Towards a Mediation System Framework for Transparent Access to Largely Distributed Sources

The MediaGrid project[1]

Christine Collet*, Khalid Belhajjame, Gilles Bernot, Christophe Bobineau, Gennaro Bruno, Beatrice Finance, Fabrice Jouanot, Zoubida Kedad, David Laurent, Fariza Tahi, Genoveva Vargas-Solar, Tuyet-Trinh Vu, Xiaohui Xue

*LSR-IMAG Lab., Institut National Polytechnique de Grenoble
BP 72, 38402 Saint-Martin d'Hères, France
Christine.Collet@imag.fr

Abstract. This paper presents the MediaGrid project whose goal is the definition of a mediation framework for transparent access to largely distributed sources. *Frameworks* are reusable pieces of design being expressed as a set of interfaces and components together with the description of their collaboration. Research topics addressed by the MediaGrid project include meta-data design, generation of mediation queries and adaptive and interactive query evaluation.

1 Introduction

The increasing use of computers and the development of communication infrastructures have led to a wide range of information sources being available through networks. Data integration systems or mediation systems have been proposed as a solution to provide a transparent and efficient access to multiple heterogeneous, distributed and autonomous sources [DD99,Wie92]. These systems handle underlying data source managers, operating systems and networks heterogeneity thereby giving users and applications the illusion that they deal with a unique data source.

The complexity of mediation systems increases with respect to the number, the types and the capacities of data sources. Moreover, huge amounts of knowledge (source descriptions, schemas, semantic relations between schemas) have to be maintained. This increases the difficulty to design assertions between sources. Future mediation systems would be highly dynamical: hence they will have to manage data sources evolution and the adding or removal of sources. Source availability should be considered by the query processing, where queries may need to

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

dynamically change their execution plan, produce partial results or materialize results. Finally, users and applications want to control the query processing.

The MediaGrid project (*http://www-lsr.imag.fr/mediagrid*) takes up these challenges. The objective is to propose a mediation framework, i.e. a reusable design (of a mediation system) expressed as a set of interfaces (or components) and the way their instances collaborate. MediaGrid mediation systems built from the framework are 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 which is able to dynamically adapt itself to the execution environment and which accepts user interaction during query execution.

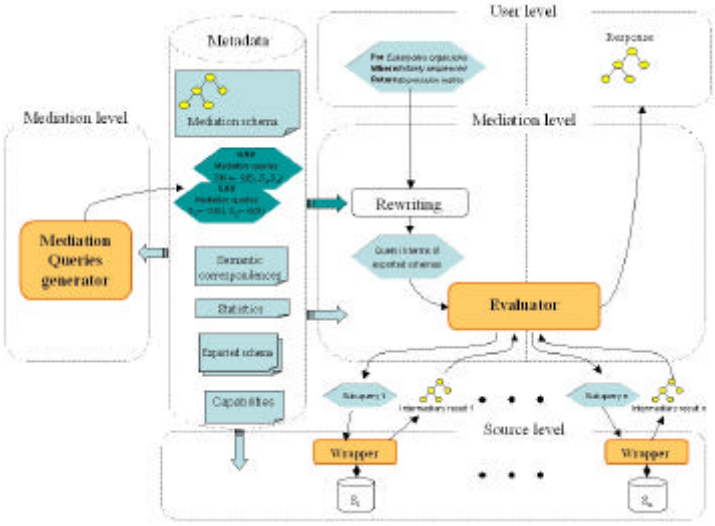

Fig. 1 – General architecture of MediaGrid mediation system

The general architecture of a MediaGrid system is given in Figure 1. It follows the classical three-layer architecture. Users or applications access data contained in local sources through the mediation layer. Queries are formulated over a mediation schema (global schema) and are rewritten in terms of exported schemas. Both kinds of schemas are defined using a XML syntax. The mediation (or global) schema describes integrated data manipulated at the mediation level. The exported schemas result from the subscription of sources to the system. During this process, sources are wrapped and data descriptions (schemas, DTD, types, etc.) are translated as exported schemas. Advanced information such as source capabilities and some statistics on sources are also extracted.

