University of Tennessee, Knoxville

TRACE: Tennessee Research and Creative Exchange

Masters Theses

Graduate School

12-2003

An information management system for a large-scale biological collaboration

Hsin-Neng Wang

Follow this and additional works at: https://trace.tennessee.edu/utk_gradthes

Recommended Citation

Wang, Hsin-Neng, "An information management system for a large-scale biological collaboration. " Master's Thesis, University of Tennessee, 2003.
https://trace.tennessee.edu/utk_gradthes/5315

This Thesis is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

www.manaraa.com

To the Graduate Council:

I am submitting herewith a thesis written by Hsin-Neng Wang entitled "An information management system for a large-scale biological collaboration." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Life Sciences.

Frank Larimer, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

المنارة للاستشارات

www.manaraa.com

To the Graduate Council:

I am submitting herewith a thesis written by Hsin-Neng Wang entitled "An Information Management System for a Large-Scale Biological Collaboration." I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Life Sciences.

Frank Larimer, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost and Dean of Graduate Studies

Thesis
2003
.W35

AN INFORMATION MANAGEMENT SYSTEM

FOR A LARGE-SCALE

BIOLOGICAL COLLABORATION

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Hsin-Neng Wang
December 2003

Dedication

This thesis is dedicated to my parents, Chun-Chi Wang and Hsiu-Chun Chiang Wang as well as the rest of the family, for their enduring love and support.

I would also like to dedicate this thesis to my former supervisors in Taiwan,
Dr. Lan-Yang Ch'ang and Dr. Hui-Ling Chen
for guiding and inspiring me during the most overwhelming time in my life.

Acknowledgments

I wish to thank all those who helped me in completing my Master of Science degree in Life Sciences. First of all, I would like to thank my supervisor, Dr. Jay Snoddy, for giving me the opportunity to work with him and always giving me constructive advice and endless support throughout the project. I am grateful to my major professor, Dr. Frank Larimer, and committee members, Dr. Bruce Whitehead and Dr. Naima Moustaid-Moussa for their guidance and support.

I also wish to thank Denise Schmoyer and Dr. Erich Baker for helping me on issues relating to database design and management. I thank Adam Tebbe for providing assistance on issues relating to the system integration with the BioTeams. I also thank Harold Shanafield and Leslie Galloway for being ever helpful on interface design. I am especially grateful to Dr. Oakley Crawford and Suzanne Baktash for proofreading this thesis.

I would like to thank my fellow graduate students and staff members in the UT-ORNL Graduate School of Genome Science and Technology, Dr. Bing Zhang, Dr. Stefan Kirov, Jane Razumovskaya and Xinxia Peng, for their academic help and respect, as well as for providing laughter in the lab.

Finally, I would like to thank my loving family, and friends in the Taiwanese Students Association of the University of Tennessee, Knoxville, as well as my best friends in Taiwan, for their support and cheering me up during my graduate studies. Thank you for everything you ever gave me.

Abstract

The main goal of this project is to design and construct a collaborative information management system to promote the research of the Integrative Neuroscience Initiative on Alcoholism (INIA), a consortium formed to explore the neural mechanisms that link stress, anxiety and excessive alcohol consumption. Building this collaborative bioinformatics system across geographic and disciplinary barriers poses several challenges. We are developing several systems to gather and analyze data from the different INIA researchers. The INIA Core System (INIACS) has been designed and developed to support the exchange of data about samples and results between the core analysis labs and the research labs that request services from the cores. This system is also testing a method of storing data using a compromise between XML (Extensible Markup Language) and RDBMS (Relational Database Management System) technologies. This compromise can give us the architectural flexibility to adapt to the evolution of the data models for the different kinds of biological data that this INIA consortium will produce. Moreover, the design allows for robust data integration, especially around the genes and gene products that are identified in different data sets.

The INIACS is currently implemented in Oracle 8*i* RDBMS and a user-interface has been developed in the PHP4 programming language. This thesis discusses the requirements, which are needed to develop this information management system and then presents the INIACS' design and implementation phases in detail.

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

Many biologists currently do research by examining individual genes or gene products one at a time. However, phenotypes of interest are created out of highly interconnected networked systems of genes and gene products. In order to understand the entire highly networked system and move toward a 'systems biology' approach (1), researchers with expertise in several different genes, gene products, or systems have to collect and integrate relevant data or knowledge that has been discovered from various research fields. However, it is hard to acquire data from other research groups to gain knowledge about important components in the networked system because they are often located in different research institutions around the world. Due to the diversity and complex nature of biological data, it is also difficult to integrate the information to form a systematic view.

We are now entering the "post-genome" era. With the significant advancements in high-throughput techniques (i.e. DNA microarrays, yeast two-hybrid systems and mass spectrometry), biologists are able to investigate a set of thousands of genes and gene products simultaneously and also study the relationships among them at the system level. However, advances in experimental technology also result in the accumulation of large amounts of data in one single experiment, which cannot be handled easily by the traditional approach, (i.e. laboratory notebooks and simple analyses). Larger scale data sets lead to a number of challenges, including data analysis, access, storage and management. Moreover, because research in different disciplines may focus only on selected components in a networked system, the expert knowledge about each component in the system is often distributed at different locations. Functioning in disparate locations increases the difficulty of data collection

and data analysis and collaboration as a whole. Yet another challenge is how to get access to a wide variety of distributed resources that may be needed to understand these networks. Typically, these resources include, for example, domain expertise, technical expertise, equipment and computational resources. To gain knowledge from multiple research fields, there is a growing trend toward collaborative research projects that join researchers across or from multiple disciplines and institutions (2). This trend results in resources being distributed geographically and requires advances in information technology to support access to them.

To overcome these challenges and facilitate new advances in the biomedical sciences, biologists will need information technology (IT) and bioinformatics tools to acquire, manage, share and integrate complex and distributed data (3). Moreover, those data sets must be organized so that sophisticated computational biology tools can be applied.

The Integrative Neuroscience Initiative on Alcoholism (INIA) is a consortium that was recently formed to investigate the complex neurobiological systems and processes that are associated with stress and alcohol abuse using model organisms and humans (4). The neurobiological phenotypes seen in response to stress, the phenotypes associated with alcohol and other substance abuse, and possible networked interconnections among these phenotypes are a fundamental problem of national importance. This INIA consortium is using a number of model organisms and model systems to understand the fundamental aspects of these phenotypes and the genetic and molecular networks that help cause these phenotypes.

This consortium, like other large-scale collaborative research projects, will have multiple needs for new advances in bioinformatics that can support system-wide analysis. For example, to understand how sets of genes, gene products, and other

factors interact within networked systems to create complex phenotypes, it is essential to integrate and compare multiple levels of biological data from molecular, cellular and organismal levels. A prerequisite to data integration is bioinformatics that can acquire large amounts of data from all of the relevant researchers and store it in semantically rich databases that allow integrated data viewing and analysis. Since research in these networked biological systems will require expertise that is diverse and geographically distributed, the bioinformatics will need to work in an Internet environment or other Wide Area Network (WAN), preferably via standard web technologies. Knowledge discovery in these widely accessible databases will also require interactions among these different groups to process and interpret data sets, as well as provide better computational analysis tools.

1.1 An Introduction to INIA

In 2002, the National Institute on Alcohol Abuse and Alcoholism of the National Institutes of Health funded a 5-year initiative, called Integrative Neuroscience Initiative on Alcoholism (INIA), to explore the neural mechanisms that link stress, anxiety and excessive alcohol consumption. There are two principal scientific consortia involved in this INIA funding. These are the INIA-West and INIA-East consortia comprising multiple institutions across the United States. The INIA-West consortium (5), headed by Dr. George Koob of The Scripps Research Institute (TSRI), San Diego, California, brings together scientists from the University of Texas-Austin, the Oregon Health Sciences University, Indiana University School of Medicine, the University of Colorado Health Sciences Center, and Stanford University and SRI International. The INIA-East consortium, headed by Dr. Kathleen Grant of the

Wake Forest University School of Medicine, also brings together scientists from the University of Memphis, the University of Tennessee Health Sciences Center, Vanderbilt University, the UT-ORNL Graduate School of Genome Science and Technology and other researchers. The project described herein is developing an information infrastructure specifically for the INIA-East consortium.

1.1.1 Background

Recent studies on the neural mechanisms involved in the development of alcoholism have shown that excessive alcohol consumption and alcoholism result from a complex interaction between multiple genetic and non-genetic factors. An example of possible genetic factors is the $GABA_A$ receptors. $GABA_A$ receptors are the principal postsynaptic receptors for gamma-amino-butyric acid (GABA). GABA is required as an inhibitory neurotransmitter in the central nervous system (CNS) and is widely distributed throughout the brain. Chronic use of alcohol results in alterations in the $GABA_A$ receptor system; these alterations dampen the function of the GABAergic system. Previous studies have suggested that the decreased inhibitory function of the GABAergic system may contribute to alcohol withdrawal symptoms (6). In addition, two $GABA_A$ subunit genes, GABRB3 and GABRA5, which have been found on human chromosome 15 (7, 8), were reported to have alleles that are potentially involved in the risk for alcoholism using family-based association tests (9).

Stress is an environmental factor widely thought to contribute to excessive drinking and alcoholism. For example, stress related to job, social or financial problems is thought to contribute to increased alcohol use and abuse. One assumption is that alcohol relieves emotional pain and is often perceived as a loyal friend when

people are suffering stressful life events (10). However, stress is induced by a complex spectrum of behavioral, biological and emotional reactions that all have neurobiological components involved. The reactions of the brain to stress appear to involve biochemical components that vary with the individual's genotype. For instance, the Corticotropin-Releasing Hormone (CRH) system is known to regulate endocrine responses to stress (11). CRH is a 41-amino acid neuropeptide produced mainly in a brain region called the hypothalamus. The CRH signal is transmitted by CRH1 and CRH2 receptors with different expression patterns in the brain. Researchers have found that the lack of a functional CRH1 receptor may lead to increased alcohol consumption in mice (12).

Anxiety is one of the body's responses to exposure to stressful life events. The relationship between alcoholism and anxiety has also been addressed in previous studies. For example, many patients with social anxiety disorder, which is an excessive fear of social situations such as public speaking, are likely to suffer from alcohol abuse or dependency problems. They appear to use alcohol to relieve their stress and fears. Although there is some evidence to indicate a high degree of co-occurrence of anxiety and alcoholism, the complex mechanisms of how stress may contribute to excessive alcohol consumption is unclear. In addition, it is unclear how genes and genetic factors influence these complex mechanisms and work with environmental factors (13).

In this INIA consortium, researchers use a multi-disciplinary approach to identify the neural mechanism involved in stress-alcohol interactions at all levels, from genetics through molecular biology and up to behavior (4). One primary mode of collaboration is that all members of the consortium share and utilize various core facilities so that researchers can access different types of data, resources, technologies and expertise that are beyond the usual scope of a single lab. By sharing resources and

information within the consortium, this large-scale collaborative project is expected to assist researchers to discover new knowledge and further achieve its ultimate goal of improving treatment and lessening the need for intervention.

1.1.2 Organization of the INIA

The current organizational structure of the INIA consortium is as follows. It contains nine research project components (see table 1.1) to focus on the neural effects of stress-alcohol interactions that contribute to excessive alcohol intake at all levels from molecules to behavior in several organisms ranging from mice to monkeys and humans. These research components are fully supported by five distributed core facilities (see table 1.2) where the researchers can access resources, technologies and expertise that are beyond the scope of any single research laboratory. In addition, it currently contains five pilot project components (see table 1.3) to provide new research activities, directions or opportunities related to the overall INIA aims. It is anticipated that the consortium will recruit more experts from different research groups in other research projects and initiate pilot projects to help establish state-of-the-art knowledge resources for alcohol research. The Informatics Core will need to build systems that can accommodate the additional research groups and pilot projects being added.

Table 1.1: Current INIA research project components

Project	Title
1	System Mechanisms in Stress-Alcohol Interactions
2	Genetic Analysis of Alcohol Consumption and Stress
3	Complex Trait Analysis of Alcohol and Stress Interactions
4	Stress and Ethanol Self-Administration in Monkeys
5	Schedule-Induced Polydipsia Ethanol Drinking in Mice
6	Chimeric Analysis of Alcohol and Stress Interactions
7	Early Stress and Alcoholism: Neurobiological Analysis, David Friedman
8	Ethanol Stress and Dopamine
9	Ethanol Dependence and Stress Effects on Ethanol Drinking

Table 1.2: Current INIA core facilities

Core	Title	Conduct Institution
1	Inducible Knockout Mouse Core	Vanderbilt University
2	Informatics Core	UT-ORNL Graduate School of Genome Science and Technology
3	Genotyping Core	University of Tennessee Health Sciences Center
4	Bioanalytical Core	University of Memphis
5	Neurohistology Core	University of Tennessee Health Sciences Center

Table 1.3 Current INIA pilot project components

Pilot	Title
1	Identification of Single Nucleotide Polymorphisms (SNPs) in the Human Serotonin Transporter of Alcoholics
2	Regulation of Synaptic Transmission in BNST by Norepinephrine and Alcohol
3	Stress and Ethanol Self-Administration in Mice
4	Adaptions to Stress/Alcohol Exposure in the Amygdala
5	Limbic Neuron Responses to GABA, Gluamate, and Ethanol in Knockout Mice

1.1.3 Interrelations between Core Services and Research Project Components

The research addressed by research project components overlaps somewhat but generally uses different approaches. Through formation of the INIA, researchers are expected to provide complementary information and resources for each other. For example, in Project 1, research in the impact of gene knockouts on alcohol effects in stress-related brain regions will provide researchers with the information about the role of these genes that can help to explain the alcohol-related phenotypes observed in Project 5. The examination of the effects of stress on excessively drinking monkeys in Project 4 will help researchers in Project 1, Project 2 and Project 5 to find new stress-alcohol interactions related to heavy drinking phenotypes in rodents.

Sometimes the expertise or technologies that are needed by the research projects may not be found in the investigator's laboratory or institution but in INIA's cores, which may be located at another university, or institution. Core facilities are cost-effective because they provide researchers with access to expensive instruments

and advanced expertise that their laboratories or universities may not have. Researchers are also allowed to access the new technologies or research models developed by core facilities. For example, the Inducible Knockout Mouse Core will provide several research components with a number of gene-targeted or region-specific and inducible knockout mice used to examine stress-anxiety-alcohol interactions. The Bioanalytical Core will provide researchers with access to microarray techniques and specific services on the analysis of stress-related hormone levels. In addition to providing services, the cores can establish new model systems from findings in research components. For example, the new alcoholism-related genes or specific brain regions identified in Project 1 and Project 6 components can be used as hypotheses for the Inducible Knockout Mouse Core to create new knockout model systems in mice.

INIA research labs are expected to request services from these cores. INIA researchers need to communicate with core services to arrange the analysis. Depending on what kind of analysis is needed, researchers will need to send samples or raw data to core labs. The core labs will do the required analysis and give the results back to the researchers. All of the data and information generated during this collaboration process will be managed through the Informatics Core and stored in a central data repository where researchers will be able to track sample movements, access data and share or exchange information with other INIA members.

In addition to providing collaboration systems for collecting and storing data, the Informatics Core is responsible for developing computational analysis tools to facilitate the data interpretation and information dissemination. For example, expression array data from the Bioanalytical Core may need the Informatics Core to provide computational analysis and data mining. In addition, the QTL (Quantitative

Trait Loci) data generated in the Project 3 component will need advanced data mining and computational analyses. In order to gain knowledge from this multi-disciplinary approach, the Informatics Core will devise methods for integrating data that are generated from different levels of analysis within this INIA consortium.

1.2 The Project

As the Informatics Core of the INIA, our lab is responsible for developing and maintaining appropriate information infrastructures for this INIA consortium. We are developing several systems to accomplish these tasks. This project initiated work on one of those tasks. As a part of the funding project under the INIA consortium, the objective of this project is to meet the collaborative information technology needs of the INIA. The following section summarizes the ideas of major design requirements. This user requirements analysis would be the driver for the design and implementation of the system.

1.2.1 INIA's Needs

As mentioned in section 1.1, this project is specifically for the INIA-East consortium, widely distributed collaboration. Most major projects are scattered around Tennessee and North Carolina. With the complex interrelationships among the INIA components (refer to section 1.1.3), many data sets will need to be transferred back and forth among these collaborators.

Without an advanced IT infrastructure, the INIA researchers could exchange data and communicate with others only via e-mail, paperwork or telephone. Research

labs could arrange their requests with core labs via these methods but these requests would not be easily archived for researchers or for the INIA project administration core. In addition, integrated access to the data would not be feasible. Raw data or result files would be stored in several local computers or database systems. Also the data would be stored with no enforced transactional integrity or enforced semantic structuring. Thus, these ad hoc methods would make it difficult to access or exchange data as well as to integrate or share data among the consortium members due to the geographic limitation.

To overcome these problems, two components are needed in this INIA consortium. The first one is a central database system using a database management system (DBMS) to assist researchers in accessing, exchanging and managing collaborating data. A DBMS is a software program designed to handle all access to the data in a database (14). Typically, DBMSs need networking support to allow multiple users to run operations on data sets remotely and simultaneously. There are several advantages that could add value to unstructured or semistructured data when a DBMS is used. For example, a DBMS can enforce standard processes, ensure transactional integrity, provide archived backups, automate routine procedures, and help enforce semantic structuring that can promote data integration.

Researchers can store their data in this database system. Once researchers submit their requests and send samples to core labs, the system will assist these core labs to manage and organize relevant data or information. In addition, researchers can share their new discovery with others easily without heavy workload.

The second required component is an interface layer that enables users and applications to access the DBMS. In this project, the user interface will be web-based, given the wide range of users. The interface layer is responsible for the communication

between end users and the application and data layers. A user interface can deliver user's requests to the application layer, which is responsible for sending those requests to DBMS (15). In the Oracle DBMS, for example, the interactions between the application layer and the database server can be accomplished through the Oracle Call Interface (OCI). A web-based user interface is also responsible for formatting the query results into HTML pages that allow people to access and exchange geographically distributed data via the Internet no matter where they are. This interface should be compatible with most common web browsers, e.g. Internet Explorer (IE), and also easy to use on different platforms such as Windows or UNIX environment. With this component, researchers can confirm and monitor their interactions with other labs, track samples, and receive analysis results on the web site as long as they have an Internet network connection.

With high-throughput techniques, such as those used in some INIA research, some laboratories may generate tremendous amounts of data per day. In such cases, it can be very cumbersome to process and manage data manually. Among the requirements for higher sample throughput is making the process automatic and efficient. To meet this demand, an IT infrastructure is often used by projects to automate and otherwise facilitate the routine processes of data discovery. For example, a Laboratory Information Management System (LIMS) can be used to manage the collaborative resources and deliver the resources to designated areas. These areas could be in the same laboratory where the resources are generated or other collaborative labs at different physical locations (16, 17, 18).

1.2.2 Proposed Solution: A Web-based Laboratory Information Management System

These collaborative resources could be stored in a central LIMS data repository and then distributed through a web-based user interface. In this project, a LIMS-like information management system called INIA Core System (INIACS) is developed as a model, which is expected to meet the requirements of this large consortium. This system must provide certain basic functionality, for example, storing the information that is relevant to each individual sample, tracking sample movements and test processes, and reporting results to users (19). Additional development beyond the scope of this project may be needed for the specialized internal needs of each core.

The INIACS system is designed to centralize data collection, storage, analysis and reporting through a set of database applications that include a relational database management system (RDBMS) and several software packages, which can enforce the INIA "business rules" and provide user interfaces.

In addition, this project will address some issues that are not usually found in information management systems for small research groups. For example, in large collaborative projects, multiple users need to access the database simultaneously but with different privileges, e.g. some data sets may be accessed by one particular group of users but not others. Thus, those data must be stored in a secure and accessible way that can control its access to users (20). In other words, the system needs to ensure that users could access what they should and cannot access what they should not. Moreover, the data should be sharable among the consortium or off-site collaborations. In such a way, other researchers will be able to add their expert interpretation or annotation on the shared data.

In order to exchange or integrate heterogeneous data, this system will be designed to test a method of storing data using a compromise between XML (Extensible Markup Language) and RDBMS. This compromise can give us the architectural flexibility to adapt to rapidly evolving data models for the different types of biological data that this INIA consortium will produce. The data can be kept in XML files and some of the most useful data items will be decomposed into relational tables. This partial decomposition will be described later in section 6.2.

This system should be scalable so that it could be integrated with other local LIMSs, which already exist in core labs. This system can serve as a core system within this consortium. Researchers could capture the data stored in local LIMSs through this core system and avoid duplication of data from one site to another. As a level above other systems, it could provide users a consistently appearing interface so that users can easily navigate the interface across different systems.

In this project I have designed and developed an IT infrastructure for the INIA consortium. The system was developed to assist INIA researchers in acquiring, managing, sharing, and analyzing large data sets. This IT infrastructure should enable researchers to better collaborate and gain insights from shared data sets. In addition, this infrastructure can serve as a framework to support the integration of additional bioinformatics tools that can further assist researchers in finding insights from these large data sets. The insights that are fostered by this IT system should lead to new hypotheses and knowledge about the neurobiology of stress and alcoholism. Additional developments beyond the scope of this initial project may be needed for the specialized needs of each core, which will become apparent as this first system is tested. The INIACS system we implemented should allow this integration. Additional bioinformatics research is ongoing that can help end users manipulate some types of

data that will be stored in INIACS and find fundamental insights from the data. The INIACS LIMS framework we have created is required to make these advanced analysis tools really useful for the analysis of large data sets from the INIA collaborations.

Chapter 2 INIACS: An Overview

The INIA Core System (INIACS), an information management system, is developed to support data exchange among the INIA consortium. In addition, it should be able to assist the routine processing and knowledge discovery of large-scale collaborative projects. Like other information management systems, the INIACS should meet the significant needs of the consortium. As new technologies rapidly develop in life sciences, there is an explosion in data generation and format diversity that would be a challenge for developing the INIACS. This rapid development can cause difficulty in data transfer and integration. To address this problem, the first version of the INIACS is designed and implemented as a prototype having both flexibility and extensibility. It should be able to integrate with other information systems and add more functionality in adapting to changes in technologies. The objectives of this project are defined in this chapter. This will be followed by a description of the system architecture and an overview of the key features of the INIACS.

2.1 Objectives of the Project

According to the needs of the INIA consortium, there are two major goals defined for this project. Each goal has one specific aim to achieve.

2.1.1 Goal 1: Promote INIA Collaboration and Information Management

The INIACS should support the geographically distributed collaborations among the INIA consortium members. Due to the diverse research fields within this consortium, the system should also support various data sets in such a way as not to limit those data sets to any particular field. In other words, a data repository is needed and it should accept many types of data that this INIA consortium will produce.

- **Specific Aim: Generic Database System Development**

  To achieve this goal, we need to develop a generic database system to handle all kinds of information derived from the collaborations. For robust interactions between researchers and core services, it should provide researchers with the facilities needed to track the request process and sample movement. The database details will be discussed in chapter three.

2.1.2 Goal 2: Promote Data Sharing and Resource Use

The INIA consortium was formed to integrate research knowledge from multidisciplinary approaches in order to understand the behavioral neuroadaptive process. To this end, it is crucial to allow researchers to access all resources that are useful in assisting them to build model hypotheses for new knowledge discovery. Thus, the INIACS should build a data sharing and accessing model that is suitable for this consortium.

- **Specific Aim: Web-based Interface Development**

  To achieve this goal, we need to develop a database-backed web interface that allows consortium members and other interest groups access to shared resources. The interface design will be discussed in chapter four.

2.2 Architecture of the INIACS

Several components are needed in building an information management system to support collaborations, such as hardware, software and network components. Each of them plays a specific, important role in the system. For example, a software application with networking support such as a web-based user interface allows multiple users to run operations on the same data set simultaneously.

Typically, most application programs have three major functional and logical layers: an interface layer, a business logic layer and a data layer.

The INIACS is designed to have the interface and business logic layers together on one machine and the data layer could be separated on another database server machine. The interface and business logic layers were implemented in PHP, a web-friendly programming language. PHP is a server side language with the Application Programming Interfaces (APIs) support to the connection of the data layer so that the application can be accessed from different directions (see figure 2.1).

The INIACS data layer was implemented within the ORACLE 8*i* RDBMS. ORACLE provides an API called Oracle Call Interface (OCI) that allows the applications to interact with the Oracle servers. PHP provides functions that can use the OCI to access Oracle databases.

