GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

LITERATURE SURVEY ON RELATIONAL KEYWORD SEARCH SYSTEMS

Divya Bharathi N
PG Scholar, Department of CSE, Adhiyamaan College of Engineering, Hosur, India

ABSTRACT

Data mining refers to the extraction of knowledge from large dataset. Information Retrieval is one of the popular data mining techniques which obtaining the information resources relevant to an information need from a collection of information resources. Keyword search is the most popular information retrieval method. A large number of approaches have been proposed and implemented, there remains a severe lack of standardization for system evaluations which resulted in contradictory results from different evaluations. In this paper I surveyed different approaches in relational keyword search system to design a new Keyword search system over relational databases which will provide improved execution and response time.

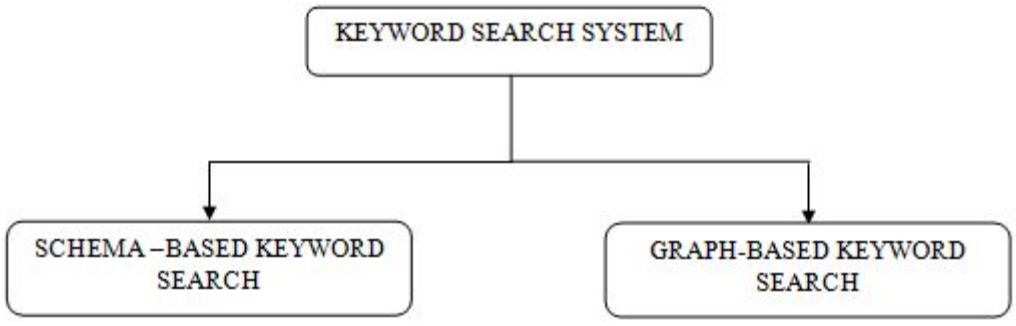
Keywords- *Data mining, Information retrieval, Keyword search, Relational Databases.*

I. INTRODUCTION

Web is one of the main sources of information. The amount of information in the web is increasing exponentially. Search engines provide the interface to access the information from web. Users retrieve their relevant information through the input query which does not prove to be effective as input query entered by the user. To retrieve the information according to a particular information need from a pool of information available on the web is a big challenge. Keyword search is an important concern related to Information retrieval. It has proven to be an effective method to discover and retrieve information online as evidenced by the success of Internet search engines. Unfortunately, many common information management systems do not support the familiar keyword search interface that people now expect. Keyword search over relational databases has recently received significant attention. Web sites, corporations, and governments all use relational databases to manage information, but keyword search in relational databases is difficult due to data transformations that eliminate redundancy and ensure consistency. Relational keyword search enables users to retrieve information and to explore the relationships among that information all via a familiar interface. Current Keyword search systems have unpredictable running times. For a given queries it takes too long to produce answers, and for others the system may even fail to return.

II. RELATIONAL KEYWORD SEARCH

Keyword searching is an effective method for finding information in any computerized database. It can be classified into two types, one is schema based keyword search and other is graph based key word search. Keyword search has been applied to retrieve useful data in documents, texts, graphs, and even relational databases. In Relational keyword search(R-KWS), the basic unit of information is a tuple/record. In contrast to Keyword search on documents, results in Relational keyword search cannot simply be found by inspecting units of information (records) individually. Instead, results have to be constructed by joining tuples. R-KWS has benefits over SQL queries. First, it frees the user from having to study a database schema. Second, R-KWS allows querying for terms in unknown locations (tables/attributes). Finally, a single R-KWS query replaces numerous complex SQL statements.



BANKS [1], and DISCOVER [4] were the first systems that supported keyword search over relational databases. As more structured data becomes available at organizations and on the Web, and as more untrained users want to use such data, we expect that more efforts will be devoted to building such systems in the near future.

A. Graph-Based Keyword Search

Graph-based approaches assumed that the database is modeled as a weighted graph where the importance of relationships is indicated by the weights of the edges. Here, the relational database is explicitly converted to a graph. Tuples are act as vertices in the graph, and edges denote foreign keys. In this approach the results of the keyword queries are modeled as trees that connect nodes which matching the keywords.

