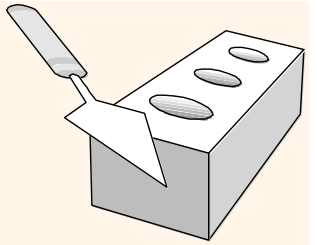
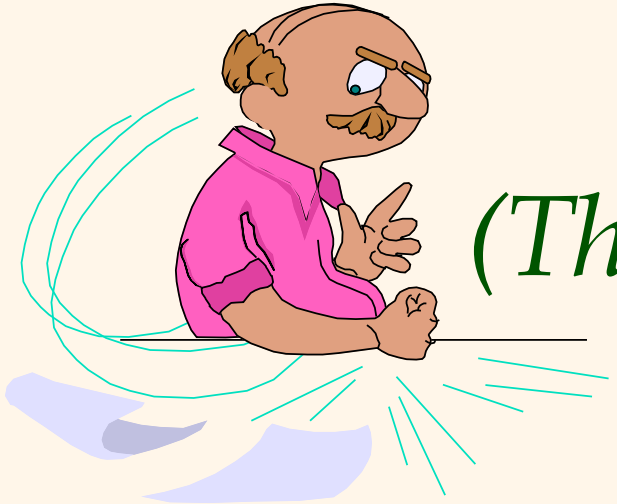
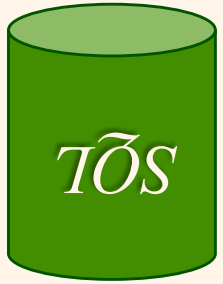


Introduction to Data Management

**** The "Flipped" Edition ****





Lecture #1

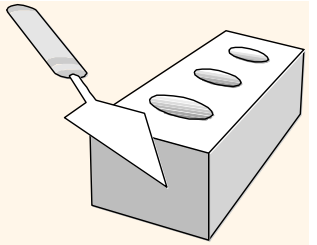
(The Course "Trailer")

Instructor: Mike Carey
mjcarey@ics.uci.edu

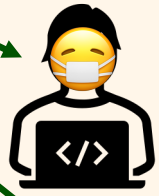
Today's Notices

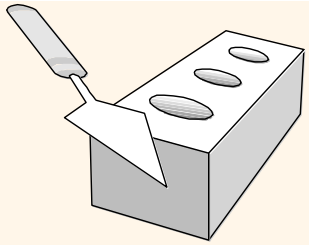
- Again: Welcome to my flipped CS122A!
- Read (and live by!) the course wiki page:
  - http://www.ics.uci.edu/~cs122a/
- Also follow (and live by) the Piazza page:
  - http://piazza.com/uci/fall2020/cs122aeecs116/home
- ***Note*:** There *will* be a quiz in this coming week's flipped lectures and discussion sessions...!
  - *Again*: **Please** attend/watch the sessions you're signed up for!

Reminders



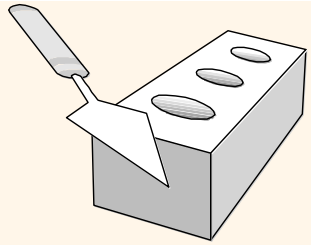
- We've all been ***Zoom'***ing for a long time - so don't forget, for the live sessions, to bring your...
  - Masks
  - Pants
  - Shoes





What is a Database System?

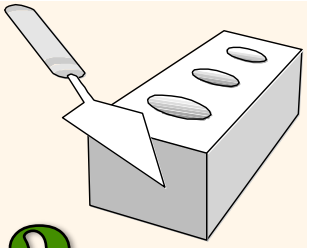
- What's a *database*?
  - A very large, integrated collection of data
- Usually a model of a *real-world enterprise*
  - ***Entities*** (*e.g.*, students, courses, Facebook users, …) with attributes (*e.g.*, name, birthdate, GPA, …)
  - ***Relationships*** (*e.g.*, Susan is *taking* CS 234, Susan is a *friend of* Lynn, …)
- What's a *database management system* (DBMS)?
  - A software system designed to store, manage, and provide access to one or more databases



File Systems vs. DBMS

- Application programs must sometimes *stage large datasets* between main memory and secondary storage (for buffering huge data sets, getting page-oriented access, etc.)
- *Special code needed* for different queries, and that code must be (stay) correct and efficient
- Must *protect data from inconsistency* due to multiple concurrent users
- *Crash recovery* is important since data is now the currency of the day (corporate jewels)
- *Security and access control* are also important(!)

