CS 540 – Database Management Systems
Midterm Report

Keyword Query Interface for Relational Databases

Shajith Ravi
ravish@onid.orst.edu

Lakshman Madhav Kollipara
kollipal@onid.orst.edu

Prathamesh Patkar
patkarp@onid.orst.edu

Varun Rajgopal
rajgopav@onid.orst.edu

Description of the problem

There are various application in which plain text coexist with data structure such as google and yahoo search. Commercial RDBMS provides ability to query for text attributes that envisaged Information Retrieval(IR) relevance ranking strategy , but this feature needs queries to specify the exact column or column against which a given list of keyword to be matched. This requirement is undesirable and inflexible from users perspective, in which good queries must be assembled by joining tuples or multiple relations. In the other words Naive users do not have knowledge on databases and SQL. Thus they need an easy query interface through which they can explore their relational databases. Thus, researchers have developed keyword query interfaces for relational databases, where users can submit their queries in form of keywords and get their answers (like Google keyword search interface). This keyword queries are not well specified as SQL queries, but they are interfaced like Google where it returns ranked list of results that may contain desired outcome. The advantages of the keyword search on database are, it is easy to use and enables discoveries to access data in web and scientific application. That is relevant data are scattered but correct data to query should be displayed as result. This uses relevance ranking. Ranking function leverages and extends the ability of modern relational database system to provide keyword search on individual attributes and tuples. Ranking function leverages and extends the ability of modern relational database system to provide keyword search on individual attributes and tuples.

The scope of project is to implement changes in database design and empirically study the changes to the schema design on the ranking quality of the answers provided by query interface. Mainly we are focusing on horizontal decomposition and composition of schema. Ranking function leverages and extends the ability of modern relational database system to provide keyword search on individual attributes and tuples.

A description of what you have done so far and the problems and issues you have faced in implementing / developing your solutions.

As part of our project we have selected a part of IMDB implementation of freebase, which includes information about various movies, genre and description and it consists of 1 million records.

We have imported the freebase dataset into MySQL database. We have chosen MySQL because it is most widely used open source relational database management system. It is easy to use, scalable and thoroughly tested to prevent memory leaks.

We have considered working on the following three key-word interfaces Querious, Sphinx and SQLyog. By the end of the week we will finalise on one interface based on the performance, usability and algorithms.

Querious is a GUI based query interface for MySQL. It provides a set of working algorithms that lets you store and manage your data. Querious supports direct connections to MySQL over SSL and SSH tunneling using authenticated passwords or keys that are set by the user. We successfully connected the Querious with MySQL database. Using the advanced filters of Querious we were able to query the database by passing in relevant key words. The disadvantage of using Querious is that it does not provide us with the information on their algorithms.

Sphinx is a free software/open source full-text search engine designed to provide full-text search functionality to client applications. It can be used to communicate with DBMS by using native protocols of MySQL. We have integrated Sphinx for querying MySQL database. It provides a proper interface to work with. Sphinx provides us with some information on their algorithms.

SQLYog is a powerful GUI MySQL manager and admin tool combining various features of MySQL workbench, schema design and optimization, creating indexes and keyword search.

A brief plan on how you plan to solve address issues and problems and the rest of the term. For instance, if you have to evaluate your or some system, you should explain how you plan to evaluate the system and what experiments you like to perform.

There are a few papers that talk about importance of data schema and ranking quality of answers.

Data Quality and Data Cleaning: An Overview by Theodore Johnson at AT&T Research Labs.

Data quality problems are expensive and pervasive where problems cost millions of each year. Resolving data quality problems is often the biggest effort in a data mining study.
The problems we face in data retrieved may be one or more of the following:

- Immeasurable
  - Accuracy and completeness are extremely difficult, perhaps impossible to measure.

• Context independent
– No accounting for what is important. E.g., if you are computing aggregates, you can tolerate a lot of inaccuracy.
• Incomplete
– What about interpretability, accessibility, metadata, analysis, etc.
• The conventional definitions provide no guidance towards practical improvements of the data.

