Data Management for Peer-to-Peer Computing: A Vision[1]

Philip A. Bernstein[2], Fausto Giunchiglia[3], Anastasios Kementsietsidis[4], John Mylopoulos[3], Luciano Serafini[5], and Ilya Zaihrayeu[2]

Abstract. We motivate special database problems introduced by peer-to-peer computing and propose the Local Relational Model (LRM) to solve some of them. As well, we summarize a formalization of LRM, present an architecture for a prototype implementation, and discuss open research questions.

1. Introduction

Peer-to-peer (hereafter P2P) computing consists of an open-ended network of distributed computational *peers*, where each peer can exchange data and services with a set of other peers, called *acquaintances*. Peers are fully autonomous in choosing their acquaintances. Moreover, we assume that there is no global control in the form of a global registry, global services, or global resource management, nor a global schema or data repository. Systems such as Napster and Gnutella popularized the P2P paradigm as a version of distributed computing lying between traditional distributed systems and the web. The former is rich in services but requires considerable overhead to launch and has a relatively static, controlled architecture. The latter is a dynamic, anyone-to-anyone architecture with little startup costs but limited services. By contrast, P2P offers an evolving architecture where peers come and go, choose whom they deal with, and enjoy some traditional distributed services with less startup cost.

We are interested in data management issues raised by this paradigm, where each peer may have data to share with other peers. For simplicity, we assume that each peer's database is relational. Since the data residing in different databases may have semantic inter-dependencies, we allow peers to specify *coordination formulas* that explain how the data in one peer must relate to data in an acquaintance. For example, the patient database of a family doctor and that of a pharmacy may want to coordinate their information about a particular patient, the prescriptions she has received, and the dates when these prescriptions were filled. Coordination may mean something as simple as propagating all updates to the Prescription and Medication relations, assumed to exist in both databases. In addition, we'd like a query expressed with respect to one database to be able to use relevant databases at acquaintances, acquaintances of those acquaintances, and so on. To accomplish this, we expect the P2P data management system to use coordination formulas for recursively decomposing the query into sub-queries that are evaluated with respect to the databases of acquaintances. Coordination formulas may also act as soft constraints or guide the propagation of updates. In addition, peers need an acquaintance initialization protocol where two peers exchange views of their respective databases and agree on levels of coordination between them. The level of coordination should be dynamic, in the sense that acquaintances may start with little coordination, strengthen it over time with more coordination formulas, and eventually abandon it when tasks and interests change.

In such a dynamic setting, we cannot assume the existence of a global schema for all databases in a P2P network, or even those of all acquainted databases. Moreover, peers should be able to establish and evolve acquaintances, preferably with little human intervention. Thus, we need to avoid protracted tasks by skilled database designers and DBAs required by traditional distributed and multi-database systems [9,10].

This paper introduces the Local Relational Model (LRM) as a data model specifically designed for P2P applications. LRM assumes that the set of all data in a P2P network consists of local (relational) databases, each with a set of acquaintances, which define the P2P network topology. For each acquaintance link, *domain relations* define translation rules between data items, and *coordination formulas* define semantic dependencies between the two databases. The main goals of the data model are to allow for inconsistent databases and to support semantic interoperability in the absence of a global schema [13].

[1] © 2002 Bernstein, Giunchiglia, Kementsietsidis, Mylopoulos, Serafini, Zaihrayeu.
[2] Microsoft Corporation, One Microsoft Way, Redmond WA, 98052-6399. philbe@microsoft.com
[3] University of Trento, 38050 Povo, Trento, Italy. {fausto,ilya}@dit.unitn.it.
[4] University of Toronto, Toronto, Canada, M5S 3H5. {jm, tasos}@cs.toronto.edu
[5] ITC-IRST, 38050 Povo, Trento, Italy. serafini@itc.it

The main objectives of this paper are to introduce the LRM through examples and to identify a set of open research questions on its design and implementation. Section 2 presents a motivating scenario. Section 3 sketches a formalization of LRM. Section 4 offers a preliminary architecture for an LRM-based system and relates it to past work, while conclusions appear in section 5.

