Conceptual Design.
The Entity-Relationship (ER) Model

CS430/630
Lecture 12

Slides based on “Database Management Systems” 3rd ed, Ramakrishnan and Gehrke

Database Design Overview

- Conceptual design
  - The Entity-Relationship (ER) Model, UML
  - High-level, close to human thinking
  - Semantic model, intuitive, rich constructs
    - Not directly implementable
- Logical Design
  - The relational data model
  - Machine-implementable, fewer and more basic constructs
  - Logical design translates ER into relational model (SQL)
- Physical Design (not in this course)
  - Storage and indexing details

Conceptual Design – ER Model

- What are the *entities* and *relationships* in a typical application?
  - What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules*
  - Key constraints
  - Participation constraints
- Representation through *ER diagrams*
  - ER diagrams are then mapped into relational schemas
  - Conversion is fairly mechanical

Entities and Entity Sets

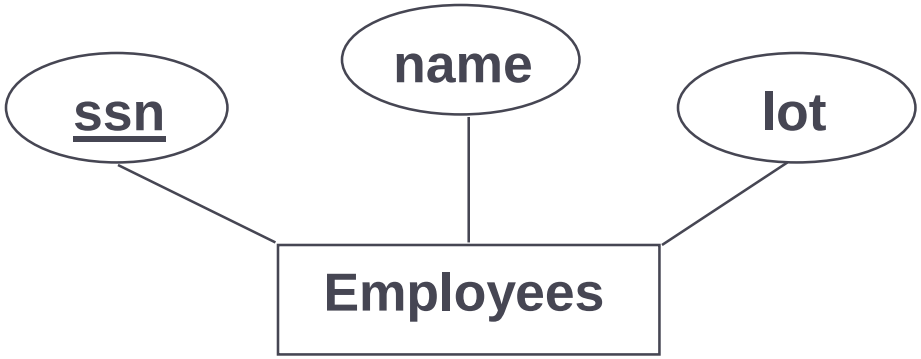
- *Entity:* represents a real-world object
  - Characterized using set of *attributes*
  - Each attribute has a *domain* – similar to variable types

- *Entity Set:* represents collection of similar entities
  - E.g., all employees in an organization
  - All entities in an entity set share same set of attributes

Keys

- Each entity set has a *key*
  - Set of attributes that uniquely identify an entity
  - Multiple *candidate keys* may exist
  - *Primary key* selected among them

Entity Set Representation



Representation Convention:

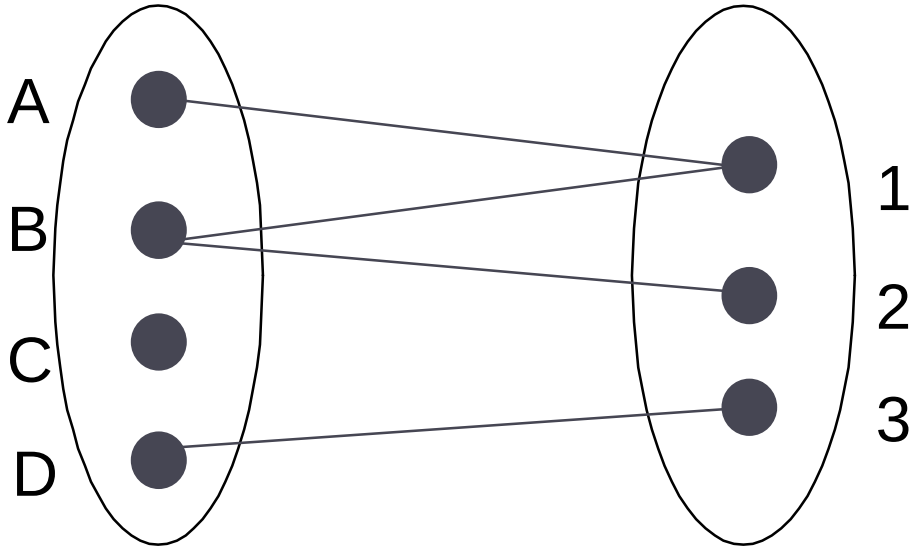
- Entity sets: rectangles
- Attributes: ovals, with key attributes underlined
- Edges connect entity sets to attributes

