CptS 451- Introduction to Database Systems

Conceptual Modeling of Data and ER Model

(DMS Ch-2)

Instructor: Sakire Arslan Ay

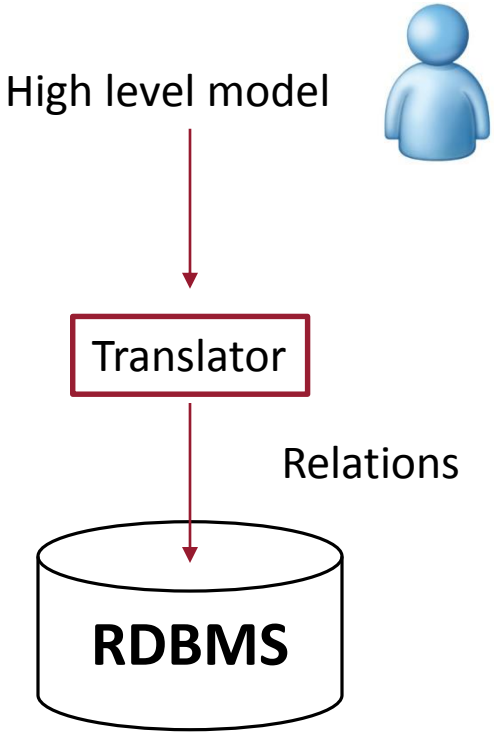
WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Outline

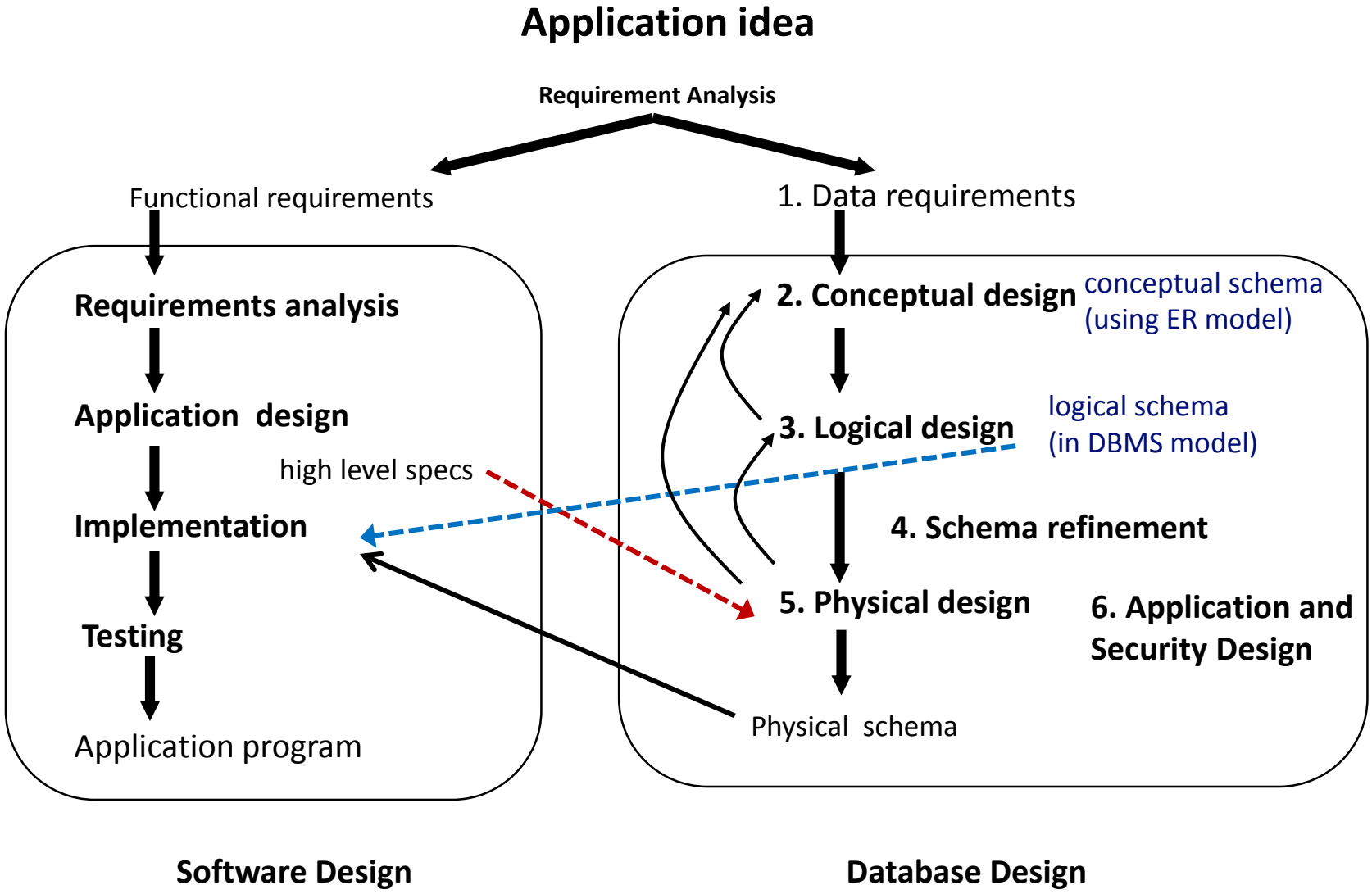
- Database design process
- Entity/Relationship Model
- Examples
- Good Database Design Principles

Database Design Process

- Data modeling



Database Design Process



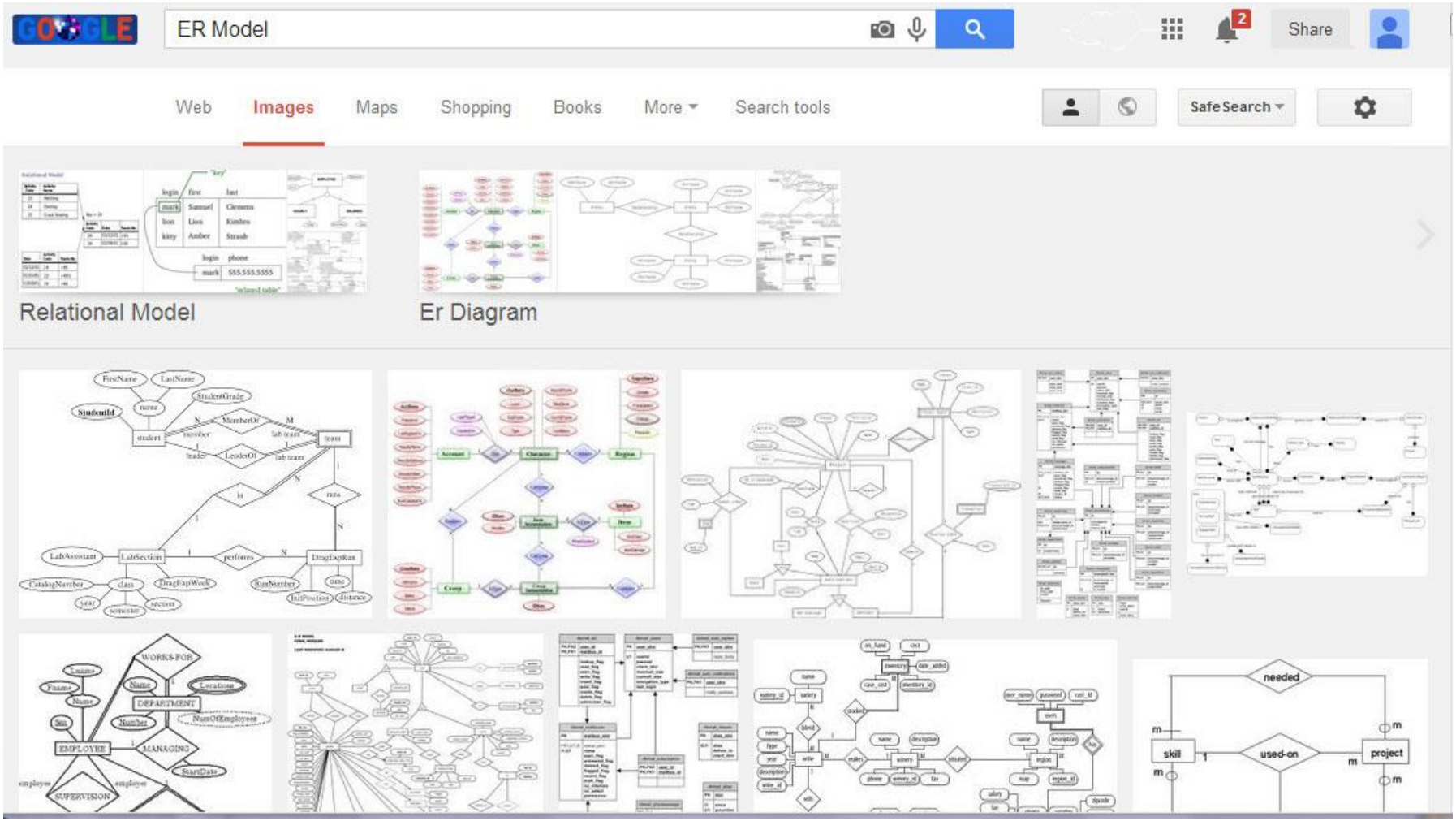
Database Design: A 6-Step Program

1. Requirements Analysis: data requirements, critical operations on the data
2. Conceptual DB Design: high-level description of data and constraints - typically using ER model
3. Logical DB Design: conversion into a schema
   - pick a type of DBMS, relational DBMS is most popular and is our focus
4. Schema Refinement: normalization (eliminating redundancy)
5. Physical DB Design: consider workloads, indexes and clustering of data
6. Application/Security Design

