Benefit and Cost of Query Answering in PDMS

Armin Roth[1] and Felix Naumann[1]

Humboldt-Universität zu Berlin
Unter den Linden 6, 10099 Berlin, Germany
aroth,naumann@informatik.hu-berlin.de

Abstract. Peer data management systems (PDMS) are a natural extension to integrated information systems. They consist of a dynamic set of autonomous peers, each of which can mediate between heterogenous schemas of other peers. A new data source joins a PDMS by defining a semantic mapping to one or more other peers, thus forming a network of peers. Queries submitted to a peer are answered with data residing at that peer and by data that is reached along paths of mappings through the network of peers. However, without optimization methods query reformulation in PDMS is very inefficient due to redundancy in mapping paths.
We present a decentral strategy that guides peers in their decision along which further mappings the query should be sent. The strategy uses statistics of the peers own data and statistics of mappings to neighboring peers to predict whether it is worthwhile to send the query to that neighbor—or whether the query plan should be pruned at this point. These decisions are guided by a benefit and cost model, trading off the amount of data a neighbor will pass back, and the execution cost of that step. Thus, we allow a high scale-up of PDMS in the number of participating peers.

1 PDMS and Data Quality

Integrating semantically relevant information is a pressing problem. In practice, it can be observed that a decentralized P2P fashion of data sharing is preferred over centralized data integration systems. Users desire to pose queries to their own schema, and let the queries be transferred via schema mappings to similar peers in the neighborhood. Such requirements are addressed by *peer data management systems* (PDMS) [1–3]. Peers serve both as data sources and as mediators and queries are translated and transferred using semantic relationships between peers, so-called mappings, as shown in Fig. 1.

Example application areas include partnerships between companies for developing complex technical products, cooperations of scientific institutions, and ad hoc crisis management [2]. PDMS can also serve as a decentralized infrastructure for mediation between ontologies in the semantic web. Like any information system integrating data from autonomous sources, PDMS are vulnerable to poor data quality in the sources, poor mappings to the sources, and thus poor data

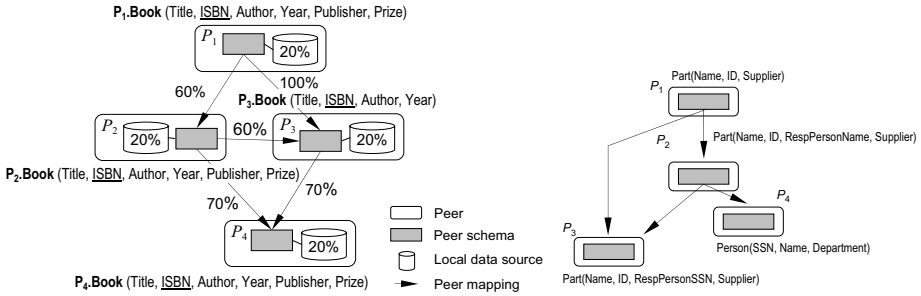

Fig. 1. Two example PDMS with annotated source coverages, mapping selectivities, and schema.

quality of query results. Compared to conventional integrated information systems, this problem is particularly large in PDMS. The data is passed through numerous mappings, each of which can decrease data quality by cumulative projections and selections.

In general, information quality (IQ) is an important discriminator of data sets. Here, content-related IQ-criteria are of major interest, including accuracy, relevancy, and completeness. Due to their autonomy, local sources at peers are likely to show quite different information quality. For this reason, consideration of information quality in query answering for PDMS consisting of a large number of peers is an important problem. In this paper we concentrate on the IQ criteria *coverage* and *density*, which together form an overall *completeness* measure for data sources and query results [4]. Current query planning algorithms, such as [2], find *all certain answers* to a query. This is not feasible in a web-scale PDMS: The search space becomes enormous and query plans become very long, increasing chance of information loss along the paths. In this paper we propose methods to free PDMS from this restriction and relax the notion of completeness turning it into an optimization goal instead.

