1998

Student database access from the web

Prashanthi Sundaram

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

Part of the Digital Communications and Networking Commons

Recommended Citation

http://scholarworks.lib.csusb.edu/etd-project/1785

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

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

by

Prashanthi Sundaram

September 1998
STUDENT DATABASE ACCESS FROM THE WEB

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Prashanthish Sundaram

September 1998

Approved by:

Dr. Josephine Mendoza, Project Advisor, Computer Science

9/3/98

Date

Dr. Tong Yu

8/18/98

Date

Dr. George Georgiou, Graduate Program Coordinator

8/18/98

Date
Abstract

In most of the commercial and non-commercial enterprises the need to maintain and access data is essential. In addition, accessing the data must be easily accessible over the network. One such example of is a database that stores student information and that can be accessible via the Internet. Database Access through the Web (DAW), implements a database to store academic and general information of graduate and undergraduate students in the department of Computer Science, CSUSB and provides access to the database from the web.

The student information that are being stored include SSN, name, GPA, course taken etc. and other general information. The main users of DAW are faculty and department staff. Depending upon the access privilege of a DAW user, different sets of operations are permissible.

DAW stores the data in a Postgres SQL database and provides an intuitive and easy to use graphical user interface to access, edit, update and administrate student information. The GUI is written using HTML 3.0 and Common Gateway Interface (CGI) using C. Postgres SQL is accessed through Open Data Base Connectivity (ODBC) from the CGI programs. DAW architecture is modular, extensible. It uses standard components so that it can easily be ported across many systems.
Acknowledgements

I would like to thank Dr. Georgiou, the graduate coordinator, for suggesting DAW and providing valuable suggestions and support. I would like to thank Dr. Mendoza, my advisor, for the numerous discussions on the database design and functionality that vastly improved the quality of DAW. I would like to thank Dr. Yu for clarifying all my questions on implementation and system structure and providing the tools that made DAW possible. I would also like to thank Dr. Voigt and Dr. Murphy for teaching me the programming and theoretical skills. Finally, I thank the faculty and the staff of the Department of Computer Science, who made my stay pleasant in the past two years.
# Table of Contents

Abstract .................................................................................................................. iii

Acknowledgements ............................................................................................... iv

Chapter 1. Introduction ......................................................................................... 1

Chapter 2. DAW Architecture .............................................................................. 3

Chapter 3. Database Design .................................................................................. 5

  3.1 Data Analysis .................................................................................................. 5

  3.2 Database Schema Conceptual Model - ER Diagram ...................................... 5

  3.3 Database Schema Logical Model - Relational Schema Design ...................... 8

Chapter 4. Functional Description ....................................................................... 11

  4.1 Functional Partition ....................................................................................... 11

  4.2 Create Operation ......................................................................................... 11

  4.3 Delete Operation ......................................................................................... 12

  4.4 View Operation ........................................................................................... 12

  4.5 Update Operation ....................................................................................... 13

  4.6 Query Operation ......................................................................................... 13

  4.7 Administration ............................................................................................. 14

  4.8 Maintenance ................................................................................................. 14

Chapter 5. Project Implementation ..................................................................... 15

  5.1 DAW Graphical User Interface (GUI) Design ................................................ 15

    5.1.1 DAW Login ......................................................................................... 15

    5.1.2 Accessing Student Information ............................................................ 16
5.1.3 Accessing Course Information .................................................. 17
5.1.4 Performing Student Queries .................................................... 19
5.1.5 Performing Course Queries ..................................................... 19
5.1.6 Address Listing ................................................................. 20
5.1.7 E-Mail/Phone Listing .......................................................... 20

5.2 DAW Implementation .............................................................. 21
5.2.1 Checking Authorization ......................................................... 21
5.2.2 HTML Pages ..................................................................... 22
5.2.3 Input And Output Processing ............................................... 22
5.2.4 Database Query Module ...................................................... 23

Chapter 6. System Validation ......................................................... 25
6.1 Unit And Integration Testing ..................................................... 25
6.2 General Functionality Testing .................................................... 26

Chapter 7. Future Developments ...................................................... 28
7.1 Security .............................................................................. 28
7.2 Transaction Log ................................................................... 29
7.3 Binary Large Object (BLOBs) ................................................... 30
7.4 Object-Relational Model ......................................................... 30

APPENDIX A: User Manual ............................................................ 31

APPENDIX B: System Manual And Source Code Documentation .......... 32
B.1 DAW Source File Organization And Naming Convention ............. 32
B.2 Code Organization ............................................................... 34
List Of Figures

1.1 Overview Of DAW ................................................................. 1
2.1 DAW Architecture .............................................................. 3
3.1 ER Diagram ........................................................................ 6
3.2 DAW Database Schema ........................................................ 8
5.1 DAW Directory Structure ..................................................... 21

List Of Tables

6.1 DAW Testing Check List ....................................................... 26
Chapter 1. Introduction

This project, Database Access through the Web (DAW), implements a database to store academic and general information of graduate students in the Department of Computer Science, CSUSB and provides access to the database from the web. The motivation of the project comes from needs of the Graduate Coordinator, professors and department staff to access through the Internet student information concurrently.

![Diagram of DAW System](image)

**Figure 1.1 Overview Of DAW**

DAW is designed to store, access, edit, update and administrate student information through a user-friendly and intuitive graphical user interface (GUI) and through an Internet browser like Netscape Navigator or Internet Explorer. On-line help for the GUI is provided to the user on the web. Since the nature of the data stored in a student database is not public domain, DAW provides restricted access to a set of registered users, who need passwords to read, write and administrate the database.
DAW application package contains the source code (.c files, .h files, makefile, source code document), html files, gif files, data base administration files (.sql files to create, update, delete and clean the student database directly on the server). There is also a README file that describes the DAW tree and gives a step by step procedure to install DAW and get started on using DAW. At present, DAW server can be installed on any UNIX system with an ODBC driver for PostgreSQL.
Chapter 2. DAW Architecture

The components needed to implement DAW are a database server, a database interface Application Programming Interface (API) to programmatically access the database, a web server, a web browser, graphical user interface (GUI) components and an interface between the GUI and the application. The following figure describes the interaction among the components used in DAW.

![Figure 2.1 DAW Architecture](image)

The components used to build DAW were chosen with the following criteria:

(i) the components should be shareware, i.e., available freely for non-commercial purposes,
(ii) be part of a standard, i.e., they do not depend on a specific operating system and hence are easily portable across systems with ease,
(iii) database server independent, so that new and different versions of the server can be plugged in easily.

