

**A WEB DATABASE SYSTEM TO MANAGE  
GRADUATE STUDENT INFORMATION**

**BY**

**JING YANG**

**Bachelor of Management Engineering**

**Beijing Institute of Machinery Industry**

**Beijing, P. R. China**

**1993**

**Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December 2004**

A WEB DATABASE SYSTEM TO MANAGE  
GRADUATE STUDENT INFORMATION

Thesis approved:

*M. Samadzadeh H.*

Thesis Advisor

*J. Chandler*

*Paul K. Hill*

*A. J. S. S.*

Dean of the Graduate College

## PREFACE

The use of Web pages has grown beyond simple information display to collection and presentation of data. The volume and structure of data presented on a Web page warrant storing and organizing the data into a database and then generating Web pages based on this database. Database management systems (DBMS) such as MySQL, MS Access, Oracle, and SAS provide the software tools needed to organize large amounts of data in a flexible manner. The thesis intended to develop a customized database application to store and extract graduate student information, and build a data driven Web site to provide users with the ability to view the database using a web browser. This thesis report includes the process of evaluation and selection of an appropriate database management system and the corresponding server-side Web language (e.g., ASP and PHP).

This thesis reports on the preparatory work done to select the MS Access DBMS from among three database information management systems. The ASP server-side scripting language was chosen to develop a dynamic and interactive database-driven Web site for the thesis work. This thesis report also contains the details of the design and implementation of a database management system that was used to obtain more detailed and specific requirements from the potential end-users of the system. A Graduate Student Web Database system was proposed and successfully implemented. Web visitors can secure login into this database-driven Web site and view the student information extracted from the database.

## ACKNOWLEDGEMENTS

There are many people I would like to thank who helped me with this thesis. First and foremost I would like to acknowledge the continued support and guidance from my thesis advisor, Dr. Mansur H. Samadzadeh. He provided deep insight and technical advice on all aspects of this thesis work.

I am greatly thankful to Dr. John P. Chandler and Dr. Nohpill Park for taking time out of their busy schedules and providing valuable feedback while serving as members of my graduate committee.

Special thanks go to Mr. Terry Wright, our Systems Manager. He has been a tremendous help in many areas.

I especially thank Mr. & Mrs. Duncan and Mr. & Mrs. Johnson. They have given me a big help in my English studies.

I am grateful to my husband Jian Chang for his patience and suggestions. A special thank-you must go to my son Matthew. He gives me many joyous moments in my life.



## TABLE OF CONTENTS

Chapter	Page
I INTRODUCTION .....	1
II BACKGROUND .....	3
III DESIGNING AND BUILDING A DATABASE STRUCTURE .....	5
3.1 Planning Database Creation .....	5
3.1.1 Purpose of a Database .....	5
3.1.2 A DBMS Used to Manage Student Information .....	7
3.1.3 Comparison of Current DBMS .....	8
3.1.3.1 Availability .....	8
3.1.3.2 Ease of Use for End-Users .....	9
3.1.3.3 Ease of Use for Database Managers .....	11
3.1.3.4 Security .....	13
3.1.4 Solution .....	15
3.2 Database Design and Implementation .....	16
3.2.1 Categorization of the Applicants .....	16
3.2.1.1 Determining the Fields .....	17
3.2.1.2 Determining the Tables .....	19
3.2.1.3 Determining Relationships Among Tables .....	22
3.2.2 Data Entry and Creating Other Database Objects .....	24
IV WEB DATABASE DISPLAY .....	28
4.1 Creating ASP Files .....	29
4.2 Adding Code in Order to Read the Database .....	30
4.3 Publishing the ASP Files .....	31
4.4 Running the Page and Reviewing the Result .....	32

Chapter	Page
V USER GUIDE .....	38
5.1 Database System .....	38
5.1.1 Database Tables .....	38
5.1.2 Database Queries .....	47
5.1.3 Database Forms .....	52
5.2 Web Database Display .....	56
5.2.1 Hosting .....	56
5.2.2 ASP Files .....	56
5.2.3 Create or Edit ASP File .....	60
VI SUMMARY AND FUTURE WORK .....	63
6.1 Summary .....	63
6.2 Future Work .....	63
REFERENCES .....	65
APPENDICES .....	68
APPENDIX A: GLOSSARY .....	69
APPENDIX B: TRADEMARK INFORMATION .....	71
APPENDIX C: WEB VISITOR MANUAL .....	72
APPENDIX D: WEB DATABASE MANAGER MANUAL .....	75
APPENDIX E: SYSTEM ADMINISTRATOR MANUAL .....	78
APPENDIX F: CSDB TABLES AND QUERIES LIST .....	82
APPENDIX F.1: TABLES .....	82
APPENDIX F.2: QUERIES .....	83
APPENDIX G: CSDB ASP AND IMAGE FILES LIST .....	86
APPENDIX G.1: ASP FILES .....	86
APPENDIX G.2: IMAGE FILES IN THE IMAGES FOLDER .....	90
APPENDIX H: CSDB ASP CODES .....	92
APPENDIX H.1: HEADER.ASP CODE .....	92
APPENDIX H.2: SEARCH.ASP CODE .....	94

## LIST OF FIGURES

Figure	Page
1. MySQL Statements on a Unix System .....	10
2. MySQL Display Overflow on a Windows System .....	10
3. Categorization of the Applicants .....	17
4. Academic History Table .....	21
5. Thesis and Dissertation Table .....	22
6. CS Graduate Student Database Relationships .....	23
7. CS Graduate Student Information Form .....	24
8. “Applied Students for Each Year” Query Using QBE .....	26
9. SQL Code for “Applied Students for Each Year” Query .....	26
10. ASP Retrieves Data to Produce Web Pages .....	29
11. Login Page .....	32
12. Enrolled Student Name List Page .....	33
13. Student Information View Page .....	34
14. Statistics Page .....	35
15. Applied Students for Each Year Page .....	36
16. Search Page .....	37
17. CSGraduateStudent Tables .....	39
18. Datasheet View of Personal Information Table .....	40

Figure	Page
19. Design View of Personal Information Table .....	42
20. Validation Rule on the Degree Field of the Enrollment Table .....	44
21. Create a New Table in Design View .....	46
22. Create New Relationships Among Tables .....	47
23. CSGraduateStudent Queries .....	48
24. Design View of Enrolled M.S. Students Name Query .....	49
25. Enrolled Students for Each Year_Crosstab Query .....	51
26. Courses Form .....	54
27. Header.asp Page .....	59
28. Search_Results.asp Page .....	59

## LIST OF TABLES

Table	Page
I. Some Database Design Questions and Possible Responses .....	6
II. Ease of Use for Database Managers .....	12
III. Input Mask Characters .....	43
IV. Query Criteria .....	50
V. Code Structure of CS Graduate Student ASP Files .....	61

## CHAPTER I

### INTRODUCTION

A database is an organized collection of data. Database management systems (DBMS) [Shelly et al. 2001] such as MySQL, Microsoft Access (MS Access), Oracle, and Statistical Analysis System (SAS) provide the software tools needed to organize large amounts of data in a flexible manner. A DBMS includes facilities to add, modify, or delete data from a database, ask questions about the data in a database, and produce reports summarizing selected contents of a database. DBMSs can be divided into two categories: desktop databases and server databases. Desktop databases (e.g., MS Access) are oriented toward single-user applications and typically reside in standard personal computers. Server databases (e.g., SQL Server or Oracle) contain mechanisms to ensure the reliability and consistency of data, and are geared toward multi-user applications. Server databases are designed to run on high-performance servers and carry a correspondingly higher price tag.

This thesis work involved building a database application for the Computer Science Department graduate students. The design included the capability to generate a number of statistical results that can be obtained from the database through a user-friendly interface. The goal was to provide the ability to conveniently store and extract information about graduate students for advising, career counseling, and generating summary reports. The resulting Web database system is designed to require a login procedure to access the

database. The built-in search capabilities can be used to produce textual and statistical reports about graduate students. One of the system's design goals was to provide the users with the ability to view and search the information using a Web browser.

The rest of this Thesis is organized as follows. Chapter II provides a general background for database systems. A number of different types of databases are discussed with a brief history of database processing. Chapter III contains a comparison of three database systems, a discussion of the components of a database system, and an overview of the process of building a database system. Chapter IV introduces Active Server Pages and describes the use of a Web browser to access and view database information. Chapter V is the user guide for the system which offers advice on how to edit and use the database system and the ASP pages. Several examples are used to introduce creation, design, implementation, and proper use. Chapter VI covers the summary of the thesis and some possible areas of future work.

## CHAPTER II

### BACKGROUND

Databases have been in use since the earliest days of electronic computing, but the building of a database has traditionally been a complex and expensive process. This situation has changed since the dramatic decline in the cost of computers and the explosive growth of the Internet and the Web.

It is now relatively easy to produce a database, but this does not guarantee that the database will be useful. A house is only as good as its foundation. This principle is equally true in the world of databases. Creating a database is analogous to laying down the foundation and frame of a house.

There is a large number of database products on the market today. In computing, databases are sometimes classified according to their organizational approach. The dominant model in use today is the relational model, a tabular database in which data is defined so that it can be reorganized and accessed in a number of different ways [Sarin 2000]. The relational model is used in the Structured Query Language, or SQL data manipulation/analysis tool [Kroenke 1997]. SQL is a standard language for making interactive queries and updating a database such as MySQL, the MS Access system, and database products from Oracle. There are also the hierarchical models and the network models, and data manipulation/analysis tools such as SAS and Focus.



Three database systems (namely, MySQL, MS Access, and SAS) were considered and evaluated comparatively for use in the construction of the graduate student database system. They are introduced below.

MySQL [Axmark and Widenius 1999] is an open source software (OSS) database management system offered by MySQL AB under its "dual license" business model. It supports the standard Unix interface and more than 20 other interfaces. It is a relational DBMS and is very tightly integrated with Perl and PHP [Atkinson 2000]. The first MySQL version was released in 1996. MySQL is essentially an SQL server - it responds to requests for information that are written in SQL. MySQL is a small, compact database server ideal for small applications.

MS Access [Getz et al. 2001] is a relational DBMS that runs under the Windows operating system. The first version of Access came out in 1992. As a component of the Microsoft Office Suite, MS Access has the same look and feel as other Office products such as Word and Excel. This database is relatively easy to use. Access stores an entire database application in a single file. An Access .mdb file can contain data objects such as tables, indexes, and queries as well as application objects such as forms, reports, and Visual Basic code.

SAS [Delwiche and Slaughter 2002] stands for Statistical Analysis System. SAS software is developed and distributed by the SAS Institute Inc. The first product was released in 1976. It is a statistical information system that performs simple or complex statistical analysis, predictive and descriptive modeling, calculations, forecasting, and simulation design. It is "an integrated system of software providing complete control over data access, management, analysis, and presentation" [Schlotzhauer and Littell 1997].

## CHAPTER III

### DESIGNING AND BUILDING A DATABASE STRUCTURE

#### 3.1 Planning Database Creation

Good databases are well planned and they are designed to suit their purposes. The first step in designing a database is to determine its purpose and how it is to be used. From that, we can determine what facts we need to store in the database and to what subject each fact belongs. These facts correspond to the fields (columns) in the database, and the subjects that those facts belong to, correspond to the tables. Without a good database design, there may be irrelevant data that will not be used, omitted data, inappropriate representation of entities, and lack of integration between various parts of the database.

##### 3.1.1 Purpose of a Database

Creating a robust, high-performance database requires careful thought and planning on every aspect of the DBMS. Database designers should plan carefully and ask questions about the information needed and the services desired. A database cannot be designed well if the right questions are not asked. Table I presents some of the questions that a database designer should ask and some possible responses. The questions in this table are based on the work of Sarin [Sarin 2000].

Questions	Possible Responses
What do you need the database designer to do for this project?	<ol style="list-style-type: none"> <li>1. Allow an assortment of contents to be saved in a secure dynamic database.</li> <li>2. Provide the ability to extract all types of saved contents in the database to produce reports and statistical results about graduate students.</li> </ol>
How is this database going to be used?	It is going to be used as a data warehouse online for looking up graduate student information and generating summary reports.
How large is the database? How will the database change over time?	It is going to store more than 100 fields for each student. It should have capacity to keep more than 1000 students' records.
What database system will be used for this project? What primary programming language will the database designer use?	Compare several commonly available database systems and select a database system and the associated programming language to create a new database system.
How many simultaneous users will the database need to accommodate?	A fixed number (5-10) or a variable number.
How many standard reports will be needed to be generated from this database?	One per week, one per day, on demand, etc.
What factors are most likely to become performance bottlenecks?	Performance related initialization parameters or frequent updates.
What security is needed? Who should be able to redefine the schema - new attributes, objects, object classes? Who should be able to edit and update?	The database manager is responsible for redefining schema, editing and updating database.
What are the operational requirements for the database? Will it need to be available 24 hours a day, 7 days a week?	The operational configuration archive log mode should be available 24x7 or on weekdays only.
What is the primary operating system that will be used in this database project?	Unix, Linux, Windows XP/NT/2000, etc.

Table I. Some Database Design Questions and Possible Responses

With the possible responses to the questions in mind, several DBMSs were compared, a database system was selected to store CS graduate students' data, and a dynamic database-driven Web site was created for the CS Department.

### 3.1.2 A DBMS Used to Manage Student Information

There are many different types of DBMSs in existence and they are suitable for different requirements. The objective of this thesis work was to design and build a CS Graduate Student database and a database-driven Web site. So, the focus was on the DBMSs used in universities. An effort was made to evaluate some DBMSs to discover whether a sophisticated multi-user server platform (such as Oracle) or a desktop database (such as MS Access) would meet our needs.

Our university's Student Information System, the University of Oklahoma, and Texas A&M use the Oracle DBMS since they need a big and complicated system to store and manage all the student and staff information. These universities need Oracle's professional technicians for installation and support. This DBMS and its technical support are expensive. This multi-user server platform is large and more expensive than we desired.

For the purpose of this thesis, the objective was some small-scale DBMS that can provide a relatively inexpensive and powerful database solution. A brief list of some desktop databases that are currently used at OSU follows. The Library, Department of Agricultural Economics, College of Business Administration, College of Human Environmental Sciences, and OSU-Tulsa use MS Access to generate and keep track of customized reports. The Information Technology Division uses SAS to store and analyze

both student and staff information. There are also some science or technology departments that use SAS for data analysis. The Computer Science Department uses MySQL to manage the student email system.

### 3.1.3 Comparison of Current DBMS

Given a specified purpose, database designers will need to choose a DBMS and a primary programming language to design the database. For the thesis work, three database information management systems were examined and evaluated for the database requirement. These three, which were available on the Computer Science Department's computer systems, were MySQL, MS Access, and SAS.

The four aspects of comparison considered were availability, ease of use for end-users, ease of use for database managers, and security. These aspects are discussed in the following four subsections.

#### 3.1.3.1 Availability

A brief description of these three systems and their operating system, and platform requirements is listed below.

MySQL is supplied by MySQL AB that provides it at no cost under the free software GNU General Public License (GPL) [MySQL 2004]. It is available for free download from the MySQL AB Web site. MySQL compiles on a number of platforms and has multithreading capabilities on Unix/Linux servers. MySQL can also run as a service on Windows NT and as a normal process in Windows machines. It can be said that MySQL is cross-platform [Axmark and Widenius 1999], which means that the database can be

developed under Windows and also on a Unix/Linux platform. The application programming interface (API) for MySQL is the MySQL language and SQL.

MS Access is one of the MS Office products that run on Windows 2000/XP/NT. It is not an open source database and cannot be executed directly under the Unix/Linux system. To connect to a MS Access database from a Unix/Linux platform, a driver with the Server component is needed to run on the server side. The API for MS Access is SQL and Query By Example (QBE).

The SAS System is an integrated suite of software for information delivery and analysis [Delwiche and Slaughter 2002]. Applications of the SAS System include executive information systems, data management, statistical and mathematical analysis, and application development. SAS is very powerful and is particularly useful for dealing with large data sets. SAS runs on Windows 2000/XP/NT and Unix/Linux, and it is possible to transfer a SAS data table from a Unix environment to a PC environment [Spector 2001]. The API for SAS is the SAS language and SQL.

#### 3.1.3.2 Ease of Use for End-Users

When the end-users view the reports and statistical results about graduate students from this database, the database source of the information is of no interest to them. They need an easy to use interface, fast responses, and correct results from the database. A database that can be searched and sorted according to the rules for each specific requirement is very important for anyone developing database products.

