COMP 530: Database System Implementation

Description

After you complete this class, you will be able to answer the question: How does one design and build a database system? During the semester, we will discuss database management system architecture, buffer management, query processing and optimization, transaction processing, concurrency control and recovery, data storage, indexing structures, and related topics. Most significantly, students will build a database system from the ground up, that will involve writing at least a couple of thousand lines of C++ code, with a bit of Prolog code thrown in.

If you are a graduate student who has no knowledge of databases, I'll give you a crash course on relational databases so that you can still take the class. If this is you, you'll need to enroll in the 4-credit version of COMP 530. For those enrolled in the 4-credit version, there will be 4 additional, 2-hour, evening lectures (time and location TDB), an additional exam, and an additional programming assignment where you will get some practice writing code in SQL.

Important note: COMP 530 is not simply the graduate version of COMP 430. I'll repeat: COMP 540 is *not* an introduction to database systems from a user's standpoint. That class is COMP 430. If you just want to know how to use a database system, then take COMP 430; COMP 530 is the wrong class. However, if you want to know how to *build* a database, or you want experience designing and implementing a large software system using a "real" system programming language (C++), or you just like building software, this is the class for you!

Prerequisite: Basically, students must be competent at systems programming, and have knowledge of databases.

On the *systems side*, students must have reasonable systems programming skills, be somewhat comfortable using the C language, and the desire to build a large software system. For undergraduates, that means (at least) that you should have taken COMP 321. Note that while we will be using C++, if a programmer is comfortable using C and Java, it is possible to quickly pick up C++. We will also be using Prolog for one of the assignments, but there is no assumption that anyone knows Prolog.

On the *database side*, you must have a basic knowledge of relational databases. For undergraduates, that means you must have taken 430. For graduate students, it means that you should have taken an undergraduate database course, or it means that you should enroll in the 4-credit version of this course.

Credits: 3 or 4. If you are a graduate student and have not taken a class on relational databases, you need to enroll in the 4-credit version of the class.

Instructor: Chris Jermaine (cmj4@rice.edu)

A Note to Students Who Have Taken COMP 430

In the past, COMP 430 was a strange amalgam of “Intro to Databases” and “Database Implementation”. Going forward, the implementation content will mostly be removed from 430. But if you’ve taken COMP 430 at Rice in the past, you’ve seen some implementation before, and some of the material in COMP 530 will be review. But the class will still (hopefully!) be interesting, and there will be a lot of material that you have not seen before. The class project has no relation to any project in COMP 430.

Textbook

My lectures will closely follow the material in Database System Implementation by Garcia-Molina, Ullman, and Widom. The material in that book is also available in Database Systems: The Complete Book, by the same authors. Either one of these books will be valuable, but not necessary.

Lectures, Meeting Times and Locations

Class will be held Monday, Wednesday, Friday from 11:00 to 11:50 in HRZ 210 (plus 4 additional evening lectures for those students taking the class for 4 credits; the location of those lectures is TBD). Aside from the evening lectures for students taking the extra credit hour, all lectures will exclusively be given using the blackboard, old school. Most students enjoy this more than slides. But the downside is that if you miss class, you’ll need to obtain the notes from a friend.

Registration

You’re responsible for registering for COMP 530 with the university registrar. For undergrads, prerequisites will be strictly enforced. For graduate students, I’ll be more open to letting people in (but the 59 student limit is quite firm).

Communication

The class will have a Piazza forum for all day-to-day communication:

https://piazza.com/rice/spring2016/comp530/home

It is expected that if you have a technical question on an assignment or an upcoming exam, you will post it to the forum rather than sending an email to the instructors This guarantees a fast response and means that everyone can benefit from the question and the answer. In general, only inquiries of private or personal nature should be made directly to the instructor ("I need to go out of town on Oct 22nd, can I have an extra day..."). Everything else should be posted on Piazza. You’ll get faster feedback from the group than you can get from your instructors.

If you have any communication of a more personal nature and wish to contact the instructor of the class, please send email to Chris, **and include the word "530" in the subject line**. Please realize that O get a lot of random email, so if you do not include 530 in the subject line, your email will likely be ignored.

Assignment handouts and turnins, as well as your grades, will be on Owlspace. Everything else will be on Piazza.

Grading and Evaluation

If you are taking the class for 3 credits, your grade is based upon a set of seven programming assignments (70% of your grade; each is worth 10% of your grade), and a set of eight, short, in-class quizzes (30% of your grade). Each quiz will take approximately 20 minutes and the date will be announced beforehand. You can drop the two lowest score that you receive on the quizzes; all others are worth 5% of your grade each.

