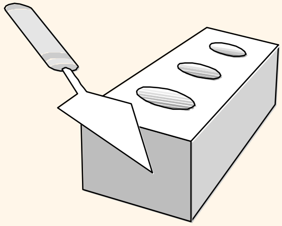
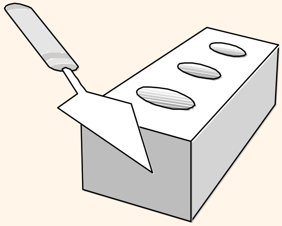
The Entity-Relationship Model

Chapter 2



Overview of Database Design

- Building a conceptual database design should be done after informal discussions with the customers.
- Your design should reflect all the details and operations that your customer need
- Examples of customers are: University databases, Company database, Banking, Airline reservation,…



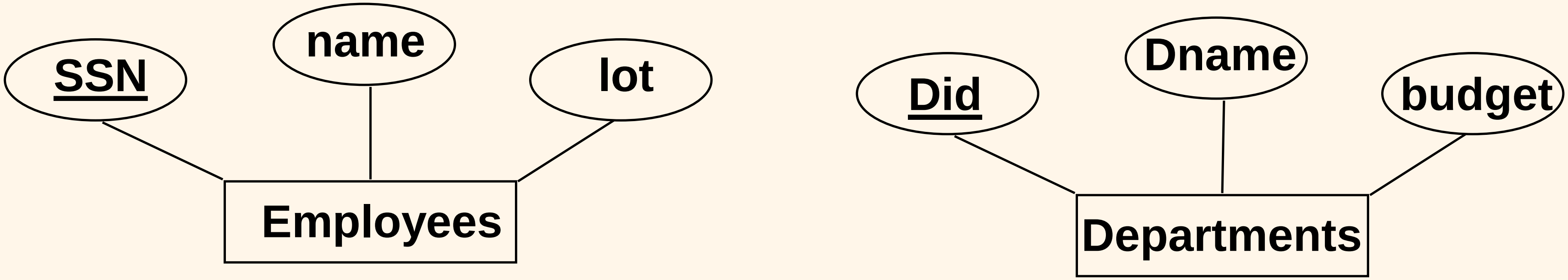
Overview of Database Design

- *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 relational schema.

ER Model Basics

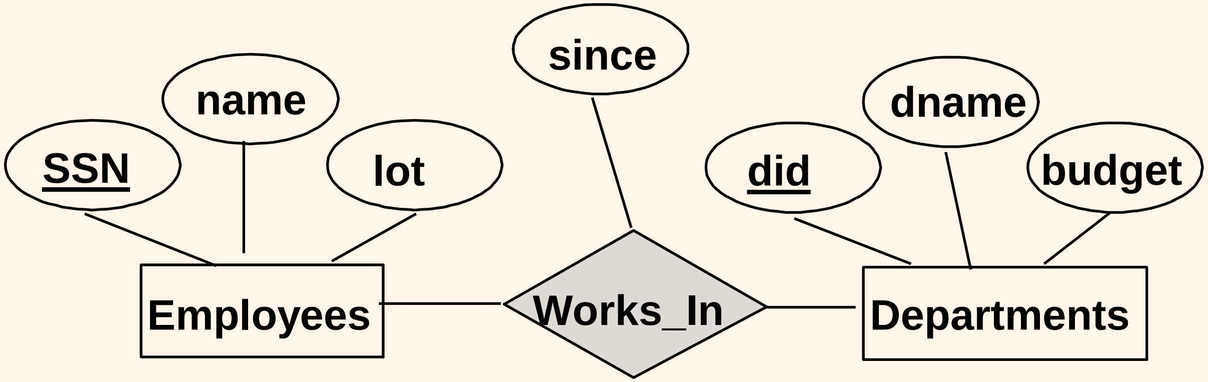
- *Entity:* Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)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

ER Model Basics (Entity)



