CPSC 471 Final Report

Timothy Mealey, Ben Roberts, Cory Jensen, Scott Saunders April 9, 2017

Abstract

The current process for creating a class schedule is to go through many periferal websites to acquire all the information needed to make a logical decision in semester schedule building. This process is distrupted with the introduction of the Dauwtrappen Website, which allows you to get all the required information, and create a schedule, in one internet browser tab.

Introduction

Problem

The existing tools for planning semester schedules are decentralized, making them difficult to use together. When building schedules with these tools, it is common to have more than four tabs open in one's internet browser, which is an unnecessarily complicated state for such a simple task. The goal with this project was to simplify the semester planning process by creating a single tool that encompassed the functionality of the UofC calendar, myUofC course search, degree navigator, and the UofC schedule builder.

Implementation

A course search feature was built which displays course names, descriptions, prerequisites, and whether they are offered in any upcoming or current semesters. It is easily navigable using a searchable sidebar tree organized by faculty, department, and course number - selecting any of which loads that node in the main page. Selecting a node also updates the browser URL to a shareable link, without reloading the page, so that students can easily share their course planning with each other.

Similar to the courses page, the schedule page contains a sidebar tree from which one can choose courses. Differing from the courses page, this sidebar is organized by semester, using a dropdown menu, before dissecting into the different course branches. When one selects a course from the tree, its lectures, tutorials, and labs are added to a sidebar on the right. These sections can then be added to the calendar, located in the centre of the page, from this right side bar. After adding the desired sections to the calendar, the user can save their progress as a schedule.

A login system was started, from which users can be added to the database using an email and password. If a user exists in the database, they can log in using their unique credentials.

Design

Users

1. Course Browsing

A user interested in courses offered by the University of Calgary can navigate to the Dauwtrappen website. Once on the website, a series of SQL queries are executed which load the entire course list. These queries are executed in alphabetical order, by faculty, so that the user can begin interacting with the system before all of the data is loaded. Once some or all of the course data is loaded, the user can either browse through the courses using the course tree or perform a search for preferred courses by department, course numbers, and course names. Courses that fulfill their selection/search criteria display the corresponding course information such as their name, number, description, prerequisites, and when/whether they are offered.

A user would use the Dauwtrappen website, as opposed to the other course viewing websites, for the following three reasons: Firstly, the intuitive search tool. Secondly, the Dauwtrappen website has information about when courses are offered alongside the course descriptions. Thirdly, the overall integration of all aspects of the Dauwtrappen website, namely the schedule builder alongside the course viewer.

2. Schedule Building

   A student, or perspective student, can navigate to the schedule building page of the Dauwtrappen website to get organized for the upcoming semester. Again, a series of SQL queries are executed, loading the course data in a cascading fashion. The user can browse through the list or use the same search tool that is present in the course selection. The user can then click on a course to add it to the selected courses sidebar,located on the right side of the screen. This sidebar will then display the various lectures, tutorials, and labs available for that course, as well as their corresponding times. The user can individually click on these sections to add them to their schedule. To remove a selected course, the user can click the X button on that courses section in the sidebar. This will remove it from the selected course sidebar and from the schedule itself. Once the user completes their schedule, they can save it to the database by clicking the save button. They can give it a name if they wish, but this is not required. They may also load schedules from the database by pressing the load button and selecting the schedule they wish to load.