Evolution of DBMS



Files

Manual Coding
• Byte streams
• Majority of application development effort goes towards building and then maintaining data access logic

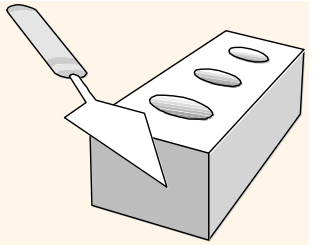
CODASYL/IMS

Early DBMS Technologies
• Records and pointers
• Large, carefully tuned data access programs that have dependencies on physical access paths, indexes, etc.

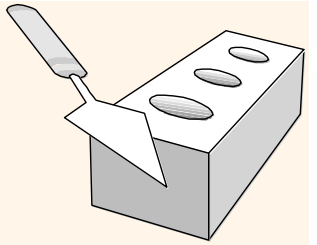
Relational

Relational DB Systems
• **Declarative approach** • Tables and views bring "data independence" • Details left to system
• Designed to simplify data-centric application development

Why Use a DBMS?



- Data independence.
- Efficient data access.
- Reduced application development time.
- Data integrity and security.
- Uniform data administration.
- Concurrent access, recovery from crashes.

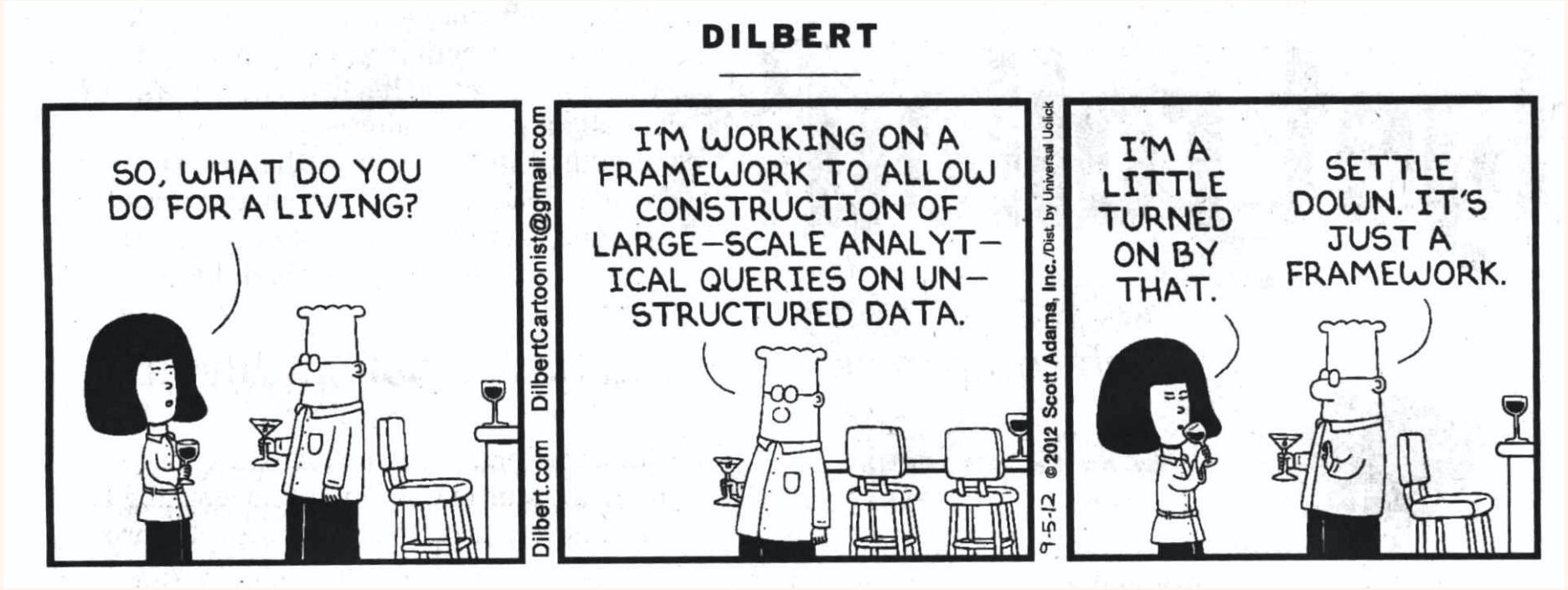


Why Study Databases?

