Project Number:	IST-2001-33570
Project Acronym:	INFOMIX
Project Full Name:	Infomix: Boosting the Information Integration
Deliverable Number:	D6.1
Title:	**Heterogeneous data source type description**
Workpackage:	WP6
Document Type:	Deliverable
Distribution:	Infomix Consortium
Status:	Final
Document file:	WP6 T1 D6.1 0102.pdf
Version:	1.2
Number of pages:	19
Due Date:	July 31, 2003
Delivery Date:	August 20, 2003
Partners contributed:	TUWIEN
Partners owning:	TUWIEN
Man Months:	6

Short Description:

The purpose of this document is to specify the different heterogeneous data source types which should be handled by the INFOMIX prototype system at the DAT layer. It also specifies a homogenized data format at the source level, the INFOMIX Source Data Format (ISDF), to which data in the heterogeneous data source formats are converted. This document serves as a basis for a forthcoming report on methods for data acquisition and transformation and is furthermore relevant for the implementation of the INFOMIX prototype.

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	June 24, 2003	First draft
v.1.1	August 05, 2003	Second draft after meeting RODAN
v.1.2	August 18, 2003	Final version

1 Introduction

The INFOMIX project is aimed at exploiting advanced reasoning capabilities in order to provide a set of techniques and tools for next-generation information integration. An important aspect of this problem is that data and information is in general not available in a homogeneous format, but rather in different formats, which range from structured formats such as relational data or objection-oriented data formats, over semi-structured formats like XML to completely unstructured data in the form plain text. The INFOMIX information integration system shall be able to deal with different data formats and support their logical integration. To this end, the INFOMIX system (D2.2) has been equipped with a conceptually layered architecture, in which the lowest level, the Data Acquisition and Transformation Layer (DAT Layer), provides "low-level" access to the data sources which should be integrated (see Figure 1).

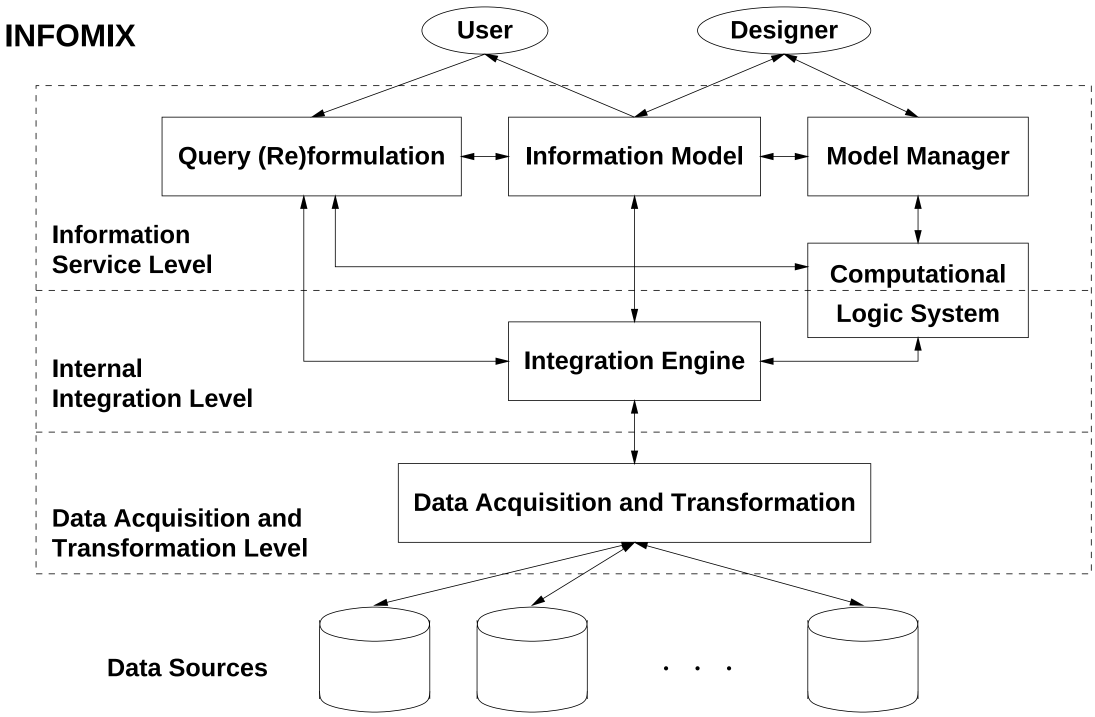

Figure 1: Conceptual Layers of the INFOMIX information integration system

Objectives. The aim of this document is to specify the different heterogeneous data source types which should be handled by the INFOMIX prototype system at the DAT layer. It also specifies the homogenized data format at the source level, the INFOMIX Source Data Format (ISDF), to which data in the heterogeneous data source formats are converted.

