Defining Peer-to-Peer Data Integration using Both as View Rules

Peter McBrien[1] and Alexandra Poulovassilis[2]

[1] Dept. of Computing, Imperial College London, pjm@doc.ic.ac.uk
[2] School of Computer Science and Information Systems, Birkbeck College, Univ. of London, ap@dcs.bbk.ac.uk

Abstract. The loose and dynamic association between peers in a peer-to-peer integration has meant that, to date, implementations of peer-to-peer systems have been based on the exchange of files identified with a very limited set of attributes, and no schema is used to describe the data within those files. This paper extends an existing approach to data integration, called both-as-view, to be an efficient mechanism for defining peer-to-peer integration at the schema level, and demonstrates how the data integration can be used for the exchange of messages and queries between peers.

1 Introduction

The Internet has made available to almost all computer users the basic physical capability to exchange data. The challenge today is how to effectively harness this physical connectivity in order to effectively share data between users in a manner where their participation in data integration is not subject to centralised control, but instead is conducted in a **peer-to-peer (P2P)** fashion.

In [MP03] we described the **both-as-view (BAV)** approach to data integration, and compared it with the alternative approaches **global-as-view (GAV)** and **local-as-view (LAV)** [Len02]. In BAV, schemas are mapped to each other using a sequence of bidirectional schema transformations which we term a transformation **pathway**. From these pathways it is possible to extract a definition of the global schema as a view over the local schemas (*i.e.* GAV) and it is also possible to extract definitions of the local schemas as views over the global schema (*i.e.* LAV). The BAV approach has been implemented as part of the AutoMed data integration system being developed at Birkbeck and Imperial Colleges (see `http://www.doc.ic.ac.uk/automed`).

As we discussed in [MP02,MP03], one advantage of BAV over GAV and LAV is that it readily supports the evolution of *both* global and local schemas, including the addition or removal of local schemas. Such evolutions can be expressed as extensions to the existing pathways. New view definitions can then be regenerated from the new pathways as needed for query processing. This feature makes BAV very well suited to the needs of P2P data integration, where peers may join or leave the network at any time, or may change their set of local schemas, published schemas, or pathways between schemas. This paper describes how BAV can be used in this setting.

Previous work on data integration in P2P environments has used combinations of LAV and GAV rules between schemas and a combination of GAV and LAV query processing techniques [HIST03,HIMT03]. In our approach described here, we specify a BAV pathway between any pair of schemas. Due to the implicit presence of a GAV specification within such BAV pathways, and assuming no cycles in the inter-connection network between schemas, query answering in our approach is normally a simple matter of query unfolding using the GAV parts of the BAV pathways. However, it would also be possible to extract the LAV parts of the BAV pathways and use LAV query rewriting techniques.

A similar approach to ours is taken by [CDD$^+$03] in this proceedings, where the GLAV rules [FLM99] are used to specify the constructs of each schema in terms of the constructs of some set of other peer schemas, and hence it is possible to write rules that specify the mapping in both directions between schemas.

Other complementary work to ours has been carried out within the Edutella project [Nej03,LNWS03] which uses a superpeer based network topology to provide better scalability than pure peer-to-peer networks. RDF Schema is used as the common data model for heterogeneous information sources. Routing indexes at superpeers store information about the metadata available at the peers directly connected to them, and aid in the forwarding of query requests only to relevant peers. Correspondence assertions between global and local schema constructs are stored at the superpeers, and these correspondence assertions could be generated using the BAV techniques we describe here.

The need for a superpeer is avoided in the **local relational model (LRM)** [BGK$^+$02], where peers are directly related by a combination of a **domain relation** that specifies how the data types of the peers are related, together with **coordination formulae** that specify that if one predicate is true in one peer, then another predicate is true in another peer. The BAV approach has previously been shown to provide such a direct mapping between data sources [MP99a], and between different data modelling languages [MP99b].

The approach we present in this paper combines the advantages of Piazza and LRM, by having common virtual **superpeer schemas** — allowing peers to reuse the existing integration of other peers with the superpeer schema — but having no physical superpeer nodes that may act as a bottleneck in the system — in particular, we show how any peer can combine the different integrations of other peers with a superpeer schema in order to form direct pathways between peers for query and update processing.

We begin the paper with an overview of the BAV data integration approach. In Section 3 we describe how BAV can be extended to apply in a P2P setting, having originally been developed for a federated or mediated architecture. Then in Section 4 we describe one approach to P2P data integration using BAV, showing how update and query requests can be exchanged between peers via superpeer schemas. In Section 5 gives a more detailed comparison of our approach with closely related work. Finally we give a summary and our conclusions.

2 An Overview of BAV Data Integration

The basis of the BAV approach to data integration is a low-level **hypergraph-based data model (HDM)** and a set of primitive schema transformations for schemas expressed in this HDM [PM98,MP99a]. Facilities are provided for defining higher-level modelling languages and primitive schema transformations for them in terms of this lower-level HDM. For example, previous work has shown how relational, ER, UML, XML, RDF and semi-structured data models can be defined in terms of the HDM [MP99b,MP01,WP03,Kit03]. For each type of modelling construct of each modelling language (*e.g.* Relation, Attribute, Primary Key and Foreign Key in a relational model; Element, Attribute and Parent-Child relationship in XML) there will be a set of primitive schema transformations for adding such a construct to a schema, removing such a construct from a schema and, in the case of constructs with textual names, renaming such a construct.

Schemas are incrementally transformed by applying to them a sequence of such primitive schema transformations $t_1, \ldots, t_r$. Each primitive transformation t_i makes a 'delta' change to the schema by adding, deleting or renaming just one schema construct.

The general form of a primitive transformation that adds a construct c of type T to a schema s in order to generate new schema s' is $\mathsf{add}T(c, q_s)$, where q_s is a query over s specifying the extent of c in terms of the existing constructs of s. The logical semantics of this kind of transformation are

$$\forall \boldsymbol{x} \, . \, c(\boldsymbol{x}) \leftrightarrow q_s(\boldsymbol{x})$$

and for this reason we term add an **exact transformation**. In the AutoMed system, q_s is expressed in a functional **intermediate query language (IQL)** [JPZ03], and we shall use IQL for example queries in this paper.

When it is not possible to specify the exact extent of the new construct c being added in terms of the existing schema constructs, the primitive transformation $\mathsf{extend}T(c, q_s)$ must be used instead of add. The logical semantics of this kind of transformation are

$$\forall \boldsymbol{x} \, . \, c(\boldsymbol{x}) \leftarrow q_s(\boldsymbol{x})$$

and so we term extend a **sound transformation**. The query q_s may just be the constant Void, indicating that the extent of the new construct cannot be specified even partially. In this case the query can be omitted from the transformation, and a value of Void is implied.