- Shift from *computation* to *information*
  - At the "low end": explosion of the web (a mess!)
  - At the "high end": scientific applications, social data analytics, …
- Datasets increasing in diversity and volume
  - Digital libraries, interactive video, Human Genome project, EOS project , the Web itself, …
  - Mobile devices, Internet of Things, …
  - ***… need for DBMS exploding!***

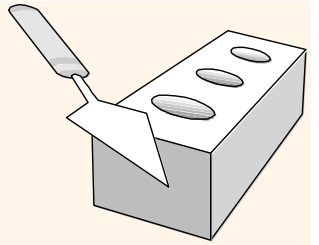
- DBMS field encompasses most of CS!!
  - OS, languages, theory, AI, multimedia, logic, …

Why Study Databases (Really)?



→ ***Big Data!*** ☺

https://www.cio.com/article/3292983/what-is-a-data-engineer.html



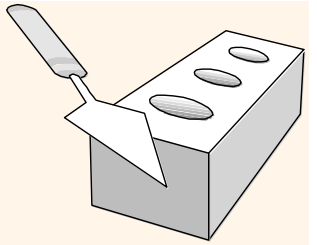
Data Models

- A *data model* is a collection of concepts for describing data
- A *schema* is a description of a particular collection of data, using a given data model
- The *relational model* is (still) the most widely used data model today

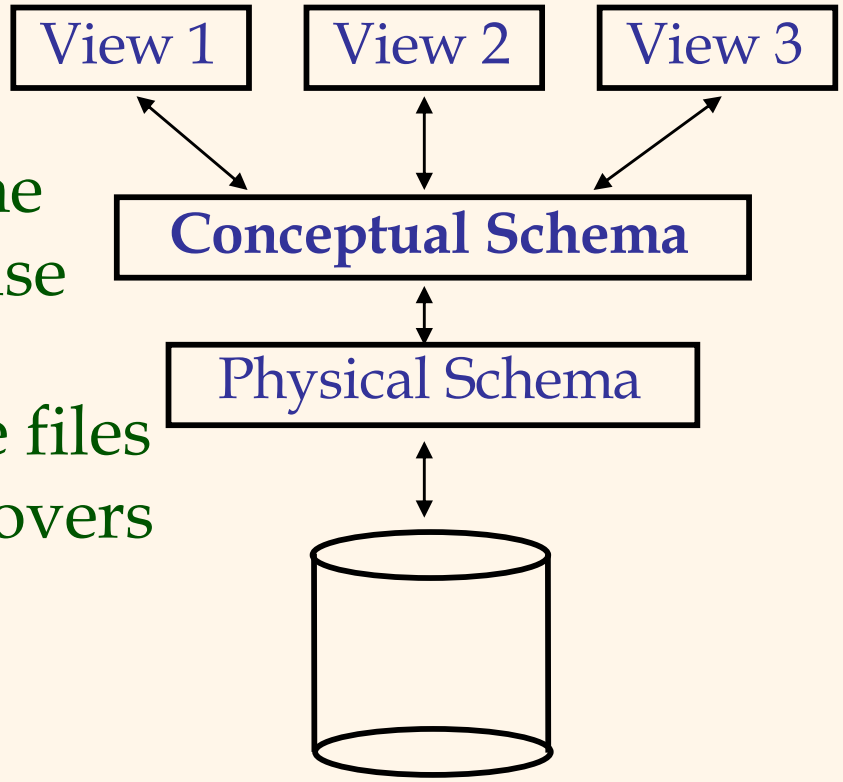
Ex: Gradescope uses PostgreSQL under the hood!