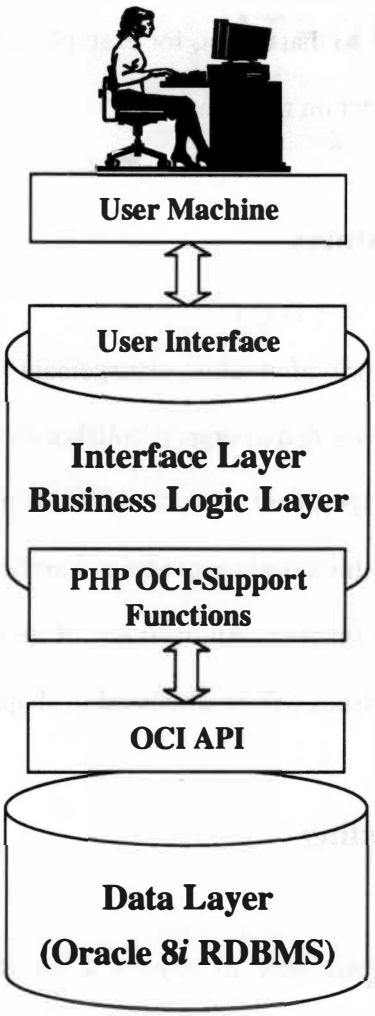


Fig.2.1 The architecture of the INIACS

With this architecture, the INIACS allows multiple users to access the same data sets simultaneously from different computers, geographical locations, programs, and protocols. Currently, the major route of access to INIACS is via the Web and the PHP software layers, but other methods, such as Open Database Connectivity (ODBC) connections can be added so that a user, for example, could also use Microsoft Excel or Access to view and report on the data.

2.3 INIACS Key Features

Like other laboratory information management systems, the INIACS provides certain basic functionality needed to support collaborations: service requesting, sample login, sample tracking, experiment scheduling, result reporting and other processing components. In addition, this system provides researchers with a project management function and data sharing function. An overview of these key features is given in this section. Detailed design issues will be discussed in chapter four.

2.3.1 Service Requesting

INIA researchers are able to request a service through this functionality. Meanwhile, for the purpose of properly organizing the relevant data according to the request, a project is automatically created by the system (see figure 2.2). All of the relevant data including samples and results will be associated with the project (refer to section 2.3.6).

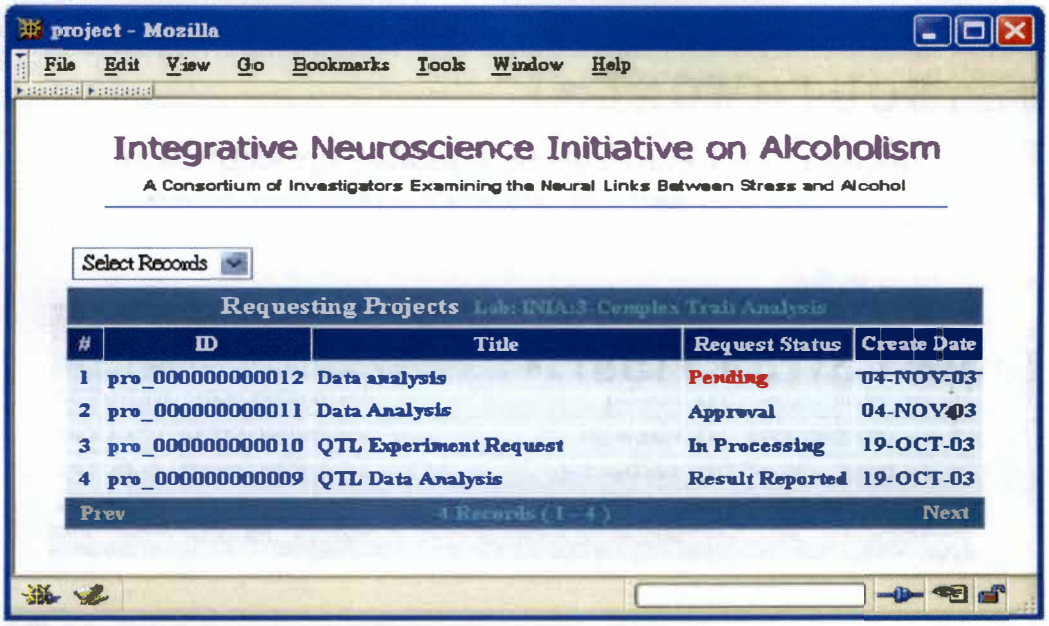


Figure 2.2: Service requesting feature

2.3.2 Sample Login

This functionality allows users to log samples into the system. Since multiple samples need to be analyzed in a large project, users are allowed to create a sample set for logging multiple samples. Users can also create multiple sample sets associated with a single project (see figure 2.3). A sample item could be added right after the sample set has been created or sometime later by assigning the sample to an existing sample set.

2.3.3 Sample Tracking

This functionality allows users to track samples throughout the distributed collaboration. The INIACS provides a hierarchy tracking function. Multiple sample movements can be tracked by using a project ID or a sample set ID, and a single

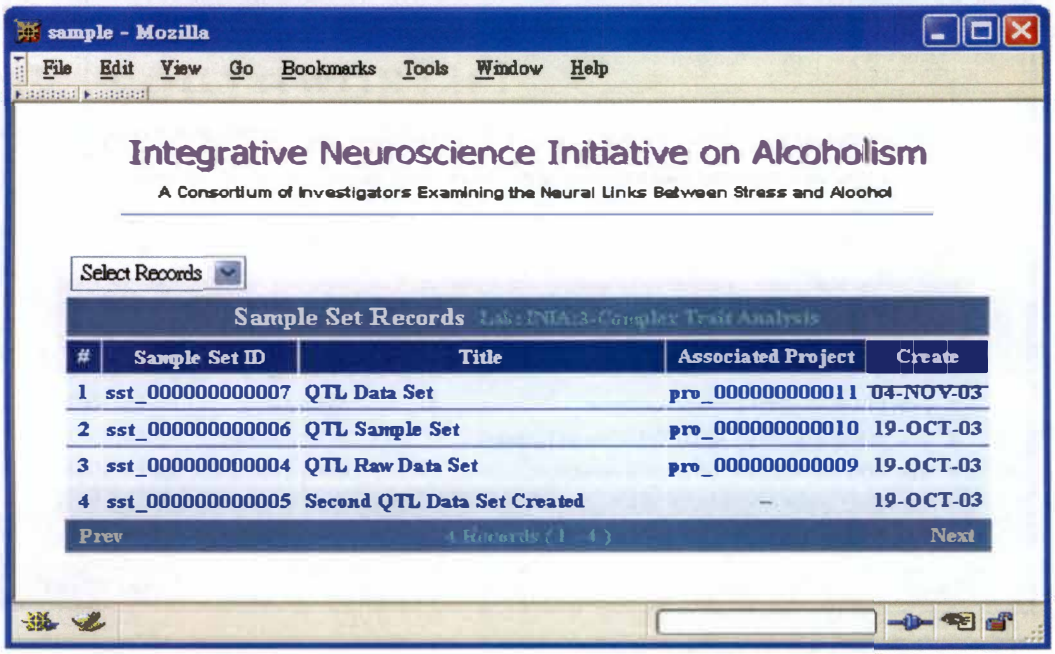


Figure 2.3: Sample login feature

sample item can be tracked by using a file ID or a shipment ID (see figure 2.4). An example of tracking multiple sample movements using a project ID is shown below (see figure 2.5).

2.3.4 Experiment Scheduling

This functionality allows core labs to schedule experiments that are needed for working on a project (see figure 2.6). In addition, this design should allow core services to create customized worksheets according to the way they analyze or assign the standard protocol to an experiment in a routine process. This will be discussed in section 6.1.

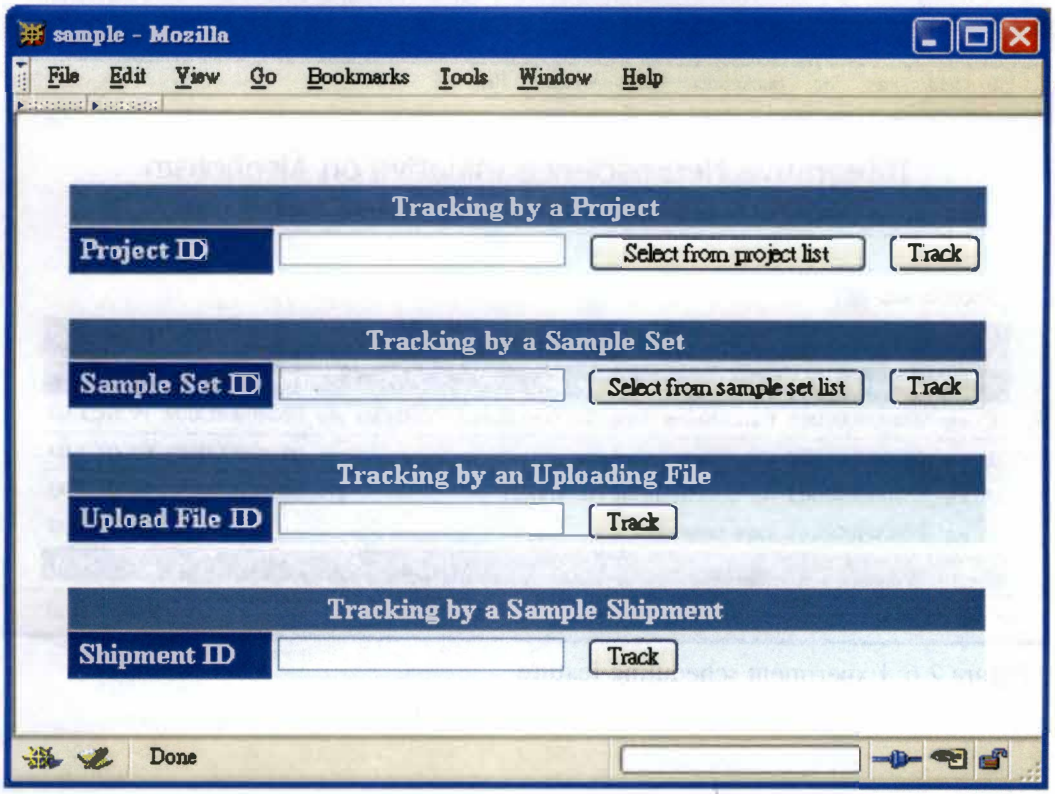


Figure 2.4: Hierarchy sample tracking function

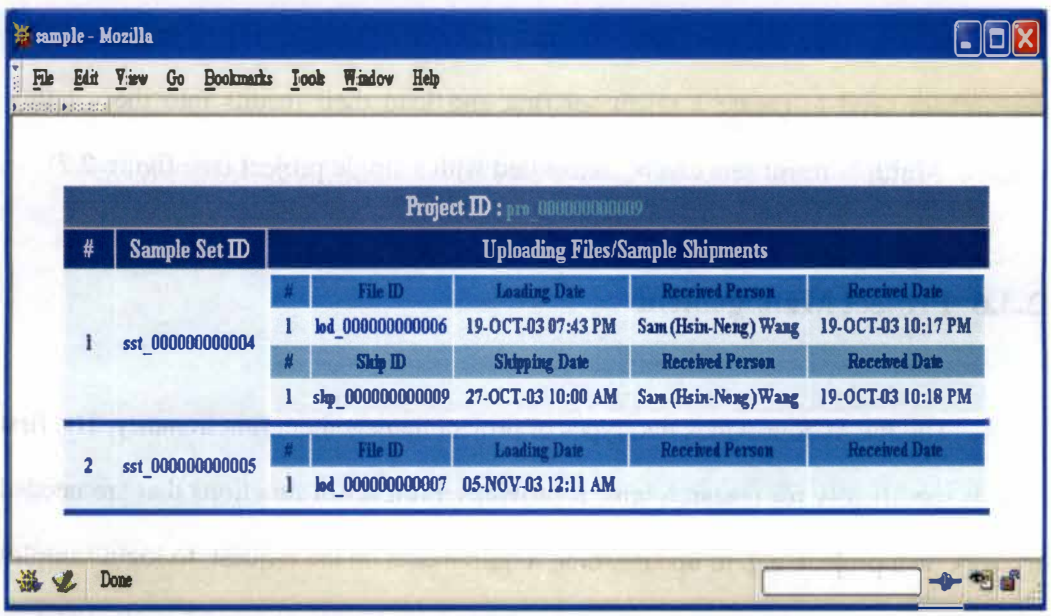


Figure 2.5: An example of tracking multiple sample movements using a project ID

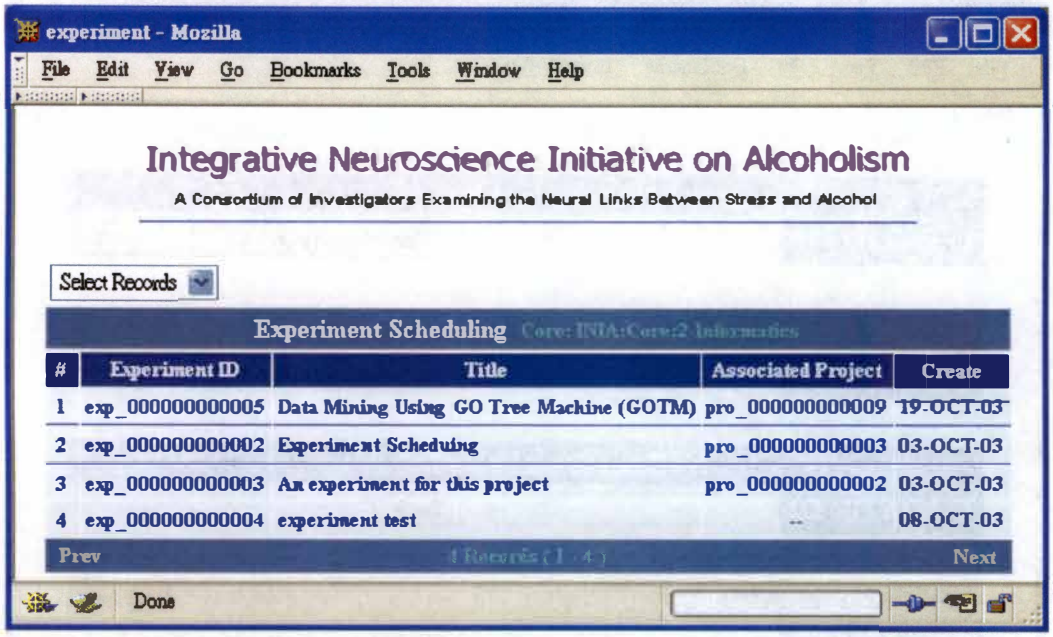


Figure 2.6: Experiment scheduling feature

2.3.5 Result Reporting

Core labs are able to report results through this functionality. Results can be stored in any type of electronic format. Like the sample login function, core researchers need to create a result set first and load their results into the set they created. Multiple result sets can be associated with a single project (see figure 2.7).

2.3.6 Project Management

The INIACS provides two types of project management functionality. The first type is specifically for research labs. It provides a full set of functions that are needed to work on a project, e.g. to update some requirements on the request, to login samples or to track sample movements. It also allows research labs to acquire the relevant data

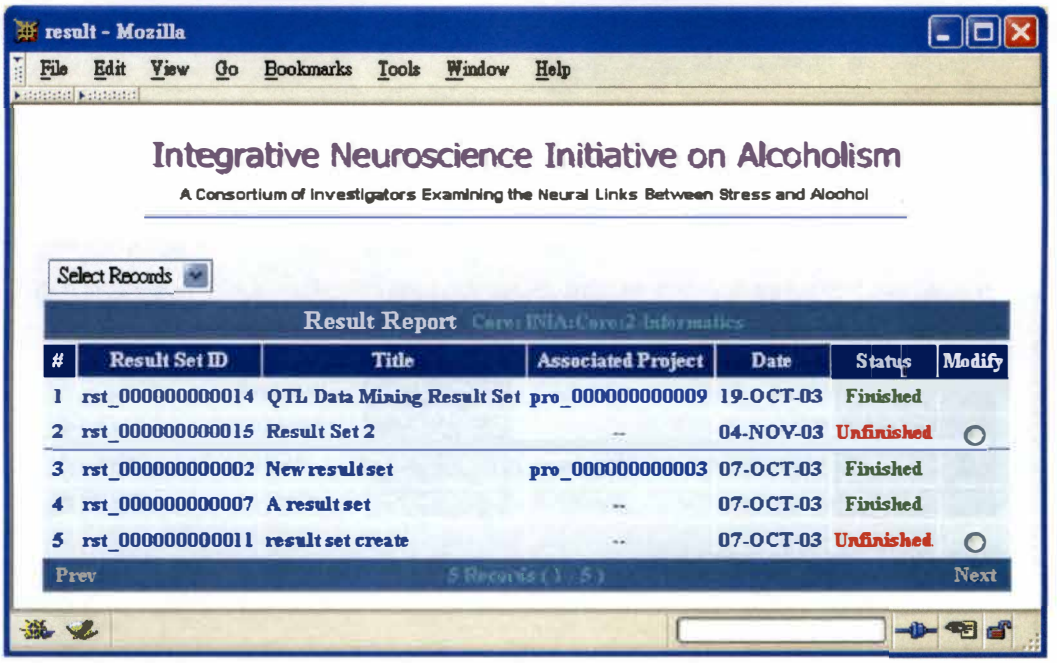


Figure 2.7: Result reporting feature

from core labs such as responses, experiment schedules or results (see figure 2.8).

The second type is specifically for core labs. Thus, it provides a full set of functions that are needed for core labs to work on a project, e.g. to make responses, to receive samples, to schedule experiments or to report results. In addition, with this functionality core labs can update process status or move finished projects to archives (see figure 2.9).

2.3.7 Data Sharing

The ultimate goal of the consortium is to share data among all researchers in the field. However, not all researchers want to share their raw data before analysis. Thus, some preliminary data will need to be temporarily restricted from public view until it is validated.

project - Mozilla

File Edit View Go Bookmarks Tools Window Help

Integrative Neuroscience Initiative on Alcoholism

A Consortium of Investigators Examining the Neural Links Between Stress and Alcohol

Update this project | Upload/Ship Sample | Sample Tracking

QTL Data Analysis

Research Lab Activities			Core Service Activities		
Project Creator	Sam (Hsin-Neng) Wang		Core Service	INIA:Core:2-Informatics	
Create Date	19-OCT-03		Active Mode	Ongoing	
Request Status	Submitted		Process Status	Result Reported	History
Access Mode	INIA East		Responses	1 response(s)	View Records
Request Update	0 update record(s)	View Records	Experiments	1 experiment(s)	View Records
Sample Sets	2 sample set(s)	View Records	Result Sets	1 result set(s)	View Records

The long description about this request goes here.....

Done

Figure 2.8: Project management function for research labs

core_project - Mozilla

File Edit View Go Bookmarks Tools Window Help

Integrative Neuroscience Initiative on Alcoholism

A Consortium of Investigators Examining the Neural Links Between Stress and Alcohol

Response this project | Setup Experiment | Result Report

QTL Data Analysis

Research Lab Activities		Core Service Activities			
Request Lab	INIA:3-Complex Trait Analysis	Process Status	Result Reported	Change Status	History
Manager	Sam (Hsin-Neng) Wang	Project Status	Ongoing	Change Status	
Request Date	19-OCT-03	Responses	1 response(s)	View Records	
Request Status	Submitted	Experiments	1 experiment(s)	View Records	
Access Mode	INIA East	Result Sets	2 result set(s)	View Records	

The long description about this request goes here.....

Sample Uploading/Shipping Information : 2 sample set(s)

1 sst_000000000004 19-OCT-03

Sample Archives : 2 record(s) selected

1 lod_000000000006 19-OCT-03 application/vnd.ms-excel QTL Raw Data File

2 slp_000000000009 27-OCT-03 DNA A DNA Sample

2 sst_000000000005 19-OCT-03

New Sample Loading : 1 record(s) selected

1 lod_000000000007 05-NOV-03 application/vnd.ms-excel Receive

Figure 2.9: Project management function for core labs

From this point of view, we have to design and implement a rule to manage and secure the information resources. The control of the data passes to the research lab once the core lab has completed the analysis. The research lab can then release the data to public domains when it is ready. With this functionality the researchers who requested the services are able to share their projects with different public domains by changing the mode of the project (see figure 2.10). Three public domains are defined: INIA-East, INIA-All and World domains (see figure 2.11). If the project is private and not sharable with others, it will be in the "Ownership" mode.

2.4 Collaborative System Integration

During the software development phases, the cost of developing a tool is an important consideration. Sometimes, to achieve the goals and aims in an economic solution, it is possible to choose and integrate existing tools as well as to develop code for special needs like the INIA.

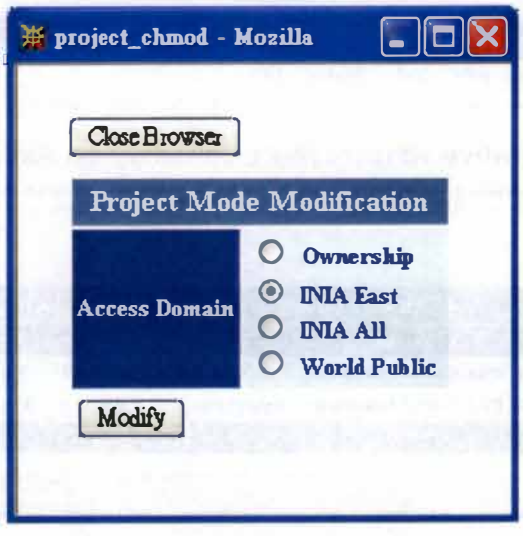


Figure 2.10: Data sharing function

public - Mozilla

File Edit View Go Bookmarks Tools Window Help

Integrative Neuroscience Initiative on Alcoholism

A Consortium of Investigators Examining the Neural Links Between Stress and Alcohol

Select Records

Public Projects Access Domain: INIA-East

#	ID	Service	Title	Request Status	Process Status	Date
1	pro_000000000009	INIA:Core:2-Informatics	QTL Data Analysis	Submitted	Result Reported	19-OCT-03
2	pro_000000000003	INIA:Core:2-Informatics	test	Submitted	Result Reported	27-SEP-03

Prev 2 Records (1 - 2) Next

public - Mozilla

File Edit View Go Bookmarks Tools Window Help

Integrative Neuroscience Initiative on Alcoholism

A Consortium of Investigators Examining the Neural Links Between Stress and Alcohol

Select Records

Public Projects Access Domain: INIA-All

#	ID	Service	Title	Request Status	Process Status	Date
1	pro_000000000010	INIA:Core:4-Bioanalytical	QTL Experiment Request	Submitted	In Processing	19-OCT-03
2	pro_000000000002	INIA:Core:2-Informatics	test	Submitted	In Processing	26-SEP-03

Prev 2 Records (1 - 2) Next

public - Mozilla

File Edit View Go Bookmarks Tools Window Help

Integrative Neuroscience Initiative on Alcoholism

A Consortium of Investigators Examining the Neural Links Between Stress and Alcohol

Select Records

Public Projects Access Domain: World

#	ID	Service	Title	Request Status	Process Status	Date
1	pro_000000000011	INIA:Core:2-Informatics	Data Analysis	Submitted	Approval	04-NOV-03
2	pro_000000000001	INIA:Core:2-Informatics	request test	Submitted	ResultReported	26-SEP-03

Prev 2 Records (1 - 2) Next

Figure 2.11: Three defined public domains: INIA-East (top), INIA-All (middle) and World (bottom)

The INIACS is designed to handle and manage the distributed data that will be produced in the collaborative process, e.g. samples, sample movement information, experimental data, and results. However, there are other types of information that can help foster geographically distributed collaborations and need to be handled properly. This would include the information about scheduling meetings or teleconferences, personal contact information, or the researcher expertise. Moreover, some information (e.g. personal contact information) may need to be shared with other systems, e.g. MuTrack, a system used by some INIA researchers to manage mouse information. Since the INIACS will not support these resources, we need to choose and integrate other tools that can help us to handle those resources with INIACS.

Fortunately, there are many existing open-source and modifiable collaborative systems, which can be found on the Internet, e.g. a lot of free software can be found at the SourceForge.net which is the world's largest open source software development website (21). For example, another group in our lab is currently developing one collaborative system called BioTeams-ver0.5 (see figure 2.12) which was modified from the Ultimate Team Organization Software (TUTOS) (22), an open system found at SourceForge.net to manage the organizational needs of small groups or teams. The BioTeams system is designed to be a model of "Person-Expertise Database" for biological collaborations to manage data about people, such as the contact information and personal role in a collaborative project. The project administrator can easily manage the human resources involved in the project or assign a particular role or task to a person. By integrating with the BioTeams system, it is expected to enhance the capability of the INIACS to handle many kinds of resources.

