

Meaningful Keyword Search over Databases with Complex Schema

Mehdi Kargar*, Aijun An[#], Parke Godfrey[#],
Jaroslaw Szlichta[§], and Xiaohui Yu[#]

*University of Windsor, Windsor, Canada

[#]York University, Toronto, Canada

[§]University of Ontario Institute of Technology, Oshawa, Canada
mkargar@uwindsor.ca, aan@cse.yorku.ca, godfrey@cse.yorku.ca,
jaroslaw.szlichta@uoit.ca, xhyu@yorku.ca

Abstract. Keyword search in databases helps users who are not familiar with the database schema or a query language to explore the database efficiently. An answer is a join tree of tuples that contains the query keywords. We present a new framework for finding meaningful answers to the keyword search problem over databases with complex schema. The framework uses schema-based ranking methods to rank join trees that cover the keyword roles. The experiments on TPC-E database establish the naturalness of the proposed ranking methods.

1 Introduction

Much of the world’s high-quality data stays under lock and key in relational databases. Access is gained through relational query languages such as SQL. However, a lay user—anyone who does not know SQL *or* who is not well versed in the given schema—is effectively locked out [1, 4]. As the schemas of the datasets that organizations field become more complex, we all effectively become lay users. Keyword search has recently been used for extracting information from relational databases.

An answer to the query is a set of tuples from the database that *cover* the keywords of the query, and a natural *structure* (i.e., a tree from the database’s schema) that spans those tuples. An important issue in keyword search is to score answers for *relevance*. Prior work has addressed relevance. In [3], a simplistic solution of scoring relevance as the reciprocal of the number of edges in an answer’s tree was proposed. In [2], the authors apply information retrieval (IR) measures to rank answers. However, both of these approaches fail to return relevant answers when the schema is large and complex.

To motivate our approach and illustrate the shortcomings of existing methods, consider the query “*Arden Cynthia*” over the TPC-E¹ schema. Assume that *Arden* refers to a company name and *Cynthia* refers to a customer name. Fig. 1 shows three possible join trees of different sizes that could produce answers for this query. If we rank the trees according to their size (i.e., the number of edges), the first tree (a) gets the highest rank. An answer derived from this tree would

¹ <http://www.tpc.org/tpce/>

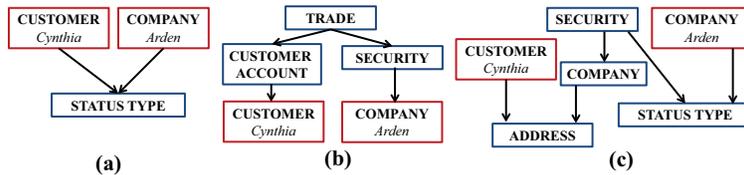


Fig. 1: Possible trees (i.e., MJNSs) for the query “Arden:company Cynthia:customer”.

say that both *Arden* and *Cynthia* are *active*. Looking at the TPC-E database, it turns out that all the customers and companies have the *Active* status. Therefore, any given customer and any given company in the TPC-E database share the same status through the *status type* table. Thus, the first found relationship is in fact, not that interesting or relevant. The second tree (b) states that company *Arden*’s stock is traded by customer *Cynthia*. This is one of the strongest relations between a customer and a company and is related to the purpose of the TPC-E schema. The third tree (c) is the largest and therefore it is not straightforward to interpret. This tree states that the company *Arden* has the same status as a security which is related to a company that has the same address as customer *Cynthia*. Two relations, *address* and *status type*, that are involved in this tree makes it less meaningful and more difficult to interpret for the user. Using the IR score ranking also ranks the first tree as the best answer.

In this paper, we use schema-based ranking to solve the problem of keyword search over relational databases. We *redefine* answer via *roles* that captures important answers missed by previous techniques. We devise importance measures for nodes, importance measures for edges, and a hybrid measure of the two. Then, we devise relevance measures for join trees that are derived from the schema relevance. Last, we like to mention this short paper is a summary of our recently published results in [5, 6].

