Simplified Access to Structured Databases by Adapting Keyword Search and Database Selection

Mohammad Hassan†, Reda Alhajj†
†School of Informatics
Bradford University
Bradford, West Yorkshire
United Kingdom

Mick J. Ridley‡, Ken Barker‡
‡ADSA Lab
Dept of Computer Science
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

This paper presents a tool that enables non-technical (naive) end-users to use free-form queries in exploring distributed 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 to publish their databases for web browsers exploring. The proposed approach can be used for both Internet and Intranet application areas. Our approach depends on identifying first databases that are most likely to provide useful results to the raised query, and then searches only the identified databases. In our work, we developed and extended an estimation technique to assess the usefulness measure of each database. 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. Such a usefulness measure enables nave users to make decisions about databases to search and in what order.

Categories and Subject Descriptors
H.3.3 [INFORMATION STORAGE AND RETRIEVAL]: Information Search and Retrieval

General Terms
Design.

Keywords
database selection, information retrieval, keyword search, relational databases.

1. MOTIVATION AND CONTRIBUTIONS

Accessing web-based information sources by nave users is becoming a more common and essential daily activity. In addition to documents, a huge amount of information is stored in structured databases. However, the way we search for data is one of the most important differences between relational and text or document databases, at least from an end user’s perspective. It is more complicated to access structured databases, especially in a dynamic environment like the web, where varying or unknown database structure makes the query formulation process a very difficult task. An approach taken in some cases is to export data from structured databases to web pages, and then provide text search on web documents. This approach results in duplication of data, with resultant problems of keeping the versions up-to-date, in addition to space and time overheads. Further, with highly connected data, it is not feasible to export every possible combination.

The problem handled in this paper can be stated as follows. Given a set of distributed heterogeneous relational databases and a query q, which consists of some keywords connected by logical operators, it is required to allow users to access and manipulate certain information by using a free-form search to submit q such that no a priori knowledge of the database schemas or the place of the required information is necessary. It is also required to rank the databases to provide the most effective response to q. The proposed mechanism accomplishes these goals without affecting the autonomy and heterogeneity of the participating databases.

The contributions of our work can be enumerated as follows. 1) we present an extension to the structure of the inverted file used for traditional IR. This is necessary to let the structured information buried in relational databases be taken into account in a free search methodology. 2) We developed an estimation method with statistical foundations to estimate the usefulness of individual relational databases. Some experiments have been conducted to demonstrate the effectiveness of the proposed method in determining promising sources for a given query. 3) As naive end-user satisfaction is a main target and motive, we provide a prototype system with a friendly web-based user interface that accomplishes our goals in a simple and straightforward method.

The rest of the paper is organized as follows. The related work is summarized in Section 2. The proposed system that employs keyword search and database selection is described in Section 3, which also covers the estimator function with illustrative examples. The evaluation parameters used in the selection process are presented in Section 4. Section 5 includes a summary and the conclusions.

2. RELATED WORK

Adapting keyword search to structured databases has al...
ready attracted the attention of several researchers, e.g., DBXplorer [1], DataSpot [4], DISCOVER [11], BBQ [13] and others [3, 7, 9, 12, 14]. However, existing approaches mainly concentrate on searching a single database. They also have one thing in common; almost all of them use a graph to implement the basic database representation. The main limitation of this work is that all keywords must be contained in the same tuple.

In addition to keyword search, the research project described in this paper intersects with some major research areas, mainly multidatabases [5], federated databases [10], IR [15], and database selection [8].

Our proposed solution to the usefulness estimation problem is an extension of the approach of GIOSS [8]; we took the level of search and the structure of the relational database into account. The frequency information that GIOSS keeps about the databases are different from ours. The frequency information that our system keeps is related to the granularity of each relational database within the system.

3. THE PROPOSED APPROACH

Given a set of relational databases, our system offers a flexible interface to access only databases relevant to a given query. A database is relevant if it contains some information to participate to the answer of the raised query. This requires executing a search process which is facilitated in our system by providing the following efficient and effective approach.

![Figure 1: The proposed system architecture](image)

Each site constructs its indexing vocabulary and computes its local database statistics; these represent levels of database frequencies for each term in the vocabulary, and the summary index that contains the total number of elements at each level. The system requires that each database cooperates and periodically updates these frequencies, following some predefined protocol. Those indexes could be easily built mechanically in any RDBMS. We developed an application for building these indexes as part of our prototype using both Oracle and MS-Access.

