An Architecture for Building Federated Databases by Example

François G. Barbançon, Daniel Miranker

November 14, 2001

Abstract

We define an architecture for an easy to use graphical system to query data from multiple heterogeneous sources. The contemporary development of computer networks and relational database systems immediately revealed the difficulties of integrating data from multiple heterogeneous sources. After nearly 30 years of extensive research efforts and hundreds of commercially developed tools, this integration remains a complex and labor intensive tasks.

In an informal proof [KLK91], Krishnamurthy, Litwin and Kent demonstrated that merging data from multiple sources requires higher order syntactic constructs to overcome schematic inconsistencies across data sources. They deduce that first order relational languages such as SQL or Datalog can be extended with a higher order syntax. Then become expressive enough to describe the necessary class of transformation.

We put forth an architecture which directly contemplates the issues of data integration with respect to the precise nature of the schematic inconsistencies that must resolved. The primary focus is on designing a system in a way that a naive database user may specify and effect a solution.

1 Introduction

Federated database architectures are concerned with integrating heterogeneous components and transparently providing database services around a federated database. We leave that broad perspective aside and focus on the subproblem of specifying data federating transformations. The goal of this architecture is to provide a framework for an interface allowing complete semantic specification of federating queries. We surmise that with existing or proposed data federation solutions, defining the mappings from heterogeneous sources to a set of federated views remains a labor intensive task. Moreover because of the inherent complexities and cost of using existing tools, the task is only accessible to a small group of highly trained professionals. This observation remains fully valid despite the emergence of XML and its associated tools and languages.

Graphical interfaces (GUIs), built to specify relational queries, are available with all the major commercial databases. These are designed for desktop users and simplify specification of a large class of relational queries. The graphic input available from a GUI system is insufficient however to fully specify a data federating query as they exist in federated database systems. The specification from graphic input is incomplete. Some of the missing data can be interpolated, and the rest has to be extrapolated.

The approach suggested by our architecture has a lot in common with an expert system. We propose a system capable of exploiting information from graphic input, drawing from data mining of the data sources, and asking the user relevant questions to fully specify a federating query. The spirit of this approach is for the system to assist the user in taking away complex or repetitive tasks. In counterpart, the user will provide the system some basic knowledge about the data, which would be extremely difficult to hypothesize correctly.

This architecture has three layers which parallel the process followed by an engineer building an ad-hoc data federation. The first step is to recognize the relationships which exist between data in different component databases. In effect this is analyzing and comparing the design of the component databases. This work is done in the first layer by closely examining the data and the meta-data. This layer draws on data mining tools to construct relationship diagrams from existing data. These relationships are intricately linked with the nature of the data transformations that will be necessary or desirable when federating data.

The second layer is an interactive learning system incorporating a user interface. It allows the user to specify data federating queries. This learning system draws on the knowledge acquired by the data mining layer to build a search space for its learning algorithm. The learning system incorporates sample selection and active learning methods in order to help the user specify the desired query as efficiently and unambiguously as possible.

The last layer is an execution engine. It executes the federating query and is capable of materializing the federated view by drawing on the data sources. Taken together all the layers of the architecture could enable us to build the functional equivalent of a 'mediator' component. This is not however the purpose of this architecture. Rather, the execution engine is a necessary part of an interactive learning system capable of differentiating output examples from competing hypotheses in order to solicit query specification.

2 Motivation

2.1 Querying Multiple Databases

With the advent of the internet, every database in the world is accessible from your own computer. This availability is pushing the demand for real-time connectivity standards. Querying multiple databases simultaneously has become a reality. Given a set of standard interfaces such as ODBC or JDBC, it becomes possible to express queries that aggregate content ranging over a set of component databases.

Building a federated view over data sources must be done by engineers drawing on specialized tools and libraries. Tools are necessary to bridge semantic schema differences. Commercial solutions to this challenge typically range from using DOM, XSLT, or JDBC programming. Unfortunately the use of those tools is time consuming and requires significant programming skills. Thus a paradoxical situation has been reached. The cost of maintaining networked data sources has never been lower, thanks to the explosion of database capacities and of networks. Meanwhile, the cost of bridging heterogeneity in data sources has remained comparatively high. As satisfying solutions to this problem have yet to be found, dozens, perhaps hundreds of commercial or scientific solutions are constantly trying to fill that gap.

2.2 Why Building Queries over Multiple Databases is Hard

In 1991, Krishnamurthy, Litwin and Kent demonstrated through a real life example, that a relational language with higher order syntax was a requirement to federate heterogeneous databases.

Krishnamurthy's convincing argument is that a query defining data federation almost always needs to incorporate variables ranging over schema elements: attribute names, table names, etc.. This requires a syntax that can do such manipulations: a higher order syntax.

Higher order languages have been implemented. One instance is an extension of SQL named SchemaSQL ([LSS96]), contains the syntactic elements shown in Figure 1. The syntax is almost identical to SQL except for the placement of variables where SQL would allow only constants, such as