Don't Mind Your Vocabulary: Data Sharing Across Heterogeneous Peers

Md. Mehedi Masud[1], Iluju Kiringa[1], and Anastasios Kementsietsidis[2]

[1] SITE,University of Ottawa
{mmasud, kiringa}@site.uottawa.ca
[2] School of Informatics, University of Edinburgh
akements@inf.ed.ac.uk

Abstract. The strong dynamics of peer-to-peer networks, coupled with the diversity of peer vocabularies, makes query processing in peer database systems a very challenging task. In this paper, we propose a framework for translating expressive relational queries across heterogeneous peer databases. Our framework avoids an integrated global schema or centralized structures common to the involved peers. The cornerstone of our approach is the use of both syntax and instance level schema mappings that each peer constructs and shares with other peers. Based on this user provided mapping information, our algorithm applies generic translation rules to translate SQL queries. Our approach supports both query translation and propagation among the peers preserving the autonomy of individual peers. The proposal combines both syntax and instance level mappings into a more general framework for query translation across heterogeneous boundaries. We have developed a prototype as a query service layer wrapped around a basic service providing heterogeneity management. The prototype has been evaluated on a small peer-to-peer network to demonstrate the viability of the approach.

1 Introduction

In the past few years, Peer-to-Peer (P2P) applications have emerged as a popular way of sharing data in decentralized and distributed environments. In such environments, involved data sources, called peers, act autonomously in their sharing of data and services. A peer database system (PDBS) consists of a peer or node of a P2P network which has been augmented both with a conventional data base management system and an interoperability layer that enables data sharing across (usually) heterogeneous boundaries. The local databases on each peers are called *peer databases*. Each PDBS is independent of others and maintains its own peer databases. In addition to the latter, each PDBS needs to establish value correspondences between its local data and data on remote PDBSs for the purpose of data sharing. Such value correspondences constitute logical links that we call *acquaintances*. For example, we may consider a network of peer database systems of family doctors, hospitals, medical laboratories, and pharmacists that are willing to share information about treatments, medications and

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 292–309, 2005.

test results of their patients. Consider a situation where a patient has an accident in a city where he is currently visiting. He then visits a walk-in clinic. The doctor in the clinic needs to know the patient's previous medications and treatments from the patient's family physician. The associated doctor in the walk-in clinic establishes acquaintances with the patient's family doctor and pharmacist for sharing information about the patient. The doctor in the walk-in clinic then retrieves information from the acquainted peers, for instance family physicians and medical laboratories. In this perspective, we need particularly fine-grained data coordination, sharing and query processing strategies. Coordination rules need to be dynamic because peers join and leave the system freely. In such a dynamic environment, the existence of a global schema for the entire databases is not feasible [4].

Query processing techniques in peer database networks can be compared to similar techniques used in the traditional distributed and multi database systems which use global or mediated schema to allow viewing of all involved databases as one single database.

On the contrary, in general, peer database systems have no such global schemas and permit true autonomy to peers. In addition, queries are posed on local database and results are retrieved from the local database as well as from the peer database network. In this paper, we make the following contributions.

- First, we present a query translation mechanism that does not use a restrictive global/mediated schema and that considers the heterogeneity and autonomy of peer databases. The algorithm translates arbitrary Select-Project-Join SQL queries, where the selection formula may contain both conjunctive and disjunctive normal form with negation.
- Second, the algorithm translates queries according to a collection of user-generated rules that combine a restricted form of syntactic schema mappings with the data instance level mappings. We also present a mapping stack consisting of four layers of mappings that each peer uses partially or fully in order to create semantic relationships with other peers.
- Finally, we implemented our query translation algorithm on top of the increasingly reliable P2P JXTA framework [1]. We ran the implementation on six JXTA-based peers. We show measurements that indicate that the algorithm is efficient.

The remainder of the paper is organized as follows. Section 2 describes a usage scenario. In Section 3, we discuss peer-to-peer mapping semantics, followed by Section 4 that discusses the peer-to-peer query semantics. Section 5 presents the query translation rules and framework and Section 6 describes the query translation algorithm. In Section 7, we show experimental setup and results. Section 8 treats related work.

Finally, we address the scope for future work and conclude in Section 9.

2 Motivating Example

We use a motivating example from the Health Care domain where hospitals, family doctors and pharmacies, all share information about patients. This information includes health histories, medications, and exams. Figure 1 depicts the schemas used for this domain. The Family Doctor peer has a unique Ontario Health Insurance Patient (OHIP) number assigned to each patient and record is kept of name, illness, date of visit and medication of patients. The relation $MedicalExam$ stores the information about the patients' medical examinations. The Hospital peer has relations *Patients* and *Labtest*. Table 2 shows partial instances of both peer databases. Assume that a patient has been admitted to a hospital. The doctor in the hospital needs the medical examination that the patient has gone through on admission as well as the patient's recent test reports from the patient's family doctor. To get the medical examination, the doctor in the hospital may pose the following query against the local Hospital peer database:

*Q1: select * from LabTest where PATID="243" AND Test="C0518015"*

To get the test reports, the very same doctor needs to have the following query posed against the remote peer database of the patient's family doctor:

*Q2: select * from MedicalExam where OHIP="501NE" AND TestName="homoglobin"*

Normally, due to the autonomy of peer databases, the doctor in the hospital should be able to pose the later query in terms of the schema and instance data values of his local peer database. However, Figure 2 shows that there are differences in the way patients' information is represented both at the schema and at the data instance level in the two peer databases. This representational discrepancy raises the need for some way of mapping one representation to the other, both at the schema and at the data instance levels. Such a mapping will permit interpretability between these heterogeneous scenarios for query translation. We will use mapping tables [15] to resolve heterogeneity at the data level and syntactic mappings of schema element for schema level heterogeneity during