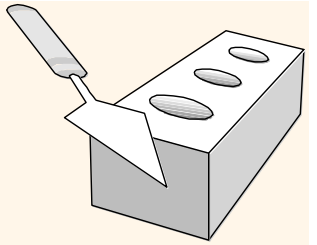
  - *Relation* – basically a table with rows and (named) columns
  - *Schema* – describes the tables and their columns

Levels of Abstraction



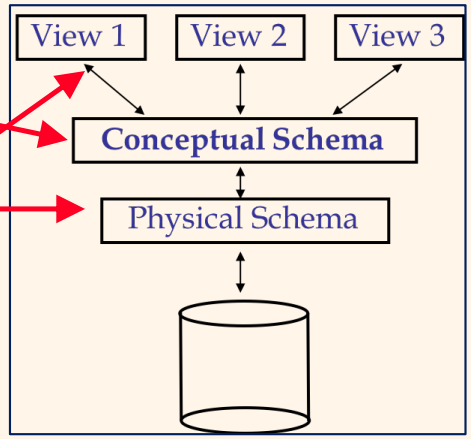
- Many *views* of one *conceptual (logical) schema* and an underlying *physical schema*
  - Views describe how different users see the data.
  - Conceptual schema defines the logical structure of the database
  - Physical schema describes the files and indexes used under the covers

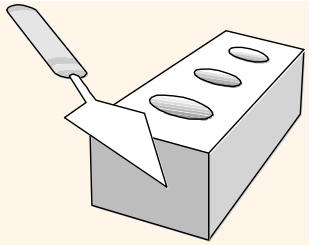




Example: University DB

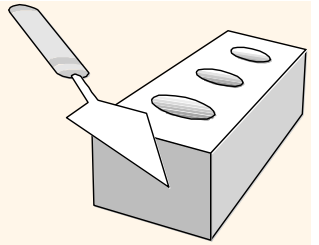
- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa: real)*
  - *Courses(cid: string, cname: string, credits: integer)*
  - *Enrolled(sid: string, cid: string, grade: string)*
- Physical schema:
  - Relations stored as unordered files
  - Indexes on first and third columns of *Students*
- External schema (*a.k.a.* view):
  - *CourseInfo(cid: string, cname: string, enrollment: integer)*





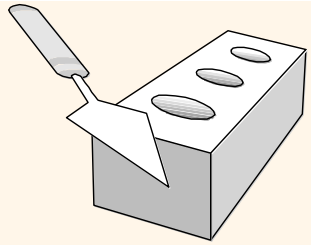
Data Independence

- Applications are *insulated* (at multiple levels) from how data is actually structured and stored, thanks to schema layering and high-level queries
  - *Logical data independence*: Protection from changes in the *logical* structure of data
  - *Physical data independence*: Protection from changes in the *physical* structure of data
- *One of the most important benefits of DBMS use!*
  - Allows *changes* to occur – ***w/o application rewrites***!



University DB Example (cont.)

- End user query (in SQL, against the external schema):
  - *SELECT c.cid, c.enrollment*
    FROM CourseInfo c
    WHERE c.cname = 'Computer Game Design'
- Equivalent query (against the conceptual schema):
  - *SELECT e.cid, count(e.*)*
    FROM Enrolled e, Courses c
    WHERE e.cid = c.cid AND c.cname = 'Computer Game Design'
    GROUP BY c.cid
- Steps under the hood (against the physical schema)
  1. Access ***Courses*** – use index on *cname* to find associated *cid*
  2. Access ***Enrolled*** – use index on *cid* to count the enrollments

