Data Sharing in the Hyperion Peer Database System

Patricia Rodríguez-Gianolli[1] Maddalena Garzetti[4] Lei Jiang[1]
Anastasios Kementsietsidis[3] Iluju Kiringa[2] Mehedi Masud[2]
Renée J. Miller[1] John Mylopoulos[1]

[1]University of Toronto, Toronto, Canada - {prg,leijiang,miller,jm}@cs.toronto.edu
[2]University of Ottawa, Ottawa, Canada - {kiringa, mmasud}@site.uottawa.ca
[3]University of Edinburgh, Edinburgh, UK - akements@inf.ed.ac.uk
[4]University of Trento, Trento, Italy - garzetti@itc.it

Abstract

This demo presents Hyperion, a prototype system that supports data sharing for a network of independent Peer Relational Database Management Systems (PDBMSs). The nodes of such a network are assumed to be autonomous PDBMSs that form acquaintances at run-time, and manage mapping tables to define value correspondences among different databases. They also use distributed Event-Condition-Action (ECA) rules to enable and coordinate data sharing. Peers perform local querying and update processing, and also propagate queries and updates to their acquainted peers. The demo illustrates the following key functionalities of Hyperion: (1) the use of (data level) mapping tables to infer new metadata as peers dynamically join the network, (2) the ability to answer queries using data in acquaintances, and (3) the ability to coordinate peers through update propagation.

1. Introduction

Peer-to-Peer (P2P) computing has become popular as an alternative model of distributed computing, compared to traditional client-server architectures. In P2P computing, no centralized control is assumed and communication is based on direct links between nodes, or *peers*, in a distributed network.

This paradigm shift aggressively promotes the direct sharing of data between peers, since each peer is now assumed to be both a producer and consumer of data. Within this paradigm, database researchers have sought to develop techniques for data management, assuming that peers are (or include) databases [1]. In the Hyperion project [2], each peer includes a database with its own schema and data. Peers can join or leave the network at their own discretion. Moreover, a peer may form an *acquaintance* with another peer, for data sharing purposes. Peers belong to *interest groups*, such as physicians, medical laboratories or airline companies. When peers become acquainted, logical metadata necessary to allow data sharing are exchanged semi-automatically. These metadata take the form of mappings, both at the data level and schema level, and they help to bridge semantic and syntactic heterogeneities between peers. Metadata at the data level are expressed as *mapping tables* [4]. Mapping tables specify correspondences between data values of acquainted databases.

Run-time management of metadata provides the basic layer on top of which higher-level services can be supported. Our demo offers instances of such services in the form of query translation and update propagation mechanisms.

Basic and higher-level services are supported in each peer by augmenting a conventional (relational) DBMS with a P2P layer that lets peers use each other's data, despite the fact that the underlying databases are heterogeneous. Such a layer plays the role that interoperability layers play in traditional multidatabase or federated systems. We call the DBMSs augmented in this way Peer DBMSs (PDBMSs for short). Contrary to traditional multidatabase or federated systems, Hyperion supports a *dynamic* network of peer DBMSs that use their P2P layers to coordinate and share data. Traditional systems do not handle gracefully the arrival or departure of peers. Adding a peer to an existing federation often results in the re-organization of the federated schema and issues of heterogeneity between the federated peer sources may need to be revisited.

In Hyperion, the interoperability layer addresses heterogeneity issues between pairs of acquaintances. Moreover, the system is able to leverage at run-time, pair-wise acquainted peers in order to support data sharing among peers that are indirectly connected in the network.

Another distinguishing characteristic of Hyperion is that it addresses the problem of sharing data at the data level (in

**Proceedings of the 31st VLDB Conference,
Trondheim, Norway, 2005**

terms of mapping tables) between heterogeneous sources. This means that the mechanisms supported by Hyperion are driven by correspondences between data values, rather than schema information, for data sharing. In this respect, data sharing stands in contrast to mechanisms used for data integration [9] and data exchange [10] that define the relationships between sources in terms of mappings at the schema level. Different logical interpretations of schema mappings have been used in P2P projects (Piazza [7], coDB [11], Hyper [12], etc.) and in many integration and exchange projects (Information Manifold [13], Clio [14]). However, data sharing deals with the exchange of data that may represent different real world domains where mappings cannot always be specified at the schema level.

