Data Warehousing from the Web

Chris Fernandes and Michael Whalen
Department of Computer Science
Union College
Schenectady, NY 12308

Abstract

Data warehousing is the ability to collect information from various data repositories and combine them into a single structured repository that can be queried for new information such as performance trends, decision modeling, predictions, and association rules. Internet web sites are data repositories containing useful but unstructured data. In this paper, we describe a data warehouse, developed from the registration web pages at Union College, which allows faculty and students to get on-line access to course enrollment trends, classroom availability, student class schedules, and other pertinent information. The results of this project were so successful in the type of information that could be obtained that the administration became concerned about student privacy issues.

Introduction

Traditional database systems, such as those used by bank tellers, librarians, and airline reservation assistants, are often characterized as online transaction processing (OLTP) systems. They are required to process frequent queries, usually in real-time, that request information about the current status of specific objects and events, such as a bank account balance or availability of a library book. As the status of these entities change, an OLTP system must update the database to reflect these changes so that the database always represents a snapshot of the current state of the world. On the other hand, an online analytical processing (OLAP) system is a database that keeps track of historical data and processes more complicated queries involving summaries and trends rather than individual entities. Table 1 summarizes the two systems.

A data warehouse is a common OLAP system in use today. Retail stores use them to keep track of buying trends. This enables them to stock inventory more accurately. The National Basketball Association uses a system called Advanced Scout to record details about games and extract patterns such as the effectiveness of certain players when on the court with another given player [1]. In general, the creation of the data warehouse is a crucial first step in *data mining*—the process of extracting useful associations to facilitate managerial decision-making.

In recent years, the Web has become a popular source from which to form a data warehouse. It contains a great deal of easily accessible raw data with its main drawback being that it is unstructured. Creating a warehouse from a subset of it would solve that problem and permit analytical queries to be issued upon it.

OLTP	OLAP
Frequent queries	Infrequent complex queries
Short-term analysis	Long-term analysis
Queries both read and update database	Queries are primarily read-only
Database state represents a snapshot of the current status	Database state represents historical, summarized data over time
Queries relate to specific entities and events	Queries relate to general trends and statistics

Table 1: Contrasting OLTP and OLAP

A Warehousing Project

At Union College, every Computer Science major in the senior year must undertake an independent capstone design project. Its purpose is to encourage the student to bring together concepts from his/her coursework and allow the student to do independent research on a subject of interest. One student's idea for a project stemmed from a request from a professor in the Computer Engineering department. The professor lamented about the difficulty in arranging group meetings with her four advisees. It was usually done over email and took several days since most initial suggested times conflicted with at least one student's classes. She wanted an application that could have access to students' schedules and suggest meeting times based on common availability. While commercial applications for this very task do exist (products by Meeting Maker, Inc. for example) she believed it would make a good senior project.

The student (hereafter referred to as the author) came up with a project design that relied on the way Union's registrar provides information to the campus community through its web pages. At Union, both faculty and students have access to a set of pages that lists every department's course offerings by term. The pages list complete information about the courses including course number, meeting time and place, instructor, and many other details. A screenshot of such a page is shown in Figure 1. In addition, users via a link can see a roster of students currently registered for any given course. Such a roster includes each student's full name, major(s), and advisor(s). The pages are automatically regenerated nightly during registration periods so no roster is more than 24 hours out of date.

