Graphical Metadata Management for the Context Mediation System

by

Usman Y. Mobin

Submitted to the
Department of Electrical Engineering and Computer Science

January 28, 2002

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

A system is presented which allows users to author, edit, and view the ontological, contextual and other metadata knowledge required by the reasoning engine of the Context Interchange Mediator[2]. Context Mediation[1] involves bridging the interpretational discrepancies between data provided by heterogeneous sources based on a common semantic framework. This semantic framwork[4] is encapsulated by the *ontology* provided to the mediation engine, while the actual interpretational differences between the sources are represented in the *context* for each source.

The proposed system makes it easy for the user to represent, author, and edit such information and makes possible a certain level of collaboration between individuals in maintaining such information. The ontology editor is implemented in the Java™ Programming Language[7] thus making it architecture independent and also allowing it to be embedded in a web-based system. The metadata management system, on a whole, is a web-based application which is developed using the JavaServer Pages technology[20] with an Oracle database as the backend. This allows for performance, reliability, and above all, portability. Thus, the system can be used on any Interent terminal worldwide through the Java Virtual Machine[8] of a web browser without the need to install any specialized software locally.

Thesis Supervisor: Stuart E. Madnick
Title: J N Maguire Professor Of Information Technologies, Sloan School Of Management; and Professor of Engineering Systems, School of Engineering.

Table of Contents

1 Introduction

The Context Mediation System[2] allows users to see a contextually unified[1] view of data obtained from disparate sources that potentially differ in their interpretation of the actual data elements. The problem of the reconciliation of such semantic heterogeneity has been elaborated most appropriately in [1] and the reader is referred to that paper if elucidation is necessary. More details regarding our solution to the problem can be found in [43] and [44].

1.1 The Contexts of Sources and Recipients.

As an elaboration, consider two databases that provide information about the wealth of individuals. Database One is based in the United Kingdom and contains a record of the form wealth(name: 'William H. Gates', wealth:42111000) and Database Two is based in the United States and contains a record of the form value(name: 'Bill Gates', wealth:56133). On a purely physical level, Database One says that the "wealth-relation" of "William H. Gates" is "42111000" and Database Two says that the "value-relation" of "Bill Gates" is "56133." Instantly, we see that these two sources of information are not comparable unless there is some explicit elaboration of the underlying assumptions about the data elements. For example, the name fields use different formats for the name[2]. Secondly, the wealth field in Database One could be in Pounds Sterling, quoted as thousands, and the mean value of the wealth over a specified period[3]. While Database Two might quote the wealth field as millions, in the currency of the country of residence of the individual, and the instantaneous value at a particular point in time.

Likewise, the user who wants to know the wealth of "Gates Bill" (perhaps, because his culture makes the assumption that last name goes first[4]) should not receive a "record not found" error because the database really does contain the required information. Also, the user should not receive a mere "56133" as an answer because the user might be expecting the data in, say, Japanese Yen. Thus, the recipient of information also has hidden assumptions about the data.

A *context* is an embodiment of the array of hidden assumptions each source or recipient makes about data elements.

1.2 The Semantic Framework

Consider now the case where the user wishes to obtain the wealth value by averaging[5] the values provided by the two sources[6]. It is obvious that both the data elements need to be *elevated* to a common semantic vocabulary before they may be compared on similar grounds. In fact, even if the user requests a single data item from a single source, the data item still needs to be elevated to a semantic vocabulary before an appreciation can be made of the ways in which it differs within contexts.

[1] Of course, this contextually unified view is in the recepient's own context.
[2] This particular example of disparity would be quite impossible to reconcile as there are no standards pertaining to the names of individuals.
[3] The mean value could be a good measure as most of the wealth is in risky assets and thus subject to a variance.
[4] Like many Asian cultures do.
[5] To perhaps be subject to a lower standard deviation.
[6] To be able to perfectly reconcile these two sources, we need to ignore the fact that one is a mean value and the other is an instantaneous value.

The ways in which each *semantic type* can differ across contexts is encapsulated in the concept of a *modifier*. In the example in section 1.1, the "wealth" field in the "value" relation in Database Two could be elevated to a semantic type called, say, MoneyAmount. We have already seen that this semantic type differed across the two databases in terms of the scale factor and the currency among other possible ways. Thus, the semantic type MoneyAmount has at least two modifiers: scale-factor, and currency. Other modifiers would pertain to time-frame, taxation, and so on.

Now suppose that the "value" relation in Database Two had been value(name, wealth, currency) instead of just value(name, wealth) then we notice that the currency for the MoneyAmount corresponding to "wealth" is now explicitly mentioned in the relation itself. Thus, MoneyAmount no longer contains a modifier currency but instead contains an *attribute* currency. An attribute is some information about a semantic type that we obtain directly from the database and does not vary based on context (except at the interpretational level).

The ontology or the domain model is the specification of the various semantic types involved in the system and an enumeration of their modifiers (the values of which are obtained based on context and alter the interpretation of the underlying data) and their attributes (the value of which are obtained from the database).

We will describe the contexts, ontology, and other metadata required by the system in more detail later.

1.3 The Presented System

The presented system provides the users of the Global Context Mediation System, as well as its earlier implementation, the Context Interchange Mediator[2], with an easy-to-use interface to author and edit the semantic framework for context mediation.

1.3.1 Motivation

The motivation for an easy-to use interface to author the semantic information comes from the fact that the internal representations of the ontology; contexts; conversion functions; elevation rules; and source descriptions, upon which the reasoning engine operates, are somewhat non-intuitive and it is inappropriate to expect the end user to program directly in them. In fact, authoring the metadata directly in the internal representation of the reasoning engine is a nontrivial exercise in the least, and requires extensive training. This is not desirable for a system that is intended for a reasonable sized audience.

Secondly, the internal representations are subject to change as the reasoning engine is augmented, or rewritten, to support a wider range of phenomena. Each such change would make the older metadata obsolete, and warrant retraining on the part of the users.

Thirdly, the internal representation of the semantic framework, inasmuch as it is expressive, does not render a very visible manifestation of the big picture of the framework. Such inability hampers further development of the semantic framework, and also does not yield gracefully to end-users with minimal training.

Finally, and quite importantly, the different categories of metadata have varying levels of dependence on each other. For example, if the ontology defines a semantic type "money amount" and we change its name to, say, "financial unit," then such a change will have repercussions for all other parts of the system. Changing the name of a semantic type, for example, makes the rest of the metadata inconsistent with the ontology.

The presented system automatically keeps all data consistent at all times. So, if the user changes the name of "money amount" to "financial unit," the context definitions and the

conversion functions and the other categories of metadata are automatically updated to be consistent with the new vocabulary. Without the presented system, maintenance of an application with very large amounts of metadata would become virtually impossible. Automatic change propagation has proved to be a very useful feature even for very modest-sized applications.

1.3.2 Objectives

The primary objective is to make the Context Mediation System usable for the end-user. Thus, the system enables an end-user to author and edit applications of the Context Mediation System with minimal training.

1.3.2.1 Intuitive Design

By providing a graphical interface to the Mediation System's semantic framework, the system makes the design of the metadata structure more intuitive as compared to working with the internal representation itself. The internal representation might be closer at heart to the reasoning engine, but not to the end user. Modifying or authoring the semantic framework by manipulating the internal representation might be quite a non-intuitive process for people who didn't actually develop the system. Also, authoring the metadata directly in the internal representation files is subject to consistency and maintenance problems as mentioned above. This arises from the fact that most parts of the metadata have a dependence on other parts.

1.3.2.2 Independence from the Reasoning Engine

The system provides a greater level of independence from the implementation of the reasoning engine. By not working directly at the internal representational level, the end-user is buffered from possible augmentations in the reasoning engine itself. Furthermore, by storing the semantic framework in an intermediate representation independent of the internal representation, all changes to the internal representation require changes only to the module that converts from the relevant database relations (intermediate representation) to the internal representation. Thus, the architecture of the system may be broadly illustrated as in figure 1:

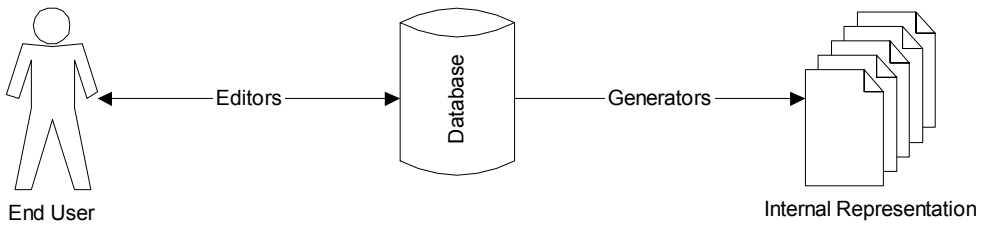

figure 1.1: The big picture

Notice that changes in the internal representation, as mandated by changes in the reasoning engine, are buffered from the structure of data in the database. Thus, a different reasoning engine requires merely programming new generators for the new internal representations. The presented system has the generators for the ontology, contexts, source descriptions, elevation rules, and conversion functions for both the Context Interchange Mediator[2] and its newer and more portable version, the Global Context Mediation System.

1.3.2.3 Minimal Training

The system requires very little training on the part of the end users as compared to the older set-up where the user had to author the semantic framework directly at the level of the internal representation. This makes it easier for the end-user to conceptualize about the entire semantic picture. Of course, the design of the internal representation should be such too that its use requires not-too-much training; however, working with any textual representation is not quite scalable in terms of the conceptualization of the big picture. Thus, a graphical representation seems necessary for complex semantic frameworks. Also, the nature of the metadata is such that small changes in certain parts thereof need to propagate to various other parts of the metadata. Working directly with a raw editor interface to the internal representation offers no luxury of automatic propagation of changes in the metadata to be reflected everywhere. Thus, the user has worry about consistency of, say, naming in the metadata and so on. The presented system alleviates the user of such unnecessary worries making the task of authoring and maintenance of metadata for context mediation applications a much more manageable exercise.

2 Related Work

Various editors for metadata have been developed over the years. However, most of them have been quite restricted in scope. Firstly, metadata editors have focused on particular aspects of metadata and have failed to capture the entire body of knowledge that might need representation in an environment where the semantic reconciliation of data can take place[4]. Just as it is important to have a metadata representation that is rich enough to describe sufficient data semantics to be useful in methods of identification and reconciliation of semantic heterogeneities, it is important to have editors that are able to support such constructs so as to generate such rich representations. Secondly, the editors have been primarily textual to semi-graphical and thus subject to considerable training requirement.

The editors presented in this system appear to be a first effort at graphically representing the entire body of knowledge that might be required by an automatic-conflict-detection based context mediation system[2], like the one described in [32].

The generators presented in this system, although re-programmable to generate any equivalent representation of metadata, currently generate the one required by the Context Interchange Mediator[5] and its later derivative, the Global Context Mediation System. This representation is attributed to Cheng Goh[28], and as he notes in [28], is guided by the ideas presented by Siegel and Madnick[4].

A significant body of related efforts has focused primarily on the edition and representation of ontological knowledge, with no particular emphasis on the applications towards the reconciliation of semantic heterogeneity.

A related effort was the domain modeler of Chris Leung, which was presented as his master's thesis in our research group. Much of his work, unfortunately, has been lost. His domain modeler was, in its functionality, similar to the ontology editor module of the Graphical Metadata Management system presented here

Another related work is VisioDAML[33]. VisioDAML is a Visio application that extends Visio so that it can be used to create graphical representations of DAML+OIL[17] ontologies. The implementation described in [33] is intended to provide, as close as possible, a direct one-to-one mapping between the DAML+OIL language constructs and their graphical representation. Although VisioDAML requires Visio, ontologies authored using that system may be viewed using Microsoft's freely available Visio Drawing Viewer[34] in case Visio is not available.

Netbryx Technologies have developed an editor that, on the related side, allows users to edit XML Document Type Definitions[35]. Currently, the editor, which the company has named "EditML," is in version 2.5.

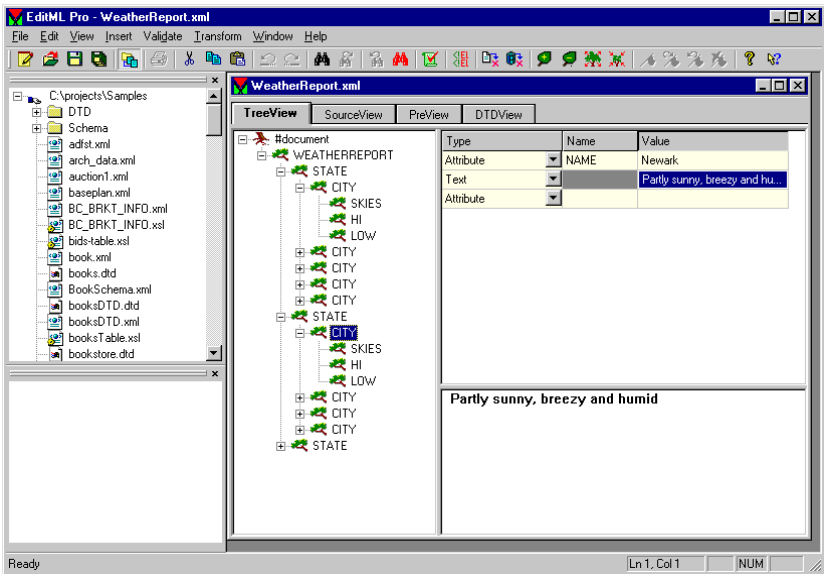

Figure 2.1: A screen shot of EditML in "TreeView" mode

Figure 2.1 shows a screenshot of EditML[36] when it is running in a mode called "TreeView," allowing the user to see the data structure in a hierarchical tree view. Of greater concern to us is the "DTD-view" of this program. This is shown in figure 2.2.

Notice that the editor is not very graphical, and at best the graphics are restricted to the views afforded by a hierarchical tree structure.

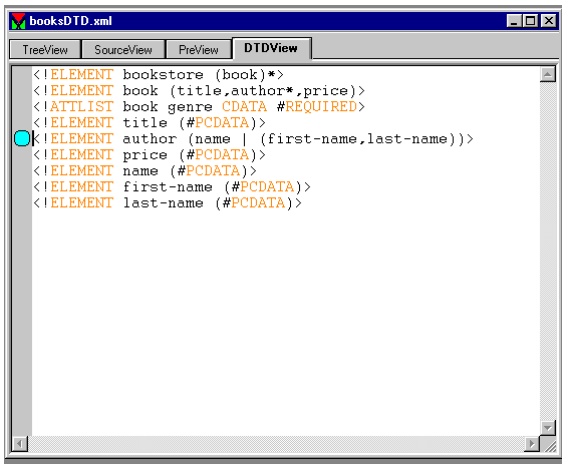

Figure 2.2: EditML in DTD-view mode

A simple Metadata Editor is the Reggie Metadata Editor[37], which is a Java-based Metadata editor created by the Resource Discovery Unit of the Distributed Systems Technology Center that exports HTML 3.2, HTML 4.0 and Resource Description Framework[3]. However, users have to author their own schema files, or use one of the predefined schema available in that system. For more details, the reader is referred to [37].

A related effort is the KRAFT (Knowledge Reuse And Fusion/Transformation) Ontology Browser developed at the University of Liverpool in the United Kingdom. Their system is described in [38] and the KRAFT Ontology Browser is presented in [39].

A Java-based ontology editor, the Java Ontology Editor (JOE) is maintained by the Department of Computer Science and Engineering of the University of South Carolina. This system is presented in [40]. These efforts are not suitable for our purpose as they are not sufficiently customizable and are not as freely graphical as suits our fancy.

Popkin Software has developed a program, Envision XML, which on the related side of things allows users to author document type definitions. Figure 2.3 shows a related screen shot from this program. Details can be found in [41].

A very prominent related effort is Stanford's project protégé[18]. Its current version, protégé-2000, is a tool which allows the user to construct a domain ontology, customize knowledge-acquisition forms, and enter domain knowledge; and a platform which can be extended with graphical widgets for tables, diagrams, animation components to access other knowledge-based systems embedded applications. The program requires the Java 2 plug-in to function (actually, so does our ontology editor). However, the graphical presentation is limited to a highly structured tree-hierarchy and does not allow the dexterity offered by tools that allow users to place objects on a canvas (like our ontology editor, for example).

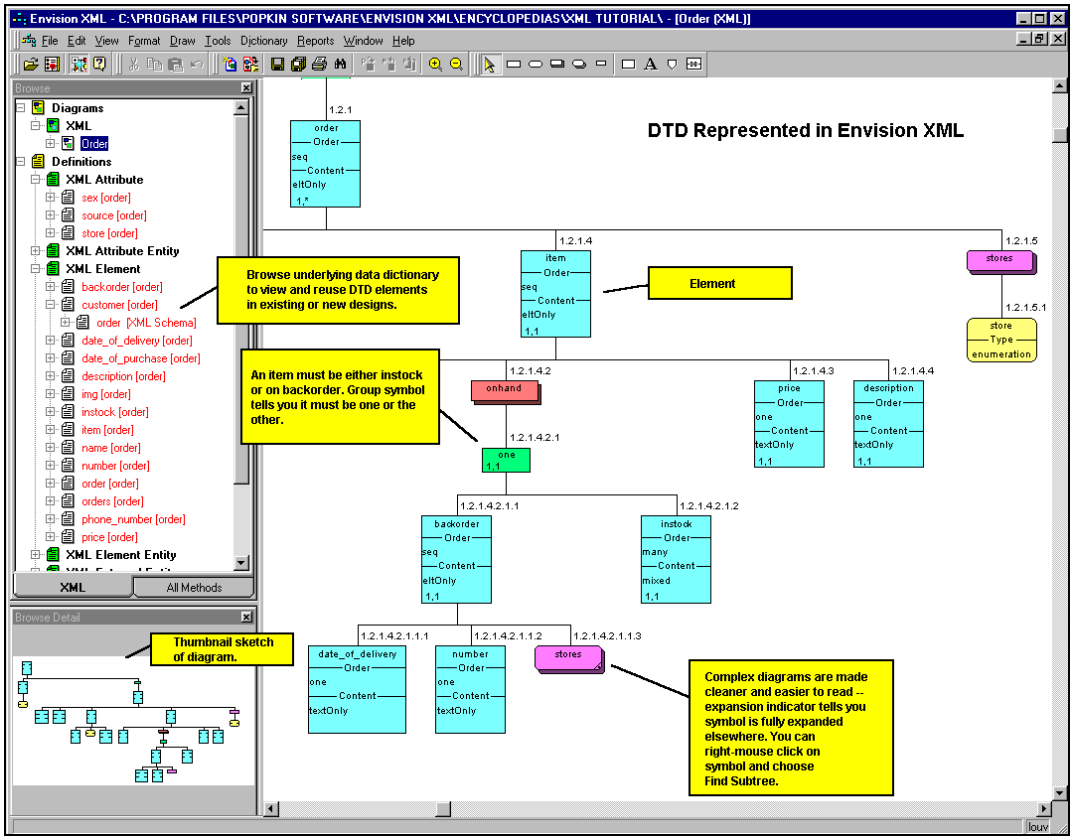

Figure 2.3 Popkin Software's EnvisionXML

3 Design Overview

This chapter presents an overview of the design of the system and describes the role of this work in the grand scheme of the context mediation system. This chapter also provides motivation for the context mediation system, and also for the Graphical Metadata Management system. The chapter ends with the high-level details of the overall structure of the Graphical Metadata Management system.

3.1 Overview of the system

This system works as the graphical front-end of the context mediation system. The context mediation system may be divided into five main components. At the front are the graphical metadata management tools that allow users to author the necessary metadata for their applications using a friendly interface. This tool generates the internal representation of the metadata for use by other parts of the system.

Also at the front is the Structured Query Language to datalog converter[7] which first takes in a SQL query and parses it to generate the relevant datalog query. A typical SQL query such as

```
select COMPANY_NAME
from DiscAF
where COMPANY_NAME='DAIMLER-BENZ';
```

first gets converted to its datalog equivalent, which may be considered as a query in the Prolog realm. The datalog for the above query would look like

```
answer('V7') :-
      'DiscAF'('V7', 'V6', 'V5', 'V4', 'V3', 'V2', 'V1'),
      'V7' = "DAIMLER-BENZ".
```

So basically, the system looks up the sources description and determines that the first column in the DiscAF relation (variable V7 above) pertains to the requested column. This datalog query gets elevated onto semantic space as follows. We will have more to say on semantic space in our description of elevation rules.

```
answer('V9') :-
      'DiscAF_p'('V8', 'V7', 'V6', 'V5', 'V4', 'V3', 'V2'),
      value('V8', datastream, 'V9'),
      'V1' = "DAIMLER-BENZ",
      value('V8', datastream, 'V1').
```

This result is an elevated datalog query[8] in the receiver's context. Here the receiver's context is datastream.

The next main component of the context mediation system is the reasoning engine which uses abductive logic to mediate the elevate datalog query so as to present it in the context of the recipient. The result from the abduction engine for the above query would look like:

```
answer("DAIMLER-BENZ") :-
      'DiscAF'('V7', 'V6', 'V5', 'V4', 'V3', 'V2', 'V1'),
      'Name_map_Dt_Ds'("DAIMLER-BENZ", 'V7').
```

[7] Which was programmed by me earlier this summer.
[8] Also referred to as context-sensitive datalog.

The details of this process are not the subject of this thesis and are described in much detail in [28] and [43]. The reader is referred to those papers if he or she would like to get the mathematical model behind these transformations.

At the backend resides the executioner[29]. This system takes the output from the mediation system and dispatches relevant queries to relational databases. The output from the databases is then consolidated and presented to the end-user. This architecture of the context mediation system is illustrated in figure 3.1, which is reproduced from Madnick[1].

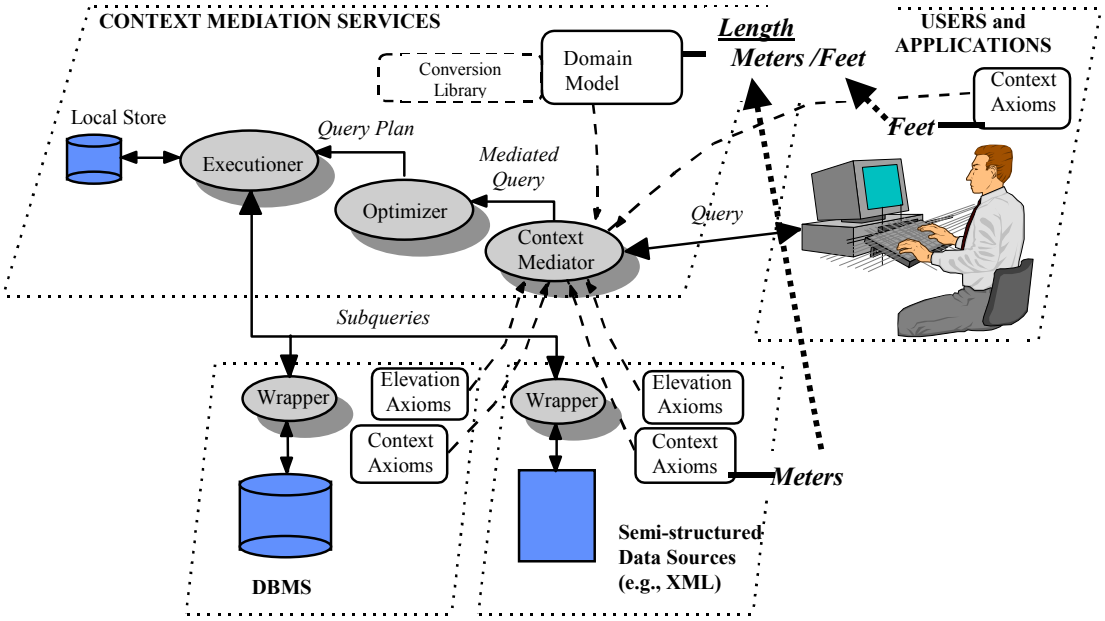

Figure 3.1: Architecture of the context mediation system.

Finally, under development at our research group, is the query-builder and application-browser tool. This component manages the registry[9] and allows the users to create and browse applications of the context mediation system. Of course, the metadata in those applications is authored and managed using the graphical metadata management system presented here.

[9] Which integrates with the relational model of data used by this graphical metada management system

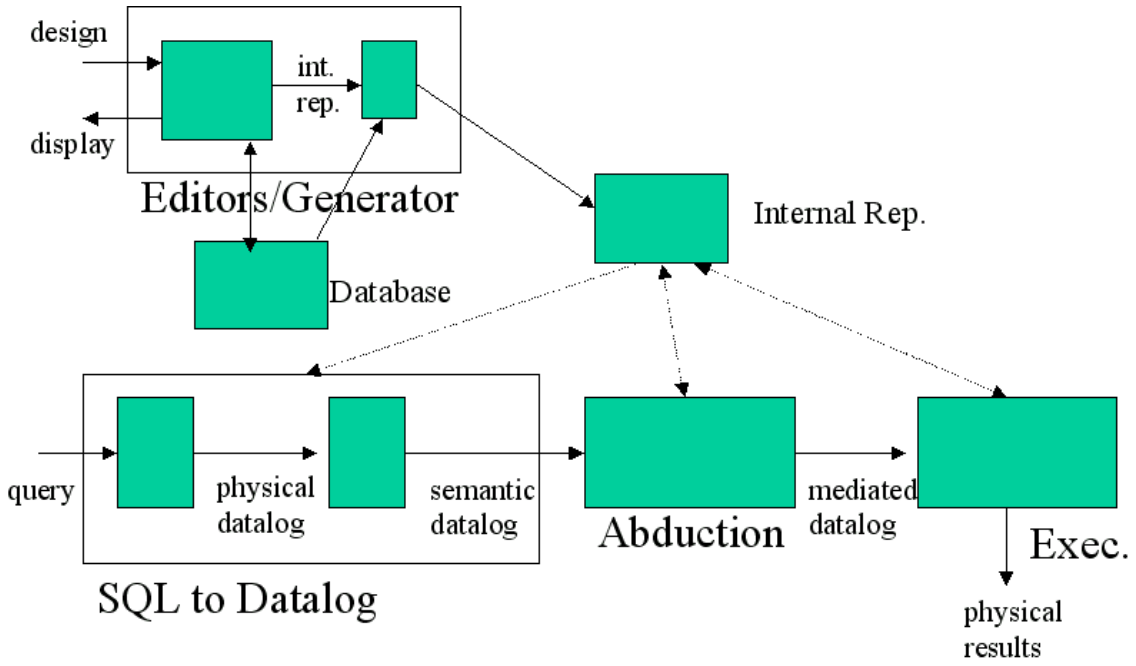

Figure 3.2: the big picture

Figure 3.2 shows the big picture of the context mediation system. However, the application browser and query builder modules under development at our research group are not shown in this figure. They are discussed more in a later section on future work.

