Spring 1998

Development of a framework for internet based education system

Srinivas Lanka
New Jersey Institute of Technology

ABSTRACT

DEVELOPMENT OF A FRAMEWORK FOR INTERNET BASED EDUCATION

Development of a framework for Internet based education has demonstrated the use of Oracle tools for the use of delivery of education on the World Wide Web. This also has proved that for an efficient dynamic education scenario, the use of a database-based system with a proper retrieval system is required.

We have designed the system on an Oracle backend system with Designer/2000. Developer/2000 helped us with the design and development of the system. Oracle Web Server 2.1 helped us with the retrieval of the web pages.

The entire design of the system and the reasoning behind the system has been documented. Appendix A provides a glossary of currently used terminology in the field of Internet based systems. The Appendix B provides actual screen prints of the Designer/2000 phases of design, Developer/2000 graphical user interface screens and the actual code involved in the design and development of the system. The fact that the entire system is based on the Oracle Repository makes the system very dynamic in terms of the data and can be used to present the student with a course material rapidly.

DEVELOPMENT OF A FRAMEWORK FOR INTERNET BASED EDUCATION SYSTEM

by
Srinivas Lanka

A Thesis
Submitted to the Faculty of
New Jersey Institute Of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer and Information Science

Department of Computer and Information Science

May 1998

APPROVAL PAGE

DEVELOPMENT OF A FRAMEWORK FOR INTERNET BASED EDUCATION SYSTEM

Srinivas Lanka

5/7/98

Dr Murat Tanik, Thesis Adviser — Date
Associate Professor of Computer and Information Sciences, NJIT

5-13-98

Dr Franz Kurfess, Committee Member — Date
Assistant Professor of Computer and Information Sciences, NJIT

5/12/98

Dr Ali H. Dogru, Committee Member — Date
Visiting Associate Professor of Computer and Information Sciences, NJIT

BIOGRAPHICAL SKETCH

Author:	Srinivas Lanka
Degree:	Master of Science
Date:	May 1998

Undergraduate Education:

- Master of Science in Computer and Information Science,
  New Jersey Institute of Technology, Newark, NJ, 1998

- Bachelor of Engineering in Electronics and Communications,
  Andhra University, Vishakapatnam, India, 1985

Presentations and Publications:

Srinivas Lanka, S.H.Agarwal, D.N. Murmu, N.N. Shitut
"Multitasking in Microprocessor Based systems in Power Plants"
National Symposium of Computer Applications in Power Plants,
Bhabha Atomic Research Center, Bombay, December 1991.

To my beloved family

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to Dr Murat M. Tanik who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance.

I would also like to thank Professor Leon Jololian who provided invaluable technical help and the software resources.

Special thanks are due to Dr Franz Kurfess and Dr. Ali Dogru for actively participating in my committee.

I also wish to thank Ms. Annette Damiano for her assistance in verifying that my document format was as per guideline.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

2.2 Virtual Teaching Mechanisms

Virtual Teaching Mechanisms to deliver courses are as follows:

Instructional Television Fixed Service

Microwave

Cable and Public Television

Satellite

Videotape

CD-ROM

Internet

These techniques may be used to impart education over different geographical regions like local, statewide, countrywide or including other countries.

For *local regions* Instructional Television Fixed service could be used. This would mean connecting different sites like companies and local schools with this system and they would be fed in the programs from one or many local sources. Cable and Public television also can be used to server the local region. This could enable students from homes and wherever a television is available to get the educational program.

For *statewide regions*, the microwave is one of the options that can be used. This can enable quality audio and video to reach students across the length and breadth of the state.

For *national and international regions*, U. S. satellites or foreign satellites can be used to uplink educational programs. This could be a big network containing a number of

CHAPTER 1

INTRODUCTION

The powerful delivery mechanism of the World Wide Web (WWW) is prompting many organizations to carefully consider their application strategies. The Web has immense potential in making information more accessible, more dynamically customizable and visually very appealing to the user. As a result, the number of software applications that are web based is increasing day by day. This introduces challenges on the tools that would be needed to develop these applications. These tools, should if possible have a short learning curve, be able to build and deploy applications rapidly, provide a wide variety of features that are currently available in the market for web development, and finally should be economical. This section describes the types of web applications that users are trying to build, reviews the benefits and briefly explains Oracle's tools for the internet. This section also provides the introduction to the project developed for this thesis.

1.0 The Evolution of Web Applications

Several types of web-based applications have evolved with user and organizational demands:

1. Static web pages

These web applications consist of static web pages. These are very popular on the Internet today and can be easily created by the average user. It is possible to integrate different kinds of media like text, images, audio and video into these pages. The user can go from one page to another by what are called as links. Due to the pages being static the pages need to be modified whenever there is a change of information. Visual appeal of the Web page and attractive content holds the attention of the user.

2. Dynamic web publishing applications

Since the user needs up to date information, more so in a corporate environment, this type of applications came into the market. These applications have a database containing all the information that needs to be displayed and the web pages display the data to the user on demand. The database is continuously updated and the user always sees the latest information in the database. Changing data that requires modifications to the Web pages are avoided here.

3. Transaction-oriented applications

Every organization will have some type of client-server or legacy application. The transaction-oriented application are developed to coexist with the current application or in some cases sunset the old application. The functionality of this new application is the same as the previously existing application, except that now the application is available to a wider audience at very small or negligible cost.

4. Electronic commerce

The World Wide Web brings about a much closer interaction between organizations, customers and suppliers. Users can connect to organizations and get the information they require without any special hardware or software setup. Customers can directly connect to the company without the need of any sales persons. Suppliers can monitor the levels of the companies they are supplying to and regulate their supply based on the specified levels.

1.1 Benefits of the Web

Some of the benefits of the Web applications are as follows:

1. Reduction in Administrative Costs

It is generally said that the side affect of client-server systems is increased administrative costs. Many applications had software installed on the client computer, generally a personal computer, and these must be compatible with other existing applications. Any modification in the client computer software meant going to each client computer and upgrading this computer with the new software. This compounds the condition as more and more client computers are introduced. Hence the increased administrative cost.

Since Web applications are server based and have a very thin client, a browser, this greatly reduces the time and expense that would be involved in going to each client computer to upgrade the client software piece.

2. A Reduction in Resource Requirements

Generally the client-server applications require a sizeable amount of memory to execute. This would be over and above the memory needed for the execution of the operating system and the windowing environment. The disk space also becomes a constraint under these circumstances.

Web applications generally have lower resource requirements. The application actually executes on the application server or the web server and as a result the memory requirement is less. The application also resides on the application server and hence the disk space constraint is also addressed.

3. Application Portability

Deploying client-server applications on more than one windowing platform has been a challenge in most cases. This is primarily due to the peculiarities of the operating systems, the windowing software, as well as the operating environment. Organizations have had to modify applications in order to make them available on multiple platforms. Maintaining these applications is a nightmare.

Web applications execute under the browser and are machine or platform independent. The browser presents the application to the user and browsers are today available for nearly every platform. Hypertext Markup Language(HTML) and Java are also machine independent and hence the user will see the same form of the application irrespective of what platform he/she is on.

1.2 Oracle Tools for Information Publishing

Oracle offers two development tools for building and deploying publishing applications on the web: Designer/2000 and Developer/2000. Developer/2000 is a visual development environment for building database-oriented applications. Designer/2000 is a business modeling and design tool from which applications can be generated, including Developer/2000 applications.

Designer/2000

Designer/2000 enables developers to create visual representations of their applications -- database and application module diagrammers are used to define both server-side and client-side application definitions and logic. These definitions, together with templates and user-defined standards, form the basis for the generation of complete applications. Designer/2000 makes use of the Oracle WebServer, an HTTP server tightly integrated with Oracle7, to create web applications. The Oracle WebServer retrieves data from the database and formats it into HTML pages using the PL/SQL language, Oracle's standard extension to SQL.

The process of creating a web application using Designer/2000 is as follows (assuming a database has already been defined):

A module is created, defining which database objects to use, how to use them, and how different objects are related to one another. Links are created between multiple modules to enable navigation through multiple screens (for example, from master to detail information). The module definitions are used to generate PL/SQL packages that are then installed in the Oracle WebServer. The PL/SQL packages, are used to generate dynamic HTML pages which incorporate the database access set up for that module and the layout and format defined using the templates and standards in Designer/2000. The HTML pages can be viewed using any web browser.

Web applications can also be generated from previously defined modules in the Designer/2000 repository, or even from existing Developer/2000 or Oracle Forms applications that have been reverse engineered into Designer/2000.

Developer/2000

Developer/2000 is Oracle's graphical user interface (GUI) tool. This tool can be used to create forms, reports and graphics. These objects can be viewed from the World Wide Web when used in conjunction with Oracle Web Server. The slick visual interface along with the powerful features encompassed into Developer/2000 make it a powerful tool for development if user interface forms, whether Client-Server type or web type. Once the database is in place, the user can develop the navigation forms as well as maintenance forms to achieve the result.

1.3 Project Introduction

The development for this thesis was done at NJIT using IBM compatible PC and Oracle Tools under the invaluable guidance of Dr M. M. Tanik and Professor L. K. Jololian. The following topics are covered by this thesis:

1. Education and Internet
2. Oracle Tools And Internet
3. System Documentation of Framework for Delivery of Internet Based Education.

The documentation including code and screen prints are included in Appendix A. Chapter 4 of this document gives detailed information on the design and development of this system.

CHAPTER 2

VIRTUAL TEACHING AND INTERNET

2.1 Introduction

There are generally two forms of teaching:

1. Teaching with the student actually being present in the class with the professor.

2. Virtual Teaching.

In the first case the student attends the live lectures by the professor at the school. This could typically be about 15 lecture sessions. The student attends the seminars and presentations conducted by the professor. The professor gives the student assignments and it is expected that the student delivers the assignment in the time given by the professor. The student is also expected to attend the exams set by the professor. The professor grades the student based on his overall performance.

In the second case, Virtual Teaching, the student does not attend the live lecture sessions at the school. The delivery mechanism determines how the student gets the education. There is very little or no direct interaction between the professor and the student in this case. The student gets the course material goes through it and does his submissions as stipulated in the course material.

2.2 Virtual Teaching Mechanisms

Virtual Teaching Mechanisms to deliver courses are as follows:

Instructional Television Fixed Service

Microwave

Cable and Public Television

Satellite

Videotape

CD-ROM

Internet

These techniques may be used to impart education over different geographical regions like local, statewide, countrywide or including other countries.

For *local regions* Instructional Television Fixed service could be used. This would mean connecting different sites like companies and local schools with this system and they would be fed in the programs from one or many local sources. Cable and Public television also can be used to server the local region. This could enable students from homes and wherever a television is available to get the educational program.

For *statewide regions*, the microwave is one of the options that can be used. This can enable quality audio and video to reach students across the length and breadth of the state.

For *national and international regions*, U. S. satellites or foreign satellites can be used to uplink educational programs. This could be a big network containing a number of sub systems wherein these programs could be beamed to the satellite from any part or any place of the world to any part or place that you need to display the program.

Broadcast quality videotapes may be created by the school in a popular format like VHS and the student may obtain these from the school. These videotapes will contain lectures, seminars, presentations which comprise the course. The professor along with the video production department who create these may use overhead projectors, computer systems, presentation software to make the course rich in content.

Likewise, the audio department can cut Compact Disk Read Only Memory devices (CD ROMS) which the student can listen to with a CD Player. These will typically contain the professor's lectures, relevant articles in audio form.

The Internet is a relatively new mechanism for delivery of education. The student connects to the school via the internet and gets the course material from the web pages created for the course by the professor/faculty.

2.3 Performance Related Issues

2.3.1 Drawbacks of Live Class Session Delivery Mechanism

1. Heavy Student-Professor interaction.
2. Student has to take the course for the full duration of the course(about 15 class sessions).
3. If student wishes to complete his course earlier, he cannot do so.
4. Professor has to be available for the full duration of the course at the college, especially in the class sessions.

2.3.2 Drawbacks of Virtual Teaching Excluding Web Based Systems

The drawbacks of virtual Teaching excluding web based systems are as follows:

1. The student no longer has the close interaction with the professor but is restricted to the course material and the submissions.

2. Since the pressure on the student is reduced, there is a tendency to relax and as a result this will affect the student's performance.

3. If the cassettes and the course material are not updated regularly, the student may end up having to read obsolete material.

2.3.3 Advantages of Web Based Education Systems

1. The student can rapidly go through the entire course material.

2. If the professor agrees he/she can submit his exercises, projects and exams earlier than the he stipulated 15 weeks complete his/her course earlier. This could be one of the biggest advantages of this system.

3. The professor can keep on adding detail to his course material and know the student can access this from any part of the world. This is especially useful to students who work and attend school part time.

4. The student can send email to the professor for any queries or doubts he has and can get a reply back from the professor.

2.4 Internet as a Delivery Mechanism for Education

2.4.1 Background

The internet was started about twenty-five years ago. It was initially called ARPAnet, which supported military research for the US Department of Defense. ARPAnet connected local computer networks, research universities and military bases. Since most computers those days were IBM mainframes and it would have seemed logical to have a large mainframe house the internet but since damage to that system would bring down the network, the designers decided to set up a distributed system. Thousands of computers connected world wide, form the backbone of the Internet. Since most of these computers were, and still are, Unix based systems, the natural protocol to use for information exchange was Transport Control Protocol/Internet Protocol (TCP/IP). This protocol has been adopted as a common standard and this has contributed greatly to the growth of Internet. Today all operating systems like Windows 95, Windows NT and IBM OS 390 support TCP/IP.

In 1989, in Europe, a system called the World Wide Web (WWW) was developed. WWW brings about Graphical User Interface (GUI) to world wide networking.

This allowed the integration of full color graphics text of varying typefaces, animation, sound and video. For complete access one would need multimedia equipment like speakers to hear sound and a web browser. Graphics cards are now available to render high quality graphics at faster rates.

Each website would typically consist on a homepage, which has links to connect to detailed information.

2.4.2 Components of Internet

Hardware and Software

To develop instruction and deliver it on the World Wide Web one will need certain hardware and software configurations. The client PCs will also need certain hardware and software. The provider of instruction who is going to provide the instruction on the World Wide Web will need a high end computer with connectivity to the internet and Hyper Text Transfer Protocol (HTTP) server software. The software developer who will develop the software will need Hyper Text Markup Language (HTML) editors or tools. The end user or the student in this case will need a computer with a modem, internet connectivity software and a Web browser.

Following is a list of tools :

Computers and Connections

Today Web software is available on nearly every platform you are likely to come across. These include PCs, Macintoshes, and Unix workstations, all of which can be used for designing, providing, and browsing Web Based Education. A PC or a Macintosh can very easily accomplish the task of designing and browsing. The HTML files that will be accessible to students on the internet need to be on a very powerful machine which has a fast internet connection. This is needed to ensure that the large number of students who will be accessing the system will have a good response.

A fact that one must remember is that while one browses the Web from a computer that has internet connectivity, Single Line Internet Protocol (SLIP) or Peer to Peer Protocol (PPP) , is used by the computer to connect and transfer data from the PC to the host computer. Most SLIP/PPP programs have a dynamic address which means that every time the student's computer connects to the service provider, it gets a different address. Though this is true for the client PC, the server must necessarily have unique Internet Protocol IP address. Generally a service provider will have a block of addresses to his internet service and the users can use these addresses to connect.

Browsers and Servers

- ***Servers***

The World Wide Web has several good lists of available server software. Yahoo HTTP Server Index and the W3Org Server Index are two of the best.

- ***Browsers***

There is a wide variety of packages that allow a user to browse the World Wide Web. These are available for nearly every platform and operating system. They differing factor among these is how they handle multimedia, which version of HTML they read, and which special extensions they allow. Netscape, Internet Explorer, Mosaic are the most popular of these. Web Based Education developers need to be familiar with different browsers, because each of these may have different characteristics which will in turn affect the content viewed by the student using the browser. There are certain text based graphics like Lynx which will not recognize any inline graphics that may be in the instruction. Search engines like Yahoo, Infoseek, Alta Vista, ExCite, America Online Net Find will give a list of browsers and related software when invoked.

HTML and Editors

- ***HTML***

HTML is the language used to markup a document in such a manner that browsers will correctly interpret what is in the document and display it properly. This is a simple but a very powerful tool for creating interactive hypermedia internet based applications. All it requires is a text editor or a word processor and a developer can create a HTML page using simple commands. HTML was first developed in 1989 at CERN, the European Particle Physics Laboratory. Currently there are several versions of HTML which include 1.0 to 3.0. There are certain browsers which will only read certain version of HTML while there are certain browsers like Netscape which have added extension to HTML which are read only by their own browser. The WWW Consortium's History of HTML on the world wide web can provide more details on the different versions.

- ***Editors***

HTML documents have essentially components called tags which differentiate between the title, heading, body. Editors are software tools which allow the developer to create web pages without writing the HTML tags. There are three types of HTML editors:

Standalones

Converters

Templates/add-ons.

Applications that work independent of other applications are termed as stand alone applications. The HTML editor can edit the document in plaçe and add, edit or delete the tags.

Converters change text from some format to HTML. Such editors which do conversion to HTML can speed up the development and are very beneficial to HTML development. Templates and add-ons work from within other programs. Typically word processors are there that allow you to create HTML formatted documents within those applications.

Viewers and Other Helper Applications

Text and GIF images are the most common form of content on web pages and most browsers are capable of displaying these. Quite a few of them also support images with some even going further and displaying audio and video. File types that are not handled internally by the browser software are handled by external programs. These are called viewer or helper applications. These can be specified in the individual browser configuration. These helper application programs include QuickTime, movie viewers, RealAudio, Telnet and other applications. These applications must preexist on the system in order to be able to get called by the browser. These applications differ from platform to platform and quite a few are readily available on the Internet.

2.4.3 Creating the Instruction

HTML provides an easy method of putting information together and creating web pages. Though this is easy, creating web based instruction has more to this. Instruction is the deliberate organization and presentation of information with the end goal of promoting specific learning. The design of Web based instruction must have the instructional aspects as the top priority. Though the web site may look very glamorous it could be totally ineffective in achieving the goal of educating the student and this would defeat the purpose of Web based education.

None of the current design models deal specifically with the design of Web Based Instruction (WBI), but we can extrapolate from them when creating our instruction. There are two major schools of theory in instructional design today; the Objectivists, whom we will represent with Dick and Carry's Instructional Systems Design model[1]; and the Constructivists, whom we will represent with the Hypermedia Design Model, based on Spiro's Cognitive Flexibility Theory[6]. WBI can be designed under either paradigm.

The Instructional Systems Design Model (ISD)

Traditional Objectivist Instructional Systems Design models try to provide a series of steps that will, if followed, lead inexorably to the production of effective instructional materials. These models take little account of individual learners and the differences in prior knowledge and motivation that each brings to the instruction, save that they all meet a preset list of entry behaviors. The final goal is not that the learner will know "x" or understand "y". Instead the outcomes are all described in terms of behaviors.

The learner will be able to do "z". This makes evaluation relatively simple. Can the learner exhibit the required behavior? Which may well be appropriate if the knowledge is procedural and can be exhibited. But if the instruction deals with declarative knowledge, or more importantly with higher levels of thinking and learning, theses models and the instruction produced by them can prove ineffective.

The traditional models break down the process of creating instruction into seven basic steps, to be completed in order. These steps can be paraphrased as:

1. Identify the instructional goal for the module in terms of terminal behaviors, that is what will the learner be able to do when they have finished the instruction.

2. Break that behavior down into a hierarchy of subordinate skills. Which skills the learner have to have to perform the terminal behavior? What skills are required to perform those behaviors? And so on into the most basic levels.

3. Examine your hierarchy and determine the minimum level of skills with which you expect your learners to come to the instruction. If your terminal behavior is to be able to prove a given geometry theorem, can you reasonable expect your learners to come to the instruction able to read and perform basic calculations.

4. Performance objectives differ from instructional goals in that performance objectives are the behaviors the learner will evince at the end of each subsection of the instruction to show she has mastered the subordinate skills.

5. Create test items based on the performance objectives.

6. Develop the actual instruction. This step includes media selection, strategy development, and production.

7. Finally the designer has to evaluate the effectiveness of the instruction. Can the learner actually do what the designer intended for her to do?

In this paradigm, all instructional objectives are set by the designer. It is assumed that all learners coming to the instruction are either intrinsically or extrinsically motivated to learn the behavior set in the instructional goal. Individual learner differences are often either ignored or generalized.

Cognitive Flexibility and the Hypermedia Design Model

Cognitive Flexibility Theory (CFT) deals with "the special requirements for attaining advanced learning goals, given the impediments associated with ill-structured features of knowledge domains" [5] and suggests a metaphor "of the criss-crossed landscape with its suggestion of a nonlinear and multidimensional traversal of a complex subject matter". This desire for multiple perspectives and knowledge criss-crossing is well supported in the Internet environment, especially using the hypermedia of the World Wide Web in conjunction with one of the Net's discussion facilities.

"We have referred to the need for rearranged instructional sequences, for multiple dimensions of knowledge representation, for multiple interconnections across knowledge components, and so on. Features like these correspond nicely to well known properties of hypertext systems, which facilitate ... multiple linkages among content elements."

Following Spiro's discussion of the "criss-crossed landscape", the Hypermedia Design Model (HDM) uses a geography/cartography design metaphor. It is important to note the difference between design metaphor and instructional or interface metaphor. The design metaphor deals with how the designers organize the learning domain during the creation of the learning environment; while interface metaphor deals with how the learner accesses knowledge within the environment. One of the assumptions behind the model is that the role of the guide will be taken by the instructional medium rather than by a teacher in a classroom situation. Another important aspect of this model is that it differentiates between design goals and learner objectives. Design goals are that knowledge which the designers hope that the learner will construct from the environment.

Learner objectives are what the learner actually comes to the environment wanting to learn. Traditional ISD models deal almost exclusively with design goals, where as HDM is designed with the idea that learner objectives are paramount, but that guidance can be provided to help the learners reach those objectives.

Define the Learning Domain

One must set the boundaries on what is to be presented to the learner. The larger the boundaries, the less detail one will be able to include within them. For example there are very few maps of the whole world that show street level detail of your home town. The opposite is also true, if the domain is smaller then the level of detail that must be encompassed must be very high, which implies that detailed instruction needs to be on this kind of a web page. Domains are rarely complete and unique. Physics, for instance, overlaps with mathematics, chemistry, history, and any number of other domains. The same is true for almost every traditionally defined learning domain. In the instruction one needs to define where the domain that one is going to cover with this particular environment begins and ends and what is included. An environment for exploring Newtonian thermodynamics, for instance, might include equations for the various laws but not contain the algebra skills necessary to solve those equations. CFT, and therefore this model, deals mostly with the pursuit of intermediate and expert knowledge in complex and ill-structured learning domains. If the learning domain is simple or well structured, one might want to consider using a more traditional ISD model.

2.4.4 Internet Application

The internet became very popular for delivery of electronic mail(email) messages throughout the world. In a matter of hours an email can be delivered from one place to another. Since the Internet spans a number of countries, this becomes a very powerful delivery mechanism.

The use of Internet has changes dramatically in the past couple of years. The current applications of Internet include:

1. As a medium to market products, receive orders, send invoices, perform help desk kind of activities.
2. As a medium to send upgrades to users, especially software companies. E.g. Users can download anti-virus software upgrades from Symantec Corporation using the Internet.
3. As a worldwide library resource.
4. As a medium for teleconferencing with tools like Microsoft NetMeeting.

2.4.5 Internet and Education

The World Wide Web (WWW) deals with how an instruction is organized and how it is presented. It encompasses the delivery medium, content provider, and subject matter. Information on the Web is organized in an ever expanding network of nodes and links that represent the more traditional domains of knowledge. Teachers and designers can create navigational paths to guide the students. These navigational paths or web pages are created using a very simple language called HTML.

The web comprises of text, graphics, video and audio. Reiser and Gagne's media selection diagram [4] and Merrill and Goodman's strategy and media selection technique [3] indicate that these properties make the Web most useful when used to explore intellectual and verbal knowledge, and when exploring affective learning. The web, with it's versatility and interconnectedness offers one of the most effective ways to work with learners who are spread over a wide geographical region.

If the learners have no access to an Internet connected computer the web is not an option for delivery of education. Since text is the primary mode of communication on the Web it is necessary that your learners be literate, or at least be approaching literacy. Hence the use of web in early childhood learning is not recommended. Also, it not advisable to use the web if the instruction requires a great deal of audio and video since this tends to slow down the web page retrieval.

Through helper applications and internal mechanisms the Web can connect a learner to almost any part of the Internet. Because of this the Web shares the advantages and disadvantages of the rest of the Internet. McManus' description of the Internet fits the World Wide Web equally well.

"The Internet can deliver video, but not as quickly as videotape, television, or CD-ROM. It can carry real time personal interaction, but not as well as telephone or video conferencing. It can display textual information, but not a usefully as a book or magazine. Why then should the Internet ever be used. The Net has two real advantages over other media. It combines advantages of other media so that it conveys video and sound better than a book, is more interactive than a videotape and, unlike a CD-ROM, it can link people from around the world cheaply. The second advantage, and one that is often overlooked when discussing the Internet as a delivery system, is that it can also be a content provider. The Internet is, arguably, the largest and most diverse information resource in the world today. It is possible to incorporate the wealth of information available on the Net in your design. For instance if you are designing a module on renaissance art history, you can include links to the Vatican Library and the Louvre, as well as to the Art History exhibit of the Australian National University, just to name a few. This sort of immediate access to information and resources can not be found with any other medium." [2]

CHAPTER 3

ORACLE AND INTERNET

3.0 Introduction

Oracle Corporation is the market leader in database products. Oracle is also spearheading its products in the direction of the world wide web. Today it has a range of products geared develop a web based product from its full life cycle. These products include

1. Oracle Designer 2000
2. Oracle Developer 2000
3. Oracle Web Server.

The Designer 2000 is the tool enables complete design of the project along with the database and the forms. This product is composed of a set of utilities which take the project through its various component sub systems. Designer 2000 Version 1.3.2 is used for the development of this project. Section 3.1 deals with Oracle Designer 2000.

The Developer 2000 is Oracle's graphical user interface design tool. This tool helps the developer to build forms, reports and graphics using visual tools. Developer 2000 Release 1.5w is used for the development of this project. Section 3.2 deals with Oracle Developer 2000.

The Oracle Web Server is the server to which the remote user will be connecting to. This web server in turn will make calls to the Forms Server which will connect to the database. Oracle Web server 2.1 is used for this project. Section 3.4 deals with Oracle Web Server.

In addition Oracle 7.3.3 is used as the database engine for this project. This relational database management system is very widely used in the industry. Section 3.3 deals with Oracle 7.3.3.

3.1 Designer/2000

3.1.0 Introduction

The building of Oracle 7 based systems involves a number of major considerations including,

- Performance
- Integration of other application systems
- Documenting the system
- Communicating the design and usage
- Providing a scaleable architecture
- Avoidance of reinventing the wheel

The section will cover in detail the following areas:

1. Designer/2000 Product Overview
2. Database Design
3. Procedural server-side logic design
4. Server Generation and Maintenance
5. Design Recovery

3.1.1 Designer/2000 Product Overview

Designer/2000 is Oracle Corporation's new generation modeling and generation toolset, and is rapidly becoming adopted as a valuable aid for development teams that are building and maintaining applications based on Oracle 7 technology.