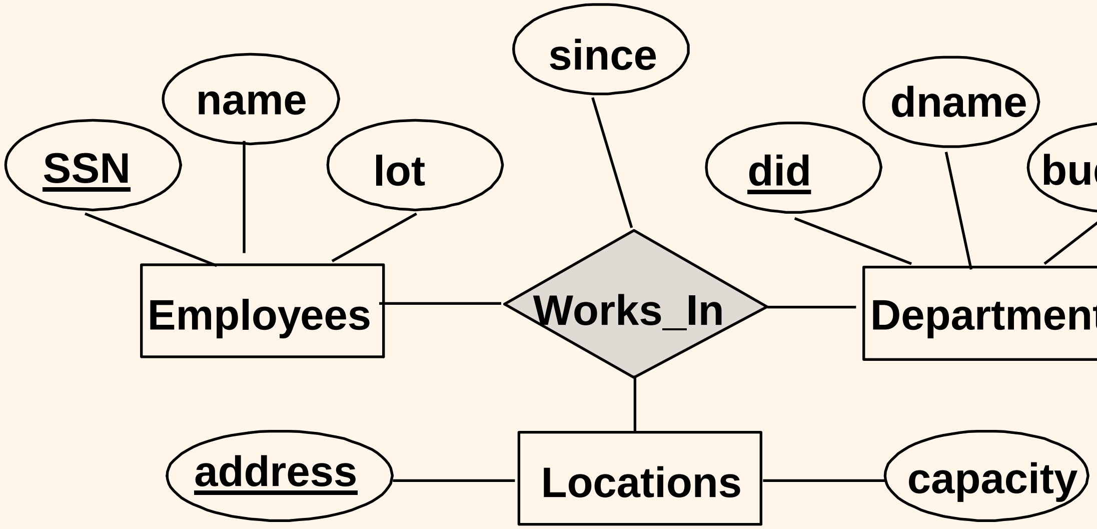
- Based on the discussion with our customers, we find that:
  - Each employee has there attributes: SSN, name, and parking lot
  - Each department has three attributes: number, name, and budget
- Employees can be uniquely identified by their SSN (SSN), departments can be identified by their numbers (Did).
- A *key* is a minimal set of attributes whose values uniquely identify an entity in the set
  - If there are more than one *candidate* key, we designate one of them as the *primary* key

ER Model Basics (Relationship)



- *Relationship*: Association among two or more entities.
  - E.g., Attishoo works in Pharmacy department.
- *Relationship Set*: Collection of similar relationships.
- A relationship set may have *descriptive* attributes:
  - E.g., John starts working in the CS department since August 2005

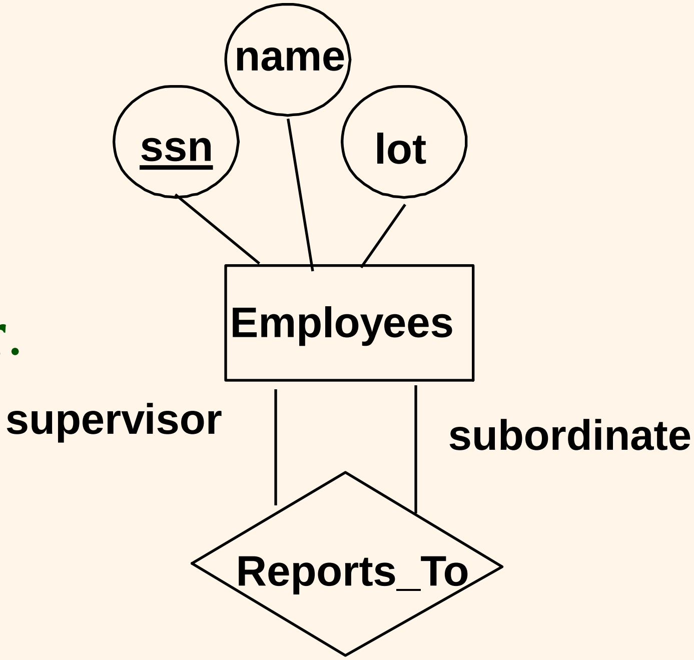
N-ary Relationship



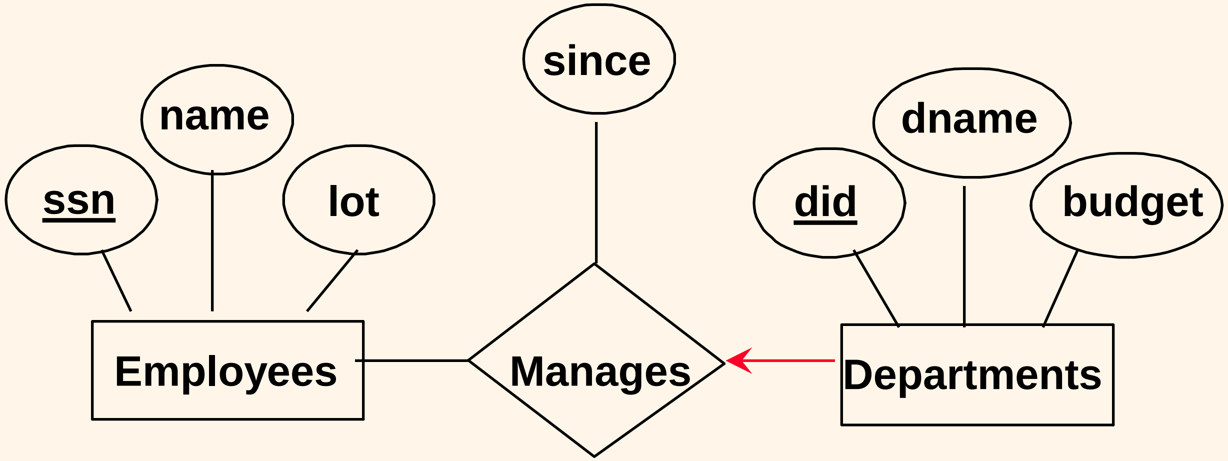
- An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1   E1, ..., En

