Spring 5-1-2004

Normalization and Retrieval of Graduate Office Information for Dakota State University

Chris Olson
Dakota State University

DAKOTA STATE UNIVERSITY

MSIS
PROJECT APPROVAL FORM

Student Name: Chris Olson

Expected Graduation Date: May 2004

Master's Project Title: Normalization and Retrieval of Graduate Office Information for Dakota State University

Date Project Plan Approved: September 12, 2003

Date Project Coordinator Notified and Grade Submitted: May 4, 2004

Approvals/Signatures:

Student: Chris Olson — Date: May 18, 2004

Faculty supervisor: Wayne Pauli — Date: May 18, 2004

Committee member: Mike Shan — Date: May 18, 2004

Committee member: Lanis Dennis — Date: May 18, 2004

To Ensure Certification of Completion:
Student must bring or send the original to the Graduate Programs Office.
Copies on acid free paper go to the library with the reports for binding.

Original to Graduate Programs Office
Acid-free copies with written reports to library
Copies to: Project supervisor and committee and to MSIS Coordinator

Revised 1/28/03

Normalization and Retrieval of Information for the Graduate Office at Dakota State University

Dakota State University
GRADUATE PROGRAMS OFFICE

Chris Olson

A project submitted in partial fulfillment of the requirements for the
Master of Science in Information Systems
Dakota State University
2004

Abstract

The following report was written to detail the steps necessary to create a prototype that could potentially serve as the back end database for the redundant information that exists on the current graduate programs' portion of Dakota State University's web site. All of the pages mentioned are currently static. By creating a central repository for this data, it becomes possible to reuse the same web page to display different content. The use of this database with dynamic pages for data retrieval would be more efficient than the current way of doing things by eliminating the data anomalies and lessening the labor intensive process of changing the data in the multiple instances in which it presently occurs.

Preface

This MSIS project serves as the final piece of the puzzle in my education at Dakota State University. However, the overall learning puzzle will never be completed. I hope the knowledge that I've gained through my wonderful experiences at DSU – both educational and personal – will serve as a stepping stone to further my education or pursue a career in the Information Technology field.

I've learned many valuable lessons in life in since I began my college career – the most important being to have a vision for the future. Life can get in the way at times to cloud these visions, but overcoming obstacles that are thrown at a person makes succeeding all the more glorious.

I could not have made it this far without some wonderful influences at DSU. I can't possibly include everybody, but the following people (in no particular order) have contributed to making my education possible:

- Tom Farrell – In the spring semester of 1997, I took my first programming class from Tom. He showed me the basic aspects of programming and instilled an excitement in me that showed me I had chosen the correct career path. I took several classes from Tom throughout my years at DSU. I can vividly remember walking into his office the afternoon of January 25th, 2001. I had just been offered a job as a programmer at Federated Insurance and was looking for guidance. Although I was apprehensive to move so far from home, Tom's reassurance led me to accept the position. To this day, I wish I was able to fulfill that obligation I made, but life got in the way. I also took my first graduate level course from Tom. It was my first class after my paralysis, and it gave me the confidence to continue.
- Brent Tulloss – I had the privilege of working under Brent as an MIS Support Assistant at Rosco Manufacturing. I worked with him for ten months as an intern. Brent's technical and leadership abilities astounded a younger, impressionable version of me. The knowledge I gained from him and working in a real world environment supplemented my academic education more than I could have ever imagined. It was Brent that gave me my first exposure to a relational database. When I first started at Rosco, I attempted to run a query on the relational database that was the back end to the company's ERP system. When the system was nearly brought to a standstill, I was able to leave work early that afternoon. The next day I remember Brent saying to me, "I don't want to give you a complex, but…" My poorly designed query had brought the system to its knees. It was then that I began to better understand the importance of using the proper techniques for data management. DSU is now very lucky to have him as an instructor.
- Terry Dennis – Terry's understanding of how technology is used in the business world is beyond anyone's I have ever known. His lectures held my attention better than any other instructor I've ever had. He and his wife, Laurie, can be very proud of what they have done to build successful graduate programs at DSU. His expertise is outstanding, and I will think of him anytime I am trying to figure out how IT can be used to create a competitive advantage.

- Laurie Dennis – I don't even know how to begin to express my gratitude for all that Laurie has done for me. The accommodations that she has provided me in the last two and a half years through the graduate office have made my continued education possible. This project originally was done to make her job easier. I am sorry that I won't get to repay her for all that she has done for me. I am also very sorry that future graduate students won't have the privilege of getting to know her. She will be sorely missed at DSU, but I wish her nothing but the best as she moves on to the next stage in her life.
- Dr. Shan – I owe most of my database understanding to Dr. Shan. I took the undergraduate database courses at DSU from him, as well as the graduate level courses. He has served as a member of my project committee and has always been readily available to aid me in designing my database. He genuinely cares about the education of his students and is a great instructor.
- Wayne Pauli – Wayne served as my MSIS project advisor. I couldn't have asked for a better advisor. He has been both encouraging and reassuring as I went through the various stages of the turmoil involved with this project. Although I've never had the privilege of taking a course from Wayne, I can only hope that I could convey the same compassion and enthusiasm that he has as an instructor – if I choose that route. Above all, I am most honored that I consider Wayne a great friend. I have known him for a number of years and am most happy to know that he has always been concerned about my well being as a person.
- My family – My family has always been a source of encouragement to me in continuing my education, constantly reminded me of how proud it makes them to see me overcome my physical condition and work toward my vision.
- Tana Thompson – Tana was the biggest part of the vision I had for my life. She and I had a life planned together. The accident that left me paralyzed took her from me. I would have given up long ago if it wouldn't have been for her. Although she is no longer here physically, her spirit helps me get through each day. She has helped me to realize that there is only where I am today and what happened yesterday to put me here; and yes, there is a tomorrow, a tomorrow I cannot foresee or guarantee, but one that I can only prepare for by doing my best to hang in there today. Thank you, Tana.

There are several others that have influenced my life, but the above people are the ones most responsible for the success of my MSIS project and education.

Table of Contents

List of Appendices

List of Tables

List of Figures

Introduction

This project that is being submitted as a partial fulfillment to graduate with a Master of Science of Information Systems from Dakota State University is one that holds some valuable lessons about working as a team in a real world situation. It has undergone a series of changes that will be discussed before arriving to this point. This project has become a prototype, one that can be built on, added to, and utilized by the Graduate Office at Dakota State University to decrease the labor intensive tasks that are currently performed on a regular basis.

This report will outline in detail all of the steps that were taken to deliver a product that can be useable and productive in the future. This report will discuss:

- The problem statement
- Variance in plans over time
- The current way of doing things
- Reasons for solving the problem
- The deliverable products
- The scope of the project
- Results and conclusions

This project has utilized many skills that have been acquired over the last few years in the MSIS program. Without the valuable course content and knowledgeable faculty members, this project and subsequent report would not have been possible.

Statement of Problem

When searching for an MSIS project, it was a desire that one could be found that would:

- Be challenging
- Be useful
- Use knowledge gained in the MSIS core courses
- Use expanded knowledge from each of the specialization track courses from which skillsets were drawn – electronic commerce and data management.

Laurie Dennis, the Director of the Graduate Programs Office, sent an email with a problem that she had dealt with in her years at Dakota State University.

Following is an excerpt of that email:

----- Original Message -----
From: Dennis, Laurie
To: 'Chris Olson'
Sent: Wednesday, September 03, 2003 3:01 PM
Subject: RE: Fall 2003

The grad office web pages are static. But we use the same information and documents in several different places (orientation, MSET program, MSIS program documents). In addition, I use the same data in several different documents that I keep on file - for application packages, handbook, cost documents, etc. I'm pretty organized, but I have a difficult time making sure I keep all related documents updated when changes occur. I know it is possible to link Word documents so that when changes are made to the master document, the related documents are updated automatically. But I have not done that and that works only for the ones on my hard drive.

But I want to start with a dynamic website. I want more than a database that updates specific identical documents. I'd like to see if there is a way to have data in specific cells that would automatically update all documents that have that data included. For example, costs are included in several different documents on the website. Some are in tables, some in word documents as part of a different kind of table. When the tuition changes, I have to go to every instance that uses a tuition figure. Can we set up some sort of database such that every time a tuition figure is used, it gets that figure from the database?

Figure 1: Laurie's Email

That email became the starting point for the project. A dynamic website that would allow a user to change the data in one place and display that data in multiple places became the primary goal. Several secondary goals came and went during various phases of this project, but the primary goal was always the focus.

As it was alluded to in the email, the graduate office personnel currently seeks out all locations of the data that needs to be changed on the website. This would mean opening *Microsoft FrontPage*, finding the location of the data, and changing it in all of the places it appears. This is both time consuming and inefficient.

The first issues that had to be dealt with were which database to use as a back end and which type of scripting language should be used to retrieve the data. Correspondence was held with the Graduate Programs Director, the Graduate Office, the project advisor, and a project committee member regarding the best way to tackle the problem at hand. This took place early in the fall semester of 2003. It was decided that it be best to use *Microsoft SQL Server* or *Oracle* as the back end database because *Microsoft Access* was not used by industry in production. The idea of combining XML and Oracle was toyed with, before deciding on using *SQL Server* with *ASP.NET*. Dakota State University utilizes many *Microsoft* products, so this seemed to be a natural fit. Dr. Shan, DSU's database guru, also aided in this decision by suggesting *SQL Server* "for the project scale and cost savings purpose."

