Document Number:	D1.3
Project Name:	Infomix: Boosting the Information Integration
Project Number:	IST-2001-33570
Title:	**Review of Techniques and Systems for Acquisition and Transformation of Heterogeneous Data**
Authors:	Thomas Eiter, Wolfgang Faber, Michael Fink, Daniel Kurushin (TUWIEN)
Workpackage:	WP1
Document Type:	Deliverable
Classification:	Confidential
Distribution:	Public
Status:	Draft
Document file:	
Version:	1.1
Date:	July 31, 2002
Number of pages:	35

Document Change Record		
Version	**Date**	**Reason for Change**
v.0.1	July 9, 2002	Document and index creation
v.0.2	July 12, 2002	Included Daniel Kurushin's material
v.0.3	July 14, 2002	Written Taxonomy section, consolidate material
v.0.4	July 15, 2002	Laid down Introduction and Conclusion
v.0.5	July 16, 2002	Consolidate references, overhaul Taxonomy
v.0.6	July 17, 2002	Fix features and properties tables
v.0.7	July 18, 2002	Include the tools by Rodan Systems
v.0.8	July 22, 2002	Incorporate suggestions by Rodan Systems
v.0.9	July 26, 2002	Incorporate some suggestions by UNIROME and UNICAL
v.0.95	July 28, 2002	Change taxonomy, reordering
v.0.96	July 29, 2002	Add bibliography, enhance some system descriptions
v.1.0	July 29, 2002	Restructure and comment overview tables
v.1.1	July 30, 2002	More references, little fixes

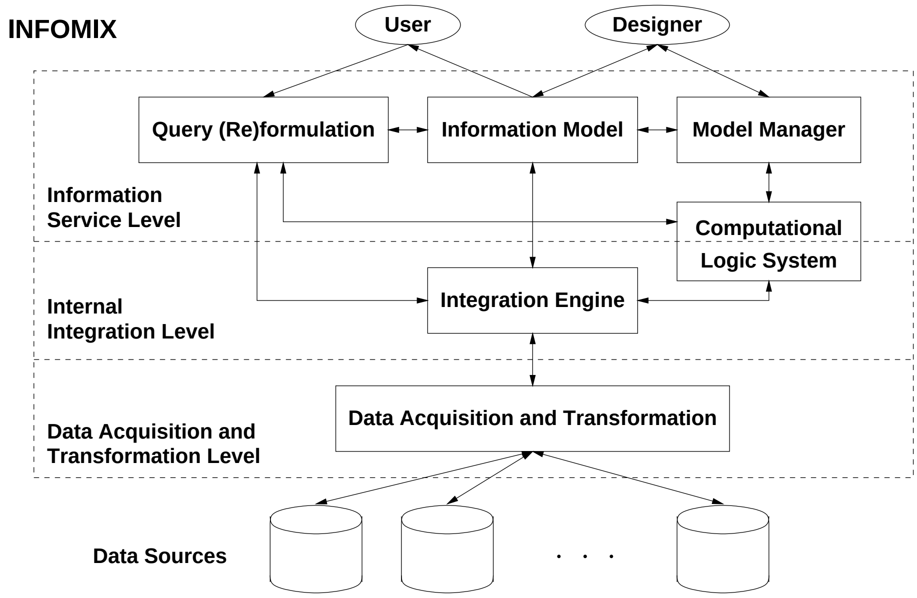

Figure 1: Prospective INFOMIX Architecture

1 Introduction

The INFOMIX project aims at providing a uniform interface to various pre-existing data sources, so as to enable users to focus on specifying what they want, rather than thinking of how to obtain the answer. INFOMIX will free the users from the tasks of finding the relevant data sources, interacting with each source in isolation, and selecting, cleaning, and combining data from multiple sources. In order to achieve this ambitious goal, it is necessary to provide a solid machinery at the low level, which interacts with the various information sources and accesses, queries and extracts those informations, and subsequently transforms them into a uniform intermediate representation.

Research on multi- and federated database systems has made available a number of techniques and systems for accessing heterogeneous data at a homogenized level. One of the INFOMIX objectives is to make use of selected existing results and techniques in the area of heterogeneous data acquisition and transformation, and to integrate them into the architecture of an advanced information integration system. Besides retrieval based methods, also the use of techniques for extracting information should be considered. As a result, a much more powerful system for combining heterogeneous data will be provided.

The prospective architecture of the INFOMIX system consists of three principal layers, as depicted in Figure 1. This document focuses on the bottom layer, which accomplishes Data Acquisition and Transformation (DAT). The Data Sources in Figure 1 can be of heterogeneous nature, ranging from fully structured data (most prominently relational databases) over semi-structured data (like XML data) to poorly structured or even unstructured data (like HTML data or plain text documents).

In the literature, the term "wrapper" is frequently used to describe the kind of software modules which will be the basic part of the DAT layer. We use this term in a very broad sense: In our terminology, anything which "wraps around" some data source in order to extract, query or otherwise access and possibly also transform the information provided by the data source will be referred to as a *wrapper*.

Therefore, a wrapper could be anything from an SQL interface to a database system to an intelligent data extraction and data merging programming toolkit. Note that in recent literature [68, 46, 11, 64, 25], the term wrapper is frequently used in a much more narrow sense, namely for tools which turn unstructured or poorly structured data into structured data, or even more specific, HTML extraction tools.

The goal of this document is to survey and categorize the most important methods and existing software in this area. Doing this, it has to be borne in mind that the definitive architecture of the INFOMIX system has not been fixed yet; in particular, the interface between the DAT and the Internal Integration Level, which is characterized by an intermediate data format, has not been specified yet.

The remainder of the document is organized as follows: In Section 2, parameters for categorizing the available methods and software will be devised. Then, in Section 3, these methods and tools will be presented in detail and analyzed along these parameters. In Sections 4 and 5, an overview from the viewpoint of the features and the properties of the systems and methods will be given in a concise form. Finally, in Section 6, recommendations are given based on the analysis in the previous sections, and conclusions regarding the future of INFOMIX are drawn.

2 Taxonomy

In this section we will develop a taxonomy for methods and systems for acquisition and transformation of heterogeneous data to be used in the INFOMIX project. We will evaluate taxonomies of related reviews and will adapt them to better suit the requirements of the INFOMIX project.

2.1 Existing Taxonomies

In [69], available at `http://www.wifo.uni-mannheim.de/~kuhlins/wrappertools/`, and also in [68], a detailed analysis of wrapper-generation software is given. There, various tools are classified according to various criteria. The principle differentiation is between "Freeware" and "Commercial Tools," with further criteria being:

- Data output;
- Availability of Java API;
- Open Source;
- Language of source code;
- Presence of GUI;
- Presence of a builtin editor;
- Scripting language;

- Further features;
- Release dates.

This taxonomy does therefore not consider some properties, which are crucial for the INFOMIX DAT layer: The input data is not clearly specified, and low-level data integration features are not or only superficially considered. On the other hand, several of the criteria (like GUI or built-in editor existence, source code language) are not crucial or even irrelevant for the INFOMIX DAT layer.

Another type of classification is given in [46]. In this paper, web data extraction tools are classified as

- Languages for wrapper development;
- HTML-aware tools;
- NLP[1]-based tools;
- Wrapper induction tools;
- Modeling-based tools;
- Ontology-based tools.

So, while [69] is a feature-based classification, the one in [46] stresses the mode of wrapper creation, which is not of great interest for the INFOMIX project. From this perspective, the taxonomy in [69] would be better suited for the purposes of this document.

Both of these classifications focus on wrapper-generation tools, but not on the wrappers proper, while in this document the nature of the wrappers is actually the main focus. The most important parameters are the supported input and output datatypes of the wrappers, the internal features they can provide, and to a lesser extent the nature and availability of the work associated to the wrappers and wrapper generators.

2.2 A New Taxonomy

Based on the considerations on existing work and additional desiderata in the previous paragraphs, we will now propose a novel taxonomy to be used in the sequel of this document in order to classify wrapper generators and the wrappers they produce.

The taxonomy consists of four principal criteria:

1. **Input data format** (relational, semi-structured, unstructured);
2. **Output data format** (XML, OEM, HTML ...);
3. **Features** (data cleaning, data merging, parsing, changing structure);
4. **Sort of project** (theoretical, open source, commercial).

[1]Natural Language Processing

Concerning **input data format**, we will further divide systems and methods into three sub-categories:

1. Relational data format.
   These are usually systems working on a top of some SQL server like MySQL, Oracle or PostgreSQL. Most of these wrappers are used to convert data from SQL server into some textual format. They can use an SQL-like language to do some work on data integration.

2. Semi-structured data format.
   These are methods and systems ranging from XML query languages to advanced integration engines. All of them explicitly exploit the semi-structured nature of the data sources.

3. Unstructured or poorly structured data format.
   These wrappers are usually fuzzy data extractors. Due to the lack of structure of the input data, methods from natural language processing, machine learning and similar research areas are used here. The most severe problem to be tackled in these approaches is the inherent inexactness of the wrappers.

Concerning the **output data format**, one could also differentiate between systems producing relational, semi-structured or unstructured, but for this review we are interested mainly in systems which can produce some sort of semi-structured data, and in particular XML or a subset thereof, so we will only present systems which fall into this category. A criterion is also the concrete format of the output data, for example XML, OEM or formatted text.

