Toward Industrial-Strength Keyword Search Systems over Relational Data

Akanksha Baid, Ian Rae, AnHai Doan, Jeffrey F. Naughton

Computer Science Department, University of Wisconsin, Madison

1210 W. Dayton St, Madison, WI, USA.

{baid, ian, anhai, naughton}@cs.wisc.edu

Abstract – **Keyword search (KWS) over relational data, where the answers are multiple tuples connected via joins, has received significant attention in the past decade. Numerous solutions have been proposed and many prototypes have been developed. Building on this rapid progress and on growing user needs, recently several RDBMS and Web companies as well as academic research groups have started to examine how to build industrial-strength keywords search systems. This task clearly requires addressing many issues, including robustness, accuracy, reliability, and privacy, among others. A major emerging issue, however, appears to be performance related: current KWS systems have unpredictable run time. In particular, for certain queries it takes too long to produce answers, and for others the system may even fail to return (e.g., after exhausting memory).**

In this paper we begin by examining the above problem and arguing that it is a fundamental problem unlikely to be solved in the near future by software and hardware advances. Next, we argue that in an industrial-strength setting, to ensure real-time interaction and facilitate user adoption, KWS systems should produce answers under an absolute time limit and then provide users with a description of what could be done next, should he or she choose to continue. Next, we show how to realize these requirements for DISCOVER, an exemplar of a recent KWS solution approach. Our basic idea is to produce answers as in today's KWS systems up to the time limit, then show users these answers as well as query forms that characterize the unexplored portion of the answer space. Finally, we present some preliminary experiments over real-world data to demonstrate the feasibility of the proposed solution approach.

I. INTRODUCTION

The success of search engines demonstrates that naïve users are comfortable using keyword search to find documents of interest to them. Over the past decade, this success has spawned 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 [1], DISCOVER [15], and BANKS [4] were amongst the first systems that supported keyword search over relational databases, and many other systems [e.g. 2, 9, 13, 14, 16, 20, 22, 24, 25, 26, 28, 31] have since been developed. As more structured data become available at organizations and on the Web, and as more naïve users want to use such data, we expect that more and more efforts will be devoted to transition research solutions into industrial-strength systems.

Naturally, building such systems requires addressing many issues, including robustness, accuracy, reliability, and privacy, among others. However, from our own effort, we have been struck by how quickly we were hit with a roadblock: current KWS solutions have unpredictable performance issues. Specifically, while for many user queries the system produces answers quickly, for many others it takes unacceptably long (e.g., tens of minutes), or even fails to produce any answer after exhausting memory. Clearly, such a performance profile is unacceptable in an industrial-strength system.

We argue that in an industrial-strength setting, KWS systems should produce answers under an absolute time limit (say a few seconds); even if such answers are only partial in some sense. We then argue that when showing these (possibly partial) answers, the system should also somehow characterize the portion of the answer space that is as yet unexplored, so the user knows what it is that he or she is potentially missing. The system then provides a way for the user to explore this portion of the answer space should he or she choose to do so. We propose that one way to do so is to provide form interfaces to characterize the yet-unexplored answer space. Overall, we believe that 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.

Overall, we hope that our work will be viewed as an attempt to open the debate on what it takes to transition the wealth of current KWS solutions from prototype to industrial-strength systems that will be widely adopted by real-world customers.

Toward this goal, we make the following contributions:

(1) An examination of the performance bottlenecks of current KWS solutions
(2) A proposed fix, based on the ideas of "time limit" and "overview of the yet-unseen using query forms" for industrial-strength KWS systems.
(3) A prototype implementation of the fix for DISCOVER, an exemplar of a recent major KWS solution approach, and
(4) Experiments with a real-world dataset to demonstrate the effectiveness and feasibility of the proposed approach

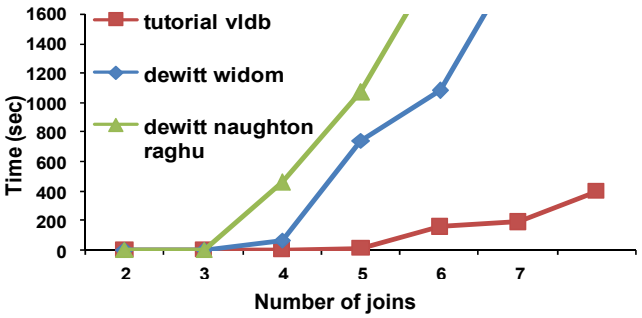

Figure 1: The performance of KWS degenerates as the number of joins grows.