When the project began early in the spring semester of 2004, Dr. Shan was asked where the *SQL Server* database would be hosted and how it could be accessed. He then stated that DSU did not have the software for teaching and learning purposes and that

Microsoft Access could be used instead. This was the first time the plan had changed, but it was viewed as a fortunate change because:

- The advisor to this MSIS project, Wayne Pauli, pointed out that *Microsoft Access* is a viable database used by multi-million dollar foundations and businesses in the industry.
- *Microsoft Access 2003* allows for easy creation of Data Access Pages specifically geared for web applications. These pages allow a user to easily view, update, add, or delete data via the web.
- This project was designed for the graduate office personnel. *Microsoft Access* has a much lower learning curve than *SQL Server*. This would allow a novice user to more easily make structural changes to the database in the future, if needed.
- The Report feature in *Access* could easily generate reports from the data to be exported into *Microsoft Word*.
- Since DSU leans so heavily on the Microsoft Office Suite anyway, it doesn't make financial sense to purchase and host more database software that isn't as user-friendly.

Changing Focus

After a meeting with Laurie Dennis toward the end of the fall 2003 semester, the scope of the project changed for the first time. The possibility of making her local documents dynamic, along well as the website, was discussed. Because this was added, and the Masters of Science in Information Assurance program would also be added as a DSU graduate program, the scope of the project was scaled back so it would just focus on

the MSIS portion of it. The plan that was submitted reflects this, but it was changed shortly there after.

Nearly a month into the spring semester of 2004, the project underwent a dramatic change. Up to that point, most of the work had involved sorting through the MSIS portion of the graduate office website at http://www.departments.dsu.edu/gradoffice/ identifying the structured data that should be placed in the database that was going to be developed. The current graduate programs web site was being looked at when it was decided by others that the graduate programs' web site at DSU was to be completely redesigned. This meant that the design, layout, and content would be changing.

After being up in the air for a week or so, a new plan of action was put into place regarding the redesigning of the web site. Following is an excerpt of an email that was received regarding the new plan of action:

----- Original Message -----
From: Pauli, Wayne
To: Chris Olson
Cc: Dennis, Laurie ; Pauli, Wayne
Sent: Tuesday, February 10, 2004 1:22 PM
Subject: RE:

Chris, it appears as though the graduate office web site is going to become a coordinated project between four separate efforts:

1. Project Management
2. Web Development and Design
3. ASP development
4. Database

The web development and the ASP work is being directed to undergraduate students, the project management would become a project for another MSIS student, and that would leave your project for the coordination and the development of the back end. The web site is looking more like a hybrid of pages, with both static and dynamic functionality.

Figure 2: Wayne's Email

This new plan was very exciting, as it meant that the database portion would be an integral part of a group working to completely redesign the web site. This would also give this MSIS project more of a real world feel, since it would be a group collaboration, complete with a project manager for guidance.

After this was announced, the scope of the project had to be redefined again. Since another person would be responsible for the *ASP.NET* portion to retrieve the data, coupled with the new course of action that was put in place to redesign all of the graduate programs' web site, the back end database would now include the data found throughout the web site. In other words, the project was no longer limited to dealing with just the MSIS program. The new scope of the group effort required the structured data in the MSET and the newly added MSIA programs to be placed into the database as well.

A meeting was held February 18th with the project manager, the project advisor Wayne Pauli, and the graduate programs coordinator Terry Dennis. Laurie Dennis was absent due to illness, but had sent documentation that outlined which information on the current web site was dynamic and which was static. At the meeting several ideas and technologies were discussed.

The primary targeted user, Laurie Dennis, was spoken with the following week. User input is critical to the success of any IT project. More options were discussed at that time, including the possibility of adding a component to the database that would lessen the amount of labor currently done in the graduate office. Laurie was using a *Microsoft Excel* spreadsheet to track which forms students had completed. She would also have to review each student's Plan of Study and compare it with the courses the students were actually taking or had taken to see if students were on track to graduate.

Part of the initial appeal of this project was to make less work for the personnel in the graduate office. Because of this, automating the process of tracking student forms by having them available online in *ASP.NET* pages was discussed. When a student completed a form, it would be routed to the proper faculty members for approval. Once approved, the database could be updated by the use of a scripting or programming language. Also, students' coursework could be entered into forms each semester and compared to the Plan of Study. Reports could easily be generated by *Microsoft Access* and viewed each semester. This would aid the graduate office by showing the status of students – whether they were admitted to the program, for example, - if a student was missing forms, or if a student was going to graduate at the end of the semester. All of these examples were being performed by a very labor intensive process.

After discussing this new proposal of adding a "Student" portion to the database with the project manager and thinking about the complex way in which this would work, it was decided against adding this to the scope of the project. The rationale for this decision was as follows:

- This would add several hours to the ASP.NET programming portion of the project.
- A system was already in place for the users that was working and they were comfortable with.
- Colleague is a state system so the database could not be tied to it. This would mean that graduate office personnel would actually have to do *more* work by entering the courses the students were enrolled in each semester.

- This would increase the scope of the project dramatically, and the words, "But I want to start with a dynamic website" were reread. A "Students" database may be useful in the future to lessen the workload for the graduate office, but it was outside the realm of the current project.
- Time was becoming a factor due to the indecisiveness and changes the project had already undergone.

Project Collaboration

Online conversations were had with the project manager concerning the data that was unstructured. The different programs had paragraphs of text that were unique to them on the graduate web site. This was problematic because it was hard to logically fit this data in a normalized database. Different possibilities were discussed for using *Microsoft Office 2003* to deal with this. One idea was to move all of the forms, documents, etc. that were stored both locally and available for download on the web site to the web server. This would eliminate the process of updating a document locally, saving it locally, and then uploading it to the web server to ensure that it was current. Storing these documents on the web server would also eliminate the need for local back ups because DSU would back up the documents that were hosted on its servers.

After the correspondence with others involved in the project, the logical design phase of the database commenced. During this phase, the documentation that Laurie had prepared was used and the pages of the graduate programs' web site were viewed to determine what relations to use and the attributes that they would contain. As the complexity of the database increased, an inquiry was sent out about who was going to be

brought on to do the *ASP.NET* programming. This MSIS project was about to be redefined again, causing the problem to be reevaluated.

An *ASP.NET* programmer for this group effort could not be found. When informed of this, the possibility of adding the retrieval programming to the project was briefly discussed. This would mean a continuation for the project, as the programming would add a considerable amount of time to the database portion of the project.

Reevaluation of Project Deliverables

The semester was nearly half over. A programmer had not been found. The project manager decided to step down because the project wasn't the scale he thought it would be and there weren't as many people involved. All of these factors led to the reevaluation of the project. This included redefining the problem, restating the objectives, and retooling the Work Breakdown Structure and Gantt chart.

The project was now to become a prototype and would not be implemented. The concept of a prototype will be discussed later in this paper. This was hard to accept at first, because implementation of it could save work. However, it shows what could be done in the future to create a dynamic website for Dakota State University's graduate website.

A great deal of time has been spent discussing the underlying problem that was trying to be solved. This discussion was to show all of the changes that this MSIS project has undergone.

- Original project: the problem that needed to be solved was to make all documents and web pages that contained the same data dynamic instead of static. This would eliminate the update, insert, and delete anomalies. The first statement of the

problem only dealt with those documents and pages related to the MSIS program. The project was to encompass creating the database and retrieving the data.

- First Revision: the problem now included all of the graduate programs and focused on the web pages. The scope of the project was limited to the creation of the database. The retrieval was to be done by a separate ASP.NET programmer.
- Second Revision: the group project was aborted. This project became a prototype. An example of retrieval in a dynamic web page was to be included.

Defining the problem is a critical step before beginning any IT project. The program manager at Keller Graduate School of Management, William V. Leban, says the main reason projects fail is a lack of proper project definition and scope.[1] While this project wasn't a failure, it became very difficult at times because it was constantly being redefined. Scope will be addressed in the appropriate section.

Skillsets Utilized

This project was of great value. In order to address the problem, skills learned from the following courses were used:

- INFS 720 - Systems Analysis and Design Using Case-based Tools
- INFS 724 - Project and Change Management
- INFS 734 - Client Server Technologies
- INFS 760 - Database Design and Performance Analysis
- INFS 764 - Information Retrieval

[1] Chalfin, Natalie, "Four Reasons Why Projects Fail," *PM Network,* PMI (June 1998) 7.

Although the group effort was short lived, a valuable, real-world lesson in group IT projects was learned. This obstacle was overcome and a prototype that solves the problem – creating a dynamic website – was accomplished. Because of all of the changes that were made and the short time frame in which to work, this project isn't nearly what was originally envisioned. If it had been known from the start that project would also include the ASP.NET programming, a user-friendly web interface for manipulating and viewing the data could have properly developed.

Objectives

The project goal was to effectively organize all of Dakota State University's graduate programs' structured data and create a prototype database that could be used in the future to create a dynamic web site. Dynamic documents could also be easily generated from the data. This plan was put into action by normalizing the data and placing it into a database.

Throughout much of the semester, the impression was given that this database would be implemented. With less than half of the semester left, the project goal was to create a prototype. Prototyping is the process of building a model of a system; it is sometimes referred to as a "quick-and-dirty" program that provides a minimal amount of features.[2] All of the data would be in place to be used in the future if desired. Prototyping can come in many forms, with varying degrees of complexity. It is a useful tool in building an information system.

This project contains a *Microsoft Access* database with twelve different relations. These relations contain a total of 612 records at this time. The development of a database in third normal form was the primary goal. This was accomplished.

