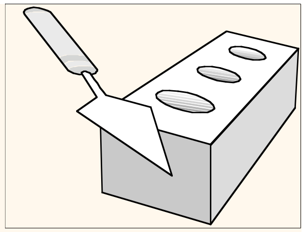
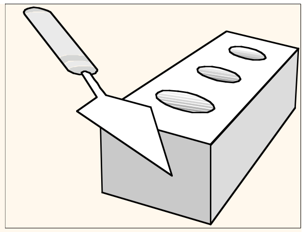


# *The Entity-Relationship Model*

Chapter 2

Slides modified by Rasmus Pagh for
Database Systems, Fall 2006
IT University of Copenhagen

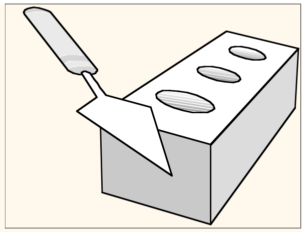
## *Today's lecture*



- ❖ Data modeling overview.
- ❖ The entity-relationship (ER) data model.
- ❖ ER design choices.

- ❖ Running example: ER modeling case study based on RG exercise 2.6.

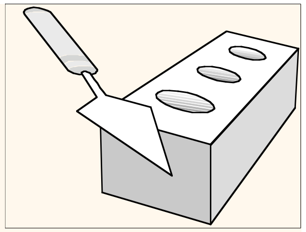
## *Database design process*



- Starts with requirements analysis

- Ends with a (relational) database schema
  (plus type info, constraints, indexes, triggers, ...)

# *Database design process*

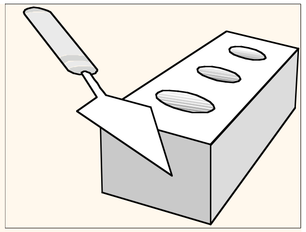


- ❖ Starts with requirements analysis

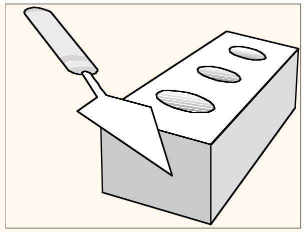
- ❖ Conceptual data modeling - ER model (today)
- ❖ Logical data modeling - relations (next time)
- ❖ Schema refinement, physical design (later in course)

- ❖ Ends with a (relational) database schema (plus type info, constraints, indexes, triggers, ...)

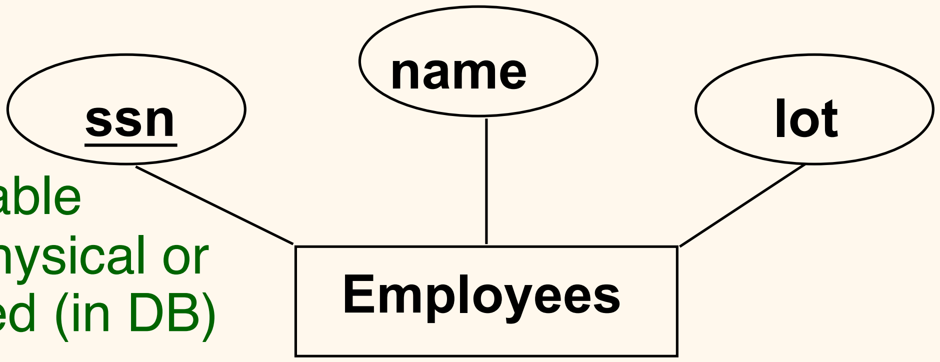
## *Conceptual and logical design*



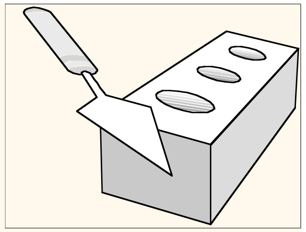
- *Conceptual design*: *(ER Model is used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database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



# *ER Model Basics*



- *Entity:* "Object" distinguishable from other objects. Can be physical or abstract. 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 *(primary) key* - one or more attributes that uniquely identify its entities.
  - Each attribute has a *domain*. In (pure) ER modeling, the domain cannot be sets of any kind (*atomicity*).