2. A Motivating Scenario

Consider, again, the example of patient databases. Suppose that the Toronto General Hospital owns the TGHDB database with schema:

Patient(<u>TGH#</u>,OHIP#,Name,Sex,Age,FamilyDr,PatRecord) PatientInfo(<u>OHIP#</u>,<u>Record</u>)
Treatment(<u>TreatID</u>,TGH#,Date,TreatDesc,PhysID) Medication(<u>TGH#</u>,<u>Drug#</u>,Dose,<u>StartD</u>,EndD)
Admission(<u>AdmID</u>,OHIP#,AdmDate,ProblemDesc,PhysID,DisDate)

The database identifies patients by their hospital ID and keeps track of admissions, patient information obtained from external sources, and all treatments and medications administered by the hospital staff.

When a new patient is admitted, the hospital may want to establish immediately an acquaintance with her family doctor. Suppose the view exported by the family doctor DB (say, DavisDB) has schema:

Patient(<u>OHIP#</u>,FName,LName,Phone#,Sex,PatRecord) Visit(<u>OHIP#</u>,<u>Date</u>,Purpose,Outcome)
Prescription(<u>OHIP#</u>,<u>Med#</u>,Dose,Quantity,<u>Date</u>) Event(<u>OHIP#</u>,<u>Date</u>,Description)

Figuring out patient record correspondences (i.e., doing object identification) is achieved by using the patient's Ontario Health Insurance # (e.g., OHIP#=1234). Initially, this acquaintance has exactly one coordination formula which states that if there is no patient record at the hospital for this patient, then the patient's record from DavisDB is added to TGHDB in the PatientInfo relation, which can be expressed as:

$\forall$fn $\forall$ln $\forall$pn $\forall$sex $\forall$pr.(DavisDB : Patient(1234,fn,ln,pn,sex,pr) $\rightarrow$
TGHDB : $\exists$ tghid $\exists$n $\exists$a.(Patient(tghid,1234,n,sex,a,Davis,pr) and n = concat(fn,ln)))

When TGHDB imports data from DavisDB, the existentially quantified variables tghid, n and a must be instantiated with some concrete elements of the TGHDB database. This amounts to generating a new TGH# for tghid, inserting the Skolem constant <undef-age> for a (which will be further instantiated as the patient's age) and generating name n by concatenating her first name fn and last name ln contained in DavisDB. Later, if patient 1234 is treated at the hospital for some time, another coordination formula might be set up that updates the Event relation for every treatment or medication she receives:

$\forall$d $\forall$desc.(TGHDB : $\exists$tid $\exists$tghid $\exists$pid $\exists$n $\exists$sex $\exists$a $\exists$pr.(Treatment(tid,tghid,d,desc,pid) and
Patient(tghid,1234,n,sex,a,Davis,pr)) $\rightarrow$ DavisDB : Event(1234,d,desc)
$\forall$tghid $\forall$drug $\forall$dose $\forall$sd $\forall$ed.(
TGHDB: Medication(tghid,drug,dose,sd,ed) and $\exists$n $\exists$sex $\exists$a $\exists$pr.Patient(tghid,1234,n,sex,a,Davis,pr)
$\rightarrow$ DavisDB : $\forall$d.(sd $\leq$ d $\leq$ ed $\rightarrow$ $\exists$desc.(Event(1234,d,desc) and desc = concat(drug,dose,"at TGHDB"))))

This acquaintance is dropped once the patient's hospital treatment is over.

Along similar lines, the patient's pharmacy may want to coordinate with DavisDB. This acquaintance is initiated by DavisDB when the patient tells Dr. Davis which pharmacy she uses. Once established, the patient's name and phone are used for identification. The pharmacy database (say, AllenDB) has the schema:

Prescription(Prescr#,CustName,CustPhone#,DrugID,Dose,Repeats)
Sales(CustName,CustPhone#,DrugID,Dose,Date,Amount)

Here, we want AllenDB to remain updated with respect to prescriptions in DavisDB:

$\forall$fn $\forall$ln $\forall$pn $\forall$med $\forall$dose $\forall$qt.(
DavisDB: $\exists$ohip $\exists$date $\exists$sex $\exists$pr.(Prescription(ohip,med,dose,qt,date) and Patient(ohip,fn,ln,pn,sex,pr))
$\rightarrow$ AllenDB: $\exists$cn $\exists$amount.(Prescription(cn,pn,med,qt,dose,amount) and cn = concat(fn,ln)))

Of course, this acquaintance is dropped when the patient tells her doctor that she changed pharmacy.

