ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

A Survey on Keyword Query Routing

Prachi M. Karale[1]
Computer Department
Vishwabharti College of eng, A.nagar
A.nagar, India

Natikar S.B[2]
Assistant prof, Computer Department
Vishwabharti College of eng,
A.nagar, India

Abstract: ***It is difficult for typical web users to exploit web data by means of structured queries using languages like SQL. To end this keyword search has proven intuitive. Keyword search is a familiar and potentially effective way to find information of interest that is "locked" inside databases. Keyword search is based on results obtained by searching linked data sources on the web. There are two direction of work keyword search and database selection. In this paper we have explained keyword search approaches i.e. schema-based and schema-agnostic. Lastly we explained how to find out most relevant databases. Routing plans gives the best result for keyword search. We have given survey of all the keyword searching technique and lastly given best solution and solution is routing plans.***

Keywords: M-KS, G-KS, KRG, KERG, KEM, Keyword Query Routing

I. INTRODUCTION

There are two types of databases text databases and relational databases. In text databases, the basic information units searched by users are documents. For a given keyword query, IR systems compute a numeric score for each document and rank the documents by this score. The top ranked documents are returned as answers. In relational databases, however, information is stored in the form of columns, tables and primary key to foreign key relationships. The logical unit of answers needed by users is not limited to an individual column value or even an individual tuple; it may be multiple tuples joined together.

The web is collection of textual documents and also interlinked data sources. Linked data is about using the web to connect related data that wasn't previously linked, or using the web to lower the barriers to linking data currently linked using other methods. Linked data compromise hundreds of sources containing millions of links. Same-as-link denotes two RDF resources represents same real world. Linked data is given in fig. 1.

It is difficult for any non technical web users to get the data. Technical users have knowledge of SQL language, so that he can easily exploit the web data. Every user can get data using keyword search. Keyword search do not required any knowledge of structured queries i.e. query languages, the schema, or the underlying data.

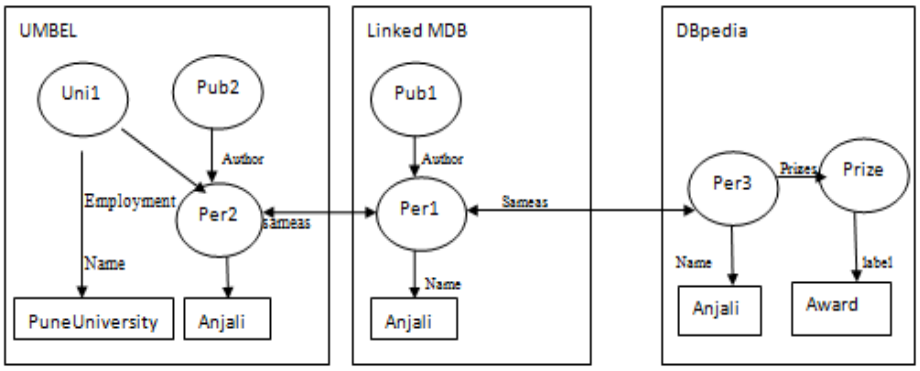

Fig .1 Extract of the web data graph

In database search, solutions have been proposed which given keyword query, retrieve the most relevant structured results[9], [10], [11] ,[12] or simply, select most relevant databases [1], [2]. But, these approaches are applicable for single source solutions. They are not directly applicable to the web of linked data. In linked data results are not bounded to single source but might include several linked data sources. As opposed to the source selection problem [6], [7], which is focusing on computing the most relevant sources, the problem here is to compute the most relevant combinations of sources. The goal is to produce routing plans, which can be used to compute results from multiple sources.

To this end, keyword query routing [5] plays main role. It investigates the problem of keyword query routing for keyword search over a large number of structured and Linked Data sources. Routing keywords only to relevant sources can reduce the high cost of searching for structured results that span multiple sources. Previous work uses keyword relationships (KR) collected individually for single databases [1], [2]. While, existing system represents relationships between keywords as well as those between data elements. They are constructed for the entire collection of linked sources, and then grouped as elements of a compact summary called the set-level keyword-element relationship graph (KERG). Summarizing relationships is essential for addressing the scalability requirement of the Linked Data web scenario. IR-style ranking has been proposed to fit in relevance at the level of keywords [2]. To cope with the increased keyword ambiguity in the web setting, they employ a multilevel relevance model, where elements to be considered are keywords, entities mentioning these keywords, corresponding sets of entities, relationships between elements of the same level, and inter-relationships between elements of different levels.