Patient (OHIP, Lname, Fname, Illness, Date)
MedicalExam(OHIP, TestName, Result)

(a) Family doctor database

Patients (PATID, Primary_Desc, Name)
Labtest (TestID, Test, Result, PATID)

(b) Hospital database

Fig. 1. Database schemas

OHIP	Lname	Fname	Illness	Date
233GA	Lucas	Andrew	Headache	Jan/04
501NE	Davidson	Ley	Allergy	Jan/04

(a) Patient table instance

OHIP	TestName	Result
233GA	whitebloodcount	9755 c/mcL
501NE	homoglobin	14.6 g/dL

(b) MedicalExam table instance

PATID	Name	Primary_Desc
243	Davidson,ley	StomachPain
359	Lucas, Andrew	Heart Problem

(c) Patients

TestId	Test	Result	PATID
4520	C0427512	633 c/mcL	359
4521	C0518015	12.5 g/dL	243

(d) LabTest table instances of hospital database

Fig. 2. Database instances

query translation. Based on these mappings, we develop several generic query translation rules that translate a query q posed on a peer P to a set of queries $Q' = \{q_1, \cdots, q_n\}$, where each one of the q_i's is the original query q translated to the vocabulary of a given acquainted peer P_i.

3 Peer-to-Peer Mappings

In what follows, we assume that sources are fully autonomous and may partially or fully share information at different levels. We consider pairwise mappings between peers with their shared schema elements. A mapping consists of four layers. Figure 3 shows the four layers of mappings. The top layer is the peer-to-peer mapping, which defines the acquaintance between two peers. Schema elements are exchanged between peers as part of the acquaintance protocol. The second layer is the schema level mapping which helps to overcome schema level heterogeneity. The next layer is the attribute level mapping that is used to overcome heterogeneity at the attribute level. The last one is called instance/data level mapping. We use this layer if there are any differences in data vocabularies between attributes of two peer relations. We use the concept of mapping tables to create the data layer mapping.

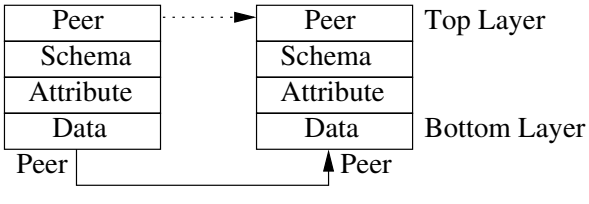

Fig. 3. Peer to Peer mapping stack

There may be four different types of mappings between schema elements. They are as follows: one to one, one to many, many to one, and many to many. To take on one these, in a one to many mapping, a schema element such as an attribute/ relation of one peer is mapped to more than one attribute/relation of another peer. For example the attribute Name of relation Patients is mapped to attributes *Lname* and *Fname* of relation *Patient*. Moreover, a set of attributes of one relation in one peer may be mapped to more than one relation in another peer.

3.1 Mapping Tables

Mapping tables [15] are used to represent the data value correspondences between attributes of two different relations to account for differences in data vocabulary. Intuitively, a mapping table is a relation over the attributes $X \cup Y$, where X and Y are non-empty sets of attributes from two peers. For example, Figure 4 shows a mapping table representing a mapping from the set of attributes X=$\{OHIP\}$ to the set of attributes Y=$\{PAITID\}$. The same table shows another mapping table that relates $MedicalExam.TestName$ and $LabTest.Test$.

Mapping tables represent expert knowledge and are typically created by domain specialists. Currently the creation of mapping tables is a time-consuming and manual process performed by a set of expert curators. Still there is no complete automated tool to facilitate the creation, maintenance and management of these tables [15]. However, the paper [15] introduces two mechanisms: i. *Infer new mapping tables* to find a set of all mapping tables that are valid and available over a network of peers and ii. *Determine consistency of mapping table* to ensure consistency when update occurs in mapping tables. However, both of these mechanisms play an important role in helping a curator understand and correctly specify the semantics of a set of mapping tables.

OHIP	PATID
501NE	243
233GA	388

(a) Mapping table OHIP2PATID

TestName	Test
homoglobin	C0518015
whitebloodcount	C0427512

(b) Mapping table TestName2Test

Fig. 4. Example of mapping tables

We can treat mapping tables as constraints in exchanging information between peer databases [15]. In the present paper, we use mapping table in this sense. We assume a closed world semantics for mapping tables.

3.2 Data and P2P Network Model

This section introduces some basic notions that will be used throughout the paper. A *database schema* is any nonempty, finite set $DB[W] = \{R_1[U_1], \ldots, R_n[U_n]\}$

of relations, where U_i is a subset W, the set all available attributes, and R_i is a relation name; $R_i[U_i]$ denotes a relation R_i over a set U_i of attributes. Given a relation R, and a query q, we will use the notation $att(R)$ and $att(q)$ to denote the set of attributes mentioned in R and q respectively. *Instances* of a relation schema $R_i[U_i]$ and a relational database $DB[W]$ are defined in the usual way.

Now we formally introduce the notion of a network of PDBSs.

Definition 1. *A network of PDBSs is a pair $(\mathcal{N}, \mathcal{M})$. Here, $\mathcal{N} = \{(\mathcal{P}, \mathcal{L})\}$ is an undirected graph, where $\mathcal{P} = \{P_1, \cdots, P_n\}$ is a set of peers, $\mathcal{L} = \{(P_i, P_j) | P_i, P_j \in \mathcal{P}\}$ is a set of acquaintances. Each peer P_i is associated with an instantiated relational database – a peer database – with schema $DB_i[W_i]$, and each acquaintance (i, j) is associated with a set $\mathcal{M}_{ij} \in \mathcal{M}$ of mapping tables.*

We will interchangeably call networks of PDBSs "P2P networks". that peers make only a subset of its schema visible to its peers. That is, we assume that each peer P_i exports a (possibly empty) subset $V_i \subseteq DB_i[W_i]$ of schema elements (i.e. attributes or relations) called *export schema* of P_i. For simplicity, we assume that $V_i \cap V_j = \emptyset$, for all $i \neq j$.

