

Efficient Keyword Search Methods In Relational Databases

Sharmili C¹, Rexie J. A. M²

¹Post-Graduate Student, Department Of Computer Science And Engineering, Karunya University, India

²Assistant Professor, Department Of Computer Science And Engineering, Karunya University, India

Abstract

Now a days the keyword search is a mechanism used by all type of organizations. In relational databases the keyword search is used to find the tuples by giving queries. But most of the methods that contain low performance and more storage space for storing the results. So we must find the efficient method for keyword search in relational databases. So we go for the comparative study for the keyword search in relational databases.

Keywords: relational databases, keyword search, IR ranking, banks, discover.

I. INTRODUCTION

Data mining is the process that attempts to discover patterns in large data sets. It utilizes methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (association rule mining). This usually involves using database techniques such as spatial indexes. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection and data preparation, nor result interpretation and reporting are part of the data mining step, but do belong to the overall KDD process as additional steps.

Keyword search is a proven and widely accepted mechanism for querying in textual document systems and the World Wide Web. The database research community has recently recognized the benefits of keyword search and has been introducing keyword-search capabilities into relational databases, XML databases, graph databases, and heterogeneous data sources.

Keyword search provides an alternative means of querying relational databases, which is

simple to people who are familiar with using web search engines. One important advantage of keyword search is that it enables users to search for information without having to know complex structured query languages (e.g., SQL) or prior knowledge about the structures of the underlying data.

Keyword search over relational databases finds the answers of the tuples in the databases which are connected through primary/foreign keys and contain query keywords. Keyword search provides a simple and user-friendly query interface to access the data in web and scientific applications.

Keyword search is considered to be an effective information discovery method for structure and semi-structured data. It allows users without prior knowledge of schema and query language to search. The database research community has recently recognized the benefits of keyword search and has been introducing keyword-search capabilities for effective keyword search.

Keyword search over relational databases finds the answers of tuples in the databases which are connected through primary/foreign keys and contain query keywords. Existing three types of methods: candidate-network-based methods, Steiner-tree-based algorithms, and Backward expanding keyword search approaches. These methods are explained in the comparative study.

II. KEY CONCEPTS

A. Keyword searching using banks

Relational databases are commonly searched using structured query languages. The user needs to know the data schema to be able to ask suitable queries. Search engines on the Web have popularized an alternative unstructured querying and browsing paradigm that is simple and user-friendly. Users type in keywords and follow hyperlinks to navigate from one document to the other.

No knowledge of schema is needed. In relational databases, information needed to answer a keyword query is often split across the tables/tuples, due to normalization. The BANKS system enables data and schema browsing together with keyword-based search for relational databases.

BANKS enables a user to get information by typing a few keywords and interacting with, controls on the results; absolutely no query language or programming is required. BANKS greatly reduces

the effort involved in publishing relational data on the Web and making it searchable.

B. Discover: keyword search in relational databases

DISCOVER returns qualified joining networks of tuples, that is, sets of tuples that are associated because they join on their primary and foreign keys and collectively contain all the keywords of the query. DISCOVER proceeds in two steps. First the Candidate Network Generator generates all candidate networks of relations, that is, join expressions that generate the joining networks of tuples. Then the Plan Generator builds plans for the efficient evaluation of the set of candidate networks, exploiting the opportunities to reuse common sub expressions of the candidate networks. DISCOVER provides a simple interface where the user simply types the keywords as would do on a search engine.

According to DISCOVER, an association exists between two keywords if they are contained in two associated tuples, i.e., two tuples that join through foreign key to primary key relationships, which potentially involve more tuples. As the amount of information stored in databases increases, so does the need for efficient information discovery. Keyword search enables information discovery without requiring from the user to know the schema of the database.

C. IR Ranking

With the amount of available text data in relational databases growing rapidly, the need for ordinary users to search such information is dramatically increasing. Even though the major RDBMSs have provided full-text search capabilities, they still require users to have knowledge of the database schemas and use a structured query language to search information.

This search model is complicated for most ordinary users. Inspired by the big success of information retrieval (IR) style keyword search on the web, keyword search in relational databases has recently emerged as a new research topic.

The differences between text databases and relational databases result in three new challenges: (1) Answers needed by users are not limited to individual tuples, but results assembled from joining tuples from multiple tables are used to form answers in the form of tuple trees. (2) A single score for each answer (i.e. a tuple tree) is needed to estimate its relevance to a given query. These scores are used to rank the most relevant answers as high as possible. (3) Relational databases have much richer structures than text databases. Existing IR strategies are inadequate in ranking relational outputs. In this paper, we propose a novel IR ranking strategy for effective keyword search.