II. Related Work

There are two directions of work

1. Keyword search approaches compute the most relevant structured results.
2. solutions for source selection compute the most relevant sources.

2.1 Keyword Search

There are two main categories:

2.1.1 Schema Based Approaches

There are schema-based approaches implemented on top of off-the-shelf databases [14], [9], [10], [11]. A keyword query is processed by mapping keywords to elements of the database (called keyword elements). Then, using the schema, valid join sequences are derived, which are then employed to join ("connect") the computed keyword elements to form so-called candidate networks representing possible results to the keyword query.

» Effective Keyword Search in Relational Databases[10]

In relational databases, we have three key steps for processing a given keyword query. (1) Generate all candidate answers, each of which is a tuple tree by joining tuples from multiple tables. (2) Then compute a single score for each answer. The scores should be defined in such a way so that the most relevant answers are ranked as high as possible. (3) And finally return answers with semantics.

DBXplorer [16], DISCOVER [15], BANKS [18], and Hristidis et al. are systems that support keyword search on relational databases. For the first step, they generate tuple trees from multiple tables as answers. The first three systems (DBXplorer [16], DISCOVER [15], BANKS [18]) require an answer containing all keywords in a query, while the last one only requires an answer containing some but not necessarily all keywords in the query. Efficiency has been the focus for the first step: rules are designed to avoid generation of unnecessary tuple trees, and more efficient algorithms are proposed to improve the time and space complexities. For the second step, the first two systems use a very simple ranking strategy: the answers are ranked in ascending order of the number of joins involved in the tuple trees. When two tuple trees have the same number of joins, their

ranks are determined arbitrarily. Thus, all tuple trees consisting of a single tuple are ranked ahead of all tuples trees with joins. The ranking strategy of the BANKS system is to combine two types of information in a tuple tree to compute a score for ranking: a weight (similar to PageRank for web pages) of each tuple, and a weight of each edge in the tuple tree that measures how related the two tuples are. The strategy of DBXplorer and DISCOVER and the strategy of BANKS for the second step do not utilize any state-of-the-art IR ranking methods, which have been tremendously successful. A state-of-the-art IR ranking method is used to compute a score between a given query and each text column value in the tuple tree. A final score is obtained by dividing the sum of all these scores by the number of tuples (i.e. the number of joins plus 1) in the tree. However, they only concentrate on the efficiency issue of the implementation of the ranking strategy and do not conduct any experiments on the effectiveness issue.

This paper focuses on search effectiveness. They propose a novel ranking strategy that ranks answers (tuple trees) effectively and returns answers with basic semantics

Their key contributions are as follows:

1. Identify four new factors that are critical to the problem of search effectiveness in relational databases ((1) tuple tree size normalization, (2) document length normalization, (3) document frequency normalization and (4) inter-document weight normalization).
2. Propose a novel ranking strategy to solve the effectiveness problem. Answers are returned with basic semantics.

» Kite system

A system called Kite extends schema-based techniques to find candidate networks in the multisource setting [12]. Many practical settings, however, require that they combine tuples from multiple databases to obtain the desired answers. Such databases are often autonomous and heterogeneous in their schemas and data. Kite gives a solution to the keyword-search problem over heterogeneous relational databases. It employs schema matching techniques to discover links between sources and uses structure discovery techniques to find foreign-key joins across heterogeneous sources. Such joins are critical for producing query results that span multiple databases and relations.

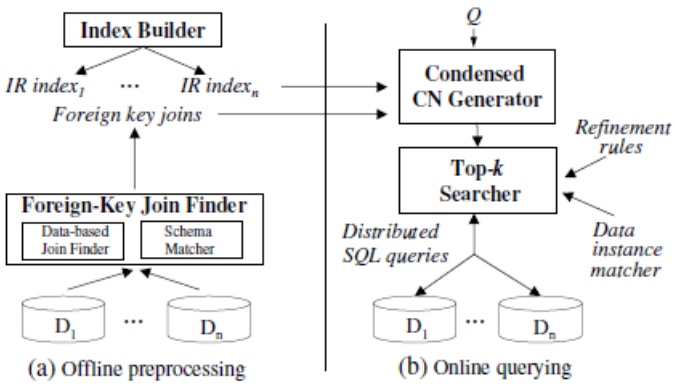

Fig.2 Kite architecture

Kite operates in two phases: offline preprocessing and online querying. In the *offline preprocessing phase* (Fig. 2), the index builder constructs standard inverted IR indexes on the text attributes of the databases. Then, the FK join finder leverages data-based join discovery and schema matching methods to identify FK joins across the databases. In the *online querying phase*, given a top-k keyword query Q, the condensed candidate network (CN) generator employs the FK joins and the IR indexes to quickly identify a space of possible answers to Q.