Keyword searching in BANKS [1] is done using proximity based ranking, based on foreign key links and other types of links. It operates on data graph where each tuple is a node and each foreign key relationship between tuples is represented as bidirectional edge. It performs Backward Expanding search, starting at nodes matching keywords and working up toward confluent roots, is commonly used for predominantly text-driven queries. But it can perform poorly if some keywords match many nodes, or some node has very large degree.

BANKS-II [2] presented the bidirectional strategy to improve the efficiency of keyword search over graph data, which uses both forward and backward expansion. However, their method still works by identifying Steiner trees from the whole graph, which is inefficient as it is rather difficult to identify structural relationships through inverted indices because bidirectional expansion may miss some shortest paths. It performs well for a variety of keyword queries, but its performance significantly degrades in the presence of high-degree nodes during the expansion process.

BLINKS[3] is a graph based approach avoids the NP-hard Steiner tree problem by giving up on completeness of answers. Specifically, it uses “distinct root” semantics and only explores a portion of the search space. This allows for efficient answer generation at the cost of some coverage and greatly improves the run-time performance. BLINKS[3] also uses data partitioning in addition to bidirectional search proposed in BANKS[1]. This system relies heavily on the ranking function used and performance guarantees cannot be made if the ranking-function is a black-box. The system returns only the roots of the answers and their distances from each keyword query. Reconstructing the answer trees from this information requires extra work. Additionally, the graph and bi-level index used in BLINKS must fit in memory for BLINKS to be efficient.

B. Schema-Based Keyword Search

Schema based techniques offers keyword based search over relational database by directly executing the SQL commands. These approaches model the relational schema as a graph where relational tables are act as vertices and foreign keys between tables are act as edges. Query processing follows three phases. First, database tuples that contain search terms are identified. Second, candidate networks (SQL expressions) that could relate these tuples are enumerated systematically. Third, the execution engine executes these SQL expressions to identify results, which are then returned to the user. Because there are many possible ways to relate the search terms, the execution engine is generally responsible for selecting only the most promising SQL expressions. The remainder need not be executed once the top-k highest-ranking results are known.

DISCOVER [4] use 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, but it is limited to Boolean AND semantics for queries which only considers conjunctive semantics. It requires that all query keywords appear in the tree of nodes or tuples that are returned as the answer to a query.

DISCOVER-II [5] considers the problem of keyword proximity search in terms of disjunctive semantics, It can handle queries with both AND and OR semantics, and exploits the sophisticated single-column text-search functionality often available in commercial RDBMSs. DISCOVER-II [5] uses three algorithms, namely, the Sparse algorithm, the single-pipelined algorithm, and the Global pipelined algorithm to find a proper order of generating Minimal total joining network of tuples .All algorithms are based on the attribute level ranking function which has the property of tuple monotonicity. single/global pipeline algorithm may incur many unnecessary join checking.

To handle non-monotonic score functions SPARK [6] have been proposed. It makes use of two algorithms, namely, Skyline-sweeping and Block-pipelined algorithm. Skyline sweeping has the minimal number of accesses to the database and does not perform any unnecessary checking. To improve the performance of Skyline sweeping algorithm, Block-pipelined algorithm was implemented.

III. CONCLUSION

In this article, I surveyed some main results on finding structural information in Relational database for keyword searching. The current work will focus on Kruskal's algorithm that finds the minimum spanning tree which leads to provide improved execution and response time.

REFERENCES

1. *G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword Searching and Browsing in Databases using BANKS," in Proceedings of the 18th International Conference on Data Engineering, ser. ICDE '02, February 2002, pp. 431–440.*
2. *V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional Expansion For Keyword Search on Graph Databases," in Proceedings of the 31st International Conference on Very Large Data Bases, ser. VLDB '05, August 2005, pp. 505–516.*
3. *H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: Ranked Keyword Searches on Graphs," in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '07, June 2007, pp. 305–316.*
4. *V. Hristidis and Y. Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases," in Proceedings of the 29th International Conference on Very Large Data Bases, ser. VLDB '02. VLDB Endowment, August 2002, pp. 670–681.*
5. *V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-style Keyword Search over Relational Databases," in Proceedings of the 29th International Conference on Very Large Data Bases, ser. VLDB '03, September 2003, pp. 850–861.*
6. *Y. Luo, X. Lin, W. Wang, and X. Zhou, "SPARK: Top-k Keyword Query in Relational Databases," in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '07, June 2007, pp. 115–126.*