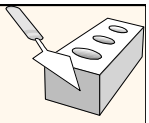
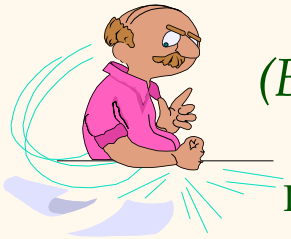
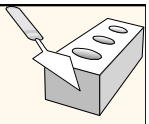


Introduction to Data Management



Lecture #2
(Big Picture, Cont.)

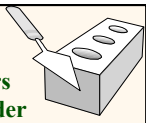
Instructor: Mike Carey
mjcarey@ics.uci.edu



Announcements

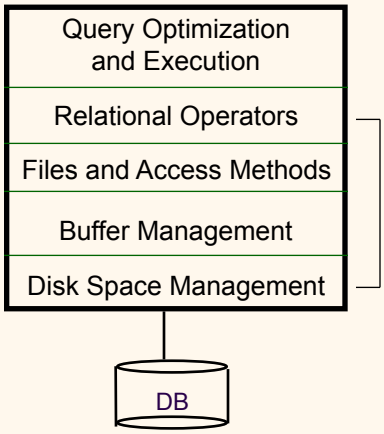
- Still hanging in there? (If you do think you may drop, please decide soon so that others who are waiting can get in – *thanks…!* ☺)
- Today's plan:
  - More detail about DBMS architectures
  - Then on to logical DB design!
- Reminder:
  - Sign up on Piazza! (Only ~half have done this.)
  - HW #1 and Project Part 1 coming next week!
- Any lingering *Q*'s from last time?

Structure of a DBMS

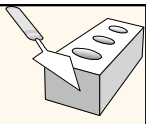


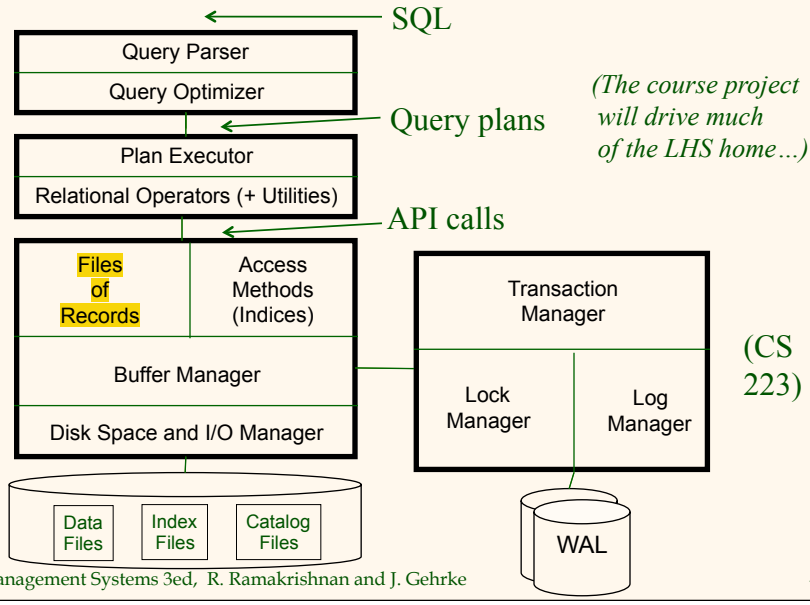
These layers must consider concurrency control and recovery

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components (CS 223).
- This is one of several possible architectures; each system has its own variations.



