Entity-Relationship Model

Chapter 2

Outline

- **Database design and ER diagrams**
- Entities, attributes and entity sets
- Relationships and Relationship sets
- Additional features of ER model
- Conceptual design using ER-model
- Conceptual design for large enterprises
- The unified modeling language
- Case study: Internet shop

Introduction

- Entity relationship data model
  - Describes data involved in the enterprise in terms of **objects and relationships**
  - Provides useful concepts from an informal description of what users want from database to a more detailed description

Overview of Database Design

- The process of database design can be divided into six steps
  - Requirements Analysis
  - Conceptual DB Design
  - Logical database design
  - Schema refinement
  - Physical database design
  - Application and security design

1. Requirements Analysis

- Understand what data is to be stored in DB, what applications must be built on top of it, and what operations are most frequent and subject to performance requirements.
- Involve discussions with user groups, a study of current operating environment and how it is expected to change, analysis of any available documentation on existing applications

2. Conceptual DB Design (ER Model)

- Develop high-level description of data to be stored in DB, along with constraints that are known to hold over this data (ER model)

Logical DB Design, Schema Refinement, Physical DB Design

3. Logical DB Design

– Choose a DBMS to implement our DB design, convert conceptual DB design into a DB schema in the model of the chosen DBMS, e.g., convert an ER schema into relational DB schema (chap 3)

4. Schema Refinement

– Analyze collection of relations obtained in our relational DB schema to identify potential problems (insert/delete/update anomalies), and to refine it (chap 19)

5. Physical DB Design

– Consider typical expected workloads that our DB must support and further refine DB design (chap 20)

6. Security Design

- Identify different user groups and different roles played by various users. For each role, identify the parts of DB that they be able to access and the parts of DB they should not be allowed to access (chap 21)

Conceptual 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.
  - Example: the manager of toy dept

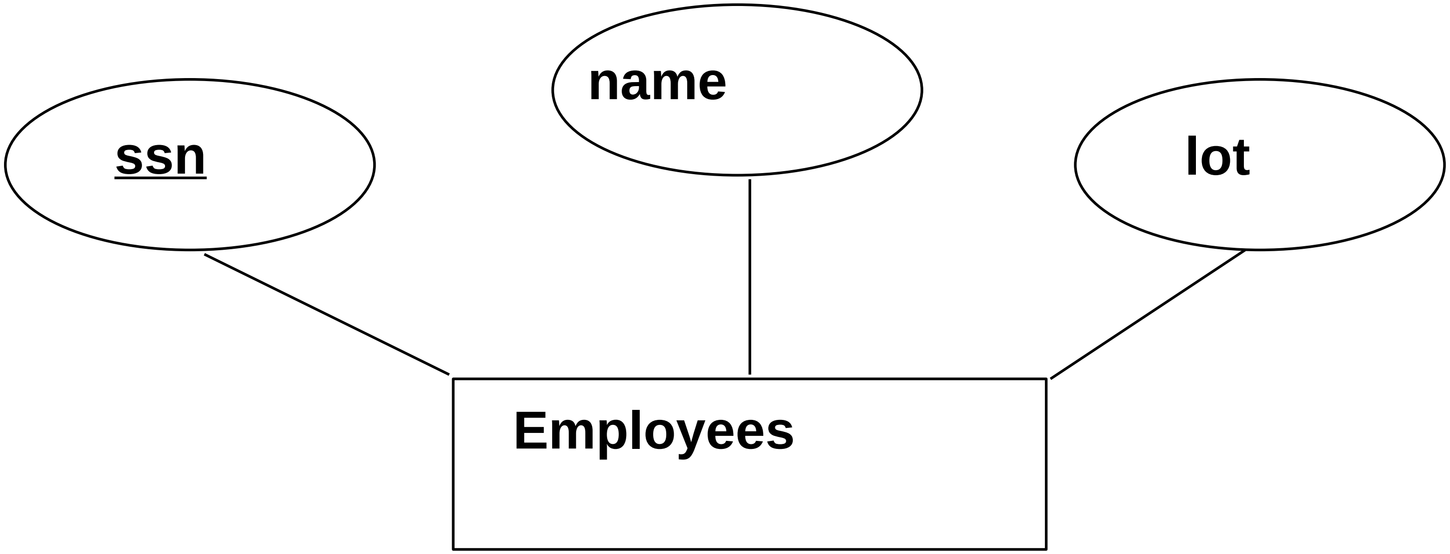
- An entity is described (in DB) using a set of ***attributes***.

  - The choice of attributes represent the level of details

ER Model Basics..

- ***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*** (**minimal set of attributes whose values uniquely identify an entity in the set*)***. There could be more one **candidate key**; if so, we designate one of them as ***primary*** **key**
  - Each attribute has a ***domain*** (set of possible values an attribute may take).

Depiction of
Entity, Entity set and attributes



name

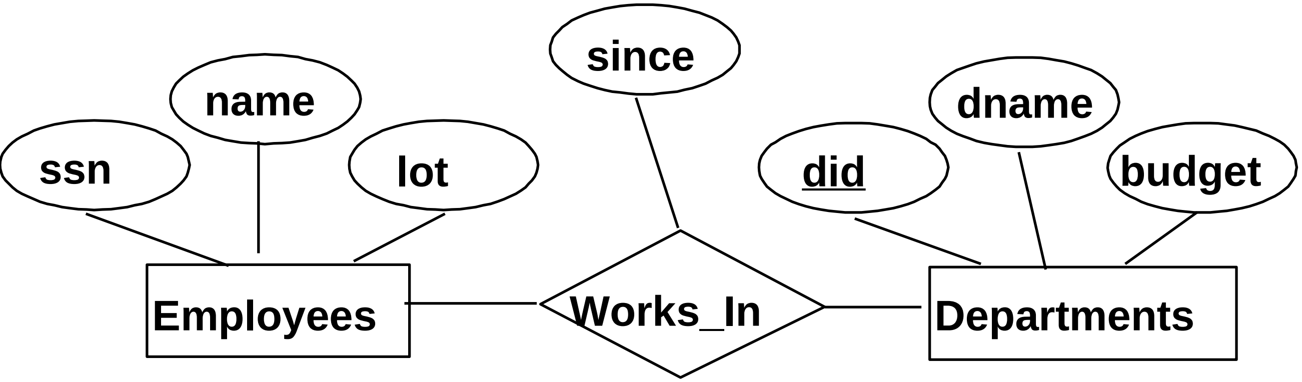
ssn

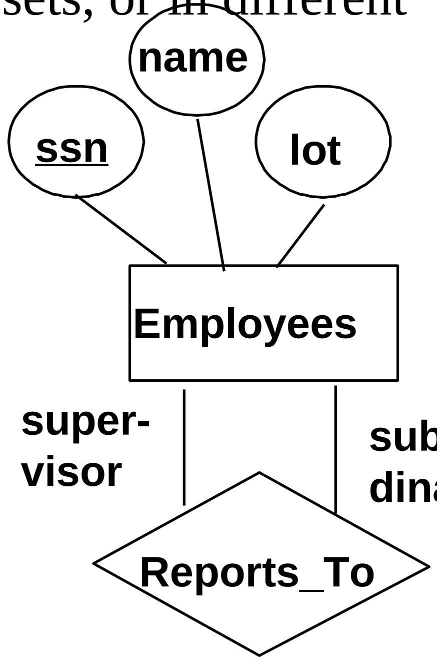
lot

Employees

ER Model Basics (Contd.)