One of the key components of the Designer/2000 product is the underlying repository. The repository is a controllable multi-user environment that provides for the design and generation of all Oracle 7 functionality. In fact the repository is built as a standard Oracle 7 database, and utilizes an API written in PL/SQL.

The Designer/2000 product covers the full spectrum of systems development from business process engineering through to delivery and maintenance of client/server systems.

Data Diagrammer

The Data Diagrammer provides a graphical environment for developing and managing databases. Developers represent tables, views, keys etc. The Data Diagrammer creates, manages and displays logical or physical database designs.

The Data Diagrammer also offers the database designer:

1. Database Design Wizard that creates normalized database designs directly from entity relationship models.
2. Facilities to specify client/server partitioning, allowing designers to cause constraints to be validated on the client-side, the server-side or both.
3. Facilities to specify client/server partitioning, allowing designers to cause constraints to be validated on the client-side, the server-side or both.
4. The ability to define data security, users, groups of users, roles and access rights.

5. Reverse engineering of database definitions and procedural components that were not created in the Designer/2000 environment.

6. Access to the Server Generator that transforms repository-based schema definitions into SQL scripts either for Oracle 7 databases or ANSI standard SQL databases.

The Data Diagrammer delivers complete database designs, either from reverse engineered instance databases, from business models (entity/relationship) or directly entered designs. The Data Diagrammer can manage multiple diagrams, for example, to display partitioned database designs, but basing them on a single repository. It can use color, font and linestyle to customize diagrams, and the user can specify preferences for the appearance and content of diagrams that are retained for subsequent sessions.

Usage:

The Data Diagrammer has been extensively used in this project. After the creation of the Entity-Relationship model the mapping was done. Then the mapped entities were included in the Data Diagrammer to generate the data schema. This was then checked if the tables and fields were created ok. Then the Data diagrammer was used to generate the Data Definition Language (DDL) code to create the tables, indexes, keys. This code was then executed to create physical tables in the Oracle Database. Appendix A lists the Data Diagrammer figures.

Module Logic Navigator

Using Oracle 7, developers have the option to specify procedural logic to implement business rules on the server. These can be invoked as a result of pre-defined triggers on data changes, for example after insertion into the Customer table, or on specific application events, for example whenever the Customer Account Status is changed. The Module Logic Navigator is specifically designed to assist developers in creating and modifying such procedures.

A developer using the Module Logic Navigator can:

1. Create any valid PL/SQL statement by drag and drop from a statement template library.
2. Reference any valid data definitions in the Repository by drag and drop.
3. Navigate large modules by using an auto-synchronizing outliner.
4. Find incomplete editing from the current or previous sessions.
5. Syntax check PL/SQL.
6. Create cross-reference documentation of which tables and columns a module accesses.

PL/SQL modules created in this way can then be implemented using the Server Generator. And, of course, PL/SQL can be reverse engineered. The power of the Module Logic Navigator and the repository management tools are then available to help database administrators and designers get existing databases under control.

Repository Object Navigator

The Repository Object Navigator (RON) presents the developer with a Windows 95 explorer style interface to any and every object in the Designer/2000 Repository, arranged by type. The Repository Object Navigator is not simply a read-only browsing tool. It provides update of all the objects, and allows the update of sets of objects disjointedly selected. Update can be either by direct input or from the paste buffer. For example, if a database column definition has been used in ten or twenty screens and reports, and the database designer changes that definition, it is essential to assess the impact and make changes.

Using the Repository Object Navigator the project manager navigates to the column changed, copies the changed property to the paste buffer, she or he then navigates to the set of modules that use that column, selects the ones which are to inherit the change and then pastes the new property value or values onto the target set. The change is effected simply and quickly, but still under user control.

Usage:

The Repository Object Navigator (RON) was extensively used in this project to maintain entities, relationships, tables, fields, keys, sequences. The entire application maintenance of this project for the Designer/2000 part was done from the RON.

Server Generator

The Server Generator reads definitions held in the Repository, including the specified distribution of constraint enforcement across the client/server divide and creates:

1. Database definitions: tables, columns, views.
2. Keys and constraints for data integrity.
3. Indexes and clusters for performance.
4. Oracle 7 role-based security model.
5. Distribution design: nodes and databases.
6. Replication snapshots and automatic synonym generation for data distribution.
7. Server-side PL/SQL logic.

The Server Generator creates 100% database implementations directly from the Oracle Repository.

Usage:

The Server Generator was used on this project to create the data definition language code to create the tables, primary and foreign keys, and indexes. Appendix A has figures indicating the usage of the Server Generator.

3.1.2 Database Design

It is very essential that "designing" an Oracle 7 database system be considered. There are a number of important reasons:

1. Performance - the initial design of a system can have tremendous impact on its final performance.

2. Integration - without some sort of design specification it becomes very difficult to for a team of developers working on an application to produce a common application, this is equally true when the system being build needs to be integrated into existing systems.

3. Communication - if a design exercise is undertaken, the decisions in that process need to be communicated to all members of the team.

4. Reuse - it may well be possible to reuse existing successful design solutions.

Some development organizations are familiar with the concept of producing an information model for the business requirement they are developing as a database system. Some organizations represent that information model as an entity relationship model. This section will cover how we can translate the information model into a physical database using Designer/2000.

Using the Database Design Wizard the Entity Relationship Model can be translated into a database design, there are a number of significant advantages to doing this which are as follows:

1. Save development time.
2. Achieve consistent results.
3. Avoid a great deal of repetitive work.
4. Maintain the database design.

In practical terms the Database Design Wizard performs the following tasks for a database designer automatically and without error or misinterpretation.

1. Entities become tables.
2. Attributes become columns.
3. Unique identifiers UIDS become primary and unique key constraints.
4. Relationships become foreign key columns and constraints.
5. If an entity has several uids, the designer can choose the most appropriate for the primary key constraint, the rest become unique key constraints.
6. Sub type entities are implemented as multiple or single tables under designer control.
7. Many to many relationships are automatically resolved by the creation of intersection tables.

The Database Design Wizard produces a complete design and the definitions produced could be used to generate the SQL DDL scripts. Often though, the first cut database design has been achieved, the data diagrammer is used to refine the database design and there a number of reasons as to why designers may want to go through this exercise.

For example, convenience database objects to may need to be added to the database schema, i.e. views may need to be created to hide the complexity of the design, sequences allowing the generation of certain column values. The application may also require additional columns for auditing , journalling etc. There is also the whole area of performance, certain columns in the database will need to be indexed for performance, and the structure of certain tables may need to be changed to speed up access times. Using the Data Diagrammer the designer can use the column definition spread table to specify and modify column definitions, the benefit of doing this is so that we can utilize other Designer/2000 utilities.

The order in which the columns appear in the table Datatype are, the datatype of the column, Average Length Estimate, average length Max Len Column, length Dec Places, Accuracy of number columns, Optional, Whether the column will allow NULL values, Initial Volume Estimated percentage, Final Volume Estimated percentage, Default Value.

The Oracle 7 server allows for the definition of data integrity constraints that are automatically enforced by the server. The types of integrity constraint are :

1. Nulls - i.e. mandatory or optional columns
2. Unique Column Values - all values in a column are unique
3. Primary Key Values - ensures that all rows in a table can be uniquely identified by the values in a column (or set of columns), the implication being that the column is mandatory and all of its values are unique.
4. Referential Integrity - a rule defined for a column that allows inserts or updates only values that exist in another table. Also includes rules that dictate the type of data manipulation allowed on referenced data and what actions to be performed on referencing data as a result of such manipulation.

5. Complex Integrity Checking - user defined rule for a column that allows or disallows certain values.

The Data Diagrammer allows you to simply create and utilize the above constraints .

1. NULLS - simply define a column as mandatory by setting it's "Optional?" property to false.

2. Primary Keys - The Database Design Wizard automatically creates a primary key constraint for the source entity primary uid, or using the Data Diagrammer directly you can edit or create primary key constraints and components in the edit table dialog.

3. Unique Keys - use the Data Diagrammer edit table dialog.

4. Foreign Key - diagrammatically or in edit table dialog.

5. Foreign Key Delete and Update rules :

   a. null - no delete/update rule
   b. cascades - deletes/updates a row from the table if parent row is deleted/update
   c. restricted - prevents deletion/update of parent rows
   d. nullifies - updates the foreign key to null if the parent row is deleted/updated
   e. defaults - updates the foreign key to a specified value if the parent row is updated.

Oracle 7 Implemented

ONLY the Delete Cascade, Delete Restricted and the Update Restricted rules are enforced by the Oracle 7 server. The Forms Generator however, Designer/2000 can create the application code to support all delete and update rules at the application level.

Check Constraints - are implemented using the Data Diagrammer edit table dialog.

3.1.3 Procedural Server-side Logic Design

There are two areas of interest in writing server side procedural logic , PL/SQL stored procedures, packages and functions and database triggers. We will first consider the area of database triggers.

Database triggers can be used to supplement declarative integrity constraints to enforce business rules. Because the triggers are written in PL/SQL the rules can be completely customized. Triggers can be commonly used for:

1. Implement referential integrity rules not supported by declarative constraints.
2. Enforce complex business rules.
3. Automatically generate derived column values.
4. Provide transparent event logging and auditing.
5. Enforce complex security authorizations.

The Module Logic Navigator provides a graphical environment to aid creating and maintaining both free standing PL/SQL modules and database triggers. The Module Logic Navigator provides a selection tree and an edit window to define the PL/SQL trigger module. *The Selection Tree* provides correct PL/SQL constructs and allowable data values that can be dragged and dropped to the editor or the outliner.

Server Generation

The server generator is used after the designer has finished the database design, the generator will produce DDL command files that can be used to build a "live" database. The files generated are SQL*Plus command files and are therefore portable to any Oracle environment Below is a table that describes all of the database objects that the server generator will produce scripts for.

Table 3.1 Oracle Designer/2000 Object Generation

Oracle 7 Object Name	**Generate Y/N**
Table	Y
View	Y
Synonym	Y
Index	Y
Sequence	Y
Cluster	Y
Integrity Constraints	Y
including Primary Key	Y
Foreign Key	Y
Check Clause	Y
Default Value	Y
Null	Y
Snapshots	Y
Rollback Segments	Y
Tablespaces	Y
Database	Y
Database Trigger	Y
PL/SQL Package	Y
PL/SQL Procedure	Y
PL/SQL Function	Y
Database Link	Y
Role	Y
User	Y
Grants	Y

In the previous sections we have seen how the Data Diagrammer and Module Logic Navigator can aid database design. The other significant tool that is extremely powerful is the Repository Object Navigator. This tool supports the designer in allowing him/her to manipulate repository objects extremely easily. For example, a designer can create a space definition in the repository, this space definition can be applied to multiple database objects simply by selecting the required objects and changing the space definition property to the name of your newly created space definition. This means that whenever those objects are generated they will automatically be generated with the correct space definition. This reuse capability is available in many parts of the Designer/2000 product set and significantly aids in the speed and quality of the generated database structures.

The quality of the outputs of the Designer/2000 product provide an excellent repository of documentation. The SQL DDL script files are fully commented, thus allowing subsequent users understand their functionality. Below is sample output from the Server Generator.

```
REM This ORACLE7 command file was generated by Oracle Server Generator
REM Version 5.5.7.0.0 on 26-OCT-95
REM For application OBS version 1 database OBS7
REM
REM TABLE REM CUSTOMERS

REM PROMPT PROMPT Creating Table CUSTOMERS

CREATE TABLE customers( cs_reference NUMBER(6,0) NOT NULL,
cs_address_line_1 VARCHAR2(40) NOT NULL, cs_last_name VARCHAR2(30) NOT
NULL, cs_first_name VARCHAR2(30) NOT NULL, cs_telephone_number
VARCHAR2(15) NOT NULL, cs_address_line_4 VARCHAR2(40) NULL,
cs_address_line_2 VARCHAR2(40) NULL, cs_address_line_3 VARCHAR2(40) NULL,
cs_lc_code VARCHAR2(4) NULL, cs_acc_balance NUMBER(11,2) NULL,
cs_od_allowed VARCHAR2(1) DEFAULT 'N' NOT NULL CHECK ( cs_od_allowed IN
( 'Y' , 'N' ) ) ) STORAGE ( INITIAL 777 NEXT 11 MINEXTENTS 1 MAXEXTENTS
99 ) ;

COMMENT ON TABLE customers IS 'Created from Entity CUSTOMER by OBS on
24-AUG-94';
```

```
COMMENT ON COLUMN customers.cs_reference IS 'A unique system-generated id for
this customer';

COMMENT ON COLUMN customers.cs_address_line_1 IS 'First address line';
COMMENT ON COLUMN customers.cs_last_name IS 'Last Name of Customer';
COMMENT ON COLUMN customers.cs_first_name IS 'First name of Customer';
COMMENT ON COLUMN customers.cs_telephone_number IS 'Telephone area code
and number';
COMMENT ON COLUMN customers.cs_address_line_4 IS 'Fourth address line';
COMMENT ON COLUMN customers.cs_address_line_2 IS 'Second address line';
COMMENT ON COLUMN customers.cs_address_line_3 IS 'Third address line';
COMMENT ON COLUMN customers.cs_lc_code IS 'A 4 character code to uniquely
identify the Company.';
```

The SQL DDL script files that are generated are defined by types, i.e. the table creation is performed initially, and then any constraints that are required are applied by a separate script file, similarly with index files etc. This gives the Designer and DBA a lot of flexibility in how they build the final database.

Maintenance

Designer/2000 benefits the maintenance stage of an Oracle 7 based system as well in the construction of that system. The maintenance stage of an application is typically where a large amount of effort is expended in the overall life cycle of that application, and that is why it is important to have a clear understanding of the application and all its dependencies. The use of the Designer/2000 repository can help dramatically in a number of ways in understanding dependencies:

1. Objects have relationships and these can be clearly documented and understood by using the repository.

2. Objects need to be documented for future reference and understanding.

3. Typically there will be iterations and new versions of the application over a period of time, and it is important to be able to move the application forward with those changes.

3.1.4 Design Recovery

Many existing systems have little or no formal documentation but typically manage valuable data about a business. These are often described as "legacy systems." They must be maintained but the information to do so effectively is missing, the maintenance programmers need to know.

1. Where the data is being stored.
2. What format the data is in and how the data is used by the existing system.
3. How the application system fits in with the current business practice.

Eliciting this information is known as Design Recovery.

The Reverse Engineer Database utility and Table to Entity Retrofit utility automates this reverse engineering process. Repository object definitions can be derived form the elements of the existing system if the system was developed on an Oracle 7 database. When reverse engineering operations have been performed, the new application system definition or definitions can be used as the basis for system development.

Stages Involved in Reverse Engineering

A reverse engineering operation can involve the following stages.

Reverse Engineering - Stage One

This stage covers reverse engineering database objects, creating module data usages of existing PL/SQL modules.

Reverse Engineering - Stage Two

This stage covers further steps required for recreating a business model, rationalizing the top-down business model with the reverse engineered data and completing the cross-referencing.

If primary, foreign and unique keys are present in the Oracle Data Dictionary, they will have been automatically identified. Below is a table that details the database objects that can be reversed engineered.

Table 3.2 Oracle Designer/2000 Object Re-generation

Oracle 7 Object Name	**Reverse Engineer Y/N**
Table	Y
View	Y
Synonym	Y
Index	Y
Sequence	Y
Cluster	Y
Integrity Constraints	Y
including Primary Key	Y
Foreign Key	Y
Check Clause	Y
Default Value	Y
Null	Y
Snapshots	Y
Rollback Segments	Y
Tablespaces	Y
Database	N
Database Trigger	Y
PL/SQL Package	Y
PL/SQL Procedure	Y
PL/SQL Function	Y
Database Link	N
Role	N
User	N
Grants	Y

The Repository can now be used to define and implement changes and enhancements to the existing database. If definitions of new tables or other objects are created in the Repository they can be implemented in the existing Oracle Data Dictionary using the Generate DDL utility. If changes to existing repository objects (e.g., adding a new column to a table, add a new foreign key) are defined in the Repository they can be implemented in the Oracle Data Dictionary using the Alter Database Command Generator utility. If the Repository and Oracle Data Dictionary are out of step during maintenance, the Cross Reference utility can be used to indicate the differences, allowing the appropriate steps to be identified and enacted.

3.2 Developer/2000

3.2.1 Introduction to Developer/2000 for the Web

Developer/2000 for the Web is a new generation of Oracle development tools that enable you to deploy new and existing applications on the World Wide Web, either on an internal company intranet, or on the Internet. Developer/2000 for the Web takes advantage of the ease and accessibility of the Web, and elevates it from a static information-publishing mechanism to an environment capable of supporting complex, dynamic applications. Oracle Developer/2000 empowers organizations with the ability to rapidly and productively build sophisticated systems which can scale from workgroups to the enterprise. With support for RAD, user interface portability, and open interoperability, as well as unique capabilities including unified client/server development, objects by example, full object component (e.g. OLE2 and VBX3) support, and drag-and-drop application partitioning, Developer/2000 is the leading choice for second generation client/server development.

Developer/2000 includes forms, reports, graphics, client/server development, and translation management components. Developers can build systems using any combination of these features, as well as complementary functionality provided by third-parties, specifically tested and guaranteed to work well together.

While today's enterprises have many options available for developing applications, only Oracle Developer/2000 supports second generation client/server development, with the productivity, reliability, and scalability to satisfy existing mainframe and desktop users.

3.2.2 The Best of the Web and Client-server

Developer/2000 for the Web provides solutions that enable you to leverage the benefits of the Web, while maintaining the strengths of client-server computing. The Web greatly reduces the costs of administering and maintaining applications, while allowing for a thin, low-cost client. At the same time, it allows you to leverage your existing client-server applications, which may be mission-critical applications that support—and often drive—your business practices. These applications must be scalable to large numbers of users and open to all of your environments.

3.2.3 Three-Tiered Architecture

In most client-server implementations today, running applications is a highly client-intensive process. Though data is extracted from a remote database server, applications run on client machines, which often have limited processing power and memory capacity.

Developer/2000 for the Web supports a three-tiered architecture that delivers the benefits of both client-server and the Web in a single application. In a Web implementation, application logic and processing are focused on a middle tier of application servers instead of on desktop client machines. These include the following as indicated in the Table 3.3.

Table 3.3 Three Tier Architecture

TIER	Remarks
Front-end	Any number of client desktop machines
Middle	One or more application servers
Back-end	One or more database servers

3.2.4 Forms Web Architecture

To run new or existing Forms applications on the Web, you should install and operate Developer/2000 for the Web on the middle tier of a distributed three-tiered architecture.

3.2.4.1 About the Forms Client and Forms Server: The Forms component of Developer/2000 for the Web consists of two components:

the Forms Client and the Forms Server.

Forms Client

The Forms Client is a Java applet—downloaded at runtime from an application server to an end user's Web browser—that displays the form's user interface and manages interaction between end users and the Forms Server. The Forms Client receives "bundles" of interface commands from the Forms Server and translates them (in sets) into interface objects for the end user. Some interface events handled by the Forms Runtime Engine in a client-server implementation (such as typing characters in a text field, or moving around a dialog) occur only on the Forms Client in the Web implementation, with no interaction with the Forms Server Runtime Engine.

The Forms Client is:

1. *Generic:* You are not required to deploy a separate Java applet for each application you wish to deploy on the Web.

2. *Dynamic:* The Forms Client dynamically reacts to the current form at runtime, requesting and displaying only the information and user interface elements necessary to represent the current state of the application at any given time.

3. *Feature-rich:* The Forms Client supports all user interface widgets and tools available in a client-server implementation. Due to Java object standards, the look and feel of some Forms widgets may vary slightly when deployed on the Web.

4. *Thin:* At startup, only those class files necessary to render the initial state of an application are downloaded to the end user's machine. Additional class files are downloaded dynamically (as needed) to support additional user interface functionality.

3.2.4.2 ***Forms Server:*** The Forms Server consists of two components:

1. *Listener:* The Forms Server Listener initiates the Forms runtime session and establishes a connection between the Forms Client and the Forms Server Runtime Engine.

2. *Runtime Engine:* The Forms Server Runtime Engine is a modified version of the Forms 4.5 Runtime Engine, with user interface functionality redirected to the Forms Client. It handles all form functionality except UI interaction, including trigger and commit processing, record management, and general database interaction.

Forms Web Architecture

There is a significant difference between deploying a Forms applications in client-server mode and deploying the same application on the Web:

1. *Client-server:* The Forms Runtime Engine (and all application logic) are installed on end users' desktop machines. Although your application can include database-server-side triggers and logic, typically all user interface and trigger processing occurs on client machines.

2. *Web:* The Forms Server Runtime Engine (and all application logic) are installed on application servers, not on client machines. All trigger processing occurs on database and application servers, while user interface processing occurs on the Forms Client.

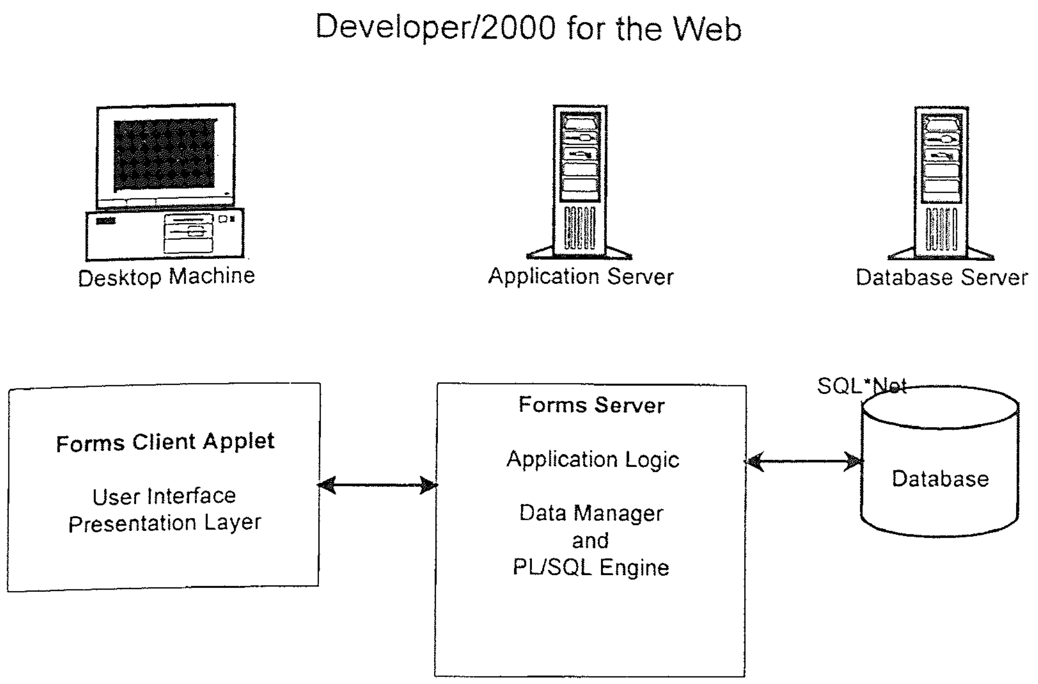


Figure 3.1 Web Architecture

3.2.5. Call and Response: Forms Client and Forms Server

Once a direct network connection is established between the Forms Client and Forms Server, the two components communicate through a series of requests and responses—via compressed messages passed over a network.

Requests from the Forms Client are events (such as "click button" or "display LOV"). Responses from the Forms Server are a series of changes to the user interface (such as value changes, and adding and removing components), all of which the Forms Client converts to display objects.

For example, the Forms Client might receive a response from the Forms Server similar to "create four green text items on canvas CAN_12." The Forms Client translates the response into actual interface objects, in this case, the verdant text items.

The Forms Client contacts the Forms Server when users perform:

1. High-level operations (such as accepting or canceling a dialog).
2. Operations (such as checking a checkbox or navigating between fields) that involve validation processing and cause default and user-defined triggers to fire.

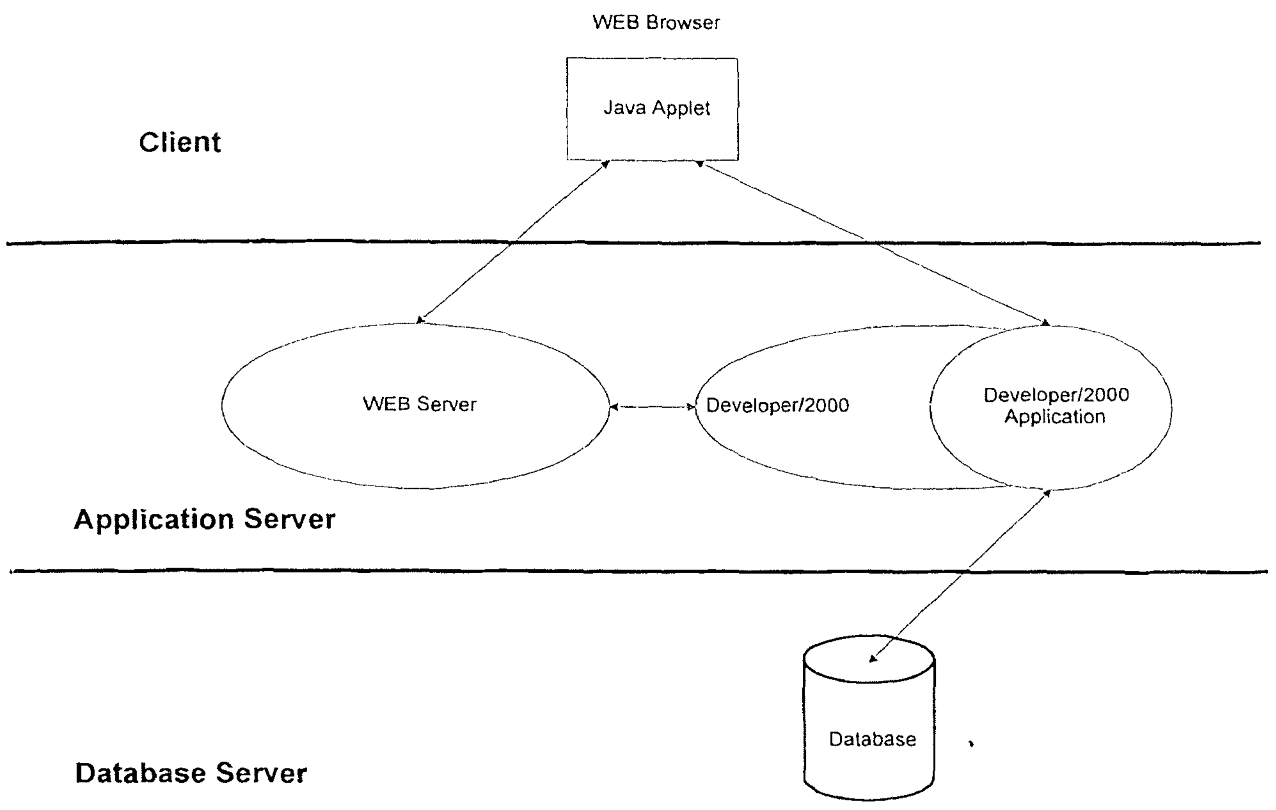


Figure 3.2 Developer/2000 Web Application

To start and run a Forms application on the Web, end users use a Java-enabled Web browser to access a URL. The following sequence occurs automatically:

1. The URL corresponds either to a static (non-cartridge) HTML page, or to an application cartridge, residing on the application server.

