Querying a super-peer in a schema-based super-peer network [⋆]

Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio Vincini

Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905, 41100 Modena, Italy
bergamaschi.sonia@unimore.it

Abstract. We propose a novel approach for defining and querying a super-peer within a schema-based super-peer network organized into a two-level architecture: the low level, called the peer level (which contains a mediator node), the second one, called super-peer level (which integrates mediators peers with similar content).
We focus on a single super-peer and propose a method to define and solve a query, fully implemented in the SEWASIE project prototype.
The problem we faced is relevant as a super-peer is a two-level data integrated system, then we are going beyond traditional setting in data integration. We have two different levels of *Global as View* mappings: the first mapping is at the super-peer level and maps several *Global Virtual Views* (GVVs) of peers into the GVV of the super-peer; the second mapping is within a peer and maps the data sources into the GVV of the peer. Moreover, we propose an approach where the integration designer, supported by a graphical interface, can implicitly define mappings by using *Resolution Functions* to solve data conflicts, and the *Full Disjunction* operator that has been recognized as providing a natural semantics for data merging queries.

1 Introduction

Current peer-to-peer (P2P) networks support only limited meta-data sets such as simple filenames. Recently a new class of P2P networks, so called schema based P2P networks have emerged (see [1–4]), combining approaches from P2P as well as from the data integration and semantic web research areas. Such networks build upon peers that use metadata (ontologies) to describe their contents and semantic mappings among concepts of different peers' ontologies. In particular, in Peer Data Management Systems (PDMS) [2] each node can be a data source, a mediator system, or both; a mediator node performs the semantic integration of a set of information sources to derive a global schema of the acquired information.

[⋆] This research has been partially funded by the UE-IST SEWASIE project and the italian MIUR PRIN WISDOM project.

As stated in a recent survey [5], the topic of semantic grouping and organization of content and information within P2P networks has attracted considerable research attention lately (see, for example, [6, 7]). In super-peer networks [8], metadata for a small group of peers is centralized onto a single super-peer; a super-peer is a node that acts as a centralized server to a subset of clients. Clients submit queries to their super-peer and receive results from it; moreover, super-peers are also connected to each other, routing messages over this overlay network, and submitting and answering queries on behalf of their clients and themselves. The *semantic overlay clustering* approach, based on partially-centralized (super-peer) networks [9] aims at creating logical layers above the physical network topology, by matching semantic information provided by peers to clusters of nodes based on super-peers.

In this paper we propose an approach which combines the schema-based and super-peer network approaches, that is a *schema-based super-peer* network (called SEWASIE network from the UE IST project where it was developed - www.sewasie.org) organized into a two-level architecture: the low level, called the peer level (which contains a mediator node), the second one, called super-peer level, (which integrates mediators peers with similar content). More precisely,

- a *peer* contains a data integration system, which integrates heterogeneous data sources into an *ontology* composed of: an annotated *Global Virtual View* (GVV) and *Mappings* to the data source schemas.
- a *super-peer* contains a data integration system, which integrates the GVV of its peers into an *ontology* composed of a GVV of the peers GVVs and Mappings to the GVVs of its peers.

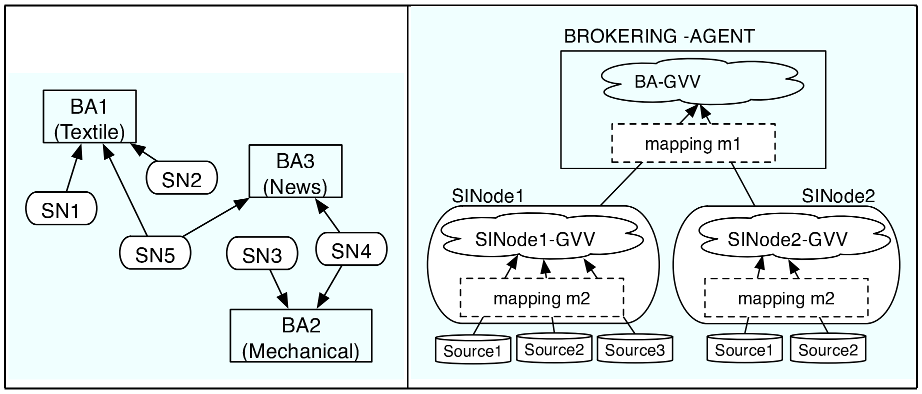


Fig. 1. (a) The SEWASIE network; **(b)** The Brokering Agent/SINodes architecture

A typical scenario of the SEWASIE network is shown in Figure 1.a, where many data peers, called `SINodes` (SN1 to SN5) are linked to different super-peers, called *Brokering Agents* (BA1 to BA3), according to their semantic content. For example, SN1, SN2, SN5 contain semistructured data sources related to the *textile domain* (textile enterprises, news, categories, ...) data sources and are clustered in the same BA (BA1). The same for BA2, that refers to *mechanical domain* which contains links to SN3 and SN4. Furthermore, peer nodes may belong to more than a BA, for example SN4 and SN5 belong to BA2 and BA1 respectively and to the "news" super-peer BA3.