Figure 3.3 zooms in on the presented system in the big picture

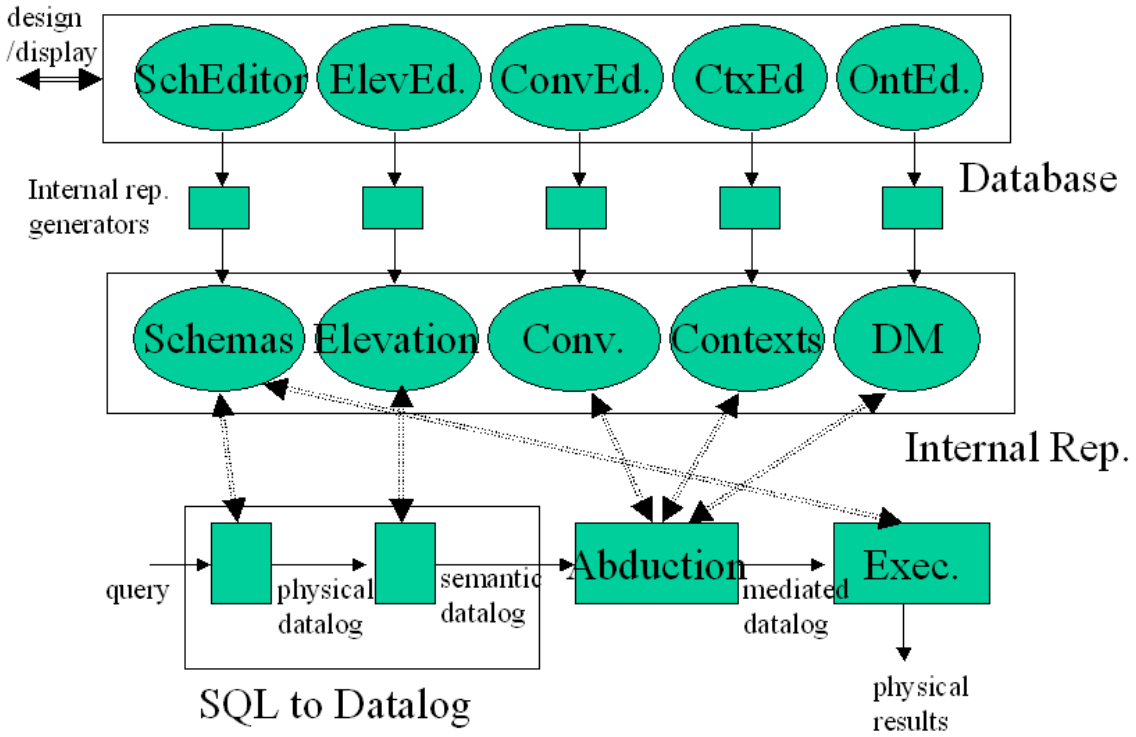

Figure 3.3: The medium sized picture

3.2 Motivation for Context Mediation and Graphical Front-End

Users of the context mediation system use the system in one of two roles: information retrievers, and information providers. Information retrievers use applications of the context mediation system by entering SQL queries (as already shown in the previous section) and obtaining mediated results back from the system in the context of their choosing.

Information providers are the class of users that author and manage the metadata required for the applications of the context mediation system. The graphical front-end presented here is a valuable tool for users operating in both the roles

In the past, information retrievers were provided with little or no knowledge of the source structure and the contexts available in the application. It was assumed that the end-user somehow knew the table definitions of remote sources as well as the context definitions for the contexts in the system. The presented system allows the end user to see the source structure and context definitions in a very user-friendly form. This equips the user with the information needed to make queries to the system.

In this section, we describe a motivational example for the context mediation system with an emphasis on the role of the graphical front-end in making the system usable. This example will allow the reader to develop an appreciation for the entire system as well as the need and value-added by the graphical front-end. Primary emphasis in this section will be on the value of the system for the user operating in his or her role as the information retriever. We will provide an example of building an application for the information provider, in a later section.

The context mediation system makes the assumption that the relational data model is the one exposed to the user[26] and the user makes queries in SQL or some extension thereof. Of course, this does not mean that the actual data source must be relational. For example, World Wide Web pages may be used as data source if they are appropriately encapsulated with a relational interface. Such a program is referred to as a relational web-wrapper[16] and allows the end user to obtain a relational view of a web source[10]. However, as pointed out in [28], the choice of a relational data model is one of convenience and does not signify any constraint imposed by the context mediation strategy. Consider the user has access to three data sources containing financial information pertaining to companies: disclosure, datastream, and datastream. Suppose the user wants the income per assets of, say, British Telecom. Firstly, the user would like to query the database to know what information is provided by each of the data sources. The user queries the database to obtain information about the data sources disclosure and worldscope

```
SQL*Plus: Release 8.1.7.0.0 - Production on Fri Feb 1 04:35:44 2002

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Enter user-name: system@coin/manager

Connected to:
Personal Oracle8i Release 8.1.7.0.0 - Production
JServer Release 8.1.7.0.0 - Production

SQL> desc disclosure
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------

 COMPANY_NAME                                       VARCHAR2(40)
 LATEST_ANNUAL_DATA                                 VARCHAR2(8)
 CURRENT_SHARES_OUTSTANDING                         NUMBER
 NET_INCOME                                         NUMBER
 NET_SALES                                          NUMBER
```

[10] Of course, the capabilities of relationally wrapped web sources are less than those of innately relational sources.

```
 LOCATION_OF_INCORP                                        VARCHAR2(20)

SQL> desc worldscope
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------

 COMPANY_NAME                                        VARCHAR2(80)
 LATEST_ANNUAL_FINANCIAL_DATE                        VARCHAR2(10)
 CURRENT_OUTSTANDING_SHARES                          NUMBER
 SALES                                               NUMBER
 TOTAL_ASSETS                                        NUMBER
 COUNTRY_OF_INCORP                                   VARCHAR2(40)
```

The user notices that disclosure provides information regarding income, whilst worldscope provides the information regarding the assets. The user first decides to query disclosure to obtain the net income of British Telecom.

```
SQL> select company_name, net_income
  2  from disclosure
  3  where company_name = 'BRITISH TELECOM';

no rows selected
```

Notice that the database returned no results. The reason is that there is a contextual difference between the user and disclosure in the naming of the company the user thinks is "British Telecom." The user makes the assumption that the company's name is "British Telecom" while disclosure assumes a different name that the user, perhaps, is not *exactly* aware of. The user and disclosure, in this case, ascribe to different standards.

The user figures that disclosure might be using a different string to represent British Telecom. So the user queries disclosure for all companies with names beginning with "B"

```
SQL> select unique(company_name)
  2  from disclosure
  3  where company_name like 'B%';

COMPANY_NAME
----------------------------------
B F GOODRICH CO
BAKER HUGHES INC
BALL CORP
BANTA CORP
BASF CORP
BAUSCH & LOMB INC
BAXTER INTERNATIONAL INC
BECKMAN INSTRUMENTS INC
BECTON DICKINSON & CO
BEMIS CO INC
BERKSHIRE HATHAWAY INC

COMPANY_NAME
----------------------------------
BETHLEHEM STEEL CORP
BETZ LABORATORIES INC
BLACK & DECKER CORP
BLOCK DRUG CO INC
BLOUNT INC
BOEING CO
BOISE CASCADE CORP
BORDEN INC
BORG WARNER AUTOMOTIVE INC
BOWATER INC
```

```
BRIGGS & STRATTON CORP

COMPANY_NAME
--------------------------------
BRISTOL MYERS SQUIBB CO
BRITISH TELECOMMUNICATIONS PLC
BROWN FORMAN CORP
BRUNSWICK CORP
BURLINGTON INDUSTRIES EQUITY INC
BURLINGTON RESOURCES INC

28 rows selected.
```

Notice, that among the list is "BRITISH TELECOMMUNICATIONS PLC." Without going into the philosophical issues of identity, the user can safely assume that this is the company he or she was looking for. Using this newfound knowledge, the user can reissue the initial query regarding the income of British Telecom.

```
SQL> select company_name, net_income
  2  from disclosure
  3  where company_name = 'BRITISH TELECOMMUNICATIONS PLC';

COMPANY_NAME                              NET_INCOME
----------------------------------------- ----------
BRITISH TELECOMMUNICATIONS PLC            1767000000
```

Good. The user next wants to find out the assets information from worldscope. The user would ideally just like to "select total_assets from worldscope where company_name = 'BRITISH TELECOM'." However, now based on experience the user know that he or she does not have the luxury of making queries in his or her own "context." So the user issues the following query:

```
SQL> select company_name
  2  from worldscope
  3  where company_name like 'BR%';

COMPANY_NAME
---------------------------------

BRACKNELL CORPORATION
BRIGGS & STRATTON CORPORATION
BRISTOL-MYERS SQUIBB CO.
BRITISH TELECOMMUNICATIONS PLC
BROWNING-FERRIS INDUSTRIES, IN
BRUNO'S INCORPORATED
BRUNSWICK CORPORATION
BRUNSWICK MINING AND SMELTING

8 rows selected.
```

It just so happens that disclosure and worldscope ascribe to the same standard in naming "British Telecom." This is not always the case though, as data sources can vary widely on the naming standards used. As an example, notice the results from the following three queries:

```
SQL> select company_name
  2  from disclosure
  3  where company_name like 'DAIMLER%';

COMPANY_NAME
----------------------------------------
DAIMLER BENZ CORP
```

```
SQL> select company_name
  2  from worldscope
  3  where company_name like 'DAIMLER%';

COMPANY_NAME
--------------------------------------------
DAIMLER-BENZ AG


SQL> select name
  2  from dstreamaf
  3  where name like 'DAIMLER%';

NAME
------------------------------
DAIMLER-BENZ
```

Coming back to our example, the user now finds the total assets of British Telecom as follows:

```
SQL> select total_assets
  2  from worldscope
  3  where company_name = 'BRITISH TELECOMMUNICATIONS PLC';

TOTAL_ASSETS
------------
    33528882
```

The data sources seem to work, though the user had to find out what name was assumed by each source for British Telecom. Finally, the time to obtain the final result that the user had sought out to explore at the beginning, the ratio of income to assets for British Telecom:

```
SQL> select (disclosure.net_income / worldscope.total_assets)
  2  as ratio_of_income_to_assets
  3  from disclosure, worldscope
  4  where disclosure.company_name = 'BRITISH TELECOMMUNICATIONS PLC'
  5  and worldscope.company_name = 'BRITISH TELECOMMUNICATIONS PLC';

RATIO_OF_INCOME_TO_ASSETS
-------------------------
             52.7008327
```

Fifty-three? Something certainly is amiss here. Yes, something *is*. Integrating data from heterogeneous sources is not as easy a task as it might appear at face value. Not only did the user encounter contextual differences with the sources regarding the naming of the company, the sources themselves differ in their assumptions underlying their reported data.

The user would have to browse through the fact sheets accompanying each data source to find out that datastream reports all financial amounts in thousands, in the currency of the country of incorporation of the company; disclosure, on the other hand, uses a scale factor of unity for financial amounts and, again, the official currency of the country of incorporation of the pertinent company; finally, worldscope uses a scale factor of a thousand for financial amounts and United States Dollar as the currency.

Let's decouple the query back into two columns so that no single column in the result is based on more than one data source:

```
SQL> select disclosure.net_income, worldscope.total_assets
```

```
  2  from disclosure, worldscope
  3  where disclosure.company_name = 'BRITISH TELECOMMUNICATIONS PLC'
  4  and worldscope.company_name = 'BRITISH TELECOMMUNICATIONS PLC';

NET_INCOME TOTAL_ASSETS
---------- ------------
1767000000     33528882
```

Based on the previous paragraph about the underlying assumptions of these sources, we know the following about the results returned by the query:

	Scale Factor	Currency
NET_INCOME	One	Official currency of country of incorporation of the company
TOTAL_ASSETS	One Thousand	United States Dollars

So the user would have to first look up the country of incorporation of British Telecom (which, quite obviously is the United Kingdom) and then look up the currency of the United Kingdom (British Pounds Sterling). Finally, suppose the user wants to see both results in Japanese Yen, he or she would have to look up the conversion rates for both United States Dollars and Pounds Sterling to Japanese Yen, and then finally have the information he or she sought out to obtaining in the beginning.

This was a simple example. It involved two sources, two data items, one company, and three currencies. Unfortunately, the real world is quite a bit more complex. Consider an example involving ten data sources, five hundred companies spread over one hundred countries, using fifteen standards for company names…

The context mediation system automatically reconciles these contextual differences between heterogeneous data sources having different assumptions about their underlying data, and presents the results to the user in a form that is familiar to the user. We have developed an application of the Context Mediation System which is aimed at users like the one above, and performs context mediation on the financial databases mentioned above.

Suppose the user above wanted to know the net income and total assets of British Telecom using the context mediation system. The user would simply go to the sources page of the graphical metadata manager for the appropriate application by clicking on "sources" next to "TASC financial example" in the page shown in figure 3.4:

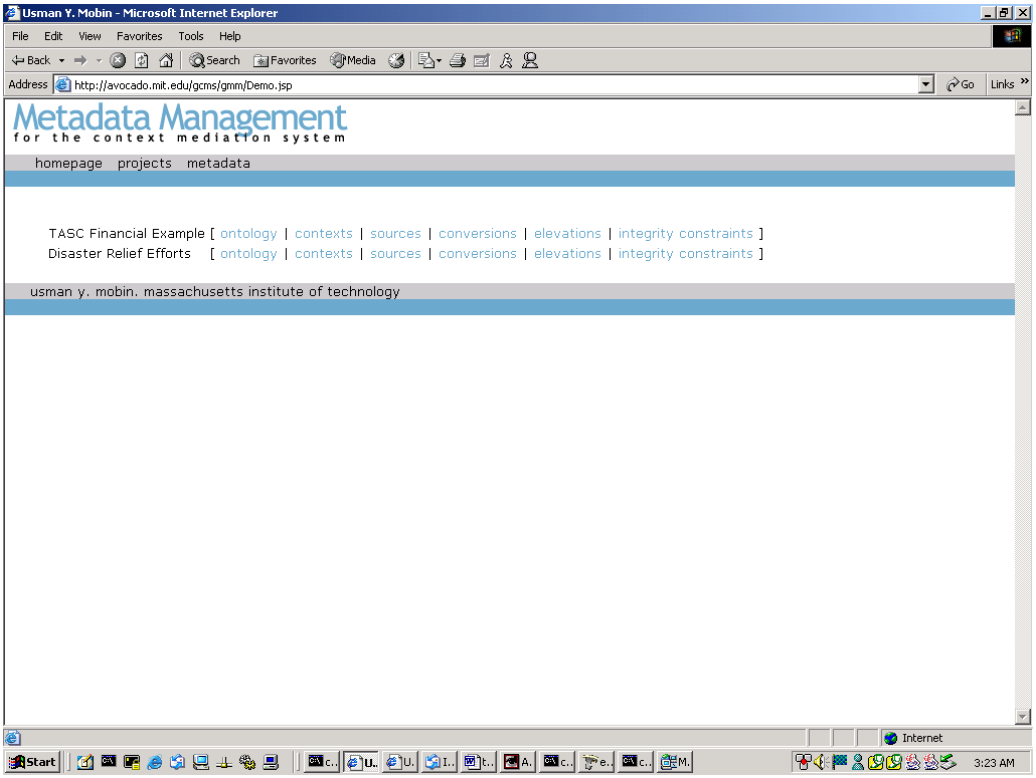

Figure 3.4: temporary applications page

However, as we mentioned earlier, the main query building and application browsing system is in the works at our research group and as such, the page shown in figure ? is a temporary placeholder for application browsing system.

Once the user clicks on sources, he or she is able to find out what databases are available and what columns exist within each table. This makes the task of querying much simpler for the end-user as the procedure for discovery of the underlying physical schema description is much easier with the graphical metadata management system. Figure 3.5 shows the sources description that the system displays to the user.

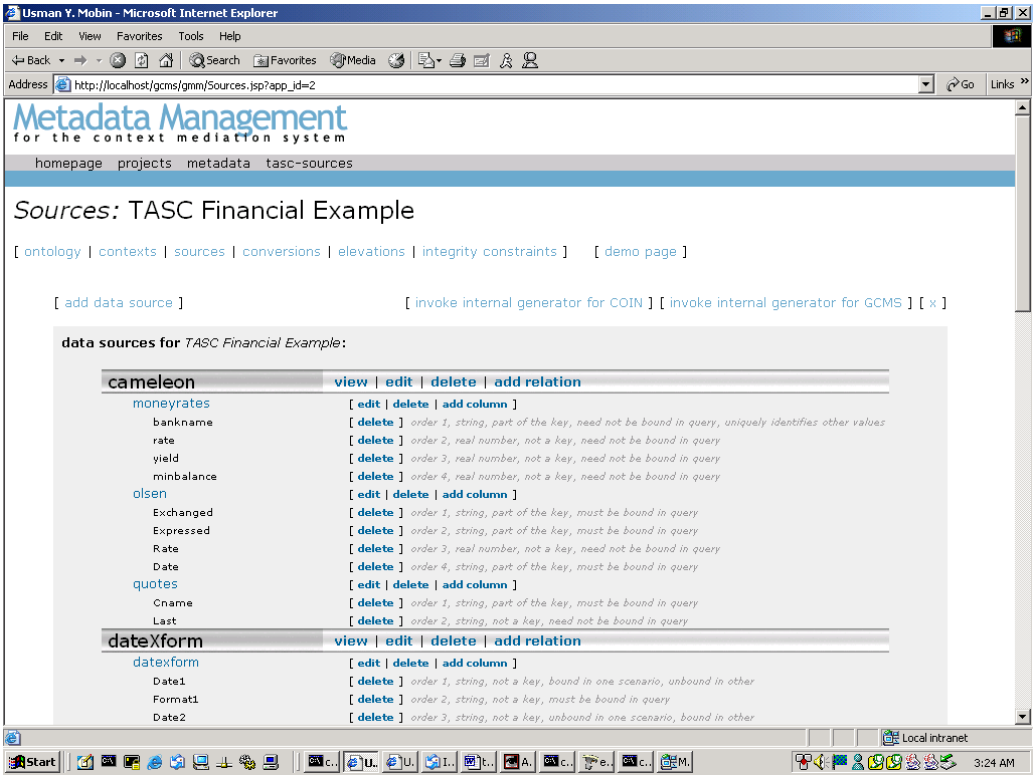

Figure 3.5: sources description page for financial application

Since the sources descriptions are a little too long, they are all not visible on the screen shot in figure 3.5. So for the benefit of the reader, they are reproduced in textual form below:

data sources for *TASC Financial Example*:

cameleon	view | edit | delete | add relation
moneyrates	[edit | delete | add column]
bankname	[delete] *order 1, string, part of the key, need not be bound in query, uniquely identifies other values*
rate	[delete] *order 2, real number, not a key, need not be bound in query*
yield	[delete] *order 3, real number, not a key, need not be bound in query*
minbalance	[delete] *order 4, real number, not a key, need not be bound in query*
olsen	[edit | delete | add column]
Exchanged	[delete] *order 1, string, part of the key, must be bound in query*
Expressed	[delete] *order 2, string, part of the key, must be bound in query*
Rate	[delete] *order 3, real number, not a key, need not be bound in query*
Date	[delete] *order 4, string, part of the key, must be bound in query*
quotes	[edit | delete | add column]
Cname	[delete] *order 1, string, part of the key, must be bound in query*
Last	[delete] *order 2, string, not a key, need not be bound in query*
dateXform	view | edit | delete | add relation
datexform	[edit | delete | add column]
Date1	[delete] *order 1, string, not a key, bound in one scenario, unbound in other*
Format1	[delete] *order 2, string, not a key, must be bound in query*
Date2	[delete] *order 3, string, not a key, unbound in one scenario, bound in other*
Format2	[delete] *order 4, string, not a key, must be bound in query*
oracle	view | edit | delete | add relation
Currency_map	[edit | delete | add column]
char3_currency	[delete] *order 1, string, not a key, need not be bound in query, uniquely identifies other values*

char2_currency	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
Currencytypes	[edit | delete | add column]
country	[delete] *order 1, string, not a key, need not be bound in query, uniquely identifies other values*
currency	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
DiscAF	[edit | delete | add column]
company_name	[delete] *order 1, string, part of the key, need not be bound in query*
latest_annual_date	[delete] *order 2, string, part of the key, need not be bound in query*
current_shares_outstanding	[delete] *order 3, integer, not a key, need not be bound in query*
net_income	[delete] *order 4, integer, not a key, need not be bound in query*
net_sales	[delete] *order 5, integer, not a key, need not be bound in query*
total_assets	[delete] *order 6, integer, not a key, need not be bound in query*
country_of_incorp	[delete] *order 10, string, not a key, need not be bound in query*
DStreamAF	[edit | delete | add column]
as_of_date	[delete] *order 1, string, part of the key, need not be bound in query*
name	[delete] *order 2, string, part of the key, need not be bound in query*
total_sales	[delete] *order 3, integer, not a key, need not be bound in query*
total_extraord_items_pre_tax	[delete] *order 4, integer, not a key, need not be bound in query*
Earned_for_ordinary	[delete] *order 5, integer, not a key, need not be bound in query*
currency	[delete] *order 6, string, not a key, need not be bound in query*
Name_map_Ds_Ws	[edit | delete | add column]
ds_names	[delete] *order 1, string, not a key, need not be bound in query, uniquely identifies other values*
ws_names	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
Name_map_Dt_Ds	[edit | delete | add column]
dt_names	[delete] *order 1, string, not a key, need not be bound in query, uniquely identifies other values*
ds_names	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
Name_map_Dt_Ws	[edit | delete | add column]
dt_names	[delete] *order 1, string, not a key, need not be bound in query, uniquely identifies other values*
ws_names	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
Ticker_Lookup2	[edit | delete | add column]
comp_name	[delete] *order 1, string, part of the key, need not be bound in query*
ticker	[delete] *order 2, string, not a key, need not be bound in query, uniquely identifies other values*
exc	[delete] *order 3, string, not a key, need not be bound in query*
WorldcAF	[edit | delete | add column]
company_name	[delete] *order 1, string, part of the key, need not be bound in query*
latest_annual_financial_date	[delete] *order 2, string, part of the key, need not be bound in query*
current_outstanding_shares	[delete] *order 3, integer, not a key, need not be bound in query*
net_income	[delete] *order 4, integer, not a key, need not be bound in query*
sales	[delete] *order 5, integer, not a key, need not be bound in query*
total_assets	[delete] *order 6, integer, not a key, need not be bound in query*
country_of_incorp	[delete] *order 7, string, not a key, need not be bound in query*
WorldcAFT	[edit | delete | add column]
company_name	[delete] *order 1, string, part of the key, need not be bound in query*
latest_annual_financial_date	[delete] *order 2, string, part of the key, need not be bound in query*
current_outstanding_shares	[delete] *order 3, integer, not a key, need not be bound in query*
net_income	[delete] *order 4, integer, not a key, need not be bound in query*
sales	[delete] *order 5, integer, not a key, need not be bound in query*
total_assets	[delete] *order 6, integer, not a key, need not be bound in query*
country_of_incorp	[delete] *order 7, string, not a key, need not be bound in query*

Now, suppose the user wants to issue the query for net income and total assets for British Telecom and obtain the results in worldscope's context. The user would simply issue the query to the context mediation system, and that's it! All the cumbersome contextual conflicts
Are handled automatically!