Relations to other documents. This report is directly related to the documents D1.3 (Review of Techniques and Systems for Acquisition and Transformation of Heterogenous

Data), D2.1 (Functional Specification of the INFOMIX System), D2.2 (INFOMIX System Architecture), and to the forthcoming report D6.2 (Methods for Data Acquisition and Transformation). It is furthermore relevant for the implementation of the INFOMIX prototype, in particular for the Information Integration Management Layer prototype (Deliverable 7.1) and for the Data Acquisition Layer prototype (Deliverable 7.3).

Intended audience and usage guidelines. This report is accessible to the public.

Notational conventions. Nothing particular.

2 Preliminaries

In this section, we briefly recall some conclusions and findings from previous documents of the INFOMIX project which are relevant to this document.

2.1 DAT functionality and architecture

The functionality of the DAT Layer, and a rough architecture, has been described in the INFOMIX system architecture (D2.1), in which the Design-Time and the Run-Time architecture of the system has been described, shown in the Figures 2 and 3.

The DAT Layer is, since it is a conceptual layer, not explicitly outlined in this functional system architecture. The Design-Time part, however, comprises the *Wrapper Generation Interface* and the *Wrapper Generator*, while the Run-Time part comprises the *Wrapper Executor*. The functionalities of these components have been amply described in report D2.1, and we do not repeat it here.

Basically, the Wrapper Executor has to supply the data from the (external) heterogeneous data source for the internal integration in a homogenized data format. Other optional standard services such as a possibility to browse the data sources (i.e., viewing the result of wrapper execution), which have not been explicitly incorporated in the INFOMIX architecture, are added value features but not of primary interest. The Wrapper Generator Interface allows to incorporate data sources into the system, by specifying the necessary pieces of information, from which then individual Wrappers for the data sources are generated. It is envisaged to provide (semi-)automatic support for this task besides mere manual integration, depending on the type of data source and the information needed.

2.2 DAT tools

In document D1.3, we have provided a review of techniques and systems for acquisition and transformation of heterogenous data from the perspective of the envisaged DAT layer of the INFOMIX system. In it, the data from the (physical) sources should be mapped into a logical (virtual) format which amounts to a restricted fragment of XML. From the wide range of DAT tools which are available, the report D1.3 concluded that, for