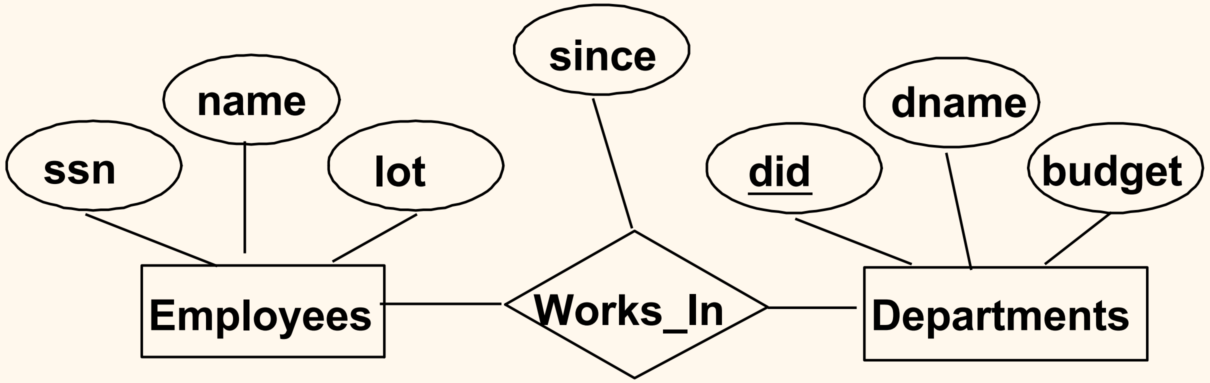


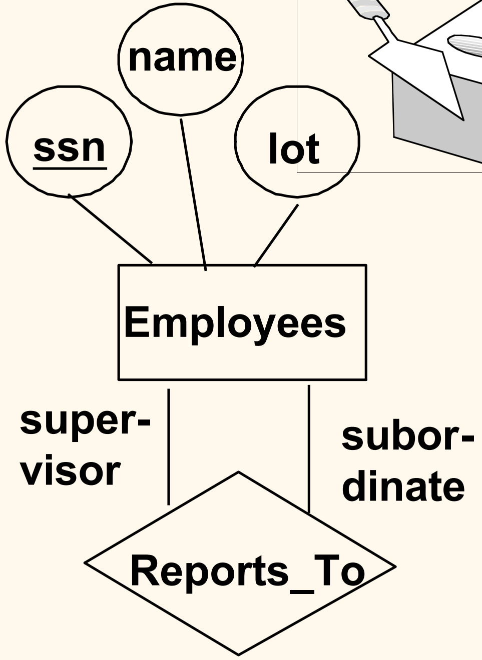
## *Case study (1/4)*

All information related to Dane County Airport is to be organized using a DBMS, and you have been hired to design the database.Your first task is to organize the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.

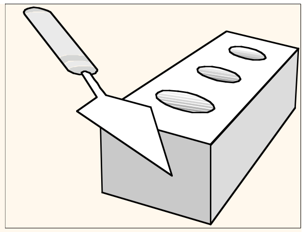
# ER Model Basics (Contd.)



