# Self-Extending Peer Data Management

Ralf Heese Sven Herschel
{rheese|herschel}@informatik.hu-berlin.de
Humboldt-Universität zu Berlin
Databases and Information Systems
Berlin, Germany

Felix Naumann Armin Roth
{naumann|aroth}@informatik.hu-berlin.de
Humboldt-Universität zu Berlin
Information Integration Group
Berlin, Germany

**Abstract:** Peer data management systems (PDMS) are the natural extension of integrated information systems. Conventionally, a single integrating system manages an integrated schema, distributes queries to appropriate sources, and integrates incoming data to a common result. In contrast, a PDMS consists of a set of peers, each of which can play the role of an integrating component. A peer knows about its neighboring peers by mappings, which help to translate queries and transform data. Queries submitted to one peer are answered by data residing at that peer and by data that is reached along paths of mappings through the network of peers.

The only restriction for PDMS to cover unbounded data is the need to formulate at least one mapping from some known peer to a new data source. We propose a Semantic Web based method that overcomes this restriction, albeit at a price. As sources are dynamically and automatically included in a PDMS, three factors diminish quality: The new source itself might store data of poor quality, the mapping to the PDMS might be incorrect, and the mapping to the PDMS might be incomplete. To compensate, we propose a quality model to measure this effect, a cost model to restrict query planning to the best paths through the PDMS, and techniques to answer queries in such Web-scale PDMS efficiently.

## 1 An Ever-growing PDMS

The step from centralized database systems (DBMS) to distributed and then to federated database systems (FDBMS) removed the assumption that data must be located at the same site as the query. A federated database provides a global schema that represents the data it can access locally and remotely. The global schema is related to the local schemata via *schema mappings*, which specify how the schema of a local database maps to the global schema. The federated database accepts a query against its global schema and distributes it according to the schema mappings to the different sites where the data resides. Those sites execute the partial queries and send results back to the requesting peer. Again, the schema mappings specify how data is to be translated to conform to the global schema. The results are further processed and combined to be finally fused into a single response to the user.

A natural extension to this paradigm is to remove the assumption that queries are only

asked against a single integrating site. Peer data management systems (PDMS) are built of multiple peers, each of which provides a schema and accepts queries against the schema. Again, the peers are connected by mappings among their schemata. However, instead of forming a tree with a single root, each peer can be connected to any number of other peers. Queries against a schema of one peer can be answered using the data of the entire PDMS, as long as appropriate mappings have been formed (see Fig. 1). In general, a query

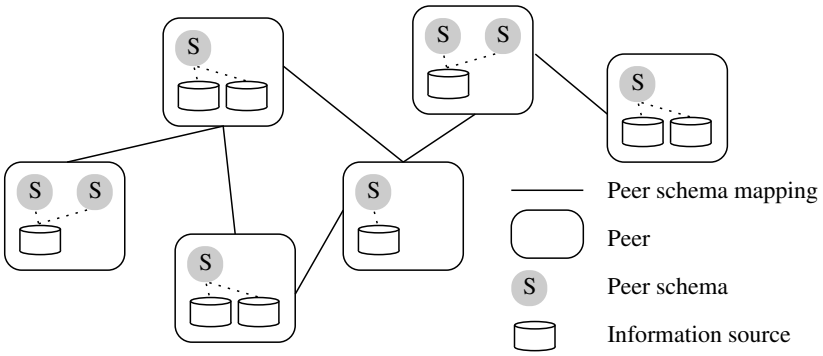


Figure 1: Peer Data Management System

planner iteratively follows mappings through the network of peers until all reachable and potentially useful peers have been visited. Each peer executes a part of the query and sends the result back along the path to the peer where the query originated. That peer then is left with the task of assembling a final result. In this sense, a peer has multiple roles: It acts as a data provider and it acts as the global component of a federated system. Furthermore, peers can act as a mediator, merely passing along queries without contributing to the result. The main advantage of a PDMS over federated DBMS is their flexibility. A new peer only needs to generate a mapping to the schema of some (similar) peer and is thus immediately part of the PDMS. That is, queries submitted to the new peer can be answered by data from other peers (if they are reachable by the mapping) and the new peer's data can be used to answer queries at other peers. We note, however, that this need for at least one mapping (preferably more) is an obstacle towards scaling up a PDMS to Web-size. In addition, the expansion of conventional PDMS is not dynamic at query-time but performed manually at setup-time.

In this paper we propose an architecture for a Web-scale PDMS. As in a conventional PDMS, each peer has a schema against which queries can be posed and each peer has mappings to one or more other peers. Additionally, a peer can have a mapping to one or more ontologies. Ontologies reflect a common and agreed upon semantics of a given application domain. While it has been argued that creating and maintaining an ontology is very difficult, there is reason to believe that in the near future there will be prominent ontologies for many domains. In fact, there already are prominent examples of ontologies that are heavily in use today, such as the Gene Ontology [gen04]. We use ontologies to *short-circuit* long paths through schema mappings that support the same ontology. We describe how a classical query planning algorithm is extended to allow for mappings through

an ontology back to another peer. If the discovered peer is relevant, a direct mapping for future use is constructed.

A Web-scale PDMS yields the additional problem of locating sources that are relevant to the query at hand – a problem which the peer-to-peer (P2P) research community is trying to solve. In this paper we propose the use of P2P indexing techniques to discover relevant and yet unreachable peers. This is accomplished by annotating each peer's *peer model* with quality vectors on multiple levels in order to find the locally optimal set of peers to forward the query to.