The category **features** covers abstract features of wrapping technology which do not strictly determine the internal organization of the described systems, for example *data cleaning*, *data merging*, HTML or XML *awareness* or *changing structure* of wrapped data. The simplest wrappers can only extract some data from a source and make it available for further processing. More complex wrappers usually can parse source data and retrieve its structure. Also some work on changing structure of source data can be done at wrapper level. The most complex projects allow for some limited forms of data cleaning or even data merging. Usually such wrappers possess a language which allows for describing their behavior.

While discussing the **sort of project** we will differentiate between *theoretical works*, *open source software* and *commercial software* (both freeware and non-freeware). When considering theoretical works we will pay special attention to the papers in which new approaches to data wrapping or data merging are introduced or in which some language is described. Concerning systems an important point is the availability of source code because it could be useful for further development within the INFOMIX project. Software for which the source code is unavailable can still be useful for use as a tool or just as a model for creating new software.

In the following, we will use the **input data format** as the principal characterizing criterion. As outlined above, we will differentiate between the following types of wrappers:

1. Structured Data Wrappers

2. Semi-structured Data Wrappers

3. Unstructured Data Wrappers

While wrappers for structured and semi-structured data are usually only capable of dealing with their respective input data format, wrappers for unstructured data can typically also be used for wrapping semi-structured data, but normally not for fully structured data.

3 Systems and Methods Considered

Along the taxonomy developed in the previous section, with input data format being the principal differentiator, we will consider structured data wrappers in Section 3.1, semi-structured data wrappers in Section 3.2, and unstructured data wrappers in Section 3.3.

3.1 Structured Data Wrappers

In this section we consider tools for dealing with relational data provided by a traditional database or strongly related systems. While in principle any (relational) database management system ([R]DBMS) provides a (limited) form of wrapping functionality, we will only consider systems which either attach themselves to an RDBMS or which provide significantly more functionality than a traditional DBMS.

3.1.1 DBIWrapper

DBIWrapper [60] has been originally created by PC&Web Xperience, Inc. for use in some internal web projects and has subsequently been published as open source software.

The acronym DBI stands for "the Perl database interface" [20, 21] and is a database interface module for the programming language Perl. It defines a set of methods, variables and conventions that provide a consistent abstract database interface independent of the actual database being used. DBI can interface to a variety of RDBMSes, among them Oracle, Sybase, Ingres, Postgres, and Informix.

DBIWrapper is a further abstraction of DBI, providing high level methods for reading, writing, committing and rolling back transactions. It currently supports MySQL, PostgreSQL and ODBC DBD drivers. Since DBIWrapper is a Perl module, it naturally provides a Perl API.

DBIWrapper is able to produce XML output via its `readXML()` function. Basically, one can have the result of an SQL query returned as an XML document which includes the issued query, answer status, and the rows of data returned from the back-end.

An example XML output is given in Figure 2. All the data is inside the tag `resultset`. In the `select` tag, attribute `sql` holds the original SQL statement, which produced the output in question. `plug` would contain data passed through the Perl interface. The attribute `status` tells whether the execution produced an error. An attribute value `Ok` for `result`, as in this example, indicates that the execution was successful, otherwise `result` would be `Error`, and attribute `error` would provide an error message.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<resultset version="1.1">
  <select sql="SELECT c1,c2 FROM r" plug="" />
  <status result="Ok" error=""/>
  <rows numRows="2" columns="1">
    <row>
      <column name="c1" value="v1" />
      <column name="c2" value="v2" />
    </row>
    <row>
      <column name="c1" value="v3" />
      <column name="c2" value="v4" />
    </row>
  </rows>
</resultset>
```

Figure 2: Example DBIWrapper XML output

Finally, the tag `rows` specifies the query result, `numrows` indicates the number of tuples in the result, while `columns` determines the syntax of the following `row` tags. In our example, the value `1` for `columns` indicates that each `row` tag (representing a tuple) will contain `column` tags, representing the values for each columns of the respective tuple. Otherwise, if the attribute `columns` is `0`, then each `row` tag will contain the columns and their values as attribute-value pair, as opposed to separate `column` tags. Figure 3 shows the same result data as in Figure 3 in the alternative format. In this second representation, the column names must be valid attribute strings in XML.

Concerning the sort of project, both DBI and DBIWrapper are released under the (Perl) Artistic License (cf. `http://www.opensource.org/licenses/artistic-license.php`), which means that the software is open source, free, and may be embedded in

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<resultset version="1.1">
  <select sql="SELECT c1,c2 FROM r" plug="" />
  <status result="Ok" error="" />
  <rows numRows="2" columns="0">
    <row c1="v1" c2="v2" />
    <row c1="v3" c2="v4" />
  </rows>
</resultset>
```

Figure 3: Example DBIWrapper XML output

commercial applications, as long as the interface is hidden by the application. We are not aware of any theoretical work directly associated with DBI or DBIWrapper.

3.1.2 Bonddb

Bonddb [40] is an object oriented wrapper for RDBMSes. Bonddb is part of the BOND (building object network databases) project [39], which focuses on easing the engineering of database applications.

It is a fast data abstraction layer written in C for C/C++ applications to allow an easy access to class objects. Bonddb is not a stand-alone object oriented database, but works on top of SQL and transparently implements features like persistent objects.

Bonddb can be viewed as an object-oriented wrapper around a relational database system, so its input is structured data, and its output is object-oriented data, which can be processed via the featured API. At the moment only PostgreSQL is supported as the underlying relational database system.

The base part of Bonddb is the 'Object' structure. This would typically contain an SQL select query of some kind. Using this object one can then navigate rows by using the `db_moveto()` function. It is also possible to set and get values of specific fields in the Object based on a current row.

Normally it is not necessary to use SQL statements to get 'Objects', there is a host of functions already provided for creating objects based on commonly requested queries. An example of this would be db_loadall or db_loadobjectbyobject etc. Bonddb looks after the relationship information, so one does not have to specify the joins.

An interesting feature of Bonddb is the ability to link objects together, so one object can contain many sub-objects which are automatically updated based on the parent. dbmapme.c provides a lot of this functionality. The overall logging system will also update all objects currently loaded if their contents change on the database side. This is useful while doing a lot of requests at the same time and one does not want to have to refresh manually all the record sets every time a writing is performed.

Unfortunately, concerning the features in our taxonomy, Bonddb cannot provide any of data cleaning, data merging, nor is it XML or HTML aware. Changing structure of the data is only possible via the API. Bonddb is released under the Gnu Public License (GPL), which means that it is open source and free, but also all applications incorporating it must be released under the GPL.

3.1.3 DBedit

DBedit [1] is a system for integrating development web database applications. It is being developed and used by the Globewide Network Academy to create and edit database tables over the web.

DBedit is unique among web database products because it abstracts the display of the information from the storage. In order to display a database, you create a straight HTML form, and then run a Perl script on it which generates a CGI wrapper to the script. The advantages of this are:

- Possibility to add arbitrary Perl code to database applications

- Integration of an application with any database by writing a driver routine. Because display and storage are separated, the user can then change the database implementation without rewriting any forms, or change some forms without rewriting the database.

- It runs with DBI so switching between databases should be easy.

Right now there is a driver for DBI which has been tested with PostgreSQL. It should also work with Oracle and MySQL. There is also a driver for flat ASCII text files using the file format for the commercial database system "/rdb" by Revolutionary Software. DBedit is a set of Perl scripts and the only requirement is that Perl (Perl5.004) must be installed on the system. Both software systems run on Windows95, NT and Unix based machines.

Since the database parts of the code are modularized, it should be a very simple matter to interface the system with other text formats and database systems.

3.1.4 XBSQL

XBSQL [62] is an SQL Wrapper for the XBase library which is a C++ library that supports access to XBase type data files and indexes (ie., .dbf and related files). It provides access at the record level to these files.

XBSQL is a wrapper library which provides an SQL-subset interface to XBase DBMS, built on xbase-1.8.1. Currently, XBSQL is in a rather preliminary stage of development, so various features are not implemented yet, however there is probably sufficient functionality to declare the library to be at least usable. Unfortunately, the documentation is very limited. There is a simple command-line client, `xql`, which can be used as an example application.

XBSQL currently supports the following basic commands:

- `create table tabname (column`$_1$`, ...)`

- `drop table tabname`

- `insert into tabname values (expr`$_1$`, ...)`

- `insert into tabname (column`$_1$`, ...) values (expr`$_1$`, ldots)`

- `select expr`$_1$`, ...from table`$_1$`, ...where condition`$_1$`, ...order by order`$_1$`, ...`

- `update tabname set column`$_1$` = expr`$_1$`, ...where condition`$_1$`, ...`

- `delete from tabname where condition`$_1$`, ...`

XBSQL supports expression names and table aliases, so it is possible to formulate SQL queries like:

- `select T1.a as A1, T2.b as A2 from Table T1, Table T2 where T1.x = T2.y`

XBSQL also supports single-column `.ndx` indexes, hence the following indexes the columns `i` and `t`. The library can make use of these indexes for simple join cases.