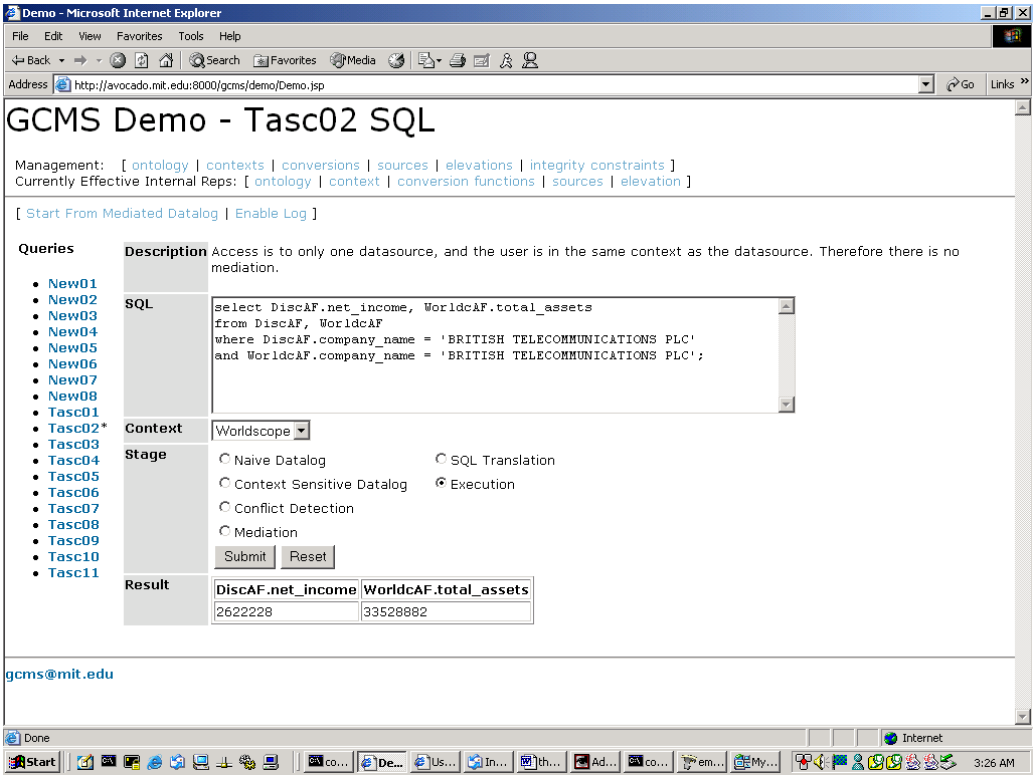

Figure 3.6: user issues query to the system

Notice that the results are:

DiscAF.net_income	WorldcAF.total_assets
2622228	33528882

Figure 3.7: results of user query

The important thing to notice is that all conflict resolutions, currency conversions, and contextual disparities are transparently handled by the system and the user sees the results in the context of his or her choosing. Of course, with the graphical metadata management system, not only is finding out the source descriptions easier, it is also much more easy to find context definitions. All the user has to do to view the context definitions is to click on the "contexts" link in the applications window. In the case of our financial application, the context window would look like:

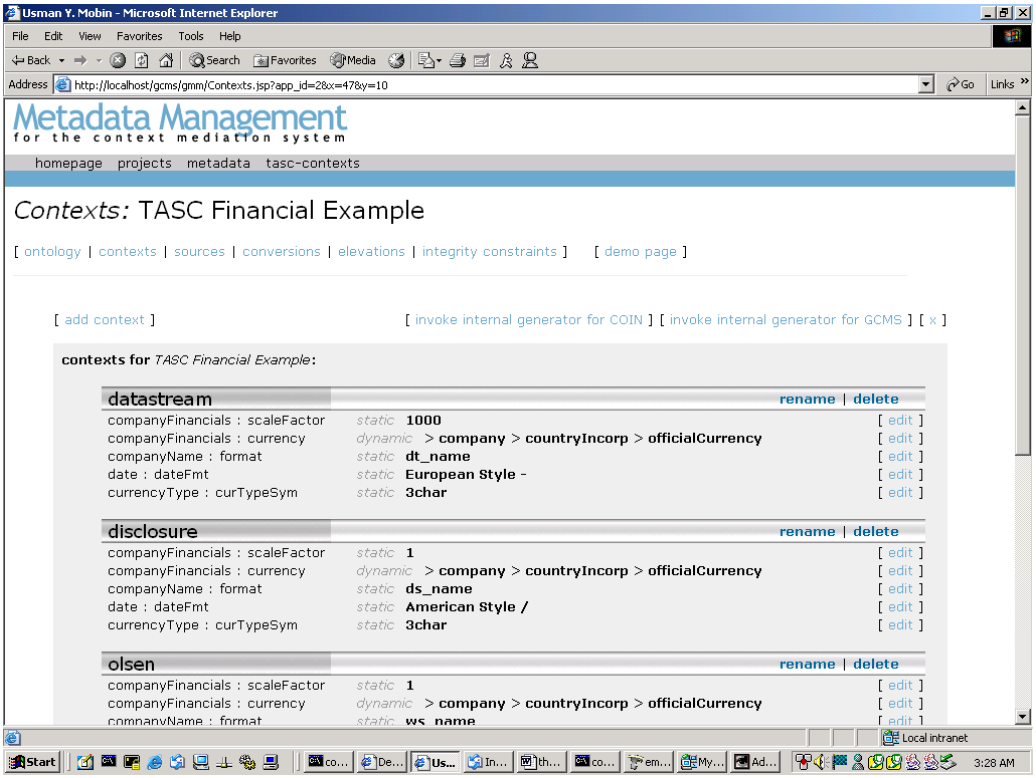

Figure 3.8: contexts for the financial application

The user can scroll down to view the context definition of worldscope. This is shown in figure 3.9:

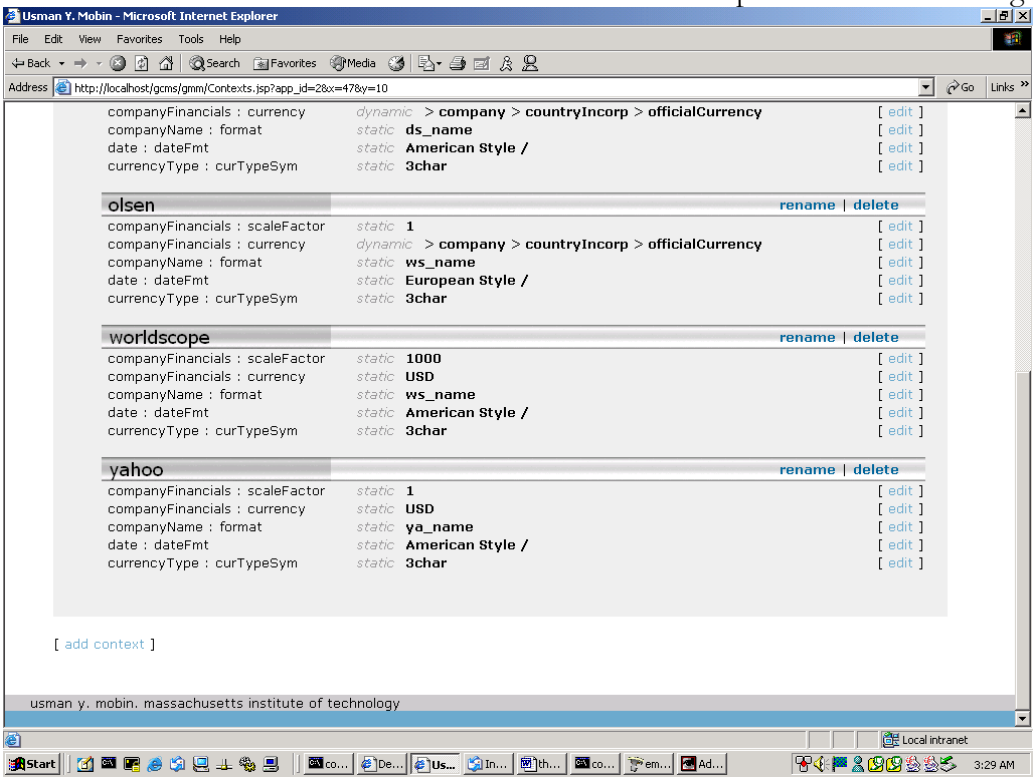

Figure 3.9: context definition for worldscope

Notice, that the context for worldscope is to report financial amounts in thousands of United States Dollars, among other things. This is exactly what the user found after going through the cumbersome procedure of browsing the worldscope fact sheets.

3.3 Motivational Guided Tour for Graphical Metadata Management

Suppose the user mentioned above subscribes to his or her own naming standards[11] and makes frequent queries based on those. For the purpose of this example, we will assume that the user is an entity named "Microsoft" and assumes that financial amounts are in United States Dollars with a scale factor of a hundred and, among other things, has a vocabulary of company names in which worldscope's "British Telecommunications Plc" is named as "British Telecom."

The Graphical Metadata Management system makes it very easy for users to author their own contexts. Without the ability to author one's own contexts, one is restricted to viewing data in one of the existing contexts in the system. This might not be acceptable to users who make frequent queries.

The user, while going through the sources description shown earlier, notices that the table DiscAF reports both the net income and total assets for companies. Now to obtain these values from DiscAF, the user needs to issue the following query:

```
select net_income, total_assets
from DiscAF
where company_name = 'BRITISH TELECOMMUNICATIONS PLC';
```

which, in disclosure's context returns the following results:

DiscAF.net_income	DiscAF.total_assets
1767000000	22565000000

Figure 3.10: results from user query on DiscAF

Of course, the user would have preferred to query the database on "British Telecom" directly irrespective of the vocabulary used by the queried database. Also, assume that the user wants to know the results in hundreds[12] of United States Dollars. To do this, the user can author his or her own context.

Authoring contexts is much easier than before with the Graphical Metadata Management system. First, the user goes to the contexts page shown earlier in figure 3.8 and clicks on the "add context" link. This brings the user to the context addition dialog page shown in figure 3.11

[11] As is not uncommon for big organizations to do.

[12] Hundred has not been a popular scale factor in recent historical times but is being used here just to illustrate the power of authoring one's own contexts.

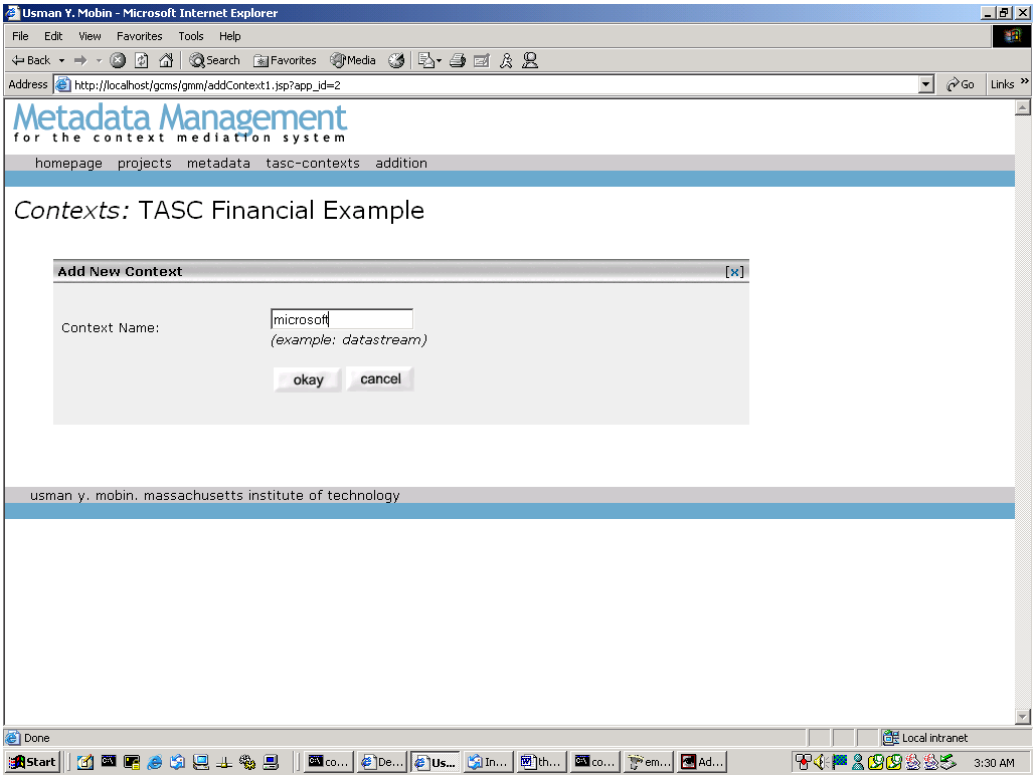

Figure 3.11: context addition dialog

The user specifies a name for the context. We will assume that the user names the context "microsoft." After filling in the context name field, the user clicks on the "okay" button. This displays a confirmation and brings the user back to the contexts page. However, now the context page displays "Microsoft context" as well as the previously existing contexts. Notice that all the modifier values in this context display "null value." This simply signifies that the user has not specified any modifier values in that context till now. This is illustrated in figure 3.12 below

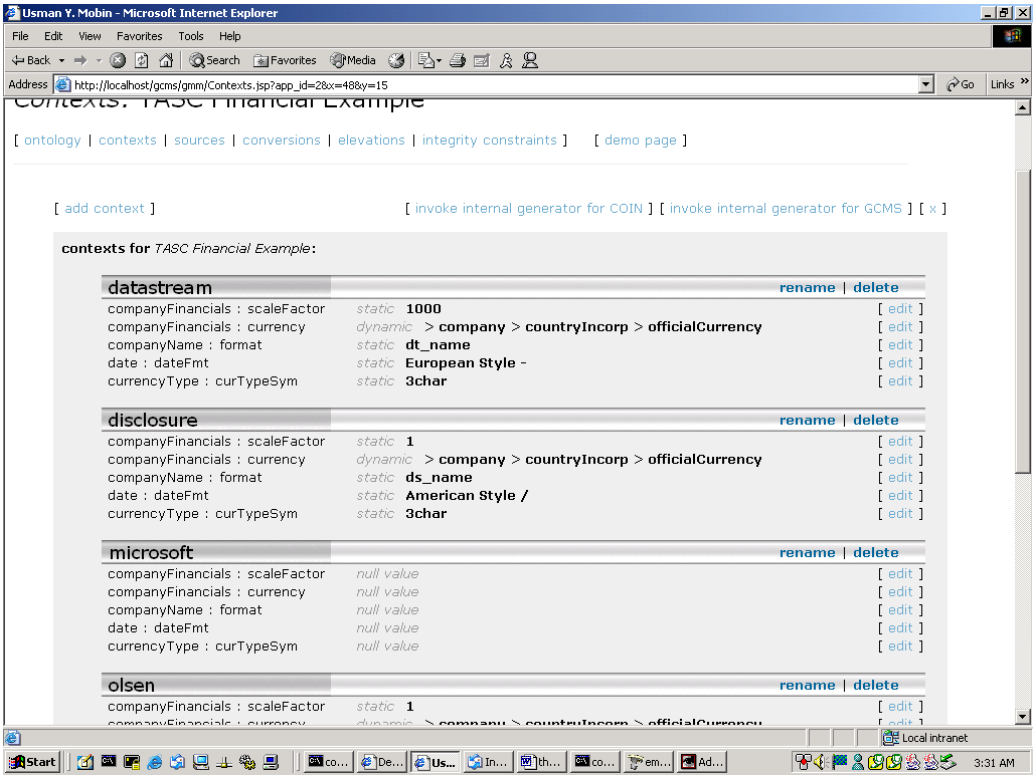

Figure 3.12: a newly created context

As mentioned before, a context contains modifier values. In the figure above, "companyFinancials" is a semantic type and "scaleFactor" is a modifier thereof. The user wants to see net_income and total_assets in relation DiscAF in hundreds of United States Dollars. To do so, the user needs to set the appropriate modifier values to 100 and "USD." But before the user can do that, he or she needs to know which semantic types are net_income and total_assets elevated to when semantic reconciliation takes place. Again, the Graphical Metadata Management system comes to the rescue. The system makes it very easy for the user to find this information. All the user has to do is click on the "elevations" link and elevations page shows up with all the appropriate elevation information for all the physical relations in the data sources for the application. See figure below.

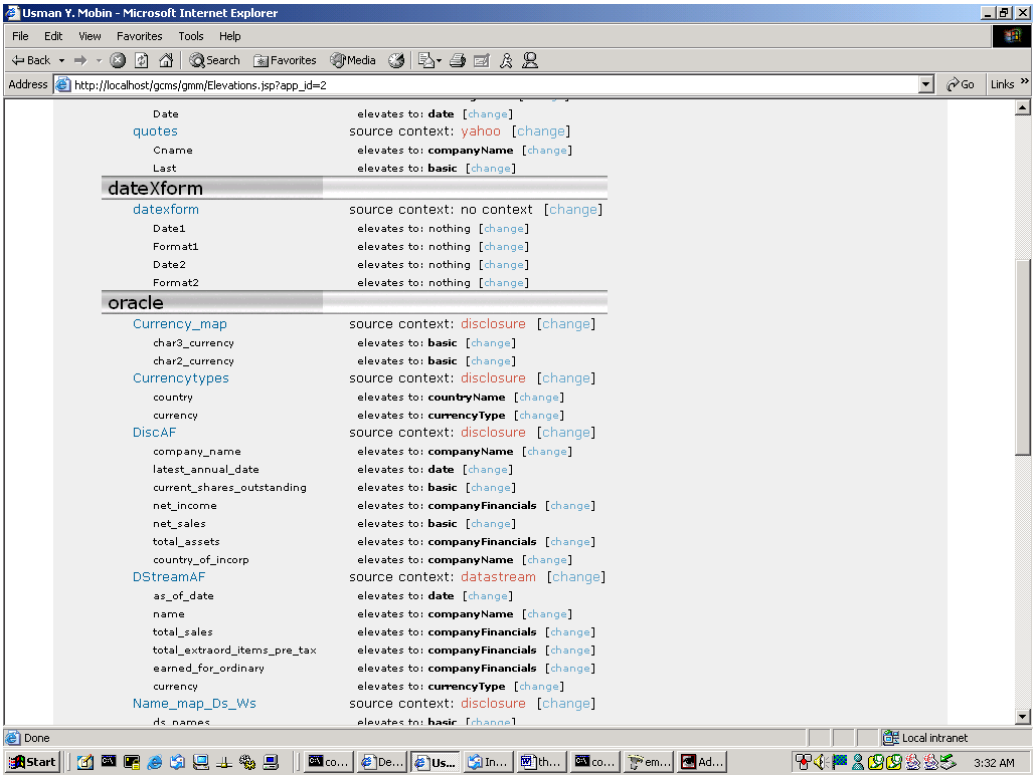

Figure 3.13: finding elevations information

The user can easily see that both net_income and total_assets in relation DiscAF (see under source named "oracle") elevate to semantic type "companyFinancials" so coming back to the contexts page, the user now knows that he or she needs to set the value of the scaleFactor modifier for companyFinancials to 100 and the currency modifier to "USD." To edit the modifier values, the user clicks on the "edit" link for the particular modifier. Clicking on the "edit" link for "companyFinancials : scaleFactor" brings the user to the page illustrated in figure 3.14 below.

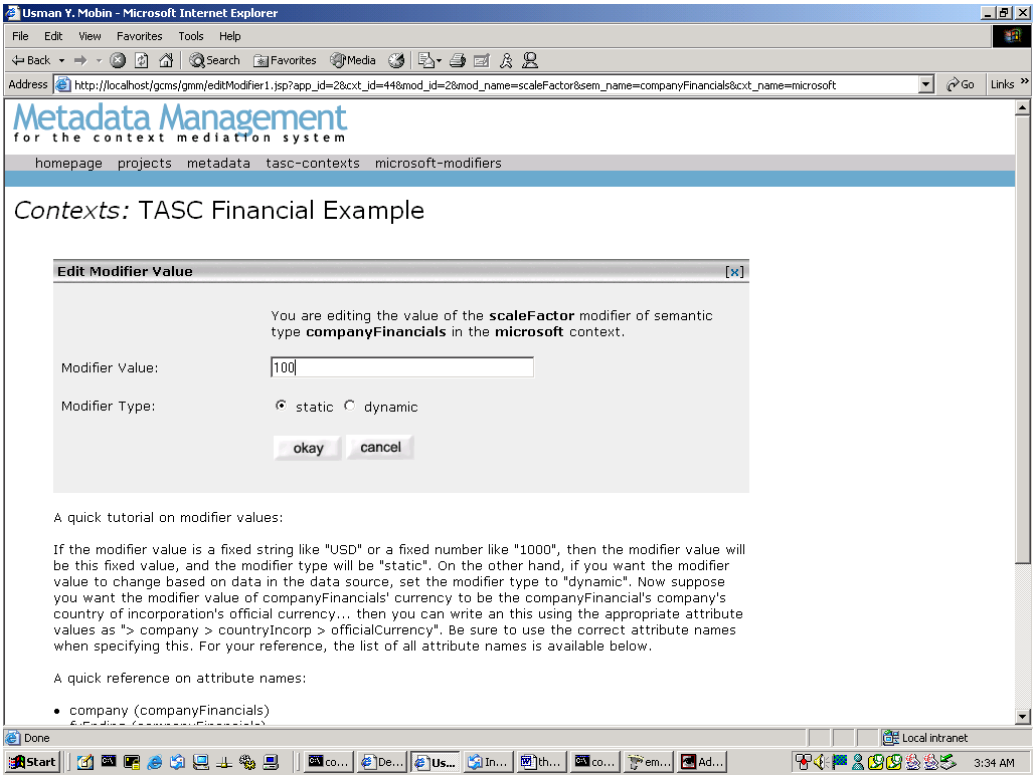

Figure 3.14: specifying a modifier value

A quick tutorial is also shown on this page to help the user utilize some of the advanced features offered by the context mediation system. Notice that the tutorial is dynamically generated and changes based on the semantic framework of the application.

Coming back to the example, after the user has entered the appropriate modifier values, the contexts page would show something like figure 3.15:

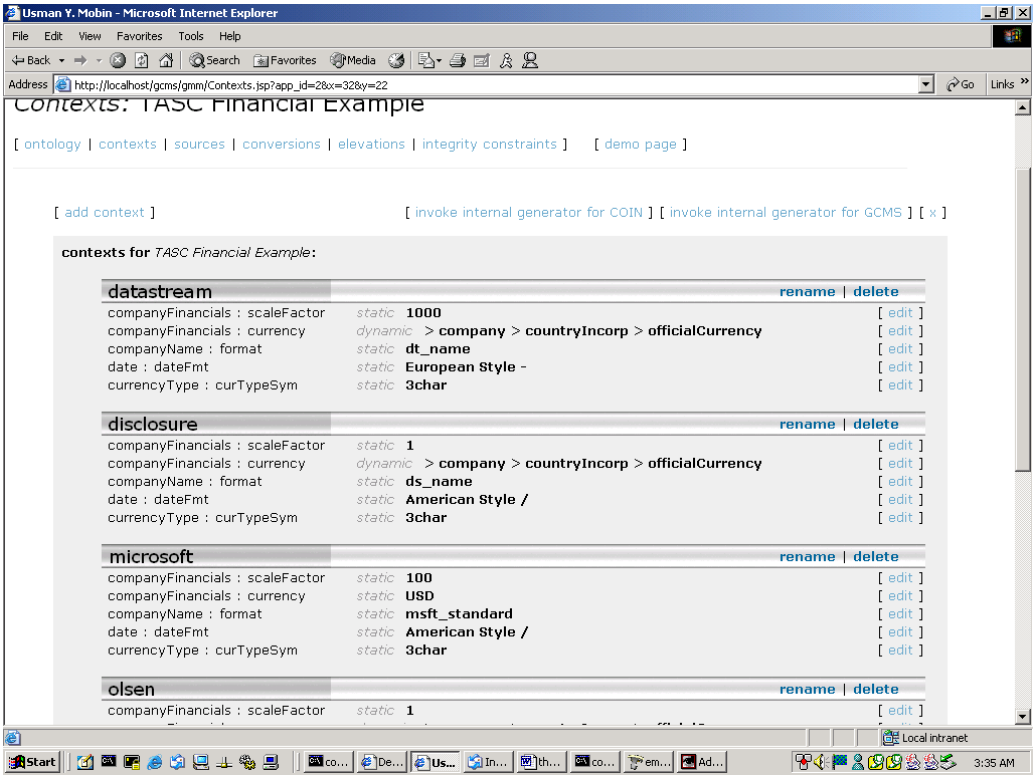

Figure 3.15: user specified context

Next, the user clicks on the "invoke internal generator for GCMS" link. This updates the reasoning engine with the information about the new context and displays the generated internal representation on the screen (see figure 3.16)

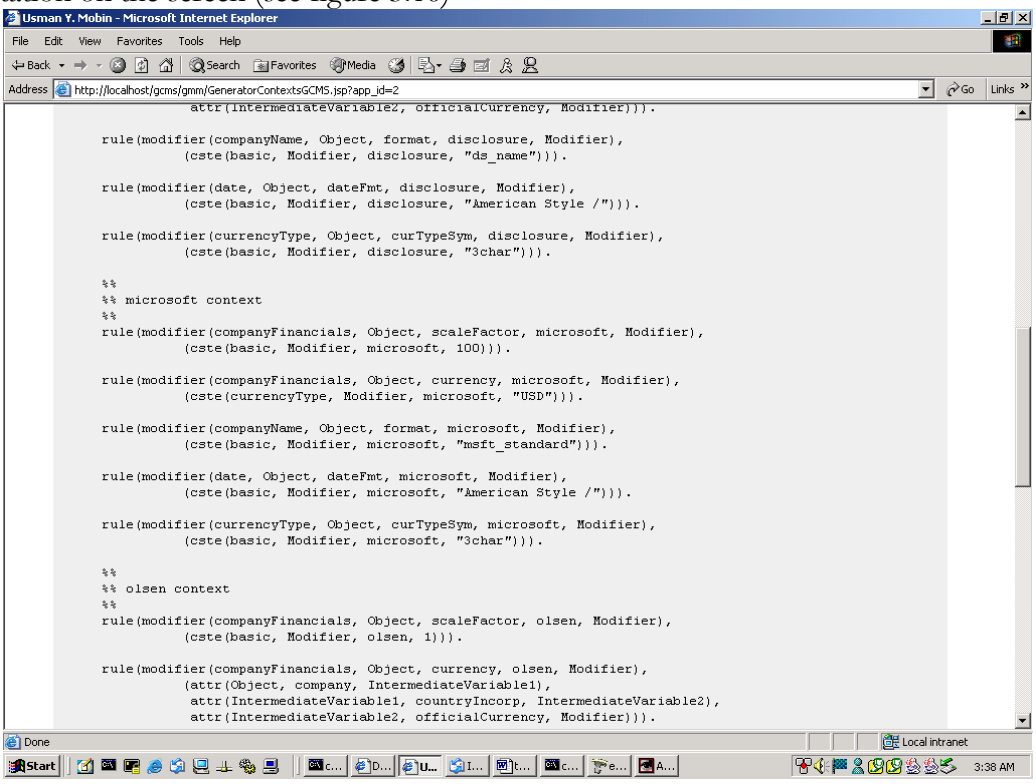

Figure 3.16: updating the reasoning engine with the new context information

Now, the user needs to create a table that provides the system with conversion information to and from Microsoft's vocabulary for company names (named "msft_standard" in the context above). For the time being, we will create a table that maps names in the Microsoft standard to ones in the Disclosure standard. Let's call this table "name_map_msft_ds"

```
SQL*Plus: Release 8.1.7.0.0 - Production on Sun Feb 3 03:43:53 2002

(c) Copyright 2000 Oracle Corporation.  All rights reserved.

Connected to:
Personal Oracle8i Release 8.1.7.0.0 - Production
JServer Release 8.1.7.0.0 - Production

SQL> create table name_map_msft_ds (
  2      msft_names    varchar(100),
  3      ds_names      varchar(100)
  4  );

Table created.
```

As a start, we will add British Telecom to this table. Recall that Microsoft's context wants to refer British Telecommunications Corporation as "British Telecom" while the Disclosure context refers to the same company as "BRITISH TELECOMMUNICATIONS PLC"

```
SQL> insert into name_map_msft_ds
  2  (msft_names, ds_names)
  3  values
  4  ('British Telecom', 'BRITISH TELECOMMUNICATIONS PLC');

1 row created.

SQL> commit;

Commit complete.
```

Of course, by adding a new table to the system, we have changed the sources information. We will need to add this relation to the sources description to make it available in the application. The Graphical Metadata Management system makes this very easy. The user just needs to go to the sources editor by clicking on the "sources" link and then click on the "add relation" link for the database oracle. This brings up the "relation addition" form shown in figure 3.17

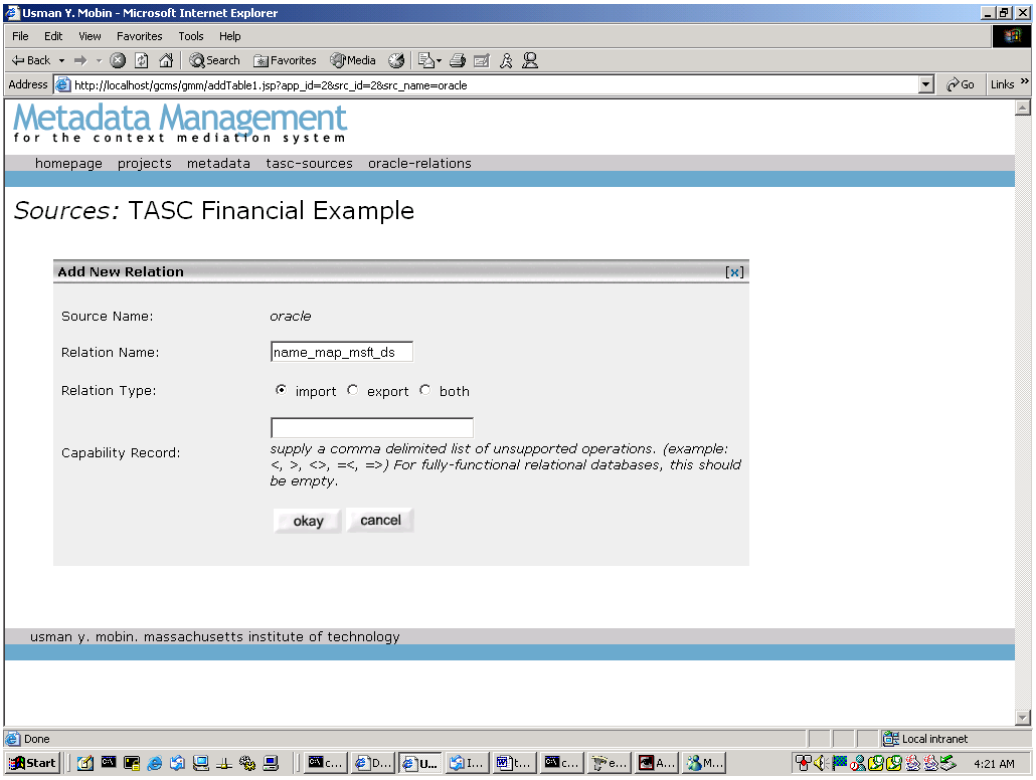

Figure 3.17: relation addition form

Once we add the relation, it now appears on the sources editor but without any columns underneath it. To add columns, we click on the "add column" link next to the relation and add both the columns one after the other:

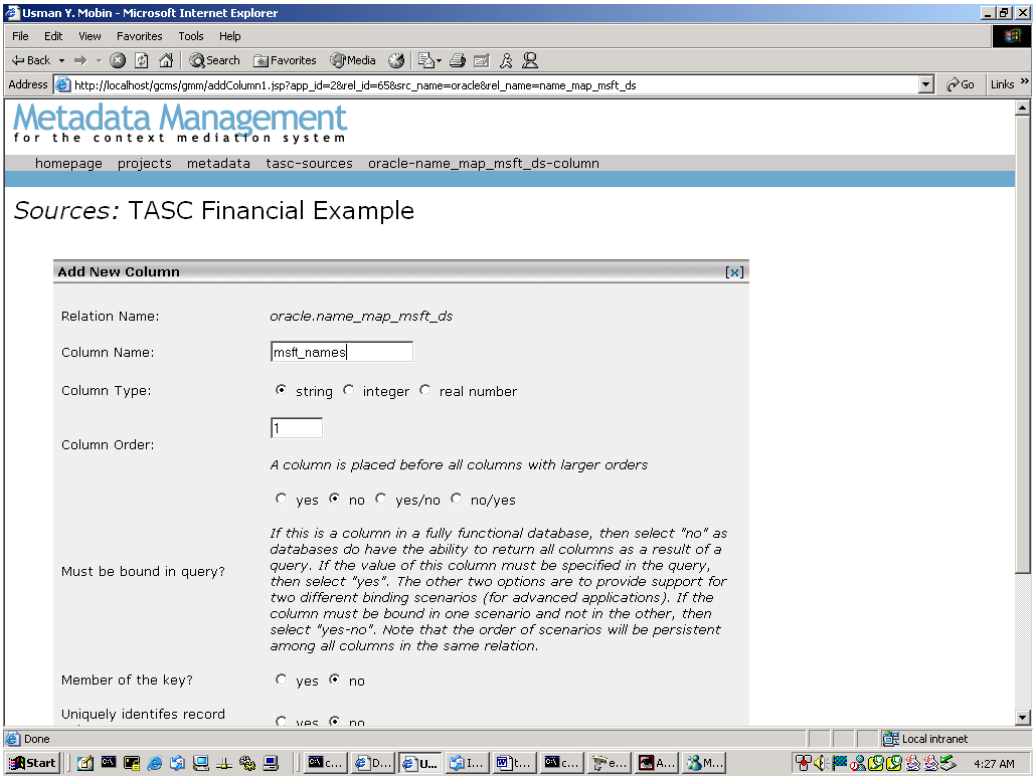

Figure 3.18: column addition to a relation

Once, we have added both the columns for the source, our source description is complete. We can then click on the "invoke internal representation generator for GCMS" link to update the appropriate internal representation file.

