Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 6, June 2015, pg.929 – 936

RESEARCH ARTICLE

Secured Search Optimization using Dynamic Query Form

Prajkta Vijay Dagade[#1], Mansi Bhonsle[*2]

[#]Student, Department Of Computer Engineering, University of Pune, GHRCEM, Ahmednagar, Maharashtra, India

[1] prajkta.dagade@gmail.com

[*]Asst. Professor, Department Of Computer Engineering, University of Pune, GHRCEM, Pune, Maharashtra, India

[2] mansi.bhonsle@gmail.com

Abstract- Modern real word databases such as scientific databases and web databases contains over hundred or even thousands of relations and attributes. Traditional data mining technologies cannot work with large, unstructured and heterogeneous data. Query form is largely used interfaces for querying database. In various information management systems traditional query forms are used. These are designed and pre-defined by developer or DBA. But it is difficult to design set of static query forms to satisfy various ad-hoc database queries on those complex databases and extracting the useful information with this traditional query form from huge dataset is not possible. This paper describes the development and the implementation of a Secured Search Optimization using Dynamic Query Form in Intranet. This system take the query as the input from the user and then it uses the techniques like indexing and searching to get desired result. The failed queries can be resolved in less time due to the continuous updating of the system. The purpose of this application is to reduce the manual efforts of searching to greater extend. This system will provide a user friendly graphical interface.

Keywords- Query Form, Indexing, Searching, Dynamic Query, Information extraction

1. INTRODUCTION

Traditional database systems are powerful and expensive which require the user to construct the database query from language primitives [13].But such a system are not easy to use especially when user is not familiar with the database scheme. The continuous advancement of WAN technology has resulted in the demand for information access over the internet from a diversity of clients and rapid growth in number of data sources available online. Query form is one of the most important user interface used by users for querying databases. With the fast development of web information and modern databases, scientific databases became giant and complex. Many databases such as DBPedia, Freebase have thousands of structured web entities [14] [16]. Hence it is difficult to design set of static query forms to satisfy different ad-hoc database queries on those large databases.

With the growth in computer storage capacity, the amount of information stored on personal computers as text files has increased remarkably at the same time communication between the people over the network in order to perform intended task has also been increased. Many database management and development tools, such as Cold Fusion [18], Easy Query [17], SAP and Microsoft Access, provides several mechanisms to let users create customized queries on databases. This customized query totally depends on user's manual editing [10]. In order to search data it will be critical for user if user is not familiar with the database schema and those thousands of data attributes confuse user.

1.1 Problem Definition

It's challenging for non-expert users to make use of relational database as they are aware to it. Many database interfaces help users to query the relational database without SQL. Query form is one of the most widely used interfaces. Secured Search Optimization using Dynamic Query form provides the query form at the run time according to user's desire.

Searching for a particular file on hard drives has become time consuming. This has led to the development of several search systems which in return has helped users to locate files on a desktop effectively. Search systems have changes the way people access and search information, permitting information about almost any subject to be speedily as well as easily retrieved within seconds. Primary goal is to facilitate information retrieval through dynamic query. Indexing and searching steps to be supplemented with parsing and analysis in order to achieve the best search results. The whole process of information retrieval can be divided into several sequential steps. The complete system consists of 3 major steps they are:

1. Indexing
2. Searching
3. Ranking

This system will also improve the performance by using some techniques like Parallel Indexing, Multithreading and Incremental Indexing.

Table1. Interaction with the system

Query From user	**Recommends a query form components to user, user fills out the query form**
Fetching and indexing the data	**Submit a query** **Fetching and indexing data**
Searching for Query, Execution and display of result	**System executes the query and displays the desired results**

2. RELATED WORKS

Existing database clients and tools make great efforts to help developers design and generate the query forms. They provide visual interface for the developers to create or customize query forms. This system allows end users to customize the existing query form at runtime. It provides a mechanism to let a user fill form to express the desired query. But an end-user may not be familiar with the database. If the database scheme is very large, it is difficult for them to find appropriate database entities and attributes and to create desired query form. In the analysis of the query form which is one of the most useful interfaces for users for querying database,

Various approaches have been proposed for prompt response to user when user wants to search the data. Lots of research works focus on database without structured query language.

Searching information on the internet is nothing but the information retrieval system that helps us to find the information on World Wide Web. A primary search engine indexes most or all site on the web. For example, Yahoo!, Google, MSN will generate the majority of the traffic to a particular web site. Secondary search engines such as Lycos, Miva, Looksmart are targeted at specific audiences or set of people. They generate less traffic but they're useful for regional and more narrowly focused searches. Targeted search engines for example CitySearch, Travel are very narrowly focused to a general topic, like branches of science, travel, sports.

