A WEB BASED DISTRIBUTED MEDICAL RECORD AND IMAGING ENTRY AND VISUALIZATION SYSTEM

By

WEI ZHENG

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2000

Dedicated to My Parents

ACKNOWLEDGMENTS

I would like to express my gratitude to the chairman of my supervisory committee, Dr. Richard Newman, for his invaluable advice, guidance and patience. Also I would like to express my sincere thanks and appreciation to Dr. Vincent Frouhar, for providing continuous guidance, excellent advice and support throughout the course of my work. It was really a great pleasure and precious experience working with him. I also thank Dr. Sanguthevar Rajasekaran for serving on my committee and the many words of advice he gave me during my study in this department.

Last, but not the least, I thank my parents for their love. Without their encouragement and endurance, it would be hard for me to have this achievement.

TABLE OF CONTENTS

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

A WEB BASED DISTRIBUTED MEDICAL RECORD AND IMAGING ENTRY AND VISUALIZATION SYSTEM

By

Wei Zheng

August 2000

Chairman: Richard Newman
Major Department: Computer and Information Science and Engineering

This thesis presents the design and implementation of a web based distributed medical record and imaging and visualization system. Sophisticated yet easy to use, the RCET_Sub system lets users interactively input, manipulate and manage medical information on the World Wide Web.

The system is based on a distributed heterogeneous database system. The database replications are set up between external database1 (Oracle7/8i) and RCET (Oracle8/8i), between external database2 (Microsoft Access) and RCET (Oracle8/8i). Therfore any change to the external database1 or external database2 can be available to RCET_Sub users.

The system provides an advanced graphical user interface (GUI) on the Internet. Using the interface, users can perform all the database operations allowed by the medical protocol. The interface changes dynamically with data and result of transactions. To be

error proof, the system was developed to provide all necessary system information and instructions to users.

Image database is another important feature of the system. The images are managed by the database. Users can use the interface to insert and delete images from the database.

Testing is carried out according to the specifications. Results show that the system is a stable, robust and efficient system.

.

CHAPTER 1
INTRODUCTION

Today, the World Wide Web changes every aspect of human's life. It is fast, interactive, multimedia, and allows for multi-tiered information access. Medicine in general and radiation oncology specifically are deriving great benefits from this technology.

1.1 Overview of the RCET and RCET_Sub

Resource Center for Emerging Technologies (RCET) system is a web database system for collecting, sharing and distributing information generated by participating Quality Assurance (Q/A) institutions. The system consists of a centralized database, web servers, 3D data visualization ActiveX and Java applications and a file transaction server. The RCET_Sub system is a part of the RCET system. The RCET_Sub system mainly deals with the medical record and imaging entry and management. The RCET_Sub system is to provide the participating institutions real time and on demand archived data which include time dependent treatment delivery documentation and imaging. The benefits of RCET_Sub system are time saving, information sharing and efficient management.

1.1.1 Time

In the medical area, time is life. In many cases, the timing of these activities is critical. For example, in most protocols with a radiotherapy component, the submitted information needs to be independently reviewed prior to commencing a patient's

treatment. In some cases, this process takes place during or immediately after the treatment period and/or the review, verification, notification and any correction must occur within 24-48 hours of treatment.

With on-line, real-time RCET_Sub system, the data form submission schedule is created and stored in the computer as soon as the patient is registered on a study. The logic associated with protocol treatment is built into web interface. To submit patient related data, the user only needs to log onto the system, select a patient, and choose the data form. The correct form appears on the screen permitting on-line, real-time data entry. Much of the data entered are checked for validity. The user will be able immediately to correct any errors in the data before it is stored in the database. The local investigators can view data from their own institution and print both blank and completed data screens for their local paper files.

1.1.2 Information Sharing

RCET_Sub system provides a set of services for the Domain of Collaboration, which is defined as RCET, UF Radiotherapy clinics, POG (Pediatric Oncology Group), QARC (Quality Assurance for Radiation Oncology Clinics), RTOG (Radiation Therapy Oncology Group) and other Q/A centers, individuals, and institutions that participate in a protocol-driven study. Participants in a protocol study can send information required for each study case, using an easily accessible browser interface. A typical set of data includes textual information, image sets, and planning information. Web technology allows protocol participants easily to enter and to access textual information in a unified database system. One set of data can be shared by all the participating institutions through the web locally or remotely. In addition, the system must allow the participants to enter the information at any site, share, and analyze it remotely on demand.

Reducing information redundancy and quickly sharing information are the benefits to the participate institutions. RCET_Sub system provides a time- and cost-efficient system to share and manage the information.

1.1.3. Efficient Management

Once a patient is entered on to a study protocol, there is an extensive set of data to be received, archived, categorized, analyzed and disseminated by the RCET_Sub centralized database. Typically, a set of patient data includes 50-400 MB of image data, 1-10 MB of plan data and extensive information regarding patient history, diagnosis, treatment, and outcome. This set of data is large and requires very efficient managing and archiving mechanisms. The main issue has been to design and implement a system, which functionally analyzes, stores, and presents the data on demand. In most cases, the goal is to monitor a patient's progress by correlating and viewing sets of images. Thus, the database design and the tools required for presenting the images and RT plan must be sophisticated enough to provide correlated information at the visual and conceptual level.

The treatment planning visualization and image collection in conjunction with the textual data has neither been developed nor implemented by collaborative groups. One of our goals has been to link textual data, electronic imaging, and treatment planning data and provide a system to manage, archive, and analyze the data efficiently.

1.2 Organization of the Thesis

I began this thesis by explaining the reasons we develop RCET_Sub system and the benefits RCET_Sub system can bring to the radiation oncology field in Chapter 1. Then in Chapter2, I overviewed the technology involved in this project. These underlying techniques are very necessary for understanding the design and

implementation of RCET_Sub system. From Chapter 3, we begin to concentrate on the system design and implementation. We outline some of the software requirements in Chapter 3. In Chapter 4, we discuss the system high-level design. With the design in mind, we discuss the system implementation in detail in chapter 5. Chapter 6 presents test plans to verify whether RCET_Sub system meets the specifications.

CHAPTER 2
UNDERLYING TECHNOLOGIES

Before we discuss RCET_Sub system, we would like to take a close look at some of the important technologies underlying the system. According to the different technology domain used in this project, we can divide the technologies into three groups: database, sever technologies and web interface. In the database part, we will discuss the database link for heterogeneous database, replication and how to put image in the database. We also will discuss active scripting for web sites and security for the WWW in server side. In the end, we will discuss medical objects and technology, overview of the technologies for visualization tools in the browsers. These are the technologies used for the user interface.

2.1. Database Link for Heterogeneous Database

To link the individual databases of a distributed database system, a network is necessary. Also since the databases are heterogeneous, it is necessary to use different software to build up the link.

2.1.1 ODBC

The Standard Open Database Connectivity (ODBC) is Microsoft's strategic interface for accessing data in a heterogeneous environment of relational and non-relational database management systems. Using ODBC interface, an application written in SQL can access remote data.

In the ODBC architecture, the link between an application and server requires the use of a driver, a library that is dynamically connected to the applications. The driver masks the differences of interactions due not only to the DBMS, but also the operating system and to the network protocol used. The driver thus masks all the problems of heterogeneity and facilitates the writing of applications. In order to guarantee the compatibility with the ODBC standard, each DBMS supplier must guarantee drivers that allow for the use of that DBMS within the environment of a specific network and with a specific operating system [1]. The structure of ODBC is shown in figure 2.1.

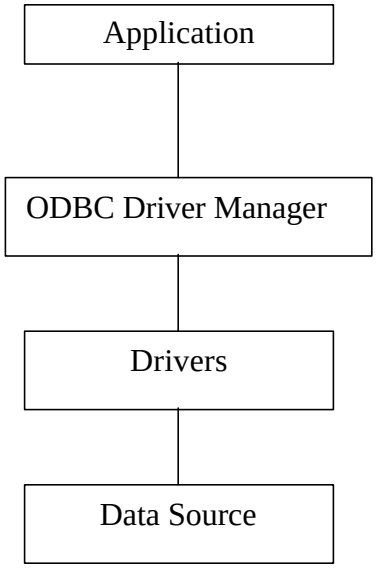


Figure 2.1: Architecture of ODBC [1]

2.1.2 JDBC

JDBC technology is an API that lets user access virtually any tabular data source from the Java programming language. It provides cross-DBMS connectivity to a wide range of SQL databases. The JDBC API allows developers to take advantage of the Java platform's "Write Once, Run Anywhere" capabilities. With a JDBC technology-enabled

driver, a developer can easily connect all corporate data even in a heterogeneous environment [2].

2.1.3. SQL*NET and NET8

In Oracle, the data links between client-server or server-server is provided by NET8 (SQL*Net). Net8, the networking layer formerly known as SQL*Net, uses the Transparent Network Substrate (TNS) and standard industry network protocols to connect a client to a server and establish an Oracle session. The following figure (figure 2.2) shows the basic connectivity architecture and process, with a connection request originating from a client side application. The database returns the queried information in the reverse direction. The role of Net8 is to establish and maintain a connection between the client application and the server and exchange messages between them. It provides both client to server and server to server communications across any network. It enables client tools to access, modify, share, and store data on Oracle8 Servers over a network. The communication between client applications and servers takes place across one or more networks and is referred to as client/server communication [3].

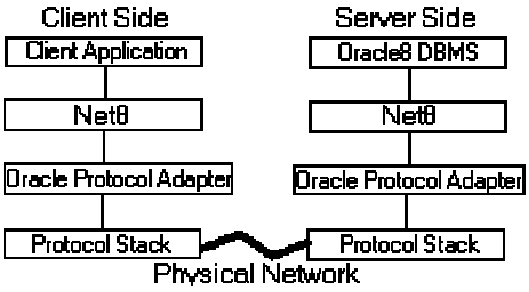


Figure 2.2: Using NET8 (SQL*NET) to connect [3]

2.2. Replication

Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system. Replication can improve the performance and protect the availability of applications because alternate data access options exist. In the RCET database, an application might normally access a local database rather than a remote server to minimize network traffic and achieve maximum performance. Furthermore, the application can continue to function if the local server experiences a failure, but other servers with replicated data remain accessible. Database replication can be divided into two kinds depending on different ways of propagation.