- `create table tabname (i integer index, t char(80) index)`

Further supported features include

- Aggregate functions, including `count(*)`
- `group by` and `having`
- Sub-selects
- `create table tabname as ...`
- `alter table tabname ...`

3.1.5 Multiplex

The Multiplex [7, 57, 58] system is a formal model for working with multi-databases. Its architecture is sketched in Figure 4. Multiplex is a simple extension of the relational model, so it works with structured data. Multiplex is based on formal assumptions of integrability, which distinguish between scheme and instance reconcilability among independent databases. Multiplex supports database heterogeneity, and it provides several degrees of freedom that allow it to model actual situations encountered in multi-database applications. In addition, in situations in which a single answer is not obtainable (either because the global query is not answerable, or there are multiple candidate answers), Multiplex defines approximative answers. Finally, Multiplex provides a practical platform for implementation. A prototype of such an implementation is described as well.

Multiplex is an extension of the relational model. It extends the definition of schemes, constraints, queries and answers to an environment of multiple databases, and it provides a common ground for other definitions as they become necessary. The extension also maintains the attractive simplicity of the relational model, with relatively few new concepts. Multiplex defines formal assumptions of integrability. Most integration methods have operated under tacit assumptions regarding the mutual consistency of the schemes and/or the instances of the underlying databases. Multiplex defines two kinds of inconsistency (intensional and extensional) and formulates two assumptions of integrability: the Model Consistency Assumption and the Instance Consistency Assumption. These explicit assumptions, which have been absent in previous work, provide an unambiguous framework, and help to classify other integration approaches. Multiplex supports heterogeneity among the independent databases with no restriction on the underlying data models; the only requirement is that they communicate their results in tabular form. Consequently, the independent databases in a Multiplex multi-database may be relational, object-oriented, or, in general, stored in any software system that responds to requests with tabulated answers.

Multiplex has a powerful SQL-like language to describe behavior in situation of inconsistent data. It enables the formulation multi-database queries, but also implements some new operations.

One of them is the "resolve" statement with the following structure:

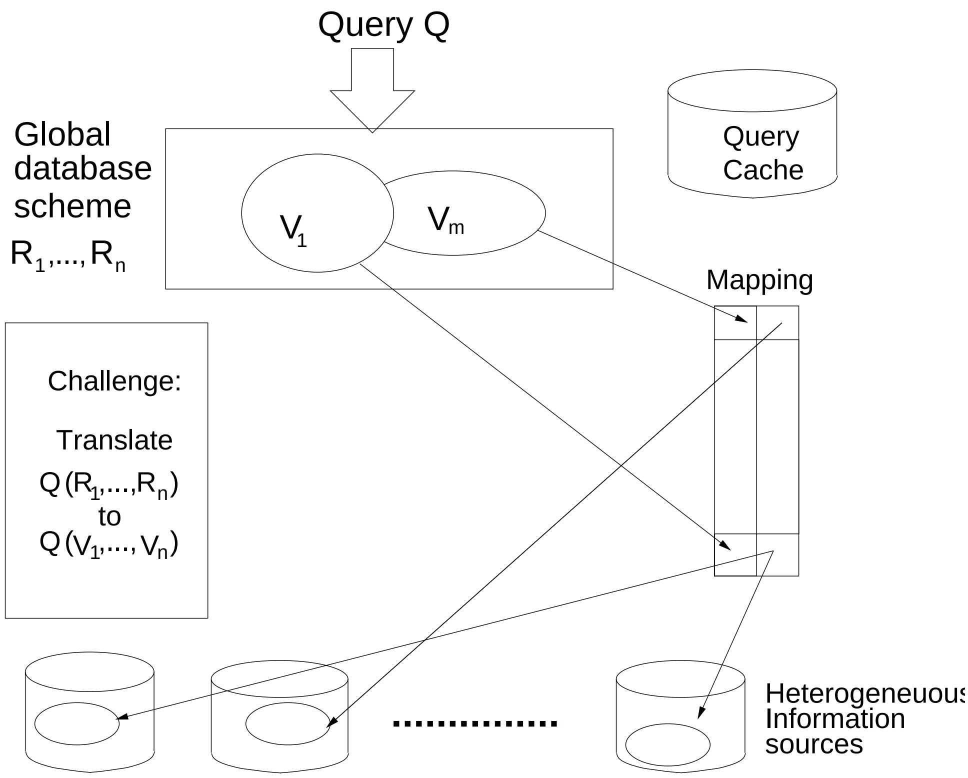

Figure 4: Architecture of the Multiplex system

```
resolve permissive, restrictive R
  if F1 then statements of strategy1
  else [if F2 then...]  statements of strategy2
```

where R is a relation name, "permissive" or "restrictive" indicates how comparisons with null values should be handled: "permissive" treats all comparisons as true and "restrictive" as false.

The "strategy" has following structure:

```
select value-fusion
  from contribution-qualification where
  value-elimination
```

Unlike SQL, the "from" clause describes not a database or table for selecting but filtration condition. Example:

```
resolve permissive Employee
  select random(FirstName),mode(lastName),min(Salary)
  from quality ≥ 0.7
```

`where Salary` $\geq$ `40000`

It selects "Employee" data with good quality and only if "Salary" is more than 40000. The "quality" is a semantic field which can be described in different ways. For example it can be derived from timestamp of the record.

The Multiplex system is not available for downloading or online testing, so we will place here a short description of it.

The prototype consists of two software components: a generic relational DBMS for managing independent databases, and a generic multi-database management system, called Multiplex, that integrates the independent databases. The classical relational DBMS consists of four simple parts:

1. user interface;
2. query parser;
3. query optimizer;
4. query processor.

The system processes product-selection-projection queries specified in SQL-like syntax. It uses a database scheme file that describes the relation schemes, and it stores all its relation instances in B+ trees. The Multiplex DBMS has a very similar structure:

1. user interface
2. query parser
3. query translator
4. view retriever
5. query optimizer
6. query processor

The user interface, the query parser, the query optimizer, the query processor, and the database scheme file are essentially identical to those of the generic DBMS.

3.1.6 HERMES

HERMES [3, 4] (A HEterogeneous Reasoning and MEdiator System) is a system based on Hybrid Knowledge Bases (HKB) theory. It provides a general, declarative language for defining a mediator, which expresses semantic integration of information from diverse data sources and reasoning systems.

The main difference of HERMES from other systems described here is its ability to work with relational data sources:

- relational data in multiple heterogeneous formats in ASCII text files,

- relational databases under Borland's Paradox DBMS,
- relational databases under DBase V DBMS,
- text or other unstructured data.

Like other systems HERMES provides a mediator specification language ("Mediator Language") which allows for performing queries on heterogeneous data sources. HERMES assumes data to be presented as a set of *domains* and a mediator language allows access to some domain specific functionality by expressions like:
`domainname:<domainfunction>(<arg`$_1$`,...,arg`$_n$`>)`
where `<domainfunction>` is some domain-specific function and `<arg`$_1$`,...,arg`$_n$`>` are the arguments it takes. More examples of the mediator language syntax are given in [3].

There are some *toolkits* included into the HERMES system:

- The Domain Integration Toolkit;
- The Conflict Resolution Toolkit;
- The Information Pooling Toolkit;

The **Domain Integration Toolkit** enables a mediator to execute functions defined inside of a domain being integrated and converts the output into a form suitable for the mediator.

The purpose of the **Conflict Resolution Toolkit** is to assist the mediator author in dealing with semantic conflicts that might arise during semantic integration. For each predicate defined in the mediator one conflict resolution strategy can be provided.

The **Information Pooling Toolkit** consists of some *subtoolkits* which are designed for integration within a given domain as well as across multiple domains. Each subtoolkit is designed for specific task within integration process and aimed for working with a specific data source.

Also in [3] some examples of applications developed with HERMES are given. The HERMES software is available for Solaris and Windows (only 3.1 version is mentioned in the paper). For now the project is closed and online demos are disabled.

3.1.7 OfficeObjects Repository

OfficeObjects Repository is a versatile data repository system developed by Rodan Systems. It provides a variety of storing, accessing, searching, querying, and managing functionality for numerous types of heterogeneous data. Its architecture is schematically sketched in Figure 5.

OfficeObjects Repository features fast key-word search as well as full text search on the objects' attributes, semi-structured data and unstructured data (a significant number of native formats is supported, among them e.g. Microsoft Word). Conceptual trees support for recall and precision increase in search and categorization. The effectiveness of the search is achieved by relational database indices (all attributes of objects are stored in XML compliant format while searchable attributes are duplicated in relational database

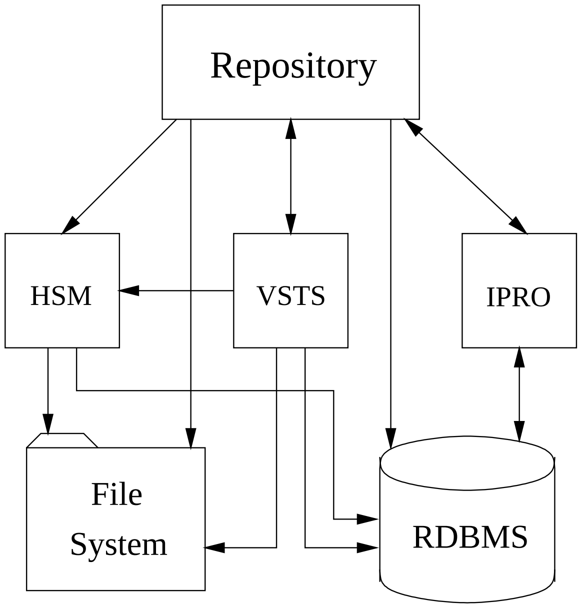

Figure 5: Architecture of OfficeObjects Repository

tables). OfficeObjects Repository offers the possibility of distribution of repository subsystems (repository kernel, full text search, programmable repository agents subsystem, relational database) over separated computers. Strong support for import and export (interoperability) of XML compliant bulk data (XML representation is explicitly stored in the repository) is given.

