Mediating and Metasearching on the Internet

Luis Gravano
Computer Science Department
Columbia University
www.cs.columbia.edu/ gravano

Yannis Papakonstantinou
Computer Science and Engineering Department
University of California, San Diego
www.cs.ucsd.edu/ yannis

1 Introduction

The Internet emerges as the largest database. Increasingly, users want to issue complex queries across Internet sources to obtain the data they require. However, finding relevant information sources and querying them manually is problematic: there are numerous sources, and they vary in the type of information objects they contain and in the interface they present to their users. Some sources contain text documents and support simple query models where a query is just a list of keywords. Other sources contain more structured data and provide query interfaces in the style of relational query languages. Furthermore, users have to manually fuse the query results by merging information, removing redundancies, ranking the answer objects in the appropriate order, and so on.

Since it is tedious to contact several heterogeneous sources, users can benefit from *metasearchers* and *mediators*, which are services that provide users with a virtual integrated view of the heterogeneous sources. Users access the view using a unified query interface that offers *location*, *model*, and *interface transparency*, i.e., users have the illusion of a single database and do not have to be aware of the location and interface of the sources. Although users and applications might access data directly through wrappers, mediators and metasearchers offer an integrated view of the world, where information related to the same entity has been fused together, redundancies have been eliminated, and inconsistencies have been removed.

The architecture of metasearchers and mediators are virtually identical (Figure 1). Wrappers export a common data model view of each source's data. Wrappers also provide a common query interface. After receiving a query, a wrapper translates it into a source-specific query or command, hence giving interface transparency to the user. Then, the wrapper translates the query results from the underlying source into the common data model or format.

To evaluate a user query over multiple heterogeneous databases, both metasearchers and mediators will typically perform three main tasks:[1]

- **Database Selection:** Choose the databases that have data relevant to the user query.
- **Query Translation:** Find the query fragment to be evaluated at each of the databases chosen in the previous step, translate these fragments so that they can be executed at their corresponding databases, and retrieve the query results from the databases.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] Note that this query processing strategy assumes that the integrated view is not materialized. An alternative would be to materialize the integrated view and evaluate queries on it directly. The query processing simplicity of the materialized view approach, together with the steep decline in disk storage prices has made it the predominant approach followed by data warehouses that integrate and consolidate corporate information.

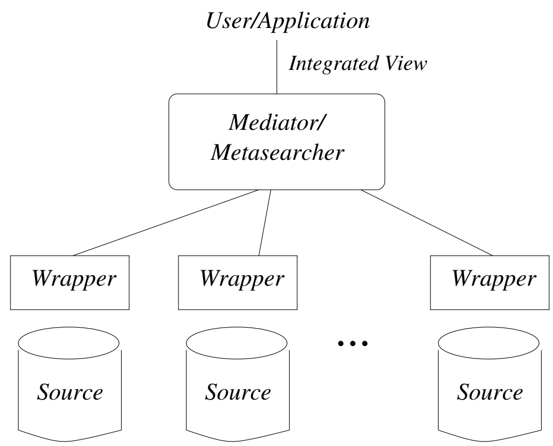

Figure 1: Both metasearchers and mediators use wrappers around the underlying databases. These wrappers provide a relatively uniform view of these databases.

- **Result Merging**: Combine the query results from the above databases into the final query answer.

In spite of their shared goals and architecture, the focus of the research on metasearchers has been quite different from that of the research on mediators. Two key issues explain this difference:

- **View Complexity:** Metasearchers typically operate on top of document databases, and the view that metasearchers export to users is generally some kind of union of the underlying databases. On the other hand, mediators usually integrate multiple relations or objects with complementary information. Thus, fusion of objects from several databases is not uncommon when defining mediator views. The higher complexity of the mediator views requires powerful view definition languages, together with powerful languages to query the integrated view.

- **Query Matches and Result Completeness:** The interaction of a user with a mediator is very similar to the interaction of a user with a relational database system; the user sends a query and the mediator typically returns the *complete answer* to the query. In effect, mediators generally operate over databases where query results are well defined sets of objects, as in the relational model. Metasearchers, however, usually deal with collections of unstructured text documents that return document ranks as the answer to a query, where the ranks are computed using undisclosed algorithms. The matches between queries and documents are "fuzzy." For example, *vector-space* databases compute how "similar" a document and a query are, where this similarity is a number between 0 and 1 [23]. Furthermore, these sources might return *only the best matches* for the query. Hence, metasearchers have to handle query results that have been computed using unknown matching algorithms. Also, metasearchers are aware that *partial answers* to queries are usually acceptable on the Internet, thus abandoning the goal of producing complete answers. This decision has strong implications on the query processing strategies that metasearchers might use.

