ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Implementation of Efficient Keyword Search in Relational Databases

Dr.M.Mayilvaganan, K.Monika

Associate Professor, Dept. of Computer Science, PSG College of Arts and Science, Coimbatore, India

Research Scholar, Dept. of Computer Science, PSG college of Arts and Science, Coimbatore, India

ABSTRACT: It is difficult for typical web users to exploit web data by means of structured queries using languages like SQL. To end this keyword search has proven intuitive. Keyword search is a familiar and potentially effective way to find information of interest that is "locked" inside databases. Keyword search is based on results obtained by searching linked data sources on the web. In this a novel method for computing top-k routing plans based on their potentials to contain results for a given keyword query. In this project we propose to automatically identifying query groups is helpful for a number of different search engine components and applications, such as query suggestions, result ranking and collaborative search. In this system user search results is obtained based on the user role and the keyword given. Ranking and prediction is done based on the previous search history.

KEYWORD: Search Engine, database, top-k routing .

I. INTRODUCTION

THE web is no longer only a collection of textual documents but also a web of interlinked data sources (e.g., Linked Data). One prominent project that largely contributes to this development is Linking Open Data. Through this project, a large amount of legacy data have been transformed to RDF, linked with other sources, and published as Linked Data. Collectively, Linked Data comprise hundreds of sources containing billions of RDF triples, which are connected by millions of links (see LOD Cloud illustration at http://linkeddata.org/). While different kinds of links can be established, the ones frequently published are sameAs links, which denote that two RDF resources represent the same real-world object. A sample of Linked Data on the web is illustrated in Fig. 1. It is difficult for the typical web users to exploit this web data by means of structured queries using languages like SQL or SPARQL. To this end, keyword search has proven to be intuitive. As opposed to structured queries, no knowledge of the query language, the schema or the underlying data are needed.

II. RELATED WORK

Existing work can be categorized into two main categories:

- schema-based approaches
- Schema-agnostic approaches

There are schema-based approaches implemented on top of off-the-shelf databases. A keyword query is processed by mapping keywords to elements of the database (called keyword elements). Then, using the schema, valid join sequences are derived, which are then employed to join ("connect") the computed keyword elements to form so called candidate networks representing possible results to the keyword query. Schema-agnostic approaches operate directly on the data. Structured results are computed by exploring the underlying data graph. The goal is to find structures in the data called Steiner trees (Steiner graphs in general), which connect keyword elements. Various kinds of algorithms have been proposed for the efficient exploration of keyword search results over data graphs, which might be very large. Examples are bidirectional search and dynamic programming Existing work on keyword search relies on an element-level model (i.e., data graphs) to compute keyword query results.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

DISADVANTAGES OF EXISTING SYSTEM:

- The number of potential results may increase exponentially with the number of sources and links between them. Yet, most of the results may be not necessary especially when they are not relevant to the user.
- The routing problem, we need to compute results capturing specific elements at the data level.
- Routing keywords return the entire source which may or may not be the relevant sources
- User Search queries are no longer visible.
- It involves a high computational cost, since we would have to repeat a large number of query group similarity computations for every new query.
- Time consumption is high when performing the same task repeatedly.

III. PROPOSED ALGORITHM

We propose to route keywords only to relevant sources to reduce the high cost of processing keyword search queries over all sources. We propose a novel method for computing top-k routing plans based on their potentials to contain results for a given keyword query. We employ a keyword-element relationship summary that compactly represents relationships between keywords and the data elements mentioning them. A multilevel scoring mechanism is proposed for computing the relevance of routing plans based on scores at the level of keywords, data elements, element sets, and subgraphs that connect these elements. Based on modeling the search space as a multilevel inter-relationship graph, we proposed a summary model that groups keyword and element relationships using **K-means clustering** at the level of sets, and developed a **Page ranking algorithm** to incorporate relevance at different dimensions. In this project user can search data by providing the keywords of data. This also provides user to search based on their role. Search results are obtained based on the ranking. Prediction can be done by the previous search result.