Operational mappings between the exported schemas and the mediation one are specified using mediation queries. Such queries are often supposed to be generated manually, which is generally a very complex process considering the amount of knowledge to take into account. Indeed, besides knowing the content of all the sources, the designer has to know semantic links between the sources and the mediation schema (e.g., functional dependencies, referential constraints and value constraints, domains compatibility, semantic equivalence between attributes and instances of key attributes, etc.).

The complexity of this task increases with the number of data sources. A first answer has been proposed in [KBo99], followed by a valuable result from the Clio project [YMH+01]. The Mediation Queries generator automates this process.

Meta-information also plays a very important role in query processing. When querying data, mediation queries are used as input of the (unfolding) algorithm to rewrite user queries into sub-queries executed at local sources. Mediation queries have therefore to be considered as meta-data. Queries are rewritten in terms of exported schemas and evaluated by the Evaluator component. It is important that this component takes into account sources capabilities to avoid a huge data transfer over the network by delegating some tasks to the sources. Returned results from sources are then combined and sent back to applications or users.

Query evaluators may use complicated techniques resolving problems related to network delays, lack of memory, etc. Moreover, applications (or users) may have different requirements for processing data such as source preference, time limit for query evaluation, number of results being handled by an application, economic cost limit for accessing data in case of paying sources, etc. Some of them can wish to get results in brief delay even if they are not complete while others need complete and exact results. Different mechanisms [SAC+79,GM93,KD98,BFMV00,UF00,AH00] have been proposed to respond to one or several of these requirements. However, such mechanisms have been designed and implemented for systems having specific characteristics. It is difficult to have them working together in an efficient manner within a mediation layer. Therefore, we propose to give programmers of a MediaGrid system some tools to build an evaluator providing the « exact » querying capabilities for the applications requirements. The evaluator is built from a Query Broker Framework (QBF) integrating several mechanisms proposed in distributed and parallel database management systems and coming from adaptive and interactive query processing techniques [HFC+00]. This innovative approach offers different adaptation techniques which can be used over different data models (relational, semi-structured or object), in an uniform way, to reach application (or user) requirements, even if these techniques have not been originally designed for those data models.

To illustrate these aspects we consider a mediation system that allows biologists with a means to correlate expression levels of a gene -- whose data are stored within the three data sources GOLD [BEK01], SMD [SHK+01] and SGD [BDD+00] sources -- and to observe their evolution. An example of a query defined at the application level is “looks for the organisms completely published and eukaryote”.

Outline of the Paper - The remaining of this paper focuses on the main contributions of the project. Section 2 discusses the meta-data supported by MediaGrid systems. Section 3 introduces the main steps of its query generation process and Section 4 describes the QBF approach to provide adaptive and interactive query evaluators. Finally, Section 5 concludes and gives some information on the current status of the project and the way we validate our approach.

2. Metadata management

Meta-data are defined to support the generation of mediation queries and the evaluation of global queries. They describe mediation and exported schemas, mediation queries, semantic correspondences, source capabilities and statistics.

2.1 Schema representation

Mediation and exported schemas in MediaGrid are represented using the XML model. The meta-representation of a XML schema is based on a graph modeled as a set of nodes. Thus, using meta-representation concepts a Schema is modeled as a collection of Node classes that can be linked by a relationship of type path. The Node class can be specialized into ExportedNode that represents an exported schema node, and MediationNode that represents a mediation schema node. Each of them has the Node structure presented in Figure 2.

Fig. 2 – XML Schema Node representation

A Node in a Schema has an identifier (id) and a name type (see Figure 2). A node is of type TextNode or NonTextNode. The TextNode class describes text nodes (i.e. integer, string) and it can be specialized into Attribute and TextElement. Constraints such as primary and foreign keys are also represented. The class Key represents primary keys. A key has an identifier (id), a type and a name[2]. It can be of type ID, Key or Unique. A foreign key is characterized by an identifier, a name and a type (IDREF or KeyRef). Relationships isKeyNode and isKeyRefNode represent the association of a key to the set of nodes that compose it. A primary key can be used either for identifying a NonTextNode or a TextElement. The relationship identifies is used for repre-

[2] This name represents the name of the primary key constraint and does not correspond to the name of the node playing the role of primary key.

senting this situation. Finally, in a XML schema, the key scope may be limited to a part of a XML document. The relationship scope links a key to the NonTextNode representing its scope.

