An Overview and Classification of Mediated Query Systems

Ruxandra Domenig*, Klaus R. Dittrich
Department of Information Technology, University of Zurich
{domenig|dittrich}@ifi.unizh.ch

Abstract

Multimedia technology, global information infrastructures and other developments allow users to access more and more information sources of various types. However, the "technical" availability alone (by means of networks, WWW, mail systems, databases, etc.) is not sufficient for making meaningful and advanced use of all information available on-line. Therefore, the problem of effectively and efficiently accessing and querying heterogeneous and distributed data sources is an important research direction. This paper aims at classifying existing approaches which can be used to query heterogeneous data sources. We consider one of the approaches – the *mediated query approach* – in more detail and provide a classification framework for it as well.

1 Introduction

Progress in both, persistent storage for all kinds of data and computer network technology, is the main reason for the explosive growth of data that are available on-line. New technologies, most notably the World Wide Web (WWW), allow anybody to access data very easily, independent of its physical location. However, a uniform presentation interface to distributed data is by far not enough; users need means to make intelligible use of large amounts of heterogeneous data:

- Users must be able to describe *what* information they are looking for, i.e., *what criteria* it should meet, irrespective of which source might offer it. When searching, information needs can be very diverse. The search may range from an "approximate" one, where no knowledge about the structure and the characteristics of the underlying data sources is available, to an exact one, where the user knows the underlying data sources and their query capabilities as well as the (kind of) information he can expect. We call the first kind of search *imprecise* and the second *precise*.
- Users must be able to correlate information of different types and from different sources.
- Information meeting specified search criteria must be presented in a uniform way and may eventually require further manipulation like ordering or grouping.

Summarizing, we consider heterogeneous data sources which provide a query interface and we are looking for systems which allow for retrieving these data in a comprehensive and unified way. There is a multitude of such approaches. In this paper, we will first give an overview of some of them. Our focus is on querying, not on manipulation of the data. We assume that information manipulation is in most cases done on a system-by-system basis, working directly with the participating component systems. Our second aim is to present one approach in detail. Its realization is based on *mediators* [Wie92], which were introduced with the argumentation that they "simplify, abstract, reduce, merge and explain data". In other words, mediators add value to data and thus help to exploit more of their information. As a consequence, *mediated query systems* (MQS) were developed. An MQS is a system

- build on top of heterogeneous data sources,
- implemented using mediators, and
- which allows for querying the content of these data sources.

Users of MQS can express queries in a unified way, but data are still stored in the local systems. Internally, a query is decomposed into subqueries which are sent to those data sources that may be able to answer them, and finally the results are combined and presented in a uniform way.

*The work of R. Domenig is supported by Ubilab, the IT Laboratory of UBS.

The remainder of the paper is structured as follows: Section 2 classifies the approaches for integrating heterogeneous data and Section 3 presents the classification features for mediated querying systems. Section 4 concludes the paper.

2 A Taxonomy of Systems for Querying Heterogeneous Data

The huge amount of data available in electronic form today has triggered the development of various approaches for heterogeneous data, retrieval and information extraction which we classify as depicted in Figure 1.

First, we distinguish between *materialized* and *virtual* approaches. In a materialized approach, data originating from local sources are integrated into one single new database on which all queries can operate. In a virtual approach, data remains in the local sources. Thus, queries operate directly on them and data integration has to take place "on the fly" during query processing. On a lower level, the classification further distinguishes approaches with respect to the structural heterogeneity of the queried data[1] and their origin (i.e., whether the retrieved data is more or less entirely stored in the underlying data sources – *native* – or whether it may also be *derived* from the data stored in underlying sources). Other features not directly visible from this figure include the kind of supported queries (precise and/or imprecise), and the extensibility of the system, i.e. whether additional data sources can be easily accommodated or not.

There are essentially two variants of materialized systems:

- One possibility is to migrate the data from the local systems to a "universal" DBMS, which is able to handle all (or many) types of information. Examples are object-relational DBMS or object-oriented DBMS. Data from local systems are extracted, integrated and stored in the central database. Thereafter, the local systems are not used any more, at least in principle.

