CPS 196.3 Fall 2002
Course Project
Assigned: Thursday, September 12

Important Dates

Milestone 1: Thursday, October 3
Milestone 2: Thursday, November 14
Final Project Due Date: Thursday, December 5
Project Demo Period: December 9-12

Overview

You have the option of doing either a "standard" or an "open" course project. The "standard" option is to build a database-driven Web application from the ground up. You will receive a fair amount of instructions and examples as well as a few project ideas to help you get started. No prior experience in developing database-driven Web applications is assumed. On the other hand, if you already have considerable experience in developing such applications, you may choose the "open" option and build anything of your liking (subject to approval by the course staff), provided it is related to data management. You will need to write a detailed project proposal and a final report, so expect more work than the standard option. For detailed information and deliverables on each due date, please refer to the appropriate sections below.

Platform Issues

To develop the project you may use the IBM DB2 server maintained by the course staff. The server runs on `rack40.cs.duke.edu`, a Linux-powered Intel machine. You can perform most of your development work on this machine. The course staff plans to support the following:

- Programming languages: Java; Perl.
- Database API's with DB2 drivers: JDBC; Perl DBI.
- Web/application servers: Sun's J2EE server; Apache Web server with Perl CGI.

Example code as well as detailed instructions on how to set up your development environment on `rack40` will be posted on the course Web site.

There are many other ways to develop Web and database applications. If you wish, you may use alternative languages and tools, or run servers on your own machines (regardless of whether you choose the standard or the open project option). Setting up the whole thing can be a rewarding experience, although it may be quite time-consuming. The course staff will only support the languages and tools discussed above.

Teamwork

The project may be completed individually or in teams of two; the choice is up to you. An equal amount of work is expected and the same grading scale will be applied. In general, it is not necessarily easier to work in teams; on the other hand, you will gain valuable teamwork experience by doing that. Both partners in a team will receive identical grade for the project.

Any complaints of the form "my partner did nothing" will not be entertained. Please choose your partner carefully.

"Standard" Course Project

The "standard" project is to build a database-driven Web application. Specifically, you will need to complete the following tasks through the course of this semester. Note that different team members can work on some of these tasks concurrently.

1. Pick your favorite data management application. It should be relatively substantial, but not too enormous. Several project ideas are described at the end of this section, but you are encouraged to come up with your own. When picking an application, keep the following questions in mind:
    a. How do you plan to acquire the data to populate your database? Use of real datasets is recommended. You may also use program-generated "fake" datasets if real ones are too difficult to obtain.
    b. How are you going to use the data? What kind of queries do you want to ask? How is the data updated? Your application should support both queries and updates.
2. Design the database schema. Start with an E/R diagram and convert it to a relational schema. Identify any constraints that hold in your application domain, and code them as database constraints. If you plan to work with real datasets, it is helpful to go over some sample data to validate your design. Do not forget to apply database design theory and check for redundancies.
3. Create a sample database using a small dataset. You may generate this small dataset by hand. You will find this sample database very useful in testing, because large datasets make debugging more difficult. It might be a good idea to write some scripts to create/load/destroy the sample database automatically; they could save you lots of typing when debugging.
4. Design a Web-based user interface for your application. Think about how a typical user would use your site. Optionally, it might be useful to build a "canned" demo version of the site first (i.e., with hard-coded rather than dynamically generated responses), while you brush up your basic HTML skills at the same time. Do not spend too much time on refining the look of your interface; you just need to understand the basic "flow" in order to figure out what database operations are needed in each step of the user interaction.
5. Write SQL queries that will supply dynamic contents for the Web pages you designed for Task 4. Also write SQL code that modifies the database on behalf of the user. You may hard-code the query and update parameters. Test these SQL statements in the sample database.
6. Choose an appropriate platform for your application. Java, Perl, or PHP? J2EE Web server or Apache? MySQL or DB2? Start by implementation a "hello world" type of simple database-driven Web application, deploy it in your development environment, and make sure that all parts are working together correctly. The course staff will provide working examples on supported platforms.
7. Acquire the large "production" dataset, either by downloading it from a real data source or by generating it using a program. Make sure the dataset fits your schema. For real datasets, you might need to write programs/scripts to transform them into a

format that is appropriate for loading into a database. For program-generated datasets, make sure they contain enough interesting "links" across rows of different tables, or else all your join queries may return empty results. If you are using the shared DB2 server on `rack40`, please keep the dataset to a reasonable size.

8. Test the SQL statements you developed for Task 5 in the large database. Do you run into any performance problems? Try creating some additional indexes to improve performance.
9. Implement and debug the application and the Web interface. Test your Web site with the smaller sample database first. You may need to iterate the design and implementation several times in order to correct any unforeseen problems.
10. Test your Web site with the production dataset. Resolve any performance problems.
11. Polish the Web interface. You may add as many bells and whistles as you like, but they are completely optional because they are not the focus of this course.

