Overcoming the limitations of PDB format

Shifali Mehta
University College of Engineering
Punjabi University, Patiala
mehta.shifali@yahoo.co.in

Amardeep Singh
Professor
University College of Engineering
Punjabi University, Patiala
Amardeepdhiman@gmail.com

ABSTRACT

The Protein Data Bank is a repository for the 3-D structural data of large biological molecules, such as proteins and nucleic acid. The PDB is a key resource in the areas of structural biology, structural genomics. Most major scientific journals, and some funding agencies, such as the NIH in the USA, now require scientists to submit their structure data to the PDB. If the contents of the PDB are thought of as primary data, there are hundreds of derived databases that categorize the data differently. For example, both SCOP and CATH categorize structure according to type of structure and assumed evolutionary relations; GO categorize structures based on genes. In this paper, we will describe how to overcome the limitations of PDB format.

Keywords: PDB, XML, XCML.

INTRODUCTION:

Protein Data Bank

The Protein Data Bank was established at Brookhaven National Laboratories in 1971 as an archive for biological macromolecular crystal structures. In the beginning the archive held seven structures, and with each year a handful more were deposited. In the 1980's the number of deposited structures began to increase dramatically. This was due to the improved technology for all aspects of the crystallographic process, the addition of structures determined by the nuclear magnetic resonance (NMR) methods, and changes in the community views about data sharing.

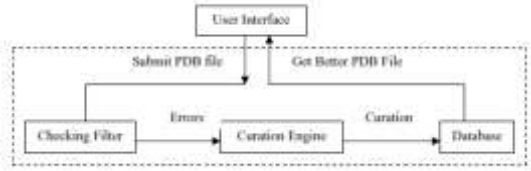
PDB Data Curation

It is well known that the human genome project has resulted in rapidly growing databases in bioinformatics. And bioinformatics provide some opportunities to develop novel data analysis method, but there are still some challenges including protein structure prediction, genomic sequence analysis, etc. For example in protein structure prediction, scientists are using PDB file to apply amino acid sequence to determine the secondary and tertiary structure of proteins.

Unfortunately, PDB supports several different formats, thus it suffers from inconsistencies in how the data formats are constructed over time. PDB format is old, ambiguous and inadequate, but it is still the most widely used format since all relevant software can read it. Although some users claim they do not mind the errors of PDB file, which may result in the partial or wrong information to affect the accuracy of derived and propagated data. Thus it is very necessary to develop a system to curate data in PDB file, to provide easy understanding and better services for protein scientists and computer scientists.

There are two architectures for PDB data curation : one simply uses checking filter and curation engine to curate data. The other one uses the first one by adding XCML curation part to further curate data, which can deeply clean the PDB data by using more constraints. These two tools can be used not only for cleaning existing PDB files and also for creating new PDB files.

Figure1. PDB Data curation system architecture



The above picture shows the architecture for PDB data curation system. From user interface, checking filter gets PDB file, which checks the error and reports them to curation engine, and then curated data by curation engine are sent to database storage. The users can get better PDB file from our database after this process.

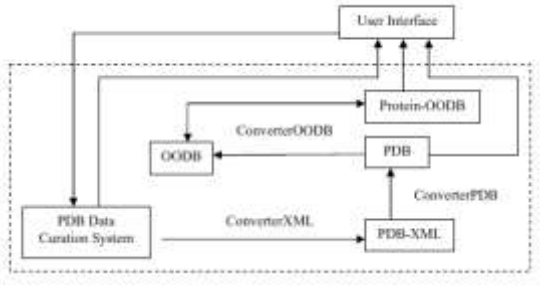
Most errors are fixed in original PDB file after it is through this PDB data curation system. But however, still some issues related to interoperability with the latest technology are still left, which may again can create errors.

Architecture of PDB Data Curation System with XCML

The Extensible Markup language (XML) is self describing, and is standardized by the World Wide Web Consortium (W3C). It is human and machine readable, flexible, extensible and has already became the standard format for exchanging the information through the network. It usually uses the XML schema or document type definition (DTD) to define syntax and data types. But it can not verify semantic constraints, avoid erroneous data and is domain dependent. Therefore we will not get any better data if we just convert PDB file into XML format. In order to avoid disadvantages of XML and make use of XML's metrics to get better data, we apply an XML constraint language, extensible constraint markup language (XCML), which is more powerful.

