­­­­­­­Databases Intro – covers course intro and chap 1 of book

Standard stuff (see slide)

Pass out syllabus.

* webpage: will have day-to-day schedule, homework, project information
  + use this whenever you need to know what’s happening or miss a day
* Textbook: show it to class. Get it somewhere. I will be assigning problems and readings out of it. Keep up with the reading. Used is fine. Available online [pirated] if you look.
* Pre-req: CS241. If you do not have this, come talk to me.
* Coursework.
* Be prepared to bring laptops every so often.
* Talk about group project (see slide)
* INTRODUCTIONS
* Jump into why this class is awesome.

Why study databases? [ use slide! ]

* Academic
  + DBs involve many aspects of computer science.
  + Fertile area of research.
  + Four Turing awards in awarded to people who did fundamental research on databases. [given by the ACM every year, Nobel prize of CS]
* Programmatic
  + Numerous applications out there use or should use database systems to organize information.
  + Pretty much any kind of application which is designed around CRUD uses a database.
* Business
  + Every company needs databases, even if they’re not software companies.
  + Rhodes College has an entire division of IT, called data services, that just handles the college-level databases that store lots of information about the college, its students, and faculty.
    - Whenever you use bannerweb, you are interfacing with a database behind the scenes.
  + Lots of money to be made, and lots of jobs to be had!
* Student
  + Need this class to graduate!
  + Professor Kirlin is awesome.
  + Want a job and hope someone will hire me.
    - Five major “traditional” database companies are Oracle, IBM (DB2), Microsoft (SQL Server), SAP, Teradata
    - Online cloud databases: Amazon, Google (Google Storage), MongoDB
    - Other companies: pretty much everyone needs databases!
* **What will you learn? (use slide)**
  + Three main focuses of this course.
  + [see 3 on slide]
* **What is the goal of a database?**
  + use slide
* **DBMS features**
  + Support massive amounts of data.
    - The World Data Centre for Climate:
      * 220 Terabytes
      * used for climate simulation
    - National Energy Research Scientific computing Center:  based in Oakland, ca
      * 2.8 petabytes
      * owned and operated by Lawrence Berkeley national laboratory and the Dept of Energy.
      * Atomic energy research, high energy physics experiments, simulations of the early universe.
    - AT&T, Google, Sprint, LexusNexus, Youtube, amazon, CIA,
  + Persistent storage. 🡪 different than storing data in regular programs, where everything disappears when the program ends.
  + Efficient and convenient access.
  + **Secure**: Certainly includes preventing hackers from the outside from getting in, but also includes features for controlling who is allowed to do what to the database. Certain people will have certain privileges for adding, changing, or deleting information.
  + **Concurrent**: Let multiple people access the database at one time.
  + **Atomic**: allows users to group multiple database operations together and ensure that either everything in the group happens, or none of it happens (prevents leaving the database in an inconsistent state).
    - Example: Bank database needs to move money from one account to another.
* Build a better workday
  + What are some queries?
  + What is my gpa?
  + What is my class schedule?
  + What classes are offered at a given time?
  + Use manila folders
    - Advantages – cheap, easy to use (could be secure)
    - Disadvantages – not electronic, not easy to access, hard to query
  + Text files and python
    - Advantages – electronic
    - Disadvantages – hard to program in every possible question we’d want to ask.
  + Text files and python++ CSV
    - Advantages – standardized data format – easy to read into code, lots of libraries out there to read data
    - What is a disadvantage? – still no concurrency. not easy to access by multiple people. no security? What if deleted?
  + So this is looking pretty nice. What happens with more lines in the CSV file?
    - Now data is duplicated. Now we have redundancy. (e.g., what if we need to change Hermione’s R-number?)
    - We also have things linked together than don’t need to be. What if a new student comes but isn’t registered for classes yet? Where do we put them in the file? We can’t, because there has to be a class+grade.
  + Now we split up the data into multiple files. Our first file can just have people and R-numbers.

