Inter model data integration in a P2P environment

Duc Minh Le and Andrew Smith

Imperial College London

Abstract. The wide range of data sources available today mean that the integration of heterogeneous data sources is now a common and important problem. It is even more challenging in a P2P environment, peers often do not know in advance which schemas of other peers will suit their information needs and there is potentially a greater diversity of data modelling languages in use. In this paper, we propose a new approach to P2P inter model data integration which supports data model diversity whilst allowing peers to have the flexibility of choosing how to integrate their schemas. We will show how this approach can be used to integrate multiple heterogeneous data sources (CSV text files, XML and SQL) and how the metadata of the (partially) integrated schemas can be distributed and made available for discovery by other peers. We also briefly describe a system being developed by our group that has the discussed features.

1 Introduction

P2P inter model data integration is the process whereby data stored in autonomous heterogeneous data sources under different data models is made accessible to everyone on a P2P network.

There are three main aspects to this. The first is how to express represent schemas from different data models in a **common data model (CDM)** [1] in which all the constructs of the various data models in the integration can be represented. In this paper we make use of the HDM, a graph based model which can represent a wide range of higher level data constructs making it a good choice as a CDM in an inter model integration. It has been used to model XML, UML, SQL [2] and RDF [3] amongst others.

The second aspect of the problem is how to enable peers to find the schemas created in the previous step. The difficulty in distributed schema metadata management lies in the ability to effectively handle a potential explosion in the number of schemas on the network as the network size increases and peers integrate their schemas in add-hoc fashion. Further, as the number of schemas on the network increase, searching for relevant schemas desires more rigorous selection criteria.

Finally, once we have found the relevant schemas we need to integrate them to provide a unified view of the relevant data sources to users of the P2P network. We use the both-as-view (BAV) [4] data integration technique which provides good support for schema evolution [5], an important advantage in a dynamic P2P environment.

Existing P2P data sharing systems such as PeerDB [6] address this problem by defining some form of metadata using the constructs of the local data source model. Further, these systems assume the use of a single data model across the network and do not tackle the issue of integrating different data models as with our approach. On the other hand, existing P2P data integration systems either do not address the schema metadata management (e.g. Piazza [7] and CoDB(z) [8]) or concentrate this functionality as part of query processing in certain fast nodes (e.g. Edutella [9]) which can lead to the overloading of those nodes.

In this paper we propose a method of effectively performing inter model data integration in a P2P environment by using a generic CDM capable of representing a wide range of data models and a schema metadata management method based on this generic model. To the best of our knowledge this has not been attempted before. In particular we make the following contributions:

- we show how the combination of the HDM and the BAV data integration method are particularly suited to integrating schemas represented in different data modelling languages in a P2P environment
- we define a general framework, independent of the data models of the peer data sources, for representing schema metadata and for managing these metadata on the P2P network. We use a DHT-based repository as this scales well with the network, and focus on load fairness amongst peers
- we formally define the schema search problem and propose an algorithm for finding the relevant schemas on the P2P network based on schema object scheme matching and satisfiability check of predicate-based queries against the schema metadata

The rest of the paper is structured as follows: Section 2 presents an example scenario and briefly describes our CDM. Section 3 describes our approach to schema metadata management. Section 4 describes our data integration technique. Section 5 describes the related work and Section 6 gives conclusions and directions for future work.

2 Representing different data models in a CDM

Consider a group of colleagues at different research institutions connected via a P2P network collaborating on a research project into the correlation between blood sugar level (BSL) and body mass index (BMI). They wish to share their data but each researcher stores their data in a different way. Researcher A uses an SQL database, researcher B an XML file and C stores her results in a spreadsheet that she exports as a CSV text file. Figures 1, 2 and 3 show fragments of the data sources provided by the three researchers.

In this case there is no obvious candidate model to use as a CDM. One option is to use one of the local schema models and transform the other schemas into that model. This is often the approach taken when there are only 2 different data models involved [10]. A more flexible approach, when the participating data sources use more than 2 different data modelling languages, is to use a generic model that is capable of easily expressing all the constructs of the different local schemas [11, 2, 12]. This is the approach we adopt, using the **Hypergraph Data Model (HDM)** [13] as a CDM.

nhs_num	blood		weight	
id	idblood	bsl	idweight	bmi
100	100	3	100	17
101	101	3	102	22
102	103	5	103	17
103	104	4		
104				

blood.idblood ⟵ nhs_num.id
weight.idweight ⟵ nhs_num.id

Fig. 1. SQL tables

```
<results>
    <result id = "102">
        <bsl>4</bsl>
        <bmi>17</bmi>
    </result>
    <result id = "103">
        <bsl>5</bsl>
        <bmi>19</bmi>
    </result>
    <result id = "104">
        <bsl>5</bsl>
        <bmi>18</bmi>
    </result>
</results>
```

Fig. 2. XML

id	bsl	bmi
105	4	19
106	3	17
108	3	15
109	5	22
110	4	14

Fig. 3. Spreadsheet

The HDM is a graph based generic data model that makes use of 3 simple constructs: nodes, edges and constraints. It has been used to represent a wide variety of data models making it particularly suited as a CDM in inter model data integration. The fact that it is graph based offers the added advantages of easy visual comparison of data structures and the ability to represent all schemas in an irreducible form [14]. RDF is another graph based modelling language which has been used as a CDM in P2P applications [12] but HDM's simple semantics and compact syntax make it more flexible than RDF and thus more suitable to an inter model data integration scenario.

