Unavailable Data Sources in Mediator Based Applications*

Philippe Bonnet
GIE Dyade
655 Avenue de l'Europe
38330 Montbonnot, France
Philippe.Bonnet@dyade.fr

Anthony Tomasic
INRIA Rocquencourt
78153 Le Chesnay, France
Anthony.Tomasic@inria.fr

Abstract

We discuss the problem of unavailable data sources in the context of two mediator based applications. We discuss the limitations of existing system with respect to this problem and describe a novel evaluation model that overcomes these shortcomings.

1 Introduction

Mediator systems are being deployed in various environments to provide query access to heterogeneous data sources. When processing a query, the mediator may have difficulty accessing a data source (due to network or server problems). In such cases the mediator is faced with the problem of *unavailable* data sources.

In this paper, we discuss the problem of unavailable data sources in mediator based applications. We first introduce two applications that we are currently developing. The first application concerns a hospital information system; a mediator accesses data sources located in the different services to provide doctors with information on patients. The second application concerns the access to documentary repositories within a network of public and private institutions; a mediator accesses the data sources located in each institution to answer queries asked through a World Wide Web application. We detail the characteristics of these applications in Section 2. We show that these applications are representative of large classes of applications.

We then discuss, in Section 3, the impact of unavailable data sources on the design of both applications. We illustrate the limitations of classical mediator systems. We give in Section 4 an overview of a novel sequential model of interaction which fits the needs of both applications and overcomes some of the above mentioned shortcomings. We review related work in Section 5. We conclude and give directions for future work in Section 6.

2 Mediator Based Applications

2.1 Basque Health Service

Increase in medical care activity and constraints on health care expenses have lead the San Sebastian hospital complex to adapt its information system. Previously, different information systems coexisted; each service (administration, laboratory, radiology, etc.) had designed and developed an application customized to its own needs. A first improvement in the hospital information system has consisted in developing the CLINIC application which integrates data located in different services of one hospital while maintaining the existing applications. The main objective of the CLINIC application was to integrate data concerning the medical care of a given patient. The CLINIC application was deployed in 1996.

*This work has been done in the context of Dyade, joint R&D venture between Bull and INRIA.

The evolution of the San Sebastian hospital complex has led to new requirements on the information system. The CLINIC application has displayed several limitations with respect to these new requirements. First, the integration of the services data sources is hand-coded in the application; it is thus extremely expensive to integrate new data sources in the application. New data sources are needed due to the evolution of the hospital information system: services from three hospitals are being integrated as well as outlying health centers. Second, only predefined queries programmed in the application can be asked in CLINIC; there is no support for ad-hoc queries. Third, the CLINIC application has been designed for medical specialists. A new goal of the hospital information system is to provide information to primary care doctors (who should be limited to access information concerning their own patients), and hospital managers (who should be able to monitor all activities). These limitations of CLINIC have lead the Basque Hospital Complex to join the MIRO-Web Esprit project [10] to develop, together with a software house, the Basque Health Service (BHS) application [9], an evolution of the CLINIC application based on a mediator infrastructure.

A mediator provides a unique entry point and a uniform view over a set of data sources. Applications can thus access access a uniform representation of the underlying data sources through a declarative query language. Mediators accept queries and transform them into sub-queries that are distributed to data sources.

Each service administrates its own data source concerning patients. Administrative data are located on a AS/400 system. Clinical episodes are stored on ORACLE database systems located in each service (laboratory, radiology, pathology, surgery).

The main integration problem consists in the identification of patients. Each patient has a different identifier in each system, be it a key based on the social security number or a specific code. The solution is to establish for each patient a mapping between his identifiers in the different service data sources. This solution consists in a central mapping table that associates the different identifiers of a patient, each of these identifiers is used in the relations located on the service data sources which store administrative data or clinical episodes related to a patient. Such a solution is necessarily expensive in terms of storage space and maintenance. Each query asked on the mediator involves the mapping table and one or several relations located on the service data sources.