2.2 Semantic correspondences

Exported schema nodes are semantically linked with mediation schema nodes through semantic correspondences. Figure 3 illustrates the meta-representation of this type of correspondences (see for example the association between ExportedNonTextNode and MediationNonTextNode classes and ExportedTextNode and MediationTextNode ones). For the time being only one-to-one correspondences and no transformation functions are represented.

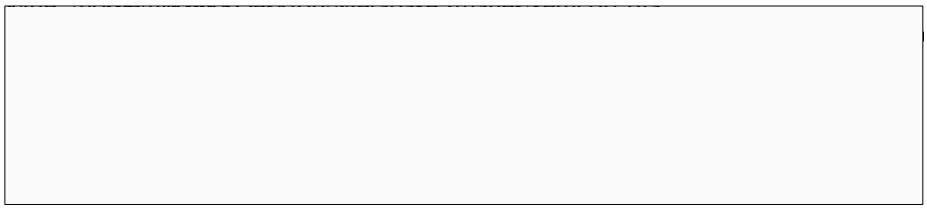

Fig. 3 – Semantic correspondences

2.3 Mediation query representation

A mediation query represents a strategy that can be used for populating a mediation schema by integrating instances of the exported schemas from the sources. A mediation query (MediationQuery class) is described by an identifier (id) and a query definition (see Figure 4). A mediation schema can have several associated mediation queries. On the other hand, a mediation query can be associated to one and only one mediation schema. This relationship is represented by the association between the classes Schema and MediationQuery.

Fig. 4 – Mediation Query

2.4 Source capabilities

Figure 5 (non-grayed zone) shows the UML diagram modeling local sources capabilities. A source (DataSource class) is hosted by an access provider (Provider class) and exports one or more native interfaces (NativeInterface class). A NativeInterface class can be mapped to one or more wrapper interfaces (WrapperInterface class). Each wrapper manages only one schema and one schema definition can be shared by different wrappers. Concerning computation capabilities, the queryOperator class models all possible operators that can be applied to the nodes of an exported schema. An operator has an input (hasInput relationship) and one or two operands, according to the kind of operation. One operator is associated to one or more predicates. Each predicate is applied over a non-finite set of nodes.

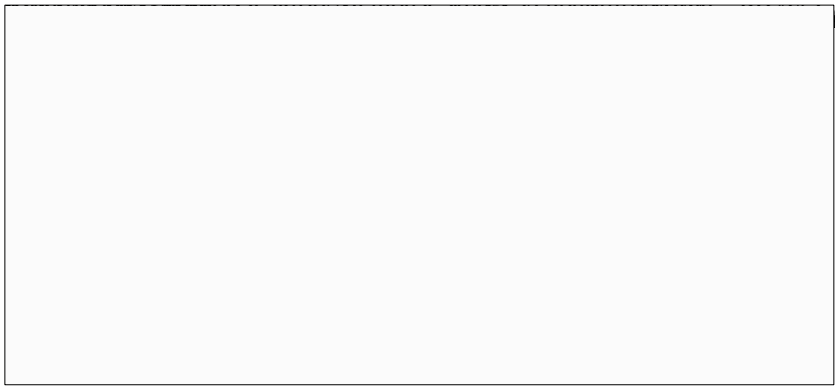

Fig. 5 – Source capabilities

2.5 Statistics

Statistics play a very important role in query evaluation. They can be obtained from data sources when they registered to a mediation system or they can be derived at execution time. We propose a meta-representation for two kinds of statistics (see Figure 6): (i) the Datastatistics class linked to the Node class represents data-oriented statistics -- characterized by the node cardinality, its min value and its max value ; (ii) the SystemStatistics class represents information about communication between mediators and wrappers such as data network rate.

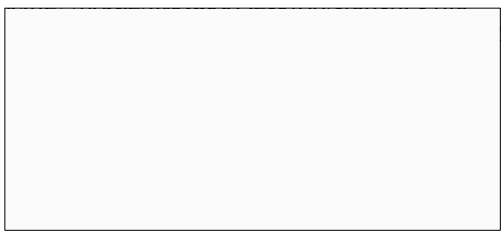

Fig. 6 – Statistics UML Diagram

3. Mediation queries generation

In a mediation system, given the descriptions of the mediation schema and the exported schemas, mediation queries are defined in order to express how instances of the mediation schema are derived from the exported schemas. These mediation queries represent mappings between the mediation schema and the exported schemas. Mappings are used for: (i) translating queries expressed on the mediation schema to sub-queries on exported schemas, and (ii) translating and integrating sub-queries results to produce a global result.