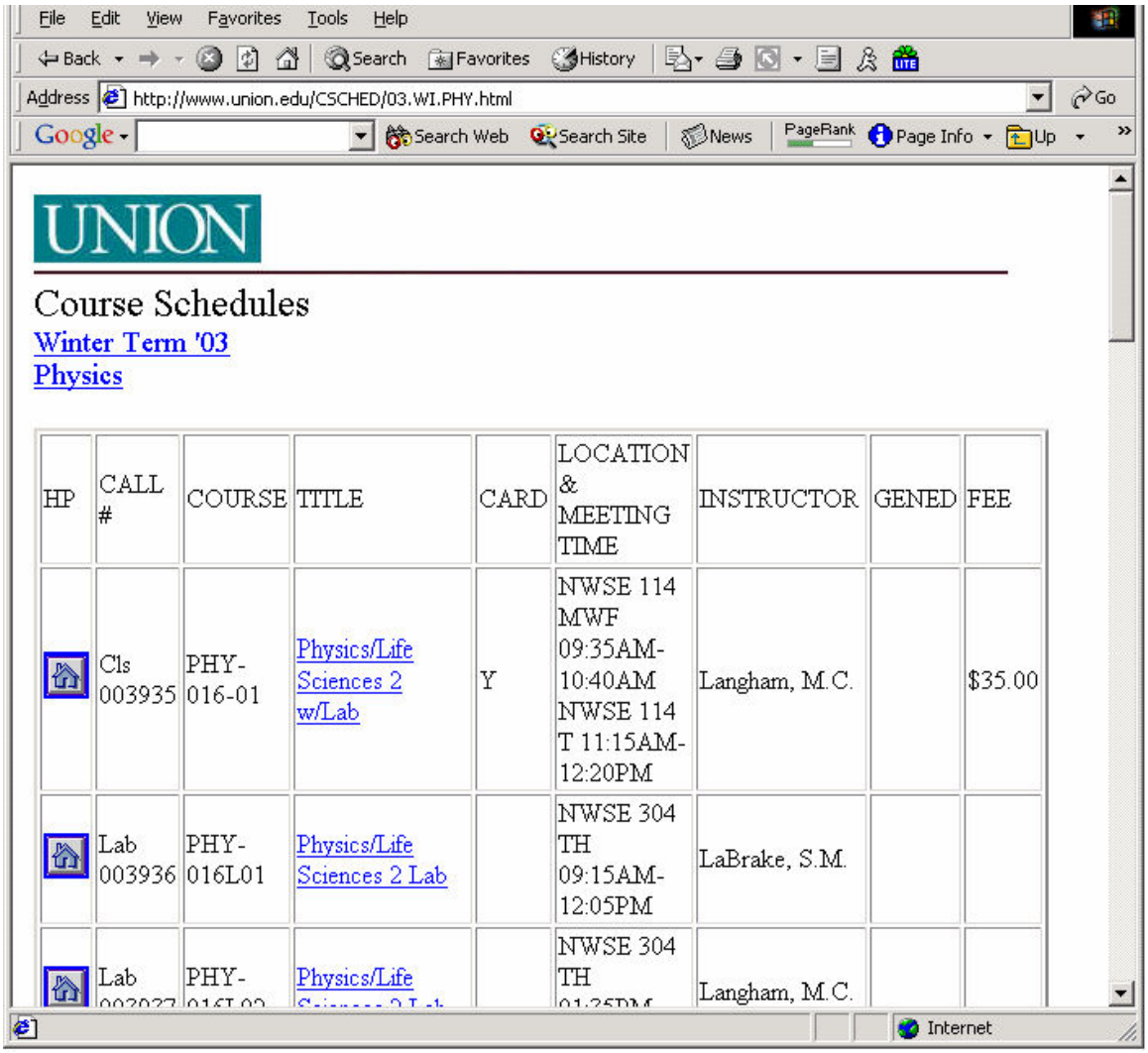


Figure 1: Example of a Registrar Scheduling Webpage

While the registrar's own database is not accessible by any student, these web pages are accessible. By retrieving the complete roster information of every class in a term, a new database could be built and queried to obtain any student's class schedule. This would solve the meeting scheduling problem. However, at the request of his project advisor, the author decided to expand the project into a data warehouse where all of the data listed in the registrar's web pages would be collected, parsed, and organized. This back-end component, called SCOUR (Search Contents Of Union's Registry), would be a completely separate module from whatever front-end interface that wished to use the data. Thus, the project was designed in a way that promoted the reusability of the warehouse since the SCOUR engine is a stand-alone component not dependent on any particular purpose for the data. To emphasize the versatility of the back-end, the author then created a series of interlinked front-end web pages that queried the warehouse and presented results in a variety of ways. This set of front-end pages, called USE (Union Scheduling Environment), issued queries relating to enrollment, classroom use, and other

student-pertinent and faculty-pertinent issues, including the original meeting scheduling problem.

SCOUR

The SCOUR engine consists of a database created with the mySQL database management system (DBMS) and a Java application to crawl through, extract, process, and store the roster information. SCOUR works by taking advantage of the fact that each of the registrar's web pages showing department course offerings has a uniform URL that can be dynamically generated from the current term, year, and department. Thus, SCOUR automatically produces the list of web pages from which it must extract information. The extraction module automatically runs each night, after the registrar's own database is updated. SCOUR downloads the HTML code that comprises each of the roster web pages and separates the formatting information from the content. The content is then systematically transferred to the database, as shown in Figure 2. Any cleansing of the data is done at this stage. For example, if a student was double majoring, the associated advisors would become two separate entries in the warehouse, though they do not appear as such in the source data. This nightly routine only updates the current term's rosters and does not update previous terms (which should not be changing anyway.) The process takes the system offline for approximately ten minutes. In the event of catastrophic failure, SCOUR can extract data from all terms and completely recreate the warehouse. This involves crawling through three years of courses in approximately 35 departments and programs and takes approximately one hour.