More on Relationship Set

- The entity set in a relation may not be distinct
  - An Employees can supervise another employee
  - In this case, a role indicator should be labeled (supervisor. subordinate)
- Same entity set could participate in different relationship sets, or in different "roles" in same set.

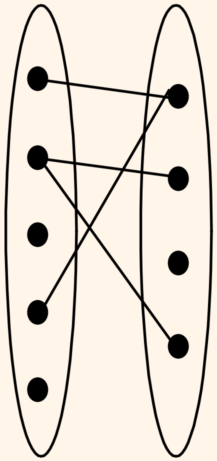


Key Constraints

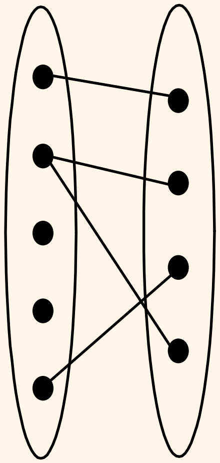


- Each dept has at most one manager, according to the *key constraint* on Manages.

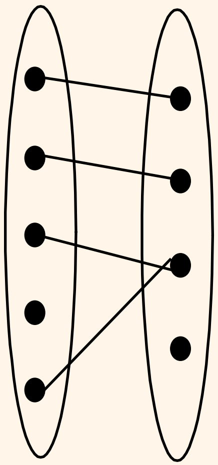
Key Constraints (Cont.)



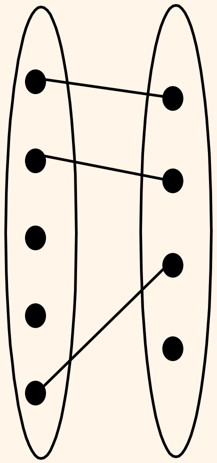
Many-to-Many



1-to Many



Many-to-1



1-to-1

Works_In: An employee can work in many departments; a department can have many employees.

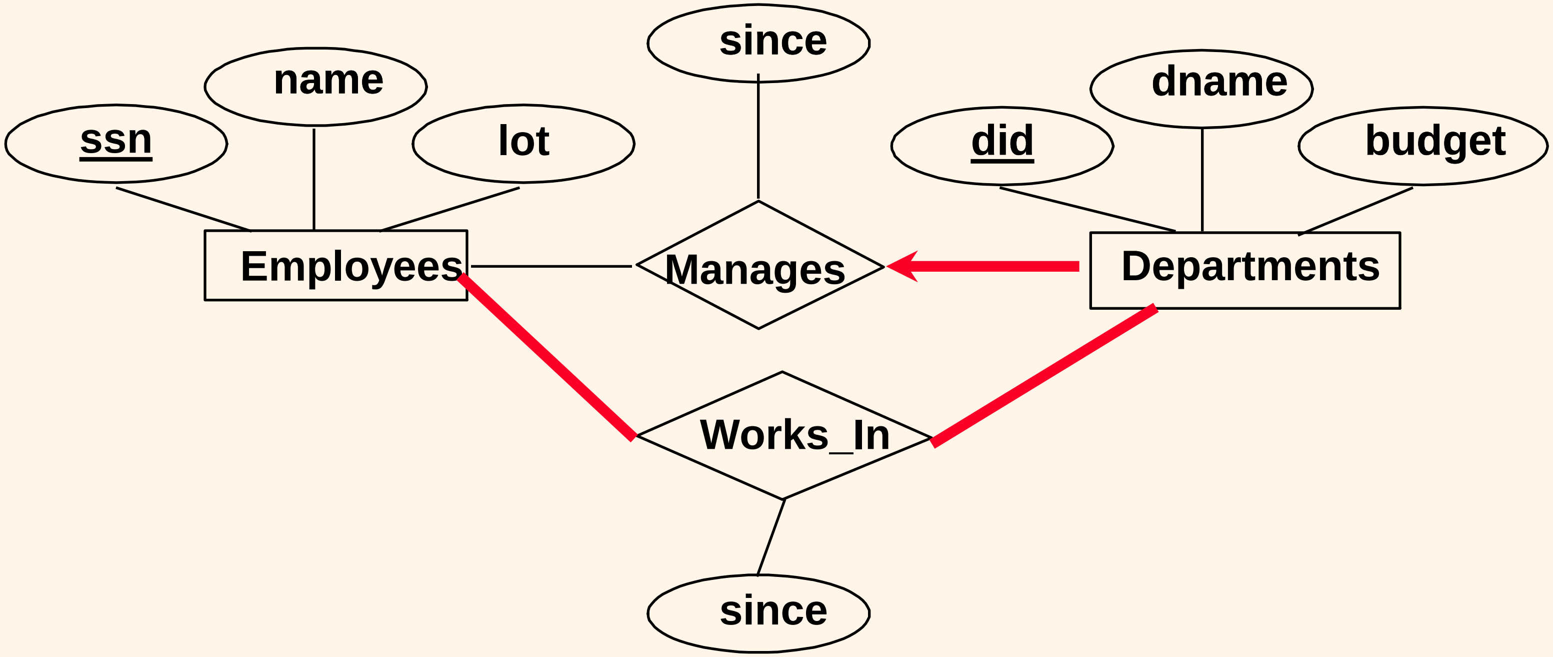
Manages: A department is managed only by one employee. One employee can manage many departments

