Hyper: A Framework for Peer-to-Peer Data Integration on Grids

Diego Calvanese[1], Giuseppe De Giacomo[2],
Maurizio Lenzerini[2], Riccardo Rosati[2], and Guido Vetere[3]

[1] Faculty of Computer Science, Free University of Bolzano/Bozen,
calvanese@inf.unibz.it
[2] Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza",
{lastname}@dis.uniroma1.it
[3] IBM Italia
gvetere@it.ibm.com

Abstract. Data Grids allow for seeing heterogeneous, distributed, and dynamic informational resources as if they were a uniform, stable, secure, and reliable database. According to this view, current proposals for data integration on Grids are based on the notion of global schema built over a collection of autonomous information sources. On the other hand, in dynamic and distributed environments, such a hierarchical and centralized architecture is not well suited for effective information integration. Peer-to-peer data integration aims at overcoming these drawbacks by modeling autonomous information systems as peers, and establishing mappings among peers without resorting to any hierarchical structure. In this paper, we present Hyper, a joint research initiative of Università di Roma "La Sapienza" and IBM Italia, which aims at developing principles and techniques for peer-to-peer data integration on a Grid infrastructure. The main contributions presented are a semantic characterization of P2P data integration, the deployment of our P2P framework on a Grid architecture, and the design of a query answering algorithm that is coherent both with the semantics and with the Grid infrastructure.

1 Introduction

Integrating heterogeneous computational resources and databases, which are distributed over highly dynamic computer networks, is the crucial challenge at the current evolutionary stage of IT infrastructures. Large enterprises, business organizations, e-government systems, and, in short, any kind of internetworking community, need today an integrated and virtualized access to distributed information resources, which grow in number, kind, and complexity. The notion of Virtual Organization denotes a set of individuals and/or institutions sharing data and computing resources by a range of collaborative strategies [10].

Grids aim at providing a suitable infrastructure for Virtual Organizations [14], based on standardized services that implement well-established and largely supported models. This kind of infrastructure hides the complexity of

M. Bouzeghoub et al. (Eds.): ICSNW 2004, LNCS 3226, pp. 144–157, 2004.

heterogeneous and distributed data sources, and handles the dynamics of the underlying networking environment. This motivates the current trend of modeling complex business infrastructures as Grids, and this is why Grid technologies, which have been developed in the research community, attract today so much interest in the industry. In fact, the Open Grid Services Architecture (OGSA) [13] is part of the foundational layer of IBM's on demand operating environment.

In particular, Data Grids allow for seeing heterogeneous, distributed, and dynamic informational resources as if they were a uniform, stable, secure, and reliable database, with the aim of facilitating the application development, speeding up business integration, and ultimately for the users' sake. Grid extensions, such as OGSA's Data Access and Integration Services (DAIS) [3], specifically address the issue of modeling Data Services over a Grid, and supply a homogeneous interface to a variety of data sources, such as relational DBMS or XML documents. Basically, these services wrap the access to physical data, so that Grid-enabled applications can easily locate, connect, and query virtual data sources by means of uniform interfaces, can transparently elaborate their data, and finally provide results to either users or other applications.

Grid-based Virtual Databases [21,20] are essentially loosely-coupled database federations, which integrate heterogeneous sources, with the purpose of responding to business demands in a flexible manner. The integration logic is generally contained in specific applications (called "analysts"), that achieve integration at "functional" level. In fact, "analysts" build the integrated virtual schema based on views over the source schemas, and the integration semantics they implement is usually enmeshed in their 'ad-hoc' internal code. As a consequence, a change in data sources (i.e. adding a new node, or changing metadata) would require reprogramming "analyst" applications. Researchers in the field of Semantic Grids (see e.g. [5]) claim that this is a threat in the implementation of many real interesting application scenarios, and their effort is aimed at overcoming this limitation. Indeed, "analyst-based" Grid Databases suffer of a certain rigidity, which limits the exploitation of Data Grids in many real situations.

Current proposals for data integration on Grids are based on a traditional architecture relying on the notion of global schema built over a collection of autonomous information sources. In this paper, we present a framework for data integration in peer-to-peer (P2P) systems built on a Grid infrastructure. In our framework, each peer represents an autonomous information system, and information integration is achieved by establishing mappings among the various peers without resorting to any hierarchical structure. Queries are posed to one peer, and the role of query processing is to exploit both the data that are internal to the peer, and the mappings with other peers in the system.