Milestone 1. You should have completed Tasks 1-5 and started thinking about 6 and 7. You should have a concrete plan on where and how to get the large dataset, although you do not have to be able to load it into your database yet. Submit a progress report (hardcopy; no more than 4 pages) including the following:

- A brief description of your application.
- A plan for getting the data to populate your database.
- A list of assumptions that you are making about the data being modeled.
- An E/R diagram for your database design.
- A list of database tables with keys declared.
- A description of the Web interface. You can write a brief English description of how users interact with the interface (e.g., "The user selects a car model from a pull-down menu, clicks on the "go" button, and a new page will display all cars of this model that are available for sale"). Or, instead, you can simply provide the URL of a canned demo version of the Web site.
- A pointer to your source code, either the URL of a downloadable .zip or .tar.gz archive, or the path to the source code directory on `rack40`. This pointer must be accessible at the midnight of the due date and remain accessible for at least 24 hours. The source code directory should at least contain:
    - A `README` file describing how to create and load your sample database.
    - Files containing the SQL code used for creating tables, constraints, stored procedures and triggers (if any).
    - A file named `TEST-SAMPLE.SQL` containing the SQL statements you wrote for Task 5.
    - A file named `TEST-SAMPLE.OUT` showing the results of running `TEST-SAMPLE.SQL` over your sample database. You can create the file by running "`db2 -tf TEST-SAMPLE.SQL > TEST-SAMPLE.OUT`" on `rack40`.

Milestone 2. You should have completed Tasks 1-8 and have started working on 9. Submit a progress report (hard copy; no more than 4 pages) including the following:

- New assumptions, E/R diagram, and list of tables (if they have changed since Milestone 1).
- A brief description of the platform you chose in Task 6.

- Changes you made to the database during performance tuning in Task 8, e.g., additional indexes created.
- A pointer to your source code. At this point, your source code directory should at least contain:
    - A `README` file describing how to transform the "production" dataset and load it into your database.
    - Code that extracts and transforms (or generates) the production dataset.
    - A file named `TEST-PRODUCTION.SQL` containing the SQL statements you wrote for Task 5. You may wish to modify some queries to return only the top 10 result rows instead of all result rows (there might be lots for large datasets).
    - A file named `TEST-PRODUCTION.OUT` showing the results of running `TEST-PRODUCTION.SQL` over the production dataset. You can create the file by running "`db2 -tf TEST-PRODUCTION.SQL > TEST-PRODUCTION.OUT`" on `rack40`.

Final due date. Submit, through email to the instructor (`junyang@cs.duke.edu`), a pointer to your source code. Again, this pointer should be the URL of a downloadable .zip or .tar.gz archive, or the path to the source code directory on `rack40`. This pointer must be accessible at the midnight of the due date and remain accessible for at least 24 hours. The source code directory should contain a `README` file describing how to set up your servers and database, and how to compile and deploy your application.

Project Demo Period. During this period, you need to present a working demo of your system to the course staff in a 20-minute time slot. Instructions on how to sign up will be given during the last week of the class.

Below is a list of possible project ideas for which high-quality datasets exist. Of course, you are welcome to come up with your own ideas.

1. A movie site where users can look up information on movies, actors, and actresses. A complete dataset is available from `http://www.imdb.com/interfaces`. The data comes in text files that are relatively easy to parse. There are all types of data available, but you probably want to choose only a subset to model in your database. Additional features you could implement include updates to the database and user reviews.
2. A computer science bibliography search engine. You can download the data from `http://dblp.uni-trier.de/xml/` in XML format. You need to map the data into a relational schema and provide search capabilities. Besides regular search features, you can develop more interesting queries using the citation links available in the dataset.
3. A course registration system. You can write scripts to download Duke's schedule of classes from the Web, parse the downloaded pages, and populate your database. Your database should also model students, although you do not need real student data. Students can browse, search, add, and drop courses. It would be nice for your application to check for schedule conflicts, enrollment limits, etc., automatically.
4. An online auction database, a project developed at Stanford for an undergraduate database course. Visit `http://www.db.stanford.edu/~widom/cs145/` for details. A snapshot of the auction data on eBay is available in XML format.

"Open" Course Project

The open option is a chance to build something that you really want, provided it is related to data management. You will need to write a detailed project proposal and a final report. The course staff will work with you to ensure that your project meets the minimum requirements of depth and scope. You are encouraged to build novel systems and tackle challenging problems; if you do, your "risk factor" will be considered in grading. Because of limited time, it is important to stay focused and ensure that certain pieces of your project are completely done; it is difficult to judge a project where nothing works.

Milestone 1. Submit a project proposal (hardcopy; no more than 3 pages) including the following components (not necessarily in this order):

- A description of the problem you wish to solve or the application you wish to develop, and, more specifically, what you plan to demonstrate at the end of this project.
- How it is important, interesting, and/or useful.
- Initial thoughts on how to approach the problem or build the application, including the preliminary system architecture and the platform you plan to use.
- Survey of previous and/or related work, including discussions of how they relate to your problem as well as their limitations and/or flaws.

Milestone 2. Submit a project status report (hardcopy; no more than 2 pages) including the following:

- Changes/updates to your original proposal (if any).
- Summary of progress so far, e.g., components built, tasks completed.
- A list of tasks to be completed before the final due date.

Final due date. Submit a final project report (hardcopy; no more than 8 pages) including the following:

- The problem description, motivation, and survey of related work as in the project proposal, possibly refined.
- A detailed description of the approach you took, including the final system architecture and any design choices you made involving trade-offs.
- Any limitations in your current implementation and thoughts on how you might fix them in the future.
- A pointer to your source code, either the URL of a downloadable .zip or .tar.gz archive, or the path to the source code directory on `rack40`. This pointer must be accessible at the midnight of the due date and remain accessible for at least 24 hours. The source code directory should contain a `README` file describing how to set up your servers and database, and how to compile and deploy your application.

Project Demo Period. During this period, you need to present a working demo of your system to the course staff in a 30-minute time slot. Instructions on how to sign up will be given during the last week of the class.