III. COMPARATIVE STUDY

This section includes a study on some of the algorithm that searches the tuples from the database but it not efficient in their performance.

A. Backward expanding search

Backward Expanding Search algorithm offers a heuristic solution for incremental computing query results. Assume that the graph fits in memory. This is not unreasonable, even for moderately large databases, because the in-memory node representation need not store any attribute of the corresponding tuple other than the RID. The only other in-memory structure is an index to map RIDs to the graph nodes. Indices to map keywords to RIDs can be disk resident. As a result the graphs of even large databases with millions of nodes and edges can fit in modest amounts of memory. BANKS models tuples as nodes in a graph, connected by links induced by foreign key and other relationships. Answers to a query are modeled as rooted trees connecting tuples that match individual keywords in the query. Answers are ranked using a notion of proximity coupled with a notion of prestige of nodes based on inlinks, similar to techniques developed for Web search. It presents an efficient heuristic algorithm for finding and ranking query results.

B. Candidate Network Generator

The Candidate Network Generator inputs the set of keywords k_1, \dots, k_m , the non-empty tuple sets R_{k_i} and the maximum candidate networks' size T and outputs a complete and non-redundant set of candidate networks. The key challenge is to avoid the generation of redundant joining networks of tuple sets. The solution to this problem requires an analysis of the conditions that force a joining network of tuples to be non-minimal the condition for the totality of the network is straightforward.

As the amount of information stored in databases increases, so does the need for efficient information discovery. Keyword search enables information discovery without requiring from the user to know the schema of the database, SQL or some QBE-like interface, and the roles of the various entities and terms used in the query. In DISCOVER databases that do not require knowledge of the database schema or of a querying language, DISCOVER is a system that performs keyword search in relational databases. It proceeds in three steps. First it generates the smallest set of candidate networks. Then the greedy algorithm creates a near-optimal execution plan to evaluate the set of candidate networks. Finally, the execution plan is executed by the DBMS.

C. Steiner-tree-based search

A relational database can be modeled as a database graph $G = (V, E)$ such that there is a one-to-one mapping between a tuple in the database and a

node in V . G can be considered as a directed graph with two a edge: a forward edge $(u, v) \in E$ iff there is a foreign key from u to v , and a back edge (v, u) iff (u, v) is a forward edge in E . An edge (u, v) indicates a close relationship between tuples u and v (i.e., they can be directly joined together), and the introduction of two edge types allows differentiating the importance of u to v and vice versa. When such separation is not necessary for some applications, G becomes an undirected graph.

To support keyword search over relational data, G is typically modeled as a weighted graph, with a node weight $\omega(v)$ to represent the “prestige” level of each node $v \in V$ and an edge weight $\omega(u, v)$ for each edge in E to represent the strength of the closeness relationship between the two tuples.

Most of existing methods of keyword search over relational databases find the Steiner trees composed of relevant tuples as the answers. They identify the Steiner trees by discovering the rich structural relationships between tuples, and neglect the fact that such structural relationships can be pre-computed and indexed. Tuple units that are composed of most relevant tuples are proposed to address this problem. Tuple units can be precomputed and indexed. Existing methods identify a single tuple unit to answer keyword queries. They, however, may involve false negatives as in many cases a single tuple unit cannot answer a keyword query. Instead, multiple tuple units should be integrated to answer keyword queries.

To address this problem, in this paper, we study how to integrate multiple related tuple units to effectively answer keyword queries. It devises novel indices and incorporates the structural relationships between different tuple units into the indices. It uses the indices to efficiently and progressively identify the *top-k* relevant answers. It have implemented our method in real database systems, and the experimental results show that our approach achieves high search efficiency and accuracy, and outperforms state-of-the-art methods significantly.

D. Joined Tuple Tree Algorithm

In this paper, it describes the effectiveness and the efficiency issues of answering top-k keyword query in relational database systems. They propose a new ranking formula by adapting existing IR techniques based on a natural notion of virtual document. Compared with previous approaches, our new ranking method is simple yet effective, and agrees with human perceptions. It has conducted extensive experiments on large-scale real databases using two popular RDBMSs.

It focuses on the problem of supporting effective and efficient top-k keyword search in relational databases. While many RDBMSs support full-text search, they only allow retrieving relevant tuples from within the same relation.