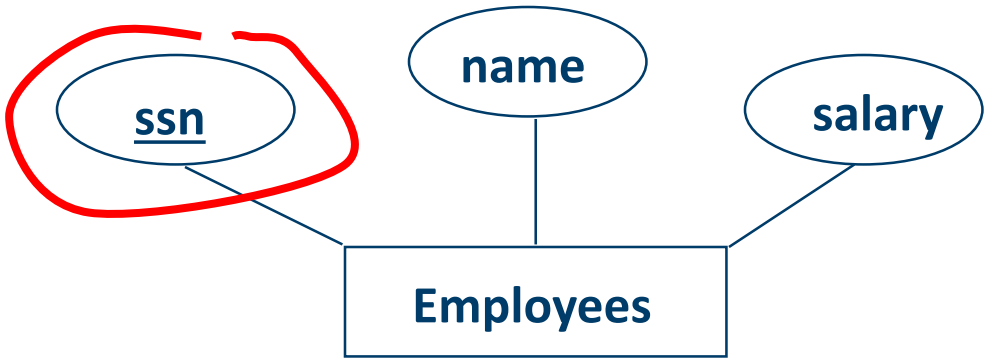
Overview of Entity-Relationship (ER) Model

- A visual model to specify
  - what information the database must hold
  - what are the relationships among components of that information
- Proposed by Peter Chen in 1976
- What we will cover
  - Basic stuff (entities, relationships, roles)
  - Constraints
  - Weak entity sets
  - Multi-way relationships
  - Subclass/superclass Relationships
  - Aggregation

ER Model


ER Model
Share
Web
Images
Maps
Shopping
Books
More
Search tools
Safe Search
Relational Model
Er Diagram

ER Model

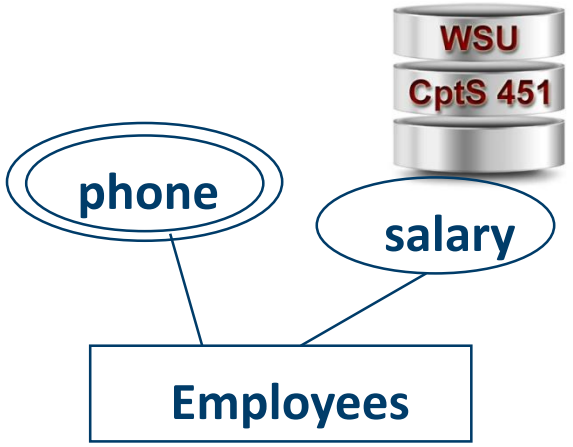


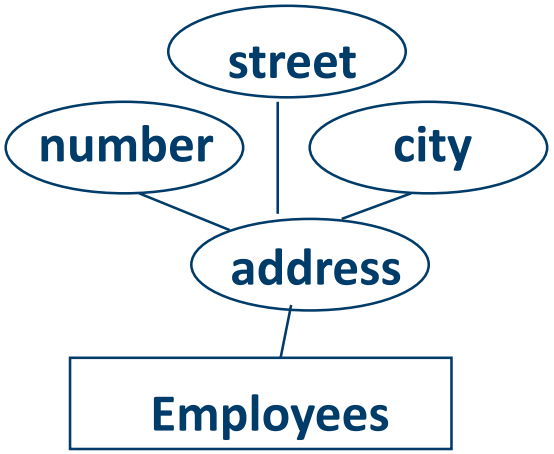
- ***Entity***: Real-world object, distinguishable from other objects,
  - e.g., 'an employee', 'John Yates'
  - An entity is described (in DB)using a set of ***attributes***.
- ***Entity Set***: A collection of similar entities. e.g., all employees.
  - All entities in an entity set have the same set of attributes.
  - Each entity set has a ***key*** *(underlined)*.
  - Each attribute has a ***domain***.

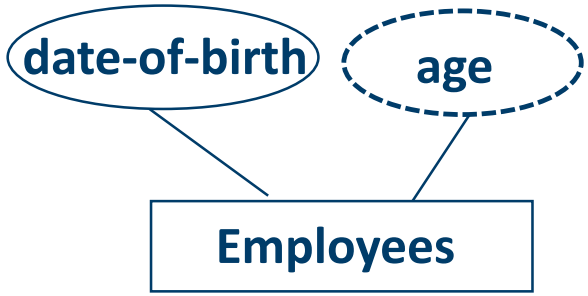
$\Rightarrow$ In OO terminology, an entity set is similar to a class, and an entity similar to an instance

Entity Set- Attributes

- *single-valued vs multi-valued:*
  - phone number of an employee could be multi-valued
  - salary of employee is single-valued
- *atomic vs composite*:
  - salary of a person is atomic
  - address of a person could be composite
- *stored vs derived:*
  - derived attributes are those that can be derived from other attributes or entities, e.g., age can be derived from date of birth.
  - All other attributes are stored attributes



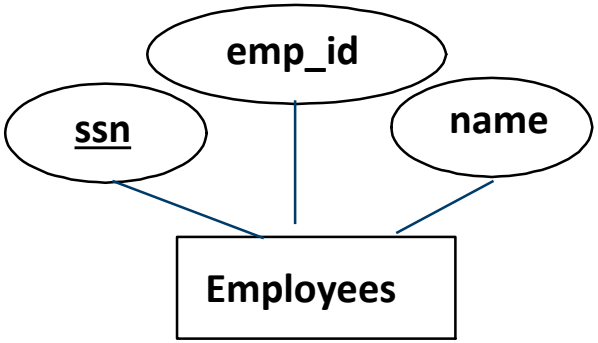


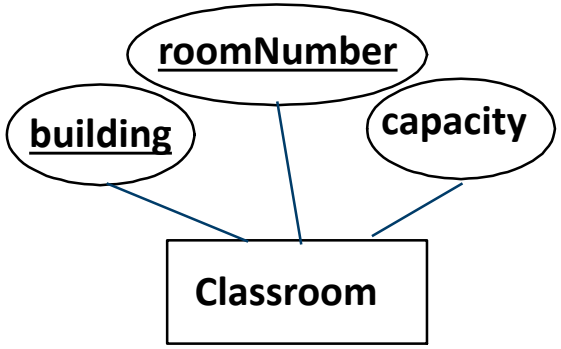


Entity Set: Keys and Domains

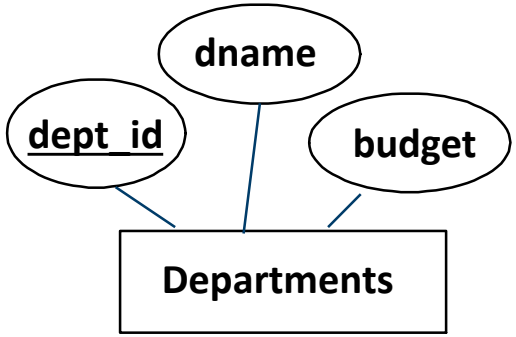
Key attributes of an entity:

- With each entity set a notion of a ***key*** is associated.
- A key is a set of attributes that uniquely identify an entity in entity set.
- Denoted in ER diagram by underlining the attributes that form a key
- Examples:
  - designer may specify that {ssno} is a key for a entity set employees
  - designer may specify that {emp_id} is also a key , that is, no joint accounts are permitted.
- Multiple keys may exist in which case one chosen as ***primary key*** and underlined.
  - Other keys are called candidate keys





The key is a set of attributes (composite key).