Advantages:

- Individual searching history is monitored for the quick extraction of data.
- Keywords are grouped based on user role.
- Information is retrieved in fast manner which helps in time reduction
- Prediction is achieved to enhance the searching optimization

IV. LITERATURE SURVEY

BLINKS: Ranked Keyword Searches on Graphs

Query processing over graph-structured data is enjoying a growing number of applications. A top-*k* keyword search query on a graph finds the top *k* answers according to some ranking criteria, where each answer is a substructure of the graph containing all query keywords. Current techniques for supporting such queries on general graphs suffer from several drawbacks, e.g., poor worst-case performance, not taking full advantage of indexes, and high memory requirements. To address these problems, we propose BLINKS, a bi-level indexing and query processing scheme for top-*k* keyword search on graphs. BLINKS follow a search strategy with provable performance bounds, while additionally exploiting a bi-level index for pruning and accelerating the search. To reduce the index space, BLINKS partitions a data graph into blocks: The bilevel index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate search within blocks. Our experiments show that BLINKS offers orders-of-magnitude performance improvement over existing approaches. Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Search process; H.3.1 [Content Analysis and Indexing]: Indexing methods.

DataGuides: Enabling Query Formulation and Optimization in Semi structured Databases

In *semistructured* databases there is no schema fixed in advance. To provide the benefits of a schema in such environments, we introduce *DataGuides*: concise and accurate structural summaries of semistructured databases. DataGuides serve as dynamic schemas, generated from the database; they are useful for browsing database structure,

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

formulating queries, storing information such as statistics and sample values, and enabling query optimization. This paper presents the theoretical foundations of DataGuides along with algorithms for their creation and incremental maintenance. We provide performance results based on our implementation of DataGuides in the *Lore* DBMS for semistructured data. We also describe the use of DataGuides in Lore, both in the user interface to enable structure browsing and query formulation, and as a means of guiding the query processor and optimizing query execution.

Introduction

Traditional relational and object-oriented database systems force all data to adhere to an explicitly specified schema. Yet a typical site on the World-Wide Web demonstrates that much of the information available online is *semistructured*. Although the data may exhibit some structure, it is too varied, irregular, or mutable to easily map to a fixed schema. Recent research has focused on data models, query languages, and systems that do not require a schema to accompany each database [AQM+96, BDHS96, BDS95, KS95, MAG+97]. Beyond its use to define the structure of the data, a schema serves two important purposes:

□ □ □ A schema, in the form of either tables and their attributes or class hierarchies, enables users to understand the structure of the database and form meaningful queries over it.

□ □ □ A query processor relies on the schema to devise efficient plans for computing query results. Without a schema, both of these tasks become significantly harder. Although it may be possible to manually browse a small database, in general forming a meaningful query is difficult without a schema or some kind of structural summary of the underlying database. Further, a lack of information about the structure of a database can cause a query processor to resort to exhaustive searches. To address these challenges in "schema-free" environments, we introduce *DataGuides*, dynamically generated and maintained structural summaries of semistructured databases. This paper makes several contributions:

- We give a formal definition of DataGuides as concise, accurate, and convenient summaries of semistructured databases. Further, we motivate and define *strong* DataGuides, well-suited for implementation within a DBMS.
- We provide simple algorithms to build strong DataGuides and keep them consistent when the underlying database changes.
- We show how to store sample values and other statistical information in a DataGuide.
- We demonstrate how DataGuides have been successfully integrated into *Lore* [MAG+97] (for *Lightweight Object Repository*), a DBMS for semistructured data under development at Stanford University. DataGuides are vital to Lore's user interface: users depend on the DataGuide to learn about the structure of a database so they can formulate meaningful queries. In addition, users may specify and submit queries directly from the DataGuide.
- Finally, we explain how a query processor can use a strong DataGuide to significantly optimize query execution.