Works_In: If we add a constraint that each employee can work in only one department

Manages: If we add a constraint that one employee can manage only one department

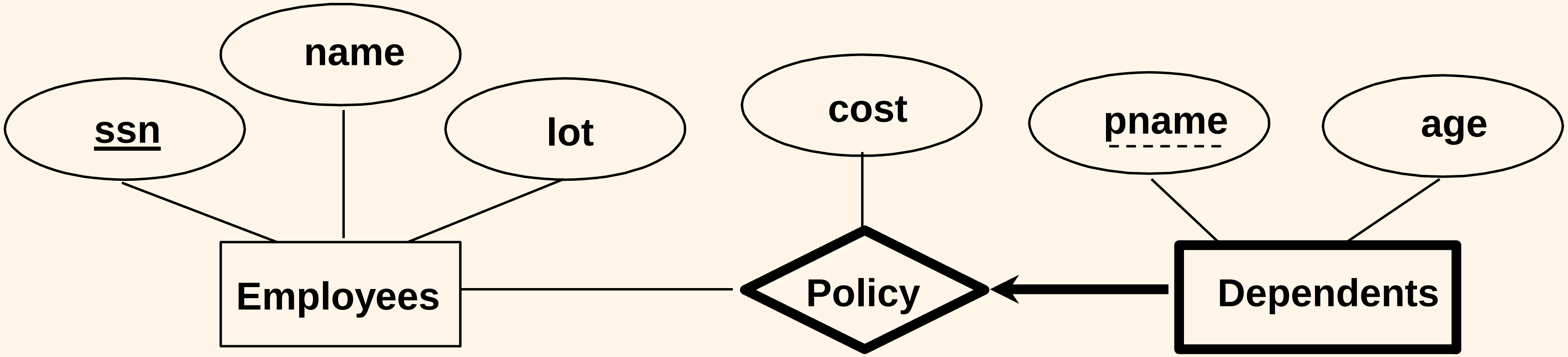
Participation Constraints

- 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 *did* value in Departments table must appear in a row of the Manages table (with a non-null *ssn* value!)