- Asynchronous propagation, also known as deferred transactions, is when changes are deferred or queued at the site where they occur and are pushed to the other site at timed intervals.

- Synchronous propagation, or real-time propagation, is when the changes are applied to the local replicate and to the other entire replication sites in a single transaction.

2.2.1. Snapshots

In Oracle, asynchronous propagation is obtained by snapshot. A read-only table snapshot is a local copy of table data that originates from one or more remote master table. An application can query the data in a read-only table snapshot, but cannot insert, update, or delete rows in the snapshot. The following figure (figure 2.3.) explains about read-only table snapshots and basic replication. The data that a snapshot presents does not necessarily match the current data of its master tables. A table snapshot is a transaction-consistent reflection of its master data as that data existed at a specific point

in time. To keep a snapshot's data relatively current with the data of its master, Oracle must periodically refresh the snapshot. A snapshot refresh is an efficient batch operation that makes that snapshot reflect a more current state of its master [4].

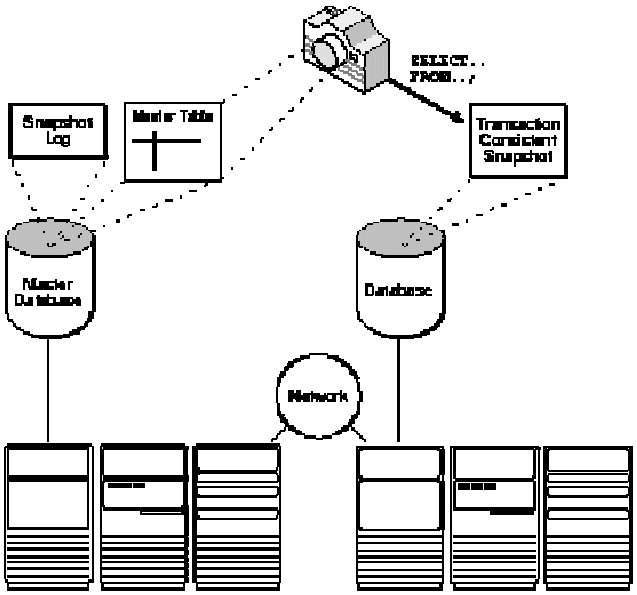


Figure 2.3: Mechanism of Snapshots [4]

2.2.2. Triggers

For Synchronous propagation, we can use triggers to maintaining duplicated data in Oracle. Once there is an event changing the data in the master table, a trigger will be fired to let the duplicated data be updated. The trigger updates replicated data on a transaction-by-transaction basis.

For data replication between Microsoft Access and Oracle, Microsoft Access only support the trigger mechanism. In Microsoft Access, we create a trigger event procedure when data change is in the master table. The trigger event procedure needs to be written in Microsoft Visual Basic and makes use of the ODBC link.

2.2. 3. Snapshots vs. Triggers

Both snapshots and triggers can maintain data replication. When choosing between them, we need consider the following points.

The following points are in favor of snapshots.

- Replication-type triggers are executed each time a row changes in the master table. If the table is transaction-intensive, this may place a large burden on network. Snapshots bundle all of the changes into one single transaction.

- Trigger transactions can not be scheduled, they occur when the transactions on the master tables occur. Snapshot replications can be scheduled. It can avoid large replication transaction loads during peak usage times.

Triggers are better under the following circumstances.

- If the data in the replica has to be constantly synchronized with the master table, then we have to use triggers.

- If the master table has a trigger on it, and the transaction on the remote copy fails, then the transaction on the master table will fail too, thus maintain the consistency of the two tables.

Which mechanism to choose all depends on the requirements of the application.

2.3. Image in Database

With the development of database technology, management of multiple media is becoming an important aspect of database systems. In the RCET_Sub system, we deal

with many different formats of image such as X-ray images, CT images and so on. Management of all the medical images is a major part of the system.

To solve the problem of multimedia in data base, Oracle8 supports LOBs - large objects which can hold up to 4 gigabytes of raw, binary data (e.g., graphic images, sound waveforms, video clips, etc.) or character text data. Oracle8 regards LOBs as being of two kinds depending on their location with regard to the database -- internal LOBs and external LOBs, also referred to as BFILEs (binary files).

Internal LOBs, as their name suggests, are stored inside database tablespaces in a way that optimizes space and provides efficient access. Oracle treats an Internal LOB the same as other Oracle object (like number, string…). In other words, all the ACID properties (Atomic, Consistency, Isolation, Duranility) that pertain to using database objects pertain to using internal LOBs. An internal log can be up to 4 gigabytes in size.

External LOBs (BFILES) are large binary data objects stored in operating system files outside of database tablespaces. External LOBs do not participate in transactions. Any support for integrity and durability must be provided by the underlying file system as governed by the operating system. The BFILE maximum size is operating system dependent [5].

2.4. Active Scripting for Web Sites

Nowadays, web users want content fresh, exciting, up-to-date, and tailored to their personal information needs. This expectation requires a new web-development method - active scripting. According to the location, active scripting can be divided into client- and server side- scripting.

2.4.1. Microsoft Active Server Pages (ASP)

Microsoft Active Server Pages (ASP) is a server-side scripting environment that you can use to create and run dynamic, interactive Web server applications. With ASP, you can combine HTML pages, script commands, and ActiveX components to create interactive Web pages or powerful Web-based applications. When you make a change on the ASP file on the server, you need only save the changes to the file—the next time the Web page is loaded, the script will automatically be compiled.

When a browser requests an ASP file from web server, the web server calls Active Server Pages to read through the ASP file, executing any of the commands contained within and sending the resulting HTML page to the browser. An ASP file can contain any combination of HTML, script, or commands. The script can assign values to variables, request information from the server, or combine any set of commands into procedures. ASP uses the delimiters ("<%" and "%>") to enclose script commands. Also ASP includes five standard build-in objects for global use [6]:

- Request—to get information from the user,
- Response—to send information to the user ,
- Server—to control the Internet Information Server,
- Session—to store information about and change settings for the user's current Web-server session, and
- Application—to share application-level information and control settings for the lifetime of the application.

ASPs are web pages that are preprocessed by the web server before being sent to the client web browsers. Because these pages are preprocessed, the web server has to support Active Server Pages. Microsoft Internet Information Server (IIS) and Personal Web Server (PWS) support ASP.

2.4.2. Macromedia Drumbeat 2000

Macromedia Drumbeat 2000 is the fastest, easiest way to build web interfaces to databases and applications that take full advantage of ASP and automatically work in different browsers. Drumbeat 2000 is a very powerful tool and has many features for web database building [7]. We just discuss the features used in this thesis.

- Powerful DataForm Wizard**:** With the new DataForm Wizard, the user can create Active Server Pages that include a full range of database interactions without writing a line of code. The wizard can create drop-down lists, checkboxes, radio buttons and edit boxes within the Wizard, or use the Data Loop component to display multiple records on a page.

- SQL Query Builder**:** This allows the user to construct simple database queries right within Drumbeat or import Views and Queries that have been constructed in other ODBC database applications; access stored procedures that execute SQL logic on the server; and use dynamic queries to execute SQL logic at run time with real-time variables.

- Server Side Scripting with JavaScript**:** When you enable ASP support for a web site, the user can use JavaScript for server-side scripting. The Script Center makes writing client and server-side scripts in JavaScript easy and

convenient, with a scripting tree that contains all the objects, methods, properties and events you need.

- Interactions Center: The interactions center lets user view, filter and apply interactions to the web applications. Sever-side COM and ASP object can be included in the point-and-click interactions.

Drumbeat 2000 is designed for Web-site builders who need to speed the deployment of database-driven web applications that access and update data in real-time.

2.4.3. Cookies and ASP Session

A cookie is a token that the Web server embeds in a user's Web browser to identify the user. The next time the same browser requests a page, it sends the cookie it received from the Web server. Cookies allow a set of information to be associated with a user.

One of the challenges to developing a successful Web application is maintaining user information over the course of a visit (session), as the user jumps from page to page in an application. HTTP is a stateless protocol, meaning that your Web server treats each HTTP request for a page as an independent request; the server retains no knowledge of previous requests, even if they occurred only seconds prior to a current request. ASP provides a unique solution for the problem of managing session information. Using the session object and a special user ID generated by your server, you can create clever applications that identify each visiting user and collect information that your application can then use to track user preferences or selections.

The first time a user requests an ASP file within a given application, ASP generates a SessionID. The SessionID uniquely identifies each user's session. At the

beginning of a new session, the server stores the SessionID in the user's Web browser as a cookie. All the session variables in this session are associated by the SessionID. By reusing the SessionID cookies, ASP minimizes the number of cookies sent to your browser [8].

2.5. Security for the WWW

With the expansion of the World Wide Web, people and businesses have access to more information than ever before. So security become a serious topic.

Secure Sockets Layer (SSL) protocol has been universally accepted on the World Wide Web for authenticated and encrypted communication between clients and servers. The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP or IMAP. It uses TCP/IP on behalf of the higher-level protocols, and in the process allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection.

The heart of the SSL consists of the two protocol layers: a Handshake protocol and a Record Layer protocol. The Handshake protocol is responsible for establishing the write keys (communication session keys for data privacy) and MAC (message authentication check for data integrity) secret and for validating the authenticity of clients and servers. The Record layer protocol is responsible for fragmentation, compression/decompression and encryption/decryption of message records [9].

The handshake protocol involves server (and optionally client) authentication, determining what cryptographic algorithms are to be used, and the generating of a secret

key by which all future SSL Data Exchanges are to be encrypted by. The steps of handshake protocol are shown in the following graph [10].

After the handshake, Secure socket layer has been established data is ready to be exchanged. All subsequent socket messages are encrypted with the newly established cipher algorithm and secrete keys.