MySQL's commands are analogous to the Unix commands. The user should type the commands line by line to enter the requirements and to get results. Figure 1 is an

example of MySQL statements running on a Unix system. The MySQL command line interface allows you to put a statement on one line or spread it across multiple lines. If there are too many fields in a table, the display of the table will overflow which does not give a good view (Figure 2). MySQL AB offers several levels of technical support, from email access to complete 24/7 telephone support.

```
mysql> select * from table01;
+-----+-----+-----+-----+-----+
|  ID  | Order | Index          | Date      | Time      |
+-----+-----+-----+-----+-----+
|    1 | first  | new info      | 2004-10-22 | 15:29:01 |
|    2 | second | another       | 2004-10-23 | 15:29:01 |
|    3 | third one | more foo for you | 2004-10-24 | 15:29:01 |
+-----+-----+-----+-----+-----+

mysql> delete from table01 where ID=3;
Query OK, 1 row affected (0.01 sec)

mysql>
```

Figure 1. MySQL Statements on a Unix System

```
mysql> select * from table01;
+-----+-----+-----+-----+-----+-----+-----+
| field01 | field02 | field03 | field04 | field05 | field06 | field07 | field08 | field09 |
+-----+-----+-----+-----+-----+-----+-----+
|    1 | first  | 1999-10-22 | 06:22:18 | NULL    | NULL    | NULL    | NULL    | NULL    |
NULL | NULL |
|    2 | second | 1999-10-23 | 10:30:00 | NULL    | NULL    | Another | NULL    | NULL    |
NULL | NULL |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Figure 2. MySQL Display Overflow on a Windows System

Using MS Access, users can do dynamic backups, either incremental or complete, of the database while it is in use [Kroenke 1997]. This database does not force the users to exit the database to back up the data. Access provides an integrated application environment, which enables users to perform query and analysis of enterprise data sources from the interface. The users use the mouse to view the data back and forth, and easily sort the results by clicking buttons. MS Access has an easy to use interface.

The SAS system supports many different devices and modes of editing, displaying, running, and reporting of the SAS jobs. The SAS interface enables users to run a display manager, the program editor, log, and output screens [Delwiche and Slaughter 2002]. It also provides high quality, bit-mapped graphics output. SAS enables users to view the updated information dynamically.

### 3.1.3.3 Ease of Use for Database Managers

Table II shows some of the features that make a database easy to use for a database manager. Some of the features mentioned in the table are explained below.

As far as design view is concerned, MySQL is a function-call API for data access and management. The database manager would write the associated command to build the database and extract the data from the database library. MS Access has templates to help database managers get started creating the database. Database managers can create pages, forms, and reports using toolbars, toolbox, themes, and other features that are similar to the tools they use to create Word and Excel files. SAS uses the SAS language to perform statistical analyses. To use SAS effectively, database manager must have some knowledge of SAS software, statistics, or both [Olson et al. 2003].



The second feature is the capability to analyze data and to make projections. Using MS Access or SAS, database managers can organize data in different ways, make projections, do complex calculations using a spreadsheet control, and view data graphically in a chart. They are easy to use and their graphic user interfaces can do a lot more in graphical analyses than the mainframe or UNIX versions can. MS Access offers the ability to print a visual diagram of the relationships window, which makes it easier for users to see how the database is structured.

Feature/Capability	MS Access	SAS	MySQL
Design View	MS Access	SAS system	Unix system
Analyze Data	Yes	Yes	No
Web Display Text	HTML, ASP	HTML	HTML, PHP
Reduce Data Duplication	Can be reduced by good design	No	No
View Data Graphically in a Chart	Yes	Yes	No
Output can be cut, pasted, dragged, and dropped to other applications	Yes	Yes	No
Sort and Filter Records	Yes	Sort only	No
Help Menu	Yes	Yes	No

Table II. Ease of Use for Database Managers

Considering the third feature, which is displaying HTML text, MySQL, MS Access, and SAS can store HTML code in fields of a database and display it as formatted HTML text on a Web page. Unfortunately, HTML is a static language. We cannot directly make

database calls from HTML. To access the database from a Web page, we need an intermediary language such as ASP (Active Server Pages), JSP (Java Server Pages), or PHP (PHP Hypertext Preprocessor) [Meloni 2000]. ASP, JSP, and PHP are server-side scripting languages with embedding code within an HTML page, and are processed by the Web server. To develop a dynamic and interactive database-driven Web site, the best match for MySQL is PHP and the best match for of MS Access is ASP.

The reduction of data duplication is a desired feature of a DBMS. A well-designed MS Access database can reduce data duplication. If existing tables contain redundant data, the database manager can use the Table Analyzer Wizard to split the tables into related tables to store data more efficiently.

Having the capability to cut, past, drag, and drop outputs to other applications is useful and convenient. The field list of MS Access enables users to easily add information to a data access page view simply by dragging-and-dropping the field names from an easily accessible list. When a user renames a field in a table, the change is automatically going through the dependent objects such as queries and forms so the user can continue to work with the application. Using SAS, database managers can edit and manipulate files in a number of different formats such as MS Word or PowerPoint.

#### 3.1.3.4 Security

For database administrators, security ranks as a top concern. The security system is typically applied at two levels: server connection verification and database request verification. When a user tries to access a database, that user must first have access privileges to the database server. Server and database access must be blocked to

unauthorized users. Authorized users can also potentially damage data accidentally or maliciously.

MySQL database uses security based on Access Control Lists (ACLs) [Widenius et al. 2002] for all connections and queries or other operations that a user may attempt to perform. MySQL has a privilege and password system to verify that the user has connection rights to the server before the user can request to use a database that exists on the server. MySQL and PHP work together to build a database-driven Web site. PHP goes with a Web server with Apache's track record means security remains a top priority.

MS Access has a security wizard that may secure any database that has been created while joined to the System. It has been claimed that "the MS SQL Server can integrate with the Windows NT operating system security to provide a single logon to the network and the database" [MS 2001]. MS Access database on a server is better protected since unauthorized users cannot get to the database file directly due to the fact that they must access the server first. MS Access and ASP work together to build a database-driven Web site. ASP runs on IIS (Internet Information Services) [Spector 2001]. IIS has a number of security access-control mechanisms that can be applied to help make for safe surfing. Unfortunately, IIS has a long history of vulnerabilities. Whether these weaknesses are because of inherent problems or because IIS is a red flag to hackers, is irrelevant. Those systems have a history of being hacked and compromised.

SAS can support different security requirements. SAS login definitions on the enterprise directory ensure that only authorized portal users can obtain access to the SAS data and processes [Delwiche and Slaughter 2002]. Each SAS server has a login definition that specifies which users or groups of users can access the server.

### 3.1.4 Solution

MySQL cannot fully analyze data and it is in general not easier to design than an MS Access interface or a SAS interface. MS Access and SAS are easier to use because they have design interfaces that users can use to construct the necessary tables and reports. Occasionally, it may not be necessary for users to write programs, which reduces the overall compile and debug time. MySQL cannot build a table relationship chart and draw other charts [Axmark and Widenius 1999]. Users must write specific MySQL programs to design data queries.

SAS needs a specialized language to perform statistical analysis and generate reports. A database manager must learn the specialized knowledge of SAS software and be knowledgeable about statistics to be able to use SAS effectively. MS Access provides users with a programmer-friendly API that facilitates the rapid development of database-oriented custom applications. When using MS Access, the regular users of MS products will notice the familiar Windows “look and feel” as well as the tight integration with other MS Office family of products.

Overall, MS Access has the following advantages. The design tools and the API generally facilitate the rapid development of database applications. Specialized knowledge or training is not required. Users are generally able to build a dynamic database application, i.e., if a record is changed in the database, the change automatically appears throughout the database. The disadvantage of MS Access is that the security is not better than using MySQL which runs on Unix systems. In the future, the IIS security model will no doubt continue to mature, hopefully allowing even finer levels of access control to allow for greater ease, flexibility, and security in the generation of active content for the Web.

Based on the comparative evaluation mentioned above, this thesis work focused on the construction of a dynamic database application using MS Access.

### 3.2 Database Design and Implementation

In this section, a MS Access DBMS named CSGraduateStudent.mdb, which was designed and implemented as part of this thesis, is discussed.

#### 3.2.1 Categorization of the Applicants

The files of the graduate applicants to the CS Department constitute the data of this database. The students who satisfy the admission requirements are accepted, otherwise they are rejected. Most of the candidates, who meet the requirements and are admitted, enroll within the required time period, there are also some candidates who postpone or abandon their admission. Students are considered active if they enroll in a minimal number of credit hours each semester. A student who interrupts his/her enrollment for one year or does not earn enough credit hours is considered inactive. Some of the active students graduate and others continue to work toward their degrees. The overall categorization of the graduate applicants is shown in Figure 3.

The categorization of the applicants helps the database designer to design the fields and tables, determine which table each field belongs to, and build relationships among the tables. The following three subsections address these three issues respectively.

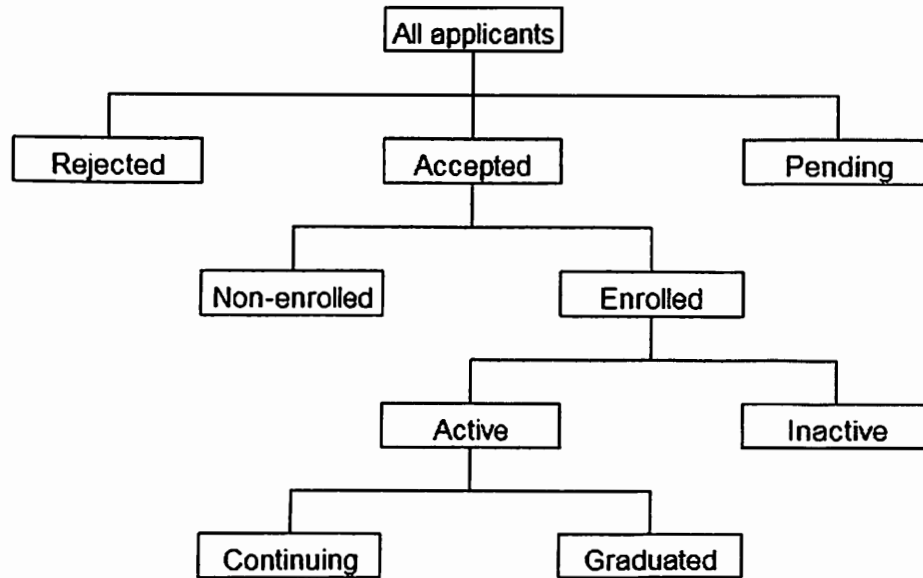


Figure 3. Categorization of the Applicants

### 3.2.1.1 Determining the Fields

Each field is a fact about a particular subject. For example, database managers might need to store background information about the students, e.g., major, GRE score, colleges attended, address, and GPA. A separate field should be created for each of these items. The following are some examples of fields for all graduate applicants.

- Personal Information
  - Last Name
  - First Name
  - Middle Name -- optional
  - OSUID -- OSU identification number
  - Gender
  - Date of Birth
  - Country of Birth
  - Visa Type --for international students
- Address
  - Street Address

- City
- State
- Zip code
- Phone -- optional
- Fax -- optional
- Email Address
  
- Academic History
  - TOEFL Score -- two kinds of TOEFL
  - GRE/Verbal -- score and percentile
  - GRE/Quantitative -- score and percentile
  - GRE/Analytical -- score and percentile
  - GRE/Advanced -- the type of subject tests, score, and percentile
  - Colleges Attended -- full name of the institutions
  - Degrees Earned -- all degree of bachelor, master, or Ph.D.
  - Major
  - GPA
  
- Enrollment
  - Application Date
  - Requested Semester and Year
  - Degree Sought -- M.S., Ph.D., or Ed.D.
  - Status -- rejected, postponed, pending, graduated, enrolled, withdrawn
  - Rejection Reason -- for the rejected students
  - Semester and Year Enrolled -- for the enrolled students
  - Semester and Year Graduated -- for the graduated students
  - Miscellaneous

If the status of a student is enrolled or graduated, more fields should be included in the database as shown below.

- Students and Courses
  - CourseID -- prerequisite or core course
  - Grade -- prerequisite or core course grade and GPA
  
- Thesis and Dissertation
  - Advisor -- the principal advisor
  - Committee Members
  - Thesis or Dissertation Title -- also provide a link to the thesis





Information table.

In order for MS Access to connect the information stored in separate tables -- for example, to connect a student's personal information with the same student's enrollment information -- each table in the database must include a field (or a set of fields) that uniquely identifies each individual record in the respective table. Such a field (or set of fields) is called a primary key [Shelly et al. 2001]. Without a primary key, a duplicate record may exist in the database and this may cause the database to become unusable. The primary key is used to link the subordinate table records to the corresponding record in the master table. In most of CSGraduateStudent database design and implementation, the primary key of each table is the OSUID.

Some of the students continue their degree from M.S. to Ph.D. or Ed.D. in the Computer Science Department. These students keep the same OSUID but have different enrollment, thesis or dissertation, and course records. In such cases, the candidate key and foreign key are introduced. A candidate key is a combination of one or more fields whose value uniquely identifies a record in a table [Shelly et al. 2001]. When a key in one table is referenced in another table, the key in the second table is called a foreign key [Shelly et al. 2001]. In the Enrollment and Thesis and Dissertation tables, the "Degree" field is added as a candidate key, and in the Students and Courses table, the "Degree" and "CourseID" fields are added as candidate keys. That are no two records in these three tables can have the same key values. In the Courses table, the field "CourseID" is foreign key into the Students and Courses tables.

A number of design principles, which can be used to design tables after determining the required fields [Kroenke 1997], are listed below.

1. Add each field to one table only. When each piece of information is stored only once, it will be updated in one place. This is more efficient and it also eliminates the possibility of duplicate entries that may contain different information.
2. Define the primary key.
3. A table should not contain duplicate information, and information should not be duplicated among the tables.

Based on the fields listed earlier in this section, there are at least eight main tables that should be included in this database: Personal Information, Address, Academic History, University, Enrollment, Students and Courses, Thesis and Dissertation, and Assistantship.

For security considerations, a table named Users, which includes the authorized users' IDs and passwords, was included in this CSGraduateStudent database. For ease of sorting and calculation, other supplementary tables were added. For example, a table named Courses was created with three fields: CourseID, the description of the course, and the credit hours for each course, which works with Students and Courses table to calculate the students' GPA of core courses and the prerequisite courses. Figures 4 and 5 show some samples of these main tables from the implementation of the graduate student database.

	OSUID	TOEFL	GRE/V	GRE/Q	GRE/A	GRE Adv	GPA	
+	0000000000000001	573					3.43	Adams Sta
+	0000000000000002	563					2.32	South Dak
+	0000000000000003	617	470	600	600		3.57	Portland St
+	0000000000000004	607					2.19	Texas Wes
+	0000000000000005	627					2.25	University c
+	0000000000000006	603	430	730	570		3.45	Troy State
+	0000000000000007	550	430	780	760		2.28	New Mexic

Record: 6 of 400

Figure 4. Academic History Table

	OSUID	Degree	Advisor	committee 1	committee 2	committee
+	0000000000000001	PHD	Dr. Chandler			
+	0000000000000003	MS	Dr. Chandler			
+	0000000000000006	MS	Dr. Chandler			
+	0000000000000007	MS				
+	0000000000000008	MS	Dr. Chandler			
+	0000000000000009	MS	Dr. Chandler			
+	0000000000000010	MS				

Record: 10 of 325

Figure 5. Thesis and Dissertation Table

### 3.2.1.3 Determining Relationships Among Tables

A relationship is an association established between the common fields (columns) in two tables [Spector 2001]. After setting up the different tables for the subject in the database, the database manager needs to tell the DBMS (MS Access in this case) how to bring the related information together. The first step in this process is to define relationships among the tables. After that, queries, forms, and reports can be created to display information from several tables.

A relationship works by matching data in primary and foreign key fields (usually a field with the same name in both tables). As it was pointed out previously, most of the tables in the graduate student database contain an OSUID field which works as a connection among tables. Also, the candidate and foreign keys work with the primary key to define the relationships.

This design and implementation involves two types of relationships among the tables:

- One-to-one relationship

A one-to-one relationship is created if both of the related fields are primary keys or

have unique indexes. The relationships among Personal Information, Address, and Academic History are one-to-one relationships.

- One-to-many relationship

A one-to-many relationship is the most common type of relationship. It is created if only one of the related fields is a primary key or has a unique index. A record in Courses table can have many matching records in Students and Courses table, but a record in Students and Courses table has only one matching record in the Courses table. The relationship between Personal Information and Enrollment tables is another example of one-to-many relationships.

Figure 6 illustrates the relationships among the thirteen tables existing in the current implementation of the CSGraduateStudent database.

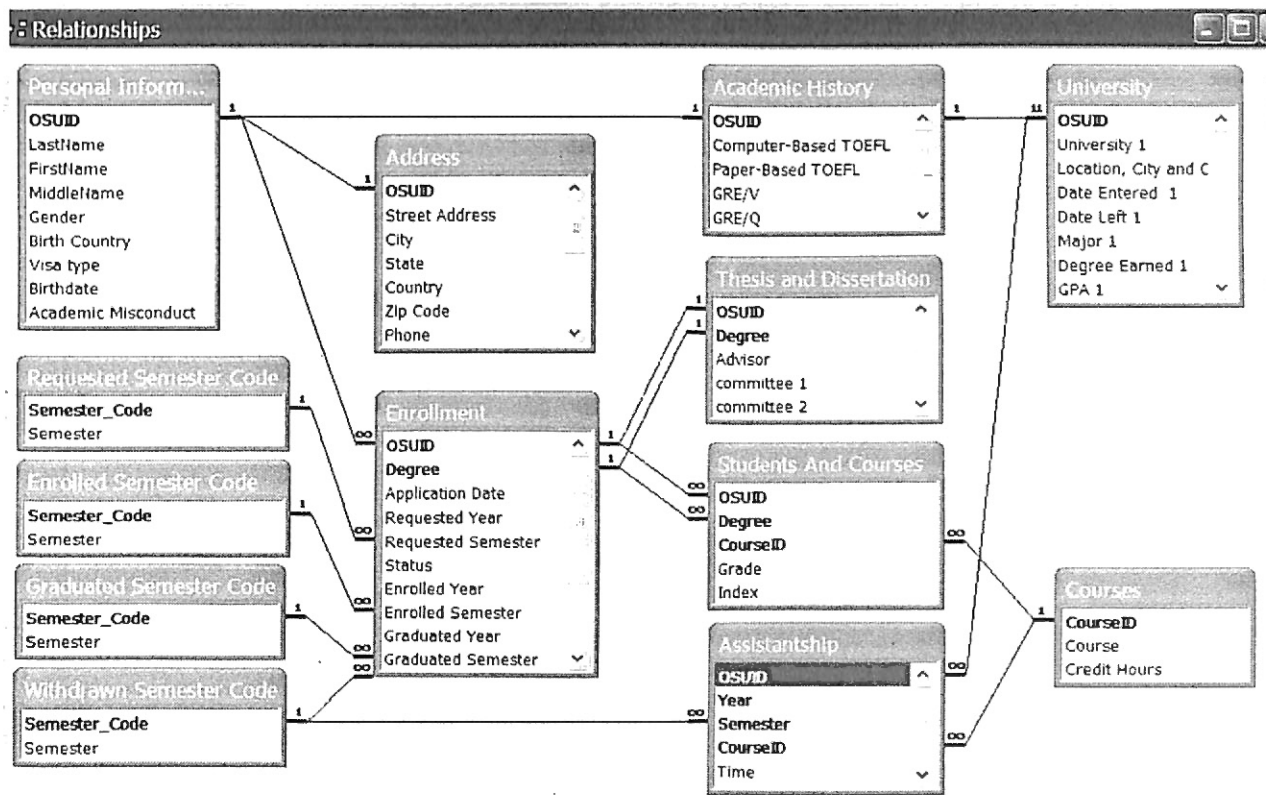


Figure 6. CS Graduate Student Database Relationships

### 3.2.2 Data Entry and Creating Other Database Objects

Once the table structures are finalized, it is time to add all the existing data to the database. Then the other database objects such as forms, queries, reports, and data access pages can be created. There are several ways to enter data. Data can be entered through tables or forms.

To enter the entire data for each student, the easy and convenient method is through the use of forms. A form named Computer Science Graduate Student Information was created in the CSGraduateStudent database to enter, display, and edit data in the fields. The resulting form is shown in Figure 7.

**Computer Science Graduate Student Information**

**Personal Information**

OSUID

Last Name

First Name

Middle Name

Gender

Birth Country

Academic Misconduct

Visa Type

Birthdate  (MM/DD/YYYY)

**Address**

Street Address

City  State

Country  Zip Code

Phone

Fax

E-mail Address

Record:     226 of 400

Figure 7. CS Graduate Student Information Form

Each page of this form shows the entire information for a student in the database. The database manager may enter and edit the data through the use of a selected OSUID. When the data in this form is changed at any time, the associated data fields in the related tables will be changed correspondingly.

Forms are also good at editing or updating data in the fields. Some fields in tables may be changed and these changes may involve several tables. For example, if a student's OSUID is changed, this change should go through 7 tables; if a student's Degree is changed, this change should go through 4 tables. The database manager should update all these data in the tables, otherwise errors will be generated or the database will no longer provide correct information.

After data are stored in tables, users want to query the data to answer questions or to identify problems or particular situations. Queries are questions about data stored in tables, or a request to perform an action on the stored data [Spector 2001].

A query can bring together data from multiple tables to serve as the source of data for a form, report, or data access page. Queries offer the ability to retrieve data, filter data, and calculate summaries. Well designed queries in a database are very important, they make the database more flexible, easier to use, and yield better performance. In this thesis work, more than 80 queries were created to suit the users' perceived requirements. The following list contains some queries created in this CSGraduateStudent database:

- Applied M.S./Ph.D. Students for Each Year/Semester
- Applicants Who Graduated from OSU
- Graduated M.S./Ph.D. Students for Each Year/Semester
- Pending Student Information
- Rejected Reason
- Rejected University
- Assistantship Query
- Core/ Prerequisite Courses

- Gender by Year/Semester
- TOEFL and GRE Scores for Each Year/Semester

There are a number of ways that a query can be expressed. A database manager may design queries by using Query By Example (QBE) [Kroenke 1997] design view or by writing SQL code. Figure 8 shows the creation of “Applied Students for Each Year” query using QBE, and Figure 9 shows that query using SQL code.

	Requested Year	MS	PHD	Total Applied Student
	1996	1		1
	1997	21	2	23
	1998	40	4	44
▶	1999	53	4	57
	2000	72	8	80
	2001	67	5	72
	2002	26	3	29
	2003	49	3	52
	2004	30	5	35
	2005	8	1	9

Record: [First] [Previous] 4 [Next] [Last] of 10

Figure 8. “Applied Students for Each Year” Query Using QBE

<b>TRANSFORM</b>	Sum([Applied Students for Each Year].[Applied Student])
	AS [SumOfApplied Student]
<b>SELECT</b>	[Applied Students for Each Year].[Requested Year],
	Sum([Applied Students for Each Year].[Applied Student])
	AS [Total Applied Student]
<b>FROM</b>	[Applied Students for Each Year]
<b>GROUP BY</b>	[Applied Students for Each Year].[Requested Year]
<b>PIVOT</b>	[Applied Students for Each Year].Degree;

Figure 9. SQL Code for “Applied Students for Each Year” Query

Before using this graduate student MS Access database to enter actual data, it is a good idea to refine the design. After the tables, table relationships, fields, and forms have been built, it is time to study the design and detect any flaws that might remain. To test the design of this CSGraduateStudent database, contrived data for 400 students were entered. All necessary queries and forms were created and tested to see if they show the answers expected. Data duplications were carefully examined and eliminated. It is obviously easier to change the database design while using contrived data than it will be after the tables have been filled with actual student data.



## CHAPTER IV

### WEB DATABASE DISPLAY

In addition to designing and building a database, this thesis work involved creating a dynamic database-driven Web site which allows the content of the site to reside in a database and to be dynamically pulled from the database to create Web pages for Web visitors to view with a regular Web browser.

ASP was used to create and run dynamic, interactive Web server applications. A Web page containing ASP cannot be run by just simply opening the page in a Web browser. The page must be requested through a Web server that supports ASP.

After a Web page is created, database managers can “publish” the page to Web folders or a Web server. Publishing refers to the process of exporting datasheets, forms, or reports to static HTML or server-generated HTML such as ASP pages, and setting up these files and all related files as a Web application on a Web server [MS-2 2003].

As shown in the Figure 10 [Yank 2003] below, the ASP scripting language processes the page request, fetches the data from the MS Access database, and generates it dynamically as the formatted HTML page that the browser expects.

Creating a Web database display using MS Access involves four steps [Buyens, 2000]: create ASP files, add code in order to read the database and add the corresponding HTML code, publish the ASP files and the database to the Web folders or a Web server,

and run the page and review the result. These steps are briefly discussed below.

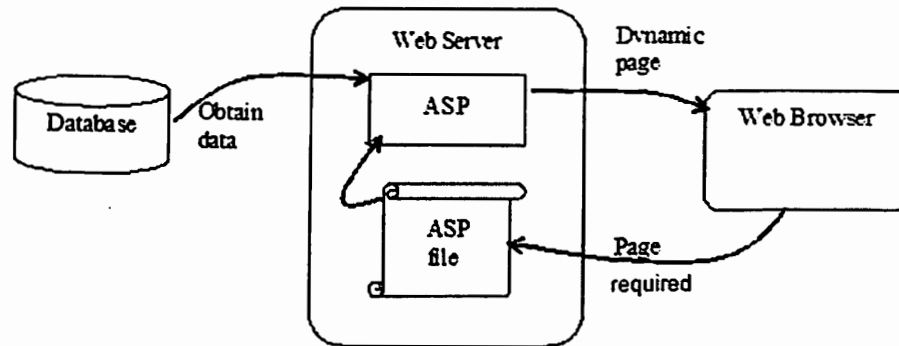


Figure 10. ASP Retrieves Data to Produce Web Pages [Yank 2003]

#### 4.1 Creating ASP Files

An ASP file is a text file with the extension .asp that contains any combination of the following: text, HTML tags, and server-side scripts. ASP pages are essentially HTML pages with embedded ASP code. So an ASP page contains HTML which is static as well as ASP tags which are processed on the server.

Since ASP pages have the extension .asp instead of .htm or .html, when a page with the extension .asp is requested by a browser, it is interpreted before sending the resulting HTML document to the browser [Corkhill 2001]. So the ASP is run on the Web server and not passed to the Web browser. If a file is saved with a .htm file name extension, the Web sever would not recognize or execute the respective VBScript code.

An ASP command needs to be placed between tags, `< %.....%>`, to indicate server-side script. The code extracts the roster from the database and presents it as HTML. The following example shows a simple HTML page that contains an ASP script command:

```
<html>
<body>
Hello, World!
<%
= now()
%>
</body>
</html>
```

The VBScript function now() returns the current date and time. When the Web server processes this page, it replaces <%= now() %> with the current date and time, and returns the page to the browser with the following result:

Hello, World! 9/18/2004 8:33:22 AM

A database manager needs to create and manage HTML and ASP Web page. MS FrontPage editor is a way to create and manage Web sites.

#### 4.2 Adding Code in Order to Read the Database

For security consideration, authorized Web visitors have to enter their user name and password to gain access to the site. All of the login details are stored in a CSGraduateStudent database table. The ASP files involved in the graduate student database include codes to check if the login information is matched in the database.

To read the database, we must first open it. Here is the code to connect to the Student Information query of the CSGraduateStudent database.

```
<%
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open xDb_Conn_Str
strsql = "SELECT * FROM [Student Information] WHERE
        OSUID = " & key & " AND Degree = " & s & " "
Set rs = Server.CreateObject("ADODB.Recordset")
rs.Open strsql, conn %>
```

In the graduate student database, similar database open, connect, and close commands are included in the ASP pages which involve extracting data from tables and queries.

#### 4.3 Publishing the ASP Files

In graduate student database implementation, more than 100 ASP files and more than 40 graphic files are used to create the database-driven Web site. The followings files are the ASP files dealing with the currently enrolled students' information that will be published to the Web site.

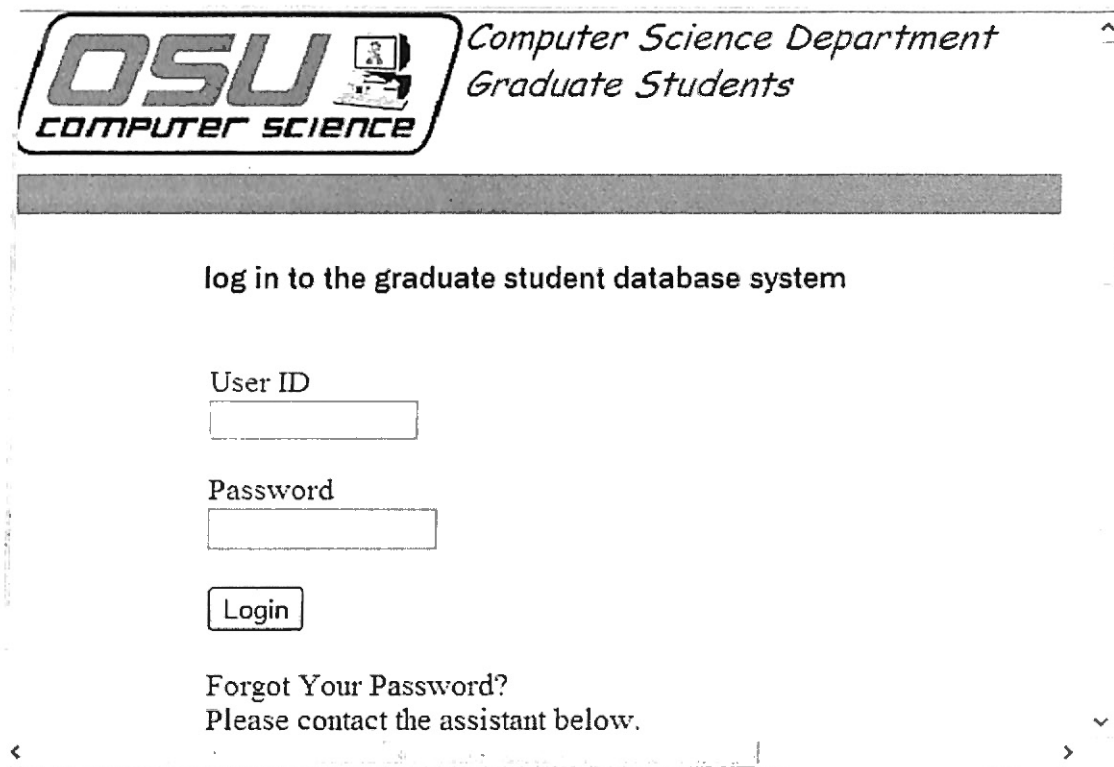
- Assistantship\_Querylist.asp
- Enrolled\_MS\_Graduated\_from\_OSUlist.asp
- Enrolled\_MS\_Students\_Namelist.asp
- Enrolled\_Students\_by\_Countrylist.asp
- Enrolled\_Students\_for\_Each\_Yearlist.asp
- Gender\_Enrolled\_by\_Yearlist.asp
- Student\_Information\_View.asp
- TOEFL\_and\_GRE\_Enrolled\_for\_Each\_Yearlist.asp

To see the database Web pages in action, all files should be copied to a properly configured location on the Web server. A folder should be created at this location to store all the associated files. Care must be taken to copy the CSGraduateStudent database, all ASP files, and all graphics files to this specified folder which is located on the Web server.

There are two requirements for proper configuration of publishing ASP files [Buyens 2000]. First, the Web server must have the software necessary (e.g., MS Personal Web server 4 or later) to process the VBscript code and the software necessary (e.g., IIS) to process the Access database. Second, the Web server folder where the ASP files and all other related files are placed must be flagged as executable.

#### 4.4 Running the Page and Reviewing the Result

The ASP script runs under Windows 2000/XP Web Server which supports IIS. Once a page is uploaded, a Web page designer can test the page by typing the Uniform Resource Locator (URL) into a browser. A display of the implementation of the CS Graduate Student Web Database System login page is shown in Figure 11.



**OSU**  
COMPUTER SCIENCE

*Computer Science Department  
Graduate Students*

---

**log in to the graduate student database system**

User ID

Password

[Forgot Your Password?](#)  
Please contact the assistant below.

Figure 11. Login Page

In the login page, the database manager creates user-level security by requiring user ID and password to allow Web visitors to access the database from the Web page. The user ID and password record set must match the user ID and password that Web visitors enter in the dialog boxes that appear when the database manager outputs the ASP files. Then, a Web visitor has entered the site, or an error message will appear through this page.

There are five main link directories in this Web page: Home page, Students page, Statistics page, FAQ page, and Search page. These Web pages are explained below.

In the Students page (Figure 12), there are eight hyperlinks that the Web visitor can follow: Enrolled students page, Graduated students page, Postponed students page, Pending Students page, Rejected students page, Withdrawn students page, All students page, and Assistant students page.

The screenshot shows the OSU Computer Science Department Graduate Students page. At the top left is the OSU Computer Science logo. To its right, the text reads "Computer Science Department" and "Graduate Students". Below the logo is a vertical navigation menu with buttons for Home, Students, Statistics, FAQ, Search, and Computer Science Department. To the right of the menu is a horizontal navigation bar with buttons for Enrolled, Graduated, Postponed, Rejected, Pending, Withdrawn, and Ass. The "Enrolled" button is selected. Below the navigation bar, the text "Student > Enrolled Students" is displayed. Underneath, it says "Enrolled MIS Students (Click to see PhD Students)". A table lists the names of enrolled MIS students with columns for Last Name, First Name, Middle Name, and a View link.

<u>Last Name</u>	<u>First Name</u>	<u>Middle Name</u>	
Poole	Ethan		<a href="#">View</a>
Eton	Olivia		<a href="#">View</a>
Strain	Kevin	H	<a href="#">View</a>
Thompson	William		<a href="#">View</a>
Garcia	David	R	<a href="#">View</a>
Lewis	Christopher	C	<a href="#">View</a>
Lee	Daniel	L	<a href="#">View</a>

Figure 12. Enrolled Student Name List Page

For each hyperlink, the page shows a name list of M.S or Ph.D. students. Web visitors may click a student First Name, Last Name, and Middle Name to sort records in ascending or descending order. For all ASP Web pages in the CS Graduate Student Web Database System, if there is an underline under a record name, this record may be sorted in

ascending or descending order by clicking on it. Web visitors may also click the Students Information View link to see the detailed information of the selected student.

The key that connects the Students page and Students Information View (Figure 13) is the student's OSUID number.

The screenshot shows the OSU Computer Science Department Graduate Students Information View page. At the top, there is a logo for OSU Computer Science and the text "Computer Science Department Graduate Students". Below this is a navigation bar with tabs for "Personal Information", "Address", "Enrollment", "Academic History", and "Courses at". On the left side, there is a vertical menu with buttons for "Home", "Students", "Statistics", "FAQ", "Search", and "Computer Science Department". The main content area is titled "Personal Information" and contains a table with the following data:

OSUID	0000000000000217
Last Name	Poole
First Name	Ethan
Gender	M
Country	China
Enrollment Status	F1
Birth Date	4/17/1976
Marital Status	N

Below the table, there is a section titled "Address -- Current" with a form field for the address.

Figure 13. Student Information View Page

The Students Information View page lists all CS Department students' information. To hyperlink different locations in the same page, bookmarks [MS-2 2003] were used to identify the locations within the files that Web visitors can later refer or link to. When Web visitors click on a student's View link, there are five bookmarks they may link to: Personal

Information, Address, Academic History, Enrollment information, and Courses and Thesis/Dissertation record.

The statistics page (Figure 14) provides statistical information extracted from the CSGraduateStudent database.

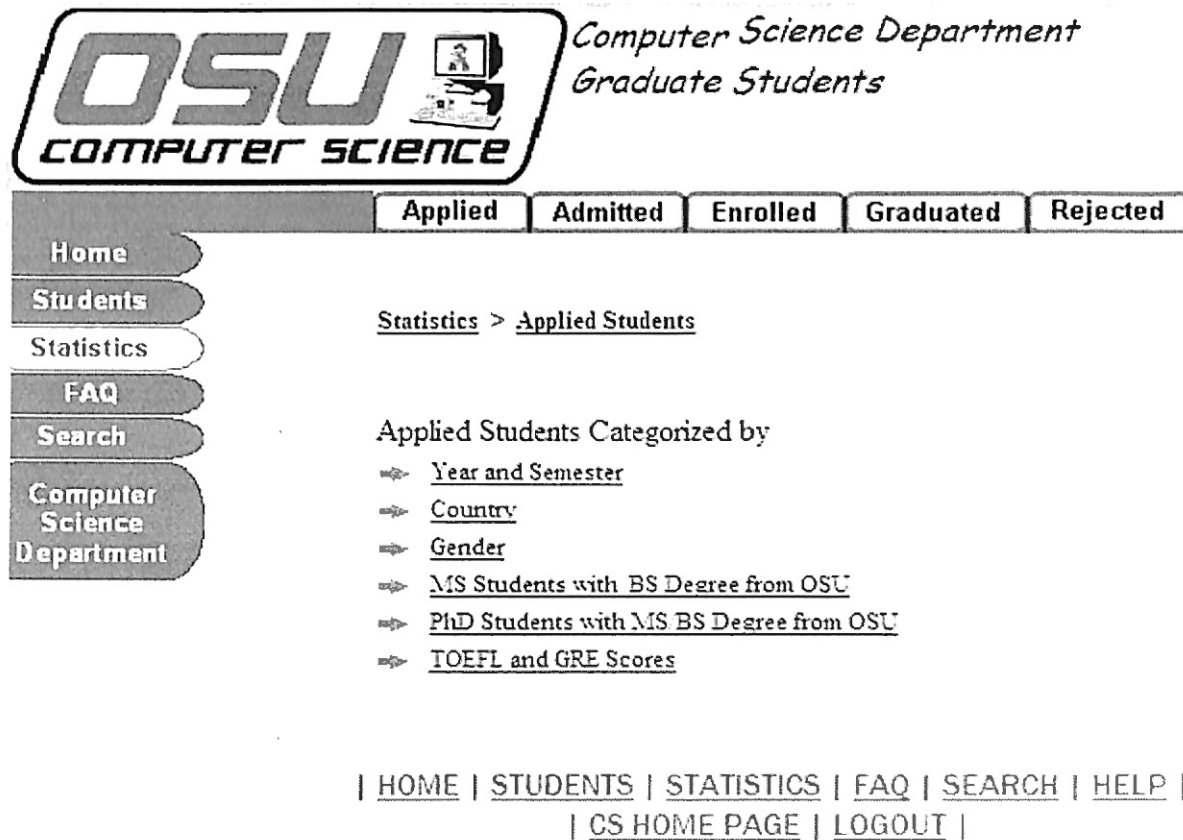


Figure 14. Statistics Page

In this page, there are six category hyperlinks that a Web visitor can follow: applied students statistics links page, enrolled students statistics links page, admitted students statistics links page, graduated students statistics links page, rejected students statistics links page, and over all students statistics links page.



Following any one of the hyperlinks, a Web visitor will get the statistics associated with the selected category. Some of the major topics of interest are the data grouped by year and semester, country, gender, and TOEFL/ GRE scores. In the current implementation, 65 statistics pages have been created for this thesis work. Most statistical data for these pages come from queries. Figure 15 is an example statistics page for “Applied Students”.

Applied Students for Each Year ( Click to view [Semester](#))


<u>Requested Year</u>	<u>Degree Program</u>		<u>Total</u>
	Master	PhD	
1996	1		1
1997	21	3	24
1998	40	4	44
1999	53	4	57
2000	72	8	80
2001	67	5	72
2002	26	3	29
2003	49	3	52
2004	30	4	34
2005	8	1	9
<b>TOTAL</b>	<b>367</b>	<b>35</b>	<b>402</b>

Figure 15. Applied Students for Each Year Page

The FAQ page addresses the questions that are regularly asked by Web visitors or the database manager of the database-driven Web site, statistics requirements, and problem reports.

The search page (Figure 16) offers a number of ways in which Web visitors can find the resources they are seeking. Web visitors can perform database tables or queries

search by entering a student's First and Last Name, OSUID Number, or a Key Word into the respective box. When the search is finished a list of matches will appear.



*Computer Science Department  
Graduate Students*

---

- Home
- Students
- Statistics
- FAQ
- Search
- Computer Science Department

Student Information

Student Name

Exact phrase    All words    Any word

OSUID Number

Exact phrase    All Number    Any Number

Key Word

Exact phrase    All words    Any word

Figure 16. Search Page

## CHAPTER V

### USER GUIDE

There are two components that make up CS Graduated Student Web database system. The first component is the MS Access database named CSGraduateStudent which includes tables, queries, and forms associated with the CS graduate student information. CSGraduateStudent database runs on a Windows sever machine. The second component consists of the CS Graduate Student Web pages that run on the Windows IIS server machine. The CS Graduate Student Web pages use the ASP scripts language to communicate with the CSGraduateStudent database.

#### 5.1 Database System

The CSGraduateStudent database will eventually contain all graduate student data. There are a total of 14 tables, 81 queries, and 3 forms in the current implementation of the database. These object numbers could be increased as needed. The following three subsections describe the tables, queries, and forms of the database.

##### 5.1.1 Database Tables

A table is a collection of data about a specific topic, such as Personal Information or Enrollment. Using a separate table for each topic means that we store the data only once.

This results in a more efficient database and fewer data-entry errors. Tables organize data into columns (called fields) and rows (called records).