In this paper we propose a novel approach for querying a super-peer within a schema-based super-peer network. We focus on querying a single BA (super-peer) (for querying the SEWASIE network for more than one BA see [10, 11]) and propose a method fully implemented in the SEWASIE project prototype.

The problem we faced is relevant as a BA is a two-level data integrated system then we are going beyond traditional setting in data integration.

We have two different levels of mappings (figure 1.b): The first mapping (*m1*) is at the BA level and maps several GVVs of SINodes to the GVV of the BA; the second mapping (*m2*) is done within an SINode and maps the data sources into the GVV of an SINode.

Halevy et al [12] showed that, in general, the mapping from the data sources to the BA Ontology is not simply the composition of *m1* and *m2*; Fagin et al [13] showed that second order logic is needed to express composition.

In [14, 11] is proved that if *m1* and *m2* are GAV (Global as View) mappings, like in SEWASIE, the mapping is indeed the composition of *m1* and *m2*; this implies that query answering can be carried out in terms of two reformulation steps

1. **Reformulation w.r.t. the BA ontology** (mapping *m1*): this step reformulates the query in terms of the SINodes known by the BA;
2. **Reformulation w.r.t. the SINode ontology** (mapping *m2*): this step reformulates each SINode query obtained in the first step in terms of the data sources known by the SINode;.

This is the algorithm proved to be sound and complete for a two-level data integration system [14].

The paper is organized as follows. Section 1.1 gives an overview of the architecture of the SEWASIE system. In section 2, we introduce a two-level data integration system and in section 3, we define the query reformulation process for this system. In section 3.2 the agent-based prototype for Query Processing in the SEWASIE system is briefly presented. For more detailed description see [15, 11].

1.1 SEWASIE Architecture

The SEWASIE network is an agent-based network developed within the UE IST SEWASIE project, and the overall architecture is shown in figure 2.

A user is able to access the system through a central user interface where (s)he is provided with tools for query composition, for visualizing and monitoring query results, and for communicating with other business partners about search

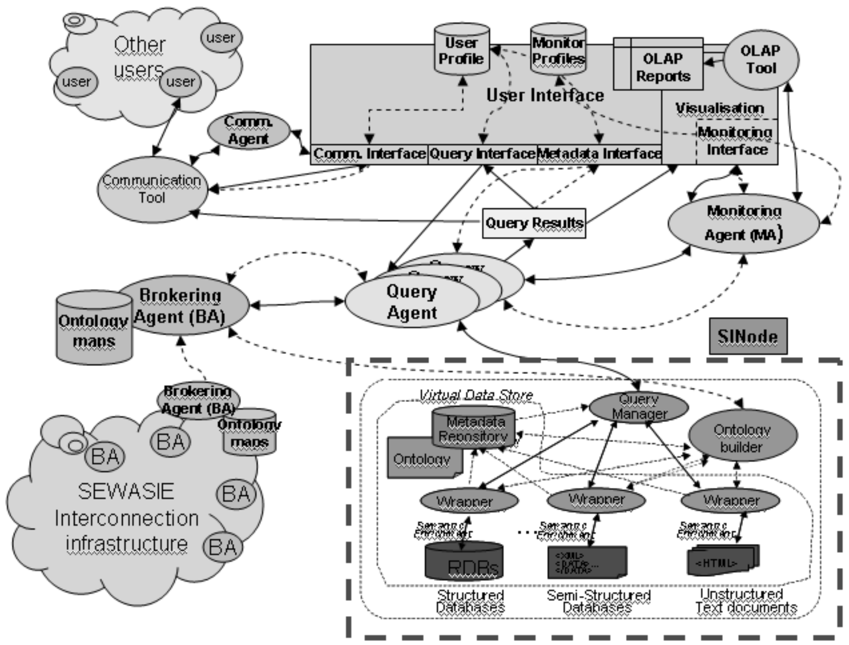


Fig. 2. SEWASIE Architecture

results, e.g. in electronic negotiations. Within a SINode, wrappers are used to extract the data and metadata (local schemas) from the sources. The *Ontology Builder* - based on the MOMIS framework [16, 17], is a semi-automatic tool to create a domain ontology as a Global Virtual View (GVV) which is annotated w.r.t. a lexical ontology (Wordnet [18], Multiwordnet).

