

Graduate Student Database Project

Nicholas Wallen
Department of Computer Science
Florida State University

Major Professor: Dr. David Whalley

In partial fulfillment of the requirements for the Degree of Master of Science

Introduction

Background

In the Computer Science Department at Florida State University, tracking a student's progress through the graduate program has been handled separately and disjointly by the different staff members based on what they oversee. However, as multiple aspects of the students data must be shared for various tasks, this data is often shared via email and stored redundantly in excel spreadsheets across the department. This manner of storage often leads to hardships with regards to how the data was managed, updated, and shared; when data needs to be updated, all copies must be updated manually with great care so that out of date information is not kept and recirculated through the system. In addition, various governmental and provisional bodies request periodic surveys to be done on the makeup of the student body. Currently, this is done by sifting through hundreds of papers for all the students and counting them manually. With a centralized system, it could potentially take a single query to count the students on a moments notice.

Goal

The objective of this master's project is to create a database to centrally handle the information of all the graduate students in the Computer Science Department, and to provide access to this information with an easy to use web-based interface that can be accessed by any device with basic html rendering capabilities.

Requirements

Requirements for the system fall into three categories, those tending towards the usability of the system, those towards the maintenance and alteration of the system, and those towards the security of the system. For the first requirement, accessibility was addressed by making the system accessible

from the web via a standard web browser, and no required extensions, such as java, javascript, or flash. The system was also designed so that the users would be able to complete the repeatable tasks in a streamlined manner to cut down on wasted time, and in a concise way to switch between tasks. To address the maintenance of the system, a modular design was used. This was done so that bugs can easily be found and additional features can easily be added to the system. To address the security of the system, users are required to run sessions over Hypertext Transfer Protocol over Secure Socket Layer, https.

Functional Description

Method of Use

There are multiple users for this system defined by their role. First is the Director of Graduate Studies. It is this person's job to initiate the student into the system by using the accept function. Using this functionality, when a student arrives at the university, the director accesses the list of accepted students and creates an entry for the new student into the graduate student database. This is done by changing the status under "attending" to either "yes" or "no" and hitting the submit button. The list of students can be filtered by year, semester, and whether the sought degree is MS or PhD.

The screenshot shows a web browser window displaying the 'Computer Science Graduate Student Database'. The page has a yellow background and a sidebar on the left with various navigation links. The main content area contains a form for accepting students and a table of student records.

webmast's links

- View Info on Students (viewall)
- Accept a student into the program (accept)
- Detailed Student View (viewdetail)
- Alter Student Information (alterdetail)
- Add Funding Source (fundingsource)
- Add a TA (addta)
- View System Status (status)
- Modify the faculty member list (modifyfaculty)
- Defence Detail of a student (defencedetail)
- Add a RA (addra)
- Department Course

Action accept
Subsection

select firstName, lastName, status, joined, studentID, degreeSought from GradApplicants2007 where studentID>=0 and term='Spring' and degreeSought='PhD' and joined='Unknown' and (status='Admit - No Assistantship' or status='Admit - TA/RA') order by lastName

Admit Semester Spring
Attending Year
Degree Level PhD

submit

Student	Degree	Admitted	Attending	
Connor, Michael	PhD	Admit - TA/RA	Unknown	92
Flood, Randy	PhD	Admit - TA/RA	Unknown	44
Gavin, Peter	PhD	Admit - TA/RA	Unknown	91
Gower-Winter, Kyle	PhD	Admit - TA/RA	Unknown	90
Krishnan, Mukund	PhD	Admit - No Assistantship	Unknown	745
Roy, Arjun Guha	PhD	Admit - TA/RA	Unknown	730
Subedi, Mahesh	PhD	Admit - No Assistantship	Unknown	751

submit

Fig. 1: Accept Function

The next job of the supervisor is to initialize the data when the student meets with the director to discuss the degree program requirements. At this time the director records into the system the class prerequisites, and advisor status for the student. Later as the student progresses through the program, the director can set the faculty members that are on the student's defense committee if the student intends to defend a dissertation, project, or thesis. To edit a student's information in the system, the supervisor selects the "Alter student information" button on the left panel, and selects the student to edit (see Fig 2). The students can be queried based on their status in the system. The choices are "attending" for ordinary students, "uninitialized" for students new to the system, and "graduated", "left", and "expelled" for students no longer in the department. Selecting the student to edit will take the director to the page to alter that student's data (Fig 3).

