A Framework for Querying Relational Database using Keyword Search

Phyo Thu Thu Khine, Khin Nwe Ni Tun
University of Computer Studies, Yangon
phyothuthukhine@gmail.com, knntun@gmail.com

Abstract

Digital library (DL) research and development has concentrated primarily on collections and on the services to build and access them. And when the large quantity of Myanmar document is getting archived by the digital libraries, there is a need Myanmar Keyword Search system to easily search and retrieve these documents. Keyword search has been the most widely used kind of querying nowadays, especially for searching documents on the web because of its user-friendly way. Although, there are so many Keyword Search Systems over Relational Databases that exist on Internet, none of these can't fully support for searching with Myanmar language. Therefore we propose a system that can search and retrieve the document from the digital library by using Myanmar language keyword. At first, we translate keyword query into the available language in the digital library, and then execute them in RDBMS by using Hybrid Algorithm to retrieve top-k results.

1. Introduction

Without knowing the database schema or writing SQL queries, casual users or Web users can use Keyword Search over Relational Databases (KSORD) techniques to access databases in a fashion similar to using search engines to search the Web. In fact, the amount of information stored in Deep Web is 400 or 500 times larger than that in the visible Web. If database systems support keyword search, publishing or searching a database is expected to be simpler and easier in the web, and the deep web problem can be alleviated. However, keyword search techniques on the Web cannot directly be applied to databases because data on the Internet and database are in different forms. In databases, the information is viewed as data tables and their relationships, and query results may be a single tuple or joining tuples. Accordingly, the challenge is how to apply keyword-based search to find sorted relevant results in databases [3].

KSORD systems may be classified into two types according to their query processing mechanism. One is offline systems, which retrieve results for a keyword query from an intermediate representation generated by ”crawling” the database in advance, such as EKSO[20] or from some indexes created beforehand, such as ObjectRank [8] and ITREKS [19]. The other is online systems, which convert a keyword query into many SQL queries and retrieve the database itself. Furthermore, online KSORD systems can be classified into two types [1] according to their data model, schema-graph-based and data-graph-based. Data-graph-based Online KSORD (DO-KSORD) systems includes BANKS [4], BANKS II [5], NUITS[6,7], ObjectRank [8] and etc., while Schema-graph-based Online KSORD (S0-KSORD) systems include DBXplorer [9], DISCOVER [10], IR-Style [11] and SEEKER [12], SPARK [13], and etc.

The offline KSORD Systems execute queries relatively efficiently, but they can't query the up-to-date data in time, and also need a long preprocessing time to generate the intermediate representation and large physical space to store it. On the contrary, online KSORD systems can retrieve the latest data from the database, but the execution is usually inefficient because the converted SQL queries often contain many join operators as for SO-KSORD systems and the data graph search algorithms cannot scale to the number of query keywords and the size of data graph as for DO-KSORD systems.

As we all know, a wide number of languages are spoken by human beings in the world and most of the people prefer to have information in their own language. A person who is not particularly familiar with English should able to search with his native language. However, there are so many Keyword search over Relational Databases systems that exist on Internet, none of these can't fully support for searching with Myanmar language. For these reasons, our proposed system intends to assist searching and retrieving the document from the digital library by using Myanmar language keyword.

The rest of this paper is structured as follows: Section 2 briefly introduces related works. Section 3 introduces several basic definitions. Section 4 presents the framework for processing keyword-based queries and describes the architecture of the system. Section 5 concludes the paper.

2. Related Work

Querying using keywords is the most common method that is used today. Querying of a database relies on query languages that are inappropriate for end-users who have little experience with databases. There are many models of keyword-based querying in relational databases. The earlier survey [1] overviewed systems such as BANKS [4], DBXplorer [9], DISCOVER [10], and ObjectRank [8], and briefly summarized the key techniques from several aspects. They support free-form keyword search on relational databases and return tuple trees as answers for a given keyword query. One focus of the above works is to generate tuple trees efficiently. BANKS [4] finds all tuple trees from the data graph directly using a Steiner tree algorithm. In the data graph, they use PageRank style methods to assign weights to tuples and assign weights to edges between tuples.