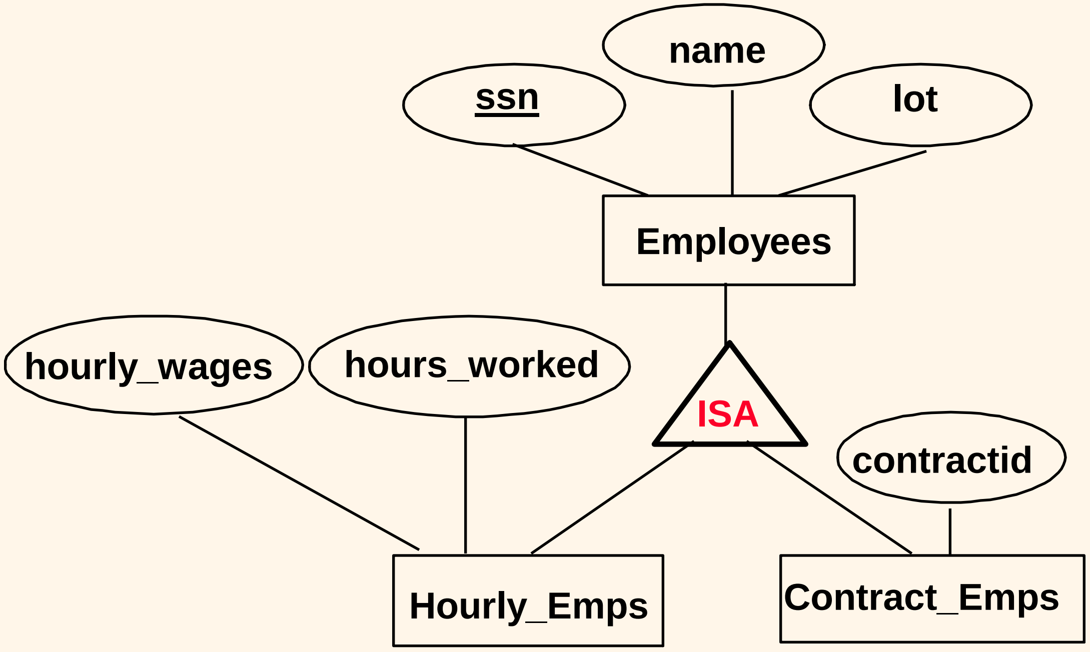


Weak Entities

- 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.



ISA (`is a') Hierarchies



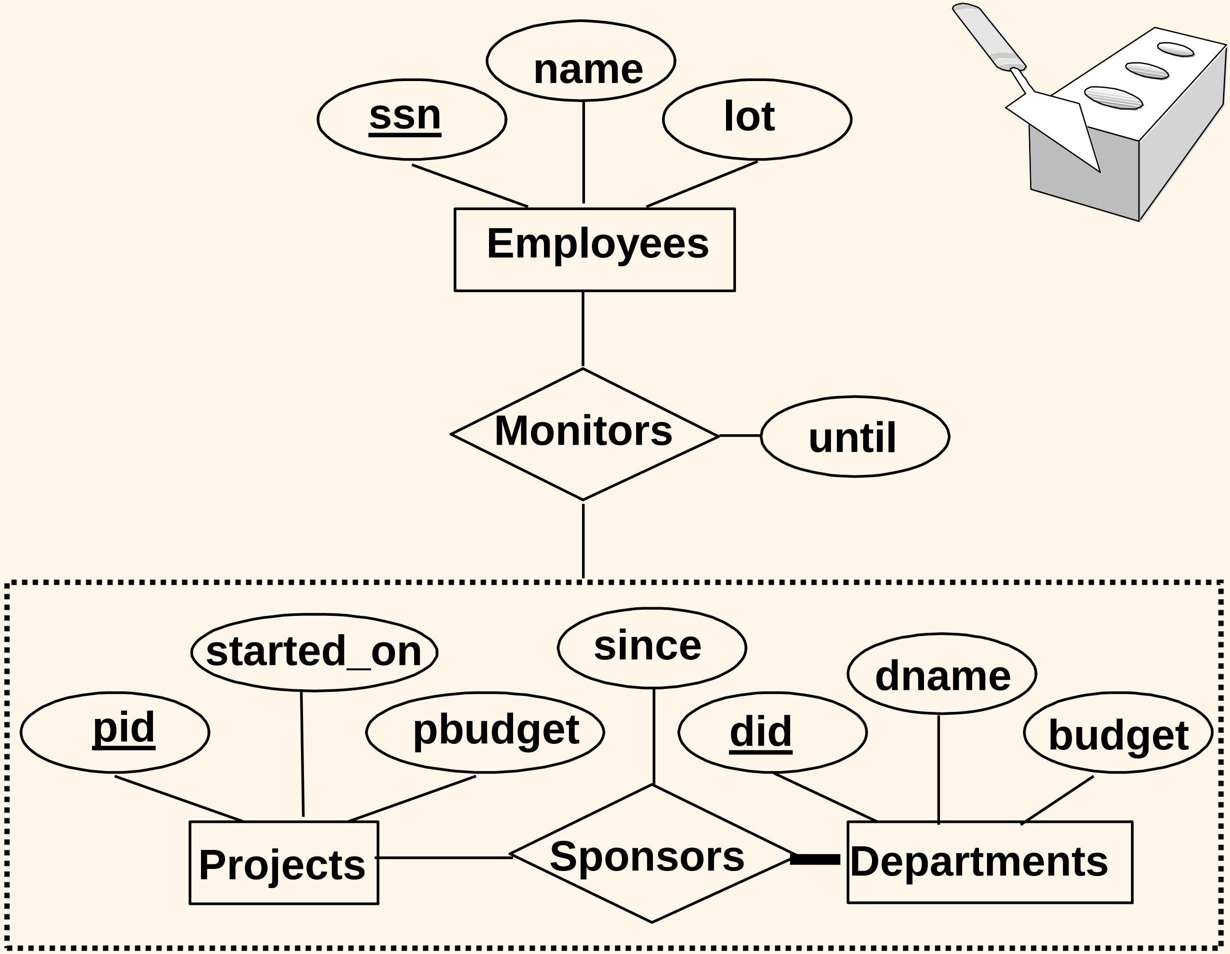
- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.
- ISA Relationship can be viewed as either specialization or generalization

ISA (`is a') Hierarchies

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

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.