Allowing a PDMS to scale in such a dynamic and unsupervised way comes at a cost: The information quality of a query result can be poor, i.e., does not satisfy the user's demands. First, data sources on the Web often store data of poor quality. Data can be outdated, erroneous, of dubious origin, incomplete, etc. Second, the mappings leading to the data can also be incomplete or incorrect. Allowing a PDMS to scale to such a potentially enormous size also comes at a cost: Query planning and query execution can become rather inefficient. The more sources are available, the more alternatives a query planner must consider. The more sources are in a query plan, the more data must be shipped back through the PDMS to the querying peer. Taken together, there is great potential to render a PDMS useless. To solve this dilemma, we propose a data quality model that serves two goals: pruning query plans and ranking query results. Query plans in a PDMS are represented by trees that follow mappings through the network of peers. Extending a query plan to the last scrap of information of a dubious source is surely unnecessary in most cases. Thus, to reduce query planning and query execution time, we use a quality measure to abort planning once the added value of the next planning step is below a quality threshold. Because the sources of data have been evaluated qualitatively during planning, it is a simple task to also rank the query results according to their quality, reducing the effect of users being overwhelmed by enormous amounts of information.

**Structure of this paper.** The paper draws from several different research areas, which we review in Section 2. In the order of our previous arguments, we iteratively add to the final goal of a universal PDMS: Along the lines of known PDMS we define a simplified model of a peer data management system with its peers, schemata, and mappings in Section 3. To allow automated and semi-automated growth of a PDMS we present techniques of the semantic web and how they are applied in our context of mapping between heterogeneous schemata in Section 4. There, we also introduce the overall architecture of our system. Because unlimited growth is not always useful to users, Section 5 introduces quality considerations that demonstrate the trade-off between quality and efficiency. Unlimited growth also complicates the selection of data sources with high quality, so Section 6 shows how new techniques of the P2P area guide query planning algorithms. Section 7 concludes this paper presenting an outlook on the further steps required to reach this goal.

## 2 Related Work

Integrated access to information that is spread over multiple, distributed and heterogeneous sources has been recognized as an important problem by many researchers in the past years. The *mediator-wrapper architecture* [Wie92] defines a framework to solve important problems, such as schema heterogeneity and information overlap. In such systems, the mediator stores a schema (mediator schema), which semantically subsumes the interesting parts of the source schemata. Technical and syntactical heterogeneity in the sources is hidden by wrappers which offer a uniform interface to the mediator. In [NLF99] we extend the ideas to the quality driven information integration. They propose algorithms for query plan generation based on information quality criteria. In contrast to our approach the system only enables a static and centralist integration of domain-specific information sources, e.g., sources are tightly coupled to a centralized mediator before query execution. Furthermore, we locate relevant information sources and compute query plans dynamically at query execution time.

The literature reports on two principally different approaches for PDMS. One approach is reflected in the Piazza project of Halevy et al. [HIST03], the work of Bernstein et al. [BGK+02], and the work of Aberer et al. [ACMH03], which propagate queries and data only along mappings between peers. On the other hand, the Edutella approach [LNWS03] uses a semantic overlay networks so-called super-peers to distribute queries to suitable peers. None of the approaches mentioned above deals with web-scale networks of peers.

The mediation between schemata of a PDMS is the main concern of [HIST03]. Concessions to the quality of query results are mentioned, but not discussed in detail. New algorithms usable for the optimization of query reformulation are contributed by Tatarinov and Halevy [TH04]. However, they are independent of information quality, which remains an open challenge according to the authors.

The approach of [ACMH03] uses cycles in mapping networks to detect loss of information. That is, instead of explicitly modeling information quality as in our approach, the authors use instance sampling to assess IQ criteria. Since a simple data model is used, mappings are defined only between attributes and cannot contain selections.

In a so-called schema-based peer-to-peer network like Edutella semantic overlay networks consist of clusters (super-peers) of semantically "similar" peers [LNWS03]. Every peer is assigned to exactly one super-peer, which distributes queries mapped to its integrated schema. Thus, this approach does not utilize direct mappings between peer schemata. However, this approach is similar to ours as our ontologies can be compared with their super peer schemata. In our approach however, the ontologies are only *one* possible way to route queries.

Consideration of *inaccuracies* and *uncertainties* in query processing for PDMS is an important research perspective [MBDH02, AC03]. In [MKIS00] Mena et al. admit inaccurate mappings between concepts of ontologies and the corresponding loss of information is examined. Another promising approach to deal with uncertainty of mappings is given by the statistical technique discussed in [AC03].

## 3 A Simple Model of a PDMS

Our formal framework for a PDMS combines elements from [CGLR04] and [HIST03]. We formalize a PDMS $\Pi$ as a set $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ of peers each of which comprises local data sources and mappings both to the local sources and to other peers.

### 3.1 Peers, Mappings, and Semantics

In general, a single peer is perceived as a data integration system consisting of a *peer schema* $S$ and some local data sources. Additionally, other peers can play the role of external data sources for a certain peer. The peer schema describes data that the peer provides to other peers.

Local data sources are specified by a set $\mathcal{R}$ of *stored relations* and are connected to the peer schema by a set $\mathcal{M}_L$ of *local mappings*. Other peers are related to a peer schema by so-called *P2P-mappings* (peer-to-peer mappings) contained in the set $\mathcal{M}_P$. P2P-mappings act as semantic relationships between peers. To summarize formally, we have a peer represented by the tuple $P = (S, \mathcal{R}, \mathcal{M}_L, \mathcal{M}_P)$. We extend this definition of a peer during this paper.

Our approach is based on so-called GLAV mappings (global-local-as-view mappings) both for local and P2P mappings. This means that local mappings are of the form $Q_R(\mathcal{R}) \subseteq Q_S(S)$, where $Q_R$ and $Q_S$ are *conjunctive queries* of same arity. Note that $Q_R$ may contain joins over the stored relations in $\mathcal{R}$.