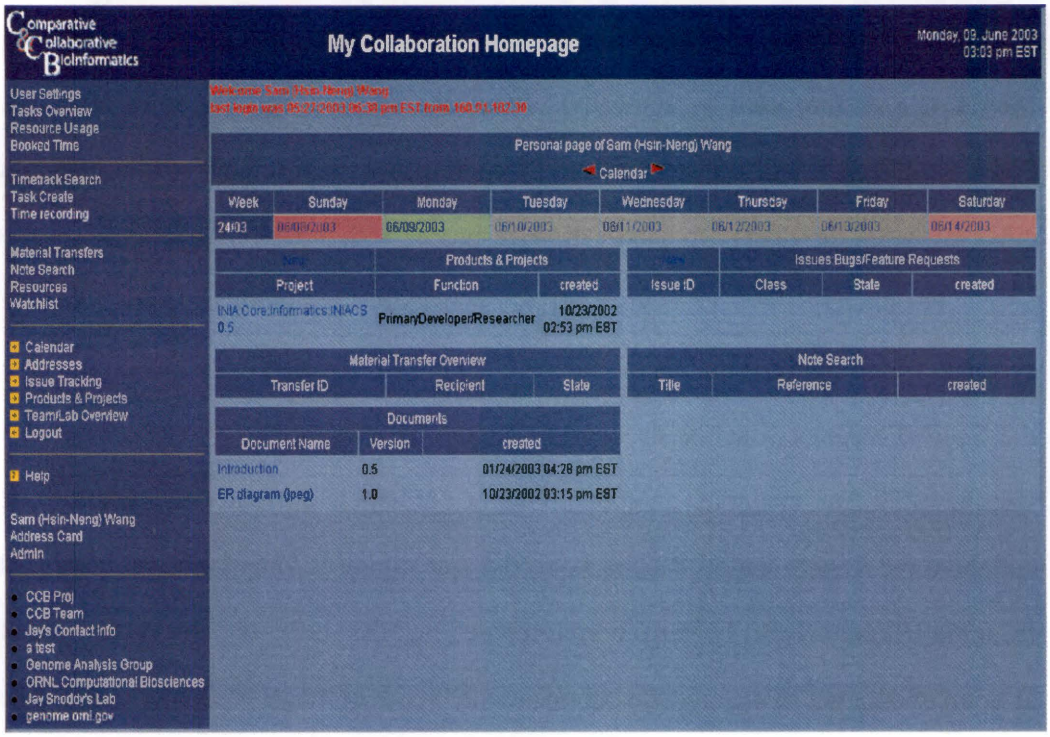


Figure 2.12: The interface of BioTeams system

Chapter 3 Database Design

This chapter describes the approach that has been taken to achieve the first goal of this project as defined in section 2.1.1. As previously mentioned, access to the geographically distributed resources for INIA researchers requires a central database system. Since many types of resources need to be managed through this system, the database design is a crucial issue in this project if we are to be able to use the modest informatics resources in an efficient way. A good database schema is needed to capture data robustly with sufficient semantic structuring and to help automate transactions for the required processes.

A well-designed database should follow the "business logic" rules and support the processes of its users. This database must also have the flexibility to adapt to changes in the data model, because new types of data are likely to be produced by INIA researchers. In this project, a generic database system that can meet the basic requirements of this INIA consortium has been designed and implemented. This system is designed to permit refinements and customizations to support the specific needs of different core service labs and research labs.

A relational database model has been chosen to design and implement the INIACS schema in Oracle 8*i* RDBMS. In section 3.1, the basic concept of a relational database model is introduced as well as the advantages when it is used. Following that, the issues about the INIACS database design is discussed, including the requirements analysis, the definition of the integrity constraints (business rules), the logical design (with Entity-Relationship (ER) modeling), the translation of the schema into Structured Query Language (SQL) and some important issues needed to address in order to meet the requirements.

3.1 An Introduction to Relational Databases

In general, the efficiency of a DBMS is largely determined by the database model created during the design process. A database model is a collection of logical constructs used to describe the structure of a database. Since 1960s, the evolution of DBMS has been driven by the emergence of several database models. Those are the hierarchical database model, the network database model, the relational database model and the object-oriented database model. Although each model has its own advantages and disadvantages when used in developing a DBMS, the relational database has become the dominant product in today's database marketplace (15).

The relational database model, first introduced by E.F. Codd of IBM Research Laboratory in 1970 (23), is implemented through a Relational Database Management System (RDBMS). RDBMSs provide functions to make the relational database model understandable and implementable. It is believed that the first RDBMS product was the Multics Relational Data Store (MRDS) offered by Honeywell Information Systems in 1976 (24). Later on the INGRES developed at the University of California, Berkeley was commercialized by Relational Technology, Inc. in the late 1970s. In the 1980s, IBM also released two commercial RDBMS products, SQL/DS and DB2 (25). Today, the most popular commercial RDBMSs include Oracle, Sybase, and Microsoft SQL Server and the most commonly used open-source RDBMSs are MySQL and PostgreSQL.

The concept of the relational database model provides that the database can be broke down into separate but related tables (relations). A table is essentially a two-dimensional structure composed of rows and columns. Each row (tuple) describes an item (entity) and each column (attribute) describes one of the characteristics of an

item. All values in a column must conform to the same data type (domain). Generally, any given entity (row) must be uniquely identified by an attribute that is the primary key of the table. The primary key must be unique so that it can maintain the integrity of the database. Related items can be linked through the use of these keys. In other words, a relationship between rows (tuples) in two tables can be created if they share a common field in which the primary key of one table column can appear again as a potential link (foreign key) in another table column. In such way, it can ensure the referential integrity of the database (14, 15, 25).

There are several advantages when a relational database model is used. Within a relational database system, the RDBMS manages all of the physical data storage details. The database designer does not need to consider how to physically storing the data when the database structure is determined. The RDBMS can still access the data even the database structure has been changed. This characteristic is known as the "structural independence" (15). Thus, this model allows one to ignore—at one level—the physical data storage characteristics; hence one is able to concentrate on the logical view of the database. The structured query language (SQL) gives this model an ad hoc query capability. SQL facilitates manipulations and interoperations of data as sets. SQL is easy to learn and makes database design, implementation, management and use much easier than some other database models. Because it is a standard language for dealing with relational databases, almost all RDBMS products support variants of a shared SQL standard set of commands (15). These advantages together contribute to the reasons why the RDBMS has become the dominant product in today's database market and the reason why it has been chosen as the model in this project. When reduced to practice, however, there are some complexities that can occur in the relational database design. A fully normalized data model avoids a

number of difficulties in data update or other problems, but this design can also come with some costs in ease of construction and ease of use. We have tended to follow a "best practices" view of what works given our resource limitations, rather than an overly strict adherence to relational database theory.

There are three types of the relationships that an entity can have with another one. The three types are as follows:

- **One-to-one (1:1) relationship:** One row in the first table (parent) can be associated with only one row in the second table (child), and vice-versa.

- **One-to-many (1:N) relationship:** One row in the first table (parent) can be associated with many rows in the second table (child). But one row in the second table is associated with only one row in the first table.

- **Many-to-many (M:N) relationship:** One row in the first table may be associated with many rows in the second table; furthermore one row in the second table may also be associated with many rows in the first table.

In the following sections, design considerations for the INIACS database will be presented in several categories: requirements analysis, logical design (including ER modeling and logical data model), the schema definition in SQL and remaining issues needed to address.

3.2 Requirements Analysis

Requirements analysis is the first step in system development. Although requirements analysis in most current software engineering methodologies is somewhat iterative with other steps, it is still the first step to be taken. A prerequisite to deliver an effective database product is that one must first evaluate what are the most important *user requirements* of the business or organization. In other words, before designing this database, it is crucial for designers to have a clear understanding of the research processes and business rules that have to be followed. An overall requirement of the INIACS is to support the research processes that occur among core labs and research labs. An equally important overall requirement is that the data sets that are accumulated during these processes will need to be archived, shared among INIA researchers and, as appropriate, presented to the public. Meeting these requirements is the prerequisite if we are to propose computational analysis and user-assisted data mining that use these large and shared data sets to gain insights and hypotheses.

3.2.1 User Populations and Lab Classification

A part of requirements analysis is to establish the potential users and their roles. The primary end users would be those researchers who are involved in any one of the INIA's research project components or core laboratories. Some other researchers informally involved in collaborations may be among the potential users. Eventually, other researchers that may want access to the public data may also become users. These users may be involved in more than one research component or core. Furthermore, people involved in the same component or core could be from different

physical locations and laboratories. These components may work together via Internet networking and teleconferencing. Hence a "laboratory" is defined as a "virtual" lab in this system and need not be located at a particular physical place. (For example the Bioanalytical Core includes researchers in both Tennessee and North Carolina.) Based on this definition, two types of INIA laboratories can be classified as follows:

- **Research labs:** A project just doing fundamental research is classified as a research lab in this system. Researchers in this type of lab are able to request services from any one of the core facilities.

- **Core (service) labs:** The second type of laboratories is a core (or service) lab. These cores may involve some research, but they also provide services, expertise, instruments, or other resources that other labs might be able to use to advance their research. Thus, researchers in this type of lab are funded by this project to help respond to service requests from other labs. They obtain service requests, do some sort of analysis or work, and provide the results of their work back to requesting labs. A core lab may also function as a research lab in that it may make a request to another core service if it needs help in the area of the other core's expertise.

The external level of the INIACS database consists of several user views which can be classified into two categories: research lab view and core lab view, so that researchers could access data through either one of them depending on what kind of lab they are acting as.

3.2.2 Research and Collaboration Processes

Another phase of requirements analysis is to examine the "business processes" that users may use. For this collaboration, most processes this project initially needs to support are transactions between research labs and core service labs. In a likely scenario, this consists of two overall processes. The fist set of processes involves a researcher from one research lab submitting a request to a core service lab. The second set of processes includes the core service lab responding to that request, doing the analysis, and returning the results to the research lab. In these collaboration processes, core labs can also play a research lab role (role transition state) if there is a need to request a service from other cores. An example of this would be the Bioanalytical Core asking the Informatics Core to provide computational analysis on microarray gene expression data. Figure 3.1 shows these collaboration processes among this INIA consortium.

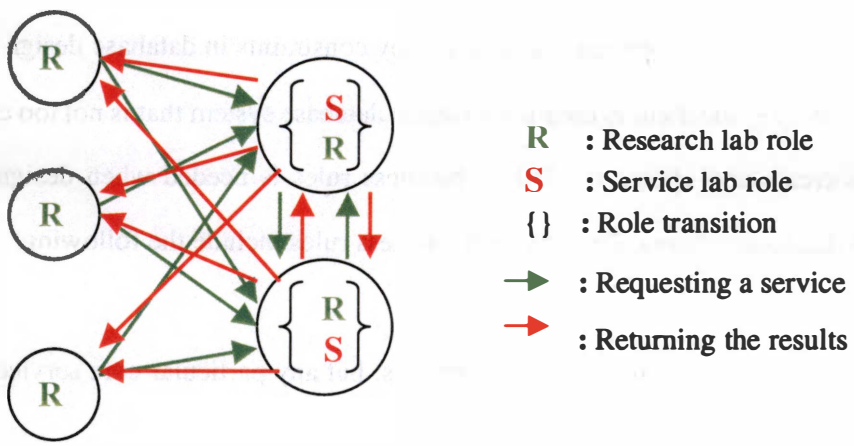


Fig. 3.1 Research and collaboration processes of the INIA

There is a need for more detailed processes, and some communications between research labs and core labs are also involved in the collaboration processes. For example, research labs might update their requests after submitting; alternatively, core labs may need to negotiate with research labs to decide when their samples can be sent or to request more samples if the received samples are in bad condition. More details on the workflow will be discussed in chapter four.

3.2.3 Business Rules

Doing requirements analysis developing "business rules" from the prior analysis of people and processes is helpful to guide development of the system. For example, these rules help create integrity constraints. These constraints are the rules that define the policy, procedure, or principle of the business for building the applications (15). Business rules are intended to be implemented in code or in the database that enforces the behavior of the system. In other words, business rules ensure that data in the database are correct, so that they help to preserve the data integrity of a database. In addition, business rules help designers to develop appropriate entity relationships and foreign key constraints in database design phases. They also help guide them in creating a simple database system that is not too complex. Thus, to create and update the INIA's business rules is needed when designing the INIACS database schema. The potential business rules include the following:

- One lab can provide multiple core services, but any particular core service can be offered only by one lab.

- One request can be submitted to only one service, but one service can accept multiple requests from different research labs at the same time.

- One request can have multiple updates, but one update can belong to only one request.

- One request can have multiple sample sets, but one sample set can belong to only one request.

- One sample set can include multiple sample items, but one sample item can belong to only one sample set.

- One sample item could be an electronic data file or a shipping sample.

- One response can belong to only one request, but one request could get multiple responses

- One experiment can be scheduled only for one request, but one request could require multiple experiments.

- One result set can be created only for one request, but one request can have multiple result sets.

- One result set can include multiple result items, but one result item can belong to only one result set.

- One result item could be an electronic data file, a resource web link or a gene identified in the analysis.

3.3 Entity Relationship Modeling

Entity-Relationship (ER) modeling is a popular high-level conceptual data model that was first introduced by Peter Chen in 1976 (26). This ER model presents the database entities and their relationships in a graphical structure that makes the complex database schema is more easily understood by designers, programmers and somewhat trained nontechnical users (15).

This model is used to put the requirements and the entity relationships into a meaningful diagram. First, however, the main entities of this system must be identified based on the business rules, so that the conceptual schema of INIACS database can be presented in an ER diagram.

3.3.1 Main Entities of INIACS

- **LABS:** As mentioned in section 3.2.1, in INIACS a lab is defined as a virtual lab derived from a research project component or a core facility of the INIA consortium. People involved in the same component or core are in the same lab despite their physical locations.

- **SERVICES:** This entity describes any service offered by a core lab.

- **REQUESTS:** This entity describes each request made by a research lab to get a particular service performed for a set of samples or set of data.

- **PROJECTS:** A large amount of information could be associated with one request and accessed by both of the research lab and core lab; this entity is responsible for grouping all of this information together. A project is created whenever a request has been made. Because one project is created for only one request, it has one-to-one relationship with the request entity.

- **PROJECT_STATUS:** This entity keeps a history of changes on the status of the request and project.

- **UPDATE_REQUEST:** During the collaboration process, researchers are able to update their requests if needed. This entity describes the update records and keeps the original request information in the request entity.

- **SAMPLE_SETS:** As mentioned in the business rules (section 3.2.3), one request can have multiple physical samples and data sets that are needed in processing the analysis. This entity is responsible for grouping multiple samples together into a sample set.

- **UPLOAD_FILES and SHIP_SAMPLES:** The system should have the capability to handle multiple types of samples. Here, two types of samples are defined as follows: **UPLOAD_FILES** entity describes the electronic type that needs to be uploaded in the operation system (OS). For example, microarray raw data in a

Microsoft Excel documents about the experimental details for the RNA samples belongs to this type. **SHIP_SAMPLES** describes the physical thing that needs to be shipped to the core lab, e.g. tissues, mice, DNA, RNA and proteins.

- **FORMATS:** This entity describes the format of the electronic data including sample or result files. For example, MAGE-ML is a XML based markup language to describe microarray gene expression data.

- **RESPONSES:** During the collaboration process, core labs might need to response to or communicate with research labs. For example, core labs need to contact with research labs if there were some circumstances occurred during the test. This entity provides core labs to be able to communicate with research labs.

- **EXPERIMENTS:** Multiple experiments might be needed for processing one single request. This entity describes the experiments performed for each project/request.

- **RESULT_SETS:** Multiple results might be produced from one single experiment. Similar to the SAMPLE_SETS entity, this entity is responsible for grouping multiple results into a set.

- **RESULT_BLOBS:** A major type of results is assumed to exist in this system. This type includes all kinds of electronic format data, e.g. images, text files, Excel files and XML files etc. In order to store the various types of data, a Binary File (BFILE), which is a type of SQL Large Object (LOB) data type, is used as an

attribute in this entity to indicate where the binary file is stored on the file system of the database server. This entity describes result items produced by core labs.

- **GKDB_LINKS:** Another goal of the INIACS Informatics Core is to provide INIA consortium a biological data mining environment. For example, to assist researchers in drawing the regulatory network of genes discovered from different analysis. To achieve this goal, the INIACS is proposed to integrate with the GeneKeyDB, a data mining environment developed in Dr. Jay Snoddy' lab. With the GeneKeyDB, researchers can analyze a set of genes at a time. This entity describes the characteristic of genes that are of interest in any data set from the analysis, (e.g. the LocusLink IDs of genes that are mutated from the mouse knockout core.) These keys can be used as the input of the GeneKeyDB application (this will be discussed in section 6.2).

- **WWW_LINKS:** This entity describes hypertext links by which the online resources might be related to a result. For example, the Neurohistology Core may produce very large Tagged Image File Format (TIFF) files that may not need to be stored directly in the database, but the URL could be used to retrieve the TIFF image from its location, if the smaller jpeg image is not adequate.

3.3.2 ER Diagram Representation

After identifying the main entities, an entity relationship diagram (ER diagram) can be drawn to represent the ER model of the INIACS database. The basic components of an ER diagram are as follows:

- **Entities:** An entity is shown as a rectangle containing the name of the entity. In general, an entity is mapped to a relational table in the logical design phase.

- **Relationships:** A relationship is represented by a diamond containing the name of the relationship. If two entities are related to each other, a diamond is used to connect the two rectangles, which represent the two entities. As mentioned in section 3.1, there are three types of relationships that an entity can have with other entities: one-to-one (1:1), one-to-many (1:N) and many-to-many (M:N). The type of a relationship is denoted next to each rectangle. In this project, the relationships can be defined based on the business rules mentioned in section 3.2.3.

- **Properties:** A property of an entity is a set of attributes that describe the overall characteristics of the entity. Attributes are represented by ovals connected to their entity.

Figure 3.2 shows the ER diagram that represents the conceptual schema of the INIACS database. In order to keep the diagram simple and easily understandable, the attributes of entities are omitted in this diagram. They will be defined in section 3.4.

3.4 Logical Data Model

After designing the INIACS conceptual schema, the next step of database design is the logical design phase where the ER model is transformed into a more detailed relational data model that maps each entity to a relational table. The attributes of each entity are determined in this step. Finally, the primary key and foreign keys of each table are also determined in this step.

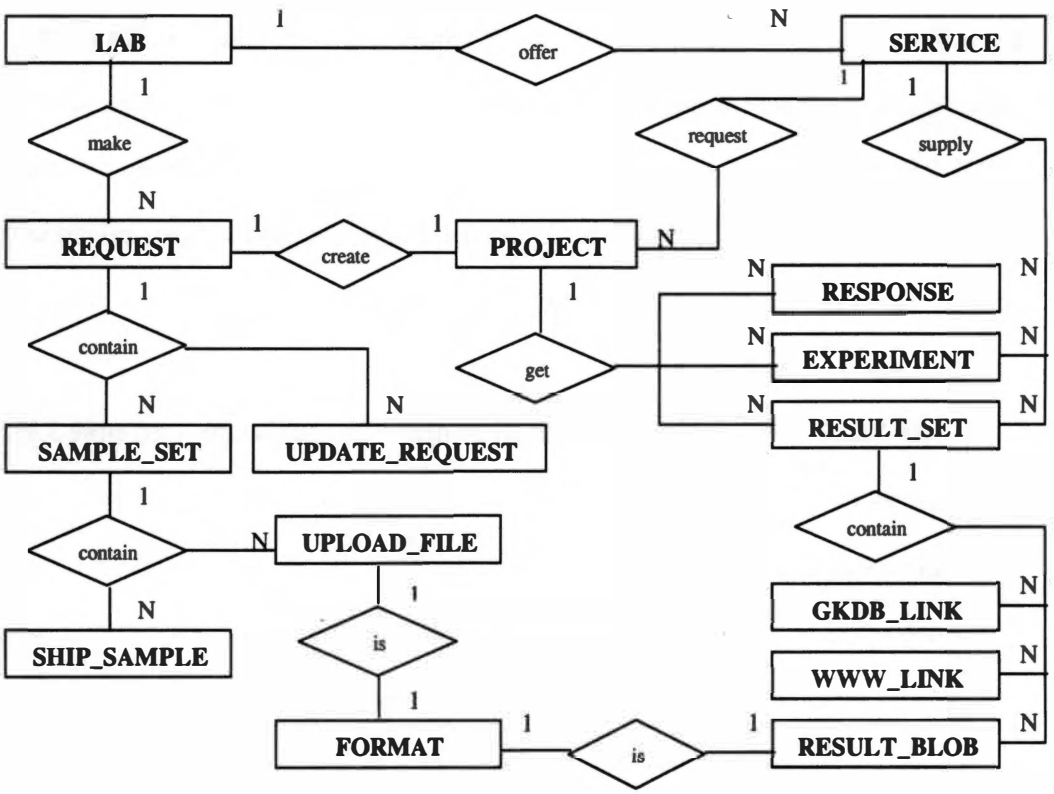


Figure 3.2: The entity relationship diagram (ERD) of the INIACS

Generally, to enforce that no tuple will be duplicated in a table, a unique identifier (ID), which is referred to as a primary key, will be assigned to each tuple. If two tables are related to each other, the primary key of the first table will appear as the foreign key in the second one to enforce the referential integrity. In addition, for a better integration with BioTeams software, some attributes are added in several tables for referring to the records stored in the BioTeams' database. For example, an "investigator" attribute is used in several tables to refer to the primary key (ID) of the "ADDRESSES" table in the BioTeams' database. This section is written to describe those tables that are mapped from the ER model. The attribute types are omitted in this section. However, they can be seen in Appendix B.

- **LABS Table:** The lab_id attribute is the primary key of this table (see table 3.1). In order to integrate with the BioTeams system, the tutos_pro_id attribute, which refers to a project created in the BioTeams, and the tutos_pro_creator attribute, which refers to the project creator, are needed to describe the relationships between the two systems (the integration of the two systems will be discussed in section 3.6.1). To enforce the system security and data security, each lab has one defined role and data access level. The Database Administrator (DBA) can grant certain object privileges to a lab according to its role. The access level attribute describes which public domain the lab can access (refer to chapter 3.6.3 and 3.6.4)

- **SERVICES Table:** The service_id attribute is the primary key of this table (see table 3.2). The lab_id attribute is needed as a foreign key to refer to a lab that offers this service.

Table 3.1: The definition of the LABS table

Attribute Name	Description
lab_id	A unique identifier
tutos_pro_id	A key referring to a product ID of BioTeams' products table
tutos_pro_creator	A key referring to the product creator user ID in BioTeams
lab_title	The Title of the lab
lab_login	The username of a database user account assigned to the lab
lab_password	The encrypted password of the database user account
enc_key	The key for encrypting the password
lab_addr	The mailing address of the lab
lab_role	The role of the lab, e.g. guest, client, service or administrator
access_level	The data access level of the lab
lab_desc	The description of the lab

Table 3.2: The definition of the SERVICES table

Attribute Name	Description
service_id	A unique identifier
lab_id	A key referring to the offering lab
service_title	The title of the service
service_desc	The description about the service

- **REQUESTS Table:** The request_id attribute is the primary key (see table 3.3). The lab_id, service_id and service_lab_id attributes are the foreign keys. The req_status attribute allows the request can be in pending status and then submitted sometime later.

- **PROJECTS Table:** The purpose of this table is to group all of the relevant data together about a single request (see table 3.4). The project_id attribute is the primary key of this table. There is a one-to-one relationship with the REQUESTS table; the request_id attribute is a foreign key in this table. Other foreign keys are request_lab_id, service_id and service_lab_id. The manager of a project is defined as an individual who made the request. The pro_status attribute allows research labs to know the process status of their requests. The process can be in one of the four states. These states are 1) waiting approval, 2) approval, 3) in processing and 4) result reported. The pro_active_mode attribute indicates that the project is ongoing or finished. The pro_access_mode attribute allows users to share their data with others (this will be discussed in section 3.6.3).

- **UPDATE_REQUEST Table:** The update_id attribute is the primary key of this table (see table 3.5). There is a one-to-many relationship with the PROJECTS table; the project_id attribute is a foreign key. The lab_id attribute is another foreign key to refer to the LABS table.

Table 3.3: The definition of the REQUESTS table

Attribute Name	Description
request _id	A unique identifier
lab_id	A key referring to the investigator's lab
service_id	A key referring to the requested service
service_lab_id	A key referring to the service offering lab
investigator	The person who made this request
req_status	The request status e.g. pending or submitted
stat_alt_person	The person who changed the request status
stat_alt_date	The date when the request status was changed
req_date	The date when this request was made
req_s_desc	The short description of the request (project title)
req_l_desc	The long description of the request

Table 3.4: The definition of the PROJECTS table

Attribute Name	Description
project_id	A unique identifier
request_id	A key referring to a request
request_lab_id	A key referring to the requester's lab
service_id	A key referring to the requested service
service_lab_id	A key referring to the service offering lab
manager	The person who made the request and created this project
req_date	The date when this project was created
pro_status	The process status, e.g. waiting approval and approval etc.
stat_alt_person	The person who changed the process status
stat_alt_date	The date when the process status was changed
pro_active_mode	The active mode of the project, e.g. ongoing or finished
act_alt_person	The person who changed the project active mode
act_alt_date	The date when the project active mode was changed
pro_access_mode	The public domain of the project where it is sharing to
acc_alt_person	The person who changed the data access domain
acc_alt_date	The date when the data access domain was changed

Table 3.5: The definition of the UPDATE_REQUEST table

Attribute Name	Description
update_id	A unique identifier
project_id	A key referring to a project
lab_id	A key referring to the investigator's lab
investigator	The person who made the update
upd_date	The date when the request was updated
upd_s_desc	The short description of the update (update tile)
upd_l_desc	The long description of the update

- **PROJECT_STATUS Table:** The status_id is the primary key of this table (see table 3.6). There is a one-to-many relationship with the PROJECTS table; the project_id attribute is a foreign key. The purpose of this table is to keep a history for the process status of a project.