Users of the BHS application are primary care doctors, medical specialists and hospital managers. The user interface allows to consult a predefined form centered on a patient or to formulate ad-hoc queries. The predefined form corresponds to a set of queries concerning all episodes undergone by a given patient. Each of these queries is a join between the central mapping table and one or several relations located on the service data sources. Typical ad-hoc queries are *find all patients with a stay of over x days and a diagnosis y*, or *find all patients with a stay of over x days and are waiting to be operated on*, or *find all patients older than x years-of-age and are in the service y*, or *find the medicines taken by patients with diagnosis x*, or *find the average number of X-rays requested by patients of service x*.

2.2 Redoc

The Redoc project has been initiated in 1993 in order to develop and improve access to the numerous documentation sources spread around the city of Grenoble, France. One of the aspects of the Redoc project consists in providing a World Wide Web (WWW) application for accessing document repositories within a network of public and private institutions. We have started a collaboration with the Redoc project to deploy this application based on the Disco mediator infrastructure [15]. Again the mediator provides uniform access to a set of data sources through a uniform declarative query language.

Each institution participating in the Redoc network manages its own document repository. A document repository contains bibliographical notes on books, proceedings, journals, thesis or CDs. They are stored in relational databases or in structured files. All of these repositories are already accessible from the WWW via boolean search engines (only the responses that match a query are

returned). In a first step, we are developing a mediator based application accessing the boolean search engines. In a second step, we are planning to integrate directly some of the repositories located on relational databases.

There are approximately forty document repositories. Each of them contains between a thousand and several hundred thousand records. From a technical perspective, it would be possible to construct a data warehouse storing the bibliographical notes from all document repositories. Such a data warehouse would not occupy more than a couple of gigabytes. However this warehouse cannot be constructed for legal and political reasons.

All data sources export similar schema concerning bibliographical notes. The mediator provides views for families of document repositories, for instance environment repositories, legal repositories, or computer science repositories. Such views are union queries over a predefined set of data sources. It is also possible for the user to indicate a set of data sources and to formulate a union query over these data sources. A typical query is *find the bibliographical notes of all articles written in 1994 by Smith in all computer science related repositories*, or *find the bibliographical notes of all books whose title contains glacier in all repositories related to environment and mountain.*

2.3 Discussion

We have identified a set of parameters that characterize mediator-based applications. The characteristics of the network, the number of data sources, the type of data sources and the type of queries that differentiate mediator-based applications[1]. Table 1 summarizes the characteristics of the BHS and Redoc applications. The first stage of the BHS application integrates five data sources on top of a local-area network. In the second stage approximately 20 data sources distributed over a metropolitan-area network will be integrated. These data sources are Oracle relational servers and AS/400 files. The queries that are asked involve the central mapping table and one or several relations located in the service data sources: the shape of their query graph[2] is thus a star. The Redoc application integrates around forty data sources on top of a wide-area network; these data sources are boolean search engines located on WWW servers. The queries that are asked are union queries.

Parameter	BHS (1st stage)	BHS (2nd stage)	Redoc
Network characteristics	LAN	MAN	WAN
Number of Data Sources	5	20	40
Type of Data Sources	files and relational	files and relational	boolean search engines
Type of Queries	star	star	union

Table 1: Characteristics of the BHS and Redoc applications.

3 Unavailable Data Sources

When processing a query, the mediator may have difficulty accessing a data source due to network or server problems. Networks become congested and servers are overloaded. Both can cease to function due to power loss, administrative operations, etc. In such cases the mediator is faced with the problem of *unavailable* data sources.

In this section, we describe the problem of unavailable data sources in the context of the two applications described above. We discuss the influence of the application parameters on the

[1] Additional parameters such as the approximate size of answers or the average number of relations involved in queries could be used. Their value is however difficult to evaluate

[2] In a query graph, vertices are base relations and each edge represents a joining condition between two relations.

problem of unavailable data sources. We then discuss the limitations of existing mediator systems with respect to this problem.