Kite employs a combination of structure discovery and schema matching methods that empirically outperforms current join discovery algorithms. Kite develops a novel adaptive solution for selecting strategies, which monitors and changes exploration strategies on-the-fly, whenever the current strategy no longer appears effective

The problem of keyword search over multiple heterogeneous relational databases is important in many practical settings, and will become increasingly so as the number of such databases grows. They showed that a multi-database setting raises several novel challenges, and renders current single-database algorithms ineffective. To address these challenges, they introduced Kite algorithm. Experimental evaluation suggests that Kite scales well to multiple databases, significantly outperforms our baseline adaptation of single-database algorithms, and produces high-quality results with no need for human reconciliation of the different databases.

2.1.2 Schema-Agnostic Approaches

Systems for "schema-agnostic" keyword search on databases, such as DBXplorer [16], BANKS [18] and Discover [15], model a response as a tree connecting nodes (tuples) that contain the different keywords in a query (or more generally, nodes that satisfy specified conditions). Here "schema-agnostic" means that the queries need not use any schema information (although the evaluation system can exploit schema information). For example, the query "Gray transaction" on a graph derived from DBLP may find Gray matching an author node, transaction matching a paper node, and an answer would be the connecting path; with more than two keywords, the answer would be a, connecting tree. The tree model has also been used to find connected Web pages, that together contain the keywords in a query .

Schema-agnostic approaches [3], [4], [8], [13] operate directly on the data. Structured results are computed by exploring the underlying data graph. The goal is to find structures in the data called Steiner trees (Steiner graphs in general), which connect keyword elements [8].

» EASE: An Effective 3-In-1 Keyword Search Method For Unstructured, Semi-Structured And Structured Data[8]

In this paper, they propose an efficient and adaptive keyword search method, called EASE, for indexing and querying large collections of heterogeneous data. To achieve high efficiency in processing keyword queries, we first model unstructured, semi-structured and structured data as graphs, and then summarize the graphs and construct graph indices instead of using traditional inverted indices. They propose an extended inverted index to facilitate keyword-based search, and present a novel ranking mechanism for enhancing search effectiveness. They have conducted an extensive experimental study using real datasets, and the results show that EASE achieves both high search efficiency and high accuracy, and outperforms the existing approaches significantly.

» BLINKS: Ranked Keyword Searches on Graphs[4]

A top-*k* keyword search query on a graph finds the top *k* answers according to some ranking criteria, where each answer is a substructure of the graph containing all query keywords. BLINKS, a bi-level indexing and query processing scheme for top-*k* keyword search on graphs. BLINKS follow a search strategy with provable performance bounds, while additionally exploiting a bi-level index for pruning and accelerating the search. To reduce the index space, BLINKS partitions a data graph into blocks: The bi-level index stores summary information at the block level to initiate and guide search among blocks, and more detailed information for each block to accelerate search within blocks.

Their main contributions are the following:

1. *B*etter search strategy.
2. Combining indexing with search.
3. Partitioning-based indexing.

2.2 Database Selection

The wide popularity of free-and-easy keyword based searches over World Wide Web has fueled the demand for incorporating keyword-based search over structured databases. However, most of the research work focuses on keyword-based (2.1) searching over a single structured data source. With the growing interest in distributed databases and service oriented architecture over the Internet, it is important to extend such a capability over multiple structured data sources. One of the most important problems for enabling such a query facility is to be able to select the most useful data sources relevant to the keyword query.

More closely related to this work existing solutions to database selection, where the goal is to identify the most relevant databases. The main idea is based on modeling databases using keyword relationships. A keyword relationship is a pair of keywords that can be connected via a sequence of join operations. For instance, <PuneUniversity, Award> is a keyword relationship as there is a path between uni1 and prize in Fig. 1. A database is relevant if its keyword relationship model covers all pairs of query keywords. MKS [1] captures relationships using a matrix. Since M-KS considers only binary relationships between keywords, it incurs a large number of false positives for queries with more than two keywords. This is the case when all query keywords are pairwise related but there is no combined join sequence which connects all of them.

