Practical Free-Form Search over Relational Databases

Mohammad Hassan,
Computer Science Dept.
Zarqa University,
Zarqa-Jordan
mohdzita@zu.edu.jo

Abstract: **This paper presents a tool that enables non-technical end-users to use free-form queries in exploring relational databases with simple and direct technique, in a fashion similar to using search engines to search text files on the web. This allows web designers and database developers specially in small business sectors to publish their databases for web browsers exploring in practical way. We assume that each database to be processed in our approach is 'about' a particular type of object, where only certain tables and certain attributes will be the 'public' ones and likely to be searchable on the web. The proposed approach can be used for both Internet and Intranet application areas. Our technique has been borrowed from the similar techniques used for information retrieval (IR), mainly for text and document databases; it supports working smoothly with the structured information stored in relational databases.**

Keywords: **Relational Databases, information retrieval, keyword-based search.**

I. INTRODUCTION

Keyword search is a method of extracting information from data sources by providing a set of terms as input [1]. Keyword search is the most popular information discovery method because the user does not need to know either a query language or the underlying structure of the data. The search engines available today provide keyword search on the top of sets of documents. When a set of keywords is provided by the user, the search engine returns all documents that are associated with these keywords. Users can follow hyperlinks to navigate from one document to the other. No knowledge of schema is needed.

In addition to documents, a huge amount of information is stored in relational databases, but information discovery using keyword search on relational databases is not supported well and usually need advanced techniques to operate it. The user of relational database needs to know the schema of the database, SQL or some QBE-like interface, and the roles of the various entities and terms in the query [2]. That requires users to know the organization and the contents of the database they are accessing. Since this is not always the case, specially when users are searching a public Web-database. It is important that users be able to search relational databases without a priori knowledge of the databases' schemas or the location of required information, that enable users to seamlessly search information stored in these databases as well. Searching databases on the Internet and on private networks (Intranets) today is primarily enabled by customized web applications closely tied to the schema of the underlying databases, allowing users to direct searches in a structured manner. While such structured searches over databases are no doubt useful, unlike the documents world, there is little support for keyword search over relational databases.

Our goal is to enable such searches without necessarily requiring the users to know the schema of the respective databases. Yet, today's customized web applications as described above and traditional SQL applications require knowledge of the schema.

Keyword-based search is a well studied problem in the world of text documents [3, 4] and Internet search engines. Inverted lists are common data structures used for solving keyword queries. However, it is not an easy process when structured databases are involved. It is even more complicated in a dynamic environment like the Internet where varying or unknown database structure makes the query formulation process a very difficult task. Furthermore, most commercial systems that provides keyword search on data stored in relational databases (or documents i.e., search engines) tends to hide how exactly these systems handle keyword quires, (often for commercial reasons). So, detail reference materials are not publicly available [5].

The rest of the paper is organized as follows. Related works are reported in Section 2. The proposed approach is described in Section 3. Prototype is reported in Section 4. Section 5 includes the summary and the conclusions.

II. RELATED WORKS

A number of different approaches and techniques for enabling keyword search over relational database have been proposed. For instance, BANKS [6] is a system that enables keyword-based search on relational databases, together with data and schema browsing. Its algorithm works on a graph (data graph), with tuples as nodes and the weighted link between them signifies the degree of correlation between them. A drawback of this approach is that a graph of the tuples must be created and maintained for the database, and the algorithm works on a huge data graph. Furthermore, the query evaluation with keywords matching a large number of tuples (nodes within the data graph) can be slow. This problem arises because of performing backward search from large number of nodes, which is the technique used for such

queries. DataSpot [7] is a commercial system that supports keyword-based search by extracting the contents of the database into a hyperbase. Thus, this approach duplicates the contents of the database, making data integrity and maintenance difficult.

DBXplorer [8] describes a multi-step system to answer keyword queries in relational databases. Even this approach is using a schema graph (not a data graph) that can be considered smaller than the data graph used in BANKS, the graph must be created and maintained for database. EasyAsk (www.easyask.com) is another commercial system that provides natural language search (including keyword search) on data stored in relational databases. EasyAsk does a variety of tasks such as approximate word matching and natural language understanding. However, details of how they handle keyword queries are not publicly available [5]. Several other approaches were also presented such as [9, 10].

Most of the keyword search techniques have one thing in common, namely the basic representation of the database. They all implement it using graphs. Although some call it a semantic network, data graph, schema graph, and some other abstraction graph, they basically have the same underlying graph structure [1]. These graph structures are necessarily for these approaches, since they should take all relationships between tables in the database into account for identifying the required results. In our approach we have not used graph structures that consider an expensive and difficult task.

