

California State University, San Bernardino

CSUSB ScholarWorks

Theses Digitization Project

John M. Pfau Library

2004

The web-based database management system for the computer science graduate program

Dung Tien Vu

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Vu, Dung Tien, "The web-based database management system for the computer science graduate program" (2004). *Theses Digitization Project*. 2557.

<https://scholarworks.lib.csusb.edu/etd-project/2557>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

THE WEB-BASED DATABASE MANAGEMENT SYSTEM
FOR THE
COMPUTER SCIENCE GRADUATE PROGRAM

A Project
Presented to the
Faculty of
California State University,
San Bernardino

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science


by
Dung Tien Vu
March 2004

THE WEB-BASED DATABASE MANAGEMENT SYSTEM
FOR THE
COMPUTER SCIENCE GRADUATE PROGRAM

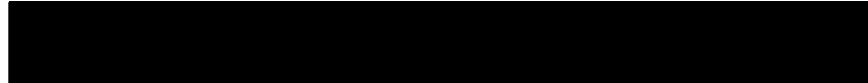
A Project
Presented to the
Faculty of
California State University,
San Bernardino

by
Dung Tien Vu
March 2004

Approved by:


Dr. Josephine G. Mendoza, Computer Science

3-17-04
Date


Dr. Keith Schubert


Dr. Kerstin Voigt

ABSTRACT

Based on successful experience with the existing database system using Microsoft Access at the Department of Computer Science, California State University at San Bernardino, we propose a web-based database management system using Oracle 9i. This project will provide faculty and students a secure access to graduate student resources. The project is designed to be an integrated system serving many aspects of a graduate program. New features can be added without significant modification. The project covers database design, web development, security, migration and deployment of the new system. The project implements the following concepts: dynamic roles and functions, dynamic grouping, task assignment, automatic document generation, automatic database monitoring, and dynamic privileges verification. The Web-Based Database System for the Computer Science Graduate Study Program implements the vector data structure, SQL language, Java Server pages, Java Bean, relational database model, and XML technology for a production application.

ACKNOWLEDGMENTS

First of all, I would like to express my special thanks to Dr. Josephine G. Mendoza, who assisted me in envisioning an integrated management system for the graduate program at the Department of Computer Science. She has advised me in every aspect of the project. My sincere thanks to Dr. Keith Schubert, who I considered as my second advisor, not only gave me valuable advice in security, but also encouraged me to complete the project of highest value. I also express sincere thanks to Dr. Kerstin Voigt, who got me admitted into the MS program. Dr. Voigt has been very enthusiastic with my project and has offered me every support. My sincere thanks to Dr. Arturo Concepcion, Dr. Owen Murphy, Dr. George Georgiou, Dr. Kay Zemoudeh, Dr. Richard Botting, and Dr. Yasha Karant who have given the knowledge and support to bring this project to fruition.

I am immensely grateful to my father, my mother for their guidance and sacrifice. My gratitude is also bestowed to my siblings, who give me valuable encouragement and support to pursue my education, especially, my brother's family: Drs. Khue Vu, Thu, Truong, and Hieu, who have shared not only difficulties but happiness as well in this US soil.

Last but not the least, I thank my wife and my son, who are happy with my dream to pursue higher education.

The support of the National Science Foundation under award 9810708 is gratefully acknowledged.

TABLE OF CONTENTS

| | |
|--|------|
| ABSTRACT | iii |
| ACKNOWLEDGMENTS | iv |
| LIST OF FIGURES | viii |
| CHAPTER ONE: INTRODUCTION | |
| 1.1 Introduction | 1 |
| 1.2 Purpose of the Project | 2 |
| 1.3 Limitations | 2 |
| 1.4 Organization of the Project | 3 |
| CHAPTER TWO: SOFTWARE REQUIREMENTS SPECIFICATION | |
| 2.1 Introduction | 4 |
| 2.1.1 Scope | 4 |
| 2.2 Overall Description | 4 |
| 2.2.1 Product Perspective | 5 |
| 2.2.2 Product Functions | 7 |
| 2.2.3 User Characteristics | 13 |
| 2.3 Specific Requirements | 14 |
| 2.3.1 External Interface Requirements | 14 |
| 2.3.2 Functional Requirements | 15 |
| 2.3.3 Performance Requirements | 15 |
| 2.3.4 Software System Attributes | 15 |
| CHAPTER THREE: DESIGN AND IMPLEMENTATION | |
| 3.1 System | 17 |
| 3.1.1 Authentication and Authorization | 17 |

| | | |
|--|--|----|
| 3.1.2 | Updating and Retrieving Information from the Oracle Database..... | 17 |
| 3.1.3 | Processing | 18 |
| 3.1.4 | Messaging | 19 |
| 3.1.5 | Connectivity and Transmission | 19 |
| 3.2 | Architecture | 20 |
| 3.2.1 | Oracle Database | 20 |
| 3.2.2 | Web Components | 20 |
| 3.2.3 | Java Beans | 22 |
| 3.3 | Design Details | 24 |
| 3.3.1 | Oracle Database System | 24 |
| 3.3.2 | Functionality Modules | 38 |
| 3.3.3 | Security | 62 |
| CHAPTER FOUR: DEPLOYMENT | | |
| 4.1 | System Requirements | 66 |
| 4.2 | Installation | 67 |
| 4.2.1 | Installation of Servers | 67 |
| 4.2.2 | Installation of Web Component | 67 |
| 4.2.3 | Migration from the Existing Database System..... | 67 |
| CHAPTER FIVE: CONCLUSION AND FUTURE DIRECTIONS | | |
| 5.1 | Technology Highlights and Conclusion | 71 |
| 5.2 | Extensions | 73 |
| APPENDIX: LIST OF PROGRAM FILES | | 75 |
| REFERENCES | | 84 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1: The Use Case Diagram | 9 |
| Figure 2: Class Diagram of Public Functions and Java Beans | 23 |
| Figure 3: UML Model for Roles and Functions | 41 |
| Figure 4: UML Model for Grouping | 45 |
| Figure 5: Pseudo_Codes of Storing SQL Logic | 45 |
| Figure 6: Pseudo-Codes of SQL Reconstruction | 46 |
| Figure 7: UML Model for Graduate Students | 48 |
| Figure 8: UML Model for Documentation | 50 |
| Figure 9: Pseudo_Codes to Generate Letter of Classified Status | 52 |
| Figure 10: UML Model for Courses and Grading | 54 |
| Figure 11: UML Model for Task | 57 |
| Figure 12: Pseudo_Codes for GPA Monitor | 60 |
| Figure 13: Pseudo_Codes to Monitor Contact Update ... | 60 |
| Figure 14: Pseudo_Codes to Monitor Academic Standing | 61 |
| Figure 15: Pseudo_Codes for Task Renewal | 62 |
| Figure 16: Recommended Deployment | 66 |
| Figure 17: A Migration Flowchart | 68 |

CHAPTER ONE

INTRODUCTION

1.1 Introduction

The existing database management system for the graduate program in the Department of Computer Science at California State University was implemented in 2001 as a stand-alone system using Microsoft Access 2000. The system has these features: student information search, pre-defined and "any kind" ad-hoc report generation, CSCI 698 Extended Learning list management, statistics generation, automatic computation of GPA (quarter and/or cumulative), graduate committee management, and weekly automatic smart backup.

Although this existing system has efficiently assisted in the information and processing needs of the graduate coordinator for advising (GCA) for the Master of Science in Computer Science (MS CSCI) degree program, it has limitations.

The existing system, located in the department office is only available for access by the GCA and the MS CSCI program assistant (PA). To get reports and information on graduate students, faculty and students need to refer these requests to the PA. Course types (core, elective and

prerequisite) are fixed for specific courses. The number of courses taken is limited (five courses, five electives and nine prerequisite). The number of faculty in a committee is also limited to three. Contact information of the students (current address, email, phone) is kept up-to-date after a student has filled out a paper personal profile sheet.

1.2 Purpose of the Project

This project, "Web-based Database Management System for the Computer Science Graduate Program", implements an integrated system which provides all management functionalities of the existing database management system, allows faculty and graduate students Internet access to appropriate information from the database, and also acts as a "communication hub" between faculty and graduate students. This facilitates making and finding out about announcements, messaging, scheduling appointment, and assigning tasks.

The system also serves as a prototype that can be adapted by different graduate programs without significant modifications.

1.3 Limitations

The project designs and implements a robust, flexible and portable system and creates a framework for many

desirable features to be integrated. Several important concepts and function modules have been developed such as dynamic roles and functions, grouping, document generation, task assignment, and automatic monitoring. Due to the complexity and time constraints the announcement and schedule appointment modules will be left as future enhancements to the system.

1.4 Organization of the Project

This document is organized in five chapters: (1) Introduction, (2) Software Requirements Specification, (3) Design and Implementation, (4) Deployment and (5) Technology Highlights, Conclusion and Future Development. The appendix provides program files used in this system.

CHAPTER TWO
SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Introduction

The scope of the project will be described in Section 2.1.1. Section 2.2 provides an overall description of the product while Section 2.3 details the specific requirements.

2.1.1 Scope

The proposed web-based database system is an improvement of the existing Master of Science in Computer Science stand-alone database management system. The existing database system uses Microsoft Access 2000 and is maintained in a standalone workstation in the MS Program Assistant's office. This project uses Oracle 9i as a database engine and allows faculty and graduate students to have a secure Internet access to appropriate information stored in the database. This project will also implement automatic features necessary to manage and monitor the database system and generate alerts and PDF documents.

2.2 Overall Description

The existing database management system for the graduate program in the Department of Computer Science was designed in 2001 and developed using Microsoft Access 2000

by the Graduate Coordinator for Advising, Dr. Josephine G. Mendoza and me with funding from the National Science Foundation Minority Institution Infrastructure (NSF-MII) Grant Program.

This new version is designed to open access to the database information to Computer Science Department faculty and graduate students via the web-based approach. The system is implemented in Oracle 9i and takes advantage of Oracle 9i features in particular, security. This version consists of seven modules: (1) Dynamic Roles and Functions, (2) Dynamic Grouping, (3) Search and Update Student Information, (4) Courses and Grades, (5) Document Generation, (6) Task Assignment, and (7) Resident and Scheduled System Tasks.

The system will be web-based, and will be accessible by all authorized users using any modern web browsers (e.g. IE or Netscape). The system requires high security that allows access by authorized users only. Communication between users at the client side and the system at the server side must be secured. Student information must be protected, but must be convenient for access and update.

2.2.1 Product Perspective

The system will be implemented using JSP, Java Beans using Java 1.3 or higher version. Oracle Application Server

9i Release 9.0.3 and Oracle Database 9i Release 9.0.2.0.1 are required with the latest patches.

2.2.1.1 System Interfaces. The system with the Oracle Application Server with HTTP and Object Container for J2EE (OC4J) at the server side interacts with a user at the client side in the following manner:

- (1) User starts access to the database system by entering the URL in the browser.
- (2) System will authenticate and authorize the user using a user ID and password. If a user is authenticated, the system will identify this user's corresponding role and direct the user to his or her customized home page. If authentication fails, the system will issue an error message and will require the user to login again.
- (3) The authenticated and authorized user will have privileged access to all functions defined according to the user's role. Any attempt by the user to access an unauthorized page will be denied.
- (4) Based on the user's response, the system may access the Oracle database, e-mail server, document generation processor and return the result or message back to the user through an HTML page.
- (5) System automatically monitors the database system. For example, an update of a student's grades may trigger

the GPA monitor module. If the computed GPA is lower than a pre-defined threshold (i.e. 3.0), the system will send an alert to the PA and the GCA. The system also automatically executes processes at certain frequencies.

2.2.1.2 User Interfaces. All users may use any web browser to interface from the client side. There is no restriction on the operating system used on the client side.

2.2.1.3 Software Interfaces. Software interfaces are provided with the Oracle Application 9iAS Release 9.0.3 and Oracle Database 9i Release 9.0.2.0.1. Oracle is a trademark of the Oracle Corporation. Apache is a trademark of Apache Formatting Object Processor. Qmail is a trademark of Qmail.

