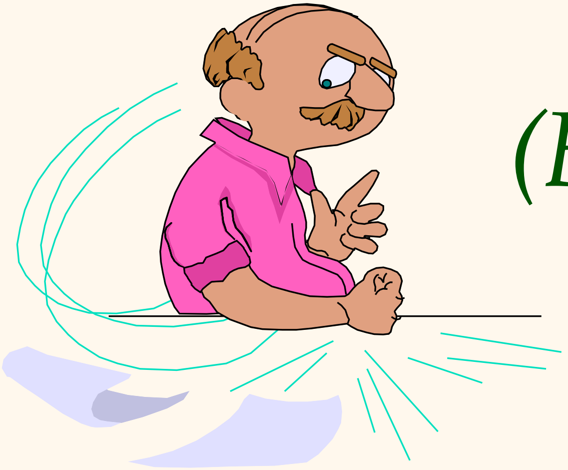
Introduction to Data Management



Lecture #2
(Big Picture, Cont.)

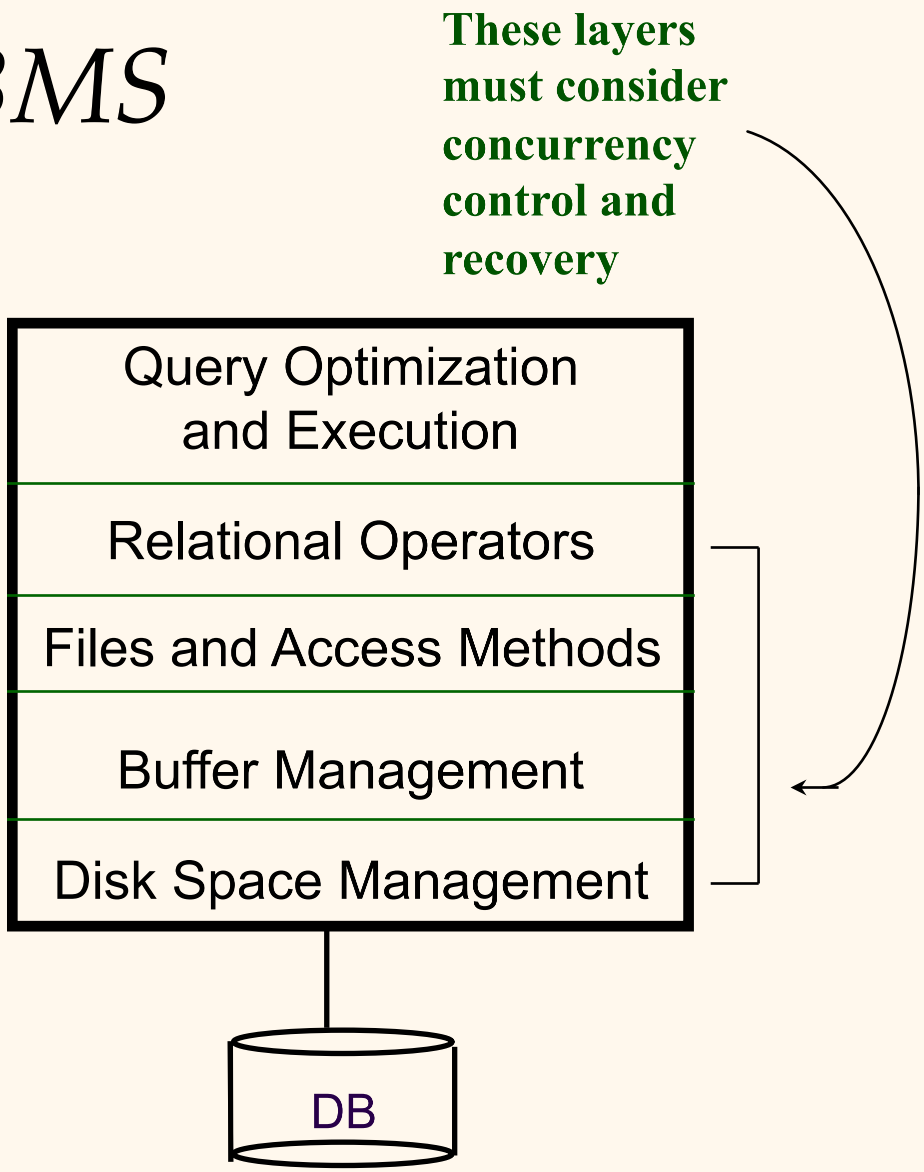
Instructor: Chen Li

Announcements

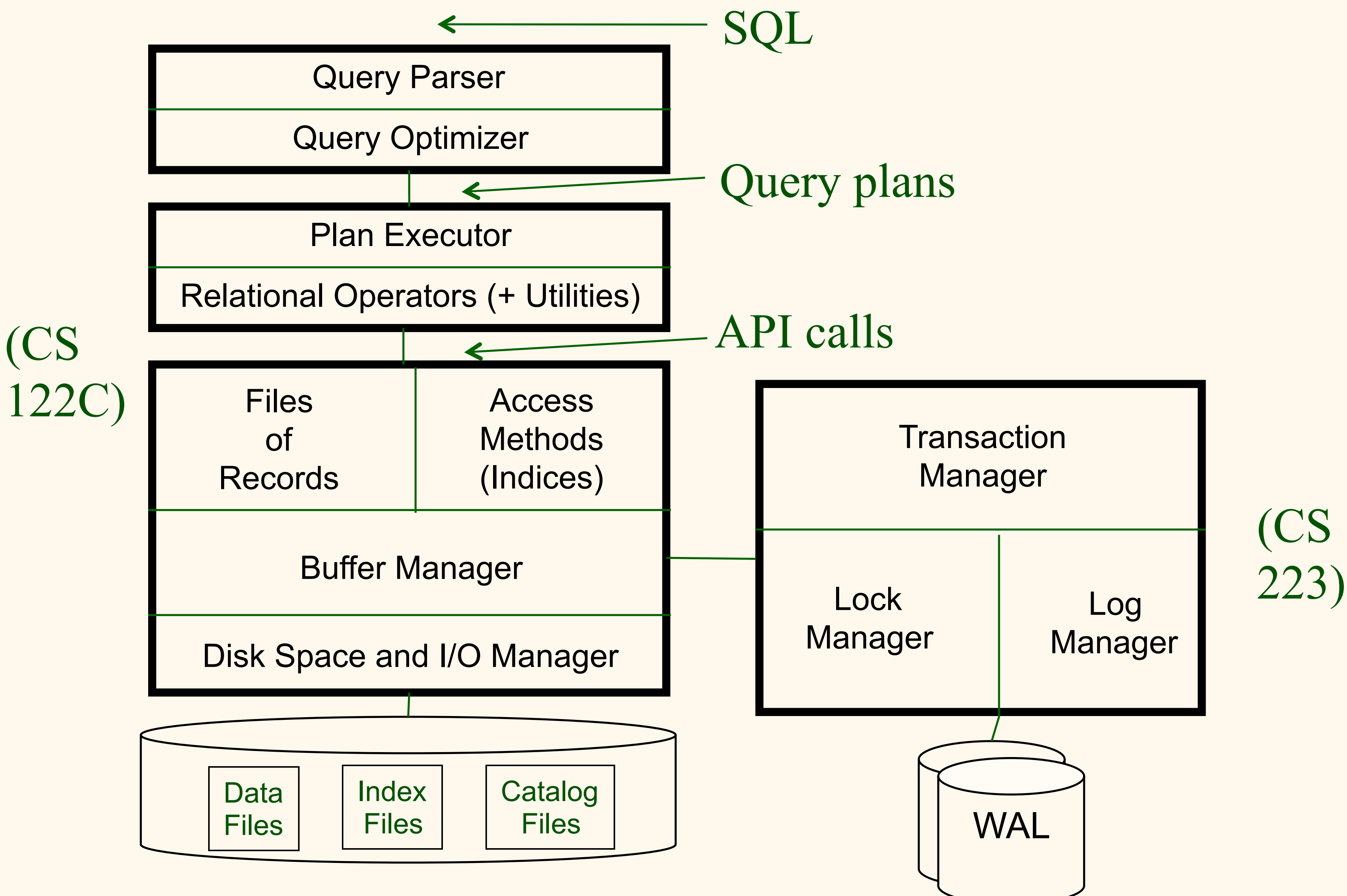
- We added 10 more seats to the class for students on the waiting list
- Deadline to drop the class: tomorrow (Friday)
- Sign up on Piazza today
- For general questions, use Piazza not email
  - Email: add “CS122A” in the subject
- Form a group of 3 students by coming Tuesday
  - Approval needed for groups of 1 or 2 people
- Discussion session switch allowed, and you need to figure out how to do it officially
- Assignment 1 to be released this week (you have two weeks to do it)

