Chapter 0: Getting Started with Database and Computer Science

Database and SQL: Course Notes | Brendan Shea, Ph.D. (Brendan.Shea@rctc.edu)

Hello, Database Students! This short set of introductory notes provides some general guidance on strategies for succeeding in a class like this. If you have any questions or concerns about the course, please let me know—I'm here to help.

--Brendan

1 Contents

2 How to Succeed in Computer Science Classes

This is an introductory course in computer science (CS), focused on databases and Structured Query Language (SQL). This might be among the first computer science classes you've ever taken. Or, you might already have completed several semesters' worth of prior coursework, or have considerable on-the-job experience. Regardless of your background, it's worthwhile to think about what we'll be doing in this class and how you can get the most out of it.

Like most computer science classes, this one will contain a mix of abstract/theoretical issues with more practical/applied issues:

1. On the more theoretical side, we'll discuss the definitions and properties of key concepts like **databases, data models, relations (or tables),** and **normalization.**
2. On the more practical side, we'll learn the programming language **SQL** and the creation of **entity-relation diagrams**.
3. Many topics, such as database design, involve both practical and theoretical problems.

Many students prefer either the theoretical or the practical side of computer science, which is perfectly fine! However, it's crucial to think about *why* both sides matter. Without knowing something about the "theory" behind databases, it can be (practically!) challenging to figure out why things have gone wrong or to fix things that have broken. By contrast, without knowing something about the "practical" end of things, the *reason* that we study certain theoretical concepts (such as those underlying the "relational model" of databases) can easily get lost. Keeping this in mind can help keep you motivated when (sometime in mid-semester) you start asking yourself questions like, "Why do I have to learn this, anyway?" 😊.

2.1 Focus on learning the material and not (too much) on your grade

There is nothing wrong with wanting to get a good grade in this class, and the overwhelming majority of students who follow the advice on this handout will pass the course with *at least* a C (and many will get As and Bs). However, focusing too much on your grade can be an unneeded source of stress, and can get in the way of your learning the material (and weirdly enough, of getting a good grade!):

1. On some of the "easier" assignments, you might find it easy to get a "good grade" on the assignments, even without doing the reading or looking closely at the problems. In these cases, I *strongly* encourage you to go back and look over the problems again, and make sure you understand how the problem "works" and not just what the "right answer" is. Later material in the class will build on the early material, so it's essential to focus on really understanding the material and not on getting through it as fast as possible.
2. On some of the "tougher" assignments, you might run into an occasional problem that you just can't "crack," even after you've spent 30 minutes (or occasionally, 3 hours) staring at it. This is OK (it happens to professional computer scientists all the time, actually…), and having occasional problems will not doom you to failing the course. In these cases (after giving it your best shot), it might be time to take a break for the day.
3. The grading system for the class is designed to ensure that, if you diligently do your work, you *can* pass the class. Remember that, for the toughest assignments, there is absolutely nothing wrong with getting a 50%. This score doesn't mean "you've failed to learn anything" but rather "you've already learned a lot, even if you haven't learned everything."

2.2 Practice computer science (almost) Every day

Generally, a three-credit college course should take around nine hours per week (including lecture, reading, and homework). For many students, computer science classes (like math or logic classes) tend to take a bit more time than do other sorts of courses. With this in mind, I'd encourage you to plan out the time you will spend on your computer science homework at the beginning of the week (as opposed to trying to "fit it in" whenever you have time). Some general pointers:

1. **Make CS a priority.** Make sure to do CS homework when you are awake and aware, not when you are tired and grumpy (this really makes a difference!). If you've had trouble in "math-y" classes in the past, I'd encourage you to do your homework before doing your homework for other classes, cleaning the house, etc.
2. Getting better at CS requires **frequent practice** (similar to learning to play a musical instrument or practicing a sport). You'll learn much more (and find it much more enjoyable) if you spend an hour or two every night practicing SQL queries than if you sit down and try to do an entire homework assignment four hours before it is due.
3. **Take a break.** A good study session is around two hours, with 5-minute breaks every half hour or so where you get up and walk around (don't spend the breaks getting stressed out on social media!). After this, give yourself 30 minutes to relax before returning to your homework. Trying to write code from 9 pm to 3 am is a recipe for disaster.
4. **Plan your study time.** One good way to get your nine hours if: plan on spending two hours on the class Mon-Fri (with a break thrown in), and then take the weekend off.

