# Survey paper for keyword Search over RDBMS

Vaibhav Ambavkar, Venkatesan N.

SKN SIT's Lonavla,Vaibhav.ambavkar@vpmmpcoe.org and 7066384889

**Abstract**— Keyword search (KWS) over relational databases has recently received significant attention. Many solutions and many prototypes have been developed. This task requires addressing many issues, including robustness, accuracy, reliability, and privacy. An emerging issue, however, appears to be performance related current KWS systems have unpredictable running times. In particular, for certain queries it takes too long to produce answers, and for others the system may even fail to return. we say that as today's users have been spoiled by the performance of Internet search engines, KWS systems should return whatever answers they can produce quickly and then provide users with options for exploring any portion of the answer space not covered by these answers. basic idea is to produce answers that can be generated quickly as in today's KWS systems, then to show users query forms that characterize the unexplored portion of the answer space. Combining KWS systems with forms allows us to bypass the performance problems inherent to KWS without compromising query coverage. This project uses the concepts of Priority Queue Data Structure to build a framework for developing, Testing and applying search algorithms on large volume of data streams.

**Keywords**— Keyword Search, Keyword Priority, Keyword Indexing, Doument Search, Document Indexing, Keyword Asociation, Object Summerization, Keyword Recommendation.

## INTRODUCTION

The success of search engines demonstrates that untrained users are comfortable using keyword search to find documents of interest to them. Over the past decade, this success has made tremendous interest in keyword search (KWS) over relational databases, in order to accommodate users who cannot issue a formal structured query or are unaware of the database schema.DBXplorer [12], DISCOVER, and BANKS [2] were among the first systems that supported keyword search over relational databases, and many other systems have since been developed. As more structured data becomes available at organizations and on the Web, and as more untrained users want to use such data. Building such systems requires addressing many issues, including robustness, accuracy, reliability, and privacy.

The current KWS solutions have unpredictable performance issues. Specifically, while the systems produce answers quickly for many queries, for many others they take an unacceptably long time, or even fail to produce any answer after exhausting memory. Clearly, such a performance profile is unacceptable for a real-world system. The system should then allow for a way for the user to explore this portion of the answer space should he or she choose to do so. one way to do so is to provide form interfaces to characterize the yet-unexplored answer space. the above two requirements of time limit and overview of the yet-unseen can help increase the chances that the KWS system will be perceived as useful and will be widely adopted by real-world customers.

A keyword search is a basic search technique that involves searching for one or more words within a collection of documents. Typically, a keyword search involves a user typing their search request, or query, into a search engine such as Google, which then returns only those documents that contain the search terms entered. The documents returned by the search engine are called the search results.

### Keyword Search Techniques & Keyword Search Tools

#### Keyword search normal parameters

The syntax in the search string. Use of the keywords with or without stemming. Use of keywords with certain wildcard specifications and the syntax for said wildcards. Case-sensitivity of keywords used in searches and whether the keyword should match both cases; and The target data sources to be searched. Whether the query can be applied to any specific fields such as email To/From or Subject. Whether the query can be applied to any specific date range such as an email Sent Date between the date ranges of January 1, 2001 through December 31, 2001.

#### Assumed Parameters

The character encoding of the text UTF-8, UTF-16, CP1252, Unicode/Wide Char etc. Language of the keyword, to select appropriate stemming. Any special handling of characters such as diacritics, accents etc. If de-compounding of the keyword needs to be performed, usually when working with languages such as German. If there is a set of special characters, what the special characters are, and how an escape character is specified. If there is a tokenization scheme present, what the token delimiters are, and the impact of tokenization on search ability of documents. Searches may not be precise when these tokenization characters are present in the keyword.

**Phrase Search**

The following should be considered when conducting a phrase search: Phrases that contain a double-quote in any of its keywords require escaping of the double quote. Example: to search for the quoted expression: He said: don't do it.

**Wildcard Specifications**

Single character wildcard x?y matches all strings that begin with substring x, end with substring y, and have exactly one-character in between x and y Multiple-character wildcard x*y matches all strings that begin with substring x, end with substring y, and have 0 or more characters between x and y.

**Truncation Specification**

Truncation specification is one way to match word variations. Truncation allows for the final few characters to be left unspecified. Truncation is specified using the following syntax's! matches all strings that begin with substring x. !x matches all strings that end with substring x. x! y when specified within a phrase, the truncated match on the words with !, and exact match on the others.