2.2.1.4 Communication Interfaces. Communication interfaces between the server side and the client side is implemented with secure Hypertext Transfer Protocol (HTTP) and Secure Socket Layer (SSL).

2.2.1.5 Memory Constraints. Oracle Server 9i and Application 9iAS server needs at least 512MB of internal memory. Qmail server needs at least 256MB of internal memory.

2.2.2 Product Functions

There would be no fixed functions and roles. In addition to the basic functions such as login, search and

update student information, new functions can be added into the system and assigned to different roles. Following is the use case diagram showing the roles and the functions.

USE CASE DIAGRAM

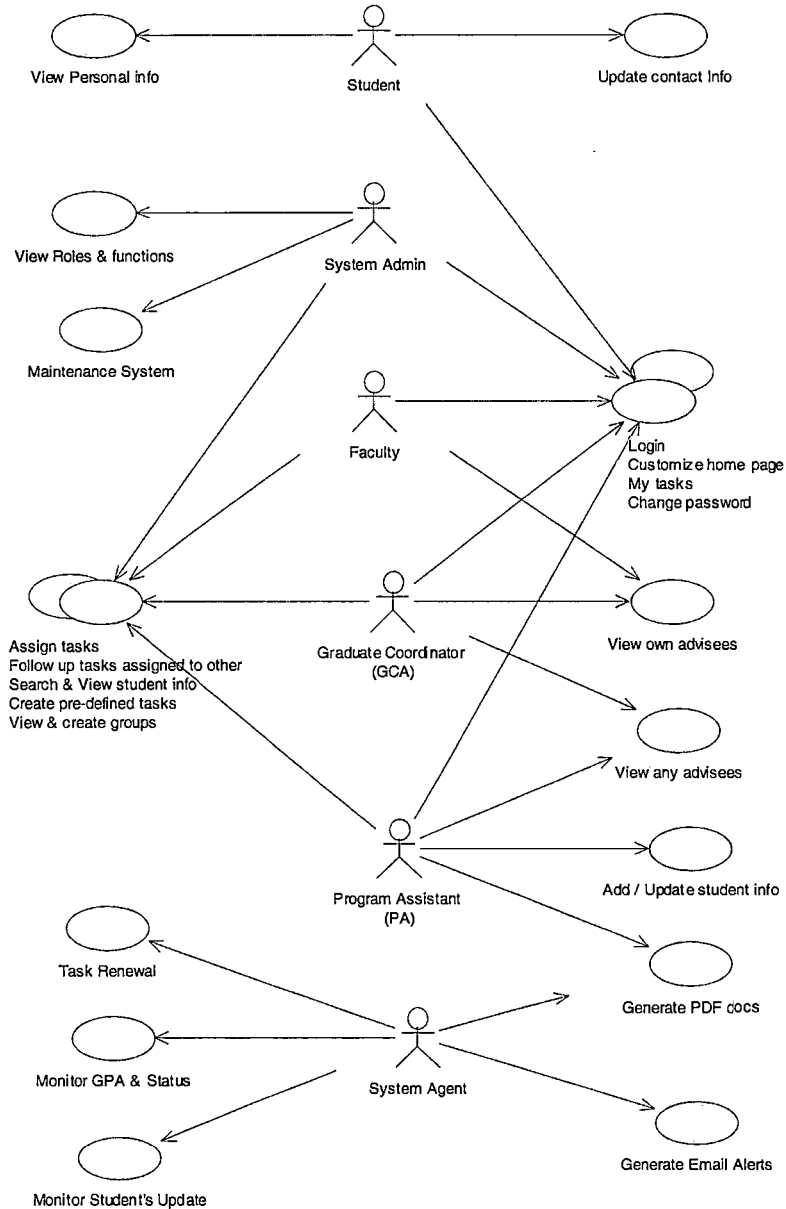


Figure 1. The Use Case Diagram

2.2.2.1 Login. Authenticate and authorize a user with user ID and password. Depending on the user's role, he or she is given the function privileges appropriate to the user's role.

2.2.2.2 Assign Tasks. Assign tasks to program staff, graduate students and department faculty. After a task is assigned the task assignee is optionally notified by email, and he or she can view the assigned task in detail (via My Task function icon at login). A task can be assigned to a group of users (see Create & View Group) in which every member in the group will be assigned the same task automatically.

2.2.2.3 My Tasks. View tasks that have been assigned to the user. The user can view the task details, report progress and result, and view the evaluation when the task is completed. A user is not allowed to remove or deny a task unless the task is removed by the task assigner.

2.2.2.4 Follow-up Tasks Assigned to Others. Monitor tasks, which have been assigned to other users. Tasks are categorized for staff, student and system agents.

The assignor can edit a task after it has been created, and evaluate the task when it is reported as completed. When a task is removed or evaluated, it is removed from the task list of the assignee.

2.2.2.5 Change Password. Allow a user to change password by requiring the user to resubmit the existing user ID and password for verification before a new password can be entered.

2.2.2.6 Create and View Group. Create a new group and view members in the groups. A group is defined by specifying selection criteria. The selection criteria may consist of many fields combined with AND/OR Boolean operators. Each group has a distinctive name and is saved for later viewing or editing. Only the group owner/creator can edit or remove a group. When a group is created as private, the group cannot be viewed by other users. On the other hand, a group created as public can be viewed by any other users. A user can assign a task to a group by declaring the group name in the Assign Task function module.

2.2.2.7 Customize Home Page. Customize a home page to appear as the first page after a user successfully logins.

2.2.2.8 Roles and Functions. Create a new role, or change a role for any user. This can only be done by a user (Admin User) given an 'administrator' role. Admin User can assign or remove a function for a role. A function is developed and registered with a JSP page. After a new

function is developed, the Admin User can register this new function and assign it to a role.

2.2.2.9 View Student Info. Display student's information. A user can select a student from a list or enter a search word as either the student's first name or last name. When the search word matches many students, the system will display a list of matching records.

2.2.2.10 View Own Advisees. Display information on advisees of the faculty advisor. The faculty advisor is either the student's advisor or the student's graduate committee member. The advisees to be displayed can either be currently active graduate students or have already graduated (i.e. alumni).

2.2.2.11 View Any Advisees. Display information on any advisee including advisees of other faculty members. This function can be done only by the GCA.

2.2.2.12 Update Contact Information. Allow a graduate student to modify contact information -- mailing address, e-mail, and telephone number. The system agent will notify the PA which information has been updated by the student.

2.2.2.13 Update Student Information. Modify student information -- status, personal information, course grades (core, elective, prerequisite) and advisor and graduate

committee members. This function is assigned to the authorized PA role.

2.2.2.14 Create Pre-defined Tasks. To expedite assignment of frequently used tasks and standardize regulations and procedures, an authorized role can register the tasks in the pre-defined task pool, and assign to the appropriate users.

2.2.2.15 Add New Student. Insert new student information into the database system. This function is assigned to the PA.

2.2.2.16 Document Generation. Automatically generates appropriate PDF documents. A letter of probation is generated when a student's GPA is lower than 3.0. A letter of Classified status or Advanced to Candidacy status is generated when a student's academic standing status is changed to classified or advanced to candidacy.

2.2.2.17 View Personal information. Student users can view only their own information including personal information, academic progress, courses and grades.

2.2.3 User Characteristics

The following capabilities are assumed for the various users defined in the system.

2.2.3.1 Application Users. Include graduate student, MS program assistant, faculty member, and graduate program coordinators who know their role's capabilities.

2.2.3.2 System Administrator. Install and administers the Oracle 9i database system, the Oracle 9i Application Server and the Qmail server.

2.2.3.3 Application Developer. Needs to have knowledge and skills in web application (HTML, JSP, Java Bean, Java Script) and database system with Oracle relational database schema, SQL language, Oracle procedures and triggers. The developer also needs to know how to develop queries using Microsoft Access to migrate information from the existing database system.

2.3 Specific Requirements

2.3.1 External Interface Requirements

After a user starts the browser with the database management system's URL, the users sees the login page. When a user is authenticated, he or she must have the appropriate role and all privileges for functions defined and authorized for the role.

2.3.2 Functional Requirements

All functions of the system must provide correct results, and all data errors or inappropriate access will not cause the functions to crash.

2.3.3 Performance Requirements

The graduate system will be able to serve multiple users. Faculty members and graduate students in the Department of Computer Science can simultaneously access the Oracle Application Server and the Oracle Database Server. The problem of memory leak should be avoided.

2.3.4 Software System Attributes

2.3.4.1 Reliability. The system should handle failure when the Oracle database server is down. The system must provide sufficient connection to the database system to maintain high availability.

2.3.4.2 Security. The system must be protected from any security risks. The Oracle database server as well as the web server must be protected. A user will be able only to access function pages authorized for the user. Transmission over the Internet must be encrypted.

2.3.4.3 Maintainability. The database system and the web system should be designed to cope with changes and a variety of requirements that will not require a significant modification.

2.3.4.4 Portability. The system should be able to expand its service to different departments without significant modification.

CHAPTER THREE

DESIGN AND IMPLEMENTATION

3.1 System

The web-based database management system for the Computer Science graduate program consists of five main system components: (1) Authentication and Authorization, (2) Updating and Retrieving Information from the Oracle Database, (3) Processing, (4) Messaging, and (5) Connection and Transmission. These are discussed in detail in the following sections.

3.1.1 Authentication and Authorization

This component authenticates a user with a user ID and password at login. Based on a user's role, it will grant the user with access privilege codes for the authorized pages. When a user attempts to access a secure web page, this component will verify the access privilege required for the page. It will log the user out if the user does not have an appropriate privilege.

3.1.2 Updating and Retrieving Information from the Oracle Database

This component retrieves and updates information from the Oracle database. It consists of statements in SQL language and is called by Java methods.

3.1.3 Processing

Processing components is carried out at the server side on the Oracle application server, Oracle database server and Apache Formatting Object Processor (FOP).

3.1.3.1 Processing On the Application Server. This component processes requests from a user browser and generates presentation results in HTML. This component consists of JSP pages. Depending on the processing logic, this component may call other system components, such as updating or retrieving information from the Oracle database.

3.1.3.2 Processing On the Oracle Database Server. This component processes information on the Oracle Server. There are two categories of processing: Oracle triggers and regular procedures in SQL language.

- Oracle Triggers. This component consists of event-based procedures in SQL language, which automatically execute when updates of the database satisfy certain pre-defined conditions.

- Oracle Procedures. This component consists of regular procedures in SQL language, which is called by other procedures or by other triggers.

3.1.3.3 Processing On the Formatting Object Processor. This is an external executable component, which generates

PDF documents. The FOP will render an XML based formatting object using XML Style Sheet Formatting Object (XSLFO) style sheet.

3.1.4 Messaging

This component dispatches email messages using Qmail server. Among the available mail servers, the Qmail server is selected because of its better security features. The Qmail has been developed from scratch with security awareness since 1997, and no security holes have been found. [21]

3.1.5 Connectivity and Transmission

The connectivity component generates a connect pool between the Oracle Application Server and the Oracle Server. After a user completes an update or retrieval of information from the database, the connection will be closed and returned back to the connection pool to provide connection requested by other users.

The transmission component facilitates communication between a user and the system over the Internet. The transmission is secured with an encryption using Secure Socket Layer.

3.2 Architecture

The system includes two important architecture components: web application and database components. These components are described starting in Section 3.2.1 and ending in Section 3.2.3.

3.2.1 Oracle Database

This component includes the Oracle database engine that manages 27 relational tables. This component is described in detail in the Detailed Design section.

3.2.2 Web Components

This architecture component includes web pages, user-defined public functions, and Java Beans. Web pages are the core part in handling a user's request while Java Beans assist in processing and maintaining user and student information throughout a web session.