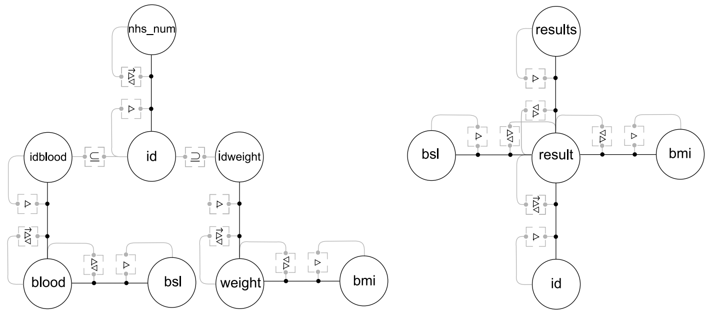
Figure 4 shows the data sources from the example represented in the HDM. In Figure 4(a) the database tables nhs_num, blood and weight are represented by nodes and their columns by nodes linked to the respective 'table' nodes by an edge. The **extent** of a node or edge is the set of values from the data source that the node or edge represents. For example the **extent** of the 'column' node $\langle\langle \mathsf{bmi} \rangle\rangle$ [1] is $\{17, 22, 17\}$ and of $\langle\langle \_, \mathsf{weight}, \mathsf{bmi} \rangle\rangle$, the edge linking $\langle\langle \mathsf{weight} \rangle\rangle$ to $\langle\langle \mathsf{bmi} \rangle\rangle$ is $\{\langle 100, 17\rangle, \langle 102, 22\rangle, \langle 103, 17\rangle\}$

The HDM constraint constructs are shown in the grey boxes in the diagram. HDM supports six distinct primitive constraint operators: union, inclusion, exclusion, mandatory, unique and reflexive that can be used to model the constraint constructs of higher level languages [2]. For example we use the unique ($\lhd$), mandatory ($\rhd$) and reflexive ($\xrightarrow{id}$) constraints to model the primary key constraint from the nhs_num table. The fact that a primary key column cannot be null is represented by the mandatory constraint from nhs_num to the edge. This means that any value in the extent of nhs_num must appear in the edge, it's uniqueness is represented by the `unique` constraint. The `reflexive` constraint ensures that any tuple in the extent of the edge is an identity tuple.

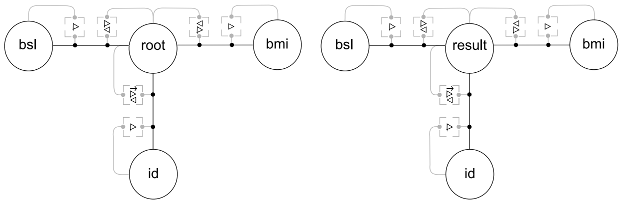
The structure and semantics of the other two data sources are represented in the HDM in a similar way. In Figure 4(b) the XML elements and attributes become nodes in the HDM and the hierarchical structure is represented by edges linking parent elements to their children. For example the parent element results is linked to its child result by an edge. The associations between result and its attributes are also shown as edges. In Figure 4(c) the first column in the CSV file is represented by two nodes one acting as

[1] We use double angle brackets $\langle\langle\rangle\rangle$ to denote the schema objects in a schema

the root node of the and the other acting as a key node linked to the root by an edge. The other columns in the file become nodes also linked to the root node by edges.



(a) HDM representation of SQL datasource (b) HDM representation of XML datasource



(c) HDM representation of spreadsheet datasource (d) The public schema

Fig. 4. HDM representation of the data sources

In a P2P setting peers initially do not know which schemas on the network would be relevant to their needs. Representing all the data models in a CDM as we have done makes it possible for the schema search to work across different data models rather than a single model as in PeerDB. In the next section, we present a schema metadata management method which allows peers to efficiently find the public schemas and transformation pathways they need by making effective use of schema metadata.

3 Schema Metadata Management

To store and maintain schema metadata, peers in a data integration domain form a distributed schema metadata repository (SMR). We will first give the formal definitions of schema metadata based on our HDM data model before discussing the semantics of an SMR and its key functions.

Summary query A summary query q of a schema object so of a schema S is an extended query extent of the so's BAV transformation which considers the values range $\mathsf{r} = [\mathsf{min}(\mathsf{Vals}(\mathsf{so})), \mathsf{max}(\mathsf{Vals}(\mathsf{so}))]$. Vals(so) represents the set of actual data values of the schema object so.