2 Ranking Models

In our framework, the role of a query keyword is identified first. For each query keyword, we first find the list of columns (and relations) that contain the keyword using an inverted index. Then, a user-interface helps the user to specify the role of each keyword without needing to know the database schema (see [5] for screen shots of the interface). With specified *keyword roles*, our algorithm searches for answers to cover the input keywords. Each answer is a *minimal joining network of tuples that covers the input roles*. For brevity, we refer to such answers as *final answer*. The final answers defined above can be generated through a sequence of join operations on the database. To generate such answers, we first generate the *minimal joining networks of schemas (MJNSs)* that represent the join operations for producing the final answers. Three possible MJNSs for the query “Arden:company Cynthia:customer” over TPC-E schema are shown in Fig. 1. To generate MJNSs, we use a breadth-first search algorithm [3, 6]. After the MJNSs are generated, final answers can be produced by creating an execution plan (i.e., SQL query) to evaluate the MJNSs. We focus on how to rank MJNSs so that the most interesting answers will be presented to the user first.

Since many final answers can be generated for a query but some answers may not be interesting, we aim at producing top- k most interesting final answers. To achieve this, we first limit the number of nodes in an MJNS. This limits the size of final answers too. The rationale is two-fold. First, if two tuples in a final answer are far away from each other, it is not easy to interpret the answer. Second, executing the query associated with a large MJNS is time consuming. Thus, a size control parameter, D_{max} , is used to specify the maximum number of allowed nodes in an MJNS. In addition and **more importantly**, we rank the generated MJNSs according to an interestingness measure so that final answers from the top-ranking MJNS are produced first. If the number of final answers produced so far is less than k , the next MJNS is used to produce more final answers until k final answers are produced. We propose and use the following methods for ranking minimal joining networks of schemas (MJNSs).

Ranking by Importance of Nodes: Given a database D , this method assumes that the importance of an MJNS M is related to the importance of the tables in D that instantiate the schemas in M . We propose a ranking method called *key entropy transfer* (KE) that ranks the tables in D . Consider G_D as an undirected graph representing a database D . The *KE* method builds a node-to-node transition probability matrix M based on the entropies of table attributes, and performs a random walk on G_D with M . The steady-state probabilities of the random walk are assigned as the importance scores of the tables. The entropies of table attributes are calculated as follows. Let $r.A$ denote an attribute A in table r , and let a represent a value of $r.A$. The *entropy* of $r.A$ is: $H(r.A) = -\sum_{\forall a \in r.A} p(a) \times \log p(a)$ where $p(a)$ is the probability that a occurs in column $r.A$ (i.e. $P(r.A = a)$).

Ranking by Importance of Edges: Intuitively, edge strength can be measured by the fraction of the join key values being instantiated. The more fraction of primary key values are instantiated, the more important the edge is. We propose the following measure, called *instantiation fraction* (IF), to quantify the importance of an edge based on the fractions of instantiated key values: $ST_{IF}(r.A, r'.A') = \frac{N_{r.A}^{inst}}{N_r} \times \frac{N_{r'.A'}^{inst}}{N_{r'}}$. In this equation, $N_{r.A}^{inst}$ is the number of tuples in relation (i.e., table) r that instantiates the edge between $r.A$ and $r'.A'$, and N_r is the total number of tuples of table r . To rank the MJNSs by edge importance, we compute a score for each MJNS using its average edge importance score.

The Hybrid Ranking Model: Our last approach to ranking MJNSs is to rank them based on the importance of both nodes and edges. The hybrid formula is defined as follows: $ST_{IF-KE}(R.A, R'.A') = ST_{IF}(R.A, R'.A') \times \frac{KE(R)+KE(R')}{2}$. Since the value of $ST_{IF}(R.A, R'.A')$ lies between zero and one, $KE(R)$ and $KE(R')$ should be normalized into range $[0, 1]$. To rank the MJNSs, the score of an MJNS is computed as the average hybrid score of the relations in the MJNS.