In a similar manner, the exact transformation $\mathsf{delete}T(c, q_s)$ when applied to schema s' generates a new schema s with construct c of type T removed. The extent of c may be recovered using the query q_s on s, and

$$\forall \boldsymbol{x} \, . \, c(\boldsymbol{x}) \leftrightarrow q_s(\boldsymbol{x})$$

Note that this implies that from a primitive transformation $\mathsf{delete}T(c,q_s)$ used to transform $s' \to s$ we can automatically derive that $\mathsf{add}T(c,q_s)$ transforms $s \to s'$, and *vice versa*.

When it is not possible to specify the exact extent of the construct c being deleted from s' in terms of the remaining schema constructs, the sound transformation $\mathsf{contract}T(c,q_s)$ must be used instead of delete, where

$$\forall \boldsymbol{x} \, . \, c(\boldsymbol{x}) \leftarrow q_s(\boldsymbol{x})$$

Again, it is possible that q_s may just be Void, indicating that the extent of c cannot be specified even partially, in which case it can be omitted from the transformation. Note that from a primitive transformation $\mathsf{contract}T(c,q_s)$ used to transform $s' \to s$ we can automatically derive that $\mathsf{extend}T(c,q_s)$ transforms $s \to s'$, and *vice versa*.

Finally, the transformation $\mathsf{rename}T(c, c')$ causes a construct c of type T in a schema s to be renamed to c' in a new schema s', where in logical terms

$$\forall \boldsymbol{x} \, . \, c(\boldsymbol{x}) \leftrightarrow c'(\boldsymbol{x})$$

Note that this implies that from $\mathsf{rename}T(c, c')$ used to transform $s \to s'$ we can automatically derive that $\mathsf{rename}T(c', c)$ transforms $s' \to s$, and *vice versa*.

GAV defines a global schema as a set of views v over the local schemas, and LAV defines a local schema as a set of views v over a global schema. We relate v to a set of BAV schema constructs by a rule of the form $v(\boldsymbol{x}) = c_0(\boldsymbol{x_0}), \ldots, c_n(\boldsymbol{x_n})$ where $c_0(\boldsymbol{x_0}), \ldots, c_n(\boldsymbol{x_n})$ is a lossless decomposition of $v(\boldsymbol{x})$. For example, assuming the specification of the relational data model in terms of the HDM we gave in [MP03], if v is a relation $r(\boldsymbol{k}, a_1, \ldots, a_n)$ where $\boldsymbol{k}$ are its key attributes and $a_1, \ldots, a_n$ its non-key attributes, then c_0 would be a **Relation** construct $r(\boldsymbol{k})$ and $c_1, \ldots, c_n$ would be **Attribute** constructs $r\_a_1(\boldsymbol{k}, a_1), \ldots, r\_a_n(\boldsymbol{k}, a_n)$.

In [JTMP03] we discuss how LAV, GAV and GLAV views can be extracted from a BAV pathway $s_{local} \to s_{global}$. For a GAV view v defining a construct of s_{global} in terms of constructs of s_{local}, the algorithm uses the add and extend transformations that create $c_1, \ldots, c_n$. If all these transformations are exact then, in the terminology of [Len02], v is an **exact** view definition. If any one of $c_1, \ldots, c_n$ is defined using a sound transformation, then v is a **sound** view definition. For a LAV view v defining a construct of s_{local} in terms of constructs of s_{global}, the same algorithm is applied to the reverse pathway $s_{global} \to s_{local}$ (which, in BAV, is automatically derivable from $s_{local} \to s_{global}$). The only difference is that what in the GAV case were sound view definitions will in the LAV case be **complete** view definitions with respect to the global schema. Extraction of GLAV view definitions uses LAV view extraction and in addition also uses the bodies of add and extend transformations to generate GLAV rules (see [JTMP03]).

3 Developing BAV for P2P Data Integration

When building an integrated database, one must consider both the **logical integration** of the schemas and their logical extents, and the **operational inte-**

gration of the actual data, defining where data is to be materialised (*i.e.* permanently stored) and where data will be virtual (*i.e.* may be queried, but not permanently stored). We make the assumption that the logical extent of the local schemas equates to the materialised data within such schemas. In past work on data integration, there have been three basic approaches to the operational integration of data:

- **virtual global schema**: in the **federated database** [SL90] and **mediator** [Wie92] approaches, data is only materialised in local schemas. Any queries on the global schema are answered by rewriting the queries to execute on one or more local schemas, and the logical extent of the global schema equates to results of those queries. Hence the operational extent of the global schema is virtual, and equates to its logical extent.
- **materialised global schema**: in the **data warehouse** approach [JLVV02], data is materialised in both local and global schemas, and queries on each are answered directly from the data held within each schema. Hence the operational extent of the global schema is fully materialised. However the logical extent of the global schema is defined in the same way as for the federated database approach.
- **partial virtual global schema**: in the **workflow** approach [vdAvH02], the global schema is implied by some message format standard, and the logical extent of the global schema is the union of all the valid messages that all the local schemas may generate in the format. The operational extent of the global schema is simply those messages that are in transit at any one time.

In P2P networks, local schemas will be autonomous and membership of the network is likely to be highly dynamic. Thus, maintaining a materialised global schema is likely to be unachievable in practise, and even answering queries on the global schema is difficult due to the varying nature of the local schemas. Hence we regard the workflow model as the most promising for development as a basis for P2P integration, but we use **superpeer schemas** (see Section 4 below) to make explicit the notion of a global schema that is only implied in the workflow approach. We do not assume physical superpeer nodes; rather, we rely on peers publishing via a directory service such as UDDI [Bel02] their integration with standard superpeer schemas that might be owned by any peer.

3.1 BAV Sound Queries and Complete Queries

To use BAV for P2P data integration, it is now necessary that we are able write transformation rules that capture the looser relationship between local and global schemas. BAV sound transformation rules allow local schemas to provide a lower bound on what data is available in the global schema, but up to now BAV did not have a method of specifying that the logical extent of the global schema is an upper bound on the logical extent of the local schema. For the purposes of applying BAV to P2P data integration, we now extend it to support this facility. In particular, we extend the extend and contract transformations discussed above to take a second query as an argument:

The transformation $\mathsf{extend}T(c, q_l, q_u)$ adds a new construct c of type T to a schema s to form a schema s', where q_l determines from s what is the minimum extent of c in s' (and may be Void if no lower bound on the extent can be specified) and q_u determines from s what is the maximal extent of c in s' (and may be Any if no upper bound on the extent can be specified). We write the extent of c, $Ext(c)$, specified by such a transformation as an interval $[q_l, q_u]$. In logical terms it means that

$$\forall \boldsymbol{x} \,.\, c(\boldsymbol{x}) \leftarrow q_l(\boldsymbol{x}) \ \wedge\ \forall \boldsymbol{x} \,.\, c(\boldsymbol{x}) \rightarrow q_u(\boldsymbol{x})$$

Note that the semantics of add are such that $\mathsf{add}T(c, q_s) \equiv \mathsf{extend}T(c, q_s, q_s)$.