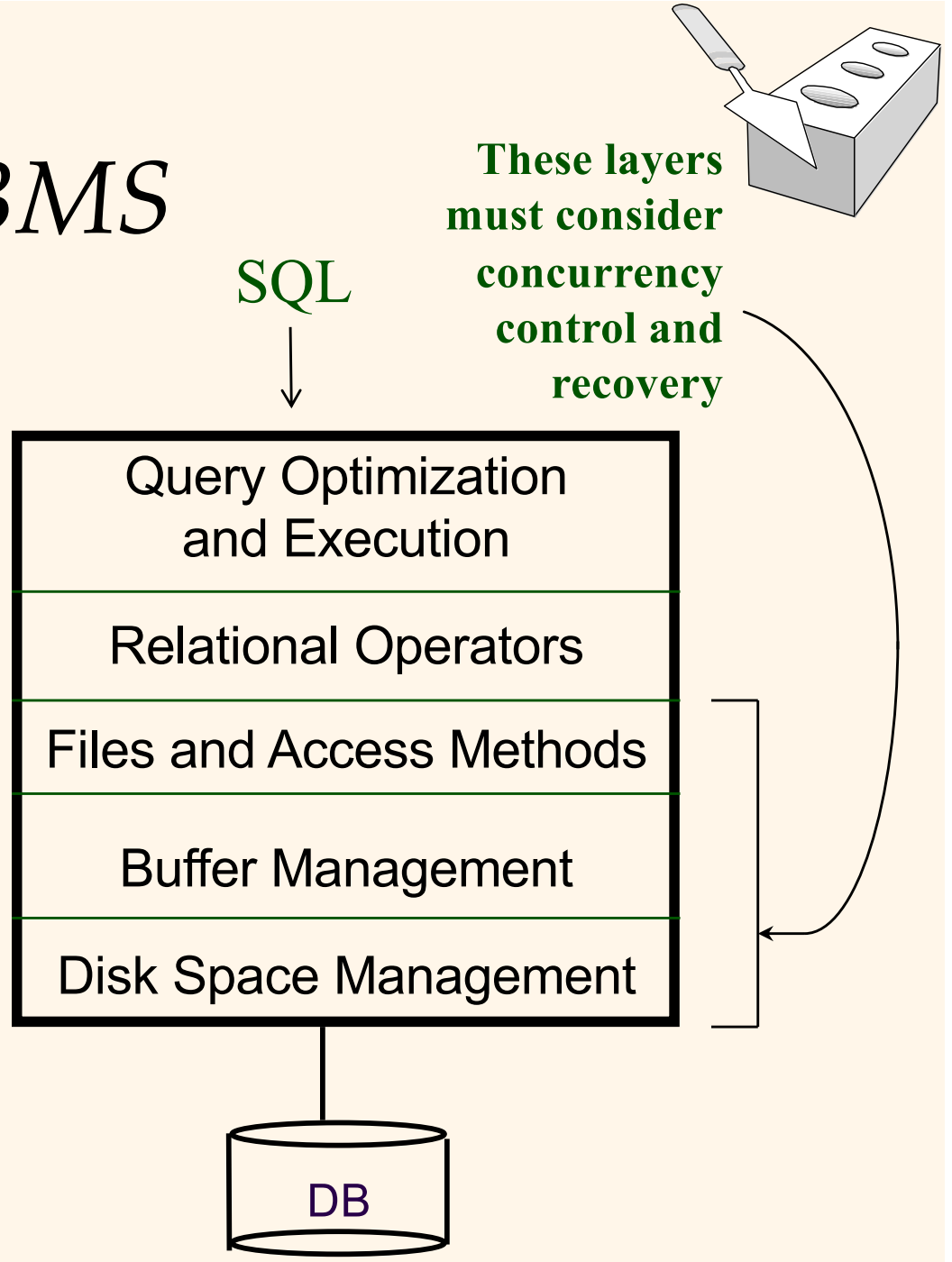


Concurrency and Recovery

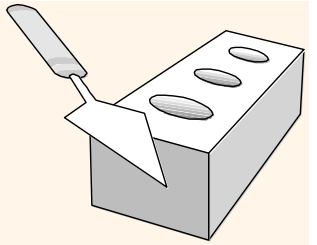
- *Concurrent execution* of user programs is essential to achieve good DBMS performance.
  - Disk accesses are frequent and slow, so it's important to keep the CPUs busy by serving multiple users' programs concurrently.
  - Interleaving multiple programs' actions can lead to inconsistency: e.g., a bank transfer while a customer's assets are being totaled.
- *Errors or crashes* may occur during, or soon after, the execution of users' programs.
  - This could lead to undesirable partial results or to lost results.
- DBMS answer: Let users/programmers pretend that they're using a reliable, single-user system!