3.2.2.1. Web Pages. These pages facilitate most of the system's features. They consist of JSP pages, HTML and JavaScript. They receive requests either via instructions or data from users, process and generate HTML results. To complete the tasks, they may refer to other JSP pages, user-defined public functions, Java Beans, and Oracle statements or procedures. Please see the appendix for a list of 160 file names.

The presentation HTML integrated in JSP page uses Oracle Cascade Style Sheet (CSS), which provides a consistent and standardized presentation format for all the web pages. Some JSP pages are embedded with JavaScript codes that provide convenient responses on client browsers.

3.2.2.2 User-Defined Public Functions. Besides Java methods imported from Oracle and Java packages, this component includes the most frequently used user-defined methods.

String GetSession(). get a string attribute from session object, and handle null value.

String GetRequest(). get a string parameter from request object, and handle null value.

String GetResult(). get a string field from a result set.

ResultSet GetResultSet(). get a result set by querying the database.

String SubReplace(). a recursive method to replace a string with a string.

String TextHTML(). convert special characters < > & into HTML presentation string.

String RandString(). generate random String with a desired length.

Boolean CheckUserPrivileges(). Check user privileges if they satisfy a privilege token.

3.2.3 Java Beans

Java Beans used in the system are categorized into user-defined Java Beans and Oracle Utility Beans:

3.2.3.1 User-Defined Java Beans. Include UserBean bean and studentBean bean with the following capabilities.

- UserBean. Maintains user information throughout a web session. When a user (either a faculty member or a student) logs in, the Bean stores user's information in session object and makes them available for any web pages. UserBean also protects sensitive information such as user ID and password by not exposing the information as parameters in the URL request.

- StudentBean. Maintains student information. StudentBean significantly assists in viewing and updating student information across web pages.

PUBLIC FUNCTIONS &
JAVA BEANS

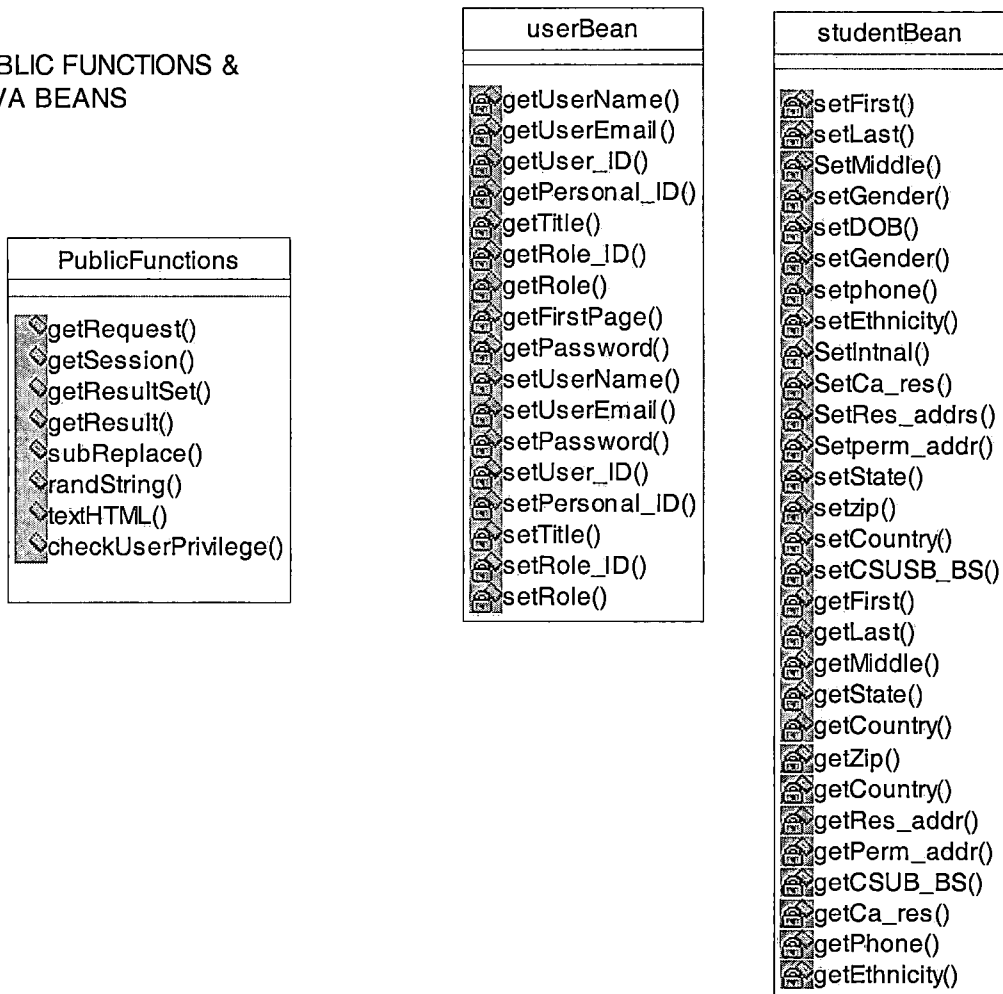


Figure 2. Class Diagram of Public Functions and Java Beans

3.2.3.2 Utility Beans. Include SendMailBean and

OracleConnectionBeanImple beans:

- Oracle SendMailBean. Implements the connection with the Qmail Server and relays email message.

- OracleConnectionCacheImpl. Generates a connection pool with the Oracle Server and distributes the connection to a user as per request. As soon as the user completes accessing the Oracle database that takes just a fraction of a second, the connection is returned to the pool for other connection requests. As a result, one connection can “simultaneously” serve a lot of requests with the connection pools. Connectivity is enhanced with a fixed-Wait-Schema setup, which lets a user wait until a connection is available from the connection pool. Generating a connection to a database server is expensive in terms of time and resources. Therefore, connection pool significantly improves the performance and availability of the Oracle Server.

3.3 Design Details

The design to the Web-based database system consists of the design of the Oracle database system, seven web-based functionality modules, and security.

3.3.1 Oracle Database System

The database system is designed to satisfy the project’s goal in which the system features are data-driven, portable and cope with changes that require minimal modification. Information requirements are analyzed, and

the relational database model includes 26 entities associated to each other in binary and ternary relationships plus five sequences, which are used to generate index keys. All tables are normalized to meet the 3NF.

3.3.1.1 Look-Up Tables. Following are data dictionary of Look-Up tables. Their schema consists of an Identification (ID) field and a value field, which defines the value of the ID. The tables are populated with data when the system is setup. Additional records can be added later.

1. ACADSTANDING. Defines academic standing of students -- probation, conditionally classified, classified, or advanced to candidacy.

```
STANDING_ID      [char(1), Primary key(PK)]
STANDING         [varchar2(30)]
```

2. ADMITSTATUS. Defines admission status of students -- classified, conditionally classified, probation classified, or probation conditionally classified.

```
ADMIT_ID         [char(1), PK]
ADMIT_STATUS     [varchar2(30)]
```

3. COUNTRIES. Defines countries where international students come from.

```
COUNTRY_ID      [char(3) PK]
```

COUNTRY [varchar2(15)]

4.COURSES. Defines courses taught in the graduate program including prerequisite, elective and core courses.

COURSE_ID [varchar2(15) PK]

COURSE_NAME [varchar2(50)]

UNITS [char(1)]

5.COURSETYPES. Defines course types - core, elective and prerequisite core course.

COURSETYPE_ID [char(1) PK]

COURSETYPE [varchar2(15)]

6.CURRENTSTATUS. Defines current status of students -- Active, Incoming, Graduated, Inactive Attended, On Leave, Dismissed, Never Attended, Academic Probation, Withdraw.

CUR_STATUS_ID [char(1) PK]

CUR_STATUS [varchar2(30)]: Current status.

7.ETHNICSORGN. Defines ethnic origin of student.

ETHNICITY_ID [char(1) PK]

ETHNICITY [varchar2(60)]

8.GRADES. Defines grades and corresponding scores .

GRADE [char(1) PK]

SCORES [number(2.3)]

9.TASKSCHEDULE. Defines frequencies for repeated tasks. The frequencies cover the periods: Daily, Every 2 days, Every 3 days, Every 4 days, Every 6 days, Every 10

days, Weekly, Every 2 weeks, Every 3 weeks, Quarterly, Yearly.

SCHEDULE_ID [char(1) PK]
SCHEDULE [varchar2(30)]: Schedule name
SCHEDULE_DAYS [number(3.0)]: Schedule's duration

10.QUARTER. Defines quarters (fall, winter, spring, summer) when courses are taken. In semester system there are only fall and spring values.

QUARTER_ID [char(1) PK]
QUARTER [varchar2(10)]: Fall, Winter , Spring,

Summer.

11.ROLES. Defines roles of users. Currently the system has five roles: Student, Faculty, Coordinator, MS Program Assistant, Administrator and System.

ROLE_ID [char(1) PK]
ROLE [varchar(30)]

12.FUNCTIONS. Defines all registered functions to assign to roles and to users.

FUNCTION_ID [number(3) PK]
FUNCTION [charchar2(40)]: Function name
FUNCTIONPAGE [charchar2(30)]: Function's JSP page
PRIVILEGE_CODE [charchar2(10)]: Function's execution

privilege token

13.XMLDOCS. Defines all pre-defined document templates in XML type to generate PDF documents.

DOC_CODE [char(3) PK]

DOC_DESC [varchar2(50)]: Document descriptions --
Letter of Probation, Letter of Classified Status, Letter of Advanced to Candidacy Status

DOC_XML [xmltype]: XML based document template

14.JOBINTERVALS. Defines frequencies when the system will automatically execute the scheduled system tasks.

INTERVAL_ID [number(2)] PK] Interval ID

DESCRIPTION [varchar2(50)] interval description -Run once, Every day, Every 2 days, Every 3 days, Every 4 days, Every 6 days, Every 10 days, Every Week, Every 2 weeks, Every 3 weeks, every Monday, every Tuesday, every Wednesday...

INTERVAL [varchar2(50)] interval formula in SQL language to compute the date of the next execution of the corresponding interval based on SYSDATE (today's date), for example, every 10 days - SYSDATE + 10, every Monday - NEXT_DATE(SYSDATE, 'MONDAY').

15.TASKSTATUS. Defines status of a task - denied, complete, failed, not yet evaluated, and incomplete.

STATUS_ID [char(1)] Status ID

STATUS: [varchar2(30)] Status description

3.3.1.2 Data Tables. These tables are empty when the system is setup and grows when the system is in used (adding, editing or removing data).

16.GRADSTUDENTS. This is a main table to store student information. Using the relational model most fields are foreign keys with referential integrity constraint with lookup tables. This dramatically reduces the record size and input errors when foreign key fields must satisfy referential integrity constraints.

STD_ID [char(9)]: Student Identification Number
SSN [char(9)]: Social Security Number
LAST [varchar2(20)]: Last name
FIRST [varchar2(20)]: First name
MIDDLE [varchar2(15)]: Middle name
DOB [date]: Date of birth
GENDER [char(1)]: Gender - F / M for female / male
ETHNICITY [char(1), Foreign key]: Ethnicity code with referential integrity constraint with ETHNICSORGN table
EMAIL [varchar2(30)]: E-mail address
PHONE [varchar2(13)]: Phone number
INTNAL [varchar2(30)]: Y if the student is an international student
CA_RES [char(1)]: Y if the student is a California resident

CSUSB_BS [char(1)]: Y if the student obtained a BS degree from CSUSB

RES_ADDRS [varchar2(100)]: Street address where the student is residing

CITY [varchar2(20)] City

STATE [char(2)]: State

ZIP [varchar2(9)]: Zip code

PERM_ADDRS [varchar2(100)]: Home address

COUNTRY[char(3)]: Home country address

QTR_ADMIT [char(1)]: Quarter of admission, has referential integrity constraint with QUARTER table

YR_ADMIT [char(4)]: Year of admission

QTR_CLASSIFIED [char(1)]: Quarter of classification, has referential integrity constraint with QUARTER table

YR_CLASSIFIED [char(1)]: Year the student is classified

ACAD_STAND [char(1)]: Academic standing status, has referential integrity constraint with ACADSTANDING table

CUR_STATUS [char(1)] Current Status, has referential integrity constraint with CURRENTSTATUS table

CUR_QTR_START [char(1)]: Quarter the current status started

CUR_YR_START [char(1)]: Year the current status started, has referential integrity constraint with QUARTER table.