When considering Targeted search engines for search optimization purposes, it should be taken care of that many of these search engines are much more narrowly focused than primary and secondary search engines.

Query by example and Query form are two most widely used database querying interfaces. Query form has been utilized in most real-time business or scientific information systems. Main goal in this work is to focus how to generate Query forms and to discover techniques that assist users who are unfamiliar to and do not want to use Structured Query Language in posing ad-hoc structured queries over relational databases. However, the creation of customized queries totally depends on user's manual editing [12].

Dynamic Faceted search is a type of search engines where relevant facets are presented for the users according to their navigation paths [12], [11]. Dynamic faceted search engines are similar to dynamic query forms if we only consider *Selection* components in a query. However besides *Selections*, a database query firm has other important components such as *Projection* components. Projection components control the output of the query form and needs to be considered. Moreover, design of *Selection* and *Projection* has inherent influences to each other.

Auto Completion for Database Queries, in [5], [11], novel user interfaces have been developed to assist the user to type database queries based on query workload, the database schema. Queries in their work are in the forms of SQL and keywords.

Keyword search for Querying of database generates a lot of query forms in advance. The user inputs several keywords to find relevant query forms from a large number of pre-generated query forms. The system proposes to take as input a target database and then generate and index a set of query forms offline [8]. At query time, a user with a question to be answered issues standard keyword search query; but instead of returning tuples, the system returns forms relevant to the question. The user may then build a structured query with one of these forms and submit it back to the system for evaluation. It works well in the databases which have rich textual information in data tuple and schemas. But it is not appropriate when the user does not have concrete keywords to describe the queries at the beginning. Sometimes the result may not be appropriate after using all of the existing query forms.

Usher is the probabilistic approach that can be used to design intelligent data entry forms that promote high data quality. Usher basically learns the probabilistic model over the questions of the form. Usher then applies this model at every step of the data entry process to improve data quality.

This system focuses on:

1) Data-driven approach 2) Learning a model for data entry, i.e. it uses probabilistic model of the data, represented as a Bayesian network over form questions, 3) Question Ordering, 4) Question re-asking, 5) Evaluation of the benefits of Usher and on the quality of this model [3].

A novel data-driven approach, called query by output (QBO), can enhance the usability of database systems. The central goal of QBO is as follows: Given the output of some query Q on a database D, denoted by Q(D), we wish to construct an alternative query Q_0 such that Q(D) and Q_0(D) are instance equivalent. QBO targeted at improving the usability of database management system [9]. QBO proposes a method to recommend an alternative database query based on results of a query. It does not require the knowledge of the query and can be used in simple database, where the user knows what he wants. But in case of large database predicting an output becomes difficult, hence the method won't work.

3. SYSTEM ARCHITECTURE

We propose a Secured Searched Optimization using Dynamic Query Form system for the query processing, optimization of a relational complex database and it also allow the keyword based search. This system takes the input from the user and gives the desired results by using the techniques like indexing and searching. Lucene is a Search library that serves only as the indexing and searching component. It is important that Lucene does not provide all the functionality of a search application. Figure 1 shows that processes handle by the Lucene and the application.

Step	**Responsibility**
Obtain Content	**Application**
Build document	**Application**
Analyze document	**Lucene**
Index document	**Lucene**
Acquire Query	**Application**
Search index	**Lucene**
Display search result	**Application**

The set of steps in search application is important. For better comprehension of the process, sequence of the steps may be clustered into two major categories. The first four steps, from acquiring the content to indexing the document, are part of indexing, while the steps from obtaining the query to presenting the search results are part of searching. The Figure 2 shows the work-flow of the system architecture.