![Figure 2: Example screen on searching with the “and operator at the record level](image)

A central database selection server imports such information from each remote database. We developed and implemented in Java and using JDBC an application to import this information from distributed remote databases.

The server creates a union vocabulary and a union summary index. We developed and implemented an application using Java and JDBC to build these two unions.

Given a query \( q \) that consists of terms, logical operators, search-levels and search criteria, the conceptual database selection algorithm proceeds as follows:

1. Compute the terms estimator according to the specified levels and the specified logical operator.
2. Compute the chosen set according to the specified search criteria.
3. Sort the databases according to the chosen set results, if the relevant search criterion is chosen.

All this process has been incorporated in a prototype developed as a web-based application using Java and JDBC. The architecture of the prototype is shown in Figure 1. An example screen of the graphical user interface of our system is shown later in Figure 2.

So, consider a query \( q \), which is to be evaluated over a set of relational databases \( DB \), the system selects a subset of \( DB \), which consists of relevant candidate databases for submitting \( q \). To make this selection, the system uses an estimator function, which assesses how relevant each database in \( DB \) is with respect to \( q \).

In the rest of this section, first we discuss the extension to the inverted file; second we describe query representation and database frequency information; and finally we cover the estimator function.

3.1 Extending Inverted File

The inverted file is accepted as the classical index structure for keyword search. Inverted lists [2] are the main data structure utilized by traditional IR techniques for enabling keyword search in document collections; they help in efficiently identifying documents containing a query keyword.

In its simplest form, an inverted file contains records each consists of the pair: \( (\text{word}, \text{document}) \), to correlate each \text{word} with the \text{document} in which it can be found. To serve our purpose, we extended the inverted file to support keyword search and to find relevant database sources or database elements in a distributed environment. We need to record
whether a keyword is the name of a table (or a word in a table), the name of an attribute (or a word in an attribute), or the value of an attribute (or a word in the value of an attribute). Such a distinction is not required in traditional IR systems because they ignore the structure and only depend on the granularity of documents. As a result, we propose to use an inverted file that has records with the following structure: \( (w, \text{ld}, \text{word}, \text{freq}, \text{level}, \text{db,ld}) \), where \( w, \text{ld} \), \( \text{freq} \), and \( \text{db,ld} \) respectively, stand for the word id, frequency and database id, while level specifies the depth (granularity of the database) at which the word appears.

Another schema is also needed to store the totals summary of the frequencies for all databases. It contains the following information: \( (\text{db,ld, r_total, c_total, t_total, URL}) \), where \( r_{\text{total}}, c_{\text{total}} \) and \( t_{\text{total}} \) represent the total number of elements at the three levels record, column and table of the database, and URL is the address of the particular database site. Our retrieval approach is based upon this structure in computing the relevance of structured elements with respect to a given query.

### 3.2 Query Representation and Word-Frequency Information

We consider boolean queries that consist of atomic subqueries connected by boolean operators, including AND, OR, and NOT denoted as \( \land \), \( \lor \) and \( \neg \), respectively. An atomic subquery is a keyword. In addition, a query in our system is expressed using a user-friendly interface designed to serve both naive and frequent users. A sample illustrative screen is shown later in Figure 2.

Sometimes, keyword(s) may be associated with what we call a level of search, which is a structure as such as "table", "column" or values within a "record". In our system, we allow expressing the relationship between the words and the levels, which may be specified as "table", "column", "record" or "all"; the last one indicates a combination of the other three choices. For instance, the following statement represents a query in our system:

*find Honda \& red within record level.*

This query has two atomic subqueries (keywords), namely *Honda*, and *red*; and the record level has been specified for the search process.

To facilitate the search process, the system keeps the following information about the participating databases:

- \( IS(db) \), the level size of database \( db \), which is the total number of elements at each level \( l \) of \( db \), \( \forall db \in DB \) and \( l \in \{t, c, r\} \), where \( t, c \) and \( r \) stand for table, column, and record, respectively.
- \( f(w, l, db) \), the number of elements that contain keyword \( w \) at level \( l \) of database \( db \), \( \forall db \in DB \), \( l \in \{t, c, r\} \), and for all possible keywords \( w \).

Note that the system does not have the actual "inverted lists" corresponding to each keyword and each database, but just the lengths of these inverted lists. Further, the system does not need to store the value of \( f(w, l, db) \) explicitly if it is zero. This way, if no information is found by the estimator about a certain \( f(w, l, db) \), then it is assumed zero. In our implementation of the system, it is required that each database cooperates and periodically submits these frequencies to the system following some predefined protocol.