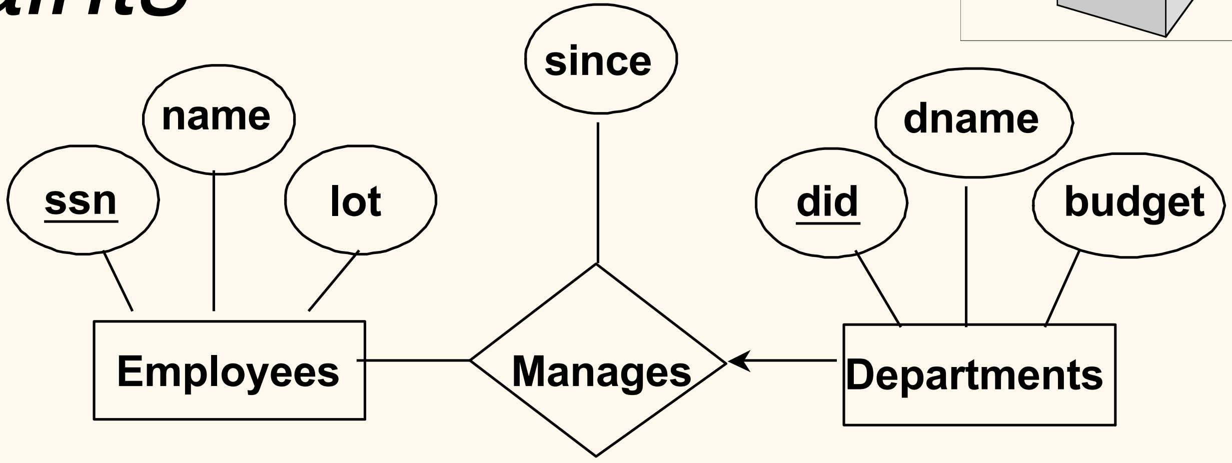


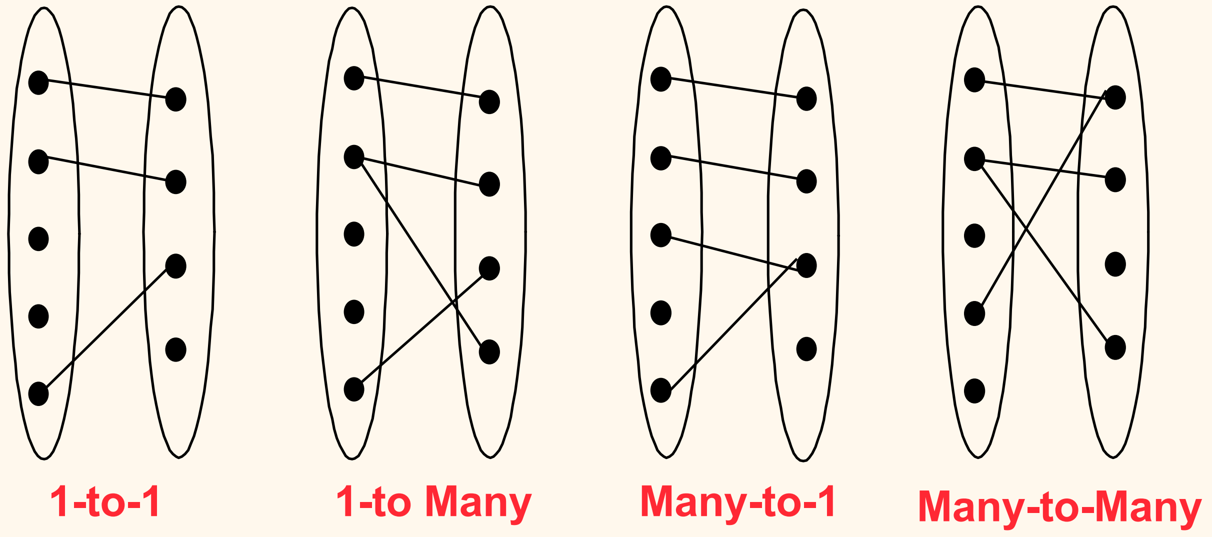
- *Relationship*: Association among two or more entities. E.g., Attishoo works in Pharmacy department.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1   E1, ..., en   En
    - Same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

# *Key Constraints*



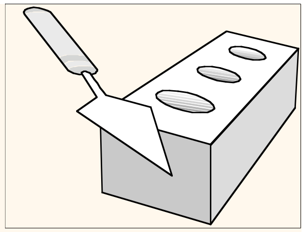
- Consider Works_In: An employee can work in many departments; a dept can have many employees.
- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.





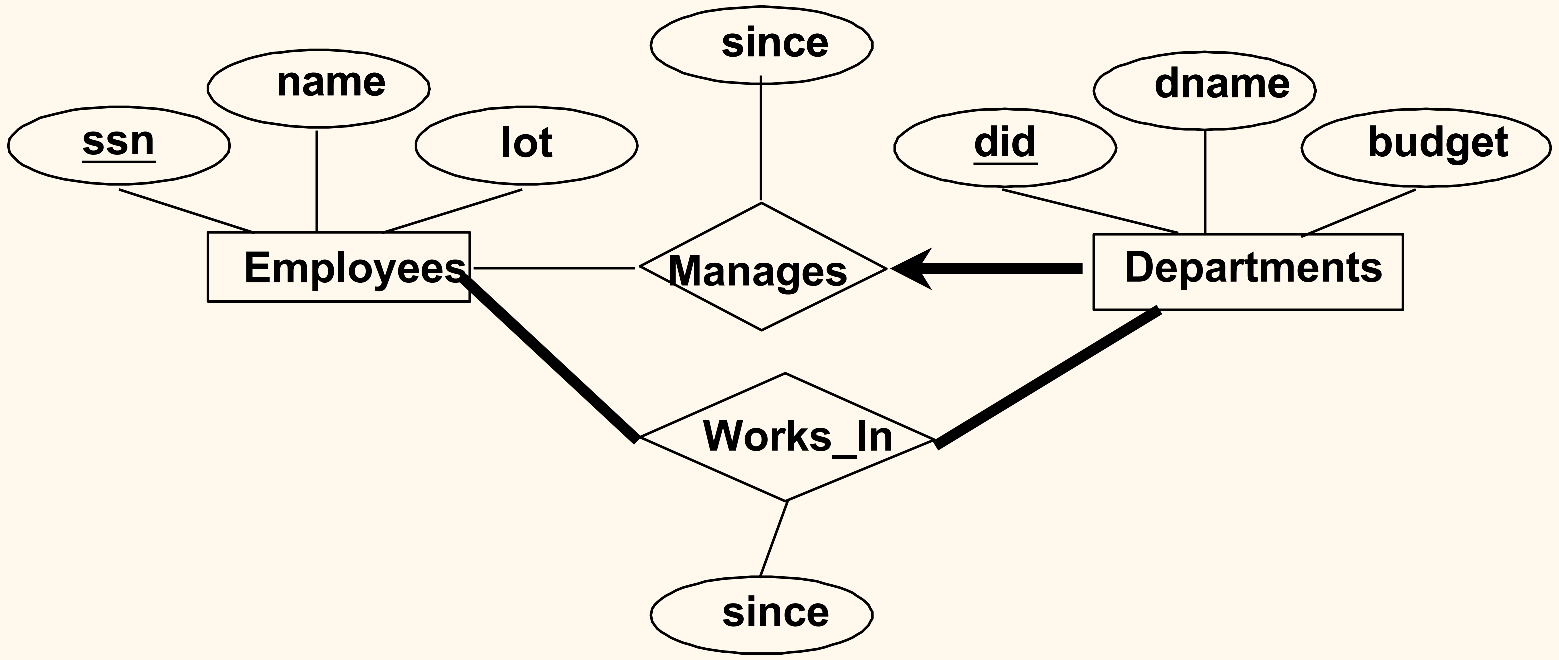
## Case study (2/4)

- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.

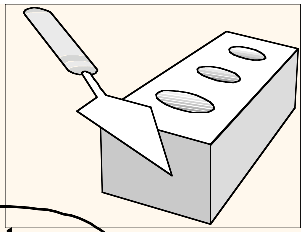


## *Participation Constraints*

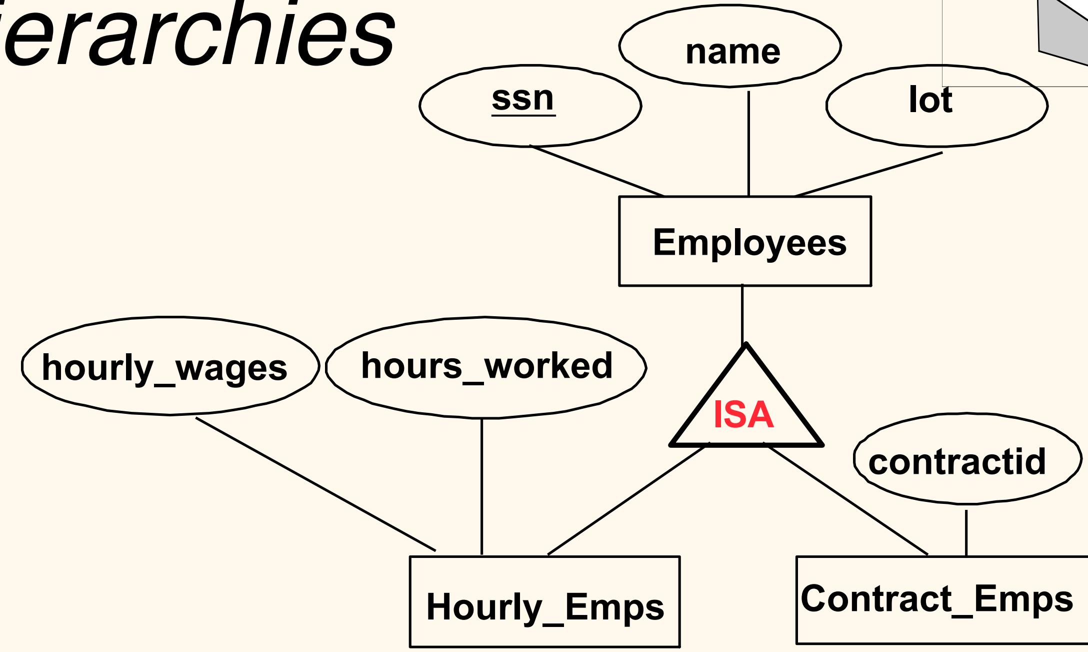
- Does every department have a manager?
  - If so, this is a *participation constraint*: the participation of Departments in Manages is said to be *total* (vs. *partial*).
    - Every Departments entity must appear in an instance of the Manages relationship.



# *ISA (`is a') Hierarchies*



- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.