Similarly, a P2P-mapping $Q_1(\mathcal{P}_1) \subseteq Q_2(\mathcal{P}_2)$ establishes a relationship between the peer schemata of the two *sets* of peers $\mathcal{P}_1$ and $\mathcal{P}_2$. As in [HIST03], $Q_1$ and $Q_2$ may refer to any relation of the respective peer schemata. Intuitively, this means that $Q_1$ always returns a subset of the result set of tuples of $Q_2$. Hence, the GLAV mappings are directed from $\mathcal{P}_2$ to $\mathcal{P}_1$. In practice, we usually encounter mappings where $Q_1$ and $Q_2$ are queries over a *single* peer, respectively. The special case where $Q_1$ is a query over the peer $P_2$ and $Q_2$ is a single relation without projections amounts to the GAV formalism. In that approach, queries are reformulated by unfolding the appropriate views. On the other hand, if $Q_1$ is a single relation without projections, then $Q_1$ resembles the special case LAV [MH03]. This formalism requires an algorithm for answering queries using views [Hal01].

When posing a query to a peer, the user perceives a PDMS as a single database. The semantics of a PDMS define the meaning of the answer to a query. To define the semantics of a PDMS, we resort to a logical formalism [CGLR04]. For each peer we introduce a first order logic theory $T_P$. It consists of an alphabet containing all relation symbols of a peer schema $S$ and the stored relations of $\mathcal{R}$. The axioms of $T_P$ comprise all constraints of $S$ and one logical formula representing each local GLAV mapping. We extend our formalism to the complete PDMS by considering a source database $\mathcal{D}$ for the PDMS $\Pi$ that is the disjoint union of the source databases of all peers, i.e. the stored relations mentioned in our framework.

The semantics of $\Pi$ with respect to $\mathcal{D}$ is the set of all logical interpretations of $\Pi$ relative to $\mathcal{D}$ that satisfy all peer theories $T_{P_i}$ and all formulas following from all P2P-mappings. Given a query $Q$ submitted to a peer $P_i$ the certain answers to $Q$ based on $\mathcal{D}$ are the set of tuples after applying $Q$ to every of the logical interpretations above.

### 3.2 Query Answering in PDMS

To translate a query posed to a peer schema, all of the query's relations (subgoals) are treated separately. They are reformulated and passed along the mappings between peers. If a peer acts as a mediator the resulting graph of reformulations branches, thus building a tree. Within each branch the process terminates when a cycle is found or a peer has used all relevant mappings to its neighboring peers. The leaves of this search tree contain stored relations of local data sources. All correct query plans can be determined from the search tree as described in [HIST03].

**Example:** Consider the PDMS depicted in Fig. 2. Note that only the peers $P_1$ and $P_2$ are part of the PDMS since $P_3$ is not connected by any P2P mapping. We assume that the following simple P2P mapping is given: $P_1$.Jobs(id, job_desc, organization, start_date, naics) $\supseteq$ $P_2$.Joboffer(id, job_desc, industry, organization, start_date). Now we show how the query q $\leftarrow$ Jobs(id, 'Admin', 'Novell', '2005-01-01') posed to $P_1$ is processed in our simple PDMS. This query is equivalent to the SQL query:

```
SELECT id FROM Jobs
WHERE job_desc='SystemAdmin' AND organization='Novell'
  AND start_date='2005-01-01'
```

First, peer $P_1$ uses its local mapping to reformulate the single subgoal (Jobs) of the Datalog query into its stored relations which are omitted in the figure. Then it passes this query to peer $P_2$ using the mapping $m$ to $P_2$. Peer $P_2$ follows the same procedure. Since it has no further mappings, the construction of the reformulation tree terminates. Based on this tree query plans are generated. During the query execution phase the reformulated queries are evaluated against the stored relations and their results are merged into the overall query result returned by $P_1$.

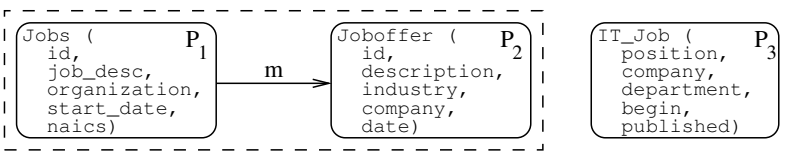


Figure 2: Example of schema mappings between peers.

## 4 Adding the Semantic Dimension

In a PDMS a peer can only be involved in query answering if there exists at least one mapping to some peer of the PDMS. That is, a peer is only reachable through its mappings. In a small PDMS consisting of about a hundred peers this approach may guarantee a complete and correct answer. However, scaling up a PDMS to Web-size we face additional problems. With the size of the network the number of different peer schemata and the number of possible mapping paths in the PDMS increases. Therefore, not all peers which could contribute to the result of a query can be reached due to the increasing length of the mapping path and due to response time constraints. Longer mapping paths cause the query to be reformulated many times. As a consequence it is more likely that the semantics of a query cannot be preserved, e.g., attributes cannot be mapped. To achieve a better reachability in a conventional PDMS we would have to find similar peers in the PDMS and generate schema mappings manually.

Instead of generating a large number of mappings, our approach places an ontology layer on top of a conventional PDMS architecture. We use this layer to discover relevant peers that were previously not reachable by mapping paths. In a second step, we automatically construct schema mappings between the peer issuing the query and the discovered peers. This establishes a direct P2P-mapping, a so-called *shortcut*, between the peers. The shortcut is stored locally at both peers to answer similar queries without resorting to the ontology layer in the future. In this section, we first extend the simple model of the PDMS to reflect ontologies and mappings between peers and ontologies. Afterwards, we describe the query processing in the PDMS using ontology mappings.

### 4.1 Extending the PDMS Model by Ontologies

The lower part of Fig. 3 shows a conventional PDMS including peers $P_i$ with their peer schemata $S_i$ and some schema mappings $m_i$. The upper part of the figure displays the ontology layer of the *Extended PDMS*. This layer comprises ontologies $O_i$, mappings $o_i$ between them, and mappings $p_i$ between peers and ontologies. From our point of view ontologies reflect common and agreed upon semantics of a given application domain. In this paper, we do not focus on the creation and matching of ontologies but assume the existence of ontologies and mappings between them. The relationships between the concepts of different ontologies are represented by OWL class and OWL property axioms, e.g., *owl:equivalentClass, owl:intersectOf, rdfs:subClassOf*. Each mapping between a peer schema and an ontology simply assigns each schema element, e.g., a table name or an attribute of a table, to a concept of an ontology.