4 Peer-to-Peer Query Semantics

We assume that each query in PDBSs is defined in terms of the schema of a single peer. Each user is only aware of the local database schema. We also assume unrestricted select-project-join SQL queries. There are two types of queries in PDBs, namely local and global queries [8]. A local query is defined in terms of a local peer database schema, while a global query is defined over the peer database schema of the P2P network that are directly or indirectly acquainted with the peer where the global query is initiated. Semantically, a global query is a set of queries translated from the local query, all destined to acquainted peers. A global query is generated when a user wants results from all reachable (directly or indirectly) acquainted peers in the P2P network.

This informal semantics of queries can be formalized as follows (slightly modifying the semantics given in [14]). Suppose a network $\mathfrak{N} = (\mathcal{N}, \mathcal{M})$ of PDBSs, where $\mathcal{N} = (\mathcal{P}, \mathcal{L})$ and $\mathcal{P} = \{P_1, \cdots, P_n\}$. A local query q in a peer $P_i \in \mathcal{P}$ is defined over (a subset of) the schema $DB_i[W_i]$ of P. The answer to q is a relation over the instance db_i of $DB_i[W_i]$. A global query $q_{\mathfrak{N}}$ over the P2P network $\mathfrak{N}$ is a set $\{q_1, \cdots, q_k\}$ $(1 \leq k \leq n)$, where each query q_i $(1 \leq i \leq k)$ is a *component query* defined over the schema of a reachable peer. The intuition behind this definition is the following: each component query q_i is a user defined query that has been forwarded to all reachable peers via translation through the given mapping tables. The answer to the global query $q_{\mathfrak{N}}$ is the set $\{q_1(db_{i_1}), \cdots, q_k(db_{i_k})\}$, where $q_j(db_{i_j})$ $(1 \leq j \leq k)$ is a relation over the instance db_{i_j} of peer P_j.

For query propagation we use a $translation - and - forward$ mechanism. When a user poses a query on a peer then the peer first translates the query for all acquainted peers and sends the translated queries to its acquaintances. Before

sending the translated query the peer first adds a tag, we say global identification (GID) of the query as well as the peer identification (PID). When a remote peer receives the query, the peer either translates and/or forwards the query adding its PID with the query. This $translation-and-forward$ process continues until no further propagation is possible. The global identification (GID) is used to avoid duplicate translation of a query in a peer because a peer may receive queries in different form but with same GID from multiple acquaintances. Therefore, if a peer receives a query with the same GID as a query already seen, then the new query is rejected. The path tag makes this possible and thus helps avoid cycles. So the path tag is used to trace from which peer the query has been originated and the list of peers the query has been visited. Therefore, looking at the path tag, a peer knows whom to forward the query.

5 Query Translation

Query translation is the problem of transforming the query vocabulary of q over the schema of a local peer P to a query q' over the schema of an acquainted peer P'. The query vocabulary refers to three types of information, namely the set of *attributes* that represent the answer or result of the query, the *data values* mentioned in query search conditions, and the *relation names* mentioned in the query.

Example 1. Consider the following query which retrieves OHIP number and test result of a patient with $Lname$ = "$Lucas$", $Fname$ = "$Andrew$ " and $TestName$ = "$whitebloodcount$".

Q3: select OHIP, Result, Date from Patient, MedicalExam
where Patients.OHIP=MedicalExam.OHIP AND (Lname="Lucas"
AND Fname="Andrew") AND Test="whitebloodcount"

From the above query we find the following query vocabularies: i. the set of attributes represent the answer or result of the query:(OHIP, Result, Date) ii. the set of query search conditions (Lname="Lucas", Fname="Andrew", and Test="whitebloodcount") and iii. the set of relations (Patient, MedicalExam).

In order to pose the query in terms of vocabularies of acquainted peers for example the peer $Hospital$, the translated query should be as follows.

Q4: select PATID, Result from Patients, LabTest
where Patients.PATID=LabTest.PATID AND Name="Lucas, Andrew"
AND Test="C0427512"

In order to translate the query vocabularies that means query attributes, relations and data vocabularies in search conditions we use the notion of correspondence assertions and introduce translation rules. We simply translate the query attributes and relations from correspondence assertions and search conditions from translation rules. The semantics of correspondence assertion and translation rules are defined in the following two sections. We later show how these two things are used to translate queries.

5.1 Correspondence Assertion (CA)

In our peer to peer system we assume that each peer exports part of their schema as a shared schema that are made available to the system to create acquaintances with other peers. Mapping tables are also placed in peers to resolve difference in data vocabulary with acquainted peers. Therefore, we need to formally characterize the relationship between exported elements (attributes/relations) between acquainted peer schemas. We capture this relationship in the notion *correspondence assertion* (CA) between peers' shared schemas. We can also create correspondence assertions between two attributes that form a mapping table. In general, a mapping table $m[X \cup Y]$ encodes, in addition to the set of data associations, an attribute correspondence between the set of attributes X and Y. This generation of CAs occurs at acquaintance time.

Example 2. Consider the instances in Figure 2 and the mapping tables in Figure 4. Suppose the Family Doctor peer has the following export schema:

V = $\{OHIP, Lname, Fnane, TestName, Result\}$

Also suppose the Hospital peer has the following export schema:

V = $\{patid, name, test, result\}$

Therefore we create the following correspondence assertions.

