Towards a new Foundation for Keyword Search in Relational Databases

Riccardo Torlone

UNIVERSITÀ DEGLI STUDI ROMA TRE

AMW 2014

8th Alberto Mendelzon Int. Workshop on Foundations of Data Management – June 2-6 Cartagena, Colombia

Keyword search over relational data

- Input: a relational database

r

r_1

Emp	City	Dept
John	Rome	CS
Bob	Bogotà	EE
Ann	Paris	CS
Jim	Rome	MS

r_2

Dept	Head
CS	Mike
EE	Beth
MS	Rose

r_3

Emp	Proj
John	Nana
John	Trudy
Ann	Nana
Jim	Dante

r_4

Proj	Manager
Nana	Bill
Trudy	Rose
Dante	Mike

- Query: a set of keywords

$$\{Rome, Mike\}$$

- Result: a set of tuples involving the keywords

Emp	City	Dept	Head
John	Rome	CS	Mike

Emp	City	Dept	Proj	Manager
Jim	Rome	MS	Dante	Mike

Keyword search over relational data

- Goal:
  - high-level access to data
  - free the user from the knowledge of:
    - query languages
    - data organization
- A lot of work on the practical side
- Few theoretical studies
  - Kimelfeld and Sagiv, 2006.
  - Qin, Yu, Chang, 2009.

Traditional approach

r_1

Emp	City	Dept
John	Rome	CS
Jim	Rome	MS

r_2

Dept	Head
CS	Mike

r_3

Emp	Proj
John	Nana
Jim	Dante

r_4

Proj	Manager
Nana	Bill
Dante	Mike

$\{Rome, Mike\}$

1

Emp	City	Dept	Head
John	Rome	CS	Mike

2

Emp	City	Dept	Proj	Manager
Jim	Rome	MS	Dante	Mike

Let us change the schema

r_2

Dept	Head
CS	Mike

r_1

Emp	City	Dept
John	Rome	CS
Jim	Rome	MS

$\{Rome, Mike\}$

r_3

Emp	Proj	Manager
John	Nana	Bill
Jim	Dante	Mike

1

Emp	City	Dept	Head
John	Rome	CS	Mike

1

Emp	City	Dept	Proj	Manager
Jim	Rome	MS	Dante	Mike

Let us change the schema

r_2

Dept	Head
CS	Mike

r_1

Emp	City	Dept	Proj	Manager
John	Rome	CS	Nana	Bill
Jim	Rome	MS	Dante	Mike

$\{Rome, Mike\}$

1

Emp	City	Dept	Proj	Manager
Jim	Rome	MS	Dante	Mike

2

Emp	City	Dept	Proj	Manager	Head
John	Rome	CS	Nana	Bill	Mike

Another problem

- Database

r_1

Emp	City	Dept
John	Rome	CS
Bob	Bogotà	EE
Ann	Paris	CS
Jim	Rome	MS

- Query

$$\{John, Ann\}$$

- Result

Emp	City	Dept
John	Rome	CS
Ann	Paris	CS

Limitations of the traditional approach

- It depends on the specific distribution of data in relational tables
- 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.
- Some meaningful results are not captured

Our proposal

- A new framework for investigating, in a systematic way, the problem of keyword search in relational databases
- The framework is based on the **weak instance model**, an old tool from relational database theory
- In the weak instance model a database is considered as a whole, regardless of the way in which data are decomposed in the various relation schemes.

The weak instance model

- U: a finite set of attributes U
- F: a set of FDs over U
- **R** = $\{R_1(X_1), \ldots, R_n(X_n)\}$: a relational database schema such that $\bigcup_{i=1,..,n} X_i = U$.
- An instance **r** of **R is** (globally) consistent if there is a relation w on U, called a *weak instance* for r, that satisfies F and contains the relations of **r** in its projections over the respective relation schemes, that is: $\pi_{Xi}(w) \supseteq r_i$, for $1 \leq i \leq n$

The representative instance

- The definition of global satisfaction is not practical
- A special relation on U that can be built to test for the existence of a weak instance
- **State tableau** T_r for a relational database **r**: formed by taking the union of all the relations in **r** extended to U by means of unique variables.
- **Representative instance** RI_r for **r**: the tableau obtained by chasing T_r using the given FDs