3.1 Basque Health System

In CLINIC the application and the data sources are connected via a local-area network. Network congestion is thus not an issue. The problem of unavailable data sources is here essentially due to server outages. Each service administers its own data source: the AS/400 or Oracle systems can be shut down by their administrator without notification to the other services or to the mediator. When contacting a data source which has been shutdown for administration, maintenance or repair, the mediator is faced with the problem of unavailable data sources. Server outages have not been frequent in the CLINIC application and the problem of unavailable data sources has not been specifically treated. Actually, the application returns an error or simply hangs in case it contacts a data source which is unavailable.

We consider that the problem of unavailable data sources will grow with the evolutions of the hospital information system (i.e. in the second stage of the BHS application). The mediator infrastructure used in the BHS application allows to integrate new data sources (CLINIC was limited to 5 data sources). The probability of server outage will thus increase. Moreover, some data sources will be located outside of the hospital local-area network thus introducing the problem of unpredictable network congestions.

Two types of queries have been identified for the BHS application, first, fixed queries which access episodes undergone by a given patient, and second, ad-hoc queries concerning hospital management or medical care analysis. If a service data source is unavailable when accessing episodes undergone by a patient, the doctor wants all available episodes concerning this patient. The consultation of the patient record should not be blocked because one or several data sources are unavailable. It can be the case that the doctor is interested in the radiology episodes and that the radiology data source is unavailable. In this case, the doctor cannot obtain the relevant information until the radiology data source is up again. The information that the radiology data source is actually unavailable is important for her; she can thus communicate with the administrator of the radiology data source and discuss her problem. In case the doctor is mainly interested in radiology episodes while the surgery data source is unavailable, the doctor does not want to be blocked until the surgery data source is available again to obtain information from the radiology data source.

If a service data source is unavailable when an ad-hoc query is asked, the user wants to know which sources were unavailable, i.e. which sources are temporarily shut down. The user then decides either to ask alternative queries while the service data source is shut down or to wait until the data source is available again to obtain the complete answer to his query.

3.2 Redoc

In the Redoc application the mediator is connected to about forty data sources via a wide-area network. When developing the application we have experienced congestion in the network. Data sources are difficult to contact at some periods of the day, in particular in the afternoon. It also happens that after a connection has been established, the transfer of data starts but is interrupted after a while. In summary, data sources are frequently unavailable (several times a day), for a short period of time (a couple of minutes).

The number of data sources is important and there are some periods of the day where the availability of data sources is low. In these periods, the probability that all data sources involved in a query are available simultaneously is low.

The queries that are processed by the mediator are union queries over some data sources. In case some data sources are unavailable, the user should be notified that the complete answer could not be produced. This notification should be extended with the list of unavailable data sources. Most importantly, the system should allow users to access the result of a union query similar to the

original query but only concerning the data sources which were available. In addition, the system should allow users to access the complete answer efficiently.

3.3 Discussion

3.3.1 Application Characteristics and Unavailable Data Sources

We have in Section 2.3 identified some parameters characterizing mediator based applications. We discuss here the influence of these parameters on the problem of unavailable data sources.

The problem of unavailable data sources is due to network congestions or server outages. The type of network and the number of data sources involved in the system are thus the important parameters from the point of view of the unavailable data sources problem. Probability of network congestions are much more important in a wide-area network than in a local area network. Probability of server outages increase with the number of data sources involved in the system.

The description of unavailable sources in the BHS and Redoc applications show that network congestion is a dominant parameter. In the Redoc application, network congestion is a frequent cause of unavailable data sources; server outages are rather rare events.

3.3.2 Limitations of Existing Systems

Existing mediator systems fail ungracefully in the presence of unavailable data sources. When processing a query q, existing systems either silently ignore missing data or generate an error notification n. In either case, to obtain the complete answer, the query must be resubmitted to the system and reprocessed from scratch. If some sources are unavailable, the system will again generate an error and again the query must be resubmitted. The complete answer a to a query will be generated only when all data sources are available simultaneously. Thus, we can model the sequential interaction between the application program and the mediator as the following sequence of steps: $q, n, q, n, \ldots\ q, n, q, a$. We call this sequential model of interaction a *classical evaluation model*.