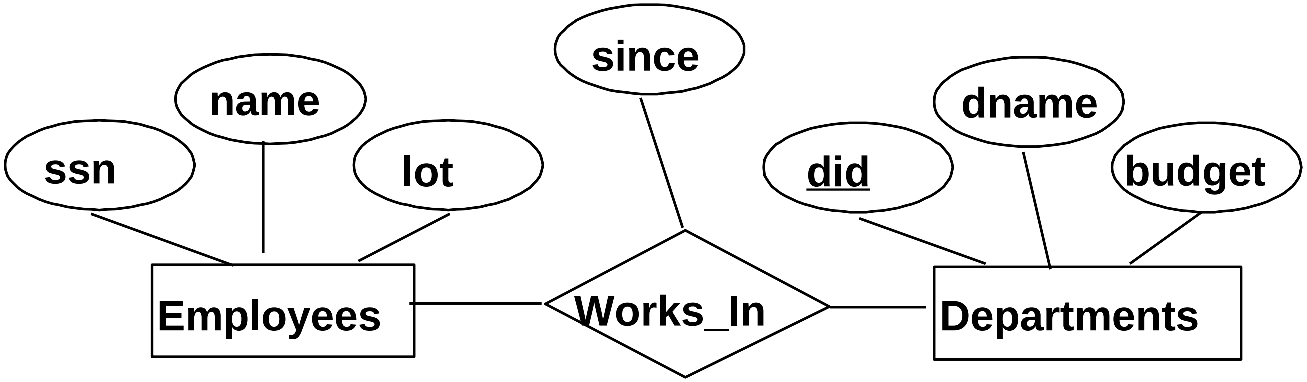
- **Relationship:** Association among two or more entities. E.g., Attishoo works in Pharmacy department.
- **Relationship Set:** Collection of similar relationships.
  - An n-ary relationship set R relates "n" entity sets E1 ... En; each relationship in R involves entities $e_1 \in$ E1, ..., $e_n \in$ En
  - Same entity set could participate in different relationship sets, or in different "roles" in same set.

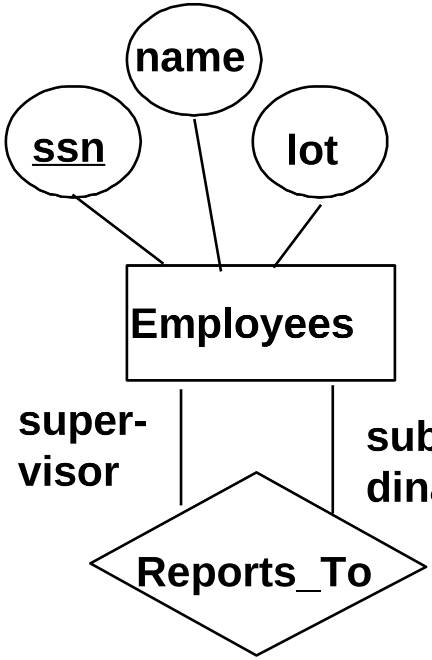




ER Model Basics (Contd.)

- ***A relationship can also have descriptive attributes*****:** Descriptive attributes are used to record information about the relationship, e.g., **since** attribute associated with Works_In relationship
- ***Instance of a relationship*****:** a set of relationships.

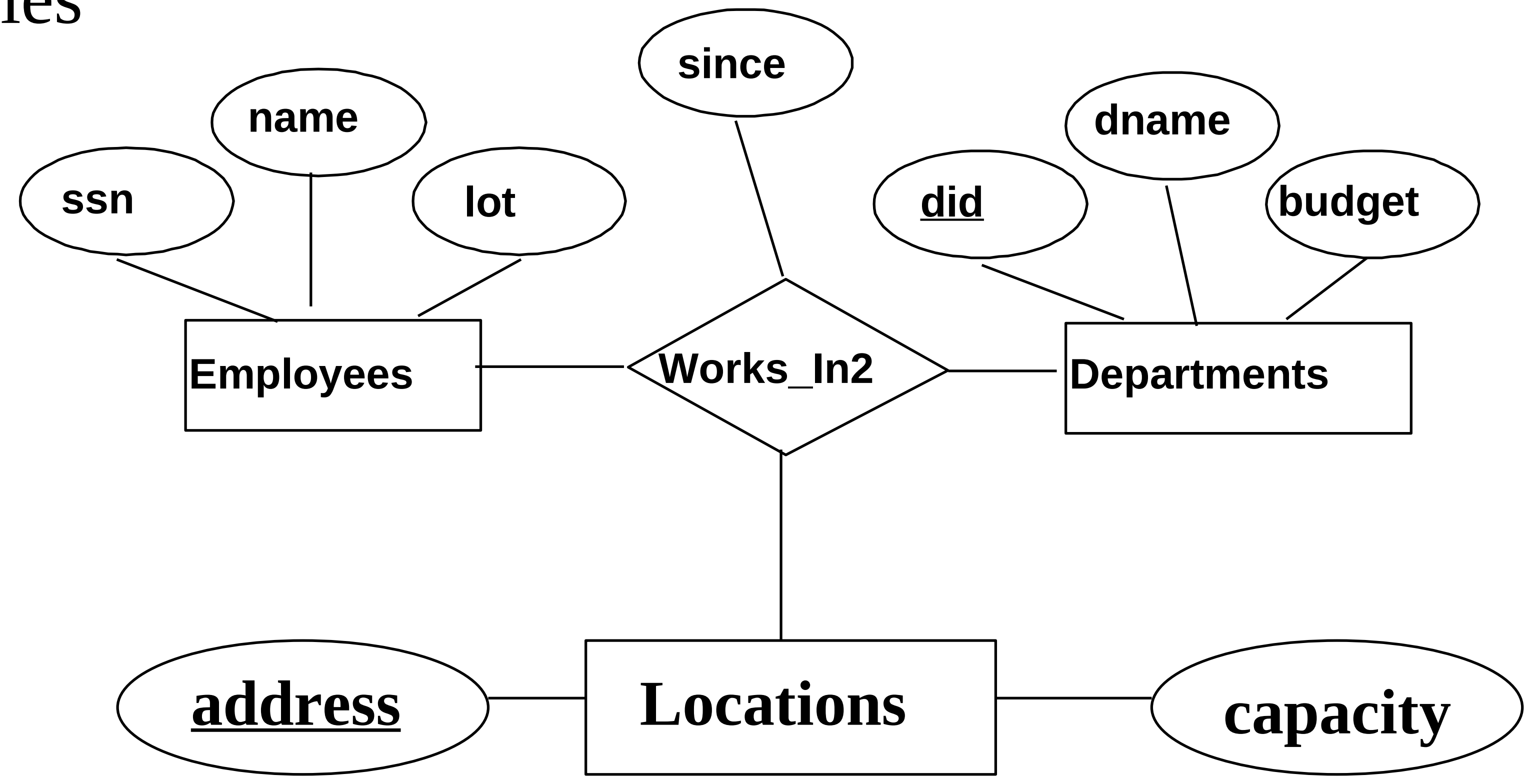




- Employee is allowed to work in many depts
- Each dept is allowed to have several employees

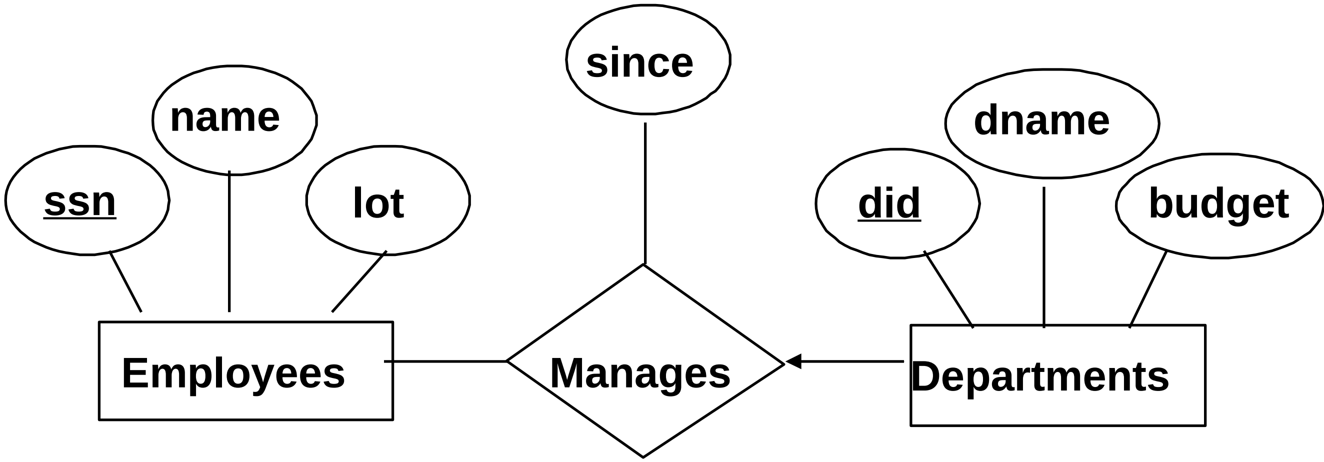
ER Model Basics (Cont.)