2. An HTML page, and then the Forms Client applet, are downloaded from the application server to the user's browser.

3. The Forms Client sends a request to the Forms Server Listener (which resides on a specific port of the machine from which the Forms Client was downloaded).

4. The Listener contacts the Forms Server Runtime Engine and connects to a Forms runtime process (either by starting a new process, or by connecting to an existing process). If included in the HTML page, Forms command-line parameters (such as form name, user ID and password, database SID, menu name, and so on) and any user-defined Forms parameters are passed to the process by the Listener.

5. The Listener establishes a direct socket connection with the Runtime Engine, and sends the socket information to the Forms Client. The Forms Client then establishes a direct socket connection with the Runtime Engine. The Forms Client and Runtime Engine then communicate directly, freeing the Listener to accept startup requests from other end users. The Forms Client displays the application's user interface in an applet window outside the main window of the end user's Web browser.

6. As in a client-server implementation, the Runtime Engine communicates directly with the database through SQL*Net (or another driver, for non-Oracle datasources).

3.2.5 Security and Encryption

Data passed between the database, the Forms Server, and the Forms Client is automatically encrypted before—and decrypted after—transmission by the following protocols:

1. RSA RC4 40-bit encryption (for transmissions between the Forms Client and the Forms Server).

2. SQL*Net SNS/ANO (for transmissions between the Forms Server and the database server) Encryption is enabled by default, but can be disabled. To disable encryption between:

   a. The Forms Client and Forms Server, set the environment variable FORMS45_MESSAGE_ENCRYPTION to FALSE.

   b. The Forms Server and the database, is clearly described in Oracle Advanced Networking Option Administrator's Guide.

3.3 Oracle 7 Relational Database System

3.3.1 Introduction

There are a number of major database packages currently in the market. These include the following:

1. Oracle
2. Sybase
3. Informix
4. Microsoft SQL Server

Oracle is the market leader in this field. This product has been in the market for a number years now and with its broad range of products which always aim at the newest technology in the market, Oracle Corporation has carver a niche for itself. The heart of the company is the Oracle Relational Database Management System. The other products built around this database. The Oracle Web Developer Suite is among the newest suites aimed at web developers with a string database engine at the back. This section deals with the scalability of Oracle and the internals of the Oracle relational Database Management System .

3.3.2 Attributes of a Scalable Development Environment

As mentioned above, ease of use is important, but it should not come at the expense of functionality and an ability to create and maintain large, complex applications. As the size of a development team and project grows, so does the importance of an efficient development environment. The ability to create a team programming environment, to easily reuse application objects, and to support the entire application life cycle, are key components in the quest for productivity gains and reduced maintenance costs. Since one tool cannot do it all, integration with other best-of-breed products such as GUI testing tools and transaction processing monitors, also contributes to the creation of a scalable development environment. Further, tools that enable the modeling of business rules and processes through defined methodologies and support rapid application development, are also more likely to produce consistent applications that ensure data integrity and enforce corporate standards.

3.3.3 Using Shared SQL and Bind Variables

Client/server applications typically contain a number of SQL statements. Every instance of an application typically sends the same SQL statement to the database. Oracle7 enables applications to share executed SQL statements by allowing them to be parsed and stored in memory only once. This leads to lower memory utilization on the server and dramatic improvement in client/server performance. To take advantage of this shared SQL area feature, the SQL statements used must be identical. Thus the shared SQL area will not be used.

SELECT * FROM emp WHERE empno = 10

is issued by one application, and:

SELECT * FROM emp WHERE empno = 20

is issued by another.

To make the use of the shared SQL area possible, Oracle provides bind variables that allow the same statement to be issued regardless of the data being accessed:

SELECT * FROM emp WHERE empno = :empno --> :empno relates to a field in the client application. Regardless of its usage, the execution of any SELECT statement involves five basic steps: Parse, Bind, Define, Execute, and Fetch. Each of these five steps involves an exchange between the client process and the server. In the example, if the first user supplies '10' for the value of the ':empno' place-holder variable during the bind phase, and the second user supplies '20', the SELECT statement is identical from Oracle7's point of view. Hence Oracle7 omits the parse and define steps and saves on parse and define time for the second and all subsequent users. These steps are similar for other DML operations like INSERT, UPDATE and DELETE, except that only the SELECT statement needs to fetch a result set. Bind variables are especially important in complex SQL statements that are executed many times.

The lack of support for bind variables by a client tools affects the amount of network traffic, the number of IO's executed and the amount of memory required on the server to support client processes. Research shows that an average application has 50 SQL statements with each SQL statement containing at least 15 columns. Tests were conducted using a typical SQL statement from a production tracking application. In Test A, the SQL statement was executed with bind variables enabled in a client tool. In Test B, the use of bind variables was disabled. Both Test A and Test B were performed with 2 users. There is a definite performance Improvement using Bind Variables.

The effect of using bind variables in Test A is evident in the reduced network load as the shared SQL area was used when the second user executed the SQL statement. Although these numbers cannot be extrapolated linearly to 100s or 1000s of users, it is reasonable to expect that each subsequent user not using bind variables will not only incur a similar overhead to that described above but also a significant differential in time elapsed. Note that even though a client tool may allow you to define SQL statements with bind variables, this doesn't necessarily mean that the statement sent to the server will include them since the client tool may not be architect to support them. One of the ways to find out if a client tool uses bind variables is to set parameter SQL_TRACE= true in Oracle's init.ora file. This will generate a SQL trace file that enables to find out about the usage of bind variables. Please refer to Appendix C to see where to look for bind variables in a SQL trace file. Alternatively you can somehow make the query sent by the client tool fail and use the client debugger to check the SQL statement prepared by the client to be sent to the server.

3.3.4 Database Cursors

Oracle7 uses database cursors as a means of accessing and scrolling through data in a database table. Typically, one cursor per SQL statement is created as a pointer to a row in a table. It is possible for multiple cursors to be opened and maintained by a single application at any time. Thus, cursor 1 may be opened for execution of a query against a table, cursor 2 may be opened for the update of rows in the table, and so on. Cursors may be automatically created when a SQL statement is issued or in some cases, may be defined manually using a client tool's scripting language.

Client tools that do not provide support for multiple open cursors commonly suffer from severe performance degradation when the number of rows being retrieved from the database increases even above a few hundred records. The following example demonstrates why this is so:

Assume a master-detail application (Departments and Employees). Under optimal operation, cursor 1 is opened to perform a query on the Departments table and cursor 2 is opened for the query on the Employees table. The benefits of having these cursors remain open is twofold: users can scroll freely (up or down) through the records of either table, and developers have control over how many rows should be retrieved for the user at once (since cursors remain open, additional rows can be retrieved later with minimal additional cost).

Using the same scenario but with a client tool that does not support multiple cursors: cursor 1 is opened for the query on Departments, but then that cursor is closed so that the cursor for the query on Employees can be established. Since the cursor on the Departments table is lost, the only way to allow users to scroll through Department records at will, is to query all the Department rows at once. If the table being queried has only a few rows, the potential impact of this problem is small, but for large databases, the network performance when retrieving all the rows from the table at once is likely to be unacceptable. In addition, this style of retrieval will require more memory on the machine to buffer records retrieved.

Some tools attempt to work around this problem by establishing a new database connection for each query being issued. In this way, a single cursor architecture can be maintained in the tool since only a single cursor will be maintained in any particular database connection. However, this approach has other ramifications in terms of the database connection time and the ability to provide read consistent transactions across all the connections and cannot be considered a viable solution.

3.3.5 Array Fetch

Oracle7 supports arrays which are a collection of related data items associated with a single variable name. Arrays can ease programming effort and offer improved performance. They let you manipulate an entire collection of data items with a single SQL statement. The size of an array determines the number of rows that are returned to a calling client application by the server in a single network operation. In most cases it is the client tool that determines whether arrays are used or not.

The following example demonstrates the benefit of arrays in a client/server environment. Suppose you want to query for 10,000 employees into the EMP table. Your select statement would look like this:

```
EXEC SQL BEGIN DECLARE SECTION ;
          NUMBER empno ;
           VARCHAR ename[30] ;
           NUMBER sal ;
           NUMBER deptno ;
          EXEC SQL
END DECLARE SECTION ;
SELECT empno, ename, sal, deptno FROM emp ;
```

To select 10,000 rows, the SELECT statement has to be executed 10,000 times. But if empno, ename, sal, deptno are declared as an array of size 100, the SELECT statement would look like this:

```
EXEC SQL BEGIN DECLARE SECTION ;
 NUMBER empno[100][30] ;
 VARCHAR ename[100][30] ;
 NUMBER sal[100][30] ;
NUMBER deptno[100][30] ;
 EXEC SQL END DECLARE SECTION ;
 EXEC SQL SELECT empno, ename, sal, deptno FROM emp;
```

In the above example, Oracle selects all 10,000 rows into the EMP table in 100 trips. Tests were conducted to evaluate the performance of the above statements with and without using an array. In Test A, a SQL statement was used to select 10,000 rows with array size 100. In Test B no array size was specified. The use of arrays in Test A has a great impact on performance. When array processing is used, Oracle7 is instructed to transmit data in bigger, efficient batches to the client. The performance gains from the use of arrays enables client tools to support applications that scale to support larger number of rows.

A simple way to find out if a client tool supports array fetching is to make sure that it has the ability to explicitly specify an array size, either through parameters or programmatically. Note that different queries may be more efficient with different array sizes. Regardless of the ability to support arrays, it is critical that a tool can control at least screen-at-a-time fetches instead of bringing back all the rows from the server at once, causing unacceptable delays while all the rows are being retrieved. Tools that don't support array fetch often force developers to restructure applications and rewrite SQL just to get acceptable performance.

3.3.6 PL/SQL

Independent SQL statements normally result in at least 2 calls to the Oracle database. PL/SQL (Oracle's procedural extension to the SQL language) allows developers to package multiple SQL statements so that they can be sent to the server as a single network packet. Oracle7 processes SQL statements sequentially and returns the resultant rows once the complete PL/SQL block is finished, greatly reducing network traffic. The second advantage of PL/SQL is, when a client tool supports a local PL/SQL engine, the client tool can process PL/SQL blocks by passing them to it's local PL/SQL engine. The engine executes all procedural statements on the client, and sends only SQL statements to the server. This gives a client the ability to process application-related logic such as numerical or data calculations and non-database validations on the client platform, reducing requests to the server. With a local client PL/SQL engine, the client tool does not have to consume time and resources to understand and then interpret Oracle7's PL/SQL.

For example, if a client tool has no PL/SQL engine then to get employee name in uppercase into a ename column, the following statement has to be issued:

```
SELECT UPPER(ename) INTO :ename FROM EMP;
```

This will generate at least two network round trips. However if the client tool has a local PL/SQL engine then to get an employee name in uppercase into a ename column, the following statement can be issued:

```
BEGIN
:ename := UPPER(:ename) ;
END;
```

In the above example, when the client tool has a local PL/SQL engine, no network traffic is generated as no request is sent to the server for validation.

PL/SQL in a client/server environment

SQL Statement

SQL Statement SQL Statement

SQL Statement SQL Statement

SQL Statement SQL Statement

SQL Statement Network

PL/SQL Procedure Client Server

In the following example of a PL/SQL block , we want to find the first employee who has a salary over $4000 and employee number higher than employee 7902 in the chain of command. These series of commands, embedded into a single PL/SQL procedure are sent over a network as one network packet:

```
DECLARE
salary emp.sal%TYPE;
mgr_num emp.mgr%TYPE;
last_name emp.ename%TYPE;
starting_empno CONSTANT NUMBER(4) := 7902;
BEGIN
SELECT sal,mgr INTO salary, mrg_num FROM emp WHERE empno = starting_empno;
/* Find employees whose salary is greater than $4000 */ WHILE salary > 4000 LOOP
SELECT sal, mgr, ename INTO salary, mgr_num, last_name FROM emp WHERE
empno = mgr_num;
END LOOP;
INSERT INTO emp_tbl VALUES (NULL, salary, last_name);
COMMIT;
```

```
END;
```

In the above example when these series of commands are embedded into a single PL/SQL procedure, then the procedure is sent over the network and executed in one network round-trip. However if these SQL commands were to be executed individually it would generate at least six network round-trips (provided only 1 employee has a salary greater than 4000) and many more as the number of employees whose salary is greater than 4000 increases.

3.3.7 Stored Packages and Procedures

If a client tool is tightly integrated with Oracle7 then it can take better advantage of stored procedures and database triggers to perform multiple lookup/validation operations on the server. The benefits of doing this are that memory usage on the server can be reduced as every user executing a given stored procedure shares the same in-memory copy of the procedure, and network traffic will decline as single remote calls are sent to the server. Multiple users issuing the same stored procedure automatically uses the shared PL/SQL area, eliminating runtime parsing and enabling the creation of more scalable applications. Tests were conducted using a procedure extracted from a call tracking system. The procedure simply validates an appropriate product in the database. In test A an anonymous block of PL/SQL was executed from a client tool. In test B the same PL/SQL block was executed as a stored procedure in the server. The results of the tests showed that although with this particular procedure the difference in number of network packages is not substantial (mainly due to the simple stored procedure used in the test), the performance improvement in terms of the number of bytes sent over the network is overwhelming. The procedure in test A created 6 times more network traffic than the stored procedure in test B. In production applications, procedures will likely be much larger and more complex, and defining them as stored procedure will improve modularity and performance. Thus, stored procedures and functions offer better application performance and reduced server resource usage and network traffic, but only if a client tool can exploit them without limitations.

Support for Application Partitioning

With the advent of database stored procedures, and given the performance and resource benefits of creating logic in the server, it no longer makes sense for the majority of application logic to be defined in client applications. It is therefore important to have the ability to easily migrate this application logic to the server to take full advantage of a client/server environment.

This requirement is not restricted just to existing applications. One of the key decisions a client/server developer has to make, is where to locate the logic for an application: in the client or in the server. Given the newness of client/server, many wrong decision are made due to lack of experience. So once again, the ease with which logic can be moved from client to server is an important factor in correcting a poor design decision. The ability to move code between client and server is known as application partitioning. In order to perform application partitioning, it is mandatory that the same language be able to execute both on the client and the server. In the case of an Oracle application, this may be a client tool which supports the PL/SQL language and has the ability to move code between client and server. The potential gains from doing this are shown in the example above.

One of the ways to determine that the client tool supports application partitioning is to make sure that from within the development environment of the client tool you can move the application logic to either a client or a server side, compile the logic and run the application. If the application at this point runs as before unaware as to where the logic is stored then the client tool supports application partitioning.

3.3.8 Transaction Processing

One of the goals of GUI client/server tools is to make applications easy to use. To meet this objective, it is important to give immediate feedback to users as soon as they enter invalid data. Some client tools defer all referential integrity and cross-table validation checks to the server when data is committed. The effect of this is that users making multiple updates at once may have to go back and reenter values that violated validation rules long after they actually entered them. As a result, applications become less user-friendly and are less productive. It is thus important that database integrity constraints be enforced in client applications as well. This is especially true in transaction intensive applications with larger volumes of data and more complex processing logic. A scalable client tool should be able to read the constraints from the database and automatically apply them to the client application. Most tools should allow the definition of the rules manually, however this is duplicate effort and leads to consistency errors and higher maintenance costs.

Oracle 7 supports both the pessimistic and optimistic locking models. This allows client application behavior to be customized to best suit the environment. As Oracle 7 supports reduces resource contention through row-level locking, a pessimistic model is optimal for OLTP (on line transaction processing) applications. Client applications can ensure the use of row-level locking by issuing a "SELECT FOR UPDATE..." statement when an update to a field is attempted.

For example, if user A is updating a row in table EMP and user B tries to update the same row, a scalable client tool should be able to inform user B immediately that the row is locked by another user. User B should not have to wait for the transaction to be sent to the server before finding out it cannot be updated. A tool can enforce this behavior by taking advantage of SELECT FOR UPDATE, thus ensuring that contention for resources is reduced and that updates by users are consistent. In some client tools SELECT FOR UPDATE is issued automatically (the preferred case), in others it has to be programmed an synchronized with the UPDATE statement for the row. A SQL trace file will show whether SELECT FOR UPDATE is being used or not.

Oracle7 supports the concept of SAVEPOINTs. A SAVEPOINT is a placeholder within a transaction to which a SQL statement may be rolled back. If a client tool automatically issues SAVEPOINTS when a transaction is being processed, it is possible to rollback the transaction to the point of failure, and then successfully commit the entire transaction when the data is corrected by the user. If no SAVEPOINTs are issued, then the user has to reenter all the updates after a ROLLBACK occurs. This leads to end user productivity loss and also increases the potential for inconsistent data to be committed to the database in multi-step transactions.

A SQL trace file will show whether SAVEPOINTs are being used or not.

Oracle7 supports the concept of ROWID. A ROWID is the logical address of a row, and it is unique within the database. ROWID is the fastest means of accessing a row from a database. Client tool which does not use ROWID in a DML statement as a means of accessing an Oracle database will hamper the performance and affect scalability.

A SQL trace file will show whether a ROWID is being used or not.

3.3.9 Conclusion

Oracle 7 is an excellent repository for all database applications and considering its scaleable nature, it all the more imperative that Oracle 7 be used for robust and scaleable projects.

3.4 Oracle Web Server 2.1

3.4.1 Introduction

This section offers a conceptual overview of the Oracle WebServer. It is intended to give context for the more specific task-oriented information that follows. The subsequent sections are specifically targeted at those who manage Oracle WebServer sites

(WebServer Administrators) and those who write application programs for them (application developers). This section covers:

1. Overview.
2. The Web Listener. This is the portion of the WebServer that interfaces to the local network or the World Wide Web.
3. The Secure Sockets Layer. Introduction to SSL public key encryption.
4. The Web Server Manager. The set of Web pages you can use to perform most WebServer administration.
5. The CGI Interface. The standard Web mechanism for executing applications on a Web server.
6. The Web Request Broker (WRB). The core of the WebServer. An asynchronous request broker with an open API that Oracle WebServer uses to execute applications on the server.
7. The PL/SQL Agent. The program the Oracle WebServer uses to execute procedures written in PL/SQL, Oracle's application development language, on the Oracle7 Server.
8. Java. A new language for developing distributed network applications. Oracle WebServer enables you to execute Java on the WebServer itself.
9. LiveHTML. A way to embed dynamic content in Web pages. This content can be either other Web pages or the output of scripts run by the Operating System.

   LiveHTML is an Oracle extension of the NCSA standard Server Side Includes functionality.

3.4.2 Overview

The Oracle WebServer is a Hyper Text Transfer Protocol (HTTP) Internet Server with unprecedented database integration and a powerful development environment. When the WebServer receives a Uniform Resource Locator (URL) from a browser located either on the World Wide Web or on a local network using the Web's protocol (HTTP), it draws on information from the database and the operating system's (OS) file system as necessary or for CGI scripts that do not access the database, and the database is used for Web pages that are generated at runtime using "live" data. Although you can run the Oracle WebServer without Oracle7, one of the great advantages of the product is its tight integration with the Oracle7 Server, the leading database product in the world.

The Web Listener is the component that receives a URL from a Web browser and sends back the appropriate output. When the Web Listener receives a URL, it determines whether the request requires the use of a Service to be accessed through the Web Request Broker (WRB), a program to be accessed through the CGI interface, or whether access to the file system of the machine on which the Listener resides is sufficient. If WRB access is required, the Listener passes the request to WRB Dispatcher for processing; then it returns to the task of listening for more incoming HTTP requests.

The WRB Dispatcher handles requests with the aid of a pool of processes called WRB Executable Engines (WRBXs). Each WRBX interfaces to a back-end application using the WRB API. These applications are called WRB cartridges. The WRB API is designed so third parties can add their own cartridges. The combination of a cartridge and the WRB API is known as a WRB Service. Currently, Oracle WebServer support three kinds of WRB Services:

1. PL/SQL cartridges. These execute stored PL/SQL procedures to generate HTML dynamically using Oracle data.

2. Java™ cartridges. These execute Java code on the Server.

3. LiveHTML. These embed Web pages within one another, and store in Web pages the output of scripts executed by the Operating System.

3.4.3. The Web Listener

The Oracle Web Listener is a HTTP engine that responds to requests for hypermedia documents from web browsers (clients). This section summarizes the Web Listener's capabilities.

Network Communication

Every Oracle Web Listener process accepts connections from web browsers (clients) on one or more IP address/port combinations using HTTP to encode requests for hypertext documents, and the Transmission Control Protocol/ Internet Protocol (TCP/IP) as the underlying connection protocol. Several Web Listener processes may run on single host computer at the same time (see the Oracle Web Listener Administration Form for more information).

Ports

A port is a number that TCP uses to identify a communication channel associated with a specific program. For example, the login program usually accepts login requests on port 49, and the Domain Name Service (DNS) usually accepts name lookup requests on port 53. HTTP engines, such as the Oracle Web Listener, usually accept ordinary connections on port 80 and secure connections on port 443.

Programs must execute with root permissions to access port numbers 1 through 1023, whereas any program can access port numbers 1024 through 65535.

IP Addresses

An Internet Protocol (IP) address is a unique number that identifies exactly one computer on the Internet, although each computer can be represented on the Internet by several addresses. A computer can use different addresses for different Internet functions. For example, a single computer might have one IP address on which it acts as an email routing server, and another address on which it acts as a DNS server.

DNS Host and Domain Names

A fully qualified host name is a unique character string that identifies exactly one computer on the Internet, although each computer can be represented on the Internet by several host names. The Domain Name Service (DNS) maintains a distributed database that maps host names to IP addresses. DNS servers provide Internet computers with lookup services for this database. hal.us.oracle.com
is an example of a fully qualified host name.

A domain is a named DNS host name space, such as us.oracle.com. All host names within a domain must be unique. For example, there must be only one host named hal in the us.oracle.com domain.

3.4.4. Secure Connections

The Oracle Web Listener can accept secure connections using the Secure Sockets Layer (SSL), an emerging standard for secure data transmission. (The combination of IP address and port number is called a socket.) See The Secure Sockets Layer for more information on the Oracle WebServer's implementation of SSL.

Clients use a secure version of HTTP called HTTPS to establish secure connections with SSL. To establish a secure connection with a Web Listener process, a request URL must use "https" instead of "http" must specify a port on which the Web Listener has enabled SSL, for example:

https://www.blob.com:443/

Proxies

You can configure an Oracle Web Listener process to act as a proxy server, an HTTP engine running on a firewall machine that allows clients inside the firewall to access web sites outside the firewall. The Oracle Web Listener implements the proxy behavior defined by Luotonen and Altis in World-Wide Web Proxies (http://www.w3.org:80/hypertext/WWW/Proxies/) under the auspices of the World-Wide Web Consortium (W3C).

File Handling

The Oracle Web Listener uses a virtual file system to keep track of the files it makes available to clients. The virtual file system maps the pathnames used in request URLs (Uniform Resource Locators) to the file system maintained by the host machine's File.

Memory Mapping

The Web Listener uses the host operating system's memory mapping capability when opening a file requested by a client so that the Web Listener's virtual address space refers directly to the file's contents. This speeds access, and makes it possible for several clients to share the same copy of the file, which conserves the Web Listener's memory resources.

File Caching

Ordinarily, when the Web Listener opens a requested file, the file remains open and mapped into memory until all clients using the file are finished with it, at which point the Web Listener closes the file and frees the memory mapping associated with it. The Web Listener allows you to specify files to be cached. Cached files are opened when a client requests them, and remain open for the life of the Web Listener process. This optimizes access times for files, such as your home page, that are requested often.

File Protection

The Oracle Web Listener allows you to make secure specific virtual files or directories by assigning authentication and/or restriction schemes to protect them.

Authentication Schemes

When a file or directory is protected by an authentication scheme, a client requesting access to it must provide a user name and password. Authentication schemes allow you to define named groups of user name/password combinations, and named realms that are groups of these groups. You can then assign user, group, and realm names to virtual files and directories, requiring any client requesting access to input one of the specified user name/password combinations.

The Web Listener supports two authentication schemes: basic authentication and digest authentication. Both schemes are identical, except that digest authentication transmits passwords from client to server in an encrypted form called a digest, whereas basic authentication sends unencrypted passwords, making it considerably less secure. Some older web browsers don't support digest authentication, but for files or directories that require authentication, you should use digest authentication whenever possible.

Restriction Schemes

When a file or directory is protected by a restriction scheme, only a client accessing the Web Listener from a trusted group of host machines may access it. The two restriction schemes that the Web Listener supports are IP-based restriction and Domain-based restriction. IP-based restriction allows you to define groups of trusted hosts identified by IP address, whereas Domain-based restriction allows you to define groups of trusted hosts identified by DNS host or domain name.

File Format Negotiation

The Oracle Web Listener can maintain several versions of a document in different formats, and provide it to clients in the formats they prefer.

Language Formats

For example, if a client requests a document in French, the Web Listener checks to see if it has a French version of the document, and if so, returns that version to the client. Otherwise, the Web Listener returns the version of the document in its own default language (usually English).

MIME Formats

Similarly, a client can request a document of a particular Multipurpose Internet Mail Extensions (MIME) type. If the Web Listener can find a version of the requested file in the requested MIME format, it returns that version to the client. Otherwise, it returns the version of the file that has the smallest size.

Encodings

The Web Listener can also maintain versions of a document that have been encoded by programs such as a compression utilities. For example, if a client can uncompress a document that has been compressed by the gzip program, the Web Listener can return to the client the compressed version of document instead of the uncompressed document, saving transfer time.

Filename Extensions

The Web Listener uses filename extensions to identify a file's format, which can represent language, MIME type, or encoding. For ease of maintenance, it's best to advertise a file to clients only by its base name, allowing clients and server to negotiate formats transparently.

Dynamic Document Generation

Using the Oracle Web Listener, your web site can respond to client requests by generating HTML documents dynamically. This allows you to customize your WebServer's responses.

Like most HTTP engines, the Oracle Web Listener allows clients to use the Common Gateway Interface (CGI) to run programs on the server machine to perform special processing and return data to the client.

The Web Request Broker

Unlike other HTTP engines, the Oracle Web Listener provides an interface called the Oracle Web Request Broker (WRB), which allows clients to run programs on the server machine and return data much more efficiently than CGI allows. To do this, the Web Listener passes requests intended for these programs to the WRB Dispatcher, which maintains a pool of processes to which it can assign the requests. See The Web Request Broker (WRB) for more information.

