

Most of my students describe my classes as “*challenging but rewarding*.” It’s honest feedback. I don’t strive to make my classes difficult but I do aim to construct *authentic learning* experiences, where students tackle real-world problems, and through *self-directed learning* build a conceptual framework that they can easily access and apply to solve future problems [5, 6]. Real-world problems are often nuanced and complex, hence the challenge. Learning requires effort and engagement, hence the hard work. And the ability to apply what we learn to solve a variety of problems that we encounter beyond the classroom is ultimately rewarding. I see my role as a teacher as one who facilitates student learning by providing the necessary instructional scaffolds, illuminating for each individual student their learning pathway and their current knowledge gaps, and keeping them sufficiently motivated to achieve ambitious goals. I will illustrate my teaching methods through examples drawn from the three courses that I developed and taught at NYUAD over the past ten years: a required computer science course - Operating Systems (OS), an elective computer science course - Database Systems (DB) and a core class - Data. As a researcher by training, I approach teaching with an experimental mindset, trying out different lecturing techniques, books, case studies, etc, and exploring different ways of addressing externalities such as the shift to remote learning during COVID, the impact of AI, and the constantly evolving field of computer science. Thus, while no two iterations of a course are exactly the same, and some experiments failed, I will focus in this brief statement on teaching principles that consistently led to good learning outcomes and have become characteristic of my courses.

The Living Book & Other Scaffolds. The sudden shift to remote learning at COVID’s onset amplified two issues students were already grappling with. These issues, (i) an abundance of disconnected sources such as live lectures, textbook chapters, online videos, demos and papers to draw information from, and (ii) a multitude of technological interfaces and gates to weed through to get to relevant information (e.g. NYU Classes or Brightspace for syllabi, lecture slides etc.; online or print textbooks; email for class announcements; the syllabus to determine weekly course readings; student-only WhatsApp or Facebook groups to simply know what’s up!), created a distracted learning experience. While the presence of a wide variety of good learning content and the increase of ed-tech support tools are welcome hallmarks of today’s learning environment, the absence of a structure that holds them together, compounded by zoom-fatigue and a reduction in one-on-one interactions isolated students and exacerbated feelings of being lost that were detrimental to learning. I set out to bring structure into the learning experience of my database systems course during COVID through a

living book. This became the student's central point of information: an immediately accessible online course book with a short, straightforward URL, that expanded weekly with a new chapter (See Figure 1). Each chapter motivated and outlined the main learning objectives and concepts of the week. Integrated into its text were mini-lecture videos (I tried to keep those short to meet our shrinking attention spans), illustrative and interactive visualizations that students can explore to better understand an algorithm or a data structure, and references to additional readings, external lectures, etc. for those who want to explore more. Each weekly unit was concluded by a low-stakes assessment whose purpose was to allow students to determine if they truly grasped the material. They could submit responses as often as they wished within the one week period. These assessments guided rather than evaluated learning: if students couldn't answer a question, they revised the material, searched for answers or reached out to me or their peers for help. I used technology intentionally throughout. For example, Gradescope helped me create clear rubrics for the weekly assessments and helped me scale my ability to provide substantive feedback and determine common student misunderstandings. In class, I addressed these gaps through *patch-the-gap* sessions, which students came to value not only for how they clarified the material but also for normalizing making mistakes as part of the learning process. The liveness of the book gave me the flexibility to respond to student needs by introducing content in response to their interests and to engage with real world events: e.g. introducing a contemporary news article or reading of relevance to what we are currently studying. The impact that the structure brought about by the living book on student learning, as evidenced by student evaluations and personal notes, was so powerful that I continue to use it now and not only for DB but for all my courses.

Instructional scaffolding is about finding where students struggle with learning and providing necessary support structures. The living book is a scaffold to help bring structure into a messy learning experience. Organizing the entirety of a subject into three or four meta-ideas around which we build detail, i.e. constructing a conceptual framework, is another scaffold that I employ to tackle content complexity. In my OS course, I address student inexperience with larger code bases and low-level systems programming by introducing recitations to tackle these issues. Across my courses, I address students struggling with the comprehension and analysis of research papers through a dedicated coaching session on how to read papers. I break down my reading process by answering a series of questions: "*How did I identify the paper's main design argument? What did I do when I could not understand a specific section? Which references in the paper did I choose to read? ...*" From paper reading, I move onto paper critiquing where I engage students in the process as well. I ask them to evaluate the work in light of the world we live in now and the assumptions the authors make. "*Did*

their design deliver on the goals they set out to achieve? Are the evaluations comprehensive and convincing? ...”

Dedicating a class to paper reading and critiquing has improved the quality of critiques and in-class debates. Realizing the impact of active, question-driven reading on comprehension and analysis, I advised a capstone student to create an AI tool that generates questions to encourage active reading [1]. My goal is to make this tool freely available.

