ExperiBase:

An Integrated Software Architecture to Support Modern Experimental Biology

By

Shixin Zhang

B.S., Mechanical Engineering (1992), Tsinghua University
B.S. (Minor), Electronics & Computer Technology (1992), Tsinghua University
M.S., Mechanical Engineering (2001), Massachusetts Institute of Technology

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2004

Signature of Author.....
Department of Mechanical Engineering
April 1, 2004

Certified by.....
C. Forbes Dewey, Jr.
Professor of Mechanical Engineering and Bioengineering

Accepted by.....

Chairman, Department Committee on Graduate Students

ExperiBase: An Integrated Software Architecture to Support Modern Experimental Biology

By

Shixin Zhang

Submitted to the Department of Mechanical Engineering
on April 1, 2004 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Mechanical Engineering

Abstract

Over the past several years, the explosive growth of biological data generated by new high-throughput instruments has virtually begun to drown the biological community. There is no established infrastructure to deal with these data in a consistent and successful fashion. This thesis presents a new informatics platform capable of supporting a large subsection of the experimental methods found in modern biology. A consistent data definition strategy is outlined that can handle gel electrophoresis, microarray, fluorescence activated cell sorting, mass spectrometry, and microscopy within a single coherent set of information object definitions. A key issue for interoperability is that common attributes are made truly identical between the different methods. This dramatically decreases the overhead of separate and distinct classes for each method, and reserves the uniqueness for attributes that are different between the methods. Thus, at least one higher level of integration is obtained.

The thesis shows that rich object-oriented modeling together with object-relational database features and the uniform treatment of data and metadata is an ideal candidate for complex experimental information integration tasks. This claim is substantiated by elaborating on the coherent set of information object definitions and testing the corresponded database using real experimental data.

A first implementation of this work – ExperiBase – is an integrated software platform to store and query data generated by the leading experimental protocols used in biology within a single database. It provides: comprehensive database features for searching and classifying; web-based client interfaces; web services; data import and export capabilities to accommodate other data repositories; and direct support for metadata produced by analysis programs. Using JDBC, Java Servlets and Java Server Pages, SOAP, XML, and IIOP/CORBA's technologies, the information architecture is portable and platform independent.

The thesis develops an ExperiBase XML according to the single coherent set of information object definitions, and also presents a new way of database federation – translating heterogeneous database schemas into the common ExperiBase XML schema and then merging the output XML messages to get data federated.

ExperiBase has become a reference implementation of the I3C Life Science Object Ontologies group.

Thesis Committee:

Professor C. Forbes Dewey, Jr., Mechanical Engineering and Bioengineering
Mr. Paul Ferraiolo, Infinity Pharmaceuticals
Professor Ian W. Hunter, Mechanical Engineering and Biological Engineering
Dr. John S. (Pete) Wishnok, Biological Engineering Mass Spectrometry Lab
Dr. H. Steven Wiley, Pacific Northwest National Laboratory

Finally, I thank my family for their support over the past four years. Thanks to my Mom and Dad, and thanks to my wife, Xuhong, for making this experience a wonderful and happy one.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Over the past several years, both the generation and the analysis of biological experimental data are becoming increasingly widespread, and many fields of biology are moving incrementally toward high-throughout approaches. Techniques are also increasing in complexity as the relevant technologies evolve. The explosive growth of biological data generated by new high-throughput instruments has virtually begun to drown the biological community.

Every drug company and research laboratory faces the daunting task of dealing with mounting streams of experimental data from many diverse biological experimental methods. These methods have traditionally grown up along separate paths, and the scientific domain expertise associated with one experimental method may not overlap with that of other methods. In the end, the data must be used on collaborative analyses. These data cover a wide spectrum of traditional wet lab biology, proteomics, and clinical trials. The explosive growth of those data requires a standard to guide the capture, storage, and dissemination of them.

The need for experimental information integration is paramount in many biological disciplines, because of the large heterogeneity in both the types of data involved and in the diversity of approaches taken by biologists to study the same or correlated phenomena. A grand goal in many disciplines of biological research is to understand the mechanism of a biological process and how the interplay of different structural, chemical and electrical signals in the biological tissues gives rise to natural and disease processed. To achieve such a goal, however, it is essential to develop an integrated understanding of very different, but conceptually correlated studies and data produced from diverse biological subdisciplines. Most importantly:

Biologists may use high-throughput analysis pipelines to study a single research problem. The pipelines may consist of physical separation of samples by gel electrophoresis, size-exclusion and/or affinity chromatography, followed by mass spectrometric examination of separations and protein identification by bioinformatic analysis. Thus, they may wish to record the data generated from the above pipelines in an efficient organic way. The way should capture every detail of both the methods used and the data generated in the pipelines. The way should facilitate further data analyses.

Biologists may use different experimental methods to study different aspects of the same biological sample or verify the results using the different methods. Thus, for a given research problem, they may wish to integrate information about cell sample treatments, BCA (bicinchoninic acid) arrays for estimating the concentration of proteins, Western blot detections, and Flow Cytometry to determine the level of a factor expression in a cell culture.

Biologists study the same biological system from multiple perspectives. For example, in the study of calcium regulation, researcher A may take an anatomical approach, mapping the distribution of different isoforms of calcium regulatory proteins and the organelles that express them; a biochemist B may study signal transduction cascades and levels of protein activity using Western blots and assay systems; a

pharmacologist C may use a panel of channel blockers, agonists or antagonists to study the response in single cells to alterations in calcium regulation.[1]

However, it seems that the very heterogeneity makes the task of experimental information integration very difficult because two approaches studying different aspects of the same phenomena seems not to share common attributes in their schema description. Although several individual standards have been established or are being formed for some experiment methods, so far, the existing experimental databases either cannot support new data types generated from other experimental methods, or cannot interchange data between different databases. Even no database schema exists at all for some biological experimental methods.

The representation of gene and genome sequence data is fairly well standardized; the databases and tools for their analysis are widely used. A standard representation of both the methods used and the data generated in microarray experiments is the MIAME (Minimum Information about A Microarray Experiment)[2] guidelines for transcriptomics and the associated MAGE[3] (MicroArray Gene Expression) Object model[4] and XML implementation[5]. In recent years, we have also witnessed an increasing interest in proteomics. A standard research for proteomics experiments is PEDRo[6] (the Proteomics Experiment Data Repository). We noticed that there are many researches in different disciplines towards to build an exchange language or guide to facilitate the data capture, storage and dissemination, such as CDISC (the Clinical Data Interchange Standards Consortium) laboratory data interchange standard[7],[8], CytometryML[9] (Cytometry Markup Language), OME[10] (Open Microscopy Environment), and so on. However, all of these are designed and applied in a certain discipline; there is no established infrastructure to handle a large subsection of the various experimental protocols in a consistent and successful way. No common scientific ontology exists in the experimental biology.

The goal of this thesis is to present a new informatics platform to handle a large subsection of the experimental protocols that currently exist. A consistent data definition strategy is outlined that can handle gel electrophoresis, microarray, fluorescence activated cell sorting, mass spectrometry, and microscopy within a single coherent set of information object definitions. A key issue for interoperability is that common attributes are made truly identical between the different methods. This dramatically decreases the overhead of separate and distinct classes for each method, and reserves the uniqueness for attributes that are different between the methods. Thus, at least one higher level of integration is obtained. The architecture presented in the thesis can integrate different studies and analyses conducted by biologists performing different experiments, such that the integrated body of information can be queried and navigated across. The thesis shows that the rich object-oriented modeling together with object-relational database features and the uniform treatment of data and metadata is an ideal candidate for complex experimental information integration tasks. This claim is substantiated by elaborating on the coherent set of information object definitions and testing the corresponded database using real world data from a variety of biological experimental methods. In the thesis, it presents a simple UML (Unified Modeling Language) approach to express the inheritance of the various biological experimental data, describes XML (extensible markup language) schema and SQL (structured query language) implementations of those models, and discusses capture, storage, and dissemination strategies. The implementation

of this work is named as ExperiBase. ExperiBase is conceived as an integrated software platform capable of supporting a variety of experimental methods found in modern biology. It provides: comprehensive database features for searching and classifying; web-based client services; data import and export capabilities to accommodate other data repositories; and direct support for metadata produced by analysis programs. The software notably organizes in a database the raw data collected with experiments and link them to experimental procedure and analysis results. It provides an easy way for biologists to keep track of the numerous data and experimental settings, and more convenient to search and browse the experimental data. Also, because ExperiBase supports many different experimental methods, and the semantic contexts that support them, the creation of this software platform provides a major contribution to normalizing semantic support for these different methods. Using JDBC, Java Servlets and Java Server Pages, SOAP, XML, and IIOP/CORBA's technologies, the information architecture is portable and platform independent.

Chapter 2. Background

This chapter begins with understanding the biological analysis process, to illustrate why a database solution is needed and important in the process. Then introduces several existing or ongoing biological experimental databases, and reviews the recent researches of semantic systems for biology. The chapter is concluded with a description of current data, integration tools that have been applied in the biological area.

2.1 The Biological Analysis Process and Experiment Data

A biological analysis process usually involves three contents: acquiring data from biological experiments; applying data to biological models; and interpreting the results (Fig. 1). For a complex biological project, the processes may involve many persons or institutes. Sometimes an individual data set contains information of value to multiple biological areas[11]. The data could be available to participants anywhere, anytime if the data are open to public and disseminated in a suitable way. The methods of data communication such as transferring file copies may be convenient within a research group but are not feasible in the case that the project involves multiple groups and even different locations.

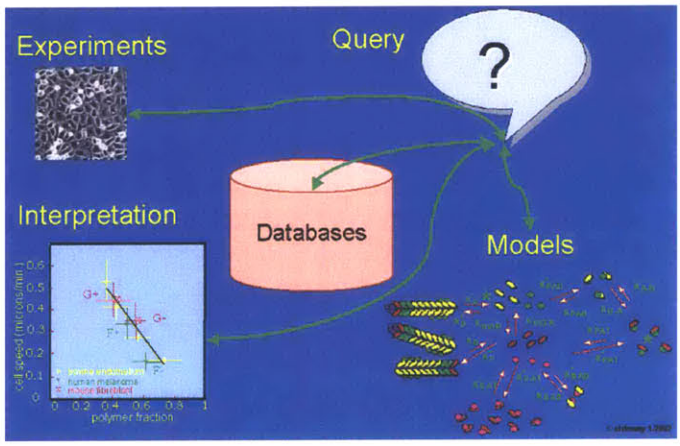


Figure 1 A picture of biological analysis process[12]

Large amounts of experimental data are produced every day[13,14] using experimental protocols, such as microarrays, high throughput kinase activity assays [15] and high throughput screening microscope images [16]. New high throughput methods (cDNA microarrays, antibody microarrays) are being developed to extend the number of signal parameters that are measured[17]. The data from these methods differ in thousands of ways. They may be one-time data or time course data, such as the picture below (Fig. 2a) showing time series data analysis of apoptosis. The data can be text files, Excel spreadsheets or other format files; can be tables, images, graphs (Fig. 2b) and videos as well. Only using a file system to manage and store these data is awkward and ineffective. New methods for storing and querying the data are required.

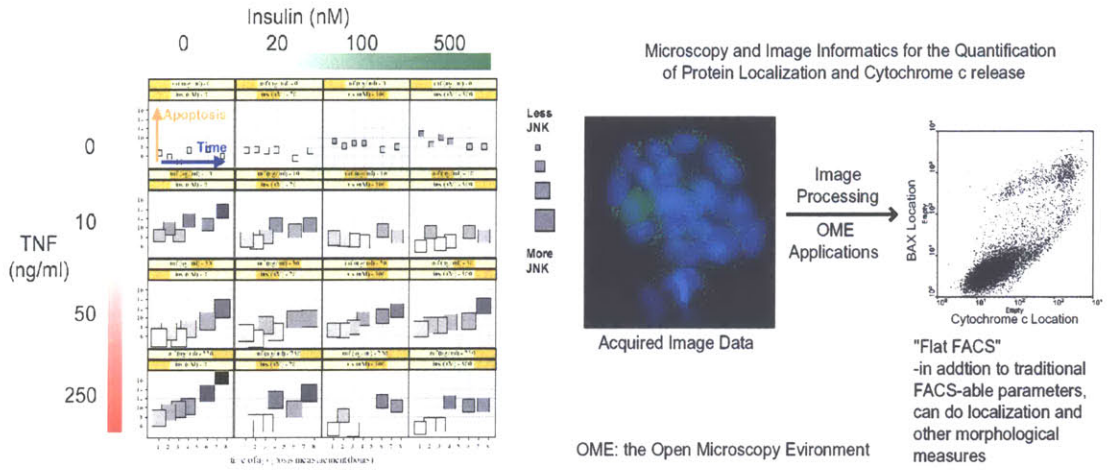


Figure 2 Examples of biological experimental data. (a). time series data analysis of gel electrophoresis; (b). image data got from flow cytometry. (Ref: Suzanne Gaudet, "Using quantitative immunoblots to assay the effects of insulin on signaling networks in TNF-treated HT-29 cells – cellular decision"; John Albeck, "Quantitative monitoring of apoptotic and early signaling responses to TNF and insulin. Symposium on Apoptosis Signaling Networks in Human Cells)

Continuing the diagram of Fig. 1, it illustrates that there is a strong coupling between experimental data and interpretive models in modern biology. A recent expression of the classical scientific method has been termed "the network biology approach"[18] (See Fig. 3). The models used for interpretation typically involve hundreds of states and kinetic parameters and coupled ordinary differential equations (ODEs) describing the biochemical reaction network[19,20,21]. The modelers need many sets of experimental data to test and improve their models and therefore predict the experimental results. The processes are recursive and iterative. Therefore, in the data-exchange diagram of Fig. 3, a database solution is the best way to manage, store and retrieve data. It plays a very important role in the loop. Good designs for the data models, databases, and software architectures will simplify the process of storing/retrieving data, it is then possible to share and reuse data in a more convenient and intelligent way. If database access is implemented with a good web-enabled user interface design, it can allow users to access and enter data from any location at anytime. Retrieving data from database is also more flexible than accessing a file system. One set of data may be suitable with ODEs, Bayesian Networks, Principle Component Analysis (PCA), and so on by using different querying process. With the database and corresponding interface collaborations, the generation of more sophisticated models of the signaling network will enable predictions of outcome to be made in the presence of incomplete information and will be useful for understanding quantitative dependencies between the nodes of the network.

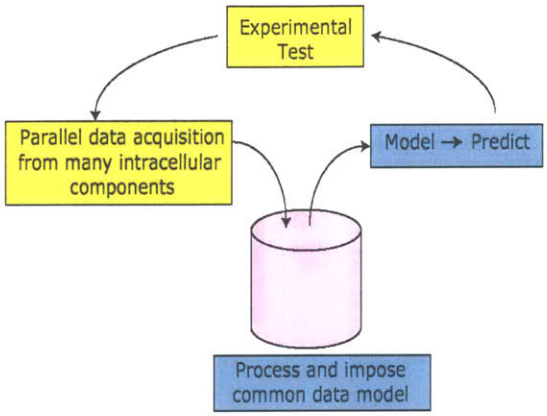


Figure 3 A network biology approach (Ref: Suzanne Gaudet, 2002)

This thesis focused on experimental databases. In following paragraphs, I will discuss existing experimental databases that perform limited functions for individual data collections. I will also discuss current attempts to create semantic structures for other biological data. We conclude this section with a description of current data, integration tools that have been applied in the biological area.

2.2 Existing or Ongoing Experimental Databases

The Stanford Microarray Database (SMD)[22] and the ArrayExpress[23] database are two good experimental databases in the specific biological area. SMD stores raw and normalized data from microarray experiments, as well as their corresponding image files; ArrayExpress is a public repository for microarray data, which is aimed at storing well annotated data in accordance with Microarray Gene Expression Data (MGED) [24] recommendations. Although the National Center for Biotechnology Information (NCBI) developed the Gene Expression Omnibus[25] (GEO) to serve as a public repository for a wide range of high-throughput experimental data, GEO is not an experimental database. It doesn't store any raw data, such as images. So here the detail information about GEO will not be introduced.

A. Stanford Microarray Database

The Stanford Microarray Database (SMD) stores raw and normalized data from microarray experiments, and provides web interfaces for researchers to retrieve, analyze and visualize their data. The initial purposes for SMD are to serve as a storage site for microarray data from ongoing research at Stanford University, and to facilitate the public dissemination of that data once published, or released by the researcher. SMD provides the links to the relevant biology, including Saccharomyces Genome Database (SGD)[26,27], Saccharomyces cerevisiae (YPD) and Caenorhabditis elegans (WormPD)[28], Unigene[29], dbEST[30] and SWISS-PORT[31].

Since SMD uses Oracle 8 as the database management system, it is implemented as a relational database. Scripts are implemented using the Perl language, in conjunction with the database independent interface (DBI)[32] module and the GD[33] modules (the

interface to the GD graphics library[34]). Some of intensive tasks are implemented in the C programming language.

SMD's loading program allows Stanford users to load their experiments into the database via a web form, either individually or via a batch procedure. The database accepts data produced by both GenePix[35] and Scanalyze[36]. Data normalization is performed during loading and both the normalized and original data are stored. SMD does not however act as a public repository for data, but instead makes all of its source code available to enable other institutions to set up their own microarray databases using SMD's model.

One future goal of SMD is the support of a data exchange format that will allow researchers to easily exchange microarray data. SMD intends to support and help define the format being discussed by the Array XML working group (http://beamish.lbl.gov).

Note that the SMD effort is limited to arrays and has no counterparts for other related experimental data.

B. ArrayExpress Database

ArrayExpress is a public database of gene expression experiments implemented by European Bioinformatics Institute (EBI). These experiments consist of multiple parts (or objects), in order to simplify querying they have provided a query interface which allows users to retrieve the most useful information relating to each experiment, the array(s) and the protocols(s) used. Experiment, array and protocol all are able to be queried at present. Each of these query objects has associated query fields.

Array, Experiment and Protocol all have assigned accession numbers in the format A-XXXX-1, E-XXXX-1, P-XXXX-1 (where X is an alphabetic character and the A/E/P prefix denotes accession number type) and can be retrieved directly by querying on their respective accession numbers or on text fields related to them. Experiments reference both protocols and array designs, though these also can be retrieved by querying their respective accession numbers. The data for protocols/array design is only stored once within the database, and all experiments that use the protocol or array design refer to the assigned identifier. This means that data is stored in a non redundant way and allows queries on e.g. all experiments that use a particular array.

MAGE-OM (Microarray Gene Expression - Object Model) was used as a source for automatic generation of a relational schema. ArrayExpress runs under Oracle, however no Oracle-specific features have been used and they believe it is easy to port this schema to other database platforms.

They claim that the mapping from object model to tables is quite simple, where there is a separate table for each class as well as for many-many relations; it is easy to load data into such schema from MAGE-ML (Microarray Gene Expression - Markup Language that derived from MAGE-OM); however for purposes of efficient queries some data de-normalization is thought to be needed. Like the SMD effort, ArrayExpress is limited to arrays and has not counterparts for other related experimental data.

C. USC brain project Time Series Database (TSDB)

Another kind of biological experimental databases we can find from the internet is neuroscience experimental database. One representative is the USC (University of Southern California) brain project Time Series Database (TSDB). TSDB[37] is a database for the storage of neuroscience data that can be extended to meet a specific lab's requirements. The TSDB group represents a union between two research laboratories, Dr. Thompson's Lab and Dr. Berger's Lab, which share a common goal in archiving neurophysiological data in a unified manner.

NeuroCore is the relational database framework for TSDB for the storage of neuroscientific data that can be extended to meet a specific lab's requirements. NeuroCore consists of a number of core tables; these core tables can be extended differently to meet the needs of a specific laboratory. Because the extended databases are based upon the same core database schema, they can share information. NeuroCore also contains a number of tools that allow users to work with the database framework; these tools include the Schema Browser, JADE (Java Applet for Data Entry)[38], the Database Browser, and the Protocol Viewer. They claim that the DB Browser is a web-enabled tool that allows a neuroscience researcher to view and query experiment data directly from the database; it is an interface designed for the NeuroCore database schema and offers the same flexibility and extendibility as the NeuroCore design. By using the NeuroCore schema as a basis, the new laboratory would share common elements with other laboratories so that they will be compatible. I doubt the project is still ongoing.

Although they list the links of DB Browser on their website, the links do not work. It seems to me that the project is only for internal usage. We can not tell whether the queries and schema are robust or not.

D. Neuroscience Federated Databases

Neuroscience Federated Databases[1, 39] is a knowledge-based integration of neuroscience data sources, developed jointly by the San Diego Supercomputer Center (SDSC) and the National Center for Microscopy and Imaging Research (NCMIR), UCSD. This team has developed a wrapper-mediator architecture which extends the conventional data- and view-oriented information mediation approach by incorporating additional knowledge-modules that bridge the gap between the heterogeneous data sources. The semantic integration of the disparate local data sources employs F-logic[26] as a data and knowledge representation and reasoning formalism. They show that the rich object-oriented modeling features of F-logic together with its declarative rule language and the uniform treatment of data and metadata (schema information) make it an ideal candidate for complex integration tasks. They substantiate this claim by elaborating on their integration architecture and illustrating the approach using real world examples from the neuroscience domain. The complete integration framework is currently under development. They claim that a first prototype establishing the viability of the approach is operational. But it is still a demo version and the function is limited.

E. Other efforts in biological database research

Today there are many biological knowledge-based databases or literature collections being developed, such as, GenBank, Apoptosis DB, PubMeb, PathDB, NCBI

Entrez, ExPASy Prosite, ExPASy Enzyme, WIT, KEGG, and U. of Minn. Biocatalysis/Biodegradation Database and so on. There were several trials to build up heterogeneous biology database systems[40] but there seems to be no successful story. K2/Kleisli and GUS[41] are two systems for the biomedical community to access to heterogeneous data sources, which are developed at the University of Pennsylvania. These systems focused on the knowledge-based biological databases, such as GenBank, SWISS-PORT, OMIM, and so on. In the paper that they published, they mentioned that the most difficult parts of data and software integration are semantic integration - a semantic layer must be available in data integration (page 526 in the K2/Kleisli paper). But they haven't stepped into this yet. Kenneth Baclawski and his colleague developed the database techniques for biological materials & methods[42] in 1995. But that notebook software was focus on research literature only. We won't try to make such a federated database that collects all biological knowledge together. We will focus only on the biological experimental records themselves to find a way to describe, catalog and store the data and facilitate the dissemination of that data. We believe there are some common entities can describe biological experimental records.

Although the above databases perform limited functions for individual data collections, how to design a database/system, which can serve for a variety of biological experimental methods, not limited in microarray only, is still a challenging topic. This thesis will discuss a unique way to design this kind of database and software architecture - the challenge is how to generalize the common semantic entities and schema suitable with various experimental methods. Current initiatives to standardize various aspects of information representation and sometimes the experiments themselves (i.e. the information gathering process) are of paramount importance[43]. To introduce such standards, first we have to understand clearly what the essential information is that needs to be captured for a particular technology or application. Next we have to develop a formal language or an object model able to represent this information. In the following paragraphs I will discuss current attempts to create semantic structures for the biological data.

2.3 Semantic Systems for Biology

There are several ongoing initiatives particularly devoted to various semantic standardization issues in life sciences, such as MicroArray and Gene Expression Markup Language (MAGE-ML), PEDRo (the Proteomics Experiment Data Repository), and Open Microscopy Environment (OME). They are good representatives of semantic standardization in microarray, proteomics, and microscopy, respectively. The following is just a simple introduction about them, and more details will be discussed in the Chapter 5.

A. MicroArray and Gene Expression – MAGE

The MAGE group aims to provide a standard for the representation of microarray expression data that would facilitate the exchange of microarray information between

different data systems. Currently, this is done through the OMG (Object Management Group) by the establishment of a data exchange model (MAGE-OM) and data exchange format (MAGE-ML) for microarray expression experiments. MAGE-OM has been modeled using the Unified Modeling Language (UML) and MAGE-ML has been implemented using XML (eXtensible Markup Language). MAGEstk (or MAGE Software Toolkit) is a collection of packages that act as converters between MAGE-OM and MAGE-ML under various programming platforms. MAGE-OM is MIAME-compliant. MAGE-ML can describe microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data and data analysis results. (Ref: http://www.mged.org/Workgroups/MAGE/mage.html)

B. The Proteomics Experiment Data Repository – PEDRo

PEDRo is developed by Chris Taylor, et al, in the U.K. PEDRo is intended to capture all the relevant information from any proteomics experiment, such as details of the experimenter, the sample source, the methods and equipment employed, and (of course) any results and analyses (both results and metadata). The data model described in PEDRo is offered as the starting point for a definition of the minimum set of information required about a proteomics experiment. Two general criteria are used in the definition: the repository should contain sufficient information to allow users to recreate any of the experiments whose results are stored within it; the information stored should be organized in a manner reflecting the structure of the experimental procedures that generated it. As the PEDRo model is independent of any particular implementation, several implementation structures are derived from it for use in different settings: for examples, a PEDRo-compliant, Java-based data entry tool, and an XML Schema representation of the PEDRo model (PEML, the proteomics experiment markup language) for use as a data interchange format. (Ref: http://pedro.man.ac.uk/home.shtml)

Although the attributes for the fields in each class (including data types and which fields are compulsory) of the PEDRo data model are described in the corresponding relational database definition, the database is not actually in use so far.

C. Open Microscopy Environment – OME

OME is a joint academic-industrial project, based in the United States at MIT, Cambridge Mass, at the NIH in Baltimore, MD and in Europe at the Wellcome Trust Biocentre, University of Dundee, Scotland. Its design goal is as an object oriented, database driven software system for quantitative analysis of biological images; modular programming environment for image analysis; universally readable XML file format for storage and transport of images; open source software under LGPL license (GNU Lesser General Public License). OME consists of four basic components: (i) a XML-encoded file standard that they will promote as a universal method to exchange biological image data in a web-readable format; (ii) a relational database system that is the core of the modular programming environment; (iii) a set of analytic routines that transform and manipulate images to extract quantitative information; (iv) a set of reference implementations in which cutting-edge biological problems are solved through the application of OME software. Since the goals of OME are broad, currently they are

working on the first production implementation, Version 2 and the installable production version of OME is no yet finished. (Ref: http://www.openmicroscopy.org/)

Like MAGE-ML, I list here several other biological markup languages being developed, such as,

- Cytometry Markup Language (CytometryML),
- Bioinformatic Sequence Markup Language (BSML)[44],
- CellML[45] (is an XML-based markup language being developed by Physiome Sciences Inc. The purpose of CellML is to store and exchange computer-based biological models.)
- Systems Biology Markup Language (SBML)[46],
- Molecular Dynamics [Markup] Language (MoDL)[47],
- Protein Extensible Markup Language (PROXIML)[48],
- BIOpolymer Markup Language (BIOML)[49],
- Gene Expression Markup Language (GEML)[50],
- GeneX Gene Expression Markup Language (GeneXML)[51],
- Microarray Markup Language (MAML)[52],
- XML for Multiple Sequence Alignments (MSAML)[53],
- Genome Annotation Markup Elements (GAME)[54].

The purpose of these markup languages is to exchange data or store information as XML-based format. They don't create databases in and of themselves. Some of the mentioned semantic standardization aspects can be addressed by bioinformatics databases, but they still only address some of the standardization problems. We can convert some of the above object models to database schemas and then translate the messages written by the above languages into the databases. The databases can be federated using the technique developed by Professor Dewey's group[84-87]. Additionally, the researches from other fields worth us to concern as well, like, Chemical Markup Language[55], StarDOM - Transforming Scientific Data into XML[56], Mathematical Markup Language (MathML)[57], OpenMath[58], Resource Description Framework (RDF)[59], and so on. For example, we can use RDF to define metadata and describe the characteristic of each individual system; we may consider semantic experimental record exchange using RDF or other XMLs with SOAP[60] (Simple Object Access Protocol), which will be introduced in the following paragraphs.

2.4 Data Integration Methods

There are several studies that focus on biological data integration. More and more institutes in many countries are interested in this topic, such as the United States, Canada, and Germany. Three typical examples are the Strawman project[61] for October I3C Hackathon in Hinxton - using ontologies to describe biological services and data, I3C's Bio-Google[62], and IBM's DiscoveryLink[63]. I will introduce them in detail in the following paragraphs.

A. Using Ontologies to Describe Biological Services and Data

The Strawman project for I3C Hackathon in Hinxton (October, 2003) - using ontologies to describe biological services and data, is based on myGrid[64]. They want to have a feature-rich, highly interactive, collaborative, plug & play bioinformatics workbench to develop and reuse workflows that execute on their distributed systems. That workbench would be linked with the runtime system to allow the user to monitor and interact with those workflows. Semantic ontologies describing the services and data sources used as components in those workflows offer tantalizing possibilities in tackling this goal. The goal of the hackathon is to advance the use of the Life Science Identifiers[65] (LSIDs) to name, describe, and facilitate discovery of life science resources. In preparing the hackathon, Mike Niemi from IBM proposed the possible system diagram as shown in the below.

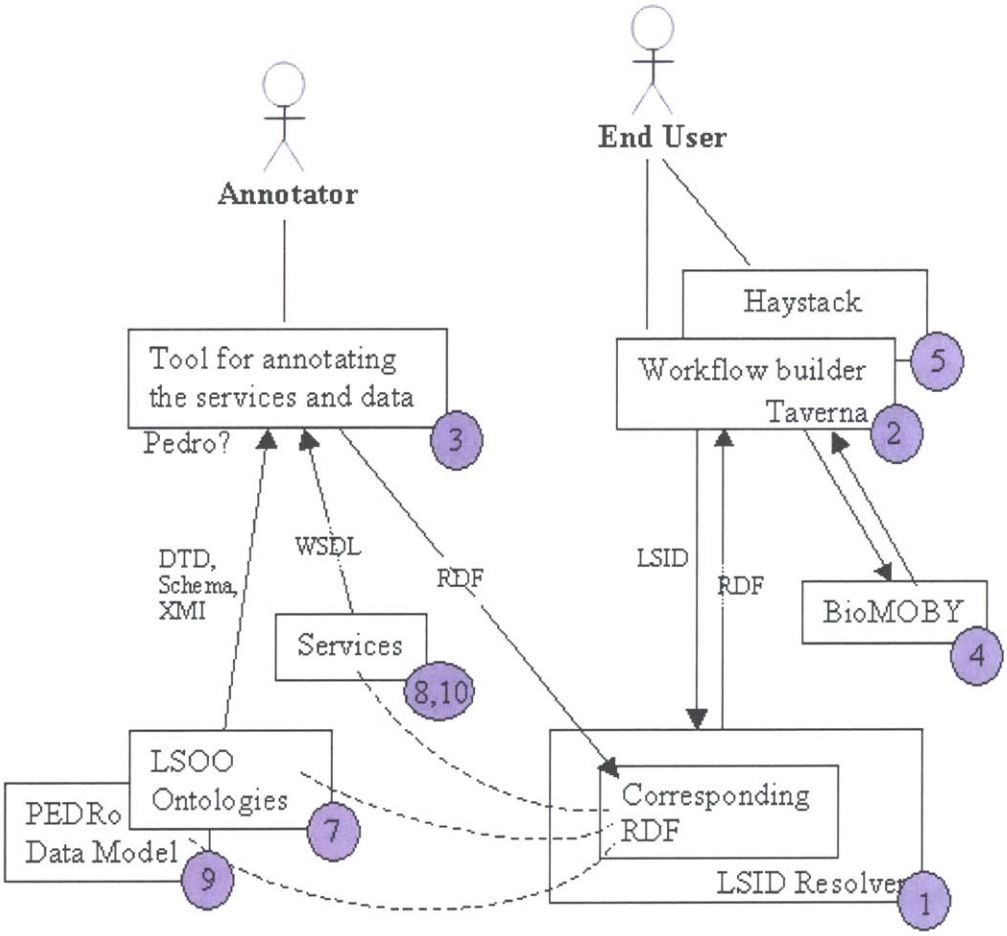


Figure 4 The possible system diagram proposed by Mike Niemi from IBM in the October I3C Hackathon in Hinxton, 2003

The numbers in the diagram represent participants from different projects, listed in the following table:

Table 1 Participants in the October I3C Hackathon in Hinxton

	Project
1	LSID Resolver[66]
2	Taverna[67]
3	Pedro tool[68]
4	BioMOBY[69]
5	Haystack[70]
6	Gene Ontology (GO)[71]
7	LSOO[72]
9	PEDRo data model
10	Soaplab[73]
11	Jena[74]
12	myGrid
13	I3C Registry Group[75]

The main idea of the diagram is to foster interoperability through the unambiguous naming of life sciences resources (using LSIDs), description of them (using metadata such as RDF), and their discovery (using query and search techniques involving registries, "google-like" services, and Web crawlers). During the hackathon, they concluded the following ideas:

- better integration of LSIDs with the BioMOBY registry
- using metadata (in particular RDF) to describe resources
- using semantics to assist type checking and conversion within workflows
- using LSIDs in workflows to annotate the results of a workflow execution
- enhancing Haystack as a desktop client for these environments
  But so far they haven't come out a workable demonstration yet.

B. I3C's Bio-Google Demonstration

The Technical Architecture Working Group of the I3C has prepared a demonstration (Fig. 5) showing one way life sciences interoperability standards can be used. The demonstration is designed to validate the submission of a new I3C standard called LSID (Life Science Identifier), and to show how BSML (Bioinformatic Sequence Markup Language) can be used to integrate applications from multiple vendors and open source projects.

The demonstration will show a global Bio-Google-like search capability that lets researchers search a federated group of multi-species genomics databases to find orthologs—genes that share a common evolutionary history—and then visualize the results. They believe the orthologs are significant to researchers because they tend to have similar functions, even across species. Executing this type of search is not feasible with current technology because interoperability standards do not exist for querying multiple distributed databases and visualizing results. The demonstration will illustrate the use of a new I3C standard submission, LSID, and the integration of a number of popular vendor products using BSML, a common XML-based format. They claim that

LSID and BSML help provide the interoperability infrastructure required to assemble diverse applications and multiple data sources to solve important scientific problems.

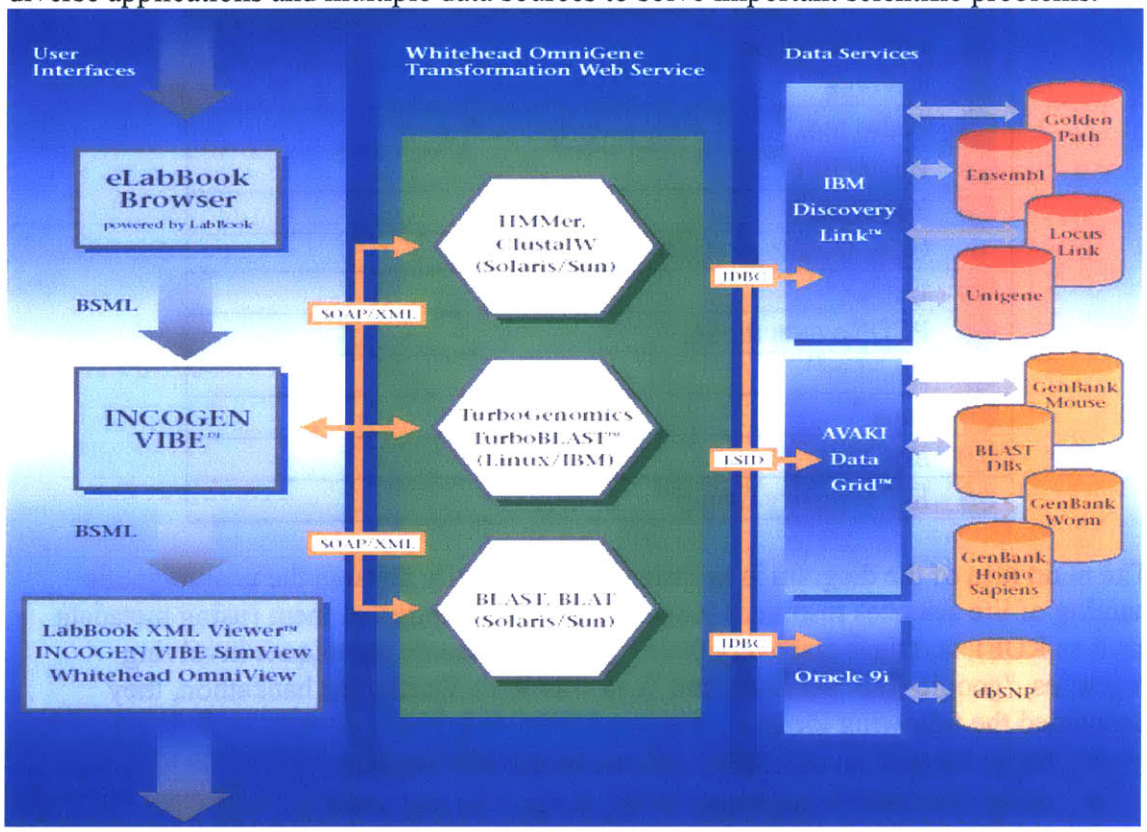


Figure 5 I3C's Bio-Google demonstration

However, in the demonstration, there are still many details they didn't consider carefully, such as the links to IBM DiscoveryLink and to Oracle 9i, etc. The demonstration is a frame structure only at present; they haven't implemented it yet.

C. IBM's DiscoveryLink

To meet the challenge of integrating and analyzing large quantities of diverse scientific data from a variety of life sciences domains, IBM claims that it has developed a versatile solution — DiscoveryLink™ — that can help dramatically increase R&D productivity with single-query access to existing databases, applications, and search engines. DiscoveryLink technology enables and enhances data integration by providing the critical interface between front-end applications and data sources. It works transparently with a multitude of databases, applications and client tools. DiscoveryLink is not a front-end application and therefore does not have its own specific front-end "look." The administration of DiscoveryLink is performed using a graphical application named Control Center™ (Fig. 6). The Web-based demonstration was developed to demonstrate one manner in which DiscoveryLink can be used in a life sciences environment. The demonstration uses Java™ technology. (Ref: http://www-1.ibm.com/industries/lifesciences/doc/content/solution/939513121.html)

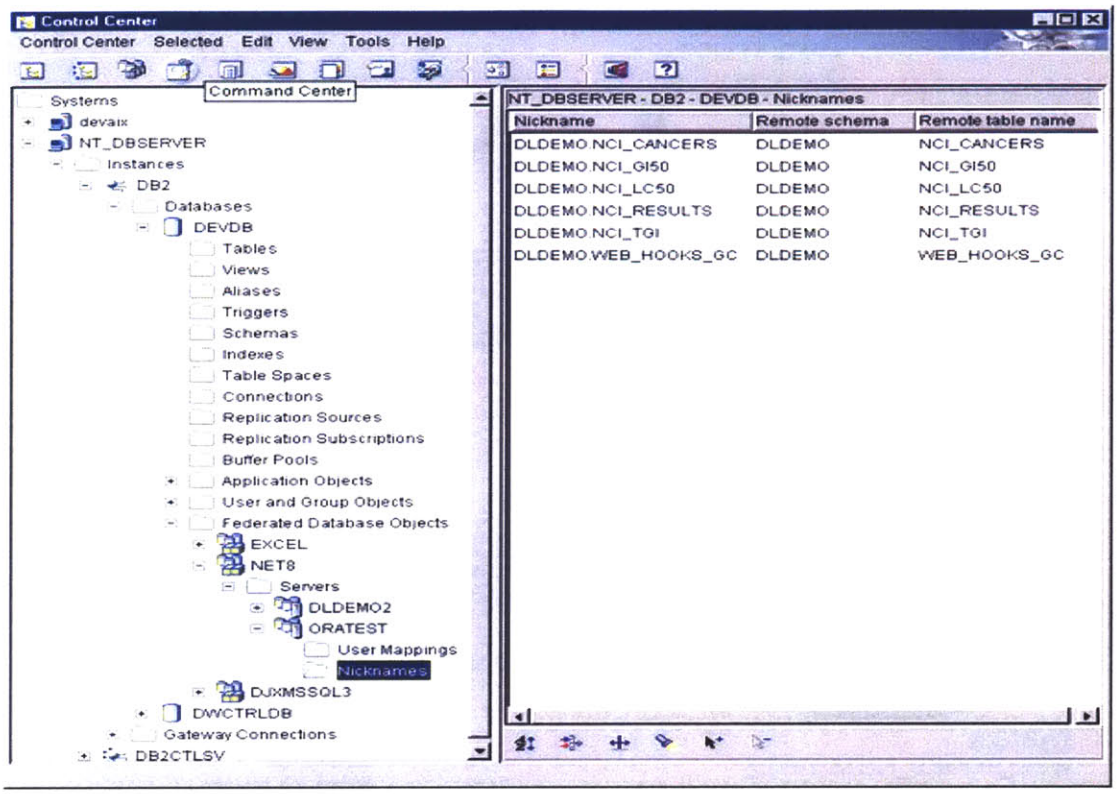


Figure 6 Highlight of DiscoveryLink registered "NICKNAMES"

The DiscoveryLink solution includes the combined resources of DiscoveryLink software and IBM Life Sciences Services. Built on IBM DB2 Universal Database™ technology, DiscoveryLink includes; DB2® software, DB2 Relational Connect™ middleware tailored specifically to life sciences research, and DB2 Life Sciences Data Connect™. IBM DB2 Life Sciences Data Connect™ enables a DB2 federated system to integrate a wide variety of genetic, chemical, biological, and other research data from heterogeneous distributed sources. When used in conjunction with DB2 Relational Connect, SQL statements are used to query, retrieve, and join data located in life sciences data sources as well as relational databases from IBM, Oracle, Sybase, and Microsoft. The current set of data supported by DB2 Life Sciences Data Connect is from table-structured files. A federated server communicates with a data source by using a wrapper. The wrapper allows the server to perform operations such as connecting to a data source and retrieving data from it. This enables data in a table-structured file to be joined with relational data or data in other table-structured files. The information in these data sources can be accessed as if it were one large database (Fig. 7 Query architecture).

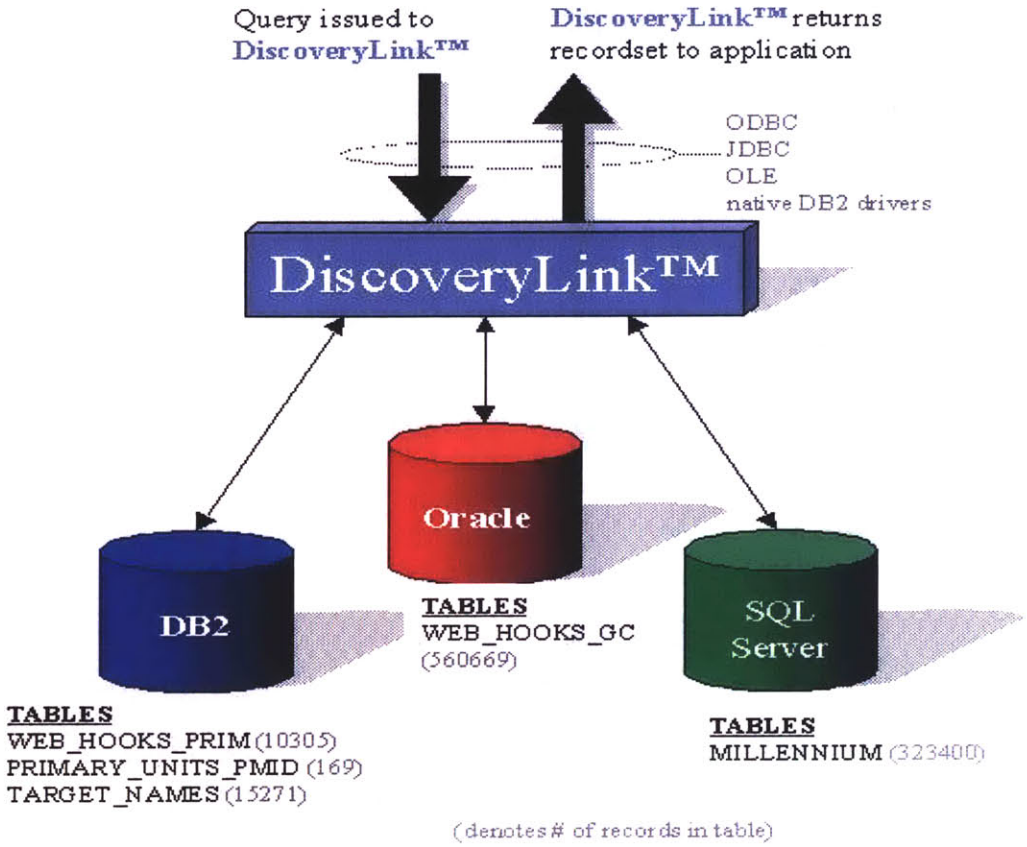


Figure 7 Query architecture of DiscoveryLink

DiscoveryLink is in the demo stage and Bio-Google is even worse.

Whatever the software architecture is, a major issue they face is the unique naming or identifier for the heterogeneous biological data. I3C is developing the Life Sciences Identifier (LSID) [76], which provides for scalable, secure, and migration-transparent naming of biologically significant data. But the idea of LSID is still a draft for review. Robert Kincaid, the Life Science Technologies Laboratory of the Agilent Technologies developed a DNS-inspired biomolecule naming service – BNS[77]. BNS provides standards-based infrastructure and interfaces that easily resolve existing molecular names and identifiers. BNS is based on the Lightware Directory Access Protocol (LDAP). This protocol limits the use of BNS.