Because a programmer was never brought on board for this project within a project, an example of a web interface for retrieving the data was also created. *ASP.NET* was used for this. Some *Microsoft Data Access* pages were also created so that a user could view, update, add, or delete records in the database via the internet.

---

[2] Dennis, A., & Wixom, B.H., (2000). *Systems analysis and design*, New York, John Wiley & sons, inc., ISBN: 0471241008 pp. 11

Scope of the Project

All of the work involved in creating the products of the project and the processes used to create them are defined as scope.[3] Because the problem definition changed so frequently throughout this project, the scope also changed. The total scope of the project wasn't decided on until there was just a little over a month left in the semester.

During the analysis phase, it was determined that a database that contained well-structured relations was to be built. For relations to be well-structured, they must contain minimal redundancy and allow users to insert, modify, and delete rows in a table without errors or inconsistencies. Simply placing the data in tables would be unacceptable, as it would cause anomalies. An anomaly is an error or inconsistency that may result when a user attempts to update a table that contains redundant data.[4] The three types of anomalies are insertion, deletion, and modification (sometimes called update). Table 1 illustrates examples of these as it pertains to the data that was used.

Insertion Anomaly	Suppose a user wanted to add a new faculty member to tblProgramFaculty. In order to do this, the FacultyName must be added, as well as the ProgramID(since the relation has a composite primary key). This is an anomaly because a user should be able to add a faculty member without adding information to other tables. This has been eliminated and will be explained later.

[3] Schwalbe, Kathy, (2002). *Information Technology Project Management,* Boston, Course Technology/Thomson Learning pp. 84

[4] Hoffer, Jeffrey A., Prescott, Mary B., McFadden, Fred R., (2002). *Modern Database Management 6th Edition*, Prentice Hall pp174

Deletion Anomaly	If a faculty member leaves DSU and is to be deleted, the corresponding courses the faculty teaches would also have to be deleted, resulting in loss of course information.
Modification Anomaly	If a course that is taught in different programs or specialization has a change to its title, the change would have to be recorded in multiple places without properly normalized data.

Table 1: Anomalies

If those examples were a problem, the data in the database would be in relations that were not well-structured.

Data Redundancy and the ERD

Faculty and courses were the obvious attributes that were listed for each of the three programs in the graduate programs' website. The first step that was taken was to create an entity relationship diagram (ERD) that showed how all of the data was related. An ERD is a graphical representation of the entities and relationships between them.[5] It shows the entities, attributes, and relationships. It also displays connectivity and cardinality.

Identifying the entities and attributes was relatively easy. It became more complex when it came to defining relationships and cardinality – the number of occurrences that may exist between occurrences of two related entity types.[6] The cardinality may be one to one, one to many, or many to many.

[5] http://www.esri.com/library/glossary/e_h.html
[6] http://www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm

Before the cardinality constraints are broken down, the entities that were obvious will be listed. These were PROGRAMS, FACULTY, and COURSES. Without the complicated relationships, diagramming and normalizing would have been simple. This was not the case.

Cardinality

- **FACULTY teaches one or many COURSES.**

This was an obvious one and easy to identify. It mainly comes into play when discussing the schedules of the different courses, which depend upon the program and the semester.

- **FACULTY teaches in one or many PROGRAMS.**

Since some of the courses overlap between programs, specifically between the MSIS and MSIA programs, the faculty overlaps as well.

- **FACULTY has one or many DEGREES.**

Different faculty members have different numbers of degrees. This creates a multi-valued attribute.

- **A COURSE belongs to one or many PROGRAMS.**

A course may belong to more than one program. An example would be INFS 724, which is required in both the MSIS program and the MSIA program.

- **A COURSE has zero or many PREREQUISITES.**

A course may have zero, one, or many prerequisites.

- **A COURSE belongs to zero or many CATEGORIES.**

This cardinality rule really complicated things. As an example, INFS 732 belongs in three categories in the MSIS program – the Electronic Commerce Specialization, the

General Specialization, and the Network Administration and Security Specialization – and one category in the MSIA program – the Internet and E-Commerce Specialization. It is important to assign these courses to the proper specialization so that courses can be queried properly. If a part of a web page is to display the courses that must be completed in order to obtain a concentration/specialization, it's crucial that these courses are placed in the tblCourseCategory properly to do so.

Once the entities and the relationships with cardinalities were identified, the entity relationship diagram was constructed:

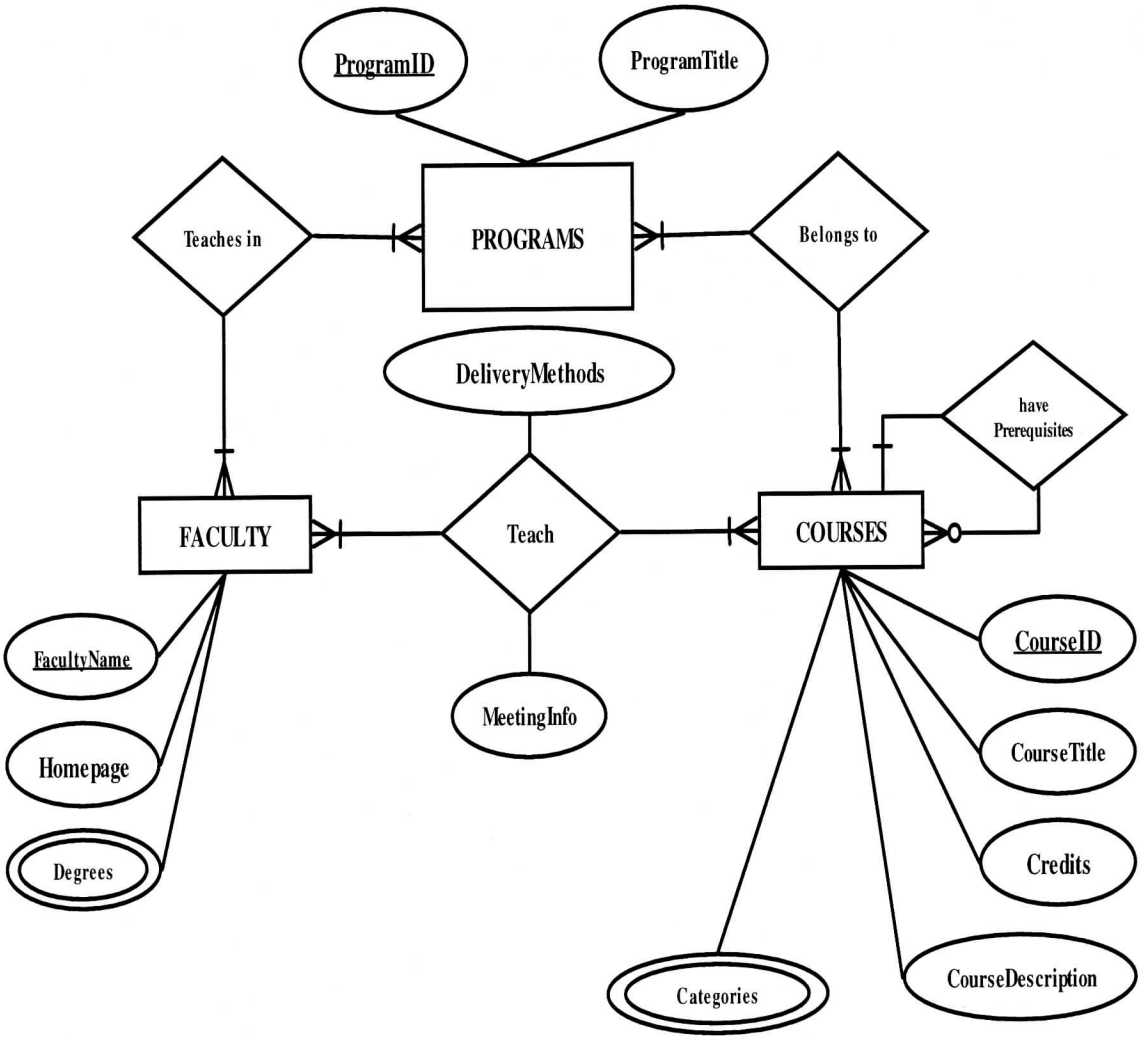


Figure 3: Entity Relationship Diagram

This ERD was submitted to a member of the project committee, Dr. Shan, at this time. He approved of the ERD so it was on to the next stage.

Normalization Process

After constructing the ERD, the process of normalization began. This involves decomposing relations with anomalies to produce smaller, well-structured relations. Normalization occurs in stages by applying rules. To create the database, relations that were in third normal form were designed.

First Normal Form	Any multi-valued attributes have been removed, so there is a single value at the intersection of each row and column of the table.
Second Normal Form	A relation in first normal form in which every nonkey attribute is fully functionally dependent on the primary key.
Third Normal Form	A relation that is in second normal form and has no transitive dependencies present.

Table 2: Normal Forms

A functional dependency is a constraint between two attributes or two sets of attributes. Partial functional dependency is a functional dependency in which one or more nonkey attributes are functionally dependent on part, but not all, of the primary key. Transitive dependencies are functional dependencies between two or more nonkey attributes. [7]

[7] Hoffer, Jeffrey A., Prescott, Mary B., McFadden, Fred R., (2002). *Modern Database Management 6th Edition*, Prentice Hall pp. 189-193

Relational Schema

After careful analysis of the data that had been picked out to place in the database, creating relational schema was the next step. This identified all of my relations, primary keys, attributes, and relationships. Because of the overlap between MSIS and MSIA courses and faculty, along with the one to many relationships and multi-valued attributes, designing the relational schema was more of an undertaking than was initially thought.