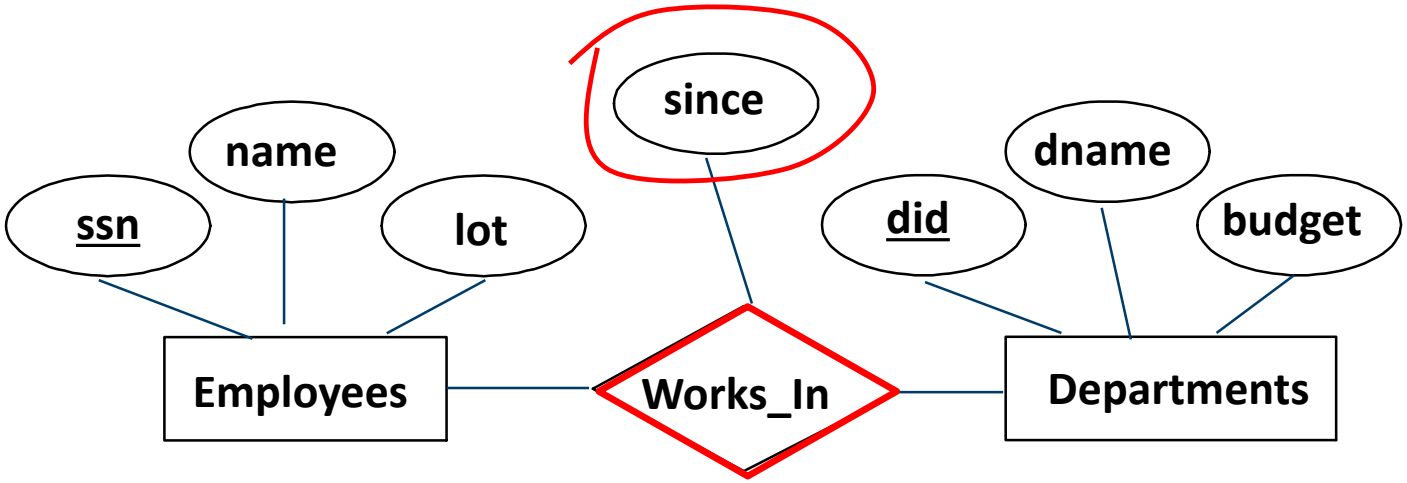


Entity Set: Keys and Domains (cont.)

Domains of Attributes:

- A domain is associated with each simple attribute.
- The value of the attribute for the entity is constrained to be in the domain.

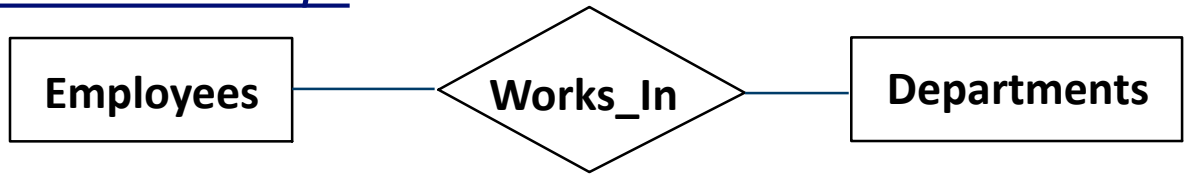
ER Model - Relationships



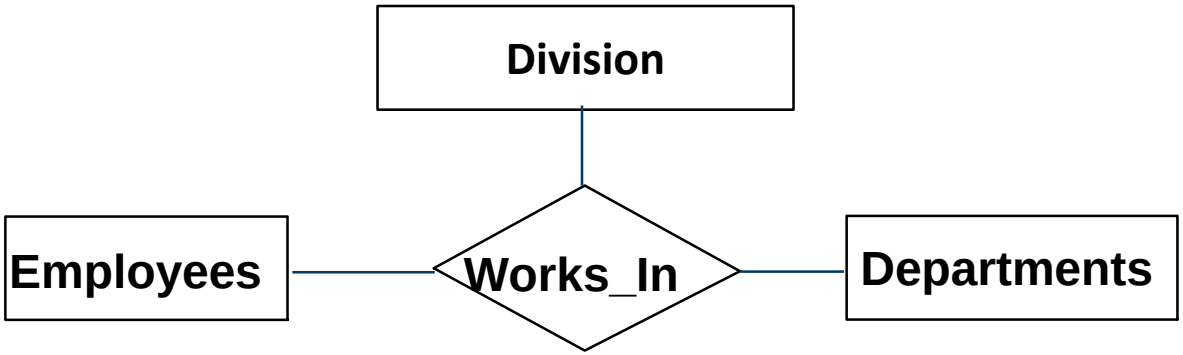
- *Relationship*: Association among two or more entities. E.g., "John Yates works in EECS department".
- *Relationship Set*: Collection of similar relationships.
  - relationships may have their own attributes.
  - Binary, Ternary, 4-nary, … relationship sets

ER Model – Relationships (cont.)

- A *binary relationship* involves two entities :

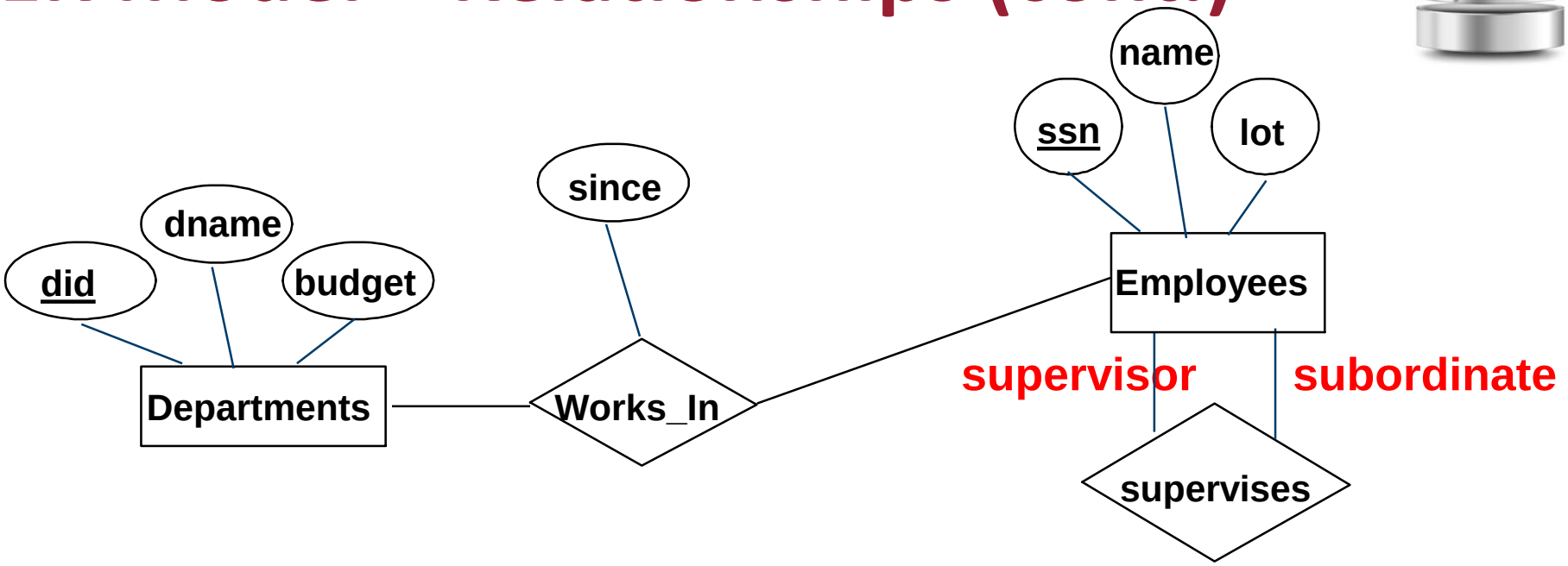


- A *ternary relationship* involves three entities :



- An *n*-ary relationship set *R* relates *n* entity sets $E_1 \ldots E_n$; each relationship in *R* involves entities $e_1 \in E_1, \ldots, e_n \in E_n$

ER Model – Relationships (cont.)



- Same entity set can participate in different relationship sets, or in different "*roles*" in the same relationship set .
- The function that an entity plays in a relationship is called its *role*
  - Roles are normally not explicitly specified unless the meaning of the relationship needs clarification
  - Roles needed when entity set is related to itself via a relationship.

