Denis Mindolin
April 26, 2008
CSE718 – Seminar on Database Interfaces

iTrails: Pay-as-you-go Information Integration in Dataspaces

Overview

In their work, the authors consider a novel model of heterogeneous data organization called *dataspaces*. It is aimed to improve the two most popular approaches used in data integration: the *schema first approach (SFA)* and the *no schema approach (NSA)*. The SFA approach requires semantically integrated view over integrated data sources. Query languages generally have formal semantics, however maintaining the mediated schema here is generally costly. The NSA approach requires no semantically integrated view and considers sources as XML-like documents. Since no mediated schema is present, no semantic mappings between schemas are required. At the same time, query languages here a limited to XPath-like languages and keyword search. The model considered by the authors is in between there two approaches which gives it advantages (and some disadvantages) of both. It starts with NSA and gradually approaches SFA in a pay-as-you-go manner by providing semantic hints (or, *trails*) to query engine.

A *trail* is represented by a pair of queries $Q_L \rightarrow Q_R$. The essence of the query evaluation process with trails can be expressed by the sentence: *whenever you query for Q_L, also query for Q_R*. In order to apply this principle, the authors propose a query rewriting algorithm consisting of the following three steps: *matching*, *transformation*, and *merging*. Given a user query and a trail, the query engine performs the following. In the *matching* phase, it finds maximal parts Q_M of the query that contain Q_L. In the *transformation* phase, for every instance of Q_M, it constructs an instance Q_R' of Q_R (this step can be non trivial if Q_M involves a component projection). In the *merging* phase, it rewrites the query by replacing all instances of Q_M in the original query with $Q_M \cup Q_R$'.

Being a straightforward in the presence of a single trail, the query rewriting process becomes problematic when there are more than one trail is defined. Namely, the transformed parts Q_R of the modified query after one step of the algorithm can match other trails and we have to reapply the algorithm once again. Thus, if trails induce cycles (e.g. $Q1 \rightarrow Q2 \rightarrow \ldots \rightarrow Q1$), the rewriting process can be infinite. To tackle this problem, the authors propose an algorithm which forbids reapplication of a trail to all query parts which are results of application of the same trail before.

Thus, the algorithm eventually terminates. However, the possible number of trail applications is exponential in the number of trails in general. This is obviously a problem if the number of trails is large.

In order to improve the runtime of the algorithm, the authors propose a number of techniques to prune the number of trail applications. Namely, instead of taking in each step of the algorithm all possible trails which match the query; one can pick only some of them. To constructs such a subset, all trails have scores associated with them, and the subset will contain only the top *k* trails based on their score value. Another optimization approach is to limit the number of query rewriting steps by a parameter passed to the algorithm from outside.

Detailed comments

The major strength of the paper is the incremental nature of the proposed integration method. Assume that a company deploys a query engine which uses a large number of various data sources. Then, instead of defining all hints in advance, the integrators can collect user query statistics, find the most common queries and define trails for the most popular queries first. Moreover, if the set of the most popular queries changes over time, the trails for less popular queries can be disabled to reduce the system workload. Another advantage of this approach is that trails can be defined independently by domain expects for each data domain. Overlaps and inconsistencies in trails are possible since the query returns the union of the results satisfying all trails. However, such universality comes with disadvantages. First, trails are global and thus every rewritten query is evaluated over every data source. However, the same trail can have different meaning for different data sources. Second, it seems that to obtain the reasonable quality of query results, the trails have to be defined manually which must be a problem for large systems. Thus, a good system of trail mining and weighting would be helpful here.

The paper provides a large number of technical details of the proposed approach. This makes it easy to understand even for readers with little expertise in the area since most concepts used in the paper are intuitive. Moreover, the definitions and algorithms introduced in the paper are illustrated with a good number of examples.

The paper is technically sound. The definitions provided in the paper are clear and complete. All the new results provided in the paper are accompanied with proofs. The experimental results do not contradict to the strengths and weaknesses of the proposed approach.