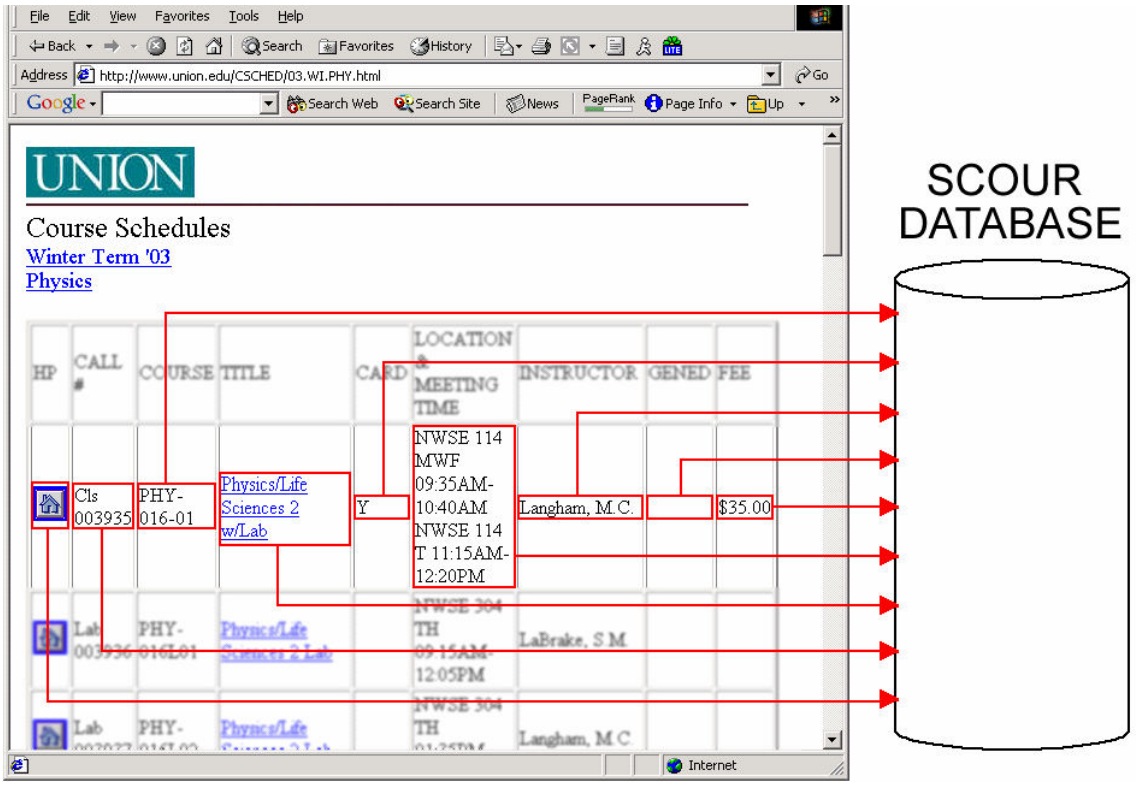


Figure 2: Webpage content extracted to SCOUR

USE

The front-end to the SCOUR engine was designed to be accessible to users without previous database or data warehouse experience. As previously mentioned, a variety of queries were implemented to illustrate the modularity of the engine. Each query was implemented as a separate web page. For example, the query that allows a user to see the availability of any classroom on campus is shown in Figure 3. Note that, wherever possible, form elements are dynamically instantiated with data from the warehouse. To initiate the classroom availability query, for example, the user chooses from a menu of all available classrooms. Instead of a static classroom list, USE automatically generates the choices in this menu from the warehouse by querying it for all rooms currently (or historically) in use by classes. The choices presented by that menu automatically account for any new classrooms simply by the roster source pages listing that room as a meeting location. This minimizes upkeep and ensures freshness of data. Similar dynamic form elements are employed throughout the system.