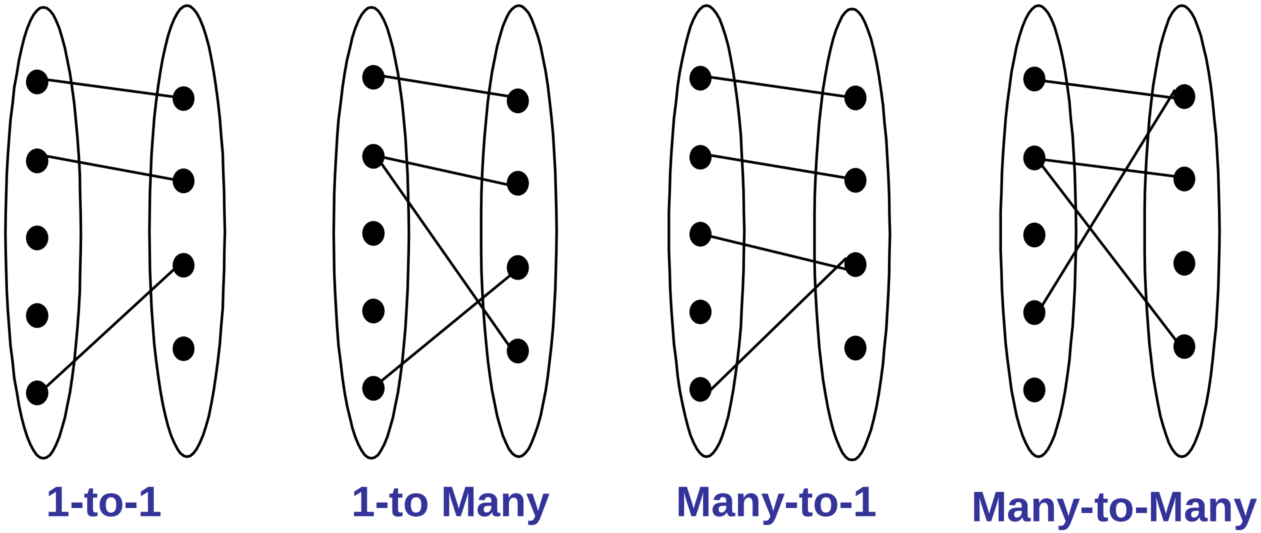
- **Ternary relationship:** a relationship involving three entities



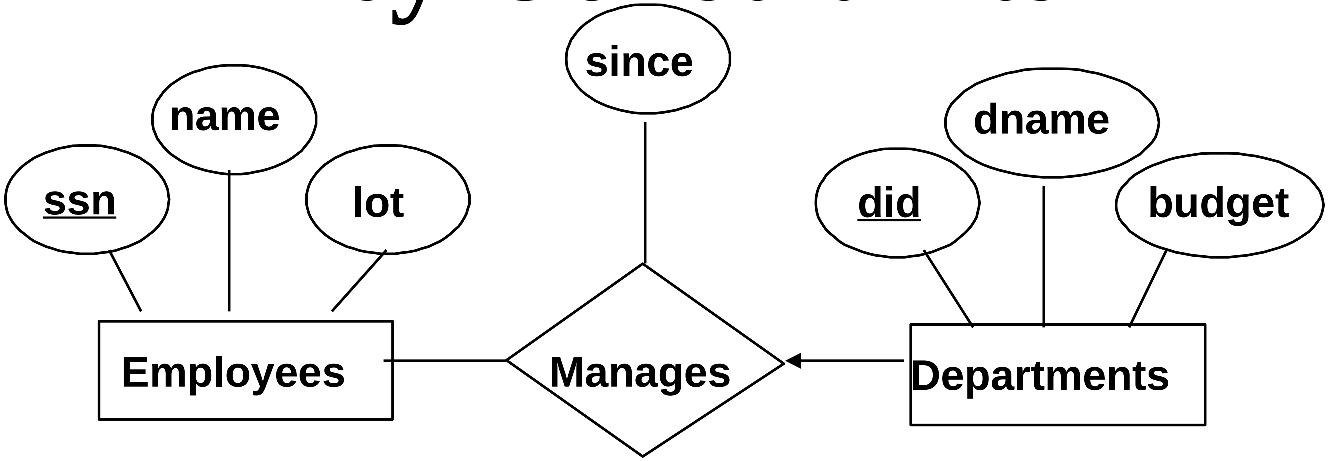
Key Constraints

- 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.