III. The Proposed Keyword Search Approach

Given a relational database, our system offers a flexible interface to access/retrieve only database items relevant to a given query. This requires executing a search process that depends on the database representative.

A. Database Representative

Information retrieval systems are concerned with providing the information that the user seeks. In traditional IR systems, different types of database representatives may be available to the search system.

These representatives that characterize the contents of the database are used to identify the data items within the database (for example: documents) with the highest potential to satisfy that information need. Depending on the kind of information and the database representatives available, different approaches have been proposed to identify such data items. Most approaches require information about the terms that appear in the database and the statistical information related to these terms, such as, term frequencies, document frequencies and term weights. One of the most popular representatives used in distributed search systems and information retrieval systems is the *inverted file*. The inverted file is accepted as the classical index structure for keyword search (in the world of text documents). In its simplest form, an inverted file contains records of the form: <word, document>, where the word can be found in the document.

To serve our purposes, we modify the structure of these inverted files to support keyword search over relational databases and to determine whether the database (or database granularity) contains useful information. Traditional IR systems depend on the granularity of documents, whereas our proposed approach that use the modified inverted file stores information at database granularity –specifically at row granularity- as we will explain it later in this section. This modified inverted file is an essential part of the required database representative in our approach, which we call the *database word-frequency information.*

Now we consider how the database representative of each database can be created. This is a two part process in which firstly a *collective relation* is created. This collective relation is then used to build the database word-frequency information.

B. The Collective Relation

We define the collective relation as the:

(outer) join of all publicly accessible (possibly renamed) attributes.

We cannot generally define the attributes or nature of the join. Without loss of generality, we assume that each database to be processed in our approach is 'about' a particular type of object, where only certain tables and certain attributes will be the 'public' ones. This is the sort of database and the type of data that we might expect to find on the web and is the sort of database that we are considering the keyword search problem for. We include the possible renaming of attributes to reflect how the database appears publicly rather than with names for attributes that may be used within the database.

The selection of data portions to feed our proposed selection process is not new. Many applications require squeezing data from multiple relations into a single table in order to meet the processing requirements, for example Data Mining processes. Such applications require the data to be structured in only one relation (or data file), so the preparation of the data to be submitted to them is a process usually considered as part of cleaning and pre-processing steps. Several works have been proposed, aiming to enable exploratory data analysis of massive datasets, and to help in finding interesting data subsets to process. The problem of attribute selection is part of these works [11].

The collective relation enables our approach to represent and handle structural information in a simple and easy way (no table joins are involved). Extracting the database word-frequency information from a Collective

relation will not take the relationships between tables into account and they will not be represented, hence the database word-frequency information will be much smaller. As a consequence there will be no need for using any kind of graph structures, the technique used in most (keyword-based) relational database searching approaches. This may be considered the important aim of using the collective relation technique.

We should reconfirm that we are regarding each database as about something in particular (cars, books,… etc.) and that the collective relation is the possibly unnormalized information about that 'entity' or object and that all other tables are not part of that.

The concept of creating a collective relation in our approach is similar (to a certain extent) to work on universal relations [12], where a database is viewed as a single universal relation for querying purposes, thus hiding the complexity of schema normalization. Although the proposed collective relation in our approach is similar to the universal relation concept (viewing the database as single relation), there are many differences between the two concepts, and some important differences are:

- The collective relation in our approach contains only the data that is made publicly available and therefore searchable externally, not all data stored in a database as in universal relations.
- No assumptions are imposed while building the collective relation, while in universal relation several assumptions are essential such as the uniqueness assumption and the universal instance assumption. Moreover, normalization is not a requirement in the proposed collective relation, such a relation could be built by combining the needed attributes (tables) using (normal) join operations.

The concept of collective relation in our approach is also similar to work on master relations that introduced in a previous paper by me and other authors [13], that serve different purposes. So, the first step in our approach is to create the collective relation that combines all the required information needed to be searched in order to create the database representation which serves the purpose of our approach.

Creating the collective relation, which considered the first step of creating the database word frequency information that will form the database representative, will be done by the database administrator. The database administrator has to decide what portions of the database should be included within the collective relation. It is important to note that in our approach, choosing the required data elements should in all cases be an easy task, because the administrator should have knowledge about the searching goal.

The task starts by creating a collective relation (table) that combines all the data required to be searched within the database. The administrators should choose only the tables, fields and records that may be searched while building this relation. Data items included within this relation have to be chosen carefully, such that each of them is semantically meaningful for the keyword-based queries. Many tables or fields or even records will not be usefully included within this relation; such fields usually do not contain significant data while applying a keyword search over the data. We assume here that the database administrator has or can obtain the information above.