Next, we describe how metasearchers (Section 2) and mediators (Section 3) address the three tasks that we discussed above to provide a unified view of the underlying heterogeneous databases.

2 Metasearchers

Web indexes like AltaVista (`http://www.altavista.digital.com`) are centralized full-text indexes of HTML documents. Unfortunately, the current web indexes do not manage to index every HTML document there is. Hence, if a user's web index of choice does not return satisfactory results for a query, the user might be forced to contact other indexes until the right documents are found. To complicate matters further, the contents of many text sources are hidden behind search interfaces (e.g., Knight-Ridder's Dialog information service, or the NC-STRL sources [2]). Web crawlers will typically not index the contents of such sources, since these sources' documents are only available in response to queries, not by following hypertext links. Hence, to access the contents of these *search-only sources*, users typically have to contact the sources themselves one by one. In either case, users need to query several autonomous, potentially heterogeneous sources.

Building metasearchers over Internet document sources is difficult because, in general, sources are too numerous. Therefore, finding the best sources for a query is a challenging task. Also, even if we know the document ranking algorithms that sources use, extracting the best objects for a query according to the metasearcher might be an expensive operation, since the sources' ranking algorithms might differ radically from that of the metasearcher's. Building metasearchers is also difficult because different sources are largely incompatible. In effect, the interfaces and query models of these sources vary from source to source. Even individual organizations use *search engines* from different vendors to index their internal document collections. In general, text search engines use different query languages, rank documents in the query results using secret algorithms, and do not export information about the sources in a standard form.

Several metasearchers already exist on the Internet for querying multiple web indexes. However, not all of them completely support the three major metasearch tasks described above. Examples include MetaCrawler [24] (`http://www.metacrawler.com`), SavvySearch (`http://guaraldi.cs.colostate.edu:2000/`}, and ProFusion [7].

STARTS, the *Stanford Protocol Proposal for Internet Retrieval and Search* [9], is an emerging protocol whose goal is to facilitate the three metasearching tasks above. *STARTS* has been developed in a unique way. It is not a standard, but a group effort involving 11 companies and organizations, coordinated by the Digital Library project at Stanford University.

Next, we discuss the three metasearching tasks in more detail, together with some related work. In particular, we summarize the main *STARTS* components and how they facilitate these tasks. For a detailed description of *STARTS*, please refer to [9].

2.1 Database Selection

A metasearcher might have thousands of sources available for querying. Some of these sources might charge for their use. Some of the sources might have long response times. Therefore, it becomes crucial that the metasearcher only contact sources that might contain useful documents for a given query, for which the metasearcher needs information about each source's contents.

To characterize the sources, we could manually write descriptions of their contents. For example, the Information Manifold system [15, 17] relies on declarative descriptions of the sources' contents and capabilities, written in a version of description logic. These descriptions are useful to prune the search space for evaluating user queries efficiently.

Alternatively, metasearchers might rely on automatically extracted summaries of the sources' contents. The *GlOSS* system [12, 10] uses source summaries that include the document frequency for each word in the source's vocabulary. [1] has applied inference networks (from information retrieval) to the database selection problem. Their approach summarizes databases using the same type of information that *GlOSS* keeps, together with the "inverse collection frequency" of the different terms. An inference network then uses this information to rank the databases for a given query.

To extract content summaries automatically from the sources, metasearchers require cooperation from the sources. If a source exports all of its contents (e.g., many web sites), then it is not as critical to have it describe

[2]The NCSTRL sources constitute an emerging library of computer science technical reports (`http://www.ncstrl.org`).

its collection to the metasearchers. After all, the metasearchers can just grab all of the sources' contents and summarize them any way they want. This is what the crawlers of web indexes like AltaVista do. In practice, some sources freely deliver their entire document collection, but others do not. Often, those sources that have for-pay information are of the second type. Morever, for performance reasons, it may still be useful to require that sources export a more succinct description of themselves. If a source "hides" its information (e.g., through a search interface), then it is even more important that the source can describe its contents. Otherwise, if a source does not export any kind of content summary, it becomes hard for a metasearcher to assess automatically what kind of information the source covers.

