INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

Data Modeling for Sequence Quality Control and Assembly of a cDNA library

Yan Meng

A Thesis

in

The Department

of

Computer Science

Presented in Partial Fulfilment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

April 2003

© Yan Meng, 2003

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-77717-0

Canada

ABTRACT

Data Modeling for Sequence Quality Control and Assembly of a cDNA library

Yan Meng

Much scientific data can be characterized by properties like complexity, large volume, low update frequency, and indefinite retention, which brings up some different issues than those found in conventional business environments. There are a number of influences that should guide the development of data models in bioinformatics. These range from experience of the scientific database community across a range of disciplines, the current best practice in bioinformatics system, available data models and schemas, the impact of emerging standards, and the trend towards ontologies. Influenced by them, I have developed a conceptual object data model for the sequencing quality control and assembly pipeline for the genomics project creating a cDNA library for *Aspergillus niger* and implemented it using the relational database MySQL. This experience is summarized in a set of guidelines for data modeling in bioinformatics.

ACKNOWLEDGEMENTS

I gratefully acknowledge Dr. Gregory Butler for his advice and encouragement. I would also like to thank Yimin Liu, Jian Sun, Yueqin Chen for their precious discussion.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the development of the technology for efficient, automated DNA sequencing has led to the accumulation of large databases of DNA and protein sequences, and growth of a new field of study known as "Bioinformatics". Bioinformatics is the use of mathematical, statistical and computing methods to solve biological problems using large amounts of DNA and amino acid sequences and related information, which includes the fields of sequence analysis, comparative genomics, functional genomics, proteomics, structural genomics, medical informatics, computational biology etc.

This thesis addresses an open problem in this field, namely data modeling for bioinformatics databases. The huge amount data generated from high throughput experiments and the complexity of the data itself has made this a non-trivial job. It should support large volume data storage, efficient retrieval and exchange, ease of evolution, and other criteria. The approach taken here is to develop an object-oriented conceptual data model for the sequence quality control and assembly pipeline for the genomics project with *Aspergillus niger* and implement it in the relational database MySQL. The work

described is essentially interdisciplinary in nature in that, database techniques are used for data modeling and implementation, while the basic subject matter is biological.

1.1 Background of Data Modeling

What is Data Modeling?

The most important task in developing new or enhanced applications for institutional data and processes is the initial analysis of client requirements (Tim McLellan, 1995). Before purchasing any software and hardware, and before storing data in a database, it is vital to analyze the client's requirements to develop the appropriate solution. Time spent in analysis is proportional to the effectiveness of the resulting application. Since the early 1960s, and despite the rapid evolution since then, the initial analysis is still the principal activity that an application developer undertakes.

One technique commonly used in analyzing the client's requirements is data modeling. The purpose of data modeling is to develop a complete and accurate model, or graphical representation, of the client's information needs and work processes. The data model acts as a framework for the development of the new or enhanced application.

There are two major methodologies used to create a data model: the relational data model and the object data model. This thesis compares these two approaches, and addresses their strength and weakness in the context of bioinformatics databases.

Data modeling in the context of database design

The database design process roughly follows five steps: 1) planning and analysis; 2) conceptual design; 3) logical design; 4) physical design; and 5) implementation (Simsion, Graeme, 1994, Teory, Toby J, 1999).

In the context of database design, the conceptual data model is one part of the conceptual design process. It is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules, which govern operations on the objects. To put this in the context of the relational database, the data model is used to design the relational tables; in the context of the object-oriented database, the data model is used to design object classes.

A data model is independent of hardware or software constraints. Rather than try to represent the data, as machine would see it, the data model focuses on representing the data as the user sees it in the "real world". It uses easily understood notations and natural language, so that it can be reviewed and verified by the end-users. The data model is also detailed enough to be used by the database developers to use as a "blueprint" for building the physical database. So, it serves as a link between the concepts that make up real-world events and processes and the physical representation of those concepts in a database.

Data Modeling in Bioinformatics

A bioinformatics database is a scientific database, which shares some features with conventional business database since they are both databases, but also differs from it because of the nature of data. Much scientific data can be characterized as complex, large volume, low update frequency, and indefinite retention. This brings up some different

issues than those found in conventional business environments. There are a number of influences that should guide the development of data models for databases in bioinformatics. These range from experience of the scientific database community across a range of disciplines, the current best practice in bioinformatics systems, available data models and schemas, the impact of emerging standards, and the trend towards ontologies.

1.2 Goals

The aim is to develop a data model for the sequence quality control and assembly pipeline for the genomics project to create a cDNA library for *Aspergillus niger* and to implement it using the relational database MySQL.

The general structure of the project is to grow the fungi under a range of conditions of interest, and sample the mRNA present in the organisms under these conditions. We sample about 18,000 cDNA clones of *Aspergillus niger*, which should provide representatives of approximately 5,000 genes, of which at least 30% will be new genes. We expect to identify about 130 target genes of interest. Those targets identified by similarity will be the "easy" ones with known relatives amongst the enzyme families of interest. They should be easier to characterize, but have less novelty. They will form about 80% of all targets. Those targets identified from the gene expression studies will be more difficult to characterize, but are guaranteed to be novel in some sense. They will form about 20% of all targets. This 80--20 split is chosen because of risk management and overall budget constraints.

In many ways, this is a typical genomics project, covering sequencing, sequence analysis, gene expression analysis using cDNA microarrays, and follow-up work in the

wet lab to verify hypothesis's generated in silico. To support this project, others in bioinformatics group at Concordia have developed three pipelines for cDNA sequencing and gene expression studies using microarrays. Pipeline 1 is responsible for quality control of the raw cDNA sequences, their assembly, and their normalization to remove redundancy. Pipeline 2 is responsible for collecting information about the sequences similar to each gene, and classifies the function of the gene into the appropriate Gene Ontology category. Pipeline 3 is responsible for quality control of the data from microarrays and the determination of those genes that are highly expressed under each condition. My goal is to develop a data model for the database in support of pipeline 1.

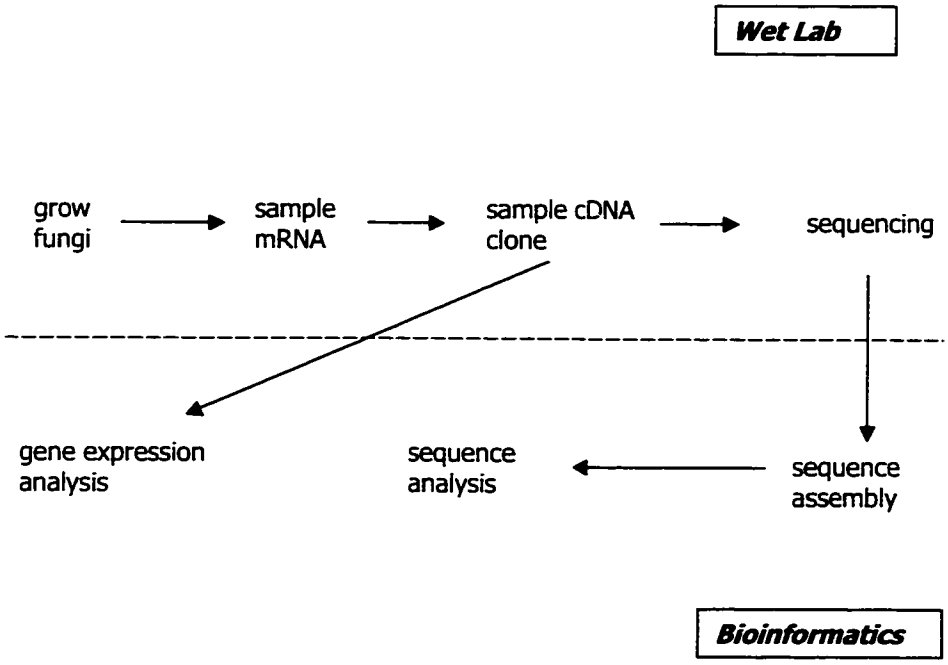
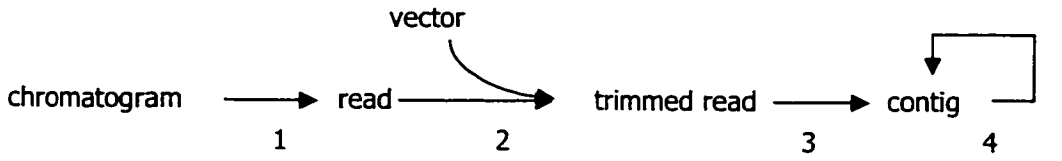


Figure 1.1 Schematic Diagram of Genomic Project

1.2.1 Introduction of Pipeline 1

The goal of pipeline 1 is to perform quality control of the raw cDNA sequences, assemble the sequences, and remove redundancy. The inputs to the pipeline are the chromatograms from the sequencers, and the final output is a list of trimmed, full-length, normalized cDNA clones representing the "genes" of the organism.

The best practice tools for basic sequence quality and assembly-- phred, lucy, phrap are adopted. For the purpose of data modeling, I need to know the input and output of each intermediate step, and analyze what is required to be kept in database. The following is a brief description of each step in the pipeline.



1. phred
2. Lucy
3. phrap and cross_match
4. consed

Figure 1.2 Schematic Diagram of Sequence Quality Control and Assembly Pipeline

The first step is base calling of raw sequence data – chromatogram data with phred. Phred reads the sequencing machine chromatograph outputs and generates sequence base calls in a sequence file together with a quality assessment file for each base call made. The input data are sequencer trace data (binary) and dye primer data. Sequencer trace data can come from SCF, ABI model 373 and 377 DNA sequencer chromatogram, and MegaBACE ESD chromatograms files (primer ID) and dye primer data. Dye primer data include Primer ID, chemistry, length, sequencing vector, priming and cloning site, dye, sequencing machine type, LI-COR gel created SCF files. Phred needs environment variable in this file for its process. For the level of interpretation perspective, the input data is raw/sensor data. The output data are base calls (validated data); quality values of sequence (interpreted data) in FASTA format.

In the second step, the sequence fragments (reads) are processed to generate trimmed sequences for assembly using lucy.

Lucy is a utility that prepares raw DNA sequence fragments for sequence assembly. It reads the quality information for sequences, computes good quality regions, chops off splice sites from the sequences, and then remove vector insert sequences. Lucy accepts output files from phred as input data files: the *sequence file*, the *quality file*, and (optionally) a *second sequence file* for comparison purposes. All three files should be in the plain FASTA format. The first sequence file and its accompanying quality file are obtained from phred, the optional second sequence file usually comes directly from the sequencing machine itself and is used to reassure/enhance the quality of the sequences. The output data of Lucy are the cleaned sequence file (validated data) with markers for

good quality regions, and a companion quality file (interpreted) data. They are ready for third step--fragment assembly.

The third step is assembly of trimmed sequence fragments using phrap. phrap is a program for assembling DNA sequence data. It uses cross_match for sequence alignment. cross_match is a general-purpose utility (based on a "banded" version of SWAT, an efficient implementation of the Smith-Waterman algorithm) for comparing any two sets of (long or short) DNA sequences. It can be used to compare a set of reads to a set of vector sequences and produce vector-masked versions of the reads; a set of cDNA sequences to a set of cosmids; contig sequences found by two alternative assembly procedures (e.g. phrap and xbap) to each other; or phrap contigs to the final edited cosmid sequence. Here it is used to compare a set of reads for assembling. phrap takes output from Lucy as input, allows use of entire read (not just trimmed high quality part); uses a combination of user-supplied and internally computed data quality information to improve accuracy of assembly in the presence of repeats; constructs contig sequence as a mosaic of the highest quality parts of reads (rather than a consensus); provides extensive information about assembly (including quality values for contig sequence) to assist trouble-shooting. These output data are derived data from validated data in the second step (Lucy). phrap is usually used in conjunction with phred; and with the sequence editor/assembly viewer, consed.

The last step is to edit the assembly using consed. consed is a program for viewing and editing assemblies assembled with the phrap assembly program. Output data can come from several assemblies from phrap, so the relationship between input data

(assemblies from phrap) and output data can be considered as a "many to many" relationship.

1.2.2 Requirements of Data to be Modeled

The ultimate goal of a data model is to completely and accurately represent the data requirements of the end users. Since the end users are scientists, I communicated with them and summarized their requirements below:

Requirement 1: The database should store chromatogram file —the raw data.

This is the original data generated from web lab, in a binary format. It is not used often in later analysis, but sometimes, scientists need to trace back to the original data when new analysis tool emerges. For example, new software tools for sequence quality control, vector screening etc.

Requirement 2: The database should store sequence data —the validated data.

In this pipeline, validated data refers to the output from phred. All data derived from raw data are referring to this validated data, including the trimmed sequence data from lucy, the contig sequence data from phrap, the final edited assembled sequence from consed.

Requirement 3: The database should store trimmed sequence data—the derived data from validated sequence data.

The trimmed sequence data are used for sequence assembly.

Requirement 4: The database should store data related to automatic contig assembly—the derived data from trimmed sequence data.

Trimmed sequences are assembled to contig, which is also derived data. Which trimmed sequences are used in assembly and how they are assembled should be recorded, so that later on, when the sequence assemblies are required to be modified by humans, the relevant information can be retrieved.

Requirement 5: The database should store final edited contig sequence data—the derived data from trimmed sequence data.

The final edited contig sequence data are the final output from pipeline 1, which is then used for sequence analysis. It should also include assembly information as is in automatic assembly step; moreover, it should include the consensus sequences of contig for future use of sequence analysis.

Requirement 6: The database should store quality data for all sequence data—the metadata.

Quality data is essential for scientist to see the quality of the sequence data they are using for sequence analysis, and especially when it is necessary to edit the assembly.

Requirement 7: The database should store information about software used in each step of the pipeline—the metadata.

It is important to know how the data are processed in each step. There are many software tools available, and each has its special feature, the software will evolve too. This information is useful because, sometimes, scientists need to know how the raw data is processed for diagnostic purpose, or when they want to update the tool, it is good to know what they are using right now.

Requirement 8: The database should store features about sequence—metadata.

Information related to sequence, such as the creator of the sequence, the date it was created, and the wet lab protocol for generating the raw data are metadata of the sequence data, which should be stored as well in order to re-generate the whole process used for producing the data.

1.3 Organization of the Thesis

This thesis develops a conceptual data model of sequence quality control, and assembly pipeline for the genomics project to create a cDNA library for *Aspergillus niger*: The chapters are organized so as to describe what guides the development of this data model, and eventually how the model is generated. Thus, Chapter 2 covers the status of data modeling in bioinformatics, comparing the scientific database versus conventional database; relational data model versus object data model; summarizing the current standards, and best practice, trends in bioinformatics. Chapter 3 discusses how the conceptual model is developed, and the implementation of the conceptual model in MySQL schema. In the end, Chapter 4 briefly reviews the goals of the project, the work done to reach the goals, summarizes a set of guidelines of data modeling in bioinformatics from this project, and outlines the potential future work extending from this thesis.

In my thesis, I use some notational conventions to distinguish table name, table column and text body. Table names appear in arial 12 for lower case, ARIAL 10 for upper case, table columns appear in *arial narrow italic 12* for lower case, *ARIAL NARROW ITALIC 10* for upper case.

Chapter 2

The Status of Data Modeling in Bioinformatics

There is a wealth of experience and material to draw upon to guide the development of data models for databases in genomics and bioinformatics. (Frishman *et al.* 1998,Letovsky 1999). These range from surveys of bioinformatics databases (Frishman *et al.* 1998, Letovsky 1999), the experience of the scientific database community across a range of disciplines (French *et al.* 1990), basic data modeling principles (Simsion, Graeme 1994, Teory, Toby J, 1999), the current best practice (Ensembl/Apollo/DAS, phred/Lucy/phrap, MIAME etc), available data models and schemas (Ensembl, *Saccharomyces* genome database etc), the impact of emerging standards, and the trend towards XML (Achard *et al.* 2001), ontologies (The Gene Ontology Consortium}, Distributed Annotation System (DAS) which is being used for the annotation of the human genome.

2.1 Scientific Database versus Conventional Database

Scientific databases are often very large. But size alone does not necessarily distinguish between scientific and commercial databases. Here I compare the similarities and differences between these two, and emphasize on characteristics of the scientific data.

2.1.1 Similarities

Many issues regarding databases in a variety of scientific disciplines and those in conventional business environments are similar. For example, in both data modeling, it is important to know the applications and analysis in which the data will be used; to communicate with database users about their needs, be clear on the project goals, to try to plan for future trends in analysis and application of the data. This is a non-trivial task, especially for scientists who, unlike business people with experience in assisting in data modeling, have little or no experience. In addition, flexible, efficient query processing is vital in each environment.

2.1.2 Differences

What make the scientific database different from business database are the three fundamental characteristics of the data and the anticipated operational environment. First, the scientific entities are usually more complex with significant interrelationships. Consequently, object-oriented technology often seems more appropriate. Second, scientific databases are rarely transaction oriented. Recorded data is to be preserved for posterity; therefore often data has low update frequency. Retrieval of relevant data is far more important. Third, retrieval of data is often "volumetric, " that is, based on two or

more conjunctive range searches. Genome sequencing projects are producing complete genome sequences of organisms. In the post-genomic era, high-throughput technology is producing a huge amount of data on functional genomics, proteomics, transcriptomics and metabolomics. Very often, scientists need to analyze data collected from multi-sources. These characteristics of scientific data bring up some different issues than those found in conventional business databases.

A major report (French *et al.* 1990) on scientific databases discusses both the user perspective and the technology perspective. Much of what it said from the user perspective is still valid today. A later report (Williams *et al.* 1999) confirms the needs of the user, and despite the availability of the web and XML still notes major technological hurdles for scalability, data integration, presentation of information, and interoperability.

