The Entity-Relationship Model

Chapter 2

Overview of Database Design

- Iterative process

Requirement analysis

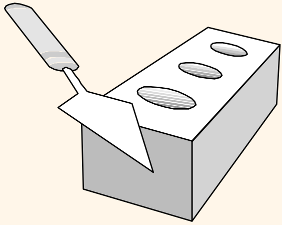
Conceptual DB design

Logical DB design

Schema refinement

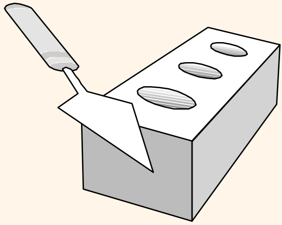
Physical DB design

App & DB security



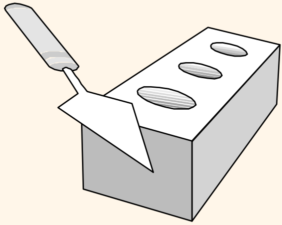
Overview of Database Design

- 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

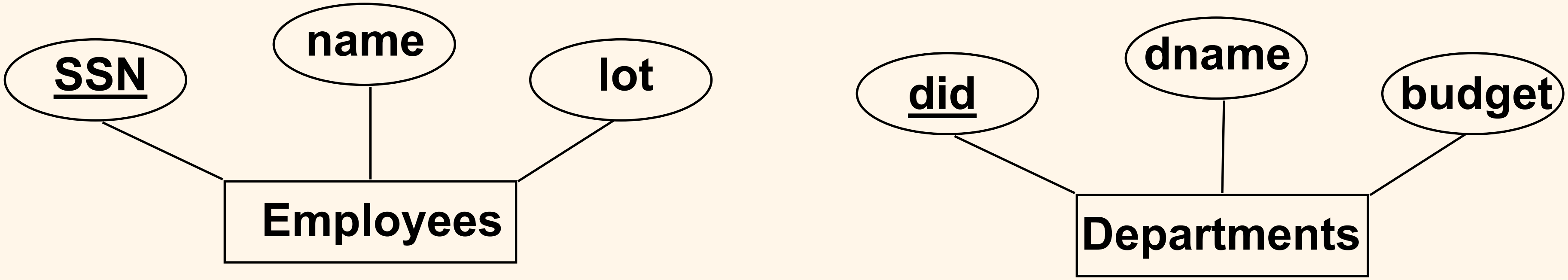
- *Conceptual design*: *(ER Model is used at this stage.)*
  - What are the *entities* and *relationships* in the enterprise?
  - What information about these entities and relationships should we store in the database?
  - What are the *integrity constraints* or *business rules* that hold?
  - A database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



ER Model Basics

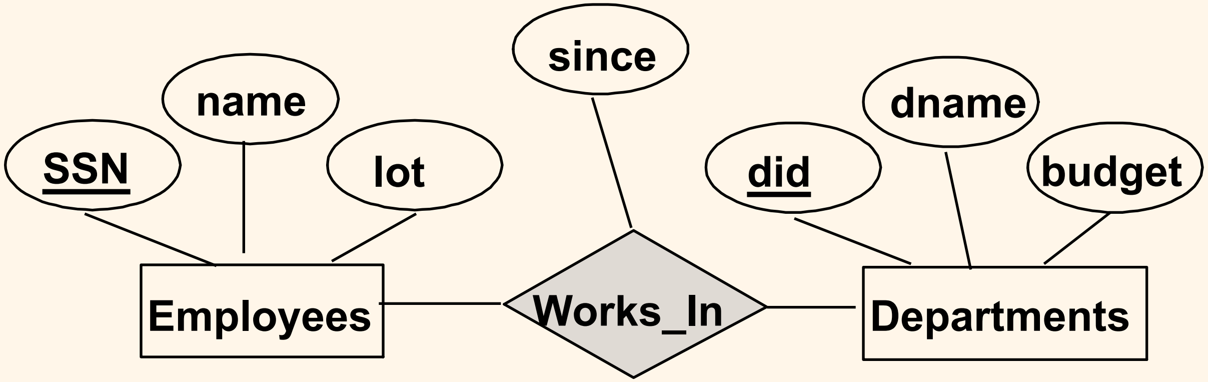
- *Entity:* Real-world object distinguishable from other objects. An entity is described (in DB) using a set of *attributes*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)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

