Scalable Publish and Search Strategies for Effective Keyword-Based Database Search

D.J.S. Sako & O.E. Taylor
Department of Computer Science
Rivers State University
Port Harcourt, Nigeria.
sunday.sako@ust.edu.ng, taylor.onate@ust.edu.ng

Abstract
Data, the backbone of all enterprise application, is often locked up in one or more databases. Looking for particular information across multiple databases can be quite tedious and time consuming if all databases in the domain are all searched for information that may not be found in some of them. In this paper, we present an approach and the implementation of a system to query and search multiple relational databases to produce more effective and efficient result. Our approach depends on registering an existing database appilication with the system to enable the database or part of it for keyword search, identifying, ranking and searching only the published/registered databases relevant to a given query and is most likely to provide useful results. A database is relevant if it contains some information to participate to the answer of the raised query. The implication is that databases with zero or negative score do not contain the query terms and need not participate in the search process thereby reduceing the time required to search the databases for keywords, as only participating database(s) will be searched for query terms. We discussed the implementation of our system including results of experimental evaluation to demonstrate the scalability and effectiveness of our system.

Keywords: *Publish, Scalability, Query, Search, Crawling, Indexing, and Relational database.*

1. Introduction

When a user submits a query, which usually consists of one or more keywords that reflect the user's information needs, to a search engine, the search engine returns a list of items from the set of web pages covered by the search engine. Usually, retrieved information are displayed to the user based on how well they are deemed to match with the query, with better-matched ones displayed first (Yu & Meng, 2003). Since a lot of information is stored in databases (and not as Hyper Text Markup Language documents), it is important to provide a similar search paradigm for databases, where users can query a database without knowing the database schema and database query languages such as Structured Query Language (SQL).

Databases provide the content storage for many sites, which dynamically create web pages around them. Intranets often contain large amount of information stored in database as well (Anto, 2015). Agrawal, Chaudhuri, and Das (2002) had noted that a significant amount of the world's enterprise data resides in these relational databases and that it is important that users be able to seamlessly search and browse information stored in these databases as well. According to them, searching databases on the internet and intranet 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

A study by (Asadi & Jamali, 2004) estimated that 80% of the data on the World Wide Web is in the hidden web. The hidden web refers to information that can be accessed on the World Wide Web, but which current search engines cannot find (nor can the internet users who subsequently use those search engines). It contains Web pages that are not publicly indexable. A major part of the hidden web consists of information tucked away in databases.

Agrawal, Chaudhuri, Das and Gionis (2003) asserted that there is the need to develop query-processing strategies that build on a crucial characteristic of IR-style keyword search: only the few most relevant matches –according to some definition of "relevance"– are generally of interest. Consequently, rather than computing all matches for a keyword query, which leads to inefficient executions, the techniques should focus on the top-*k* matches for the query, for moderate values of *k*. In a work done by (Agrawal, Chaudhuri, Das, 2002) when a DBXplorer is given a set of query keywords, it returns all the rows (either from single table, or by joining tables connected by foreign-key joins) such that each row contains all the keywords.

Given a set of published/registered relational databases, the system offers 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 (Hassan, Alhajj, Ridley, and Barker, 2004). Clearly, if databases are optimally ranked for a query, then it is sufficient to search the first k database D with the query. The implication is that databases with zero or negative score do not contain the query terms and need not participate in the search process. This optimization technique greatly reduces the time required to search the databases for keywords, as only participating database(s) will be searched for query terms.

With its capability to handle multiple databases, search any number of these databases without changing code, and to provide a common search interface for applications (databases) without the need for application interface themselves, the task of searching information scattered across databases is simplified.

In this paper, we discuss the implementation of an approach to publishing, crawling/indexing and searching multiple relational databases to produce more effective and efficient result. The system is implemented using commercial relational databases and Web Server, and users can interact with it via a browser front-end.

The rest of the paper is organized as follows: In section 3, we define the problem of publishing and keyword search over multiple databases. In section 4, we present the proposed approach and an overview of the search system including brief discussion and the algorithms of major modules or processes. Section 5 presents the implementation of the search system and experiments that demonstrate the scalability and effectiveness of our solution. An appendix containing some screenshots of user interactions with the search system is also included.

2. Related Work

Adapting keyword search to structured databases has already attracted the attention of several researchers. These include works done by (Agrawal, Chaudhuri, and Das, 2002), (Bhalotia, Hulgeri, Nakhe, Chakrabari & Sudarshan, 2001), (Dar, Entin, Geva & Palmon, 1998), (Sarda & Jain, 2001), (Hassan, Alhajj, Ridley & Barker, 2004) and a number of others.

DBXplorer by (Agrawal, Chaudhuri, and Das, 2002), BANKS by (Bhalotia, Hulgeri, Nakhe, Chakrabari & Sudarshan, 2001), DISCOVER by (Hristidis & Papakonstantinou, 2002),

DataSpot by (Dar, Entin, Geva & Palmon, 1998) and Mragyati by (Sarda & Jain, 2001) had discussed only the case where there is a single database and as asserted by (Hassan, Alhajj, Ridley & Barker, 2004), 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. (Hassan, Alhajj, Ridley & Barker, 2004) had extended the implementation to multiple databases where only databases relevant to a given query will be accessed and their proposed solution to the usefulness estimation problem is an extension of the approach of (Gravano, Garcia-Molina, & Tomasic, 1999) taking into account the level of search and the structure of the relational database.