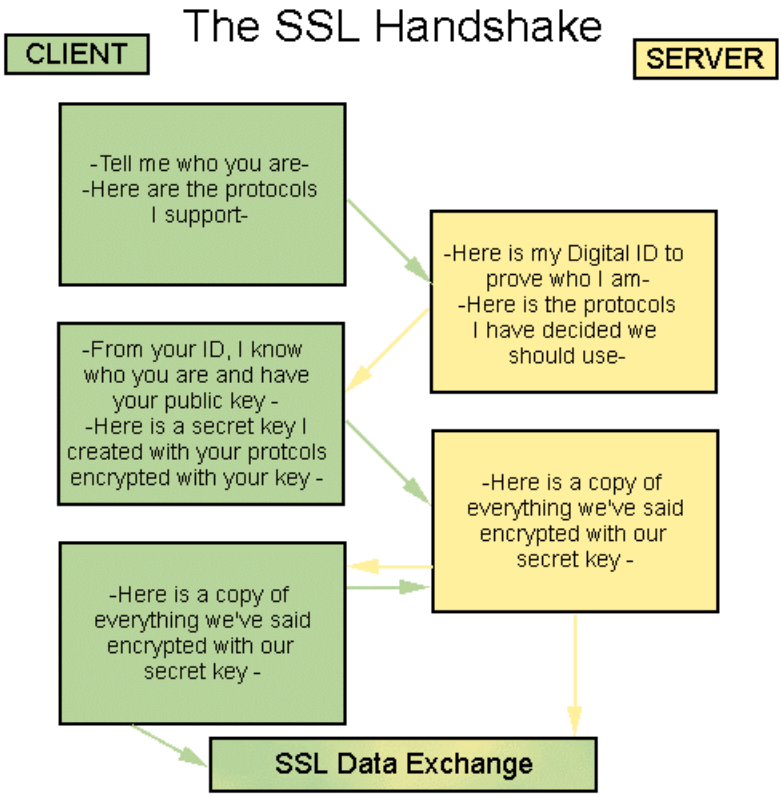


Figure 2.4: Mechanism of SSL [10]

2.6. Medical Object and Technology

To transfer and manage medical information electronically, many medical objects and technologies are developed. Here, we only discuss the medical object and technology used in the system.

In the 1980's, it became clear to radiologists and the manufacturers of medical imaging equipment that the tremendous growth in image acquisition systems, display workstations, archiving systems, and hospital-radiology information systems made vital the connectivity and interoperability of all pieces of equipment. In order to simplify and improve equipment connectivity, medical professionals joined forces with medical equipment manufacturers in an international effort to develop DICOM, the Digital Imaging and Communications in medicine standard. When DICOM is built into a medical imaging device, it can be directly connected to another DICOM-compatible device, eliminating the need for a custom interface. DICOM defines the interface.

There are four DICOM RT objects that have been implemented today [11]:

- The RT Structure Set contains information related to patient anatomy, for example structures, markers, and isocenters. These entities are typically identified on devices such as CT scanners, physical or virtual simulation workstations, or treatment planning systems.

- The RT Plan contains geometric and dosimetric data specifying a course of external beam and/or brachytherapy treatment, for example beam angles, collimator openings, beam modifiers, and brachytherapy channel and source specifications. The RT Plan entity may be created by a simulation workstation, and subsequently enriched by a treatment planning system before being passed on to a record and verify system or treatment device. An instance of the RT Plan object usually references a RT Structure Set instance to define a coordinate system and set of patient structures.

- The RT Image specifies radiotherapy images that have been obtained on a conical imaging geometry, such as those found on conventional simulators and portal imaging devices. It can also be used for calculated images using the same geometry, such as digitally reconstructed radiographs (DRRs).

- The RT Dose contains dose data generated by a treatment planning system in one or more of several formats: three-dimensional dose data, isodose curves, DVHs, or dose points.

- The RT Brachy Treatment Record, and RT Treatment Summary Record, contain data obtained from actual radiotherapy treatments. These objects are the historical record of treatment, and are linked with the other “planning" objects to form a complete picture of the treatment.

DICOM is now a mature standard, and the core radiotherapy objects have been ratified since 1997. After a lot of hard work understanding, developing and testing product inter-operability in the radiotherapy context, a large number of manufacturers now have products available that support one of more of the radiotherapy DICOM objects.

2.7. Overview of the Technology for Visualization Tools in the Browsers

Visualization tools are very important component of the web browser. There are many technologies available. We only discuss the technologies used in the system.

2.7.1. Java Applet

A Java applet is a particular type of Java program that is intended to be embedded in a web page. Technically, applets are subclasses of the panel container of Abstract

Windowing Toolkit (AWT). The derive much of their function and form from panels, just as the process of imbedding container upon container makes panels so powerful, the same is true about the versatility of applets.

Java applets are automatically downloaded from the serer and executed on the client. They can significantly increase the visual data interaction between the client application and user, and allow tasks to be executed on the client. Java applets are interpreted on the client by the Java Virtual Machine (JVM), which is usually embedded in the Java-enabled browsers such as Netscape's Navigator, Microsoft's Internet Explorer and so on. Java applets, however, must be download every time they are used and performance therefore depends much on the available bandwidth.

2.7.2.COM and DCOM

The Component Object Model (COM) is a software architecture that allows applications to be built from binary software components. COM is the specification of how application components communicate and control each other in a Windows environment. COM is supported and facilitated by the operating system.

COM and its successor DCOM (Distributed COM) developed by Microsoft, are methodologies for the creation of binary software components. Thus, COM allows developers to create software building blocks that can be used by their clients without intimate knowledge of the component, but still lets them leverage the expertise of the developer. DCOM also allows for network-enabled inter-process communications and so makes it feasible to let software entities running on different machines work together as if the components were located on the same machine. In fact, from the programmer's point of view, it does not make a difference if the components are located on the same or on

different machines. Thus, DCOM can, for example, be used to let several machines connected via a network work in parallel on the same problem [12].

2.7.3. ActiveX Controls

ActiveX controls are among the many types of components that use COM technologies to provide interoperability with other types of COM components and services. ActiveX controls are typically small applications (though they can be any size, and some are huge) that can operate only within a compatible container application.

Web component technology based on Microsoft's ActiveX architecture represents the framework for a next generation visualization system. Visual data manipulation is provided at the client side through industry-standard components. Highly interactive user interface tasks can be delivered that provide point-and-click navigation and information drill-down through multidimensional data structures. Visual data interfaces such as information drilling, moving a cutting plane through an anatomical volume data set, etc. can be supported. Unlike Java applets, which remain on a system only while they are used, ActiveX components stay on the system. Therefore, they do not need to be downloaded each time they are going to be used. In this way they are similar to browser plug-ins. Clearly, full-featured visual data analysis tools based on the ActiveX component technology has many advantages over the rudimentary offerings of Java applets and HTML query forms.

ActiveX controls can be embedded within Web browsers like Internet Explorer and applications such as Office97. Since ActiveX controls can be written in any language, they let application developers use the performance offered by languages such as C and C++ to solve a certain time-critical problem.

With the increasing use of Microsoft's electronic documents (Office 97), distributed by Intranet and Internet, the opportunity to provide easy-to-use, advanced interactive data visualization techniques within electronic documents on the powerful PC desktop has become possible. Customizable components allow the author of a report to distribute the relevant electronic information coupled with an embedded data analysis-viewer smartdoc, which allows the recipients to interactively examine the data in the same way as the original analyst. Instead of being a dumb document, the smartdoc would be a complete data exploration application through which a reader could, via embedded ActiveX visualization/analysis components, explore the data underlying the report. Instead of being a static summary document, the report becomes much more like a very natural interface or portal onto the data [13].

2.8. Summary

This chapter elaborated the technologies that underlie the RCET_Sub system. In the database part, we discussed the technologies involved in database link and database replication. We also discussed the Oracle solution to image database. We then overviewed the active scripting and security technology for WWW. We also gave some background information about the medical object and technology used in this system. In the end, we overviews the technologies for visualization tools in the browser.

In the next chapter, we will discuss the RCET_Sub system’s requirements in database and user interface.

CHAPTER 3
SOFTWARE REQUIREMENTS

RCET system is a practical system with the mission to receive, validate, archive and present medical information through the Internet. The ultimate requirement of the RCET system is to provide a user-friendly web based interface with fully featured functions. We can group the requirements into two parts. They are database and web interface. Their requirements will be presented separately.

3.1. Requirements of Database

All database applications have common characteristics, such as the ability to handle a group of tables or schemas, various data operation, and so on. We are going to tailor these general requirements to our specific needs.

1. User can perform insert, update and delete operation to the data in database using the web interface.
2. According to the medical information, a set of pages should be related by RCET number. There is only one unique RCET number for a page set. Different page sets have different RCET numbers.
3. If the tables and data corresponding to the RCET number exist in the database, the system can fetch the data and dynamically produce the page set. Meanwhile, the user is not allowed to insert new data according to this RCET number. The operations available to users are updating and deleting.

4. According to medical protocol, the tables in the database can be divided be into two groups: base tables and extent tables. One row in base table will point to an extent table.
5. All the pages can be divided into upper part and lower part. The upper is populated by one row of data in the base table. The lower part includes extent tables.
6. If the data in the upper part is deleted, all the tables in the lower part are deleted in cascade fashion.
7. Medical images should be a part of database and user can perform database operations on the images with the corresponding textual data through the web interface.
8. When users upload the images, the system should automatically organize them and put them in database.
9. When users delete images, the system should delete them from the database and delete the files from the disk.
10. Replication should be built in the RCET database (Oracle 8/8i) to external database1 (Oracle 7/8i). So that all data in the Oracle 7/8i database will be available to the RCET database. If there is any change to the external database1, the RCET database must be updated too.
11. Replication should be built in the RCET database (Oracle r/8i) to external database2 (Microsoft Access). So that all data in the external database2 will be available to the RCET database. If there is any change to the external database2, the RCET database must be updated too.