2.5 Previous Research of Professor Dewey's Group

Professor Dewey's group did a lot of research on the information architecture for physiological models, clients and databases in these recent years, such as, DICOM database[78,79] and server[80,81], computational server and model database[82,83], and federation platform for gene databases[84,85], etc. Professor Dewey and his colleagues have developed the Class Mapper concept to federate databases and reusable interfaces/classes providing the connections between information entities (Fig. 8). Ben Fu has developed the first

demonstration of the federation platform using the Class Mapper concept (See Fig. 9 and 10, Ben Fu's thesis page 21 and 31). All of these technologies are the solid foundation of the thesis research.

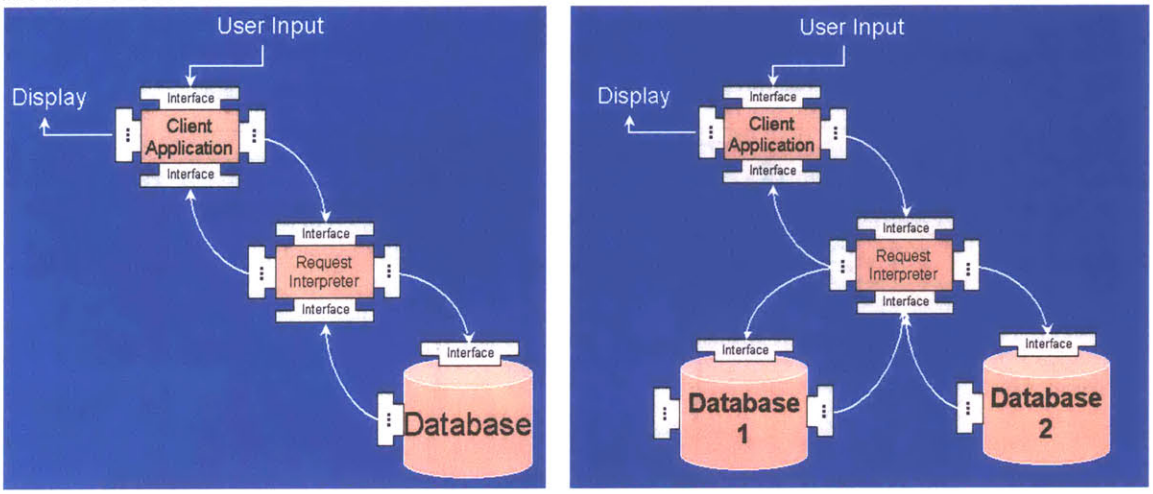


Figure 8 A: Conceptual connections between information entities. (Left)
B: Accessing multiple information entities. (Right) (Ref: Professor Dewey presentation)

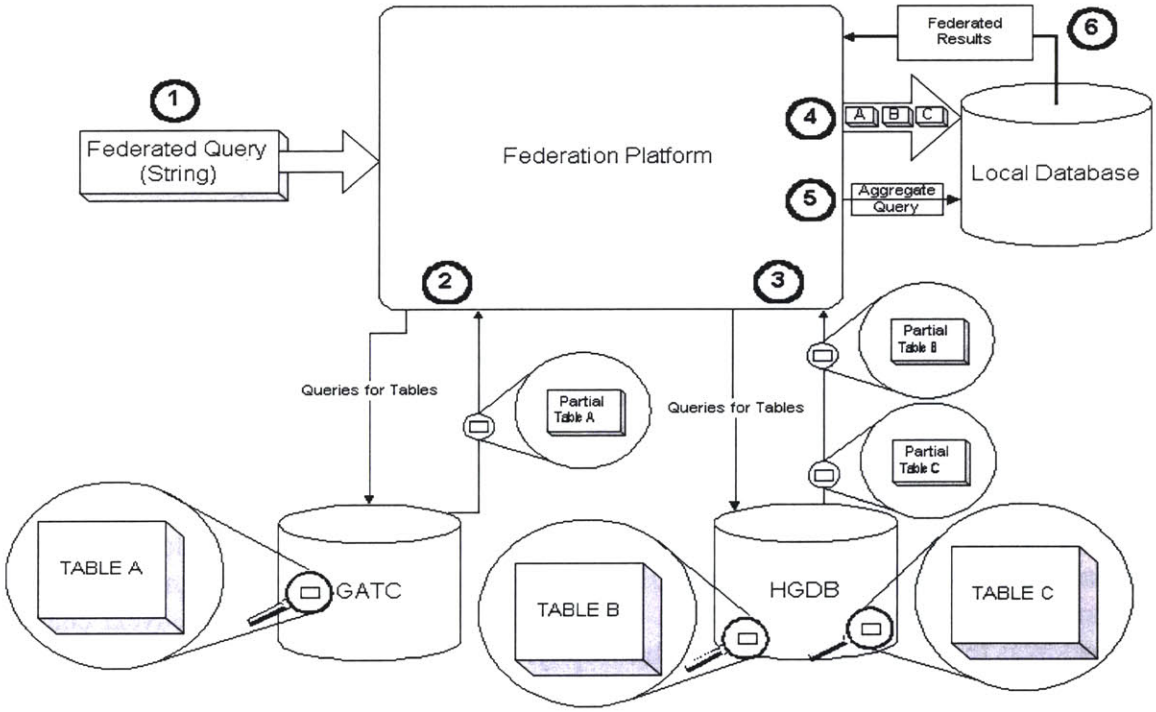


Figure 9 The transactions during a federated query. (Ben Fu's thesis, page 21)

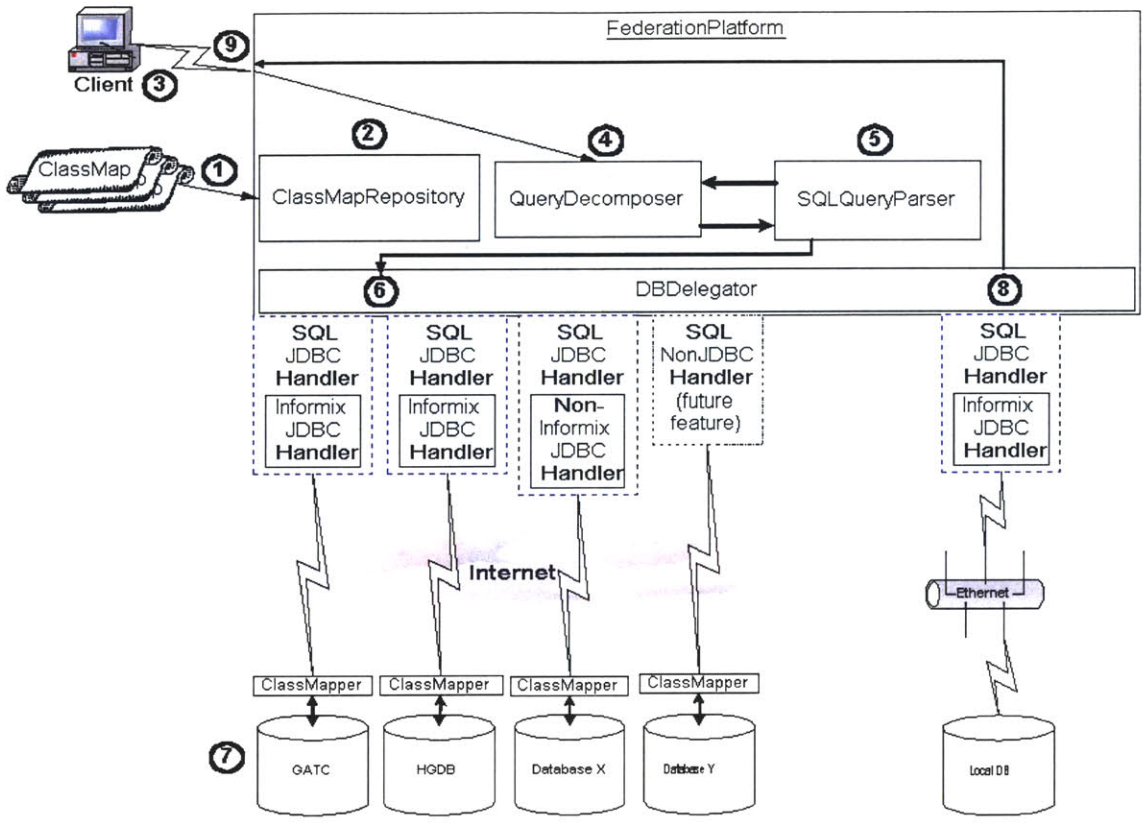


Figure 10 The federation platform architecture. (Ben Fu's thesis, page 31)

This chapter has introduced several existing experimental databases, discussed current attempts to create semantic structures for other biological data, and described current data, integration tools that have been applied in the biological area. Although the mentioned databases perform limited functions for individual data collections, how to design a database/system, which can serve for a variety of biological experimental methods, not limited in microarray only, is still a challenging topic. The markup languages are to exchange data or store information as XML-based format. They don't create databases in and of themselves. Some of the mentioned semantic standardization aspects can be addressed by bioinformatics databases, but they still only address some of the standardization problems. This thesis will discuss a unique way to design the common semantic entities and schema suitable with various experimental methods. Next chapter will address the design proposal and philosophy.

Chapter 3. Design Proposal and Philosophy

This thesis needs to design such an integrated software platform (named as ExperiBase) that should be general enough to support data/metadata generated from both conventional biological methods as well as new experimental methods. A key feature is its ability to integrate data from different quantitative experimental strategies. It needs to provide: comprehensive database features for searching and classifying; web-based client services; data import and export capabilities to accommodate other data repositories; and direct support for metadata produced by analysis programs. The software should have good extensibility. Because ExperiBase needs to support many different experimental methods, and the semantic contexts that support them, the creation of this software platform should provide a major contribution to normalizing semantic support for these different methods. The research needs to design a set of generic entities to describe the records of biological experimental data, like DICOM medical image archive database does, which is designed by Professor Dewey's group[86]. The user interfaces linking to data input and output should be more convenient than the current data communication methods. The database may be an open source that allows each biologist enters his/her data in and the data are released to the public at the researcher's discretion or upon publication. The databases may be federated databases, like many ongoing biological database designs; for examples, IBM DiscoveryLink, I3C Bio-Google, and the idea of Neuroscience Federated Databases. The ExperiBase system may provide links with many other public biological knowledge-based databases, such as, apoptosis-DB[87], GenBank (Entrez system)[88], PDB[89], and so on. ExperiBase should enable a researcher to filter experimental data and retrieve only that with which he or she wishes to work, and then perform analyses on that data. It should store any types of raw data, processed data and metadata from a variety of biological experiments, and provide web interfaces for researchers to retrieve, analyze and visualize their data. It should provide a friendly and flexible connection for the data communication between experimenters and modeling researchers.

The immediate testing of ExperiBase is to serve as a storage site and smart data communication tools for the PNNL project and the DARPA BioInfoMicro project at MIT and to facilitate the dissemination of that data.

The biological experimental methods supported in the first generation of ExperiBase should include at least:

–gel electrophoresis,
–microarrays,
–microscope images,
–mass spectrometry.

The detail proposal of the thesis included: common entities definition (data normalization and UML models); database schema; federated databases and data integration; web-enabled interfaces and adapters; system implementation as below. This covers data modeling and databases, Web access and development, middleware, data communications and integration technologies.

- Define a set of common semantic elements/entities to describe the records of various biological experimental methods.

As we discussed in the Chapter 2, either we may define the semantic normalization by generalizing the entities used in the biological markup languages, such as CellML, SBML, MoDL, GAME, PROXIML, BSML, BIOML, GEML, GeneXML, MAML, MAGE-ML, MSAML, and so on; or, by collecting the data from the experimenters or literatures and studying their current methods and tables, we can generalize a set of common semantic elements/entities to describe the data from a variety of biological experimental methods. The concept of entities should be consistent between information object definitions. It should be object-oriented, inheritable and nested. We may consider defining the metadata by using Resource Description Framework (RDF) as well.

- Database design (schema and implementation).

The schema should reflect objects. It should be extensible and reusable. It should be suitable for directed and expanded queries. New data type definitions should support complex data types (such as, allows powerful image content queries), the nested hierarchy and keep semantic relationships.

Designing a particular database schema for each biological experimental method and then federating them together using Class Mapper concept is one consideration of the system design. Or, an alternative is designing a single generic database schema to serve for a variety of biological experimental methods.

We may use XML-DBMS Java packages[90],[91] to convert the biological markup language schemas to database schemas and therefore create various biological databases. And then we can translate the XML messages into the databases. We can use federation techniques to combine those databases together. Or, an alternative is translating the biological markup language schemas to a generic XML schema; and therefore creates a generic biological database only. This solution would be more convenient to support a variety of biological markup languages.

The database schema should be able to dynamically extend. As far as the way of storing raw data and processed data, we may store the data files as objects directly; or, we may convert the data files to database tables as long as possible. The second method will be more convenient for the modeling researchers to filter the data than the first one does.

- Web-enabled user interface implementation.

The information architecture should be portable and platform independent. We may consider using SOAP/XML, IIOP/CORBA, JDBC, Java Server Pages' technologies, and RDF. ExperiBase should be accessed over the Internet using a web browser, which allows for significant flexibility and remote access without the need for special software installation on client computers. In order to make multi platforms (Windows, UNIX, Linux, and MacOS) access the web pages without difficulty, a solution could be using Java Server Pages' technologies to produce the client web pages dynamically. SOAP/XML is an alternative protocol which we can choose for the data communication

between clients and servers. ExperiBase may support a variety of biological markup language messages. We may consider semantic experimental record exchange using RDF or other XMLs with SOAP so that our databases can support those biological markup language messages. Some friendly tools should be provided to facilitate data communication, such as, a class to convert Excel spreadsheets (or other types of spreadsheets) to database schema, a class to parse data out from FACS files, a class to convert ImageQuant[92] image to TIF format, and so on. Some adapters should be provided as well, such as an adapter for Matlab users to allow their programs to access data directly. We may consider providing web services for ExperiBase so that other developers or systems can use our services by importing the interfaces to their software directly.

- Federated databases and data integration

The paramount importance of ExperiBase is the data integration of a variety of experimental methods as well as the connection of experimental data with many public resources, such as SWISS-PROT[93], Unigene[94], GenBank, PubMed[95], and so on. The federating and linking process should be done automatically. We may provide the connection of one experimental data with the other biological data that pertains to the cells, proteins, chemicals, genes, pathway, etc.

- System implementation (performance and scalability)

We should choose a good solution to ensure data safe, consistency and recovery. Access privileges should vary with the different components of the system. That means, for example, the un-released data should prevent being stolen and so on. The performance and scalability of the system/database should be one important consideration of the system design. Once the databases open to the public, data may expand rapidly. We hope the performance of the system will be linear.

As a summary, ExperiBase should present a new informatics platform to handle a large subsection of the experimental protocols that currently exist. A consistent data definition strategy should be outlined that can handle gel electrophoresis, microarrays, fluorescence activated cell sorting, mass spectrometry, and microscopy within a single coherent set of information object definitions. ExperiBase should provide: comprehensive database features for searching and classifying; web-based client services; data import and export capabilities to accommodate other data repositories; and direct support for metadata produced by analysis programs. The creation of this software platform should provide a major contribution to normalizing semantic support for those different methods. Using JDBC, Java Servlets and Java Server Pages, SOAP, XML, and IIOP/CORBA's technologies, the information architecture should be portable and platform independent.

We should obey the following design philosophy:

1. The design of biological experimental information entities is based on ontology standards; we will adapt any which has existed.
2. We should keep the conceptual consistency between different experimental methods.

3. The ontological "Capital" (i.e. key concepts) should be reused between different experimental methods.
4. The information architecture should be portable and the platform is independent of OS.

Chapter 4. Database Technology Review

The term 'database' has many interpretations; one definition is a "collection of persistent data" which is specially organized for rapid search and retrieval. As computer processing and storage power has increased over the years, so has the need to store larger quantities and widely varying types of data. These driving forces have resulted in an ongoing evolution in the types of databases available, which can be divided into four main groups. These four main groups usually mean the hierarchical data model, the relational data model, the object-oriented data model, and the object-relational data model. In addition, the earliest databases were flat files containing a set of records, each of which consisted of one or more fields. These fields are the basic units of data and each field corresponded to a particular attribute of the data stored in the database.

The research involved in the development of a database data model is a massive undertaking that typically requires years and the cooperation of many individuals. This research project does not aim to develop a new database data model. Instead, this project uses work others have done and applies it to the biological experimental domain. This chapter gives an overview of existing data models and then presents a discussion why object-relational data model is the best candidate to represent the biological experimental information. The reviews are basically summarized from Ngon D. Dao's thesis[96], William Chuang's thesis[97], Patrick J. McCormick's thesis[98], and "Database Models"[99] written by David R. Frick & Co., CPA.

4.1 Flat File Database

In database's most simple form, a flat-file database is nothing more than a single, large table (e.g., a spreadsheet). A flat file contains only one record structure; there are no links between separate records. Access to data is done in a sequential manner; random access is not supported. Access times are slow because the entire file must be scanned to locate the desired data. Access times can be improved if the data is sorted but this introduces the potential for error (e.g., one or more records may be misfiled). Other problems with a flat-file database include 1) data redundancy; 2) data maintenance; and 3) data integrity. For example, an 'orders' file might require fields for the order number, date, customer name, customer address, quantity, part description, price, etc. In this example, each record must repeat the name and address of the customer (data redundancy). If the customer's address changed, it would have to be changed in multiple locations (data maintenance). If the customer name were spelled differently in different orders (e.g., Acme, Acme Inc, Acme Co.) then the data would be inconsistent (data integrity). Flat file data structures are only viable for small data processing requirements. (Ref: http://www.frick-cpa.com/ss7/Theory_Models.asp)

4.2 Hierarchical Data Model

Two of the earliest data models are the hierarchical data model and its expansion, the network database model. The hierarchical and network database models preceded the relational model; today very few commercial databases use either of these models. A

hierarchical database is a series of flat-files linked in structured 'tree' relationships. The network database model expands on the hierarchical model by providing multiple paths among segments (i.e., more than one parent/child relationship). One of the first widespread medical data systems, the Massachusetts General Hospital Utility Multiprogramming System (MUMPS), used the hierarchical data model to store information. IBM's IMS (Information Management System) database, often referred to by the name of its proprietary language, DL/I (Data Language I), is the only viable commercial hierarchical database still in use today, primarily on older mainframe computers (Ref: http://www.frick-cpa.com/ss7/Theory_Models.asp).

The concept for the hierarchical database model was originally based on a bill of materials (BOM). Data is represented as a series of parent/child relationships. For example, student records might be grouped under a record describing their departments of study. The corresponding data structure will contain student nodes groups under the department nodes, with the relevant fields associated with each type of node. This concept is fundamentally different from that used in relational model where data resides in a collection of tables without any hierarchy and that are physically independent of each other. In the hierarchical model, a database 'record' is a tree that consists of one or more groupings of fields called 'segments'. Segments make up the individual 'nodes' of the tree (e.g., the 'customers' record may consist of 'customer' and 'order' segments). The model requires that each child segment can be linked to only one parent and a child can only be reached through its parent, which limits the ways in which users can access the data. The requirement for a one-to-many relationship between parent and child can result in redundant data (e.g., 'orders' might be a child of 'customers' as well as a child of 'parts'). To get around the data redundancy problem, data is stored in one place and referenced by links or physical pointers in other places (e.g., the 'customers' record contains actual data in the 'orders' segment while the 'parts' record contains a pointer to the 'orders' data in 'customers'). To create a sales report, you have to access 'customers' to get to 'orders'. This is fundamentally different from the way a relational database operates; in a relational database there is no hierarchy among tables and any table can be accessed directly or potentially linked with any other table; there are no hard-coded, predefined paths among the data. (Note that while a primary key-foreign key combination in a relational database represents a logical relationship among data, it does not necessarily limit the possible physical access paths through the data). In the hierarchical model, the link established by the pointers is permanent and cannot be modified; in other words, the links are hard-coded into the data structure. The hard-coding makes the hierarchical model very inflexible; a design originally optimized to work with the data in one way may be totally inefficient in working with the data in other ways. In addition, the physical links make it very difficult to expand or modify the database; changes typically require substantial rewriting efforts.

4.3 Relational Data Model

The relational data model was first proposed in 1970 by Codd[100], but it is still the prevailing model in use today. It is the most mature data model to date. A relational database is one in which the data consists of a 'collection of tables related to each other through common values'. The two most prominent characteristics of a relational database

are 1) data stored in tables composed of rows and 2) relationships between tables. A table (also known as an entity or relation) is a collection of rows and columns. A row (a.k.a. a record or tuple) represents a collection of information about a separate item (e.g., a customer). A column (a.k.a. a field or attribute) represents the characteristics of an item (e.g., the customer's name or phone number); each column has one of a limited set of data types. A relationship is a logical link between two tables. The relational database model is based firmly in the mathematical theory of relational algebra and calculus. Specially, a relation is a set of ordered tuples defined over a set of no necessarily distinct domains. Each domain is itself a set. For example, given the sets (domains) D1 and D2, in Figure 11, there can be a relation R defined over these two domains such that each column in R contains one element from each of the two domains. The logical representation of this relation is a table containing two columns, one for each domain, to store attribute values. Each table row is a tuple representing an instantiation of the information entity represented by the table. In Figure 11, PK means primary key; FK means foreign key.

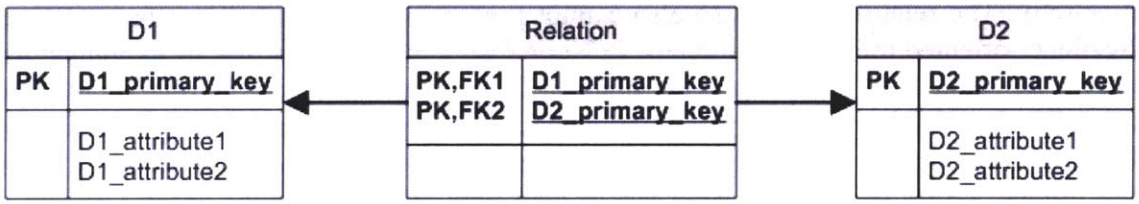


Figure 11 A relationship is a logical link between two tables.

A relational database management system (RDBMS) uses matching values in multiple tables to relate the information in one table with the information in the other table. The presentation of data as tables is a logical construct; it is independent of the way the data is physically stored on disk. In other words, in a relational database, the logical design is independent of the physical design. Queries against a RDBMS are based on logical relationships and processing those queries does not require pre-defined access paths among the data (i.e., pointers). The relational database provides flexibility that allows changes to the database structure to be easily accommodated. Because the data reside in tables, the structure of the database can be changed without having to change any applications that were based on that structure. For example, you add a new field for e-mail address in the customers table. If you are using a non-relational database, you probably have to modify the application that will access this information by including 'pointers' to the new data. With a relational database, the information is immediately accessible because it is automatically related to the other data by virtue of its position in the table. All that is required to access the new e-mail field is to add it to a SELECT list. The structural flexibility of a relational database allows combinations of data to be retrieved that were never anticipated at the time the database was initially designed. (Columns in different tables can be "joined" together to form powerful queries). In contrast, the database structure in older database models is "hard-coded" into the application; if you add new fields to a non-relational database, any application that access the database will have to be updated.

In practice, there is significant confusion as to what constitutes a relational database management system (RDBMS). Dr. E.F. Codd provided 12 rules that define the basic characteristics of a relational database but implementation of these rules varies from vendor to vendor. No RDBMS product is fully compliant with all 12 rules but some are

more so than others. When distinguishing DBMS products, there are typically three key areas on which to focus: 1) query formulation and data storage/access; 2) application integration; and 3) processing architecture. (Ref: http://www.frick-cpa.com/ss7/Theory_Models.asp).

While the relational data model is mature and reliable, it has shortcomings when the data are complex[101]. The most important shortcomings are the lack of a natural mechanism to represent nested information entities and a limited set of supported data types. These two limitations preclude the storage of unstructured data such as images and audio, and render the representation of complex data difficult at the very least. The relational model is not object-oriented. Pure relational databases don't support object inheritance and map between objects and tables. Without inheritance, an object-oriented information model stored in a relational database loses all of the semantic content represented in its inheritance hierarchy. Without this semantic content, lots of complicated joins will be needed when querying all sub-classes of a class inheritance hierarchy. The relational database also cannot present object-oriented interfaces defined by object-oriented information models. In Ngon Dao's thesis, (page 35), he explained this very well using DICOM information objects. Other weaknesses of the relational model are related to its limited set of supported data types. One direct consequence of this limitation is the poor support for nested or hierarchical data. Another problem of this is the limited ability of relation databases to deal with BLOBs. Binary Large OBjects or BLOBs are complex data types such as images, spreadsheets, documents, CAD, e-mail messages, and directory structures. There is no specific mechanism in the relational model to allow for the retrieval of parts of a BLOB. A relational database can store BLOBs but they are stored outside the database and referenced by pointers. The pointers allow the relational database to be searched for BLOBs, but the BLOB itself must be manipulated by conventional file I/O methods[102]. The object-oriented database and object-relational database have solved this problem.

4.4 Object-Oriented Data Model

The object-oriented database (OODB), also referred to as the 'post-relational' database model, addresses some of the limitations of the relational model. OODB evolved in the mid-to-late 1980's following the development of object-oriented programming languages such as C++. Object-orientation is the notion that entities in the application domain can be modeled as independent objects that are related to one another by means of classification. The external behavior and characteristics of objects are externalized whereas the internal implementation details of the object remain hidden. Object-orientation lets you capture the similarities and differences among objects in your application domain and group those objects together into related types. Objects of the same type behave in the same way because they share the same set of type-specific behaviors, reflecting the behavior of your objects in the application domain (Ref: IBM® DB2® Universal Database, SQL Reference, page 267).

At its most basic level, 'data' is a sequence of bits ('1s' and '0s') residing in some sort of storage structure. 'Traditional' databases are designed to support small bit streams representing values expressed as numeric or small character strings. In OODB, bit stream data is atomic; it cannot be broken down into small pieces. BLOBs are large and non-

atomic data; they have parts and subparts and are not easily represented in a relational database. Object-orient databases provide native support BLOBs. Under the general concept of an object-oriented database, everything is treated as an object that can be manipulated. Object-oriented databases use a data model that incorporates unique object identifiers, data encapsulation, and inheritance. Objects inherit characteristics of their class and have a set of behaviors (methods) and properties that can be manipulated. The hierarchical notion of classes and subclasses in the object-oriented database model replaces the relational concept of atomic data types. The object-oriented approach provides a natural way to represent the hierarchies that occur in complex data. For example, a Word document object consists of paragraph objects and has method to 'draw' itself.

A clear advantage of the OODM is the ability to directly represent the object-oriented biological experimental information model. Several other advantages of the OODM come from its support of object inheritance. The relationships, that individual objects may inherit properties from objects farther up in the hierarchy, can serve as the basis for defining rules for expanded query scopes. These relationships also facilitate the creation of classes by allowing new classes to inherit existing class attributes and/or methods. Another benefit of inheritance is the re-use of objects and methods. Besides inheritance, the OODM allows object attributes to have any data structure as their domains. A final advantage of the OODM is the notion of enforcing semantic integrity constraints. The integrity constraint for a class in an OODB can imply a PART-OF or CONSISTS-OF relationship between an object and the other objects that it references. These implied constraints can be made explicit to form the notion of composite objects, objects that are made of many component objects. This ability to strong type composite objects is very useful for the biological experimental information model because the model is riddles with PART-OF or CONSISTS-OF relationships; this fact is similar to the DICOM information model (Ref: Ngon Dao's thesis, page 38).

Unfortunately, there is no clear model, national or international standard, or framework for the object-oriented database like the one Codd provided for the relational database. The weaknesses of the object-oriented data model arise from this fact that the OODM, and hence commercial OODBs, are not as mature as relational databases. There are a limited number of commercial object-oriented database systems available; most commercial OODBs remain unstandardized, resulting in several different proprietary data models and interfaces. In addition, they also lag behind relational databases in transaction management, replication, and development environment.

Within recent years, several neutral object model and query languages have proposed, but they are far from widespread acceptance. Two representatives are ODMG standard[103] (Object Data Management Group) and Frame Logic (F-logic)[104]. The ODMG group completed its work on object data management standards in 2001 and was disbanded. F-logic is a declarative language with rich modeling capabilities and a powerful rule language. It accounts in a clean and declarative fashion for most of the structural aspects of object-oriented and frame-based languages. These features include object identity, complex objects, inheritance, polymorphic types, query methods, encapsulation, and others. In a sense, F-logic stands in the same relationship to the object-oriented paradigm as classical predicate calculus stands to relational programming. F-logic has a model-theoretic semantics and a sound and complete resolution-based proof

theory. A small number of fundamental concepts that come from object-oriented programming have direct representation in F-logic. A good example using the F-logic is the "Knowledge-Base Integration of Neuroscience Data Sources" [105], published by Amarnath, et al, from UCSD. In their design, they used an F-logic database (Florid[106]).

In a way, object-oriented concept represents a 'Back to the Future' approach in that it is very similar to the old hierarchical database design. Relational databases are not obsolete and may evolve by adding additional support for BLOBs and inheritance.

4.5 Object-Relational Data Model

Stonebraker[107],[108] and, independently, Kim[109],[110] developed the object-relational data model (ORDM) in the 1990's. The ORDM combines the best of both the relational model and the object-oriented model. It enables you to realize many of the benefits of object technology while building on the strengths of relational technology. In a relational system, data types are used to describe the data in columns of tables where the instances (or objects) of these data types are stored. Operations on these instances are supported by means of operators or functions that can be invoked anywhere that expressions are allowed. With the object extensions, you can incorporate object-oriented (OO) concepts and methodologies into your relational database.

The ORDM exploits all of the mature relational data model technologies and standards because the model is based on relations. Algorithms developed for concurrency control, transaction management, and query optimizations are all still valid within the ORDM framework because the notion of relations and tuples are preserved. Since it extends the relational data model, the object-oriented notions are well supported, such as encapsulation, inheritance, and object definitions having attributes and methods. In an ORDM, columns can now contain many different data types for complex data, instead of the limited set offered by the relational model. Columns can contain entire rows (row objects) or even user-defined types. User-defined types let you control the semantics of your objects. For example, in the biological domain, we might require a type named as "LSID" (Life Science Identifier, which is a string, could be a distinct type) or "gene", which is a collection of gene object attributes, so called as structure type. In an object-relational database (ORDB), tables can have constraints, storage options, triggers, indexes, and methods. The inheritance is not only supported by tables but also data types. The inheritance allows tables (subtables) / data types (subtypes) to inherit all or some of the columns, constraints, storage options, etc. of other tables (supertables) / data types (supertypes). In the modern object-relational databases, such as Oracle 9i[111] and DB2[112] v7 or above, many new object-relational features are implemented, like type inheritance, object view hierarchies, type evolution, user-defined aggregate functions, generic and transient datatypes, function-based indexes, multi-level collections, object array, nested tables, even directly Java object storage, and so on. For example, Oracle has three special generic SQL (Standard Query Language) datatypes (i.e., sys.anytype, sys.anydata, and sys.anydataset) that enable you to dynamically encapsulate and access type descriptions, data instances, and sets of data instances of any other SQL type, including object and collection types. The three generic SQL types are implemented as opaque types. In other words, the internal structure of these types is not known to the database: their data can be queried only by implementing functions for the purpose. For example, in the below

diagram (Figure 12), a table of biological experiments has a field for describing biological samples used in each experiment. Because the sample can be a cell, a piece of gene, or a type of chemical, the data type of this sample can be implemented as an opaque type. Or the sample can be (an object reference) referred to the supertype of cell, gene, and chemical (Figure 13). You can obtain the exact type of the object reference using dereferencing function. The object instance returned by dereferencing may be of the declared type of the reference or any of its subtypes. Certainly it is possible to express this schema in a strictly relational fashion, but the variant object reference design is more efficient and organic. A good research using the similar idea of variants and nested table is the database of experimental results on globin gene expression[113], published by Cathy Riemer, et al.. Continuing the example in Figure 12, the field of experimenter can be a subtable with its own subrows and subcolumns. Furthermore, the fields in the subtable can have their own sub-subtables or column objects; in the diagram, the address is a column object. In an ORDM, the tables are nested instead of flat.

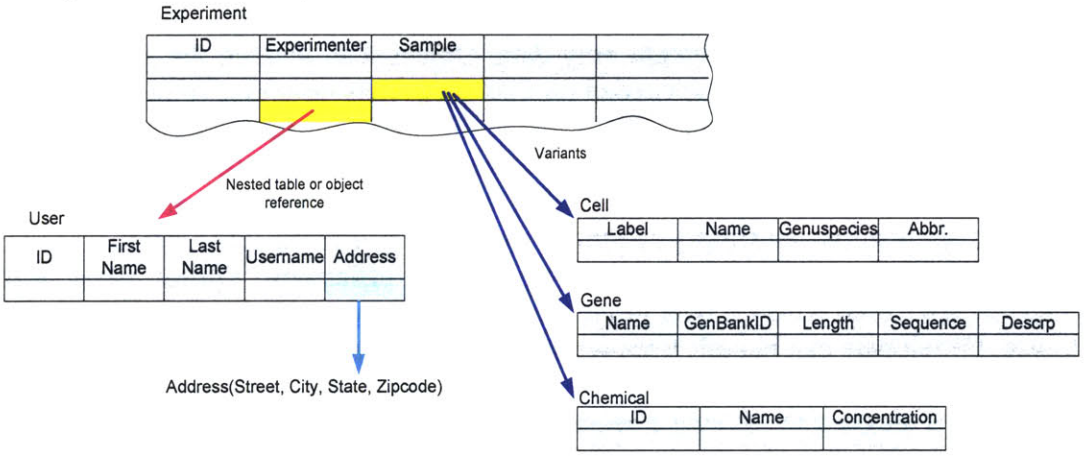


Figure 12 A simplified conceptual view of a portion of a schema for biological experiments, illustrating table nesting, object reference, and variants.

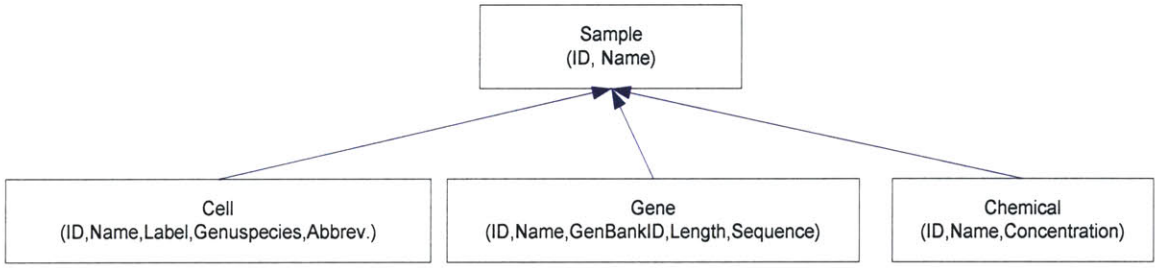


Figure 13 A data type "Sample" is defined as a supertype of cell, gene, and chemical

Since in DB2 or Oracle, type inheritance is supported within an ORDM schema, expanded query scope can be realized. Expanded query scope refers to the ability of querying all sub-objects of an object inheritance hierarchy using only one query. For example, a query against sample will query cell, gene, and chemical as well if the query scope is the parent class and all subclasses of the parent class (Figure 13). Oracle also provides implicit dereferencing of REFs. For example, consider the following:

```
CREATE TYPE SAMPLE AS OBJECT (
        name VARCHAR2(30),
        sample_origin REF sample);
```

If X represents an object of type SAMPLE, then the SQL expression:

x.sample_origin.name;

follows the pointer from the sample X to another sample, the original sample of X, and retrieves the sample_origin's name.

There are several other specific advantages that objects offer over a purely relational approach. (Quoted from Oracle 9i Application Developer's Guide- Object-Relational Features, page 1-4, 1-5)

A) Like classes, objects make it easier to model complex, real-world business entities and logic, and the reusability of objects makes it possible to develop database applications faster and more efficiently. By natively supporting object types in the database, ORDBs enable application developers to directly access the data structures used by their applications. No mapping layer is required between client-side objects and the relational database columns and tables that contain the data.

B) Object abstraction and the encapsulation of object behaviors also make applications easier to understand and maintain. Objects can encapsulate operations along with data. Database tables contain only data. Objects can include the ability to perform operations that are likely to be needed on that data. Thus an image (picture) object might include a method to calculate the average intensity of its background. Or a flow cytometry experiment object might have methods to return the experiment's name, experiment protocol, instruments, samples, data files, or even the analysis on the raw data. An application can simply call the methods to retrieve the information.

C) Using object types makes for greater efficiency. Object types and their methods are stored with the data in the database, so they are available for any application to use. Developers can benefit from work that is already done and do not need to recreate similar structures in every application. You can fetch and manipulate a set of related objects as a single unit. A single request to fetch an object from the server can retrieve other objects that are connected to it by object references. For example, you select a gene object and get the gene's name, reference number, and the multiple parts of experiments in which it was studied in a single round-trip between the client and the server.

D) Objects can represent Part-Whole relationships. In a relational system, it is awkward to represent complex part-whole relationships. A lens and a microscope have the same status in a table for stock items. To represent lenses as parts of microscopes, you must create complicated schemas of multiple tables with primary key-foreign key relationships. Object types, on the other hand, give you a rich vocabulary for describing part-whole relationships. An object can have other objects as attributes, and the attribute objects can have object attributes too. An entire parts-list hierarchy can be built up in this way from interlocking object types.

E) Objects are organic. Object types let you capture the "thingness" of an entity, that is, the fact that its parts make up a whole. For example, an address may need to contain a number, street, city, state, and zip code. If any of these elements is missing, the address is incomplete. Unlike an address object type, a relational table cannot express that the columns in the table collectively make up an organic whole.

A key advantage of the ORDM is that the industry-standard query interfaces can be used with it. Although the SQL was designed for relational databases, it has been extended by the various object-relational database vendors to accommodate user-defined

data types and inheritance. The industry-standard Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) interfaces have been extended to deal with the different results that object-relational SQL may return.

A disadvantage of ORDB is currently no nationally accepted standard for defining or implementing object-relational data models. For example, DB2 v8 supports table inheritance, but Oracle 9i doesn't; Oracle 9i support nested tables, but DB2 v8 doesn't. There are several other areas where vendors differ in their implementation of ORDMs, including mechanisms to support collections and user-defined data types. Although the new SQL-3 standard provides a complete language and model for the management of persistent complex objects, the implementations among vendors are still different. Besides the above disadvantage, there is an impedance mismatch between the data types used by SQL and the data types used in the programming languages used to create objects, which only serves to make the process more difficult.

Although the incompatibilities currently exist between different ORDMs, it should be noted that leaded by ISO, vendors are working together on the SQL standard to solve the incompatibilities[114],[115]. The mismatch between the data types used by SQL and the data types used in the programming language are being solved by ISO, using the techniques of mapping SQL types to XML types[116],[117].

4.6 Database Schema Exchange

Because in different ORDB or relational database specifications the SQL syntaxes are different and adhere to strict formats, there are instances where a given SQL implementation of a database schema works on one database system but not on another database system. Sometimes a given SQL data type is incapable of capturing the semantics of the information being exchanged. As schemas become more complicated, there will be a need to query the data models for information about themselves; this information is known as metadata, and allows a data model's user to learn the context and assumptions upon which the data model is built. The XML schema[118] has solidly emerged as the technology to solve the problem of the database schema exchange[119],[120],[121]. XML allows data to be structured in a hierarchical format. The "extensibility" of XML allows user-defined markup tags and data types, to fit a specific domain. The new associated XML schema is used to define the structure; it has been improved more than its previous candidate DTD (Document Type Definition). It has been shown that object-oriented structures can be mapped directly to the structure of XML[122], which makes it eminently suited for transporting and exchanging data from relational and object-relational database. The flexibility and portability of XML and its schema have gained them industry-wide acceptance.

4.7 Data Model Selection and Summary

As we discussed above, since the object-relational data model combines the best of both the relational model and the object-oriented model, it supports all of the key database techniques and the OO methodologies, such as relational technologies, user-defined data types and functions, complex data and object definitions, encapsulation, and inheritance. Therefore, this project uses the object-relational data model. The ORDM

makes it easier to model complex, real-world biological experimental information entities and logic, and the reusability of the entities (objects) makes it possible to develop database applications faster and more efficiently. XML and XML schema is used in the project to convey the results of a data query and database schema exchange.

Chapter 5. The Object Ontologies (OO) for Experimental Biology

As we discussed in the Chapter 3, the project needs to define a set of common information entities to describe the event and data of each biological experimental method. By analyzing the characteristics of gel electrophoresis, flow cytometry, microarray, microscope image, and mass spectrometry, this chapter is going to discuss the information objects involved in the experimental methods; introduce the existing standards which could be adapted to describe the information. And then, for each experimental method, a set of information entities is going to be defined to describe the event and data of the method. The entities will be able to completely characterize the results of each experiment. Because the design of the information entities for each method is going to use a single coherent set of information object definitions, a consistent data definition strategy will be proved that can handle those five modalities within the single coherent set of information object definitions. Because of a similar analysis approach applied to every experimental method, we will give out the detail analyses only on gel electrophoresis and flow cytometry, but less detail on microarray, mass spectrometry, and microscope image in the following paragraphs.

5.1 Gel Electrophoresis

In this section, a set of information entities is going to be defined to describe the event and data of the gel electrophoresis experiments. The following paragraphs give out the analysis process about the entities. It includes an introduction to gel electrophoresis; illustrations of the experimental technique of Western blot and 2D gel electrophoresis; example experiments of Western blot; and the information objects that are used to describe the experiments.

A. Gel Electrophoresis

Gel electrophoresis[123] is a method that separates macromolecules-either nucleic acids or proteins-on the basis of size, electric charge, and other physical properties. A gel is a colloid in a solid form. The term electrophoresis describes the migration of charged particle under the influence of an electric field. Thus, gel electrophoresis refers to the technique in which molecules are forced across a span of gel, motivated by an electrical current.

Many important biological molecules such as amino acids, peptides, proteins, nucleotides, and nucleic acids, posses ionisable groups and, therefore, at any given pH, exist in solution as electrically charged species either as cations (+) or anions (-). Depending on the nature of the net charge, the charged particles will migrate either to the cathode or to the anode.

Separation of large (macro) molecules depends upon two forces: charge and mass. When a biological sample, such as proteins or DNA, is mixed in a buffer solution and applied to a gel, these two forces act together. The electrical current from one electrode

repels the molecules while the other electrode simultaneously attracts the molecules. The frictional force of the gel material acts as a "molecular sieve," separating the molecules by size. During electrophoresis, macromolecules are forced to move through the pores when the electrical current is applied. Large molecules have difficulty getting through the holes in the well (pore) matrix. Small molecules move easily through the holes. Because of this, large fragments will lag behind small fragments as the molecules migrate through the gel, as shown in the below figure. Their rate of migration through the electric field depends on the strength of the field, size and shape of the molecules, relative hydrophobicity of the samples, and on the ionic strength and temperature of the buffer in which the molecules are moving. After staining, the separated macromolecules in each lane can be seen in a series of bands spread from one end of the gel to the other.

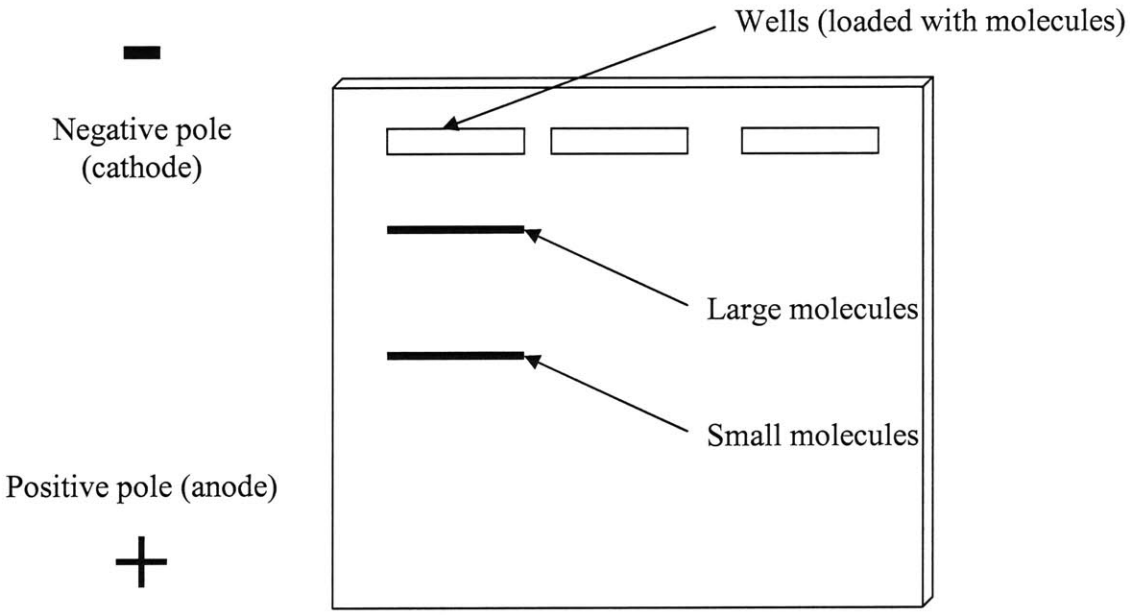


Figure 14 Separation of molecules in gel electrophoresis.[124]

Gel electrophoresis is one of the staple tools in molecular biology and is of critical value in many aspects of genetic manipulation and study. One use is the identification of particular DNA molecules by the band patterns they yield in gel electrophoresis after being cut with various restriction enzymes. Viral DNA, plasmid DNA, and particular segments of chromosomal DNA can all be identified in this way. Another use is the isolation and purification of individual fragments containing interesting genes, which can be recovered from the gel with full biological activity. (Ref: http://www.bergen.org/AAST/Projects/Gel/)

Two kinds of gel electrophoresis techniques are going to be introduced here: one is Western blot, also called as 1D gel; another is 2D gel electrophoresis.