Schema object metadata A schema object metadata so^m of a schema object so of a public schema S is a tuple $\langle peer, schema, scheme, q\rangle$, where peer is the URL of the peer implementing schema S, schema is the name of the schema S, scheme is the string representation of the HDM-construct instance that defines so, and q is the summary query of so. In HDM, a scheme is a list of elements $\{\mathsf{e}_1, \mathsf{e}_2, \ldots, \mathsf{e}_\mathsf{m}\}$ where e_i are names of the HDM nodes that are referred to by so. We call two schema objects $\mathsf{so}_\mathsf{a}^\mathsf{m}$ and $\mathsf{so}_\mathsf{b}^\mathsf{m}$ k-factor *matchable* w.r.t searching if their schemes share a continuous sequence of k scheme elements starting from the first scheme position, i.e. $\exists 1 \leq \mathsf{k} \leq \mathsf{m} : \mathsf{so}_\mathsf{a}^\mathsf{m}.\mathsf{scheme} \cap \mathsf{so}_\mathsf{b}^\mathsf{m}.\mathsf{scheme} = \{\mathsf{e}_1, ..., \mathsf{e}_\mathsf{k}\}$. We contend that for schema metadata management purposes schema object schemes do not need to be exact but must be matchable. We denote $\mathcal{SO}$ and $\mathcal{SO}^\star$ be the set of all schema object metadata of a schema and on the network respectively.

Schema metadata A schema metadata S^m of a schema S is the set of all schema object metadata defined for S. The metadata of a schema is defined either when the schema is defined from the data source or when it is integrated from other schemas.

Example As shown in Figure 4(d), the public schema has four nodes and three edges. Taking the XML public schema S_xml with identifier by s_xml (lower case s) as an example, it is implemented by peer P_a and its schema metadata contains the schema object metadata in Table 1. We use the notation P_a (capitalised P) to refer to the URL (e.g. IP address) of a peer a. Note from this table that although the extents of the three schema objects $\langle\langle\mathsf{result}\rangle\rangle$, $\langle\langle\mathsf{id}\rangle\rangle$, and $\langle\langle\_, \mathsf{result}, \mathsf{id}\rangle\rangle$ are same we still define for each of them a separate schema object metadata. There are important two reasons for this. First, because these metadata entries are registered to different SMR peers it will be easier to search for the associated schema S_xml. Second, although the extents of these three objects are initially the same, when schema S_xml is integrated with other schemas by different peer applications these extents would be updated in different ways. This would then lead to different summary queries be generated by these applications for the schema object metadata.

Schema Objects	**Schema object metadata**			
	Peer	Schema	Schemes	Summary queries
$\langle\langle\mathsf{result}\rangle\rangle$	P_a	s_xml	$\{\mathsf{result}\}$	$[\{x\} \mid \{x\} \leftarrow \langle\langle\mathsf{result}\rangle\rangle; x \geq 100; x \leq 107]$
$\langle\langle\mathsf{bsl}\rangle\rangle$	P_a	s_xml	$\{\mathsf{bsl}\}$	$[\{x\} \mid \{x\} \leftarrow \langle\langle\mathsf{bsl}\rangle\rangle; x \geq 3; x \leq 5]$
$\langle\langle\mathsf{bmi}\rangle\rangle$	P_a	s_xml	$\{\mathsf{bmi}\}$	$[\{x\} \mid \{x\} \leftarrow \langle\langle\mathsf{bmi}\rangle\rangle; x \geq 17; x \leq 19]$
$\langle\langle\mathsf{id}\rangle\rangle$	P_a	s_xml	$\{\mathsf{id}\}$	$[\{x\} \mid \{x\} \leftarrow \langle\langle\mathsf{id}\rangle\rangle; x \geq 100; x \leq 107]$
$\langle\langle\_, \mathsf{result}, \mathsf{id}\rangle\rangle$	P_a	s_xml	$\{\mathsf{result}, \mathsf{id}\}$	$[\{x\} \mid \{y, x\} \leftarrow \langle\langle\_, \mathsf{result}, \mathsf{id}\rangle\rangle; x \geq 100; x \leq 107]$
$\langle\langle\_, \mathsf{result}, \mathsf{bsl}\rangle\rangle$	P_a	s_xml	$\{\mathsf{result}, \mathsf{bsl}\}$	$[\{x\} \mid \{y, x\} \leftarrow \langle\langle\_, \mathsf{result}, \mathsf{bsl}\rangle\rangle; x \geq 3; x \leq 5]$
$\langle\langle\_, \mathsf{result}, \mathsf{bmi}\rangle\rangle$	P_a	s_xml	$\{\mathsf{result}, \mathsf{bmi}\}$	$[\{x\} \mid \{y, x\} \leftarrow \langle\langle\_, \mathsf{result}, \mathsf{bmi}\rangle\rangle; x \geq 17; x \leq 19]$

Table 1. Schema object metadata example

The schema metadata of the other two schemas S_sql and S_csv can be defined in similar manner.

3.1 Schema Metadata Repository

A schema metadata repository is a distributed data store formed by a set of peers $\mathcal{P}$ in a data integration domain $\mathcal{D}$ for indexing and searching schema metadata contributed by peers in $\mathcal{D}$. Each peer P is assigned a unique identifier $\mathsf{p_k}$ (lower case p) by a mapping function $\mathcal{M}^{\mathsf{p}}$. Likewise a schema object metadata is given an identifier from a mapping function $\mathcal{M}^{\circ}$. The domain of function $\mathcal{M}^{\circ}$ is taken from the value domains of the elements of the schema object metadata. To ensure that there is always at least one peer found for a schema object metadata, both $\mathcal{M}^{\mathsf{p}}$ and $\mathcal{M}^{\circ}$ map to the same identifier range. We will refer to both $\mathcal{M}^{\mathsf{p}}$ and $\mathcal{M}^{\circ}$ as mapping functions.