Example 1. Consider the simple PDMS depicted on the left in Fig. 1. Its peers share a single relation Book. However, peer P_3 only exports four attributes, whereas the others provide two additional attributes, which are projected out by mappings. Furthermore, the mappings between the peers are annotated with selectivity scores, representing selection operations at the mappings. They are a measure for the fraction of data provided by the peer at the mapping head that is expected to be transferred via the mapping, e.g., the mapping from P_2 to P_3 removes all but 60% of the tuples provided at P_3.

To achieve our goal of efficient and effective query answering in large PDMS, this paper contributes a comprehensive completeness model for peers and mappings within a PDMS. We show how to calculate completeness scores for PDMS query plans. This can be used to improve the search for query plans providing high completeness by pruning mappings which show considerable loss of information. Furthermore, our completeness-aware query planning algorithm needs no central catalog, thus leaves maximum autonomy to the peers.

The remainder of the paper is organized as follows. The completeness model for PDMS is introduced in Sec. 2. Then, we provide a brief overview on query planning in PDMS in Sec. 3. Our approach to prune subplans is explained in Sec. 4 and experimentally studied in Sec. 5. Related work is discussed in Sec. 6 and we conclude in Sec. 7.

2 PDMS and Completeness

2.1 Data sources and mappings in PDMS

We formalize a PDMS 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.

Peers. In general, a single peer can be perceived as a data integration system consisting of a *peer schema* S and local data sources. The peer schema describes data that the peer provides to users, applications, and to other peers. Local data sources are specified by a set $\mathcal{L}$ of *local schemas*. They are connected to the peer schema by a set $\mathcal{M}_L$ of *local mappings*. Other peers are related to a peer schema by *peer mappings*, which form the set $\mathcal{M}_P$. In all, each peer is represented by the four-tuple $P = (S, \mathcal{L}, \mathcal{M}_L, \mathcal{M}_P)$. We use Datalog notation to express the relational data model used for schemata, queries, and mappings.

Mappings. Our approach is based on Global-Local-as-View mappings, or GLaV mappings. Local mappings are of the form $Q_L(\mathcal{L}) \subseteq Q_S(S)$, where Q_L and Q_S are *conjunctive queries*. Similarly, a peer mapping $Q_1(\mathcal{P}_1) \subseteq Q_2(\mathcal{P}_2)$ establishes a relationship between the relations of peer schemas of the two *sets* of peers $\mathcal{P}_1$ and $\mathcal{P}_2$. Intuitively, the peer mapping means that Q_1 always returns a subset of the resulting set of tuples of Q_2. GLaV mappings in their general form can be transformed to a combination of a GaV and a LaV mapping with a fresh relation symbol [2].

Selections play an important role when regarding the completeness of query results. Selection predicates in mappings express implicit knowledge about peer schemas. For instance, in writing a mapping to a peer of a company Ford and its relation Product, one can express that the peer models products of this company *only*. A way to express this selection in a GLaV mapping from a peer of a company AllParts, which offers knowledge about producers of parts, is to insert a selection predicate: Ford.Product(Name, ID, Supplier) $\subseteq$ AllParts.Part(Name, ID, Supplier), Supplier = "Ford". Also, there may be projections in local and peer mappings that can affect the completeness of query results. Consider a mapping from the Part relation of a company Bosch, which includes contact information, to the peer Ford. Because the Product relation of Ford does not offer this information, we must use a projection in the mapping: Ford.Product(Name, ID, Supplier) $\subseteq$ Bosch.Part(Name, ID, -, Supplier).

2.2 Completeness of data sets

In many scenarios, users of large integrated information systems are not interested in *all* certain answers, because they are not able to examine them all in

detail. Another constraint to large PDMS is the limitation of resources for query evaluation and transmission of query results. Facing these restrictions, a user may be satisfied with a small number of answers of highest quality

Coverage. Extensional completeness, also *coverage*, describes the proportion of the size of a tuple set to the number of *all* tuples stored within a PDMS. The measure applies both to the data set a peer actually stores and to a query result.