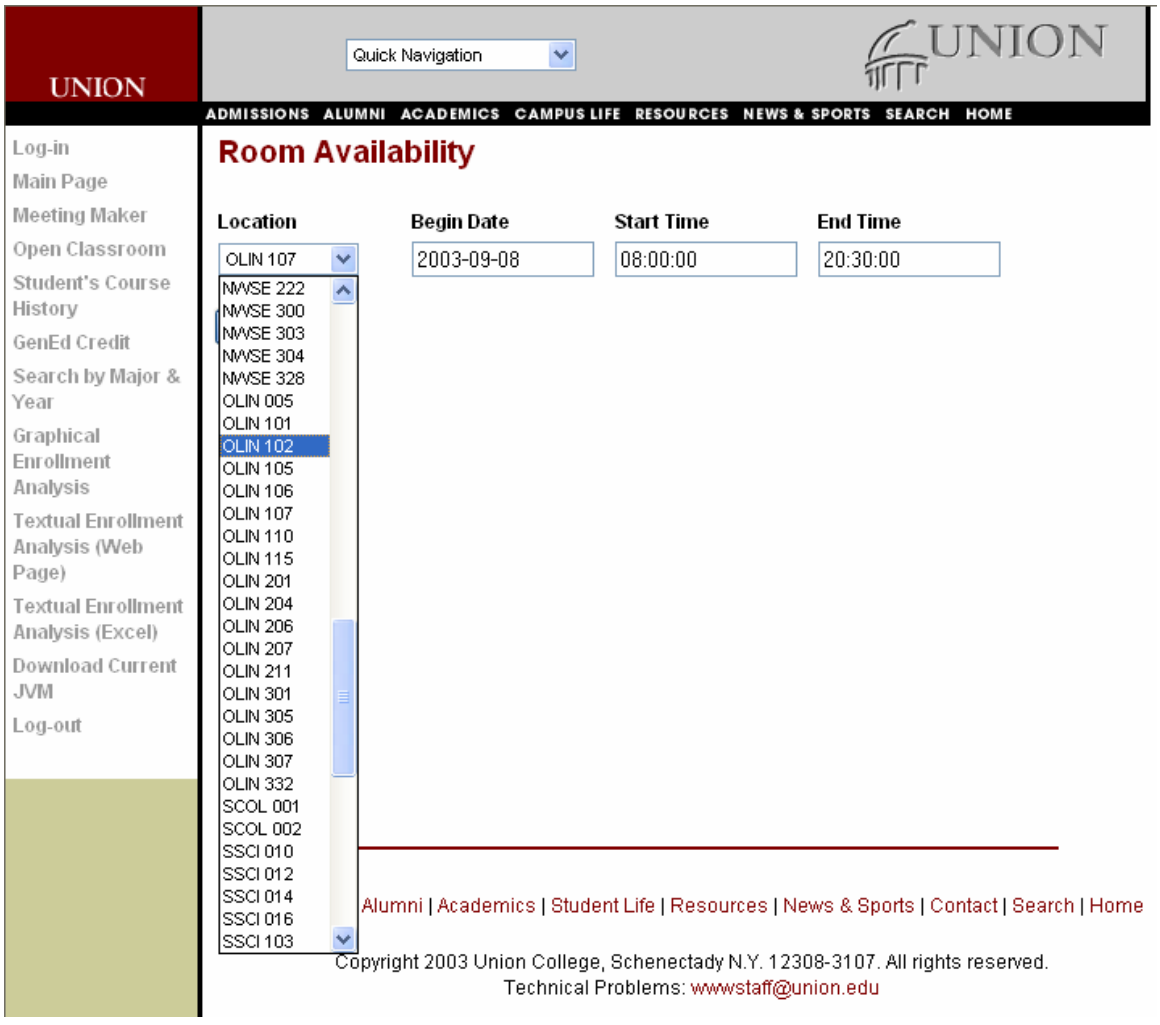


Figure 3: Example query page with dynamically generated form elements

Sample results of the room availability query are shown in Figure 4. These results are applet-based, thus allowing a variety of visualizations. As another example, Figure 5 shows a sample result from the meeting scheduler query. The result is a seven-day schedule where different shadings indicate the unavailability of students. White indicates a time of common availability. Colored areas ranging from light yellow to red indicate the number of conflicts the students have with that time, with red meaning that most, if not all, of the students are unavailable at that time slot. Note that this result does not guarantee a full solution to the original problem since extracurricular student events would not be in the warehouse. However, it does give the query facilitator a useful starting point. In actuality, the author did include space in the warehouse design for extracurricular events, but that aspect was not utilized in the SCOUR prototype. In addition to this scheduling histogram, other visualizations currently supported by SCOUR include text tables, line graphs (Figure 6), and spreadsheet imports (Figure 7). In theory, any visualization is feasible.