The main contributions presented in this paper are the following: (i) a semantic characterization of P2P data integration (described in Section 2); (ii) the deployment of our P2P framework on a Grid architecture (presented in Section 4); (iii) the design of a query answering algorithm that is coherent both with the semantics and with the Grid infrastructure (reported in Section 3).

2 The Hyper Framework for Data Integration

In this section, we set up a general framework for P2P data integration in a Grid operating environment. We base our proposal on the work reported in [9]. We refer to a fixed, infinite, denumerable, set Γ of constants. Such constants are shared by all peers, and are the constants that can appear in the P2P system. Moreover, given a relational alphabet A, we denote with $\mathcal{L}_A$ the set of function-free first-order logic (FOL) formulas whose relation symbols are in A and whose constants are in Γ.

A *conjunctive query* (CQ) of arity n over an alphabet A is written in the form

$$\{\mathbf{x} \mid \exists \mathbf{y}\ body_{cq}(\mathbf{x}, \mathbf{y})\}$$

where $body_{cq}(\mathbf{x}, \mathbf{y})$ is a conjunction of atoms of $\mathcal{L}_A$ involving the free variables (also called the *distinguished* variables of the query) $\mathbf{x} = x_1, \ldots, x_n$, the existentially quantified variables (also called the *non-distinguished* variables of the query) $\mathbf{y} = y_1, \ldots, y_m$, and constants from Γ.

A *P2P system* $\mathcal{P}$ is constituted by a set of *datapeers* and a set of *hyperpeers*. Each datapeer $D \in \mathcal{P}$ is a system that exports data (possibly coming from different sources) in terms of an exported schema.

Hyperpeers instead do not have access to local data but are interconnected with both hyperpeers and datapeers from which they extract data. Formally, a hyperpeer $H \in \mathcal{P}$ is a tuple $HP = (G, M)$, where:

- G is the *schema* of HP, which is a relational schema over a relational alphabet A_G (disjoint from the other alphabets in $\mathcal{P}$) called the *alphabet* of HP.
- M is a set of *P2P mapping assertions*, each of which is an expression of the form

  $$cq' \rightsquigarrow cq$$

  The query cq, called the *head* of the assertion, is a conjunctive query over the peer (schema of) HP, while the query cq', called the *tail* of the assertion, is a conjunctive query of the same arity as cq, over (the schema of) one of the other peers in $\mathcal{P}$.

In a hyperpeer $HP \in \mathcal{P}$, a P2P mapping assertion $cq' \rightsquigarrow cq$, where cq is a query over the schema of the peer HP, expresses the fact that HP can use data retrieved by cq' from the peer P' over (the schema of) which cq' is expressed. Such data are mapped to the schema of HP according to what is specified by the query cq. This form of mapping is one of the most expressive among those studied in the P2P and data integration literature. Indeed, in terms of the terminology used in data integration, a hyperpeer connected to datapeers only corresponds to a GLAV *data integration system* [15,18] managing a set of sound data sources defined in terms of a (virtual) global schema.

Observe that no limitation is imposed on the topology of the whole set of P2P mapping assertions in the peer system $\mathcal{P}$, and hence the set of all P2P mappings may be cyclic.

Finally, we assume that *queries* that are posed to the P2P system $\mathcal{P}$ are in fact posed to one of the peers (datapeer or hyperpeer) P of $\mathcal{P}$. Such queries are expressed in a certain relational query language $\mathcal{L}_P$ (which must include conjunctive queries) over the schema of P. For now, we make no specific assumption on the query language $\mathcal{L}_P$, except that the peer P can indeed process queries belonging to $\mathcal{L}_P$, and we say that the queries in $\mathcal{L}_P$ are those *accepted by* P.

We assume that the peers are interpreted over a fixed infinite domain Δ. We also fix the interpretation of the constants in Γ (cf. previous section) so that: (i) each $c \in \Gamma$ denotes an element $d \in \Delta$; (ii) different constants in Γ denote different elements of Δ; (iii) each element in Δ is denoted by a constant in Γ.[1] It follows that Γ is actually isomorphic to Δ, so that we can use (with some abuse of notation) constants in Γ whenever we want to denote domain elements.

We focus first on the semantics of a datapeer DP. We call *database* for DP a finite relational interpretation D of the relation symbols in schema of DP. Let q be a query of arity n, expressed in the query language $\mathcal{L}_{DP}$ accepted by DP, and let D be a database for DP. We denote by $ans(q, DP, D)$ the set of n-tuples of constants in Γ obtained by evaluating q in the database D, according to the semantics of $\mathcal{L}_{DP}$.