The user interface components are built using HTML 3.0 forms and frames and the applications are launched using the Common Gateway Interface (CGI). CGI was used because the standard compilers could be used to create a CGI program. Also, it is easy to pass the user input to the application through CGI.
The CGI programs are written in C. The choices for creating a CGI program were through Perl scripts, C/C++ etc. C was chosen since it is easy to build large applications and is widely used in academia and the standard compilers are available everywhere as opposed to C++.

The database choices available to DAW were PostgreSQL and miniSQL. PostgreSQL is a real multi-user database as opposed to miniSQL, which is meant for a single user, desktop application. Since, DAW can be accessed by many users at the same time it is better to use PostgreSQL (which serializes the transactions, etc) than miniSQL which simulates a multi-user scenario. Also, PostgreSQL server supports remote connections through remote procedure call (RPC).

Another reason for the choice of PostgreSQL is the availability of the ODBC driver for PostgreSQL so that the application can be written using ODBC API alone instead of having embedded SQL statements or embedded SQL client API. Having the database independent client API has two major advantages. Firstly, the application is not affected by upgrading the version of the ODBC driver for the database, as it is always backward compatible. Secondly, the same code base could be used to link with the drivers of other databases, thereby making it database independent.
Chapter 3. Database Design

3.1. Data Analysis

The data for designing and implementing the schema of the database was primarily obtained from the Department of Public Administration (DPA), which currently has a running version of a graduate student database application for the department’s use. Although, the database designed for the Department of Computer Science is very much similar to those of the DPA, there were many significant modifications that need to be done to represent all the relevant information and only the relevant information that are required by the Computer Science Department are incorporated into the database. For instance, GMAT score is only required for a Public Administration graduate student whereas Project or Thesis topic, advisor and committee members information are required only for a Computer Science graduate student. The data for the schema design also came from numerous consultations with the academic staff of the Department of Computer Science.

3.2. Database Schema Conceptual Model – ER Diagram

In designing the schema for the DAW database, two distinct entities have been identified. First is the student information and the second is the course information. The student entity (and its attributes), course entity (and its attributes) and the relationship between the two entities are described in detail in the entity-relationship (ER) diagram shown in Figure 3.1.
Figure 3.1 ER Diagram
Figure 3.1 ER Diagram (contd)
3.3. Database Schema Logical Model – Relational Schema Design

The conceptual model ER diagram maps into the following relational table design. In the following tables, bold fields indicate the primary key.

**CourseListing:**

<table>
<thead>
<tr>
<th>CourseNumber</th>
<th>Quarter</th>
<th>Units</th>
<th>Description</th>
<th>CourseType</th>
<th>CourseHistory</th>
</tr>
</thead>
</table>

Course Type: Course, Pre-requisite, Elective

**GeneralInformation:**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Gender</th>
<th>Ethnicity</th>
<th>VISA Status</th>
</tr>
</thead>
</table>

VISA Status International, Domestic

**PreschoolInformation:**

<table>
<thead>
<tr>
<th>SSN</th>
<th>Degree/Major</th>
<th>School</th>
<th>GraduationYear</th>
<th>GPA</th>
</tr>
</thead>
</table>

**GreToefl:**

<table>
<thead>
<tr>
<th>SSN</th>
<th>TOEFL</th>
<th>GREVerbal</th>
<th>GREQuant</th>
</tr>
</thead>
</table>

Note: -99 represents waived
>0 represents score

**JobInformation**

<table>
<thead>
<tr>
<th>SSN</th>
<th>EmployerName</th>
<th>EmployerAddress</th>
<th>WorkStatus</th>
<th>JobDescription</th>
</tr>
</thead>
</table>

Figure 3.2 DAW Database Schema
**Program Information:**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>AdmissionDate (text)</th>
<th>ProgramStatus (enum)</th>
<th>MastersOption (enum)</th>
</tr>
</thead>
</table>

Program Status: Conditional, Classified, Prelim-Candidacy, Probationary, Graduate, Advanced, Inactive, Active,
Masters Option: Project, Thesis, Undecided

**Program History:**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>Status (text)</th>
<th>FromDate (text)</th>
<th>ToDate (text)</th>
</tr>
</thead>
</table>

**Course Taken:**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>CourseNumber (text)</th>
<th>Year (text)</th>
<th>Grade (text)</th>
<th>Comments (text)</th>
</tr>
</thead>
</table>

**Prerequisites:**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>CourseNumber (text)</th>
<th>Taken (text)</th>
</tr>
</thead>
</table>

**Total GPA (view)**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>TotalUnits (int)</th>
<th>GPA (float)</th>
</tr>
</thead>
</table>

**TA and RA Information:**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>TA (text)</th>
<th>RA (text)</th>
</tr>
</thead>
</table>

Note: Values in Total GPA will be computed by DAW

**Contact Information**

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>Address (text)</th>
<th>HomePhone (text)</th>
<th>WorkPhone (text)</th>
<th>E-Mail (text)</th>
<th>URL (text)</th>
<th>FAX/Page (text)</th>
</tr>
</thead>
</table>

*Figure 3.2 DAW Database Schema (contd.)*
Committee Information:

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>Advisor (text)</th>
<th>Committee (text)</th>
<th>Topic (text)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Student Notes:

<table>
<thead>
<tr>
<th>SSN (text)</th>
<th>Notes (text)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.2 DAW database schema (contd.)
Chapter 4. Functional Description

4.1. Functional Partition

DAW is designed to perform six different types of operations viz., create, delete, view, update, query, administration and maintenance.

4.2. Create Operation

A user can create a new student record or course record through a create operation. To create a student record, the user must specify a name and a SSN. The SSN must be unique, i.e., there cannot be multiple student records in the database with the same SSN. However, there can be multiple student records with the same student name. While creating a student record the user can also specify gender, ethnicity, VISA status, previous school information, TOEFL, GRE Verbal, GRE Quantitative, waiver for GRE/TOEFL (if any), job information (if any), admission date, program status, list of courses taken, list of pre-requisites needed, GPA, masters option information (advisor/committee), contact information and TA/RA information. All unspecified information is set to a value of 0 (for the integer and float data types) or an empty text (for the text data type) or a default value (for the enum data types). While creating a course record the user must specify a unique course number. The user can also specify the quarter(s) (in which the course is offered), the number of units, course description, course type and previous course information. Once again, all
specified information is given the appropriate default value. DAW supports concurrent creates to the database. However, when two users try to create the same record, a database error will occur.

4.3. Delete Operation

A user can delete an existing student or course record. To delete a student record the user can either select a name from a given set of student names or enter a name or enter a student’s SSN. Since DAW delete operation needs to identify a unique student record, the user must provide the SSN in the case where there are many students with the same name. Otherwise DAW displays an error message. Similarly, the user can delete an existing course record by selecting a course number from a given set of courses.