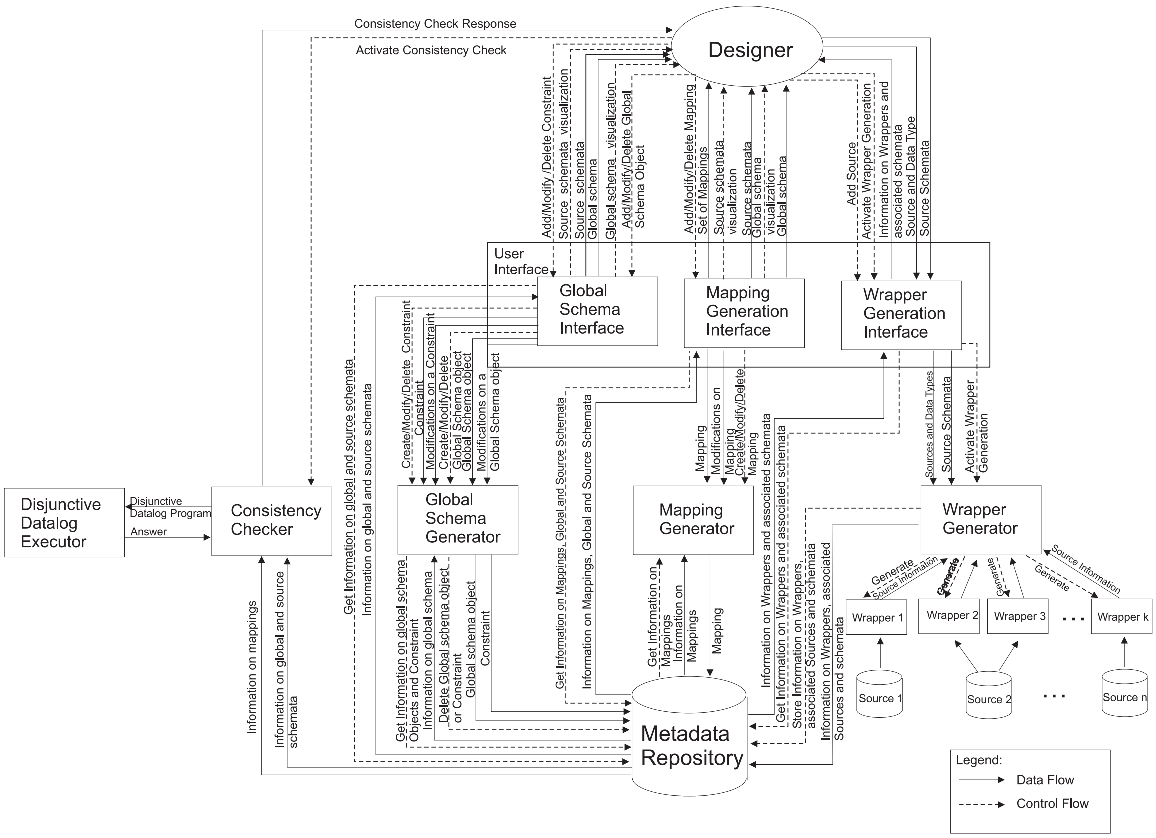

Figure 2: Design-Time Architecture of the INFOMIX system

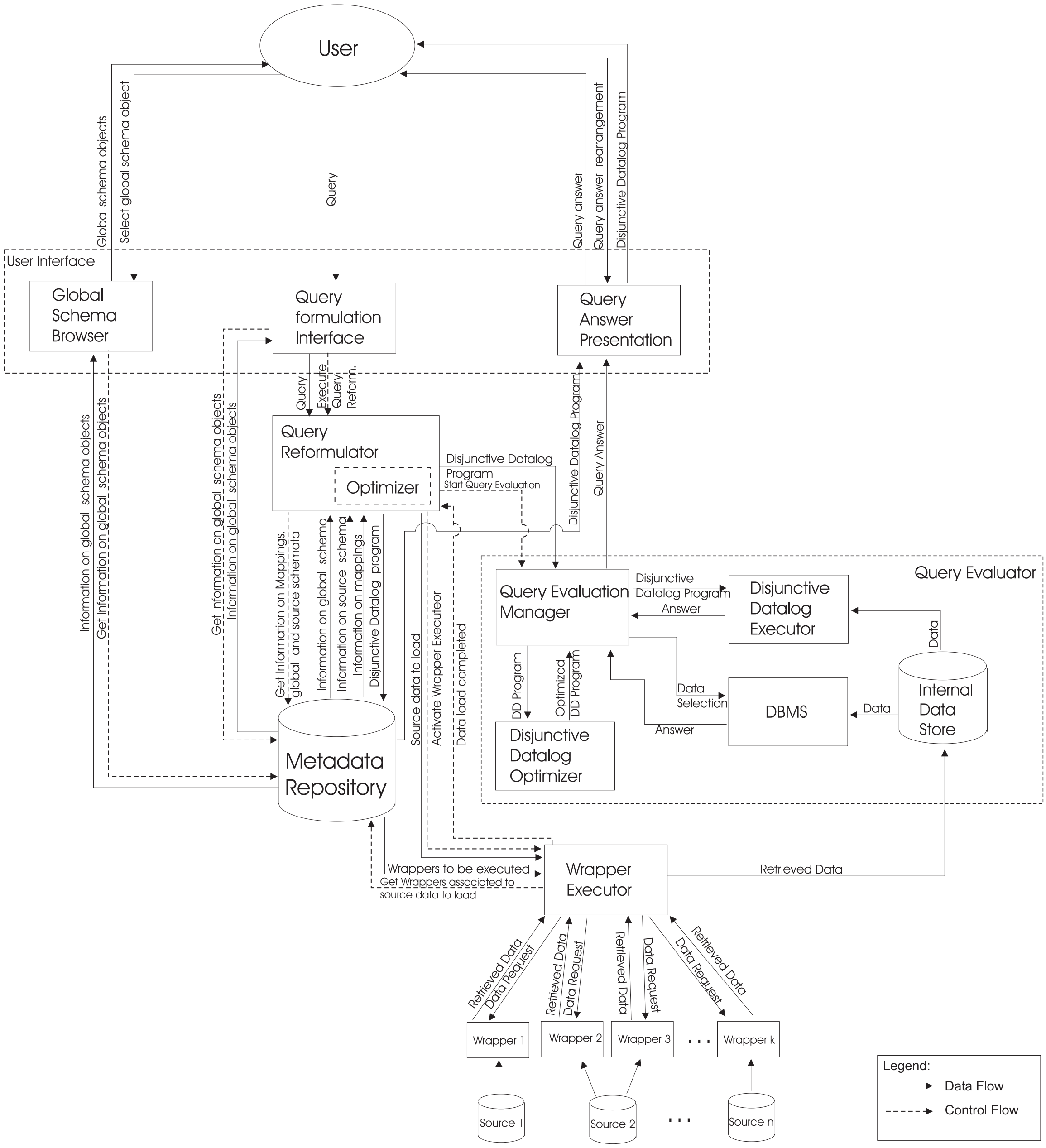

Figure 3: Architecture of INFOMIX system - Run-Time part

various reasons, the following two tools should be find application in the development of the INFOMIX system prototype:

1. The OfficeObjects Portal, in particular the *OfficeObjects Repository* together with *OfficeObjects Data Extractor*, of RODAN, for wrapping structured and also semistructured data; and

2. LiXto Suite, comprising *LiXto Visual Wrapper* and the *LiXto Transformation Server*, of LiXto GmbH for wrapping unstructured, poorly structured, and semistructured data.

3 DAT-relevant data formats in INFOMIX

The DAT layer of INFOMIX is concerned with different data formats, which correspond to the data flow in the INFOMIX system (see Figure 4).