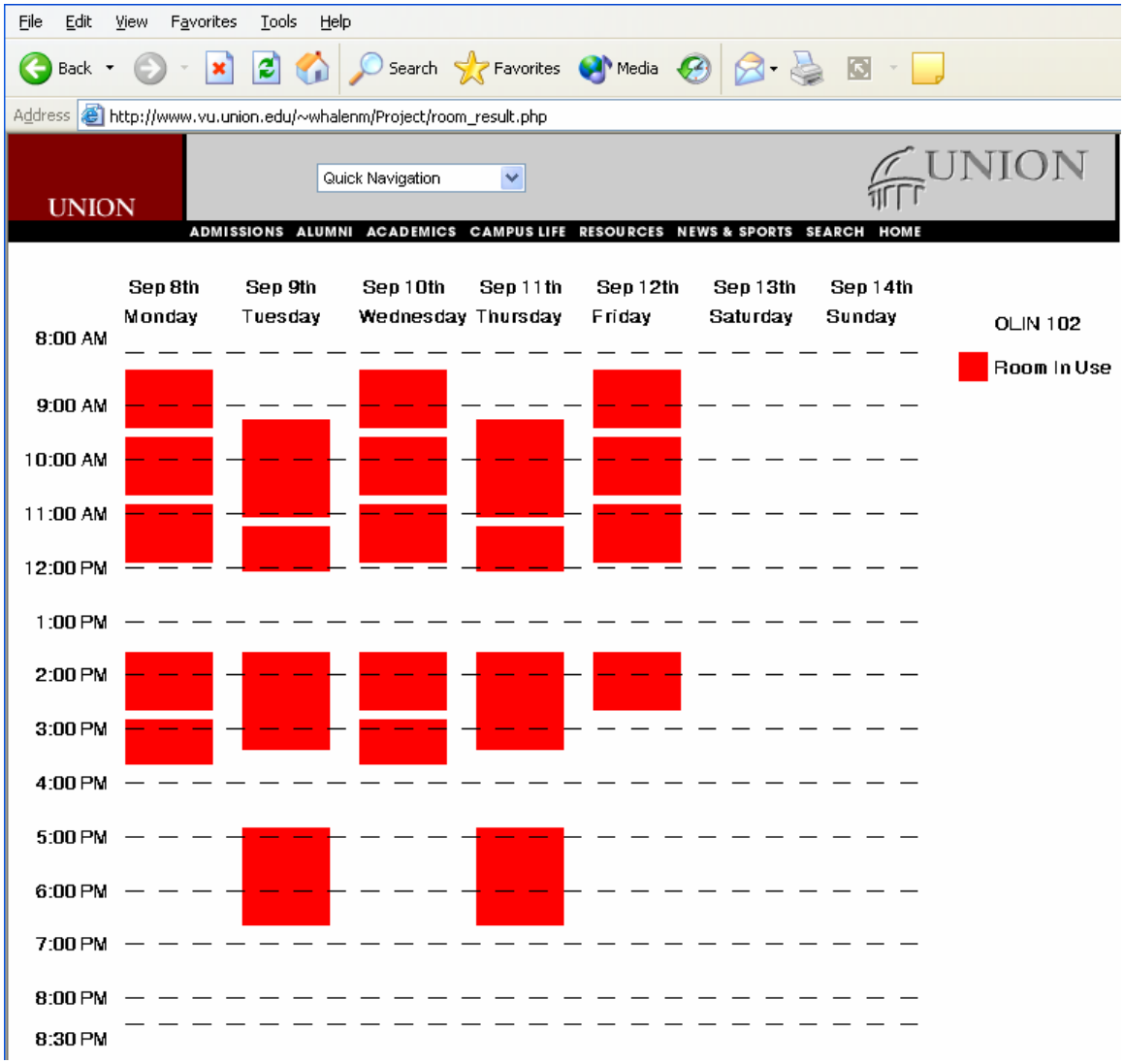


Figure 4: Example results from room availability query

USE has several features to ensure security and privacy. When starting a session, a user must login with a username and password. The password is stored in an encrypted form in the warehouse. Once logged in, the user is assigned one of three security levels: Student, Teacher, or Administrator. At the Student level, only a limited number of queries are available. In addition, some of the available queries are limited in their functionality. For example, a query that allows a student to see his/her course history (all courses taken so far) can only be run for the student himself/herself. The student cannot ask about other students' histories. At the Teacher level, all queries are available and all functionality is restored. A user at the Administrator level has complete access to queries and also has access to USE's administrative front end. At this special web page, the administrator can update the current term to control what data SCOUR should refresh nightly. The administrator can also set the security level of each query which controls who has access to it. Since each query is a separate web page with a separate URL, USE automatically checks to ensure that any page accessed was done so via an appropriate link. This protects against anyone attempting to access a query page directly without going through the login process.