These issues are also relevant to bioinformatics (Frishman *et al.* 1998). From a point of view of the user, the following basic questions need to be answered in order to enhance the scientific research environment:

What data is available to me?

Where is the data located?

How can I get it?

What can I do with it?

2.1.3 Scientific Data in Three Dimensions

Since scientific data are so complex and large, it was suggested that the scientific databases could be characterized in three dimensions: level of interpretation, intended

scientific analysis, and data source. These dimensions may exist in commercial databases environment, but the data are relatively simple in each dimension.

2.1.3.1 Level of Interpretation

Data undergoes many transformations from raw sensor data to fully interpreted results that appear in the scientific literature. Some classifications of data include:

Raw/sensor data – raw values obtained directly from the measurement device, (rarely saved);

Calibrated data -- raw physical values corrected with calibration operators (usually saved);

Validated data -- calibrated data that has been filtered through quality assurance procedures, (commonly used data for scientific purposes);

Derived data -- any data that is the result of transformation on other data;

Aggregated data – derived data that is the result of an aggregation transformation, such as averages, standard deviations etc.

Interpreted data -- derived data that is related to other data sets, or to the literature of the field.

This sequence of successively increasing levels of interpretation and information about the processing is crucial to fully understanding and using a data set.

2.1.3.2 Intended Scientific Analysis

It is assumed that all scientific data are subject to further analysis, such as computations, queries, and other applications made of the data. Since the nature of such

subsequent analysis frequently determines the desirable representational format, it is critical to answer some key questions:

- What operators will be applied to the data?
- Does the data representation support these operators efficiently?
- Are the operators intrinsic to the DBMS or not?

Earth science data is often viewed as time-sequences and multidimensional tables because it is analyzed statistically..Space science data analysis may consist of multi-spectral Fourier transformations of very large 2D and 3D arrays of numbers. Genome data analysis may mean pattern matching over linear character sequence data, for instance, BLAST (Basic Local Alignment Search Tool) is a set of similarity search programs using heuristic algorithm to detect relationships among sequences of either protein or DNA, which are presented as linear character sequence data, such as a, t, c, g for nucleotide of DNA, K, S, V, E etc for amino acid of protein.

2.1.3.3 Data Source

This dimension is generally not mentioned in the database literature, particularly commercial databases which are usually single-source database environments. For scientific databases, it is common that all necessary data can either be obtained from a single data repository, or may need to be collected and integrated from many data sources. Genomics tends to have multiple data sources, with users sometimes referencing single repositories for detailed work, but often requiring accessing multiple sources for their work. In this they may use Entrez or SRS for *ad hoc* queries, or create an internal data warehouse in the case that they plan extensive data analysis or data mining.

Key to answering most of the above questions for a user is metadata, which consists of information that characterizes data. Metadata are used to provide documentation for data products. In essence, metadata answer **who**, **what**, **when**, **where**, **why**, and **how** about every facet of the data that are being documented. Since the metadata is very complex, standards of metadata become necessary to describe all possible data. For example, The Federal Geographic Data Committee approved the Content Standard for Digital Geospatial Metadata (FGDC-STD-001-1998) in June 1998. The Microarray Gene Expression Data (MGED) society established MIAME - standard of the minimum information about a microarray experiment required to interpret and verify the results.

Data, together with the metadata associated with it, provides self-descriptive data. It should be sufficient information for a scientist, either now or in the future, to decide whether the data is relevant to their investigations, whether an analysis technique may be applied to the data or not, and how much confidence the scientist can have in the result of the analysis.

2.2 Relational Data Model versus Object Data Model

Since the 1980s, the relational data model has been the dominant in DBMS. In the beginning, relational database systems supported a small, fixed collection of data types (e.g., integers, dates, strings), which has proven adequate for conventional domains, including most commercial areas. Later on, market of microsoft SQL server and MySQL started to grow, contributing to advanced developments in relational database management system. In the late 1980s and the 1990s, advances have been made to store

new data types such as image and text (IBM's DB2, Oracle 8, Informix UDS). In many application domains, however, much more complex kinds of data must be handled. Typically this complex data has been stored in OS file systems or specialized data structures, rather than in a DBMS (Raghu Ramakrishnan *et al*, 2000). Examples of domains with complex data include the Computer Aided Design (CAD), Computer Aided Manufacturing (CAM) and Computer Aided Software Engineering (CASE) applications.

As the data volume increases, the advantages of the many features offered by a DBMS, such as reduced application development time, concurrency control and recovery, indexing support, and query capabilities, become more obvious. In order to support such applications, a DBMS must support complex data types. That is how the object-database system began to emerge.

Object-database systems have developed in two distinct paths: Object-oriented database systems and Object-relational database systems, with the latter considered an extension of relational database system. In this section, I focus on the object-oriented database system, and compare its data models with relational data model.

There are concepts in the relational data model that are similar to those in the object data model. A table in a relational data model can be considered to be analogous to a class in an object data model. A tuple is similar to an instance of a class in that they are both instance of a table or a class. A column in a tuple is similar to a class attribute in that they both describe the attributes of the table or the class. However, these concepts have other distinct features, which distinguish them despite their similarities. A tuple has no behaviors, but an instance of class has. A column in a tuple can hold only primitive data types while a class attribute can hold data of any type. Finally classes have methods that

can be applied to objects of the class, but there is no analog to methods in relational models (Raghu Ramakrishnan *et al*, 2000).

Below is a list of advantages and disadvantages of using an OODBMS over an RDBMS with an object oriented programming language.

2.2.1 Advantages of Object-Oriented Database:

(1) More data types: Different from relational DBMS, Object-Oriented DBMS stores objects rather than simple data types. When handling image, graphics, and audio/video data, relational DBMS has to use BLOB (Binary large object), however, object-oriented DBMS can handle it more elegantly by defining the specific data types (Raghu Ramakrishnan *et al*, 2000).

(2) Composite Objects and Relationships: An object is a better model of the real world entity than the relational tuples with regards to complex objects, and it usually has hierarchical characteristics. Objects in an OODBMS can store an arbitrary number of atomic types as well as other objects. It is thus possible to have a large class, which holds many medium sized classes which themselves hold many smaller classes, ad infinitum. In a relational database this has to be done either by having one huge table with lots of null fields or via a number of smaller, normalized tables, which are linked via foreign keys. A join has to be performed when one wants to query data based on the "object". The analogy (attributed to Ester Dyson) would be the requirement of disassembling your car before storing it in your garage each night and then reassembling it before using it each morning. This has both development productivity and runtime performance issues. Programmers spend more coding time mapping the program object to the database. At runtime, similar performance penalties are paid moving data between the database and

the running application. Depending on the complexity of the data being handled, an OODBMS can outperform an RDBMS by ten to a thousand times.

(3) No querying language: A relational data model models the static parts of the system, and uses query language to perform dynamic operations or rules that change the state of the data in the system. An object data model models both static and dynamic parts of the system. ODBMS provides transparent bindings to object-oriented languages such as C++, Java and Smalltalk, rather than force a single query language (i.e. SQL) as its interface. An entire application can thus be comprehensively modeled in one UML diagram (McFarland, Gregory, 1999).

(4) No Impedance Mismatch: In a typical application that uses an object oriented programming language and an RDBMS, a significant amount of time is usually spent mapping tables to objects and back. When the models are changed, this mapping layer gets more complex, more error prone, and very hard to maintain. There are also various problems that can occur when the atomic types in the database do not map cleanly to the atomic types in the programming language and vice versa. This "impedance mismatch" is completely avoided when using an OODBMS, since whatever your application objects are, they are taken as-is by the OODBMS.

2.2.2 Disadvantages of Object Database:

(1) In OODBMS, the objects may be extremely large, and may contain redundant data, so they require more space than the tuples in RDBMS, in which the relations are normalized to reduce redundancy.

(2) The field of Object-oriented database is relatively new compared to that of relational database, and the ODBMS is still under improvement, so it does not function as well as most RDBMS (Oracle, MySQL etc.) although in many aspects they may outperform RDBMS in certain circumstance. For example, OODBMSs excel at managing objects, especially in environments where the operations to be performed on those objects are reasonably well known when the database is designed. They rarely perform well when called upon to deal with ad hoc query environments or applications requiring significant use of traditional data such as numbers and character strings. SQL is well designed to deal with that traditional data, and virtually all SQL products today are quite efficient when dealing with unpredictable -- and unpredicted -- queries and combinations of data.
(3) Lack of Ad-Hoc Queries: In an RDBMS, the relational nature of the data allows one to construct ad-hoc queries where new tables are created from joining existing tables then querying them. Since it is currently not possible to duplicate the semantics of joining two tables by "joining" two classes then there is a loss of flexibility with an OODBMS. Thus the queries that can be performed on the data in an OODBMS are highly dependent on the design of the system.

Taking both advantages and disadvantages into consideration and together with some practical reasons, many available ODBMSs are hybrid systems, or object-relational DBMSs. However, these systems are able to handle complex data types equally well.

For bioinformatics databases, since the data is complex, the object model is more appropriate than the relational model, but by experience, object database technology is not yet ready to support bioinformatics applications. Hence, most bioinformatics databases use RDBMS, such as Ensembl, SGD etc. Nevertheless, ODBMS is also used in

bioinformatics considering its many advantages; the most successful OODBMS is AceDB. AceDB was originally developed for the *C. elegans* genome project, from which its name was derived (**A** *C. elegans* **D**ata**B**ase). However, the tools in it have been generalized to be much more flexible and the same software is now used for many different genomic databases from bacteria to fungi to plants to man. It is also increasingly used for databases with non-biological content.

Having an object model prepares for future trends where object database technology may become the technology of choice for implementing genomics databases. It also recognizes the advantages of object modeling at the conceptual level. Of course, the mapping between an object-oriented conceptual model and the relational logical model must be documented. I document conceptual models using the Unified Modeling Language (UML), the standard notation for object-oriented modeling.

2.3 Standards, Best Practice and Trends.

2.3.1 Standards

Some basic standards for databases are adopted in bioinformatics. Many of other standards that arise in bioinformatics are based on best practice, which describes the community consensus on how best to store, exchange, and analyze genomics data. In the course of time, various international bodies have created official standards closely based on these best practices in use.

Standards for databases

SQL is the standard for relational databases, for example, SQL-92 was developed by the INCITS Technical Committee H2 on Database. SQLJ was developed by The SQLJ Group, a consortium comprised of database vendors and Sun Microsystems. ODMG is the standard for object-oriented database, for example, ODMG 3.0 was developed by the Object Data Management Group (ODMG) (Web service).

Standards for data interchange

As a result of high-throughput methodologies, huge amount of data have been produced everyday, scientists need to work with large volumes of data in different formats and coming from a variety of sources, so data exchange becomes key to genomics. Standards for data interchange are critical because without which coordination and viewing of the data has been difficult. At the BIO (Biotechnology Industry Organization) 2001 Conference, the Interoperable Informatics Infrastructure Consortium (I3C)'s model of a common open platform uses XML and Java™ technology as the basis for exchanging and sharing life science data in an open consistent manner.

Based on the general standards for data exchange, there are various standards developed for specific data.

Standard for functional annotation of genomes

The Gene Ontology has become a standard for functional annotation of genomes. It can be applied to all organisms even as knowledge of gene and protein roles in cells is accumulating and changing.

Standards for microarray data

The Microarray Gene Expression Data (MGED) society, an international organization for facilitating the sharing of microarray data from functional genomics and proteomics experiments, is establishing standards for microarray data annotation and representation, facilitating the creation of microarray databases and providing infrastructure for dissemination of experimental and data transformation protocols. There are four major standardization projects being established.

- MIAME - standard of the minimum information about a microarray experiment required to interpret and verify the results.
- MAGE - standard of a data exchange format (MAGE-ML) and conceptual object model (MAGE-OM) for microarray experiments.
- Ontologies - standard for microarray experiment description and biological material (biomaterial) annotation in particular.
- Normalization - standard for experimental controls and data normalization methods.

2.3.2 Best Practice

Best practice describes the community consensus on how best to store, exchange, and analyze genomics data. Many of the standards arise in bioinformatics from best practice because the community has found them to be workable, available, and useful.

Available schemas can be found in the literature for conceptual models, and at web sites for XML schemas for data exchange, and for relational database schemas (usually for MySQL, but also Oracle). A number of models, schemas, and ontologies have been published recently including a broad conceptual model. For those like us who are

studying fungi, the Saccharomyces Genome Database (SGD), with a well-documented Oracle relational schema, is a valuable resource. Ensembl uses MySQL and documents its schemas, and MAGE provides models and schemas for microarray data.

To date, the best practice of the large national and international sites, such as Sanger, EBI, TIGR, NCBI, and DDBJ, generally lead the way in disseminating high quality software and systems. One widely used platform for annotation and sequence analysis is Ensembl/Apollo/DAS being developed at EBI, Sanger, Cold Springs Harbor, and elsewhere. For basic sequence quality and assembly, the best practice tools (phred, lucy, phrap) come from TIGR and University of Washington. There are competitors in the MIAME-compatible database arena, including ArrayExpress from EBI, M-CHIPS from DFKZ, and BASE from Lund, while the Institute for Systems Biology has a basic microarray data analysis pipeline.

Controlled vocabularies aid communication with scientists, add precision to terminology amongst modelers, provide precise scalar values for scalar data types, and may be used for hierarchical classification schemes.

With these standards, and best practice available, we should avoid reinventing the wheel, follow the standards and best practices. For example, use phred, lucy, phrap for sequence assembly, use distributed annotation system (DAS) to facilitate annotation, use gene ontology for function annotations and use MIAME guideline of MGED to model microarray data.

2.3.3 Trends

The trends in bioinformatics projects are to provide XML schemas for data exchange (Achard *et al.* 2001), use DAS for data integration, and use ontologies for annotation.

XML stands for eXtensible Markup Language. It provides a framework for defining markup languages, where each XML language is targeted at its own application domain that defines its own tags, tailored to the kind of information used in that domain.

XML separates syntax from semantics and from presentation in that browser rendering semantics is completely defined by stylesheets. XML is platform independent, and self-descriptive. It has proven itself invaluable for data exchange and interoperability. Based on the XML data standard Bioinformatics Sequence Markup Language (BSML) – XML Data Standard for Genomics, many XML definitions for molecular biology have been developed, such as, MSAML - an XML for Multiple Sequence Alignments; MAGE-ML – an XML for microarray gene expression data; PROXIML - an extensible XML Schema definition for automated exchange of protein data etc.

The Distributed Annotation System is being developed by Lincoln D. Stein, Sean Eddy and Robin Dowell. The distributed annotation system (DAS) is a client-server system in which a single client integrates information from multiple servers. It allows a client to collect genome annotation information from multiple web sites, collate the information, and display it to the user. DAS consists of a reference sequence server, and one or more annotation servers. It relies on there being a common "reference sequence" on which to base annotations. Each annotation is anchored to the genome map by way of a start and stop position relative to one of the reference subsequences. In this way, reference data and annotation data are decoupled, and it is easy to re-compute features, or extend the types of features included in the annotation without impacting the underlying reference data. Currently, DAS servers are running at WormBase, Flybase, Ensembl, TIGR and UCSC.

An Ontology is the working model of entities and interactions in some particular domain of knowledge or practice, such as molecular biology or bioinformatics. One definition (GO 2002) is "An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms." An ontology should be reusable, and it is this characteristic that distinguishes an ontology from a database schema. A database schema is intended to satisfy only one application, but ontology could be reused in many applications. Ontologies vary in their coverage and level of detail. They range from controlled vocabularies such as glossaries and data dictionaries, to terms organized into (generalization) hierarchies such as the Gene Ontology, all the way to formal mathematical specifications using description logics such as OWL (DAML+OIL).

Ontologies have a variety of roles in knowledge engineering (GO 1999):

1) A reference shared by a community. The knowledge is authored in a single language, and converted into a different form for use in multiple target systems. Benefits include knowledge re-use, improved maintainability and long term knowledge retention.

2) A conceptual database schema, or a definition of a common vocabulary for database annotation. The ontology is a specification, ensuring that a common vocabulary is available for description, sharing and posing questions. Benefits include documentation, maintenance, reliability, sharing and knowledge re-use.

3) A common access point to information. The information to be shared may be expressed using unfamiliar vocabulary. The ontology helps to render the information

intelligible by providing a shared understanding of the terms or mapping between the terms. Benefits include interoperability, and more effective use and re-use of knowledge resources.

4) A query mechanism over databases. Ontology is used for searching an information repository. Queries can be refined by following relationships within the ontology, by moving up and down the generalization hierarchy within the ontology, or by narrowing the value of an attribute. Benefits include more effective access and hence more effective use and re-use of knowledge resources.

5) A "world of discourse" model for database annotation and technical literature, that allows sharing of information and supports natural language processing (NLP). These ontologies are designed to link domain knowledge to linguistic structures such as grammar and lexicons.

Ontologies clarify important abstractions and relationships, covering a range of precision; we should choose the most appropriate level of precision and mathematical sophistication for individual project. When formal ontologies are not used, controlled vocabulary is necessary; it should cover concepts, entities, relationships, and classification schemes.

Chapter 3

Data Modeling of Sequence Quality Control and Assembly Pipeline

Based on the requirement analysis in Chapter 1, I summarize the data and metadata that need to be modeled for our project. The data include Chromatogram data, read, trimmed read, contig, and consensus data. The metadata are the source of data, transformation of data, lab equipment, experiment, protocol (reagents, dyes, vectors, primers...), and personnel. Having these initial goals, I examined the current best practice of sequence database, and analyzed the design principle used to see how they model data and metadata, and adapted it into context of my project.