Brokering Agents integrate several GVVs from different SINodes into BA Ontology, that is of central importance to the SEWASIE system. On the one hand, the user formulates the queries using this ontology. On the other hand, it is used to guide the Query Agents to the SINodes providing data for a query.

The SEWASIE network can have multiple brokering agents, each one representing a collection of SINodes for a specific domain. Mappings between different brokering agents may be established. A Query Agent receives the queries (expressed in terms of a specific BA ontology) from the user interface, rewrites the query in terms of the GVVs of the SINodes (in cooperation with the brokering agent) and sends the queries to the SINodes. The result is integrated and stored in a result repository, so that it can be used by the various end-user components.

For example, *Monitoring Agents* can be used to store a query result in a permanent repository. The monitoring agent will then execute the query repeatedly, and compare the new results with previous results. The user will be notified if a document has changed that fits her monitoring profile. Furthermore, the monitoring agent can link multidimensional OLAP reports with ontology-based information by maintaining a mapping between OLAP models and ontologies.

Finally, the *Communication Tool* provides the means for ontology-based negotiations. It uses query results, the ontologies of the Brokering Agents, and specific negotiation ontologies as the basis for a negotiation about a business contract. In addition, it uses several agents to support the negotiators in their decision process (e.g. by filter and ranking offers of potential business partners, or by monitoring the available resources of a company).

The SEWASIE consortium is constituted by the University of Modena and Reggio Emilia, the coordinator, which developed the Ontology Builder, the Query Agent and the agent architecture in collaboration with the University of Roma La Sapienza and University of Bolzano respectively. The user interface is a join effort of University of Roma La Sapienza and University of Bolzano.

2 The SEWASIE System

In this section, we describe the two-level data integration system.

An Integration System $IS = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$ is constituted by:

- A Global Virtual View (GVV), which is a schema expressed in ODL_{I^3} [16], a modified version of the *Object Definition Language*[1]. In particular, in the GVV we have *is-a* relationships and both `key` and `foreign key` constraints.
- A set $\mathcal{N}$ of *local sources*; each local source has a *schema* also expressed in ODL_{I^3}.
- A set $\mathcal{M}$ of GAV mapping assertions between GVV and $\mathcal{N}$, where each assertion associates to an element g in GVV a query $q_{\mathcal{N}}$ over the schemas of a set of local sources in $\mathcal{N}$.

  More precisely, for each global class $C \in GVV$ we define:
  1. a (possibly empty) set of local classes, denoted by $L(C)$, belonging to the local sources in $\mathcal{N}$.
  2. a conjunctive query $q_{\mathcal{N}}$ over $L(C)$.

Intuitively, the GVV is the intensional representation of the information provided by the Integration System, whereas the mapping specifies how such an intensional representation relates to the local sources managed by the Integration System in an SINode.

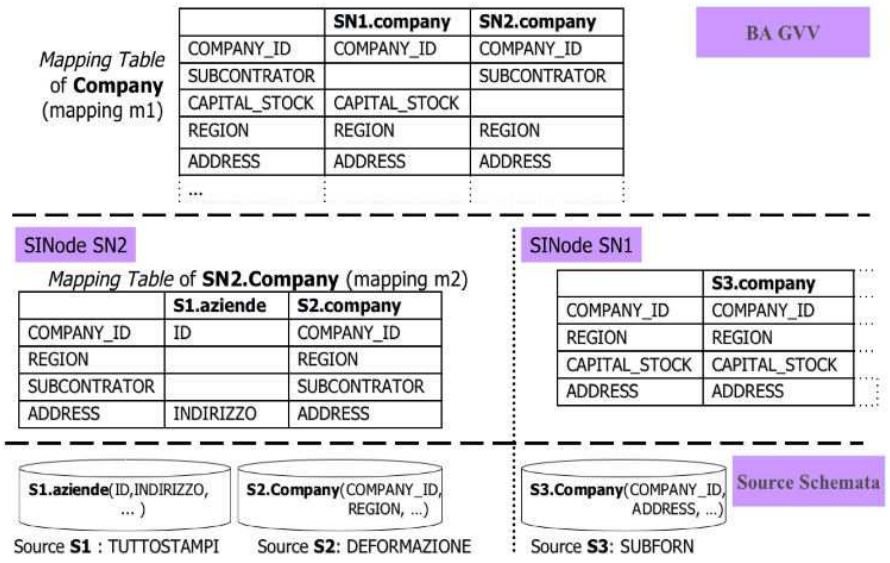
A **SEWASIE system** is constituted by:

- A set of *SINodes* $\mathcal{SN} = \{SN_1, SN_2, \ldots, SN_n\}$, where each SINode is a Integration System $SN = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$ such that $\mathcal{N}$ is a set of data sources.
- A Brokering Agent BA, which is an Integration System $BA = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$ where $\mathcal{N} = \mathcal{SN}$, i.e., the *local* sources of BA are the SINodes.

The semantics of an Integration System, and then of the SEWASIE system, is defined in [19, 11].

In many papers (see [16, 17]) we described the MOMIS/SEWASIE approach for the semi-automatic building of the GVV starting from a set of local sources and giving rise to a Mapping Table (MT) for each global class C of GVV ,

[1] www.service-architecture.com/database/articles/odmg_3_0.html


BA GVV
Mapping Table of Company (mapping m1)
SINode SN2
Mapping Table of SN2.Company (mapping m2)
SINode SN1
S1.aziende(ID,INDIRIZZO, ...)
S2.Company(COMPANY_ID, REGION, ...)
S3.Company(COMPANY_ID, ADDRESS, ...)
Source Schemata
Source S1 : TUTTOSTAMPI
Source S2: DEFORMAZIONE
Source S3: SUBFORN

	SN1.company	SN2.company
COMPANY_ID	COMPANY_ID	COMPANY_ID
SUBCONTRATOR		SUBCONTRATOR
CAPITAL_STOCK	CAPITAL_STOCK	
REGION	REGION	REGION
ADDRESS	ADDRESS	ADDRESS
...		

	S1.aziende	S2.company
COMPANY_ID	ID	COMPANY_ID
REGION		REGION
SUBCONTRATOR		SUBCONTRATOR
ADDRESS	INDIRIZZO	ADDRESS

	S3.company
COMPANY_ID	COMPANY_ID
REGION	REGION
CAPITAL_STOCK	CAPITAL_STOCK
ADDRESS	ADDRESS

Fig. 3. Example of Mapping in the Mechanical domain

whose columns represent the local classes $L(C)$ belonging to C and whose rows represent the global attributes of C. An element $MT[GA][LC]$ represents the set of local attributes of LC which are mapped onto the global attribute GA. As an example, figure 3 shows part of the Mapping Table of the global class `Company` (of a BA-GVV) that groups the local class `Company` of SINode1 and the local class `Company` of SINode2. At the level of a SINode, we have that (we consider SINode2), the global class `SN2.company` is mapped into the local classes `S1.aziende` and `class S2.company` (where SI and S2 are data sources).

In this paper we face and solve a new problem, that is how to define the conjunctive query $q_{\mathcal{N}}$ associated to a global class C. Our approach is the following: starting from the Mapping Table of C, the integration designer, supported by the Ontology Builder graphical interface [20], can implicitly define $q_{\mathcal{N}}$ by:

1. using and extending the Mapping Table with
   – *Data Conversion Functions* from local to global attributes
   – *Join Conditions* among pairs of local classes belonging to C
   – *Resolution Functions* for global attributes to solve data conflicts of local attribute values.
2. using and extending the *Full Disjunction* operator [21], that has been recognized as providing a natural semantics for data merging queries [22].

Data Conversion Functions
The designer can define how local attributes are mapped onto the global attribute GA by means of *Data Conversion Functions*: for each not null element $MT[GA][L]$ we define a *Data Conversion Function*, denoted by $MTF[GA][L]$,

which represents how the local attributes of L are mapped into the global attribute GA. $MTF[GA][L]$ is a function that mut be *executable/supported* by the local source of the class L. For example, for relational sources, $MTF[GA][L]$ is an SQL value expression; the following defaults hold: if $MT[GA][L] = LA$ then $MTF[GA][L] = LA$ and, if $MT[GA][L]$ contains more than one string attribute, then $MTF[GA][L]$ is the string concatenation.
$T(L)$ denotes L transformed by the Data Conversion Function; the schema of $T(L)$ is composed of the global attributes GA such that $MT[GA][L]$ is not null.

