# Towards OLAP Query Reformulation in Peer-to-Peer Data Warehousing

Matteo Golfarelli
DEIS - Univ. of Bologna
V.le Risorgimento 2
Bologna, Italy
matteo.golfarelli@unibo.it

Federica Mandreoli*
DII - Univ. of Modena
Via Vignolese, 905/b
Modena, Italy
federica.mandreoli@unimo.it

Wilma Penzo*
DEIS - Univ. of Bologna
V.le Risorgimento, 2
Bologna, Italy
wilma.penzo@unibo.it

Stefano Rizzi
DEIS - Univ. of Bologna
V.le Risorgimento, 2
Bologna, Italy
stefano.rizzi@unibo.it

Elisa Turricchia
DEIS - Univ. of Bologna
V.le Risorgimento, 2
Bologna, Italy
elisa.turricchia2@unibo.it

## ABSTRACT

Inter-business collaborative contexts prefigure a distributed scenario where companies organize and coordinate themselves to develop common and shared opportunities. Traditional business intelligence systems do not provide support to this end. Peer Data Management Systems (PDMSs) have been proposed as architectures to support sharing of operational data across networks of peers while guaranteeing peers' autonomy, based on semantic mappings that mediate between the heterogeneous schemata exposed by peers. In line with the PDMS infrastructure, in this paper we envision a peer-to-peer data warehousing architecture based on a network of heterogeneous peers, each exposing query answering functionalities aimed at sharing business information. To enhance the decision making process, an OLAP query expressed on a peer needs be properly reformulated on the other peers. In this direction, we present a language for the definition of mappings between the multidimensional schemata of peers, and we introduce a query reformulation framework that relies on the translation of these mappings towards relational schemata. Finally, we sketch the query reformulation algorithm by outlining the reformulation steps of typical OLAP queries.

## Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Types of Systems—*decision support*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems*; H.2.5 [**Database Management**]: Heterogeneous Databases

## General Terms

Algorithms

*Research affiliation: IEIIT - CNR, Bologna, Italy


*DOLAP'10,* October 30, 2010, Toronto, Ontario, Canada.


## 1. INTRODUCTION

Business intelligence (BI) transformed the role of computer science in companies from a technology for passively storing data into a discipline for timely detecting key business factors and effectively solving strategic decisional problems. However, in the current changeable and unpredictable market scenarios, the needs of decision makers are rapidly evolving as well. To meet the new, more sophisticated user needs, a new generation of BI systems (often labeled as *BI 2.0*) has been emerging during the last few years.

One of the key features of BI 2.0 is the ability to become pervasive and extend the decision-making process beyond the boundaries of a single company [12]. Users need to access information anywhere it can be found, by locating it through a semantic process and performing integration on the fly. This is particularly relevant in inter-business collaborative contexts where companies organize and coordinate themselves to share opportunities, respecting their own autonomy and heterogeneity but pursuing a common goal. In such a complex and distributed business scenario, traditional BI systems —that were born to support stand-alone decision-making— are no longer sufficient to maximize the effectiveness of monitoring and decision making processes. Accessing local information is no more enough, users need to transparently and uniformly access information scattered across several heterogeneous BI platforms [8].

Peer Data Management Systems (PDMSs) have been proposed in the literature as architectures to support sharing of operational data across networks of peers while guaranteeing peers' autonomy, based on interlinked semantic mappings that mediate between the heterogeneous schemata exposed by peers [7]. In line with the PDMS infrastructure, we envision a peer-to-peer data warehousing architecture called *Business Intelligence Network* (BIN) and sketched in Figure 1. A BIN is based on a dynamic and collaborative network of heterogeneous and autonomous peers; each peer is equipped with an independent data warehouse system, that relies on a local multidimensional schema to represent the peer's view

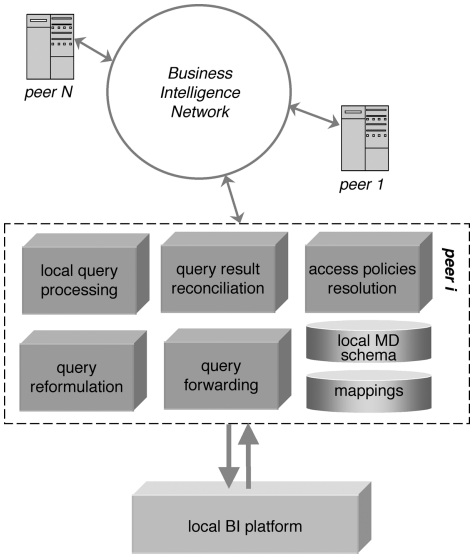


**Figure 1: Envisioned architecture for a BIN**

of the business and exposes query answering functionalities aimed at sharing business information.

To enhance the decision making process, BIN users transparently access business information distributed over the network. A typical user interaction is the following: (1) A user formulates an OLAP query $q$ by accessing the local multidimensional schema of her peer. (2) Query $q$ is forwarded to the network and reformulated on the other peers in terms of their own multidimensional schemata. (3) Each involved peer locally processes the reformulated query and returns its results to the querying peer. (4) Finally, the results are integrated and returned to the user based on the lexicon used to formulate $q$.

At step (2), an OLAP query expressed on a peer needs be properly reformulated on the other peers, which is a challenging task due to the presence of aggregation and to the possibility of having information represented at different granularities and under different perspectives in each peer. In this paper we focus on this task. The original contributions we give are: (a) We present a language for the definition of semantic mappings between the schemata of peers, using predicates that are specifically tailored for the multidimensional model (Section 4). To overcome possible differences in data formats, mappings can be associated with encoding functions. (b) We introduce a query reformulation framework that relies on the translation of mappings towards the underlying relational schemata (Section 5). In particular, we will use standard star schemata for simplicity. (c) We sketch the query reformulation algorithm by outlining the reformulation steps of typical OLAP queries (Section 6).

## 2. RELATED WORKS

Decentralized sharing of data between autonomous sources has been deeply studied as an evolution of mediator systems in the data integration field [7]. In this context, declarative schema-mapping languages are used to locally specify the relationships between neighboring peers' schemata. In [13], the authors present a structural characterization of schema-mapping languages and show the basic tasks that every language ought to support. These languages have been conceived for OLTP databases, so they do not accommodate the peculiar characteristics of the multidimensional model. Moreover, while in the literature the problems of differences in data formats have only marginally been considered, declaring useful mappings in the OLAP context necessarily requires also the level of instances to be taken into account.

Only a few works are specifically focused on strategies for data warehouse integration and federation. Indeed, in this context, problems related to data heterogeneity are usually solved by ETL processes that read data from several data sources and load them in a single repository. While this centralized architecture may fit the needs of stand-alone companies, it is hardly feasible in the context of a BIN, where the dynamic nature of the business network, together with the independence and autonomy of peers, call for more sophisticated solutions. See [1] for a discussion of the benefits of a peer-to-peer architecture for data warehousing.