CA1: OHIP⟶PATID, CA2: Name, Fname⟶Name, CA3: TestName⟶Test

5.2 Translation Rule

In this section we introduce some query translation rules that are used to translate query search conditions and their data vocabularies. For example consider the query Q3 and Q4. In the query Q3, the search condition for patient name is given using (Lname="Lucas", Fname="Andrew") and test name is given using (Test="whitebloodcount"). But the patient name is represented in peer *Hospital* is represented in different format and the condition should be translated as name='Lucas, Andrew' to be recognized in peer *Hospital*. Also the test name "whitebloodcount" is represented in peer *Hospital* with different data vocabulary "C0427512". For this purpose we need some kind of translation rules to resolve these heterogeneity. In this paper we address four translation rules are named as Merge (M), Separation (S), Data Association (DA), and Data Conversion (DC). Each one of the following sections describes each one of these query translation rules and shows how these rules translate query search conditions.

Merge Rules (M). This rule resolves differences in format. Therefore, we need a mechanism to translate the data format in the selection formula of the source query to the format of the formula that represents the target selection formula. We represent a merge rule as follows:

$$\sigma_{\bigwedge A_i=x_i} : \sigma_{B=y} : y = \Pi_B(R' \bowtie T_{\bigwedge A_i=x_i}), \tag{1}$$

where $R' = f_M(R(A_1, \cdots, A_n), B)$; $\sigma_{\wedge A_i = x_i}$ is a pattern of a selection formula in the source query and $\sigma_{B_i = y}$ is the translated selection formula for the target query. The value of y for attribute B is determined by the formula defined in the translation expression of the rule. The semantics of formula above is as follows:

- $f_M(R(A_1, \cdots, A_n), B)$ is a function that creates a temporary relation R' from relation R with attributes $A_1, \cdots, A_n$, and B; $A_1, \cdots, A_n$ are mentioned in $\sigma_{\wedge A_i = x_i}$ and B is the target attribute. Values of attribute B are generated with a user-provided function f that is applied on attributes $A_1, A_2, \cdots, A_n$.
- $T_{\bigwedge A_i = x_i}$ is a tabular representation of the term $\bigwedge A_i = x_i$.

Separation Rule (S). The separation rule is the reverse of the merge rule. The formal representation of the separation rule is:

$$\sigma_{A=x} : \sigma_{\bigwedge B_i = y_i} : y_i = \Pi_{B_i}(R' \bowtie T_{A=x}), \tag{2}$$

where $R' = f_S(R(A), (B_1, \cdots, B_n))$; $\sigma_{A=x}$ is a pattern of a selection formula and $\sigma_{\bigwedge B_i = y_i}$ is the translated selection formula. The value of $y_i's$ for attribute $B_i's$ are determined by the formula defined in the translation expression of the rule. The semantics of formula above is as follows:

- $f_S(R(A), (B_1, \cdots, B_n))$ is a function that creates a temporary relation R' from relation R with attributes A *and* $B_1, \cdots, B_n$; A is mentioned in $\sigma_{\wedge A = x}$ and $B_i's$ are the target attributes for the translated query. Values of attribute $B_i's$ are generated with a user-provided function f that is applied on attributes A.
- $T_{A = x}$ is a tabular representation of the term $A = x$.

Data Association Rule (DA). Our third translation rule translates queries based on mapping tables. This rule deals with data level heterogeneity. When a query mentions a data value that differs from data values used in an acquainted peer, then we need to translate the predicate term in such a way that other peers are able to decipher it. We use mapping tables to translate this type of predicate term. The formal representation of the rule is:

$$\sigma_{\bigwedge A_i = x_i} : \sigma_{\bigwedge \bigvee B_j = y_j} : y_j = \Pi_B(m(A, B) \bowtie T_{\wedge A_i = x_i}), \tag{3}$$

where $\sigma_{\wedge A_i = x_i}$ is a pattern of a selection formula in a local query q and $\sigma_{\vee B_j = y_j}$ is the translated selection formula for target queries, and A and B are sets of attributes. The value of y_j for attribute B_j is determined by the formula $\Pi_B(m(A, B) \bowtie T_{\wedge A_i = x_i})$. Here, attributes A_i are those mentioned in $\sigma_{\bigwedge A_i = x_i}$.

Data Conversion (DC). This rule is used for translating data from one domain to another. For example, consider an attribute $height - in - inches$ from one database and an attribute $height - in - centimeters$ from another.

The value correspondence of these two attributes can be constructed by defining a conversion function f. Formally we represent this rule as follows:

$$\sigma_{A\ op\ x} : \sigma_{\vee B\ op\ y} : y = \Pi_B(R' \bowtie T_{A\ op\ x}), \tag{4}$$

where $R' = f_{DC}(R(A), B)$; $\sigma_{A\ op\ x}$ is a pattern of a selection formula in a local query q and $\sigma_{B\ op\ y}$ is the translated selection formula for the target query; f_{DC} is a user-defined function for data conversion; and $T_{A\ op\ x}$ is the tabular form of A *op* x. If the domain of attribute A and B are same, then function f_{DC} is called an identity function.

6 Query Translation Process

We start our query translation process defining the relationship between correspondence assertions and translation rules. We define the relationship between a correspondence assertion and a translation rule as a pair (CA, R). We use the following syntax to represent the relationship between CA and translation rule R.

$$\text{A} \longrightarrow \text{B} : \text{r}$$

Where $A \subset V$ and $B \subset V'$. V and V' are the exported schemas from peer P and P'. Each rule $r \in \mathcal{R}$ defined in section 5.2 describes the relationship between attributes of correspondence assertions and how the vocabularies are translated in terms of structural, format, and data values. The translation process starts when a user poses a query on its local database and wants a global execution of the query. query to its acquaintances. The algorithm to translate queries is shown in Figure 5. The algorithm mainly translates the selection part of a query. There are two steps for translating a selection of a query. Firstly, the query is analyzed syntactically and the query condition is transformed into a *query condition tree* which is defined as follows.