2. System Architecture

The logical architecture of Hyperion, inspired by [8], is presented in Figure 1. A *Hyperion P2P Database Network* consists of a set of peer nodes which participate in data sharing by clustering themselves into interest groups (dashed ovals in Figure 1) and establishing pair-wise acquaintances between them (arrows connecting nodes in Figure 1). We assume that all peer nodes have identical architectures, that is: each peer node conforms to the Hyperion Peer Database System architecture.

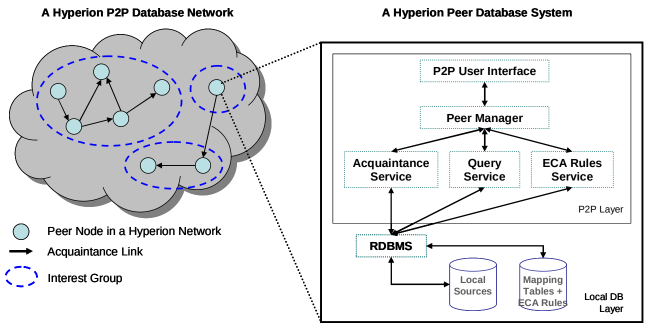


Figure 1: Logical Architecture of Hyperion

A *Hyperion Peer Database System* consists of a *P2P Layer* and a *Local Database Layer*. The former facilitates peer-to-peer data sharing by taking data residing in acquainted peers and resolving the semantic heterogeneity using mapping tables and ECA rules. The latter encompasses typical local database functionality, such as managing access (queries and updates) to local data sources and the collection of mapping tables and ECA rules that relate local data to data residing in acquainted peers.

The P2P Layer uses the following modules:

- **P2P User Interface**: this is the interface where queries are posed to the system. Queries may be either local or global, meaning that data should be locally retrieved or both locally retrieved and complemented with additional data from acquainted peers. We assume that the user is unaware of schemas of remote PDBMSs and formulates his queries only in terms of the local schemas. Finally, this interface is also used to specify distributed ECA rules describing the patterns of data coordination between acquainted peers.
- **Peer Manager**: this module handles a set of extensible services offered by a peer node. A service encapsulates a piece of distributed computation performed by the peer on behalf of its acquaintances. Following an ad-hoc P2P application pattern, each service implements its own messaging system to carry out local or remote requests.
- **Acquaintance Service**: this module manages the exchange of public schemas, mapping tables, and coordination rules between new acquaintances and the inference of new mappings.
- **Query Service**: this module provides the ability to execute local and global queries. Local queries are executed as in traditional DBMSs; global ones are executed by applying query rewriting in terms of the schemas and mapping tables of acquainted peers (please refer to Section 3 for details).
- **Peer Coordination Service**: this module manages and executes distributed ECA rules in order to enforce consistency policies and coordinate updates between peers. Our rule mechanism decomposes each rule into sub-rules, one for each peer involved in the rule's event expression. See [3] for details.

3. Algorithms

Our demonstration illustrates a number of new algorithms used to achieve P2P data sharing. First, we present an acquaintance-time algorithm that infers new mapping tables from existing ones. This algorithm is based on an optimized, distributed semi-join-like strategy that respects the semantics of mapping tables. Second, we illustrate our query translation algorithm which may use potentially large mapping tables. Once computed, query translations are stored for reuse within other computations. Finally, we illustrate Hyperion's update mechanism which permits local updates to be translated (using mapping tables) and propagated to acquaintances. The Hyperion prototype implements execution semantics for distributed ECA rules [3].

There is a basic algorithm in [4] for generating mapping tables that has the following feature: Mapping tables are generated on demand, i.e., entire tables are generated in each step from existing ones. The input to the algorithm is (1) a path P,P1,P2,…,Pn,P' of peers going from a peer P to a peer P' over intermediary peers such that there is a set of mapping tables between two consecutive peers on the path, and (2) two subsets U and U' of attributes of P and P', respectively. The output of the algorithm is a set of mapping tables linking peers P and P'. A naïve algorithm works as follows: peer P forwards all the mappings between itself and P1 to the latter which uses its own mappings and the mappings received from P to compute the mappings between P and P2. Then, P1 sends the resulting mappings between P and P2 to P2. This computation is repeated until the penultimate peer Pn is reached, at which point Pn computes mappings between P and P' and sends them back to P. However, this algorithm has two major drawbacks: first, the algorithm can forward mapping tables between the peers that might prove to be useless for the computation and second, the algorithm fails to take advantage of the distributed nature of the system since it utilizes the resources of one peer at a time. An improved algorithm is described in [4] to remediate these drawbacks. The improved algorithm consists of an information gathering phase during which information is collected to help in a subsequent computation phase that streams computed mappings between peers.

