CMSC 23500
Introduction to Database Systems

Department of Computer Science
University of Chicago

Spring 2010 Quarter

Dates: March 30 through June 1, 2010
Lectures: TuTh 12:00-1:20 in Ryerson 276
Labs: Tu 3:00-4:20 in Maclab (A-level of Regenstein)
Website: `http://www.classes.cs.uchicago.edu/current/23500-01/`

Instructor: Sharon Salveter
E-mail: salveter@uchicago.edu
Office: Ryerson 150
Office hours: Tuesday 1:30-2:30 pm, Thursday 10-11 am, and whenever else you can find me.

Lab TA: Xueyuan Zhou
E-mail: zhouxy@uchicago.edu
Office: Ryerson 177–c7

TA: Tim Armstrong
E-mail: tga@uchicago.edu
Office: Ryerson 178–d4

Office hours: Open door policy (see page 7)

Contents of this Document

Course description 2
Course goals. 2
Course organization 2
Lab Project 3
Books 5
Grading 6
Policy on academic honesty 6
Asking questions 7
How to hand in weekly homework 7

Course description

This course is an introduction to database design and programming using the relational model. Topics include DBMS architecture, entity-relationship and relational models, relational algebra, relational calculus, functional dependencies and normal forms, web DBs and PHP, query optimization, and physical data organization. The lab section will guide students through the collaborative implementation of a relational database management system, allowing students to see topics such as physical data organization and DBMS architecture in practice, and exercise general skills such as collaborative software development.

Prerequisites

CMSC 15300 and 15400 are prerequisites to this course. These prerequisites will be enforced.

Course goals

This course has the following goals:

1. To introduce students to the fundamental concepts of database design and programming using the relational model.
2. To teach students how to apply these fundamental concepts by designing their own relational databases, using SQL to manipulate and query those databases, and create applications around them.
3. For students to build on the knowledge and skills they gained in previous CS courses by applying them towards implementing a complex software system such as a relational database management system (DBMS).
4. To build and practice collaborative software development skills, such as the use of version control tools, code reviews, and project management.

Course organization

The class meets two times a week for lectures and once a week in a computer lab. The course calendar, including the contents of each lecture, is shown in Table 1.

Homework

Students will have a homework assignment every week. Homework will be available by Thursday morning, and be due at 7:00 pm the following Wednesday. These assignments involve doing exercises related to the topics covered in class so far, including both paper-and-pencil exercises and short SQL programming exercises.

Lab sessions and lab project

There will be a weekly lab session to complement the class lectures. These sessions will be used to present and discuss an ongoing Lab Project that will be developed throughout the quarter (see next Section for details), and to do exercises related to content covered in the lectures. Although no work has to be handed in at the end of lab sessions, attendance to the lab sessions is nonetheless mandatory.

Lab Project

This Lab Project was developed by Borja Sotomayor.

Parallel to the lectures, students will develop a project that will be submitted in three parts. This project involves implementing a relational database management system (RDBMS) from the ground up, starting with file-based B-Tree structures and culminating in a SQL query optimizer. This project will be developed collaboratively first in groups of 2 and then in groups of 4 (depending on class size, the group size may be modified). The three parts of the project are the following:

— **Part I: χdb B-Trees** (due April 16). χdb is a didactic RDBMS, designed for teaching how a RDBMS is built internally. Students will be provided with the specification of the χdb file format, and will have to implement a series of database file and B-Tree manipulation operations in C.

— **Part II: χdb Database Machine** (due May 14). SQL queries in an RDBMS are rarely translated into low-level B-Tree operations. Rather, they are converted into an intermediate language that will be executed by a "database machine" which, in turn, will perform the lower-level operations. In this part of the project, students will be provided with a specification of the χdb architecture, and will implement the χdb Database Machine. They will also be provided with a SQL shell and a SQL parser that they can build on to produce unoptimized intermediate code for testing their Database Machine implementation.

— **Part III: χdb Query Optimization** (due May 28). In this part, students will improve the provided SQL parser to optimize queries based on the algorithms discussed in class.

This project will be developed collaboratively through the department's PhoenixForge site (`http://phoenixforge.cs.uchicago.edu/`). Through this site, students will have access to an SVN repository where they will develop their code. The collaboration will progress in the following stages:

1. Part I will be developed in pairs. Students must let the Lab TA know how they are paired before the lab session on Week 2. Each pair will get their own private project on PhoenixForge.