Formally, we extend our simple model of a PDMS from Section 3 to $\Pi = (\mathcal{P}, \mathcal{O}, \mathcal{M}_O)$, where $\mathcal{O} = \{O_1, \ldots, O_n\}$ denotes the set of ontologies known by some peers of the PDMS and the set $\mathcal{M}_O = \{o_1, \ldots, o_m\}$ contains mappings between these ontologies. A peer is defined as the tuple $P = (S, \mathcal{R}, \mathcal{M}_L, \mathcal{M}_P, \mathcal{M}_{PO})$ where $\mathcal{M}_{PO} = \{p_1, \ldots, p_k\}$ represents the set of mappings of the peer schema $S$ to concepts of some ontologies from $\mathcal{O}$

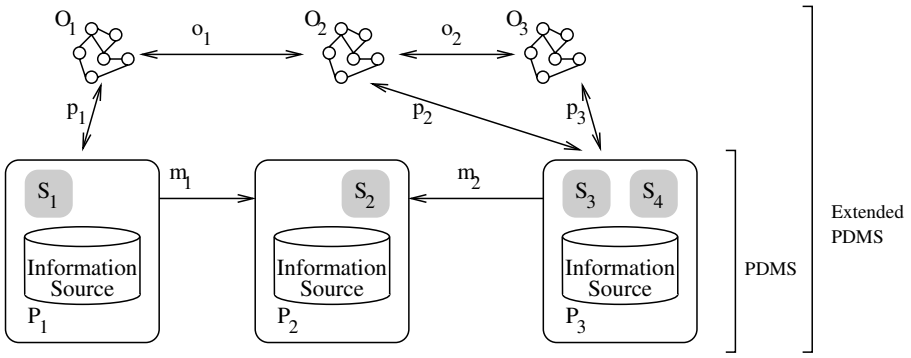


Figure 3: Architecture of the Extended PDMS

(compare Section 3.1). Similar to the schema mappings between peers, the mappings from schemata to ontologies must be constructed manually.

### 4.2 Query Processing

Before a subgoal of a query is passed to another peer in a conventional PDMS, it is translated to conform to the target schema by applying a GLAV mapping. A query is passed to other peers as long as appropriate mappings exist. By adding the ontology layer more mapping paths become available. Hereby, two essential problems arise: (i) Discover the peers that have semantically similar schemata and (ii) generate a direct schema mapping between these peers. The solution of the first problem is addressed in Section 6 while we discuss the second problem now.

Fig. 4 illustrates the basic idea of using ontologies for query processing in an extended PDMS. After a relevant peer has been identified by the ontology layer, the source peer generates a direct mapping, a so-called *shortcut*, to this peer. Having a shortcut defined the peer translates the query into the terms of the target peer and forwards the query.

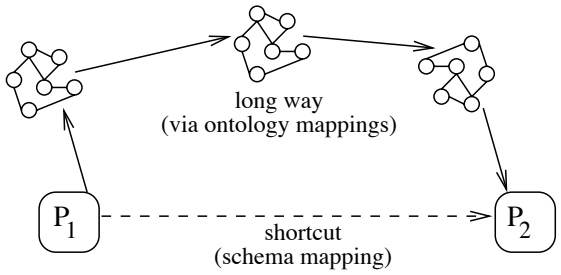


Figure 4: Using ontology mappings to generate a direct mapping between peer schemata (shortcuts)

The generation of mappings between relational schemata is the goal of schema match-

ing algorithms, which have been in the focus of research for several years [MBR01, Doa02, RB01]. While these approaches mainly focus on finding semantically similar schema elements, we use inference over ontology mappings to compute matching elements. As a result we obtain a path through the ontologies between two schema elements. This path consists of concepts of the ontologies which are interlinked by properties like *owl:equivalentClass, owl:intersectOf, rdfs:subClassOf*. For example, if two schema elements are linked by *owl:equivalentClass* then the source schema element can simply be substituted by the target schema element. The property used to map schema elements has an impact on the quality of the query result, e.g., using *rdfs:subClassOf* implies that the result contains less (or more) tuples than expected.

In our running example there only exists a peer mapping between peer $P_1$ and $P_2$ (see Fig. 2). While a query issued at the first peer can easily be translated to the schema of $P_2$, we need additional mappings to query peer $P_3$. Table 1 shows an example mapping from the schemata of $P_1$ and $P_3$ to the concepts of a simple job ontology[1]. For instance, the mapping of *start_date* and *begin* to the concept *job:start_date* indicates that both properties are used with identical meaning. Therefore, *begin* can be substituted by *job:start_date* and vice versa in a query.

| $P_1$ | $\Rightarrow$ | $O_{Job}$ | $\Leftarrow$ | $P_3$ |
|---|---|---|---|---|
| Jobs | $\rightarrow$ | job:jobPosition | $\leftarrow$ | IT_Job |
| job_desc | $\rightarrow$ | job:description | $\leftarrow$ | position |
| organization | $\rightarrow$ | job:organization | $\leftarrow$ | company |
| | | job:department | $\leftarrow$ | department |
| start_date | $\rightarrow$ | job:start_date | $\leftarrow$ | begin |
| naics | $\rightarrow$ | job:naics | | |
| | | job:publish_date | $\leftarrow$ | published |

Table 1: Example mapping of $P_1$ and $P_2$ to a sample job ontology

## 5 Adding the Quality Dimension

In huge PDMSs the creation of the complex and highly branched search tree consumes considerable time and storage. In the spirit of [TH04] our goal is to optimize this process and to apply semantic and quality criteria on properties of the query result. Our goal is to find an optimal trade-off between efficiency and quality of query results. In this section we expand our view on PDMS by the aspect of information quality (IQ) and describe open challenges in this area.