ER Model Basics (Entity)



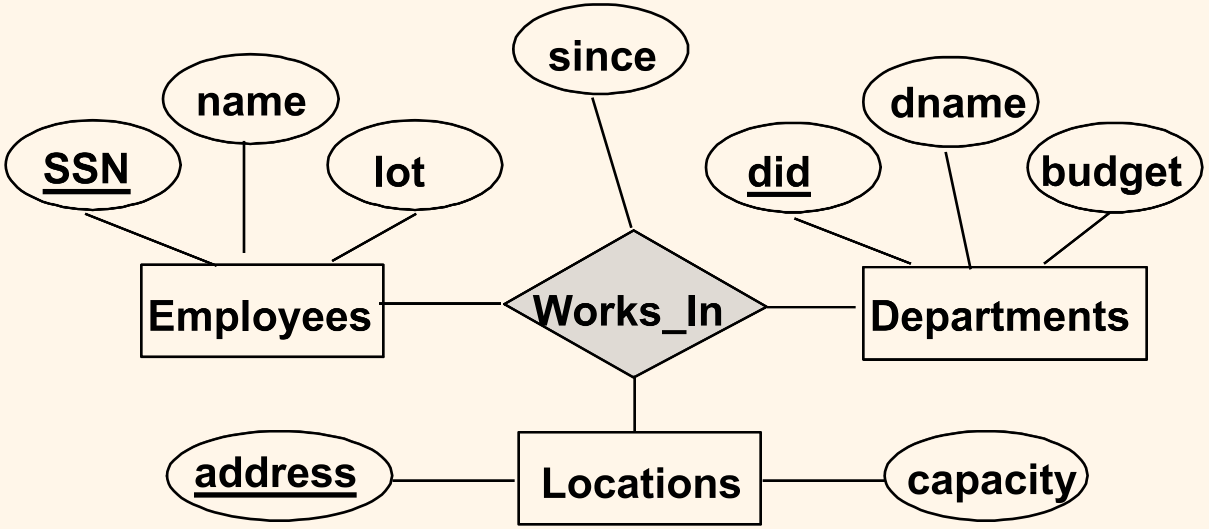
- Based on the discussion with our customers, we find that:
  - Each employee has there attributes: SSN, name, and parking lot
  - Each department has three attributes: number, name, and budget
- Employees can be uniquely identified by their SSN (SSN), departments can be identified by their numbers (did).
- 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