(Yu, Li, Solins & Tung, 2007) studied the database selection problem for relational database and proposed a method that effectively summarises the relationships between keywords in a relational database based on its structure. As an illustration, they looked at two example databases DB1 and DB2 shown in Figure 1, in which the arrowed lines drawn between tuples indicate their connections based on foreign key references. Suppose we are given a keyword query Q = {*multimedia; database; V LDB*}.

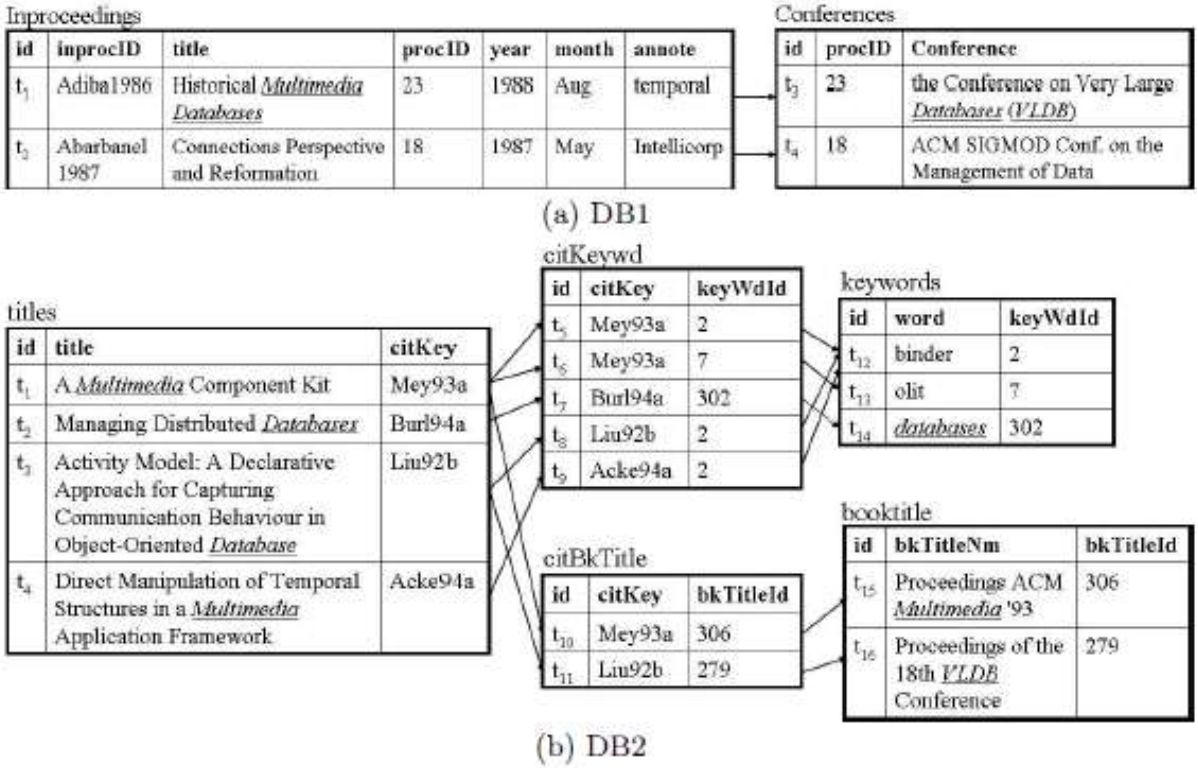


Figure 1: Example of Databases (Yu, Li, Solins & Tung, 2007)

It is observed that DB1 has a good result to Q, which is the result of joining tuple t_1 with t_3. On the contrary, DB2 cannot provide relevant results to Q - there are no trees of connected tuples containing all the query keywords. But, if we evaluate the two databases for Q based on the keyword frequency style summaries (denoted as KF-summary in their work, and KF-summary (DB1) = {... *multimedia:1, database:2, VLDB:1,* ...}, and KF-summary(DB2) = {... *multimedia:3, database:3, VLDB:1,* ...}), DB2 will be selected over DB1. They, therefore, argued that the usefulness of a relational database in answering a keyword query is not only decided by whether it has all the query keywords, but more importantly, it depends on whether the query keywords can be connected meaningfully in the database.

In this work we have extended these techniques to keyword search over multiple databases by taking into account the scalability of database publishing and search. The system can search any number of databases without changing code and new database can be published/registered for crawling/indexing without affecting the overall performance of the system, takng into account the fact that only databases published and relevant to a given query will be ranked and accessed.

3. Problem Definition

We now define the problem of publishing and keyword search over multiple databases wihin a specific domain.

1. Given a set of databases DB_1, …, DB_n, to be registered/published, effectively produce a scalable publishing strategy that will fit the new database(s) into the already existing search system setup in such a way that their addition would not negatively affect the overall performance of the system.

2. Given set of published databases DB_1, …, DB_n, a keyword Query Q, and a scoring function, effectively produce the top-*k* answers for Q from DB_1,…,DB_n, such that these answers closely approximate the ideal top-*k* results for Q. Each database has *n* relations $R_1, \ldots, R_n$. Each relation R_i has m_i attributes $a_1^i, \ldots, a_{mi}^i$, a primary key and possibly a directed graph that captures the foreign keys into other relations.