The algorithm for query translation using multiple mapping tables is described in [5]. The algorithm supports Select-Project-Join queries, where the selection formula is positive. To translate a query, the algorithm represents the query as a T-Query. A T-query is a tabular representation of the query. This representation is used because of uniformity with the representation of mapping tables. The paper [5] presents algorithms to compute both sound and complete translations of a query.

3.1 An Example

As an example of a domain that Hyperion can be applied to, consider a physician prescribing medications. The prescribing physician may need to know what medications her patient is taking, what the patient's white blood cell count is, and other details of the patient's medical history. This information may be stored not only in the prescribing physician's database but also in the database of an associated specialist physician, medical laboratory, or pharmacy.

Figure 3 shows partial instances of databases for this scenario. Peer databases belong to physicians, hospitals, medical laboratories and pharmacies. Acquaintances are established between associated physicians, between physicians and associated laboratories, and so on. The example databases shown are those of a family physician Dr. F and a medical laboratory Lab A, whose database schemas are as follows:

```
Dr_Patients(ohip,name,primarydr)
Dr_Tests(tid, type, class, test, result, ohip)
LabA_Patients (pid, name, referring)
LabA_Results (testid, test, result, pid)
```

Call these databases DrF_DB and LabA_DB. Figure 4 shows examples of mapping tables used to map data between the peer databases. For each row in these tables, the value on the left side of the double vertical bar is mapped to the value on the right side. Based on the mapping tables, acquainted peers can use the contents of each other's databases to answer queries. In our example, a user intending to find out the results of any test of white blood cell count for L. Davidson could issue the query:

```
select result
from DrF_Tests
where ohip="5017266094NE" AND test="whitebloodcount"
```

Ohip	Name	Primarydr
2330447896GA	A. Lucas	Goldbach
5017266094NE	L. Davidson	F

(a) DrF_Patients Instance

tid	class	test	result	ohip
H6117	hem	whitebloodcount	9755 c/mcL	5017266094NE
H8250	hem	hemoglobin	14.6 g/dL	3074550527GA

(b) DrF_Tests Instance

pid	name	referring
243-23-6572	L. Davidson	Jensen
359-00-4711	Gonzalez	Barton

(c) LabA_Patients Instance

testid	test	result	Pid
4520	C0427512	6339 c/mcL	243-23-6572
4521	C0518015	12.5 g/dL	243-23-6572

(d) LabA_Results Instance

Figure 3: Instances of two databases

Using the mapping tables, this query is expressed in terms of the LabA database schema as follows:

```
select result
from LabA_Results
where pid="242-23-6572" AND test="C0427512"
```

DrF.ohip	**LabA.pid**
5017266094NE	243-23-6572
2330447896GA	388-17-8848

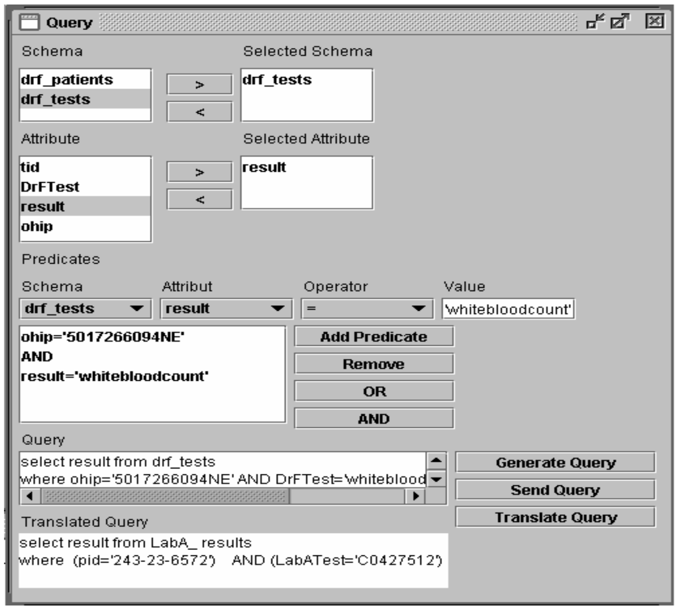
DrF.test	**LabA.test**
hemoglobin	C0518015
whitebloodcount	C0427512