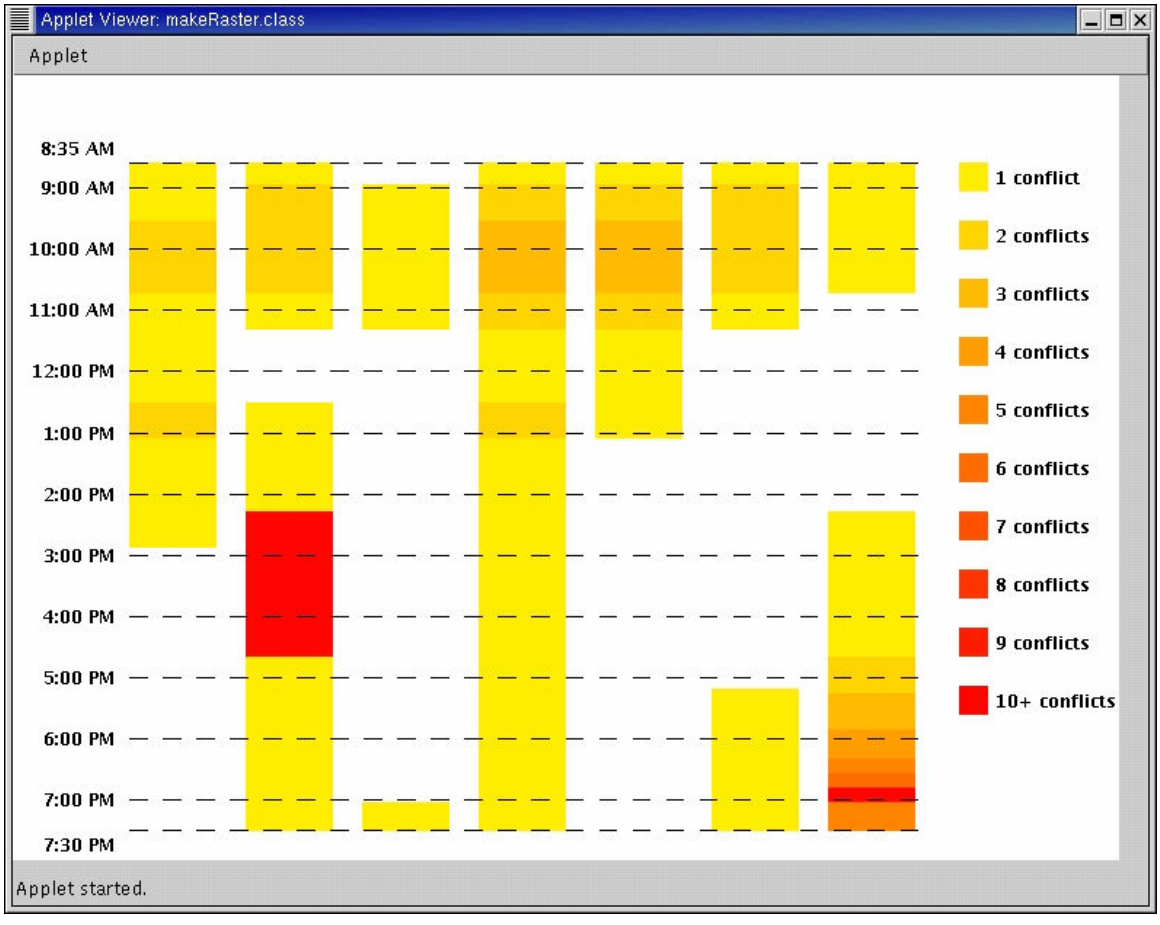


Figure 5: Example results from meeting scheduler query

Results and Discussion

Once the original meeting scheduler query had been implemented, the author implemented several other queries, each with its own web-based front-end, to emphasize the versatility of SCOUR. As the system became more widely known across the campus community, many of these queries grew out of requests from faculty and administrators, including the following:

- Members on a committee for academic affairs were interested in queries to find the number of majors in a given department, including the number of double majors and organizing theme (ad hoc) majors.
- A professor in the Geology department specifically requested a query to list every class taught in a particular year along with professor and enrollment.
- The Dean of Arts and Sciences was interested in the graphical depiction of the enrollment trend for any given class for the past four years. Figure 6 shows an example result of this query.
- The organizer for terms abroad for engineering students wished to automatically keep track of the number of students who went on a particular term abroad and compare this number to previous years. This query was not actually implemented due to time constraints.