3.2. Requirements of Web User Interface

Because the users of this system are medical professionals, not necessary programmers, we should emphasize user-friendliness and intuitive features of the web interface. Moreover, The system will be used in clinic where high pressure and intensity are a part of everyday life. Users under such condition are nervous and very likely to make mistakes. User-friendliness and error prevention of the web user interface is essential to the RCET_Sub system.

12. The interface should be consistent with the existing forms (in paper), so additional training for the operators is not needed.
13. The system should give a reaction page to show whether the operation succeeded or not once an operation is submitted to the server.
14. The system should enable/disable buttons according to the medical protocol and database requirement to prevent user from performing wrong operations.
15. Upon a successful transaction, the data frame should be populated with the new data. For an insert operation, the blank data field is filled in with new data. For an update operation, the changed data is shown. Also after a successful transaction, the availability of the button changes.
16. Users are required to input some basic information (e.g., RCET Number, Last Name…) according to the medical protocol in the basic information page. Without the basic information, system won’t allow any further operation. All the basic information can be updated and deleted only in the first page.

17. Common fields in different pages should be identical. Once the users fill them in the first time, the users won't need to provide them in the other pages.
18. The system should do form verification according to the medical protocol before submission of the form. If an error is found, the data will not be submitted, and the system should give a warning message that helps user to come up with the right data.
19. The system should give a warning message instead of performing the operation if the user pushes any button with no data to submit or pushes a button repeatedly.
20. The system should give a confirmation page before a delete operation. This is to prevent the user from deleting the data when accidentally pushing the DELETE button.
21. The interface should be optimized to reduce network transactions. This is very important for RCET system because the system will do a lot of image transaction. Speed will be a major issue.

There are some additional security requirements that are not addressed here. With all the system requirements, we can design and implement the system.

CHAPTER 4
SYSTEM HIGH-LEVEL DESIGN

With the software requirements and system architecture in mind, we are going to present a high-level design of the system.

4.1. System Architecture

Before we present the system design in detail, we would like to view that system in a big picture – the system architecture. It is not possible to understand the design well without understanding the system architecture.

Generally, the system has three major components: web browser, web server and database server.

4.1.1.Web Browser

The web browser has the web textual interface and visualization tools. The use can enter textual information through the textual interface. After successful validation of the textual data, the data is submitted to the web server. On the other hand, when the user wants to submit images, the visualization tools can be launched from the web browser. The user can input or manage the image using the visualization tools. After the submission request from the user, the visualization tools will submit the image data to the web server using the upload mechanism to a temporary directory.

4.1.2.Web Server

The web server will take care of encryption/decryption and organization of the data. After receiving input data from the client, the web server will send the textual data

to the database server through the database interface. As for the image data, the web server will move the files from the temporary directory to the designed directory, then put the pointer to the image file into the database. After a successful transaction, a transaction success page will be generated by the web server and sent to the web browser. Meanwhile, if the web server receives search requests from a client browser, it will query the database server and generate the pages with the reply data from the database. It will send the textual data to the textual interface and the image data to the visualization tools. Upon receiving the image data, the visualization tools will be launched to display the corresponding information in the client browser.

4.1.3.Database Server

The database server will deal with organizing, managing, archiving the textual and image data. The database in the system is a distributed database. It consists of data replicated from external database1 and external database2.

The system architecture of the system is shown in figure 4.1.

In the rest of the chapter, we will discuss the design of the individual components of the system in detail.

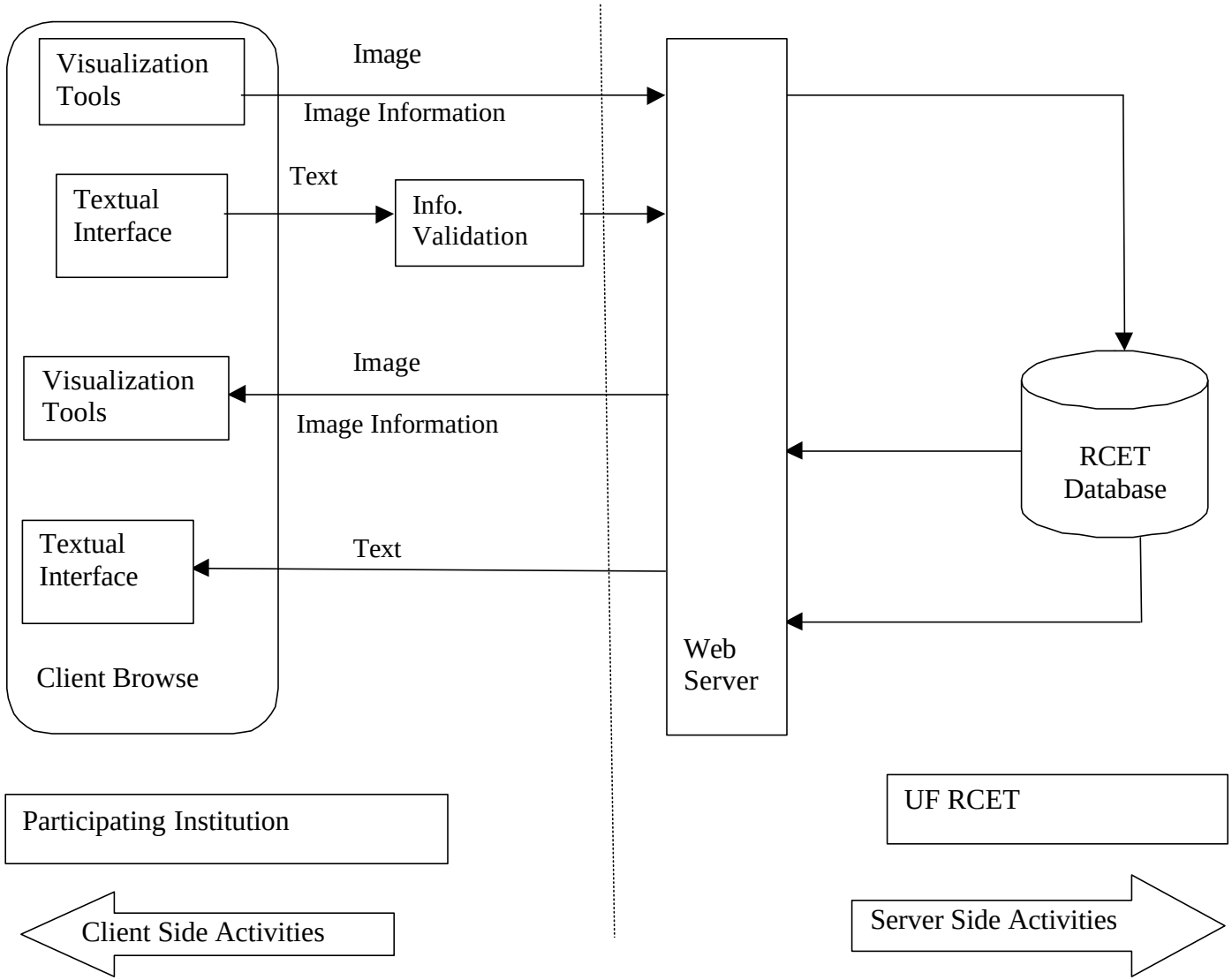


Figure 4.1. System architecture of the system

4.2. Replication

From the security and efficiency requirements, we know that replications are needed to setup between RCET database (Oracle 8/8i) and external database1 (Oracle 7/8i), between RCET database (Oracle 8/8i) and external database2 (Microsoft Access). Figure 4.2 is the structure of the replications.

The first set in replication is setup data link. To support application access to the data and schema objects throughout distributed database system, database links needed to be created.

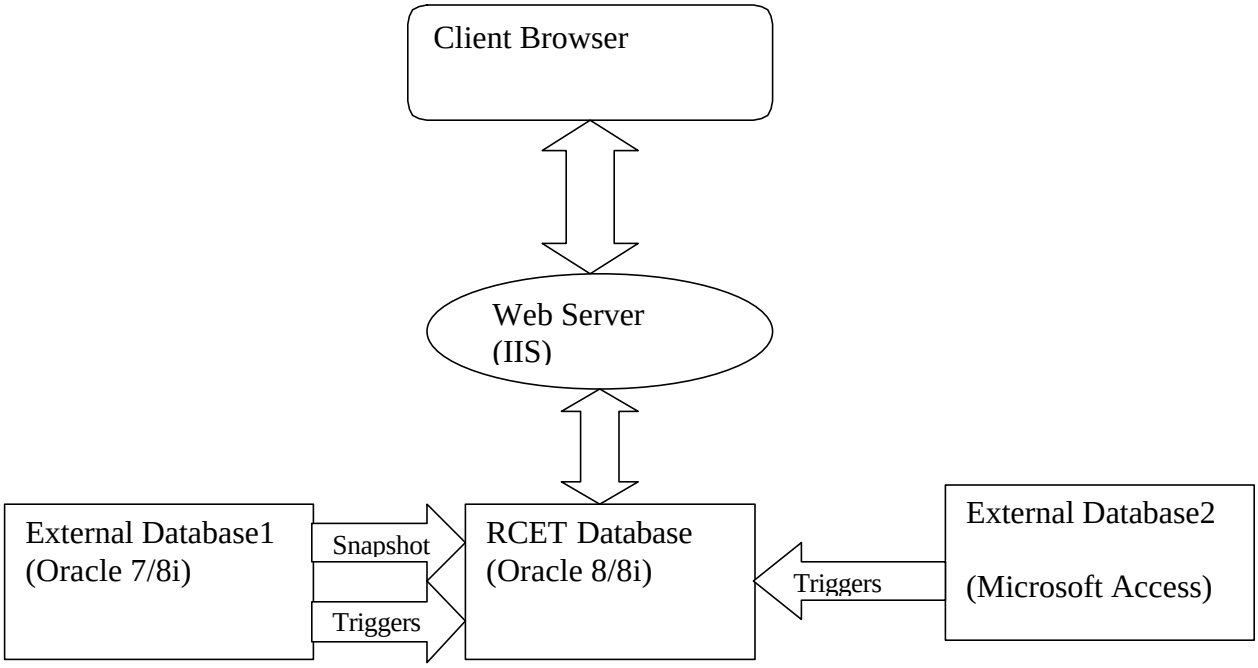


Figure 4.2. Database structure of the RCET_Sub system.