To calculate coverage we make the closed world assumption for the whole system. Users perceive a PDMS as a single database described by the schema of the peer. Thus, the size of the world $|w_Q|$ referred to by a query Q against this schema is the number of tuples matching the query that can be reached using the network of mappings. In practice, however, knowing this number precisely is not necessary, because it plays only the role of a normalizing factor.

Definition 1 (Coverage). *Let $\mathcal{D}_\mathrm{Q}$ be a set of tuples answering a query Q. The coverage of $\mathcal{D}_\mathrm{Q}$ with respect to a world w_Q is $c(\mathcal{D}_\mathrm{Q}) := |\mathcal{D}_\mathrm{Q}|/|w_Q|$.*

Density. The intension queried by the user is the set of attributes $\mathcal{A}_Q$ asked for in the query. Intensional completeness of data sets, also *density*, first suffers from `null` values in data sources. Secondly, attributes that are mentioned in the query may not be available at certain data sources in the PDMS. The user may be nevertheless interested in having tuples in the query result despite their missing attributes. Values of missing attributes are filled with `null` values, thus creating incomplete tuples. Attribute density is used as a measure for this kind of completeness. The query-dependent density is the arithmetic mean over all attributes occurring in a query.

Definition 2 (Attribute density). *Let a_R be an attribute of a relation R. A projection of a tuple t of this relation to a_R is denoted by $t[a_R]$. With $\perp$ denoting* `null`*, the attribute density of a tuple set $\mathcal{D}$ for R is defined as $d(a_R) := |\{t \in \mathcal{D} \,|\, t[a_R] \neq \perp\}|/|\mathcal{D}|$.*

Completeness. Intuitively, overall completeness can be regarded as an aggregated measure for the ratio of the amount of data in a certain data set to the amount of data in the world w_Q. It is a combination of coverage and density, which we aim to maximize. In [4] it is shown that the completeness score of a data set $\mathcal{D}$ can be calculated as $C(\mathcal{D}) = c(\mathcal{D}) \cdot d(\mathcal{D})$ and $0 \leq C \leq 1$.

3 Query Planning and Completeness

In this section we review a query planning procedure for PDMS and show how to value peer mappings according to their completeness. The following Section 4 uses this measure to prune poor plans or subplans.

3.1 PDMS query planning

To translate a query, the subgoals are reformulated and passed on along the mappings to other peers, which in turn recursively send the query to their neighboring

peers, etc. Reformulation terminates when all branches of recursion have reached local sources, where the queries can be evaluated on actual data. Clearly, this process can be performed fully decentrally. To show how the query reformulation actually uses the mappings between the peers and between the local sources and the peers, we briefly review the reformulation algorithm of [2].

Creation of the reformulation tree. Consider a query Q posed to some peer. We aim at a set of query plans, which only contain subgoals representing relations from *local* schemas. The answer to Q is the "union" of the results of all these query plans. The reformulation algorithm published in [2] constructs a so-called rule-goal tree (Fig. 2 on the left). The goal nodes are formed by (reformulated)

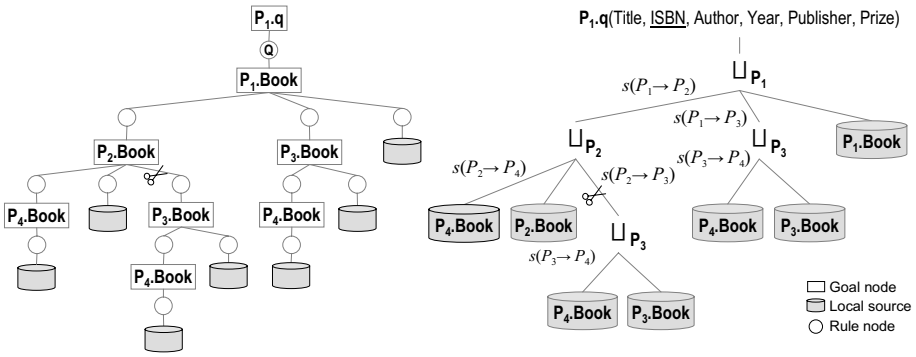

Fig. 2. Rule-goal tree (left) and query plan (right) of our example from Sec. 1. Selections are represented by the annotated mapping selectivities s.

