XMLpipeDB:
Relational Database Schemas
Piped From
XML Descriptions
In Near-real Time

User's Manual
Tutorial Documentation
Release Instructions

BIOL 498 / CMSI 698
Loyola Marymount University
Spring Semester, 2006

XMLpipeDB:

RDB Schemas Piped From XML Descriptions

Special Studies: Bioinformatics
BIOL 498 / CMSI 698 – Spring 2006

Kam Dahlquist, PhD., John Dionisio, PhD.,
J. Barrett, J. Boyle, A. Carasso, D. Hoffman, B. Johnson, B. Naffas, J. Nicholas, R. Ruiz, S. Spicer
Loyola Marymount University

1. Introduction

Since the completion of the human genome sequencing effort, the amount of data that is available to researchers has skyrocketed. Not only has the number of data banks containing the information increased, but the amount of data that is stored in those databases has grown exponentially. There are now databanks containing information for all facets of gene investigation, protein sequencing, viral investigations, and many other topics. The sheer volume of data, along with the sophisticated tools required to manipulate, store, retrieve, study, and understand that data, requires extensive application of technology. Hence, the emergence of *Bioinformatics*.

Bioinformatics is the application of Information Technology to biological data. This information technology provides the ability to represent, organize, manipulate, distribute, maintain, and use the data. It is an interdisciplinary scientific endeavor, comprised primarily of researchers in biology and computer science, but incorporating mathematics, chemistry, physics, engineering, and other disciplines as well.

The ultimate goal of Bioinformatics is to allow scientists to predict the traits of an organism and the organism's response to its environment based only on its genome sequence. But there is a problem, namely, the veritable deluge of data being produced. How does one begin to analyze data from thousands of genes, taken over multiple time points, that are stored in dozens of data banks all over the world?

1.1. The Problem of Storage

The obvious solution, and the realistic application, is the use of the Internet. Almost every databank has internet connectivity, allowing users to access and download information. Most have methods to upload new information to add to the knowledge base, with different methods of verification for this data. This causes a number of problems.

First, there is the difference between the underlying structure, or *schema*, of the database itself. Although a certain amount of unification of structure has been achieved, owing to de-facto standardization, there is still a great amount of variability.

Second, there are problems created by the normal updating process of the stored data in these databases. Since new data is made available frequently, it is possible that previously downloaded data may be obsolete, or even proved to be incorrect. While frequent updating is good for the databank that contains all the latest data, it is difficult for its subscribers to keep up. In addition, these updates may require a data bank to modify its format or its schema, to change locations to a different Web address altogether, or even to disappear completely.

Third, there is significance to the effect of having different methods of verification of the data itself, a process known as *curation*. Some data obtained experimentally, while other data are obtained from other data. The validation of these data, whether done by hand or by machine, is critical to the correctness of the data. The computer science principle of "garbage in = garbage out" applies heavily here.

Finally, there are issues of interoperability between different database engines and different operating system platforms, as well as issues with funding of the projects involved.

These problems add several layers of complexity to an already complex problem. We believe the researcher should be able to concentrate on her research, rather than having to be bogged down with the "computer science" of the underlying computer application tools she is using.

1.2. The Unifying Solution

Modern internet technology can solve these problems by allowing the researcher to install and maintain copies of the data of interest on local computers. The data can be downloaded from any of a wide range of sources, usually for free or for a nominal cost. With proper application of computer technologies, this local data can be stored in a unified manner, so that the differing schemas of the various data banks can be unified coherently on the local storage. By unifying the differing database descriptions, the researcher is provided with the ability to search, compare, and perform various other manipulations on the data without regard to the differing storage designs of the native databank from which each data set has been extracted. The "catch" to this method is that the researcher must come up with the local storage design, and must consider the computer science behind the research, which is not an effective use of their talents or valuable and expensive research time.

The current bleeding edge of information systems is a confluence of relational database management systems, object-oriented programming, and XML. Learning curves for each of these technologies can be steep (depending on what you already know) — but for now we'll talk about them breadth-first: figure out how everything fits together and connects first, then proceed with learning more about each component individually… ultimately, we want to come up with an actual, living, breathing piece of software.

