The OXL format for the exchange of integrated datasets

Jan Taubert[1*], Klaus Peter Sieren[2], Matthew Hindle[1], Berend Hoekman[3], Rainer Winnenburg[1], Stephan Philippi[2], Chris Rawlings[1], Jacob Köhler[1]

[1] Department of Biomathematics and Bioinformatics, Rothamsted Research, AL5 2JQ, Harpenden, UK
[2] Department of Computer Science, University of Koblenz, Germany
[3] Department of Bioinformatics, Wageningen University, The Netherlands

Abstract

A prerequisite for systems biology is the integration and analysis of heterogeneous experimental data stored in hundreds of life-science databases and millions of scientific publications. Several standardised formats for the exchange of specific kinds of biological information exist. Such exchange languages facilitate the integration process; however they are not designed to transport integrated datasets. A format for exchanging integrated datasets needs to i) cover data from a broad range of application domains, ii) be flexible and extensible to combine many different complex data structures, iii) include metadata and semantic definitions, iv) include inferred information, v) identify the original data source for integrated entities and vi) transport large integrated datasets. Unfortunately, none of the exchange formats from the biological domain (e.g. BioPAX, MAGE-ML, PSI-MI, SBML) or the generic approaches (RDF, OWL) fulfil these requirements in a systematic way.

We present OXL, a format for the exchange of integrated data sets, and detail how the aforementioned requirements are met within the OXL format. OXL is the native format within the data integration and text mining system ONDEX. Although OXL was developed with the ONDEX system in mind, it also has the potential to be used in several other biological and non-biological applications described in this paper.

Availability: The OXL format is an integral part of the ONDEX system which is freely available under the GPL at http://ondex.sourceforge.net/. Sample files can be found at http://prdownloads.sourceforge.net/ondex/ and the XML Schema at http://ondex.svn.sf.net/viewvc/*checkout*/ondex/trunk/backend/data/xml/ondex.xsd.

1 Introduction

The importance of database integration for all Life Sciences is generally recognised. Especially in the Pharmaceutical Industry [1; 2] data integration is a crucial technology for the drug discovery process [3; 4]. Although many different approaches to data integration exist, and have been reviewed by [5], there are no standard formats, specifically designed for the exchange of integrated datasets. Thus, users of database integration systems have to rely on the proprietary interfaces and exchange formats from the different data integration platforms. Although the use of XML, RDF and OWL simplifies the exchange of integrated datasets, none of the existing XML, RDF and OWL based formats is suitable as a generic format for the exchange of integrated datasets.

Data integration has to deal with a broad range of heterogeneous data sources. Traditionally, databases were distributed using proprietary flatfile formats or tab delimited database dumps. It is a popular myth that the appearance of XML has made these formats obsolete, however our experience shows that still only about 5% of all databases provide an XML based format

* To whom correspondence should be addressed.

[6]. Several databases are still exclusively distributed in proprietary flatfile formats or as database dumps. A high percentage of these databases provide no exchange format and can only be accessed through HTML based web-pages. Standardised exchange formats have great potential in improving and simplifying database integration, as generic tools and interfaces can be (re-)used. Widespread adoption of an exchange format leads to improved data documentation and will inevitably improve the exchange formats as formal agreements on data content and level of detail are reached between data providers.

We begin by introducing the requirements for the exchange of integrated data sets and proceed to discuss and compare existing biological data formats like BioPAX, MAGE-ML, PSI-MI and SBML. We conclude that none of the outlined exchange formats sufficiently meet the needs of data integration. We describe how our requirements form the specification for the OXL exchange format, which we developed specifically for the exchange of integrated datasets. We demonstrate its usability for data exchange within the different components of the database integration framework ONDEX [7; 8] and external applications. Finally we discuss the various applications of OXL, and give an outlook on the extension of the format and our plans to improve OXL's supporting tools.

2 Requirements for exchanging integrated data sets

Several XML-based exchange formats have been developed for representing data and models in specific biological areas, including the "Biological Pathway Exchange" (BioPAX) format [9], CellML markup language for describing mathematical models [10], "Chemical Markup Language" (CML) [11], "Microarray Gene Expression Language" (MAGE-ML) [12], "Protein Markup Language" (ProML) [13], "Proteomics Standards Initiative's Molecular Interaction" (PSI-MI) format [14] and "Systems Biology Markup Language" (SBML) [15].

Some of these exchange formats cover a broader biological area than others. SBML, for example, is a language for describing models common to research in many areas of computational biology, including cell signalling pathways, metabolic pathways, gene regulation, and others [16]. SBML has become a *de facto* standard for representing formal quantitative and qualitative models at the level of biochemical reactions and regulatory networks [15]. BioPAX enables the integration of diverse pathway resources by defining an open file format specification for the exchange of biological pathway data. BioPAX includes representations for metabolic pathways, molecular interaction and promises future support for signalling pathways. It adopts some of the mechanism used in the PSI-MI format [9].

Several bioinformatics tools such as Cellerator (SBML) [17], Cytoscape (BioPAX) [18], E-Cell (SBML) [19], Gepasi (SBML) [20], Pathway Tools (BioPAX) [21] and VisANT (BioPAX) [22] use these formats for the importing/exporting of domain specific data. However, these formats are not normally used for exchanging a broad range of integrated data, which involves several data integration specific requirements. These requirements are listed in Table 1.

In biology, exchange languages are normally designed for a very specific application domain, such as the exchange of protein interactions (PSI-MI), description of microarray experiments (MAGE-ML), representation of formal quantitative and qualitative models (SBML) or exchanging pathway information (BioPAX); thus they do not satisfy the first requirement specified in Table 1.

Many of the above mentioned formats have only a limited functionality to incorporate new complex data structures (see second requirement in Table 1). Structural representation of complex data is predefined by the given format, for example the use of MathML [23] within

the PSI-MI format to model equations. Adding complex data outside the defined representations therefore requires a change to the underlying schema of the format.

In defined ontology formats like BioPAX, which is represented in OWL-DL, all entity classes are predefined (may oppose third requirement in Table 1). For example if one likes to describe H+ as an inorganic substance (as subclass of *physicalEntity*) and not as an instance of the *smallMolecule* subclass of BioPAX, this requires the modification of the BioPAX schema to enable document validation by OWL reasoners.

