu, W.Meng, "Effective Keyword Search in Relational Databases," Proc. 2006 ACM SIGMOD Int. conference on Management of data, 2006, pp. 563-574.

[4] J.X.YU, L.Qin, L.Chang, "Keyword Search in Relational Databases: A Survey," IEEE Data Engineering Bulletin, Vol. 33, 2010, pp. 67-78.

[5] K.Stefanidis, M.Drosou, E.Pitoura, "PerK: Personalized Keyword Search in Relational Databases through Preferences," Proc. 13th Int. Conference on Extending Database Technology, EDBT, 2010, pp. 585-596.

[6] L.Qin, J.X.Yu, L.Chang, "Keyword Search in Databases: The Power of RDBMs," Proc. 35th SIGMOD Int. Conference on Management of data, 2009, pp. 681-694.

[7] V.Hristidis, Y.Papakonstaninou, "DISCOVER: Keyword Search in Relational Databases," Proc. 28th Int. Conference on Very Large Data Bases, 2002, pp. 670-681.

[8] V.Hristidis, L.Gravano, Y.Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," Proc. 29th Int. Conference on Very Large Data Bases, 2003, pp. 850-861.

[9] Y.Luo, W.Wang, X.Lin, X.Zhou, "SPARK2: Top-k Keyword Query in Relational Databases," TKDE Special Issue: Keyword Search on Structured Data, 2011.

[10] Y.Xu, Y.Ishikawa, J.Guan, "Effective Top-k Keyword Search in Relational Databases Considering Query Semantics," APWeb/WAIM Int. Workshops, 2009, pp. 172–184.

[11] http://www.dblp.uni.trier.de.