Currently, Relational databases are the prevalent standard for storing, managing, and querying large amounts of data, while Object-oriented programming is the prevalent paradigm for developing applications (client or server, Web or non-Web). Additionally, XML is gaining momentum as a "common ground" for sharing information. However, the relational model is not the same as the object model, which in turn differs from the XML model, even though *all three models are capable of holding the same information*. All three of these models are currently in use for storing genomic, proteomic, and other information, which further complicates the unification effort.

One solution to unification is to develop an object-oriented database or define a "serialized" object format — but the reality is that each paradigm has its strengths and a momentum of its own. A second approach is to *manually map* information across these boundaries — but this is tedious and errorprone, and leads to software that may not be reusable in all cases. Therefore, the current thinking for the unification effort is to automate the process at all possible levels. This

Demonstrate how important it is that research be performed on the data without having to worry about the underlying application. Show how our application allows the researcher to focus on the information rather than the application.

1.3. Document Organization

The remainder of this document is organized as follows. Section 2 contains a tutorial about the XMLpipeDB processing. Section 3 contains a User's Guide, including all installation instructions and an example database translation case. Section 4 presents contact information, links to data banks, and other supporting information.

2. Tutorial

The following is a brief tutorial on the XMLpipeDB translation process, and how to use the program to create a translated database. We will then describe the process of how to translate and store data, and how to access that stored data.

2.1. XMLpipeDB Translation Process

There are two key technologies that present themselves for the automated translation process. First, the E**X**tensible **M**arkup **L**anguage (XML) standard includes mechanisms that can *specify* the structure of a given XML file, using Document Type Definition (DTD) and/or XML Schema Definition (XSD) technologies. Second, techniques and conventions have been developed for translating a relational structure, as defined in a DBMS, into an object structure for use with object-oriented technologies — an activity called *object-relational mapping (*ORM). The interrelationship of these paradigms is shown in Figure 1.

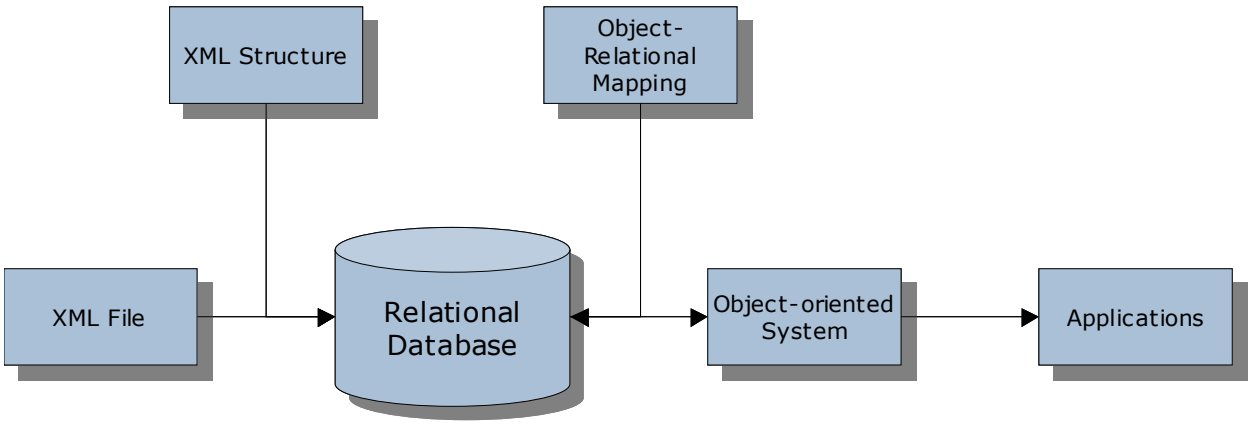

Illustration 1: Object Relational Mapping - Basic System

Originally, the XML standard provided for a DTD file, which specified the structure and content for different types of XML files. This standard has since evolved into XML Schema, using the *XML schema definition* (XSD) format which is itself in XML. XML files can specify the DTD or XSD to which they conform; validators have been developed to compare a given file against a DTD or XSD to see whether that file "complies" with the claimed structure. One can make an observation that an XSD is analogous to defining a *class* in an object-oriented programming language; thus, an individual XML file can be viewed as containing one or more *instances* of the class defined by an XSD. This observation has manifested

itself in the existence of programs that reads an XSD file and *generates* Java classes that correspond to that file — Castor, JAXB, XGen, XMLBeans, and many more. It is this technique that provides XMLpipeDB with its translational engine. Figure 2 shows the resulting process as a set of blocks describing the target architecture.