  The main drawback of this approach is that existing applications for the local systems have to be rewritten for the new database. Moreover, the process of data migration can be very expensive, since the old data has to be transformed and often semantically enriched for the new system (the new database usually has a richer data model). Another aspect is that the migrated data can be accessed not only for reading, but also for writing. For this reason, new access control policies have to be established. Nevertheless, migration can be a good solution, for example, if users or applications need the whole functionality of a DBMS (and not just the query functionality) and the old systems' applications are no longer needed, at least in their former form [BGD97].

- In the second approach, *data warehousing*, data from the local data sources are imported into one DBMS, the data warehouse. The difference to the previous case is that the underlying data sources are still operational, so in fact the data is replicated. The warehouse data is typically not imported in the same form and volume as it exists in the local data systems. It may be transformed, cleaned and prepared for certain analysis tasks, like data mining and OLAP (*Online Analytical Processing*). Data warehouses often do not make the most recent data available, since a data warehouse is usually not updated immediately after a local data source has changed. However, they store historical data, as required by OLAP and data mining applications.

With respect to querying, both approaches have the advantage that real DBMS functionality is available, so precise searching is supported. However, the overhead for building such systems is significant and imprecise search is not supported.

In the *virtual* approach, data remain in their local systems. A layer is built on top of them which takes the query from the user, processes it, sends (parts of) it to the appropriate sources and presents the results. Three major approaches can be distinguished:

- Regarding querying of unstructured sources, *(meta)search engines* have gained importance, mainly due to the popularity of the Web.

  Search engines on the Web allow to search data which are physically stored at distributed sites and thus provide a single access point for them. Data is homogeneous, since it usually consists of textual parts of HTML files.

  In order to increase retrieval effectivity, metasearch engines have been introduced. Examples are SavvySearch [DH97], MetaCrawler [SE97] and the current version of Informia [BBMS98]. In a metasearch engine, queries are

[1] Data may be structured, semistructured or unstructured; Section 3 will give a definition of these notions. Until then, an intuitive understanding is sufficient.

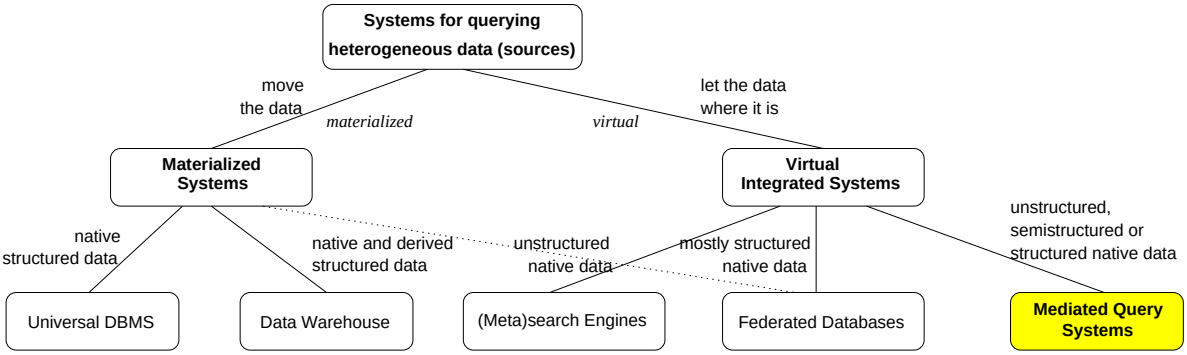

Figure 1: Classification of Systems for Querying Heterogeneous Data

sent to different search engines whose results are collected and then presented to the user in a unified way. The main focus of metasearch engines lies with the combination of results.

Summarizing, (meta)search engines are suitable for unstructured data and support imprecise search.

- The aim of *federated databases* is to give the user the impression of working with one DBMS, but in fact the data is managed by several individual DBMS. Since a federated database still provides typical DBMS functionality, queries support only precise search.

  Federated database systems may also be regarded to follow the materialized approach (as indicated in Figure 1 through the dotted line), because they may store parts of the underlying data in an internal repository (for example, FRIEND [MJD97]). However, this materialization is only partial and/or temporary (to enhance performance, for example). The data is still managed in and for most cases retrieved from the local systems, in contrast to the previously defined materialized approach.