### 3.3 The Estimator Function

Given the values for \( f \) and \( IS \) of each database in a set of databases \( DB \), the system can use an estimator \( E \) to select a subset of \( DB \) to which a given query \( q \) could be submitted. This is possible by computing an estimation of the size of the result of \( q \) at level \( l \) of each database \( db \), denoted \( E(q, l, db) \). This guess should estimate the actual number of elements satisfying \( q \) at level \( l \) of \( db \), denoted \( RS(q, l, db) \), i.e., \( E(q, l, db) \approx RS(q, l, db) \). Once \( E(q, l, db) \) has been computed, the subset of databases relevant to query \( q \) is specified as follows: \( CE(q, db) = \{ db \in DB \mid E(q, l, db) > 0 \} \).

This last equation identifies databases that have positive values for their estimators. These values assess the degree of relevance. Finally, the located subset is reported in descending order by the computed estimator values.

We applied the independent estimator in our application. It is built upon the assumption that keywords appear at any level of a database following independent and uniform probability distribution. Under this assumption, given a database \( db \), the total number of elements at each level \( l \) of \( db \), and any \( n \) keywords \( w_1, \ldots, w_n \), the probability that level \( l \) of \( db \) contains all of \( w_1, \ldots, w_n \) is given by:

\[
\frac{f(w_1, l, db)}{IS(db)} \times \cdots \times \frac{f(w_n, l, db)}{IS(db)}
\]

So, the estimated number of elements, which are present at level \( l \) in \( db \), and satisfy the query:

\[
q = \text{find } w_1 \land \neg w_n \text{ within level } l,
\]

is calculated according to the independent estimator as:

\[
E(q, l, db) = \prod_{i=1}^{n} \frac{f(w_i, l, db)}{IS(db)}
\]

and then the set \( CE(q, db) \) is computed using the corresponding equation given above.

The estimator we used in our system extends that of GLOSS [8]; our extension takes the level of search into account. To illustrate our approach, we will refer to the example database frequency information given in Table 1, which contains a portion of the information kept by our running prototype for three example databases. Finally, the following example illustrates the database selection process described above.

**Example 3.1 (relevant search: AND operator).**

Consider three databases, \( db_1, db_2 \) and \( db_3 \), and suppose that the system has collected their related statistics, a part of which is shown in Table 1. Further, assume that the system received the query:

\[
q = \text{find Honda \& red within record level.}
\]

This query is specified in our system using the graphical user interface as shown in Figure 2; it searches at the record level for both words *Honda* and *red*. The system has to estimate the number of matching words at the specified record level in each of the three databases. The computation for the three databases is performed using the independent estimator, where \( l = r \):

\[
E_{d1}(q, l, db_1) = \frac{2 \times 2}{60} = 0.2; \quad E_{d2}(q, l, db_2) = \frac{2 \times 1 \times 11}{60} = 2.00; \quad E_{d3}(q, l, db_3) = \frac{8 \times 0}{3} = 0.96.
\]

Since all the computed values are positive, then all the three databases are considered as relevant to \( q \). The estimator reports the databases in descending order based on the above computed values as, \( \{db_2, db_3, db_1\} \); this is the same order shown in Figure 2.

Issuing \( q \) to each individual database directly will return
Table 1: A portion of the database frequency information extracted from three example databases

the following results: $RS_1(q, l, db_1) = 1$, $RS_2(q, l, db_2) = 2$, and $RS_3(q, l, db_3) = 1$. This shows that the numbers assessed by the estimator are approximately relative - in most cases - to the actual number of elements at the specified level of each database $db$ satisfying $q$, i.e., $E(q, l, db) = RS(q, l, db)$.

In general, for the independent estimator $E$, we can say that for a given database $db$: if $f(w_1, l, db) = 0$, for some $1 \leq i \leq n$ and for a specific level $l$, then $E[w_1 \land \ldots \land w_n, l, db] = 0$. This means that database $db$, which satisfies this if-statement will not be included in the chosen subset of databases $C_E(q, db)$.

The second logical operator in our system is OR, denoted $\lor$; it is handled as follows. Given a database $db$, the total number of elements at each level $l$, and any $2$ keywords $(w_1, w_2)$ - this may be generalized to $n$ keywords - the probability that level $l$ of $db$ contains $w_1 \lor w_2$ is computed as:

$$f(w_1, l, db) + f(w_2, l, db) - f(w_1, l, db) \times f(w_2, l, db)$$

So, the estimated number of elements, which are present at level $l$ in $db$ and satisfy the query: $q = \text{find } w_1 \lor w_2 \text{ within level } l$, estimated according to an independent estimator $E(q, l, db)$ computed as:

$$\frac{f(w_1, l, db)}{IS(db)} + \frac{f(w_2, l, db)}{IS(db)} - \frac{f(w_1, l, db)}{IS(db)} \times \frac{f(w_2, l, db)}{IS(db)} \times IS(db)$$

As a result, the chosen subset $C_E(q, db)$ is then computed using the corresponding equation given above. The OR operator is illustrated in the following example.