Such a classical evaluation model presents the following problems. First, the error notification which is returned only indicates that a problem occurred when processing the query. The notification sent back to the application program should contain more specific information, e.g. the list of sources that were unavailable and the list of sources that were available. Second, the complete answer can only be returned if all data sources involved in the query are available simultaneously. This is a major constraint on the availability of complete answers. We have seen that in the Redoc application, the probability that all data sources are available simultaneously is low at certain periods of the day. Third, the query is resubmitted several times to obtain the complete answer and every time it is reprocessed from scratch. Obviously some work could be saved and reused between consecutive processing of the same query in order to reduce total processing time. Finally, the lapse of time between first submitting the query and obtaining the complete answer may be quite long. The application program might be interested in obtaining data related to its query before the complete answer is returned. In Redoc, the system could allow users to obtain a subset of the answers in presence of unavailable data sources, i.e. the union performed over all available data sources. In BHS, the system could allow also users asking ad-hoc queries to access data if the complete answer to their query cannot be computed.

4 Incremental Query Processing

We have described [1] [7] [6] [5] a novel evaluation model which accounts for unavailable data sources.

When a query is evaluated, the system first determines which sources are available or unavailable. If all sources are available, then the complete answer is produced in the classical way. If at

least one data source is unavailable, the complete answer cannot be produced. In this latter case the execution phase proceeds via a second pass on the plan. This phase *materializes* some available parts of the plan. The temporary relations that are materialized constitute the mediator state.

The mediator then uses its state to construct an *incremental query* i which is equivalent to the original query but cheaper to evaluate. Once the incremental query is constructed, the mediator returns a notification to the application program. This notification allows the application program to access information such as the list of available data sources or the list of unavailable data sources; it also allows the application program to retrieve the incremental query. The application program submits the incremental query to the mediator in order to get the complete answer. An example of a sequence for this model of interaction is $q, n_1, i_1, n_2, i_2, \ldots, n_k, i_k, a$. A different incremental query is used, in general, in each step of the sequence because the mediator makes incremental progress towards the complete answer a depending on the sources that are available at each step. When the incremental query is used, the mediator can return the complete answer even if all data sources are not available simultaneously.

Incremental queries save work; in addition, the mediator state contains information of interest for the user. The application program can extract information from the mediator's state by submitting a secondary query, called a *parachute query* ρ. The answer to a parachute query, called a *parachute answer* α, can be computed given enough information in the mediator state; it can then be displayed to the user.

References [6] and [5] present algorithms for the evaluation of queries in the presence of unavailable data sources, the construction of incremental queries and the construction of parachute queries.

We give, in the rest of the section examples of interactions including incremental queries and parachute queries for the two applications presented above.

4.1 Basque Health System

The schema contains a relation identifying patients (*patient*), a relation associating the local identifiers of patients (*map*), and two relations identifying medical treatments (*surgery* and *radiology*). The data from the *patient*, *surgery* and *radiology* relations are located respectively on the services data sources. The *map* relation is located on the mediator. (We assume there is no semantic heterogeneity between data sources; this problem is tackled in [8] for instance.)

A typical query that can be asked to the mediator is *find all information concerning patients with a stay of over 10 days and are waiting to be operated on.*

```
SELECT PATIENT.NAME, PATIENT.ROOM, RADIOLOGY.ID, RADIOLOGY.COMMENT
FROM PATIENT, MAP, SURGERY, RADIOLOGY
WHERE PATIENT.ID = MAP.PID      AND PATIENT.STAY > 10
  AND MAP.SID    = SURGERY.ID   AND SURGERY.STATUS = 'WAITING'
  AND MAP.RID    = RADIOLOGY.ID;
```

In case the radiology data source is unavailable, a classical system would return an error when processing this query. A system implementing an incremental evaluation model would first detect that this data source is unavailable and then materialize data obtained from the available data sources. Here a temporary relation X1 is created to store the result of the join between the patient, map and surgery relations. Exactly, X1 results from the following statement:

```
INSERT INTO X1
SELECT *
FROM PATIENT, MAP, SURGERY
WHERE PATIENT.ID = MAP.PID      AND PATIENT.STAY > 10
  AND MAP.SID    = SURGERY.ID   AND SURGERY.STATUS = 'WAITING';
```