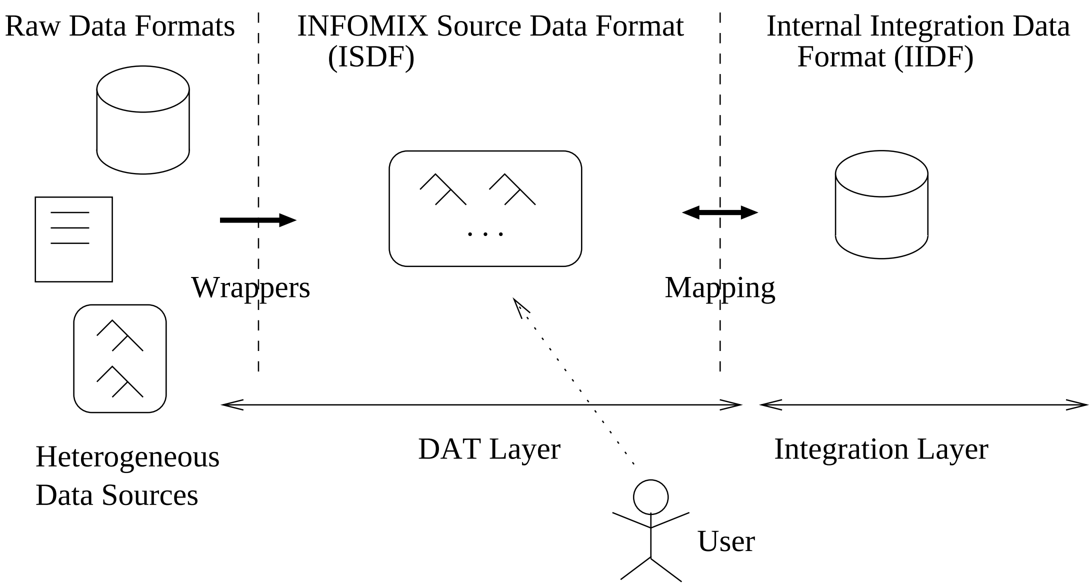

Figure 4: DAT-Relevant Data Formats

Raw Data Formats: At the input side to the system are the source data in their individual formats. We refer to all these formats as *raw data formats*. The raw data formats will be considered in the Section 4.

INFOMIX Source Data Format (ISDF): The heterogeneous raw data are transformed to the homogenous INFOMIX Source Data Format (ISDF), by means of wrappers as specified in the INFOMIX architecture. The ISDF provides a (logical) view of the wrapped data which is accessible to the user, and is the uniform logical format of source data for defining mappings between the global data format at the user side, and the local data format at the (wrapped) sources.

Towards dealing with semi-structured data at the source level, and in order to make the wrapped data also easily accessible to other tools and systems, the ISDF is designed as a (rather plain) fragment of XML Schema which comprises in essence complex value data that are relational in the core. The ISDF will be considered in Section 5 below.

Internal Integration Data Format (IIDF) The source data need to be transformed into an internal format which is used at the internal integration level. The IIFDF, which is not subject of this report, will be considered in the forthcoming documents D4.2 (Algorithms and Implementation Techniques for the Internal Integration Formalisms) and D6.2 (Methods for Data Acquisition and Transformation). However, by the selection of the Internal Integration Formalism (D4.1) the IIDF will be based on the relational data format.

4 Raw-data formats

In this section, we specify in detail the raw data formats of possible sources in order to be handled by the INFOMIX system. We roughly classify the raw data into:

- relational data,
- object-oriented data,
- XML data, and
- HTML data.

However, this classification is not strict, e.g., we also allow for non-HTML data that can be embedded into HTML, such as comma separated values, Microsoft Excel data, etc., but classify them as HTML data. For each class, we explicitly specify the concrete format that is accepted by the INFOMIX system and list exceptional cases if any.

4.1 Relational format

For several reasons, relational data plays a predominant role in INFOMIX. Historically, also because the relational data model, developed in the early seventies, is the oldest data format among the formats considered here, the first information integration systems considered the task of integrating relational databases. In spite of its long tradition – compared to the integration of heterogeneous data from the World Wide Web (WWW), which is still in its infancy – the integration of relational data still faces some problems that have not been addressed conveniently (which was a major motivation for the INFOMIX project; cf. deliverable D1.1 for more details).

Moreover, relational database management systems (RDBMSs) have been commercially very successful. They are a quasi-standard in commercial and industrial data management (which is not yet true for applications based on Web data such as, e.g. XML). The success of the relational data model has led to a plethora of RDBMSs for different platforms and each system uses its own proprietary mechanism for data storage. Thus, the raw data format for relational data depends on the RDBMS used for data storage and management. Due to the importance of the relational format, however, the goal for any information integration system is to support as many RDBMSs as possible.

In order to fulfill this requirement we resort to the Open Database Connectivity (ODBC) standard. ODBC technology provides a common interface for accessing heterogeneous relational databases. ODBC is based on the Structured Query Language (SQL) [5, 4] as a standard for accessing data. By means of this interface a single application can access different SQL DBMSs through a common set of code. This enables application developers to build applications without targeting a specific DBMS. ODBC database drivers are then used to link an application to a particular DBMS.