However, specifying the sources description is not enough, we need to provide elevation information for our new table. Again, the Graphical Metadata Management system makes this a trivial job. All the user needs to do is go to the Elevations editor we mentioned earlier and showed in figure 3.13 From here, the user can scroll down to the place where "name_map_msft_ds" is mention. First, the user needs to set the context of this table. To do so, the user clicks on the "change" link next to context and comes to the edit context form shown below.

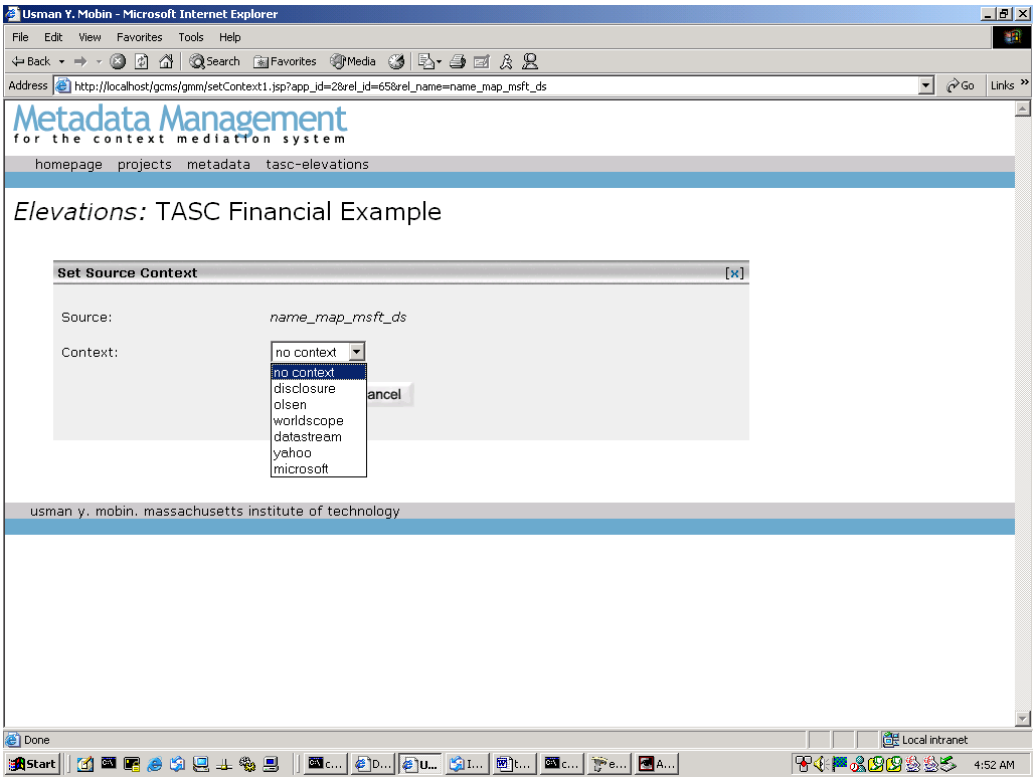

Figure 3.19: context specification for sources

Notice that the context specification form automatically generates a drop box with all the available contexts to choose from. This makes the task of the user easier and also ensures that a non-existent context cannot be specified for a source, preventing metadata corruption. Also, if the context name is renamed from the contexts page, all references to that context are automatically updated elsewhere in the metadata. Likewise, back at the elevations page, the user can specify the semantic type "basic" to elevate both the columns to. Notice again that the column elevation form already has a dynamically generated drop box with all the semantic types define in the application using the ontology editor of the Graphical Metadata Management system.

Now the final steps. The user needs to specify a conversion function for the semantic type companyName with respect to the format modifier (recall, that the user had selected "msft_standard" as the modifier value for companyName:scaleFactor). The Graphical Metadata Management system makes this task quite a bit easier than before. The user can click on the

"conversions" link (from the temporary applications page, or from any other editor's main page) to come to the conversion functions editor.

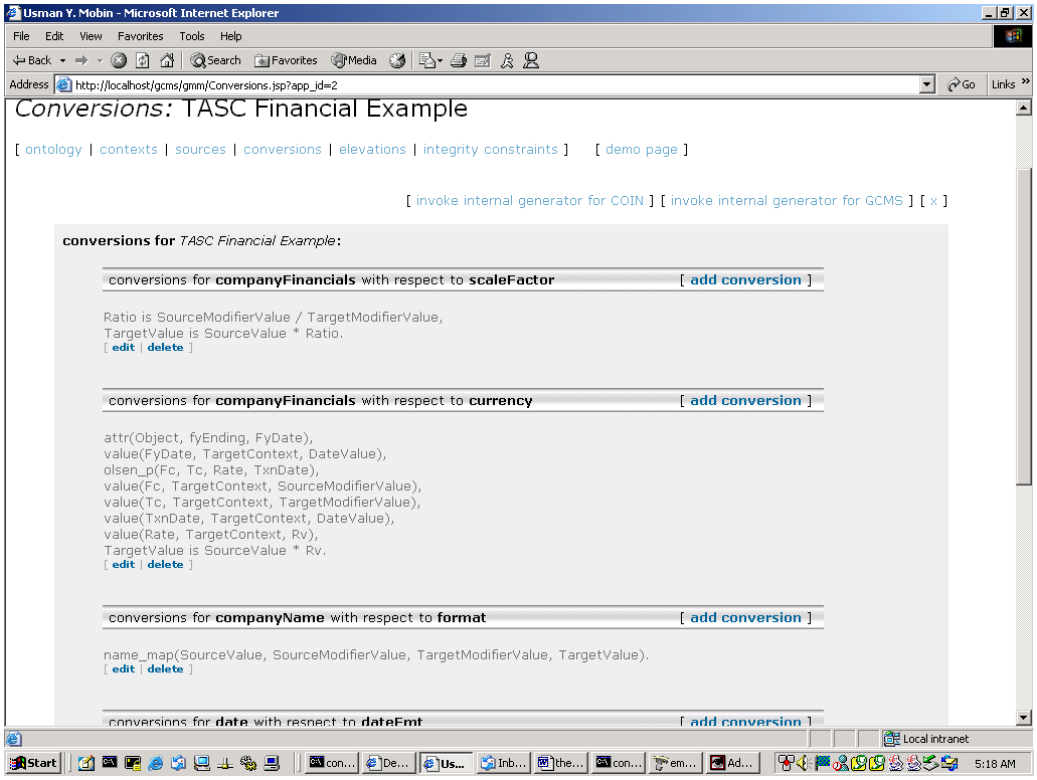

Figure 3.20: the conversion functions editor

Notice that there is one conversion functions already specified for the semantic type companyName with respect to modifier format.

```
name_map(SourceValue, SourceModifierValue, TargetModifierValue, TargetValue).
```

To be able to use this to make conversions from "msft_standard" to "ds_name," we need to be able to add a rule in the system like:

```
name_map(V1, "msft_names", "ds_name", V2) :-
      name_map_msft_ds(V1, V2).
```

And likewise, for the reverse conversion, we'll need a rule of the form:

```
name_map(V1, "ds_name", "msft_names", V2) :-
      name_map_msft_ds(V2, V1).
```

Details can be found in [9]. However, there is one difference between the descriptions of [9] and our current mediation system. The rules described in [9] are for the Context Interchange Mediator[2], COIN, while our current implementation, the Global Context Mediation System, GCMS, uses a deterministic variation of the rules used by [2]. Thus, the general format for a rule:

```
<rule-coin> ::= <head-clause> :- <body-clauses>.
<body-clauses> ::= <body-clause>, <body-clauses>
```

```
                    | <body-clause>
```

On the other hand, the GCMS metadata rules are:

```
<rule-gcms> ::= rule(<head-clause>, (<body-clauses>)).
<body-clauses> ::= <body-clause>, <body-clauses>
                    | <body-clause>
```

Thus, the two rules for name_map shown above would be written in the Global Context Mediation System as:

```
rule(
      name_map(V1, "msft_names", "ds_name", V2),
            (name_map_msft_ds(V1, V2))).

rule(
      name_map(V1, "ds_name", "msft_names", V2),
            (name_map_msft_ds(V2, V1))).
```

These need to be entered in the "custom abduction-time code" section of the metadata. This editor can be invoked by going to the elevations editor and then clicking on the "custom abduction-time code" link. Finally, generating the internal representation for the elevations by clicking on the "invoke internal representations for GCMS" from the elevations editor page. The user also needs to invoke the internal representation generator on the conversion functions page. This concludes the task of specifying one's own context that uses a custom vocabulary for company names. To add conversions for specific company names, all the user would need to do in the future is to populate the name_map_msft_ds table with the appropriate name pairs. The reason for choosing this non-trivial example of different vocabularies as opposed to a simpler example with only scale factor differences[13] was to illustrate the power of the Graphical Metadata Management system and the generality of the context mediation system. Of course, the custom abduction-time code option allows the user to make use of the power of a programming language[13], if needed, to make the system extremely powerful.

Now, the user can actually make queries in, and get results in, his or her own context (context "microsoft" in this case). The context mediation system would accept queries in the user context, queries like:

```
select net_income, total_assets
from disclosure
where company_name = 'British Telecom';
```

and return the results in the user's context. This is precisely what makes the context mediation system such a remarkably powerful tool in a world that contains a myriad of context-bearing entities and data sources.

3.4 Modules of the System

As of this writing, the Graphical Metadata Management system consists of 72 Java source files, totaling over 15,000 lines of code. The system also defines, and makes use of numerous database tables, sequences, indices, and integrity constraints. This section starts to describe the top-level structure of the system. Subsequent chapters describe the source files of the system, the interfaces

[13] would not have required any changes to sources and elevations and custom abduction-time code.

defined, and the data structures shared. The primary audience of this section and future chapters are developers of other parts of the context mediation system, as well as the future developer or maintainer of the Graphical Metadata Management system. We begin to give an in-depth knowledge of the details of the Graphical Metadata Management system and equip the future developer with the knowledge necessary to customize[14] and extend this system to adapt to future needs or augmentations. Also, subsequent chapters describing the various subsystems provide the reader with some useful end-user oriented documentation in addition to the technical documentation.

For our convenience, we will divide the system into eleven top-level modules: The applications page; the editors for ontology, sources, contexts, elevations, and conversions; and the generators for ontology, sources, contexts, elevations, and conversions.

3.4.1 The Applications Page

The applications page displays the list of applications in the system. This page is a temporary placeholder for the much-more-functional applications-browser and query-builder application under development at our research group. Thus, emphasis has not been given to exposing functionality to the user through this page. The applications-browser and query-builder will allow users to create, save, edit, and copy their applications in much the same way as the editors presented in this thesis allow users to create, save, edit, and copy metadata within applications. Currently, the applications page looks something like:

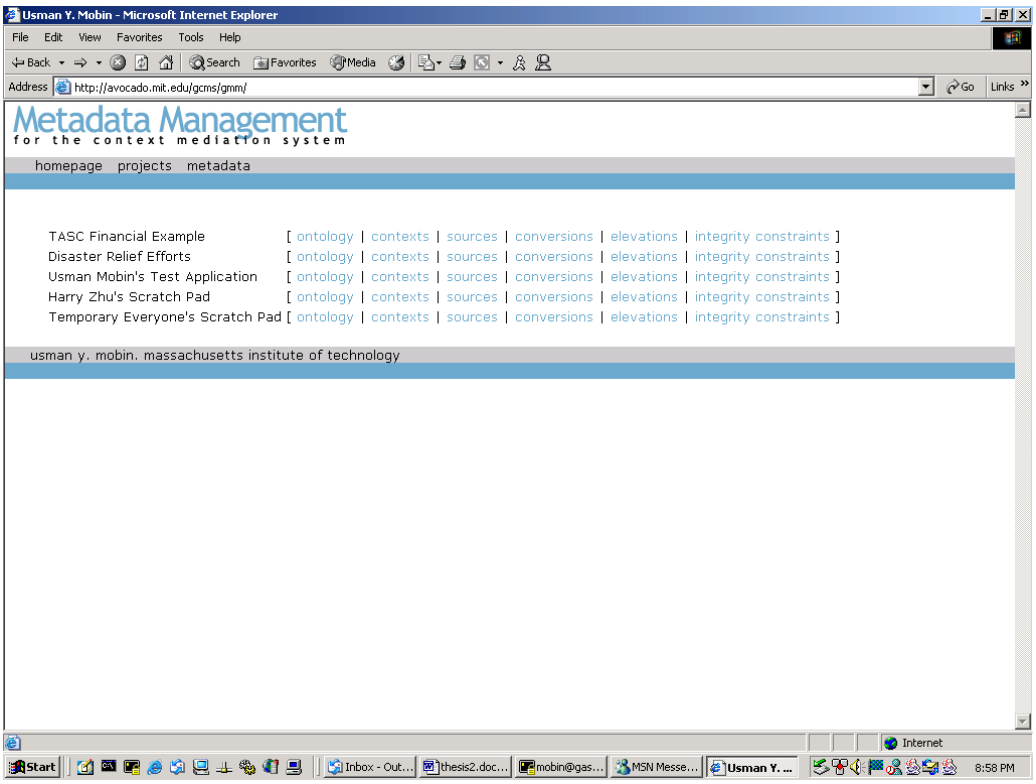

Figure 3.21: The Applications Page

3.4.1.1 The Underlying Applications Architecture

[14] Including, but not limited to, the ability to support future context mediation reasoning engines.

For the benefit of the future developer, or anyone who wishes to gain a detailed insight into the functionality of the applications subsystem, we now present a precise definition of what data is stored in the system and what relationships exist between the various entities in the system. The complete object-model would be too large to fit in any one page, so we have broken it down into six parts. The top-level object model pertaining to the Graphical Metadata Management system is shown in figure ? below. It tells us precisely what entities can exist in the system at any one time, and what phenomena need to be handled by any physical implementation of the data model of the system. Also, the object model tells us about the limitations of the system[15]

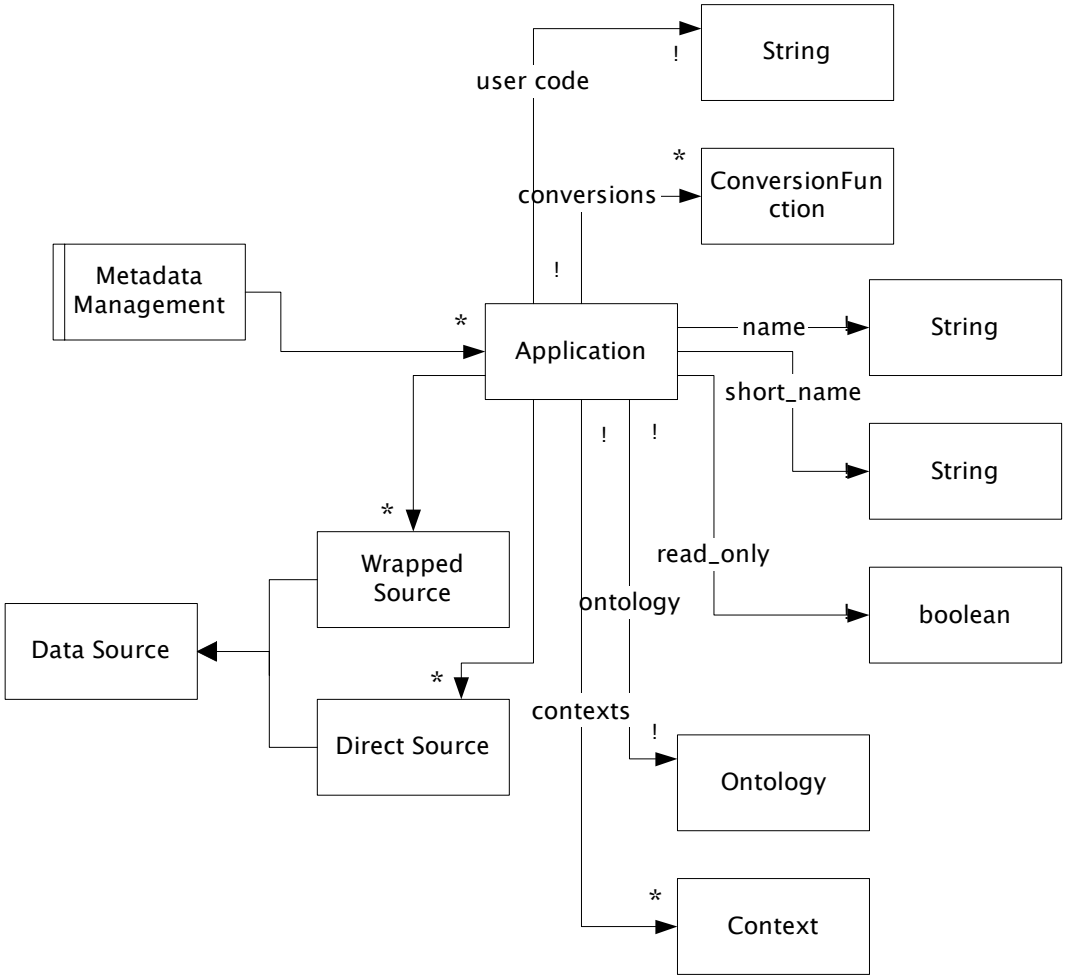

Figure 3.22: top level object/data structure of the system

To gain more insight into the interpretation of such specification as depicted in the diagram above, the reader is referred to [30]. The marks (*), (+), (!), (?) represent cardinalities of "zero or more," "one or more," "exactly one," and "zero or one." Thus, as an example, the mark of (*) on the arrow-head from 'Application' to 'Context' depicts the fact that one application can have zero or more contexts. The (!) on the base of that arrow represents that a context belongs to one application (a limitation future versions of the system might want to relax).

More than just providing a big picture of the relationships between the entities in the system, the above diagram allows us to arrive at an efficient and normalized[22] relational data model[21] for

[15] Which future developers of the system might wish to relax in their extensions of the system.

the system. Based on this, the information relating to applications is stored in the table coin_apps. The oraclized[10] data definition language[31] instruction for this table is given below

```
create table coin_apps (
      app_id      integer primary key,
      app_short_name    varchar(50) not null,
      app_name          varchar(500) not null,
      read_only_p       varchar(1) default 'f'
);
```

The future developer will need to do either of two things to support additional functionality pertaining to applications: either the developer will need to alter this table to add new columns, or create new tables which have a column that references the primary key of this table. Also, for the developer's convenience, a sequence has been defined which allows us to generate the primary keys for adding applications to this table.

```
create sequence coin_apps_app_id_seq start with 2;
```

Thus, for the addition of a new application, the following data modification language[31] instruction needs to be issued:

```
insert into coin_apps
(app_id, app_short_name, app_name, read_only_p)
values
(coin_apps_app_id_seq.nextval, 'application',
'Our sample application', 'f');
```

4 Ontology

This section focuses on the design, implementation, and usage of the ontology editor, its underlying data model, and the generators that generate the internal representations from the database tables populated by the ontology editor.

4.1 The Ontology Editor

The ontology editor allows users to author and edit ontologies for their applications. The ontology forms the semantic framework based on which the reconciliation of heterogeneous sources takes place[1]. We have discussed this earlier in more detail.

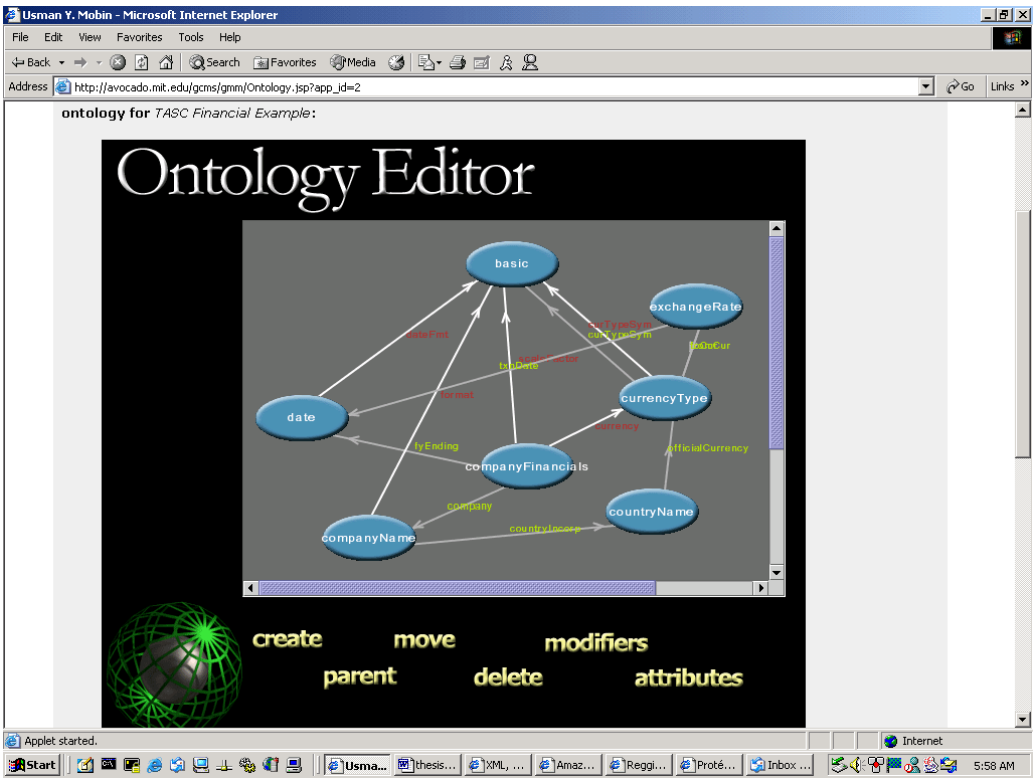

Figure 4.1: The ontology editor

Figure 4.1 shows a screen shot of the ontology editor for the TASC Financial Example application first developed for the initial context interchange mediator[5] and later ported to the Global Context Mediation System.

4.1.1 Usage Guide

The ontology editor can be invoked by clicking on the "ontology" link from the applications page. The editor requires the Java 2 Plug-in[41], just like Stanford's protégé[18] does, to function from within a web browser. This usage guide is reproduced at the bottom of the ontology editor web page.

4.1.1.1 Ontology Color Guide

The blue ovals represent semantic types. the dashed arrows represent parental relations with the arrow pointing at the parent. The solid white arrow represents modifiers with modifier name on the arrow in a dark red and the modified semantic type as the source of the arrow. A gray arrow represents attributes likewise.

4.1.1.2 Semantic type creation and deletion

To create a semantic type, click on the create button. This puts the editor in the "create" mode. Now when you move your mouse over the canvas area, a rectangular outline will appear showing where the new semantic type would appear if you clicked the mouse. If this outline is red then you are either overlapping with an existing semantic type or too close to it. Clicking will pop-up a dialog box requesting the name of the new semantic type. You should choose a unique name for the semantic type. If a type with the requested name already appears on the canvas then the request to create the new semantic type will be rejected. (Note: the canvas automatically resizes as the ontology size grows so you don't have to worry about the initial canvas size).

To delete a semantic type[16],. click on the delete button and then select the semantic type to be deleted. Answer yes on the confirmation dialog

4.1.1.3 Modifier Addition

To add a modifier: click on the "modifiers" button. First select the semantic type that is being modified, and then select the semantic type of the modifier itself. Name the modifier in the dialog box. If a modifier for the initially selected semantic type already exists with the name you request, then the old modifier's target semantic type will be overwritten.

4.1.1.4 Attribute Addition

To add an attribute, click on the "attributes" button. First select the semantic type that is being attributed, and then select the semantic type of the attribute itself. A dialog box will pop-up asking you for the attribute's name. Name the attribute in this dialog box. If an attribute for the initially selected semantic type already exists with the name you request, then the old attribute's target semantic type will be overwritten.