Similarly, the transformation $\mathsf{contract}T(c, q_l, q_u)$ removes a construct c of type T from a schema s' to form a new schema s, where q_l determines from s what is the minimum extent of c in s', and q_u determines from s what is the maximal extent of c in s'. As before, q_1 may be Void and q_u may be Any. In logical terms it means that

$$\forall \boldsymbol{x} \,.\, c(\boldsymbol{x}) \leftarrow q_l(\boldsymbol{x}) \ \wedge\ \forall \boldsymbol{x} \,.\, c(\boldsymbol{x}) \rightarrow q_u(\boldsymbol{x})$$

Note that the semantics of delete are such that $\mathsf{delete}T(c, q_s) \equiv \mathsf{contract}T(c, q_s, q_s)$.

We refer to the first query in an extend or $\mathsf{contract}$ transformation as defining the **sound** extent of the construct, and the second query as defining the **complete** extent; and the transformation as a whole is a **sound-complete** transformation. In the terminology of [Len02], when used to generate GAV views the first query generates sound views and the second query generates complete views. When used to generate LAV views the first query generates complete views and the second query sound views.

In general, a construct c in a global schema will be derived from a number of local schemas $l_1, \ldots, l_n$ with an extent $[q_{l_i}, q_{u_i}]$ derived from each local schema. Hence, there will be a number of lower bound queries over the local schemas, $q_{l_1}, \ldots, q_{l_n}$, and a number of upper bound queries, $q_{u_1}, \ldots, q_{u_n}$. The extent of c will have a lower bound which is the union of all the lower bounds, and an upper bound which is the intersection of all the upper bounds. Hence, writing the extent of c as an interval, we have $Ext(c) = [q_{l_1} \cup \ldots \cup q_{l_n}, q_{u_1} \cap \ldots \cap q_{u_n}]$.

4 P2P Data Integration via Superpeer Schemas

With the extentions proposed in the previous section, BAV could in principle be used to map directly between peer schemas. However, defining pairwise mappings between peer schemas does not scale as the number of schemas grows. Thus, we explore in this section a method of undertaking P2P BAV data integration via superpeer schemas. We also give a method for message exchange and query processing in this P2P integration framework.

We assume that in a P2P network peers are able to create schemas and make them available to the other peers (*i.e.* to publish them) — we term such publicly available schemas **superpeer** schemas. We will see below how BAV transformations with sound queries and complete queries within them give a natural method for defining superpeer schemas.

In addition to such public schemas, peers may also manage one or more local schemas, which may be either materialised or virtual. Each peer is able to create transformation pathways between its own local schemas and superpeer schemas made public by itself or other peers. Such pathways can be stored at the peer, but we assume that peers are able to publish the fact that they support a pathway to a superpeer schema (without necessarily publishing the actual pathway). A superpeer schema has a logical extent that is the union of all the peer schemas from which there exists a pathway to the superpeer schema.

Suppose now a peer P wishes to send a query or update request formulated with respect to one of its local schemas, l, to other peers that have access to data semantically related to l. P can find out to which superpeer schemas, s, there exists in its own metadata repository a pathway $l \to s$. P can also find out which other peers support pathways to s. Suppose P' is such a peer; then P can request from P' its set of pathways to s. Suppose $l' \to s$ is one of this set of pathways. P can then combine the reverse pathway $s \to l'$ with its own pathway $l \to s$ to obtain a pathway from l to l' (consisting of $l \to s$ followed by $s \to l'$). P can then use this pathway to translate a query or update request expressed on its own schema l to an equivalent query or update expressed on l' which can then be sent to P' for processing.

4.1 Method for Generating Pathways

To integrate a peer schema ps_i with a superpeer schema sps, the following steps can be followed in order to generate a pathway $ps_i \to sps$:

1. Decide which constructs of sps have no relationship with ps_i and use extend transformations with a [Void,Any] extent to add these constructs.
2. For each remaining construct c in sps, use extend transformations on ps_i in order to derive c with a sound query that specifies how constructs in ps_i can be used to derive global instances of c.
3. Decide which constructs of ps_i have no relationship with sps, and use contract transformations with a [Void,Any] extent to remove these constructs.
4. For each remaining construct c in ps_i, use contract transformations on ps_i in order to derive c with a complete query that specifies how constructs in sps can be used to derive local instances of c.

4.2 An Example

The schemas in Figure 1 are adapted from the example given in [BGK$^+$02]. Peer schema PS_1 is the schema for a hospital's patient database. Each patient is assigned a unique hospital identifier (hid) and a record is kept of their national insurance number (ni), name, sex, age, and the name of their gp (General Practitioner, or family doctor). Patients receive treatments. Each treatment has a unique identifier (tid), and a record is kept of the patient (via their hid), date, description and the Consultant who authorised the treatment.

Peer PS_2 is the schema for the database maintained by General Practitioner Dr Davies. He identifies his patients by their ni number and records their first name (fName), last name (lName), sex and address. His database also records in the event table all treatments and consultations for each of his patients as plain text descriptions within the field desc.

PS_1	hpatient(hid,ni,name,sex,age,gp)
	treatment(tid,*hid*,date,desc,consultant)
PS_2	patient(ni,fName,lName,sex,address)
	event(ni,date,desc)
SPS_1	allPatients(ni,name,sex,gp)
	allTreatments(ni,date,desc)

Fig. 1. Peer schemas

In Figure 1, a possible superpeer schema SPS_1 is given. Let us suppose that the hospital owning PS_1 wishes to exchange the information in its hpatient table, which we denote PS_1.hpatient, with other peers. Any patient record in PS_1 might be sent to another peer conforming to SPS_1. Conversely, a patient record from another peer conforming to SPS_1 might be imported into PS_1. The BAV pathway from PS_1 to SPS_1 is as follows:

$\mathsf{PS}_1 \rightarrow \mathsf{SPS}_1$

1. $\mathsf{contractTable}(\langle\langle \mathsf{treatment}, \mathsf{tid}, \mathsf{hid}, \mathsf{date}, \mathsf{desc}, \mathsf{consultant}\rangle\rangle, \mathsf{Void}, \mathsf{Any})$
2. $\mathsf{extendTable}(\langle\langle \mathsf{allTreatments}, \mathsf{ni}, \mathsf{date}, \mathsf{desc}\rangle\rangle, \mathsf{Void}, \mathsf{Any})$
3. $\mathsf{contractAtt}(\langle\langle \mathsf{hpatient}, \mathsf{age}\rangle\rangle, \mathsf{Void}, \mathsf{Any})$
4. $\mathsf{contractAtt}(\langle\langle \mathsf{hpatient}, \mathsf{hid}\rangle\rangle, \mathsf{Void}, \mathsf{Any})$
5. $\mathsf{extendTable}(\langle\langle \mathsf{allPatients}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle, \langle\langle \mathsf{hpatient}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle, \mathsf{Any})$
6. $\mathsf{contractTable}(\langle\langle \mathsf{hpatient}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle, \mathsf{Void}, \langle\langle \mathsf{allPatients}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle)$

The first two transformations above mean that no association is drawn between the PS_1.treatment and SPS_1.allTreatments tables. The next two transformations remove information about patients' ages and hospital identifier from the schema. Transformation 5 indicates that SPS_1.allPatients is a superset of PS_1.hpatient, while the final transformation 6 indicates that PS_1.hpatient is a subset of SPS_1.allPatients.

Also note that the contractTable and extendTable transformations are a shorthand for a sequence transformations on the Rel construct and its associated Att constructs. For example, contractTable is defined as follows, where $\boldsymbol{k}$ denotes the sequence of primary key attributes of relation r:

$$
\begin{aligned}
\mathsf{contractTable}(\langle\langle \mathsf{r}, \mathsf{a}_1, \ldots, \mathsf{a}_n\rangle\rangle, \mathsf{q}_1, \mathsf{q}_2) = {} & \mathsf{contractRel}(\langle\langle \mathsf{r}\rangle\rangle, \pi_{\boldsymbol{k}}\mathsf{q}_1, \pi_{\boldsymbol{k}}\mathsf{q}_2) \\
& \mathsf{contractAtt}(\langle\langle \mathsf{r}, \mathsf{a}_1\rangle\rangle, \pi_{\boldsymbol{k},\mathsf{a}_1}\mathsf{q}_1, \pi_{\boldsymbol{k},\mathsf{a}_1}\mathsf{q}_2) \\
& \ldots \\
& \mathsf{contractAtt}(\langle\langle \mathsf{r}, \mathsf{a}_n\rangle\rangle, \pi_{\boldsymbol{k},\mathsf{a}_n}\mathsf{q}_1, \pi_{\boldsymbol{k},\mathsf{a}_n}\mathsf{q}_2)
\end{aligned}
$$

The composite transformations extendTable, addTable and deleteTable are similarly defined in terms of a sequence of extend, add and delete primitive transformations on the Rel construct and its associated Att constructs.

As we saw in Section 3.1 above, each BAV primitive transformation has an automatically derivable reverse transformation, in that each add/extend transformation is reversed by a delete/contract transformation with the same arguments, while each rename transformation is reversed by another rename transformation with the two arguments swapped. Hence the BAV pathway $\mathsf{SPS_1} \rightarrow \mathsf{PS_1}$ is automatically derivable as the reverse of the above pathway $\mathsf{PS_1} \rightarrow \mathsf{SPS_1}$:

$\mathsf{SPS_1} \rightarrow \mathsf{PS_1}$

❻ extendTable(⟪hpatient, ni, name, sex, gp⟫, Void, ⟪allPatients, ni, name, sex, gp⟫)
❺ contractTable(⟪allPatients, ni, name, sex, gp⟫, ⟪hpatient, ni, name, sex, gp⟫, Any)
❹ extendAtt(⟪hpatient, hid⟫, Void, Any)
❸ extendAtt(⟪hpatient, age⟫, Void, Any)
❷ contractTable(⟪allTreatments, ni, date, desc⟫, Void, Any)
❶ extendTable(⟪treatment, tid, hid, date, desc, consultant⟫, Void, Any)

Similarly, let us suppose that Dr Davies maintaining $\mathsf{PS_2}$ wishes to exchange the information contained in his patient table with other peers. Any patient record in $\mathsf{PS_2}$ might be sent to another peer conforming to $\mathsf{SPS_1}$. Conversely, a patient record from another peer conforming to $\mathsf{SPS_1}$ might be imported into $\mathsf{PS_2}$. The BAV pathway from $\mathsf{PS_2}$ to $\mathsf{SPS_1}$ is as follows:

$\mathsf{PS_2} \rightarrow \mathsf{SPS_1}$

⑦ contractTable(⟪event, ni, date, desc⟫, Void, Any)
⑧ extendTable(⟪allTreatments, ni, date, desc⟫, Void, Any)
⑨ addAtt(⟪patient, gp⟫, [(x, 'Davies') | x ← ⟪patient⟫])
⑩ addAtt(⟪patient, name⟫, [(x, concat(y_1, ' ', y_2)) |
(x, y_1) ← ⟪patient, fName⟫; (x, y_2) ← ⟪patient, lName⟫])
⑪ deleteAtt(⟪patient, fName⟫, [(x, substring(z, 0, pos(z, ' '))) |
(x, z) ← ⟪patient, name⟫]
⑫ deleteAtt(⟪patient, lName⟫, [(x, substring(z, pos(z, ' ') + 1)) |
(x, z) ← ⟪patient, name⟫]
⑬ contractAtt(⟪patient, address⟫)
⑭ extendTable(⟪allPatients, ni, name, sex, gp⟫, ⟪patient, ni, name, sex, gp⟫, Any)
⑮ contractTable(⟪patient, ni, name, sex, gp⟫, Void, ⟪allPatients, ni, name, sex, gp⟫)

Again the pathway $\mathsf{SPS_1} \rightarrow \mathsf{PS_2}$ is automatically derivable as the reverse of this. The pathway $\mathsf{PS_1} \rightarrow \mathsf{PS_2}$ is just the composition of $\mathsf{PS_1} \rightarrow \mathsf{SPS_1}$ and $\mathsf{SPS_1} \rightarrow \mathsf{PS_2}$. Similarly, the pathway $\mathsf{PS_2} \rightarrow \mathsf{PS_1}$ is the composition $\mathsf{PS_2} \rightarrow \mathsf{SPS_1}$; $\mathsf{SPS_1} \rightarrow \mathsf{PS_1}$ or, equivalently, the reverse of $\mathsf{PS_1} \rightarrow \mathsf{PS_2}$

Such BAV pathways between peers, going via common superpeer schemas, can be used for translating messages between peers. Starting with a message expressed with respect to a schema of one peer, say $\mathsf{PS_1}$, the transformations in the pathway to a schema of another peer, say $\mathsf{PS_2}$, can be used to translate the message so that it is expressed in terms of $\mathsf{PS_2}$. Messages may contain update requests or query requests, both of which can be translated using the techniques we described in [MP99a] in the context of federated database architectures but which apply also in this context. The translation uses the queries present within each transformation to incrementally rewrite the message, and we discuss it in more detail now.