B. Western Blot

Western blot[125],[126],[127],[128] analysis is the most commonly used immunochemical technique for detection of epitope-tagged proteins. It allows the detection of epitope-tagged proteins in complex mixtures such as cell or membrane extracts (for instance, Canfield and Levenson[129], 1993). When combined with immunoprecipitation, it can reveal information about the interaction of the tagged protein with other cell components (for instance, as in Dietzen, Hastings and Lublin[130], 1995).

In a Western blot, proteins are electro-phoretically separated on an acrylamide gel, then transferred to a membrane detected with one or more antibodies. The following picture shows a typical experiment process of the quantitative gel electrophoresis (Western blot) methods. The process includes treating cells, lysis, gel electrophoresis, blotting, applying antibodies, scanner (for instance, FluorImager[131]) reading out gel images and data processing by software (for instance, ImageQuant[132]).

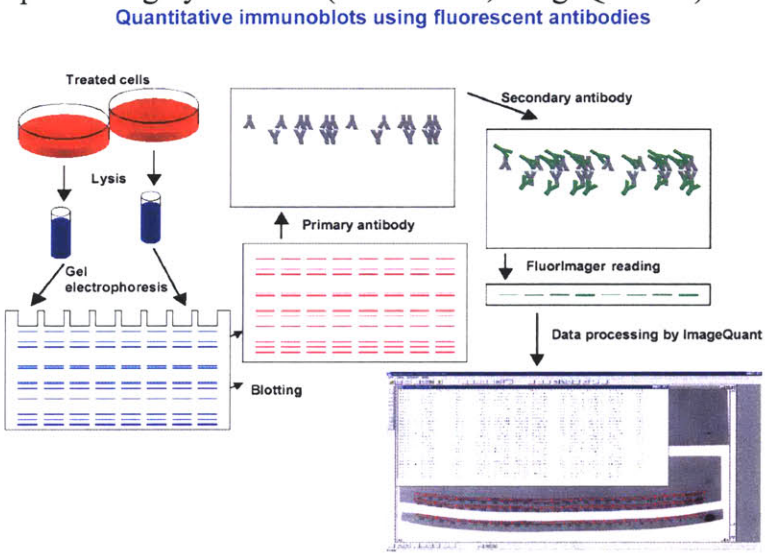


Figure 15 An example experimental method of quantitative immunoblots using fluorescent antibodies (Ref: Suzanne Gaudet, 2002)

C. 2D Gel Electrophoresis

This is a method for the separation and identification of proteins in a sample by displacement in 2 dimensions oriented at right angles to one another. This allows the sample to separate over a larger area, increasing the resolution of each component. A good reference on this subject is Görg et al, (2000)[133].

Isoelectric focusing (IEF)[134] is used in the 1st Dimension (Righetti, P.G., 1983)[135]. This separates proteins by their charge (pI, isoelectric point). Sodium Dodecyl Sulphate Polyacrylamide Gel Electrophoresis (SDS-PAGE)[136] is used in the 2nd Dimension. This separates proteins by their size (Molecular Weight, MW). The procedure is known as ISO-DALT: iso for isoelectric focusing and dalt for Dalton weight.

Isoelectric focusing (IEF) can be described as electrophoresis in a pH gradient set up between a cathode and anode with the cathode at a higher pH than the anode. Because of the amino acids in proteins, they have amphoteric properties and will be positively charged at pH values below their IpH and negatively charged above. This means that proteins will migrate toward their IpH. Most proteins have an IpH in the range of 5 to 8.5.

Under the influence of the electrical force the pH gradient will be established by the carrier ampholytes, and the protein species migrate and focus (concentrate) at their isoelectric points. The focusing effect of the electrical force is counteracted by diffusion which is directly proportional to the protein concentration gradient in the zone. Eventually, a steady state is established where the electrokinetic transport of protein into the zone is exactly balanced by the diffusion out of the zone.

Sodium dodecyl sulphate (SDS) is an anionic detergent which denatures proteins by "wrapping around" the polypeptide backbone - and SDS binds to proteins fairly specifically in a mass ratio of 1.4:1. In so doing, SDS confers a negative charge to the polypeptide in proportion to its length – i.e., the denatured polypeptides become "rods" of negative charge cloud with equal charge or charge densities per unit length. It is usually necessary to reduce disulphide bridges in proteins before they adopt the random-coil configuration necessary for separation by size: this is done with 2- mercaptoethanol or dithiothreitol. In denaturing SDS-PAGE separations therefore, migration is determined not by intrinsic electrical charge of the polypeptide, but by molecular weight. (Ref: http://web.uct.ac.za/microbiology/sdspage.html)

D. An Example of Western Blot Experiment

The following paragraphs in italic font format are the real Western blot data report got from Loel Kathmann, PNNL. It gives us a feeling that how the Western Blot experimental data looks like.

110703 HMEC TNF+/-PD pERK, total ERK western

Purpose

To measure pERK and total ERK levels in HMEC cells after 1) TNF stimulation and 2) inhibition of EGFR phosphorylation using PD153035 followed by TNF stimulation. This is a repeat of a previous experiment to check for reproducibility. See document 102703 HMEC ELISA panel pEGFR notes.doc for pERK image from previous sample set.

Sample Set

102903 HMEC TNF+/- PD

Note: Same sample set was used for pTyr probe on 103003

Sample Description

Human Mammary Epithelial Cells (HMEC) were plated in DFCI-1 media at 1.5M per 100mm plate (Falcon) on day 1 and allowed to condition for 28hours. On day 2 the cells were rinsed once with and them administered 10mL DFB (DFCI-1 base +P/S, L-glut, 0.1% BSA) and allowed to condition for 20hours. At time of treatment (t=-15min) media from plates was aspirated and 6mL DFB +/- 20nM PD153035 was added as a pretreatment. Plates were incubated 15 minutes at 37°C, 5% CO2. At t=0, 0.5mL TNF stock solution at 260ng/mL was added to the 6mL of pretreatment media to yield final concentration TNF=20ng/mL. Samples were prepared for western blot at t=0, 5, 15, 30 min, 1h, 2h such that plates were placed on ice, rinsed once in 4mL cold PBS, scraped in 0.6mL lysis buffer (150mM NaCl, 10mM Tris, 10mM NaF, 1%NP40 pH=7.4; 2mM NaO and protease inhibitors added just before use), sonicated, centrifuged, and the supernatant aliquoted and frozen at -80°C until analysis.

pERK Western Conditions:

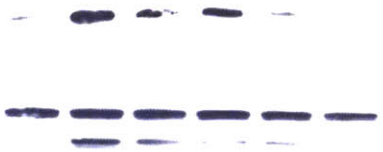
- *protein concentration excel file: 110603 PD, LY HMEC BCA*
- *#µg loaded per sample: 25ug*
- *% gel:12*

- *running conditions: 100V, 120min*
- *Markers: Kaleidoscope (Bio-Rad #161-0324), biotin-labeled (CST #7727)*
- *Membrane type: Nitrocellulose*
- *Transfer conditions: 280mA 30 min*
- *Blocking agent: 5% milk in TTBS*
- *Primary antibody conditions: pERK (CST #9101S, p44/p42 MAPK Thr202/Tyr204), 1:1000@4 degree o/n in 5% milk in TTBS*
- *Secondary antibody conditions: alpha-rabbit (Southern Bitechnology #4050-02, goat alpha-rabbit-HRP), 1:1000 + alpha-biotin (kit component of CST #7727), 1:1000 @ RT 1hr in 5% milk in TTBS*
- *Chemiluminescent exposure times run: 2, 3, 5min*

Results

- *12 bit lumi-imager file names: 110703 HMEC pERK 2min*
- *8 bit converted image shown below: 110703 HMEC pERK 3 min 8 bit*
- *Loading Order:*

Top: TNF +/- LY treatment discussed in 110703_HMEC_TNF_LY_pERK_ERK.doc
Bottom: CST marker, TNF 0, 5, 15,30min, 1h, 2h, PD/TNF 0, 5, 15, 30min, 1h, 2h



ERK Western Conditions

- *The pERK blot was stripped and reprobed for total ERK.*
- *Conditions for ERK blot are exactly the same as those for the pERK blot with the following single exception: Primary Antibody is ERK (Santa Cruz #sc-)*
- *Chemiluminescent exposure times run:*

Comments
The pERK and ERK expression pattern proves to be reproducible.

E. The Information Object Definition (IOD) of Western Blot Experiment

For the above data, we can normalize them into the following tables: project description, experiment description, sample set, sample description, sample treatment, protocol, parameter set, raw data and processed data. After we examined the data from a lot of western blot experiments since we began the database design last year, we found that those data can be grouped into five packages: project (also called as study plan), sample, experiment, high level analysis, and administration (as shown in Figure 16).

- Project package describes the information about biological projects, including descriptions, hypotheses, references, and project reports.
- Sample package describes the information about ideal biological samples, sample sets, derived samples, and measured samples.
- Experiment package describes the information about biological experiments, including experimental protocols, hardware, and software, the descriptions of the experiments, sample treatments, raw data/results, and preprocessed data.
- High level analysis package describes the information about the advanced analyses of results.

- Administration package describes the information about person (experimenter, user) and laboratory, security, authorization, and auditing record.

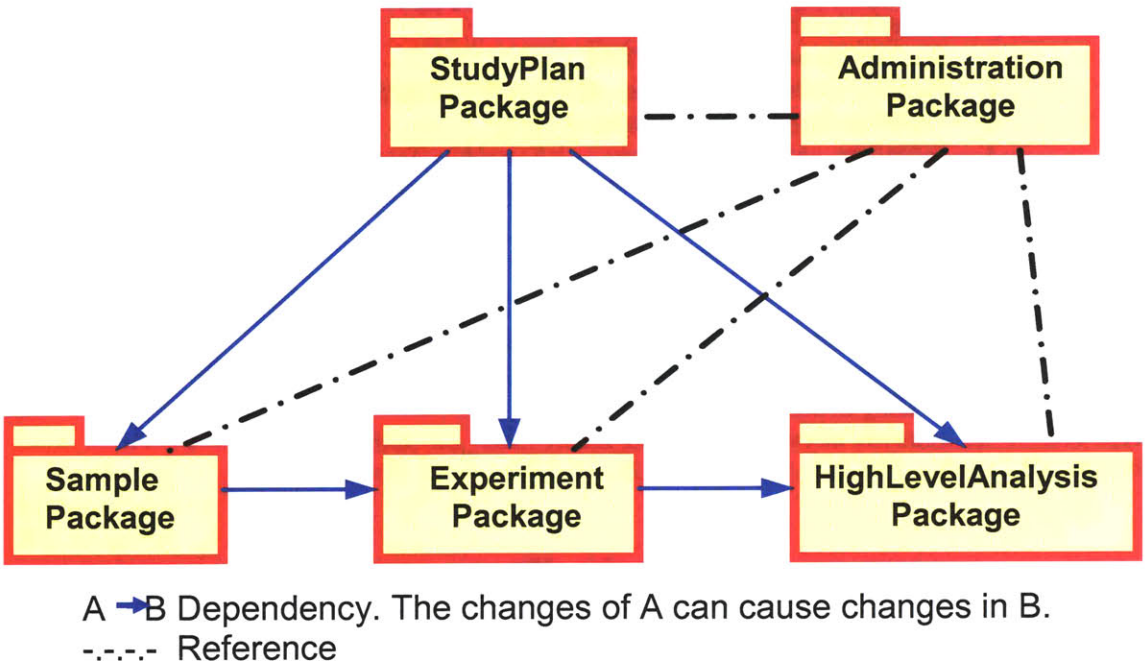


Figure 16 Biological experimental data can be grouped into five packages: project (also called as study plan), sample, experiment, high level analysis, and administration package.

A study plan / project can involve multiple samples, do multiple experiments, and have several result sets from multiple high level analyses; one sample can be used in many experiments; a set of high level analysis can be associated with multiple results generated from different experiments. The administration information, specially auditing, is stamped in every record of the five packages.

Following the idea of the five package's design and object-oriented concept, the first schema we designed for western blot is shown in Figure 17. The figure is a UML diagram showing the inheritance of the information objects. For examples, the root element is WesternBlotIOD, which has four attributes (date_created, created_by, date_modified, modified_by). Since all other entities in the diagram are children or grandchildren of the root element, they all inherit the four auditing attributes from the root element. It is the same thing that the entities "StudyPlanDescr", "Hypothesis", "Reference", "Ontology", and "ProjectReport" share the same study plan UID (unique identifier) because they have a same parent – "StudyPlan". Note that the diagram doesn't show the reference relationship between two entities which are in the different packages; instead, I use the postfix "_ref" to represent the reference to make the diagram clean. The schema is highly modular, structured and extendable. The following paragraphs will discuss how the schema gets extended without a change of the structure.

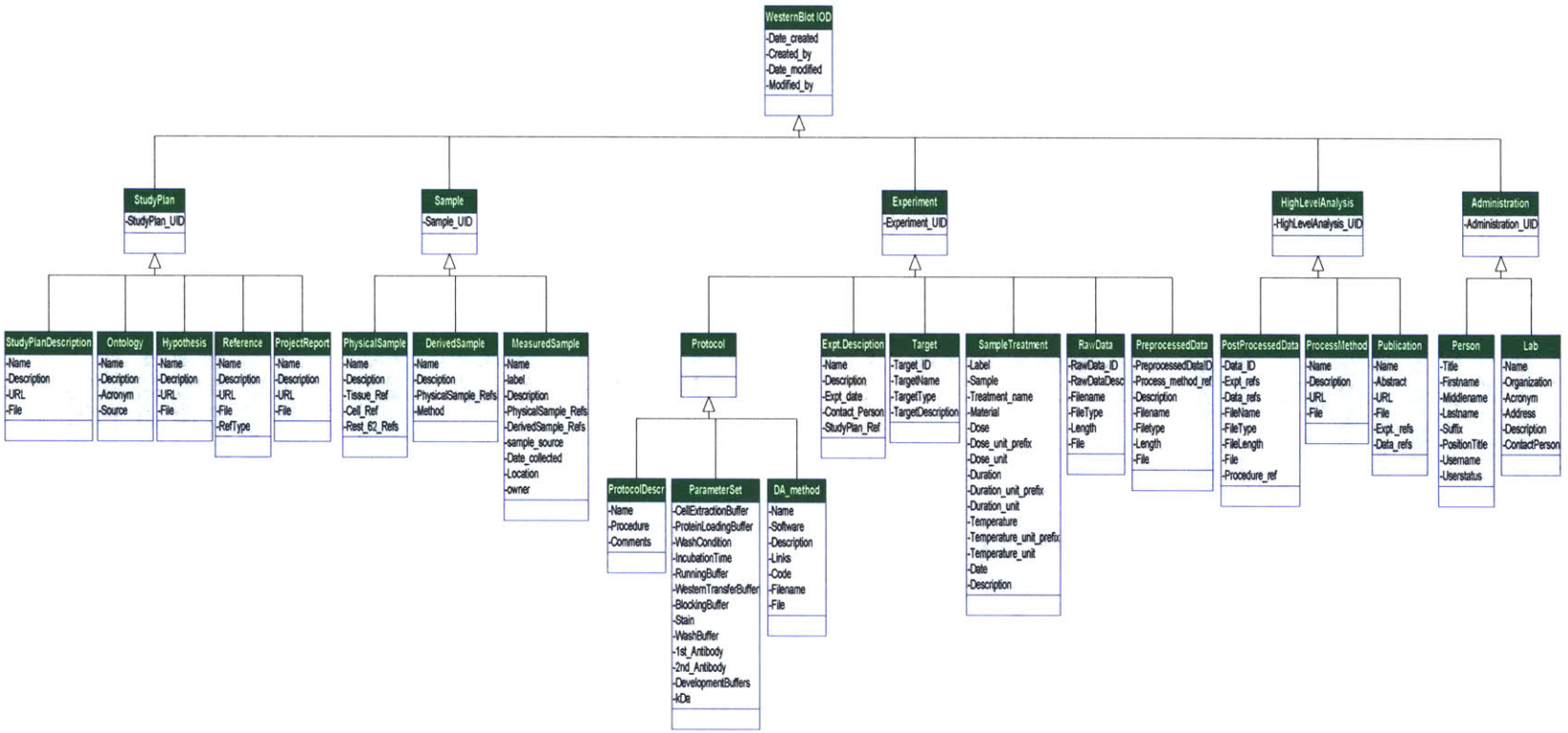


Figure 17 The first version of the information object definition of Western blot experiments

Chris Taylor, et al developed a systematic approach to modeling, capturing and disseminating proteomics experimental data[137] in 2003. In their design, they presented a schema to describe the event and data of 1D gel and 2D gel electrophoresis, as shown in Figure 18. The attributes describing 1D gel that they summarized are the description, raw image, annotated image, software version, warped image, warping map, equipment, percent acrylamide, solubilization buffer, stain details, protein assay, in-gel digestion, background, pixel size, denaturing agent, mass start / end, run details, and the information about band, etc.. The information entities about gel process and images in their design are valuable for our schema design. But their schema is unstructured; it can not be extended to other experimental methods. It is not object oriented. Therefore, we have redesigned and summarized a completed schema for Western blot IOD as shown in Figure 21, based on their ideas and our previous design for Western blot.

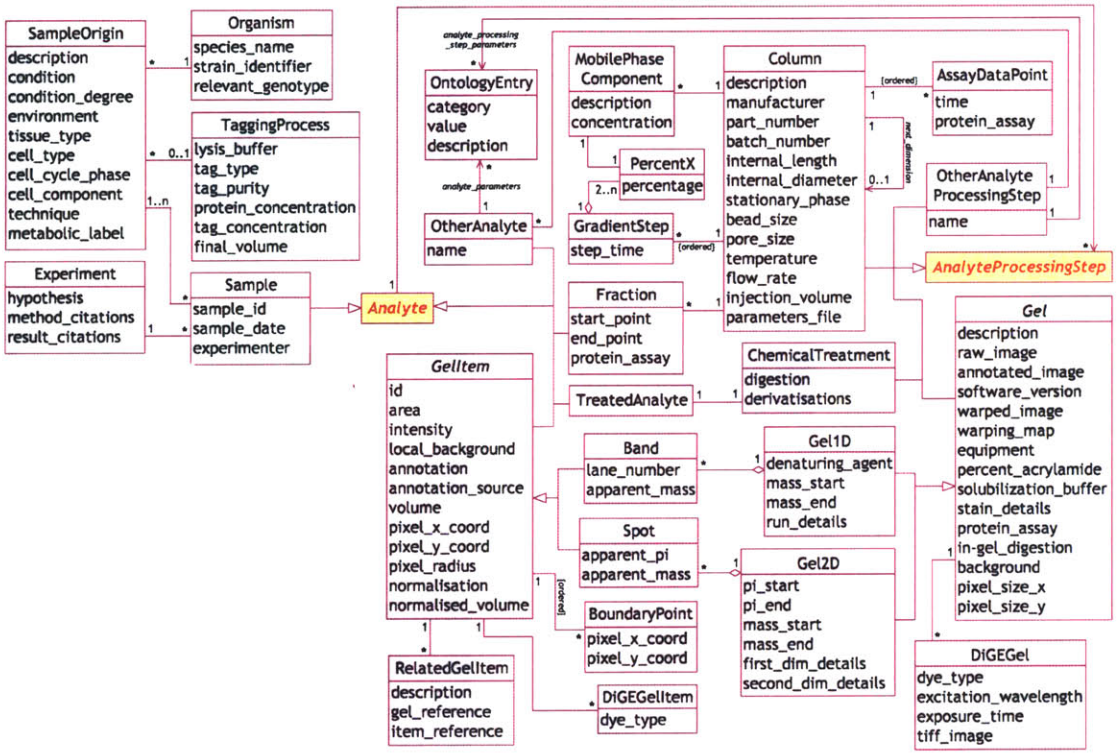


Figure 18 The PEDRo UML class diagram provides a conceptual model of proteomics experiment data, which forms the basis for the XML and relational schemas. Colors denote: blue – sample generation; pink – sample processing.

Let us look at Figure 19 first. Figure 19 shows some detail analysis on the PEDRo SQL implementation and redesigning process of the information entities. For examples, the table "experiment" of PEDRo describes the hypothesis, method and result citations of an experiment; in our design, instead, we have the entities hypothesis, reference, and report in the project package; and we use a table "project_experiment" to describe the relation between a project and an experiment. The true meaning of the table "sample" of PEDRo is to describe the relation between a sample and an experiment. Therefore, according to the rules of data normalization (five normal forms)[138], it is better for us to re-normalize it: put the sample information, such as sample date into the table "measured

sample"; put the experiment information, such as experimenter into the table "experiment description"; and then in the relation table of the sample and experiment, only two attributes are needed, which are sample ID and experiment ID, as shown in Figure 20. The organism of PEDRo is an ideal sample of our design; the tagging process belongs to the tree of the sample treatment; the "sampleorigin" is equivalent to the sample set in the derived sample of our design; the purpose of the "sample_mtm_sampleorigin" is to describe the relation between measured sample and sample set. Because the relation is "one to many", we can put it into the table "measured sample" directly. The conversion of the rest entities of PEDRo is similar to the above analysis process. After several iterations and adapting some ideas from other standards, such as CDISC, DICOM, and others mentioned in the Chapter 2, a new information object definitions of Western blot is shown in Figure 21.

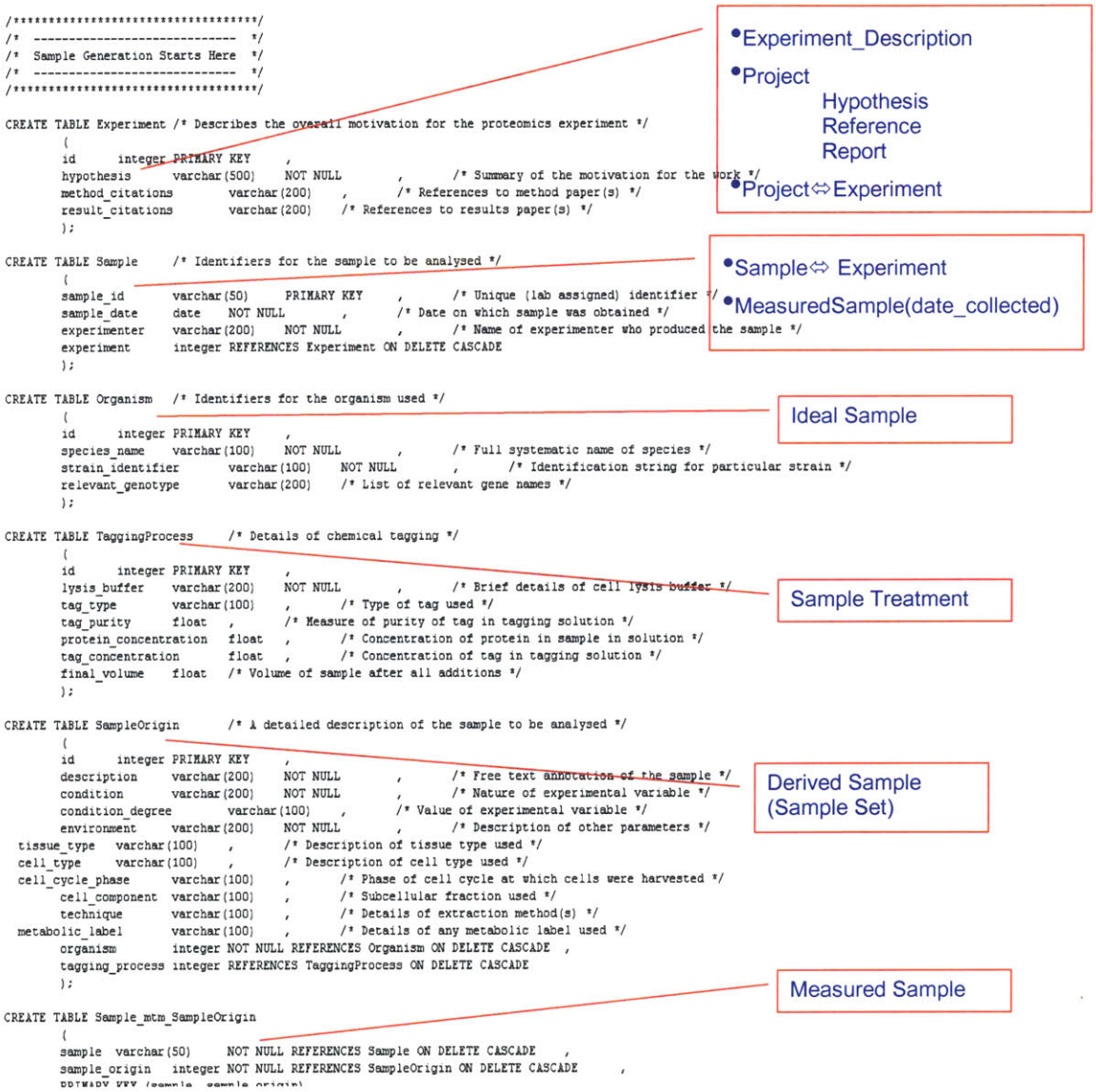


Figure 19 The re-interpretation and redefinition of the PEDRo entities

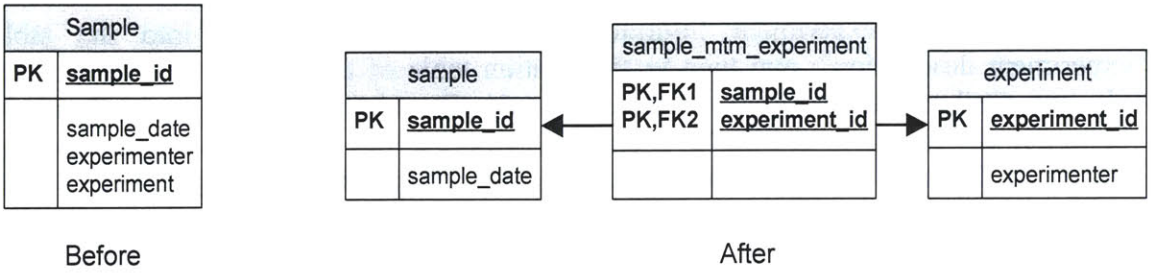


Figure 20 Normalizing the entity "Sample" of the PEDRo schema: "Before" is a non-normalized representation used in the PEDRo schema because experimenter and sample date are stored more than once for samples that are used in more than one experiment; "After" is a normalized representation using an extra relationship table. The relation between sample and experiment is "many to many" relationship.

In the diagram of Figure 21, we have adapted several design ideas from PEDRo, such as, images are grouped into raw image, annotated image, warped image, and warping map. We have taken account of the information about instruments and software, unit, and the detail information about an image, such as the description of a band in a gel. One principal rule of the entities design is that we try to keep text descriptions as minimal as possible. For example, the sample treatment should be abstracted as a table format with numerical values, like,

Sample_Treatment (sampleID, treatment_name, label, chemical_reference, dose, dose_unit_prefix, dose_unit, concentration, concentration_unit_prefix, concentration_unit, duration, duration_unit_prefix, duration_unit, pH, temperature, temperature_unit, date, description)

This makes it possible for us to build a mathematical model to associate image numerical data, such as intensity with the values of sample treatment. It also facilitates data mining and data analysis.

The new schema proves that we can easily add new objects into the schema without any change of the structure. The schema supports any kind of data and events generated from Western blot experiments. It can store any information about labs, experimenters, projects, samples, sample sets, experiments, protocols, instruments, images, results, and so on. For example, the example experiment in the previous section D can be translated into database tables as shown in the following tables (The prefix of "oracle.sql.REF@" means that the data is an object reference / pointer). All information described in the report can be uploaded into the database tables completely.

Table 2 The database tables translated from Loel Kathmann's lab notes

STUDYPLANDESCR

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	STUDYPLAN_UID	NAME	DESCRIPTION	URL	FILENAME	FILETYPE	FILELENGTH	BINARYFILE
2004-1-21.8.4. 31. 0	2004-1-21.8.4. 31. 0	null	null	PNNL_bio_Loel_project1	HMEC TNF+/-PD pERK, total ERK	To measure pERK and total ERK levels in HMEC cells after 1) TNF stimulation and 2) inhibition of EGFR phosphorylation using PD153035 followed by TNF stimulation. This is a repeat of a previous experiment to check for reproducibility. See document 102703 HMEC ELISA panel pEGFR notes.doc for pERK image from previous sample set.	http://bsioracleserve.pnl.gov:9080/ExperiBase	Document1.rtf	rtf	10000	blob

WB_EXPTDESCR

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	EXPERIMENT_UID	EXPTNAME	DESCRIPTION	CONTACTPERSON	EXPTDATE
2004-1-28.1.31. 57. 0	2004-1-28.1.31. 57. 0	null	null	PNNL_bio_loel_pERK_western_1	110703 HMEC TNF+/-PD pERK, total ERK western	null	oracle.sql.REF@e48f028d	2002-11-11 00:00:00.0

WBPROTOCOL_L

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	EXPERIMENT_UID	PROTOCOL	COMMENTS
2004-1-28.1.33. 6. 0	2004-1-28.1.33. 6. 0	null	null	PNNL_bio_loel_pERK_western_1	oracle.sql.REF@88b9153b	none

CELL

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	SAMPLE_UID	ABBREV	COMMONNAME	GENUSSPECIES	TYPE	SOURCE	LABEL	CONTENT
2004-1-21.10.18. 0. 0	2004-1-21.10.18. 0. 0	null	null	PNNL_bio_HMEC	HMEC	Human Mammary Epithelial Cell	Human	Epithelial Cell	PNNL Loel	HMEC	null

WB_SAMPLEORIGIN

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	SAMPLE_UID	NAME	LABEL	DESCRIPTION	CONDITION	CONDITION_DEGREE	ENVIRONMENT	TISSUE_TYPE	CELL_TYPE	CELL_CYCLE_PHASE	CELL_COMPONENT	TECHNIQUE	METABOLIC_LABEL	ORGANISM	TAGGING_PROCESS
2004-1-21.10.36. 47. 0	2004-1-21.10.36. 47. 0	null	null	PNNL_bio_Loel_Sample_set_102903	102903 HMEC TNF+/-PD	102903 HMEC TNF+/-PD	Note: Same sample set was used for pTyr probe on 103003	null	null	null	null	Epithelial Cell	G1, S, G2	Human Mammary Epithelial Cell	null	null	null	null

WESTERNBLOT_MSP

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	SAMPLE_UID	NAME	LABEL	DESCRIPTION	PHYSICAL_SAMPLE	DERIVED_SAMPLE	SAMPLE_SOURCE	DATE_COLLECTED	LOCATION	OWNER
2004-1-21.9.0. 41. 0	2004-1-21.9.0. 41. 0	null	null	PNNL_bio_pERK_westernblot_102903_0	102903 HMEC TNF+/-PD time 0	TNF 0'	Samples were prepare for western blot at time 0 using the treatment in the Experiment with uid(PNNL_bio_loel_pERK_western_1)	oracle.sql.REF@e48b028d	oracle.sql.REF@e4ea028d	Loel Kathmann	2002-11-12.12.0. 0. 0	oracle.sql.REF@88c7153b	oracle.sql.REF@e48f028d

WB_TREATMENT

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	EXPERIMENT_UID	LABEL	SAMPLE	TREATMENT_NAME	MATERIAL	DOSE	DOSE_UNIT_PREFIX	DOSE_UNIT	CONCENTRATION	CONCENTRA_UNIT_PREFIX	CONCENTRA_UNIT	PH	DURATION	DURATION_UNIT_PREF	DURATION_UNIT	TEMPERATURE	TEMPERATURE_UNIT_P	TEMPERATURE_UNIT	TREATMENT_DATE	DESCRIPTION
2004-1-21.8.36. 9. 0	2004-1-21.8.36. 9. 0	null	null	PNNL_bio_loel_pERK_western_1	102903 HMEC TNF+/- PD	oracle.sql.REF@e4ea028d	step 1	oracle.sql.REF@e4ef028d	1.5	null	oracle.sql.REF@e495028d	null	null	null	null	28	null	oracle.sql.REF@e4d2028d	null	null	null	2002-11-11.12.0. 0. 0	HMEC cells were plated in DFCI-1 media at 1.5M per 100mm plate (Falcon) on day 1 and allowed to condition for 28 hours.
2004-1-21.8.52. 54. 0	2004-1-21.8.52. 54. 0	null	null	PNNL_bio_loel_pERK_western_1	102903 HMEC TNF+/- PD	oracle.sql.REF@e4ea028d	step 5	oracle.sql.REF@e4f5028d	.5	oracle.sql.REF@e4e2028d	oracle.sql.REF@e4d0028d	260	null	oracle.sql.REF@e4d3028d	null	null	null	null	null	null	null	2002-11-12.12.15. 0. 0	At t=0, 0.5mL TNF stock solution at 260ng/mL was added to the 6mL of pretreatment media to yield final concentration TNF=20ng/mL.
2004-1-21.8.49. 43. 0	2004-1-21.8.49. 43. 0	null	null	PNNL_bio_loel_pERK_western_1	102903 HMEC TNF+/- PD	oracle.sql.REF@e4ea028d	step 4	oracle.sql.REF@e4f3028d	null	null	null	.05	null	null	null	15	null	oracle.sql.REF@e4d1028d	37	null	oracle.sql.REF@e4b1028d	2002-11-12.12.15. 0. 0	Plates were incubated 15 minutes at 37 degree Celsius, 5%

																						CO2.
2004-1-21.8.46. 39. 0	2004-1-21.8.46. 39. 0	null	null	PNNL_bio_loel_pERK_western_1	102903 HMEC TNF+/- PD	oracle.sql.REF @e4ea028d	step 3	oracle.sql.REF @e4f1028d	6	oracle.sql.REF @e4e2028d	oracle.sql.REF @e4d0028d	null	null	null	null	15	null	oracle.sql.REF @e4d1028d	null	null	null	2002-11-12.11.45. 0. 0	At time of treatment (t=-15) media from plates was aspirated and 6ml DFB +/- 20nM PD153053 was added as a pretreatment.
2004-1-21.8.40. 10. 0	2004-1-21.8.40. 10. 0	null	null	PNNL_bio_loel_pERK_western_1	102903 HMEC TNF+/- PD	oracle.sql.REF @e4ea028d	step 2	oracle.sql.REF @e4f1028d	10	oracle.sql.REF @e4e2028d	oracle.sql.REF @e4d0028d	null	null	null	null	20	null	oracle.sql.REF @e4d2028d	null	null	null	2002-11-12.12.0. 0. 0	On day 2 the cells were rinsed once with and then administered 10 ml DFB (DFCI-1 base + PIS, L-glut, 0.1% BSA) and allowed to condition for 20 hours.

WB_RAWDATA_DESCR

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	EXPERIMENT_UID	RAWDATA_ID	DESCRIPTION	SOFTWARE_REF	EQUIPMENT_REF	PERCENT_ACRYLAMIDE	PROTEIN_ASSAY	BACKGROUND	PIXEL_SIZE_X	PIXEL_SIZE_Y	MASS_START	MASS_END
2004-1-21.12.6. 46. 0	2004-1-21.12.6. 46. 0	null	null	PNNL_bio_loel_pERK_western_1	110703 HMEC pERK 2min	TNF +/- LY treatment discussed in 110703_HMEC_TNF_LY_pERK_ERK.doc	null	null	.12	null	null	null	null	null	null
2004-1-21.12.7. 11. 0	2004-1-21.12.7. 11. 0	null	null	PNNL_bio_loel_pERK_western_1	110703 HMEC pERK 3min	TNF +/- LY treatment discussed in 110703_HMEC_TNF_LY_pERK_ERK.doc	null	null	.12	null	null	null	null	null	null

WB_RAWIMAGE

DATE_CREATED	DATE_MODIFIED	CREATED_BY	MODIFIED_BY	EXPERIMENT_UID	RAWDATA_ID	DESCRIPTION	FILENAME	FILETYPE	FILELENGTH	BINARYFILE
2004-1-21.9.26. 54. 0	2004-1-21.9.26. 54. 0	null	null	PNNL_bio_loel_pERK_western_1	110703 HMEC pERK 2min	TNF +/- LY treatment discussed in 110703_HMEC_TNF_LY_pERK_ERK.doc	1213_8_8hr_2.tif	tif	10000	blob
2004-1-21.9.28. 23. 0	2004-1-21.9.28. 23. 0	null	null	PNNL_bio_loel_pERK_western_1	110703 HMEC pERK 3min	CST marker, TNF 0,5,15,30min, 1h, 2h, PD/TNF 0,5,15,30min,1h,2h	1213_8_8hr_4.tif	tif	10000	blob

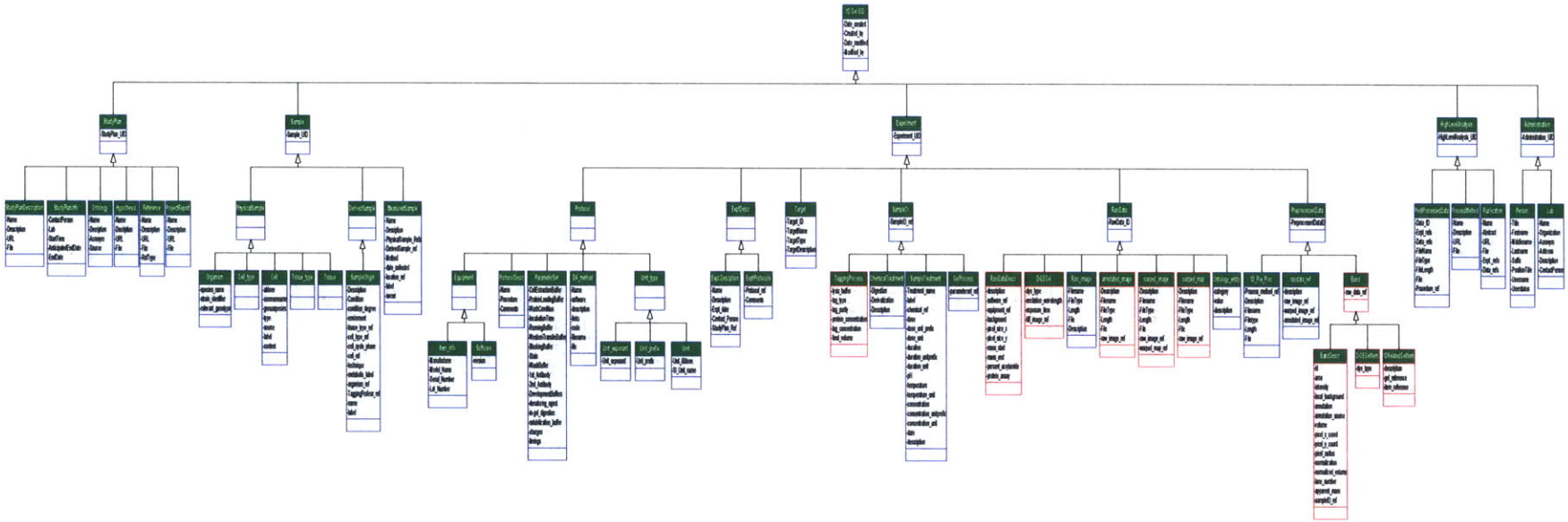


Figure 21 A new schema combining the previous IOD of Western blot experiments with the PEDRo schema.

F. The Information Object Definition of 2D Gel Electrophoresis Experiment

Using the same analysis and design process, we can design a set of the information object definition for 2D gel electrophoresis based on the PEDRo schema and our Western blot IOD. The schema is shown in Figure 22. The differences between Western blot IOD and 2D Gel IOD are shown in red color. I.e., gel process is different from the Western blot protocol; in 2D gel image, the interested element is spot instead of the band in 1D gel image. The rest entities are the same as the Western blot IOD. This proves that we can reuse the object definitions between two methods and keep conceptual consistency.

G. Discussion and Summary of Gel Electrophoresis' IODs

The above discussion illustrated that we can group the data of gel electrophoresis experiment into five packages, which are project, sample, experiment, high level analysis, and administration. Basically, the project package consists of a description of the project, contact info, ontology, reference, hypothesis, and project report. The sample package has three types, which are ideal sample (or physical sample), derived sample (can be sample set), and measured sample. The experiment package includes experimental protocol, description, sample treatment, raw data, and preprocessed data. The high level analysis records publications, or any advanced data analysis results which may use the raw data from multiple experimental methods. The administration package stores the information about person (user, experimenter), lab, security and any information about auditing events. The administration package and project package have their own nearly-fixed structure, which will be proved that they do not vary with experimental methods. The entities in the ideal sample domain are common to any biological research. This kind of separation of information objects also provides freedoms so that: one study plan can have multiple samples, multiple experiments and multiple high level analyses; one sample can be used in several experiments; one set of experiment results can be analyzed in multiple high level analysis processes; an experiment can have multiple sample treatment steps, multiple stain steps, multiple results, and multiple data analyses; auditing information, such as date created / modified and contact person is recorded in every information entity. The consistent object inheritance allows us to be able to implement variant object reference. For example, in the sample treatment table, the sample refers to the "Sample" data type; this means that any children data type of sample is allowed; i.e., the reference may be a cell data type, or an organism data type, or a sample set. This reflects real situations in biological experiments.

The IODs of Western blot and 2D gel electrophoresis are currently very complete ontology. They were summarized from a lot of gel electrophoresis experiments done by several labs and persons. They are able to completely characterize the results of gel electrophoresis experiments. The proposed ontology for 1D and 2D gel data by Chris Taylor et al, and the way describing LIMS' data by CDISC and DICOM, have been used as the basis of the definitions. The IODs are easily updated. The ontology is ready for international standardization consideration.

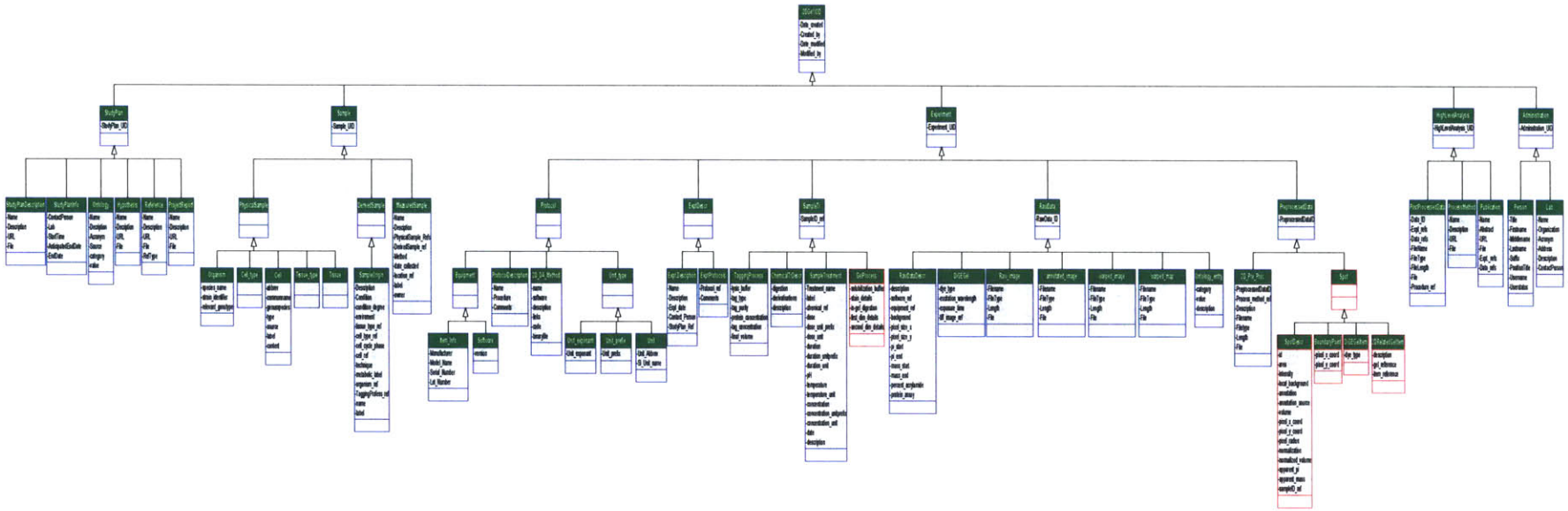


Figure 22 The information object definition of 2D gel electrophoresis experiments

5.2 Flow Cytometry

The following paragraphs give out the analysis about an information object definition for the biological data produced in a Fluorescence Activated Cell Sorting (FACS) experiment. It includes an introduction to flow cytometry; illustrations of the experimental technique of FACS; example experiments of FACS; and a set of information entities to describe the event and data of the experiments. The entities are able to completely characterize the results of the experiments.

