The Hyperion Project: From Data Integration to Data Coordination

Marcelo Arenas[1] Vasiliki Kantere[1] Anastasios Kementsietsidis[1]

Iluju Kiringa[2] Renée J. Miller[1] John Mylopoulos[1]

[1] Dept. of Computer Science
University of Toronto

[2] School of Inf. Technology
and Engineering
University of Ottawa

ABSTRACT

We present an architecture and a set of challenges for peer database management systems. These systems team up to build a network of nodes (peers) that coordinate at run time most of the typical DBMS tasks such as the querying, updating, and sharing of data. Such a network works in a way similar to conventional multidatabases. Conventional multidatabase systems are founded on key concepts such as those of a global schema, central administrative authority, data integration, global access to multiple databases, permanent participation of databases, etc. Instead, our proposal assumes total absence of any central authority or control, no global schema, transient participation of peer databases, and constantly evolving coordination rules among databases. In this work, we describe the status of the Hyperion project, present our current solutions, and outline remaining research issues.

1. INTRODUCTION

The concept of peer-to-peer (P2P) computing is being touted as an architecture for Internet networking that favors a direct and dynamic node-to-node model of communication. Peers use a dedicated naming space, act as both clients and servers, and have primary responsibility for the content provided over the network. Peers are also autonomous with respect to the control and structuring of the network. New peers may join or leave the network at any time. Once a peer joins the network, it can establish *acquaintances* with other peers. Acquainted peers can then share and coordinate data. Of course, acquaintances are transient too. Though the P2P paradigm has found popular applications, mainly for file sharing (e.g., Gnutella [1] and Kazaa [2], to name just the most celebrated), P2P applications generally treat the shared item, e.g. a file, as a unit and do not manage its content. More specifically, existing P2P applications still largely ignore data management issues.

In this paper, we present an architecture for *peer database management systems* (in the sequel: PDBMSs). This architecture instantiates the vision of logical P2P data coordination laid out in [4]. We envision a PDBMS as a conventional DBMS augmented with a P2P interoperability layer. This layer implements the functionality required for peers to share and coordinate data without compromising their own autonomy. Elements of this functionality may be incorporated into conventional DBMS, but for conceptual clarity we present it here as a separate layer. The P2P layer allows a PDBMS to establish or abolish an acquaintance (semi-)automatically at runtime, thereby inducing a *logical* peer-to-peer network.

Our focus in this work is on the specification and management of the logical meta-data that enables data sharing and coordination between independent, autonomous peers. Key elements of our vision include the following.

- We consider data coordination in which each source behaves as an access point for both local and shared data. Unlike more traditional data integration systems [11], we are not considering the problem of providing uniform access to a collection of heterogeneous data. Rather, each peer defines an independent access interface, and we consider the specification and management of meta-data required to coordinate and share data. Data coordination may involve the reconciliation and integration of data at query time or the maintenance of consistency in data contained within different peers. To support the former, we present new techniques that enable each peer to define its own view of shared data using constraints placed not on the content of acquainted peers, but on the way in which peers exchange data. To support the latter, we present new techniques for specifying data propagation and coordination policies.
- We consider data sharing both within and across domains. While views and GLAV (global-and-local-as-view) mappings have been used to integrate and exchange data within a common domain [5, 11, 15], we consider what meta-data is

required to share data across multiple worlds [9, 12]. We present new solutions for specifying and managing *mapping tables* [9]. A mapping table associates values residing in different peers and provides a lightweight mechanism for sharing data.

- We propose a distributed active rule mechanism which uses mapping tables and mapping expressions. Unlike similar proposals for the conventional multi-database setting (e.g., [18]), we do not deal with a setting that is static. Instead, we have a highly dynamic and scalable setting. Very few of the existing solutions can cope with this extreme requirement imposed on the peer network in terms of event detection, condition evaluation, and action propagation.

In this paper, we present an architecture that permits lightweight coordination between peers as their data evolve. We also identify the challenges encountered in fleshing out the main components of the architecture. In addition, we outline techniques for some of these challenges, namely for query answering and data coordination.

