Indexing for Efficient Term-Based Search Using Mapping Functions

K.K.Basheer
Asst.Prof. CSE Dept.,
VITS, Proddatur,
Kadapa (dist), A.P. INDIA.

Srinivasa Rao.P
Asst. Prof, CSE Dept.,
GMRIT, Rajam, Srikakulam(dist)
A.P.,INDIA.

V. Venkata Ramana
Asst. Prof., CSE Dept.,
SSITS, Raychoti,
Kadapa (dist), A.P. INDIA.

ABSTRACT
The transformation from the received item to the searchable data structure is called Indexing. Indexing (originally called Cataloging) [11] is the oldest technique for identifying the contents of items to assist in their retrieval. The basic difference between the existing methods and the one discussed here is that these methods rely on a structure of web page linkages that lead from or to the indexed page. In contrast, our method uses the content of the pages linked to or from the indexed page for indexing. So, our method uses a structure of words used by the linked pages, whereas the current methods use the structure of the connections between linked pages.

GENERAL TERMS
Mapping function, DB2 Net Search Extender

KEYWORDS
Indexing, Catalog, Mapping, Crawl, Real time-scalable, Data Centre, Audit Logs, Conventional Hosting, Bandwidth, Administrative Privileges.

1. INTRODUCTION
Experimental results show that index size is manageable and keyword query response time is interactive for typical queries [2]. Update experiments demonstrate that our concurrent incremental update mechanism does not significantly hinder query performance [9]. We also present a user study confirming the superiority of keyword-based search over query processing for a range of database retrieval tasks.

By incorporating more structure into text objects, we can exploit search met capabilities for semi structured data [5]. We are going to investigate how to generate text objects with structural information, focusing first on "**How to mapping the processing tokens using efficient indexing mechanism**".

Indexing Performance: **We plan** to investigate better computation sharing when constructing text objects, as well as using parallelism to speed up indexing using index mapping mechanism.

We dealt with automatic indexing based on

A probabilistic model of the distribution of word tokens within a document (text) [8]; here we will be concerned with the distribution of index terms over the set of documents making up a collection or file. We shall be relying heavily on the familiar assumption that the distribution of index terms throughout the collection, or within some subset of it, will tell us something about the likely relevance of the any given document or token.

We shall attempt to use simple probability theory to tell us what a matching function should look like and how it should be used. The arguments are mainly theoretical but in my view fairly conclusive. The only remaining doubt is about the acceptability of the assumptions, which we shall try and bring out as we go along it. The data used to fix such a matching function are derived from the knowledge of **the distribution of the index terms** throughout the collection of some subset of it. If it is defined on some subset of documents then this subset can be defined by a variety of techniques: sampling, clustering, or trial retrieval. The data thus gathered are used to set the values of certain parameters associated with the matching function. Clearly, should the data contain relevance information then the process of defining the matching function can be iterated by some feedback mechanism [6]. In this way the parameters of the matching function.

Can be 'learnt'. It is on matching functions derived from relevance information that we shall concentrate. It will be assumed in the sequel that the documents are described by binary state attributes, that is, absence or presence of index terms [3].

2. RELATIONAL DATABASE MANAGEMENT SYSTEMS BASED SEARCH VERSUS KEYWORD BASED SEARCH
Relational Database Management Systems provide a convenient data model and versatile query capabilities over structured data. However, casual users must learn SQL and know the schema of the underlying data even to pose simple searches. For example, suppose we have a customer order database, shown in Figure 1, which uses the TPC-H [4] schema in Figure 2. We wish to search for Bob's purchases related to his aquarium fish hobby. To answer this query, we must know to join the Customer, Line item, Orders, and Part relations on the appropriate attributes, and we must know which attributes to constrain for "Bob" and "fish". The following SQL query is formulated:

SELECT c.custkey, o.orderkey, p.partkey FROM Customer c, Line item l, Orders o, Part p

WHERE c.custkey = o.custkey AND l.partkey = p.partkey AND o.orderkey = l.orderkey AND c.name LIKE '%Bob%' AND p.name LIKE '%fish%'

In our example database instance, this query returns a single tuple _custkey, orderkey, partkey_ of _100, 10001, 10_. In contrast, keyword-based search, exemplified by web search engines, offers an intuitive interface for searching textual content that requires little knowledge about the underlying data. In this paradigm, for the above example a user should be able to enter the keywords "Bob fish" and find the relevant information from the results.

3. WORKING OF KEYWORD BASED SEARCH

The general architecture of the EKSO (Efficient Keyword Search through Offline Indexing) consists of three main components. This expressive keyword-based search over interconnected relational database content that operates within the database engine and performs full offline indexing for highly efficient searches

1. The **Crawler** traverses the entire database offline, constructing virtual documents from database content.
2. Virtual documents are sent to the **Indexer**, which builds an inverted index over the virtual documents in the usual Information Retrieval (IR) style
3. At search time, the **Keyword Search Processor** takes a keyword query and probes the inverted index (again in an IR style) to find the virtual documents satisfying the query.