The following architecture shows how to improve our PDB data curation system by using XCML. Firstly PDB file is submitted to PDB data curation system, after curation, PDB file can be converted into XCML format by converter XML, which uses some constraints to further curate. PDB-XML can be in protein-OODB format after it is handled by converter PDB and converter OODB. At this moment users can get better data that are in domain specific objet-oriented format.

Figure2. Below diagram shows the architecture of PDB Data curation system.



Major issues of PDB data

Data Identification: PDB users usually want to know who generates the data, which paper, which lab, which date so that users can identify the confidence of PDB data. In PDB data generation, but some users may change the spelling without any intention, such as changing AUTHOR to AUTHRO, TITLE to ITTLE. Therefore our system is designed to automatically check those errors and provide users the authority to change them. In addition, our data curation system also gives information about missing data such as no HEADER, no AUTHOR, and no DATE and so on. Since header of PDB file is very important that contains information's about the original literature citation, full names of ligands etc, it is very necessary to include it in PDB file. Author, paper and lab give us information about respects of different data resources which can confirm the confidence of data. Also date in PDB file headers show the time that the atomic coordinates were deposited, modified and published at the protein data bank, which helps keep data up-to-date. Therefore identifying these data will give users and helpful information.

Data Errors: Data Errors are generated when users change the data. The size of PDB file is very huge such that the users who are using them may remove or change some data without any intention. We store the original data and changed parts of PDB file in our storage to avoid this kind of errors. If users have changed data, the system sends alert signal which can avoid users to change data without any attention. But if users really want to change data, our system compares the current file and previous file to avoid the users to make mistake. The system will suggest users to use the previous data file if it finds the errors from new data file.

Redundant Data: PDB data may be redundant which wastes storage and user's time, therefore our system provides filter to automatically or manually check data redundancy. The system can remove redundant data without any loss of accuracy of data to make sure that data file does not have repeatable data by using following method- the system firstly checks REMARK part of PDB file and sees whether there is redundant data, if it has the system will compare protein sequence and atomic structure to get rid of duplicated part in protein sequence by applying structures matching method.

Ambiguous Data: PDB data may have same name for different objects. It can be fixed by defining those ambiguous side chain atoms in a hash table which lists all different them when they appear in protein structures. In addition, the are difficult to check. But we can check them by saving all corresponding objects for each name in a table. We manually check them according to the protein structures, if they are not the same object, we will give them the different names.

Problem with Protein data bank

With the passage of time, numerous research initiatives are coming to a level of maturity, in which the biological data size have become exponentially big as well as leading to data inconsistency problems like Formatting, Ambiguous data, Data identification, Data Errors, Redundant data, interoperability of the legacy biological data with the current technologies. Therefore there is an urgent need to overcome these issues. So we need to develop a framework which can to bridge the connectivity of these old legacy database technologies with new contemporary technologies like DB4o.

Since, the PDB file needs a special text processing functions for its extraction of information for usage in bioinformatics applications, there is a need to reduce the need for more text processing and have a simple framework in which the queries are simple to process and data can be represented as objects for its maximum maintainability.

The text processing required to process the PDB file takes extra efforts and resources, For example in certain area the coordinate calues get merged with each other and it is very difficult to process and get separate coordinate values based on white space and delimiters. So we need to overcome these typical issues and move ahead with contemporary technology integration.

Also it is not clear whether the residue name is 2, 3 or 5 digit or alphabet, they also get merge. It is also difficult to process and get separate value of residue as it gets merged with the next column value.

DB4o

DB4o is an embeddable open source object database for JAVA and .NET developers. DB4o is written in JAVA and .NET and provide the respective API's. DB4o can run on any operating system that supports JAVA and .NET.

DB4o represents an object-oriented data base model. One of its main goal is to provide an easy and native interface to persistence for object-oriented languages. Development with DB4o database does not require a separate data model creation; the applications class model defines the structure of the data in DB4o database. DB4o attempts to avoid the object/relational impedance mismatch by elimination the relational layer from a software project.

Features of DB4o database

One-line-of-code database

DB4o contains a function to store any object with a single command:

ObjectContainer.Store(NEW SomeClass( ));

Embeddable

DB4o is designed to be embedded in clients or other software components completely invisible to end-user. Thus DB4o needs no separate installation mechanism, but comes as just one easily deployable library with a very low footprint of ~670 kb in .NET version and around 1MB in java version.

Client-Server mode

Client/Server version allows DB4o to communicate between client and sever-side applications. DB4o uses TCP/IP for client-server communication and allows configuring port number. Communication is implemented through messaging.