Definition 2. *Suppose q is a query. The Query Condition Tree(QCT) of q is a tree representing the selection formula of q, where the inner nodes are boolean operators AND, OR, or NOT, and leaves are atomic conditions of the form A op x, where A is an attribute, op is a comparison operator* $<, >, =, \leq, \geq$, *or* $\neq$, *and x is a constant value.*

Secondly, the query condition part is decomposed in terms of the correspondence assertions. The function *FindPartition* performs this task. The function *FindPartition* is shown in Figure 6. The algorithm takes as input a query tree and predefined set of correspondence assertions. Its output is a set $P = \{P_1, ...P_n\}$, where each P_i is a triple (T, CA, R); T is a set of predicate terms that are resulted from partitioning the original query predicate terms; CA is the corresponding correspondence assertion that maps the terms in T, and R is the corresponding translation rule that will be used to translate the predicate terms in T. The function *FindPartition* finds the potential partitions that

QueryTranslation (Q)
Input: A query Q from user.
Output: Translated query Q'
begin QC_T = Create AND-OR-NOT tree from the
query conditions;
CM = FindPartition(QC_T);
/*CM: Set of matchings for query conditions */
for each $cm_i \in CM$ **do**
Apply rule r_i associated with cm_i;
Translate the constraint using translation rules;
Translate (cm_i);
end for
end

Fig. 5. Query translation main procedure

can be translated independently. That means partitions are disjoint and there is no dependency between the terms in the partitions. Consider the following complex query:

Q5: select lname, fname, result from Patient,MedicalExam
where Patient.OHIP=MedicalExam.OHIP AND
(((lname="Hall" OR lname="Hull") AND (fname="Andrew")) OR
(OHIP="233GA")) AND (TestName="whitebloodcount" AND Date="Jan/04")

where the predicate P is as follows:

((((lname="Hall" OR lname="Hull")AND(fname="Andrew"))OR(OHIP="233GA"))
AND (TestName="whitebloodcount" AND Date="Jan/04"))

Consider that we have following associations between correspondence assertions and rules:

ca1: $lname \longrightarrow name$: 4, ca2: $lname, fname \longrightarrow name$: 1
ca3: $OHIP \longrightarrow PAITD$: 3, ca4: $TestName \longrightarrow Test$: 3
ca5: $Date \longrightarrow Dt$: 4

We assume that correspondence assertions ca1, ca5 are bound to rule 4(Data Conversion), ca2 is bound to rule 1(Merge Rule) and ca3, ca4 are bound to rule 3(Data Association). Finding potential partitions is important, because the composition of two terms in a query predicate may map to a particular correspondence assertion. Also sometimes, it is possible that a term can not be translated independently but only in combination with another term can the composed term be mapped with a correspondence assertion. The lines 2-9 of the algorithm *FindPartition* first find the mappings based on correspondence assertions. At this point we find the following initial mappings for our example:

FindPartition(CTree, CA)
Input: A query condition tree CTree and set of Correspondence Assertions CA.
Output: Potential partitions $P = \{P_1, ...P_n\}$ of constraints begin
1. $CM =$ /* CM is a set of pair (t, ca_i) where t is a atomic term
   and ca_i is corresponding correspondence assertion that covers attribute of t
2. **for each** term t at leaf, $t \in T$ /* T is a set of predicate terms in CTree*/
3. $M = \{\}$ /* M is a set of matches found in CA for term t */
4. **for each** correspondence assertion $ca_i \in CA$
5. **if** $attribute(c) \in attribute(ca_i)$ then
6. $M = M \cup ca_i$
7. $CM = CM \cup (t, M)$
8. **end for**
9. **end for**
10. **for each** $m_i \in CM$
11. $m_k = \varnothing$
12. **for each** $m_j \in CM$ *and* $m_i \neq m_j$
13. **if** $(m_j(ca) \cap m_i(ca)) \neq \oslash$ then
14. $m_k(t, ca) = \{\{m_i(t) \cup m_j(t)\}, \{m_i(ca) \cap m_j(ca)\}\}$
15. $P = P \cup m_k(t, ca)$ /*Forming partition*/
16. $CM = CM - m_j$
17. **end for**
18. **if** $(m_k = \varnothing)$
19. $P = P \cup m_i(t, ca)$
20. $CM = CM - m_i(t, ca)$
21. **end for**
22. **end for**
23. **return** P
end

Fig. 6. Finding partition

lname = "Hall", $M = [lname \longrightarrow name, lname, fname \longrightarrow name]$
$CM1 = [lname = \text{“}Hall\text{”}, [lname \longrightarrow name, lname, fname \longrightarrow name], 4]$
lname = "Hull", $M = [lname \longrightarrow name, lname, fname \longrightarrow name]$
$CM2 = [lname = \text{“}Hull\text{”}, [lname \longrightarrow name, lname, fname \longrightarrow name], 4]$
fname = "Andrew", $M = [lname, fname \longrightarrow name]$
$CM3 = [fname = \text{“}Andrew\text{”}, [lname \longrightarrow name, lname, fname \longrightarrow name], 1]$
OHIP="233GA", $M = [OHIP \longrightarrow PATID]$
$CM4 = [OHIP = \text{“}233GA\text{”}, [OHIP \longrightarrow PATID], 3]$
TestName="whitebllodcount", $M = [TestName \longrightarrow Test]$
$CM5 = [TestName = \text{“}whitebloodcount\text{”}, [TestName \longrightarrow Test], 3]$
Date="Jan/04", $M = [Date \longrightarrow Dt]$
$CM6 = [Date = \text{“}Jan/04\text{”}, [Date \longrightarrow Dt], 4]$

The lines 10-22 perform the task of finding potential partitions from the above mappings. Therefore we get the following partitions.