**Stemming Specifications**

Stemming specification is another method for matching word variations. Stemming is the process of finding the root form of a word. The stemming specification will match all morphological inflections of the word, so that if you enter the search term sing, the stemming matches would include singing, sang, and song. Note that even though a stemming search will return singing for a search term of sing, this is different from wildcard search. A wildcard search for sing* will not return sang or song, while it will return Singsong. x matches all morphological variations (inflections) of the word. Exactly how a search implementation identifies these inflections is not specified. x y when specified within a phrase, the stemming variations match on the words with , and exact match on the others.

**Fuzzy Search**

Fuzzy search allows searching for word variations such as in the case of misspellings. Typically, such searching includes some form of distance and score computations between the specified word and the words in the corpus. Fuzzy-search(x,s) For the search word x, find fuzzy variations that are within the score s. The score is specified as a value from 0.0 to 1.0, with values closer to 1.0 being a closer match. The word itself, if present, will match with a score of 1.0. Fuzzy-search(x,s,n) For the search word x, find fuzzy variations that are within the score s and limit the results to the top n by score.

**Boolean Search**

AND This is specified between two keywords and/or phrases, and specifies that both of the items be present for the expression to match. OR This is specified between two keywords and/or phrases, and specifies that either of the two items be present for the expression to match. NOT Negates the truth value of the expression specified after the NOT operator. NOT w/n specifies that the terms and/or phrases to the right of the w/n specification must not be present within the specified number of words. ANDANY This
is specified between two keywords and/or phrases, and specifies that items following the ANDANY operator are optional. w/n Connects keywords and/or phrases by using a nearness or proximity specification. The specification states that the two words and/or phrases are within n words of each other, and the two words/phrases can be in either order. the specified number of words implies that there are n-1 intervening other words between the two. Noise words are counted in the specification. Pre/n Connects keywords and/or phrases by using a nearness or proximity specification.

The specification states that the two words and/or phrases are within n words of each other, and the order of the words is important. W/Para The two keywords and/or phrases are found within the same paragraph, and order is not important. pre/Para The two keywords and/or phrases are found within the same paragraph, and order is important. W/sent The two keywords and/or phrases are found within the same sentence, and order is not important. Pre/sent The two keywords and/or phrases are found within the same sentence, and order is important. Start/n the keyword/phrase is present at the start of the document or section, within n words of the start. End/n the keyword/phrase is present at the end of the document or section, within n words of the end.

**Synonym Search**

Synonyms are word variations that are determined to be synonyms of the word being searched. Such searching includes some form of dictionary or thesaurus based lookup. Synonym search is specified using the operator: synonym-search. Synonym-search(x) For the search word x, find synonym variations. Synonym search may be combined with other search constructs. Synonym search may also be included as part of a phrase or Boolean construct.

**Related Words Search**

Related words search allows a legal professional to specify a word and other words that are deemed to be related to it. Typically, such related words are determined as either part of concept search or by statistical co-occurrence with other words. Related word search is specified using the operator related-word-search. Related-word-search(x) For the search word x, find other related words. Related words search may be combined with other search constructs, and be included as part of a phrase, or Boolean constructs.

**Concept Search**

Concept search allows a legal professional to specify a concept and documents that describe that concept to be returned as the search results. It can be a useful technique to identify potentially relevant documents when a set of keywords are not known in advance. Concept search solutions rely on sophisticated algorithms to evaluate whether certain set of documents match a concept. There are three broad categories of concept search that a legal practitioner may need to understand and evaluate its applicability.

**Latent Semantic Indexing**

Latent semantic indexing (sometimes also referred to as Latent Semantic Analysis is a technology that analyzes co-occurrence of keyword terms in the document collection. In textual documents, keywords exhibit polysemy (which refers to a single keyword having multiple meanings) as well as synonymy (which refers to multiple words having the same meaning). An additional factor is certain keywords are related to the concept in that they appear together. These relationships can be is-a relationship such as motorcycle is a vehicle or a containment relationship such as wheels of a motorcycle.”is-a part “of relationship also can be achieved with the use of latent semantic indexing. it shows the generalized relationship among two or more keywords to map the exact correlation factor among the keywords. it will create index on such a keyword, based on the concept of small keyword set correlation with ranking.