- **SAMPLE_SETS Table:** The sample_set_id is the primary key (see table 3.7). There is a one-to-many relationship with the PROJECTS table; the project_id attribute is a foreign key.

- **UPLOAD_FILES Table:** The file_id attribute is the primary key (see table 3.8). The sample_set_id, upload_lab_id, service_lab_id and format_id attributes are the foreign keys to describe the relationships with SAMPLE_SETS, LABS and FORMATS tables. Because the uploading files are stored in the file system of the database server instead of storing in the database, the userfile attribute (a BFILE data type) is used to indicate the location where the file is stored. The rece_person and rece_date attributes allow users to track their sample movements.

Table 3.6: The definition of the PROJECT_STATUS table

Attribute Name	Description
status_id	A unique identifier
project_id	A key referring to a project
status	The process status of the project
investigator	The person who changed the process status
state_date	The date when the process status was changed

Table 3.7: The definition of the SAMPLE_SETS table

Attribute Name	Description
sample_set_id	A unique identifier
project_id	A key referring to a project
investigator	The person who created the sample set
sset_date	The date when the sample set was created
sset_s_desc	The short description of the sample set (sample set title)
sset_l_desc	The long description of the sample set

Table 3.8: The definition of the UPLOAD_FILES table

Attribute Name	Description
file_id	A unique identifier
sample_set_id	A key referring to a sample set
upload_lab_id	A key referring to the investigator's lab
servcie_lab_id	A key referring to the service offering lab for the project
format_id	A key referring to the format of the file
investigator	The person who uploaded the file
load_date	The date when the file was uploaded
userfile	A BFILE to point to the file stored in the file system
file_type	The type of the file, e.g. text file, image file or excel file
file_size	The size of the file
filename	The file name which is defined by the investigator
file_s_desc	The short description of the file
file_l_desc	The long description of the file
rece_person	The person in the core lab who received the file
rece_date	The date when the file was received by the core lab

- **SHIP_SAMPLES Table:** The ship_id attribute is the primary key (see table 3.9). The sample_set_id, ship_lab_id and ship_to_lab_id attributes are the foreign keys to describe the relationships with SAMPLE_SETS and LABS tables. The sample_type attribute is used to indicate the type of the sample. The rece_person and rece_date attributes allow users to track their sample movements.

- **FORMATS Table:** The format_id attribute is the primary key (see table 3.10). The format_dtd attribute allows the system to find out the Document Type Definition (DTD) of the format. For example, MAGE-ML is a XML based markup language to describe the microarray data. Its DTD can be found at http://cgi.omg.org/cgi-bin/doc?lifesci/01-11-02.

- **RESPONSES Table:** The response_id is the primary key (see table 3.11). The lab_id and project_id attributes are the foreign keys to describe the relationships with LABS and PROJECTS tables.

- **EXPERIMENTS Table:** The experiment_id attribute is the primary key (see table 3.12). The lab_id and project_id attributes are the foreign keys to describe the relationships with LABS and PROJECTS tables.

- **RESULT_SETS Table:** The result_set_id attribute is the primary key (see table 3.13). The lab_id and project_id attributes are the foreign keys to describe the relationships with LABS and PROJECTS tables. The rset_status attribute allows core labs to pre-login their results (unfinished status), and then report the result after collecting all of the data (finished status).

Table 3.9: The definition of the SHIP_SAMPLES table

Attribute Name	Description
ship_id	A unique identifier
sample_set_id	A key referring to a sample set
ship_lab_id	A key referring to the investigator's lab
ship_to_lab_id	A key referring to the lab where the sample shipped to
investigator	The person who shipped the sample
ship_date	The date when the sample was shipped
sample_type	The sample type, e.g. DNA, RNA, proteins, tissues or mice
ship_s_desc	The short description of the sample
ship_l_desc	The long description of the sample
rece_person	The person in the core lab who received the sample
rece_date	The date when the sample was received by the core lab

Table 3.10: The definition of the FORMATS table

Attribute Name	Description
format_id	A unique identifier
format_dtd	The location where the format DTD is stored
format_name	The name of the format
format_desc	The description of the format

Table 3.11: The definition of the RESPONSES table

Attribute Name	Description
response_id	A unique identifier
lab_id	A key referring to the service lab
project_id	A key referring to a project
investigator	The person who made the response
resp_date	The date when the response was made
resp_s_desc	The short description of the response
resp_l_desc	The long description of the response

Table 3.12: The definition of the EXPERIMENTS table

Attribute Name	Description
experiment_id	A unique identifier
lab_id	A key referring to the service lab
project_id	A key referring to a project
investigator	A person who scheduled the experiment
exp_date	The date when the experiment was scheduled
exp_s_desc	The short description of the experiment
exp_l_desc	The long description of the experiment

Table 3.13: The definition of the RESULT_SETS table

Attribute Name	Description
result_set_id	A unique identifier
lab_id	A key referring to the service lab
project_id	A key referring to a project
investigator	The person who created the result set
rset_status	The result status, e.g. finished or unfinished
service_title	The title of the requested service
service_lab_title	The title of the service offering lab
report_date	The date when the result set was created
rset_s_desc	The short description of the result set
rset_l_desc	The long description of the result set

- **RESULT_BLOBS Table:** The blob_id attribute is the primary key (see table 3.14). The result_set_id, format_id, lab_id attributes are the foreign keys to describe the relationships with RESULT_SETS, FORMATS and LABS tables. Similar to the UPLOAD_FILE table, the blob_file attribute is a BFILE data type to indicate the location where the file is stored.

- **GKDB_LINKS Table:** The gkdb_link_id attribute is the primary key (see table 3.15). The result_set_id and lab_id attributes are the foreign keys to describe the relationships with RESULT_SETS and LABS tables. The LLINK_ID attribute describes the LocusLink ID of a given gene that can be an input data when the GeneKeyDB is used. The acc_type attribute describes the accession type of the LocusLink ID, which could be protein, DNA or RNA.

- **WWW_LINKS Table:** The www_link_id is the primary key (see table 3.16). The result_set_id and lab_id attributes are the foreign keys to describe the relationships with RESULT_SETS and LABS tables. The url attribute describes the address at which the online resource is.

3.5 The Schema Definition in SQL

This section describes the style of the SQL DDL (Data Definition Language) syntax used in this project to transform the relational data model into database tables. The basic principles of transformation include transforming attributes into columns, choosing an appropriate data type for each of the columns, defining primary keys, foreign keys, and data or referential constraints.

Table 3.14: The definition of the RESULT_BLOBS table

Attribute Name	Description
blob_id	A unique identifier
result_set_id	A key referring to a result set
format_id	A key referring to the format of the result file
lab_id	A key referring to the investigator's lab
investigator	The person who uploads the result file
blob_file	A BFILE to point to the file stored in the file system
blob_type	The type of the result files
blob_size	The size of the result files
filename	The file name which is defined by the investigator
load_date	The date when the result file was uploaded
blob_s_desc	The short description of the file
blob_l_desc	The long description of the file

Table 3.15: The definition of the GKDB_LINKS table

Attribute Name	Description
gkdb_link_id	A unique identifier
result_set_id	A key referring to a result set
lab_id	A key referring to the investigator's lab
investigator	The person who added the link
llink_id	The LocusLink ID of the gene
acc_type	The accession type of the link, e.g. protein, DNA or RNA
link_date	The date when the link was added
gene_s_desc	The short description of the gene
gene_l_desc	The long description of the gene

Table 3.16: The definition of the WWW_LINKS table

Attribute Name	Description
www_link_id	A unique identifier
result_set_id	A key referring to a result set
lab_id	A key referring to the investigator's lab
investigator	The person who added the link
url	The address of the link
link_date	The date when the link was added
link_s_desc	The short description of the link
link_l_desc	The long description of the link

According to the logical data model defined in the previous section, sixteen tables need to be defined by using DDL "CREATE TABLE" statement. Two examples shown below illustrate the SQL syntax used in the ORACLE 8*i* RDBMS to create database tables.

In figure 3.3, the SQL syntax can be divided into two parts. In the first part (from line 2 to 11), a suitable data type is assigned to each attribute such as varchar2 or number followed by a 'not null' constraint or a default value. The syntax is given below:

- **Syntax (1) for defining an attribute:**

  1. attribute-name **DATATYPE** (length) OR { **NOT NULL** / **DEFAULT** value }

The second part is to define the data and referential constraints such as primary key, unique and check (from line 13 to 19). The syntax is given below:

```
1.  CREATE TABLE LABS (
2.      lab_id              varchar2(16)        not null,
3.      lab_title           varchar2(50)        not null,
4.      lab_login           varchar2(200)       not null,
5.      lab_password        varchar2(200)       not null,
6.      lab_addr            varchar2(250)       default null,
7.      lab_role            number              not null,
8.      access_level        number              default 1,
9.      lab_desc            varchar2(4000)      default null,
10.     tutos_pro_id        number,
11.     tutos_pro_creator   number,
12.
13.     CONSTRAINT  labs_pk   PRIMARYKEY(lab_id),
14.     CONSTRAINT  labs_uk1  UNIQUE(lab_id),
15.     CONSTRAINT  labs_uk2  UNIQUE(tutos_pro_id),
16.     CONSTRAINT  labs_uk3  UNIQUE(lab_title),
17.     CONSTRAINT  labs_uk4  UNIQUE(lab_login),
18.     CONSTRAINT  labs_ck1  CHECK(access_level IN (1,2,3)),
19.     CONSTRAINT  labs_ck2  CHECK (lab_role IN (0,1,2,3))
20. );
```

Figure 3.3: SQL syntax used to create LABS table

- **Syntax (2) for defining a constraint:**
  1. **CONSTRAINT** constraint-name
  2. **CONSTRAINT-TYPE** (attribute-name **IN** (values))

From line 14 to 19 in figure 3.3, it is shown that each lab should have a unique lab ID, a unique project ID created in the BioTeams system, a unique title, a unique database login name and a unique password. In addition, the access_level (refer to section 3.6.3) and lab_role (refer to section 3.6.4) are defined there as a number data type with a check constraint. The potential values of these two attributes are as follows:

- **ACCESS_LEVEL:**

  1 = Lab can access data public to INIA-East, INIA-All and World domains
  2 = Lab can access data public to INIA-All and World domains
  3 = Lab can only access data public to World domain

- **LAB_ROLE:**

  0 = Lab is an administration core

  1 = Lab is a core lab

  2 = Lab is a research lab

  3 = Lab is a one with the guest role

Figure 3.4 shows the table definition of the SERVICES table. In the previous section, the ER diagram (see figure 3.2) shows that there is a one-to-many relationship between labs and services entities. Because the services entity is on the "many" (called child) side, the lab_id is needed as a foreign key in the SERVICES table. The syntax shown below is used to define foreign keys.

- **Syntax for defining a foreign key:**

  1. **CONSTRAINT** constraint-name **FOREIGN KEY** (attribute-name)
  2. **REFERENCES** table-name (attribute-name)

The command of these two lines forces the system to follow the rule of referential integrity. For example, a service cannot be offered by a lab that does not exist in the LABS table.

```
1.  CREATE TABLE SERVICES (
2.      service_id      varchar2(16)     not null,
3.      lab_id          varchar2(16)     not null,
4.      service_title   varchar2(50)     not null,
5.      service_desc    varchar2(4000)   default null,
6.
7.      CONSTRAINT      services_pk      PRIMARY KEY(service_id),
8.      CONSTRAINT      services_uk1     UNIQUE(service_id),
9.      CONSTRAINT      services_uk2     UNIQUE(service_title),
10.
11.     CONSTRAINT      services_fk      FOREIGN KEY (lab_id)
12.                                      REFERENCES LABS (lab_id)
13. );
```

Figure 3.4: SQL syntax used to create SERVICES table

3.6 Implementation of Database Design

At the time of writing, the INIACS database has been created in ORACLE 8*i* RDBMS according to the database schema designed in the previous sections. Besides the creation of database tables in a RDBMS, however, there are many other issues that have to be addressed in the database design, e.g. system and data security issues in the multiple users environment. These remaining issues needed for implementation will be discussed in following sections.

3.6.1 System Access and User Authentication

The INIACS is designed to allow multiple users to access the database at the same time. Frequently, the solution in such kind of multiple users system is to create a user table for recording each user's personal information including username and password. In this case, when an end user initiates a login to the system from the application by providing his personal username and password, the application will login to the RDBMS using a default database user account and then authenticate the end user's accessing privilege by comparing the username and password with those stored in the user table. In other words, end users actually share one database user account that was created by DBA using the "create user" DDL command.

The above method can work but it provides system and data security only on the application level, not on the database level.

To address this problem, more than one database user will be created in order to restrict users' object privileges (e.g. insert, select or delete) on the database level by using database roles (this will be discussed in section 3.6.4). Since the fundamental

unit of the INIACS database is a lab, all members in a lab are assumed to have the same privileges to access data that belongs to the lab. With this assumption, a database user account is created for each lab and all members in a lab use the same account for accessing database. In such way, the INIACS is designed to provide researchers with a multiple users environment using database roles to control users' privileges.

However, this solution also poses another problem. Because there is no table directly in the system that provides the information about the members of a lab, it would be difficult for system to know who can use which user account to login. Hence, a user authentication method is needed to check users' access privileges when they login to the system.

Fortunately, the BioTeams system can be used as a tool to manage the human resources of the INIA consortium (refer to section 2.4). This system acquires these data, as well. To approach the problem mentioned above, a lab in the INIACS will be virtually derived from an INIA project created in the BioTeams system and each project has a unique identifier (pro_id). In addition, each INIA researcher must have an entry (called an address) in BioTeams so that each researcher can be identified by his unique address ID (adr_id). In such way, the relationships between researchers and labs can be identified through the BioTeams database.

A user authentication process used in this system is presented here (see figure 3.5). When a user logins the INIACS from the user interface, the system will enforce the application to insert some information about the current user (e.g. the username used to login and the user's adr_id) into a temporary table called SESSION_TABLE via SESSION_TABLE_VIEW which will fire an "INSTEAD OF" trigger called IOFIUFER_SESSION_TABLE_VIEW to check if the current user is an authorized user or not by querying BioTeams database.

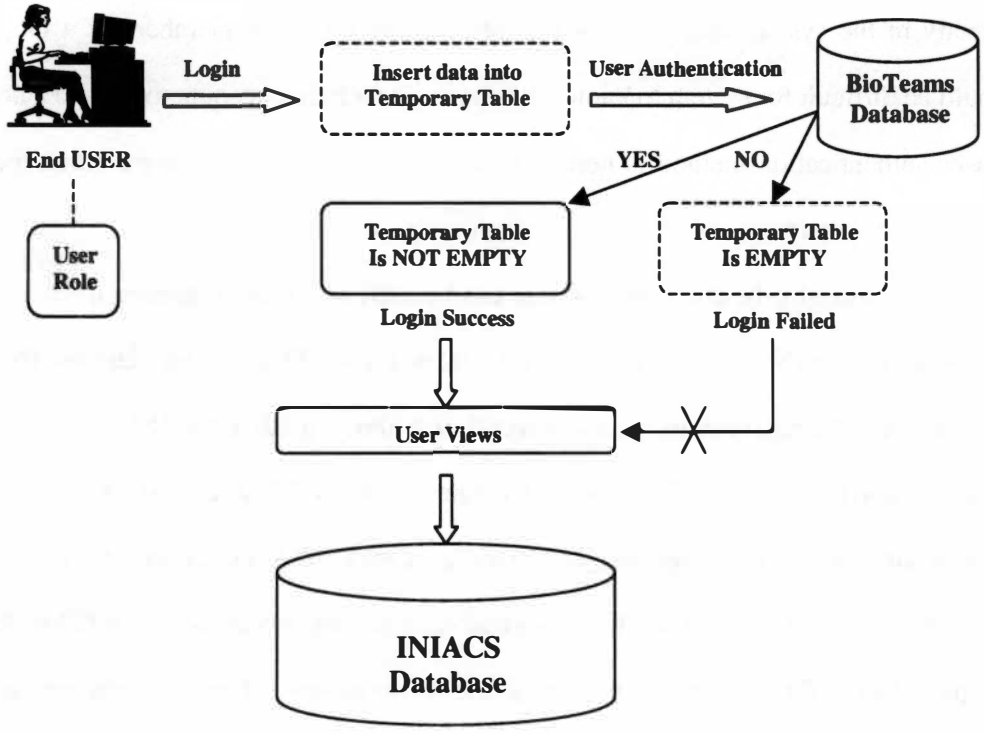


Figure 3.5: User authentication process

Because the SESSION_TABLE is a temporary table whose data exist only for the life of a session, the system is able to identify the current user through this table. If the current user is a member of the lab, the inserting is executed and then the user's information will exist in the SESSION_TABLE. Otherwise, the inserting fails indicating that the current user is not authorized to login to the INIACS. Figure 3.5 illustrates the authentication process mentioned above.

3.6.2 Data Access through Views

The INIACS system allows multiple users to access stored data at the same time. Sometimes, users might need to query database though complicated relationships across multiple tables. For example, if researchers want to track their sample shipments through particular project IDs, they need to query three tables, which are PROJECTS, SAMPLE_SETS and SHIP_SAMPLES tables. More complicated queries than this could be met and would possibly decrease system performance, especially when multiple users access the same data at the same time. To address this problem, the system provides users with several defined views (called relational views) to hide the complexity and enhance performance.

A relational view is a virtual table that does not store data but just stores a query definition. Only records with selected fields that meet the definition will be placed in this virtual table at the running time so that it can be used to enhance performance. An example is given below to solve the query problem above by providing a view called SHIP_TRACK_VIEW (see figure 3.6).

```
1.   CREATE VIEW SHIP_TRACK_VIEW as
2.
3.       SELECT p.project_id,
4.              s.sample_set_id,
5.              s.ship_id,
6.              s.ship_date,
7.              s.rece_person,
8.              s.rece_date
9.
10.      FROM   PROJECTS p, SAMPLE_SETS ss, SHIP_SAMPLES s
11.
12.      WHERE  s.sample_set_id=ss.sample_set_id
13.      AND    ss.project_id=p.project_id
14.
15.      AND    (p.request_lab_id= (select lab_id from labs
16.                 where lab_login= ( select lab_login from session_table
17.                                    where lab_login=USER ) )
18.
19.      OR     p.service_lab_id= (select lab_id from labs
20.                 where lab_login= ( select lab_login from session_table
21.                                    where lab_login=USER ) ) )
22.  /
```

Figure 3.6: SQL syntax used to create SHIP_TRACK_VIEW

The command lines (1 - 13) are the query definition to track sample movements. In addition, a view can be used to enhance data security by defining who can access the data. In this example, the definition from line 15 to 17 allows members of the requesting lab to access the data, and the definition from line 19 to 21 allows members of the core lab that has been requested to access the data. The same definition can also be seen in most of the relational views created in this system and end users can access data only through these views but not those base tables.

3.6.3 Data Sharing Method

In section 3.4, an attribute called access_level in the LABS table is defined for determining the data access level of a lab. Initially in this system, three data access levels that a lab can have are defined as follows: INIA-East, INIA-West and World. Through this access_level attribute, the access privilege of a lab on shared data can be

restricted.

As mentioned in section 1.1, there were two INIA consortia (INIA-East and INIA-West) founded by the NIH. In this system, labs belonging to INIA-East consortium are in the INIA-East access level that allows them to access any sharing data including that is public to INIA-East, INIA-All or World domains (refer to section 2.3.7). Likewise, labs belonging to INIA-West consortium are in the INIA-West access level that restricts them to accessing only data that are public to INIA-All or World domains. Otherwise, if a lab does not belong to either one of the consortia, then this lab is in the World access level and its members can access only the data that are public to the World domain. The public domain of data can be defined in the pro_access_mode attribute in the PROJECTS table. Both of the access_level and pro_access_mode attributes are NUMBER data types with CHECK constraints to restrict their potential values as follows:

- **ACCESS_LEVEL**
  1 = Lab in INIA-East access level
  2 = Lab in INIA-West access level
  3 = Lab in World access level

- **PRO_ACCESS_MODE**
  0 = Data can be accessed by requesting labs and cores which receive the requests
  1 = Data sharing with labs in INIA-East access level
  2 = Data sharing with labs in INIA-East or INIA-West access level
  3 = Data sharing with labs in INIA-East, INIA-West or World access level

A view called PUBLIC_PROJECT_VIEW (see figure 3.7) is created to define a rule to ensure that the method of data sharing proposed above is correctly implemented.

```
1.  CREATE VIEW PUBLIC_PROJECT_VIEW as
2.
3.      SELECT p.project_id
4.             p.request_lab_id,
5.             rl.lab_title as "REQ_LAB_TITLE",
6.             p.service_id,
7.             s.service_title,
8.
9.
10.     FROM   PROJECTS p,REQUESTS r, LABS rl, SERVICES s, LABS sl
11.
12.     WHERE pro_access_mode >= (select access_level from LABS
13.                where lab_login= ( select lab_login from session_table
14.                                   where lab_login = USER ) )
15.
16.     AND    r.request_id = p.request_id
17.     AND    p.request_lab_id = rl.lab_id
18.     AND    p.service_id = s.service_id
19.     AND    p.service_lab_id = sl.lab_id
20. /
```

Figure 3.7: SQL syntax used to create PUBLIC_PROJECT_VIEW

In Figure 3.7, part of the query definition is omitted. The WHERE clause of the SQL statement (line 12 to 14) defines the rule that can be translated into the following English sentence:

IF (the value of project access mode is larger than or equal to the value of lab access level) THEN (the lab can access the project)

An example is given below: Assuming that a login lab is in the INIA-West access level (access_level=2). Then the members of the lab can access the sharing projects that are public to the INIA-All domain (pro_access_mode=2) or public to the World domain (pro_access_mode=3), because the pro_access_mode is larger than or equal to the access_level. In such way, those shared resources can be accessed via this view.

3.6.4 Database Roles and Security

As mentioned in section 3.6.1, each lab uses one particular user account to login to the INIACS. This indeed enhances the system security on the database level. However, in order to ensure the data security, certain database privileges on those user accounts need to be restricted, e.g. one database user cannot "write" data into a particular table but another database user not only can "read" data from the table or view but also can "write" data into the base table. This requirement can be accomplished by using database roles to control users' privileges.

A database role is a database object, which collects either object or system privileges such as select, insert or update privileges on some particular tables or views (27). A database administrator can define database roles of the system and then grant defined roles to database users based on their jobs in the organization. In this system, four database roles have been created as follows.

- **INIA_ADMIN_ROLE:** A collection of all privileges needed for administrators
- **INIA_SERVICE_ROLE:** A collection of all privileges needed for core labs
- **INIA_CLIENT_ROLE:** A collection of all privileges needed for research labs
- **INIA_GUEST_ROLE:** A collection of all privileges needed for labs other than core or research labs

Several relational views have been created for accessing data stored in the base tables. In order to restrict user's privileges on views, the SQL "grant" command is used to grant certain privileges on those relational views to roles. For example, a view called SHIP_RECEIVE_VIEW, which can be accessed only by core labs, will allow

core labs to update the rece_person and rece_date columns in the SHIP_SAMPLES table. Hence, the select and update privileges on the view need to be granted to the INIA_SERVICE_ROLE. In addition, as mentioned in section 3.2.2, according to the collaboration process, core labs can request services from other cores. Thus, the privileges that a research lab has should also be granted to core labs in that a core is able to request services from other cores. A granting plan can be seen in figure 3.8 that shows privileges can be granted from one role to another one. In any event, the INIA_ADMIN_ROLE has all privileges and the INIA_SERVICE_ROLE also has privileges that the INIA_CLIENT_ROLE has.

3.6.5 Recording a Log of Changes to a Table

A table called "LOGS" is created to record logs by firing triggers when certain tables are modified (see figure 3.9). For certain columns, it will also record the original value and the new value. This can be used as a backup plan for a record that is changed by mistake and allow modest traceability in changes to the data.

The LOG_ID is a unique identifier for each change. The following information can be retrieved from this table:

- Who did this change and when?
- Where did the user login the system?
- Which table was changed?
- What kinds of actions (inserts, update or delete) were performed?
- Which record was changed?
- Which column was modified?
- What are the original and modified values?