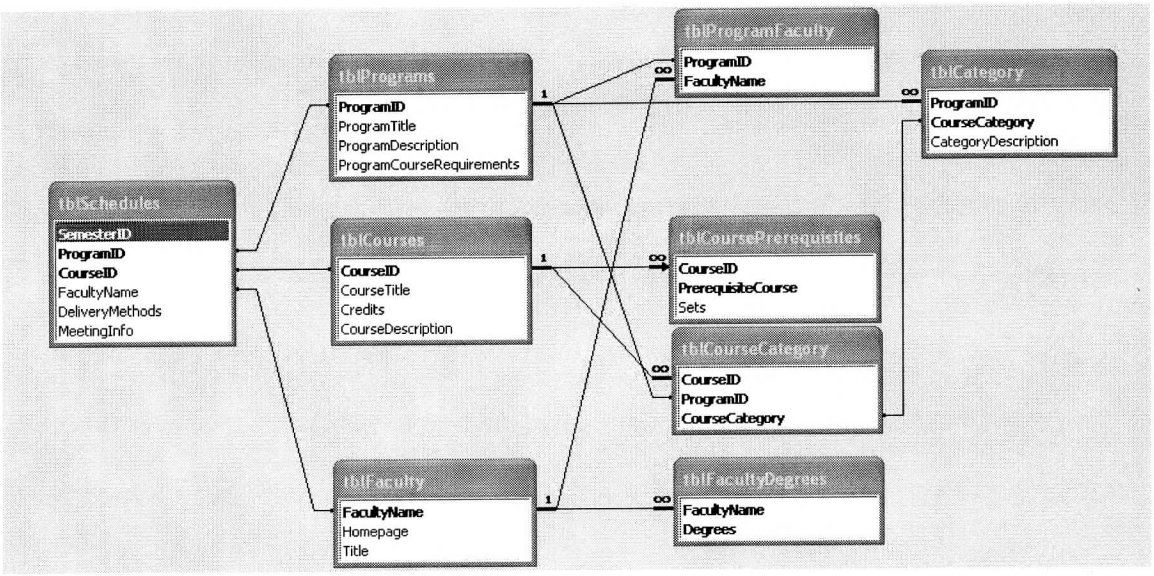


Figure 4: Relational Schema

This schema shows the relations that were created for the structured data. The primary keys are shown in bold. Because of the nature of the relationships, many of the tables have composite keys to uniquely identify each record.

Besides the tables shown in the schema, the database also contains three other tables: tblFees, tblHousing/Meal, and tblTuition. These are not shown in the ERD or relational schema because they aren't related to each other or any of the other tables in the database. They are included because these were examples of dynamic attributes

originally identified in the problem. Although all three graduate programs share the same cost information on the current website, many documents that are currently stored locally also need to access the attributes contained in these tables.

Documentation and Population

Once the tables had been created in *Microsoft Access*, documentation was added to describe the attributes contained in the tables. This was done before populating the tables in an effort to have a well-structured and well-defined database before adding the data. The following is an example of the documentation that was added to each table:

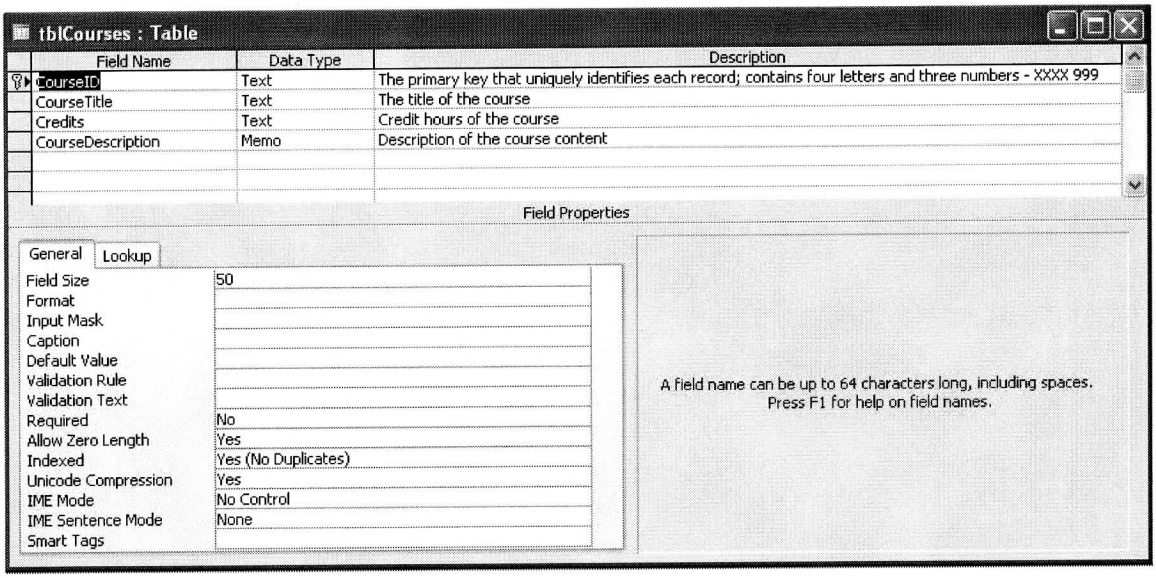


Figure 5: Table Documentation

Please see other table documentations in Appendix G.

At this point in the project, it was time to undertake the task of inserting the data into the tables that had been identified. If shortcuts were taken in the analysis and design phases, a database prototype that didn't solve the problem identified (even though it changed frequently) or meet my objectives may have been created.

The hours it would take to input the data into database were grossly underestimated. Although several of the over 600 records are short, many hours were spent getting the data in the proper fields. Different methods were used for the initial input – mainly using *Microsoft Access* forms and/or tables.

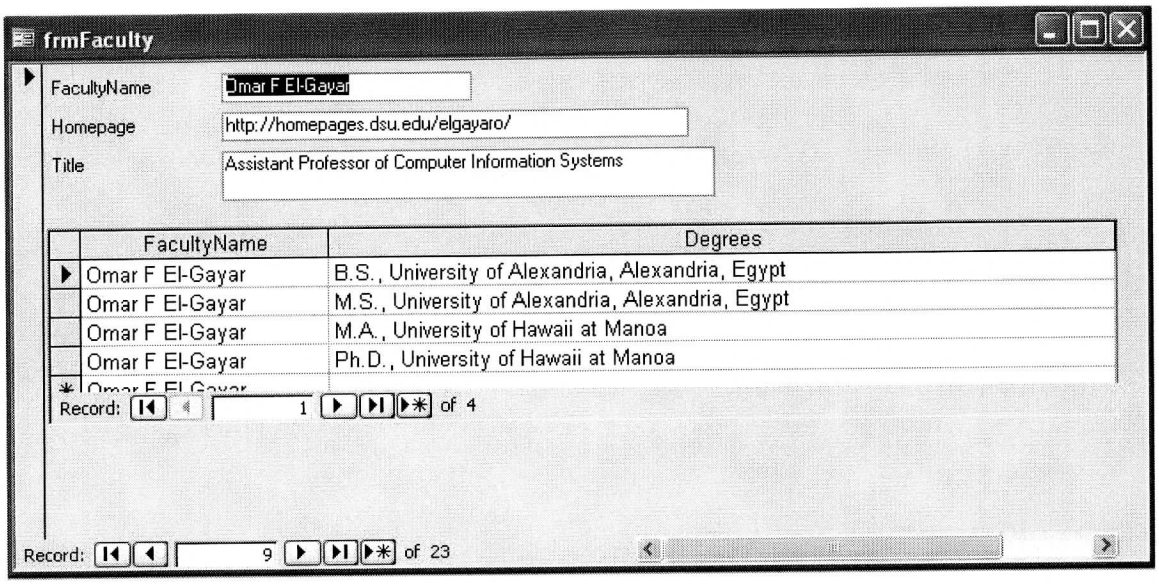


Figure 6: An Example of a Form

The preceding form was used to enter the data into tblFaculty and tblFacultyDegrees. The main form enters the data into tblFaculty, and the sub form enters the data into tblFacultyDegrees, which was created to handle the multi-valued attribute Degrees.

Creating forms for data entry purposes posed somewhat of a dilemma. It was always taken into consideration that the project was originally supposed to use *Oracle* or *Microsoft SQL Server* as my back end database. The student in me wanted to do an excellent MSIS project and considered generating a script to create and populate the tables. The realist in me remembered that this was to be used for a relatively small website that was currently updated and maintained using *Microsoft* products. If I was

doing this professionally, I would have spent a great deal of time creating super user-friendly forms. Keeping in mind the end user, it makes more sense from a business standpoint to have the user open *Microsoft FrontPage*, open the *Microsoft Access* database and manipulate the data using well built forms than it does to pay for the countless hours that would be needed to program web pages to accomplish the exact same thing. If I had more time, I would have created more forms for the data manipulation. As I begin to digress, I'll make my point – it took a long time to populate the tables.