Suppose the hospital has no information on its new patient with OHIP# 1234 and needs to find out if she is receiving any medication. Here, the hospital uses its acquaintance with an interest group of Toronto pharmacies, say TPhLtd. TPhLtd is a peer that has acquaintances with most Toronto pharmacists and has a coordination formula that allows it to access prescription information in those pharmacists' databases. For example, if we assume that TPhDB consists of a single relation

Prescription(Name,Phone#,DrugID,Dose,Repeats)

then the coordination formula between the two databases might be:

$\forall$fn $\forall$ln $\forall$pn $\forall$med $\forall$dose.(

DavisDB: ∃ohip ∃qt ∃date ∃sex ∃pr .(Prescription(ohip,med,dose,qt,date) and Patient(ohip,fn,ln,pn,sex,pr))
→ TPh: ∃name ∃rep.(Prescription(name,pn,med,dose,rep) and name = concat(fn,ln)))

Analogous formulas exist for every other pharmacy acquaintance of TPhLtd. Apart from serving as information brokers, interest groups also support mechanisms for generating coordination formulas from parameterized ones, given exported schema information for each pharmacy database.

On the basis of this formula, a query such as "All prescriptions for patient with name N and phone# P," evaluated with respect to TPhLtdDB, will be translated into queries that are evaluated with respect to databases such as AllenDB. The acquaintance between the hospital and TPhLtd is more persistent than those mentioned earlier. However, this one too may evolve over time, depending on what pharmacy information becomes available to TPhLtd.

Finally, suppose the patient in question takes a trip to Trento and suffers a skiing accident. Now the Trento Hospital database (TrentoHDB) needs information about the patient from DavisDB. This is a transient acquaintance that only involves making the patient's record available to TrentoHDB, and updating the Event relation in DavisDB.

3. A Formal Semantics for LRM

Traditionally, federated and multi-database systems have been treated as extensions of conventional databases. Unfortunately, formalizations of the relational model (such as [12]) don't apply to these extensions where there are multiple overlapping databases, which may be inconsistent and may use different vocabularies. We launch the search for implementation solutions that address the scenario described in the previous section with a formalization of LRM.

The model-theoretic semantics for LRM is defined in terms of relational spaces each of which models the state of the databases in a P2P system. These are mathematical structures generalizing the model-theoretic semantics for the Relational Model, as defined by Reiter in [12]. Coordination between databases in a relational space is expressed in terms of coordination formulas that describe dependencies between a set of databases. These formulas generalize many forms of inter-schema constraints defined in the literature, such as [1,2,5,8,15,17].

3.1 Relational spaces

A relational space is a finite set of relational databases. Database i is associated with a logical language L_i, which formalizes its schema. Abstractly, L_i is a first order language with a set of relational symbols corresponding to the relations of database i, no functions symbols, and a non-empty set of constants dom_i corresponding to the domain of database i. For instance, the language of DavisDB contains relational symbols such as Patient(x,y,z,w,v,t) and Visit(x,y,z,w), also the constant symbol 1234.

The content of database i is defined by a set of first order interpretations db_i of the language L_i on the domain dom_i. Each interpretation $m \in db_i$ interprets the constant symbol $d \in L_i$ as itself and the relational symbol $R(x_1,\dots, x_n)$ as a finite set of n-tuples of elements of dom_i, which are the tuples in the relation R. To emphasize that in LRM there is no global model, we call each db_i a local database.

In LRM, there is no notion of global consistency for a set of local databases. However, we do retain a notion of local consistency. Each local database can be in a (locally) consistent or inconsistent state, and consistent and inconsistent databases can coexist in a single relational space. For instance the local databases $db_a=\{m_1\}$, $db_b=\{m_2,m_3\}$, and $db_c=\emptyset$ are respectively complete, incomplete, and inconsistent. Generally, db_i is *complete* if $|db_i| = 1$, *incomplete* if $|db_i| > 1$ and *inconsistent* if $db_i = \emptyset$.

In a relational space, overlapping databases represent information about a common part of the world. This overlap has nothing to do with the fact that the same constant appears in both databases. For instance, the fact that the constant Apple appears in a database describing computers and another describing Italian agricultural products does not imply that these databases overlap. Rather, overlap is determined by the meaning of constants, i.e., when the entities denoted by constants in different databases are the same.