[1] In [BHM+05] Bizer et al. describe the development of a job portal including a job ontology.

### 5.1 Information Quality and Loss of Information in PDMS

Information quality is an important discriminator of data sets. It is usually perceived as an aggregation of several IQ-criteria [WS96]. We highlight three content-related IQ-criteria that are especially important for our context of PDMS:

*Extensional completeness* describes the proportion of the size of a set of objects to the number of *all* objects accessible within a PDMS. The measure is applicable both to the data set provided by a peer and to the result of a query. For a query result it is based on the overall number of objects that fulfill the query predicate. Our goal is to establish a causal relationship between the extensional completeness of the local data sources at the peers and an integrated query result.

*Intensional completeness* can be defined (orthogonally to the extension) as the proportion of the schema elements of a certain data set and the schemata (intension) of *all* peers. Here the challenge is to maximize the intension accessible to a user. An intensionally complete query result provides data for all schema elements (usually attributes) mentioned in the user query.

*Relevancy* is the degree of conformance of a query result with the information demand of a user. Understanding the needs of a user requires knowledge about his or her semantic interpretation of the queried schema, which usually only can be assessed. As mappings mediate between heterogeneous sources, estimating their impact onto relevancy of query results is an important problem.

Loss of information means the decrease of the scores of the above IQ-criteria [MKIS00]. In [NFL04] we report how concessions to the completeness of query results enable mediator-based information systems to increase the efficiency of query planning drastically. The problem of query planning for PDMS is more subtle, because data sources cannot be accessed directly. Rather, the query planner must determine on the fly, which mapping paths reach data sources that contribute to the query result. Since the loss of information is propagated along such mapping paths, reasoning about IQ-criteria and the corresponding concessions for query answering is especially valuable for PDMS. We see considerable potential in pruning the search tree using IQ-criteria. Mapping composition (see [MH03]) executed prior to query processing provides a chance to use IQ-criteria as well. Providing suggestions for new peer mappings might be another valuable result of the IQ-analysis. Achieving all these goals requires understanding how the schema mappings between peers influence the information quality of query results. To extend these ideas to web-scale, we take up these quality criteria in Section 6 and refine them to apply to specific concepts of an ontology, thus supporting web-scale ontology-based query routing.

### 5.2 Impact of Mappings on Information Quality

This section shows how extensional and intensional completeness of a query result are influenced by a mapping. Consider the simple mapping depicted in Fig. 5. It connects the peers $P_1$ and $P_2$, having possibly heterogeneous schemata. Let the mapping $M_{P_1 \to P_2}$

consist of a formula $R_{21}(a, b), R_{22}(b, c) \subseteq R_{11}(a, c)$. Intuitively, this means that both $P_1$ and $P_2$ provide data for relation $R_{11}$. Note that multiple occurrences of the same variable on one side of the formula indicate a join (here a join over attribute $b$).

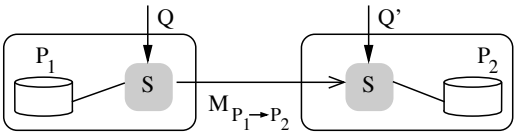


Figure 5: Query reformulation along a mapping.

To answer a query $Q$ posed over relation $R_{11}$ of $P_1$, it has to be evaluated both at $P_1$ and $P_2$. Because $P_2$ uses a different schema, the query is reformulated using the mapping $M_{P_1 \to P_2}$. This amounts to a global-as-view style query processing. Thus, in $Q$ the relation $R_{11}$ is replaced using the mapping and the reformulated query $Q'$ contains a join between $R_{21}$ and $R_{22}$.

Given IQ-scores for the relations at $P_2$, our goal is to determine the IQ-score of the query results considering both peers. We assume that queries and mappings only use select-project-join queries. Based on completeness scores of the relations of peers $P_1$ and $P_2$, and using theorems of [NFL04], we calculate the expected extensional and intensional completeness for the result of the join over $R_{21}$ and $R_{22}$.

Selections in mappings may be used to express implicit knowledge about peer schemata. For instance, in writing a mapping to a peer of a certain company that offers information about products one can assume, that the peer *only* models products of this company, unless the company deals with products. Most works about data integration exclude *projections* from being part of schema mappings. In contrast, we allow mappings to comprise projections to map between peer schemata which provide intensionally different data.

Selections and projections in the user query and the mappings influence completeness scores: Selection predicates in the user query potentially increase extensional completeness, because, intuitively, less information is being asked for. Selection predicates in the mappings potentially decrease extensional completeness, because less information is provided. Likewise, projections in query and mapping influence intensional completeness: User queries with projections intuitively ask for less data elements, thus increasing the chance of a source to intensionally fulfill the entire query. Mappings with projections, on the other hand, decrease the chance of a source to provide the schema elements asked for, thus possibly decreasing intensional completeness.

Extending these concepts to the semantic level introduced in the last section, quality degression is interpreted as the loss of information. Quality degression is caused by peers that do not support all concepts or properties of the ontology or by erroneous mappings between the peer schema and the ontologies. In all cases the quality indicator is used to terminate the query processing procedure once a user-defined quality threshold is reached.

## 6 Adding the Web Dimension

The goal of this paper is to extend current PDMS into several interesting directions, one of which is the web dimension, enabling PDMS to scale to large numbers of peers. One contribution to accomplish this goal is the consideration of the loss of information quality described in Section 5, where the loss of information quality along the P2P mappings is used as a termination criterion for query evaluation. In this section we introduce a distributed semantic index used to facilitate lookups of suitable information sources even if they are not reachable through the peer mappings (see Section 3). The index is based on ontological annotation and thus heavily relies on the semantic dimension introduced before (see Section 4).