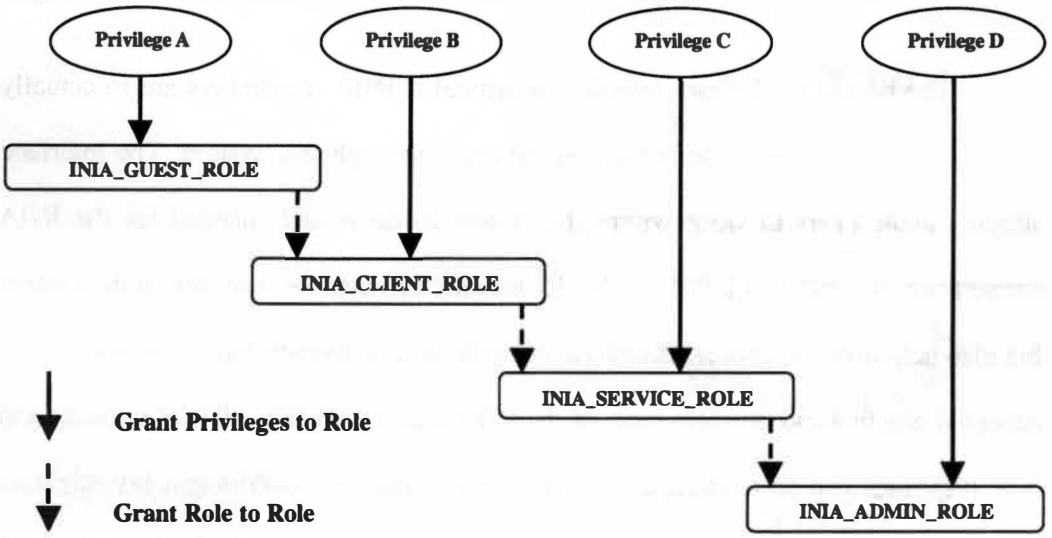


Fig.3.8 The plan for granting privileges to database roles

```
1.  CREATE TABLE LOGS (
2.
3.      log_id          varchar2(16)     not null,
4.      log_lab_login   varchar2(20)     not null,
5.      log_date        date             default sysdate,
6.      remote_addr     varchar2(20)     default null,
7.      log_person      number,
8.      alt_table       varchar2(32)     not null,
9.      action          varchar2(8)      not null,
10.     rec_id          varchar2(16)     not null,
11.     alt_col         varchar2(50)     default null,
12.     old_val         varchar2(4000)   default null,
13.     new_val         varchar2(4000)   default null,
14.
15.     CONSTRAINT      logs_pk   PRIMARY KEY (log_id),
16.     CONSTRAINT      logs_uk   UNIQUE (log_id)
17. )
18. /
```

Figure 3.9: SQL syntax used to create LOGS table

Chapter 4 Interface Design

The design of the user interface is critical if INIA researchers are to actually access and utilize these distributed core facilities through this system. The interface should enable users to work within the system to do what is needed for the INIA consortium. A good user interface should not only increase the usability of the system but also help users do processes and automate tasks in a manner that is consistent with accepted practice and business rules of the INIA consortium (e.g. all user requests and core responses can be tracked and summarized so that the co-Principal Investigators (coPIs) and the program managers can ensure that the requests are all appropriate and all the responses are timely).

The primary criterion for a good user interface largely is to support the goals of the users. In general, a user interface must make a user's job easier than by other methods and it should be easy to use. Users should be able to quickly learn how to use the tool. It should be clear to the users how they could use the system to automate routine tasks or otherwise facilitate their work. The INIACS user interface is designed to be reasonably straightforward to use.

In order to meet various requirements of users from different labs with different roles, the INIACS interface is also designed to provide different functionality to users according to the role of their lab(s) in the collaboration. For example, research labs need the functionality to request a service but the core labs need the functionality to receive and respond to the request.

4.1 Client/Server Networking

As mentioned in section 2.2, the INIACS is built according to a classic client/server architecture with networking support. The basic concept of a client/server networking is that one computer (client) makes requests from another computer (server) on the network, and then the server responds to the client with the requested information (see figure 4.1) (15). In the INIACS, the server side is a database built in Oracle 8*i* RDBMS which has been discussed in chapter three, and the client side is a set of applications including web browsers and a user interface, which directly interact with end users. We are currently anticipating that users would use the web browser as the client, but communications between clients and servers would be accomplished by a middleware component including an application programming interface (API) that is responsible for connecting to the database, for example using the Oracle Call Interface (OCI) for connecting to the Oracle 8*i* RDBMS.

This chapter describes the INIACS user interface that allows end users to interact with the system.

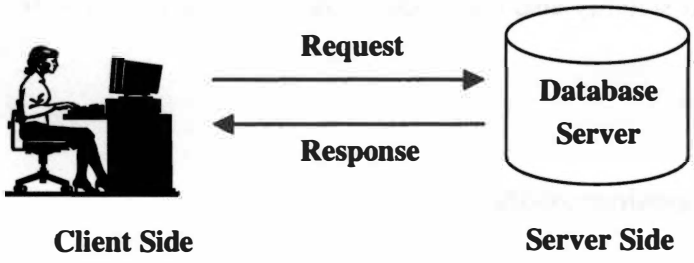


Figure 4.1: The basic concept of the client/server networking

4.1.1 Client-Side Technology: Using PHP

In order to support the interactions between research labs and core labs during the research process, the INIACS interface should be interactive. In other words, an appropriate programming language needs to be chosen in order to generate "dynamic" web pages instead of the "static" pages written in raw HyperText Markup Language (HTML) that can be driven from the database.

In this project the user interface is developed using PHP (PHP Hypertext Preprocessor) programming language, which meets the requirements described above. PHP is a cross-platform, HTML-embedded and server-side web scripting language (28). Several features of PHP programming language are as follows: First, PHP is simple and easy to learn, and also it is free and open source so that the designers do not need to rely on a third provider. Second, it supports connections to a wide variety of databases including Oracle, MySQL and PostgreSQL by using OCI, ODBC, JDBC or other database-specific APIs. Moreover, because PHP scripts are run on the server-side, users should be able to operate the application on the Web without installing application code on their computers. In such way, the interface can be easily maintained and modified by the designer. Other features including high speed, portability and stability also contribute to the reasons for using PHP in building this interface.

4.2 User Considerations

Most of the potential users in the INIACS are experimental biologists. Although some of them might have some database experience, most users do not have

one. Thus, to the extent that is feasible, the INIACS interface should provide users with an easy way to run complicated operations. Because most of the users should have some experience in using a web browser, e.g. knowing how to navigate web sites, these considerations will guide us to design a web-base interface that is easy to use. However, it will be difficult to meet all users' requirements, especially in such large-scale collaborations. Thus, this interface will still need to be tested, evaluated and improved.

4.3 Interface Navigation

In order to help achieve the goal of easy navigation, a menu of the INIACS interface is generated by a dynamic tree menu program, which is developed in an open source project named "HTML_TreeMenuXL" (29). This program generates a menu in a tree structure and can be expanded. The folders (or called "nodes") on the first level of the tree are the main entries for particular functionality. Multiple links related to the same functionality can be added on the second or deeper level. This folder and subfolder structure is a common navigation metaphor used in a number of systems familiar to most users. Because the two types of labs need different functionality to perform their jobs in the collaboration, two types of the tree menus are created: one for research labs and the other one for core labs (see figure 4.2). Depending on the role of the login lab, only one type of menu will be seen on the main screen.

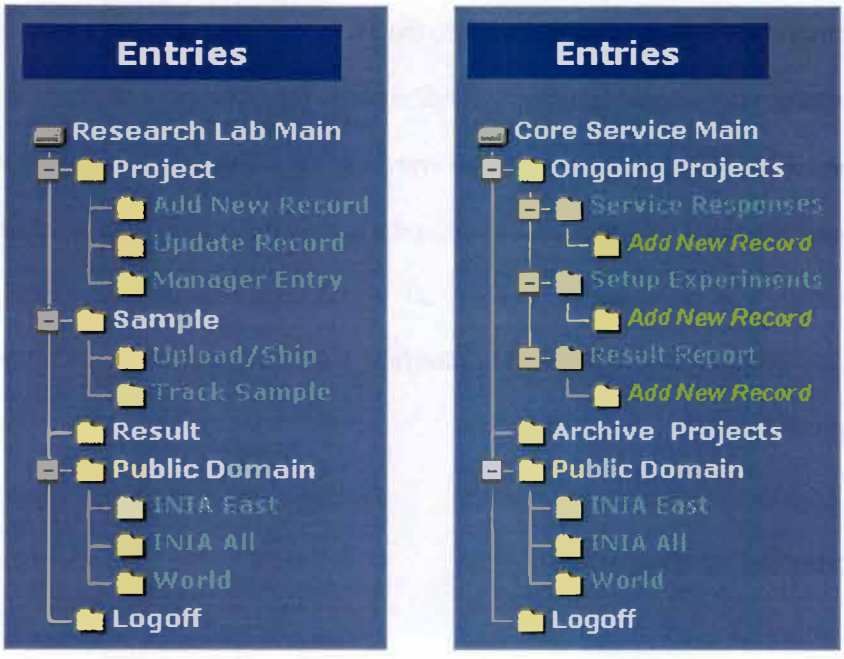


Figure 4.2: Two types of tree menus: Research Lab Menu (left) and Core Lab Menu (right)

4.3.1 Research Lab Menu

This type of menu provides research labs with all of the functions needed to request a service from core labs. Users can easily navigate the interface through the entire workflow such as to request a service or to upload or ship samples. The workflow will be discussed in section 4.4. The nodes on the first level include "Project", "Sample", "Result", "Public Domain" and "Logoff". The functionality of each node is described below:

- **Project:** This node allows users to request a service and also to manage their projects. Users can access these functions through the nodes on the second level which are described as follows: The "Add New Record" node allows them to fill in a requesting form and submit it to core labs. The "Update Record" node allows

users to update their requests. The "Manager Entry" node allows users to submit requests or share their data with others.

- **Sample:** This node allows users to upload or ship samples. The "Upload/Ship" node on the second level is responsible for creating sample sets and loading samples into sample sets. In addition, users can track the sample movements through the "Track Sample" node.

- **Result:** This node allows research labs to retrieve results from the database.

- **Public Domain:** This node facilitates data sharing. As mentioned in the previous chapter, three public domains are defined as follows: INIA-East, INIA-All and World. The three nodes on the second level are the entries for these three domains.

- **Logoff:** This node allows users to logoff from this system.

4.3.2 Core Lab Menu

This menu provides core labs with all of the functions needed to process requests, including responding to requests and reporting results to research labs. The nodes on the first level include "Ongoing Projects", "Archive Projects", "Public Domain" and "Logoff". The functions of "Public Domain" and "Logoff" nodes are as same as those in the research lab menu. The functionality of "Ongoing Projects" and "Archive Projects" are described below:

- **Ongoing Projects:** This node provides core labs with the functionality to work on unfinished requests (defined as ongoing projects in this system). The functions needed to process a request can be accessed through the following nodes on the second level. The "Service responses" node allows core labs to give feedback on the requests. The "Setup experiments" node allows core labs to schedule their experiments needed to process the requests. The "Result report" node allows core labs to report their results to the research labs. The "Add New Record" node on the third level provides a fill-in form for them to record their data into the database.

- **Archive Projects:** This node is responsible for archiving projects, which have been finished and are no longer active.

This program can also allow the interface developers to add more nodes in an appropriate place with any additional functionality that is requested by users. Those additional nodes should be visible to all users interested in a particular area of functionality.

4.4 User Interfaces to Support Workflow

The workflow schema used in the INIACS is discussed in this section (see figure 4.3). Workflow is a concept to describe how information flows through the system. This concept also defines the basic business rules followed by the application. The flow chart in figure 4.3 shows how the system delivers data from research labs to core labs and where the researcher interacts with the system.

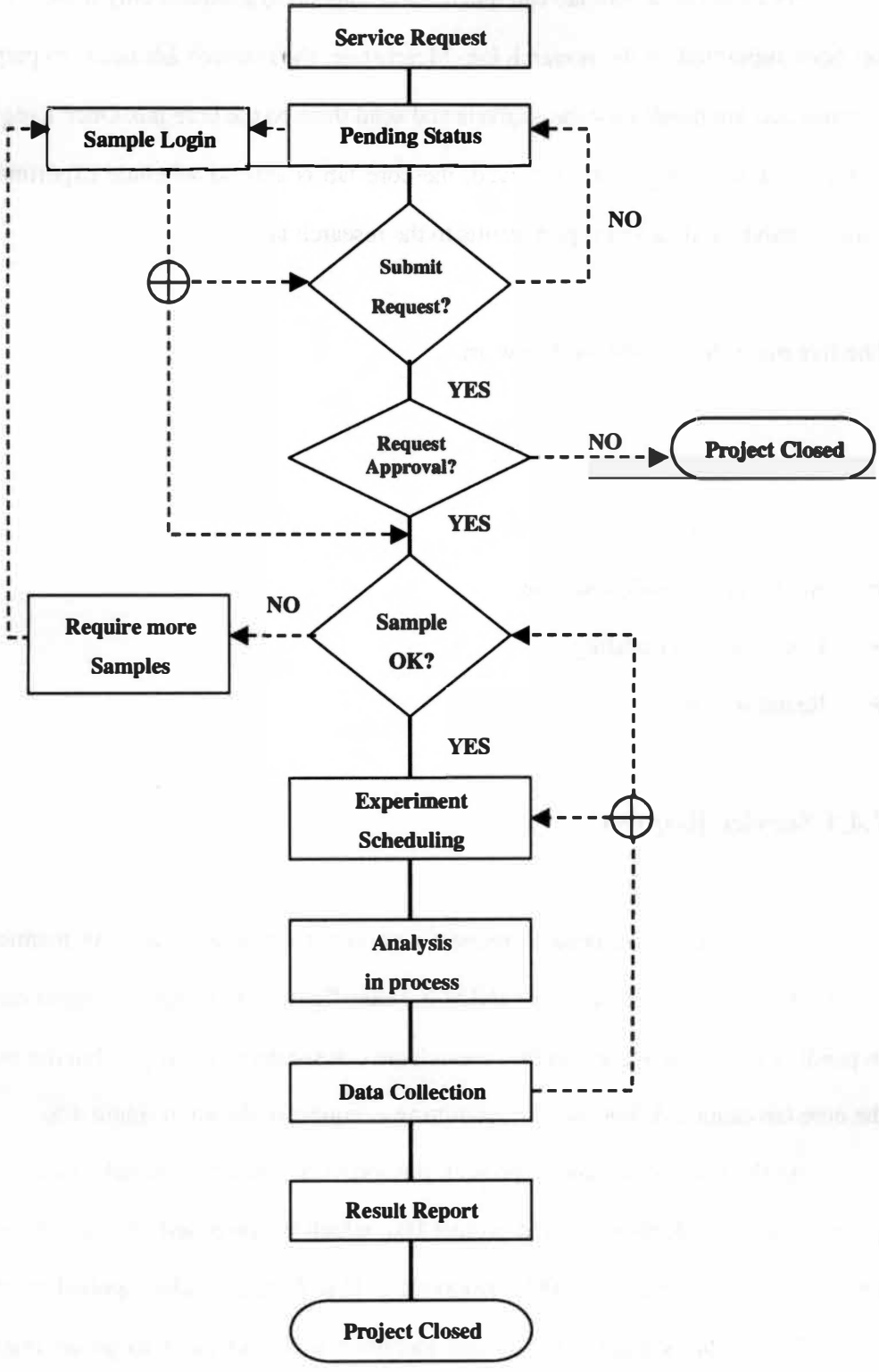


Figure 4.3: INIACS workflow schema

For example, a core lab can receive and respond to a request only if the request has been submitted by the research lab. Meanwhile, the research lab needs to prepare samples that are needed for the analysis and send them to the core lab. Once a request is approved and samples are received, the core lab is able to schedule experiments, collect analytical data and report results to the research lab.

The five main steps in the workflow are:

- Service Request
- Sample Login
- Request Approval/Response
- Experiment Scheduling
- Result Report

4.4.1 Service Request

In this step users need to request a service from a core lab. (As mentioned earlier, this creates a project in the INIACS.) (See figure 4.4) A request/project can be in pending status while users in the research lab can see it and modify it, but the one in the core lab cannot. A function for submitting a request is shown in figure 4.5.

At the time of creating a project, the INIACS will automatically assign each project a unique identifier (called project ID), which is composed of four characters and twelve digits, e.g. "pro_000000000001". This format is also applied to other record IDs in this system. This unique identifier will help users to group relevant collaborative data together.

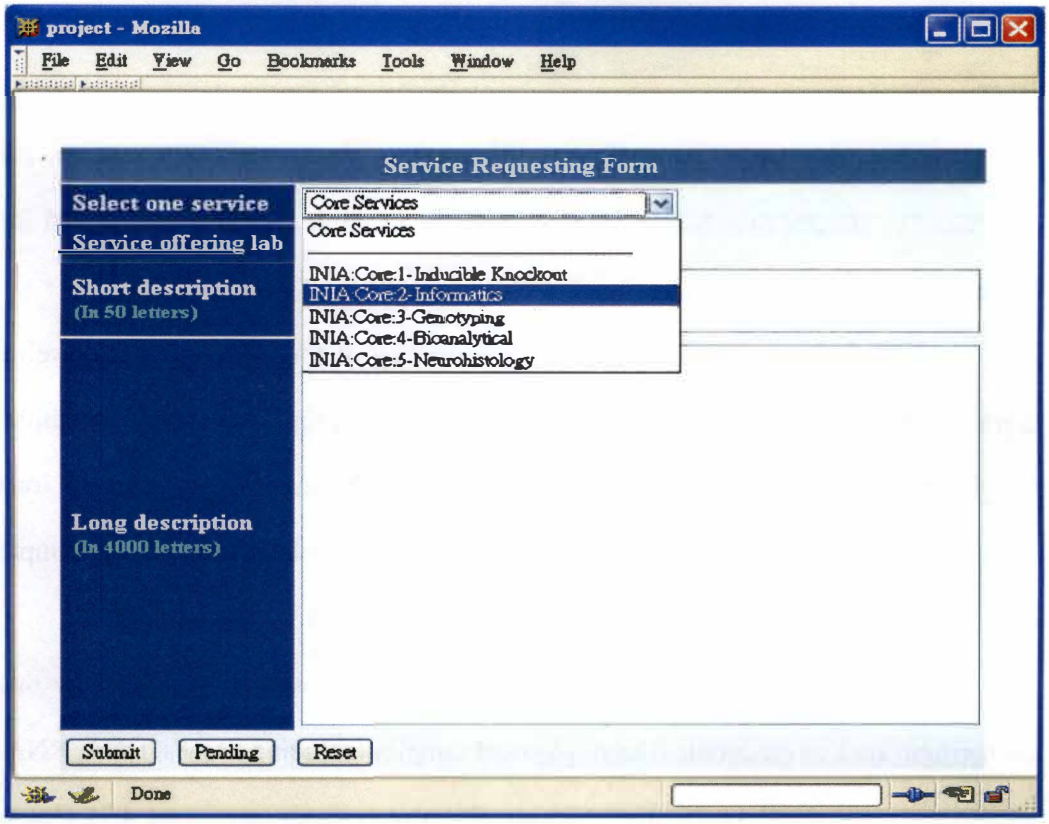


Figure 4.4: A fill-in form for requesting a service

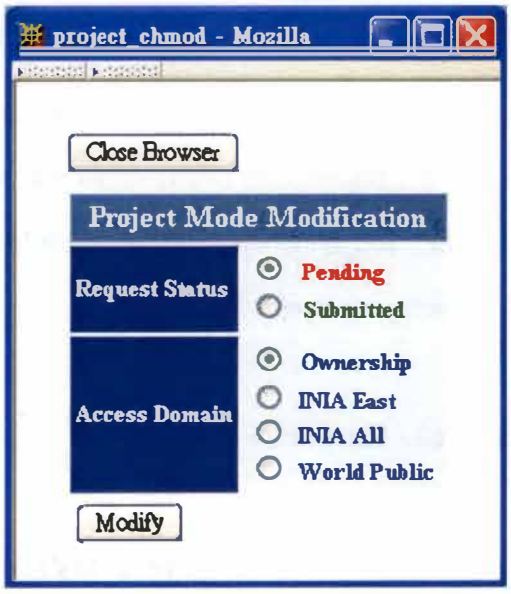


Figure 4.5: A function for submitting a request. This figure also shows the function for sharing data to different public domains.

4.4.2 Sample Login

In this step users are able to load samples into the INIACS system. As mentioned in chapter two, before loading samples, a sample set has to be created for organizing multiple samples (see figure 4.6).

An identifier (called sample set ID), consisting of four characters and twelve digits, will be automatically assigned to each sample set. For example, "sst_000000000001" is a sample set ID. The INIACS also allows users to load samples separately at different time by assigning those samples to an existing sample set.

Because of the wide variety of sample types that will be produced by this consortium, such as electronic files or physical samples including tissues, DNA, RNA, protein and mice, the INIACS offers two sample entry methods (see figure 4.7). One is for uploading electronic files (see figure 4.8); another one is for shipping the physical samples (see figure 4.9). Both types of samples could be grouped into a sample set. For the purpose of better sample management, two types of sample IDs are created. One type with prefix 'lod_' followed by twelve digits will be assigned to uploading files; another type with prefix 'shp_' followed by twelve digits will be assigned to shipping samples. For example, the "lod_00000000001" is an uploaded file ID and the "shp_000000000001" is a shipping sample ID.

4.4.3 Request Approval / Response

Once a request is submitted, the core lab needs to decide if it has the time and resources to carry it out. If the core lab must agree, it so indicates by changing

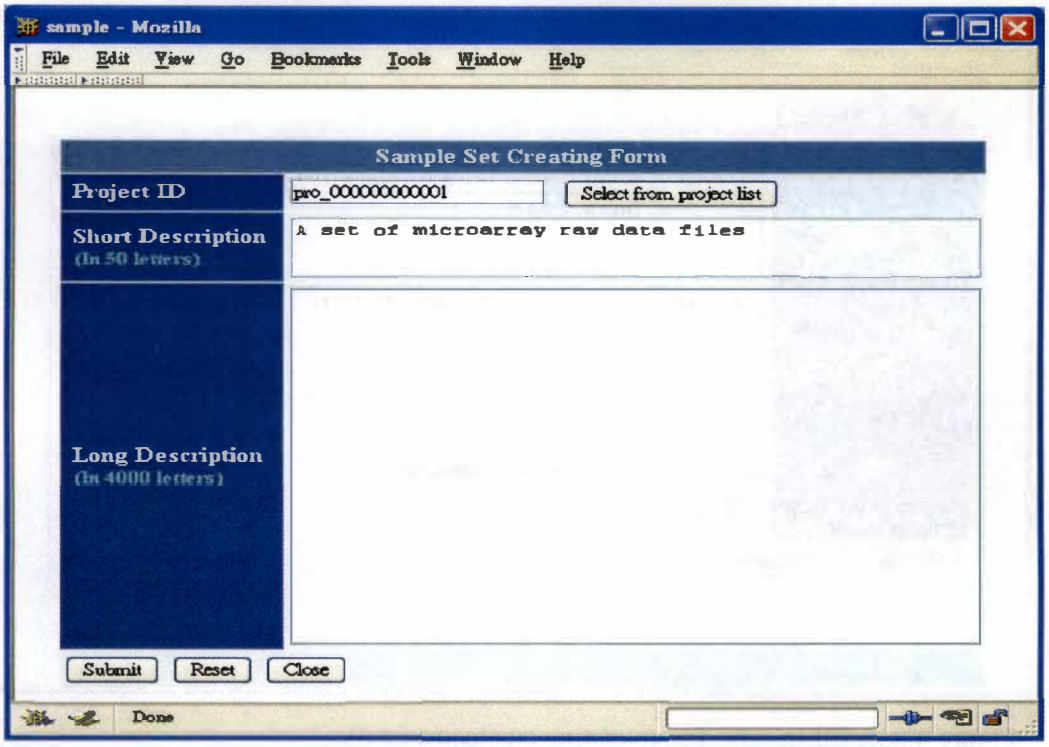


Figure 4.6: A fill-in form for creating a sample set

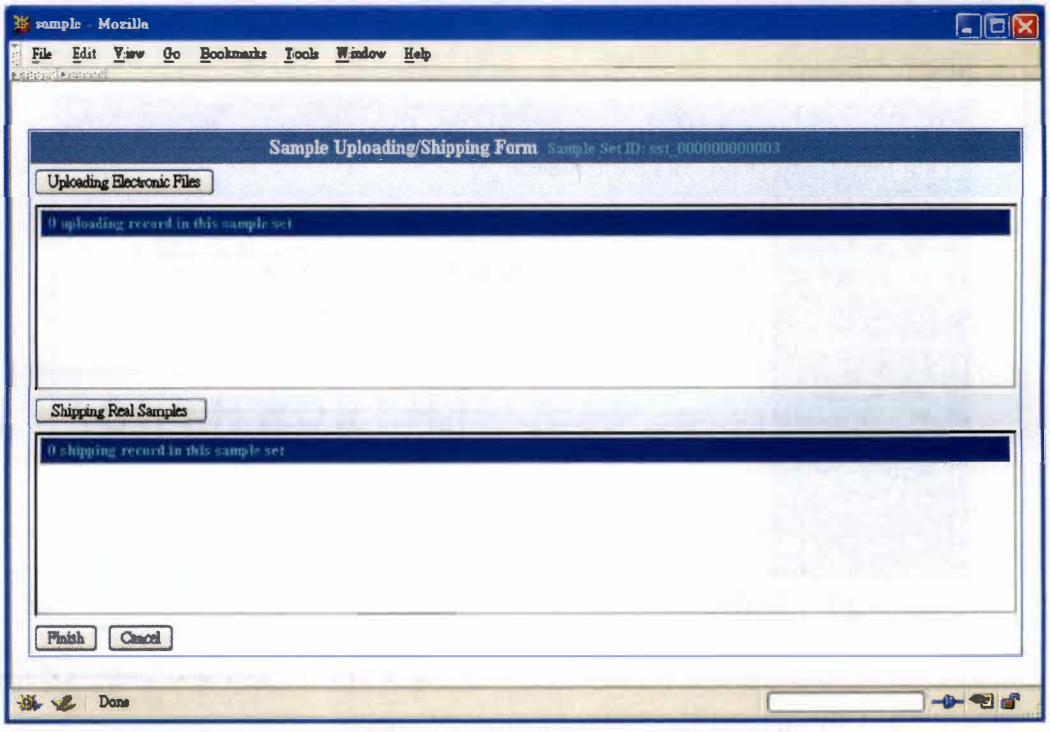


Figure 4.7: A function for adding sample items into a sample set

sample_uploading - Mozilla

Close Browser

File Uploading Form

Short Description (In 50 letters)	QTL Raw Data File
Long Description (In 4000 letters)	The long description about this file goes here
Upload File	ktop\QTL_RawData.xls Browse...
Format Type	Miscellaneous

Upload Reset

Figure 4.8: A fill-in form for uploading an electronic file

sample_shipping - Mozilla

Close Browser

Sample Shipping Form

Ship To	INIA:Core:2-Informatics
Sample Type	DNA
Ship Date & Time (mm/dd/yyyy)	10/27/2003 Show Calendar 10:00 AM
Short Description (In 50 letters)	A DNA Sample
Long Description (In 4000 letters)	The long description about the sample goes here

Shipping Reset

Figure 4.9: A fill-in form for shipping a physical sample

the process status from "Waiting Approval" to "Approval" (see figure 4.10). Core labs can also communicate with research labs via RESPONSE functionality (see figure 4.11). For example, if a request cannot be processed, the core lab can respond with the reason. The INIACS system will also assign an identifier to response record, which is called response ID, e.g. "rsp_000000000001" is a response ID.