### View an Existing Table

- ▶ Double click on the CSGraduateStudent.dbm file and a Database window appears. For security reasons, you should change the database's name when you upload your database and database-driven Web pages to the server. As a result, it will not be easy for others to try to track the system's database.
- ▶ In the Database window, click on Tables under Objects. A name list of all tables in the CSGraduateStudent database is displayed as Figure 17.

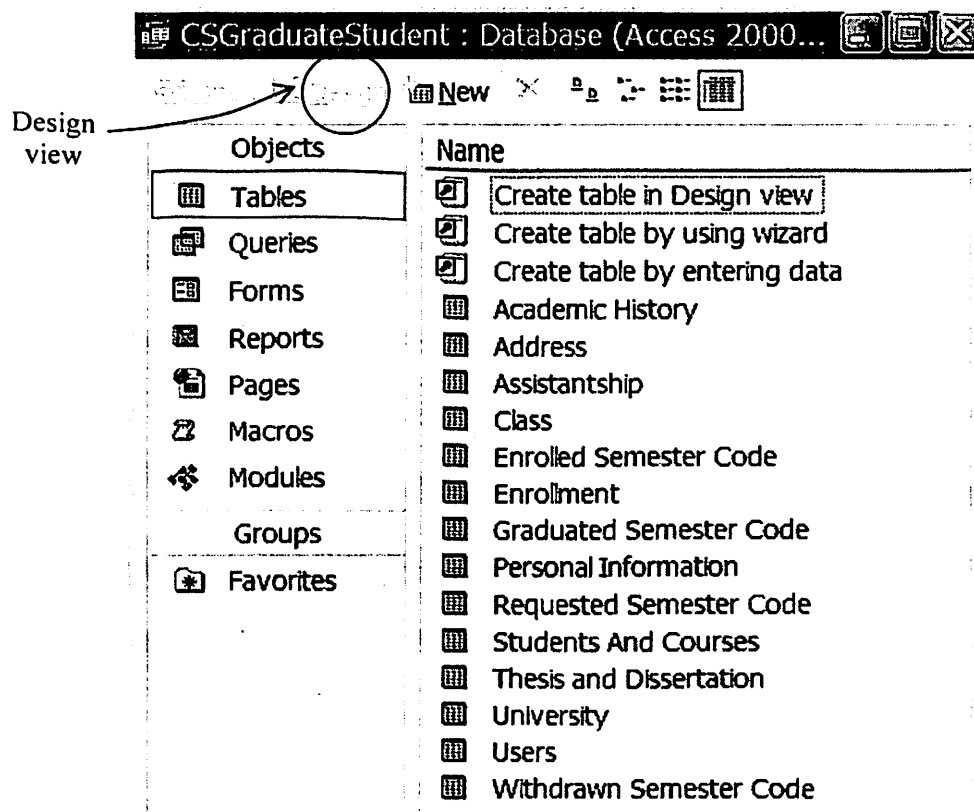


Figure 17. CSGraduateStudent Tables

- ▶ Click on the table you want to open. The default view window is Datasheet view (Figure 18) which displays data from a table, form, or query procedure in a row-and-column format. In this table datasheet, you can edit field names, add or delete data, and sort data in a field. You also can open the table in the Design view by clicking on Design (Figure 17) on the Database window toolbar. A detailed discussion of the Design view is covered in the field design section below.

OSUID	LastName	FirstName	MiddleName	Gender	Birth Country	Vis
0000000000000001	Garcia	Carol		F	US	
0000000000000002	Calder	Benjamin	T	M	US	
0000000000000003	Caldwell	Nathan		M	China	
0000000000000004	Chadwick	Logan	R	M	US	
0000000000000005	Chilton	Kevin	K	M	Bangladesh	
0000000000000006	Church	Gabriel		M	China	
0000000000000007	Clay	Robert		M	India	

Figure 18. Datasheet View of Personal Information Table

### Add New Records to an Existing Table

Open the CSGraduateStudent database and the table you want to edit. If the table is empty and you want to add a bunch of data at one time from other format out of this database, click file -> Get External Data -> Import, give the import data path, then click Import. If the table is not empty and you want to add a bunch of data from another format file, copy all the data you want to enter from the other format file, then go back to Datasheet view window of the table, click the “\*” at the end of the table, then paste. You can also simply enter a record directly in the associated field.

There are three things that should be emphasized when some new records are added

to an existing table.

- The primary key field is unique and not empty.
- The data type you entered should match the defined data type of each field.
- If the table is related to other tables, enter the data necessary to other related tables.

### **Add, Rename, or Delete a Field in the Existing Table**

#### **1. Add a Field**

A Field is a logical group of bytes in a record. The Design view of a table (Figure 19) is used to add some new fields. To create a field, you should carefully define the Field Name, Data Type, and Field Properties. The Description is optional to display the text of what the field is.

In Design view, you can insert a new field at any position you want. To insert a new column at the desired position, you can click anywhere on the column that will succeed it. On the main menu, click Insert -> Rows and type the field name. To add a new column at the end of the table, click the first empty field under Field Name and type the field name.

The Field Name column is used to type a name for each field. The Data Type column is used to choose the data type you want. The data type starts out as default Text, however there are many other data types that can be used. Each field can have only one data type. All data types used in the CSGraduateStudent database are explained as below.

#### **o Text Data Type**

This type is text or a combination of text and numbers, as well as numbers that do not require calculation, such as OSUID. Almost all of the data in the CSGraduateStudent database are this type.

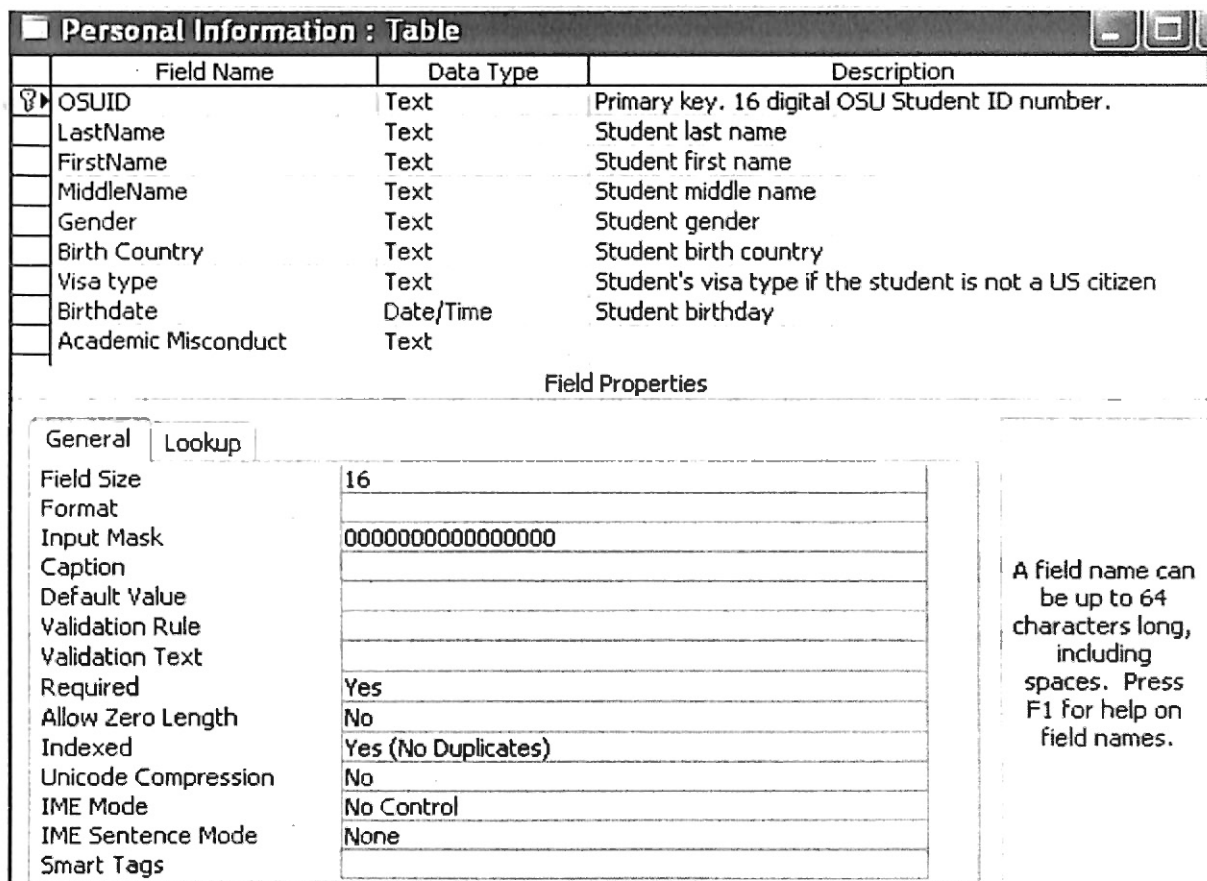


Figure 19. Design View of Personal Information Table

- o Number Data Type

A numeric data is used in mathematical calculations. You can select options, such as Integer and Double, from Field Size. TOEFL and GRE Score fields in Academic History table are examples of this kind of data type.

- o Date/Time Data Type

This type shows date and time values for the years 100 through 9999. There are a variety of formats that can be found under Format of Field Properties. Fields named Birthdate and Defense Date are of Date/Time data type using the mm/dd/yyyy format.

After a data type is selected, field properties should be determined. Field properties are field size, input mask, validation rule, and required, as explained below.

- o Field Size

It is up to 255 characters.

- o Input Mask

An input mask ensures that the data will fit in the format you define, and you can specify the kind of values that can be entered in each blank space. For example, the input mask of OSUID is “0000000000000000”, which requires that all entries contain exactly 16 digits and only digits be entered in each field. Some Input Mask characters are given below in Table III.

0	Digit must be entered
9	Digit or space - entry not required
#	Digit or space - entry not required - spaces removed when data saved
L	Letter - entry required
A	Letter or digit - entry required
&	Any character or space - entry required
.,;-/	Placeholders/separators
<	Conversion to lowercase
>	Conversion to uppercase

Table III. Input Mask Characters

- o Validation Rule

You can use a validation rule to restrict data entry to certain types of data. The validation rule allows the field designer to put in an expression to test the data for acceptability. The associated error message can be put in the Validation Text to



display. For example, the validation rule on Computer-Based TOEFL field is “<=300 And >=0” and the Degree field is “Like "B.S." Or "M.S." Or "Ph.D.””. To input this validation rule, you may go to the “Validation Rule” of the field, press the build button (three dots) to print up the expression builder, and then enter the validation rule as shown below in Figure 20.

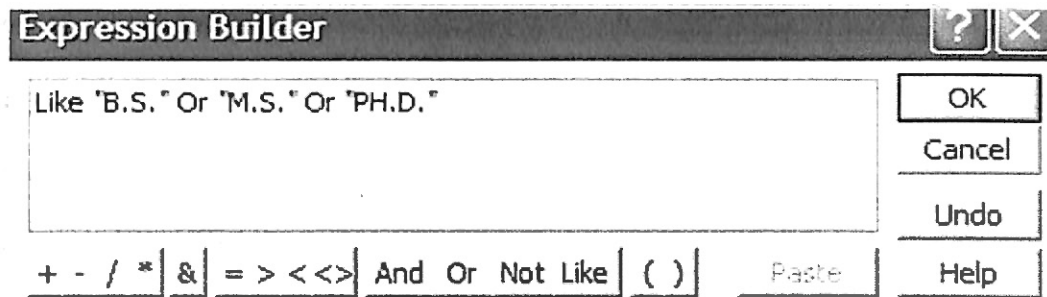


Figure 20. Validation Rule on the Degree Field of the Enrollment Table

- o Required

If a required entry is set to “yes”, it means that something must be entered in this field. The required property of the OSUID field, shown in Figure 19, is set to yes and it does not allow users to enter a record without an OSUID data, or only delete an OSUID data from an existing record.

## 2. Rename a Field

One of the jobs involved with database design and maintenance is to review fields and make sure they are explicit enough for the user. A field's name is stored in the corresponding table and is involved in performing calculations and other programming issues. If the renamed fields are involved in some relationships, you should be careful

when deciding to change their name. To rename a field in the Design view, click it and type the new name. After you have typed the name, the new name will replace the old one.

### 3. Delete a Field

When in the Design view, you can delete a field that you do not need anymore or was added by mistake. If the delete field is part of one or more relationships, you can not delete it unless you delete its relationships in the Relationship window first. To delete a column, you should carry out the following steps:

- ▶ right-click anywhere on its line and click Delete Rows,
- ▶ click Yes to permanently delete the field(s), and
- ▶ save and close the table.

### **Add a New Table**

The three ways to create a new table are: create a table in the Design view, create a table by using the Wizard, and create a table by entering data. The preferred way is using the Design view, which consists of creating a list of field names, specifying the data type, and controlling their properties. Follow these steps to create a new table in the Design view.

- ▶ Click Insert -> Table -> Design view on the menu bar. You can also click Tables under Objects, double click Create table in the Design view on the right side, then Figure 21 will be displayed.
- ▶ Follow “Add a Field” which was discussed on page 41 to define each of the fields in this table.

- ▶ Select the primary key.
- ▶ Click View bar (circled in Figure 21), click Yes to save the table.
- ▶ Name the table and click OK.

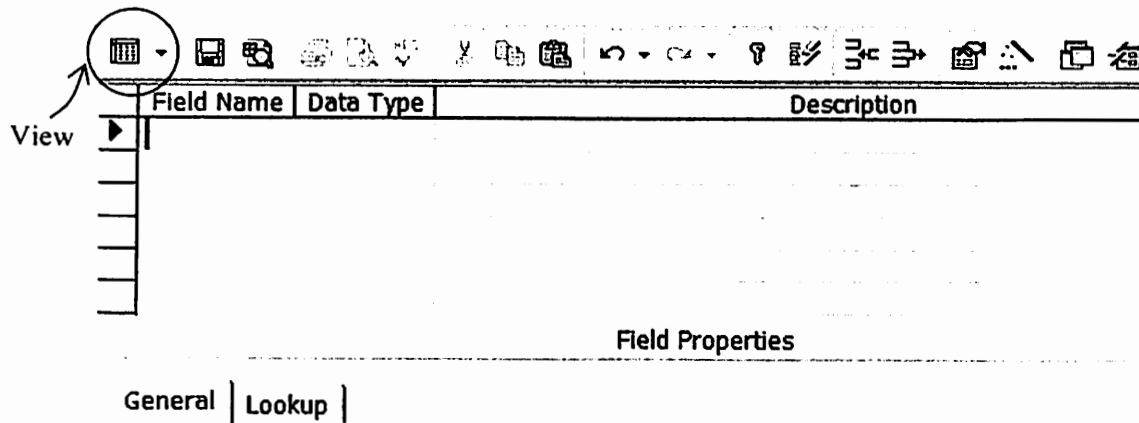


Figure 21. Create a New Table in Design view

### Build Relationship among Tables

When fields, which have relationships, are changed or a new table has been created, one or more new relationships between the tables could be built. The related fields do not have to have the same names. However, related fields must have the same data type and properties setting. The followings are the necessary steps to create a relationship between two tables.

- ▶ Click Relationships icon on the menu bar.
- ▶ Click Relationships -> Show Table.
- ▶ Double click the table you want. Then it is shown in the Relationships window.
- ▶ Click on the common field in this table and drag the mouse to the common field in the

related table. A dialog box (Figure 22) is brought up in which you can specify the relationship between the two tables.

- ▶ Click Create.

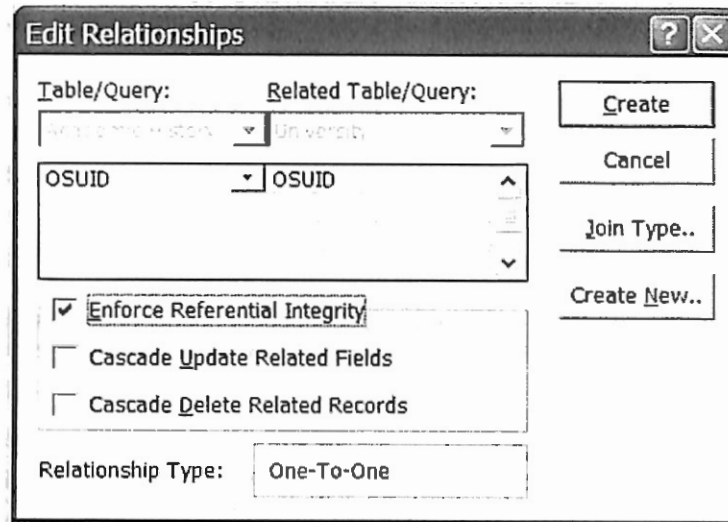


Figure 22. Create New Relationships Among Tables

### 5.1.2 Database Queries

A Query is issued to find and retrieve just the data that meets the specified conditions, update or delete records, or perform predefined or custom calculations on data. It may include data from multiple tables or other queries and also can be used as a source of records for forms and for other queries. When dealing with a query whose data originate from more than one table, those tables must have been previously joined. You can add a query if the data you need does not exist in the query, or remove a query if you decide you do not need them.

There are several types of queries in MS Access. The CSGraduateStudent database only involves the select query and the crosstab query. In the current complementation of

the database, there are 62 select queries and 19 crosstab queries.

### Create a Select Query

A select query is the most common type of query. It retrieves data from one or more tables and displays the results in a datasheet. It is also used to group records, count records, and calculate sums. Creating a select query can be accomplished by using either the query Design view or the Query wizard (Figure 23).

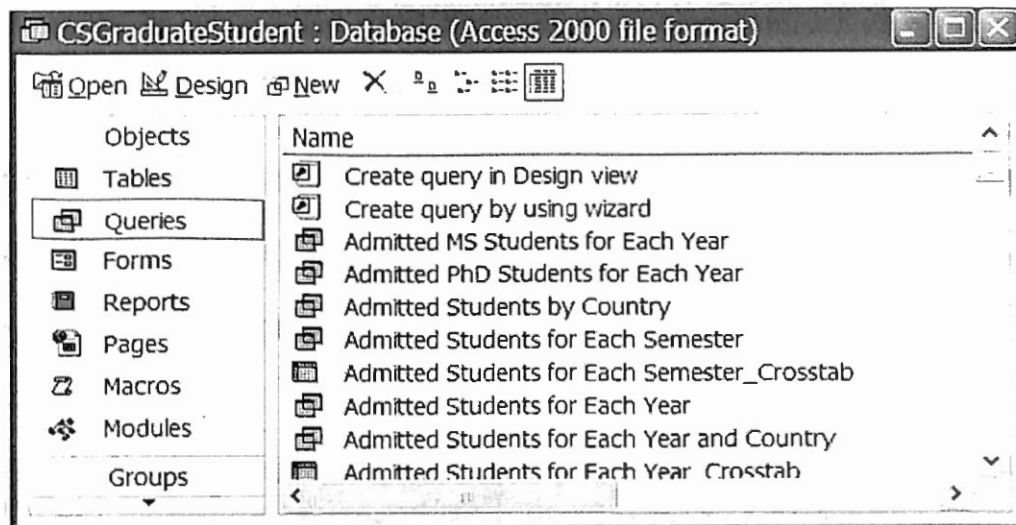


Figure 23. CSGraduateStudent Queries

The steps to create a select query by using wizard are:

- ▶ Click on the Queries under Objects in the Database window.
- ▶ Pull down the Tables/Queries list and choose the table or query source. Fields should be displayed in the Available Fields.
- ▶ Double click the fields you want in Available Fields, then these fields are displayed in Selected Fields.

- ▶ Click on the Next button to move to the next item.
- ▶ Give your new query a name.
- ▶ In the final step, the wizard will create the new query with the option of open or modify. If you choose “Open the query” to view the information and click on the Finish button, the wizard will execute the query and show the data. If you choose “Modify the query design”, the wizard will switch to the Design view to allow modifications to the query, as depicted in as Figure 24.