File Edit View History Bookmarks Tools Help

Computer Science Graduate Student Database

webmast's links

View Info on Students

Accept a student into the program

Detailed Student View

Alter Student Information

Add Funding Source

Add a TA

View System Status

Modify the faculty member list

Defence Detail of a student

Add a RA

Department Course

Action	viewdetail
Subsection	

Progress Status	Attending
==	?
==	?

The following students meet this criteria:

Student	Degree	Status	Select
Achury, Monika	MS-cb	Attending	653
Angara, Sriharsha	MS-cb	Attending	489
Ayers, Kenneth	MS-cb	Attending	86
Boindala, Ajay	MS-pr	Attending	688
Brailsford, Frank	MS-th	Attending	537
Cabrera, Alejandro	PhD	Attending	102
Cedeno, Vanessa	MS-cb	Attending	112
Chang, Daniel	PhD	Attending	703
Chansarkar, Anupam	MS-cb	Attending	158
Chatmon, Christy	PhD	Attending	704
Chatterjee, Samidh	PhD	Attending	596
Chauhan, Saransh	MS-cb	Attending	612

Fig. 2: Alter/View Detail Function – Student Select Page

File Edit View History Bookmarks Tools Help

Computer Science Graduate Student Database

webmast's links

View Info on Students

Accept a student into the program

Detailed Student View

Alter Student Information

Add Funding Source

Add a TA

View System Status

Modify the faculty member list

Defence Detail of a student

Add a RA


Department Course Information

Change Password

Graduation CheckList

Advising Defaulter List

Action	viewdetail
Subsection	1



Information about student:

CS Username	
Degree Status	Attending
Degree Type	CS
Degree Level	MS-cb
Degree Time	Full Time
Semester Student Began Pursuing Degree	Summer
Year Student Began Pursuing Degree	2007
Student Office	RF2 294
Post Degree Employment Information	
Advisor	Sudhir Aggarwal
Co-Advisor	Grad Supervisor
Committee member 1	Not Set
Committee member 2	Not Set
Committee member 3	Not Set
Committee member 4	Not Set
Chairs Representative	Not Set
Representative at Large	
Prerequisite: (cop4530) Data Structures	true
Prerequisite: (cop4531) Algorithms	true
Prerequisite: (cda3100) Computer Organization I	true
Prerequisite: (cda3101) Computer Organization II	true
Prerequisite: (cop4610) Operating Systems	true
Prerequisite: (cot4420) Theory of Computation	true
Prerequisite: (mac2311) Calculus I	true
Prerequisite: (mac2312) Calculus II	true
Prerequisite: (mad2104) Discrete Math I	true
Prerequisite: (mad3105) Discrete Math II	true
Prerequisite: (sta4442) Introduction To Probability	true
Comment	
GPA	
GRE	
TOeFUL	

Done

Fig. 3: Alter/View Detail Function – Student Detail Page

The final job of the director is to indicate that the student is no longer an active student upon the student's graduation or otherwise departure from the school. This is done by again going into the “modify permanent info” form and selecting “degree status” and changing it to either expelled, left, or

graduated.

The next set of users of the system is the RA manager and TA manager. It is their job to record the semesterly information regarding the students' position and their related pay. For first time students, the data must be initialized in the semester data table. To do this, the manager clicks on the "Add a TA" or the "Add a RA" button on the navigation panel, and then chooses "enter a single student" and then fills out the appropriate data for that student. Recurring students may be entered at the same time by instead choosing "enter students in batch mode," which copies a subset of the students from the previous semester to the current one. Finally, the managers can edit the information with the TA/RA matrix manager.