To represent the overlap of two local databases, one may use a global schema, with suitable mappings to/from each local database schema. As argued earlier, this is not feasible in a P2P setting. Instead, we adopt a localized solution to the overlap problem, defined in terms of pair-wise mappings from the elements of the domain of database i to elements of the domain of database j. Specifically, the overlap of databases i and j is represented by two relations, called *domain relations*, $r_{ij} \subseteq dom_i \times dom_j$ and $r_{ji} \subseteq dom_j \times dom_i$. The domain relation r_{ij} represents the ability of database j to import (and represent in its domain) the elements of the domain of database *i*. In many cases, domain relations are not symmetric, for instance when r_{ij} represents a currency exchange, a rounding function, or a sampling function. In a P2P setting, domain relations need only be defined for acquainted pairs of peers.

Definition (Relational space). A *relational space* is a pair $<db, r>$, where *db* is a set of local databases on *I* and *r* is a function that associates to each $i, j \in$ I, a domain relation from r_{ij} from *i* to *j*.

3.2 Coordination in relational spaces

Semantic inter-dependencies between local databases are expressed in a declarative language, independent of the languages supported by local databases. The formulas of this language describe properties of schemas as well as the contents of local databases in a relational space. This language is a generalization of interpretation constraints defined in [3].

Definition (Coordination formula). The set of *coordination formulas RF* on the family of relational languages $\{L_i\}_{i\in I}$ is defined as follows:

$$RF ::= i : \phi \mid RF \rightarrow RF \mid RF \wedge RF \mid RF \vee RF \mid \exists i : x.RF \,/\, \forall i : x.RF$$

where $i \in I$ and ϕ is a formula of L_i.

The basic building blocks of coordination formulas are expressions of the form $i : \phi$, which means "ϕ is true in database *i*". Connectives have the usual meaning, while quantifiers require further consideration. The formula $\forall i : x.A(x)$ should be read as "for all elements of the domain dom_i, *A* is true". Likewise, $\exists i : x.A(x)$, is read as "there is an element in the domain dom_i such that *A* is true". Notice that a variable *x* in the scope of a quantifier $\exists i : x$ *or* $\forall i : x$, can occur in a formula $j : \phi(x)$, allowing quantification across domains. Specifically, we allow that within the scope of a dom_i formula, one can quantify over another domain dom_j exploiting the domain relations r_{ij} and r_{ji}.

Example. The coordination formula $\forall i : x . (i{:}P(x) \rightarrow j : Q(x))$ is satisfied if whenever *P(d)* is true in database *i* and $<d, d'> \in r_{ij}$, then *Q(d')* is true in database *j*. Analogously the formula $\exists i : x . (j : P (x))$ is true if there is an element *d* in dom_i such that $<d, d'> \in r_{ij}$ and *P(d')* is true in database *j*. A complete formalization of truth for coordination formulas is described in [13].

Coordination formulas can be used in two different ways. First, they can be used to define constraints that must be satisfied by a relational space. For instance, the formula $\forall 1 : x . (1 : p(x) \vee 2 : q(x))$ states that any object in database 1 either is in table *p* or its corresponding object in database 2 is in table *q*. This is a useful constraint when we want to declare that certain data are available in a set of databases, without declaring exactly where. As far as we know, other proposals in the literature for expressing inter-database constraints can be uniformly represented in terms of coordination formulas.

Coordination formulas can also be used to express queries. In this case, a coordination formula is interpreted as a deductive rule that derives new information based on information already present in other databases. For instance, a coordination formula $\forall 1: x\ (1: \exists y.p(x, y) \rightarrow 2 : q(x))$ allows us to derive *q(b)* in database 2, if *p(a, c)* holds in database 1 for some *c*, and $<a,b> \in r_{12}$

Let *q* represent a query posed by a user to database *i*, and *A(x)* be the coordination formula body of the query. We have the following.

Definition (*i*-query). An *i-query* on a family of relational languages $\{L_i\}_{i\in I}$, is a coordination formula of the form $A(\mathbf{x}) \rightarrow i : q(\mathbf{x})$, where $A(\mathbf{x})$ is a coordination formula, *q* is a new *n*-ary predicate symbol of L_i and $\mathbf{x}$ contains *n* variables.