2. For Part II, students will have a common implementation of Part I so they can develop Part II on even ground. This common implementation will be based on the submitted solutions for Part I. After Part I is handed in, everyone's PhoenixForge project will become *public* and viewable by all students registered in the course. Students will also be given commit access to a shared χdb project on PhoenixForge. By April 22nd, students must commit the

Table 1: CMSC 23500 Spring 2010 Calendar

Week	Date	Lecture	Text Chapters	χdb Lab Project
1	30 March	Course overview. Relational in a nutshell. SQL by rote.		Project assigned
	1 April	DBMS Architecture. Data Models. DDL and DML. Relational Model.	1, 2, 5	
2	6 April	Relational Model. Constraints. Referential integrity.	5	Groups for Part I must be formed
	8 April	Relational Algebra and tuple calculus.	6	
3	13 April	Relational algebra and tuple calculus.	6	
	15 April	SQL.	8	
	16 April			Part I (B-Trees) due 11:59pm
4	20 April	SQL. Views. Aggregation. Triggers.	8, 24.1	
	22 April	Entity-Relationship Model.	3	Shared version of Part I must be uploaded to SVN. Groups for Parts II and III must be formed.
5	27 April	ER $\rightarrow$ Relational Translation.	7	
	29 April	**MIDTERM TEST**	All the above	
6	4 May	Web DB and PHP	26	
	6 May	Web DB and PHP	26	
7	11 May	Query Optimization	15	
	13 May	Query Optimization	15	
	14 May			Part II (DB Machine) due 11:59pm
8	18 May	Functional Dependencies.	10	
	20 May	Armstrong's Axioms. FDs and normal forms. 2NF, 3NF, BCNF.	10	
9	25 May	Decompostion.	11	
	27 May	Transaction Processing. Serializability.	17	
	28 May			Part III (Query Optimization) due 11:59pm
10	1 June	**FINAL TEST**	All the above	

common Part I implementation which will be shared by all students in Part II. Students are expected to review each other's solution, and decide which one should be used, or come up with a consolidated solution that incorporates the best aspects of multiple solutions. There is no formal review mechanism, nor any evaluation of this review. Authorship of the "selected solution" will have no impact on a student's grade, but does confer considerable bragging rights.

3. Part II will be developed in groups of four. Students must let the Lab TA know the composition of the groups before April 22th. Each group will get a new private project on PhoenixForge.

4. Part III will be developed in the same groups as in Part II.

Part I is worth 30% of the lab grade. Part II is worth 50% of the lab grade. Part III is worth 20% of the lab grade.

Team composition policies

1. If the number of students is odd, a single group of three will be allowed for Part I.

2. If one of the students in a pair drops out during Part I of the Lab Project, the remaining student will have two options:

   (a) Complete the project individually. In this case, the Lab TA will reduce the amount of work due.

   (b) Join a pair of students to form a group of three.

3. If the number of students n is not a multiple of four, then $(4 - (n \bmod 4))$ groups of three will be allowed for Part II, with reduced workload.

4. If a student drops out during Part II, Part III, or the Web Project, the Lab TA will reduce the amount of work due by the group.

5. If members of a group feel that a student is not contributing to the group's effort, they may petition the Lab TA to intercede and, only if the group unanimously requests it, to expel the student from the group. The group's workload will be reduced as if the student had dropped out. The student will receive a zero for that part of the project.

6. All other cases will be resolved on a case-by-case basis.

Books

The required text for the course is Fundamentals of Database Systems, 5th Edition, Elmasri and Navathe, Addison Wesley 2007. ISBN 0-321-36957-2. The book will be available for purchase from the Seminary Co-op Bookstore (5757 South University Avenue).

Grading

Grading for the course will be based on: laboratory assignments (50%), homework assignments (10%), midterm (20%), final (20%).

Types of grades

Students may take this course for a quality grade (a "letter" grade), a pass/fail grade, or as an auditor. Students will declare on the final exam whether, depending on their final grade, they want to receive a letter grade, a pass/fail grade or withdraw from the course (a *W* grade). For example, students can declare "If my final grade is a C+ or lower, I will take a *P* (Pass) instead of a letter grade and, if my grade is an *F*, I wish to take a *W*".

> Note: *Students taking this course to meet Computer Science minor or major requirements must take the course for a letter grade.*

Late assignments

Late homework submissions will not be accepted. Late project submissions will not be accepted, and students will be graded based on the version of the code in their SVN repository at the time of the deadline. Mitigating circumstances may be taken into account, but must be brought to the attention of the instructor *before* the submission deadline.