CUR_YR_END [char(4)]: Year when current status will end

QTR_LASTATD [char(1)]: Quarter the student was admitted to the program, has referential integrity constraint with QUARTER table

YR_LASTATD [char(4)]: Last quarter attended, has referential integrity constraint with QUARTER table

TOEFL_WAIVED [char(1)] Y if the TOEFL test requirement has been waived

TOEFL-SCORES [number(5)]: TOEFL score

TOEFL_DATE [date]: date TOEFL was taken

GRE_VERB [number(5,2)]: GRE verbal score

GRE_QUANT [number(5,2)]: GRE quantitative score

GRE_ANAL [number(5,2)]: GRE analytical score

GRE_DATE [date]: Date GRE was taken

GRE_SUBJ [number(5,2)]: GRE Computer Science subject's scores

GRESUBJ_DATE [date]: Date GRE Subject test was taken

QTR_CANDIDACY : Quarter the student is advanced to candidacy, has referential integrity constraint with QUARTER table

YR_CANDIDACY [char(4)]: Year the student is advanced
to candidacy

PRO_THESIS [char(1)]: Type of graduation work - P / T
for project or thesis

TITLE [varchar2(100)]: Title of project / thesis

ORAL_EXAM [varchar2(5)]: Oral exam score

PASS [char(1)]: Y if pass; N if fails

PRESENTAION [date]: date of presentation of project
/thesis

NOTES [varchar2(300)]: Notes / comments

17.STAFF. Stores staff information including faculty,
staff or other employees, who are involved in the graduate
program.

STAFF_ID [char(9) PK]: Staff identification (ID)
number

TITLE [varchar2(20)]: title - Professor, Associate
Professor, Chair, President

LAST [varchar2(20)]: Last name

FIRST [varchar2(20)]: First name

MIDDLE [varchar2(15)]: Middle name

ISFACULTY [char(1)]: Y if the staff member is faculty

EMAIL [varchar2(40)]: E-mail address

PHONE [varchar2(20)]: Telephone number including area
code

POSITION [varchar2(40)]: MS program Assistant or
Graduate Coordinator

18.COMMITTEE. Stores advisor or graduate committee members of a student who has been advanced to candidacy. Each record represents a faculty member for a student. Thus, if a student has one advisor and two committee members, he or she will have a total of three records.

STD_ID [char(1) PK]: Student identification number
STAFF_ID [char(9) FK] Staff-ID of the student's faculty, has referential integrity constraint with STAFF table

ISADVISOR [char(1)]: Y if the faculty is the advisor, blank if the faculty is a committee member.

19.STUDENTCOURSES. Stores courses taken by a student. Each student has as many records as courses he or she has taken.

STD_ID [char(9) PK] Student identification number, has referential integrity constraint with GRADSTUDENTS table

COURSE_ID [varchar2(7), FK]: Course identification number, has referential integrity constraint with COURSES table

QUARTER_ID [char(1), FK]: Quarter the course was taken, has referential integrity constraint with QUARTER table

(STD_ID, COURSE_ID, QUARTER_ID) constitutes the primary key of the table

COURSETYPE [char(1), FK]: Course type code (for core, elective, prerequisite course), has referential integrity constraint with COURSETYPES table

YEAR [char(2)]: Year the course was taken

GRADE_ID [char(2)]: Grade for the course taken, has referential integrity constraint with GRADES table

20.ROLEFUNCTIONS. Stores functions assigned to each user role. Each record represents a function assigned to a role. Therefore, a role will have as many records as functions assigned to the roles.

ROLE_ID [char(1) FK]: Role identification number, has referential integrity constraint with ROLES table

FUNCTION_ID [number(3) FK]: Function assigned to a role, has referential integrity constraint with ROLES table

(ROLE_ID, FUNCTION_ID) constitutes the primary key for the table

21.TASKMANUAL. Defines all pre-defined tasks.

TASK_CODE [char(3) PK] Task identification number

TASK_ACTION [varchar2(20)]: brief description of the task

GUIDE [varchar2(100)]: Guide to complete the task

22.JOBS. Defines pre-defined job for pre-defined tasks. Each task may include one or more jobs.

JOB_ID: [char(5),PK]: Job ID number

JOB:[varchar2(20)]: Job name.

JOB_SCRIPT [varchar2(150)]: Description of the job

23.TASKBOOK. Stores jobs of each pre-defined task. Each record represents a pre-defined job in a pre-defined task. Each pre-defined task has as many records as number of jobs assigned to the task.

TASK_CODE [char(1), FK] Task identification number, has referential integrity constraint with TASKMANUAL table.

JOB_ID [char(5), FK] pre-defined job ID, has referential integrity constraint with JOBS table.

JOB_ORDER [number(2)]: an step-by-step order the job need to be done to complete the task.

24.TASKREGSTR. Store tasks assigned to each student, faculty or staff member. Each record represents a task.

TASK_ID [char(6), PK] Task identification number

TASK_CODE [char(3), FK]: pre-defined task code, has referential integrity constraint with TASKMANUAL table

TASK_OWNER [varchar2(9), FK]: ID of the staff who assigns the task, has referential integrity constraint with STAFF table

TASK_ASSIGNEE [varchar2(9), FK]: ID of the staff or student who needs to complete the task.

DATE_ASSIGNED [date]: Date the task was assigned

DATE_DUE [date] : Date the task is expected to be completed by the assignee

SCHEDULEID [char(1)]: How often the task repeats, has referential integrity constraint with TASKSCHEDULE table.

DATE_COMPLETE [date]: Date the task is actually completed by the assignee

DATE_EXPIRED [date]: Date the task is no longer valid. A task with frequency of weekly, will no longer need to be done after the expiration date.

NOTES [varchar2(100)]: Brief note about the task

STATUS [char(1)]: Status of the task whether -- complete, fails, denied, incomplete, not evaluated -- has referential integrity constraint with TASKSTATUS table.

REPORT [varchar2(100)]: Report of the assignee about the task given

COMMENTS [varchar2(100)]: Task assignor's comments

DOCUMENT [varchar2(100)] Any viewable document related to the task

25.USERS. Stores user account information for faculty, staff, student accounts.

USER_ID [varchar2(10)]: User identification number

PWORD [varchar2(10)]: Password for authentication

ROLE_ID [char(1)]: User's role, has referential integrity constraint with ROLES table.

STAFF_STD_ID [char(9)]: Staff identification number of staff / Student identification number of Student

ISSTAFF [char(1)]: Y if the user is a staff member

FIRSTPAGE [number(3)]: User's customized home page

26.XMLGROUPS. Stores all student groups

GROUP_ID [number(4)]: group identification number

GROUP_OWNER [char(9)]: Staff identification number of the group's owner

GROUP_DESC [varchar2(100)]: Group name

GROUP_CRITERIA [xmltype]: XML document that stores the selection criteria.

GROUP_CAT [char(1)]: Group category - A / P for private or public group

27.DOCUMENTS. Stores electronic documents. Each record represents a document of a student.

STD_ID [char(9) PK] student identification number, has referential integrity constraint with GRADSTUDENTS table

DOC_CODE [char(3) FK] Document template identification number, has referential integrity constraint with XMLDOCS table

DATE_ISSUE [date]: Date the document is created

DOC_XML [XMLTYPE]: Document content in XML format

DOC_TYPE [char(1)]: Document type - P / W / T for PDF, Word, Text document.

DOC_FILE [varchar2(50)] Document 's file name

3.3.2 Functionality Modules

The system features are categorized into eight essential modules. Each module consists of a series of related functions and includes features, module functions and the design of the modules. Each module may also include a UML data model and pseudo-codes. Please refer to the data dictionary for the definition of the tables, which are used in the implementation of the modules.

3.3.2.1 Dynamic User Roles and Functions. Dynamic User Roles and Functions provides a robust mechanism in adding and modifying roles and functions. New roles and functions will be added to the system as needed.

The system administrator and system agent are setup as initial roles. System Administrator will create other roles -- Coordinator, MS Program Assistant, Faculty, Staff, and Student. Coordinator and System Administrator can assign and change roles of any user. When a user is assigned to a role, he or she will be authorized to execute all functions defined for that role.

A function is implemented as one or many linked JSP pages. The developers can develop any new function page and load it into the FUNCTION table. System Administrator can then assign the new function to any role.

These features make the system completely robust and portable. System administrator and developer can design/edit any role and any function to suit many requirements without modifying the application codes.

Due to the dynamic role and function feature, a function can be optionally assigned and removed from any role. The execution privilege, therefore, of a function is no longer fixed to any role but needs a privilege granting mechanism. This mechanism is described in the Security Section.

Functions. The Dynamic Roles and Functions consists of the following functions:

1. Add new Roles. Register new roles into the system.
2. Assign Roles to Users. When a new role is registered with the system, it can be assigned to any staff user.
3. Add/Edit Functions. Register a new function into the system, edit the function's description and the function's JSP pages.
4. Assign / Reassign Functions to User Roles. When a function is assigned to a role, all users with that role

will automatically have execution privilege for the function.

Design. USERS, ROLES, FUNCTIONS, ROLEFUNCTIONS tables are designed with the following relationships: ROLES has M - N relationship with FUNCTIONS. ROLEFUNCTIONS table implements this relationship. USERS table has a 1 - M relationship with ROLES table. (See Figure 3: UML Model for Roles & Functions.)

UML Model of "Rational Rose" software is used to design and illustrate database schema with primary keys and foreign keys. "Strong entity" tables (USERS, GRADSTUDENTS) with primary key have "non-identifying" relationships with other strong entities using primary key - foreign key link. In UML model this relationship is represented as a straight line with cardinality at each end. "Weak entity" tables (ROLEFUNCTIONS), on the other hand, have "identifying" relationships with their parent tables. Their primary keys are composed from primary keys of their parent tables. In UML model this relationship is represented as a straight line with diamond symbol at parent side.

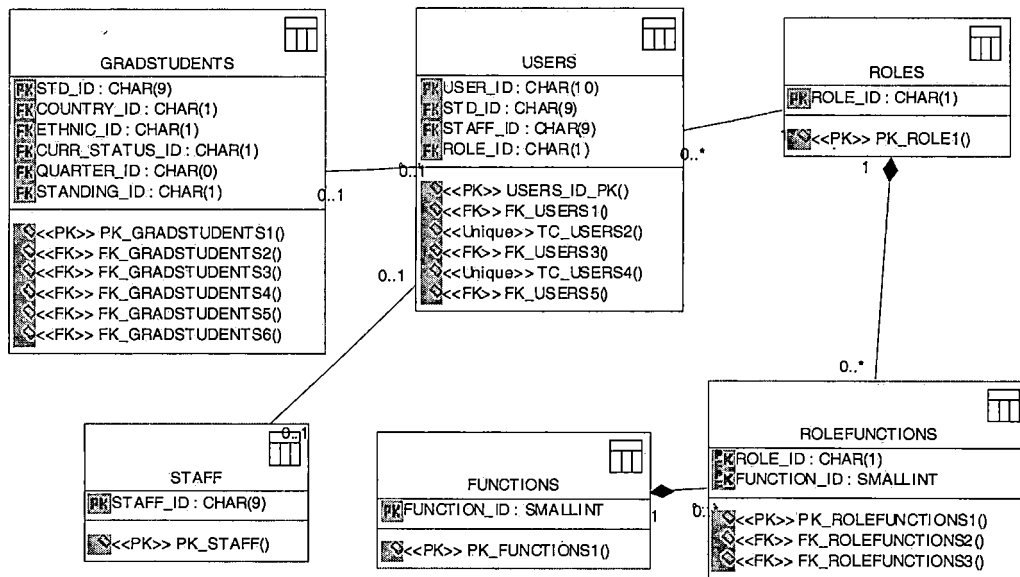


Figure 3. UML Model for Roles And Functions

3.3.2.2 Dynamic Grouping. Dynamic grouping produces a named subset of users that satisfies selection criteria. The subset is not fixed but reflects the database status at run time. The Program Assistant, faculty and program coordinator can create a group by naming the group and defining the selection criteria.