  Summarizing, federated databases are suitable for structured data and support precise search. Federated databases have been investigated in the research community for a long time. In the last years, commercial products have been made available, for example DataJoiner from IBM [Co.97], Cohera Data Federation System from Cohera [COH], MIRACLE from ORACLE [Hug96], EDI/S from Information Builders Inc. [Inc97]. They implement many issues of a federated database, but do not offer complete solutions yet.

- The last approach presented in Figure 1 is the *mediated query approach*, roughly characterized in Section 1. Query processing in this case is very similar to the metasearch case, with the difference that data in the underlying sources may be heterogeneous, i.e. structured, semistructured or unstructured.

Since the approaches presented in Figure 1 often have common aspects, their classification does not always render a sharp distinction for each and every aspect. For example, there are data warehousing systems which use the concept of mediation for the task of data integration (SIRIUS [VGD99], H2O [ZHKF95]). However, the classification defines at least some boundaries between the existing approaches and highlights their advantages and disadvantages. Table 1 summarizes the strengths and shortcomings of the approaches classified in Figure 1.

The explosive growth of data in the WWW also led to the development of systems which apply the database-style of querying and management to Web data, for example WebSQL [MMM97], WebOQL [AM98], Strudel [FFK+97], etc. These systems use some techniques and mechanisms from the approaches presented above, but only for the Web. [FLM98] presents a survey of existing systems for managing data in the Web using database technology.

Since we are interested in systems for both precise and imprecise search, we focus on mediated query systems in the remainder of the paper.

Universal DBMS	+	full DBMS functionality
	+	suitable for precise search
	−	not suitable for imprecise search
	−	static
	−	migration of existing applications
	−	expensive migration
Data Warehouses	+	architected/optimized for special purposes
	−	not all data available
	−	mostly static
Federated DBMS	+	full DBMS functionality
	+	suitable for precise search
	−	not suitable for imprecise search
	−	mostly static
(Meta)Search Engines	+	suitable for unstructured data
	+	high effectivity of imprecise search
	−	not useful for precise search
Mediated Query Systems	+	support all sorts of possible data
	+	support precise and imprecise search
	+	support dynamic set of data sources
	−	only queries (no updates)

Table 1: Evaluation of Heterogeneous Data Systems with Regard to Querying.

3 Classification Features for MQS

This section provides a rough classification of features which characterize mediated query systems. A considerable number of mediated query systems has been proposed, including Garlic [Cea95] from IBM Almaden Research Center, TSIMMIS [PGMW95] from Stanford University, InfoSleuth [Bea97] from Microelectronics and Computer Technology Corporation (MCC), DIOM [LP97] from the University of Alberta and the Oregon Graduate Institute, Information Manifold (IM) from AT&T Labs Research [LRO96], MAGIC [KRR97] from the University of Karlsruhe, SIMS [AKH96] from the University of Southern California, DISCO [TRV96] from INRIA and SINGAPORE [DD99] from the University of Zürich.

In our classification we distinguish between *functional* and *implementation* features. *Functional* features characterize the way the MQS presents itself to the outside, i.e., to the users or applications. The internal, technical realization of systems is characterized by *implementation* features. Obviously, there is a strong interconnection between both: the functionality exported to the outside depends on the technical realization of the system, and vice versa. Table 2 summarizes the implementation and functional features we consider.

3.1 Functional Features

Functional features can be classified according to various criteria, which are presented below.

3.1.1 Query Characteristics

The way users can access the system, i.e., the interface through which they can express their information needs and the means through which they can understand the "information space" to be queried, is obviously of utmost importance for the usability of an MQS. Users must be able to do both, precise and imprecise search. Therefore, database and information retrieval techniques should be combined. The following list includes aspects which can be used to characterize queries:

- *Query language.* A query language is used to express the information needed by users. It can be a database-like query language, where users can retrieve information based on attribute names, types and their operations. At the other end of the spectrum are search engines, where a query is expressed using combinations of keywords (like boolean operators "and", "or", "not") or sometimes even natural language (i.e. sentences).

Functional Features	Implementation Features
Query Characteristics	Architecture
Query Language	Centralized/Decentralized Architecture
Query Type (Exact/Vague)	Functionality Source Layer
Schema Dependence	Functionality Mediation Layer
Result Presentation	Internal Data Representation
Ranked List	Simple/Complex Model
Relevance Feedback	
Structural Properties	Query Processing
Unstructured Data	Source Selection
Structured Data	Query Splitting
Semistructured Data	Query Optimization
	Query Semantics
Extensibility	Metadata
Global Model Extensibility	Metadata Content
Wrapper Extensibility	Metadata Acquisition
Mediator Extensibility	
Metadata Extensibility	

Table 2: Functional and Implementation Features for Mediated Query Systems

Since MQS should support the querying of all kinds of information, a good solution would be to combine both styles of querying.

