in Proceedings of the Fifth International Conference and Exhibition on the Practical Applications of Prolog

Overview of a Prolog Implementation of the COntext INterchange Mediator

Stephane Bressan, Kofi Fynn, Cheng Hian Goh,
Stuart E. Madnick, Tito Pena, Michael D. Siegel

Sloan WP# 3980 CISL WP# 97-13
March 1997

The Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02142

Overview of a Prolog Implementation of the COntext INterchange Mediator *

[In Proceedings of The Fifth International Conference and Exhibition

on The Practical Applications of Prolog.]

Stéphane Bressan Kofi Fynn Cheng Hian Goh† Stuart E. Madnick
Tito Pena Michael D. Siegel
Sloan School of Management
Massachusetts Institute of Technology
Email: context@mit.edu

March 18, 1997

Abstract

The *Context Interchange* strategy [5; 12] presents a novel solution to mediated data access in which semantic conflicts among heterogeneous systems are automatically detected and reconciled by a *Context Mediator* using the *contexts* associated with the systems involved. We have implemented these mediation services using the *ECLiPSe*[1] platform and have deployed them in several applications giving access to on-line database and web services.

Keywords: Prolog, context, heterogeneous databases, logic and databases, mediators, semantic interoperability.

1 Introduction

In the last few years the number of datasources (traditional database systems, data feeds, and applications providing structured or semi-structured data) has increased dramatically. This has been spurred on by various factors. Examples of these are, ease of access to the Internet (which is emerging as the de facto global information infrastructure) and the rise of new organizational forms (e.g. adhocracies and networked organizations) which mandate new ways of sharing and managing information. Also, advances in networking and telecommunications have increased physical connectivity (the ability to exchange bits and bytes) amongst disparate datasources and receivers.

*This work is supported in part by ARPA and USAF/Rome Laboratory under contract F30602-93-C-0160, the International Financial Services Research Center (IFSRC), and the PROductivity From Information Technology (PROFIT) project at MIT.

†Financial support from the National University of Singapore is gratefully acknowledged.

[1]ECLiPSe: The ECRC Constraint Logic Parallel System. More information can be obtained at `http://www.ecrc.de/eclipse/`.

Unfortunately, these new technologies do not provide sufficient logical connectivity (the ability to exchange data meaningfully). Logical connectivity is crucial because users of these sources expect each system to understand requests stated in their own terms, using their own concepts of how the world is structured. As a result, any data integration effort must be capable of reconciling semantic conflicts among sources and receivers. This problem is generally referred to as the need for *semantic interoperability* among distributed datasources.

2 Overview of the COIN Project

The *COntext INterchange* (COIN) strategy seeks to address the problem of *semantic interoperability* by consolidating distributed datasources and providing a unified view to them. COIN technology presents all datasources as SQL databases by providing generic wrappers for them. The underlying integration strategy, called the COIN model, defines a novel approach for mediated [15] data access in which semantic conflicts among heterogeneous systems are automatically detected and reconciled by the *Context Mediator*. Thus the COIN approach integrates disparate datasources by providing *semantic interoperability* (the ability to exchange data meaningfully) among them.

The COIN Framework. The COIN framework is composed of both a data model and a logical language, COINL, derived from the family of *F-Logic* [9]. The data model and language are used to define the *domain model* of the receiver and datasource and the *contexts* [10] associated with them. The data model contains the definitions for the "types" of information units (called *semantic-types*) that constitute a common vocabulary for capturing the semantics of data in disparate systems. *Contexts*, associated with both information sources and receivers, are collections of statements defining how data should be interpreted and how potential conflicts (differences in the interpretation) should be resolved. Concepts such as *semantic-objects*, *attributes*, *modifiers*, and *conversion functions* define the semantics of data inside and across *contexts*. Together with the deductive and object-oriented features inherited from F-Logic, the COIN data model and COINL constitute an appropriate and expressive framework for representing semantic knowledge and reasoning about semantic heterogeneity.