Structure of a DBMS

- 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



Components' Roles

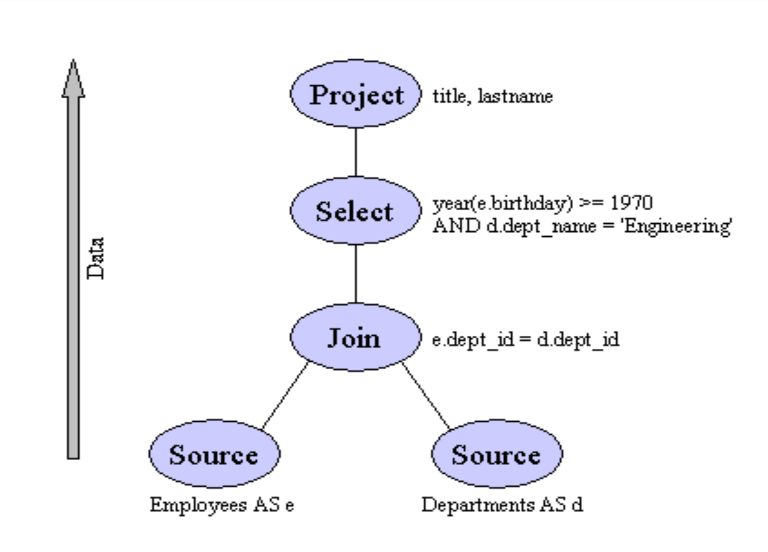
```
SELECT e.title, e.lastname
FROM Employees e, Departments d
WHERE e.dept_id = d.dept_id AND
      year (e.birthday >= 1970) AND
      d.dept_name = 'Engineering'
```

- Query Parser
  - Parse and analyze *SQL query*
  - Makes sure the query is valid and talking about tables, etc., that indeed exist
- Query optimizer (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 all 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 lookups based on field values
  - We'll look in more depth at *B+ tree* indexes in this class (as 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, …)
  - Remember: database data is *persistent (!)*

Components' Roles (continued)

- System Catalog (or "Metadata")
  - 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 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, …)
- 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 (also including "NoSQL"): 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
- Now - on to DB design…!

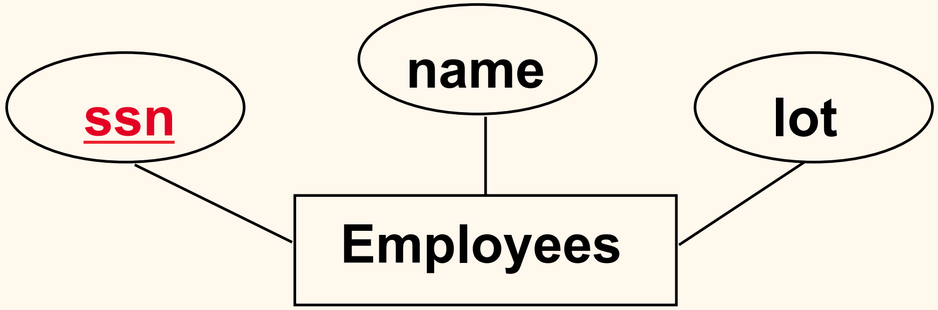
Entity–relationship (ER) model

- Peter Chen (March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems 1 (1): 9–36
- http://dl.acm.org/citation.cfm?doid=320434.320440
- Peter Chen: "*The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. It incorporates some of the important semantic information about the real world.*"

Overview of Database Design

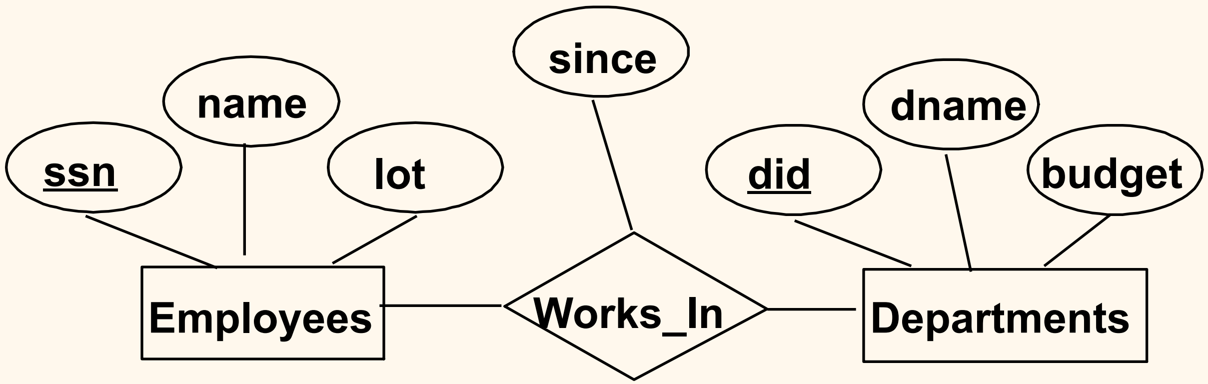
- *Conceptual design*: (*ER Model used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database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

