Genome Databases
Current Implementation Practices

Hitomi HASEGAWA, University of Helsinki

***Abstract*—Genome databases are a mean to provide a repository for biological information. However, modeling biological data imposes several challenges. In particular, regarding the architecture and design options in which they are based.**

In this paper, we describe the major currently available implementation possibilities and describe their advantages and disadvantages. In addition we also evaluate some of the main databases that support the human genome project

***Index Terms*—architecture, databases, genome, object oriented, relational**

I. Introduction

THE human genome refers to the complete set of genes required to create a human being [14]. The goal of the Human Genome Project (HGP) is to obtain the complete sequence of the 3-4 billion nucleotides that comprise the human genome. The approximate number of genes spread in the 23 human chromosomes is not yet completely understood and has changed over time. Likewise, not all of the genes are known. To date, the human genome sequence is close to being complete. The DNA sequence itself does not reveal the secrets of human life. However, with the use of other data we can understand questions in other fields such as genetics, biochemistry, and medicine. The availability of the human genome sequence will provide a framework to which a large variety of human data can be integrated, such as gene structure, variation and disease.

In addition to the human genome, many other organisms have been or are currently being sequenced. All of these sequences are stored in electronic repositories. The challenge of storing genetic data is that it is complex and multilayered. Such information can be structural, functional, clinically-based, population-based, gene-based, etc. Moreover, it includes a large number of different data types. For example, the data that describes the DNA structure is composed of other elements including: DNA, sequence/nucleotide bases, nucleotide position, chromosomal assignment, gene locus, gene organization, messenger RNA, transfer RNA, ribosomal RNA and map position. Furthermore, each of these elements can be broken down into another set of complex data types. In addition, many of the different elements have interactions with each other and can as well belong to certain clinical or geographical populations. As a result, the organization of this large amount of data is very complex.

In this paper, we describe some of the major currently available implementation possibilities to develop such repositories (databases). In addition, we also evaluate some of the main databases that support the human genome project.

The remainder of this paper is organized as follow. Section II describes the different implementation approaches, namely flat-file, XML, relational and object oriented. In Section III we describe some of the main databases supporting the human genome project such as GenBank, Genome Database, OMIM and AceDB. In Section IV we describe the experiences of Shamkant's project [14], which implemented the same biological database using three different approaches. Finally, in Section V we summarize the paper and provide conclusions.

II. Implementation Approaches

This section provides an overview of the current models and approaches used to implement biological databases.

A. Flat File

A flat database is a database designed around a single table. With this approach all the information is gathered in one single large table with fields to represent each parameter. The flat file commonly specifies each record in a single line. The fields of each record can be delimited in different ways, such as by using commas, tabulator, whitespaces or other characters.

Whilst the use of a flat file can be easy to implement initially, it is usually prone to errors and data corruption. The reason is that, by using a single large table, there might be multiple entries with duplicate data. Furthermore, the process of merging or integrating multiple flat files can be very difficult. In particular, because it is very possible there are multiple fields with duplicate data.

Despite the limitations of flat-files, there are many biological databases based on proprietary flat files. Therefore, the integration of this kind of databases requires parsers for different flat-file formats. A parser breaks data into smaller elements, according to a set of rules that describe its structure. The development and maintenance of database specific file parsers is a non trivial and time consuming task. This particularly affects large scale integration efforts. In general, flat files are parsed into XML representations, which are more