The *STARTS* protocol specifies that sources should export partial data about its contents [9]. This data is automatically generated, is orders of magnitude smaller than the original contents, and has proven helpful in distinguishing the more useful from the less useful sources for a given query [12, 10]. The *STARTS* summary for a source includes a list of all the words that appear at documents in the source, possibly with various tags, like the position in the documents where each word occurs, and some associated statistics that are easily computed from the source's index structures. For example, a source S might report that the English word *algorithm* appears in the title of 53 documents in S, while the Spanish word *datos* appears in the title of 12 documents. The summary might also tell us that there are 892 documents in the source. A metasearcher can use this information to decide whether a given query is likely to have good matches in source S.

2.2 Query Translation

A metasearcher submits queries over multiple sources. But the interfaces and capabilities of these sources may vary dramatically. Even the basic query model that the sources support may vary. Some search engines (e.g., Glimpse) only support the *Boolean retrieval model* [23]. In this model, a query is a condition that documents either do or do not satisfy. The query result is then a *set* of documents. For example, a query *distributed and systems* returns all documents that contain both the words *distributed* and *systems* in them.

Alternatively, most commercial search engines also support a variation of the *vector-space retrieval model* [23]. In this model, a query is a list of terms, and documents are assigned a score according to how *similar* they are to the query. The query result is then a *rank* of documents. For example, a query *"distributed systems"* returns a rank of documents that is typically based on the number of occurrences of the words *distributed* and *systems* in them. [3] A document in the query result might contain the word *distributed* but not the word *systems*, for example, or vice versa, unlike in the Boolean model above.

Even if two sources support a Boolean retrieval model, their query syntax often differ. More serious problems appear if different attributes are available for searching at different sources. For example, a source might support queries like *abstract "databases"* that ask for documents that have the word *databases* in their abstract, whereas some other sources might not support the *abstract* attribute for querying.

Another complication results from different stemming algorithms or stop-word lists being implicit in the query model of each source. (Stemming is used to make a query on *systems* also retrieve documents on *system*, for example. Stop words are used to not process very frequent words like *the* in the queries.) If a user wants documents about the rock group *The Who*, knowing about the stop-word behavior of the sources would allow a metasearcher to know whether it is possible to disallow the elimination of stop words from queries at each source.

As a result of all this heterogeneity, a metasearcher would have to translate the original query to adjust it to each source's syntax. To do this translation, the metasearcher needs to know the characteristics of each source. The work in [3, 4] illustrates the complexities involved in query translation. Querying multiple sources is much easier if the sources share a common query language.

The *STARTS* protocol specifies a flexible query language for sources. Even if support for most of this language is optional, query translation is much simpler if sources reveal what portions of the language they support. Thus,

[3] These ranks also typically depend on other factors, like the number of documents in the source that contain the query words.

STARTS asks that sources export detailed information on their searching capabilities. This information includes, for example, what attributes are supported for searching at each source (e.g., *author*, *title*, *body of text*).

2.3 Result Merging

A source that supports the vector-space retrieval model ranks its documents according to how "similar" the documents are to a given query. In practice, there are many ways to compute these similarities. To make matters more complicated, the ranking algorithms are usually proprietary to the search engine vendors, and their details are not publicly available.

Merging query results from sources that use different and unknown ranking algorithms is hard. For example, source S_1 might report that document d_1 has a *score* of 0.3 for some query, while source S_2 might report that document d_2 has a score of 1,000 for the same query. If we want to merge the results from S_1 and S_2 into a single document rank, should we rank d_1 higher than d_2, or vice versa? (Some search engines are designed so that the top document for a query always has a score of, say, 1,000.)

It is even hard to merge query results from sources that use the same ranking algorithm, even if we know this algorithm. The reason is that the algorithm might rank documents differently based on the collection where the document appears. For example, if a source S_1 specializes in computer science, the word *databases* might appear in many of its documents. Then, this word will tend to have a low associated weight in S_1 (e.g., if S_1 uses the *tf·idf* formula for computing weights [23]). The word *databases*, on the other hand, might have a high associated weight in a source S_2 that is totally unrelated to computer science and contains very few documents with that word. Consequently, S_1 might assign its documents a low score for a query containing the word *databases*, while S_2 assigns a few documents a high score for that query. Therefore, it is possible for two similar documents d_1 and d_2 to receive very different scores for a given query, if d_1 appears in S_1 and d_2 appears in S_2.