The rest of the paper describes the solution to these problems

4. Methodology

4.1 The Proposed Approach

The methodology that we propose enabling keywords search requires retrieving the *n* most similar documents across multiple databases for a given query consists of the following steps.

- *Publishing/registering the databses.* Database(s) that will participate in the search process is initially registered as part of the search system.
- *Crawling and indexing the published database:* The search system builds up its own information base when an application database is initially registered with it. It provides interfaces to select tables/columns within the database to crawl and index.
- Rank the published databases using some ranking or scoring function – to reduce the search time and optimize the search process. That is a database with a higher rank will be searched before a database with a lower rank.
- Search the published databases according to their rank in a certain manner to retrieve *n* documents. Rank the published databases using some ranking or scoring function. Search only the most relevant databases (higly ranked) to find the (approximately) top k results. A database is relevant if it contains some information to participate to the answer of the raised query.

These combined publishing and search techniques greatly reduce the time required to search the databases for keywords, as only registered, participating and relevant database(s) will be searched for query terms.

4.2 System Architecture

We now describe the architecture of our search system. The system provides a browser-based

interface. The user supplies a query string consisting of a number of keywords and the query results are also displayed in the browser. The main building blocks of the system are shown in figure 2.

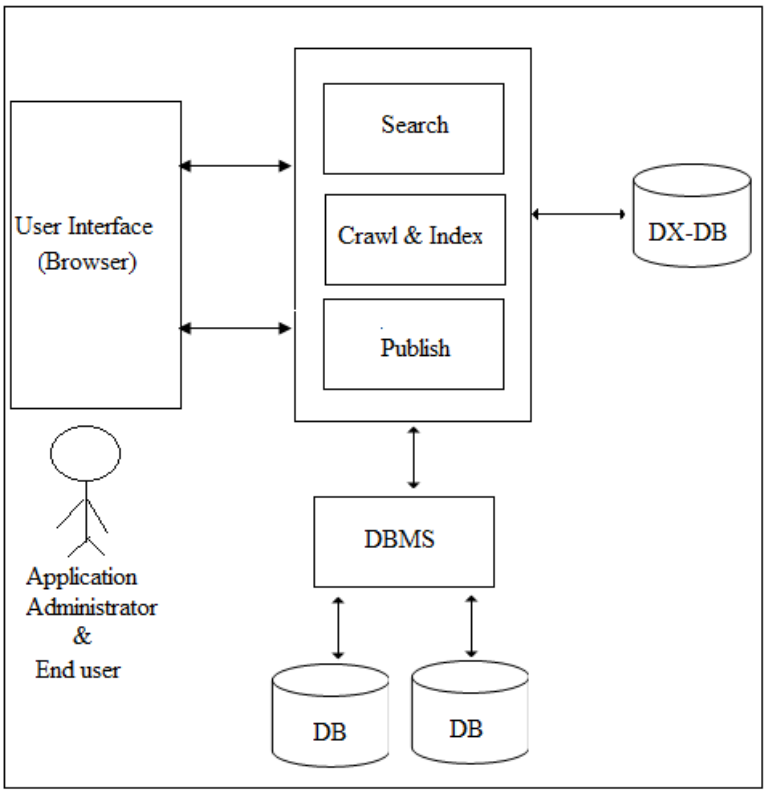


Figure 2: The Architecture of the Search System.

4.2.1 User Interface (Browser)

The client component provides a simple administrative console and the search interface. The application admininstrative console provides for the publishing/registration of databases, making all the indexing configurations for a database and the scheduling of the indexing process. The search interface provides for user to enter the query terms and a default view for showing the search results. The administrative console functions are hidden from the general end-users who only issue query to the system and get the result of the query.

4.2.2 Publish Module

An existing database application is first registered with the system. The publish component or module builds the metadata by consulting the database catalogue. It provides interfaces to: select database(s),

The search system builds up its own information base when an application database is initially registered (published) with it. This information base contains metadata, 'value-attribute' mapping, etc.

4.2.3 Crawler/Indexer

The list of databases that needs to be search-enabled is given as input to the crawler with the configuration parameters specific to each of the data sources (through the publish module). The crawler takes the input database, scans through the tables and the data available in each

of the given database and builds up its own information base and creating certain index information about the database and computing the database statistics. The important index information stored about the data includes:

- The location of the occurrence of a word
- The count of the occurrences of a particular word
- The type of the column in which a particular word occurs
- The relationship between different tables.

The crawler also has interface that enables you to select tables/columns within the database to crawl and index. The system requires that each database cooperates and periodically updates this index information, following some predefined protocol.

4.2.4 Search/Query Processor

Search component takes care of fetching the right information from the participating database(s), given the search words, based on the index information.

For a given set of keywords, the search component provides interfaces to retrieve matching databases from a set of published databases, and selectively identify tables, columns/rows that need to be searched within each database identified based on the index information. The specific interfaces include for a given set of keywords:

- Find all the matching databases
- For a given set of ranked matching database(s), find all rows in the database/tables that contain all or most of the keywords.