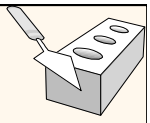
DBMS Structure In More Detail





(The course project will drive much of the LHS home…)

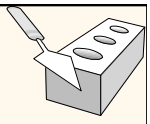
(CS 223)



Components' Roles

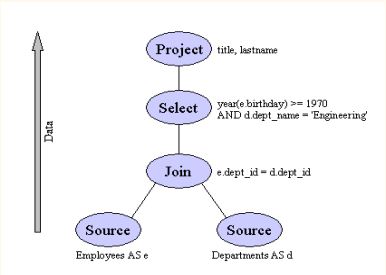
- Query Parser
  - Parse and analyze SQL query
  - Makes sure the query is valid and talking about tables, etc., that indeed exist
- Query optimizer (often w/2 steps)
  - Rewrite the query logically
  - Perform cost-based optimization
  - Goal is a "good" query plan considering
    - Physical table structures
    - Available access paths (indexes)
    - Data statistics (if known)
    - Cost model (for relational operations)

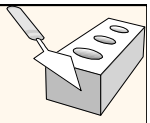
(Cost differences can be orders of magnitude!!!)



Components' Roles (continued)

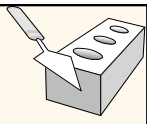
- Plan Executor + Relational Operators
  - Runtime side of query processing
  - Query plan is a tree of relational operators (drawn from the relational algebra, which you will learn about in this class)





Components' Roles (continued)

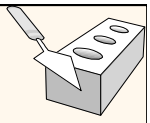
- Files of Records
  - OSs usually have byte-stream based APIs
  - DBMSs instead provide record-based APIs
    - Record = set of fields
    - Fields are typed
    - Records reside on pages of files
- Access Methods
  - Index structures for access based on field values
  - We'll look in a fair bit of depth at B+ tree indexes in this class (they are the most commonly used indexes across all commercial and open source systems)



Components' Roles (continued)

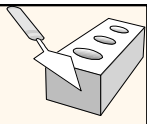
- Buffer Manager
  - The DBMS answer to main memory management!
  - All disk page accesses go through the buffer pool
  - Buffer manager caches pages from files and indices
  - "DB-oriented" page replacement scheme(s)
  - Also interacts with logging (so undo/redo possible)
- Disk Space and I/O Managers
  - Manage space on disk (pages), including extents
  - Also manage I/O (sync, async, prefetch, …)

Components' Roles (continued)

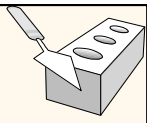


- System Catalog
  - Info about physical data (volumes, table spaces, …)
  - Info about tables (name, columns, types, …); also, info about their constraints, keys, etc.)
  - Data statistics (e.g., value distributions, counts, …)
  - Info about indexes (types, target tables, …)
  - And so on! (Views, security, …)
- Transaction Management
  - ACID (Atomicity, Consistency, Isolation, Durability)
  - Lock Manager for Consistency+Isolation
  - Log Manager for Atomicity+Durability

Miscellany: A Few Terms

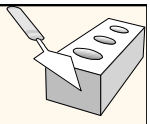


- Data Definition Language (DDL)
  - Used to express views + logical schemas (using a syntactic form of a a data model, e.g., relational)
- Data Manipulation Language (DML)
  - Used to access and update the data in the database (again in terms of a data model, e.g., relational)
- Query Language (QL)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



Miscellany (Cont'd.): Key Players

- Database Administrator (DBA)
  - The "super user" for a database or a DBMS
  - Deals with things like physical DB design, tuning, performance monitoring, backup/restore, user and group authorization management
- Application Developer
  - Builds data-centric applications (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



A Brief History of Databases

- Pre-relational era: 1960's, early 1970's
- Codd's seminal 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: 2005-present

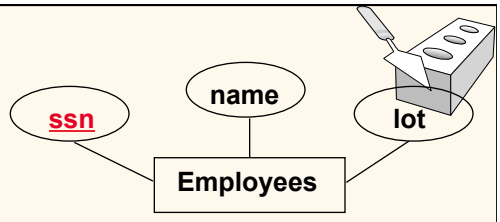
So Now What?