In the context of a federated data warehouse architecture, [14] describes two methods for integrating dimensions belonging to different data marts and provide a set of rules to check their compatibility. The problem of how to define mappings between concepts is not considered. The work proposed in [2] present a complete algorithm for matching multidimensional structures, with a strong emphasis on the process of calculating similarity between complex concepts. However, the data-related aspects —that could enrich the matching definitions— are not considered, and no model is provided to formalize the mapping predicates.

Another work centered on interoperability issues among heterogeneous data warehouses is the one by [11], that emphasizes the importance of a semantic layer to enable communication among different entities. This approach supports the exchange of business calculation definitions and allows for their automatic linking to specific data warehouses through semantic reasoning. The approach is flexible because it could be implemented by adopting different formalisms for each layer; however, the work proposes specific techniques to deal with measures only, so it cannot be used to completely solve a typical aggregate query.

Finally, in the distributed data warehouse context, in [10] the authors propose a peer-to-peer architecture for supporting OLAP queries focusing on the definition of a caching model to improve query rewriting. They also define adaptive techniques that dynamically reconfigure the network structure in order to minimize the query cost. In [9] a hybrid approach between the centralized and the full-federation configuration is described, suggesting the use of a federation server to store aggregated data from different remote sources. The goal is to improve the performance of complex OLAP queries exploiting the correlation among different queries.

## 3. PRELIMINARIES

In this section we introduce a basic formal setting to manipulate and query multidimensional data, and we propose a running example. In particular, our formalization of md-schemata is an extension of the one used in [4]. At this stage, to get rid of summarizability problems, we only consider distributive operators for measure aggregation.

DEFINITION 1 (MD-SCHEMA). *An* md-schema *is a triple* $\mathcal{M} = \langle A, H, M \rangle$ *where:*

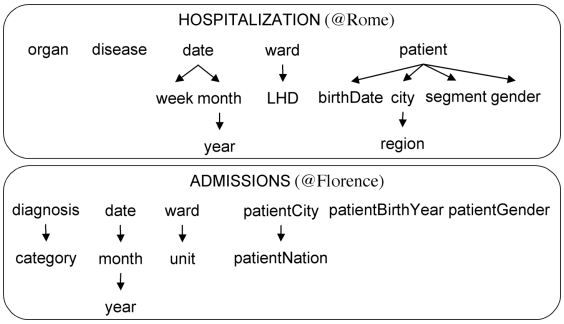


**Figure 2: Roll-up orders for the hierarchies in the HOSPITALIZATION and ADMISSIONS md-schemata (LHD stands for local health department)**

- $A = \{a_1, \ldots, a_p\}$ *is a finite set of* attributes*, each defined on a categorical domain* $Dom(a_i)$*;*
- $H = \{h_1, \ldots, h_n\}$ *is a finite set of* hierarchies*, each characterized by (1) a subset* $Attr(h_i) \subseteq A$ *of attributes (such that the* $Attr(h_i)$*'s for* $i = 1, \ldots, n$ *define a partition of A) and (2) a* roll-up *tree-structured partial order* $\succeq_{h_i}$ *of* $Attr(h_i)$*;*
- *a finite set of* measures $M = \{m_1, \ldots, m_l\}$*, each defined on a numerical domain* $Dom(m_i)$ *and aggregable through one distributive aggregation operator* $Agg(m_i)$*.*

*For each hierarchy* $h_i$*, the root attribute of the order is called* dimension*, denoted by* $DIM_i$*, and determines the finest aggregation level for the hierarchy.*

A group-by set is a subset of $A$, and defines a possible way to aggregate data.

Definition 2 (Group-by Set). *Given* $\mathcal{M} = \langle A, H, M \rangle$*, a* group-by set *of* $\mathcal{M}$ *is a subset of attributes* $G \subseteq A$*.*

Example 1. *In the working example we adopt in this paper, a set of local health-care departments participate in a collaborative network to integrate their data about admissions so as to enable more effective analysis of epidemics and health-care costs by the Ministry. For simplicity we will focus on two peers: the first, located in Rome, hosting data on hospitalizations at patient-level detail; the second, located in Florence, hosting data on admissions grouped by patient gender, residence city, and birth year. The underlying md-schema for the Rome peer is called* HOSPITALIZATION *and includes the additive measures* cost *and* durationOfStay*; the one for the Florence peer is called* ADMISSIONS *and includes the additive measures* totStayCost*,* totExamCost*,* totLength*, and* numAdmissions*. The roll-up orders for the two md-schemata are shown in Figure 2. Assuming that each hierarchy is named after its finest-level attribute, but capitalized, it is* $DIM_{\mathsf{Patient}} = \mathsf{patient}$ *and* $\mathsf{city} \succeq_{\mathsf{Patient}} \mathsf{region}$*. An example of group-by set of* HOSPITALIZATION *is* $G = \{\mathsf{gender}, \mathsf{region}, \mathsf{month}\}$*.*

The expressiveness we assume for OLAP queries is that of GPSJ (Generalized Projection-Selection-Join) queries [5]. Without loss of generality, we will only consider atomic selection predicates; besides, we will assume that each measure $m$ can only be aggregated using its own aggregation operator, $Agg(m)$. To avoid getting burdened with the details of a specific multidimensional query language, such as MDX, we will express queries using an abstract syntax.

Definition 3 (OLAP query). *An* OLAP query *is a 5-tuple* $q = \langle \mathcal{M}, G, m, p, \alpha \rangle$ *where* $\mathcal{M} = \langle A, H, M \rangle$ *is an md-schema,* $G$ *is a group-by set of* $\mathcal{M}$*,* $m \in M$ *is a measure,* $p$ *is an optional Boolean predicate involving attributes in* $A$ *and/or* $m$*,* $\alpha = Agg(m)$ *is the aggregation operator used to aggregate* $m$*.*

Example 2. *The query, expressed at the Rome peer, that computes the total hospitalization cost of female patients for each region and year is formalized as* $q = \langle$HOSPITALIZATION, $\{\mathsf{region}, \mathsf{year}\}$, cost, $(\mathsf{gender} = \text{'}F\text{'})$, sum$\rangle$*.*

# 4. MAPPING LANGUAGE

In this section we describe the language we devised for the definition of semantic mappings between the md-schemata of peers. As mentioned in the Introduction, these mappings play a key role in a BIN because, as we will show in Section 5, they enable query reformulation.

