Answering Top-k Keyword Queries on Relational Databases

Myint Myint Thein, Mie Mie Su Thwin
mmyintt@gmail.com, drmiemiesuthwin@mmcert.org.mm
University of Computer Studies, Mandalay
Myanmar

Abstract

Keyword search in relational databases allows the user to search information without knowing database schema and using structural query language. As results needed by user are assembled from connected tuples of multiple relations, ranking keyword queries are needed to retrieve relevant results. For a given keyword query, we first generate candidate networks and also produce connected tuple trees according to the generated candidate networks by reducing the size of intermediate joining results. We then model the generated connected tuple trees as a document and evaluate score for each document to estimate its relevance. Finally, we retrieve top-k keyword queries by ranking the results. In this paper, we propose a new ranking method based on virtual document. We also propose Top-k CTT algorithm by using the frequency threshold value. The experimental results are shown by comparison of the proposed ranking method and the previous ranking methods on IMDB and DBLP datasets.

Keywords— Candidate Network; Connected Tuple Tree; Top-k; Keyword Query; Keyword Search; Relational Databases;

1. INTRODUCTION

The most critical and valuable amount of data such as business data has been stored in relational databases. Relational database management system (RDBMS) is a DBMS in which data is saved in tables and the relationships among the data are saved in tables. The data can be reassembled and accessed in many different ways without change the table forms. Most commercial RDBMS uses a structured query language (SQL) to access the database. With more and more data being stored in relational database, it has become crucial for users to be able to search and browse the information stored in them. Keyword search in relational databases enables ordinary users, who do not understand the database schema and SQL, to find the connected tuple sets among the tuples stored in relations, with a given set of keywords. 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.

In schema based keyword search in relational database, it has a common method that is generating candidate network in schema graph transformed from relations. Data is stored in the form of columns, tables and primary key to foreign key relationships in relational databases. According to develop the schema graph, we illustrate two schema graphs as examples. Figure 1 shows the schema graph of publication database from DBLP dataset. The movies database schema graph of IMDB dataset shows in Figure 2. 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 databases, generating minimum joining tuples sets of relations that contained keyword is called candidate network (CN), such as SQL. A candidate network must satisfy the two conditions, total and minimal. Because it is meaningless if two tuples in a candidate network are too far away from each other, the maximum numbers of tuples allowed in a candidate network are needed to specify (Yu & Qin, 2010).

Suppose user wants to get the papers written by “Jinlin Chen” from DBLP database. The system generates the relevant CNs, such as Person $\bowtie$ Relation-Person-InProceeding $\bowtie$ InProceeding, with multiple tuples from different relations joined by foreign keys. Generating all valid candidate networks that are called connected tuple trees (CTTs) by joining tuples from multiple relations. There are many connected tuple trees that can be results for the query. These results are not surely useful to the user. We need to compute a single score for each CN in order to rank the relevant results. So, a ranking method is essential for getting user satisfaction. For the ranking method, some systems considered each text column as a collection and each value in the text column as document by using IR weighting methods. The results are ranked according to a final score that is obtained by dividing the sum of all these scores by the number of tuples in the tuple trees. These methods can help improve the keyword search quality in relational database.

Recently, several researchers have been done on keyword search systems for relational databases (Baid, 2010; Qin & Yu, 2009; Stefanidis & Drosou, 2010; Wang, 2007; Li & Zhu, 2008). In candidate network generation, DISCOVER (Hristidis & Papakonstaninou, 2002) and Liu & Yu & Meng (2006) developed the CN generation algorithm based on a breadth-first traversal in the search space. SPARK (Luo & Wang, 2011) is subsequently improved the CN generation algorithm by canonical. In the ranking strategy, IR-Style (Hristidis & Gravano, 2003) incorporated a state-of-the-art IR ranking method to address the retrieval effectiveness issue and presented several efficient query execution algorithms optimized for returning top-k relevant answers. Liu & Yu & Meng (2006) improved the ranking method in IR-Style by adapting four normalizations. Both two systems considered each text column as a collection and each value in the text column as document. BANKS (Aditya & Bhalotia, 2002) and BANKSII (Kacholia & Pandit, 2005) took another approach by modeling the database content as a graph and proposed sophisticated ranking and query execution algorithms. The theoretical aspect of efficient query processing for top-k keyword queries is studied in (Ding & Yu, 2007; Kashem & Chowdhury, 2010, Xu & Ishikawa, 2009; Kimelfeld & Sagiy, 2006). SPARK (2011) further modified the ranking method of IR-Style by introducing the concept of a virtual document and presented efficient query evaluation algorithms for their ranking method. Despite the existing studies, there are still several issues with existing ranking methods. Some of the existing ranking methods may produce the meaningless results which are disappointed for user.

In this paper, we focus on retrieving the top-k rank relevant results with given keyword query. We propose a new ranking method based on virtual document. The proposed ranking method can produce meaningful results by applying two factors, content factor and structural factor. The proposed ranking method can evaluate the accurate scores of the relevant results from relational database for the user. If query processing algorithms is not optimized for the ranking method and top-k queries, the query execution time will become prohibitively large for large databases. We also propose the Top-k CTT algorithm by using frequency threshold value. The proposed Top-k CTT algorithm can retrieve the efficient top-k queries. We conduct the experimental results on DBLP and IMDB databases. The results show that the proposed ranking method supports effective keyword search on large amounts of relational data.

The rest of the paper is organized as follows: Section 2 discusses the related work. Section 3 presents the basic concept of keyword query and CN. Section 4 presents the CN generation. Section 5 illustrates the CN evaluation. Section 6 discusses the ranking method and the query processing in Section 7. Section 8 shows the experimental results and Section 9 concludes this paper.

2. RELATED WORK

The techniques to support keyword search in relational databases can be divided into two categories. 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. One important issue of keyword search in relational databases is the efficiency and effectiveness.

Several researchers have been done on early keyword search systems for relational databases (Agrawal & Chaudhuri, 2002; Hristidis & Papakonstaninou, 2002). Early works in schema graph approach like DBXplorer (Agrawal & Chaudhuri, 2002) and DISCOVER simply consider the number of joins in the tuple trees. DISCOVER (2003) adapted IR-style document-relevant ranking strategies to the problem of processing free-form keyword queries over RDBMSs. DISCOVER also proposed IR-style ranking method in straightforward manner to rank tuple trees by assuming OR semantics for answers. This method had not considered the effectiveness of the query results. Liu & Yu & Meng (2006) described the ranking formula by adapting four normalizations: tuple tree size normalization, document length normalization, document frequency normalization and inter-document weight normalization. This score function is not monotonic due to the four normalizations.

SPARK (2011) proposed to model a joining tuple tree as a virtual document. SPARK studied the tree level ranking function which does not satisfy tuple monotonicity. In order to handle such non-monotonic score functions, a new monotonic upper bound function is introduced. The intuition behind the upper bound function is that, if the upper bound score is already smaller than the score of a certain result, then all the upper bound scores of unseen tuples will be smaller than the score of this result due to the monotonicity of the upper bound function. SPARK proposed two new algorithms that are Skyline-Sweeping algorithm and Block-Pipelined algorithm.