Context Mediator. The *Context Mediator* is the heart of the COIN project. It is the unit which provides mediation for user queries. Mediation is the process of rewriting queries posed in the receiver's *context* into a set of mediated queries where all potential conflicts are explicitly solved. This process is based on an abduction [8] procedure which determines what information is needed to answer the query and how conflicts should be resolved by using the axioms in the different *contexts* involved. Answers generated by the mediation unit can be both extensional and intensional. Extensional answers correspond to the actual data retrieved from the various sources involved. Intensional answers, on the other hand, provide only a characterization of the extensional answer without actually retrieving data from the datasources. In addition, the mediation process supports queries on the semantics of data which are implicit in the different systems. These are referred to as *knowledge-level queries* as opposed to *data-level queries* which are enquires on the factual data present in the datasources. Finally, integrity knowledge on one source or across sources can be naturally involved in the mediation process to improve the quality and informative content of the mediated queries and ultimately aid in the optimization of the data access.

System Perspective. From a systems perspective, the COIN strategy combines the best features of the *loose-* and *tight-coupling* approaches to *semantic interoperability* [13] among autonomous and heterogeneous systems. Its modular design and implementation funnels the complexity of the system

into manageable chunks, enables sources and receivers to remain loosely-coupled to one another, and sustains an infrastructure for data integration.

This modularity, both in the components and the protocol, also keeps our infrastructure scalable, extensible, and accessible [6]. By *scalability*, we mean that the complexity of creating and administering the mediation services does not increase exponentially with the number of participating sources and receivers. *Extensibility* refers to the ability to incorporate changes into the system in a graceful manner; in particular, local changes do not have adverse effects on other parts of the system. Finally, *accessibility* refers to how the system is perceived by a user in terms of its ease-of-use and flexibility in supporting a variety of queries.

Application Domains. The COIN technology can be applied to a variety of scenarios where information needs to be shared amongst heterogeneous sources and receivers. The need for this novel technology in the integration of disparate datasources can be readily seen in the following examples.

In the realm of financial decision support there exist a number of different datasources (e.g. foreign stock exchanges) which provide information to investors. This information may be presented in different formats and must be interpreted with different rules. For example, sources and receivers make assumptions about scale-factors and currency of monetary figures. Mismatches among these assumptions across sources or inside one source can be critical in the process of financial decision making.

In the domain of manufacturing inventory control, the ability to access design, engineering, manufacturing, and inventory data pertaining to all parts, components, and assemblies is vital to any large manufacturing process. Typically, thousands of contractors play roles and each contractor tends to setup its data in its own individualistic manner. Managers may need to reconcile inputs received from various contractors in order to optimize inventory levels and ensure overall productivity and effectiveness.

Finally, the modern health care enterprise lies at the nexus of several different industries and institutions. Within a single hospital, different departments (eg.internal medicine, medical records, pharmacy, admitting, billing) maintain separate information systems yet must share data in order to ensure high levels of care. Medical centers and local clinics not only collaborate with one another but with State and Federal regulators, insurance companies, and other payer institutions. This sharing requires reconciling differences such as those of procedure codes, medical supplies, classification schemes, and patient records.

3 Mediation: A Scenario

Let us consider the simplified scenario shown in Figure 1. Data on "revenue" and "expenses", for some collection of companies, are available in two autonomously administered datasources, each comprised of a single relation. The data reported in the two sources differ in the currencies and scale-factors of "company financials" (i.e., financial figures pertaining to the companies, which include `revenue` and `expenses`). In Source 1, all "company financials" are reported using the currency shown and a scale-factor of 1; the only exception are those "company financials" that are reported in Japanese Yen (`JPY`) in which case the scale-factor is `1000`. Source 2, on the other hand, reports all "company financials" in US Dollars (`USD`) using a scale-factor of `1`. Suppose a user, in context c_2, is interested in knowing the names and revenues of all profitable companies. This query can be formulated directly on the export schemas of the two sources as follows:

Q1:
```
SELECT r1.cname, r1.revenue FROM r1, r2
WHERE r1.cname = r2.cname AND r1.revenue > r2.expenses;
```