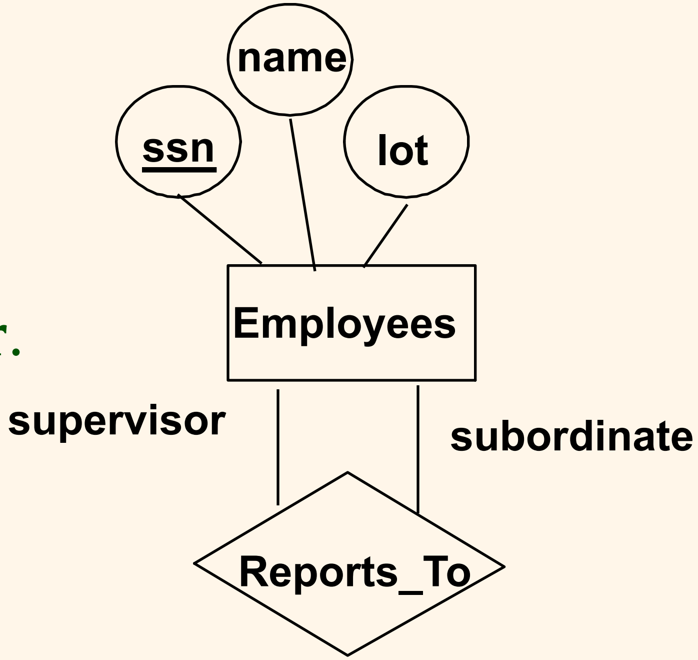
N-ary Relationship



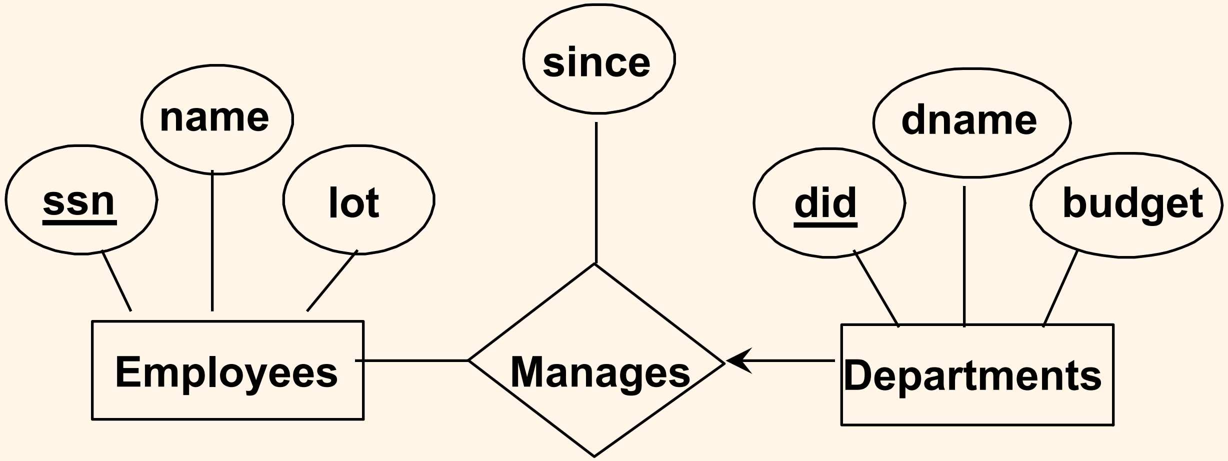
- An n-ary relationship set R relates n entity sets E1 … En; each relationship in R involves entities e1   E1, …, En

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.

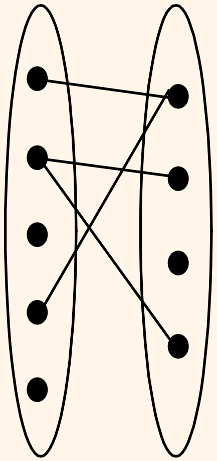


Key Constraints



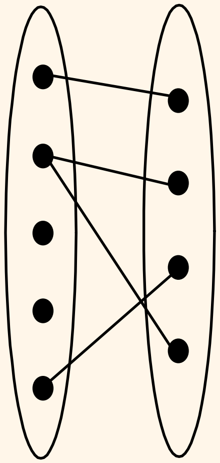
- Each dept has at most one manager, according to the *key constraint* on Manages.

Key Constraints (Cont.)