One benefit of "spreading out" your work on this class is that it gives your brain time to process all the new things you'll be learning. In many cases, I've found that problems that seemed *impossible* when I went to bed on Tuesday night seem pretty doable by Wednesday morning.

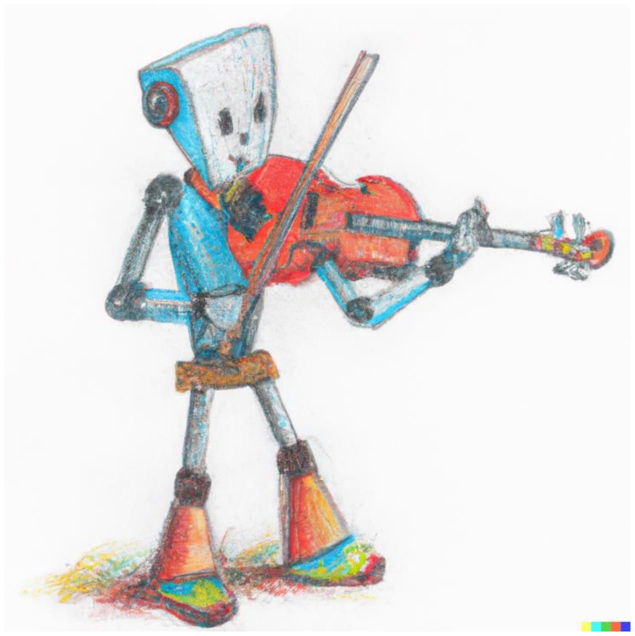
Figure 1 Practice makes perfect (Brendan Shea x Dall-E)

2.3 Do the Homework! Read the Book! Attend Class!

Every teacher you've ever had has probably told you this, but it is necessary to do the homework to succeed in this class. Many assignments and lectures "build" on previous ones, so missing one or two assignments can snowball quickly. This does NOT mean you have to get an "A" on every assignment, but you have to sit down and give it your best shot.

2.4 Practice Active Reading (and Listening)

Reading a computer science textbook (or lecture notes) is NOT like reading a novel or reading a textbook for your other classes. If you try to read without taking notes, you probably won't get much out of it (even if you try rereading it multiple times). Instead, you should plan on reading (or listening) *actively,* ensuring you understand what is going on, and asking yourself *questions* about the text (and/or lecture). Here is my general strategy for reading:

1. **Skim** for general format—what topics are covered? (This will often give you a rough idea of what will be covered on the homework.) How much space is devoted to each?
2. Look for **key terms and concepts**—what terms/concepts are defined? What are the definitions (I often write these down on a separate piece of paper)? Make sure to pay attention to the *specifics* of the definition, and not just to the "general idea." In classes like this one, *every word* of a definition matters.
3. **Read** the textbook chapter in its entirety, paying special attention to how the sample problems are solved. It's *normal* to spend 10 (or sometimes 20) minutes being confused by a sample problem. Just go through it slowly until you start to "get" what is happening.
    a. Studies have found that **rereading** textbook material doesn't substantially increase understanding. It's better just to read once (Actively! Slowly!) than to just go over it again and again.
4. If you get confused about a concept, it's OK to look at Wikipedia or StackOverflow (I know I do!). It can often help to see an idea explained in several different ways.
5. Try to do a few of the **problems** in the textbook on your own, and see if you can get the correct answer (nearly every textbook gives the answers to some of the textbook problems).
6. **Understand. Don't memorize.** In college-level computer science courses, you shouldn't try to memorize "solutions" to problems. Every problem will be a bit different, and you'll have to use your understanding of the concepts to devise your *own* solution to the problems. Once you understand the concepts, remembering them won't be nearly as difficult.

