Using Knowledge of Redundancy for Query Optimization in Mediators

Vasilis Vassalos*
Stanford University
vassalos@cs.stanford.edu

Yannis Papakonstantinou
University of California, San Diego
yannis@cs.ucsd.edu

1 Introduction

Many autonomous and heterogeneous information sources are becoming increasingly available to users through the Internet, especially through the World Wide Web. In order to make the information available in a consolidated, uniform, and efficient manner, it is necessary to integrate the different information sources. The integration of Internet sources poses several challenges that have not been sufficiently addressed by work on the integration of corporate databases residing on an Intranet [LMR90]. We believe that the most important ones are heterogeneity, large number of sources, redundancy, availability, source autonomy, and diverse access methods and querying interfaces.

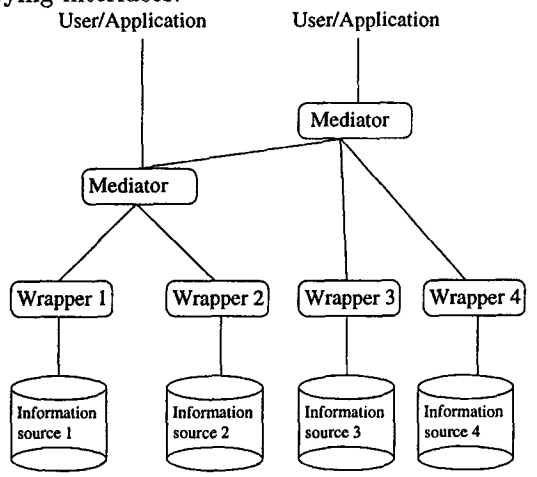

Figure 1: Common mediation architecture

With the exception of the challenges posed by information redundancy and availability, the above mentioned challenges have been significantly addressed by a number of recent mediator systems [GM+97a, LYV+98, LRO96, TRV95, S+, AKH93]. These systems have adopted a *mediation* [Wie92] architecture as shown in Figure 1 to address the heterogeneity and autonomy issues. The main component of these systems is the mediator, which is essentially a query planner.

The recent advances in query planning and optimization have not addressed the challenges and opportunities introduced by the presence of redundant and overlapping information in sources, a situation very common on the World Wide Web. First, the system must avoid retrieving the same set of data from multiple sites. Second, redundancy can be exploited to further increase query performance and system availability by having the system collect information from the "cheapest" available sources.

Using knowledge of redundancy we can reduce the number of source accesses that have to be performed to retrieve the answer to the user query. In Section 2, we discuss this problem in more detail and formulate it as a scheduling problem with AND/OR precedence constraints.

Partial Answers

The amount of information available online often motivates users to ask for (and be satisfied with) *partial answers* to their queries. For example, a user of an integrated bibliographic system asking about books published in 1997 on the Middle East will be more interested in receiving information about most of them quickly rather than waiting for a long time and spending computational resources to get the full set of answers. Having probabilistic information about source overlap can help derive efficient query plans. However there are some challenges that need to be addressed:

1. The amount of information necessary to completely specify source overlaps is exponential in the number of sources.

2. The naive algorithm that uses the source overlap information and chooses the best sources to give

*Research partially supported by NSF grant IRI-96-31952, Air Force contract F33615-93-1-1339 and the L. Voudouri Foundation.

the required partial answer is also exponential in the number of sources.

In Section 3 we formalize the problem, describe the optimization framework and propose approximations that make efficient use of the source overlap information to provide suboptimal solutions.

1.1 Related Work

Most theoretical work in the area of information integration (e.g., [LMSS95, LRU96, RSU95, DL97, DG97, PGMU96, VP97, VP, KW96]) has been in query processing and query planning. In particular, the generation of sound and complete plans for queries over multiple sources has been studied for a variety of data models and query languages.

There has been work on using local completeness information [FW97, Lev96] to create query plans that guarantee getting complete answers to a query while also identifying irrelevant sources. Ashish et al. in [AKL97] discuss a method for discriminating between useful and useless sources during query execution, by adding "sensing" subqueries to the query plan.

Using probabilistic information about overlap to help in query planning has recently been proposed in [FKL97], where the goal is to pick the k most useful[1] sources to access. In the framework described in [FKL97], the system knows and uses the overlap of the *domains* of the source information and not the overlap of the source contents themselves. That makes the problem more tractable; there is an exponential blowup when using directly source overlap information.