4.1.1.5 Specifying parental relations

To specify a parent for a semantic type, click on the "parent" button. First select the semantic type whose parent you want to specify, next click on the semantic type desired to be the parent. This should create a parental relation. Notice, that the ontology editor imposes a single inheritance model.

To remove a parental relation, while in the "parent" mode, click on the child semantic type and then click on it again (to make it its own parent, and thus effectively removing the parental relationship).

4.1.1.6 Modifier and Attribute Deletion

To delete a modifier, click on modifiers button to go to "modifiers" mode and then select the semantic type whose modifier is to be removed. Click again on this semantic type and then type the

[16] This should be used with extreme caution

name of the modifier in the pop-up box. This will remove the modifier. Similar instructions for attribute removal.

4.1.1.7 Moving Semantic types

To move semantic types: click on the "move" button if not already in the "move" mode and then select the semantic type to be moved. An outline rectangle will appear wherever you move the mouse, click again to move the semantic type to the rectangular box's location. Notice that the rectangular box becomes red when the target location is not empty and thus the semantic type cannot be moved there.

4.1.2 Dynamic Canvas Resizing

The ontology editor canvas (the place where the user draws the ontology) automatically resizes itself when the system feels that the user ontology needs more space. The canvas resizing is determined by the value of the extensionMargin variable in the OntEdCanvas class.

4.1.3 Graphics-related variables of interest to the developer

The ontology editor canvas offers considerable room to the future developer for customization. All the developer has to do is to initialize many of the class variables in the OntEdCanvas class to the desired values. These variables of interest are:

```
private final int bubbleHeight = 52;
private final int bubbleWidth = 98;
private int preferredWidth = 500;
private int preferredHeight = 250;
private int extensionMargin = 150;

private ImageIcon semtypeIcon;
private ImageIcon semtypeIconHighlighted;
private ImageIcon mazeBackgroundIcon;
private Font semtypeFont;
private Font modifierFont;
private Font attributeFont;
private Stroke lineStroke;
private Stroke dashStroke;
private float[] dash1 = {5.0f};

private int canvasHeight;
private int canvasWidth;

private Color selectboxColour;
private Color semtypeFontColour;
private Color parentArrowColour;
private Color modifierArrowColour;
private Color modifierTextColour;
private Color attributeArrowColour;
private Color attributeTextColour;
private Color parentArrowAntialiasColour;
```

Also, from the constructor to the OntEdCanvas class, here are the values these variables are initialized to, in the current version:

```
public OntEdCanvas(Image bubbleImage,
                   Image bubbleImageHighlighted,
```

```
            Image mazeBackgroundImage) {
super(new BorderLayout());

this.setBackground(new Color(110,110,110));
mode = OntEditor.DEFAULT_MODE;
g_canvas = getGraphics();
semtypeFont = new Font("Times", Font.BOLD, 12);
modifierFont = new Font("Times", Font.BOLD, 10);
attributeFont = new Font("Times", Font.BOLD, 10);
lineStroke = new BasicStroke(2, BasicStroke.CAP_ROUND,
                        BasicStroke.JOIN_ROUND);
dashStroke = new BasicStroke(2, BasicStroke.CAP_ROUND,
                        BasicStroke.JOIN_ROUND,
                        10.0f, dash1, 0.0f);
types = new Vector();

this.setPreferredSize(new Dimension(preferredWidth, preferredHeight));
this.revalidate();
canvasListener = new CanvasEventListener();
this.addMouseListener(canvasListener);
this.addMouseMotionListener(canvasListener);
mouseX = 0;
mouseY = 0;
mouseAdjX = 0;
mouseAdjY = 0;
subMode = 0;
displaySelectbox = false;
overrideBoxHide = false;
selectboxColour = Color.lightGray;
semtypeFontColour = Color.white;
parentArrowColour = Color.white;
modifierArrowColour = Color.white;
modifierTextColour = new Color(178,48,48);
attributeTextColour = new Color(183,233,0);
attributeArrowColour = new Color(180,180,180);
parentArrowAntialiasColour = Color.gray;

semtypeIcon = new ImageIcon(bubbleImage);
semtypeIconHighlighted = new ImageIcon(bubbleImageHighlighted);
mazeBackgroundIcon = new ImageIcon(mazeBackgroundImage);
```

4.1.4 Source Files

The following table lists the source files used by the ontology editor and a short description of their use:

Filename	Extends	Lines	Function
CancelException.java	java.lang.Exception	31	This is thrown when the user clicks on the cancel button in any dialog box.
DatabaseAccess.java	java.lang.Object	56	Used for making connection to the databases. Makes use of the oracle.jdbc.driver.OracleDriver class, which is contained in the classes12.zip file in the same directory.

InputFrame.java	javax.swing.JDialog	158	A generic frame for getting textual input from the user
OntEdCanvas.java	javax.swing.JPanel	1,214	This class implements the actual canvas on which ontologies are drawn
OntEdCellRenderer.java	javax.swing. DefaultListCellRenderer	56	Cell rendering class for lists
OntEditor.java	javax.swing.JApplet	625	The main ontology editor applet.
SemanticType.java	java.lang.Object	50	Abstract data type for a semantic type
SemanticTypeGraphical.java	java.lang.Object	56	Abstract data type for a semantic type also encapsulating graphical information such as rendering coordinates
SemtypeEditFrame.java	javax.swing.JDialog	177	A frame window for editing semantic type s
YesNoFrame.java	javax.swing.JDialog	134	A generic Yes-No dialog box
Ontology.jsp	-	171	The main JavaServer Page which contains the ontology editor

Thus, the ontology editor comprises of 2,778 lines of code in eleven files (18% of the code base).

4.1.5 Underlying Data Structures

We continue the object model we presented in an earlier section on "applications" and show the relevant relationships for the ontology-related entities in the system

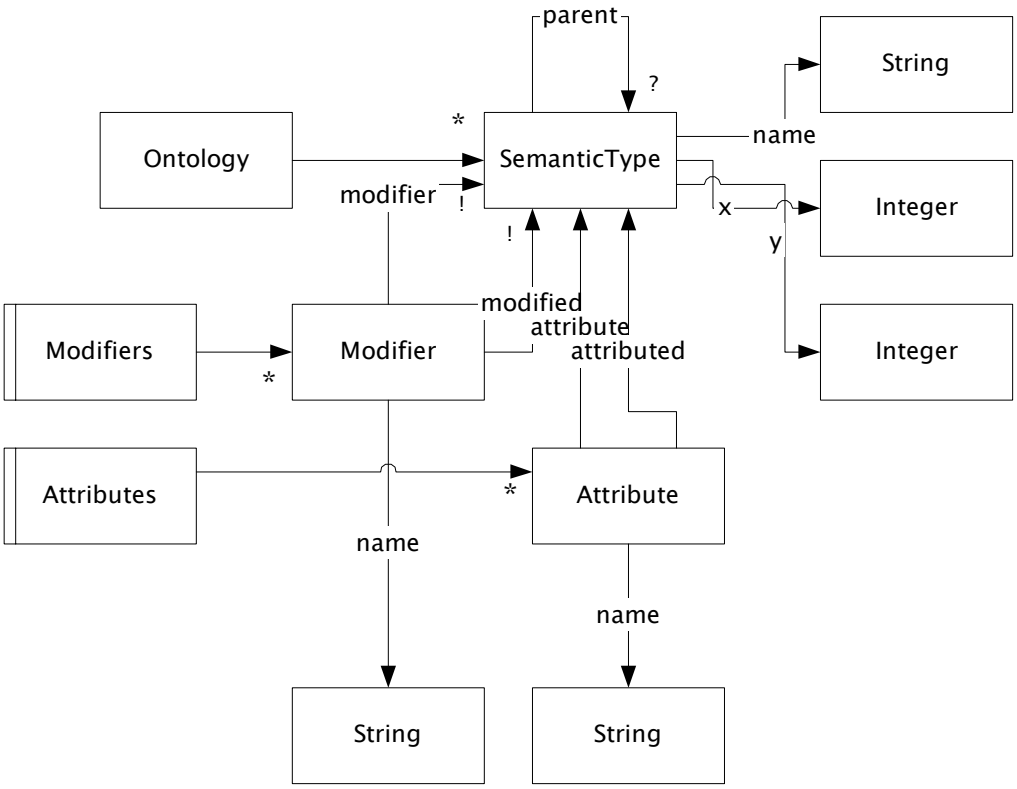

Figure 4.2 Object/data structure of the ontology subsystem

4.1.5.1 Semantic Types

As seen from figure 4.2, each ontology contains zero or more semantic types. We store the information about these semantic types in the table ont_semtypes[17] that is presented below:

```
create sequence ont_semtype_id_seq start with 2;

create table ont_semtypes (
       semtype_id          integer primary key,
       parent_id           integer references ont_semtypes(semtype_id) on
delete set null,
       ontology_id         integer references coin_apps,
       name                varchar(100) not null,
       render_x            integer,
       render_y            integer,
       render_height       integer,
       render_width        integer,
       parent_arrow_x1     integer,
       parent_arrow_y1     integer,
       parent_arrow_x2     integer,
       parent_arrow_y2     integer,
       parent_arrow_curve_depth integer,
       parent_arrow_curve_intensity integer,
       parent_arrow_special_text    varchar(100),
       constraint ont_semtype_unique unique(name, ontology_id)
);

create index ont_semtypes_idx1 on ont_semtypes(semtype_id, parent_id);
```

[17] Please note that all the tables for the Graphical Metadata Management system are stored in an Oracle schema named "gmm"

4.1.5.2 Modifiers

To ensure normalized data structure[12], the modifiers and attributes are stored in tables of their own. Modifiers are stored in the table ont_modifier_map:

```
create sequence ont_modifier_id_seq start with 2;

create table ont_modifier_map (
       modifier_id             integer primary key,
       modified_semtype        references ont_semtypes on delete cascade,
       modifier_name           varchar(100) not null,
       modifier_semtype        references ont_semtypes on delete cascade
);
```

Each row of this table represents one modifier between modified_semtype and modifier_ semtype (both reference ont_semtypes).

4.1.5.3 Attributes

Attributes are stored similarly to modifiers, in the table ont_attribute_map:

```
create sequence ont_attribute_id_seq start with 2;

create table ont_attribute_map (
       attribute_id            integer primary key,
       attributed_semtype      references ont_semtypes on delete cascade,
       attribute_name          varchar(100) not null,
       attribute_semtype       references ont_semtypes on delete cascade
);
```

4.2 Ontology Generators

The ontology generators read data from the tables described above and generate some representation of this data. The biggest advantage in decoupling the actual editor from the generation of the final form of the data is that it allows the editor to be used as a front-end to many systems (each system requiring the programmer to program a generator). Also, it ensures that if there is a change in the internal representation to be generated then the editor does not need to be changed in response.

There are two internal representation generators provided for the ontology in the current system. These are summarized in the table below:

Filename	Lines	Functionality
GeneratorOntologyCOIN.jsp	189	Generates the internal representation of the domain model required by the context interchange mediator[5]
GeneratorOntologyGCMS.jsp	199	Generates the internal representation of the domain model required by the Global Context Mediation System, which is a successor to the context interchange mediator. In case the future developer would like to program a new generator (in case there is a change in the

	internal representation, or a change in the reasoning engine itself,) he or she should take a look at this file and the SQL queries it makes. Also, this file creates an internal representation file and places it in the value of the "applicationPath" parameter in the servlet context's environment (the web server) thus updating the abduction engine with the new file. For customized functionality on this front, the future developer should, again, look at this file.

Recall that there is a simple transformation for converting a rule in the COIN format to one in the GCMS format. This was discussed earlier in chapter 3.

5 Contexts

This section focuses on the design, implementation, and usage of the contexts editor, its underlying data model, and the generators that generate the internal representations from the database tables populated by the contexts editor.

5.1 Contexts Management Page

The contexts management page allows the user to create, edit, and manage the contexts defined within each application. The notion of contexts has been explained earlier in this thesis and an illustration of how they are used in the reconciliation of semantic heterogeneity has also been demonstrated. However, if the reader would like to gain a precise understanding of the role played by information contained within contexts in the functioning of the system, he or she is referred to [28].

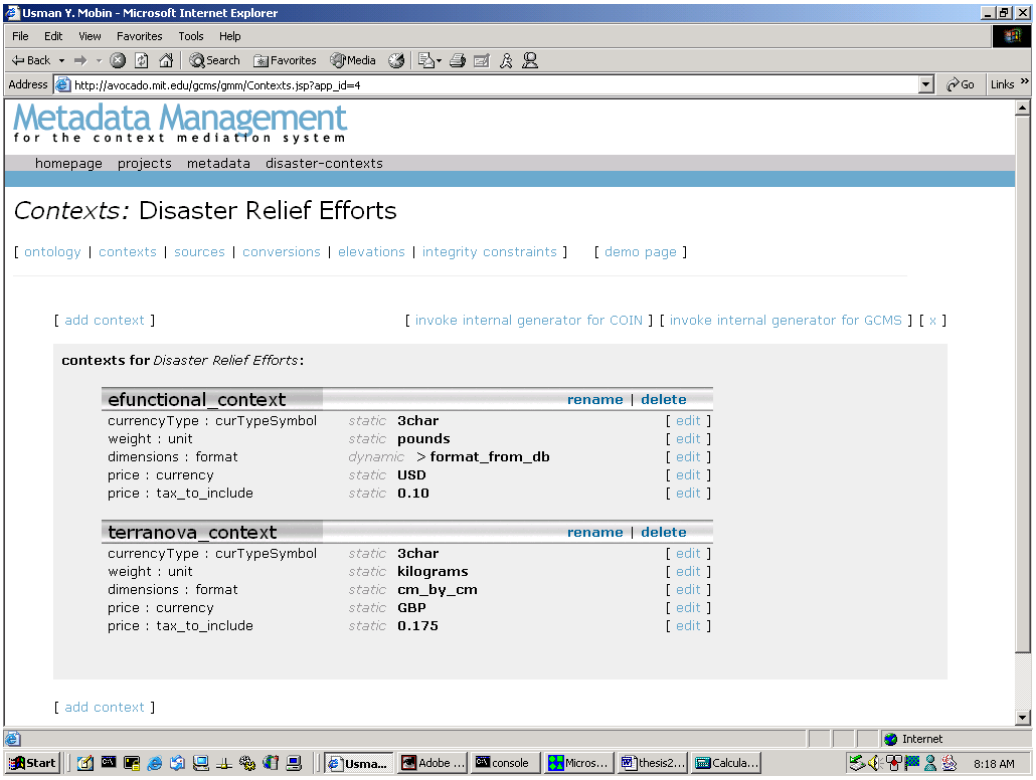

Figure 5.1: Contexts Management Page

Figure 5.1 above displays the contexts management page for an application that contains two contexts.

5.1.1 Usage Guide

To add a context, the user needs to click on the "add context" link above the contexts display panel. The exercise of adding contexts has been explained in much detail in an earlier part of this thesis.

Renaming and deleting contexts can be achieved by clicking on the "rename" and "delete" links, respectively, in the gray bar that displays the name of the relevant context.

Editing the value of a modifier within a context can be achieved by clicking on the "edit" link in line with the entry for that particular modifier. This brings up the "edit modifier value" page. A quick tutorial which appears on this page, is also shown in figure 5.2

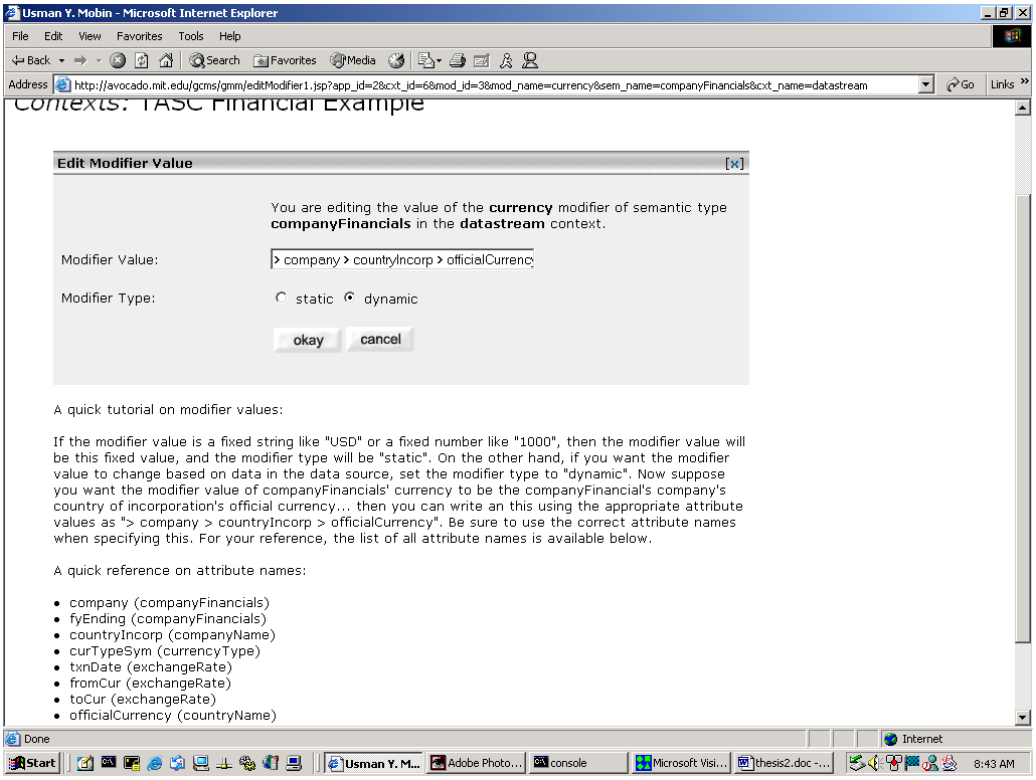

Figure 5.2: Edit Modifier Value Page

5.1.2 Dynamic Modifier Listing

The contexts editor automatically generates a list of modifiers when a new context is created. This was illustrated earlier when the user created the context "Microsoft." However, the important thing to note is that changes to the ontology are automatically reflected in the contexts page. So if the ontology defines a new modifier, all contexts automatically get a slot for the newly defined modifier[18].

Now, we can explain why we noted earlier that deletion of semantic types should be exercised with utmost care. In deleting a semantic type, we implicitly delete all the attributes and modifiers that it has. Thus, the modifiers would disappear automatically from all contexts as the implicitly deleted modifiers would not exist anymore.

5.1.3 Source Files

The following table lists the source files used by the contexts editor and a short description of their use:

Filename	Lines	Functionality

[18] the initialvalue assigned to this modifier is null in all contexts.

Contexts.jsp	181	This JavaServer page source generates the main context management page for each application. It requires as input the value of variable "app_id" to be bound to the app_id of the application (the same app_id used in the coin_apps table, of course).
addContext1.jsp	162	This page shows the "add context" form. The user can click on cancel to go back to Contexts.jsp or click on okay to proceed to addContext2.jsp
addContext2.jsp	192	This page creates new contexts based on the values passed to it by addContext1.jsp. Displays a confirmation or error in case of success or failure respectively.
delContext1.jsp	160	Displays the confirmation form before deleting a context. Pressing "okay" proceeds to delContext2.jsp
delContext2.jsp	164	This page deletes contexts based on the values passed to it, by delContext1.jsp, using the HTTP GET method.
editContext1.jsp	166	This page displays the "edit context" form and allows the user to rename the context. Pressing okay on the form proceeds to editContext2.jsp with the new name entered by the user
editContext2.jsp	190	Uses the new name entered by the user in the previous form to rename contexts. Displays a confirmation or error in case of success or failure respectively
editModifier1.jsp	234	Brings up the form that allows one to change the value of a modifier within a context. Pressing okay transfers control to editModifier2.jsp with the user's inputs on the form
editModifier2.jsp	191	Takes the inputs from the previous form to edit the value of a modifier in the database. Displays a confirmation or error in case of success or failure respectively

The contexts editor thus consists of 9 source files with 1,640 lines of code (11% of the Graphical Metadata Management system's code base).

5.1.4 Underlying Data Structures

We continue the object model we presented in an earlier section on "applications" and show the relevant relationships for the contexts-related entities in the system

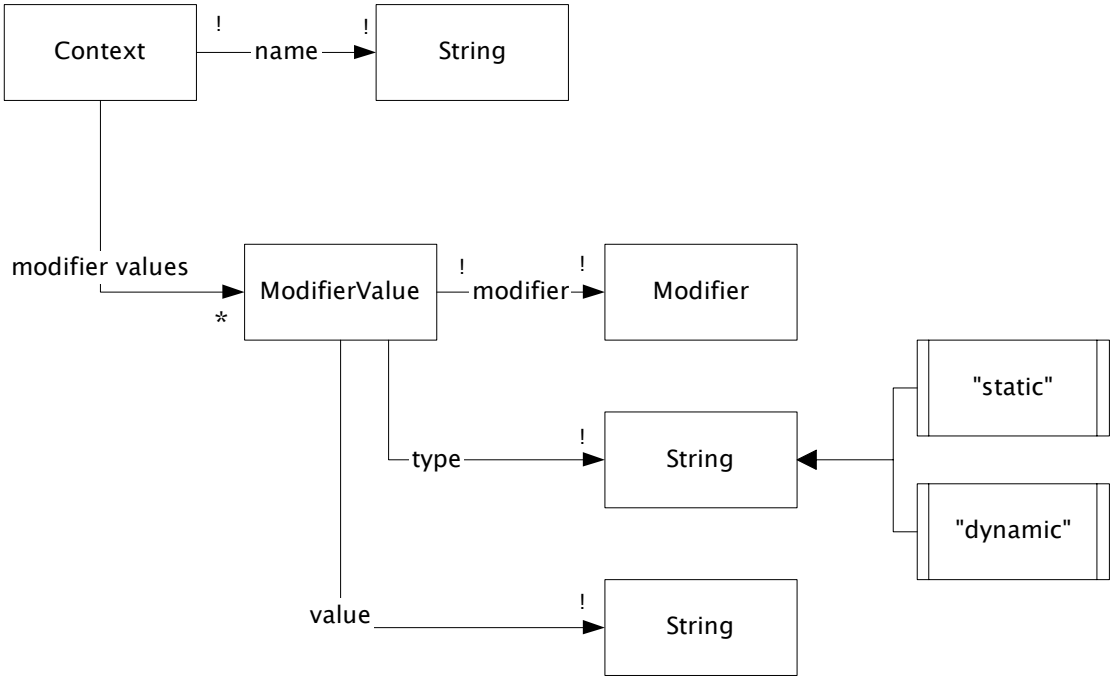

Figure 5.3: Object/data model for contexts subsystem

5.1.4.1 The Contexts Table

From figure 5.3, we see that a context contains a name. Also, from our earlier object model from the "applications" page, we saw that an application contains zero or more contexts. Thus, a context also contains a notion of which application it belongs to. Using this, we arrived at the following table for storing basic information about contexts:

```
create sequence cxt_context_id_seq start with 2;

create table cxt_contexts (
       context_id         integer primary key,
       name               varchar(100) not null,
       owner_id           integer references coin_apps on delete cascade,
       parent_id          integer references cxt_contexts
);
```

5.1.4.2 Table for Modifier Values

The values for modifiers within contexts are stored in the table cxt_modifier_values. This is shown below:

```
create table cxt_modifier_values (
       modifier_id      integer references ont_modifier_map
                        on delete cascade,
       context_id       integer references cxt_contexts on delete
                        cascade,
       modifier_type  varchar(100) default 'static' not null,
       constraint cxt_modifier_type_ck check(modifier_type in
                                        ('static', 'dynamic')),
       modifier_value   varchar(4000) not null
);
```

5.2 Internal Representation Generators

The internal representation generators for contexts read data from the tables described above and generate both the COIN-based, and the GCMS-based internal representations of this data. The biggest advantage in decoupling the actual editor from the generation of the final form of the data is that it ensures that if there is a change in the internal representation then the editor does not need to be changed in response.

There are two internal representation generators provided for the contexts in the current system. These are summarized in the table below:

Filename	Lines	Functionality
GeneratorContextsCOIN.jsp	214	Generates the internal representation of the contexts required by the context interchange mediator[5]
GeneratorContextsGCMS.jsp	229	Generates the internal representation of the contexts required by the Global Context Mediation System, which is a successor to the context interchange mediator.

Appendix A.2 contains an internal representation file for contexts generated by GeneratorContextsGCMS.jsp.

6 Sources

This section focuses on the design, implementation, and usage of the sources editor, its underlying data model, and the generators that generate the internal representations from the database tables populated by the sources editor.

6.1 Sources Management Page

The sources management page allows the user to create, edit, and manage the source descriptions defined within each application.

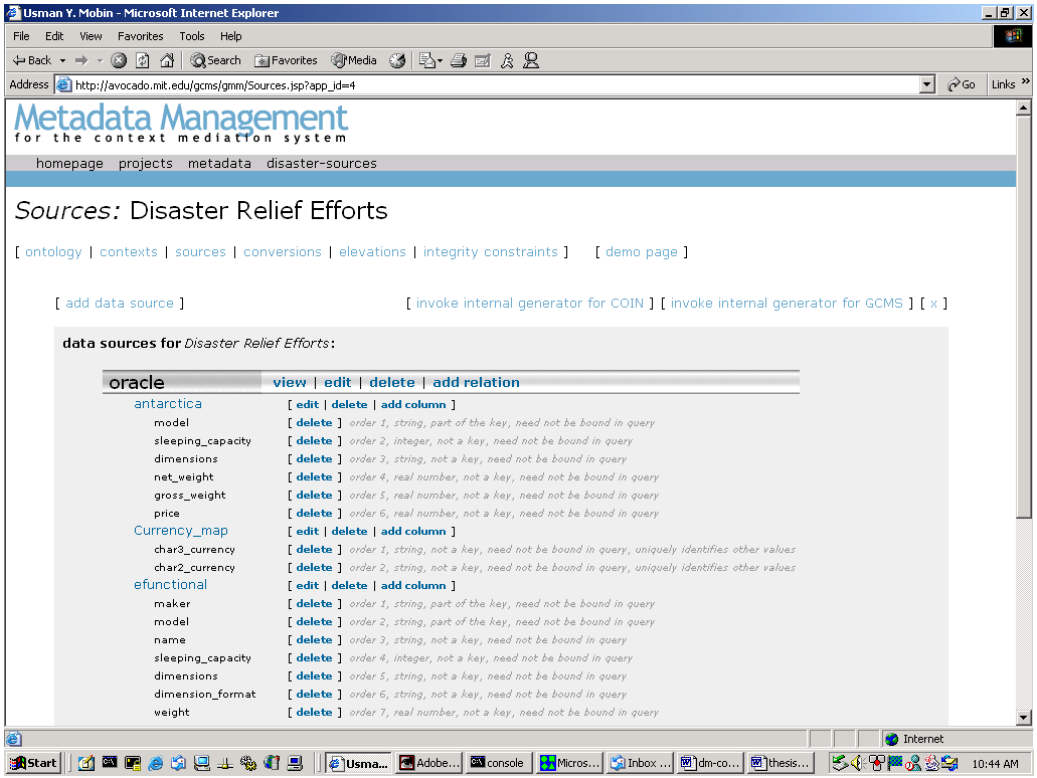

Figure 6.1: Sources Management Page

The main page for the sources editor is shown in figure 6.1. It lists in a structured hierarchical form, the databases, their tables and the columns within these.