The goal of the mediation queries generation process is to discover candidate mediation queries given the descriptions of the mediation and exported schemas, which are given using XML schema[Fal01]. Generated mediation queries are XQuery ones [CCD+01].

Our approach comprises three main steps: (i) identifying the relevant portions of each data source considering the mediation schema, (ii) identifying the candidate operations between the relevant portions of data sources and (iii) generating mediation queries from candidate operations.

3.1 Identifying relevant portions of data sources

For each exported schema, the first step identifies the relevant portion, called a relevant schema, with respect to the mediation schema. To produce such a schema, we consider that some metadata is available, consisting mainly in a set of semantic correspondences defined between elements of the exported schemas and the mediation schema. A relevant schema is composed of elements of the exported schema involved in semantic correspondences and the keys and the references defined in the exported schema. The result of this step gives, for each exported schema, a relevant schema and a query allowing deriving this schema from the corresponding source. Figure 7 shows an example of relevant schema for the exported schema of the GOLD source[BEK01].

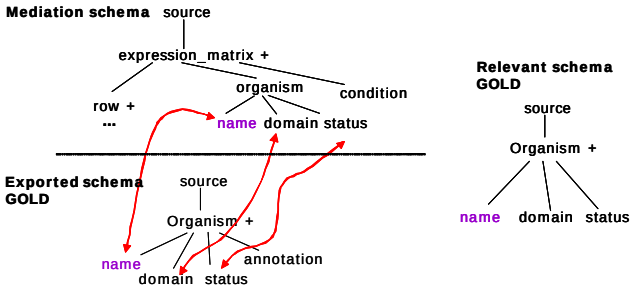

Fig. 7 – Identify the relevant portion of the GOLD data source

3.2 Identifying candidate operations

Once all relevant schemas are defined, the next step consists in searching for some candidate operations between them. In our work we have considered the join operator.

Using the semantic correspondences, the keys and references defined in the relevant schemas, candidate join operators are derived. A join operator can either be defined in the same relevant schema or between two relevant schemas. Each pair of relevant schemas can be combined with one or several candidate joins.

Figure 8 shows an example where we have the generated relevant schema of the GOLD data source (see Fig. 7) and another relevant schema corresponding to the biological data source SMD [SHK+01]. The only candidate operation between these two relevant schemas is a join in which the path "GOLD/source/organism/name" must correspond to the path "SMD/source/expression_data/org".

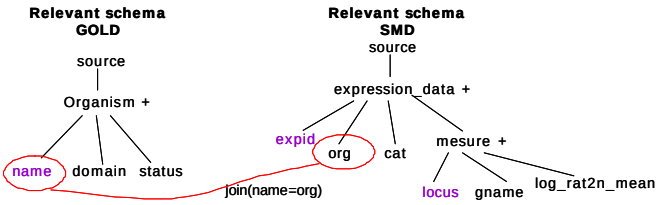

Fig. 8 – Identifying candidate operations between two relevant schemas

3.3 Generating mediation queries

Given the set of relevant schemas and the candidate operations between them, we consider different parts in the mediation schema; each part is a sub-tree in the mediation schema such that the root *n* of the sub-tree is either a multi-valued node or the root of the mediation schema, and the nodes of the sub-tree are all the mono-valued children of *n*. For each part of the mediation schema, we define a set of partial mappings, each of them corresponding to a way of populating the considered part from the exported schemas. A mediation query is defined as a combination of partial mappings such that there is one partial mapping for each part of the mediation schema. All the combinations of partial mappings are considered, some of them will lead to mediation queries. The result of our approach is a set of mediation queries having different semantics. The union of a sub-set of the derived mediation queries is also a mediation query.

4 Query evaluation

Mediation queries are used as input of the (unfolding) process to rewrite a global

query into expressions upon local sources. These expressions are then evaluated by an evaluator, so-called a Query Broker built from QBF, a Query Broker Framework.