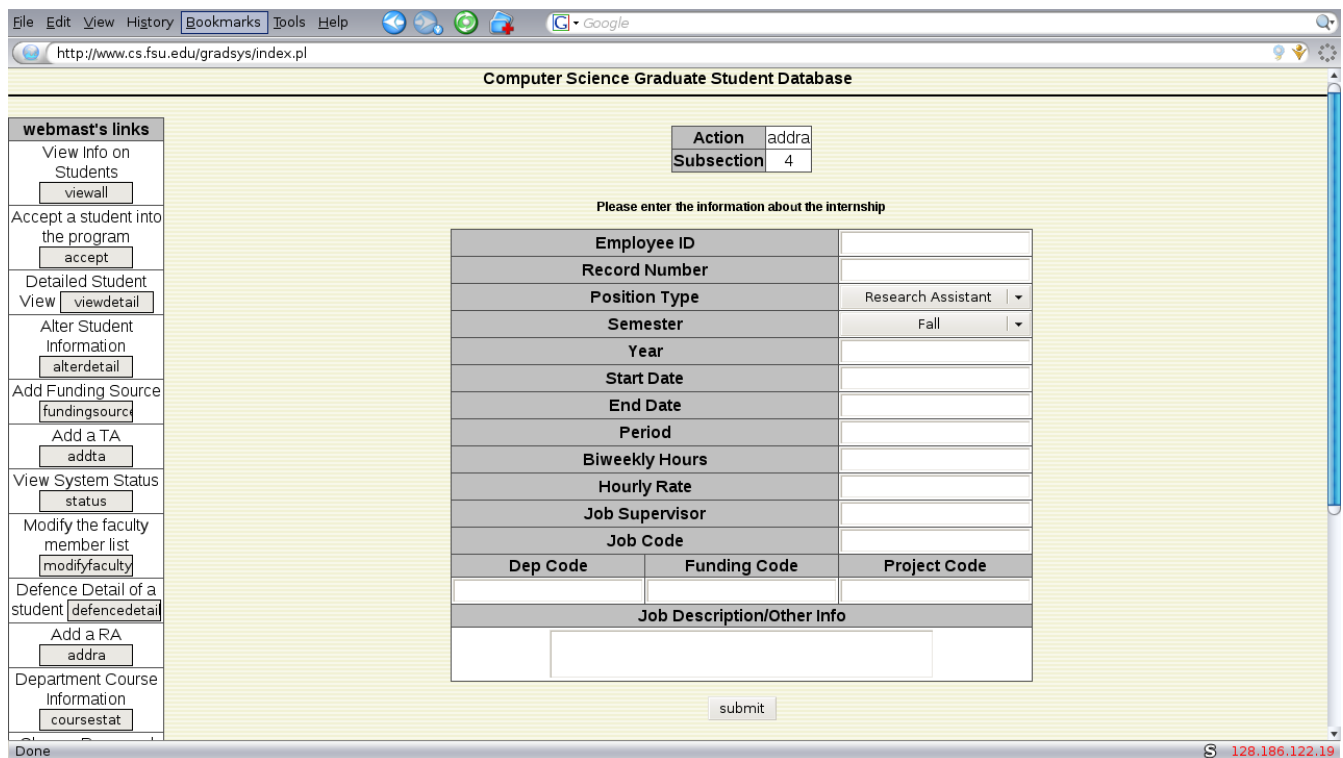


Fig 4 Add Single TA/RA

The person in charge of payroll also has access to the system to view the costs associated with employing the TAs and RAs. The payroll manager has read access to the TA/RA matrix that the RA

and TA managers use to input the financial information. The names of the students were omitted from this viewing of the matrix.

Employee Info		Pay period		Funding Code			Wage			Pay			Job Information		
Employee ID	Record	Start Date	End Date	Dep Code	Fund Code	Proj Code	Hourly	Weekly	Periods	Fringe	Biweekly	Period	Position	Supervisor	Description
68151	4	2008-12-01	2008-12-22	80003	520	19674	16.00	20.00	9.40	\$6.016	\$640	\$6016	RA	Aggarwal	research
68151	10	2008-08-08	2008-11-28	80003	520	19674	16.00	20.00	7.32	\$4.6848	\$640	\$4684.8	RA	Aggarwal	reappt
76651	5	2008-12-01	2008-12-22	80003	520	19674	16.00	20.00		\$0	\$640	\$0	RA	Aggarwal	
76651	11	2008-08-08	2008-11-28	80003	520	19674	16.00	20.00		\$0	\$640	\$0	RA	Aggarwal	reappt
77699	12	2008-08-08	2008-11-28	80003	520	19674	17.00	20.00		\$0	\$680	\$0	RA	Aggarwal	reappt
77699	6	2008-12-01	2008-12-22	80003	520	19674	17.00	20.00		\$0	\$680	\$0	RA	Aggarwal	
56663	13	2008-08-08	2008-11-28	80003	520	19674	18.00	20.00		\$0	\$720	\$0	RA	Aggarwal	reappt
56663	7	2008-12-01	2008-12-22	80003	520	19674	18.00	20.00		\$0	\$720	\$0	RA	Aggarwal	
75862	14	2008-08-08	2008-11-28	80003	520	19674	18.00	20.00		\$0	\$720	\$0	RA	Aggarwal	reappt
75862	8	2008-12-01	2008-12-22	80003	520	19674	18.00	20.00		\$0	\$720	\$0	RA	Aggarwal	
76877	15	2008-08-08	2008-11-28	80003	520	19674	16.00	20.00		\$0	\$640	\$0	RA	Aggarwal	reappt
76877	9	2008-12-01	2008-12-22	80003	520	19674	16.00	20.00		\$0	\$640	\$0	RA	Aggarwal	
63658	16	2008-08-08	2008-09-04	80003	520	16247	18.00	20.00		\$0	\$720	\$0	RA	Baker	