Relationships and Relationship Sets

- *Relationship*: Association among two (or more) entities
  - "Gabriel works in CS department"
  - Can have descriptive attributes: e.g., "since 9/1/2011"
    - But relationship must be fully determined by entities!
  - *Binary*, *ternary* or *multi-way (n-way)* relationships

- *Relationship Set*: Collection of similar relationships
  - Contains *n*-tuples $(e_1, \ldots, e_n)$, where e_i belongs to entity set E_i
  - *Instance*: "snapshot" of relationship set at some point in time

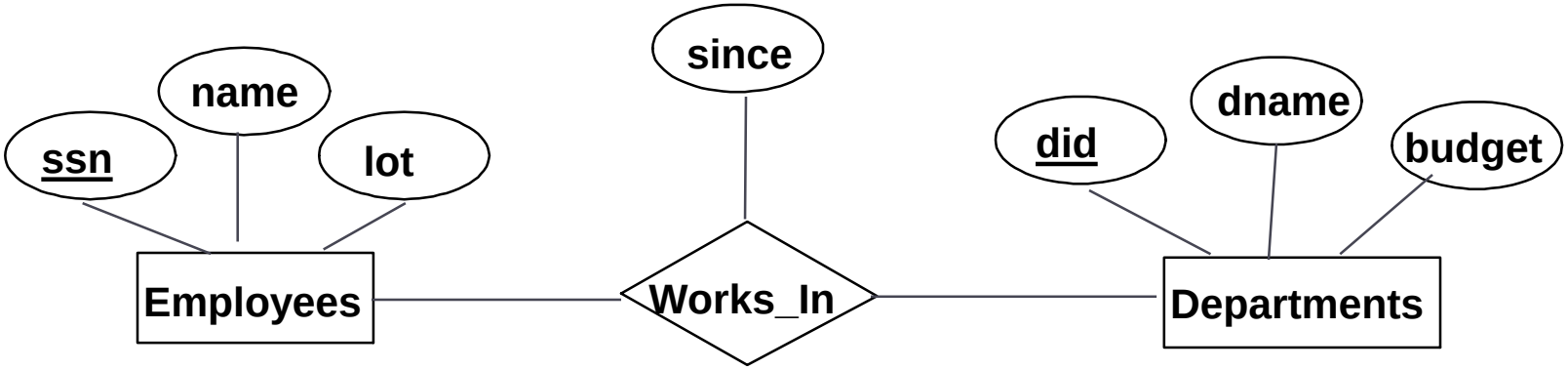
Visualizing Relationships and Rel. Sets



(A, 1)
(B, 1)
(B, 2)
(D, 3)