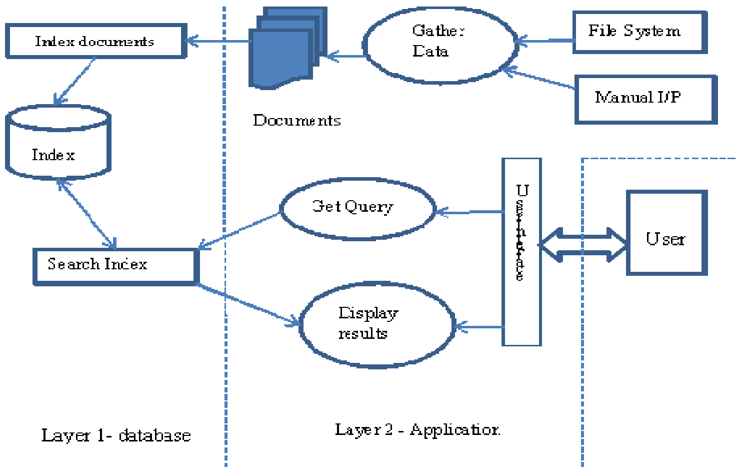
Indexing:

First step of application is indexing. Its objective is to process original data to highly efficient cross-reference lookup in order to facilitate rapid searching.

The first step of indexing is acquiring the data which in turns gather all the data that needs to be indexed. The job is simple when data is already in textual format and its location is known.

The next step is to build the documents out of the content. The raw data that needs to be translated into the units called as documents. The document contains separately named fields with values, such as title, body, abstract, author etc. Application builds the document from raw data. Then the textual field in a document has to be broken into a series of individual atomic elements called as tokens. This process is known as document analysis. Each token corresponds roughly to a word in the language, and the analyzer determines how the textual fields in the document are divided into a series of tokens.

The final step is to index the document. During the indexing step document is added to the index. Lucene provide everything for this step.



Searching:

The process of looking up words in an index to find documents where they appear is called as searching. To search, search query is obtained by the user through user interface. Once user interacts with the interface he or she has to be allowed to submit the search request.

After obtaining and processing the search request, final step is to present search result after processing the query.

4. Mathematical Module

The GenerateQuery algorithm executes a query form. A query form denoted by F is a tuple (AF, RF, σF), which represents a database query as follows: F= (Select A1, A2,…..,$A_K$from RF where σF), Where, AF = {A1, A2,…..A_K} are k attributes for projection. AF is the set of the columns of the result table.σFis the set of input components for users to fill or is a conjunction of expressions for selections on relations in RF, Where RF is the relation. The database query is generated from the query form by using the GenerateQuery algorithm, based on the given set of projection attributes A_{in} with selection expression σin. The query construction algorithm is used to get the attributes and condition given by user each time and is given to GenerateQuery algorithm for query generation and execution. And the result is displayed to the user.

Every time a document matches a search, system computes a score (a numeric value of relevance) and assigns the score to the document. The score reflects how good the match is. The score is computed for each document (d) matching each term (t) in a query (q).

Score(q;d) = coord(q,d)_queryNorm(q)_Σ(tf(t ϵ d)_idf(t)_boost(t:fieldϵ d)_lenghtNorm(t:fieldϵ d))

We can see that term frequency and inverse document frequency somehow normalized. Functions used in the formula are described as:

Function	Explanation
tf(t ϵ d)	Normalised number of times the term appears in a document(term frequency)
idf(t)	Total number documents/number of documents containing the term(inverse document frequency)
coord(q,d)	How many of the query terms are found in the document (Coordination factor)
queryNorm(q)	Normalising factor assuring comparable scores between queries (Normalization value for query)
boost(t:field ϵ d)	Index time boost specified during indexing
lenghtNorm(t:field ϵ d)	Normalising factor assigning shorter fields bigger boost(normalization value of a field)

5. ANALYSIS AND RESULTS

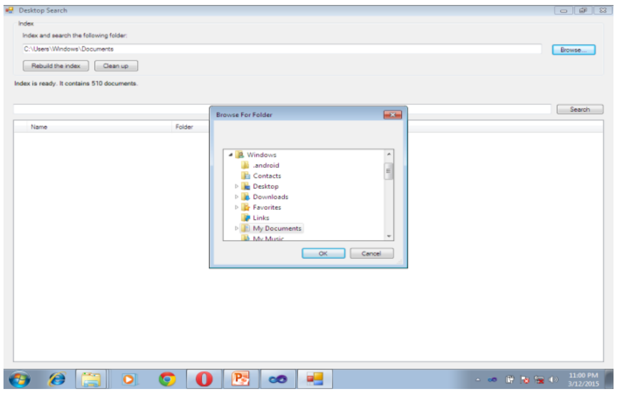
A. Data sets

We examine the benefits of Secured Search optimization using Dynamic Query Form by taking all the files available in the system such as simple text files, Microsoft Word documents, XML documents, HTML documents, PDF files and many more.

