A mediation framework

for a transparent access to biological data sources[1]

The MediaGRID project

Contact: Christine.Collet@imag.fr

Abstract: This paper introduces the main aspects of the MediaGRID project (http://www-lsr.imag.fr/mediagrid) whose objective is to contribute to the definition of a mediation framework for the Grid. « Framework » means a reusable design (of a mediation system) expressed as a set of abstract classes (or components) and they collaborate. The main research topics include the generation of mediation queries and the evaluation of queries in a distributed, adaptive and interactive way.

1 Introduction

The increasing use of computers and the development of communication infrastructures have led to a wide range of information sources available through networks. Data integration systems or mediation systems have been proposed to provide a transparent and efficient access to these multiple heterogeneous, distributed and autonomous sources [DD99, Wie92]. Such systems give users and applications the illusion of dealing with a unique data source whereby handling heterogeneity of underlying data source managers, operating systems and networks.

The general architecture of a mediation system is composed of one or several wrappers and mediators. Wrappers are used to transform data from sources into a mediation common format, according to exported schemas. Mediators support the large virtual unique source that is described as a set of mediation schemas. At the mediation level, the operational mappings between the exported schemas and the mediation one are specified by mediation queries. They express the way data of the mediation level are computed. This task of building mediation queries is often executed by a human operator. When querying data, mediation queries are used as input of the (unfolding) algorithm to rewrite queries into subqueries which are executed at local sources. Returned results of sources are combined and sent back to applications.

[1] This work is supported by the French Ministry of Research through the ACI-GRID program. Partners of the project are: the LSR-IMAG Laboratory – Grenoble University, the PRiSM Laboratory – Versailles University and the LaMI Laboratory – University of Evry-Val-d'Essone.

The objective of the MediaGRID project (http://www-lsr.imag.fr/mediagrid) is to contribute to the definition of a mediation framework for the Grid. Mediation systems built from our framework would be able to (i) support more and more available sources, (ii) consider sources containing weakly structured data, (iii) authorize partial results for queries in case of data sources unavailability and/or satisfy user interests, (iv) support a query evaluator able to adapt itself to the execution environment and to allow dynamic user control. Research topics include two main aspects: the generation of mediation queries and the evaluation of queries in a distributed, adaptive and interactive way. The following presents these aspects and give some indications on the way we plan to validate our proposal in a biological context.

2 Mediation queries generation

Mediation queries are often supposed to be generated manually, which is a very complex process, considering the amount of required knowledge: the designer has not only to know the content of all the sources, but also the semantic links between the sources and the mediation schema. The complexity of this task increases with the number of data sources. An early answer has been proposed in [KBo99], followed by a valuable result from the Clio project [YMH+01].

Our goal is, given the descriptions of the mediation schema and the data sources, to generate the queries that will populate instances of this schema in terms of the data sources. Exported data source schemas and mediation schemas are described using XML. Our approach comprises three main steps: (i) identifying the relevant portions of data sources, (ii) identifying the candidate operations between data sources and (iii) generating mediation queries.

Identifying relevant portions of data sources. For each local schema the first step identifies the relevant portion, called a mapping schema, with respect to the mediation schema. To produce such a mapping schema, we consider that some metadata is available, consisting mainly in a set of semantic correspondences defined between elements of local schemas and the mediation schema. A mapping schema is composed of elements of mainly the local schema involved in such correspondences and other elements, such as the keys and the references defined in the schema. The result of this step is, for each local schema, a mapping schema and a query allowing deriving these instances of this schema from the corresponding source.

Identifying candidate operations. Once all the mapping schemas are defined, the next step consists in searching for some candidate operations (especially joins) between mapping schemas. For each pair of mapping schemas, join operations are determined using the semantic correspondences existing between the schemas and their key elements; given a pair of mapping schemas, there might be several ways to join between them. The set of candidate operations can be represented by an operations graph, where each node represents a mapping schema, and each edge between two nodes represents a candidate join operation between the mapping schemas represented by these nodes.

Generating mediation queries. The operations graph is then used to generate mediation queries; for this purpose, computation paths are identified on the graph: a computation path is either a mapping schema containing all the elements of the mediation schema, or an acyclic and connected sub-graph involving all the elements of the mediation schema. For each computation path, a mediation query is searched for. The result of this step is a set of Xquery mediation queries, each of them corresponding to a specific semantic.

3 Adaptive and Iterative query processing

To provide an adaptive and iterative query evaluator, one can benefit from techniques proposed in distributed and parallel database management systems and also from adaptive and interactive query processing techniques [HFC+00]. We provide a framework called QBF (Query Broker Framework) to integrate in a uniform way all these existing optimization and execution mechanisms.