3.1 References of the Project to Best Practice

For the purpose of data modeling, it helps to refer to the current best practice of bioinformatics databases and systems, such as Sanger, EBI (European Bioinformatics Institute), TIGR (The Institute for Genomic Research), NCBI (The National Center for

Biotechnology Information, and DDBJ (DNA database of Japan), which are the large national and international sites. One widely used platform for annotation and sequence analysis is Ensembl/Apollo/DAS being developed at EBI, Sanger, Cold Springs Harbor, and elsewhere. Ensembl uses MySQL and documents its schemas. For those like us who are studying fungi, the Saccharomyces Genome Database (SGD), with a well-documented Oracle relational schema, is a valuable resource. A great advantage of these systems is that access to all the data produced by the project, and to the software used to analyze and present it, is provided free and without constraints.

The DAS (distributed annotation system) protocol, as mentioned before, is providing distributed genome annotation in many databases, such as Ensembl, WormBase, FlyBase, TIGR, and UCSC (University of California at Santa Cruz). The basic concept of "reference server" can be used in modeling the sequence assembly.

Other than the databases and systems, a number of conceptual models have been published recently. A conceptual modeling of genomic data by Paton *et al.* provides conceptual models that describe eukaryotic genome sequence data and genome organization, plus a number of important functional data sets, namely protein interaction data, transcription data, and results from gene deletions. The information models presented were developed in the context of a project that is focusing initially on the management of *Saccharomyces cerevisiae* data. Since our project is also about Fungi, so it can be easily adapted into our project. Note, however, that the emphasis in that paper is on fully sequenced genomes, no models are provided for mapping data, and nevertheless, the broad concept covered in this paper can serve as an important reference for sequence analysis. There is a conceptual model for mapping data provided by Barrilot *et al.* (1999).

It addresses the problem that there are many databases that store genome maps using different schemas to describe their content, often, it is necessary to compare and integrate the data from different databases, but transforming one representation into another is difficult and requires detailed knowledge of every database. They proposed a standard representation written in IDL (Interface Definition Language). The goal is that if each database provides an interface based on this IDL, clients only need to know this interface and can retrieve data from heterogeneous databases in a homogeneous fashion. This interface has been defined as a consensus between laboratories involved in maintaining databases for genome mapping (Infobiogen and EBI). Ensembl is one of the EBI databases, so on one hand, this conceptual model can helps us to understand the Ensembl schema, on the other hand, the Ensembl schema gives an example of the implementation of the conceptual model, it will certainly help me both in data modeling and implementation in MySQL.

In summary, considering the degree of relevance to our project, I decide to use Ensembl, SGD, DAS and conceptual model of Barrilot *et al.* as main references for data modeling and implementation of pipeline 1. There are four advantages of using Ensembl over SGD as a good reference and starting point of my project. First, Ensembl project uses MySQL for database implementation, which I will use in our project too. Then, I can be freed from worrying too much about the implementation detail, such as database constraints, SQL syntax etc, and focus on the content of the schema. Second, the DAS protocol is used for data distribution; it provides a good example of the protocol. Third, a conceptual model about mapping (Barrilot *et al.* 1999) is partly (totally Infobiogen and EBI) based on information of genome mapping at EBI, which is the home of Ensembl. It

describes the genome maps in the conceptual level, so it provides the most direct reference for data modeling of pipeline 1 of our project. Fourth, it adopts CVS to keep track of version changes; it is useful for understanding the principle for those changes. Therefore, it is worth the effort to deeply understand the design principle and schema of Ensembl. The advantage of using Saccharomyces Genome Database (SGD) as a reference is that the subject of the genome project is also fungus, therefore the context is similar. Although it is implemented in Oracle, with a well-documented Oracle relational schema, it still provides valuable information about fungi sequence assembly and analysis. The comparison of Ensembl and SGD is shown in Table 3.3.1.

In this section, I briefly describe each of the references, in the perspectives of how they model the data, and what part of the model is relevant to our project, and develop a conceptual data model for our project.

Table 3.1.1 Comparison of Ensembl Database and SGD Database.

	Ensembl	SGD
Implementation	MySQL	Oracle
DAS	Yes	No
CORBA interface	Yes	No
Organism	Human, mouse (non-fungus)	Yeast (fungus)
cDNA	No, genomic DNA	No, genomic DNA
Updates	Use CVS	No.
Conceptual model	No.	No.
Schema	Yes	Yes

3.1.1 Conceptual Model about Mapping

The conceptual model about mapping developed by Barrilot *et al.* concentrates on the mapping aspect of genomic data (see Figure 3.1). They use separation of location and objects. The types of the mappable objects are for markers: STS (sequence and tagged site, see glossary), EST (expressed sequence tag), etc.; for clones:YAC(Yeast Artificial Chromosome, BAC (Bacteria Artificial Chromosome), P1 (P1 phage), etc.; for maps: genetic, physical, radiation hybrid, sequence, etc. A Mappable object can be a point or a segment. Points have a point location, for example marker. Segments have some extent, defined for instance by flanking markers or left and right positions. Examples are clones or chromosome bands. A Segment can be Map and Clone. Genome maps are simply segments; other segments and points are objects on this map. All these objects are Mappable objects. The location of all Mappable objects is defined in a separate data structure—MapElement. Maps can span entire chromosome or only certain regions. Maps can contain other maps. This is useful if one wants to dynamically assemble a chromosome-wide map from a number of regional maps. A MapElement defines location information of mappable objects; they are classified in five categories. (1) point position, which is a single coordinate (for instance for exactly positioned markers); (2) ordered position, which is an order information (a rank, for maps where objects have only an order, but no absolute position), for instance, in Ensembl there is sticky_rank for exon when one exon expands two contigs; (3) range position, which is a left and a right flanking Mappable object. (4) vague position, which has no location information. (5) interval position, which is a left and a right position (for instance for clones, which have a left and a right end coordinates).

Using this model, a physical map is represented as a linear map. Clones (Mappable objects) are connected to it through an interval position (MapElement), while markers (Mappable objects) are connected through a point position (MapElement). A genetic map can be treated like a physical map of markers, but with a different unit (centiMorgan instead of base pair).

The separation of object and location can be seen in Ensembl schema, which is

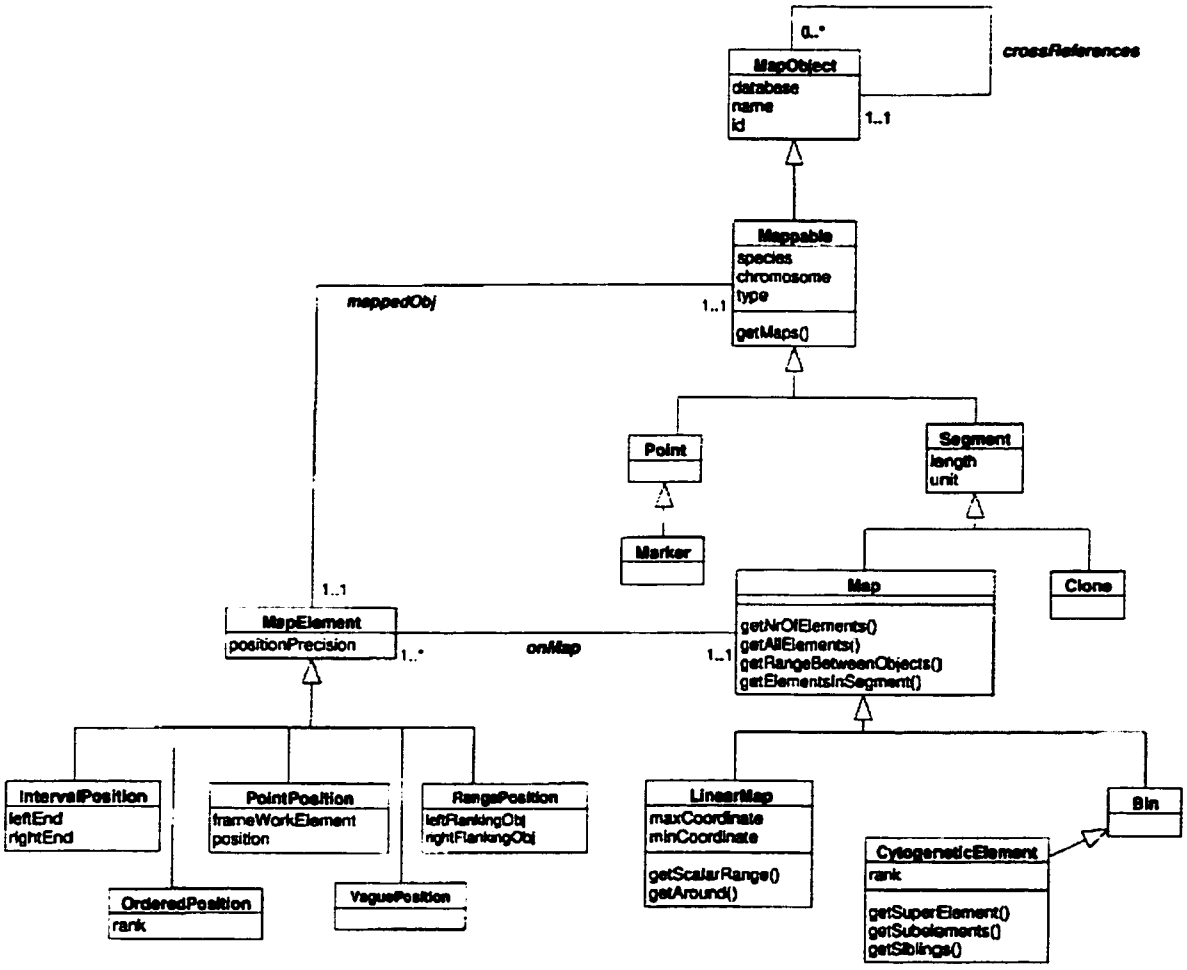


Figure 3.1 Graphical representation of the genome map in UML notation

3.1.2 Ensembl

Ensembl is a joint project between EMBL - EBI (European Molecular Biology Laboratory) and the Sanger Institute to develop a software system which automatically tracks all the sequenced pieces of the human genome, attempts to assemble them into large single stretches and then analyze the assembled DNA to find genes and other features of interest to biologists and medical researchers. Ensembl is primarily funded by the Wellcome Trust.

The Ensembl project aims to provide a curated, distributed, non redundant view of the genomes of model organisms. The first release of Ensembl concentrated on human genomic data, both finished and unfinished. Subsequent releases have presented mouse, Fugu, zebrafish and mosquito data. Ensembl will concentrate on metazoan genomes, but the system is being adapted by other groups for use with plant and fungal genomes.

As a software/database system Ensembl can be best described as a hybrid of a scripting programming language (Perl) and a relational database (MySQL). Ensembl Perl software inherits from a tradition of biological object-design developed through BioPerl (www.bioperl.org), so that developers can create reusable pieces of software that would faithfully describe biological entities such as gene, transcript, protein, genomic clone, or chromosome. The Ensembl database is based on a relational database called MySQL, which is available free of charge for noncommercial developers. Simple queries of the database can be handled using the SQL language, but for complex queries demanded by most biological analyses the Ensembl MySQL server is best accessed using Ensembl Perl objects.

It would be ideal if Ensembl as well as other databases had published both their conceptual model, and schema, then we would have adapted the conceptual model to our project with less effort, and get implementation example from the schema. For some reasons, none of them has documented it, or don't want to make them accessible to public. But fortunately, they do provide MySQL schema, and a graphic schema diagram for the latest release (version 9.30.1), which does help understanding the design concepts. So what we need to do in the design phase is, in a sense, to reverse the design concepts from the implementation (schema), and generate our object conceptual model. The implementation in MySQL is straightforward following the schema in Ensembl.

The Ensembl schema in MySQL is made publicly available on the internet. While developers gained more experience, they frequently made improvements to the schema. The schema has changed a lot since the version 1, it was version 8 when I started my project in May 2002, schema was updated on September. 3rd, 2002, and updated again in November, generating the current version 9.30.1, however, the basic concepts are not changed much.

In the core database, the tables can be classified in four categories based on their contents: assembly, gene, feature, map and reference objects. Raw data in Ensembl is contig represented by contig table. Ensembl stores all coordinates relative to contigs. For assembly coordinates you always need to join to the assembly table. All the positions with *xxx_start* and *xxx_end* correspond to interval position of MapElement defined in Barillot's mapping model. (For schema diagram, see the appendix.)

3.1.2.1 Tables in Ensembl Core Database

Assembly Tables

Clone table represents one accession number in EMBL/GenBank (accession number is the embl_acc). The *clone_id* will link out to other tables.

contig table represents each continuous piece of sequence inside an EMBL/GenBank accession. For finished clones there is just one contig per clone. For unfinished (draft) there is more than one. The offset of the contig in EMBL/GenBank coordinates is in the offset column.

The DNA table holds the sequence of the contigs from the contig table, and has a one to one mapping with it.

chromosome table represents individual chromosome in the human genome. *known_genes*, *unknown_genes* and *snps* represent the number of each category on this chromosome.

The assembly table represents the assembly. For each chromosome there is a list of contigs. The *chr_start* and *chr_end* columns indicate where on the chromosome this raw contig is placed. The *contig_start* and *contig_end* columns indicate which part of the contig is used. The *contig_ori* indicates the orientation of the raw contigs. Each contig is in some "fingerprint clone contig" and the fingerprint clone contig is identified by field *superctg_name* and the position of the specified bit of the contig within its FPC contig is given by fields *superctg_start* and *superctg_end*.

Nearly everything that does something with an assembly (e.g., number of genes on a chromosome) has to join to the assembly table via the *contig_id* column from a contig column.

Feature tables

Feature tables have undergone major changes between version 8 and 9. There were four feature tables in version 8, feature, repeat_feature, protein_feature, and supporting_feature. In the course of time, the feature table was becoming large and unwieldy. Some columns in the feature table were becoming redundant for some analyses. Therefore, in the latest version (version 9), the feature table has been distributed into the following tables:

Table 3.1.2 Feature Tables of Ensembl Database.

Feature Table Name	Content (Stored Analyses)
dna_align_feature	Blastn & BlastGenscanDNA
prediction_transcript	Genscan, Genefinder & FGenesh
protein_align_feature	Blastx & BlastGenscanPep
repeat_consensus	TRF & RepeatMasker (via a standalone script)
simple_feature	markers (EPCR), Eponine, tRNAs, CPG islands & GC content
repeat_feature	RepeatMasker & TRF (Tandem Repeat Finder)
protein_feature	relevant for BLAST results alignment feature of translation.
supporting_feature	a linkage table which links a record to its alignment information stored in one of the *_align_feature tables.

In the new feature tables, *contig_id* is the contig id this feature is on. *contig_start*, *contig_end*, *contig_strand* are self evident. In dna_align_feature and protein_align_feature tables, *hit_start*, *hit_end* are the "hit" coordinates (relevant for BLAST results). In repeat_feature table, *repeat_start* and *repeat_end* are for repeats for the repeat consensus positions. The *analysis_id* column links to the analysis table that tells you what program and database wrote in these features. Some tables have unique

column for derived data calculated from analysis, such as *score*, *p_value* (in *prediction_transcript* table), *perc_ident* (percent of sequence that is identical), *evalue* (in dna_align_feature and protein_align_feature tables) etc.

The storing of supporting features has been modified to complement the new feature tables in version 9. In previous releases complete alignment information was stored in the supporting_feature table. This has been refactored such that alignment information is now stored in the relevant dna_align_feature and protein_align_feature tables. The supporting feature has now simply become a linkage table which links a record to its alignment information stored in one of the xxx_align_feature tables.

gene_description stores feature of gene, mapdensity and karyotype table record feature of chromosome.

Gene tables

Genes are a set of transcripts, with one-to-many relationships between gene and transcript. Transcripts have an ordered set of exons. One exon can be in more than one transcript. Only the exons have positions on the DNA. So exons and transcripts have a many-to-many relationship, which are mapped by exon_transcript table.

Some exons cross two contigs, (everything is stored relative to contig coordinates); these exons are called "sticky" exons and then are built up of a series of components (contig) with the same id but different *sticky_rank*s. The exon is assembled by concatenating the exon components in the *sticky_rank* order. The position here corresponds to *ordered position* of MapElement defined in Barillot's mapping model.

Each transcript has one translation. In theory two transcripts could have the same translation, but they do not currently have that. A translation indicates the start *(exon_start_id)* and end *(exon_end_id)* exon of the coding sequence and the relative offset to the exon of the start codon (*seq_start*) and stop codon (*seq_end*). (The relative offset is necessary to handle start/stops on sticky exons). genes, transcripts, translations and exons all have a *stable_id* which Ensembl attempts to keep stable over assemblies as separate tables.

Map tables

The FPC Map delivers some information about the current existing clones from human BAC libraries.

mapfrag table represents a fragment of the regional or genome map. It can be clone, supercontig, assembly, band and chromosome. One mapfrag can have many DNA fragments (dnafrag table), for example, assembly contains many raw contigs.

Map sets are sets of related maps. A mapset have many map fragments, with one-to-many relationship.

mapannotation table and mapannotationtype table store features of map fragment.

External Link

The xref set of tables provides external links for genes (e.g., HUGO names, Swissprot ids). There is a many-to-many relationship between external items and Ensembl items and in each link optionally we have the "strength" of the link. Knowing this the tables fall out quite normally – object_xref links an Ensembl object to an xref

object. An xref object has a name and description and comes from an external_db. There is a synonym table for xrefs. The strength of the link is given by the identity_xref table which links to the auto-generated id in object_xref, which represents each unique link between an Ensembl object and an xref object.

3.1.2.2 Data modeled in Ensembl

Having understood the schema of Ensembl core database, it is obvious to see that the most relevant tables to our project are assembly tables. However, understanding the whole structure is important to grasp the key of the design principle, and to prepare for further development of data models. Therefore, I summarize, based on core database, what data are modeled, and what metadata are recorded in the core database on the conceptual level.

Here, I describe the data from the perspective of different interpretation levels.