II. Limitations of Current Keyword Search Solutions over Relational Data

KWS systems suffer from a fundamental problem: they cannot guarantee a performance "cap" in the sense that certain queries may take a very long time to run. To illustrate this point, we have conducted some experiments with DISCOVER on DBLife (dblife.cs.wisc.edu) [7, 8], a real-world data set that describes entities and relationships in the database research domain.

We posed some randomly sampled queries to DISCOVER and measured the time it takes from when a query is posed until all its answers are displayed. Figure 1 shows this time for three queries plotted against the allowed number of joins. The figure shows that for the queries "dewitt widom" and "dewitt naughton raghu", as we increase the number of allowed joins beyond 3 or 4, the run time increases dramatically.

There are two fundamental reasons for this degradation of performance. First, at its core the candidate network (CN) generation problem contains the problem of searching a graph to find all sub-graphs that satisfy certain properties. It is well known that variations of this problem are NP-complete [21]. It has been shown that it is possible to further optimize DISCOVER to reduce the absolute run time as well as memory consumed [11, 21]. However, it is unlikely that we can escape the exponential growth for both, due to the NP-completeness of the problem.

Second, the SQL execution step also often takes a large amount of time. Perhaps not surprisingly, as the number of allowed joins increases, increasingly more expensive SQL queries are generated. Query optimization carried out by DISCOVER mitigates the problem somewhat. Nevertheless, the fundamental problem remains.

This *unpredictability* is fundamental, and poses a serious problem for adopting such systems in practice. In the next section we propose our solution to address this problem.

III. Basic Ideas of our Solution

Our solution builds on two key ideas. First, we believe that an industrial-strength KWS solution, if it is to be adopted widely, must be *responsive* and *predictable* in that it should produce some answers in a matter of seconds.

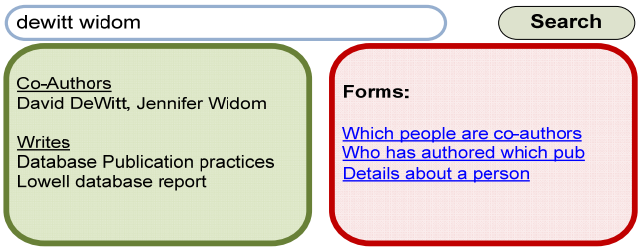

Figure 2: A mock screenshot of our proposed solution approach, which returns both data tuples (left pane) and forms (right pane) as the answer to a keyword query.

Second, keyword search is often *interactive* and *iterative* by nature, in that a user issues a keyword query, examines the result, then issues another query, then examines the result, and so on. To enable this paradigm, it is important that keyword search takes only a few seconds, so that users *feel* that the process is naturally interactive and iterative.

To implement the above idea, a natural solution is to impose an *absolute time limit* on the keyword search system: when the time limit has been reached, a result must be returned to the user no matter what. We note that a similar time-limit idea is used in virtually all major keyword search engines on the Web.

Now suppose the system has reached the time limit and has returned some (possibly partial) answers to the user. If the system terminates here, then clearly it has explored only a portion of the answer space, and so it runs the risk of having shown the users only some suboptimal answers, or worse, missed desired answers all together.

The above observation leads us to the second idea. In such cases, we believe it is highly desirable that the system can somehow give the user an idea about what the unexplored portion of the answer space "looks like" and what the user can do to explore that portion. This will also give the user a way to continue the search for the desired answer, in an interactive and iterative fashion.

We propose that one way to do this is to use query forms (KWS-F) as proposed in [6]. Intuitively the answer space can be characterized using a set of SQL queries. This set of SQL queries in turn can be characterized using a set of query forms, which most users know and can easily use. Figure 2 illustrates this proposal. Given the keyword query "dewitt widom", after an absolute time limit has been reached, the system returns a set of answers on the left (of the result page), just like in a traditional KWS system. But unlike these KWS systems, our system also returns a set of forms to characterize the unexplored potion of the answer space (on the right of the result page). The first form in this figure, for example, would allow the user to explore all answers that relate persons via co-authorship.