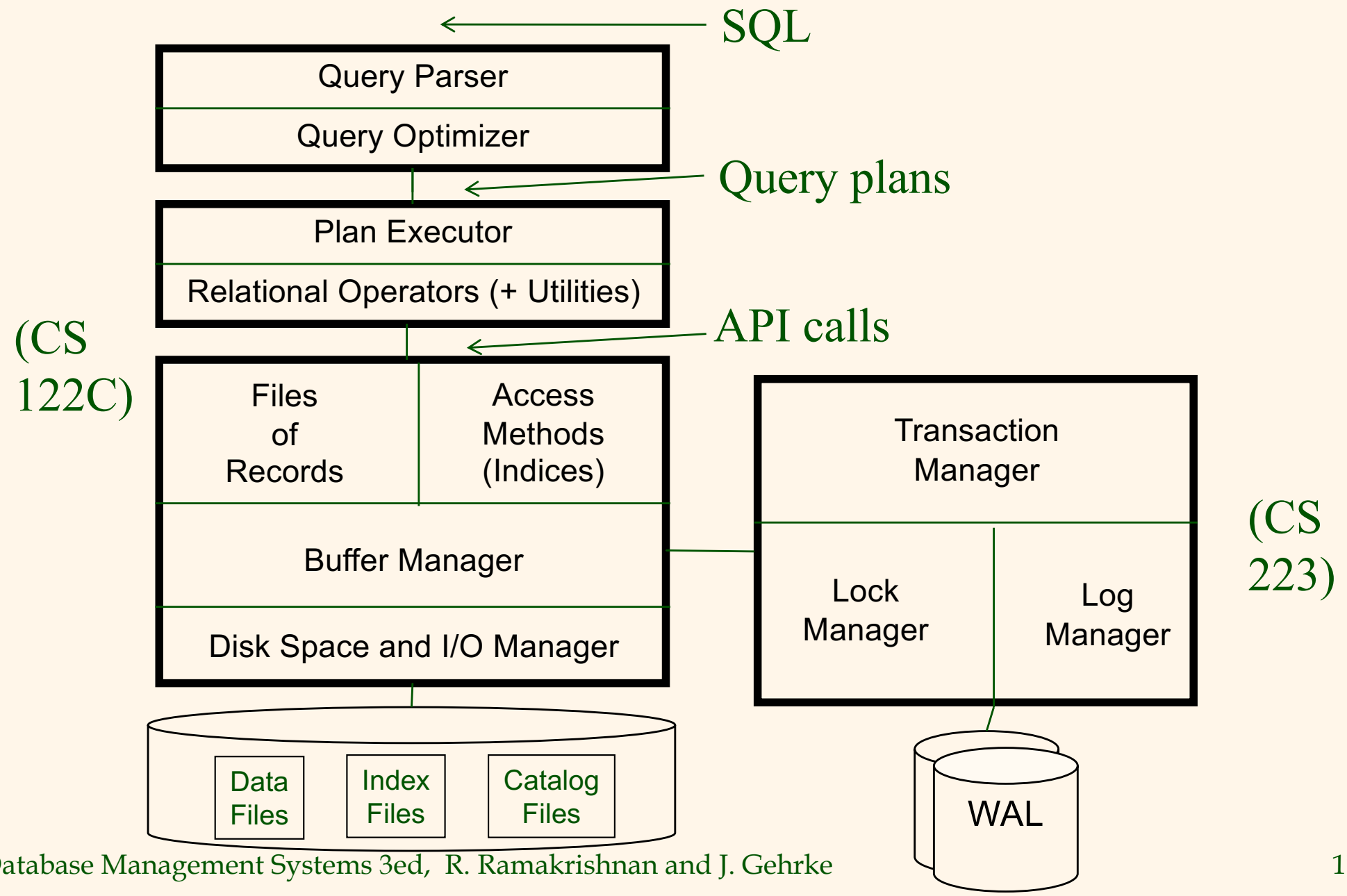
Structure of a DBMS

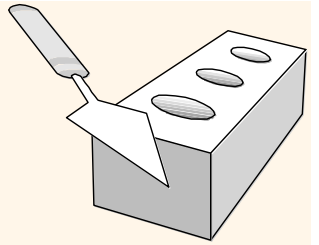
- A typical DBMS has a layered architecture.
- This figure leaves out the concurrency control and recovery components.
- This is one of several possible architectures; each RDBMS has its own variations.