4.4.4 Experiment Scheduling

In this step, core labs are able to schedule experiments needed to process their requests (see figure 4.12). An experiment ID (e.g. exp_000000000001) will be assigned to each experiment record. At the time of writing, this function allows core labs to provide descriptions of planned experiments but does not provide any custom worksheet for particular types of experiments. However, in order to support the variety of experiments needed in different core labs, a future feature is expected to allow core labs to define their worksheets for particular types of experiments and to link to their protocols, which are stored in the system. This feature will be discussed in section 6.1.

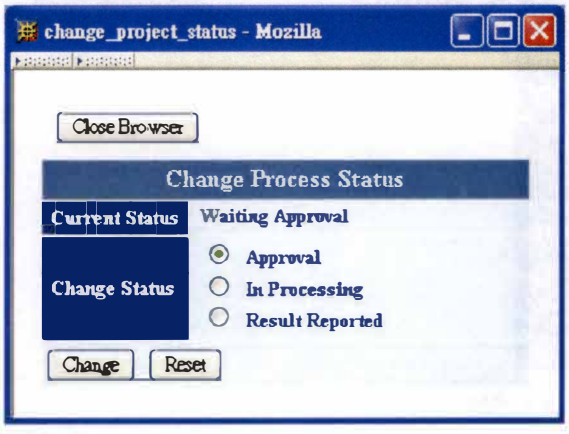


Figure 4.10: A function for changing the process status of a project

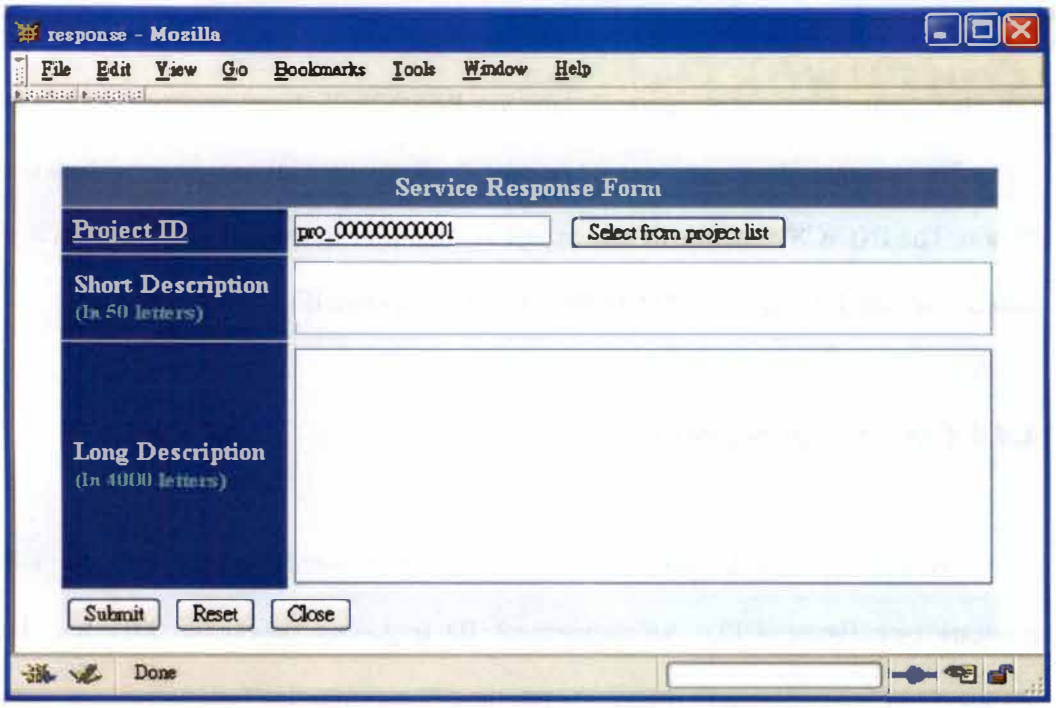


Figure 4.11: A fill-in form for responding to a request

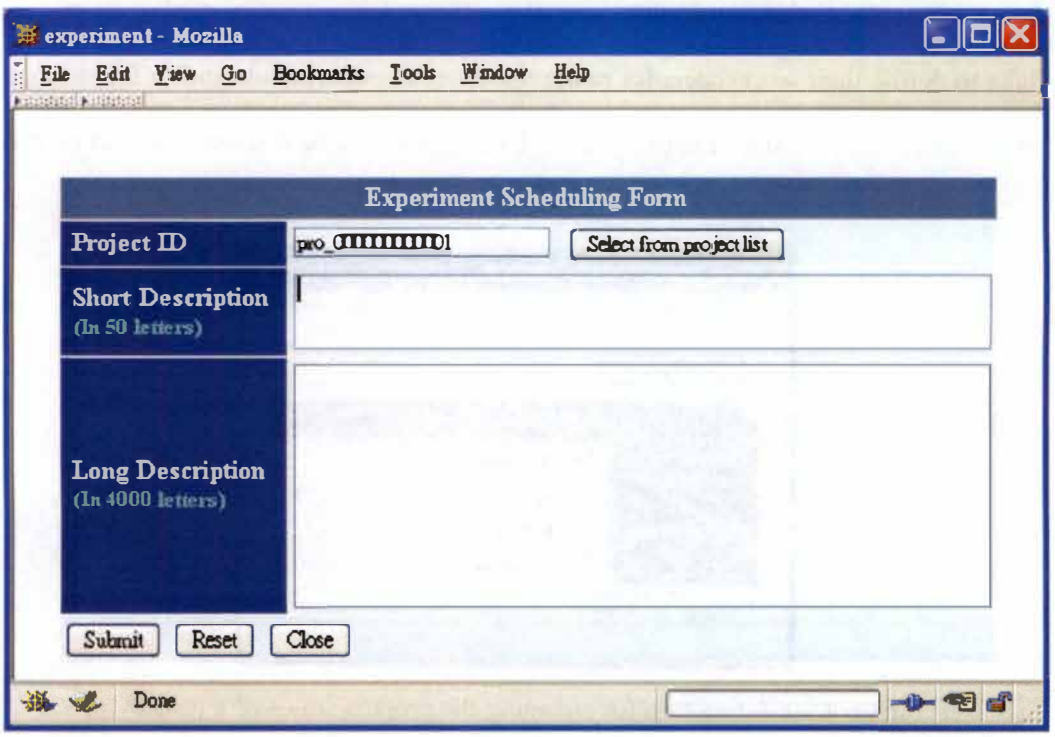


Figure 4.12: A fill-in form for scheduling an experiment

4.4.5 Result Report

After a core lab has completed its work, it will want to report results to the requesting lab. In this step, researchers need to create result sets for collecting all result data (see figure 4.13). A result set ID (e.g. rst_000000000001) will be assigned to each result set. During the process of data collection, a result set could be in "unfinished" status. If the result is ready for reporting, researchers have to change the status to "finished" (see figure 4.14). This will allow research labs to retrieve their results from the system. Figure 4.15 shows the function to allow core labs to collect multiple result data into a result set. To support various types of data formats that this consortium will produce, the INIACS allows core labs to upload result data in all types of formats including text files, images, Microsoft Excel spreadsheet files, XML format data, etc. (see figure 4.16). In addition, a URL link that refers to online resources can be added in a result set (see figure 4.17).

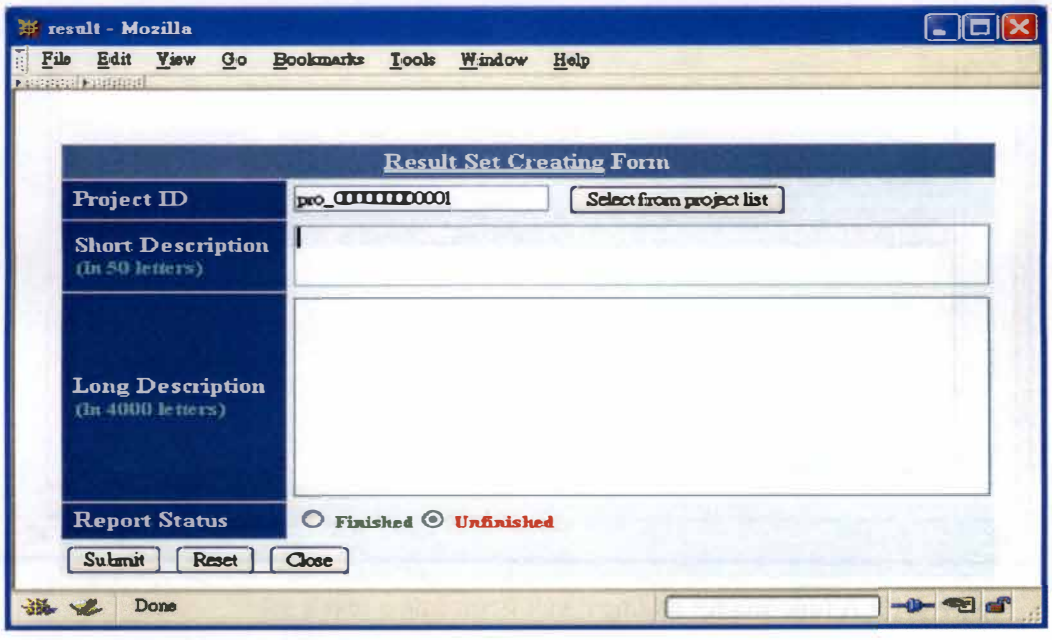


Figure 4.13: A fill-in form for creating a result set

Figure 4.14: A function for changing the status of a result set

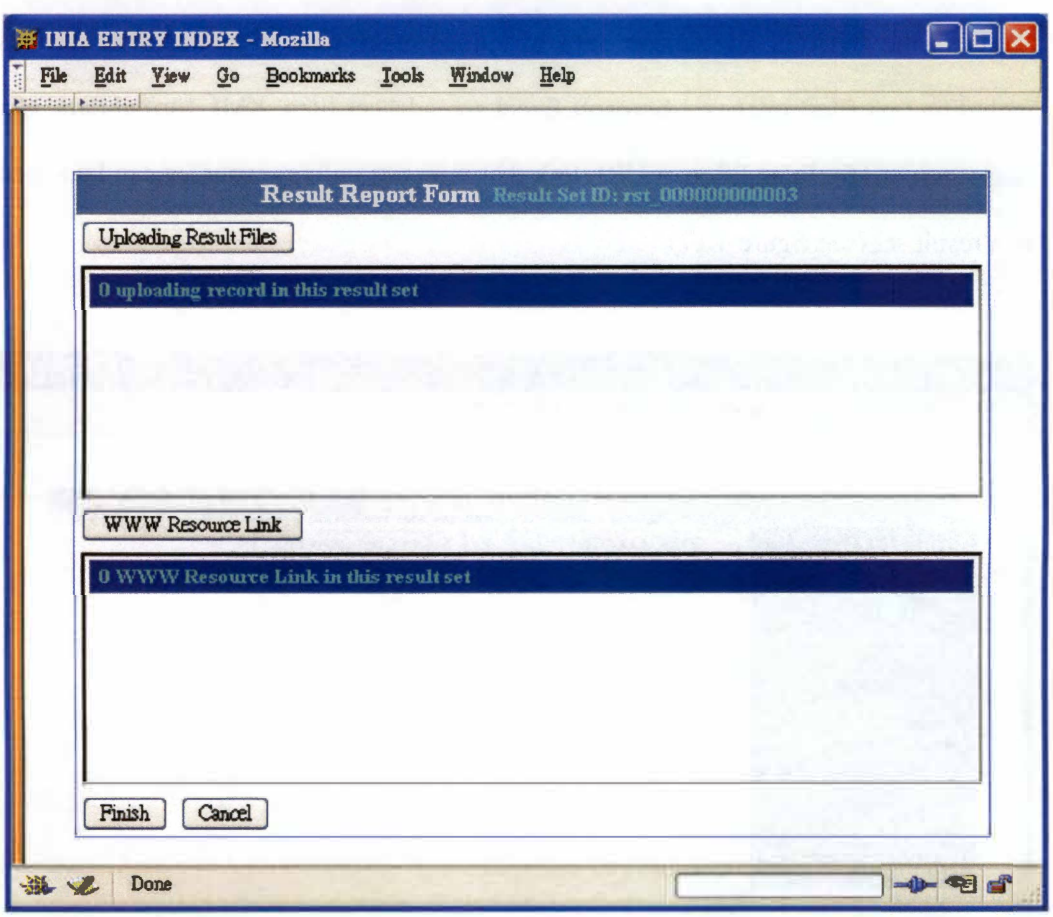


Figure 4.15: A function for adding result items into a result set

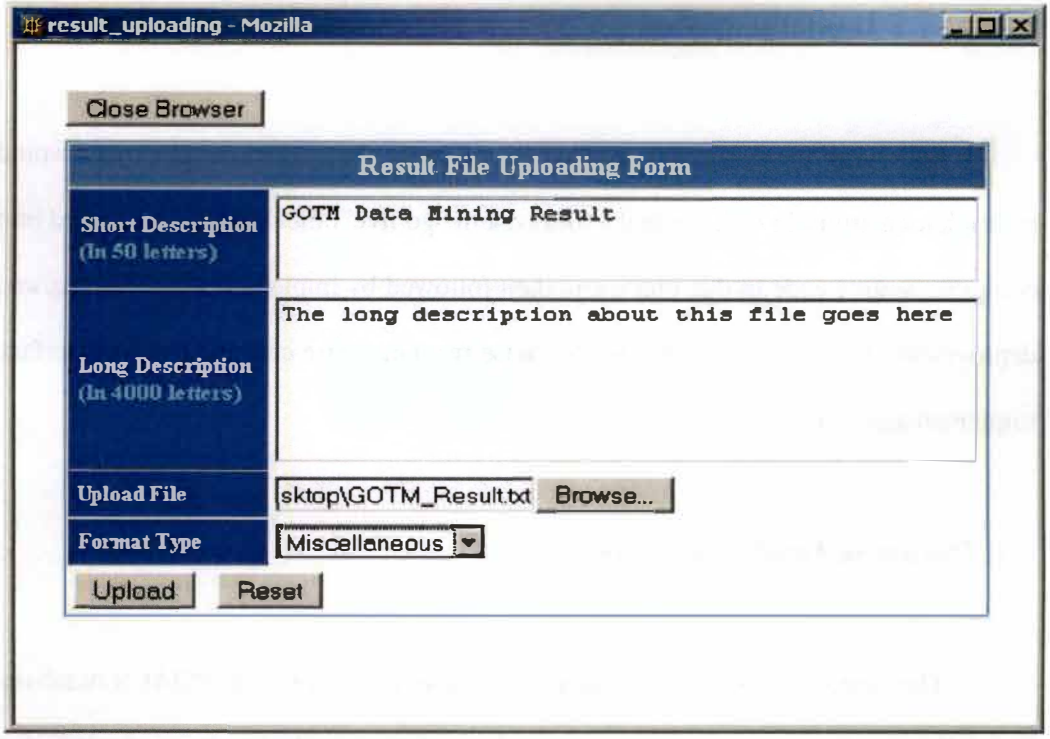


Figure 4.16: A fill-in form for uploading a result data file

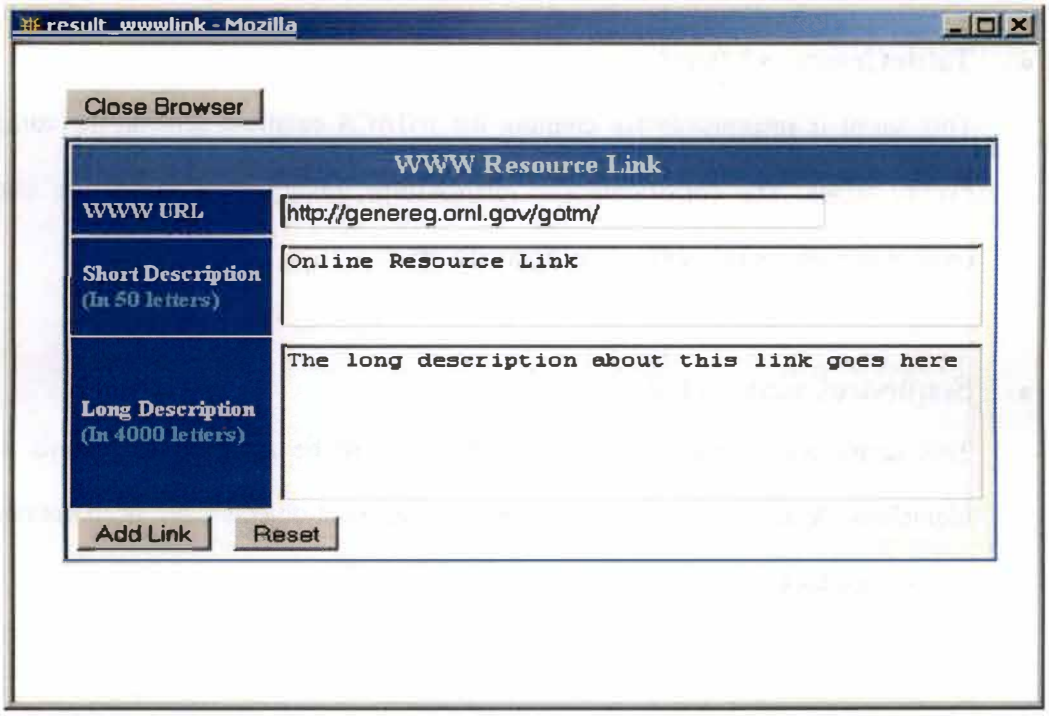


Figure 4.17: A fill-in form for adding a link referring to online resources

Chapter 5 Implementation

Following the design phase, the next phase of the system development would be implementation. In order to make this system "go live", the design is translated into computer source code in this phase and then followed by implementation, testing, and deployment. This chapter describes this phase from database creation to user interface implementation and testing.

5.1 Database Implementation

The first stage of the implementation phase is to create the INIACS database. The following SQL scripts are implemented based on the schema design described in chapter three.

- **TableCreate_v1.0.sql**

  This script is responsible for creating the INIACS database schema including twenty tables. The constraints and relationships among these tables are also created in running this script (see Appendix B).

- **SequenceCreate_v1.0.sql**

  This script creates the sequence objects that will be assigned to records as identifiers. At the time of writing, eighteen sequence objects have been created (see Appendix C).

- **FunctionCreate_v1.0.sql**

  This script written in the PL/SQL language creates database functions. Currently, one function is created in the database (see Appendix D).

- **ViewCreate_v1.0.sql**

  This script creates 53 relational views for the purpose of data access mentioned in section 3.6.2. In order to enhance data security mentioned in section 3.6.4, different views will be granted to different roles with certain privileges (see Appendix E).

- **TriggerCreate_v1.0.sql**

  This script creates 50 triggers including triggers for assigning sequence to records as identifiers, triggers for recording the log data when certain tables are modified, and triggers for inserting records into base tables instead of the views (instead of triggers). One trigger, called AIUDFER_REQUESTS_TRIGGER, is also responsible for creating a project when a request is made (see Appendix F).

- **RoleCreate_v1.0.sql**

  This script creates four database roles as follows: INIA_GUEST_ROLE, INIA_CLIENT_ROLE, INIA_SERVICE_ROLE and INIA_ADMIN_ROLE.

- **GrantPrivilegeToRole_v1.0.sql**

  This script is responsible for granting certain privileges (e.g. SELECT, INSERT or UPDATE) on views (created by ViewCreate_v1.0.sql script) and sequence objects (created by SequenceCreate_v1.0.sql script) to roles. To secure the

database, database administrators (DBAs) need to grant roles to database users depending on what role the user has (refer to section 3.6.4).

5.2 User Interface Implementation

The second stage of the implementation is to develop the user interface, using the PHP language. The work undertaken in this stage is basically nothing more than coding the idea of the workflow schema, defined in chapter four, into web applications. Furthermore, the interface also performs the functions to connect database server, to translate the request into SQL Data Manipulation Language (DML) and to pass the query. At the time of writing, the interface application consists of about 100 PHP script files, which have been installed in an Apache web server. These scripts can be grouped into the following ten categories according to their functionality.

- **Login**

  The INIACS must obtain a user's identifier from the BioTeams' database before he can be logged in to the INIACS. Therefore, the user identifier must be entered into INIACS from the BioTeams' application. After obtaining the user identifier, the INIACS will communicate with the BioTeams' database to authenticate the user's privilege. Thus, INIACS users would login to BioTeams first and then login to the INIACS (see section 3.6.1 for a detailed explanation).

  Since each end user must have a login account for BioTeams, it would be confusing for him to remember the second username and password for the INIACS if he just wants to get into the INIACS. Besides, based on the system design schema, all members in an INIA project use the same username and password for the INIACS. This may cause serious security problems when some people are no

longer members of the project. To address this problem, it would be better to allow users to login to the INIACS automatically from the BioTeams so that users do not need to know the passwords. This would entail having the server side application go through the authentication process and then connect to the INIACS if the accession is appropriate. In such way, the database administrator could routinely change the passwords and the system can be implemented in a secure manner.

A PHP script called **iniacs_logon.php** has been created under the BioTeams' application directory to achieve this task. In this script, the application will generate a web page to list the INIA projects, which the user is participating (see figure 5.1). This script will retrieve the username and encrypted password, and then login to the INIACS by calling other INIACS application scripts: **logon.php** and **index.php**. These two scripts will perform the actions for login to the INIACS. Figure 5.1 shows the login has been approved by the server side application.

- **Menu Tree**

The menu tree structure is generated by the following scripts: **TreeMenu.php**, **TreeMenuXL.php**, **TreeMenu.js**, **ccBrowserInfo.php** and **TreeMenu.css**, which were developed in one open source project "HTML_TreeMenuXL" (32). As we mentioned in the section 4.4, the type of the menu appeared on the screen depends on the role of the login lab such as core lab or research lab. The script **menu.php** is created to generate the different types of menu dynamically according to role of the login lab (refer to section 4.3).

