The Distributed Information Search Component (Disco) and the World Wide Web *

Anthony Tomasic
INRIA Rocquencourt
Rocquencourt, France
Anthony.Tomasic@inria.fr

Rémy Amouroux
Bull
Grenoble, France
Remy.Amouroux@dyade.fr

Philippe Bonnet
Bull
Grenoble, France
Philippe.Bonnet@dyade.fr

Olga Kapitskaia
INRIA Rocquencourt
Rocquencourt, France
Olga.Kapitskaia@inria.fr

Hubert Naacke
INRIA Rocquencourt
Rocquencourt, France
Hubert.Naacke@inria.fr

Louiqa Raschid
University of Maryland
College Park, Maryland, USA
louiqa@umiacs.umd.edu

Abstract

The Distributed Information Search COmponent (Disco) is a prototype heterogeneous distributed database that accesses underlying data sources. The Disco prototype currently focuses on three central research problems in the context of these systems. First, since the capabilities of each data source is different, transforming queries into subqueries on data source is difficult. We call this problem the *weak data source* problem. Second, since each data source performs operations in a generally unique way, the cost for performing an operation may vary radically from one wrapper to another. We call this problem the *radical cost* problem. Finally, existing systems behave rudely when attempting to access an unavailable data source. We call this problem the *ungraceful failure* problem.

Disco copes with these problems. For the weak data source problem, the database implementor defines precisely the capabilities of each data source. For the radical cost problem, the database implementor (optionally) defines cost information for some of the operations of a data source. The mediator uses this cost information to improve its cost model. To deal with ungraceful failures, queries return partial answers. A partial answer contains the part of the final answer to the query that was produced by the available data sources. The current working prototype of Disco contains implementations of these solutions and operations over a collection of wrappers that access information both in files and on the World Wide Web.

1 Introduction

The Disco (Distributed Information Search COmponents) [1, 2] project is developing a research prototype of components for searching and integrating information over distributed heterogeneous data sources. The data sources can be databases, files, dedicated data servers (e.g. a multimedia server or an information retrieval engine), or HTML pages. Thus, data can be structured, semi-structured or unstructured. The target applications of Disco are those of Internet and Intranet which typically require integration of large numbers of data sources. The main objective of Disco is to provide uniform and optimized access to the underlying data sources using a common declarative query language.

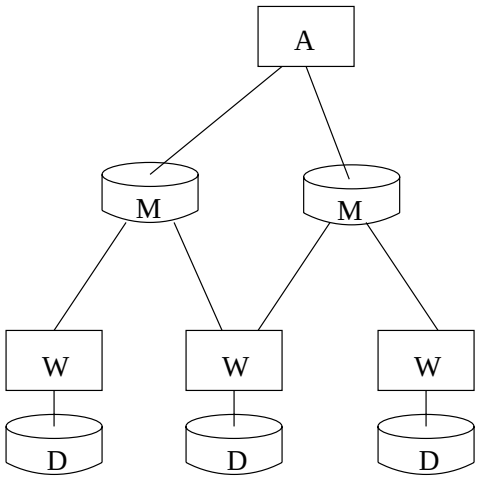

Figure 1: Disco architecture. Boxes represent stateless components, and disks components with state. A stands for application, M for mediator, W for wrapper, and D for data source. Arcs represent exchange of queries and answers.

To scale up to large numbers of data sources, Disco adopts a mediator [3] distributed architecture of specialized components (common to many existing projects, e.g. [4, 5, 6]) consisting of applications, mediators, wrappers and data sources, as shown in Figure 1. End users interact with applications (A) written by application programmers. Applications access a uniform representation of the underlying sources through a uniform (SQL-like) declarative query language. *Mediators* (M) encapsulate a representation of multiple data sources for this query language. They typically resolve conflicts involving the dissimilar representation of knowledge of different data models and schema, and conflicts due to the mismatch in querying power of each data

*This work is done within the Groupement d'Intérêt Economique Dyade, a consortium established by Bull and INRIA, France.

source. This architecture permits mediators to be developed independently and to be combined, providing a mechanism to deal with the complexity introduced by a large number of data sources. To permit multiple data sources to be accessed in a uniform way, mediators accept queries and transform them into subqueries that are distributed to data sources. In DISCO subqueries are expressed in an algebraic language that supports relational operations.