An example

r

r_1

Emp	Dept
John	CS
Bob	EE
Ann	CS
Jim	EE

r_2

Dept	Floor
CS	1
EE	5
MS	3

r_3

Emp	Proj
John	Nana
John	Trudy
Ann	Nana
Ann	Dante
Jim	Dante

$$Emp \rightarrow Dept, \quad Dept \rightarrow Floor$$

$T_{\mathbf{r}}$

Emp	Dept	Floor	Proj
John	CS	v_1	v_2
Bob	EE	v_3	v_4
Ann	CS	v_5	v_6
Jim	EE	v_7	v_8
v_9	CS	1	v_{10}
v_{11}	EE	5	v_{12}
v_{13}	MS	3	v_{14}
John	v_{15}	v_{16}	Nana
John	v_{17}	v_{18}	Trudy
Ann	v_{19}	v_{20}	Nana
Ann	v_{21}	v_{22}	Dante
Jim	v_{23}	v_{24}	Dante

Two chase steps

T_r

Emp	Dept	Floor	Proj
John	CS	v_1	v_2
Bob	EE	v_3	v_4
Ann	CS	v_5	v_6
Jim	EE	v_7	v_8
v_9	CS	1	v_{10}
v_{11}	EE	5	v_{12}
v_{13}	MS	3	v_{14}
John	v_{15}	v_{16}	Nana
John	v_{17}	v_{18}	Trudy
Ann	v_{19}	v_{20}	Nana
Ann	v_{21}	v_{22}	Dante
Jim	v_{23}	v_{24}	Dante

Emp	Dept	Floor	Proj
John	CS	v_1	v_2
Bob	EE	v_3	v_4
Ann	CS	v_5	v_6
Jim	EE	v_7	v_8
v_9	CS	1	v_{10}
v_{11}	EE	5	v_{12}
v_{13}	MS	3	v_{14}
John	CS	v_{16}	Nana
John	v_{17}	v_{18}	Trudy
Ann	v_{19}	v_{20}	Nana
Ann	v_{21}	v_{22}	Dante
Jim	v_{23}	v_{24}	Dante

$Emp \rightarrow Dept$

Emp	Dept	Floor	Proj
John	CS	v_1	v_2
Bob	EE	v_3	v_4
Ann	CS	v_5	v_6
Jim	EE	v_7	v_8
v_9	CS	1	v_{10}
v_{11}	EE	5	v_{12}
v_{13}	MS	3	v_{14}
John	CS	1	Nana
John	v_{17}	v_{18}	Trudy
Ann	v_{19}	v_{20}	Nana
Ann	v_{21}	v_{22}	Dante
Jim	v_{23}	v_{24}	Dante

$Dept \rightarrow Floor$

The final representative instance

RI_{r}	Emp	Dept	Floor	Proj
t_1	John	CS	1	Nana
t_2	John	CS	1	Trudy
t_3	Bob	EE	5	v_1
t_4	Ann	CS	1	Nana
t_5	Ann	CS	1	Dante
t_6	Jim	EE	5	Dante
t_7	v_2	MS	3	v_3

Properties of the representative instance

- A database state is consistent if and only if the corresponding representative instance can be built without encountering contradictions [Honeyman, 1982].
- For every consistent state r and for every X, the set of total tuples in RI_r on X (called the X-*total projection* of RI_r) is equal to the set of tuples that appear in the projection on X of *every* weak instance of r [Maier, Ullman, and Vardi, 1984].
- Total projection of the representative instance = the relation over X *implied* by the current state.
- Different databases with the same representative instance
  - have the same set of weak instances
  - represent the same information

Keyword search over weak instances

- Given a tuple t over X ⊆ U,
  - t *covers* a set of constants C if, for each c in C, there is an attribute A in X such that t[A] = c,
  - T *x-belongs* to database **r** (in symbols t∈r) if t belongs to the X-total projection of the representative instance of **r**.
- **Definition 1.** A **base result** of a keyword query Q on a database **r** is a set of total tuples R such that, for every tuple t in R:
  - t covers Q and
  - t x-belongs to **r** for some X ⊆ U.