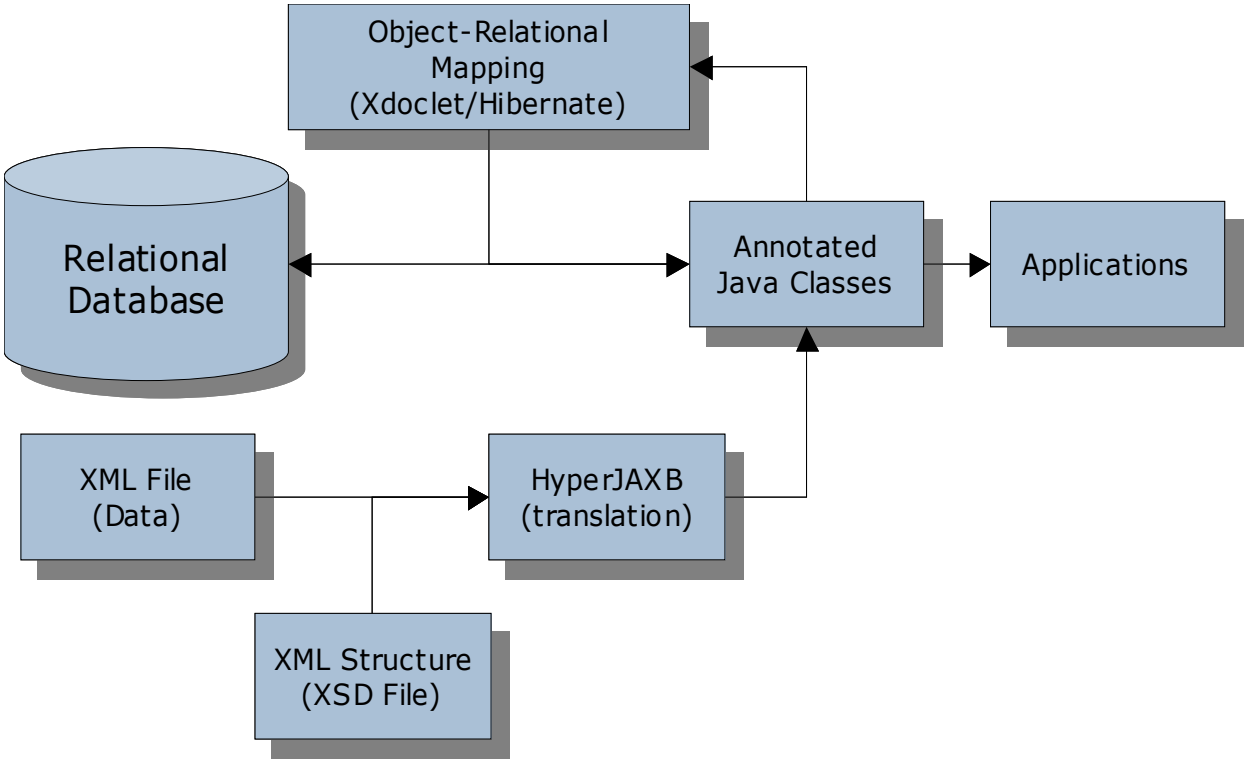

Illustration 2: XMLpipeDB Target Architecture

So, our flow changes slightly: we still want to load XML files into a relational database, but instead of direct loading, we represent the XML data as objects and classes first, and *then* store the objects in the database. Note how the object-oriented system becomes the unifying representation; it thus serves as the *conceptual* or *canonical data model* for the overall system. Finally, since all these technologies have existing tools based on the Java Programming Language, access to the bottom layer (the Relational Database Management System engine) can be achieved through use of any RDBMS that has a JDBC driver — a standard Java interface to a relational database.