Entering most of the data was relatively straightforward, just time consuming. There was one relation, tblSchedules, which consumed a lot of time. I knew from experience as a graduate student at DSU that Laurie Dennis was always sending out the schedules for the upcoming academic years to the students. These schedules were also available on the website. From the outset of my project, it was a goal to make her job easier. Although it isn't a possibility to know which faculty member will be teaching a course in the fall semester of 2007 (as an example), the course rotation tables show which semesters courses will be offered in. Using these tables, the data was entered for the schedules. For the schedules of courses for the upcoming academic semesters, a much more complete job was done because the faculty teaching the courses, the meeting information, and delivery methods are already available. As a benefit of this table, it's easy to envision the scenario where those schedules either exist in dynamic documents or web pages - even though they may not be as complete as the upcoming ones. As those aspects of the schedule are decided upon, they could easily be inserted into the table. Then the director of the graduate office would simply have to open a *Microsoft Word*

document that contains that data to email them to students. Better yet, a graduate student who is browsing the graduate programs' website could simply click on a link and a dynamic web page would open with the data. A single web page could display the course, title, credits, prerequisites, and all other information available by using *ASP.NET* pages to pass query parameters.

The following is a table that shows the course rotation for the MSIA program:

			FA 04	SP 05	SU 05	FA 05	SP 06	SU 06	FA 06	SP 07
Core Courses										
INFA	711	Computer Security Planning and Procedures		X			X			X
INFA	713	Managing Information Security Risk	X			X			X	
INFA	715	Privacy, Fraud and Identity Theft	X			X			X	
INFA	717	Advanced Network Security	X			X			X	
INFA	719	Advanced Software Applications Security		X			X			X
INFA	721	Computer Forensics				X			X	
INFA	723	Applied Cryptography				X			X	
INFS	724	Project and Change Management					X			X
Wireless and Mobile Security Specialization										
INFA	751	Introduction to Wireless Security		X			X			X
INFA	753	Building and Securing Wireless Networks				X			X	
Banking and Financial Security Specialization										
INFA	741	Banking Assurance				X			X	
INFA	743	IA in Financial Transactions					X		X	
INFA	745	Securing Transactions		X			X			X
Internet and E-Commerce Security Specialization										
INFS	732	Electronic Commerce				X			X	
INFS	734	Client Server Technologies					X			X
INFA	745	Securing Transactions		X			X			X
Electives										
LAW	862	Cyberlaw					X			X
CSC	692	Advanced Cryptology					X			X
INFA	770	Software Engineering Management					X			X
INFA	794	Internship			X			X		
INFA	792	Special Topics (*when offered*)								
Knowledge & Prerequisite Courses										
MATH	609	Foundational Math			X			X		
CSC	150	CS1:Principles Of Programming			X			X		
CSC	250	Computer Science II			X			X		
CSC	310	Data Structures and Algorithms			X			X		
CIS	350	Com Hrdwr Data Comm & Netwrk			X			X		

Table 3: MSIA Course Rotation

Using these course rotation tables from the three programs, the SemesterID, ProgramID, and CourseID were entered into the tblSchedules relation. This generated 258 records, but could prove to be a useful, time saving tool in the future.

Web Application Utilizing ASP.NET

Although time was becoming a factor, a web page was programmed using *ASP.NET* to give an example of how this database could be the back end to dynamic pages. A simple HTML page generated with Microsoft FrontPage was the starting point. The interface of the page is very simple:

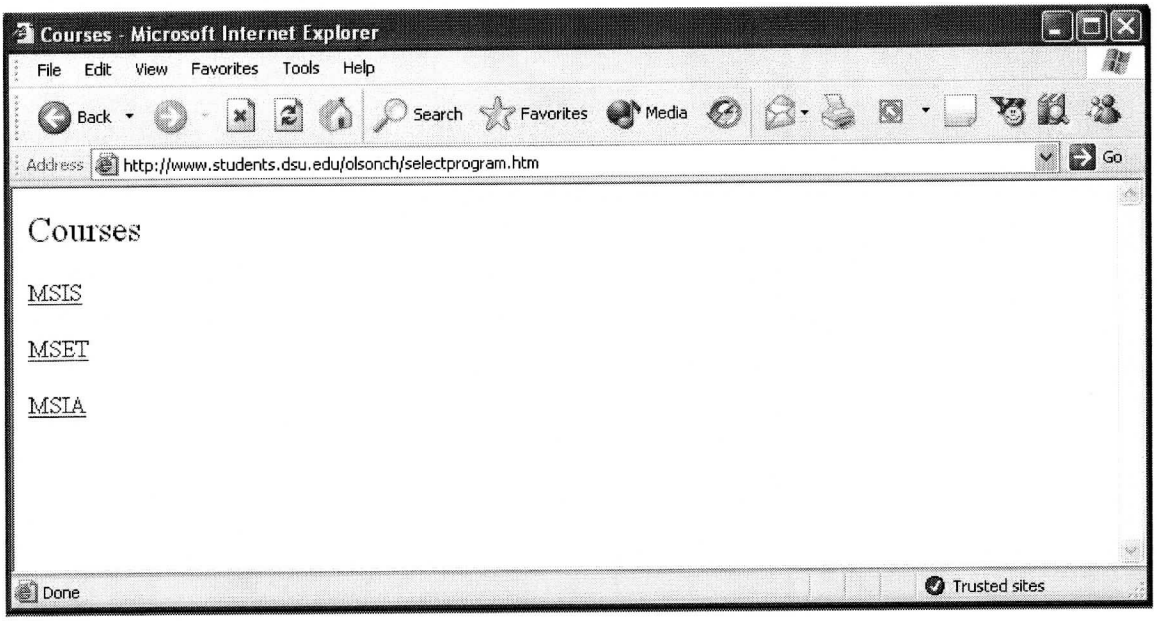


Figure 7: Selectprogram.htm

The page is very simple, but this excerpt of code shows how clicking on a hyperlink passes the value of the ProgramID to the *ASP.NET* page in order to show all of the courses in that program:

```
<p><a href="ProgramCourses.asp?programID=MSIS">MSIS</a></p>
<p><a href="ProgramCourses.asp?programID=MSET">MSET</a></p>
```

** MSIA **

All of these hyperlinks point to the same *ASP.NET* page, but the querystring (that appears after the '?' in the hyperlink) identifies the program that the user is looking to view the courses for.

If a user clicks on MSIS or MSIA, the following pages appear:

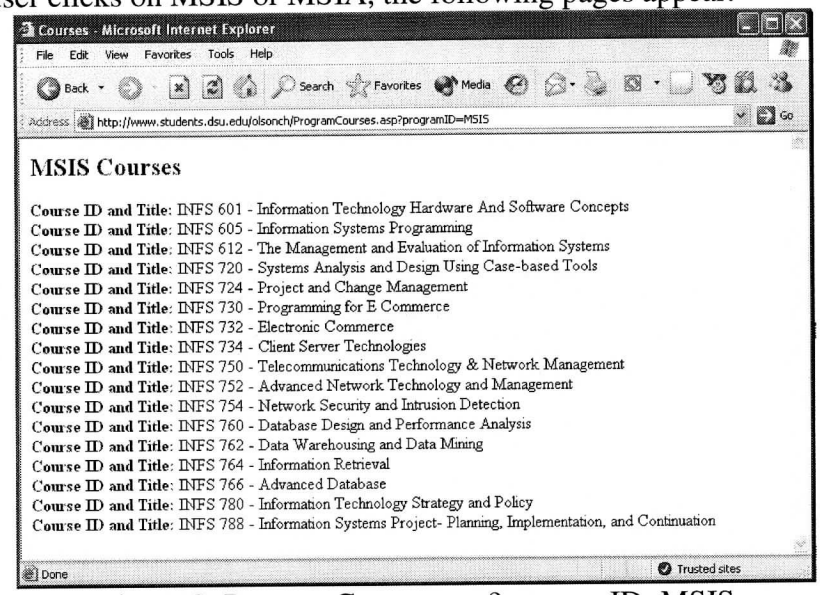


Figure 8: ProgramCourses.asp?programID=MSIS

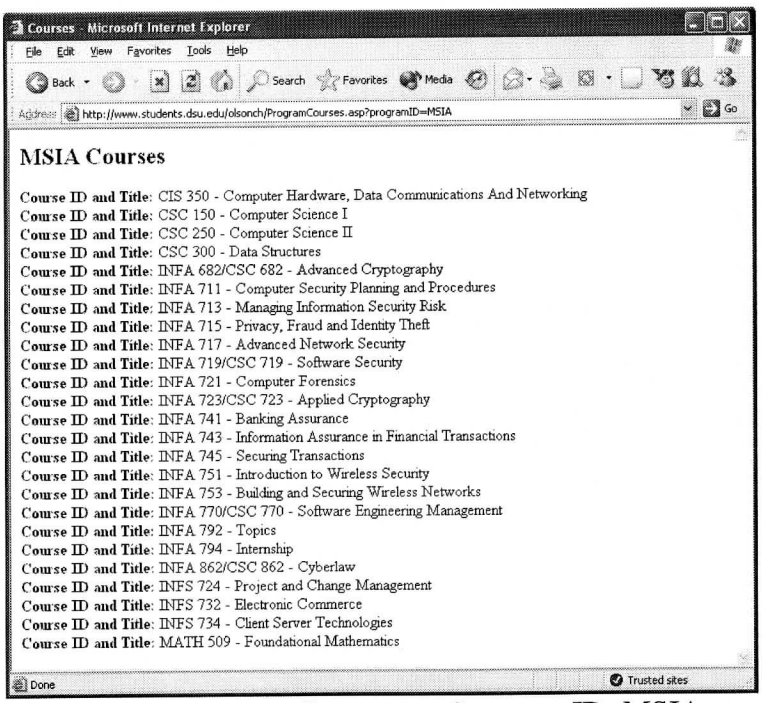


Figure 9: ProgramCourses.asp?programID=MSIA

Before the code needed to generate these pages is dissected, a few things need to be pointed out. If you look at the URL of each of the pages, you'll notice that it is the same page, but the content is different, based on the querystring. Also, if you look at Figure 9, you'll notice that some courses with the prefix INFS appear. This is because some courses are taught in both programs.