DBXplorer [9] and DISCOVER [10] exploit the RDBMS schema, which leads to relatively efficient algorithms for answering keyword queries because the structural constraints expressed in the schema are helpful for query processing. However, all of them just assume AND semantics for an answer whereas our approach supports metadata queries with both AND and OR semantics. Hristidis et al. proposed the extension of DISCOVER that handles non metadata queries with both AND and OR semantics. Kacholia et al. [5] presented the bidirectional strategy to improve backward expanding search in BANKS by allowing forward search strategy. However, it still works by identifying Steiner trees from a whole graph. More recent approaches have been attentively proposed ranking methods. ObjectRank [8] uses an authority-based ranking strategy to keyword search in relational databases. It returns a set of the individual tuple as an answer. The ranking function is based on link analysis and term frequencies of query keywords.

Recent work focuses on brining more effective ranking from IR literatures and its related query processing methods. Specifically, [13] improves the ranking method in [11] by the following normalizations: tuple tree size normalization, refined document length normalization, document frequency normalization, and inter document weight normalization. Offline systems (such as EKSO [14]) usually preprocess the data to generate an intermediate representation for the database.

Keyword search over XML databases has also attracted interest recently [16, 17, 18]. Florescu et al. [16] extend XML query languages to enable keyword search at the granularity of XML elements, which helps novice users formulate queries. This work does not consider keyword proximity. Hristidis et al. [18] view an XML database as a graph of “minimal” XML segments and find connections between them that contain all the query keywords. They focus on the presentation of the results and use view materialization techniques to provide fast response times. Finally, XRANK [17] proposes a ranking function for the XML “result trees”, which combines the scores of the individual nodes of the result tree. The tree nodes are assigned PageRank-style scores off-line.

3. Framework of Keyword-Based Query

In this section, the detail processing of the proposed system is described with an example.

3.1. Query Model

Consider a database with n relations R_1, . .,R_n. Each relation R_i has m_i attributes ai_1,…,ai_{mi} , a primary key and possibly foreign keys into other relations. The schema graph $G_s(V,E)$, is a directed graph that captures the primary key to foreign key relationships in the database schema. G_s has a node in V for each relation R_i of the database and an edge $R_i \rightarrow R_j$ in E for each primary key in R_i to foreign key in R_j relationships. Figure 1 shows the schema graph of DBLP used in this paper.

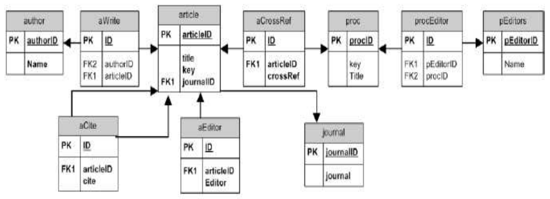
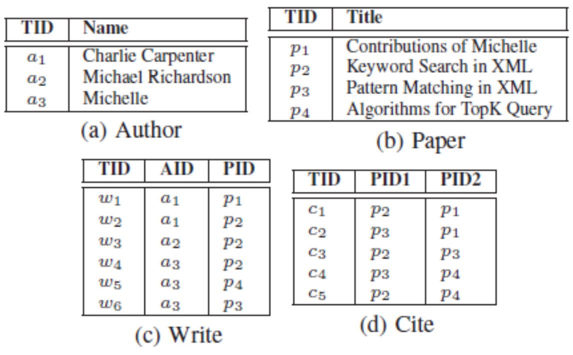