Another essential component of the SMR is a routing strategy $\mathcal{R}$ executed on every peer to determine for a schema object metadata identifier so_i^m the next hop peer $\mathsf{p_k}$ which is either the peer responsible for so_i^m or the most favourable neighbour for forwarding so^m. Different routing strategies use different cost factors to determine what is meant by the most favourable neighbour of a peer.

In addition, we introduce to the SMR usage statistics $\mathcal{U}$ about the associated schema objects. These statistics are gathered and registered to the SMR by peers when they use a schema for integration and/or query processing. An example of usage statistics of a schema object so of a schema S would be a *quality* factor which is measured from the transformation pathway of S. As so is used by peers, its usage statistics are updated to take into consideration, for example, how frequently and accurately it is used for answering queries and/or how popular is its use in integrating peers schemas. The schema search function of the SMR discussed later can utilise these statistics to improve the search result.

Therefore we define an SMR as a tuple:

$$\text{SMR} = \langle \mathcal{D}, \mathcal{P}, \mathcal{SO}^{\star}, \mathcal{M}^{p}, \mathcal{M}^{o}, \mathcal{R}, \mathcal{U} \rangle$$

Note that by defining a SMR along the boundary of a data integration domain, we can more effectively manage the schema search space whilst at the same time allowing the system to easily scale to accommodate other related domains.

A peer participating in a SMR performs three high-level functions: (1) **register schema metadata**: indexes and registers a schema object metadata in SMR and (2) **search schema**: searches for the relevant schemas matching some criteria and (3) **update schema metadata**: updates the definition of a schema object metadata. Schema registration directly uses the mapping and routing functions of the SMR whilst schema search uses more sophisticated techniques from database and information retrieval. A formal treatment of the update function is outside the scope of this paper. Therefore, we will assume a naive update approach which deploys a time-to-live mechanism for automatically ageing out outdated metadata entries from the SMR.

Before explaining functions 1 and 2 in details, we first give an example of a DHT-based SMR and uses it as the basis for further discussion. The primary benefits of using DHT-based techniques are that they scale gracefully ($O\,\mathsf{log}(\mathsf{N})$) with the network size and they guarantee with high probability a random distribution of the value objects [15]. The latter focuses on load fairness among peers.

A DHT-based SMR uses DHT-based hashing and routing techniques [15] for indexing and organising schema metadata. The mapping function $\mathcal{M}^{\circ}$ is a family hash function $\mathcal{H}$ which maps for each $\mathsf{so}^{\mathsf{m}} \in \mathsf{SMR}$ an identifier $\mathsf{so}_{\mathsf{i}}^{\mathsf{m}} = \mathsf{h}(\mathsf{so}^{\mathsf{m}}.\mathsf{scheme})$ ($\mathsf{h} \in \mathcal{H}$). To preserve the structure of the schema object scheme through hashing, we would enforce that $\mathcal{H}$ be order-preserving (e.g. [16]). This gives us the flexibility of indexing and searching schema object metadata based on the matchability rather than the exact equality of schema object schemes.

In fact, a number of DHT-based techniques, such as P-Grid [17], use a prefix-based routing strategy which naturally supports the matchability property of our schema object schemes. For example in a P-Grid network, each peer is responsible for storing a pool of value objects which share a common prefix. Further, a peer's decision to serve (or forward) a search request of a value object is based on whether there exists a common prefix between its identifier (or its neighbours') and the object key. Therefore, by our definition when two schema object schemes are matchable it is highly likely that they share a common prefix and would therefore be located at the same peer.

The concept of a DHT-based SMR will become clearer when we discuss the schema metadata registration function with an example from P-Grid in the next section.

3.2 Schema Metadata Registration

In principle, to register a schema object metadata so^{m} of a public schema S to the SMR, a peer P first generates an identifier $\mathsf{so}_{\mathsf{i}}^{\mathsf{m}} = \mathsf{M}^{\circ}(\mathsf{so}^{\mathsf{m}}.\mathsf{scheme})$. After that, peer P applies the routing strategy $\mathcal{R}$ to forward so^{m} to a destination peer p_{k}. Figure 5 illustrates this using P-Grid to index and organise the schema object metadata of the example in Section 2. We will assume the general case in which there are seven peers named arbitrarily as shown. Three of these seven peers are the peers of the three research groups in our example.

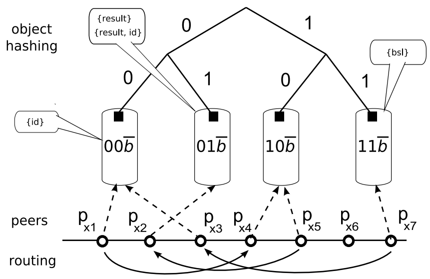


Fig. 5. An SMR example using P-Grid

Using P-Grid, we would initially distribute the schema object metadata when peers first join into the network. A peer and its neighbour divide the pre-constructed key space into halves starting from a common prefix of the peers' identifiers. Once these peers have arranged the key distribution among them, they can start exchanging schema object metadata accordingly. Figure 5 shows, for example, how the schema object metadata of schema $\mathsf{S_{xml}}$ would be registered. The schema object metadata $\{\mathsf{result}\}$ and $\{\mathsf{result}, \mathsf{id}\}$ are both located at the peers responsible for the tree path with prefix 01 because they share the first scheme element. On the other hand, $\{\mathsf{bsl}\}$ and $\{\mathsf{id}\}$ do not share a common prefix and are pushed to two separate paths 11 and 00 respectively. As shown in Figure 5 the peers $\mathsf{p_{x5}}$ and $\mathsf{p_{x3}}$ are responsible for paths 10 and 00 respectively. Further, for fault tolerance, peer $\mathsf{p_{x1}}$ is also assigned to the tree path 00.