A unique feature of keyword search over RDBMSs is that search results are often assembled from relevant tuples in several relations such that they are inter-connected and collectively be relevant to the keyword query [1], [3]. Supporting such feature has a number of advantages. Firstly, data may have to be split and stored in different relations due to database normalization requirement. Such data will not be returned if keyword search is limited to only single relations. Secondly, it lowers the barrier for casual users to search databases, as it does not require users to have knowledge about

Query languages and database schema. Thirdly, it helps to reveal interesting or unexpected relationships among entities [19]. Lastly, for websites with database back-ends, it provides a more flexible search method than the existing solution that uses a fixed set of pre-built template queries.

E. Tastier Approach

In this paper the proposed a novel approach to keyword search in the relational world, called Tastier. A Tastier system can bring instant gratification to users by supporting type-ahead search, which finds answers “on the fly” as the user types in query keywords.

A main challenge is how to achieve a high interactive speed for large amounts of data in multiple tables, so that a query can be answered efficiently within milliseconds. It proposes efficient index structures and algorithms for finding relevant answers on-the-fly by joining tuples in the database. It devises a partition-based method to improve query performance by grouping relevant tuples and pruning irrelevant tuples efficiently. It also develops a technique to answer a query efficiently by predicting highly relevant complete queries for the user. It has conducted a thorough experimental evaluation of the proposed techniques on real data sets to demonstrate the efficiency and practicality of this new search paradigm.

IV. FUTURE RESEARCH DIRECTIONS

Emphasize on the efficiency and effectiveness of keyword search over relational databases by shortening and indexing tuples in the fundamental relational databases. It proposes the concept of tuple units to effectively answer keyword queries. It generates and materializes the tuple units, which are composed of relevant tuples connected by primary foreign- key relationships. They identify the most relevant and meaningful tuple units to answer keyword queries. Moreover, they examine the techniques of indexing and ranking to enhance the search efficiency and search accuracy.

A tuple unit is a set of highly relevant tuples which contain query keywords. Moreover tuple units can be precomputed and indexed, and it can use the indexed tuple units to efficiently answer a keyword query.

Propose a structure-aware index based method to integrate multiple related tuple units to effectively answer keyword queries. It discovers the structural relationships between different tuple units and stores them into structure-aware indices, and progressively finds the top-k answers using such indices. Propose a novel method, namely Saint (Structure-Aware INDEXing for finding and ranking Tuple units), to answer keyword queries over relational databases.

Devise two structure-aware indexes, single-keyword based structure-aware index (SKSA-Index) and keyword-pair based structure-aware index (KPSA-Index), to capture structural relationships between tuple units. We use the indexes to on-the-fly integrate multiple tuple units to answer a keyword query.

Keyword query $K=(k_1; k_2; \dots ; k_n)$ and a relational database, first identify the tuple units with the top-k highest scores (using the scoring function $SCORE(K,u)$). Then, construct the subtrees which are rooted at each identified tuple unit and contain the paths from the tuple unit to the corresponding pivotal tuple units for each keyword.

Finally, take the subtrees as the answers. Note that the subtree is composed of multiple highly relevant tuple units. Indexing in keyword search is used to find the keywords in the tuples in an efficient manner.

To efficiently retrieve IR scores, propose a single keyword- based structure-aware index, called SKSA-Index, which is similar to the traditional inverted index. The entries of SKSA-Index are also the keywords that are contained in the underlying database.

Different from inverted indexes which only maintain the tuple units that directly contain the keyword, each entry of SKSA-Index preserves the tuple units that directly or indirectly contain the keyword in the form of a triple $\langle \text{TupleUnit}, \text{Score}, \text{TupleUnitLists} \rangle$, where the Score is the assigned score of the keyword in the tuple unit TupleUnit, and TupleUnitLists preserves the tuple unit lists from Unit to the corresponding pivottupleunits, which can be obtained from the pivotal tuple unit matrix.

The tuple units are sorted by the corresponding scores in descending order. Most importantly, SKSA-Index captures the rich structural relationships, as each entry preserves the paths from a given tuple unit to the corresponding pivotal tuple unit and also keeps the structure-aware scores of tuple units indirectly or directly containing keywords.

V. CONCLUSION


The above comparative study algorithm are not efficient because of the disadvantages. These algorithms needed more space for storing the information and it also complex for finding the tuples from the relational database.