Fig 5 TA/RA Matrix Manager

Implementation

The system was written to work with a mysql database back-end to store the data, and is written in Perl to create the pages to serve via apache. The content of the database is divided into four major components. The first is the applicant information that the student submits when requesting admission into the program. This table is preexisting before the creation of this system, and was adapted for its new use. This table is used to look up the student ID, name and standardized testing scores. The second table used is for data that is static for most of the student's time in the department. This table holds the following data:

Field	Type
id	int(11)
username	varchar(20)
link	varchar(100)
degreeStatus	enum('Awaiting', 'Attending', 'Graduated', 'Left', 'Expelled')
degreeType	enum('CS', 'SE', 'IS')
degreeLevel	enum('MS-cb', 'MS-pr', 'MS-th', 'PhD')
degreeTime	enum('Full Time', 'Part time')
startSem	enum('Fall', 'Spring', 'Summer')
startYear	int(11)
room	varchar(50)
postDegreeEmp	varchar(30)
MajorProf	int(11)
coMajorProf	int(11)
committee1	int(11)
committee2	int(11)
committee3	int(11)
committee4	int(11)
committee5	int(11)
committee6	varchar(100)
cop4530	tinyint(1)
cop4531	tinyint(1)
cda3100	tinyint(1)
cda3101	tinyint(1)
cop4610	tinyint(1)
cot4420	tinyint(1)
mac2311	tinyint(1)
mac2312	tinyint(1)
mad2104	tinyint(1)
mad3105	tinyint(1)
sta4442	tinyint(1)
comment	text

31 rows in set (0.00 sec)

Fig. 6 Permanent Info Table

The id is the student ID that links the permanent information to the students entry in the applicant database. The username is the student's username and email address, and link is a link to their picture. DegreeStatus describes whether the student has been initialized, has left the program, or is a currently attending student. DegreeType describes which degree the student is pursuing: computer science, software engineering, or information security. DegreeLevel represents whether the student is pursuing a PhD, or a course, project, or thesis based MS degree. StartSem and startYear comprise the semester and year the student enters the system. Room is the student's office room number. PostDegreeEmp lists the students employment after leaving the system.

The third table used by the system is used to track the semester to semester information of the student's information through the degree program. This table also uses an ID field for referencing a

student, and a semester and year field for the semester and year this entry tracks. startDate and endDate mark the dates by which the appointment begin and end. FundingType describes the type of position the student has and can be one of the following: 'TA – MA' for a student teaching a class in the CS major, 'TA -NM' for students teaching a non-majors class, 'TA – LI' for those teaching a literacy class, 'TA -OR' for any other teaching assignment, 'RA' for research assistants, 'SG' for those in the systems group, and 'OR' for all other job positions. Jobcode, projectcode, fundingcode, record, and employeeID all represent codes needed by the school to describe the student's job position internally. Jdescription describes the job and jSupervisor lists the staff or faculty member that the student reports to for the job. Hours lists the number of hours the student works weekly. Teach is a flag that is set if the student is in a teaching position.