Let two peers in a collaborative BI network be given. The language we propose to express how the local md-schema $t$ of the target peer maps onto the local md-schema $s$ of the source peer includes five *mapping predicates*, namely same, equi-level, roll-up, drill-down, and related, that will be discussed in detail below. In general, a mapping establishes a semantic relationship from an ordered list of concepts (either measures or attributes) of $t$, $c = \langle a_1, \ldots, a_j \rangle$, to an ordered list of concepts of $s$, $d = \langle b_1, \ldots, b_k \rangle$, and enables a query formulated on $t$ to be (exactly or approximately) reformulated on $s$. Optionally, a mapping can be associated with an encoding function that specifies how values of $c$ can be obtained from values of $d$. If this function is available, it is used during query reformulation and data integration to return more query-compliant results to users. In this paper, mappings on measures (same predicate) take a simpler form (exactly one measure on the left side) to avoid incorrect reformulations due to a wrong use of aggregation operators.

- same predicate: $m$ $\mathsf{same}_f$ $d$, where $d$ only includes measures. This mapping predicate is used to state that measure $m$ in $t$ has the same semantics as a set of measures in $s$. If knowledge is available about how values of $m$ can be derived from values of $d$, it can be expressed by an encoding function $f : Dom(d) \rightarrow Dom(m)$ (where $Dom(d) = Dom(b_1) \times \ldots \times Dom(b_k)$). The semantics of this function is that, whenever $m$ is asked in a query on $t$, it can safely be rewritten as $f(d)$ on $s$.
- equi-level predicate: $c$ $\mathsf{equi\text{-}level}_f$ $d$, where $c$ and $d$ only include attributes. This predicate is used to state that a set of attributes in $t$ has the same semantics and granularity as a set of attributes in $s$. If knowledge is available about a transcoding between $c$ and $d$, it can be expressed by an injective encoding function $f : Dom(d) \rightarrow Dom(c)$ that establishes a one-to-one total relation between $c$ and $d$, and is used to integrate data returned by the source and target peers.
- roll-up predicate: $c$ $\mathsf{roll\text{-}up}_f$ $d$. This predicate states that a set $c$ of attributes in $t$ is a roll-up of (i.e., it aggregates) a set $d$ of attributes in $s$. If knowledge

| | |
|---|---|
| $\omega_1$ | ‹cost› same ‹totalStayCost, totExamCost› |
| $\omega_2$ | ‹durationOfStay› same ‹totLength› |
| $\omega_3$ | ‹LHD› roll-up ‹unit› |
| $\omega_4$ | ‹ward› equi-level ‹ward› |
| $\omega_5$ | ‹year› equi-level ‹year› |
| $\omega_6$ | ‹month› equi-level ‹month› |
| $\omega_7$ | ‹date› equi-level ‹date› |
| $\omega_8$ | ‹week› roll-up ‹date› |
| $\omega_9$ | ‹disease, organ› equi-level ‹diagnosis› |
| $\omega_{10}$ | ‹patient› drill-down ‹patientGender, patientCity, patientBirthYear› |
| $\omega_{11}$ | ‹gender› equi-level ‹patientGender› |
| $\omega_{12}$ | ‹segment› related ‹patientGender, patientCity, patientBirthYear› |
| $\omega_{13}$ | ‹birthDate› drill-down ‹patientBirthYear› |
| $\omega_{14}$ | ‹city› equi-level ‹patientCity› |
| $\omega_{15}$ | ‹region› roll-up ‹patientCity› |

**Figure 3: Complete mapping from Rome (target peer) to Florence (source peer)**

is available about how to roll-up values of $d$ to values of $c$, it can be expressed by a non-injective encoding function $f : Dom(d) \rightarrow Dom(c)$ that establishes a one-to-many relation between $c$ and $d$, and is used to aggregate data returned by the source peer and integrate them with data returned by the target peer.

- drill-down predicate: $c$ drill-down$_F$ $d$. This predicate is used to state that $c$ is a drill-down of (i.e., it disaggregates) $d$. If knowledge is available about how to drill-down values of $d$ to values of $c$, it can be expressed by the injective encoding function $F : Dom(d) \rightarrow 2^{Dom(c)}$ that returns sets of values of $c$, thus establishing a many-to-one relation between $c$ and $d$. The function $F$ cannot be used to integrate data returned by $t$ and $s$ because this would require disaggregating data returned by $s$, which obviously cannot be done univocally; however, it can be used to empower the presentation of the results given to users, for instance by emphasizing the values of $c$ each value of $d$ drills down to.

- related predicate: $c$ related$_F$ $d$. This predicate is used to state that $c$ has a many-to-many relationship with $d$. The function $F : Dom(d) \rightarrow 2^{Dom(c)}$ is non-injective and establishes a many-to-many relation between $c$ and $d$. Like in the previous case, though $F$ cannot be used neither for query rewriting nor for result integration, it could be used to improve result presentation.

EXAMPLE 3. *The complete set of mappings for our healthcare example is reported in Figure 3. For instance, the predicate* $\langle \mathsf{cost} \rangle$ same$_f$ $\langle \mathsf{totStayCost}, \mathsf{totExamCost} \rangle$ *with*

$$f(\langle \mathsf{totStayCost}, \mathsf{totExamCost} \rangle) = \mathsf{totStayCost} + \mathsf{totExamCost}$$

*states that measure* cost *in Rome can be derived by summing* totStayCost *and* totExamCost *in Florence. On the other hand, the predicate* $\langle \mathsf{disease}, \mathsf{organ} \rangle$ equi-level$_f$ $\langle \mathsf{diagnosis} \rangle$ *with*

$$f(\langle \mathsf{diagnosis} \rangle) = \langle \mathsf{disease} : \mathit{substring}(\mathsf{diagnosis}, 1, 20), \mathsf{organ} : \mathit{substring}(\mathsf{diagnosis}, 21, 40) \rangle$$

*states that the diagnosis codes used in Florence are obtained by concatenating the fixed-length disease and organ codes used in Rome. Finally,* $\langle \mathsf{week} \rangle$ roll-up$_f$ $\langle \mathsf{date} \rangle$*, with* $f(\langle \mathsf{date} \rangle) = \langle \mathsf{week} : \mathit{weekOf}(\mathsf{date}) \rangle$*, states that weeks are an aggregation of dates.*

# 5. A REFORMULATION FRAMEWORK

In the BIN architecture, OLAP queries are formulated on a peer schema and answers can come from any other peer which is connected to the queried peer through a chain of mappings. The key step to this end is *reformulating* a peer's query over its immediate neighbors, then over their immediate neighbors, and so on. More precisely, reformulation takes in input an OLAP query on a *target* schema $t$ and the mappings between $t$ and the schema of one of its neighbors, the *source* schema $s$, and it outputs an OLAP query that refers only to $s$.

In our proposal, queries are reformulated by means of the underlying relational (star) schema.