2 Minimizing Source Accesses

The query planning algorithms employed by many mediator systems (see Figure 1), such as the Information Manifold [LRO96] or TSIMMIS [GM+97a, LYV+98], first decompose the user query into plans for querying the sources. We define a *query plan* in this context to be a set of source queries to be executed, followed by relational operations (select, project, join) to be applied in the mediator to the results of the source queries. By executing these plans, the mediator collects every relevant piece of information. Note that the same piece of information could be collected from more than one source. The results of executing these source query plans are then integrated in the mediator.[2]

In the presence of redundancy in the data, or in the case of unavailable sources, such a strategy can be very inefficient or even infeasible. If it is possible to determine that certain query plans will retrieve the same information, it is more efficient to execute only the ones that are necessary.

It is possible to determine equivalence for query plans if some sources are complete in the information they provide. We discuss this scenario further in the next subsection. Moreover, if we can encode replication information in some finer granularity, using for example constraints that state that parts of sources are equivalent, then the mediator can use that information to infer that some query plans it generates are in fact equivalent or contained in one another. In general, the problems of encoding redundancy information in a mediator and of using the encoding to make inferences about source queries are very interesting and challenging problems that we will not be addressing in this paper. We will focus on what needs to be done after such inferences have been made. A set of query plans has already been divided into equivalence classes; at least one query plan from each class needs to be executed to obtain a complete[3] answer to the user query. We want to pick these *representatives* from each class in a way that minimizes the total cost of answering the user query.

In the next subsection we present an integration framework where the optimization issue under discussion arises naturally. Sections 2.2 and 2.3 describe in more detail the optimization framework, express this query planning problem as a scheduling problem with AND/OR precedence constraints and discuss its solution.

2.1 Information Integration Using Logical Views

In the common integration architecture shown in Figure 1, a *mediator* is responsible for going to the sources and finding answers to the user query. Let us assume a mediator that is modeled after the Information Manifold [LRO96]. The mediator (and the user) knows about a collection of global predicates, in terms of which all queries are posed. A *wrapper* describes the contents of its underlying source via one or more views [Ull97]. To make the scenario concrete, let us assume that the views are expressed as conjunctive queries over the global predicates [Ull88].

Let us also assume (in contrast to the Information Manifold) that sources are complete over their contents: a source that contains information on Japanese movies after 1960 really contains all of these movies. Therefore, a view `jap_mov(T,Y)` defined as

```
jap_mov(T,Y) :- movies(T,Y,'japan') & Y>1960
```

[1] Useful in terms of amount of provided relevant information.

[2] In the simplest case, the mediator just takes the union of the results provided by the different plans.

[3] As complete as possible using the available sources.

means that jap_mov contains *all* the title/year pairs for movies from Japan, made after 1960.

The mediator tries to answer a user query (over the global predicates) by finding all the minimal *solutions*, i.e., all minimal conjunctive queries over the views that are *contained* in the user query and are not contained in each other[4]. The number of potential minimal solutions is exponential in the size of the user query and polynomial in the number of views available [Ull97]. As mentioned earlier, each view models part of the contents of an information source. Each subgoal of a solution is evaluated by the mediator by sending a query to the corresponding wrapper asking for the view contents.

Example 1

Our example integration system mediates between movie sources and follows the architecture of Figure 1 and Section 2.1. The global predicates the system knows about are

- `movies(T,D,L)`, giving the title, director and lead actor/actress of a movie.
- `review(T,Re)`, giving the title of a movie and a review for it.
- `rating(T,Ra)`, giving the title of a movie and its rating.
- `prod_info(T,D,C,S,Y,Cy)` giving the title, director, year, cinematographer, scriptwriter, year and country of a movie.

The system integrates 4 movie databases:

1. Source S_1 knows all movie titles, directors, lead actors and year for movies made after 1988. It is described by the following view:

```
mov(T,D,L,Y) :- prod_info(T,D,C,S,Y,Cy)
                & movies(T,D,L) & Y>1988
```

2. Source S_2 knows all movies made by Wim Wenders:

```
wen(T,D,L,C,S,Y) :- prod_info(T,D,C,S,Y,Cy)
                    & movies(T,D,L)
                    & D='wenders'
```

3. Source S_3 gives reviews of rated movies made in the 1990's:

```
rev(T,Re) :- review(T,Re) & rating(T,Ra)
             & Y>=1990
```