The example database used in the experimental results, are about used cars agent. A realistic used car database may contain many tables and attributes (or records) that should not be included in the collective relation, because they do not contain significant data for the process our approach is concerned with. For example, attributes like:

> Vehicle plate number, engine number, chassis number, engine capacity, maximum capacity, vehicle weight, number of axles,.. etc.

and tables about:

> Vehicle License with attributes: place of issue, (date: from - to).
> Finance Data with attributes: cost price.
> Vehicle Status with attributes: status (booked/sold), buyer detail.
> Vehicle History with attributes: previous owners, maintenance records, accidents.

should not be included within the collective relation of the database, because they do not contain significant data while applying a keyword search over the data (cars) or in any advertising about the cars by any used car dealer. It is obvious that all cars with sold or booked status should not be represented within the database representative. In this case all records where the status attribute is (booked or sold) should not be included within the collective relation.

C. The Database Word-Frequency Information

Using the collective relation introduced above, we can create the database word-frequency information (that has records with the following structure: <*word, row#*>, where the word (keyword) can be found at row# granularity of the collective relation.

While creating this database word-frequency information, we may reduce the words in the collective relation to their stems using a stemming algorithm. We also may leave out frequently occurring words with little semantics (stop words) using a stopping algorithm.

In a relational database environment, creating the collective relation can be done easily either using an SQL or some QBE-like interface (for example: make table query). Creating the database word-frequency information could also be done automatically using simple algorithms that work on the previously described collective relation. We developed an application for building these statistical information as part of our prototype using Java and JDBC. Table-1 shows a portion of the database word-frequency information extracted from the collective relation of the example database.

Table 1: A portion of the database word-frequency information extracted from the collective relation of the example database.

row# word	1	2	3	4	5	.....	.....	73	74	75
4WD	0	0	0	0	0			0	0	0
Accord	0	0	0	0	0			0	0	0
Audi	1	1	0	0	0			0	0	0
Honda	0	0	0	0	0			0	0	0
Japan	0	0	0	0	0			0	0	0
Manual	1	0	1	1	0			1	0	1
Red	0	1	0	0	0			1	0	0
Sedan	0	1	1	1	0			0	1	1

D. Query Representation

Our approach considers Boolean queries that consist of atomic subqueries (single keywords) connected by a Boolean operator ("and", "or" / "not" may also included). So, a query in our model can be represented by an unordered subset of *W*, where $W=\{w_1,w_2, ...,w_n\}$ is the set of all words in the database word-frequency information relation.

We consider Boolean queries, even if the Boolean model is not favored in all situations compared to other model types, such as: vector space model, probabilistic model, natural language model, etc. which could be adopted easily. However, most current commercial online services and information vendors worldwide as well as traditional library systems support Boolean query models to access their databases, offering well-maintained information in many fields such as science, business, and law. Furthermore, Boolean queries could be used and easily expressed in searching relational databases by utilizing standard RDBMS query language functionality. Therefore, we believe that supporting the Boolean model is critical for providing integrated access to both modern and legacy systems in order to provide access to their valuable contents.

E. The proposed approach

Our keyword-based approach is easy to describe. Given a relational database along with its collective relation and its database word-frequency information. Given a query that consists of terms and logical operator. We view our approach as retrieving all records/rows (in the collective relation) that satisfy the query.

IV. PROTOTYPE SYSTEM

The objective of our prototype system is to demonstrate the feasibility of building the proposed keyword search approach. As we introduced in the previous section, the process starts by creating a single collective relation that combines all the data required to be searched within the database. Collective relations are considered the basic structures in our approach. Using the collective relation, we can build the database word-frequency information needed by the system.

To perform experiments we used (a real) example database. The example database is small for exposition purposes and suitable to illustrate the general concept that make up keyword search algorithm and the functionality associated with it. The example database store information about "cars" of a small used cars selling company. The database is managed by SQL-Server.

All of the above process has been incorporated in a prototype developed as a web-based application using Java and JDBC. An example screen of the graphical user interface of our system along with a search query for two terms with "AND" operator is shown in Figure 1.

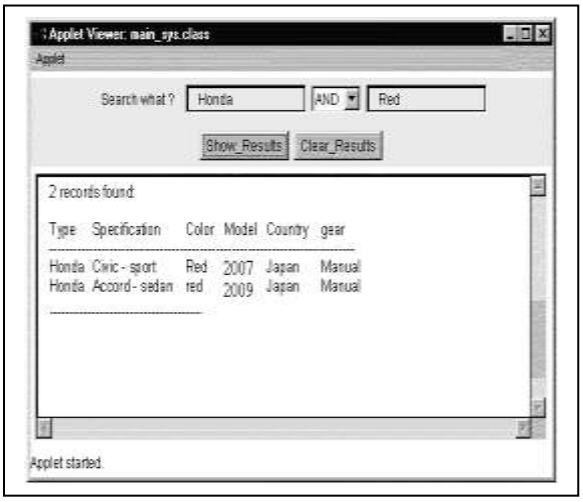

Figure 1: The User Interface and Search for "Honda" and "red" keywords.

