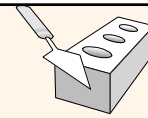


Introduction to Database Design

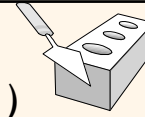
Chapter 2



Overview of Database Design

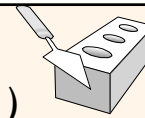
- ❖ *Requirements Analysis*
 - The very first step in designing a database application.
 - We focus on the design of the database.
- ❖ *Conceptual design:* (*ER Model is used at this stage.*)
 - What are the *entities* and *relationships* in the enterprise?
 - What information about these entities and relationships should we store in the database?
 - What are the *integrity constraints* or *business rules* that hold?

Overview of Database Design (Cont.)



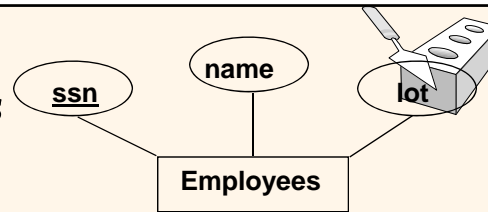
- A database `schema' in the ER Model can be represented pictorially (*ER diagrams*).
- ❖ Logical Database Design
 - Choose a DBMS to implement our database design.
 - Map an **ER diagram** into a **relational schema** (Chapter 3).

Overview of Database Design (Cont.)



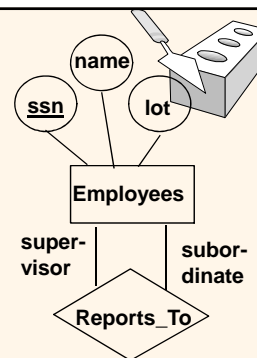
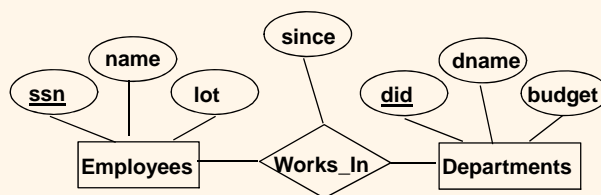
- ❖ Schema Refinement (Chapter 19)
 - Analyze the current schema to identify potential problems and to refine it (performance criteria).
 - Theory – *normalizing relations*.
- ❖ Physical Database Design (Chapter 20)
 - Build indices on some tables and clustering some tables.
- ❖ Application and Security Design (Chapter 21)
 - Consider aspects of the application that go beyond the database itself.

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** (primary key - **SSN**).
 - Each attribute has a **domain**.

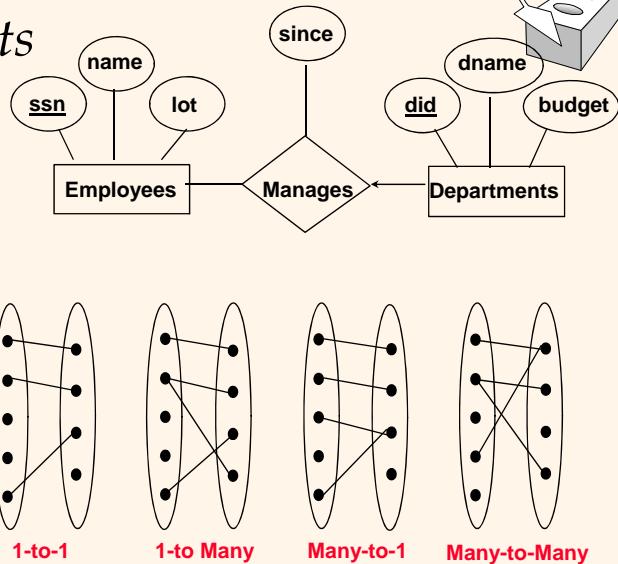
ER Model Basics (Contd.)



- ❖ **Relationship:** Association among two or more entities. E.g., Barbara works in the CS department.
- ❖ **Relationship Set:** Collection of similar relationships.
 - An n-ary relationship set R relates n entity sets $E_1 \dots E_n$; each relationship in R involves entities $e_1 \in E_1, \dots, e_n \in E_n$
 - Same entity set could participate in different relationship sets, or in different “roles” in same set.

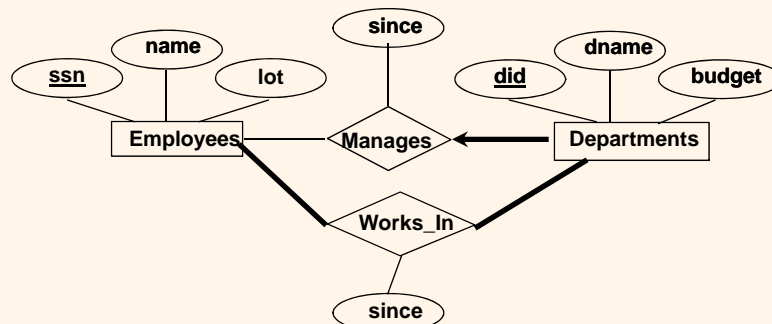
Key Constraints

- ❖ 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.



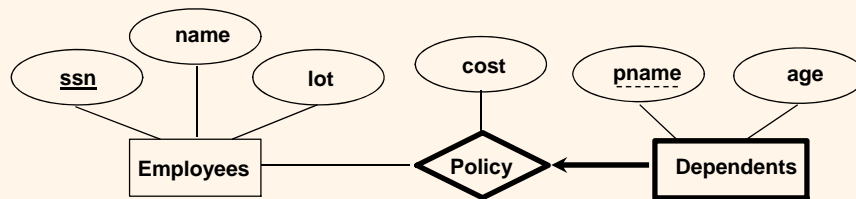
Participation Constraints

- ❖ 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.



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*.
 - Identifying owner - SSN + pname (*partial key*).



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*) *No by default*
- ❖ *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (*Yes/no*) *No by default*
- ❖ Reasons for using ISA:
 - To add descriptive attributes specific to a subclass.
 - To identify entities that participate in a relationship.