Constraints on Relationship Sets

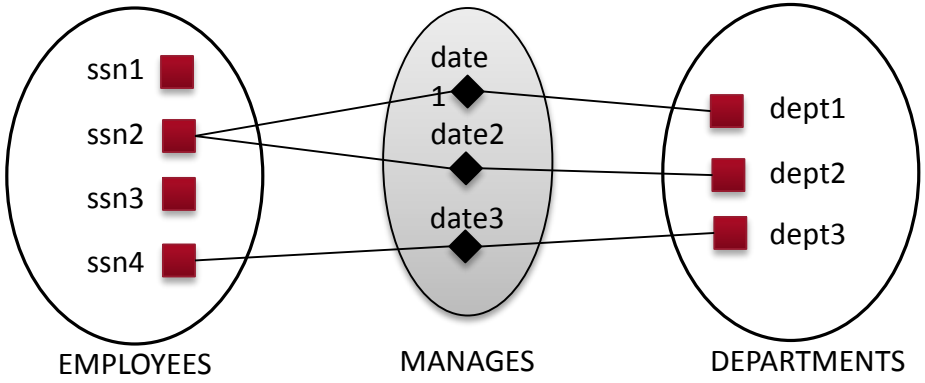
Limit the possible combinations of entities that may participate in the corresponding relationship set.

1) Key Constraints (or Multiplicity)

- an *employee* is allowed to manage more than one department, but each department has at most one manager *(one-to-many).*

2) Participation Constraints

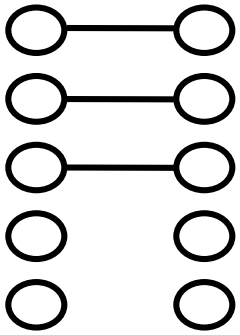
- *Not all employees are managers (partial participation).*
- *Each department needs to have a manager (total participation)*



Key Constraints (Multiplicity)

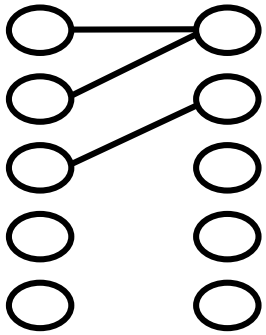
Consider binary relationship set R between entity sets **A** and **B**

- ***One to one:*** an entity in **A** is associated with at most one entity in **B**, and an entity in **B** is associated with at most one entity in **A**.
- ***Many to One*****:** An entity in **A** is associated with at most one entity in **B**, an entity in **B** is associated with many entities in **A**.
- ***Many to Many:*** An entity in **A** is associated with many entities in **B**, and an entity in **B** is associated with many entities in **A**.



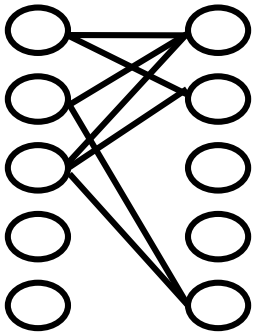
One-to-one

an *employee* has only one *spouse* in a *married-to* relationship



Many-to-one

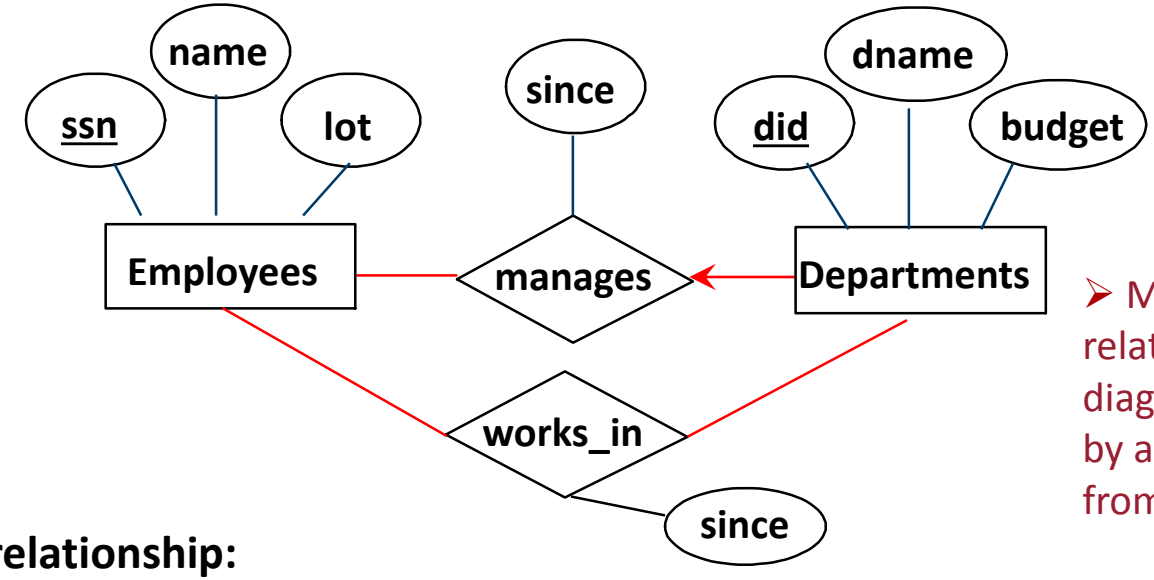
an *employee* works in a single *department* but a *department* consists of many *employees.*



Many-to-many

A *customer* may have many *bank accounts*. *Accounts* may be joint between multiple *customers*

Key Constraints (Multiplicity)(cont.)



➢ Multiplicity of relationship in ER diagram is represented by an arrow pointing from "many" to "one"

→ **"works_in" relationship:**

- An employee can work in many departments; a department can have many employees (many-to-many).

→ **"manages" relationship:**

- Each dept has at most one manager, but one employee can manage multiple departments, according to the *key constraint* on manages (one-to-many).

illegal

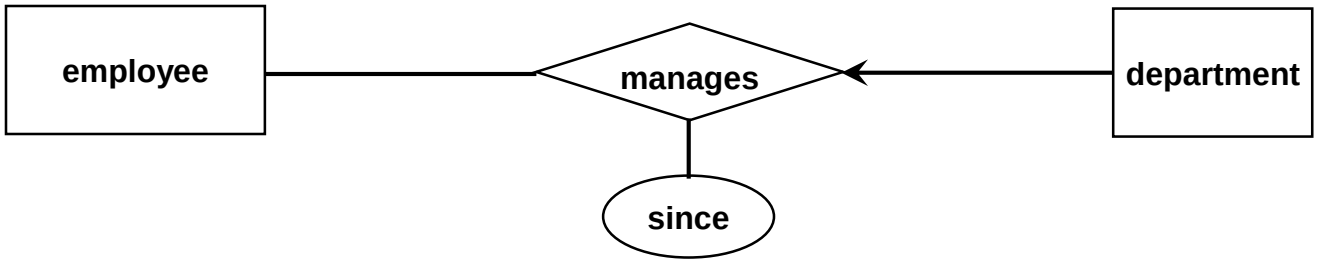
Manages	Department	Since
John Yates	EECS-ITS	1/1/1995
Vasiliy Bunakov	EECS-ITS	1/1/2011

legal

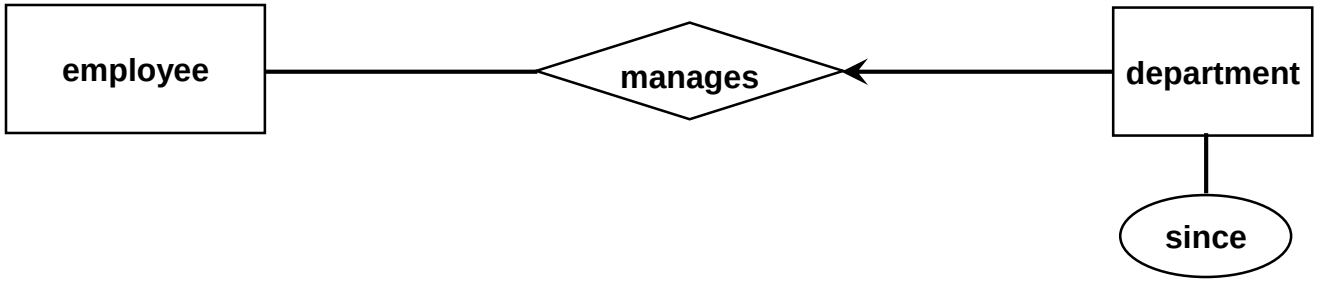
Manages	Department	Since
John Yates	EECS-ITS	1/1/1995
John Yates	CEA-ITS	1/1/2000

Key Constraints (Multiplicity)(cont.)