G-KS [2] addresses this problem by considering more complex relationships between keywords using a keyword relationship graph (KRG). Each node in the graph corresponds to a keyword. Each edge between two nodes corresponding to the keywords <ki, kj> indicates that there exists at least two connected tuples ti →tj that match ki and kj. Moreover, the distance between ti and tj are marked on the edges. Keyword relationship graphs are utilized for computing the similarity between each database and a KS query, so that, during query processing, only the most promising databases are searched. *G-KS*, a novel method for selecting the top-*K* candidates based on their potential to contain results for a given query.

» A Graph Method for Keyword-based Selection of the topK Databases [2]

» This paper proposes *G-KS*, a novel graph-based method for relational KS over multiple databases. They show how to construct, maintain and compress the *KRGs* in order to reduce the pre-processing and storage overhead.

» They present an IR-inspired method to compute the importance of nodes and edges, and an algorithm to estimate the potential of a join solution containing all query keywords.

» Confirm the efficiency and effectiveness of *G-KS* through extensive experiments with real datasets in a client-server architecture.

Compared to M-KS, G-KS computes more relevant sources due to these differences: G-KS adopts IR-style ranking to compute TF-IDF for keywords and for keyword relationships. Further, it helps to reduce the number of false positives. It provides an additional level of filtering, validating connections between keywords based on complex relationships and distance information in the KRG. *M-KS* considers all keyword terms to be equally important, and simply counts the number of term co-occurrences to compute the weights of distance relationships in the matrix. For instance, a database that contains frequent co-occurrences for a pair of query terms would achieve a large score, although these co-occurrences do not have high discriminating value (indeed, because of their frequency). In contrast, *G-KS* normalizes the weights of nodes and edges, solving the problem of popular terms.

The second difference between the two methods is due to the use of compound nodes in *KRG*. Compound nodes represent multiple terms that occur a single time in the database, and at the same tuple. This concept leads to effective compression of the *KRG*, since all terms in a compound node share the same set of distance relationships. In contrast, *M-KS* explicitly represents the distance relationship between all pairs of terms, without utilizing any compression mechanism. The resulting *KRM* has more entries than the corresponding *KRG*, consumes more space and incurs higher cost during query processing.

The third, and most important, difference regards the effectiveness of pruning. *M-KS* simply uses the binary relationships of queried keywords in *KRM* to calculate similarity scores. Consequently, it incurs a large number of false positives for queries with more than two keywords. In contrast, the concept of the candidate graph allows *G-KS* to rise above binary relationships, and treat the query terms holistically.

Both M-KS and G-KS assume that sources are independent and answers reside within one single source. The KRG, its keywords, relationships between keywords as well as scores are derived from and built for one single database. While Keyword query routing, proposed for modeling and scoring relationships is geared toward the entire Linked Data collection. Moreover, this paper make use of a summary, which instead of capturing relationships at the level of keywords (and data elements), it operates at the level of element sets.

III. KEYWORD QUERY ROUTING

This paper uses a graph-based data model to characterize individual data sources. In that model, they distinguish between an element-level data graph representing relationships between individual data elements, and a set-level data graph, which captures information about group of elements.

Definition 1 (Element-level Data Graph). An element-level data graph g(N, E) consists of

» the set of nodes N, which is the disjoint union of $N_E \uplus N_V$, where the nodes $N_E$represent entities and the nodes N_V capture entities' attribute values, and

» the set of edges E, subdivided by E = $E_R \uplus E_A$, where E_R represents interentity relations, $E_A$stands for entity-attribute assignments. We have e(n_1, n_2) ε $E_R$iff n_1, n_2ε $N_E$and e(n_1, n_2) ε $E_A$iff n_1ε $N_E$and n_2ε N_V.

The set of attribute edges E_A (n)= {f(n, m) ε E_A} is referred to as the description of the entity n.

Definition 2 (Set-level Data Graph)

A set-level data graph of an element-level graph g($N_E \uplus N_v$, $E_R \uplus E_A$) is a tuple $g' = (N', E')$. Every node n' ε N' stands for a set of elementlevel entities $N_{n'} \subseteq N_E$, i.e., there is mapping type : $N_E \rightarrow N'$ that associates every element-level entity n ε N_E with a set-level element n' ε N'. Every edge e' (n_i', (n_j') ε E' represents a relation between the two sets of element-level entities (n_i' and (n_j'. We have E' = {e' (n_i', (n_j')|e(ni, nj) ε E_R ,type (n_i) = n_i',type (n_j) = n_j'}.