2. MOTIVATING EXAMPLE

We consider an example drawn from the domain of airline-ticket reservations. In such a setting, peer databases belong to travel agencies and airlines. Acquaintances are established between affiliated travel agencies, between travel agencies and airlines, and between partner airlines or subsidiaries thereof. In the example, we consider two databases that belong to two partner airlines. Their schemas are shown below.

AA_Passenger	(pid, name)
AA_Flight	(fno, date, dest, sold, cap)
AA_Ticket	(pid, fno, meal)

(a) Schema for the Alpha-Air Airline

BA_Fleet	(aid, type, capacity)
BA_Passenger	(pid, name)
BA_Flight	(fno, date, to, sold, aid)
BA_Reserve	(pid, fno)

(b) Schema for the Beta-Air Airline

Alpha-Air stores for each passenger her identifier and name. For each flight it stores the flight number, date, destination city, number of tickets sold, and the capacity of the flight. For simplicity, we assume that all flights shown have the city of Toronto as their origin. Finally, for each ticket sold, the airline stores the passenger identifier, the flight number, and the passenger's meal preference. Beta-Air stores information about its fleet, namely, the identifier of each plane along with its type and capacity. In addition, it stores the identifier and name of each passenger, and the flight number, date, code of destination airport, tickets sold and identifier of the airplane used for each flight. Again, for simplicity we assume that all flights leave from the Toronto airport. Finally, for each reservation, Beta-Air stores the passenger identifier and the corresponding flight number. Figure 1 shows instances of the two databases.

The two airlines can start their interaction by establishing an acquaintance. Through this acquaintance, the airlines aim to both exchange general flight and passenger information and to coordinate their flights to common destinations. However, since their two schemas differ, some form of integration or reconciliation is necessary before any data is exchanged. Traditionally, data integration and exchange between heterogeneous sources is provided mainly through the use of views that map and restructure data between different schemas [11]. As such, views have been effectively used as constraints on the contents of sources where the contents of one source determines the virtual or materialized contents of another. However, in a peer-to-peer setting we assume that the contents of sources are independent of one another and each source brings into the system its own data. In our example, each airline has its own clientele and its own schedule of flights. Thus, what is needed are mechanisms that support the meaningful exchange of information between independent sources. Moreover, such mechanisms should impose constraints on the *exchange* of information between sources instead of on the sources themselves.

In Hyperion, we propose to use *mapping expressions* and *mapping tables* [9] to support information exchange between peers. Figure 2 shows examples of both constructs. Specifically, Figure 2(a) shows an example of a schema mapping expression. Schema mapping expressions are generalizations of GLAV expressions that can be constructed manually or by automated techniques [15]. However, we do not use these expressions to define or restrict the set of valid instances of the peers. Rather, we use these expressions as constraints on the exchange of data. For example, the expression shown in the figure states that, at *query time*, every passenger of Beta-Air is also considered a passenger of Alpha-Air. This is not to say that every passenger in an instance of the former schema must also appear in the latter. Instead, it means that in the event of a query in the Alpha-Air database that asks for all its passengers a query must also be executed in the Beta-Air database to retrieve all the passengers that reside there.

Our previous work has shown the benefits of using mapping tables [9]. In brief, mapping tables respect peer autonomy and can be used for the integration of seemingly unconnected databases, referred to as *mediation across multiple worlds* [12]. Moreover, although mapping tables provide elementary schema-level associations, their primary use is in defining correspondences between values in different peers. This is of particular importance in P2P systems where there are no naming standards, as peers depend on the use of internal naming conventions. Our example illustrates these characteristics. Figures 2(b), (c) and (d) show three mapping tables between the two airline databases. Mapping table 2(b) associates city names in the Alpha-Air database to airport codes in the Beta-Air database, while table 2(c) associates flight numbers of the two databases when the corresponding flights have the same destination city.

AA_Passenger

pid	name
1	Renee
2	Verena
3	Iluju

AA_Ticket

pid	fno	meal
1	AA210	Meat
2	AA378	Veg.