4.3 Sending Update Requests Over BAV Pathways

We assume that update requests to be sent from a data source S_1 to another data source S_2 will be of the form insert m, delete m, or update m for some expression m. The message translation process has two aspects, the first performing a **logical translation** of the message so that it *may* be applied to S_2, and the second performing an **operational interpretation** of the message to decide if it *should* be applied to S_2.

For the logical translation, we regard the data as having a lower bound d_l which is minimum set of values that should be inserted, deleted or updated, and an upper bound d_u which is the maximal set of values that should be inserted, deleted or updated. We write this range as the interval $[d_l, d_u]$. When some data d is to be sent from S_1 to S_2, we begin with $[d, d]$ appearing within the message expression m. The translation then proceeds as follows for each successive transformation t in the transformation pathway $S_1 \to S_2$:

1. if $t = rename(c, c')$, rename all occurrences of c in m by c';
2. if $t = add(c, q)$ and q is not Void, add c to m with range $[q,q]$.
3. if $t = extend(c, q_l, q_u)$ and q_u is not Void, add c to m with range $[q_l, q_u]$. The value of its associated extent will be within $[q_l, q_u]$ but this is decided by the operational aspect of the process.
4. if $t = contract(c, q_l, q_u)$ or $t = delete(c, q)$ and c appears in m then remove c from the m.

The operational aspect of the process determines how the values associated with constructs that appear in *extend* transformations are handled: if the logical value for a construct derived from a pathway is $[q_l, q_u]$, the **maximal** interpretation will give the construct the value q_u and the **minimal** interpretation will give the construct the value q_l.

For sending data from a peer schema ps_i to a peer schema ps_j via a superpeer schema sps, we define the **superpeer minmax interpretation** as taking the minimal interpretation for deriving the extent of constructs in sps from ps_i, and then the maximal interpretation for deriving the extent of constructs in ps_j from sps. Intuitively, the superpeer minmax interpretation ensures that only definite information from ps_i is transmitted to ps_j, and that all such information is transmitted. Example 1 illustrates this process applied to a message to convert an insert request expressed on PS_1 to an insert request expressed on PS_2, using the superpeer maxmin interpretation.

Example 1 Using pathways to update requests Suppose the update request insert hpatient(10000,'ZS341234P','Joe Bloggs','M',56,'Davies') is to be sent from PS_1 to PS_2. Transformations ③ and ④ convert the record to

hpatient('ZS341234P','Joe Bloggs','M','Davies')

making it union compatible with allPatients in schema SPS_1. Transformation ⑤ then states that this hpatient data is a lower bound of what should be inserted into allPatients in the superpeer schema SPS_1, and using the superpeer minmax interpretation, giving the range of values of the message in SPS_1 as:

[allPatients('ZS341234P','Joe Bloggs','M','Davies'),
allPatients('ZS341234P','Joe Bloggs','M','Davies')]

Transformation 15 then states upper bound of the the patient record in PS_2 is that of allPatients in SPS_1. Transformation 13 then inserts a Void value for the patient's address while transformations 12, 11 and 10 break up and replace the name attribute to make the record

patient('ZS341234P','Joe','Bloggs','M','Davies',Void)

Finally, transformation 9 removes the gp attribute; note that the query associated with this transformation allows one to reject records which do not have Void or 'Davies' as the gp attribute. This finally gives

insert patient('ZS341234P','Joe','Bloggs','M',Void)

as the message that will be sent to PS_2. If the address is nullable, then this insertion can be applied at PS_2 without further processing. However if address cannot be Null, the insertion will be rejected. A user must then be prompted to find the value of the address attribute before the insertion can be performed. □

4.4 Sending Query Requests Over BAV Pathways

We assume that query requests to be sent from a data source S_1 to another data source S_2 will be of the form query e where e is some IQL expression expressed on the constructs of S_1. The message translation process again has two aspects, the first performing a **logical translation** of the message so that is *may* be applied to S_2, and the second performing an **operational interpretation** of the message.

The logical translation of e proceeds as follows for each successive transformation t in the pathway $S_1 \rightarrow S_2$:

1. if $t = rename(c, c')$, rename all occurrences of c in e by c';
2. if $t = add(c, q)$ or $t = extend(c, q_l, q_u)$ ignore t as c cannot appear within the current query expression e;
3. if $t = del(c, q)$ replace all occurrences of c in e by the interval $[q, q]$;
4. if $t = contract(c, q_l, q_u)$ replace all occurrences of c in e by the interval $[q_l, q_u]$;

The operational aspect of the process determines how the interval queries associated with constructs that appear in *contract* transformations are handled: if an interval query is $[q_l, q_u]$, the **maximal** interpretation will select q_u and the **minimal** interpretation will select q_l.

For sending query requests from a peer schema ps_i to a peer schema ps_j via a superpeer schema sps, we define the **superpeer maxmin interpretation** as taking first the maximal interpretation for selecting queries over the intermediate schema sps, and then the minimal interpretation for selecting queries over the target schema ps_j. Intuitively, the superpeer maxmin interpretation ensures that only definite information from ps_j will be used to answer the query request and that all such information will be used.