To define the semantics of hyperpeers we resort to epistemic logic[2]. The advantages of using epistemic logic instead of First Order Logic (FOL) are illustrated in [9]. In synthesis:

- While in the traditional FOL interpretation the whole is modeled as a flat theory, in our setting peers are modeled as autonomous sites that exchange information and the modular structure of the system is explicitly reflected in the definition of its semantics.
- The new semantic characterization leads to a setting where query answering is decidable, and polynomially tractable in the size of the data.
- The topology of the mapping assertions among the peers in the system is not limited in any way. In particular, while in FOL acyclicity of mapping assertions is often adopted in order to ensure decidability of query answering, in our setting we do not need to impose such a limitation.

It should be noted that the resulting semantics is weaker than the one based on FOL, but we argue that this is exactly the price to pay in order to get all the above advantages.

The use of epistemic logic is based on the idea that P2P mappings are formalized as axioms of an epistemic logic theory. More precisely, a P2P system $\mathcal{P}$ is formalized as an epistemic theory $M_{\mathcal{P}}$ formed by one axiom of the form

$$\forall \mathbf{x}\ (\mathbf{K}(\exists \mathbf{y}\ (body_{cq_1}(\mathbf{x}, \mathbf{y}))) \supset \exists \mathbf{z}\ body_{cq_2}(\mathbf{x}, \mathbf{z}))$$

for each P2P mapping assertion $cq_1 \rightsquigarrow cq_2$ in the peers of $\mathcal{P}$. Intuitively, this formalization of the P2P mapping assertions reflects the idea that only what is

[1] In other words the constants in Γ act as *standard names* [19].

[2] Technically we resort to epistemic FOL with standard names, and therefore with a fixed domain, and rigid interpretation of constants [19].

known by the peers (denoted by the **K** operator) mentioned in the tail of the assertion is transferred to the peer mentioned in the head.

Then, let a *system database for* $\mathcal{P}$ be a database formed by the disjoint union of the various databases of the datapeers in $\mathcal{P}$. Let a *FOL model of* $\mathcal{P}$ *based on a system database* $\mathcal{D}$ be any FOL model for the disjoint union of the schema of the peers in $\mathcal{P}$, such that the extensions of the relations in the schemas of the datapeers are those sanctioned by $\mathcal{D}$. An *epistemic model of* $\mathcal{P}$ *based on* $\mathcal{D}$ is a pair $(\mathcal{I}, \mathcal{W})$ such that $\mathcal{W}$ is a set of FOL models of $\mathcal{P}$ based on $\mathcal{D}$, $\mathcal{I} \in \mathcal{W}$, and $(\mathcal{I}, \mathcal{W})$ satisfies all the axioms of $M_{\mathcal{P}}$. In particular, the axiom

$$\forall \mathbf{x}\ (\mathbf{K}(\exists \mathbf{y}\ (body_{cq_1}(\mathbf{x}, \mathbf{y}))) \supset \exists \mathbf{z}\ body_{cq_2}(\mathbf{x}, \mathbf{z}))$$

in $\mathcal{M}_{\mathcal{P}}$ is satisfied by $(\mathcal{I}, \mathcal{W})$ if for every tuple $\mathbf{t}$ of objects in Γ, the fact that $\exists \mathbf{y}\ body_{cq_1}(\mathbf{t}, \mathbf{y})$ is satisfied in every FOL model in $\mathcal{W}$ implies that also $\exists \mathbf{z}\ body_{cq_2}(\mathbf{t}, \mathbf{z})$ is satisfied in every FOL model in $\mathcal{W}$.

Finally, given a query q over one of the peers P in $\mathcal{P}$ and a system database $\mathcal{D}$ for $\mathcal{P}$, we define the *certain answers* $ans_{\mathbf{K}}(q, \mathcal{P}, \mathcal{D})$ to q in $\mathcal{P}$ based on $\mathcal{D}$, as the set of tuples $\mathbf{t}$ of constants in Γ such that for every epistemic model $(\mathcal{I}, \mathcal{W})$ of $\mathcal{P}$ based on $\mathcal{D}$, we have that $\mathbf{t} \in ans(q, P, \mathcal{I})$, where we consider P as a datapeer and $\mathcal{I}$ as a database for P.