Manuscript received February 28, 2008

H. Hasegawa is with University of Helsinki, FINLAND (corresponding author e-mail: hitomi.hasegawa@helsinki.fi).

flexible. Figure 1 shows an example of a flat-file from GenBank.[10]

```
LOCUS       BA000030             9025608 bp    DNA    ...
DEFINITION  Streptomyces avermitilis genomic DNA, com...
ACCESSION   BA000030
VERSION     BA000030.1  GI:47118309
KEYWORDS    .
SOURCE      Streptomyces avermitilis MA-4680
  ORGANISM  Streptomyces avermitilis MA-4680
            Bacteria; Actinobacteria; Actinobacterida...
            Streptomycineae; Streptomycetaceae; Strep...
REFERENCE   1
  AUTHORS   Omura,S., Ikeda,H., Ishikawa,J., Hanamoto...
            Shinose,M., Takahashi,Y., Horikawa,H., Na...
            Kikuchi,H., Shiba,T., Sakaki,Y. and Hatto...
            Sakaki,Y. and Hattori,M.
  TITLE     Direct Submission
  JOURNAL   Submitted (29-MAR-2002) Director-General ...
...
COMMENT     This work was done in collaboration with ...
            Ishikawa(*2), Akiharu Hanamoto(*3), Chigu...
            Shinose(*3), Hiroshi Horikawa(*4), Hideka...
            Osonoe(*4), Norihiro Kushida(*4), Hisashi...
...
//
LOCUS       BA000013              351911 bp    DNA    ...
DEFINITION  Mesorhizobium loti plasmid pMLa DNA, cpmp...
            strain:MAFF303099.
...
```

Figure 1. Excerpt from a GenBank Flat-File

B. XML

A different kind of flat file database can be implemented by using eXtensive Markup Language (XML).However, it must be noted that even though a single XML file can contain all the database records, its structure is standardized opposed to an ordinary flat file.

XML is a popular way of storing data in plain text files. However, the main advantage of using XML is that it supports the use of complex nested data structures and contains the definition of the data. Moreover, XML is very flexible and additional definitions and tags can be added for each particular need.

Even though XML is not necessarily as robust as other models, it is widely used in web based applications. In addition, some database implementations based on other models (e.g. Relational) accept XML data as input and are able to render and map it into their native format.

XML database have several key benefits. They are scalable, fast to access and reliable. However, converting to and from XML to support other database models can result in mismatches between the original XML structure and the resulting table. By implementing the database in XML, this problem can be avoided.[16]

In addition to the benefits mentioned above, new approaches for exploiting the flexibility of XML have been developed. For instance, by using wrappers data access and manipulation can be improved. [7]

The data organization in native XML data storages is shown in Figure 2**.** The main elements are an interface to map the particular application to the underlying framework, and a storage manager to manage data access for querying or updating. [16]

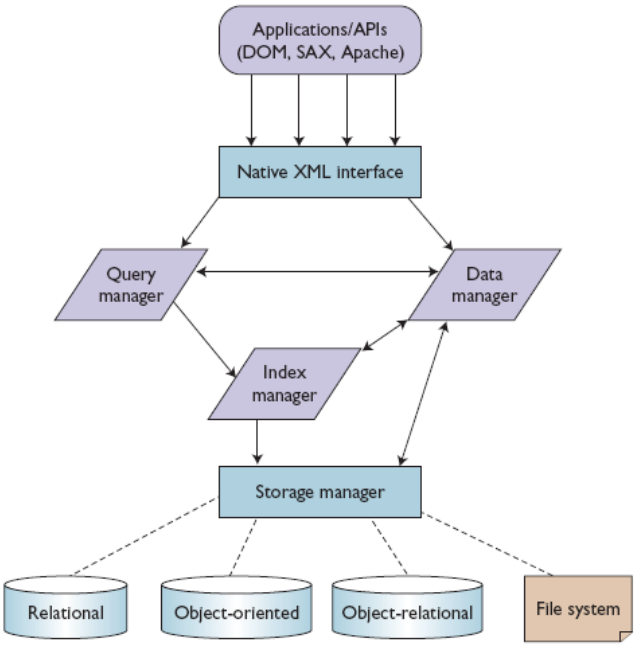

Figure 2. XML Data Management Framework [16]

C. Relational

The entity relation is an abstract conceptual representation of structured data. It was developed in 1976 by Chen. By the use of entity relationship diagrams, relational databases can be modeled.[4]

Relations in this model are defined by entities (nouns) and the relations between them (verbs) (see Figure 3). Entities and relationships can have attributes. In addition, every entity must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.