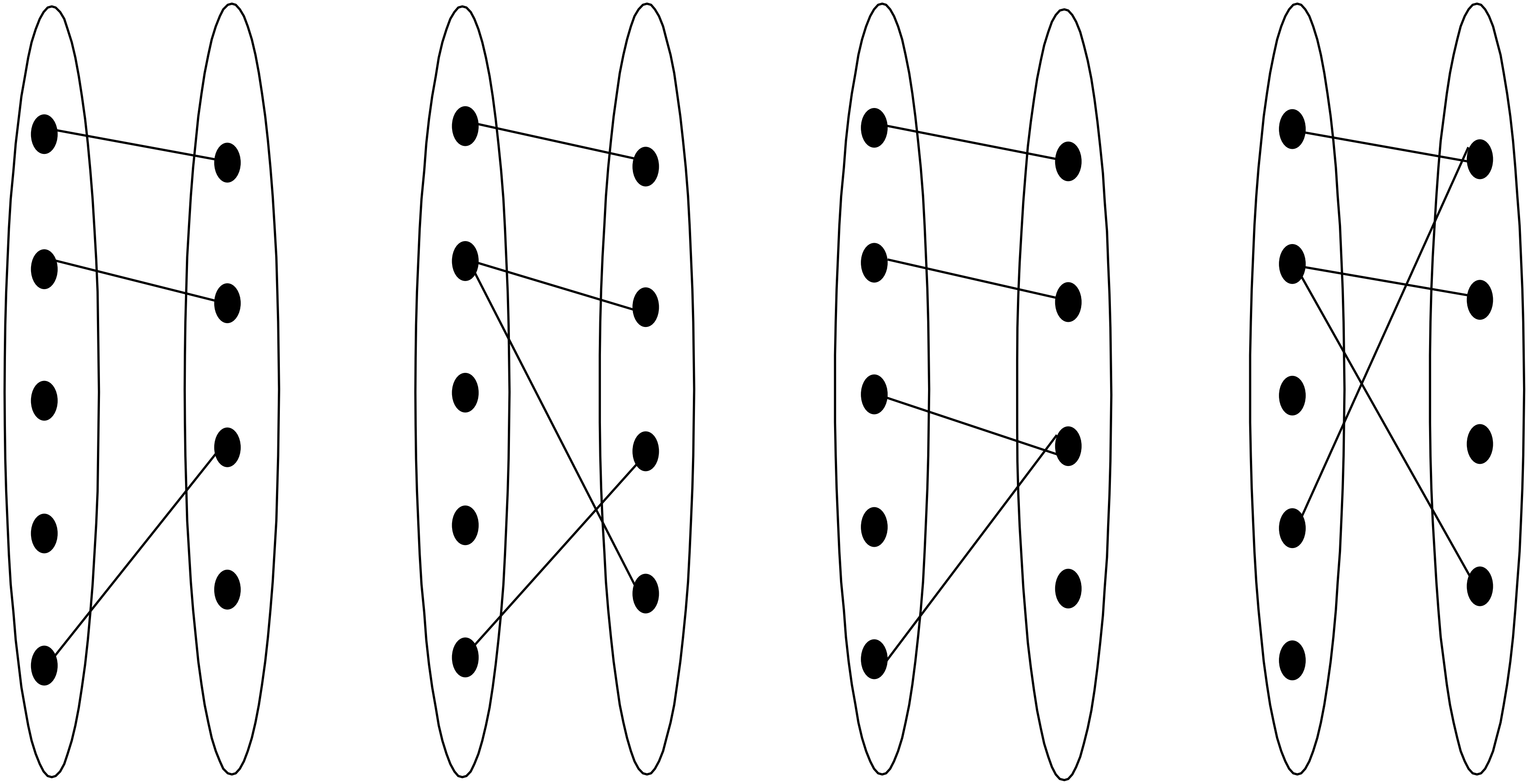


Key Constraints



- In contrast, each dept has at most one manager, according to the *key constraint* on Manages.
- Each department appears at most one manages relationship
  - An arrow from Departments to Manages
- Note: two departments can be managed by one person !
  - Many departments to one manager
  - Also, single employ can manage more than one dept

Key Constraints



1-to-1

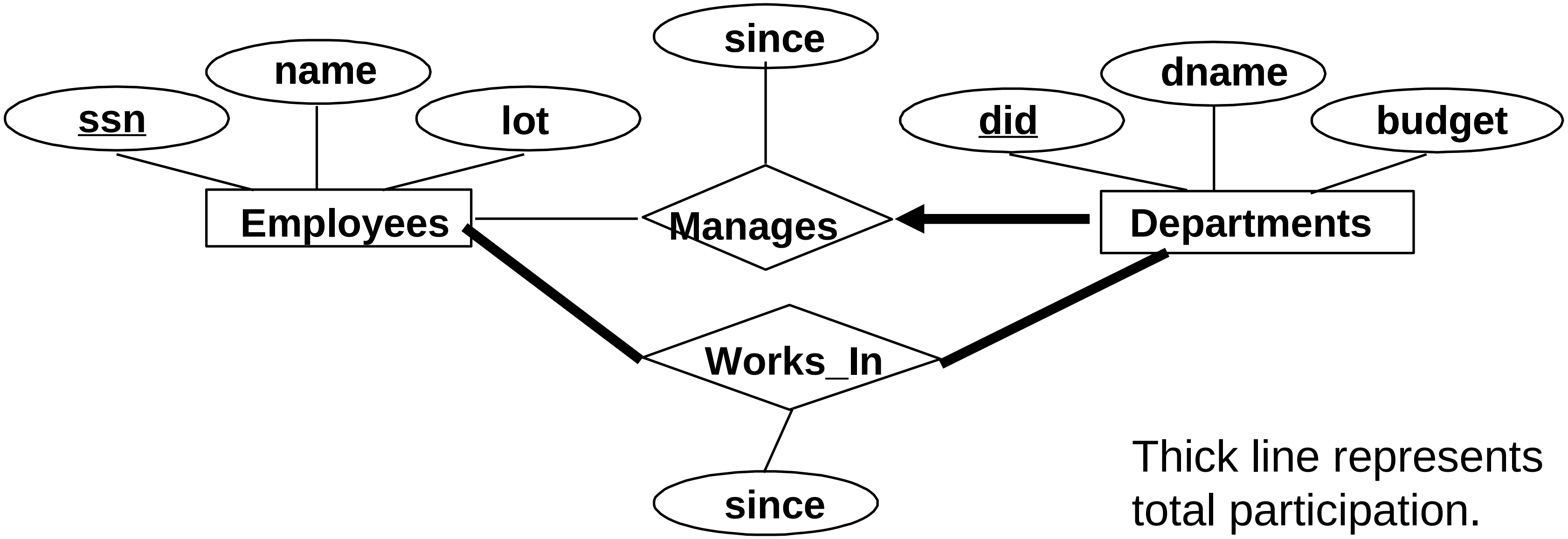
1-to Many

Many-to-1

Many-to-Many

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!)



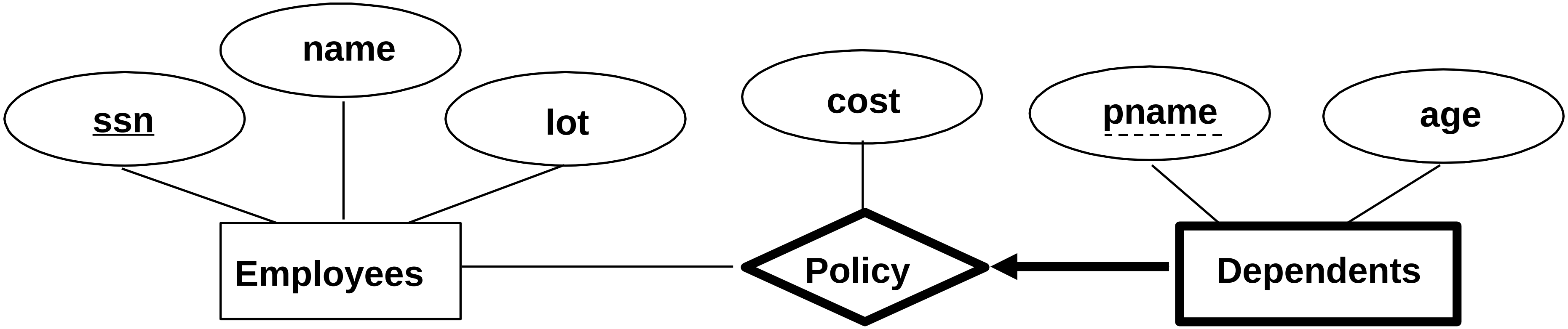
Thick line represents total participation.

Weak Entities

- So far, the attributes associated with entity have a key.
- The assumption is not valid!
- Example: Employee purchases insurance policies for dependents.
- We wish to record information who is covered by which policy ?
- If the employee quits, we want to terminate the policy and all the information including dependent information is deleted!

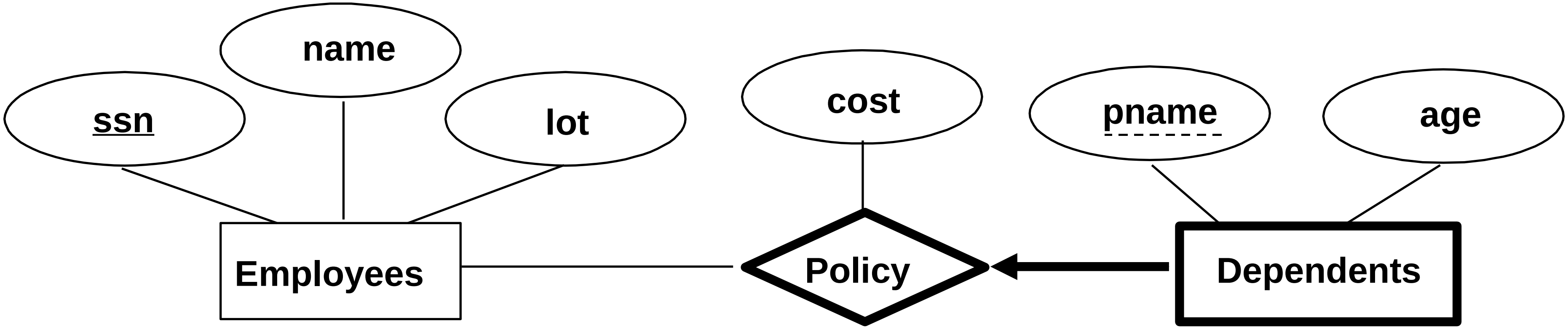
Weak Entities

- 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.



