Towards a new Foundation for Keyword Search in Relational Databases

Riccardo Torlone
Università Roma Tre, Italy

1 Introduction

The idea of querying relational databases using keywords emerged a decade ago [4] as a way to provide an high-level access to data and free the user from the knowledge of query languages and data organization. The common approach to this problem is as follows: the database is viewed as a graph G in which the nodes represent tuples and the edges represent foreign key references between them, a query is a set of strings Q (the keywords), and the result is a subgraph G' of G whose nodes contain the keywords in Q. Usually it is assumed that: (i) all the keywords should appear in the result and (ii) the result should have minimal size. In this framework, query answering usually relies on rather complex, graph-based techniques.

We believe that this approach has a main drawback: it relies on the specific distribution of data in relational tables, which may depend on aspects that are not related to the actual content of the database. Indeed, today the degree of normalization is usually based not only on data redundancy, but also on the database workload, which has to do with the type and frequency of queries and updates. It follows that the result of a keyword query can change by just modifying the organization of the database (e.g., for optimization purposes) even if its actual content does not change.

We then propose a new approach to keyword search in relational databases that relies on the weak instance model [7, 6], an old yet still fascinating tool from relational database theory in which a database is considered as a whole, regardless of the way in which data are decomposed in the various relation schemes. In this model, we present a simple semantics for the result of a keyword query that is independent of the database schema and show that, by suitably extending a basic definition, we can introduce a relevance criterium among different results of a query. We also discuss the computational complexity of keyword search by means of a basic method for query answering.

2 Keyword search over weak instances

Let U be a finite set of attributes and $\mathbf{R} = \{R_1(X_1), \ldots, R_n(X_n)\}$ the schema of a relational database such that the union of the X_i's is U. We say that an instance $\mathbf{r}$ of $\mathbf{R}$ (globally) satisfies a set of functional dependencies (FDs) F if there is a relation w on U, called a *weak instance* for $\mathbf{r}$, that satisfies F and contains the relations of $\mathbf{r}$ in its projections over the respective relation schemes, that is: $\pi_{X_i}(w) \supseteq r_i$, for $1 \leq i \leq n$.

Let $T_{\mathbf{r}}$ for $\mathbf{r}$ be *tableau* formed by taking the union of all the relations in $\mathbf{r}$ extended to U by means of unique variables. The *representative instance* for $\mathbf{r}$, indicated with $RI_{\mathbf{r}}$, is the tableau obtained by chasing [5] $T_{\mathbf{r}}$ with respect to F.

Consider for instance a database scheme with relations $R_1(ED)$, $R_2(DF)$, $R_3(EP)$ and the functional dependencies $E \rightarrow D$, $D \rightarrow F$ as constraints. Figures 1 shows a database state on this scheme and the corresponding representative instance. It

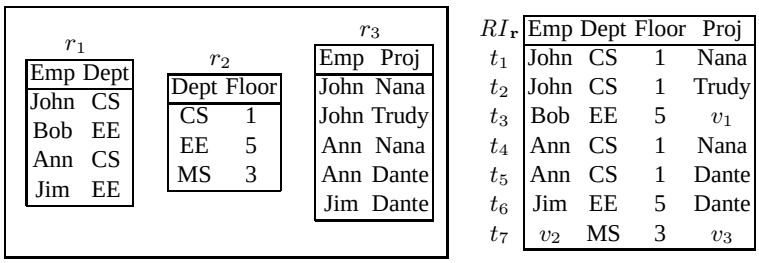

Fig. 1. A database state and its representative instance.

has been shown that a database state is consistent if and only if the corresponding representative instance can be built without encountering contradictions [3]. Also, for every consistent state $\mathbf{r}$ and for every X, the set of *total* tuples (i.e., without variables) in $RI_{\mathbf{r}}$ on X (called the *X-total projection* of $RI_{\mathbf{r}}$ and denoted by $\pi^{\downarrow}_X(RI_{\mathbf{r}})$) is equal to the set of tuples that appear in the projection on X of *every* weak instance of $\mathbf{r}$ [6]. According to this definition, $\pi^{\downarrow}_X(RI_{\mathbf{r}})$ is the relation over X *implied* by the current state.

We assume that a *keyword query* Q is simply a finite and non-empty set of constants. Given a tuple t over $X \subseteq U$, we say that: (i) t *covers* a set of constants C if, for each $c \in C$, $c = t[A]$ for some $A \in X$, and (ii) t *X-belongs* to database $\mathbf{r}$ if it belongs to the X-total projection of the representative instance of $\mathbf{r}$ (that is: $t \in \pi^{\downarrow}_X(RI_{\mathbf{r}})$).