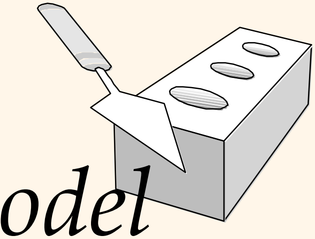


* *Aggregation vs. ternary relationship*:

v Monitors is a distinct relationship, with a descriptive attribute.

v Also, can say that each sponsorship is monitored by at most one employee.

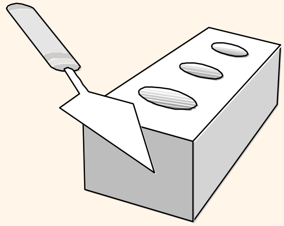
Conceptual Design Using the ER Model



- 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.

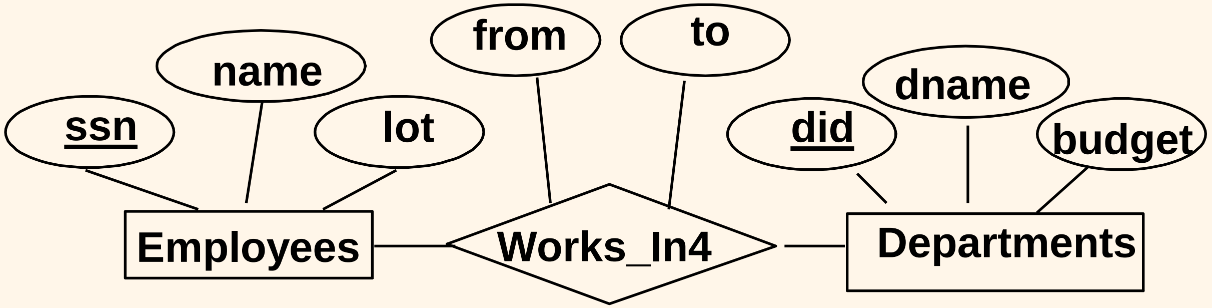
Entity vs. Attribute

- 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).

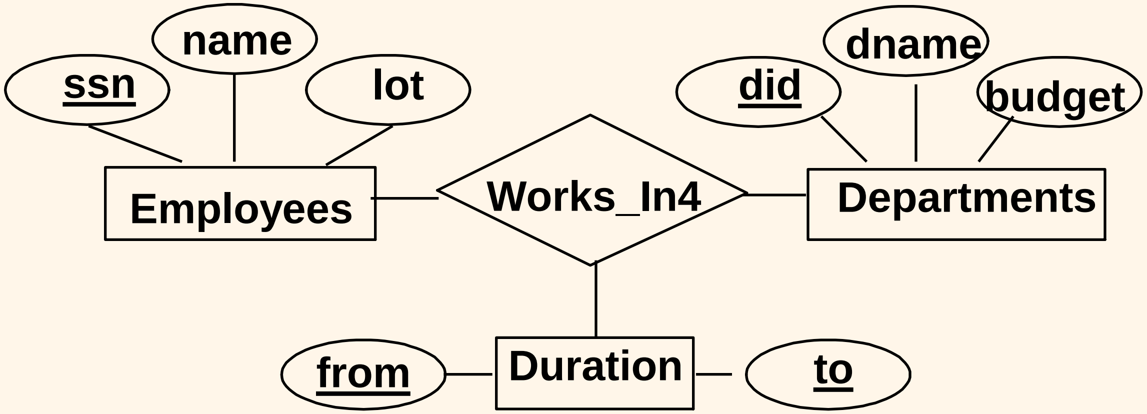


Entity vs. Attribute (Contd.)