To deal with the heterogeneous nature of data sources, *wrappers* (W) give a structured view of the data source and transform subqueries from the mediator to the particular language of the *data source* (D). A wrapper supports the functionality of transforming queries appropriate to the particular data source, and reformatting answers (data) appropriate to each mediator. The database implementor (DBI) writes wrappers for each type of data source.

The DISCO prototype examines three central research problems in the context of this architecture. First, since the capabilities of each data source is different, some data sources may not support the entire algebra of operations used by the mediator. This lack of support introduces problems into the transformation of the query into subqueries. We call this problem the *weak data source* problem. Second, since each wrapper implements the algebraic operations for a data source in a generally unique way, the cost for performing an operation may vary radically from one wrapper to another. Thus, the construction of the best transformation of queries into subqueries, from the performance point of view, is also not straightforward. We call this problem the *radical cost* problem. Finally, since data sources may be unavailable during query processing, it may be impossible to produce the answer to a query. Generally existing systems behave rudely by either stalling while waiting for the data source, or generate an error, or silently ignore the unavailable data source. We call this problem the *ungraceful failure* problem.

The design of DISCO provides special features to deal with these three problems. To deal with the weak data source problem, the DBI *defines* precisely the capabilities of each data source [7]. That is, when a new wrapper is constructed, the DBI chooses a subset of the algebraic language that the wrapper supports. When the mediator registers the wrapper (before query processing), the wrapper transmits a description of the subset that it supports to the mediator. The mediator incorporates this information automatically in the query transformation process. To deal with the radical cost problem, the DBI can *optionally* choose to define cost information for some or all of the algebraic operations supported by the wrapper [8]. The cost information is also transmitted to the mediator when the wrapper is registered. The wrapper specific cost information overrides the general cost information used by the mediator to produce a more accurate cost model. The cost model is used by the DISCO optimizer to produce the best possible query processing plan. To deal with the ungraceful failure, queries return *partial answers* [9]. A partial answer contains the part of the final answer to the query that was produced by the available data sources. A partial answer also contains a query representing the finished and unfinished parts of the answer. When the unavailable data sources become available, the partial answer can be resubmitted as a new query to obtain the final answer to the original query.

2 Query Language

Consider a system that contains two data sources `r0` and `r1`. Suppose the `r0` data source contains a person relation with a person Mary whose salary is 200, and `r1` contains a person relation with a person Sam whose salary is 50. A mediator models `r0` and `r1` as extents `person0` and `person1`, of type `Person`.

To access the data sources, users express queries in the DISCO query language, a simple relational-like language. For example, the query

```
select x.name,y.name
from x in person0, y in person1
where x.salary > y.salary
```

constructs a bag of the names of the persons from `r0` who have a salary greater than someone in `r1`. The answer to this query is a bag of strings `Bag(("Mary","Sam"))`.

3 Mediator deployment

Interactions between mediators and wrappers occurs in two phases: the *registration* phase and the *query processing* phase. During the *registration* phase, the mediator registers various wrappers. During registration, the wrapper communicates the local schema it supports, the capabilities for query processing, and any specific cost information. The mediator administrator defines a global schema and views to connect the global schema to local schemas. The application is written with respect to the global schema.

During the *query processing* phase, the application issues a query to the mediator, the mediator transforms the query into a plan consisting of subqueries and a composition query. The plan has been optimized with respect to the cost information imported from the wrappers and the plan respects the capabilities of a wrapper. The mediator then executes the plan by issuing the subqueries to the wrappers. The available wrappers process the subqueries by communicating with the associated data source and returning subanswers. If all wrappers are available, the mediator combines the subanswers by using the composition query and returns the answer to the application. The application displays the answer to the user. If some wrappers are unavailable, the mediator returns a partial answer to the application. The application extracts some information from this partial answer and displays the partial information to the user (what information is extracted depends on which wrappers are available).