If the user does not find what he or she wants in the answers on the left, he or she can examine the forms on the right, and then click on a desired form to fill and submit.

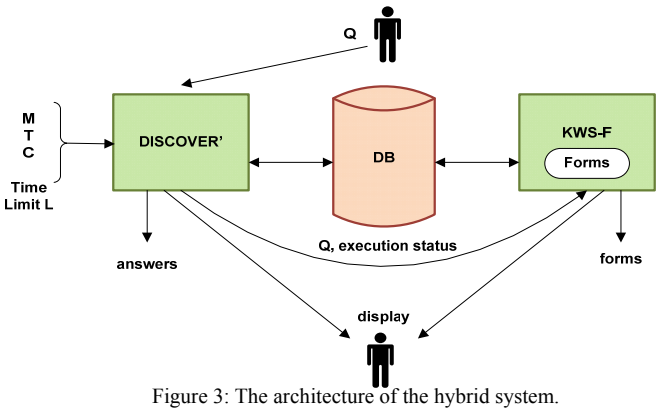

Figure 3: The architecture of the hybrid system.

While applicable, we have found that KWS-F alone is clearly not a good idea. While KWS-F is highly appropriate for hard queries (with long join sequences), it is not well suited for easy queries (for which KWS systems are both more efficient and more user-friendly). We argue that adding KWS-F, the form-based approach, to DISCOVER in a sense is combining the best of both approaches. Based on this argument we propose a hybrid approach in the next section.

IV. Hybrid Approach

As pointed out earlier, CN generation can take up a large portion of the total query execution time in DISCOVER. DISCOVER restricts CN generation time by restricting the size of the join trees considered with a tunable parameter M_{max}. Doing so however can severely restrict query coverage and leave the user with no option for queries that DISCOVER cannot evaluate. In the hybrid approach, we propose that forms be displayed for any queries for which CNs were not generated.

Having this option of presenting forms for which CNs were not generated yields a two-fold benefit – (i) the user can now issue queries that DISCOVER could not express, and (ii) overall CN generation time can be reduced by setting lower termination thresholds for CN generation without compromising query coverage.

In the hybrid approach, we allow CN generation to be restricted using two parameters – (1) *Timeouts*: The idea here is simple. Most users expect the search results to be returned to them in a reasonable time. In our implementation of the hybrid approach we limit CN generation time to a tunable threshold time *T*. All CNs generated within time *T* are evaluated using DISCOVER. Forms are displayed for the remaining CNs. (2) *Number of Joins*: Like DISCOVER, the hybrid approach also uses a value *M* (maximum number of allowed joins) to restrict the number of CNs that the system evaluates.

We also find that as the number of queries grows, a small number of queries (typically those involving long joins) take up a disproportionately large portion of the execution time. Evaluating these expensive queries might be unnecessary if we are not even sure if the user is interested in them. Consequently, in the hybrid system the queries generated from the CNs are partitioned into two sets.

The first set contains queries that have a cost <= *C,* where *C* is measured in *disk-page-fetches.* These queries are executed on the underlying data and the corresponding tuples are returned to the user. The second set contains all queries with estimated cost > *C.* The queries in the second set are not executed. Instead forms corresponding to these queries are presented to the user and the queries are evaluated only if the user chooses to do so.

The system architecture for the proposed hybrid approach is presented in Figure 3. In this system, a keyword query Q is first sent to DISCOVER', which is the DISCOVER system modified to operate within the limits of M, T, C and L (where, L is defined as a combination of M , L and C).

Specifically, the modified system does not generate CNs of size greater than M. It takes only up to time T to generate CNs, and it executes only SQL queries estimated to cost no more than C. Eventually DISCOVER' will terminate, either because (a) it has finished executing all generated SQL queries, or (b) the time limit L has been reached, whichever is earlier.