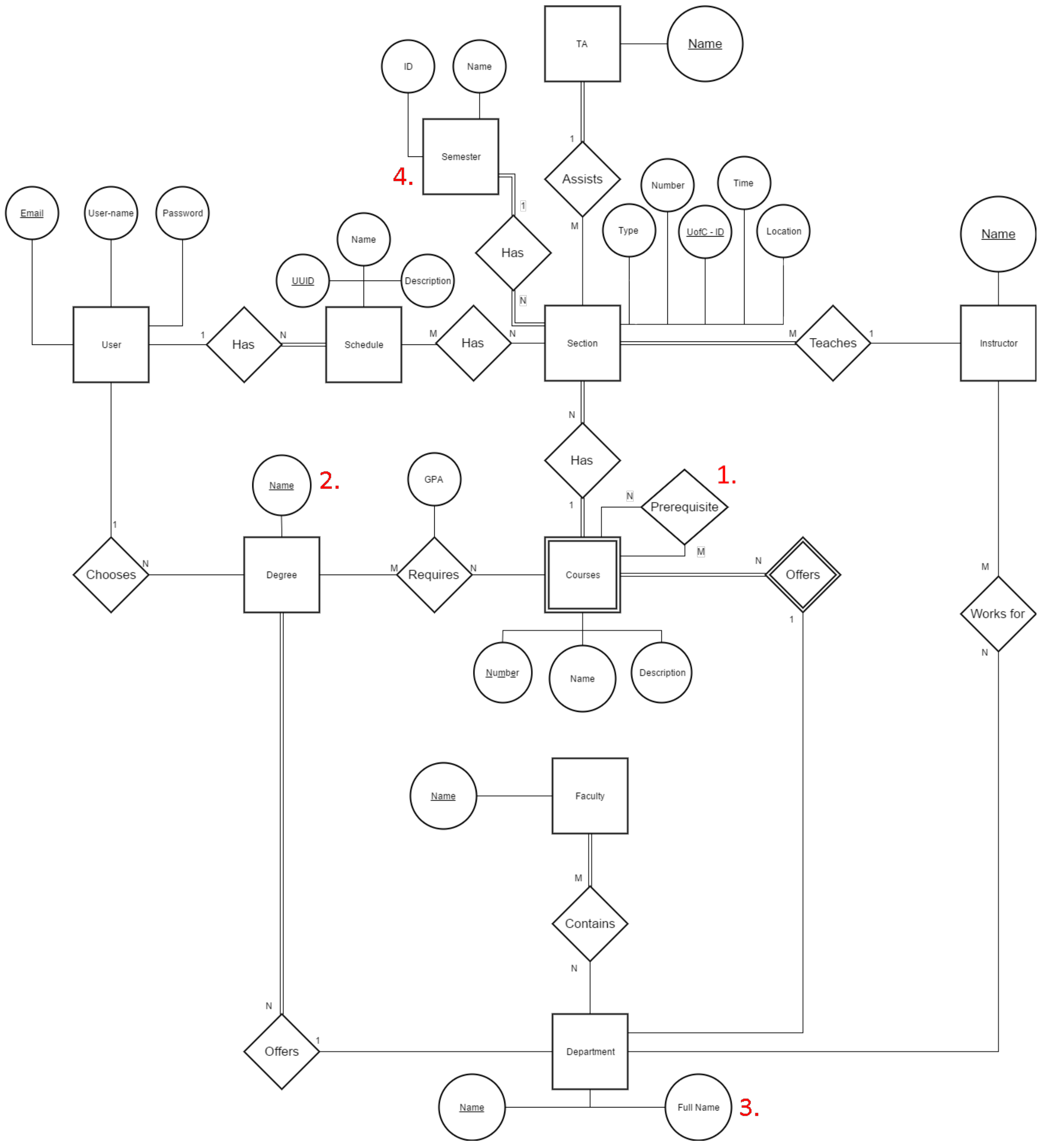
   Currently the user login system is not session based, so all schedules are accessible by all users of the system.

3. Login System

   A user who wishes to login to the website navigates to the login page. Once there, if they already have an account, they can enter their email and password, and then press the login button. This executes an SQL query to confirm that the correct information has been entered. If they do not have an account, they can enter their desired email and password, and then press the create account button. This will create an entry in the database table for that user, enabling them to login to the system.

   The website does not currently have sessions, so logging in does not carry over to the courses or schedule page.

Entity Relationship Diagram



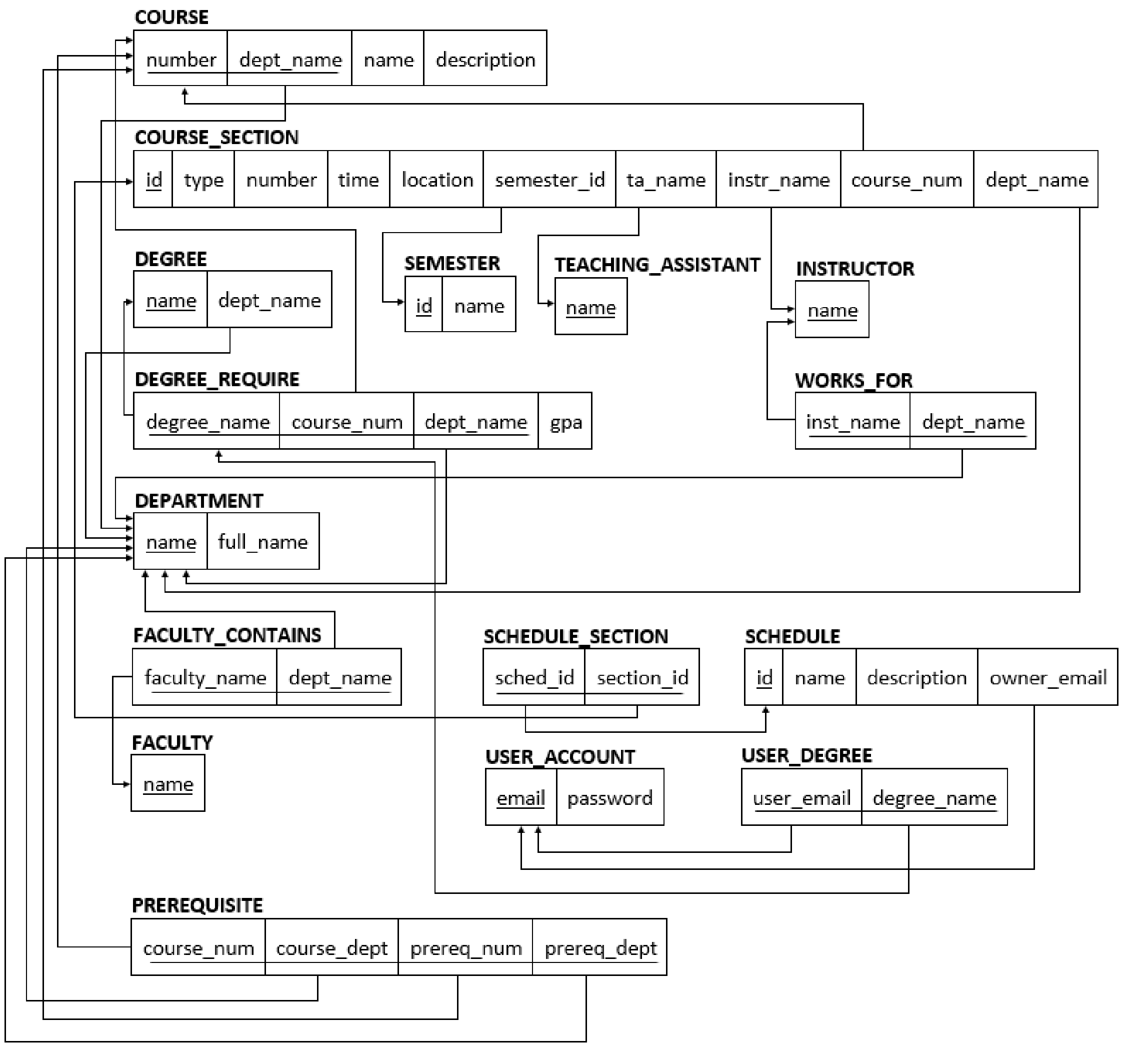
The red numbers on the diagram mark the changes from the initially presented diagram, and correspond to the list below.