Thus, in XMLpipeDB, there are three phases of the translation process; database definition, database loading, and data retrieval. Database definition consists of the steps of describing the remote data, translating that description into objects, and creating the local database. Database loading consists of retrieving the remote data in XML format, then using the translating description from step one to load the data into the local database. Database retrieval comes into play when recovering the stored data from the local database. The following sections describe all three phases of this process.

2.2. Using XMLpipeDB to Create a Database

The database definition process begins with finding the data required for your research. This might be contained in any of the large number of genomic databanks currently available (major databanks are listed in Section 4). Many of these repositories include links to files that describe the database in XSD format. Downloading the XSD file provides the description needed to create the object-relational mapping and the annotated Java classes. This XSD file is place in a specific directory for the project, so that the HyperJAXB translator can find it in the next step of the process.

When the XSD file is downloaded and installed, an automated processing utility can be run to invoke the translator utility. This process will read the XSD file, along with other supporting data supplied with XMLpipeDB, and will generate the annotated Java Classes required to properly handle the actual gemonic data, processing which is part of the next step.

2.3. Using XMLpipeDB to Translate and Store Data

Once the database translatoin process is complete, the download and storage of data can begin. The XML file which matches the XSD description file is downloaded and placed in the same directory as in the previous step. Annotated classes generated in the previous step are used with the data to generate relational objects that are stored in the RDBMS, and – *voíla* – automated data translation and storage is complete.

2.4. Accessing the Stored Data

The final step is to access the data that has been stored.

3. Example Database Translation

This section provides an example using a databank entry from the UniProt Protein Data Bank.

3.1. Assumptions / Requirements

3.2. Obtaining Required Files

There are a number

3.3. Installation of the local Database Engine

3.4. Installation of the Eclipse Integrated Development Environment

3.5. Installation and Configuration of the XMLpipeDB Files

3.6. Downloading of the data bank database description (XSD file)

3.7. Processing the XSD File

3.8. Downloading and Processing the Remote Data

3.9. Accessing the Local data

3.10. Customizing the Application

4. Support Information

The following URL links, papers, and books provide more information about Bioinformatics and the background of the XMLpipeDB, along with links to many of the major data banks for which this project is initially intended.

4.1. Internet Links (URLs)

NCBI: National Center for Biotechnology Information:

GenBank: DNA and protein sequence (contains same info as EMBL and DDBJ)

http://www.ncbi.nlm.nih.gov/Genbank/index.html

RefSeq: curated records from the above

http://www.ncbi.nlm.nih.gov/RefSeq/

UniGene: expressed sequence tags (ESTs)

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=unigene

Gene: one stop shop for information about a particular gene/protein

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=gene

PubMed: biomedical literature

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?DB=pubmed

OMIM/OMIA: Online Mendelian Inheritance in Man

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM

Taxonomy: hierarchy of organism nomenclature

http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM

BLAST: not a database, but useful tool

http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/information3.html

UniProt (formerly SwissProt): hand-curated protein sequence database

http://www.uniprot.org

TrEMBL: translated EMBL, not curated

http://www.expasy.org/sprot/

PDB: Protein Structure Database

http://www.rcsb.org/pdb/Welcome.do

Pfam: protein families

http://www.sanger.ac.uk/Software/Pfam/

Gene Ontology (GO): structured, controlled vocabularies and classifications widely used in genome annotation

http://www.geneontology.org/

Model Organism Databases

Saccharomyces Genome Database (SGD)

http://www.yeastgenome.org/

Mouse Genome Informatics (MGI)

http://www.informatics.jax.org/

WormBase

http://www.wormbase.org/

FlyBase

http://flybase.bio.indiana.edu/

TAIR (The *Arabidopsis* Information Resource)

http://www.arabidopsis.org/

TIGR (The Institute for Genome Research) hundreds of microbial genomes

http://www.tigr.org/

Human Genome Browser (UC Santa Cruz)

http://genome.ucsc.edu/

KEGG: metabolic pathways

http://www.genome.jp/kegg/

BioCyc: family of metabolic pathway databases

http://www.biocyc.org/

EcoCyc

http://ecocyc.org/

HumanCyc

http://humancyc.org/

4.2. Books

4.3. Interesting Papers