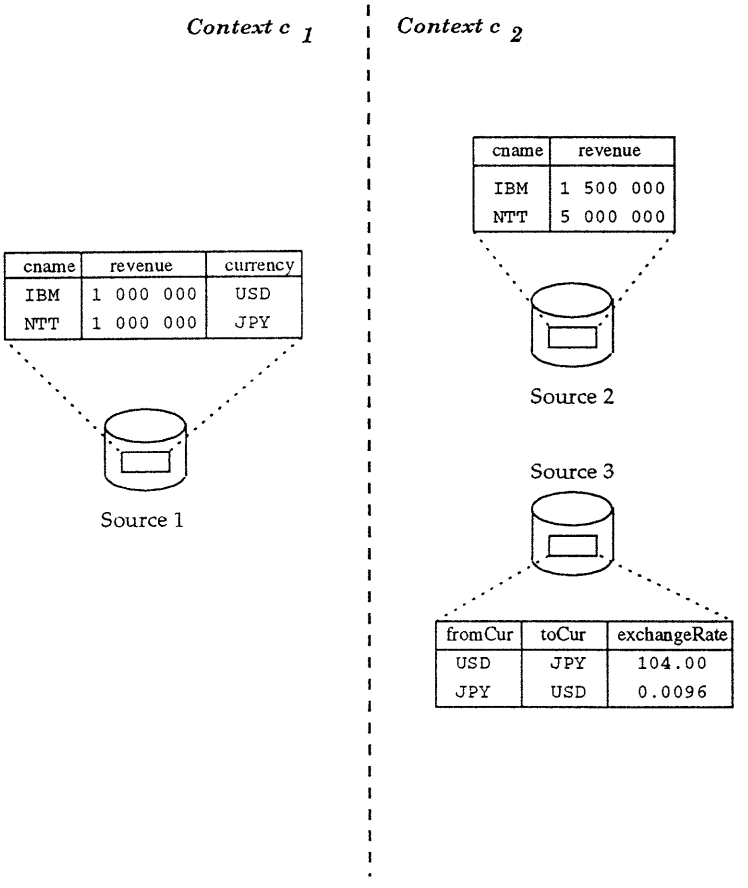

Figure 1: Scenario for the motivational example.

In the absence of any mediation, this query will return the empty answer. This answer returned by executing Q1 is clearly not the correct answer. The reason is because this query does not take into account the fact that the data sources have different *contexts*: i.e., they may embody different assumptions on how information contained therein should be interpreted. Thus the revenue of NTT in US Dollars is $1,000,000 \times 1,000 \times 0.0096 = 9,600,000$ where $1,000$ is the scale-factor, 0.0096 is the exchange rate from JPY to USD, and $1,000,000$ is the revenue in thousands of JPY as reported in r1. This quantity is numerically larger than the expenses ($5,000,000$) reported in r2.

Our Context Mediator will generate a new query which when executed over the data set shown in Figure 1 will return the correct answer. The Context Mediator utilizes meta-data in the form of the domain model, the context axioms for the datasources and receivers, and the elevation axioms (mapping the source export schema to the domain model), to automatically detect and reconcile the conflicts among the assumptions in the different datasources. This meta-data is a COINL representation of the underlying assumptions in each of the datasources and is formly known as a *context.*

The mediated query MQ1 corresponding to Q1 is:

MQ1:
```
SELECT r1.cname, r1.revenue FROM r1, r2
WHERE r1.currency = 'USD' AND r1.cname = r2.cname
AND r1.revenue > r2.expenses
UNION
SELECT r1.cname, r1.revenue * 1000 * r3.rate FROM r1, r2, r3
WHERE r1.currency = 'JPY' AND r1.cname = r2.cname
AND r3.fromCur = r1.currency AND r3.toCur = 'USD'
AND r1.revenue * 1000 * r3.rate > r2.expenses
UNION
SELECT r1.cname, r1.revenue * r3.rate FROM r1, r2, r3
WHERE r1.currency <> 'USD' AND r1.currency <> 'JPY'
AND r3.fromCur = r1.currency AND r3.toCur = 'USD'
AND r1.cname = r2.cname AND r1.revenue * r3.rate > r2.expenses;
```

This mediated query considers all potential conflicts between relations r1 and r2 when comparing values of "revenue" and "expenses" as reported in the two different *contexts*. Moreover, the answers returned may be further transformed so that they conform to the *context* of the receiver. Thus in our example, the revenue of NTT will be reported as 9 600 000 as opposed to 1 000 000.