4.4. View Operation

A user can view a student academic/general information or course information. A student record can be viewed by either selecting a student name from a set of student names or by entering a student’s name or by entering the student’s SSN. In case of multiple student records with the same name, the user can distinguish between selections by entering the SSN. The student academic information contains student SSN, name, admission date, program status, list of courses taken, list of pre-requisites, GPA, masters option information (advisor/committee), contact information and TA/RA information. The student
general information contains student SSN, name, gender, ethnicity, nationality, previous school information, TOEFL, GRE Verbal, GRE Quantitative, waiver for GRE/TOEFL (if any), job information (if any). Similarly, the user can view a course record by choosing a course from a given set of course numbers. The course information contains the quarter (in which the course is offered), the number of units, course description, course type and previous course information.

4.5. Update Operation

A user can update the student general/academic information or course information. All fields in the student general information can be updated except the student SSN. To modify the SSN, the user has to first delete the student record corresponding to the old SSN and create a fresh entry with the new SSN. All fields in the student academic information can be updated except the student SSN and student name. All the fields in the course information can be updated except the course number.

4.6. Query Operation

DAW provides a basic query generator through which a user performs the commonly used student and course queries. For example, the user can select a set of students based on their GPA range, status along with their pre-requisites.
Also a set of courses can be selected based on course type. The queries will be typically used for report generation.

4.7. Administration

A user with administrative privilege can create new DAW users, delete existing DAW users, and update the existing DAW user privileges. Also, a user with administrative privileges can access and maintain notes on a student’s academic information.

4.8. Maintenance

DAW provides an option for backing-up all the records in a table in the database to an output file in a predefined format. For example, all entries in a table named ContactInformation will be copied to a file named ContactInformation.txt. Each record will be delimited by a new-line character, i.e., ‘\n’, and every fields in a record will be delimited by |. The files will reside under $(DAWHOME)/public_html/backup. This option is available only for the users with administrative privileges.
Chapter 5. Project Implementation

5.1. DAW Graphical User Interface (GUI) Design

DAW GUI is easy to use, intuitive, and can be extended beyond the scope of DAW. The GUI is written only using Hyper Text Markup Language (HTML) Version 3.0 forms. Hence, the DAW GUI is portable across any web-server that supports HTML 3.0.

The GUI components that have been used include Text Box, Select Option, Text Area, Button, Image Button, Mail Tool, Hyper Links, Tables and HTML Frames. The GUI has four logical groups -- Student, Course, Reports and Administration. The following sub sections explain the UI work flow.

5.1.1. DAW Login

The user logs in by providing a user id and a password. After verifying the user id and password, the UI control goes to the DAW home page. From now on, the DAW navigator is available at the left-hand side on all UI forms. The user can launch all operations from the navigator.
5.1.2. Accessing Student Information

**Student Create Sequence**

<table>
<thead>
<tr>
<th>DAW Navigator</th>
<th>DAW Student</th>
<th>DAW Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click Student Create</td>
<td>Fill Form, Click Create</td>
<td>Displays created record</td>
</tr>
</tbody>
</table>

The user can create a student record by filling the student form, and clicking the create button. The created student record will be displayed with appropriate default values.

**Student Delete Sequence**

<table>
<thead>
<tr>
<th>DAW Navigator</th>
<th>DAW Student</th>
<th>DAW Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click Student Delete</td>
<td>Select SSN or Name, Click Delete</td>
<td>Display the deleted status</td>
</tr>
</tbody>
</table>

The user can delete a student record by selecting a student Name or by entering a student’s name or by entering student’s SSN from a choice box and clicking the delete button. The deleted status will be displayed.

**Student General Info Sequence**

<table>
<thead>
<tr>
<th>DAW Navigator</th>
<th>DAW Student</th>
<th>DAW Student</th>
<th>DAW Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click Student General Info</td>
<td>Select Name or SSN Click View</td>
<td>Displays General Info</td>
<td>Displays updated General Info</td>
</tr>
<tr>
<td>Displays General Info, Edit fields</td>
<td>Click Update</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16
The user can view a student’s general information by selecting a student’s name and clicking the View button. The user can update the general information by modifying the student form and clicking the update button. The updated general information (with appropriate default values) will be displayed.

Student Academic Info Sequence

The user can view a student’s academic information by selecting a student’s name and clicking the View button. The user can update the academic information by modifying the student form and clicking the update button. The updated academic information (with appropriate default values) will be displayed.

5.1.3. Accessing Course Information
The user can create a course record by filling the course form, and clicking the create button. The created course record will be displayed with appropriate default values.

The user can delete a course record by choosing a course number from the choice box and clicking the delete button.

The user can view a course information by selecting a course name and clicking the View button. The user can update the course information by
modifying the course form and clicking the update button. The updated course information (with appropriate default values) will be displayed.

5.1.4. Performing Student Queries

![Student Query Sequence Diagram]

The user can perform student queries by clicking the student query button, generate the query through the choices and selections and clicking the View button to execute the query. The results of the query including student SSN, name, GPA, Program Status and Pre-requisites will be displayed sorted by the SSN.

5.1.5. Performing Course Queries

![Course Query Sequence Diagram]

The user can perform course queries by clicking the course query button, which generates the query through the choices and selections; clicking the View
button executes the query. The results of the query includes Course Number, Course Type, Quarter offered, Units and Course Description will be displayed in a sorted order (by course number).

5.1.6. Address Listing

By clicking the address listing option from the navigator the user can display the addresses of all students sorted by the SSN.

5.1.7. E-Mail/Phone Listing

By clicking the address listing option from the navigator the user can display the e-mail address and telephone number of all students sorted by the SSN.
5.2. DAW Implementation

As explained in Chapter 2, DAW architecture consists of five components, checking access rights, generating pre-defined and on the fly HTML forms, input and output processing, creating database queries and managing login information.

DAW component directory structure is shown below:

![DAW Components Directory Structure](image)

![Figure 5.1 DAW Directory Structure](image)

5.2.1. Checking Authorization

Every DAW operation requires the user to have certain privileges. Certain types of operations are available only for the users with special privileges. For instance, only a user with administrative privilege can add a new DAW user. Before the user performs a DAW operation the DAW Authorization Module verifies the user’s authorization. This is done by reading the username and password (either in the hidden field of an HTML form or in the
QUERY_STRING) provided by the user during login process and verifying it against the database.

5.2.2. HTML Pages

There are 3 pre-defined HTML pages, -- daw_main.html, daw_login.html, daw_library.html and daw_explanation.html. They are found under the directory $(DAWHOME)/public_html/html. daw_main.html contains a brief description of DAW and provides an entry point for registered users. DAW_login.html enables a registered user to log in. A detailed description (on-line help) of the functionality, input formats and other relevant information could be found in daw_explanation.html and daw_library.html. The generated html pages are part of the CGI/C programs. The generated html variable names are defined in file daw.h. The html forms are generated through a set of helper functions defined in $(DAWHOME)/daw/src/utility.[ch] and across many other files. Refer to the source code documentation for further details.