Between Oracle servers, we used Net8 to link them together. After we installed and verified the setup of Oracle Networking Products, we configured a network by creating the client configuration files. After configuration, we can query the data on the external database1.

Between Oracle and Microsoft Access, we need to set up the data source name (DSN). An ODBC system DSN stores information about how to connect to the indicated data provider. A system data source is visible to all users on the machine. Using the ODBC drivers provided by Oracle and Microsoft, we can access the oracle data in Microsoft Access.

For the replication between RCET and external database1, we divided the master tables in external database1 into two groups based on the requirements of the application. In one group, the tables need to be synchronized with the master table in every transaction. Then we used triggers to maintain replication. In the other group, the tables don't need to be synchronized very frequently. We made snapshots on them and scheduled the refresh time at night to avoid the peak usage times.

For the replication between RCET and external database2, Microsoft Access only supports the trigger mechanism. In the Microsoft Access, we create a trigger event procedure when data change in the master table happens. The trigger event procedure needs to be written in Microsoft Visual Basic and make use of the ODBC link.

4.3. Design of Database Schema

According to the medical protocol, all tables in the database should be related by RCET number. RCET number is unique in one table, so we chose the RCET number as the primary key to all the tables in the database.

According to medical protocol, the tables in the database can be divided be into two groups: base tables and extent tables. The base tables are the tables holding the information that appears in the upper part of every page. Meanwhile, the extent table is the tables in the lower part of every page. Actually, the information shown in the upper part is just one row of the base table. Each row in the base table has a field that points to an extent table. The relationship between the base tables and extent tables is shown on figure 4.3. The base tables and extent tables are related by the foreign key relationship. We also enable the delete cascade so that when one row in the base table is deleted, the tables correspond to that row will be deleted too.

Image database operations are very important features of the system. The way we store the image in database is by storing the relative pointer to the image file. Because of the following reasons we did not choose the LOB mechanism in Oracle.

- If using LOBs, the tablespace will be very huge.
- If using LOBs, Oracle will treat it as common database object, so that redo log and rollback log will be huge. It will greatly increase the volume of transactions so as to decrease the speed of all transactions.
- This will put heavy load on the ODBC link.

Using the relative pointer to the image files, the address of image files will be generated dynamically according to the virtual directory structure of the web server.

RCET NUMBER	**1**	**2**	…	**TABLEID**
1				10
2				20
…	…	…	…	…

BASE TABLE (BASE_TABLE)

1	**2**	**…**	**TABLEID**
			10
			10

EXTENT TABLE 1(EX_TABLE1)

1	**2**	**…**	**TABLEID**
			20
			20

EXTENT TABLE 2 (EX_TABLE2)

.
.
.
.

Figure 4.3. The relationship between the base tables and extent table

There are two operations the system supports to be performed on images in the database: insert and delete.

- For Insert Operation, user uploads the images in a temporary folder in RCET. After the user submits the text information related to the image to database, the database will trigger an operating system process to move the files to a particular directory and store the file name in the database. The particular directory is decided by the primary key in the database associated with this image. Figure 4.4 shows how the insert operation is performed.

- Delete Operation: After the user submits the request to delete an image, the database will trigger an operating system process to remove the files. After successful removal, the database will remove the corresponding record from the database. The mechanism of delete is shown in figure 4.5.

In this way, we can provide the user with image database operations with relative high speed.

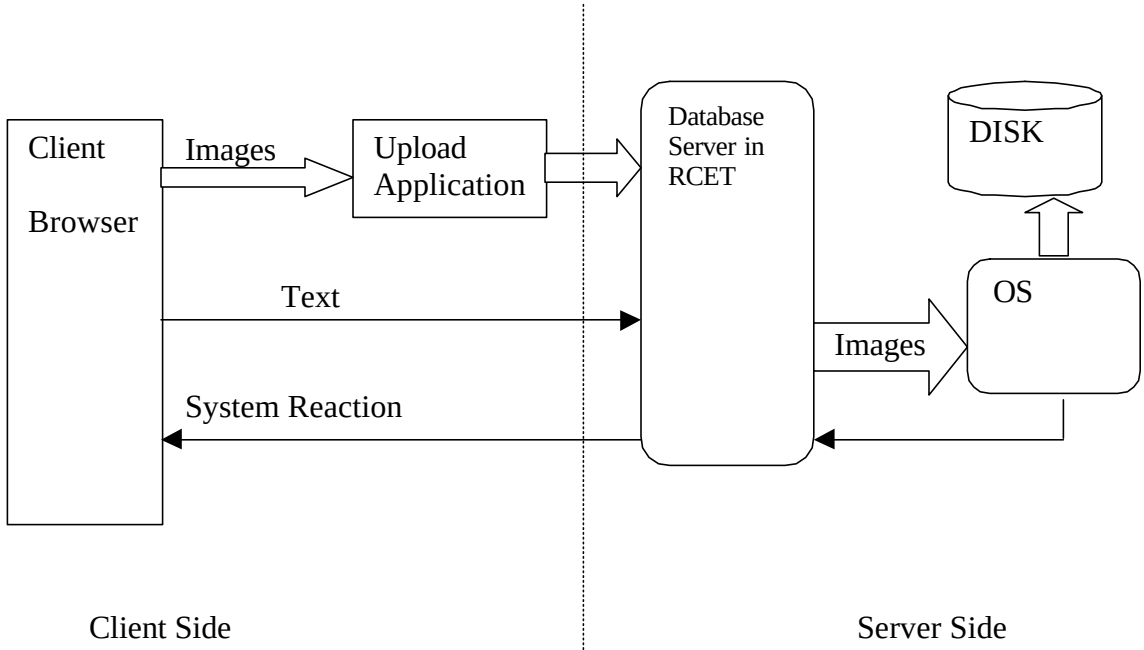


Figure 4.4. The Upload Mechanism

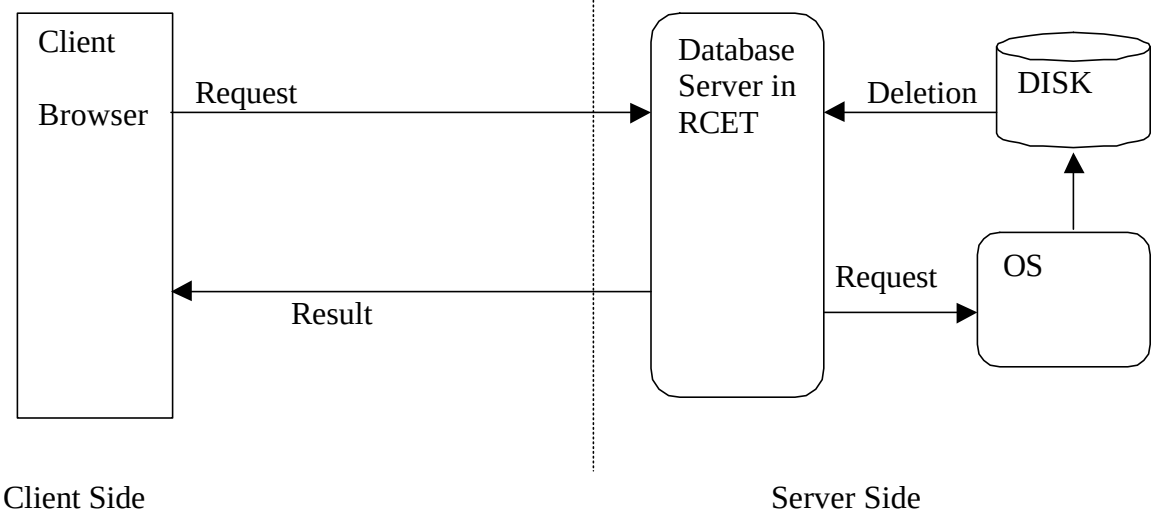


Figure 4.5. The Delete Mechanism

4.4. Design of Web Interface

The most important requirement on the web interface is its consistency with the existing forms (in paper). This will not create confusion and additional training for the operators will not be needed. We designed the layout of the web interface to be as similar to possible as the existing forms (in paper).

The system should have a strong exception handling capability so that the system won't crash very easily. There are many form validations before the forms are submitted to the database. If an error is detected, a warning message will pop up with the suggestion for right data. After the user click OK on the warning message, the cursor will be moved to where the error is.