The problem of merging document ranks from multiple sources has been studied in the information retrieval field, where it is often referred to as the *collection fusion* problem. Given a query, the goal is to extract as many of the *relevant* documents as possible from the underlying document collections, where relevance is a subjective notion. Key decisions include how far "down" each document rank to explore, and how to translate the scores computed by the sources into the metasearcher's scores. An approach to address these problems is to learn from the results of training queries. Given a new query, the closest training queries are used to determine how many documents to extract from each available collection, and how to interleave them into a single document rank [29, 30]. Another approach is to calibrate the document scores from each collection using statistics about the word distribution in the collections [1].

The *STARTS* protocol asks sources to report term statistics in their query results. For example, the entry for a document d in the result for a query containing the word *database* might report the number of words in d, and that the query word *database* occurs 15 times in d, among other statistics on the document and its source. This way, metasearchers can try and merge query results in meaningful ways without having to retrieve the entire documents.

3 Mediators

Mediator systems [31] provide users with an integrated view of multiple heterogeneous information sources. These sources are not necessarily structured databases like relational and object oriented databases. In particular, the sources might contain semistructured data such as HTML (and very soon XML) documents, chemical abstracts, genome data, biology "metadata" (e.g., annotations of raw data), and bibliographic entries. Furthermore, the sources provide different and typically limited query interfaces to the data. Next, we describe how mediators handle the database selection and query translation tasks. We do not address the result merging task here, which is much simpler for mediators than it is for metasearchers. (See Section 2.3.)

3.1 Database Selection

When a mediator receives a query it first performs a *query decomposition* step where it computes the sources that are relevant to the query and decides what data is needed from each source [15, 13, 19, 18]. Mediators always use human-provided descriptions of the relationship between the integrated view and the sources in order to decompose queries. Assuming a significant amount of abstraction, we categorize the view/source relationship specifications in the following two broad classes and we provide a high level comparison of the virtues and disadvantages of the two approaches. A detailed technical comparison of the two approaches can be found in [26]. In practice, a hybrid of the two approaches is both desirable and possible.

The *view definition approach*, which is usually favored by the database community, describes the integrated view collections as views of the underlying source collections. For example, we may join a `salary` relation from the `payroll` source with an `employer` relation from the `human resources` source to produce an integrated view `complete_emp_info` that a mediator will export. Then, given a query, the mediator will expand the references to the integrated view with its definition, hence producing a rewritten query that has references to source collections only. Notice that query processing is similar to conventional database query processing and hence it can be based on existing query processors.

The view definition approach is not particular to the relational model. Indeed, recent mediator projects have preferred object oriented or semistructured models [6] for this purpose and they have also tuned the view definition language to the particular needs of integration.

The main advantage of the view definition approach is that source information can be integrated and transformed in complex ways. [4] On the other hand, this approach lacks modularity when it comes to adding new sources. For example, consider a view that is the union of collections of car advertisements. We will need to update the view specification every time a new car advertisement source becomes available. Furthermore, this approach misses a feature that is very desirable for query optimization: there is no direct description of the contribution of a source to the integrated view. For example, there is no way to express in the view definitions that one source covers only Honda's, while another source covers only BMW's. Hence, a mediator will not be able to direct a query on Honda's to the first source only. The source definition approach addresses this problem.

The *source definition approach*, which originated in the artificial intelligence and description logic communities, assumes the existence of global predicates and collections – say, a collection `car_ads` – and then defines the source contents with respect to the global predicates. For example, the Honda source above is defined to contain `car_ads` tuples with `make = Honda`. However, this statement should not be taken as a view definition of the source with respect to the global integrated view; the definition does not imply that the Honda source has all Honda `car_ads` tuples that appear in the integrated view. It only specifies that cars found in this source have to be Honda's.

3.2 Query Translation

Once the mediator computes what data is needed from each source, an optimizer develops an efficient plan that specifies what queries must be sent to the sources, and how the results will be combined. Note that the mediator has to retrieve the data required from a wrapper using only queries that the wrapper can translate into source specific queries or commands. For example, a mediator should not send a join query to a document retrieval source that can do selections only. Instead, it must decompose the join query into simple selection queries that can be handled by the source. At the same time, the capabilities of the most sophisticated sources must be exploited. This precludes the use of a simplistic "lowest common denominator" approach. For example, assuming that only one-condition selection queries are supported by the sources will also lead to inefficient plans because the mediator will not exploit the query processing abilities of sources that can process joins and multiple conditions. Indeed, in most cases, it is beneficial to produce *algebraically optimal* plans, i.e., plans that push as much work

[4] In theory, the only limit is the computational complexity of the view definition language.

as possible down to the sources. Focusing on select-project-join queries, a plan P is algebraically optimal [21] if there is no other plan P' such that for every query w sent to a wrapper by P there is a corresponding query w' of P' such that the sets of relations and conditions of w' are a superset of the corresponding sets of w and the set of exported attributes of w is a superset of the set of exported attributes of w'.