For a better understanding, the index is introduced as a centralized index to demonstrate its concepts. Later we generalize to a distributed index by adding index update mechanisms.

### 6.1 Concept Stores in Peers

The underlying idea for finding suitable information sources is to match a graph representation (i.e. an RDF graph: see figure 6) of the query against the graph representations of all locally known peers and then to forward the query to the peers best meeting the requirements. If the returned data is satisfactory to the requesting peer, it may generate a relational mapping using the ontology mappings along the path and use it as a *relational shortcut*, making it much faster to reach the target peer in the future (see figure 4). To provide this functionality, we need the following concepts:

**Peer Graph.** Each peer in our PDMS network exports its *peer graph* containing all concepts and properties supported by this peer. The *peer graph* is a graph $G_{pm} = \{V, E\}$ of the supported resources $V$ connected by the property edges $E$. It can be interpreted as the result of applying the ontology mappings $\mathcal{M}_{PO}$ onto the peer schema. Each *peer graph* is further annotated with several quality criteria. Annotated peer graphs are stored in a *Concept Store*.

**Quality Vector.** The peer graph is annotated with a *quality vector* using a quality annotation function $q : \mathcal{P} \rightarrow \mathcal{Q}$, which associates each peer with its quality vector. The components of a quality vector represent the values of quality criteria described below.

**Query Graph.** A query graph $G_q$ is an RDF graph that is constructed by applying the peer-ontology mapping to the submitted query. The `SELECT` clause of the original query is represented in $G_q$ by a parameter marker '?', which must provide an *rdf:type* property to indicate its type. Following the notation of RDF graphs, Fig. 6 shows the query graph for the query of our running example.[2]

[2]Theoretically, parameter markers might also occur at edges, i.e., as properties, but this type of binding is not useful in our scenario where mappings originate from the relational model. Thus, querying meta-information about data objects, e.g., "Which attribute does the value $v$ belong to?" is not a supported query.

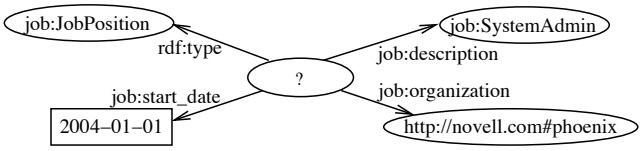


Figure 6: Query Graph

Fig. 7 shows our architecture extended by *concept stores*. Our formal peer model from Section 4 is therefore extended to $P = (S, \mathcal{R}, \mathcal{M}_L, \mathcal{M}_P, \mathcal{M}_{PO}, C)$, with $C$ representing the concept store of a peer.

Note that concept stores may comprise concepts of arbitrary ontologies.

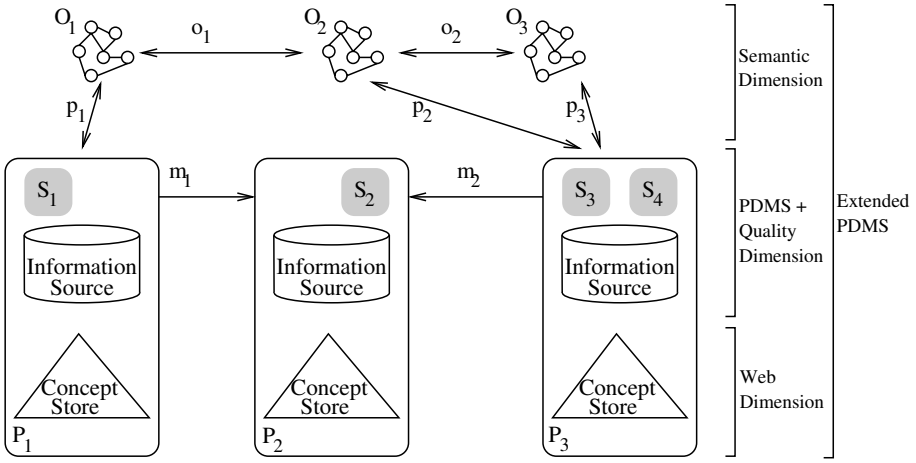


Figure 7: Architecture of the Extended PDMS

Figure 7 shows the distributed version of the extended PDMS where each peer maintains its own concept store. As mentioned before, the explanations in the next section will refer to a centralized index for a better understanding.

### 6.2 Graph-Based Peer Selection

#### 6.2.1 The quality vectors

To rank information sources semantically, we propose the following properties that will be stored for each peer model. We distinguish three levels of properties: ontology, concept, and concept-property level.

On the ontology level the quality annotation function $q$ associates values for the quality criteria *concept coverage* $CC$, *timeliness* $TLN$, and *peer count* $PC$ to each peer. Hereby,

$CC$ is the number of concepts supported by this peer relative to the number of concepts contained within the ontology. We interpret this ratio as a completeness indicator on the schema level. $TLN$ indicates the freshness of the stored data. This might influence the query planner when reliable, up-to-date information is important. $PC$ is the number of peers known to the respective peer supporting this specific ontology.

The function $q$ assigns to each peer a value for *instance count* $IC$ on the concept level. It is the number of instances stored at the peer for each concept. We use this as an indicator of the amount of data available at the data source, i.e., a completeness indicator on the instance level.

Furthermore, $q$ associates a *Property-Concept Instance Count* $PCIC$ to each peer. This count represents the number of instances of a specific concept type actually annotated with a specific property. It can be interpreted as the level of detail, i.e., number of non-null-values of the specific peer.

