System P: Query Answering in PDMS under Limited Resources

Armin Roth Felix Naumann Tobias Hübner Martin Schweigert
Humboldt-Universität zu Berlin
Berlin, Germany
{aroth, naumann, thuebner, martin.schweigert}@informatik.hu-berlin.de

ABSTRACT

Peer data management systems (PDMS) consist of autonomous peers with mappings to other peers. Queries submitted at 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. System P is a full-fledged relational PDMS featuring peers distributed over the network, relational schemata and multiple local sources at each peer, LaV and GaV mappings between peer schemata, fully localized query planning and execution. In addition, it serves as a test bed with automated deployment of schemata and data across the network and an independent monitor peer to perform scalability measurements.

We have observed that the number of mappings and peers that must be traversed to obtain a complete query answer reaches the point of intractability for already relatively small PDMS (50+ peers). We present and evaluate a set of decentral strategies that guide peers in their decision along which further mappings the query should be sent. Thereby we compromise on the completeness of the query result but gain feasibility of query answering in large PDMS.

1. PDMS AND DATA QUALITY

Sharing and integrating relevant information is a pressing problem. The main motivation of information integration is a complete view of the world. This requires consideration of as many relevant and heterogeneous data sources as possible. Centralized data integration systems, e.g., data warehouses or federated DBMS, seek to fulfill these requirements by using an integrated global schema. In practice, it can be observed that a decentralized fashion of data sharing is preferred. Users desire to pose queries to their own schema. Such requirements are addressed by several works on *peer data management systems* (PDMS) [1, 2, 3, 4, 6]. In a PDMS a peer can serve both as data source and as a mediator. Queries are translated and transferred using semantic relationships between peers, so-called mappings, as shown Fig. 2. Examples for application areas of PDMS are partnerships between companies for developing complex technical products, cooperations of scientific institutions, and ad hoc disaster management. PDMS can also serve as a decentralized infrastructure for mediation between ontologies in the semantic web [2, 5, 7].

Query Planning. Queries are posed at any peer in the PDMS and initially answered by locally stored data. Additionally, the query is passed on to other peers via mappings, which may reformulate the query so that it conforms to the other peer's schema. Henceforth, the query is passed through the system and answers are sent back, again via the mappings, to the initial peer. A query answer is complete if there remain no further possible reformulations of the query at any peer.

Several approaches to PDMS query answering have emerged. An important distinction is whether they assume some form of global knowledge, such as the schemata and mappings in [13], or whether each peer functions fully autonomically, such as System P presented here. System P adopts the rule-goal-tree approach of [6], but does not form a global plan that can be optimized. Rather, each peer forms one part of the rule-goal-tree and decides independently how to reformulate the query using mappings to other known peers.

Scalability. In implementing System P we were able to confirm our original suspicion: Maintenance, query planning, and query execution in PDMS does not scale well beyond 50 peers, depending on the number of mappings in the system. We address all three problems by eliminating the need of global knowledge (see paragraph above) and by compromising on the completeness of query results.

In PDMS with very many sources a complete result cannot be expected anyway, be it due to peer dynamics, network failures, application requirements, etc. To achieve constraints on network load and/or response time, we present several pruning strategies that bound the space of all query plans. The strategies are implemented locally by each peer, and usually limit the set of mappings along which to pass a query. The goal of System P is to achieve optimal completeness within given cost/time constraints. Our completeness model estimates both the result cardinality and the richness of the result, i.e., the number of non-null values (Sec. 3.2). The query planning strategies we propose exploit this completeness model for their decisions.

Related work. Upon close examination of related work we believe to present the first full-fledged PDMS comprising optimization techniques to trade off benefit and cost during exploring the search space. This means System P is not simulated but works on multiple machines and acutally processes relational data. Of course the concept of PDMS and simulations have been presented before: Most prominently *Piazza* guided our design and query semantics [6]. Concessions toward completeness of query results are mentioned, but not discussed in detail. Although the system includes pruning techniques [13], these are 'safe' and thus

WWW2006, May 22–26, 2006, Edinburgh, UK.
.

do not principally solve the scalability problem. Also they involve non-local coordination between peers, whereas our strategies are strictly local to autonomous peers.

The *Semantic Gossiping* approach 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 coDB system [4] is quite similar to System P, but includes no optimization approaches. As the peers in the SomeWhere system [2] reason over theories consisting of propositional clauses, it is not directy comparable to other Pdms using database techniques.