**Example 3.2 (Relevant Search: OR Operator).** Consider again the three databases that have a part of their related statistics given in Table 1, and assume the system received the query:

$q = \text{find Honda} \lor \text{Red within record level}$

This query searches at the record level for any of Honda or Red; the system has to estimate the number of matching words at the specified level in each of the three databases.

Since all the three databases contain both words at the record level, then using the independent estimator, we perform the following computations, where $l = r$:

$$E_1(q, l, db_1) = \left[\frac{1}{50} + \frac{1}{50} - \frac{1}{50} \times \frac{1}{50}\right] \times 50 = 6.8$$

$$E_2(q, l, db_2) = \left[\frac{49}{100} + \frac{11}{100} - \frac{49}{100} \times \frac{11}{100}\right] \times 100 = 27.91$$

$$E_3(q, l, db_3) = \left[\frac{49}{50} + \frac{1}{50} - \frac{49}{50} \times \frac{1}{50}\right] \times 75 = 16.04$$

As all the computed values are positive, then all the three databases are considered as relevant to $q$. The estimator reports the databases in descending order as, $db_2, db_3, db_1$.

To compare these estimated values with the actual results, assume that we issue query $q$ to each individual database; this leads to the following results: $RS_1(q, l, db_1) = 6$, $RS_2(q, l, db_2) = 27$, and $RS_3(q, l, db_3) = 16$. As in the case of the AND operator, here also we can see that the numbers assessed by the estimator are approximately relative - very close in this case- to the actual numbers of elements satisfying the given query at the specified level of each database.

Finally, let’s consider the case of having the NOT operator, denoted "\neg". Consider a database $db$, the total number of elements at level $l$, and any $2$ keywords $w_1$ and $w_2$ - this may also be generalized to $n$ keywords - the probability that level $l$ of database $db$ contains $w_1 \land w_2$, which is equivalent to $w_1 \land w_2$ is given by:

$$f(w_1, l, db) \times (1 - f(w_2, l, db))$$

So, consider the query: $q = \text{find } w_1 \land w_2 \text{ within level } l$, the estimated number of elements present at level $l$ in database $db$ and satisfy query $q$ is given according to the independent estimator by:

$$E(q, l, db) = \left[\frac{f(w_1, l, db)}{IS(db)} \times (1 - \frac{f(w_2, l, db)}{IS(db)})\right] \times IS(db)$$

Then, the chosen subset of databases $C_E(q, db)$ is computed using the corresponding equation given above.

**Example 3.3 (Relevant Search: NOT Operator).** Consider the three databases that have a part of their related statistics given in Table 1, and assume the system received the query: $q = \text{find Honda and not Red within record level}$.

This query searches the record level to locate records that contain the words Honda and not Red. The system has to estimate the number of matching words at the specified level in each of the three databases. As a result, the following values are computed using the independent estimator, with $l = r$:

$$E_1(q, l, db_1) = \left[\frac{1}{50} \times (1 - \frac{1}{50})\right] \times 50 = 4.8$$

$$E_2(q, l, db_2) = \left[\frac{49}{100} \times (1 - \frac{11}{100})\right] \times 100 = 16.91$$

$$E_3(q, l, db_3) = \left[\frac{49}{50} \times (1 - \frac{1}{50})\right] \times 75 = 7.04$$

Since the three returned values are positive, then all the three databases are considered as relevant to the query; they are returned in the following order: $db_2, db_3, db_1$. On the other hand and for the purpose of comparing the results, in case the same query was issued to each individual database, then the following results would have been obtained: $RS_1(q, l, db_1) = 4$; $RS_2(q, l, db_2) = 16$; and $RS_3(q, l, db_3) = 7$, which are only slightly different from the above estimated values. This demonstrates the power of the developed estimation based approach.

4. **Evaluation Parameters**

To evaluate the set of databases that the system reports for a given query, we present a framework based on the precision and recall metrics in IR, which could be expressed as follows. Given a query $q$ and a set $S$ of documents relevant to $q$, precision $P$ is the fraction of documents in the answer to $q$ from $S$, and recall $R$ is the fraction of $S$ in the answer to $q$.