Join Conditions
Merging data from different sources requires different instantiations of the same real world object to be identified; this process is called *object identification* [23]. The topic of *object identification* is currently a very active research area with significant contributions both from the artificial intelligence [24] and database [25, 26] communities.

To identify instances of the same object and fuse them we introduce *Join Conditions* among pairs of local classes belonging to the same global class. Given two local classes $L1$ and $L2$ belonging to C, a Join Condition between $L1$ and $L2$, denoted with $JC(L1, L2)$, is an expression over $L1.A_i$ and $L2.A_j$ where A_i (A_j) are global attributes with a not null mapping in $L1$ ($L2$). As an example, for `BA-GVV.Company` the designer can define JC(SN1.Company,SN1.Company) :
SN1.Company.COMPANY_ID = SN2.Company.COMPANY_ID.

Resolution Functions
The fusion of data coming from different sources taking into account the problem of inconsistent information among sources is a hot research topic [27–29, 23, 30]. In the context of MOMIS/SEWASIE we adopt the *Resolution Function* proposed in [23]. A Resolution Function for solving data conflits may be defined for each global attribute mapping onto local attributes coming from more than one local source.

Homogeneous Attributes : If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*; this is the default in our system. Of course, for homogeneous attributes resolution functions are not necessary. A global attribute mapped onto only one source is a particular case of an homogeneous attribute.

As an example, in `BA-GVV.Company` we define all the global attributes as Homogeneous Attributes except for `Address` where we used a precedence function: `SN1.Company.ADDRESS` has a higher precedence than `SN2.Company.ADDRESS`.

Full Disjunction
We want to define $q_{\mathcal{N}}$ in such a way that it contains a unique tuple resulting from the merge of all the different tuples representing the same real world object. This problem is related to that of computing the natural outer-join of many

relations in a way that preserves all possible connections among facts [22]. Such a computation has been termed as *Full Disjunction* (*FD*) by Galindo Legaria [21].

In our context: given a global class C composed of $L1, L2, ..., Ln$, we consider $FD(T(L1), T(L2), \ldots, T(Ln))$, computed on the basis of the *Join Conditions.*

The problem is how to compute FD. With two classes, *FD* corresponds to the full (outer) join: $FD(T(L1), T(L2)) = T(L1)$ `full join` $T(L2)$ `on` $(JC(L1, L2))$.

In [22] was demonstrated that there is a natural outer-join sequence producing *FD* if and only if the set of relation schemes forms a connected, acyclic hypergraph. In our context, a Global Class C with more than 2 local classes is a cyclic hypergraph, then we cannot use the algorithms proposed in [22]; the computation of *FD* is performed as follows. We assume that: (1) each L contains a key, (2) all the *join conditions* are on key attributes, and (3) all the join attributes are mapped into the same set of global attribute, say K. Then, it can be demonstrated that: (1) K is a key of $\mathcal{C}$, and (2) *FD* can be computed by means of the following expression (called *FDExpr*):

```
(T(L1) full join T(L2) on JC(L1,L2))
         full join T(L3) on (JC(L1,L3) OR JC(L2,L3))
          ...
         full join T(Ln)  on (JC(L1,Ln) OR JC(L2,Ln) OR ... OR JC(Ln-1,Ln))
```

Finally, $q_{\mathcal{N}}$ is obtained by applying Resolution Functions to the attributes resulting from *FDExpr*: for a global attribute GA we apply the related Resolution Function to $T(L1).GA, T(L2).GA, \ldots, T(Lk).GA$; This query $q_{\mathcal{N}}$ is called *FDQuery.*

3 Query Reformulation in the SEWASIE system

The query reformulation takes into account two different levels of mappings (figure 1.b): in [14, 11] is proved that if *m1* and *m2* are GAV mappings, the mapping is indeed the composition of *m1* and *m2*; this implies that query answering can be carried out in terms of two reformulation steps: **1. Reformulation w.r.t. the BA ontology** and **2. Reformulation w.r.t. the SINode ontology**. These reformulation steps are similar and are defined by considering the reformulation process for an Integration System $IS = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$, that is constituted by:

1. **Query expansion**: the query posed in terms of the GVV is expanded to take into account the explicit and implicit constraints in the GVV: all constraints in the GVV are compiled in the expansion, so that the expanded query can be processed by ignoring constraints. Then, the atoms in the expanded query are extracted from the expanded query.
2. **Query unfolding**: the atoms in the expanded query are unfolded by taking into account the mappings $\mathcal{M}$ between the GVV and the local sources in $\mathcal{N}$.