Some formats, like MAGE-ML or PSI-MI are able to include metadata about the information sources and methods that were used to generate the contained information. But not all of the above mentioned formats are completely able to satisfy the fourth and fifth requirement of Table 1. The last requirement in Table 1 is also not always satisfied, often because sophisticated document validation is involved. For some formats, especially OWL-DL based formats this is a demanding computational task with high inherent time and space complexity; from our experience this is the limiting factor for existing tools. Therefore, none of the existing formats satisfy all of the aforementioned requirements for exchanging integrated biological data.

Motivation	Requirement	Example
Data integration needs to include all kinds of biological data, e.g. pathway data, gene expression data, biochemical reactions etc.	**i) Cover data from a broad range of application domains.**	Combined analysis of microarray and metabolomic data in the context of integrated pathway data.
Biological research progresses and new understanding may emerge that result in novel complex data structures.	**ii) Be extensible to combine many different complex data structures.**	The current transition from a gene-centric to network-based representation of molecular biology. [Mewes, NAR, 2006]
Biological ontologies are subject to frequent changes.	**iii) Be flexible with meta data and semantical information from other sources.**	During the evolution of the PSI-MI format to the current release (version 2.5), major changes have occurred.
Not all relationships between biological entities are present in the data sources.	**iv) In addition to integrated data also include inferred information.**	New relationships between biological entities may be identified by data integration and analysis methods.
Integrated data may originate from several different data sources or be inferred computationally.	**v) Identify the original data source for integrated entities.**	Integrating several pathway databases, like KEGG and BioCyc at once.
Biological knowledge is steadily growing, as is the data contained in biological databases.	**vi) Transport large amounts of integrated data.**	The KEGG database in its flat file representation is already more than 4GByte large.

Table 1 Requirements for exchanging integrated data

3 The OXL format

This section describes the OXL exchange format and why we choose XML Schema to implement it. OXL was originally developed to exchange integrated data sets between different components of the ONDEX system [7], and with external applications. However, we believe that OXL also has several potential applications independent of ONDEX.

3.1 A brief history of OXL

The design of the OXL format is closely coupled with how data is represented in the ontology based data structure of the ONDEX system. Integrated data in ONDEX is modelled as a graph, where the nodes are termed concepts and the edges relations. Concepts correspond to biological entities, e.g. a gene, a protein, an enzyme or a pathway. Relations describe how these biological entities interact or relate to each other, e.g. a protein is encoded by a gene; an enzyme can take part in a pathway. Whether a particular concept is a gene, a protein or a pathway is determined by the concept class metadata of this concept. The nature of the relation that connects two concepts is specified by the relation type metadata for each relation. The use of metadata enables us to cover data from a broad range of application domains (see first requirement in Table 1) and to be flexible with changes in metadata and semantic information from other sources (see third requirement in Table 1). Each concept and relation is marked with the data source from which (controlled vocabulary) it originates, and the method that was used to create it in ONDEX (evidence): to keep track of provenance in the process of data integration (see fourth and fifth requirement in Table 1).

Before we had started to develop the OXL format, we carefully investigated existing formats such as SBML and BioPAX. Using one of these formats for the ONDEX system would have had several advantages such as good tool support through, e.g. libSMBL (see http://www.sbml.org/software/libsbml/), the Jena API (see http://jena.sourceforge.net), and improved compatibility with other bioinformatics tools. Unfortunately, as we have reported in the previous section, despite SBML and BioPAX being well developed and successful exchange languages, they are not suitable as a generic exchange language to describe integrated data from multiple sources, or to exchange data between the different components of the ONDEX system. Given the obvious strengths of these data formats we include import and export functionality from ONDEX to a range of different formats, including certain aspects in SBML.

The overriding priority when selecting the OXL technical structure was a well-defined and widely adopted software-readable format (see last requirement in Table 1). We chose XML, the eXtensible Markup Language [24], because of its portability and increasingly widespread acceptance as the standard data language for bioinformatics [25]. There are, however, different approaches to data representation in XML and we also considered the RDF and OWL XML Schema.

Modelling the ontology based data structure of ONDEX in OWL would have restricted us to predefined metadata for concepts and relations (e.g. concept class and relation type): defined as OWL classes and sub classes. This would have limited us to a static set of metadata at runtime. Although this is attractive in terms of the required programmatic complexity and computational reasoning; we decided to look for a different way, which would not require changing the OWL schema file every time we introduce new metadata for concepts or relations. Also special tool support in the form of reasoners is required to make full use of the expressiveness of OWL-DL. Validation of documents in OWL-DL has a inherently high time and space complexity. This tool support was missing and we experienced the same problems with OWL-DL as described by [26].

Tool support for the RDF format is better than for OWL, because in comparison the complexity of document validation is reduced. The preferred way of representing relationships in RDF is in the form of *subject – predicate – object*. Here *subject* and *object* correspond to ontology concepts and *predicate* to the relation between concepts. RDF does not allow the direct association of complex metadata (complex relation types, evidence types etc.) with the *predicate*. One workaround is to model a relation from our ontology based data structure as an *object*; references are added between the first participant in the relation to the *object* and the second participant in the relation. However, this reduces compatibility with existing tools, as this kind of modelling is not within the scope of standard RDF design. Representing ternary relationships in RDF is difficult and not an explicit part of the syntax elements. However, there exists several model variants for n-ary relations mentioned in (see http://www.w3.org/TR/swbp-n-aryRelations/). For cross system compatibility, we have developed an RDF based export for our ontology based data structure. However, due to the aforementioned problems and other minor issues encountered, some information loss in the conversion to RDF is unavoidable.

3.2 OXL as XML Schema

Our final conclusion was in favour of XML Schema because of its already widespread adoption and abundance of tools. XML Schema gave us the most flexibility in modelling our ontology based data structure. The principles used are reflected in the XML Schema of OXL as shown in Figure 1. The start data element *ondexdata* includes either the *ondexmetadata* or the *ondexdataseq* element. This facilitate the use of one XML Schema for both, describing metadata (*ondexmetadata*) in terms of a controlled vocabulary and defined evidences, and a complete graph structural representation for concepts and relations (*ondexdataseq*). An OXL file containing only metadata is required to initialize the ONDEX data integration framework with a common set of agreed metadata.

