Querying Connected Tuple Trees for Relational Keyword Search

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 database. Even though 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 a challenging task to apply the keyword search techniques in information retrieval (IR) to DB. A common method to performing keyword search in relational database is to generate the minimum connected tuple sets in schema graph transformed from relations. Although existing candidate network (CN) generation methods retrieve a set of joining tuples, they are still problem which is causing large overhead for CNs generation. In this paper, we propose a new candidate network generation algorithm (Heuristic_CNGen) based on the iterative deepening A* (IDA*) algorithm. The proposed algorithm produces a minimum number of CNs according to the maximum number of tuple set. We generate CNs for a given keyword query. And then, we identify the connected tuple tree as a result according to generated CNs. We evaluate the proposed method on DBLP.**

Keywords-Keyword-based; Relational Database; Candidate Network; Connected Tuple Tree;

I. INTRODUCTION

Keyword-based search in relational database enables ordinary users, who do not understand the underneath schema and SQL, to find the connected tuple sets among the tuples stored in relations, with a given set of keywords. In traditional search model in relational database, 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. The database research community has recently recognized the benefits of keyword search and has been introducing keyword search capabilities into relational database. The existing methods of keyword search in relational database can be broadly classified into two categories that are schema based method and graph based method.

In schema based keyword search in relational database, it has a common method that is generating the candidate network in schema graph transformed from relations. Existing schema-based keyword search methods exploit the breadth-first search algorithm to generate the CNs for the keyword query [1, 3, 8, 14]. The breadth-first search algorithm expands all nodes of the schema graph at once. Since all the nodes at each state are stored in order to generate the nodes at the next state, the search spaces of breadth-first search are exhausted the available memory long before an appreciable amount of time is used. This space requirement of breadth-first search algorithm is its most critical drawback. IDA* is an optimal heuristic search algorithm for the memory requirement [13, 15]. It is optimal in terms of memory usage and time complexity.

In relational database, data is stored in the form of columns, tables and primary key to foreign key relationships. For a given keyword query, the logical unit of answers needed by users is not limited to an individual column value or ever an individual tuple. It may be multiple tuples joined together. Given keyword search in relational database, generating minimum connected tuples sets of relations that contained keyword is called candidate network, such as SQL. 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. Consider a DBLP database maintains publication records in several relations in a relational database. If user wants to get the papers written by "Marc Shapiro", the system generates the relevant CNs, such as Person ⋈ Relation-Person-Inproceeding ⋈ Inproceeding, with multiple tuples from different relations joined by foreign keys. Generating all valid candidate networks that are called connected tuple trees by joining tuples from multiple relations.

Recently, keyword search on relational database has been developed. DISCOVER [8], S-KWS [2], Liu et al. [3], and SPARK2 [10] are systems that support keyword search on relational database. They generated tuple trees as answer for the CN generation. The first two systems need to reduce the cost of generating minimal CNs, while the last two systems cannot solve the growing number of CNs for small CN size. 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 CNs generation.

In this paper, we focus on generating the valid CNs on the data bond. We develop a new approach to generate all minimum connected trees of tuples in the database with no more than the maximum number of tuple set. We propose a candidate network generation algorithm based on IDA* algorithm to find relevant answers on-the-fly by joining tuples in the database. We conduct the experimental results on DBLP database.

The rest of the paper is organized as follows: Section II discusses the related work. Section III presents the basic concept of keyword query and CN. Section IV presents the Candidate Network Generation. Section V shows the experimental results. Section VI concludes this paper.

II. 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 database where structured data are stored.

Several researchers have been done on early keyword search systems for relational databases [2, 8, 9, 10]. 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. DBXplorer [9] used undirected graph to construct each SQL statements according to each tuple tree. This system accessed to symbol table to get tuples' information, and then calculated tuple tree according to schema graph.

DISCOVER [8] 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. As the number of partial CNs can be exponentially large, arbitrarily expanding will make the algorithm extremely inefficient. The problem with this algorithm is that the cost of generating the set of CNs is high and kept in memory for further extension.

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.

Liu et al. [3] described the answer graph generation algorithm to generate tuple trees. Although they produced duplication-free CNs by assigning the different alias, they had not considered the efficiency of answer generation. SPARK2 [10] developed the duplication-free algorithm by canonical form but it did not solve the number of CNs grows very large for small CN size. In contrast, we propose the CN generation algorithm to apply heuristic value for generating minimal CNs. We can reduce the generated partial results by estimating the cost of adjacent nodes in schema graph.