QBF is an innovative reusable design represented by a set of component interfaces and the way they collaborate. Its implementation provides the basic functionalities for flexibly evaluating queries. Building query brokers from QBF means creating new subclasses and instances and configuring these instances together according to application requirements. Query Brokers can also adapt themselves to changes of the execution environment and/or of user and application requirements during query evaluation.

The following concentrates on queries representation within a broker, its general architecture, and the way its components interact to provide an adaptive query evaluation and to authorize interaction during query evaluation. More details are given in [CV04,VC04].

4.1 Query representation

The internal representation of a query is a standardized, canonical query tree so-called query plan. Query plan nodes, represented by the OperNode interface, are operators such as Select, Project, Join, Union, etc. Each operator can have one or several useable algorithms (Algorithm interface). These algorithms consume and produce sequences of items (tuples, entities or objects). More precisely, operator algorithms are implemented in the *iterator model* and provide the `open, next` and `close` operations.

Also, operator nodes are annotated by their properties such as the estimated size of the result, or the cost of operator execution. These properties are represented by the Property interface and are regrouped by an instance providing the Annotation interface.

A query also has a Context that determines constraints to be checked during query processing. Some examples of constraints are number of results handled by client, time limit or source preferences. A context is represented as a list of parameters, i.e. couples of (name, value).

Figure 9 represents a query plan for the example: "looks for the organisms completely published and eukaryote". This plan requires an evaluation on all the three biological sources (GOLD, SMD and SGD). Join nodes are used to combine data from the sources.

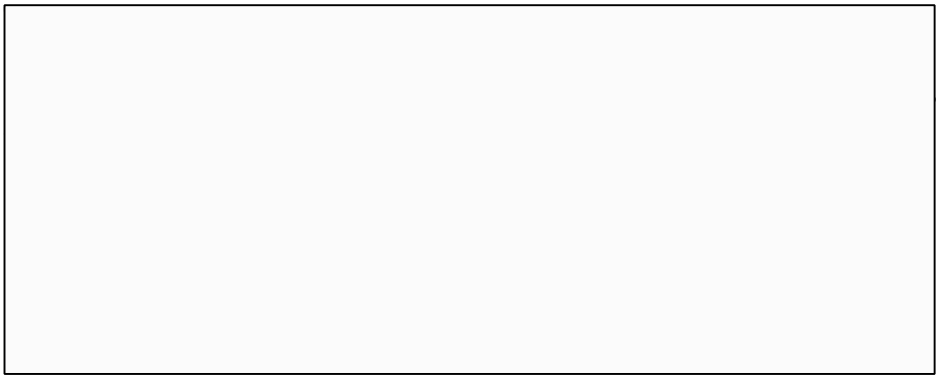

Fig. 9 – Example of query

4.1 Query broker components

Figure 10 shows the components of any query broker: each of them is designed to cope with a well-identified query evaluation concern. The separation of concerns of query execution and optimization is based on our analyses of existing query systems and optimization techniques such as in [SAC+79,SWA89,KD98,GD87,GM93]. The QueryManager component provides the interface of a query evaluator. It coordinates other components to evaluate queries. The PlanManager and the ContextManager components provide tools for managing query plan and query context parts, respectively. The BufferManager component provides storage capability for processing queries. These base components are required for building any query evaluator.

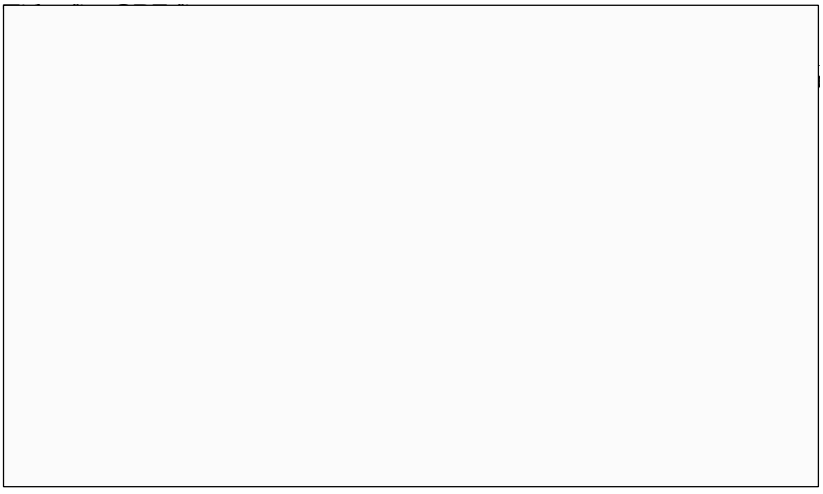

Fig. 10 – Query broker components

Fig. 11 – Plan manager components

Building an adaptive query evaluator needs use of the Monitor and the RuleManager components. These two components enable the observation of the query execution (Monitor) and define the way in which query evaluator reacts according to changes of the environment during query execution (RuleManager).