5.2.3. Input And Output Processing

All inputs and outputs are processed by the CGI programs in $(DAWHOME)/public_html/bin. The CGI program reads the input either from a HTML form using the POST method or through a hyperlink using QUERY_STRING. The CGI program parses the input through the helper functions available in utility.c and sends the tokens to the database query.
module. The output obtained from the database query module will be displayed to the user through the generated HTML pages.

5.2.4. Database Query Module

The database query module is a set of functions that performs fetch, update and query on different tables in the database. Every database operation in DAW is implemented through a set of ODBC C API. For example a connection to the database is done through SQLAllocEnv (allocates the environment), SQLAllocConnect (allocates the connection handle) and SQLDriverConnect (performs the connection). A database query is performed through SQLAllocStmt (allocates the statement handle) and SQLExecDirect. The results of query are obtained through SQLBindCol (which binds the resulting column to the host variable) and SQLFetch (fetches single resulting row into the bound host variable). There is a set of helper functions, available in utility.c, to create, delete and update rows.

DAW assumes that an update operation alters all the attribute values. A DAW update operation does not always correspond to altering a set of fields in a row. For tables that have multiple rows for a student's SSN (for example, CoursesTaken), it is possible to have an update operation that involves addition of $m$ new rows and deletion of $n$ existing rows with $m$ not equal to $n$. Hence in DAW an update operation is realized through deletes followed by inserts. Since, every database operations involves inter-process and RPC communication which
are expensive, DAW does some optimization to minimize the number of those operations
Chapter 6. System Validation

6.1. Unit And Integration Testing

The unit testing of DAW included the functionality groups: access privileges (read, read-write and administrator), student create, student delete, student view, student update, course create, course delete, course view, course update, student query, course query, address listing, e-mail listing and student notes. All the testing was done through the user interface. Every testing was cross checked by directly making database queries through psql. The integration testing was minimal as every DAW operation involves executing a separate CGI program.

Create operations were tested by giving different sets of input values and checking the value through psql and the view operation. As create and view were independent operations, the above testing provides a good measure of the program stability and robustness. Delete operations were tested by deleting different values and cross checking with psql and view operation. View operations were tested by cross checking with psql. Update operations were tested by updating different sets of values and cross checking with psql and view operation.

Query and listing operations were tested by enumerating over all possible choices for the query strings and cross checked by executing the same sql statements using psql.
6.2. General Functionality Testing

DAW has been used by different faculty and staff of the department of computer science and others who gave feedback on intuitiveness and ease of use of the user interface, and suggested modifications which increased the usefulness and quality of DAW. The following table describes the validations performed.

<table>
<thead>
<tr>
<th>Type of Operation</th>
<th>Checking Criteria</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student Notes</td>
<td>Check display of student notes image in the academic information if the user has administrative privileges</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of student notes in a new web browser</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Modify and click Update. Check modified record through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Modify and click Update. Check modified record through GUI</td>
<td>x</td>
</tr>
<tr>
<td>Login Create</td>
<td>Display Empty Form if user access is administrator</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Click create button, Check the username and password requirement</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Add Values, click create button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the created login record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td>Login Information</td>
<td>Check display of &quot;no login information&quot; message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the username in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Choose username.Click View Display all login information in the form</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Change Values, click update button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the modified login record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td>Login Delete Information</td>
<td>Check display of &quot;no login information&quot; message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the username in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion message in the GUI</td>
<td>x</td>
</tr>
</tbody>
</table>

Table 6.1 DAW Testing Check List
<table>
<thead>
<tr>
<th>Type of Operation</th>
<th>Checking Criteria</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student Create</strong></td>
<td>Display Empty Form</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Click create button, Check the SSN and name requirement</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Add Values, click create button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the created student record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Student General, (academic) Information</strong></td>
<td>Check display of <em>no student name</em> message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the student name in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Choose name. Click View Display all general (academic) information in the form</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Change Values, click update button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the modified student record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Student Delete Information</strong></td>
<td>Check display of <em>no student name</em> message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the student name pair in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion message in the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Course Create</strong></td>
<td>Display Empty Form</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Click create button, Check the Course number requirement</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Add Values, click create button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the created course record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Course Information</strong></td>
<td>Check display of <em>no course information</em> message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the course number in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Choose course number. Click View Display all course information in the form</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Change Values, click update button, check the database contents through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the validity of the modified course record through the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Course Delete Information</strong></td>
<td>Check display of <em>no course information</em> message if database is empty</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check display of the student course number in the select option</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion through psql</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Check the deletion message in the GUI</td>
<td>x</td>
</tr>
<tr>
<td><strong>Student Query</strong></td>
<td>Check the display of the student query form parameters</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Enumerate over all query parameter. Click View. Check display of results and its accuracy</td>
<td>x</td>
</tr>
<tr>
<td><strong>Course Query</strong></td>
<td>Check the display of the course query form parameters</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Choose course type and click view. Check display of results and its accuracy</td>
<td>x</td>
</tr>
<tr>
<td><strong>Address listing</strong></td>
<td>Check accuracy of all student address listing in sorted order</td>
<td>x</td>
</tr>
<tr>
<td><strong>e-mail/phone listing</strong></td>
<td>Check accuracy of all student e-mail and phone listing in sorted order</td>
<td>x</td>
</tr>
</tbody>
</table>

*Table 6.1 DAW Testing Check List (contd.)*
Chapter 7. Future Developments

7.1. Security

One area where DAW is not fool-proof is security. There are two ways of passing information between HTML forms and CGI program. One is the post method and the other is using QUERY_STRING environment variable. In order to make DAW more secure, the user name and password, which is required for verifying a user’s authorization, is passed in the hidden field of HTML form or as a query string to another CGI program. A potential problem with the first approach is that a web browser’s source view option will list the value of a hidden field. The problem with the QUERY_STRING variable is the password is displayed as part of the URL. DAW uses both QUERY_STRING and hidden fields to pass access information across the HTML pages.

An alternative approach is to create a unique transaction id and pass the transaction id instead of the user name and password. For every login, there is a row created in the login-transaction table that contains the transaction id, user name and access privileges. This will also help maintain a user log. This method can be improved further as follows: generate a random key and encrypt the user id and password using the key, store it in the log table and pass the key and the log table entry.