The three-part query shown above can be interpreted as follows. The first subquery takes into account the tuples for which the revenue is in USD using a scale-factor 1; in this case, there is no conflict. The second subquery handles tuples for which the revenue is in JPY, using a scale-factor of 1000. Finally, the third subquery considers the cases where the currency is neither JPY nor USD and the scale-factor used is 1, in which case only currency conversion is needed. Conversion among different currencies is aided by the ancillary datasource r3 which provides currency conversion rates. This new mediated query, when executed, returns the correct answer consisting of only the tuple <'NTT', 9 600 000>.

4 General Architecture of the COIN System

The feasibility and features of this proposed strategy have been concretely demonstrated in a working system which provides mediated access to both on-line structured databases and semi-structured data sources such as web sites. The infrastructure leverages on the World Wide Web in a number of ways.

First, we rely on the HyperText Transfer Protocol for the physical connectivity among sources and receivers and the different mediation components and services. Second, we employ the HyperText Markup Language and Java for the development of portable user interfaces.

Figure 2 shows the architecture of the COIN system. It consists of three distinct groups of processes.

- *Client Processes* provide the interaction with receivers and route all database requests to the Context Mediator. An example of a client process is the *multi-database browser* [7], which provides a point-and-click interface for formulating queries to multiple sources and for displaying the answers obtained. Specifically, any application program which issues queries to one or more sources can be considered a client process.

- *Server Processes* refer to *database gateways* and *wrappers*. Database gateways provide physical connectivity to databases on a network. The goal is to insulate the Mediator Processes from the idiosyncrasies of different database management systems by providing a uniform protocol for database access as well as a canonical query language (and data model) for formulating the queries. Wrappers, on the other hand, provide richer functionalities by allowing semi-structured documents on the World Wide Web to be queried as if they were relational databases. This is accomplished by defining an *export schema* for each of these web sites and describing how attribute-values can be extracted from a web site using a finite automaton with pattern matching [11].

- *Mediator Processes* refer to the system components which collectively provide the mediation services. These include the Context Mediator (which rewrites a user-query to a mediated query), the Planner/Optimizer (which produces a query evaluation plan based on the mediated query), and the Executioner (which executes the plan by dispatching subqueries to the Server Processes, collating and operating on the intermediary results, and returning the final answer to the Client Process).

Of these three distinct groups of processes, the most relevant to our discussion are the Mediator processes.

5 Implementation of the Mediator Processes

The Mediator processes comprise the Context Mediator, the query planner/optimizer, and the multi-database executioner. In addition, we describe the query and COINL compilers which generate the internal representation of queries and the axioms of the COIN framework, respectively.

COINL-to-Datalog Compiler

The COINL to Datalog compiler takes a COINL program as the source language and produces a Datalog program. COINL is used to describe the domain model, the contexts of receivers and datasources, and integrity constraints over one or more sources. It is a deductive object-oriented language inspired largely by Gulog and somewhat less by F-Logic [9]. The language supports many of the features of the object-oriented paradigm such as objects, parameterized methods, non-monotonic inheritance, classes, object instances, and static and runtime overriding of methods.

Since the Context Mediator evaluates Datalog rules, it is necessary to transform a COINL program into an equivalent set of Datalog rules on which the mediation process can operate. Currently the compiler is stand alone, like gcc or cc. Given an input file the compiler will generate a file with the