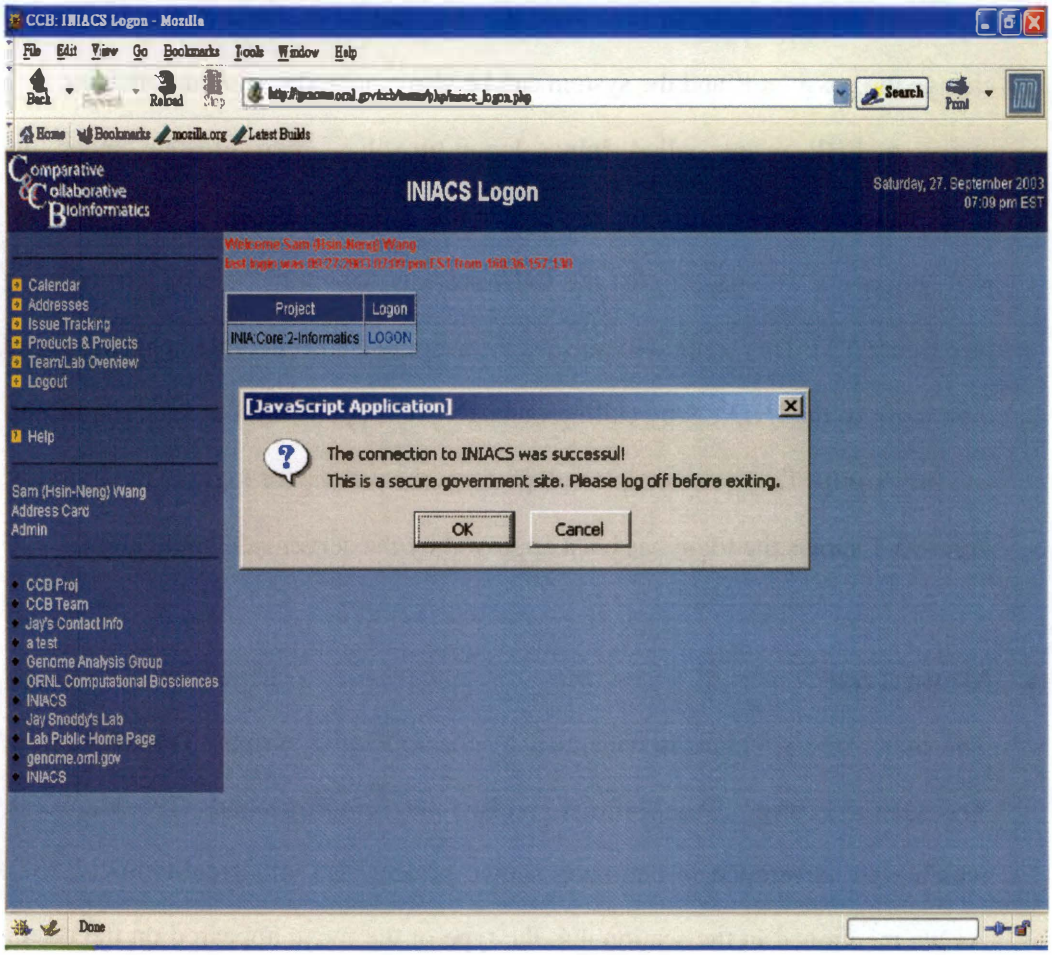


Figure 5.1: Login to INIACS system from the BioTeams application

- **Web Elements**

  In the script **common.inc**, we defined several common functions that are responsible for generating web pages with XHTML format, formatting the query result, and setting up global or local environment variables. For a better coding layout, the object-oriented feature of PHP language is used to define a Table object and a Form object in the scripts **TableClass.inc** and **FormClass.inc**. The methods defined in these two objects are used for the application programmer in generating web pages. The script **main.css** defines a cascading style sheet (CSS) for generating formatted web pages. The functions, objects and CSS defined in the above scripts are needed in other scripts, which are responsible for generating web pages.

- **Database Actions**

  The functions defined in the script **actions.inc** are used in logging in and logging off the system, querying the INIACS database by using PHP OCI-support functions, and storing query results into an array. These functions will be called by other scripts for the actions to communicate with the INIACS database.

- **Project**

  Scripts that are responsible for actions on requesting projects fall into this class. For example, the script **project.php** displays project information on a web page, and it is also responsible for the different actions on projects. For instance, to access **project_ins.php** if there is a request to create a new project. In project_ins.php a request is translated into a Data Definition Language (DDL) statement (see figure 5.2).

```
/* Parse parameters from $arr_request array */
$service=$arr_request['service'];
$lab=$arr_request['lab'];
$req_status=$arr_request['req_status'];
$req_status--;
$req_s_desc=$arr_request['req_s_desc'];
$req_l_desc=$arr_request['req_l_desc'];

/*************************************************
*                                               *
* SQL: Insert a new record into request_entry_view *
*                                               *
*************************************************/
$query="insert into $DataBase.request_entry_view
        (lab_id, service_id, service_lab_id,req_status, investigator, req_s_desc, req_l_desc)
        values ('$logon_labid', '$service','$lab','$req_status', $adr_id, '$req_s_desc', '$req_l_desc')";

$exe=affy_SQLExe($query);
```

Figure 5.2: The application code in project_ins.php is for translating a user's request into a DDL statement and executing the query command

The script **ProjectViewRecord.inc** generates a web page that displays project information in detail for research labs and also provides the complete set of functions needed to work on a project, i.e. request updating or sample shipping/uploading. Through this script researchers can also get responses and results back from core labs.

Most of the scripts in this class have the prefix 'project_' or 'Project' added to the filename, e.g. **project_select.php** generates a drop down box using JavaScript language, allowing a user to select a particular project (see figure 5.3).

- **Sample**

The scripts in this class provide a set of functions for research labs such as creating sample sets, shipping or uploading samples. In addition, core labs can check out sample information via these scripts.

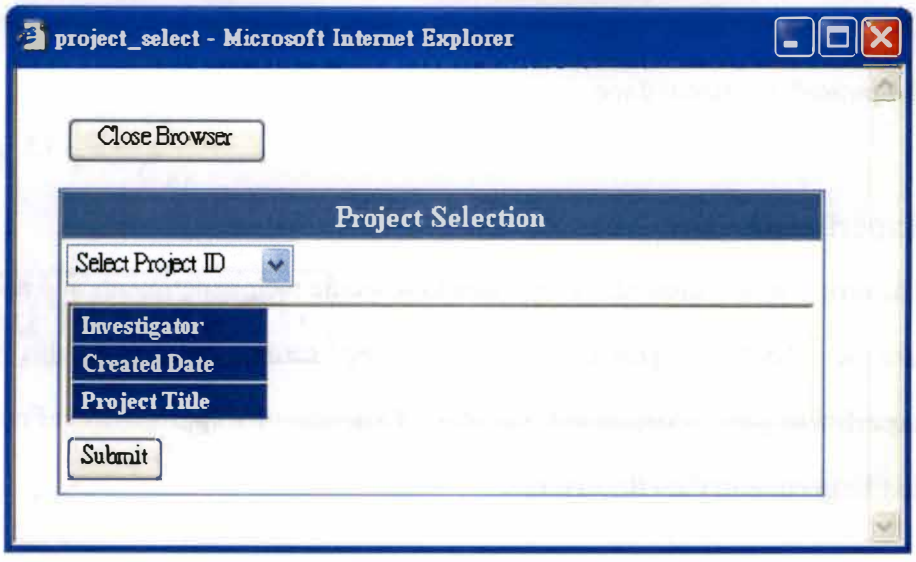


Figure 5.3: A function for selecting a particular project

The function of the script **sample.php** is similar to project.php. It generates a web page that displays sample set information and allows users to retrieve detailed information via hyperlinks.

There are other scripts in this class performing different functions. For example, **SamplePreForm.inc** and **sample_set_ins.php** are responsible for creating sample sets. **UploadShipForm.inc**, **sample_uploading.php**, **sample_shipping.php**, **SampleLoadingList.inc** and **SampleShippingList.inc** are responsible for shipping samples or uploading data files. The following scripts are needed to track sample movements: **SampleTrack.inc**, **SampleAllTrack.inc**, **SampleSetTrack.inc**, **SampleFileTrack.inc** and **SampleShipTrack.inc**.

- **Response**

The scripts in this class allow core labs to make responses for requesting projects and allow research labs to view responses. The following scripts belong to this

class: **response.php**, **response_ins.php**, **Rsponse/ResponsePreForm.inc** and **Response/ViewRecord.inc**.

- **Experiment**

  The scripts in this class allow core labs to schedule their experiments and research labs to check on progress. The following scripts belong to this class: **experiment.php**, **experiment_ins.php**, **Experiment/ExperimentPreForm.inc** and **Experiment/ViewRecord.inc**.

- **Result**

  The scripts in this class allow core labs to create result sets, to upload result files, to provide relevant web links and to report results. Once results have been reported, research labs can get the results back via these scripts.

- **Public**

  The idea of data sharing among the INIA consortium or with the people around the world (refer to section 2.3.7) is implemented by the scripts in this class. The script **DomainSelect.inc** allows users to select one public domain to enter. The scripts **INIAEastDomain.inc**, **INIADomain.inc** and **WorldDomain.inc** allow users to acquire shared data or projects. The script **PubViwRecord.inc** presents projects in detail, but at the time of writing it does not provide any functionality for users to work on them.

5.3 Current System Status

Currently, this system is tested on different web browsers including Internet Explorer, Mozilla and Netscape. In order to ensure that the web pages generated by the PHP scripts are displayed correctly on different browsers, these web pages are validated by using the W3C MarkUp Validating Service. In addition, the cascading style sheet (CSS) is validated by using the W3C CSS Validating Service. Some modifications have been made on some PHP scripts in order to generate standard XHTML web pages.

To date the implementation of the system has made a good progress toward achieving its objective of supporting large-scale biological collaborations. After these tests, it will be ready to be deployed in the INIA consortium and to move forward to another cycle of system development. This new collaborative system will be demonstrated at the INIA-East 2003 Retreat meeting, in order to collect feedback from our users. One major task at this meeting is to get users involved in the system deployment phase, having them test and evaluate the system. It is expected to ascertain more additional requirements of users from their feedback and evaluation. Some of these requirements might be core specific or research lab specific needs. To meet these additional requirements is the direction of future work and will be discussed in chapter 6.

Chapter 6 Future Work

As mentioned before, the first version is a prototype, and the system is still in the implementation and testing phases at the time of writing. Although its major elements are in place, still much more functionality has to be developed in order to meet the full requirements of the INIA consortium. The remaining issues will be discussed in the following sections.

6.1 More Functionality Required

The following work is required to add more functionality either on the database layer or on the application layer.

- **Expanding the functionality of experiment scheduling**

  As mentioned in the section 2.3.4 and section 4.4.4, this functionality so far is nothing more than a fill-in form that allows users to describe the experiments needed to work on their samples. This functionality could assist core labs to ensure that no samples are overlooked and allow research labs to get more information on what work has been done on their samples. However, in order to meet the quality assurance (QA) requirement for samples, we need to expand this functionality by adding a feature of test method assignment (16). With this feature, core labs will be able to save test protocols in the system and then assign test methods to samples. In addition, default values of the parameters can be adjusted, when needed.

- **Annotation on shared data**

This system provides a gateway to allow people to share their data with others. Sometimes the results of an analysis are required to be explained or analyzed by experts at different locations. To promote such collaborations, the experts should be able to add their opinions and interpretations on shared data. This system will need to capture and present these annotations on the Web.

- **A better integration with BioTeams**

As mentioned in the section 2.4, the BioTeams system would be a good tool to help INIA researchers to manage human resources among this consortium. For example, people could be assigned to a particular INIA project so that the database could allow them to login to INIACS. We have already discussed this in chapter 5.

In addition to the login issue, there are many other resources that could be used to promote collaborations by integrating with BioTeams. For example, since a large number of INIA researchers are also associated with other projects, e.g. the Tennessee Mouse Genome Consortium (TMGC), some resources would be used among these projects, e.g. personal contact information. To avoid errors in duplicating these resources to different places, we are using the BioTeams' database as a central repository to manage these resources. For example, if a researcher wants to send an email to another one, we want the correct email address to be shared from the BioTeams' database. To achieve this, one server-side script is needed to query BioTeams' database and display the contact information on a web page. In such way the INIACS can promote better communication between research labs and core labs, as well as foster additional related projects.

6.2 XML/RDBM Overlap: Partial Decomposition Strategy

Nowadays, more and more data analysis applications are using XML (eXtensible Markup Language) to describe the structure and content of electronic documents (30). According to the list of XML resources given by Dr. Paul Gordon at University of Calgary, Canada, more than 50 XML formats have been defined for research in biological science (31). For example, the Bioinformatic Sequence Markup Language (BSML) describes biological sequence information including DNA, RNA, protein sequences and phenotypical characteristics in a semantic way (32). The MicroArray Gene Expression Markup Language (MAGE-ML), which was developed by MGED (the microarray gene expression data group, http://www.mged.org), is designed to describe information about microarray-based experiments (33). MAGE-ML has been proposed as a standard data-exchange format for communication of the Minimum Information About a Microarray Experiment (MIAME) supportive data between local databases, central archives and stand-alone analysis applications. More recently, MIAME has been recognized as a standard for recording and reporting microarray-based gene expression data (34) and some analysis packages have been designed to export result data files to MAGE-ML or MAGE-ML compliant XML documents, i.e. ArrayExpress from EBI (35), GDAC-Exporter SDK from Affymetrix (36) or BioArray Software Environment (BASE) system (37).

Data generated by INIA core labs can be stored in XML formats. For example, the Informatics Core can analyze microarray expression data by using the BASE application and outputting the results in MAGE-ML or other standard XML formats. As we can see, a lot of gene information will be stored in a huge XML file and some of them could be useful for further analysis, e.g. we can compare sets of genes

simultaneously in GeneKeyDB, which is a data mining environment developed in Dr. Jay Snoddy's lab, to elucidate the complex gene regulatory networks. To query GeneKeyDB, researchers need to search and extract a subset of data, i.e. the LocusLink IDs of genes, as an input from an entire XML file. This could be a very cumbersome task and lead to poor performance due to the huge size of the file. Also, because the XML format does not describe the relationships among the data, querying these XML files would be difficult.

To address this problem, a solution proposed here is to use a "partial decomposition" strategy to partially extract a subset of data (i.e. LocusLink IDs in our case) into RDBMS for quick access and manipulation by sets, so that the function of using XML (structuring content) can still be kept and also of using SQL (enforcing relational constraints and organizing data) (38). Basically, the components of partial decomposition are the following:

- Keeping the full data in XML format
- Storing the XML file as an instance in the database
- Extracting a needed subset of data from XML into relational tables

In fact, the INIACS allows users to keep their files in any kind of format and store them in the database as BFILE instances. For a standard XML format, such as MAGE-ML format, the system will reference the XML DTD or schema for each data set via the FORMATS table (refer to section 3.4 for the definition of FORMATS table). In order to implement this partial decomposition strategy, the third component is needed in the future version of the system. For the case mentioned above, a script is needed to extract the LocusLink IDs from XML files into the GKDB_LINKS table

that already exists in the INIACS database (refer to section 3.4). With this component, to retrieve a set of genes is just a SQL 'select' query on the GKDB_LINKS table and one no longer needs to parse huge XML files.

6.3 Integration with the MicroArray Database

Microarray expression data is one major resource this consortium will produce, particularly in the Bioanalytical Core. This core receives samples and processes microarray experiments in a high-throughput manner. Although the current INIACS can assist this core to acquire samples from research labs, however, more information about requests or samples is needed to process analysis, e.g. the array (chip) information or the treatment on samples. This core-specific need is not supported in the current INIACS. Moreover, once the experiment is completed, the current INIACS does not enforce semantic structuring of the microarray expression data, which is also needed to help this core on data mining.

To meet these additional requirements, the INIACS will be integrated with the MicroArray Database (MAD), another system that is being developed in our lab. The MAD is developed specifically to assist the Bioanalytical Core to handle the microarray experimental data including sample information and result data. It is designed as a MIAME-compliant database system that could export the result data to MAGE-ML format files. The integration of the INIACS and MAD (INIACS-MAD) is expected to provide a robust data exchange and integration on those microarray expression data for the Bioanalytical Core.

Chapter 7 Conclusion

The primary goal of this project is to provide the INIA consortium with an IT infrastructure to support large-scale biological collaborations and to give this consortium a broader access to distributed resources. This infrastructure will allow people to access, utilize and share the distributed resources found in a number of laboratories, including specialized expertise and equipment. Moreover, because many different roles are involved in such large scale collaborations, developers need to ensure data can be accessed or stored only by authorized users; the right to "read" data or "write" data needs to be restricted depending on the user's job. The INIACS system was developed and accomplished the overall objectives.

Many issues needed to be addressed in building this collaborative system. This thesis has discussed the needs of the INIA consortium and the current modes of collaboration. Accessing distributed resources requires a central data repository or database. This database has been built in a generic schema that allows easy maintenance and modification.

Relational views are created not only to allow users to access the resources stored in the database but also to prevent unauthorized data accessing, enhancing data security. To enforce system security, database roles are used to restrict users' privileges. Users can be granted with certain object privileges (e.g. to select or insert on relational views), but not system privileges. Also, numbers of triggers on relational views have been defined to check users' "write" privileges when users try to store data in the database.

In order to support the geographically distributed collaborations, a web-based user interface has been built and implemented to allow users to work together and

access distributed resources regardless of their physical locations. The dynamic web pages are generated in standard XHTML format by PHP scripts so that the requested information can be displayed correctly on most web browsers. Many features have been added to support the workflow that occurs in this large-scale collaboration. For example, requesting services, loading samples or reporting analytical results. In addition, this system allows researchers to track their sample movements and to share their collaborative data with others through its data sharing function. Integration with the BioTeams system allows designers to focus on issues of collaborative data management and leave the people management to the BioTeams system.

Nowadays, with advancements in experimental and information techniques, biological research is moving towards systematic approaches for understanding complex biological networks. INIA consists of a team working across geographic boundaries to meet the challenge of understanding such networks. A prerequisite for its success is a collaborative bioinformatics that allows researchers access the geographically distributed resources of INIA. The INIACS has been developed to meet this requirement and is now ready for deployment. Although future work will be needed in the next development cycle, it is foreseen that, once deployed, it will provide a robust access to INIA's distributed resources including expertise, people, equipment and analytical processes residing in cores regardless of their physical locations.

List of References

1. Ideker, T., Galitski, T., and Hood L. (2001) A new approach to decoding life: systems biology, Annual Review of Genomics and Human Genetics 2:343
2. Jakobovits R., Soderland SG., Taira RK., Brinkley JF. (2000) Requirements of a Web-Based Experiment Management System, Proc AMIA Symp. 2000:374
3. Kanehisa M. and Bork P. (2003) Bioinformatics in the post-sequence era, Nature Genetics 33 Suppl:305
4. Integrative Neuroscience Initiative on Alcoholism, http://www.iniastress.org/
5. INIA-West consortium, http://www.scripps.edu/np/inia/index.html
6. Malcolm RJ. (2003) GABA Systems, Benzodiazepines, and Substance Dependence, J Clin Psychiatry 64(suppl 3):36
7. Sinnett D, Wagstaff J, Glatt K, Woolf E, Kirkness EJ, Lalande M. (1993) High-resolution mapping of the gamma-aminobutyric acid receptor subunit beta 3 and alpha 5 gene cluster on chromosome 15q11-q13, and localization of breakpoints in two Angelman syndrome patients, Am J Hum Genet. 52(6):1216
8. Glatt K, Glatt H, Lalande M. (1997) Structure and organization of GABRB3 and GABRA5, Genomics. 41(1):63
9. Song J, Koller DL, Foroud T, Carr K, Zhao J, Rice J, Nurnberger JI Jr, Begleiter H, Porjesz B, Smith TL, Schuckit MA, Edenberg HJ. (2003) Association of $GABA_A$ receptors and alcohol dependence and the effects of genetic imprinting, Am J Med Genet. 117B(1):39
10. Brady KT., Sonne SC. (1999) The role of stress in alcohol use, alcoholism treatment, and relapse, Alcohol Research and Health 23(4):263

11. Vale W, Spiess J, Rivier C, Rivier J. (1981) Characterization of a 41-residue ovine hypothalamic peptide that stimulates secretion of corticotropin and beta-endorphin, Science 213:1394

12. Sillaber I, Rammes G, Zimmermann S, Mahal B, Zieglgansberger W, Wurst W, Holsboer F, Spanagel R. (2002) Enhanced and delayed stress-induced alcohol drinking in mice lacking functional CRH1 receptors, Science 296(5569):931

13. Book SW., Randall CL. (2002) Social anxiety disorder and alcohol use, Alcohol Research and Health 26(2):130

14. Date C.J. (2000) An Introduction to Database Systems, seventh edition, Addison Wesley, ISBN 0-201-38590-2

15. Rob P. and Coronel C. (2000) Database Systems: Design, Implementation, and Management, forth edition, Course Technology, ISBN 0-7600-1090-0

16. Hinton MD. (1995) Laboratory Information Management Systems, Marcel Dekker, ISBN 0-8247-9458-3

17. Gibbon G. (1996) A brief history of LIMS, Laboratory Automation and Information Management 32:1

18. Gillespie H. (1997) New technology impacts and drives LIMS evolution, http://www.limsource.com/library/limszine/applica_tions/apgen97.html

19. Paszko C., Turner E. (2002) Laboratory Information Management Systems, second edition, Marcel Dekker, ISBN 0-8247-0521-1

20. Bentley D. (1999) Analysis of a Laboratory Information Management System (LIMS), http://www.umsl.edu/~sauter/analysis/LIMS_example.html

21. SourceForge.net, http://sourceforge.net/

22. TUTOS, http://www.tutos.org/homepage/index.html

23. Codd EF. (1970) A Relational Model of Data for Large Shared Data Banks, Communications of the ACM 13(6):377

24. Weeldreyer JA., Friesen OD. (1978) Multics Relational Data Store: An Implementation of A Relational Data Base Manager, Proceedings of the Eleventh Hawaii International Conference on Systems Sciences 1:52

25. Elmasri R., Navathe SB. (2000) Fundamentals of Database Systems, third edition, Addison Wesley, ISBN 0-8053-1755-4

26. Chen P. (1976) The Entity-Relationship Model: Toward a Unified View of Data, ACM Transactions on Database Systems 1:1

27. Dillon S., Beck C., Kyte T. (2002) Beginning Oracle Programming, Wrox Press Ltd, ISBN 1-8610-0690-X

28. Castagnetto J., Rawat H., Schumann S., Scollo C., Veliath D. (1999) Professional PHP Programming, Wrox Press Ltd, ISBN 1-8610-0296-3

29. Heyes R. (2002) HTML_TreeMenuXL open source project, http://www.chipchapin.com/WebTools/MenuTools/HTML_TreeMenuXL/

30. Achard F, Vaysseix G, Barillot E. (2001) XML, bioinformatics and data integration, Bioinformatics 17(2):115

31. Gordon P, XML for Molecular Biology http://www.visualgenomics.ca/gordonp/xml/#community

32. Bioinformatic Sequence Markup Language (BSML) http://www.bsml.org

33. Spellman PT, Miller M, Stewart J, Troup C, Sarkans U, Chervitz S, Bernhart D, Sherlock G, Ball C, Lepage M, Swiatek M, Marks WL, Goncalves J, Markel S, Iordan D, Shojatalab M, Pizarro A, White J, Hubley R, Deutsch E, Senger M, Aronow BJ, Robinson A, Bassett D, Stoeckert CJ Jr, Brazma A. (2002) Design and implementation of microarray gene expression markup language (MAGE-ML), Genome Biology 3(9):research0046

34. Brazma A, Hingamp P, Quackenbush J, Sherlock G, Spellman P, Stoeckert C, Aach J, Ansorge W, Ball CA, Causton HC, Gaasterland T, Glenisson P, Holstege FC, Kim IF, Markowitz V, Matese JC, Parkinson H, Robinson A, Sarkans U, Schulze-Kremer S, Stewart J, Taylor R, Vilo J, Vingron M. (2001) Minimum information about a microarray experiment (MIAME) – toward standards for microarray data, Nature Genetics 29:365