subgoals to be answered, whereas the rule nodes represent the mappings. The algorithm continues by expanding leaf nodes using either peer or local mappings. Depending on the form of the mapping at hand, either a GaV- or a LaV-style reformulation is performed. In the former case new goal leaves are obtained by unfolding the view forming the mapping. If the mapping represents a view *from* any peer or local source on the schema to which the leaf goal node belongs, the MiniCon algorithm for answering queries using views is employed [5]. Please note that this algorithm may be performed fully decentrally by the peers.

Determining query plans. Two approaches are possible to create a query plan. To achieve first query answers quickly, several query plans may be determined sequentially in the way shown in [2]. In contrast, if we want to calculate the overall completeness of the query result, it is more useful to derive a single overall query plan from the rule-goal tree (Fig. 2). To obtain this single query plan, we recursively traverse the rule-goal tree. Several outgoing mappings at a certain goal node lead to a branch in the rule-goal tree. In our query plan all those subtrees are combined by a union operator. Branching rule nodes are created by a GaV expansion containing a join. Such a situation is reflected by a join operation in the query plan. Due to space limitations, we refer to [2] for transforming LaV expansion into the query plan.

3.2 Completeness of query plans

Intuitively, query reformulation for PDMS is a search problem. During exploration of the search space, we lack information about the completeness contribution of local data sources not reached yet. As a consequence, to intelligently explore the search space collecting high quality query results, we must decide which mappings promise to be useful. This leads to the question how mappings (and the data sources "behind" them) contribute to the overall completeness of the query result.

In this work, we assume the mappings to include only select-project-join (SPJ) queries. In the following, we show how to calculate the influence of S, P, and J operations used in the mappings on the coverage and density of query results. Additionally, query plans contain union-type operators, which collect results returned by alternative mapping paths starting from a certain peer.

Influence of selections and projections. Applying a selection σ to a tuple set of a relation R reduces the set of tuples by a selectivity factor s. Hence, we can calculate coverage of the selection result as $c(\sigma(R)) = s \cdot c(R)$. Assuming that `null`-values are distributed equally over all tuples, density is not affected by a selection $d(\sigma(R)) = d(R)$. If no statistics about s are available, sampling techniques may be employed to assess it (Sec. 7). Note that this selectivity is applied to the data of the target of a mapping but also on all other sources reachable through that mapping. This observation is the foundation of our heuristic for reducing the reformulation effort (Sec. 4).

Without concessions to the completeness of query answers, projection of query attributes would not be allowed in mappings. Attributes projected out have to be padded with `null`-values. Since a projection $R[\mathcal{A}_P]$, which reduces the attribute set $\mathcal{A}_R$ of R to the attribute set $\mathcal{A}_P$, leaves the number of tuples of R unchanged, the extensional completeness of the result is not affected: $c(R[\mathcal{A}_P]) = c(R)$. In contrast, the *query-dependent* density value is recalculated subtracting the density of the set of attributes projected out: $d_Q(R[\mathcal{A}_P]) = d_Q(R) - d_Q(R[\mathcal{A}_R \setminus \mathcal{A}_P])$. For simplicity, we assume here that projections do not reduce coverage, i.e., duplicates generated by projection are not eliminated. We use this observation in our heuristic to decide which mappings suffer from loss of information (Sec. 4).

Example 2. Suppose we are given the following mapping between two relations at different peers with the attribute densities listed thereafter: P_4.Book(Title, ISBN, Author, Year, Publisher, Price) $\subseteq$ P_3.Book(Title, ISBN, Author, Year) (Fig. 1).