A. Flow Cytometry

Flow cytometry is a means of measuring certain physical and chemical characteristics of cells or particles as they travel in suspension one by one past a sensing point. In one way flow cytometers can be considered to be specialized fluorescence microscopes.[139]

We can measure physical characteristics such as cell size, shape and internal complexity and, of course, any cell component or function that can be detected by a fluorescent compound can be examined. So the applications of flow cytometry are numerous, and this has led to the widespread use of these instruments in the biological and medical fields.

In addition, many flow cytometers have the ability to sort, or physically separate, particles of interest from a sample, which can be particularly useful.

B. FACS (Fluorescence Activated Cell Sorting)

FACS is an experimental technique that permits the isolation of different cell populations with different surface antigens stained with different fluorescent antibodies.

In general, flow cytometers use a principle involving the electrostatic deflection of charged droplets similar to that used in ink-jet printers.

Figure 23 illustrates the mechanism of FACS. Cells in the sample are first stained with specific fluorescent reagents to detect surface molecules and are then introduced into the vibrating flow chamber of the FACS. The cell stream passing out of the chamber is encased in a sheath of buffer fluid. The stream is illuminated by laser light and each cell is measured for size (forward light scatter) and granularity (90° light scatter), as well as for red and green fluorescence, to detect two different surface markers. The vibration in the cell stream causes it to break into droplets which are charged and may then be steered by deflection plates under computer control to collect different cell populations according to the parameters measured.

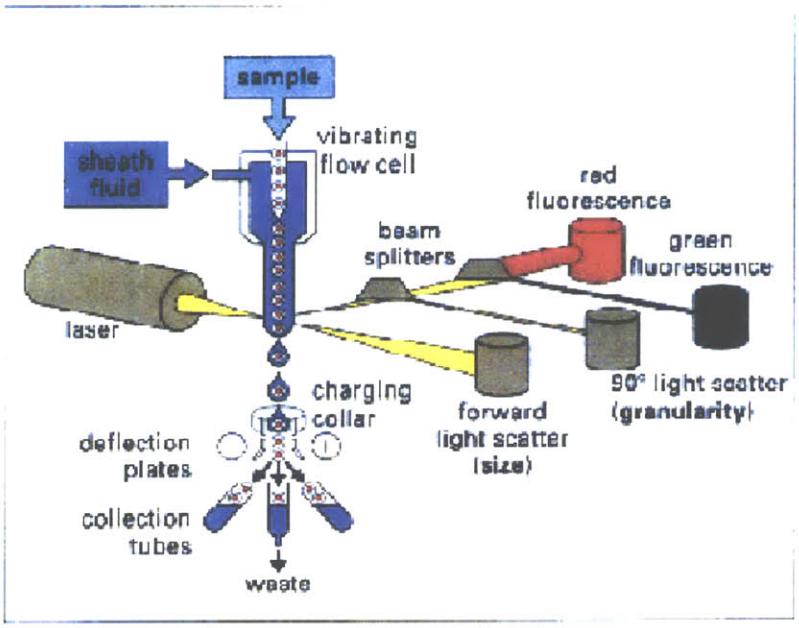


Figure 23 FACS (Fluorescence Activated Cell Sorting)[140].Ref: http://biology.fullerton.edu/courses/biol_426/Web/426GP2/facs.htm

C. Two Examples of FACS Experiments

This section gives out two examples of FACS experiments. One (in the frame) is one of Loel Kathmann's data reports, from PNNL. Another one is from "FlowJo.com", which is one of their demo data.

Example 1: pHMEC EGF commitment, Loel Kathmann

Project
Data Structures for Mapping Cellular Pathways

Experiment

Date	012304
Name	012204 pHMEC EGF commitment
Researcher	Loel Kathmann

Purpose To measure the period of time necessary for HMEC A1 (parental HMEC) cells synchronized via nutrient depletion to be exposed to EGF 50ng/mL such that they will be committed to move through the cell cycle. The effect of the presence of TNFα in samples treated with EGF for 24hr will also be examined.

Description HMEC A1 (parental HMEC) cells synchronized via nutrient depletion for 40hours were exposed to DFB + EGF 50ng/mL for 8, 10, 12, 14, or 24 hours. For cells exposed to EGF for less than 24hours, the media was aspirated after the desired exposure time, the cells washed 2x in DFB and then administered 10mL DFB and returned to the incubator. All samples were collected 24hours after initial EGF exposure such that they were fixed in 70% EtOH and stored to later be used for flow cytometry analysis of cell cycle distribution using propidium iodide.

Cells treated with DFB + TNF at 1, 10, or 20ng/mL +/- EGF 50ng/mL were administered both reagents simultaneously and collected 24hours after TNF/EGF exposure such that they were fixed in 70%EtOH

and stored to later be used for flow cytometry analysis of cell cycle distribution using propidium iodide.

Protocol Links	*Protocol Description*
Insert file name here	Cell Plating and Dosing
Insert file name here	70% EtOH fixation
Insert file name here	PI processing
Insert file name here	Flow Cytometry Setup for Cell Cycle Distribution
Insert file name here	Data Analysis

Protocol Notes and Deviations
Due to time constraints the period of starvation in protocol (insert file name here) step (insert step number here) was reduced to 40 hours.

Sample Set
Date 011404

Description Samples collected via 70%EtOH fixation on 011404 and processed on 012104 for flow cytometry analysis of cell cycle distribution using propidium iodide (PI).

Sample Unique Identifier	*Sample Notes*
012104_starvationtime	control sample collected at time of starvation
012104_logphase_0hr	log phase sample collected at beginning of experiment
012104_logphase_24hr	log phase sample collected at end of experiment
012104_DFB_0hr	40hr starve; collected at beginning of experiment
012104_DFB_24hr	40hr starve; fed fresh DFB; collected at end of experiment
012104_EGF8hr_24hr	40hr starve; 8hr EGF exposure; collected at 24hr
012104_EGF10hr_24hr	40hr starve; 10hr EGF exposure; collected at 24hr
012104_EGF12hr_24hr	40hr starve; 12hr EGF exposure; collected at 24hr
012104_EGF14hr_24hr	40hr starve; 14hr EGF exposure; collected at 24hr
012104_EGF24hr_24hrA	40hr starve; 24hr EGF exposure; collected at 24hr
012104_EGF24hr_24hrB	independent replicate
012104_EGF24hr_24hrC	independent replicate
012104_TNF1_24hr	40hr starve; TNF 1ng/mL exposure for 24hr
012104_TNF10_24hr	40hr starve; TNF 10ng/mL exposure for 24hr
012104_TNF20_24hr	40hr starve; TNF 20ng/mL exposure for 24hr
012104_TNF1EGF_24hr	40hr starve; TNF 1ng/mL + EGF exposure for 24hr
012104_TNF10EGF_24hr	40hr starve; TNF 10ng/mL + EGF exposure for 24hr
012104_TNF20EGF_24hr	40hr starve; TNF 20ng/mL + EGF exposure for 24hr

Results
File types to be linked to include: FACSCalibur raw data
FACSCalibur image data in Word
WinMDI2.8 FCS files
Excel files generated via Cylchred Analysis

Comments and Conclusions

Example 2: An experiment titrating a newly made antibody reagent[141]

The antibody reagent was produced by conjugating purified anti-CD8 antibody to the fluorochrome Flourescein (FITC). A titration experiment was carried out to determine the appropriate reagent concentration for sample staining.

Human peripheral blood mononuclear cell preparation from a single healthy donor was divided into eight tubes. The first tube is the unstained control. The subsequent seven tubes were stained with a

serial two-fold dilution of the anti-CD8-FITC reagent starting at 2 μg/ml. Cells were stained for 20 minutes, washed and resuspended in staining media.

The cell treatments are listed in the below table:

Sample ID	sample	description	stain	dose	unit	duration	duration unit	temperature	temperature unit
1	Human peripheral blood mononuclear cell	The unstained control.							
2	Human peripheral blood mononuclear cell	The subsequent seven tubes were stained with a serial two-fold dilution of the anti-CD8-FITC reagent starting at 2 μg/ml.	anti-CD8-FITC	2	μg/ml	20	minute	room temperature	°C
3	Human peripheral blood mononuclear cell		anti-CD8-FITC	1	μg/ml	20	minute	room temperature	°C
4	Human peripheral blood mononuclear cell		anti-CD8-FITC	0.5	μg/ml	20	minute	room temperature	°C
5	Human peripheral blood mononuclear cell		anti-CD8-FITC	0.25	μg/ml	20	minute	room temperature	°C
6	Human peripheral blood mononuclear cell		anti-CD8-FITC	0.125	μg/ml	20	minute	room temperature	°C
7	Human peripheral blood mononuclear cell		anti-CD8-FITC	0.0625	μg/ml	20	minute	room temperature	°C
8	Human peripheral blood mononuclear cell		anti-CD8-FITC	0.031	μg/ml	20	minute	room temperature	°C

The goal of this titration experiment is to determine the concentration of the anti-CD8-FITC reagent that achieves saturating staining, i.e., maximal staining of the CD8 T cells under defined conditions (20 minutes at room temperature). The data analysis consists of identifying the subpopulation of cells that are stained with anti-CD8 antibody and calculating a median fluorescence intensity statistic on this subpopulation. The computed statistics is used to determine the concentration of anti-CD8-FITC reagent to use in subsequent experiments.

Examples of the instrument settings:[142] (the view is generated by XMLSpy[143] software)

The data files were acquired with 10,000 events collected for each sample. This data can be found on the FlowJo CD in the Basic Tutorial folder or we can download it from www.flowjo.com/tutorial.html. The files are "Sample1(Control).fcs" , "Sample2.fcs" , "Sample3.fcs" , "Sample4.fcs" , "Sample5.fcs" , "Sample6.fcs" , "Sample7.fcs" , and "Sample8.fcs". The files are in Flow Cytometry Standard (FCS) format[144].

Flow Cytometry Standard (FCS) was created to standardize software researchers use to analyze, transmit, and store data produced by flow cytometers and sorters. The format, as shown in Figure 24, needs specific tools to parse the data out. There are several free tools available on the internet, such as WinMDI[145], FlowExplorer[146], and Facsimile[147].

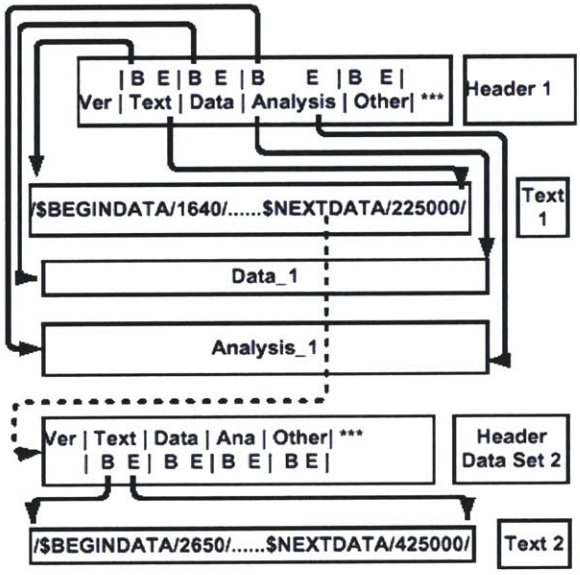


Figure 24 Flow Cytometry Standard (FCS) format

D. FACS Data Analysis

There are five types of data displays used in FACS data analysis, which are the frequency histogram, dot, density, or contour plot, and 3D display. I quote the introductions from G.W Osborne[148], 2000 here (the metadata are provided by me):

1) The simplest way of displaying flow cytometry data is the frequency histogram. Frequency histograms display relative fluorescence or scattered light signals plotted against the number of events.

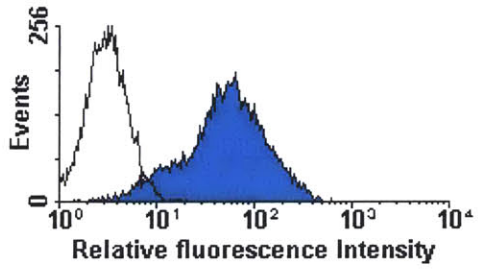


Figure 25 FACS frequency histogram (Ref: http://jcsmr.anu.edu.au/facslab/analysis.html)

The metadata associated with this kind of plot is usually like:

```
File: 5LMEP004 Sample:
Date: 12/13/95 Parameters: 4
Total Events 10000 Gated Events 10000  100.00%
System: Log Parameter Means: Geometric
Param name     M  Low,High Events %Total  %Gated   GMean     CV   Peak,Value
FSC-Heigh      0    0, 255  10000 100.00  100.00    74.09  62.27     204,74
SSC-Heigh      0    0, 255  10000 100.00  100.00    48.54  95.99     303,27
FL1-Heigh      0    0, 255  10000 100.00  100.00   295.06  38.85     210,9646.62
FL2-Heigh      0    0, 255  10000 100.00  100.00    57.37  79.70     849,1
```

2) To see the relative levels of other parameters which were collected at the same time, one needs to use one of the forms of bivariate displays namely dot, density or contour plots. In these types of displays, one parameter is plotted against another in an X versus Y axis display.

Dot plot is also known as bivariate display, "scattergram" or bitmap. This type of display plots one dot or point on the display related to the amount of parameter x and y for each cell which passed through the instrument.

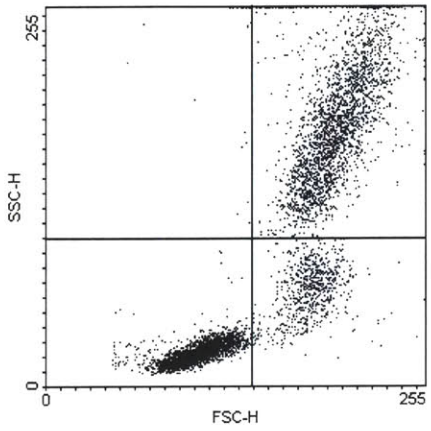


Figure 26 A dot plot of FACS

The metadata associated with dot plot is like:

```
File: 5LMEP004 Sample:
Date: 12/13/95 Parameters: 4
Total Events 10000 Gated Events 10000  100.00%
System: Log Parameter Means: Geometric
Quad Stats
FL2-Height(Log) vs FL1-Height(Log)
Quadrant x,y: 80,80
 Quad  X-Mean Y-Mean Events %Total  %Gated
1 UL    3.6  287.8   3467  34.67   34.67
2 UR 1005.7 1073.0   5036  50.36   50.36
3 LL    2.2    4.0   1468  14.68   14.68
4 LR   31.2    8.4     29   0.29    0.29
```

It lists the mean or median values of each quadrant.

3) Density plots simulate a three dimensional display of events with the "third" parameter being the number of events.

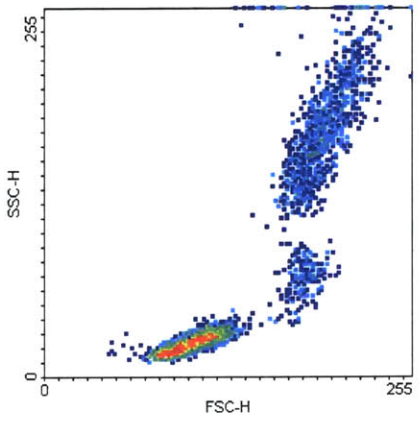


Figure 27 A density plot of FACS

4) Contour plots are another two dimensional display of relative X and Y amounts of two parameters, with contour lines being drawn to form x and y co-ordinates which have similar numbers of cells.

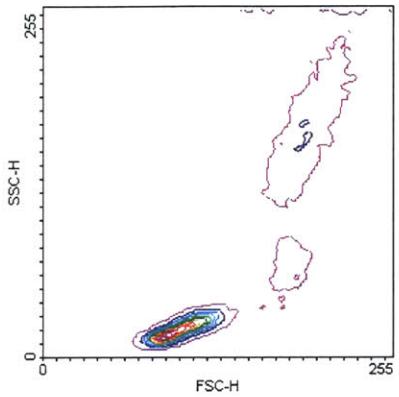


Figure 28 A contour plot of FACS

5) The metadata associated with density plots and contour plots are the same as dot plots. Since density plots and contour plots simulate a three dimensional display, sometimes we can use a 3D plot instead.

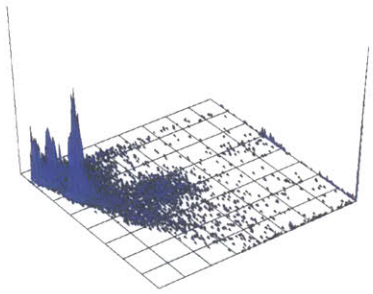


Figure 29 A 3D plot of FACS

E. The Information Object Definition of FACS

Before we define the information objects to describe the event and data of FACS experiments, let us review two good references first. One is the U.S.-Canadian consensus recommendations on the immunophenotypic analysis of hemotologic neoplasia by Flow Cytometry (Raul C. Braylan, et al)[149]; another one is CytometryML (Robert Leif, et al, 2003)[150].

According to the U.S.-Canadian consensus recommendations on the immunophenotypic analysis of hemotologic neoplasia by Flow Cytometry (Raul C. Braylan, et al), an experimental report of flow cytometry should include the information about patient, sample, sample preparation/staining, cell analysis, data analysis, interpretation, and so on, as shown in the below table.

Table 3 Summary of recommendations for reporting results of FCM analysis of hematologic neoplasia (Ref: Raul C. Braylan, et al, 1997)

Information	Always Report	Report if relevant
Patient	X	
Name/Id #	X	
Date of birth	X	
Sex	X	
Referring physician name(s)/phone	X	
Referring institution	X	
History/relevant clinical information and diagnoses	X	
Reason for FCM request	X	
Previous relevant therapy	X	
Previous FCM studies		X
Other lab results (WBC, differential count)		X
Sample		
Sample source/type	X	
Sample description		X
Sample date/time collected	X	
Sample date received	X	
Other materials received		X
Other procedures on original sample (imprints, smears, freezing, genetics, fixation, etc.)		X
Sample saved/stored		X
Specimen number	X	
Sample preparation/staining		
Cell suspension preparation method (RBC lysis, Ficoll-Hypaque)		X
Cell suspension preparation date/time		X
Cell yield		X
Cell viability		X
Microscopic control (cytospins)		X
Other tests on cell suspension (DNA content, cytochemistry, genetics, other)		X
Sample date/time stained		X
Nonviable cell staining		X
Cells saved/stored		X
Antibodies used (CDs)	X	
Antibodies used (trade name)		X
Fluorochrome combination used (surface and/or cytoplasmic)		X
Cell analysis		
Date/time FCM analysis		X
Data analysis		
Qualitative description of light scatter and/or immunophenotypic features of cells of interest	X	
Fluorescence intensity for relevant markers	X	
Relative counts in PB in special circumstances		X
% of abnormal cells relative to a defined population		X
Kappa:Lambda ratio		X
CD4:CD8 ratio		X
Morphologic description of cell suspension		X
Other pertinent tests on sample: microscopy, cytochemistry, immunohistochemistry, DNA content, genetics, etc.		X
Interpretation		
If no abnormal population is identified, a description of the normal populations present is provided. If an abnormal population is detected, its phenotype is described and a differential diagnosis is provided. If additional relevant clinical and/or laboratory data are available, a more definite diagnosis is included	X	
Additional elements		
Representative histograms/plots		X
Recommendations for additional studies		X
Cosignature by professional with proper expertise		X
Documentation of discussion with referring physician(s) or verbal reporting (date/time)		X
Selected references		X
Consultations		X
Date/time of final report		X
	X	

CytometryML, an cytometry XML, is developed by Robert C. Leif, Suzanne B. Leif, et al, from XML_Med, a Division of Newport Instruments, to describe the data produced by flow, laser scanning cytometry, and microscopic image cytometry. As is shown in Figure 30, CytometryML is composed of data types from multiple standards. The various parts of the Cytometry schema import data types from DICOM (the Digital Imaging and Communications in Medicine) including types from DICOM foreign Coding schemes, such as LOINC, SNOMED, and UCUM. They claimed that each of these data type sources and others such as MathML and FCS were or could be expressed as an XML schema. The three fifths of the information described in CytometryML are about the settings of hardware, such as detectors, amplifiers, lights, excitation filters, etc., as shown in Figure 31. CytometryML mapped the FCS list-mode to the DICOM Waveform Information Object, and extended the existing mapping of FCS to DICOM standard. It created a collection of XML schemas to express the DICOM analytical cytology text based data-types except for large binary objects.

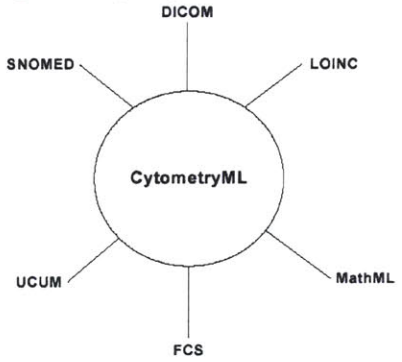


Figure 30 CytometryML schema organization (Ref: Robert C. Leif, et al, 2003).

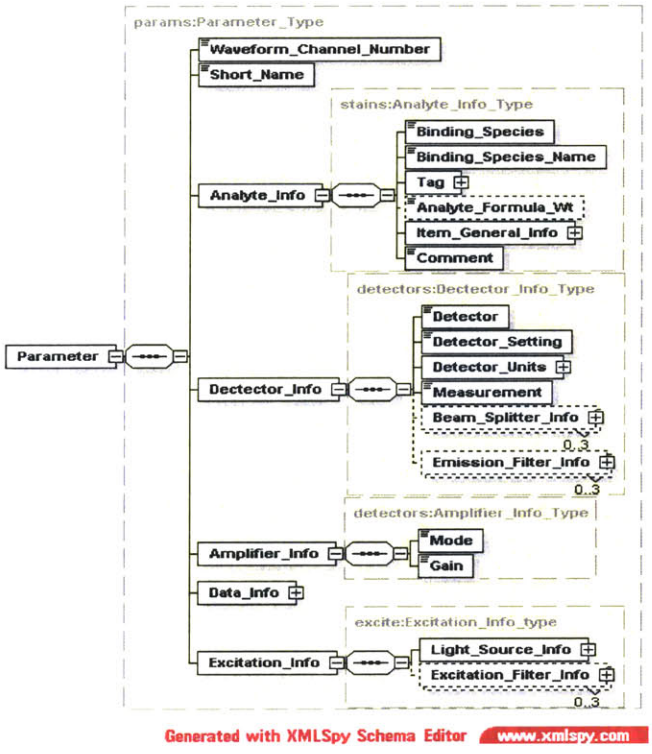


Figure 31 A portion of CytometryML. The view is generated with XMLSpy schema editor

CytometryML and the recommendations are two good references for us to develop the information object definitions of FACS experiments. And along with the descriptions in the previous paragraphs, we can summarize the information objects of FACS experiments as described in the following workflow:

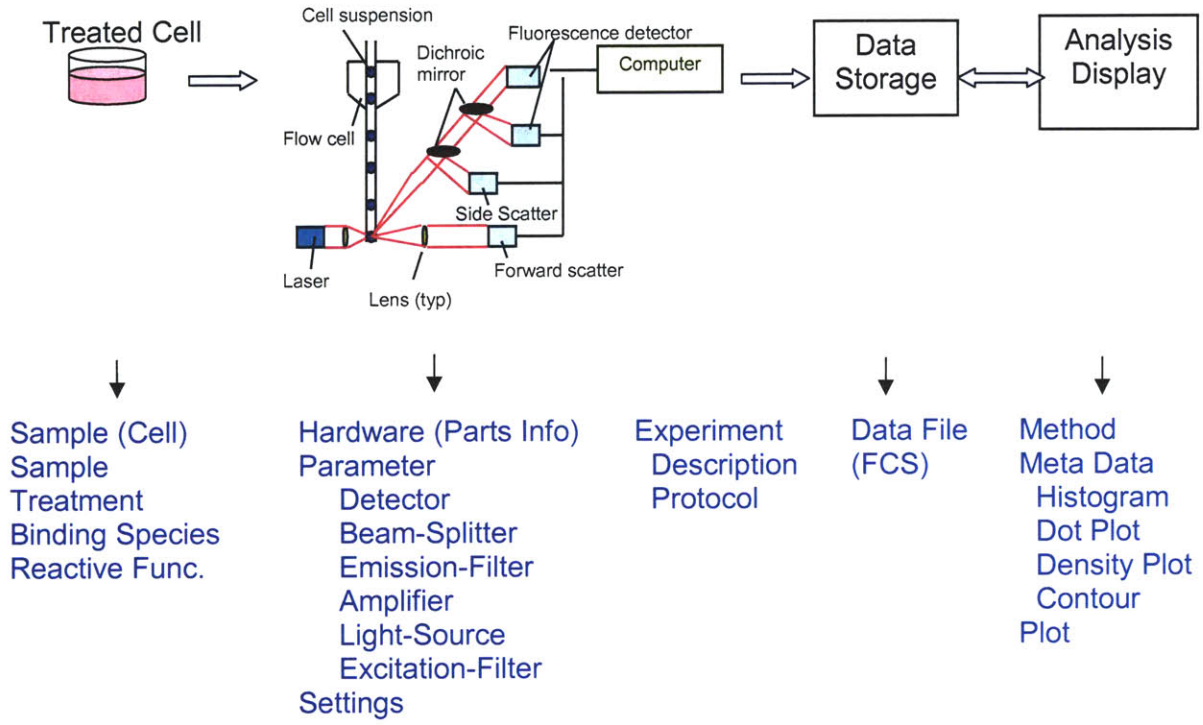


Figure 32 The workflow of FACS experiments (The instrument diagram[151] is from http://dpalm.med.uth.tmc.edu/faculty/bios/nguyen/DXPowerPoint/sld004.htm)

In order to characterize the experimental process, we need to describe: sample information, sample treatment process; hardware information and settings, such as detector, amplifier and light source; experiment description and protocol; raw data files which are in the FCS format; data analysis method; metadata generated from analyses, such as plots, etc. All of these information entities can be divided into the five packages as same as we did in the Western blot discussion, which are project (i.e. study plan), sample, experiment, high level analysis, and administration package. Using the five package's design idea, the information object definition of FACS is shown in Figure 33. Note that many objects are reused from the Western blots IOD. The only differences are the portions of the raw data and preprocessed data in the experiment package. The ontology proposed by CytometryML has been used as the basis of the object definition of the raw data portion. The object-oriented XML schema of CytometryML has been converted to the object-relational database schema of our design and the objects have been reorganized.

As a summary, this section proved that the structure of the five package design is suitable not only for Western blot but also flow cytometry experiments. The IOD of flow

cytometry can completely characterize the results of the experiments. It covers the information about:

- project (description, contact info, ontology, reference, hypothesis, report);
- sample (ideal sample, sample set, measured sample);
- experiment (protocol (hardware, data analysis method, and unit), description, and sample treatment, raw data (FCS file, trigger, analyte info, detector info, excitation info, and amplifier info), and preprocessed data (dot plot, histogram and their metadata));
- high level analysis, and administration.

The above two sections 5.1 and 5.2 have given out a very detail discussion about what are the experimental methods, example experiments, which information objects are involved in the experiments, and the information object definition to characterize the event and data of the experiments. Because the design idea and process is same, discussions will be condensed in the following sections about microarray, mass spectrometry, and microscope images.

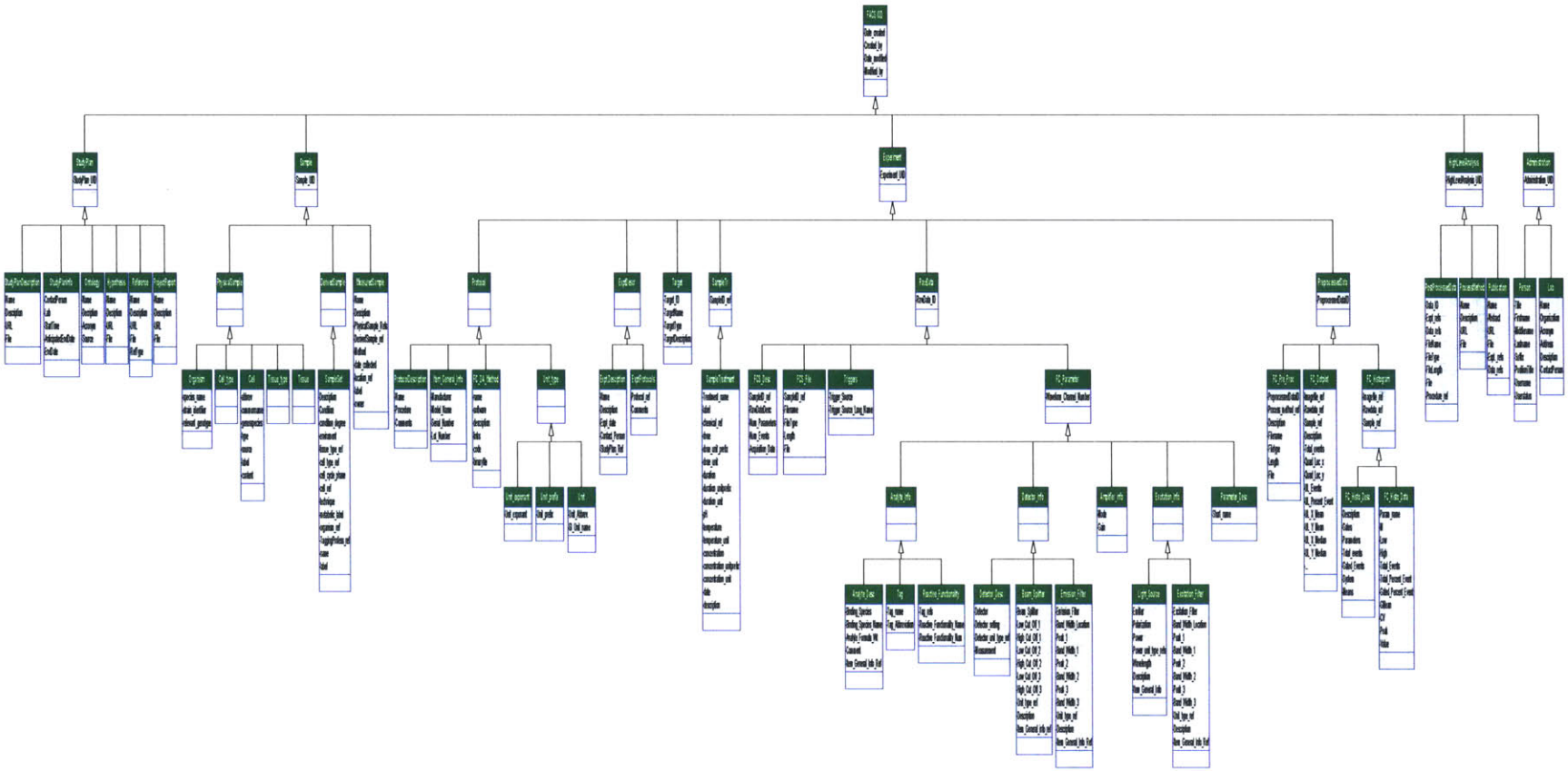


Figure 33 The information object definition of FACS experiments

5.3 Microarray

Microarray is the most fascinating technology in the millennium. A microarray is an array of DNA or protein samples that can be hybridized with probes to study patterns of gene expression. It is a tool for studying how large numbers of genes interact with each other and how a cell's regulatory networks control vast batteries of genes simultaneously. The array can be manufactured by a robot to precisely apply tiny droplets containing functional DNA to a substrate, like glass sides. There are two formats of DNA chip technology. Format 1 is Synteni/Incyte[152] technology; format 2 is Affymetrix[153] technology. In format 1, DNAs are first immobilized to a solid surface such as a polylysine and aminosilane coated glass slide or membrane; target DNAs (spotted on arrays) are PCR products. Single stranded cDNA probes are synthesized with fluorescent tags and then attached to the DNA spots. Hybridization is finally measured with a laser scanner and displayed as a pseudo color (so called two color array). In format 2, oligonucleotide probes are synthesized in situ on the chip or by conventional synthesis followed by on chip immobilization; semiconductor photolithography technology is used to synthesize oligos in situ on a glass substrate with a size of one cm square; proprietary combinatorial technology generates an ordered and high density array of a large number of oligo probes. cRNA targets are used to hybridize the array and stained with streptravidin-phycoerythrin conjugate. Hybridization is then measured with a scanner[154, 155].

Figure 34 illustrates a typical workflow of microarray experiments. First, prepare samples and allocate them into the wells of plates. Second, design arrays using an array design software. Third, use an arrayer / spotter to spot the samples from the plates onto a substrate according to the array design. The array slides are then manufactured. Fourth, apply probes to hybridize the slides. Fifth, use a scanner to measure the hybridization and display as an image. The final step is to acquire the numerical values of the spots on the image and do data analysis.

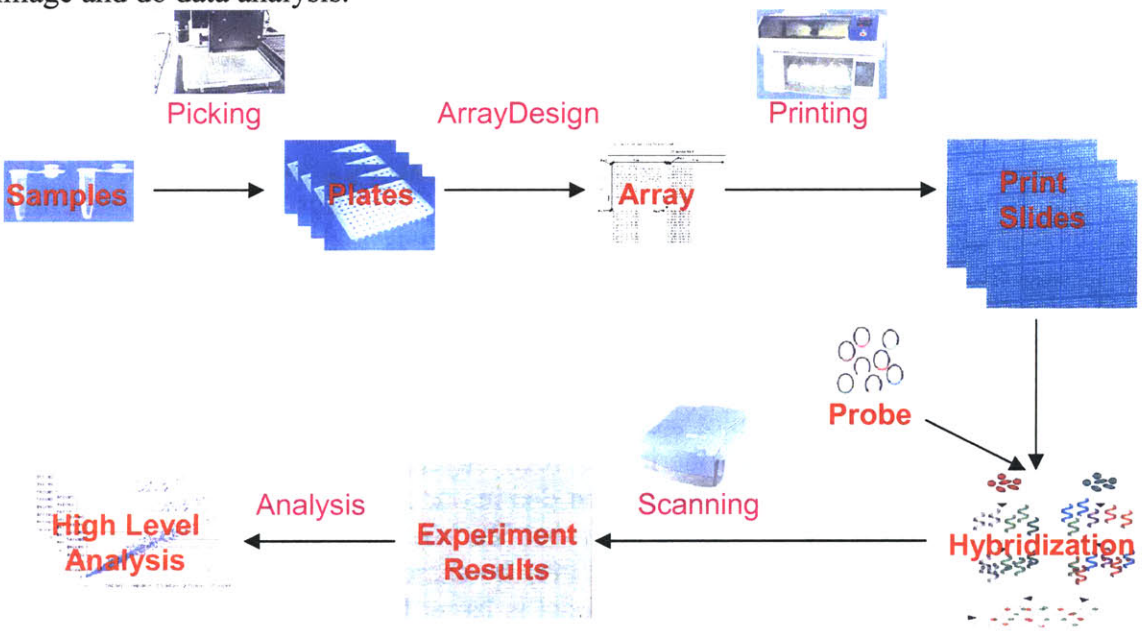


Figure 34 The workflow of microarray experiments

As we introduced in the Chapter 2, there are several initiatives particularly devoted to the semantic standardization issues in microarray experiments, and several microarray databases existing. Figure 35 shows the MicroArray and GeneExpression Object Model (MAGE-OM)[156,157] developed by the MGED Group (Microarray Gene Expression Database Group). The model starts from an experiment entity; each experiment has five main entities: array, bioarray, bioarray data, high level analysis, and biosequence. There are several supporting tables in the design: protocol, biomaterial, bioevent, design element, measurement, and so on.

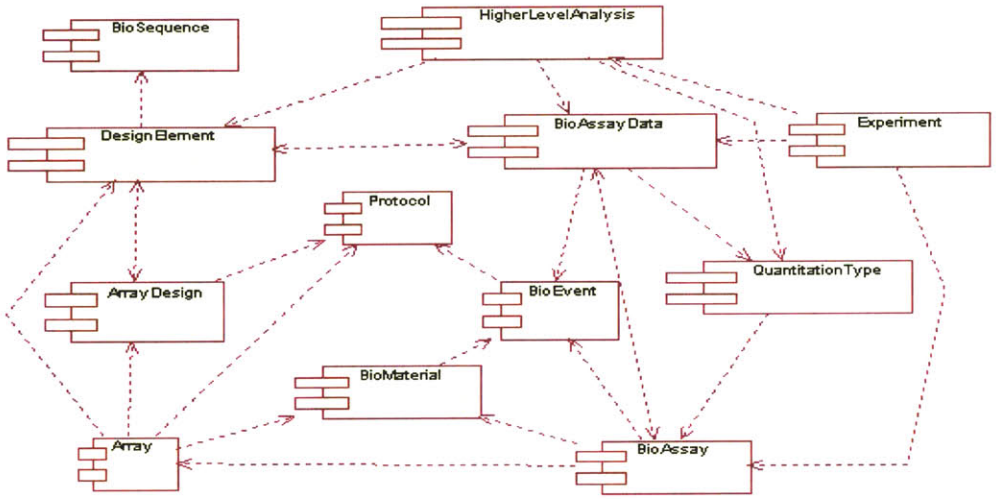


Figure 35 MicroArray and GeneExpression Object Model (MAGE-OM) developed by the MGED Group (Microarray Gene Expression Database Group) [158]

In our previous schema designed for gel electrophoresis and flow cytometry, we have the similar entities definition. Table 4 gives a way to incorporate MAGE object into our design (we call ExperiBase). The red color represents ExperiBase entities; the black color represents MAGE entities. There is about 90 percent overlap between our design for gel electrophoresis data and MGED's design for array data. This indicates that, indeed, there are common features among the various biological experimental data. Therefore, we can develop an integrated schema and semantic entities to describe the data derived from various biological experimental methods. This effort will be very helpful to the biological community. Professor Dewey's group is taking an active role in the international standards group Interoperable Informatics Infrastructure Consortium (I3C) that is working to establish such standards across the biological community.

After the mapping of MAGE to our five package's design, a new object model of microarray is defined in Figure 36. We keep the top two levels of objects same as the ones in the IODs of Western blot and flow cytometry. We reorganized MAGE objects and incorporated them into the inheritance tree. Therefore, nearly a hundred percent of information described in MAGE can be recorded in our design. Since the structure of our object model is designed in such a way that the structure can support any kind of experimental method, and the information may be shared in different methods, such as sample, project, the scope of our object model is larger than MAGE; the applications of ours are wider than MAGE.

Table 4 Incorporate MAGE objects into ExperiBase

ExperiBase		MAGE
StudyPlan	Study Plan	
Sample	Physical Sample	Biomaterial, Compound, BioSource, BioSample, BioSequence
	Derived Sample	LabeledExtract
	Measured Sample	Array, BioAssay
Experiment	Protocol	Protocol, Hardware, Software, ArrayDesign, Measurement, Unit, QuantitationType
	Sample Treatment	Bioevent
	Target	BioAssaySequence
	Description	Experiment, Description, DesignElement
	Raw Data	RawBioAssayData, PhysicalBioAssay
	Pre-Processed Data	BioAssayData
HighLevelAnalysis	High Level Analysis	HighLevelAnalysis, BioAssayDataCluster
Administration	Personnel	Contact (Person, Organization)
	Audit and Security	Audit and Security

About the Stanford Microarray Database discussed in the Chapter 2, we also converted their relational database schema to our five packages - object-relational database schema, as shown in Figure 37. Let us look at the sample package, "standforSeq, SeqType, Organism, and Patient" are interpreted as physical sample (or ideal sample); the plate and platesample are treated as derived sample; and the print and spotlist are treated as measured sample. This new interpretation of the above entities keeps the exact relationship between each other in the real world. It follows the path of the workflow shown in Figure 34, which the first step is the samples, and the second is the plates (derived sample), and then is the print slides (measured sample). This new schema has been tested by thousands set of microarray data from the Stanford Microarray Database. And also, since the schema shares sample information with other experimental methods, we are able to query all experiments done on the same samples (such as cells).

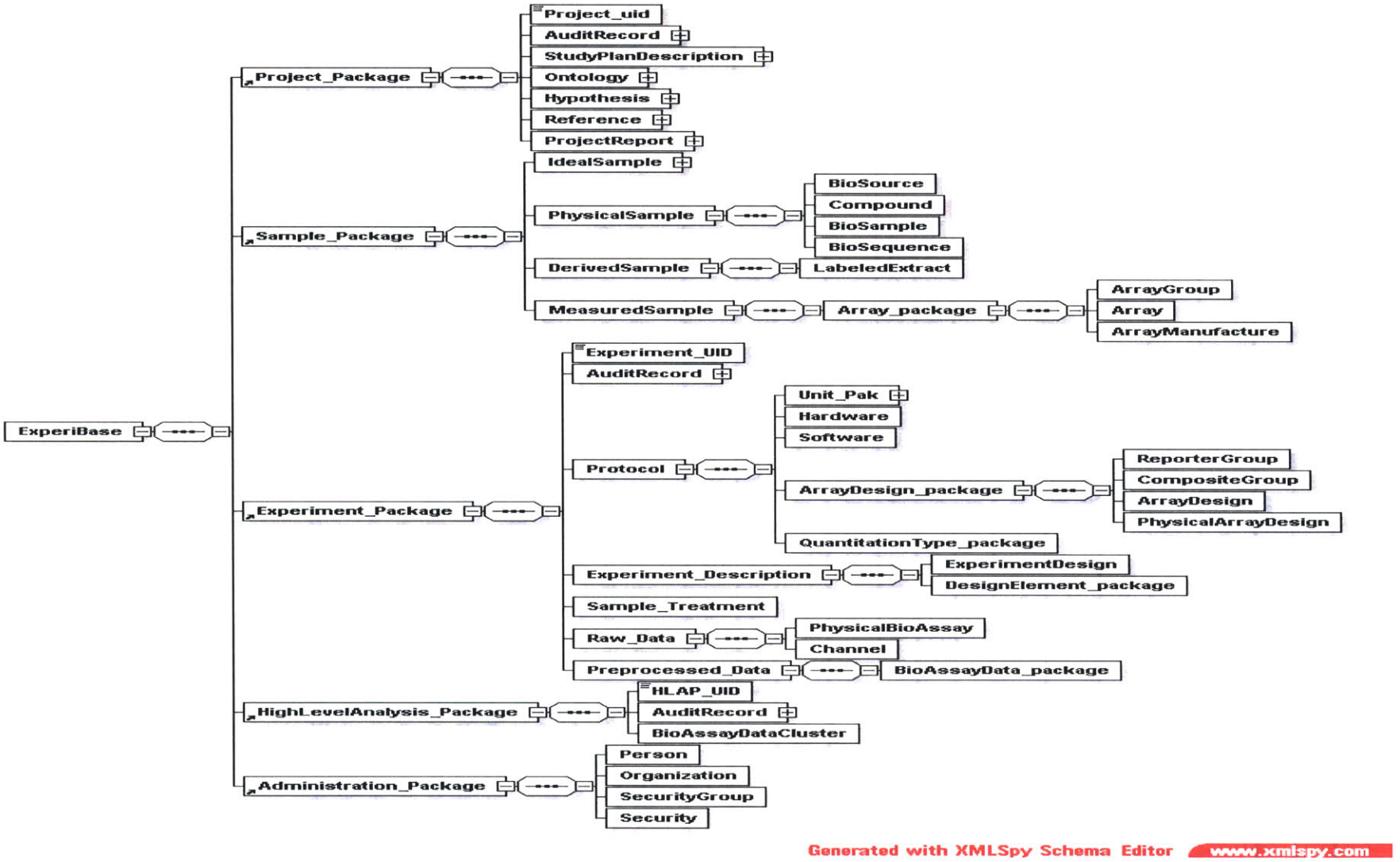


Figure 36 The information object definition of microarray, based on MAGE-ML

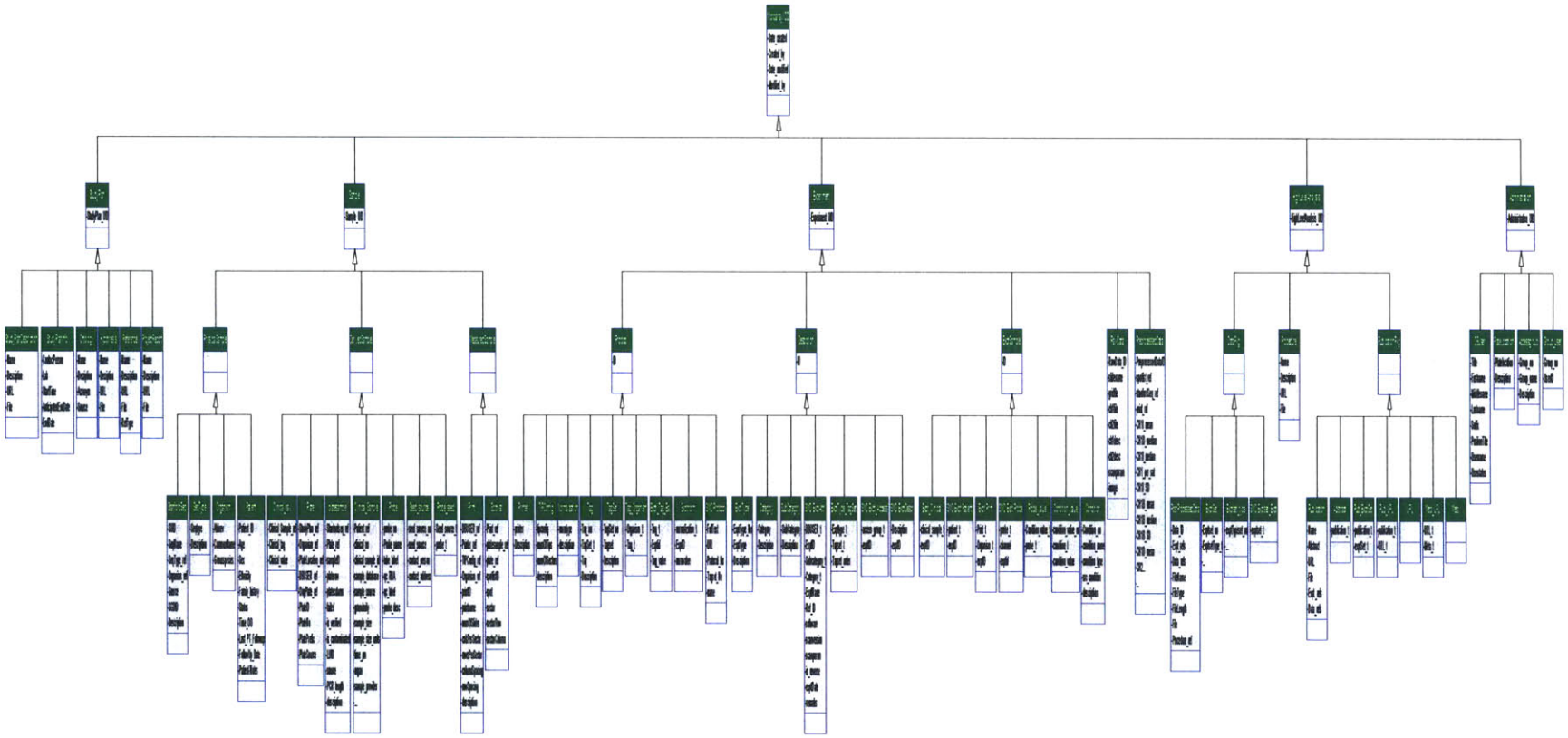


Figure 37 The information object definition of microarray, based on the Standard Microarray Database

5.4 Mass Spectrometry

Mass spectrometry is a method for identifying molecules based on the detection of the mass-to-charge ratio ("molecular fingerprint") of ions generated from the molecules by vaporization and electron bombardment. Deflection of the ions through a magnetic field results in a characteristic pattern used to and the ions are separated by their mass to charge ratio. The enough energy into a target molecule causes the molecule's ionization and disintegration[159]. This analytical technique can identify the chemical constitution of a substance – called also mass spectroscopy[160].

