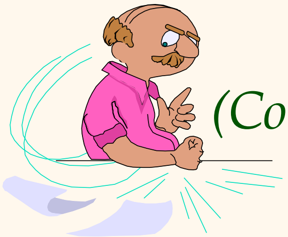
Introduction to Data Management



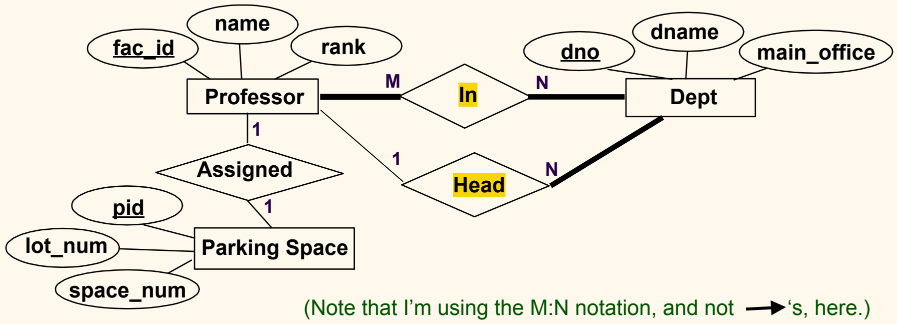
Lecture #3 (Conceptual DB Design)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- Today's plan:
  - More about logical DB design!
    - Basic ER concepts review and examples
    - Advanced ER concepts
- Reminders:
  - Sign up on Piazza! (Only 2/3 have done this.)
  - HW #1 and Project Part 1 coming mid-week!
  - First quiz in discussion section tomorrow!
- Any lingering *Q*'s from last time?

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.)

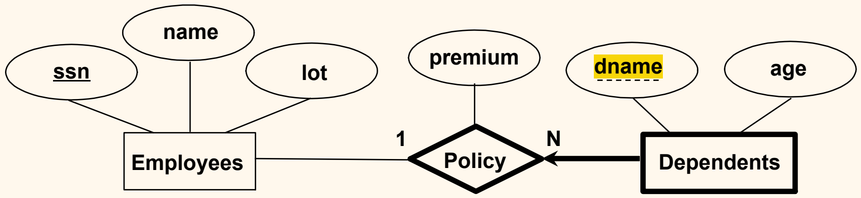
- Unique attributes:
  - *Professor*.fac_id, *Dept*.dno, *Parking Space*.pid
- Faculty parking:
  - 1 space/faculty, one faculty/space
  - Some faculty can bike or walk (☺)
  - Some parking spaces may be unused
- Faculty in departments:
  - Faculty may have appointments in multiple departments
  - Departments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

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

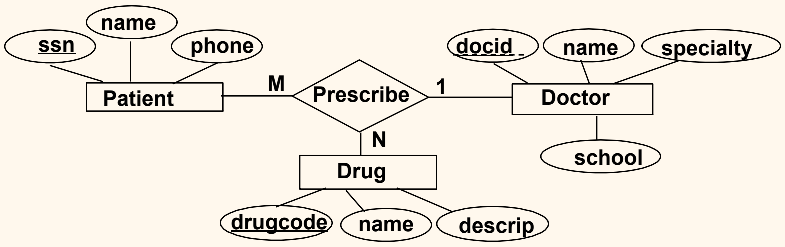
Q: Can a faculty member head a department that he or she isn't actually in?

Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have *total* participation in this *identifying* relationship set.
  - Dependent identifier is unique only within owner context (_ _ _ _), so its fully qualified key here is (ssn, dname)

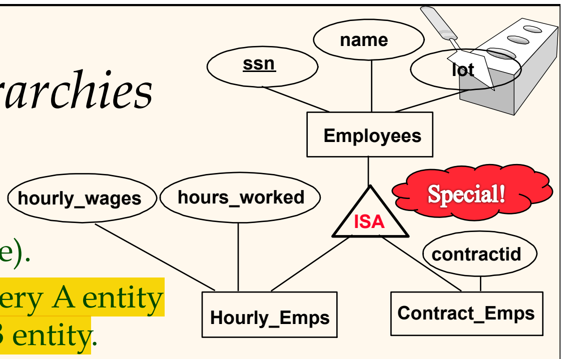


Ternary Relationships (and beyond)