	Title	ISBN	Author	Year	Publisher	Price
P_4	90%	100%	80%	60%	40%	70%
P_3	90%	100%	80%	60%	0	0

The attribute densities and the query-dependent density of $P_4.Book$ as it is exported by peer P_4 amounts to $d_Q(P_4) = 73\%$. Due to the two projections in the mapping (attributes Publisher and Price) the density of the *same* data set is perceived at P_3 with a value of $d_Q(P_3) = 55\%$, assuming the query asks for all attributes. As can be seen, projecting out a third of the attributes reduces

the query-dependent density by about a third in this example. It is important to note that we do not need any statistics to compare mappings wrt. loss of completeness due to projections.

Influence of joins. Suppose we are given the tuple sets of the two relations R_1 and R_2 together with their respective coverage and density values. We aim to calculate completeness for the result of the join $R_1 \bowtie R_2$. If we assume *independence* of the representation of objects, which means that there is no knowledge about extensional overlap, we can draw the following formulas for the expected coverage and density from [4], where $\mathcal{A}$ denotes the union of the attribute sets of R_1 and R_2:

$$c(R_1 \bowtie R_2) = c(R_1) \cdot c(R_2) \tag{1}$$

$$d(R_1 \bowtie R_2) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} (d_{R_1}(a) + d_{R_2}(a) - d_{R_1}(a) \cdot d_{R_2}(a)) \tag{2}$$

Influence of unions. In general, the contributions to a goal node's answer are not union-compatible. We use the full outerjoin-merge operator $\sqcup$ from [4], which is similar to the "outer union" but allows all common attributes but the key attribute to have conflicting data values. The following equations provide the expected coverage and density of a full outerjoin-merge for the case of independent data sets:

$$c(R_1 \sqcup R_2) = c(R_1) + c(R_2) - c(R_1) \cdot c(R_2) \tag{3}$$

$$d_{R_1 \sqcup R_2}(a) = \frac{d_{R_1}(a) \cdot c(R_1)}{c(R_1 \sqcup R_2)} + \frac{d_{R_2}(a) \cdot c(R_2)}{c(R_1 \sqcup R_2)} - \frac{d_{R_1}(a) \cdot d_{R_2}(a) \cdot c(R_1 \bowtie R_2)}{c(R_1 \sqcup R_2)} \tag{4}$$

Using this, the density of $R_1 \sqcup R_2$, which comprises the attribute set $\mathcal{A}$ is the arithmetic mean of the attribute densities: $d(R_1 \sqcup R_2) = 1/|\mathcal{A}| \sum_{a \in \mathcal{A}} d_{R_1 \sqcup R_2}(a)$. In [4], associativity is shown for the coverage criterion for $\bowtie$ and $\sqcup$; density is proven to be associative only for independent data sets.

Calculating query plan completeness. Using the means presented in the previous section, we now can calculate the *expected* completeness of query plans. In this way we may compare different strategies for completeness-driven PDMS query reformulation. Please note that such calculations cannot be performed by a single peer in a real PDMS, because it is based on global information. Rather, calculations are accumulated along peers as the reformulated query is passed on.

We pass along between the peers information about the aggregated mapping selectivity for every distinguished variable and the accumulated projections. Clearly, several selections based on the same variable may be aggregated along a mapping path by multiplying their selectivities, which we assume to be the result of statistical assessment methods (Sec. 7). If we further assume the variables of the user query being independent of each other, their respective mapping selectivities can also be aggregated by multiplication. The main idea of our algorithm to calculate the query plan completeness is to traverse the rule-goal tree recursively in the same way as for single query plan creation (Sec. 3.1), and to combine the completeness scores on the way back starting with the given values

at the leaf nodes, which represent only local sources. Combinations of coverage and density scores of subtrees are required at the occasions highlighted below. Coverage and density scores of data originating from the local sources are decreased by selectivities and projections in the mappings on the path to a certain peer. If every goal node returns its coverage and density based on the user query, we can calculate the coverage and density the receiving peer perceives using that peer's aggregated mapping selectivity and projections for the path from the peer to the root of the rule-goal tree.

Branching goal nodes. In this case we use the results from the rule node children, which are coverage scores referring to the goal node's peer. As in query plan creation we must perform a union and thus may use Equation (3) to calculate the resulting coverage. Attribute densities are calculated using Equation (4).