A notification is returned to the user. This notification allows to retrieve information stored in the mediator state such as the list of data sources that were unavailable. The mediator constructs an incremental query that the user can obtain and later re-submit to efficiently obtain the complete answer. In this example the generated incremental query is:

```
SELECT X1.NAME, X1.ROOM, RADIOLOGY.ID, RADIOLOGY.COMMENT
FROM X1, RADIOLOGY
WHERE X1.RID = RADIOLOGY.ID;
```

The user may ask parachute queries to extract data from the mediator state. Here a relevant parachute query could be *find the name and room number of patients with a stay of over 10 days and are waiting to be operated on.*

```
SELECT PATIENT.NAME, PATIENT.ROOM
FROM PATIENT, MAP, SURGERY
WHERE PATIENT.ID = MAP.PID      AND PATIENT.STAY > 10
  AND MAP.SID    = SURGERY.ID   AND SURGERY.STATUS = 'WAITING';
```

This parachute query can be submitted to the mediator and evaluated on X1 using a query/subquery matching algorithm.

4.2 Redoc

Each document repository exports one relation. All exported relations have a similar schema. They all export the title of the document, its authors, its date. Some attributes are specific to a relation for instance ISBN number for books, summary or keywords for some repositories. The mediator integrates all these schema and provides views for families of document repositories, e.g. the view *cs_repositories* is defined for computer science repositories.

A typical query that can be asked to the mediator is *find the bibliographical notes of all articles written in 1994 by Smith in all computer science related repositories*

```
SELECT DOC.AUTHORS, DOC.TITLE, DOC.BOOKTITLE, DOC.ADDRESS
FROM CS_REPOSITORIES DOC
WHERE DOC.AUTHORS LIKE '%SMITH%' AND DOC.DATE = 1994;
```

This query over the cs_repositories relation is actually a union query over the 4 computer science related repositories present in the Redoc application. If one of these repository is unavailable, say relation cs_repository1, when this query is evaluated then a temporary relation, X2 , is created to store the result of the union between the three available relations. A notification is returned to the user. This notification allows to retrieve the list of data sources that were unavailable. The mediator constructs the incremental query. It is the union between the temporary relation that has been materialized and the relation that was unavailable. The user can obtain this incremental query and later re-submit it to efficiently obtain the complete answer.

```
SELECT DOC.AUTHORS, DOC.TITLE, DOC.BOOKTITLE, DOC.ADDRESS
FROM X2 DOC
UNION
SELECT DOC.AUTHORS, DOC.TITLE, DOC.BOOKTITLE, DOC.ADDRESS
FROM CS_REPOSITORY1 DOC
WHERE DOC.AUTHORS LIKE '%SMITH%' AND DOC.DATE = 1994;
```

The user can extract the data obtained from the available data sources and materialized in the mediator state using a parachute query. This parachute query is generated by the system[3]

```
SELECT DOC.AUTHORS, DOC.TITLE, DOC.BOOKTITLE, DOC.ADDRESS
FROM X2 DOC;
```

[3]Such a parachute query is generated using an extension to union queries of the algorithm presented in [5].

5 Related Work

There do not exist many descriptions of mediator based applications. References [3], [4] and [2] discuss applications example and demonstrators based on the MINT, Garlic and TSIMMIS mediator prototypes respectively. References [12] and [14] describe applications based on commercial systems: IBM DataJoiner and Sybase OmniConnect respectively. Reference [11] describes Gloss: a system tailored to the needs of applications based on text databases, such as Redoc .

An alternative to our techniques in dealing with unavailable data sources is *replication*. Replication can increase the availability of all data sources to the point that queries almost always execute. Replication suffers from economic disadvantages – first, the hardware for each data source must be replicated at a different physical site (to avoid communication failures) and second the software has higher cost because replication impacts its complexity. Replication also (usually) impacts update performance. In addition, in an environment with autonomous data sources, replication may not be possible simply because the data source forbids it. However, note that incremental query processing is completely compatible with replication – in the case that a data source is replicated, the probability that it will be unavailable is simply smaller.