V. CONCLUSIONS, EVALUATION AND FUTURE WORK

In this paper, we described a mechanism that enables users to search relational databases with simple word-based queries. We provided an overview of our proposed approach, which has been implemented as a Web based application.

To support relational query processing using keyword search, a collective relation for the database was introduced. Such a relation is supposed to combine all data, which may

be searched within the database. The collective relation is considered the source of the database representative, from which the database word-frequency information could be extracted. Also, we gave the necessary modification to the inverted file to take the structure of relational databases into account. Finally, a prototype for each part of the proposed approach has been developed and implemented.

One possible drawback of using a collective relation is that the size of this relation may become large (depending on the size and the structure of the database). This might be particularly significant with databases including numerous one-to-many relationships where an almost exponential growth could in theory be possible. However, for the class of database we consider here, limiting the tables/attributes/records to those publicly available for querying on the web, there will be a limit to this growth.

In this work we will limit our study to the restricted class of database likely to be searchable on the web as outlined above. We do not investigate the issues of possible distortion of the relative weighting of some database attributes or alternative representations.

Regarding the evaluation of the proposed approach, researchers generally use the two well-known parameters, *recall* and *precision*. Recall and precision may be considered as an attempt to measure what is known as the *effectiveness* of the retrieval system. Effectiveness is purely a measure of the ability of the system to retrieve relevant documents (records in our approach) while at the same time holding back non-relevant ones. It is assumed that the more effective the system is, the more it will satisfy the user. It is also assumed that precision and recall are sufficient for the measurement of effectiveness.

According to our approach, issuing queries to the system is equivalent to issuing queries to the collective relation that represent the actual database, which controlled by the standard RDBMS query language functionality. So, the results of such queries will be exact. This means, for any query, all of the retrieved results/records are relevant (recall), and in the meanwhile all of the actual relevant records will be retrieved (precision). Hence, the effectiveness of such approach depends on the collective relation and how much it reflects the required data within the actual database.

Currently, we are working on different query models along with evaluating the storage requirements of the proposed approach.

REFERENCES

[1] R. Nasre, Keyword Search in Databases, Mtech Project, Department of Computer Science and Engineering, IIT, Mumbai, India, 2002.

[2] V. Hristidis, and Y. Papakonstantino, DISCOVER: Keyword Search in Relational Databases. Proceedings of the 28th International Conference on Very Large Databases, Hong Kong, China, 2002.

[3] G. Salton, Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison Wesley, 1989.

[4] L. Gravano, Querying text databases and the web: beyond traditional keyword search: Proceedings of the First International Workshop on Keyword Search on Structured Data, Rhode Island, USA, 2009.

[5] A. Hulgeri, G. Bhalotia, , C. Nakhe, and S. Chakrabarti, Keyword Search in Databases, IEEE Computer Society Technical Committee on Data Engineering, 2001.

[6] A. Hulgeri, G. Bhalotia, , C. Nakhe, and S. Chakrabarti, Keyword Searching and Browsing in Databases using BANKS. Proceedings of the 18th International Conference on Data Engineering, California, USA, 2002.

[7] S. Dar, G. Entin, , S. Geva, , and E. Palmon, DTL's DataSpot: Database Exploration as Easy as Browsing the Web, ACM SIGMOD RECORD, p: 590-592, 1998.

[8] S. Agrawal, , S. Chaudhuri, , and G. Das, DBXplorer: A System for Keyword-Based Search over Relational Databases, Proceedings of the 18th International Conference on Data Engineering, California, USA, 2002.

[9] K. Stefanidis, M. Drosou, and E. Pitoura, PerK: personalized keyword search in relational databases through preferences. In EDBT, pages 585–596, 2010.

[10] X. Jeffrey, L. Qin, L. Chang, Keyword Search in Relational Databases: A Survey. IEEE Data Engineering Bulletin,Vol. 33, No. 1, 2011.

[11] S. D'zeroski, and L. De Raedt, Multi-Relational Data Mining: a Workshop Report, 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-02), Edmonton, Canada, 2002.

[12] C. Date, An Introduction to Database System. Addison-Wesley, Publishing Company, sixth edition, 1994.

[13] M. Hassan, R. Alhajj, M. Ridley, and K. Barker, Simplified Access to Structured Databases by Adapting Keyword Search and Database Selection". Proceedings of ACM Symposium on Applied Computing SAC2004 - Nicosia, Cyprus, 2004.