Dynamic Schema Evolution

DB4o supports automatic object schema evolution for the basic class model changes. More complex class model modifications, like field name change, field type change, hierarchy move are not automated out-of-the box, but can be automated by writing small utility update program.

This feature can be viewed as an advantage over relational model, where any change in schema results in mostly manual code review and upgrade to match the schema changes. In most cases the code upgrade can not be automated as the actual query language is string based (SQL) and is not recognized by IDE auto completion and code generation tools like intellisense.

Native Queries

Rather than using string-based API's (such as SQL, OQL), programming language itself to access the database and thus avoid a constant, productively-reducing context switch between programming language and data access API.

LINQ

LINQ support is fully integrated in DB4o for .NET version 3.5. LINQ allows creating object-oriented queries of any complexity with the benefit of any complexity with the benefit of compile time checking, IDE intellisence integration and automated refactoring.

Due to integration with some open-source libraries DB4o allows optimized LINQ queries on compact framework.

LINQ can be used both against relational and object data storage, thus providing a bridge between them, which can be valued for projects using both technologies, or for project migration between the two. It can also be used as an abstraction layer, allowing to easily switch the underlying database technology.

Portability and cross-platform deployment

DB4o was successfully tested on java FX and Silverlight.

DB4o runs out of the box on Android.

DB4o uses a custom feature called "generic reflector" to represent class information, when class defamations are not available, which allows to use it in a mixed java-.NET environment, for example java client-.NET server and vice-versa.

Documentation and Support

DB4o provides various sources of documentation: tutorial, reference documentation, API documentation, online pair casts and blogs.

Object Manager

Object Manager Enterprise (OME) is a DB4o database browsing tool. Object Manager Enterprise allows to browse classes and object in the database, connect to a database server, build queries using drag and drop, view database statistics etc.

In addition to graphic interface to DB4o database OME provides some administrative functions are:

Indexing

Defragmentation

Backup

Community

Over the years the community of DB4o registered members has grown to over 60,000 members. Many important DB4o related projects, such as standalone project manager, encryption support, Mono support etc, are fully driven by community members.

DB4o provides free access to its code, documentation, forums and releases to the community members. From time to time, DB4o holds different contests allowing the community members to come up with the best suggestion for an improvement of a specific DB4o aspect, which are later on integrated into the core code.

Conclusion

From the above discussion it is clear that by converting PDB file into DB4o, all the problematic issues of PDB file such as Data identification, Data Errors, Redundant data and ambiguous data can be avoided. So better version of PDB file is obtained after this conversion.

Future Work

For future directions we suggest that one should study more biological databases, which need attention sue to scope of their research and applications.

They also need high level of text processing before they can be used and require some bridging technology which can help them to be utilized with contemporary technologies like cloud applications. Therefore for future we suggest that work should be done to develop such conversion and bridging technologies which would help other legacy data sets to be used efficiently and high usage value.

REFERENCES:

[1] Christine Kehyayan(2008), "*Evolutionary algorithm for Protein Structure Prediction"* , International Conference on Advanced computer theory and Engineering.

[2] Gerard J.Kleywegt, Mark R. Harris. (2004), "*The Uppsala Electron-Density Server"*

[3] PROTEIN DATA BANK http://en.wikipedia.org/wiki/Protein_Data-Bank

[4] DB40 http://en.wikipedia.org/wiki/Db4o

[5] Yancho Wang (2006), "*PDB data curation",* Proceedings of the 28th IEEE EMBS international conference New York City, USA, Aug 30-sept 3, 2006

[6] T.N Bhat, P.Bourne, Z.Feng, G.Gilliland, S.Jain, wt al "The PDB data uniformity project".Nucleic acids Research,2001, vol.29, No.1,pp214-218.

[7] P.Lord, A. Macdonald, L.Lyon, D.Giaratta, "Data Deluge to Data Curation",

[8] T.Warnock, L.V.Einde and R.Moore:NEESit Data Curation Roadmap. http://it.nees.org/documentation/pdf/TR-2005-046.pdf

[9] J.Grey, A.S.Szalay, A.R Thakar, C.Stoughton, J.vandenBerg:Online Scientific Data Curation, Publication and Archiving.

[10] P.Buneman, S. Khanna and W.Tan: Data Provenance:Some basic issues, http://db.cis.upenn.edu.DL/fsttcs.pdf