Example 2 Using pathways to translate queries Suppose the following query is to be sent from PS_1 to PS_2:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{hpatient}\rangle\rangle; (\mathsf{x},\mathsf{s}) \leftarrow \langle\langle\mathsf{hpatient},\mathsf{sex}\rangle\rangle; \mathsf{s} = \text{'F'}; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{hpatient},\mathsf{ni}\rangle\rangle]$

Transformation 6 results in this query on the superpeer SPS_1, where the notation q_1 .. q_2 denotes a pair of set-valued queries respectively returning a lower and upper bound:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \mathsf{Void}..\langle\langle\mathsf{allPatients}\rangle\rangle; (\mathsf{x},\mathsf{s}) \leftarrow \mathsf{Void}..\langle\langle\mathsf{allPatients},\mathsf{sex}\rangle\rangle; \mathsf{s} = \text{'F'};$
$(\mathsf{x},\mathsf{n}) \leftarrow \mathsf{Void}..\langle\langle\mathsf{allPatients},\mathsf{ni}\rangle\rangle]$

Retaining only the upper bound queries, by the superpeer maxmin interpretation, gives:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{allPatients}\rangle\rangle; (\mathsf{x},\mathsf{s}) \leftarrow \langle\langle\mathsf{allPatients},\mathsf{sex}\rangle\rangle; \mathsf{s} = \text{'F'};$
$(\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{allPatients},\mathsf{ni}\rangle\rangle]$

Transformation 14 now results in this query on schema PS_2:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{patient}\rangle\rangle..\mathsf{Any}; (\mathsf{x},\mathsf{s}) \leftarrow \langle\langle\mathsf{patient},\mathsf{sex}\rangle\rangle..\mathsf{Any}; \mathsf{s} = \text{'F'};$
$(\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{patient},\mathsf{ni}\rangle\rangle..\mathsf{Any}]$

Retaining only the lower bound queries, by the superpeer maxmin interpretation, gives this final query on PS_2:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{patient}\rangle\rangle; (\mathsf{x},\mathsf{s}) \leftarrow \langle\langle\mathsf{patient},\mathsf{sex}\rangle\rangle; \mathsf{s} = \text{'F'}; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{patient},\mathsf{ni}\rangle\rangle]$

After this is evaluated at PS_2, the resulting set of records can be translated back to PS_1 using the translation scheme for update requests in Section 4.3. □

Example 3 Queries which cannot be answered As an example of P2P query processing involving unavailable information suppose the following query is to be sent from PS_1 to PS_2:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{hpatient}\rangle\rangle; (\mathsf{x},\mathsf{a}) \leftarrow \langle\langle\mathsf{hpatient},\mathsf{age}\rangle\rangle; \mathsf{a} > 65; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{hpatient},\mathsf{ni}\rangle\rangle]$

Transformations 3 and 6 result in this query on the intermediate schema SPS_1:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \mathsf{Void}..\langle\langle\mathsf{allPatients}\rangle\rangle; (\mathsf{x},\mathsf{a}) \leftarrow \mathsf{Void}..\mathsf{Any}; \mathsf{a} > 65;$
$(\mathsf{x},\mathsf{n}) \leftarrow \mathsf{Void}..\langle\langle\mathsf{allPatients},\mathsf{ni}\rangle\rangle]$

Applying the maxmin interpretation, this becomes:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{allPatients}\rangle\rangle; (\mathsf{x},\mathsf{a}) \leftarrow \mathsf{Any}; \mathsf{a} > 65; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{allPatients},\mathsf{ni}\rangle\rangle]$

Transformation 14 results in this query on schema PS_2:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{patient}\rangle\rangle..\mathsf{Any}; (\mathsf{x},\mathsf{a}) \leftarrow \mathsf{Any}; \mathsf{a} = \text{'F'}; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{patient},\mathsf{ni}\rangle\rangle..\mathsf{Any}]$

Applying the maxmin interpretation, this finally simplifies to:

$[(\mathsf{x},\mathsf{n}) \mid \mathsf{x} \leftarrow \langle\langle\mathsf{patient}\rangle\rangle; (\mathsf{x},\mathsf{a}) \leftarrow \mathsf{Any}; \mathsf{a} = \text{'F'}; (\mathsf{x},\mathsf{n}) \leftarrow \langle\langle\mathsf{patient},\mathsf{ni}\rangle\rangle]$

The presence of Any in the above query implies an absence of information and the query will evaluate to the empty set: as we would expect, no information can be extracted from PS_2 for the original query since it involves people's ages, about which there is no information in SPS_1. □

In general, a peer may wish to assemble results to a query from more than one peer that can provide such results, or from all such peers. This is easily supported in our framework:

Suppose a peer P wishes to send a query request formulated with respect to one of its local schemas, l, to other peers that have access to data semantically related to l. P can find out to which superpeer schemas, s, there exists in its own metadata repository a pathway $l \to s$. P can also find out which other peers support pathways to s and request from them the pathways from their local schemas to s. P can then construct a set of pathways from its local schema l going via a superpeer schema s to other peers' local schemas. A query request can be sent individually to each of these local schemas and the data returned merged at P in order to answer the original query expressed on l.

We note that multiple levels of superpeer schemas are possible with our approach, c.f. [BRM02], and the inter-connection network between schemas in our P2P network may be a tree of arbitrary depth as opposed to having just one level. At present we assume no cycles in this inter-connection network, and it is an area of further work to explore what their impact would be on query processing, c.f. [HIST03]. For translating update requests and data sent from one peer schema l to another l', we can use 'minmax' semantics with respect to the lowest common ancestor superpeer schema, *sps*, of l and l' while for translating queries from l to l' we can use 'maxmin' semantics with respect to *sps*.

We finally note that the BAV approach can also handle OO models. Once a BAV pathway has been specified between two schemas, any query that uses inheritance needs to be rewritten to make that inheritance explicit, before being translated using the techniques described above, and then making the inheritance implicit again. For example, suppose we had a subclass of hpatient called inpatient in PS_1 which inherits the attributes of hpatient. The query $\langle\langle \mathsf{inpatient}, \mathsf{name} \rangle\rangle$ on PS_1 would first be translated to $[(\mathsf{x}, \mathsf{y}) \mid (\mathsf{x}) \leftarrow \langle\langle \mathsf{inpatient} \rangle\rangle; (\mathsf{x}, \mathsf{y}) \leftarrow \langle\langle \mathsf{hpatient}, \mathsf{name} \rangle\rangle]$ before being translated using the techniques already described.

4.5 Changes to an Integration Network and Schema Evolution

The highly dynamic nature of P2P integration means that we must handle two types of changes in an efficient manner. First, a peer might wish to change what parts of one of its local schemas are taking part in an integration network. Second, the local schemas or superpeer schemas may evolve, and thus we need to reuse the old integration network to form a new one.

To handle alterations to what constructs of a local schema take part in an integration network we simply need to keep a record of what actions were taken when performing steps (1)–(4) in Section 4.1, and see if those actions need to be reviewed. In particular, if it is decided that a superpeer construct is now to be related to the peer schema, then the transformation for the construct covered by step (1) would be replaced by one or more covered by step (2). If a construct in the peer schema is then in part derivable from the superpeer schema, then the transformation for the construct covered by step (3) would be replaced by one or more covered by step (4).