3 Query Answering in Hyper

In order to address query processing in Hyper, we start by defining a preliminary transformation of a P2P system. Given a hyperpeer $HP = (G, M)$, we break each P2P mapping assertion $cq' \rightsquigarrow cq$ in M between a peer P' and HP in two halves, introducing an auxiliary predicate r of the same arity as cq'. We then denote as q_r the query cq' in the tail of the corresponding P2P mapping assertion, and denote as P_r the peer P', i.e., the peer over which the query q_r is expressed. Then, we replace the mapping $cq' \rightsquigarrow cq$ by a *local* mapping assertion $\{\mathbf{x} \mid r(\mathbf{x})\} \rightsquigarrow cq$ (that has the form of a LAV mapping [18]) and a simplified P2P mapping $cq' \rightsquigarrow \{\mathbf{x} \mid r(\mathbf{x})\}$ (that has the form of a GAV mapping [18]). For each peer HP, we call *auxiliary alphabet* of HP, denoted as $AuxAlph(HP)$, the set of new predicate symbols thus defined.

Such a system modification transforms each hyperpeer $HP = (G, M)$ in a LAV data integration system $(\mathcal{G}, \mathcal{S}, M)$ [18], where the global schema $\mathcal{G}$ is the original schema G of the peer HP, the schema of the local sources $\mathcal{S}$ is constituted of the set of auxiliary predicate symbols $AuxAlph(HP)$, and the mapping from the local to the global elements is provided by the local mapping assertions introduced above.

From the semantic point of view, the local mapping assertion is interpreted (without involving the knowledge operator) simply as:

$$\forall \mathbf{x}\ r(\mathbf{x}) \supset \exists \mathbf{y}\ body_{cq}(\mathbf{x}, \mathbf{y}))$$

Informally, in each peer the local sources corresponding to the predicates in the auxiliary alphabet are used to "simulate" the effect of the P2P mapping

assertions with respect to contributing to the data of the peer. It is possible to show that this new formulation of a hyperpeer is semantically equivalent to the original one.

We then define a distributed algorithm for answering queries in $\mathcal{L}_U$. Again the algorithm proposed is a version of that in [9], suitably specialized for the Hyper Framework. More specifically, we define the two main functionalities that each peer must provide in order to answer a user query to any peer in the system. Such functionalities are executed over a given source database $\mathcal{D}$, which represents the state of the datapeers when the query is issued by the user.

Each user query q to the peer P is the input of the extensionalQueryAnswering of P. If P is a datapeer, this module simply activates the query answering service of the datapeer to answer q. Therefore, in the following we assume that the query q is posed to a hyperpeer P. In such a case, this module first initiates a transaction, that is identified in the system by a unique transaction id, then passes the query q to its own intensionalQueryAnswering. Such a functionality returns a Datalog program DP (which involves queries to the datapeers of $\mathcal{P}$). The evaluation $Eval(DP)$ of such a program DP constitutes the answer set of the query q.

The intensionalQueryAnswering computes the Datalog program corresponding to the query q as follows:

1. first, a module computes the *perfect reformulation* of the query q with respect to the local mapping assertions in the peer. This step consists of expressing the query q over the schema of P in terms of an "equivalent" query q' over the local sources of P, i.e., in terms of the auxiliary alphabet $AuxAlph(P)$, which corresponds to the well-known problem of query rewriting using views [16,18].
2. then, for each predicate r in $AuxAlph(P)$ occurring in such a reformulation: (i) if the predicate r is generated by a mapping to a datapeer P_r, then a special rule $r \leftarrow q_r : address(P_r)$ is added to the Datalog program. In such a special rule, $address(P_r)$ represents the URL of the node corresponding to the datapeer P_r and q_r is the query to P_r associated with the auxiliary predicate r; (ii) if r is generated by a mapping to a hyperpeer P_r, then the P_r.intensionalQueryAnswering(q_r) is called and the resulting Datalog rules are added to the Datalog program.

As concerns $Eval(DP)$, such a procedure executes the following steps: (i) the procedure first retrieves the data corresponding to the extensional predicates of the program DP as follows. For each extensional predicate r there is a special rule $r \leftarrow q_r : address(P_r)$ in DP: the procedure asks the query q_r to the node P_r (whose URL is represented by $address(P_r)$); the answer set thus obtained constitutes the extension of the predicate r. Let D be the EDB (i.e., the set of facts) thus constructed; (ii) the Datalog program DP' obtained from DP by deleting all special rules and adding the EDB D is evaluated in the standard way.