Each peer keeps in its routing table the identifier of its first neighbour and the identifiers of one or more peers at the same level but on the other side of the identifier tree. These peers act as next hops for keys that do not map to the tree path of a peer. In our example, peer $\mathsf{p_{x1}}$ keeps a reference to peer $\mathsf{p_{x4}}$ for routing objects with key prefix 1 while peer $\mathsf{p_{x7}}$ keeps a reference to peer $\mathsf{p_{x3}}$ for routing objects with key prefix 0.

3.3 Search Schema

Intuitively, searching for schemas comes down to looking for the relevant schema object metadata in the SMR that satisfy a predicate-based search query $\mathsf{Q^s}$ which is formulated from a set of schema object schemes $\mathsf{SSchemes}$ and their predicates SPreds. The search scheme predicates SPreds are used for filtering schema object metadata based on the satisfiability of their summary queries. To further reduce the number of schema objects returned from search, we introduce an optional quality factor qa^ℓ which asks the search algorithm to rank the result set based on the schema object usage statistics and to return only those above the specified threshold. With this in mind, we define the general schema search problem as follows:

Definition Given a search query $\mathsf{Q^s} = \langle \mathsf{SSchemes}, \mathsf{SPreds} \rangle$ and a quality factor qa^ℓ retrieve the set of schema object metadata $\mathsf{SO^M}$ such that $\forall \mathsf{so^m} \in \mathsf{SO^M}$ the followings are true: (1) $\exists\, \mathsf{sc} \in \mathsf{SSchemes}$ and its predicate $\rho_{\mathsf{sc}} \in \mathsf{SPreds}$: $\mathsf{match}(\mathsf{so^m}.\mathsf{scheme}, \mathsf{sc}) \wedge \mathsf{checkSatisfiability}(\mathsf{so^m}.\mathsf{q}, \rho_{\mathsf{sc}})$; (2) $\mathsf{so^m}.\mathsf{schema}$ is *fully complied*; (3) $\mathsf{rank}(\mathsf{so^m}) \geq \mathsf{qa}^\ell$

Property 1 states that the schema metadata object $\mathsf{so^m}$ must match with at least one of the searched schemes and also satisfy all the predicates associated with the matched schemes. Property 2 ensures that the schema to which $\mathsf{so^m}$ belongs is fully compliant, i.e. none of its schema metadata objects which match with query $\mathsf{Q^s}$ and yet violates the $\mathsf{checkSatisfiability}$ condition of Property 1.

The match function lexically compares two schemes for either an **exact** or a **partial** match. We use exact matching if search schemes predicates are specified and partial matching when there are no predicates. In the later case, we will allow the searching application to specify a custom k-factor value with the match function through which to fine-tune the search results.

The $\mathsf{checkSatisfiability}$ function is a special case of the query answering using views problem [18]. For example, in the context of the MiniCon [19] algorithm which we implemented for query processing our search schema algorithm can be considered as a distributed form of the first phase in which the MiniCon descriptions are formed. Therefore, checkSatisfiability runs faster than a general-purpose query answering using views algorithm because we omit the final phase in which the satisfied views are combined into the final rewritings.

Property 3 ranks and compares the result schemas based on the quality factor qa^{ℓ}. The ranking algorithm needs to consider usage statistics associated to the schema object metadata in order to measure the schema object quality. We plan to tackle the search problem with property 3 as part of our future work.

We outline in Algorithm 1 the basic search schema algorithm for a DHT-based SMR which supports Properties 1 and 2.

Algorithm 1 $p_n.searchSchema(Q^s)$

Require: query $\mathsf{Q^s} = \langle \mathsf{SSchemes}, \mathsf{SPreds} \rangle$
Ensure: set $\mathsf{SO^M}$ to contain all schema object metadata that both match and satisfy $\mathsf{Q^s}$
1 set $\mathcal{S}^{\neg}$ to contain identifiers of all schemas that are not fully complied with $\mathsf{Q^s}$.
2 **for** search scheme $\mathsf{sc} \in \mathsf{Q^s.SSchemes}$ and its predicate $\rho \in \mathsf{Q^s.SPreds}$ **do**
3 Generate search key $\mathsf{k} = \mathsf{h(sc)}$ and look up peer $\mathsf{p_k}$ for k
4 Forward sub-query $\mathsf{q^s} = \langle \{\mathsf{sc}\}, \{\rho\} \rangle$ to $\mathsf{p_k}$ and execute $\mathsf{p_k.checkSatisfiability(q^s)} \rightarrow \langle \mathsf{SO^M_k}, \mathcal{S}^{\neg}_{\mathsf{k}} \rangle$
5 Add $\mathsf{SO^M_k}$ to $\mathsf{SO^M}$, $\mathcal{S}^{\neg}_{\mathsf{k}}$ to $\mathcal{S}^{\neg}$
6 **end for**
7 Remove from $\mathsf{SO^M}$ all schema object metadata whose schema is in $\mathcal{S}^{\neg}$