ODBC features are:

- Applications are not tied to a proprietary vendor API.
- SQL statements can be explicitly included in source code or constructed on the fly at run time.
- An application can ignore the underlying data communication protocols.
- Relational data can be sent and received in a format that is convenient to the application.
- ODBC is designed in conjunction with the international ISO Call-Level Interface standard.

While initially developed for the Microsoft Windows platform, today ODBC implementations for UNIX and LINUX platforms are available, as well. Moreover, there exist ODBC database drivers for the most popular database such as Clipper, dBase, FoxPro, Informix, Ingres, Microsoft Access, Microsoft MSQL Server, MySQL, Oracle, and Postgres, just to mention a few.

INFOMIX support. The INFOMIX system supports any RDBMS for which an ODBC driver exists and that operates on a platform for which an ODBC implementation exists.

Limitations and Exceptions. None.

However, we remark that data wrapping will be a unidirectional process, i.e. only querying facilities will be needed; no updates of the underlying databases will take place.

4.2 Object-oriented format

The object-oriented data model is a more recent database model. It is based on the concept of objects, which are a collection of data items and the operations which can be executed on them. Researchers became interested in it in the late seventies and early eighties as the concept of object-oriented systems started to appear. Object-oriented databases evolved out of the need for system maintainability and the desire for persistence of data: The first databases of this type were created for storing and sharing objects.

When database capabilities are integrated with object programming language capabilities, the result is an object database management system (ODBMS). An ODBMS makes database objects appear as programming language objects in one or more object-oriented programming languages. This enables application developers to use the same language to describe databases as they use for programming. However, usually an ODBMS extends the object-oriented programming language with transparently persistent data, concurrency control, data recovery, associative queries, and other capabilities.

One goal of the research in object-oriented databases also was to overcome restrictions imposed by the relational model on certain types of data. However, these limitations started to become blurred as relational systems started to add object-oriented functions and interfaces leading to a hybrid object-relational model. Commercial ODBMSs have not been by far as successful as RDBMSs and still are a specialized field. However, they still are expected to continue gaining market acceptance.

Unlike the relational model, for the object-oriented model no high level query language like SQL emerged as a quasi-standard, and thus, a high degree of low level control over an ODBMS is usually left to the programmer. On the other hand, the ability to alter low level methods used in an ODBMS hampered the development of standards concerning object-oriented data models and query languages. The most well-known standardization effort for object-oriented databases and object-relational databases is the ODMG standard.

The ODMG standard has been developed by the Object Data Management Group (ODMG) [1], a consortium of vendors and interested parties that work on specifications for object-oriented databases and object-relational mapping products. It is a portability specification, designed to allow for portable applications that can run on more than one specific ODBMS.

Components of the ODMG specification are [1]:

- Object Model: The common data model supported by ODMG implementations. It is based on the OMG Object Model, designed – besides further objectives – to be a common denominator for ODBMSs and object programming languages.

- Specification Languages: Two specification languages are defined, namely the Object Definition Language (ODL) and the Object Interchange Format (OIF) language. ODL (based on the OMG Interface Definition Language IDL) is a specification language to define object types conformant to the ODMG Object Model. OIF is a specification language used to store and load to/from a file or set of files.

- Query Language: The Object Query Language (OQL) is a declarative language for querying and updating objects. SQL was used as the basis for OQL.

- Language Bindings: Bindings between the ODMG Object Model (ODL and OML) and the C++, Smalltalk, and Java programming languages are defined for ODMG implementations. The language bindings also include a mechanism to invoke OQL and procedures for operations and transactions.

Concerning the OQL query language, we remark that, unlike SQL, OQL is to some degree tailored to the specific programming languages, in order to provide an integrated environment for programming and data manipulation.

The ODMG group completed its work on the ODMG standard in 2001 and was disbanded. The final release of the standard is ODMG 3.0, including the components described above. However, the ODMG Java binding has been superceded by Java Data Objects (JDO). A reference implementation of JDO has recently been developed as part of the Java Community Process. Despite the fact that the ODMG Java binding served as a basis for JDO, the latter lacks the support for OQL invocation and implements its own query language (called JDOQL). The latest implementation of the ODMG Java binding implements version 2.0 of the standard.

INFOMIX support. The INFOMIX system supports any ODBMS that conforms to the ODMG standard and which is accessible via the ODMG 2.0 Java binding. In particular, the ODBMS must be accessible by OQL queries (more specifically, by the subset of OQL supported by the ODMG Java binding).

Limitations and Exceptions. Although usually supported by OODBMSs, multimedia data types are not supported by the INFOMIX system and explicitly excluded.

Again, only querying facilities will be used for wrapping and no updates will take place. Furthermore, we remark that, in general, the need to translate the object-oriented data to fit the INFOMIX source data format defined in Section 5 (which is part of the wrapper generation process and will be dealt with in deliverable D6.2) may incur implicit limitations. Thus, (semi-)automatic support for wrapper generation might only be available for object-oriented data conforming to a restricted data model.

4.3 XML

The significant boom of the World Wide Web (WWW) made a vast amount of information available which is usually subsumed by the term *Web data*. While still most of these data is stored (and transmitted) in the Hypertext Markup Language (HTML) [8, 6], it has early been recognized that this simple format – tailored for document formatting and linking – faces several limitations with respect to data modeling and processing.