At this point, DISCOVER' sends query Q together with a status report on its execution to the KWS-F form-based system. This system executes query Q to obtain a ranked list of forms like in a KWS-F system [6]. The hybrid system then combines this list of forms with the list of answers produced by DISCOVER', then presents this combination to the user.

V. Experimental Evaluation

We now experimentally evaluate our approach. We used a 40M snapshot of the DBLife data set as of June 2007, which has 801,189 tuples in 14 tables. We ran our experiments using PostgreSQL 8.3.6 on an Intel(R) Core(TM) 2 Duo 2.33 GHz system with 3GB of RAM. All algorithms were implemented in Java and JDBC was used to connect to the database.

We focus on the three "hard" queries – (a) "tutorial vldb", (b) "dewitt widom", and (c) "dewitt naughton raghu". For DISCOVER, we measure its time to be the time between the moment a query is issued and the time the system stops. For the hybrid approach we measure the time it takes from when a query is issued until when the system stops.

We set the time limit to 90 seconds and T = 60 seconds and C = 30,000 pages. Given this setting, Figure 4 shows the run time of DISCOVER versus the hybrid approach on the three selected queries, as we increase the number of allowed joins (parameter M) on the X-axis. The figure shows a stark difference in run time. First, it is clear that after a certain M threshold, DISCOVER's time grows exponentially. In fact, it failed to return results for queries (b) and (c) for large M values. In contrast, the hybrid system maintains a low run time throughout.

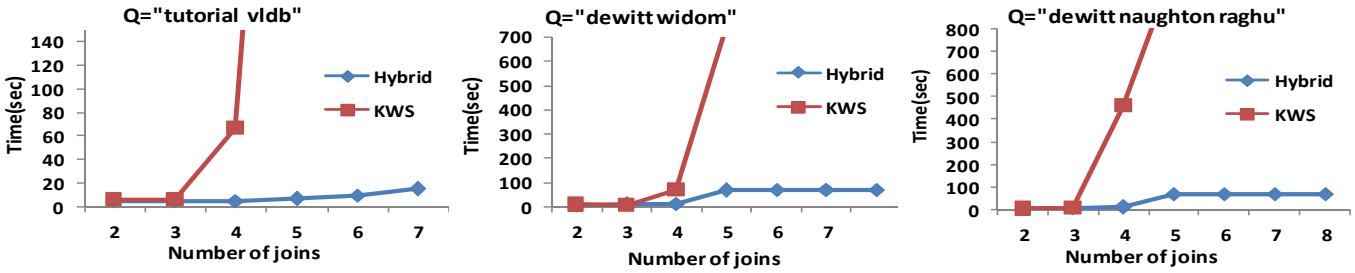

Figure 4: Comparison of the performance of KWS vs. Hybrid on three selected queries.

We easily found cases where DISCOVER could not answer the queries due to the many joins required or to the large intermediate results. Examples include finding papers co-authored by DeWitt and three other persons, finding papers co-authored by four persons, and finding interesting relationships among four given persons. In all of these cases, the hybrid approach stopped, the right forms appeared in the list of forms displayed, and executing these forms took only milliseconds.

In summary, the experiments demonstrate that DISCOVER has a severe performance problem for certain queries, and that the hybrid approach consistently maintained a low run time and could handle these queries.

VI. CONCLUSION

Our goal in this paper was to explore techniques that allow keyword search over relational data to be implemented in such a way that the system can guarantee a reasonable response time. Our main idea is to let the traditional keyword search generate all the answers it can within some time bound, and to augment the search with a form-based approach that "covers" potential answers that the keyword search could not find in the specified time limit. Results from experiments with this approach indicate that it is successful in always returning a covering combination of answers and forms in a bounded and predictable amount of time. We regard this work as a first step toward building this kind of system, and hope that it is a springboard for follow-on work that improves the performance and quality of such systems.

ACKNOWLEDGEMENT

We thank Yannis Papakonstantinou and Vagelis Hristidis for providing us with their implementation of DISCOVER.

REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System for Keyword-Based Search over Relational Databases. ICDE, 2002.

[2] Balmin, A., Hristidis, V., Papakonstantinou, Y. "ObjectRank: authority-based keyword search in databases", *VLDB*, 2004.