An example

$\mathbf{r}$

r_1

Emp	Dept
John	CS
Bob	EE
Ann	CS
Jim	EE

r_2

Dept	Floor
CS	1
EE	5
MS	3

r_3

Emp	Proj
John	Nana
John	Trudy
Ann	Nana
Ann	Dante
Jim	Dante

$RI_{\mathbf{r}}$

	Emp	Dept	Floor	Proj
t_1	John	CS	1	Nana
t_2	John	CS	1	Trudy
t_3	Bob	EE	5	v_1
t_4	Ann	CS	1	Nana
t_5	Ann	CS	1	Dante
t_6	Jim	EE	5	Dante
t_7	v_2	MS	3	v_3

$$Emp \rightarrow Dept, \quad Dept \rightarrow Floor$$

$$\{CS, Nana\}$$

Emp	Dept	Floor	Proj
John	CS	1	Nana
Ann	CS	1	Nana

A refinement of query result

- A base result correlates values only on the basis of the functional dependencies.
- Other meaningful relationships can be established between values in the database.

K-result

- Two tuples t_1 and t_2 are *joinable* if $t_1[A] = t_2[A]$ for some A in U
- A tableau T is *k-connected* if there is an enumeration $t_1, \ldots, t_k$ of the k tuples in T such that t_i is joinable with t_j, for $1 \leq j < i \leq k$.
- A set of total tuplesT covers a set of constants C if, for each c in C, there is a tuple t in T that covers $\{c\}$.
- **Definition**. A **k-result** of a keyword query Q over a database **r** is a minimal set of total tuples R_k such that:
  - R_k covers Q,
  - every tuple t in R_k x-belongs to **r** for some $X \in U$, and
  - the tableau R_k^* is k-connected.

An example

r

r_1

Emp	Dept
John	CS
Bob	EE
Ann	CS
Jim	EE

r_2

Dept	Floor
CS	1
EE	5
MS	3

r_3

Emp	Proj
John	Nana
John	Trudy
Ann	Nana
Ann	Dante
Jim	Dante

$RI_{\mathbf{r}}$	Emp	Dept	Floor	Proj
t_1	John	CS	1	Nana
t_2	John	CS	1	Trudy
t_3	Bob	EE	5	v_1
t_4	Ann	CS	1	Nana
t_5	Ann	CS	1	Dante
t_6	Jim	EE	5	Dante
t_7	v_2	MS	3	v_3

$$Emp \rightarrow Dept, \quad Dept \rightarrow Floor$$

$$\{Trudy, Dante\}$$

Emp	Dept	Floor	Proj
John	CS	1	Trudy
Ann	CS	1	Dante

Another example

$\mathbf{r}$

r_1

Emp	Dept
John	CS
Bob	EE
Ann	CS
Jim	EE

r_2

Dept	Floor
CS	1
EE	5
MS	3

r_3

Emp	Proj
John	Nana
John	Trudy
Ann	Nana
Ann	Dante
Jim	Dante

$RI_{\mathbf{r}}$	Emp	Dept	Floor	Proj
t_1	John	CS	1	Nana
t_2	John	CS	1	Trudy
t_3	Bob	EE	5	v_1
t_4	Ann	CS	1	Nana
t_5	Ann	CS	1	Dante
t_6	Jim	EE	5	Dante
t_7	v_2	MS	3	v_3

$Emp \rightarrow Dept, \quad Dept \rightarrow Floor$

$$\{Nana, EE\}$$

Emp	Dept	Floor	Proj
Ann	CS	1	Nana
Ann	CS	1	Dante
Jim	EE	5	Dante

Computing the k-results

- A brute force algorithm

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

- three main steps:
  1. the construction of the representative instance (line 1),
  2. the search for the R_1 results (lines 2),
  3. the recursive search for the subsequent R_k results (lines 3-6)

Complexity

- Step 1: polynomial time in the size of the database.
- Step 2: linear time in the size of RIr, which is proportional to $|r|$.
- Step 3: 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 $|RI_r|^k$.
- **Result.** *Algorithm 1 computes, for some finite* $k > 0$, *all the first* k-*results of a fixed keyword query* Q *over a database state* r *of size* n *in time* $O(n^k)$.

Possible optimizations

- The representative instance does not need to be completely computed.
- For significant classes of schemas, the total projection of the representative instance can be computed efficiently by means of simple SPJ expressions [Sagiv 1983].
- The search for joinable tuples can be made efficient by exploiting indexes and by adopting backtracking strategies.

Conclusions

- We have proposed a formal framework for the investigating the problem of querying a relational database with a set of keywords.
- The approach is based on the weak instance model, which provides a view of a relational database that is independent of its specific organization in a set of relations.
- We have shown that, in this model
  - the problem can be expressed in a simple and natural way
  - rhe computation of the first top-k results remains tractable
- Future work
  - Refinment of the model
  - More efficient techniques for computing the k-results
  - Investigation in this model of other issues related to keyword-search