- *Query Type.* Since the information needs of users are very diverse, they can express this by different types of queries. They can send an *exact* query to structured systems, provided they know the sources and their query capabilities, i.e., they execute a precise search. When sources (their structure and query capabilities) are unknown to users, they can send a *vague* query, i.e., they execute an imprecise search.

- *Schema Dependence of Queries.* The way users query MQS is related to the existence or not of a (global) schema and its use. Two main cases can be distinguished.

  In the first case, the MQS has a global schema and the user has to use it for querying the system.

  In the second case, a schema is not necessarily needed, but the schemas of the underlying sources (if they have one) are still expressed internally in terms of the global model and the user can refer to them or not in order to query the system. Expressed differently, there is no need for a global schema, but if there is one, the user can (but need not) use it. For example, an MQS may offer the possibility to search for an attribute value in a relational database, without having to specify which attribute has this value.

  The first case can be further split into two subcases, depending on how the global schema was defined for the MQS. First applications may have been analyzed and then, based on the result, a schema for the system was created (i.e. database design as usual). This means the MQS has an *a priori* defined schema and all integration and translation steps – and also the queries – are performed against this well-defined schema.

  Alternatively, every source exposes its local schema in terms of the global model. These homogenized local schemas are then integrated to yield the global schema for the MQS. All queries are performed against this global schema.

3.1.2 Result presentation

The way results are presented to the user or application is strongly related to the way an MQS is queried. DBMS support only precise search. In the case of a search engine, on the other hand, the results are only "related" to the user's query, they represent possible answers to the query. Thereby, two problems can occur. First, too many results may be produced so that the user is not able to go through all of them in a meaningful way. Second, other information items may exist in the system which were not returned, but which would answer the query anyway. Two techniques from information retrieval can be applied in the case of MQS:

- *Ranked List.* The first technique to present the results is to calculate a list of the retrieved information items and present them in decreasing relevance order. The relevance is calculated using different heuristics and intuitively represents the probability that an information item answers the given query. The main challenge is to combine the concept of relevance (for imprecise search) with the concept of structured information (for precise search).

- *Relevance Feedback.* The main idea behind this technique is to give users the possibility to improve the computation of results by specifying more facts about the information they are looking for than just the query. Usually, relevance feedback means that the MQS presents a list of information items which can answer a query, and the user marks them as relevant or not. The system then calculates a new list, based on the initial query and the relevant information items.

3.1.3 Structural Data Properties

As already mentioned, we assume that data are heterogeneous. One important kind of heterogeneity is structural heterogeneity, i.e., data from the underlying data sources may be structured, semistructured or unstructured. In order to define this "structuredness", we consider how data are stored physically and also what kind of operations are available for them. We assume that all data is composed of *data elements*. Then, we distinguish between:

- *Structured Data Element.* A data element is called structured if it

  - adheres to a well-defined schema that defines its (recursive) composition out of other data elements, or
  - is an instance of a simple atomic data type like integer, real, or character.

  A schema has the following properties:

  - It is defined using a *type system*.
  - It is defined *a priori*, i.e., before the data element is stored.
  - It is *explicit*, i.e., it is stored separately from the data.
  - It is *rigid*, i.e., the data element always must "obey" the structure.
  - It is *exposed*, i.e., it can be queried and can be used when querying the data element.

  Examples of structured data elements are data stored in DBMS. A query against a structured data element is a *structured query* and is used for precise search. A structured query is based on the structure of the data elements and the type system.

- *Unstructured Data Element.* A data element is unstructured if:

  - it is not of a simple atomic data type like integer, real, character, or
  - does not adhere to any underlying schema, or
  - its schema does not define any composition beyond simple bytes or character strings.

  Examples of unstructured data elements are text, video and audio, since they can only be decomposed into characters or bytes. A query against unstructured data consists of sequences of characters or bytes and operations on them (for example, for boolean text retrieval, one has the operations "and", "or", "near", etc.). We call this an *unstructured query*.