BANKS (2002) also found tuple trees from the data graph directly by using the Steiner tree algorithm. For a data graph, it uses PageRank style methods to assign weights to tuples and edges between them. A combination of tuple weights and edge weights is used to compute the confidence of a tuple tree. BANKSII (2005) is an improvement of BANKS which introduces a novel technique of bidirectional expansion to improve search efficiency. Li & Feng (2008) proposed a new concept referred to as a compact Steiner Tree, which can be used to approximate the Steiner tree problem for answering top-k keyword queries efficiently. They also proposed a novel structure-aware index to support keyword search. In order to balance the importance of individual nodes and the structural cohesiveness of the results, Xiaohui & Huxia (2012) proposed a random walk model with message passing to match the characteristics of keyword search in databases.

3. PRELIMINARIES

3.1 Data Model

A relational database can be viewed as a graph which represents a relational model such as schema graph G_s (V, E). A relation 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_j 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.

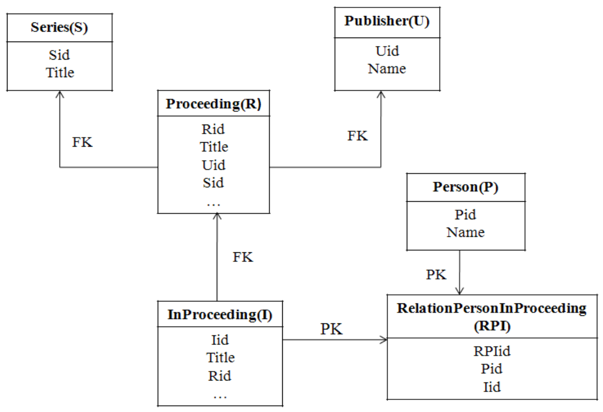


Figure 1. Publication Database Schema Graph

We use directed schema graphs that show in Figure 1 and Figure 2 as the schema graph of publication database and movies database schema graph. 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 primary-foreign key relationship between any two relations.

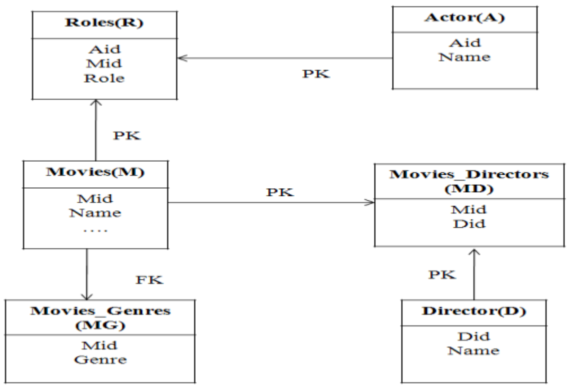


Figure 2. Movies Database Schema Graph

3.2 Connected Tuple Tree

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. Each node t_i is a tuple in the database, and each pair of adjacent tuples in CTT is connected via a foreign key to primary key relationship. Suppose (Ri,Rj) is an edge in the schema graph. Let ti Є Ri, tj Є Rj, and (ti join tj) Є (Ri join Rj). Then (ti,tj) is an edge in the connected tuple tree. The size of a CTT is the number of tuples involved. Note that a single tuple is the connected tuple tree with size 1. The size of CTT can have arbitrarily large size, when there exists a many to many relationship in the schema graph. Therefore, the size of connected tuple tree is needed to only data bound.

3.3 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 belong 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 is a tree of tuple sets R^N or 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 edge (R_i,R_j) in the schema graph G_s. The size of a CN is the number of its tuple sets.

In the framework of RDBMS, a keyword query is processed in the two main steps that are candidate network generation and candidate network evaluation. In candidate network generation step, it generates a set of CNs over schema graph G_s. The set of CNs shall be sound or complete and duplicate-free upon the maximal size. In candidate network evaluation step, it evaluates the generated CNs by reducing the size of intermediate joining results. We present how to generate minimal number of CNs and how to evaluate the generated CNs in Section 4 and Section 5.

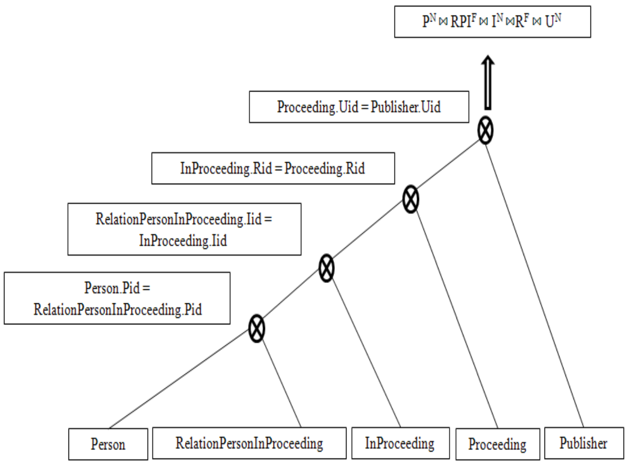
4. CANDIDATE NETWORK GENERATION

In schema-based keyword search in relational database, the generating all candidate networks for keyword query Q satisfy the two properties, such as complete and duplication-free, which are listed below.

Property1. The set contains all CNs with no more than MAXN (completeness).

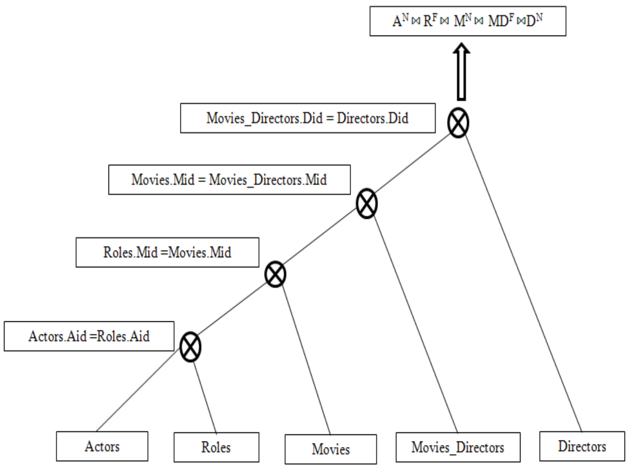
Property2. Every two CNs are not isomorphic to each other (duplication-free).

We proposed a new CN generation (AT_CNGen) algorithm based on adjacent tuple list in our previous work (Thein & Thwin, 2012). AT_CNGen algorithm generate all CNs for a given query Q and schema graph SG. In order to generate valid CNs, AT_CNGen algorithm first accepts the adjacent tuple lists as input. During each CN generation, AT_CNGen calls the two procedures which get a query tuple sets T for a given query Q and take the adjacent tuple list for getting query tuple sets. Each adjacent tuple list d adds a CN, if d is a valid CN and is not duplicated on each others. If d is invalid and identical, d is pruned. Finally, AT_CNGen algorithm generates all candidate networks no more than the maximal number of tuple sets for the user input keywords. The generated CNs is only data bounded by following Properties 1 and 2. For example, we illustrate the adjacent tuple list for query Q = "Chen Web Springer" in DBLP and the adjacent tuple list for query Q = "Black Jack David" in IMDB that are shown in Figure 1 and Figure 2.