Table 2 summarizes the definitions of the quality criteria mentioned above.

| | |
|---|---|
| **Ontology Level** | |
| concept-coverage $C$ | $C = \frac{C_{support}}{C_{total}}$, where $C_{support}$ is the number of concepts of the ontology supported by this information source and $C_{total}$ is the total number of concepts in the ontology. |
| timeliness $TLN$<br><br>the semantic annotations about the information source | indicator of the up-to-dateness of the semantic annotations about the information source. The higher the timeliness, the better. |
| peer count $PC$ | number of other peers known to the data source to support this ontology. |
| **Concept Level** | |
| instance count $IC$ | number of instances for the given concept. Can be interpreted as extensional completeness in the relational model. |
| **Concept-Property Level** | |
| property-concept instance count $PCIC$ | number of instances of a specific concept type which are annotated with a given property. Similar to intensional completeness in the relational model. |

Table 2: Parameters for semantic ranking of peer model in a concept store

### 6.2.2 Determining suitable peers

To determine whether the current peer may execute the query, it consults all peer models in its concept store and searches for sets of peers supporting the concepts requested in the query graph. In a second step, the peer sets are ordered according to the quality requirements. To be able to answer a query, the peer must meet the following conditions:

**Concept coverage.** All concepts of the query $Q$ must be known in the concept store: $Concepts(Q) \subseteq \bigcup V$ where $\bigcup V$ is the set of all concepts known in the concept store of the peer.

**Concept-Property coverage.** All concept-property-combinations must be covered, i.e. it is not sufficient for query processing to cover a concept by some peer and the property by another. Both belong inextricably together and may not be separated: $\forall (v, e) \in (V, E)_Q : (v, e) \in \bigcup (V, E)$ where $(V, E)_Q$ is the set of concept-property combinations in the query graph posed to the peer and $\bigcup (V, E)$ is the union of all concept-property combinations known to in the concept store of the peer.

Following these basic requirements the peers best suited to answer the query or a part of the query are determined using the following procedure:

**Graph matching.** Find a list of peer sets, each of them being able to answer the entire query, i.e., find peers such that the union of the peer models overlaps the entire query graph. This means that each concept-property combination must be supported by some peer in the group.

It is important to note that each query graph must have an *rdf:type* property at each parameter marker vertex. To be able to perform the graph matching the two vertexes connected by *rdf:type* are collapsed. This process is illustrated in Fig. 8.

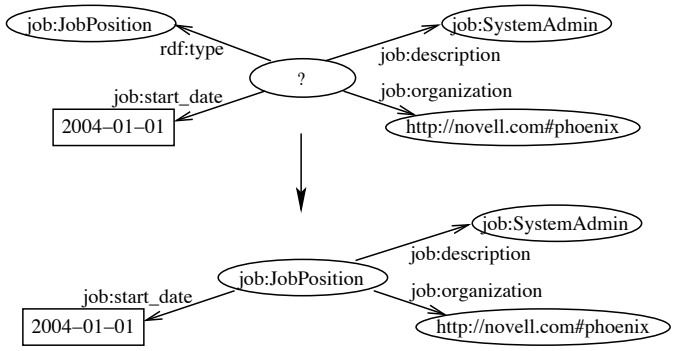


Figure 8: Preparing the query graph for comparison

The graph matching process is not very expensive because all vertexes are annotated with a unique URI for each concept. It is really not a matching process in the graph theoretical sense, but rather a search for matching concepts and their property-links.

**Ranking.** For each set in the list created above, a value needs to be calculated indicating how promising it is to send the query to the peers in the set. This value is calculated using the quality vectors introduced above. For this paper we use a simplistic model using only the peer count ($PC$) values – we expect the other values of the vector to prove useful in the future. When determining coverage of the query graph, the following cases might occur:

1. Only one peer covers the entire query: The query is forwarded to this peer.
2. Many peers, each by itself, cover the entire query: The query is forwarded to the N peers with the highest aggregate of the quality vectors, in our example to the peers with the highest $PCIC$ and $PC$ values for the requested concepts.
3. The entire query is covered only by a union of peers: The N best suited sets of peers are determined by appropriate aggregates of their quality vectors. In our example we use a function that considers the $PC$ and the $PCIC$ values. The latter criterion is important to achieve a high probability that "join predicates" can still be evaluated.
4. The entire query cannot be evaluated: In this case we abort query processing since no suitable peer exists for processing.

### 6.3 Distributing the Index

In the previous sections we assumed that there was a centralized index having global knowledge about all peer models of the extended PDMS. This has several disadvantages: The cost for operating the central index at web-scale is very high and the entire network depends on the correct operation of the index. Furthermore, entry barriers for potential new peers are substantially higher in a central-index scenario. The problem, however, of distributing the index, is far from trivial:

- Since only local knowledge exists, it is virtually impossible to find the optimal query processing strategy. A standard optimization approach is a similarity measure in order to facilitate the routing process (see "Experts and consultants" below).
- Efficient algorithms are needed to bootstrap and update the index.

In order to support the distributed index, each peer is extended by an individual *concept store* for other peers. We assume that a "standard" P2P infrastructure is already in place, i.e., using the JXTA[3] P2P framework. JXTA uses a peer/super peer approach with flooding as the peer search mechanism.

#### 6.3.1 Updating the Concept Store

When joining the network the peer retrieves the concepts stores of all 1-hop neighbors and updates its 1-hop neighbors, i.e., it serves as a hub for its neighbors. Whenever an index

[3] www.jxta.org

lookup is performed, the first N peers considered suitable are polled for their concept store which is integrated into the peer's concept store. Without further limitations this approach would lead to an index entirely materialized on each and every peer, an unacceptable solution.

Therefore, we propose mechanisms for peers which allow them to become *expert* or *consultant* peers.

### 6.3.2 Experts and consultants

Using additional mechanisms we expect the development of *expert* and *consultant* peers within the network. Casually spoken, experts have a deep knowledge of a limited area of interest while consultants support a large area of interest, independently of the amount of information they support.