The problem of effective keyword search over relational databases is the major issue in the databases. Here proposes to integrate multiple relevant tuple units to effectively answer keyword queries.

It is devised two novel structure-aware indexes, SKSA-Index and KPSA-Index, which incorporate the structural relationships between tuple units and the textual relevancy between input keywords into the indexes. It is suggested a novel ranking mechanism by taking into consideration both the textual relevancy in IR literature and the structural compactness of tuple units from the DB viewpoint.

REFERENCES

- [1] Jianhua Feng, Guoliang Li, and Jianyong Wang, "Finding Top-k Answers in Keyword Search over Relational Databases Using Tuple Units" IEEE Transactions On Knowledge And Data Engineering, Vol. 23, No. 12, December 2011.
- [2] Balmin.A., Hristidis.V., and Papakonstantinou.Y. (2004) "Objectrank: Authority-Based Keyword Search in Databases," Int'l Conf. Very Large Data Bases (VLDB), 564-575.
- [3] Bhalotia.G., Hulgeri.A., Nakhe.C., Chakrabarti.S., and Sudarshan.S. (2002) "Keyword Searching and Browsing in Databases Using Banks," Int'l Conf. Data Eng. (ICDE), 431-440.
- [4] Ding.B et al. (2007) "Finding Top-k Min-Cost Connected Trees in Databases," IEEE Int'l Conf. Data Eng. (ICDE).
- [5] Feng.J., Li.G., Wang.J, and Zhou.L. (2010) "Finding and Ranking Compact Connected Trees for Effective Keyword Proximity Search in XML Documents," Information Systems, vol. 35, no. 2, 186-203.
- [6] He.H., Wang.H., Yang.J., and Yu.P. (2007) "Blinks: Ranked Keyword Searches on Graphs," ACM SIGMOD Int'l Conf. Management of Data.
- [7] Hristidis.V., and Papakonstantinou.Y. (2002a) "Discover: Keyword Search in Relational Databases," Int'l Conf. Very Large Data Bases (VLDB), 670-681.
- [8] Li.G., Ji.S., Li.C., and Feng.J. (2009a) "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," SIGMOD Int'l Conf. Management of Data, 695-706.
- [9] Li.G., Ooi B.C., Feng.J., Wang.J., and Zhou.L. (2008b) "Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," ACM SIGMOD Int'l Conf. Management of Data, 903-914.

- 
- [10] Li.G., Zhou.X., Feng.J., and Wang.J. (2009c) "Progressive Keyword Search in Relational Databases," IEEE Int'l Conf. Data Eng. (ICDE),1183-1186.
- [11] Liu.F., Yu.C., Meng.W., and Chowdhury.A. (2006) "Effective Keyword Search in Relational Databases," ACM SIGMOD Int'l Conf. Management of Data, 563-574.
- [12] Luo.L., Lin.X., Wang.W., and Zhou.X. (2007) "Spark: Top-k Keyword Query in Relational Databases," ACM SIGMOD Int'l Conf. Management of Data.
- [13] Li.G., Feng.J., and Zhou.L. (2008) "Retune: Retrieving and Materializing Tuple Units for Effective Keyword Search over Relational Databases," Int'l Conf. Conceptual Modeling (ER), 469- 483.
- [14] Li.G., Feng.J., and Wang.J. (2009a) "Structure-Aware Indexing for Keyword Search in Databases," ACM Conf. Information and Knowledge Management (CIKM),1453-1456.
- [15] Li.G., Feng.J., Zhou.X., and Wang.J (2011b) "Providing Built-in Keyword Search Capabilities in RDBMS," The VLDB J., vol. 20, no. 1,1- 19.
- [16] Markowetz.A., Yang.Y., and Papadias.D. (2007) "Keyword Search on Relational Data Streams," ACM SIGMOD Int'l Conf. Management of Data.
- [17] Qin.L., Yu.J.X., and Chang.L. (2009) "Keyword Search in Databases: The Power of RDBMS," SIGMOD Int'l Conf. Management of Data, 681-694.
- [18] Sayyadian.M., LeKhac.H., Doan.A., and Gravano.L. (2007) "Efficient Keyword Search across Heterogeneous Relational Databases," IEEE Int'l Conf. Data Eng. (ICDE).
- [19] Yu.B., Li.G., Sollins.K., and Tung.A.K.H. (2007) "Effective Keyword- Based Selection of Relational Databases," ACM SIGMOD Int'l Conf. Management of Data, 139-150.