The tool currently supports the following platforms: Windows NT, SUN, HP, IBM. A Linux version will soon be available. Oracle, Informix, Sybase, and SQL Server can be used as underlying databases.

As RODAN is one of the INFOMIX project partners, the source code of this commercial product is available to the project. The development languages are C/C++, esql, ODBC, and JAVA. EXPAT is used as the XML parser.

Together with OfficeObjects Data Extractor (see Section 3.1.8,), OfficeObjects Repository can serve as highly versatile wrapper and extractor for data in a vast number of formats, even if they are not locally managed by OfficeObjects Repository.

3.1.8 OfficeObjects Data Extractor

OfficeObjects Data Extractor is a query by example (QBE) tool that is currently being developed by Rodan Systems to simplify the process of preparing reports from views on relational databases.

Unfortunately the term "query by example" is used in different ways in the literature; in the context of database management systems, QBE refers to a method of forming queries in which the database program displays a template record with a blank for each field. The user can enter conditions for each field that she wants to be included in the query. For example, if she wanted to find all records in which a fictitious "AGE" field is greater than 65, she would enter > 65 in the "AGE" field blank. In general, QBE systems are considered easier to learn than formal query languages.

OfficeObjects Data Extractor has been built as an application on top of relational database views that are SQL2 compliant, but can take any views on a relational database as input format. Typically, the views on which it works are fixed (defined through queries by the system designer), as constants for the particular implementation of the tool.

The tool works as follows: the user first picks a view from the given list of named views. Subsequently, she can choose the columns that she wants to use. She can order the columns and set the following options for each column:

- to show or not to show the column in the report
- to use or not to use the column as a parameter for the report
- specified sorting constraints for the column

As a feature, the settings by the user so far can be saved as a template.

Furthermore, in a table interface with three rows, three alternative constraints on the columns can be given, which is interpreted as a disjunction of a conjunction of constraints for each column. Finally, according to the template and the constraints from the table, a report is generated, and represented as a table.

The output format is alternatively a CSV table, a HTML table (which is tree-structured) or an XML data file. The output table can be imported in e.g. XSL for further processing. The tool has been designed in such a way that adaptations of the output format should not be hard.

OfficeObjects Data Extractor does not have additional features like data cleaning or integration. It does not have an API, rather it is a stand-alone application at the moment. However, an interface to the Rodan OfficeObjects Portal (see section 3.1.7) is currently being developed.

Work on this project is still going on; there is a working prototype, but the final version and the documentation is not in a final state yet. The product is scheduled to be finished by September 2002.

As RODAN is one of the INFOMIX project partners, the source code of this commercial product is available to the project. The tool is written in Java, and therefore there are no limitations as to the underlying computing platform: Windows, various Unix flavors etc. are all suited. The software requirements for running this tool are Java 1.3 and a relational database.

3.2 Semi-structured Data Wrappers

In this section we consider tools for dealing with semi-structured data, which can be XML, but also alternative formats like OEM (Object Exchange Model).

A number of *query languages for semi-structured data* has been proposed in the recent literature, and they could indeed be considered as wrappers for semi-structured data. Overviews of XML query languages are given in e.g. [30] and [13]. The most famous semi-structured query languages are Lorel [2, 53, 36], XML-QL [22, 23] and XQL [63, 65].

While these languages naturally produce XML output, their functionality as a wrapper is actually fairly limited. For this reason we will not deal with these language in this report.

3.2.1 XMLwrapp

XMLwrapp [43] is a clean and easy to use C++ wrapper around popular XML parsers. It does not mean to expose every feature of the back-end parser but instead to support a high level API for working with XML data.

Currently, XMLwrapp only supports LibXML2 as a back-end parser. It is written in a way such that it should be very easy to add more back-ends. As soon as the API receives acceptance, work will begin on adding more back-ends.

Features of XMLwrapp:

- Tree parsing. XML data is parsed and a tree of XML::node objects is created.
- Event parsing. XML data is parsed and protected member functions of an event class are called.
- It is easy to construct an XML tree using XML::node objects. Any XML::node may be inserted into an IOStream causing translation to XML text data.
- Complete isolation from the back-end parser due to the private implementation (pimpl) idiom.

3.2.2 XWRAP

XWRAP can be classified both as unstructured and semi-structured data wrapper. XWRAP is actually an HTML wrapper generation software, but its algorithm, which produces wrappers for HTML documents [49, 48, 14], uses XML for the representation of the content of the document. At the step *source normalization*, the HTML file is converted into an XML-tree with standard HTML tags.

Functionality of wrappers generated by XWRAP is only wrapping itself – no data mediation, data cleaning or conflict detection can be accomplished. The authors of XWRAP suppose this work to be done at the level of the application using the wrappers.

The wrapper generator uses a three-step algorithm:

- The first step is *information extraction*. At this step, a structure analyzer allows the user to define regions in the HTML code which will be put into corresponding XML containers. Some heuristic analysis of HTML is used at this step to identify which parts of the document can be headings.
- The second step is *source-specific XML generation*. This is done by a special *generator* which consists of the XML Template Engine and the XML parser. The XML Template Engine generates XML statements based on XML templates which are parsed by XML parser. XML template explains how wrapped data should be inserted into the output. Templates can also contain *repetitive parts* which correspond to loops in programming languages.

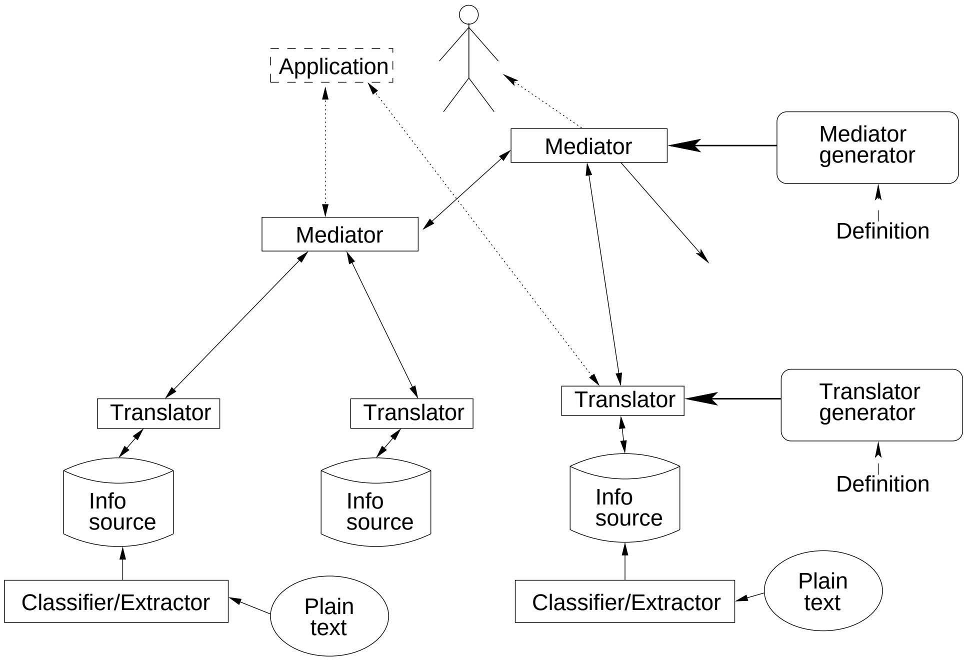

Figure 6: General structure of TSIMMIS

- The last step is *XML-based content filtering* which is done by an interface between the mediator application and the wrapper. The main tasks of filters are:
  - Accept a mediator query, identify network locations of the pages needed to answer the query, and call the data wrapper manager to wrap the source pages into XML format.
  - Translate the complex content-sensitive mediator queries over given Web pages to SQL-like XML queries.

3.2.3 Infomaster

Infomaster [35, 24] is an information integration tool which has been developed at Stanford University. Infomaster is a realization of a integration system based on *knowledge interchange format* (KIF) [67]. Infomaster can also make use of the Z39.50 [12] format for bibliographic data. Z39.50 is a particular format to represent semi-structured data.

Being a data integration system Infomaster contains an *interface*, a *query engine* and a *wrapper layer*. Infomaster can also work with SQL databases and HTML pages by using special wrappers.

3.2.4 TSIMMIS

The TSIMMIS [16, 33, 34, 38, 37] system has also been developed at Stanford University. TSIMMIS is an acronym for "The Stanford-IBM Manager of Multiple Information Sources" Its basic structure is shown in Figure 6. At present the project is closed, but the source code is available for free download for scientific usage.

The goal of the TSIMMIS project has been to develop tools that facilitate the rapid integration of heterogeneous information sources that may include both structured and unstructured data. TSIMMIS consists of a number of wrappers and mediators which allow it to work with different data sources. Each data source requires a special wrapper ("translator").

The data model used in TSIMMIS is called "self-describing" or "Object Exchange Model" (OEM). OEM is very simple, while providing power and flexibility which is needed for information integration. The query language for this model was also developed by the TSIMMIS group. This language, called OEM-Q, is an SQL-like query language.