In our running example, it might be decided to relate treatment information held in PS_1 with that in SPS_1. This will cause an update to pathway $\mathsf{PS}_1 \to \mathsf{SPS}_1$ to be made, replacing the subpathway where the treatment table was contracted (transformations 1 and 2) by a new subpathway that transforms PS_1.treatment into SPS_1.allTreatments as follows:

⑯ addAtt(⟨⟨treatment, ni⟩⟩, [(x, z) | (x, y) ← ⟨⟨treatment, hid⟩⟩;
(y, z) ← ⟨⟨hpatient, ni⟩⟩])
⑰ deleteAtt(⟨⟨treatment, hid⟩⟩, [(x, z) | (x, y) ← ⟨⟨treatment, ni⟩⟩;
(z, y) ← ⟨⟨hpatient, ni⟩⟩])
⑱ contractAtt(⟨⟨treatment, consultant⟩⟩, Void, Any)
⑲ contractAtt(⟨⟨treatment, tid⟩⟩, Void, Any)
⑳ extendTable(⟨⟨allTreatments, ni, date, desc⟩⟩, ⟨⟨treatment, ni, date, desc⟩⟩, Any)
㉑ contractTable(⟨⟨treatment, ni, date, desc⟩⟩, Void, ⟨⟨allTreatments, ni, date, desc⟩⟩)

Evolution of peer or superpeer schemas can be handled using our existing techniques for schema evolution in BAV [MP02]. The approach is that any evolution of a schema should be described as a BAV pathway from the original schema. It is then possible to reason about the composite pathway between other schemas, the old schema and the new schema. For example, let us suppose that a peer decides to publish a new version of SPS_1 called SPS'_1 such that allPatients now includes an attribute age. This may be expressed by the single-transformation pathway $\mathsf{SPS}_1 \rightarrow \mathsf{SPS}'_1$:

㉒ extendAtt(⟨⟨allPatients, age⟩⟩, Void, Any)

Now if a peer schema PS_3 integrates with SPS'_1 via pathway $\mathsf{PS}_3 \rightarrow \mathsf{SPS}'_1$ it is possible to exchange messages and data between PS_1 and PS_2 and the new schema PS_3 using the techniques already described.

4.6 Optimising Pathways

Once schema evolution has taken place over a period of time, it is likely that some degree of redundancy may exist in pathways. Also, given the use of the pathway for certain types of query processing, some elements of the pathway may be removed to optimise the processing of the pathway during query and message reformulation between schemas.

Using the techniques of [Ton03], the pathways may be analysed to determine if the integration of the peer schemas with superpeer schemas might be refined after schema evolution. For example, considering the pathway $\mathsf{PS}_1 \rightarrow \mathsf{SPS}'_1$ which now consists of ⑯ –㉑ ,③ –⑥ ,㉒ , it can be shown that ㉒ may be reordered to appear just before ⑤ if we change ⑤ to include the age attribute, and change ㉒ to operate on the hpatient table instead of the allPatients table. A suggestion can then be made to the peer managing PS_1 that the age attribute of hpatient might be included in the data integration since there is an apparently redundant contract and extend of the ⟨⟨hpatient, age⟩⟩ construct.

Once a particular operational interpretation has been decided upon for a long term exchange of messages between a set of peers, it also is possible to adapt techniques in [Ton03] to simplify pathways to give a direct pathway between the peer schemas. For example, if we have a large number of updates to send form PS_1 to PS_2 under minmax semantics, we may reorder transformations from the $\mathsf{PS}_1 \rightarrow \mathsf{PS}_2$ so that we have the subpathway:

⑤ extendTable(⟨⟨allPatients, ni, name, sex, gp⟩⟩, ⟨⟨hpatient, ni, name, sex, gp⟩⟩, Any)
⓯ extendTable(⟨⟨patient, ni, name, sex, gp⟩⟩, Void, ⟨⟨allPatients, ni, name, sex, gp⟩⟩)

Then taking the minimum extent of ⑤ (since it is originally before SPS_1) and the maximum extent of ⓯ (since it was originally after SPS_1) gives the simplified transformation:

㉓ $\mathsf{extendTable}(\langle\langle\mathsf{patient}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle, \mathsf{Void}, \langle\langle\mathsf{hpatient}, \mathsf{ni}, \mathsf{name}, \mathsf{sex}, \mathsf{gp}\rangle\rangle)$

Note that this simplified pathway is *only* suitable for message exchange between the peers using minmax semantics; since it does not form a semantically correct relationship between the peers. This is because it is incorrectly stating the $\mathsf{PS}_2 : \langle\langle\mathsf{patient}\rangle\rangle$ is a subset of $\mathsf{PS}_1 : \langle\langle\mathsf{hpatient}\rangle\rangle$.

5 Comparison with Related Approaches

In the context of data integration, GLAV is regarded as subsuming the expressive capabilities of LAV and GAV. Thus we focus our comparison here on examining how BAV compares with GLAV. In the original GLAV work [FLM99], rules were permitted to have a conjunction of source schema relations in their head. In [MH03] this was extended to allow any query expressed in the source schema query language to appear in the head of a GLAV rule. In [CDD$^+$03], the distinction between source and global schemas is removed, and any number of GLAV rules may be specified between schemas, hence in part moving towards the BAV approach of specifying rules for each schema being integrated. However, [CDD$^+$03] does not differentiate between sound, complete and exact rules, as GLAV rules are always sound.

In [CDD$^+$03], integrating schemas PS_1, PS_2, $\ldots \mathsf{PS}_\mathsf{n}$, there might be GLAV rules $\mathsf{PS}_1 \leftarrow \mathsf{PS}_2, \mathsf{PS}_3, \ldots, \mathsf{PS}_\mathsf{n}$, $\mathsf{PS}_2 \leftarrow \mathsf{PS}_1, \mathsf{PS}_3, \ldots, \mathsf{PS}_\mathsf{n}$, *etc.* By contrast, BAV can define a single network of transformations that integrates the peer schemas. This may be done by direct association between peers or, as described in this paper, via one or more superpeer schemas. One advantage of the BAV approach is that it separates the logical specification of a mapping between schemas from the procedural aspects of performing query or update processing over the mapping. Another advantage of BAV is that it allows common concepts shared between schemas to be explicitly stated. For example, if hospitals insisted that all patients register with a GP (*i.e.* $\mathsf{hospital\_patient} \subseteq \mathsf{gp\_patient}$), then in BAV we could integrate hospital peer schemas to specify a superpeer schema containing $\langle\langle\mathsf{hospital\_patient}\rangle\rangle$, and integrate GP peer schemas to specify a superpeer schema containing $\langle\langle\mathsf{gp\_patient}\rangle\rangle$, and then relate the two superpeer schemas by:

$\mathsf{extendTable}(\langle\langle\mathsf{gp\_patient}, \ldots\rangle\rangle, \langle\langle\mathsf{hospital\_patient}, \ldots\rangle\rangle, \mathsf{Any})$
$\mathsf{contactTable}(\langle\langle\mathsf{hospital\_patient}, \ldots\rangle\rangle, \mathsf{Void}, \langle\langle\mathsf{gp\_patient}\rangle\rangle)$

By contrast in GLAV, each hospital would need to state a rule saying that its patients were a subset of gp_patient, in effect repeating the definition of gp_patient at every peer, and omitting the concept of hospital_patient.