Although *Microsoft Visual Studio.NET* was used to aid with the code, the textbook *Beginning Active Server Pages 3.0*[8] was referenced to help with database connectivity.

Dim strProgram

Dim objConn, objRS, strSQL

Set objConn = Server.CreateObject("ADODB.Connection")

Set objRS = Server.CreateObject("ADODB.Recordset")

The first two lines shown declare the variables. The second two lines define the connection and the recordset.

strProgram = request.querystring("programID")

This line sets the variable strProgram to be equal to the value passed from the query string (in this case MSIS, MSIA, or MSET).

objConn.provider="Microsoft.Jet.OLEDB.4.0;"

Simply states the engine used for the connection. As a side note, there are very minimal changes needed here to connect to a *Microsoft SQL Server* database.

objConn.open Server.MapPath("./fpdb/gradoffice.mdb")

This code maps to the location where the database resides on the server.

---

[8] Buser, D. et al. (1999), *Beginning Active Server Pages 3.0*, published by Wrox Press Ltd., ISBN: 1-861003-38-2

<H2><%response.write strProgram%> Courses</H2>

If you'll notice Figures 8 and 9, not only does the content change, the above line changes the title as well.

```
strSQL = "SELECT DISTINCT tblCourseCategory.CourseID,
tblCourses.CourseTitle " _
 & "FROM tblCourses INNER JOIN tblCourseCategory " _
 & "ON tblCourses.CourseID = tblCourseCategory.CourseID " _
 & "WHERE [ProgramID]= '"&strProgram&"'"
```

This is the heart of the code that assigns the query to the variable strSQL. For simplicity's sake, only the CourseID and CourseTitle were included, but more information like the description, credits, and prerequisites could have easily been added. The 'WHERE' clause in this query is what makes the page truly dynamic; it finds the courses that match the strProgram variable (which comes from a passed value). 'DISTINCT' was also included in the SELECT statement. If this had been omitted, some of the courses would appear more than once because they appear in more than one category. As an example, INFS 732 appeared three times under the MSIS program because it belonged to: the Electronic Commerce Specialization, the General Specialization, and the Network Administration and Security Specialization. This portion could have been simplified by using *Microsoft Access*'s built-in query option and connecting directly to that, but this wasn't done to demonstrate two things – an academic understanding due to the data management specialization, and that other back end databases could be queried in the same manner using *ASP.NET*.

```
objRS.Open strSQL, objConn, 0, 1
```

This code opens the recordset and assigns the values retrieved in the SQL query to it.

```
While Not objRS.EOF
 Response.Write "<B>Course ID and Title:</B> " & objRS("CourseID") & _
 " - " & objRS("CourseTitle") & "<BR>"
 objRS.MoveNext
Wend
```

This code simply loops through the recordset and outputs the data to the page. The complete code for this can be found in Appendix F.

The logic for including this example was to show how passing values through *ASP.NET* allows different data to be displayed on the same page. The querystring was used in this example, but other methods, such as session variables, can accomplish the same thing.

Data Manipulation via the Internet

The next step was to find a method to manipulate the data via the web. *ASP.NET* was not used for three reasons:

1. There wasn't time
2. As mentioned earlier, if this were done in a real world scenario, the powerful forms that *Microsoft Access* allows users to create should be used
3. *Microsoft* provides another tool

Microsoft Data Access pages are special types of web pages designed for working with or viewing from the Internet or an intranet. They can be used with data from a *Microsoft Access* database or a *Microsoft SQL* database. These pages are connected directly to the database. Any modifications, additions, or deletions to the fields affect the underlying database.[9] The data access pages are easily created in *Microsoft Access* to be saved as web pages. The following is an example generated from a form, but they can also be generated from tables.

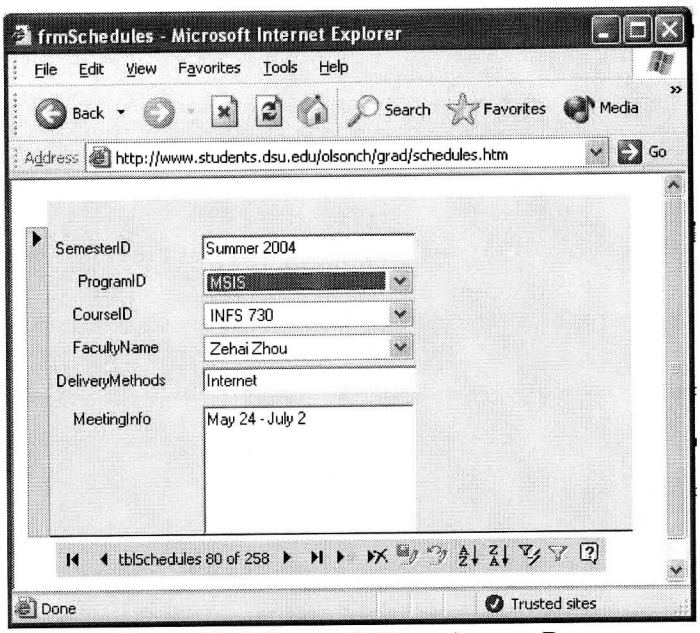


Figure 10: Microsoft Data Access Page

This example shows a data access page generated from a form that connects to tblSchedules. The toolbar at the bottom allows for record navigation, addition, and deletion. There are also buttons that allow for sorting and filtering the data, but a data access page viewed in *Microsoft Internet Explorer* is a copy of the data. The filters and

[9]http://office.microsoft.com/assistance/preview.aspx?AssetID=HP052796371033&CTT=4&Origin=CH062526501033

sorts are only applied to the view and don't affect the data, but any modifications, additions, or deletions do permanently change the data in the tables.

Results and Conclusions

The goal of creating a database that could be used to retrieve data for Dakota State University's graduate office was accomplished. The data is in place so documents and web pages can be made dynamic.

As stated earlier with the discussion on the lack of proper problem definition, this project didn't turn out as well as it was hoped to.

Factor	Rank
Lack of user input	1
Incomplete requirements and specifications	2
Changing requirements and specifications	3
Lack of executive support	4
Technology incompetence	5
Lack of resources	6
Unrealistic expectations	7
Unclear objectives	8
Unrealistic time frames	9
New Technology	10

Table 4: Factors Causing IT Project Problems[10]

The preceding table outlines the primary factors that cause problems for IT projects. Most of these factors were encountered. Because of my immobility early in the

[10] Johnson, Jim. "CHAOS: The Dollar Drain of Information Technology Project Failures," Application Development Trends (January 1995) *www.standishgroup.com/chaos.html*

semester, I blame myself for the lack of user input. Laurie Dennis was always helpful and available, and I should have been in contact with her much more. With her retirement, the personnel in the graduate office will be changing as well. It's difficult to get input from a user when you don't know who the user will be.

The requirements and specifications were incomplete from the start and went through several revisions throughout the time it was worked on. Those factors definitely caused problems, but contributed to the learning experience with IT projects. Adapting to the changes was a valuable lesson.

The last four factors listed in the table certainly also came into play. I set unrealistic expectations and objectives for myself. There is so much more I would have liked to do with this project beyond the database, but the time in which I had to work on this project became very limited with the indecision surrounding it. And finally, my lack of knowledge with the new technologies available made it hard for me to go beyond the database. The use of *XML*, *VBA*, and *ASP.NET* could provide limitless ways to effectively manipulate and retrieve the data.

A great prototype database that can be built upon and used in the future was created. There is so much more I would have liked to have accomplished, but having the data in well-structured relations is the first and most crucial step when creating a repository that is to be used for dynamic web pages and documents. I feel I did this, and did it well.

References

About Data Access Pages: Microsoft Office Online. *http://office.microsoft.com/assistance/preview.aspx?AssetID=HP052796371033&CTT=4&Origin=CH062526501033*

BC Government Information Resource Management Glossary. *http://www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm*

Buser, D. et al. (1999), *Beginning Active Server Pages 3.0*, published by Wrox Press Ltd.

Chalfin, Natalie, "Four Reasons Why Projects Fail," *PM Network,* PMI (June 1998)

Dennis, A., & Wixom, B.H., (2000). *Systems analysis and design*, New York, John Wiley & sons, inc.