B. Performance Measurement and Results

A perfect retrieval system would retrieve only the relevant documents and no irrelevant documents. However perfect retrieval system does not exist, and will never exist, because search statements are necessarily incomplete, and relevance depends on the subjective opinion of the user. A typical search method visits every single document that is the candidate for matching the search, and only accepts documents that meet every condition of the query. Finally it gathers the top results and returns them to the user.

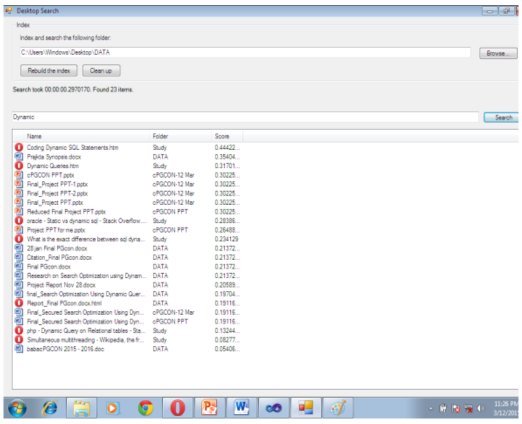
Any information retrieval system is only as good as its search capabilities. It is not only about searching, though. Figure shows how system works:



Index and search the following folder: C:\Users\Windows\Documents

Index is ready. It contains 510 documents.

We select particular drive where we want to perform content based search in order to locate file and get the results.



Index and search the following folder: C:\Users\Windows\Desktop\DATA

Search took 00:00:00.2970170. Found 23 items.

Dynamic

Name	Folder	Score
Coding Dynamic SQL Statements.htm	Study	0.44422...
Prajkta Synopsis.docx	DATA	0.35404...
Dynamic Queries.htm	Study	0.31701...
cPGCON PPT.pptx	cPGCON-12 Mar	0.30225...
Final_Project PPT-1.pptx	cPGCON-12 Mar	0.30225...
Final_Project PPT-2.pptx	cPGCON-12 Mar	0.30225...
Final_Project PPT.pptx	cPGCON-12 Mar	0.30225...
Reduced Final Project PPT.pptx	cPGCON PPT	0.30225...
oracle - Static vs dynamic sql - Stack Overflow....	Study	0.28386...
Project PPT for me.pptx	cPGCON PPT	0.26488...
What is the exact difference between sql dyna...	Study	0.234129
28 jan Final PGcon.docx	DATA	0.21372...
Citation_Final PGcon.docx	DATA	0.21372...
Final PGcon.docx	DATA	0.21372...
Research on Search Optimization using Dynam...	DATA	0.21372...
Project Report Nov 28.docx	DATA	0.20589...
final_Search Optimization Using Dynamic Quer...	DATA	0.19704...
Report_Final PGcon.docx.html	DATA	0.19116...
Final_Secured Search Optimization Using Dyn...	cPGCON-12 Mar	0.19116...
Final_Secured Search Optimization Using Dyn...	cPGCON PPT	0.19116...
php - Dynamic Query on Relational tables - Sta...	Study	0.13244...
Simultaneous multithreading - Wikipedia, the fr...	Study	0.08277...
babacPGCON 2015 - 2016.doc	DATA	0.05406...

Searching is the process of looking up words in an index to find documents where they appear. The quality of the search is typically described using precision and recall metrics. Recall measures how well the search system finds relevant documents: precision measures how well the system filters out the irrelevant documents.
In our simulation experiments, the quality of search is typically described using precision and recall metrics.

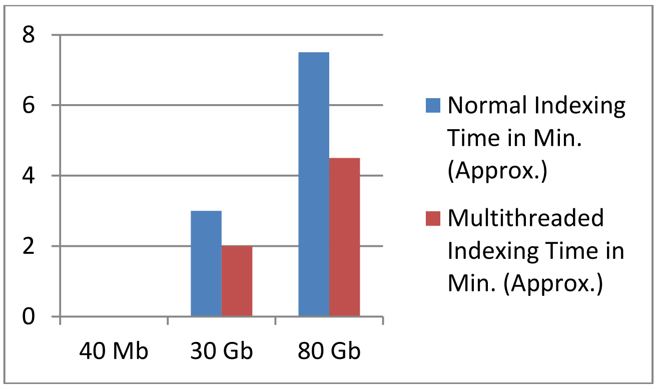
Precision:
Measures how well the system filters out the relevant documents.

$$\text{precision}\,\frac{\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}}{\{\text{retrieved document}\}}$$

Recall:
It measures how well the search system finds relevant documents.

$$\text{recall}\,\frac{\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}}{\{\text{relevant document}\}}$$