**Text Clustering**

Text clustering is a technology that analyzes a document collection and organizes the documents into clusters. This clustering is usually based on finding documents that are similar to each other based on words contained within it (such as noun phrases). Text clustering establishes a notion of distance between documents and attempts to select enough documents into the cluster so as to minimize the overall pair-wise distance among all pairs of documents. In the process, new clusters are created from documents that may not belong to a cluster.

**Bayesian Classifier**

Bayesian classifier is a process of identifying concepts using a certain representative documents in a particular category. As an example, one may select a small sample of responsive documents and feed them to a Bayesian classifier. The classifier then has the ability to discern other responsive documents in the larger collection and place them in a category. Typically, a category is represented by a collection of words and their frequency of occurrence within the document. The probability that a document belongs to a category is based on the product of the each word of the document appearing in that category across all documents. Thus, the learning classifier is able to apply words present in a sample category and apply that knowledge to other new documents. In the e-discovery context, such classifier can quickly place documents into confidential, privileged, responsive documents and other well-known categories.

**Concept Search Specification**

Effectiveness of concept search in an e-discovery project depends greatly on the type of algorithm used and its implementation. Given multiple different technologies, the EDRM Search specification proposes that a concept search was used for fulfilling a search request and a registered concept-search implementation/algorithm was used, and an identifier (name) of the concept that was used in the search. Concept search is specified using the operator concept-search. Concept-search (concept-implementation, x, vendor-param-1, vendor-param-2,) given a concept x and concept-implementation, locate all documents that belong or describe that concept. Some vendor implementations may require additional parameters. To indicate the type of concept-implementation, concept search vendors are encouraged to register their implementation name. It is not required to disclose the internal algorithms the vendor utilizes to implement the search. Concept search may be combined with other search constructs, and also be included as part of a phrase or Boolean clause.

**Occurrence Count**

Occurrence count search allows a legal professional to specify theta word appear a certain number of times for the document to be selected. Occurrence count search is specified using the operator occurs. Occurs(x,n) For the search word x, count the number of times it appears, and select the document if the specified occurrence count is matched.

**Errors to Avoid**

Stemming may cause additional unintended keyword matches. Wildcard expansions may cause results to be overly broad. If tokenization is based on certain text characters being interpreted as delimiters, they may not be searchable as a keyword. Consider using a phrase as a search. Case-sensitivity may need to be considered carefully. If a word in a document contains a hyphen and the keyword matches any or all of the hyphenated word, depending on how the document is indexed the hyphen may prevent a match. For example, if the keyword is known and the document contains well-known, there is a chance that the search engine will not recognize the two as a match. If the document is structured as a compound document (i.e., has multiple sections such as Title, Body etc.), Keyword-based searches should be performed with care.

**Keyword search Tool**
These keyword research tools should make it easier to create a list of relevant search terms. we should make sure to create awesome landing pages for keywords you want to be found on. we should also think about cornerstone content articles and a great internal Linking structure in order to make your SEO strategy completes.

**Google Ad words Keyword Planner**
The Google Ad words Keyword Planner to find new and related keywords, but ignore the search volume data. The search volume data in the planner is really only useful for keywords that actually spending money to advertise on. Otherwise, these volumes are not reliable. While not really helpful to decide which keyword is most used by potential audience, Google Ad words Keyword Planner makes a useful tool in coming up with ideas for potential keywords.

**Yoast suggests**
Joost developed his own keyword research tool to come up with keywords as well! Yoast Suggests uses the Google Suggest functionality you know from searching in Google. It finds the keyword expansions Google gives and then requests more of them. So if we type any example keyword, it will also give you the expansions for example a till example z etc.

**Google Trends**
Google Trends allows you to compare the traffic for sets of keywords.

## Literature Review

**Keyword Search Engine Architecture**
A conceptual look at how a general-purpose search engine like Google or Yahoo! might set up their architecture so that these and other new features could be easily added while maintaining overall architecture coherency. This is a purely intellectual exercise - no doubt each of the major search engines will evolve their own strategy and architecture to deal with these issues.