We use these notations to define metrics for the database selection problem: for a given query $q$ and a given set of relevant databases $S$, precision $P$ is the fraction of databases in the answer to $q$ which are also in $S$, and recall $R$ is the fraction of $S$ in the answer to $q$.

Let $DB$ be a set of databases and $q$ be a query. In order to evaluate an estimator $E$, we need to compare its prediction against what is actually the right set of databases.
from $DB$, denoted $R(q, DB)$, which are relevant to query $q$.

We define the right set $R(q, DB)$ as the set of all databases in $DB$ such that the corresponding specified level contains information that matches the set of all items -databases in this context- relevant to the given query $q$. Formally:

$$R(q, DB) = \{db \in DB : RS(q, l, db) > 0 \}$$

where $RS(q, l, db)$ is the actual number of elements, which are present at level $l$ of database $db$ and satisfy query $q$.

Finally, to evaluate the set $CE = (q, DB)$ approximates $R(q, DB)$, we defined the following two functions $P_E^R$ and $R_E^R$, based upon the precision and recall parameters, where $P_E^R$ is the fraction of selected databases which are right, and $R_E^R$ is the fraction of the right databases selected according to the relevant search criteria. Formally,

$$P_E^R = \frac{|CE(q, DB) \cap R(q, DB)|}{|CE(q, DB)|} , \quad |CE(q, DB)| > 0 \quad (7)$$

$$R_E^R = \frac{|CE(q, DB) \cap R(q, DB)|}{|R(q, DB)|} , \quad |R(q, DB)| > 0 \quad (8)$$

**Example 4.1 (Evaluation Parameters).** Assume that the system received the query: $q = \text{find Mercedes 2000 within record level}$. For this query, the following estimator values are computed: $E_1(q, l, db_1) = 0.48$, $E_2(q, l, db_2) = 3.24$, and $E_3(q, l, db_3) = 0.747$, which are all positive, i.e., the three databases are considered as relevant to the query, and hence the reported subset is $\{db_2, db_3, db_1\}$.

If we issue $q$ to each individual database, we will get the following results: $RS(q, l, db_1) = 0$, $RS(q, l, db_2) = 2$ and $RS(q, l, db_3) = 1$. So, according to the definition of the right relevant subset, $R(q, DB) = \{db_2, db_3\}$. The considered precision and recall parameters are computed as:

$$P_E^R = \frac{|\text{db}_2, \text{db}_3|}{|\text{db}_2, \text{db}_3|} = \frac{2}{3} = 0.67$$

$$R_E^R = \frac{|\text{db}_2, \text{db}_3|}{|\text{db}_2, \text{db}_3|} = \frac{2}{3} = 1$$

$P_E^R(q, DB) = 0.67$, means two third of the selected databases are in the right set. On the other hand, $R_E^R(q, DB) = 1$, means all of the databases in the right set are included in the selected set CE(q, DB).

Note that, by definition of $P_E^R$, if $|CE(q, DB)| = 0$, we may consider $P_E^R(q, DB) = 1$, in order to capture the fact that no database in $CE(q, DB)$ is not right. Similarly, by definition of $R_E^R$, we may consider $R_E^R(q, DB) = 1$, whenever $|R(q, DB)| = 0$, since in this case all of the Right databases are included in $CE(q, DB)$.

5. SUMMARY AND CONCLUSIONS

In this paper, we described a mechanism that enables users to find information of interest and databases that contain such information without any a priori knowledge about its location. This mechanism employs the statistical information extracted from each local database to estimate the number of potentially useful databases. Our estimation methods are based upon established statistics theory. This mechanism also accomplishes these goals with no effect on the autonomy and heterogeneity of the participating databases. Further, the flexibility and ease with which the databases join and leave the system, and organize themselves within the system is one of the most important features of our system.

To support relational query processing using keyword search, we gave the necessary extensions to the inverted file to take the structure of relational databases into account. A prototype for each part of the architecture has been developed and implemented: building the index and summary tables for each component database; extracting the index and summary tables from individual databases; building the union index and the union summary tables; and building the main system with a web-based interface. Finally, the results of the conducted experiments are encouraging.

Currently, we are working on different evaluation parameters and different search criteria, including sample single database, best single database and best k-databases. The latter criteria are important for users not interested in the whole result and/or when the result is to be delivered within pre-specified limited time. We are also planning to evaluate the storage requirements of the proposed approach.

6. REFERENCES