  Querying unstructured data is comfortable for the user, since he does not need to care about any structure of the data or data types. However, his information need may only be approximately fulfilled, which is often undesired.

- *Semistructured Data Element.* A data element is semistructured if

  - it has structure, but the structure is *not rigid*, and/or
  - the structure definition or parts of it is not necessarily separated from the data element, i.e. it may be *implicit*.

  For the first aspect, consider as an example an *address* which is usually composed of a street, a number, a zip code, and a location. However, an address can also comprise a name of a house, a zip code, and a location. This means an address has structure, but the structure is not rigid. In this case, a possible schema for an address might look like this (we use ODMG's ODL notation):

struct(**string** street, **short** number,
short zip-code, **string** location)

or

struct(**string** street, **short** number,
struct(**string** name,
short zip-code) location)

or

struct (**string** house_name, **short**
zip-code, **string** location),

similar to *variant types* in programming languages [Coo80]. A concrete address value may in this case match any one of the given type definitions.

The second issue is related to the way the schema is defined. For DBMS, the schema is defined separately and *a priori* and the data is stored accordingly. For semistructured data, the schema or parts of it might not (and cannot be) defined in this way, but may be "hidden" in the data themselves. In this case the possible schema of an address could be just

(**string** address),

and one such string might be

(house_name:"Casa della Neve",
zip-code:"6098", location:"Magadino").

Thus, the *address* is self-describing, and for this reason its structure is implicit.

3.1.4 Extensibility

MQS should be extensible in the sense that new sources can be registered and existing ones can be disconnected. This may affect various components of the system and the goal is to allow such extensions with as little manual effort as possible, despite the heterogeneity of sources.

3.2 Implementation Features

The realization of the functional features depends on the design of the system, i.e., its implementation features. We classify those as follows (Table 2):

3.2.1 Architecture

Mediated query systems have a three-tier architecture ([Wie92]): the lowest layer includes the *data sources layer*, the middle layer is the *integration layer* and the upper layer is the *user or application layer*. The data source layer contains the sources and components coupling them to MQS. These are so-called *wrappers* which export the functionality and the data in a way that makes all sources "look alike" to the integration layer. The wrappers are implemented based on the query capabilities of the sources. If a source does not have any query capabilities, the wrapper may even be able to retrofit those. The mediation layer is concerned with the processing of queries, integration issues and result combination. The user layer is the interface which provides the functional features of an MQS. There are two issues related to architecture:

- *Centralized/Decentralized Architecture.* The mediation layer can be designed with either a decentralized or a centralized architecture in mind. In the decentralized approach, mediation is performed by a network of components where each one achieves some identifiable task (e.g. a query planning component which defines plans for the processing of a query). Obviously, components can be added and replaced with rather small effort. Components export their functionality and communicate with other components using a communication language (for example KQML in InfoSleuth [Bea97]). In a centralized architecture, the system cannot be easily extended, as for the decentralized case. However, the overhead for communication and management of the components is not required in this case.

- *Source/Mediation Layer Functionality.* This issue is concerned with the functionality of the user and the mediation layer and how it should be split between them. One possibility is to build *fat wrappers*. A fat wrapper receives a query expressed in the global query language as its input and outputs an information item expressed in the global data model. Fat wrappers thus implement the whole source-specific functionality and (semantic) adaptations to the global system. The advantage of a fat wrapper is the fast processing of queries in the mediation layer. Query processing here means just to find out those sources which could answer the query, split the query and produce subqueries expressed in the global query language. Obviously, using a fat wrapper affects the extensibility of an MQS, because

whenever a new source is added, a lot of functionality has to be implemented in the new wrapper. Better extensibility is provided if thin wrappers are used. In this case, a fat mediator layer is required. It must provide as much functionality as possible and includes even parts of the syntactic translations for the underlying sources. This implies that a fat mediator layer has to cover a large variety of models and languages and it may also hamper query processing efficiency.

3.2.2 Internal Data Representation

Data and queries need to be represented in the integration layer of the global data model. If the MQS is designed to just select simple records of data and simple relationships between them, a simple data model is sufficient. If the system has to represent more complex relationships and also behavioral elements, then a more complex model has to be chosen (like e.g. an object-oriented model). Obviously, complex functionality of the components leads to a complex data model.