[3] A. Bernstein and E. Kaufmann. Making the semantic web accessible to the casual user: Empirical evidence on the usefulness of semiformal query languages. IEEE TKDE, under review.

[4] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword Searching and Browsing in Databases using BANKS. ICDE, 2002.

[5] K. Chakrabarti, S. Chaudhuri, S. Hwang. Automatic Categorization of Query Results. SIGMOD 2004.

[6] E. Chu, A. Baid, X. Chai, A. Doan, J. Naughton, "Combining Keyword Search and Forms for Ad Hoc Querying of Databases" SIGMOD 2009.

[7] P. DeRose et al. DBLife: A Community Information Management Platform for the Database Research Community. CIDR 2007 Demo.

[8] P. DeRose, W. Shen, F. Chen, A. Doan, R. Ramakrishnan "Building Structured Web Community Portals: A Top-Down, Compositional, and Incremental Approach". VLDB 2007.

[9] Ding, B., Yu, J. X., Wang, S., Qin, L., Zhang, X., Lin, X. "Finding topk min-cost connected trees in databases", *ICDE*, 2007.

[10] D. W. Embley. NFQL: The Natural Forms Query Language. ACM Transaction Database System, 1989.

[11] K. Golenberg, B. Kimelfeld, Y. Sagiv, "Keyword proximity Search in Complex Data Graphs". SIGMOD 2008.

[12] A. Halevy et al. Crossing the Structure Chasm. CIDR 2003.

[13] He, H., Wang, H., Yang, J., Yu, P., "BLINKS: ranked keyword searches on graphs", *SIGMOD*, 2007.

[14] V. Hristidis, L. Gravano, Y. Papakonstantinou. Efficient IR-Style Keyword Search over Relational Databases. VLDB 2003.

[15] V. Hristidis, Y. Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. VLDB, 2002.

[16] Hristidis, V., Papakonstantinou, Y., Balmin, A. "Keyword proximitysearch on XML graphs", *ICDE*, 2003.

[17] M. Jayapandian, H. V. Jagadish. Automated Creation of a Form-based Database Query Interface. VLDB 2008.

[18] M. Jayapandian, H. V. Jagadish. Automating the Design and Construction of Query Forms. ICDE 2006.

[19] M. Jayapandian, H. V. Jagadish. Expressive Query Specification through Form Customization. EDBT 2008.

[20] Kacholia, V., et al., Bidirectional expansion for keyword search on graph databases, VLDB 2005.

[21] B. Kimelfeld , Y. Sagiv, "Finding and Approximating Top-k Answers in Keyword Proximity Search". PODS 2006.

[22] R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, H. Zhu. Using Structured Queries for Keyword Information Retrieval. IBM Technical Report RJ 10413.

[23] G. Koutrika, Z. Mohammadi Zadeh, H. Garcia-Molina. Data Clouds: Summarizing Keyword Search Results over Structured Data. EDBT 2009.

[24] F. Liu, C. Yu, W. Meng, A. Chowdhury. Effective Keyword Search in Relational Databases. SIGMOD 2006.

[25] Luo, Y., Lin, X., Wang, W., Zhou, X. "SPARK: Top-*k* keyword queryin relational databases", *SIGMOD*, 2007.

[26] Markowetz, A., Yang, Y., Papadias, D., "Keyword search on relational data streams", *SIGMOD*, 2007.

[27] A. Nandi , H.V. Jagadish, "QUNITS :queried units for database search". CIDR 2009

[28] M Sayyadian, H. LeKhac, A. Doan, L Gravano, "Efficient keyword search across heterogeneous relational databases", *ICDE*, 2007.

[29] A. Simitsis, G. Koutrika, and Y. Ioannidis. Pr´ecis:from unstructured keywords as queries to structured databases as answers. VLDB Journal, pages 117–149, 2008.

[30] R. Song, Z. Luo, J. Wen, Y. Yu, and H. Hon. Identifying ambiguous queries in web search. WWW 2007.

[31] Tata, S. and G.M. Lohman, SQAK: doing more with keywords. SIGMOD 2008.

[32] M.M. Zloof. Query-by-Example: the Invocation and Definition of Tables and Forms. VLDB 1975.