$P^N \bowtie RPI^F \bowtie I^N \bowtie R^F \bowtie U^N$
Proceeding.Uid = Publisher.Uid
InProceeding.Rid = Proceeding.Rid
RelationPersonInProceeding.Iid = InProceeding.Iid
Person.Pid = RelationPersonInProceeding.Pid
Person
RelationPersonInProceeding
InProceeding
Proceeding
Publisher

Figure 3. Adjacent Tuple List for DBLP



$A^N \bowtie R^F \bowtie M^N \bowtie MD^F \bowtie D^N$
Movies_Directors.Did = Directors.Did
Movies.Mid = Movies_Directors.Mid
Roles.Mid =Movies.Mid
Actors.Aid =Roles.Aid
Actors
Roles
Movies
Movies_Directors
Directors

Figure 4. Adjacent Tuple List for IMDB

5. Candidate Network Evaluation

We present the generating CTTs by executing the generated CNs in order to get the results. For a given query Q, the connected tuple tree is generated according to an evaluated 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 SG. We proposed D_CNEval algorithm for the CN evaluation in our previous work (Thein & Thwin, 2012). D_CNEval algorithm is observed that there is substantial evaluating the common join expressions among CNs. As a consequence, the computational efforts can be saved if multiple CNs can be executed in a calculated way that minimizes the sizes of joining intermediate results. Each CTT that defined satisfaction in order to properties 3 and 4 as follow:

Property3. If a node in connected tuple tree is one of tuples in relation, it contains at least one keyword in query Q (completeness).

Property4. There is no duplicate tuple with each other in the connected tuple tree (duplication-free).

The examples of evaluated CN for Q = "Chen Web Springer" in DBLP and Q = "Black Jack David" in IMDB that are shown in Figure 10 and Figure 11.

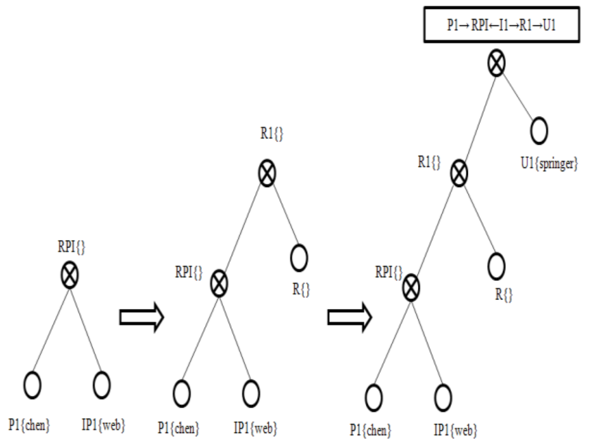


Figure 5. Processing of Evaluated CN on DBLP

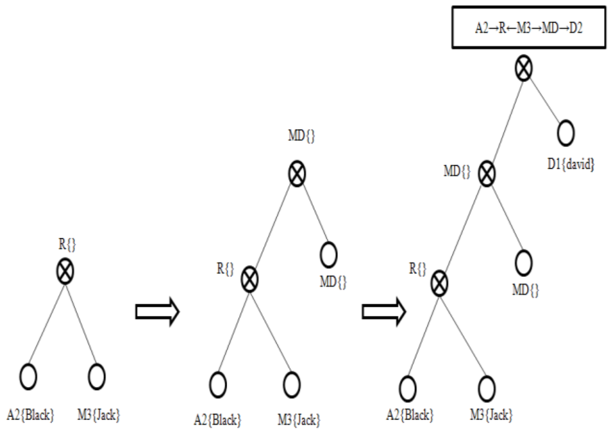


Figure 6. Processing of Evaluate CN on IMDB

6. RANKING METHOD

6.1 Problems of Existing Ranking Methods

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 (Liu & Yu, 2006; Xiaohui & Huxia, 2012). 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.

In general, retrieval effectiveness is vital to keyword search on relational database due to the fuzzy nature of keyword queries. In a keyword query, each query result is assigned a relevance score and all results are presented in decreasing order of that score. There are three ranking methods considered by existing relational keyword search systems. The first ranking method is the IR score of attribute values. The IR score of an attribute value is computed according to the number of keywords it contained. Traditional information retrieval weighting methods, such as TF-IDF weighting, can be used to compute the IR score. In DISCOVER, the IR score of an attribute value is computed by database system. The second ranking method is the structure or semantics of result trees. Result trees refer to connection trees used by BANKS, join trees used by DBXplorer, and candidate networks used by DISCOVER. A result tree is scored by its size which is the number of nodes or edges in BANKS, DBXplorer and DISCOVER. In addition, it is useful to score a result tree by its semantics. The third ranking method is the semantics of links. The score of a node depends on the links between it and the other nodes. The score functions used by existing relational keyword search systems are related to the definition of query result. If a query result contains many nodes, the IR score of attribute values and the structure of result trees are considered. If a query result contains only one node, the structure of result trees is not necessary to be considered.

In summary, most of existing ranking systems have considered the size of an answer as a ranking factor to compute the relevance. The basic idea of the ranking method is: assign to each tuple in the joined tuple tree a score by using a standard IR-ranking formula, and combine the individual scores together by using an aggregation function, such as SUM, to obtain the final score (Xu & Ishikawa, 2009). In this method, the ranking results contain a large amount of one keyword query over results that contain all or most keyword queries but only once. This method is contradicted to user perception by ranking results. To solve this problem, we propose a new ranking method based on the virtual document to retrieve the relevant ranking results by supporting modified IR ranking score.

6.2 Proposed Ranking Method

The proposed ranking method is presented by applying two factors, such as content factor and structural factor, on virtual document. The content factor is computed with the local score and global score. The structural factor is calculated for the size normalization.

6.2.1 Modeling Connected Tuple Tree as a Virtual Document

A model is proposed on the idea of modeling a connected tuple tree as a virtual document in our previous work (Thein, 2012). Consequently, the entire results produced by a CTT will be modeled as a document collection that is a string. The rationale is that most of the CTTs carry certain distinct semantics. E.g., P1 → I2 gives all details about author and their related inproceeding that are collectively relevant to the query and form integral logical information units. In fact, it was split into multiple tables due to the normalization requirement imposed by the physical implementation of the RDBMSs. SPARK proposed the idea of modeling a CN as a virtual document. In our model, we first execute a CTT as SQL queries. Then, the executed queries are modeled as a string and this string is dynamically stored into hash table. By the way, we avoid the modeling identical document. We show the related virtual document for each CTT as example in Figure 7.

CTT	Document
P1→I2	Jinlin Chen→An Adaptive Web Content Delivery System
P1→I1	Jinlin Chen→Visual Based Content Understanding towards Web Adaptation
P2→I3	Peter P.Chen→ER Model, XML and the Web

Figure 7. Related Virtual Document for a CTT

6.2.2 Content Factor

To compute content factor of a CTT, we use two IR ranking methods such as TF-IDF and Extended Boolean Model. The TF-IDF score emphasizes keyword matching, while Extended Boolean Model score emphasizes the similarity between keyword and a virtual document as a whole. To calculate the global score of a CTT, we model a CTT as a virtual document by concatenating the text contents of tuples in the CTT and a TF-IDF score for the virtual document is calculated without assuming tuples matching a certain keyword are uniformly and independently distributed in each relation. We obtain a local score for each virtual document to calculate a numeric score for each keyword.