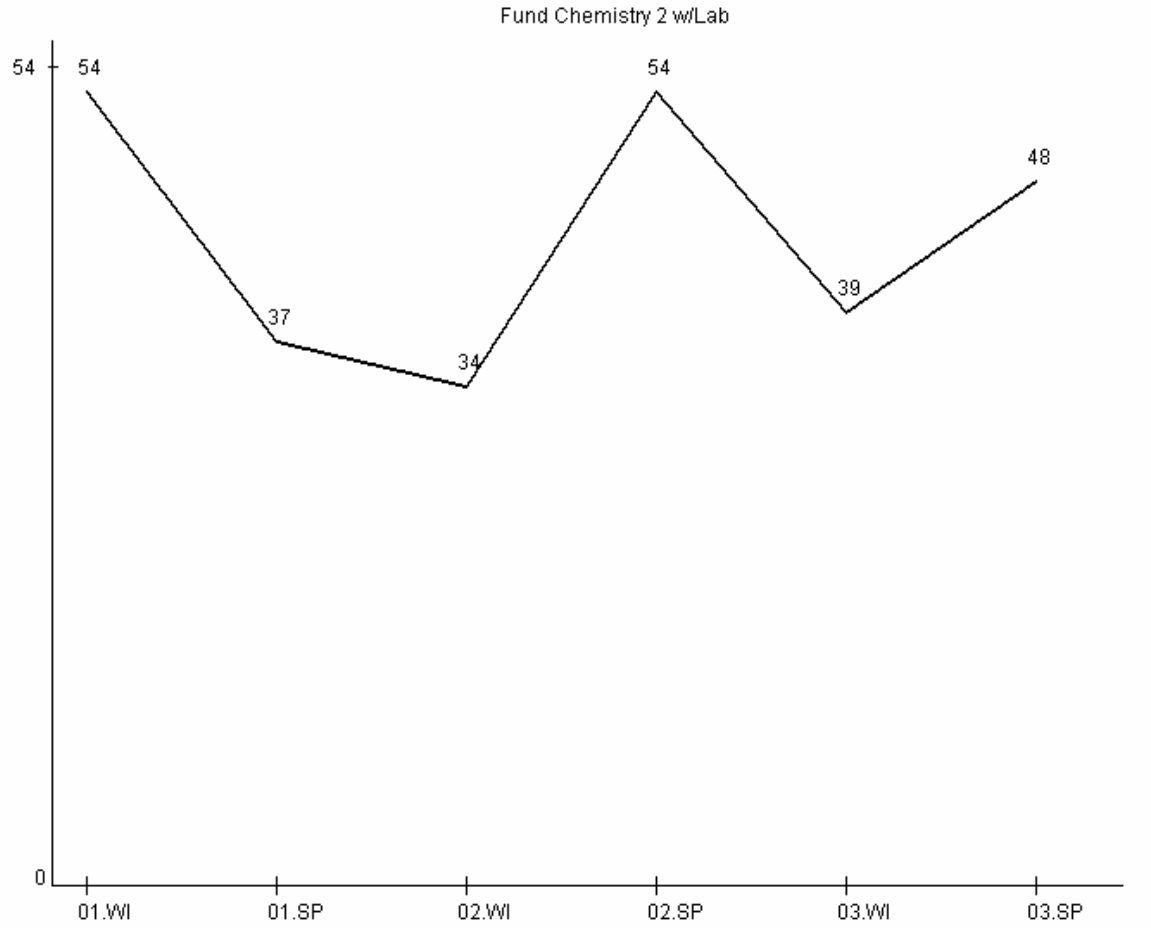


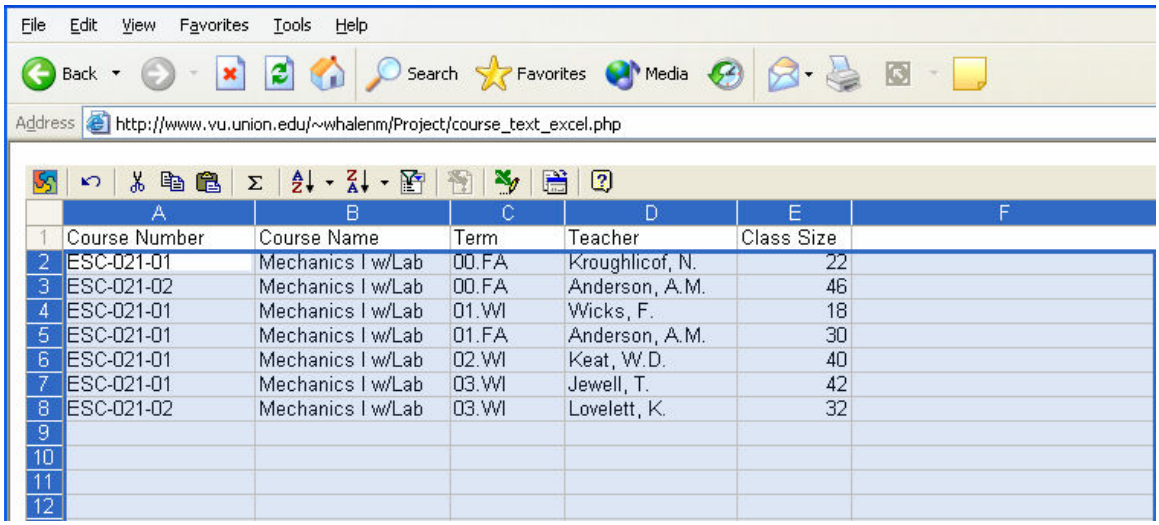
Figure 6: Example results from course enrollment query

- A query that would give a student a complete list of courses attempted at Union was also implemented since online unofficial transcripts are not currently available to students. This query ended up being not quite as useful as the others since the original roster pages do not record last-minute course drops. Thus, the results were not 100% accurate for all students. This is not as significant for other queries where trends are looked at more than data for a single student.

When the potential and usefulness of this system became more apparent, the Dean of Undergraduate Studies offered to fund the author during the summer of 2003 to continue refinements, which the author accepted. He was also given the opportunity to present his work to the college deans, the registrar staff, and the staff of the Information Technology Services (ITS) department.

The meeting with the registrar staff brought up concerns regarding data security. The registrar's own database is on a secure server to which no one has access except registrar personnel. The roster web pages are the primary means of disseminating registrar information to the campus community. With the advent of SCOUR, some of the functionality of the registrar's secure database had been replicated, and the registrar wanted to be sure that security was enforced. It was for this reason that the author subsequently implemented the security features described in the previous section. In addition, permissions were set such that the database was not accessible outside of Union College.