1. A prerequisite relation from Course to Course was added, as certain courses are needed before enrollment in other courses is allowed.
2. Removed the "Course Requirements" attribute from Degree because it was redundant. It's function is handled by the Requires relation from Degree to Course.
3. Added the "Full Name" attribute to Department because departments have a short code-name as well as a full English name. For example, "CPSC" and "Computer Science."
4. A Semester entity type was added, and the Has relation from Semester to Section. This is needed to indicate when courses are offered.

Implementation

Relational Schema Diagram

Below you will find the relational schema diagram, which was created using the algorithm, described in class, that converts ERDs to RSDs. There is a tuple for each entity type and relationship from the ER diagram, with arrows pointing from foreign keys to the primary keys they reference.



During the conversion from the ERD to the RSD, redundancies were found and required information was missing, leading to the changes that are described above the diagrams. There were also some constraints put on us by the DBMS that was used. For example, the "User" entity was renamed to "user_account" because "user" was a reserved keyword.

Database Management System

PostgreSQL is the DBMS used because it is a full-featured SQL implementation with good support and documentation. Here are the SQL queries that were written:

Get a list of all the semesters and the faculties they contain:

```
SELECT s.*, f.name AS fac_name
FROM faculty AS f,
  faculty_contains AS fc,
  course_section AS c,
  semester AS s
WHERE c.semester_id=s.id AND
  c.dept_name=fc.dept_name AND
  f.name=fc.faculty_name
GROUP BY s.id, f.name
```

Gets a list of all the semesters:

```
SELECT s.* FROM semester AS s
```

Gets a list of all the course sections from a faculty in a semester:

```
SELECT DISTINCT s.*, c.*,
  fc.faculty_name AS fac_name,
  d.full_name AS dept_full_name
FROM course AS c,
  faculty_contains AS fc,
  course_section AS s,
  department AS d
WHERE s.semester_id=<SEMESTER_ID> AND
  s.dept_name=fc.dept_name AND
  c.number=s.course_num AND
  c.dept_name=s.dept_name AND
  fc.faculty_name=<FACULTY_NAME> AND
  d.name=s.dept_name
```

Gets a list of all the course sections from a faculty:

```
SELECT DISTINCT s.*, c.*,
  fc.faculty_name AS fac_name,
  d.full_name AS dept_full_name
FROM course AS c,
  faculty_contains AS fc,
  course_section AS s,
  department AS d
WHERE s.dept_name=fc.dept_name AND
  c.number=s.course_num AND
  c.dept_name=s.dept_name AND
  fc.faculty_name=<FACULTY_NAME> AND
  d.name=s.dept_name
```

Checks if a user exists in the database with a given email and password:

```
SELECT email, password
FROM user_account
WHERE email=<EMAIL> AND password=<PASSWORD>
```

Checks if a user exists in the database with a given email:

```
SELECT email
FROM user_account
WHERE email=<EMAIL>
```

Adds a user account to the database with an email and password:

```
INSERT INTO user_account
VALUES (<EMAIL>, <PASSWORD>)
```

There are many more INSERT queries in the various scrapers and test-data scripts that populate the database with course information. These have been left out to shorten this document.

User Interface

"Present a brief description of your interface design, including several screenshots."

Every page is easy to navigate to, with there being only a small amount of pages to go to for users to create a schedule, gather information or login/create an account. The information for every course is initally displayed when a user tries to search for a course, which are sorted alphabetically by department and then are stored in their department titles as a node. This enables users to quickly find course names, with descriptions, in a short amount of time.