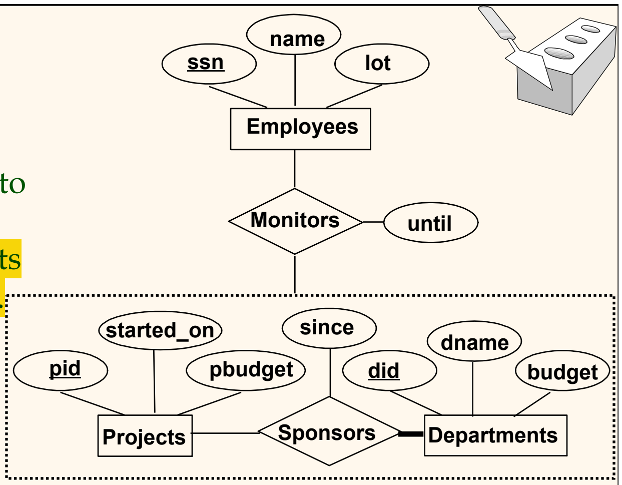
- A prescription here is a 3-way relationship between a patient, a doctor, and a drug
- As modeled above, a given patient/drug combination will always be associated with one doctor
  - *(General) note:* Relationship key = (entity keys)

ISA ("is a") Hierarchies



- As in C++ or other PLs ER attributes are inherited (including the key attribute).
- If we declare A **ISA** B, every A entity is also considered to be a B entity.
- *Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed* or ***disallowed)***
  - *Ex:* Hourly_Emps OVERLAPS Contract_Emps (else pick 1 of the 3 types)
- *Covering constraints: Does every Employees entity also have to be either an Hourly_Emps or a Contract_Emps entity? (Yes* or ***no)***
  - *Ex:* Hourly_Emps AND Contract_Emps COVER Employees (pick 1 of 2 vs. 1 of 3)
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify subclasses that participate in a relationship.
- Design: specialization (top-down), generalization (bottom-up)

Aggregation



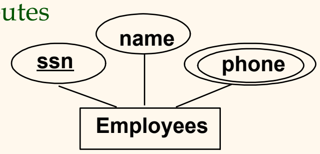
- Used when we have to model a relationship involving (entitity sets and) a *relationship set.*
  - Aggregation allows us to treat a relationship set as an entity set for purposes of participating in (other) relationships.

☛ *Aggregation vs. ternary relationship:*

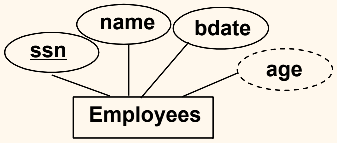
- Monitors is a distinct relationship; even has its own descriptive attribute.
- Also, can say that each sponsorship is monitored by at most one employee.

Some Advanced ER Features

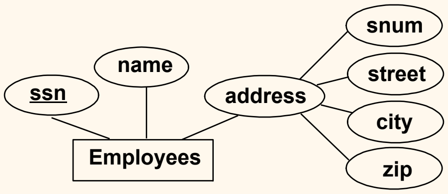
- Multi-valued (vs. single-valued) attributes



- Derived (vs. stored) attributes



- Composite (vs. atomic) attributes



NOTE: Can model (two of) these using additional entity and relationship types.

Conceptual Design Using the ER Model

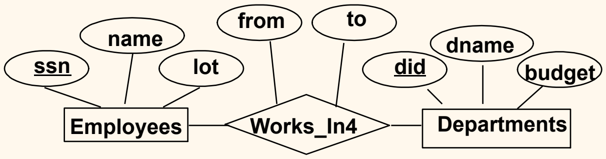
- Design choices:
  - Should a given concept be modeled as an entity or an attribute?
  - Should a given concept be modeled as an entity or a relationship?
  - Characterizing relationships: Binary or ternary? Aggregation? …
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But, not all constraints cannot be captured by ER diagrams. (*Ex:* Department heads from earlier…!)

Entity vs. Attribute

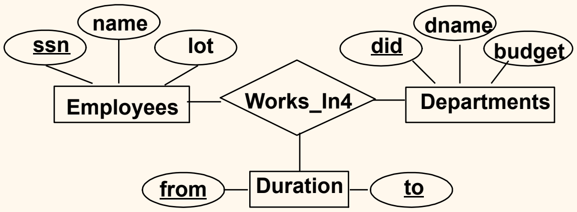
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued w/o advanced modeling goodies).
  - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic w/o advanced modeling goodies).
  - If the address itself is logically separate (e.g., the property that's located there) and refer-able, it's rightly an entity in any case!

Entity vs. Attribute (Cont'd.)

- Works_In4 does not allow an employee to work in a department for two or more periods.

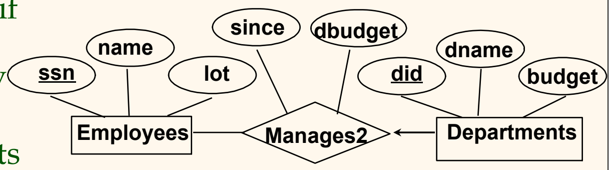


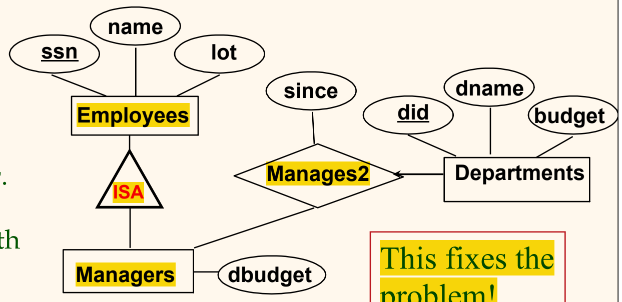
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Can be accomplished by introducing new entity set, Duration.



Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.

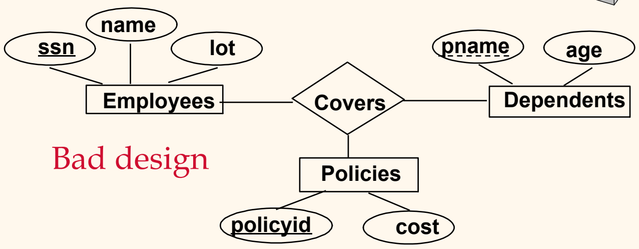




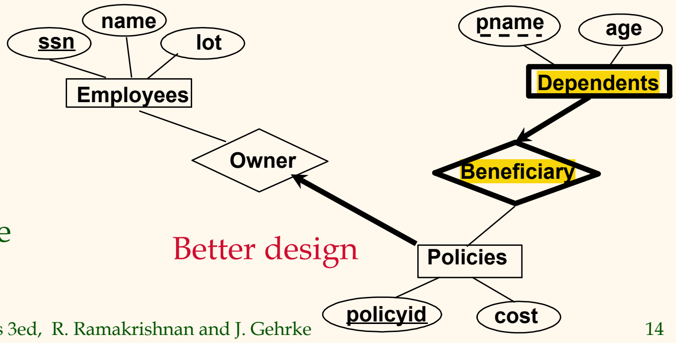
This fixes the problem!

Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- *Q:* What are the additional constraints in the 2nd diagram?



Bad design



Better design

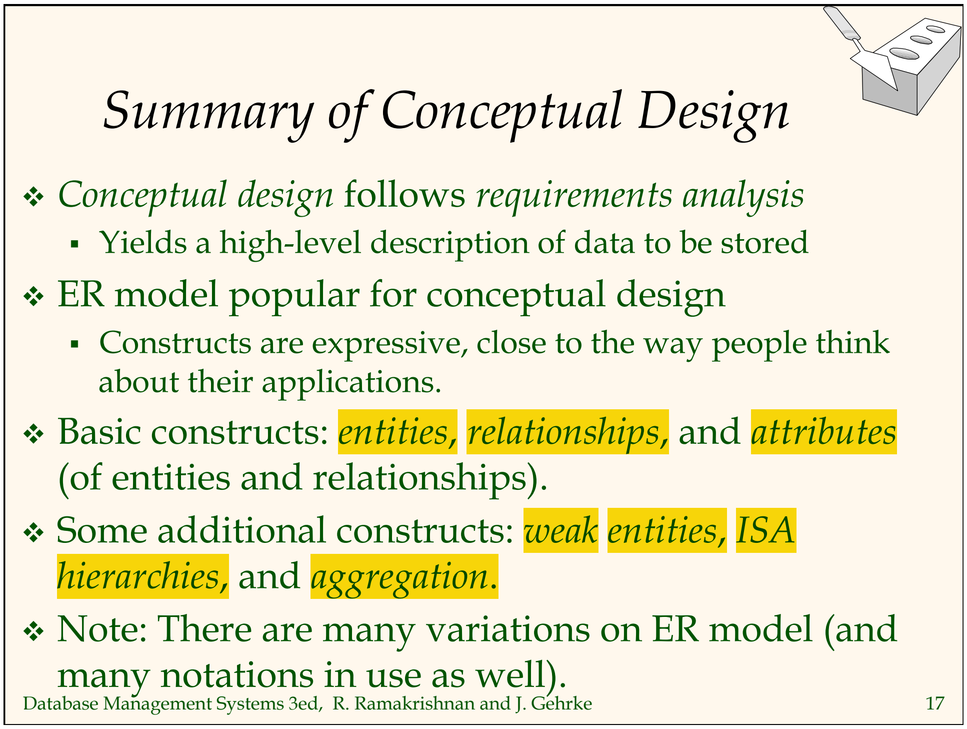
Binary vs. Ternary Relationships (Cont'd.)

- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute *qty*. No combination of binary relationships is an adequate substitute:
  - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
  - How do we record *qty*?