DBMS Structure (More Detail)





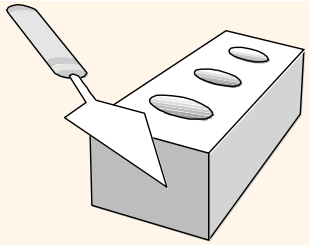


Components' Roles

```
SELECT e.title, e.lastname
FROM Employees e, Departments d
WHERE e.dept_id = d.dept_id AND
      year (e.birthday) >= 1970 AND
      d.dept_name = 'Engineering'
```

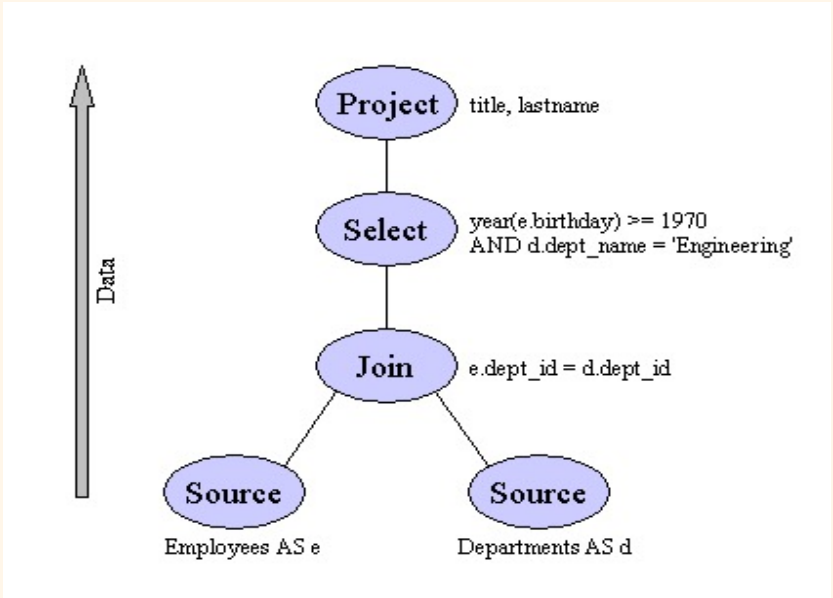
- Query Parser
  - Parse and analyze *SQL query*
  - Makes sure the query is valid and talking about tables, etc., that indeed exist
- Query optimizer (usually has 2 steps)
  - *Rewrite* the query logically
  - Perform cost-based *optimization*
  - Goal is finding a "good" query plan considering
    - Available access paths (files & indexes)
    - Data statistics (if known)
    - Costs of the various relational operations

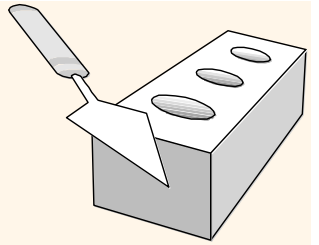
(Cost differences are often orders of magnitude!!!)



Components' Roles (continued)

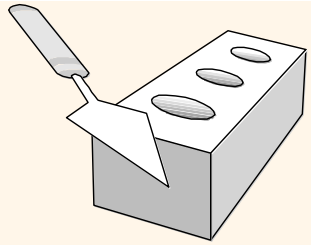
- Plan Executor + Relational Operators
  - Runtime side of query processing
  - Query plan is a tree of relational operators (drawn from the ***relational algebra,*** which you will learn all about in this class)





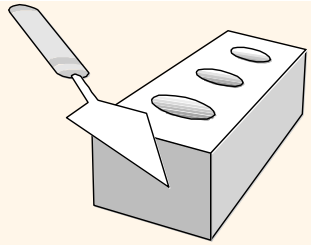
Components' Roles (continued)

- Files of Records
  - DBMSs have ***record*** based APIs under the hood
    - Record = set of fields
    - Fields are typed
    - Records reside on pages of files
- Access Methods
  - Index structures for lookups based on field values
  - We'll look in more depth at ***B+ tree*** indexes in this class (the most commonly used index type for both commercial and open source DBMSs)



Components' Roles (continued)

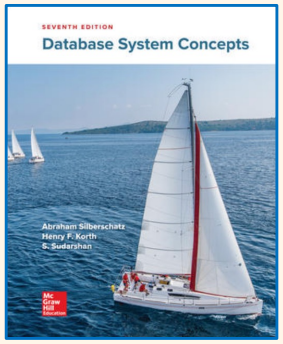
- Buffer Manager
  - The DBMS answer to *main memory* management!
  - All disk page accesses go via the buffer pool
  - Buffer manager caches pages from files and indexes
- Disk Space and I/O Managers
  - Manage space on *disk* (pages)
  - Also manage I/O (sync, async, prefetch, …)
  - Remember: database data is ***persistent*** *(!)*