4. Source S_4 gives information (including reviews) about German movies made in the 1990's:

```
all(T,D,L,Re) :- prod_info(T,D,C,S,Y,Cy)
                 & movies(T,D,L)
                 & review(T,Re) & Y>=1990
                 & rating(T,Ra)
                 & Cy='germany'
```

Assume the user wants to see reviews for films made by Wim Wenders in Germany:

```
ans(T,Re) :- prod_info(T,D,C,S,Y,'germany')
             & movies(T,D,L) & review(T,Re)
             & rating(T,Ra) & D='wenders'
```

There are 3 ways to get a minimal solution to this query: Combining sources S_1, S_3, combining sources S_2, S_3 or using S_4. If each source is complete over its contents, then all 3 solutions are equivalent to the following query:

```
ans(T,Re) :- prod_info(T,D,C,S,Y,'germany')
             & movies(T,D,L) & D='wenders'
             & review(T,Re) & rating(T,Ra)
             & Y>=1990
```

The system should be able to decide, based on a cost function, which solution to pick.

2.2 Framework and Cost Model

Let $\mathcal{V} = V_1, \ldots, V_n$ be a set of logical views, each one describing the contents of an information source, per Section 2.1. Given a user query Q, let $\mathcal{P} = P_1, \ldots, P_m$ be the set of *solutions* that the mediator comes up with, as described in Section 2.1. Let $\mathcal{V}^{P_i} = V_1^{P_i}, \ldots, V_l^{P_i}$ be the set of views that solution P_i needs to access. Finally, let the solutions be divided into k equivalence classes, such that all solutions in each class produce the same part of the answer to Q. We will denote the solutions in class j by $P_{1j}, P_{2j}, \ldots, P_{m_j j}$. Our objective is to pick at least one solution from each class in a way that minimizes the total cost of executing the chosen solutions (and therefore the user query).

We adopt a simple cost function: the cost C_i of accessing a view V_i is constant, and independent of the particular query that is issued to retrieve the view contents.[5] The cost of a solution P is the sum of the costs for *accessing* the views $\mathcal{V}^P$:

$$C_P = \sum_{V_i \in \mathcal{V}^P} C_i$$

[4] Notice that a minimal solution is *maximally* contained in the query, as it returns a maximal subset of the query result.

[5] This cost function implies that the startup source connection costs — money or otherwise — completely dominate the costs associated with computing and transmitting the necessary view contents.

Moreover, we assume that in the course of answering Q each view is accessed at most once, i.e., results of view accesses are cached so if a second solution needs to access the same view, it can reuse the cached copy.[6] Because of this reasonable assumption, and the fact that different solutions access the same views, it is not enough to just pick the cheapest representative from each class. We want to pick $\mathcal{P}_r = R_1, \ldots, R_k$ such that R_i belongs to class i and $\sum_{R_i \in \mathcal{P}_r} C_{P_i}$ is minimized. Notice that this formulation naturally allows us to model unavailable sources, by assigning extremely high access cost to their logical views.

Example 2

Assume a user query Q is answered by the mediator by the following solutions: P_1, accessing views V_1, V_2, V_3; P_2, accessing views V_3, V_4; and P_3, accessing views V_4, V_5. Let us also assume that the cost of accessing each view is equal to 1. Assume we have determined that solutions P_1 and P_2 actually provide the same information, thus they are equivalent. Then it is obvious that the mediator should answer the user query by executing only P_2 and P_3, since this results in fewer view accesses (only 3) and therefore smaller cost than executing P_1 and P_3 (5 view accesses required).

Notice that to make this decision it is not enough to know the cost of executing P_1, P_2 and P_3: $C_{P_1} + C_{P_3}$ overestimates the cost C_{P_1,P_3} of executing both P_1 and P_3, because of our caching assumption.

We can formulate the problem of choosing the solutions in $\mathcal{P}_r$ as an AND/OR scheduling problem with precedence constraints, as in Figure 2. We label each view as a *leaf task*, each solution as an *AND-task* (for obvious reasons), each equivalence class as an *OR-task* and the user query as an *AND-task*. An *AND-task* cannot be scheduled before all its *predecessors* are scheduled, while an *OR-task* cannot be scheduled before at least one of its predecessors is scheduled. Our goal is to schedule the query node and the optimization criterion is to minimize the number of *leaf-tasks* scheduled. It should be obvious that this is just a simplification of our original problem, where each view V_i has cost $C_i = 1$. Our instance of AND/OR scheduling has *internal-tree* constraints: there are no cycles and all non-leaf nodes have at most one incoming edge.[7]

[6] It is interesting of course to also consider more detailed cost models.