AA_Flight

fno	date	dest	sold	cap
AA210	01/05/03	L.A.	120	256
AA341	01/15/03	N.Y.	160	160
AA378	01/21/03	S.F.	90	124

(a) Alpha-Air Database Instance

BA_Passenger

pid	name
1	John
2	Renee

BA_Fleet

aid	type	capacity
B-1	Boeing 747	340
B-2	Boeing 737	130
B-3	Boeing 737	107

BA_Flight

fno	date	to	sold	aid
BA1023	01/07/03	LAX	67	B-3
BA1078	01/15/03	JFK	118	B-2
BA1109	01/15/03	ORD	164	B-1

BA_Reserve

pid	fno
1	BA1023
2	BA1078
2	BA1109

(b) Beta-Air Database Instance

Figure 1: Instances for the two airline databases

$$\text{AA\_Passenger(p, n)} \quad \supseteq \quad \text{BA\_Passenger(p, n)}$$

Mapping expression 2(a)

dest	to
N.Y.	JFK
N.Y.	LGA
L.A.	LAX
L.A.	ONT

Mapping table 2(b)

fno_{AA}	fno_{BA}
AA210	BA1023
AA341	BA1078
AA341	BA1080

Mapping table 2(c)

date_{AA}	date_{BA}
$\mathcal{X}$	$\mathcal{X}$

Mapping table 2(d)

Figure 2: Mapping tables and expressions

Finally, table 2(d) uses a variable ($\mathcal{X}$) to represent the identity function, i.e., each date value of the first database is mapped to itself in the second. Currently, the creation of mapping tables is primarily a manual process. However, we can envision settings where the tables are created semi-automatically by employing techniques from data-mining. In our own work [9], we are able, given an initial set of tables, to automatically create new tables that are inferred from this initial set.

Once the two databases are acquainted, and mapping tables or expressions are in place, the peers can use each other's contents during query answering. For example, assume that a user of the Alpha-Air database intends to retrieve the dates of flights destined for Los Angeles with a query such as the following:

```
q:  select date
    from   AA_Flight
    where  dest = "L.A."
```

In our framework, query q is *translated* into a query q' that is then executed in the context of the Beta-Air database to retrieve *semantically relevant* information. The translation is guided by any available mapping expressions and mapping tables. The translated query q' may have the following form:

```
q':  select date
     from   BA_Flight
     where  to = "LAX" OR to = "ONT"
```

Note that we can also use these mapping tables to support scenarios where a query in Beta-Air concerning dates of flights to *"LAX"* retrieves dates of flights to *"L.A."* from the Alpha-Air database.

Apart from querying, peers are also able to coordinate their data with those of their acquaintances. For some applications, run-time reconciliation will not be sufficient and we will want to reconcile the data as it is updated. In this example, suppose the two partner airlines wish to reconcile their data to conform to the mapping expression of Figure 2(a). Using this mapping expression, we can derive a rule to ensure the two databases stay consistent as new passengers are entered into the Beta-Air peer. To enforce this rule, and other similar business rules, we propose a mechanism which uses event-condition-action (ECA) rules with the distinctive characteristic that events, conditions and actions in rules refer to multiple peers. An example of such an ECA rule is given below (expressed for convenience as an SQL trigger).

```
create trigger passengerInsertion
after   insert on BA_Passenger
        referencing new as NewPass
for each row
begin
        insert into AA_Passenger values NewPass
        in Alpha-Air DB;
end
```

Notice that an event that is detected in the Beta-Air database causes an action to be executed in the Alpha-Air database. Specifically, each passenger insertion in the BA_Passenger relation generates an event which triggers the above rule. The rule has no condition while its action causes the insertion of an identical passenger

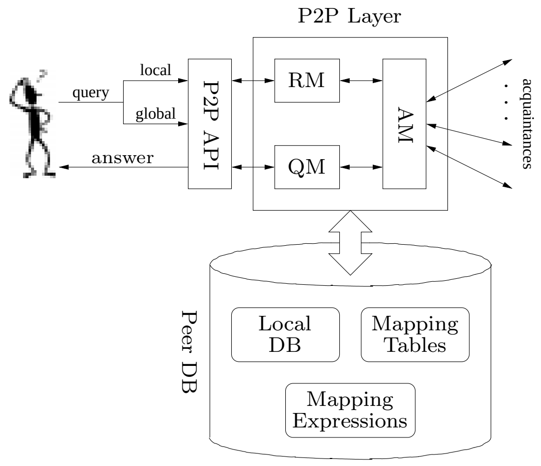

Figure 3: Architecture for a PDBMS

tuple in the AA_Passenger relation. In this work, we consider rule evaluation and decomposition techniques amenable to P2P environments.