The data modeled in Ensembl are:

Contig — It is validated data from DNA sequencing process. It can be considered as the "raw data" of Ensembl.

Chromosome — It is also derived data from contig.

Exon — It is derived data from contig.

Transcript — It is derived data from exon.

Gene — It is derived data from transcript.

3.1.2.3 Metadata modeled in Ensembl

Feature — Ensembl records information about the sequence (contig) as features, which are stored in a set of feature tables.

Data transformation — Information about transformations of data are stored in analysis table. Derived data such as *evalue*, *score*, *p_value* are generated from analysis.

Interpretated data — Derived data such as *evalue*, *score*, *p_value* are generated from analysis.

Data change — Ensembl keeps track of the data changes, without actually changing the data by version in a set of tables of stable_id (gene_stable_id, transcript_stable_id, translation_stable_id, and exon_stable_id).

Data exchange — Information of external links for genes is stored in a set of xref tables.

3.1.3 SGD:

The SGD is funded by the National Human Genome Research Institute at the US National Institutes of Health. The SGD is in the Department of Genetics at the School of Medicine, Stanford University. The *Saccharomyces* Genome Database (SGD) project collects information and maintains a database of the molecular biology of the yeast *Saccharomyces cerevisiae*. This database includes a variety of genomic and biological information and is maintained and updated by SGD curators. The software package used to store the data is Oracle. The table specifications, schema diagrams, and business rules for the SGD database are provided.

SGD website maintains complete latest schema specification, including the table specifications, schema diagrams, and business rules for the SGD database. It provides

more information than Ensembl does, and the well-documented implementation makes it easier for developer to understand the design concept behind the implementation. What is even better is that there is a specific set of schema used for storing other fungal sequences. Since our project is a fungal project, it serves as a closest example.

In the core database, in order to compare SGD and Ensembl, I classify the tables in the same five categories as in Ensembl, based on their contents: assembly, gene, feature, map and reference objects.

3.1.3.1 Tables in SGD Fungi Specification

SGD schema contains similar information to those of Ensembl, but they model the information in different ways, and classify the tables into different categories than that of Ensembl. To compare these two sets of schema, I classify them into the same categories as in core database of Ensembl, i.e. the assembly tables and feature tables, gene tables, map tables and external link tables. (See table 3.1.3), and there are two relevant schema diagrams in the appendix (B, C), appendix B is the fungi table specification. Schema in appendix C is obtained from extracting relevant tables from other diagrams, without indicating the all relationships among the tables, which is however described in the text. For complete schema, see SGD schema specification at SGD website: http://genome-www4.stanford.edu/Saccharomyces/SGD/doc/db_specifications.html.

The mostly relevant category is assemble tables, of which I will demonstrate the design principle by comparing the schema of Ensembl and SGD (see Figure 3.2).

Assembly Tables

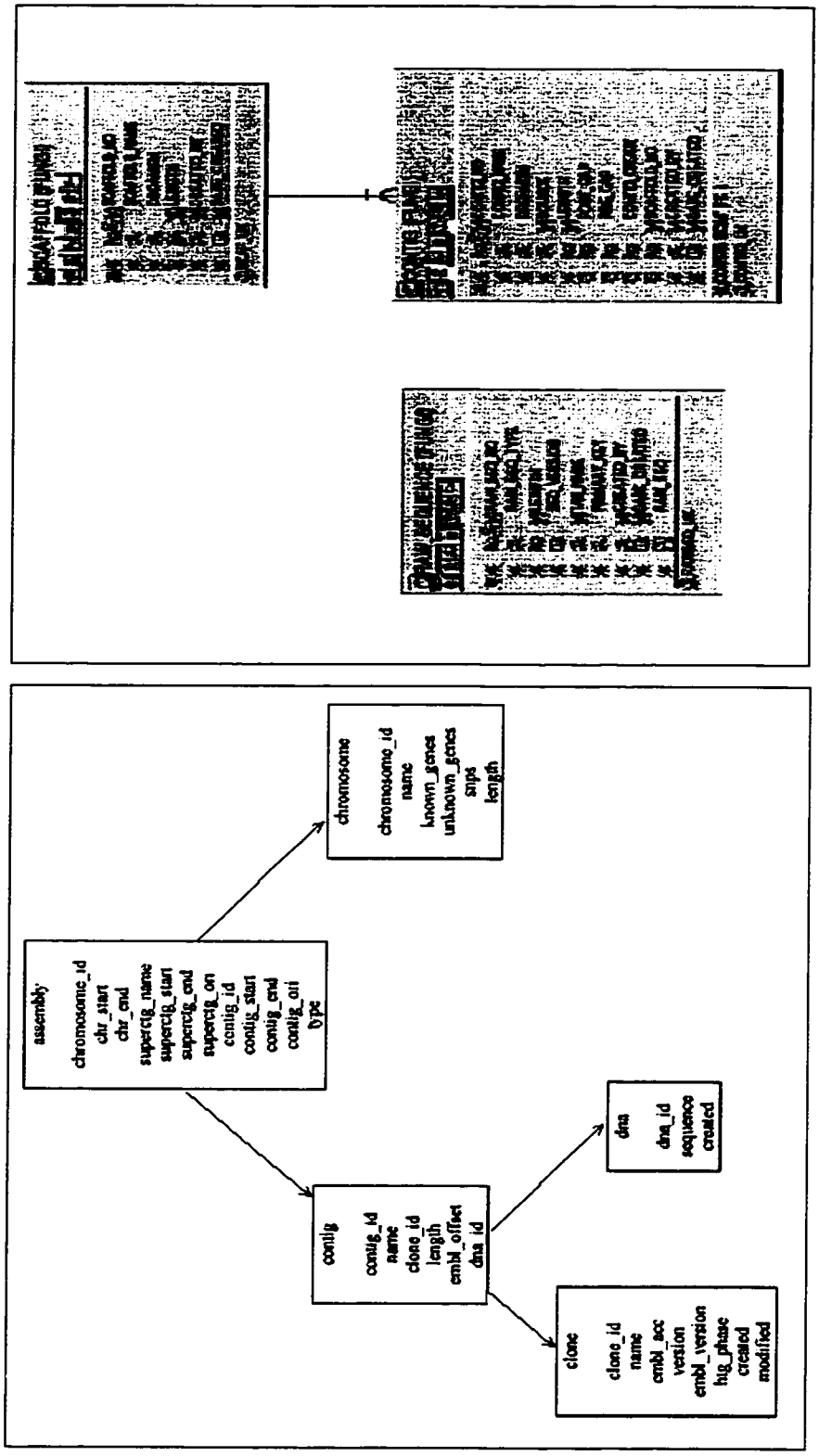


Figure 3.2 Comparison of Assembly in Ensembl and SGD

CONTIG table represents contigs used for sequence assembly. It includes assembly information represented by *PRE_GAP*, *POST_GAP*, *CONTIG_ORDER*, and *SCAFFOLD_NO*; it corresponds to contig table and assembly table in Ensembl. It uses contig order instead of coordinates (as in Ensembl) to indicate the position of contig in the assembly (scaffold). It is similar to the way Ensembl handle sticky exons.

The RAW_SEQUENCE table holds the real sequence data of a sequence (e.g. contig, scaffold), identified by sequence name (*TAB_NAME*) and id (*PRIMARY_KEY*).

The SCAFFOLD (or super-contig) table assembles the contigs in order.

Note that, the contig orientation is not recorded explicitly. As explained by Dr. Kara Dolinski (personal communication), who is Database Curator at SGD, this is due to the first set of data that they loaded into the schema. The way that they received the data and the subsequent display of it required them to do an on-the-fly determination of contig orientation in their display. In the future, they will likely add a column in the contig table that indicates its orientation explicitly.

It can be seen from Figure 3.2 that the separation of object and position is observed in Ensembl schema, representing object with tables, contig, chromosome, and position with table assembly. However, in SGD, object and position are not always separated, object and position is represented with one schema CONTIG (both object and position), which corresponds to contig (object) and assembly (position) in Ensembl.

Feature tables

ALIGNMENT table stores information about an alignment between two features, more precisely two subsequences of chromosome where the feature belongs. This table

includes a great amount of information about alignment, e.g. the coordinates of features (sequences) being aligned, the method that was used to align the sequences (e.g. *WU-BLAST*), the matrix used in the analysis. It also stores derived data from the alignment, e.g. the percent of the sequence that is similar (*PCT_SIMILAR*). The percent of the sequence aligned (*PCT_ALIGNED*) and the percent of the sequence that is identical (*PCT_IDENTICAL*). This table together with FEATURE_TYPE corresponds to three tables in Ensembl, i.e. dna_alignment_feature, protein_align_feature and analysis.

Gene tables

CHROMOSOME stores genetic and physical map information. It is used as coordinates of features.

FEATURE stores genomic features (e.g. ORF, tRNA etc) in the coordinates relative to chromosome. It corresponds to the gene table and transcript table in Ensembl.

FEATURE_TYPE stores the type of the feature (e.g. ORF, tRNA, DNA, protein).

SUBFEATURE stores subfeatures associated with features (e.g. introns, exons, etc) in the coordinates relative to feature. It corresponds to the exon table in Ensembl.

FEATURE_CONTIG links features to contigs in the coordinates of contigs.

Gene is also indicated by genetic locus represented by LOCUS table, which is a genetic location on chromosome.

GENE_PRODUCT contains gene product information for a locus. It is analogous to translation table in a sense that translation is a kind of gene product.

Note that, although the names of the tables are all features, they represent the information of genes, transcripts, exons in Ensembl, and more than that, they also store

information of intron, protein, tRNA that are modeled in different tables in Ensembl (not specified here).

I would like to draw attention on that chromosome is not directly involved in assembly process, but the feature tables. To get any chromosome information related to scaffold or contig, the FEATURE table, FEATURE_CONTIG table have to be joined.

Map tables

SGD now offers a Combined Physical and Genetic Map for each chromosome that is generated from the most current genetic and physical map data in SGD.

Genetic map of genome is built up from genetic two-point mapping. CHROMOSOME serves as a coordinate system for locus represented by LOCUS table. There are many loci on one chromosome.

TWO_POINT table contains two point mapping data between the locus and a marker on the same chromosome. A mapset may have many map fragments, with one-to-many relationship.

SGD provides an on-the-fly generated physical map that includes the locations of ATCC clones, ORFs and other sequence features. Again, CHROMOSOME serves as a coordinate system for clones represented by the CLONE table.

The reason both genetic map and physical map can be built for *Saccharomyces cerevisiae*, and only physical map (FPC Map) is built for human in Ensembl is that cross between different strains can be easily done in yeast, as well as other low level organisms. However, it is impractical (impossible) in human, so different methodologies are adopted to obtain the map.

External link

Two tables provide external links in other databases.

EXTERNAL_ID contains all external IDs (e.g., PIR, Swiss-Prot) for various components in the database.

Table 3.1.3 Comparison of Ensembl schema and SGD schema.

For schema detail, see the schema diagram in appendix.

Table Category	Ensembl	SGD
Assembly	clone, contig, dna, chromosome, assembly	CONTIG, SCAFFOLD, RAW_SEQUENCE
Feature	dna_align_feature, protein_align_feature, prediction_transcript, repeat_consensus, simple_feature, repeat_feature, protein_feature, supporting_feature, analysis mapdensity, karyotype.	ALIGNMENT, FEATURE_TYPE
Gene	gene, transcript, exon, translations, gene_descrition, gene_stable_id, transcript_stable_id, exon_stable_id, translation_stable_id,	CHROMOSOME, FEATURE, FEATURE_TYPE, SUBFEATURE, FEATURE_CONTIG, GENE_PRODUCT
Map	mapfrag, dnafrag, mapfrag_mapset, mapset, mapannotation, mapannotationtype	CHROMOSOME, TWO_POINT, CLONE
External Link	xref, identity_xref, object_xref, external_db, external_synonym	EXTERNAL_ID, TEMPLATE_URL, EI_TU, EXPRESSION_CONNECTION

TEMPLATE_URL contains template URLs used to link to specific entries in other databases.

EI_TU table links table between EXTERNAL_ID and TEMPLATE_URL tables

There is no counterpart of object_xref which links Ensembl items to external items, and each link is used directly by the SGD component, for example, EXPRESSION_CONNECTION table has a column *WEBSITE_URL* linking to external URL for the expression dataset through tables TEMPLATE_URL, EI_TU, and EXTERNAL_ID.

3.1.3.2 Data modeled in SGD

Once again, the most relevant tables to our project are assembly tables. From the comparison between schema of Ensembl and SGD, I have shown that SGD modeled almost the same data (in conceptual level) as that in Ensembl.

The data modeled in Ensembl are:

Contig — It is validated data from DNA sequencing process. It can be considered as the "raw data" of SGD.

Scaffold — It is also derived data from contig.

Chromosome — It stores current information about chromosome.

Feature — It is mixture of derived data (subregion on chromosome) and metadata of the data (features).

Subfeature — It is derived data from feature, and also mixed with metadata of the data (features of the subsequence).

3.1.3.3 Metadata Modeled in SGD

Feature — SGD records information about the sequence (contig, chromosome) as features, which are stored in a set of feature tables.

Data transformation — Some information about transformations of data such as data related to alignment are stored in the ALIGNMENT table. However, they are not as complete as what is modeled in the analysis table in Ensembl.

Interpretated data — Derived data such as evalue, score, pvalue are generated from analysis, such as score in SCORE table, some data (e.g. *PCT_SIMILAR*) in table ALIGNMENT.

Data change — SGD keeps track of the data changes, without actually changing the data by version in FEATURE table, CHROMOSOME table, but no tables are dedicated to this purpose like in Ensembl (stable_id tables)

Data exchange — Information of external links for genes is modeled in EXTERNAL_ID and TEMPLATE_URL tables, and referred by tables in SGD.

3.1.4 DAS

DAS (Distributed Annotation System) decentralizes sequence annotation among multiple third-party annotators and allows client-side software to integrate annotation on an as-needed basis. A single server is designated the "reference server." It serves essential structural information about the genome, for example, the physical map, which relates one entry to another (where an "entry" is an arbitrary segment of the sequence, such as a sequenced BAC or a contig), the DNA sequence for each entry, and the standard authorship information. Multiple sites then act as third-party "annotation servers".

The distinction should be emphasized between annotation and curation. Annotation is extraction, definition, and interpretation of features on the genome sequence derived by integrating computational tools and biological knowledge. Curation is exploration of genomic annotations at many levels of detail, validation of annotations; expert curation is usually performed by researchers. DAS is making an internet wide distributed annotation framework to facilitate comparison among different centers' annotations, no attempt is made to automatically resolve contradictions between different third-party annotations which is the task of curation. At Ensembl, DAS servers are currently running for annotation. The on-going Apollo Project is aiming to solve the "genomic curation editor" problem. This is not the same in any way as the DAS project.

Inspired by the design of the reference server, I adopted the DAS style for data modeling of our project, where the assembly sequence can be designated as a "reference", corresponding to "reference server", and multiple sequences (reads) are stored in coordinates of the reference, which are analogy to third-party "annotation servers."

3.2 Conceptual Models

The conceptual model is modeled in UML (Figure. 3.3). In UML class diagrams, classes are drawn in rectangles, with the name of the class at the top, and optionally with the attributes and operations of the class shown below. Relationships between objects are represented by lines connecting the objects.

chromatogram: It is an object category, which satisfies Requirement 1. The attributes are described below:

id -- identity of chromatogram data.

name -- the name of the chromatogram (sometimes it is given nick name).

organism -- the organism it belongs, it is kept for the purpose of future sequence analysis, such as comparison of sequences among different species.

chromatogram_seq – the binary chromatogram data of the sequence, this is the raw data that was often thrown away before, it was realized vital later on when raw data are required to trace the transformation of data.

primer_id -- the primer used for sequencing experiment, it, as well as dye, chemistry and machine are necessary for subsequent steps of sequence quality control and assembly.

dye -- the dye used dye -- the dye used for sequencing experiment, see *primer_id*.

chemistry -- the chemistry used for sequencing experiment, see *primer_id*.

machine -- the sequencing machine used for sequencing experiment, see *primer_id*.

read: It is an object category, which satisfies Requirement 2, 6. The attributes are described below:

id -- identity of read sequence.

name -- the name of the read (sometimes it is given nick name).

organism -- the organism it belongs, it is kept for the purpose of future sequence analysis, such as comparison of sequences among different species.

length -- the length of the sequence.

analysis_id – analysis performed to get this read, for example, in our pipeline, base calling is performed by a software program phred.

experiment_id – the wet lab experiment performed to obtain the data.

chromatogram_id – the chromatogram data corresponding to the read.

dna_seq – the ASCII sequence data generated by base calling. The possible values are A, T, C, G, X, and N. This is the most original human readable sequence data, which is used most often by scientists.

dna_quality_seq – the quantitative quality of each base call. It is important for the following steps of sequence assembly.

analysis: It is an object category, which satisfies Requirement 7. The attributes are described below:

id -- identity of the analysis.

name -- the name of the analysis (sometimes it is given nick name).

program -- the software program which performs the analysis.

program_version – the version of the software program. The software program usually evolves with time, the functionalities and feature usually are not the same between different version, so keep this information will assure the proper use of the program.

program_file – the program as well as the user manual of the program.

parameters – usually program allows user to set up parameter values according to special situation. The values of those parameters are important for explaining the result.

description – general information of the analysis, which are not included in other attributes.

experiment: It is an object category, which is used to fulfill Requirement 8. The attributes are described below:

id -- identity of the experiment.

date_created – the date the experiment is performed, web lab scientists usually keep lab notebooks daily. Therefore it is important information when it is necessary to trace back to wet lab experiment performed to generate the sequence data.

place – the place where the experiment is performed.

creator – the person who performed the experiment.

trimmed_read: It is an object category, which satisfies Requirement 3, 6. The attributes are described below:

id -- identity of trimmed_read sequence.