2.5 Ask for Help!

When learning computer science, teamwork plays a big role. Here are some suggestions:

1. When you get confused, ask for help! In many cases, working through problems with an "expert" for 15 or 20 minutes can help you figure out (pretty quickly) what has been going "wrong." A note: It isn't easy to do effective CS tutoring via e-mail. So, I encourage you to meet with me in person or via Zoom. So, attend office hours, or go to the campus learning center (where they have CS tutors).
2. Find a **"study buddy."** While the homework should reflect your own work, there is nothing wrong with getting together with a classmate (or two) to bounce ideas off. Research has consistently shown that one of the best ways of learning new concepts is to (a) listen to your peers' explanations of them (since they are closer to your "level" than the instructor is) and (b) to try to *teach* new things to your peers. Plus, working with other people makes the homework more fun!
3. If something interferes with your ability to succeed (such as illness, family emergency, etc.), let people (such as your instructor, academic advisor, etc.) know! We are here to help!

As a side benefit, students who connect with their instructor/classmates/tutors tend to deal better with adversity (when things get tough, they have someone to talk with) and get better grades overall.

2.6 General Strategies for College Success

To close with, here are two general strategies that should be of use to you in *any* college class:

1. **"Fake it until you make it."** Many students feel they are "bad" at Computer Science (or English, Math, Philosophy, etc.) even when they are doing OK in the course. They think, "it shouldn't be so hard for me!" or "I'm getting the right answer, but I don't understand how!" *This is an entirely normal feeling when you are learning something new.* The fact that some problems are challenging doesn't mean you are missing something—they are *supposed* to be tough, and it's *good* to struggle with them. Just keep doing what you are doing, and it will turn out OK.
2. **Work less. Sleep more. Exercise. Eat Healthily.** While we all know these things on some level, it's important to remember that things like work-related stress, poor sleep, a lack of exercise, or unhealthy diets make a (big!) difference in college. Students who work 25 hours a week and sleep 8 hours a night will find it easier to succeed in college than those who work 50 hours a week and sleep 4 hours a night. Obviously, it's impossible for most of us to be perfect in these sorts of things (life gets in the way!), but there is usually *something* we can do to make things a bit easier for ourselves.

2.7 Review Questions and Activities

Take a few minutes and reflect on your answers to the following questions. (You don't need to submit this to me—this is just for your own use).

1. What are your **learning goals** for this course? What skills or knowledge do you want to gain (besides getting a good grade)?
2. How are you going to structure your studying for this class? What days/times/locations work best for you?
3. If something goes wrong (you get stuck on a homework problem, fall behind because of illness, etc.), what is your plan for addressing it?

3 Reading: How to Get Excited About Topics That Bore You (by Barbara Oakley)

Summary. The ability to develop new skills – and new passions — is particularly important in today's fast-paced business climate. But what if you don't like the subject matter you need to learn? Rest assured that

it *is* possible to learn to like — and even love — subject areas that seem boring, or that you once loathed. The first step in building passion for a subject you don't like is to identify a reason to learn it. One of the best motivators is making an improvement in your life. Wanting a lifestyle upgrade allows you to make a mental contrast between where you are now (say, an office assistant) and where you want to be (perhaps a certified public accountant). Know that it's perfectly normal to *not* understand new subject material on your first try. It takes time and practice for your brain to develop new neural patterns, which are "chunks" of learning. The bigger your collection of solidly constructed neural chunks related to solving different problems, the better your expertise. And the better your expertise, the more you will like what you're learning.

I used to be *the* prototypical young mathphobe: I flunked or barely passed any math course I was forced to orbit. For me, graduating from high school was thrilling in that I would never have to touch a math or science book again. Math not only didn't make sense but was also worthless and painfully frustrating; the same went for technology.

The young version of me would have been shocked to learn that I would eventually become a professor of engineering, enchanted with mathematics and comfortable in the world of technology. As I've discovered from both personal experience and research, it *is* possible to learn to like — even to grow to love — subject areas that look boring or that you once loathed. In today's fast-changing business environment, the ability to develop new passions is particularly important. Here's how to do it.