3.2.3 Query processing

The steps between receiving a query at the user interface and sending (parts of) it to the underlying sources are implemented in the query processing component of an MQS. The concrete query processing algorithm to be used depends on many factors: what is the global query language of MQS, what kinds of queries are supported, which sources should be involved in query processing, etc. The following features are used to characterize query processing:

- *Source Selection.* The first step of query processing is to find the sources that could contribute to answer the query. Many heuristics are available for this task. Besides the sources specified in the query, the MQS can select other sources based on their *content description*, provided it is stored in the system. Next, the *availability* of the sources and their *performance* can be considered .Sources can also be selected based on *structural information* in the query. If, for example, a query specifies an attribute "Title", structured or semistructured sources containing this attribute can be selected.

- *Query Splitting/Optimization/Semantics.* The next step is to split the query. This process takes source selection into account, but also the semantics of the query, i.e., the meaning of attributes and operations used in the query. Often, for an MQS the semantics must support precise and imprecise search (for example by extending DBMS query languages with additional features). Another important issue is query optimization during the process of query splitting. One has to consider optimization at the underlying sources, but also global operations which could affect the efficiency of the global query.

3.2.4 Metadata

All query processing components rely on extensive knowledge about available data sources and their abilities. Most of this metainformation has to be collected and stored in the so-called metadata repository, where the following features are of interest:

- *Metadata Content/Ontologies.* The first issue is the content of the metadata. It depends on the requirements for the system (for example, if the MQS is used for a certain application, data about it has to be stored), but also on its internal realization (if the MQS is, e.g., designed to have fat wrappers, part of the metadata will be hidden in the wrappers). The metadata repository may also include ontological knowledge[2].

- *Metadata Acquisition.* There are two ways to store metadata in the repository. In one approach, wrappers are responsible for this job and thus have to be programmed accordingly.[3] Another possibility is to build a separate component for metadata registration which provides an appropriate specification language. In this case, the system administrator has to find out the relevant information and specify it to the registration component. While this approach is not automated, it also allows for more flexibility and probably leads to a more complete description of metadata. Metadata acquisition is also related to the evolution of sources: e.g., when information about a source changes (for example, the schema), this information has to be forwarded to the metadata repository.

[2] An ontology is "a specification of a conceptualization" ([Gru93]). In particular it is related to the problem that similar information is represented using different vocabularies (for example a "lecture" or a "seminar" are similar concepts).

[3] When a new source is added to the system, a new wrapper has to be implemented (specified) as well.

4 Discussion

The aim of this paper was to present various aspects of a new technology (MQS) which has emerged to solve the problem of efficient and effective retrieval of information from heterogeneous data sources. We have first shown that some approaches exist in related areas, which offer partial solutions and are adequate for special cases. We claim that for developing an MQS, it is possible to build on these, by taking one of them as a starting point and combining it with features from others. For example, database concepts (more specifically those of federated database systems) can be extended with information retrieval concepts (like those of metasearch engine). How this combination is actually done, depends on the requirements to MQS which we presented and classified in the second part of the paper. Our classification can serve as a starting point for developing an MQS and is a framework for comparing different implementations.

Commercial products for accessing heterogeneous data are available today (DataJoiner, Cohera, MIRACLE, EDI/S). In our classification in section 2, they fit best the federated database approach, but none of them offers full database functionality. They give an integrated view over data (mostly stored in relational databases) but do not allow the flexible way to query data as we are aiming at for MQS. [RH98] presents and compares most of these approaches.

Lastly, we want to mention that in order to implement an MQS, one can make use of various existing technologies, including e.g. APIs like ODBC, JDBC [JDB], OLE DB [Bla96] which can be used for implementing wrappers and which offer generalized access to a large class of data sources. For the mediation layer, we mention the query service specification of CORBA [COR98] which provides mechanisms needed for query processing in such an integrated environment.

Acknowledgments

We thank Dimitrios Tombros, Martin Schönhoff and Anca Vaduva for their help during the preparation of this paper. We also thank Ubilab for supporting the work of Ruxandra Domenig.

References

[AKH96] Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the SIMS information mediator. In *Advanced Planning Technology*, AAAI Press Menlo Park CA, 1996.