$P1 = [lname = \text{“}Hall\text{”}, fname = \text{“}Andrew\text{”}, [lname, fname \longrightarrow name], 1]$
$P2 = [lname = \text{“}Hull\text{”}, fname = \text{“}Andrew\text{”}, [lname, fname \longrightarrow name], 1]$

$P4 = [OHIP = \text{“}233GA\text{”}, [OHIP \longrightarrow PATID], 3]$
$P5 = [TestName = \text{“}whitebloodcount\text{”}, [TestName \longrightarrow Test], 3]$
$P6 = [Date = \text{“}Jan/04\text{”}, [Date \longrightarrow Dt], 4]$

Notice that, we can not simply translate the predicate terms independently without looking for the potential dependency from other terms. In the example, the term fname =“Andrew” does not have any correspondence assertion but by combining it with the term lname=“Hall”, we can translate the *subquery* (lname=“Hall”, fname=“Andrew”). Because the terms can be mapped with the correspondence assertion $lname, fname \longrightarrow name$. After rewriting the selection formula we apply, the corresponding translation rule to each newly generated leaves of the *query condition tree*.

The query translation algorithm depends on mapping tables and correspondence assertions. The algorithm translates SPJ queries, where selection formula may contain both conjunctive and disjunctive normal form with negation. Sometimes a situation may arise where a query is not translatable. There are three reasons for this case. First, no mapping exists in the mapping table to translate a predicate term. Second, there is no correspondence assertion that maps a predicate term. Third, there is no rule to support the translation of a predicate atom. In all such cases the query is rejected.

6.1 Sound Translation of Queries

In peer database systems, the translation of a given query is not unique [14]. There could be many possible global queries for a particular query, because there is no certain control of query propagation in peer database networks. Also peers are free to join and leave the system at will. Therefore, in most cases, we do not get *complete answer* (meaning all the matching tuples in the network), but at least we get some answer (sound answer) which can be recognized as complete enough with respect to the set of active peers.

In this section we describe the soundness (correctness) of query translation. Soundness ensures that the translated query retrieves correct data from acquainted peers which are relevant to the result of the original query. Consider two peers $P1$ and $P2$ with export schemas $V_1[U_1] \subseteq DB_1[W_1]$ and $V_2 \subseteq DB_2[W_2]$, respectively. Assume that a query translation rule r and a mapping table m exist along with correspondence assertions between the peers. Suppose that a query q_1 is posed over V_1 and that q_2 is the translation of q_1 over V_2. We use the notation $q_1 \longmapsto q_2$ to state that q_2 results from the translation of query q_1. Intuitively, ensuring a correct translation means that the translation should be such that q_2 retrieves from $P2$ only the data that are related to those that could be retrieved from query q_1 in peer $P1$. It is important to establish such a notion of correctness. Here, we extend the definition of correctness of query translation with respect to mapping tables given in [14].

Definition 3. *[Soundness w.r.t to Mapping Tables] Let q_1, q_2 be queries over peer P_1 and P_2, respectively; Let $q_1 = \sigma_E(R_1 \bowtie .... \bowtie R_k)$, where E is a selection*

formula and $R_1, \ldots, R_k$ are relations in P_1. Then q_2 is a sound translation of q_1 with respect to a set $\mathcal{M} = \{m_1(X_1, Y_1), \cdots, m_k(X_k, Y_k)\}$ of mapping tables, denoted by $q_1 \overset{\mathcal{M}}{\longmapsto} q_2$, where $X_i, i = 1..k$, and $att(q_1)$ are subsets of $\bigcup_{i=1}^{k} X_i$, if for every relation instance r_2 of P_2 and $t_2 \in q_2(r_2)$, there exists a valuation ρ of $\mathcal{M}$, and a tuple $t \in \sigma_E(\rho(\mathcal{M})), i = 1 \ldots k$ such that $\pi_{att(q_2)}(t) = t_2$.

In this paper the translation algorithm incorporates the mapping tables into data association rules. Formally we must extend the definition of soundness to incorporate the translation rules seen in Section 5.

Definition 4 (Soundness w.r.t Translation Rules). *Let q_1, q_2 be queries over peer P_1 and P_2, respectively; Let $q_1 = \sigma_E(R_1 \bowtie .... \bowtie R_k)$, where E is a selection formula and $R_1, \ldots, R_k$ are relations in P_1. Then query q_2 is a sound translation of query q_1 with respect to a set of translation rules $\mathcal{R}$ and correspondence assertions $CA = \{ca_1, \cdots, ca_n\}$ between $P1$ and $P2$, denoted by $q_1 \overset{\mathcal{R}, CA}{\longmapsto} q_2$, if $att(q_1) \subseteq att(CA)$, $att(q_2) \subseteq att(CA)$, and for every relation instance I_2 of P_2 and $t_2 \in q_2(I_2)$, there exists a tuple $t \in q_1(I_1)$, where I_1 is an instance of P_1, such that for all $r \in \mathcal{R}$,*

1. *if r is a merge rule (See rule (5.2)), then, for all selection terms $\sigma_{B=y}$ mentioned in q_2 there is a complex selection term $\sigma_{\bigwedge A_i = x_i}$, $1 \leq i \leq n$, mentioned in q_1, such that*

$$y = \Pi_B(f_M(R(A_1, \cdots, A_n), B) \bowtie T_{\bigwedge A_i = x_i}).$$

2. *if r is a data association rule, then use Definition 3.*

The cases of the separation and data conversion rules are treated similarly to the merge rule.

7 Implementation

We implemented the query translation algorithm described in this paper. The architecture of the query translation framework is depicted in Figure 7. The translation process starts when a user poses a query through the graphical user interface and selects the global execution of the query. If the query is local then the query is processed locally. The main component of the architecture is the Query Translation Component which is the implementation of the algorithm in Figure 5.

The monitor component looks for incoming queries in the network. It receives and forwards queries to the acquainted peers.