6.2.2.1 Local Score

By adopting such a virtual document, we assign an IR ranking score, such as score$_a$, to a CTT by using Equation (1).

$$\mathrm{score}_a(k,D) = \frac{\mathrm{ntf}}{\mathrm{ndl}} * \ln(\mathrm{idf}) \qquad (1)$$

$$\mathrm{ndl} = \sum_{k\in D}(1-s) + s * \frac{dl_{CTT}}{avgdl(CTT)} \qquad (2)$$

$$\mathrm{ntf} = \sum_{k\in D} 1 + \ln(tf_k(CTT)) \qquad (3)$$

$$tf_k(CTT) = \sum_{k\in D} \frac{kf_i}{\max\{kf_1,\ldots,kf_i\}} \qquad (4)$$

$$\mathrm{idf} = \sum_{k\in D} \frac{df_k(CTT)+1}{N(CTT)} \qquad (5)$$

, where ntf indicates the normalized term frequency, ndl is the normalized document length which is the length of modeling CTT. idf is the inverse document frequency and tf$_k$(CTT) denotes the number of occurrences of the CTT in a document.

Equation (2) used to compute value of document length normalization, which dl$_{CTT}$ is the length of CTT. In our experiment, s is complete to 0.2 for all condition. Equation (5) computes the inverse document frequency for each modeling connected tuple tree. We can get the value of N(CTT) which is the number of CTT. We observe that the idf equation in IR cannot directly use because this equation cannot calculate to get normalize value. For this case, we compute idf value in order to Equation (5) to get normalize value. Equation (3) evaluates the normalized trem frequency, whereas Equation (4) used to compute the number of occurrences of the CTT which belongs to the connected tuple tree such as document.

Table 1. Evaluating Different Scores for Query “chen web content”

CTT	t Є CTT	tf_{chen}	tf_{web}	$tf_{content}$	Score$_a$
P1→I2	P1	1	0	0	2.66
	I2	0	1	1	
P1→I1	P1	1	0	0	2.67
	I1	0	1	1	
P2→I3	P2	1	0	0	2.17
	I3	0	1	0	

For example, we compute each score value with “chen web content” query step by step. For this query, some examples of the connected tuples trees include: P1→I2, P1→I1, P2→I3 and P3→I4. Note that P3→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 results for this query may be: $P1 \rightarrow I2$, $P1 \rightarrow I1$, and $P2 \rightarrow I3$ whereas nodes P1 and P2 contain the keyword “chen”, and nodes I1 and I2 contain two keywords “web” and “content”, I3 contains the keyword “web”. Then, we model a document for each CTT according to Figure 7. For the modeling query of each CTT, we calculate each score value by using $score_a$ that is shown in Table 1.

6.2.2.2 Global Score

The AND semantics and OR semantics are considered for the upper bound of keyword query. We believe that users usually prefer documents matching many query keywords to those matching only few keywords. To quantify this factor, we propose to multiply a global score to the raw IR ranking score. The proposed global score is derived from the Extended Boolean Model. The idea of the Extended Boolean Model is to make use of partial matching and term weights as in the vector space model. It combines the characteristics of the vector space model with the properties of boolean algebra and rank the similarity between queries and documents. This way a document may be somewhat relevant if it matches some of the queried terms and will be returned as a result. So, we apply the global score that is shown in Equation (6).

$$score_b(k, D) = 1 - \left(\frac{\sum_{j<m} \left(1 - w_j\right)^p}{m} \right)^{1/p} \qquad (6)$$

where w_j is the weight score of a CTT that is defined as follow:

$$w_j(k, D) = ntf * \ln(idf) \qquad (7)$$

In Equation (6), p is a tuning parameter. p can smoothly switch the completeness factor biased towards the OR semantics to the AND semantics, when p increases from 1.0 to ∞. In our experiment, we observed that a p value of 2.0 is already good enough to enforce the AND-semantics for almost all the queries tested.

6.2.3 Structural Factor

The size of the CN or CTT is also an important factor. A larger CTT tends to have more occurrences of keywords. There is a common intuition behind all existing structural score. They all try to define a structural size of a CTT and then use the inverse of this structural size to measure the structural score. Many structural size definitions have been proposed. For example, existing schema based keyword search approaches use the total number of tuples in a CTT as the structural size and existing graph based keyword search approaches use the aggregate of edge weights as the structural size. There is a common underlying assumption: it is easier to find a CTT matching all given keywords in a larger CN.

After computing the modified IR scoring method, we then evaluate a score value for the size of CTT and the size of the given query, especially for a keyword query whose relevant results are connected tuple tree involving multiple tuples, each of which contains a subset of the keywords query. To approximately the user perception, we define the structural score for a query that is as follow:

$$score_c(k, D) = \frac{\ln(size(Q))}{\ln(size(CTT))} \qquad (8)$$

6.2.4 Final Score

The proposed ranking method can be conceptually thought as first merging all tuples in a CTT into a virtual document, and then obtaining its local score in Equation (1), the global score in Equation (6), and the structural score in Equation (8). Finally, the final score of a CTT to a keyword query is the product of all the three scores:

$$score(T, Q) = score_a(k, D) * score_b(k, D) * score_c(k, D) \qquad (9)$$

We can get the significant score for the highest relevant keyword query after computing the final score (T,Q) for each connected tuple tree. We evaluate each CTT with the proposed ranking method by using two relations: person and inproceeding of DBLP dataset. Table 2 shows the relevant results for query “chen web content” with CTT and its final score according to multiply each score value of $score_a$, $score_b$ and $score_c$. In order to this table, we can see that score value of P1→I2 is increased with the highest relevant score value.

Table 2. Relevant Keyword Queries for Query “chen web content”

CTT	Score
Jinlin Chen→An Adaptive Web Content Delivery System	1.41
Jinlin Chen→Visual Based Content Understanding towards Web Adaptation	1.33
Peter P.Chen→ER Model, XML and the Web	1.08

7. QUERY PROCESSING

7.1 Problems of Query Processing

The efficiency problem is also important to user experience. Generating all possible CTTs will induce prohibitively long query time for large databases. So, we need to consider efficiency issues of the relational keyword search system. In existing relational keyword search system, the system first generates all possible CNs. Assume that we can estimate an upper bound for the highest result score of all CTTs corresponding to a CN. Each time, the CN with the highest upper bound is chose. So, we can find a CTT with the highest score, corresponding to the chosen CN. Next, we enumerate possible CTTs using the top-k algorithms for the chosen CN. For each set of candidates such as non-free tuple sets, we then generate all corresponding CTTs by joining candidates together with free tuples. For each generated CTT, we update top-k results. This process is not efficient if the number of tuples is large, since it needs to join all tuples in each CN and store a large number of CTTs with the highest score. Finally, the system stops when the highest CN upper bound is not higher than the current top-k result scores (Zhang & IIyas, 2011; Zeng & Bao, 2012). If the system estimates an invalid upper bound for the highest result score of all CTTs, it cannot reach to terminate efficiently. In general, there are still efficiency issues although the existing relational keyword search systems consider generating the efficient results.