name -- the name of the trimmed_read (sometimes it is given nick name).

organism -- the organism it belongs, it is kept for the purpose of future sequence analysis, such as comparison of sequences among different species.

length -- the length of the sequence.

seq_vector_id – the vector that should be trimmed from read to get trimmed_read.

analysis_id – analysis performed to get this trimmed_read, for example, in our pipeline, trimming is performed by a software program Lucy.

read_id – the read data corresponding to the trimmed_read.

dna_seq – the ASCII sequence data generated by base calling. The possible values are A, T, C, G, X, and N. This is the most original human readable sequence data, which is used most often by scientists.

dna_quality_seq – the quantitative quality of each base after trimming.

precontig: This object category describes the contig sequence assembled from trimmed sequence data, which satisfies Requirement 4, 6. The attributes are described below:

id -- identity of precontig sequence.

name -- the name of the precontig (sometimes it is given nick name).

organism -- the organism it belongs.

length -- the length of the sequence.

version -- the version of the precontig. Assembly sometimes are performed many times, it is important to keep all the data.

analysis_id – analysis performed to get this precontig, for example, in our pipeline, assembly is performed by a software program phrap and crossmatch.

read_id – the read data corresponding to the trimmed_read.

list_of_trimmed_reads – the list of trimmed_reads assembled in the precontig, and information about how they are assembled.

contig: This object category describes the final edited contig sequence, which satisfies Requirement 5, 6. The attributes are described below:

id -- identity of contig sequence.

name -- the name of the contig (sometimes it is given nick name).

organism -- the organism it belongs.

length -- the length of the sequence.

version -- the version of the contig. Assembly sometimes are performed many times, it is important to keep all the data.

analysis_id – analysis performed to get this contig, for example, in our pipeline, assembly is performed by a software program consed.

editor -- the person who edited the assembly manually.

concensus_seq -- the assembled sequence data in ASCII text. It is the sequence for further analysis.

concensus__quality_seq -- the quality of assembled sequence data. It is necessary when scientists interpret data analysis results.

list_of_trimmed_reads – the list of trimmed_reads assembled in the contig, and information about how they are assembled.

seq_vector: This object category describes sequence vector used for removing vector in raw sequence to get trimmed sequence, which satisfies Requirement 8. Vectors are used during sequencing, and the vector may contaminate the reads, and it should be trimmed out of the read. Since as a sequence itself, it has its own properties, and it is not the sequence data that eventually we need, it is better to define an object category separately from trimmed sequence data as defined next. The attributes are described below:

id -- identity of sequence vector.

name -- the name of the contig (sometimes it is given nick name).

version -- the version of the vector. Sometimes, scientists use the same kind of vectors, but may not be exactly the same vector, so information about producing the vector should be recorded.

vector_seq -- the sequence of the vector in ASCII text.

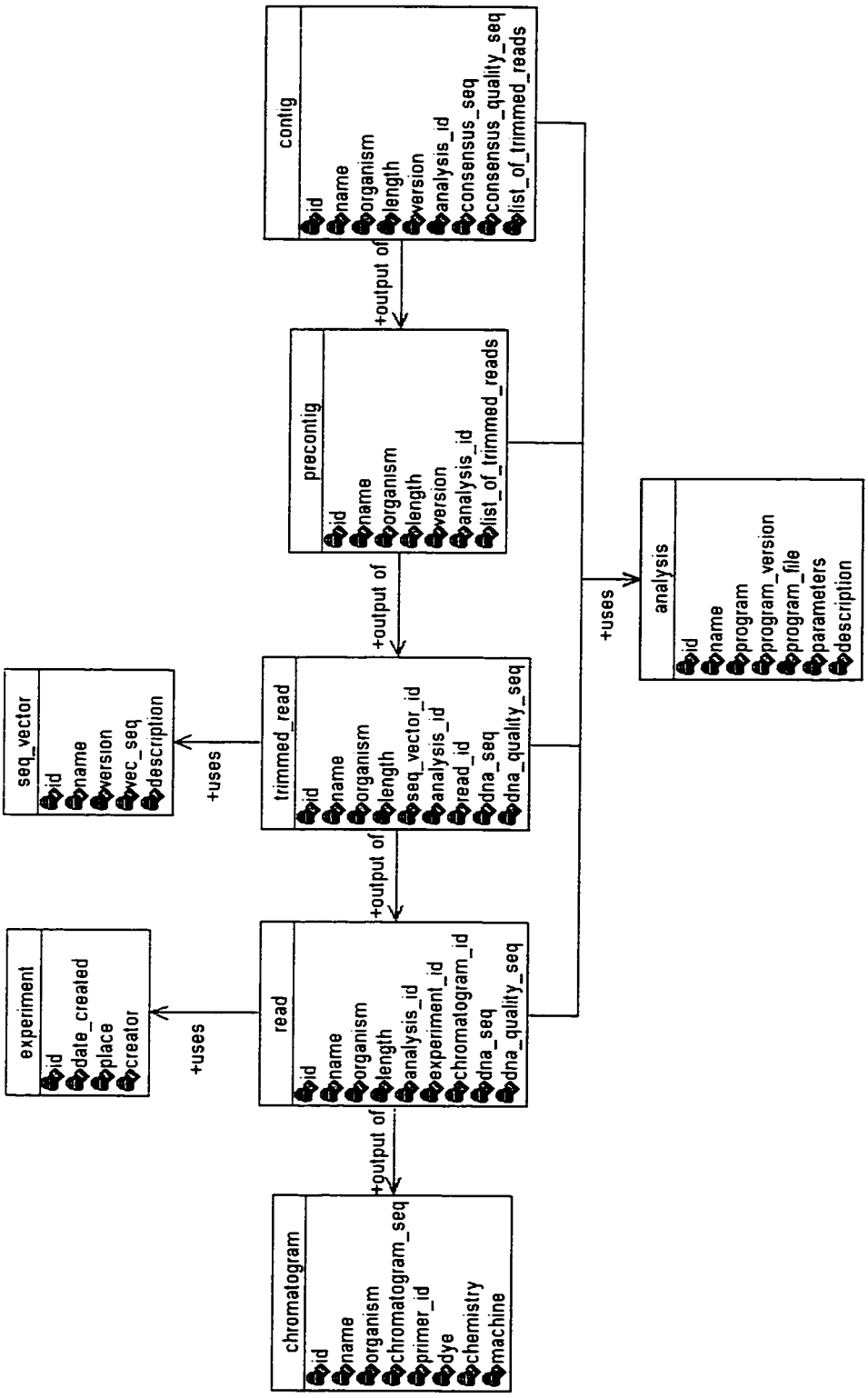


Figure 3.3 Conccptual model of Pipeline 1.

3.3 MySQL Schemas

I implemented the conceptual model using the relational database MySQL. Each conceptual object in conceptual model is represented by one or more entities in the relational schema (see table 3.3.1). The arrow in the diagram connecting entities describes the foreign key, with the head of the arrow pointing to the entity being referenced. I adopted the naming convention similar to that of Ensembl, which nearly always has an integer primary key (often auto-generated) named as *table_name_id*. Wherever there is a column in another table referring to this table it gets the same column name (e.g., *read_id* in *trimmed_read* points to the read table). The schematic overview of the schema is shown in Figure. 3.4. The Schema Diagram for relational data model is modeled using Rational Rose. Only one-to-many relationships are represented, others are all one-to-one relationships. Each schema is explained in detail, and the tables are in Appendix C.

Table 3.3.1: object of conceptual model versus relational schema

Object of Conceptual model	Relational schema
chromatogram	chromatogram
read	read, dna, dna_quality
analysis	analysis
experiment	experiment
trimmed_read	trimmed_read, dna, dna_quality
seq_vector	seq_vector
precontig	precontig, autoassembly
contig	contig, assembly, dna, dna_quality,

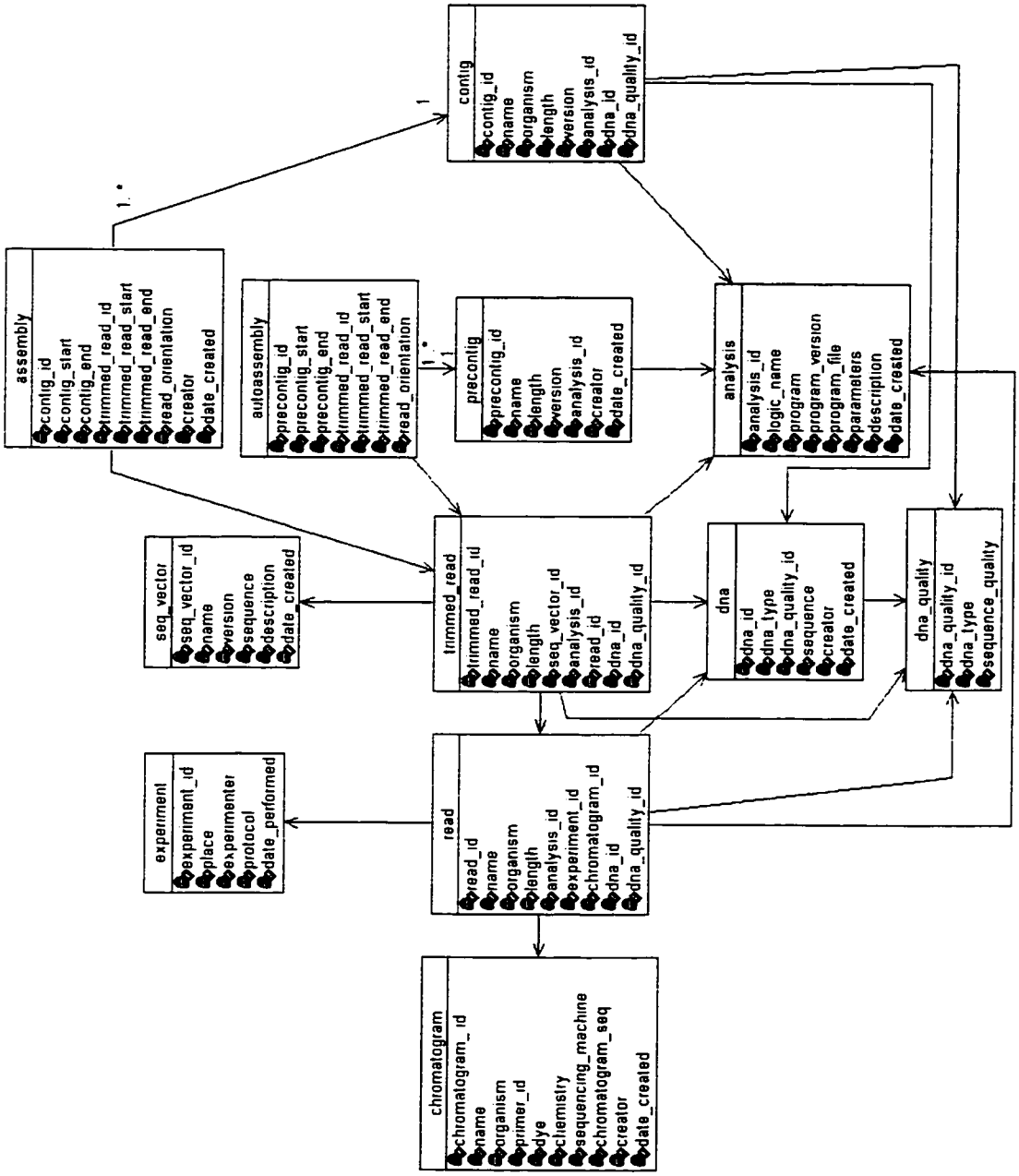


Figure 3.4 Schema Diagram of Pipeline 1.

The object category chromatogram in conceptual model maps to the data in chromatogram table. The type used is mediumblob, one of the blob types. A BLOB is a binary large object that can hold a variable amount of data. The only difference between BLOB and TEXT types is that sorting and comparison is performed in case-sensitive fashion for BLOB values and case-insensitive fashion for TEXT values. In other words, a TEXT is a case-insensitive BLOB. In Ensembl, the type of sequence data is mediumtext, so here I assume mediumblob is big enough to accommodate the binary data of chromatogram.

The object category read in conceptual model maps to tables read, dna, dna_quality in relational schema. DNA data is usually stored separately in database, so should DNA quality data. The *dna_quality_id* in table read can also be obtained by joining tables dna and dna_quality. However, assuming that dna_quality will be selected frequently, it is put here to reduce cost of joining.

The object category trimmed_read in conceptual model maps to tables trimmed_read, dna, dna_quality in relational schema. The *dna_quality_id* in table trimmed_read is put here for the same reason as in table read.

The object category precontig in conceptual model maps to tables precontig, autoassembly in relational schema. The data in precontig table defines first attempt of assembling trimmed reads to contigs. Table autoassembly stored information of assembly, corresponding to the attribute *list_of_trimmed_reads* of object precontig in conceptual model. Each row represents a trimmed read (*trimmed_read_id*, foreign key from trimmed_read table) at least part of which is present in the preassembly. The part of the trimmed_read that is in the path is delimited by columns *trimmed_start* and

trimmed_read_end (start < end), and the absolute position within the autoassembly (*precontig_id*) is given by *precontig_start* and *precontig_end*. The purpose of it is to store intermediate assembly information generated by automated software, since the DNA sequence of the final contig will be stored in the database explicitly, storing one-to-one corresponding DNA sequence for precontig will be redundant, the DNA sequence of precontig can be obtained by joining tables precontig, autoassembly, trimmed_read and dna.

The object category contig in conceptual model maps to tables contig, assembly, dna and dna_quality in relational schema. The data in contig table defines assembly edited by people. It is ready for sequence analysis. Consensus sequence is the sequence of the final assembled contig. The data in assembly table defines the final assembly sequence either automatically assembled by software or manually edited by human. Similarly, each row represents a trimmed read (*trimmed_read_id*, foreign key from trimmed_read table) at least part of which is present in assembly. The part of the trimmed_read that is in the path is delimited by columns *trimmed_start* and *trimmed_read_end* (start < end), and the absolute position within the assembly (*contig_id*) is given by *contig_start* and *contig_end*.

No matter whether the contig was edited by people, it is convenient to store the sequence instead of getting the sequence data on-the-fly by joining tables contig, autoassembly, trimmed_read, and dna, since it is the starting point for sequence analysis. The *dna_quality_id* is put here for the same reason as in Table read.

The dna table holds the sequence of the reads, trimmed reads, contig, super_contig from corresponding table. The type used for sequence is mediumtext, one of the TEXT

types. In Ensembl, the type of sequence data is mediumtext, so I adopt the same type here. *dna_quality_id* is put here for easy tracking of dna_quality. The *dna_quality_id* can also be obtained by joining table dna, and corresponding tables (e.g. trimmed_read, contig).

The dna_quality table holds the sequence quality of the reads, trimmed reads, contig, super contig from corresponding tables. The type used for sequence is mediumtext, same as the type for sequence, and the length of the data should be the same as the length of corresponding dna data.

The object category experiment in conceptual model maps to table experiment in relational schema.

The object category analysis in conceptual model maps to table analysis in relational schema.

The object category seq_vector in conceptual model maps to table seq_vector in relational schema. It holds the vector sequence that is used in the sequencing process, which is used in the trimming procedure.

For all tables, attributes *date_created* and *creator* are either added (such as chromatogram) or can be obtained by joining tables (such as joining read and dna), which records the date the data is entered into database.

Chapter 4

Conclusion

I reviewed the standards, best practices and trends in Bioinformatics. Considering the degree of relevance to our project, I decided to use the conceptual model of mapping (Barrilot *et al.*), Ensembl, SGD, DAS and as main references for data modeling and implementation for the sequencing quality control and assembly pipeline for the genomics project with *Aspergillus niger* at Concordia University. From the conceptual model of mapping, I adopted the idea of separation of object and location in sequence assembly processes. Ensembl uses MySQL for implementation, hence I implemented the data model in MySQL, I adopted its data type, naming convention. Ensembl also has advantages of using DAS for data distribution, complying with conceptual modeling of mapping, and using CVS to keep track of schema update. SGD is database of yeast, i.e. fungus, therefore, provides an example for data modeling in the context of fungus. DAS decentralizes sequence annotation among multiple third-party annotators and allows client-side software to integrate annotation on an as-needed basis. I adopted the idea for implementation of data modeling; designate assembly sequence as reference sequence, and reads as "annotation sequence".

CHAPTER 4 CONCLUSION

Data modeling is the design process, which needs testing and refining in later stage. The future work generated from this work is to test the databases built using this data model and refine the model. At the implementation level, Ensembl is again a good reference to look at. Ensembl effectively passes all the low-level bioinformatics stuff, essentially sequences, translations and flat file writing, to Bioperl. One of the most useful things which the Bioperl layer provides is the ability to write sequence objects to flat files, such as fasta format flat file, EMBL and GenBank flat files.

Our future work should include writing perl script using bioperl to automatically exchange data between flat file and database, develop XML schema for future data exchange, develop an object data model in ODL and a complete glossary, implement DAS system for future data integration.

Based on the influence of others and my own experience, I have summarized a set of guidelines of data modeling in Bioinformatics.

1. Set up clearly short-term and long-term goals and analyze current scientific requirements, and potential requirements in the future.
2. Develop object-oriented conceptual models, but implement using relational database technology.
3. Use versioning of data, do not modify data.
4. Record metadata, including quality data, confidence of the data, metadata on transformations used to manipulate, analyze and interpret data, all the information necessary so that the process of interpretation of data can be re-generated.

5. Follow the available standards, and refer to best practices.
6. Plan for data integration and interchange; adopt distribution annotation system (DAS), XML and Ontology.
7. Use controlled vocabulary when no ontology is available, and prepare for the semantic web.