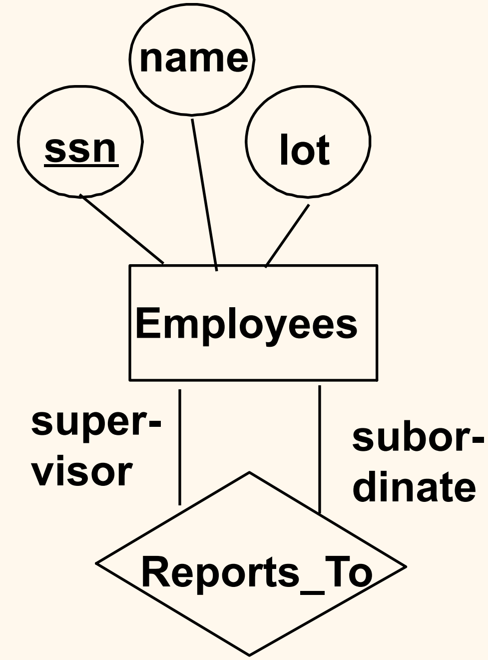
ER Model Basics



- *Entity:* Real-world object, distinguishable from all 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 get to ISA hierarchies… ☺)
  - Each entity set has a *key* (a unique identifier); this can be one attribute (an "atomic" key) or several attributes (a "composite" key)
  - Each attribute has a *domain* (similar to a data type).

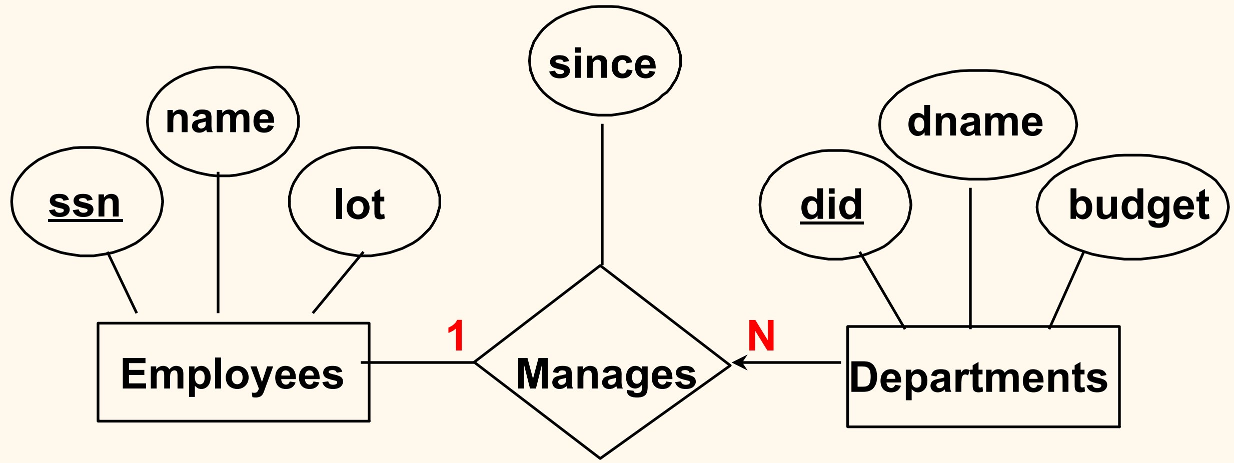
ER Model Basics (Contd.)