Each group is given a distinctive name and saved for later viewing or editing. When a group is created as private, other users cannot view the group. On the other hand, any other users can view a group created as public.

Grouping has many uses:

- Assist in searching for student(s) who satisfy specific selection criteria.

- Assist in assigning a task to a group of students by assigning a task to the group. The task will be automatically assigned to each group member.

- Assist in producing reports by specifying a student group that meets defined selection criteria. For example, to produce a list of students who have been advanced to candidacy in a specified quarter, we just create a group with the selection criteria and produce a report for this group. The report may be in different formats about desired columns, page layout (landscape/portrait), document type (PDF, Word, text).

- Assist in communicating with a group of students of the same category or with the same interest. When a message / announcement is sent to a group, all its members will receive the message.

Functions. The dynamic grouping has the following functions:

1.Add/ Modify/ Remove Group. a staff user can create a new student group using a selection criteria. The group is saved for later viewing, and can be modified by changing the selection criteria.

2.View Group / Criteria-Based Search. a user can select a group and view its members. A user can also narrow

his or her search for a student(s) by defining selection criteria, which the student(s) will satisfy.

3.Assign Task to Group. a staff member can assign a task to multiple students by assigning task to its group name. The task will be automatically distributed to all students in the group.

Design. STAFF and XMLGROUP tables have the following relationships: STAFF has 1-M relationship with XMLGROUP. (See Figure 4: UML model for grouping.)

The XML data type is used to save selection criteria and restore the SQL logic to produce the query results. How the SQL logic is stored and reconstructed is described in the next section.

- Storing SQL logic. Tags are used to describe a SQL query logic. Each tag is named after a field name, and its attribute represents a relational operator such as =, <, >, <=, >=. To represent a more complicated logic each field can have a maximum number of tags. For example, the query logic for admission status of a student,

```
... ( ADMIT_STATUS = "Classified" ) AND ...
```

is interpreted as the following tags and attributes:

```
...
```

```
<ADMIT_STATUS_PAB ADMIT_STATUS_PAB = "(" />
```

```
<ADMIT_STATUS ADMIT_STATUS_BOOL = ".EQ."
ADMIT_STATUS = "C" />
<ADMIT_STATUS_PAE ADMIT_STATUS_PAE = ")" />
<ADMIT_STATUS_REL ADMIT_STATUS_REL = "AND" />
```

...

If Admission Status is not a selection criterion then these tags are not required. Therefore only tags needed are generated. Combined with other fields these tags are stored in a single XML document as follows:

```
<?xml version="1.0" encoding="ISO-8859" ?>
<CRITERIA>
...
Tags
...
</CRITERIA>
```

This document is stored in a single GROUP_CRITERIA field of XMLTYPE data type of XMLGROUPS table. Each row of the table represents complete selection criteria.

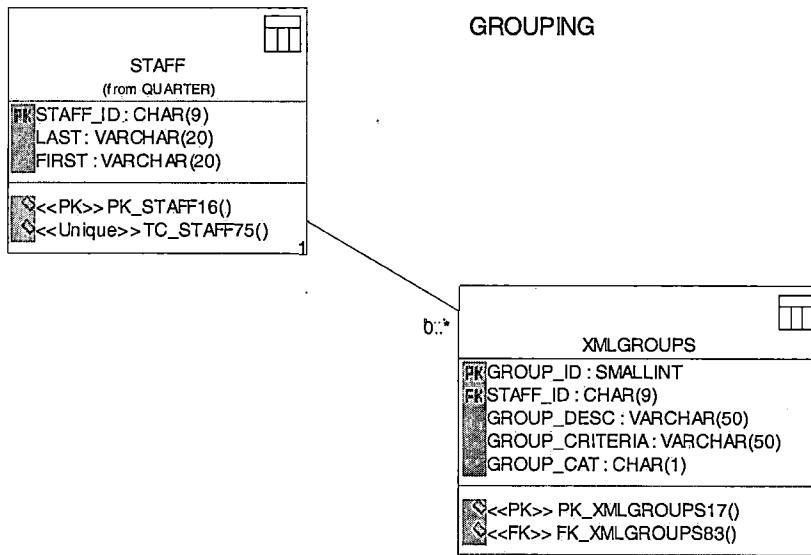


Figure 4. UML Model for Grouping

```

Get parameters for selection criteria from request object
Construct SQL query statement
If execution of the SQL statement fails then
    send a alert and return
else then
    start XML document tag XMLTEXT="<?xml
version="1.0"encoding="ISO-8859"?><CRITERIA>";
    For each parameter if its value is not blank then
        Append its tag and attribute value to XMLTEXT
    Endif
    Add </CRITERIA> closing tag to XMLTEXT
    Insert a new record to XMLGROUPS table where GROUP_CRITERIA field
    is populated with XMLTEXT content
  
```

Figure 5. Pseudo_Codes of Storing SQL Logic

- Reconstruction of SQL logic. SQL logic is extracted using EXTRACT function of the Oracle utility package, and SQL query is constructed to produce query results. Following are the pseudo-codes of the SQL reconstruction

Get GROUP_ID parameter from request object
Retrieve a record from XMLGROUPS table matching GROUP_ID parameter
Extract values of GROUP_CRITERIA XML datatype field
Construct SQL statement
Execute SQL statement to get result of the reconstructed selection criteria

Figure 6. Pseudo-Codes of SQL Reconstruction

3.3.2.3 Search and Update Student Information.

Searching assists a staff user in looking for a specific student or group of students who satisfy a condition. A user can look for a student from (1) a popup name list, (2) a list of students having the same name (first or last name or both of them), or (3) list of students matching a search criteria.

Update student information assists a user in adding new student or updating a student's personal and academic information.

Students can participate in keeping their contact information up-to-date such as current address, phone and e-mail.

Functions. Search and Update Student Information

includes:

1. Name List-based Search. Search for student from a popup student list.

2. Name Keyword-based Search. Search for student by either last name or first name key word or both.

3. Criteria-based Search. Search for student by selection criteria. This function is implemented as a grouping.

4. Update student contact. Students update their contact information.

5. Update student information. Information to be updated include academic summary, personal information, core, elective and prerequisite courses, and graduate committee composition.

6. Add new student. Insert a new student into the database.

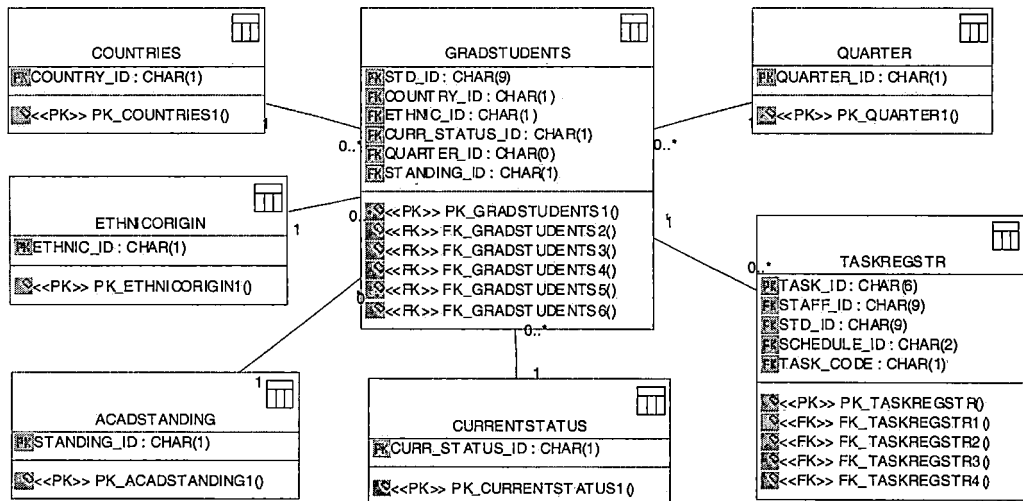


Figure 7. UML Model for Graduate Students

Design. GRADSTUDENTS table uses STD_ID field as a primary key, which is generated from STD_INDEX sequence. Instead of storing the whole value such as advisor's name, course names, quarter, country, ethnic origin and status, the table only stores the foreign keys, which refers to other lookup tables. As a result, the size of the system tables and processing cost are dramatically reduced.

Using foreign keys also enforces referential integrity between two tables that are in 1-M, or M-N, or M-N-P relationships. Values entered, as codes into the foreign key fields must maintain referential integrity with their lookup tables. As a result, data entry is faster and less prone to typing errors.

A generated STD_ID field used as an index key instead of Social Security Number will comply with the requirement of security and privacy protection.

3.3.2.4 Document Generation. PDF Documents are manually or automatically generated when a student update satisfies a pre-defined condition. Currently, the system will generate a letter of probation when a student's GPA drops below than 3.0, a letter of classified status and letter of advanced to candidacy status when a student's status is changed to classified and candidacy accordingly. Related users such as PA, GCA will be notified by e-mail / task to get the print out of the documents.

Functions. Document generation produces the following letters:

1. Letter of Probation
2. Letter Advanced to Candidacy
3. Letter of Classified Status

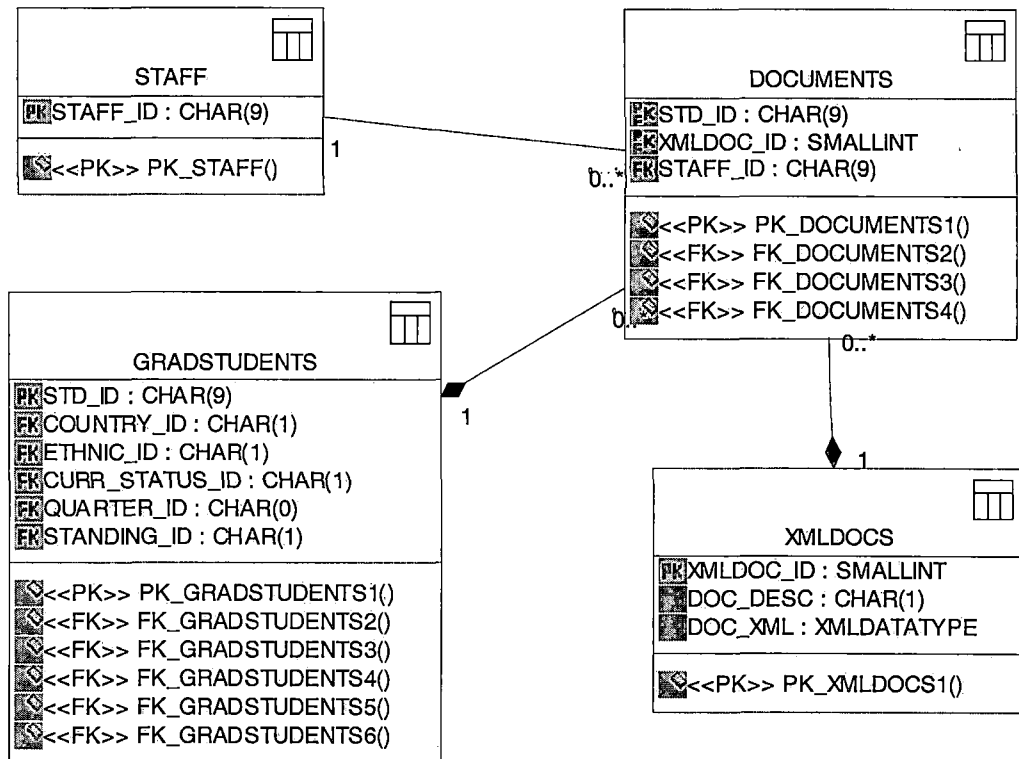


Figure 8. UML Model for Documentation

Design. PDF documents are generated by Apache Formatting Object Processor (FOP). This processor is automatically activated by Oracle triggers (See Resident System Tasks), which monitor activity of the database system.