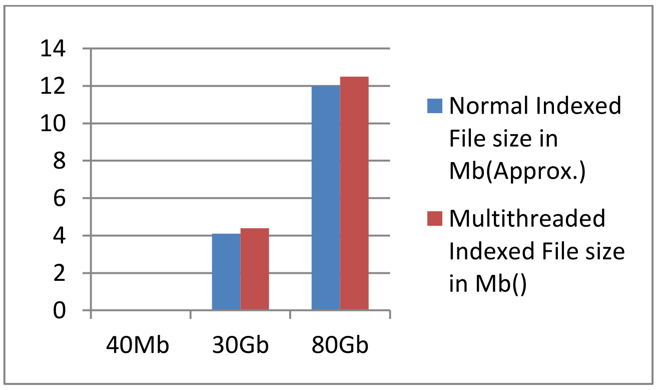
Time in minutes



Size of data to be indexed

We can improve indexing performance by creating multiple index files by using the concept of multithreading. Figure 4 above illustrates the performance with normal indexing against multithreaded indexing.

Indexed File size in Mb



Size of data to be indexed

When indexing is done index file are created which are approximately 10% to the actual size of data. Along with this when indexing is done multiple times; earlier index files will be overwritten. Figure 5 illustrates size of index files for respective data sizes which are to be indexed.

Data Size	Normal Indexing Time in Min. (Approx.)	Index File size in Mb (Approx.)	Multithreaded Indexing Time in Min. (Approx.)	Index File size in Mb (Approx.)
40 Mb	0.33	4	0.033	5
30 Gb	30	450	20	482
80 Gb	75	1200	45	1280

Table 2. Comparison of indexing performance.

6. CONCLUSIONS

Thus, we can develop Secured search optimization using dynamic form which will help in faster information retrieval for user. With this application user can search text, HTML, document files present on user system. A form based interface is the gateway to many database systems and it determines the usefulness and availability of the data. We focused on an innovative approach called Secured Search Optimization using dynamic Query Form in Intranet. Dynamic approach used in this system offer a dramatic change from existing static and other approaches and leads to the higher success rate and simpler query.
As a future work we can incorporate natural language processing in this dynamic query form system. We can also develop this system to work in Web.

REFERENCES

[1] Sathu G Rajan, K. SathyaSeelan. Dynamic Query recommendation for interactive database Exploration, 2014.

[2] Dr. Anil Rajput. Multicriteria data Retrieval in database using Advanced Database Operator. March 2014

[3] Kaung Chen, Harr Chen, Neil Conway. USHAR: Improving Data Quality with Dynamic Forms, June 2011.

[4] C. Li, N. Yan, S.B.Roy, L. Lisham, and G. Das. Facetedpedia: Dynamic generation of query-dependent faceted interfaces for Wikipedia. In proceeding of WWW, pages 651-660, Raleigh, North Carolina, USA, April 2010.

[5] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In Proceedings of WWW, pages 781-790, Raleigh, North Carolina, USA, April 2010.

[6] N. Khoussainova, Y. kwon, M, Balazinska, and D. Suciu.Snipsuggest: Context-aware autocompletion for sql. PVLDB, 4(1):22-23, 2010.

[7] Lucie Molkova, “Indexing Very Large Text Data”, Master’s Thesis, in Brno, sring 2011.

[8] E. Chu, A. Baid, X. Chai, A. Doan, and J.F.Naughton.“Combining keyword search and forms for ad hoc querying of Databases”, June 2009.

[9] Q. T. Tran, C.Y. Chan, and S. Parthasarathy. “Query by Output”. In Proceeding of SIGMOD, pages 535-548, providence, Rhode, USA, Sep. 2009.

[10] M. Jayapandian, H. V. Jagadish. Automated Creation of aform-based database Query Interface.VLDB 2008.

[11] A. Nandi and H. V. Jagadish. Assisted querying using instant response interfaces. In proceeding of ACM SIGMOD, pages 1156-1157, 2007.

[12] Eugene Inseok Chong. “An Efficient SQL-based RDF Querying scheme”, 2005.

[13] Jeffry D. Ullman. Principles of Database System.Computer Science Press, 1980.

[14] DBPedia. http://DBpedia.org

[15] http://wiki.apache.org/lucenejava/ImproveIndexingSeach

[16] Freebase. http://www.freebase.com

[17] EasyQuery.http://www.easyquery.com

[18] ColdFusion. http://www.adobe.com/products/coldfusion.