Data provided by some data source and transformed by a wrapper arrive at a mediator. A mediator is a system that refines information from one or more sources. A mediator has some knowledge which is necessary for processing a specific type of information.

The TSIMMIS system features the special language MSL (Mediator Specification Language), which allows for describing the behavior of the system. This language is used to describe the generators for wrappers and mediators. The following example of an MSL query shows how OEM objects can be searched. Consider an OEM object of the following structure:

```
<ob1, person, {sub1, sub2, sub3, sub4, sub5}>
    <sub1, last_name, 'Smith'>
    <sub2, first_name, 'John'>
    <sub3, role, 'faculty'>
    <sub4, department, 'CS'>
    <sub5, telephone, '415-5141292'>
```

`ob1`, `sub1` to `sub5` are object-ids, which uniquely define an object, with `sub1` to `sub5` being sub-objects of `ob1`. Each object has a label explaining its meaning (e.g., `department`), and a value (e.g., `'CS'`). Such a value can be a scalar such as an integer or a string, or it can be a set of (sub)objects (e.g., the value of the "person" object).

A query searching for top-level person objects (i.e., objects with `person` labels) containing a `last_name` sub-object with value `'Smith'`:

```
*P :- <P person {<L last_name 'Smith'>}>
```

The query consists of a single head and a single tail separated by the ":-" symbol. Variables are represented by identifiers starting with capitals (P, L). The tail describes the search pattern while the head is the object-id of objects to be retrieved. The $*$ in the head of the query indicates that sub-objects are retrieved too. Without the asterisk, a single object is retrieved.

Queries can be organized in the following way:

```
*P :- <P label1 {...}>
 P :- <P label2 {...}>
    ...
 P :- <P labeln {...}>
```

In this case TSIMMIS will search for each tail until the search is successful.

TSIMMIS works with object-identity-based information fusion. It runs a query at mediation level and sends requests for data to the wrappers based on the labels of the data. If some data sources on the same query Q return different data D_1, D_2 and D_3 TSIMMIS should block all data sources except one. It looks as if it chooses data from a random data source in this case. So, the answer will be $\{D_i\}$, where i can be 1, 2 or 3. On the other hand, if answers are A, B and C the answer will be $\{A, B, C\}$. Also in case of A,B_1 and B_2 the answer is $\{A, B_i\}$ where i is 1 or 2.

As we have mentioned above TSIMMIS is available for download from `http://www-db.stanford.edu/tsimmis/download.html`. Unfortunately compiling produced a lot of problems with our computing environment. TSIMMIS requires a number of header files which were not included in our UNIX distribution, and we were not able to produce a working system. Also since this project has already been closed in Stanford, there is no technical support or any new technical information available. Online demos of TSIMMIS are also disabled.

TSIMMIS seems to be a good theoretical model for logic programming queries on heterogeneous data, and there are a lot of publications available.

A serious drawback of TSIMMIS is that it cannot work with XML which has become a standard tool for representing semi-structured data.

3.3 Unstructured Data Wrappers

In this section we consider systems and methods which allow wrapping poorly structured data (such as HTML data) or unstructured data (such as plain text documents). While there is actually a large body of related literature and systems on this topic (cf. [46, 68]), we will pick out the systems which seem to suit the INFOMIX needs best. We will briefly comment on the several unconsidered systems at the very end of this section.

3.3.1 LoPiX

The system LoPiX [51] (Logic Programming in XML) has evolved from work on FLORID [32, 50] and is fully XML aware. The basic structure of LoPiX is sketched in Figure 7.

An XML database can be loaded into internal structures of LoPiX by a single operation:

```
object[@xml->‘url‘ url type].
```

for example:

```
mondial[@xml->‘file:mondial.xml‘ isa url].
```

LoPiX was developed as an implementation of XPath-Logic, described in [52] As XML was designed and accepted as the generic framework for semi-structured data, a number of special languages for querying XML were created by different organizations or scientific groups. LoPiX was developed to extend operations with XML from only querying to manipulation with the stored data. Unlike the Document Object Model (DOM), LoPiX provides a logic-based approach to data manipulation. An XML instance is mapped to a semantical structure which allows XPath-Logic formulas to be interpreted. These formulas can be used for specifying constraints and reasoning on XML documents.

The LoPiX system can construct XML data, which is demonstrated in the following example which uses the geographical database “mondial” which is included in the LoPiX

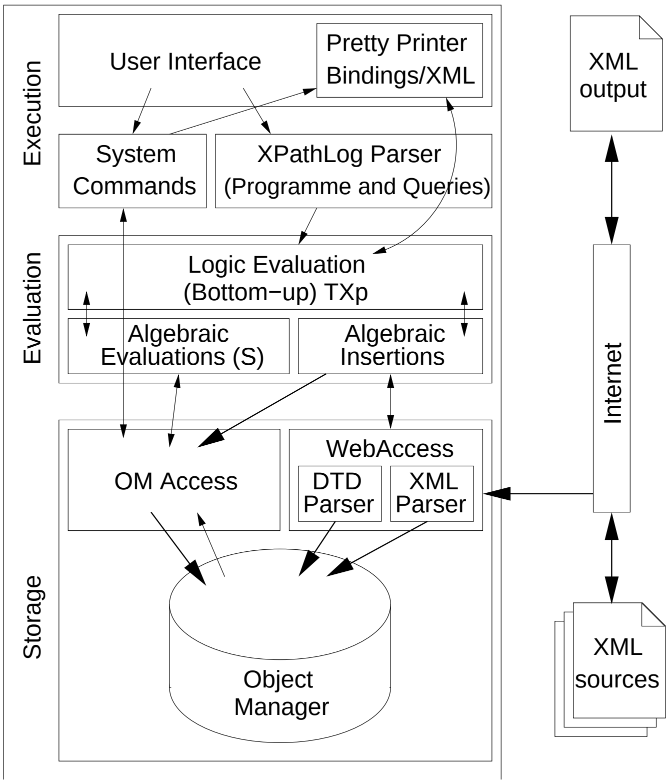

Figure 7: General structure of LoPiX

distribution:

```
bavaria :- //country-> C[name/text()-> "Bavaria"].
//bavaria/ethnicgroups[text()->E and @percentage->P] :-
    germany/ethnicgroups[text()->E and @percentage->P].
```

produces:

```
<country id=‘bavaria‘ capital=‘f0_237‘ car_code=‘BAV‘>
  <name id=‘f0_811‘ >
  <ethnicgroups id=‘f0_812‘ percentage=‘0.4‘>
    .........
  <city id=‘f0_237‘ country=‘D‘>
    <name id=‘f0_238‘ > <population id=‘f0_242‘ year=‘95‘>
  </city>
    .........
</country>
```

In this example the first line searches for countries with name “Bavaria” in an example database and puts the result into the structure `bavaria`. The second line takes this structure and searches for `ethnicgroups` with the same name `E` and percentage `P` in the structure `germany`. So it will print out everything from `<country><ethnicgroups>`

with text "Bavaria" if percentage and text of `ethnicgroups` are the same in Bavaria and Germany.

Like TSIMMIS, LoPiX works with labeled data; the only difference is how it is processed before it comes to the query engine. In the LoPiX system, XML data is processed by a parser which extracts objects from it and puts them into the "object manager". The idea is to work with data in a "tree view". Some tree operations are defined:

- **Projection by Signature** is specified by a root node and a description specifying which elements and attributes should belong to the view ("signature").
- **Generating an Isolated Result Tree** is an opposite strategy to a projection. It's creating a new tree from scratch by collecting nodes and building new elements and sub-elements relationships.
- **Generating a Result Tree by Selecting and Linking Subtrees** is a composite between the two "extreme" strategies described above.

Also some merging operations are available:

- **Synonyms** are a means for taking a whole property from the source tree to the result tree. This does not introduce new children but defines a new access path.
- **Fusing elements and Subtrees**. Here, attributes and sub-elements from both original elements (or subtrees) are collected.

The result of merging is always a mixture of source data. No work on inconsistency or conflict resolution is implemented. If a query produces object O with attribute values A and B, the result will be $O\{A, B\}$. If attribute names are also different, the result will be $O\{a = A, b = B\}$, where a and b are attribute names.

LoPiX is available for download as a binary distribution from `http://www.informatik.uni-freiburg.de/~may/lopix/`.

3.3.2 LiXto

LiXto [11, 10, 8] is a tool for visual wrapper generation. It parses HTML by using a Java Swing parser and allows user to select regions to copy data from them into the target XML file. LiXto offers two basic mechanisms of data extraction – tree and string extraction.

- For tree extraction, LiXto identifies elements with their correspondent tree paths and possibly some properties of the elements themselves. For example `*.table.*.tr` is a valid tree path.
- The second extraction method relies on strings. In the HTML parse tree, strings are represented by the text of content leaves. In LiXto, strings are associated to every node of a parse tree available as a value of the attribute *elementtext*. A substring of elementtext of a HTML tree is denoted as *string source*. For extracting substrings, regular expressions are used.

For parser generation LiXto uses the *Elog* programming language [9]. Elog is a datalog-like language designed for wrapper generation. Examples of programs in Elog are given in [11, 9]. LiXto stores Elog programs in XML format but a user can edit the source code without XML tags. The Elog language allows data to be cleaned by setting some constraints for each field. Constraints can be applied on the type of data in a field – string or number or on value of data.