For brevity, other relevant Pdms projects, such as Edutella [8], Hyperion [10], or the work by Calvanese et al. [3], are not discussed here. Please see [11] for a more elaborate discussion. Also, related work on the completeness model (Sec. 3.2) has been previously discussed in [9].

Contributions. This paper presents System P, an up-and-running Pdms, and its features to address the problem of scalability. Section 2 presents details of the system architecture, its mapping and query processing capabilities. Furthermore it describes our test bed, which allows a parameterized generation of many different schemas and their extensions, and which allows their deployment on Pdms graphs of different shapes. Section 3 then gives an overview of different pruning strategies to achieve high scalability. We include some experimental results showing that high completeness can already be achieved at low cost. Finally, Section 5 concludes and gives an outlook on future plans, such as a refined result cardinality estimation.

2. SYSTEM P

To investigate our query reformulation and pruning strategies experimentally, we have developed a Java-based Pdms implementation called System P. This system is based on the JXTA framework (`www.jxta.org`), which supports communication between distributed peers or nodes across the Internet. System P is fully functional; initial tests have used up to 50 peers across a network and have answered queries against data distributed across all peers. Further information, a recorded demo, and a WebStart enabled version can be found at `www.informatik.hu-berlin.de/mac/SystemP`.

2.1 Features

As shown in Fig. 1, peers in System P consist of a relational peer schema, a set of local sources connected to the peer schema via local mappings, and peer mappings from one peer schema to the schemas of other peers. Mappings and queries are formulated as datalog rules.

In System P, any existing RDBMS with a JDBC driver can be embedded as a local source. If no DBMS is available, System P uses an internal HSQLDB database for storing temporary data and to perform joins as well as unions.

Query planning is fully decentralized and implemented locally at the peers. Based on the given local-as-view (LaV) and global-as-view (GaV) peer mappings, a local rule-goal tree is created at the peer receiving the original query as well as at every peer that is contacted during query processing. After local optimization and possibly pruning, the resulting local query plan is executed. This process requires no global knowledge at all; query planning, optimization, and pruning strategies are based on local peer information only.

For experimental purposes System P features a specialized monitor peer to which the regular peers report their activities. This peer can be used to create, visualize, manipulate, and query a Pdms. During query execution this peer can perform a live-visualization of the mapping paths already used for query reformulation and passing back results. The monitor peer is also able to collect numerous metadata that can be used for query execution analysis and statistics, for instance execution time, number of database operations, number of peer queries, number of peers used and local mappings, density of result set, global query plan (composed of local query plans).

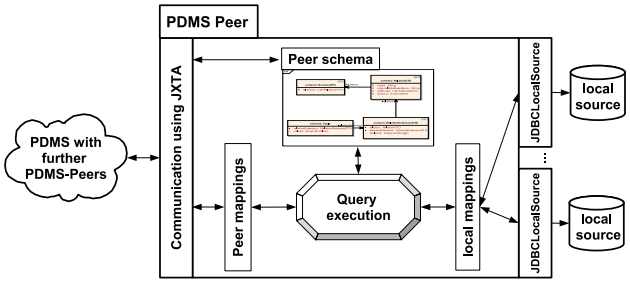

Figure 1: Components of a System P peer.

2.2 PDMS instance generation

To perform meaningful experiments, it is important to create different kinds of real-world Pdms instances and execute queries on them. To simplify this step, System P includes a Pdms generator. Based on a given reference schema this generator creates a given number of peers, local sources, various heterogeneous schemas, mappings including comparison predicates for the created peer graph (nodes = peers, edges = set of peer mappings). As the relationships between the reference schema and the derived schemas are known, the generator can create appropriate mappings between the derived schemas. Additionally, instances of local sources are created, again using data given with the reference schema or by generating new data values. All steps are highly parameterized to enable experiments on a large variety of situations. Finally, these peers are assigned to the differently shaped Pdms graphs, such as simple chains, circles, trees, and completely random graphs, and then distributed across the network to the different peers.

In summary, System P is a fully functional, non-simulated relational peer data management system with a test bed environment to generate peers, distribute them, query them, and collect query execution statistics.