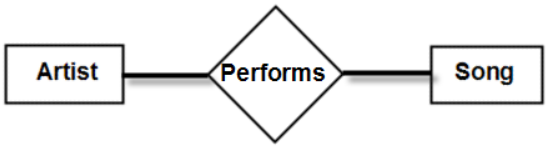

Figure 3. Two Related Entities

According to [5], the entity relation model provides additional semantic information that is neither explicit nor available in the relational model. The principle of the relational model is that any table is a relation (see Figure 4). The entity relation model adds the links that exist between different entities. [5]

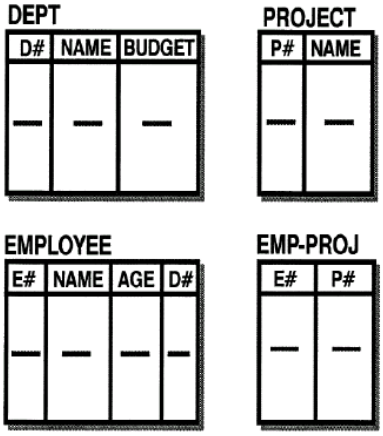

Figure 4. Relational Model of Data [5]

D. *Object Oriented*

In an object oriented approach, a database is a collection of objects. Each object represents a perception (instance) of an abstract or concrete entity in the real world. Objects having the same properties are grouped together into classes, and the same object could belong to multiple classes. For example, Person is a class. An object is related to its class via the instance-of relationship. [11]

The properties of objects are specified via the attributes of the class they derive from. An attribute maps objects in the class to values or to other objects. The later are called references. The attributes can be scalars or sets. Scalars can be defined or undefined, and might also be unique. For example, class Student has attributes *student id* and *major,* where *student id* is unique and *major* is a reference to class Department. The class Department has attributes *college* and *courses,* where college is a scalar, and *courses* is a set. [11]

Classes are related with a subclass relationship, which also forms hierarchies. Hierarchies also provide inherited attributes. That is, an object that derives from a sub class will inherit all the attributes of the parent class. Figure 5 shows the concept of inheritance.

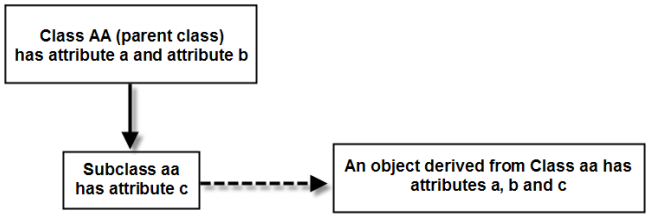

Figure 5. Inheritance Concept

In object oriented databases, every object has a unique object identifier. Two classes have the same key if they share a subclass, and the key of a subclass is the same as the key of its parent classes. Different classes could have attributes with the same name, but attributes of the same class have different names. As a consequence, explicit attributes of a class are not explicit attributes of its subclasses, and two classes with attributes of the same name do not share subclasses.

For example, dob is an explicit attribute of Person and an implicit attribute of Student. Grad is an explicit subclass of Student and an implicit subclass of Person. The key of Person, Student, and Teacher is name."[11]

It is possible to migrate relational databases to object oriented designs. Even though it is not easy, there are a number of papers describing how to do as well as actual examples of such tasks. [1] [12]

According to [13], even though the benefits of deploying object oriented databases are evident, due to the immaturity of commercial products it had a rough start and lacked adoption. Therefore, many opted for implementing based on the relational model. Finally, Table 1 shows an overview of the four approaches presented.