Indeed, much research work has been done in the context of OLTP databases, mostly with regard to mediator systems for data integration as well as in the field of distributed semantic data sharing systems [6]. Research in the data integration area has provided rich and well-understood schema mediation languages. The two commonly used formalisms are the *global-as-view* (GAV) approach, in which the mediated schema is defined as a set of views over the data sources, and the *local-as-view* (LAV) approach, in which the contents of data sources are described as views over the mediated schema. In the context of PDMSs both approaches are possibly used, since each peer can serve as both a data source and a mediator.

Schema mapping languages are used for specifying relationships between schemata and are typically based on logical formalisms, according to a set of formulas called *source-to-target tuple generating dependencies* (s-t tgd's) [13]. Informally, s-t tgd's assert that if a pattern of facts appears in the source, then another pattern of facts must appear in the target. They are also known as *global-and-local-as-view* (GLAV) dependencies, and can accommodate both GAV and LAV formalisms.

Our approach uses s-t tgd's for the translation of the semantic mappings expressed between md-schemata. In this way, we do not let our approach be guided by the syntax of the language used at the OLAP level. Indeed, the mapping language presented in Section 4 gives users powerful predicates to express their specification needs. Nevertheless, reformulating queries at the OLAP level would mean adopting a syntax-oriented approach. Instead, we abstract from the mapping language syntax and we found our proposal on the semantics of the transformations data is subjected to along the reformulation process. This allows the mapping predicates expressed at the OLAP level to be bound to classes of mappings on the basis of two different types of transformations: depending on the presence or absence of the encoding function for expressing the relationship between data values, mappings can be either expressed as full s-t tgd's, that are logically equivalent to finitely many GAV dependencies, or as LAV dependencies. This vision simplifies the query reformulation task. As to this point, several sound and complete algorithms have been presented in the literature [6]. However, the existing solutions cannot be straightforwardly applied to OLAP queries, thus they need specific extensions. This is precisely the main objective of our work.

To this end, in this section we introduce the framework for query reformulation depicted in Figure 4, where md-schemata, OLAP queries, and mappings at the OLAP level are translated to the relational model. An OLAP query is

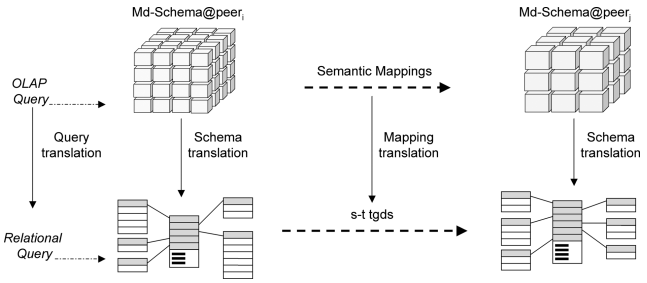


**Figure 4: Reformulation framework**

HospFT(organ,disease,date,ward,patient,cost,durationOfStay)
OrganDT(organ) DiseaseDT(disease)
DateDT(date,week,month,year) WardDT(ward,LHD)
PatientDT(patient,birthDate,city,region,segment,gender)

AdmFT(diagnosis,date,ward,patientCity,patientBirthYear,patientGender,
totStayCost,totExamCost,totLength,numAdmissions)
DiagnosisDT(diagnosis,category) DateDT(date,month,year)
WardDT(ward,unit) PatientCityDT(patientCity,patientNation)
PatientBirthYearDT(patientBirthYear) PatientGenderDT(patientGender)

**Figure 5: Star schemata for the Rome (top) and Florence (bottom) peers**

then reformulated starting from its relational form, using the mappings expressed at the relational level as s-t tgd's.

## 5.1 Translating schemata

Let $\mathcal{M} = \langle A, H, M \rangle$ be an md-schema. Without loss of generality, we assume that $\mathcal{M}$ is stored as a standard star schema: a fact table, $ft(\underline{DIM_1}, \ldots, \underline{DIM_n}, m_1, \ldots, m_l)$, and one dimension table for each hierarchy $h_i$, $dt_i(\underline{DIM_i}, a_{i_1}, \ldots, a_{i_k})$ where $\{DIM_i, a_{i_1}, \ldots, a_{i_k}\} = Attr(h_i)$. The star schema corresponding to the HOSPITALIZATION and ADMISSION md-schemata are shown in Figure 5.

It is worth noting that, while at the OLAP level both mappings and queries directly use the attributes and measures *names*, the query and mapping languages we borrow from the relational model use ROLAP tables under the *unnamed* perspective, i.e., the specific attribute names are ignored, and only the number of attributes of each relation schema is available. These languages, both stemming from mathematical logic, view a database schema as a tuple $\mathbf{R} = (r_1, \ldots, r_z)$ of relation symbols, each of which has a fixed arity. To switch from the named perspective of the OLAP level to the unnamed one of the relational level, a dimension table-coding function $\delta_{dt} : A \to \mathbb{N}$ is necessary to associate each attribute $a \in Attr(h_i)$ with the corresponding coordinate (i.e., position) in $dt_i$, as well as a fact table-coding function $\delta_{ft} : M \cup \{DIM_1, \ldots, DIM_n\} \to \mathbb{N}$ which associates each measure and dimension with the corresponding coordinate in $ft$.

## 5.2 Translating queries

Using a classical logic-based syntax, we represent a PSJ query on a star schema as a conjunction of relational atoms and a Boolean predicate having the following rule-based form:

$$q(\overline{x}) \leftarrow r_1(\overline{x_1}), \ldots, r_z(\overline{x_z}), p(\overline{y})$$

where each $r_i$ refers to a relation, each $\overline{x_i}$ is a tuple of variables and/or constants of the same arity of $r_i$, $p$ is an atomic Boolean predicate (e.g., $\mathsf{totStayCost} + \mathsf{totExamCost} \leq 9$) involving variables in $\bigcup_{i=1\ldots z} \overline{x_i}$, and all the variables of the tuple $\overline{x}$ appear in the body.

A GPSJ query is then represented as a conjunctive query with an aggregate term in its head:

$$q(\overline{x}, \alpha(\overline{z})) \leftarrow body$$

where $\alpha$ is an aggregate function, no variable in $\overline{x}$ occurs in $\overline{z}$, and all variables in $\overline{x}$ and $\overline{z}$ appear in the condition *body*.