The *ondexmetadata* element consists of an OndexMetaDataSeqType, containing a list of all possible kinds of metadata used in the ontology based data structure (see third requirement in Table 1). Each metadata element is represented by a set of values containing: a unique identifier (*id*), name (*fullname*) and free text description (*description*) to represent human readable information. These identifying values are common to all metadata elements, which are detailed in the following. The *cvs* element contains a list of databases, called controlled vocabulary (*cv*). The units of properties assigned to concepts and relations are grouped together in the *units* element. Types of *evidence* for concepts and relations are contained within the *evidences* element. A *unit* can also be part of an attribute name (*attrname*) contained in the *attrnames* element. An attribute name is the first participant in a name-value pair, which together is termed a generalised data structure (GDS) [8] element; this can be assigned to any concept or relation. An *attrname* element beside the common identifier set also contains a *datatype* element and a *specialisationOf* element. The *datatype* element usually specifies the JAVA (see http://java.sun.com) class of the GDS value, but is not limited to JAVA classes in general. This allows for representing complex data structures such as protein structure, or Position Weight Matrices (PWM), which describes the DNA binding motifs of transcription factors (see second requirement in Table 1). The *datatype* element intentionally avoids the use of predefined data types in XML Schema and thus enables addition of new data types without having to modify the XML Schema of OXL. The *specialisationOf* element contains another *attrname* element, and represents the model hierarchy within attribute names. The same principle for modelling hierarchy within the metadata in OXL is implemented in both concept classes (*cc*) which are wrapped in the *conceptclasses* element, and relation types (*relation_type*) which are contained within the *relationtypes* element. A *relation_type* element has additional attributes that characterise the

properties of the relation according to the OBO Relation Ontology [27]. Relation types can be grouped together in the *rtset* element of a relation type set (*relationtypeset*) in the list of relation type sets (*relationtypesets*). This provides the most flexible way of specifying the type of a relation, where the data source requires that relation types are a conglomerate of several relation types, e.g. MeSH use the same relation type to represent is-a and part-of relationships. A relation can have: a set of one or more relation types without a given hierarchy, a set of one or more relation types in a full defined hierarchy, or a mixture of both.

The *ondexdataseq* element is an OndexDataSeqType containing lists of concepts (*concepts*) and relations (*relations*). Each *concept* is identified by a unique *id* element (an integer). Textual information about a concept is contained in *annotation* and *description*. Additionally *concept* contains a *pid* element. This can be an alternative textual identifier for the concept which is more understandable. The *elementOf* element defines the controlled vocabulary (*cv*) from which the concept originates. The *ofType* tag represents the concept class (*cc*) for the concept. The types of evidences (*evidence*) for the concept are collected within the *evidences* element. A concept can have synonyms which are expressed as concept names (*conames*), references to other data sources termed concept accessions (*coaccessions*), and arbitrary name-value pairs, encompassed by the generalised data structure (*cogds*). A *relation* is identified by the unique combination of *fromConcept*, *toConcept*, an optional *qualifier* concept, and a relation type set (*ofTypeSet*). The addition of the qualifier concept enables OXL to represent ternary relationships. A relation is assigned evidence types (*evidences*) and arbitrary name-value pairs (*relgds*), which are handled in an equivalent way to concepts.

Except the reuse of unique concept IDs for *fromConcept*, *toConcept* and *qualifier* elements in a relation, no other cross document references are made: all elements are always fully expanded, in a similar way to that used by the expanded form of the PSI-MI format [14]. Thus these elements always contain a full copy of a possible already existing other element that is associated as a property of the current element. This facilitates the merging of several OXL documents and enables easy transformation of the OXL format into streamlined formats like HTML using XSLT stylesheets. However, this introduces redundancy in the file format and thus increases the average size of OXL documents. However, due to the higher redundancy when compressed these files are only marginally bigger than the equivalent non-redundant file format.

We plan to introduce a versioning system for future releases of OXL to keep track of the changes within the document versions and provide scripts for upgrading existing data stored in OXL to the newest version. The format described here is the first official release of the OXL format.

3.3 Support for data integration and text mining

The OXL format was designed to support the task of data integration and text mining in ONDEX. Metadata plays an important role in the course of integrating data in a semantically consistent way. Equivalent entities in different data sources, which represent the same biological object, e.g. genes imported from KEGG [28] and genes imported from BioCyc [29], and thus share a common semantic definition, will also share a common metadata association (see third requirement in Table 1). Therefore, all data in OXL uses the same set of metadata compiled for the corresponding application domain (see first requirement in Table 1).

Sets of metadata include concept classes, relation types and attribute names. Concept classes, relation types and attribute names can be part of a hierarchical structure in the form of a taxonomy. For concept classes the top level root element is *Thing*. All other concept classes like *Gene*, *Protein* or *Pathway* are inherited from the root element. An example of a hierarchy

of attribute names is given by the biological sequence types: cDNA is a specialisation of DNA, and mRNA is a specialisation of RNA. The OXL Relation types are based primarily on the OBO Relation Ontology [27]. Figure 2 shows an excerpt for the OBO relation type *preceded_by* (pr_by) emphasizing on the fully expanded *specialisationOf* element containing the more general *is_related_to* (r) relation type.

Provenance of data is tracked using controlled vocabulary (CV) and evidence types (see fifth requirement in Table 1). The controlled vocabulary marks the data source the data was imported from: e.g. KEGG [28], Transfac [30], Transpath [31]. Instead of technically merging equivalent concepts and relations from different data sources, these concepts and relations are only aligned to each other using the special relation type *equivalent* (equ). This makes it possible to disentangle integrated data sets (see fourth requirement in Table 1). Evidence codes are used for keeping track of how data was integrated in ONDEX. This facilitates the extraction or filtering of relations generated by a specified data analysis method. Evidence codes can also be applied to inferred (generated) concepts and relations (see fourth requirement in Table 1).