3. PRUNING STRATEGIES

Typically, query plans in Pdms become surprisingly large even for relatively small Pdms with tens of peers. For instance, a PDMS of 31 peers each with mappings to a maximum of five other peers, the query plan for a single relation query had a depth of 21 and submitted almost 70,000 subqueries along mappings between distributed peers. Therefore, computing all certain answers in web-scale Pdms is not feasible. To meet this scalability problem, we exploit the influence of mappings on the query results to decide which mapping paths are not worth following. Our approach tries to identify mapping paths that preserve potential completeness of the intermediate query results "behind" these mappings.

We use the completeness information and its calculation to prune away subplans that are not promising. In [11] we presented several straightforward strategies, which simply pruned mappings with an expected completeness below a certain threshold. However, that approach did not guarantee executions within a given resource constraint and in extreme cases it may prune every result available in the PDMS.

In this approach, the benefit and cost of query answering are solely dependent on the threshold for the completeness measure of the peer mappings. As a result, pruning too less mappings may lead to inacceptable response time.

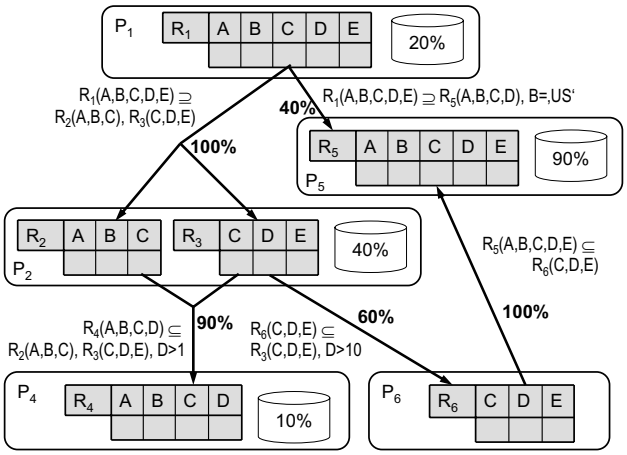

Figure 2: A small PDMS with mappings, source coverages, and filter factors.

To overcome these restrictions, this paper shows how to spend a given budget to maximize the benefit of query answers while guaranteeing a limited amount of work.

To this end and in the spirit of the Mariposa system [12], we propose a budget-driven approach, where peers are assigned a budget to use for query answering. Every peer is free to decide about how to spend its budget and in which way to refund unspent budget to other peers. Intuitively, this exchange of budget is a means to establish a weak coordination between peers. In this paper, we examine different fully local strategies to spend and possibly refund budget. The strategies differ in the way peers allot budget to their outgoing mappings.

The main idea behind our budget spending strategies is to distribute the budget to several mappings based on their expected loss of information. Mappings are lossy if they include projections and/or selections. The impact of these selections and projections on the completeness of data returned by the mappings is highlighted in the following example and in Sec. 3.2.

3.1 Illustrative Example

To illustrate our approach, we introduce an example of a simple PDMS with some schemas and mappings between them, and guide the reader through the process of query answering while considering completeness (Fig. 2).

Each inclusion mapping [6] is shown in the usual Datalog notation. The mappings can be incomplete, for instance the mapping $P_1 \rightarrow P_5$ does not map attribute E. Also, mappings may have predicates, whose effect as filter factors are expressed as percentages. For instance, the mapping $P_2 \rightarrow P_6$ has a predicate (D > 10), which removes all but 60% of the tuples. In reality of course such filter factors depend upon the actual data sets passed through the mapping, as discussed in Sec. 5.

We now regard query planning with a simple cost model and our completeness model, discussed in detail in Sec. 3.2. In our cost model, accessing a peer has a cost of 1. The fixed budget assigned along with the initial query serves to bound the resources spent for query planning and evaluation. To ensure that we spend the budget wisely, i.e., that we retrieve as many results as possible, we employ a simple completeness model. It counts the number of retrieved attribute values in relation to the overall number of attribute values. Please note, that it is not necessary to know the latter, because it only acts as a normalizing factor.

Assume that the entire PDMS holds data about 100 items in the five attributes A, B, C, D, E. For instance, peer P_5 stores 90 tuples, each with data across all five attributes. Thus, P_5 has a completeness of 90%. However, if this data is passed through the mapping to P_1, the number of tuples is reduced to 36 (40% of 90 tuples) tuples and the number of attributes is reduced to four. Thus, P_5 has a completeness of approx. 29% (144/500) from the perspective of P_1. This effect of decreased completeness is accumulated along mapping paths.