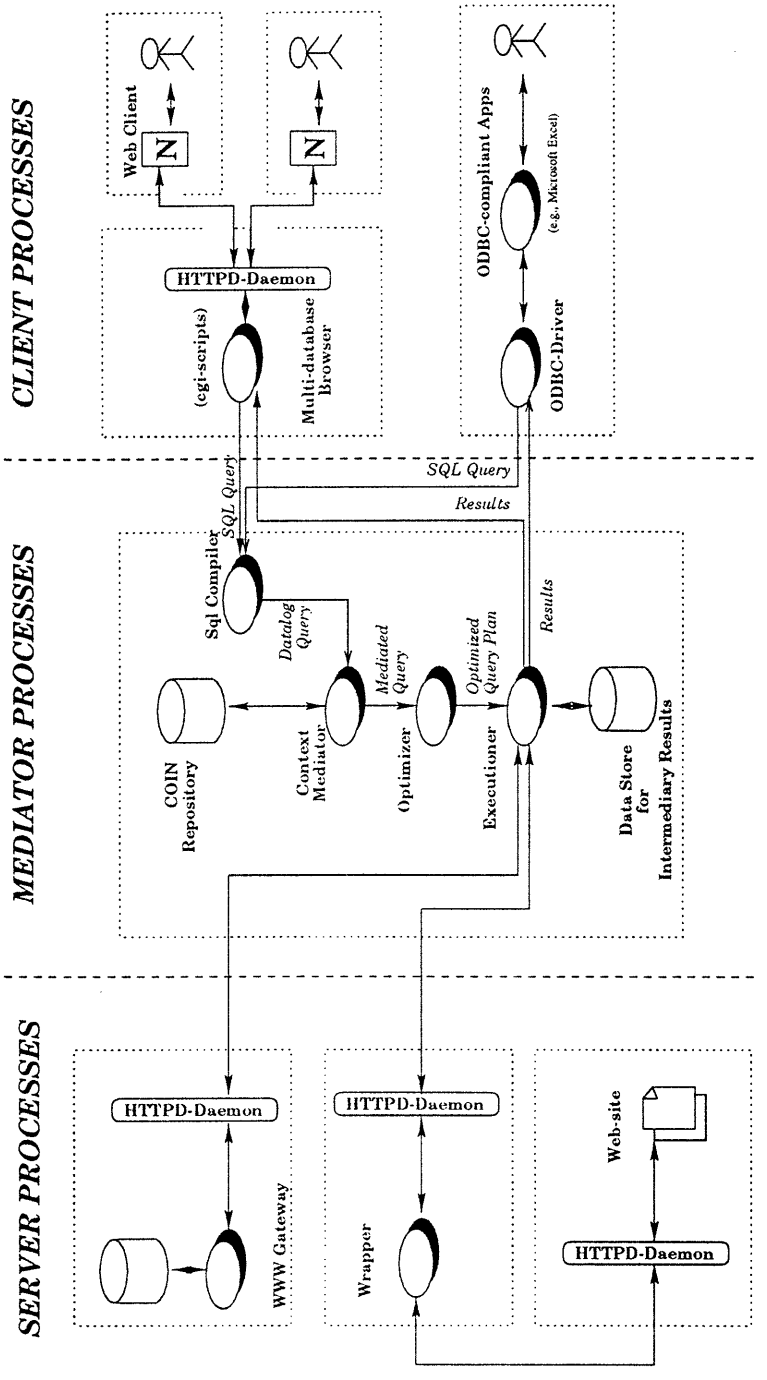

Figure 2: Architectural overview of the Context Interchange Prototype.

Datalog transformations. The generated files are then statically stored in the COIN repository serving the mediation engine at run-time.

A major portion of the transformation rules for the compiler have evolved from two particular works, namely [3] and [1]. At present, there is no existing implementation for either of these proposals. Thus not only is this compiler used in a fielded environment, but it also demonstrates the feasibility of the ideas and theories suggested in [3; 1].

The COINL compiler is implemented in *ECLiPSe* Prolog and uses definite clause grammars (DCG's).

SQL Query Compiler

The query compiler takes a conjunctive SQL query and a reference to the receiver's context as inputs and produces a Datalog query as output. The Datalog query is a direct translation of the SQL query augmented with a mapping of the value of each attribute in the receiver's context to its corresponding object in the semantic domain.

In our example, we had the following SQL query in context c_2:

```
Q1:   SELECT r1.cname, r1.revenue FROM r1, r2
      WHERE r1.cname = r2.cname AND r1.revenue > r2.expenses;
```

Sending this SQL query through the query compiler, the following equivalent Datalog query is generated:

```
answer(CNAME, REVENUE) <-
                r1(R1_CNAME, R1_REVENUE, _),
                r2(R2_CNAME, R2_EXPENSES),
                value(R1_CNAME, c2, CNAME),
                value(R2_CNAME, c2, CNAME),
                value(R1_REVENUE, c2, REVENUE),
                value(R2_EXPENSES, c2, V2),
                REVENUE > V2.
```

Here, r1 and r2 are the relations referenced by the SQL query. The compiler also generates a **value** predicate for each attribute in r1 and r2. This predicate maps the value of each attribute in the receiver's context to its corresponding object in the semantic domain. For example, the predicate `value(R1_CNAME, c2, CNAME)` states that the value of attribute `R1_CNAME` in context c_2 is `CNAME`. This resulting Datalog query is then input to the mediation engine.