The LiXto Suite [11] is a high-level leading-edge data extraction and transformation software for retrieving data from Web sources and structuring this information according to the business needs. The LiXto Suite comprises the *LiXto Visual Wrapper* and the *LiXto Transformation Server*.

- **LiXto Visual Wrapper**
  This tool offers the capability to transform Web contents into structured XML in an interactive, example-based way. The user can load HTML pages just like in a browser by entering the URL of page, then she can select regions in the page and construct correspondent elements of the resulting XML tree. This component is responsible for locating and extracting desired data from the Web. While working with LiXto Visual Wrapper, a developer generates a wrapper program which converts relevant information from HTML to XML. LiXto Visual Wrapper generates so-called XML companions for given Web pages. LiXto Visual Wrapper can also be used independently.

- **LiXto Transformation Server**
  The LiXto Transformation Server can access the World Wide Web as the largest data repository of the world and automatically integrate Web data and Web applications into enterprise application. LiXto Transformation Server enables to format, transform, merge, and deliver XML data to various devices.

  The LiXto Transformation Server is built on standard technologies such as XML, XSLT, and J2EE. The data processing model defines the following stages which are supported by the built-in software components:

  - Acquiring the required content from source locations;
  - Integrating and merging the content;
  - Transforming the content;
  - Delivering the content to various devices and applications.

3.3.3 DEByE

DEByE [45, 47, 61] (Data Extraction By Example) is a tool for extracting semi-structured data from Web sources. It is currently under development at the Database Group Laboratory, Computer Science Department, Federal University of Minas Gerais.

With DEByE, the extraction process is fully based on examples specified by the user. To assist the user with the specification of the examples, DEByE features a graphical user interface that adopts a visual paradigm based on nested tables. The examples specified by the user are then used to generate extraction patterns. Using such patterns, the DEByE

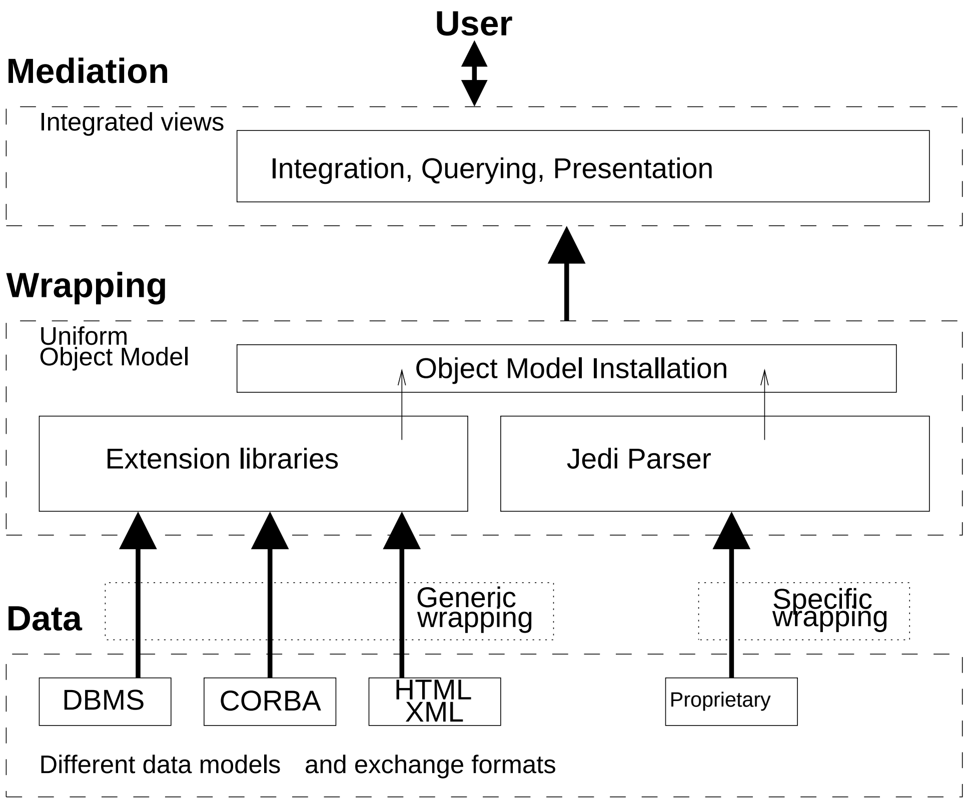

Figure 8: Architecture of Jedi

extractor module executes a flexible bottom-up extracting strategy which is very effective with various Web sources, as we have verified through experimentation. The extractor module plus an extraction pattern comprise a wrapper.

A number of companion modules were developed to be used with DEByE. Among them are QSByE, a QBE-like query interface, ASByE, a tool for generation of agents for collecting Web pages, DEByE-SM, a module for storing extracted data into a relational database, and WebView, a tool for defining and maintaining views over Web sources.

3.3.4 Jedi

Jedi [42] (Java based Extraction and Dissemination of Information) is a wrapper-generation software. Jedi has such key functions as data extraction, data integration and data reconciling. It uses attribute grammars for wrapper creation, which are evaluated with a fault-tolerant parsing strategy to cope with ambiguous grammars and irregular sources. For mediation it uses a simple generic object-oriented model and works with HTML, XML and relational data.

The architecture of Jedi is illustrated in Figure 8. The wrapping layer transforms heterogeneously modeled sources into a uniform object model. There are two kinds of wrappers involved – generic and specific ones. Generic wrappers allow Jedi to work with standard data sources like HTML, XML, CORBA etc. Specific wrappers can be used to get data from proprietary format, such as online product catalogs or bibliographies. For the construction of wrappers Jedi supports an extraction specification language based on attribute grammars, evaluated by a fault tolerant parser.

The mediation layer allows to define and query integrated views on multiple sources. Such views can relate data from multiple sources and homogenize the structure and semantics of data. The user can interact with an integrated view like with a single source. This feature of Jedi makes it very close to a DAT frameworks, but the authors of Jedi position it as a data extraction system. Unlike most DAT systems described in this document Jedi features a GUI (like most data extraction tools). Jedi has its own C-like language, named for the system – Jedi [42]. This language allows developers to describe new parsers and mediators.

3.3.5 LAPIS

LAPIS [55], which is short for "Lightweight Architecture for Processing Information Structure", is essentially a web browser with an integrated text editor, that can be programmed to process and manipulate text documents loaded within the program. The individual manual manipulation steps are contained in an executable script in a proprietary scripting language. To start the automatic data manipulation process, the script created by LAPIS is simply activated within the toolkit.

LAPIS is equipped with a pattern recognition language [56], which allows the user to specify and restrict the required data in a given document very precisely. In comparison to regular expressions, this scripting language is supposed to have a more intuitive syntax.

There are several possibilities for defining relevant data in a document. The user can manually apply the text constraints language to define a specific pattern for all of the required data in a document, or the user can highlight only a few relevant text regions and LAPIS will try to generate automatically a suitable pattern. For complex HTML documents, a combination of both methods has proven to be successful. Firstly, LAPIS tries to generate a suitable pattern, which the user can then manually adapt if necessary. The toolkit provides further assistance by testing for mistakes in developed patterns. LAPIS achieves this by separately highlighting the text areas which lie within the valid definition of a given pattern.

Furthermore, LAPIS has a built-in command shell, which allows the programmer to process Tcl script commands. This allows LAPIS to be integrated with external software.

3.3.6 Further Unstructured Data Wrappers

Several other wrapper tools and frameworks exist today, for an overview, see [46, 25]. Among the most important ones we do not discuss explicitly in this paper are the induction-based tools WIEN [44], SoftMealy [41], and STALKER [59]. These systems apply induction on a set of training examples. The system NoDoSE [5, 6] is a modeling-based tool, which means that the structure of the wrapped data must be modeled for generating a wrapper. The "Data-Extraction Tools" by the Data Extraction Research Group of the Brigham Young University [28, 29, 27, 26] require an ontology definition of the data to be wrapped, yielding more robust wrappers for the price of additional development efforts for constructing the ontology. Furthermore, the Araneus system [54] incorporates the Minerva formalism [17], which provides a grammar-based wrapper generation framework.

Several systems from the area of Natural Language Processing (NLP) are also related — they focus primarily on completely unstructured data and are in this sense limited. Among these systems are RAPIER [15], SRV [31], and WHISK [66]. Finally, some systems directly assume the underlying data to be in HTML or XML format, and exploit this knowledge, thus also limiting their input data range. Such systems are e.g. W4F [64] or RoadRunner [18, 19].

4 Feature Overview

In this section we will briefly present an overview of features in the considered systems along the taxonomy outlined in Section 2. We have gathered these informations as exhaustively as possible, but unfortunately in some cases we could not find out about the availability or unavailability of a feature in a system. For these unclear cases, we add a "?" as value.

4.1 Feature Overview for Structured Data Wrappers

A summarization of features of the structured data wrapping systems discussed in Section 3.1 is given in Table 1. For each system we have specified a row containing the system name and an account of the various features outlined in the taxonomy. The most important features are input and output data format. Concerning input formats, several database systems are supported, some systems (such as XBSQL), are rather restrictive with respect to the supported database systems.

For structured data wrappers, the output data format is often not explicit, as data is output either via an API or printed in a visual form. OfficeObjects Repository and Data Extractor are actually the only systems which can directly generate XML code, DBIWrapper has an API function for generating XML, and for the others XML generation can at most be programmed in a host language outside the tool.