Figure 1: DBLP Schema Graph



TID	Name
a_1	Charlie Carpenter
a_2	Michael Richardson
a_3	Michelle

(a) Author

TID	Title
p_1	Contributions of Michelle
p_2	Keyword Search in XML
p_3	Pattern Matching in XML
p_4	Algorithms for TopK Query

(b) Paper

TID	AID	PID
w_1	a_1	p_1
w_2	a_1	p_2
w_3	a_2	p_2
w_4	a_3	p_2
w_5	a_3	p_4
w_6	a_3	p_3

(c) Write

TID	PID1	PID2
c_1	p_2	p_1
c_2	p_3	p_1
c_3	p_2	p_3
c_4	p_3	p_4
c_5	p_2	p_4

(d) Cite

Figure 2: DBLP Database Example

3.2. System Architecture

When a user keyword query comes (Figure 3), the keyword query is transliterated into the available language in the digital library. Then Tuple Set Creator creates tuple sets for each relation which has text attributes with full-text index, and only those non-empty tuple sets are left. Then, Candidate Network Generator outputs a complete and non-redundant set of Candidate Networks whose sizes are not greater than MaxCNsize through a breadth-first traversal of G_{ts}. Finally, CN executor runs a top-k algorithm to execute CNs to get top-k results for this query.

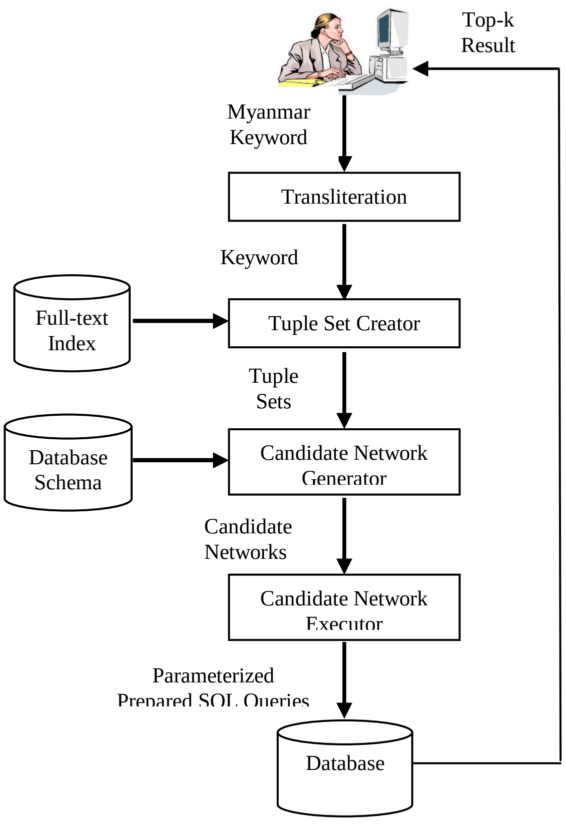


Figure 3: Architecture of proposed system

3.2.1. Query Processing

In this section, the components of query processing scheme (See Figure 2) and the structure of their inputs and outputs are described as follows.

Tuple Sets Creator

The Tuple Set Creator inputs a set of keywords $k_1,\ldots,k_m$ and outputs tuple sets for all subsets of keywords. A set of Tuple Sets (TSs) is the result sets created for a user keyword query by making use of the full-text search functionality of RDBMS. At most one Tuple Set can be created for a relation with text attributes and full-text indexes. Only those non-empty result sets are left, called as non-free tuple sets (nfTS).