Weak Entities: Notations

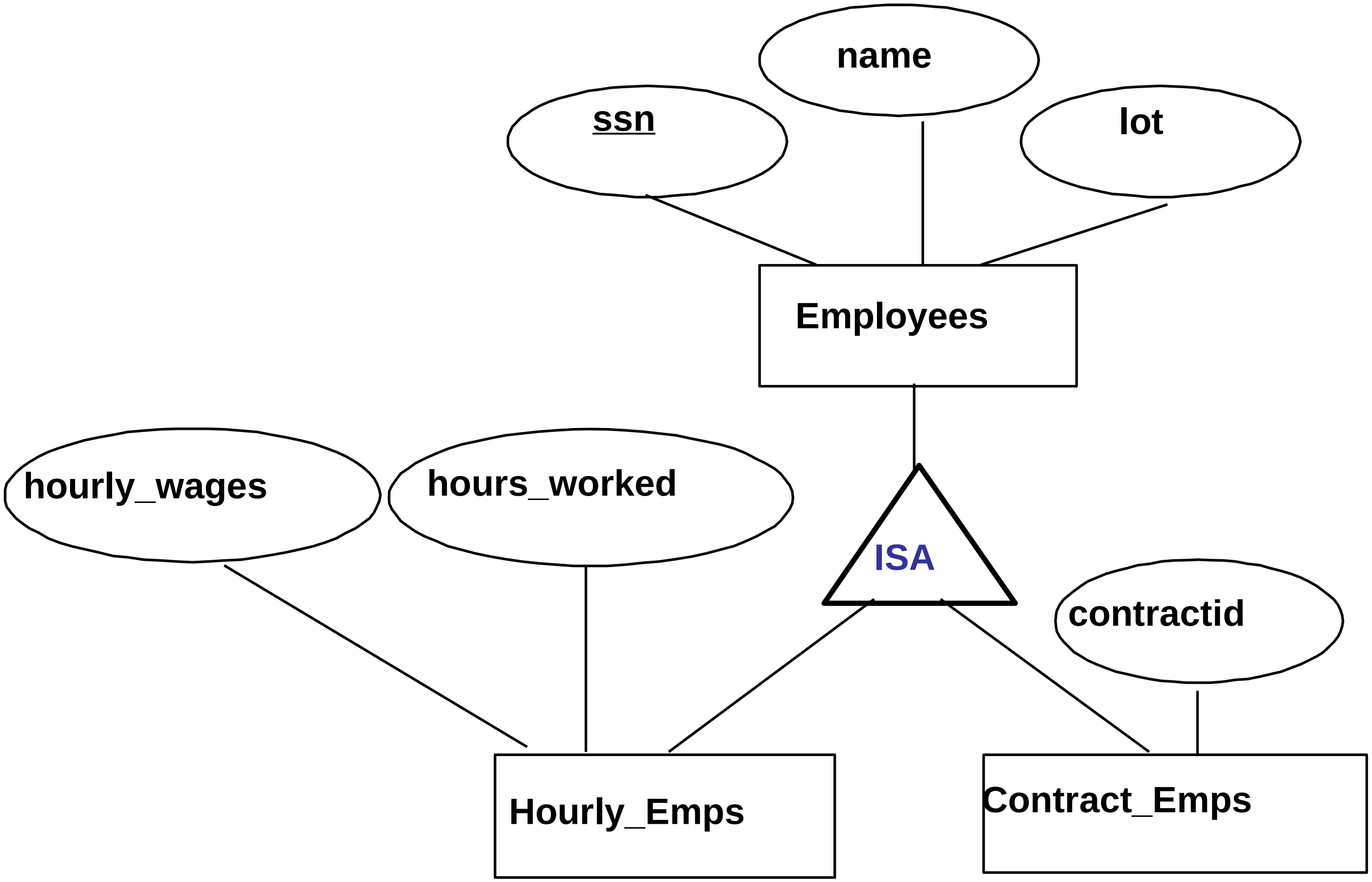
- Both entity and relationship are drawn with thick line to indicate that dependent is a weak entity associated with policy.
- To indicate “pname” is a partial key we underline with a broken line.
  - It means there will be two dependents with same name value.



ISA hierarchy Generalization/Specialization

- Sometimes it is natural to divide the entity set into sub classes
  - Divide the employ set into **hourly-emps set** and **contract-empls set.**
  - The attributes of “hourly-emps set” is equal to the attributes of “employ set” plus “hourly-emps”
  - We say the attributes of entity set Employees are **inherited** by the entity set Hoursly-emps
  - Hourly-emps **ISA** Employees.

ISA hierarchy
Generalization/Specialization



ISA (`is a’) Hierarchies

- A IS a hierarchy can viewed in one of two ways.
- **Specialization:**
  - It is a process of identifying the subsets (subclass) of entity set (superclass)
  - Supercalss is defined first and subclasses are defined next.
- **Generalization**
  - It consists of identifying some common characteristics of a collection of entity sets and creating entity set containing entities possessing these common characteristics
  - Subclasses are defined first, the superclass is defined next.

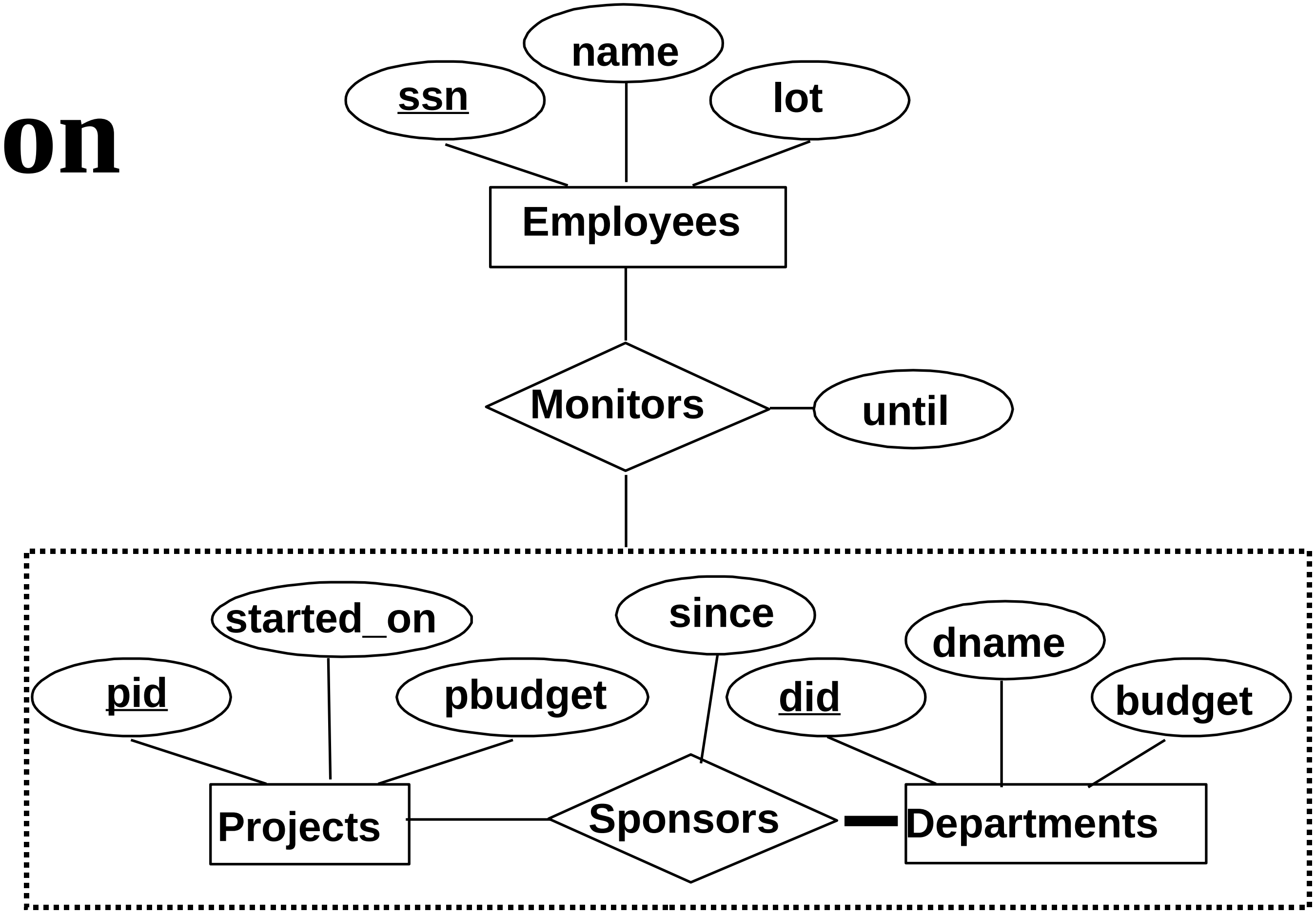
Constraints related to ISA (`is a’) Hierarchies