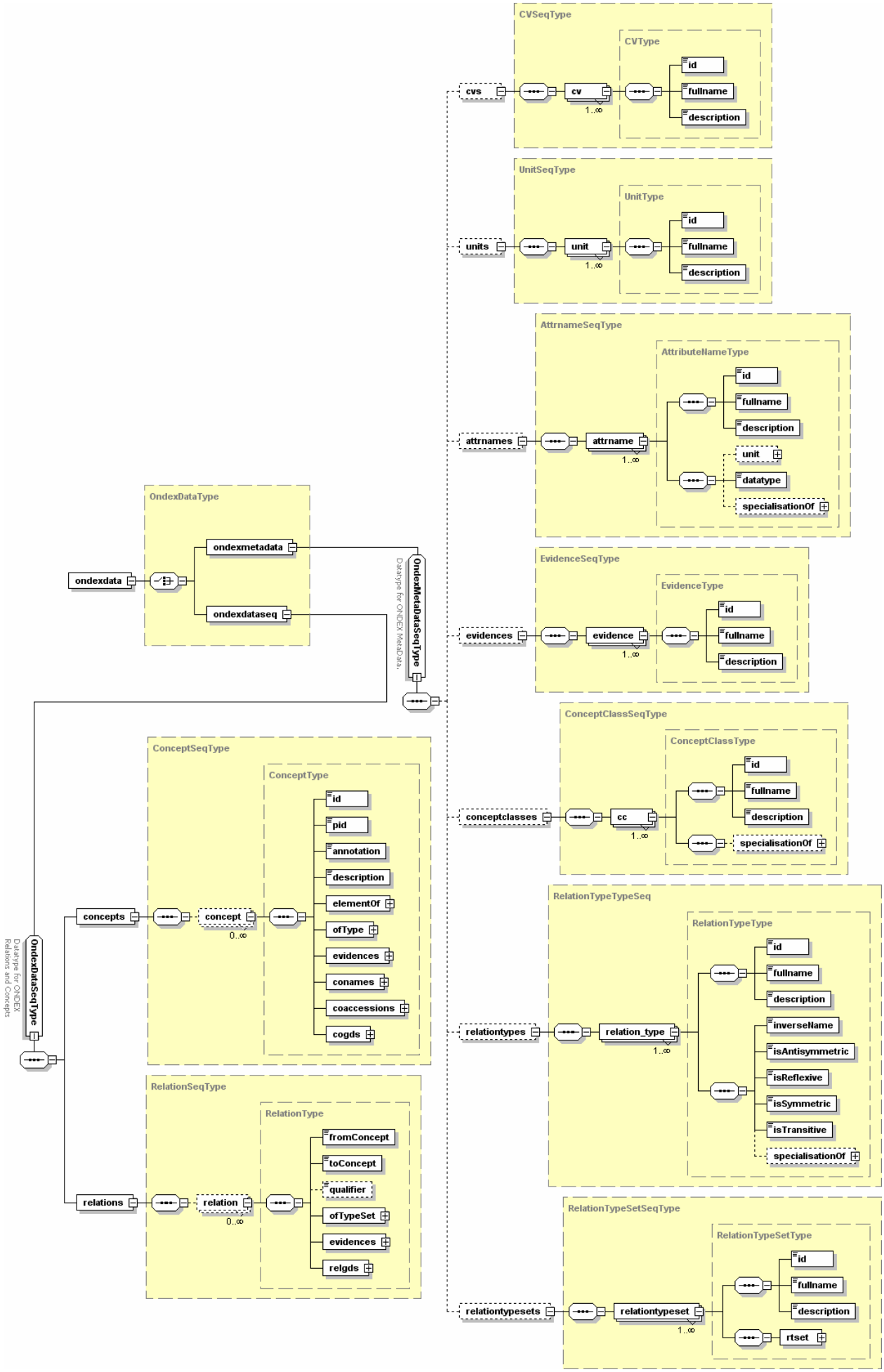

Figure 1 Showing the XML Schema of OXL

```
<relation_type>.
        <id>pr_by</id>.
        <fullname>preceded by</fullname>.
        <description>A is the direct outcome of B</description>.
        <inverseName>none</inverseName>.
        <isAntisymmetric>false</isAntisymmetric>.
        <isReflexive>false</isReflexive>.
        <isSymmetric>false</isSymmetric>.
        <isTransitive>true</isTransitive>.
        <specialisationOf>.
                <id>r</id>.
                <fullname>r</fullname>.
                <description>is related to</description>.
                <inverseName>relation type</inverseName>.
                <isAntisymmetric>false</isAntisymmetric>.
                <isReflexive>false</isReflexive>.
                <isSymmetric>false</isSymmetric>.
                <isTransitive>false</isTransitive>.
        </specialisationOf>.
</relation_type>.
```

Figure 2 OXL metadata excerpt for pr_by relation type

4 Tool support and applications of OXL

OXL was developed as part of the ONDEX data integration framework. Thus it has a close relationship to the core data structure. There are three possible technical approaches for generating an OXL document. Firstly, a combination of a document objects model (DOM) and an XML writer/reader to handle OXL. This is an appropriate approach for the conversion of small custom data models into OXL. Secondly, stream based XML parsing/writing approaches such as SAX (see http://www.saxproject.org/) or StAX (see http://www.jcp.org/en/jsr/detail?id=173) which can be used to directly read or write XML documents in the syntax of OXL. These techniques use the smallest memory footprint and are the only viable approach for large data sets. We use StAX based parsing within ONDEX. Thirdly, for very large data models, we recommended the use of the ONDEX core API (*"core"*). This API includes methods for the efficient construction and manipulation of very large graphs and OXL exports and imports. As ONDEX stores all data on disk in its own persistency layer and keeps only a cached subset of data in memory the handling of very large data sets is possible. Furthermore, the *"core"* can be used as a persistent management system for custom data integration applications in JAVA. The current ONDEX system which is available at http://ondex.sourceforge.net includes support tools for reading and writing of data in the OXL format.