Consider a query Q at Peer P_1 asking for all objects of relation R_1. Let the budget of this query be 5, i.e., we can access five peers to answer the query. The database at P_1 can answer the query itself with a completeness of 20%. P_1 has mappings to two other peers, P_2 and P_5, and it must now decide, which paths to follow and how much budget to allot to each path.

Using one of the strategies outlined later, P_1 decides to pass along the query only to P_2. Since we assume full autonomy of peers, we employ a fully local optimization strategy; if P_1 had global knowledge about the benefit of other peers, it might have chosen the path to P_5, because it promises more tuples, despite the selective filter on the mapping. Albeit, in a dynamic PDMS scenario with very many peers, one cannot assume to have such knowledge.

The aggregated completeness is now incremented to include the data at P_2: $20\% + 40\% - (20\% \cdot 40\%) = 52\%$. The subtraction accounts for duplicates among P_1 and P_2, assuming for now independence of tuples stored in P_1 and P_2 (Sec. 3.2). Next, P_2 has a remaining budget of 3, decides to spend it all for the path to P_4, and forwards the query. P_4 increments completeness to 55.45%, accounting for the fact that it supplies only four of the five attributes and only 10% of the tuples. Because there are no further mappings to follow, P_4 refunds the remaining budget of 2 to P_2, which in turn spends it on the originally ignored path to P_6.

As P_6 lacks a data source of its own, it cannot contribute to the overall completeness and passes the query to P_5. Thus, the large data set of P_5 is reached after all, however, along a different path. This alternative path conserves more data on the way back to P_1. The final result after all peers answer the query and send the results back along the path of mappings has a completeness of 67.68%.

While this simple example is meant to convey the main idea of the paper, real PDMS must deal with additional difficulties that we do not address in this paper but discuss as future work in Sec. 5.

3.2 Completeness Model

Building upon previous work we model completeness in two dimensions: coverage and density (see [11] and [9] for more details). Briefly, *coverage* $c(\mathcal{D})$ describes the proportion of the size of a tuple set $\mathcal{D}$ 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. For a query result it is based on the overall number of tuples that fulfill the query predicate, i.e., it is query-dependent.

Density on the other hand describes the number of attribute values for each result tuple in relation to the attributes of the query. 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 low-density tuples. Similar to coverage, the density $d(\mathcal{D})$ of a data set is also query-dependent.

Finally, overall *completeness* can be regarded as an aggregated measure for the ratio of the amount of data in a certain data set, e.g., the result set of a query, to the amount of data in the whole PDMS. In [9] 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 $0 \leq C \leq 1$.

As described in Sec. 2, a peer creates a local query plan, which is determined using all mappings usable for reformulating the query at hand. We are interested in calculating the influence of peers "behind" a certain mapping on the completeness of this local query plan. To this end, we briefly recall the main aspects of our completeness model [11, 9]. We assume that queries and mappings are select-project-join (SPJ) queries. Additionally, query plans contain union-type operators, which collect results returned by alternative mapping paths starting from a certain peer.

Applying a ***selection*** to a tuple set accessible through a mapping reduces the set of tuples by a mapping selectivity factor s. Assuming independency of the attributes occuring in the selection predicates of the form $x > c$ with a variable x and a constant c, s can be calculated as the product of all mapping selectivites $s(x)$. We assume that these mapping selectivities $s(x)$ are known. It is subject to future research to provide robust selectivity estimation techniques for our context.

Assuming that `null`-values are distributed equally over all tuples, density is not affected by a selection. For simplicity we assume that a ***projection*** in a mapping leaves the number of tuples unchanged but reduces the number of non-`null`-attribute-values as shown in our example in Sec. 3.1.

To calculate completeness for the result of the ***join*** $R_1 \bowtie R_2$ we assume *independence* of the representation of objects, which means that there is no knowledge about extensional overlap of R_1 and R_2. Then, we can draw formulas for the expected coverage and density of $R_1 \bowtie R_2$ from [9]. The issue of independence and of possibly known overlap and join selectivities is addressed in [9]. The answer to a query posed to a PDMS is usually made up of the ***union*** of many contributions from alternative mapping paths. Since these contributions may comprise different sets of attributes, we employ the full outerjoin-merge operator $\sqcup$ from [9] along with formulas to calculate coverage and density of $R_1 \sqcup R_2$.