Edge = Relationship
Set of Edges = Relationship Set

Relationship Set Representation

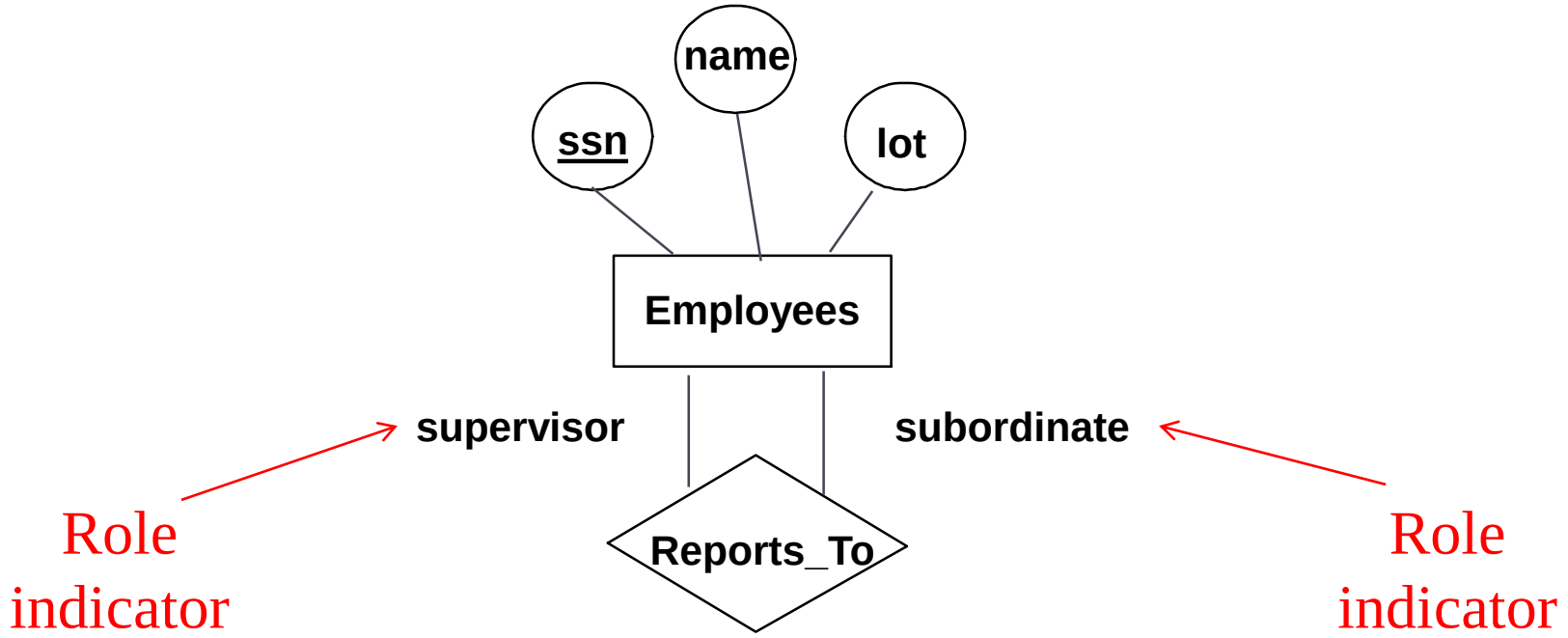


Representation Convention:

- Relationship sets: diamonds
- Edges connect relationship sets to entity sets, and relationship sets to relationship set attributes