Definition 3 (Web Graph)

The web of data is modeled as a web graph W*(G*, N*, $E_i^* \uplus E_e^*$), where G* is the set of all data graphs, N* is the set of all nodes, E*i is the set of all "internal" edges that connect elements within a particular source, and E*e is the set of all "external" edges, which establish links between elements of two different sources, i.e.,G* ={g1(N_1^*, E_1^*), g2(N_2^*, E_2^*), . . . , gn(N_n^*, E_n^*)}, N* = $\cup_{l=1}^{n} N_l^*$, E_e^*={e(ni, nj)| ni ε N_i^*, nj ε N_j^*, $N_i^* \neq N_j^*$} , and E*= $\cup_{l=1}^{n} E_{l1}^*$ U E_{l1}^* When considering the nodes and edges only, we simply use W*(N*, E*). We use W(G,N, E) to distinguish the element-level web graph from the set-level web graph W'(G', N', E').

Definition 4 (Keyword Query Result)

A web graph W(N, E) contains a result for a query K = {k1, k2, . . . , $k_{|K|}$ } if there is a subgraph also called Steiner graph $W^S(N^S, E^S)$, which for all ki ε K, contains a keyword element node n_i ε N^S whose description EAðniÞ matches ki, and there is a path between $n_i$and n_j ($n_i \leftrightarrow n_j$) for all ni, nj ε N^S.

Definition 5 (Keyword Routing Plan)

Given the web graph W =(G,N, E) and a keyword query K, the mapping μ : K→2^G that associates a query with a set of data graphs is called a keyword routing plan RP. A plan RP is considered valid w.r.t. K when the union set of its data graphs contains a result for K.

Definition 6 (Set-level KERG)

Given W(G, N, E) and $W'(G', N', E')$, a set-level keyword-element relationship graph is a tuple $W'_K = (N'_K, E'_K)$. Every keyword-element node n'_K ε N'_K is a tuple (k, n', g) where n' ε N' is the corresponding set-level node it represents, g ε G is the graph it is associated with, and k is a keyword that matches some descriptions EA(n) of the entities n ε n'. There is a keyword-element relationship (KER) e'_K = <n'_{Ki}, n'_{Kj}> ε E'_Krepresenting a relationship between the keyword-element n'_{Ki} = (ki, n'_i, gi) and n'_{Kj} = (ki, n'_j, gj), iff there is an ni ε n'_i that mentions ki, an nj ε n'_j that mentions kj, and ni → nj. A dmax-KERG is a KERG, which captures only paths between ni and nj with length dmax or less (ni $\leftrightarrow^{dmax}$ nj).

Definition 7 (Routing Graph)

Given a keyword query K = {k1, k2, . . . ,k |K| } and a KERG W'_K (N'_K , E'_K), a routing graphW'^S_K (N'^S_K, E'^S_K), is a sub-graph of W'_K that satisfies the following properties:

For each query keyword k ε K, there is a node n'_{Ki}(k,n',g) εN'^S_K

It is a complete graph, i.e., for every pair of nodes n'_{Ki}, n'_{Kj} ε N'^S_Kthere is an edge < n'_{Ki}, n'_{Kj}> ε E'^S_K.

A routing graph contains a set of data sources such as routing plan. Moreover, a routing graph contains information that enables the user to assess whether it is relevant: a plan is relevant only if the nodes mentioning the keywords and relationships between them correspond to the intended information need. This additional information will be used in the evaluation to assess the effectiveness of ranking.

The procedure for computing routing plans is described in Algorithm. Given a query K and the summary W'_K, the algorithm computes a set of routing plans [RP]. For this, it first determines the join plan JP. Based on this plan, KERG relationships are retrieved for every keyword pair, and joined with the intermediate result table T. This table contains candidate routing graphs, including the scores of their constituent elements and their combined score. When the join plan is worked off, the combined score is computed for every tuple in T , i.e., for every routing graph W'^S_K. Routing graphs, which represent the same set of sources, are aggregated into one single result. This is because we want to output only those plans that capture unique combination of sources.

PPRJ – Compute Routing plan (K, W'_K)

Input : keyword query K, the summary W'_K (N'_K , E'_K)

Output : Set of routing plan [RP]

JP – join plan that contains all <k_i, k_j> ε 2^k

T – a Table where every tuple capture a join sequence of KERG relationship e'_kε E'_k

The score of each $e'_k$and combined score of the join sequence it is initially empty.