[7] Notice that if an internal AND-node has two incoming edges from two OR-nodes, that means that the corresponding view query belongs to two equivalence classes that need to be collapsed because of transitivity; thus the two OR-nodes will become one, and the constraint will be satisfied.

2.3 Known Results and Open Problems

The AND/OR scheduling problem is in general NP-complete [GM97b]. The more interesting question is whether there are good polynomial approximation algorithms for this problem. Our instance of the problem has two AND-levels and one OR-level.

- If there are only one AND-level and one OR-level, the problem becomes set cover, so it is polynomially approximable to no better than a $\log n$ factor [Hoc97].
- If the graph has two AND-levels and two OR-levels (alternating) and *internal-tree* constraints, then the problem is not polynomially approximable to even a linear factor [GM97b].

The query planning problem under consideration is open; we are currently working on showing it is as hard as AND/OR scheduling with two AND- and two OR- levels (alternating). If this is the case, it is essential to evaluate any proposed heuristics in real-life query planning scenarios, to determine their behaviour in real situations.

3 Probablistic Source Overlap: Optimizing for Partial Answers

Internet sources very often do not provide complete information. For example, among sources for automobile classified ads, it is improbable that any source contains all ads for Hondas. It is also not always possible to provide precise logical constraints on heterogeneous sources. Thus we may not be able to decide that any query plans are equivalent to each other. But we may have information about how much overlapping information the sources provide.

Continuing the previous example, we may know that only sources A, B contain information about Honda ads, and source A has 90% of all Honda ads that source B has. There may be Honda ads of B that do not appear in A and vice versa. Even though plans involving source A might not be equivalent to plans involving source B, the mediator should be able to use such information to obtain more efficient query plans for *partial answer* queries, i.e., queries that are satisfied with a percentage of the possible answers. In particular, the mediator should deliver the requested part of the answer and at the same time avoid retrieving overlapping information, hence reducing the computational cost (and possibly the financial cost as well.) Note that optimizing for parts of the answer serves perfectly users who browse. In the above example, if the user asks for 80% of ads for Honda cars, the mediator can infer that only source A needs to be accessed in order to get the required part of the answer.

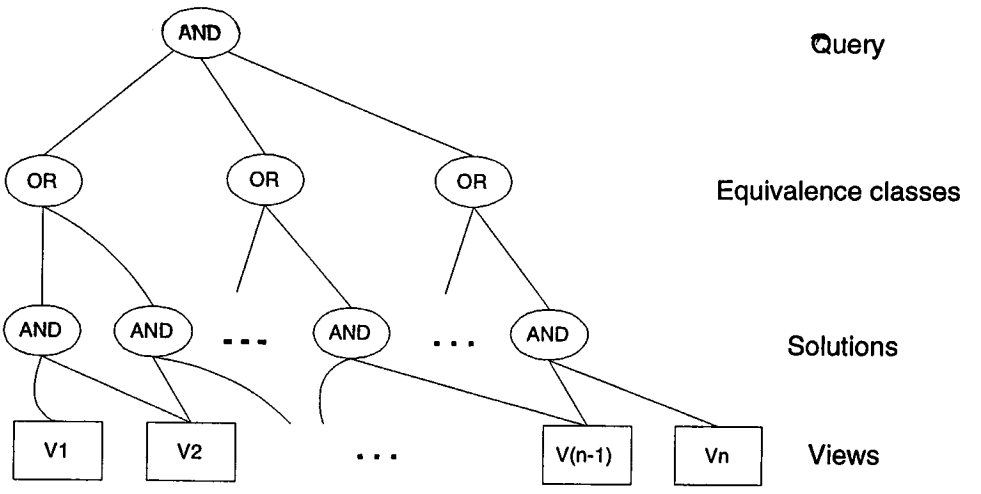

Figure 2: Solution choice as AND/OR scheduling with internal-tree constraints

For simplicity and clarity, we will discuss optimization of partial answer delivery for queries that perform just selections over unions. In the next paragraphs we present the optimization criteria of the mediator and give the requirements for efficient heuristics. The performance of these heuristics will have to be evaluated with many real world experiments.