Definition 1 (Base result). *A* base result *(also called* 1-result*) of a keyword query Q on a database $\mathbf{r}$ is a set of complete, total tuples R such that, for every tuple $t \in \mathsf{R}$: (i) t covers Q and (ii) t X-belongs to $\mathbf{r}$ for some $X \subseteq U$.*

For instance, a base result of the keyword query $\{CS, Nana\}$ over the database $\mathbf{r}$ in Figure 1 is composed by the tuples t_1 and t_4 of $RI_{\mathbf{r}}$.

Let us now now refine this notion by assuming that the keywords in the query can appear in different tuples of the representative instance that are connected through common values on common attributes. We say that a tableau T is *connected* if for each $t \in T$ there is another tuple $t' \in T$ that is joinable with t (that is, they share values on the same attributes) and that a set of total tuples T *covers* a set of constants C if each $c \in C$ appears in some tuple $t \in \mathsf{T}$.

Definition 2 (K-result). *A k-result of a keyword query Q on a database $\mathbf{r}$ is a minimal set of total tuples R^k such that: (i) R^k has size k, is connected, and covers Q, and (ii) every tuple $t \in \mathsf{R}^k$ x-belongs to $\mathbf{r}$ for some $X \in U$.*

For instance, the query $\{Nana, EE\}$ has one 3-result ($\mathsf{R}^3 = \{t_4, t_5, t_6\}$) and no 2 or 1-results. This shows that the parameter k captures the *relevance* of the result and then provides an effective tool to order (and possibly limit) the tuples to return to the users.

3 Computing the results of a keyword query

In the framework we have defined, the first question focuses on finding, possibly in an efficient way, the top-k results of a keyword query and the computational complexity of this problem. In this section we provide a preliminary result by discussing the Algorithm that follows, which implements a basic, "brute force" technique for solving this problem.

Algorithm 1: Computation of the top-k results of a keyword query

Input : A consistent database state $\mathbf{r}$, a keyword query Q, a limit $k > 0$
Output: The R^i-results of Q on $\mathbf{r}$ (for $1 \leq i \leq k$)

1 Build the representative instance T of $\mathbf{r}$;
2 **foreach** *tuple* t *in* T **do if** t covers Q **then output** t;
3 **for** $(i = 2; i \leq k; i++)$ **do**
4 **foreach** *tuple* t *in* T *that covers some* $c \in Q$ **do**
5 search and return the R^i-results including t with a depth-first visit of T from t;
6 remove t from T

The algorithm consists of three main steps: the construction of the representative instance (line 1), the search for the R^1 results (line 2), and the search for the subsequent R^k results, for $k > 1$ (lines 3-6). It is known that the first step requires polynomial time in the size of the database. In step 2 all the tuples of the representative instance are checked to verify if they (completely) cover the query and so it requires linear time in the size of $RI_{\mathbf{r}}$, which is proportional to $|\mathbf{r}|$. Finally, step 3 involves, for each tuple of $RI_{\mathbf{r}}$ that covers some keyword in the query, a depth-limited search in a graph G where the nodes represent the tuples and the edge represent the joinability relationship. In the worst case, the cost of this task is proportional to the maximum number of k-long paths in G, which is bounded by $|RI_{\mathbf{r}}|^k$. It is then possibile to show the following result.

Theorem 1. *Algorithm 1 computes, for some finite $k > 0$, all the first k-results of a keyword query Q of size q over a database state $\mathbf{r}$ of size n in time $O(n^q)$.*

Algorithm 1 can be optimized in several ways. In particular, the representative instance does not need to be built since, for significant classes of schemas, its total projection on a set of attributes can be computed efficiently by means of simple SPJ expressions [2]. Using these results, together with a suitable use of an inverted index, we could restrict our attention only to the relevant portion of the database. These issues and other extensions of the framework presented here will be subject of future studies.

References

1. A.V. Aho, C. Beeri, and J.D. Ullman. The theory of joins in relational databases. *ACM Trans. on Database Syst.*, 4(3):297–314, 1979.
2. P. Atzeni and E.P.F. Chan. Efficient and optimal query answering on independent schemes. *Theoretical Computer Science*, 77(3):291–308, 1990.
3. P. Honeyman. Testing satisfaction of functional dependencies. *Journal of the ACM*, 29(3):668–677, 1982.
4. V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.
5. D. Maier, **A.O. Mendelzon**, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. on Database Syst.*, 4(4):455–468, 1979.
6. D. Maier, J.D. Ullman, and M. Vardi. On the foundations of the universal relation model. *ACM Trans. on Database Syst.*, 9(2):283–308, 1984.
7. Y. Sagiv. A characterization of globally consistent databases and their correct access paths. *ACM Trans. on Database Syst.*, 8(2):266–286, 1983.