(Sarda & Jain, 2001) provides discussion on how user query is analyzed and translated in which the query module uses application vocabulary and stop words list to translate user's terms into internal values wherever necessary and construct query tree.

4.3. Ranking the Databases

Consider a set of relational databases, (DB_1, $DB_{2, \ldots,}$ DB_n), that have been registered and indexed with the system. Given a keyword query Q = (k_1, k_2, …, k_q), we would like to rank the databases based on their usefulness to answer query Q. To evaluate the set of databases that the system reports for a given query, (Hassan, Alhajj, Ridley & Barker, 2004) presented a framework which is adopted in this work. It is based on the precision and recall metrics in information retrieval (IR), which could be expressed as follows. Given a query q and a set S of documents relevant to q, precision is the fraction of documents in the answer to q from S, and recall is the fraction of S in the answer to q.

These notations are used to define metrics for the database selection and ranking 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. There is the 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. The right set R(q,DB) is defined as the set of all databases in DB such that it contains information that matches the set of all items – databases in this context-relevant to the given query q.

Formally: R(q,DB) = {db ∈DB|RS(q,db) >0} where RS(q, db) is the actual number of elements, which are present in database db and satisfy query q.

Finally, to evaluate how the set C_E = (q,DB) approximates R(q,DB), (Hassan, Alhajj, Ridley and Barker, 2004) 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,

$$P_E^R = \frac{|C_E(q,DB) \cap R(qq,DB)|}{|C_E(q,DB|}, \qquad |C_E(q, DB)| > 0 \qquad (1)$$

$$R_E^R = \frac{|C_E(q,DB) \cap R(qq,DB)|}{|R(q,DB|}, \qquad |R(q,DB)| > 0 \qquad (2)$$

Consider three databases, *db₁*, *db₂*, and *db₃* and supposed that they have been published, indexed and their related statistics collected. Assume that the system received the query: *q* = *Hardy Thomas Cmputing*. For this query, we assume that the estimations, using the Estimator Function discussed in ((Hassan, Alhajj, Ridley & Barker, 2004), for all the databases are 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 get the following results: RS_1(q, db1) = 0, RS_2(q,db2) = 2 and RS_3(q,db3) = 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:

$$P_E^R = \frac{|\{db_2, db_3, db_1\} \cap \{db_2, db_3\}|}{\lceil\{db_2, db_3, db_1\}\lceil} = \frac{2}{3} = 0.67$$

$$R_E^R = \frac{|\{db_2, db_3, db_1\} \cap \{db_2, db_3\}|}{\lceil\{db_3, db_1\}\lceil} = \frac{2}{2} = 1$$

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

By definition of P_E^R, if |C_E(q, DB)|=0, we may consider P_E^R(q, DB)=1, in order to capture the fact that no database in C_E(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 C_E(q,DB).

With this, we can effectively rank a set of databases DB (DB_1, DB_2, …, DB_n) for a given keyword query. Specifically, the ranking is a mapping from DB to {1,2,…,N} such that rank(DB_i) < rank(DB_j) ⇔ rel(Q; DB_i) ≥ rel(Q;DB_j), where rel(Q;DB_i) denotes the relationship score of Q in DB_i.

Clearly, if databases are optimally ranked for a query, then it is sufficient to search the first k database DB with the query. The implication is that databases with zero or negative score do not contain the query terms and need not participate in the search process.

4.4 Algorithms

We now present algorithms for the core operations and processes in our system.

4.4.1 Publish

This module provides the necessary information that the crawler/Indexer needs to traverse a particular database. **D**atabase(s) are registered and enabled for keyword search through the following steps.

Algorithm PUBLISH
Input: database
Output: Information base

Begin
Identify a database to be registered with the system
Create information base that support keyword searches.
End

Figure 3: Publishing a Database

4.4.2 Crawling/Indexing

The Index table created is used at search time to efficiently determine the locations of query keywords in the database (ie the tables, columns, rows they occur in for a particular database)

Algorithm INDEXER
Inputs: Database(s) [to be initially registered with the system]
Outputs: An Index table, **DX**
Begin
Identify database, along with the set of tables and columns within the database to crawl.
//Compute Index table **DX**
for I = 1 to no of database published
scan database and for each table, T, in database
for each keyword **K** in column c
If **K** is a stop word then
ignore **K**
else
Insert(K, c,) into **DX** if it does not already exist.
Endif
End inner for loop
End outer for loop
End Outtermost for loop
End

Figure 4: Crawling and Indexing Algorithm

4.4.3 Search

Consider a query q, consisting of a set of keywords, 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 a scoring function, which scores how relevant each database in DB is with respect to q.

Algorithm SEARCH
Inputs: A query consisting of keywords k_1, $k_{2,\ldots,}k_k$
Outputs: All database rows matching all or some of the keywords

Begin
//preprocess query
Tokenize query terms/stream
Recognize query terms vs. special operators.
// Lookup the application vocabulary stop-words table for query conversion
//and stop words elimination
delete stop words (if any)
Stem words
Create query representation
Compute the score for each participating database in relation to q.
Sort/Rank the database according to the set results.
//Search Index table:
Look up index table to determine the tables, columns or cells containing query keywords
//Search for rows:
Construct and execute SQL statement to retrieve matching rows
End

Figure 5: Search Algorithm

5. Implementation and Experimental Results

5.1 Input to the system:

The input to the system principally is a query which consists of a set of keywords q (k_1 , . . . , k_m). Other input includes the database(s) to be published which is/are also periodically updated and indexed to reflect the current state of the database(s).