Executing as an external command, FOP transforms document templates from XML format into PDF documents using Formatting Object (XSLFO) style sheet. The style sheet defines a standard format for generated PDF documents.

Some template XML documents such as "Letter of Classification", "Advanced to Candidacy Letter" have been pre-designed. These template documents include parameters to hold student's information such as name, address, etc. that will be filled when the documents are generated. These template documents are stored in the Oracle XMLDOCS table.

When a student update matches a pre-defined condition, Oracle will trigger a procedure that generates a corresponding XML document. This XML document is transformed into a PDF document using XSLFO style sheet. For example, when a student status is changed to classified, Oracle will trigger a procedure to generate in PDF format the letter informing the student that status has been changed from conditionally classified to classified status. The Oracle trigger also notifies the MS Program Assistant by e-mail and task to print out the PDF document and the Graduate Coordinator to sign the letter. More complicated triggers and procedures are presented in the Trigger Section. Following is the pseudo-code to generate the letter of classified status.

```

Procedure Letter_Classified_Gen {
Begin
  Assign the document code for letter of classification
  if DOCUMENTS table already has the letter for the student then
    return
  endif
  Retrieve the letter template from XMLDOCS table
  Insert the template letter into DOCUMENTS table
  Fill the template with student information: name, address,..
  Call EXECFOP java function that executes an external
    FOP command line to generate PDF document.
  Store the PDF file in DOCUMENTS Folder.
  Alert MS Program assistant about the student's status change
  and printout the letter
End
}

```

Figure 9. Pseudo_Codes to Generate Letter of Classified Status

3.3.2.5 Courses and Grades. Courses and grades provide a flexible way to record a student's academic progress. Courses taken are not limited to a fixed number. Course types (core, elective, prerequisite) are not restricted to any course or any year. A course may be taken as an elective by a student while it could be taken as a prerequisite or a core course by another student. Furthermore, the system also can handle the same course is retaken in different quarters. Handling course grade and course type at student level helps solve a limitation of the existing database system, in which course type for a course is fixed. Number and which courses required for

core, elective and prerequisite may change throughout academic years. As a result, It will cause an inflexibility in recording courses and course types.

This advantage will make the system portable for different graduate programs, where required core, elective and prerequisite courses are totally different.

Entering and updating courses and grades are also efficient and able to avoid data entry mistakes. When a new student is admitted, in addition to the student personal information, the PA will enter all pre-requisite courses that the student needs to fulfill. When a pre-requisite is completed, information is updated by entering the specific grade received and quarter/year that the pre-requisite was taken. At any time, GCA can know which pre-requisite a student has fulfilled along with the cumulative GPA.

Functions. Courses and Grades function has the following capabilities:

1. View and update a student's personal and progress information. authorized users can view a student's personal and academic information. This allows the PA to add a new student and update an existing student record.

2. View and update Core/Elective/Prerequisite courses. user can view all courses and when (quarter, year) the courses were taken.

3. GPA computing. Cumulative GPA is computed at run time to reflect the latest value.

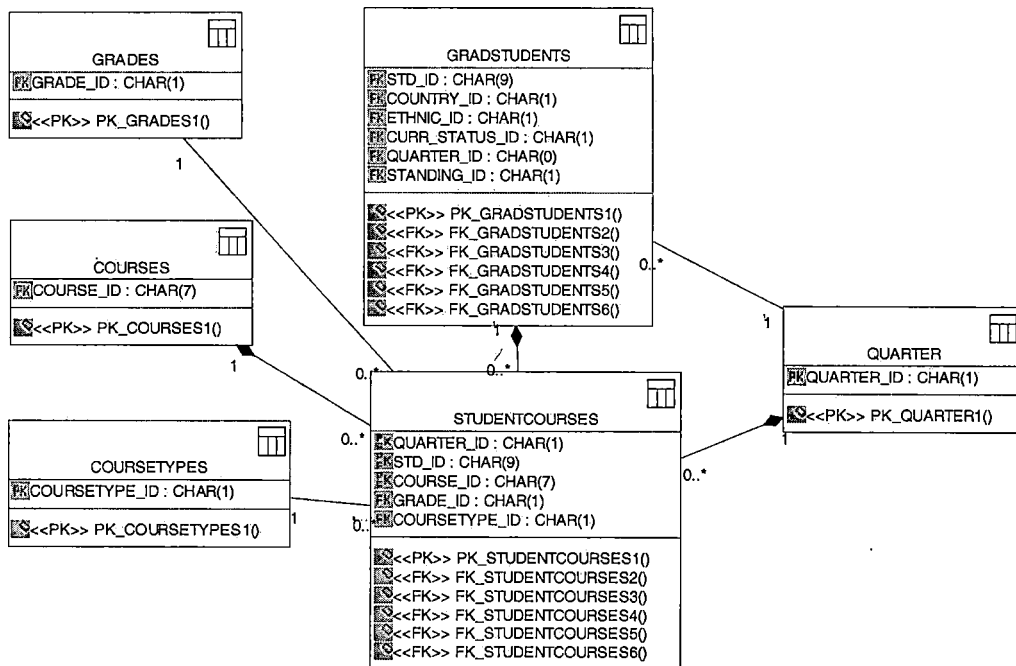


Figure 10. UML Model for Courses and Grading

Design. GRADSTUDENTS, STUDENTCOURSES, COURSES, COURSETYPES and GRADES QUARTER tables are used for implementation.

GRADSTUDNETS table has a L-M-N relationship with COURSES table and QUARTER table. This relationship is implemented by STUDENTCOURSES table, in which each record represents one course taken by a student in a quarter. Each Student has as many records as the courses he or she takes,

and a student can take the same course in different quarters.

STUDENTCOURSES table has M - 1 relationship with COURSE table via a foreign key. One course of COURSE table can be in many student-courses of STUDENTCOURSES table.

STUDENTCOURSES table has M - 1 relationship with COURSETYPES table via a foreign key. One course type of COURSETYPES table can be in many student-courses of STUDENTCOURSES table.

3.3.2.6 Task Assignment. Task Assignment assists faculty and staff in managing tasks assigned to other users. Task assigners and task assignees can interactively communicate about a task's status via report / comments / evaluation. When a task is assigned to a group (See grouping module.), task assignment module will assign tasks to all group members. Admin users can add frequently assigned and complicated tasks, which may consist of ordered jobs, to a task book for later assignment. Task Assignment is also assisted by e-mail service in notifying task assignees of their tasks.

Functions. Assign/Remove/Edit tasks to Staff/Students: Staff users can assign task to student and other staff. They also can edit and remove the tasks, which already have been assigned.

1. Create/edit pre-defined tasks. Faculty and GCA can standardize regulations, frequently used and complicated tasks by registering pre-defined tasks with the system. A pre-defined task may be a single or multiple requirement tasks, which consists of many jobs that must be completed in order. Users just select the pre-defined task to assign, and the assignees will received all the task details.

2. Create/Edit pre-defined jobs. As mentioned in multiple requirement pre-defined tasks, staff users can define jobs to constitute pre-defined tasks.

3. Follow-up tasks Assigned to others. Staff users can follow up and monitor the progress of tasks assigned to others. Staff users can send additional advice and view a feedback or comments from an assignee.

4. Users view / report result & progress of tasks assign to themselves. Task assignees can view details of the tasks assigned to them, send feed back and report progress and result to task assignor.

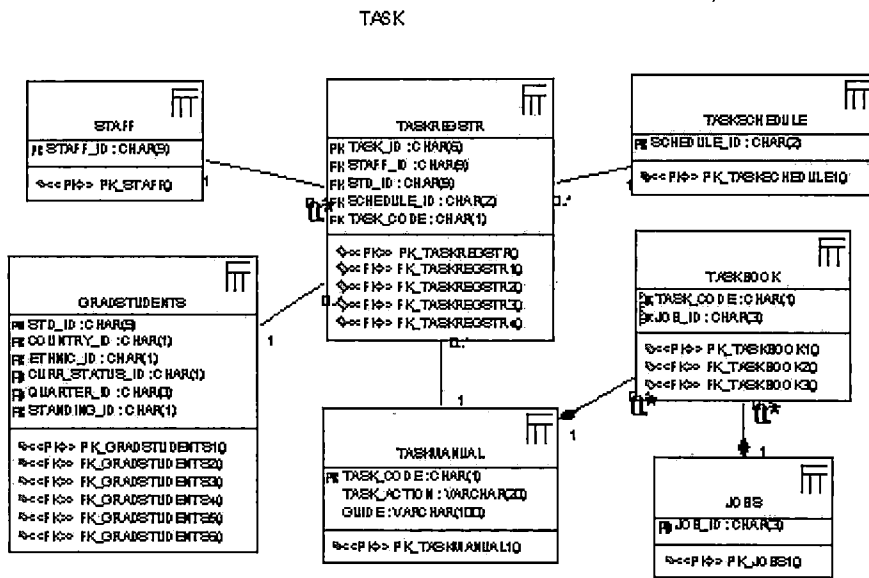


Figure 11. UML Model for Task

Design. GRADSTUDENTS, TASKREGSTR, STAFF, TASKBOOK, TASKMNUAL Apache Software Foundation and JOBS tables are described in detail below.

GRADSTUDNETS table has M - N relationship with STAFF, and STAFF table has M - N relationship with itself. This relationship is implemented by TASKREGSTR table, in which each record represents a task assigned by a staff member to a student or to another staff member. One staff member can assign tasks to many students of GRADSTUDENTS table or many staff members of STAFF table, and one student or many staff members can assign one staff member task.

TASKMANUAL table has M -N relationship with JOBS table. This relationship is implemented by TASKBOOK table. One task in TASKMANUAL table can consist of many jobs, and one pre-defined job in JOBS table can be used by many tasks of TASKMANUAL table.

3.3.2.7 Resident System Tasks and Scheduled System Tasks. Resident System Tasks automatically execute when changes in the database satisfy pre-defined conditions. These tasks are important tools to monitor database activities, e.g., monitor GPA of students.

Scheduled System Tasks, on the other hand, execute at scheduled frequencies. These tasks will automate pre-defined tasks at desired periods.

Two kinds of system tasks will facilitate system automation. Furthermore, an authorized Admin User can have the options to load and enable/disable the tasks at any time.

Functions. Resident System Tasks and Scheduled System Tasks have the following capabilities:

1. GPA Monitor
2. Contact Update Monitor
3. Academic Standing Monitoring
4. Load / Unload / Rescheduling scheduled system tasks

Design. System tasks design including design of resident System Tasks, which stay resident in memory, and scheduled system tasks which run at scheduled times.

1. Resident System Tasks. This task is implemented as Oracle triggers. These triggers will automatically execute when an update of certain fields of the database tables meet pre-defined conditions.