Table 1. Storage Approaches [16]

Framework	Storage Format	Advantages	Disadvantages
Flat-File	ASCII files stored in the file system or database management system	Easy implementation and suitable for small XML sets	Accessing and updating are difficult
XML	Ad-hoc data models or typical database models	Flexibility and improved access performance	Less mature than conventional database management systems
Relational	Tables	Scalability, reliability and easy implementation	Might require many joins to access relevant data
Object Oriented	Tables and objects	Easy implementation and abstract data type support	Document Factorization

III. Genome Databases

This section describes some of the main biological databases supporting the human genome project as well as their implementation design.

A. *GenBank*

GenBank is a sequence database which contains published and unpublished DNA and RNA sequences as well as bibliographical information. As of 2005, GenBank had over 52 million sequences stored [6]. It is the main genome database in the world and it is maintained by the National Center for Biotechnology Information (NCBI). Originally, it was a repository for sequence data, but it was expanded to include EST data, protein sequence data, 3-D protein structure, taxonomy, and links to the biological literature (MEDLINE). [14] The structure of GenBank consists of four main entities: physical context data, functional context data, features data and bibliographical data. [3]

The system is a combination of flat-files, relational databases and files containing ASN.1 structures [14]. ASN.1 is a formal notation for describing messages to be exchanged by

telecommunication protocols, including Internet [2]. For the relational databases, GenBank is implemented using Sybase Relational Database Management System (RDBMS). Table 2 shows a simplified GenBank entity relation schema.

Table 2. The Entities and Relationships Under Each GenBank Object [3]

Objects	Component Objects				
	Entities			Relationships	
Physical context	Sequence Source Entry	Text Taxonomy Taxlevel	Gencode	Seqel Secacc Nathost	
Functional context	Gene Region	Product		Genreg Genocc	Genprod
Features	Featocc	Featqual	Featkey	Qualval	Compfeat
Biblio-graphic	Reference Person Submission	Publication Scan Address	Keyword Comment Other Entities	Relink Authref Refstat	Edpub Keylink Comlink

Average users are unable to access the structure of the database directly for querying or other functions, most likely for security reasons, such as SQL injection [9]. Instead, queries are performed via the Entrez application or web interface. Entrez allows searches via keyword, sequences or GenBank UIDs. [14]

B. The Genome Database (GDB)

The Human Genome Database is a catalog of human gene mapping data that supports biomedical research, clinical practice, and professional and scientific education by providing human genome mapping through GDB and genetic disease information through OMIM. It is managed by the John Hopkins University School of Medicine and the William H. Welch Medical Library. Mapping can be classified into two types, genetic and physical maps. Mapping is used to determine the relative position and distance of genes or identifiable landmarks on the chromosomes. [3][14][15]

The gene mapping data is used to associate a piece of information with a particular location on the human genome. The GDB data includes map information (distance and confidence limits) and PCR probe data (experimental conditions, PCR primers and reagents used). GDB is maintained as a relational database. The data are in many different tables which represent nine primary data objects (loci, probes, maps, polymorphisms, mutations, cell lines, libraries, citations, contacts). There are also links to the Enzyme Data Bank via EC numbers and the Genome Sequence Data Bank (GSDB) via DNA sequence accession numbers. [14][15]

GDB also allows users to search information in OMIM through a direct searching link using MIM numbers. The sources for GDB are primary books, journals, and direct submissions by human genome mapping committees for individual chromosomes. [3]

The GDB is implemented in Sybase RDBMS. Since it covers a very large amount of biological data, it actually comprises of four independent relational databases. The databases are *gdb, gdb_admin, gdb_aux_db,* and *gdb_project.* The gdb database contains biological information, references and certain administrative information. Table 3 shows a simplified entity relationship schema diagram for the Map components. [3]

Table 3. The Components of Map [3]

Object	Component Objects		
	Entities	Relationships	
Map	Contig element Info	Order links	Consens compon ref
	Contig set info	Order link dist dict	Contact contig red
	Dist unit dict	Order sets	Elem contains ref
	Linkage map set info	Order sets end	Map method valid ref
	Map method qual dict	Order sets dead	Method order set ref
	Map strings	Order set flat map	
	Map cumm lines	Order set loc mods	
	Order class dict	Order set method dict	
	Order elements	Order set type dict	
	Order elem set orient	Orient dict	