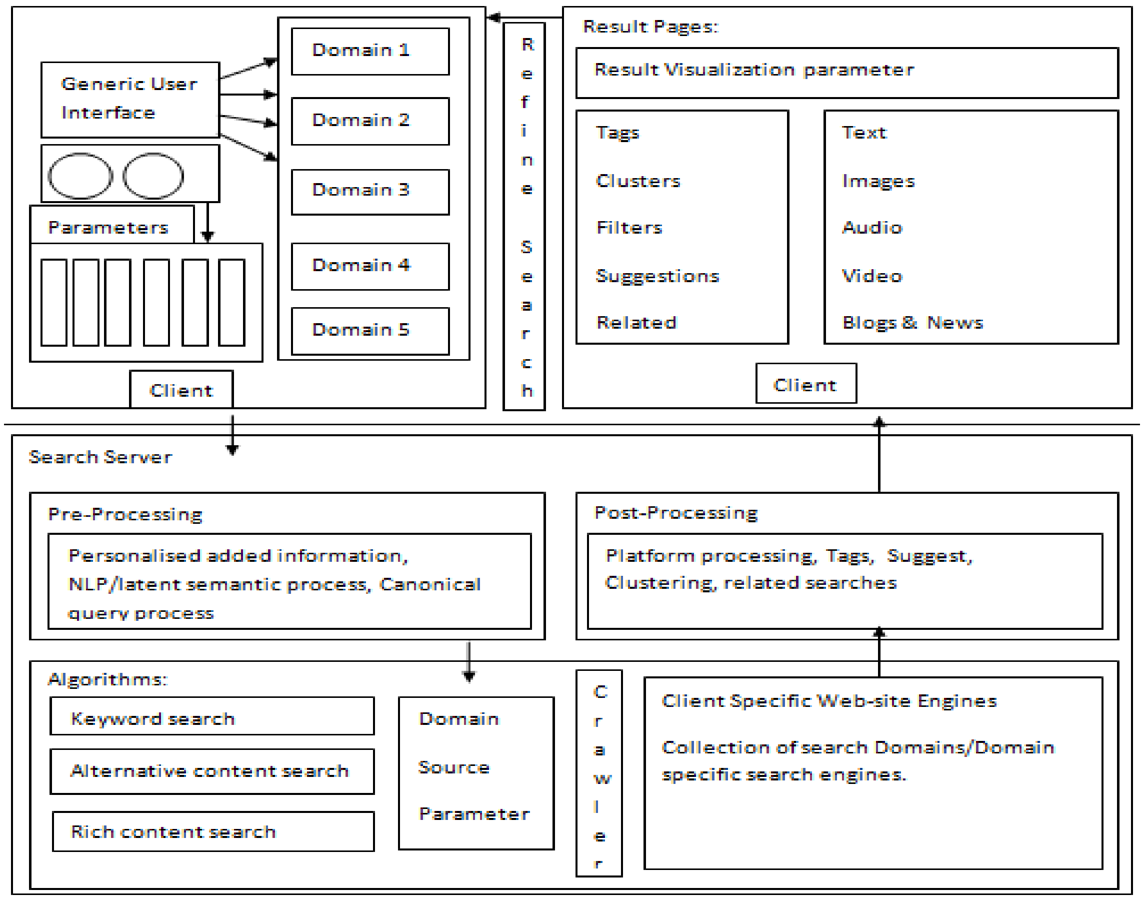
**Query Interface**
One key change to the query interface in the future is the likely addition of search parameters, which can use the magic of Ajax to appear automatically as needed. Parameters can be classified into two types: General parameters, such as freshness dates and content type, and Domain-specific parameters for vertical search queries.

**Server components**
The simple "search box" on the Google front page could hide a variety of specialized search engines behind it Pre-processing support: Personalization, Natural language processing, semantic analysis - Algorithmic changes: Rich content search, social input (reputation-based), self-optimization - Source restrictions: Restricting the scope of the search to trusted sources and/or to a specific vertical - Post-processing support: Clustering, related tags, support for services

**Results interface**
The results interface should include support for enhanced types of results visualization, such as clustering and related tags, query refinement (using filters or suggestions), along with support for saving searches (user agents) and alternative results platforms - such as Mobile, RSS feeds, RIAs, Emails and Web Services.

**Keyword Search on Relational Databases Approach**
The effectiveness of the scoring and ranking functions is an important aspect of keyword search [4]. As more than one result may match any keyword query, it is desirable to assign each result a score and rank the list of results according to their scores. Combining KWS and KWS-F [3], forms can potentially offer a good transition to go from an unstructured keyword query to the results of a structured query. Bernstein et al. make a similar observation when they state that when querying structured data, a partially structured query interface is preferable to a completely unstructured interface because of its guidance effect.