The prototype P2P setting is shown in Figure 8. We use MySQL as our DBMS for the Local DB. Data have been collected from United Airline, Air Canada, KLM, Luftansa, Air France and Alitalia flight information. There are around 50 mapping tables to map flight numbers, destinations, etc between partner airlines. Each mapping table contains an average of 150 records. We choose JXTA for implementation platform because it provides all resources and functionalities

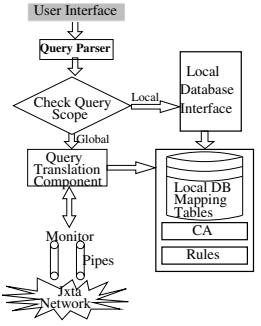

Fig. 7. Architecture

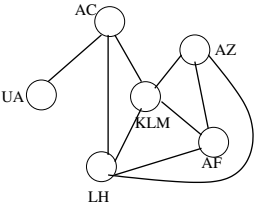

Fig. 8. The P2P Network

for developing P2P application. For example, it provides basic protocols and communication links (called pipes) between peers. It also provides basic peer functionalities such as creation a peer on a network; creation of messages and messages communication onto pipes, discover peers, creation of peer groups and join a peer group, etc. JXTA is an open network computing platform for P2P computing. It provides IP independent naming space to address peers and other resources. Java is used for the programming language. The P2P platform is simulated on IBM computers with windows XP operating system. The CPU is Pentium 4 3.0 GHz and RAM is 760MB.

7.1 Experimental Results

To evaluate our algorithm, we measure various run times. We first find mapping times of query predicates because query translation mainly depends on the size of predicate in the query. shows a query and a resulting translated query. Figure 10 shows the mapping times and predicate translation times of queries.

```
****Original Query***
select lname,fname,address from lh,customer where (fno='LH402' OR fno='LH404')
OR (date='10/06/2003' AND deptime='02') AND dest='shanghai'
Parse Time=0.015
Elapsed time of Mapping:0.016
Elapsed time of PredicateTranslation:0.062
Elapsed time of Query Translation:0.093
*****Translated Query:****
select name, city,pcode from ac, passenger where (((fno='AC700') OR (fno='AC702
) OR (fno='AC704') OR (fno='AC706') OR (fno='AC712') OR (fno='AC714') OR (fno='
C716') OR (fno='AC718') OR (fno='AC720') OR (fno='AC724'))) OR (((fno='AC700')
R (fno='AC702') OR (fno='AC704') OR (fno='AC706') OR (fno='AC712') OR (fno='AC7
4') OR (fno='AC716') OR (fno='AC718') OR (fno='AC720') OR (fno='AC724'))) OR ((
date=10/05/2003 AND deptime=20))) AND (dest='shanghai')
```

Fig. 9. An output of a translated query

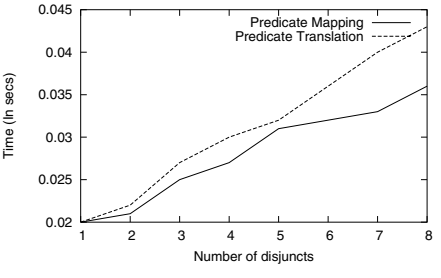

Fig. 10. Predicate mapping and translation

We run this experiment with 8 queries where the number of search conditions gradually ranges from 1 to 8. That is, the first query has one condition,

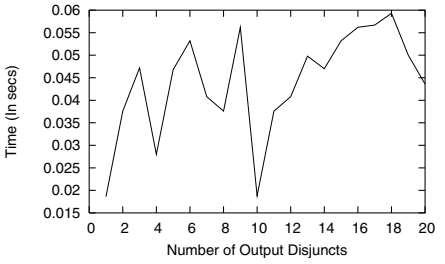

Fig. 11. Query translation time with size of output queries

```
***Original Query***
select * from lh where fno='LH9766'
****Translated Query:****
For Peer:ac
select fno, date,deptime, dest from ac where (((fno='AC700') OR (fno='AC702')
 (fno='AC704') OR (fno='AC706') OR (fno='AC712') OR (fno='AC714') OR (fno='AC7
') OR (fno='AC718') OR (fno='AC720') OR (fno='AC724')))
For Peer:klm
select fno, date,deptime, dest from klm where (((fno='KL0641') OR (fno='KL0643
'))
For Peer:af
select fno, date,deptime, dest from af where (((fno='AF804') OR (fno='AF806')
 (fno='AF808') OR (fno='AF822') OR (fno='AF6426') OR (fno='AF6498') OR (fno='A
F90') OR (fno='AF8994')))
For Peer:az
select fno, date,deptime, dest from az where (((fno='AZ00610') OR (fno='AZ0761
')))
Time Elapsed:0.375
```

Fig. 12. Query translation for different peers

the second two condition, and so on. We see from the figure that the mapping times increase gradually. The queries are chosen in such a way that there are interdependency between predicate terms in the query. Nicks on the curves are points where there is a change in the amount of interdependency among predicate terms. between the time required to perform a query translation and the size, in terms of predicate terms, of the translated (or output) query. Figure 11 shows the translation times for queries which gradually produce a number of terms ranging from 1 to 20. The interesting point to notice from Figure 11 is that it is not the number of terms in the output query which a relevant factor for the running times to generate these terms. These times mainly depend on the number of terms in the input query and on the dependency between predicate terms.