Another approach would be to use the Web browsers Secure Socket Layer (SSL). The primary goal of the SSL Protocol is to provide privacy and reliability between two communicating applications. The protocol is composed
of two layers. At the lowest level, layered on top of some reliable transport protocol is the SSL Record Protocol. The SSL Record Protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the SSL Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. One advantage of SSL is that it is application protocol independent. A higher level protocol can layer on top of the SSL Protocol transparently. The SSL protocol provides connection security that has three basic properties:

- The connection is private. Encryption is used after an initial handshake to define a secret key. Symmetric cryptography is used for data encryption.
- The peer’s identity can be authenticated using asymmetric, or public key, cryptography.
- The connection is reliable.

7.2. Transaction Log

Another improvement to DAW is to maintain a user transaction log to monitor the access of the student database by the DAW administrator. The log table can include the time of transaction, user name, user privilege, type of operation performed (view, update, delete, create etc.).
7.3. Binary Large Objects (BLOBs)

At present none of the DAW tables store binary data (BLOBs). It may be necessary to store images (for example pictures of graduate students, advisor and committee) and documents (for example application forms, letter of recommendation, etc.) in the DAW tables.

7.4. Object-Relational Model

Since DAW is designed to store only graduate student information, it does not fully exploit the object-relational capabilities of PostgreSQL. DAW can be extended to store both graduate and undergraduate student information by improving the database design and by incorporating some of PostgreSQL's special features such as inheritance, functions etc.
APPENDIX A:

User Manual

DAW provides an on-line document and a navigator frame that navigates through the user interface and the DAW functionality. The naming convention used in the GUI has been carefully chosen so that it is clear, consistent and uniform throughout the application, and easy to use for a first time user.
APPENDIX B:

Systems Manual And Source Code Documentation

B.1. DAW Source File Organization And Naming Conventions

$(DAWHOME)/daw/src contains all the source code (.c and .h) files, the
makefile and a rebuild.exe (which makes all the CGI programs and transfers
them to $(DAWHOME)/public_html/bin directory). The source files can be
partitioned into six components -- global module, student module, course
module, report module, administrator module and utilities.

The global module contains
daw.h - defines all HTML variable names and max size of the variable,
content.c - implements the navigator column,
login.c - implements the login entry to DAW.

The student module contains
s_create.c – provides a form for entering new student information,
s_create_update.c – updates the database with new student information,
s_delete – provides a form for deleting a student record,
s_delete_update.c – performs the deletion of a student record,
s_general.c – provides the form for displaying the student general information,
s_general_value.c – displays student general information,
s_general_update.c – updates the student general information,
s_academic.c – provides the form for displaying the student academic information,

s_academic_value.c – displays student academic information,

s_academic_update.c - updates the student academic information.

The course module contains

c_create.c – provides a form for entering new course information,

c_create_update.c – updates the database with new course information,

c_delete – provides a form for deleting a course record,

c_delete_update.c – performs the deletion of a course record,

c_view.c – provides the form for displaying the course information,

c_view_value.c – displays course information,

c_view_update.c – updates the course information.

The report module contains

r_student_query.c – provides the html form for student query

r_student_query_value.c – performs and displays student query

r_course_query.c – provides the html form for course query

r_course_query_value.c – performs and displays course query

r_address.c – displays the address of all students

r_email.c – displays the e-mail of all students

The administrator module contains

l_create.c – provides a form for entering new login information,

l_create_update.c – updates the database with new login information,
l_delete – provides a form for deleting a login record,
l_delete_update.c – performs the deletion of a login record,
l_view.c – provides the form for displaying the login information,
l_view_value.c – displays login information,
l_view_update.c – updates the login information.

The maintainence module contains
m_backup.c – provides a form for entering backup directory,
m_backup_do.c – performs backup operation

The utility module contains
utility.h – contains the utility function prototypes
utility.c – contains the utility function definitions.

B.2. Code Organization

DAW code is modular, of commercial quality, easy to test, modify, extend
and contains very little redundant code. Errors are handled all throughout, and
the system outputs meaningful error messages in case of failure. Additional
functionality can be easily incorporated in the framework with minimal changes.
Though it is written in a procedural language, it simulates an object oriented
approach.
B.3. Naming Conventions And Comments

DAW follows a strict and uniform standard to name static and non-static functions, global, static global, local variables. All variable names follow the Hungarian notation, i.e., start with a lower case and use upper case to identify different words, for example *displayStudentNotes*. All global variables begin with a ‘g’, for example *gCourseTypeCount*. All pointer variables start with a ‘p’, for example *pUserName*. All non-static function names starts with an uppercase and expanded fully. For example *UTIL_ConnectToPostgreSQL()*. All static functions start with an ‘S’, for example, *S_GetStudentAcademicInformation()*.

Almost all functions return an enum RETURN_CODE with exceptions wherever necessary. All functions will fit in a single screen and their code complexity is minimized. Comments are provided wherever necessary and by choosing the variable names carefully, the code is made to self-document.

B.4. Extending DAW Functionality

One of the strong points of the DAW design is that the database management functions and the GUI functions are independent of each other. Adding a new DAW functionality entails either database schema changes or GUI changes or both.
B.4.1. Extending The Database

A change in the database schema involves creating/deleting a table or modifying an existing table. When a table is added, the system administrator (SA) has to modify the SQL script $(DAWHOME)/daw/dbase/createdaw.sql appropriately, implement fetch, insert and update functions by following the templates provided in utility.c. When a table is deleted the SA does not have to do anything (or he can remove the appropriate fetch, insert and delete functions).

When a table schema is changed, the SA has to modify the corresponding fetch, update and insert functions. DAW follows a strict naming convention for function names. For example, if the table PreschoolInformation is modified, the SA has to modify UTIL_UpdatePreschoolInformation(UpdateOperation) and UTIL_FetchPreschoolInformation(). The input and output parameters of all database functions follow a specific standard.

B.4.2 Extending The GUI

A change in the GUI involves adding/removing a HTML component, modifying an existing component format/name. The HTML component can be a choice box, text box, text area, select option, etc. The HTML pages of DAW are either pre-defined or generated by a CGI program. Changing pre-defined HTML pages are straightforward as all the DAW pages are written using HTML 3.0.
Updating the GUI can be illustrated by the following examples. In the first example, suppose the SA wants to add new input box to the student academic information. The SA needs to perform the following steps: (i) update the definitions HTMLTerms, TotalHTMLTerms, gHTMLTermStrings, StudentCreateUpdateHTMLTerms, StudentAcademicValueHTMLTerms, StudentAcademicUpdateHTMLTerms in daw.h. (ii) change the generated HTML pages in s_create.c, s_create_update.c, s_academic.c, s_academic_value.c, s_academic_update.c (here s_ stands for student).

In the second example, suppose the SA wants add an option to available select option, for example adding to Job Status, the SA needs to update gJobStatusCount, gJobStatus in daw.h.