5.2 Output from the system

The major output from the system is the result which represents the relevant documents to the query q.

5.3 Experimental Evaluation

We implemented the techniques described and performed comprehensive experiments to evaluate the efficiency and effectiveness of our approach.

5.3.1 Data Set

Databases: Two databases are used in the experiments. The first is the Northwind Database from Microsoft that contains information about customers, employees, orders, order details, etc. The second is a publication database that contains information about stores, publishers, writers, sales, jobs, authors, titles, etc.

Query Set: We used a query set consisting of 1 to 5 keywords for evaluation. For each query, we identify a set of relevance answers in the published databases.

5.3.2 Testing and Evaluation of Results

Different parameters are used to measure the scalability, efficiency and effectiveness of the Search System and evaluate the performance.

1. Publishing, Crawling and Indexing Time.
2. Response time of the different programs in the search engine implementation, as it is scaled up to include more documents. A query response time is recorded.
3. The scalability of the indexed table and the performance of the search engine is evaluated with the set of databases by varying the number of keywords which are indexed and the distribution of the data size.
4. The standard information retrieval measures of precision and recall is computed to evaluate the results and the set of databases that the system reports for a given query.

Precision is the proportion of documents that the search engine retrieves for a keyword that are actually relevant to the query. It is computed by the formula:

$$\text{Precision} = \frac{\text{Retrieved documents that are relevant}}{\text{Total retrieved documents}} \quad (3)$$

Recall is the proportion of relevant documents (out of the "complete" collection) that are retrieved and computed by the formula

$$\text{Recall} = \frac{\text{Relevant document that are retrieved}}{\text{Total relevant keyword in collection}} \quad (4)$$

1. Publishing, Crawling and Indexing Time

It involves scanning the database, processing the data to store in the indexed table and populating the indexed table. . The publishing time is dominated by the time required to scan the data and populate the index table. Populating the indexed table takes about 70% of the total publishing time.

2. Number of Keyword in Search – Response Time.

We show that search scales with the number of query keywords. Figure 6 shows that the program response time is linear and will scale up well to include a number of keywords. The keywords were selected randomly from the underlying databases.

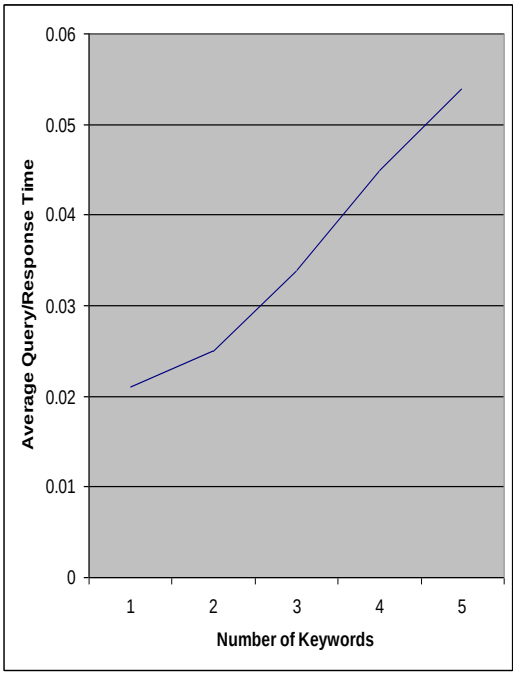


Figure 6: Query performance with keywords (Response time)

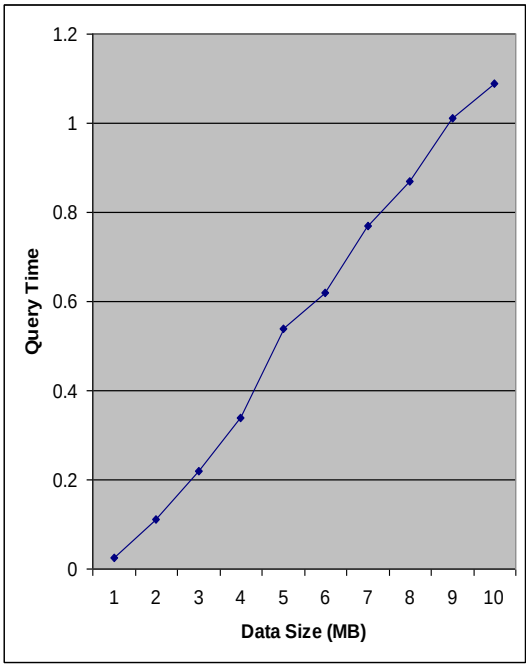


Figure 7: Query Performance with Data Size

2. Data Size and Distribution

We varied the data size from 2 to 10 MB real databases. In these databases the number of distinct keywords is proportional to the data size.

The space required by the index table for database data varies almost linearly with data size. The publishing time also increased almost linearly with data size. Figure 7 shows that the average query execution time increases very slowly as the data size is increased. This is due to a small increase in index table look-up time (recall that the index table sizes increase proportionately with data size).