The PlanManager defines operations for manipulating query plans. This component covers all aspects of query optimization, i.e. search space, cost estimation and search strategy. It is composed of the Planner (providing the search strategy) that coordinates the activities of some sub-components dedicated to query plan optimization such as the Annotator, the Transformer and the Translator (see Fig. 11). The Annotator allows calculating properties of query nodes including the cost of query operations. The Transformer and the Translator provide possible query plan manipulations (logical and physical, respectively) which define the search space of a query plan.

The interaction between sub-components of the PlanManager is given by the sequence diagram in Figure 12. It is important to note that the number and the order of method calls from the Planner to the Annotator, to the Transformer and to the Translator are not fixed. This depends on the search strategy supported by the Planner.

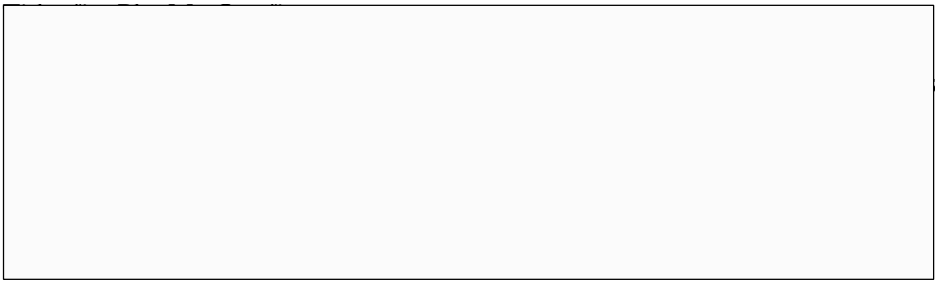

Fig. 12 – Sequence diagram for query optimization

The Monitor is responsible for detecting unexpected conditions in the execution

environment (such as network delays, use of resources or query context). It manages a list of observation elements which can be accessed through the PropertyMonitor interface. Each element has a check method defining how to detect unexpected condition, and a notify method being responsible of throwing events to the RuleManager. Sub-classes of the PropertyMonitor have been defined to monitor specific properties, like the arrival data rate (RateMonitor), the number of data processed (SizeMonitor) and the execution time (TimeMonitor).

The RuleManager receives events and launches one or more of the set of corresponding Event-Condition-Action (ECA) rule(s) to adapt the query execution (to changes in the execution environment) according to a pre-defined strategy (Strategy interface).

The RuleManager component can also support a complicated rule model composed of an event model and a reaction model. A detailed description of the rule manager is out of the scope of this work but can be found in [CVG00,VC02].

4.2 Adaptive evaluation

Different techniques for adaptive evaluation such as the ones in [KD98, AFT+96, HFC+00, AH00] have been integrated in a uniform way. Besides, an adaptive evaluation can also be ensured by using adaptive operators such as XJoin [UF00], ripple-join [HH99], etc. We can integrate all of these techniques in query brokers by implementing the corresponding rules and operators. For example, to provide the *query scrambling* technique initially proposed in [UFA98], we consider that network delays are detected by instances of the RateMonitor class. Query scrambling reacts to such delays in two ways: *rescheduling* and *operator synthesis* through the following two rules:

(1) When timeout
    If schedulable
    Do materialize(op)

(2) When timeout
    Do reoptimize(root,join-related)

The first rule (1) aims at rescheduling query plan when a delay is detected. The condition is a method to verify if another operator can be executed during the detected delay. In such case, a *Buffer* operator is inserted into the query plan so as to materialize results of the op operator, i.e. op will be executed during previously detected delay (call of materialize method) in order to delay the access to the missing data source.

The second rule (2) corresponds to operator synthesis aiming at reordering join operators so as to be able to evaluate a sub-plan during detected delay. The corresponding method is called to reoptimize the query with a particular strategy. Please note that a simple rule execution strategy should be defined over these two rules: rule (2) is executed when rule (1) cannot be executed anymore.