The Secure Sockets Layer

The Secure Sockets Layer (SSL) is an emerging standard for secure data transmission over the Internet. One problem with communicating sensitive information over the Internet is that almost every connection between two computers over a network involves many intermediate steps—a chain of computers that successively receive and forward the information until it reaches its destination. This process, called routing, is fundamental to all Internet communication, and any computer in the routing chain has complete access to all the data it receives.

This makes it easy for the unscrupulous to intercept your private conversations, steal your credit card numbers, or illegally obtain confidential or proprietary information. The Oracle WebServer's implementation of SSL addresses this problem by scrambling data sent from the server to clients (web browser programs) in such a way that the clients can unscramble the information when they receive it. This way, any intermediate computers involved in routing the information see only gibberish that they can't decipher. This kind of security has three aspects:

1. Encryption—a mechanism for scrambling and unscrambling data.
2. Authentication—a mechanism by which the one party proves its identity to another party.
3. Data integrity—a mechanism for verifying that all of the data transmitted, and only the data transmitted, is received correctly.

Encryption

A traditional encryption system, called a secret-key system, uses a single large number called a key both to scramble (encrypt) and unscramble (decrypt) messages. Secret-key encryption systems are very fast, but they rely on one party communicating the secret key to another party, often by way of a third party such as a courier, before the two parties can exchange encrypted messages. This makes keys vulnerable to theft or tampering while in transit.

Public-Key Encryption

To avoid this problem, SSL uses a form of encryption called public-key encryption to encrypt and decrypt transmitted data. Unlike secret-key encryption systems, a public-key system uses pairs of keys (key pairs). One key, called the public key, is used to encrypt messages, while the other, called the private key, is used to decrypt messages. The two keys are large numbers that are related mathematically in such a way that it takes a very long time to calculate the private key from the public key.

If you want to receive encrypted messages using public-key encryption, you must first run a program that generates a key pair. You must then publish the public key in a public database or directory, and store the private key in a secure location on your computer. This is critical. The effectiveness of public-key encryption depends entirely on the secrecy of the private key.

Anyone who wants to send you an encrypted message must look up your public key in a directory, use it encrypt the message, and send you the encrypted message. Only your private key can decrypt the message, so if you have kept your private key secret, no one else can read this encrypted message.

Because public key encryption is much slower than secret-key encryption, SSL uses it only when the client first connects to the WebServer to exchange a secret key called a session key, which both client and server use to encrypt and decrypt transmitted data.

Authentication

Another application of encryption is authentication. Authentication using public-key encryption involves using a digital signature, an electronic proof of identity analogous to a handwritten signature.

If you want to "sign" an electronic document in a verifiable and legally binding way, you must first possess a key pair. You must then run a program that generates a digital signature using the private key and the document itself. You can then attach the digital signature to the document and send it. Anyone who receives this document, together with its digital signature, can then use the your public key to verify your identity, and to verify that the document has not been tampered with.

Certificates and Certifying Authorities

When clients connect to your web site for a transactions that require them to transmit sensitive information, they must be assured that they haven't connected to an impostor pretending to be you. Clients therefore require your WebServer to authenticate itself before such transactions can proceed.

To authenticate itself, your WebServer must present the client with the proper credentials, called a certificate. When you set up a secure WebServer, you must obtain a certificate from a trusted third-party company called a certifying authority (CA).
When you contact a certifying authority to request a certificate, you must provide them with certain legal information about your organization, which they can use to certify that your organization is legitimate and should be certified.

3.4.5. The Web Server Manager

To help you manage your Web site, the Oracle WebServer provides a set of Web pages that you can use to perform most common administration tasks. These pages are simply an easy way to edit the configuration files that the WebServer uses, so you can always use another tool to edit those files directly, although Oracle discourages this.

The WebServer Manager can be used from any Web browser. It has the following sections:

1. The Listener Pages
2. The WRB Pages
3. The PL/SQL Agent Pages
4. The Oracle7 Server Pages

The Listener Pages

The largest group of pages deal with administering the Web Listener. For the most part, however, these pages are just forms where you fill in the values for various configuration parameters. The exception is the security pages, where you specify which sorts of security schemes will be used, including the following:

1. If you select Restriction, the IP addresses or domain names that are granted various levels of access.
2. If you select Authentication, the user names and passwords that are granted various levels of access.

The WRB Pages

These pages let you specify the actual and virtual directories for WRB cartridges, as well as the number of WRBX's to assign to each.

The PL/SQL Agent Pages

Administering the PL/SQL Agent means administering the Database Connection Descriptors (DCDs) that it uses to establish its identity when communicating with the database.

The Oracle7 Server Pages

The Oracle7 Server is an extremely sophisticated product, and it is properly administered using Oracle Server Manager or directly through the SQL language. The pages provided here enable you to do a few basic things, such as start up and shut down database instances, and browse database contents.

3.4.6. The CGI Interface

The Common Gateway Interface (CGI) is the standard technique used by an HTTP server to execute a program that generates HTML output. Using CGI, you can run PL/SQL, and thus interface to the Oracle7 Server, from Internet servers that do not support the WRB. This technique provides dynamic content rather than static content from files on disk. Oracle WebServer is fully compliant with CGI version 1.1.

When the Web Listener recognizes an incoming URL as a request to execute a CGI application, it spawns a separate process to perform the operation (The Web Request

Broker (WRB) circumvents this need to spawn a new process for each request, thereby improving performance). The Web Listener passes the URL to this process and also maintains communication with it through standard input and output. Therefore, the CGI process can get the input it needs from the URL itself and/or the standard input. It sends its output back to the Listener through the standard output, and the Listener transmits it in turn to the client's Web browser.

The fact that CGI applications spawn a new process each time they are used is costly in terms of performance. For this among other reasons, Oracle recommends that you use the WRB instead.

3.4.7. The Web Request Broker (WRB)

The WRB (Web Request Broker) is an asynchronous request handler with an API (Application Program Interface) that enables it to interface dynamically and seamlessly to various back-end technologies called "WRB Services". It consists of a WRB Dispatcher that allocates Service requests among several running processes known as the WRBX's. The result is that the Listener is free to receive and validate URLs coming in, while each request is handed off to a process that executes it in the background. The WRB API is designed so that third parties can write their own cartridges to extend the WebServer.

Whenever it receives a URL that calls for the WRB, the Web Listener passes execution of the request to the WRB Dispatcher or simply Dispatcher. The Dispatcher maintains communication with a pool of processes called WRB Executable Engines

(WRBX's). Each WRBX interfaces, through the WRB API, to a WRB cartridge, which can be of the following types:

1. The PL/SQL Agent. This cartridge executes PL/SQL commands stored in the database. It is better optimized for database access than the Java cartridge, but doesn't have all of Java's functionality.

2. The Java™ Interpreter. This cartridge lets you execute Java on the server to generate dynamic Web pages. You can also execute PL/SQL from within Java using this cartridge.

3. The LiveHTML Interpreter. This cartridge is Oracle's implementation and extension of the industry-standard Server Side Includes functionality. LiveHTML enables you to include in your Web pages the output of any program that your Operating System can execute.

The combination of a cartridge and its WRB API constitute a WRB Service. WRBX's are instances of WRB Services, so that there are three WRB Services corresponding to the three WRB cartridges, while the WRBX's are created and destroyed as needed to handle the workload.

The Dispatcher creates and maintains the WRBX's. The data passed to a WRBX consists of:

1. The URL triggering the request.

2. The desired language of the result.

3. The desired character set of the result (how the language is encoded, for example, ISO or Unicode)

4. CGI environment variables, which allow WRB cartridges to be run by applications written for CGI.

5. If the request involves the use of the PL/SQL Agent, which of the Database Connection Descriptors (DCDs) to use.

The Dispatcher continually adjusts the load by controlling how many WRBX's are running at a given time, subject to certain parameters. These parameters are set by the

WebServer Administrator, who decides the maximum and minimum number of WRBX's running. The Dispatcher creates new WRBX's as needed and connects them to the appropriate WRB Services. The Dispatcher keeps track of which WRBX's are executing requests and which are free, and periodically checks the WRBX's to see how long they have been idle. If a WRBX's idle time is beyond the maximum set by the WebServer Administrator, that WRBX is killed. The WRBX's assigned to a given WRB Service always have requests assigned in the same order, so that traffic will be concentrated in a small number of them, and most of them will not be idle. The WRBX's are single-threaded processes that communicate with the WRB Dispatcher using the dataflow mechanism appropriate to the Operating System (such as a pipe in Unix).

3.4.8 PL/SQL Agent

WRBX's interfaced to the PL/SQL Agent connect to the database when they start up, so that the execution of requests can proceed more quickly. The WRBX's connect to the database schemas specified in their configuration files.

Specifying the Use of PL/SQL Agents

If the URL for a given request contains the string "owa", properly placed, the Listener fulfils the request using the PL/SQL Agent. Whether this is done through the WRB or through CGI depends on how the WebServer Administrator has configured the directory mappings. The PL/SQL Agent executes application code written in PL/SQL and returns the output in HTML form for the Web Listener to output as a Web page.

When a URL is received for the PL/SQL Agent, it contains a DCD (Database Connection Descriptor). The DCD determines both the database access privileges the PL/SQL Agent has when executing this request and the schema (portion of the database) that it accesses.

PL/SQL procedures are stored in the database. The PL/SQL Agent invokes these by issuing commands to the database, which then performs the actual execution and sends the output and status messages back to the PL/SQL Agent.

Oracle WebServer also provides you with Java classes that can invoke PL/SQL.

The PL/SQL Developer's Toolkit

To make it easier for you to develop Web applications using Oracle data, Oracle WebServer provides you a group of PL/SQL packages that you can use to easily generate Web pages from data stored in an Oracle database. These packages are called the PL/SQL Developer's Toolkit. The intent is for you to create PL/SQL procedures that access and process the Oracle data you wish to place on the Web. From within these procedures, you call the PL/SQL Developer's Toolkit procedures you need to create the HTML you want. You store the procedures you write in the database, just as other PL/SQL packages, including the tookit, are stored. You also design your Web pages, including the dynamically-generated ones, to produce URLs that call the PL/SQL procedures you want in response to specified user actions. Having your code executed within the database brings many performance, security, and portability benefits.

3.4.9 Java

Java is an object-oriented language for creating distributed applications on the Internet or other networks. Modules of Java code known as "applets" can be downloaded from the Internet or a local network in real time and locally executed. Java applets themselves can call and execute other applets, so that a Java application as executed on the user's machine can be constructed "on the fly" from a repertoire of standard parts that reside on the net. You might call this the "building block" approach to programming. Among the important features of Java are the following:

1. It is extremely portable. Java code is compiled to a form known as "bytecode". This is a sort of generalized computer code that is not executable by any particular machine, but is recognized by the "Java Virtual Machine". It is as bytecode that Java applets transverse the net. The Java Virtual Machine (VM) resides on the computer where the applet is to be executed and converts the bytecode to the native code for that machine. Currently, Java VMs exist or are planned for all current versions of Windows, Solaris, MacOS, OS/2, Linux, Amiga, and other platforms.

2. It is fully object-oriented. Languages like C++ that add object-oriented features to non-object-oriented languages. Java applets do not allow violation of object-oriented principles such as "encapsulation".

3. It uses a C-like syntax. This makes the language easier for C programmers to learn.

4. It is multi-threaded, in effect executing several chains of control flow concurrently. The Java language itself provides tools for managing the threads, rather than relying exclusively on the OS.

5. It prohibits direct memory manipulation. In Java, there are no pointers and no direct memory allocation. This eliminates a rich source of C's functionality, and an even richer source of its bugs.

6. You can embed calls to Java applets in Web pages, and the applet will be executed by the browser, provided it is Java-enabled. Most major browsers plan to support Java.

Oracle WebServer supports Java because of its rich features and wide acceptance.

Client vs. Server Side Java

Oracle WebServer supports the use of Java either on the client, which is to say any Java-enabled Web browser, or on the server. Code to be executed on the client is for the most part extracted and manipulated like other data. The best way to handle such code is to store it in the OS file system and extract it in real time.

You can also execute Java as a WRB cartridge on the WebServer itself. You might want to do this, for example, to perform graphical manipulation for which PL/ SQL is ill-suited. For example, you can combine several graphics from the database into a single image. Each region of the image would be a separate button that the user can click, and each button clicked would produce a different effect. In HTML, this is called an "image map". Using Java on the server, you could generate such image maps dynamically, with the components of the image being based on the results of a database query.

To execute Java on the server, you use the WRB API to interface directly to the Java Interpreter residing in the WebServer. This interpreter finds and executes the Java code and returns the results, through the WRB interface, to the Web Listener.

Using PL/SQL Within Java

Since PL/SQL code is actually part of the database, you can call it from within Java, which enables you to create applications that combine the strengths of both languages. Because PL/SQL execution takes places in the database, doing this does not hinder the portability of the application. A PL/SQL application can execute without modification on any platform where the Oracle7 Server runs, just as a Java applet can execute without modification on any platform that has a Java Virtual Machine (VM).

3.4.10 LiveHTML

LiveHTML is Oracle's implementation and extension of the standard Server Side Include functionality defined by the NCSA. The LiveHTML Interpreter enables you to include dynamic content in otherwise static Web pages. At the point in your Web page where you want to interject dynamic content, you place a tag that points to one of the following:

1. A static Web page.
2. Another LiveHTML Web page.
3. A script that is executed on the server and outputs HTML. This script may but need not conform to the CGI standard.
4. A system variable, for example: FMODDATE.

You can use a variable to determine at runtime the Web page, variable, or script to which the tag points. This enables you to have a Web page that selects dynamically from among any number of static Web pages or scripts, based, for example, on values a user provides in an HTML form. The result is a dynamic Web page built of static Web page components, variables, and HTML output from scripts.

A Web page that is to use LiveHTML must be parsed by the WebServer. For this reason, it differs slightly from ordinary Web pages written in HTML, which the WebServer simply delivers to the browser. To have the LiveHTML tags executed on the server, you must use the WebServer Manager to specify that a given Web Listener is to parse files for LiveHTML You have the option of having the Listener parse all files or just those with certain extensions.

Enabling users to execute scripts on the server can create security and other risks. For this reason, you can specify that a specific Listener allows only "crippled" includes. This means that LiveHTML parsed by that Listener will be able only to call static HTML, environment variables, or other server parsable files, not executable scripts.

You frequently use LiveHTML when you have standard components, such as menus, that you want on many pages. If desired, you can use LiveHTML to run the PL/SQL Agent under CGI and thereby incorporate dynamic Oracle data in hardcoded Web pages.

CHAPTER 4

DESIGN FOR INTENET BASED EDUCATION FRAMEWORK

4.1 Introduction

This chapter deals with the life cycle of the development involved in this project. The initial requirements were gathered with meetings held with Dr Tanik and Prof. Jololian. Though the scope of the work is immense, I have attempted to complete as much functionality as possible. The remaining part can be scoped out for future development. We considered the Oracle Web Developer suite as the set of tools in this project. This suite can help us in every set of the development as well as the actual presentation of the forms to the user. The database is an essential component of this system in order to provide dynamic functionality to the system. The data in the database could be manipulated without affecting the forms that the user would see. Also since reliability and robustness of the database is a very important factor the Oracle database engine is an optimum choice for this project

4.2 Requirement Specification

The requirements of this project are as follows:

1. Student should be able to select department and the course.
2. Student can see the information on any course selected.
3. To enter a particular course, Student should login.
4. Student be able to login from the web.
5. Student should be validated.

6. Student should then be able to see content and the submissions.
7. In content student should be able to see Notes, Articles, Audio, Video.
8. In submissions, student should be able to see Exams, Exercises, Projects.
9. Professor should be able to see what the student sees.

4.3 Hardware and Software Requirement

The hardware and software requirements for the system on the server is as follows:

Hardware:

Pentium 166 MHz PC with

64MB RAM, 2 GB of available hard disk

TCP/IP connectivity and access to the internet

Software

Windows NT Operation System

Oracle 7.3.3 Database

Oracle Web Server 2.1

Oracle Developer 2000 Version 1.5W

Oracle Designer 2000

The *hardware* for the client or student PC could be a

486 PC or a Pentium PC with about 16MB of Memory

Internet Connectivity should be present which includes modem, telephone connection and connectivity to the world wide web possibly through an internet solution provider(ISP)

Software Required would be an internet browser like Netscape Navigator or Microsoft Internet Explorer.

4.4 Conceptual Model of the Framework

Figure 4.1 indicates the student model of the system. The student will connect to the Internet using his PC and Modem. Then he can put in the address of the education system web server. Once connected the system will permit the user to see information on the selected course. If the user needs to proceed further, He must be registered in the course. The system will now prompt the user to enter the userid and password. If the user is a valid student for the selected course, the system will allow the user to proceed further or he will be returned to the main course selection form.

On the other hand, if the user is a valid student then the student will now get the options:

(1) View content

(2) View submission

When the student selects View Content, the student will be made to select the educational unit. Once this is done, the student will get all the content associated with that educational unit for that particular course. The student can select whichever content he wants and view it. Upon completion of the viewing the student is can navigate back to the previous screen to select either mode content associated with this educational unit or he can select another educational unit for viewing and view the contents associated with that educational unit.

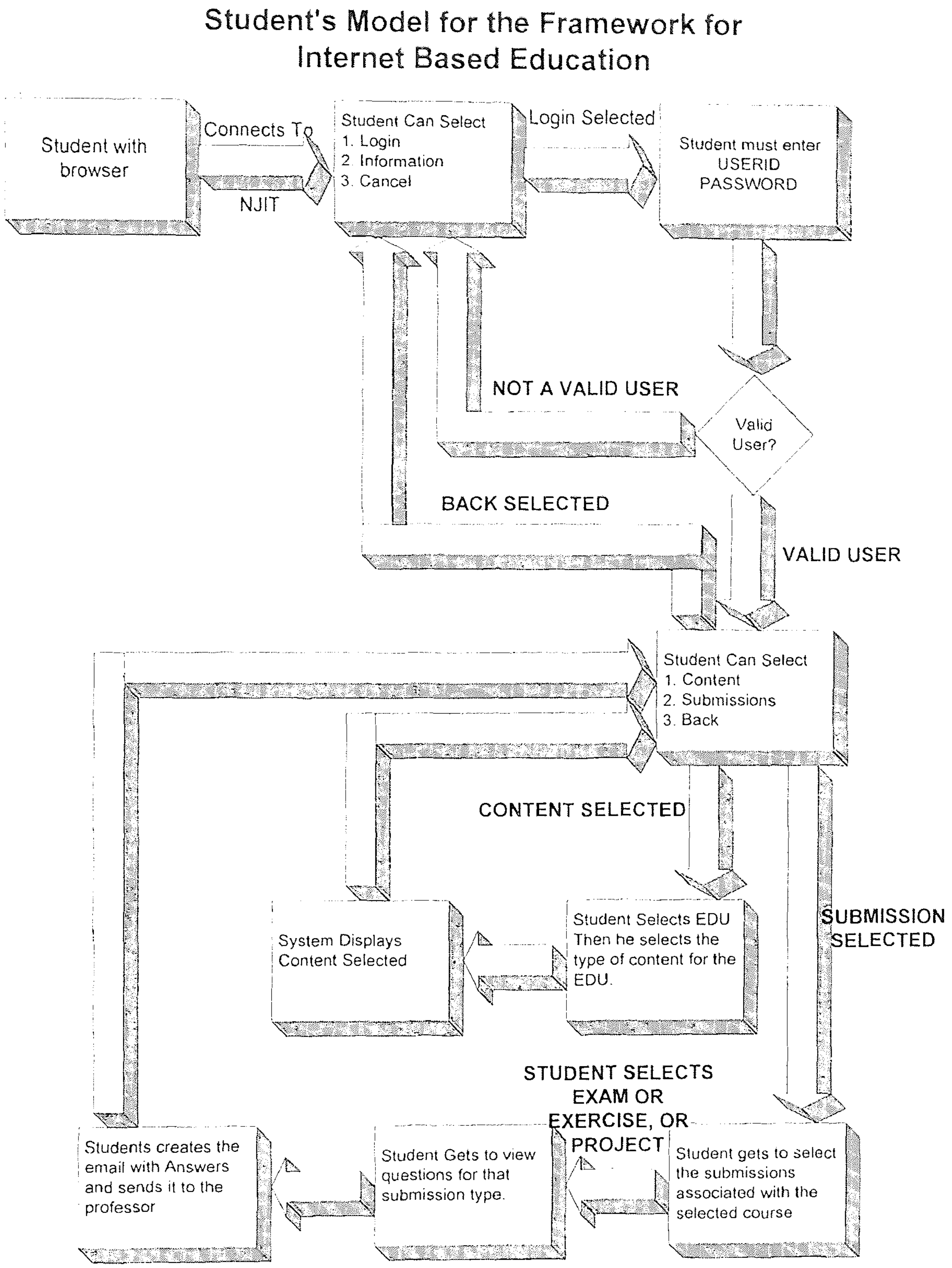

Student's Model for the Framework for Internet Based Education
Student with browser
Connects To
NJIT
Student Can Select
1. Login
2. Information
3. Cancel
Login Selected
Student must enter USERID PASSWORD
NOT A VALID USER
Valid User?
BACK SELECTED
VALID USER
Student Can Select
1. Content
2. Submissions
3. Back
CONTENT SELECTED
System Displays Content Selected
Student Selects EDU Then he selects the type of content for the EDU.
SUBMISSION SELECTED
STUDENT SELECTS EXAM OR EXERCISE, OR PROJECT
Students creates the email with Answers and sends it to the professor
Student Gets to view questions for that submission type.
Student gets to select the submissions associated with the selected course

Figure 4.1 Student's Model

When the student selects View Submission, the student will get options to select exams, exercises or projects. Once he/she selects one of these, the system will display a list of the available submissions in this category. The student can select one of these and he will see the questions pertaining to that particular submission. The email address of the concerned professor will be provided as an option. The student can then email his submission to the specified address. The student will receive and email from the professor after the professor has viewed his submission.

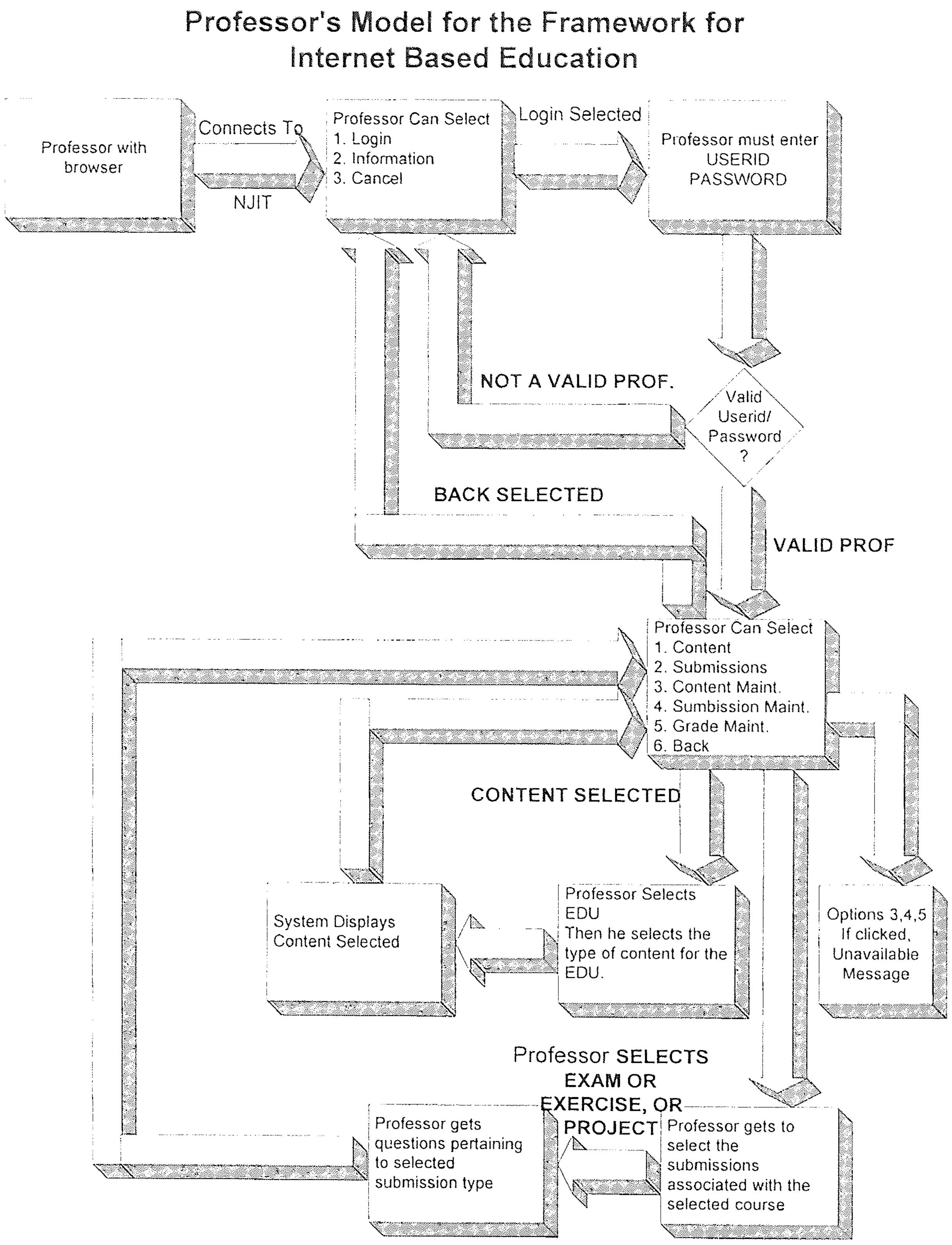

Professor's Model for the Framework for Internet Based Education
Professor with browser
Connects To
NJIT
Professor Can Select
1. Login
2. Information
3. Cancel
Login Selected
Professor must enter USERID PASSWORD
Valid Userid/ Password ?
NOT A VALID PROF.
BACK SELECTED
VALID PROF
Professor Can Select
1. Content
2. Submissions
3. Content Maint.
4. Sumbission Maint.
5. Grade Maint.
6. Back
CONTENT SELECTED
System Displays Content Selected
Professor Selects EDU Then he selects the type of content for the EDU.
Options 3,4,5 If clicked, Unavailable Message
Professor SELECTS EXAM OR EXERCISE, OR PROJECT
Professor gets questions pertaining to selected submission type
Professor gets to select the submissions associated with the selected course

Figure 4.2 Professor's Model

Figure 4.2 Indicates the professor's model of the system. Just as in the student's case the professor follows the path as indicated herein. The professor will connect to the Internet using his PC and Modem. Then he can put in the address of the education system web server. Once connected the system will permit the professor to see information on the selected course. If the professor needs to proceed further, he must login into the course. The system will now prompt the professor to enter the userid and password. If the professor is a has a valid account for the selected course, the system will allow him/her to proceed further or he/she will be returned to the main course selection form.

On the other hand, if the professor has a valid account then the student will now get the options:

(1) View content

(2) View submission

(3) Maintain Content

(4) Maintain Submission

(5) Maintain Grade

When the professor selects View Content, he/she will be made to select the educational unit. Once this is done, he/she will get all the content associated with that educational unit for that particular course. The professor can select whichever content he/she wants and view it. Upon completion of the viewing the professor can navigate back to the previous screen to select either mode content associated with this educational unit or he/she can select another educational unit for viewing and view the contents associated with that educational unit.