Finding Top-k Min-Cost Connected Trees in Databases

It is widely realized that the integration of database and information retrieval techniques will provide users with a wide range of high quality services. In this paper, we study processing an l-keyword query, p1; p2; _ _ _ ; pl, against a relational database which can be modeled as a weighted graph, G(V;E). Here V is a set of nodes (tuples) and E is a set of edges representing foreign key references between tuples. Let Vi _ V be a set of nodes that contain the keyword pi. We study finding top-k minimum cost connected trees that contain at least one node in every subset Vi, and denote our problem as GST-k. When k = 1, it is known as a minimum cost group Steiner tree problem which is NPComplete. We observe that the number of keywords, l, is small, and propose a novel parameterized solution, with l as a parameter, to find the optimal GST-1, in time complexity O(3ln + 2l((l + log n)n +m)), where n and m are the numbers of nodes and edges in graph G. Our solution can handle graphs with a large number of nodes. OurGST-1 solution can be easily extended to support GST-k, which outperforms the existing GST-k solutions over both weighted undirected/directed graphs. We conducted extensive experimental studies, and report our finding.

Introduction

Over decades, sophisticated database techniques have been developed to provide users with effective and efficient ways to access structural data managed by DBMS using SQL. At the same time, due to the rapid growth of hypertext data available on Web, advanced information retrieval techniques have been developed to allow users to use keyword

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

queries (a set of keywords) to access unstructured data that users are most likely interested in, using ranking techniques. It is widely realized that the integration of information retrieval (IR) and database (DB) techniques will provide users with a wide range of high quality services. The recent studies on supporting IR style queries in RDBMS include DBXPlore [1], IR-Style [17], DISCOVER [18], ObjectRank [4], BANKS-I [6], and BANKS-II [25]. All consider a RDBMS as a graph where nodes represent tuples/relations and edges represent foreign key references among tuples cross relations. And with a keyword query, users can find the connections among the tuples stored in relations without the needs of knowing the relational schema imposed by RDBMS. We show a motivation

Contributions of this paper: (1) We identify the characteristics of the group Steiner tree problem, for l-keyword query, over the database graph G, where the numbers of nodes and edges are n and m. The characteristics are: l is a small number, and G is a sparse graph with a large number of nodes. In brief, l _ log n, and m _ n2. (2) We propose a parameterized solution, with l as a parameter. We can obtain the optimal GST-1 with the worst-case time complexity O(3ln + 2l((l + log n)n +m)) and space complexity of O(2ln). Note: this parameterized algorithm works efficiently on the condition that l is small, and we are not solving the problem in a general setting where l can be any large. (3) We extend our solution to find GST-k in a progress manner. That is, we needn't compute/sort all group Steiner trees and then find GST-k. Our approach outperforms existing approaches with high quality and efficiency. (4) Our approach supports undirected graphs as well as directed graphs, with node/edge weights. We can handle the class of additive cost functions to score group Steiner trees.

Bidirectional Expansion For Keyword Search on Graph Databases

Relational, XML and HTML data can be represented as graphs with entities as nodes and relationships as edges. Text is associated with nodes and possibly edges. Keyword search on such graphs has received much attention lately. A central problem in this scenario is to efficiently extract from the data graph a small number of the "best" answer trees. A 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. In this paper we propose a new search algorithm, Bidirectional Search, which improves on Backward Expanding search by allowing forward search from potential roots towards leaves. To exploit this flexibility, we devise a novel search frontier prioritization technique based on spreading activation. We present a performance study on real data, establishing that Bidirectional Search significantly outperforms Backward Expanding search.