3. P2P INFRASTRUCTURE

In this section, we present the components that form the backbone for information sharing in our architecture. This section also summarizes our past work in the Hyperion project. Then, in the next section, we show how the existing system components are to be used to support higher-level services, namely, query answering and active peer coordination.

3.1 Peer Architecture

Figure 3 depicts the architecture of a PDBMS with its main functionalities. A PDBMS consists of three main components: an interface (P2P API), a P2P layer, and a DBMS. The latter component contains the local data and the set of mapping tables and expressions that are used to exchange data with other peers, while the P2P API is the user interface to the whole system. Through the P2P API the user is able, among other things, to pose queries and specify whether these are to be executed only locally or they should also consider the data residing in other peers. The P2P layer in Figure 3 is the key component of the whole system. Its basic modules and their roles have as follows:

Acquaintance manager (AM): This module is responsible for semi-automatically establishing acquaintances among PDBMSs. In more detail, we say that an acquaintance is established between two PDBMSs when a set of mapping tables or expressions is created between them. To create these constructs, the AM relies both on user input and on automated tools for the creation of mapping tables [9]. Using similar functionality, the AM also manages mapping tables and expressions since we do not assume that these are static. Indeed, throughout the lifetime of an acquaintance, mapping tables and expressions change, contributing to the dynamic nature of the system.

Query manager (QM): An important assumption in our approach is that each query is defined with respect to the schema of a single peer. Our thesis is that a user need only be aware, and knowledgeable, of the local schema. In terms of execution, however, queries are classified into two categories. A *local* query is executed using only the data in the local peer, while a *global* query uses the P2P network to complement or reconcile locally retrieved data with data that reside in other peers.

In the case of global queries, the QM rewrites a query posed in terms of the local schema to a query over the schema of acquainted peers. To do so, the QM uses the services of the AM. Specifically, the AM provides the mapping tables and expressions that are to be used for the rewriting.

Rule manager (RM): The objective of the RM is to enforce consistency policies between peers. The policies are declaratively specified through the P2P API and have the form of ECA rules.

3.2 Mapping tables

Mapping tables and mapping expressions are the basic tools for exchanging information between peers. In what follows we offer an overview of our work on mapping tables [9, 10].

Consider two peers that expose attributes U and V respectively. A mapping table is then a relation over the attributes $X \cup Y$, where $X \subseteq U$ and $Y \subseteq V$ are non-empty sets of attributes from the two peers [9]. For example, a mapping table from a set of attributes $X = \{\text{fno}_{\text{AA}}\}$ to a set of attributes $Y = \{\text{fno}_{\text{BA}}\}$ is shown in Figure 2(c). A vertical double line is used to separate the two sets of attributes.

To represent different semantics for mapping tables and values within, we follow the standard convention of using variables. For instance, Figure 2(d) shows a mapping table containing variables. Every valuation of these variables gives us a value of date_{AA} that can be mapped to a value of date_{BA}. Since this mapping table contains the same variable in its two columns, every valuation gives us a tuple of the form (a, a), where a is a constant in the domains of date_{AA} and date_{BA}. Thus, in this case variables offer a compact and convenient way of representing the identity mapping table.

Mapping tables do not restrict the content of peer databases. Instead, they restrict the ways in which information can be exchange between them. Consider relations r_1 and r_2 residing in peers P_1 and P_2 respectively, and also consider a mapping table m from X to Y, where X and Y are subsets of the set of attributes exposed by P_1 and P_2, respectively. Given a valuation ρ of the variables of m, a value $x \in \pi_X(\rho(m))$ is associated with a certain set of values in the domain of Y, namely, with the set $\pi_Y(\sigma_{X=x}(\rho(m)))$. As a result, a tuple $t_1 \in r_1$ such that $t_1[X] = x$ can be mapped, with respect to table m and valuation ρ, only to tuples $t_2 \in r_2$ for which $t_2[Y] \in \pi_Y(\sigma_{X=x}(\rho(m)))$. In general, to know whether a value in the domain of X can be mapped to a value in the domain of Y we need to considered all possible valuations of the variables of m.