- **Relationship Attribute in a Many to One Relationship**

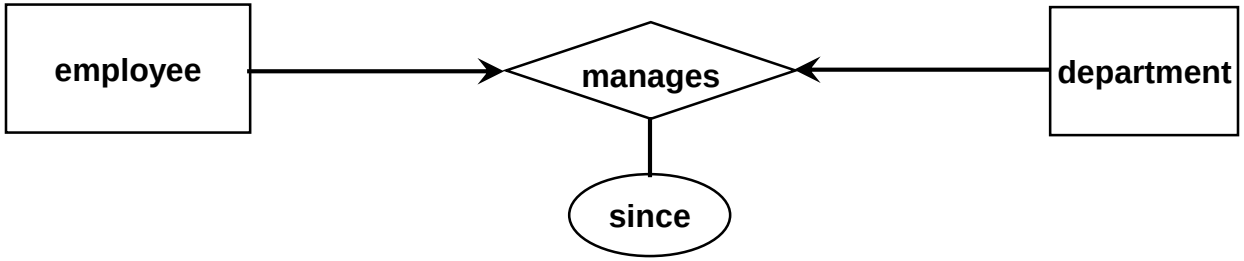


— In a Many-to-One relationship, relationship attributes can be repositioned to the entity set on the **many side**.

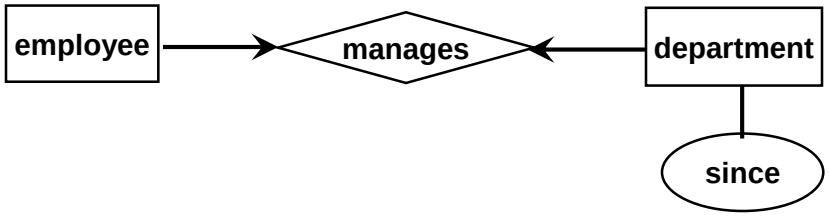


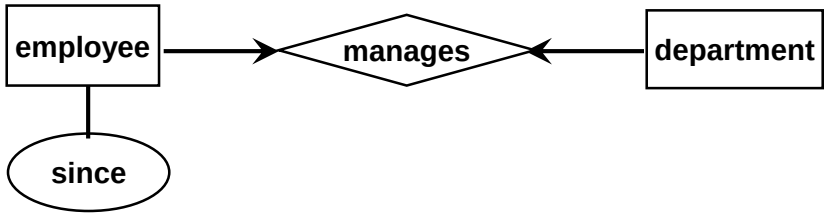
Key Constraints (Multiplicity)(cont.)

- **Relationship Attribute in a One to One Relationship**
  - If an additional constraint states that an employee can be a manager for at most one department, than "manages" will be a one-to-one relationship.
    - 1 employee can manage 1 department
    - 1 department can be manages by 1 employee



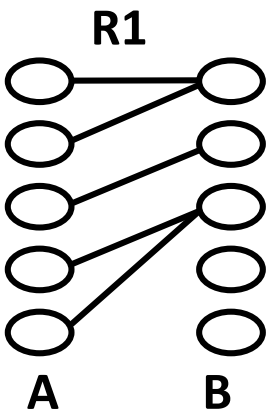
  - In a One-to-one relationship, relationship attributes can be shifted to either of the entity sets.





Participation Constraints

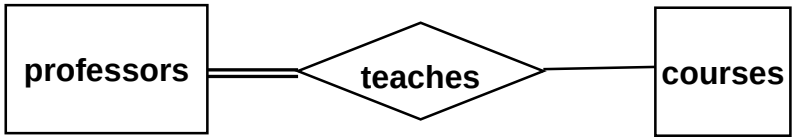
- ***Total*** vs. **Partial** participation



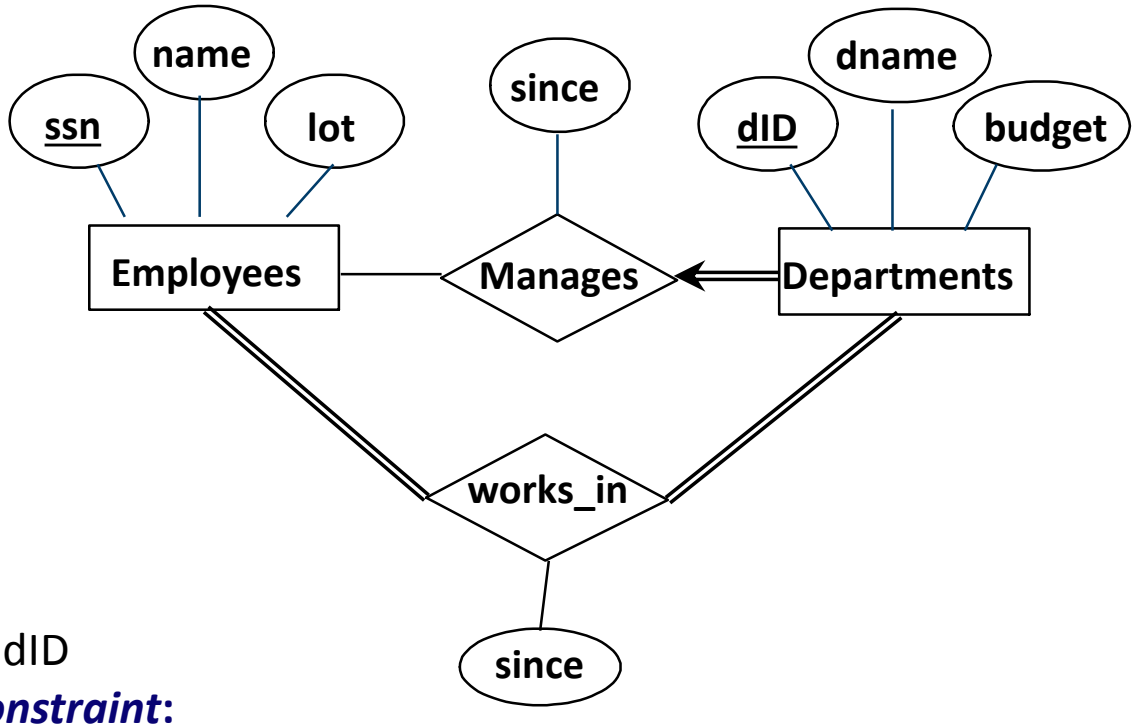
  - Participation of an entity set A in the relationship set R1 is ***total***
    - Each entity in entity set A is constrained to be related to other entities via relationship R1.
  - Participation of an entity set B in the relationship set R1 is ***partial***
    - Not every entity in entity set B is constrained to be related to other entities via relationship R1.

- *Example*
  - Suppose each *professor* must *teach* at least one *course*
  - Participation of entity set *professor* in the relationship *teaches* with the entity set *courses* is total.

➢ In ER diagram, total participation represented using a double line between the relationship and entity set that totally participates in the relationship