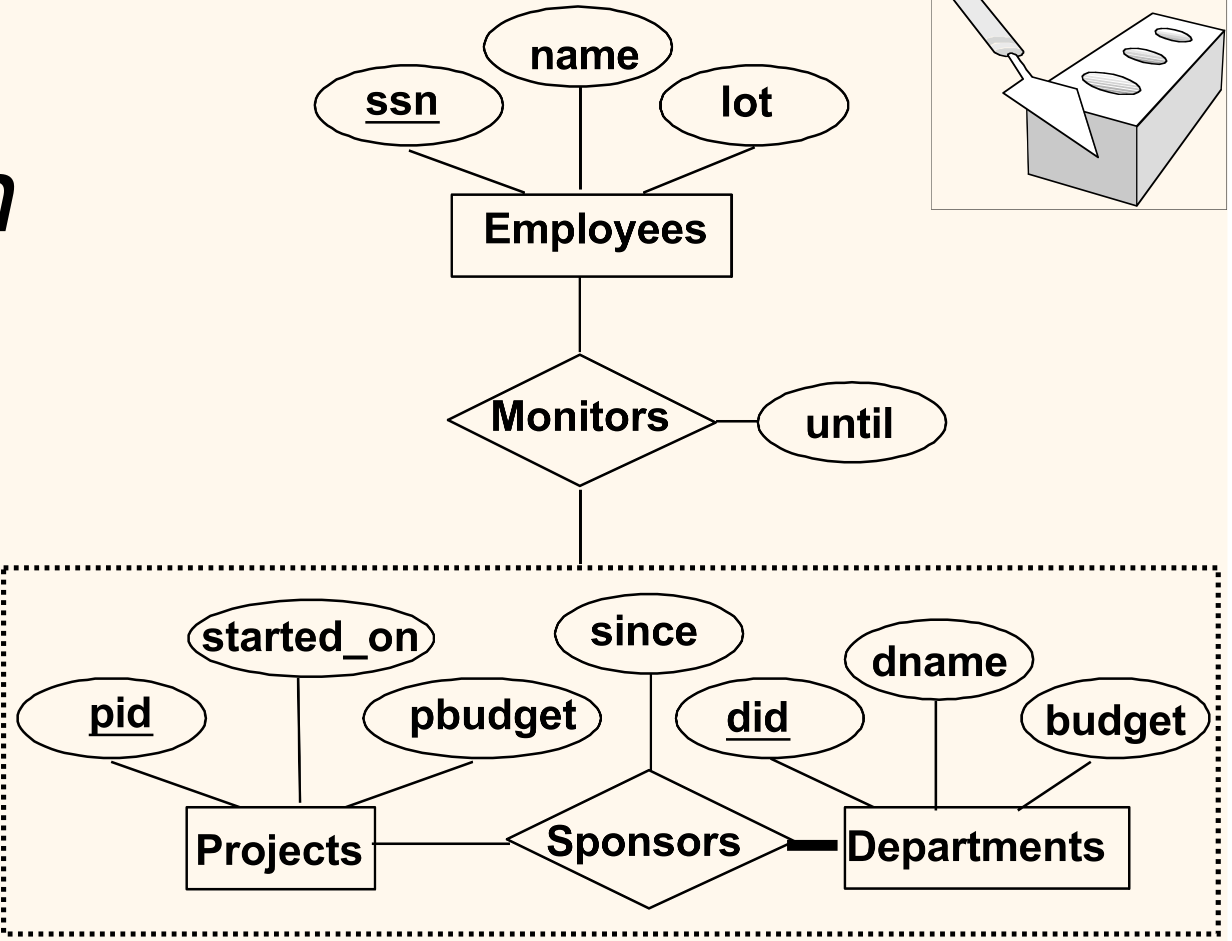
- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)
- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/No)*
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entitities that participate in a relationship.

## *Case study (3/4)*

- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee.
- You can assume that each employee is uniquely identified by a social security number.

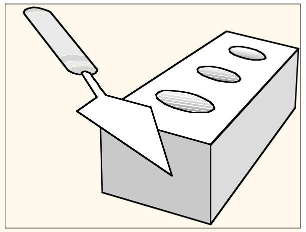
# *Aggregation*

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



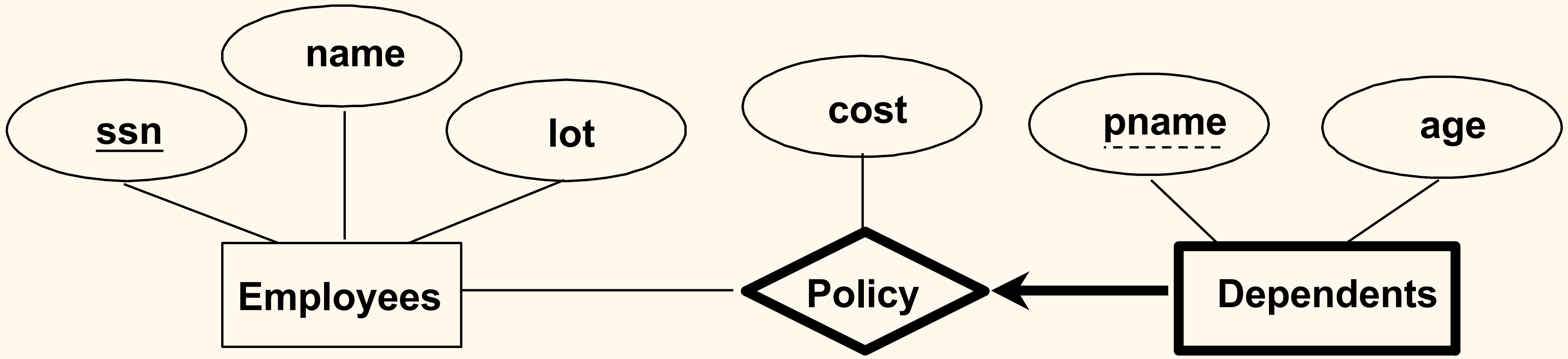
☛ *Aggregation vs. ternary relationship*:

- Monitors is a distinct relationship, with a descriptive attribute.
- Also, can say that each sponsorship is monitored by at most one employee.