### Edit a Select Query

Open up the “Enrolled M.S. Students Name query” in the Design view by highlighting the name of the query and clicking on the Design view button (circled in Figure 24). After the query is modified, click View bar to see the datasheet, save and close this query to return to the Database window.

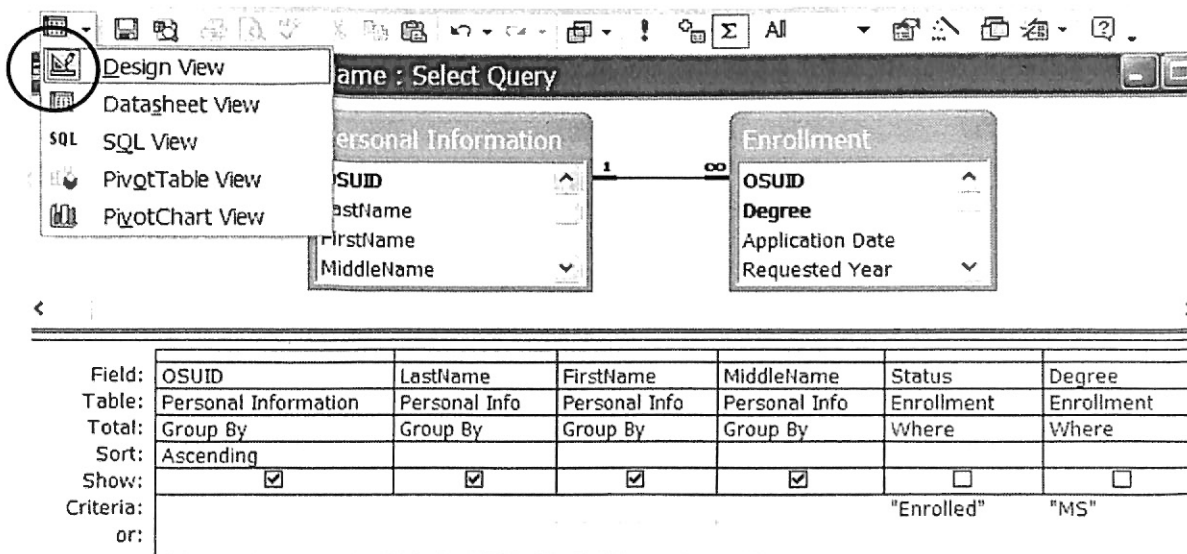


Figure 24. Design View of Enrolled M.S. Students Name Query

The Query Design view has two major sections. In the top section, the tables used for the query are displayed along with the available fields. In the bottom section, those fields that have been selected for use in the query are displayed. Each field has several options associated with it: Field, Table, Total, Sort, Show, and Criteria.

- Total

Use "Group By" to calculate separate amounts for groups of records in a field. Use an aggregate function, such as Sum Count or Avg, to calculate one amount for all the records in each field.

- Sort

You should arrange the fields you want to sort from left to right, since the leftmost field will be sorted first. The "Sort" order is by ascending, descending, or not sorted.

- Show

This field will be displayed in the query output or not.

- Criteria

The most important part of the query design is the criteria which controls how to limit the records in the query's results. The "Or" row is used for alternative criteria in the same field. Some examples of criteria are:

Expression	Meaning
>550	TOFEL score greater than 550
"M.S."	list all M.S. students
In("Enrolled","Graduated")	The status is one of the values in parentheses
Is Null, Is Not Null	The field is blank or not blank
And, Or, Between, Not	compound condition

Table IV. Query Criteria

For this example, to filter the records to only display enrolled Master students, “Enrolled” is typed by clicking the Criteria area beneath the Status field and “M.S.” is typed by clicking the Criteria area beneath the Degree field.

### Create a Crosstab Query

Crosstab query is used to calculate and restructure data for easier analysis. Enrolled Students for Each Year\_Crosstab query (Figure 25) in the CSGraduateStudent database is an example of Crosstab query. There are at least three fields of data to create a crosstab query. In this example, the Requested Year field is displayed as row heading, the Degree field (M.S. and Ph.D.) is displayed as column heading, and then calculated the total enrolled student by Degree and Requested Year.

View

Requested Year	MS	PHD	Total Enrolled Student
2000			5
2001	21	3	24
2002	24	3	27
2003	45	3	48
2004	8	1	9

Record: 1 of 5

Figure 25. Enrolled Students for Each Year\_Crosstab Query

To create the Enrolled Students for Each Year\_Crosstab crosstab query, do the following steps.

- ▶ Click on the Queries under Objects in the Database window.



- ▶ Click the New button at the top of the Database window.
- ▶ Select the Crosstab Query Wizard and click OK.
- ▶ Select the table or query you would like to pull your fields from. In this case, “Enrolled Students for Each Year query” is selected. Then click Next.
- ▶ Select the field (Required Year) you would like to use as row a header. Click Next.
- ▶ Select the field (Degree) you would like to use as a column header.
- ▶ Select the type of calculation you like to perform. In this case, Sum is selected. Click Next.
- ▶ Name the created query. In this case, the default name, “Each Year\_Crosstab crosstab”, is used. Click Finish and Figure 25 is displayed. If you like to make changes to the query, click the View button at the top left of the screen.

### **Field Replacement or Deletion in a Query**

A query provides just a temporary means of studying the information in the database, and we can add, insert, delete, replace, or move the fields at will.

If you do not need a field in a query any more, you can either replace it with another field or delete it. To replace a field, click the arrow on the combo box of Field and select a different field name from the list. To delete a field, click the gray bar of the column header in the Design view to select the whole field, click the right mouse button, and press Cut.

#### **5.1.3 Database Forms**

Forms are used as an alternative way to easily view, enter, and change data directly in a table. It is often impossible to enter all of the information from a single table,

especially if the set of tables has relationships. In the CSGraduateStudent database, there are more than 100 fields, and 5 tables are involved to each graduate applicant when a student data are first entered. The Computer Science Graduate Student Information form (Figure 7) is designed to help the database manager when larger amount of data are entered.

### **Create Forms by Using the Wizard**

To create a form using the assistance of the wizard, follow these steps:

- ▶ Click the Create form by using the wizard option on the Database window.
- ▶ Select the table or query from the Tables/Queries drop-down menu.
- ▶ Select the fields that will be included on the form by highlighting each on the Available Fields window and clicking the single right arrow button > to move the field to the Selected Fields window. To move all of the fields to Select Fields, click the double right arrow button >>.
- ▶ Click the Next button to move on to the next screen.
- ▶ Select the layout of the form and click Next.
- ▶ Select a style of the form and click Next.
- ▶ Name the form and select "Open the form to view or enter information" to open the form in Form View or "Modify the form's design" to open it in the Design view.
- ▶ Click Finish to create the form.

### **Create Subform by Using Wizard**

The CSGraduateStudent database also contains two other forms: Course form and

Assistantship form. These two forms deal with the enrolled students whose records need to be frequently edited. Subform and drop-down lists controls (Figure 26) are added to these forms to make data entry easier and more reliable.

A subform is a form that is placed in a parent form, called the main form. Subforms are particularly useful to display data from tables and queries that have one-to-many relationships. For example, in Figure 26 below, data on the main form is drawn from student Personal Information table while the subform contains all of the courses for that student. "One" enrolled student has "many" related courses.

The screenshot shows a window titled "Courses" with two subforms. The "Students" subform has the following fields:

- OSUID: 0000000000000001
- Degree: PHD
- Last Name: Garcia
- First Name: Carol
- Middle Name: (empty)

The "Courses" subform is a table with the following data:

CourseID	Grade	Index
cs5423	A	C
cs5553	B	C
cs5653	B	C
cs5663	B	C
cs1003		
cs1103		
cs1113		
cs2133		
cs2301		
cs2331		
cs2351		
cs2432		

A dropdown menu is open over the table, showing a list of course IDs from cs1003 to cs2432. The status bar at the bottom indicates "Record: 1 of 402".

Figure 26. Courses Form

Follow these steps to create a subform within a form:

- ▶ Double-click Create form by using the wizard on the Database window.

- ▶ Select the first table or query from Tables/Queries drop-down menu and choose the fields that should be displayed on the form.
- ▶ Select another table or query from the same window and choose the fields that should be displayed on the form. Click Next.
- ▶ Choose a table as the main form by selecting form with subform(s), and click Next.
- ▶ Do the rest of the steps as in “Create Form” by Using the Wizard.

### **Add or Edit Records Using a Form**

A new record can be created by clicking the New Record button at the bottom of the form window or by clicking Tab after the last field of the last record, as shown in Figure 26. Fill out the data into the blank fields of the form. The records or data are automatically stored into table as they are entered.

### **Editing Forms**

The following points may be helpful when modifying forms in the Design view.

- ▶ Grid Lines - By default, a series of lines and dots underlie the form in Design view so form elements can be easily aligned.
- ▶ Resizing Objects - Form objects can be resized by clicking and dragging the handles on the edges and corners of the element with the mouse.
- ▶ Change Form Object Type – Right click on the object with the mouse and select “Change To” and select an available object type from the list.
- ▶ Label/Object Alignment - To change the position of the object and label in relation to each other, click and drag the large handle at the top left corner of the object or label.

- ▶ Form Appearance - Change the background color of the form by clicking the Fill/Back Color button on the formatting toolbar and click one of the color swatches on the palette. The font and size, font effect, font alignment, border around each object, the border width, and a special effect can also be modified using the formatting toolbar.
- ▶ Page numbers can also be added to these sections by selecting Insert| Page Numbers.
- ▶ A date and time can be added from Insert| Date and Time.

## 5.2 Web Database Display

The CSGraduateStudent database, ASP pages, and Images folder are stored in a CD named CSDB. Appendix G lists all the .asp pages and image files in this database-driven Web site. Upload all the files from the CD to the directories indicated. If any files need to be overwritten, back up your old files for safety. If you do not back up these files and you want to start over, you may have a problem.

### 5.2.1. Hosting

An ASP file on the Web-server is requested by the way of a URL, such as <http://www.domainName/YourAccount/login.asp>, where the domainName is the Web server name of the CS computer system hosting this database-driven Web site, and YourAccount is given by the system administrator which provides you the space to upload all the Web database files.

### 5.2.2. ASP Files

Some ASP pages which will be introduced here are: Login.asp, DatabasePath.asp,

Default.asp, Header.asp, and Search.asp.

### **Login.asp**

The user ID and password record set have been stored in one of the CSGraduateStudent database table named Users. The login details are picked up from the login page using the Request method [Buyens 2000], i.e., if the record set is found, the login status is changed to login and the Web visitor is redirected to enter the Website, otherwise an error message is shown.

### **DatabasePath.asp**

To connect to the database using the physical path to the ASP file, you must first determine the path. DatabasePath.asp file provides the data source for other ASP files. As it has been mentioned before (Section 5.1.1), the database name must be changed from CSGraduateStudent to something else when the Web-driven database site is actually run. The following DatabasePath.asp code is included in most ASP pages by using `<!--#include file=" DatabasePath.asp"-->`.