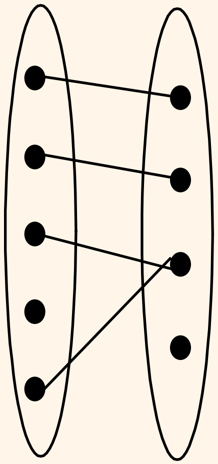
Many-to-Many

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



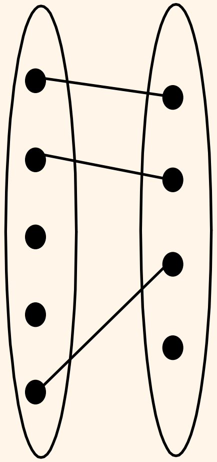
1-to Many

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



Many-to-1

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

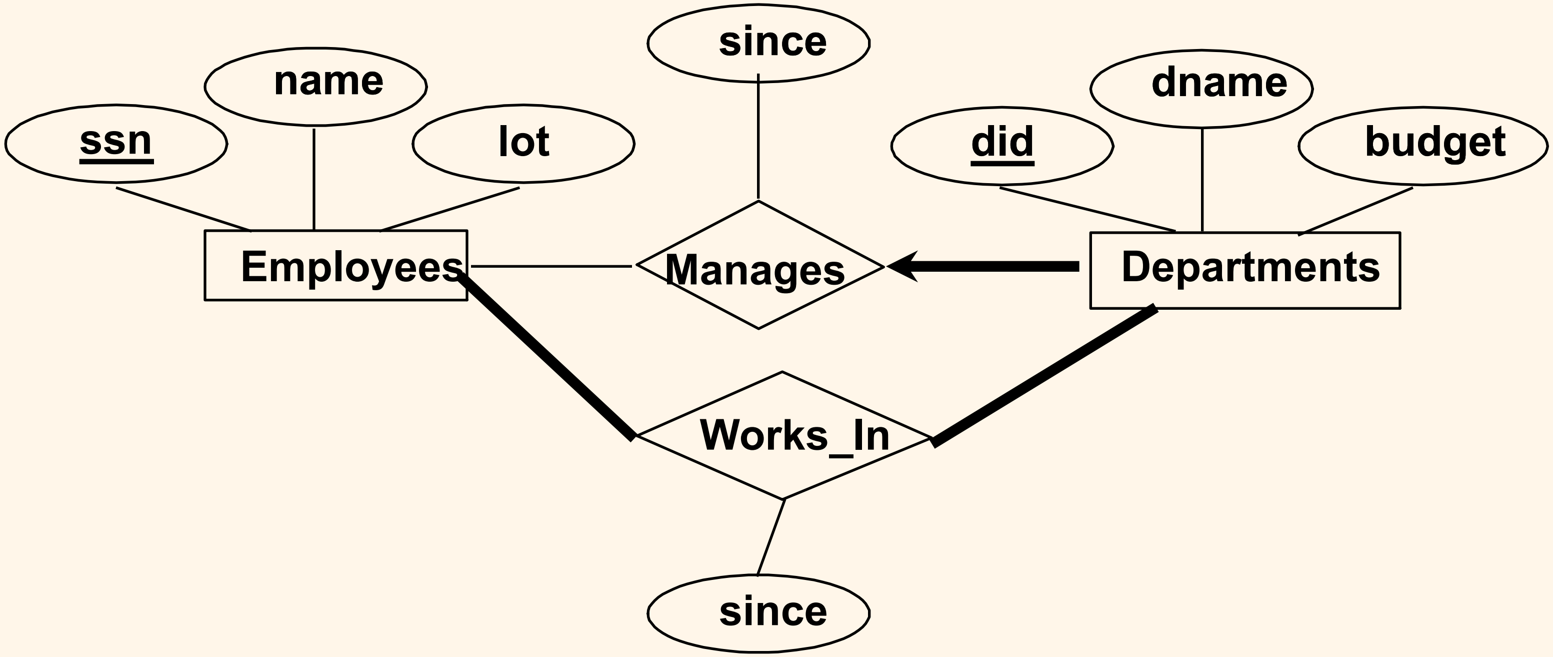


1-to-1

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