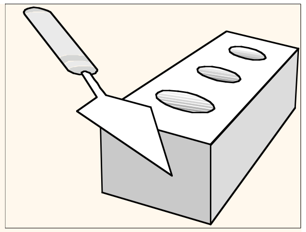


## *Weak Entities*

- 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.

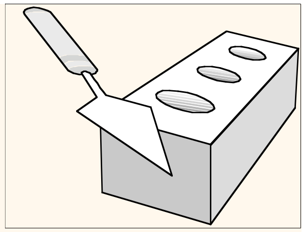


## *Case study (4/4)*



- The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score.
- The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a given test. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score the airplane received on the test.

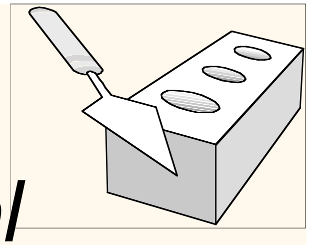
# *What is a good data model?*



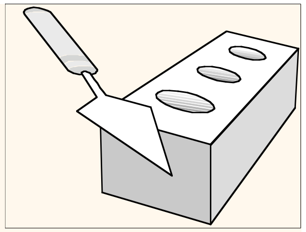
Characteristics of a good data model:

- It is easy to write correct and understandable queries
  - elements of the model have an intuitive meaning
  - the model is as simple as possible, but not simpler!
- No change is needed for minor changes in the problem domain (sufficiently abstract).
- If the problem domain changes significantly, it is easy to modify the data model and associated programs (flexible).
- Sometimes it is also necessary to consider the data model's impact on the efficiency of database operations.

## *Conceptual Design Using the ER Model*



- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

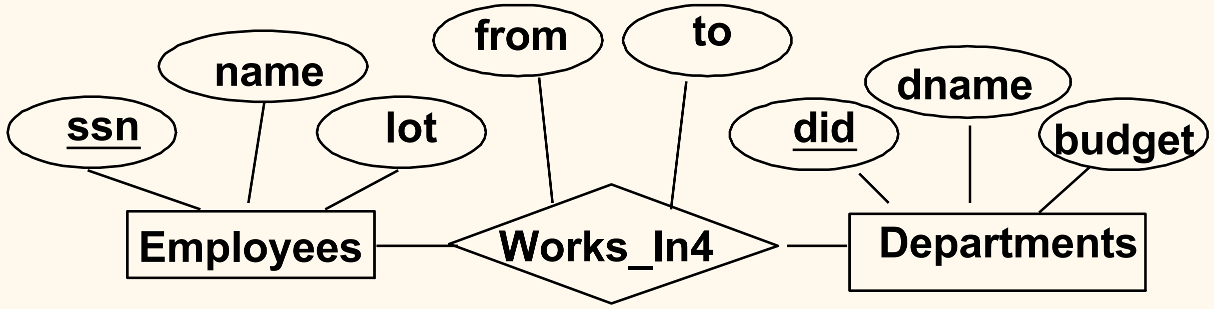


## *Entity vs. Attribute*

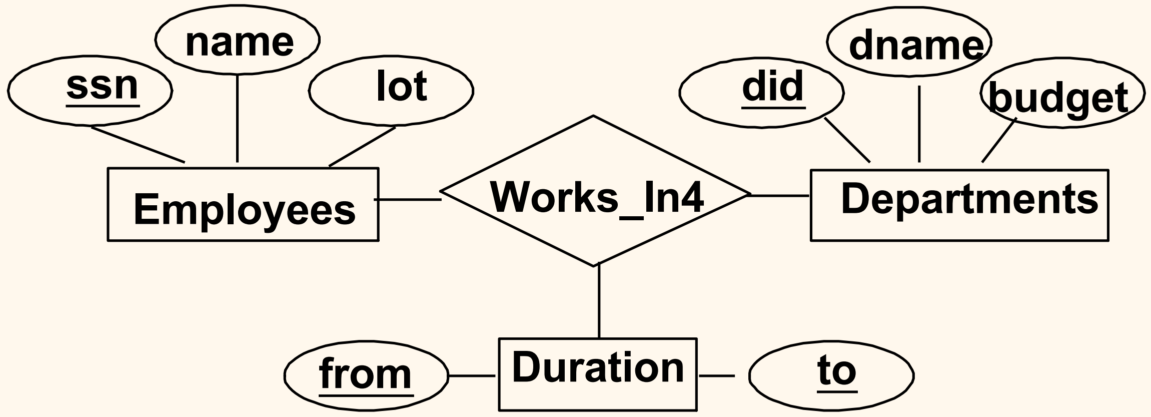
- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
  - 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).