EXAMPLE 4. *The OLAP query shown in Example 2 is translated on the* HOSPITALIZATION *star schema as*

$$q(R, Y, \mathtt{sum}(C)) \leftarrow \mathsf{HospFT}(\_, \_, D, \_, P, C, \_), \mathsf{DateDT}(D, \_, \_, Y), \mathsf{PatientDT}(P, \_, \_, R, \_, G), G = \text{'F'}$$

*which corresponds to the following relational algebra expression:*

$$\pi_{\mathsf{region},\mathsf{year},\mathtt{sum}(\mathsf{cost})} \sigma_{\mathsf{gender}=\text{'F'}}(\mathsf{HospFT} \bowtie \mathsf{DateDT} \bowtie \mathsf{PatientDT})$$

Formally, let $q = \langle \mathcal{M}, G, m, p, \alpha \rangle$ be an OLAP query, where $G = \{a_1, \ldots, a_u\}$. Let $h_{i_1}, \ldots, h_{i_z}$ be the hierarchies involved in $q$. The translation of $q$ to the relational level relies on a variable-assignment function that associates measure $m$ and each attribute $a$ involved in $q$ (either in the group-by set or in the selection predicate) with a free variable $\mu(m)$ and $\mu(a)$, respectively. To enable the join between dimension tables and the fact table to be specified, also the dimension $DIM_i$ of each involved hierarchy is associated with a variable $\mu(DIM_i)$, with the proviso that $\mu(a) = \mu(DIM_i)$ whenever dimension $DIM_i$ is directly involved in $q$. The relational translation of $q$ is then the aggregate query

$$q(\mu(a_1), \ldots, \mu(a_u), \alpha(\mu(m))) \leftarrow ft(\overline{x}), dt_{i_1}(\overline{x_1}), \ldots, dt_{i_z}(\overline{x_z}), \mu(p)$$

where

- the tuple $\overline{x}$ is such that $\overline{x}.\delta_{ft}(DIM_i) = \mu(DIM_i)$ for the $i$-th involved hierarchy, and $\overline{x}.\delta_{ft}(m) = \mu(m)$. The other variables in $\overline{x}$ are anonymous and denoted with the symbol _;
- each tuple $\overline{x_i}$ is such that $\overline{x_i}.\delta_{dt}(a) = \mu(a)$ for each attribute $a$ involved from the $i$-th hierarchy. The other variables in $\overline{x_i}$ are anonymous;
- with a slight abuse of notation, $\mu(p)$ denotes the substitution of the measure $m$ and each attribute $a$ in $p$ with the corresponding variable $\mu(m)$ and $\mu(a)$, respectively.

We assume that, to enforce peer autonomy and privacy, the underlying star schemata of each peer are not directly accessible from outside. This means that the interface for a peer is the set of all the possible OLAP queries (see Definition 3) supported by the md-schema it exposes. From the reformulation point of view, this means that each reformulated query does not directly address the source tables, but rather it refers to one out of all the possible OLAP queries over the source md-schema $s$. To this end, we consider each possible OLAP query as a (typically virtual) view $v(\mu(a_1), \ldots, \mu(a_u), \alpha(\mu(m)))$ over source schema $\mathbf{S}$.

## 5.3 Translating mappings

In line with the adopted approach, we represent mappings as s-t tgd's. More precisely, given a source schema $\mathbf{S}$ and a target schema $\mathbf{T}$, an s-t tgd has the form

$$\forall \overline{x}(\phi(\overline{x}) \rightarrow \exists \overline{y}\psi(\overline{x}, \overline{y}))$$

where $\phi(\overline{x})$ is a conjunction of atomic formulas over $\mathbf{S}$ and $\psi(\overline{x}, \overline{y})$ is a conjunction of atomic formulas over $\mathbf{T}$. In spite of their syntactic simplicity, s-t tgd's can express many data interoperability tasks arising in applications. As mentioned before, an s-t tgd can accommodate both GAV and LAV formalisms. In particular, in a GAV dependency the right-hand side of the implication consists of a single atomic formula:

$$\forall \overline{x}(\phi(\overline{x}) \rightarrow U(\overline{x'}))$$

where the variables in $\overline{x'}$ are among those in $\overline{x}$, while in a LAV dependency the left-hand side of the implication consists of an atomic formula:

$$\forall \overline{x}(R(\overline{x}) \rightarrow \exists \overline{y}\psi(\overline{x}, \overline{y}))$$

In the following, we present the translation of the mapping predicates proposed in Section 4 into s-t tgd's.[1]

We start from those predicates that can contain encoding functions to be used for translating source values to the target domains, i.e. same, equi-level, and roll-up. The proposed s-t tgd's express the constraints between the two star schemata given by the function that relates measure $m$ to a set of measures $d$, or the attribute sets $d$ and $c$.

EXAMPLE 5. *With reference to Figure 5 and Example 3, we report some examples of mapping translations:*

- *Mapping* $\langle \mathsf{cost} \rangle$ $\mathsf{same}_f$ $\langle \mathsf{totStayCost}, \mathsf{totExamCost} \rangle$ *translates to*

$$\forall S, E, C(\mathsf{AdmFT}(\_, \ldots, \_, S, E, \_, \_), C = S + E \rightarrow \mathsf{HospFT}(\_, \ldots, \_, C, \_)$$

- *Mapping* $\langle \mathsf{disease}, \mathsf{organ} \rangle$ $\mathsf{equi\text{-}level}_f$ $\langle \mathsf{diagnosis} \rangle$ *translates to*

$$\forall D, D', O(\mathsf{DiagnosisDT}(D, \_), D' = f(D).1, O = f(D).2 \rightarrow \mathsf{DiseaseDT}(D'), \mathsf{OrganDT}(O))$$
$$\forall D, D', O(\mathsf{AdmFT}(D, \_, \ldots, \_), D' = f(D).1, O = f(D).2 \rightarrow \mathsf{HospFT}(O, D', \_, \ldots, \_))$$

- *Mapping* $\langle \mathsf{week} \rangle$ $\mathsf{roll\text{-}up}_f$ $\langle \mathsf{date} \rangle$ *translates to*

$$\forall D, W(\mathbf{S}.\mathsf{DateDT}(D, \_, \_), W = f(D) \rightarrow \mathbf{T}.\mathsf{DateDT}(\_, W, \_, \_)$$

The fact that two different predicates, equi-level and roll-up, translate to the same s-t tgd form could sound strange. Indeed, an equi-level predicate states that two attribute sets have the same granularity, whereas the roll-up predicate states a one-to-many relationship between them. Intuitively, differently from the equi-level predicate, when a roll-up predicate is used to connect two attribute sets, users know that it is necessary to aggregate source data in order to be compatible with the target domain. On the other hand, in the translation of the mapping language to the relational level, we disregard syntactic forms but, rather, we focus on the kind of tuple dependencies each mapping predicate defines, which is exclusively dependent on the function properties. For instance, roll-up functions are non-injective and this implies additional aggregations. Nevertheless, given the mapping $\langle \mathsf{disease}, \mathsf{organ} \rangle$ $\mathsf{equi\text{-}level}_f$ $\langle \mathsf{diagnosis} \rangle$, when only the disease attribute is required, it is necessary to aggregate the diagnosis values returned by the source peer on their disease values because $f$ limited to the first co-domain could no longer be injective.