```
<%  
xDb_Conn_Str = "Provider = Microsoft.Jet.OLEDB.4.0;  
Data Source = " & server.mappath ("CSGraduateStudent.mdb") & ";"  
%>
```

So, after the database name is changed, the only thing you need to do is to update the DatabasePath.asp script to specify this path. Otherwise, you should go through all the ASP files related to the database.

### **Default.asp:**

When you successfully login to the Web database site, the server calls default.asp to direct an ASP file. In the following default.asp code, Students.asp page will be shown on the Web site. You may change the Students.asp to what you want as the default page.

```
<% Response.Redirect "Students.asp" %>
```

### **Header.asp**

A header file will feature a CS top banner and a menu with different links that will be featured on the left hand side of each CS Graduate Student Web page. By design, all pages across the CS graduate student domain will look and feel the same. If the CS Department wants to change the look and feel of the Web page design, the designer can make one change in a single location and all Web pages that use the header file will be changed.

In this CS Graduate Student Web site, there are several header ASP files that are included in other ASP pages for various required purposes. These header files appear after the login status is checked and the database connection is opened at the top of the page. And they should be placed before the data is displayed. For example, the Header.asp (Figure 27) is included in all statistics ASP files. The code for Header.asp is included in Appendix H.

### **Search.asp**

The Search.asp facility allows the Web visitors to search the student record through the CSGraduateStudent database. Search.asp refers to Search\_Results.asp page (Figure 28)

to give a list of OSUID and student name records which contains the searched Keyword.  
 Click on the “View” to see the individual records form the database.

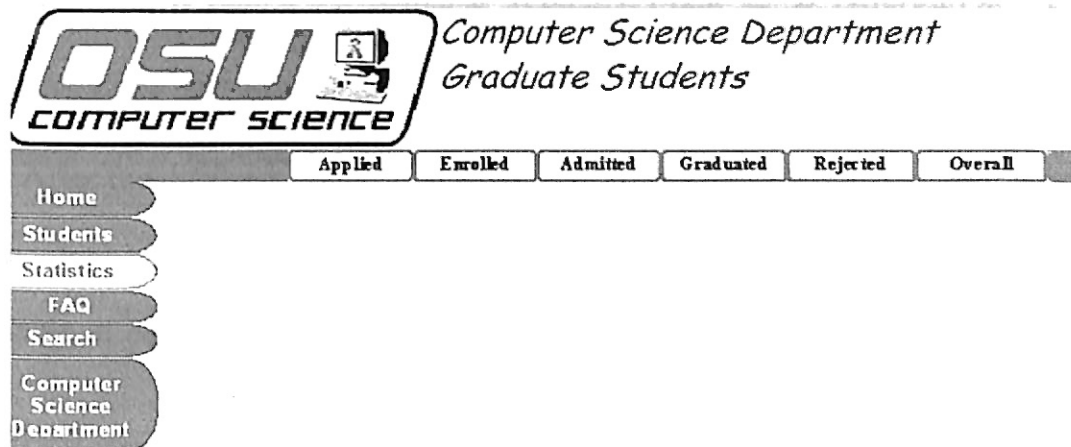


Figure 27. Header.asp Page

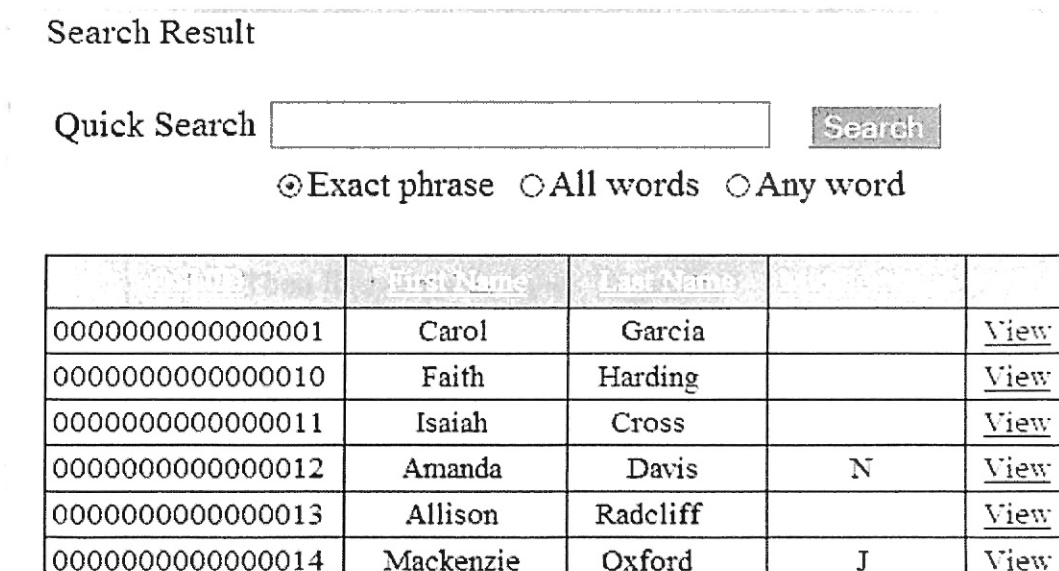


Figure 28. Search\_Results.asp Page



The main code in Search.asp is given below.

```
For Each kw In arpSearch
    b_search = b_search & "("
    b_search = b_search & "[OSUID] LIKE '%" & Trim(kw) &
        "%' OR "
```

After the database connection is created, the `b_search` builds the search criteria of the data which equals a value- OSUID, FirstName, LastName, etc. The code for Search.asp is available in Appendix H.

### 5.2.3. Create or Edit ASP File

The structure and sample code of ASP pages which deal with the extracting data from tables and queries are shown in Table V. We will follow this structure to create and edit the ASP files.

1. Check the Login Status: The code for checking the status follows.

```
<% If Session("CSDB_status") <> "login"
    Then Response.Redirect "login.asp"
%>
<%
Response.expires = 0
Response.expiresabsolute = Now() - 1
Response.addHeader "pragma", "no-cache"
Response.addHeader "cache-control", "private"
Response.CacheControl = "no-cache"
%>
```

The first three lines of the above code indicate that the Web visitors must re-login to the database if the default login status is expired. The rest of the lines tell the server that the login will expire if this Web page does not get a response within a finite time. The

Session object may keep the variables stored until the Web visitor leaves the domain. The timeout property can be set to the number of minutes that the Session lasts before it is ended by calling the Abandon method and then redirecting to a new page.

1. Check the login status	<% If Session("CSDB_status") <> "login" Then Response.Redirect "login.asp" %>
2. Include the database path file	<!--#include file="DatabasePath.asp"-->
3. Open the database connection	<% Set conn = Server.CreateObject ("ADODB.Connection") conn.Open xDb_Conn_Str %>
4. Build SQL	<% strsql = "SELECT * FROM [Graduated MS Students Name] order by ClassID" %>
5. Include the header file	<!--#include file="header.asp"-->
6. Read the data	<% X_OSUID = rs("OSUID") %>
7. Display the data	<% Response.Write X_ClassID %>
8. Close the database connection	<% conn.Close Set conn = Nothing %>

Table V. Code Structure of CS Graduate Student ASP Files

2. Include the database path file at the top of your source code by using an include statement.
3. Create and Open Database Connection Object: Server is an object that is built into the ASP scripting environment and CreateObject is a method to create objects. Creating a

database connection object does not automatically open communications with any database. The `conn.open` method is called to open a connection.

4. **Build SQL:** Embedded SQL allows programmers to connect to a database and process data from a database. The standard SQL commands such as `Select`, `From`, `Where`, and `Order-by` can be used to accomplish almost everything that one needs to do with a database.
5. **Include the header file** by using an `include` statement.
6. **Read the Data:** You should read and define all of the data you need in order to program or display from the database tables or queries.
7. **Display the Data:** The `Response` Object is responsible for sending data from the client to the server. For example, you can print strings such as HTML and character sets. The `Write` method writes a specified string to the current HTTP output using this syntax: `“Response.Write variant”`, where `Variant` can be any data type supported by the Visual Basic Scripting data type, including characters, strings, and integers.
8. **Close Database Connection:** When the database is finished accessing, use the `conn.Close` to close the connection. If you forget to close a `Connection` object, the ASP scripting processor will eventually find it, close it, and remove it from memory [Buyens 2000]. Even after you close a `Connection` object, it still remains in memory. To remove a `Connection` object, use `“Set conn = Nothing”` in the respective code. The server will probably be more stable if you always close and release any objects you create.

## CHAPTER VI

### SUMMARY AND FUTURE WORK

#### 6.1 Summary

This thesis work had two parts: database design and Web database display. The first part involved the design and implementation of a dynamic database application using the MS Access database management system to build tables, queries, and forms. This database has the capability to generate different types of queries and forms to suit the CS Graduate Student Information system requirements. The implementation was refined and streamlined using data constructed closely based on actual data.

The second part involved developing Web pages to display the desired tables and queries. The Web pages for the final implementation are Active Server Pages. Finally, a series of data tests and security tests were performed to insure that the system works correctly.

#### 6.2 Future Work

In the current system, the users are limited to edit or create new attributes based on existing attributes present in the corresponding tables from the Web site. In the future, the users should be able to create new attributes based on existing attributes from across multiple tables in the database. This would allow users to go in and edit their current

information, as opposed to emailing the site administrator and having the administrator remove old information, and then adding in a new entry. Due to limited time, this thesis work did not attempt to implement this function.

The current system has been designed to support only the MS Access database as the source database. In the future, the system can be enhanced to support multiple source databases.

Due to the author's limited knowledge of ASP, HTML, the MS Access database, and lack of time to learn the three completely, this DBMS does not include any error checking code. There are a vast number of errors that could be checked for, e.g., when the database manager entered a record in a table which is related to other tables, the other tables should have some necessary data entry (such as OSUID) to make them join together. An error checking code may test it and prompt the related tables and the fields.

The implementation part of this thesis work is meant to be used by a small group of users, releasing it on a wide scale would involve some improvements to the search and security functionalities.

## REFERENCES

- [Atkinson 2000] Leon Atkinson, *Core PHP Programming: Using PHP to Build Dynamic Web Sites*, 2nd edition, Prentice Hall PTR, Upper Saddle River, NJ, August 2000.
- [Axmark and Widenius 1999] David Axmark and Michael Widenius, *Linux Journal*, "MySQL Introduction", URL: <http://www.linuxjournal.com/article.php?sid=3609>, creation date: November 1999, access date: August 2004.
- [Buyens 2000] Jim Buyens, *Web Database Development Step by Step*, Microsoft Press, Redmond, WA, 2000.
- [Corkhill 2001] Bruce Corkhill, "Creating Your First ASP Page", Web Wiz Guide, URL: [http://www.webwizguide.com/asp/tutorials/first\\_asp\\_page.asp](http://www.webwizguide.com/asp/tutorials/first_asp_page.asp), creation date: September 2001, access date: January 2004.
- [Delwiche and Slaughter 2002] Lora D. Delwiche and Susan J. Slaughter, *The Little SAS Book: A Primer*, 2nd edition, SAS Inst. revised, SAS Publishing, Cary, NC, 2002.
- [Getz et al. 2001] Ken Getz, Mike Gunderloy, and Paul Litwin, *Access 2002 Developer's Handbook Set*, Sybex Inc., Alameda, CA, 2001.
- [Kroenke 1997] M. David Kroenke, *Database Processing: Fundamentals, Design, & Implementation*, 6th edition, Prentice Hall, Upper Saddle River, NJ, 1997.
- [Martella et al. 1994] Giancarlo Martella, Maria G. Fugini, and Silvana Castano, *Database Security*, Addison-Wesley, New York, NY, 1994.
- [Meloni 2000] Julie Meloni, *PHP Essentials*, Prima Tech Linux Series, Rocklin, CA, April 2000.

- [MS 2001] "When to Upsize a Microsoft Access Database to Microsoft SQL Server -- Improved Security", Microsoft Corporation, URL: [msdn.microsoft.com/library/en-us/off2000/html/acconWhenToUpsizeMDBtoSQLServer.asp](http://msdn.microsoft.com/library/en-us/off2000/html/acconWhenToUpsizeMDBtoSQLServer.asp), creation date: June 2001, access date: January 2004.
- [MS-1 2002] "Get Started with Access 2003", Microsoft Corporation, URL: <http://office.microsoft.com/assistance/preview.aspx?AssetID=HP051863841033&CTT=98>, creation date: 2002, access date: January 2004.
- [MS-2 2003] "Glossary -- FrontPage 2003 Assistance", Microsoft Corporation, URL: <http://office.microsoft.com/assistance/preview.aspx?AssetID=HP010381371033&CTT=4&Origin=CH010503741033>, creation date: February 2003, access date: January 2004.
- [MySQL 2004] "MySQL AB Trademark Policy", URL: <http://www.mysql.com/>, creation date: January 2004, access date: August 2004.
- [Olson et al. 2003] A. Michael Olson, Keith Bostic, and Margo Seltzer, "Berkeley DB", Sleepycat Software Inc., URL: <http://www.sleepycat.com/docs/ref/refs/bdbusenix.html>, creation date: December 2003, access date: January 2004.
- [Orwant et al. 2000] Jon Orwant, Larry Wall, and Tom Christiansen, *Programming Perl*, 3rd edition, O'Reilly & Associates Inc., Sebastopol, CA, 2000.
- [Sarin 2000] Sumit Sarin, *Oracle DBA Tips and Techniques*, McGraw-Hill Osborne Media, Berkeley, CA, 2000.
- [Schlotzhauer and Littell 1997] Sandra Schlotzhauer and Ramon C. Littell, *SAS System for Elementary Statistical Analysis*, 2nd edition, SAS Institute Inc., Cary, NC, 1997.
- [Shelly et al. 2001] B. Gary Shelly, Hilip J. Pratt, Mary Z. Last, and Thomas J. Chashman, *Microsoft Access 2002: Complete Concepts and Techniques*, Course Technology, Boston, MA, 2001.
- [Sleepycat 2001] Sleepycat Software Inc., *Berkeley DB*, Sams Publishing, Indianapolis, IN, 2001.

- [SPD 2001] “Sun Product Documentation-- Enable or Disable the Basic Security Module (BSM) on Solaris”, Sun Microsystems, URL: <http://docs.sun.com/db/doc/816-0211/6m6nc66nj?a=view>, creation date: May 2001, access date: January 2004.
- [Spector 2001] Paul E. Spector, *SAS Programming for Researchers and Social Scientists*, 2nd edition, Sage Publications, Thousand Oaks, CA, 2001.
- [Widenius et al. 2002] Michael Widenius, David Axmark, and MySQL AB, *MySQL Reference Manual: Documentation from the Source*, O'Reilly & Associates Inc., Sebastopol, CA, July 2002.
- [Yank 2003] Kevin Yank, *Build Your Own Database Driven Website Using PHP & MySQL*, Sitepoint Pty Ltd., Quartz Hill, CA, March 2003.



**APPENDICES**

## APPENDIX A

### GLOSSARY

ASP	In MS Access, Active Server Page is a set of software components that run on a Web server and allow Web developers to build dynamic Web pages [Buyens 2000].
Candidate Key	A combination of one or more fields whose value uniquely identifies a record in a table, hence no two records in a table can have the same key value.
Data Access Page	A Web page that has a connection to a database.
DBM	DataBase Manager is a program that consists of the server part and the client part of a distributed database.
DBMS	A DataBase Management System provides users with the software tools needed to organize and access data.
Field	A logical group of bytes in a record that is used in file processing.
Foreign Key	A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.
Form	An Access database object for taking actions or for entering, displaying, and editing data in fields.
IIS	Internet Information Server is Microsoft's suite of Internet-related software which is included with the Windows 2000 and above operating system software. IIS provides both FTP server and web server capabilities.
MS Access	Microsoft Access database is a collection of data and objects such as tables, queries, and forms.
MySQL	MySQL is an open source relational database management system that uses Structured Query Language for processing the data in a

database. It works best when used for managing content and not for executing transactions [Widenius et al. 2002].

Object	A term used to refer to tables, reports, indexes, or other structures of a database.
ODBC	Open DataBase Connectivity is a standard method of sharing data between databases and programs. ODBC drivers use Structured Query Language (SQL) to gain access to external data.
Primary Key	One or more fields (columns) whose value or values uniquely identify each record in a table. A primary key cannot allow null values and must always have a unique index [Shelly et al. 2001].
QBE	Query By Example is a style of query interface that allows users to express queries by providing examples of the results they seek.
Query	A question about the data stored in tables, or a request to perform an action on the data. A query can bring together data from multiple tables in order to serve as the source of data for a form, report, or data access page.
Report	An Access database object that prints information formatted and organized according to given specifications.
SAS	The Statistical Analysis System is an integrated system of software components providing complete control over data access, management, analysis, and presentation.
SQL	Structured Query Language is a language that is used for defining the structure and processing of a relational database. It is used as a stand-alone query language, or it may be embedded in application programs. SQL has been accepted as a national standard by the American National Standards Institute. It was developed by IBM.
Table	A database object that stores data in records (rows) and fields (columns).

## APPENDIX B

### TRADEMARK INFORMATION

ASP is a registered trademark of Microsoft Corporation.

JSP is a registered trademark of Sun Microsystems, Inc.

Microsoft, Microsoft Access, Microsoft FrontPage, SQL server, Windows, Windows 2000, Windows NT, and Windows XP are registered trademarks or trademarks of Microsoft Corporation.

MySQL is a registered trademark of MySQL AB.

OS/2 is a registered trademark of International Business Machines Corporation.

Other companies, products, and services mentioned in this document may be trademarks or registered trademarks of their respective owners.

PHP is a registered trademark of The PHP Group.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc.

Solaris is a registered trademark of Sun Microsystems, Inc.

SQL is a registered trademark of International Business Machines Corporation.

## APPENDIX C

### WEB VISITOR MANUAL

This database contains Computer Science Department Graduate Student information. When you login to the database Web site, you will be shown a CS logo and a link to the CS home page on top of all pages. All CS Web pages also contain menu bars on the left and some links on the bottom. The left bar always provides links to the different functions of the database which include Student, Statistic, Search, and FAQ. The bottom list incorporates the links from the menu, and also includes Logout and Contact. The Logout section lets you exit the Web database system and the Contact section lets you submit questions or comments to Web database manager.

#### 1. Logging on to the CS Database

Access the CS Web database through the Login page that appears when you visit the Web address provided by your system administrator. Using the UserID and password that the Web database manager provided, login to the database system. If you want to change the UserID and password, contact the Web database manager. The first screen you will see when coming to the Web database site is the Home page. This screen lays out all the main functions of the database.

## 2. Students Pages

Click Student link in left menu bars which is available from all this Web database pages, the student information screen is shown up. There is a group of links on the top of all student pages, just below the CS logo and link. The links of students are organized by enrolled student, graduate student, and so on. Choose the link of interest and you may view the specified student information in detail. Click on any blue hyperlinked database fields name to sort the record by ascending or descending.

## 3. Statistics Pages

Click Statistic link in left menu bars which is available from all this Web database pages, the Statistic screen is shown up. You may narrow the topic by clicking a link on the top menu, just below the CS logo and link. After choosing a subject, a listing of related statistic pages will appear. Select the title which you would like, it will bring you to the more detailed statistic description of the topic.

## 4. Search

Search Database is a quick search feature that lets users enter a keyword to search for student information. By selecting Search (either at the left or bottom of the screen), you can search database by keywords. You have the ability to choose a more advanced method of search by selecting any words, all words, or exact phrases.

## 5. FAQ

You can select FAQ (either at the left or bottom of the screen) to go to the FAQ

section where you may find answers to some of the more frequently asked questions.

If you need additional help of the database, please feel free to contact the database manager through the “Contact” link on the site.

## APPENDIX D

### WEB DATABASE MANAGER MANUAL

The CS Graduate Student Web Database System is supported by the Computer Science Department. In order to obtain authorization to view or update the CS Graduate Student Web Database System, you must contact a system manager at the CS Department.

#### 1. Installation of Web Database Files

Go to the C:/Inetpub/wwwroot directory of your computer. Insert the supplied CD in the CD drive. Copy the entire CSDB folder in the CD into the directory and ensure that the directory structure has been preserved when you extract the files from the CD. Then, you should have the following files in your computer:

C:\Inetpub\wwwroot\CSDB\Images

C:\Inetpub\wwwroot\CSDB\CSGraduateStudent.dbm

C:\Inetpub\wwwroot\CSDB\all the ASP files

You may rename CSDB folder if you want. Open <http://localhost/CSStudent> using your web browser, the login page should show up. You can use the UserID “guest” and password “123456” to login to the Web database. Since there is no data in the database, the Students and Statistics pages have no data.



## 2. Configuring Database and ASP File

There are a few things you should setup to make the Web database work correctly.

- The attributes of CSGraduateStudent database is set to Read Only. Highlight the file in your computer, right-click and select Attributes. Remove the tick from the Read Only check box.
- Rename the CSGraduateStudent.mdb database to whatever you want. This is very important for security considerations. Let's use Students.mdb for example.
- Find the DatabasePath.asp. Open it to edit. Change the