# *Entity vs. Attribute (Contd.)*

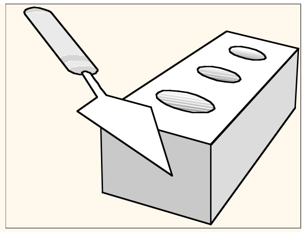
- Works_In4 does not allow an employee to work in a department for two or more periods.



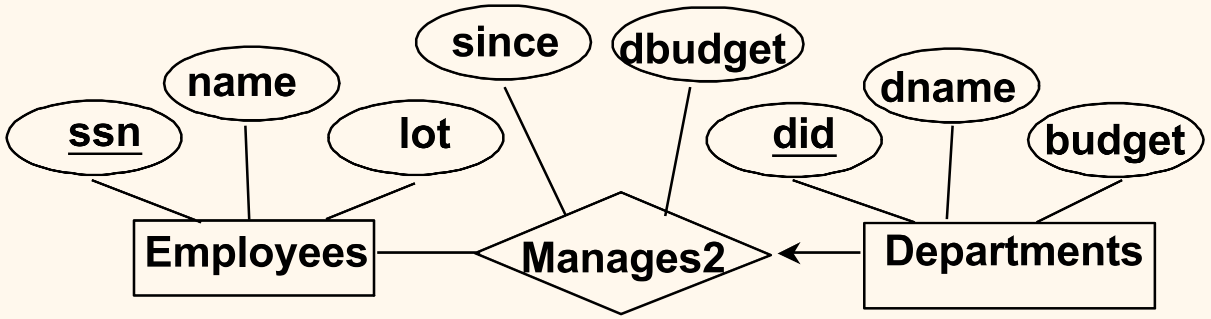
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.



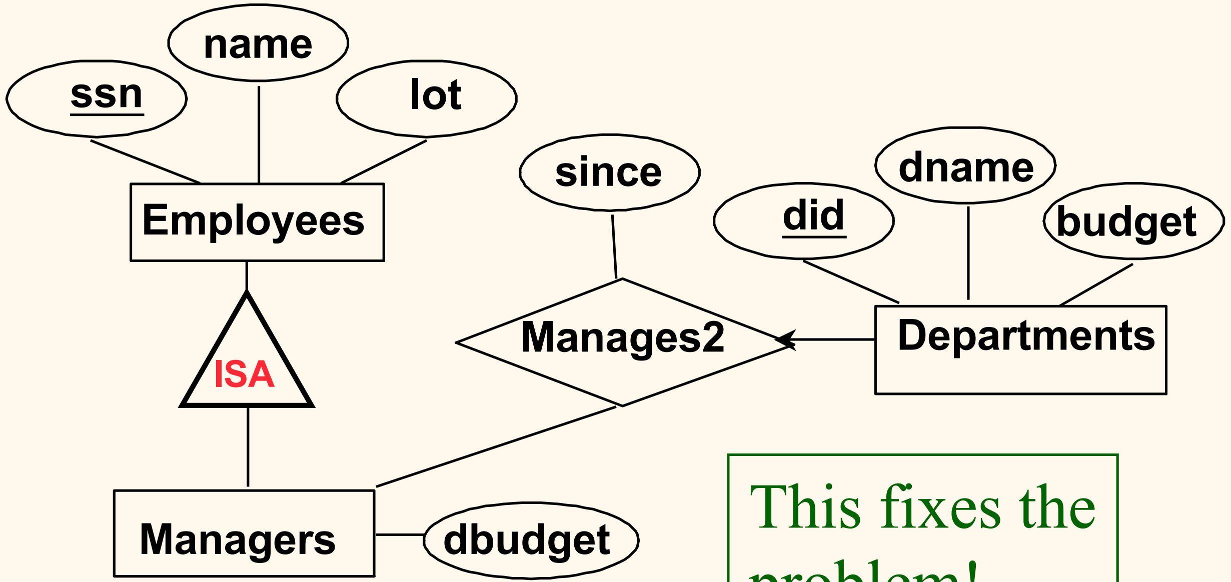
# *Entity vs. Relationship*



- First ER diagram OK if a manager gets a separate discretionary budget for each dept.