C. Online Mendelian Inheritance in Man (OMIM)

"OMIM is the online version of the human genetics database Mendelian Inheritance in Man, serving clinical medicine and the Human Genome Project. It is a comprehensive catalog of human genes and genetic disorders with full text annotations on current genetic research and clinical disorders."[15] It was initially administered by John Hopkins University, but later it was transferred to the NCBI. [14]

OMIM is divided into six sections: MIM Number, Title, Text, Allelic Variants, References, Clinical Synopsis, and Edit History. The OMIM database contains information on genetic disorders and traits. It covers five disease areas based on organs and systems: endocrine / hematology, immunology, connective tissue / skin and appendages / craniofacial / ear, physochological / neurological / muscular, cardiac / gastrointestinal / pulmonary / renal / genital, and inborn errors of metabolism. [14]

The entire database is a ASN.1 flat-file/relational format. In its initial full-text form, all these information was not easily accessible via search engines, and only a limited number of links between mapping and disease data were available. The full text entries were converted to ASN.1 structured format when OMIM was transferred to the NCBI. This change improved greatly the ability to link OMIM to other databases as well as give a structure to the data. OMIM is currently under editorial revision and some documents are presented in a restructured format instead of the traditional long, "flat" text. Since this database is updated daily, the entries may differ from the most recently published version of the OMIM book. [14][15]

D. AceDB-based Genome Databases

AceDB refers to two separate things, either a database model, or to the C. elegans database. The reason is that the acronym is used for the Caenorhabditis elegans Database and also to the

open software object oriented database model. AceDB is based on open software and has been adapted by many groups to organize molecular biology data about the genomes of diverse species. AceDB allows automatic cross-referencing of items and allows for hypertextual navigation of the links using a graphical user interface and mouse. Additionally, the software has graphical user interfaces. [15]

"ACeDB databases are available for the following genomes: C. elegans, Human Chromosome 2 1, Human Chromosome X, Drosophila melanogaster, Mycobacteria, Neurospora crassa, Saccharomyces cerevisiae, Schizosaccharomyces pombe, Arabidopsis, beans, Chlamydomonas reinhardtii, cotton, grains, maize, rice, Solanaceae, sorghum, soybeans, and forest trees, cattle, chicken, pig, and sheep." [15]

AceDB provides a generic object oriented database management system (OODBMS) supporting routines for dealing with DNA and protein sequences, genome maps, and other biological entities. AceDB is suitable for medium sized projects and those requiring rapid development. [8]

AceDB can run in two modes. In the standalone mode the software reads and writes directly to local files. Database file permissions are administered via the Windows or Unix system. Hence, multiple users can read the files simultaneously, but only one can write them at a given time. In client server mode, multiple users can connect to the AceDB server. This mode does not restrict the write access rights to a single user.