None of the systems supports data cleaning, and only the (slightly dated) data integration systems Multiplex and HERMES naturally allow for performing data integration. Concerning the means of specification and available APIs, there is a large variety of languages which are supported by the systems.

4.2 Feature Overview for Semi-structured Data Wrappers

An overview of features of the systems dealing with semi-structured input data, as discussed in Section 3.2, is given in Table 2. Again, we have specified for each system a row containing the system name and an account of the various features outlined in the taxonomy.

Concerning input formats, apart from XML, the pioneering systems Infomaster and TSIMMIS support data formats, which are nowadays considered rather exotic (w.r.t. XML, which has almost become a standard for semi-structured data).

System Name	Input Data	Output Data	Data Cleaning	Data Integration	XML Generation	Specification Means	API
DBIWrapper	MySQL, PostgreSQL, ODBC	via API	No	No	Yes	Perl	Perl
Bonddb	PostgreSQL	via API	No	No	can be programmed	SQL, C++	C++
DBedit	PostgreSQL, MySQL, Oracle, text	via API	No	No	can be programmed	Perl	Perl
XBSQL	XBase	relational	No	No	No	SQL-like language	C++
Multiplex	RDBMS	relational	No	Yes	No	SQL-like language	No
HERMES	Paradox, DBASE, TextDB	relational	No	Yes	No	Mediator language	No
OfficeObjects Repository	RDBMS, OfficeObjects Repositories	relational, text	No	No	Yes	various	C
OfficeObjects Data Extractor	RDBMS	XML, HTML, CSV	No	No	Yes	visual	No

Table 1: Features of Structured Data Wrappers

Output data formats are mostly heterogeneous, only XWRAP supports direct XML output, but apart from Infomaster (for which we are not aware of an API), XML can at least be generated by a host language program via the API.

As for structured data wrappers, none of the systems supports data cleaning, and only the older systems TSIMMIS and Infomaster support some form of data integration. Also only these two support a language for wrapper specification. Apart from Infomaster, all systems feature an API.

4.3 Feature Overview for Unstructured Data Wrappers

Finally, a summary of features of unstructured data wrappers, as discussed in Section 3.2, is provided in Table 2. The table has the same structure as in the previous sections.

Here, all systems can wrap HTML and plain text sources, most (apart from DEByE) can wrap XML, and Jedi can additionally deal with traditional relational databases.

While LoPiX and LiXto are able to create XML directly, the other systems produce data in various formats, which can be considered to be less standardized. Jedi features a standard method for generating XML, though, but it must be invoked explicitly.

Concerning data cleaning, it is again supported by LoPiX and LiXto, though LiXto

System Name	Input Data	Output Data	Data Cleaning	Data Integration	XML Generation	Specification Means	API
XMLwrapp	XML	via API	No	No	can be programmed	No	C++
XWRAP	HTML, XML	XML	No	No	Yes	No	JavaAPI
Infomaster	ODBC, Z39.50, HTML	KIF	No	No	No	SQL-like query language	?
TSIMMIS	OEM,text	OEM	No	Yes	can be programmed	Language "MSL"	No

Table 2: Features of Semi-structured Data Wrappers

appears to have more explicit support for it. In Jedi, it can be programmed in Java. All systems except LAPIS can do some form of low-level data integration.

All systems provide their own wrapper specification language. Finally, LiXto, Jedi, and LAPIS provide Java APIs.

5 Property Overview

Finally, we report the main properties of the systems and methods in Table 4. For each system we report in one line its properties. The sort of the project (as a principal taxonomy category) can be one of *open source* (OS), *freeware* (FW), *freeware for research/teaching purposes* (FWS), or *commercial* (C). We then report its state, which can be *active* (A) or *closed* (C)) and whether a *theoretical background* of the system exists. Finally, we mention for convenience the main *software requirements*, and give additional *comments* on several projects.

6 Recommendations and Conclusions

Based on our analysis, we recommend using a mixture of tools, since there is no single tool which can cover the full variety of structured, semi-structured and unstructured data in a satisfactory way.

In particular, we recommend using OfficeObjects Repository in conjunction with OfficeObjects Data Extractor for wrapping structured and also for semi-structured data. The benefits of this combination are a high number of available features, in particular the possibility to generate XML-like data, the availability of source code and support via the project partner RODAN.

Furthermore, we recommend the use of LiXto for wrapping unstructured, poorly structured, and semi-structured data. The benefits of LiXto are that it features a highly sophisticated user interface, wrapping of HTML, XML, and unstructured text as well as a Java API and a strong theoretical background. In addition it even supports data cleaning

System Name	Input Data	Output Data	Data Cleaning	Data Integration	XML Generation	Specification Means	API
LoPiX	HTML, XML, text	XML	Yes	Yes	Yes	Language "XPathLog"	No
LiXto	HTML, XML, text	XML	Yes	Yes	Yes	Language "Elog"	JavaAPI
DEByE	HTML, text	structured	No	Yes	No	Language "QByE"	No
Jedi	RDBMS, XML, HTML	objects	can be programmed	Yes	Yes	Language "Jedi"	JavaAPI
LAPIS	HTML, XML, text	text	No	No	can be programmed	Language "Text Constraints", Tcl	JavaAPI

Table 3: Features of Unstructured Data Wrappers

and low-level data integration. Even if LiXto is a commercial product, the project partner TUWIEN is closely collaborating with the LiXto company, so the availability of support and if necessary also of source code is guaranteed.

In total, using the OfficeObjects tools and LiXto for implementing the DAT level of INFOMIX will allow for a highly functional and flexible state-of-the-art means of data acquisition and transformation. Since both products are well-established systems, which have already been successfully used in several projects (both commercial and scientific), they can serve as a very solid basis for the upper levels of INFOMIX. Both systems are capable of generating XML and therefore it will be easy to produce data in the intermediate data format, which will most probably be fixed to be a reduced XML or very similar language.

References

[1] DBedit. `http://www.gnacademy.org/twiki/bin/view/Gna/DBedit` and `http://www.programfiles.com/default.asp?LinkID=9989`.

[2] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[3] S. Adali, A. Brink, R. Emery, J. Lu, A. Rajput, T. Rogers, R. Ross, V.S. Subrahmanian, and C. Ward. HERMES: A Heterogeneous Reasoning and Mediator System, 1995. Available at `http://www.cs.umd.edu/projects/hermes/overview/`.

[4] S. Adali, K.S. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proceedings of the*

System Name	Sort	State	Theory	Software Requirements	Comments
DBIWrapper	OS	A	No	DBI, Perl	
Bonddb	OS	A	No	PostgreSQL, C++	
DBedit	OS	A	No	Perl	Automatic generation of HTML forms
XBSQL	OS	A	No	XBase	SQL implementation for XBase
Multiplex	?	A	Yes	No	Able to work as conflict resolution system
HERMES	?	C	Yes	Solaris, MS Windows	Conflict resolution toolkit
OfficeObjects Repository	C	A	Yes	RDBMS	
OfficeObjects Data Extractor	C	A	No	Java, DBMS	
XMLwrapp	OS	A	No	UNIX, C++	C++ API for XML
XWRAP	C	A	Yes	MS Windows	
Infomaster	C	C	Yes	No	Works with Z39.50 and ODBC
TSIMMIS	OS	C	Yes	UNIX	Requires several non-standard libraries
LoPiX	FWS	A	Yes	?	
LiXto	C	A	Yes	JRE	Works together with LiXto Wrapper and LiXto InfoPipes
DEByE	FWS	A	Yes	JRE	Tools for semi-structured data exist
Jedi	FW	A	Yes	JRE	
LAPIS	OS	A	Yes	JRE	End-user tool for extracting data

Table 4: Properties of the Systems and Methods

1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, pages 137–148, June 1996.

[5] Brad Adelberg. NoDoSE - a tool for semi-automatically extracting semi-structured data from text documents. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA*, pages 283–294, June 1998.

[6] Brad Adelberg and Matthew Denny. NoDoSE version 2.0. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*, pages 559–561, June 1999.

[7] Philipp Anokhin and Amihai Motro. Resolving inconsistencies in the multiplex multidatabase system. Technical Report ISE-TR-99-07, Department of Information and Software Engineering, George Mason University, Virginia, USA, May 1999.

[8] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Declarative information extraction, web crawling and recursive wrapping with lixto. In *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, Proceedings*, pages 21–41. Springer, September 2001.

[9] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. The elog web extraction language. In *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conference, LPAR 2001, Havana, Cuba, Proceedings*, volume 2250 of *Lecture Notes in Computer Science*, pages 548–560. Springer, December 2001.

[10] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. The lixto wrapper generation tool (extended abstract). In *Proceedings of the Ninth Italian Symposium on Advanced Database Systems (SEBD 2001), Venice, Italy*, June 2001.

[11] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy*, pages 119–128. Morgan Kaufmann, September 2001.

[12] Biblio Tech Review. Z39.50: Part 1- an overview. `http://www.biblio-tech.com/html/z39_50.html`.

[13] Angela Bonifati and Stefano Ceri. Comparative analysis of five XML query languages. *ACM Sigmod Record*, 29(1), June 2000.

[14] David Buttler, Ling Liu, and Calton Pu. A fully automated object extraction system for the world wide web. In *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS 2001), Mesa, Arizona, USA*, pages 361–370. IEEE Computer Society, April 2001.