We remark that, in order to guarantee that the intensionalQueryAnswering never processes the same mapping query twice in the same transaction, suitable checks are implemented through the procedures setTransaction and

Algorithm P.extensionalQueryAnswering
Input: user query $q \in \mathcal{L}_U$ to the hyperpeer P
Output: $ans_{\mathbf{k}}(q, \mathcal{P}, \mathcal{D})$
begin
  generate a new transaction id T;
  $DP := P$.intensionalQueryAnswering(q, T);
  return $Eval(DP)$
end

Algorithm P.intensionalQueryAnswering
Input: query $q \in \mathcal{L}_U$ to the hyperpeer P, transaction id T
Output: Datalog program DP
begin
  $DP :=$ computePerfectReformulation(q, P);
  for each (predicate $r \in AuxAlph(P)$ occurring in the bodies of DP)
    if (getTransaction$(r, T) = notProcessed$) **{**
      setTransaction$(r, T, processed)$;
      if (r is generated by a mapping to a datapeer)
        $DP := DP \cup \{r \leftarrow q_r : address(P_r)\}$
      else /* r is generated by a mapping to a hyperpeer */
        $DP := DP \cup P_r$.intensionalQueryAnswering(q_r, T);
    }
  return DP
end

Fig. 1. Algorithms extensionalQueryAnswering and intensionalQueryAnswering in hyperpeers

getTransaction. More precisely, two different states are associated to each predicate symbol r in $AuxAlph(P)$ with respect to the transaction T. If the state of r with respect to transaction T is *notProcessed*, then the mapping query q_r still has to be processed in the transaction T, therefore the intensionalQueryAnswering has to compute the answer to such a query. If the state of r with respect to transaction T is *processed*, then the mapping query q_r has already been processed in the transaction T, so the intensionalQueryAnswering does not process it again. Of course, when a new transaction is started by the extensionalQueryAnswering, all predicates are initially in the *notProcessed* state for such a transaction.

The two algorithms are reported in Figure 1. In can be shown that the result returned by extensionalQueryAnswering invoked on a hyperpeer for a query q, is exactly $ans_{\mathbf{k}}(q, P, \mathcal{D})$, where $\mathcal{D}$ is the system database corresponding to the extension of all datapeers.

4 Implementing the Hyper Framework on Grids

This section outlines how the Hyper Framework (as described in the previous section) can be developed as an OGSA-compliant Data Access and Integration

Service. By implementing this standard, Hyper will exploit available, well supported and understood infrastructures. On the other hand, Hyper will be easily integrated with existing environments, implementations, and tools. Although Grid and P2P computing are generally regarded as two different notions, we recognize that OGSA Grids provide a suitable infrastructure for P2P computing, in that they allow nodes discovering, binding, and exchanging data the one another in their environment, without hierarchical constraints. Furthermore, we believe that P2P architectures address many interesting Grid application problems.

4.1 OGSA Data Services

Based on the Open Grid Service Architecture [13], Grid Data Access and Integration Services (OGSA-DAI) [3] provide a reference architecture for data integration in distributed and heterogeneous environments. Before introducing OGSA-DAI, we summarize the basic Grid terminology and standards. The Open Grid Service Architecture (OGSA) specifies a framework where Grid objects and flows are designed as a set of standardized services and data models. Grid services are characterized as state-full distributed objects that can be instantiated, identified, searched, monitored, notified, and destroyed. By implementing OGSA services, applications can be integrated within a distributed operating environment, and cooperate the one another. The Open Grid Services Infrastructure (OGSI) [4] supplies a Web Services substrate for OGSA Grid functionalities, while Globus Toolkit [2] provides an open source, reference implementation of OGSI. Basically, OGSI Grid Services are Web Services that implement a set of standardized interfaces to implement Grids. Currently, enhancements to OGSI are being developed, based on the WS-Resource Framework [6], with the aim of converging with the most recent developments of Web Service standards. Anyway, Web based Grids can be viewed as special kinds of distributed object systems, which leverage Web Services standards and machineries without introducing any further binding to hosting environments, thus obtaining an unprecedented world-wide interoperability of information systems. In the OGSI setting, Grid objects (i.e., GridService instances) are created by invoking *Factory Services* that provide them with network identity and binding information, based on permanent handles (GSHs) and references (GSRs), which contain (possibly changing) addressing data.

Furthermore, GridService instances maintain *Service Data Elements* (or SDEs) that allow accommodating any kind of instance attribute, with standard access, manipulation, and search methods. Also, clients can subscribe for notifications regarding changes of specific data elements.

Based on OGSA, the Data Access and Integration (OGSA-DAI) specification details a service-oriented treatment of heterogeneous data sources, by modeling them as Grid services. The main purposes of OGSA-DAI are summarized in [3], as follows:

- Integrate data sources and resources into OGSA-compliant architectures.
- Obtain information about data that may be distributed amongst several heterogeneous database environments.