Branching rule nodes. To calculate the coverage scores of rule nodes, we use the results from the underlying goal nodes. As a branching rule node is created by a join in a mapping, Equation (1) can be employed to determine the resulting coverage. In the last step we multiply with the mapping selectivity $s(m)$ to return the coverage score of the data the above peer actually receives. With x denoting a distinguished variable of the user query occurring in the mapping m, the mapping selectivity is $s(m) = \prod_{x \in m} s_x(m)$. We use Equation (2) to propagate the attribute densities.

Our algorithm captures LaV expansions as well, but for brevity we omit this description. When our recursive algorithm reaches the root of the rule-goal tree we can calculate the query dependent density and, finally, the completeness of the query. We summarize the set of statistics we assume a peer should maintain in our PDMS setting: (i) Coverage and attribute densities for all local data sources, (ii)Selectivities for all selection predicates in outgoing peer mappings, and join selectivities for outgoing mappings that contain joins over different peers.

4 Pruning Subplans

The rule-goal tree typically becomes very large and shows a high branching factor even for relatively small PDMS with tens of peers [6]. Therefore, handling queries in web-scale PDMS using the algorithm from Sec. 3.1 is not feasible. To keep query reformulation in PDMS tractable for large PDMS, e.g., for semantic web applications [7], it is crucial to optimize both query planning and evaluation to reduce latency and determine first answers quickly.

To meet these problems, we exploit the influence of mappings on the query results to decide which mapping paths are not worth following or may be deferred. Our approach tries to identify mapping paths that preserve potential completeness of the intermediate query results "behind" these mappings. In general, during the query reformulation phase a peer has no knowledge about the completeness of data it will receive during query evaluation. In the rule-goal tree this fact is expressed by having subgoals of local sources only at the leaves of the tree.

If we want to avoid coordination between peers about completeness of intermediate query results, only the "completeness" of peer mappings may be exploited to prune the search space. We build on the results from Sec. 3.2 to characterize the influence of mappings on the completeness of query results in the PDMS context. In particular, we introduce a heuristic to either prune the search space or defer expansion of certain mappings to determine the first answers showing high completeness quickly. Because query reformulation is done recursively, every mapping that needs not be used potentially saves consideration of many more mappings later. Additionally, in a highly interconnected PDMS the probability that for a certain mapping path there are alternatives with less loss of information is considerable. In Sec. 5 we present experiments showing this effect.

Using selections and projections. Selections not overlapping with the user query may significantly reduce the amount of data transported by a mapping. Based on this observation we propose the following strategy for completeness-based query reformulation in PDMS: If a goal node is related to several mappings that can be used for expansion, we order them by their selectivity, favoring less selective mappings. To break ties, we additionally regard projections, favoring mappings with less projections. In a simple strategy, we only use a threshold for a normalized measure combining selectivity and the number of variables projected out to decide which mappings to follow and which to prune (Fig. 2). This strategy is fully decentral, i.e., no coordination between peers is needed at all. However, we lose answers. That is why this approach requires a trade-off between cost and benefit of the answer, which conforms to the concession to the completeness of query answers in large PDMS described above.

Using joins. Assessing the impact of joins on the coverage of (intermediate) query results is more subtle. There may be situations where very small join hit rates between the relations to be joined lead to considerable loss of information. In such cases it is desirable to have alternative mapping paths that may help to exploit more data from the join partners. The following example illustrates such a situation.

Example 3. Regard the PDMS on the right of Fig. 1. The relation Part models parts of technical products having attributes, such as name, identifier, social security number and name of the responsible person, and the supplier. Person is a simple list of persons that are referenced by the foreign key RespPersonSSN. Assume that P_1 is queried on parts of a certain supplier.