While JP empty() **do**

<k_i, k_j> ← JP.pop();

$E'_{<k_i,k_j>}$ ← retrieve (E'_k, <k_i, k_j>);

If T.empty() then

T ← $E'_{<k_i,k_j>}$;

Else

T $E'_{<k_i,k_j>}$ ⋈ T

Compute score of tuples in T via SCORE(K, W'^{S}_{K}).

[RP] ← Group T by source to identify unique combinations of source;

Compute scores of routing plans in [RP] via SCORE(K,RP)

Sort [RP] by score.

Such a way they have offered a solution to the novel problem of keyword query routing. Based on modeling the search space as a multilevel inter-relationship graph, they proposed a summary model that groups keyword and element relationships at the level of sets, and developed a multilevel ranking scheme to incorporate relevance at different dimensions. This paper showed that when routing is applied to an existing keyword search system to prune sources, considerable performance gain can be achieved.

IV. CONCLUSION

In this paper we have given different keyword search techniques and database selection techniques. Keyword search categorized into schema-based approaches and schema-agnostic approaches. Keyword search approaches computes the most relevant structured results. Database selection computes most relevant sources and gives solutions for source selection. Recently one new system existed for keyword search and it utilizes routing plans. This system used to route keywords only to relevant sources to reduce the high cost of processing keyword search queries over all sources. In this case given keyword query is searched within relevant sources only, so the time required is less as compared to previous work.

References

1. B. Yu, G. Li, K.R. Sollins, and A.K.H. Tung, "Effective Keyword- Based Selection of Relational Databases," Proc. ACM SIGMOD Conf., pp. 139-150, 2007.
2. Q.H. Vu, B.C. Ooi, D. Papadias, and A.K.H. Tung, "A Graph Method for Keyword-Based Selection of the Top-K Databases," Proc. ACM SIGMOD Conf., pp. 915-926, 2008.
3. V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, "Bidirectional Expansion for Keyword Search on Graph Databases," Proc. 31st Int'l Conf. Very Large Data Bases (VLDB), pp. 505-516, 2005.
4. H. He, H. Wang, J. Yang, and P.S. Yu, "Blinks: Ranked Keyword Searches on Graphs," Proc. ACM SIGMOD Conf., pp. 305-316, 2007.
5. Thanh Tran and Lei Zhang, "Keyword Query Routing, Ieee Transactions On Knowledge And Data Engineering," VOL. 26, NO. 2, FEBRUARY 2014
6. G. Ladwig and T. Tran, "Index Structures and Top-K Join Algorithms for Native Keyword Search Databases," Proc. 20th ACM Int'l Conf. Information and Knowledge Management (CIKM), pp. 1505-1514, 2011.
7. Pratiksha Nikam and Prof. Srinu Dharavath, "Review of Approximate String Search in Spatial Dataset," International Journal of Multidisciplinary and Current Research Vol.2 (March/April 2014)
8. G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou, "Ease: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-Structured and Structured Data," Proc. ACM SIGMOD Conf., pp. 903-914, 2008.
9. V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-Style Keyword Search over Relational Databases," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), pp. 850-861, 2003.
10. F. Liu, C.T. Yu, W. Meng, and A. Chowdhury, "Effective Keyword Search in Relational Databases," Proc. ACM SIGMOD Conf., pp. 563-574, 2006.
11. Y. Luo, X. Lin, W. Wang, and X. Zhou, "Spark: Top-K Keyword Query in Relational Databases," Proc. ACM SIGMOD Conf., pp. 115-126, 2007.
12. M. Sayyadian, H. LeKhac, A. Doan, and L. Gravano, "Efficient Keyword Search Across Heterogeneous Relational Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 346-355, 2007.
13. B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-K Min-Cost Connected Trees in Databases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE), pp. 836-845, 2007.
14. T. Tran, H. Wang, and P. Haase, "Hermes: Data Web Search on a Pay-as-You-Go Integration Infrastructure," J. Web Semantics, vol. 7, no. 3, pp. 189-203, 2009.
15. V. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB), pp. 670-681, 2002.

16. S. Agrawal, S. Chaudhuri, and G. Das, " DBXplorer: A systembfor keyword-based search over relational databases," In ICDE, 2002.
17. A. Balmin, V. Hristidis, and Y. Papakonstantinou, " ObjectRank: Authority-based keyword search in databases," In VLDB, 2004.
18. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS, " In ICDE, 2002.