It was for these reasons that, in 1996, the World Wide Web Consortium (W3C) assigned a Working group to design a new standard, the Extensible Markup Language (XML) [7]. XML is a markup language for document description that mainly differs from HTML by the fact that XML is a subset of the Standard Generalized Markup Language (SGML) [3], while HTML is an application thereof, i.e. in HTML the allowed markup (allowed tags) is fixed. Hence, in contrast to HTML, XML exhibits the following properties [2]:

- Extensibility: Information providers can define new tag and attribute names at will.
- Structure: Document structures can be nested to any level of complexity.
- Validation: Any XML document can contain an optional description of its grammar for use by applications that need to perform structural validation.

Today, XML is largely accepted by the industry as a standard for representing and exchanging data on the Web. XML documents are modular, consisting of several distinct components which may be distributed on various file systems. The main parts of an XML document are an XML declaration, an optional document type or schema declaration, and the document instance, i.e., text and markup representing data.

In the following, we call an XML document without type or schema declaration *pure XML data* (sometimes also semi-structured XML data, since XML data is usually considered to be semi-structured), while an XML document including a type or schema declaration is termed *structured XML data*. Furthermore, an XML document is called *well-formed* if it (syntactically) conforms to the XML standard, and it is called *valid* if it additionally conforms to its document type or schema definition.

A document type definition (DTD), which is part of the XML language, is intended to specify a grammar for an XML document. In particular a DTD is a context-free grammar and thus can be recursive. To some extent DTDs can be used to define a schema for the data represented by the XML document, however, there are several limitations of DTDs as a schema formalism (see also [2]):

- There is no notion of atomic types,
- DTDs impose order,
- types assigned to elements are global (no namespaces), and
- there is no typing for references.

Moreover, DTDs are not XML documents themselves. These limitations have early been recognized and led to the proposal of other formalisms for specifying the schema of XML data. In 2001, the W3C adopted the XML Schema standard [9, 10, 11] for this purpose.

XML Schema is a schema description in XML that can replace a document type definition. XML Schema allows for namespaces and features a number of built-in simple and complex types (including types for ordered and unordered collections of data). Furthermore, XML Schema allows to define one's own types, the derivation of a type, the definition of abstract types, and the definition of uniqueness constraints and corresponding reference constraints. This makes XML Schema superior to DTDs with respect to modeling the structure of a set of XML data.

INFOMIX support. The INFOMIX system supports pure (semi-structured) XML data (according to the XML 1.0 standard) as well as XML data structured according to a corresponding XML Schema specification.

Note, however, that it is not the responsibility of the INFOMIX system to check the validity of structured XML data.

Limitations and Exceptions. The INFOMIX system will not take DTD specifications into account, i.e., any DTD is neglected and the corresponding XML data is considered to be pure XML data.

Furthermore, we remark that the need to translate semi-structured XML data to fit the INFOMIX source data format (see Section 5) may limit the support for (semi-)automatic wrapper generation to appropriately structured XML data. Note however, that wrappers for pure XML data can be obtained applying the same techniques as for HTML data (i.e., wrapper generation using LiXto), but without graphical support.

4.4 HTML

By far, the majority of Web data available today is stored in HTML format [8, 6]. Since initially conceived to format and link free text, HTML data is still considered poorly structured or even unstructured data, although methods have been developed to embed various types of other data (also structured data) into HTML Web pages (e.g. data tables stored as comma separated values).

Nowadays, almost any data can be made available on the Web by embedding it into HTML files. These data includes textual documents and graphics in various formats (e.g. postcript, PDF, Microsoft Word, etc.) as well as images, sound files, films, and animations. Data of the latter categories are often termed *multi-media data* and their integration is a subject of research on its own right, hence, out of the scope of INFOMIX.

While the rendering of HTML data is human readable, its poorly structured nature makes it difficult to process these data automatically and to single out the structure and content of such data. However, the importance and the large amount of available Web data makes it indispensable for an advanced information integration system such as INFOMIX to also deal with these data. For an application to read Web data as well, it must apply particular wrappers to extract structured data from the Web. Compared to wrappers for structured data, it is much harder to automate the development of these software components and human interaction is usually required.

INFOMIX support. The INFOMIX system supports syntactically valid Web pages conforming to the HTML 2.0, 3.2, or 4.1 standards (the latter forthcoming; as far as it is supported by standard Web browsers, i.e. by the de-facto-standard). Note, however, that it is not the responsibility of the INFOMIX system to check the validity of HTML data.

As for embedded data formats, the INFOMIX system allows for comma separated values (CSV), Microsoft Word (planned), Excel and Powerpoint (planned) documents, as well as PDF documents. We stress, that only the HTML rendered version of these documents is supported rather than processing them in their raw formats.

Limitations and Exceptions. The INFOMIX system will not handle HTML embedded data formats except those explicitly cited above. In particular, INFOMIX will not support multi-media data such as image files, sound files, or movie data.