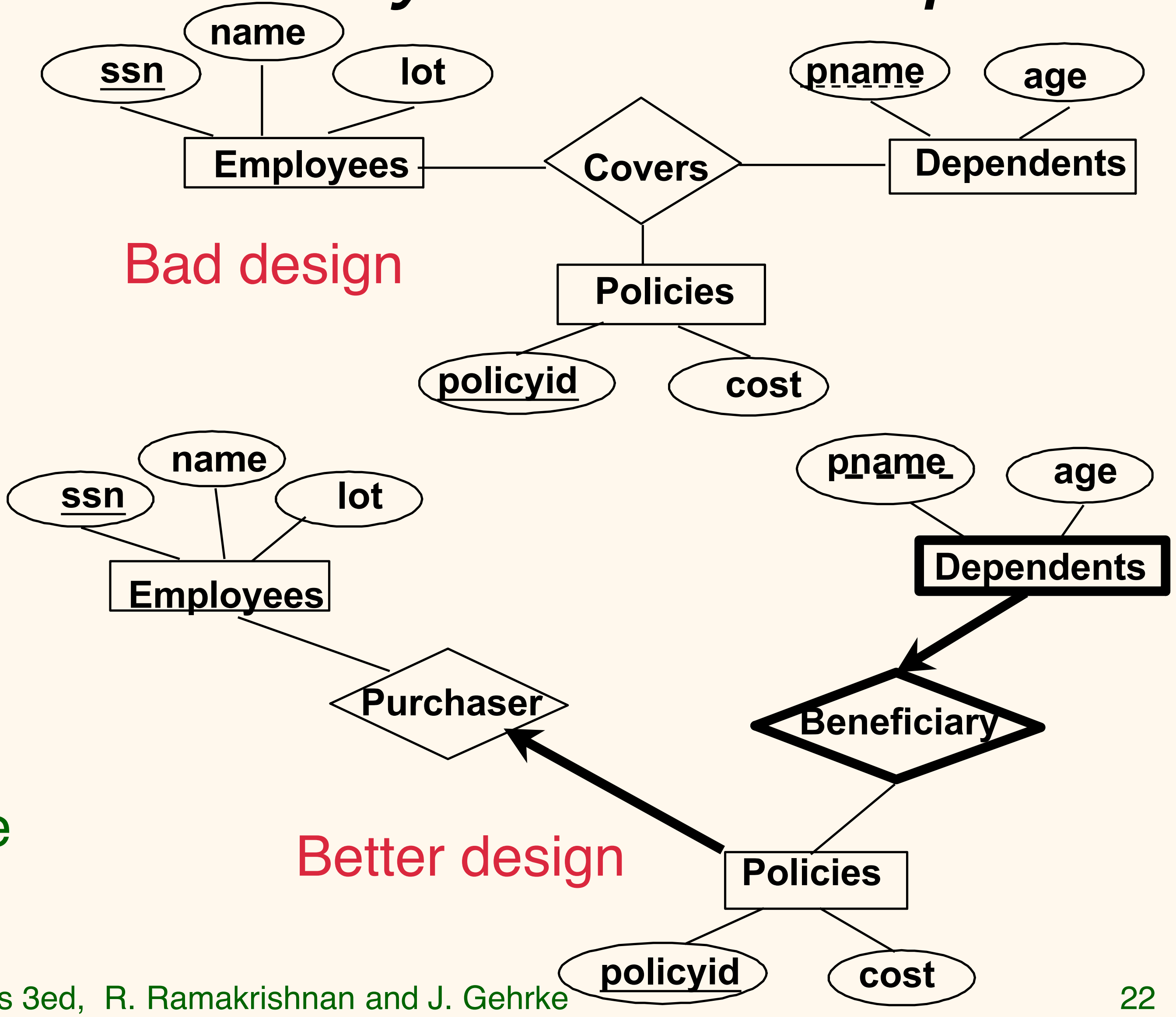
- What if a manager gets a discretionary budget that covers all managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.



This fixes the problem!

# *Binary vs. Ternary Relationships*

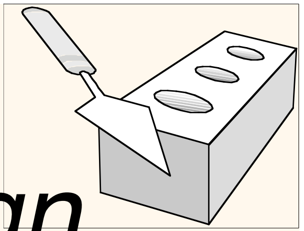
- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints in the 2nd diagram?



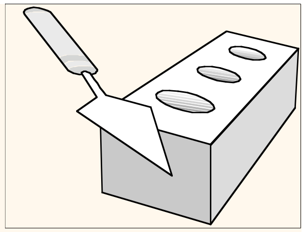
## *Binary vs. Ternary Relationships (Contd.)*

- 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*?

# *Summary of Conceptual Design*

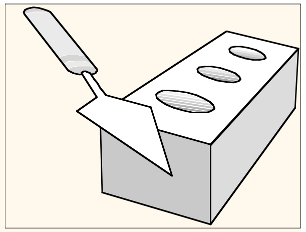


- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes*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



## *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.