GridService finds service instances, manages handlers and supplies data services

DataDescription supplies data service descriptions (e.g. metadata)
    RelationalDescription supplies relational metadata (e.g. tables and columns names)
DataAccess manages data access
    SQLAccess manages SQL queries
DataServiceFactory creates and configures GridService instances
    SQLDataServiceFactory creates and configures SQL Data Service instances

Fig. 2. DAI interface hierarchy

- Locate the data that may be distributed, or replicated, over many different types of databases, the locations of which may not be known beforehand.
- Integrate data models that may be different on distributed databases.
- Find the databases that hold the required data and to be in a position to be able to interpret that data.
- Access that data through uniform interfaces.
- Integrate data from various sources to obtain the required information.

In practice, Data Services are special kinds of GridService instances that implement a suitable set of description, access, manipulation, and query interfaces. Data Services are qualified and described by specific SDE arrays, which convey relevant information such as metadata for structured or semi-structured databases. Interestingly, Data Services standardize and virtualize the access to source metadata, as well as query interfaces, thus providing a powerful abstraction of the underlying database infrastructures. The OGSA-DAI proposal defines three main WSDL interfaces (portTypes) for describing, accessing, and instantiating data sources, called DataDescription, DataAccess, and DataServiceFactory, respectively, which can be specialized to support different metadata structures or access methods (cf. Figure 2). This way, specific database systems, such as RDBMS, can be wrapped by appropriate implementations of DAI interfaces. Moreover, implementations can represent virtual views instead of concrete sources, and, in turn, can get integrated by higher level virtual data services. In brief, OGSA-DAI allow data providers wrapping data sources and mediating their access, thus providing a virtualization mechanism in which different data models can be easily integrated.

4.2 Hyper Data Services

We will specify here Hyper Data Services (HDS) on the basis OGSA (DAI) standards (c.f. Figure 3). From OGSA-DAI, HDS inherits:

- peer identity, which is provided by GSHs and GSRs
- peer discovery and binding

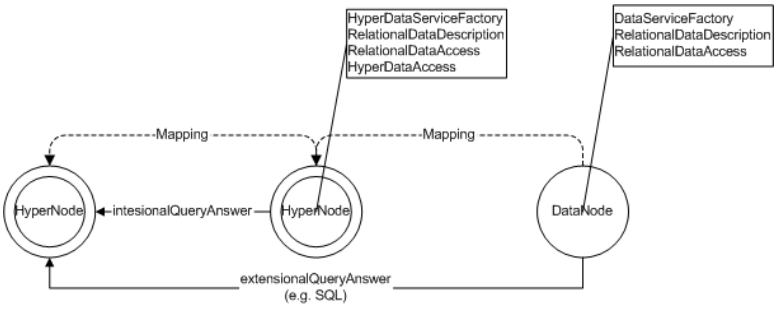

Fig. 3. Hyper Data Services

– peer typing, by means of specific SDEs
– extensional query answering services, for any suitable query language
– transaction identification for intensional queries, which relies of GSHs

First, we refer to generic DAI Service instances as *nodes*, which can be used as data sources by HDS, with provision that they supply relational metadata access services as well relational query services. We don't require nodes exposing any particular metadata format, or support any particular query language. Instead, we assume that nodes provide enough information for accessing their relational alphabets, where each relation is given, at least, a symbol and an arity. Nodes can provide such kind of information trough their DataDescription data elements. For instance, RelationalDescriptions [8], are likely to expose information about relational schemas, including tables, columns, column types, and keys. As such, they would allow dealing with relational metadata according to the abstraction outlined above. Nodes will also provide extensional query answering services with their standard means.

Then, we define *hypernode* any node that supports DataAccess methods implementing the system-wide query answering algorithms described in Section 3. With respect to the general P2P framework, hypernodes correspond to hyperpeers, whereas nodes correspond to datapeers. Hypernode metadata will be exposed with standard DataDescription ports, that is, WSDL portTypes with no standard operations that provide a suitable structure of SDEs. Once again, the format of hyper metadata could be one of those envisioned in DAI, as long as it allows handling a basic relational alphabet. Extensional query services will be supported by standardized DataAccess operations as well.

The binding to DAI metadata and access methods that characterize a specific hypernode (e.g. Relational, XML) is called *hypernode's flavour*. For each supported flavour, a conversion from the specific access language to the hypernode internal relational metadata will be implemented. This will guarantee a seamless integration of hypernodes with any other agent in their Data Grid environments. At the current stage, only an SQL flavour is being designed, but future developments could include other flavours, such as XML.