The algorithm for Query expansion is reported in [14, 11]; its output is the expanded query (called *EXPQuery*) and its atoms (called *EXPAtoms*); *EXPQuery* is an union of conjunctive queries on GVV; an *EXPAtom* is a *Single Class Query* on a Global Class of the GVV.

In the following we will discuss the unfolding process of an *EXPAtom* by taking into account the new approach to define $q_{\mathcal{N}}$ of the previous section.

3.1 Query unfolding

We explain the method by considering the BA level, i.e. the BA ontology. Given a global class C related to the local classes $L1, L2, \ldots Ln$, we consider a Single Global Query Q over C:
`Q = select <Q_select-list> from C where <Q_condition>`
`<Q_condition>` is a Boolean expression of positive atomic constraints: (GA1 op value) or (GA1 op GA2), where GA1 and GA2 are attributes of C.

As an example, we consider the following query (denoted by `expatom`):

```
expatom: SELECT NAME,CAPITAL_STOCK,REGION,ADDRESS,SUBCONTRACTOR
         FROM company
         WHERE CAPITAL_STOCK>50 AND REGION LIKE 'VENETO' AND SUBCONTRACTOR LIKE 'yes'
```

The output of the query unfolding process is

1. a set of Single Class Queries over the SINodes GVVs (FDAtoms):

   ```
   FDAtom = select <select-list> from SINode.C where <condition>
   ```

   where `C` is a Global Class of the `SINode-GVV`.
2. the *FDQuery* which computes the Full Disjunction of the FDAtoms
3. the resolution functions of the attributes in `<select-list>`

The query unfolding process is made up of the following steps:

1. Atomic constraint mapping: In this step, each atomic constraint of Q is rewritten into one that can be supported by the local class.

The atomic constraint mapping is performed on the basis of the mapping functions defined in the Mapping Table. Moreover, the atomic constraint mapping depends on the definition of the Resolution Functions for global attributes; for example, if the numerical global attribute GA is mapped onto L1 and L2, and we define AVG function as resolution function, the constraint (GA = value) cannot be pushed at the local sources, because of the AVG function has to be calculated at a global level, the constraint may be globally true but locally false. In this case, the constraint is mapped as true in both the local sources. On the other hand, if GA is an homogeneous attribute the constraint can be pushed at the local sources. For example, an atomic constraint (GA op `value`) is mapped onto the local class L as follows:

$$\begin{array}{l} (MTF[GA][L] \; op \; \texttt{value}) \text{ if } MT[GA][L] \text{ is not null and} \\ \qquad\qquad\qquad\qquad\qquad \text{the } op \text{ operator is supported into } L \\ \qquad\qquad\qquad \mathit{true} \text{ otherwise} \end{array}$$

2. Select-list computation : The select-list of a FDAtom over the local class L is computed by considering the union of

1. the attributes in `<Q_select-list>` with a not null mapping in L
2. the set of attributes used to express the join conditions for L
3. the global attributes in `<Q_condition>` with a not null mapping in L

The set of global attributes is transformed in the corresponding set of local attributes on the basis of the Mapping Table.

As an example, the set of **FDAtoms** for `expatom` is :

```
FDATOM1 =  SELECT COMPANY_ID, NAME, REGION, ADDRESS, CAPITAL_STOCK
 FROM SN1.company
 WHERE ((CAPITAL_STOCK) > (50) and (REGION) like ('VENETO'))

FDATOM2 = SELECT  COMPANY_ID, NAME, REGION, ADDRESS, SUBCONTRACTOR
 FROM SN2.company
 WHERE ((REGION) like ('VENETO') and (SUBCONTRACTOR) like ('yes'))
```

The *FDExpr* which computes the FD of `FDAtom1` and `FDAtom2` is:

```
   FDATOM1 full join FDATOM2 on (FDATOM1.COMPANY_ID=FDATOM2.COMPANY_ID)
```

The unfolded query is then obtained by applying to each query attribute of *FDExpr*, the related Resolution Function:

– for Homogeneous Attributes (e.g. `REGION`) we can take one of the related values (indifferently `FDATOM1.REGION` or `FDATOM2.REGION`);
– for non Homogeneous Attributes (e.g. `ADDRESS`) we apply the related Resolution Function (in this case the precedence function).