Find a seed of motivation. The first step in building passion for a subject you don't like is to identify a reason to learn it. One of the best motivators is wishing to make an improvement in your life. Wanting a lifestyle upgrade allows you to make a mental contrast between where you are now (say, in a job as an office assistant) and where you want to be (say, a certified public accountant).

For me, the mental contrast between where I had been, the lowest enlisted rank in the army, and the many civilian career options I dreamed of provided a powerful boost. That's how, at age 26, I found myself restudying remedial high school mathematics in an implausible attempt to become an engineer.

Overcome the pain in the brain. When we even *think* about something we don't like or want, it can activate a portion of the brain — the bilateral dorsal posterior insula — that is involved in our experience of pain. This means, for example, that thinking about math if you don't like math (or studying English if you don't like English) can actually feel physically painful. The result can be that your brain diverts your attention away from whatever sparked the pain. In other words, you procrastinate.

One of the best ways to overcome procrastination is the Pomodoro technique, a diabolically clever approach developed by Francesco Cirillo. In this technique, you:

1. Turn off all distractions (no little ringie dingies on your phone or computer).
2. Set a timer for 25 minutes.
3. Focus intently for those 25 minutes.
4. Reward yourself for at least five minutes when you're done (music, talking with a friend, getting coffee).

For me, it was a pity that I didn't know about the Pomodoro technique when I was retraining my brain. Those little breaks help the brain consolidate material, which would have allowed me to build my understanding while minimizing frustration. Speaking of frustration…

Realize it's perfectly normal to *not* understand something on your first try. People often don't realize that the brain has two quite different ways of experiencing the world. The first one occurs while focusing on a topic, which activates what psychologists call *task-positive networks*. The second way occurs when we're *not* focusing on anything in particular. Not focusing involves the default mode network and other resting state networks — a *diffuse* rather than *focused* mode of thinking.

Most people need to go back and forth between focused and diffuse modes in order to learn a topic. When you can't solve something in the first focus, you're not stupid — you just need to allow time to toggle to the diffuse mode. This second mode gives your brain a chance to consolidate and consider the material from a different perspective.

The fact that no one had ever mentioned this simple idea, that "it's *normal* not to understand," was a big reason for my early failure at math. I genuinely thought any new concept I faced in math should just "click" instantly. Since it didn't, I chalked it up to my having no talent for math and quit trying. Only years later, after the army helped me form a motivational mental contrast, did I persist long enough at individual problems to discover that I could indeed learn math and science.

Build a collection of neural "chunks." When we're learning something new that doesn't come naturally to us, we often skim instead of internalizing. Yet no one would ever sing a song just one time and think it had been truly mastered. Developing expertise involves easy recall of and automaticity with key concepts and tools.

Try this exercise if you're working in something related to math or science (you can easily devise a similar exercise in other subjects). See if you can solve a key problem entirely on paper, without looking at the solution. If you can't, try it again. And again the next day, and over the next few days. Each day of focused learning, followed by an evening's sleep, strengthens your new neural patterns, which are "chunks" of learning. Soon you'll find yourself able to mentally "sing" the problem — when you just look at it, you'll find that the solution steps will flow quickly through your mind. You've "chunked" the material. But no fair fooling yourself! You need to actively work the problem on paper a number of times before you start mentally solving it.

The bigger your collection of neural chunks related to solving different problems, the greater your expertise. And the greater your expertise, the more you will like what you're learning. Daily practice and development of neural chunks are important in learning not only math and science but also virtually anything: language, sports, music, dance — even driving a car.

By using these four steps, you can develop a passion for a subject you don't like. Some subjects take longer than others to truly master, but the approach outlined here can help get you past the worst obstacles. Like me, you'll be surprised at what you can find yourself learning to love.

Barbara Oakley, PhD, is the Ramón y Cajal Distinguished Scholar of Global Digital Learning at McMaster University and the author of the new book Mindshift: Break Through Obstacles to Learning and Discover Your Hidden Potential (Tarcher-Perigee, 2017).