- Time to dive into the first real topic:
  - Logical DB design (ER model)
- Read the first two chapters of the book
  - Intro and ER – see the syllabus on the wiki
- Immediate to-do's for you are:
  - Get yourself signed up on Piazza
  - Stockpile sleep – no homework for you yet ☺
  - Start thinking about a project partner
- On to DB design…

Overview of Database Design

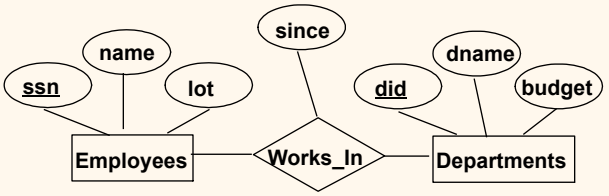
- *Conceptual design: (ER Model used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database `schema' in the ER Model can be represented pictorially (*ER diagrams*).
  - Can map an ER diagram into a relational schema (manually or using a design tool's automation).

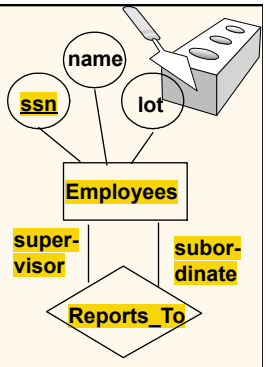
ER Model Basics



- *Entity:* Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway… ☺)
  - Each entity set has a *key* (a unique identifier).
  - Each attribute has a *domain* (similar to a data type).

ER Model Basics (Contd.)

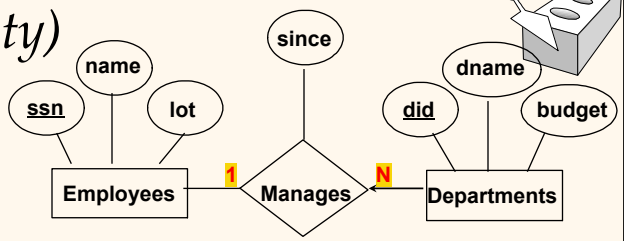


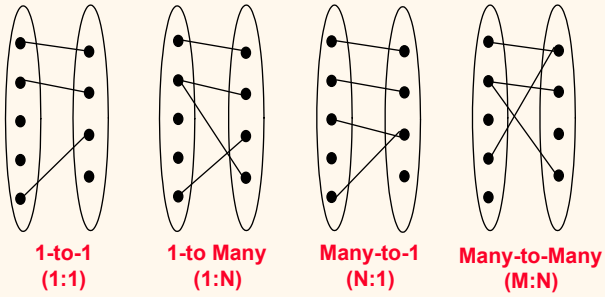


- *Relationship*: Association among two or more entities. E.g., Attishoo works in Pharmacy department.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 E1, ..., en En
    - Same entity set could participate in different relationship sets, or in different "roles" in same set.

Key (Cardinality) Constraints

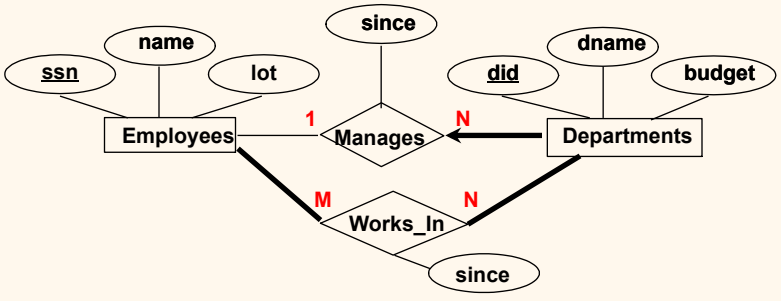
- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.



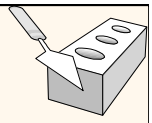


Participation Constraints

- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every Departments entity must appear in an instance of the Manages relationship
    - Ditto for both Employees and Departments for Works_In



Tune In Next Week…



- … for part two of this riveting, ER modeling cliff-hanger!