Figure 4: Mapping Tables

4. Demonstration

We implemented a prototype of our Hyperion System on top of JXTA [6]. JXTA is an open network computing platform for P2P computing. It provides a common set of protocols and an open source reference implementation for developing P2P applications. We used MySQL as our DBMS for the Local DB Layer.

In the demo, we demonstrate the main functionalities of our prototype running several peers simultaneously. Specifically, the prototype provides the necessary functionality to form interest groups dynamically. Peers can only communicate and share data with each other after they are acquainted. An acquaintance is an abstraction of a communication channel between peers. Peers can establish acquaintances within and across interest groups. Once two peers are acquainted with each other, they can share and exchange data by using each other's services. A service encapsulates a piece of distributed computation offered by the peer on which it is running. Using the Acquaintance, Query and Update services, we demonstrate how queries and updates propagate in the system by following the connectivity graph of acquainted peers.



```
ohip='5017266094NE'
AND
result='whitebloodcount'
```

```
select result from drf_tests
where ohip='5017266094NE' AND DrFTest='whiteblood
```

```
select result from LabA_ results
where  (pid='243-23-6572')   AND (LabATest='C0427512')
```

Figure 5: Querying Interface

The following assumptions are made for the demonstration:

- Each peer node is equipped with a library of schemas (and their corresponding databases) and initial mapping tables at acquaintance time.
- Peers are using both the bulk generation of mapping tables as well as the streaming version of the algorithm.

We expect viewers to be able to see newly generated mappings being streamed back to the peer that initiates a mapping table inference. They will also be able to pose queries and get back answers accumulated from peers throughout the Hyperion network. Finally, they can update data.

As an example, Figure 5 shows the querying interface. This interface allows a user to write a query for a local peer database. The system translates queries based on the peer's mapping tables. Then the user can send all translated queries by clicking the send button. A similar interface is used for update propagation.

References

[1] Special Issue on P2P Data Management. In SIGMOD Record 32(3), 2003.

[2] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. Miller and J. Mylopoulos. The Hyperion Project: from Data Integration to Data Coordination. In SIGMOD Record 32(3), 2003.

[3] V. Kantere, I. Kiringa, J. Mylopoulos, A. Kementsietsidis and M. Arenas. Coordinating Peer Databases using ECA Rules. In DBISP2P, 2003.

[4] A. Kementsietsidis, M. Arenas and R. Miller. Managing Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. In SIGMOD, 2003.

[5] A. Kementsietsidis and M. Arenas. Data Sharing through Query Translation in Autonomous Systems. In VLDB, 2004.

[6] Project JXTA, 2004. http://www.jxta.org

[7] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau and P. Mork. The Piazza Peer Data Management Project. In SIGMOD Record, 32(3):47-52, 2003.

[8] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini and I. Zaiharayeu. Data Management for Peer-to-Peer Computing: A Vision. In WebDB, 2002.

[9] M. Lenzerini. Data Integration: a Theoretical Perspective. In PODS, 2002.

[10] R. Fagin, P. Kolaitis, R. Miller and L. Popa. Data Exchange: Semantics and Query Answering. In ICDT, 2003

[11] E. Franconi, G. Kuper, A. Lopatenko and I. Zaihrayeu. Queries and Updates in the coDB Peer to Peer Database System. In VLDB, 2004.

[12] D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati and G. Vetere. Hyper: A Framework for Peer-to-Peer Data Integration on Grids. In ICSNW, 2004.

[13] A. Levy, A. Rajaraman and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In VLDB, 1996.

[14] C. Yu and L. Popa. Constraint-Based XML Query Rewriting for Data Integration. In SIGMOD, 2004.