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

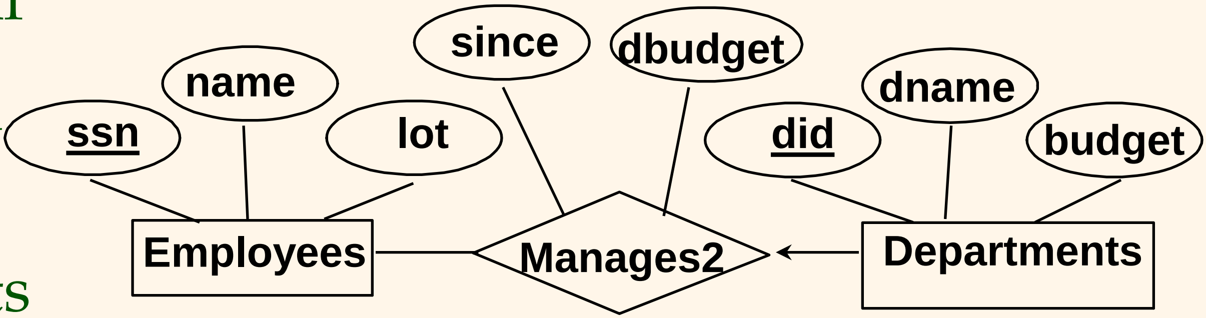


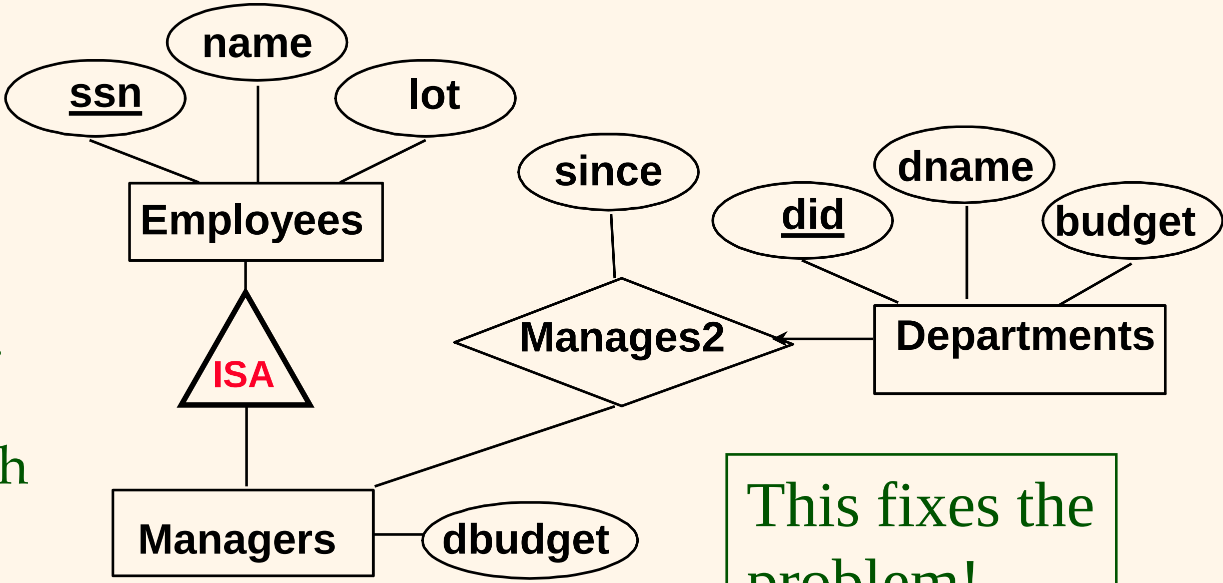
- Similar to the problem of wanting to record several addresses for an employee: We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, Duration.



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.