8 $p_k.checkSatisfiability(q^s)$:
9 set $\mathsf{SO^M_k}$ to contain resulted schema metadata objects
10 set $\mathcal{S}^{\neg}_{\mathsf{k}}$ to contain identifiers of all schemas that do not satisfy the predicates of $\mathsf{q^s}$
11 Retrieve the set $\mathsf{SO^{M'}_k}$ of all schema metadata objects matching sc
12 Add to $\mathsf{SO^M_k}$ only $\mathsf{so^m} \in \mathsf{SO^{M'}_k}$ which (1) $\mathsf{so^m.q}$ satisfies ρ and (2) $\mathsf{so^m.schema} \notin \mathcal{S}^{\neg}$
13 Add to $\mathcal{S}^{\neg}_{\mathsf{k}}$ the identifiers of schemas of all metadata entries which do not satisfy condition (1)
14 return $\langle \mathsf{SO^M_k}, \mathcal{S}^{\neg}_{\mathsf{k}} \rangle$

Note that because of the final filtering step (line 7), the body of the search loop (lines 3, 4 and 5) can be executed in parallel for all search schemes. This would significantly reduce the time delay of the overall execution. Further, in the case of a partial match at line 11 the set $\mathcal{S}^{\neg}_{\mathsf{k}}$ will be empty because the condition (1) on line 12 will not be checked.

To illustrate this let us suppose $\mathsf{p_{x5}}$ is asked to search for all schemas containing schema objects about BSL and NHS id and that the NHS id values are less than 104. The formulated search query to ask peer $\mathsf{p_{x5}}$ is $\mathsf{Q^s} = \langle \{\mathsf{id}, \mathsf{bsl}\}, \{(\mathsf{id} \leq 104)\} \rangle$. Using P-Grid as the underlying DHT protocol of the SMR in our example scenario, $\mathsf{p_{x5}}$ first determines that its tree path $10\overline{\mathsf{b}}$ does not share a common prefix with key $00\overline{\mathsf{b}}$ of id nor does it fully match the prefix $11\overline{\mathsf{b}}$ of bsl. Thus, peer $\mathsf{p_{x5}}$ looks up in its routing table to identify neighbour peer $\mathsf{p_{x7}}$ responsible for prefix 11 and the next hop peer $\mathsf{p_{x2}}$ to the path prefix 00. Next, $\mathsf{Q^s}$ is reformulated into two sub-queries: query $\mathsf{q^s_1} = \langle \{\mathsf{id}\}, \{(\mathsf{id} \leq 104)\} \rangle$ to be forwarded to $\mathsf{p_{x2}}$ and query $\mathsf{q^s_2} = \langle \{\mathsf{bsl}\} \rangle$ be forwarded to $\mathsf{p_{x7}}$.

At peer $\mathsf{p_{x7}}$, the same prefix-checking process is repeated but this time peer $\mathsf{p_{x7}}$ is in fact responsible for bsl. However, because $\mathsf{q^s_2}$ does not contain any predicates, all three metadata entries matching bsl are returned to peer $\mathsf{p_{x5}}$. Peer $\mathsf{p_{x2}}$ only shares the first bit 0 with the searched key of result, thus, proceeds with looking up its routing table for the neighbour p_{x1} to forward query q^s_1. At peer p_{x1}, three exact matches are found for scheme id but only the metadata entries from two schemas S_{xml} and S_{sql} satisfy the predicate ($\mathsf{id} \leq 104$). Therefore, peer p_{x1} returns to peer p_{x5} two metadata id entries and one unsatisfied schema identifier s_{csv}.

Finally, peer p_{x5} aggregates all search results it has received from peers p_{x2} and p_{x7} and based on it remove the metadata entry bsl of the unsatisfied schema S_{sql}. As a result, the execution of $\mathsf{p}_{x5}.\mathsf{searchSchema}(\mathsf{Q}^s)$ gives the schema object metadata of two schemas with identifiers $\{\mathsf{s}_{xml}, \mathsf{s}_{sql}\}$.

Extension to the basic search algorithm A limitation of Algorithm 1 is that it does not require the searched schemas to support all search schemes which may result in irrelevant schemas being found. However, this situation would not necessary happen if we take the view that all SMR peers are in the same domain and thus contain related schemas. But even if there are irrelevant schemas it is possible to filter them through a schema matching process [20] which is part of schema integration.

It is possible to extend Algorithm 1 to support a sub-set of or all schemes in the search. To do this, we first represent $\mathsf{Q}^s.\mathsf{SSchemes}$ as a set of scheme sets. We then generalise the definition of the sub-query q^s in line 4 to contain a set of schemes and a set of associated predicates. Finally, we extend the checking of condition (1) in line 12 to support conjunctive query rewriting similar to the first phase of MiniCon.

The URLs of the peers where the schemas are located can be found in the schema object metadata and we can wrap the definitions of schemas S_{sql} and S_{xml} from these peers onto p_{x5}. Now that we have the necessary schemas on p_{x5} we can integrate them to create the public schema.