Editing metadata within the *"core"* is enabled through the Metadata Editor, which provides a graphical hierarchical-tree based representation of the metadata. We encourage users/developers to submit their own metadata additions and modifications to the central OXL document located in the Subversion repository of ONDEX, or to suggest such changes through the developer mailing list.

To demonstrate that OXL can be used to exchange complete databases without information loss, we implemented an OXL export for the Pathogen Host Interaction database (PHI-base) [32]. This functionality is currently only provided in the beta version of PHI-base and will appear as part of the next release. It is possible to download the complete database or selected query results in the OXL format; for example a search for the entry "PHI:441" yields the result table depictured in part A of Figure 3. Part B of Figure 3 shows an excerpt from the generated OXL file and part C of Figure 3 shows a screenshot of the ONDEX Visualisation Tool Kit (OVTK), with the loaded OXL file showing an organic layout of the data.

A

Results

Genes: 1 Interactions: 4 Export XML

PHI-base accession	Gene name	EMBL accession	Phenotype of mutant	Pathogen species	Disease name	Experimental host
PHI:441 Details	BTP1	CAE55153	Reduced virulence	Botrytis cinerea	Grey mold rot	Bean
	BTP1	CAE55153	Unaffected pathogenicity	Botrytis cinerea	Grey mold rot	Tomato
	BTP1	CAE55153	Unaffected pathogenicity	Botrytis cinerea	Grey mold rot	Apple
	BTP1	CAE55153	Unaffected pathogenicity	Botrytis cinerea	Grey mold rot	Green pepper

B

```
<concept>.
 <id>1</id>.
 <pid>441</pid>.
 <annotation></annotation>.
 <description></description>.
 <elementOf>.
  <id>PHI</id>.
  <fullname>PHI-base</fullname>.
  <description>Pathogen - Host Interaction database</description>.
 </elementOf>.
 <ofType>.
  <id>Gene</id>.
  <fullname>Gene</fullname>.
  <description>Gene</description>.
 </ofType>.
 <evidences>.
  <evidence>.
   <id>IMPD</id>.
   <fullname>IMPD</fullname>.
   <description>imported from database</description>.
  </evidence>.
 </evidences>.
 <conames>.
  <concept_name>.
   <name>BTP1</name>.
   <isPreferred>true</isPreferred>.
  </concept_name>.
 </conames>.
 <coaccessions>.
  <concept_accession>.
   <accession>PHI:441</accession>.
   <elementOf>.
    <id>PHI</id>.
    <fullname>PHI-base</fullname>.
    <description>Pathogen - Host Interaction database</description>.
   </elementOf>.
   <ambiguous>false</ambiguous>.
  </concept_accession>.
  ....
 </coaccessions>.
 <cogds>.
  ....
 </cogds>.
</concept>.
```