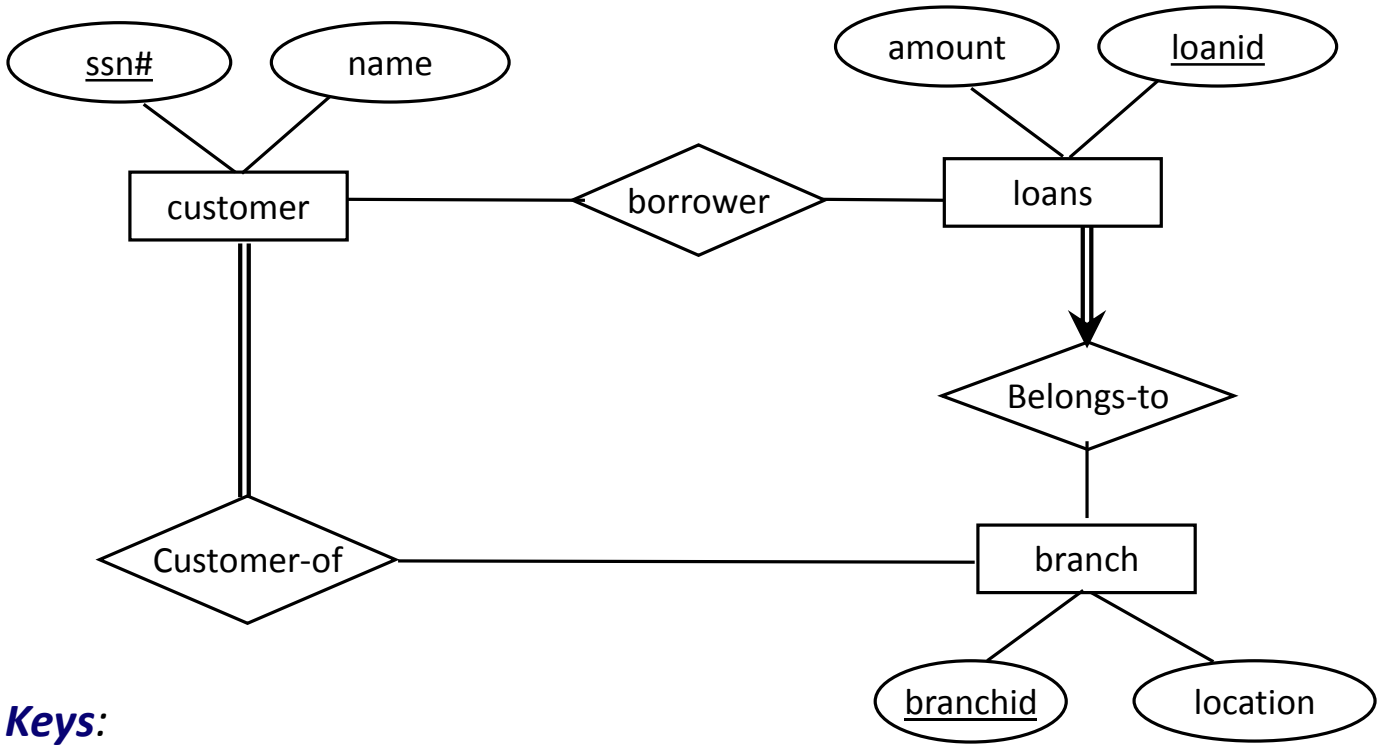


Example-1



- ***Keys***:
  - ssn, dID
- ***Key constraint***:
  - each department has a single manager
- ***Participation constraints***:
  - each employee works in at least one department
  - each department has at least one employee
  - each department has a manager

Example-2

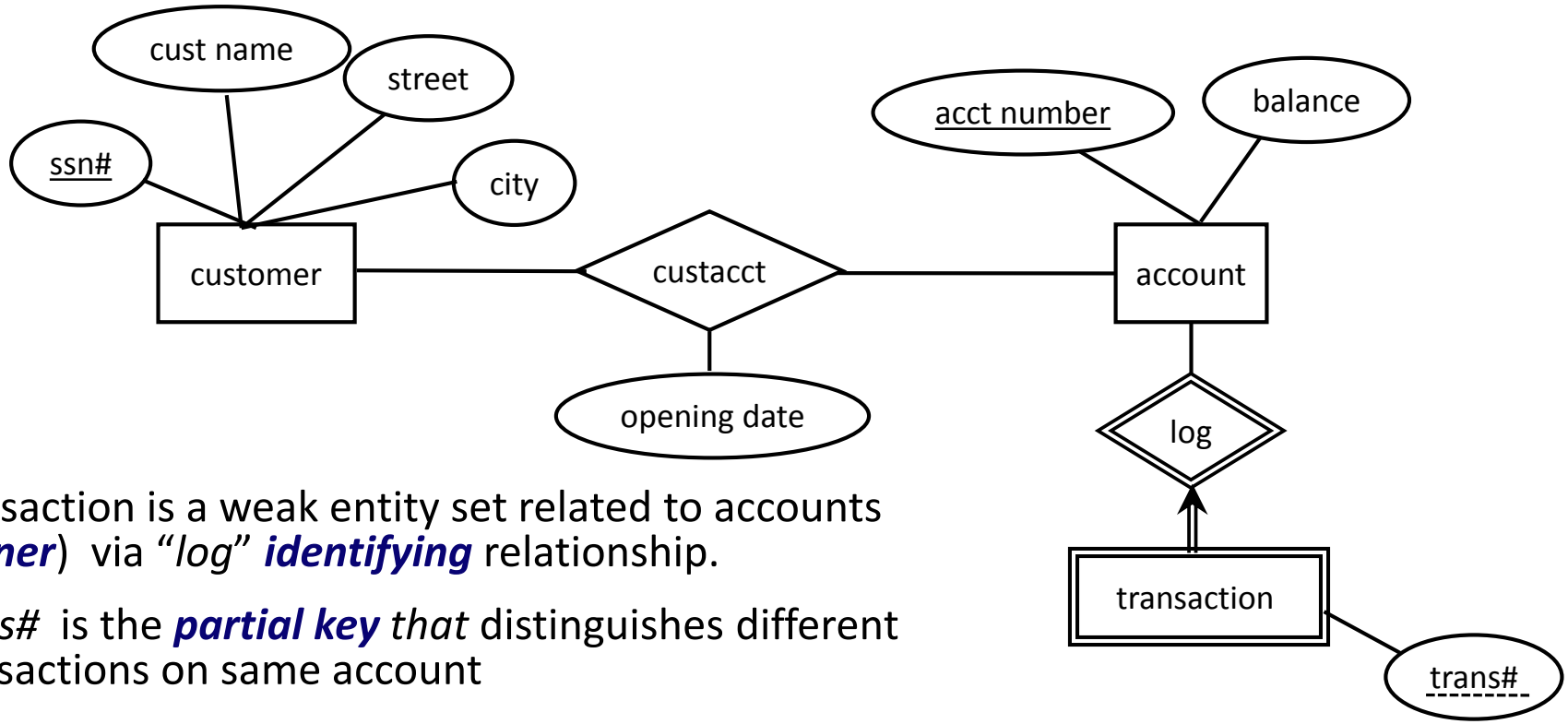


- ***Keys:***
  - ssn#, loanid, branchid
- ***Key constraint*:**
  - each loan belongs to a single branch
- ***Participation constraints*:**
  - each customer must be a customer of at least one branch
  - each loan must belong to some branch

Weak Entity Sets

- Entity sets that do not have sufficient attributes to form a key are called ***weak entity sets***.
- A weak entity can be identified uniquely only with 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.
- A weak entity set may have a ***partial key*** that distinguish between weak entities related to the same strong entity
  - key of weak entity set = key of owner entity set(s) + partial key