4 Schema Integration

We use the BAV data integration approach [4] which provides an alternative to the more commonly used GAV and LAV [21] approaches. In BAV, schemas are mapped to each other using a sequence of *bidirectional* schema transformations called a **pathway**. The technique readily supports schema evolutions [5] which can be expressed as extensions to existing pathways. This feature makes BAV well suited to 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.

A BAV pathway is created by the repeated application of one of the 5 primitive BAV transformations: **add** and its inverse **delete**, **extend** and its inverse **contract** and **rename**. The **add** and **delete** transformations include a query that defines the extent of the new or removed schema object in terms of the extents of existing schema objects. **extend** and **contract** are used when it is not possible to define the extent of the new or removed schema object in terms of existing schema objects.

The public schema we create in this way is shown in Figure 4(d). Example 1 shows some of the BAV transformations needed to transform S_{sql} into the public schema. Figure 6 shows a intermediate HDM graph after transformation 7. At the moment our technique requires the manual creation of a public schema at the query peer but we hope in future to add some automation to the process. Some work in this regard has already been done in our group [20].

These transformations create the pathway shown on the left hand side of p_{x5}. We can now extend this pathway by transforming S_{xml} in the public schema using a similar set of transformations. The bidirectional nature of BAV means that we now not only have a pathway from the data sources to the public schema but also a reverse pathway that gives us a view of both relevant data sources from the public schema. This is shown in Figure 7. We use this view to answer our query and return the result: $\{3,3,4,5\}$.

Example 1 Creating the public schema from the SQL schema

1 addNode($\langle\langle$result$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$nhs_num$\rangle\rangle]$)
2 addNode($\langle\langle$id$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$id$\rangle\rangle]$)
3 addNode($\langle\langle$bsl$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$bsl$\rangle\rangle]$)
4 addNode($\langle\langle$bmi$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$bmi$\rangle\rangle]$)
5 addEdge($\langle\langle$_, result, id$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, nhs_num, id$\rangle\rangle]$)
6 addEdge($\langle\langle$_, result, bsl$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, blood, bsl$\rangle\rangle]$)
7 addEdge($\langle\langle$_, result, bmi$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, weight, bmi$\rangle\rangle]$)
8 moveDependents($\langle\langle$_, nhs_num, id$\rangle\rangle$, $\langle\langle$_, result, id$\rangle\rangle$)
9 moveDependents($\langle\langle$_, weight, bmi$\rangle\rangle$, $\langle\langle$_, result, bmi$\rangle\rangle$)
10 moveDependents($\langle\langle$_, blood, bsl$\rangle\rangle$, $\langle\langle$_, result, bsl$\rangle\rangle$)
11 deleteEdge($\langle\langle$_, nhs_num, id$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, result, id$\rangle\rangle]$)
12 deleteEdge($\langle\langle$_, weight, bmi$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, result, bmi$\rangle\rangle]$)
13 deleteEdge($\langle\langle$_, blood, bsl$\rangle\rangle$, $[\{x, y\}|\{x, y\} \leftarrow \langle\langle$_, result, bsl$\rangle\rangle]$)
14 deleteCons($\subseteq$, $\langle\langle$idblood$\rangle\rangle$, $\langle\langle$id$\rangle\rangle$)
15 deleteCons($\subseteq$, $\langle\langle$idweight$\rangle\rangle$, $\langle\langle$id$\rangle\rangle$)
16 contractEdge($\langle\langle$_, weight, idweight$\rangle\rangle$, Void, All)
17 contractEdge($\langle\langle$_, blood, idblood$\rangle\rangle$, Void, All)
18 contractNode($\langle\langle$weight$\rangle\rangle$, Void, All)
19 contractNode($\langle\langle$blood$\rangle\rangle$, Void, All)
20 deleteNode($\langle\langle$weight$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$idweight$\rangle\rangle]$)
21 deleteNode($\langle\langle$blood$\rangle\rangle$, $[\{x\}|\{x\} \leftarrow \langle\langle$idblood$\rangle\rangle]$)

□

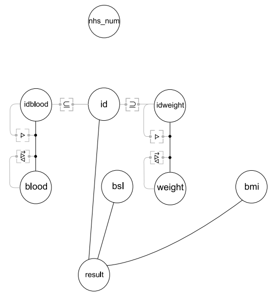


Fig. 6. Intermediate HDM graph

5 Related Work

We have identified the following research areas as relevant to the work presented in this paper.

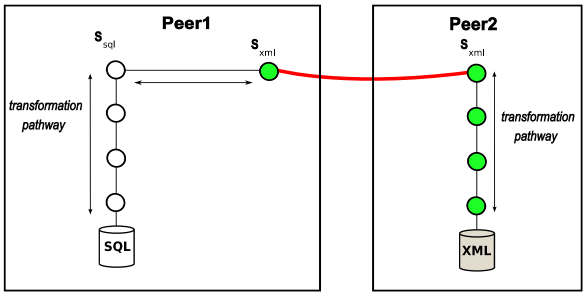


Fig. 7. Extending a Transformation Pathway between Peers

Model Management A central theme in the current P2P schema integration systems such as Piazza [7] and Edutella [9] is the use of a common data model for defining the mappings between schemas represented in different local data models. These systems both use RDF [23] to describe both query and peer correspondences. A limited number of data models now have representations in RDF but there is not the wide support that HDM provides. The languages used to transform RDF [12] are also not as flexible or as well suited to the P2P environment as BAV.