Keyword Search in Databases: The Power of RDBMS

Keyword search in relational databases (RDBs) has been extensively studied recently. A keyword search (or a keyword query) in RDBs is specified by a set of keywords to explore the interconnected tuple structures in an RDB that cannot be easily identified using SQL on RDBMSs. In brief, it finds how the tuples containing the given keywords are connected via sequences of connections (foreign key references) among tuples in an RDB. Such interconnected tuple structures can be found as connected trees up to a certain size, sets of tuples that are reachable from a root tuple within a radius, or even multi-center subgraphs within a radius. In the literature, there are two main approaches. One is to generate a set of relational algebra expressions and evaluate every such expression using SQL on an RDBMS directly or in a middleware on top of an RDBMS indirectly. Due to a large number of relational algebra expressions needed to process, most of the existing works take a middleware approach without fully utilizing RDBMSs. The other is to materialize an RDB as a graph and find the interconnected tuple structures using graph-based algorithms in memory.
In this paper we focus on using SQL to compute all the interconnected tuple structures for a given keyword query. We use three types of interconnected tuple structures to achieve that and we control the size of the structures. We show that the current commercial RDBMSs are powerful enough to support such keyword queries in RDBs efficiently without any additional new indexing to be built and maintained. The main idea behind our approach is tuple reduction. In our approach, in the first reduction step, we prune tuples that do not participate in any results using SQL, and in the second join step, we process the relational algebra expressions using SQL over the reduced relations. We conducted extensive experimental studies using two commercial RDBMSs and two large real datasets, and we report the efficiency of our approaches in this paper.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

INTRODUCTION

A conventional RDBMS provides users with a query language SQL to query information maintained in large RDBs. It requires users to understand how the information is stored in an RDB on a relational schema, and know how to specify their requests using SQL precisely. In other words, all the information that can be possibly found in an RDB is the information that can be expressed using SQL. Due to the rapid information growth in the information era, many real applications require integrating both DB and IR technologies in one system. On one hand, the sophisticated DB techniques provide users with effective and efficient ways to access structured data managed by RDBMSs, and on the other hand, the advanced IR techniques allow users to use keywords to access unstructured data with scoring and ranking. Chaudhuri et al. discussed the issues on integrating DB and IR technologies in [6]. In supporting IR-styled search, commercial RDBMSs (such as DB2, ORACLE, SQL-SERVER) support full-text keyword search using a new SQL predicate of contain(A, k) where A is an attribute name and k is a user-given keyword. With the new predicate, a built-in full-text search engine in the RDBMSs builds full-text indexes over text attributes in relations, and is used to retrieve the tuples that contain keywords in text attributes in relations efficiently.

In addition to full-text keyword search, another type of keyword search is to find how tuples that contain keywords in an RDB are interconnected. We call it structural keyword search. Consider a bibliography database maintains publication records in several relations in an RDB. It is highly desirable to find out how certain research topics and/or authors are interrelated via sequences of co-authorship/citations. For example, given three keywords, "Keyword", "DB", and "Yannis", the structural keyword search may find that "Yannis" writes a paper cited by two papers whose titles contain "Keyword" and "DB" respectively. Here, "Yannis" possibly appears in a tuple in an author relation, the three papers are three tuples in a paper relation, all are connected by foreign key references among tuples in other author paper relation and paper-citation relation. The structural keyword search is completely different from full-text search. The former focuses on the interconnected tuple structures, whereas the latter focuses on the tuple content. In this paper, we concentrate ourselves on structural keyword search, and simply call it keyword search.

In the literature, the existing works are categorized into schemabased approaches and schema-free approaches for (structural) keyword search. The schema-based approaches process a keyword query in two steps, namely, candidate network (CN) generation and CN evaluation. In the CN generation step, it generates all needed CNs (relational algebra expressions) up to a size, because it does not make sense if two tuples are far away in an interconnected tuple structure. In the CN evaluation step, it evaluates all CNs using SQL. The schema-free approaches support keyword search using graph-based in-memory algorithms by materializing an RDB as a graph. Almost all the existing work take a middleware approach except for two early work [1, 16] that evaluate CNs using SQL on an RDBMS directly. The middleware approach does not fully utilize the functionality of RDBMSs, and only uses SQL to retrieve data. In terms of the interconnected tuple structures, the majority of the works focus on connected tree. Recently [13] studies finding sets of interconnected tuples which can be uniquely identified by a root within a user given radius, and [26] studies finding sets of multi-center communities within a radius. All the three interconnected tuple structures are needed for different applications. But all are dealt in different ways and are not unified in the same framework.