**B.5 Installing DAW**

Refer to the README file under $(DAWHOME) for DAW installation procedure.
APPENDIX C:

ODBC Calls

C.1. SQLAllocConnect

Syntax: RETCODE SQLAllocConnect (henv, phdbc)

SQLAllocConnect accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HENV</td>
<td>henv</td>
<td>Input</td>
<td>Environment handle</td>
</tr>
<tr>
<td>HDBC FAR*</td>
<td>phdbc</td>
<td>Output</td>
<td>Pointer to storage for the connection handle</td>
</tr>
</tbody>
</table>

Returns: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_ERROR, or SQL_INVALID_HANDLE

Comments: SQLAllocConnect allocates memory for a connection handle within the environment identified by henv. A connection handle references information such as the valid statement handles on the connection and whether the transaction is currently open. To request a connection handle, an application passes the address of an hdbc to SQLAllocConnect. The driver allocates memory for the connection information and stores the values of the associated handle in hdbc.

C.2. SQLAllocEnv

Syntax: RETCODE SQLAllocEnv (phenv)

SQLAllocEnv accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HENV FAR*</td>
<td>phenv</td>
<td>Output</td>
<td>Pointer to storage for the environment handle</td>
</tr>
</tbody>
</table>

Returns: SQL_SUCCESS or SQL_ERROR

Comments: SQLAllocEnv allocates memory for an environment handle and initializes the ODBC call level interface for the application use. An application must call SQLAllocEnv before calling any other ODBC function. An environment handle references global information such as invalid connection handle, an application passes the
address of an *henv* to SQLAllocEnv. The driver allocates memory for the environment information and stores the value of the associated handle in the *henv*.

### C.3. SQLAllocStmt

**Syntax:** RETCODE SQLAllocStmt (*hdbc*, *phstmt*)

SQLAllocStmt accepts the following arguments:

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDBC</td>
<td><em>hdbc</em></td>
<td>Input</td>
<td>Connection handle</td>
</tr>
<tr>
<td>HSTMT FAR*</td>
<td><em>phstmt</em></td>
<td>Output</td>
<td>Pointer to storage for the statement handle</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLAllocStmt allocates memory for a statement handle and associates the statement handle with the connection specified by *hdbc*. An application must call SQLAllocStmt prior to submitting SQL statement. A statement handle references statement information, such as network information, SQLSTATE values and error messages, etc. To request a statement handle, an application connects to the data source and passes the address on an *hstmt* to SQLAllocStmt. The driver allocates memory for the statement information and stores the values of the associated handle in the *hstmt*.

### C.3. SQLBindCol

**Syntax:** RETCODE SQLBindCol (*hstmt*, *icol*, *fcType*, *rgbValue*, *cbValueMax*, *pcbValue*)

SQLBindCol accepts the following arguments:

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTMT</td>
<td><em>hstmt</em></td>
<td>Input</td>
<td>Statement handle</td>
</tr>
<tr>
<td>UWORD</td>
<td><em>icol</em></td>
<td>Input</td>
<td>Column number of result data</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLBindCol assigns the storage and data type for a column in a result set, as follows:

- A storage buffer that receives the contents of a column of data
• The length of the storage buffer
• A storage location that will receive the actual length of the column of data returned by the fetch operation
• Data type conversion

C.4. SQLDriverConnect

Syntax: RETCODE SQLDriverConnect (hdbc, hwnd, szConnStrIn, cbConnStrIn, szConnStrOut, cbConnStrOutMax, pcbConnStrOut, fDriverCompletion)

SQLDriverConnect accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDBC</td>
<td>hdbc</td>
<td>Input</td>
<td>Connection handle</td>
</tr>
<tr>
<td>HWND</td>
<td>hwnd</td>
<td>Input</td>
<td>Widget</td>
</tr>
<tr>
<td>UCHAR FAR*</td>
<td>szConnStrIn</td>
<td>Input</td>
<td>A full connection string</td>
</tr>
<tr>
<td>SWORD</td>
<td>cbConnStrIn</td>
<td>Input</td>
<td>Length of szConnStrIn</td>
</tr>
<tr>
<td>UCHAR FAR*</td>
<td>*szConnStrOut</td>
<td>Output</td>
<td>Pointer to completed connection string</td>
</tr>
<tr>
<td>SWORD</td>
<td>cbConnStrOutMax</td>
<td>Input</td>
<td>Maximum length of szConnStrOut</td>
</tr>
<tr>
<td>SWORD FAR*</td>
<td>pcbConnStrOut</td>
<td>Output</td>
<td>Number of bytes in szConnStrOut</td>
</tr>
<tr>
<td>UWORD</td>
<td>fDriverCompletion</td>
<td>Input</td>
<td>SQL_DRIVER_PROMPT, SQL_DRIVER_COMPLETE, SQLDRIVER_COMPLETE_REQUIRED, SQLDRIVER_NOPROMPT</td>
</tr>
</tbody>
</table>

Returns: SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA_FOUND, SQL_INVALID_HANDLE or SQL_ERROR

Comments: SQLDriverConnect provides the following connection options
• Connection can be established using a connection string that contains the data source name, one or more user Ids, one or more passwords, and other information required by the data source.
• Connection can be established by using a partial connection string or no additional information; in this case, the driver manager and the driver can each prompt the user for connection information.
• Connection can be established to a data source that is not defined in the odbc.ini file. If the application supplies a partial connection string, the driver can prompt the user for connection information.
Once the connection is established, SQLDriverConnect returns the completed connection string. The application can use this string for subsequent connection requests.

SQLDriverConnect uses a connection string to specify the information needed to connect to a driver and data source. A connection string has the following syntax:

\[
\begin{align*}
\text{connection_string} & : = \text{empty_string} | \text{attributes} | \text{attribute} ; \text{connection_string} \\
\text{empty_string} & : = \text{empty_string} \text{attribute} \text{attribute_value} \text{ DRIVER} \{\text{attribute_value}\} \\
\text{attribute_keyword} & : = \text{DSN} \text{ UID} \text{ PWD} \text{ driver_defined_attribute_keyword} \\
\text{attribute_value} & : = \text{character_string} \\
\text{driver_defined_attribute_keyword} & : = \text{identifier}
\end{align*}
\]