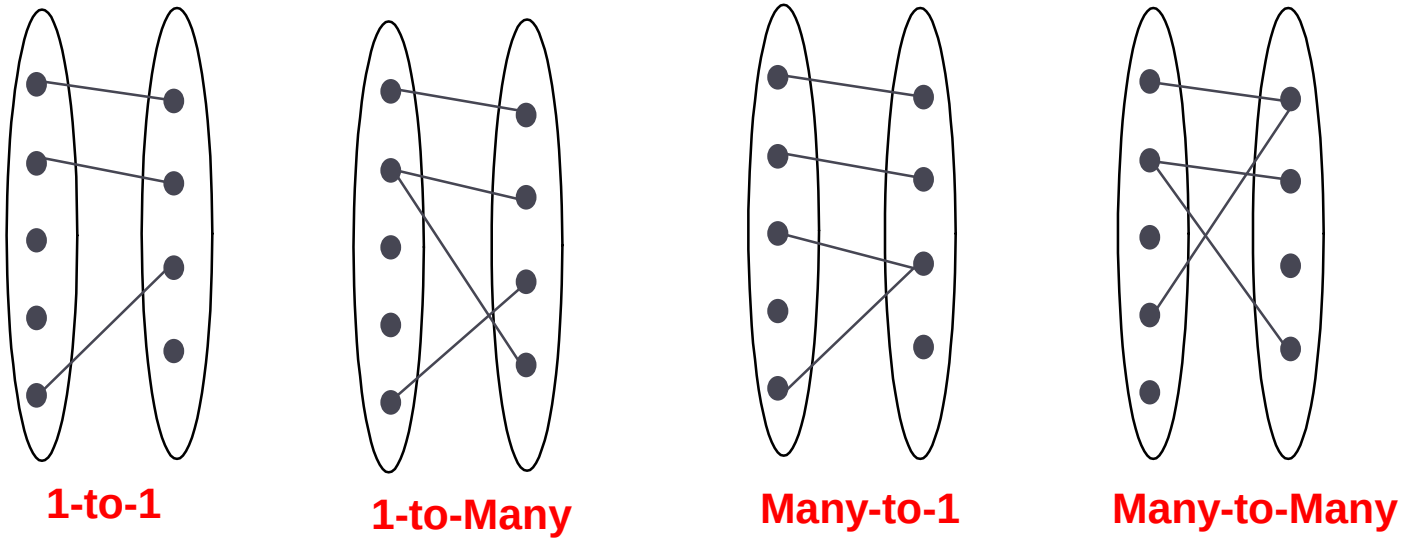
A Special Case of Relationship

- An entity set can participate in a relationship set with itself
  - Entities in same set play different ***roles*** in the relationship
  - ***Role indicators*** express the role



Key Constraints

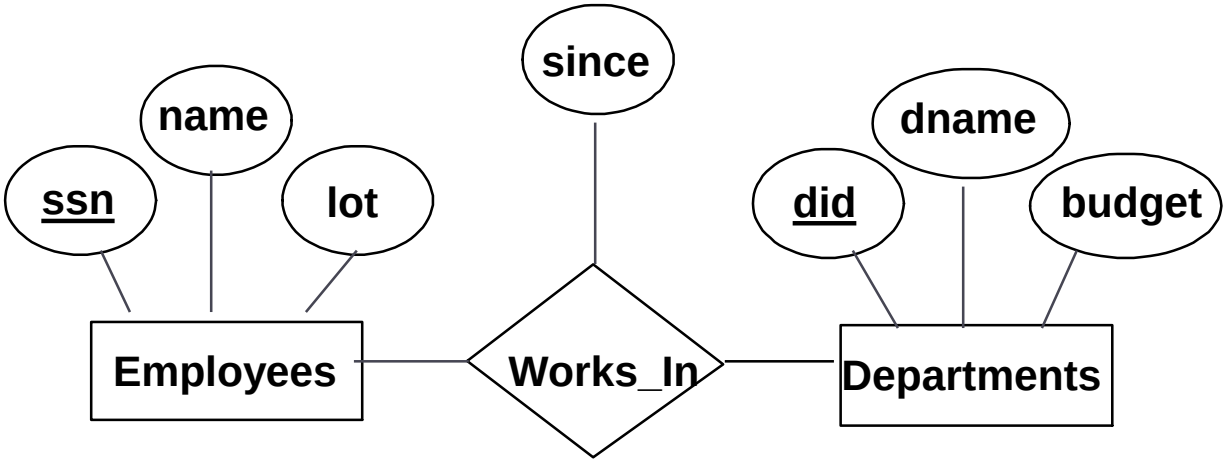
- How many other entities can an entity have a relationship with?
  - Also referred to as relationship *multiplicity*



Example 1

- Works_In relationship: an employee can work in many departments; a dept can have many employees.

many-to-many



Example 2

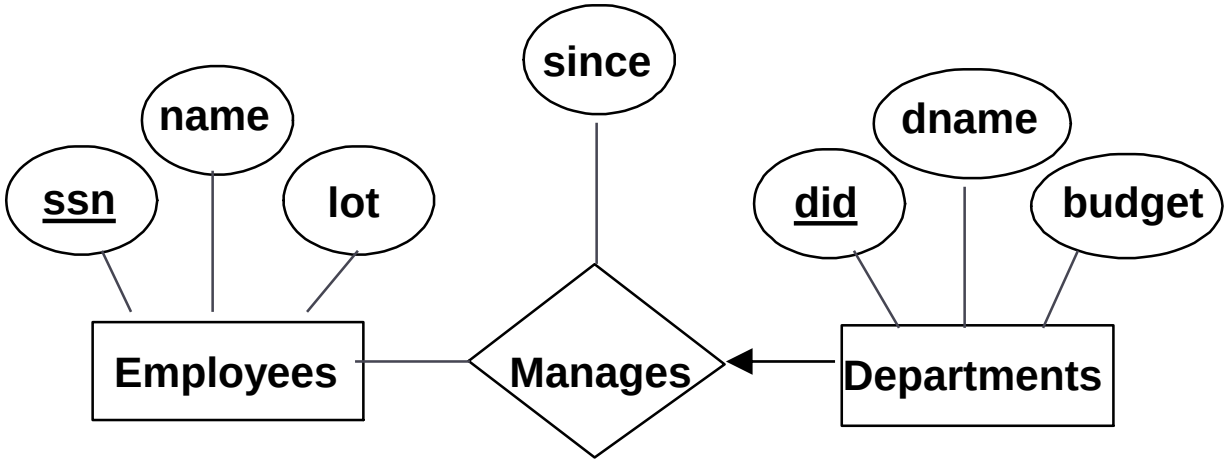
- Manages relationship: each dept has *at most one* manager