2.1 Data Models

* What is a data model? It is a pattern or notation for describing data.
  + A description of how to conceptually structure the data, what operations are possible on the data, and any constraints on the data.
  + Analogous to creating a class in Java.
  + Structure – how we view the data abstractly. What is this for class? [it is the fields of a class or struct, or what data structure you use.] In DB,
  + Operations – what is possible to do with the data? Class=methods, functions
  + Constraints – how can we control what data is legal and what is not? Class=data types, more sophisticated for DB
* Most important data model for DBs is the RELATIONAL model.
  + Structure: relation OR table
  + Operations: relational algebra OR Sql
    - Select certain rows or certain columns that have or do not have certain properties.
    - This limitation is actually a good thing. By using a language like relational algebra or SQL, this allows the DB to optimize our queries to make them as efficient as possible.
    - For instance, when you want something in a database to come out sorted, you literally add the phrase ORDER BY into your query. This allows the DB to pick them best way to do the sort. On the other hand, in python or C++, if you want to sort something, in general, you have to pick the algorithm, and no compiler is smart enough to replace your bubble sort with a quick sort.
  + Constraints: can enforce restrictions like grade must be in the set ABCDF
* Show harry potter table.
  + Structure is tabular, you can think of this as an array or list of structs. Each row represents the values of one entry in the array of structs. Behind the scenes, this probably isn’t an array or list, but some more sophisticated data structure. (Btree)
* Semistructured model is an alternative – usually resembles graphs or trees. = data that Is still “structured” but not in the relational format.
  + Structure is XML or JSON.
  + Operations usually involve following paths in the hierarchy (XQuery)
  + Constraints: document type definitions (DTDs)
  + Other data models – object databases, or object-relational
    - Graph databases
    - NoSQL, NewSQL
  + Object-relational – similar to relational, but values in a table can have their own structure, rather than being simple strings or ints or floats (they can be objects)
    - Relations can have methods associated with them
* Relational model is most common
  + Simple: built around a single concept for modeling data: the relation or table.
    - A relational database is a collection of relations.
    - Each relation is a table with rows and columns.
    - An RDBMS can manage many databases at once.
  + Supports high-level programming language (SQL)
    - Limited but useful set of operations.
  + Has elegant mathematical theory behind it.
* Terminology [start 2.2]
  + **Relation** == 2D table
    - **Attribute** == column name
    - **Tuple** == row (not the header row)
  + **Database** == collection of relations
* Relation terminology
  + A relation includes two parts:
  + The **relation** **schema** defines the column headings of the table (attributes/fields)
  + The **relation** **instance** defines the data rows (tuples, rows, or records) of the table.
  + Schema is kind of like defining a struct in C++. You define what members the class has and what their data types are.
  + Each row in a DB is an instance of the class, so an instance of the entire relation is a collection of these rows.
* Schema
  + A schema is written by the name of the relation followed by a parenthesized list of attributes.
  + Grades(First, Last, Course, Grade)
  + A **relational database schema** is the set of schemas for all the relations in a DB.
* Domains
  + A relational DB requires that every component of a row (tuple) have a specific elementary data type, or **domain**.
  + string, int, float, date, time (no complicated objects!)
  + Grades(First:string, Last:string, Course:string, Grade:char)
* Relation equivalence
  + Relation is a ***set*** of tuples, not a list. [that is, there is no order to the rows, though in the real world, there often is]
  + Attributes in a schema are a ***set*** as well. [ no order to cols either]
  + However, the schema specifies a "standard" order for the attributes.
  + How many equivalent representations are there for a relation with *m* attributes and *n* tuples?
    - How many different ways could I draw this table and still technically have the same relation?
    - Answer = m! \* n!
* Degree/cardinality
  + **Degree/arity** of a relation is the number of attributes in a relation.
  + **Cardinality** is the number of tuples in a relation.
* Keys
  + Keys are a kind of **integrity constraint**.
  + A key is a set of attributes with a specific property.
  + A set of attributes K forms a key for a relation R if:
    - we forbid two tuples in an instance of R from having the same values for all attributes of K.
  + What are keys of Grades?
* Artificial keys [slide]
  + Often, many real world databases create artificial keys, because often we want to be 100% sure that we will never have any duplicates in combinations of attributes.
  + What are the keys in these relations?
* Let's expand these relations to handle the kinds of things you'd like to see in BannerWeb.
* Keep track of students, professors, courses, who teaches what, enrollments, pre-requisites, grades, departments & their chairs.
  + Only one chair per department.
  + Student cannot enroll in multiple copies of the same course in one semester.
  + Other constraints that are logical.