3. Precision and Recall

Figure 8 shows precision and recall for the queries over the databases. Results show that average precision is 73% and average recall is 70%. When the precision values at each recall value are averaged over all test queries, an average recall-precision curve is obtained as shown in figure 9. This curve is used as the measure of the effectiveness of the system.

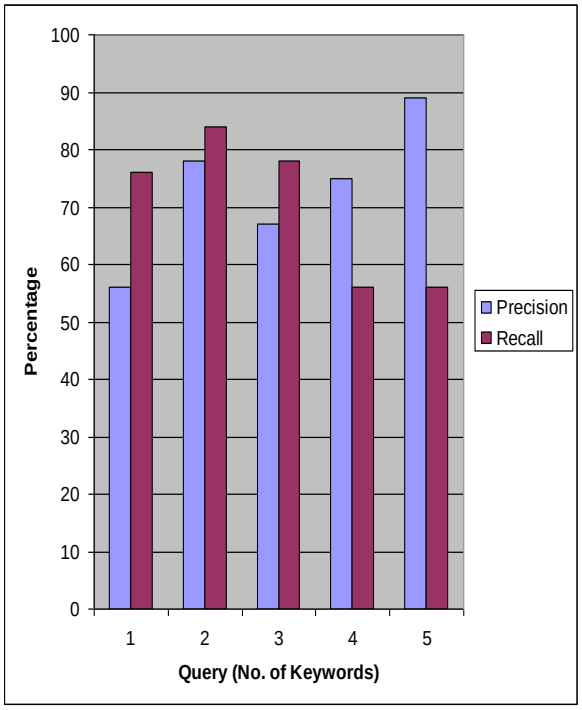


Figure 8: Queries' Precision and Recall

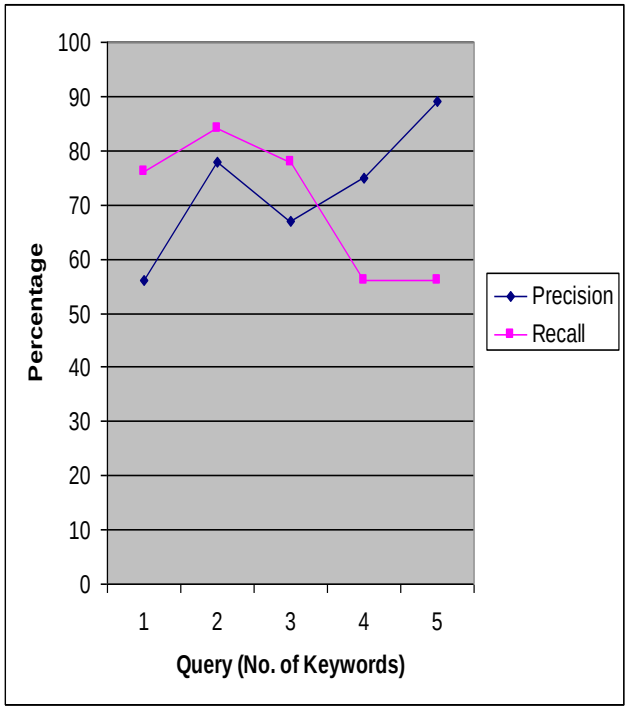


Figure 9: Average Recall-Precision Curve

5.4 Discussion of Results

In this study only small selection of search items were used to test the search system and the number of database published/registered, crawled and indexed for the search were reduced to two. However, the search system can be used to search any number of databases without changing code and any number of databases (or relations) can be registered for indexing and crawling without changing the overall performanceof the system. Within these limitations, the present results indicate that the system is optimized, scalable and efficient for database search.

6. Conclusion

In this paper we described an implementation of mechanism that enables databases to be registered seamlessly with the search system and allows users to find information of interest and databases that contain such information across multiple registered databases in the domain for a given query by selecting and rankings databases, searching the databases according to their rank in a certain manner to retrieve documents. Clearly, if databases are optimally ranked for a query, then it is sufficient to search the first k databases with the query. The implication is that databases with zero or negative score do not contain the query terms and need not participate in the search process. Unpublished databases cannot also participate in the search process.

These combined publishing and search techniques greatly reduce the time required to search the databases for keywords, as only registered, participating and relevant database(s) will be searched for query terms.

With its capability to handle multiple databases, search any number of these databases without changing code, and to provide a common search interface for applications (databases) without the need for application interface themselves, the task of searching information scattered across databases is simplified.

References

Agrawal. S., Chaudhuri, S., and Das, G. (2002). DBXplorer: A System for Keyword-Based Search over Relational Databases. In Proceedings of IEEE International Conference on Data Engineering, (ICDE). pp.152-163.

Agrawal. S., Chaudhuri, S. Das, G. and Gionis, A. (2003). Automated Ranking of Database Query Results. In Proceedings of the CIDR Conference.. pp 146-155.

Anto, Y (2015). Professional SEO Secrets. Ankara: Lambert Academic Publishing.

Asadi S. and Jamali, H.R. (2004). Shifts in Search Engine Development: A Review of Past, Present and Future Trends in Research on Search Engines. Webology, Volume 1, Number 2. pp. 14-21. http://www.webology.ir/2004/v1n2/a6.html

Bergman, M. (2001). The Deep Web: Surfacing Hidden Value. pp. 1-10 www.brightplanet.com/pdf/deepwebwhitepaper.pdf [Accessed 06/08/2017]

Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabari, S. and Sudarshan, S. (2001). Keyword Searching and Browsing in Database using BANKS. Technical Report, Indian Institute of Technology , Bombay. pp. 200-208.