4 Wrapper interface

For the DBI, DISCO provides a flexible wrapper interface. DISCO interfaces to wrappers at the level of an abstract universal algebraic machine (UAM) of logical operators. When the DBI implements a new wrapper, she chooses a (sub)set of logical operators of the UAM to support. The DBI implements the translation of the logical operators to the underlying source and the reformatting of the answers. She also implements a register call in the wrapper interface. The mediator uses the register call to get the local schema. For data source `r0`, this is

```
interface Person (extent person0) {
  attribute string name;
  attribute integer salary;
}
```

The register call also returns the capabilities For example, for the r0 data source, the sentences

```
scan [ALL]
project [ALL]
select [person0 1 {bind name}]
```

specify that the wrapper accepts the following logical expressions as queries

```
scan(person0)
project([name],0)
project([salary],0)
project([name,salary],0)
select([name=A],0)
```

where 0 is any valid UAM expression for this wrapper and A is a constant. Finally, the register call returns cost information as specified by the DBI. To express costs, the local schema is augmented with cost equations. For example, the following interface

```
interface Person (extent person0) {
  attribute string name;
  attribute integer salary;
  cardinality {count_page = 1,
               count_object = 1,
               object_size = 100}
}
```

describes the size of the person0 extent as consisting of 1 (4K byte) page, 1 object, with an average object size of 100 bytes.

5 Partial Answers

During query processing, the mediator handles unavailable data sources. As each wrapper is contacted for issuing a subquery, the wrapper communication layer will either return a connection to the wrapper or it will return an error. If an error is returned, the mediator *continues* [10] query processing by contacting the remaining available wrappers and processing as much of the query plan as possible. The mediator then returns to the application a partial answer – an object that contains the partially evaluated plan. The application then invokes various methods on this object. One method returns the list of sources that were available during the processing of this partial answer. Another method returns a new query. It can be resubmitted, and if the previously unavailable data sources are now available, the answer to the original query will be returned. The new query will contact only the previously unavailable data sources to complete processing of the query. If unavailable data sources still exist, another partial answer is returned.

6 Demonstration

The demonstration of the Disco prototype consists of an application, a mediator, three wrappers, and three data sources. The application is a Java applet that issues queries to the mediator and displays answers. The mediator is a resource executing under the Jigsaw HTTP server software [11]. The mediator, also written in Java, accepts queries from the application and issues subqueries to the three wrappers. The wrappers are Java Remote Method Invocation (RMI) servers that contact data sources. Two data sources are files, and the third is a WWW site. Thus, in response to a subquery, the wrappers for the file data sources read the associated files. The wrapper associated with the WWW reads the available HTML files, parses them and generates the appropriate answer. Communication between the application and the mediator is accomplished by the Jigsaw applet protocol (*not* CGI), communication between the mediator and wrappers is accomplished using RMI, and the communication between one wrapper and the WWW site is via HTTP.

To demonstrate the functionalities of Disco, we vary the capabilities of the wrappers, the values in the exported cost information, and availability of the wrappers for query processing. Each change produces a corresponding change in behavior in the mediator. In addition, changes in the availablity of wrappers produces a corresponding change in the behavior of the application.

References

[1] A. Tomasic, L. Raschid, and P. Valduriez, "Scaling heterogeneous databases and the design of disco," tech. rep., number 2704, INRIA, Rocquencourt, France, 1995. Extended version of ICDCS '96 paper.

[2] A. Tomasic, L. Raschid, and P. Valduriez, "Scaling heterogeneous databases and the design of disco," *Proceedings of the International Conference on Distributed Computing Systems*, pp. 449–457, 1996.

[3] G. Wiederhold, "Intelligent integration of information," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1993.

[4] A. Dogac *et al.*, "METU interoperable database system," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.

[5] M. Tork Roth *et al.*, "The garlic project," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.

[6] D. Quass *et al.*, "LORE: A lightweight object repository for semistructured data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.

[7] O. Kapitskaia, A. Tomasic, and P. Valduriez, "Dealing with discrepancies in wrapper functionality," tech. rep., INRIA, Rocquencourt, France, 1997. In preparation.

[8] H. Naacke, G. Gardarin, and A. Tomasic, "Leveraging mediator cost models with heterogeneous data sources," tech. rep., INRIA, Rocquencourt, France, 1997. In preparation.

[9] P. Bonnet and A. Tomasic, "Partial answers for unavailable data sources," tech. rep., INRIA, Grenoble, France, 1997. In preparation.

[10] L. Amsaleg, M. J. Franklin, A. Tomasic, and T. Urhan, "Scrambling query plans to cope with unexpected delays," in *International Conference on Parallel and Distribution Information Systems (PDIS)*, (Miami Beach, Florida), 1996.

[11] "Jigsaw HTTP server software and related activity." http://www.w3.org/pub/WWW/Jigsaw/Activity.html.