When the professor selects View Submission, he/she will get options to select exams, exercises or projects. Once he/she selects one of these, the system will display a list of the available submissions in this category. The professor can select one of these and he will see the questions pertaining to that particular submission.

The Maintain Content option will help the professor to maintain content associated with this course.

The Maintain Submission option will help the professor to maintain submissions associated with this course. This will be exams, exercises and projects and their associated questions

The Maintain Grade option will help the professor to maintain marks associated with the submissions in this course.

4.5 Database Design

In performing the database design care has been taken to follow the Third Normal form for database modeling. The three normal forms for database design are follows:

1. Ensure that each field represents the smallest indivisible element.
2. Keep all fields related to the primary key together in the table
3. Take away all fields that do not depend only on the primary key.

The Oracle Designer 2000, a very powerful tool, was used for the database analysis and design. The entity relationship(ER) model was first designed. The following were considered as entities to form the tables.

1. DEPARTMENTS
2. EMPLOYEES
3. STUDENTS
4. COURSES
5. COURSES_EMPLOYEES
6. EDU_UNITS
7. NOTES
8. ARTICLES
9. AUDIOS
10. VIDEOS
11. EXAMS
12. EXERCISES
13. PROJECTS
14. EXAM_QUESTION
15. EXERCISE_QUESTION
16. PROJECT_ QUESTION
17. MAILS

Table 4.1 Department

Field Name	Data Type	Size	Key	Not Null	Referencing
ID	Varchar2	10	PRIMARY	Not Null	
NAME	Varchar2	25		Not Null	

Reasoning: The department has and ID and a NAME. For example CIS is the department ID and Computer and Information Sciences is the Name.

Table 4.2 Employee

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Varchar2	9	PRIMARY	Not Null	
Name	Varchar2	25		Not Null	
Address	Varchar2	40			
Email_ address	Varchar2	20			
Designation	Varchar2	25			
Password	Varchar2	10			
Depa_id	Varchar2	10			Department
Userid		10			

Reasoning: Each employee has an ID. This is the primary key. Associated with the Id we have the name, address, email address, designation, password and userid. All these are particulars of the employee. The employee belongs to a department. Hence the Department ID is included as a foreign key.

Table 4.3 Student

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Varchar2	9	PRIMARY	Not Null	
Name	Varchar2	25		Not Null	
Address	Varchar2	40			
Email_ address	Varchar2	20			
Class	Varchar2	3			
Grade	Varchar2	2			
Gpa	Number	4,2			
Password	Varchar2	10			
Depa_id	Varchar2	10			Department
Userid		10			

Reasoning: Each student has an ID. This is the primary key. Associated with the Id we have the name, address, email address, password and userid. All these are particulars of the student. The student belongs to a department. Hence the Department ID is included as a foreign key. Also included are the class, grade and GPA of the student.

Table 4.4 Courses

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Varchar2	10	PRIMARY	Not Null	
Description	Varchar2	25		Not Null	
Semester	Varchar2	15		Not Null	
Class	Varchar2	3		Not Null	
Depa_id	Varchar2	10		Not Null	
Information	Long				

Reasoning:

Each course has an ID. This is the primary key. There is a description to this course. Each course is associated with a department, semester. Information field is included which can contain important information that needs to be displayed as the bulletin. For example CIS601001 Object Oriented Programming Spring 1998 MS CIS information.

Table 4.5 Courses_employees

Field Name	Data Type	Size	Key	Not Null	Referencing
Course_id	Varchar2	10		Not Null	
Empl_id	Varchar2	9		Not Null	

Reasoning:

This table consists of all professors or employees associated with a course.

Table 4.6 Edu_units

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Edu_unit	Varchar2	10		Not Null	
Description	Varchar2	200		Not Null	
Notes	Varchar2	1			
Articles	Varchar2	1			
Video	Varchar2	1			
Audio	Varchar2	1			
Course_id	Varchar2	10			Course

Reasoning:

The EDU_UNIT is like a chapter in a course. Each EDU_UNIT has an ID and Description. Associated with each EDU_UNIT is notes, articles, audio and video. Since the EDU_UNIT belongs to a course, the COURSE_ID is included as a foreign key in Table 4.6.

Table 4.7 Notes

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	200		Not Null	
Notes	Long				
Display_ date	Date				
Hide_date	Date				
Edu_unit	Varchar2	10		Not Null	

Reasoning:

Notes is one of the content which the user views. This will have an id and a description along with the actual note. Display date and hide date are included in order to enable display of notes based on the dates. Since the Note itself belongs to an EDU_UNIT the associated EDU_UNIT is included in this design as a foreign key.

Table 4.8 Articles

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	200		Not Null	
Articles	Long				
Display_ date	Date				
Hide_date	Date				
Edu_unit	Varchar2	10		Not Null	

Reasoning:

Articles is one of the content which the user views. This will have an id and a description along with the actual article. Display date and hide date are included in order to enable display of articles based on the dates. Since the article itself belongs to an EDU_UNIT the associated EDU_UNIT is included in this design as a foreign key.

Table 4.9 Audios

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	200		Not Null	
Audio	Long				
Display_ date	Date				
Hide_date	Date				
Edu_unit	Varchar2	10		Not Null	

Reasoning:

Audio is one of the content which the user views. This will have an id and a description along with the actual audio. Display date and hide date are included in order to enable display of audio based on the dates. Since the audio file itself belongs to an EDU_UNIT the associated EDU_UNIT is included in this design as a foreign key.

Table 4.10 Videos

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	200		Not Null	
Video	Long				
Display_ date	Date				
Hide_date	Date				
Edu_unit	Varchar2	10		Not Null	

Reasoning:

Video is one of the content which the user views. This will have an id and a description along with the actual video. Display date and hide date are included in order to enable display of video based on the dates. Since the video file itself belongs to an EDU_UNIT the associated EDU_UNIT is included in this design as a foreign key.

Table 4.11 Exams

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	2000		Not Null	
Semester	Varchar2	15		Not Null	
Display_ date	Date				
Hide_date	Date				
Course_id	Varchar2	10			Course

Reasoning:

Exams is one of the types of submission. Each exam will have an ID, description and semester. Display Date and Hide Date are included to enable display of the exam based on date. Since this exam belongs to a course, the course id is included as a foreign key.

Table 4.12 Exercises

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	2000		Not Null	
Semester	Varchar2	15		Not Null	
Display_ date	Date				
Hide_date	Date				
Course_id	Varchar2	10			Course

Reasoning:

Exercises is one of the types of submission. Each exercise will have an ID, description and semester. Display Date and Hide Date are included to enable display of the exercise based on date. Since this exercise belongs to a course, the course id is included as a foreign key.

Table 4.13 Projects

Field Name	Data Type	Size	Key	Not Null	Referencing
Id	Number	10	PRIMARY	Not Null	
Description	Varchar2	2000		Not Null	
Semester	Varchar2	15		Not Null	
Display_ date	Date				
Hide_date	Date				
Course_id	Varchar2	10			Course

Reasoning:

Projects is one of the types of submission. Each project will have an ID, description and semester. Display Date and Hide Date are included to enable display of the project based on date. Since this project belongs to a course, the course id is included as a foreign key.

Table 4.14 Exam_questions

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Question	Long			Not Null	
Exam_id	Number	10		Not Null	

Reasoning:

This table contains the questions associated with the exam. This table is the detail table associated with the master table which is the EXAMS table. Hence Exam ID is included as a foreign key in Table 4.14.

Table 4.15 Exercise_question

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Question	Long			Not Null	
Exercise_id	Number	10		Not Null	

Reasoning:

Table 4.15 contains the questions associated with the exercise. This table is the detail table associated with the master table which is the EXERCISES table. Hence EXERCISE ID is included as a foreign key in Table 4.15.

Table 4.16 Project_question

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Question	Long			Not Null	
Project_id	Number	10		Not Null	

Reasoning:

Table 4.16 contains the questions associated with the project. This table is the detail table associated with the master table which is the PROJECTS table. Hence PROJECT ID is included as a foreign key in this table.

Table 4.17 Mails

Field Name	**Data Type**	**Size**	**Key**	**Not Null**	**Referencing**
Id	Number	10	PRIMARY	Not Null	
Mail_from	Varchar2	9		Not Null	
Mail_to	Varchar2	9		Not Null	
Subject	Varchar2	2000		Not Null	
Message	Long			Not Null	
Empl_id	Varchar2	9			Employee
Empl_id_ Receives	Varchar2	9			Employee
Student_id	Varchar2	9			Student
Student_ receives	Varchar2	9			Student

Reasoning:

Table 4.17 is included in the design in order to provide a repository of the email messages that would be going from the student to the professor and vice versa.

Entity Relationship Model in Designer 2000

Keeping the mentioned reasoning in mind, the entity relationship model was designed using the Designer 2000. This Entity Relationship Diagrammer permits of drag and drop style entities and their relationships. It also allows assigning datatypes to the fields, setting of primary keys, setting up of hints, minimum and maximum values for the field, setting up indexes. In terms of relationships it allows creation of one to many, many to one, many to many and one to one relationship along with the criteria of optionality whether must have, or may have and so on. Appendix A lists the actual screen prints of the entity relationship model that was designed for this education system project.
The Entity Relationship Mapper performs the task of mapping the entity - relation diagram to logical objects.

The Data Diagrammer takes these mapped objects and creates a database schema with the mapped entities and the relationships. The Data Diagrammer schema includes the actual tables, the fields and the relationships in terms of primary and foreign keys. The representation is a diagram which indicates all these database items. It is possible to make changes to this diagram and reverse engineer the changes back to the Entity Relationship Diagrammer. This diagrammer enables the creation of SQL code to actually create the tables, indexes, clusters, primary and foreign keys. You can view the code once this is created and this can be saved to a file. This code can be executed from the diagrammer or can be done offline through Oracle SQLPLUS. Appendix A has screen prints for the Data Diagrammer with the tables and the relationships. The code generation, after code generation screen is also included in the appendix. The SQL code to create the tables is included in the Appendix A. Though I have experimented with all the other modules the Designer 2000, in this project I have used only these 3 modules.

Physical Database

The SQL code which is also the DDL or data definition language code which was created by Oracle Designer 2000 was taken and this was executed from Oracle SQL *PLUS. This code created the actual tables in Oracle along with the primary and foreign keys. Dummy data was then entered into the tables. All the data pertaining to the project is stored in this Oracle repository. Queries executed from SQL*PLUS show various aspects of the data and the database structure. Again, as mentioned before, care was taken to adhere to 3 rd Normal Form of Database Modeling. The tables were created in the name of a Oracle user. Then synonyms were created to provide access to other users.

4.6 Forms Design

Oracle Developer 2000 1.5W   was used to create forms. This tool has Forms Designer which is extensively used to create the forms. The form is the graphical user interface to the data. Though this form will be in windows client/server mode on design, Oracle Developer permits the display of these forms on the World Wide Web via the Oracle Web Server. This tool is a very versatile tool and is very widely used in the industry

Forms Logic

The following section describes the logic that was used in the design of the graphical user interface forms.

In the Opening form the user will see a list of buttons on the left side . These will be

1. Information
2. Login
3. Exit.

On the right hand side he will see a list box showing the various departments. The user , in this case could be a student or the professor. Let us assume that a student the system. The system prompts the user to select the department and click on the retrieve course button to retrieve courses for that department. The student selects the department, clicks on the retrieve button. Automatically the system will retrieve all courses in the system pertaining to that department. The student is now prompted to select the course and then click on the login button. When the student clicks on the LOGIN button the system displays a form to validate the user. The user has to enter his name and password. The system checks if the user is already present in the system and if he is not the system will not allow him to proceed further in the system

Validation

The validation is performed as follows:

The system first checks if any records in the student table match the entered userid and password. If yes, then it checks if the student is registered for this course by checking the STUDENT_DETAIL table where there exists a record for this student for this semester with the commence date. If the student is found there also then the student is considered valid and permitted to proceed further. If the student is not found, then the system checks if the userid and password match a professor by checking if there exists a record in the employee.table with this userid and password. If yes, then a check is performed against the COURSE_EMPLOYEE table and checked if the professor for this course is the specified professor. If yes then the system recognizes that a professor has logged in and gives the professor the respective screens.

Navigation

For a valid user, after the logon, the Course_Details button becomes active. On clicking this he can proceed further into the system. This screen displays 3 buttons for the student on the left side. On the right side the student gets the information bulletin. The professor can put any information in the information field of the course tablewhich appears on this screen which the student will definitely read when he comes to this screen.

The three options are as follows:

(i) View Content

(ii) View Submissions

(iii) Back

View Content will take the user to another form where he gets a list of the educational units associated with this course. The student is prompted to select an educational unit and click on the Retrieve Content button. Upon doing this the user will get buttons that represent the form of the content. For example if notes is present for this educational unit, then the notes button will become active. If the articles is present for this educational unit then the articles button will become active, and so on. When the user clicks on the notes button he/she will get a list of all the notes existing in the system for this particular educational unit. The user can then click and select the concerned unit and click retrieve content. This will then take the user to another form with buttons to go back or to the previous screen. This display of the note is in a scrollable edit box. This is the same with the articles.

View Submissions will take the user to another form where on the left side he will see buttons associated with the submission types. If projects are available for this course, then the List Projects button becomes active. Similarly, if exams are present then the List Exams button becomes active. Same is the case with Exercises. When the user clicks on one of these, say the exams, the user gets a list of all the exams associated with this course. The user clicks and selects one of these and then clicks on Retrieve Submission button. This will display another form containing the actual submission questions along with buttons to navigate between the questions. Note that this system will only display the questions. It is expected that the student email the answers to the professor separately.

The email address is provided as an option button on the previous screen. The objective behind this reasoning was to permit the student to use free form means of creating his answer sheet and this be delinked from the database as the size of the files could be very big and would unnecessarily slow down the system with these large objects in the database.

In the case of the professor, he/she will see all these that the student sees. In addition on the bulletin screen, the professor gets three options. These are

1. Maintain Content
2. Maintain Submission
3. Maintain Grade

4.7 Display of the System on the World Wide Web

Developer 2000 provides for these forms to be able to viewable form the world wide web. An initial starting HTML is required which will start this application. This application is the start template provided by Oracle Corporation. Once this hyper text markup language(HTML) page is executed from the browser, the educational system will start showing up. Thereafter it is just following the navigation as described in the previous discussion. These form could be viewed by Netscape Navigator or by Microsoft Internet Explorer browser.

4.8 Coding

All coding for this project has been done Oracle Procedural Language/Structured Query Language (PL/SQL). Triggers, procedures perform the desired functionality. The forms were designed in Oracle Forms Designer which provides a Visual Interface to design the forms. The code developed for this project is attached in the Appendix A.

CHAPTER 5

CONCLUSIONS

Education on the web is a reality. Every school and college must have a program to present educative material via the internet. The reach of this system is enormous and has immense potential. A lot of schools like Arizona State University have some web pages with courses on them. The strength of the system will be in the user friendliness of the system or the ease of use to the system to the student and the professor, and the ability to store large amounts of different data in the system.

The development of the software was done using the Oracle database 7.3.3 and Oracle tools Designer/2000 version 1.3.2, Developer/2000 version 1.5w and Web Server version2.1. There is a very steep learning curve for Oracle Designer/2000. Once this was overcome the database related aspects of the tool were thoroughly utilized. Developer/2000 was utilized to create all the graphical user interface screens and was excellent in creating the forms. The Web Server was utilized to connect the student to the system. The system we developed was user friendly and has a lot of capacity to save large amounts of data.

CHAPTER 6

FUTURE DEVELOPMENTAL SCOPE

Future development in the region can be scoped out as follows:

1. Maintenance screens For Content for the Professor
2. Maintenance screens for submission for the professor
3. Maintenance screen for grades

6.1 Maintenance Screen for Content

The conceptual design would be as follows:

When the professor selects the option, the system will prompt him with educational unit maintenance or content maintenance. If he selects educational unit maintenance, the system will display the list of educational units and the professor can select any one of them to edit. Alternatively the professor may choose to add a new one or delete an existing one. The system should be smart enough to prevent accidental deletion of the educational unit which already has content associated with it. An override may be provided if the professor is sure that he wants to delete the educational unit and all contents associated with it.

If the professor selects content maintenance then the system prompts him/her to select the educational unit. When he selects this, the system will display the types of contents in the form of buttons. These could be notes, articles, video and audio. When the professor selects any one of them, the system will display all the contents for that particular type. The professor now, may edit the existing content, add a new content, or delete existing content or he might just exit out of content maintenance.

Actual maintenance screens would be required for each of the contents and the educational unit maintenance.

6.2 Maintenance Screen for Submission

The conceptual design would be as follows:

When the professor selects the option, the system will prompt him with the submission type. The professor can select one of them. Then the professor gets the options of NEW, EDIT, DELETE, CANCEL. When he selects NEW, then the system will prompt the professor to enter the description for that submission and will give the professor fields to enter the questions. The system will now prompt the professor the options of COMMIT or CANCEL. When he/she commits, the system will commit the data and it will be available when the student next time comes into the system. When the professor selects EDIT, the system will display the currently available set of papers for this submission type. Then he/she can select one of these and the system will display the questions pertaining to this. The professor may add, edit, delete any question or he may just exit. When the professor selects DELETE the system will display the currently available list of papers for this submission type and he/she can select any one of this list and perform the delete. If the professor selects CANCEL, he/she is taken out of this submission maintenance option. This would be the case for exams, exercises or projects. Actual maintenance screens would be required for the questions of each submission type, submission type itself and navigation screens.

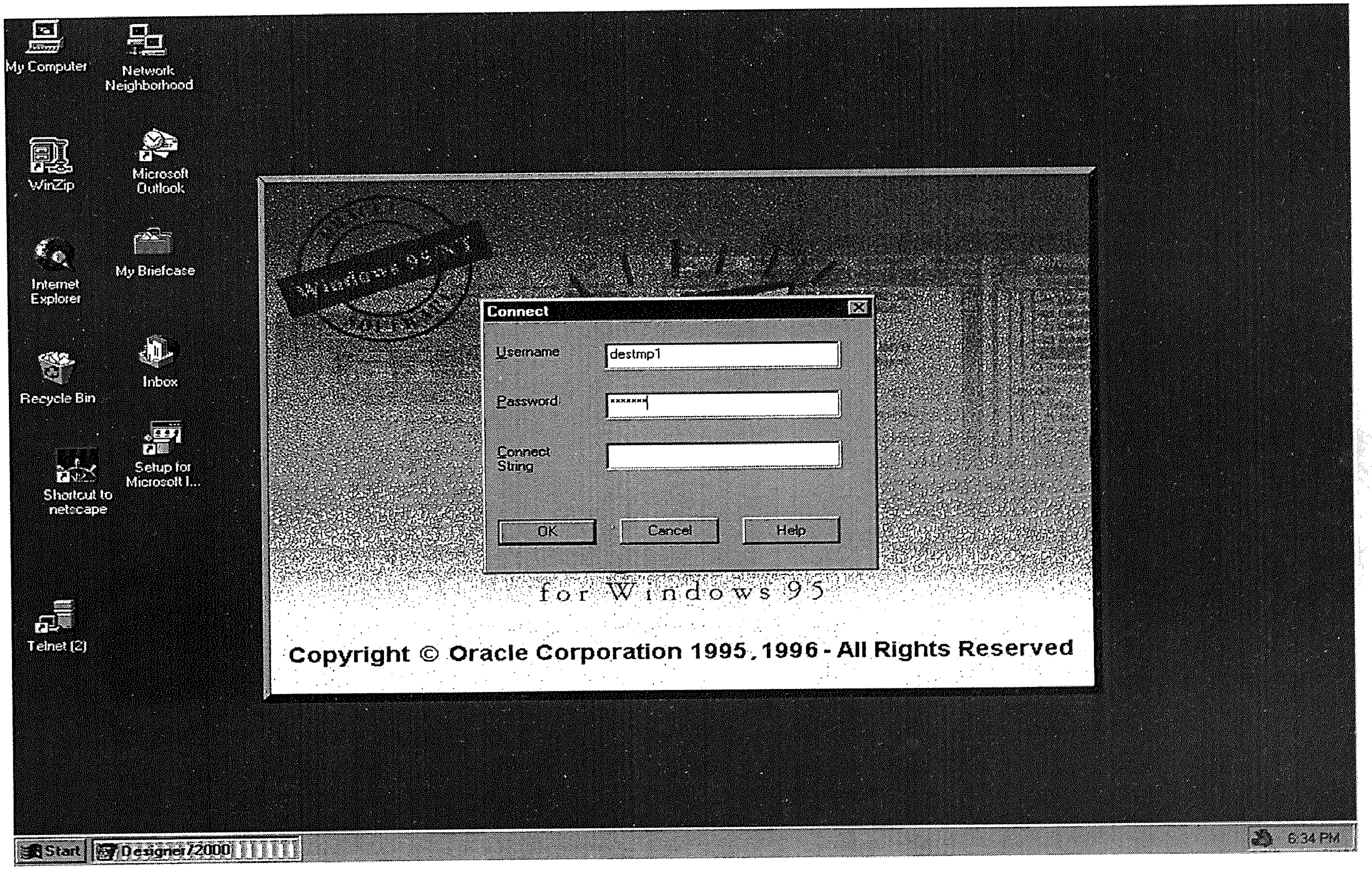
6.3 Maintenance Screen for Grade

When the professor selects this, the system will prompt him/her to select the submission type. Then the student is selected and the system will prompt the professor to enter marks for each question in that particular submission type.

There will be an option to submit the final grades, when the student is done with the course. This will trigger the record pertaining to this student, this course to get updated with the correct grade. Maintenance screens for the grade will be required along with the proper navigation and trigger events.

APPENDIX A

Screens And DDL Code



Username: destmp1

Password: *******

Connect String:

Copyright © Oracle Corporation 1995, 1996 - All Rights Reserved

Figure APP1. Designer/2000 Login Screen

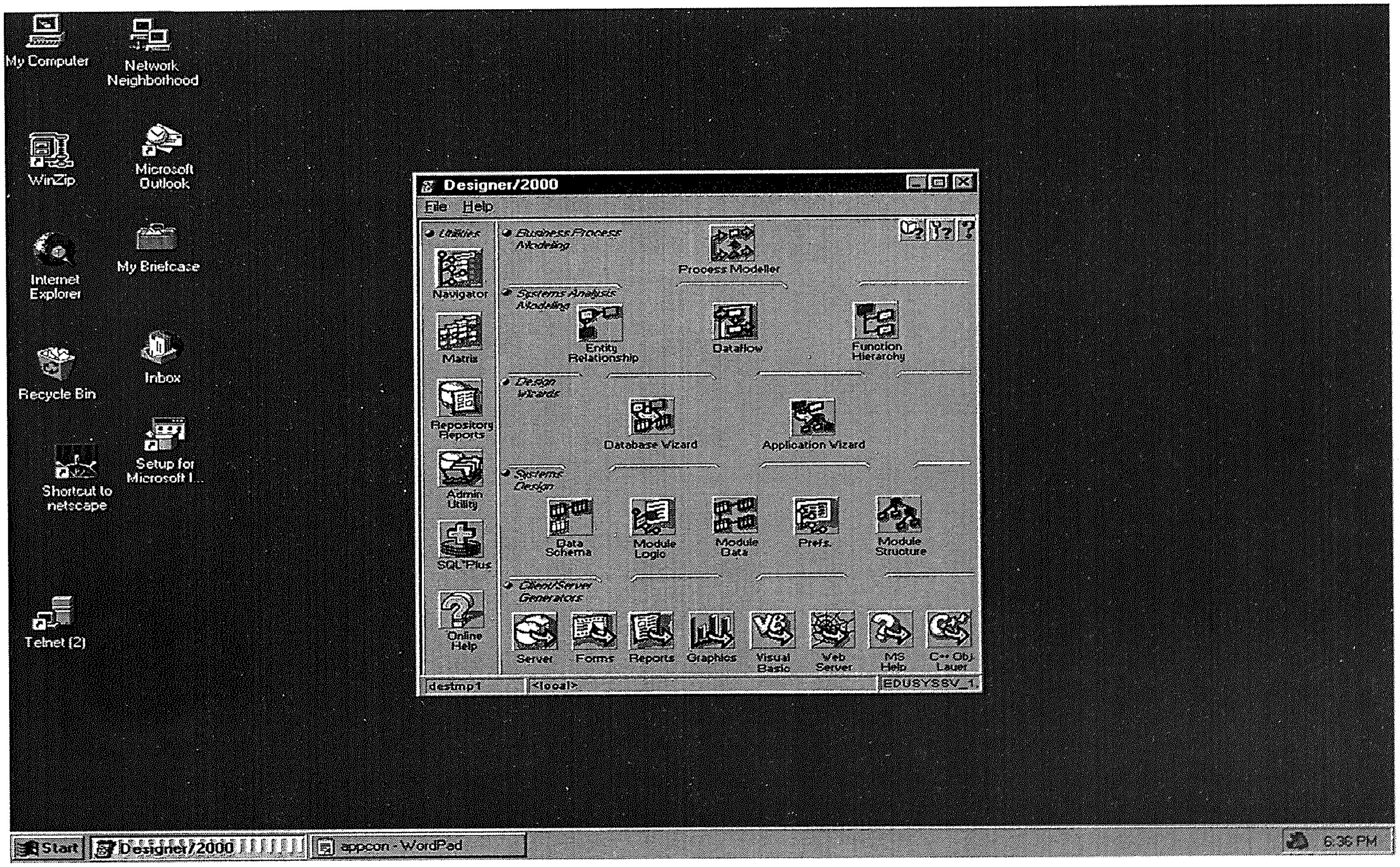


Figure APP2. Designer 2000 Screen after Login

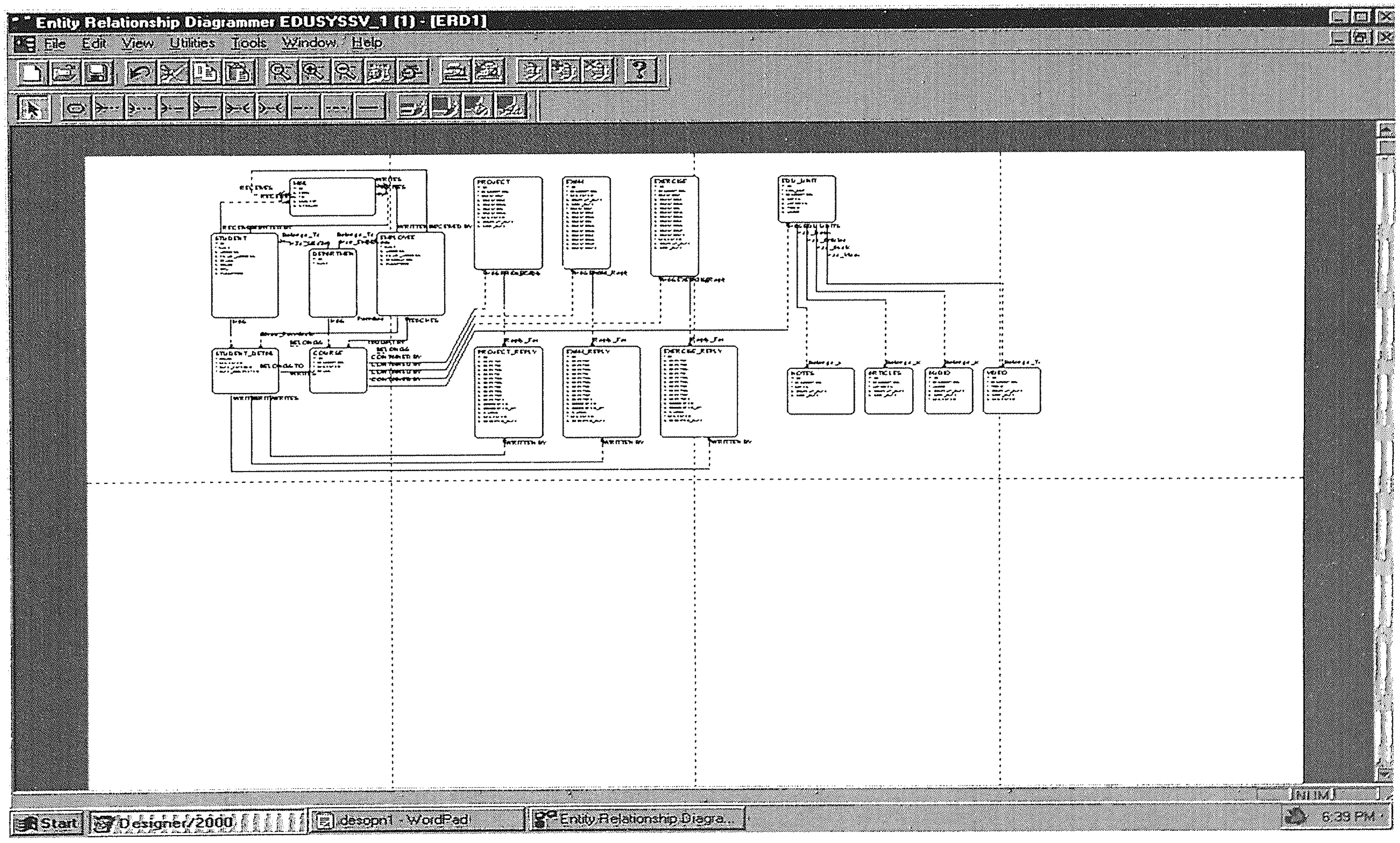


Figure APP3. Complete Entity Relationship Model in Designer/2000

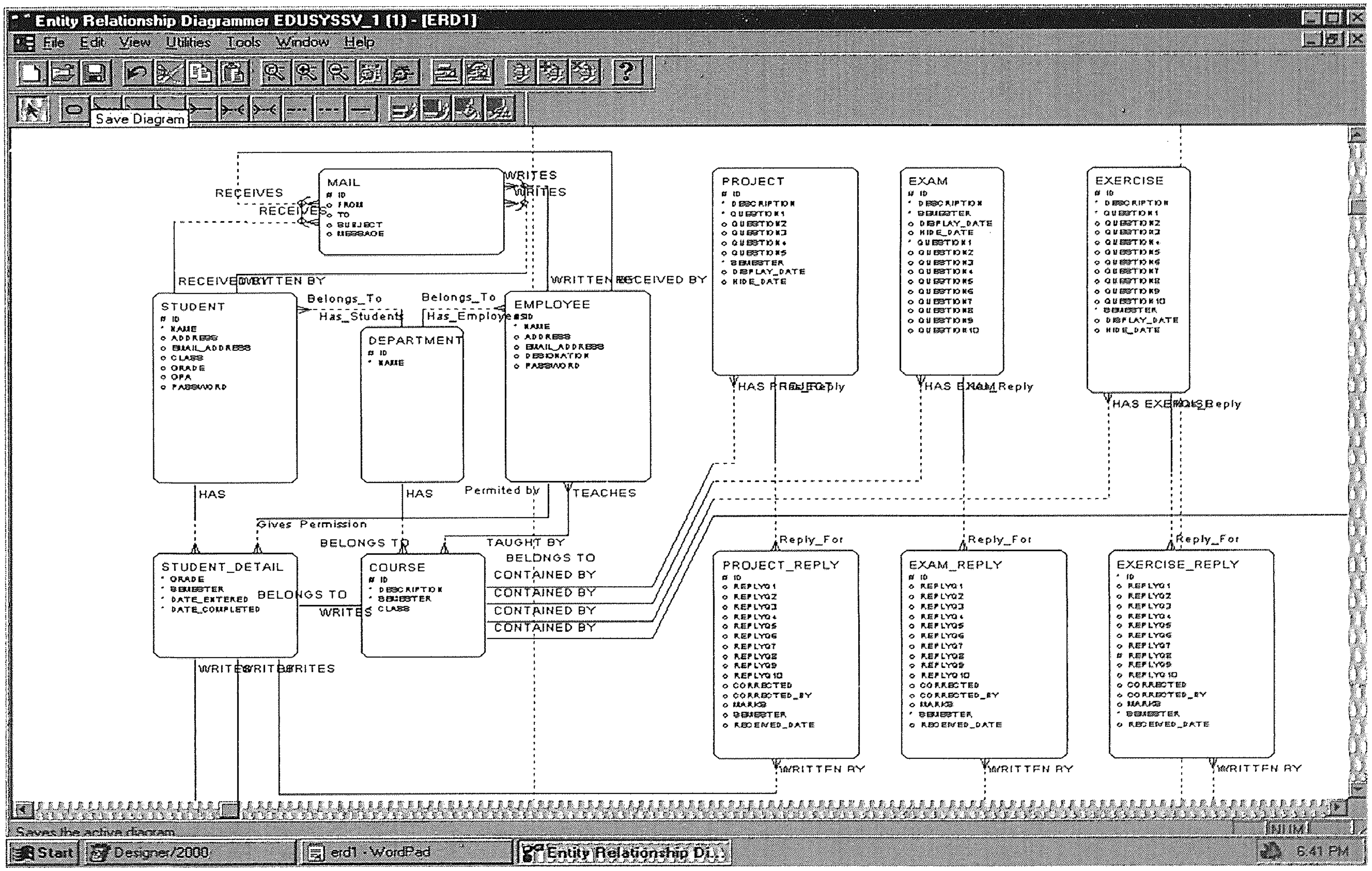


Figure APP4. Snapshot of Entity Relation Diagram - Part 1

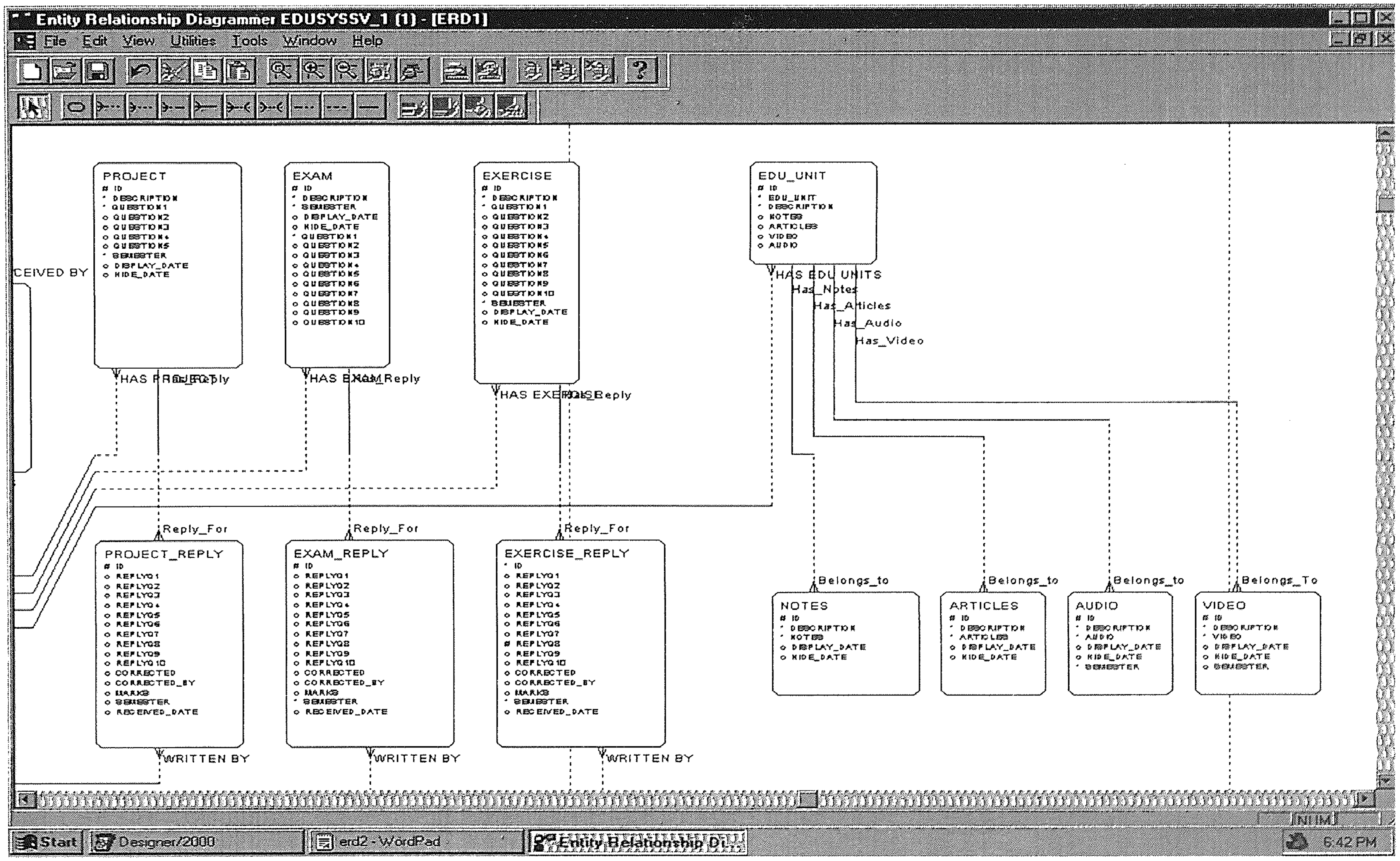


Figure APP5. Snapshot of Entity Relation Diagram - Part 2

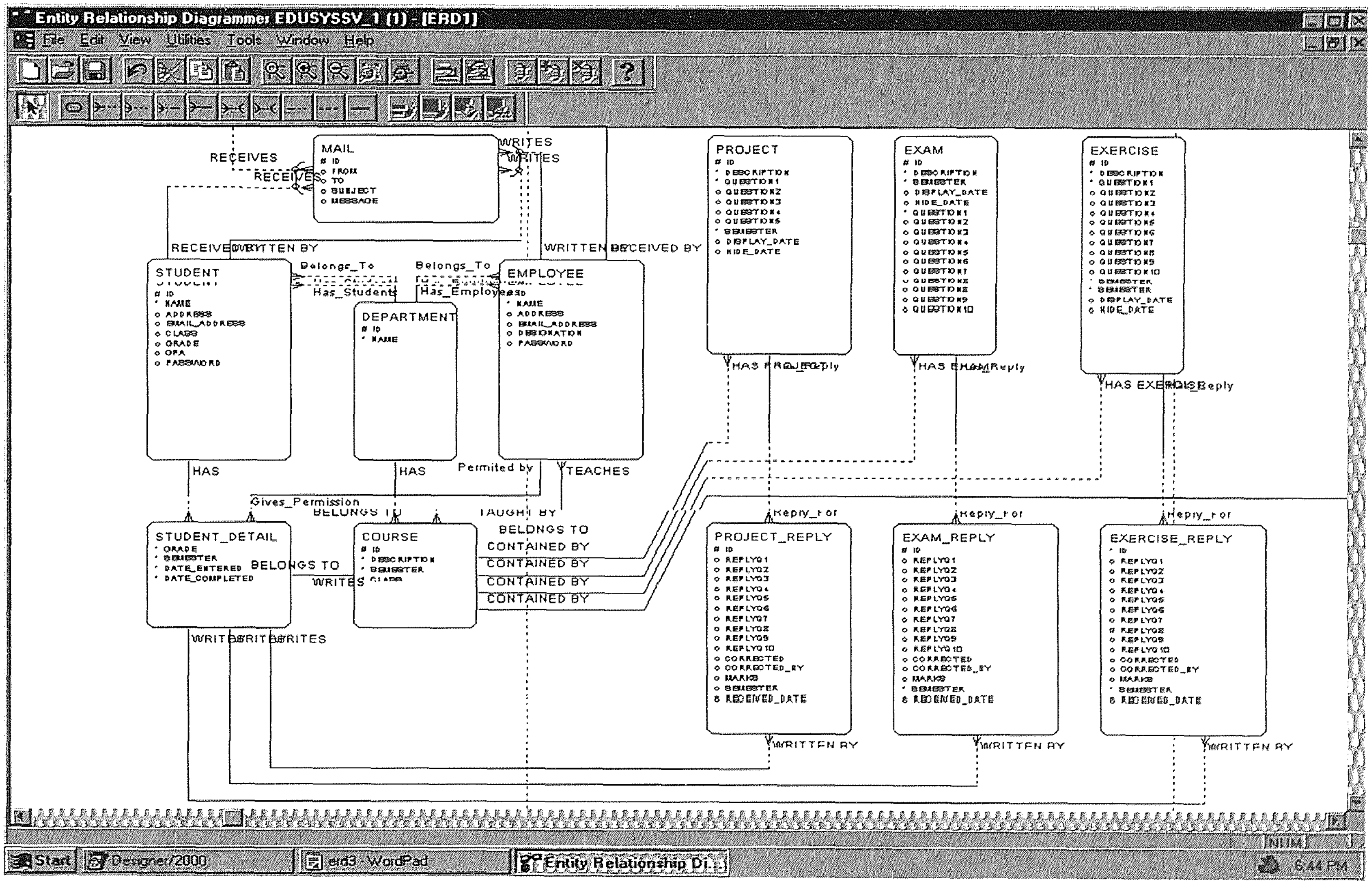
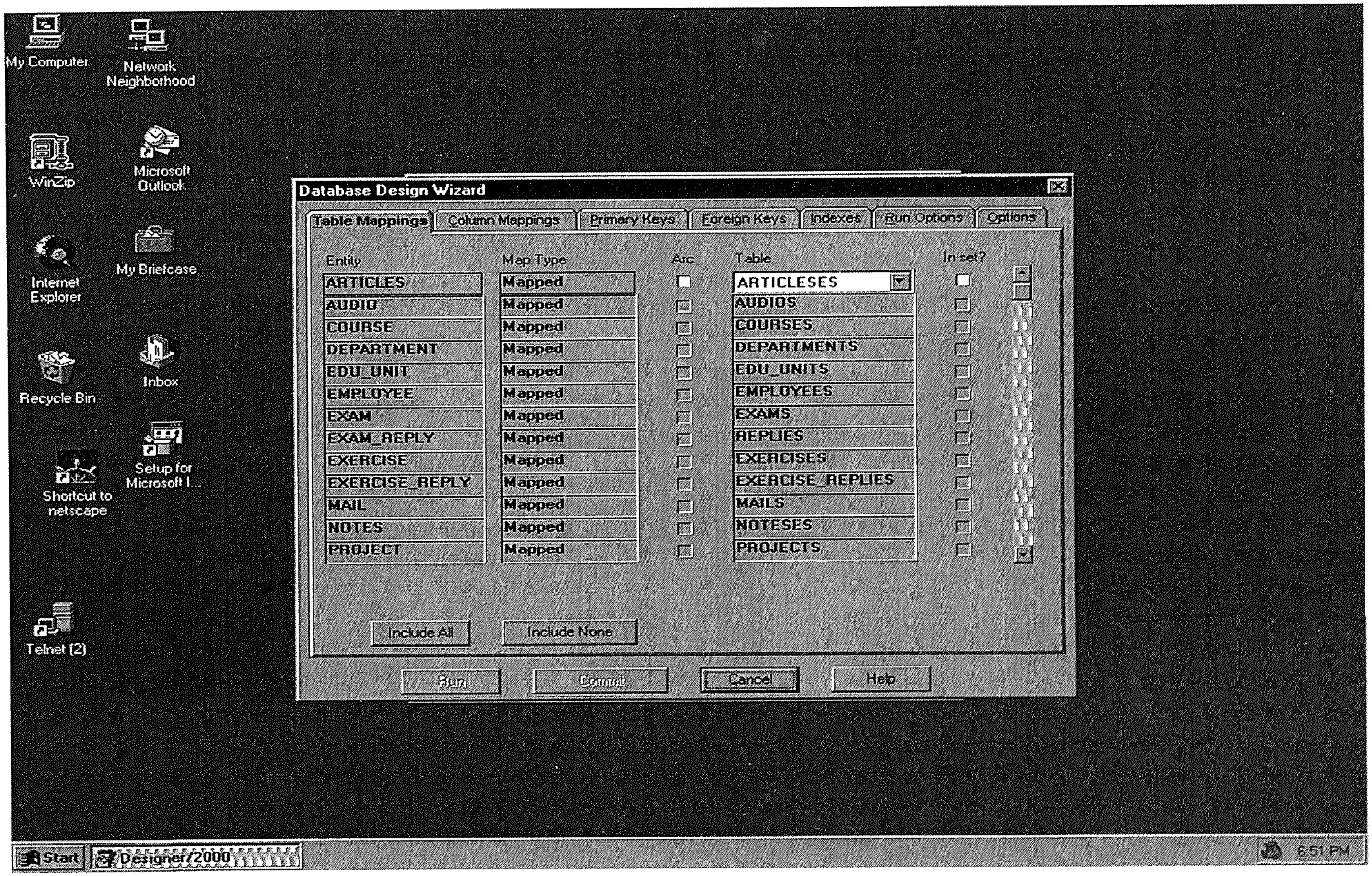


Figure APP6. Snapshot of Entity Relation Diagram - Part 3



Entity	Map Type	Arc	Table	In set?
ARTICLES	Mapped	☐	ARTICLESES	☐
AUDIO	Mapped	☐	AUDIOS	☐
COURSE	Mapped	☐	COURSES	☐
DEPARTMENT	Mapped	☐	DEPARTMENTS	☐
EDU_UNIT	Mapped	☐	EDU_UNITS	☐
EMPLOYEE	Mapped	☐	EMPLOYEES	☐
EXAM	Mapped	☐	EXAMS	☐
EXAM_REPLY	Mapped	☐	REPLIES	☐
EXERCISE	Mapped	☐	EXERCISES	☐
EXERCISE_REPLY	Mapped	☐	EXERCISE_REPLIES	☐
MAIL	Mapped	☐	MAILS	☐
NOTES	Mapped	☐	NOTESES	☐
PROJECT	Mapped	☐	PROJECTS	☐