Mediation Engine

The mediation mechanism is based on an abduction engine [8]. Given a Datalog query Q and a program P composed of the domain model axioms, the contexts for the different sources and receivers, and the axioms mapping the sources export schemas, it computes a set of abducted queries Q_A such that $\forall q$ where $q \in Q_A$, $P \wedge q \models Q$. In addition, the set of abducted queries Q_A is such that $\forall q$ where $q \in Q_A$, $P \wedge q$ verifies the set of integrity constraints I.

We use constraint handling rules (CHR) [4] to represent the integrity constraints in the system. These are rules whose left hand side is composed of literals corresponding to relations in the export schemas of the sources and whose right hand side is composed of constraint literals such as inequalities, equalities, or comparisons.

For instance a functional dependency in a source relation R corresponding to the logic formula:

$$\forall X, Y : R(X, Y1) \wedge R(X, Y2) \rightarrow Y1 = Y2$$

will have the following CHR representation:

```
ic1 @ R(X,Y1), R(X,Y2) ==> Y1 = Y2
```

The CHR for the constraint stating that the New York Stock Exchange only accepts companies whose earnings before tax is over 2.5 million US Dollars is:

```
ic1 @ pre_tax_earning(Company, Earning) ==> Earning > 2500000
```

where `pre_tax_earning(Company, Earning)` is a relation between a company and its earnings before taxes.
The abduction engine works by constructing an input resolution tree from the initial query Q and the program P. Literals in the residue which correspond to constraints (inequalities, equalities or comparisons) or exported relations are posted to a constraint store. The constraint handling rules are then fired automatically and detect contradictions in the store. These contradictions correspond to integrity constraints which cannot be satisfied. When the abduction process reaches the end of the resolution tree with an empty clause, the posted constraints are collected. The abducted query is the conjunction of the literals in the constraints store. These literals are referred to as abducible. This scheme is a particular application to query mediation of the scheme presented in [14].

The mediated query is the union of all the generated queries. In our example, the mediated query would be as follows:

```
answer(CNAME, REVENUE) <-
              r2(CNAME, EXPENSES),
              r1(CNAME, REVENUE, "USD"),
              REVENUE > EXPENSES.

answer(CNAME, REVENUE1) <-
              r2(CNAME, EXPENSES),
              r1(CNAME, REVENUE, CURRENCY),
              r3(CURRENCY, "USD", EXCHANGERATE),
              V1 is REVENUE * EXCHANGERATE,
              r2(CNAME, EXPENSES),
              r1(CNAME, REVENUE, CURRENCY),
              r3(CURRENCY, "USD", EXCHANGERATE),
              V1 is REVENUE * EXCHANGERATE,
              V1 > EXPENSES,
              REVENUE1 is REVENUE * EXCHANGERATE,
              not CURRENCY = "JPY",
              not CURRENCY = "USD".

answer(CNAME, REVENUE1) <-
              r2(CNAME, EXPENSES),
              r1(CNAME, REVENUE, "JPY"),
              r3("JPY", "USD", EXCHANGERATE),
              SCALEFACTOR1 is 1000 / 1,
```

```
V1 is REVENUE * SCALEFACTOR1,
V2 is V1 * EXCHANGERATE,
V2 > EXPENSES,
SCALEFACTOR2 is 1000 / 1,
V3 is REVENUE * SCALEFACTOR2,
REVENUE1 is V3 * EXCHANGERATE.
```

The constraints may prune the search space and reduce the number of abducted queries. For instance a query asking for companies in the NYSE whose earnings before tax are lower than 20 million in a context where money amounts are expressed in Japanese Yen would be abducted to an empty set since the constraint above guarantees that no such company exists (20 million Japanese Yen is lower than 2.5 million US Dollar).