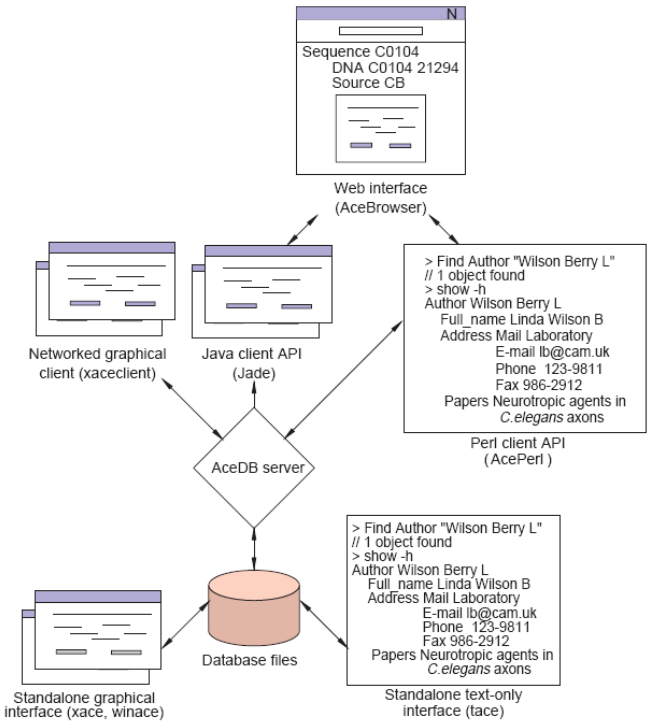

Figure 6. AceDB Architecture [8]

AceDB's data model aggregates records that closely represent biological data. For example Figure 7 shows an author object. The class from this object is created contains the relevant attributes for a bibliographic entry. "Within each object, the data is organized hierarchically into a tree either containing data cells or human-readable tags, which organize and give structure to the tree." [8]

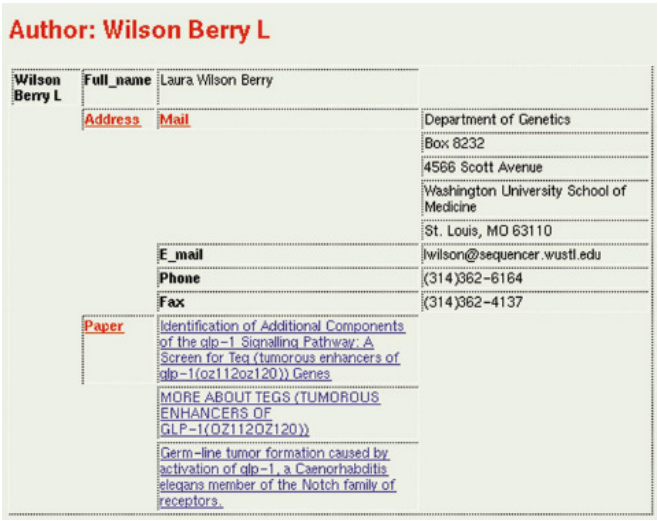

Figure 7. An AceDB Object [8]

An advantage of using objects rather than tables is that they allow multiple values. This is of particular relevance for biological data, which often has multiple values. Moreover, AceDB supports new data type definitions that can represent biological data in a more accurate way than with common data types (i.e. string, integer, double). [8]

One of the major disadvantages of AceDB is that, despite its large amount of features, it is based on open source software. Hence, it does not have the same level of support and stability that a commercial product would provide. For this reason, a trend has been to gradually replace projects based on AceDB to commercial relational systems when they are mature. At this point stability might play a more important role. [8]

Table 4 and Table 5 summarize the main characteristics of the database presented in this section.

Table 4. Summary of Major Genome Databases [14]

Database	Content	Initial Technology	Current Technology
GenBank	DNA/RNA sequence, protein	Text files	Flat-file/ASN.1
GDB	Genetic map linkage data	Flat-file	Relational
OMIM	Disease phenotypes and genotypes, etc.	Index cards/ Text files	Flat-file/ASN.1
AceDB	Genetic map linkage data, sequence data	Object oriented	Object oriented

Table 5. Summary of Problems of the Major Genome Databases [14]

Database	Database Problem Areas
GenBank	Schema browsing, schema evolution, linking to other databases
GDB	Schema expansion/evolution, complex objects, linking to other databases
OMIM	Free entries are unstructured, linking to other databasesFlat-file/ASN.1
AceDB	Schema expansion/evolution, linking to other databases