Finally, whenever an encoding function for the same, equi-level, and roll-up predicates is not available and for the other predicates where encoding functions cannot be used for data transformation, no constraint between the star schemata can be stated and the proposed s-t tgd's do not express any relationship between the right hand and the left hand variables.

EXAMPLE 6. *If an encoding function $f$ is not available, then the* equi-level *mapping of the example above becomes:*

$$\forall D(\mathsf{DiagnosisDT}(D, \_) \rightarrow \exists D', O(\mathsf{DiseaseDT}(D'), \mathsf{OrganDT}(O))$$
$$\forall D(\mathsf{AdmFT}(D, \_, \ldots, \_) \rightarrow \exists D', O(\mathsf{HospFT}(D', O, \_, \ldots, \_))$$

The formal translation of all the predicates is shown in Table 1. Without loss of generality, we assume that each involved attribute $a_i$ or $b_i$ is associated with the hierarchy $h_{a_i}$ or $h_{b_i}$, respectively, and that the first $l$ attributes ($0 \leq l \leq j$) $a_1, \ldots, a_l$ of $c$ and $l'$ attributes ($0 \leq l' \leq k$) $b_1, \ldots, b_{l'}$ of $d$ are dimensions. Then, tuples are defined as follows (all the unspecified variables are anonymous):

- the tuple $\overline{x_s^m}$ is such that $\overline{x_s^m}.\delta_{ft}(m_i) = \mu(m_i)$ for $i$ from 1 to $k$;
- the tuple $\overline{x_t^m}$ is such that $\overline{x_t^m}.\delta_{ft}(m) = \mu(m)$;
- each tuple $\overline{x_{a_i}}$ is such that $\overline{x_{a_i}}.\delta_{dt}(a_i) = \mu(a_i)$;
- the tuple $\overline{x_{b_i}}$ is such that $\overline{x_{b_i}}.\delta_{dt}(b_i) = \mu(b_i)$;
- the tuple $\overline{x_s^a}$ is such that $\overline{x_s^a}.\delta_{ft}(b_i) = \mu(b_i)$ for $i$ from 1 to $l'$;
- the tuple $\overline{x_t^a}$ is such that $\overline{x_t^a}.\delta_{ft}(a_i) = \mu(a_i)$ for $i$ from 1 to $l$.

# 6. THE REFORMULATION ALGORITHM

We are now ready to define the query reformulation problem at the relational level.

DEFINITION 4 (QUERY REFORMULATION). *Given a GPSJ query $q$ on target schema $\mathbf{T}$, a source schema $\mathbf{S}$, and a set of s-t tgd's between $\mathbf{S}$ and $\mathbf{T}$, a reformulation of $q$ on $\mathbf{S}$ is a GPSJ query $q'$ that refers only to the views of $\mathbf{S}$, together with a variable set mapping $var : 2^V \rightarrow 2^V \times \mathcal{F}$ where $V$ is the set of all possible variable names and $\mathcal{F}$ is the set of all possible many-to-one and many-to-many functions $F$ on the attribute domains.*

The set $\mathcal{F}$ includes the functions $F$ that can appear in the drill-down and related mappings but do not participate to the mapping translation. These functions are associated to $q'$ through function *var* because, as already mentioned in Section 4, they could be used to improve result presentation.

[1] In case of ambiguity, we will use prefixes $\mathbf{S}$ and $\mathbf{T}$ to distinguish source tables from target ones.

| Predicate | Translation |
|---|---|
| $m$ $\mathsf{same}_f$ $\langle m_1, \ldots, m_k \rangle$ | $\forall \mu(m_1), \ldots, \mu(m_k), \mu(m)(\mathbf{S}.ft(\overline{x_s^m}), \mu(m) = f(\mu(m_1), \ldots, \mu(m_k)) \rightarrow \mathbf{T}.ft(\overline{x_t^m}))$ |
| $\langle a_1, \ldots, a_j \rangle \mathsf{equi\text{-}level}_f \langle b_1, \ldots, b_k \rangle$ | $\forall \mu(a_1), \ldots, \mu(a_j), \mu(b_1), \ldots, \mu(b_k)(\mathbf{S}.dt_{b_1}(\overline{x_{b_1}}), \ldots, \mathbf{S}.dt_{b_k}(\overline{x_{b_k}}),$ $\mu(a_1) = f(\mu(b_1), \ldots, \mu(b_k)).1, \ldots, \mu(a_j) = f(\mu(b_1), \ldots, \mu(b_k)).j$ $\rightarrow \mathbf{T}.dt_{a_1}(\overline{x_{a_1}}), \ldots, \mathbf{T}.dt_{a_j}(\overline{x_{a_j}}))$ |
| $\langle a_1, \ldots, a_j \rangle \mathsf{roll\text{-}up}_f \langle b_1, \ldots, b_k \rangle$ | $\forall \mu(a_1), \ldots, \mu(a_l), \mu(b_1), \ldots, \mu(b_k)(\mathbf{S}.ft(\overline{x_s^a}), \mu(a_1) = f(\mu(b_1), \ldots, \mu(b_k)).1, \ldots,$ $\mu(a_l) = f(\mu(b_1), \ldots, \mu(b_k)).l \rightarrow \mathbf{T}.ft(\overline{x_t^a}))$ |
| $m$ $\mathsf{same}$ $\langle m_1, \ldots, m_k \rangle$ | $\forall \mu(m_1), \ldots, \mu(m_k)(\mathbf{S}.ft(\overline{x_s^m}) \rightarrow \exists \mu(m)(\mathbf{T}.ft(\overline{x_t^m})))$ |
| $\langle a_1, \ldots, a_j \rangle \mathsf{equi\text{-}level} \langle b_1, \ldots, b_k \rangle$ | $\forall \mu(b_1), \ldots, \mu(b_k)(\mathbf{S}.dt_{b_1}(\overline{x_{b_1}}), \ldots, \mathbf{S}.dt_{b_k}(\overline{x_{b_k}})$ $\rightarrow \exists \mu(a_1), \ldots, \mu(a_j)(\mathbf{T}.dt_{a_1}(\overline{x_{a_1}}), \ldots, \mathbf{T}.dt_{a_j}(\overline{x_{a_j}})))$ |
| $\langle a_1, \ldots, a_j \rangle \mathsf{roll\text{-}up} \langle b_1, \ldots, b_k \rangle$ | |
| $\langle a_1, \ldots, a_j \rangle \mathsf{drill\text{-}down}_F \langle b_1, \ldots, b_k \rangle$ | $\forall \mu(b_1), \ldots, \mu(b_{l'})(\mathbf{S}.ft(\overline{x_s^a}) \rightarrow \exists \mu(a_1), \ldots, \mu(a_l)(\mathbf{T}.ft(\overline{x_t^a})))$ |
| $\langle a_1, \ldots, a_j \rangle \mathsf{related}_F \langle b_1, \ldots, b_k \rangle$ | |

**Table 1: Translation of the mapping predicates**

This section presents an algorithm that reformulates queries in three steps. First, it matches the body of $q$ with the set of mappings. Then, using the selected mappings, it reformulates $q$ into a query $q'$ which only refers to the source schema $\mathbf{S}$. Finally, it selects the set of views on $\mathbf{S}$ to be used for $q'$. In the following we provide an overview of each single step, also by means of the following example.