A key issue we are studying in this work is how to support the three interconnected tuple structures (all connected trees up to certain size, all sets of tuples that are reachable froma root tuple within a radius, and all sets of multi-center subgraphs within a radius) in the same framework on RDBMSs without middleware. The main contributions of this work are summarized below. We propose a middleware free approach, to support three types of keyword queries to find the three different interconnected tuple structures. We take a tuple reduction approach using SQL without additional new indexing to be built and maintained and without any precomputing required. To compute all connected trees, we propose a new approach to prune tuples that do not participate in any resulting connected trees followed by query processing over the reduced relations. To compute all multi-center subgraphs, we propose a new three-phase reduction approach to effectively prune tuples from relations followed by query processing over the reduced relations. We use the similar mechanism for computing all multicenter subgraphs to process sets of tuples that are reachable from a root tuple within a radius. We conducted extensive performance studies using two commercial RDBMSs and two large real datasets to confirm the efficiency of our proposed approaches.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

V. CONCLUSION AND FUTURE WORK

We have presented a solution to the novel problem of keyword query routing. Based on modeling the search space as a multilevel inter-relationship graph, we proposed a summary model that groups keyword and element relationships at the level of sets, and developed a multilevel ranking scheme to incorporate relevance at different dimensions. The experiments showed that the summary model compactly preserves relevant information. In combination with the proposed ranking, valid plans (precision@1 ¼ 0:92) that are highly relevant (mean reciprocal rank ¼ 0:86) could be computed in 1 s on average. Further, we show that when routing is applied to an existing keyword search system to prune sources, substantial performance gain can be achieved.he performance.

REFERENCES

[1] V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), pp. 850-861, 2003.
[2] F. Liu, C.T. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Databases," Proc. ACM SIGMOD Conf., pp. 563-574, 2006.
[3] Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-K Keyword Query in Relational Databases," Proc. ACM SIGMOD Conf., pp. 115-126, 2007.
[4] M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano, "Efficient Keyword Search Across Heterogeneous Relational Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 346-355, 2007.
[5] B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-K Min-Cost Connected Trees in Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 836-845, 2007.
[6] B. Yu, G. Li, K.R. Sollins, and A.K.H. Tung, "Effective Keyword- Based Selection of Relational Databases," Proc. ACM SIGMOD Conf., pp. 139-150, 2007.
[7] Q.H. Vu, B.C. Ooi, D. Papadias, and A.K.H. Tung, "A Graph Method for Keyword-Based Selection of the Top-K Databases," Proc. ACM SIGMOD Conf., pp. 915-926, 2008.
[8] V. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB), pp. 670-681, 2002.
[9] L. Qin, J.X. Yu, and L. Chang, "Keyword Search in Databases: The Power of RDBMS," Proc. ACM SIGMOD Conf., pp. 681-694, 2009.
[10] G. Li, S. Ji, C. Li, and J. Feng, "Efficient Type-Ahead Search on Relational Data: A Tastier Approach," Proc. ACM SIGMOD Conf., pp. 695-706, 2009.

BIOGRAPHY

Dr. M.Mayilvaganan is a Research Associate in the computer Science Department, PSG College of Arts and Sciences, Coimbatore, India. He received Master of Computer Applications (MCA) degree in 1990 and also completed in PhD doctorate in 2009. His research interests are Data Mining, Big Data Analysis, networking etc...