After the query reformulation process, we need to consider query processing techniques to evaluate queries over our two-level data integration system. This was not a focus of our present investigation and of the SEWASIE project; at present, we have just implemented a "naive approach" in an agent-based prototype, that will be described in the next section. Techniques for adaptive query processing [31] are well suited for our context.

3.2 An Agent-based prototype for Query Processing

Figure 4 shows the functional architecture of the system prototype for Query Management. The coordination of query processing is performed by the **Query Agent**, which accepts the query from the Query Tool Interface, interacts with a BA and its underlying SINode Agents, and returns the result as a materialized view in the SEWASIE_DB.

Playmaker : performs the reformulation of the query w.r.t. the BA ontology. It has two components: the **Expander**, which performs the Query expansion, and the **Unfolder**, which performs the query unfolding. Once the execution of the PlayMaker is completed, the output of the Play Maker computation is sent from the BA to the QA with a single message.

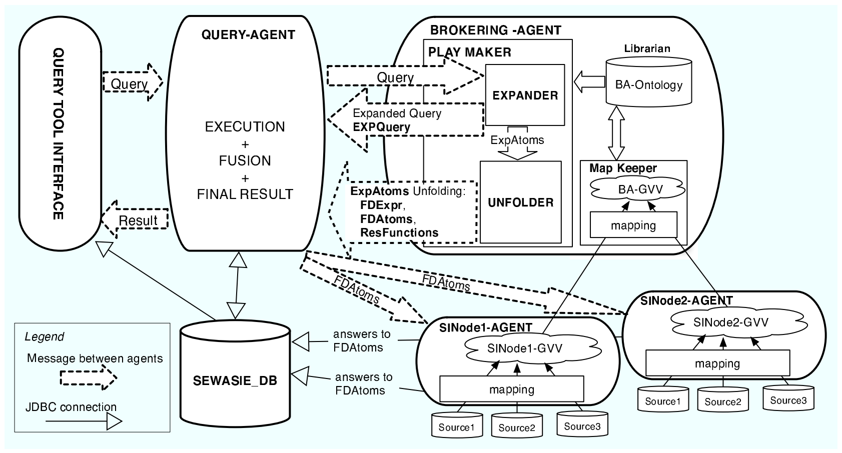


Fig. 4. Functional Architecture

Query Agent : it performs the following 3 steps:

1. **Execution :** for each FDAtom (Parallel Execution)
   - **INPUT**: FDAtom
   - **MESSAGES**: from QA to an SINode Agent
   - **OUTPUT**: a table storing the FDAtom result in the SEWASIE_DB
2. **Fusion :** For each EXPAtom (Parallel Execution):
   - **INPUT**: FDAtoms, FDExpr, Resolution Functions
     (a) Execution of FDExpr (Full Disjunction of the FDAtoms)
     (b) Application of the Resolution Functions on the result of (a)
   - **OUTPUT**: a view storing the EXPAtom result in the SEWASIE_DB
3. **Final result**.
   - **INPUT**: Output of the FUSION step
     (a) Execution of the Expanded Query
   - **OUTPUT**: Final Query result view stored in the SEWASIE_DB

A this point, the Query Agent sends a message to the Query Tool Interface with the name of the Final Query result.

SINode Agent : One of the modules of the SINode Agent, the **SINode Query Manager**, executes queries on the SINode GVV, with a query processing similar to the one explained at the BA level.

4 Conclusion and Future Work

Future work will be devoted to investigate efficient query processing techniques to evaluate queries over two-level data integration systems. Furthermore we will investigate efficient query techniques for querying the super-peer network.

The above issues will be the goal of our research group within the running Italian MIUR founded project WISDOM (http://dbgroup.unimo.it/wisdom-unimo), which is coordinated by our group.