We are currently involved in efforts to extend our use of integrity constraints and CHR in two directions. First, we are trying to integrate new constraint solvers such as arithmetic solvers to increase the reasoning capabilities of our mediation-engine. Second, we are investigating the use of constraint handling rules for the implementation of the different strategies of semantic optimization as described in [2].

The Planner/Optimizer and Multi-Database Executioner

The Planner/Optimizer takes the set of Datalog queries output by the Mediator and produces a query plan.

It ensures that an executable plan exists which will produce a result that satisfies the initial query. This is necessary because of the potentially limited capabilities of some sources. Typically, some sources may require that some input values be provided for some attributes. Sources might also have restrictions on the operators they can handle. Once the planner ensures that an executable plan exists, it generates a set of constraints on the order in which the different subqueries can be executed. Under these constraints, the optimizer applies standard optimization heuristics to generate the query execution plan.

The executioner executes the query plan. It dispatches the subqueries to the remote sources and combines the results. The executioner uses the Bang file system of *ECLiPSe* as a temporary storage. We take advantage of the multidimensional indexing mechanism of bang to provide an efficient access to temporary data in the case of ad-hoc queries.

Conclusion

In a coordinated effort we are strengthening our web wrapping technology which allows us to extract structured information from semi-structured data. We are also extending the interface technology to offer access to the mediation services from a variety of environments ranging from traditional Web browsers to ODBC applications.

A demonstration version of this environment is available upon request.

References

[1] ABITEBOUL, S., LAUSEN, G., UPHOFF, H., AND WALKER, E. Methods and rules. In *Proceedings of the ACM SIGMOD Conference* (Washington, DC, May 1993), pp. 32–41.

[2] CHAKRAVARTHY, U. S., GRANT, J., AND MINKER, J. Logic-based approach to semantic query optimization. *ACM Trans. on Database Sys. 15*, 2 (June 1990), 162.

[3] DOBBIE, G., AND TOPOR, R. On the declarative and procedural semantics of deductive object-oriented systems. *Journal of Intelligent Information Systems 4* (1995), 193–219.

[4] FRÜHWIRT, T., AND HANSCHKE, P. Terminological reasoning with constraint handling rules. In *Proc. Workshop on Principles and Practice of Constraint Programming* (1993).

[5] GOH, C. H., BRESSAN, S., MADNICK, S. E., AND SIEGEL, M. D. Context Interchange: Representing and Reasoning about Data Semantics in Heterogeneous Systems. Tech. Rep. #3928, Sloan School of Management, MIT, 50 Memorial Drive, Cambridge MA 02139, Oct. 1996.

[6] GOH, C. H., MADNICK, S. E., AND SIEGEL, M. D. Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, Nov 29–Dec 1 1994), pp. 337–346.

[7] JAKOBISIAK, M. Programming the web – design and implementation of a multidatabase browser. Tech. Rep. CISL WP#96-04, Sloan School of Management, Massachusetts Institute of Technology, May 1996.

[8] KAKAS, A. C., KOWALSKI, R. A., AND TONI, F. Abductive logic programming. *Journal of Logic and Computation 2*, 6 (1993), 719–770.

[9] KIFER, M., LAUSEN, G., AND WU, J. Logical foundations of object-oriented and frame-based languages. *JACM 4* (1995), 741–843.

[10] MCCARTHY, J. Generality in artificial intelligence. *Communications of the ACM 30*, 12 (1987), 1030–1035.

[11] QU, J. F. Data wrapping on the world wide web. Tech. Rep. CISL WP#96-05, Sloan School of Management, Massachusetts Institute of Technology. February 1996.

[12] SCIORE, E., SIEGEL, M., AND ROSENTHAL, A. Using semantic values to facilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems 19*, 2 (June 1994), 254–290.

[13] SHETH, A. P., AND LARSON, J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys 22*, 3 (1990), 183–236.

[14] WETZEL, G., KOWALSKI, R., AND TONI, F. Procalog: Programming with constraints and abducibles in logic. JICSLP Poster session, Sept. 1996.

[15] WIEDERHOLD, G. Mediators in the architecture of future information systems. *IEEE Computer 25*, 3 (March 1992), 38–49.