Multiplex [13] tackles the issue of unavailable data sources in a multi-database system and APPROXIMATE [16] tackles the issue of unavailable data in a distributed database. Both systems propose an approach based on approximate query processing. In presence of unavailable data, the system returns an approximate answer which is defined in terms of subsets and supersets sandwiching the exact answer.

Multiplex uses the notions of subview and superview to define the approximate answer. A view V1 is a subview of a view V2 if it is obtained as a combination of selections and projections of V2; V2 is then a superview of V1. These notions can be a basis to define the relationship between a query and its associated parachute queries. APPROXIMATE uses semantic information concerning the contents of the database for the initial approximation. In our context, we do not use any semantic information concerning the data sources. None of these system produce an incremental query for accessing efficiently the complete answer.

6 Conclusion

We have described two mediator based applications that we are currently developing. These applications are exposed to the problem of unavailable data sources. We have identified network congestion as the main reason for data sources unavailability. We have briefly described the incremental evaluation model that allows to overcome the limitations of existing systems with respect to the problem of unavailable data sources.

We have studied in [5] the performance improvements of our approach. Once the applications are developed we will get usability feedback on the evaluation model that we have described. We are particularly interested in evaluating the impact of parachute queries and in developing tools to aid users formulate parachute queries.

Acknowledgments The authors wish to thank Jose Manuel Muñoz for his comments concerning the BHS application.

References

[1] L. Amsaleg, Ph. Bonnet, M. J. Franklin, A. Tomasic, and T. Urhan. Improving responsiveness for wide-area data access. *Bulletin of the Technical Committee on Data Engineering*, 20(3):3–11, 1997.

[2] Garlic Group at IBM Almaden. Garlic project home page. http://www.almaden.ibm.com/cs/garlic/homepage.html.

[3] COIN Group at MIT Sloan School of Management. MINT home page. http://context.mit.edu/ coin/demos/.

[4] Tsimmis Group at Stanford. TSIMMIS home page. http://www-db.stanford.edu/tsimmis/tsimmis.html.

[5] Ph. Bonnet and A. Tomasic. Incremental query processing in the presence of unavailable data sources, 1998. Submitted for publication.

[6] Ph. Bonnet and A. Tomasic. Parachute queries in the presence of unavailable data sources. Technical Report RR-3429, INRIA, 1998.

[7] Ph. Bonnet and A. Tomasic. Partial answers for unavailable data sources. In *Proceedings of the Conference on Flexible Query Answering Systems*, Roskilde, Denmark, 1998.

[8] S. Bressan and C.H. Goh. Answering queries in context. In *Proceedings of the International Conference on Flexible Query Answering Systems, FQAS'98*, Roskilde, Denmark, 1998.

[9] I. Dorronsoro, I. Portillo, J.M. Munoz, and I. Mokoroa. Hospital application requirements definition (d7a.1). MIRO-WEB EP 25208 Deliverable.

[10] P. Fankhauser, G. Gardarin, M. Lopez, J. Munoz, and A. Tomasic. Experiences in federated databases: From IRO-DB to MIRO-Web. In *Proceedings of the 24th International Conference on Very Large Data Bases (Industrial Session)*, New York, NY, USA, 1998.

[11] L. Gravano, H. Garcia Molina, and A. Tomasic. The effectiveness of gloss for the text database discovery problem. 1994. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, Minneapolis, USA, 1994.

[12] IBM. DB2 DataJoiner case studies. http://www.software.ibm.com/data/datajoiner/casestudy.html.

[13] A. Motro. Multiplex: A formal model for multidatabases and its implementation. Technical Report ISSE-TR-95-103, George Mason University, 1995.

[14] Sybase. OmniConnect product description. http://www.sybase.com:80/products/entcon/omni.html.

[15] A. Tomasic, R. Amouroux, Ph. Bonnet, O. Kapitskaia, H. Naacke, and L. Raschid. The distributed information search component (DISCO) and the World-Wide Web. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, USA, 1997.

[16] S. V. Vrbsky and J. W. S. Liu. APPROXIMATE: A query processor that produces monotonically improving approximate answers. *Transactions on Knowledge and Data Engineering*, 5(6):1056–1068, December 1993.