In this section, we focus on retrieving top-k query to improve efficiency, when we presented the CN generation and CN evaluation that is considered the efficiency. We implement the top-k query processing in the application level. We discuss the proposed Top-k CTT algorithm in the next section.

7.2 Top-k CTT Algorithm

The user is more interested in the top-k query answers in the potentially huge answer space. So, we focus on how to retrieve the top-k results for keyword query. After generating all CTTs for a keyword query, Top-k CTT algorithm is presented to retrieve top-k rank result for the query processing that is shown in Figure 8. The Top-k CTT algorithm is devised to perform this task.

We use the non-free tuple sets as input, which are belong to that CN. And then we use the frequency threshold T for each list pointing to the current element. For each CTT, if the generated CTT is not null, the algorithm first computes the score according to Equation (9). It adds the CTT with related score value into hash table H. After that the sort function sorts each CTT with maximum score value. If T is equal or less than the size of H, the algorithm add CTT with the maximum score value into the top-k as result until T is equal or less than the size of H and T is not null. Then if the size of H is less than the threshold T, the size of H set to T. Again the algorithm adds CTT with the maximum score value into the top-k as result until T is not null. With computing the threshold T, the algorithm terminates when T is greater than the size of H. Finally, the algorithm generates the top-k results as the above processes.

In Top-k CTT algorithm, the for loop is achieved at most |M| times for each CTT in M, where |M| is the size of CTT that is not null. If CTT in M is null, we can reduce its time in

O(1). Each CTT is ranked in hash table by calling RankScore function. This step is increasing the computation time in O(1). So, the whole hash table is worked in the same |M| time. After sorting CTT with each score, we check the size of hash table. If this size is not zero, while loop is performed at most |N| times for every connected tuple sets in hash table, where |N| is the given threshold value. If N is less than size of result sets in hash table, we add each result into array. This step is increasing the computation time in O(1). If N is greater than maximun number of tuple sets, the algorithm is terminate. In this step, we can reduce its time in O(N). Hence the total execution time takes in the worst case time O(|M|-|N|).

```
Input: A set of CTT, Threshold T
Output: A set of top-k queries Top-k
Let H be the hash table that contains CTT and related
score.
1.Top-k ← Φ
2. H ← Φ
3. For each c Є CTT do
5.   If (CTT != Null) Then
6.      H.push(CTT,RankScore(CTT))
             // According to the Equation (9).
7.   End if
8.  End for
9.  H′ ← sortByScore(H)
10. s ← sizeof(H′)
11. If (H′.isEMPTY()) Then
12.     If (T ≤ s) Then
13.       While (T ≤ s && T != Φ) {
14.                 Add H′.pop-max() into Top-k }
15      End if
16.     Else if (s < T) Then
17.             T ←  s
18.          While (T != Φ) {
19.                 Add H′.pop-max() into Top-k }
20      End if
21.     Else
22.         break.
23. End if
24. Return Top-k.
```

Figure 8. Top-k CTT Algorithm

8. Performance Evaluation

8.1 Evaluation Setup

The search efficiency of proposed algorithms is evaluated on DBLP and IMDB datasets. All queries generating algorithms were implemented in Java, and JDBC was used to connect to the database. We conducted all the experiments on Core(TM) 2 Duo CPU and 2GB memory laptop running XP. We take the average executing time on running 15 times.

Dataset: We use two real datasets the Original Digital Bibliography and Library Project (DBLP) dataset (dblp.uni.trier) and the Internet Movie Database (IMDB) (imdb.com) in our evaluation. DBLP contains publications records. IMDB contains movies records. Table 3 and Table 4 show the schema and statistic of two datasets.

Table 3. Statistics of DBLP Dataset

Relation Schema	#Tuples
Person(Pid,Name)	174,709
InProceeding(Iid,Title,Pages,Rid)	212,273
Proceeding(Rid,Title,Uid,Sid,…)	3,007
Publisher(Uid,Name)	86
Series(Sid,Title)	24
RelationPersonInProceeding(Pid,IPid)	491,777
Total Number of Tuples	881,876

Table 4. Statistics of IMDB Dataset

Relation Schema	#Tuples
Actors(Aid,Name)	817,718
Directors(Did,Name)	86,880
Movies(Mid,Name,Year,Rank)	388,269
Movies-Directors(Mid,Did)	406,967
Movies-Genres(Mid,Genre)	417,784
Roles(Aid,Mid,Role)	3,432,630
Total Number of Tuples	5,550,248

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 and test on keyword queries on 200 queries for two datasets. According to the space, we present 20 queries in two datasets that is shown in Figure 9 and Figure 10.

Query	Keywords
Q1	nikos constraint
Q2	chen web content
Q3	agent based system
Q4	knowledge based processing
Q5	java programming
Q6	query semantic by davis
Q7	natural language processing
Q8	non-monotonic reasoning
Q9	compiler generator
Q10	relational databases

Figure 9. Keywords Queries on DBLP

Query	Keywords
MQ1	alexander
MQ2	hollywood
MQ3	bill harry fighting men
MQ4	blake death
MQ5	elley love story
MQ6	mile allen
MQ7	countdown
MQ8	come away 2005
MQ9	gold anderson
MQ10	black jack david

Figure 10. Keywords Queries on IMDB

8.2 Evaluation Results

To measure the effectiveness, we adopt two metrics used in previous studies (Liu & Yu & Meng, 2006; Xu & Ishikawa, 2009): (1) number of top-1 results that are relevant (#Rel), and (2) reciprocal rank (R-Rank), for a given query. The reciprocal rank is 1 divided by the rank at which the first correct answer is returned or 0 if no correct results are returned. In order to find the relevant results, we used the ranking strategies: DISCOVER (Hristidis & Gravano, 2003), SPARK (Luo & Wang, 2011) and our ranking method for the same query. Then, we manually evaluated the results and selected the relevant result for each query. Also, the experiments have been conducted on our proposed ranking method by varying p from 1 to 3 that is shown in Table 5. As the default p = 1 already returns relevant results, but R_Rank values are affected by the varying p. With increasing value of p such as p = 3, the R_Rank value decrease. We observe that p = 2 return the relevant results and the R_Rank value increase.

Table 5. P's Impact on #Rel and R_Rank

	P = 1	P = 2	P = 3
#Rel	185	200	60
R_Rank	≥0.5	1	≤0.166

The evaluation results of DISCOVER, SPARK and the proposed method are compared by using the same DBLP and IMDB datasets. The manually evaluated relevant results are based on the AND semantics for keyword queries. Table 6. shows the #Rel and R_Rank values of previous methods and proposed ranking method on the top-10 results of the same queries. When we test all methods on #Rel and R_Rank, we observe that there is significantly dissimilar value on these methods. The proposed ranking method always returns the relevant results as top-1 result for 100 tested queries on DBLP. So, R_Rank value of proposed ranking method is 1. SPARK actually performs better than DISCOVER, because it often returns relevant results within the top-6 results, while DISCOVER method often fails to find any relevant result in the top-10 results. This is reflected in their R-Rank measures.