Bruno, N. Gravano, L. and Marian A. (2002). Evaluating Top-k Queries over Web Accessible Databases. In ICDE. pp. 105 – 109.

Dar, S, Entin, G, Geva, S & Palmon, E. (1998). DTL's DataSpot.,VLDB.

Gravano, L., Garcia-Molina, H., and Tomasic, A. (1999). GIOSS: Text-Source Discovery over the Internet. ACM Transactions on Database Systems, 24(20). pp. 229-264.

Hassan, M., Alhajj R., Ridley M.J., and Barker K. (2004). Simplified Access to Structured. Databases by Adapting Keyword Search and Database Selection. ACM. pp. 261 – 273.

Hristidis, V., Gravano, L., and Papakonstantinou. Y. (2003) Efficient IR-Style Keyword Search over Relational Databases. In Proceedings of VLDB. pp. 120-135.

Hristidis, V. and Papakonstantinou. Y. (2002) DISCOVER: Keyword Search over Relational Databases. In Proceedings of VLDB. pp. 605 – 615.

Liu, F., Yu, C., Meng, W., and Chowdhury, A. (2006). Effective Keyword Search in Relational Databases. In SIGMOD, ACM. pp. 411-421.

Masermann, U. and Vossen, G. (2000). Design and Implementation of a Novel Approach to Keyword Searching in Relational Databases. In Proceedings of ADBIS-DASFAA Symposium on Advances in Databases and Information Systems. pp 171-184.

Sarda, N.L. and Jain, A. (2001) Mragyati: A System for Keyword Based Searching in Databases. Computing Research Repository. pp. 456-463. http://arxiv.org/abs/cs.DB/010052.

Sayyadian, M., LeKhac, H., Doan, A. and Gravano L. (2004). Efficient Keyword Search Across Heterogenous Relational Databases. pp. 345-352. http://arxiv.org/abs/cs.DB/010252.

Wheeldon, R., Levene, M. and Keenoy, K. (2004) DBSurfer: A Search and Navigation Tool for Relational Databases. Annual British National Conference on Databases. pp. 341-355. http://www.dcs.bbk.ac.uk/~mark/download/dbsurfer-short.pdf [Accessed 12/08/2017]

Yu, C & Meng, W. (2003). Web Search Technology. In The Internet Encyclopedia. Vol 3. Edited by Bidgoli, H. New Jersey: John Wiley & Sons, Inc.

Yu, B., Li, G., Solins, S. & Tung, A.K.H. (2007) Effective Keyword-Based Selection of Relational Databases. SIGMOD. ACM. Pp 450-463

APPENDIX: USER INTERFACES

We illustrate typical interactions with the search system via screen shots. The database(s) is/are search-enabled through registration/publication by the administrator (Figures A1-A4).