Field	Type
id	int(11)
semester	enum('fall', 'spring', 'summer')
year	int(4)
startDate	date
endDate	date
period	decimal(3,2)
fundingType	enum('TA - MA', 'TA - NM', 'TA - LI', 'TA - OR', 'RA', 'SS', 'OR', 'SG')
jobCode	varchar(20)
deptCode	int(11)
projectCode	mediumint(8) unsigned
fundingCode	smallint(5) unsigned
hourlyWage	decimal(3,2)
jDescription	varchar(20)
jSupervisor	varchar(20)
hours	decimal(5,2)
teach	tinyint(1)
employeeID	int(11)
record	int(4)
assignment	varchar(100)
preference	varchar(100)

20 rows in set (0.01 sec)

Fig 7 Semester Information Table

Three other tables are used by the database to manage users and their privileges. GradUsers, GradPages, and GradPrivileges (see Fig 8). GradUsers contains the username and the password for the

account. GradPages contains pageID, used to uniquely identify the page, pageLink, a link that the script uses to find the correct section of code for that page, and 'pageDesc' which is a description for the end users to know what that page is for. GradPrivileges describes user privileges by mapping a user's username to the pageID's of the pages he has access to.

```
File Edit View Terminal Tabs Help
mysql> desc GradUsers;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20)   |      | PRI |          |       |
| passwd| varchar(20)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> desc GradPrivileges;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name  | varchar(20)   | YES  | MUL | NULL    |       |
| pageID| int(4)        | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> desc GradPages;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| pageID| int(4)        |      | PRI | NULL    | auto_increment |
| pageLink| varchar(100)  | YES  |     | NULL    |               |
| pageDesc| varchar(200)  | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Fig 8 User Privileges Tables

Sessions

User sessions are managed by the Perl module `cgi:sessions`. `cgi:sessions` provides two ways to track users of the web page. The first is through a cookie management system. This management system was not chosen for this project so that users without a cookie supporting browser could use the system. The second way of detecting users is by calling `$session->id()`, which returns the id of the session, and embedding the value in the web page. When a related web page is submitted to the webserver, if it has a session id, the perl script connects to the database server, looks up the id, and if it

exists, loads the saved data into the session variable. If the page does not send a session id, the script generates one for it, and if it sends a faulty id, it returns the user to the login screen.

HTML

Web page design was done with the aid of the perl module `html:template`. This allows for the html to be written in a template file separate from the perl code to increase code readability and reusability. When a template is called, the script loads data from the template and substitutes any perl variables in the template with the corresponding variable in the script. Many different templates were designed for the various pages, as well as a generic template for pages with less content. All templates load two frames: 'content', for the specific content of the page, and 'noncontent' for items standard to each page such as the links to the different section and the logout button.

Future Work

While this project has supplied the basics for a system to keep track of students, many further enhancements are desired for greater control of the information.

The first major enhancement would be for advisors to be able to log in and see the information for their students. They should be able to see what prerequisites the students have taken, and still need to take, as well as information about their job, pay, room assignment, and email address. The advisors should also be able to fill out semester and yearly progress reports on phd students who have advanced to candidacy. Also useful would be the ability to record which classes have been recommended by the student's advisor, as well as taken classes and grades made in said classes.

Another useful addition to the system would be an extended system by which a student should exit the system. Currently, the Grad Supervisor simply marks 'graduated', 'left', or 'expelled' in a

students status when the student leaves; instead, a system could be set that allows the supervisor to simply click a button and have the system automatically check which graduation requirements have been met, which are currently being worked on, and which are unmet.

Other improvements to the system should relate to the maintenance of the system. There should be some method for the users of the system to correct mistakes made by themselves that do not require them to contact the webmaster to modify the mysql tables to fix the problem. Similarly, user account generation, password management, and course and advisor status should be modifiable from within the user interface for privileged users.

Another way the system could be improved upon is with the matrix manager. Currently, it requires the user to type in the values for each of the fields that are in the matrix. The burden of typing in the users could be eased by creating pull down menus for enumerated values and booleans.

A prototype for a more visually appealing and intuitive template was worked on and implemented for the matrix manager. This new, more user friendly design could be propagated throughout the system.

Finally, a general query capability should be added. This feature will be very useful when attempting to fill out surveys regarding our graduate students, to answer questions from administrators, or to obtain information for various reports.

Conclusions

In conclusion, a database is a far more efficient mechanism to store and organize data than spreadsheets; it allows for a centralized facility that can easily be modified and quickly shared among multiple users. Having a web based front end removes the requirement of users having to understand and use a database directly, and allows users to connect from anywhere with an internet connection and

a basic web browser. It also allows the possibility of queries to obtain information for various surveys. Due to the number of users reading and modifying student data in the department, it is an ideal use for such a system.