An adaptation in the case of unavailable data could be the generation of partial results. This allows better responding to user requirements and enables user interactions during query processing. Partial results are returned using techniques for directing and/or redirecting data flows between query operators. The same approach is adopted to authorize partial results, i.e. considering specific operators such as *switch*, *dummy* and specific adaptation rules [VC03].

4.3 Interactive evaluation

Looking at the first (incomplete) results, users can refine their long running queries as in [RRH99]. They can also modify their ongoing query (both the query context, as in [RH02], and the query plan) or 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 property parameters. No new input data is accessed but the system can continue to return results with data in process. This phase aims also at maintaining the coherence of data processing. The second phase directs and/or redirects data

flows between query operators in order to minimize query plan updates.

For enabling user interactions, we implement the UserInteractionMonitor class specializing the PropertyMonitor one. This class defines all possible user interactions for modifying query context, adding new selection condition, removing an existing selection condition, etc. as events. Examples of user interaction events are *addOperator*, *setupContextParameter* or *requestResult*. Rules for handling these events are defined and managed by the RuleManager component. These rules define system behaviors toward user interaction. The following rule is implemented to enable building partial results according to user requests:

When requestResult do returnResult

When the rule is launched, the system produces partial results using one of the available computing partial result strategies. For example, the simplest strategy is that each unavailable data source is replaced by a *dummy* operator, producing *any* item which can match with any item.

5. Conclusion

This paper presents results of the MediaGrid project whose objective is to contribute to the definition of an open mediation framework for accessing heterogeneous and largely distributed sources. This paper puts emphasis on the metadata management, the generation of mediation queries, and the adaptive and interactive evaluation of user queries.

Implantation of instances of the framework needed to build the specific mediation system providing a transparent access to the biological GOLD, SMD and SGD sources is in progress [Col03]. XML is used at the mediation level for meta-data and XQuery is used to formulate queries. Another on-going work concerns the architecture aspect of the framework and some tools to facilitate the construction of a mediation system itself and its deployment [BVC03].

The next step is to validate our approach considering: complexity of components, scalability, and performances (caching, context-aware operators, etc.).

References

[AH00] Avnur, R., Hellerstein, J.M.: Eddies: Continuously adaptive query processing. In Proc. of International Conference on Management Data (SIGMOD). (2000)

[AFT+96] Amsaleg, L., Franklin, M-J., Tomasic, A., Urhan, T. : Scrambling query plans to cope with unexpected delays. In: Proc. of Conference on Parallel and Distributed Information Systems (PDIS). (1996)

[BDD+00] Ball, C-A., Dolinski, K., Dwight, S-S., Harris, M-A., Issel-Tarver, L., Kasarskis, A., Scafe, C-R., Sherlock, G., Binkley, G., Jin, H., Kaloper, M., Orr, S-D., Shroeder, M., Weng, S., Zhu, Y., Botstein, D., Cherry, J-M. : Integrating functional genomic information into the Saccharomyces Genome Database. In: Nucleic Acid Res. (2000)

[BEK01] Bernal, A., Ear, U., Kyrpides N. : Genomes OnLine Database (GOLD) : a monitor of genome projects world-wide. In: Nucleic Acid Res. (2001)

[BFMV00] Bouganim, L., Fabret, F., Mohan, C., Valduriez, P.: Dynamic query scheduling in