C

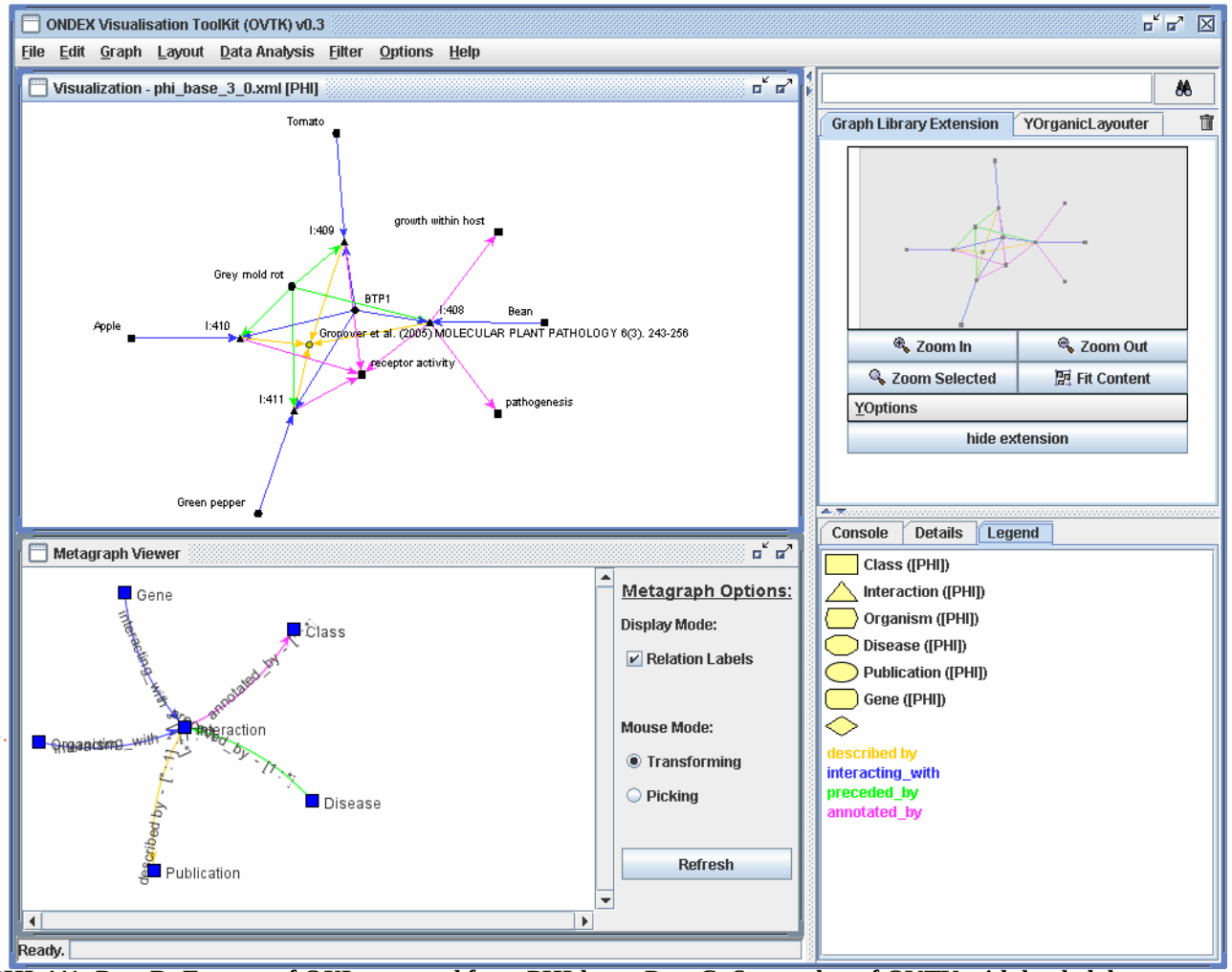

Figure 3 Part A: Result table for PHI-base entry PHI:441; Part B: Excerpt of OXL exported from PHI-base; Part C: Screenshot of OVTK with loaded data from OXL displaying relationships of PHI:441. The actual graph of relationships is displayed in the upper Visualization frame, whereas an overview of the involved metadata (concept classes like Gene, Disease, and relation types between members of these concept classes like preceded_by, interacting_with) is given in the Metagraph Viewer. On the right side a birds-eye overview is presented and the different relations type colours and concept class symbols are depicted in the lower part.

5 The role of OXL in the ONDEX data integration framework

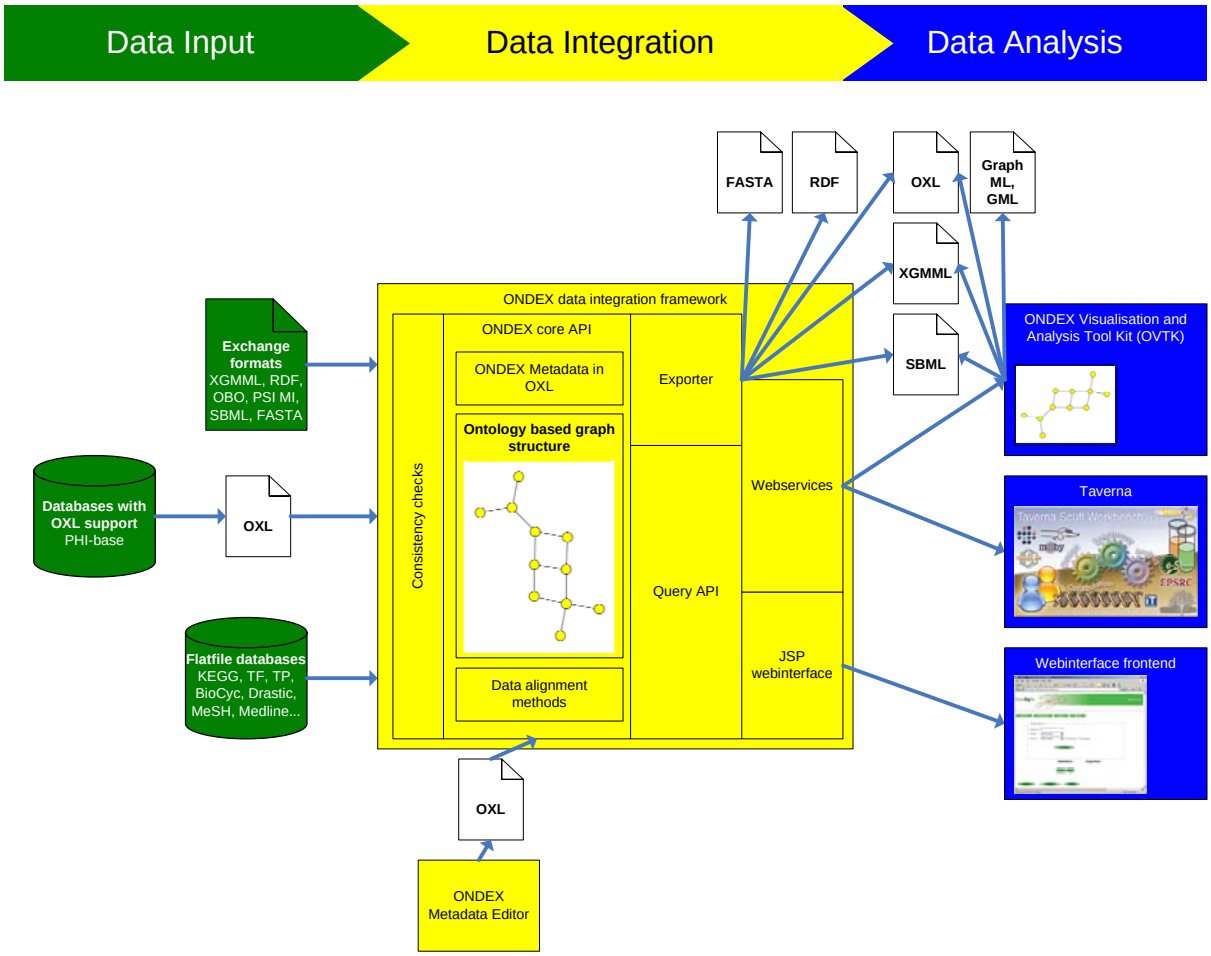

Figure 4 shows the different components of ONDEX and how ONDEX makes use of OXL as well as other standard exchange formats. Integration runs in ONDEX can be divided into three steps: input of data from different data sources, the integration process in the ONDEX core API, and data analysis using different tools and interfaces.

The data integration framework consists of three parts. First data is loaded or parsed from different data sources into the ONDEX core API (*"core"*), passing several consistency checks. The *"core"* uses ONDEX Metadata and data alignment methods to perform the data integration step from the original data into the ontology based graph structure which is hold in its own persistency layer. The data can be retrieved using the Query API and associated web services or by the JSP based web interface. We also provide several exporters, which work directly with the *"core"*. Once the data has been retrieved, it can finally be applied to data analysis. The ONDEX Visualisation Tool Kit (OVTK) is one such data analysis tool that can be applied to data retrieved from the ONDEX system. The OVTK uses the same *"core"* and provides additional graph-based visualisation and analysis methods.