**Keyword Aggregate Query Based on Query Template Approach**
Keyword aggregate query has been divided into three stages for processing. The first stage is keywords preprocessing, the keywords is located in the database during this stage, generating the query items. The second stage is generating the templates, by which we can get the query templates [7]. The third stage is rating the results for the query template.


Generic User Interface
Parameters
Client
Domain 1
Domain 2
Domain 3
Domain 4
Domain 5
Refine Search
Result Pages:
Result Visualization parameter
Tags
Clusters
Filters
Suggestions
Related
Text
Images
Audio
Video
Blogs & News
Client
Search Server
Pre-Processing
Personalised added information, NLP/latent semantic process, Canonical query process
Post-Processing
Platform processing, Tags, Suggest, Clustering, related searches
Algorithms:
Keyword search
Alternative content search
Rich content search
Domain
Source
Parameter
Crawler
Client Specific Web-site Engines
Collection of search Domains/Domain specific search engines.

**Proficient Search in Relational Database Based on Keyword Approach**
Keyword based search and ranked the retrieved results based on the degree of relevance to the user. The ranking was done using the score [6]. The performance of the system using different metrics was also evaluated, also performed the searching and ranking of the certain and structured data of the relational databases. In future they would like to extend our searching and ranking to uncertain and probabilistic databases.

**Keywords Retrieval in Relational Databases Based on Index Structure Approach**
A query mapping index method different of the full-text index method, which creates a mapping table [11], and match index table and the corresponding key words. So it can improve the query speed. In the query mapping index method, setting as the threshold, it can Use PSO algorithm to learn keywords number. According to the number of occurrences to each of the query keywords, it can determine the new keywords. So it can retrieve the log to mining, and achieve the goal of retrieving through the query the user's usage and form.

**BANKS System Approach**
The BANKS system supports keyword search on databases storing structured/semi-structured data. Answers to keyword queries are ranked, and as in IR systems, the top answers may not be exactly what a user is looking for. Further interaction with the system is required to narrow in on desired answers. it describe some of the new features that are added to the BANKS[9] system to improve user interaction. These include an extended query model, richer support for user feedback and better display of answers.

**Other related Works for KWS Approach**
Wei Wang ,Xuemin Lin, Yi Luo worked out keyword search on the relational databases using data models with graph based relational data models approach obtained results with top-k keyword query result and indexing algorithm top-k index optimization.

Bin Zhu, Fang Yuan, Yu Wang used keyword processing with aggregate query and produced one query template generation algorithm(create QT algorithm),result generation algorithm(Translate QT to result) which will estimate keyword based on average precision, recall factor for keyword query.

B. Aditya ,Soumen Chakrabarti have estimate BANKS system for keyword search in relational databases,they have worked out XML relational query which will try to find k-nearest search with tree weight model processing node selection based on proximity and ranking keyword functions.

R.Suresh, K.Saranya, S.Dhivya, K.Thilagapriya were carried out proficient search in relational databases based on Query tree approach (Q-tree) they have processed query by expanding the keyword search using ranked aggregation method.

Haizhou Fu, Sidan Gao, Kemafor Anyanwu have specified degree of keyword disambiguity with Top-k generation algorithm,optimal deep segmentation algorithm for keyword search on RDF schema , Data graph and derived results in terms of a hit keyword k from graph,hit keyword array to find disambiguous keyword.

## Algorithm

**Algorithm keyword Search algorithm**

1. Accept input Search term, search Query
2. Apply query processing technique on query
3. Analyze search term from query
4. Model Relational DB with all databases, Tables, Domains and tuples
5. Find presence and absence of search term
6. Correlate search term keyword with Relational DB objects
7. Create matrix to represent the relationship between database tulles
8. Compute proximity factor proximity distances for keyword and tuples
9. Prioritize keyword based on applying indexing ranking functions on keywords with tuples
10. Create priority queue containing set of keywords
11. Create final keyword relationship matrix based on priority queue containing set of keywords and tuples which will map keyword with highest priority first
12. Return prioritized set of Keyword to refine the input search term or search query

The goal is to choose k to minimize the Proximity Distance as presented in the following function.