Inclusive & Individual Learning Pathways. Every student should feel capable of learning and succeeding in their chosen major. Members of underrepresented groups in computer science can feel inadequate, intimidated or alienated. Even after completing the first two years of the major, these feelings persist and may be compounded by a lack of exposure to different academic or technical experiences. I strive to make my classes inclusive of all students and accessible to them regardless of their background or their circumstances. On the first day of class, I make it clear to all students that they will all have to engage in class discussions. To ensure equal participation, I set a cap on the number of consecutive responses by a student. I also never embarrass a student. If a student goes on a tangent or provides an off-the-mark solution, I immediately point out the positive in the response, often by rephrasing it, and then articulating any flaws through a ‘yes, but’ feedback form that allows the students to reformulate another response. I try to answer all questions without hindering class progress. As students engage in positive class discussions, they become more confident in their problem-solving abilities. I also ensure that no student should fall off a metaphorical assessment cliff. My courses do not disproportionately weigh one form of assessment, such as a midterm exam, over another, such as a problem set. This ensures that a bad day or week does not negatively impact the student’s overall performance in the course. Exams are typically open-book and assess understanding and problem-solving rather than rote memory. Moreover, I have a flexible deadline policy: students have 100 hours of lateness forgiveness that they can use throughout the course for any problem set or project submission deadline. Finally, I hold students accountable for their own learning journeys. Clearly articulated learning objectives and goals allow students to determine where they are and what is missing from their understanding of a concept or their ability to apply it to solve problems. I often flip-the-script on students during office hours: instead of answering questions, I would ask them questions that help them pinpoint any source of confusion they have, and devise a plan to resolve the confusion. As a founder and the past director of the Unix-lab, a peer tutoring and learning center for CS students, I have trained many tutors on how to effectively guide their peers to find their own solutions, fostering a student culture of self-driven learning.

Motivation. The first two weeks of my OS office hours are usually spent convincing students that they can build a kernel despite their self-doubt. A combination of confidence-boosting — “*What makes you think you are not prepared? ... Here is what you do know, this is how the course will help you learn the rest*” — and demonstrating value — “*the skills and concepts you learn in the process of building a kernel will help you professionally in the following ways ...*” — works well. I also assign a fun reading, *The Night Watch* [4], to help students recognize that the intensive systems programming involved in courses like OS and DB will bear fruit in their careers as systems builders, designers or researchers. In Data, I ask students to begin thinking of a burning question that drives them to use data to answer on day one of class: we work together throughout the course to propose, implement and present a data project that tackles this question. As students realize that the scale and nature of the course assignments mimic projects that they might encounter or have encountered in life, they not only become more engaged, they ask more questions and spend more time finding answers. More importantly, they self-reflect and take control of their own learning by defining and working towards achieving learning goals. Connecting the classroom to real-life is not limited to the course projects. I ground in-class lectures and discussions with practical, real-life case studies and examples. In Data, for example, we study data and algorithm bias through the case study of biased recidivism scoring systems in criminal sentencing [3].

I cannot stress enough the importance of motivation. My greatest challenge as an educator is witnessing rising levels of student disengagement [2]. From the impact of world events, to the addictive pull of social media and amusement at our fingertips, we can no longer assume that simple carrot-and-stick approaches (good lectures and fear of bad grades) are enough to motivate our students to learn. Students are increasingly asking “*why should I learn this?*” and they deserve a compelling response. I believe education should enlighten a person: I motivate my students by showing them how even the most technical of concepts can inform how they live their lives. For example, when learning about OS multitasking, we take a moment to discuss the impact of multitasking and context-switching (say from email, or tik-tok to studying) on our cognitive performance, or when learning database system failure recovery protocols, we discuss how we can recover from our own mistakes. When I receive an email from a student years after they graduated where they say I applied this concept in a recent professional or personal problem and it worked, that is when I know that I succeeded in facilitating their learning and growth.

The Database Systems Course Book

Search

Table of contents

Welcome to Database Systems Overview

Part I Models & Languages

1 The Relational Model

2 Relational Algebra

3 Database Design

4 Normalization

Part II Query Execution & Optimization

5 The Database File Abstraction

6 Database Indexes: B+ Trees

7 Buffer Pool Management

8 Sorting & Hashing

9 Query Processing & Joins

10 Query Optimization

Part III Transactions

11 Concurrency Control

12 Recovery

Appendix

A SQL Deep Dive





B Benchmarking

C Group Work

D Labs & Problem Sets

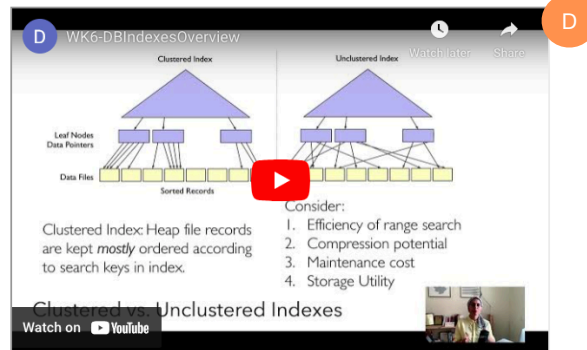
6 Database Indexes: B+ Trees

Week 6: March 6 - March 12, 2022

-  COWS Chapter 10 Tree-Structured Indexing
-  COWS Sections 20.2-20.4 (Fig 20.2 is quite cool) How to select indexes?
-  Tree Indexing on Gradescope.
-  Tree Indexes