In the PEDRo paper (Chris Taylor et al), they also developed a schema for mass spectrometry experiment, shown in the below figure.

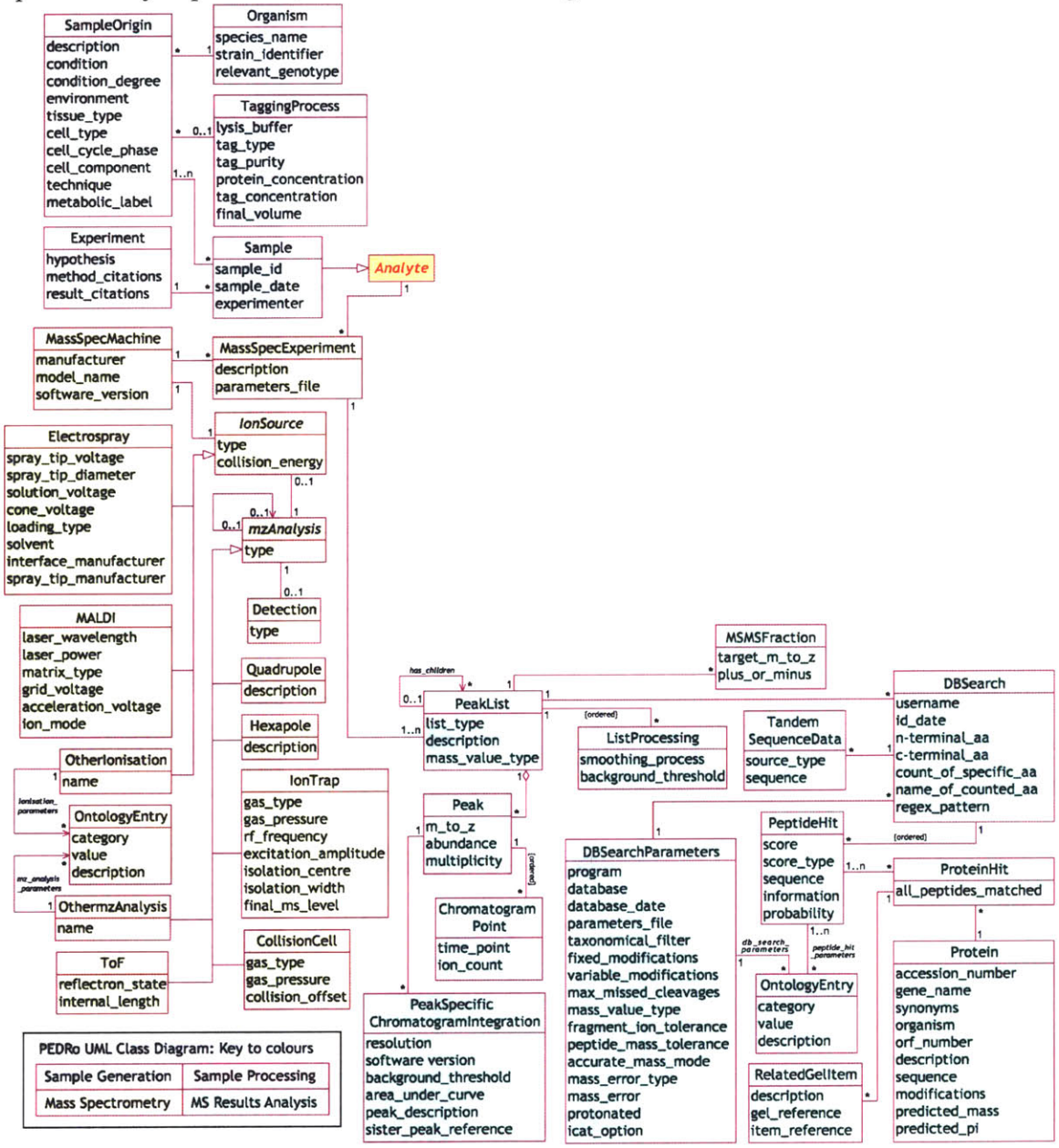


Figure 38 The schema of mass spectrometry in the PEDRo paper

I converted the PEDRo schema to our five package design, as shown in Figure 39. Basically, I incorporated the portion of the yellow color of the PEDRo schema into the protocol subtree of our new design; the portion of the green color is reorganized and put in the subtrees of raw data and preprocessed data. The new design is a tree structure. It has all of benefits which we have discussed in the previous sections (5.1, 5.2, and 5.3). We reused the five package design idea; the top two level entities are kept the same as before; the entire entities of project and sample package, and some entities of the experiment package are the same as the ones in the IODs of Western blot and FACS. The reference of the table "experiment_protocol" is pointing to the entire protocol data type, which means any child of protocol is allowed. This is the same mechanism as the "sample reference" in the sample treatment table, which is a variant object reference, can refer to any type in the sample package. We know in the OO notion, an "object" means a software packet containing a collection of related data and procedures for operating on that data. Our new design is using the same concept. For example, the ionization type contains electro-spray, MALDI (Matrix Assisted Laser Desorption Ionization), and other ionizations. The tree structure is easier understood and better organized than the PEDRo schema.

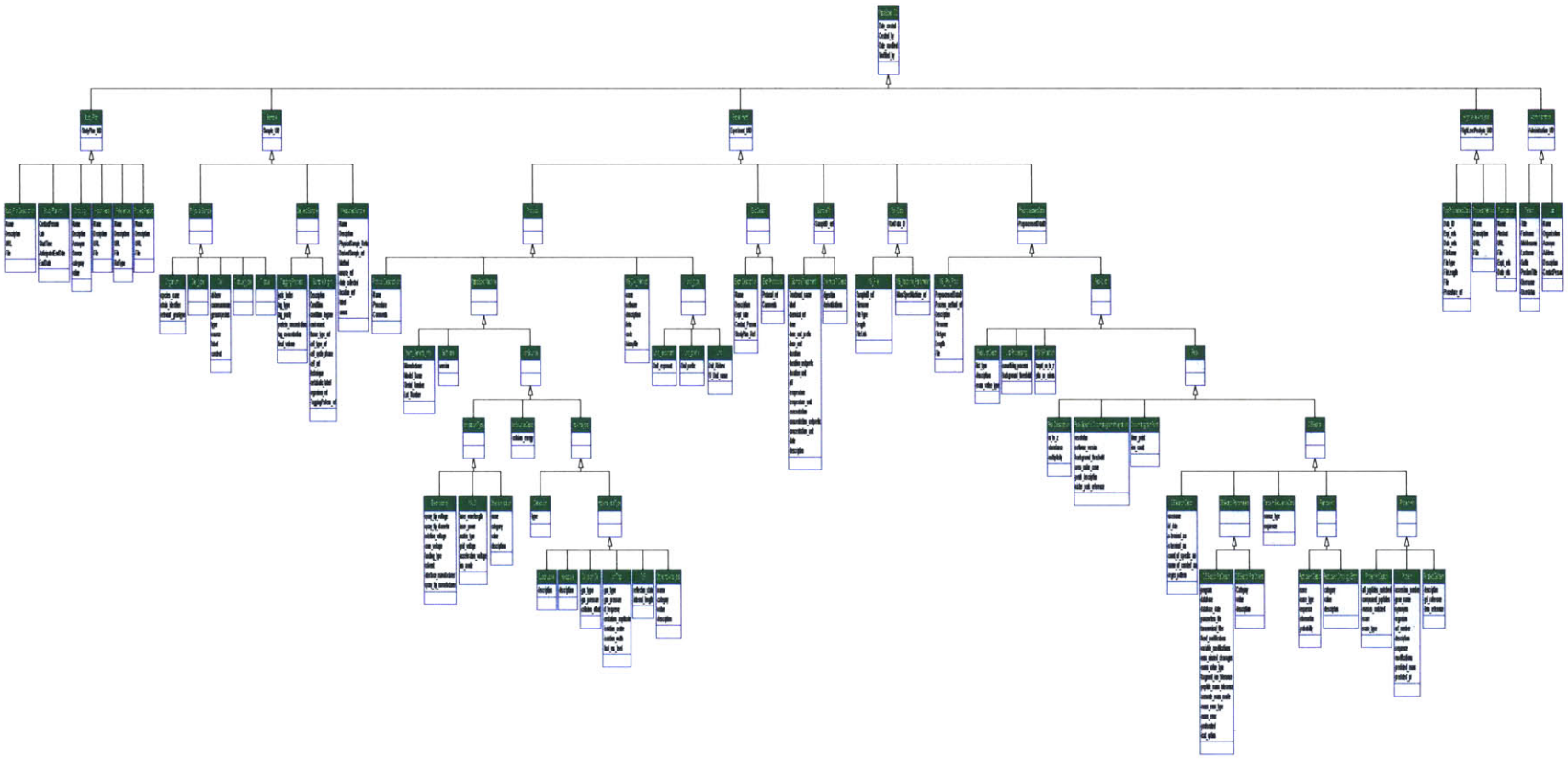


Figure 39 The information object definition of mass spectrometry experiments

5.5 Microscope Images

Microscope images of live and fixed cells have long been an important tool in cell biology. As the same approach as we did on other experimental methods, we use OME (Open Microscopy Environment) as the basis of the information object definition for this experimental method. OME has been introduced in the Chapter 2. The below diagram is the XML schema of OME. From this schema, we cannot find the sample information, such as cells (Table 5). But other entities are valuable for us. Using the same procedure as before, I have incorporated the objects of OME into our five package design, which is shown in Figure 47.

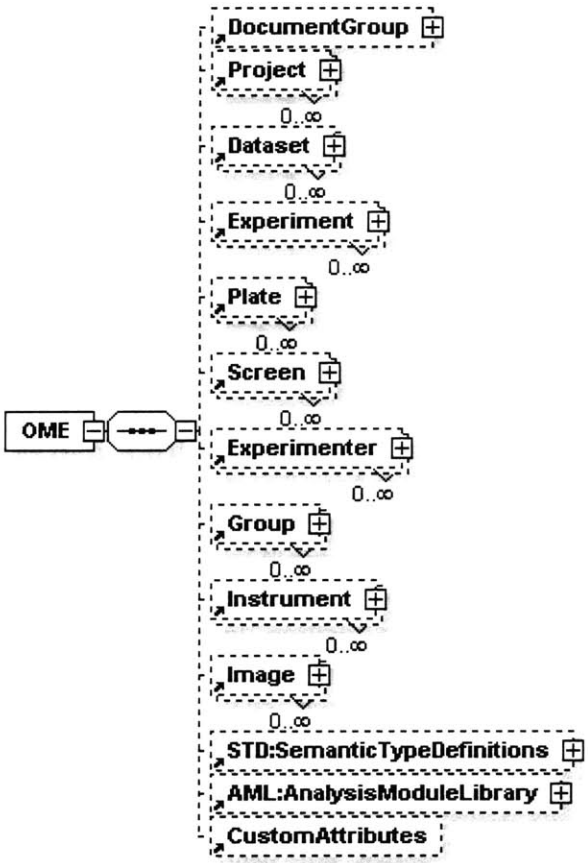


Figure 40 The XML schema of OME (Ref: http://www.openmicroscopy.org/docs.html)

Our new design reuses the entities of project, ideal sample, derived sample, administration, and a portion of experiment package from the IODs of Western blot. The entities about instrument parameters, raw data, and high level analysis are converted from OME. We extended the OME schema by adding the information about sample, treatment, and protocol into the schema. The new schema is nearly completed.

Table 5 Incorporate OME objects into ExperiBase

ExperiBase		OME
StudyPlan	StudyPlan Description	Project
	Reference	DocumentGroup
Sample	Physical Sample	
	Derived Sample	
	Measured Sample	Plate, Screen
Experiment	Protocol	Instrument, Microscope, LightSource, Detector, Objective, Filter, OTF
	Sample Treatment	PlateRef
	Target	
	Description	Experiment
	Raw Data	Image, ChannelInfo, DisplayOptions, Feature, StageLabel
	Pre-Processed Data	Pixels, Thumbnail
HighLevelAnalysis	High Level Analysis	Dataset, AnalysisModelue, Program
Administration	Personnel	Experimenter, Group
	Audit and Security	

Figure 47 shows that the schema is easily updated and extensible. For example, the schema now supports self-adjustment experiments, which we haven't discussed in the previous sections. We assume that an experiment (or an instrument, like microscope) can adjust itself to trace a feature (such as, Feature A) or decide which step the experiment should do next according to the current properties of Feature A (Figure 41). The properties of Feature A vary with time or position. In order to design a schema to support this kind of experiments, first, we need to design tables to record the information about the properties of Feature A at any time point, such as, the position, signal, envelope, mask, ratio, threshold, distance, delta, velocity, trajectory, extension, and so on (Figure 45). Second, we need to design information models to express the analysis modules, analysis chains, and analysis paths. In my design, each analysis module is recorded as XML format in the "MODULES" entity, and each module is associated with formal inputs and formal outputs which are in XML format (Figure 42). The "analysis chain links" records the information about a chain node from a module to another module. The "analysis paths" stores the nodes in a path of an analysis graph. (The design idea comes from OME). Then we can design the information entities for the analysis modules, chains, and paths like the diagram of Figure 46.

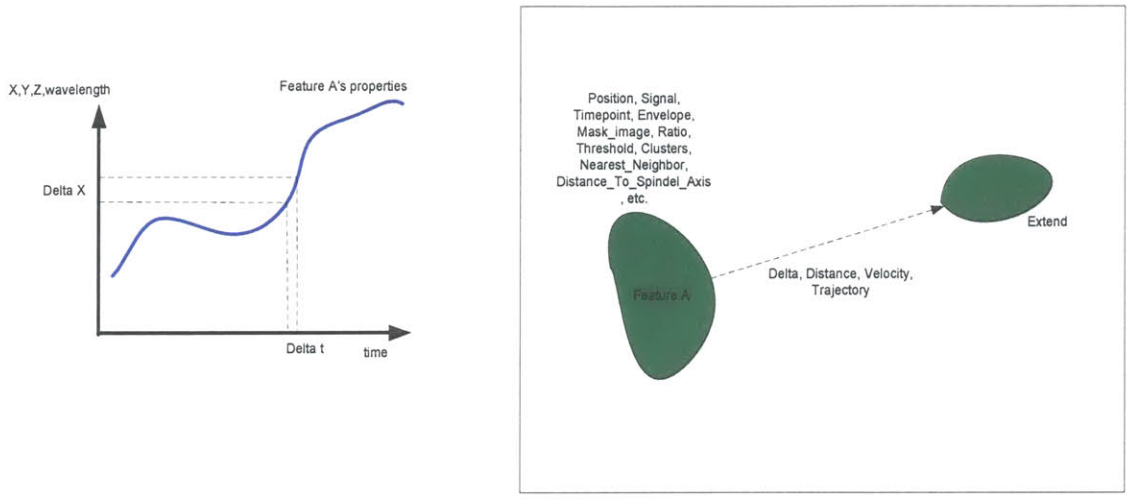


Figure 41 An example of self-adjustment experiments. The properties of Feature A vary with time or position. We assume that an experiment (or an instrument, like microscope) can adjust itself to trace Feature A or decide which step the experiment should do next according to the current properties of Feature A.

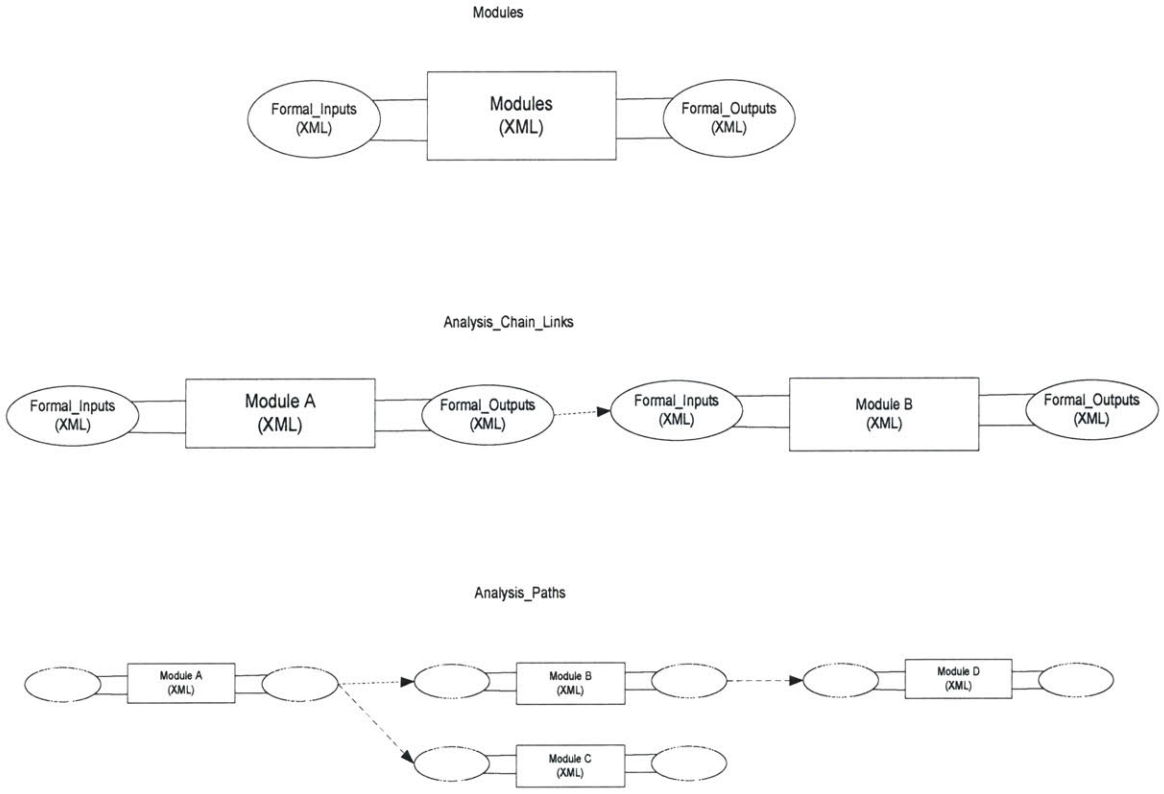


Figure 42 The information models to express analysis modules, analysis chain links, and analysis paths. (Ref: Open Microscopy Environment)

During the self-adjustment experiment, first, we load a module's XML description from the database into a computational engine; second, we feed the module

with the properties of Feature A at the current time point t1 and execute it; and then we use the outputs to decide what next step is (Figure 43). We can record the decisions step by step to form an actual analysis path (Figure 44) and record the properties at any time points as well.

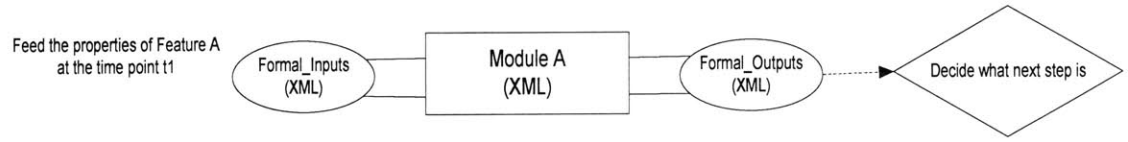


Figure 43 The procedure to decide what next step is in a self-adjustment experiment

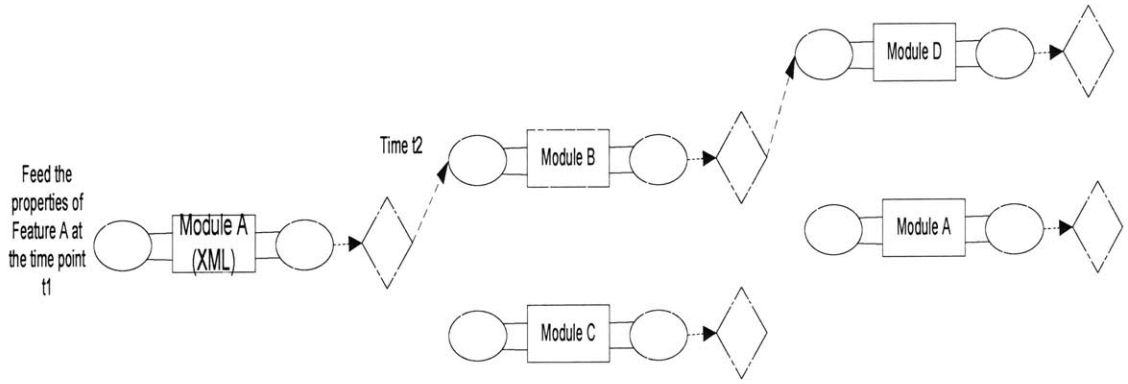


Figure 44 Using the actual decisions to form an analysis path of a self-adjustment experiment

After the above design for the self-adjustment experiments, we can incorporate the information entities into our five package design without any changes of the structure. Those entities are under the nodes of "analyses" and "features" in Figure 47. This proves that the five package design is extensible.

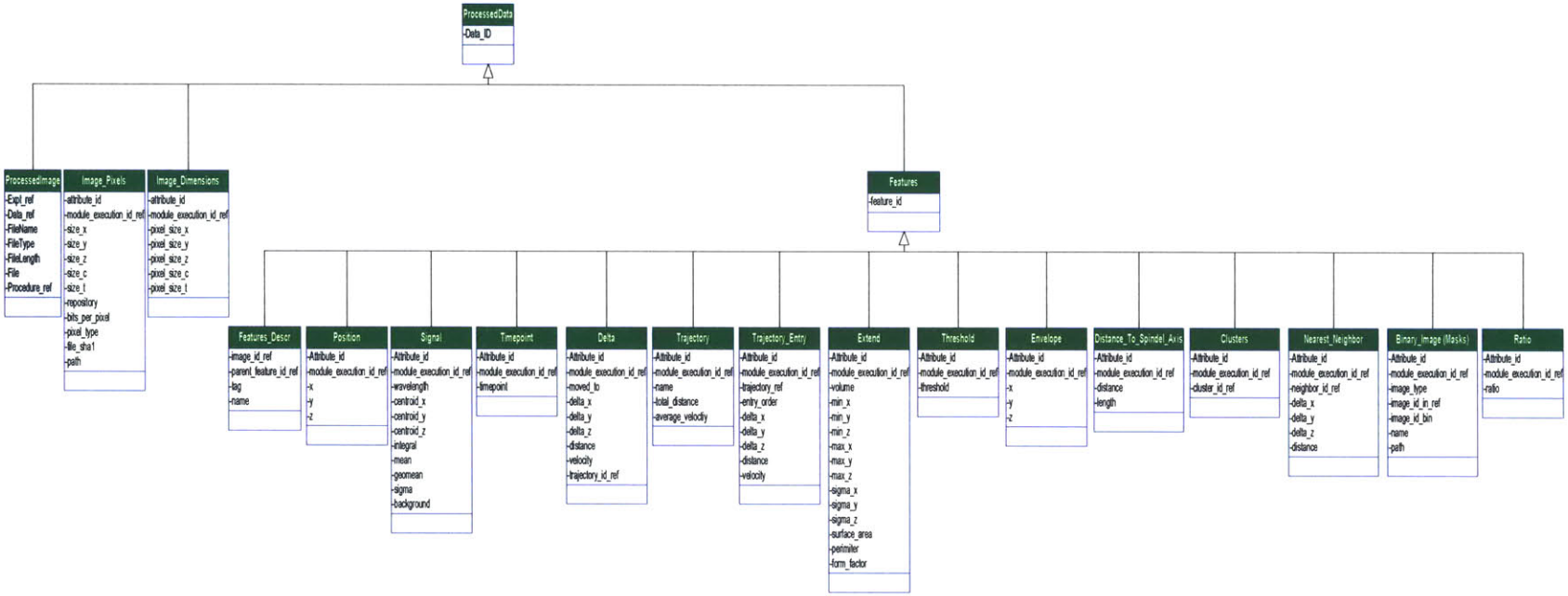


Figure 45 The information entities to record the information about the properties of features (converted from OME)

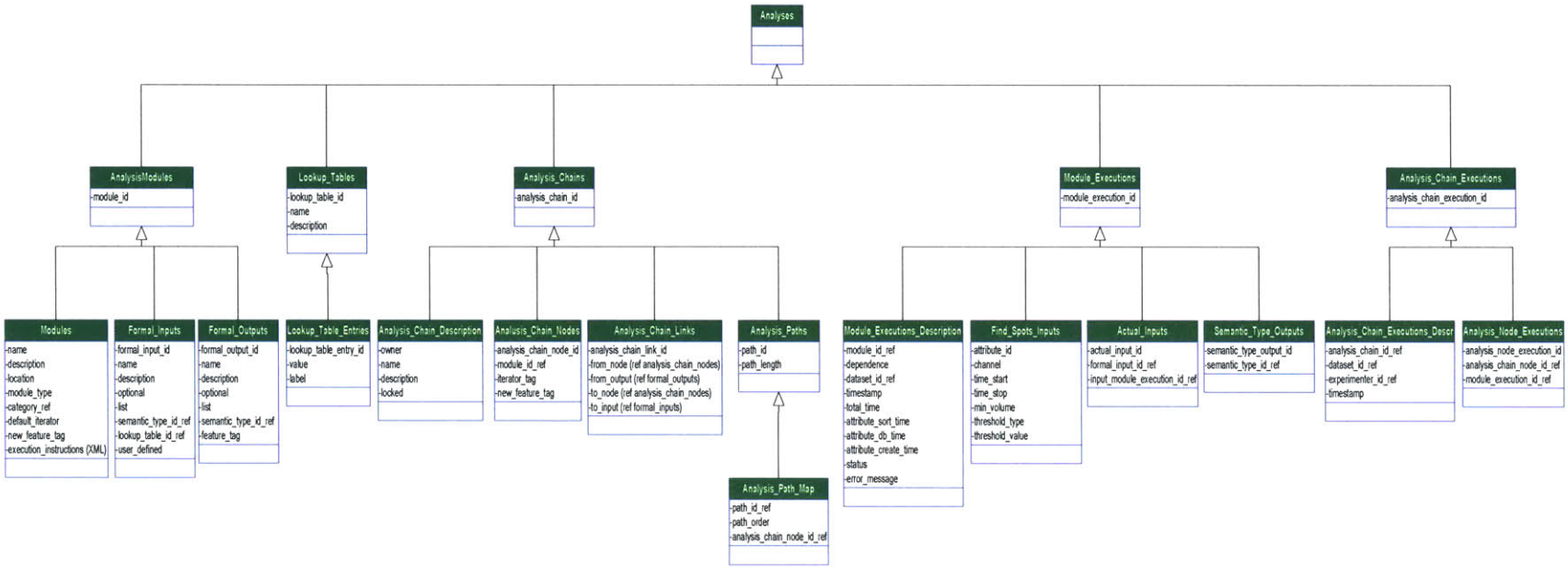


Figure 46 The information entities of analysis modules, chains, and paths (converted from OME)

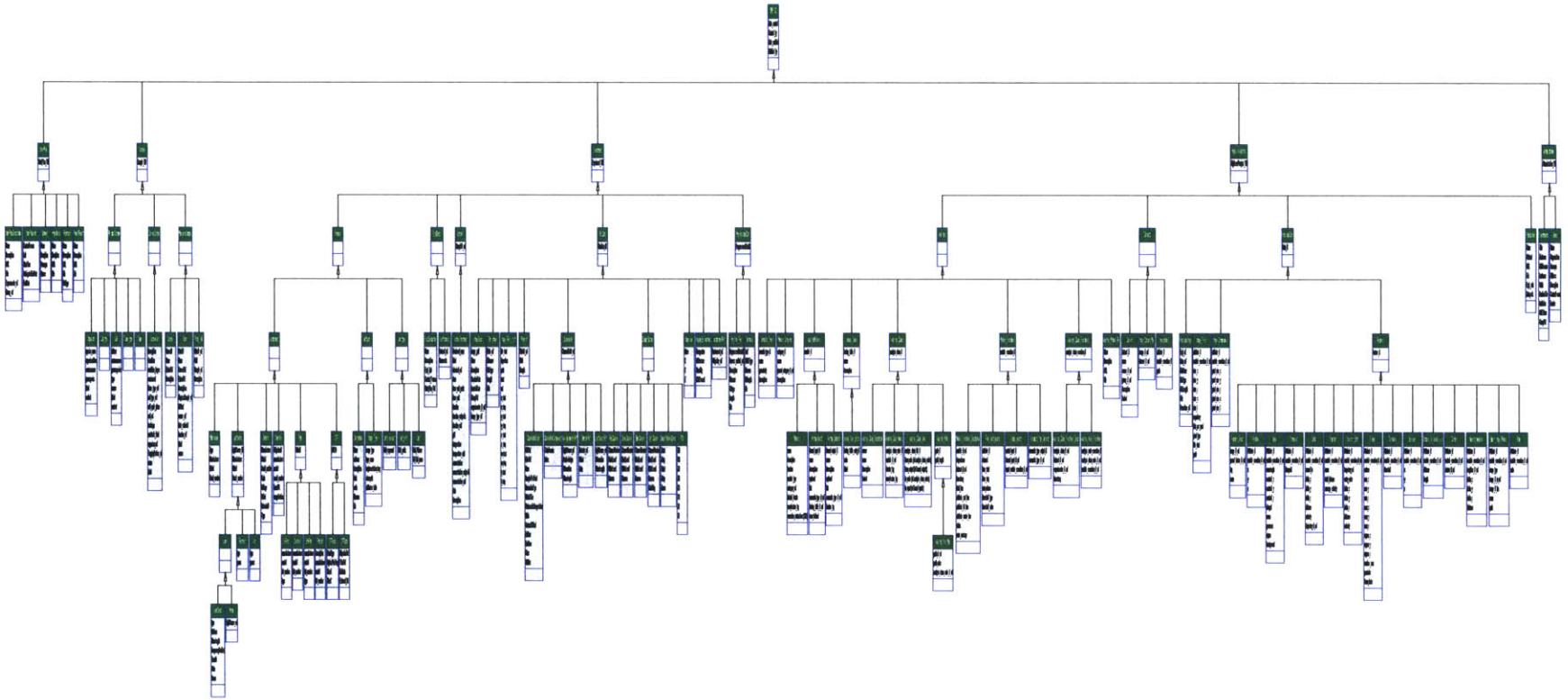


Figure 47 The information object definition of microscope images

5.6 Summary

This chapter has discussed the information object definitions for gel electrophoresis, flow cytometry, microarray, mass spectrometry, and microscope image. We have also introduced the existing standards which can be adapted or incorporated to our design. Because the design of the IODs for each method has used a single coherent set of information object definitions, a consistent data definition strategy has been proved that can handle those five modalities within the single coherent set of information object definitions. The entities are able to completely characterize the results of each experiment.

We have presented a five package design idea for biological experimental data, which are project, sample, experiment, high level analysis, and administration. The project package consists of a description of the project, contact info, ontology, reference, hypothesis, and project report. The sample package has three types, which are ideal sample (or physical sample), derived sample (can be sample set), and measured sample. The experiment package includes experimental protocol, description, sample treatment, raw data, and preprocessed data. The high level analysis records publications, or any advanced data analysis results which may use the raw data from multiple experimental methods. The administration package stores the information about person (user, experimenter), lab, security and any information about auditing events. The administration package and project package have their own fixed structure, which do not vary with experimental methods. The entities in the ideal sample domain are common to any biological experiments. These reflect the real situation in biological experiments. This kind of separation of information objects allows that: one study plan can have multiple samples, multiple experiments and multiple high level analyses; one sample can be used in several experiments; one set of experiment results can be analyzed in multiple high level analysis processes; an experiment can have multiple sample treatment steps, multiple stain steps, multiple results, and multiple data analyses; the auditing information, such as the date created / modified and contact person is recorded in every information entity. The IODs are object-oriented and their structures are trees. The consistent object inheritance allows us to be able to implement variant object references. Each node of the tree contains a collection of related data of an object and procedures for operating on that data. The IODs are easily updated.

The IODs of Western blot and 2D gel electrophoresis are currently very complete ontology. They were summarized from a lot of gel electrophoresis experiments done by several labs and persons; and the proposed ontology for 1D and 2D gel data by Chris Taylor et al, and the way describing LIMS' data by CDISC and DICOM, have been used as the basis of the definitions. The ontology is ready for international standardization consideration.

The ontologies proposed by CytometryML, MAGE, OME, and PEDRo have been used as the basis of the object definitions of flow cytometry, microarray, microscope images, and mass spectrometry, respectively. Their XML schemas or relational database schemas have been converted to the object-relational database schemas of our design and their information entities have been reorganized, incorporated, or even extended (such as OME). Therefore, nearly a hundred percent of information described in Cytometry, MAGE, OME, and PEDRo can be recorded in our design. Since the structure of our

object model is designed in such a way that the structure can support any kind of experimental method, and the information may be shared in different methods, (such as sample, project), the scope of our object model is larger than any above individual standard; the applications of ours are wider than them as well. The tree structure of our IODs is easier understood and better organized. The new interpretation and reorganization of information entities guarantees the exact relationships between each other in the real world. The grouping of information entities reflects the workflow of each experimental method. And also, since many information entities are shared with various experimental methods, we are able to query all of experiments which are related to a certain piece of information (such as a cell) even though the experimental methods are different. The new schema converted from the Stanford Microarray Database has been tested by thousands set of microarray data from the database.

Since common attributes are made truly identical between the different methods, this dramatically decreases the overhead of separate and distinct classes for each method, and reserves the uniqueness for attributes that are different between the methods. Thus, at least one higher level of integration is obtained.

Chapter 6. Generalization and Implementation

Because the design of the IODs which discussed in the previous chapter for each experimental method has used a single coherent set of information object definitions, the IODs can be integrated. This chapter will discuss the generalization of the IODs, and also introduce the system and database implementation (we call the system as ExperiBase). Since the implementation is actually time-consuming but not too much difficult, here I am only going to stress some important issues about the implementation.

6.1 Generalization

In the Chapter 5, we have summarized that the five packages (in Figure 16) can support any kind of biological experimental data, which are:

- project (description, contact info, ontology, reference, hypothesis, report);
- sample (ideal sample / physical sample, derived sample / sample set, measured sample);
- experiment (protocol (hardware, data analysis method, and unit), description, and sample treatment, raw data, and preprocessed data);
- high level analysis (post-processed data, data process method, publications);
- administration (person, lab),

Since common attributes are made truly identical between the different methods, we are able to generalize a consistent data definition strategy as shown in Figure 48. The blue color represents that the entities are common to all experimental methods; the red color represents variances. This strategy is extensible. Except the project package, administration package, and the ideal (physical) sample, each common node provides a sub-node for each experimental method. All relevant entities of the experimental method should be the children of the sub-node. If an experimental method has multiple implementations, such as SMD and BASE are two implementations of microarray, the node of the experimental method should have multiple sub-nodes, given that the entities of the implementations can not be easily merged. The project package, administration package, and the ideal sample sub-tree do not vary with experimental methods. They are fixed and common to all experimental methods. We can follow the design rules and keep adding any new experimental methods without any change of the structure; for example, in the below diagram (Figure 49), method X and Y are added into the raw data sub-tree.

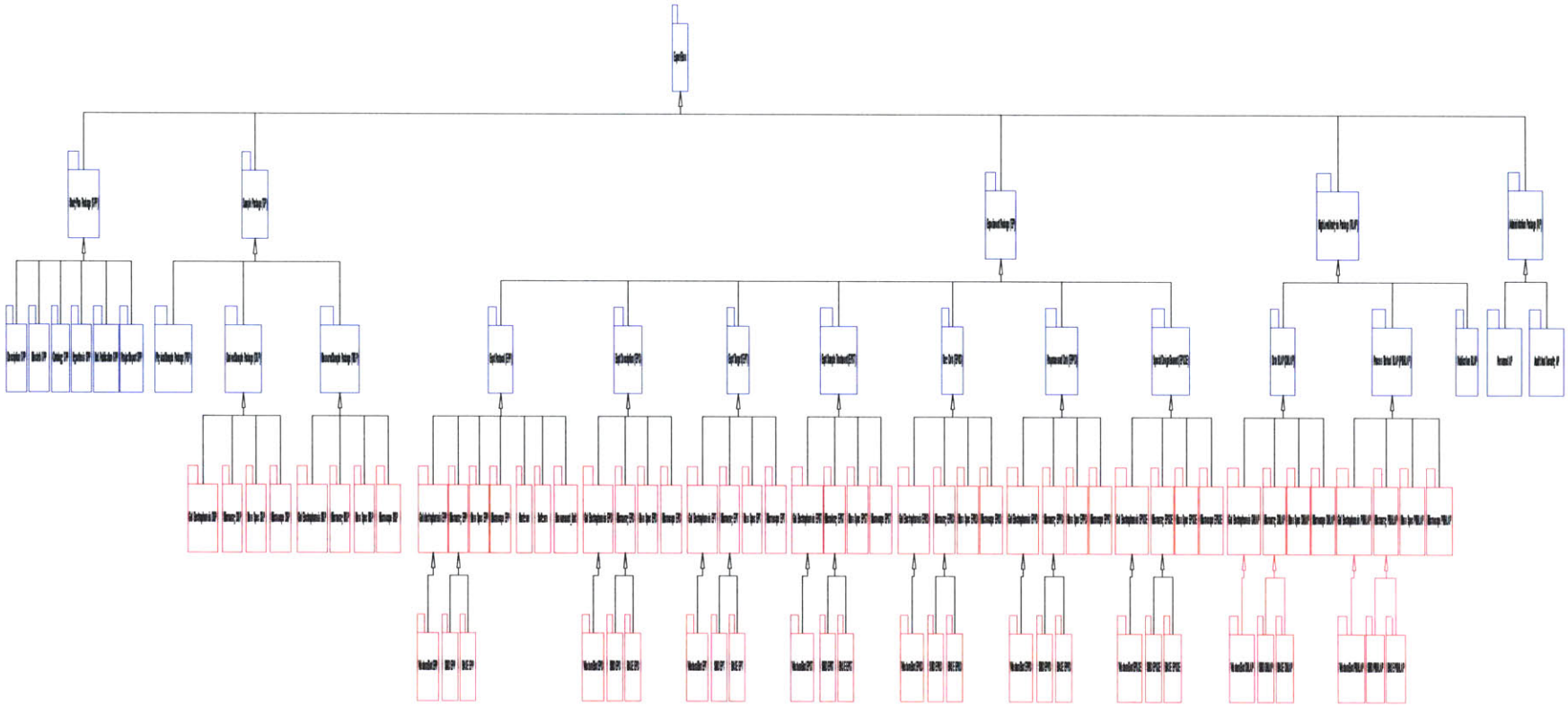


Figure 48 ExperiBase inheritance

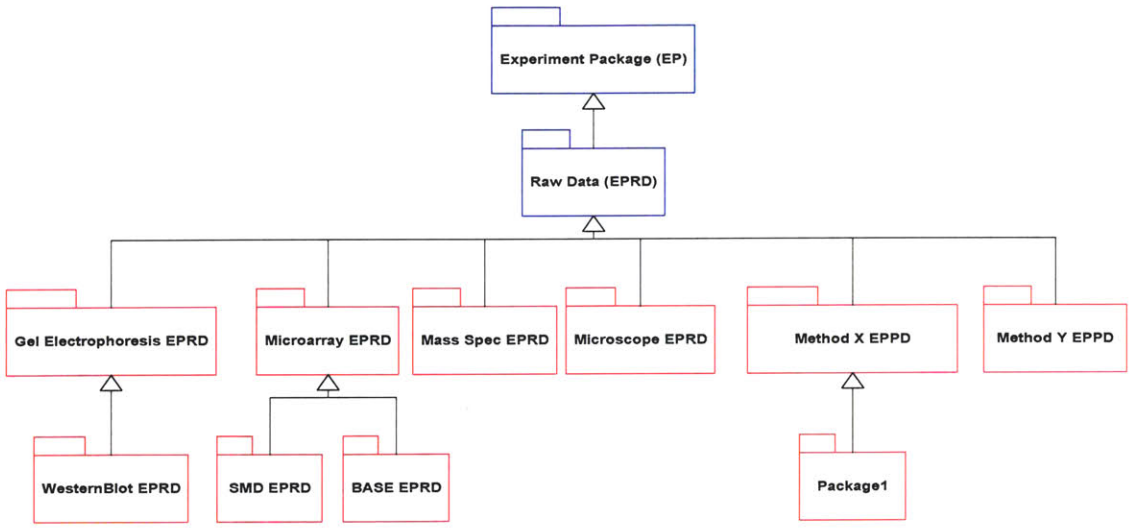
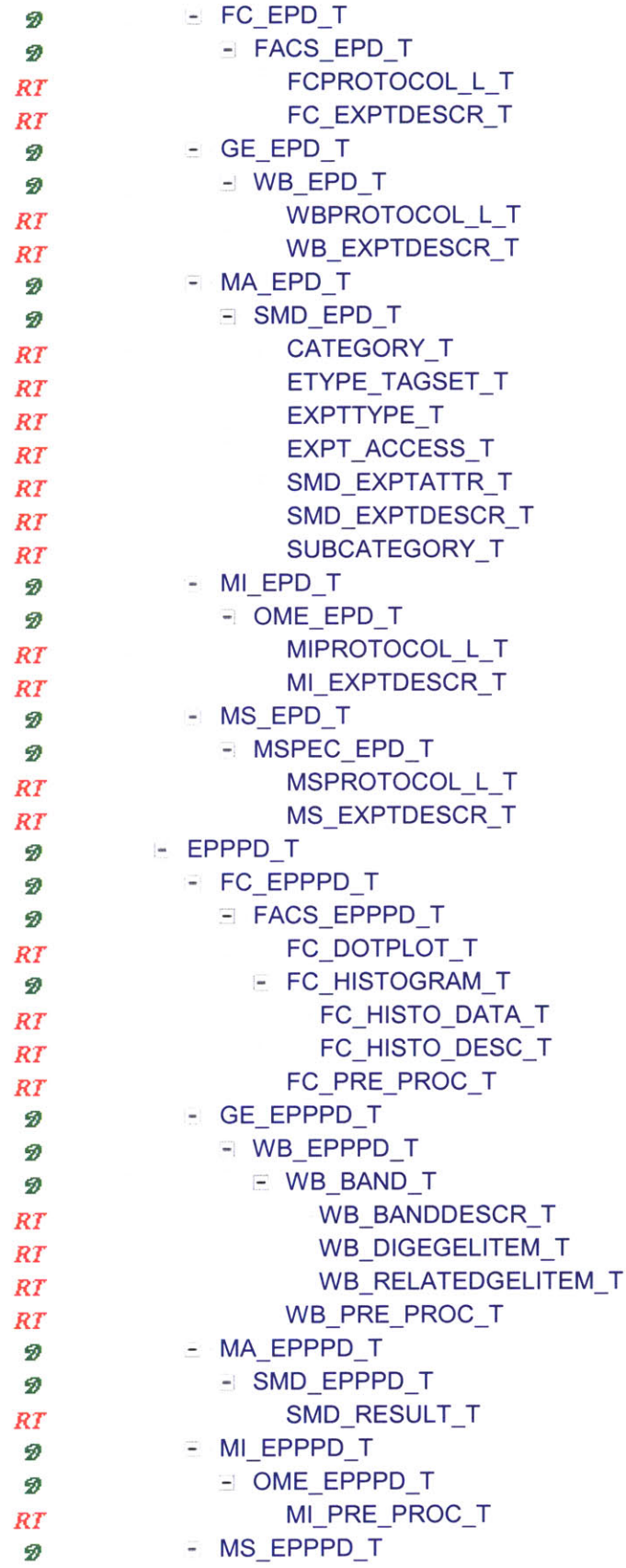
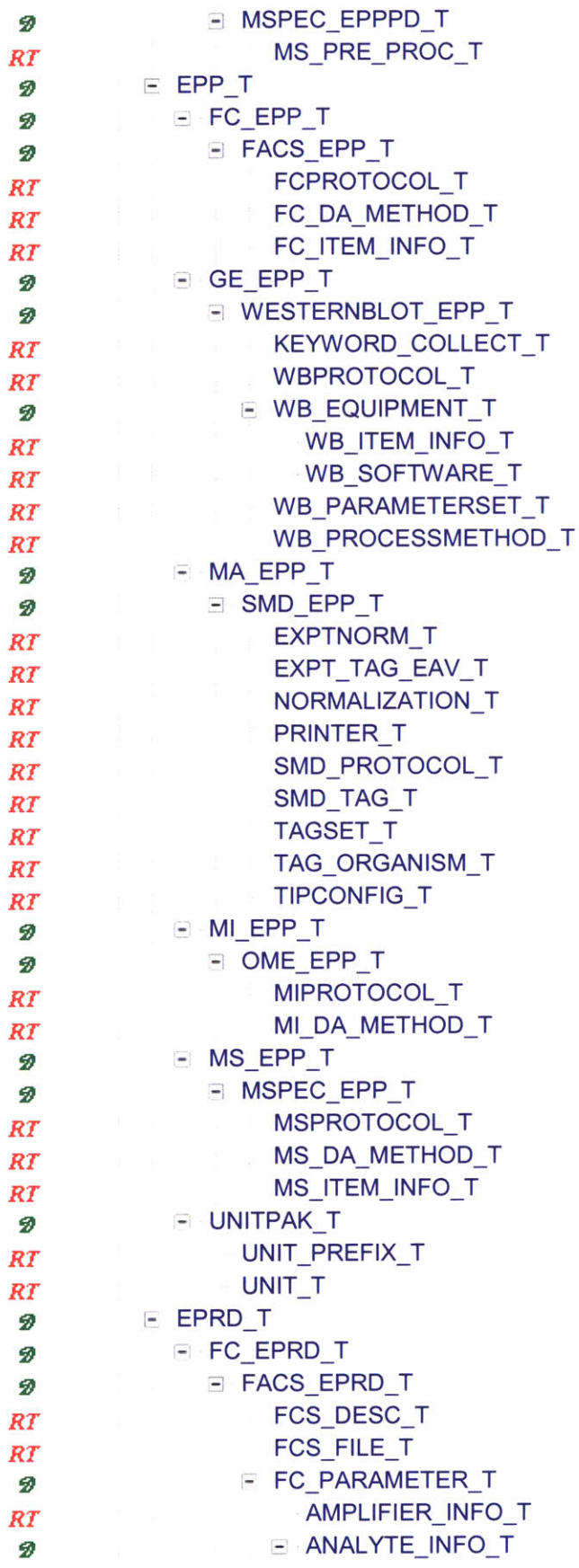