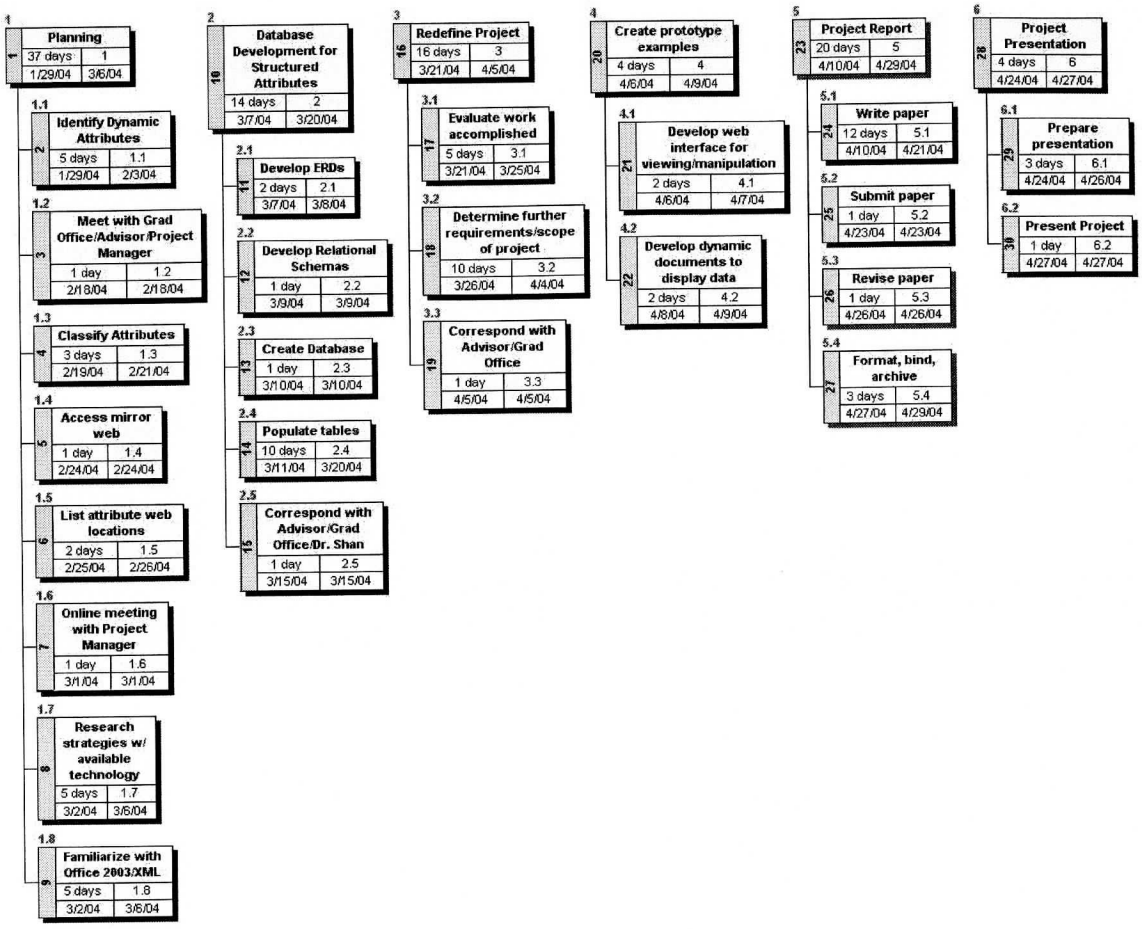
ESRI GIS and Mapping Software.*http://www.esri.com/library/glossary/e_h.html*

Hoffer, Jeffrey A., Prescott, Mary B., McFadden, Fred R., (2002). *Modern Database Management 6th Edition*, Prentice Hall

Johnson, Jim. "CHAOS: The Dollar Drain of Information Technology Project Failures," Application Development Trends (January 1995) *www.standishgroup.com/chaos.html*

Schwalbe, Kathy, (2002). *Information Technology Project Management,* Boston, Course Technology/Thomson Learning