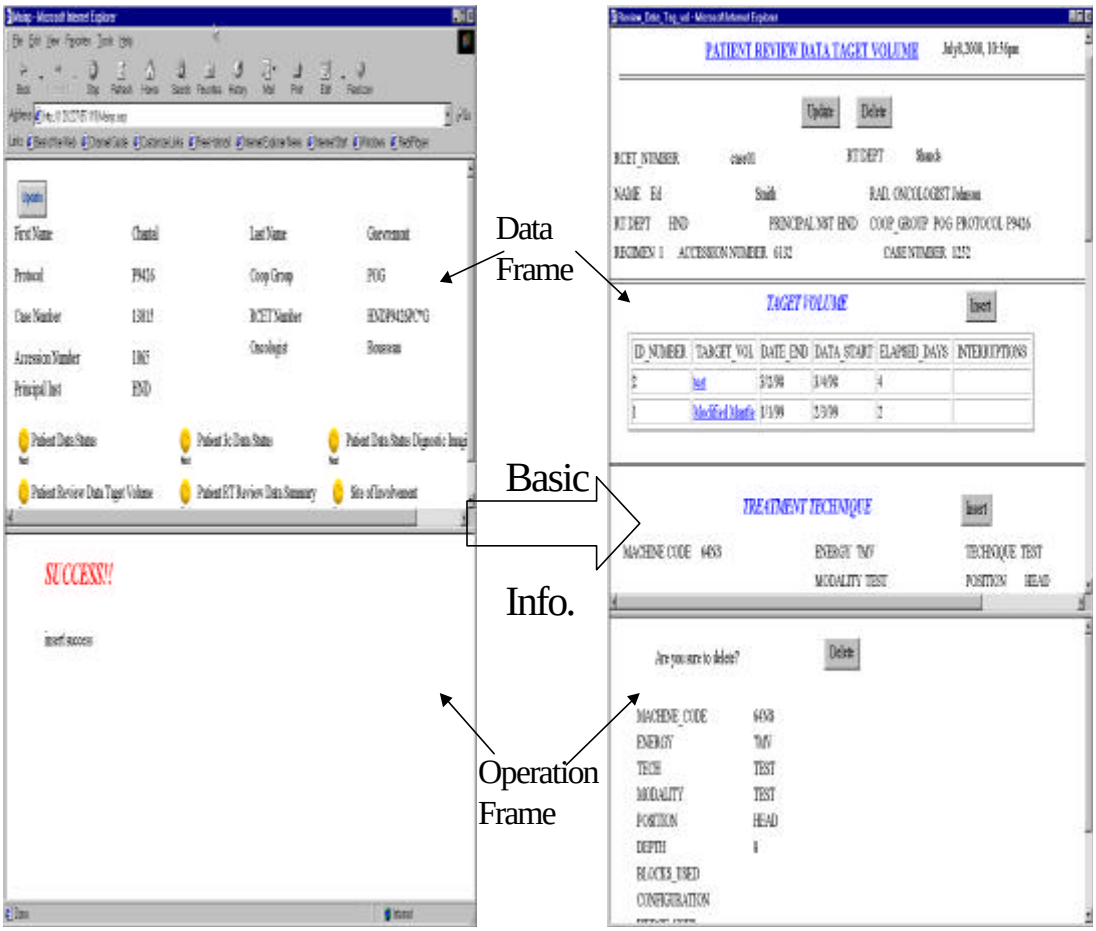
Since the system will be used in clinic and people are prone to make mistakes in the clinic environment, the system should help lead the user to input correct data. This is one of the main targets of this project. The system provides some directions for the data validation on the interface for the input field (e.g., put MM/DD/YYYY after a date input) to direct the user to put in the right data. We use radio buttons for binary input and drop-drown list if we know the choice to fill in. We also limit the length of an edit box if the maximum length of input is known.

We use check boxes, text edit box, buttons etc. as the fundamental components to built the web interface. The system enables and disables buttons according to the medical protocol and database requirement to enable/disable some operations to the user. The system disables the browser's tool buttons to prevent user from jumping between pages. By doing so, the system prevent user jumping between transactions. We also use buttons instead of hyperlinks because the latter would expose to a hacker which file will be invoked. Hyperlinks constitute a security vulnerability.

When a user want to enter the RCET system, the login page is the first page he needs to fill in. After successfully logging in, the basic information page will be displayed. The user is asked to insert the necessary information in the basic information page. After inserting data in the basic information page, the user can go to the following

pages (Patient Data Status page, Patient 3c Data Status page, Patient Data Status Diagnostic Imaging Inventory page, Patient Review Data Target Volume page, Patient RT Review Data Summary page and Site of Involvement page) through links on the basic information page. When the link is clicked, another window is popped up with the following page. If the data according to the RCET number already exists in the database, all the following page will be populated by the data from the database. If the data does not exists in the database, all the information inserted in the base information page will be carried over to the following pages.

The page is divided into two frames. The upper frame is the data frame, which displays the data use queried in the database. All the operations can be carried out on the data are displayed by the buttons in the upper frame. The lower part is the operation frame, in which user can do all the operations to the data. Also reactions from the system are displayed there. For example, initially the upper part displays the data and the lower part is blank. During a data insertion operation, the user first pushes the INSERT button in the upper frame. The edit boxes for insertion are then shown in the lower part. After the user inserts the data, the user pushes the INSERT button in the lower frame. Upon successful transaction, an insert success page will be displayed in the lower part and the data just inserted will be displayed in the upper part. The organization of web pages is shown in the figure 4.6.



Basic Information Page Following Pages

Figure 4.6. Organizations of Web pages

4.5. Design of Visualization Tools

Since most of the medical information is in the form of images and 3D objects, the user must have tools for visualizing, manipulating and sorting images and components of a treatment plan as well as being able to interacting with RCET database.

In the traditional Web-enabled visualization environment, the client is effectively reduced to a viewer of information supported by a server. A true Web client is not capable of program execution unless the executables are downloaded to the client as Java applets. This client is normally referred to as the "thin" client. A thin client, by definition,

has minimal software requirements necessary to function as a user interface front-end for a Web enabled server application. Local visual data analysis, information drill-down technique, object picking and other interactive user interface functions that traditionally have been available on the client are now being exercised by the visualization engine. In this "thin" client model, nearly all functionality is delivered from the server side by the visualization engine while the client perform very simple display and querying GUI functions based on HTML forms.

However, in the "fat" client scenario, the entire visualization process takes place at the client-side. The user performs data manipulation locally, selecting the dimension mapping and visualization paradigm. The solution is to move the data rendering process (turning your numerical data into geometry) from the server-side to the client-side. Any of the following techniques are applicable: "Plug-ins", "Helper Applications", Java applets, ActiveX controls or JavaBeans.

In data visualization analysis context of the system, it is clear that visualization tools need to be more than simply presentation vehicles. They need to be tightly integrated in to the "data navigation paradigm". The notion of slice and drilldown through data space need to be mapped into the data visualization navigation concept. Therefore, the concept of a "thin" client raises the issue, where must the data visualization take place to provide maximum data exploitation.

The Java applets that deliver locally available executables are, however, still dependent on the network bandwidth. Depending on the scope and application, applets and its data sets must be downloaded every time they are being used. Java applets are only resident during execution and are therefore removed from the local disk after the

completion of the task. As the demand for larger applets and data sets grows, significant download time could be incurred and the network becomes the bottleneck. Keeping commonly used applets resident on the client would significant reduce download time, although this practice is counter to the Java applet architecture. For our applications, Java technology is not adequate for rendering very complex medical scenes.

Web component technology based on Microsoft's ActiveX architecture provide a more efficient tool visualization. Visualization ActiveX control can be used to let the user (Client) read a script language, which controls the visualization type, attributes and the data to be visualized. The Web browser knows about the Visualization ActiveX control, and will automatically launch it and load the Plug-in into it once the data transfer finishes. The Visualization ActiveX control will perform the data manipulation and rendering locally at the client side. When the ActiveX control resides on the user’s computer it will not need to be downloaded again.

The size of our ActiveX visualization components is about 3Mb, but it needs to be installed only once on each client's system. When the embedded with such ActiveX document is transferred over the Internet, it will only refer to the component and thus the total size of the document remain small (30Kb). Any project can therefore exchange Word97 documents with embedded advanced 3D visualization over the Internet using pre-installed ActiveX components at the client side. Local stored data is accessed by a "data reader" in the component or transferred over the Internet using the "URL reader".

4.6. Summary

This chapter began with a brief outline the system architecture of the RCET_Sub system. We then focused on each component of the system and discussed the design of them in detail.

With the design in mind, we will discuss the implement of the system in the next chapter.

CHAPTER 5
SYSTEM IMPLEMENTATION

In this chapter, we only present the implementation of the database and web interface parts of the system. The visualization and security parts are under development. We satisfy each software requirement with the most efficient programming technique.

5.1. Implementation of the Replication

From figure 4.2, we know this project involves setting up database replication between external database1 and RCET, external database2 and RCET. To complicate matters, the databases in different are from different vendors or are different versions. So the database part of the RCET_Sub system is a distributed heterogeneous database. There are two steps to implement replication: configure the network and database link, set up snapshots and triggers.

5.1.1. Configure the Network and Database Link

To implement replication, we first need to configure the system and set up the database link. Between Oracle servers, we used Net8. After we installed and verified the setup of Oracle Networking Products, we configured a network by creating the client configuration files: TNSNAMES.ORA and SQLNET.ORA.

TNSNAMES.ORA contains the names and addresses of services on the network. This configuration file is used by both clients and distributed servers to identify destinations servers.

SQLNET.ORA used by all clients and servers on the network. The file contains client profile information including optional diagnostic parameters, client information about Oracle Names, and other optional parameters such as native naming and security.

In the next step, we need to create a database link that establishes connections to the corresponding schema at the master site. The following SQL command script completes the database link setup.

```
CREATE DATABASE LINK dbs1 CONNECT TO scott IDENTIFIED
     BY tiger;
```

After configuring files and setting up database link, we can query the table in the master site by using table name@database link as if it were a local table (e.g., `Select * from table@dbs1;`).

For Microsoft Access, we needed to setup the ODBC data source name (DSN). An ODBC system DSN stores information about how to connect to the indicated data provider. A system data source is visible to all users on the machine. In configuring DSN, we choose a DSN that specifies the location and type (ODBC driver) of the database you want to access. Using the ODBC driver provided by Oracle and Microsoft, we can access the Oracle data in Microsoft Access.

Once the data link is set up, thanks to location transparency in Oracle and ODBC, the user can query all databases as if they were local.

5.1.2. Setup Snapshot and Trigger

Before creating snapshots, create the master site snapshot logs that are needed to support fast refreshes for the snapshots. Once the necessary snapshot logs are in place, we can create the snapshots. After creating the snapshots, we make sure to assign all related snapshots to a refresh group that the snapshots will use to refresh.

The following SQL command script demonstrates how to create snapshot logs and snapshots at the master site to support the snapshots. It also creates and schedules the refresh group REFGRP1 at the snapshot site and assignes to it the new TABLE1 snapshots.

```
CREATE SNAPSHOT LOG ON table1;
CREATE SNAPSHOT table1 AS SELECT * FROM table1@dbs1;
DBMS_REFRESH.MAKE(name=>'refgrp1',  list  =>  'table1',
     next_date     =>     SYSDATE,     interval     =>
     'SYSDATE+1/24');
 COMMIT;
```

Using the snapshot, the changes at the master site will be available at the slave site only after refresh. In this RCET_Sub system, we divided the master into groups based on the requirements of the particular application. The tables don't need to be synchronized very frequently. We made snapshots on them and scheduled the refresh time at night to avoid the peak usage times.