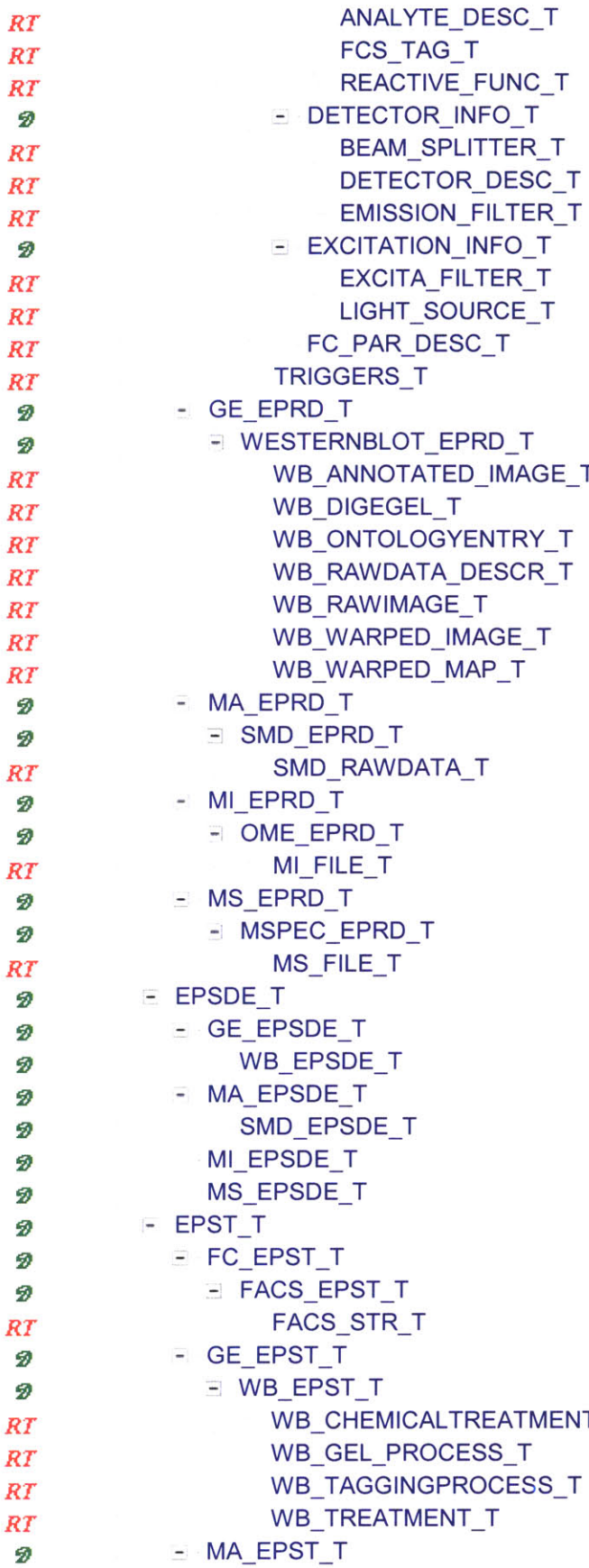
Figure 49 The extensible design of ExperiBase

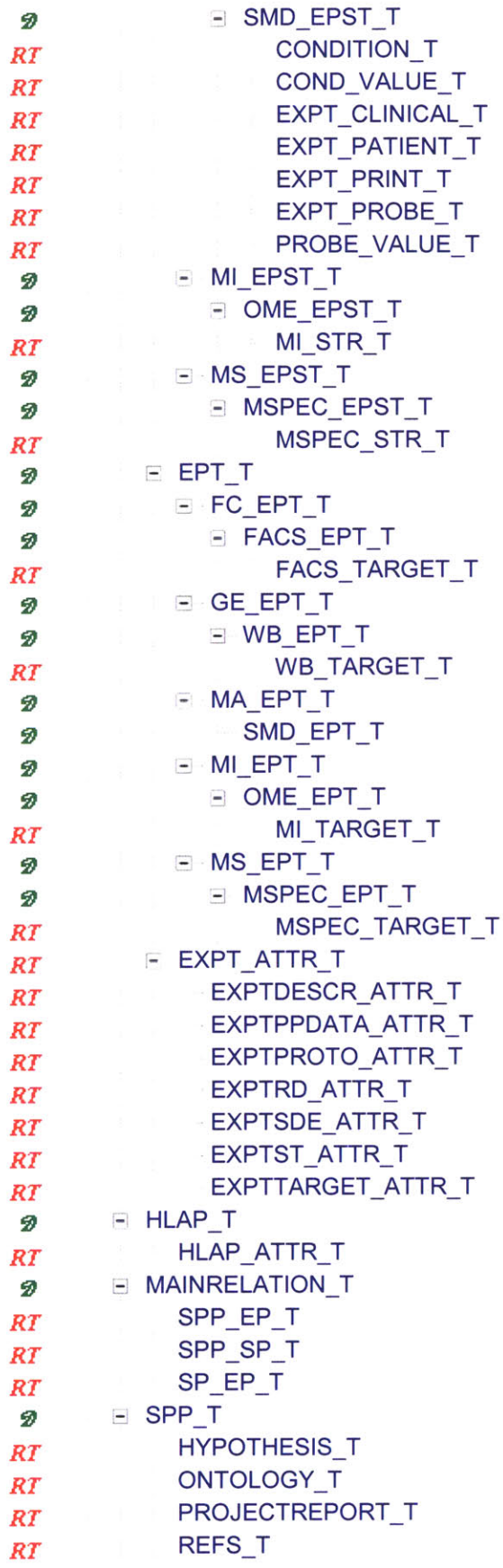
Using the generalization strategy, I have combined the five IODs of the previous chapter together to form a powerful single schema. Therefore a single database has been implemented to support the five experimental methods. The below is a list of the entities of the single schema (current implementation) to support the five experimental modalities. "D" means that the entity is only a data type; "RT" means that the entity is a data type and has a matched table using the data type. "AP" means Administration Package; "SP" means Sample Package; "SPP" means Study Plan Package; "HLAP" means High Level Analysis Package; "EP" means Experimental Package. The tables "ADMIN_T", "SAMPLE_ATTR", "STUDYPLAN_ATTR", "EXPT_ATTR", and "HLAP_ATTR" are summary/annotation tables to record all UIDs in the relevant package. "FC", "GE", "MA", "MI", and "MS" means Flow Cytometry, Gel Electrophoresis, MicroArray, Microscope Image, and Mass Spectrometry, respectively. "EPD" means experiment description; "EPT" means experiment target; "EPP" means experiment protocol; "EPRD" means experiment raw data; "EPPPD" means experiment preprocessed data; "EPSDE" means experiment special design element. "PSP" means physical sample package; "DSP" means derived sample package; "MSP" means measured sample package.

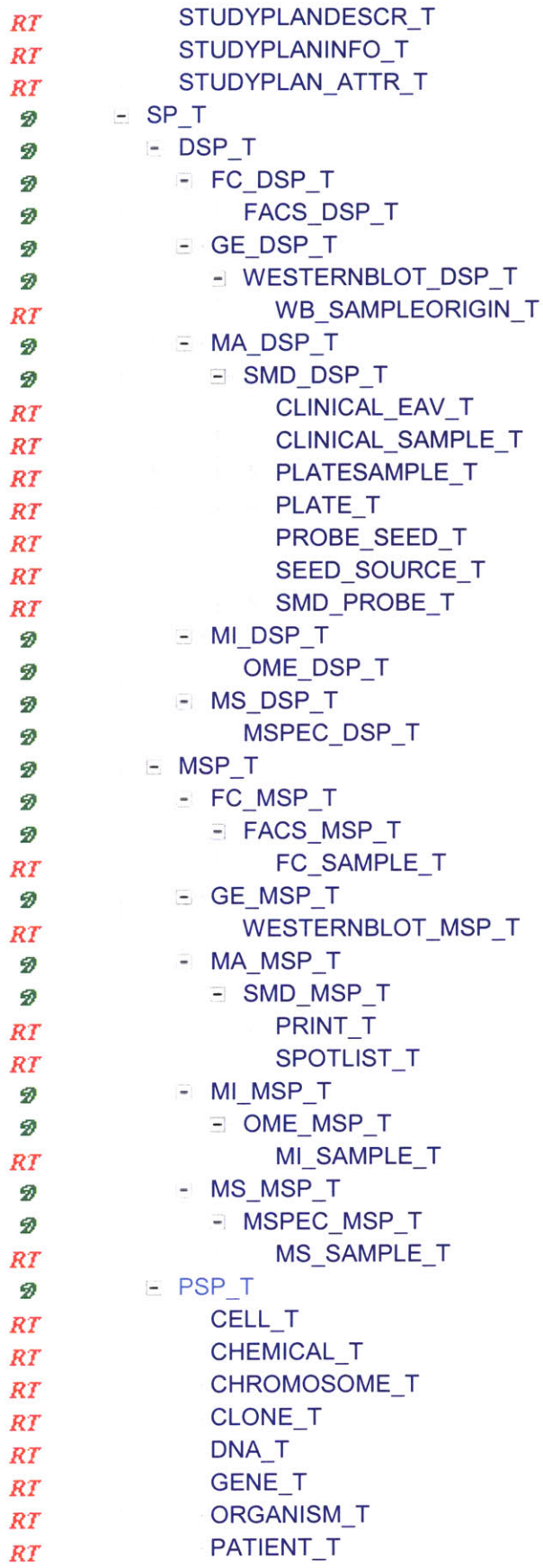
- EXPERIBASE_T
  - AP_T
    - RT ADMIN_T
    - MIT_T
      - RT LAB_T
      - RT PERSON_T
    - SMD_T
      - RT ACCESS_GROUP_T
      - RT DBUSER_T
      - RT GROUP_USER_T
      - RT PLATELOCATION_T
  - EP_T
    - EPD_T


FC_EPD_T
FACS_EPD_T
FCPROTOCOL_L_T
FC_EXPTDESCR_T
GE_EPD_T
WB_EPD_T
WBPROTOCOL_L_T
WB_EXPTDESCR_T
MA_EPD_T
SMD_EPD_T
CATEGORY_T
ETYPE_TAGSET_T
EXPTTYPE_T
EXPT_ACCESS_T
SMD_EXPTATTR_T
SMD_EXPTDESCR_T
SUBCATEGORY_T
MI_EPD_T
OME_EPD_T
MIPROTOCOL_L_T
MI_EXPTDESCR_T
MS_EPD_T
MSPEC_EPD_T
MSPROTOCOL_L_T
MS_EXPTDESCR_T
EPPPD_T
FC_EPPPD_T
FACS_EPPPD_T
FC_DOTPLOT_T
FC_HISTOGRAM_T
FC_HISTO_DATA_T
FC_HISTO_DESC_T
FC_PRE_PROC_T
GE_EPPPD_T
WB_EPPPD_T
WB_BAND_T
WB_BANDDESCR_T
WB_DIGEGELITEM_T
WB_RELATEDGELITEM_T
WB_PRE_PROC_T
MA_EPPPD_T
SMD_EPPPD_T
SMD_RESULT_T
MI_EPPPD_T
OME_EPPPD_T
MI_PRE_PROC_T
MS_EPPPD_T


MSPEC_EPPPD_T
MS_PRE_PROC_T
EPP_T
FC_EPP_T
FACS_EPP_T
FCPROTOCOL_T
FC_DA_METHOD_T
FC_ITEM_INFO_T
GE_EPP_T
WESTERNBLOT_EPP_T
KEYWORD_COLLECT_T
WBPROTOCOL_T
WB_EQUIPMENT_T
WB_ITEM_INFO_T
WB_SOFTWARE_T
WB_PARAMETERSET_T
WB_PROCESSMETHOD_T
MA_EPP_T
SMD_EPP_T
EXPTNORM_T
EXPT_TAG_EAV_T
NORMALIZATION_T
PRINTER_T
SMD_PROTOCOL_T
SMD_TAG_T
TAGSET_T
TAG_ORGANISM_T
TIPCONFIG_T
MI_EPP_T
OME_EPP_T
MIPROTOCOL_T
MI_DA_METHOD_T
MS_EPP_T
MSPEC_EPP_T
MSPROTOCOL_T
MS_DA_METHOD_T
MS_ITEM_INFO_T
UNITPAK_T
UNIT_PREFIX_T
UNIT_T
EPRD_T
FC_EPRD_T
FACS_EPRD_T
FCS_DESC_T
FCS_FILE_T
FC_PARAMETER_T
AMPLIFIER_INFO_T
ANALYTE_INFO_T


ANALYTE_DESC_T
FCS_TAG_T
REACTIVE_FUNC_T
DETECTOR_INFO_T
BEAM_SPLITTER_T
DETECTOR_DESC_T
EMISSION_FILTER_T
EXCITATION_INFO_T
EXCITA_FILTER_T
LIGHT_SOURCE_T
FC_PAR_DESC_T
TRIGGERS_T
GE_EPRD_T
WESTERNBLOT_EPRD_T
WB_ANNOTATED_IMAGE_T
WB_DIGEGEL_T
WB_ONTOLOGYENTRY_T
WB_RAWDATA_DESCR_T
WB_RAWIMAGE_T
WB_WARPED_IMAGE_T
WB_WARPED_MAP_T
MA_EPRD_T
SMD_EPRD_T
SMD_RAWDATA_T
MI_EPRD_T
OME_EPRD_T
MI_FILE_T
MS_EPRD_T
MSPEC_EPRD_T
MS_FILE_T
EPSDE_T
GE_EPSDE_T
WB_EPSDE_T
MA_EPSDE_T
SMD_EPSDE_T
MI_EPSDE_T
MS_EPSDE_T
EPST_T
FC_EPST_T
FACS_EPST_T
FACS_STR_T
GE_EPST_T
WB_EPST_T
WB_CHEMICALTREATMENT_T
WB_GEL_PROCESS_T
WB_TAGGINGPROCESS_T
WB_TREATMENT_T
MA_EPST_T


SMD_EPST_T
CONDITION_T
COND_VALUE_T
EXPT_CLINICAL_T
EXPT_PATIENT_T
EXPT_PRINT_T
EXPT_PROBE_T
PROBE_VALUE_T
MI_EPST_T
OME_EPST_T
MI_STR_T
MS_EPST_T
MSPEC_EPST_T
MSPEC_STR_T
EPT_T
FC_EPT_T
FACS_EPT_T
FACS_TARGET_T
GE_EPT_T
WB_EPT_T
WB_TARGET_T
MA_EPT_T
SMD_EPT_T
MI_EPT_T
OME_EPT_T
MI_TARGET_T
MS_EPT_T
MSPEC_EPT_T
MSPEC_TARGET_T
EXPT_ATTR_T
EXPTDESCR_ATTR_T
EXPTPPDATA_ATTR_T
EXPTPROTO_ATTR_T
EXPTRD_ATTR_T
EXPTSDE_ATTR_T
EXPTST_ATTR_T
EXPTTARGET_ATTR_T
HLAP_T
HLAP_ATTR_T
MAINRELATION_T
SPP_EP_T
SPP_SP_T
SP_EP_T
SPP_T
HYPOTHESIS_T
ONTOLOGY_T
PROJECTREPORT_T
REFS_T


RT STUDYPLANDESCR_T
RT STUDYPLANINFO_T
RT STUDYPLAN_ATTR_T
SP_T
DSP_T
FC_DSP_T
FACS_DSP_T
GE_DSP_T
WESTERNBLOT_DSP_T
RT WB_SAMPLEORIGIN_T
MA_DSP_T
SMD_DSP_T
RT CLINICAL_EAV_T
RT CLINICAL_SAMPLE_T
RT PLATESAMPLE_T
RT PLATE_T
RT PROBE_SEED_T
RT SEED_SOURCE_T
RT SMD_PROBE_T
MI_DSP_T
OME_DSP_T
MS_DSP_T
MSPEC_DSP_T
MSP_T
FC_MSP_T
FACS_MSP_T
RT FC_SAMPLE_T
GE_MSP_T
RT WESTERNBLOT_MSP_T
MA_MSP_T
SMD_MSP_T
RT PRINT_T
RT SPOTLIST_T
MI_MSP_T
OME_MSP_T
RT MI_SAMPLE_T
MS_MSP_T
MSPEC_MSP_T
RT MS_SAMPLE_T
PSP_T
RT CELL_T
RT CHEMICAL_T
RT CHROMOSOME_T
RT CLONE_T
RT DNA_T
RT GENE_T
RT ORGANISM_T
RT PATIENT_T

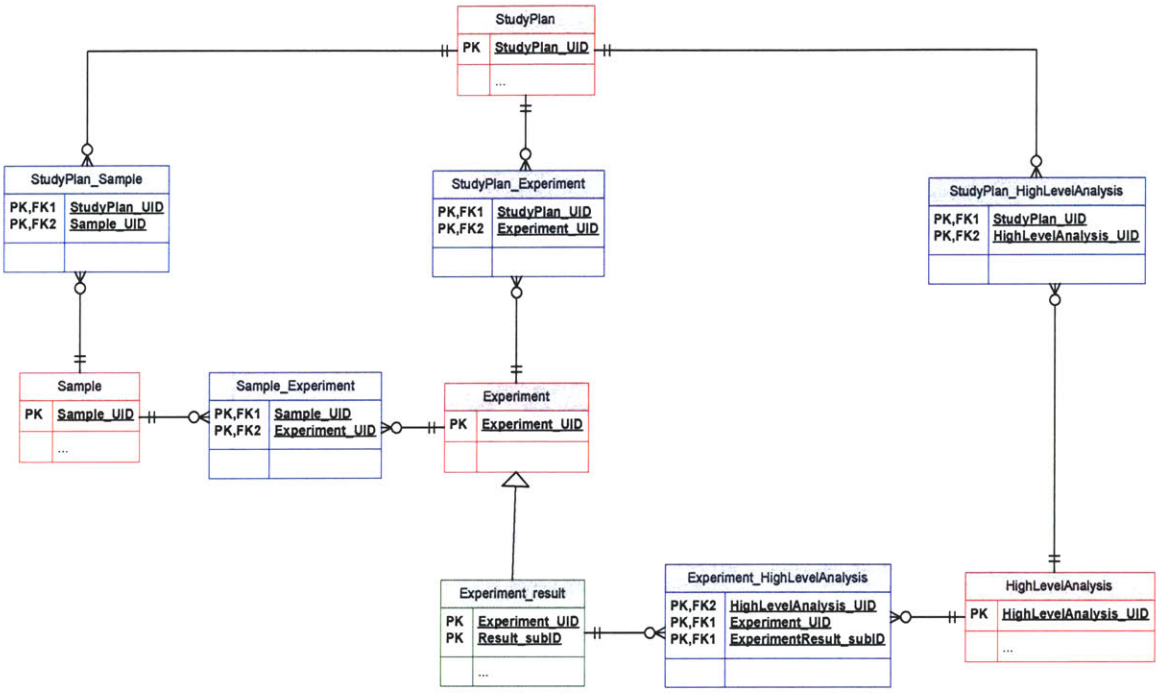
RT PEPTIDE_T
RT PROTEIN_T
RT RNA_T
RT SEQTYPE_T
RT SPECIES_T
RT STANFORDSEQ_T
RT TISSUE_T
RT SAMPLE_ATTR_T

6.2 Database Implementation

Since the IODs are using the object-relational data model, we need a database management system which can support this feature. As we discussed in the Chapter 4, Informix, DB2 and Oracle are three good database management systems that support object-relational schema. Although there are other free database systems available, their object-relational features are so limited that can not represent our IODs. Therefore, in this thesis, I have used Informix, DB2 and Oracle to implement the database.

To implement the many-to-many relations between project (i.e., study plan), sample, experiment, and high level analysis, we use intermediate tables, as shown in the below diagram. The diagram indicates that:

- A project can have multiple samples; a sample can be used in many projects.
- A project can do multiple experiments; the data of an existing experiment can be reused or cited by multiple projects.
- A project can have many high level analyses; a set of high level analysis can be used or cited by multiple projects.
- A sample can be used in many experiments; an experiment can examine multiple samples.
- A set of results of an experiment can be analyzed by multiple high level analysis procedures; a high level analysis can use multiple results from experiments.



Red color: represents that the box is the main entity of a package.
Green color: represents that the box is a child entity of a package.
Blue color: represents that the box is an intermediate entity between two packages.

Figure 50 Main relationships of ExperiBase

Auditing information has been recorded in every entity in our schema. Besides that, we have designed four intermediate tables to implement the many-to-many relations of administration to project, sample, experiment, and high level analysis, as shown in the below diagram. The diagram means that a person can take charge of multiple projects, own multiple samples, do multiple experiments, and have multiple high level analyses; a project can be leaded by multiple persons; a sample can be prepared by multiple persons; an experiment can be done by multiple persons; and a high level analysis can be done by multiple persons.

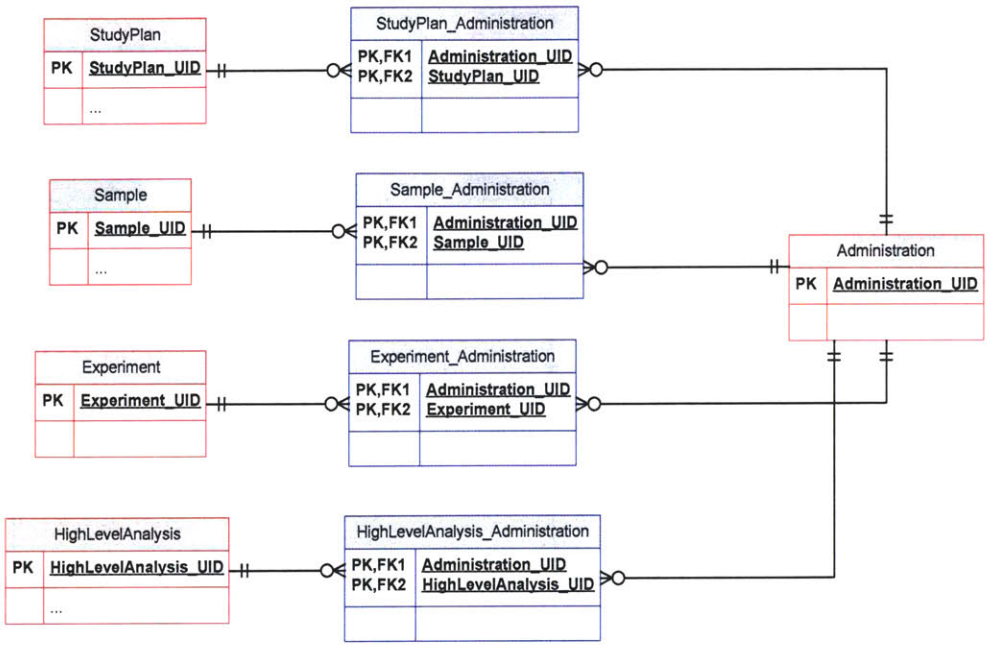


Figure 51 The relations of administration with project, sample, experiment, and high level analysis

The single generalized ExperiBase schema has been implemented in DB2 and Oracle. I have used the feature of user defined data types. Each entity in the IODs has been implemented as a user defined data type. The inheritance of the entities of the IODs has been implemented as an inheritance of the user defined data types. For example, the below codes implement the inheritance from the entity "ExperiBase" (root), to "AP" (Administration Package), to "person".

```
----------Root

CREATE TYPE ExperiBase_t AS OBJECT
(DATE_CREATED TIMESTAMP,
DATE_MODIFIED TIMESTAMP)
NOT INSTANTIABLE
NOT FINAL;
/

----------Admin package

CREATE TYPE AP_t UNDER ExperiBase_t () NOT INSTANTIABLE NOT FINAL;
/

CREATE TYPE MIT_t UNDER AP_t ()
NOT INSTANTIABLE
NOT FINAL;
/

CREATE TYPE Person_t UNDER MIT_t
(Admin_UID VARCHAR2(128),
Title VARCHAR2(255),
Firstname VARCHAR2(64),
Middlename VARCHAR2(64),
Lastname VARCHAR2(64),
```

```
Suffix VARCHAR2(32),
PositionTitle VARCHAR2(128),
Username VARCHAR2(16),
Password VARCHAR2(16),
Userstatus CHAR(16),
CREATED_BY VARCHAR2(16),
MODIFIED_BY VARCHAR2(16))
NOT FINAL;
/

CREATE TABLE Person OF Person_t;
```

Note that some of the user defined data types don't necessarily need matched tables, for example, the root element "ExperiBase" and the "AP" data type. Because the mechanisms of table inheritance are different between DB2 and Oracle, to make our design independent of platforms, I haven't use the mechanism of table inheritance. Instead, I have used type inheritance. In our database implementation, I have also used many other object-relational database features, such as scoped references, referential integrity, triggers, user defined methods, stored procedures, and external classes associated with tables or types. For example, the code "Sample REF SP_t" in the below codes implements the scope reference:

```
CREATE TYPE SP_t UNDER ExperiBase_t
(Created_By REF Admin_t ,
Modified_By REF Admin_t )
NOT INSTANTIABLE
NOT FINAL;
/

CREATE TYPE PSP_t UNDER SP_t () NOT INSTANTIABLE
NOT FINAL;
/

CREATE TYPE Organism_t UNDER PSP_t
(Sample_UID VARCHAR2(128),
OrganismAbbrev VARCHAR2(10),
CommonName VARCHAR2(30),
GenusSpecies VARCHAR2(60),
label VARCHAR2(64),
content VARCHAR2(500))
NOT FINAL;
/

CREATE TYPE Cell_t UNDER PSP_t
(Sample_UID VARCHAR2(128),
abbrev VARCHAR2(10),
CommonName VARCHAR2(30),
GenusSpecies VARCHAR2(60),
type VARCHAR2(30),
source VARCHAR2(64),
label VARCHAR2(64),
content VARCHAR2(500)
)
NOT FINAL;
```

```
/
...
...
...

CREATE TYPE WB_Treatment_t UNDER WB_EPST_t
(Experiment_UID VARCHAR2(128),
label VARCHAR2(64),
Sample REF SP_t,
treatment_name VARCHAR2(32),
material REF chemical_t,
dose real,
dose_unit_prefix REF unit_prefix_t,
dose_unit REF unit_t,
concentration real,
concentra_unit_prefix REF unit_prefix_t,
concentra_unit REF unit_t,
ph real,
duration real,
duration_unit_pref REF unit_prefix_t,
duration_unit REF unit_t,
temperature real,
temperature_unit_p REF unit_prefix_t,
temperature_unit REF unit_t,
treatment_date timestamp,
description VARCHAR2(240))
NOT FINAL;
/
CREATE TABLE WB_Treatment OF WB_Treatment_t
(dose_unit_prefix SCOPE IS unit_prefix,
dose_unit SCOPE IS unit,
concentra_unit_prefix SCOPE IS unit_prefix,
concentra_unit SCOPE IS unit,
duration_unit_pref SCOPE IS unit_prefix,
duration_unit SCOPE IS unit,
temperature_unit_p SCOPE IS unit_prefix,
temperature_unit SCOPE IS unit,
material SCOPE IS Chemical,
Date_Created NOT NULL,
Created_By SCOPE IS Admin,
Date_Modified NOT NULL,
Modified_By SCOPE IS Admin);
```

This means that the attribute "sample" of the "WB_treatment_t" can refer to any child data type of SP_t, such as organism, or cell. We can use the DEF function to know what the exact data type is. Since the underlying schema is highly modular, the expanded query scope is extremely powerful.

The below diagram illustrates the database implementation about the inheritance, attributes, and the main relationships. The box 3 represents a relational table to record the many-to-many relations between sample UID and experiment UID. In each sub-tree, the relationships of objects are inherited relationship. Each object has attributes, methods, and constraints.

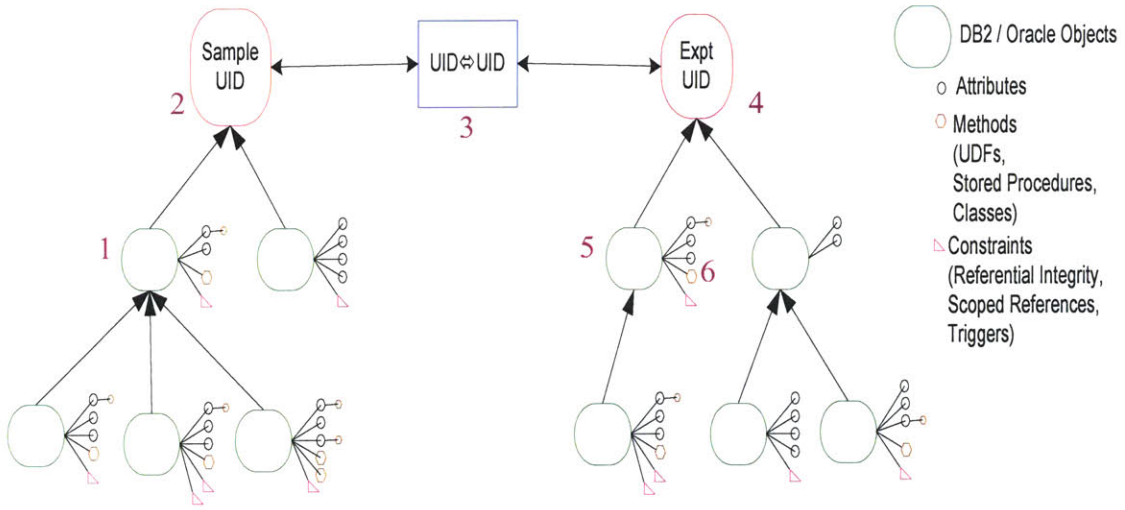


Figure 52 Using object-relational features to implement the ExperiBase design schema

How will the query processes work? For example, suppose that we want to find out all experiments which have the sequence "E. Coli #33'' expression (R/G) ratio>2.0, then do a hierarchical clustering analysis. We can present the query process as

Clustering(Expt_UID(Sample_UID->sequence.name(E.Coli#33))->preprocessData.rgRatio>2.0)

The meaning is that first we need to query against the sequence table to find out the sample UID of a sequence with the name as "E. Coli #33", (the ellipse 1 and 2); second, we use the box 3 to find out the matched experiment UID(s); and then, filter out the preprocessed data with R/G ratio large than 2.0; and finally load the external class "hierarchical clustering" to analyze the data.

6.3 The Web-based System of ExperiBase

The thesis provided a web-based system - ExperiBase for browsing database, uploading data into database, and querying database. All of these functions have web-enabled friendly user interfaces which can run on any browser. I list several web pages and results here. (http://ExperiBase.mit.edu). Figure 53 is the homepage of ExperiBase. The web pages for browsing the database are called as ExperiBase Explorer. The Figures 54-61 are the snapshots of ExperiBase Explorer. We can search the information about a study plan, a sample or an experiment by its UID (Figure 55); or using the inheritance tree to browse the entire database (Figure 56). The tree reflects the inheritance of database objects. Each node of the tree is clickable and contains the information about the node, which is either a data type or a table definition. The web interfaces can query any kind of data and display them online. I provided the functions about parsing Excel Spreadsheets and displaying in the HTML format; converting ImageQuant images to the JPEG format and displaying the images online; on-line data analysis for FACS data (Figure 58). Figure 59 is the query pages, which can search any information about project, sample, experiment, high level analysis, and administration. The results are displayed as XML messages first, and provide links to any UIDs in the messages; and then you can click a button to display the messages as tables. Figure 60 is the upload pages. Figure 61

is some old web pages of ExperiBase v1.0. It provides three types of web pages: for viewers, experimenters, and superusers.

Figure 53 The homepage of ExperiBase. It provides three main functions: browse database, query database, and upload data into database. (http://ExperiBase.mit.edu)