[AM98] G. Arocena and A. Mendelzon. WebOQL: Restructuring Documents, Databases and Webs. *Proc. of 14th. Intl. Conf. on Data Engineering (ICDE 98), Florida*, 1998.

[BBMS98] M. L. Barja, T. Bratvold, J. Myllymaki, and G. Sonnenberger. Informia: a mediator for integrated access to heterogeneous information sources, http://www.informia.com. *Proceedings of the Conference on Information and Knowledge Management CIKM'98*, 1998.

[Bea97] R. J. Bayardo and W. Bohrer et al. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. *SIGMOD Record*, 1997.

[BGD97] A. Behm, A. Geppert, and K. R. Dittrich. On the migration of relational schemas and data to object-oriented database. In *Proc. 5th International Conference on Re-Technologies for Information Systems, Klagenfurt, Austria*, 1997.

[Bla96] J. Blakeley. Data Access for the Masses through OLE DB. *Proc. of SIGMOD, Montreal*, 1996.

[Cea95] M. J. Carey and L. M. Haas et al. Towards heterogeneous multimedia information systems: The Garlic approach. In *Research Issues in Data Engineering*. IEEE Computer Society Press, March 1995.

[Co.97] IBM Co. DB2 DataJoiner: Administration guide and application programming. *IBM Co., San Jose*, 1997.

[COH] Cohera. *http://www.cohera.com/*.

[Coo80] S. Cook. Some more on variant records. Technical report, Queen Mary College, Department of Computer Science, 1980.

[COR98] CORBA Services Book. *http://www.omg.org/corba/sectran1.html*, 1998.

[DD99] K. R. Dittrich and R. Domenig. Towards exploitation of the data universe: Database technology for comprehensive query services. *Third international conference on Business Information Systems*, April 1999.

[DH97] D. Dreilinger and A. E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, July 1997.

[FFK+97] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. STRUDEL: A Web site management system. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2), 1997.

[FLM98] D. Florescu, A. Levy, and A. Mendelson. Database techniques for the World Wide Web: A Survey. *SIGMOD Record*, September 1998.

[Gru93] T. R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2), 1993.

[Hug96] K. Hughes. ORACLE Transport Gateway - Installation and User's Guide for IBM DRDA fro RS/6000. *ORACLE Co.*, 1996.

[Inc97] Information Builders Inc. EDA/SQL Manuals. *Information Builders Inc.*, 1997.

[JDB] The JDBC Database Access API. *http://java.sun.com/products/jdbc*.

[KRR97] B. König-Ries and C. Reck. An architecture for transparent access to semantically heterogeneous information sources. In *Proceedings ot the First International Workshop on Cooperative Information Agents*, Berlin, February 1997.

[LP97] L. Liu and C. Pu. An adaptive object-oriented approach to integration and access of heterogeneous information sources. *Distributed and Parallel Databases*, April 1997.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the twenty-second international Conference on Very Large Data Bases, India*, 1996.

[MJD97] T. Meyer, D. Jonscher, and K. R. Dittrich. Middleware zur Integration geographischer Daten. *INFORMATIK 4:5*, October 1997.

[MMM97] A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web. *Journal of Digital Libraries*, 1(1), 1997.

[PGMW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the 11th International Conference on Data Engineering*, March 1995.

[RH98] F. Rezende and K. Hergula. The Heterogeneity Problem and Middleware Technology: Experiences with and Performance of Database Gateways. *Proc. of the 24th anual international conference on Very Large Databases*, 1998.

[SE97] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, pages 11–14, January–February 1997.

[TRV96] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous databases and the design of Disco. In *ICDCS '96; Proceedings of the 16th International Conference on Distributed Computing Systems; May 27-30, 1996, Hong Kong*, May 1996.

[VGD99] A. Vavouras, S. Gatziu, and K.R. Dittrich. The SIRIUS Approach for Refreshing Data Warehouses Incrementally. In *Proceedings of BTW'99*, March 1999.

[Wie92] G. Wiederhold. Mediators in the architecture of future information systems. *Computer*, 25(3), March 1992.

[ZHKF95] G. Zhou, R. Hull, R. King, and Jean-Claude Franchitti. Supporting data integration and warehousing using H2O. *IEEE Data Engineering Bulletin, Special Issue on Data Warehousing*, 1995.