Using the above completeness model, we can calculate the contribution of all peers "behind" a certain mapping to a peer's query result as follows. The coverage and density scores of the potential result returned by the mapping are unknown at query planning time. However, it suffices to assume them being 1 always, because we only need to compare alternative mappings. The contribution of a certain mapping is determined by calculating the completeness of the local query plan (1) with and (2) without considering that mapping. Then the contribution of that mapping is the difference in completeness between the results of (1) and (2). Intuitively, this algorithm yields the impact of the information loss of a mapping on the query result of the peer. Based on this comparison, we now introduce several strategies to distribute the peer's budget to alternative mappings.

3.3 Budget Spending Strategies

Both of the following budget spending strategies allot the same fraction of a peer's budget to each subgoal of a conjunctive query. They differ in how they distribute the budget of a query subgoal to alternative mappings covering a certain subgoal.

Strategy 1: Weighted. In this strategy, a peer considers the weights of the potential completeness contributions of several outgoing mappings in a breadth-first manner. The peer distributes the budget in inverse proportion to the information loss of alternative mappings. This strategy prefers to explore the neighborhood of a peer rather than more remote peers and thus results in shorter mapping paths with less likely loss of information (and higher semantic relevancy). As a further advantage, this approach enables parallel usage of alternative mappings.

However, in cases where a peer faces to a high fraction of mappings with information loss, it is forced to spend budget on poor mappings. Thus, this approach should be used in conjunction with a threshold-based pruning strategy [11]. Additionally, it has to be considered that mappings resulting in local-as-view style reformulations [6] cover several relations at a certain peer. Tuples retrieved over such mappings promise to contribute more completeness to the overall result than mappings that substitute only a single relation in a global-as-view fashion.

Strategy 2: Greedy. To maximally exploit mappings which are supposed to return the maximal amount of data, this strategy assigns the entire budget of a query subgoal to the mapping with the lowest potential loss of information. In effect, this leads to a greedy, depth-first traversal of the search space along mapping paths preserving comparably much information.

This strategy promises to return the most amount of data for very small budgets compared to the size of the search space. A disadvantage of this approach is that the PDMS might not be explored equally, because the different recursive reformulation paths operate independently at different peers. However, this does not concern the equal assignment of budget to the subgoals of a query mentioned above. Observe that the amount of data a certain mapping will return on one hand depends on the selection predicates in the query. Second, it can vary with the size of the budget it is assigned for further exploration of the search space. It is subject to further research to investigate and consider the latter issue.

Clearly, the effectiveness of these two strategies strongly depends on the ratio between the amount of budget available at all and the size of the search space. Additionally, the *rank* of a PDMS (the average number of peer mappings of a

peer) is an important factor. The more interconnected the peers, the more likely it is that there are alternative mapping paths, which in turn may reach peers that otherwise would not have been found. Most notably, the fraction and distribution of loss of information in the peer mappings strongly influences the effectiveness of our strategies. The more information loss is encountered, the better the strategies work. On the other hand, if there would be no information loss, the Weighted strategy would behave equally as the expansion of all mappings. We provide first experiments on both strategies in Sec. 4, but first turn to two variations.

3.4 Budget Refunding

Variation 1: Altruistic Greedy and Altruistic Weighted. Both the Greedy and the Weighted strategies are performed strictly locally at the peers. However, under limited resources it may be better to allow some cooperation between peers to intelligently spend a limited budget: In this altruistic variation a peer can refund budget back to where it came from if it has no more (promising) mappings to follow. Peers getting back budget either can spend it to mappings they have ignored initially, as it was shown in our example in Sec. 3.1, or can in turn refund the budget if they already have used each of a set of alternative mappings. In effect, budget may be refunded back over several peers.

The altruistic variation can be combined with both strategies. It works well for the Greedy strategy, because along a path back to the peer the query originates from there usually are many mappings not expanded yet. So they might use the refunded budget. In contrast, using the strategy Weighted almost leads to expanded query subgoals along a path to the origin of a query.

The only opportunities to use refunded budget in this case are mappings pruned earlier. This means that refunded budget is either used for mappings with information loss or is not used at all, i.e., due to recursive refunding the budget may return to the peer to which the inital query was posed to. In the latter case, it may be reasonable to allot unused budget a second time to mappings which have contradicted their completeness assumption by returning more data than expected. We leave this aspect to further research.