- **Overlap constraints:**
  - Determine whether two subclasses allowed to contain the same entity.
  - Can Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
    - suppose there is another Senior_Emps subclass, and an employee can be both a Contract_Emps entity and a Senior_Emps entity; we denote this by writing ‘Contract_Emps OVERLAPS Senior_Emps’
- **Covering constraints:**
  - Determine whether the entities in subclasses collectively include all entities in the superclass.
  - Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
  - If every Motor_Vehicles entity have to be either a Motorboats entity or a Cars entity, then we write ‘Motorboats AND Cars COVER Motor_Vehicles’

Aggregation

- So far, relationship is an association between two entity sets.
- Sometimes, we have to model the relationship between a collection of entities and relationships.
- *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.
- Example:
  - Consider entity set called “projects”, and each project is sponsored by one or more departments. The sponsor relationship captures this information.
  - Support the department assigns employees to monitor the projects.
  - The Monitor is a relationship that associates a Sponsors relationship with Employees entity.

Aggregation



☛ *Aggregation vs. ternary relationship:* Aggregation monitors is a distinct relationship, with a descriptive attribute. Also, can say that each sponsorship is monitored by at most one employee.

Conceptual Design Using the ER Model

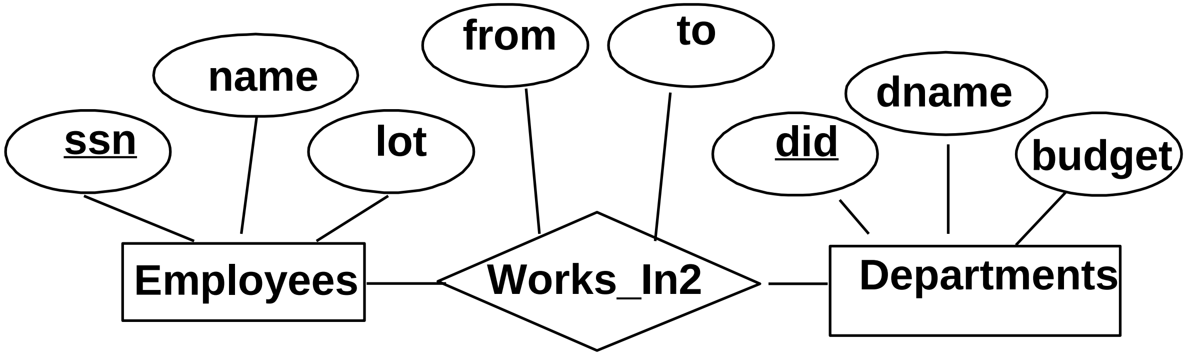
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But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

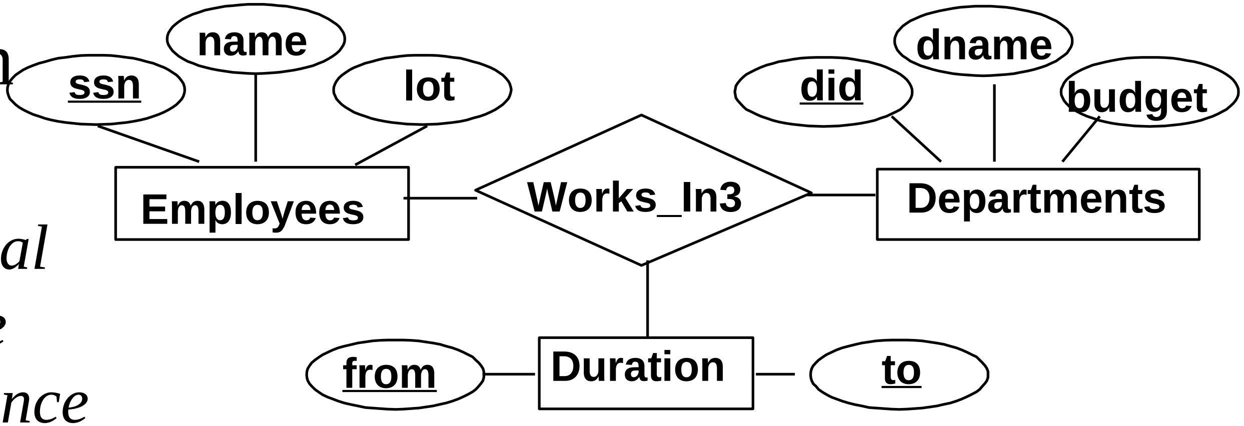
Entity vs. Attribute

- 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_In2 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.*