We consider a situation where the overlap between P_3.Part and P_4.Person is very small compared to the size of both relations. As a consequence, the join P_3.Part $\bowtie$ P_4.Person filters out all parts from P_3 where the corresponding author is not included in P_4.Person. It follows that P_2.Part may offer only a small fraction of parts stored at P_3. However, our query posed to P_1 does not ask for responsible persons. Hence, with respect to the query, the join performed at P_2 loses most of the "coverage" available at P_3. In recognizing this situation query reformulation could decide not to expand the goal node representing P_2.Part. Here, this would not affect coverage of the final query result, because there is

an alternative mapping from P_1.Part to P_3.Part, which may help to retrieve all parts at P_3.

As a conclusion from this example we propose a look-ahead strategy to handle *joins between peers*: (1) Check if the query to be answered requires to perform the join. (2) If so, try to assess the information overlap between the join relations. (3) If the *loss* of coverage is above a certain threshold, prune or defer the expansion performing the join. (4) If the joined peer that contributes tuples to the query result (P_3 in the example) can be reached on an alternative mapping path, finalize the decision of the last step. Observe, that this approach requires coordination between a small set of peers.

5 Experiments

Due to the complex structure of PDMS the effect of most algorithms and strategies can be validated only experimentally. Our PDMS implements the pruning strategy exploiting selections and projections in the peer mappings. The query reformulation is simulated on a single computer. However, we note that this does not imply any restrictions compared to a fully decentral query reformulation on multiple sites. In our experiments every peer covers only 5% of the size of the world. The following table lists the data sets and their characteristics:

	Peer schema	#Peers	*rank*
$\mathcal{P}_1$	Single relation	10	5.7
$\mathcal{P}_2$	Single relation	30	3.4
$\mathcal{P}_3$	Heterogeneous	50	2.5

Measurements. Alternative paths in highly interconnected PDMS lead to a quite strong convergence of the completeness to the final result, as can be seen in Fig. 3 on the left (solid line). There, the final coverage value is almost reached after about half of the cost for obtaining all answers has been spent. This means that in the second half of the query reformulation phase many mapping paths are exploited that are expected to contribute almost nothing to the final result. We use our approach of threshold-based pruning of mapping paths. Regard the results depicted on the left in Fig. 3. They show that pruning of mappings containing selections with medium selectivities (11% of all mappings concerned, dashed line) may still yield the same completeness as without pruning, but at half of the cost compared to the experiment without pruning. Moreover, if we choose an even stronger pruning condition (33% of all mappings concerned), the cost decreases to 10 times less than without any pruning. The results for the dataset $\mathcal{P}_2$ are displayed on the right in Fig. 3. It also shows a cost reduction of more than an order of magnitude along with improved completeness at any time during query reformulation. In this experiment maximally 20% of all mappings fell under the pruning condition. With the heterogenous PDMS $\mathcal{P}_3$ we yield similar results. There, about 10% of all mappings include selections and a fourth projects out some attributes. We pruned all mappings which are expected to lose more than a third of the data of the peer at their head (34 prunings and 1175

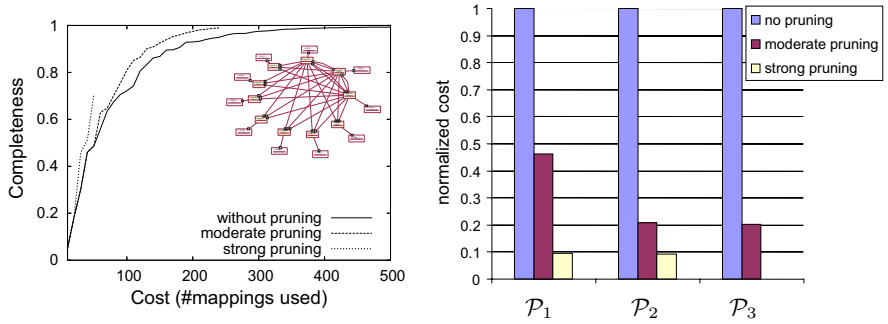

Fig. 3. Results and topology of $\mathcal{P}_1$ under different pruning thresholds and cost reduction. Normalized cost are based on the cost for using all mapping paths. Notice that ending graphs indicate a cost reduction (explicitly depicted on the right).

reformulations in total) while still achieving 100% completeness. Observe that pruning has to be applied with care: As depicted in the left-most graph in the left diagram in Fig. 3 very strong pruning can lead to a significant reduction of query answer completeness. Of course, choosing suitable pruning thresholds is a matter of experience. We believe that statistics about query answers could help (Sec. 7). In summary, our experiments clearly show effectiveness and considerable efficiency gains by applying a rather simple IQ-based pruning of the search space.