Table 6. #Rel and R_Rank for Existing Methods and Proposed Method on DBLP

	DISCOVER	SPARK	Proposed Method
#Rel	≤ 25	≤ 55	100
R_Rank	≤ 0.3	≤ 0.83	1

The effectiveness is also measured with this two metric on IMDB, where the effectiveness of relevant results in IMDB are similar to DBLP.

In Table 7, it shows the comparison of DISCOVER, SPARK and the proposed ranking method by using R_Rank and #Rel metrics in order to most queries. Although DISCOVER got some relevant results on #Rel for small queries, we see that DISCOVER is more affected on R-Rank than SPARK. SPARK actually performs better than DISCOVER, because it often returns relevant results within the top-5 results, while DISCOVER method often returns relevant results in the top-10 results. At that time, the proposed ranking method can return relevant results as top-1 results for 100 tested queries on IMDB. In practice, the proposed ranking method achieves more relevant results than the existing ranking methods.

Table 7. #Rel and R_Rank for Existing Methods and Proposed Method on IMDB

	DISCOVER	SPARK	Proposed Method
#Rel	≤ 27	≤ 63	100
R_Rank	≤ 0.33	≤ 0.89	1

8.3 Effectiveness of Proposed Ranking Method and Existing Ranking Methods

The effectiveness of DISCOVER (Hristidis & Gravano, 2003), SPARK (Luo & Wang, 2011) and the proposed ranking method is compared on the same DBLP and IMDB datasets. The impact of top rank relevant answers on each query is shown. For query “nikos clique”, results are shown in Table 8, Table 9 and Table 10. DISCOVER ranks in top-3 answer for this query. But SPARK can rank in top-1 result. The proposed ranking method can rank in top-1 result and the next four answers are the other papers that are written by the author “nikos”. In this query, the proposed ranking method and SPARK rank the same inproceeding paper.

In query “data mining”, the proposed method and SPARK rank the same paper for top-1 result, while DISCOVER rank the other paper as top-1 result. In this case, all methods rank top-1 result that contains the two keywords “data” and “mining”. So, all answers in Table 11 are acceptable. Then, we test query “peter for semantic web IFIP” in top-2 answers. For this query, DISCOVER and SPARK cannot rank relevant answers that contain all keywords. The proposed method ranks in top-1 result with the complete keywords. In top-2 result, it is a meaningful result that contains three keywords although this result cannot all keywords. All results are shown in Table 12, Table 13 and Table 14.

Table 8. Top-5 Answers on DISCOVER for Query “nikos clique”

Rank	Top-5 Answers on DISCOVER
1	Marcello Pelillo → RPI ← Clique Finding Relaxation Labeling Networks
2	Haris Papageorgiou, Nikos Lourados, Symeon Retalis,Dimitrios Retalis → RPI ← Kairos: A Web-Based System for Automatic Generation of Weather Forecasts in Two Languages, Greek-English
3	**Nikos** Mamoulis, Dimitris Papadias → RPI ← Constraint-Based Algorithms for Computing **Clique** Intersection Joins
4	Bruno Courcelle, Johann A. Makowsky, Udi Rotics → RPI ← Linear Time Solvable Optimization Problems on Graphs of Bounded Clique Width
5	Nikos Fakotakis, Kyriakos N. Sgarbas, George K. Kokkinakis → RPI ← Incremental Construction of Compact Acyclic NFAs

Table 9. Top-5 Answers on SPARK for Query “nikos clique”

Rank	Top-5 Answers on SPARK
1	**Nikos** Mamoulis, Dimitris Papadias → RPI ← Constraint-Based Algorithms for Computing **Clique** Intersection Joins
2	Mountaz Hascoft-Zizi, Nikos Pediotakis → RPI ← Visual Relevance Analysis
3	Nikos Mamoulis, Dimitris Papadias → RPI ← Hierarchical Constraint Satisfaction in Spatial Databases
4	Nikos Fakotakis,Kyriakos N. Sgarbas → RPI ← Machine Learning in Human Language Technology
5	Theodoros Bozios,Nikos B. Pronios → RPI ←Multimedia Synchronization: The Role of the Communication System

Table 10. Top-5 Answers on Proposed Method for Query "nikos clique"

Rank	Top-5 Answers on Proposed Method
1	**Nikos** Mamoulis, Dimitris Papadias → RPI ← Constraint-Based Algorithms for Computing **Clique** Intersection Joins.
2	Siegfried Reich,Dimitris Christodoulakis, Nikos Karousos,Manolis Tzagarakis → RPI ← Naming as a fundamental concept of open hypermedia systems.
3	Dimitris Papadias, Nikos I. Karacapilidis → RPI ← Hermes: Supporting Argumentative Discourse in Multi-Agent Decision Making.
4	Nikos Mamoulis, Vasilis Delis, Dimitris Papadias → RPI ← Assessing Multimedia Similarity: A Framework for Structure and Motion.
5	Theodoros Bozios,Nikos B. Pronios → RPI ←Multimedia Synchronization: The Role of the Communication System

Table 11. Top-1 Answer on Query "data mining"

Method	Top-1 Answer
Proposed Method	Christos Faloutsos, Tara M. Madhyastha, Mengzhi Wang, Ngai Hang Chan → RPI ← **Data Mining** Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic
DISCOVER	Ben Shneiderman → RPI ← Inventing Discovery Tools: Combining Information Visualization with **Data Mining**
SPARK	Christos Faloutsos,Tara M. Madhyastha, Mengzhi Wang, Ngai Hang Chan → RPI ← **Data Mining** Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic

Table 12. Top-2 Answers on DISCOVER for Query "peter for semantic web IFIP"

Rank	Top-2 Answers on DISCOVER Method
1	**Peter** A. Flach → RPI ← An Analysis of Various Forms of "Jumping to Conclusions" → R → Lecture Notes in Computer Science
2	Maryline Laurent → RPI ← Security Flows Analysis of the ATM Emulated LAN Architecture → R → **IFIP** Conference Proceedings

Table 13. Top-2 Answers on SPARK for Query "peter for semantic web IFIP"

Rank	Top-2 Answers on SPARK
1	Denis Yaro → RPI ← Cooperative management → R → **IFIP** Conference Proceedings
2	Nicklas Lundblad → RPI ← Digital Evidence → R → **IFIP** Conference Proceedings

Table 14. Top-2 Answers on Proposed Method for Query "peter for semantic web IFIP"

Rank	Top-2 Answers on Proposed Method
1	**Peter** M. D. Gray, Kit-ying Hui, Alun D. Preece → RPI ← Mobile Constraints for **Semantic Web** Applications → R → **IFIP** Conference Proceedings
2	**Peter** M. D. Gray, Suzanne M. Embury → RPI ← Compiling a Declarative High-Level Language for **Semantic** Integrity Constraints → R → **IFIP** Conference Proceedings

After studying the effectiveness of keyword queries on DBLP, we also discuss the effectiveness of keyword queries on IMDB. In query "godfather", the proposed method, DISCOVER and SPARK rank the same movie for top-1 result that contains the keyword "godfather". So, all answers in Table 15 are acceptable. When we test the next query "mile allen", both our proposed method and SPARK rank in top-1 result for this query. While DISCOVER can rank in top-6 result but it cannot rank in top-1 result. All answers are shown in Table 16, Table 17 and Table 18.