6 Summary and Conclusions

We have defined in this paper an extension to the BAV data integration approach to allow it to specify both sound queries and complete queries in transformations, and have demonstrated how this extension may be used for P2P data integration. The sound queries are used where a minimum answer is required, and serve as the basis for moving data from peer schemas to superpeer schemas, and for answering queries on a superpeer schema from a peer schema. The complete queries are used where a maximal answer is required, and serve as the basis for moving data from superpeer schemas to peer schemas, and for answering queries on a peer schema from a superpeer schema. Hence we use 'minmax' semantics in moving data from a peer schema via a superpeer schema to another peer schema, and 'maxmin' semantics in moving queries over a similar path.

We have shown how the pathways are easy to update to reflect changes in the P2P data integration, and our previous work [MP02] has demonstrated how we handle schema evolution. Redundancy in the pathways between schemas may be removed using the techniques described in [Ton03].

The BAV approach has been adopted within the AutoMed data integration system (`http://www.doc.ic.ac.uk/automed`). All source schemas, intermediate schemas and global schemas, and the pathways between them are stored in AutoMed's metadata repository [BMT02], and a suite of tools have been developed for creating and editing a schema integration network, processing queries (using GAV query processing) [JPZ03], and analysing the contents of schemas to suggest integration rules. Future work will extend this suite of tools to support the new P2P extensions reported in this paper. We also plan to apply this technology in the SeLeNe project (`http://www.dcs.bbk.ac.uk/selene`) which is investigating techniques for semantic integration of RDF/S descriptions of learning objects stored in P2P networks.

References

[Bel02] T. Bellwood *et al.* UDDI version 3.0. Technical report, UDDI.ORG, July 2002.

[BGK+02] P.A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: a vision. In *Proceedings of WebDB02*, pages 89–94, 2002.

[BMT02] M. Boyd, P.J. McBrien, and N. Tong. The automed schema integration repository. In *Proceedings of BNCOD02*, volume 2405 of *LNCS*, pages 42–45. Springer-Verlag, 2002.

[BRM02] Z. Bellahsene, M. Roantree, and L. Mignet. A functional architecture for large scale integration. Technical report, Univ. of Montpellier, 2002.

[CDD+03] D. Calvanese, E. Damagio, G. De Giacomo, M. Lenzerini, and R. Rosati. Semantic data integration in P2P systems. In *Proceedings of DBISP2P*, Berlin, Germany, 2003.

[FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th National Conference on Artificial Intelligence*, pages 67–73. AAAI, 1999.

[HIMT03] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *WWW 2003*, 2003.

[HIST03] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of ICDE03*. IEEE, 2003.

[JLVV02] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis. *Fundamentals of Data Warehouses.* Springer-Verlag, 2nd edition edition, 2002.

[JPZ03] E. Jasper, A. Poulovassilis, and L. Zamboulis. Processing IQL Queries and Migrating Data in the AutoMed toolkit. Technical report, AutoMed TR Number 20, 2003. Available from`http://www.doc.ic.ac.uk/automed/`.

[JTMP03] E. Jasper, N. Tong, P.J. McBrien, and A. Poulovassilis. View generation and optimisation in the automed data integration framework. Technical report, AutoMed TR Number 16, Version 3, 2003. Available from `http://www.doc.ic.ac.uk/automed/`.

[Kit03] S. Kittivoravitkul. Transformation-based approach for integrating semi-structured data. Technical report, AutoMed Project, http://www.doc.ic.ac.uk/automed/, 2003.

[Len02] M. Lenzerini. Data integration: A theoretical perspective. In *Proceedings of PODS 2002*, pages 233–246. ACM, 2002.

[LNWS03] A. Loser, W. Nejdl, M. Wolpers, and W. Siberski. Information integration in schema-based peer-to-peer networks. In *Proceedings of CAiSE 2003*, LNCS. Springer-Verlag, 2003.

[MH03] J. Madhavan and A.Y. Halevy. Composing mappings among data sources. In *Proceedings of 29th Conference on VLDB*, pages 572–583, 2003.

[MP99a] P.J. McBrien and A. Poulovassilis. Automatic migration and wrapping of database applications — a schema transformation approach. In *Proceedings of ER99*, volume 1728 of *LNCS*, pages 96–113. Springer-Verlag, 1999.

[MP99b] P.J. McBrien and A. Poulovassilis. A uniform approach to inter-model transformations. In *Advanced Information Systems Engineering, 11th International Conference CAiSE'99*, volume 1626 of *LNCS*, pages 333–348. Springer-Verlag, 1999.

[MP01] P.J. McBrien and A. Poulovassilis. A semantic approach to integrating XML and structured data sources. In *Proceedings of 13th CAiSE*, volume 2068 of *LNCS*, pages 330–345. Springer-Verlag, 2001.

[MP02] P.J. McBrien and A. Poulovassilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *Advanced Information Systems Engineering, 14th International Conference CAiSE2002*, volume 2348 of *LNCS*, pages 484–499. Springer-Verlag, 2002.

[MP03] P.J. McBrien and A. Poulovassilis. Data integration by bi-directional schema transformation rules. In *Proceedings of ICDE03*. IEEE, 2003.

[Nej03] W. Nejdl *et al.* Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *WWW 2003*, 2003.

[PM98] A. Poulovassilis and P.J. McBrien. A general formal framework for schema transformation. *Data and Knowledge Engineering*, 28(1):47–71, 1998.

[SL90] A. Sheth and J. Larson. Federated database systems. *ACM Computing Surveys*, 22(3):183–236, 1990.

[Ton03] N. Tong. Database schema transformation optimisation techniques for the automed system. In *Proceedings of BNCOD*, volume 2712 of *LNCS*, pages 157–171. Springer-Verlag, 2003.

[vdAvH02] W. van der Aalst and K. van Hee. *Workflow Management: Models, Methods, and Systems.* MIT Press, 2002.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

[WP03] D. Williams and A. Poulovassilis. Representing RDF and RDF Schema in the HDM. Technical report, AutoMed Project, http://www.doc.ic.ac.uk/automed/, 2003.