[15] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on on Innovative Applications of Artificial Intelligence, Orlando, Florida, USA*, pages 328–334, July 1999.

[16] Sudarshan S. Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of the 100th Anniversary Meeting of the Information Processing Society of Japan*, pages 7–18, October 1994.

[17] Valter Crescenzi and Giansalvatore Mecca. Grammars have exceptions. *Information Systems*, 23(8):539–565, 1998.

[18] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Towards automatic data extraction from large web sites. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy*, pages 109–118. Morgan Kaufmann, September 2001.

[19] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. RoadRunner: Automatic data extraction from data-intensive web sites. In *SIGMOD 2002, Proceedings ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA*, June 2002.

[20] Alligator Descartes and Tim Bunce. Dbi – the perl database interface.

[21] Alligator Descartes and Tim Bunce. *Programming the Perl DBI.* O'Reilly, February 2000.

[22] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. XML-QL: A query language for XML. In *W3C Note*, August 1998.

[23] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A Query Language for XML. In *Proc. 8th International World Wide Web Conference (WWW8)*, 1999. Computer Networks 31(11-16): 1155-1169. See also `http://www.w3.org/TR/NOTE-xml-ql/`.

[24] Oliver M. Duschka and Michael R. Genesereth. Infomaster - an information integration tool. In *WS-12 Intelligent Information Integration, Workshop im Rahmen der 21. Deutschen Jahrestagung fr Künstliche Intelligenz, KI-97*, September 1997.

[25] Line Eikvil. Information extraction from world wide web — a survey. Technical Report 945, Norwegian Computing Center, Oslo, Norway, July 1999.

[26] David W. Embley, Douglas M. Campbell, Y. S. Jiang, Stephen W. Liddle, Yiu-Kai Ng, Dallan Quass, and Randy D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering*, 31(3):227–251, 1999.

[27] David W. Embley, Y. S. Jiang, and Yiu-Kai Ng. Record-boundary discovery in web documents. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, Philadelphia, Pennsylvania, USA*, pages 467–478, June 1999.

[28] David W. Embley, Yiu-Kai Ng, and Li Xu. Recognizing ontology-applicable multiple-record web documents. In *Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, Yokohama, Japan, Proceedings*, volume 2224 of *Lecture Notes in Computer Science*, pages 555–570. Springer, November 2001.

[29] David W. Embley and Li Xu. Record location and reconfiguration in unstructured multiple-record web documents. In *The World Wide Web and Databases, Third International Workshop WebDB 2000, Dallas, Texas, USA, Selected Papers*, volume 1997 of *Lecture Notes in Computer Science*, pages 256–274. Springer, 2000.

[30] Mary Fernandez, Jérôme Siméon, Philip Wadler, Sophie Cluet, Alin Deutsch, Daniela Florescu, Alon Levy, David Maier, Jason McHugh, Jonathan Robie, Dan Suciu, and Jennifer Widom. XML query languages: Experiences and exemplars. Available from `http://www-db.research.bell-labs.com/user/simeon/xquery.html`, 1999.

[31] Dayne Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000.

[32] Jürgen Frohn, Rainer Himmeroder, Paul-Thomas Kandzia, Georg Lausen, and Christian Schlepphorst. FLORID: A prototype for F-Logic. In *Proceedings of the Thirteenth International Conference on Data Engineering, Birmingham, U.K.*, page 583, April 1997.

[33] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In *Proc. AAAI Symp. on Information Gathering*, pages 61–64, 1995.

[34] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *J. of Intelligent Information Systems*, 8(2):117–132, 1997.

[35] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA*, volume 26(2) of *SIGMOD Record*, pages 539–542, New York, May 1997. ACM Press.

[36] Roy Goldman, Jason McHugh, and Jennifer Widom. From semistructured data to XML: Migrating the lore data model and query language. In *ACM SIGMOD Workshop on The Web and Databases (WebDB'99), Philadelphia, Pennsylvania, USA*, pages 25–30, June 1999.

[37] Joachim Hammer, Hector Garcia-Molina, Svetlozar Nestorov, Ramana Yerneni, Markus M. Breunig, and Vasilis Vassalos. Template-based wrappers in the TSIMMIS system. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA*, pages 532–535, June 1997.

[38] Joachim Hammer, Jason McHugh, and Hector Garcia-Molina. Semistructured data: The TSIMMIS experience. In *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), St.-Petersburg, Russia*, pages 1–8, September 1997.

[39] Andru Hill. Bond — building object network databases. `http://bond.treshna.com/`.

[40] Andru Hill. Bonddb — object oriented database abstraction layer. `http://bonddb.treshna.com/`.

[41] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[42] Gerald Huck, Peter Fankhauser, Karl Aberer, and Erich J. Neuhold. Jedi: Extracting and synthesizing information from the web. In *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, New York City, New York, USA*, pages 32–43. IEEE Computer Society, August 1998.

[43] Peter Jones. Xmlwrapp. `http://pmade.org/pjones/software/xmlwrapp`.

[44] Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1–2):15–68, April 2000.

[45] Alberto H. F. Laender. DEByE – data extraction by example. `http://abrolhos.lbd.dcc.ufmg.br/~debye/`.

[46] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, Altigran S. da Silva, and Juliana S. Teixeira. A brief survey of web data extraction tools. *ACM Sigmod Record*, 31(2), June 2002.

[47] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, and Altigran Soares da Silva. DEByE — data extraction by example. *Data & Knowledge Engineering*, 40(2):121–154, 2002.

[48] Ling Liu, Calton Pu, and Wei Han. XWRAP: An XML-enabled wrapper construction system for web information sources. In *Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA*, pages 611–621. IEEE Computer Society, March 2000.

[49] Ling Liu, Calton Pu, Wei Han, David Buttler, and Wei Tang. Building an extensible wrapper repository system: A metadata approach. In *Proceedings of the Third IEEE META-DATA Conference*, April 1999.

[50] Bertram Ludäscher, Rainer Himmeröder, Georg Lausen, Wolfgang May, and Christian Schlepphorst. Managing semistructured data with FLORID: A deductive object-oriented perspective. *Information Systems*, 23(8):589–613, 1998.

[51] Wolfgang May. LoPiX: A system for XML data integration and manipulation. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, Roma, Italy*, pages 707–708, September 2001.

[52] Wolfgang May. XPath-logic and XPathLog: A logic-based approach for declarative XML data manipulation. Technical Report 149, Institut für Informatik, Universität Freiburg, Germany, February 2001.

[53] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A Database Management System for Semistructured Data. *SIGMOD Record*, 26(3):54–66, 1997.

[54] Giansalvatore Mecca, Paolo Atzeni, Alessandro Masci, Paolo Merialdo, and Giuseppe Sindoni. The araneus web-base management system. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA*, pages 544–546, June 1998.

[55] Robert C. Miller and Brad A. Myers. LAPIS: Smart editing with text structure, April 2002.

[56] Robert C. Miller and Brad C. Myers. Lightweight structured text processing. In *Proceedings of the 1999 USENIX Annual Technical Conference, Monterey, California, USA*, pages 131–144, June 1999.

[57] Amihai Motro. Multiplex: A formal model for multidatabases and its implementation. Technical Report ISE-TR-95-103, Department of Information and Software Engineering, George Mason University, Virginia, USA, March 1995.

[58] Amihai Motro. Multiplex: A formal model for multidatabases and its implementation. In *Next Generation Information Technologies and Systems, 4th International Workshop, NGITS'99, Zikhron-Yaakov, Israel, Proceedings*, volume 1649 of *Lecture Notes in Computer Science*, pages 138–158. Springer, July 1999.

[59] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[60] James A. Pattie. DBIWrapper. `http://dbiwrapper.sourceforge.net/`.

[61] Berthier A. Ribeiro-Neto, Alberto H. F. Laender, and Altigran Soares da Silva. Extracting semi-structured data through examples. In *Proceedings of the 1999 ACM CIKM International Conference on Information and Knowledge Management, Kansas City, Missouri, USA*, pages 94–101. ACM, November 1999.

[62] Mike Richardson. Xbsql: An sql wrapper for xbase library. `http://www.quaking.demon.co.uk/xbsql.html`.

[63] Jonathan Robie, Joe Lapp, and David Schach. XML query language (XQL). In *QL '98 – The Query Languages Workshop, Boston, Massachussets, USA*, December 1998. `http://www.w3.org/TandS/QL/QL98/`.

[64] Arnaud Sahuguet and Fabien Azavant. Building intelligent web applications using lightweight wrappers. *Data & Knowledge Engineering*, 36(3):283–316, March 2001.

[65] David Schach, Joe Lapp, and Jonathan Robie. Querying and transforming XML. In *QL '98 – The Query Languages Workshop, Boston, Massachussets, USA*, December 1998. `http://www.w3.org/TandS/QL/QL98/`.

[66] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1–3):233–272, 1999.

[67] American National Standard. Knowledge interchange format. `http://logic.stanford.edu/kif/dpans.html`.

[68] Ross Tredwell. Toolkits for generating wrappers. Seminar paper at the University of Mannheim, April 2002.

[69] Ross Tredwell and Stefan Kuhlins. Wrapper generating tools, 2002. Published on `http://www.wifo.uni-mannheim.de/~kuhlins/wrappertools/`.