Definition (Global answer to an *i*-query). Let $<db, r>$ be a relational space on $\{L_i\}_{i\in I}$. The *global answer of an i-query* of the form $A(\mathbf{x}) \rightarrow i : q(\mathbf{x})$ in $<db, r>$ is the set:

$$\{\mathbf{d} \in (dom_i)^n \mid <db, r> \models \exists i : \mathbf{x}.(A(\mathbf{x}) \wedge i : \mathbf{x} = \mathbf{d})\}$$

An intuitive reading of the above formulas is as follows. The global answer to an *i*-query is computed by locally evaluating in db_i all atomic coordination formulas $i_k : \phi$ in *A,* and by recursively composing and mapping (via the domain relations) these results according to the connectives and quantifiers that comprise the coordination formula *A*. For instance, to evaluate the query

$$i : P(x) \vee j : Q(x) \wedge k : R(x,y) \rightarrow h : q(x,y)$$

we separately evaluate *P(x), Q(x)* and *R(x,y)* in databases i, j and k, respectively. We map these results via r_{ih}, r_{jh} and r_{kh} respectively obtaining three sets A_P, A_Q, and A_R in the domain dom_h. We then compose A_P, A_Q, and A_R using query connectives, obtaining $A_P \times A_Q \cap A_R$. This is the global answer to *q(x,y)*.

4. A Preliminary Architecture for LRM

Databases in a P2P system resemble heterogeneous distributed databases, often called *multi-database systems,* e.g., Multibase [14], TSIMMIS [4], Garlic [1], and Information Manifold [8]. In most systems of this sort, a user issues queries to a global schema, and the system (called a *mediator* in [16]) maps the queries to subqueries on the underlying data sources. Each data source has a wrapper layer that maps subqueries into its native query language. A database designer is responsible for creating the global schema and the mappings that define its relationship to the data sources, and for maintaining the schema and mappings as data sources enter and leave the system and as their schemas evolve. At this level of detail, the overall architecture has not changed since the earliest multi-database prototypes, over 20 years ago.

Like most multi-database systems, we assume that all peer nodes have identical architectures consisting of an LRM layer running on a local data server (e.g., a DBMS). As shown in Figure 1, the LRM Layer has four modules: *User Interface* (UI), *Query Manager* (QM*), Update Manager* (UM) and *Wrapper*. UI allows a user to define queries, receive results and messages from other nodes, and control other modules of the P2P Layer. QM and UM are responsible for query and update propagation. They manage domain relations, coordination formulas, coordination rules, acquaintances, and interest groups. Wrapper provides a translation layer between QM and UM, and LIS.

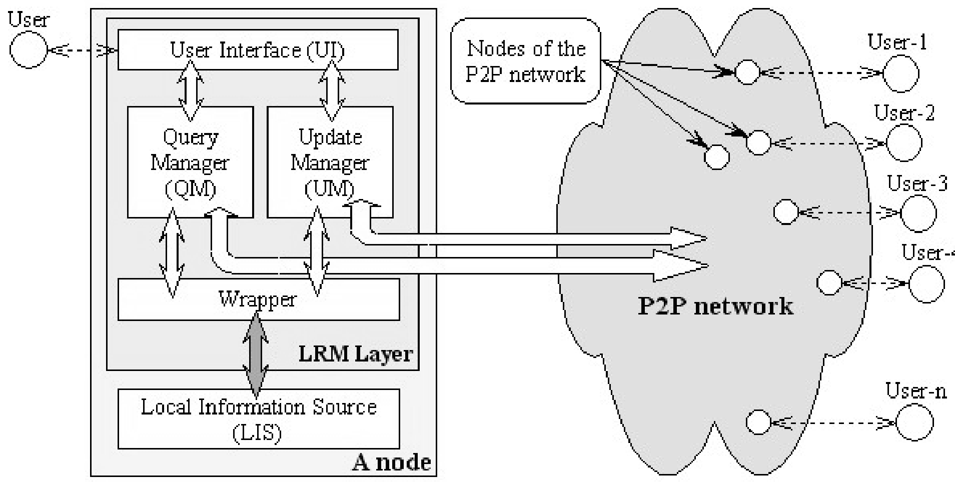

Figure 1: Architecture of an LRM node

Peers communicate through QM and UM using XML messages. Inter-module communication is also XML-based, shown as white arrows. The shaded arrow that connects Wrapper and LIS is different because the communication language is LIS-dependent (SQL, HTTP, …).

Strategies for query and update propagation are encoded in a set of *coordination rules*, which in most cases are expressed as ECA (Event Condition Action) rules. Coordination rules describe *when, how* and *where* a query or update must be propagated. A single formula may result in several rules. Some of these express parts of LIS as views of acquaintances, while others describe update propagations. For instance, consider the formula $\forall x.(1 : T(x) \rightarrow 2 : S(x))$. A reasonable coordination rule for query propagation from peer 2 might be: E = "receive a query Q", C= "S(x) occurs in the query Q", A = "submit the query T(x) to peer 1".