**Mathematical Modeling Objective function**

$$\mathbf{R[I, J] = r[i, j] = \Sigma\, yd * wd\ (ki, kj)}$$

## Potential significance and Applications

**Document Search in relational databases**

Internet search engines have popularized the keyword based search paradigm. While traditional database management systems offer powerful query languages, they do not allow keyword-based search. Keyword based Xplorer system that enables keyword based search in relational databases. DBXplorer has been implemented using a commercial relational database and web server and allows users to interact via a browser front-end.

**Information retrieval in Cloud based environment**

As cloud computing becomes most general, the important information is centralized into the cloud server. To protect the data stored in the cloud, the data must be encrypted. Although traditional encryption techniques allows the user to securely search through the keyword and return retrieved files, these techniques are useful only for exact keyword search. Providing fast searching and increase the performance by considering Keyword substring from the given search query string

## CONCLUSION

The Keyword search algorithm will be the hybrid approach to avoid Performance problems in traditional KWS systems and try to find solution over how to generate the top-k list of interpretations or structurizations that represent the most likely meanings intended by the user.

The effectiveness of the scoring and ranking functions is an important aspect of keyword search. As more than one result may match any keyword query, it is desirable to assign each result a score and rank the list of results according to their scores. The top results in the ranked list are more relevant to the query than those at the bottom.

## REFERENCES:

[1] Georgios J. Fakas, Zhi Cai, and Nikos Mamoulis ”Versatile Size-l Object Summaries for Relational Keyword Search” , IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 26, NO. 4, APRIL 2014.

[2]Soumen Chakrabarti, B. Aditya ”User Interaction in the BANKS System: A Demonstration ” , IEEE Transactions on Data Engineering, Feb. 2002, (Volume:1 ).

[3]Akanksha Baid, Ian Rae, Jiexing Li ”Toward Scalable Keyword Search over Relational Data ”, IEEE Transactions on Networking, Architecture and Storage (NAS), 2009 IEEE 5th International Conference.

[4] Wei Wang, Xuemin Lin ”Keyword Search on Relational Databases”, IEEE TRANSACTIONS Conference on Very Large Data Bases (VLDB 2010).

[5]Yan ZHAN, Hao CHEN ”Keywords Retrieval in Relational Databases Based on Index Structure ”,International Journal of Intelligent Information and Management Science ISSN: 2307-0692 Volume 3, Issue 5, October 2014.

[6] R.Suresh, K.Saranya, S.Dhivya, K.Thilagapriya”Proficient Search in Relational Database Based on Keyword”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 8958, Volume-1, Issue-4, April 2012.

[7] Bin Zhu, Fang Yuan”Keyword Aggregate Query Based on Query Template”, IEEE TRANSACTIONS ON 2012 International Conference on Computer and Information Application (ICCIA 2012).

[8] Haizhou Fu, Sidan Gao, Kemafor Anyanwu “Disambiguating Keyword Queries on RDF Databases Using Deep Segmentation”, In:Proc. of the 18th International Conference on Data Engineering (ICDE 2012). San Jose: IEEE Computer Society Press, 2012, pp.431.

[9] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, S. Sudarshan ”BANKS: browsing and keyword searching in relational databases”, In VLDB 2002.

[10] Y. Luo, X. Lin, W. Wang, X. Zhou”Spark: top-k keyword query in relational databases”, In SIGMOD 07.

[11] M. Sayyadian, H. Lekhac, A. Doan, L. Gravano ”Efficient keyword search across heterogeneous relational databases”, In ICDE 07.

[12] Agrawal S, Chaudhuri S, Das G ”DBXplorer: A system for keyword-based search over relational databases”, In: Proc. of the 18th International conference on Data Engineering (ICDE 2002). San Jose: IEEE Computer Society Press, 2002, pp.5-16.

[13] Jianhua Feng, Guoliang Li, Jianyong Wang ”Finding Top-kAnswers in Keyword Search over Ralational Databases UsingTuple Units”, IEEE Transactions Knowledge and Data Engineering (TKDE), 2011,23(12), pp.1781-1794.

[14] R. Goldman, N. Shivakumar, S. Venkatasubramanian,H. Garcia-Molina”Proximity search in databases”, In Proc. of the International conference on VLDB, pages 2637, 1998.

[15] K. Golenberg, B. Kimelfeld, and Y. Sagiv Keyword Proximity Search in Complex Data Graphs ,Proc. 28th ACM SIGMOD International conference Management of Data, 2008.