6.1.1 A note on terminology

In the real world, we have databases. Within these databases we have tables; and these tables consist of columns. In the Graphical Metadata Management system, we refer to a database as a "data source." And as we will see in our object model diagram for this subsystem of the Graphical Metadata Management system, a data source can either be a physical *database* or a wrapped web source[15]. In a sense, then, the term data source is being used in a more encompassing fashion, to encapsulate both relational as well as wrapped sources of data.

We refer to a table as a "relation." However, we should point out that most of the context interchange literature refers to a table as a "source." This is perhaps to signify that table is the unit in

SQL which you select from. So if you look at figure 6.1 above, all the tables are in the database oracle. Oracle is our data source, and antarctica etc are our "sources." I understand that the terminology might be confusing at first, but the easiest way to look at it is to 'think' that each source has only one table[19].

6.1.2 Source Code Files

The following table lists the source files used by the sources editor and gives a short description of their functionality:

Filename	Lines	Functionality
Sources.jsp	199	This generates the main sources management page shown in figure 6.1. The page shows a hierarchical structure of all physical data available in the application. Expects app_id as input and selects the application based on the value of app_id.
addColumn1.jsp	213	This page generates the form that allows the user to enter the name, type, and other characteristics of a column to add to a table. The results are submitted to addColumn2.jsp
addColumn2.jsp	206	This page takes its input from addColumn1.jsp and attempts to create the requested column. Displays a message on success and error on failure.
addSource1.jsp	182	This page generates the form that allows the user to enter the name, connection information, type (web-wrapped or relational database) and other characteristics of a data source to add to the application. The results are submitted to addSource2.jsp
addSource2.jsp	198	This page takes its input from addSource1.jsp and attempts to create the requested data source. Displays a message on success and error on failure.
addTable1.jsp	187	This page generates the form that allows the user to enter the name, type, and capability record of a table to add to a data source. The results are submitted to addTable2.jsp
addTable2.jsp	197	This page takes its input from addTable1.jsp and attempts to create the requested table Displays a message on success and error on failure.
delColumn1.jsp	162	This page displays the confirmation form when the user first requests the deletion of a column. Pressing cancel takes back to Sources.jsp, while pressing okay transfers control to delColumn2.jsp
delColumn2.jsp	165	This page contains code that deletes the requested column. Displays a success message on success, or error message on failure.
delSource1.jsp	160	This page displays the confirmation form when the user first requests the deletion of a data source. Pressing cancel takes back to Sources.jsp, while pressing okay transfers control to delSource2.jsp

[19] I hope this doesn't further confuse the reader. For some elaboration, the reader might want to read through chapter 3 of [28].

delSource2.jsp	164	This page contains code that deletes the requested data source. Displays a success message on success, or error message on failure.
delTable1.jsp	162	This page displays the confirmation form when the user first requests the deletion of a table. Pressing cancel takes back to Sources.jsp, while pressing okay transfers control to delTable2.jsp
delTable2.jsp	165	This page contains code that deletes the requested table. Displays a success message on success, or error message on failure.
editSource1.jsp	201	Displays the form that allows the user to edit the information stored about a data source. The updated values are passed on to editSource2.jsp for actual processing.
editSource2.jsp	197	Takes the values passed on by editSource1.jsp and makes the requested changes to the data source description.
editTable1.jsp	209	Displays the form that allows the user to edit the information stored about a table. The updated values are passed on to editTable2.jsp for actual processing.
editTable2.jsp	195	Takes the values passed on by editTable1.jsp and makes the requested changes to the table description.
viewSource.jsp	184	Allows the user to view the description of a data source.
viewTable.jsp	187	Allows the user to view the description of a table.

Thus, the sources editor consists of 19 files totaling 3,533 lines of code (23% of the Graphical Metadata Management system's code base).

6.1.3 Underlying Data Structures

We continue the object model we presented in an earlier section on "applications" and show the relevant relationships for the sources-related entities in the system:

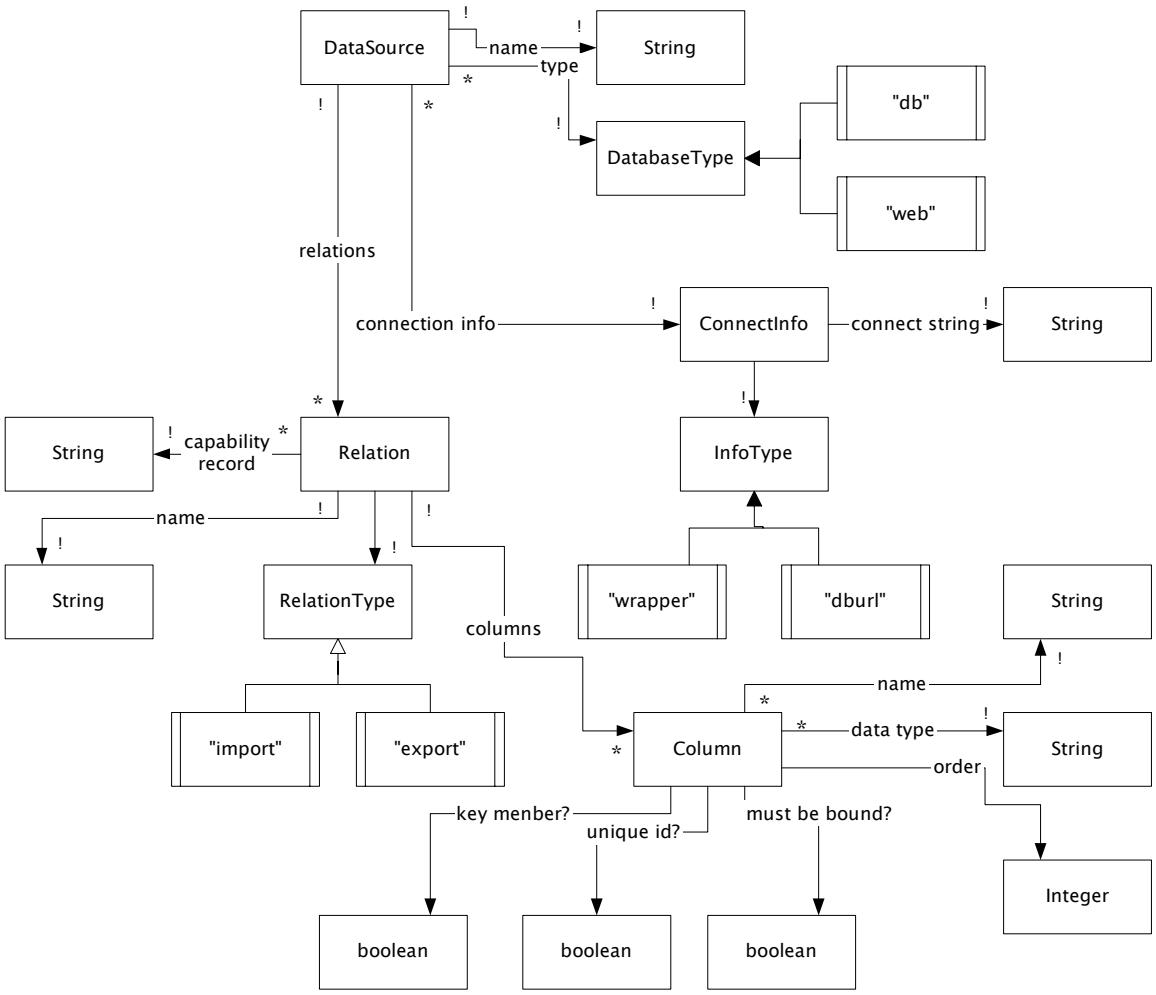

Figure 6.2: Object/data model for the sources subsystem

6.1.3.1 Data Source Descriptions

Of course, figure 6.2 suggests that for a normalized data model, we would need to store data source, relation, and column descriptions in different tables. We store the data source descriptions in the table sch_databases:

```
create sequence sch_database_id_seq start with 2;

create table sch_databases (
       database_id          integer primary key,
       name                 varchar(100) not null,
       owner_id             integer references coin_apps on delete cascade,
       database_type        varchar(100) default 'db' not null,
       constraint sch_db_type_ck check(database_type in ('web', 'db')),
       -- useful for providing wrapper url etcetra
       connect_information      varchar(1000) not null,
       connect_type             varchar(100) default 'wrapper' not null,
       constraint sch_db_connect_type_ck check(connect_type in ('wrapper',
'dburl'))
);
```

Just for the reader's information, the data model presented in sections 6.1.3.1 to 6.1.3.3 underwent the largest number of revisions before arriving at what is presented here[20].

6.1.3.2 Relation Descriptions

Likewise, we store relations in the sch_relations table. Of course, each relation has a notion of which database it belongs to.

```
create sequence sch_relation_id_seq start with 2;

create table sch_relations (
       relation_id      integer primary key,
       name             varchar(100) not null,
       parent_db        integer references sch_databases on delete cascade,
       relation_type    varchar(100) default 'both' not null,
       unsupported_operations  varchar(500),
       constraint sch_rel_type_ck check(relation_type in ('import', 'export',
'both'))
);
```

Notice that the sch_relations table does not directly contain information about which application it belongs to (This is due to a high degree of normalization in our data model). This can e inferred though. Notice that the arrow from DataSource to Relation has a (!) at its base (in figure 6.2). This means that a particular relation belongs to one data source. Now in figure 3.22, all the arrows coming out of "Application" have the (!) mark. This means that a data source belongs to one application. Thus, the application of the relation can be inferred from the application of its parent data source referenced by the parent_db column.

Normalization also means that we need to perform joins in our queries when we want to see, say, all the tables in a particular application (as opposed to the simpler case of seeing all the tables in a particular data source.) For example, to see the names of all the relations in a particular application, we would issue a query like:

```
SQL> select sch_relations.name
  2  from sch_relations, sch_databases
  3  where sch_relations.parent_db = sch_databases.database_id
  4  and sch_databases.owner_id =
  5     (select app_id from coin_apps
  6      where app_short_name = 'tasc')
  7  order by lower(sch_relations.name)
  8  asc;

NAME
--------------------------

CNames
countryIncorp
Currency_map
Currencytypes
datexform
DiscAF
DStreamAF
moneyrates
Name_map_Ds_Ws
Name_map_Dt_Ds
Name_map_Dt_Ws
```

[20] As more integrity constraints were added, cascaded deletion was introduced, columns were added to support a richer set of features etc.

```
NAME
-------------------------

name_map_msft_ds
olsen
quotes
Ticker_Lookup2
WorldcAF
WorldcAFT

17 rows selected.
```

6.1.3.3 Column Descriptions

Finally, the columns are stored in the table sch_columns whose definition is given below.

```
create sequence sch_column_id_seq start with 2;

create table sch_columns (
       column_id         integer primary key,
       relation_id       integer references sch_relations on delete cascade,
       name              varchar(100) not null,
       column_type       varchar(100) default 'string' not null,
       constraint sch_col_type_ck check(column_type in ('string', 'real',
'integer')),
       column_order      integer default 0,
       key_member_p      varchar(1) default 'f',
       constraint sch_key_memb_ck check(key_member_p in ('t', 'f')),
       -- integrity support for non-normalized data sources
       unique_id_p      varchar(1) default 'f',
       constraint sch_unique_id_ck check(unique_id_p in ('t', 'f')),
       binding_restriction    varchar(2) default 'f',
       constraint sch_col_cap_ck check(binding_restriction in ('t', 'f',
'tf', 'ft'))
);
```

6.2 Internal Representation Generators

The internal representation generators for sources read data from the tables described above and generate both the COIN-based, and the GCMS-based internal representations of this data. The advantages of decoupling the actual editor from the generation of the final form of the data have been mentioned before and will not be repeated here.

There are two internal representation generators provided for the sources in the current system. These are summarized in the table below:

Filename	Lines	Functionality
GeneratorSourcesCOIN.jsp	268	Generates the internal representation of the sources required by the context interchange mediator[5]
GeneratorSourcesGCMS.jsp	278	Generates the internal representation of the sources required by the Global Context Mediation System, which is a successor to the context interchange mediator. In case the future developer would like to program a new generator, he or she should take a look at this

	file and the SQL queries it makes. Also, this file creates an internal representation file and places it in the value of the "applicationPath" parameter in the servlet context's environment (the web server) thus updating the abduction engine with the new file. For customized functionality on this front, the future developer should, again, look at this file.

Appendix A.3 contains an internal representation file for source descriptions generated by GeneratorContextsGCMS.jsp. For more details on the structure of the internal representation, the reader is referred to section 4.4 of [9].

7 Elevations

This section focuses on the design, implementation, and usage of the elevations editor, its underlying data model, and the generators which generate the internal representations from the database tables populated by the this editor.

7.1 Elevations Management Page

The elevations management page allows the user to specify the semantic types to which physical columns in our sources elevate to.

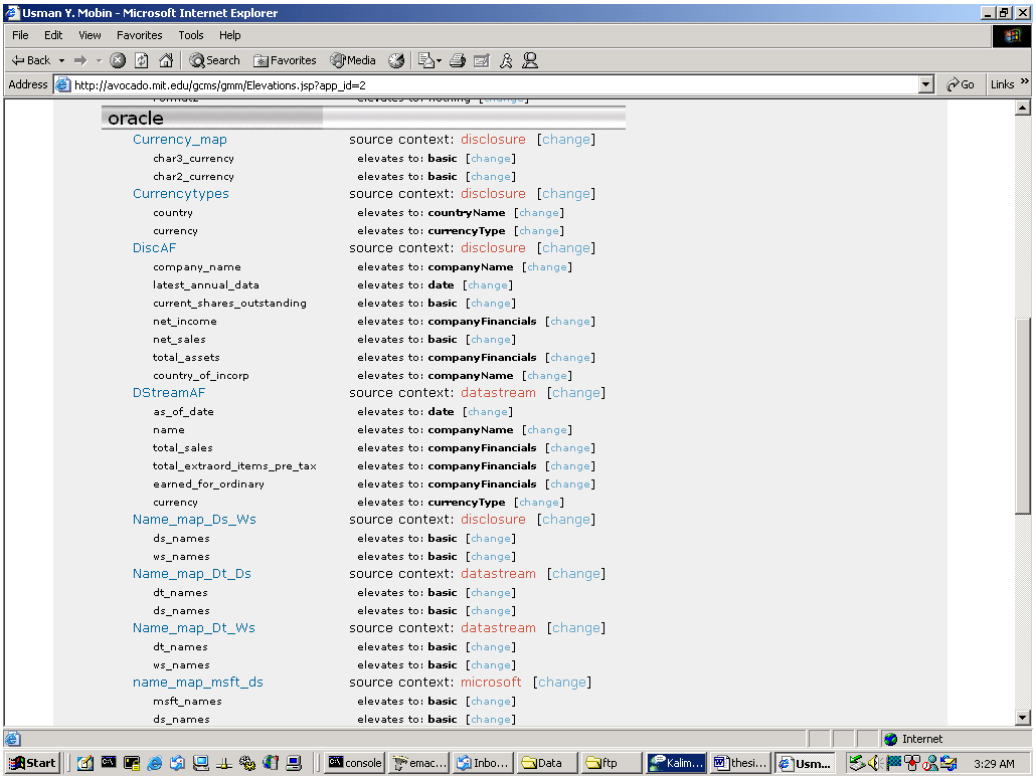

Figure 7.1: Elevations Management Page

This page also allows us to specify the context to which each source subscribes to. To we have an example of this earlier in out motivational guided tour.

The elevations management page automatically gets the list of available source descriptions for the tables described in the later part of chapter 6, and displays them on the screen. Sources for which no context has been specified, have a "source context: no context" message in front of them, others have the context name in place of "no context". Columns that elevate to a semantic type, have the name of the semantic type in bold next to them. This can be seen in figure 7.1.

7.2 Elevations through example

As an example, let us look again at figure 7.1. Notice that the column net_income in the source DiscAF elevates to the semantic type companyFinancials. All this implies is, that at the time of

semantic reconciliation, the value of the net_income column of DiscAF will be subject to modification based on the modifiers of the semantic type companyFinancials.

Assume now that there are two contexts, A and B. Suppose that context A has the value of companyFinancials : scaleFactor set to 1, while context B has this modifier value set to 1000. In this scenario, if DiscAF subscribes to context A, then whenever the value of net_income will be requested in context B, it will be subject to a division by 1000.

7.3 Internal Convention

This section describes some internal conventions followed by the Graphical Metadata Management system. This section should not concern the normal user of the system.

Firstly, the underlying system represents relations as relation_name(column1, column2, …, column-n) in much the same way as Prolog[13]. Thus, the relation DiscAF can be referenced to in the underlying system using

```
'DiscAF'(C1, C2, C3, C4, C5, C6, C7).
```

Also, it is a notational convention followed by the system that the elevated version of this relation has the same name but with "_p" appended to it[21]. Thus, the elevated axiom for DiscAF_p will look like:

```
rule(
  'DiscAF_p'(
     skolem(companyName, C1, disclosure, 1,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(null, C2, disclosure, 2,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C3, disclosure, 3,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C4, disclosure, 4,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C5, disclosure, 5,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C6, disclosure, 6,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyName, C7, disclosure, 7,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7))),
   ('DiscAF'(C1, C2, C3, C4, C5, C6, C7))).
```

Of course, the user never has to worry about writing these elevation rules as they are automatically generated. However, it might be important for advanced users to be familiar with this underlying mechanism.

7.4 Custom Abduction-time Code

The elevations page also has a link that allows the user to author custom abduction-time code for the system. These are not required of the user but are only there to facilitate the advanced user who might want to use the full power of the underlying system. Custom abduction-time code has to be authored in the prolog style and can directly reference prolog[13] constructs and the underlying metadata rules available to the reasoning engine. Of course, the rules must be transformed from the

[21] In case the reader is wondering about the significance of "_p" then it just refers to "prime," a mathematical convention of sorts in naming variables related to other variables.

pure prolog style to the GCMS-rule format to be useful to the Global Context Mediation System's reasoning engine. The custom abduction-time code for an application is stored in the elv_custom table:

```
create table elv_custom (
      owner_id    integer not null references coin_apps,
      custom_body varchar(4000)
);
```

The advanced user might want to refer to [9], [13], and [28] to be able to use this feature to the fullest. Also the user might want to review the earlier discussion on GCMS-style rules, also repeated in the next section.

7.5 Attribute Rules

The elevations page has a link that allows the user to enter and edit attribute rules in the system. These are very easy to do and are explained in the last paragraph of section 4.5.2 of [9]. Of course, the only difference is that the entered rules should be made GCMS-style. We have discussed this before. The general way of converting rules to GCMS style is presented again, just for convenience. The normal format for a prolog rule is:

```
<rule> ::= <head-clause> :- <body-clauses>.
<body-clauses> ::= <body-clause>, <body-clauses>
                 | <body-clause>
```

On the other hand, the GCMS style rules are:

```
<rule-gcms> ::= rule(<head-clause>, (<body-clauses>)).
<body-clauses> ::= <body-clause>, <body-clauses>
                 | <body-clause>
```

Thus, the following prolog:

```
(b,c,d) :- e(f), g(h,i).
j(k,l,m).
```

would be converted to GCMS-style as:

```
rule(a(b,c,d),(e(f),g(h,i))).
rule(j(k,l,m),(true)).
```

The attribute rules pertaining to an application are stored in the elv_attrs table. The data definition language instructions for this table are presented below.

```
create table elv_attrs (
      owner_id    integer not null references coin_apps,
      attrs_body  varchar(4000)
);
```

7.6 Details of the Elevations Page

This section describes the source code (JavaServer Pages) of the elevations editor and describes the underlying data structure.

7.6.1 Source Code for Elevations editor

The following table lists the source files used by the elevations editor and gives a short description of their functionality:

Filename	Lines	Functionality
Elevations.jsp	208	This is the main elevations management page. It displays all the sources described in the system, their contexts, and the elevations schedule for their columns. This is illustrated in figure 7.1. The file expects as input the variable "app_id" and displays the elevations information and the editor for the particular application that has its app_id in the coin_apps table equal to the "app_id" passed on to this page as input.
editAttrs1.jsp	165	This page invokes the form that allows the user to edit attribute rules[9]. Changes are submitted to the page editAttrs2.jsp for further processing
editAttrs2.jsp	182	This JavaServer page takes the new attribute rules passed on to it by editAttrs1.jsp and replaces the old set of rules in the database with this new one.
editCustom1.jsp	165	This page invokes the form that allows the user to edit custom abduction-time code. Changes are submitted to the page editCustom2.jsp for further processing
editCustom2.jsp	182	This page takes the new custom abduction-time code passed on to it by editCustom1.jsp and replaces the old set of custom abduction-time code in the database with this new one.
setContext1.jsp	178	This page generates a drop box containing all the contexts defined in the system and allows the user to select a context when a source is being subscribed to a context. The selected context is passed on to setContext2.jsp for back-end processing.
setContext2.jsp	166	This page takes the source and selected context from setContext1.jsp and updates the database to reflect the source as subscribing to the context. Displays a success-message on success, or an error message on failure.
setElevation1.jsp	179	This page generates a drop box containing all the semantic types defined in the ontology and allows the user to select a semantic type when a column elevation is being specified. The selected semantic type is passed on to setElevation2.jsp for back-end processing.
setElevation2.jsp	165	This page takes the column and the selected semantic type from setElevation1.jsp and updates the database to reflect the column's elevation to the particular semantic type. Displays a success-message on success, or an error message on failure

Thus, the elevations editor comprises of 9 JavaServer Pages totaling 1,590 lines of code (over 10% of the entire code-base of the Graphical Metadata Management system).

7.6.2 Underlying Data Structures

We continue the object model we presented in an earlier section on "applications" and show the relevant relationships for the elevations-related entities in the system:

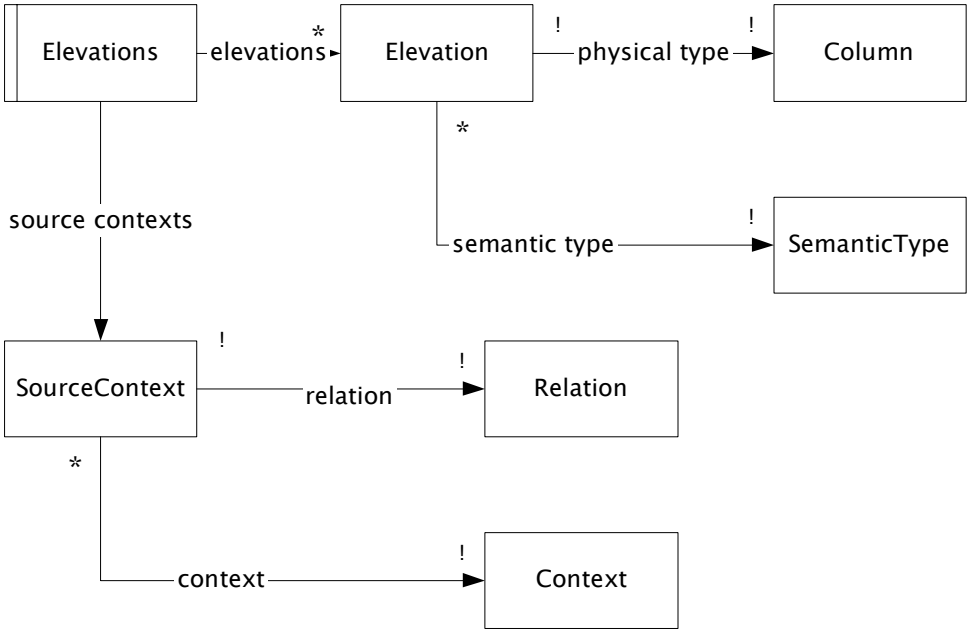

Figure 7.2: Object/data model for the elevations subsystem

7.6.2.1 Column Elevation Map

We store the elevation information for columns in the table elv_elevations. The physical_type column of this table refers to the column defined in sch_columns, while the semantic_type column of this table refers to the semantic type in ont_semtypes.

```
create table elv_elevations (
       physical_type   integer not null references sch_columns on delete
cascade,
       semantic_type   integer not null references ont_semtypes on delete
cascade
);
```

7.6.2.2 Context Subscription for Sources

The information about the context subscription of a source is stored in the elv_source_contexts table.

```
create table elv_source_contexts (
      relation    integer not null references sch_relations on delete
cascade,
      context     integer not null references cxt_contexts on delete cascade
);
```

7.7 Internal Representation Generators

The internal representation generators for elevations read data from the tables described above and generate both the COIN-based, and the GCMS-based internal representations of this data.

There are two internal representation generators provided for the elevations in the current system. These are summarized in the table below:

Filename	Lines	Functionality
GeneratorElevationsCOIN.jsp	263	Generates the internal representation of the elevation axioms required by the context interchange mediator[5]
GeneratorElevationsGCMS.jsp	276	Generates the internal representation of the elevation axioms required by the Global Context Mediation System, which is a successor to the context interchange mediator. In case the future developer would like to program a new generator, he or she should take a look at this file and the SQL queries it makes. Also, this file creates an internal representation file and places it in the value of the "applicationPath" parameter in the servlet context's environment (the web server) thus updating the abduction engine with the new file. For customized functionality on this front, the future developer should, again, look at this file.

Appendix A.5 contains an internal representation file for elevation axioms generated by GeneratorElevationsGCMS.jsp. For more details on the structure of the internal representation, the reader is referred to section 4.5 of [9].

8 Conversions

This section focuses on the design, implementation, and usage of the conversion functions editor, its underlying data model, and the generators which generate the internal representations from the database tables populated by the this editor.

8.1 Conversion Functions Page

The conversion functions management page allows the user to author, edit, and delete conversion functions for the application.