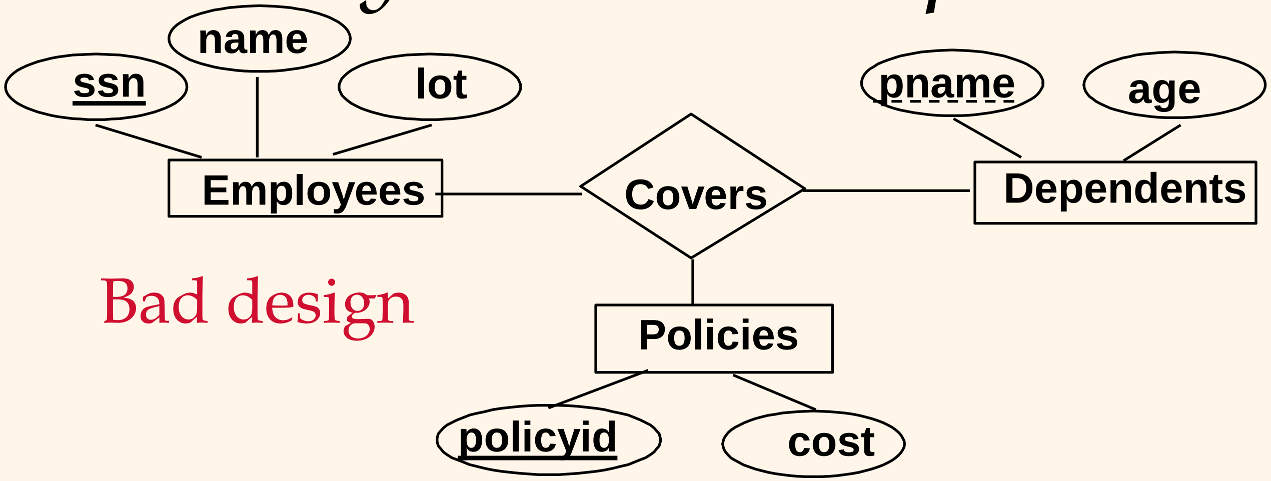




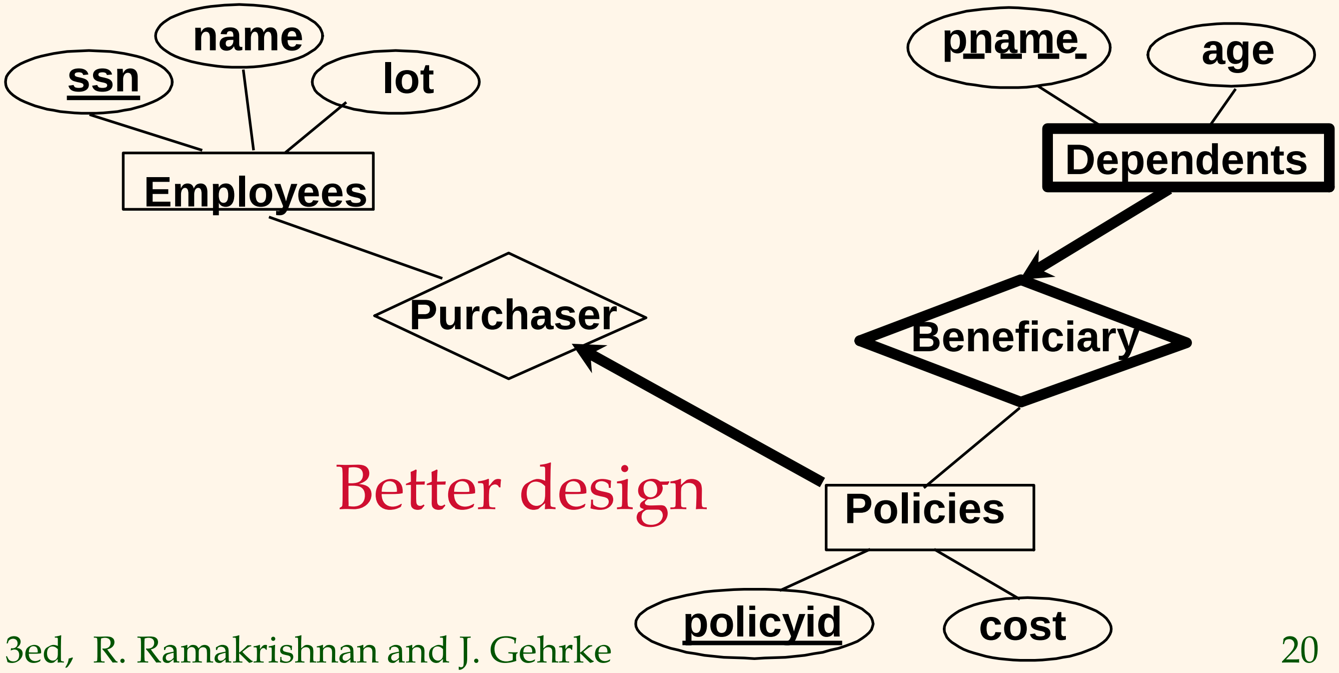
This fixes the problem!

Binary vs. Ternary Relationships

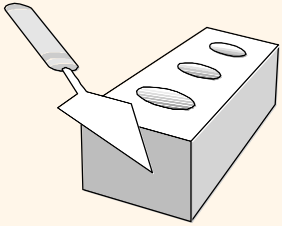
- If each policy is owned by just one employee, and each dependent is tied to the covering policy, first diagram is inaccurate.



Bad design

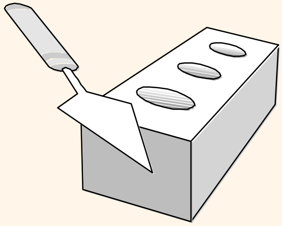


Better design



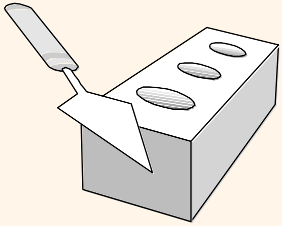
Summary of Conceptual Design

- *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.