Candidate Network Generator

The Candidate Network Generator inputs the set of keywords $k_1,\ldots,k_m$, the non-empty tuple sets R_i^K and the maximum candidate networks' size T and outputs a complete and non-redundant set of candidate networks. A breadth-first CN enumeration algorithm that is both sound and complete. The key challenge is to avoid the generation of redundant joining networks of tuple sets. In order to generate all candidate networks for an l-keyword query Q over an RDB with schema graph G_S, algorithms are designed to generate candidate networks $C = \{C_1, C_2\ldots\}$ that satisfy the following two conditions:

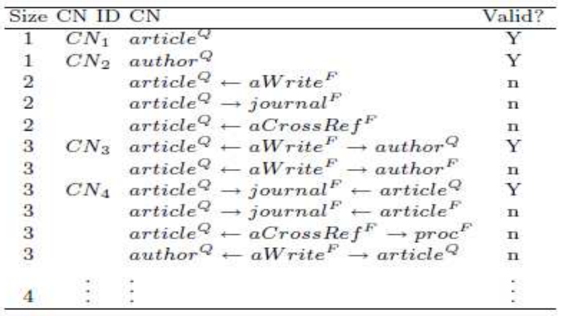
- Complete: For each solution *T* of the keyword query, there exists a candidate network $C_i \in C$ that can produce *T*.
- Duplication-Free: For every two CNs $C_i \in C$ and, *Ci* and *Cj* are not isomorphic to each other.

It can enumerate all the CNs of size no more than a specified number without violating any pruning rules. There are three pruning rules for partial CNs.

- Rule-1: Duplicated CNs are pruned (based on tree isomorphism).
- Rule-2: A CN can be pruned if it contains all the keywords and there is a leaf node, *Rj{K'}*, where $K' = \varnothing$, because it will generate results that do not satisfy the condition of minimality.
- Rule-3: When there only exists a single foreign key reference between two relation schemas (for xample, $Ri \rightarrow Rj$), CNs including $Ri\{K_1\} \rightarrow Rj\{K_2\} \leftarrow Ri\{K_3\}$ will be pruned, where K_1, K_2, and K_3 are three subsets of *Q*, and $Ri\{K_1\}$, $Rj\{K_2\}$, and $Ri\{K_3\}$ are keyword relations.

Example 1: Suppose a user wants to search papers written by "Ralf Steinmetz" with "p2p" in their titles from the DBLP3 database (its schema is shown in Figure 1). He might give a query containing two keywords: "p2p Steinmetz". Table1shows the generated candidate networks for query "p2p Steinmetz" in Example 1.

Table1. Enumerating CNs for query "p2p Steinmetz"



Size	CN ID	CN	Valid?
1	CN_1	$article^Q$	Y
1	CN_2	$author^Q$	Y
2		$article^Q \leftarrow aWrite^F$	n
2		$article^Q \rightarrow journal^F$	n
2		$article^Q \leftarrow aCrossRef^F$	n
3	CN_3	$article^Q \leftarrow aWrite^F \rightarrow author^Q$	Y
3		$article^Q \leftarrow aWrite^F \rightarrow author^F$	n
3	CN_4	$article^Q \rightarrow journal^F \leftarrow article^Q$	Y
3		$article^Q \rightarrow journal^F \leftarrow article^F$	n
3		$article^Q \leftarrow aCrossRef^F \rightarrow proc^F$	n
3		$author^Q \leftarrow aWrite^F \rightarrow article^Q$	n
4	⋮	⋮	⋮

Candidate Network Executor

Candidate Network Executor uses each Candidate Network (CN) and its corresponding tuple IDs in returned tuple sets and then translates CNs into parameterized prepared SQL queries to execute them in RDBMS to retrieve ranked results.

5. Conclusion

As the amount of information stored in databases is growing, all humanity such as students, scientists, researchers, reporters may want to find Myanmar documents relevant to their need. Thus keyword search systems over relational databases have become more and more important. In this paper, we have presented a framework for querying relational database based on keyword search. Keyword-based search system can allow search for combination of interesting terms without a-prior knowledge of the data schema and query language. By using the proposed processing scheme, the system will retrieve more relevant results that match the user needs. The proposed system in this paper will be applied efficiently and effectively in Myanmar language Keyword-based searching.

References

[1] S. Wang and K. Zhang, "Searching Databases with Keywords", Journal of Computer Science and Technology, 20(1). 2005:55-62.