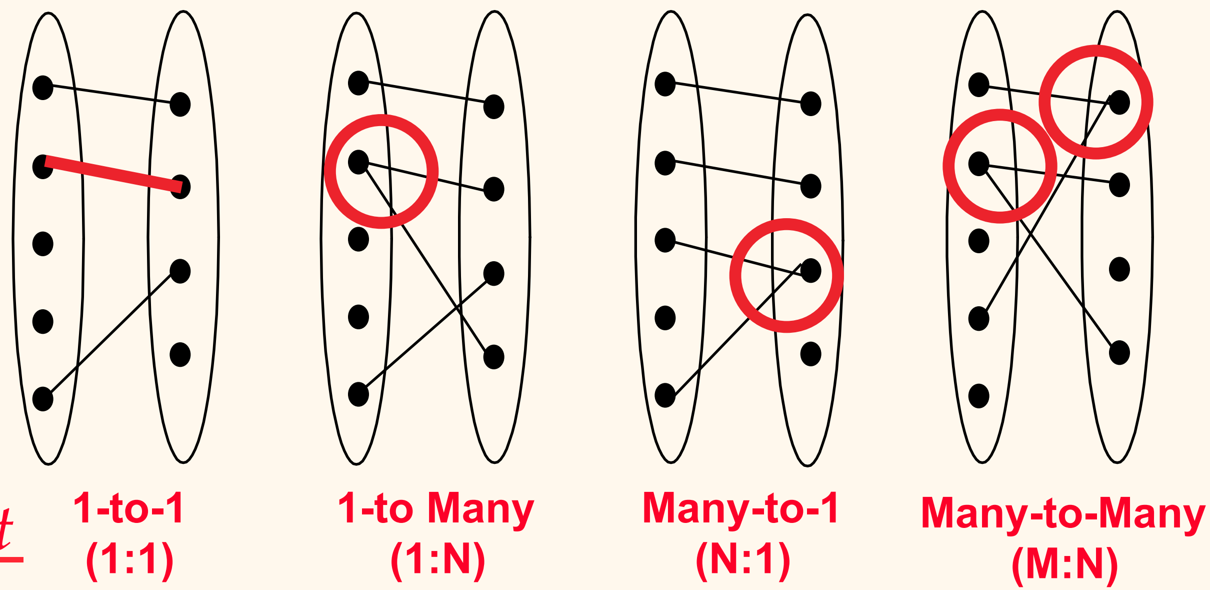


- *Relationship*: Association among two or more entities. E.g., Santa Claus works in the Toy department.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1:E1, ..., en:En
    - Same entity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

Cardinality Constraints

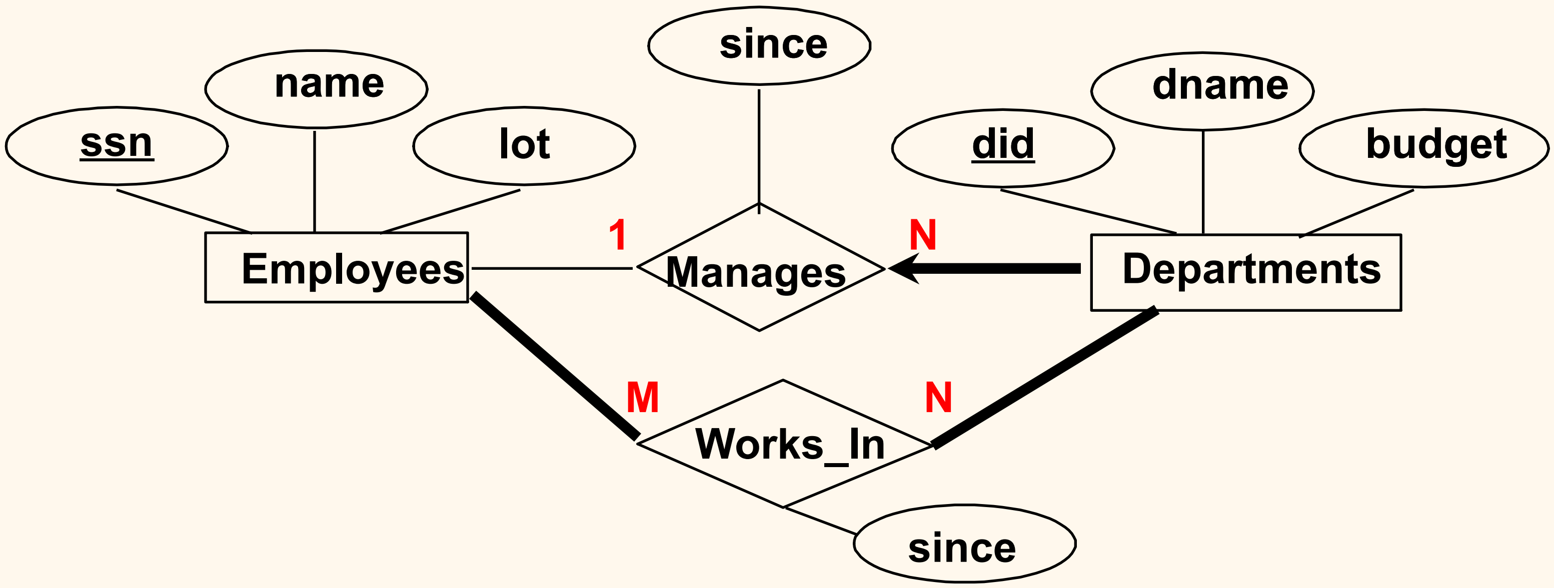


- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the *cardinality constraint* on Manages.