- GPA monitor. This trigger will automatically compute total GPA when there is an update in a student's grades. If the computed GPA falls below a threshold (3.0), this trigger will automatically generate a task alert for the MS Program Assistant and Graduate Coordinator that the student should be put under probation status. A letter of probation is also generated. This trigger is implemented in two steps to avoid mutation conflict that puts the Oracle database in a deadlock. The mutation conflict happens when Oracle attempts to update many student courses and at the same time compute the GPA. Two processes compete with each other causing a deadlock.

```

Trigger GPA_Alert_Step1 {
If Studentcourses table has update then
copy student ID to a update_temp table
End if
}
Trigger GPA_Alert_Step2 {
If update_temp table has an update then
Compute GPA
If GPA < 3 then
Generate letter of probation
Endif
Remove the update from update_temp table
}

```

Figure 12. Pseudo_Codes for GPA Monitor

- Contact Update Monitor. When a student updates his or her own contact information (current address, phone, email) via the web-based interface, this trigger will notify the PA via a task assignment and by an e-mail about the update.

```

Trigger Contact_Update_Monitor {
If update email, phone, local_address, home_address then
Add "update" email, phone, local_address, home_address to alert_message
Alert Program assistant by a task
Endif
}

```

Figure 13. Pseudo_Codes to Monitor Contact Update

- Academic Standing Monitor. When a student's academic standing is changed to classified status or advanced to candidacy, this trigger will generate the letter of classified status or advanced to candidacy status, save the letter into DOCUMENTS table, DOCUMENTS folder, and at the same time notify the MS Program Assistant and the GCA.

```
Trigger Academic_Standing_Monitor {
If Academic status changes from Conditional to classified then
    Generate letter of classification
Put the letter into DOCUMENTS table
    Alert Program assistant by a task to view and print out the letter
Else if Academic status changes from Conditional to advanced to candidacy then
    Generate letter of classification
    Generate letter of being advanced to candidacy
    Put the letter into DOCUMENTS table
    Alert Program assistant by a task to view and print out 2 letters
Else if Academic status changes from classified to advanced to candidacy then
    Generate letter of being advanced to candidacy
    Put the letter into DOCUMENTS table
    Alert Program assistant by a task to view and print out the letter
Endif
}
```

Figure 14. Pseudo_Codes to Monitor Academic Standing

2. Scheduled System Tasks. These tasks are implemented as an Oracle scheduled procedure. The task is developed as an Oracle procedure and submitted to the Oracle scheduler to execute at a pre-defined time and frequency. Following

is the automatic task renewal scheduled to execute once a day.

- Automatic Task Renewal. When a task is assigned with a frequency i.e. a weekly report and after the task is completed, it needs a mechanism that automatically renews the task for the next period. With the Oracle Scheduler, an authorized user can schedule any procedure that is designated for scheduling.

```
Procedure Task Renewal: {
Begin
  Make a collection of tasks that satisfy conditions: Have frequency and already due
  or due today plus frequency period not later than expired dates when the tasks
  do not need repetition.
  For each task of the collection {
    Generate a new task with a new due date = due date + frequency
    Notify the task assignee about new due date of the task
  }
End
```

Figure 15. Pseudo_Codes for Task Renewal

3.3.3 Security

Many security concerns are addressed in the project including encryption of transmission over the Internet, protection of application server, Oracle server and student sensitive information.

3.3.3.1 Encryption of Internet Transmission with SSL.

All JSP web pages are processed by the Oracle Application Server, which is powered by the Apache Server and Open SSL.

Communication between server and client sides are forced to transmit under SSL transmission protocol. Any attempts to access unsecured pages are redirected to secure pages. An Oracle temporary security certificate is used for a trial period.

3.3.3.2 User/Role and Password Enforced

Authentication. User authentication and authorization is implemented by user ID / password. When a user logs in, the system will verify the user's user ID and password against USER table. After a user is authenticated, the system will authorize the appropriate role for the user. The user then have access to all functions authorized for his or her role.

Due to the dynamic roles and functions feature, a function can be optionally assigned and removed from any role. The execution privilege of the function for a certain role, therefore, cannot be hard-coded but dynamically handled by a new mechanism at the function level. In the new mechanism, each function has a unique privilege token stored in PRIVILEGE_CODE field of FUNCTION table. When successfully logged in, the user will load (into PRIVILEGE container) all the privilege tokens of the functions, which are assigned to the user's role. The PRIVILEGE container of

vector data structure is maintained in a session object for speed and is available throughout the user's session.

When a user attempts to execute a page, a verifying component of the page will check if the user's PRIVILEGE container has the privilege token required for the page. If the user does not have a proper privilege token, the verifying logic will deny access and redirect the request to a login page.

Social Security Number is no longer used as an index and identification key, and its content is only accessed by Admin roles.

3.3.3.3 Controlled Input Information. All input information from a student user is restricted by fixed-length format and screened for possible malicious codes. All special characters that can lead to attack, are either prohibited or converted to presentation characters, i.e. '&' is converted to '&', '<' to '>'; ... This will reduce the risk of cross-site scripting attacks, and overflow buffer-based attacks where hackers seed their malicious codes with the above special characters in input fields, or cause an overflow buffer input with malicious codes.

3.3.3.4 Hackcheck Before Execution Of External Commands. Each external command is subject to verification before execution to avoid malicious codes that may cause an

attack. An EXECFOP Java function calls FOP external command to generate PDF documents by passing a full command line string. "C:/FOP/FOP <argument>". Before the external command string executes, it is verified that the command string must not have unexpected ' | 'or ';' characters, which can lead to an attack.

3.3.3.5 Encrypted SQL Procedures. All SQL procedures are encrypted with "wrap" Oracle command. When an unauthorized user gets access to the Oracle database he or she cannot view the procedures' contents.

3.3.3.6 Restricted Access to Oracle Database. The Oracle database is located in a private network (See Figure 16: A Recommended Deployment.), which is isolated from Internet accessible network. There are only two ways to access the Oracle server: (1) using Secure Shell(SSH) at port 22 and (2) TCP/IP protocol on port 1521 from the application server only. It means that to access the Oracle server, a user must gain access to the Application server first, and only the web component on the Application server is allowed to fetch data from Oracle server.

CHAPTER FOUR

DEPLOYMENT

4.1 System Requirements

The Oracle database server is located in a private network. The Oracle Application server is located after a firewall. Window Advanced Server 2000 or Linux Advanced Server 2.1 can be used. However, Linux 2.1 is recommended for its secure and performance features. Following is a recommendation for system deployment:

1. Firewall
2. Oracle Application server 9i AS 9.0.3 / Qmail server, minimum 256MB
3. Oracle database 9i - 9.0.2.0.1, minimum 512MB

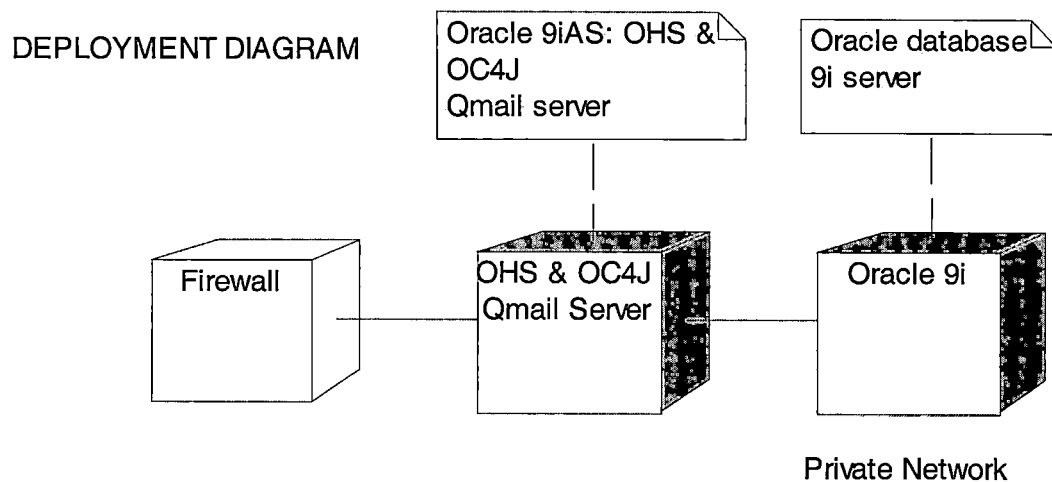


Figure 16. Recommended Deployment

4.2 Installation

4.2.1 Installation of Servers

For security reason, Oracle Corporation recommends Application server 9iAS, Oracle server 9i are installed with normal user account. Oracle server is configured to have an automatic startup when the server boots up, and shutdown properly before the server is powered off to avoid lost data. Detailed configuration for security is discussed in the Security Section.

Apache Formatting Object processor (FOP) is downloaded from Apache Software Foundation [1], and installed in the same machine with the Oracle Application server.

4.2.2 Installation of Web Component

The web component is packaged in grad. zip. Unzip the file in the D:\9iasgrad\j2ee\home\default-web-app folder. D:\9iasgrad is a folder in the Oracle Application server.

4.2.3 Migration from the Existing Database System

Migration from the existing database system, which runs Microsoft Access includes the creation of a new database schema, population of look-up tables and a migration of student data. Following is a flow chart showing the migration steps.

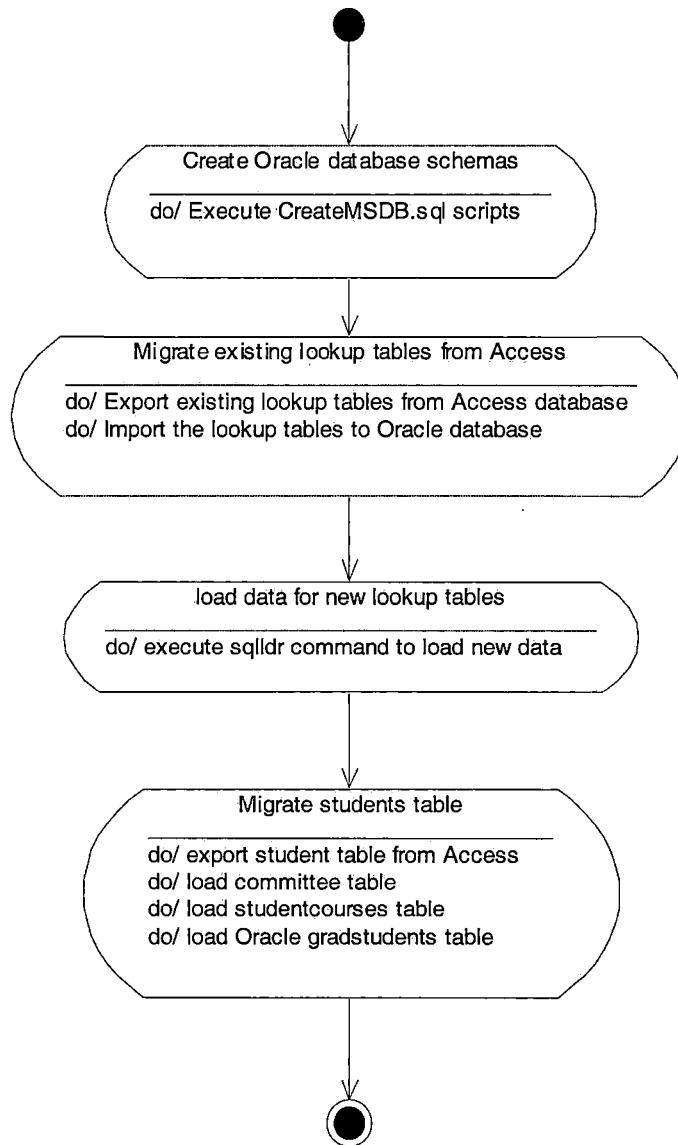


Figure 17. A Migration Flowchart

4.2.3.1 Migration of Look-Up tables. The database schema of look-up tables in the existing Access database is equivalent to that in the Oracle database, therefore the tables are migrated using Oracle SQLLoad utility. Data for additional lookup tables are created by a text editor.

4.2.3.2 Migration of the Student Table. In the existing Access database, student information is stored in a single master student table while in the new Oracle database schema design, student information is stored in several relational tables: GRADSTUDENTS, STUDENTCOURSES, COMMITTEE. (See Figure 7: UML Data Mode for Graduate Students.) The migration of the student table is implemented as follows:

1. Export data to text file with an Access query. Each record presents a course, which was taken by a student. Each student will have as many records as number of courses the student has taken.

2. Load the text file into STUDENTCOURSES table.

3. Export data to text file with an Access query. Each record presents a faculty member, who resides in a student's committee. Each advanced-to-candidacy student will have as many records as number of faculty, who reside in the student's committee.