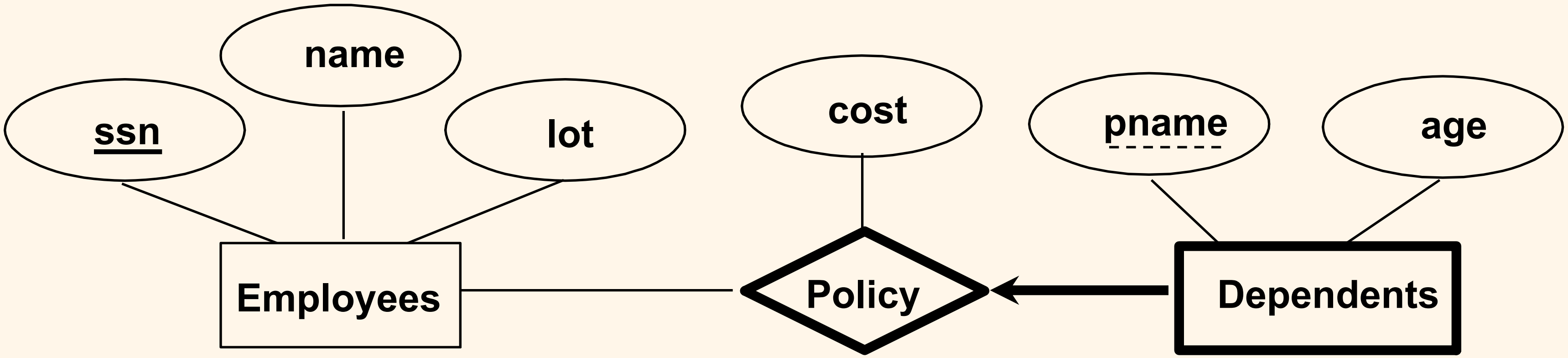
Participation Constraints

- 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!)

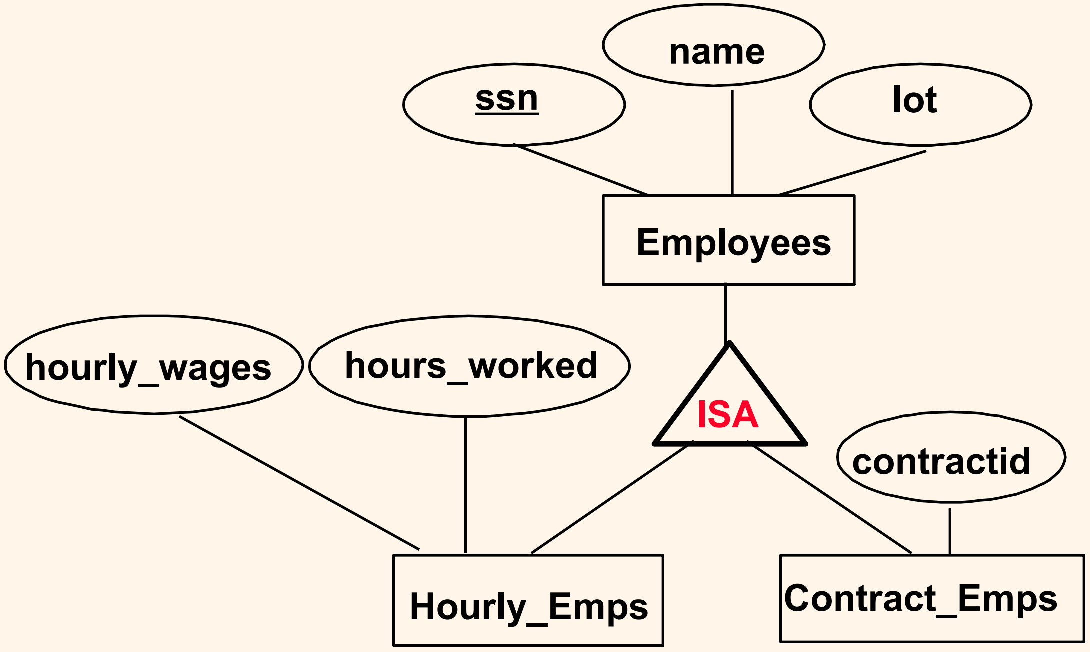


Weak Entities

- 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.

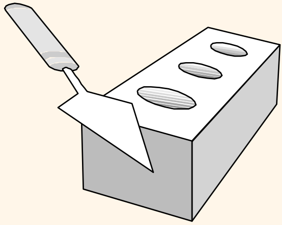


ISA (`is a') Hierarchies



- 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.