Weak Entity Sets - Example

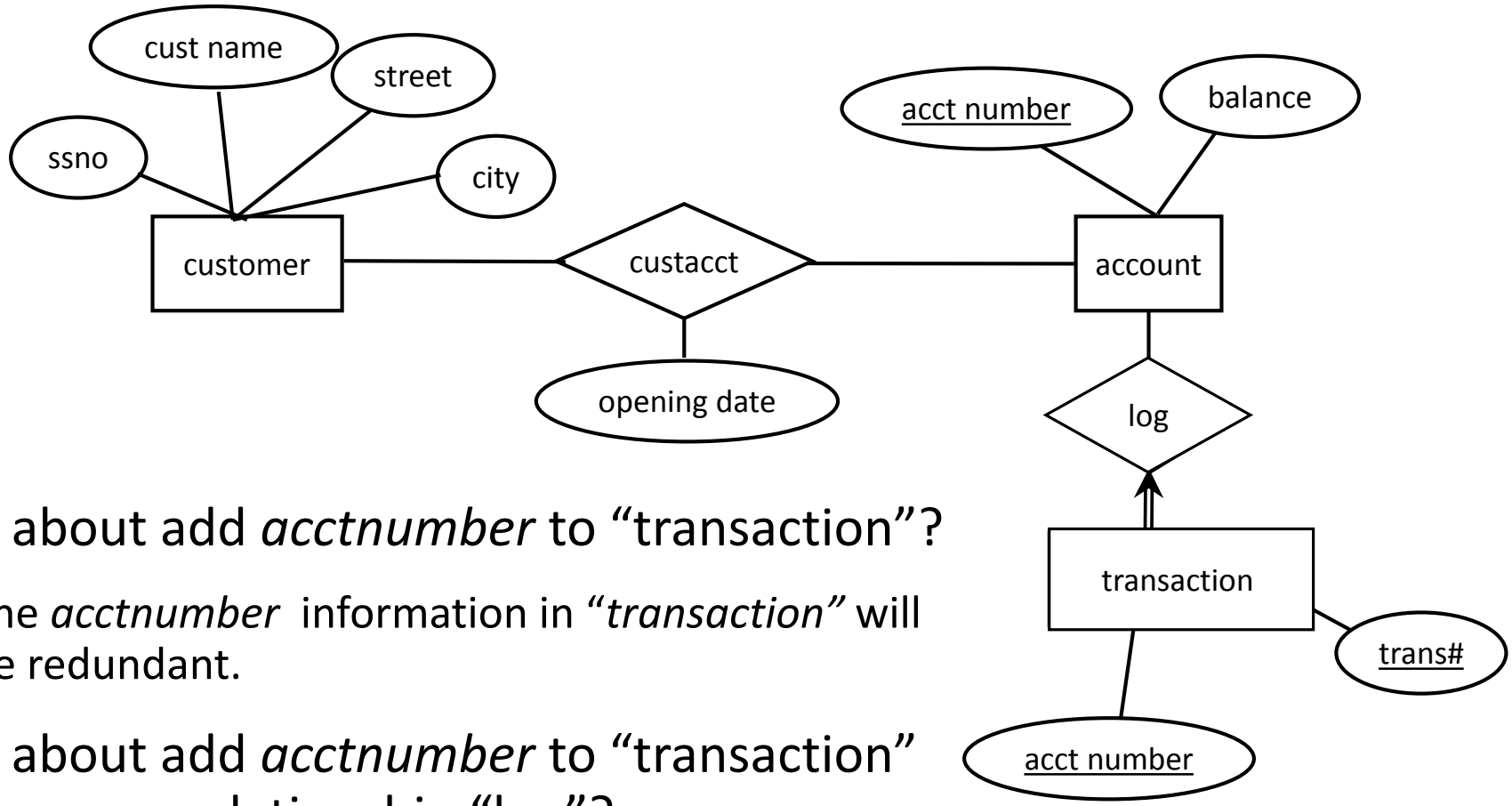


- Transaction is a weak entity set related to accounts (***owner***) via "*log*" ***identifying*** relationship.
- *trans#* is the ***partial key*** *that* distinguishes different transactions on same account
- In ER diagram:
  - weak entity is identified with double rectangle
  - the relationship set connecting the weak entity set to the owner entity id depicted by a double diamond.
  - "*partial key*" is identified with dashed underline

(note the difference in notation in different textbooks)

- Key of "*transaction*" = (*acctnumber, trans#*)

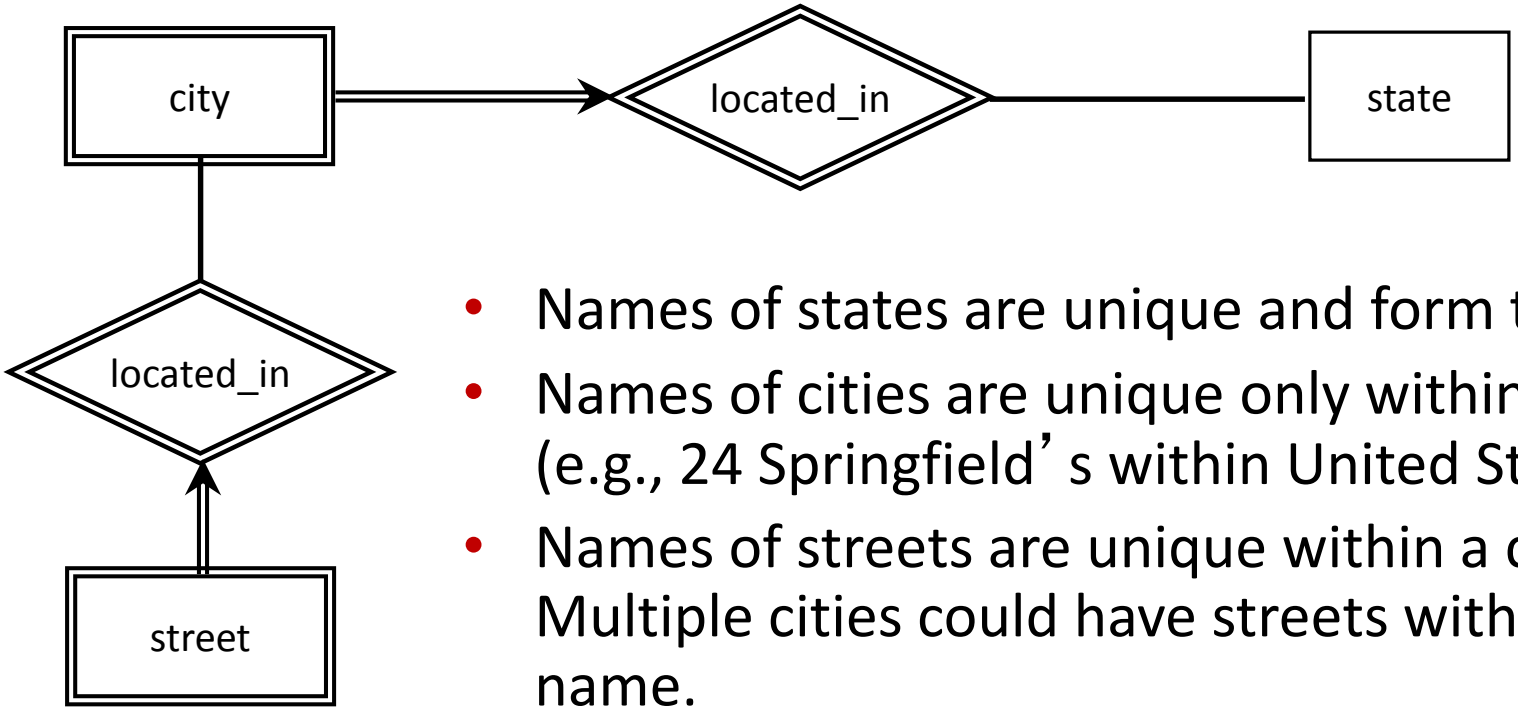
Weak Entity Sets (cont.)



- How about add *acctnumber* to "transaction"?
  - The *acctnumber* information in "*transaction*" will be redundant.
- How about add *acctnumber* to "transaction" and remove relationship "log"?
  - The relationship between "*account*" and "*transaction*" becomes implicit in an attribute which is not desirable.

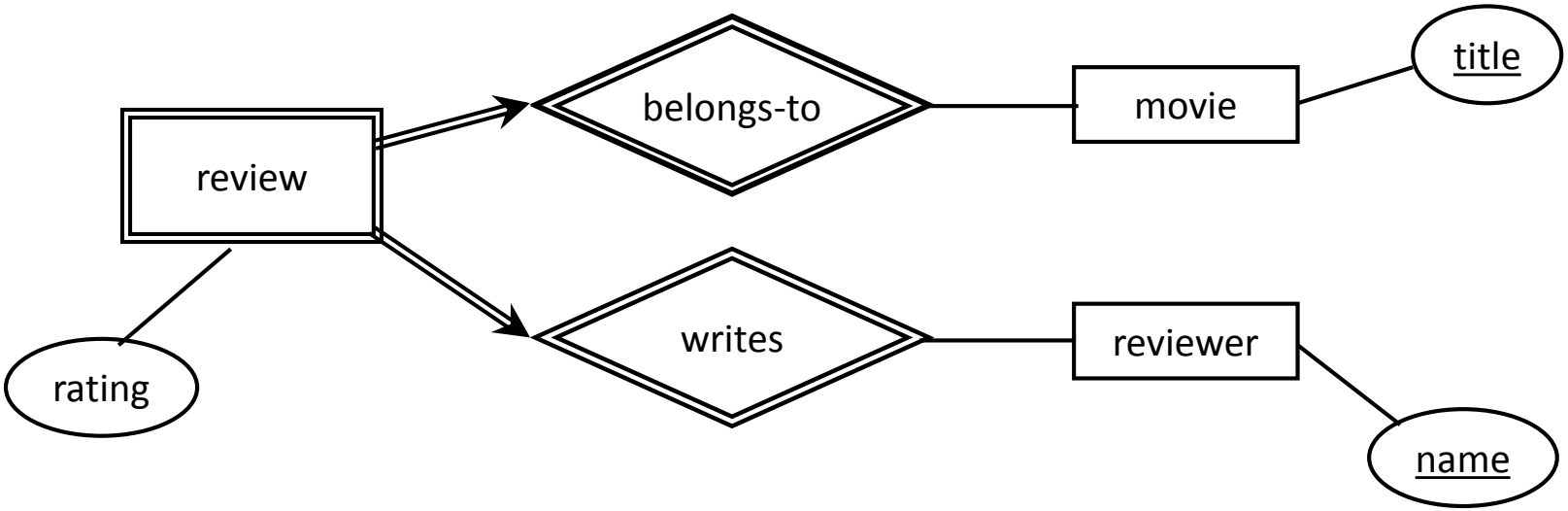
A Chain of Weak Entity Sets

- **Example:**
  - **a weak entity set might itself participate as owner in an identifying relationship with another weak entity set.**



- Names of states are unique and form the key.
- Names of cities are unique only within a state (e.g., 24 Springfield's within United States).
- Names of streets are unique within a city. Multiple cities could have streets with the same name.