This distinction is formalized as follows: A peer joins the network being an expert and turns into a consultant once the value $\frac{1}{|O|}\sum_i |PC_{O_i}|$ reaches a predetermined threshold. Here, $|O|$ is the number of ontologies supported by this peer and $|PC_{O_i}|$ is the number of peers supporting the respective ontology. The development of these two index peer types is supported by the following mechanisms:

**Forced Oblivion.** In order to keep the required storage space at each index peer at a fixed level, we require that each peer "forgets" peers in its catalog once the catalog is full or once the timeliness $TLN$ for a peer falls below a certain threshold. The entries to forget depend on the number of catalog entries for the concept about to be forgotten (many entries: *do not forget*) and the $TLN$ value (high $TLN$ value: *forget*). We will explore forced oblivion further in the future, and expect that it allows for query routing using semantic catalogs of limited size, for specialization of index peers, and for healing of network partitions due to the necessary re-exploration of the network once no suitable peers can be found.

**Concept Aggregation.** Once an index has cataloged a sufficient number of entries for one of the semantic levels *concepts* or *properties*, it may decide to aggregate this information. This means, that the peer does not store the individual information any more but aggregates it in the level above. In order to accomplish this, we store at the ontology level (for concepts) or the concept level (for properties) that the peer covers the respective concepts or properties excellently. We currently investigate exact mechanisms when content aggregation should occur and what kind of meta-information must be stored.

Using the concepts above we expect the ontological overlay to retrieve additional information sources in a PDMS that would not be found along the relational mappings of the PDMS. If it were possible to compose the mappings, a relational shortcut could be constructed connecting the two peers directly to facilitate the processing of future queries.

## 7 Conclusion and Future Work

In this paper we specified an extensible peer data management system (PDMS) to query large amounts of data or information in an heterogeneous environment. We introduced an ontology layer on top of a conventional PDMS to overcome the restriction to formulate at least one mapping to some known peer to integrate a new data source into the PDMS. Based on mappings between the peer schemata and ontologies, a peer-to-peer index network indexes long-living schema characteristics, e.g., the concepts and properties supported by a peer. Our novel approach enables a peer to easily discover semantically similar peers that would not have been found by processing the query using a conventional PDMS. Furthermore, the mappings to and from the ontologies provide valuable semantic information that are promising to simplify the automatic generation of mappings between the relational schemata of peers. Our extended PDMS enables a peer to automatically translate a query into the peer schema of a previously unknown peer. Finally, we discussed quality criteria to measure the quality of data sources and mapping paths. The motivation is to restrict query planning to the best paths through the PDMS and thus to improve scalability of the PDMS.

In the near future, we will evaluate our strategies and refine our quality model. In addition, the following issues are currently under investigation:

**Translator nodes.** Similar to the real world where many languages exist and people can still communicate with each other, we envisage translator nodes to bridge semantic heterogeneity. This concept works in the real world because there are much more interpreters than languages—we plan to bring forward this idea to our PDMS extension.

**Semantic annotations.** The quality annotations introduced in this paper form a basis for query planning and peer selection decisions. We now evaluate the impact that different semantic annotations have on these processes. Furthermore, the aggregation of the quality information stored in the local concept store will be the subject of further investigation.

**Schema matching.** In our current architecture a PDMS is self-extending in the sense that new peers and new mappings between peers are automatically discovered by using ontologies. The more conventional method of automatically finding mappings, once the target peer is known is through *schema matching* methods [RB01]. We plan to investigate the effectiveness of a combination of both approaches.

**Acknowledgments.** This research was supported by the German Research Society (DFG grants no. NA 432, GRK 316), and by the German Ministry of Research (InterVal Berlin Research Center for the Internet Economy).

## References


[AC03] E. Altareva and S. Conrad. Statistical analysis as methodological framework for data(base) integration. In *Proceedings of the International Conference on Conceptual Modeling (ER)*, 2003.

[ACMH03] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The chatty web: Emergent semantics through gossiping. In *Proceedings of the International World Wide Web Conference (WWW)*, 2003.

[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 the ACM SIGMOD Workshop on The Web and Databases (WebDB)*, 2002.

[BHM+05] Christian Bizer, Ralf Heese, Malgorzata Mochol, Radoslaw Oldakowski, Robert Tolksdorf, and Rainer Eckstein. The Impact of Semantic Web Technologies on Job Recruitment Processes. In *Proceedings of the 7th International Conference Wirtschaftsinformatik*, 2005.

[CGLR04] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical Foundations of Peer-To-Peer Data Integration. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, 2004.

[Doa02] A. Doan. Learning to map between structured representations of Data. PhD thesis, 2002.

[gen04] Gene Ontology Website. http://www.geneontology.org, 2004. Gene Ontology Consortium.

[Hal01] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4), 2001.

[HIST03] A. Y. Halevy, Z. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2003.

[LNWS03] A. Löser, W. Nejdl, M. Wolpers, and W. Siberski. Information integration in schema-based peer-to-peer networks. In *Proceedings of the Conference on Advanced Information Systems Engineering (CAiSE)*, 2003.

[MBDH02] J. Madhavan, P. A. Bernstein, P. Domingos, and A. Y. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2002.

[MBR01] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic Schema Matching with Cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 49–58, 2001.

[MH03] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2003.

[MKIS00] E. Mena, V. Kashyap, A. Illarramendi, and A. P. Sheth. Imprecise answers in distributed environments: Estimation of information loss for multi-ontology based query processing. *Intl. Journal of Cooperative Information Systems*, 9(4):403–425, 2000.

[NFL04] Felix Naumann, Johann-Christoph Freytag, and Ulf Leser. Completeness of Integrated Information Sources. *Information Systems*, 29(7):583–615, 2004.

[NLF99] Felix Naumann, Ulf Leser, and Johann Christoph Freytag. Quality-driven Integration of Heterogenous Information Systems. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 447–458, Edinburgh, UK, 1999.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[TH04] I. Tatarinov and A. Halevy. Efficient query reformulation in peer data management systems. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, 2004.

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

[WS96] Richard Y. Wang and Diane M. Strong. Beyond Accuracy: What data quality means to data consumers. *Journal on Management of Information Systems*, 12(4):5–34, 1996.