Figure APP7. Database Mapping in Database Design Wizard

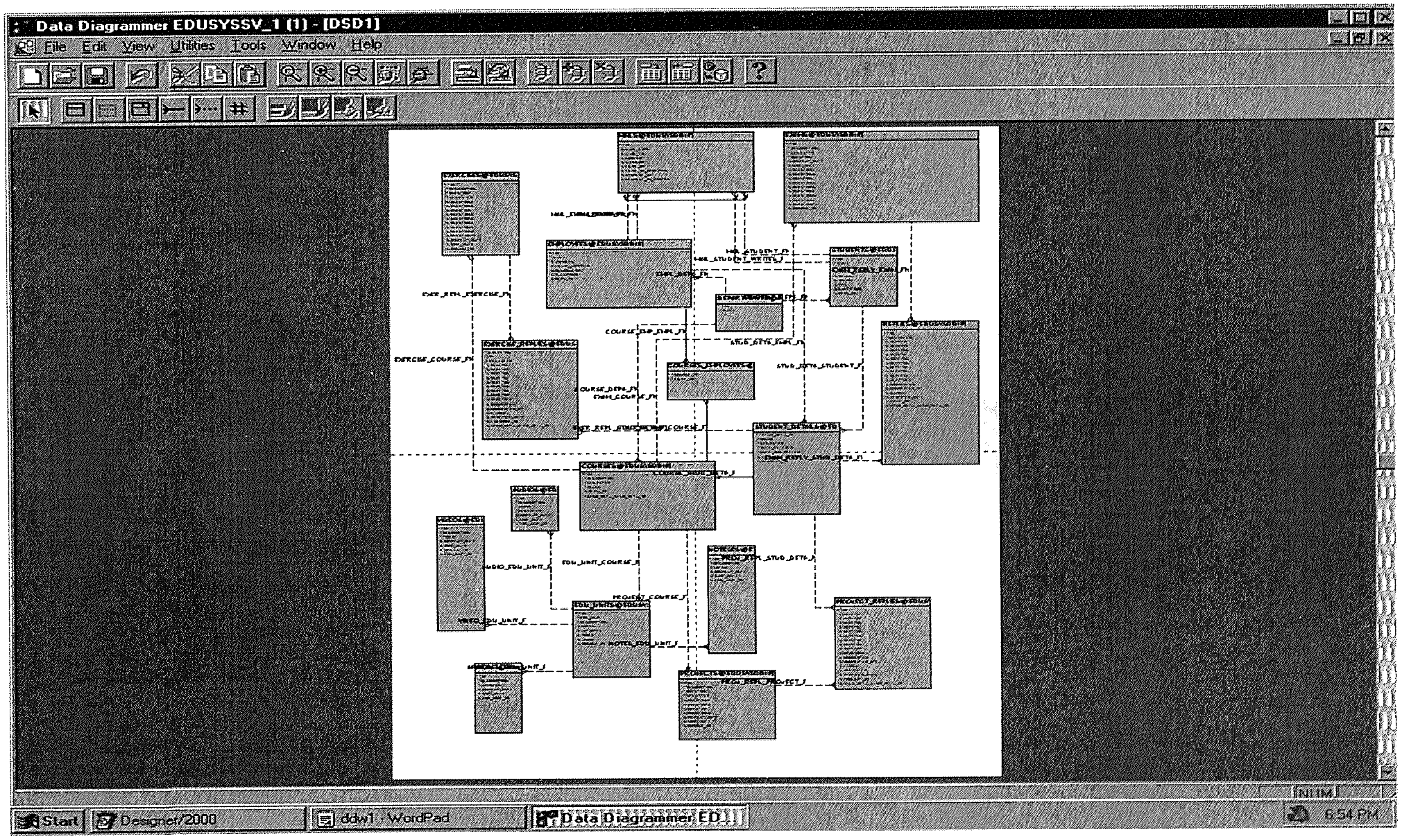


Figure APP8. Data Diagrammer Snapshot 1 - Overview

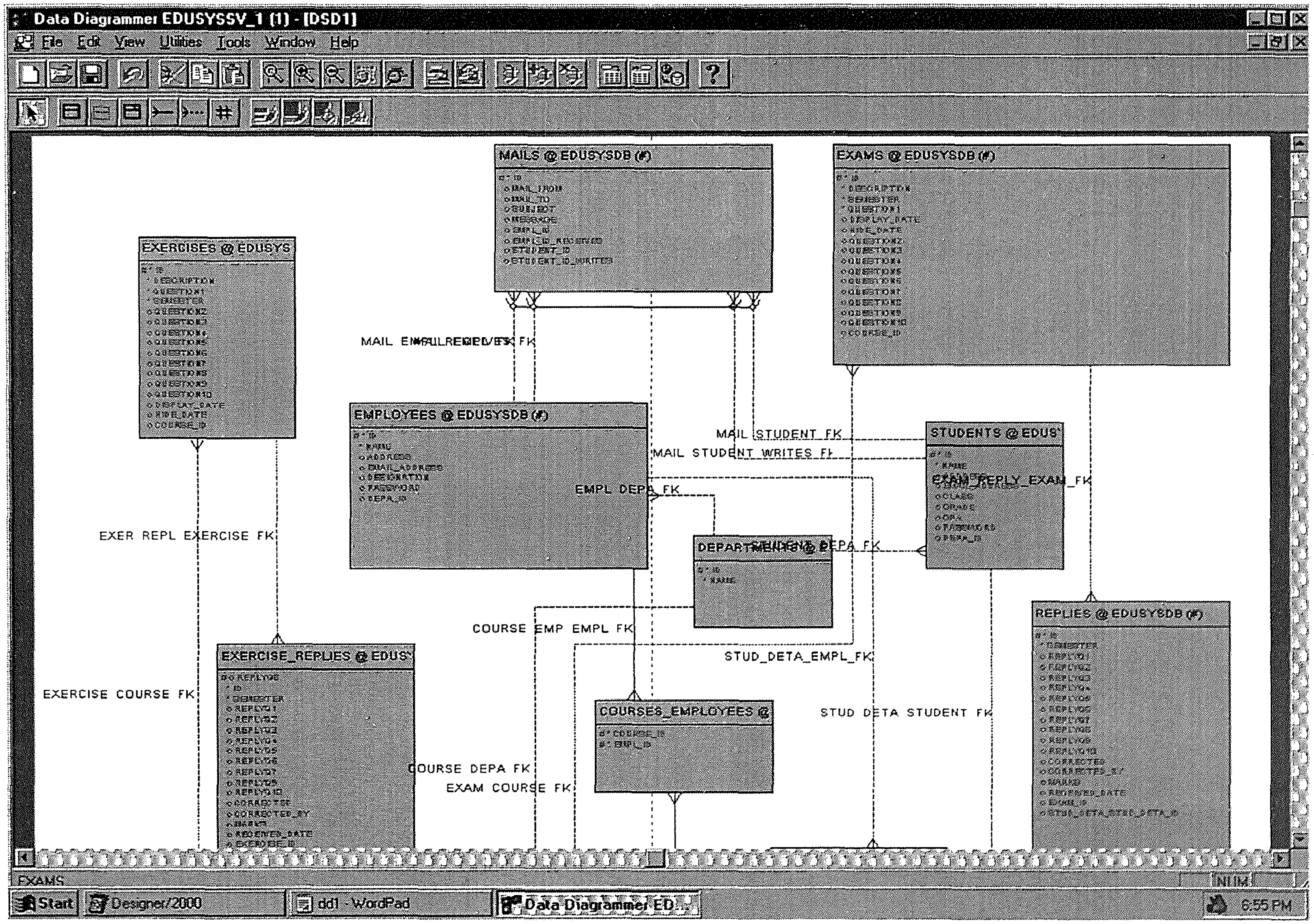


Figure APP9. Data Diagrammer Snapshot - Part 1

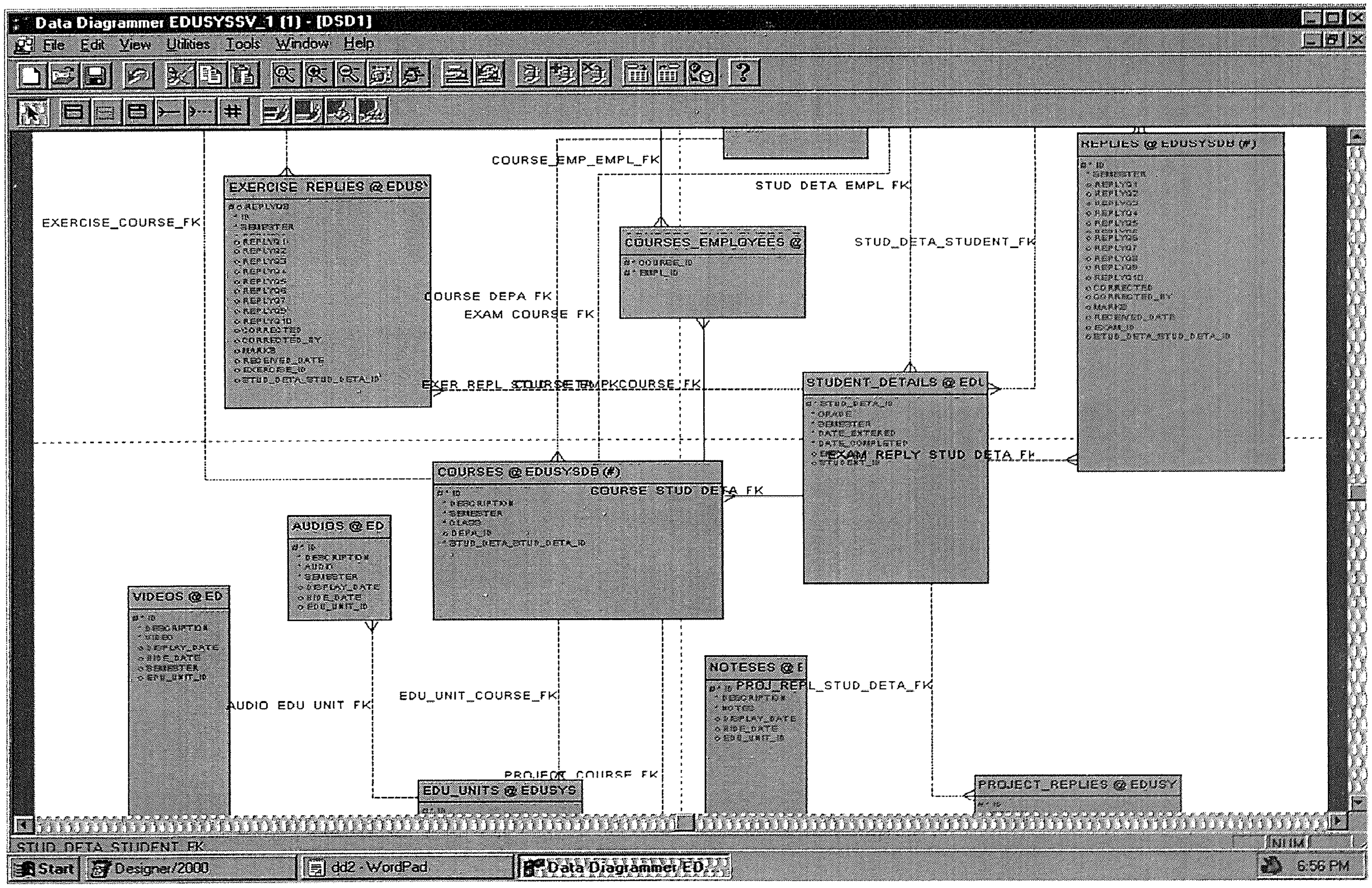


Figure APP10. Data Diagrammer Snapshot - Part 2

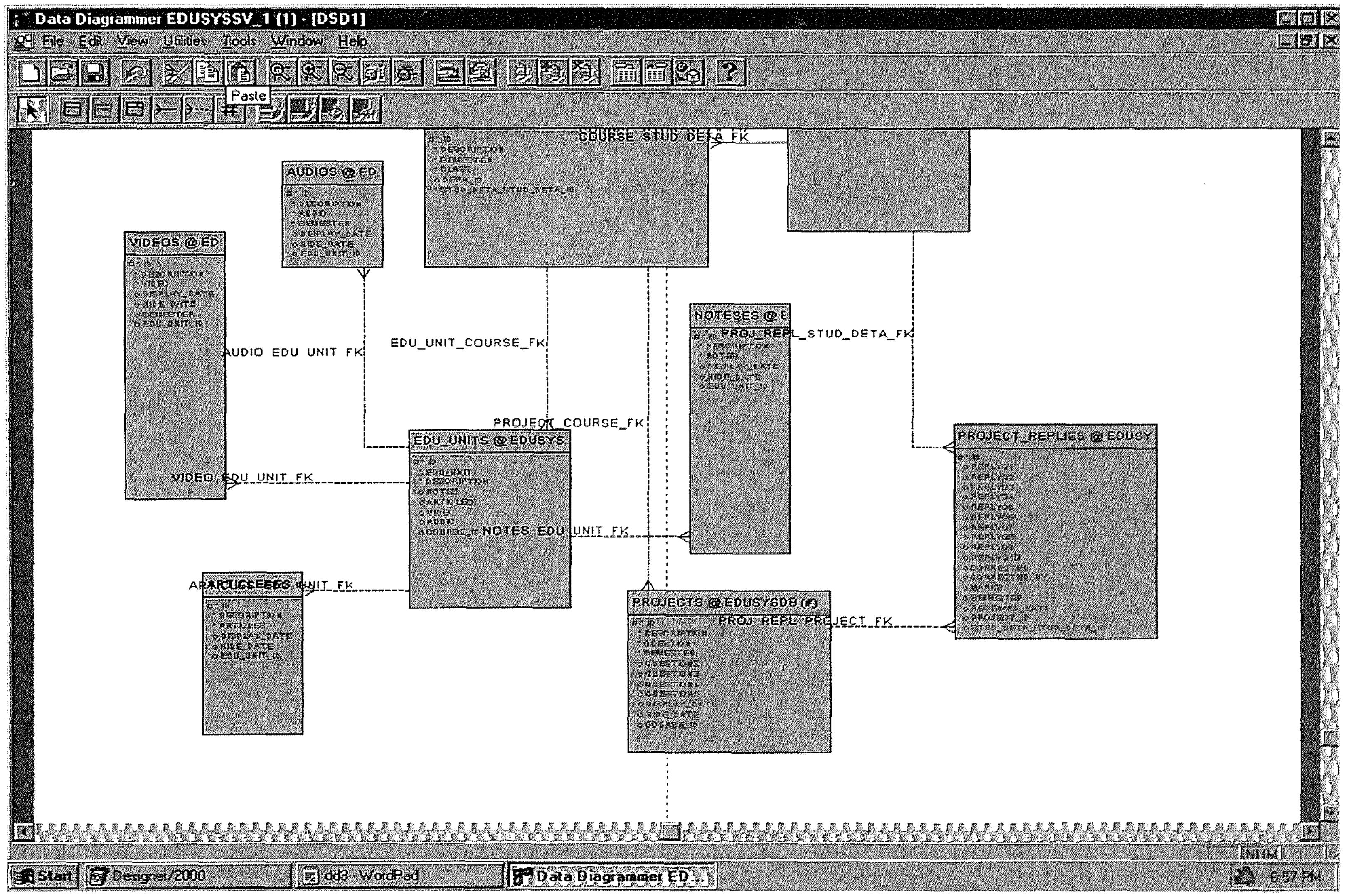


Figure APP11. Data Diagrammer Snapshot - Part 3

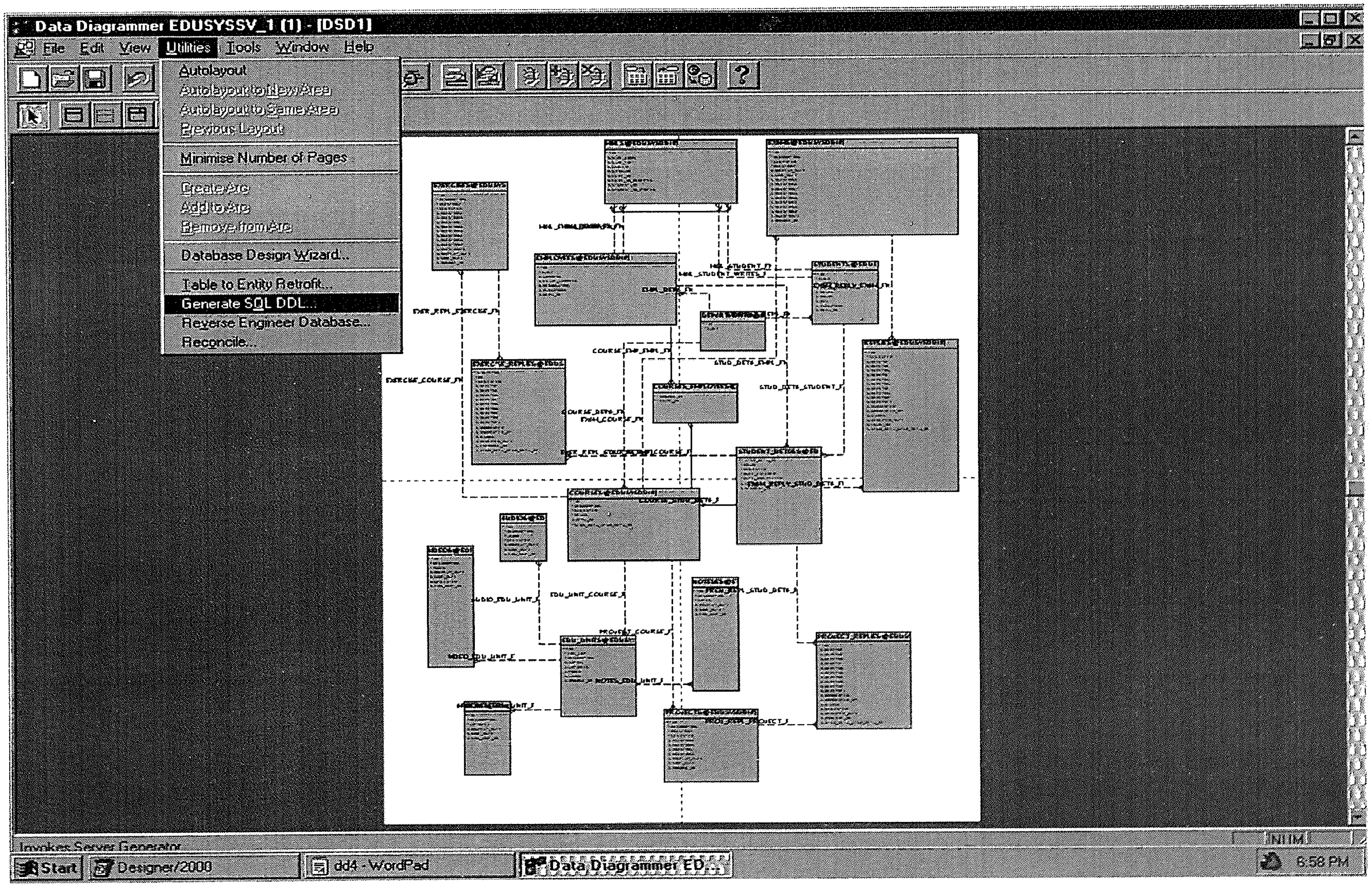
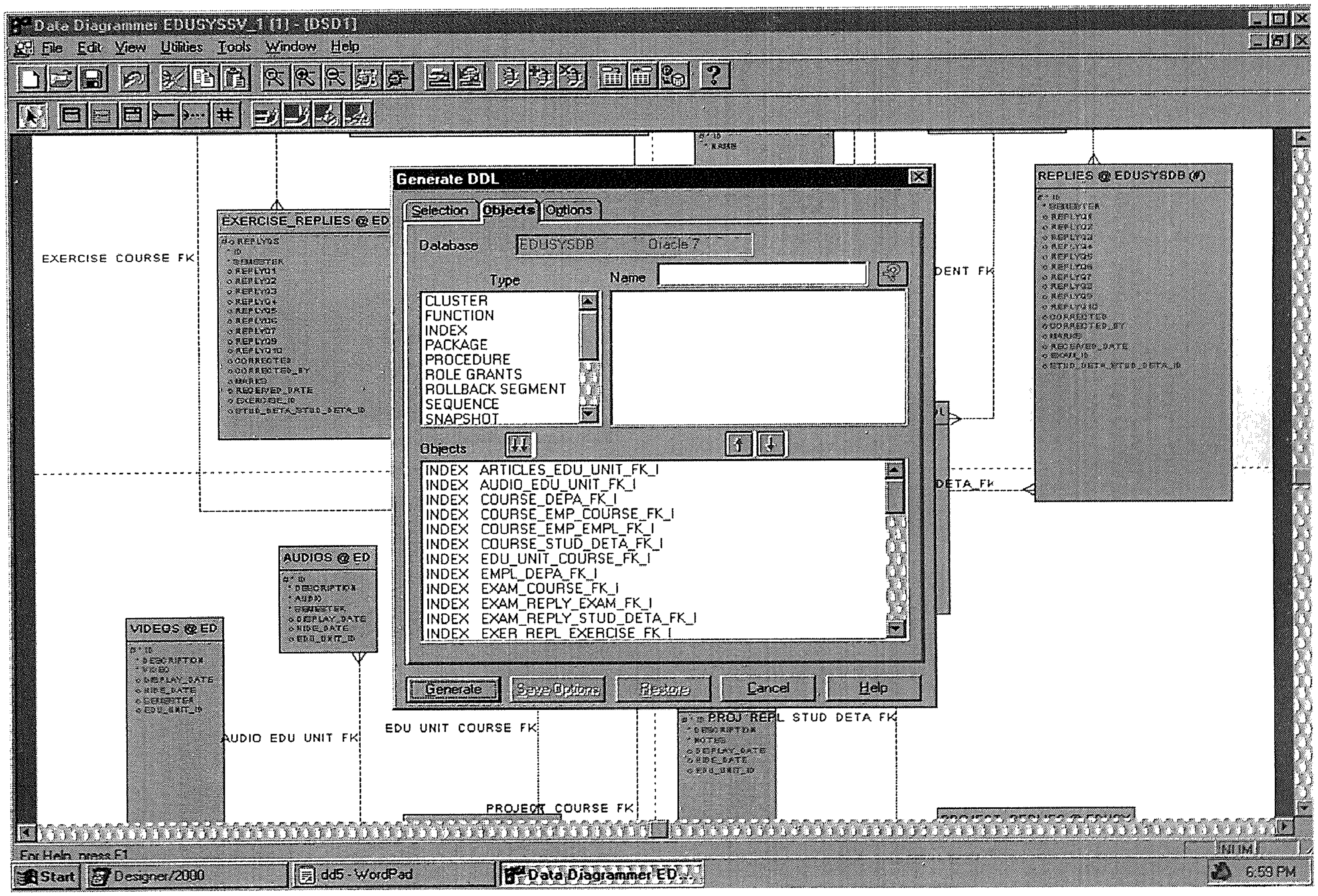


Figure APP12. Data Diagrammer Snapshot - Part 4 - Generate SQL DDL Selection



Generate DDL

Selection | Objects | Options

Database: EDUSYSDB Oracle 7

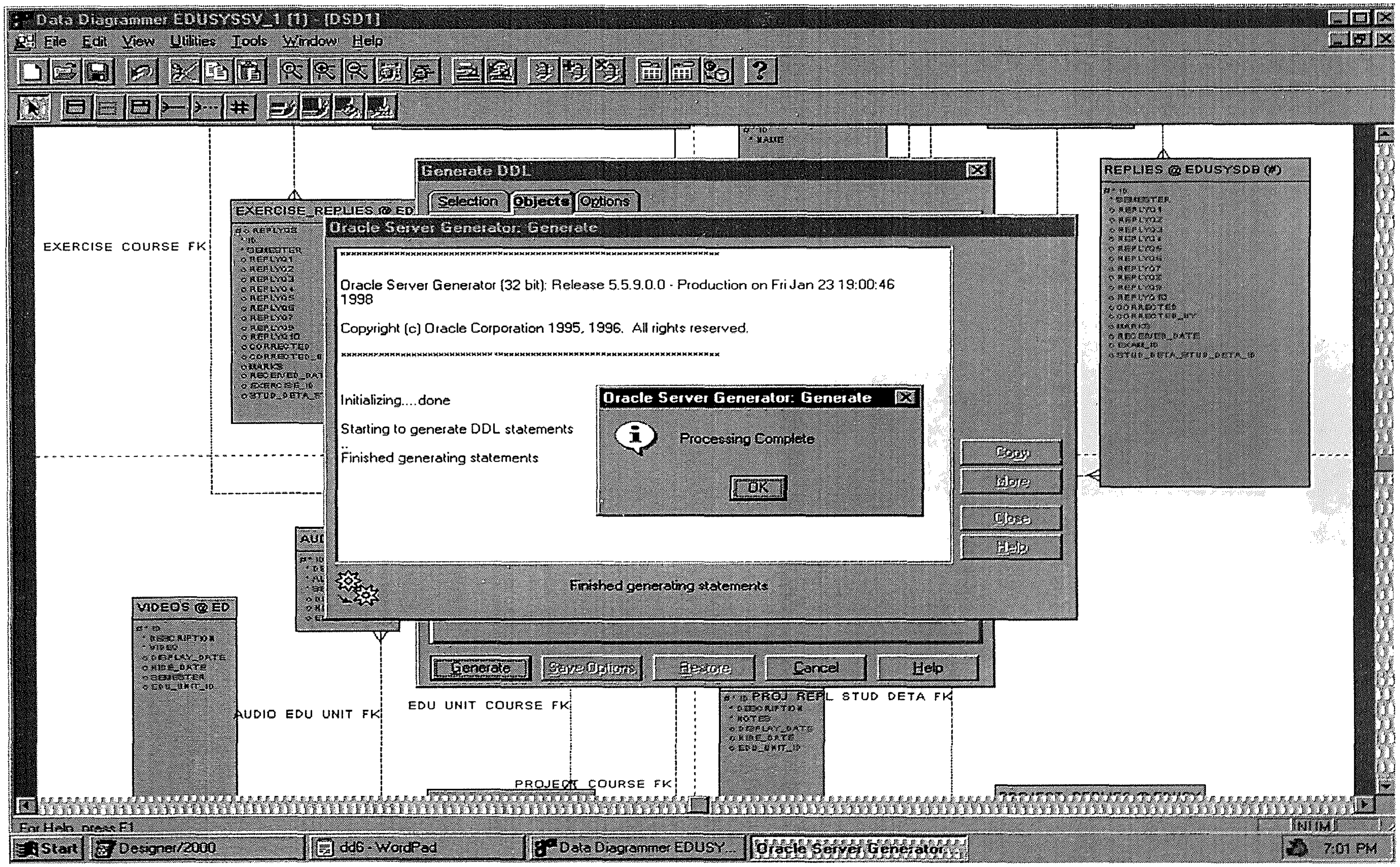
Type: Name:

```
CLUSTER
FUNCTION
INDEX
PACKAGE
PROCEDURE
ROLE GRANTS
ROLLBACK SEGMENT
SEQUENCE
SNAPSHOT
```

Objects

```
INDEX  ARTICLES_EDU_UNIT_FK_I
INDEX  AUDIO_EDU_UNIT_FK_I
INDEX  COURSE_DEPA_FK_I
INDEX  COURSE_EMP_COURSE_FK_I
INDEX  COURSE_EMP_EMPL_FK_I
INDEX  COURSE_STUD_DETA_FK_I
INDEX  EDU_UNIT_COURSE_FK_I
INDEX  EMPL_DEPA_FK_I
INDEX  EXAM_COURSE_FK_I
INDEX  EXAM_REPLY_EXAM_FK_I
INDEX  EXAM_REPLY_STUD_DETA_FK_I
INDEX  EXER_REPL_EXERCISE_FK_I
```

Figure APP13. Data Diagrammer Snapshot - Selection For Generate



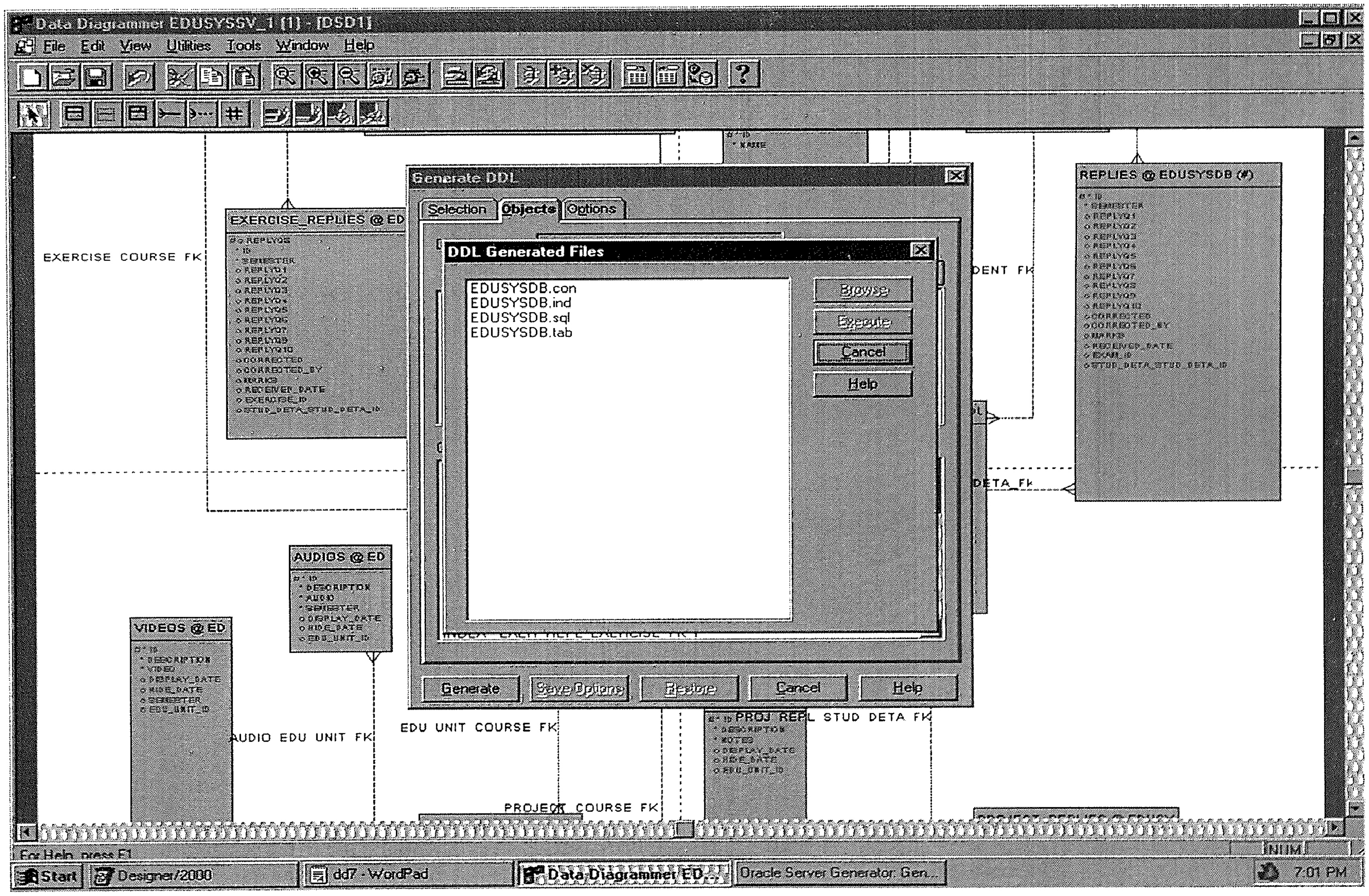
```
Oracle Server Generator (32 bit): Release 5.5.9.0.0 - Production on Fri Jan 23 19:00:46
1998

Copyright (c) Oracle Corporation 1995, 1996.  All rights reserved.

Initializing....done

Starting to generate DDL statements
..
Finished generating statements
```

Oracle Server Generator: Generate

Processing Complete

Finished generating statements

Figure APP14. Data Diagrammer Snapshot - Generation Completed



```
EDUSYSDB.con
EDUSYSDB.ind
EDUSYSDB.sql
EDUSYSDB.tab
```