6.1 What is a database index?

In this module, we will look into index access methods and how indexes enable efficient table access by attribute search keys. We first discuss database indexes and their types broadly.



Clustered Index: Heap file records are kept mostly ordered according to search keys in index.

Unclustered Index

Consider:

1. Efficiency of range search
2. Compression potential
3. Maintenance cost
4. Storage Utility

Clustered vs. Unclustered Indexes

Watch on YouTube

6.3 How a B+ tree works?

6.3.2 Insertions & Deletions

To understand how insertions work, Miro Mannino and I created this custom visualization for you. You can see all the intermediate steps for each insertion. You can select the fanout, the number of keys to insert and whether you wish values to be inserted in sorted order or in random order.

of keys to insert: fanout: sort keys before insertion?

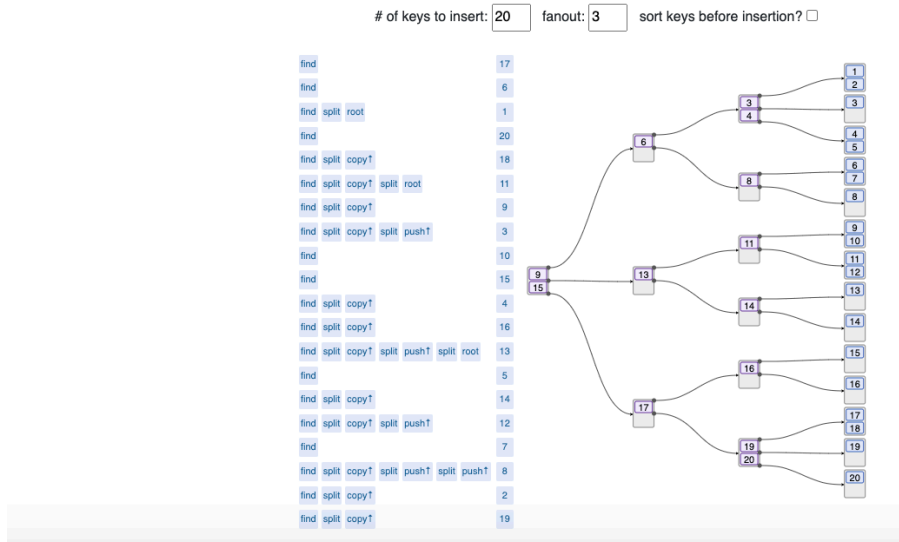


Figure 1: The Living Book. **A:** A new module or chapter is introduced every week. **B:** References to only the relevant external material are clearly linked. **C:** A low-stakes assessment guides active learning and allows students to determine their grasp of the material. **D:** Embedded in the text are mini video-lectures or other multi-media content. **E:** Interactive visualizations allow students to understand complex algorithms or data structures through active exploration.

References

- [1] Liam Richards Maldonado, Azza Abouzied, and Nancy W. Gleason. “ReaderQuizzer: Augmenting Research Papers with Just-In-Time Learning Questions to Facilitate Deeper Understanding”. In: *Companion Publication of the 2023 Conference on Computer Supported Cooperative Work and Social Computing*. CSCW '23 Companion. Minneapolis, MN, USA: Association for Computing Machinery, 2023, pp. 391–394. URL: <https://doi.org/10.1145/3584931.3607494>.
- [2] Steven Mintz. “Opinion: An Epidemic of Student Disengagement”. In: *Inside Higher Education* (Apr. 2022). URL: <https://www.insidehighered.com/blogs/higher-ed-gamma/epidemic-student-disengagement>.
- [3] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. *Machine Bias: Investigating the algorithms that control our lives*. May 2016. URL: <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>.
- [4] James Mickens. “The Night Watch”. In: *login: logout - the USENIX Magazine* (2013), pp. 5–8.
- [5] Audrey C Rule. “The Components of Authentic Learning”. In: *Journal of Authentic Learning* 3.1 (2006), pp. 1–10.
- [6] M Suzanne Donovan, John D Bransford, and James W Pellegrino. *How People Learn: Bridging Research and Practice*. National Academies Press, 1999.