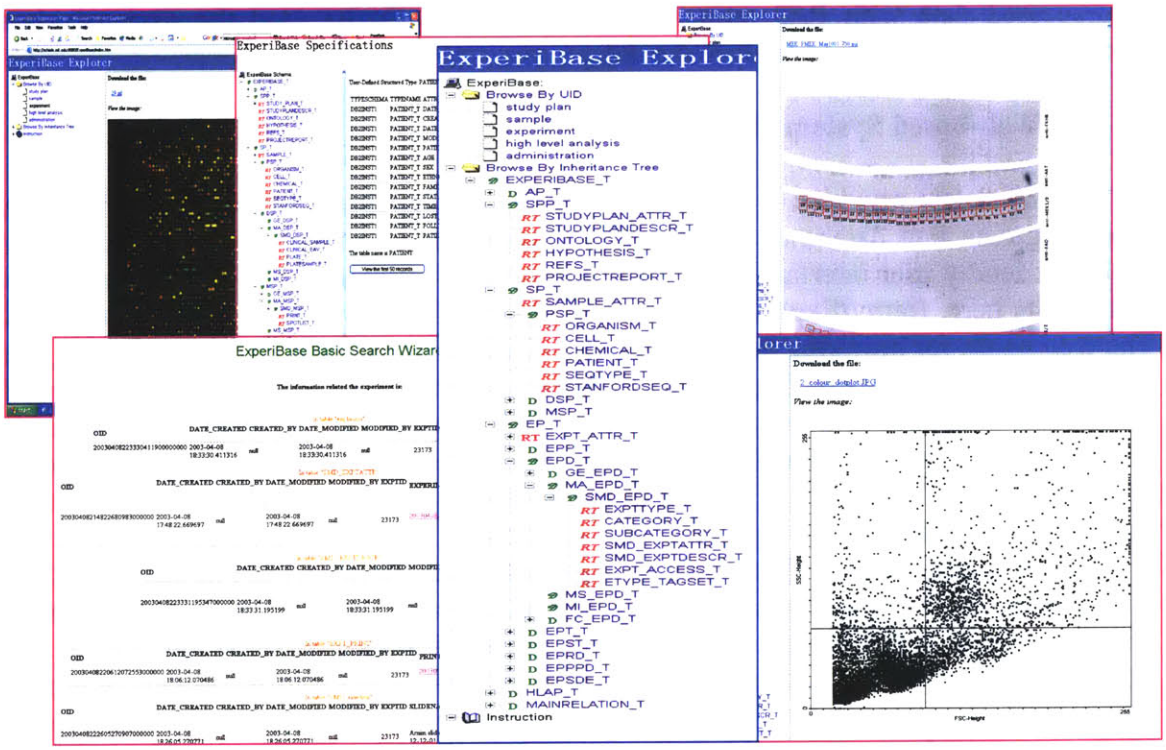


Figure 54 The snapshots of the web pages of ExperiBase

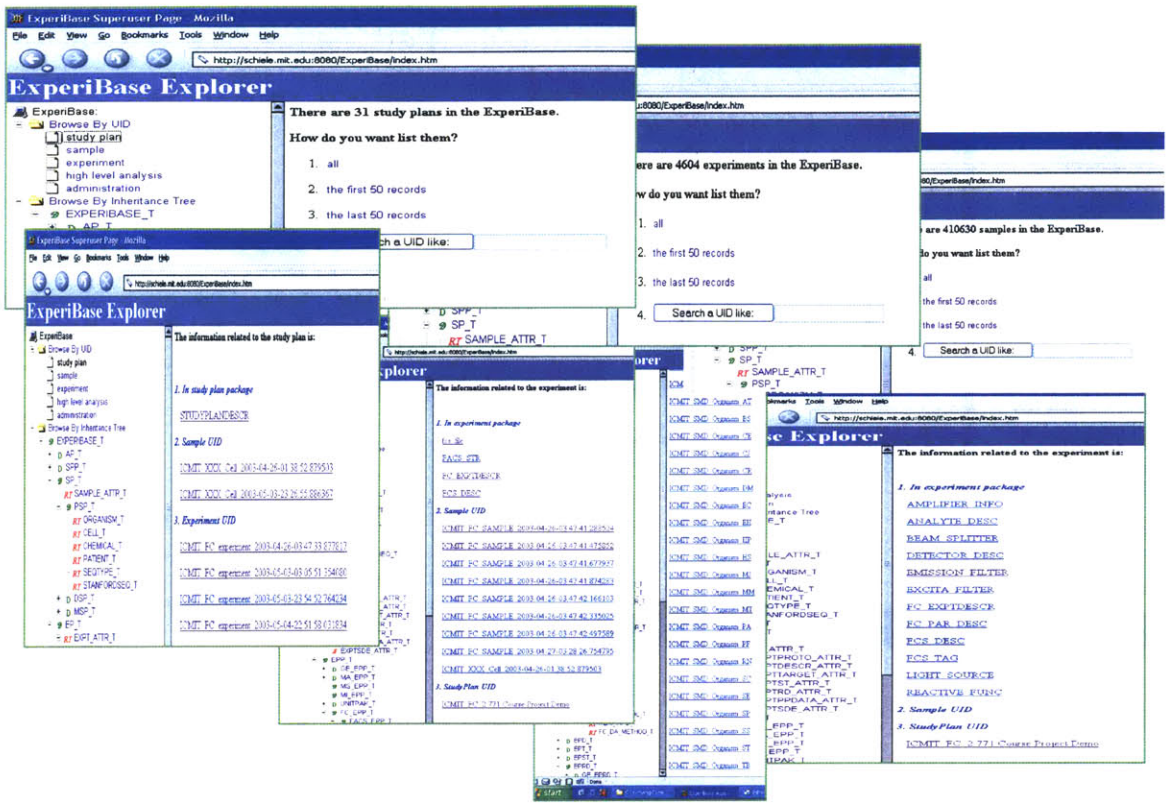


Figure 55 Search by the UIDs

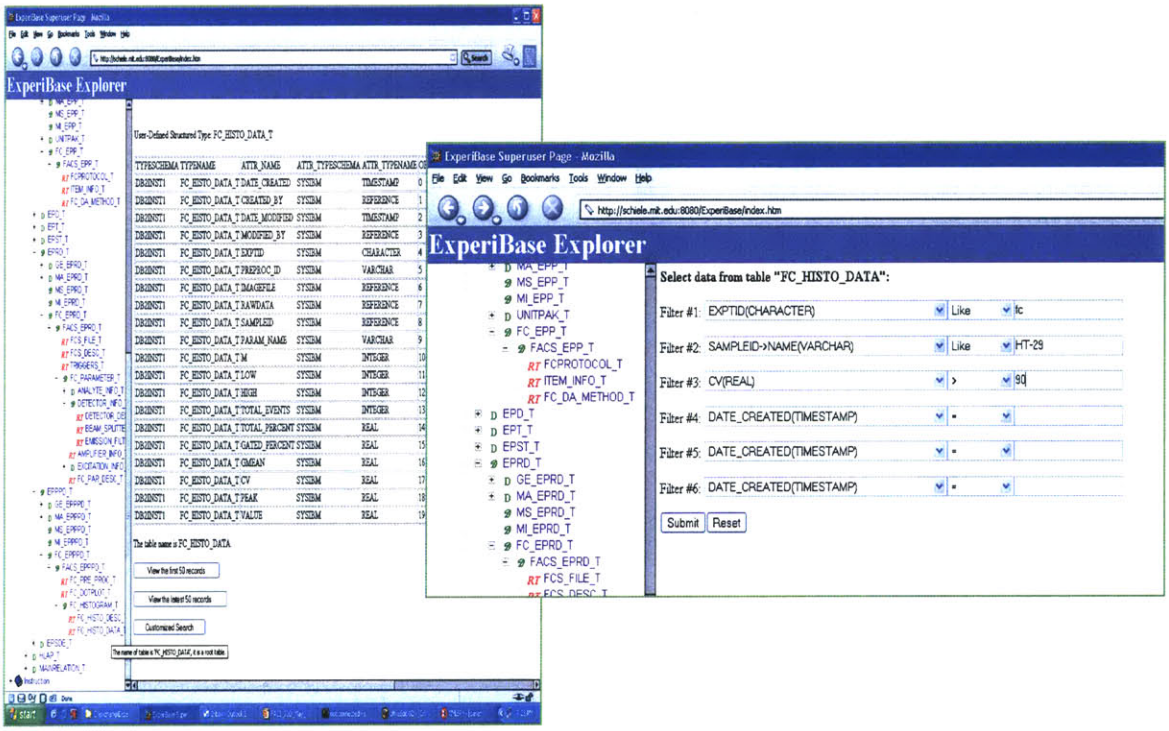


Figure 56 Browse by the inheritance tree

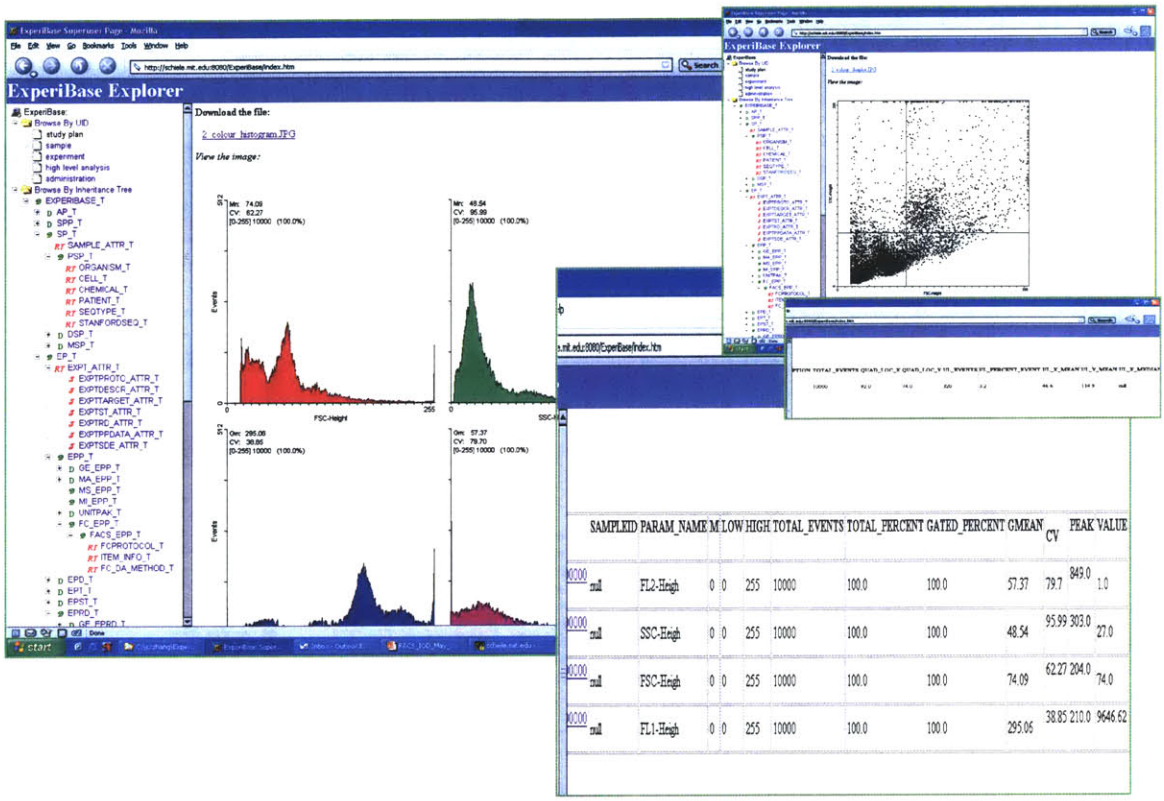


Figure 57 Plots and meta data

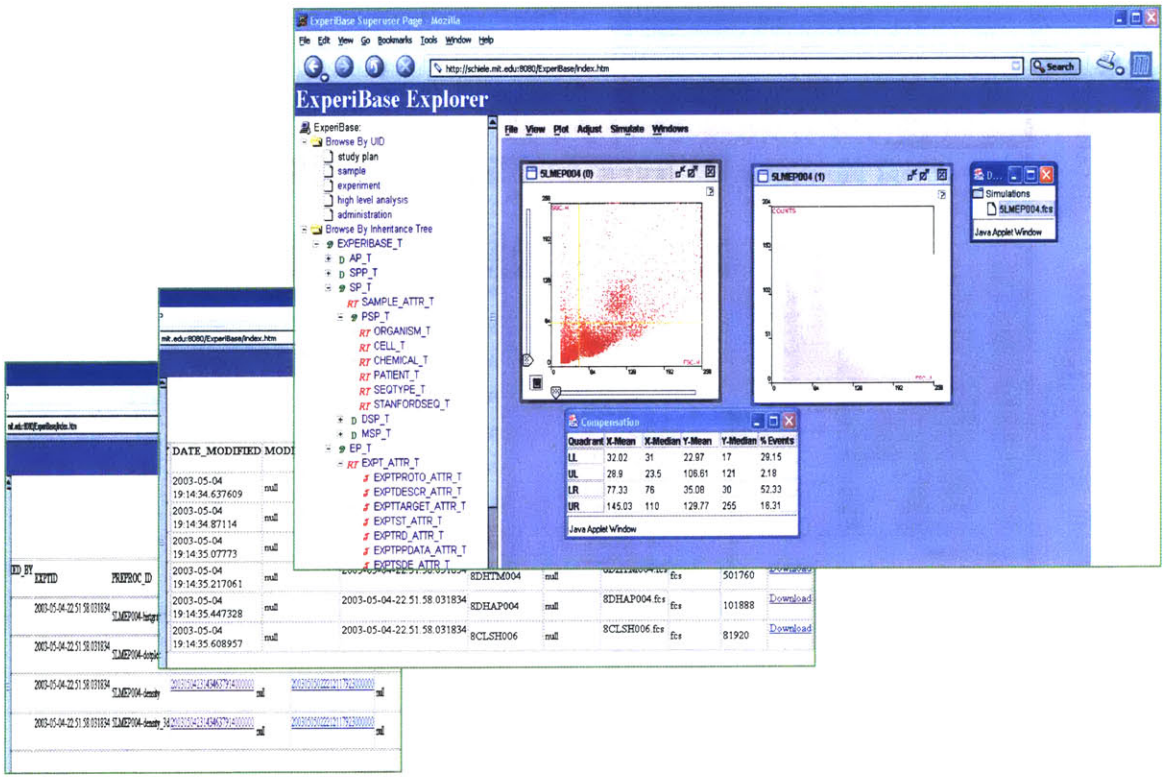


Figure 58 Data download and online analysis

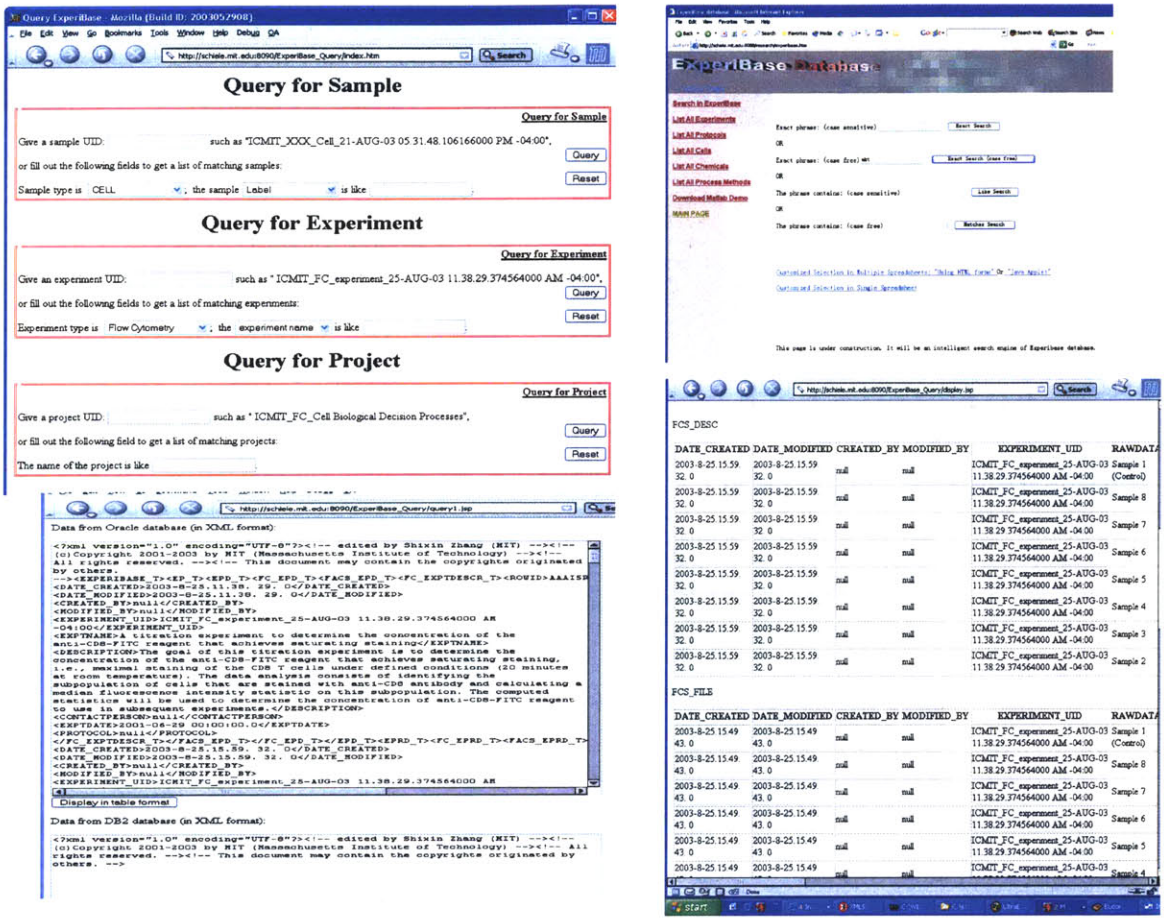


Figure 59 Query pages

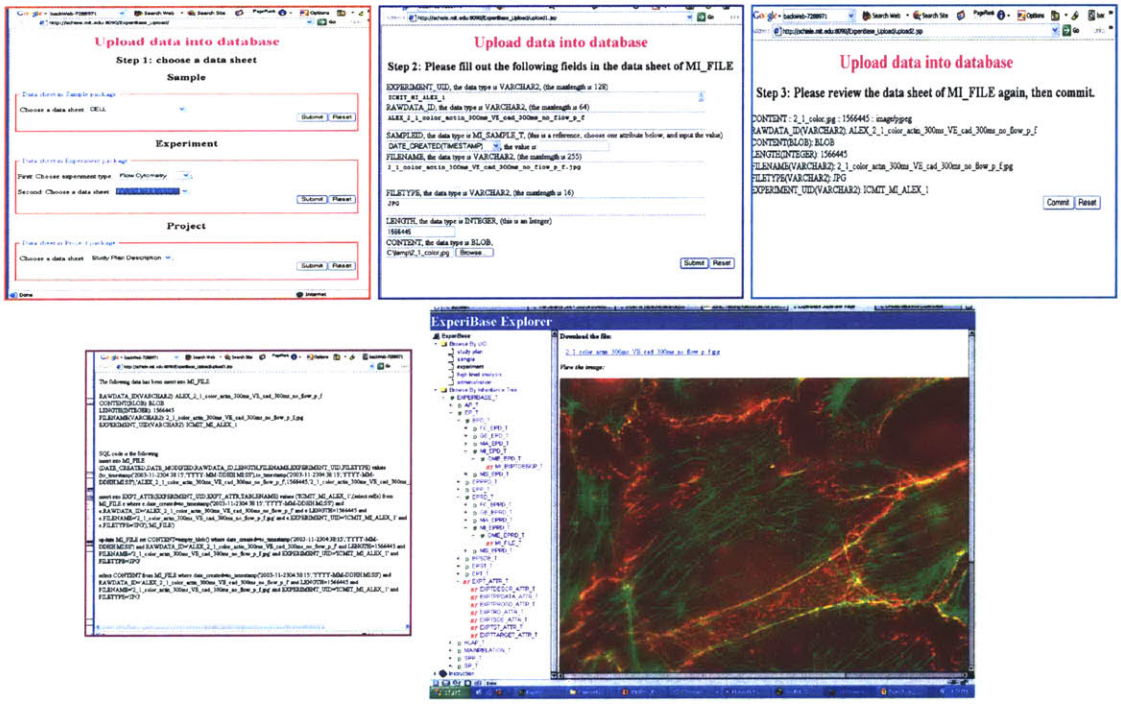


Figure 60 Upload pages

Figure 61 The old web pages of ExperiBase, providing the functions about uploading data, listing experiments, viewing, parsing Excel spreadsheets, and querying against the database

Those web pages are using the software architecture shown in Figure 62. The web pages are dynamically generated by either Java Servlets or Java Server Pages (JSP). The servlets or JSPs are using JDBC and Class Mappers communicating with database. The SOAP and CORBA/IIOP protocols are used as transporting layers. XML is used as a data transfer format between clients and servers. The architecture is platform independent. It is also extensible, because of the middleware features of SOAP or CORBA[161]. This has discussed in my master thesis. The code is generic and not hard-coded. For example, the ExperiBase explorer, query pages, and upload pages are suitable for any tables and data types; you can shift database connections from DB2 to Oracle or Informix using the configuration files, without recompiling the code. The navigation structure of ExperiBase explorer site is shown in Figure 63. Figure 64 illustrates the navigation structure of query and upload site. The Java and SQL code can be downloaded for free (http://experibase.mit.edu).

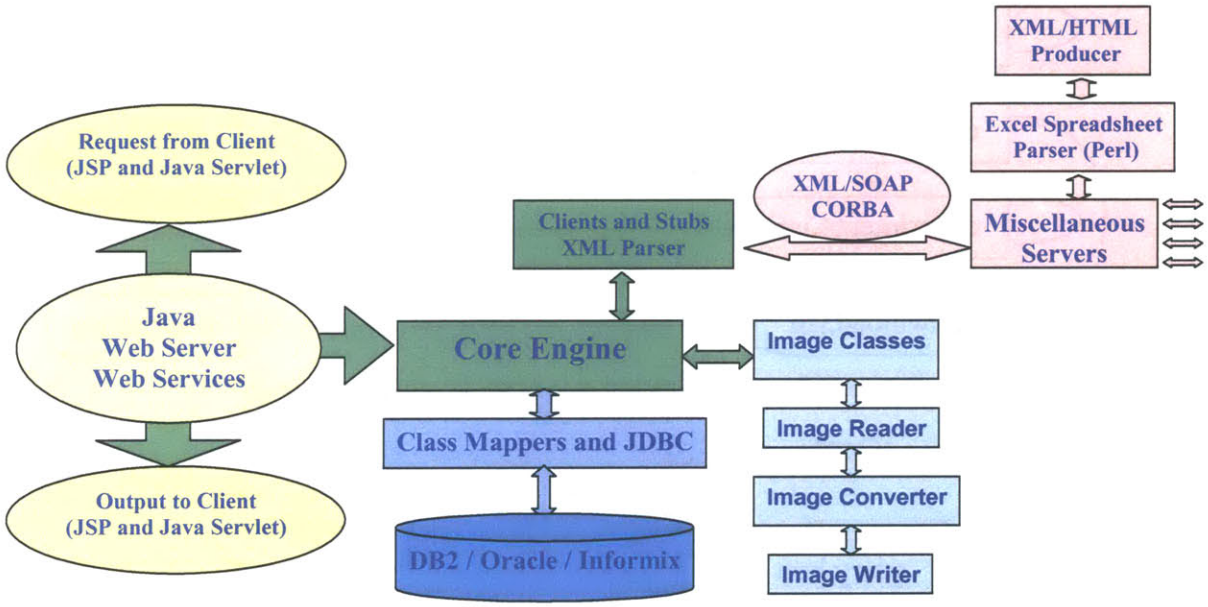


Figure 62 The software architecture of ExperiBase

Notes:

1. The data in ExperiBase comes from:
   - WesternBlot Experiments: Dr. Suzanne Gaudet, MIT and Loel Kathmann, PNNL
   - FACS Experiments: Dr. Suzanne Gaudet, MIT; Loel Kathmann; FlowJo demo data sets; Facsimile demo data sets.
   - Microarray Experiments: Stanford Microarray Database
2. The FACS data online analysis software (the Java Applet) was developed by JCSMR Flow Cytometry Laboratory[162]. The JavaScript code of tree menu is from "Morten's JavaScript Tree Menu"[163]. The Java servlet multipart parser package (com.oreilly.servlet) is from servlets.com, developed by Jason Hunter[164].

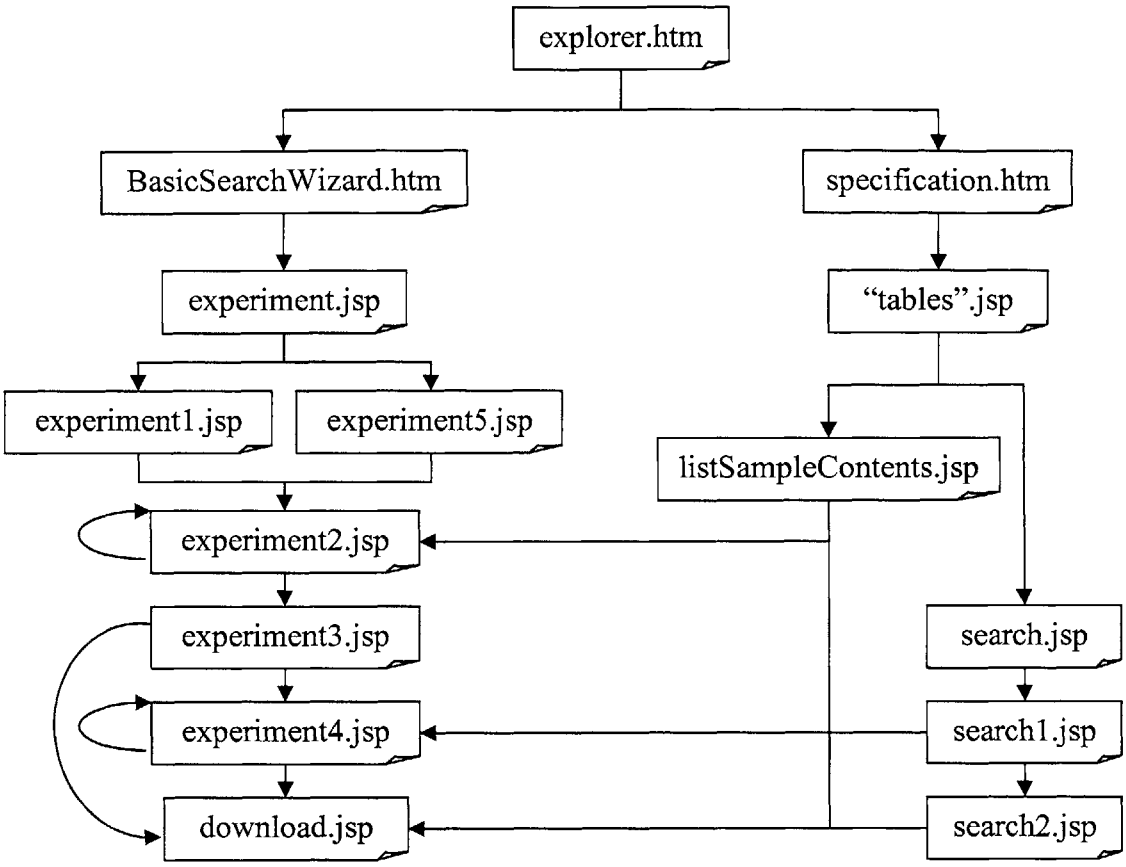


Figure 63 The navigation structure of the ExperiBase's Explorer

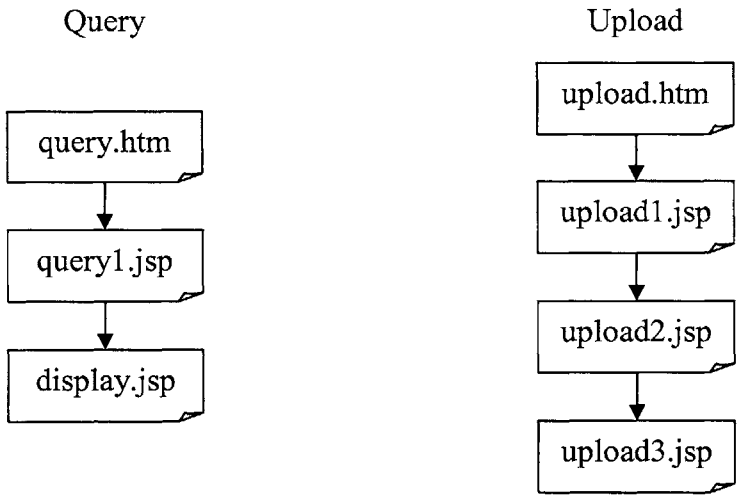


Figure 64 The navigation structure of the ExperiBase's query and upload site

6.4 Summary

This chapter has generalized a single coherent set of information object definitions to support the five experimental modalities; has introduced several important issues about implementing the IODs to an object-relational database. A web-based

system – ExperiBase has been developed and some snapshots of the web pages are listed in the chapter.

Chapter 7. Interoperability of ExperiBase

Once we have the generalized UML schema described in the previous chapter, we can design an object-relational database schema, or we can also generate an XML schema from it. Then we can import or export data in or from database using XML document format, as shown in the below diagram. This chapter is going to introduce the XML schema and some examples using the ExperiBase XML. The web services designed for the project are also going to be introduced. And then a new mechanism of database federation using the ExperiBase XML and web services is going to be discussed.

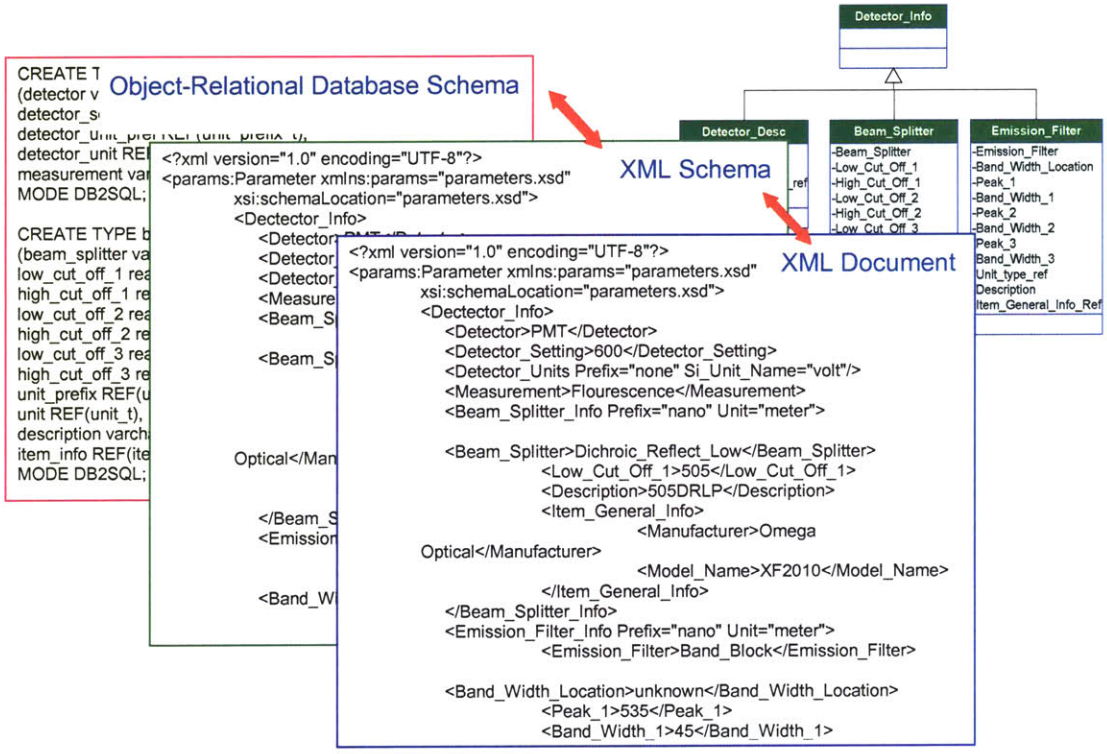


Figure 65 Import and export data from the ExperiBase database using XML format. (The XML schema and XML document is from CytometryML)

7.1 ExperiBase XML

We have introduced in the Chapter 4 that an object-oriented data model can be expressed as an XML schema. Since we have already designed the generalized IOD, we can use the model to create an ExperiBase XML schema. The XML schema has a root element, named as "ExperiBase", as shown in the below Figure 66; it has five children elements, which are administration package, project package, sample package, experiment package, and high level analysis package. Since the only difficulty of the XML schema design is learning the language and there is no further technical barrier than the IODs, the detail of the design is omitted here. You can download the ExperiBase XML schema for free from the ExperiBase web site. A portion of the schema view of the ExperiBase XML schema is shown in Figure 67.

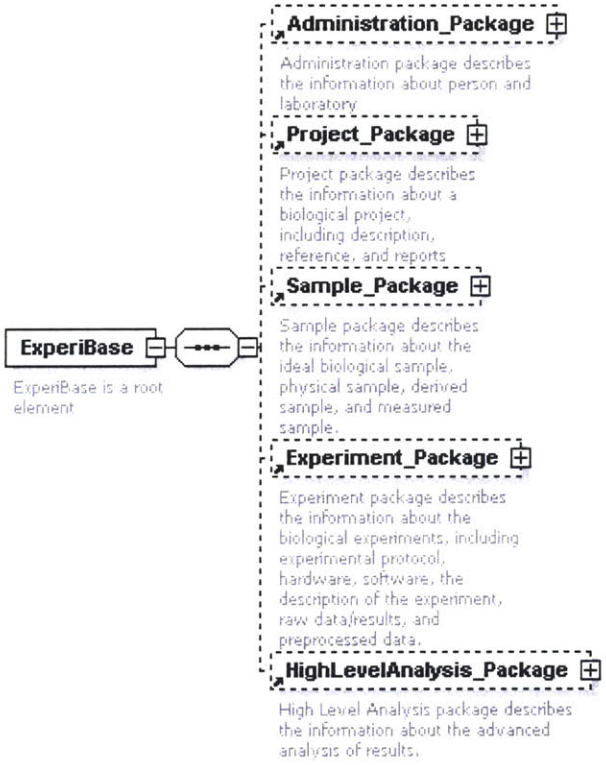


Figure 66 ExperiBase XML schema – a structure view of the top five elements.

We can either manually code by ourselves to generate the XML schema according to the IOD; or, design a program that can convert an object-relational database schema to an XML schema automatically. The below is a portion of the Java code to generate the XML schema dynamically and automatically from the ExperiBase database.

```
...
static String le="<xs:element name=\"";
static String la="\"><xs:complexType><xs:sequence>\n";
static String le1="</xs:sequence></xs:complexType></xs:element>\n";
public static void setTree(String tab,String table)
{
   String tableNames[] = getChildTable(table);
   results+=tab+le+table+la;
   for(int i = 0; i < tableNames.length; i++)
   {
      String tName=tableOrNot(tableNames[i]);
      int childrenCount = getChildrenCount(tableNames[i]);
      if(childrenCount == 0){
       if (tName!=""){
            results+=tab+"\t"+le+tableNames[i]+la;
            getColumns(tab+"\t"+"\t",tableNames[i]);
            results+=tab+"\t"+le1;
        }
        else
            results+=tab+"\t"+le+tableNames[i]+"\"/>\n";
      }else{
       if (tName!="")
            getColumns(tab+"\t"+"\t",tableNames[i]);
         setTree(tab+"\t",tableNames[i]);
      }
```

```
        }
        results+=tab+le1;
    }

    public static void getColumns(String tab,String table)
    {
        String query = "select attr_name,attr_type_name,length from user_type_attrs where type_name='"+table+"'";
//,attr_type_name,length
        String temp[][] = getOutput(query);
        for(int i = 0; i < temp.length - 1; i++)
        {
            results+=tab+le+temp[i + 1][0]+"\">\n";
            switch (temp[i+1][1]){
            case "CHAR":
                results+="<xs:simpleType>\n";
                results+="<xs:restriction base=\"xs:string\">\n";
                results+="<xs:length value=\""+temp[i+1][2]+"\"/>\n";
                results+="<xs:restriction>\n";
                results+="</xs:simpleType>\n";
                results+="</xsd:element>\n";
                break;
            case "VARCHAR2":
            case "VARCHAR":
            case "CLOB":
                results+="<xs:simpleType>\n";
                results+="<xs:restriction base=\"xs:string\">\n";
                results+="<xs:maxLength value=\""+temp[i+1][2]+"\"/>\n";
                results+="</xs:restriction>\n";
                results+="</xs:simpleType>";
                results+="</xsd:element>\n";
                break;
            case "BLOB":
            case "BIT":
            case "BIT VARYING":
                results+="<xs:simpleType>\n";
                results+="<xs:restriction base=\"xs:string\">\n";
                results+="<xs:encoding value=\"base64\"/>\n";
                results+="<xs:maxLength value=\""+temp[i+1][2]+"\"/>\n";
                results+="</xs:restriction>\n";
                results+="</xs:simpleType>";
                results+="</xsd:element>\n";
            case "INTEGER":
                results+="<xs:simpleType>\n";
                results+="<xs:restriction base=\"xs:integer\">\n";
                results+="<xsd:maxInclusive value=\"2157483647\"/>\n";
                results+="<xsd:minInclusive value=\"-2157483648\"/>\n";
                results+="</xs:restriction>\n";
                results+="</xs:simpleType>";
                results+="</xsd:element>\n";
            case "SMALLINT":
                results+="<xs:simpleType>\n";
                results+="<xs:restriction base=\"xs:integer\">\n";
                results+="<xsd:maxInclusive value=\"32767\"/>\n";
                results+="<xsd:minInclusive value=\"-32767\"/>\n";
                results+="</xs:restriction>\n";
                results+="</xs:simpleType>";
                results+="</xsd:element>\n";
        }
    }
    public static int getChildrenCount(String table)
    {
        int childrenCount = 0;
        String query = "select count(c.type_name) from user_types c, user_types p where (p.type_name='"
                + table + "' and c.supertype_name=p.type_name)";
        String temp = getOutput(query)[1][0];
        childrenCount = Integer.valueOf(temp).intValue();
```

```
        return childrenCount;
    }

    public static String[] getChildTable(String table)
    {
        String query = "select c.type_name from user_types c, user_types p where (p.type_name='"
                + table + "' and c.supertype_name=p.type_name)";
        String temp[][] = getOutput(query);
        String tableNames[] = new String[temp.length - 1];
        for(int i = 0; i < temp.length - 1; i++)
        {
            tableNames[i] = temp[i + 1][0];
        }
        return tableNames;
    }
    public static String tableOrNot(String table)
    {
      String query = "select table_name from user_object_tables where table_type='"+table+"'";
        String temp[][] = getOutput(query);
        if (temp.length<2)
            return "";
        else
            return temp[1][0];
    }
    public static String rootTableOrNot(String table)
    {
      String query = "select table_name from user_object_tables where table_type='"+table+"'";
        String temp[][] = getOutput(query);
        if (temp.length<2)
            return "";
        else
            return temp[1][0];
    }
...
```

The below lists some of the generated XML schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Shixin Zhang (MIT) -->
<!-- (c)Copyright 2001-2003 by MIT (Massachusetts Institute of Technology) -->
<!-- All rights reserved. -->
<!-- This document may contain the copyrights originated by others. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xs:element name="EXPERIBASE_T"><xs:complexType><xs:sequence>
  <xs:element name="AP_T"><xs:complexType><xs:sequence>
    <xs:element name="ADMIN_T"><xs:complexType><xs:sequence>
      <xs:element name="DATE_CREATED" type="timestamp"/>
      <xs:element name="DATE_MODIFIED" type="timestamp"/>
      <xs:element name="ADMIN_UID" type="varchar2"/>
      <xs:element name="CREATED_BY" type="admin_t"/>
      <xs:element name="MODIFIED_BY" type="admin_t"/>
      <xs:element name="ADMIN_ATTR" type="ap_t"/>
      <xs:element name="TABLENAME" type="varchar2"/>
    </xs:sequence></xs:complexType></xs:element>
    <xs:element name="MIT_T"><xs:complexType><xs:sequence>
      <xs:element name="LAB_T"><xs:complexType><xs:sequence>
        <xs:element name="DATE_CREATED" type="timestamp"/>
        <xs:element name="DATE_MODIFIED" type="timestamp"/>
        <xs:element name="NAME" type="varchar2"/>
        <xs:element name="ORGANIZATION" type="varchar2"/>
        <xs:element name="ACRONYM" type="varchar2"/>
        <xs:element name="ADDRESS" type="varchar2"/>
        <xs:element name="DESCRIPTION" type="varchar2"/>
        <xs:element name="CONTACTPERSON" type="admin_t"/>
```

```
      <xs:element name="CREATED_BY" type="admin_t"/>
      <xs:element name="MODIFIED_BY" type="admin_t"/>
    </xs:sequence></xs:complexType></xs:element>
    <xs:element name="PERSON_T"><xs:complexType><xs:sequence>
      <xs:element name="DATE_CREATED" type="timestamp"/>
      <xs:element name="DATE_MODIFIED" type="timestamp"/>
      <xs:element name="ADMIN_UID" type="varchar2"/>
      <xs:element name="TITLE" type="varchar2"/>
      <xs:element name="FIRSTNAME" type="varchar2"/>
      <xs:element name="MIDDLENAME" type="varchar2"/>
      <xs:element name="LASTNAME" type="varchar2"/>
      <xs:element name="SUFFIX" type="varchar2"/>
      <xs:element name="POSITIONTITLE" type="varchar2"/>
      <xs:element name="USERNAME" type="varchar2"/>
      <xs:element name="PASSWORD" type="varchar2"/>
      <xs:element name="USERSTATUS" type="char"/>
      <xs:element name="CREATED_BY" type="varchar2"/>
      <xs:element name="MODIFIED_BY" type="varchar2"/>
    </xs:sequence></xs:complexType></xs:element>
  </xs:sequence></xs:complexType></xs:element>
      ...
```

Once we have the generated XML schema, we can revise it manually and make it better, like:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Shixin Zhang (MIT) -->
<!-- (c)Copyright 2001-2003 by MIT (Massachusetts Institute of Technology) -->
<!-- All rights reserved. -->
<!-- This document may contain the copyrights originated by others. -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="ExperiBase">
  <xs:annotation>
   <xs:documentation>ExperiBase is a root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="Administration_Package" minOccurs="0"/>
    <xs:element ref="Project_Package" minOccurs="0"/>
    <xs:element ref="Sample_Package" minOccurs="0"/>
    <xs:element ref="Experiment_Package" minOccurs="0"/>
    <xs:element ref="HighLevelAnalysis_Package" minOccurs="0"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="Administration_Package">
  <xs:annotation>
   <xs:documentation>Administration package describes the information about person and
laboratory</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="User" maxOccurs="unbounded"/>
    <xs:element ref="Lab" minOccurs="0" maxOccurs="unbounded"/>
    <xs:choice minOccurs="0">
     <xs:element name="SMD">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="DBuser" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
           <xs:element name="Admin_uid" type="xs:ID" id="Admin_uid"/>
          </xs:sequence>
```

```
            </xs:complexType>
          </xs:element>
...
...
</xs:schema>
```

The example XML document to describe the data of a FACS experiment is listed below (only a portion). The XML is generated automatically by my Java code and the data is loaded from the ExperiBase database.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v2004 rel. 2 U (http://www.xmlspy.com) by H Steven Wiley (Pacific Northwest National
Laboratory) -->
<!-- edited by Shixin Zhang (MIT) -->
<!-- (c)Copyright 2001-2003 by MIT (Massachusetts Institute of Technology) -->
<!-- All rights reserved. -->
<!-- This document may contain the copyrights originated by others. -->
<EXPERIBASE_T>
 <EP_T>
  <EPD_T>
   <FC_EPD_T>
    <FACS_EPD_T>
     <FC_EXPTDESCR_T>
      <ROWID>AAAI5RAAJAAAD7uAAA</ROWID>
      <DATE_CREATED>2003-8-25.11.38. 29. 0</DATE_CREATED>
      <DATE_MODIFIED>2003-8-25.11.38. 29. 0</DATE_MODIFIED>
      <CREATED_BY>null</CREATED_BY>
      <MODIFIED_BY>null</MODIFIED_BY>
      <EXPERIMENT_UID>ICMIT_FC_experiment_25-AUG-03 11.38.29.374564000 AM -
04:00</EXPERIMENT_UID>
      <EXPTNAME>A titration experiment to determine the concentration of the anti-CD8-FITC reagent that
achieves saturating staining</EXPTNAME>
      <DESCRIPTION>The goal of this titration experiment is to determine the concentration of the anti-CD8-
FITC reagent that achieves saturating staining, i.e., maximal staining of the CD8 T cells under defined conditions
(20 minutes at room temperature). The data analysis consists of identifying the subpopulation of cells that are
stained with anti-CD8 antibody and calculating a median fluorescence intensity statistic on this subpopulation. The
computed statistics will be used to determine the concentration of anti-CD8-FITC reagent to use in subsequent
experiments.</DESCRIPTION>
      <CONTACTPERSON>null</CONTACTPERSON>
      <EXPTDATE>2001-06-29 00:00:00.0</EXPTDATE>
      <PROTOCOL>null</PROTOCOL>
     </FC_EXPTDESCR_T>
    </FACS_EPD_T>
   </FC_EPD_T>
  </EPD_T>
  <EPRD_T>
   <FC_EPRD_T>
    <FACS_EPRD_T>
     <FCS_DESC_T>
      <ROWID>AAAI5gAAJAAAD8uAAA</ROWID>
      <DATE_CREATED>2003-8-25.15.59. 32. 0</DATE_CREATED>
      <DATE_MODIFIED>2003-8-25.15.59. 32. 0</DATE_MODIFIED>
      <CREATED_BY>null</CREATED_BY>
      <MODIFIED_BY>null</MODIFIED_BY>
      <EXPERIMENT_UID>ICMIT_FC_experiment_25-AUG-03 11.38.29.374564000 AM -
04:00</EXPERIMENT_UID>
      <RAWDATA_ID>Sample 1 (Control)</RAWDATA_ID>
      <SAMPLEID>oracle.sql.REF@19192699</SAMPLEID>
      <RAWDATADESC>null</RAWDATADESC>
      <NUM_PARAMETERS>3</NUM_PARAMETERS>
      <NUM_EVENTS>10000</NUM_EVENTS>
      <ACQUISITION_DATE>2001-6-29.10.23. 21. 0</ACQUISITION_DATE>
     </FCS_DESC_T>
```

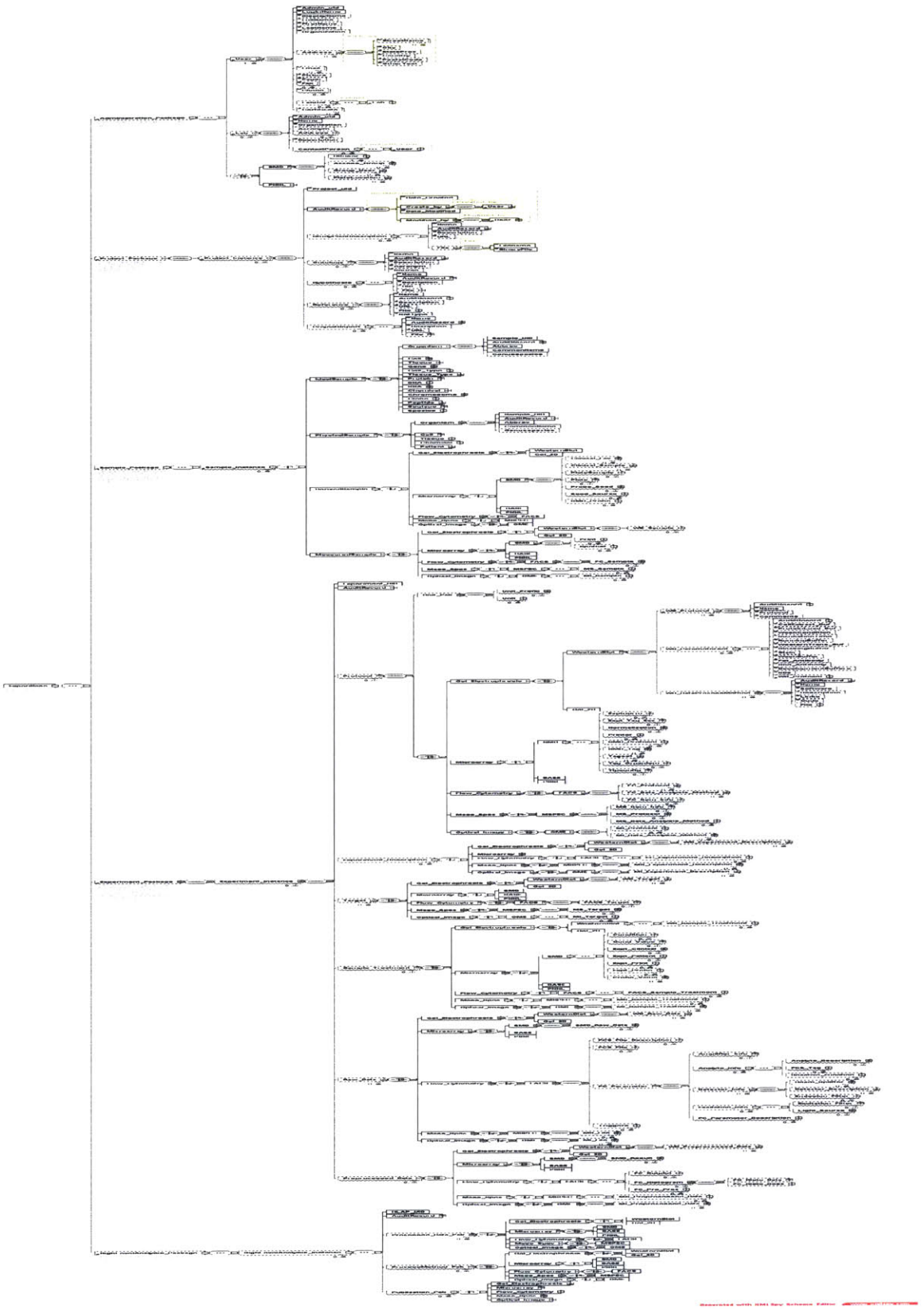


Figure 67 A portion of the ExperiBase XML schema

7.2 Web Services

Using the Java web service development package 1.3 from SUN.com, I have developed three web services for our ExperiBase:

web_service1: String query(String sqlcode);

web_service2: String queryWithTableName(String tablename, String sqlcode);

web_service3: String queryByUID(String UID,String packageName).

All returns of the services are XML documents using the ExperiBase XML schema. These web services allow other systems to integrate ours very easily, like the architecture presented by Mike Niemi (we have discussed it in the Chapter 2). The web address is http://schiele.mit.edu:8090/ExperiBase_service-jaxrpc/ExperiBase_service.

The WSDL (Web Service Description Language) is listed here:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:Foo"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
name="ExperiBaseService" targetNamespace="urn:Foo">
    <types/>
    <message name="ExperiBaseIF_query">
        <part name="String_1" type="xsd:string"/>
    </message>
    <message name="ExperiBaseIF_queryResponse">
        <part name="result" type="xsd:string"/>
    </message>
    <message name="ExperiBaseIF_queryByUID">
        <part name="String_1" type="xsd:string"/>
        <part name="String_2" type="xsd:string"/>
    </message>
    <message name="ExperiBaseIF_queryByUIDResponse">
        <part name="result" type="xsd:string"/>
    </message>
    <message name="ExperiBaseIF_queryWithTableName">
        <part name="String_1" type="xsd:string"/>
        <part name="String_2" type="xsd:string"/>
    </message>
    <message name="ExperiBaseIF_queryWithTableNameResponse">
        <part name="result" type="xsd:string"/>
    </message>
    <portType name="ExperiBaseIF">
        <operation name="query" parameterOrder="String_1">
            <input message="tns:ExperiBaseIF_query"/>
            <output message="tns:ExperiBaseIF_queryResponse"/>
        </operation>
        <operation name="queryByUID" parameterOrder="String_1 String_2">
            <input message="tns:ExperiBaseIF_queryByUID"/>
            <output message="tns:ExperiBaseIF_queryByUIDResponse"/>
        </operation>
        <operation name="queryWithTableName" parameterOrder="String_1 String_2">
            <input message="tns:ExperiBaseIF_queryWithTableName"/>
            <output message="tns:ExperiBaseIF_queryWithTableNameResponse"/>
        </operation>
    </portType>
    <binding name="ExperiBaseIFBinding" type="tns:ExperiBaseIF">
        <operation name="query">
            <input>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </input>
            <output>
```

```
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </output>
            <soap:operation soapAction=""/>
        </operation>
        <operation name="queryByUID">
            <input>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </input>
            <output>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </output>
            <soap:operation soapAction=""/>
        </operation>
        <operation name="queryWithTableName">
            <input>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </input>
            <output>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="urn:Foo"/>
            </output>
            <soap:operation soapAction=""/>
        </operation>
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    </binding>
    <service name="ExperiBaseService">
        <port name="ExperiBaseIFPort" binding="tns:ExperiBaseIFBinding">
            <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
location="http://schiele.mit.edu:8090/ExperiBase_service-jaxrpc/ExperiBase_service"/>
        </port>
    </service>
</definitions>
```

The below code is the Java interface of the web services:

```
package ExperiBase_service;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ExperiBaseIF extends Remote {
   public String query(String sqlcode) throws RemoteException;
   public String queryWithTableName(String tablename,String sqlcode) throws RemoteException;
   public String queryByUID(String uid,String pak) throws RemoteException;
}
```

The way to use the web services is very simple. The below gives out an example JSP page calling the web services:

```
<%@ page import="javax.xml.rpc.Stub,javax.naming.*,ExperiBaseService_webclient.*" %>
<%
   String resp = null;
   try {
     Stub stub = (Stub)(new ExperiBaseService_Impl().getExperiBaseIFPort());
     stub._setProperty(javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
"http://schiele.mit.edu:8090/ExperiBase_service-jaxrpc/ExperiBase_service");
     ExperiBaseIF hello = (ExperiBaseIF)stub;
     resp = hello.query(request.getParameter("sqlcode"));

   } catch (Exception ex) {
```

```
    resp = ex.toString();
  }
%>
<h2><font color="black"><%=resp%>!</font></h2>
```

7.3 Database Federation

Since the ExperiBase XML schema has been generated from the generic IOD which we have developed for a variety of experimental methods, it can certainly describe any kind of data from those experiments. It is a common XML schema. In the Chapter 2, we have introduced that Professor Dewey's group has presented a ClassMapper idea to implement database federation. But the ClassMapper is never actually implemented before this thesis. Here I present a new definition of the ClassMapper: "ClassMapper is a function translating heterogeneous database schemas to a common XML schema". We can use JDBC and Java classes to acquire database schemas from databases and then translate them to the ExperiBase XML schema automatically. The below diagram illustrates a new way to federate individual biological experimental databases together. First, a client browser sends a query request to the individual web services; the services call JDBC classes and ClassMappers to query against the individual databases; export the data in the common ExperiBase XML format; then the output XML messages are merged in the client browser. This step makes data federated. Since the XML messages use the same schema, the merge process is simple. Finally, we use XSLT[165] (XML Stylesheet Language Transforms) to display the merged XML as tables. XPath[166] (XML Path Language) or XQuery[167] (XML Query language) is used to search further interested data in the merged XML. The merged XML can be also stored in a local database.