If you are taking the class for 4 credits, there is an additional assignment equally weighted with the rest, and an exam that is worth two quizzes. To obtain your final score in the class, I'll take your average assignment percentage and multiply by 70%, and your average quiz percentage (with the exam counting as two quizzes) and multiply by 30%; the sum of these two numbers is your score in the class.

Your numeric grades will be published to you in OwlSpace.

Final grades are based on the numeric grades, where 90-100 is an A, 80-89 is a B, and so forth. We reserve the right to apply a "curve" to change this, but only for the better. That is, if you've gotten 90%, you're guaranteed at minimum an A- for your final grade, but you might do better.

Assignments

This is an assignment-oriented class. There will be seven programming assignments, all completed completed in teams of two (or alone, if you really prefer). Both students always get exactly the same score on each assignment. Students my switch or leave partners during the first two days that any assignment is out by simply letting Chris know via email.

The approximate assignment dates and due dates for the assignments will be:

A1 Buffer and file management: out Tuesday, Jan 19th, in Friday, Jan 29th
A2 Record manager: out Monday, Feb 1st, in Wednesday, Feb 10th
A3 Sorted file implementation: out Friday, Feb 12th, in Monday, Feb 22nd
A4 B+-Tree Implementation: out Wednesday, Feb 24th, in Monday, March 14th
A5 Query Optimizer: out Wednesday, March 16th, In Wednesday, March 30th
A6 SQL parser and query engine: out Monday, April 4th, in Monday, April 18th
A7 Putting it all together: out Wednesday, April 20th, demos May 2, 3, 4

Lateness and Missed Assignments

Assignments must be turned in by 11:55PM (5 minutes before midnight) on the day that they are due. You can turn in an assignment up to 24 hours late, in which case you receive a 10% penalty (that is, 10 points are subtracted from an assignment that is worth 100 points), or up to 48 hours late, in which case you receive a 20% penalty. Assignments turned in after that are not accepted. Please note that your turnin time is whatever OwlSpace says, and your turnin is whatever you turn into OwlSpace, **no exceptions**. Because we have so many people in the class, no extensions will be given. Be safe; submit early and often!

A missed quiz results in a zero, again: **no exceptions**. I realize that people travel and need to miss class, but that's why I'm dropping the two lowest scores.

We kept on saying **no exceptions**, but there are exceptions in very extreme circumstances, with proper documentation. For example, if you obtain a doctor/dentist note stating that you were so ill at the due date/time that you could not reasonably be expected to meet the deadline, it is possible to get an extension.

Regrade Requests

These must be made within **one week** of an assignment/midterm being returned, during Chris' office hours, to Chris in person. Sending an email does not constitute a regrade request. When you talk to Chris, he'll help you understand whether you've got a legitimate request. If you do, then you'll write that request down formally, print it on paper, and hand it to Chris. Chris will batch these, and them periodically and issue final grade adjustments in bulk for everybody.

Academic Misconduct

In a programming class, there is sometimes a very fine line between "cheating" and acceptable and beneficial interaction between peers. Thus, it is very important that you fully understand what is and what is not allowed in terms of collaboration with your classmates. Our goal here is to be 100% precise, so that there can be no confusion.

The rule on collaboration and communication with your classmates is very simple: you cannot transmit or receive code from or to anyone in the class in any way---visually (by showing someone your code), electronically (by emailing, posting, or otherwise sending someone your code), verbally (by reading code to someone) or in any other way we have not yet imagined. Any other collaboration is acceptable.

The rule on collaboration and communication with people who are not your classmates (or your TAs or instructor) is also very simple: it is not allowed in any way, period. This disallows (for example) posting any questions of any nature to programming forums such as StackOverflow.

As far as going to the web and using Google, we will apply the "two line rule". Go to any web page you like and do any search that you like. But you cannot take more than two lines of code from an external resource and actually include it in your assignment in any form. Note that changing variable names or otherwise transforming or obfuscating code you found on the web does not render the "two line rule" inapplicable. It is still a violation to obtain more than two lines of code from an external resource and turn it in, whatever you do to those two lines after you first obtain them. Furthermore, you should **cite your sources**. Add a comment to your code that includes the URL(s) that you consulted when constructing your solution. This turns out to be very helpful when you're looking at something you wrote a while ago and you need to remind yourself what you were thinking.

Any violations of these rules will be reported to the Honor Council. Just don't do it!

Students with Disabilities

Students with disabilities should contact the course instructor and Disability Support Services regarding any accommodations that they may need.