4. Load the text file into COMMITTEE table.

5. Export the existing student table into a text file, and load it into Oracle GRADSTUDENTS table.

6. Run SQL script to convert quarter fields from text to number (FA to 1, WI to 2, SP 3, SU to 4).

CHAPTER FIVE

CONCLUSION AND FUTURE DIRECTIONS

5.1 Technology Highlights and Conclusion

The Web-Based Database System for the Computer Science Graduate Study Program implements advantages of data structure (such as vector), SQL language, Java Sever pages, Java Bean, relational database model, and XML technology for a production application. Furthermore, several measures are implemented to secure access to the database system.

Using the relational database model with Oracle software, the graduate information is modeled and constructed from basic and simple tables using binary and ternary relationships. As a result, the database system can flexibly cope with new requirements without changes and workaround in programming. The system functionality is designed with framework architecture, in which new functions can be added by just registering the new functions into the system and assigning them to appropriate roles.

Authentication and authorization is handled in a more robust and dynamic way with the assistance of the database. Execution privilege of a function is no longer restricted to any fixed roles, and no longer hard-coded, but loaded at

run time into a vector data structure and submitted to authorization verification.

Using SQL trigger and scheduled procedures of Oracle 9i software, the database system can monitor update, and automatically handles some designated events, such as GPA monitoring and status update monitoring. The efficiency of the database system is significantly enhanced by automated processing.

The application of XML technology in storing and reconstructing SQL query logic helps facilitate the grouping concept which is used in many other functions such as searching, task assignment, and other future functions such as reporting and announcement. XML technology also assists in automatically generating PDF documents when Oracle triggers are activated by appropriate events.

Several security measures have been implemented such as protection of social security number, transmission encryption with SSL, password-enforced authentication and authorization, encryption of SQL procedures, security verification of external command. When the Oracle database is located in a private network and behind a firewall, any attempts to breach the security is not impossible but really very hard.

The Web-Based Database System for the Computer Science Graduate Study Program has been designed and developed from experiences with the existing database system and with the goal to use the new system in the department and suggest it be used or adopted by other graduate programs in the University.

5.2 Extensions

Due to the scope of my master project, the following feasible features are not yet implemented but are worthwhile to be added into the database system. To implement these features, some additional table might need to be added into the database schema.

Appointment Scheduler. System will schedule appointments for students to see faculty and program coordinator for advising.

Announcement. System will assist faculty and GCA and PA in posting their announcements to faculty, students and other special interest groups.

Reporting. System will report students with any selection criteria using grouping concept. With pre-defined XML style sheet transformation (XSLT) and query result in XML, user can produce a list in formats of choice when

plugging into the report with a designated XSLT style sheet.

Centralized management of documents. Beside managing system-generated documents, system will upload all other documents of graduate students into a centralized database and made them accessible online for faculty, PA and GCA. Beside a convenient access for the admin users, the centralized document management system also becomes an electronic secondary archive of student documents.

Extended Learning management. This is an efficient feature of the existing management system that manages registration in CSCI698 of graduate students who have already registered for a project or a thesis course but have not completed the project or thesis. It keeps track and grant students permission to register for this special course.

APPENDIX
LIST OF PROGRAM FILES

Following is list of over 160 JSP, HTML, Java Script and Cascade style sheet (CSS) files developed for the Database Management System. Naming convention is based on their functionality. They are arranged within their functions and in alphabetic order.

1. Authentication & Authorization.

bye.jsp

checkSecurePage.jsp

checkUserPrivileges.jsp

getUserPrivileges.jsp

gradLogin.jsp

NoCache.jsp

Top.htm

2. Assign Tasks.

taskManagement.jsp

taskManagementBot.jsp

taskManagementGuide.htm

taskManagementHeader.htm

taskManagementSendMail.jsp

taskManagementStaff.jsp

taskManagementStaffBot.jsp

taskManagementStaffDone.jsp

taskManagementStaffDoneBot.jsp

taskManagementStaffTop.htm

taskManagementStudent.jsp
taskManagementStudentDone.jsp
taskManagementStudentDoneBot.jsp
taskManagementStudentGroups.jsp
taskManagementSystem.jsp
taskManagementTop.jsp
taskMangementStaffBot.jsp
taskMangementStaffTop.htm
taskStaff.jsp
taskStudent.jsp
taskSystem.jsp

3. My Tasks.

myTask.jsp
myTaskDetail.jsp
myTaskStaff.jsp
myTaskStaffDetail.jsp
myTaskStudent.jsp
myTaskStudentDetail.jsp

4. Follow-up Task Assigned to Others.

taskFollowup.jsp
taskFollowupBody.jsp
taskFollowupBot.jsp
taskFollowupDetailBody.jsp
taskFollowupStaff.jsp

TaskFollowupStaffDetail.jsp
taskfollowupStaffDone.jsp
taskFollowupStaffDoneTop.jsp
taskFollowupStaffSystemTop.jsp
taskFollowupStudent.jsp
TaskFollowupStudentDetail.jsp
taskfollowupStudentDone.jsp
taskFollowupStudentDoneTop.jsp
taskFollowupSystem.jsp
taskFollowupSystemDone.jsp
taskFollowupSystemDoneTop.jsp
TaskFollowUpSystemScheduledTask.jsp
taskFollowupTop.jsp

5. Change password.

changeIDPassword.jsp
changeIDPasswordCancel.htm
changeIDPasswordDone.htm
changeIDPasswordError.htm
changeIDPasswordVerify.jsp
createInitialUserPassword.jsp
initializeUserPassword.jsp

6. View and Create Group.

groupDefinition.jsp
grouping.jsp

groupingBot.jsp
groupingModifyBot.jsp
groupingNewBot.jsp
groupingNewBotDone.htm
groupingSelectBot.jsp
groupingSelectTop.jsp
groupingTop.jsp
groupingViewBot.jsp
groupingViewBotAppend.jsp
groupingViewBotNew.jsp
groupVerify.jsp

7. Customize home page.

custFirstPage.jsp
custFirstPageAbort.htm
custFirstPageDone.htm

8. Roles and Functions.

roleManagement.jsp
roleManagementChangeRole.jsp
roleManagementFunction.jsp
roleManagementFunctionEdit.jsp
roleManagementNewRole.jsp
roleManagementRole.jsp
changeStaffRole.jsp
designNewFunctions.jsp

designNewRoles.jsp

9. View Student information.

viewbot.htm

viewdetail.jsp

viewDetailAdmission.jsp

viewDetailCandidacy.jsp

viewDetailCourses.jsp

viewDetailGRE.jsp

viewDetailNotes.jsp

viewdetailPersonal.jsp

viewDetailPrerequisite.jsp

viewGradCommitteeAdvisees.jsp

viewGraduatedAdvisees.jsp

viewPersonalInfo.jsp

viewStudentInfo.jsp

viewStudentList.jsp

viewTaskDetail.jsp

viewtop.htm

10. View own advisees.

viewAdvisees.jsp

viewAdviseesBody.jsp

viewAdviseesBot.jsp

viewAdviseesCommitteeBody.jsp

viewAdviseesCommitteeGraduated.jsp

viewAdviseesCurrentBody.jsp
viewAdviseesGraduated.jsp
viewAdviseesGraduatedBody.jsp
viewAdviseesTop.jsp

11. View Any advisees.

viewAnyAdvisees.jsp
viewAnyAdviseesBot.jsp
viewAnyAdviseesCurrentBody.jsp
viewAnyAdviseesGraduatedBody.jsp
viewAnyAdviseesTop.jsp

12. View Personal Information.

ViewPersonalAdmission.jsp
ViewPersonalCourses.jsp
ViewPersonalDetail.jsp
ViewPersonalGRE.jsp
ViewPersonalInfo.jsp
ViewPersonalMenu.htm
ViewPersonalPrerequisite.jsp

13. Update Contact Information.

updateContactInfo.jsp
updateContactInfoAbort.htm

14. Update Student Information.

updateAllCourses.jsp
updateCommittee.jsp

updateCommitteeProcess.jsp
updateCoresProcess.jsp
updateElectives.jsp
updateElectivesProcess.jsp
updatePersonalInfo.jsp
updatePersonalInfoForm.jsp
updatePersonalInfoProcess.jsp
updatePrerequisiteProcess.jsp
updatePrerequisites.jsp
updateStudentAddProcess.jsp
updateStudentInfo.jsp
updateStudentInfoList.jsp
updateStudentInfoProcess.jsp
updateSummary.jsp
updateSummaryProcess.jsp
seachStudent.jsp

15. Create Predefined Tasks.

createPredefinedTasks.jsp
preDefinedTask.jsp
preDefinedTaskJob.jsp
preDefinedTaskJobEdit.jsp
preDefinedTaskNewTask.jsp

16. Add new Student.

updateStudentAdd.jsp

17. Document Generation.

viewStudentDocProcess.jsp

ViewStudentDocs.jsp

18. Connectivity.

gradConnectBean.jsp

ConnectionError.jsp

19. Shared pages.

index.html

main.jsp

mainLeft.jsp

mainRight.jsp

publicFunction.jsp

returnFirstPage.jsp

sendOracleError.jsp

calendar.js

portals.css

REFERENCES

- [1] Apache Software Foundation *Apache XML project*,
<http://xml.apache.org/fop/>
- [2] Benoit Marchal: *XML by Example*. QUE Indianapolis,
Indiana, 2000 (ISBN 0-7879-2242-9)
- [3] CERT /CC Vulnerability Notes Database - CERT
Coordination Center- CarnegieMello Software
Engineering Institute <http://www.kb.cert.org/vuls>
- [4] David M. Geary. *Advanced Java Server Pages*. Prentice
Hall - Sun Microsystem Press 2001 (ISBN 0-13-030704-
1).
- [5] Duan K. Fields & Mark A. Kolb. *Web Development with
Java Server Pages*. Manning Publication Co., Greenwich,
CT 06830, 2000
- [6] Elliontte Rusty Harold. *XML Bible - Gold Edition*.
Hungry Minds. New York, NY. 2001 (I SBN 0-7645-4819-0)
- [7] Elmasri & Navathe. *Fundamental of Database System -
Third Edition*. Addison Wesley 2000, (ISBN 0-8053-1755-
4)
- [8] HTML 4.01 Specification. [http://www.w3.org/TR/REC-
html40/](http://www.w3.org/TR/REC-html40/)
- [9] Martin Fowler, *UML Distilled Second Edition - Brief
Guide to the Standard Object Modeling Language*. The
Addision-Wesley, Massachusetts, 1999 (ISBN 0-2016-
5783-sX).
- [10] Marty Hall. *More Servlets and Java server Pages*.
Prentice Hall - Sun Microsystem Press 2002. (ISBN 0-
13-067614-4).
- [11] Oracle - *Oracle 9i Database Utilities Release 2 (9.2)*
[http://download-
west.oracle.com/docs/cd/B10501_01/server.920/a96652/to
c.htm](http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96652/toc.htm)

- [12] Oracle - *Oracle 9i XML Database Developer's Guide - Oracle XML DB. Release 2 (9.2)* http://download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96620/toc.htm
- [13] Oracle - *PL/SQL User's guide and Reference Release 2 (9.2)*. http://download-west.oracle.com/docs/cd/B10501_01/java.920/a96655/toc.htm
- [14] Oracle - *Oracle 9i Application Server Release 2 (9.0.3) Release notes - Aug 12, 2003*
- [15] Oracle - *Oracle 9i Application Server. Best Practice - An Oracle White Paper - June 2002*
- [16] Oracle Metalink <http://metalink.oracle.com/>
- [17] Scripts in HTML documents - W3C Recommendation <http://www.w3.org/TR/REC-html40/interact/scripts.html>
- [18] Security Focus <http://www.securityfocus.com/>
- [19] David Litchfield. *Hack proofing Oracle Application Server*. NDSSoftware Insight Security Research - 2002. www.ngssoftware.com
- [20] Brett McLaughlin. *Java and XML*. O'Reilly. 2000.
- [21] D. J. Bernstein. *Qmail: Qmail security guarantee*. <http://cr.yp.to/qmail.html>
- [22] Dave Sill. *Life with qmail*. <http://www.lifewithqmail.org/lwq.html>