6 Related Work

In this section we review the PDMS literature under the aspects of information quality and efficiency of query answering. The mediation between schemas of a PDMS is the main concern of *Piazza* [2]. Concessions to the completeness of query results are mentioned, but not discussed in detail. New algorithms usable for safe pruning during query reformulation are contributed in [6]. However, they are independent from information quality, which according to the authors remains an open challenge. Additionally, this pruning approach involves non-local coordination between peers, whereas our mapping-based strategy is strictly local to autonomous peers. The *Semantic Gossiping* approach of Aberer et al. uses cycles in mapping networks to examine loss of information [1]. That is, instead of explicitly modeling completeness as in our approach, the authors use instance sampling to assess information quality criteria. The authors use a simple data model and mappings only between attributes and without selection queries. Calvanese et al. [3] propose new semantics for PDMS based on epistemic logic, which leads to general decidability. In this semantics only consistent facts are exported by a peer. However, the "weaker" logic loses some of the answers compared with our first order logic approach. In *Edutella*, semantic overlay networks consist of clusters of semantically "similar" peers [8]. This approach does not utilize arbitrary mappings between peer schemas. According to [9], inaccuracies and uncertainties in mappings are an important research perspective, which we have not adopted yet.

7 Conclusions

Peer data management systems offer a decentralized and dynamic infrastructure to share heterogeneous data between autonomous peers. To scale PDMS to a large number of peers it is crucial to optimally trade-off between the cost of query execution and the benefit of the query answers. We presented a solution for PDMS query reformulation that exploits completeness characteristics of mappings between peers. First, we described the influence of GLaV mappings on the completeness of query answers. Next, we introduced a fully local strategy to prune those mappings that have a high expected information loss based on statistics. Using experiments, we highlighted the the minimal completeness loss during query reformulation and showed the feasibility of our approach. In summary, quality based exploitation of mappings may yield efficiency gains of up to an order of magnitude. Gathering statistics in the PDMS context, relaxing the assumptions of independent variables and equal distribution of `null` values, and a detailed cost model containing network transfers are major challenges for future work. Moreover, we aim to refine our search strategy for query reformulation in presence of limited resources.

Acknowledgments. We want to thank Stefan Winkler for helpful discussions. This research was supported in part by the German Research Society (DFG grant no. NA 432).

References

1. Aberer, K., Cudré-Mauroux, P., Hauswirth, M.: The Chatty Web: Emergent semantics through gossiping. In: World Wide Web Conf. (WWW). (2003)
2. Halevy, A.Y., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: Conf. on Data Engineering (ICDE). (2003)
3. Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: Symposium on Principles of Database Systems (PODS). (2004)
4. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Information Systems **29** (2004) 583–615
5. Pottinger, R., Levy, A.Y.: A scalable algorithm for answering queries using views. In: Conf. on Very Large Databases (VLDB). (2000)
6. Tatarinov, I., Halevy, A.: Efficient query reformulation in peer data management systems. In: Conf. on Management of Data (SIGMOD). (2004)
7. Heese, R., Herschel, S., Naumann, F., Roth, A.: Self-extending peer data management. In: Conf. Datenbanksysteme in Business, Technologie und Web (BTW), Karlsruhe, Germany (2005)
8. Löser, A., Nejdl, W., Wolpers, M., Siberski, W.: Information integration in schema-based peer-to-peer networks. In: Conf. on Advanced Information Systems Engineering (CAiSE). (2003)
9. Madhavan, J., Bernstein, P.A., Domingos, P., Halevy, A.Y.: Representing and reasoning about mappings between domain models. In: Proc. of the National Conf. on Artificial Intelligence (AAAI). (2002)