Estimating the quality of answers when querying over description logic ontologies by Martin Peim, Enrico Franconi, Norman W. Paton

This paper presents an approach to estimating the soundness and completeness of queries expressed. It would be useful if information integration systems were able to provide users with estimates of the consequences of omitting certain sources from query execution plans. Such omissions can affect both the soundness (the fraction of returned answers which are returned) and the completeness (the fraction of correct answers which are returned) of the answer set returned by a plan.

How Schema Independent Are Schema Free Query Interfaces? Marianne Winslett, Arash Termehchy, Yodsawalai Chodpathumwan

In this paper, they define design independence, which captures this property for schema free query interfaces. There is a theoretical framework to measure the amount of design independence provided by a schema free query interface. The importance of mentioning the paper in our work is that this paper clearly states that most current SFQIs provide a very limited degree of design independence. SO we can come to an understanding that implementing a database schema free cannot be the solution for ranking quality of answers provided by any dataset.

AN OVERVIEW OF RELATED PAPERS AND HOW WE DEFINE OUR APPROACH CONSIDERING THE METHODS DESCRIBED.

The existing keyword search over RDBMSs are inherently inefficient because they attempt to capture all tuple trees with all query keywords. Thus these strategies do not exploit a crucial characteristic of IR-style keyword search, namely that only the top 10 or 20 most relevant matches for a keyword query –according to some definition of "relevance"– are generally of interest.

The techniques of IR-style queries over RDBMSs produce the top-k matches for a query –for moderate values of k– in a fraction of the time taken by state-of-the-art strategies to compute all query matches. Furthermore, these techniques are pipelined, so that execution can efficiently resume to compute the "next-k" matches if the user so desires. they define the problem of processing keyword-search top-k queries over RDBMSs, provide necessary notation, and describe the general architecture of the system. The relations in database are represented by a joining tree of tuples from primary key of one table to foreign key of another table. The results are given a score based on few ranking functions and listed by their ranking order. If we have multiple attributes, then we will pass single attribute scores for a joining tree of tuples T into a final score for the tree. They described the modules of Architecture of query processing system.

IR Engine:The IR Engine module of our architecture exploits this functionality to identify all database tuples that have a non-zero score for a given query. Candidate Network (CN) Generator: Receives as input the non-empty tuple sets from the IR Engine, together with the database schema and a parameter M which bounds the size in number of tuple sets, free or nonfree of the CNs that this module produces. The key role of this module is to produce CNs, which are join expressions to be used to create joining trees of tuples that will be considered as potential answers to the query.Execution Engine: Receives a set of CNs together with the non-free tuple sets as input and contacts the RDBMS's query execution engine repeatedly to identify the top-k query results.

They presented query processing algorithms for how Execution Engine efficiently identifies top-k joining trees of tuples from given set of inputs with set of CNs. Naive Algorithm, Sparse Algorithm, Single pipelined algorithm, Global pipelined algorithm, and Hybrid Algorithm.
The Naive algorithm exhaustively processes every CN associated with a query which effects performance.Sparse is the most efficient algorithm for queries with relatively few results. Single Pipelined algorithm is not an efficient choice when used separately for each CN. The correctness of Global Pipelined relies on the combining function satisfying the tuple-monotonicity property. Global Pipelined is the most efficient algorithm for queries that produce many results.
They compared results of all algorithms with Boolean AND or OR query semantics.

References:
1] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. 2003. Efficient IR-style keyword search over relational databases. In *Proceedings of the 29th international conference on Very large data bases - Volume 29* (VLDB '03), Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer (Eds.), Vol. 29. VLDB Endowment 850-861.

2] 2002. DBXplorer: A System for Keyword-Based Search over Relational Databases. In *Proceedings of the 18th International Conference on Data Engineering* (ICDE '02). IEEE Computer Society, Washington, DC, USA, 5-.

3] Fang Liu, Clement Yu, Weiyi Meng, and Abdur Chowdhury. 2006. Effective keyword search in relational databases. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (SIGMOD '06). ACM, New York, NY, USA, 563-574. DOI=10.1145/1142473.1142536 http://doi.acm.org/10.1145/1142473.1142536

4] Arvind Hulgeri and Charuta Nakhe. 2002. Keyword Searching and Browsing in Databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering* (ICDE '02). IEEE Computer Society, Washington, DC, USA, 431-.