OXL is also used for data transfer between different components of the system and the storage of integrated datasets. Besides the OXL format, ONDEX makes use of a range of different exchange formats and data sources. Data from several data sources are imported using database specific parsers or generalised importers for supported exchange formats. An up-to-date list of these exchange formats and databases can be found on the webpage (see http://ondex.sourceforge.net/feats.php). By using the OBO format for example, it is possible to import about 50 different ontologies (see http://obofoundry.org/), and the PSI-MI interface

of ONDEX provides access to 8 protein interaction databases (see http://www.psidev.info/index.php?q=node/60#data), supporting this format. There are also importers for XGMML (see http://www.cs.rpi.edu/~puninj/XGMML/draft-xgmml.html) and SBML. In addition, ONDEX provides parser for KEGG [28], Transfac [30], Transpath [31], Drastic [33], EC [34], BioCyc [29], MeSH [35] and Medline [36]. The flatfiles of these databases are stored locally and are read directly by the parsers. Therefore, it is not required that these databases are installed or that a permanent internet connection is present. Besides the currently supported exchange formats and data sources, several other data import parsers are underway and will be added in the near future.

Metadata for the data integration core is provided using OXL and can be easily edited using the Metadata Editor. New relationships between imported concepts can be identified by data alignment methods: e.g. concept accessions matching, biological sequence similarity or text mining. After the data integration run has finished, the integrated data can be accessed through web services or exported directly into several exchange formats, including OXL, XGMML and SBML. The web services are used by the ONDEX Visualisation Tool Kit (OVTK) and Taverna [37]. A web interface is currently under development. Other application can utilise one of the exchange formats provided by ONDEX to load data, e.g. XGMML can be used to load exported data into Cytoscape [18].

6 Discussion

The decision to create our own exchange format for integrated data sets is based on two main factors. The first is that none of the existing bio-specific exchange languages are capable of satisfying all the requirements mentioned in Table 1 and that generic exchange formats like RDF and OWL would have imposed overly rigid restrictions on the ontology based data structure. Our second motivation for creating OXL was to closely couple it with our ontology based data structure; so that an understanding of the ONDEX ontology based data structure confers understanding of OXL.

In addition to our native format OXL, we provide a model of our ontology based data structure in RDF, in order to utilize existing RDF tools. This model involves workarounds detailed earlier in this paper. However, this format is less intuitive and requires more time and effort for others to use. Existing RDF tools such as Jena normally use an in-memory model, and as such users of the RDF format who need to process large amounts of data can not benefit from the existing RDF tool support.

OXL satisfies all requirements listed in Table 1. The use of flexible metadata assignment to concepts and relations enables OXL to cope with broad range application domains (first requirement). Arbitrary complex data structures (second requirement) can be included using special name-value pairs, called generalized data structure (GDS). Metadata and references to information from other sources (third requirement) can easily be modified without changing the schema definitions. Inferred information (fourth requirement) and tracking of provenance (fifth requirement) is realized through special relation types and evidence types for concepts and relations. By using a StAX parsing approach, XML Schema validation and optional file compression it is possible to transport very large datasets in OXL (last requirement).

As shown in the example of PHI-base, exporting databases into the OXL format is less work than developing a flatfile parser for the ONDEX system to import the data. Because OXL is almost fully expanded, with the exception of references to unique concepts IDs as used in a relation, it is possible to create XSLT stylesheets to transform OXL into other streamlined formats like HTML. This principle is utilized by the ONDEX web interface, which is currently under development. Applications that want to make the most of integrated datasets

created by ONDEX should use the OXL format. Other exchange formats like RDF, SBML or XGMML are also provided by the ONDEX system, but have inherent limitations as to how data can be represented and therefore may not contain all the information that would otherwise be contained within OXL.

The generation and exchange of integrated data from several sources also involves a legal aspect. Licensing models for data may differ between imported data sources, which makes it important to track provenance: from where the integrated datasets originate [6]. OXL includes such a provenance tracking mechanism. Concepts and relations from different data sources are only aligned to each other: concepts from two databases that are equivalent remain as separate concepts within ONDEX, connected by an equivalence relation. It is therefore possible to limit the scope of information that is exchanged and thus satisfy license agreements.

6.1 Outlook

We will release the ONDEX core API (*"core"*) as a standalone JAVA module that can be reused by custom applications. This API includes support for reading and writing OXL files. The *"core"* also provides fast and efficient indexing and search functionality for OXL data. Also in the near future we will introduce a versioning system for the OXL format and standardized curation for OXL metadata. This we hope will encourage other database providers to support the format. The ONDEX core will soon include a configurable importer for OXL, which downloads directly from the SOAP based web service run by databases supporting OXL.

7 References

[1] J. Augen, Information technology to the rescue! Nat Biotechnol 19 Suppl (2001) BE39-40.
[2] E. Pennisi, How will big pictures emerge from a sea of biological data? Science 309 (2005) 94.
[3] R. Carel, Practical data integration in biopharmaceutical research and development. PharmaGenomics 3 (2003) 22–35.
[4] D.B. Searls, Data integration: challenges for drug discovery. Nat Rev Drug Discov 4 (2005) 45-58.
[5] J. Köhler, Integration of life science databases. Drug Discovery Today: BIOSILICO 2 (2004) 61-9.
[6] S. Philippi, and J. Köhler, Addressing the problems with life-science databases for traditional uses and systems biology. Nat Rev Genet 7 (2006) 482-8.
[7] J. Köhler, J. Baumbach, J. Taubert, M. Specht, A. Skusa, A. Ruegg, C. Rawlings, P. Verrier, and S. Philippi, Graph-based analysis and visualization of experimental results with ONDEX. Bioinformatics 22 (2006) 1383-90.
[8] J. Köhler, C. Rawlings, P. Verrier, R. Mitchell, A. Skusa, A. Ruegg, and S. Philippi, Linking experimental results, biological networks and sequence analysis methods using Ontologies and Generalised Data Structures. In Silico Biol 5 (2005) 33-44.
[9] G. Bader, and M. Cary, BioPAX - Biological Pathways Exchange Language, BioPAX Workgroup, 2005.
[10] C.M. Lloyd, M.D. Halstead, and P.F. Nielsen, CellML: its future, present and past. Prog Biophys Mol Biol 85 (2004) 433-50.
[11] Y.M. Liao, and H. Ghanadan, The chemical markup language. Anal Chem 74 (2002) 389A-390A.
[12] P.T. Spellman, M. Miller, J. Stewart, C. Troup, U. Sarkans, S. Chervitz, D. Bernhart, G. Sherlock, C. Ball, M. Lepage, M. Swiatek, W.L. Marks, J. Goncalves, S. Markel, D. Iordan, M. Shojatalab, A. Pizarro, J. White, R. Hubley, E. Deutsch, M. Senger, B.J. Aronow, A. Robinson, D. Bassett, C.J. Stoeckert, Jr., and A. Brazma, Design and implementation of microarray gene expression markup language (MAGE-ML). Genome Biol 3 (2002) RESEARCH0046.
[13] D. Hanisch, R. Zimmer, and T. Lengauer, ProML--the protein markup language for specification of protein sequences, structures and families. In Silico Biol 2 (2002) 313-24.
[14] H. Hermjakob, L. Montecchi-Palazzi, G. Bader, J. Wojcik, L. Salwinski, A. Ceol, S. Moore, S. Orchard, U. Sarkans, C. von Mering, B. Roechert, S. Poux, E. Jung, H. Mersch, P. Kersey, M. Lappe, Y. Li, R. Zeng, D. Rana, M. Nikolski, H. Husi, C. Brun, K. Shanker, S.G. Grant, C. Sander, P. Bork, W. Zhu, A. Pandey, A. Brazma, B. Jacq, M. Vidal, D. Sherman, P. Legrain, G. Cesareni, I. Xenarios, D. Eisenberg,