### C.5. SQLError

**Syntax:** RETCODE SQLExecDirect (henv, hdbc, hstmt, szSqlState, pfNativeError, szErrorMsg, cdErrorMsgMax, pcbErrorMsg)

SQLError affects the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HENV</td>
<td>henv</td>
<td>Input</td>
<td>Environment handle</td>
</tr>
<tr>
<td>HDBC</td>
<td>hdbc</td>
<td>Input</td>
<td>Connection handle</td>
</tr>
<tr>
<td>HSTMT</td>
<td>hstmt</td>
<td>Input</td>
<td>Statement handle</td>
</tr>
<tr>
<td>UCHAR FAR*</td>
<td>szSqlState</td>
<td>Output</td>
<td>SQLSTATE as null terminated string</td>
</tr>
<tr>
<td>SDWORD FAR*</td>
<td>pfNativeError</td>
<td>Output</td>
<td>Native error code</td>
</tr>
<tr>
<td>UCHAR FAR*</td>
<td>szErrorMsg</td>
<td>Output</td>
<td>pointer to error message text</td>
</tr>
<tr>
<td>SWORD</td>
<td>cbErrorMsgMax</td>
<td>Input</td>
<td>Max length of szErrorMsg</td>
</tr>
<tr>
<td>SWORD FAR*</td>
<td>pcbErrorMsg</td>
<td>Output</td>
<td>pointer to number of bytes available in szErrorMsg</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NO_DATA_FOUND, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLError return error or status information. An application typically calls SQLError when a previous call to an ODBC function returns SQL_ERROR or SQL_SUCCESS_WITH_INFO.
C.6. SQLExecDirect

**Syntax:** RETCODE SQLExecDirect (hstmt, szSqlStr, cbSqlStr)

SQLExecDirect accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTMT</td>
<td>hstmt</td>
<td>Input</td>
<td>Statement handle</td>
</tr>
<tr>
<td>UCHAR FAR*</td>
<td>szSqlStr</td>
<td>Input</td>
<td>SQL statement to be executed</td>
</tr>
<tr>
<td>SDWORD</td>
<td>cbSqlStr</td>
<td>Input</td>
<td>Length of szSqlStr</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_NEED_DATA, SQL_STILL_EXECUTING, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLExecDirect execute a preparable statement, using the current values of the parameter marker variables if any parameter exist in the statement. SQLExecDirect is the fastest way to submit an SQL statement for one-time execution.

C.7. SQLFetch

**Syntax:** RETCODE SQLFetch (hstmt)

SQLFetch accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTMT</td>
<td>hstmt</td>
<td>Input</td>
<td>Statement handle</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_STILL_EXECUTING, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLFetch fetches a row of data from a result set. The driver returns data for all columns that were bound to storage location with SQLBindCol. SQLFetch positions the cursor on the next row of the result set. Before SQLFetch is called the first time, the cursor is positioned before the start of the result set. When the cursor is positioned on the last row of the result set, SQLFetch returns SQL_NO_DATA_FOUND and the cursor is
positioned after the end of the result set. If the application called SQLBindCol to bind the columns, SQLFetch stores data into the locations specified by the calls to SQLBindCol.

C.8. SQLFreeConnect

Syntax: RETCODE SQLFreeConnect (hdbc)

SQLFreeConnect affects the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDBC</td>
<td>hdbc</td>
<td>Input</td>
<td>Connection handle</td>
</tr>
</tbody>
</table>

Returns: SQL_SUCCESS,
         SQL_SUCCESS_WITH_INFO,
         SQL_INVALID_HANDLE or
         SQL_ERROR

Comments: SQLFreeConnect releases a connection handle and frees all memory associated with the handle. Prior to calling SQLFreeConnect, an application must call SQLDisconnect for the hdbc. Otherwise SQLFreeConnect returns SQL_ERROR and the hdbc remains valid. SQLDisconnect automatically drops any hstmt open on the hdbc.

C.9. SQLFreeEnv

Syntax: RETCODE SQLFreeEnv (henv)

SQLFreeEnv affects the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HENV</td>
<td>henv</td>
<td>Input</td>
<td>Environment handle</td>
</tr>
</tbody>
</table>

Returns: SQL_SUCCESS,
         SQL_SUCCESS_WITH_INFO,
         SQL_INVALID_HANDLE or
         SQL_ERROR

Comments: SQLFreeEnv frees the environment handle and releases all memory associated with the environment handle. Prior to calling SQLFreeEnv, an application must call SQLFreeConnect for any hdbc allocated under the henv. Otherwise, SQLFreeEnv returns SQL_ERROR and the henv remains valid.
C.10. SQLFreeStmt

**Syntax:** RETCODE SQLFreeStmt (*hstmt, fOption*)

SQLFreeStmt accepts the following arguments.

<table>
<thead>
<tr>
<th>Type</th>
<th>Argument</th>
<th>Use</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSTMT</td>
<td><em>hstmt</em></td>
<td>Input</td>
<td>Statement handle</td>
</tr>
<tr>
<td>UWORD</td>
<td><em>fOption</em></td>
<td>Input</td>
<td>SQL_CLOSE, SQL_DROP, SQL_UNBIND, (or) SQL_RESET_PARAMS</td>
</tr>
</tbody>
</table>

**Returns:** SQL_SUCCESS, SQL_SUCCESS_WITH_INFO, SQL_INVALID_HANDLE or SQL_ERROR

**Comments:** SQLFreeStmt stops processing associated with a specific *hstmt*, closes any open cursors associated with the *hstmt*, discards pending results, and, optionally frees all resources with the statement handle. An application can call SQLFreeStmt to terminate processing of a SELECT statement with or without canceling the statement handle.
APPENDIX D:

URLs

2. PostgreSQL http://www.postgresql.org
3. PERL http://www.perl.com/perl
4. PERL http://www.us.ufl.edu/perl
5. CGI http://www.yahoo.com
6. CGI http://blackcat.brynmawr.edu/~nsrwobodo/prog-html.html
7. DBI http://www.hermetica.com
8. PHP/FI http://www.vex.net/php/
APPENDIX E:

DAW Screen Shots
Welcome to DAW - Data Access from the Web

Student Database Access from the Web (DAW) is used to create, query, edit, maintain and administer academic and general information pertaining to the graduate and undergraduate students of the Department of Computer Science, undergraduate and graduate course offered by the department at California State University, San Bernardino, through Internet.

The database is accessible only to the authorized users with appropriate privileges. If you are a registered user please **Login**. If you are interested in registering please contact:

The Graduate Program Director,
Department of Computer Science,
CSUSB, CA 925007.
Tel : (909)880-5326.
e-mail: georgiou@csci.csusb.edu
Welcome to DAW - Data Access from the Web

Username: ___________________________

Password: __________________________________________

[Login] [Reset]
Student Information

Create:
This creates a new student record

Delete:
This deletes an existing student record.