The guidelines serves as a review for the current status of data modeling in bioinformatics, it should evolve as new technology emerges and new standards form. I hope these guidelines will help others grasp what is going on in this field, and be able to focus on the design of data model for bioinformatics project. It certainly can be applied in data modeling of pipeline 2 and 3 or the genomic project at Concordia University.

References

Frédéric Achard. Guy Vaysseix. and Emmanuel Barillot XML, bioinformatics and data integration. Bioinformatics 2001 17: 115-125.

MA Andrade. NP Brown. C Leroy. S Hoersch. A de Daruvar. C Reich. A Franchini. J Tamames. A Valencia. C Ouzounis. and C Sander. Automated genome sequence analysis and annotation. Bioinformatics 1999 15: 391-412.

Gary D. Bader. Christopher W.V. Hogue, *BIND - A data specification for storing and describing biomolecular interactions, molecular complexes and pathways*, Bioinformatics, vol 16, no 5 (2000) 465-477.

E Barillot, U Leser, P Lijnzaad, C Cussat-Blanc, K Jungfer, F Guyon, G Vaysseix, C Helgesen, and P Rodriguez-Tome. A proposal for a standard CORBA interface for genome maps. Bioinformatics 1999 15: 157-169.

Greg Butler. Erich Bornberg-Bauer. Gosta Grahne, Franz Kurfess, Clement Lam, Joey Paquet, Isabel Rojas, Rajjan Shinghal, Lixin Tao, Adrian Tsang. The BioIT Projects: Internet, Database and Software Technology Applied to Bioinformatics. SSGRR'2000, International conference on advances in infrastructure for electronic business. science and education on the internet. July 31 -- August 5, 2000. Scuola Superiore G. Reiss Romoli SpA, Coppoto, Italy.

Grady Booch, Ivar Jacobson, and James Rumbaugh, The Unified Modeling Language User Guide. Addison-Wesley, 1999.

I-Min A. Chen and Victor M. Markowitz, An overview of the Object Protocol Model

(OPM) and the OPM data management tools, Information Systems, vol.20, No. 5 (1995) 393-418.

Hui-Hsien Chou and Michael H. Holmes. DNA Sequence Quality Trimming and Vector Removal. Bioinformatics. 2001. 17:12. pp. 1093-1104.

Mike Dean, Dan Connolly, Frank van Harmelen, James Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language 1.0 Reference. W3C Working Draft, 29 July 2002.

Simon Dear, Richard Durbin, Ladeana Hillier, Gabor Marth, Jean Thierry-Mieg, and Richard Mott. Sequence Assembly with CAFTOOLS. Genome Res. 1998 Mar;8(3):260-7.

R.D. Dowell, R.M. Jokerst, A. Day, S.R. Eddy, and L. Stein. The Distributed Annotation System. BMC Bioinformatics 2001 2: 7. http://www.biomedcentral.com/1471-2105/2/7/abstract. http://www.biodas.org.

Selina S. Dwight, Midori A. Harris, Kara Dolinski, Catherine A. Ball, Gail Binkley, Karen R. Christie, Dianna G. Fisk, Laurie Issel-Tarver, Mark Schroeder, Gavin Sherlock, Anand Sethuraman, Shuai Weng, David Botstein, and J. Michael Cherry. Saccharomyces Genome Database (SGD) provides secondary gene annotation using the Gene Ontology (GO). Nucleic Acids Res. 2002 30: 69-72. http://genome-www.stanford.edu/Saccharomyces/

EcoCyc, Chapter 4 of *Bioinformatics: Databases and Systems*, edited by Stan Letovsky. Kluwer Academic Press, Boston, 1999.

Ensembl. http://www.Ensembl.org/.

Brent Ewing and Phil Green. Base-calling of automated sequencer traces using phred. II. Error probabilities. 1998. Genome Research 8:186-194.

Brent Ewing, LaDeana Hillier, Michael C. Wendl, and Phil Green. Base-calling of automated sequencer traces using phred. I. Accuracy assessment. 1998. Genome Research 8:175-185.

James C. French. Anita K. Jones, John L. Pfaltz (editors), Scientific Database Management, Technical Report 90-21, Department of Computer Science, University of Virginia, August 1990.

Dimitrij Frishman, Klaus Heumann, Arthur Lesk, Hans-Werner Mewes, Comprehensive, comprehensible, distributed and intelligent databases: current status, Bioinformatics, vol 14, no 7 (1998) 551-561.

Simsion, Graeme. Data Modeling Essentials: Analysis, Design, and Innovation. International Thompson Computer Press, 1994.

N. Goodman, S. Rozen. L. Stein, A. Smith, The LabBase system for data management in large scale biology research laboratoies, Bioinformatics 14, 7 (1998) 562-574. Chapter 23 of *Bioinformatics: Databases and Systems*, edited by Stan Letovsky. Kluwer Academic Press, Boston, 1999.

T. Hubbard, D. Barker, E. Birney, G. Cameron, Y. Chen, L. Clark, T. Cox, J. Cuff, V. Curwen, T. Down, R. Durbin, E. Eyras, J. Gilbert, M. Hammond, L. Huminiecki, A. Kasprzyk, H. Lehvaslaiho, P. Lijnzaad, C. Melsopp, E. Mongin, R. Pettett, M. Pocock, S. Potter, A. Rust, E. Schmidt, S. Searle, G. Slater, J. Smith, W. Spooner, A. Stabenau, J. Stalker, E. Stupka, A. Ureta-Vidal, I. Vastrik and M. Clamp, The

Ensembl genome database project, Nucleic Acids Research. 2002, Vol. 30, No. 1 38-41. http://www.Ensembl.org.

T. Ideker. V. Thorsson. A. F. Siegel. and L. Hood. Testing for differentially-expressed genes by maximum-likelihood analysis of microarray data. Journal of Computational Biology 7 (6) 805-817 (2001). Institute for Systems Biology DNA Microarray Data Processing (http://www.systemsbiology.org/ArrayProcess/index.html)

J. T. Inman. H. R. Flores, G. D. May, J. W. Weller, C. J. Bell. A high-throughput distributed DNA sequence analysis and database system. IBM Systems Journal, vol 40, No 2, 2001

R. Jasper and M. Uschold. A Framework for Understanding and Classifying Ontology Applications. In Twelfth Workshop on Knowledge Acquisition Modeling and Management KAW'99, 1999. Published on-line http://sern.ucalgary.ca/KSI/KAW/KAW99/.

Peter D. Karp, An ontology for biological function based on molecular interactions. Bioinformatics, vol 16, no 3 (2000) 269-285.

KEGG, Chapter 5 of *Bioinformatics: Databases and Systems*, edited by Stan Letovsky. Kluwer Academic Press, Boston, 1999.

Stan Letovsky (editor), Bioinformatics: Databases and Systems. Kluwer Academic Press, Boston, 1999.

Gregory McFarland, Andres Rudmik, and David Lange. Object-Oriented Database Management Systems Revisited. Technical report to Data & Analysis center for Software, 31 January 1999.

Tim McLellan, Data Modeling: Finding the Perfect Fit. 1995.

http://www.islandnet.com/~tmc/html/articles/datamodl.htm

Metadata Ad Hoc Working Group. Federal Geographic Data Committee Content Standard for Digital Geospatial Metadata (CSDGM)Metadata.

http://www.fgdc.gov/metadata/metadata.html

Microarray Gene Expression Data Society (MGED). http://www.mged.org/

NCBI datamodel of biological sequence.

http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SDKDOCS/INDEX.HTML.

J. No, R. Thakur, D. Kaushik, L. Freitag, A. Choudhary, A scientific data management system for irregular applications, Proc. of the Eighth International Workshop on Solving Irregular Problems in Parallel (Irregular 2001), April 2001

H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, M, Kanehisa, KEGG: Kyoto encyclopedia of genes and genomes, Nucleic Acids Research, vol 27, no 1 (1999) 29-34.

Norman W. Paton, Shakeel A. Khan, Andrew Hayes, Fouzia Moussouni, Andy Brass, Karen Eilbeck, Carole A. Goble, Simon J. Hubbard, Stephen G. Oliver, *Conceptual modeling of genomic data*, Bioinformatics, vol 16, no 6 (2000) 548-557.

Norman W. Paton, Shakeel A. Khan, Andrew Hayes, Fouzia Moussouni, Andy Brass, Karen Eilbeck, Carole A. Goble, Simon J. Hubbard, Stephen G. Oliver, Conceptual modeling of genomic data, Bioinformatics, vol 16, no 6 (2000) 548-557.

John Pfaltz, R.F. Haddleton, J.C. French, Scalable, parallel, scientific databases, 10th

International Conference on Scientific and Statistical Database Management, IEEE Computer Society, 1998, pp. 4-11.

Paul T Spellman. Michael Miller. Jason Stewart. Charles Troup, Ugis Sarkans, Steve Chervitz. Derek Bernhart. Gavin Sherlock. Catherine Ball. Marc Lepage. Marcin Swiatek. WL Marks. Jason Goncalves. Scott Markel. Daniel Iordan. Mohammadreza Shojatalab. Angel Pizarro. Joe White. Robert Hubley. Eric Deutsch. Martin Senger. Bruce J Aronow. Alan Robinson. Doug Bassett. Christian J Stoeckert Jr. Alvis Brazma. Design and implementation of microarray gene expression markup language (MAGE-ML), Genome Biology 2002 3(9): research0046.1-0046.9

Saccharomyces genome database. http://genome-www.stanford.edu/Saccharomyces/.

R. Stevens, C.A. Goble, and S. Bechhofer. Ontology-based Knowledge Representation for Bioinformatics. Briefings in Bioinformatics, 1(4):398-416, November 2000.

Toby J. Teory, Database Modeling & Design: The Basic Principles, 3rd ed. Morgan Kaufmann Publishers. Inc., 1999.

The Gene Ontology Consortium. 2001. Creating the gene ontology resource: design and implementation. Genome Research 11:1425-1433. http://www.geneontology.org

The Knowledge Engineering Review, 13(1):31-89, 1998.

The Phred/Phrap/Consed System Home Page (http://www.phrap.org/)

M. Uschold, M. King, S. Moralee, and Y. Zorgios. The Enterprise Ontology. The knowledge Engineering Review, 13(1):31-89, 1998..

Web service. http://www.service-architecture.com.

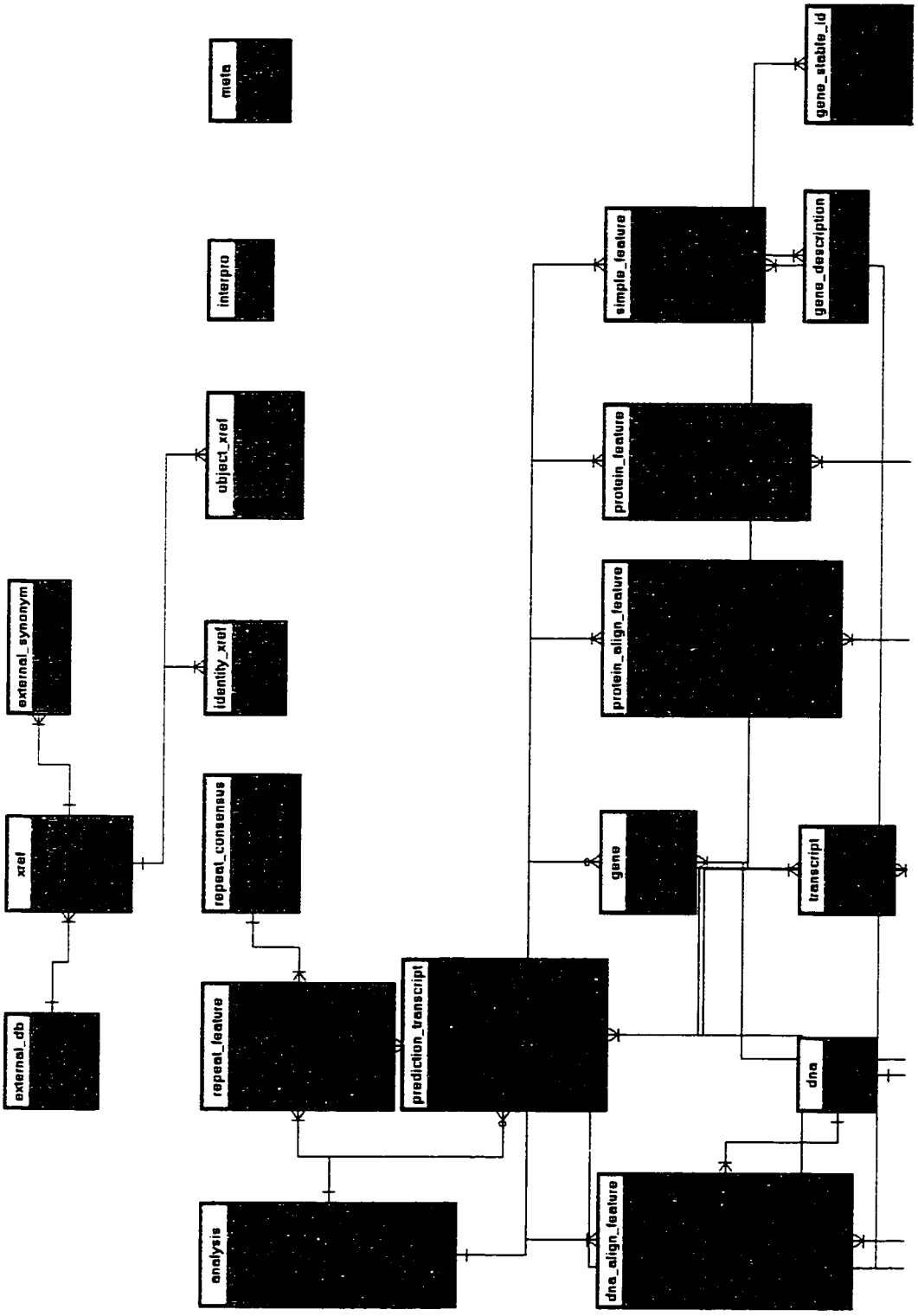
R. Williams, P. Messina, F. Gagliardi, J. Darlington, and G. Aloisio, Report on the EU-US Workshop on Large Scientific Databases, TR CACR - 179, Caltech Center for Advanced Computing Research, October 1999.

http://www.cacr.caltech.edu/euus/report.html

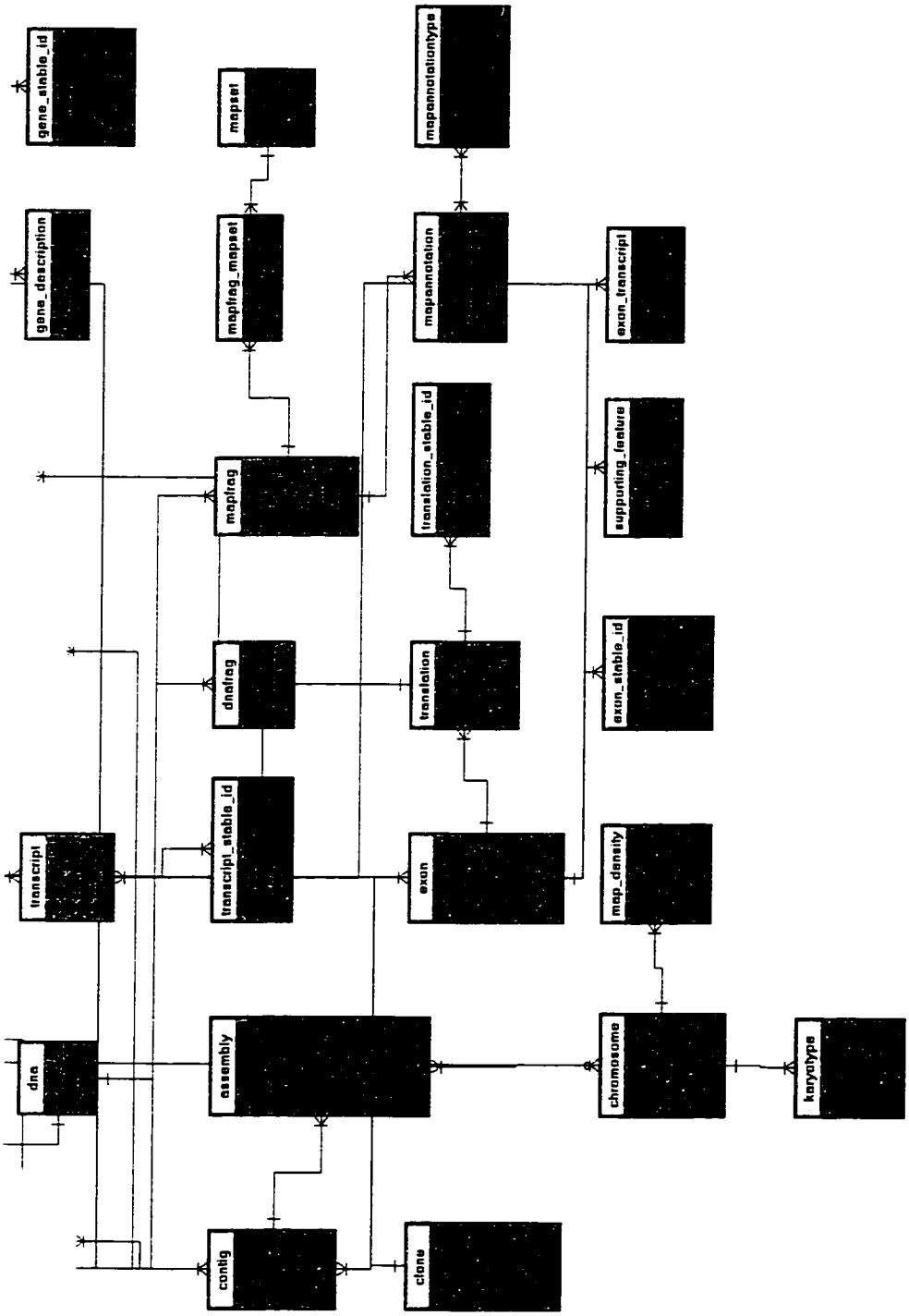
Workshop report. Data management for marine geology and geophysics. Tools for Archiving, Analysis and Visualization. WORKSHOP REPORT, LA JOLLA, CALIFORNIA. MAY 14-16, 2001.