Using trigger approach to maintain replication, we need to set a trigger on every database operation. In other word, insert, delete or update of the data at the master site will trigger a corresponding event. The following SQL command script shows how to use the trigger to replicate data in Oracle.

```
CREATE TRIGGER trig1
     BEFORE INSERT OR DELETE OR UPDATE OF ON table1
     FOR EACH ROW
BEGIN
     IF INSERTING THEN
          INSERT INTO table1@dbs1
          VALUE (:new.col1, …..)
          WHERE id=:old.id;
     END IF;
     IF UPDATING THEN
          UPDATE table1@dbs1
               SET col1 = :new.col1,
               …..
               WHERE id=:old.id;
     END IF;
     IF DELETING THEN
          DELETE FROM table1@dbs1
          WHERE id=:old.id;
     END IF;
END;
```

For data replication between Microsoft Access and Oracle, the Microsoft Access only support the trigger mechanism. In Microsoft Access, we create a trigger event procedure when data changes on the master table. The trigger event procedure is written

in Microsoft Visual Basic and makes use of the ODBC link. Below is the Visual Basic code for the insert event.

```
     Private Sub Form_AfterInsert()
          Set db = OpenDatabase("", False, False, ODBC;
uid=user1; pwd=pwd1; DSN=odbclk")
          'strSQL = "insert into tab1 values(col1,…);"
          db.SQLExecute "insert into ex11 values(col1,…)"
     End Sub
```

For the other database operations, we can use corresponding a trigger event procedure to catch them in similar fashion. In the RCET system, if the tables need to be synchronized with the master table on every transaction, we use triggers to maintain replication.

5.2 Implementation of the Database

To relate the base table with the extent table, we use the TableId as the foreign key to the extent table. It's unique in the base table to point to the extent table. To ensure the uniqueness of the TableId through the database, we create a sequence so that the database server can automatically generate unique values throughout the database. The following statement creates a sequence used to generate unique number for the TableId column of the extent tables:

```
CREATE SEQUENCE seq1
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOCYCLE
```

```
CACHE 10;
```

Every insert operation in the base table will lead to a trigger that increases the value of the sequence. Then it will insert the sequence value into the base table. Below is the code of the trigger.

```
create or replace trigger insert_base
before insert on base_table1 for each row
declare  id number;
     begin
          select seq1.nextval into id from dual;
          :new.TableId :=id;
          …
     end;
```

An insert operation in the extent table will fire a trigger that inserts the current sequence number into the extern table (as show in the following code).

```
create or replace trigger insert_ex
after insert on ex_table1 for each row
declare id number;
begin
     select seq1.currval into id from dual;
     insert into table1 values(id,…);
end;
```

For the delete operation, we should consider the dependence of the extent table on the base table. We enabled the delete cascade so that deletion of a row in the base will lead to deletion of the extent table in a cascade fashion.

For images in the database, because of the reason stated in design portion of the thesis, we don't choose the LOBs technology of Oracle to manage images. Instead, we add a column ImageAddress that stores a relative pointer to the image. The relative pointer is composed of the RCET number, Case number and file name (e.g. /RCET number/Case number/Image1.jpg). When the page is loaded, the page will automatically generate the address of the image according to the virtual directory structure of the web server.

When inserting the data, the user first uploads the image files. Then the operation fires a trigger that does the following things.

- Generate the relative pointer to the image according to the RCET number, Case number and file name structure.

- Copy the image file from the temporary directory to the directory decided by the virtual directory structure of the web server and the relative pointer. The pl/sql query

  ```
  Host(copy temp/file1 address/file1)
  ```

  will call the operation system to copy the file to the destination. After the files are successfully copied, the data and relative pointer will be inserted into the database.

When deleting the data, the operation fires a trigger using the the pl/sql query

```
Host(del address/file1)
```

to call the operating system to delete the file in the directory where the image files resides. After the files are successfully deleted, the record will be deleted from the database.

5.3. Implementation of Web Interface

The web interface pays a very important role in the RCET_Sub system. The interfaces are complicated and we have to deal with many issues. Thanks to Micromedia Drumbeat 2000, we solve many issues by applying interactions (given by drumbeat or programmed) to the web applications. Here we only discuss the most challenging issues.

5.3.1. Common Data

After the user inserts the data in the basic information page, a following page will be displayed by pushing the link. If the data according to the RCET number does not exist in the database, all the information inserted in the base information page will be carried over to all the following pages. We solve this problem by ASP Session in ASP. By using the following ASP sentence:

```
Session("Last_Name")=String(Request("LAST_NAME")),
```

we can make a session Last_Name and make it equal to the value inserted on edit box LAST_NAME. After the session is made, all the other pages can use the session as if it were a constant throughout this login. When common data is necessary to be displayed in the following page, we can just make it equal to the session. This way the user does not need to type them again.

The session can only be changed or deleted in the basic information page. So that all the common data in the insert or update pages of the following pages are not

editable, all the data are displayed in a text box instead of an edit box. This will prevent the user from accidentally changing the common data.

5.3.2.Interaction Between Frames

In the design of the layout of the page, the page is divided into two frames. The upper frame is the data frame, which displays the data use queried in the database. All the operations that can be carried out on the data are displayed by the buttons in the upper frame. The lower part is the operation frame, in which user can do all the operations to data. The reactions from the system are all displayed there. The data frame is dynamically generated once the transaction is successful. Meanwhile an "operation success" page will be display in the operation frame. To implement this, we programmed the ASP code so that if the transaction is successful, the "operation success" page will be loaded in the operation frame. At the beginning of the success page, we added the following code that refreshes the data frame.

```
parent.Main.document.location.reload(true);
```

By doing so, the new data will be populated in the data frame.

5.3.3. Display of Image Database

To display the image from the database, we first define the global variable of the site that point to the virtual address of the image, so that all the pages in the site can use this variable. An example code setting the address will be.

```
Var base_address=”http://18.227.85.214/images/”;
```

Then we put the RCET Number, Case number and relative image pointer from the current record into local variables.

```
Var Rcet_address=Recordset1.GetColumnValue(“RCET_Number”);
Var Case_address=Recordset1.GetColumnValue(“Case_Number”);
```

```
Var image_address=Recordset1.GetColumnValue("imagep");
```

The address to this image can be generated by:

```
Var address= base_address+"/"+ Rcet_address+"/"+
image_address;
```

Then at the place to display this image, we put the address variable as source address:

```
SRC="<%=(address)%>"
```

With image displayed, user can perform the database operations on them.

5.4. Summary

We have discussed how we implemented database replication, database and web user interface in detail. In the next chapter, we will test the functionality of the RCET_Sub system.

CHAPTER 6
SYSTEM TESTING

Software testing is one of crucial steps in software development. It is responsible for evaluating a system or system component by manual or automated means, to verify that the software satisfies specified requirements or to identify differences between expected and actual results.

6.1 Review of Basic Testing Concepts and Techniques

Software testing is the process of executing a program with the intent of finding errors [14]. The essence of software testing is to determine a set of test cases for the item being tested. The most obvious information in a test case is inputs; inputs are really of two types: pre-conditions (circumstances that hold prior to test case execution) and the actual inputs that were identified by testing methods. The next most obvious part of a test case is the expected outputs; again, there are two types: post conditions and actual outputs [15].

The act of testing entails establishing the necessary pre-conditions, providing the test case inputs, observing the outputs, and then comparing these with the expected outputs to determine whether or not the software passed the test [15]. There are three fundamental techniques for identifying test cases: these are known as Black Box (functional) Testing, White Box (structural) Testing and Grey Box Testing. Black Box Testing, also called data-driven or input/output-driven testing, is based solely on examination of the requirements, specifications, or user documentation while White Box

Testing is based on knowledge of the code logic and implementation. Grey Box Testing is a technique based on a limited knowledge of the code logic, or an implementation that alters a black box test.

6.2. Test Plans

Creating a test plan is the first step for software testing. Specifications are very important for making test plans. Here we will strictly follow the specifications of RCET_Sub system discussed in chapter 3 for making our test plan. We outline the testing plan as follows.

1. Database operation. The web interface should enable the user to perform insert, delete and update on the tables in database.
2. Data integrity and consistency checking. A warning message should be given and prevent those operations that violate data integrity and consistency.
3. System "remember" function testing. Once a common data are inputted, the system should never "forget" them until the user logs out.
4. Interaction between frames. When some transactions are performed in the operation frame, the data should be refreshed in the data frame and the page need to be redirected in the operation frame.
5. Friendliness and convenience of user interface should be evaluated.
6. Data replication. When the data is changed in the master table, the change should be available in the slave table.
7. Image Database. The system should enable the user display, insert, and delete images from the image database.

Based on the above test plan, we designed the following test cases.

6.3 Testing Design and Testing Results

Software testing techniques have developed quickly with the rapid growth of the software industry. Many testing techniques have been developed and used widely in the software industry. However, among them, the traditional Black Box Testing technique is still the most robust and efficient method. We used the Black Box Testing technique to design our test cases.

Case 1:

Purposes: 1. Test insert data operation.

2. Check the system reaction.

Test: 1. In the data frame, push the INSERT button

2. In the operation frame, fill in the data in the edit boxes and push the INSERT button.

Expected Results: Data should be inserted into the database.

Testing Results: 1. In the data frame, new data was filled in.

2. In the operation frame, an operation successful page was shown.

Case 2:

Purposes: 1. Test update data operation.

2. Check the system reaction.

Test: 1. In the data frame, push the UPDATE button

2. In the operation frame, update the data in the corresponding edit boxes and push the UPDATE button.

Expected Results: Data should be updated in the database.

Testing Results: 1. In the data frame, new data was filled in.

2. In the operation frame, an operation successful page was shown.

Case 3:

Purposes: 1. Test delete data operation.