Runtime Discussion: The runtime of our methods are similar to that of Discover I [3]. All MJNSs have to be found for the keyword query under the D_{max} threshold. The primary overhead is evaluating the SQL queries associated with the MJNSs. For us, this overhead is marginally more than Discover I because our networks can be marginally larger due to roles.

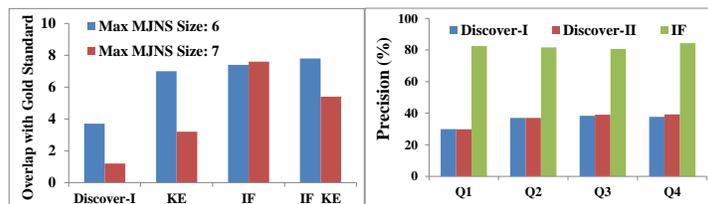


Fig. 2: The overlap with gold standard and precision of answers (top-10).

3 Experimental Evaluation

We evaluate our proposed ranking methods for finding the most meaningful MJNSs. All of the evaluated methods are implemented in Java. The experiments are conducted over the TPC-E database. TPC provides a transaction log for TPC-E which we use to generate a gold standard for ranking MJNSs. The focus of this work is ranking the MJNSs. The input to our MJNS generator is the keyword roles. That is, the main input to the ranking methods is a set of relation names (i.e., query keyword roles). We use the following four query keyword roles in the experiments: *Q1*: Customer, Company; *Q2*: Customer, Broker; *Q3*: Customer, Company, Industry; *Q4*: Customer, Company, Broker, Security. Note that in addition of the following experiment, additional experiments in [6] also verify the effectiveness of our ranking methods.

The results of the top-10 overlap with the gold standard are presented in Fig. 2. We also present the results of the Discover I algorithm that ranks final answers based on their size [3]. Note that Discover II produces the same ranking of MJNSs [2]. Generally, the edge importance ranking method *IF* works better than others. To see how effective ranking of MJNSs impacts the *final answers*, we compare the top-10 final answers from our *IF* method with the final answers produced by Discover I [3] and Discover II [2]. We use four sets of keywords related to the above keyword roles. For example, the first query is "Jacob Insurance" in which "Jacob" is associated with the *customer* table and "Insurance" is associated with the *company* table. We ask eight graduate students to judge the relevancy of the answers. A user assigns a score between 0 and 1 to each final answer, where 1 means completely relevant and 0 means completely irrelevant to the query. The top-10 precision for each query are presented in Fig. 2. Clearly, the *IF* method achieves much better precision in all the queries. The reason for Discover-I and Discover-II to have a lower precision is that in most of the cases, the tuples of the final answers are connected together by the dimension tables (e.g. *status type*) and fact tables are not involved.

4 Conclusions

We improve relevance scoring of answers based on their networks (join trees) in light of complex database schema. We propose a series of measures, and algorithms to compute them based on the importance of nodes and edges to capture the intended semantic of queries. In this work we sought to demonstrate how effective deriving relevance of the nodes and edges of the database schema could be based on just the schema and data, by no means are we advocating that auxiliary information (such as Linked Data) cannot improve it.

References

1. Bergamaschi, S., Domnori, E., Guerra, F., Lado, R.T., Velegrakis, Y.: Keyword Search over Relational Databases: A Metadata Approach. In: SIGMOD. pp. 565–576 (2011)
2. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-Style Keyword Search over Relational Databases. In: VLDB. pp. 850–861 (2003)
3. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword Search in Relational Databases. In: VLDB. pp. 670–681 (2002)
4. Kargar, M., An, A.: Efficient Top-k Keyword Search in Graphs with Polynomial Delay. In: ICDE. pp. 1269–1272 (2012)
5. Kargar, M., An, A., Cercone, N., Godfrey, P., Szlichta, J., Yu, X.: MeanKS: Meaningful Keyword Search in Relational Databases with Complex Schema. In: SIGMOD. pp. 905–908 (2014)
6. Kargar, M., An, A., Cercone, N., Godfrey, P., Szlichta, J., Yu, X.: Meaningful Keyword Search in Relational Databases with Large and Complex Schema. In: ICDE. pp. 411–422 (2015)