Student privacy was another focus of the registrar meeting. Many schools do not permit roster information of any type to be available to students. Making it accessible has allowed Union students to use it for legitimate purposes such as finding good meeting times for student groups. However, it has also been met with some resistance, and some have complained that a student could potentially be stalked with the roster web pages. Up until now, the response from the administration to such complaints has been that potentially *all* of the roster pages would have to be viewed for such a stalking to take place, and that the effort involved would thwart most



	A	B	C	D	E	F
1	Course Number	Course Name	Term	Teacher	Class Size	
2	ESC-021-01	Mechanics I w/Lab	00.FA	Kroughlicof, N.	22	
3	ESC-021-02	Mechanics I w/Lab	00.FA	Anderson, A.M.	46	
4	ESC-021-01	Mechanics I w/Lab	01.WI	Wicks, F.	18	
5	ESC-021-01	Mechanics I w/Lab	01.FA	Anderson, A.M.	30	
6	ESC-021-01	Mechanics I w/Lab	02.WI	Keat, W.D.	40	
7	ESC-021-01	Mechanics I w/Lab	03.WI	Jewell, T.	42	
8	ESC-021-02	Mechanics I w/Lab	03.WI	Lovelett, K.	32	
9						
10						
11						
12						

Figure 7: Example query results importable to a spreadsheet

students. However, SCOUR and USE takes away this effort. The subsequent implementation of limited query functionality at the Student access level, described in the previous section, attempts to minimize this threat. A student using the meeting scheduler query can only see his/her own schedule. In that sense, the student is no longer able to use the system to find a common meeting time with others but is only able to see his/her own schedule in graphical form, similar to Figure 5. While this visualization is in itself quite useful to a student during the registration process, it is important to realize that the original intent of the query has been significantly altered.

In addition to security and privacy concerns, the meeting with the registrar also raised questions about support. Should SCOUR and USE become widely used across campus, how would the system be supported once the author had graduated? The college was reluctant to use the system campus wide without knowing how new queries could be developed and inevitable breakdowns repaired. The ITS department, already understaffed and overworked with their own projects, was not willing to accept the responsibility of maintaining the author's code. These questions were addressed in the following ways. First, the author had already built into SCOUR the ability to completely recreate the database from scratch should the data become corrupted. Second, the Computer Science department gave tentative agreement that some of their graduate students could be paid to provide upkeep of the code. Not only would this provide a valuable learning experience for the graduate students in helping to maintain a system that was actually in use, but it would not be a further drain on resources since one or two CS graduate students were already being paid for administrative and academic services in and around the department, and this would merely add a small amount to their duties. Third, the author began work on a new function for the Administrator's web page: a query-building engine. This engine would allow anyone with knowledge of SQL (the language in which queries are written) to build a query complete with a web-based front-end. An interactive GUI would simplify the process of building the query interface. Only the query itself would need to be supplied. The use of this page would be well within the abilities of the CS graduate students who would be providing support.

A final meeting with representatives from ITS revealed that a lawyer for the college had been consulted regarding these issues of security, privacy, and support. The conclusion was that students may not access private information about other students, including the raw data that makes up their schedules. Thus, the idea of having CS graduate students provide upkeep and testing of the code and data was not tenable. In addition, since it had now been demonstrated that private information could be easily obtained from existing unstructured Web data, it was decided that the roster pages would be made inaccessible to students, and that faculty would only be able to view it via a login/password scheme. This change has recently been implemented, and since SCOUR relies on the ability to freely access the roster pages to keep its data up-to-date, it is no longer able to perform its nightly downloads. Once a new term begins and students' schedules are different, even the meeting scheduler query, which was the original problem to be solved, will no longer function. The author has since graduated and is now working for an investment firm outside Boston.

Conclusions

The unstructured nature of the Web and the diverse types of data that it contains provide many opportunities for the creation of data warehouses for the discovery of useful information and

trends. At Union, we have found that students have the ability to independently develop such projects and produce interesting results. Matters of security and privacy must definitely be considered, as they make the application of the warehouse problematic, but the endeavor still provides a valuable learning experience.

[1] Bhandari, I., Colet, E., Parker, J., Pines, Z., Pratap, R., Ramanujam, K., *Advanced Scout: Data Mining and Knowledge Discovery in NBA Data*. Data Mining and Knowledge Discovery, 1997.

CHRIS FERNANDES is an Assistant Professor of Computer Science at Union College, having joined the faculty there in 2001. He teaches courses on database design and artificial intelligence. His research interests include semantic interoperability in heterogeneous systems and database interface design.

MICHAEL WHALEN is a Union College alumnus, having graduated with a B.S. in Computer Information Systems in June, 2003. He is the author of the SCOUR project. Michael is currently working in the Interactive department of State Street Global Advisors, an investment firm near Boston.