[2] M.K.Bergman. "The deep web: Surfacing hidden value". White Paper, Bright Plannet, 2000.

[3] J. Saelee, and V. Boonjing, "A Metadata Search Approach to Keyword Search in Relational Databases", *ICCIT*, pages 571-576, 2008.

[4] Bhalotia, G., Hulgery, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: "Keyword Searching and Browsing in Databases using BANKS". In: Agrawal, R., et al. (eds.) Proc. of the 18th Int'l. Conf. on Data Engineering, pp. 431–440. IEEE Press, Los Alamitos (2002)

[5] Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: "Bidirectional Expansion for Keyword Search on Graph Databases". In: Böhm, K., et al. (eds.) Proc. of the 31st Int'l. Conf. on Very Large Data Bases, pp. 505–516. ACM, New York (2005).

[6] Wang, S., Peng, Z.H., Zhang, J., Qin, L., Wang, S., Yu, J., Ding, B.L.: "NUITS: A Novel User Interface for Efficient Keyword Search over Databases". In: Dayal, U., et al. (eds.) Proc. of the 32nd Int'l. Conf. on Very Large Data Bases, pp. 1143–1146. ACM, New York (2006).

[7] Ding, B.L., Yu, J., Wang, S., Qin, L., Zhang, X., Lin, X.M.: Finding Top-k Min-Cost Connected Trees in Databases. In: Proc. of the 23rd Int'l. Conf. on Data Engineering, pp. 836–845. IEEE Press, Los Alamitos (2007).

[8] Balmin, A., Hristidis, V., Papakonstantinou, Y.: "ObjectRank: Authority-Based Keyword Search in Databases". In: Nascimento, M.A., et al. (eds.) Proc. of the 30th Int'l. Conf. on Very Large Data Bases, pp. 564–575. Morgan Kaufmann Publishers, San Francisco (2004)

[9] Agrawal, S., Chaudhuri, S., Das, G.: "DBXplorer: A system for keyword-based search over relational databases". In: Agrawal, R., et al. (eds.) Proc. of the 18th Int'l. Conf. on Data Engineering, pp. 5–16. IEEE Press, Los Alamitos (2002).

[10] V. Hristidis, Y. Papakonstantinou: "DISCOVER: Keyword Search in Relational Databases". VLDB, 2002:670-681.

[11] V. Hristidis, L. Gravano, Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases", VLDB, 2003:850-861.

[12] Wen, J.J., Wang, S.: "SEEKER: Keyword-based Information Retrieval over Relational Databases". Journal of Software 16(7), 1270–1281 (2005).

[13] Luo, Y., Lin, X.M., Wang, W., Zhou, X.F.: "SPARK: Top-k Keyword Query in Relational Databases". In: Chan, C.Y., et al. (eds.) Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 115–126. ACM, New York (2007).

[14] S. Qi, W. Jennifer, "Indexing Relational Database Content Offline for Efficient Keyword-Based Search", Proceeding of IDEAS, 2005:297-306.

[15] G. Salton and M. McGill. "Introduction to Modern Information Retrieval". McGraw-Hill, 1983

[16] D. Florescu, D. Kossmann, and I. Manolescu. "Integrating keyword search into XML query processing". In WWW9, 2000.

[17]L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. "XRANK: Ranked keyword search over XML documents". In ACM SIGMOD,2003.

[18] V. Hristidis, Y. Papakonstantinou, and A. Balmin. "Keyword proximity search on XML graphs". In ICDE, 2003.

[19] J. Zhan, S. Wang. "ITREKS: Keyword Search over Relational Database by Indexing Tuple Relationship". 12th International Conference on Database Systems For Advance Applications (DASFAA), 2007.

[20] S. Qi, W. Jennifer. "Indexing Relational Database Content Offline for Efficient Keyword-Based Search". IDEAS, 2005:297-306.