References

1. Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: The chatty web: emergent semantics through gossiping. In: WWW. (2003) 197–206
2. Halevy, A.Y., Ives, Z.G., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I.: The piazza peer data management system. IEEE Trans. Knowl. Data Eng. **16** (2004) 787–798
3. Bernstein, P.A., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data management for peer-to-peer computing : A vision. In: WebDB. (2002) 89–94
4. Löser, A., Siberski, W., Wolpers, M., Nejdl, W.: Information integration in schema-based peer-to-peer networks. In Eder, J., Missikoff, M., eds.: CAiSE. Volume 2681 of Lecture Notes in Computer Science., Springer (2003) 258–272
5. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36** (2004) 335–371
6. Broekstra, J.e.a.: A metadata model for semantics based peer-to-peer systems. In: Proc. of the 1st WWW Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID 2003), Budapest, Hungary (2003)
7. Castano, S., Ferrara, A., Montanelli, S., Pagani, E., Rossi, G.: Ontology-addressable contents in p2p networks. In: Proc. of the 1st WWW Int. Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGRID 2003), Budapest, Hungary (2003) http://www.isi.edu/ stefan/SemPGRID/proceedings/proceedings.pdf.
8. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In Dayal, U., Ramamritham, K., Vijayaraman, T.M., eds.: ICDE, IEEE Computer Society (2003) 49–
9. Löser, A., Naumann, F., Siberski, W., Nejdl, W., Thaden, U.: Semantic overlay clusters within super-peer networks. In Aberer, K., Kalogeraki, V., Koubarakis, M., eds.: DBISP2P. Volume 2944 of Lecture Notes in Computer Science., Springer (2003) 33–47
10. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. [32] 241–251
11. Beneventano, D., Lenzerini, M.: Final release of the system prototype for query management. Sewasie, Deliverable D.3.5, Final Version (2005) http://www.dbgroup.unimo.it/pubs.html.
12. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In: VLDB. (2003) 572–583
13. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: Second-order dependencies to the rescue. [32] 83–94
14. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: What to ask to a peer: Ontolgoy-based query reformulation. In Dubois, D., Welty, C.A., Williams, M.A., eds.: KR, AAAI Press (2004) 469–478
15. Bergamaschi, S., P. Fillottrani, G.G.: The sewasie multi-agent system. In: Proc. of the 3rd Int. Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004), New York City, USA July 19-20, 2004. (2004)
16. Bergamaschi, S., Castano, S., Vincini, M., Beneventano, D.: Semantic integration of heterogeneous information sources. Data Knowl. Eng. **36** (2001) 215–249
17. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Synthesizing an integrated ontology. IEEE Internet Computing **7** (2003) 42–51

18. Miller, A.: WordNet: A Lexical Database for English. Communications of the ACM **38** (1995) 39–41
19. Cal, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Data integration under integrity constraints. Information Systems **29** (2004) 147–163
20. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Building an integrated ontology within sewasie system. In Cruz, I.F., Kashyap, V., Decker, S., Eckstein, R., eds.: SWDB. (2003) 91–107
21. Galindo-Legaria, C.A.: Outerjoins as disjunctions. In Snodgrass, R.T., Winslett, M., eds.: SIGMOD Conference, ACM Press (1994) 348–358
22. Rajaraman, A., Ullman, J.D.: Integrating information by outerjoins and full disjunctions. In: PODS, ACM Press (1996) 238–248
23. Naumann, F., Häussler, M.: Declarative data merging with conflict resolution. In Fisher, C., Davidson, B.N., eds.: IQ, MIT (2002) 212–224
24. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Syst. **26** (2001) 607–633
25. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB. (2002) 586–597
26. Chaudhuri, S., Ganjam, K., Ganti, V., Motwani, R.: Robust and efficient fuzzy match for online data cleaning. In Halevy, A.Y., Ives, Z.G., Doan, A., eds.: SIGMOD Conference, ACM (2003) 313–324
27. Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tackling inconsistencies in data integration through source preferences. In Naumann, F., Scannapieco, M., eds.: IQIS, ACM (2004) 27–34
28. Bertossi, L.E., Chomicki, J.: Query answering in inconsistent databases. In Chomicki, J., van der Meyden, R., Saake, G., eds.: Logics for Emerging Applications of Databases, Springer (2003) 43–83
29. Greco, G., Greco, S., Zumpano, E.: A logical framework for querying and repairing inconsistent databases. IEEE Trans. Knowl. Data Eng. **15** (2003) 1389–1408
30. Lin, J., Mendelzon, A.O.: Merging databases under constraints. Int. J. Cooperative Inf. Syst. **7** (1998) 55–76
31. Ives, Z.G., Florescu, D., Friedman, M., Levy, A.Y., Weld, D.S.: An adaptive query execution system for data integration. In Delis, A., Faloutsos, C., Ghandeharizadeh, S., eds.: SIGMOD Conference, ACM Press (1999) 299–310
32. Deutsch, A., ed.: Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France. In Deutsch, A., ed.: PODS, ACM (2004)