Several projects [22, 16, 25, 21, 20, 27] propose and discuss query processors that, given some description of the capabilities of the participating sources, adapt to the different, limited capabilities of the sources. An elegant approach to describing capabilities was introduced in [22] where the supported query interfaces were described by a finite number of parameterized views. For example, the following describes that the source `bib` supports substring conditions on the `title` field of the collection `reports`.

```
SELECT *
FROM bib.reports
WHERE reports.title LIKE $X
```

If we use these descriptions, finding a plan that consists of supported queries only is similar to finding a plan to answer queries using views. [20, 21, 27] provide languages for the specification of an infinite number of parameterized views and show that it is still possible to find all the potentially optimal plans. Finally note that other projects [14] tackle the rewriting issue as one describing acceptable plans that can be passed to the wrappers. However, the central ideas remain unchanged.

As we described in Section 2.2, the capabilities-based rewriting problem has also been addressed for text document collections [3, 4]. The system described in [3, 4] tends to produce efficient plans because it knows the semantic relationship between its own relations and access methods, and the source relations and access methods. For example, assume that the user query requests papers by *Knuth* where the title contains the word *complexity*. A source may not support a `contains word` predicate while, instead, it supports a `substring` predicate. By making the mediator aware that any document that satisfies the former predicate will also satisfy the latter one, we can rewrite the user query into one that uses `substring` and does the additional filtering at the mediator. In effect, such a mediator performs a form of semantic optimization [2].

4 Conclusion

Metasearchers and mediators provide uniform views over large numbers of heterogeneous databases. The problems that research in both areas has addressed, though, tend to differ significantly. However, some recent work has started to bridge the gap between these two areas:

- **Semistructured Data:** Metasearchers initially focused on text sources, while mediators were used for querying structured databases. However, both areas are converging on the study of sources with semistructured data. Such data has more structure than free text, but this structure is not as rigid, regular, or complete as that of relational schemas. Bibliographic entries, chemical abstracts, genome data, and a large number of HTML pages are typical examples of semistructured data. The emerging XML standard highlights the importance of an information exchange model for semistructured data.

  Database and mediator research has modeled semistructured data as graphs where the nodes carry semantic *labels* (i.e., the labels are essentially the metadata). Appropriate query and view definition languages have been designed for querying and transforming the graph data. However, the proposed query languages have logical underpinnings, and do not capture relevance ranking. Hence, they are still not well suited to many Internet querying scenarios where the query language is not logic-based.

- **Ranked Query Results:** Internet sources tend to overlap in arbitrary ways. Furthermore, users are often not interested in receiving "complete" answers to their queries. Instead, users sometimes prefer to receive the “best matches” for their queries. These characteristics of the data sources and the user expectations present both problems (we should avoid retrieving duplicate data), and opportunities (we can access fewer sources during query processing). To address these problems and opportunities several recent papers on mediators have proposed ways to define source overlap and optimization algorithms that produce efficient query execution plans [5, 28]. This work exploits the fact that users might be satisfied with efficiently computed partial answers to their queries, and produce incremental query plans for answering their queries, just like some metasearching systems already do for text documents [12, 10]. There has also been some initial work on producing ranked query results from sources of structured or semistructured data. (See [8] and DataSpot (`http://www.dataspot.com`).) The work in [11] addresses the problem of querying over multiple structured data sources that rank query results using different algorithms. One such source could be a real-estate agent that receives queries from users, and ranks the available houses according to how well they match the users' specification, for example.

Acknowledgments

We thank Roy Goldman and Vasilis Vassalos for their useful comments on the paper.

References

[1] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the Eighteenth ACM International Conference on Research and Development in Information Retrieval (SIGIR'95)*, July 1995.

[2] U. Chakravarthy, J. Grant, and J. Minker. Logic-based approach to semantic query optimization. *ACM Transactions on Database Systems*, pages 162–207, 1990.

[3] C.-C. K. Chang, H. García-Molina, and A. Paepcke. Boolean query mapping across heterogeneous information sources. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):515–521, Aug. 1996.

[4] C.-C. K. Chang, H. García-Molina, and A. Paepcke. Predicate rewriting for translating Boolean queries in a heterogeneous information system. Technical Report SIDL-WP-1996-0028, Stanford University, 1996. Accessible at `http://www-diglib.stanford.edu/cgi-bin/WP/get/SIDL-WP-1996-0028`.