one-to-many

from *Employees* to *Departments* , *or*

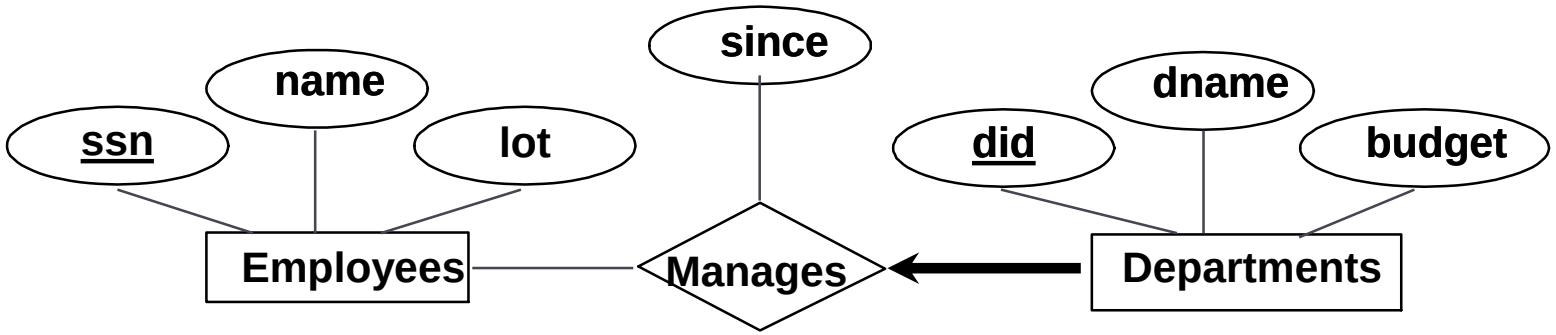
many-to-one

from *Departments* to *Employees*

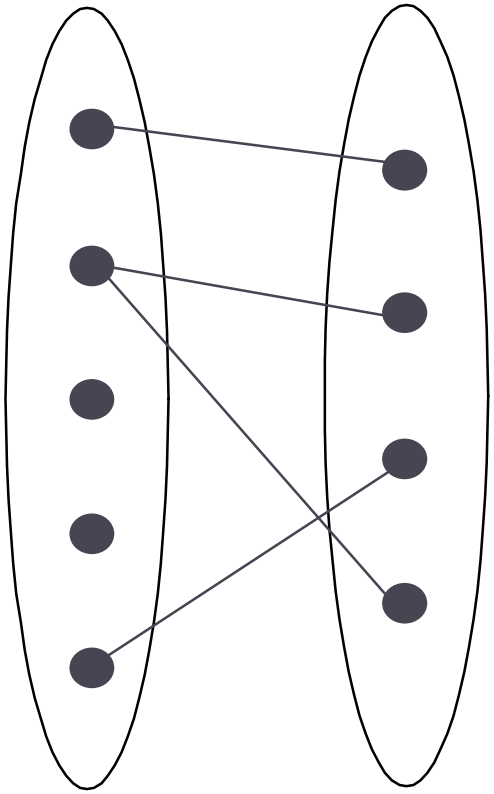


Participation Constraints

- *Total* vs *Partial* Participation
  - Total: every department must have a manager
    - "Departments" entity set has total participation in relationship
    - Represented as thickened line (there is a key constraint as well)
  - Partial: not every employee is a manager
    - "Employees" entity set has partial participation