B. Steipe, C. Hogue, and R. Apweiler, The HUPO PSI's molecular interaction format--a community standard for the representation of protein interaction data. Nat Biotechnol 22 (2004) 177-83.

[15] M. Hucka, A. Finney, B.J. Bornstein, S.M. Keating, B.E. Shapiro, J. Matthews, B.L. Kovitz, M.J. Schilstra, A. Funahashi, J.C. Doyle, and H. Kitano, Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. Syst Biol (Stevenage) 1 (2004) 41-53.

[16] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, A.P. Arkin, B.J. Bornstein, D. Bray, A. Cornish-Bowden, A.A. Cuellar, S. Dronov, E.D. Gilles, M. Ginkel, V. Gor, Goryanin, II, W.J. Hedley, T.C. Hodgman, J.H. Hofmeyr, P.J. Hunter, N.S. Juty, J.L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L.M. Loew, D. Lucio, P. Mendes, E. Minch, E.D. Mjolsness, Y. Nakayama, M.R. Nelson, P.F. Nielsen, T. Sakurada, J.C. Schaff, B.E. Shapiro, T.S. Shimizu, H.D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19 (2003) 524-31.

[17] B.E. Shapiro, A. Levchenko, E.M. Meyerowitz, B.J. Wold, and E.D. Mjolsness, Cellerator: extending a computer algebra system to include biochemical arrows for signal transduction simulations. Bioinformatics 19 (2003) 677-8.

[18] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res 13 (2003) 2498-504.

[19] M. Tomita, K. Hashimoto, K. Takahashi, T.S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J.C. Venter, and C.A. Hutchison, 3rd, E-CELL: software environment for whole-cell simulation. Bioinformatics 15 (1999) 72-84.

[20] P. Mendes, GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems. Comput Appl Biosci 9 (1993) 563-71.

[21] P.D. Karp, S. Paley, and P. Romero, The Pathway Tools software. Bioinformatics 18 Suppl 1 (2002) S225-32.

[22] Z. Hu, J. Mellor, J. Wu, T. Yamada, D. Holloway, and C. Delisi, VisANT: data-integrating visual framework for biological networks and modules. Nucleic Acids Res 33 (2005) W352-7.

[23] P. Ion, and R. Miner, Mathematical Markup Language (MathML) 1.01 Specification, W3C, 1999.

[24] T. Bray, J. Paoli, and C.M. Sperberg-McQueen, Extensible markup language. World Wide Web J. 2 (1997) 29-66.

[25] F. Achard, G. Vaysseix, and E. Barillot, XML, bioinformatics and data integration. Bioinformatics 17 (2001) 115-25.

[26] A. Ruttenberg, J.A. Rees, and J.S. Luciano, Experience using OWL DL for the exchange of biological pathway information, OWL Experiences and Directions, 2005.

[27] B. Smith, W. Ceusters, B. Klagges, J. Kohler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A.L. Rector, and C. Rosse, Relations in biomedical ontologies. Genome Biol 6 (2005) R46.

[28] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa, KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res 27 (1999) 29-34.

[29] P.D. Karp, C.A. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahren, S. Tsoka, N. Darzentas, V. Kunin, and N. Lopez-Bigas, Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. Nucleic Acids Res 33 (2005) 6083-9.

[30] E. Wingender, P. Dietze, H. Karas, and R. Knuppel, TRANSFAC: a database on transcription factors and their DNA binding sites. Nucleic Acids Res 24 (1996) 238-41.

[31] F. Schacherer, C. Choi, U. Gotze, M. Krull, S. Pistor, and E. Wingender, The TRANSPATH signal transduction database: a knowledge base on signal transduction networks. Bioinformatics 17 (2001) 1053-7.

[32] R. Winnenburg, T.K. Baldwin, M. Urban, C. Rawlings, J. Köhler, and K.E. Hammond-Kosack, PHI-base: a new database for pathogen host interactions. Nucleic Acids Res 34 (2006) D459-64.

[33] D.K. Button, K.M. Gartland, L.D. Ball, L. Natanson, J.S. Gartland, and G.D. Lyon, DRASTIC--INSIGHTS: querying information in a plant gene expression database. Nucleic Acids Res 34 (2006) D712-6.

[34] A. Bairoch, The ENZYME database in 2000. Nucleic Acids Res 28 (2000) 304-5.

[35] C.E. Lipscomb, Medical Subject Headings (MeSH). Bull Med Libr Assoc 88 (2000) 265-6.

[36] T. Greenhalgh, How to read a paper. The Medline database. Bmj 315 (1997) 180-3.

[37] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M.R. Pocock, A. Wipat, and P. Li, Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics 20 (2004) 3045-54.