Datatase Design Process (Flow)

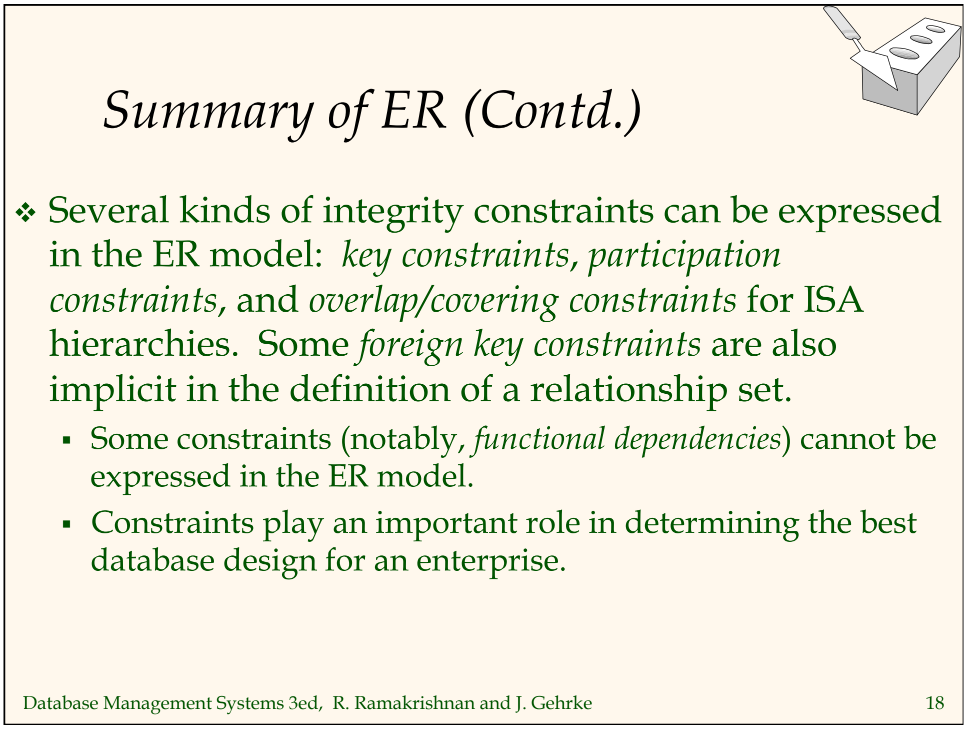
- Requirements Gathering (interviews)
- Conceptual Design (using ER)
- Platform Choice (which DBMS?)
- Logical Design (for target data model)
- Physical Design (for target DBMS, workload)
- Implement (and test, of course ☺)

(Expect backtracking, iteration, and also incremental adjustments!)



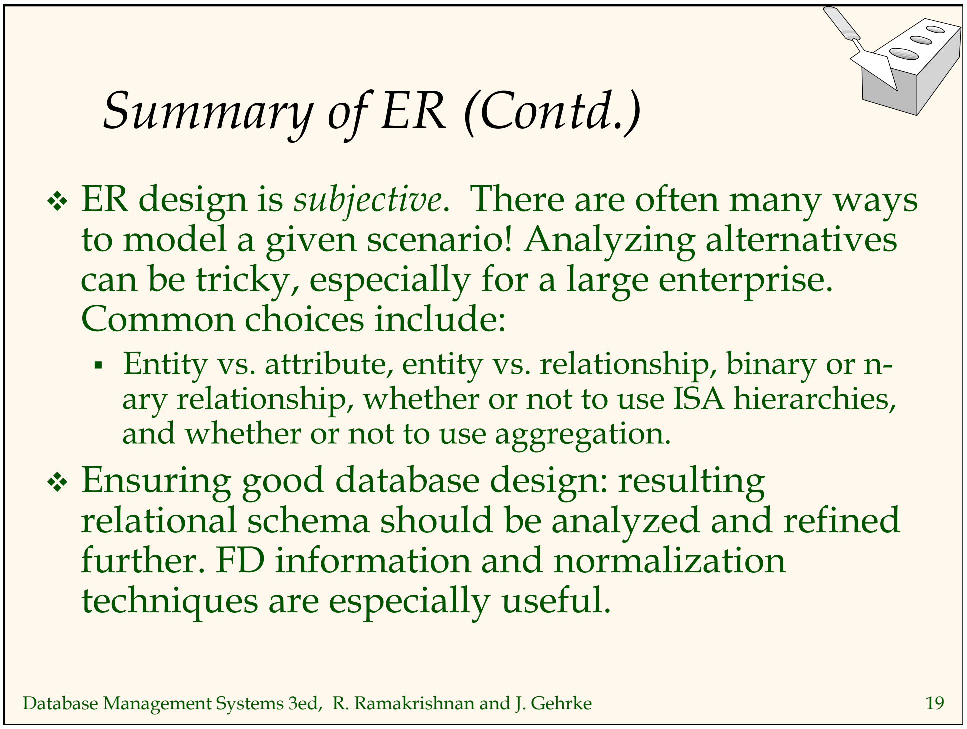
Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities, relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities, ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model (and many notations in use as well).



Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.



Summary of ER (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.