Table 15. Top-1 Answer on Query "godfather"

Method	Top-1 Answer
Proposed Method	AlBraggs → R ← Disco Godfather
DISCOVER	AlBraggs → R ← Disco Godfather
SPARK	AlBraggs → R ← Disco Godfather

Then, we test the query "gold anderson", all results are shown in Table 19, Table 20 and Table 21. For this query, DISCOVER ranks in top-4 result. Thus, SPARK rank in top-2 result. But the proposed method can rank in top-1 result and the next four answers are the other movies that are contain at least one keyword.

Moreover, we see that the proposed method can rank the top-1 result than DISCOVER and SPARK within all tested queries.

Table 16. Top-6 Answers on DISCOVER for Query "mile allen"

Rank	Top-6 Answers on DISCOVER
1	Arthur B.Allen → R ← Ebb Tide [1937] → MD ← James P. (I)Hogan
2	Allen 'Sugar Bear'Black → R ← Antone's: Home of the Blues [2004] → MD ← DanKarlok
3	BabkenAzizian → R ← Eyeball Eddie [2000] → MD ← Elizabeth (II)Allen
4	Bob (II)Burns → R ← Beyond the Rockies [1932] → MD ← Fred (II)Allen
5	Allen (I)Baron → R ← Blast of Silence [1961] → MD ← Allen (I)Baron
6	**Allen** (II)Adams → R ← 8 **Mile**

Table 17. Top-6 Answers on SPARK for Query "mile allen"

Rank	Top-6 Answers on SPARK
1	**Allen** (II) Adams → R ← 8 **Mile** [2002] → MD ← Curtis (I) Hanson
2	AbhiBhattacharya → R ← Dil De Mile Dil [1978] → MD ← BhishmKohli
3	BabkenAzizian → R ← Eyeball Eddie [2000] → MD ← Elizabeth (II) Allen
4	Bob (II) Burns → R ← Beyond the Rockies [1932] → MD ← Fred (II)Allen
5	Arthur B.Allen → R ← Ebb Tide [1937] → MD ← James P. (I) Hogan
6	Allen (I) Baron → R ← Blast of Silence [1961] → MD ← Allen (I) Baron

Table 18. Top-6 Answers on Proposed Method for Query "mile allen"

Rank	Top-6 Answers on Proposed Method
1	**Allen** (II) Adams → R ← 8 **Mile** [2002] → MD ← Curtis (I) Hanson
2	AbhiBhattacharya → R ← Dil De Mile Dil [1978] → MD ← BhishmKohli
3	Arthur B.Allen → R ← Ebb Tide [1937] → MD ← James P. (I) Hogan
4	Allen 'Sugar Bear'Black → R ← Antone's: Home of the Blues [2004] → MD ← DanKarlok
5	BabkenAzizian → R ← Eyeball Eddie [2000] → MD ← Elizabeth (II) Allen
6	Bob (II) Burns → R ← Beyond the Rockies [1932] → MD ← Fred (II) Allen

Table 19. Top-5 Answers on DISCOVER for Query "gold anderson"

Rank	Top-5 Answers on DISCOVER
1	Arthur (II)Anderson → R ← Deathdream [1974] → MD ← Bob (III)Clark
2	BarringtonBignall → R ← Exit Wounds [2001] → MD ← AndrzejBartkowiak
3	Anthony (I)Anderson → R ← Cradle 2 the Grave [2003] → MD ← AndrzejBartkowiak
4	AntonyCarrick → R ← Fields of **Gold** [2002] → MD ← Bill (III) **Anderson**
5	BobbyCanavarro → R ← Cleopatra Jones and the Casino of Gold [1975] → MD ← CharlesBail

Table 20. Top-5 Answers on SPARK for Query "gold anderson"

Rank	Top-5 Answers on SPARK
1	BobbyCanavarro → R ← Cleopatra Jones and the Casino of Gold [1975] → MD ← CharlesBail
2	AntonyCarrick → R ← Fields of **Gold** [2002] → MD ← Bill (III) **Anderson**
3	BillyBletcher → R ← Desert Gold [1936] → MD ← James P. (I) Hogan
4	Anthony (I) Anderson → R ← Cradle 2 the Grave [2003] → MD ← AndrzejBartkowiak
5	Arthur (II)Anderson → R ← Deathdream [1974] → MD ← Bob (III) Clark

Table 21. Top-5 Answers on Proposed Method for Query "gold anderson"

Rank	Top-5 Answers on Proposed Method
1	AntonyCarrick → R ← Fields of **Gold** [2002] → MD ← Bill (III) **Anderson**
2	BobbyCanavarro → R ← Cleopatra Jones and the Casino of Gold [1975] → MD ← CharlesBail
3	BillyBletcher → R ← Desert Gold [1936] → MD ← James P. (I) Hogan
4	Arthur (II) Anderson → R ← Deathdream [1974] → MD ← Bob (III) Clark
5	Anthony (I) Anderson → R ← Cradle 2 the Grave [2003] → MD ← AndrzejBartkowiak

In summary, the proposed ranking method can retrieve the relevant answers in order to two factors. Firstly, the content factor can calculate the scores to retrieves answers which match keywords for both AND semantic and OR semantic. Then, the structural factor can calculate the scores to retrieves the meaningful CTT. We observe that proposed ranking method achieve the better relevant results than the existing ranking methods by testing above the queries.

8.4 Efficiency of Query Processing

In this section, we illustrate the efficiency of query processing by measuring the computation time for sample queries of DBLP and IMDB datasets. Given a keyword query, the proposed algorithm generates the valid CNs. The generated CNs is evaluated by reducing the size of intermediate results to get the final results. The generated CTTs by evaluating CNs are produced as answers with the top-k query processing upon the highest score.

Figure 11 shows the execution times by evaluating the queries in DBLP. In this figure, Q4 and Q9 can execute the result at minimum time. Also Q1, Q2 and Q6 can evaluate the result queries at minimum time although there are larger numbers of queries than the number of queries in Q3. Thus, Q3 can produce the number of relevant queries no more than 10s.