Schema Metadata Management These aforementioned systems only consider direct mappings between peers. The underlying assumption in these systems is that there is an implicit global knowledge about peer schemas on the network so that a peer can contact any other peers for any schemas that meet its information needs. As such, these systems do not address the issue of schema metadata distribution and discovery which we feel is essential for add-hoc data integration and query processing in P2P environment.

We use an information retrieval technique for distributing and searching schemas similar in spirit to PeerDB [6]. However, PeerDB does not tackle the data integration problem as with our approach. It also relies on a simple description of a schema metadata using relation and attribute constructs from the relation model. The keyword-based search used by PeerDB may return schemas which do not actually contain any data satisfying the user's requirement. Edutella is better at data integration but restricts the reusability of schema metadata on the P2P network by coupling distributed query processing with schema metadata management at super peers. This can lead to unbalanced load distribution among the super peers whereby certain popular super peers are always busier than less popular peers.

The HDM data model allows us to have a generic graph-based representation of any integrated schemas on the network. As a result we have a schema metadata model which is independent of the underlying data models of the peer schemas. Further, to improve the quality of schema search, we define in each schema metadata definition of a schema object a query that summarises the extent of that schema object. This enables the use of a predicate-based search query to more efficiently filter out unsatisfactory schemas. We also decouple query processing and schema metadata distribution to reduce the likelihood of overloading specific peers.

6 Conclusion

In this paper, we have described a method for performing inter model data integration in a P2P environment by using a generic CDM and a bidirectional data integration method. We formally defined schema metadata management as a distributed component which enables peers to effectively organises schema metadata and makes them available for discovery on the P2P network. We are implementing the features discussed in this paper on top of an integration system called AutoMed (http://www.doc.ic.ac.uk/automed). We plan to extend the schema search algorithm to take into account usage statistics for ranking schemas and their objects.

References

1. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. **18**(4) (1986) 323–364
2. M. Boyd, P.McBrien: Comparing and transforming between data models via an intermediate hypergraph data model. To appear (2004)
3. Williams, D., Poulovassilis, A.: Combining data integration with natural language technology for the semantic web. In: Proc. of Workshop on Human Language Technology for the Semantic Web and Web Services, at ISWC'03. (2003)
4. McBrien, P., Poulovassilis, A.: Data integration by bi-directional schema transformation rules. In: ICDE. (2003) 227–238
5. McBrien, P., Poulovassilis, A.: Schema evolution in heterogeneous database architectures, a schema transformation approach. In: CAiSE. (2002) 484–499
6. Ng, W.S., Ooi, B.C., Tan, K.L., Zhou, A.: PeerDB: A P2P-based system for distributed data sharing. In: Proc. of ICDE. (2003)
7. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Piazza: Data management infrastructure for semantic web applications. In: Proc. of WWW2003, Budapest, Hungary (2003)
8. Franconi, E., Lopatenko, A.: The coDBz information integration system for autonomous data sources. In: Proc. of Interop Workshop, Porto, Portugal (2005) http://www.inf.unibz.it/ franconi/coDBz/.
9. Wolfgang Nejdl *et al*: EDUTELLA: A P2P networking infrastructure based on RDF. In: Proc. of WWW2002. (2002)
10. Florescu, D., Kossman, D.: Storing and querying xml data using an RDBMS. Bulletin of the Technical Commitee on Data Engineering **22**(3) (1999) 27–34
11. Bowers, S., Delcambre, L.M.L.: The uni-level description: A uniform framework for representing information in multiple data models. In: ER. (2003) 45–58
12. Sintek, M., Decker, S.: Triple - a query, inference, and transformation language for the semantic web. In: International Semantic Web Conference. (2002) 364–378
13. McBrien, P., Poulovassilis, A.: A general formal framework for schema transformation. In: Data and Knowledge Engineering. Volume 28. (1998) 47–71
14. Hall, P.A.V., Owlett, J., Todd, S.: Relations and entities. In: IFIP Working Conference on Modelling in Data Base Management Systems. (1976) 201–220
15. Balakrishnan, H., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Looking up data in p2p systems. In: Communications of ACM. (2003)
16. Fox, E.A., Chen, Q.F., Daoud, A.M., Heath, L.S.: Order preserving minimal perfect hash functions and information retrieval. In: Proc. of SIGIR'90, New York, NY, USA, ACM Press (1990) 279–311

17. Aberer, K.: P-Grid: A self-organizing access structure for P2P information systems. Proc. of CoopIS 2001 **2172** (2001) 179–194
18. Halevy, A.: Answering queries using views: A survey. VLDB Journal **10**(4) (2001) 270–294
19. Pottinger, R., Halevy, A.: MiniCon: A scalable algorithm for answering queries using views. VLDB Journal **10**(2–3) (2001) 182–198
20. Rizopoulos, N., McBrien, P.: A general approach to the generation of conceptual model transformations. In: CAiSE. (2005) 326–341
21. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS. (2002) 233–246
22. Smith, A., Le, D.M.: Inter model data integration in a p2p environment - extended version. Technical report : http://www.doc.ic.ac.uk/automed/techreports/index.html (2007)
23. Manola, F., Miller, E.: Rdf primer. W3C (2004) Available from http://www.w3.org/TR/rdf-primer/.