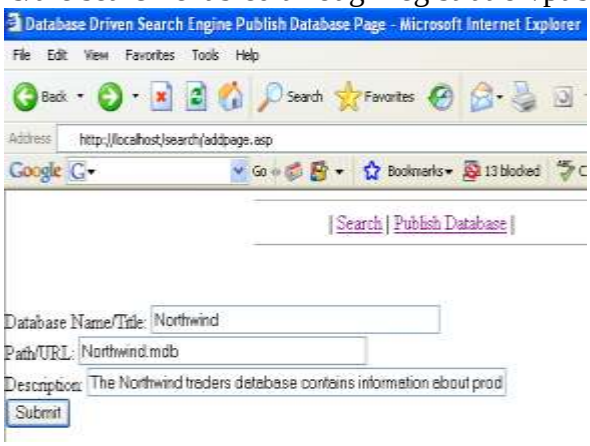


Figure A1: Publishing database

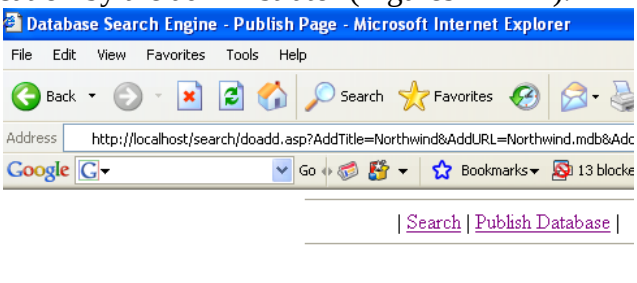


Figure A2: A Published database showing the size of the database

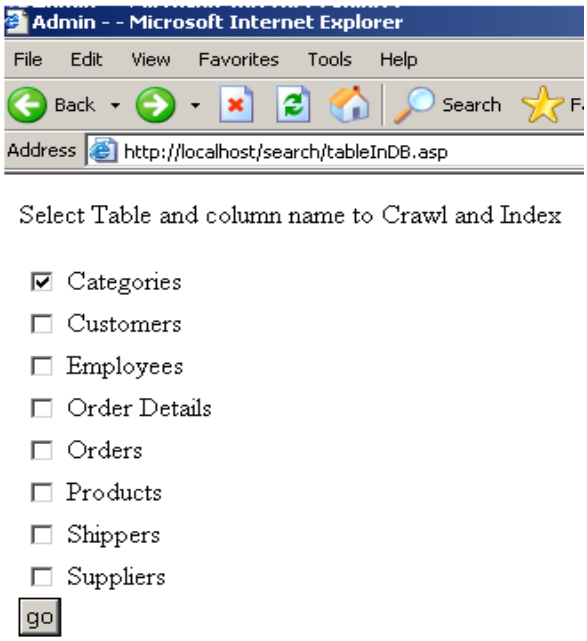
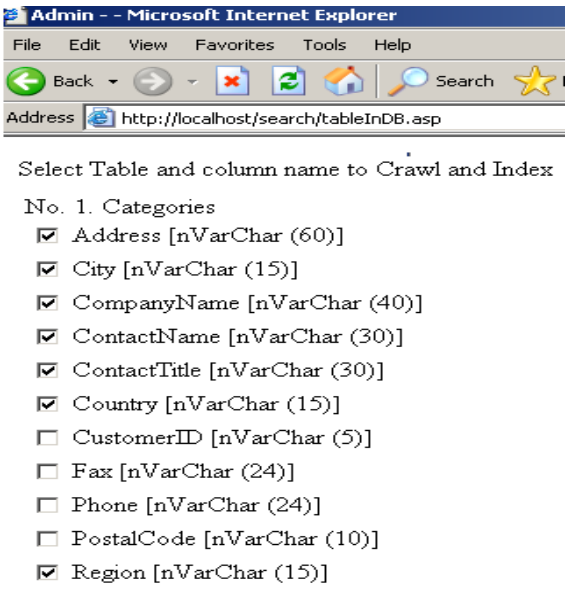


Figure A3: Database tables and columns Selection for crawling and indexing



Figur A4: Selecting the columns (fieldnames) of the selected table for crawling/indexing

Consider a query {hardy thomas computing}; perhaps the user is looking for a book by the author.

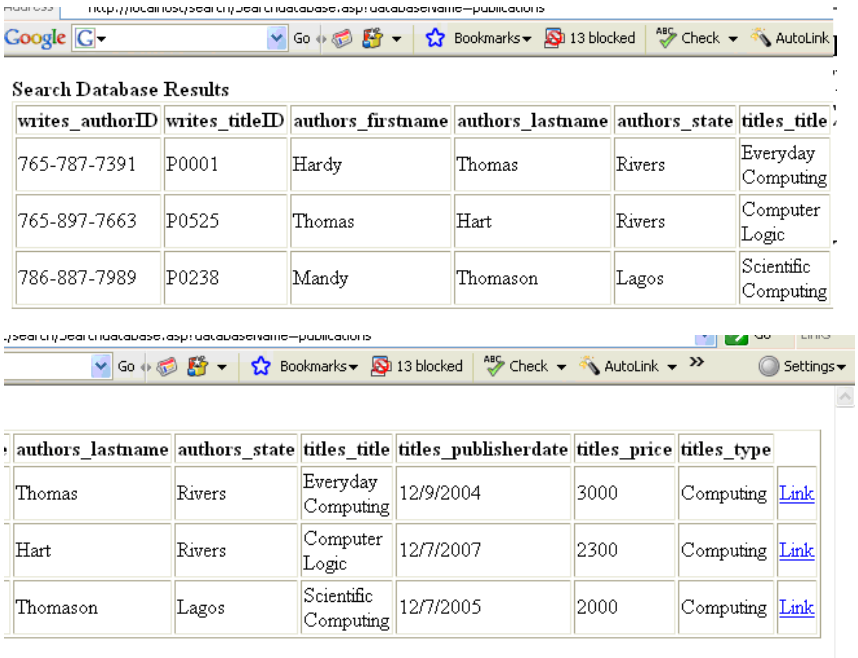
Figure A5: Searching

The system first returns the set of ranked databases that contain the given keywords, along with a brief description of each matching database. This aids the user in selecting a specific

database to be explored next. Figure A6 shows the search results indicating the matching databases similar to the interface in (Agrawal, Chaudhuri, & Das. 2002).

Figure A6: Matching Databases

In the next step, as shown in figure A7, the user explores matches within a selected database. It enumerates a list of subsets of tables and returns a ranked list of matching rows. The system also offers browsing capabilities whereby the user can explore further details of the retrieved rows by following links into related areas within the database. The link represents the foreign-key relationship between the tables.



Search Database Results

writes_authorID	writes_titleID	authors_firstname	authors_lastname	authors_state	titles_title
765-787-7391	P0001	Hardy	Thomas	Rivers	Everyday Computing
765-897-7663	P0525	Thomas	Hart	Rivers	Computer Logic
786-887-7989	P0238	Mandy	Thomason	Lagos	Scientific Computing

authors_lastname	authors_state	titles_title	titles_publisherdate	titles_price	titles_type	
Thomas	Rivers	Everyday Computing	12/9/2004	3000	Computing	Link
Hart	Rivers	Computer Logic	12/7/2007	2300	Computing	Link
Thomason	Lagos	Scientific Computing	12/7/2005	2000	Computing	Link

Figure A7: Matching rows