3.1 Framework and Optimization Criteria

Let us first motivate the challenges by considering (i) n sources $S_i, i = 1, \ldots, n$, each one exporting a relation R_i and (ii) a query $q = \cup_{i=1,\ldots,n} R_i$. The mediator has statistics estimating the *coverage* $p(R_i)$ of each available R_i in the union. For example, $p(R_1) = 0.5$ if R_1 contains half the tuples of $R_1 \cup \ldots \cup R_n$. We will discuss how to obtain these and other statistics later in this section.

We require that the mediator minimizes the total cost for the retrieval of at least a percent of the result, where the percentage a is provided by the user. (It is easy to see that the problem of retrieving the first a tuples is essentially identical.) We adopt a simple cost model, where the cost for answering a query is equal to the sum of the amounts of information that we retrieve from the sources that are accessed by the query.

The optimizer has to choose m relations $R_1, \ldots, R_m$ such that the cost of retrieving the information from them $C = \sum_{j=1,\ldots,m} p(R_j)$ is minimized and the *coverage* of the result provided by the m relations $p(\cup_{j=1,\ldots,m} R_j)$ is greater than $a\%$, that is

$$\begin{cases} \min \sum_{j=1,\ldots,m} p(R_j) \\ p(\cup_{j=1,\ldots,m} R_j) \geq a/100 \end{cases} \quad (1)$$

This problem is complicated by the fact that sources contain overlapping information: the coverage $p(R_i \cup R_j)$ is not $p(R_i \cup R_j) = p(R_i) + p(R_j)$. An obvious way to solve this problem is to compute and store estimates for all possible $p(\cup_{j=i_1,\ldots,i_m} R_j)$. Given any union query and any percentage a we can then choose the right $R_1, \ldots, R_m$ in time linear in the number of sources available (using binary search). There are two serious problems with this solution:

- There is an exponential number of unions of which to estimate the size. In small enough integration scenarios it is not infeasible to do so, since secondary storage is these days available in abundance, and the size information needs to be computed once. In particular, if we are integrating 20 sources, we need to keep a few bytes for each of 2^{20} subsets, or a few megabytes of data. But clearly this solution does not scale.
- Another important problem is that it is not easy to estimate these quantities: sampling methods cannot estimate union sizes directly. Instead we can use sampling to estimate source overlaps.

3.2 Using overlap information

Given a table C of all possible source overlaps $p(\cap_{j=i_1,\ldots,i_k} R_j)$, we can always calculate any $p(\cup_{j=1,\ldots,m} R_j)$ from the entries of C using the inclusion-exclusion formula:

$$p(\cup_{j=1,\ldots,m} R_j) = \sum_j p(R_j) - \sum_{j<k} p(R_j \cap R_k) + \ldots + \\ +(-1)^m p(\cap_{j=1,\ldots,m} R_j)$$

Of course, C still requires exponential space.[8] Moreover, calculating even one $p(\cup_{j=1,\ldots,m} R_j)$ takes exponential time in the number of sources. We discuss the issue of space requirements in Section 3.4. In

[8] A similar observation is made in [FKL97].

the following subsection we briefly discuss the problem of finding efficient heuristics that use overlap information to generate efficient query plans for partial answer queries.

3.3 Algorithm for Partial Answer Query Plans

Even assuming that we have explicitly stored all k-wise overlaps for all $k \leq n$, we still need an efficient, if possible polynomial, algorithm for choosing m relations $R_1, \ldots, R_m$ such that constraints (1) are satisfied. This optimization problem is NP-complete by easy reduction from the exact set cover problem [GJ79] and is a variant of the set cover problem. The set cover problem is only approximable by a polynomial algorithm to a $\log n$ factor [Hoc97], where n is the size of the universe of the sets. A $(\log n)OPT$ polynomial time approximation algorithm for our problem is straightforward. We are currently investigating whether there is a better approximation algorithm for this problem. We are also planning to experimentally evaluate greedy algorithms for solving this problem.

In the next subsection we discuss ways to approximate the statistics necessary for partial answer query planning.