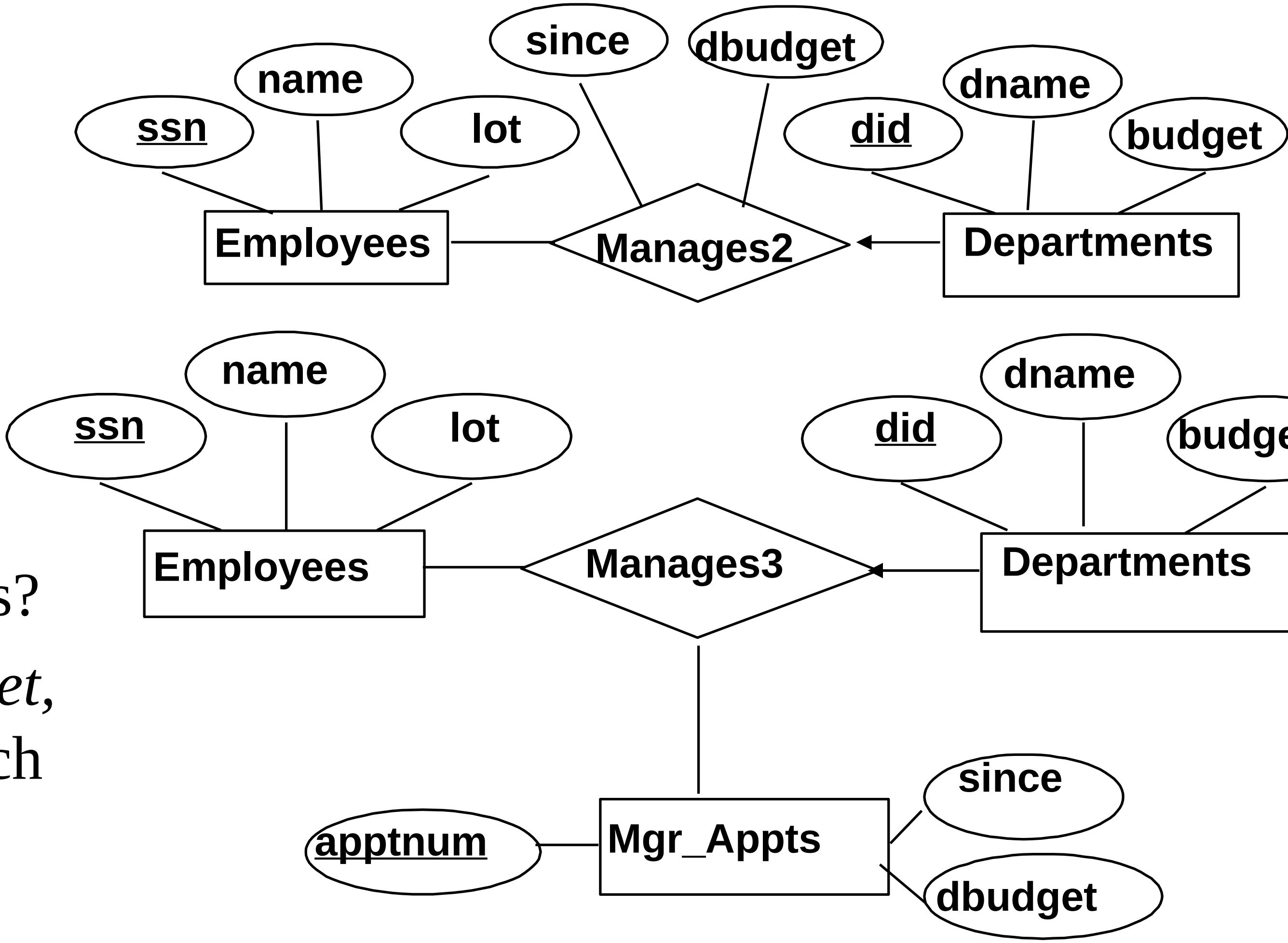




Entity vs. Relationship

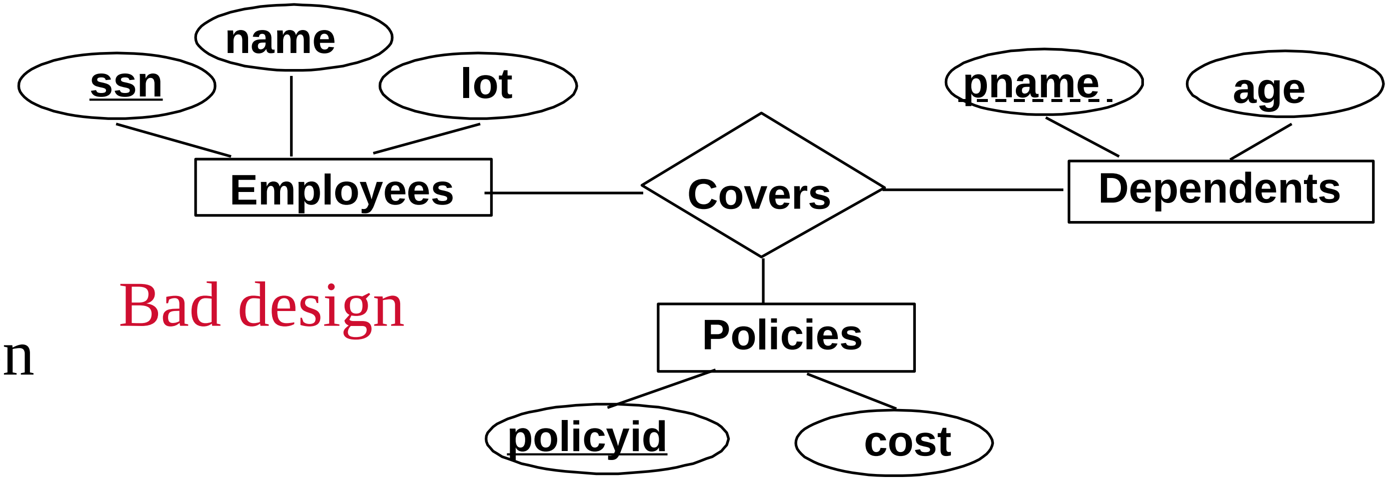
- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy of *dbudget*, which is stored for each dept managed by the manager.

Misleading: suggests *dbudget* tied to managed dept.

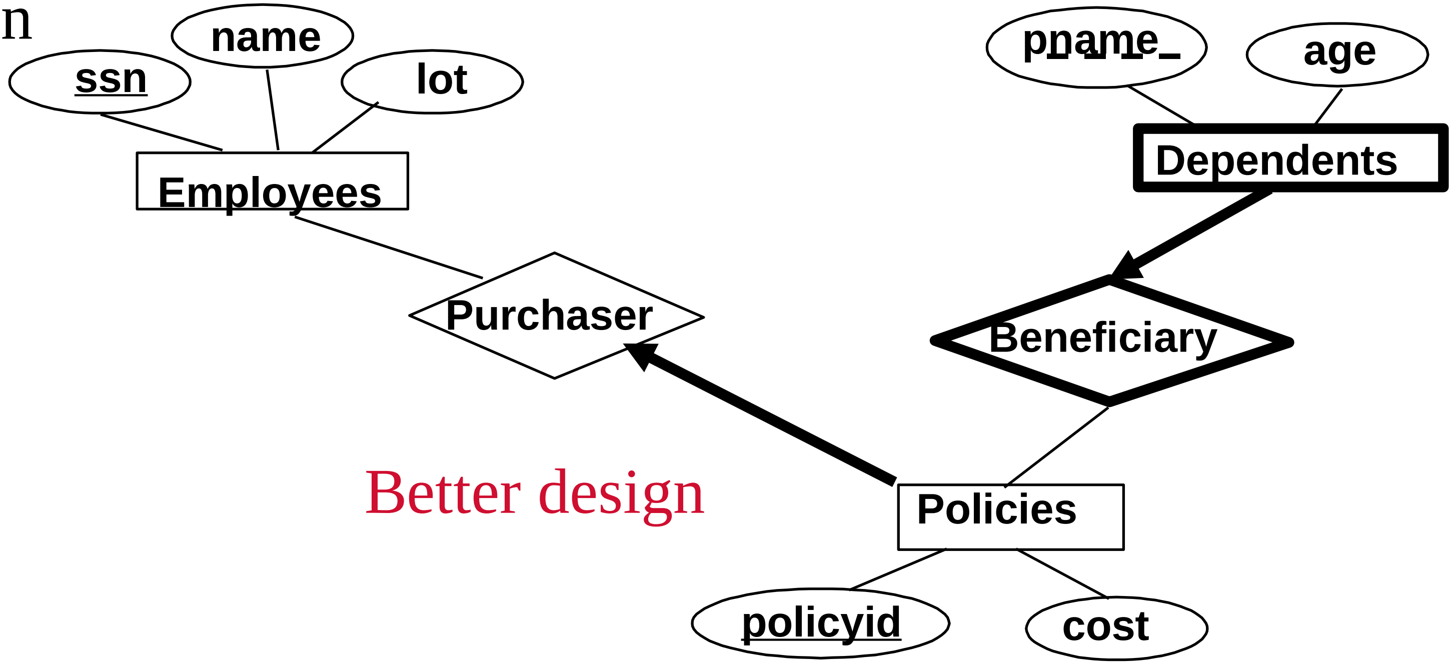


Binary vs. Ternary Relationships

- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!
- What are the additional constraints in the 2nd diagram?