Although most of this architecture is merely a modernized version of conventional multi-databases, the P2P layer also needs to address new problems, each of which is an opportunity for future research:

- The P2P layer needs a protocol for establishing an acquaintance dynamically. This protocol can use a distributed system protocol for discovering a peer by name and establishing a session, after which each peer sends the schemas it chooses to export and with what privileges.

- After an acquaintance is established, formulas and rules are needed. The P2P layer could offer semi-automated support for generating coordination formulas, e.g., by using schema matching [11]. It might also automatically derive domain relations using data mining and other techniques; e.g., if rows of two relations have the same key, then values in matching non-key columns have the same meaning. Scrubbing rules for dirty data also need attention – a multi-database problem that's harder to cope with in a dynamic P2P setting.
- The P2P layer can use classical approaches to query processing, since coordination formulas are effectively views. However, it needs to incorporate a domain mapping logic, such as that offered by LRM. It also needs a policy on how far to propagate subqueries transitively through chains of P2P connections, which can be arbitrarily long and cyclic. In addition, standard query optimization approaches may need to be modified, e.g. with new protocols to exchange cost and utility information.
- For more effective inter-node coordination, nodes should be able to advertise their data content by giving a name and description (keywords or schema), presumably using a directory service. The P2P layer interprets this information to help users at a node create acquaintances and form Interest Groups with other nodes that have similar content.
- The problem of selecting materialized views and placing them at particular nodes becomes more difficult in a P2P scenario [7].

5. Conclusions

We have highlighted two main requirements introduced by P2P databases that distinguish them from other kinds of distributed databases. First, the mappings between databases are exclusively local, with no global schema. In support of this, we have proposed a data model for expressing such mappings between peers. Second, the set of peers is highly dynamic, requiring semi-automated solutions to problems that were formerly considered design-time, such as establishing configurations and mappings. These requirements lead to a variety of interesting, hard research problems that stretch today's multi-database solutions beyond their current limits.

References

1. M.J. Carey, L.M. Haas, P.M. Schwarz, Manish Arya, W.F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas II, J.H. Williams, E.L. Wimmers: Towards heterogeneous multimedia information systems: The Garlic approach. RIDE-DOM 1995: 124-131.
2. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *International J. of Intelligent and Cooperative Info. Sys., 2(4),* 375-398, 1993.
3. S. Ceri and J. Widom. Managing semantic heterogeneity with production rules and persistent queues. In Proceedings 19th VLDB (1993), 108-119.
4. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J.D. Ullman, J. Widom. The TSIMMIS Project: Integration of heterogeneous data sources. 16th Meeting of Information Processing Society of Japan, 1994, 7–18.
5. A. Gupta and J. Widom. Local verification of global integrity constraints in distributed databases. In Proc. ACM SIGMOD Conference, 49-58, 1993.
6. A. Y. Halevy. Answering queries using views: A survey. *VLDB J. 10:4* (2001), 270-294.
7. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, D. Suciu. What can databases do for peer-to-peer? WebDB Workshop on Databases and the Web, June 2001.
8. A.Y. Levy, Anand Rajaraman, J.J. Ordille. Querying heterogeneous information sources using source descriptions. In Proceedings VLDB 1996, pp. 251-262.
9. W. Litwin, L. Mark, N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys 22:3* (1990), 267-293.
10. M.T. Öszu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall, 1999.
11. E. Rahm, P.A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J. 10:4* (2001), pp. 334:350.
12. R. Reiter. Towards a logical reconstruction of Relational Database Theory. In *On Conceptual Modeling*, pp. 191-233. Springer-Verlag, 1984.
13. L. Serafini, F. Giunchiglia, J. Mylopoulos, P.A. Bernstein. The local relational model: Model and proof theory. Technical Report 0112-23, ITC-IRST, 2001.
14. Smith, J. M., P.A. Bernstein, U. Dayal, N. Goodman, T. Landers, K.W.T. Lin, E. Wong. MULTIBASE -- Integrating heterogeneous distributed database systems. Proceedings of 1981 National Computer Conference, AFIPS Press, 487-499.
15. J.D. Ullman. Information integration using logical views. In Proceedings of the 6th International Conference on Database Theory (ICDT), 1997.
16. G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer 25:3*, 38-49, 1992.
17. J. Widom, P.W.P.J. Grefen. Integrity constraint checking in federated databases. In Proceedings First IFCIS International Conference on Cooperative Information Systems, 38-47, 1996.