III. PRELIMINARIES

A. *Query Respresentation*

A relational database can be viewed as a graph which represents a relational model such as schema graph G_s (V, E) [5, 6, 7, 12]. 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. The graph can be as a directed or undirected graph. It can be captured every granularity level of the schema elements.

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 (FK) 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 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.

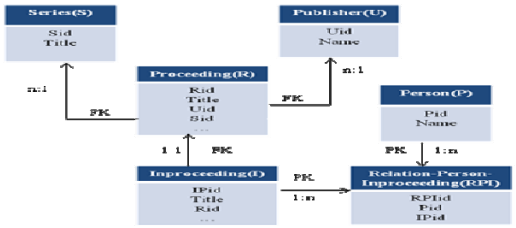

Figure 1. Publication Database Schema Graph

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_iR_j) is an edge in the schema graph. Let $t_i \in R_i$, $t_j \in R_j$, and (t_i join t_j) $\in$ (R_i 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.

Consider the query "Chen Web" as an example. For this query, some example of the connected tuples trees include: P1→RPI4←I2, P1→RPI1←I1, P2→RPI2←I3 and P3→RPI3←I4. Note that P3→RPI3←I4 is not a valid result tree to the query, as the leaf node I4 does not contribute to a match to the query. A possible answer for this query may be: P1→RPI4←I2, P1→RPI1←I1, and P2→RPI2←I3. When there exists a many to many relationship in the schema graph, the size of the connected tuple tree for a given Query 1 and Query 2 in Section IV can have arbitrarily large size in the general case. There is a foreign key to primary key relationship from relation Inproceeding to Proceeding, and from Proceeding to Publisher. We see that the size of Connected Tuple Tree is only data bound in Figure 2.

B. *Candidate Network*

Each connected tuple tree is the sets consisting of relational names that produced by a relational algebra expression, if each tuple in one relation contains a term of the keywords. For a given keyword query Q, the query tuple set R^N is a set of all tuples which belongs to relation R that contain at least one keyword of the query Q. We denote R^F the free tuple set which is the set of all tuples in relation R and we use R^Q to denote a tuple set, which can be either a non-free tuple set or a free tuple set.

A candidate network (CN) is a tree of tuple sets R^Nor R^F with the restriction that every node must be a query tuple set. Every edge (R_i^Q,R_j^Q) in a CN corresponds to an

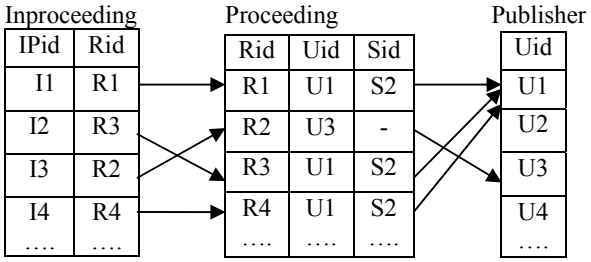

Figure 2. Data Bound of Size of Connected Tuple Tree

edge (R_i,R_j) in the schema graph G_s. A CN can be easily transformed into its equivalent SQL statement that joins a sequence of relations with selections of tuples for keywords over the relations involved. The size of a CN is the number of its tuple sets.

A set of CNs is generated over a graph schema G_s that shall be completed and duplication-free [4]. The former ensures that all minimal total connected trees of tuples are found, and the latter is mainly for efficiency consideration. In above the presented example for query "Chen Web", some example of the connected tuples trees include: P1→RPI4←I2, P1→RPI1←I1, P2→RPI2←I3 and P3→RPI3←I4. The first three connected tuple trees belong to CN: $P^N \bowtie RPI^N \bowtie I^N$ that is a valid CN according to complete and duplication-free. The last connected tuple tree belongs to CN: P^N that is not a valid CN according to non-complete, if a leaf node I^F is removed that does not contain a term of given keywords.

IV. 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.

A. Problem with Existing Candidate Network Generation

The candidate network generation generates a set of CNs over schema graph G_s, denoted as $C = \{c_1,c_2,\ldots\}$. In a CN, c_i corresponds to a relational algebra that joins a sequence of relations with selections of tuples for keywords over the relations involved. The set of CNs shall be sound or complete and duplicate-free upon the maximal size.

In schema-based keyword search in relational database, the generating all candidate networks for keyword query Q satisfy the two conditions: complete and duplication-free. First, all minimal total joining network of tuples for query Q will be produced by the set of CNs generated according to the completeness. Second, every two CNs are not isomorphic to each other due to the duplication-free.