3.1 TYPES OF CRAWLERS

Verity crawls the content of relational databases and builds an external text index for keyword searches,

1. As well as external auxiliary indexes to enable parametric searches.
2. Data Spot extracts database content and builds an external, graph-based representation called a hyper base to support keyword search. Graph nodes represent data objects such as relations, tuples, and attribute values.

Part

Part Key	Name	.
10	Discus fish	
11	Apple Power book	
12	Snow showel	

Supplier

Part Key	Name	Address	..
1000	Acne	Wisconsin	
1001	Widgets	California	

Partsupp

Part Key	Supp Key	.
10	1000	
11	1001	
12	1000	
12	1001	

Lineitem

Order Key	Part key	Supp Key	Line number	..
10001	10	1000	2	
10002	11	1001	1	
10003	12	1000	3	

Orders

Order Key	Cust key	.
10001	100	
10002	100	
10003	101	

Customer

Cust Key	Name	Address	..
100	Bob	Madison	
101	Alice	San Francisco	

Fig: Example database instance

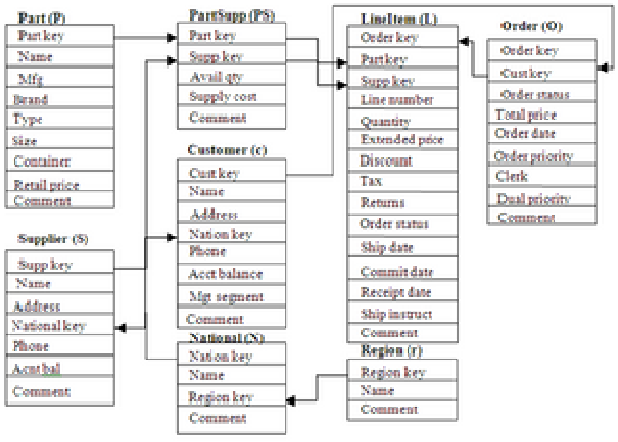

Figure 2: TPCH Schema

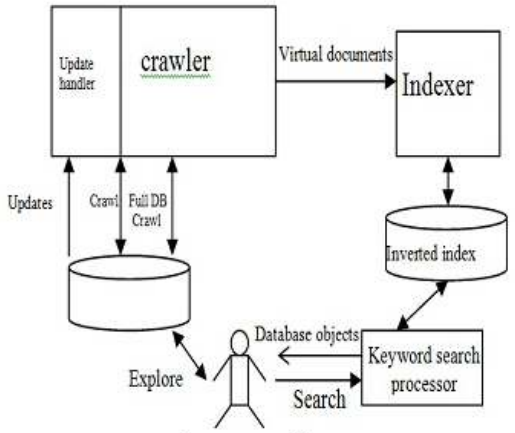

Figure 3: Architecture

3. DbSurfer [19] indexes the textual content of each relational tuple as a virtual web page.

3.2 ESKO SYSTEM

EKSO, is an instantiation of the general architecture we propose for keyword based search over structured databases

4. TEXT OBJECTS AND VIRTUAL DOCUMENTS

Our goal is to perform a full crawl of the database, generating virtual documents from database content that

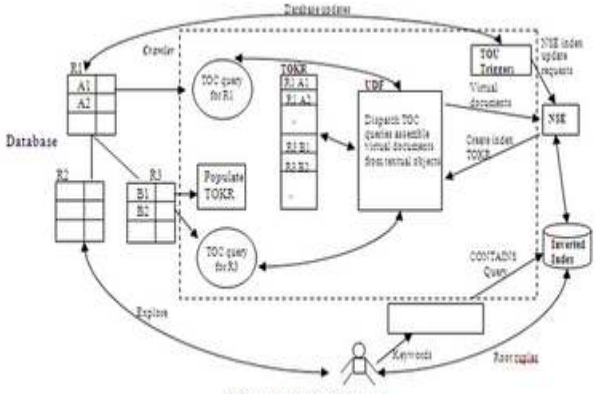

Figure 4: ESKO System

define the granularity of text search and retrieval in the system.

In ESKO system we separate two concepts

Text objects are generated from interconnected tuples based on joining relational tables.

Virtual documents are generated from text objects by concatenating their string attributes.

This separation has two advantages

1. We can use SQL queries, which are optimized automatically by the DBMS, to generate text objects.

2. We can independently plug in different functionality for generating text objects from the database, or for generating virtual documents from text objects.

4.1 Root relations and tuples

Root tuples are defined as all of the tuples in *root relations*. Root relations can be designated automatically or set by the database administrator.

Alternate definitions could be used for text objects from a given root tuple. For example, we could follow equijoins as well as foreign-key joins (with a second pass to handle

Dangling tuples), and we could backtrack.