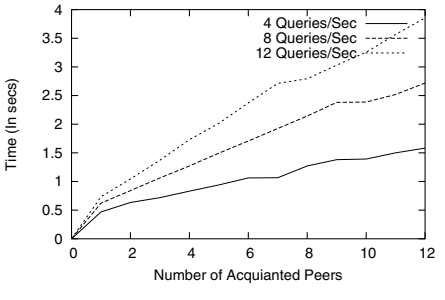

Fig. 13. Query translation time with number of peers

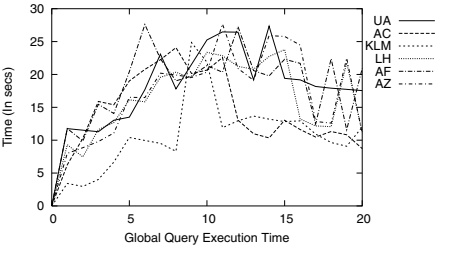

Fig. 14. Global query execution time

We also investigate the algorithm performance for query translation with the number of acquaintances. We experimented with 12 peers. We investigated the times required to translate one query (in a peer) for all the acquainted peers. Figure 12 gives a screenshot showing such a translation. We investigate with different input query frequencies. We notice that the translation times increase gradually with the number of peers and number of input queries per second. The Figure 13 shows the result.

We also investigate the execution of global queries generated from different peers in our P2P settings. We flooded the network generating 4 queries/sec from each peer with total 20 queries per peer. The total number of queries in the network was 624. The result is shown in Figure 14. The figure shows that the global queries of the KLM and LH peers finish first because they have more acquaintances than other peers in the settings. The global queries of the UA peer take the highest times because this peer is linked to the P2P network over one single acquaintance (with peer AC).

8 Related Work

Ooi et al. [16] present a peer-to-peer (P2P) distributed data sharing system called PeerDB. Query processing in PeerDB is performed through keyword matching and ranking using agents. The keyword-matching strategy in PeerDB may give irrelevant query reformulations because a keyword of a relation may match syntactically with keywords of attributes or relations of other peers without being a semantical match. The user must decide which queries are to be executed. In PeerDB, continuous user involvements are required before the user fetches required data from peers. In our approach, on the contrary, once acquaintances are in place, the user need not worry about them at query time.

The paper [9] introduces a data model called Local Relational Model (LRM) designed for P2P data management systems to describe relationships between two peer databases. The acquaintances between two peers are based upon the definition of coordination formulas and domain relations with in the system. The main goals of the data model to support semantic interoperability in the absence of global schema.

The Piazza system [17] provides a solution for query answering in a peer-to-peer environment, where the associations between peers are expressed as either global-as-view (GAV) or local-as-view (LAV) mapping. All these are schema (as opposed to our instance) level mappings.

An approach for data coordination avoiding the assumptions of global schema is introduced in [21]. The authors of [21] introduce the notion of group and define it as a set of nodes, which are able to answer queries about a certain topic. Each group has a node called Group Manager (GM), which in charge of the management of the metadata in order to run the group [21]. According to their proposal, each query must pass through the group manager. The paper does not mention how to choose a node as a group manager.

9 Conclusion

In this paper we investigated a data sharing strategy through query translation based on syntactic and instance level mappings. We addressed some translation rules based on these mappings. Our strategy can translate a query if there is appropriate correspondence assertions between the schema elements of acquainted peers. A future work, we plan to investigate the approach as a query service

built on top of a large scale peer data management system. We also plan to investigate the dynamic inference of new correspondence assertions from existing ones. Such a dynamic inference mechanism seems necessary to avoid doing so manually when peers join/leave the system.

References

1. The JXTA Project. *http://www.jxta.org*
2. M. Boyd, S. Kittivoravitkul, C. Lazanitis, P. McBrien, N. Rizopoulos. AutoMed: A BAV Data Integration System for Heterogeneous Data Sources. In *CAiSE*, 2004
3. R. Domenig, K.R. Dittrich. Query Explorativeness for Integrated Search in Heterogeneous Data Sources. In *CAiSE*, 2002
4. L. Serafini, F. Giunchiglia, J. Molopoulos, and P. Bernstei. Local Relational Model:a logocal formalization of database coordination. Technical Report, Informatica e Telecomunicazioni, University of Trento, 2003.
5. M. Lenzerini. Data Integration: A Theoretical Prespective. In *PODS*, 2001.
6. R. J. Miller, L. M. Haas and M. Hernndez. Schema Mapping as Query Discovery. In *VLDB*, 2000.
7. Z. Ives A. Halevy, M. Rodrig, and D. Suciu. What can databases do for peer-to peer? in webdb. In *WebDB*, 2001.
8. M. Arenas, V. Kantere, A. Kementsietsidis, and I. Kiringa. The hyperion project: From data integration to data coordination. In *ACM SIGMOD RECORD*, 2003.
9. P. Bernstein, F. Giunchiglia, A. Kementsietsidis, and J. Mylopulos. Data management for peer-to-peer computing: A vision. In *WebDB*, 2002.
10. C.-C. K. Chang and H. Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information source. In *SIGMOD*, 1999.
11. F. Giunchiglia and I. Zaihrayeu. Making peer databases interact-a vision. In *CIA*, 2002.
12. A. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management system. In *ICDE*, 2003.
13. V. Kantere, I. Kiringa, and J. Mylopoulos. Coordinating peer databases using ECA rules. In *P2P DBIS*, 2003.
14. A. Kementsietsidis and M. Arenas. Data sharing through query translation in autonomous systems. In *VLDB*, 2004.
15. A. Kementsietsidis, M. Arenas, and R.J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *SIGMOD*, 2003.
16. W. S. Ng, B. C. Ooi, K. L. Tan, and A. Y. Zhou. PeerDB:A p2p-based system for distributed data sharing. In *Data Engineering*, 2003.
17. I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, and X. Dong. The piazza peer data management project. In *ICDE*, 2003.
18. P. reynolds and A. Vahdat. Efficient peer-to-peer keyword searching. In *Middleware*, 2003.
19. E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. The coDB Robust Peer-to-Peer Database System. In *SEDB*, pages 382-393, 2004.
20. E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. Queries and Updates in the coDB Peer to Peer Database System. In *VLDB*, pages 1277-1280, 2004.
21. F. Giunchiglia and I. Zaihrayeu. Making Peer Databases Interact-A vision for an Architecture Supporting Data Coordination. In *CIA*, pages 18-35, 2002.