Existing keyword-search systems, such as DISCOVER and S-KWS, generate CNs temporarily through a breadth-first traversal of schema graph for any user query. The result of CNs set is to avoid the generation of redundant joining networks of tuple sets. As the number of query keywords or maximum size of CN increases, or the database schema becomes complicated, it will take much more time to generate CNs for a query Q. There can be two ways to reduce the time for the generation of CNs. One is to develop a more efficient CN generation algorithm and the other is to develop preprocessing techniques to generate CNs in advance. In this section, we develop an efficient CN generation algorithm to address above the problem.

B. Identifying Connected Tuple Trees As Results

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 follows:

- 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 4., 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.

C. Generating Candidate Networks As Results

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 1 and 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}.

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.

```
Algorithm:Heuristic_CNGen(MAXN, f_limit, f_new)
Input: schema graph SG, query Q
Output: a set of candidate networks CN
1. E: queue generated all non-free tuple sets{R1^N,R2^N,
…,Rn^N}and all free tuple sets{R1^F,R2^F,…,Rn^F } for Q.
2. While E is not empty{
3.   Pop head T from E
4.   If  T > MAXN then T is pruned
5.   Else{
6.     If T is a valid network graph then add T to CN
7.       For each Ri^NorQ in T
8.         For each Rj^N that is adjacent to Ri^N in SG{
9.                 f_new = h(Rj^N)
10.                g(Rj^N) = c(Rj^N , Ri^N)
11.                f(Rj^N) =  g(Rj^N) + f_new
12.                If f(Rj^N) ≤ f_limit then
13.                   Add Rj^N in front of  E
14.                Else add Ri^N  in front of E
15.                  f_new = f(Rj^N)
16.                  f_limit = MAX_VALUE }
           }
         }
17. For each network graph in CN, if there is more
than one network graph in CN, then the same
network graph is pruned.
18. Return CN.
```

Fig.ure 3. Proposed Candidate Network Generation Algorithm

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 4. Queries, Connected Tuples Trees and Candidate Networks

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. For example, Connected Tuple Tree 1 and Connected Tuple Tree 2 that generate CN1 and CN2 are shown in Figure 4. We generate CN with the CN generation algorithm by tracing DBLP dataset, as is shown in TABLE I. In Fig. 3, we present the candidate network generation algorithm based on IDA* to generate all network graphs for a given query Q and schema graph SG.

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 R_j^N add in front of queue E, if the estimated cost of the cheapest solution through node R_j^N is less than given f_limit value. If the estimated cost of node 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).

TABLE I. GENERATING CN FOR Query "Yui Web"

ID	Size	CN	Valid?
CN1	1	$Person^N$	Y
CN2	1	$Inproceeding^N$	Y
	2	$Person^N \bowtie$Relation-Person-$Inproceeding^F$	N
	2	$Inproceeding^N \bowtie$ Relation-Person-$Inproceeding^F$	N
CN3	3	$Person^N \bowtie$Relation-Person-$Inproceeding^F \bowtie Inproceeding^N$	Y
	2	$Inproceeding^N \bowtie Proceeding^F$	N
	3	$Inproceeding^N \bowtie$P roceeding$^F \bowtie Publisher^F$	N
	.	.	.
	5	.	.

V. EXPERIMENTAL RESULTS

In this section, we evaluate our proposed algorithm on DBLP dataset. All queries generating algorithm was implemented in Java, and JDBC was used to connect to the database.

DataSet

Database: We use the Original Digital Bibliography and Library Project (DBLP) dataset [11] in our evaluation. 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. The size of the XML file is 173MB. TABLE II. shows the statistics after the decomposition.

Query Set: We manually picked a large number of queries for evaluation. We attempted to include a wide variety of keywords and their combinations in the query sets, such as the selectivity of keywords, the size of the most relevant answers, the number of potential relevant answers, etc. We focus on a subset of the queries in this experiment. There are 15 queries with query length ranging from 2 to 6.

