KEYWORD SEARCH IN RELATIONAL DATABASES

N.Divya Bharathi[1]

[1]*PG Scholar, Department of Computer Science and Engineering,*

Adhiyamaan College of Engineering, Hosur, (India).

ABSTRACT

Data mining refers to extraction of data from the large amount of information in database. Information retrieval is a popularized method related to data mining. Keyword search is an important technique in information retrieval method. Nowadays, social networking and blogging stores large amount of data in relational database. Relational database is in the form of table which consists of set of attributes and tuples. To searching in relational databases user need to know query language and database schema Most traditional system offers only query language but the proposed system is based on keyword search over relational database which is useful for naïve user to retrieve relevant information from relational database. The proposed system makes use of two algorithms that are candidate network generation and plan generation that leads to provide improved execution and response time.

Keywords-: Database, Database Schema, Information Retrieval, Query Language, Relational Database

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. RELATED WORK

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.

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.

A BLINKS [3] is a top-k keyword search query on a graph finds the top k answers according to ranking criteria. It is a graph based approach avoids the NP- hard Steiner tree problem by giving up on completeness of answers. It makes use of Bi-level indexing algorithm. 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.

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. 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. Keyword search can be classified into two types. One is schema based approach, other is graph based approach.

3.1 Schema-Based Approach

Schema-based search techniques support keyword search over relational databases via direct execution of SQL commands. These techniques model the relational schema as a graph where vertices are relational tables and edges denote foreign keys between tables. 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 systematically enumerated. Third, these SQL expressions are executed against the database to identify results, which are returned to the user. Because there are many possible ways to relate the search terms, efficient query processing precludes executing all possible SQL expressions. Instead, only the most promising SQL expressions are actually executed against the database; the remainders are ignored once the top-k results are known.

3.2 Graph-Based Approach

Graph-based approaches assume the database is modeled as a weighted graph where the weights of edges indicate the importance of relationships. Proximity search strategies attempt to minimize the weight of result trees. Graph-based search techniques are more general than schema-based approaches, for relational databases, XML, and the Internet can all be modeled as graphs. None of the graph-based search techniques described in the literature operates on the database itself. Instead, the relational database is explicitly converted to a graph. Tuples become vertices in the graph, and edges denote foreign keys. Most search techniques also distinguish edges induced by foreign keys from “backward"edges,which connect the same vertices but reverse the edge's direction. The addition of the backward edges allows directionality to be considered; weighting the backward edges higher than the “forward" edges discourage the inclusion of common relationships that users would find uninformative.

IV. PROPOSED WORK

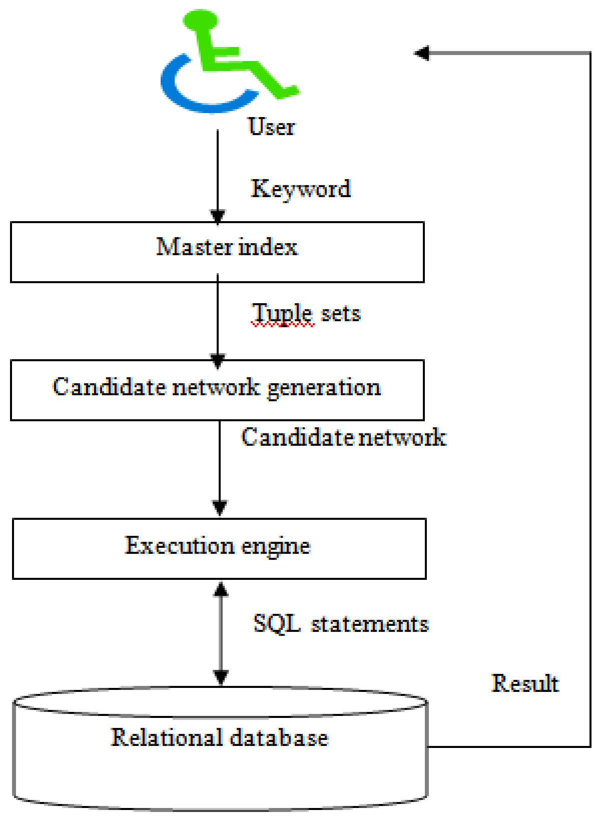

Fig1 Keyword Search System Architecture

4.1 Description of the Proposed Architecture

A high level representation of the Keyword search system architecture uses to find the joining networks is shown in figure 1. The description of the architecture is as follows,

- User enters the set of keywords
- The keywords are looked up into the master index and returns the tuple sets for each relation.
- Candidate network generator generates all candidate networks of relations, that is join expressions can be generated by joining the tuples using foreign key relationship.
- Plan generator produces on execution plan by evaluating the candidate network.
- Finally, the SQL statement is produced for each line of the execution plan then the SQL statements are passed to the relational database.
- The database returns the result based on the entered keyword.

4.2 Master Index

The master index applications are designed to uniquely identify, match, and maintain information in the data base. The master index maintains a centralized database to enabling the integration of data records.

4.3 Candidate Network Generation

Candidate network generator generates all candidate networks of relations, that is join expressions can be generated by joining the tuples using foreign key relationship.

4.4 Execution Engine

Execution engine is also called as plan generator which evaluates the candidate network to provide an execution plan and creates the SQL statement based on candidate network.

V. CONCLUSION

The proposed system is based on keyword search over relational database which enables the naïve user to retrieve the information from relational database without any knowledge of query language and database schema. The developed system is based on schema based approach that enables direct execution of SQL queries. The future work focus on graph based approach that will use kruskal’s algorithm which will be expected to provide less 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.

[7] A. Baid, I. Rae, J. Li, A. Doan, and J. Naughton, "Toward Scalable Keyword Search over Relational Data," Proceedings of the VLDB Endowment, vol. 3, no. 1, pp. 140–149, 2010.

[8] S. Chaudhuri and G. Das, "Keyword Querying and Ranking in Databases," Proceedings of the VLDB Endowment, vol. 2, pp. 1658–1659, August 2009.

[9] Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword Search on Structured and Semi-Structured Data," in Proceedings of the 35th SIGMOD International Conference on Management of Data, ser. SIGMOD'09, June 2009, pp. 1005–1010.

[10] J. Coffman and A. C. Weaver, "A Framework for Evaluating Database Keyword Search Strategies," in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ser. CIKM '10, October 2010, pp. 729–738.