HyperAccess: intensionalQueryAnswering (IN trans, IN query, OUT datalog, OUT fail)

- trans – the transaction identifier (GSH)
- query – the body of the intensional query (conjunctive query)
- datalog – the resulting Datalog program
- fail – an error code in case of abnormal termination

Fig. 4. intensionalQueryAnswering operation

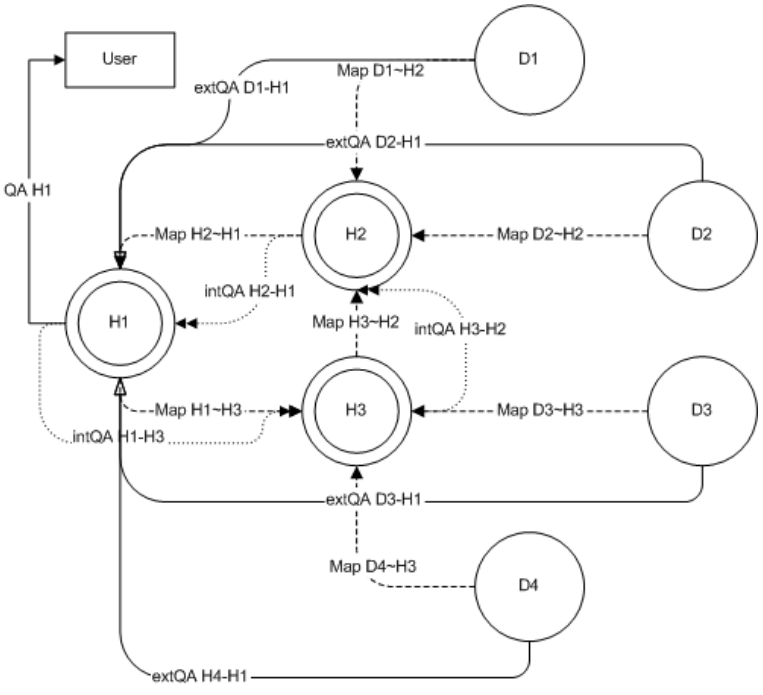

Fig. 5. Interaction among peers during query answering: an example

As mentioned above, extensional DataAccess operations will rely on standard DAI methods. Intensional queries, on the other hand, will require a specific support. This will be provided by the HyperAccess portType, which is an extension of DAI DataAccess. Therefore, hypernodes will support two DataAccess ports, which is allowable in DAI infrastructures. HyperAccess defines a single operation, named intensionalQueryAnswering (cf. Figure 5).

When a query is issued against a hypernode (cf. Figure 5), a transient DataAccess service is created, which allows accessing the result set of the query according to the node's flavour. The GSH of the newly created service identifies the query session. The inquired node examines the query against its

internal mapping structure, and lists out nodes that are relevant for answering the query, or some of its parts. Each mapped node that is a hypernode (this can be checked at run time by accessing the node service metadata), is inquired with an intensional query, which takes as input the session GSH and the query fragment which is relevant to the mapped node. Recursively, the mapped node will invoke intensional query operations on the hypernodes, according to its mappings. Note that the evaluation of cycles in mappings can be stopped thanks to the session GSH, which allows nodes detecting if an intensional query that has already been answered is issued back because of a cycle. The distributed intensional query process results in a Datalog program, which is evaluated by the node where the query was originated. This evaluation results in Grid-wide data access plan, that is, a list of extensional queries to be posed to other nodes in the Grid in order to get the integrated response. Finally, each relevant data node will be inquired through its standard extensional query interface, and the global result will be made available to the access service associated to the query session.

5 Conclusions

In this paper we have presented Hyper, a joint research initiative of Università di Roma "La Sapienza" and IBM Italia, which aims at developing a Data Grid integration system based on peer-to-peer schema mappings, which resorts on epistemic logic for providing a semantics for the mappings among peers. Also, we have presented an OGSA-DAI extension defining data services (hypernodes) which are capable of integrating any other standard data service in a Data Grid. Thanks to the peer-to-peer architecture, hypernodes can enter and leave the Grid, or change their own structure, without enforcing any change neither in other Grid participants, nor in "analyst" nodes. In fact, they get integrated in the overall information system, as far as other hypernodes are interested in establishing and maintaining a mapping with them, in a unsupervised manner. Of course, uncontrolled dynamics in a Hyper Data Grid would cause dangling mappings. However, this would only affect the information completeness, without breaking the system functionality.