```
server.mappath("CSGraduateStudent.mdb") to
```

```
server.mappath("Students.mdb")
```

Open <http://localhost/CSStudent> using your web browser, login to the CS Graduate Student Web Database System to see if the changes were made successfully.

## 3. Creating User Codes and Password

Open Students.mdb. Double click the Users table to open it. Enter a ten character UserName and a six character Password to create a new user. Then, click the Save button to record the entry. You may change the format property, and for validation one would need a UserName and a Password. The valid Web visitors may use these UserID and Password to login to the CS Graduate Student Web Database System.

## 4. Data Entry and Update

You may use tables or forms to enter or edit data *in the Student database*. The recommend method is using Students.mdb forms.

## 5. Upload Files to System Server

Every time a file is updated, you should upload the file to the server. Make sure the file property in server is marked as read.

## 6. Backing up Your Data

It is the responsibility of the Web database manager to backup data on a regular basis. The ASP files and CSGraduateStudent database may be re-installed in the event of a hard disk failure or other critical problems, but the data will not be included. The file to backup for your data is CSGraduateStudent.mdb. This database file will be found in the CSDB directory of the supplied CD. The backup must be made to a device such as a floppy, a disk, or a CD, i.e., any storage device other than the hard disk on which the original data resides.

## APPENDIX E

### SYSTEM ADMINISTRATOR MANUAL

#### **System Requirement**

##### 1. Web Database Server

- Software Requirements

Operating System: Win NT, XP, or Win 2000 Server or up

Web Server: Internet Information Server (IIS) Version 4 and up

Web Browser: Microsoft Internet Explorer 5.0 or greater

Database: Microsoft Access 2000 or 2002

Other: VBScript Version 5.0 or later.

- Hardware Requirements

CPU: 400Hz or higher

Memory: 128MB or higher

Available Disk Space: 20 MB (or greater). 10 GB if students' theses are stored also.

Network: Use 10/100 network cards for better performance.

CD ROM Drive: 48X and up

##### 2. Client Workstation

- Software Requirements

Operating System: Microsoft Windows 98/2000/XP/NT

Microsoft Internet Explorer 5.0 or greater

Microsoft FrontPage 2002 or up

Microsoft Access 2000 or 2002

- **Hardware Requirements**

CPU: Pentium 200 Mhz (or greater)

Memory: 64MB (or greater)

Available Disk Space: 20 MB (or greater)

Monitor: Color SVGA

Network: Network Interface card and cable

CD ROM Drive: 48X and up

## **Installation**

### **1. Installation on the Server**

The supplied CD includes a CSDB folder which contains ASP files and a folder named Images. By default for IIS, the CSDB is entirely installed into the /Inetpub/wwwroot directory.

Insert the CD in the CD drive. Copy all of the files in the CSDB into the directory directly. When copying files, make sure that the directory structure remains unchanged.

When this is completed, you should have the following path:

`\Inetpub\wwwroot\CSDB\Images`

`\Inetpub\wwwroot\CSDB\CSGraduateStudent.dbm`

`\Inetpub\wwwroot\CSDB\all the ASP files`

Rename CSDB folder to NewDirectory, for example, and generate the login name and password for the Web database manager so that he/she can login to the NewDirectory folder and edit the files. This folder's name is part of the CS database-driven Web site address: <http://www.ServerName/NewDirectory>. The login name and the password are given to the Web database manager for possible further reference.

## 2. Installation on Client Computer

Go to the /Inetpub/wwwroot directory. Insert the CD in the CD drive. Copy the CSDB folder in the CD into the directory. When this is completed, you should have the following path:

\Inetpub\wwwroot\CSDB\Images

\Inetpub\wwwroot\CSDB\CSGraduateStudent.dbm

\Inetpub\wwwroot\CSDB\all the ASP files

Open <http://localhost/CSDB> using your web browser to see if the installation was made successfully.

### **Backing up the CSDB System**

It is the responsibility of the system manager to backup the CSDB on a regular basis. The backup should be made to a device such as a floppy, a disk, or a CD, i.e., any storage device other than the hard disk on which the original data resides. Then, the CSDB database may be re-installed in the event of a hard disk failure or other critical problems.

## **Uninstallation**

To uninstall the program from your web server or client, just delete the NewDirectory folder and all files.

## **Running the Web Database System for the First Time**

Type the Web address in the Web browser, the CS Graduate Student Login page will now be enabled. Enter the UserID “guest” and password “123456” to login the Web database.

## APPENDIX F

### CSDB TABLES AND QUERIES LIST

This appendix contains the CSDB tables and the list of queries. The following two sections contain the names of 14 tables and 81 queries in the CSGraduateStudent database.

#### APPENDIX F.1 TABLES

Academic History

Address

Assistantship

Courses

Enrollment

Enrolled Semester Code

Graduated Semester Code

Personal Information

Requested Semester Code

Students and Courses

Thesis and Dissertation

University

Users

Withdrawn Semester Code

## APPENDIX F.2 QUERIES

Admitted MS Students for Each Year  
Admitted PhD Students for Each Year  
Admitted Students by Country  
Admitted Students for Each Semester  
Admitted Students for Each Semester\_Crosstab  
Admitted Students for Each Year  
Admitted Students for Each Year\_Crosstab  
Admitted students with Degree from OSU  
Admitted students with Degree from OSU Name  
Admitted TOEFL and GRE Scores for Each Semester  
Admitted TOEFL and GRE Scores for Each Year  
Age\_Applied/Enrolled/Graduated Students  
All Gender  
All Students TOEFL and GRE Scores for Each Semester  
All Students TOEFL and GRE Scores for Each Year  
Applicants Who Graduated from OSU\_CS  
Applicants Who Graduated from OSU\_Name  
Applicants Who Graduated from OSU\_Total  
Applied Date  
Applied Status  
Applied Students by Country  
Applied Students for Each Semester  
Applied Students for Each Semester\_Crosstab  
Applied Students for Each Year  
Applied Students for Each Year\_Crosstab  
Assistantship Query  
Assistantship Semester  
Core Courses  
Country



Country\_Crosstab  
Enrolled MS Students Name  
Enrolled PhD Students Name  
Enrolled Students for Each Semester  
Enrolled Students for Each Semester\_Crosstab  
Enrolled Students for Each Year  
Enrolled Students for Each Year\_Crosstab  
Gender  
Gender Admitted by Semester  
Gender Admitted by Semester\_Crosstab  
Gender Admitted by Year  
Gender Admitted by Year\_Crosstab  
Gender by Semester  
Gender by Semester\_Crosstab  
Gender by Year  
Gender by Year\_Crosstab  
Gender \_All Applied by Semester  
Gender \_All Applied by Year  
Gender \_Applied by Semester  
Gender \_Applied by Year  
Graduated MS Students Name  
Graduated PhD Students Name  
Graduated Students for Each Semester  
Graduated Students for Each Semester\_Crosstab  
Graduated Students for Each Year  
Graduated Students for Each Year\_Crosstab  
MS with Degree from OSU  
Pending MS Students Name  
Pending PhD Students Name  
PhD with Degree from OSU  
Postponed MS Students Name

Postponed PhD Students Name  
Prerequisite Courses  
Rejected MS Students Name  
Rejected PhD Students Name  
Rejected Reason  
Rejected Reason\_Crosstab  
Rejected Students for Each Semester  
Rejected Students for Each Semester\_Crosstab  
Rejected Students for Each Year  
Rejected Students for Each Year\_Crosstab  
Rejected University  
Rejected University-China  
Rejected University-India  
Rejected University-USA  
Status  
Status Query  
Student Information  
TOEFL and GRE Score for Each Semester  
TOEFL and GRE Score for Each Year  
Withdrawn MS Students Name  
Withdrawn PhD Students Name

## APPENDIX G

### CSDB ASP AND IMAGE FILES LIST

This appendix contains the CSDB ASP and the list of image files. The following two sections contain the names of 113 ASP pages and 47 image files of the CSDB.

#### APPENDIX G.1 ASP FILES

Admitted\_MS\_Graduated\_from\_OSUlist.asp  
Admitted\_MS\_Graduated\_from\_OSU\_Namelist.asp  
Admitted\_PhD\_Graduated\_from\_OSUlist.asp  
Admitted\_PhD\_Graduated\_from\_OSU\_Namelist.asp  
Admitted\_Students\_by\_Countrylist.asp  
Admitted\_Students\_for\_Each\_Semesterlist.asp  
Admitted\_Students\_for\_Each\_Yearlist.asp  
Agelist.asp  
All\_Students\_Nerlist.asp  
All\_Students\_TOEFL\_and\_GRE\_Scores\_for\_Each\_Yearlist.asp  
Applicants\_MS\_Graduated\_from\_OSU\_Namelist.asp  
Applicants\_MS\_Graduated\_from\_OSU\_Totallist.asp  
Applicants\_PhD\_Graduated\_from\_OSU\_Namelist.asp  
Applicants\_PhD\_Graduated\_from\_OSU\_Totallist.asp  
Applied\_MS\_Graduated\_from\_OSUlist.asp  
Applied\_Students\_by\_Countrylist.asp  
Applied\_Students\_for\_Each\_Semesterlist.asp  
Applied\_Students\_for\_Each\_Yearlist.asp  
Assistantship\_Querylist.asp  
Assistantship\_Semester.asp

Countrylist.asp  
DatabasePath.asp  
default.asp  
Enrolled\_MS\_Graduated\_from\_OSUlist.asp  
Enrolled\_MS\_Graduated\_from\_OSU\_Namelist.asp  
Enrolled\_PhD\_Graduated\_from\_OSUlist.asp  
Enrolled\_PhD\_Graduated\_from\_OSU\_Namelist.asp  
Enrolled\_PhD\_Students\_Namelist.asp  
Enrolled\_Students\_by\_Countrylist.asp  
Enrolled\_Students\_for\_Each\_Semesterlist.asp  
Enrolled\_Students\_for\_Each\_Yearlist.asp  
FAQ.asp  
Footer.asp  
Genderlist.asp  
Gender\_Admitted\_by\_Semesterlist.asp  
Gender\_Admitted\_by\_Yearlist.asp  
Gender\_Applied\_by\_Semesterlist.asp  
Gender\_Applied\_by\_Yearlist.asp  
Gender\_Enrolled\_by\_Semesterlist.asp  
Gender\_Enrolled\_by\_Yearlist.asp  
Gender\_Graduated\_by\_Semesterlist.asp  
Gender\_Graduated\_by\_Yearlist.asp  
Gender\_Rejected\_by\_Semesterlist.asp  
Gender\_Rejected\_by\_Yearlist.asp  
Graduated\_MS\_Graduated\_from\_OSUlist.asp  
Graduated\_MS\_Graduated\_from\_OSU\_Namelist.asp  
Graduated\_MS\_Students\_Namelist.asp  
Graduated\_PhD\_Graduated\_from\_OSUlist.asp  
Graduated\_PhD\_Graduated\_from\_OSU\_Namelist.asp  
Graduated\_PhD\_Students\_Namelist.asp  
Graduated\_Students\_by\_Countrylist.asp

Graduated\_Students\_for\_Each\_Semesterlist.asp  
Graduated\_Students\_for\_Each\_Yearlist.asp  
Header.asp  
Header\_A.asp  
Header\_All.asp  
Header\_Enrolled.asp  
Header\_FAQ.asp  
Header\_Graduated.asp  
Header\_Pending.asp  
Header\_Postponed.asp  
Header\_Rejected.asp  
Header\_Search.asp  
Header\_Withdrawn.asp  
Home.asp  
login.asp  
logout.asp  
pass.js  
Pending\_MS\_Students\_Namelist.asp  
Pending\_PhD\_Students\_Namelist.asp  
Postponed\_MS\_Students\_Namelist.asp  
Postponed\_PhD\_Students\_Namelist.asp  
Rejected\_MS\_Graduated\_from\_OSUlist.asp  
Rejected\_MS\_Graduated\_from\_OSU\_Namelist.asp  
Rejected\_MS\_Students\_Namelist.asp  
Rejected\_PhD\_Graduated\_from\_OSUlist.asp  
Rejected\_PhD\_Graduated\_from\_OSU\_Namelist.asp  
Rejected\_PhD\_Students\_Namelist.asp  
Rejected\_Reason.asp  
Rejected\_Students\_by\_Countrylist.asp  
Rejected\_Students\_for\_Each\_Semesterlist.asp  
Rejected\_Students\_for\_Each\_Yearlist.asp

Rejected\_UniversityD\_Chinalist.asp  
Rejected\_UniversityD\_Indialist.asp  
Rejected\_UniversityD\_USAlist.asp  
Rejected\_Universitylist.asp  
Search.asp  
Search\_Name\_Results.asp  
Search\_OSUID\_Results.asp  
Search\_Results.asp  
Statistic.asp  
Statistic\_Admitted.asp  
Statistic\_Applied.asp  
Statistic\_Enrolled.asp  
Statistic\_Graduated.asp  
Statistic\_Rejected.asp  
Students.asp  
Student\_Information\_view.asp  
TOEFL\_and\_GRE\_Admitted\_for\_Each\_Semesterlist.asp  
TOEFL\_and\_GRE\_Admitted\_for\_Each\_Yearlist.asp  
TOEFL\_and\_GRE\_Applied\_for\_Each\_Semesterlist.asp  
TOEFL\_and\_GRE\_Applied\_for\_Each\_Yearlist.asp  
TOEFL\_and\_GRE\_Enrolled\_for\_Each\_Semesterlist.asp  
TOEFL\_and\_GRE\_Enrolled\_for\_Each\_Yearlist.asp  
TOEFL\_and\_GRE\_Graduated\_for\_Each\_Semesterlist.asp  
TOEFL\_and\_GRE\_Graduated\_for\_Each\_Yearlist.asp  
TOEFL\_and\_GRE\_Rejected\_for\_Each\_Semesterlist.asp  
TOEFL\_and\_GRE\_Rejected\_for\_Each\_Yearlist.asp  
Withdrawn\_MS\_Students\_Namelist.asp  
Withdrawn\_PhD\_Students\_Namelist.asp

## APPENDIX G.2 IMAGE FILES IN THE IMAGES FOLDER

01.png  
AcaHis.gif  
Address.gif  
Coures\_Thesis.gif  
currentbanner.jpg  
Enrollment.gif  
first.gif  
firstdisab.gif  
last.gif  
lastdisab.gif  
login\_orange.gif  
Logout.gif  
menu.gif  
menu\_Admitted.gif  
menu\_All.gif  
menu\_All1.gif  
menu\_Applied.gif  
menu\_Assis.gif  
menu\_Assis1.gif  
menu\_Enrolled.gif  
*menu\_Enrolled1.gif*  
menu\_Enrolled2.gif  
menu\_FAQ.gif  
menu\_Graduated.gif  
menu\_Graduated1.gif  
menu\_Graduated2.gif  
menu\_Overall.gif  
menu\_Pending.gif  
menu\_Pending1.gif

menu\_Postponed.gif  
menu\_Postponed1.gif  
menu\_Rejected.gif  
menu\_Rejected1.gif  
menu\_Rejected2.gif  
menu\_Search.gif  
menu\_Statistics.gif  
menu\_Student.gif  
menu\_Withdrawn.gif  
menu\_Withdrawn1.gif  
newlogo4.gif  
next.gif  
nextdisab.gif  
paper\_blue.gif  
paper\_orange.gif  
PerInf.gif  
prev.gif  
prevdisab.gif



## APPENDIX H

### CSDB ASP CODES

This appendix contains the CSDB ASP codes. The following two sections contain CSDB ASP codes: Header.asp and Search.asp.

#### APPENDIX H.1 Header.asp Code