By treating mapping tables as constraints on the exchange of information between peers, we are able to automatically check the consistency of a set of mapping constraints and infer new constraints. These functionalities have been shown to be of practical interest in a number of situations including the establishment of new acquaintances [9]. Even more importantly, we have shown that the above functionalities can be implemented efficiently by algorithms whose running time scales gracefully as the number of peers or mapping tables increases.

4. P2P SERVICES

In this section we offer an overview of the higher-level services provided to peers, in the context of Hyperion. We also outline the future research directions of Hyperion and identify some of the main research issues.

4.1 P2P Querying

Querying constitutes the most important service provided in a P2P network. For file-sharing systems, like Gnutella [1] and Kazaa [2], querying involves simple keyword search. So, queries are of the form: "Retrieve all files named X (or containing the phrase X)". This simple form has proven so effective that there has been a flurry of research on making this kind of search more efficient and scalable [16; 17, and others].

In our work, the emphasis is on offering a finer-grain query facility that is comparable to that of conventional DBMS. As illustrated by our example, to accomplish this we query translation or rewriting. In the literature, view expressions between heterogeneous schemas constitute the main vehicle for rewriting queries [11]. We are investigating mechanisms for query rewriting that rely on both mapping tables and expressions. The work reported in [14] is similar in spirit to our proposal. There, the rewriting of queries relies on the use of descriptive keywords which are the only meta-data available between different schemas. Unlike our approach, however, the assumption made is that keywords are used consistently throughout the P2P network to describe relations of the same *type* (e.g. flight relations). Aberer *et al.* [3] introduce a formal framework where the quality of peer mappings is assessed by measuring the quality of query rewritings that are obtained from these mappings. There, peer mappings are functions that map the values of attributes belonging to acquainted peers. Their emphasis is more on the assessment of the quality of the mappings, and less on the mappings themselves and how these can be used to perform the rewriting. Our work, on the other hand, gives emphasis on these latter issues. Thus, the two approaches are complimentary.

A P2P network is generally open-ended and constantly changing. Accordingly, it makes sense to employ two different semantics for queries. A query under the open-world semantics follows traditional P2P query semantics where the information fetched represents a possible answer that is supported by some (though not necessarily all) peers. On the other hand, a query under the closed-world semantics only fetches information that is semantically consistent with information that is locally stored. Our intention is to investigate how the different semantics influence the rewriting of a query.

4.2 P2P Coordination

Active functionality is used in centralized DBMSs to express automatic system reactivity to events raised in a database. We propose to use such functionality in a P2P setting to implement coordination rules [4]. This functionality is necessary to automatically manage consistency and data exchange between peers. We express active functionality with ECA rules that reside in the P2P layer of a PDBMS and involve a peer and some of its acquaintances. Rules make use of mapping tables and expressions to coordinate data and meta-data (for example, the mapping tables themselves) between peers. Our work extends the distributed ECA rule language of Kantere [8].

Consider the following example. Assume that Alpha-Air and Beta-Air have agreed that whenever an Alpha-Air flight is oversold, a new flight should be created by Beta-Air to accommodate new passengers. The following rule captures this behavior.

```
create trigger AAOverSold
before  update of sold on AA_Flight
        referencing new as New
                    old as Old
        in Alpha-Air DB
when    New.sold >= New.cap
begin
        for each row
        insert into BA_Flight values
        (map(New.fno), date, map(New.dest), 0, null)
        in Beta-Air DB
end
```

Notice that the action part of the rule specifies an insertion of a new tuple which is a transformation of the updated Alpha-Air flight tuple. This transformation is accomplished using the mapping tables.

In Hyperion, we envision two sets of rules. The first set includes ones that manage consistency between the data of two peers. These rules are automatically created during the acquaintance phase and enforce the consistency expressed by mapping expressions created during the acquaintance process. Such rules comprise a simple event part and no condition part. Their action part propagates a consistency-restoring update to the appropriate peer database.

The second set of rules is created at query time, after an acquaintance has been established. Such rules support a broad range of automated data exchange. In general, rules of this sort comprise a composite event, condition and action, all of them involving several peers. Query-time rules can be either generic, with suitable parameters, or *ad hoc* user-defined rules. Parameterized rules maybe be made available from an *interest group* during the establishment of an acquaintance.