General Information:
The following information are available on the student

- Full Name
- Social security
- Contact Information

- Gender
- Ethnicity
- VISA Status

- Previous Degree and GPA
- GRE and TOEFL scores
- Other Job Information

Academic Information:
The following information are available on the student

- Admission Date
- Program Status
- TA / RA Information

- Pre-requisites
- Course Taken
- Overall GPA

- Masters Option
- Masters Option Topic
- Advisor and Committee members

Course Information

Create:
This provides a form to add new course to the database

Delete:
This removes an existing course from the database

View:
The following information are available on the course

- Course number
- Number of Units

- Course Type
- Previous Information

Course Type

Report Generation

Student Queries:
Performs queries concerning student status, GPA and Pre-requisites

Course Queries:
Performs queries concerning course type

Student Address Listing:
Displays name and addresses of all students
Course Information

Course number Course Number is alphanumeric and represents the course offered by the department.

Quarter Quarter is alphanumeric and represents the quarters the above course was/will be offered.

Units Units is numeric and represents the number of units for the above course.

Description Description is alphanumeric and is a description (title) for the above course.

Course type Course type is alphanumeric and it is either a Core course, a Pre-Requisite or an Elective.

Student Information

General Information

SSN Social Security Number is alphanumeric
Name Student Name is alphanumeric
Gender Gender is either Male or Female
Ethnicity Ethnicity is alphanumeric
Visa Status Visa Status can either be International or Domestic.

Committee Information

Address Address is alphanumeric.
Phone (home), Phone (work), Fax Phone (home), Phone (work) and Fax are alphanumeric.
E-mail, URL E-mail and URL are alphanumeric.

Preschool Information

Degree/Major Degree/Major is alphanumeric and represents the highest degree obtained by the student.
School School is alphanumeric and represents the school where the above degree was obtained.
Year Year is numeric and represents the year graduated from the above school.
GPA GPA is real and represents the Grade Point Average of the lastest Degree/Major.
Other Degree(s) Other Degree(s) stores all other degree information. The format for entering information is:
Degree School City State GPA Year
**DAW Student Create Form**

**Enter new student information**

<table>
<thead>
<tr>
<th>Name</th>
<th>Create</th>
</tr>
</thead>
<tbody>
<tr>
<td>SocialSecurityNumber</td>
<td>Gender Male</td>
</tr>
<tr>
<td>Date Of Birth</td>
<td>Visa Status Domestic</td>
</tr>
<tr>
<td>Ethnicity</td>
<td></td>
</tr>
</tbody>
</table>

**Contacts**

<table>
<thead>
<tr>
<th>Address</th>
<th>Phone (home)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Phone (work)</td>
</tr>
<tr>
<td></td>
<td>Fax</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>E-Mail</th>
<th>URL</th>
</tr>
</thead>
</table>

**Pre-School Info**

- **GRE (Verbal)**
- **GRE (Quant)**
- **GRE (Analytical)**
- **TOEFL**

**Highest Degree**

- **Major/Degree**
- **School**
- **Year**

**Other Degree (s)**

**Program Info**

- **Admission Date**
- **Program Status Conditional**

**Program History**

**Course Info**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>

51
Select student name/SSN for deletion

- Name: Abdelmessih, Victor
- Name: 
- SSN: 

Delete
## DAW Student General Information

### Select student name for viewing

- **Name**: Abdelmessih, Victor

### Contacts

- **Address**:
- **E-Mail**:
- **Phone (home)**
- **Phone (work)**
- **Fax**
- **URL**

### Pre-School Info

- **GRE (Verbal)**
- **GRE (Quant)**
- **GRE (Analytical)**
- **TOEFL**
- **Highest Degree**
- **Major/Degree**
- **Year**

### Other Degree(s)

- **School**
- **GPA**

### Job Info

- **Employer Name**: Not Applicable
- **Employer Code**: Not Applicable
- **Job Description**: Not Applicable

### Additional Information

- **Ethnicity**
- **Date Of Birth**
- **Contact Information**
- **Predominant Language**
- **Birth Certificate**
- **Visa Status**
- **Gender**
- **Race**
- **Socio-economic Status**
- **SPP**

---

53
<table>
<thead>
<tr>
<th><strong>DAW Student Academic Information</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select student name for viewing</strong></td>
</tr>
<tr>
<td>- Name: Abdelmessih, Victor</td>
</tr>
<tr>
<td><strong>Program Info</strong></td>
</tr>
<tr>
<td>Admission Date:</td>
</tr>
<tr>
<td>Program History:</td>
</tr>
<tr>
<td><strong>Course Info</strong></td>
</tr>
<tr>
<td>Total Units:</td>
</tr>
<tr>
<td>Courses:</td>
</tr>
<tr>
<td>Pre-requisites:</td>
</tr>
<tr>
<td>Masters Option</td>
</tr>
<tr>
<td>Masters Option: Undecided</td>
</tr>
<tr>
<td>Topic:</td>
</tr>
<tr>
<td>TA/RA Info</td>
</tr>
<tr>
<td>TAs:</td>
</tr>
</tbody>
</table>
### DAW Course Create Form

**Enter new course information**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course Number</td>
<td>Create</td>
</tr>
<tr>
<td>Type</td>
<td>Core Course</td>
</tr>
<tr>
<td>Quarter</td>
<td>Course</td>
</tr>
<tr>
<td>Offered Course</td>
<td>Description</td>
</tr>
<tr>
<td>Course Units</td>
<td></td>
</tr>
<tr>
<td>Units</td>
<td></td>
</tr>
</tbody>
</table>

**Previous Info**

---

---

---
Select Course number from the following

Course Number: CSCI 125

[Delete]
<table>
<thead>
<tr>
<th>Course Number</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS125</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Course Type</th>
<th>Core Course</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Course Offered</th>
<th>Course Description</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Units</th>
<th>Previous Info</th>
</tr>
</thead>
</table>

**DAW Course Information**

*Select course number for viewing*

- Course Number: CS125
- Course Type: Core Course
- Quarter Offered: 
- Units: 

**Reports**

- Student Queries
- Course Queries
- Email/Phone listing
- E-mail listing Only
- Address listing

**Administration**

- Login
- Create Login
- Delete Login
- View Login

**Maintenance**

- Back-Up
- DAW
DAW Report Student Query

Display All students with GPA and Program Status

Display Pre-requisites? Yes

Display
DAW Report Course Query

Display All courses with Course Type

[All Courses]

Display
DAW Login Create Form

Enter new login information

Username: ________________________
Password: ________________________
Access: Administrator

Create
DAW Login Information

Select username

- Username: administrator
- Password: 
- Access: Administrator

---

- Student
  - Create
  - Delete
  - General info
  - Academic info

- Course
  - Create
  - Delete
  - View

- Reports
  - Student Queries
  - Course Queries
  - Email/Phone listing
  - Email listing Only
  - Address listing

- Administrative Login
  - Create
  - Login Delete
  - Login View

- Maintenance
  - Back-Up
  - DAW
DAW Backup Form

Enter directory to store back-up files
Directory

Backup
Bibliography