data integration systems. In: Proc. of Inter. Conference on Data Engineering.(2000)
[BVC03] Bruno, G., Vargas-Solar , G., Collet, Ch.: *ADEMS, an Adaptable and Extensible Mediation Framework: application to biological sources.* In Proceedings of the Workshop on Advances in Databases and Information Retrieval , ISBN 970-36-0070-0, Tlaxcala, Mexico, Septembre 2003.
[CCD+01] Chamberlin, D., Clark, J., Robie, J., Florescu, D., Siméon, J., Stefanescu. M.: XQuery 1.0: An XML Query Language. W3C Working Draft, June 2001. http://www.w3.org/TR/xquery/.
[Col03] Collet, C. and the Mediagrid Project team, A Mediation Framework for a Transparent Access to Biological Data Sources, In Proceedings of the ECCB 2003 Conference Poster Session, Paris, novembre 2003. Long version presented at the 2003 Entity Relationship, published in the EMISA FORUM proceedings.
[CV04] Collet, C, Vu,T-T.: QBF: a Query Broker Framework for Adaptable Query Evaluation. In Proc. of the Sixth International Conference on Flexible Query Answering Systems (FQAS), June 24-26, Lyon, France.(2004)
[CVG00] Collet, C., Vargas-Solar, G., Grazziotin-Ribeiro, H.: Open active services for data intensive distributed applications. In: Proc. of International Database Engineering and Application Symposium (IDEAS). (2000)
[DD99] Domenig, R., Dittrich, K.R.: An Overview and Classification of Mediated Query Systems. In Sigmod Record.(1999)
[Fal01] Fallside. D.C. : XML Schema Part 0: Primer. W3C Recommendation. Mai 2001. http://www.w3.org/TR/xmlschema-0/.
[GD87] Graefe, G., DeWitt, D.J.: The exodus optimizer generator. In: Proc. of International Conference on Management Data (SIGMOD). (1987)
[GM93] Graefe, G., McKenna, W.J.: The volcano optimizer generator: Extensibility and effcient search. In: Proc. of International Conference on Data Engineering.(1993)
[KBo99]Kedad, Z., Bouzeghoub, M.,: 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.(1999)
[KD98] Kabra, N., DeWitt, D.J.: Effcient mid-query re-optimization of sub-optimal query execution plans. In: Proc. of International Conference on Management Data (SIGMOD).(1998)
[HFC+00] Hellerstein, J.M., Franklin, M.J., Chandrasekaran, S., Deshpande, A., Hildrum, K., Madden, S., Raman, V., Shah, M.A.: Adaptive query processing: Technology in evolution. IEEE Data Engineering Bulletin.(2000)
[HH99] Haas, P.J., Hellerstein J.M.: Ripple Joins for Online Aggregation. Proc. of SIGMOD.(1999)
[RRH99] Raman, V., Raman, B., Hellerstein, J-M.: Online dynamic reordering for interactive data processing. In: Proc. of International Conference on Very Large Data Bases (VLDB). (1999)
[RH02] Raman, V., Hellerstein, J-M.: Partial results for online query processing. In Proc. of ACM SIGMOD.(2002)
[SAC+79] Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G.: Access path selection in a relational database management system. In: Proc. of International Conference on Management Data (SIGMOD). (1979)
[SHK+01] Sherlock, G., Hernandez-Boussard, T., Kasarskis, A., Binkley, G., Matese, J-C., Dwight, S-S., Kaloper, M., Weng, S., Jin, H., Ball, C-A., Eisen, M-B., Spellman, P-T., Brown, P-O., Botstein, D., Cherry, J-M. : The Stanford Microarray Database. In: Nucleic Acid Res. (2001)
[SWA89] Swami, A.: Optimization of large join queries: Combining heuristics and combinatorial techniques. In: Proc. of Int.. Conference on Management Data. (1989)

[UF00] Urhan, T., Franklin, M.J.: Xjoin: A reactively-scheduled pipelined join operator. IEEE Data Engineering Bulletin 23.(2000)
[UFA98] Urhan, T., Franklin, M.J., Amsaleg, L Cost based query scrambling for initial delays. In: Proc. of International Conference on Management Data (SIGMOD). (1998)
[VC02] Vargas-Solar, G., Collet, C.: Adees: An adaptable and extensible event based infrastructure. In: Proc. of Database and Expert Systems Applications.(2002)
[VC03] Vu,T-T., Collet, C.: Query Brokers for Distributed and Flexible Query Evaluation. In Proc. of the Conference RIVF, Hanoi, Vietnam.(2003)
[VC04] Vu,T-T., Collet, C.: A Framework for Building Adaptable and Interactive Query Evaluators. In Proc. of the IDEAS Conf. (2004)
[Wie92] Wiederhold G.: Mediator in the Achitecture of Future Information Systems. The IEEE Computer Magazine, 25(3):38—49.(2002)
[YMH+01] Yan, L.L., Miller, R.J., Haas, L.M., Fagin, R.: Data-Driven Understanding and Refinement of Schema Mappings. SIGMOD Conference. (2001)