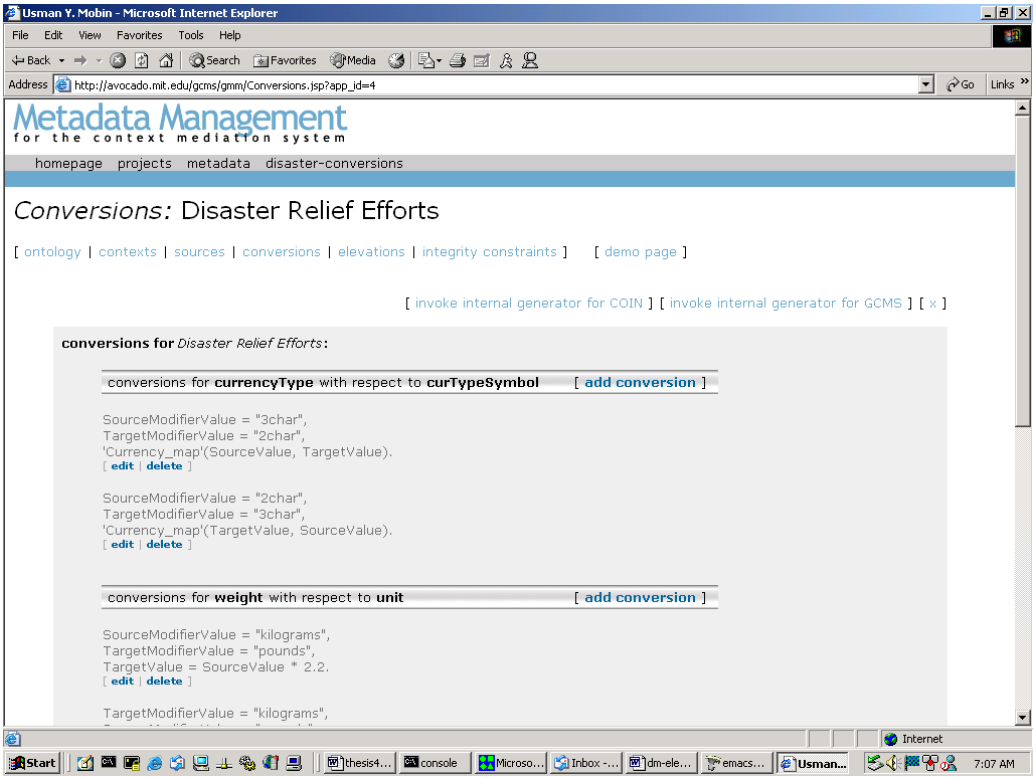

Figure 8.1: Conversion Functions Management Page

Figure 8.1 shows an example of a conversion functions main page. As is visible from the figure, this page allows us to add conversion functions, as well as edit or delete existing conversion functions.

8.1.1 Usage Guide

This section provides a quick introduction and user documentation for conversion function, and the conversion functions management page.

8.1.1.1 Our notion of conversions

Firstly, it is important to point out that our conversion functions are quite a bit different from our usual understanding of a function (mathematical construct that is applied to an input, returns an output). Our conversions define relations between variables and are used automatically whenever applicable. A conversion pertains to a semantic type with respect a modifier... so for example, a

conversion for companyFinancials with respect to scaleFactor will define a conversion scheme for a companyFinancials object across contexts based on the scaleFactor's value.

The normally understood notion of a conversion function in computer programming is something like:

Function(*input*)
1. Do something to *input* in finite number of steps
2. Return the result to the parent environment

On the other hand, our conversion functions aim to establish a relationship between a certain number of variables. Whenever, a sufficient number of these variables can be bound to actual values, the unbound variables can be deduced.

8.1.1.2 Predefined variables

The following variables are predefined and can be used directly in your conversion function. They are provided to you by the system:

Predefined Variable Name	Usage
Object	Refers to the semantic object being converted. The name of the context of the semantic object is encapsulated in the object itself, thus the conversion function does not have a predefined variable called SourceContext. See later for details.
TargetContext	Refers to the context to which we are converting to.
SourceModifierValue	This variable refers to the value of the modifier in the source context (recall that our conversion functions define conversions for semantic types with respect to modifiers)
TargetModifierValue	This variable refers to the value of the modifier in the target context.
SourceValue	This is the actual value undergoing conversion. This variable refers to this value in the source context
TargetValue	This is the actual value undergoing conversion. This variable refers to this value in the target context.

A conversion function will define a relationship between these variables. The usage of these variables will be explained below.

8.1.1.3 Simple Mathematical Conversions

Suppose, you wanted to write a conversion function for companyFinancials with respect to scaleFactor, and suppose you wanted to scale the values based on the magnitude of scaleFactor, then your conversion function will be:

```
TargetValue is SourceValue * (SourceModifierValue / TargetModifierValue).
```

Notice the use of our predefined reserved variables (see section 8.1.1.2). Also, you have the entire flexibility of prolog[13] at your hands, which means that you could have very well written the same conversion function as

```
X is SourceModifierValue / TargetModifierValue,
TargetValue is SourceValue * X.
```

Just as a refresher, variable names begin with uppercase letters while atoms begin with lowercase letters. More details about prolog conventions can be found in [13].

8.1.1.4 Database-backed Conversion Functions

Assume you wanted to write a conversion function that converted currency names from one format to another (say, a conversion for semantic type currencyType with respect to currencyTypeSymbol modifier. assume, modifier takes values "3char" or "2char"). Assume, we have a table, currency_map, in the database which has two columns and defines a mapping between the two currencyType symbols. Rows in this table would be like ("USD", "US"), ("PKR", "PK") etc. To define a conversion that is equivalent to "when the modifier value in source is 3char, and the modifier value in target is 2char then the relation between value in source and value in target is currency_map(value in source, value in target)." We define this conversion as

```
SourceModifierValue = "3char",
TargetModifierValue = "2char",
currency_map(SourceValue, TargetValue).
```

Notice how our prolog-style instructions logically represent our earlier conceptual understanding of the conversion function. Of course, if the name of the relation had begun with an uppercase character, we would have had to write the atom with single quotes as 'Currency_map'(SourceValue, TargetValue).

In order to assist the user in writing such conversion functions, the "add conversions" page lists not only this quick tutorial but also the list of all the available source relations in the system at that point in time. For example, my "add conversions" page for the financial example application lists the following relations available to me:

```
Currencytypes(country, currency)  in oracle
Currency_map(char3_currency, char2_currency)  in oracle
olsen(Exchanged, Expressed, Rate, Date)  in cameleon
datexform(Date1, Format1, Date2, Format2)  in dateXform
quotes(Cname, Last)  in cameleon
DiscAF(company_name, latest_annual_data, current_shares_outstanding,
net_income, net_sales, total_assets, country_of_incorp)  in oracle
WorldcAF(company_name, latest_annual_financial_date,
current_outstanding_shares, net_income, sales, total_assets,
country_of_incorp)  in oracle
WorldcAFT(company_name, latest_annual_financial_date,
current_outstanding_shares, net_income, sales, total_assets,
country_of_incorp)  in oracle
Name_map_Ds_Ws(ds_names, ws_names)  in oracle
Ticker_Lookup2(comp_name, ticker, exc)  in oracle
Name_map_Dt_Ds(dt_names, ds_names)  in oracle
Name_map_Dt_Ws(dt_names, ws_names)  in oracle
DStreamAF(as_of_date, name, total_sales, total_extraord_items_pre_tax,
earned_for_ordinary, currency)  in oracle
countryIncorp(company_name, country)  in view
CNames(company_name)  in view
moneyrates(bankname, rate, yield, minbalance)  in cameleon
name_map_msft_ds(msft_names, ds_names)  in oracle
```

8.1.1.5 Advanced Constructs

This section describes the use of attr/3 and value/3 in conversions, for advanced users only.

The Object variable mentioned earlier represents the data object being modified and encapsulates the source context of the object. To get the value of an attribute "whatever" of the object and assign it to a variable X, we will use a clause "attr(Object, whatever, X)". X will now contain the object represented by the "whatever" attribute of object Object (X also is a skolemized object like Object and encapsulates its source context). To get the value of a skolemized object in a different context, use the value/3 atom. For example, to get the value of X in context "reuters" and bind it to variable Y, we will use a clause "value(X, reuters, Y)". Of course, the direction of assignment in attr/3 and value/3 depends on which variable is unbound. Likewise, you can also use elevated data source relations to operate on skolemized objects. The reader is advised to refer to an existing application for illustration.

8.1.2 Source Files

The following table lists the source files used by the conversion functions editor and gives a short description of their functionality:

Filename	Lines	Functionality
Conversions.jsp	191	This is the main conversion functions management page. It displays all the modifiers described in the system (with their semantic types) and for each modifier displays the existing conversion functions defined. This is illustrated in figure 8.1. The file expects as input the variable “app_id” and displays the conversion functions information and editor for the particular application which has its app_id in the coin_apps table equal to the “app_id” passed on to this page as input
addConversion1.jsp	240	This page generates the form that allows the user to enter a new conversion function for a specific modifier to a semantic type. The results are submitted to addConversion2.jsp
addConversion2.jsp	194	Takes the newly defined conversion function from addConversion1.jsp and adds it as a conversion function for a specific semantic type with respect to a modifier.
delConversion1.jsp	159	Gets invoked when the user clicks on the “delete” link for a conversion function. Displays a confirmation dialog. If the user confirms, control is passed on to delConversion2.jsp with identification information regarding the conversion function to be deleted.
delConversion2.jsp	164	Takes the identification of a conversion function and deletes it from the database. Displays a success message on success or an error message on failure.
editConversion1.jsp	244	Displays a form that allows the user to edit an existing conversion function. The new function information is passed on to editConversion2.jsp for further processing.
editConversion2.jsp	192	Takes the new conversion function from editConversion2.jsp and replaces the old conversion function with the new one.

The conversions subsystem thus comprises of seven JavaServer Pages comprising of about 9% of the code-base of the Graphical Metadata Management system.

8.1.3 Underlying Data Structures

We continue the object model we presented in an earlier sections, and show the relevant relationships for the conversion-functions-related entities in the system:

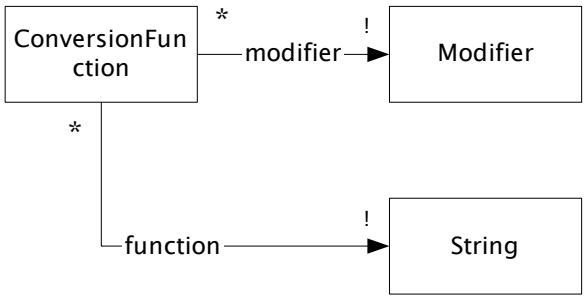

Figure 8.2: Object/data model for the conversions subsystem

Thus, in this simplistic yet flexible representation of a conversion function, we can store our conversion functions in the table below:

```
create sequence cnv_functions_id_seq start with 2;

create table cnv_functions (
       function_id          integer primary key,
       owner_id             integer references coin_apps on delete cascade,
       modifier             integer references ont_modifier_map on delete
cascade,
       function_body        varchar(4000) not null
);
```

8.2 Internal Representation Generators

The internal representation generators for conversion functions read data from the table described above and generate both the COIN-based, and the GCMS-based internal representations of this data.

There are two internal representation generators provided for conversions in the current system. These are summarized in the table below:

Filename	Lines	Functionality
GeneratorConversionsCOIN.jsp	263	Generates the internal representation of the conversion functions required by the context interchange mediator[5]
GeneratorConversionsGCMS.jsp	276	Generates the internal representation of the conversion functions required by the Global Context Mediation System, which is a successor

	to the context interchange mediator. In case the future developer would like to program a new generator, he or she should take a look at this file and the SQL queries it makes. Also, this file creates an internal representation file and places it in the value of the "applicationPath" parameter in the servlet context's environment (the web server) thus updating the abduction engine with the new file. For customized functionality on this front, the future developer should, again, look at this file.

Appendix A.4 contains an internal representation file for conversion functions generated by GeneratorElevationsGCMS.jsp. For more details, though not very insightful, on the structure of the internal representation, the reader is referred to section 4.8 of [9].

9 Conclusions and the Future

The Graphical Metadata Management system described in this thesis is running successfully on our demonstration website's developmental mirror. Also, the internal representation being used by the reasoning engine on our demonstration website[22] is the generated output from the internal representation generators of this system.

It is indeed obvious that the Graphical Metadata Management system makes the tasks for users operating in both the roles of information retrievers and information providers much easier and remarkably more automated than before. In general it appears that the Graphical Metadata Management system fulfils its aims and objectives to a considerable degree.

9.1 Need for more testing

Although some testing of the system has been performed, it has not been of a satisfactory rigor for a system of such size. The reason is that I have performed a sizeable part of the testing. For the person who develops the system, most things appear to be natural and intuitive, otherwise they wouldn't have been developed the way they were. It is important to point out that while users have tested parts of the system and played around with it, the task of developing an entire application from scratch still requires a certain amount of maturity with the underlying mediation system, and still remains to be an art. Some amount of documentation will need to be written in order to equip users with the usage-style necessary to develop entire applications using this system.

9.2 Application browsing

Although the Graphical Metadata Management system makes the task of managing metadata within applications much easier, it provides only a little amount of support for managing applications themselves. This has not been unintentional, as we have mentioned earlier in this thesis. An application-browser and manager is being developed at our research group and this tool aims to continue the task of applications management from where the Graphical Metadata Management system left it. The need for an application browser and the facilities it is expected to offer has been discussed earlier in this thesis.

9.3 Navigational improvements

Although consideration has been given to the overall navigational structure of the Graphical Metadata Management system, it is obvious that this is far from perfect. It is hoped that once the application browser and other related tools are ready, a more complete picture of the system with emerge, and thus we will be in a better position to make a navigational policy for the system that is both lasting and intuitive.

9.4 Query-building tool

After having a Graphical Metadata Management system, the next big step seems to be the query-building tool under development at our research lab. The tool would allow users to select an application (and thus is closely tied to the application-browser) and then run a query on that application.

[22] For our financial example

In much the same way as the Graphical Metadata Management system aims at the user operating in the role as an information provider, the query-builder would have its primary emphasis on the user operating in his or her role as an information retriever.

9.5 Graphical attribute rules

The Graphical Metadata Management system requires users to author attribute rules in textual form. Although this task is not a difficult one, it is felt that a more graphical way of dealing with these would greatly benefit users of the system. It might be pointed out that this task is much less trivial than it might seem at face value. In a resource-constrained (primarily time-constrained) environment, many other issues seem to take dominance over the issue of implementing graphical attribute rules. Nonetheless, this would be a very welcome future development.

9.6 Backend Improvement

There seem to be some need for improvement in the abduction engine and the optimizer of the execution engine. The abduction engine, which was adopted from the older context interchange mediator[5], has certain problems. Firstly, it has problems with integrity constraints, and the integrity constraints for the financial example seem to be hard-coded into it. There needs to be a mechanism for allowing the abduction engine to read integrity constraints from a file just like all other metadata. Secondly, the abduction engine does not have support for an ontology that has semantic types exhibiting a hierarchical structure. This needs to be implemented. Finally, the executioner needs to address some infinite recursion possibilities in its optimization code.

9.7 Registry development

The Graphical Metadata Management system introduces a new data model for the storage of all the metadata required by the context mediation system. It introduces the use of a relational database system for such storage, and provides the new "glue" for putting all the components of the context mediation system together. In future it appears to be very important that all parts of the system read their metadata directly from the database tables of the Graphical Metadata Management system. Once this advancement is made, we will be able to phase out the cryptic internal representations completely. This would truly be a giant leap for the entire context mediation system.

9.8 Portable data

Another important future development would be the introduction of generators that export data in a portable format such as the Resource Description Framework. A step further than this would be the ability to import such data too. These were in the initial goals of this project but had to be later dropped in favor of other augmentations due to time and human-energy constraints. The ability to import RDF would translate into the ability to use tools like VisioDAML[33] indirectly to author our ontologies.

References

[1] Madnick, S. E. (1999). *Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity*, 1999 IEEE Meta-Data Conference, April 6-7, 1999.

[2] Bressan, S., Fynn, K., Goh, C. H., Jakobisiak, M., Hussein, K., Kon, H., Lee, T., Madnick, S. E., Pena, T., Qu, J., Shum, A., and Siegel, M. (1997). *The COntext INterchange Mediator Prototype*, ACM SIGMOD International Conference on Management of Data, 1997

[3] *Resource Description Framework (RDF)*, http://www.w3.org/RDF/

[4] Siegel, M., Madnick, S. E. (1991). *A Metadata Approach to Solving Semantic Conflicts*. In proceedings of the 17th international conference on very large databases, pages 133-145. 1991.

[5] Daruwala, Goh, Hofmeister, Husein, Madnick, Siegel. (1995) *The Context Interchange Network Prototype*, Centre for Information Systems Laboratory, working paper 95-01, Sloan School of Management, Massachusetts Institute of Technology, February 1995.

[6] Goh, Madnick, Siegel. (1994). *Context Interchange: Overcoming the Challenges of Large-Scale Interoperable Database Systems in a Dynamic Environment*. Proceeding of International Conference on Information and Knowledge Management, 1994.

[7] Ken Arnold, James Gosling, David Holmes. (2000). *The Java™ Programming Language, Third Edition*. Addison-Wesley Pub Co., April 2000.

[8] Tim Lindholm, Frank Yellin. (1999). *The Java Virtual Machine Specification*, Addison-Wesley Pub Co., May 1999.

[9] Syed Ahmed Zaman Shah. (1998). *Design and Architecture of the Context Interchange System*. Centre for Information Systems Laboratory, working paper 98-05, Sloan School of Management, Massachusetts Institute of Technology, May 1998.

[10] G. Koch, K. Loney, Oracle8: The Complete Reference, *Oracle Press, 1997*

[11] James R. Groff, Paul N. Weinberg, SQL: The Complete Reference, *McGraw-Hill Higher Education, 1999*

[12] W. Kent, A Simple Guide to Five Normal Forms in Relational Database Theory, *Communications of the ACM, February 1983 Volume 26*.

[13] Leon Sterling. (1994). *The Art of Prolog: Advanced Programming Techniques*. MIT Press, Cambridge, Massachusetts. 1994.

[14] Guy Zimmer, S. David Kwak, Frank Manola. (2001). *The Semantic Web Problem – Disaster Relief Context*. Mitre Corporation working paper, September 2001.

[15] Firat, A., Madnick, S., Siegel, M. (2000). *The Cameleon Approach to the Interoperability of Web Sources and Traditional Relational DataBases*. Proceedings of the 10th Annual Workshop On Information Technologies and Systems, Brisbane, Queensland, Australia, 2000.

[16] Firat, A., Madnick, S., Siegel, M.. (2000). *The Cameleon Web Wrapper Engine*. Proceedings of the VLDB Workshop on Technologies for E-Services, Cairo, Egypt 2000.

[17] The DARPA Agent Markup Language Specification, http://www.daml.org/2001/03/daml+oil.daml

[18] Stanford University Project Protégé, http://protege.stanford.edu/index.shtml

[19] GraMToR Resource Description Framework schema editor, http://nestroy.wi-inf.uni-essen.de/xwmf/

[20] Hans Bergsten. (2000). *JavaServer Pages*. O'Reilly & Associates, first edition December 2000

[21] W. Kent, A Simple Guide to Five Normal Forms in Relational Database Theory, *Communications of the ACM, February 1983 Volume 26*.

[22] E. F. Codd, Normalized Database Structure: A Brief Tutorial, *ACM SIGFIDET workshop on data description, access and control, November 1971*

[23] E. F. Codd, Further Normalization of Database Relational Model, *International Business Machines Corporation, Research Report RJ909, 1972*

[24] P. Greenspun, Philip and Alex's Guide to Web Publishing. *Morgan Kaufmann Publishers, April 1999; also available online at http://www.arsdigita.com/books/panda/*

[25] C. Date, Introduction to Database Systems, sixth edition, *New York Addison-Wesley, 1995*.

[26] C. C. Fleming, B. Vonhale, Handbook of Relational Database Design, *Addison-Wesley Pub Co, 1988*

[27] G. Koch, K. Loney, Oracle8: The Complete Reference, *Oracle Press, 1997*

[28] Cheng Hian Goh, (1996). Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, Ph.D. Thesis, MIT Sloan School of Management, 1996.

[29] Kofi Fynn, (1997). A Planner/Optimizer/Executioner for Context Mediated Queries. MIT Master Thesis, Electrical Engineering and Computer Science.

[30] Guttag, Liskov, (2000). Program Development in Java: Abstraction, Specification, and Object-Oriented Design. 1st edition, Addison-Wesley Pub Co. 2000

[31] Groff, Weinberg, (1999). SQL: The Complete Reference. McGraw-Hill Professional Publishing, March 1999.

[32] Stéphane Bressan, Kofi Fynn, Cheng Hian Goh, Stuart E. Madnick, Tito Pena, and Michael Siegel, (1997). Overview of the Prolog Implementation of the COntext INterchange Prototype. Fifth International Conference on Practical Applications of Prolog

[33] VisioDAML: A Visio application to illustrate how Visio can be used to create graphical representations of DAML+OIL ontologies, http://www.daml.org/ visiodaml

[34] Microsoft Visio Drawing Viewer, http://office.microsoft.com/downloads/ 2002/VWC10.aspx

[35] Steven Holzner, (2000). Inside XML. New Riders Publishing, 1st edition November 2000.

[36] Netbryx Technologies, EditML version 2.5. http://www.editml.com/

[37] The Reggie Metadata Editor. http://metadata.net/dstc/

[38] Gray, Preece, Fiddian, Gray, Bench-Capon, Shave, Azarmi, Wiegand, Ashwell, Beer, Cui, Diaz, Embury, Hui, Jones, Jones, Kemp, Lawson, Lunn, Marti, Shao, Visser, (1997). KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases. *Eighth International Workshop on Database and Expert Systems Applications, DEXA'97*

[39] Knowledge Reuse And Fusion/Transformation Ontology Browser. University of Liverpool. http://www.csc.liv.ac.uk/~pepijn/OB/

[40] Java Ontology Editor (JOE) of the University of South Carolina. http:// www.cse.sc.edu/research/cit/demos/java/joe/

[41] Popkin Software's Envision XML, http://www.popkin.com/products/envision/ dtd/dtd.htm

[42] The Java Plug-In. http://java.sun.com/getjava/

[43] Cheng Hian Goh, Stéphane Bressan, Stuart Madnick and Michael Siegel, (1999), Context interchange: new features and formalisms for the intelligent integration of information. ACM TOIS 17(3): 270-293 (1999).

[44] Bresson, Stephane, C. Goh, N. Levina, A. Shah, and M. Siegel, "Context Knowledge Representation and Reasoning in the Context Interchange System," The International Journal of Artificial Intelligence, Neutral Networks, and Complex Problem-Solving Technologies, Volume 12, Number 2, September 2000, pp. 165-180, [SWP #4133, CISL #00-04].

Appendix A – Internal Representation of Financial Example's Metadata

This appendix presents the internal representation of the metadata for the TASC Financial Example mentioned at various places in this thesis. The representation presented here is generated by the GCMS generators of the Graphical Metadata Management system.

A.1 Ontology

```
%% domain model for TASC Financial Example
%% generation timestamp: Sun Feb 03 05:46:31 EST 2002

%% by module: ontology_gcms_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

%%
%% Semantic types
%%
rule(semanticType(basic), (true)).
rule(semanticType(companyFinancials), (true)).
rule(semanticType(companyName), (true)).
rule(semanticType(countryName), (true)).
rule(semanticType(currencyType), (true)).
rule(semanticType(date), (true)).
rule(semanticType(exchangeRate), (true)).

%%
%% Modifiers
%%
rule(modifiers(basic, []), (true)).
rule(modifiers(companyFinancials, [currency, scaleFactor]), (true)).
rule(modifiers(companyName, [format]), (true)).
rule(modifiers(countryName, []), (true)).
rule(modifiers(currencyType, [curTypeSym]), (true)).
rule(modifiers(date, [dateFmt]), (true)).
rule(modifiers(exchangeRate, []), (true)).

%%
%% Attributes
%%
rule(attributes(basic, []), (true)).
rule(attributes(companyFinancials, [company, fyEnding]), (true)).
rule(attributes(companyName, [countryIncorp]), (true)).
rule(attributes(countryName, [officialCurrency]), (true)).
rule(attributes(currencyType, [curTypeSym]), (true)).
rule(attributes(date, []), (true)).
rule(attributes(exchangeRate, [fromCur, toCur, txnDate]), (true)).

%%
%% Contexts
%%
rule(context(datastream), (true)).
rule(context(disclosure), (true)).
rule(context(microsoft), (true)).
rule(context(olsen), (true)).
rule(context(worldscope), (true)).
rule(context(yahoo), (true)).
```

A.2 Contexts

```
%% context definitions for TASC Financial Example
%% generation timestamp: Sun Feb 03 05:47:26 EST 2002

%% by module: contexts_gcms_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

%%
%% datastream context
%%
rule(modifier(companyFinancials, Object, scaleFactor, datastream, Modifier),
          (cste(basic, Modifier, datastream, 1000))).

rule(modifier(companyFinancials, Object, currency, datastream, Modifier),
          (attr(Object, company, IntermediateVariable1),
           attr(IntermediateVariable1, countryIncorp, IntermediateVariable2),
           attr(IntermediateVariable2, officialCurrency, Modifier))).

rule(modifier(companyName, Object, format, datastream, Modifier),
          (cste(basic, Modifier, datastream, "dt_name"))).

rule(modifier(date, Object, dateFmt, datastream, Modifier),
          (cste(basic, Modifier, datastream, "European Style -"))).

rule(modifier(currencyType, Object, curTypeSym, datastream, Modifier),
          (cste(basic, Modifier, datastream, "3char"))).

%%
%% disclosure context
%%
rule(modifier(companyFinancials, Object, scaleFactor, disclosure, Modifier),
          (cste(basic, Modifier, disclosure, 1))).

rule(modifier(companyFinancials, Object, currency, disclosure, Modifier),
          (attr(Object, company, IntermediateVariable1),
           attr(IntermediateVariable1, countryIncorp, IntermediateVariable2),
           attr(IntermediateVariable2, officialCurrency, Modifier))).

rule(modifier(companyName, Object, format, disclosure, Modifier),
          (cste(basic, Modifier, disclosure, "ds_name"))).

rule(modifier(date, Object, dateFmt, disclosure, Modifier),
          (cste(basic, Modifier, disclosure, "American Style /"))).

rule(modifier(currencyType, Object, curTypeSym, disclosure, Modifier),
          (cste(basic, Modifier, disclosure, "3char"))).

%%
%% microsoft context
%%
rule(modifier(companyFinancials, Object, scaleFactor, microsoft, Modifier),
          (cste(basic, Modifier, microsoft, 100))).

rule(modifier(companyFinancials, Object, currency, microsoft, Modifier),
          (cste(currencyType, Modifier, microsoft, "USD"))).

rule(modifier(companyName, Object, format, microsoft, Modifier),
          (cste(basic, Modifier, microsoft, "msft_standard"))).

rule(modifier(date, Object, dateFmt, microsoft, Modifier),
          (cste(basic, Modifier, microsoft, "American Style /"))).
```

```
rule(modifier(currencyType, Object, curTypeSym, microsoft, Modifier),
          (cste(basic, Modifier, microsoft, "3char"))).

%%
%% olsen context
%%
rule(modifier(companyFinancials, Object, scaleFactor, olsen, Modifier),
          (cste(basic, Modifier, olsen, 1))).

rule(modifier(companyFinancials, Object, currency, olsen, Modifier),
          (attr(Object, company, IntermediateVariable1),
           attr(IntermediateVariable1, countryIncorp, IntermediateVariable2),
           attr(IntermediateVariable2, officialCurrency, Modifier))).

rule(modifier(companyName, Object, format, olsen, Modifier),
          (cste(basic, Modifier, olsen, "ws_name"))).

rule(modifier(date, Object, dateFmt, olsen, Modifier),
          (cste(basic, Modifier, olsen, "European Style /"))).

rule(modifier(currencyType, Object, curTypeSym, olsen, Modifier),
          (cste(basic, Modifier, olsen, "3char"))).

%%
%% worldscope context
%%
rule(modifier(companyFinancials, Object, scaleFactor, worldscope, Modifier),
          (cste(basic, Modifier, worldscope, 1000))).

rule(modifier(companyFinancials, Object, currency, worldscope, Modifier),
          (cste(currencyType, Modifier, worldscope, "USD"))).

rule(modifier(companyName, Object, format, worldscope, Modifier),
          (cste(basic, Modifier, worldscope, "ws_name"))).

rule(modifier(date, Object, dateFmt, worldscope, Modifier),
          (cste(basic, Modifier, worldscope, "American Style /"))).

rule(modifier(currencyType, Object, curTypeSym, worldscope, Modifier),
          (cste(basic, Modifier, worldscope, "3char"))).

%%
%% yahoo context
%%
rule(modifier(companyFinancials, Object, scaleFactor, yahoo, Modifier),
          (cste(basic, Modifier, yahoo, 1))).

rule(modifier(companyFinancials, Object, currency, yahoo, Modifier),
          (cste(currencyType, Modifier, yahoo, "USD"))).

rule(modifier(companyName, Object, format, yahoo, Modifier),
          (cste(basic, Modifier, yahoo, "ya_name"))).

rule(modifier(date, Object, dateFmt, yahoo, Modifier),
          (cste(basic, Modifier, yahoo, "American Style /"))).

rule(modifier(currencyType, Object, curTypeSym, yahoo, Modifier),
          (cste(basic, Modifier, yahoo, "3char"))).
```