EXAMPLE 7. *Consider a query asking for the total hospitalization cost for each disease and LHD:*

$$q = \langle \mathsf{HOSPITALIZATION}, \{\mathsf{disease}, \mathsf{LHD}\}, \mathsf{cost}, \varepsilon, \mathtt{sum} \rangle$$

*translated at the relational level into*

$$q(D, L, \mathtt{sum}(C)) \leftarrow \mathsf{HospFT}(\_, D, \_, W, \_, C, \_), \mathsf{DiseaseDT}(D), \mathsf{WardDT}(W, L)$$

*In relational algebra:*

$$q = \pi_{\mathsf{disease,LHD},\mathtt{sum}(\mathsf{cost})}(\mathsf{HospFT} \bowtie \mathsf{DiseaseDT} \bowtie \mathsf{WardDT})$$

*Due to the lack of space, we do not provide the full set of s-t tgd's corresponding to the mappings shown in Figure 3, but we only show the s-t tgd's used in the reformulation process.*

**Step 1 (s-t tgd's matching)** In this step we match the body of $q$ against the set of s-t tgd's, denoted with $\Omega = \{\omega_1, \ldots, \omega_n\}$. A matching of a mapping $\omega_i$ is a subset of the atomic predicates in $q$ that together (partially) covers the right-hand of $\omega_i$ through a variable unifier which associates at least one non-anonymous variable of $\omega_i$ to a non-anonymous one in $q$. In this way, we associate each atomic predicate $p(\overline{x})$ with the set $\Omega(p(\overline{x}))$ of matching s-t tgd's.

EXAMPLE 8. *The s-t tgd's matching with the body of $q$ are (assuming that no encoding function is provided for $\omega_3$ and $f$ is the identity function for $\omega_4$):*[2]

$$\omega_1 : \forall S, E, C(\mathsf{AdmFT}(\_, \ldots, \_, S, E, \_, \_), C = S + E \rightarrow \mathsf{HospFT}(\_, \ldots, \_, C, \_))$$

$$\omega_3 : \forall U(\mathbf{S}.\mathsf{WardDT}(\_, U) \rightarrow \exists L(\mathbf{T}.\mathsf{WardDT}(\_, L)))$$

$$\omega_{4a} : \forall W, W'(\mathbf{S}.\mathsf{WardDT}(W, \_), W' = W \rightarrow \mathbf{T}.\mathsf{WardDT}(W', \_))$$

$$\omega_{4b} : \forall W, W'(\mathsf{AdmFT}(\_, \_, W, \_, \ldots, \_), W' = W \rightarrow \mathsf{HospFT}(\_, \_, \_, W', \_, \ldots, \_))$$

[2]When a single mapping translates into two s-t tgd's, we distinguish them by means of suffixes $a$ and $b$.

$$\omega_{9a} : \forall D, D', O(\mathsf{DiagnosisDT}(D, \_), D' = f(D).1, O = f(D).2 \rightarrow \mathsf{DiseaseDT}(D'), \mathsf{OrganDT}(O))$$

$$\omega_{9b} : \forall D, D', O(\mathsf{AdmFT}(D, \_, \ldots, \_), D' = f(D).1, O = f(D).2 \rightarrow \mathsf{HospFT}(O, D', \_, \ldots, \_))$$

*Therefore,* $\Omega(\mathsf{HospFT}(\_, D, \_, W, \_, C, \_)) = \{\omega_1, \omega_{4b}, \omega_{9b}\}$, $\Omega(\mathsf{DiseaseDT}(D)) = \{\omega_{9a}\}$, *and* $\Omega(\mathsf{WardDT}(W, L)) = \{\omega_3, \omega_{4a}\}$.

**Step 2 (query expansion)** This step expands the body of $q$ using the left side of the selected s-t tgd's. In particular, for each atomic predicate $p(\overline{x})$, we compute the reformulation $ref(p(\overline{x}))$ of $p(\overline{x})$ as follows. Starting from an empty reformulation $ref_0(p(\overline{x}))$, we consider the sequence of the matching s-t tgd's $\Omega(p(\overline{x})) = \{\omega_1^p, \ldots, \omega_k^p\}$ and, at each step, for $i = 1, \ldots, k$, we expand the reformulation obtained at step $i-1$, $ref_{i-1}(p(\overline{x}))$ by using the left-hand of $\omega_i$. Then $ref_k(p(\overline{x})) = ref(p(\overline{x}))$.

More precisely, if $\omega_i$ is a GAV-style mapping then the algorithm expands $ref_{i-1}(p(\overline{x}))$ with the left-hand of $\omega_i'$ where $\omega_i'$ is the result of unifying $p(\overline{x})$ with the right-hand of $\omega_i$. If $\omega_i$ is a (G)LAV-style mapping $\forall \overline{x}(\phi(\overline{x}) \rightarrow \exists \overline{y} \psi(\overline{y}))$, then the algorithm expands $ref_{i-1}(p(\overline{x}))$ with the left-hand of $\omega_i$. In this case, it also adds a variable set mapping $var : \overline{y'} \mapsto (\overline{x}, F)$ where $\overline{y'}$ is the result of unifying $p(\overline{x})$ with $\psi(\overline{y})$ and $F$ is the function associated with the mapping $\omega_i$, if it exists, or a null function otherwise.

Finally, it could be necessary to also modify the head of $q$. Indeed, if $var$ is not empty, some of the variables in the head of $q$ may not be in the body. Therefore, the algorithm replaces each variable set $V \in dom(var)$ with $var(V)$. If the substitution involves the queried measure $m$, then the algorithm also replaces the aggregation function with $Agg(m)$.