Bad design



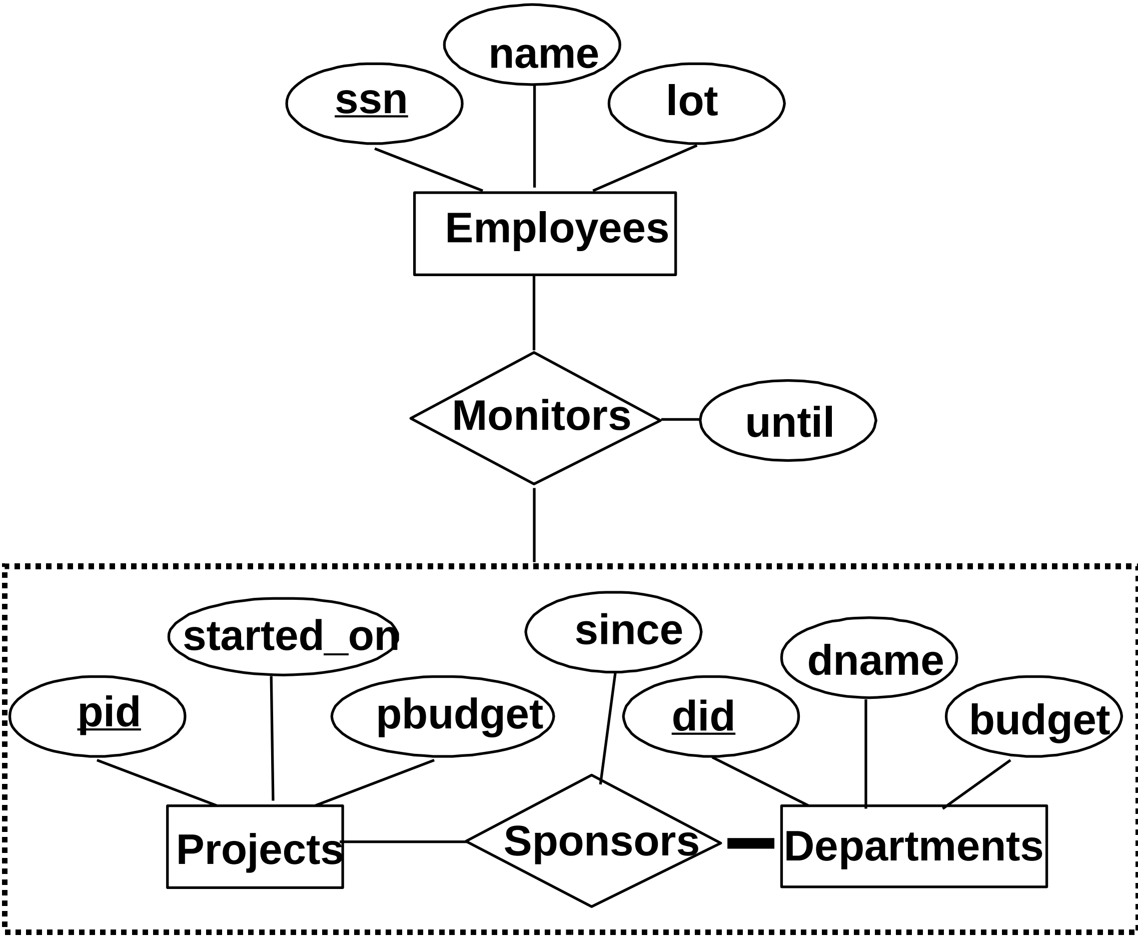
Better design

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*?

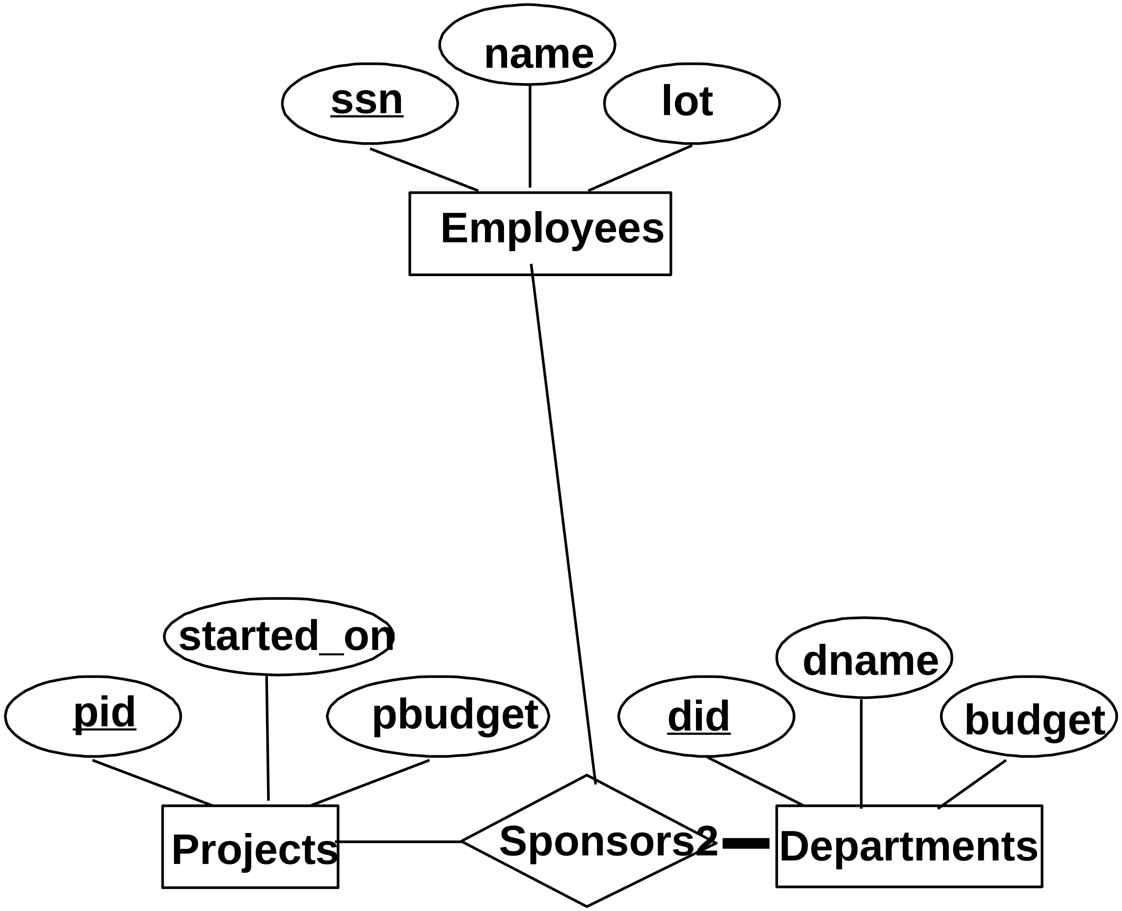
Aggregation vs. Ternary Relationships

- The choice mainly determined by the existence of relationship that relates a relationship set to an entity set.
- The choice is guided by certain integrity constraints that we want to express.
- Consider the figure
  - A project can be sponsored by multiple depts.
  - Department can sponsor one or more projects
  - Each sponsorship is monitored by one or more employees.
- The constraint that sponsorship monitored by at most one employee is represented by aggregated relationship



Aggregation vs. Ternary Relationships

- If we do not need to record the until attribute of Monitors, we can go for ternary relationship.
- However, if we want to implement the constraint that sponsorship monitored by at most one employee it is better to go for aggregated relationship



Conceptual Design for Large Enterprises

- Design requires the efforts of multiple designers
- ER model can be diagrammatically represented and easily understood
- Many people can provide input
- Important aspect
  - Design takes into account all the user requirements and is consistent.
- Schema integration
  - Develop separate schemas and integrate

The Unified Modeling Language (UML)

- Covers broader spectrum of software design process than the ER-model
  - **Business modeling**
    - Describe the business processes
  - **System modeling**
    - Identify system requirements. One part of requirements is the database requirements
  - **Conceptual database modeling**
    - Creation of ER design for the database. UML provides many constructs that parallel the ER constructs.
  - **Physical database modeling**
    - UML provides pictorial representation of physical DB design choices
  - **Hardware system modeling**
    - UML diagrams can be used to describe hardware configuration

The Unified Modeling Language (UML)

- **Activity diagrams**
  - Flow of actions of business process
- **State chart diagrams**
  - Describe dynamic intercations between system objects
- **Class diagrams**
  - Similar to ER diagrams
- **Database diagrams**
  - How classes are represented in the database
- **Component diagrams**
  - Storage aspects of database
- **Deployment diagrams**
  - Hardware aspects of the system

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.