Participation Constraints



Partial Participation

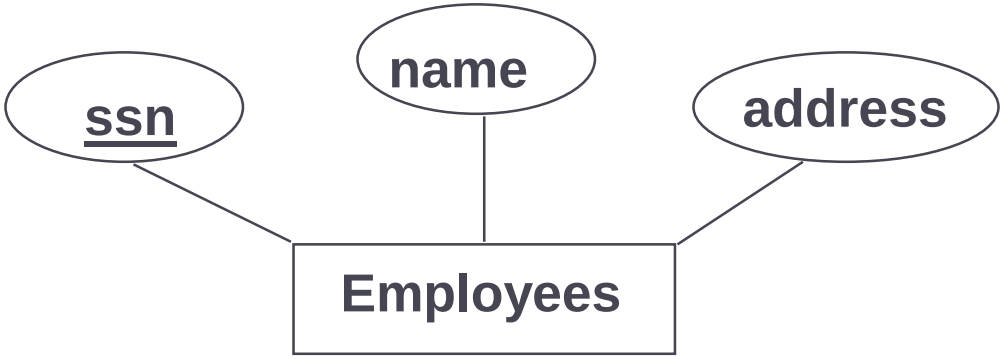
Total Participation

Design Choices in the ER Model

- Should a concept be modeled as an entity or an attribute?

- Should a concept be modeled as an entity or a relationship?
  - Considers hierarchies and inheritance
  - Outside the scope of this class

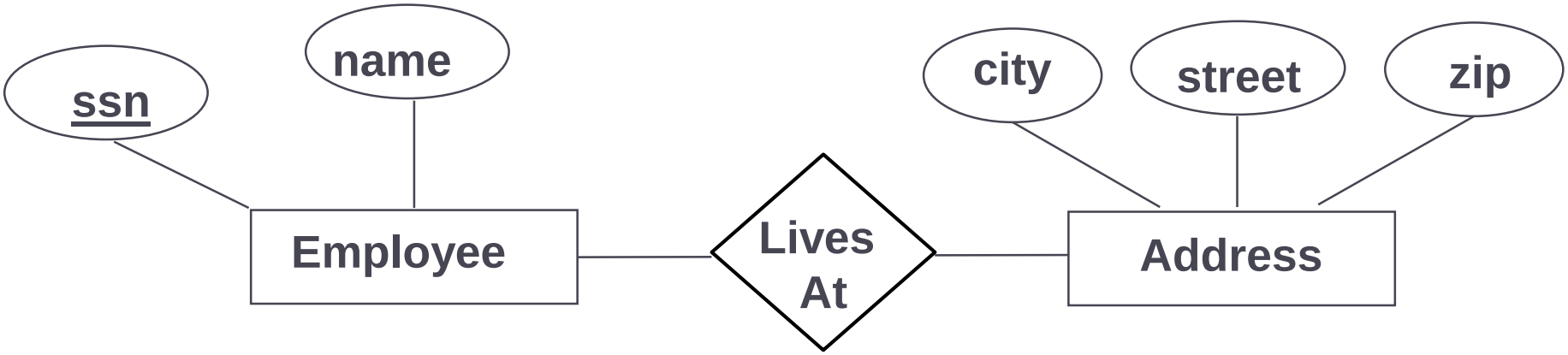
Entity vs. Attribute



- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?

Entity vs. Attribute

- Sometimes address may have to be an entity:
  - If we have several addresses per employee (since attributes cannot be set-valued)
  - If the structure (city, street, etc.) is important, e.g., retrieve employees in a given city (attribute values are atomic!)



Example

Design a database for a bank, including information about customers and their accounts. Information about customers includes their name, address, phone and SSN. Accounts have numbers, types (e.g., savings/checking) and balances.

1. Draw the E/R diagram for this database.
2. Modify the E/R diagram such that each customer must have at least one account.
3. Modify the E/R diagram further such that an account can have at most one customer.