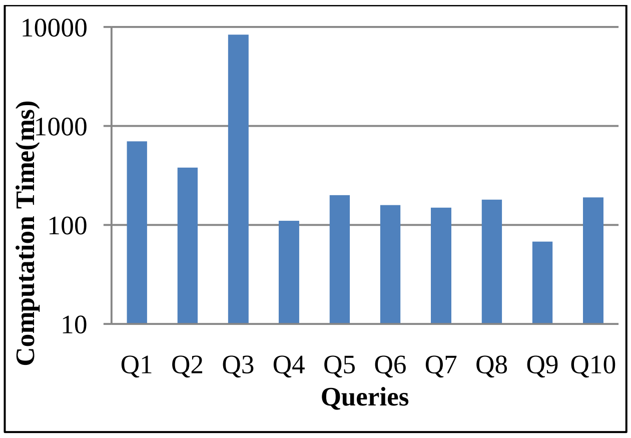


Figure 11. Execution Times for Queries of DBLP

Figure 12, we also present the evaluation of execution times for the queries in IMDB which is 6 times larger than DBLP. We can see that MQ5 can execute the result at minimum time. Also MQ2, MQ6, MQ7 and MQ9 can evaluate the result queries at minimum time although there are joined two or more relations due to the primary-foreign relationship. We can say that the execution time of MQ10 is efficient according to the data structure of IMDB dataset, although the computation time of this query is more than the other query. So, we observe that the top-k CTT algorithm execute the final result to speed up.

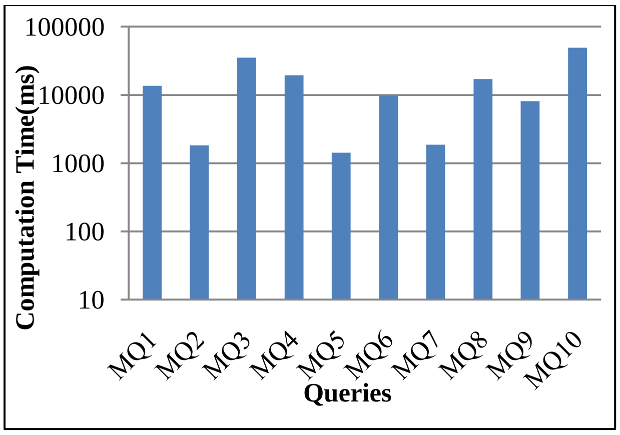


Figure 12. Execution Times for Queries of IMDB

9. CONCLUSION

A keyword search in relational databases allows ordinary users to find text information in relational databases with much higher flexibility. 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 retrieve relevant queries, we proposed a new ranking method based on the virtual document to calculate the score of CTT. The proposed ranking method can reduce the meaningless results which are disappointed for user with the ranking methods in previous works. The proposed Top-k CTT algorithm retrieved the top-k rank relevant results efficiently. We presented the experimental results on DBLP and IMDB show that the proposed method and algorithm generate the result approximately for the user desired query. And the experimental results are efficiently evaluated by using query execution strategy.

Efficiency of the system was presented with the computation times by evaluating the performance of all algorithms. Also, the relevant results on top-k value were presented by applying proposed ranking method as effectiveness. According to these experimental results, the system efficiency is affected as much as records that is causing the system overhead. To solve this problem, we consider to implement the more suitable query execution algorithms in future.

Acknowledgement

This research is partly supported by the EKSORDB System for Computer Science Research.

References

Aditya,V., Bhalotia,G., Chakrabarti,S., Hulgeri,A., Nakhe,C., Parag,S. (2002). BANKS: Browsing and keyword searching in relational databases. In VLDB, (p. 1083–1086).

Agrawal,S., Chaudhuri,S., Das,G. (2002). DBXplorer: A System for Keyword-Based Search over Relational Database. Proc. 18th Int. Conf. on Data Engineering, (p. 5-16).

Baid,A., Rae,I., Li,J., Doan,A., Naughton,J. (2010). Toward Scalable Keyword Search over Relational Data. Proc. VLDB Endowment, Vol. 3.

Ding,B., Yu,J.X., Wang,S., Qin,L., Zhang,X., Lin,X. (2007). Finding Top-k Min-Cost Connected Trees in Databases. In ICDE, (p. 836-845).

Hristidis,V., Papakonstaninou,Y. (2002). DISCOVER: Keyword Search in Relational Databases. Proc. 28th Int. Conf. on Very Large Data Bases, (p. 670-681).

Hristidis,V., Gravano,L., Papakonstantinou,Y. (2003). Efficient IR-Style Keyword Search over Relational Databases. In Proc. of the 29th VLDB Conference.

http://www.dblp.uni.trier.de.

http://www.imdb.com/interfaces.

Kashem,M.A., Chowdhury,A.S., Deb,R., Jahan,M. (2010) Query Optimization on Relational Databases for Supporting Top-k Query Processing Techniques. In JCIT, Vol. 1, p.53-58.

Kacholia,V., Pandit,S., Chakrabarti,S., Sudarshan,S., Desai,R., Karambelkar,H. (2005). Bidirectional expansion for keyword search on graph databases. In VLDB, (p. 505-516).

Kimelfeld,B., Sagiv,Y. (2006). Finding and Approximating Top-k Answers in Keyword proximity Search. In PODS, (p. 173-182).

Liu,F., Yu,C., Meng,W. (2006). Effective Keyword Search in Relational Databases. Proc. 2006 ACM SIGMOD Int. Conf. on Management of data, (p. 563-574).

Li,G., Feng,J., Lin,F., Zhou,L. (2008). Progressive ranking for efficient keyword search over relational databases. In Springer, Vol. 5071, (p.193-97).

Li.P, Zhu.Q., Wang.S. (2008). The Research on the Algorithms of Keyword Search in Relational Database. In Springer, (p.134-143).

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

Qin,L., Yu,J.X., Chang,L. (2009). Keyword Search in Databases: The Power of RDBMs. Proc. 35th SIGMOD Int. Conf. on Management of data, (p. 681-694).

Stefanidis,K., Drosou,M., Pitoura,E. (2010). PerK: Personalized Keyword Search in Relational Databases through Preferences. Proc. 13th Int. Conf. on Extending Database Technology, EDBT, (p. 585-596).

Thein,M.M. (2012). Improved Ranking Method for Keyword Queries on Relational Database. Int. Conference on Computer Applications, (p. 351-356).

Thein,M.M, Thwin,M.M.S. (2012). Efficient Schema Based Keyword Search in Relational Databases. Int. Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol. 2, (p. 13-32).

Wang,S., Zhang,J., Peng,Z., Zhan,J., Wang,Q. (2007). Study on Efficiency and Effectiveness of KSORD. APWeb/WAIM Int. Workshops, (p. 6-17).

Xu,Y., Ishikawa,Y., Guan,J. (2009). Effective Top-k Keyword Search in Relational Databases Considering Query Semantics. APWeb/WAIM Int. Workshops, (p. 172–184).

Xiaohui,Y., Huxia,S. (2012). Ranking keyword search results based on collective importance. In VLDB.

YU,J.X., Qin,L., Chang,L. (2010). Keyword Search in Relational Databases: A Survey. IEEE Data Engineering Bulletin, Vol. 33, (p. 67-78).

Zhang,N., IIyas,I.F., Ozsu,M.T. (2011). Universal Top-k Keyword Search over Relational Databases. Technical Report CS-2011-03.

Zeng.Z., Bao.Z., Ling.T.W., Lee.M.L. (2012). iSearch: An Interpretation based Framework for Keyword Search in Relational Databases”. In ACM, (p.3-9).