2. Check the system reaction.

Test: 1. In the data frame, push the DELETE button

2. In the operation frame, push the DELETE button for conformation.

Expected Results: Data should be deleted in the database.

Testing Results: 1. A confirmation page was displayed in the operation frame.

2. After pushing OK in the confirmation, all data was gone from the data frame and an operation successful page was shown in the operation frame.

Case 4:

Purposes: Check the system reaction when inserting data in the upper part of the page.

Test: 1. In the data frame, push the INSERT button in the upper part.

2. In the operation frame, fill in the data in the edit boxes and push the INSERT button.

Expected Results: 1.Data should be inserted into the database.

2. In the data frame, the INSERT button should be disabled to prevent multiple inserts.

3. The INSERT button in the lower part page in the data frame should be shown to enable the user to perform insert to extent tables.

Testing Results: 1. In the operation frame, an operation successful page was shown.

2. In the data frame, new data was filled in.

3. In the data frame, the INSERT button was disabled.

4. The INSERT button in the lower part page in the data frame was shown.

Case 5:

Purposes: Check the system reaction when deleting data in the upper part of the page.

Test: 1. In the data frame, push the DELETE button in the upper part.

2. In the operation frame, push the DELETE button for confirmation.

Expected Results: 1.That row of data in the base table should be deleted and the extent tables should be deleted.

2. In the data frame, all data in both the upper and below parts should be gone.

Testing Results: 1. A confirmation page was displayed in the operation frame.

2. After pushing OK in the confirmation, all data in both the upper and lower parts were gone in the data frame and an operation successful page was shown in the operation frame.

3. All buttons except the INSERT button of the upper part were invisible.

Case 6:

Purposes: Test whether the system can generate pages from the data existing in the database.

Test: 1. Input a RCET Number that already exists in database.

2. Push the link to go to the following pages.

Expected Results: The following pages should be filled with the data from the database.

Testing Results: All the following pages were generated with the data from the database.

Case 7:

Purposes: Check if the system “remembers” the common data.

Test: 1. Insert the information in the basic information page.

2. Go to the following pages using the links on the basic information page.

Expected Results: The following pages should be filled with the data inserted in the basic information page.

Testing Results: All the following pages were generated with the common data inserted in the basic information page.

Case 8:

Purposes: Test form validation before submit.

Test: 1. Input a number in a date field and push INSERT button.

Expected Results: The error message should be shown and the form should not be submitted.

Testing Results: 1. A warning message popped up saying “the form can not be submitted. Date1 please enter a date.”

2. After pushing OK, the page was not submitted and the cursor was positioned at the problem location.

Case 8a:

Purposes: Test form validation before submit.

Test: 1. Push INSERT button with all the edit box blank.

Expected Results: The error message should be shown and the form should not be submitted.

Testing Results: 1. A warning message popped up saying “the form can not be submitted. Edixbox1 can not be blank.”

2. After pushing OK, the page was not submitted and the cursor was positioned at the editbox1.

Case 9:

Purposes: Test replication between external database1 and RCET.

Test: 1. Insert data in a table in external database1 and commit.

Expected Results: The data should be available in the RCET database.

Testing Results: The data just inserted was in the RCET database.

Case 10:

Purposes: Test the replication between external database2 and RCET.

Test: 1. Insert data in a table in external database2 using Microsoft Access.

Expected Results: The data should be available in the RCET database.

Testing Results: The data just inserted was in the RCET database.

Case 11:

Purposes: Test the search and display of image database.

Test: 1. Choose the RCET Number from the drop-down list and push FIND.

Expected Results: The images with this RCET number should be shown in the page.

Testing Results: The images with this RCET number were displayed. Clicking on the image caused the text and image data of the case to be shown.

Case 12:

Purposes: Test the deletion from the image database.

Test: 1. In the image page push the DELETE button.

Expected Results: The data should be deleted from the database and the image file should be deleted from the disk.

Testing Results: 1. A confirmation message appeared.

2. After pushing the OK button in the confirmation message, the data was deleted from the database and the image file was deleted from the disk.

Case 12:

Purposes: Test the insertion from the image database.

Test: 1. In the image page, choose the image to upload then push the INSERT button.

Expected Results: The data should be inserted from the database and the image file should be moved to a directory according to the RCET number.

Testing Results: 1.After pushing the INSERT button, the data and image were inserted in the database

2.The image files were moved to a directory according to the RCET number.

6.4 Testing Analysis and Final Report

The above test cases are designed to verify whether the system satisfies the requirements discussed in chapter 3. Table 6.1. lists the requirements and test case(s) that show the satisfaction to the requirement. Based on the above analysis, we conclude that the system meets the software requirements and solves proposed problems. It is a practical and stable web database system with high performance. It is user-friendly and easy to use as well.

Table 6.1. Requirement vs. Test Case Table

Requirement	Requirement Area				Satisfaction shown by test case(s)
	Database Schema	Image Database	Database Replication	Interface	
1: Database Operations	X				Case1,2,3
2:Uniqueness of RCET number	X				Case 4,6
3:Generation of pages according to RCET number	X				Case 6
4:relationship between base table and extent table	X				Case 5
5:Relationship between upper part page and lower part page	X				Case 4,5
6:Deletion cascade in upper part page	X				Case 5
7:Image can be managed using database.		X			Case 11, 12,13
8:Insert image		X			Case 13
9:Delete image		X			Case 12
10:Replication between external database1 and RCET database			X		Case 9
11:Replication between external database2 and RCET database			X		Case 10
12:Interface's consistence with the existing forms				X	Case 1-13
13:The system should give reaction page.				X	Case 1,2,3,4,5,
14:The system should enable/disable button to enable/disable operation				X	Case 4,5
15:Data frame is filled with the most recent data				X	Case 1,2,3,4,5
16:The common data can only be operated on the basic info. page				X	Case 7
17:The system should "remember" the common data				X	Case 7
18:Form verification				X	Case 8, 8a
19:Blank page can not be submitted				X	Case 8a
20:Confirmation page before deletion				X	Case 3,5
21:Optimization of the interface				X	Case 1-13

CHAPTER 7
CONCLUSION AND FUTURE WORK

The goal of RCET_Sub system is to create an environment so that the users can input, manipulate, and manage medical information through the Internet. The target users are the medical professionals in the participating institutions. The benefits the system can provide to the participating institutions are time saving, information sharing and efficient management.

With the security and efficiency considerations of distributed databases, we set up database replication between external database1 and RCET database, external database2 and RCET database. Thanks to ODBC and Net8, the database links were built between Oracle and Microsoft Access, Oracle (7/8i) and Oracle (8/8i). Using snapshot and trigger mechanisms of replication, all the changes in external database1 and external database2 will be available in the RCET database.

Since the system is designed for non-programmers in a clinical environment, we tried very hard to satisfy user friendless and error rejection needs. After every successful database transaction, the operation success page and a page with new data will be shown, so that the user sees the result of every effective transaction in the database. The system helps the user to input the right data by giving out directions in the interface and providing suggestions for correct data in a pop-up window when an error happens. To ensure that the user performs the right operations according to the medical protocol, the system enables and disables the buttons in the interface according to the data in the database.

The image database is another important feature of the system. The system can dynamically generate web pages with the data and images from the database. It provides the user with facilities for inserting and deleting image information from database in conjunction with the appropriate operating system operation on the affected image files.

System testing results prove that the system meets the software requirements and solves the proposed problems. It is a practical and stable web database system with high performance. It is user-friendly and easy to use.

The goals for developing RCET_Sub system were achieved. However, there are several parts that need to be developed in the future:

- Because the system is now in the testing step, we did not put any security mechanisms in it. For medical information, strict security measures are required by law.

- Now RCET_Sub system only allows user to insert and deleted image files. In the future, we will develop visualization tools so that user can generate image and manipulate images.

LIST OF REFERENCES

[1] Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, Database Systems: Concepts, Languages and Architectures, McGraw Hill, New York, NY, 1999

[2] What Is JDBC?, http://java.sun.com/docs/books/jdbc/intro.html, March 1997

[3] Net8 Getting Started, Release 8.0.4 for Windows NT and Windows 95, Oracle Corporation, Redwood Shores, CA, Dec. 1999

[4] Oracle8 Replication, Release 8.0, Oracle Corporation, Redwood Shores, CA, Dec. 1999

[5] Oracle8 Application Developer's Guide, Release 8.0, Oracle Corporation, Redwood Shores, CA, Dec. 1997

[6] Microsoft Active Server Pages: Frequently Asked Questions, http://msdn.microsoft.com/library/backgrnd/html/msdn_aspfaq.htm, Sept. 1997.

[7] Macromedia Drumbeat 2000 Active Server Pages Version Users Guide, Macromedia Inc, San Francisco, CA, 1999

[8] Managing Sessions, http://msdn.microsoft.com/library/psdk/iisref/iiapsess.htm, Jan. 2000

[9] Randy Chow, Theodore Johnson, Distribute Operating Systems and Algorithms, Addison-Wesley, Reading, MA, 1997

[10] SSL Handshake, http://www.ece.uc.edu/~iouaiss/web_prog/misc/security/H_2.htm, 1997

[11] DICOM in Radiotherapy , http://www.sgsmp.ch/bull983b.htm, 2000

[12] DCOM Architecture, http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomarch.htm, July 1997

[13] ActiveX Controls, http://www.microsoft.com/com/tech/activex.asp?RLD=18, Jan. 2000

[14] Glenford J. Myers, The Art of Software Testing, John Wiley & Son, inc., New York, NY, 1979.

[15] Paul C. Jorgensen, Software Testing, A Craftsman's Approach, CRC Press, Boca Raton, FL, 1995.

BIOGRAPHICAL SKETCH

Wei Zheng was born in Guiyang, Guizhou Province, China. He received his bachelor's degree in physics from Wuhan University, in 1996. He received his master's degree in 1998 from the University of Miami. In the spring of 1999, he stated his graduate studies for a major in computer and information science and engineering at the University of Florida. He will receive his Master of Science degree in computer engineering from the University of Florida in August 2000. His research interests include database system, computer network and Internet development.