[5] D. Florescu, D. Koller, and A. Levy. Using probabilistic information in data integration. In *Proceedings of the Twenty-third International Conference on Very Large Databases (VLDB'97)*, Aug. 1997.

[6] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: data models and languages. *Journal of Intelligent Information Systems*, 8:117–132, 1997.

[7] S. Gauch and G. Wang. Information fusion with ProFusion. In *Proceedings of the World Conference of the Web Society (WebNet'96)*, Oct. 1996.

[8] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity searching in databases. In *Proceedings of the Twenty-fourth International Conference on Very Large Databases (VLDB'98)*, Aug. 1998.

[9] L. Gravano, C.-C. K. Chang, H. García-Molina, and A. Paepcke. *STARTS*: Stanford proposal for Internet metasearching. In *Proceedings of the 1997 ACM International Conference on Management of Data (SIGMOD'97)*, May 1997.

[10] L. Gravano and H. García-Molina. Generalizing *GlOSS* for vector-space databases and broker hierarchies. In *Proceedings of the Twenty-first International Conference on Very Large Databases (VLDB'95)*, pages 78–89, Sept. 1995.

[11] L. Gravano and H. García-Molina. Merging ranks from heterogeneous Internet sources. In *Proceedings of the Twenty-third International Conference on Very Large Databases (VLDB'97)*, Aug. 1997.

[12] L. Gravano, H. García-Molina, and A. Tomasic. The effectiveness of *GlOSS* for the text-database discovery problem. In *Proceedings of the 1994 ACM International Conference on Management of Data (SIGMOD'94)*, May 1994.

[13] L. Haas, D. Kossman, E. Wimmers, and J. Yang. An optimizer for heterogeneous systems with non-standard data and search capabilities. *Special Issue on Query Processing for Non-Standard Data, IEEE Data Engineering Bulletin*, 19:37–43, Dec. 1996.

[14] L. Haas, D. Kossman, E. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of the Twenty-third International Conference on Very Large Databases (VLDB'97)*, Aug. 1997.

[15] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI Spring Symposium Series*, Mar. 1995.

[16] A. Levy, A. Rajaraman, and J. Ullman. Answering queries using limited external processors. In *Proceedings of the Fifteenth ACM Symposium on Principles of Database Systems (PODS'96)*, pages 227–237, June 1996.

[17] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Databases (VLDB'96)*, Sept. 1996.

[18] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proceedings of the Twenty-second International Conference on Very Large Databases (VLDB'96)*, Sept. 1996.

[19] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specifications. In *Proceedings of the 1996 ICDE Conference*, pages 132–41, 1996.

[20] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman. A query translation scheme for the rapid implementation of wrappers. In *Proceedings of the 1995 DOOD Conference*, pages 161–86, 1995.

[21] Y. Papakonstantinou, A. Gupta, and L. Haas. Capabilities-based query rewriting in mediator systems. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS'96)*, June 1996.

[22] A. Rajaraman, Y. Sagiv, and J. Ullman. Answering queries using templates with binding patterns. In *Proceedings of the Fourteenth ACM Symposium on Principles of Database Systems (PODS'95)*, pages 105–112, May 1995.

[23] G. Salton. *Automatic Text Processing: The transformation, analysis, and retrieval of information by computer*. Addison-Wesley, 1989.

[24] E. Selberg and O. Etzioni. Multi-service search and comparison using the MetaCrawler. In *Proceedings of the Fourth International WWW Conference*, Dec. 1995.

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

[26] J. Ullman. Information integration using logical views. In *Proceedings of the 1997 ICDT Conference*, pages 19–40, 1997.

[27] V. Vassalos and Y. Papakonstantinou. Describing and using query capabilities of heterogeneous sources. In *Proceedings of the Twenty-third International Conference on Very Large Databases (VLDB'97)*, pages 256–266, Aug. 1997.

[28] V. Vassalos and Y. Papakonstantinou. Using knowledge of redundancy for query optimization in mediators. Technical report, Computer Science Department, Stanford University, 1998.

[29] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The collection fusion problem. In *Proceedings of the Third Text Retrieval Conference (TREC-3)*, Mar. 1995.

[30] E. M. Voorhees and R. M. Tong. Multiple search engines in database merging. In *Proceedings of the Second ACM International Conference on Digital Libraries (DL'97)*, July 1997.

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