Policy on academic honesty

The University of Chicago has a formal policy on academic honesty which you are expected to adhere to:

`http://www.uchicago.edu/docs/studentmanual/academic_honesty.shtml`

In this course, homeworks and examinations are to be completed without collaboration with any other person, and lab projects are to be completed in collaboration with the members of your group *only*. However, you may discuss with your colleagues the *general nature* of the solutions to homework problems and lab projects. Discussion of class topics, homeworks, and the lab project on the class mailing is particularly encouraged, as long as you do not post solutions (even partial ones) on the mailing list.

Violations of this policy will be reported to the College Dean of Students. In addition, the student will receive a zero grade on the homework or examination. In a lab project, the entire group will receive a zero grade (but only the violator will be reported to the College Dean of Students). If you have any questions regarding what would or would not be considered academic dishonesty in this course, please don't hesitate to ask the instructor.

Asking questions

The TAs for this course have an *open door policy* for asking questions. Instead of setting fixed office hours, you are welcome to consult with the TA at any time. Nonetheless, you should try to give the TA, whenever possible, some advance warning of your visit (by e-mail) to make sure that he will be in the office at that time.

The primary form of support for this course is though the *course mailing list*, which can be used to ask questions and share useful information with your classmates. All questions about homework assignments, lab assignments, discussion/lab sessions, and databases in general should be sent to the mailing list, and not directly to the instructor or the TA (any questions sent directly to the instructor or TA will be forwarded to the mailing list). This way, all your classmates will be able to benefit from the reply to your question. The only exception to this policy is when your question involves revealing part of your solution to a problem, or when a question applies only to you (e.g., questions on course policies, procedures, etc.)

You can subscribe to the mailing list in the following web page:

`http://mailman.cs.uchicago.edu/mailman/listinfo/cmsc23500`

How to hand in weekly homework

There will be two types of homework assignments: "paper and pencil" homework and programming homework. *Both must be submitted electronically through* `hwsubmit` *(described below) and as a hard copy.* The hard copy must correspond to the version submitted electronically.

"Paper and pencil" homework assignments

Although these homework assignments do not involve writing programs, don't take the "paper and pencil" description literally. You must write up your solution in an electronic format such as plain text, Postscript, or PDF (do *not* submit Microsoft Word, OpenOffice, or similar files). Figures may be submitted in PNG or PDF format. Before handing in a "paper and pencil" assignment, make sure you do the following:

1. Submit the electronic version through `hwsubmit` before the due date.
2. Print a hard copy. Staple all pages in order. Make sure your hard copy includes your full name on it.
3. The hard copy must be handed in at the beginning of Thursday's lecture.

Programming assignments

Before handing in a programming assignment, make sure you do the following:

1. Submit your program code through `hwsubmit` before the due date.

2. Print a hard copy of your code. An easy way to print code from a UNIX system is using the `enscript` command. This command will automatically format the code for you, and can handle most programming languages. For example, to print out SQL code, you could run enscript like this:

   ```
   enscript hw6.sql -P printer_name -Esql
   ```

   Where *printer_name* is the printer you want to send the code to. See the enscript man page for more details on using this command, and for a list of languages that enscript can handle.

3. If required by the homework assignment, print a transcript of interactions that exercise your program run on required data.

4. Staple all your pages in order: code first, program output second. Make sure your hard copy includes your full name on it.

5. The hard copy must be handed in at the beginning of Thursday's lecture.

hwsubmit

`hwsubmit` is a UNIX command that will allow you to submit your homework directly to the TA. It is available if you log into any of the Linux machines in the Maclab. Make sure all the files you want to hand in are inside a directory, and then run `hwsubmit` like this (where *dir_name* refers to the directory you want to submit):

```
hwsubmit cmsc23500 dir_name
```

For example, assuming that you are currently inside your home directory, and that you placed all the files for a particular assignment in directory `/home/myusername/hw01`, you would run hwsubmit like this:

```
hwsubmit cmsc23500 hw01
```

Printing code

An easy way to print code from a UNIX system is using the enscript command. This command will automatically format the code for you, and can handle most programming languages. For example, to print out C/C++ code, you could run enscript like this:

```
enscript hello.sh -P printer_name -Ecpp
```

Where *printer_name* is the printer you want to send the code to. See the `enscript` man page for more details on using this command, and for a list of languages that `enscript` can handle.

The `hwsubmit` command is used in other CS courses, so there is a chance you have already used it. If you're unfamiliar with it, don't wait until five minutes before the homework deadline to find out if you're using it right. During the first week of classes, the TA will be glad to assist you in making a "test submission" so you can verify that you're using `hwsubmit` correctly.