Note, that like the free text of a Web page, any embedded data is considered unstructured in the sense, that no (semi-)automatic support for wrapper generation will be available. Nevertheless, the user (system administrator) will have the possibility of graphical assistance in wrapper generation by means of the LiXto system.

5 INFOMIX source data format

The objectives for the INFOMIX source data format (ISDF) are the following:

- Use a complex value data format, that allows for multi-valued attributes and non-recursive type definitions.
- Use an XML-like syntax for the ISDF.

Since XML Schema definitions are capable of expressing the required complex value formats and since they are XML instances themselves and the corresponding data format is a subset of XML. We use (a subset of) XML Schema for specifying the ISDF, i.e. the logical view of the INFOMIX source data. Any XML data conforming to an XML Schema definition that is obtained satisfying the requirements subsequently defined is considered a valid representation of INFOMIX source data.

Essentially, the ISDF amounts to a small extension of the relational format, by allowing nesting of data and multiple attribute values. The handling of these data, and access to source data in mapping definitions by the query language, will however remain relational in its core.

5.1 Basic data types.

The complex value format allows for basic values and for complex values. As for basic data types we only allow for the following XML Schema built-in simple types:

- `xsd:string`,
- `xsd:integer`,
- `xsd:decimal`,
- `xsd:boolean`, and
- `xsd:anySimpleType`.

No user-defined simple data types (obtained by derivation) are allowed.

Example 5.1 The ISDF for an entry in a simple relational schema of a product table consisting of the attributes

- `productID` of type integer for a product identifier,
- `productName` of type string for the name of a product,
- `price` of type real for the price of a product, and

- `available` of type boolean for the availability of a product,

can be represented by the following XML Schema specification:

```
<xsd:complexType name="Product">
  <xsd:attribute name="productID" type="xsd:integer" use="required"/>
  <xsd:attribute name="productName" type="xsd:string" use="required"/>
  <xsd:attribute name="price" type="xsd:decimal" use="required"/>
  <xsd:attribute name="available" type="xsd:boolean" use="required"/>
</xsd:complexType>
```

A tuple $\langle 4711, \mathit{Perfume}, 21.90, \mathit{true}\rangle$ of the product table would then be represented in the ISDF as

```
<Product productId=4711 productName="Perfume" price=21.90
         available=true>
```

5.2 Complex data types

The type of complex values is defined via XML Schema complex type definitions (using the `xsd:complexType` tag as already seen in the previous example). The elements constituting the complex value and their corresponding types are specified either via `xsd:element` tags or by `xsd:attribute` tags, as defined in XML Schema.

However, we do not allow for the definition of quasi user-defined basic types using the `xsd:simpleContent` tag. Furthermore, empty content elements without attributes are not allowed. But, element and attribute model groups (i.e. `elementGroup` and `attributeGroup` tags) may be used.

As for the schema attributes of attribute tags, the following applies: Attribute tags always need to have the optional `use` attribute set to `required`. Additional `fixed` or `default` attributes together with a corresponding `value` attribute are allowed.

In general, complex values are not multi-valued and do not contain null values. Thus, the elements of a complex type need to be defined using the `xsd:all` model group and the `minoccurs` and `maxoccurs` schema attributes need not be set (since they default to 1). If `minoccurs` is explicitly set, then it must be set to 1. Note that in the case of the `xsd:all` model group, `maxoccurs` can only be set to 1.

Example 5.2 A soccer team could be represented as a complex structure having a name and coach, where values for coaches are complex values themselves consisting of a first name and a last name. A corresponding XML schema for ISDF specification may look like follows:

```
<xsd:complexType name="SoccerTeam">
  <xsd:all>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="Coach">
      <xsd:complexType>
        <xsd:attribute name="FirstName" type="xsd:string"
```

```
                    use="required"/>
         <xsd:attribute name="LastName" type="xsd:string"
                    use="required"/>
       </xsd:complexType>
     </xsd:element>
   </xsd:all>
 </xsd:complexType>
```

A corresponding data item in ISDF for Bayern Munich would be:

```
<SoccerTeam>
  <Name>Bayern Munich</Name>
  <Coach FirstName="Ottmar" LastName="Hitzfeld"/>
</SoccerTeam>
```

5.3 Multi-valued attributes

We also allow for multi-values, i.e. a complex data type may be defined allowing for several values for some of its elements. To this end, the elements of the complex type need to be specified in a `xsd:sequence` model group (note that attributes cannot occur more than once). Still `minoccurs` has to be 1 if explicitly set, but `maxoccurs` may be unbounded in this case.

Note that `xsd:sequence` is the default model group for complex type definitions in XML schema and, hence, the `xsd:sequence` tag may be omitted.

