CSEP 545 Transaction Processing for E-Commerce
Course Information — Winter (January – March) 2012

Instructors: Philip A. Bernstein, philbe@microsoft.com, (425)706-2838
http://www.research.microsoft.com/~philbe/

Sameh Elnikety, samehe@microsoft.com, (425) 538-2234
http://research.microsoft.com/en-us/people/samehe/

Meeting Time: Wednesdays, 6:30 – 9:20 PM, January 4 – March 7.
A one-hour project demonstration, weeks of March 5 and 12 (exact dates TBD)
A take-home midterm exam. Possibly a second exam.

Locations: (live) Microsoft Building 99 room 1915
Overflow with video feed: room 99/1927

(via video at UW) Allen Center, room 305.

If you're attending at Microsoft Building 99:

- Directions: http://research.microsoft.com/en-us/labs/redmond/visit.aspx.
- Please use the multi-story parking garage opposite to Building 99 main entrance.
- If you're not a Microsoft employee and building 99 is locked when you arrive, please wait for a student who is a Microsoft employee to let you in. If you arrive late, call 425-421-5930 (Matt McGinley, who runs our A/V). Or better yet, get the cell phone number of a friend at Microsoft who is also attending the class. Please try to be on-time, since phone calls interrupt the lecture.

Textbook: "Principles of Transaction Processing," 2nd Edition
by Philip A. Bernstein and Eric Newcomer, Morgan Kaufmann Publishers, 2009
Available at the University Bookstore on University Ave (among other places).

Teaching Assistants:
YongChul Kwon, yongchul@cs.washington.edu, http://www.cs.washington.edu/homes/yongchul/
Emad Soroush, soroush@cs.washington.edu, http://www.cs.washington.edu/homes/soroush/

Web Site: http://www.cs.washington.edu/education/courses/csep545/12wi

Mailing Lists: https://mailman.cs.washington.edu/mailman/listinfo/csep545
Please subscribe immediately to this mailing list. Post messages by sending email to csep545@cs.washington.edu. If you're emailing directly to the instructors and/or TAs, please prefix the email subject with [TP].

Lecture Outline

Below is the list of course topics. The exact order of topics may vary slightly from this sequence. Also, the mapping of topics to course meetings is hard to predict because topics are of different lengths and some class time will be consumed by discussions of assignments and projects.

1. Introduction (textbook, Chapter 1)
2. Shadow-based recovery, in support of the project (textbook, Chapter 7, Section 6).
3. Database Concurrency Control, Part 1 (textbook, Chapter 6, Sections 1-4)
4. Database Recovery (textbook, Chapter 7)
5. Basic Application Servers, in support of the project
6. Two-Phase Commit (textbook, Chapter 8)
7. Database Concurrency Control, Part 2 (textbook, revised Chapter 6 Sections 4 - 11)

8. Queuing (textbook, Chapter 4)
9. Replication (textbook, Chapter 10)
10. Business Process Management (textbook, Chapter 5)
11. Application Servers (textbook, Chapters 2 and 3)

Prerequisites

There are no specific prerequisites for the course, besides having a good general understanding of software systems. It's also helpful to have some knowledge of SQL (to be able to write simple queries), and good knowledge of either C# or Java for the project.

Assignments and Grading

There are two components to your grade:

1. There will be a take-home exam, probably handed out on 2/8/2012.
2. You will do a substantial two-person project, described below.

Your grade will be a weighted sum of the mid-term exam (20%) and project (80%). If we decide to give a second exam, then we'll probably redistribute it as 15% for each exam and 70% for the project.

There will be weekly assignments for much of the course, covering material that's easiest to learn by solving structured problems. The solution to each assignment will be discussed in lecture one week after the assignment is handed out. The assignments will not be graded. However, we recommend that you make a serious attempt at solving them, since it's an effective way to learn the material. Some of the exam questions will be variations of some of the assignment questions.

Project

Transaction processing is a systems engineering problem, with many interacting parts. The goal of the project is to deepen your understanding of how the parts fit together. It involves implementing a distributed transaction system in Java or C#. The server will allow customers to make/cancel/query travel reservations, will allow airlines to add/cancel/modify flight information, and will allow hotels and car rental agencies to post room and car availability. We will provide clients, the interface the clients expect, a lock manager, the basic application, and a general design framework for the (ultimately distributed) server. Students will build mechanisms for persistence, recovery, and two-phase commit. The transaction system will concurrently support multiple clients and multiple servers.

Make sure you have a solid working solution of basic features, and save a snapshot of that code, before adding more advanced features. Only a short write-up is required to summarize what you have done and to provide a blueprint of the code, as long as the code is readable. A final demonstration is required.

Except in rare cases with strong justification, people will work in groups of two. It helps you to have design discussions with a partner. And it helps make our reviewing load manageable.

Due dates:

- January 12 – Email message telling who you will be working with and which language (Java or C#).
- January 25 – Milestone One. You should have at least the first couple of steps implemented.
- February 15 – Milestone Two. You should have at least half of the final project features running. Hand that in, with enough of a write-up that we can understand what's in your code and what you have accomplished.
- March 7 – Final Project is due. We would like to view as many of the demos as possible this week.

Milestones are largely for your benefit, to ensure you're pacing the work and to check that you're on the right track. Milestones will be reviewed but not graded. If you do not hand in a milestone, and we later find that you're making some serious design errors, it will be *your* problem for not having given us the chance the flag the error early enough for you to fix it.