Figure APP15. Data Diagrammer Snapshot - List of Generated Files

```
REM
REM  This ORACLE7 command file was generated by Oracle Server Generator
REM  Version 5.5.9.0.0 on 23-JAN-98
REM
REM For application EDUSYSSV_1 version 1 database EDUSYSDB
REM
SET SCAN OFF

SPOOL EDUSYSDB.lst

REM TABLE CREATION
start  EDUSYSDB.tab


REM CONSTRAINT CREATION
start  EDUSYSDB.con


REM INDEX CREATION
start  EDUSYSDB.ind


REM
REM  End of command file
REM
SPOOL OFF
```

```
REM
REM  This ORACLE7 command file was generated by Oracle Server Generator
REM  Version 5.5.9.0.0 on 23-JAN-98
REM
REM For application EDUSYSSV_1 version 1 database EDUSYSDB
REM
REM TABLE
REM     ARTICLESES
REM     AUDIOS
REM     COURSES
REM     COURSES_EMPLOYEES
REM     DEPARTMENTS
REM     EDU_UNITS
REM     EMPLOYEES
REM     EXAMS
REM     EXERCISES
REM     EXERCISE_REPLIES
REM     MAILS
REM     NOTESES
REM     PROJECTS
REM     PROJECT_REPLIES
REM     REPLIES
REM     STUDENTS
REM     STUDENT_DETAILS
REM     VIDEOS

REM
PROMPT
PROMPT Creating Table ARTICLESES
CREATE TABLE articleses(
 id                    NUMBER(10,0)          NOT NULL,
 description           VARCHAR2(200)         NOT NULL,
 articles              LONG                  NOT NULL,
 display_date          DATE                  NULL,
 hide_date             DATE                  NULL,
 edu_unit_id           NUMBER(10,0)          NULL
)
;

COMMENT ON COLUMN articleses.id
   IS 'ID';

COMMENT ON COLUMN articleses.description
   IS 'DESCRIPTION';
```

```
COMMENT ON COLUMN articleses.articles
   IS 'ACTUAL TEXT';

COMMENT ON COLUMN articleses.display_date
   IS 'DISPLAY DATE';

COMMENT ON COLUMN articleses.hide_date
   IS 'HIDE DATE';

COMMENT ON COLUMN articleses.edu_unit_id
   IS 'id';

REM
PROMPT
PROMPT Creating Table AUDIOS
CREATE TABLE audios(
 id                   NUMBER(10,0)          NOT NULL,
 description          VARCHAR2(200)         NOT NULL,
 audio                LONG RAW              NOT NULL,
 semester             VARCHAR2(15)          NOT NULL,
 display_date         DATE                  NULL,
 hide_date            DATE                  NULL,
 edu_unit_id          NUMBER(10,0)          NULL
)
;

COMMENT ON COLUMN audios.id
   IS 'ID';

COMMENT ON COLUMN audios.description
   IS 'DESCRIPTION';

COMMENT ON COLUMN audios.audio
   IS 'ACTUAL AUDIO';

COMMENT ON COLUMN audios.semester
   IS 'SEMESTER - SPRING 1997';

COMMENT ON COLUMN audios.display_date
   IS 'DISPLAY DATE';

COMMENT ON COLUMN audios.hide_date
   IS 'HIDE DATE';

COMMENT ON COLUMN audios.edu_unit_id
```

```
   IS 'id';

REM
PROMPT
PROMPT Creating Table COURSES
CREATE TABLE courses(
 id                     VARCHAR2(10)          NOT NULL,
 description               VARCHAR2(25)          NOT NULL,
 semester                  VARCHAR2(15)          NOT NULL,
 class                   VARCHAR2(3)           NOT NULL,
 depa_id                   VARCHAR2(10)          NULL,
 stud_deta_stud_deta_id        NUMBER(10,0)          NOT NULL
)
;

COMMENT ON COLUMN courses.id
   IS 'Course id';

COMMENT ON COLUMN courses.description
   IS 'Course Name';

COMMENT ON COLUMN courses.semester
   IS 'This semester date';

COMMENT ON COLUMN courses.depa_id
   IS 'Department id';

REM
PROMPT
PROMPT Creating Table COURSES_EMPLOYEES
CREATE TABLE courses_employees(
 course_id                 VARCHAR2(10)          NOT NULL,
 empl_id                   VARCHAR2(9)           NOT NULL
)
;

COMMENT ON COLUMN courses_employees.course_id
   IS 'Course id';

COMMENT ON COLUMN courses_employees.empl_id
   IS 'Employee Id';

REM
PROMPT
PROMPT Creating Table DEPARTMENTS
```

```
CREATE TABLE departments(
 id                    VARCHAR2(10)          NOT NULL,
 name                  VARCHAR2(25)          NOT NULL
)
;

COMMENT ON COLUMN departments.id
   IS 'Department id';

COMMENT ON COLUMN departments.name
   IS 'Department Name';

REM
PROMPT
PROMPT Creating Table EDU_UNITS
CREATE TABLE edu_units(
 id                    NUMBER(10,0)          NOT NULL,
 edu_unit              NUMBER(10,0)          NOT NULL,
 description           VARCHAR2(200)         NOT NULL,
 notes                 VARCHAR2(1)           NULL,
 articles              VARCHAR2(1)           NULL,
 video                 VARCHAR2(1)           NULL,
 audio                 VARCHAR2(1)           NULL,
 course_id             VARCHAR2(10)          NULL
)
;

COMMENT ON COLUMN edu_units.id
   IS 'id';

COMMENT ON COLUMN edu_units.edu_unit
   IS 'Edu unit';

COMMENT ON COLUMN edu_units.description
   IS 'Description';

COMMENT ON COLUMN edu_units.notes
   IS 'Y/N';

COMMENT ON COLUMN edu_units.articles
   IS 'Y/N';

COMMENT ON COLUMN edu_units.video
   IS 'Y/N';
```

```
COMMENT ON COLUMN edu_units.audio
   IS 'Y/N';

COMMENT ON COLUMN edu_units.course_id
   IS 'Course id';

REM
PROMPT
PROMPT Creating Table EMPLOYEES
CREATE TABLE employees(
 id                    VARCHAR2(9)        NOT NULL,
 name                  VARCHAR2(25)       NOT NULL,
 address               VARCHAR2(40)       NULL,
 email_address         VARCHAR2(20)       NULL,
 designation           VARCHAR2(25)       NULL,
 password              VARCHAR2(10)       NULL,
 depa_id               VARCHAR2(10)       NULL
)
;

COMMENT ON COLUMN employees.id
   IS 'Employee Id';

COMMENT ON COLUMN employees.name
   IS 'Name';

COMMENT ON COLUMN employees.address
   IS 'Address';

COMMENT ON COLUMN employees.email_address
   IS 'Email Address';

COMMENT ON COLUMN employees.designation
   IS 'Designation';

COMMENT ON COLUMN employees.password
   IS 'Password';

COMMENT ON COLUMN employees.depa_id
   IS 'Department id';

REM
PROMPT
PROMPT Creating Table EXAMS
CREATE TABLE exams(
id                  NUMBER(10,0)          NOT NULL,
description              LONG             NOT NULL,
semester                VARCHAR2(15)           NOT NULL,
question1                LONG             NOT NULL,
display_date              DATE             NULL,
hide_date                DATE             NULL,
question2                LONG             NULL,
question3                LONG             NULL,
question4                LONG             NULL,
question5                LONG             NULL,
question6                LONG             NULL,
question7                LONG             NULL,
question8                LONG             NULL,
question9                LONG             NULL,
question10                LONG             NULL,
course_id               VARCHAR2(10)          NULL
)
;

COMMENT ON COLUMN exams.id
   IS 'id';

COMMENT ON COLUMN exams.description
   IS 'Short description';

COMMENT ON COLUMN exams.semester
   IS 'Semester - SPRING 1997';

COMMENT ON COLUMN exams.display_date
   IS 'Display Date';

COMMENT ON COLUMN exams.hide_date
   IS 'Hide Date';

COMMENT ON COLUMN exams.course_id
   IS 'Course id';

REM
PROMPT
PROMPT Creating Table EXERCISES
CREATE TABLE exercises(
 id                  NUMBER(10,0)          NOT NULL,
 description              VARCHAR2(200)          NOT NULL,
 question1                LONG             NOT NULL,
 semester                VARCHAR2(15)          NOT NULL,
question2           LONG            NULL,
question3           LONG            NULL,
question4           LONG            NULL,
question5           LONG            NULL,
question6           LONG            NULL,
question7           LONG            NULL,
question8           LONG            NULL,
question9           LONG            NULL,
question10          LONG            NULL,
display_date        DATE            NULL,
hide_date           DATE            NULL,
course_id           VARCHAR2(10)        NULL
)
;

COMMENT ON COLUMN exercises.id
  IS 'id';

COMMENT ON COLUMN exercises.description
  IS 'Exercise description';

COMMENT ON COLUMN exercises.semester
  IS 'Semseter';

COMMENT ON COLUMN exercises.display_date
  IS 'Display Date';

COMMENT ON COLUMN exercises.hide_date
  IS 'Hide Date';

COMMENT ON COLUMN exercises.course_id
  IS 'Course id';

REM
PROMPT
PROMPT Creating Table EXERCISE_REPLIES
CREATE TABLE exercise_replies(
replyq8             LONG            NULL,
id              NUMBER(10,0)          NOT NULL,
semester            VARCHAR2(15)          NOT NULL,
replyq1             LONG            NULL,
replyq2             LONG            NULL,
replyq3             LONG            NULL,
replyq4             LONG            NULL,
replyq5             LONG            NULL,
replyq6              LONG           NULL,
replyq7              LONG           NULL,
replyq9              LONG           NULL,
replyq10             LONG           NULL,
corrected            VARCHAR2(1)       NULL,
corrected_by           VARCHAR2(9)       NULL,
marks               NUMBER(5,2)       NULL,
received_date          DATE            NULL,
exercise_id           NUMBER(10,0)       NULL,
stud_deta_stud_deta_id      NUMBER(10,0)        NULL
)
;

COMMENT ON COLUMN exercise_replies.id
  IS 'ID';

COMMENT ON COLUMN exercise_replies.corrected
  IS 'Y/N';

COMMENT ON COLUMN exercise_replies.exercise_id
  IS 'id';

REM
PROMPT
PROMPT Creating Table MAILS
CREATE TABLE mails(
 id                 NUMBER(10,0)       NOT NULL,
 mail_from              VARCHAR2(9)       NULL,
 mail_to              VARCHAR2(9)       NULL,
 subject              VARCHAR2(200)      NULL,
 message               LONG           NULL,
 empl_id               VARCHAR2(9)       NULL,
 empl_id_receives          VARCHAR2(9)       NULL,
 student_id             VARCHAR2(9)       NULL,
 student_id_writes          VARCHAR2(9)       NULL
)
;

COMMENT ON COLUMN mails.id
  IS 'Mail ID';

COMMENT ON COLUMN mails.mail_from
  IS 'From ID';

COMMENT ON COLUMN mails.mail_to
```

```
  IS 'To';

COMMENT ON COLUMN mails.subject
  IS 'Subject';

COMMENT ON COLUMN mails.empl_id
  IS 'Employee Id';

COMMENT ON COLUMN mails.empl_id_receives
  IS 'Employee Id';

COMMENT ON COLUMN mails.student_id
  IS 'Student id';

COMMENT ON COLUMN mails.student_id_writes
  IS 'Student id';

REM
PROMPT
PROMPT Creating Table NOTESES
CREATE TABLE noteses(
 id                    NUMBER(10,0)          NOT NULL,
 description               VARCHAR2(200)          NOT NULL,
 notes                   LONG                 NOT NULL,
 display_date              DATE                 NULL,
 hide_date                DATE                 NULL,
 edu_unit_id               NUMBER(10,0)           NULL
)
;

COMMENT ON COLUMN noteses.id
  IS 'ID';

COMMENT ON COLUMN noteses.description
  IS 'DESCRIPTION';

COMMENT ON COLUMN noteses.notes
  IS 'ACTUAL NOTES';

COMMENT ON COLUMN noteses.display_date
  IS 'DISPLAY DATE';

COMMENT ON COLUMN noteses.hide_date
  IS 'HIDE DATE';
```

```
COMMENT ON COLUMN noteses.edu_unit_id
   IS 'id';

REM
PROMPT
PROMPT Creating Table PROJECTS
CREATE TABLE projects(
 id                    NUMBER(10,0)          NOT NULL,
 description           VARCHAR2(200)         NOT NULL,
 question1             LONG                  NOT NULL,
 semester              VARCHAR2(15)          NOT NULL,
 question2             LONG                  NULL,
 question3             LONG                  NULL,
 question4             LONG                  NULL,
 question5             LONG                  NULL,
 display_date          DATE                  NULL,
 hide_date             DATE                  NULL,
 course_id             VARCHAR2(10)          NULL
)
;

COMMENT ON COLUMN projects.description
   IS 'Short Project Description';

COMMENT ON COLUMN projects.question1
   IS 'Actual project';

COMMENT ON COLUMN projects.semester
   IS 'This semester';

COMMENT ON COLUMN projects.display_date
   IS 'Display date';

COMMENT ON COLUMN projects.hide_date
   IS 'Hide date';

COMMENT ON COLUMN projects.course_id
   IS 'Course id';

REM
PROMPT
PROMPT Creating Table PROJECT_REPLIES
CREATE TABLE project_replies(
 id                    NUMBER(10,0)          NOT NULL,
 replyq1               LONG                  NULL,
 replyq2             LONG                NULL,
 replyq3             LONG                NULL,
 replyq4             LONG                NULL,
 replyq5             LONG                NULL,
 replyq6             LONG                NULL,
 replyq7             LONG                NULL,
 replyq8             LONG                NULL,
 replyq9             LONG                NULL,
 replyq10             LONG                NULL,
 corrected           VARCHAR2(1)            NULL,
 corrected_by          VARCHAR2(9)            NULL,
 marks              NUMBER(5,2)           NULL,
 semester            VARCHAR2(15)          NULL,
 received_date          DATE                NULL,
 project_id           NUMBER(10,0)          NULL,
 stud_deta_stud_deta_id      NUMBER(10,0)          NULL
)
;

COMMENT ON COLUMN project_replies.id
  IS 'ID';

COMMENT ON COLUMN project_replies.corrected
  IS 'Y/N';

REM
PROMPT
PROMPT Creating Table REPLIES
CREATE TABLE replies(
 id                NUMBER(10,0)          NOT NULL,
 semester            VARCHAR2(15)          NOT NULL,
 replyq1             LONG                NULL,
 replyq2             LONG                NULL,
 replyq3             LONG                NULL,
 replyq4             LONG                NULL,
 replyq5             LONG                NULL,
 replyq6             LONG                NULL,
 replyq7             LONG                NULL,
 replyq8             LONG                NULL,
 replyq9             LONG                NULL,
 replyq10             LONG                NULL,
 corrected           VARCHAR2(1)            NULL,
 corrected_by          VARCHAR2(9)            NULL,
 marks              NUMBER(5,2)           NULL,
 received_date          DATE                NULL,
 exam_id                NUMBER(10,0)          NULL,
 stud_deta_stud_deta_id        NUMBER(10,0)          NULL
)
;

COMMENT ON COLUMN replies.id
   IS 'ID';

COMMENT ON COLUMN replies.corrected
   IS 'Y/N';

COMMENT ON COLUMN replies.marks
   IS '100.00';

COMMENT ON COLUMN replies.exam_id
   IS 'id';

REM
PROMPT
PROMPT Creating Table STUDENTS
CREATE TABLE students(
 id                    VARCHAR2(9)          NOT NULL,
 name                    VARCHAR2(25)          NOT NULL,
 address                 VARCHAR2(40)          NULL,
 email_address              VARCHAR2(20)          NULL,
 class                  VARCHAR2(3)          NULL,
 grade                  VARCHAR2(2)          NULL,
 gpa                   NUMBER(4,2)          NULL,
 password                 VARCHAR2(10)          NULL,
 depa_id                 VARCHAR2(10)          NULL
)
;

COMMENT ON COLUMN students.id
   IS 'Student id';

COMMENT ON COLUMN students.name
   IS 'Student Name';

COMMENT ON COLUMN students.address
   IS 'Student Address';

COMMENT ON COLUMN students.email_address
   IS 'Email Addess';
```

```
COMMENT ON COLUMN students.class
   IS 'Class to which student belongs';

COMMENT ON COLUMN students.grade
   IS 'Student Grade';

COMMENT ON COLUMN students.gpa
   IS 'Student Total GPA';

COMMENT ON COLUMN students.password
   IS 'Password';

COMMENT ON COLUMN students.depa_id
   IS 'Department id';

REM
PROMPT
PROMPT Creating Table STUDENT_DETAILS
CREATE TABLE student_details(
 stud_deta_id          NUMBER(10,0)        NOT NULL,
 grade                 VARCHAR2(2)         NOT NULL,
 semester              VARCHAR2(15)        NOT NULL,
 date_entered          DATE                NOT NULL,
 date_completed        DATE                NOT NULL,
 empl_id               VARCHAR2(9)         NULL,
 student_id            VARCHAR2(9)         NULL
)
;

COMMENT ON COLUMN student_details.grade
   IS 'Grade';

COMMENT ON COLUMN student_details.semester
   IS 'This semester date';

COMMENT ON COLUMN student_details.date_completed
   IS 'DATE OF COMPLETION';

COMMENT ON COLUMN student_details.empl_id
   IS 'Employee Id';

COMMENT ON COLUMN student_details.student_id
   IS 'Student id';

REM
```

```
PROMPT
PROMPT Creating Table VIDEOS
CREATE TABLE videos(
 id                NUMBER(10,0)        NOT NULL,
 description           VARCHAR2(200)        NOT NULL,
 video              LONG RAW          NOT NULL,
 display_date           DATE             NULL,
 hide_date             DATE            NULL,
 semester             VARCHAR2(15)         NULL,
 edu_unit_id            NUMBER(10,0)         NULL
)
;

COMMENT ON COLUMN videos.id
   IS 'ID';

COMMENT ON COLUMN videos.description
   IS 'DESCRIPTION';

COMMENT ON COLUMN videos.video
   IS 'ACTUAL VIDEO';

COMMENT ON COLUMN videos.display_date
   IS 'DISPLAY DATE';

COMMENT ON COLUMN videos.hide_date
   IS 'HIDE DATE';

COMMENT ON COLUMN videos.semester
   IS 'SEMESTER - SPRING 1997';

COMMENT ON COLUMN videos.edu_unit_id
   IS 'id';
```

```
REM
REM  This ORACLE7 command file was generated by Oracle Server Generator
REM  Version 5.5.9.0.0 on 23-JAN-98
REM
REM For application EDUSYSSV_1 version 1 database EDUSYSDB
REM
REM CONSTRAINT

PROMPT Adding PRIMARY Constraint To ARTICLESES Table

ALTER TABLE ARTICLESES ADD (
    CONSTRAINT ARTICLES_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To AUDIOS Table

ALTER TABLE AUDIOS ADD (
    CONSTRAINT AUDIO_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To COURSES Table

ALTER TABLE COURSES ADD (
    CONSTRAINT COURSE_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To COURSES_EMPLOYEES Table

ALTER TABLE COURSES_EMPLOYEES ADD (
    CONSTRAINT COURSE_EMP_PK
    PRIMARY KEY (COURSE_ID,
            EMPL_ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To DEPARTMENTS Table

ALTER TABLE DEPARTMENTS ADD (
    CONSTRAINT DEPA_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To EDU_UNITS Table

ALTER TABLE EDU_UNITS ADD (
    CONSTRAINT EDU_UNIT_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To EMPLOYEES Table

ALTER TABLE EMPLOYEES ADD (
    CONSTRAINT EMPL_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To EXAMS Table

ALTER TABLE EXAMS ADD (
    CONSTRAINT EXAM_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To EXERCISES Table
```

```
ALTER TABLE EXERCISES ADD (
    CONSTRAINT EXERCISE_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To EXERCISE_REPLIES Table

ALTER TABLE EXERCISE_REPLIES ADD (
    CONSTRAINT EXER_REPL_PK
    PRIMARY KEY (REPLYQ8)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To MAILS Table

ALTER TABLE MAILS ADD (
    CONSTRAINT MAIL_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To NOTESES Table

ALTER TABLE NOTESES ADD (
    CONSTRAINT NOTES_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To PROJECTS Table

ALTER TABLE PROJECTS ADD (
    CONSTRAINT PROJECT_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To PROJECT_REPLIES Table

ALTER TABLE PROJECT_REPLIES ADD (
    CONSTRAINT PROJ_REPL_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To REPLIES Table

ALTER TABLE REPLIES ADD (
    CONSTRAINT EXAM_REPLY_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To STUDENTS Table

ALTER TABLE STUDENTS ADD (
    CONSTRAINT STUDENT_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To STUDENT_DETAILS Table

ALTER TABLE STUDENT_DETAILS ADD (
    CONSTRAINT STUD_DETA_PK
    PRIMARY KEY (STUD_DETA_ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding PRIMARY Constraint To VIDEOS Table
```

```
ALTER TABLE VIDEOS ADD (
    CONSTRAINT VIDEO_PK
    PRIMARY KEY (ID)
USING INDEX
PCTFREE  10
)
/

PROMPT Adding FOREIGN Constraint To ARTICLESES Table

ALTER TABLE ARTICLESES ADD (
    CONSTRAINT ARTICLES_EDU_UNIT_FK
    FOREIGN KEY (EDU_UNIT_ID)
    REFERENCES  EDU_UNITS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To AUDIOS Table

ALTER TABLE AUDIOS ADD (
    CONSTRAINT AUDIO_EDU_UNIT_FK
    FOREIGN KEY (EDU_UNIT_ID)
    REFERENCES  EDU_UNITS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To COURSES Table

ALTER TABLE COURSES ADD (
    CONSTRAINT COURSE_DEPA_FK
    FOREIGN KEY (DEPA_ID)
    REFERENCES  DEPARTMENTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To COURSES Table

ALTER TABLE COURSES ADD (
    CONSTRAINT COURSE_STUD_DETA_FK
    FOREIGN KEY (STUD_DETA_STUD_DETA_ID)
    REFERENCES  STUDENT_DETAILS (
                STUD_DETA_ID)
)
/

PROMPT Adding FOREIGN Constraint To COURSES_EMPLOYEES Table

ALTER TABLE COURSES_EMPLOYEES ADD (
    CONSTRAINT COURSE_EMP_COURSE_FK
    FOREIGN KEY (COURSE_ID)
    REFERENCES  COURSES (
                ID)
)
/

PROMPT Adding FOREIGN Constraint To COURSES_EMPLOYEES Table

ALTER TABLE COURSES_EMPLOYEES ADD (
    CONSTRAINT COURSE_EMP_EMPL_FK
    FOREIGN KEY (EMPL_ID)
    REFERENCES  EMPLOYEES (
                ID)
)
/

PROMPT Adding FOREIGN Constraint To EDU_UNITS Table

ALTER TABLE EDU_UNITS ADD (
    CONSTRAINT EDU_UNIT_COURSE_FK
    FOREIGN KEY (COURSE_ID)
    REFERENCES  COURSES (
                ID)
)
/

PROMPT Adding FOREIGN Constraint To EMPLOYEES Table

ALTER TABLE EMPLOYEES ADD (
    CONSTRAINT EMPL_DEPA_FK
    FOREIGN KEY (DEPA_ID)
    REFERENCES  DEPARTMENTS (
                ID)
)
/

PROMPT Adding FOREIGN Constraint To EXAMS Table
```

```
ALTER TABLE EXAMS ADD (
    CONSTRAINT EXAM_COURSE_FK
    FOREIGN KEY (COURSE_ID)
    REFERENCES  COURSES (
             ID)
)
/

PROMPT Adding FOREIGN Constraint To EXERCISES Table

ALTER TABLE EXERCISES ADD (
    CONSTRAINT EXERCISE_COURSE_FK
    FOREIGN KEY (COURSE_ID)
    REFERENCES  COURSES (
             ID)
)
/

PROMPT Adding FOREIGN Constraint To EXERCISE_REPLIES Table

ALTER TABLE EXERCISE_REPLIES ADD (
    CONSTRAINT EXER_REPL_EXERCISE_FK
    FOREIGN KEY (EXERCISE_ID)
    REFERENCES  EXERCISES (
             ID)
)
/

PROMPT Adding FOREIGN Constraint To EXERCISE_REPLIES Table

ALTER TABLE EXERCISE_REPLIES ADD (
    CONSTRAINT EXER_REPL_STUD_DETA_FK
    FOREIGN KEY (STUD_DETA_STUD_DETA_ID)
    REFERENCES  STUDENT_DETAILS (
             STUD_DETA_ID)
)
/

PROMPT Adding FOREIGN Constraint To MAILS Table

ALTER TABLE MAILS ADD (
    CONSTRAINT MAIL_EMPL_FK
    FOREIGN KEY (EMPL_ID)
    REFERENCES  EMPLOYEES (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To MAILS Table

ALTER TABLE MAILS ADD (
    CONSTRAINT MAIL_EMPL_RECEIVES_FK
    FOREIGN KEY (EMPL_ID_RECEIVES)
    REFERENCES  EMPLOYEES (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To MAILS Table

ALTER TABLE MAILS ADD (
    CONSTRAINT MAIL_STUDENT_FK
    FOREIGN KEY (STUDENT_ID)
    REFERENCES  STUDENTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To MAILS Table

ALTER TABLE MAILS ADD (
    CONSTRAINT MAIL_STUDENT_WRITES_FK
    FOREIGN KEY (STUDENT_ID_WRITES)
    REFERENCES  STUDENTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To NOTESES Table

ALTER TABLE NOTESES ADD (
    CONSTRAINT NOTES_EDU_UNIT_FK
    FOREIGN KEY (EDU_UNIT_ID)
    REFERENCES  EDU_UNITS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To PROJECTS Table
```

```
ALTER TABLE PROJECTS ADD (
    CONSTRAINT PROJECT_COURSE_FK
    FOREIGN KEY (COURSE_ID)
    REFERENCES  COURSES (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To PROJECT_REPLIES Table

ALTER TABLE PROJECT_REPLIES ADD (
    CONSTRAINT PROJ_REPL_PROJECT_FK
    FOREIGN KEY (PROJECT_ID)
    REFERENCES  PROJECTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To PROJECT_REPLIES Table

ALTER TABLE PROJECT_REPLIES ADD (
    CONSTRAINT PROJ_REPL_STUD_DETA_FK
    FOREIGN KEY (STUD_DETA_STUD_DETA_ID)
    REFERENCES  STUDENT_DETAILS (
            STUD_DETA_ID)
)
/

PROMPT Adding FOREIGN Constraint To REPLIES Table

ALTER TABLE REPLIES ADD (
    CONSTRAINT EXAM_REPLY_EXAM_FK
    FOREIGN KEY (EXAM_ID)
    REFERENCES  EXAMS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To REPLIES Table

ALTER TABLE REPLIES ADD (
    CONSTRAINT EXAM_REPLY_STUD_DETA_FK
    FOREIGN KEY (STUD_DETA_STUD_DETA_ID)
    REFERENCES  STUDENT_DETAILS (
            STUD_DETA_ID)
)
/

PROMPT Adding FOREIGN Constraint To STUDENTS Table

ALTER TABLE STUDENTS ADD (
    CONSTRAINT STUDENT_DEPA_FK
    FOREIGN KEY (DEPA_ID)
    REFERENCES  DEPARTMENTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To STUDENT_DETAILS Table

ALTER TABLE STUDENT_DETAILS ADD (
    CONSTRAINT STUD_DETA_EMPL_FK
    FOREIGN KEY (EMPL_ID)
    REFERENCES  EMPLOYEES (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To STUDENT_DETAILS Table

ALTER TABLE STUDENT_DETAILS ADD (
    CONSTRAINT STUD_DETA_STUDENT_FK
    FOREIGN KEY (STUDENT_ID)
    REFERENCES  STUDENTS (
            ID)
)
/

PROMPT Adding FOREIGN Constraint To VIDEOS Table

ALTER TABLE VIDEOS ADD (
    CONSTRAINT VIDEO_EDU_UNIT_FK
    FOREIGN KEY (EDU_UNIT_ID)
    REFERENCES  EDU_UNITS (
            ID)
)
/
```

REFERENCES

[1] W. Dick, L. Carry, *The Systematic Design of Instruction.* Harper Collins Publishers, 1990.

[2] T. McManus, *Special Considerations for designing Internet based education. Technology and Teacher Education Annual.* Willis, D., Robin, B., Willis, J. (Eds); Charlottesville, VA: Association for Advancement of Computing in Education, 1995.

[3] M. D. Merrill, R. I. Goodman, "Selecting Instructional Strategies and Media," *National Special Media Institutes,* Washington D.C, 1972.

[4] R. A. Reiser, R. M. Gagne, *Selecting Media for Instruction.* Educational Technologies Publications, 1983.

[5] R. J. Spiro, P. J Feltovich, M. J. Jacobson, R. L. Coulson, *Cognitive Flexibility, Constructivism, and Hypertext: Random Access Instruction for Advanced Knowledge Acquisition in Ill-Structured Domains.* In T. Duffy & D. Jonassen (Eds), *Constructivism and the Technology of Instruction.* Erlbaum Assoc. Publishers 1991.

[6] R. J. Spiro, J. C. Jehng, *Cognitive Flexibility and Hypertext: Theory and Technology for the Nonlinear and Multidimensional Traversal of Complex Subject Matter.* In D. Nix & R. J. Spiro , *Cognition, Education, and Multimedia: Exploring Ideas in High Technology.* Erlbaum Assoc. Publishers 1990.

[7] Armand Seguin, Cynthia Seguin, *Window to the World.* Vocational Education Journal. http://www.emporia.edu/idt/it744/Windows.htm (1995)

[8] T.F. McManus, "Delivering Instruction on the World Wide Web." http://ccwf.cc.utexas.edu/~mcmanus/wbi.html (4/1998)

[9] Oracle Corporation's Documentation For Designer/2000, Developer/2000, Oracle RDBMS, Web Server 2.1.