```
<html>

' Setup page frame
<body leftmargin="0" topmargin="0" marginheight="0" marginwidth="0"
    background="images/newlogo4.gif">

' Begin wrapper table
<table width="760" height="322" border="0" cellspacing="0" cellpadding="0"
    style="border-collapse: collapse" bordercolor="#111111">
  <tr align="center"> <td colspan="2" height="32" width="660">
    <p style="margin-top: 0; margin-bottom: 0" align="left">
      <a href="http://www.cs.okstate.edu">
        </a><i><font color="#0000FF" size="4.5"
            face="Comic Sans MS"><span style="vertical-align: middle">Computer Science
            Department Graduate Students</font></i>&nbsp;
        <p style="margin-top: 0; margin-bottom: 0">&nbsp;&nbsp;&nbsp;</p>

<table width="216" border="0" cellpadding="0" cellspacing="0" align="left">
<tr valign="top" align="left">
  <td nowrap width="640" align="center">
  </td>
</tr>
</table>

' End wrapper table

' Build the top menu and links
<tr><td width="180" height="1" nowrap align="left" valign="bottom"></td></tr>
<tr><td width="170" height="22" nowrap align="right" bgcolor="#FF6600"></td>
```

```

<td width="580" height="22" nowrap align="right" bgcolor="#FF6600">
<p align="left">
<a href="Statistic_Applied.asp">
  </a> <a href="Statistic_Enrolled.asp"><img border="0" src=
  "images/menu_Enrolled1.GIF"></a><a href="Statistic_Admitted.asp"></a><a
  href="Statistic_Graduated.asp"></a><a
  href="Statistic_Rejected.asp"></a><a href="Statistic.asp"></a><a href="logout.asp"></a></td>
</tr>
' End the top menu and links

```

```

' Build the left column menu and links

```

```

<tr>
  <td width="180" height="266" valign="top">
    <map name="FPMap0">
      <area href="Students.asp" shape="rect" coords="0, 24, 96, 49">
      <area coords="1, 49, 96, 75" shape="rect" href="statistic.asp">
      <area coords="0, 75, 96, 96" shape="rect" href="Faq.asp">
      <area href="Search.asp" shape="rect" coords="0, 99, 96, 120">
      <area href="http://www.cs.okstate.edu" shape="rect" coords="1, 120, 96, 183">
      <area href="http://Home.asp" shape="rect" coords="1, 0, 94, 22">
    </map>
    <p>
    &nbsp;<p>&nbsp;</td>

```

```

'End the left column menu and links

```

```

'Right column

```

```

<td width="358" valign="top" height="266">
<p>&nbsp;</p>
</td>
</tr>
</html>

```

## APPENDIX H.2 Search.asp Code

```

'Check the login status

```

```

<% If Session("CSDB_status") <> "login" Then Response.Redirect "login.asp" %>
<%

```

```

Response.expires = 0
Response.expiresabsolute = Now() - 1
Response.addHeader "pragma", "no-cache"
Response.addHeader "cache-control", "private"
Response.CacheControl = "no-cache"
%>

```

```

<!--#include file=" DatabasePath.asp"-->

```

```

' Set the display size

```

```

<%
displayRecs = 20
recRange = 10
%>

```

```

' Set the search parameters

```

```

<%
dbwhere = ""
masterdetailwhere = ""
searchwhere = ""
a_search = ""
b_search = ""
whereClause = ""
%>

```

```

' Get search criteria for basic search

```

```

<%
pSearch = Request.QueryString("psearch")
pSearchType = Request.QueryString("psearchType")

```

```

' Set the search fields of the database

```

```

If pSearch <> "" Then
    pSearch = Replace(pSearch,"","")
    pSearch = Replace(pSearch,"[","[[")
    If pSearchType <> "" Then
        While InStr(pSearch, "[") > 0
            pSearch = Replace(pSearch, "[", " ")
        Wend
        arpSearch = Split(Trim(pSearch), " ")
    
```

```

' Search the fields as "Exact phrase"

```

```

For Each kw In arpSearch
    b_search = b_search & "("
    b_search = b_search & "[OSUID] LIKE '%" & Trim(kw) & "%' OR "
    b_search = b_search & "[LastName] LIKE '%" & Trim(kw) & "%' OR "
    b_search = b_search & "[FirstName] LIKE '%" & Trim(kw) & "%' OR "

```

b\_search = b\_search & "[MiddleName] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Gender] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Birth Country] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Visa type] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Academic Misconduct] LIKE '%" & Trim(kw) & "%'  
 OR "  
 b\_search = b\_search & "[Street Address] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[City] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[State] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Country] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Zip Code] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Phone] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Fax number] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[PermanentAddress] LIKE '%" & Trim(kw) & "%' OR "  
 "  
 b\_search = b\_search & "[Permanent City] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Permanent State] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Permanent Country] LIKE '%" & Trim(kw) & "%' OR "  
 "  
 b\_search = b\_search & "[Permanent Zip Code] LIKE '%" & Trim(kw) & "%'  
 OR "  
 b\_search = b\_search & "[Permanent Phone] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[EMAIL] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Non-OSU EMAIL] LIKE '%" & Trim(kw) & "%' OR "  
 "  
 b\_search = b\_search & "[Requested Year] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Requested Semester] LIKE '%" & Trim(kw) & "%'  
 OR "  
 b\_search = b\_search & "[Degree] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Status] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Enrolled Year] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Enrolled Semester] LIKE '%" & Trim(kw) & "%' OR "  
 "  
 b\_search = b\_search & "[Graduated Year] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Graduated Semester] LIKE '%" & Trim(kw) & "%'  
 OR "  
 b\_search = b\_search & "[Reason] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Postponed or Withdrawn Year] LIKE '%" & Trim(kw)  
 & "%' OR "  
 b\_search = b\_search & "[Postponed or Withdrawn Semester] LIKE '%" &  
 Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Miscellaneous] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[GRE Adv Type] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[University 1] LIKE '%" & Trim(kw) & "%' OR "  
 b\_search = b\_search & "[Location, City and Country 1] LIKE '%" & Trim(kw)  
 & "%' OR "

```

b_search = b_search & "[Major 1] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Degree Earned 1] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[University 2] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Location, City and Country 2] LIKE '%" & Trim(kw)
& "%' OR "
b_search = b_search & "[Major 2] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Degree Earned 2] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[University 3] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Location, City and Country 3] LIKE '%" & Trim(kw)
& "%' OR "
b_search = b_search & "[Major 3] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Degree Earned 3] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[University 4] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Location, City and Country 4] LIKE '%" & Trim(kw)
& "%' OR "
b_search = b_search & "[Major 4] LIKE '%" & Trim(kw) & "%' OR "
b_search = b_search & "[Degree Earned 4] LIKE '%" & Trim(kw) & "%' OR "
If Right(b_search, 4)=" OR " Then b_search = Left(b_search, Len(b_search)-4)
b_search = b_search & ") " & pSearchType & " "

```

Next

‘ Search the fields as “Any word”

Else

```

b_search = b_search & "[OSUID] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[LastName] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[FirstName] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[MiddleName] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Gender] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Birth Country] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Visa type] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Academic Misconduct] LIKE '%" & pSearch & "%'
OR "
b_search = b_search & "[Street Address] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[City] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[State] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Country] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Zip Code] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Phone] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Fax number] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[PermanentAddress] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Permanent City] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Permanent State] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Permanent Country] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Permanent Zip Code] LIKE '%" & pSearch & "%' OR
"
b_search = b_search & "[Permanent Phone] LIKE '%" & pSearch & "%' OR "

```

```

b_search = b_search & "[EMAIL] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Non-OSU EMAIL] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Requested Year] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Requested Semester] LIKE '%" & pSearch & "%' OR
"
b_search = b_search & "[Degree] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Status] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Enrolled Year] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Enrolled Semester] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Graduated Year] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Graduated Semester] LIKE '%" & pSearch & "%' OR
"
b_search = b_search & "[Reason] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Postponed or Withdrawn Year] LIKE '%" & pSearch
& "%' OR "
b_search = b_search & "[Postponed or Withdrawn Semester] LIKE '%" &
pSearch & "%' OR "
b_search = b_search & "[Miscellaneous] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[GRE Adv Type] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[University 1] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Location, City and Country 1] LIKE '%" & pSearch &
"%' OR "
b_search = b_search & "[Major 1] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Degree Earned 1] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[University 2] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Location, City and Country 2] LIKE '%" & pSearch &
"%' OR "
b_search = b_search & "[Major 2] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Degree Earned 2] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[University 3] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Location, City and Country 3] LIKE '%" & pSearch &
"%' OR "
b_search = b_search & "[Major 3] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Degree Earned 3] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[University 4] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Location, City and Country 4] LIKE '%" & pSearch &
"%' OR "
b_search = b_search & "[Major 4] LIKE '%" & pSearch & "%' OR "
b_search = b_search & "[Degree Earned 4] LIKE '%" & pSearch & "%' OR "

```

End If

End If

If Right(b\_search, 4) = " OR " Then b\_search = Left(b\_search, Len(b\_search)-4)

If Right(b\_search, 5) = " AND " Then b\_search = Left(b\_search, Len(b\_search)-5)

%>

' Build search criteria

```

<%
  If a_search <> "" Then
    searchwhere = a_search ' Advanced search
  ElseIf b_search <> "" Then
    searchwhere = b_search ' Basic search
  End If

  ' Save search criteria
  If searchwhere <> "" Then
    Session("Student_Information_searchwhere") = searchwhere

    ' Reset start record counter (new search)
    startRec = 1
    Session("Student_Information_REC") = startRec

  Else
    searchwhere = Session("Student_Information_searchwhere")
  End If
%>

' Get clear search cmd
<%
  If Request.QueryString("cmd").Count > 0 Then
    cmd = Request.QueryString("cmd")

    If UCase(cmd) = "RESET" Then
      ' Reset search criteria
      searchwhere = ""
      Session("Student_Information_searchwhere") = searchwhere

    ElseIf UCase(cmd) = "RESETALL" Then

      ' Reset search criteria
      searchwhere = ""
      Session("Student_Information_searchwhere") = searchwhere
    End If

    ' Reset start record counter (reset command)
    startRec = 1
    Session("Student_Information_REC") = startRec
  End If

  ' Build dbwhere
  If masterdetailwhere <> "" Then
    dbwhere = dbwhere & "(" & masterdetailwhere & ") AND "
  End If

```



```

    If searchwhere <> "" Then
        dbwhere = dbwhere & "(" & searchwhere & ") AND "
    End If
    If Len(dbwhere) > 5 Then
        dbwhere = Mid(dbwhere, 1, Len(dbwhere)-5) ' Trim rightmost AND
    End If
%>

' Load Default Order
<%
    DefaultOrder = ""
    DefaultOrderType = ""

' No Default Filter
    DefaultFilter = ""

' Check for an Order parameter
    OrderBy = ""
    If Request.QueryString("order").Count > 0 Then
        OrderBy = Request.QueryString("order")

        ' Check if an ASC/DESC toggle is required
        If Session("Student_Information_OB") = OrderBy Then
            If Session("Student_Information_OT") = "ASC" Then
                Session("Student_Information_OT") = "DESC"
            Else
                Session("Student_Information_OT") = "ASC"
            End if
        Else
            Session("Student_Information_OT") = "ASC"
        End If
        Session("Student_Information_OB") = OrderBy
        Session("Student_Information_REC") = 1
    Else
        OrderBy = Session("Student_Information_OB")

    If OrderBy = "" Then
        OrderBy = DefaultOrder
        Session("Student_Information_OB") = OrderBy
        Session("Student_Information_OT") = DefaultOrderType
    End If
End If

' Open connection to the database
Set conn = Server.CreateObject("ADODB.Connection")
conn.Open xDb_Conn_Str

```



```

' Build SQL
strsql = "SELECT * FROM [Student Information]"

If dbwhere <> "" Then
    whereClause = whereClause & "(" & dbwhere & ") AND "
End If

If Right(whereClause, 5) = " AND " Then whereClause = Left(whereClause,
    Len(whereClause)-5)

    If whereClause <> "" Then strsql = strsql & " WHERE " & whereClause
    End If

    If OrderBy <> "" Then strsql = strsql & " ORDER BY [" & OrderBy & "]" &
        Session("Student_Information_OT")
    End If

'Response.Write strsql
Set rs = Server.CreateObject("ADODB.Recordset")
rs.cursorlocation = 3
rs.Open strsql, conn, 1, 2
totalRecs = rs.RecordCount

' Check for a START parameter
If Request.QueryString("start").Count > 0 Then
    startRec = Request.QueryString("start")
    Session("Student_Information_REC") = startRec
ElseIf Request.QueryString("pageno").Count > 0 Then
    pageno = Request.QueryString("pageno")

    If IsNumeric(pageno) Then
        startRec = (pageno-1)*displayRecs+1

        If startRec <= 0 Then startRec = 1
        ElseIf startRec >= ((totalRecs-1)\displayRecs)*displayRecs+1 Then
            startRec = ((totalRecs-1)\displayRecs)*displayRecs+1
        End If

        Session("Student_Information_REC") = startRec
    Else
        startRec = Session("Student_Information_REC")
        If Not IsNumeric(startRec) Or startRec = "" Then
            startRec = 1 ' Reset start record counter
            Session("Student_Information_REC") = startRec
        End If
    End If

```

```

        End If
Else
    startRec = Session("Student_Information_REC")
    If Not IsNumeric(startRec) Or startRec = "" Then
        startRec = 1 'Reset start record counter
        Session("Student_Information_REC") = startRec
    End If
End If
End If
%>

<!--#include file="header.asp"-->
<p>Student Information</span></p>

' Search the student information using Student Name
<form action="Search_Name_Results.asp">
<table border="0" cellspacing="0" cellpadding="4" width="500" style="border-collapse:
collapse" bordercolor="#111111">
<tr><td width="114" bgcolor="#E1F0FF">Student Name</td>
<td width="370" bgcolor="#E1F0FF">
<input type="text" name="psearch" size="20">
<input type="submit" name="Submit" value="Search" style="color: #0000FF;
font-weight: bold; font-family: Arial"></span></td>
</tr>
<tr><td width="114" bgcolor="#E1F0FF">&nbsp;</td>
<td width="370" bgcolor="#E1F0FF">
<input type="radio" name="psearchtype" value="" checked>Exact phrase&nbsp;&nbsp;
<input type="radio" name="psearchtype" value="AND">All words&nbsp;&nbsp;
<input type="radio" name="psearchtype" value="OR">Any word</span></td></tr>
</table>
</form>

' Search the student information using Student OSUID
<p style="margin-top: 0; margin-bottom: 0">&nbsp;</p>
<form action="Search_OSUID_Results.asp">
<table border="0" cellspacing="0" cellpadding="4" width="500" style="border-collapse:
collapse" bordercolor="#111111">
<tr><td width="114" bgcolor="#E1F0FF">OSUID Number</td>
<td width="370" bgcolor="#E1F0FF">
<input type="text" name="psearch" size="20">
<input type="submit" name="Submit" value="Search" style="color: #0000FF;
font-weight: bold; font-family: Arial"></span></td>
</tr>
<tr><td width="114" bgcolor="#E1F0FF">&nbsp;</td>
<td width="370" bgcolor="#E1F0FF">
<input type="radio" name="psearchtype" value="" checked>Exact
phrase&nbsp;&nbsp;

```

```



```

```

' Search the student information using Key Word

```

```

<p style="margin-top: 0; margin-bottom: 0">&nbsp;</p>

```

```

<form action="Search_Results.asp">

```

```

<table border="0" cellspacing="0" cellpadding="4" width="500" style="border-collapse:
collapse" bordercolor="#111111">

```

```

<tr><td width="112" bgcolor="#E1F0FF">Key Word</span></td>

```

```

<td width="372" bgcolor="#E1F0FF">

```

```

<input type="text" name="psearch" size="20">

```

```

<input type="submit" name="Submit" value="Search" style="color: #0000FF;
font-weight: bold; font-family: Arial"></span></td>

```

```

</tr>

```

```

<tr><td width="112" bgcolor="#E1F0FF">&nbsp;</td>

```

```

<td width="372" bgcolor="#E1F0FF">

```

```

<input type="radio" name="psearchtype" value="" checked>Exact phrase&nbsp;

```

```

<input type="radio" name="psearchtype" value="AND">All words&nbsp;

```

```

<input type="radio" name="psearchtype" value="OR">Any word</span></td></tr>

```

```

</table>

```

```

</form>

```

```

*
```

```

<!--#include file="footer.asp"-->

```

VITA 

Jing Yang

Candidate for the Degree of

Master of Science

Thesis: A WEB DATABASE SYSTEM TO MANAGE GRADUATE STUDENT  
INFORMATION

Major Field: Computer Science

Biographical:

Personal Data: Born in Shanxi, P. R. China, on May 18, 1970, daughter of Xisheng Yang and Xiusheng Zhang.

Education: Received the Bachelor of Management Engineering degree from Beijing Institute of Machinery Industry in May 1993; completed the requirements for the degree of Master of Science in Computer Science at the Computer Science Department of Oklahoma State University in December 2004.

Experience: Worked with Tingyi International Food Corporation Ltd., China, as a manager from April 1994 to September 1999. Employed by the Computer Science Department as a Graduate Teaching Assistant from August 2001 to December 2004.

Honors: Member of Phi Kappa Phi.