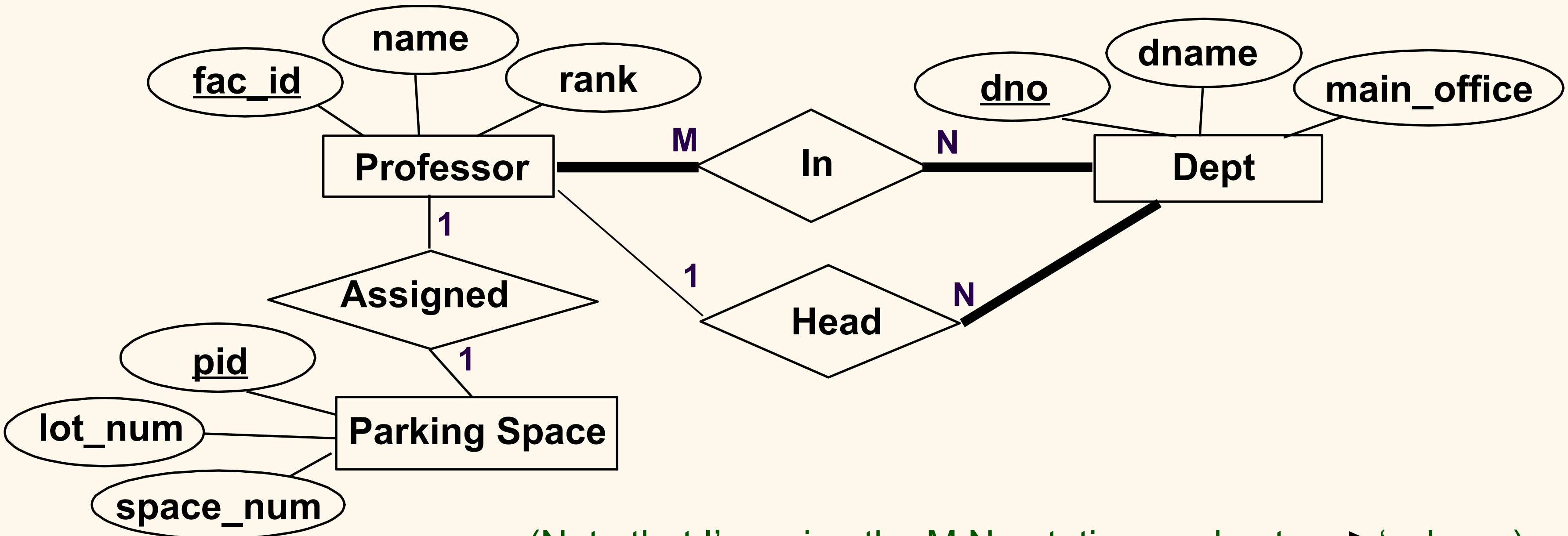


Participation Constraints

- 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 below *must* appear in an instance of the Manages relationship
    - Ditto for *both* Employees and Departments for Works_In



ER Basics: Another Example



(Note that I'm using the M:N notation, and not ➔'s, here.)

- Let's see if you can read/interpret the ER diagram above…! (☺)
  - What attributes are unique (i.e., identify their associated entity instances)?
  - What are the rules about (the much coveted) parking passes?
  - What are the rules (constraints) about professors being in departments?
  - And, what are the rules about professors heading departments?

Another Example (Cont'd.)

- Unique attributes:
  - *Professor*.fac_id, *Dept*.dno, *Parking Space*.pid
- Faculty parking:
  - 1 space/faculty, one faculty/space
  - Some faculty can bike or walk (☺)
  - Some parking spaces may be unused

NOTE: These things are all "rules of the universe" that are just being *modeled* here!

- Faculty in departments:
  - Faculty may have appointments in multiple departments
  - Departments can have multiple faculty in them
  - No empty departments, and no unaffiliated faculty
- Department management:
  - One head per department (exactly)
  - Not all faculty are department heads

Q: Can a faculty member head a department that he or she isn't actually in?

Another Example (Cont'd.)