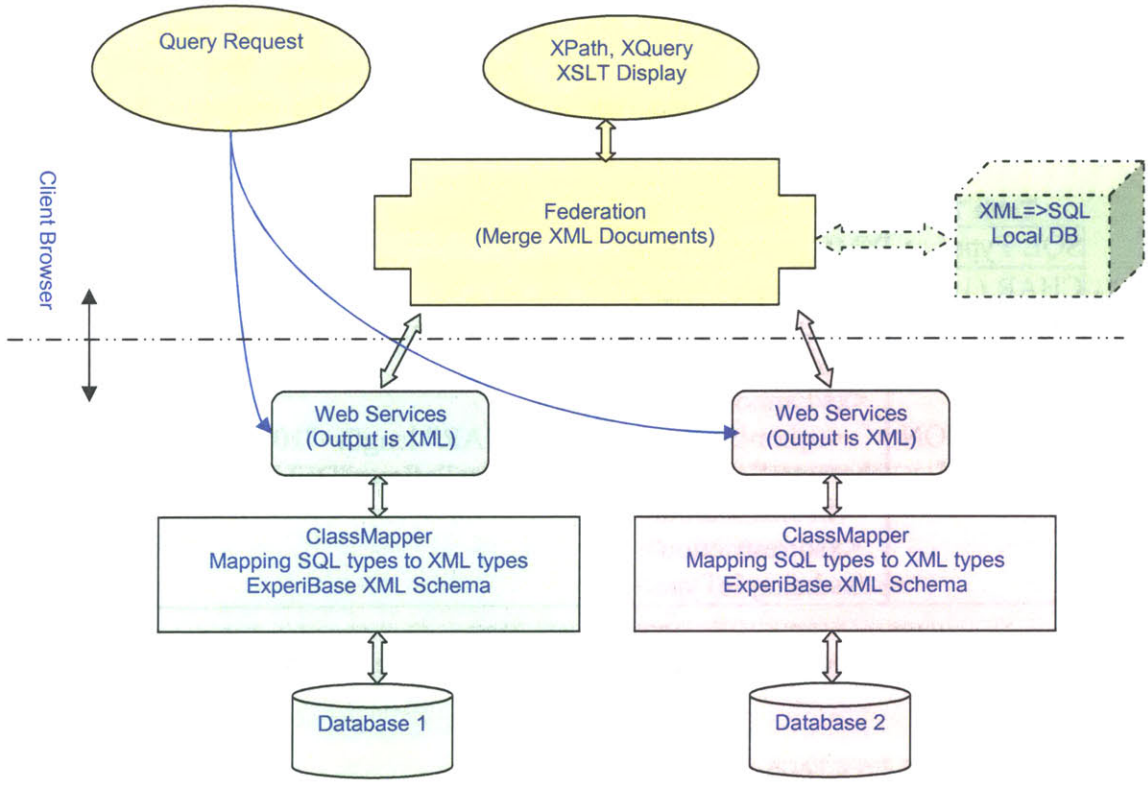


Figure 68 A new way of database federation

There are three issues which we need to consider when implementing the above architecture:

a. SQL type mappings between a variety of DBMSs

We know from the Chapter 4 that different vendors have different implementations of SQL, such as, the examples shown in the below table. Therefore, ClassMapper has to have a function to support the mappings.

Table 6 SQL type mapping
(ref: http://www.solarmetric.com/Software/Documentation/latest/docs/sql_types.html)

	STRING	CLOB	LENSTRING	BOOLEAN	BYTE
MySQL	VARCHAR(255)	TEXT	VARCHAR({0})	SMALLINT	SMALLINT
Hypersonic SQL	VARCHAR(255)	LONGVARCHAR	VARCHAR({0})	SMALLINT	SMALLINT
DB2	VARCHAR(255)	CLOB(1M)	VARCHAR({0})	SMALLINT	SMALLINT
Generic	VARCHAR(255)	CLOB	VARCHAR({0})	TINYINT	SMALLINT
InstantDB	VARCHAR(255)	TEXT	VARCHAR({0})	BYTE	BYTE
Oracle	VARCHAR2(255)	CLOB	VARCHAR2({0})	NUMBER(1)	SMALLINT
Postgres	VARCHAR(255)	TEXT	VARCHAR({0})	INT2	INT2
SQLServer	VARCHAR(255)	TEXT	VARCHAR({0})	SMALLINT	SMALLINT

b. Mapping SQL types to XML types

There are ISO standards about the mapping SQL types to XML types (the reference is "ANSI NCITS H2-2001-008, Mapping SQL types to XML types – an overview" written by Fred Zemke and Ashok Malhotra). We can use their recommendations to design the ExperiBase XML, such as the example shown in the below table.

Table 7 An example of mapping SQL types to XML types (Ref: Fred Zemke, et al, 2001)

SQL Type	XML Schema Type
CHAR (10) CHARACTER SET LATIN1 COLLATION DEUTSCH	<xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="10"/> <xsd:annotation> <sqlxml:sqltype name="CHAR" length="10" characterSetName="LATIN1" collation="DEUTSCH"/> </xsd:annotation> <xsd:restriction> </xsd:simpleType>

Therefore, the below SQL can be translated into the below XML.

- SQL:
  CREATE TYPE FACS_Str_t UNDER FACS_EPST_t
  (Experiment_UID VARCHAR2(128),

```
label VARCHAR2(32),
Sample REF cell_t,
treatment_name VARCHAR2(32),
material REF chemical_t,
dose real,
dose_unit_prefix REF unit_prefix_t,
dose_unit REF unit_t,
duration real,
duration_unit_pref REF unit_prefix_t,
duration_unit REF unit_t,
temperature real,
temperature_unit_p REF unit_prefix_t,
temperature_unit REF unit_t,
treatment_date timestamp,
description VARCHAR2(240)
)
NOT FINAL;
```

- XML

```
<xs:element name="FACS_STR_T">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="DATE_CREATED" type="timestamp"/>
   <xs:element name="DATE_MODIFIED" type="timestamp"/>
   <xs:element name="CREATED_BY" type="admin_t"/>
   <xs:element name="MODIFIED_BY" type="admin_t"/>
   <xs:element name="EXPERIMENT_UID" type="varchar2"/>
   <xs:element name="LABEL" type="varchar2"/>
   <xs:element name="SAMPLE" type="cell_t"/>
   <xs:element name="TREATMENT_NAME" type="varchar2"/>
   <xs:element name="MATERIAL" type="chemical_t"/>
   <xs:element name="DOSE" type="real"/>
   <xs:element name="DOSE_UNIT_PREFIX" type="unit_prefix_t"/>
   <xs:element name="DOSE_UNIT" type="unit_t"/>
   <xs:element name="DURATION" type="real"/>
   <xs:element name="DURATION_UNIT_PREF" type="unit_prefix_t"/>
   <xs:element name="DURATION_UNIT" type="unit_t"/>
   <xs:element name="TEMPERATURE" type="real"/>
   <xs:element name="TEMPERATURE_UNIT_P" type="unit_prefix_t"/>
   <xs:element name="TEMPERATURE_UNIT" type="unit_t"/>
   <xs:element name="TREATMENT_DATE" type="timestamp"/>
   <xs:element name="DESCRIPTION" type="varchar2"/>
  </xs:sequence>
 </xs:complexType>
 </xs:element>
```

A blob can be translated as a base64 data type:

```
<xsd:simpleType>
<xsd:restriction base="binary">
<xsd:encoding value="base64"/>
<xsd:maxLength value="1334"/>
</xsd:restriction>
</xsd:simpleType>
```

c. Accumulation and aggregation of XML data.

The accumulation and aggregation of XML can be a good research topic, such as the merge operator[168]. But since the XML messages in our case use the same ExperiBase XML schema, the merge process is pretty simple. It just combines the same nodes from each XML message together.

For example, one XML message generated from our Oracle database is:

```
<?xml version="1.0" encoding="UTF-8"?><!-- edited by Shixin Zhang (MIT) --><!-- (c)Copyright 2001-2003 by MIT
(Massachusetts Institute of Technology) --><!-- All rights reserved. --><!-- This document may contain the
copyrights originated by others. --><EXPERIBASE_T><SP_T><PSP_T><CELL_T><DATE_CREATED>2003-8-
27.10.58. 28. 0</DATE_CREATED>
<DATE_MODIFIED>2003-8-27.10.58. 28. 0</DATE_MODIFIED>
<CREATED_BY>null</CREATED_BY>
<MODIFIED_BY>null</MODIFIED_BY>
<SAMPLE_UID>ICMIT_XXX_Cell_27-AUG-03 10.58.27.962015000 AM -04:00</SAMPLE_UID>
<ABBREV>null</ABBREV>
<COMMONNAME>HT-29 (wt)</COMMONNAME>
<GENUSSPECIES>Human</GENUSSPECIES>
<TYPE>null</TYPE>
<SOURCE>ATCC</SOURCE>
<LABEL>HT-29 (wt)</LABEL>
<CONTENT>Human colorectal adenocarcinoma</CONTENT>
</CELL_T></PSP_T></SP_T></EXPERIBASE_T>
```

Another XML message generated from our DB2 database is:

```
<?xml version="1.0" encoding="UTF-8"?><!-- edited by Shixin Zhang (MIT) --><!-- (c)Copyright 2001-2003 by MIT
(Massachusetts Institute of Technology) --><!-- All rights reserved. --><!-- This document may contain the
copyrights originated by others. --
><EXPERIBASE_T><SP_T><PSP_T><CELL_T><OID>20030513232056337309000000</OID>
<DATE_CREATED>2003-05-13 19:20:56.33718</DATE_CREATED>
<CREATED_BY>null</CREATED_BY>
<DATE_MODIFIED>2003-05-13 19:20:56.33718</DATE_MODIFIED>
<MODIFIED_BY>null</MODIFIED_BY>
<SAMPLE_UID>ICMIT_XXX_Cell_2003-05-13-23.20.56.337321</SAMPLE_UID>
<ABBREV>null</ABBREV>
<COMMONNAME>HT-29 (wt)</COMMONNAME>
<GENUSSPECIES>Human</GENUSSPECIES>
<TYPE>null</TYPE>
<SOURCE>ATCC</SOURCE>
<LABEL>HT-29 (wt)</LABEL>
<CONTENT>Human colorectal adenocarcinoma</CONTENT>
</CELL_T></PSP_T></SP_T></EXPERIBASE_T>
```

Then the merged XML is:

```
<?xml version="1.0" encoding="UTF-8"?><!-- edited by Shixin Zhang (MIT) --><!-- (c)Copyright 2001-2003 by MIT
(Massachusetts Institute of Technology) --><!-- All rights reserved. --><!-- This document may contain the
copyrights originated by others. --><EXPERIBASE_T><SP_T><PSP_T><CELL_T><DATE_CREATED>2003-8-
27.10.58. 28. 0</DATE_CREATED>
<DATE_MODIFIED>2003-8-27.10.58. 28. 0</DATE_MODIFIED>
<CREATED_BY>null</CREATED_BY>
<MODIFIED_BY>null</MODIFIED_BY>
<SAMPLE_UID>ICMIT_XXX_Cell_27-AUG-03 10.58.27.962015000 AM -04:00</SAMPLE_UID>
<ABBREV>null</ABBREV>
<COMMONNAME>HT-29 (wt)</COMMONNAME>
<GENUSSPECIES>Human</GENUSSPECIES>
<TYPE>null</TYPE>
<SOURCE>ATCC</SOURCE>
<LABEL>HT-29 (wt)</LABEL>
<CONTENT>Human colorectal adenocarcinoma</CONTENT>
</CELL_T></PSP_T></SP_T><SP_T><PSP_T><CELL_T><OID>20030513232056337309000000</OID>
<DATE_CREATED>2003-05-13 19:20:56.33718</DATE_CREATED>
<CREATED_BY>null</CREATED_BY>
<DATE_MODIFIED>2003-05-13 19:20:56.33718</DATE_MODIFIED>
<MODIFIED_BY>null</MODIFIED_BY>
<SAMPLE_UID>ICMIT_XXX_Cell_2003-05-13-23.20.56.337321</SAMPLE_UID>
<ABBREV>null</ABBREV>
```

```
<COMMONNAME>HT-29 (wt)</COMMONNAME>
<GENUSSPECIES>Human</GENUSSPECIES>
<TYPE>null</TYPE>
<SOURCE>ATCC</SOURCE>
<LABEL>HT-29 (wt)</LABEL>
<CONTENT>Human colorectal adenocarcinoma</CONTENT>
</CELL_T></PSP_T></SP_T></EXPERIBASE_T>
```

The query web pages in the previous database implementation section used the architecture of Figure 68.

7.4 Summary

This chapter has discussed the ExperiBase XML schema converted from the previous information object definition and the object-relational database schema. The XML schema can be generated automatically from the database using my Java classes. Three web services developed for the project are introduced. This chapter has presented a new way of database federation that uses the common ExperiBase XML schema as a ClassMapper definition, which translates the heterogeneous database schemas into the common ExperiBase XML schema; and then exports data from individual databases in the ExperiBase XML format. The output XML messages are sent back to the client browser by the designed web services and merged in the client side to get the data federated.

Chapter 8. Discussion and Future Direction

8.1 Summary

This thesis has presented the information object definitions for gel electrophoresis, flow cytometry, microarray, mass spectrometry, and microscope image, using the object-relational data model. The IODs are able to completely characterize the results of each experiment. Because the design of the IOD for each method uses a single coherent set of information object definitions, a consistent data definition strategy has been proved that can handle those five experimental modalities. A single object-relational database schema and XML schema have been implemented according to the single coherent set of information object definitions. The schemas are highly modular and easily updated. A web-based system – ExperiBase has been developed to browse, query the database and upload data. This thesis has also presented a new way of database federation – translating heterogeneous database schemas into the common ExperiBase XML schema and then merging the output XML messages to get data federated. Three web services have been provided allowing other systems to integrate ExperiBase easily.

A five package design idea is presented to support a variety of biological experimental data. The five packages are project, sample, experiment, high level analysis, and administration. The project package consists of a description of the project, contact info, ontology, reference, hypothesis, and project report. The sample package has three types, which are ideal sample (or physical sample), derived sample (can be sample set), and measured sample. The experiment package includes experimental protocol, description, sample treatment, raw data, and preprocessed data. The high level analysis records publications, or any advanced data analysis results which may use the raw data from multiple experimental methods. The administration package is used to store the information about person (user, experimenter), lab, security and any information about auditing events. The administration package and project package have their own fixed structure, which do not vary with experimental methods. The entities in the ideal sample domain are common to any biological experiments. These reflect the real situation in biological experiments. This kind of separation of information objects allows the many-to-many relations between project, sample, experiment, and high level analysis; and also supports that an experiment can have multiple protocols, multiple sample treatment steps, multiple stain steps, multiple results, and multiple data analyses. The auditing information, such as the date created / modified and contact person is recorded in every information entity.

Two general criteria were used in the IODs: the repository should contain sufficient information to allow users to recreate any of the experiments whose results are stored within it; the information stored should be organized in a manner reflecting the structure of the experimental procedures that generated it.

Since common attributes are made truly identical between the different methods, this dramatically decreases the overhead of separate and distinct classes for each method, and reserves the uniqueness for attributes that are different between the methods. Thus, at least one higher level of integration is obtained. The consistent object inheritance allows us to be able to implement variant object references. Since the underlying schema is

highly modular, the expanded query scope is extremely powerful. The grouping of information entities reflects the workflow of each experimental method. And also, since many information entities are shared with various experimental methods, we are able to query all experiments which are related to a certain piece of information (such as a cell) even though the experimental methods are different.

The IODs of Western blot and 2D gel electrophoresis are currently very complete ontology. They were summarized from a lot of gel electrophoresis experiments done by several labs and persons; and the proposed ontology for 1D and 2D gel data by Chris Taylor et al, and the way describing LIMS's data by CDISC and DICOM, have been used as the basis of the definitions. The ontology is ready for international standardization consideration. The ontologies proposed by CytometryML, MAGE, OME, and PEDRo have been used as the basis of the object definitions of flow cytometry, microarray, microscope images, and mass spectrometry, respectively. Their XML schemas or relational database schemas have been converted to the object-relational database schema of our design and their information entities have been reorganized, incorporated, or even extended (such as OME). Therefore, nearly a hundred percent of information described in Cytometry, MAGE, OME, and PEDRo can be recorded in our schema. The new interpretation and reorganization of information entities guarantees the exact relationships between each other in the real world. Since the structure of our object model is designed in such a way that the structure can support a variety of experimental methods, and the information may be shared in different methods, (such as sample, project), the scope of our object model is larger than any above individual standard; the applications of ours are wider than them as well. The tree and modular structure of our IODs is easier understood and better organized. The new schema converted from the Stanford Microarray Database has been tested by thousands set of microarray data from the database. Other portions of our schema have also been tested by real experimental data.

The web-based system – ExperiBase (http://experibase.mit.edu) is operational and has been installed in PNNL. (It was sent to the Center for Bioinformatics, Harvard Center for Neurodegeneration and Repair as well to be a prototype of their image database project.) The site provides the functionalities about browsing, querying database, and uploading data; also provides web services, image converting classes, and Excel Spreadsheet parser, and so on. The site has all source code download for free, including the ExperiBase XML schema converted from the object-relational database schema, JSP, SQL, and Java servlets. The XML schema is generated automatically from the database using my Java classes. The query pages in the site are using the new way of database federation that uses the common ExperiBase XML schema as a key concept of ClassMapper, which translates the heterogeneous database schemas into the common ExperiBase XML schema; and then exports data from individual databases in the ExperiBase XML format. The output XML messages are sent back to the client browser by the designed web services and merged in the client side to get the data federated.

ExperiBase is extensible: 1) allowing the users to create their own protocol/table; 2) the tree and modular structure is easily updated; 3) the middleware feature of SOAP/XML and CORBA makes the software architecture extensible and platform independent. ExperiBase is flexible – it is suitable for a variety of biological experiment methods. ExperiBase is multi-functional – can store any kinds of experimental data, such

as image, video, graph, data sheet, text file, etc. The object-relational database design supports directed, customized, and powerful expanded queries. ExperiBase is WWW accessible and has friendly user interfaces with security design.

8.2 Discussion and Future direction

This thesis has being mainly focused on developing a consistent data definition strategy and an integrated software platform capable of supporting a variety of biological experimental methods. Although the ontologies of ExperiBase have become a reference implementation of I3C Life Science Object Ontologies group, which is led by Professor Dewey, the detail attributes of each information entity still needs to be examined by reality. Developing good ontologies is difficult. It requires an appreciation of knowledge representation plus a background in the domain or access to domain experts. I greatly thank Professor Dewey for bringing the ExperiBase ontologies to the I3C LSOO group, because collaboration and curation from multiple communities and providers are needed and important for developing ontologies. The areas of bio- and chem- informatics present an interesting opportunity for this. The science and technology continues to advance rapidly. The algorithms, techniques, and data sources are constantly evolving. The ontologies of ExperiBase needs corresponsively updated as well. We are lucky that the structure of ExperiBase data model allows us to be able to extend it without any barrier. From the standpoint of interoperability and system integration, the common ExperiBase XML schema provides a way to effectively glue heterogeneous biological experimental databases together.

Being a PhD student, I only had a limited time to implement the whole ontologies I designed. The work load of the implementation of one web-based system and two databases are very heavy. Only one person, I was doing the project. So there are still many things left, such as the implementation of microarray IOD, providing the links to knowledge-based databases, like taxonomy, LocusLink, and so on. The database still needs to be further tested by a large amount of experimental data. We may need to keep adding new entities to satisfy the needs of a variety of labs and experimenters. The entities of high level analysis package may still need to be further normalized; its contents may need to be researched further. Since the web-based user interfaces are generic, it may not be specific or suitable for a particular lab's needs.

All in all, ExperiBase is an integrated software platform capable of supporting a variety of experimental methods. It provides comprehensive database features; normalizing semantic support for the different methods; web-based client services; data import and export capabilities to accommodate other data repositories; and direct support for metadata produced by analysis programs.

I hope that the ontologies of ExperiBase can bring a strong impact on the ontology development of experimental biology. I hope that the ExperiBase schema can be a prototype of formal experimental biology standards. I hope that the system can be widely used by labs. Thank you!

Bibliography

[1] Amarnath Gupta, Bertram Ludäscher, Maryann E. Martone. *Knowledge-Based Integration of Neuroscience Data Sources*, 12th Intl. Conference on Scientific and Statistical Database Management (SSDBM), Berlin, Germany, IEEE Computer Society, July 2000. http://www.sdsc.edu/~ludaesch/Paper/ssdbm00.html

[2] Alvis Brazma, et al. *Minimum Information about A Microarray Experiment (MIAME) – toward standards for microarray data.* Nature Genetics, volume 29, December 2001, pp 365-371

[3] *MicroArray and Gene Expression – MAGE*, http://www.mged.org/Workgroups/MAGE/mage.html

[4] *MAGE Object Model*, http://www.mged.org/Workgroups/MAGE/mage-om.html

[5] *MAGE-ML*, http://www.mged.org/Workgroups/MAGE/mage-ml.html

[6] Chris F Taylor, et al. *A systematic approach to modelling capturing and disseminating proteomics experimental data.* Nature Biotechnology, March 2003 Volume 21 Number 3 pp 247 – 254. http://pedro.man.ac.uk/home.shtml

[7] Susan Bassion. *Toward a Laboratory Data Interchange Standard for Clinical Trials Clinical Chemistry*, 2002 48 (12): 2290-2292.

[8] Susan Bassion. *Standardizing Laboratory Data Interchange in Clinical Trials.* JALA, Volume 7, No. 5, October/November 2002

[9] R.C. Leif, S.H. Leif, S.B. Leif. *CytometryML, An XML Format based on DICOM for Analytical Cytology Data*, accepted for publication Cytometry (2003). http://www.newportinstruments.com/cytometryml/cytometryml.html

[10] *Open Microscopy Environment*, http://www.openmicroscopy.org/

[11] *Biological Timing Online Science Experiment.* http://www.cbt.virginia.edu/Olh/

[12] C. Forbes Dewey, Jr., Shixin Zhang, etc. *Information technology for multimedia bioengineering information.* Presentation in DARPA meeting, April 5, 2001, MIT.

[13] *Electronic, Fluorescent Kinase Assays for High-Throughput Screening.* http://www.nanogen.com/technology/kinase.asp

[14] *Human Genome Project.* http://www.ornl.gov/hgmis/

[15] Asthagiri, A., A.F. Horwitz, and D.A. Lauffenburger. *A Rapid and Sensitive Quantitative Kinase Activity Assay Using a Convenient 96-well Format.* Analyt. Biochem. 269: 342-347 (1999).

[16] John Albeck, Sorger Lab, MIT. *Quantitative Monitoring of Apoptotic and Early Signaling Responses to TNF and Insulin.* Symposium on Apoptosis Signaling Networks in Human Cells, MIT, January 17th, 2002.

[17] Ulrik B. Nielsen. *Monitoring Signal Transduction Networks with Antibody Microarrays.* Symposium on Apoptosis Signaling Networks in Human Cells, MIT, January 17th, 2002.

[18] Suzanne Gaudet, et al, Sorger Lab, Department of Biology, MIT. *Using Quantitative Immunoblots to Assay the Effects of Insulin on Signaling Networks in TNF-treated HT-29 Cells – Cellular Decision Processes.* Symposium on Apoptosis Signaling Networks in Human Cells, MIT, January 17th, 2002.

[19] Birgit Schoeberl. *A Mathematical Model of Caspase Activation in HT29 Cells*, Symposium on Apoptosis Signaling Networks in Human Cells, MIT, January 17th, 2002.

[20] Joseph L. Greenstein, Richard Wu, Sunny Po, Gordon F. Tomaselli, and Raimond L. Winslow. *Role of the Calcium-Independent Transient Outward Current Ito1 in Shaping Action Potential Morphology and Duration Online Data Supplement.* http://nsr.bioeng.washington.edu

[21] McGrath, J.L., Osborn, E., Hartwig, J.H. and Dewey, C.F. Jr.. *A Mechanistic Model of the Actin Cycle in Cells.* In preparation for Biophys. J. Page 1 of 22 11/29/99

[22] *Stanford Microarray Database.* http://www.dnachip.org/

[23] *ArrayExpress at the EBI*, http://www.ebi.ac.uk/arrayexpress/

[24] *Microarray Gene Expression Data Society - MGED Society*, http://www.mged.org/

[25] Ron Edgar, Michael Domrachev and Alex E. Lash. *Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.* Nucleic Acids Research, 2002, Vol. 30, No. 1, pp 207-210. http://www.ncbi.nlm.nih.gov/geo/

[26] *Saccharomyces Genome Database*, http://www.google.com/search?sourceid=navclient&q=SGD

[27] Ball,C.A., Dolinski,K., Dwight,S.S., Harris,M.A., Issel-Tarver,L., Kasarskis,A., Scafe,C.R., Sherlock,G., Binkley,G., Jin,H. et al. (2000) Nucleic Acids Res., 28, 77-80

[28] Costanzo,M.C., Hogan,J.D., Cusick,M.E., Davis,B.P., Fancher,A.M., Hodges,P.E., Kondu,P., Lengieza,C., Lew-Smith,J.E., Lingner,C. et al. *YPD, PombePD and WormPD: model organism volumes of the BioKnowledge library, an integrated resource for protein information.* (2000) Nucleic Acids Res., 28, 73-76

[29] Wheeler,D.L., Chappey,C., Lash,A.E., Leipe,D.D., Madden,T.L., Schuler,G.D., Tatusova,T.A. and Rapp,B.A. (2000) Nucleic Acids Res., 28, 10-14

[30] Boguski,M.S., Lowe,T.M. and Tolstoshev,C.M. (1993) Nature Genet., 4, 332-333

[31] Bairoch,A. and Apweiler,R. (2000) Nucleic Acids Res., 28, 45-48

[32] *DBI module*, http://search.cpan.org/search?module=DBI, http://search.cpan.org/author/TIMB/DBI-1.30/DBI.pm

[33] *GD module, Create GIFs from within Perl scripts.* http://stein.cshl.org/WWW/software/GD/

[34] *GD Graphics Library, An ANSI C library for the dynamic creation of images.* http://www.boutell.com/gd/

[35] *GenePix scanner*, http://www.axon.com/GN_GenePixSoftware.html

[36] *ScanAlyze, Microarray Image Analysis*, http://rana.lbl.gov/EisenSoftware.htm

[37] *USC Brain Project Time Series Database (TSDB),* http://www-hbp.usc.edu/Thrusts/tsdb.htm

[38] *Java Applet for Data Entry*, http://www-hbp.usc.edu/Projects/jade.htm

[39] *Neuroscience Federated Databases*, http://www.npaci.edu/DICE/Neuro/SC2000/

[40] Victor M. Markowitz. *Characterizing Heterogeneous Molecular Biology Database Systems.* http://gizmo.lbl.gov/DM_TOOLS/OPM/JCB/JCB.html

[41] Susan B. Davidson, Jonathan Crabtree, Brian Brunk, Jonathan Schug, Val Tannen, Chris Overton and Chris Stoeckert. *K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources.* IBM SYTEMS JOURNAL Vol 40 No 2, 2001, pp 512-531

[42] Kenneth Baclawski, Robert Futrelle, et al. *Database Techniques for Biological Materials & Methods.* http://www.ccs.neu.edu/home/kenb/sdb/ismb/mm.html, Feb. 1 1995

[43] A. Brazma, Editorial. *On the importance of standardisation in life sciences.* Bioinformatics, Vol 17 (2001), 113-114.

[44] *BSML, Bioinformatic Sequence Markup Language*, http://www.bsml.org/

[45] *What is CellML?* http://www.cellml.org/public/about/what_is_cellml.html

[46] *The Systems Biology Markup Language (SBML)*, www.cds.caltech.edu/erato/sbml/docs/index.html

[47] *MoDL*, http://violet.csa.iisc.ernet.in/~modl/

[48] *Protein Extensible Markup Language (PROXIML)*, http://www.cse.ucsc.edu/~douglas/proximl/

[49] *BIOpolymer Markup Language (BIOML)* , http://www.bioml.com/BIOML/

[50] *Gene Expression Markup Language (GEML),* http://www.rosettabio.com/products/conductor/geml/default.htm

[51] *GeneX Gene Expression Markup Language (GeneXML)*, http://www.ncgr.org/genex/genexml.html

[52] *Microarray Markup Language (MAML)*, http://www.oasis-open.org/cover/maml.html

[53] *XML for Multiple Sequence Alignments (MSAML)*, http://maggie.cbr.nrc.ca/~gordonp/xml/MSAML/

[54] *Genome Annotation Markup Elements (Game)*, http://www.bioxml.org/Projects/game/game0.1.html

[55] *Chemical Markup Language*, http://www.xml-cml.org/

[56] *StarDOM - Transforming Scientific Data into XML*, http://www.oasis-open.org/cover/stardom.html

[57] *Mathematical Markup Language (MathML)*, http://www.w3.org/TR/REC-MathML/

[58] *OpenMath*, http://www.openmath.org/

[59] *Resource Description Framework (RDF),* http://www.w3.org/RDF/

[60] Uche Oqbuji, *Using RDF with SOAP*, http://www-106.ibm.com/developerworks/webservices/library/ws-soaprdf/

[61] Mike Nieme., *Strawman for October I3C Hackathon in Hinxton -- Using ontologies to describe biological services and data*, http://www.i3c.org/events/Oct2003Hackathon/strawman.htm

[62] *The I3C Bio-Google Demostration*, http://www.i3c.org/bio2002/i3cdemo.pdf

[63] IBM, *DiscoveryLink – Solving the data integration dilemma for the life sciences*, http://www-3.ibm.com/solutions/lifesciences/solutions/discoverylink.html

[64] *myGrid*, http://mygrid.man.ac.uk/

[65] *Life Science Identifier*, http://www.i3c.org/wgr/ta/resources/lsid/docs/

[66] *LSID Resolver*, http://www.i3c.org/wgr/ta/resources/lsid/docs/index.htm

[67] *Taverna*, http://taverna.sourceforge.net/

[68] *Pedro tool*, http://pedrodownload.man.ac.uk/
[69] *BioMOBY*, http://www.biomoby.org/
[70] *Haystack*, http://haystack.lcs.mit.edu/
[71] *Gene Ontology (GO)*, http://www.geneontology.org/
[72] *Life Science Object Ontologies group (LSOO)*, http://www.i3c.org/wgr/lsoo/lsoo.asp
[73] *Soaplab*, http://industry.ebi.ac.uk/soaplab/
[74] *Jena*, http://www.hpl.hp.com/semweb/
[75] *I3C Registry Group*, http://www.i3c.org/wgr/reg/registry.asp
[76] *Life Sciences Identifier (LSID), draft specification for review and comment.* http://www.i3c.org/workgroups/technical_architecture/resources/lsid0/lsid-tawg-5-03-2002a.pdf
[77] Robert Kincaid, Life Science Technologies Laboratory, Agilent Technologies, Palo Alto, California, USA. *BNS: A DNS-Inspired Biomolecule Naming Service.* Poster 157B(i), ISMB 2002, 8/5/2002.
[78] Dao N, Dewey CF, Jr. *Design and Prototype of a Database for Medical Images*, (Abstract). Ann. Biomed. Eng'g 1998;26 (Suppl. 1):S-13.
[79] Dao N, Dewey CF, Jr., *Databasing Strategy for the Human Physiome,* (Abstract). Ann Biomed.Eng'g 1998;26 (Suppl. 1), S-13.
[80] Dewey, C.F. Jr., Kitney, R.I. *Creating DICOM-Enabled Clinical Systems with Robust Image-Querying Capabilities.* Proc. Towards An Electronic Patient Record '98; C. P. Waegemann, editor. 1998 May 9-15, 1998; San Antonio, TX. Medical Records Institute.
[81] Dao N, Dewey, CF Jr. *The Human Physiome as an Information Environment.* Annals of Biomedical Engineering (submitted for publication, 1999).
[82] Shixin Zhang and C. F. Dewey, Jr.. *An IIOP architecture for web-enabled physiological models.* 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Oct 25-28, 2001, Istanbul, TURKEY
[83] C. F. Dewey, Jr., B. Fu, S. Zhang, et al. *An information architecture for physiological models, clients and databases.* 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Oct 25-28, 2001, Istanbul, TURKEY
[84] B. Fu, S. Zhang, W. Chuang, and C. F. Dewey, Jr. *A database federation platform for gene chips and the human genome database.* 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Oct 25-28, 2001, Istanbul, TURKEY.
[85] B. Fu. *Design of a genetics database for gene chips and the human genome database.* M. Eng. Thesis, MIT. 2001.
[86] Ngon Dao and C. F. Dewey, Jr. *An Object-Relational Architecture for a DICOM Medical Image Archive*, http://icmit.mit.edu/pres/Nice97/sld001.htm
[87] *Apoptosis database.* http://www.apoptosis-db.org/welcome.html
[88] *GenBank and Entrez.* http://www.ncbi.nlm.nih.gov/Database/index.html
[89] *Protein Data Bank.* http://www.rcsb.org/pdb/
[90] Ronald Bourret, *XML-DBMS: Middleware for Transferring Data between XML Documents and Relational Databases*, http://www.rpbourret.com/xmldbms/
[91] Shixin Zhang, *Electronic Medical Record Systems and Databases*, http://schiele.mit.edu/sxzhang/healthcare/index.htm
[92] *ImageQuant™ for Windows™ NT*, http://www4.amershambiosciences.com/aptrix/upp01077.nsf/Content/gel_blot_software_imagequant_introduction
[93] Bairoch, A. and Apweiler, R. (2000) Nucleic Acids Res. 28, 45-48.
[94] Wheeler, D.L.,Chappery, C., Lash, A.E., Leipe, D.D., et al. (2000) Nucleic Acids Res., 28, 10-14
[95] *National Library of Medicine, PubMed.* http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?CMD=Display&DB=PubMed
[96] Ngon D. Dao. *Design and Prototype of an Object-Relational Database for Medical Images.* Mechanical Engineering Master thesis, MIT, June 1998, pp33-44
[97] William Chuang. *Design of a Genetics Database for Medical Research.* Computer Science Master thesis, MIT, May 2000, pp11-14
[98] Patrick J. McCormick. *Designing Object-Oriented Interfaces for Medical Data Repositories.* Electrical Engineering and Computer Science Master thesis, MIT, June 1999, pp6-7
[99] David R. Frick & Co., CPA. *Database Models.* http://www.frick-cpa.com/ss7/Theory_Models.asp

[100] Codd EF. *A relational model of data for large shared data banks*. Communications of the ACM. 1970; pp 13, 377-387.
[101] Turner P, Keller AM. *Reflections on object-relational applications*. OOPSLA workshop on object and relational databases: Austin, TX; 1995.
[102] W. Kim. *Object-relational database technology: a UniSQL whitepaper*. UniSQL Inc., Austin, TX. 1996
[103] Object Data Management Group. *ODMG3.0, Standard for object-relational mapping products as well as object DBMSs*, http://www.odmg.org/
[104] Michael Kifer, Georg Lausen, James Wu. *Logical Foundations of Object Oriented and Frame Based Languages*. Journal of ACM 1995, vol. 42, p. 741-843
[105] Amarnath Gupta, Bertram Ludascher, Maryann E. Martone. San Diego Supercomputer Center, UCSD. *Knowledge-Base Integration of Neuroscience Data Sources*. 12th Intl. Conference on Scientific and Statistical Database Management (SSDBM), Berlin, Germany, July 2000. IEEE Computer Society. http://citeseer.nj.nec.com/gupta00knowledgebased.html. pp 39-52
[106] *The FLORID/FloXML Project*, http://www.informatik.uni-freiburg.de/~dbis/florid/
[107] Stonebraker M, Brown P, Moore D. *Object-Relational DBMSs: Tracking The Next Great Wave*. In: Gray J, ed. Series in Data Management Systems. 2 ed. San Francisco, CA: Morgan Kaufmann; 1999:297.
[108] Stonebraker M. and Kemnitz G.. *The Postgres next-generation database-management system*. Communications of the ACM, vol. 34, no. 10, pp. 78-92, 1991
[109] Kim W. *Introduction to Object Oriented Databases*. In: Schwetman H, ed. Computer Systems. Cambridge, MA: The MIT Press; 1990:234.
[110] S. Gala and W. Kim. *Database design methodology: Converting a relational schema into an object-relational schema*. Presented at ACM Symposium on Advanced Database Systems and Their Integration, Japan, October, 1994
[111] Oracle 9i, *Application Developer's Guide – Object-Relational Features, Release 1 (9.0.1)*, June 2001, Part No. A88878-01
[112] *IBM DB2 Universal Database, Application Development Guide, Part 4. Object-Relational Programming*. IBm Corp. 2000. pp265-490
[113] Cathy Riemer, et al. A *Database of Experimental Results on Globin Gene Expression*. Genomics 53, 1998. pp325-337
[114] *SQL Readings*. http://www.wiscorp.com/SQLStandards.html
[115] *SQL 2003 standard*, http://www.wiscorp.com/sql/sql_2003_standard.zip
[116] ISO, *Data management and interchange*.
http://www.iso.ch/iso/en/stdsdevelopment/tc/tclist/TechnicalCommitteeDetailPage.TechnicalCommitteeDetail?COMMID=160
[117] *WD 9075-14 Information technology - Database Languages - SQL - Part 14: SQL/XML*.
http://www.jtc1sc32.org/sc32/jtc1sc32.nsf/e783d79e26bff9f388256b9f0061b6f5/b7ab89cadb9be5f288256d95005fcc47?OpenDocument
[118] *W3C XML schema*. http://www.w3.org/XML/Schema
[119] Fred Zemke, Ashok Malhotra. *Mapping SQL types to XML types – an overview*. ISO/IEC JTC1 /SC32 WG3:E3A-nnn ANSI NCITS H2-2001-008.
[120] Ashok Malhotra, Fred Zemke. *SQL-centric mapping between SQL types and XML types*. January 4, 2001
[121] Fred Zemke, et al. *SQL/XML candidate base document*. ISO/IEC JTC1/SC32 WG3:E3A-003, ANSI NCITS H2-2000-009
[122] L Harding. *Merging OO Design and SGML/XML Architectures*.
http://www.infoauto.com/articles/arch/rim/oo-sgml.htm, Information Assembly Automation, Inc. 1998
[123] *Gel Electrophoresis*, http://www.bergen.org/AAST/Projects/Gel/
[124] *Separation of molecules in gel electrophoresis*, http://web.utk.edu/~khughes/GEL/sld005.htm
[125] Bollag, D.M., Rozycki, M.D., and Edelstein, S.J. (1996). *Protein Methods*, Second Edition. New York: Wiley-Liss, 195-227.
[126] Brian White, MIT. Southerns, Northerns, *Westerns, & Cloning: "Molecular Searching" Techniques*.
http://esg-www.mit.edu:8001/esgbio/rdna/rdna.html
[127] *General Western Blot Protocol*. http://www.kpl.com/support/immun/Protocols/PR_1.htm
[128] *Western Blot & Immunostaining*. http://www.rndsystems.com/asp/g_sitebuilder.asp?bodyId=234

[129] Canfield, V.A. and Levenson, R. *Transmembrane organization of the Na,K-ATPase determined by epitope addition.* Biochemistry 32: 13782-13786 (1993).
[130] Dietzen DJ, Hastings WR, and Lublin DM. *Caveolin is palmitoylated on multiple cysteine residues. Palmitoylation is not necessary for localization of caveolin to caveolae.* J Biol Chem 270:6838-6842, 1995.
[131] *FluorImager*,
http://www4.amershambiosciences.com/aptrix/upp01077.nsf/Content/gel_blot_fluor_imager_introduction
[132] *ImageQuant*,
http://www4.amershambiosciences.com/aptrix/upp01077.nsf/Content/gel_blot_software_imagequant_introduction
[133] Gorg A, Obermaier C, Boguth G, Harder A, Scheibe B, Wildgruber R, Weiss W. *The current state of two-dimensional electrophoresis with immobilized pH gradients.* Electrophoresis 21, 2000, pp1037-1053. PN38143
[134] *Isoelectric focusing*, http://ntri.tamuk.edu/if/if.html
[135] Righetti, P. G. *Isoelectric focusing : theory, methodology, and applications*. New York : Elsevier Biomedical Press, IDN 0444804676, 1983
[136] *SDS-PAGE*, http://web.uct.ac.za/microbiology/sdspage.html
[137] Chris F Taylor, et al. *A systematic approach to modelling capturing and disseminating proteomics experimental data.* Nature Biotechnology, March 2003 Volume 21 Number 3 pp 247 - 254
[138] *Rules of Data Normalization*, http://www.datamodel.org/NormalizationRules.html
[139] *Flow Cytometry*, FACS Laboratory at the London Research Institute: Lincoln's Inn Fields Laboratories, http://www.icnet.uk/axp/facs/davies/Flow.html
[140] *Fluorescence Activated Cell Sorting*,
http://biology.fullerton.edu/courses/biol_426/Web/426GP2/facs.htm
[141] *FlowJo*, http://www.flowjo.com/
[142] Robert C. Leif, Suzanne B. Leif, etc, XML_Med, a Division of Newport Instruments, *CytometryML*, http://www.newportinstruments.com/cytometryml/cytometryml.html
[143] *XMLSpy*, http://www.xmlspy.com/
[144] *Flow Cytometry Standard (FCS) format*, http://www.isac-net.org/links/topics/FCS3.htm
[145] *WinMDI*, http://facs.scripps.edu/software.html
[146] *Flow Explorer*, http://wwwmc.bio.uva.nl/~hoebe/Welcome.html
[147] *JCSMR Flow Cytometer Simulator,* http://jcsmr.anu.edu.au/facslab/facsimile.html
[148] G. W. Osborne, *Flow Cytometry Software Workshop: 2000*,
http://jcsmr.anu.edu.au/facslab/analysis.html
[149] Raul C. Braylan, etc, *U.S.-Canadian Consensus Recommendations on the Immunophenotypic analysis of Hemotologic Neoplasia by Flow Cytometry: Data Reporting, Cytometry*. Communications in Clinical Cytometry 30:245-248 (1997)
[150] Robert C. Leif, Suzanne B. Leif, etc, XML_Med, a Division of Newport Instruments, *Design of an XML Format based on DICOM for Analytical Cytology Data*,
http://www.newportinstruments.com/cytometryml/cytometryml.html
[151] Andy N.D. Nguyen,M.D. *CD Marker, Flow Cytometry, Data Mining, and the World Wide Web*,
http://dpalm.med.uth.tmc.edu/faculty/bios/nguyen/DXPowerPoint/
[152] *Incyte*, http://www.incyte.com/index.shtml
[153] *Affymetrix*, http://www.affymetrix.com/
[154] Hardiman, *Overview of Microarray Technologies*, UCSD Extension,March 2003.
[155] *Microarray*, http://zdb.wehi.edu.au:8282/zf_info/glossary.html,
http://www.niehs.nih.gov/nct/glossary.htm,
http://www.genomicglossaries.com/content/instrument_tech.asp
[156] *EMBL - European Bioinformatics Institute*, http://www.mged.org/Workgroups/MAGE/mage-om.html
[157] A Brazma, P Hingamp, J Quackenbush, G Sherlock, P Spellman, C Stoeckert, J Aach, W Ansorge, C A Ball, H C Causton, T Gaasterland, P Glenisson, F C P Holstege, I F Kim, V Markowitz, J C Matese, H Parkinson, A Robinson, U Sarkans, S Schulze-Kremer, J Stewart, R Taylor, J Vilo & M Vingron *Minimum information about a microarray experiment (MIAME)—toward standards for microarray data.* Nature Genetics, vol 29 (December 2001), pp 365 - 371.
[158] *OMG model (compressed),* EMBL,
http://sourceforge.net/project/showfiles.php?group_id=16076&release_id=109998

[159] *Mass spectrometry*, www.niehs.nih.gov/nct/glossary.htm, www.dddmag.com/scripts/glossary.asp, www.genomicglossaries.com/content/instrument_tech.asp
[160] *Merriam Webster's Online Dictionary*, web site www.m-w.com (as of July 13, 2000).
[161] Shixin Zhang. *An IIOP Architecture for Physiological Models, Clients, and Databases*. MIT master thesis, 2001.6
[162] *Fasimile*, http://jcsmr.anu.edu.au/facslab/facsimile.html
[163] *Morten's JavaScript Tree Menu*, http://www.treemenu.com
[164] Jason Hunter. *com.oreilly.servlet package*, http://www.servlets.com/cos/index.html
[165] *XSL Transformations (XSLT)*. http://www.w3.org/TR/xslt
[166] *XML Path Language (XPath)*. http://www.w3.org/TR/xpath
[167] *XQuery 1.0: An XML Query Language*. http://www.w3.org/TR/xquery/
[168] *Accumulation and Aggregation of XML Data*. http://www.cse.ogi.edu/~tufte/merge.html