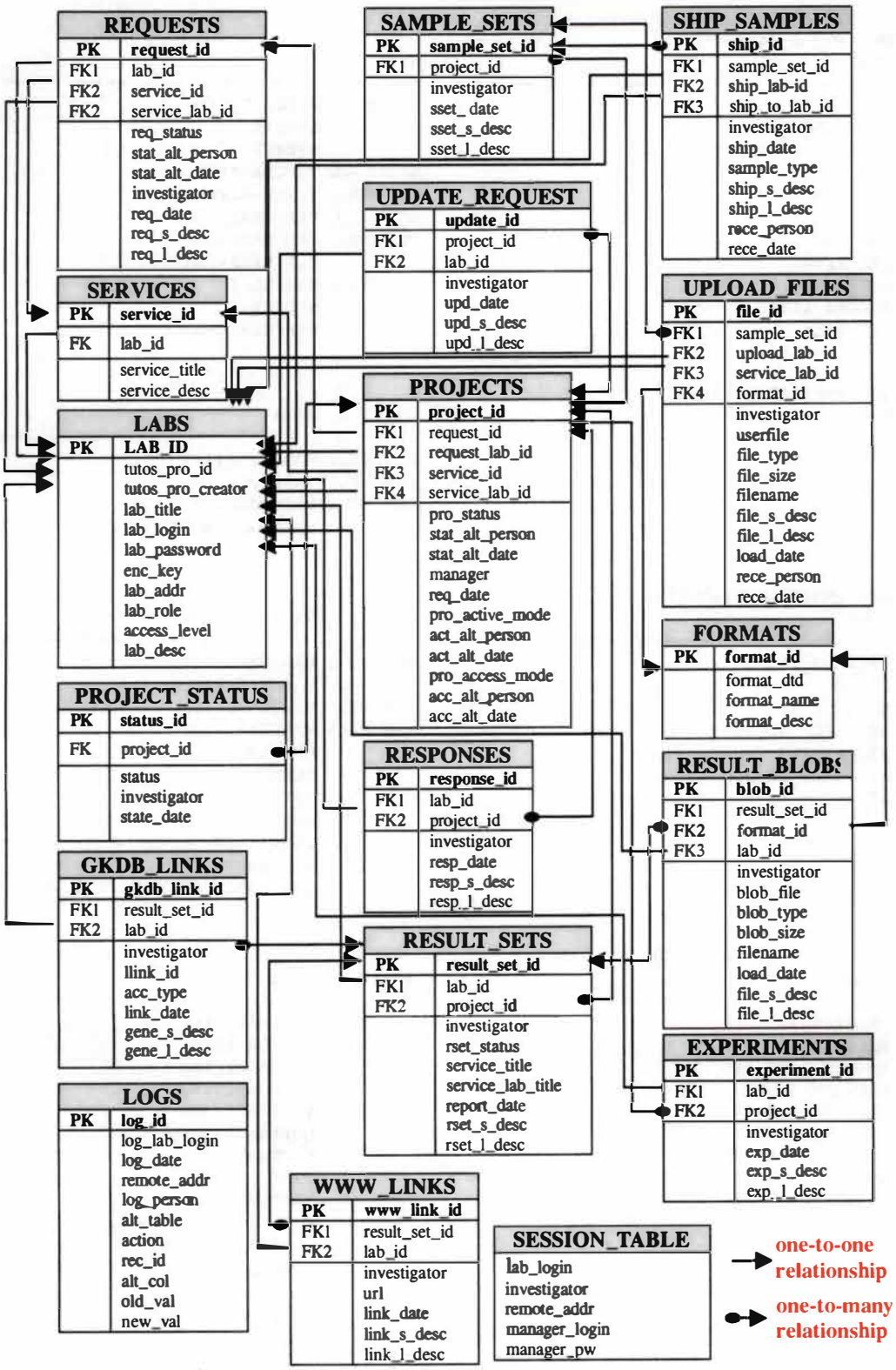
35. ArrayExpress, EBI http://www.ebi.ac.uk/arrayexpress

36. GeneChip® Data Access Components (GDAC) Exporter SDK, Affymetrix, Inc. http://www.affymetrix.com/support/developer/exporter/GDACExporter/Pages/GDACExporter_home.affx

37. Saal LH, Troein C, Vallon-Christersson J, Gruvberger S, Borg A, Peterson C. (2002) BioArray Software Environment (BASE): a platform for comprehensive management and analysis of microarray data, Genome Biology 3(8):software0003

38. Appelquist DK. (2001) XML and SQL Developing Web Applications, Addison Wesley, ISBN 0-201-65796-1

Appendix A: INIACS Database Schema



Appendix B: Table Definition

```
SQL> Describe LABS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAB_ID                                    NOT NULL VARCHAR2(16)
 TUTOS_PRO_ID                                       NUMBER
 TUTOS_PRO_CREATOR                                  NUMBER
 LAB_TITLE                                 NOT NULL VARCHAR2(50)
 LAB_LOGIN                                 NOT NULL VARCHAR2(200)
 LAB_PASSWORD                              NOT NULL VARCHAR2(200)
 ENC_KEY                                   NOT NULL VARCHAR2(200)
 LAB_ADDR                                           VARCHAR2(250)
 LAB_ROLE                                  NOT NULL NUMBER
 ACCESS_LEVEL                                       NUMBER
 LAB_DESC                                           VARCHAR2(4000)

SQL> Describe SERVICES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SERVICE_ID                                NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 SERVICE_TITLE                             NOT NULL VARCHAR2(50)
 SERVICE_DESC                                       VARCHAR2(4000)

SQL> Describe REQUESTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 REQUEST_ID                                NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 SERVICE_ID                                NOT NULL VARCHAR2(16)
 SERVICE_LAB_ID                            NOT NULL VARCHAR2(16)
 REQ_STATUS                                         NUMBER
 STAT_ALT_PERSON                                    NUMBER
 STAT_ALT_DATE                                      DATE
 INVESTIGATOR                                       NUMBER
 REQ_DATE                                           DATE
 REQ_S_DESC                                         VARCHAR2(50)
 REQ_L_DESC                                         VARCHAR2(4000)

SQL> Describe PROJECTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 REQUEST_ID                                NOT NULL VARCHAR2(16)
 REQUEST_LAB_ID                            NOT NULL VARCHAR2(16)
 SERVICE_ID                                NOT NULL VARCHAR2(16)
 SERVICE_LAB_ID                            NOT NULL VARCHAR2(16)
 PRO_STATUS                                         NUMBER
 STAT_ALT_PERSON                                    NUMBER
 STAT_ALT_DATE                                      DATE
 MANAGER                                            NUMBER
 REQ_DATE                                           DATE
 PRO_ACTIVE_MODE                                    NUMBER
 ACT_ALT_PERSON                                     NUMBER
 ACT_ALT_DATE                                       DATE
 PRO_ACCESS_MODE                                    NUMBER
 ACC_ALT_PERSON                                     NUMBER
 ACC_ALT_DATE                                       DATE
```

```
SQL> Describe PROJECT_STATUS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STATUS_ID                                 NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 STATUS                                    NOT NULL NUMBER
 INVESTIGATOR                                       NUMBER
 STATE_DATE                                         DATE

SQL> Describe UPDATE_REQUEST
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 UPDATE_ID                                 NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 UPD_DATE                                           DATE
 UPD_S_DESC                                         VARCHAR2(50)
 UPD_L_DESC                                         VARCHAR2(4000)

SQL> Describe SAMPLE_SETS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SAMPLE_SET_ID                             NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 SSET_DATE                                          DATE
 SSET_S_DESC                                        VARCHAR2(50)
 SSET_L_DESC                                        VARCHAR2(4000)

SQL> Describe FORMATS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FORMAT_ID                                 NOT NULL VARCHAR2(16)
 FORMAT_DTD                                         VARCHAR2(250)
 FORMAT_NAME                                        VARCHAR2(200)
 FORMAT_DESC                                        VARCHAR2(4000)

SQL> Describe UPLOAD_FILES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 FILE_ID                                   NOT NULL VARCHAR2(16)
 SAMPLE_SET_ID                             NOT NULL VARCHAR2(16)
 UPLOAD_LAB_ID                             NOT NULL VARCHAR2(16)
 SERVICE_LAB_ID                            NOT NULL VARCHAR2(16)
 FORMAT_ID                                          VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 USERFILE                                           BINARY FILE LOB
 FILE_TYPE                                          VARCHAR2(100)
 FILE_SIZE                                          NUMBER
 FILENAME                                           VARCHAR2(200)
 FILE_S_DESC                                        VARCHAR2(50)
 FILE_L_DESC                                        VARCHAR2(4000)
 LOAD_DATE                                          DATE
 RECE_PERSON                                        NUMBER
 RECE_DATE                                          DATE
```

```
SQL> Describe SHIP_SAMPLES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SHIP_ID                                   NOT NULL VARCHAR2(16)
 SAMPLE_SET_ID                             NOT NULL VARCHAR2(16)
 SHIP_LAB_ID                               NOT NULL VARCHAR2(16)
 SHIP_TO_LAB_ID                            NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 SHIP_DATE                                          DATE
 SAMPLE_TYPE                               NOT NULL VARCHAR2(100)
 SHIP_S_DESC                                        VARCHAR2(50)
 SHIP_L_DESC                                        VARCHAR2(4000)
 RECE_PERSON                                        NUMBER
 RECE_DATE                                          DATE

SQL> Describe RESPONSES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESPONSE_ID                               NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 RESP_DATE                                          DATE
 RESP_S_DESC                                        VARCHAR2(50)
 RESP_L_DESC                                        VARCHAR2(4000)

SQL> Describe EXPERIMENTS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 EXPERIMENT_ID                             NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 EXP_DATE                                           DATE
 EXP_S_DESC                                         VARCHAR2(50)
 EXP_L_DESC                                         VARCHAR2(4000)

SQL> Describe RESULT_SETS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 RESULT_SET_ID                             NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 PROJECT_ID                                NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 RSET_STATUS                                        NUMBER
 SERVICE_TITLE                             NOT NULL VARCHAR2(50)
 SERVICE_LAB_TITLE                         NOT NULL VARCHAR2(50)
 REPORT_DATE                                        DATE
 RSET_S_DESC                                        VARCHAR2(50)
 RSET_L_DESC                                        VARCHAR2(4000)
```

```
SQL> Describe RESULT_BLOBS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 BLOB_ID                                   NOT NULL VARCHAR2(16)
 RESULT_SET_ID                             NOT NULL VARCHAR2(16)
 FORMAT_ID                                          VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 BLOB_FILE                                          BINARY FILE LOB
 BLOB_TYPE                                          VARCHAR2(100)
 BLOB_SIZE                                          NUMBER
 FILENAME                                           VARCHAR2(200)
 LOAD_DATE                                          DATE
 BLOB_S_DESC                                        VARCHAR2(50)
 BLOB_L_DESC                                        VARCHAR2(4000)

SQL> Describe GKDB_LINKS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 GKDB_LINK_ID                              NOT NULL VARCHAR2(16)
 RESULT_SET_ID                             NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 LLINK_ID                                           NUMBER
 ACC_TYPE                                           VARCHAR2(100)
 LINK_DATE                                          DATE
 GENE_S_DESC                                        VARCHAR2(50)
 GENE_L_DESC                                        VARCHAR2(4000)

SQL> Describe WWW_LINKS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 WWW_LINK_ID                               NOT NULL VARCHAR2(16)
 RESULT_SET_ID                             NOT NULL VARCHAR2(16)
 LAB_ID                                    NOT NULL VARCHAR2(16)
 INVESTIGATOR                                       NUMBER
 URL                                                VARCHAR2(250)
 LINK_DATE                                          DATE
 LINK_S_DESC                                        VARCHAR2(50)
 LINK_L_DESC                                        VARCHAR2(4000)

SQL> Describe LOGS
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LOG_ID                                    NOT NULL VARCHAR2(16)
 LOG_LAB_LOGIN                             NOT NULL VARCHAR2(20)
 LOG_DATE                                           DATE
 REMOTE_ADDR                                        VARCHAR2(20)
 LOG_PERSON                                         NUMBER
 ALT_TABLE                                 NOT NULL VARCHAR2(32)
 ACTION                                    NOT NULL VARCHAR2(8)
 REC_ID                                    NOT NULL VARCHAR2(16)
 ALT_COL                                            VARCHAR2(50)
 OLD_VAL                                            VARCHAR2(4000)
 NEW_VAL                                            VARCHAR2(4000)

SQL> Describe SESSION_TABLE
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 LAB_LOGIN                                 NOT NULL VARCHAR2(20)
 INVESTIGATOR                                       NUMBER
 REMOTE_ADDR                                        VARCHAR2(20)
 MANAGER_LOGIN                                      VARCHAR2(200)
 MANAGER_PW                                         VARCHAR2(20)
```

Appendix C: Sequence Objects

```
SQL> Describe USER_SEQUENCES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SEQUENCE_NAME                             NOT NULL VARCHAR2(30)
 MIN_VALUE                                          NUMBER
 MAX_VALUE                                          NUMBER
 INCREMENT_BY                              NOT NULL NUMBER
 CYCLE_FLAG                                         VARCHAR2(1)
 ORDER_FLAG                                         VARCHAR2(1)
 CACHE_SIZE                                NOT NULL NUMBER
 LAST_NUMBER                               NOT NULL NUMBER

SQL> SELECT SEQUENCE_NAME, MIN_VALUE, MAX_VALUE, INCREMENT_BY FROM USER_SEQUENCES
  2  /

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY
------------------------------ ---------- ---------- ------------
BOB_ID_SEQ                              1 1.0000E+12            1
BUG_ID_SEQ                              1 1.0000E+12            1
EXP_ID_SEQ                              1 1.0000E+12            1
FMT_ID_SEQ                              1 1.0000E+12            1
GKD_ID_SEQ                              1 1.0000E+12            1
LAB_ID_SEQ                              1 1.0000E+12            1
LOD_ID_SEQ                              1 1.0000E+12            1
LOG_ID_SEQ                              1 1.0000E+12            1
PRO_ID_SEQ                              1 1.0000E+12            1
PST_ID_SEQ                              1 1.0000E+12            1
REQ_ID_SEQ                              1 1.0000E+12            1

SEQUENCE_NAME                  MIN_VALUE  MAX_VALUE INCREMENT_BY
------------------------------ ---------- ---------- ------------
RSP_ID_SEQ                              1 1.0000E+12            1
RST_ID_SEQ                              1 1.0000E+12            1
SER_ID_SEQ                              1 1.0000E+12            1
SHP_ID_SEQ                              1 1.0000E+12            1
SST_ID_SEQ                              1 1.0000E+12            1
UPD_ID_SEQ                              1 1.0000E+12            1
WWW_ID_SEQ                              1 1.0000E+12            1

18 rows selected.
```

Appendix D: Function Definition

```
SQL> Describe USER_SOURCE
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 NAME                                               VARCHAR2(30)
 TYPE                                               VARCHAR2(12)
 LINE                                               NUMBER
 TEXT                                               VARCHAR2(4000)

SQL> SELECT LINE, TEXT FROM USER_SOURCE WHERE NAME='IS_MANAGER'
  2  /

      LINE TEXT
---------- --------------------------------------------------
         1 function is_manager(adrid number, l_login varchar2
           , l_pw varchar2) return number

         2 as
         3   rowcnt number;
         4 begin
         5   select count(*) into rowcnt
         6     from tutos.people
         7    where adr_id=adrid
         8      and login=l_login
         9      and pw=l_pw;
        10
        11   if rowcnt=1 then
        12      return 1;
        13   else
        14      return 0;
        15   end if;
        16 end is_manager;

16 rows selected.
```

Appendix E: Views

```
SQL> SELECT VIEW_NAME FROM USER_VIEWS
  2  /

VIEW_NAME
------------------------------
ADMINISTRATION_VIEW
BLOB_INSERT_VIEW
BLOB_SELECT_VIEW
BUG_REPORT_VIEW
BUG_SELECT_VIEW
EXPERIMENT_SELECT_VIEW
FILE_LOADING_VIEW
FILE_RECEIVE_VIEW
FILE_SELECT_VIEW
FILE_TRACK_VIEW
FORMAT_SELECT_VIEW
GKDB_INSERT_VIEW
GKDB_SELECT_VIEW
INIA_LABS_VIEW
INVESTIGATOR_VIEW
LAB_INFO_VIEW
LAB_MEMBERS_VIEW

VIEW_NAME
------------------------------
PROJECT_MANAGER_VIEW
PROJECT_OWNERSHIP_VIEW
PUBLIC_BLOB_VIEW
PUBLIC_EXPERIMENT_VIEW
PUBLIC_FILE_VIEW
PUBLIC_GKDB_VIEW
PUBLIC_PROJECT_STATUS_VIEW
PUBLIC_PROJECT_VIEW
PUBLIC_RESPONSE_VIEW
PUBLIC_RESULT_VIEW
PUBLIC_SAMPLE_SET_VIEW
PUBLIC_SHIP_VIEW
PUBLIC_UPDATE_REQUEST_VIEW
PUBLIC_WWW_VIEW
REQUEST_ENTRY_VIEW
RESPONSE_SELECT_VIEW
RESULT_SELECT_VIEW
```

```
VIEW_NAME
------------------------------
SAMPLE_SETS_INSERT_VIEW
SAMPLE_SETS_SELECT_VIEW
SERVICE_EXPERIMENT_VIEW
SERVICE_LAB_VIEW
SERVICE_LIST_VIEW
SERVICE_RESPONSE_VIEW
SERVICE_RESULT_VIEW
SESSION_TABLE_VIEW
SHIP_RECEIVE_VIEW
SHIP_SAMPLES_VIEW
SHIP_SELECT_VIEW
SHIP_TRACK_VIEW
TRACK_PROJECT_STATUS_VIEW
UPDATE_PROJECT_ACTIVE_VIEW
UPDATE_PROJECT_STATUS_VIEW
UPDATE_REQUEST_INSERT_VIEW
UPDATE_REQUEST_SELECT_VIEW

VIEW_NAME
------------------------------
WWW_INSERT_VIEW
WWW_SELECT_VIEW

53 rows selected.
```

Appendix F: Triggers

```
SQL> SELECT TRIGGER_NAME, TRIGGER_TYPE, TRIGGERING_EVENT, TABLE_NAME FROM USER_TRIGGERS
  2  /

TRIGGER_NAME                   TRIGGER_TYPE     TRIGGERING_EVENT               TABLE_NAME
------------------------------ ---------------- ------------------------------ --------------------
AIUDFER_BUGS_TRIGGER           AFTER EACH ROW   INSERT OR UPDATE OR DELETE     BUGS
AIUDFER_EXPERIMENTS_TRIGGER    AFTER EACH ROW   INSERT OR UPDATE OR DELETE     EXPERIMENTS
AIUDFER_GKDB_LINKS_TRIGGER     AFTER EACH ROW   INSERT OR UPDATE OR DELETE     GKDB_LINKS
AIUDFER_PROJECTS_TRIGGER       AFTER EACH ROW   INSERT OR UPDATE OR DELETE     PROJECTS
AIUDFER_PROJECT_STATUS_TRIGGER AFTER EACH ROW   INSERT OR UPDATE OR DELETE     PROJECT_STATUS
AIUDFER_REQUESTS_TRIGGER       AFTER EACH ROW   INSERT OR UPDATE OR DELETE     REQUESTS
AIUDFER_RESPONSES_TRIGGER      AFTER EACH ROW   INSERT OR UPDATE OR DELETE     RESPONSES
AIUDFER_RESULT_BLOBS_TRIGGER   AFTER EACH ROW   INSERT OR UPDATE OR DELETE     RESULT_BLOBS
AIUDFER_RESULT_SETS_TRIGGER    AFTER EACH ROW   INSERT OR UPDATE OR DELETE     RESULT_SETS
AIUDFER_SAMPLE_SETS_TRIGGER    AFTER EACH ROW   INSERT OR UPDATE OR DELETE     SAMPLE_SETS
AIUDFER_SHIP_SAMPLES_TRIGGER   AFTER EACH ROW   INSERT OR UPDATE OR DELETE     SHIP_SAMPLES
AIUDFER_UPDATE_REQUEST_TRIGGER AFTER EACH ROW   INSERT OR UPDATE OR DELETE     UPDATE_REQUEST
AIUDFER_UPLOAD_FILES_TRIGGER   AFTER EACH ROW   INSERT OR UPDATE OR DELETE     UPLOAD_FILES
AIUDFER_WWW_LINKS_TRIGGER      AFTER EACH ROW   INSERT OR UPDATE OR DELETE     WWW_LINKS
BIFER_BUGS_ID                  BEFORE EACH ROW  INSERT                         BUGS
BIFER_EXPERIMENTS_ID           BEFORE EACH ROW  INSERT                         EXPERIMENTS
BIFER_FORMATS_ID               BEFORE EACH ROW  INSERT                         FORMATS

TRIGGER_NAME                   TRIGGER_TYPE     TRIGGERING_EVENT               TABLE_NAME
------------------------------ ---------------- ------------------------------ --------------------
BIFER_GKDB_LINKS_ID            BEFORE EACH ROW  INSERT                         GKDB_LINKS
BIFER_LABS_ID                  BEFORE EACH ROW  INSERT                         LABS
BIFER_LOGS_ID                  BEFORE EACH ROW  INSERT                         LOGS
BIFER_PROJECTS_ID              BEFORE EACH ROW  INSERT                         PROJECTS
BIFER_PROJECT_STATUS_ID        BEFORE EACH ROW  INSERT                         PROJECT_STATUS
BIFER_REQUESTS_ID              BEFORE EACH ROW  INSERT                         REQUESTS
BIFER_RESPONSES_ID             BEFORE EACH ROW  INSERT                         RESPONSES
BIFER_RESULT_BLOBS_ID          BEFORE EACH ROW  INSERT                         RESULT_BLOBS
BIFER_RESULT_SETS_ID           BEFORE EACH ROW  INSERT                         RESULT_SETS
BIFER_SAMPLE_SETS_ID           BEFORE EACH ROW  INSERT                         SAMPLE_SETS
BIFER_SERVICES_ID              BEFORE EACH ROW  INSERT                         SERVICES
BIFER_SHIP_SAMPLES_ID          BEFORE EACH ROW  INSERT                         SHIP_SAMPLES
BIFER_UPDATE_REQUEST_ID        BEFORE EACH ROW  INSERT                         UPDATE_REQUEST
BIFER_UPLOAD_FILES_ID          BEFORE EACH ROW  INSERT                         UPLOAD_FILES
BIFER_WWW_LINKS_ID             BEFORE EACH ROW  INSERT                         WWW_LINKS
IOFIDFER_BLOB_INSERT_VIEW      INSTEAD OF       INSERT OR DELETE               BLOB_INSERT_VIEW
IOFIDFER_BUG_REPORT_VIEW       INSTEAD OF       INSERT OR DELETE               BUG_REPORT_VIEW

TRIGGER_NAME                   TRIGGER_TYPE     TRIGGERING_EVENT               TABLE_NAME
------------------------------ ---------------- ------------------------------ ------------------------------
IOFIDFER_FILE_LOADING_VIEW     INSTEAD OF       INSERT OR DELETE               FILE_LOADING_VIEW
IOFIDFER_GKDB_INSERT_VIEW      INSTEAD OF       INSERT OR DELETE               GKDB_INSERT_VIEW
IOFIDFER_REQUEST_ENTRY_VIEW    INSTEAD OF       INSERT OR DELETE               REQUEST_ENTRY_VIEW
IOFIDFER_SAMPLE_SETS_INS_VIEW  INSTEAD OF       INSERT OR DELETE               SAMPLE_SETS_INSERT_VIEW
IOFIDFER_SERVICE_EXP_VIEW      INSTEAD OF       INSERT OR DELETE               SERVICE_EXPERIMENT_VIEW
IOFIDFER_SERVICE_RESPONSE_VIEW INSTEAD OF       INSERT OR DELETE               SERVICE_RESPONSE_VIEW
IOFIDFER_SHIP_SAMPLES_VIEW     INSTEAD OF       INSERT OR DELETE               SHIP_SAMPLES_VIEW
IOFIDFER_UPD_REQUEST_INS_VIEW  INSTEAD OF       INSERT OR DELETE               UPDATE_REQUEST_INSERT_VIEW
IOFIDFER_WWW_INSERT_VIEW       INSTEAD OF       INSERT OR DELETE               WWW_INSERT_VIEW
IOFIUDFER_SERVICE_RESULT_VIEW  INSTEAD OF       INSERT OR UPDATE OR DELETE     SERVICE_RESULT_VIEW
IOFIUFER_SESSION_TABLE_VIEW    INSTEAD OF       INSERT OR UPDATE               SESSION_TABLE_VIEW
IOFUFER_FILE_RECEIVE_VIEW      INSTEAD OF       UPDATE                         FILE_RECEIVE_VIEW
IOFUFER_PROJECT_MANAGER_VIEW   INSTEAD OF       UPDATE                         PROJECT_MANAGER_VIEW
IOFUFER_SHIP_RECEIVE_VIEW      INSTEAD OF       UPDATE                         SHIP_RECEIVE_VIEW
IOFUFER_UPD_PRO_ACTIVE_VIEW    INSTEAD OF       UPDATE                         UPDATE_PROJECT_ACTIVE_VIEW
IOFUFER_UPD_PRO_STATUS_VIEW    INSTEAD OF       UPDATE                         UPDATE_PROJECT_STATUS_VIEW

50 rows selected.
```

Vita

Hsin-Neng Wang was born in Taipei, Taiwan on May 3, 1975. He graduated from the department of biology, Fu-Jen Catholic University, Taipei and received his Bachelor's degree in biology in 1997. During the academic years in the university, he worked as an Undergraduate Research Assistant in the Institute of Biomedical Sciences, Academia Sinica, and did research for an undergraduate research project funded by the National Science Council, Taiwan, R.O.C.

From 1997 to 1999, he served in Taiwan Marine Corps as a corporal. From 1999 to 2001, he worked as a Research Assistant in the Hepatitis Research Center, National Taiwan University Hospital.

In the fall of 2001, he enrolled in the Genome Science and Technology Graduate School of the University of Tennessee, Knoxville and Oak Ridge National Laboratory, and received his Master of Science degree in December 2003.

3872 2089 88
04/14/04 MAB