Variation 2: Deferred Weighted. This strategy provides a way to decide about allotting budget based on (slightly) more global knowledge. This can be accomplished by defering the decision about spending budget to the next mapping until it is known how much budget the current mapping will refund. On the one hand, this would take more time, because answering of a single query can not run in parallel. However, this strategy guarantees that the entire budget is used up even for the strategy Weighted. In this paper, we do not examine this variation experimentally. Please note that the Greedy strategy is inherently a deferring strategy.

4. EXPERIMENTS

In this section we present only initial experiments with System P. In particular we present results regarding the completeness of query results under our budget approaches; experiments on differently parameterized PDMS instances are yet underway.

4.1 Experimental Setup

Neither the number of peers, nor the rank as the average number of mappings per peer are sufficient to characterize the complexity of query answering in PDMS, because queries may not use all peers and mappings. Therefore, in this paper we use the number of query reformulation steps as a cost measure. The benefit is given by the expected completeness of the resulting query plans determined using the formulas from [9].

We refer to real-world schema BioSQL[1] from the life sciences domain. It was varied by normalization and denormalization and the resulting schemas were assigned to peers of random graphs. We assume a scenario, where different research groups provide peers with protein information along with mutation experiments. Different sets of species covered by the peers and varying publication dates of the information result in selection predicates in the peer mappings. As some peers do not publish certain attributes of their experiments, the peer mappings may contain projections as well.

We experimented with PDMS configurations consisting of up to four different subsets of the BioSQL schema. The less data is distributed in the PDMS, the higher the effort is to find all certain answers. For this reason, in our experiments every peer covers only 5% of the size of the world. The datasets used are listed in Tab. 1.

	#Peers	*rank*	Mappings with loss
$\mathcal{P}_1$	21	1.8	35%
$\mathcal{P}_2$	50	1.3	20%

Table 1: Datasets and their main characteristics.

4.2 Measurements

The effects of the strategies Greedy and Weighted are shown in Fig. 3. There, they are compared to the full expansion ("without pruning") using all possible peer mappings. Note that the overall cost of the full expansion in the PDMS $\mathcal{P}_2$ in Fig. 3 is at about 5500. I.e., a completeness of 1 is achieved at a point well beyond the scale of the image.

As can be seen, our strategies achieve similar results as the full expansion while guaranteeing that the cost are within a given budget. Moreover, we observed that for reasonable sizes of the initial budget compared in relation to the search space, Greedy and Weighted retrieve nearly the same amount of data as the full expansion, independently of the budget: At less than 10% of the cost of full expansion, Greedy already yields nearly 100% completeness and Weighted reaches about 85%. Observe that for the Weighted strategy a considerable part (40%) of the budget is not used. This is due to the problems with budget refunding in this strategy discussed in Sec. 3.4.

To compare the different approaches we determine the ratio completeness/cost as an efficiency measure. In the PDMS configurations above, our budget-driven strategies increase efficiency from several factors up to more than an order of magnitude compared to the computation of all certain answers without pruning (see Fig. 4).

In summary, our experiments clearly show the feasibility of our strategies for query planning under a limited budget. Especially for large PDMS they achieve nearly the same completeness as computing *all* certain answers but with drastically reduced cost.

[1] `obda.open-bio.org`

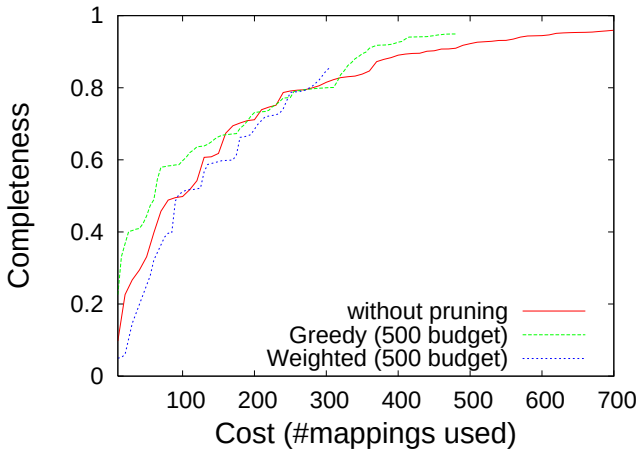

Figure 3: Results of PDMS $\mathcal{P}_2$ using the strategies Greedy and Weighted. Note that the efficiency of the expansion without pruning refers to an unlimited budget.