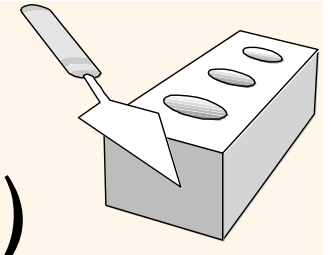


Components' Roles (continued)

- System Catalog (or "Metadata")
  - Info about tables (name, columns, column types, …);
  - Data statistics (e.g., counts, value distributions, …)
  - Info about indexes (tables, index kinds, …)
  - And so on! (Views, security, …)
- Transaction Management
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Lock Manager for Consistency + Isolation
  - Log Manager for Atomicity + Durability

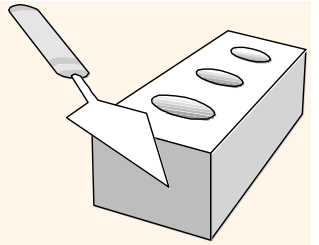
Yet Another Block Diagram





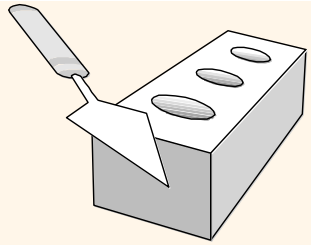
Miscellany: Terms (for parties ☺)

- Data Definition Language (DDL)
  - Used to express views + logical schemas (using a syntactic form of a a data model, e.g., relational)
- Data Manipulation Language (DML)
  - Used to access and update the data in the database (again in terms of a data model, e.g., relational)
- Query Language (QL)
  - Synonym for DML, or for its retrieval (i.e., data access or query) sublanguage



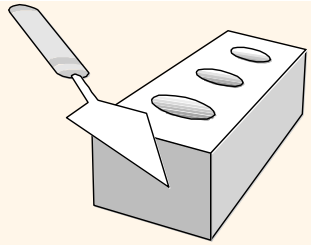
Miscellany (cont'd.): Roles

- Database Administrator (DBA)
  - The "super user" for a database or a DBMS
  - Handles physical DB design, tuning, performance monitoring, backup/restore, user/group management
- Application Developer
  - Builds data-centric applications (take CS122b!)
  - Involved with logical DB design, queries, and DB application tools (e.g., JDBC, ORM, …)
- Data Analyst or End User
  - Non-expert who uses tools to interact w/the data
- Data Engineer (*new*)
  - Develops/constructs/maintains Big Data platforms and data flows
  - Uses multiple Big Data (etc.) tools and technologies to prepare data products for consumption by Data Scientists



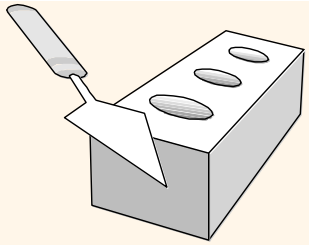
A Brief History of Databases

- Pre-relational era: 1960's, early 1970's
- **Codd's relational model paper: 1970**
- Basic RDBMS R&D: 1970-80 (System R, Ingres)
- RDBMS improvements: 1980-85
- Relational goes mainstream: 1985-90
- Distributed DBMS research: 1980-90
- Parallel DBMS research: 1985-95
- Extensible DBMS research: 1985-95
- OLAP and warehouse research: 1990-2000
- Stream DB and XML DB research: 2000-2010
- "Big Data" R&D (also including "NoSQL"): 2005-present



Introductory Recap

- DBMSs are used to maintain & query large datasets.
- Benefits include recovery from system crashes, concurrent access, quick application development, data integrity and security.
- Levels of abstraction give data independence.
- A DBMS typically has a layered architecture.
- DBAs (and Data Engineers) hold responsible jobs and they are also well-paid! (☺)
- Data-related *R&D* is one of the broadest, most exciting areas in CS.



So Now What?

- Time to dive into the first tech topic:
  - ***Logical DB design*** (ER model)
- Read the initial sections of the book
  - DB intro and ER – see the syllabus on the wiki!

Syllabus

Topic	Reading (Required!)
Databases and DB Systems	Ch. 1
Entity-Relationship (E-R) Data Model	Ch. 6.1-6.5, 6.8-6.9

- Immediate to-do's for you are:
  - Again, be sure that you're signed up on Piazza
  - And, stockpile sleep – no homework quite ***yet*** (☺)
- ***Next time: Database design…***