IV. Experimental Project

This section describes an experimental study carried out by Shamkant [14] which aimed at characterizing the pros and cons of implementing the same database with three different approaches. The database, known as MITOMAP, was implemented with the flat-file, relational and object-oriented approaches respectively. The model system was the mithocondrial system. This system is a representative system to the human genome because the human mithcondrial DNA was the first to be sequenced and afterwards it has been subject of many studies. Therefore, currently, there is an extensive array or mtDNA functional and comparative genetic information. Further, the analysis of mithocondrial data has continuously revealed interrelationship between the various types of genetic data. In addition, the mithocondrial genome has been completely sequenced. It contains 37 genes essential for life, it is 16,569 nucleotide pairs long and encodes all three major classes of genes; 13 polypetides, two rRNAs and 22tRNAS, respectively, as well as replication, transcription and RNA processing signals. Likewise, a wide variety of mtDNA mutations have been associated to human diseases. [14]

A. *Relational*

The relational model was the first prototype for MITOMAP. The database was designed in a normalized fashion. Thus, each row in the relational table represents a collection of related values that cannot be represented in simpler sets. By doing so, it helps eliminate data duplication problems, multiple update issues and the generation of non genuine rows resulting from *JOIN* operations. However, this also results in a higher decomposition of data into smaller tables. This is due the common *null* values in the tables. Since the amount of biological data is very large, it results in a great number of tables, which can make the database become very hard to manage very quickly. In addition, the null values also present additional problems for querying, because relational *joins* cannot be performed in across null fields. Therefore, even though the relational model allows an atomization of individual data items simpler, a new challenge is brought up in the comprehension and maintenance of the data structure as a whole from a domain knowledge perspective. Furthermore, the definition of relationships in the relational model is good for representing well defined binary relations. However, biological data does not always fit nicely into that model. Hence, the design of the database itself has to be made very carefully. Likewise, the formulation of queries requires knowledge of the structure of the database. This means that in practice, only expert database users might be able to formulate queries correctly. Unfortunately, common users probably do not fit that user profile. [14]

Modeling the database with the relational model has benefits too. If properly normalized, it guarantees the lack of anomalies in the database. In addition, the query response is fast. However, these benefits become irrelevant if the rows do not completely represents the data. Further, if the desired questions cannot be queried, the response speed is irrelevant as well. Also, modeling the data is difficult, especially for biological data. Other problems include the lack of methods for sharing schemas across tables or to link data from one table to another. The result of this prototype also required several implementations due to the inability to comprehensibly model all of the data types required for MITOMAP. [14]

B. *Object-Oriented*

The second prototype for MITOMAP used the object oriented model. This model provides benefits for biological data since it allows for more complex and direct mapping of real world concepts to a structure in the data model. With this approach, the data representation resembles the person's mental model that designed the object. This is due to the fact that objects can have different degrees of normalization, unlike the relational model. [14]

At the time of the MITOMAP implementation, the object oriented management systems were still immature. For that reason, some collection types were not available in a flexible enough manner to support the unpredictable nature of biological queries. Likewise, even though data resembled real world, many parts of the implementation had to be hard coded. Hard coding means that certain aspects of the data such as configuration and formatting are embedded within the code. Furthermore, the concept of inheritance did not always fit biological data properly. The biological classes are in many occasions totally unrelated and thus they do not benefit from inheriting attributes or methods from other classes. The object oriented model does however, allow modification of biological classes over time. This is particularly beneficial for biological data since it changes over time. [14]

As a whole the object oriented model provided a better and more useful way to implement MITOMAP than the relational model. However, there were several deficiencies in the model with object oriented modeling with regards to biological variability.

C. *Flat-File*

MITOMAP was also implemented as a simple flat-file. The implementation was static and hard coded. The main benefit was that the database was tailored specifically for the mithocondrial data. This provided the required functionality. However, some of the main problems with this approach is that even small changes require the application to be programmed

again. Making the changes in the code requires a significant amount of programming and expert knowledge in the database structure. Furthermore, this approach also results in all structure of the data to be inherent in programs accessing the data. Finally, the database scalability was poor and unlike the relational and object oriented approaches, did not provide any data normalization or recovery mechanisms.