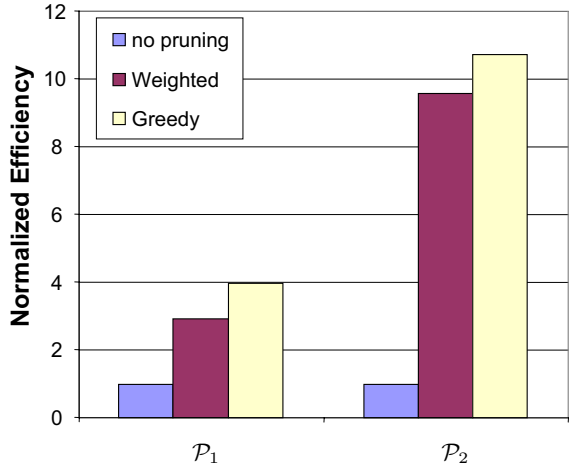

Figure 4: Normalized efficiency (ratio completeness to cost at a budget of 100 and 500 respectively).

5. 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 bound the cost of query answering while still providing enough benefit for the user.

We introduced our full-fledged PDMS System P, which implements a novel, fully decentral approach to maximize completeness of query answers under a cost limit. The main contribution were two strategies for spending a limited budget along with variations concerning budget refunding. Our experiments showed that these strategies yield satisfying completeness even with quite small budgets. In summary, our budget-driven query reformulation approach increases efficiency by up to an order of magnitude.

In future work we aim to gather and maintain statistics about the completeness "behind" peer mappings and apply them to estimate query-dependent selectivity. Second, we plan a more detailed cost model to improve the results of our strategies by considering benefit-to-cost ratios. Finally, completeness is only one of many possible information quality dimensions in the context of PDMS—others promise to be useful as well.

Acknowledgments. We thank S. Trissl and U. Leser for their support with the life science data. This research was supported in part by the German Research Society (DFG grant no. NA 432).

6. REFERENCES

[1] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. The Chatty Web: Emergent semantics through gossiping. In *Proc. of the Int. World Wide Web Conf. (WWW)*, 2003.

[2] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research*, 25:269–314, 2006.

[3] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the Symposium on Principles of Database Systems (PODS)*, 2004.

[4] E. Franconi, G. M. Kuper, A. Lopatenko, and I. Zaihrayeu. Queries and updates in the codb peer to peer database system. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, 2004.

[5] Alon Y. Halevy, Zachary Ives, Peter Mork, and Igor Tatarinov. Piazza: Data management infrastructure for semantic web applications. In *Proc. of the Int. World Wide Web Conf. (WWW)*, 2003.

[6] Alon Y. Halevy, Zachary Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proc. of the Int. Conf. on Data Engineering (ICDE)*, 2003.

[7] Ralf Heese, Sven Herschel, Felix Naumann, and Armin Roth. Self-extending peer data management. In *Proc. of the Conf. Datenbanksysteme in Business, Technologie und Web (BTW)*, Karlsruhe, Germany, 2005.

[8] Alexander Löser, Wolfgang Nejdl, Martin Wolpers, and Wolf Siberski. Information integration in schema-based peer-to-peer networks. In *Proc. of the Conf. on Advanced Information Systems Engineering (CAiSE)*, 2003.

[9] Felix Naumann, Johann-Christoph Freytag, and Ulf Leser. Completeness of integrated information sources. *Information Systems*, 29(7):583–615, 2004.

[10] P. Rodríguez-Gianolli, M. Garzetti, L. Jiang, A. Kementsietsidis, I. Kiringa, M. Masud, R. J. Miller, and J. Mylopoulos. Data sharing in the hyperion peer database system. In *Proc. of the Int. Conf. on Very Large Databases (VLDB)*, 2005. Demonstration.

[11] Armin Roth and Felix Naumann. Benefit and cost of query answering in PDMS. In *Proc. of the Int. Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P)*, 2005.

[12] M. Stonebraker, P. M. Aoki, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu. Mariposa: A wide-area distributed database system. In *VLDB Journal*, volume 5, pages 48–63, 1996.

[13] Igor Tatarinov and Alon Y. Halevy. Efficient query reformulation in peer data management systems. In *Proc. of the ACM Int. Conf. on Management of Data (SIGMOD)*, 2004.