Differently from simple key-based Semantic Overlay Networks [11], Hyper does not enforce any "clustering" based on shared conceptual structures covering the entire Grid semantics. Of course, Hyper is concerned with high quality query answering on complex structured data, and should not be compared with performance-oriented infrastructures dealing with loosely structured information. With respect to enhanced, semantically rich, and self-organizing Semantic Overlay Networks such as [7], or similar data management infrastructures such as [17], Hyper uses a simple relational language for expressing both schemas and mappings, rather than new class-based languages like RDFS and OWL. This way, Hyper facilitates the exploitation of well-supported industrial standards such as OGSA-DAI, and allows for a smooth integration of legacy business databases.

Many relevant aspects of a complete P2P data integration environment are not addressed in our system so far. Above all, we would mention the problem of

individual mapping [22,12], i.e., how to handle the identity of values denoted by different constants in different peers. If referred to objects denoted by network-wide identifiers (e.g. URIs), such a referential consistency could be ensured by a suitable "instance mapping" mechanism. This problem, however, is not addressed in our current framework.

Another open issue is that of discovering metadata correspondences, with the purpose of facilitating (at best, automating) the creation of metadata mappings. Simple relational metadata schemas, such as those considered here, would give just a little support to such a discovery. The adoption of rich ontology languages, such as those conceived for Semantic Web [1], if supplemented with good lexicalizations and links to reference ontologies (e.g. WordNet) would give an appropriate support to such a discovery. The way metadata mappings are established, however, is out of the scope of the Hyper Framework.

Acknowledgments. This research has been partially supported by the projects INFOMIX (IST-2001-33570), SEWASIE (IST-2001-34825) and INTEROP Network of Excellence (IST-508011) funded by the EU, by the project "Società dell'Informazione" subproject SP1 "Reti Internet: Efficienza, Integrazione e Sicurezza" funded by MIUR – Fondo Speciale per lo Sviluppo della Ricerca di Interesse Strategico, and by project HYPER, funded by IBM through a Shared University Research (SUR) Award grant.

References

1. W3C semantic web, 2001. `www.w3.org/2001/sw`.
2. The globus alliance, 2004. `www.globus.org`.
3. Open grid services architecture data access and integration, 2004. `www.ogsadai.org.uk`.
4. Open grid services infrastructure, 2004. `forge.gridforum.org/projects/ogsi-wg`.
5. The semantic grid, 2004. `www.semanticgrid.org`.
6. The ws-resource framework, 2004. `www.globus.org/wsrf`.
7. K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building internet-scale semantic overlay networks. Technical Report IC/2004/38, EPFL, 2004.
8. M. Antonioletti, A. Krause, S. Hastings, S. Langella, S. Malaika, J. Magowan, S. Laws, and N. W. Paton. Grid data service specification: The relational realisation. Technical report, DAIS Working Group, 2003.
9. D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the 23nd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2004)*, 2004. To appear.
10. L. Camarinha-Matos, H. Afsarmanesh, C. Garita, and C. Lima. Towards an architecture for virtual enterprises. *J. Intelligent Manufacturing*, 9(2), 1998.
11. A. Crespo and H. Garcia-Molina. Semantic overlay networks for P2P systems. Technical report, Computer Science Department, Stanford University, 2002.
12. A. Doan, Y. Lu, Y. Lee, and J. Han. Profile-based object matching for information integration. *IEEE Intelligent Systems*, 18(5):54–59, 2003.

13. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, 2002.
14. I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15(3), 2001.
15. M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.
16. A. Y. Halevy. Answering queries using views: A survey. *Very Large Database Journal*, 10(4):270–294, 2001.
17. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: data management infrastructure for semantic web applications. In *Proc. of the 12th Int. World Wide Web Conference (WWW 2003)*, pages 556–567, 2003.
18. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
19. H. J. Levesque and G. Lakemeyer. *The Logic of Knowledge Bases*. The MIT Press, 2001.
20. P. McBrien and A. Poulovassilis. Distributed databases. In M. Piattini and O. Diaz, editors, *Advanced Database Technology and Design*. Artech House, 2000.
21. N. Paton, M. Atkinson, V. Dialani, D. Pearson, T. Storey, and P. Watson. Database access and integration services on the grid. Technical Report UKeS-2002-03, UK e-Science Programme, National e-Science Centre, 2002.
22. G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. pages 4–18, 1995.