http://humm.whoi.edu/DBMWorkshop/data_mgt_report.low.pdf

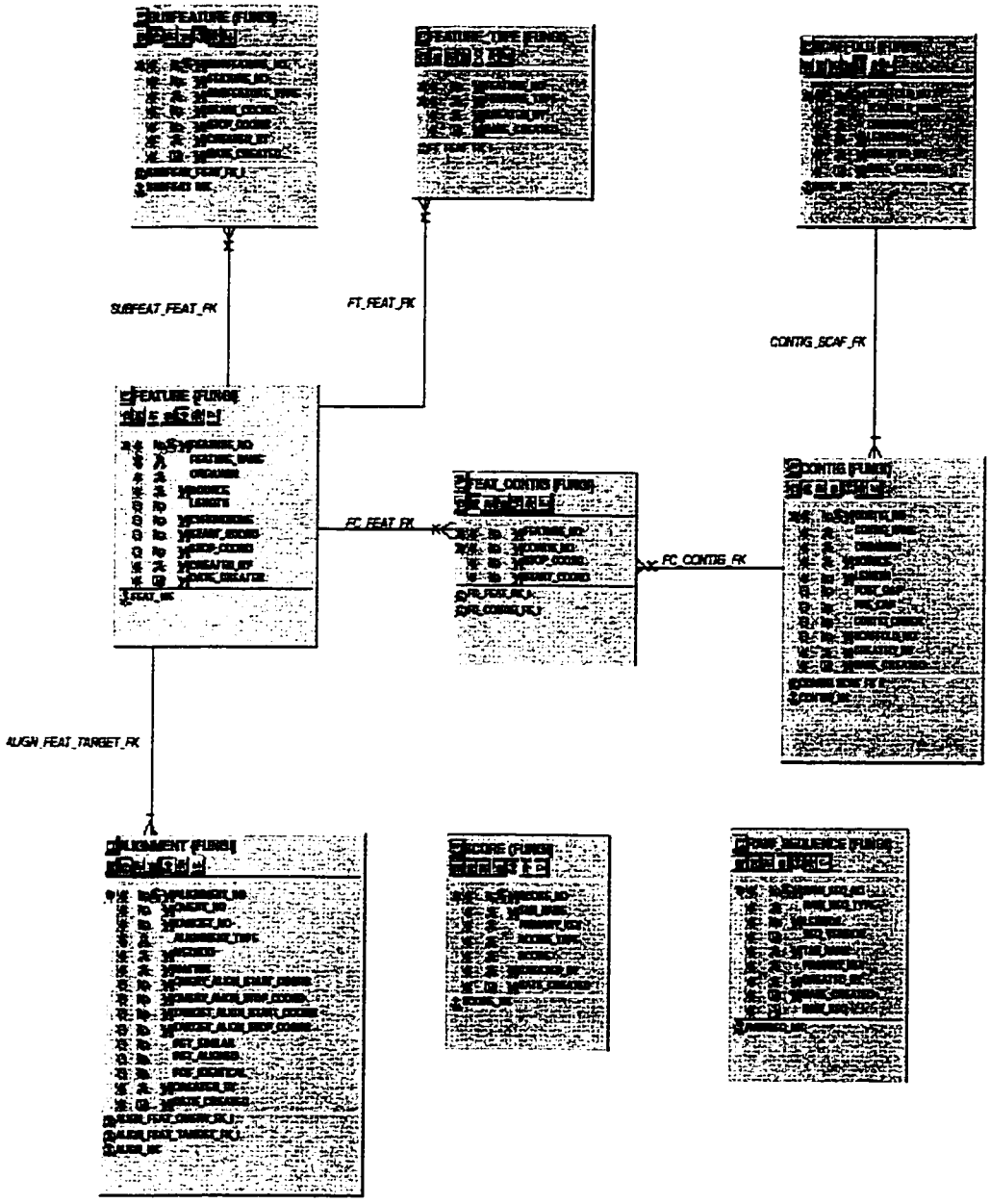
Appendix A1 Schema Diagram 1 of Ensembl (9.30.1)



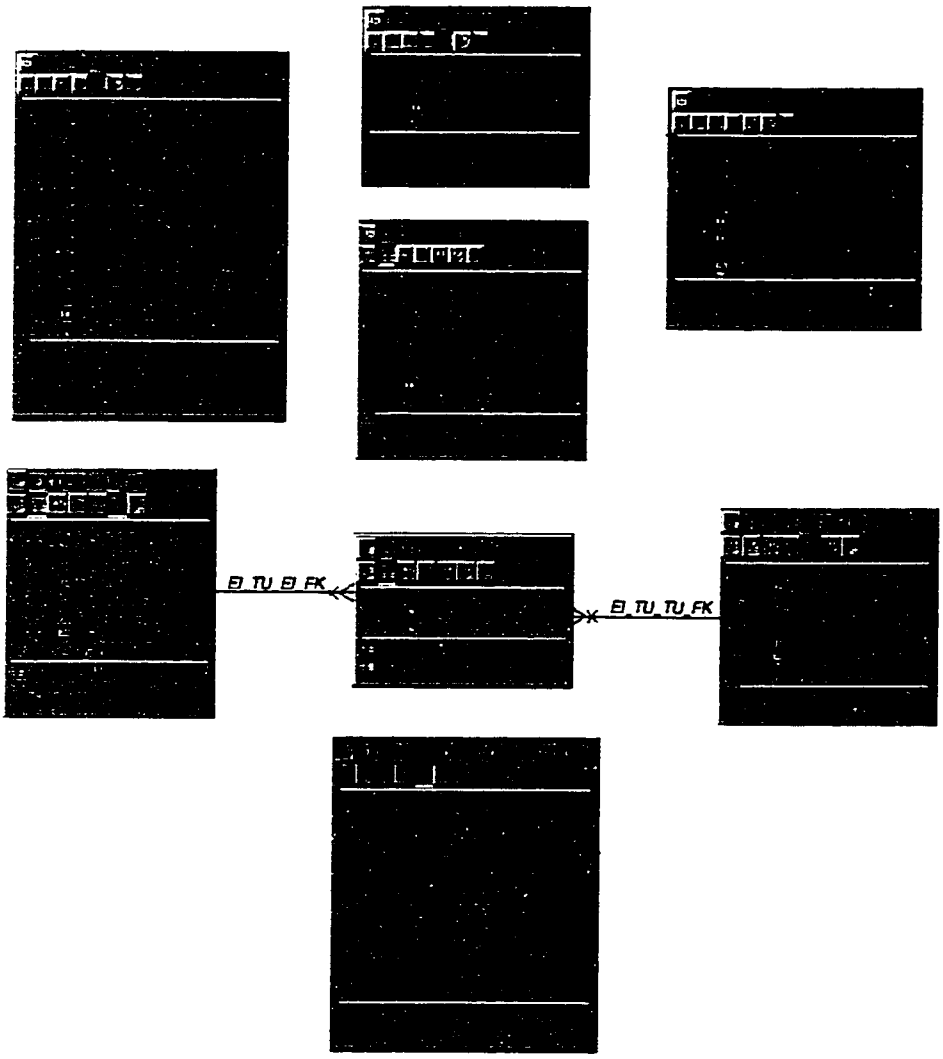
Appendix A2 Schema Diagram 2 of Ensembl (9.30.1)



Appendix B1 Schema Diagram 1 of SGD (Dec.2002)



Appendix B2 Schema Diagram 2 of SGD (Dec.2002)



Appendix C Schema Diagram of Pipeline 1

chromatogram

Column	Type	Description
chromatogram_id	int(10) unsigned not null	Assigned unique identifier for chromatogram; auto increment
name	varchar(40) not null	file name of the chromatogram
organism	varchar(40) not null	The organism belonging to the chromatogram.
primer_id	varchar(40)	primer used for sequencing
dye	varchar(40)	dye used for sequencing
chemistry	varchar(40)	chemistry used for sequencing
sequencing_machine	varchar(40)	sequencing machine used for sequencing
chromatogram_seq	mediumblob not null	
creator	varchar(40) not null	Person who entered the record into the database.
date_created	datetime not null	Date the record was entered into database

Key name	Columns	Other
PRIMARY	chromatogram_id ;	UNIQUE
file_name	name ;	-
read	read_id ;	-

read

Column	Type	Description
read_id	int(10) unsigned not null	Assigned unique identifier for read; auto increment
name	varchar(40) not null	file name of the read
organism	varchar(40) not null	The organism belonging to the read.
length	int(10) not null	The length of the read in nucleotides.
analysis_id	int(10) unsigned not null	**FK**:analysis:analysis_id
experiment_id	int(10) unsigned not null	**FK**:experiment:experiment_id
chromatogram_id	int(10) unsigned not null	**FK**:chromatogram:chromatogram_id
dna_id	int(10) unsigned not null	**FK**:dna:dna_id
dna_quality_id	int(10) unsigned not null	**FK**:dna_quality:dna_quality_id

Key name	Column	Other
PRIMARY	read_id ;	UNIQUE
file_name	name ;	-
chromatogram	chromatogram_id;	-
dna	dna_id;	-

[redacted]

[redacted]	[redacted]	[redacted]
trimmed_read_id	int(10) unsigned not null	Assigned unique identifier for trimmed_read; auto increment
Name	varchar(40) not null	file name of the trimmed_read
Organism	varchar(40) not null	The organism belonging to the trimmed_read.
length	int(10) unsigned not null	The length of the read in nucleotides.
seq_vector_id	int(10) unsigned not null	**FK**:seq_vector:seq_vector_id
analysis_id	int(10) unsigned not null	**FK**:analysis:analysis_id
read_id	int(10) unsigned not null	**FK**:read:read_id
dna_id	int(10) unsigned not null	**FK**:dna:dna_id
dna_quality_id	int(10) unsigned not null	**FK**:dna_quality:dna_quality_id

[redacted]	[redacted]	[redacted]
PRIMARY	read_id ;	UNIQUE
file_name	name ;	-
read	read_id ;	-
dna	dna_id ;	-

[redacted]

[redacted]	[redacted]	[redacted]
precontig_id	int(10) unsigned not null	Assigned unique identifier for precontig; auto increment
name	varchar(40) not null	file name of the precontig
length	int(10) not null	The length of the read in nucleotides.
version	int(10) not null	version of the precontig
analysis_id	int(10) unsigned not null	**FK**:analysis:analysis_id
creator	varchar(40) not null	Person who entered the record into the database.
date_created	datetime not null	Date the record was entered into database

[redacted]	[redacted]	[redacted]
PRIMARY	precontig_id ;	UNIQUE
file_name	name ;	-

APPENDIX

preassembly

Column	Type	Description
precontig_id	int(10) unsigned not null	Assigned unique identifier for precontig; auto increment
precontig_start	int(10) not null	start of trimmed_read in the coordinates relative to the precontig
precontig_end	int(10) not null	end of trimmed_read in the coordinates relative to the precontig
trimmed_read_id	int(10) unsigned not null	**FK**:trimmed_read:trimmed_read_id
trimmed_read_start	int(10) not null	start of trimmed_read in the preassembly
trimmed_read_end	int(10) not null	end of trimmed_read in the preassembly
read_orientation	int(10) not null	the orientation of trimmed read

Key name	Column	Other
PRIMARY	trimmed_read_id ;	UNIQUE

contig

Column	Type	Description
contig_id	int(10) unsigned not null	Assigned unique identifier for contig; auto increment
Name	varchar(40) not null	file name of the contig
Organism	varchar(40) not null	the organism the contig is used.
Length	int(10) not null	The length of the read in nucleotides.
Version	int(10) not null	version of the contig
analysis_id	int(10) unsigned not null	**FK**:analysis:analysis_id
dna_id	mediumtxt not null	**FK**:dna:dna_id, concensus dna
dna_quality_id	mediumtxt not null	**FK**:dna_quality:dna_quality_id; concensus dna quality

Key name	Column	Other
PRIMARY	contig_id ;	UNIQUE
file_name	name ;	-
consensus	consensus_id ;	-

APPENDIX

Column	Type	Description
contig_id	int(10) unsigned not null	Assigned unique identifier for contig; auto increment
contig_start	int(10) not null	start of trimmed_read in the coordinates relative to the contig
contig_end	int(10) not null	end of trimmed_read in the coordinates relative to the contig
trimmed_read_id	int(10) unsigned not null	**FK**:trimmed_read:trimmed_read_id
trimmed_read_start	int(10) not null	start of trimmed_read in the preassembly
trimmed_read_end	int(10) not null	end of trimmed_read in the preassembly
read_orientation	int(10) not null	the orientation of trimmed read
creator	varchar(40) not null	Person who entered the record into the database.
date_created	datetime not null	Date the record was entered into the database.

Key name	Columns	Other
PRIMARY	trimmed_read_id ;	UNIQUE
contig_id	contig_id, contig_start ;	-

dna

Column	Type	Description
dna_id	int(10) unsigned not null	Assigned unique identifier for read; auto increment
dna_type	enum("read", "trimmed_read", "contig_consensus")	Type of dna
dna_quality_id	int(10) unsigned not null	**FK**:dna_quality:dna_quality_id
sequence	mediumtext not null	The actual sequence of data
creator	varchar(40) not null	Person who entered the record into the database.
date_created	datetime not null	Date the record was entered into the database.

Key name	Columns	Other
PRIMARY	dna_id ;	UNIQUE

APPENDIX

dna_quality		
Column	Type	Description
dna_quality_id	int(10) unsigned not null	Assigned unique identifier for read; auto increment
dna_type	enum("read", "trimmed_read", "contig_consensus")	type of dna
sequence_quality	mediumtext not null	the actual quality of sequence.
Key name	Column	Other
PRIMARY	dna_id ;	UNIQUE

Experiment		
Column	Type	Description
experiment_id	int(10) unsigned not null	Assigned unique identifier for feature; auto increment
place	varchar(40) not null	place the experiment is performed.
experimenter	varchar(40) not null	people who did the experiment.
protocol	mediumtext not null	description of the experimental protocol of producing the data.
date_performed	datetime not null	date the experiment was performed.
Key name	Column	Other
PRIMARY	experiment_id ;	UNIQUE

analysis		
Column	Type	Description
analysis_id	int(10) unsigned not null	Assigned unique identifier for analysis; auto increment
logic_name	varchar(40) not null	name of the analysis.
program	varchar(80) not null	program for the analysis
program_version	varchar(40) not null	version of the program
program_file	varchar(80) not null	program file
parameters	varchar(80) not null	parameters of the program
description	mediumtext not null	description of the program, such as source, feature etc.
date_created	datetime not null	date the record was entered into database
Key name	Column	Other
PRIMARY	analysis_id ;	UNIQUE

Column	Type	Description
seq_vector_id	int(10) unsigned not null	Assigned unique identifier for seq_vector_id; auto increment
name	varchar(40) not null	name of the vector
version	int(10) not null	version of the vector
sequence	mediumtext not null	sequence of the vector
description	mediumtext not null	description of the vector, the use of the vector in the sequencing process.
date_created	datetime not null	date the record was entered into database

Key name	Column	Other
PRIMARY	seq_vector_id	UNIQUE

Glossary

2_point_data

This refers to data generated by tetrad analysis of a cross in which the segregation of 2 genetic markers is followed. This data yields the distance between the 2 markers (usually mutant alleles of genes) on the genetic map.

AceDB

AceDB was the database software used by SGD. However, currently SGD has moved over to ORACLE relational database and is not using AceDB any more.

Alignment

The process of lining up two or more sequences to achieve maximal levels of identity (and conservation, in the case of amino acid sequences) for the purpose of assessing the degree of similarity and the possibility of homology. For example, SCD25 protein sequence can be aligned with Chain S, Complex Of Human H-Ras With Human Sos-1.

Annotation

A combination of comments, notations, references, and citations, either in free format or utilizing a controlled vocabulary, that together describes all the experimental and inferred information about a gene or protein. Annotations can also be applied to the description of other biological systems. Batch, automated annotation of bulk biological sequence is one of the key uses of Bioinformatics tools.

Attribute

GLOSSARY

An attribute is any detail that serves to identify, qualify, classify, quantify, or otherwise express the state of an entity occurrence or a relationship. Attributes are specific pieces of information which need to be known or held.

BAC clone

Bacterial artificial chromosome vector carrying a genomic DNA insert, typically 100–200 kb. Most of the large-insert clones sequenced in the project were BAC clones.

BLAST

A set of programs, used to perform fast similarity searches. Nucleotide sequences can be compared with nucleotide sequences in a database using BLASTN, for example. Complex statistics are applied to judge the significance of each match. Reported sequences may be homologous to, or related to the query sequence. The BLASTP program is used to search a protein database for a match against a query protein sequence. There are several other flavors of BLAST.

cDNA

Complementary DNA. It is a DNA molecule made by copying RNA with reverse transcriptase; usually abbreviated cD NA.

Chromosome

A linear end-to-end arrangement of genes and other DNA, sometimes with associated protein and RNA.

Conceptual Data Model (CDM)

A CDM represents the overall logical structure of a database, which is independent of any software or data storage structure. A conceptual model often contains data objects not yet implemented in the physical databases. It gives a formal representation of the data needed to run an enterprise or a business activity.

Consensus Sequence

A single sequence delineated from an alignment of multiple constituent sequences that represents a "best fit" for all those sequences. A "voting" or other selection procedure is used to determine which residue (nucleotide or amino acid) is placed at a given position in the event that not all of the constituent sequences have the identical residue at that position. For example, for one position, three methods suggest it would be alpha helix, and one method suggests it would be coil, by consensus, it should be alpha helix.