3.4 Statistics Acquisition and Approximation

The mediator will discover overlap statistics by analyzing the results of prior queries. It may also periodically sample the sources to derive statistics that cannot be derived with significant confidence from the results of prior queries. However, given the exponential space required to store all required statistical information, the novel challenge is to approximate the statistics; a possible approximation is to precompute a subset of the entries of C and use the maximum likelihood estimator of the others. The desiderata for this summarization are that it is space efficient and that it allows us to compute the coverage of unions of sources without too much error in either direction — underestimating the coverage of a union means we are taking a hit in efficiency: we will end up computing a larger percentage of the answer, at probably a higher cost. These approximations will have to be evaluated in a practical setting; we are looking for good average performance in real integration scenarios. Lower bounds for the error in the estimation of unions using approximate inclusion-exclusion are proven in [LN90]: for a union of n sets, if we are given all k-wise intersections of these sets for all $k \leq K$, *any* approximation of the union may err by a factor of $\Theta(n/K^2)$ if $K < \sqrt{n}$.[9]

[9] If $K \geq \sqrt{n}$ then we can get a very good approximation, but this case is not very interesting, since we still need overlap information for at least $(\sqrt{2})^n$ source combinations.

The statistics acquisition and approximation problem is complicated by the fact that in practice queries also impose selection conditions, like `type = 'sedan'`, on the source data. Clearly, it is very expensive to generate a separate set of statistics for each possible query. We should only keep statistics for predicates that are significantly different than the (relevant) entries of C.[10] For example, assume that statistics information indicates that sites S_1 and S_2 have a 10% overlap on advertised cars but 60% overlap on advertised sedans. In this case it is necessary to keep the overlap information on sedans because its estimate is very imprecise.

References

[AKH93] Y. Arens, C. A. Knoblock, and C.-N. Hsu. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

[AKL97] N. Ashish, C. A. Knoblock, and A. Levy. Information gathering plans with sensing actions. In *Fourth European Conference on Planning*, 1997.

[DG97] O. Duschka and M. Genesereth. Answering queries using recursive views. In *Proc. PODS Conf.*, 1997.

[DL97] O. Duschka and A. Levy. Recursive plans for information gathering. In *Proc. IJCAI Conf.*, 1997.

[FKL97] D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *Proc. VLDB Conf.*, 1997.

[FW97] M. Friedman and D. S. Weld. Efficiently executing information-gathering plans. In *Proc. IJCAI Conf.*, 1997.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.

[GM+97a] H. Garcia-Molina et al. The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, 8:117–132, 1997.

[GM97b] M. Goldwasser and R. Motwani. Intractability of assembly sequencing: unit disks in the plane. In *Proc. of Workshop for Algorithms and Data Structures*, 1997.

[Hoc97] D. S. Hochbaum. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1997.

[KW96] C. T. Kwok and D. S. Weld. Planning to gather information. In *Proc. AAAI Conf.*, 1996.

[10] Notice that [FKL97] make the simplifying assumption that this is never the case.

[Lev96] A. Levy. Obtaining complete answers from incomplete databases. In *Proc. VLDB Conf.*, 1996.

[LMR90] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22:267–293, 1990.

[LMSS95] A. Levy, A. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. PODS Conf.*, pages 95–104, 1995.

[LN90] N. Linial and N. Nisan. Approximate inclusion-exclusion. In *Proc. STOC Conf.*, 1990.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. VLDB*, pages 251–262, 1996.

[LRU96] A. Levy, A. Rajaraman, and J. Ullman. Answering queries using limited external processors. In *Proc. PODS Conf.*, pages 227–37, 1996.

[LYV+98] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, and Y. Papakonstantinou. Capability based mediation in TSIMMIS. In *Proc. SIGMOD Conf.*, 1998.

[PGMU96] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specifications. In *Proc. ICDE Conf.*, pages 132–41, 1996.

[RSU95] A. Rajaraman, Y. Sagiv, and J. Ullman. Answering queries using templates with binding patterns. In *Proc. PODS Conf.*, pages 105–112, 1995.

[S+] V.S. Subrahmanian et al. HERMES: A heterogeneous reasoning and mediator system. Available from www.cs.umd.edu/projects/hermes/.

[TRV95] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of DISCO. Technical report, INRIA, 1995.

[Ull88] J.D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. I: Classical Database Systems.* Computer Science Press, New York, NY, 1988.

[Ull97] J.D. Ullman. Information integration using logical views. In *Proc. ICDT Conf.*, pages 19–40, 1997.

[VP] V. Vassalos and Y. Papakonstantinou. Expressive capabilities description languages and query rewriting algorithms. Stanford Technical Report.

[VP97] V. Vassalos and Y. Papakonstantinou. Describing and Using Query Capabilities of Heterogeneous Sources. In *Proc. VLDB Conf.*, pages 256–266, 1997.

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