Example 5.3 If, in the previous example, we include all the players of a team into its representation then we need multiple values for the attribute player, resulting in:

```
<xsd:complexType name="SoccerTeam">
  <xsd:sequence>
    <xsd:element name="Name" type="xsd:string"/>
    <xsd:element name="Coach">
      ...
    </xsd:element>
    <xsd:element name="Player" maxoccurs="unbounded">
      ...
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

Note that this structure for a soccer team allows a team to consist of an unlimited number of players (but at least one) and exactly one coach:

```
<SoccerTeam>
  <Name>Bayern Munich</Name>
  <Coach  FirstName="Ottmar" LastName="Hitzfeld"/>
  <Player FirstName="Oliver" LastName="Kahn"/>
```

```
  <Player FirstName="Giovanne" LastName="Elber"/>
  ...
</SoccerTeam>
```

Note that the `xsd:sequence` model group forces the elements to be ordered. However, later processing, i.e. querying will not rely on this order.

5.4 Miscellaneous

As a convenience for schema definition, non-local type declarations are allowed, i.e. complex type definitions according the above rules may be defined globally. However, global type definitions must not be cyclic.

Example 5.4 In our previous example, we could equivalently specify a global data type for a person:

```
<xsd:complexType name="Person">
  <xsd:attribute name="FirstName" type="xsd:string" use="required"/>
  <xsd:attribute name="LastName" type="xsd:string" use="required"/>
</xsd:complexType>
```

and then use the global type `Person` as the data type for coaches and players:

```
<xsd:complexType name="SoccerTeam">
  ...
    <xsd:element name="Coach" type="Person"/>
    <xsd:element name="Player" type="Person"/>
  ...
</xsd:complexType>
```

Currently, we do not permit the definition of complex types by means of derivation. This could be relaxed (as a convenience for schema specification) later on, if needed, in the case of schema extension and for certain cases of schema restriction.

Features of XML Schema definitions not explicitly mentioned in this section are not allowed. In particular choices, mixed content, integrity constraints (`unique`, `key`, and `keyref` conditions), `any` model groups, and abstract types are not permitted.

6 Conclusion

This report is the first of two reports aiming at the definition and description of a framework for engineering the "low level" data access to data of different data sources, which will be used for the design and implementation of the data acquisition and transformation layer of the INFOMIX system. In this first document, the basis for such a framework is developed by

(*i*) specifying the raw data formats for the heterogeneous sources supported by the INFOMIX system and by (*ii*) specifying a homogenized data format at the source level, the INFOMIX source data format, to which the heterogeneous source data formats are converted.

The raw data formats supported by the INFOMIX system can be roughly summarized as relational data with ODBC interface, object-oriented data according to the ODMG standard, XML documents and HTML documents.

The ISDF can be shortly characterized as a complex value data format allowing for multi-valued attributes.

Building on this, a subsequent deliverable will define methods for data acquisition and transformation, including methods for wrapper generation, methods for mapping the internal source data format to the internal integration data format, and suitable caching techniques.

Furthermore, a query language will have to be defined for the ISDF. In view of the ISDF specification of this document, using an XML query language and restricting it to our needs seems to be a reasonable approach. A further requirement on the source query language will be the ease of rewriting queries over the internal integration data format. In this regard, an SQL-like syntax of the query language might be of help.

Acronyms

1. DAT = Data Acquisition and Transformation.
2. DBMS = Database Management System.
3. DTD = Data Type Definition.
4. HTML = Hypertext Markup Language.
5. IIDF = Internal Integration Data Format.
6. ISDF = INFOMIX Source Data Format.
7. ODBC = Open Database Connectivity.
8. ODBMS = Object DBMS.
9. ODMG = Object Data Management Group.
10. OQL = Object Query Language.
11. RDBMS = Relational DBMS.
12. SGML = Standard Generalized Markup Language.
13. SQL = Structured Query Language.
14. W3C = World Wide Web Consortium.
15. WP = Workpackage.

16. WWW = World Wide Web.

17. XML = Extensible Markup Language.

References

[1] The Object Data Management Group. `http://www.odmg.org/`.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, Los Altos, CA 94022, USA, 2000.

[3] Joint Technical Committee ISO/IEC (JTC 1). ISO/IEC 8879 - Information processing – Text and office systems – Standard Generalized Markup Language (SGML). International Standard, Confirmed, 1986.

[4] Joint Technical Committee ISO/IEC (JTC 1). ISO/IEC 9075 - Part 2 (Foundation for SQL). International Standard, Working Draft, 1994.

[5] Joint Technical Committee ISO/IEC (JTC 1). ISO/IEC 9075 - Part 1 (Framework for SQL). International Standard, Working Draft, 1995.

[6] D. Raggett. HTML 3.2 Reference Specification. `http://www.w3.org/TR/REC-html32`.

[7] W3C. Extensible Markup Language (XML) 1.0. `http://www.w3.org/TR/1998/REC-xml-19980210.html`.

[8] W3C. HTML 4.01 Specification. `http://www.w3.org/TR/html401/`.

[9] W3C. XML Schema Part 0: Primer. `http://www.w3.org/TR/xmlschema-0/`.

[10] W3C. XML Schema Part 1: Structures. `http://www.w3.org/TR/xmlschema-1/`.

[11] W3C. XML Schema Part 2: Datatypes. `http://www.w3.org/TR/xmlschema-2/`.