A.3 Source Descriptions

```
%% source descriptions for TASC Financial Example
%% generation timestamp: Sun Feb 03 05:58:12 EST 2002
```

```
%% by module: sources_coin_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

%%
%% Source Information
%%
rule(source(cameleon,
       web,
       'http://context.mit.edu/~coin/demos/wrapper/cgi-bin/prolog-
wrapper.cgi'), (true)).

rule(source(datexform,
       web,
       'http://context.mit.edu/cgi-bin/qbe-dev/client/datexform.cgi'),
(true)).

rule(source(oracle,
       db,
       'avocado.mit.edu:gmm@coin/madnickrules'), (true)).

rule(source(view,
       db,
       'ignore'), (true)).

%%
%% Relations
%%
rule(relation(cameleon,
         moneyrates,
         ie,
         [[bankname, string],
          [rate, real],
          [yield, real],
          [minbalance, real]],
          cap([[0, 0, 0, 0]],
              ['<', '>', '<>', '=<', '=>'])), (true)).

rule(relation(cameleon,
         olsen,
         ie,
         [['Exchanged', string],
          ['Expressed', string],
          ['Rate', real],
          ['Date', string]],
          cap([[1, 1, 0, 1]],
              ['<', '>', '<>', '=<', '=>'])), (true)).

rule(relation(cameleon,
         quotes,
         ie,
         [['Cname', string],
          ['Last', string]],
          cap([[1, 0]],
              ['<', '>', '<>', '=<', '=>'])), (true)).

rule(relation(datexform,
         datexform,
         i,
         [['Date1', string],
          ['Format1', string],
          ['Date2', string],
          ['Format2', string]],
          cap([[1, 1, 0, 1], [0, 1, 1, 1]],
```

```
            ['<', '>', '<>', '=<', '=>'])), (true)).

rule(relation(oracle,
         'Currency_map',
         i,
         [[char3_currency, string],
          [char2_currency, string]],
          cap([[0, 0]],
              [])), (true)).

rule(relation(oracle,
         'Currencytypes',
         i,
         [[country, string],
          [currency, string]],
          cap([[0, 0]],
              [])), (true)).

rule(relation(oracle,
         'DiscAF',
         ie,
         [[company_name, string],
          [latest_annual_data, string],
          [current_shares_outstanding, integer],
          [net_income, integer],
          [net_sales, integer],
          [total_assets, integer],
          [country_of_incorp, string]],
          cap([[0, 0, 0, 0, 0, 0, 0]],
              [])), (true)).

rule(relation(oracle,
         'DStreamAF',
         ie,
         [[as_of_date, string],
          [name, string],
          [total_sales, integer],
          [total_extraord_items_pre_tax, integer],
          [earned_for_ordinary, integer],
          [currency, string]],
          cap([[0, 0, 0, 0, 0, 0]],
              [])), (true)).

rule(relation(oracle,
         'Name_map_Ds_Ws',
         i,
         [[ds_names, string],
          [ws_names, string]],
          cap([[0, 0]],
              [])), (true)).

rule(relation(oracle,
         'Name_map_Dt_Ds',
         i,
         [[dt_names, string],
          [ds_names, string]],
          cap([[0, 0]],
              [])), (true)).

rule(relation(oracle,
         'Name_map_Dt_Ws',
         i,
         [[dt_names, string],
          [ws_names, string]],
          cap([[0, 0]],
              [])), (true)).
```

```
rule(relation(oracle,
         name_map_msft_ds,
         i,
         [[msft_names, string],
          [ds_names, string]],
          cap([[0, 0]],
              [])), (true)).

rule(relation(oracle,
         'Ticker_Lookup2',
         i,
         [[comp_name, string],
          [ticker, string],
          [exc, string]],
          cap([[0, 0, 0]],
              [])), (true)).

rule(relation(oracle,
         'WorldcAF',
         ie,
         [[company_name, string],
          [latest_annual_financial_date, string],
          [current_outstanding_shares, integer],
          [net_income, integer],
          [sales, integer],
          [total_assets, integer],
          [country_of_incorp, string]],
          cap([[0, 0, 0, 0, 0, 0, 0]],
              [])), (true)).

rule(relation(oracle,
         'WorldcAFT',
         ie,
         [[company_name, string],
          [latest_annual_financial_date, string],
          [current_outstanding_shares, integer],
          [net_income, integer],
          [sales, integer],
          [total_assets, integer],
          [country_of_incorp, string]],
          cap([[0, 0, 0, 0, 0, 0, 0]],
              [])), (true)).

rule(relation(view,
         'CNames',
         e,
         [[company_name, string]],
          cap([[0]],
              [])), (true)).

rule(relation(view,
         countryIncorp,
         e,
         [[company_name, string],
          [country, string]],
          cap([[0, 0]],
              [])), (true)).
```

A.4 Conversion Functions

```
%% conversion functions for TASC Financial Example
%% generation timestamp: Sun Feb 03 05:52:23 EST 2002
```

```
%% by module: conversions_gcms_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

%%
%% conversion functions for companyFinancials
%% with respect to scaleFactor
%%
rule(cvt(companyFinancials, Object, scaleFactor, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (Ratio is SourceModifierValue / TargetModifierValue,
           TargetValue is SourceValue * Ratio)).
%%
%% conversion functions for companyFinancials
%% with respect to currency
%%
rule(cvt(companyFinancials, Object, currency, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (attr(Object, fyEnding, FyDate),
           value(FyDate, TargetContext, DateValue),
           olsen_p(Fc, Tc, Rate, TxnDate),
           value(Fc, TargetContext, SourceModifierValue),
           value(Tc, TargetContext, TargetModifierValue),
           value(TxnDate, TargetContext, DateValue),
           value(Rate, TargetContext, Rv),
           TargetValue is SourceValue * Rv)).
%%
%% conversion functions for companyName
%% with respect to format
%%
rule(cvt(companyName, Object, format, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (name_map(SourceValue,   SourceModifierValue,   TargetModifierValue,
      TargetValue))).
%%
%% conversion functions for date
%% with respect to dateFmt
%%
rule(cvt(date, Object, dateFmt, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (datexform(SourceValue,      SourceModifierValue,      TargetValue,
      TargetModifierValue))).
%%
%% conversion functions for currencyType
%% with respect to curTypeSym
%%
rule(cvt(currencyType, Object, curTypeSym, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (SourceModifierValue = "3char",
           TargetModifierValue = "2char",
           'Currency_map'(SourceValue, TargetValue))).
rule(cvt(currencyType, Object, curTypeSym, TargetContext,
    SourceModifierValue, SourceValue, TargetModifierValue, TargetValue),
          (SourceModifierValue = "2char",
           TargetModifierValue = "3char",
           'Currency_map'(TargetValue, SourceValue))).
rule(currentDate(Date),
      ({date(D),
        substring(D, 5, 3, Month),
        substring(D, 9, 2, Day),
        substring(D, 23, 2, Year)},
         month(Month, NumMonth),
       {concat_string([NumMonth, /, Day, /, Year],Date)})).
```

```
rule(month("Jan", 01), (true)).
rule(month("Feb", 02), (true)).
rule(month("Mar", 03), (true)).
rule(month("Apr", 04), (true)).
rule(month("May", 05), (true)).
rule(month("Jun", 06), (true)).
rule(month("Jul", 07), (true)).
rule(month("Aug", 08), (true)).
rule(month("Sep", 09), (true)).
rule(month("Oct", 10), (true)).
rule(month("Nov", 11), (true)).
rule(month("Dec", 12), (true)).

rule(name_map(Val, "ds_name", "ws_name", V), ('Name_map_Ds_Ws'(Val, V))).
rule(name_map(Val, "ws_name", "ds_name", V), ('Name_map_Ds_Ws'(V, Val))).
rule(name_map(Val, "dt_name", "ds_name", V), ('Name_map_Dt_Ds'(Val, V))).
rule(name_map(Val, "ds_name", "dt_name", V), ('Name_map_Dt_Ds'(V, Val))).
rule(name_map(Val, "dt_name", "ws_name", V), ('Name_map_Dt_Ws'(Val, V))).
rule(name_map(Val, "ws_name", "dt_name", V), ('Name_map_Dt_Ws'(V, Val))).
rule(name_map(Val,  "ws_name",  "ya_name",  V),  ('Ticker_Lookup2'(Val,  V,
      _8071))).
rule(name_map(Val,  "ya_name",  "ws_name",  V),  ('Ticker_Lookup2'(V,  Val,
      _8115))).
rule(name_map(Val,  "dt_name",  "ya_name",  V),  ('Name_map_Dt_Ws'(Val,  Z),
      'Ticker_Lookup2'(Z, V, _8162))).
rule(name_map(Val,  "ya_name",  "dt_name",  V),  ('Ticker_Lookup2'(Z,  Val,
      _8218), 'Name_map_Dt_Ws'(V, Z))).
rule(name_map(Val,  "ds_name",  "ya_name",  V),  ('Name_map_Ds_Ws'(Val,  Z),
      'Ticker_Lookup2'(Z, V, _8264))).
rule(name_map(Val,  "ya_name",  "ds_name",  V),  ('Ticker_Lookup2'(Z,  Val,
      _8320), 'Name_map_Ds_Ws'(V, Z))).

rule(name_map(Val,  "msft_standard",  "ds_name",  V),  (name_map_msft_ds(Val,
      V))).

rule(name_map(Val,  "ds_name",  "msft_standard",  V),  (name_map_msft_ds(V,
      Val))).

rule(currentDate_p(
    skolem(date, V, worldscope, 1,
        currentDate(V))),
(currentDate(V))).

rule('CNames'(Name),
('DiscAF'(Name, __, _8395, _8396, _8397, _8398, _8399))).

rule('CNames'(Name),
('WorldcAF'(Name, _8429, _8430, _8431, _8432, _8433, _8434))).

rule('CNames'(Name),
('DStreamAF'(_8462, Name, _8464, _8465, _8466, _8467))).
```

A.5 Elevation Rules

```
%% elevation axioms for TASC Financial Example
%% generation timestamp: Sun Feb 03 05:59:25 EST 2002

%% by module: elevations_gcms_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

rule(
```

```
  olsen_p(
     skolem(currencyType, C1, olsen, 1,
          olsen(C1, C2, C3, C4)),
     skolem(currencyType, C2, olsen, 2,
          olsen(C1, C2, C3, C4)),
     skolem(exchangeRate, C3, olsen, 3,
          olsen(C1, C2, C3, C4)),
     skolem(date, C4, olsen, 4,
          olsen(C1, C2, C3, C4))),
   (olsen(C1, C2, C3, C4))).

rule(
  quotes_p(
     skolem(companyName, C1, yahoo, 1,
          quotes(C1, C2)),
     skolem(basic, C2, yahoo, 2,
          quotes(C1, C2))),
   (quotes(C1, C2))).

rule(
  'Currency_map_p'(
     skolem(basic, C1, disclosure, 1,
          'Currency_map'(C1, C2)),
     skolem(basic, C2, disclosure, 2,
          'Currency_map'(C1, C2))),
   ('Currency_map'(C1, C2))).

rule(
  'Currencytypes_p'(
     skolem(countryName, C1, disclosure, 1,
          'Currencytypes'(C1, C2)),
     skolem(currencyType, C2, disclosure, 2,
          'Currencytypes'(C1, C2))),
   ('Currencytypes'(C1, C2))).

rule(
  'DiscAF_p'(
     skolem(companyName, C1, disclosure, 1,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(null, C2, disclosure, 2,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C3, disclosure, 3,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C4, disclosure, 4,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C5, disclosure, 5,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C6, disclosure, 6,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyName, C7, disclosure, 7,
          'DiscAF'(C1, C2, C3, C4, C5, C6, C7))),
   ('DiscAF'(C1, C2, C3, C4, C5, C6, C7))).

rule(
  'DStreamAF_p'(
     skolem(date, C1, datastream, 1,
          'DStreamAF'(C1, C2, C3, C4, C5, C6)),
     skolem(companyName, C2, datastream, 2,
          'DStreamAF'(C1, C2, C3, C4, C5, C6)),
     skolem(companyFinancials, C3, datastream, 3,
          'DStreamAF'(C1, C2, C3, C4, C5, C6)),
     skolem(companyFinancials, C4, datastream, 4,
          'DStreamAF'(C1, C2, C3, C4, C5, C6)),
     skolem(companyFinancials, C5, datastream, 5,
          'DStreamAF'(C1, C2, C3, C4, C5, C6)),
     skolem(currencyType, C6, datastream, 6,
```

```
          'DStreamAF'(C1, C2, C3, C4, C5, C6))),
   ('DStreamAF'(C1, C2, C3, C4, C5, C6))).

rule(
  'Name_map_Ds_Ws_p'(
     skolem(basic, C1, disclosure, 1,
          'Name_map_Ds_Ws'(C1, C2)),
     skolem(basic, C2, disclosure, 2,
          'Name_map_Ds_Ws'(C1, C2))),
   ('Name_map_Ds_Ws'(C1, C2))).

rule(
  'Name_map_Dt_Ds_p'(
     skolem(basic, C1, datastream, 1,
          'Name_map_Dt_Ds'(C1, C2)),
     skolem(basic, C2, datastream, 2,
          'Name_map_Dt_Ds'(C1, C2))),
   ('Name_map_Dt_Ds'(C1, C2))).

rule(
  'Name_map_Dt_Ws_p'(
     skolem(basic, C1, datastream, 1,
          'Name_map_Dt_Ws'(C1, C2)),
     skolem(basic, C2, datastream, 2,
          'Name_map_Dt_Ws'(C1, C2))),
   ('Name_map_Dt_Ws'(C1, C2))).

rule(
  name_map_msft_ds_p(
     skolem(basic, C1, microsoft, 1,
          name_map_msft_ds(C1, C2)),
     skolem(basic, C2, microsoft, 2,
          name_map_msft_ds(C1, C2))),
   (name_map_msft_ds(C1, C2))).

rule(
  'Ticker_Lookup2_p'(
     skolem(basic, C1, worldscope, 1,
          'Ticker_Lookup2'(C1, C2, C3)),
     skolem(basic, C2, worldscope, 2,
          'Ticker_Lookup2'(C1, C2, C3)),
     skolem(basic, C3, worldscope, 3,
          'Ticker_Lookup2'(C1, C2, C3))),
   ('Ticker_Lookup2'(C1, C2, C3))).

rule(
  'WorldcAF_p'(
     skolem(companyName, C1, worldscope, 1,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(date, C2, worldscope, 2,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C3, worldscope, 3,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C4, worldscope, 4,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C5, worldscope, 5,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C6, worldscope, 6,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(countryName, C7, worldscope, 7,
          'WorldcAF'(C1, C2, C3, C4, C5, C6, C7))),
   ('WorldcAF'(C1, C2, C3, C4, C5, C6, C7))).

rule(
  'WorldcAFT_p'(
     skolem(companyName, C1, worldscope, 1,
```

```
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(date, C2, worldscope, 2,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(basic, C3, worldscope, 3,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C4, worldscope, 4,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C5, worldscope, 5,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(companyFinancials, C6, worldscope, 6,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7)),
     skolem(countryName, C7, worldscope, 7,
          'WorldcAFT'(C1, C2, C3, C4, C5, C6, C7))),
   ('WorldcAFT'(C1, C2, C3, C4, C5, C6, C7))).

rule(
  'CNames_p'(
    skolem(companyName, C1, disclosure, 1,
         'CNames'(C1))),
   ('CNames'(C1))).

rule(attr(X, countryIncorp, Y),
     ('DiscAF_p'(X, _4981, _4982, _4983, _4984, _4985, Y))).

rule(attr(X, officialCurrency, Y),
     ('Currencytypes_p'(Z, Y),
      'DiscAF_p'(_5041, _5042, _5043, _5044, _5045, _5046, X),
      value(X, disclosure, U),
      value(Z, disclosure, U))).

rule(attr(X, company, Y),
     ('DiscAF_p'(Y, _5085, _5086, X, _5088, _5089, _5090))).

rule(attr(X, fyEnding, Y),
     ('DiscAF_p'(_5125, Y, _5127, X, _5129, _5130, _5131))).

rule(attr(X, company, Y),
     ('DiscAF_p'(Y, _5167, _5168, _5169, _5170, X, _5172))).

rule(attr(X, fyEnding, Y),
     ('DiscAF_p'(_5207, Y, _5209, _5210, _5211, X, _5213))).

rule(attr(X, countryIncorp, Y),
     ('WorldcAF_p'(X, _5249, _5250, _5251, _5252, _5253, Y))).

rule(attr(X, company, Y),
     ('WorldcAF_p'(Y, _5290, _5291, X, _5293, _5294, _5295))).

rule(attr(X, fyEnding, Y),
     ('WorldcAF_p'(_5330, Y, _5332, X, _5334, _5335, _5336))).

rule(attr(X, company, Y),
     ('WorldcAF_p'(Y, _5372, _5373, _5374, X, _5376, _5377))).

rule(attr(X, fyEnding, Y),
     ('WorldcAF_p'(_5412, Y, _5414, _5415, X, _5417, _5418))).

rule(attr(X, company, Y),
     ('WorldcAF_p'(Y, _5454, _5455, _5456, _5457, X, _5459))).

rule(attr(X, fyEnding, Y),
     ('WorldcAF_p'(_5494, Y, _5496, _5497, _5498, X, _5500))).

rule(attr(X, officialCurrency, Y),
     ('Currencytypes_p'(Z, Y),
      'WorldcAF_p'(_5555, _5556, _5557, _5558, _5559, _5560, X),
```

```
      value(X, worldscope, U),
      value(Z, worldscope, U))).

rule(attr(X, countryIncorp, Y),
     ('WorldcAFT_p'(X, _5599, _5600, _5601, _5602, _5603, Y))).

rule(attr(X, company, Y),
     ('WorldcAFT_p'(Y, _5640, _5641, X, _5643, _5644, _5645))).

rule(attr(X, fyEnding, Y),
     ('WorldcAFT_p'(_5686, _5687, _5688, X, _5690, _5691, _5692),
      currentDate_p(Y))).

rule(attr(X, company, Y),
     ('WorldcAFT_p'(Y, _5727, _5728, _5729, X, _5731, _5732))).

rule(attr(X, fyEnding, Y),
     ('WorldcAFT_p'(_5773, _5774, _5775, _5776, X, _5778, _5779),
      currentDate_p(Y))).

rule(
      attr(X, company, Y),
     ('WorldcAFT_p'(Y, _5814, _5815, _5816, _5817, X, _5819))).

rule(
      attr(X, fyEnding, Y),
     ('WorldcAFT_p'(_5860, _5861, _5862, _5863, _5864, X, _5866),
      currentDate_p(Y))).

rule(
      attr(X, officialCurrency, Y),
     ('Currencytypes_p'(Z, Y),
      'WorldcAFT_p'(_5920, _5921, _5922, _5923, _5924, _5925, X),
      value(X, worldscope, U),
      value(Z, worldscope, U))).

rule(attr(X, company, Y),
     ('DStreamAF_p'(_5963, Y, X, _5966, _5967, _5968))).

rule(attr(X, fyEnding, Y),
     ('DStreamAF_p'(Y, _6004, X, _6006, _6007, _6008))).

rule(attr(X, company, Y),
     ('DStreamAF_p'(_6043, Y, _6045, X, _6047, _6048))).

rule(attr(X, fyEnding, Y),
     ('DStreamAF_p'(Y, _6084, _6085, X, _6087, _6088))).

rule(attr(X, company, Y),
     ('DStreamAF_p'(_6123, Y, _6125, _6126, X, _6128))).

rule(attr(X, fyEnding, Y),
     ('DStreamAF_p'(Y, _6164, _6165, _6166, X, _6168))).

rule(attr(X, countryIncorp, Y),
     ('DStreamAF_p'(_6212, X, _6214, _6215, _6216, Z),
      attr(Y, officialCurrency, Z))).

rule(attr(X, officialCurrency, Y2),
     ('Currencytypes_p'(X, Y1),
      'DStreamAF_p'(_6271, _6272, _6273, _6274, _6275, Y2),
      value(Y1, datastream, Y),
      value(Y2, datastream, Y))).

rule(attr(X, txnDate, Y),
     (olsen_p(_6313, _6314, X, Y))).
```

```
rule(attr(X, fromCur, Y),
     (olsen_p(_6351, Y, X, _6354))).

rule(attr(X, toCur, Y),
     (olsen_p(Y, _6390, X, _6392))).

rule(attr(X, countryIncorp, Y),
     (quotes_p(X, _6457),
      'WorldcAF_p'(C, _6448, _6449, _6450, _6451, _6452, Y),
      value(X, yahoo, Y1),
      value(C, yahoo, Y1))).
```

A.6 Integrity Constraints

```
%% integrity constraints for TASC Financial Example
%% generation timestamp: Sun Feb 03 06:00:30 EST 2002
%% by module: integrity_constraint_coin_internal
%% in package: edu.mit.gcms.gmm (graphical metadata management)
%% programmed by usman y. mobin (mobin@mit.edu)
%% massachusetts institute of technology

ic1_oracle_currencytypes @ '?''Currencytypes'(A, B1),
     '?''Currencytypes'(A, B2)
     ==> B1 = B2.

ic2_oracle_currencytypes @ '?''Currencytypes'(A1, B),
     '?''Currencytypes'(A2, B)
     ==> A1 = A2.

ic3_oracle_currency_map @ '?''Currency_map'(A, B1),
     '?''Currency_map'(A, B2)
     ==> B1 = B2.

ic4_oracle_currency_map @ '?''Currency_map'(A1, B),
     '?''Currency_map'(A2, B)
     ==> A1 = A2.

%%
%% olsen(Exchanged, Expressed, Rate, Date).
%%
ic5_cameleon_olsen @ '?'olsen(A, B, C1, D),
     '?'olsen(A, B, C2, D)
     ==> C1 = C2.

%%
%% quotes(Cname, Last).
%%
ic6_cameleon_quotes @ '?'quotes(A, B1),
     '?'quotes(A, B2)
     ==> B1 = B2.

%%
%%   DiscAF(company_name,   latest_annual_data,   current_shares_outstanding,
      net_income,
%%       net_sales, total_assets, country_of_incorp).
%%
ic7_oracle_discaf @ '?''DiscAF'(A, B1, C1, D1, E1, F1, G1),
     '?''DiscAF'(A, B2, C2, D2, E2, F2, G2)
     ==> B1 = B2, C1 = C2, D1 = D2, E1 = E2, F1 = F2, G1 = G2.

%%
```

```
%%            WorldcAF(company_name,            latest_annual_financial_date,
      current_outstanding_shares, net_income,
%%       sales, total_assets, country_of_incorp).
%%
ic8_oracle_worldcaf @ '?''WorldcAF'(A, B, C1, D1, E1, F1, G1),
     '?''WorldcAF'(A, B, C2, D2, E2, F2, G2)
     ==> C1 = C2, D1 = D2, E1 = E2, F1 = F2, G1 = G2.

%%
%%           WorldcAFT(company_name,            latest_annual_financial_date,
      current_outstanding_shares, net_income,
%%       sales, total_assets, country_of_incorp).
%%
ic9_oracle_worldcaft @ '?''WorldcAFT'(A, B, C1, D1, E1, F1, G1),
     '?''WorldcAFT'(A, B, C2, D2, E2, F2, G2)
     ==> C1 = C2, D1 = D2, E1 = E2, F1 = F2, G1 = G2.

ic10_oracle_name_map_ds_ws @ '?''Name_map_Ds_Ws'(A, B1),
     '?''Name_map_Ds_Ws'(A, B2)
     ==> B1 = B2.

ic11_oracle_name_map_ds_ws @ '?''Name_map_Ds_Ws'(A1, B),
     '?''Name_map_Ds_Ws'(A2, B)
     ==> A1 = A2.

%%
%% Ticker_Lookup2(comp_name, ticker, exc).
%%
ic12_oracle_ticker_lookup2 @ '?''Ticker_Lookup2'(A, B1, C1),
     '?''Ticker_Lookup2'(A, B2, C2)
     ==> B1 = B2, C1 = C2.

ic13_oracle_ticker_lookup2 @ '?''Ticker_Lookup2'(A1, B, C1),
     '?''Ticker_Lookup2'(A2, B, C2)
     ==> A1 = A2, C1 = C2.

ic14_oracle_name_map_dt_ds @ '?''Name_map_Dt_Ds'(A, B1),
     '?''Name_map_Dt_Ds'(A, B2)
     ==> B1 = B2.

ic15_oracle_name_map_dt_ds @ '?''Name_map_Dt_Ds'(A1, B),
     '?''Name_map_Dt_Ds'(A2, B)
     ==> A1 = A2.

ic16_oracle_name_map_dt_ws @ '?''Name_map_Dt_Ws'(A, B1),
     '?''Name_map_Dt_Ws'(A, B2)
     ==> B1 = B2.

ic17_oracle_name_map_dt_ws @ '?''Name_map_Dt_Ws'(A1, B),
     '?''Name_map_Dt_Ws'(A2, B)
     ==> A1 = A2.

%%
%% DStreamAF(as_of_date, name, total_sales, total_extraord_items_pre_tax,
%%       earned_for_ordinary, currency).
%%
ic18_oracle_dstreamaf @ '?''DStreamAF'(A, B, C1, D1, E1, F1),
     '?''DStreamAF'(A, B, C2, D2, E2, F2)
     ==> C1 = C2, D1 = D2, E1 = E2, F1 = F2.

ic19_cameleon_moneyrates @ '?'moneyrates(A, B1, C1, D1),
     '?'moneyrates(A, B2, C2, D2)
     ==> B1 = B2, C1 = C2, D1 = D2.
```