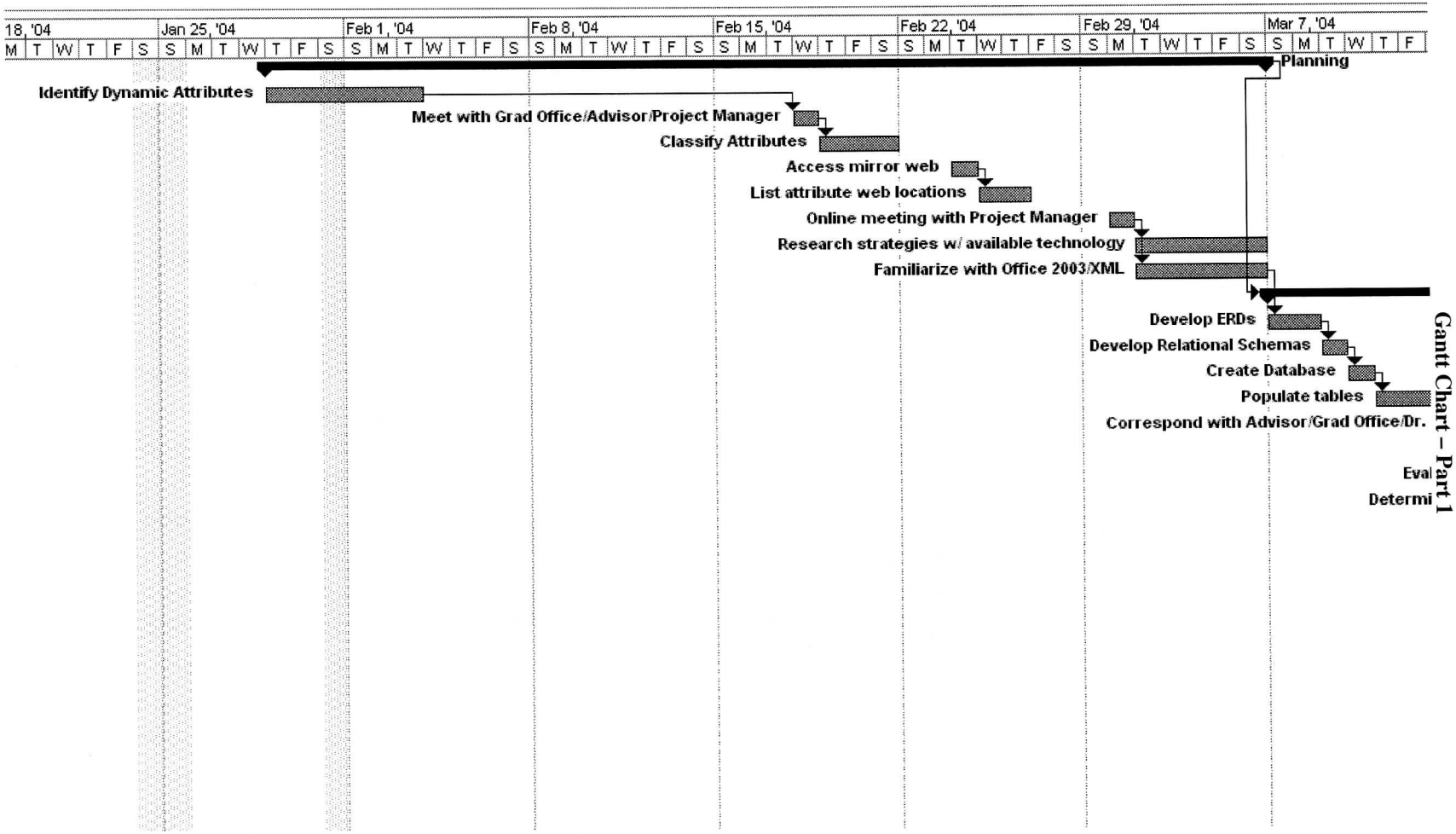
Work Break Down Structure



Gantt Chart Tasks

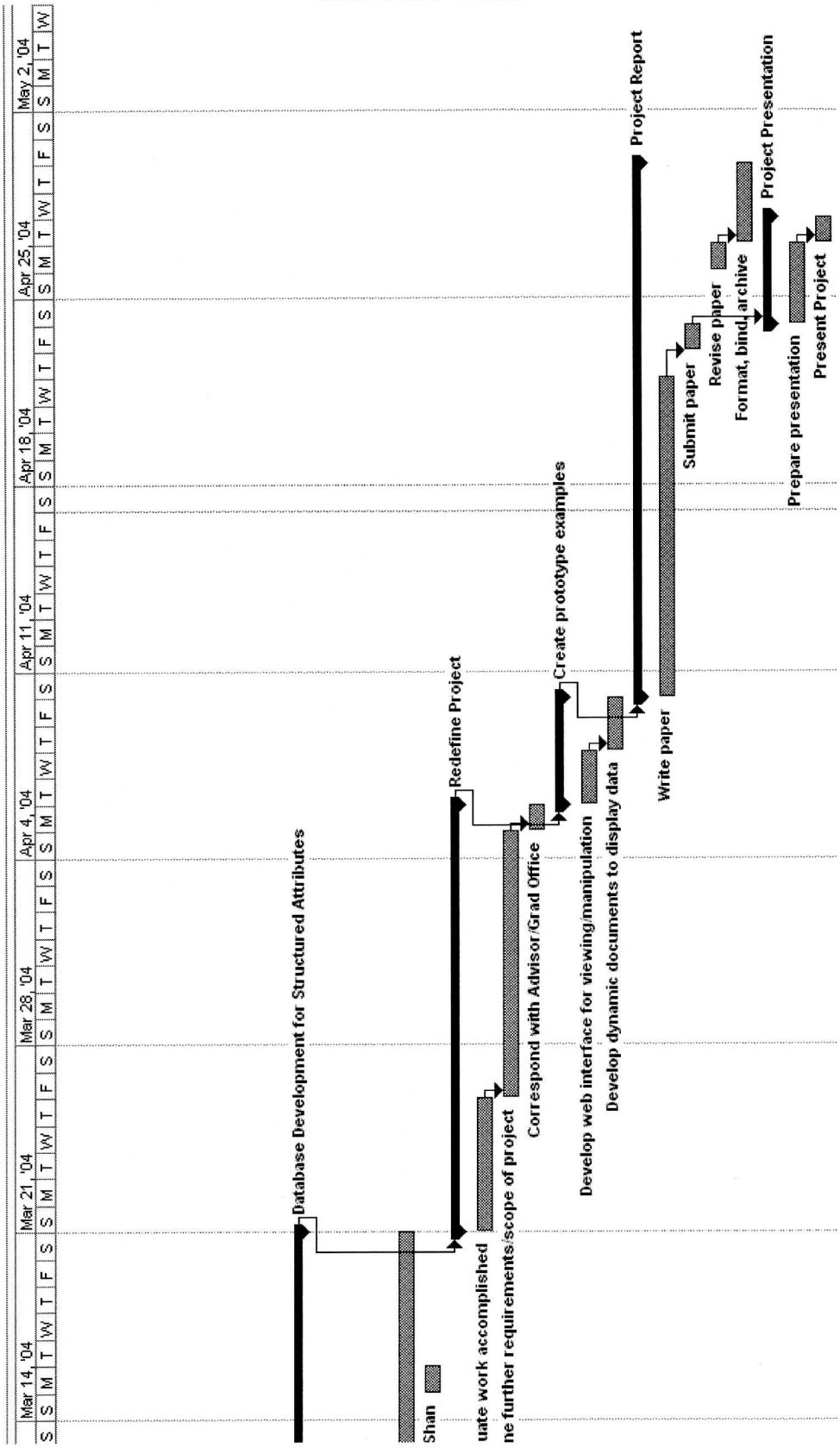
Gantt Chart

	Task Name	Duration	Start	Finish	Prede
1	**Planning**	**37 days**	**Thu 1/29/04**	**Sat 3/6/04**	
2	Identify Dynamic Attributes	5 days	Thu 1/29/04	Tue 2/3/04	
3	Meet with Grad Office/Advisor/Project Manager	1 day	Wed 2/18/04	Wed 2/18/04	2
4	Classify Attributes	3 days	Thu 2/19/04	Sat 2/21/04	3
5	Access mirror web	1 day	Tue 2/24/04	Tue 2/24/04	
6	List attribute web locations	2 days	Wed 2/25/04	Thu 2/26/04	5
7	Online meeting with Project Manager	1 day	Mon 3/1/04	Mon 3/1/04	
8	Research strategies w/ available technology	5 days	Tue 3/2/04	Sat 3/6/04	7
9	Familiarize with Office 2003/XML	5 days	Tue 3/2/04	Sat 3/6/04	7
10	**Database Development for Structured Attributes**	**14 days**	**Sun 3/7/04**	**Sat 3/20/04**	**1**
11	Develop ERDs	2 days	Sun 3/7/04	Mon 3/8/04	9
12	Develop Relational Schemas	1 day	Tue 3/9/04	Tue 3/9/04	11
13	Create Database	1 day	Wed 3/10/04	Wed 3/10/04	12
14	Populate tables	10 days	Thu 3/11/04	Sat 3/20/04	13
15	Correspond with Advisor/Grad Office/Dr. Shan	1 day	Mon 3/15/04	Mon 3/15/04	
16	**Redefine Project**	**16 days**	**Sun 3/21/04**	**Mon 4/5/04**	**10**
17	Evaluate work accomplished	5 days	Sun 3/21/04	Thu 3/25/04	
18	Determine further requirements/scope of project	10 days	Fri 3/26/04	Sun 4/4/04	17
19	Correspond with Advisor/Grad Office	1 day	Mon 4/5/04	Mon 4/5/04	18
20	**Create prototype examples**	**4 days**	**Tue 4/6/04**	**Fri 4/9/04**	**16**
21	Develop web interface for viewing/manipulation	2 days	Tue 4/6/04	Wed 4/7/04	
22	Develop dynamic documents to display data	2 days	Thu 4/8/04	Fri 4/9/04	21
23	**Project Report**	**20 days**	**Sat 4/10/04**	**Thu 4/29/04**	**20**
24	Write paper	12 days	Sat 4/10/04	Wed 4/21/04	
25	Submit paper	1 day	Fri 4/23/04	Fri 4/23/04	24
26	Revise paper	1 day	Mon 4/26/04	Mon 4/26/04	
27	Format, bind, archive	3 days	Tue 4/27/04	Thu 4/29/04	26
28	**Project Presentation**	**4 days**	**Sat 4/24/04**	**Tue 4/27/04**	**25**
29	Prepare presentation	3 days	Sat 4/24/04	Mon 4/26/04	
30	Present Project	1 day	Tue 4/27/04	Tue 4/27/04	29



Gantt Chart – Part 1

Gantt Chart – Part 2



D

Selectprogram.htm

```
<html>

<head>
<meta http-equiv="Content-Language" content="en-us">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Courses</title>
</head>

<body>

<p><font size="5">Courses</font></p>
<p><a href="ProgramCourses.asp?programID=MSIS">MSIS</a></p>
<p><a href="ProgramCourses.asp?programID=MSET">MSET</a></p>
<p><a href="ProgramCourses.asp?programID=MSIA">MSIA</a></p>

</body>

</html>
```

ProgramCourses.asp

```
<%Option Explicit%>
<HTML>
<!--
Created by Chris Olson
Displays Course Numbers and Titles for specific programs depending on which link the
user clicks
-->
<%
Dim strProgram                                    'declares variable to pass the
Program value
Dim objConn, objRS, strSQL                  'declares connection and recordset variables
Set objConn = Server.CreateObject("ADODB.Connection")
Set objRS = Server.CreateObject("ADODB.Recordset")

strProgram = request.querystring("programID")  'passes the value from the link to the
variable

objConn.provider="Microsoft.Jet.OLEDB.4.0;"
objConn.open Server.MapPath("./fpdb/gradoffice.mdb") 'navigates to the location of the
database
%>
<HEAD><TITLE>Courses</TITLE></HEAD><BODY>
<H2><%response.write strProgram%> Courses</H2>
<%
'queries the course data from the Course and CourseCategory tables
strSQL = "SELECT DISTINCT tblCourseCategory.CourseID, tblCourses.CourseTitle " _
  & "FROM tblCourses INNER JOIN tblCourseCategory " _
  & "ON tblCourses.CourseID = tblCourseCategory.CourseID " _
  & "WHERE [ProgramID]= '"&strProgram&"'"
objRS.Open strSQL, objConn, 0, 1

'cycles through the records in the recordset and outputs the data
While Not objRS.EOF
  Response.Write "<B>Course ID and Title:</B> " & objRS("CourseID") & _
  " - " & objRS("CourseTitle") & "<BR>"
  objRS.MoveNext
Wend

objRS.Close
objConn.Close
Set objRS = Nothing
Set objConn = Nothing
%>
</BODY></HTML>
```

Table Documentation

tblCategory Documentation

tblCategory : Table

Field Name	Data Type	Description
ProgramID	Text	Part of the composite primary key that identifies the program
CourseCategory	Text	Part of the composite primary key that identifies the category within the program
CategoryDescription	Memo	Description of the category

tblCourseCategory Documentation

tblCourseCategory : Table

Field Name	Data Type	Description
CourseID	Text	Part of the composite primary key that identifies the course
ProgramID	Text	Part of the composite primary key that identifies the program
CourseCategory	Text	Part of the composite primary key that identifies the category within the program; a course may belong to one or many programs and one or man

tblCoursePrerequisites Documentation

tblCoursePrerequisites : Table

Field Name	Data Type	Description
CourseID	Text	Part of the composite primary key that identifies the course
PrerequisiteCourse	Text	Part of the composite primary key that identifies the prerequisites; a composite key is needed due to the multi valued attribute
Sets	Number	Needed to deal with the and/or values present in some course prerequisites

tblFaculty Documentation

tblFaculty : Table

Field Name	Data Type	Description
FacultyName	Text	Serves as the primary key to uniquely identify each record
Homepage	Text	The URL of the faculty's web page
Title	Text	The title the faculty member holds

tblFacultyDegrees Documentation

tblFacultyDegrees : Table

Field Name	Data Type	Description
FacultyName	Text	Part of the composite primary key that identifies the faculty member
Degrees	Text	Part of the composite primary key that identifies the degree held by the faculty member; composite key needed for the multi-valued attribute

tblFees Documentation

tblFees : Table

Field Name	Data Type	Description
FeeType	Text	Primary key that describes the fee and uniquely identifies each record
Fee	Currency	The price of the fee

tblHousing/Meal Documentation

tblHousing/Meal : Table

Field Name	Data Type	Description
Housing/Meal Type	Text	Primary key that uniquely identifies each record
Cost	Currency	Cost of each

tblProgramFaculty Documentation

tblProgramFaculty : Table

Field Name	Data Type	Description
ProgramID	Text	Part of the composite primary key that identifies the program
FacultyName	Text	Part of the composite primary key that identifies the faculty; composite needed because of the one to many relationship

tblPrograms Documentation

tblPrograms : Table

Field Name	Data Type	Description
ProgramID	Text	Primary key that is a four letter acronym to uniquely identify each record
ProgramTitle	Text	The full title of the program
ProgramDescription	Memo	Describes what the programs involve
ProgramCourseRequirements	Memo	Requirements for the program courses

tblSchedules Documentation

tblSchedules : Table

Field Name	Data Type	Description
SemesterID	Text	Part of the composite primary key that identifies the semester
ProgramID	Text	Part of the composite primary key that identifies the program
CourseID	Text	Part of the composite primary key that identifies the course
FacultyName	Text	The faculty that will be teaching the course for the semester
DeliveryMethods	Text	Method of delivering the course
MeetingInfo	Text	Contains information like the days, dates, times, and rooms

tblTuition Documentation

tblTuition : Table

Field Name	Data Type	Description
TuitionType	Text	Describes the tuition rate based on different factors and uniquely identify each record
Price	Text	Price of each tuition cayegory