V. Conclusion

This paper showed the main current approaches for implementing biological databases. Our analysis shows that biological databases are complex and that no single approach is suitable or practical. Therefore, it is important that research efforts continue in this area in order to find different ways of storing and creating databases suitable for biological data.

References

[1] R. Alhajj, and F. Polat, "Reengineering Relational Databases to Object-Oriented: Constructing the Class Hierarchy and Migrating the Data",In Proceedings of 8th Working Conference on Reverse Engineering (WCRE'01), 2001.

[2] ASN.1, http://asn1.elibel.tm.fr/en/introduction/index.htm

[3] I. Kuan Cheang, Y. Bae Choi, and A. Tang, "Overview of the Structures of Heterogeneous Genome Databases", IEEE 27th Annual Hawaii International Conference on System Sciences, 1994.

[4] P. Chen, "The Entity Relationship Model – Toward a Unified View of Data" ACM International Conference on Very Large Databases, Framingham MA, Sept. 22-24, 1975.

[5] P. Chen, "Entity Relationship Modeling: Historical Events, Future Trends, and Lessons Learned", Software Pioneers: Contributions to Software Engineering, Springer-Verlag, Lecturing Notes in Computer Sciences, June 2002, pp. 110-114.

[6] Growth of GenBank. Available at http://www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

[7] Z. Lacroix, "Biological Data Integration: Wrapping Data and Tools", IEEE Transactions on Information Technology in Biomedicine, Vol. 6, No. 2, June 2002.

[8] L. Stein, and J. Thierry-Mieg "AceDB: A Genome Database Management System", Computing in Science and Engineering, Vol. 1, No. 3, pp.44-52, 1999.

[9] SQL Injection, http://msdn2.microsoft.com/en-us/library/ms161953.aspx

[10] S. Philippi, and J. Köhler, "Automated Structure Extraction and XML Conversion of Life Science Database Flat Files" IEEE Transactions on Information Technology in Biomedicine, Volume 10, Issue 4, Oct. 2006, pp. 714-721.

[11] X. Qian, and L. Raschid, "Query Interoperation Among Object-Oriented and Relational Databases", In proceedings of ICDE, 1995.

[12] C. Ramathan, and S. Koduri, "An Object-Oriented Prototype for a Geophysical Database", In Proceedings of 27th Southeastern Symposium on System Theory (SSST'95). 1995.

[13] R. Sargent et al., "The Design and Implementation of a Database for Human Genome Research", In Proceedings of 8th International Conference on Scientific and Statistical Database Systems, p. 220-225, Stockholm Sweden, June, 1996.

[14] N. Shamkant, and A. Kogelnik, "The Challenges of Modeling Biological Information for Genome Databases", Conceptual Modeling, LNCS 1565, pp. 168-182, Springer-Verlag 1999.

[15] R. Smith, "Brief Guide to Information Resources Supporting the Human Genome Project", IEEE Engineering in Medicine and Biology, Nov.-Dec. 1995

[16] A. Vakali, B. Catania and A. Maddalena, "XML Data Stores: Emerging Practices", IEEE Internet Computing, Volume 9 Issue 2, pp. 62-69 March-April 2005.

Hitomi Hasegawa (BS'06) attended Northeastern University in Boston MA. Currently she is pursuing a MS in Bioinformatics at the University of Helsinki. Prior to moving to Finland, she worked in research roles in biochemistry, imaging and pharmaceutical projects in both the United States and Japan. As a result, Hitomi has participated in the filing of several patents (currently under review). In the United States she did research for Rohm and Haas, and later for Northeastern University. In Japan she worked at Sunstar Inc. During her work at Rohm and Haas she won the award for Safety Procedures in the laboratory. In addition to research, Hitomi has carried out roles as an interpreter between transnational teams for Sunstar Inc. Her current research interests are in database research and development.