A. Evaluation o f the Candidte Network Generation

We compare the evaluation results of the native algorithm and the proposed algorithm by using the same DBLP dataset that is shown in Figure 5. We observe that proposed algorithm achieve better search performance than the existing native methods. The native algorithm cannot prune on the duplication records or node. And it

TABLE II. STATISTICS OF DBLP DATASET

Relation Schema	#Tuples
Person(Pid,Name)	174,709
Inproceeding(IPid,Title,Pages,Rid)	212,273
Proceeding(Rid,Title,Uid,Sid,…)	3,007
Publisher(Uid,Name)	86
Series(Sid,Title)	24
Relation-Person-Inproceeding(RPIid,Pid, IPid)	491,777

expands the only additional state during each iteration. The proposed algorithm can generate the valid CNs by the maximal CN size. Thus, it does not expand the additional state. If it does not correctly evaluate f-limit value, it can also expand the additional state. The proposed algorithm can produce the number of CNs by duplication-free. Then, the new generated CN is compared with all existing CNs to eliminate redundancies. The proposed method reduce the number of CNs grows very large even for small CN size. We can see that time complexity of the proposed algorithm is exponentially smaller than the native.

Fig. 6 illustrates the number of Connected Tuple Trees by different maximum CNs size. There is no significant generation of Connected Tuple Trees above the maximum CN sizes 2, 3 and 4, whereas the number of Connected Tuple Trees has been significantly increased in the maximum CN sizes 5 and 6. In the experiment, the primary-foreign keys relationship can be increased with the increased CN size. So, the increased relationship can be affected on the generated Connected Tuple Trees.

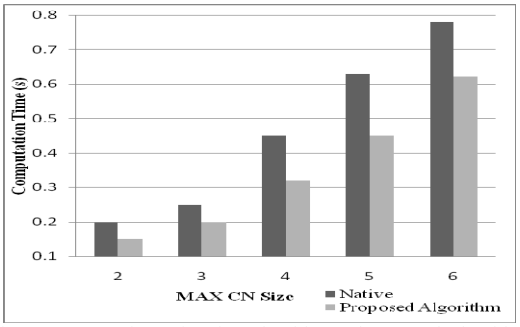

Figure 5. Comparison of Native Algorithm and Proposed Algorithm

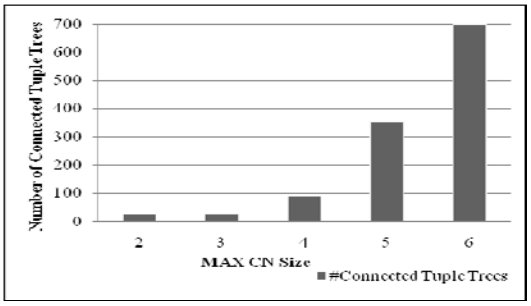

Figure 6. Connected Tuple Trees with Different MAX CN Size

I. CONCLUSION

Keyword search allows ordinary users to find text information in relational database with much higher flexibility. In this paper, we proposed a new method for querying connected tuples trees for relational keyword search. 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 keyword query consists of the minimal connected tuple trees, which potentially include tuples from multiple relations in database. To produce the 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 number of CNs for small CN size with the generating candidate network methods in previous systems implemented by the authors in the given references. It can generate the minimal number of CNs by doing minimal accesses to the database, and does not have data bound with the maximum number of tuple set. We presented the experimental results on DBLP show that the proposed method generates the result approximately for the user desired query. We plan to retrieve relevant queries in relational database by using a ranking method based on the virtual document. By ranking connected tuple trees, we hope to improve search effectiveness further.

REFERENCES

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

[2] A.Markowetz, Y.Yang, D.Papadias, "Keyword Search on Relational Data Streams," Proc. 2007 ACM SIGMOD Int. Conf. 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. Conf. 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. Conf. 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. Conf. on Management of data, 2009, pp. 681-694.

[7] P.Li, Q.Zhu, S.Wang, "The Research on the Algorithms of Keyword Search in Relational Database," Springer, 2008, pp. 134-143.

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

[9] S.Agrawal, S.Chaudhuri, G.Das, "DBXplorer: A System for Keyword-Based Search over Relational Database," Proc. 18th Int. Conf. on Data Engineering, 2002, pp. 5-16.

[10] 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.

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

[12] S.Wang, J.Zhang, Z.Peng, J.Zhan, Q.Wang, "Study on Efficiency and Effectiveness of KSORD," APWeb/WAIM Int. Workshops, 2007, pp. 6-17.

[13] A.Felner, R.E.Korf, S.Hanan, "Additive Pattern Database Heuristics," Journal of Artificial Intelligence Research, 2004, pp.279-318.

[14] 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.

[15] R.E.Korf, "Depth-First Iterative-Deepening: An Optimal Admissible Tree Search," Artificial Intelligent, 1985, pp. 97-109.