EXAMPLE 9. *By following the steps described above, given that $var : L \mapsto U$, $q'$ is defined as follows:*

$$q'(D, U, \mathtt{sum}(C)) \leftarrow \mathsf{AdmFT}(D', \_, W, \_, \_, S, E, \_, \_), C = S + E, \mathsf{DiagnosisDT}(D', \_), D = f(D').1, \mathsf{WardDT}(W, U)$$

*Note that $q$, which originally asked for costs aggregated on LHD values, has been transformed to return costs aggregated by* unit *because there is no knowledge about how to trans-*

*form units into LHDs. On the contrary,* disease *values are extracted from* diagnosis *ones through function $f$.*

**Step 3 (view selection)** In the third step, the algorithm selects the views on the source schema **S** to be used for rewriting $q'$. Note that each view is univocally identified by the set of atomic predicates in its body. Indeed, the variables specified in the dimension tables correspond to the selected attributes while the remaining variable specified in the fact table correspond to the involved measure. Finally, the involved aggregate function is the one associated with the selected measure. Therefore, the algorithm first identifies the set of candidate views $\mathcal{V}$ in the body of $q'$.

EXAMPLE 10. *The candidate views of query $q'$ are:*

$$v_1(D', U, \texttt{sum}(E)) \leftarrow \mathsf{AdmFT}(D', \_, W, \_, \_, \_, \_, E, \_, \_), \mathsf{DiagnosisDT}(D', \_), \mathsf{WardDT}(W, U)$$

$$v_2(D', U, \texttt{sum}(S)) \leftarrow \mathsf{AdmFT}(D', \_, W, \_, \_, \_, S, \_, \_, \_), \mathsf{DiagnosisDT}(D', \_), \mathsf{WardDT}(W, U)$$

*View $v_1$ computes the total exam cost while $v_2$ computes the total staying cost for each diagnosis and unit:*

$$v_1 = \pi_{\mathsf{diagnosis,unit,sum(totExamCost)}}(\mathsf{AdmFT} \bowtie \mathsf{DiagnosisDT} \bowtie \mathsf{WardDT})$$

$$v_2 = \pi_{\mathsf{diagnosis,unit,sum(totStayCost)}}(\mathsf{AdmFT} \bowtie \mathsf{DiagnosisDT} \bowtie \mathsf{WardDT})$$

The next step consists in rewriting $q'$ using $\mathcal{V}$, i.e., to derive a query $r$ over $\mathcal{V}$ such that $r$ is equivalent to $q'$. This problem has been deeply investigated in [3], that shows how aggregate views can be safely used to rewrite aggregate queries when the involved aggregate function is a commutative-semigroup one. Examples of this kind of functions are `sum`, `max`, `prod`, and `count`. Moreover, the paper also identifies special cases where it is possible to use aggregate views for functions that are not commutative-semigroup aggregation functions, such as `avg`. In our case, to have a safe rewriting, we also require that for each same mapping with encoding function $f$, $\alpha(f(\overline{x})) = f(\alpha_1(x_1), \ldots, \alpha_k(x_k))$, where $\alpha = Agg(m)$, $\alpha_i = Agg(b_i)$, and $x = \langle x_1, \ldots, x_k \rangle \in Dom(d)$.

EXAMPLE 11. *The resulting rewriting is*

$$q'(D, U, \texttt{sum}(C)) \leftarrow v_1(D', U, sumE), v_2(D', U, sumS), C = sumE + sumS, D = f(D').1$$

*Note that the query computes the total cost for each diagnosis and unit through the involved views. Then, it further aggregates the returned values by* disease*, whose values are extracted from the* diagnosis *ones through function $f$:*

$$q' = \pi_{substring(\mathsf{diagnosis},1,20),\mathsf{unit},\mathsf{sum(totExamCost+totStayCost)}}(v_1 \bowtie v_2)$$

We finally remark that, whenever $q$ contains a Boolean predicate $x \langle op \rangle val$, it can be reformulated on the source schema only if the mapping associated with $x$ contains an encoding function. Indeed, only in this case it is possible to compare the extracted values with $val$.

# 7. CONCLUSIONS

Extending the PDMS paradigm to the BI context is a challenging task that lays the foundations for BI 2.0. This paper is a first, significant step in this direction. Our future work on this topic will concern enhancing the reformulation algorithm to handle more complex queries (such as those using non-distributive aggregation operators).

Though query reformulation is at the core of a distributed approach, several other issues must be tackled to complete the BIN framework, namely: how to efficiently process queries across the peer network, how to reconcile multidimensional data returned by different peers through object fusion techniques, how to rank peer results depending on how compliant they are with the original local query, and how to deal with security depending on the degree of trust between the BIN participants.

# 8. REFERENCES


[1] S. Abiteboul. Managing an XML warehouse in a P2P context. In *Proc. CAiSE*, pages 4–13, Klagenfurt, Austria, 2003.

[2] M. Banek, B. Vrdoljak, A. M. Tjoa, and Z. Skocir. Automated integration of heterogeneous data warehouse schemas. *IJDWM*, 4(4):1–21, 2008.

[3] S. Cohen, W. Nutt, and Y. Sagiv. Rewriting queries with arbitrary aggregation functions using views. *ACM TODS*, 31(2):672–715, 2006.

[4] M. Golfarelli, S. Rizzi, and P. Biondi. MYOLAP: An approach to express and evaluate olap preferences. *IEEE TKDE, to appear*, 2010.

[5] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. In *Proc. VLDB*, pages 358–369, Zurich, Switzerland, 1995.

[6] A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB Journal*, 14(1):68–83, 2005.

[7] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza peer data management system. *IEEE TKDE*, 16(7):787–798, 2004.

[8] T. A. D. Hoang and T. B. Nguyen. State of the art and emerging rule-driven perspectives towards service-based business process interoperability. In *Proc. Int. Conf. on Comp. and Comm. Tech.*, pages 1–4, Danang City, Vietnam, 2009.

[9] H. Jiang, D. Gao, and W.-S. Li. Exploiting correlation and parallelism of materialized-view recommendation for distributed data warehouses. In *Proc. ICDE*, pages 276–285, Istanbul, Turkey, 2007.

[10] P. Kalnis, W. S. Ng, B. C. Ooi, D. Papadias, and K.-L. Tan. An adaptive peer-to-peer network for distributed caching of OLAP results. In *Proc. SIGMOD Conf.*, pages 25–36, Madison, Wisconsin, 2002.

[11] M. Kehlenbeck and M. H. Breitner. Ontology-based exchange and immediate application of business calculation definitions for online analytical processing. In *Proc. DaWaK*, pages 298–311, Linz, Austria, 2009.

[12] N. Raden. Business intelligence 2.0: Simpler, more accessible, inevitable. http://intelligent-enterprise.informationweek.com, 2007.

[13] B. ten Cate and P. G. Kolaitis. Structural characterizations of schema-mapping languages. *Commun. ACM*, 53(1):101–110, 2010.

[14] R. Torlone. Two approaches to the integration of heterogeneous data warehouses. *Distributed and Parallel Databases*, 23(1):69–97, 2008.