4.2 Virtual Documents

After a text object has been computed, we concatenate all textual attributes of all tuples in the text object to form the virtual document. The order of the tuples selected during traversal is nondeterministic due to SQL semantics, and in the current system we are not concerned about the concatenation order. However, if we extend the system to richer virtual documents, as we will discuss next, we may need to be careful about concatenation order.

4.3 Indexer

The Indexer is the box labelled NSE in the right-hand portion of Figure 4. It is managed by the DB2 Net Search Extender (NSE). NSE is a full-featured text search engine integrated with the IBM DB2 DBMS product. NSE supports the creation of text indexes on individual columns, and can index the output of applying a user-defined function to a column, which is the feature we exploit

4.4 User Study

A study was conducted involving 17 doctoral students and faculty in the Stanford Database Group to quantify the relative effectiveness of keyword and SQL interfaces in accessing content in relational databases. We used a movie database [15] consisting of three relations, Movies, Casts, Actors, and a view, Movie actor. Using our API described in Section 3.6, we deployed a servlet-based web interface using Apache Tomcat on the same PC used in our system evaluation. See technical report [14] for more details on the user study.

Two classes of questions were formulated. The first class, questions 1, 2, and 3, clearly favours keyword searches. For example, for question 1, the 3-keyword query"Burton Taylor Cinecitta" returns a single movie tuple for the movie Cleopatra, which is the correct answer. On the other hand, questions 4-7 were more difficult with keyword search: The best keyword queries return many result tuples that the user must browse to find the answer. Furthermore, for questions 5 and 6 the user must submit two consecutive keyword queries to find the correct answer. In comparison, all 7 questions could be answered with a single SQL query.

5. CONCLUSION AND FUTURE WORK

We presented a general architecture for supporting keyword-based search over structured databases, and an instantiation of the architecture in our fully implemented system EKSO. EKSO indexes interconnected textual content in relational databases, providing intuitive and highly efficient keyword search capabilities over this content. Our system trades storage space and offline indexing time to significantly reduce query time computation compared to previous approaches. Experimental results show that index size is manageable and keyword query response time is interactive for typical queries. Update experiments demonstrate that our concurrent incremental update mechanism does not significantly hinder query performance. We also present a user study confirming the superiority of keyword-based search over SQL for a range of database retrieval tasks.

We have identified at least three main directions for future work:

• **Indexing Performance**: We plan to investigate better computation sharing when constructing text objects, as well as using parallelism to speed up indexing.

• **Enhanced Search**: By incorporating more structure into text objects, we can exploit (future) search capabilities for semi structured data. We will investigate how to generate text objects with structural information, focusing first on XML and exploiting work on publishing relational content as XML

• **User Interface:** We plan to refine and enhance our current search result presentation and navigation schemes.

REFERENCES

[1]. BENTLEY, J.L. and FRIEDMAN, J.H., *Fast Algorithm for Constructing Minimal Spanning Trees in Coordinate Spaces*, Stanford Report, STAN-CS-75-529 (1975).

[2]. B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *Proc. of IEEE ICDE*, 2004.

[3]. BOOKSTEIN, A. and KRAFT, D., 'Operations research applied to document indexing and retrieval decisions', *Journal of the ACM*, 24, 410-427 (1977).

[4]. GERALD J.KOWALSKI and MARK T.MAYBURY, 'Information storage and retrieval systems', vol 2, pp. 51-52 (2002)

[5]. GOFFMAN, W., 'A searching procedure for information retrieval', *Information Storage and Retrieval*, 2, 294-304 (1977).

[6]. HARPER, D. and van RIJSBERGEN, C.J., 'An evaluation of feedback in document retrieval using co-occurrence data', *Journal of Documentation*, 34, 189-216 (1978).

[7]. Helen J. Peat and Peter Willett ' The Limitations of Term Co-Occurrence Data for Query Expansion in Document Retrieval Systems', JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE. 42(5):378-383, 1991

[8]. KU, H.H. and KULLBACK, S., 'Approximating discrete probability distributions', *IEEE Transactions on Information Theory,* IT-15, 444-447 (1969).

[9]. Qi Su and Jennifer Widom, 'Indexing Relational Database Content Offline for Efficient Keyword-Based Search', Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05).

[10]. VAGELIS.H, LUIS.G, YANNIS.P, Efficient IR-Style Keyword Search over Relational Databases, Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003

[11]. WHITNEY, V.K.M., 'Minimal spanning tree, Algorithm 422', *Communications of the ACM*, 15, 273-274 (1972).

[12]. YU, C.T. and SALTON, G., 'Effective information retrieval using term accuracy', *Communications of the ACM*, 20, 135-142 (1977).

[13] Transaction Processing Council. http://www.tpc.org

[14] Q. Su and J.Widom. Indexing elational database content offline for efficient keyword-based search. Stanford University Technical Report, 2003.

[15] G. Wiederhold. Movie database. http://www-db.stanford.edu/pub/movies/doc.html.