Contig

A length of contiguous sequence assembled from partial, overlapping sequences, generated from a "shotgun" sequencing project. Contigs are typically created computationally, by comparing the overlapping ends of several sequencing reads generated by restriction enzyme digestion of a segment of genomic DNA. The creation of contigs in the presence of sequencing errors, ambiguities and the presence of repeats is one of the most computationally challenging aspects of the role of Bioinformatics in genome analysis.

Curation

The process of collecting and compiling data about DNA sequences.

GLOSSARY

DDBJ

DNA Bank of Japan. DDBJ has been functioning as one of the International DNA Databases, including EBI (European Bioinformatics Institute; responsible for the EMBL database) in Europe and NCBI (National Center for Biotechnology Information; responsible for GenBank database) in the USA as the two other members.

DNA (deoxyribonucleic acid)

The chemical that forms the basis of the genetic material in virtually all organisms. DNA is composed of the four nitrogenous bases Adenine, Cytosine, Guanine, and Thymine, which are covalently bonded to a backbone of deoxyribose-phosphate to form a DNA strand. Two complementary strands (where all Gs pair with Cs and As with Ts) form a double helical structure which is held together by hydrogen bonding between the cognate bases.

DNA microarrays

The deposition of oligonucleotides or cDNAs onto an inert substrate such as glass or silicon. Thousands of molecules may be organized spatially into a high-density matrix. These DNA chips may be probed to allow expression monitoring of many thousands of genes simultaneously. Uses include study of polymorphisms in genes, de novo sequencing or molecular diagnosis of disease.

Domain (protein)

A region of special biological interest within a single protein sequence. However, a domain may also be defined as a region within the three-dimensional structure of a

protein that may encompass regions of several distinct protein sequences that accomplishes a specific function. A domain class is a group of domains that share a common set of well-defined properties or characteristics. For example, beta-Grasp domain and an OB-fold domain.

Element dimensions

The physical dimensions of each features

EMBL

European Molecular Biology Labs. The EMBL Nucleotide Sequence database is a comprehensive database of DNA and RNA sequences. The database is produced in collaboration with GenBank and the DNA Database of Japan (DDBJ).

Ensembl Genes/cDNAs/Peptides

The "Ensembl Genes/transcripts/peptides" have been predicted by the Ensembl gene prediction pipeline. They are referred to as "Ensembl predicted genes/cDNAs/peptides" (sometimes called also "Ensembl confirmed genes/cDNAs/peptides"). The set consists of "**known** genes" and "**novel** genes". "**Known** genes" are the ones for which there is near-full-length cDNA and/or protein sequence already available in the public sequence databases, and Ensembl has been able to assign an established gene symbol or other identifier. "**Novel** genes" are predicted on the basis of identity to ESTs and/or similarity to protein sequences - all Ensembl predicted genes are supported by evidence - but no established identifier can reliably be assigned.

Entity

An *entity* is a thing or object of significance to the business, whether real or imagined, about which the business must collect and maintain data, or about which information needs to be known or held. An entity may be a tangible or real object like a person or a building; it may be an activity like an appointment or an operation; it may be conceptual as in a cost center or an organizational unit.

Entity Relationship Diagrams

An ERD is a pictorial representation of the entities and the relationships between them. It allows the participants in the meeting to easily see the information structure of the application. Later, the project team uses the ERD to design the database and tables.

Entrez

The Entrez Search System was developed by NCBI. Entrez allows you to retrieve molecular biology data and bibliographic citations from integrated nucleotide (GenBank, DDBJ, EMBL), protein (Swiss-Prot, PIR, PRF, PDB), and bioliographic (PubMed) databases. Within SGD database pages, external links are provided to one or more of these databases.

Enzyme

A class of proteins that are capable of catalyzing chemical reactions (the making or breaking of chemical bonds). They do so by orienting their substrates into a suitable geometry in a particular location (the active site) where electrophilic or nucleophilic amino acid residues can participate in the reaction. Enzymes are protein catalyst that speeds up chemical reactions that would otherwise be prohibitively slow under

physiological conditions. For example, alcohol dehydrogenase, glucuronolactone reductase.

EST (expressed sequence tag)

Brief pieces of unique sequences that are transcribed. These serve as markers along the genome

FASTA

Program used to search simultaneously both protein and DNA sequence databases (Pearson and Lipman, 1988). FASTA uses a fast search to initially identify sequences with a high degree of similarity to the query sequence and then conducts a second comparison on the selected sequences. FASTA is slower than BLAST, but is more sensitive/sometimes yields different results.

Fingerprint clone contigs

Contigs produced by joining clones inferred to overlap on the basis of their restriction digest fingerprints.

Foreign Key

Column or combination of columns whose values are required to match a primary key in some other table.

FPC

Fingerprint clone. Every clone is digested and its fragment sizes are compared with other clones. Clones with equally sized fragments are considered to cover overlapping regions of the genome DNA. This neighbouring information is used to

build contigs of clones. From these contigs, minimum tiling paths of clones are selected for sequencing.

GenBank

GenBank is the DNA sequence database sponsored by the US National Institutes of Health. GenBank is produced in collaboration with EMBL and DDBJ. There is also a searchable DNA sequence database maintained by SGD (Yeast GenBank) that contains the subset of DNA sequences submitted to GenBank that have been derived from *S. cerevisiae* DNA. It includes results of the systematic sequencing as well as results from individual laboratories

Gene expression

The conversion of information from gene to protein via transcription and translation.

Gene Expression Measurements

Distinguish among three sets of data processing: image (raw data), image analysis and quantification, gene expression data matrix (normalized and summarized data).

Gene families

Subsets of genes containing homologous sequences which usually correlate with a common function. For example, alpha actinin family, cytokine family.

Genetic map

A genome map in which polymorphic loci are positioned relative to one another on the basis of the frequency with which they recombine during meiosis. The unit of distance is centimorgans (cM), denoting a 1% chance of recombination.

GLOSSARY

Genome

The complete genetic content of an organism: For example, human genome, C. elegans genome.

Genomic DNA (sequence)

DNA sequence typically obtained from mammalian or other higher-order species, which includes both intron and exon sequence (coding sequence), as well as non-coding regulatory sequences such as promoter, and enhancer sequences.

Genomics

The analysis of the entire genome of a chosen organism

Genscan Genes/cDNAs/Peptides

The "Genscan Genes/cDNAs/Peptides are solely based on predictions from the Genscan gene prediction program.

Hybridization protocol

Documentation of the set of steps taken in the hybridization, including: solution (e.g. concentration of solutes); blocking agent and concentration used; wash procedure; quantity of labeled target used; time; concentration; volume, temperature, and description of the hybridization instruments

Homology

Two or more biological species, systems or molecules that share a common evolutionary ancestor. (general) Two or more gene or protein sequences that share a significant degree of similarity, typically measured by the amount of identity (in the

case of DNA), or conservative replacements (in the case of protein), that they register along their lengths. Sequence "homology" searches are typically performed with a query DNA or protein sequence to identify known genes or gene products that share significant similarity and hence might inform on the ancestry, heritage and possible function of the query gene. For example, the clk-1 of C. elegans and Coq7/Cat5 has homology.

IDL:

Interface definition language.

Karyotype:

A karyotype shows the metaphase chromosomes of an individual cell, arranged in pairs and sorted according to size. There are 22 pairs of autosomal (non-sex) chromosomes and a pair of sex chromosomes (labeled "X" and "Y" in the picture). Many human disorders and malformations are a direct result of missing, broken, or extra chromosomes. In a karyotype, scientists called cytogeneticists can recognize and identify many of these large chromosomal abnormalities.

Library

A large collection of cloning vectors containing a complete (or nearly complete) set of fragments of the genome of an organism

Locus

A "locus" most often is a gene, characterized by a mutant phenotype or by a DNA sequence, which has been either genetically mapped or otherwise localized (e.g. by

DNA sequence comparison or hybridization) to a particular spot in the genome. A locus may also be a DNA sequence feature such as a centromere.

Many-to-many relationship

In a many-to-many relationship, multiple occurrences of one entity are related to one occurrence of another, and vice versa. Many-to-many relationships cannot be directly converted into database tables and relationships.

Map

If a locus has been genetically mapped, the "ORF Map" and "Genetic position" under the Sequence Coordinates section of the locus page will display details of the locus/feature. The Roman numeral to the right of "Map" indicates the chromosome to which the locus maps. The number to the right of "Genetic Position" indicates the map position of the locus (in centimorgans) from the centromere, where negative numbers indicate distances to the left of the centromere (the left arm) and positive numbers correspond to right arm distances.

Mapping_data

This displays links to all of the 2-point cross tetrad data where the locus was used as one of the markers.

Meta data:

Meta data in EBI is retrieval of valid *controlled vocabulary* values used in the EBI databases. For IDL http://corba.ebi.ac.uk/idl/meta.txt, and doc, see http://corba.ebi.ac.uk/idl/doc/meta.html

Metadata

Metadata or "data about data" describe the content, quality, condition, and other characteristics of data. For instance, a table in a database could have a lengthy business description. The description is metadata about the data in the database.

Motif

A conserved element of a protein sequence alignment that usually correlates with a particular function. Motifs are generated from a local multiple protein sequence alignment corresponding to a region whose function or structure is known. It is sufficient that it is conserved, and is hence likely to be predictive of any subsequent occurrence of such a structural/functional region in any other novel protein sequence. For example, calcium binding EF-hand in calmodulin, a ubiquitions molecule undergoing Ca-dependent conformational changes.

Multiple (sequence) alignment

A Multiple Alignment of k sequences is a rectangular array, consisting of characters taken from the alphabet A, that satisfies the following conditions: There are exactly k rows; ignoring the gap character, row number i is exactly the sequence s_i; and each column contains at least one character different from "-". In practice multiple sequence alignments include a cost/weight function, that defines the penalty for the insertion of gaps (the "-" character) and weights identities and conservative substitutions accordingly. Multiple alignment algorithms attempt to create the optimal alignment defined as the one with the lowest cost/weight score.

NCBI

The National Center for Biotechnology Information (NCBI) is part of the National Library of Medicine (NLM) in the National Institutes of Health (NIH). Its mission is to develop new information technologies to aid in the understanding of fundamental molecular and genetic processes that control health and disease. NCBI developed and maintains the Entrez Search System and PubMed database.

Normalized and summarized data

Several quantification tables are combined using data processing metrics to obtain the 'final' gene expression measurement table (gene expression data matrix) associated with the experiment

Object-Relational Database

Object databases combine the elements of object orientation and object-oriented programming languages with database capabilities. They provide more than persistent storage of programming language objects. Object databases extend the functionality of object programming languages (e.g., C++, Smalltalk, or Java) to provide full-featured database programming capability. The result is a high level of congruence between the data model for the application and the data model of the database. Object-relational databases are used in Bioinformatics to map molecular biological objects (such as sequences, structures, maps and pathways) to their underlying representations (typically within the rows and columns of relational database tables.) This enables the user to deal with the biological objects in a more intuitive manner, as they would in the laboratory, without having to worry about the underlying data model of their representation.

ODL

The data definition language of ODMG

One-to-one relationship

One-to-one relationships are between two entities where both are related to each other, once and only once for each instance of either.

OODBMS

Object-oriented database management system, sometimes shortened to *ODBMS* for *object database management system.* It is a database management system that supports the modeling and creation of data as objects.

OQL

Object query language, which supports the ODMG data model. It is complete and simple. It deals with complex objects without privileging the set construct and the select-from-where clause.

ORF

An ORF (Open Reading Frame) corresponds to a stretch of DNA that could potentially be translated into a polypeptide or RNA; i.e., it begins with an ATG "start" codon and terminates with one of the 3 "stop" codons. For an ORF to be considered as a good candidate for coding a bona fide cellular protein, a minimum size requirement is often set, e.g., many of the systematic sequencing groups define an ORF as a stretch of DNA that would code for a protein of 100 amino acids or more. An ORF is not usually considered equivalent to a gene or locus until there has been shown to be a phenotype associated with a mutation in the ORF, and/or an

mRNA transcript or a gene product generated from the ORF's DNA has been detected.

Orthologue/Ortholog

Orthologs are genes in different species that evolved from a common ancestral gene by speciation. Normally, orthologs retain the same function in the course of evolution. Identification of orthologs is critical for reliable prediction of gene function in newly sequenced genomes. (See also Paralogs.) For example, BMP2 sequences from rat, mouse, man, chicken and xenopus are orthologous since they all diverged after speciation events.

P1

P1 phage. PAC (P1 artificial chromosome) is a vector based on the P1 phage replication.

Pattern

Molecular biological patterns usually occur at the level of the characters making up the gene or protein sequence. A pattern language must be defined in order to apply different criteria to different positions of a sequence. In order to have position-specific comparison done by a computer, a pattern-matching algorithm must allow alternative residues at a given position, repetitions of a residue, exclusion of alternative residues, weighting, and ideally, combinatorial representation.

Pathways

Bioinformatics strives to define representations of key biological datatypes, algorithms and inference procedures, including sequences, structures, biological

pathways and reactions. Representing and computing with biological pathways requires ontologies for representing pathway knowledge; User interfaces to these databases; Physico-chemical properties of enzymes and their substrates in pathways; And pathway analysis of whole genomes including identifying common patterns across species and species differences. For example, metabolic pathway.

Paralog

Paralogs are genes related by duplication within a genome. Orthologs retain the same function in the course of evolution, whereas paralogs evolve new functions, even if these are related to the original one. For example, BMP2 and BMP4 genes are paralogous since they result from a gene duplication.

Physical map

A diagram showing the relative positions of physical landmarks in a DNA molecule; common landmarks include the positions of restriction sites and particular DNA sequences.

Profile

Sequence profiles are usually derived from multiple alignments of sequences with a known relationship, and consist of tables of position-specific scores and gap-penalties. Each position in the profile contains scores for all of the possible amino acids, as well as one penalty score for opening and one for continuing a gap at the specified position. Attempts have been made to further improve the sensitivity of the profile by refining the procedures to construct a profile starting from a given multiple alignment. Other representations for sequence domains or motifs do not necessarily

require the presence of a correct and complete multiple alignment, such as hidden Markov models.

Protein families

Sets of proteins that share a common evolutionary origin reflected by their relatedness in function which is usually reflected by similarities in sequence, or in primary, secondary or tertiary structure. Subsets of proteins with related structure and function.

P-value

In a BLAST search, a P-value refers to the probability of obtaining, by chance, a pairwise sequence comparison of the observed similarity given the length of the query sequence and size of the database searched. Thus, low P-values indicate sequence similarities of high significance.

Query (sequence)

A DNA, RNA of protein sequence used to search a sequence database in order to identify close or remote family members (homologs) of known function, or sequences with similar active sites or regions (analogs), from whom the function of the query may be deduced.

Raw data

The data that comes directly from the measurement device;

RDBMS

Relational database management system.

Relationship

A relationship is a named connection or association between entities. For example, in a CDM that manages human resources, the relationship Member links the entities Employee and Team, because employees can be members of teams. This relationship expresses that each employee works in a team and that each team has employees. An occurrence of a relationship corresponds to one instance of each of the two entities involved in the relationship. For example, the employee Martin working in the Marketing team is one occurrence of the relationship Member.

Secondary structure (protein)

The organization of the peptide backbone of a protein that occurs as a result of hydrogen bonds e.g alpha helix, Beta pleated sheet.

SGD

Saccharomyces Genome Database. The SGD project collects information and maintains a database of the molecular biology of the yeast *Saccharomyces cerevisiae*. This database includes a variety of genomic and biological information. SGD is funded by the National Center for Human Genome Research (NCGHR) at the U.S. National Institutes of Health. The SGD is in the Department of Genetics at the School of Medicine, Stanford University. The SGD Homepage is located at http://genome-www.stanford.edu/Saccharomyces/.

Similarity (homology) search

Given a newly sequenced gene, there are two main approaches to the prediction of structure and function from the amino acid sequence. Homology methods are the

most powerful and are based on the detection of significant extended sequence similarity to a protein of known structure, or of a sequence pattern characteristic of a protein family. Statistical methods are less successful but more general and are based on the derivation of structural preference values for single residues, pairs of residues, short oligopeptides or short sequence patterns. The transfer of structure/function information to a potentially homologous protein is straightforward when the sequence similarity is high and extended in length, but the assessment of the structural significance of sequence similarity can be difficult when sequence similarity is weak or restricted to a short region.

SNP

Single nucleotide polymorphism, or a single nucleotide position in the genome sequence for which two or more alternative alleles are present at appreciable frequency (traditionally, at least 1%) in the human population.

Structured Query Language (SQL)

SQL is a language used to communicate with a DBMS and provides a fairly common syntax for applications to use.

STS

A DNA sequence, present once per haploid genome, that can be amplified by the use of suitable oligonucleotide primers in the polymerase chain reaction in order to identify clones that contain the sequence.

Tile Path

The tile path track in human Ensembl shows the sequence entries (mostly BAC clones) that were used in the current assembly, and where they map on the assembly.

Unified Modeling Language (UML)

This is a modeling language created by the Rational Software company to implement the OOA&D methodology. The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

Vector

A DNA molecule, capable of replication, into which a gene or DNA segment is inserted by recombinant DNA techniques; a cloning vehicle.

Weight matrix

The density of binding sites in a gene or sequence can be used to derive a ratio of density for each element in a pattern of interest. The combined individual density ratios of all elements are then collectively used to build a scoring profile known as a weight matrix. This profile can be used to test the prediction of the identification of the selected pattern and the ability of the algorithm to discriminate them from non-pattern sequences.

YAC

In yeast, a cloning vector that can accept very large fragments of DNA; a chromosome introduced into yeast derived from such a vector and containing DNA from another organism.