The *execution model* of the rule language outlined above has to respect the execution autonomy of peers. We propose an execution model that supports distributed evaluation of rules in order to avoid overloading the P2P network with exchanged messages. Our mechanism decomposes each rule into sub-rules, one for each peer involved in the rule's event expression. For each peer database, the event part of its sub-rule comprises the part of the original event that refers to operations related to that peer database. Similarly, the condition part of the sub-rule describes checks to be performed. Finally, the action part of a sub-rule propagates the instance of the event and the condition of the sub-rule back to the peer database that is responsible for the management of the parent rule.

5. RELATED WORK

The *local relational model* for relational P2P database applications has inspired much of this work [4]. In this framework, as in ours, peers in a P2P network are viewed as local relational databases which establish acquaintances to define a P2P network. Moreover, each acquaintance is characterized both by a mapping between the peer databases involved (both at the schema and value level) and by a first-order theory defining the semantic dependencies between the peer databases. In Gribble *et al.* [7], the focus is on the problem of selecting views to materialize, and then selecting where to place these views within a P2P network. The authors show that this problem, which is intractable in general, has some tractable instances. To the best of our knowledge, *PeerDB* [14] is the first implementation of a P2P database management system. PeerDB is a P2P data sharing system built on top of *BestPeer* [13], a generic and self-configurable P2P architecture developed at the University of Singapore. PeerDB is a DBMS that supports fine-grain and content-based searching, integrates mobile agents, and provides a front-end for searching data. Finally, there is a proposal for a preliminary architecture for peer DBMSs [6]. Like ours, it is inspired by the theoretical foundations of [4]. Unlike ours, the focus there is on more user-oriented issues such as communities of interest.

6. CONCLUSIONS

We have described key elements used in the Hyperion project for specifying and managing logical metadata that enable data sharing and coordination between peer DBMSs. First, we considered data coordination where peers are used as access points for both local and shared data. Second, we considered data sharing both within and across domains using mapping tables. Third, we presented a vision for a query and distributed active rule mechanisms which use mapping tables and mapping rules to coordinate data sharing. Ultimately, we envision a P2P database technology that can be used on-the-fly by end-users to establish and exploit acquaintances in support of lightweight, flexible, and cost-effective data coordination in a densely populated database universe.

7. REFERENCES

[1] Gnutella. http://www.gnutelliums.com/.

[2] Kazaa. http://www.kazaa.com/.

[3] K. Aberer, P. Cudr-Mauroux, and M. Hauswirth. The chatty web: emergent semantics through gossiping. In *WWW*, pages 197–206, 2003.

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

[5] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *ICDT*, pages 207–224, 2003.

[6] F. Giunchiglia and I. Zaihrayeu Making peer databases interact - a vision for an architecture supporting data coordination. In *Cooperative Information Agents (CIA)*, pages 18–35, 2002.

[7] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *WebDB*, 2001.

[8] V. Kantere. A rule mechanism for p2p data management. Technical report, University of Toronto, 2003. CSRG-469.

[9] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *SIGMOD*, pages 325-336, 2003.

[10] A. Kementsietsidis, M. Arenas, and R. J. Miller. Managing data mappings in the Hyperion Project In *ICDE*, pages 732–734, 2003.

[11] M. Lenzerini. Data Integration: A Theoretical Perspective. In *PODS*, pages 233–246, 2002.

[12] B. Ludäscher, A. Gupta, and M. E. Martone. Model-based mediation with domain maps. In *ICDE*, pages 81–90, 2001.

[13] W. S. Ng, B. C. Ooi, and K. L. Tan. Bestpeer: A self-configurable peer-to-peer system. In *ICDE*, page 272, 2002.

[14] W. S. Ng, B. C. Ooi, K. L. Tan, and A. Y. Zhou. Peerdb: A p2p-based system for distributed data sharing. Technical report, University of Singapour, 2002.

[15] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *SIGCOMM*, pages 161–172, 2001.

[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[18] C. Turker and S. Conrad. Towards maintaining integrity in federated databases. In *Basque International Workshop on Information Technology (BIWIT)*, pages 93–100, 1997.