The internal representation of a query in QBF is a standardized, canonical query graph, so-called a query plan. It involves operators such as Select, Project, Join, Union, XOR. These operators consume and produce sequences of items (tuples, entities, or objects). A query also has a context that determines constraints to be checked when processing the query. Constraints may concern users, system resources or the underlying network. For example, users may wish to limit number of results, execution time or have preference on data, accept partial results, etc. Components of our core evaluation query system are: the classical plan manager and an execution engine, a monitor and a rule manager providing an adaptive evaluation authorizing partial results, and a component for interacting with the user.

Programmers can build their own query evaluation system, called a Query Broker, either implementing the QBF abstract classes or using our QBF implemented classes. Classes can then be extended (defining sub-classes) and/or used to create instances (e.g. specific rules, transforming patterns). Query brokers provide an adaptive query evaluation, a capability to build partial result and to authorize interaction during query evaluation. The following gives an overview of these capabilities.

Adaptive Evaluation. Adaptive query evaluation is done thru the Monitor and RuleManager abstract classes. Different monitors have been defined as classes to respectively monitor arrival data rate, number of data processed, and execution time. At query processing time, instances of theses classes are used to observe a query execution and signal significant changes happening in the query environment. These changes launch rules so as adapting query execution to changes. Different techniques for adaptive evaluation such as the ones in [KD98, AFT+96] have been integrated as rules. Besides, adaptive evaluation can also be ensured by using adaptive operators such as XJoin [UF00], ripple-join [HH99], etc. We can integrate both techniques in query brokers, coding the corresponding rules and operators. The same approach is adopted to authorize partial results (as done in [STD+00]) by controlling the components, considering specific operators (Dummy) and specific adaptation rules [VC03]. Partial results are returned by using techniques for direct and/or redirect data flows between query operators.

Interaction. Looking at the first (incomplete) results, users can refine their long running queries. Users can modify their ongoing query (both of query context and query core), request partial results. User interaction is handled in two phases. The first one aims at preparing query evaluation for this change. It detects modifications needed at query operators and monitoring properties parameters. No new input data is accessed but the system can continue to return results with data in processing. This phase aims also at maintaining the coherence of data processing. The second phase directs and/or redirects data flow between query operators such as minimizing the query plan updates.

4 Application

Our framework will be validated within a biological context. It will be applied on sources giving information related to genes cartography and expression. The objective is to provide biologists with means to correlate expression levels of a gene whose data are stored within different sources and observe their evolution. Performing such a task requires first selecting relevant data sources and then discovering correlations among them thereby being able to integrate data they give. During the processing, partial results have to be supported and given to biologists progressively so that they can intervene.

References

[AFT+96] Laurent Amsaleg, Michael J. Franklin, Anthony Tomasic, and Tolga Urhan. Scrambling Query Plans to cope with Unexpected Delays. In Proc. of PDIS, 1996.

[DD99] Ruxandra Domenig and Klaus R. Dittrich. An Overview and Classification of Mediated Query Systems. In Sigmod Record, September, 1999.

[HH99] Peter J. Haas and Joseph M. Hellerstein. Ripple Joins for Online Aggregation. In Proc. of SIGMOD, 1999.

[HFC+00] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman, Mehul A. Shah. Adaptive Query Processing: Technology in Evolution. In Bulletin of the Technical Committee on Data Engineering, 2000.

[KBo99] Z. Kedad and M. Bouzeghoub, Discovering View Expressions from a Multi-Source Information System, in Proc. of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIS), Edinburgh, Scotland, pp. 57-68, Sep. 1999.

[KD98] N. Kabra and D. J. DeWitt. Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans. In Proc. of ACM SIGMOD, 1998.

[STD+00] Jayavel Shanmugasundaram, Kristin Tufte, David J. DeWitt, Jeffrey F. Naughton, and David Maier. Architecting a Network Query Engine for Producing Partial Results. In (Informal) Proc. of WebDB, 2000.

[UF00] Tolga Urhan and Michael J. Franklin. XJoin: A Reactively-Scheduled Pipelined Join Operator. IEEE Data Engineering Bulletin, 23(2):27-33, 2000.

[VC03] Tuyet-Trinh Vu and Christine Collet. Query Brokers for Distributed and Flexible Query Evaluation. In Proc. of the Conference RIVF, Hanoi, Vietnam, 2003.

[YMH+01] L.L. Yan, R.J. Miller, L.M. Haas, R. Fagin: Data-Driven Understanding and Refinement of Schema Mappings. SIGMOD Conference 2001.

[Wie92] Gio Wiederhold. ediator in the Achitecture of Future Information Systems. The IEEE Computer Magazine, 1992.