A Weak Entity Set with Multiple Owner Entity Sets



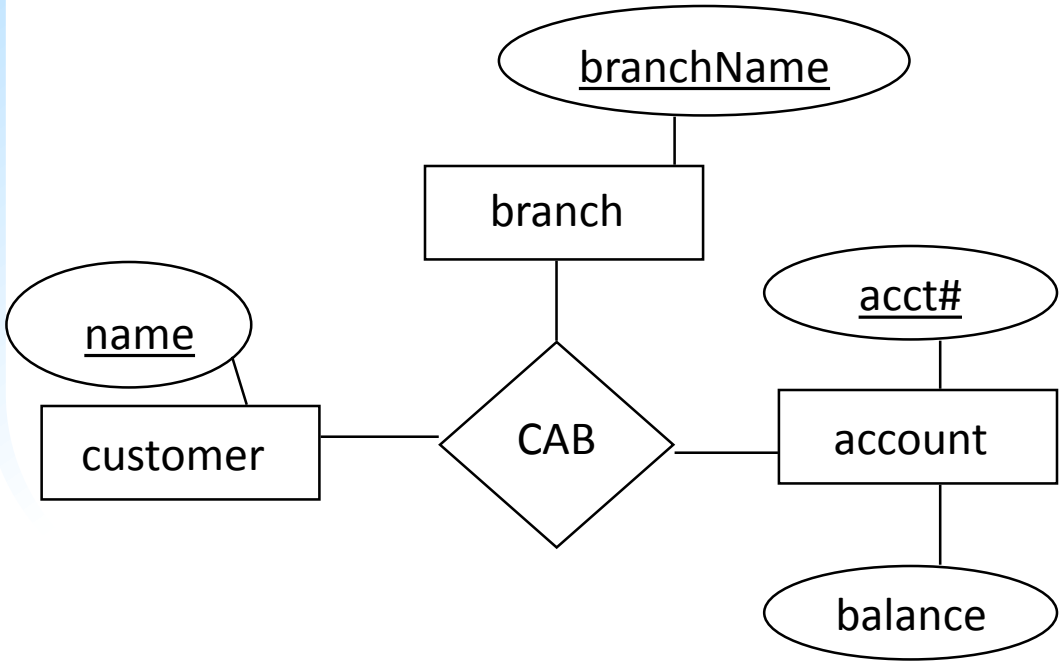
- Reviewers review movies and assign a rating for each movie
  - thumbs up 👍/thumbs down 👎.
- Review is a weak entity set whose owner sets correspond to both the movie and the reviewer entity sets.
- Key for the review entity set = key of movie + key of reviewer

Weak Entity Sets

- **Caution! Don't Overuse Weak Entity Sets**
  - Beginning database designers often tend to make all entity sets weak, supported by all other entity sets to which they are linked.
  - In reality, we usually create unique ID's for entity sets.
    - Examples include social-security numbers, automobile VIN's etc, registration numbers, etc. .

Multiway Relationships

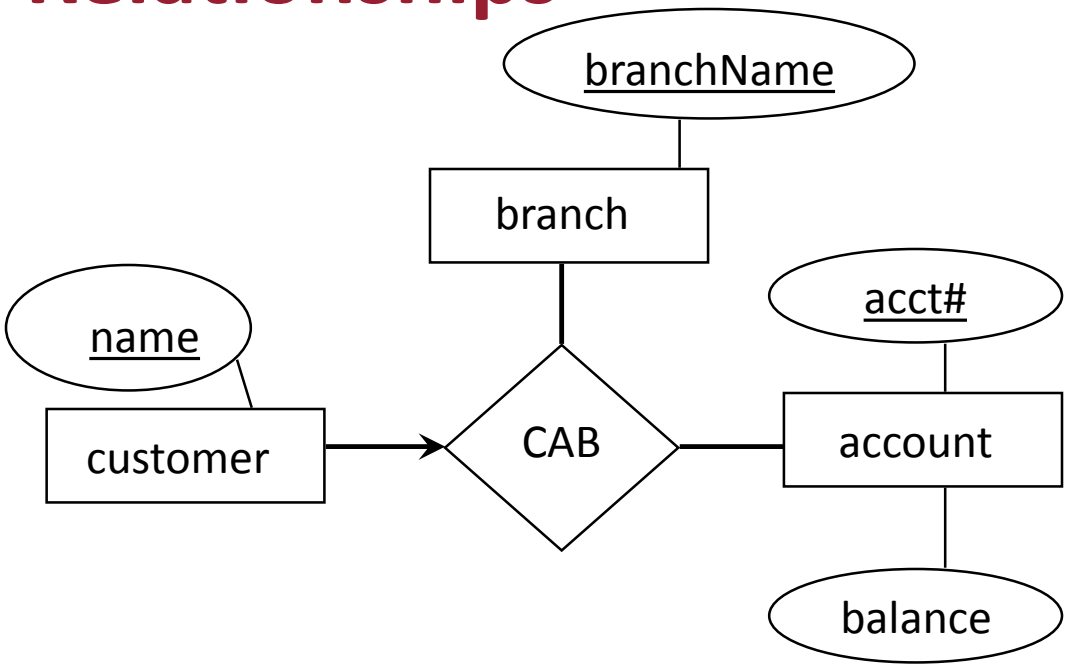
- Usually binary relationships (connecting two entity sets) suffice
- However, there are some cases where three or more entity sets must be connected by one relationship.
- Similar to binary relationship, key and participation constraints can be defined over **multiway** relationships



name	acc#	branchName	balance
John	1001	Pullman	2,000
Megan	1001	LA	1,500
Megan	2001	Seattle	5,000

CAB Relationship Set

Key Constraint over Ternary Relationships



Customer	Account	Branch
John	1001	Pullman
Megan	1002	LA
Tim	1003	LA
Jill	1003	Pullman

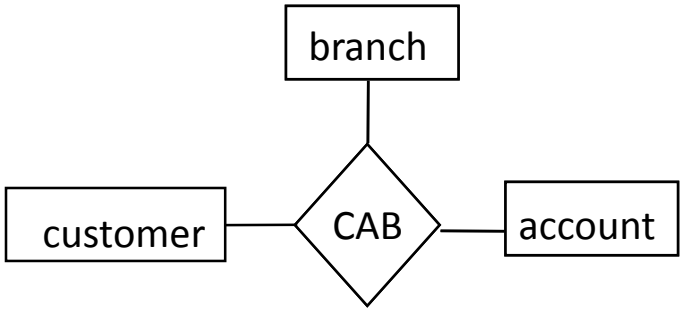
Legal

Customer	Account	Branch
John	1001	Pullman
Megan	1002	Seattle
Megan	1003	Seattle

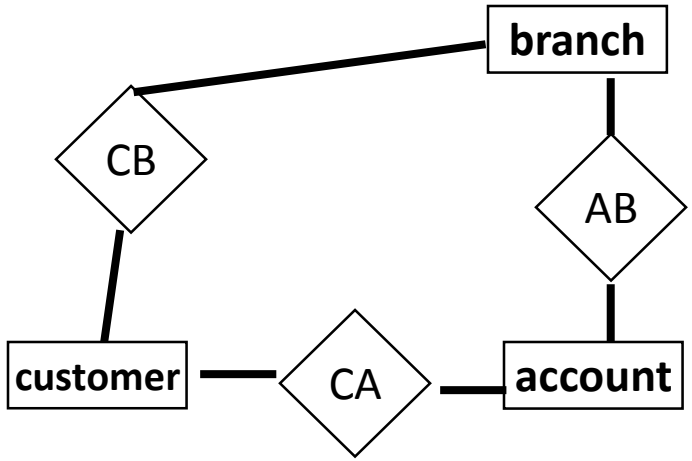
Illegal: Megan is associated with 2 accounts

- Interpretation:
  - Each customer is associated to a single account and a single branch

Representing Ternary Relationship Using Binary Relationships



Customer	Account	branch
C1	A1	B1
C1	A2	B1
C1	A1	B2
C2	A1	B1
C2	A2	B2



Customer	Account
C1	A1
C1	A2
C2	A1
C2	A2

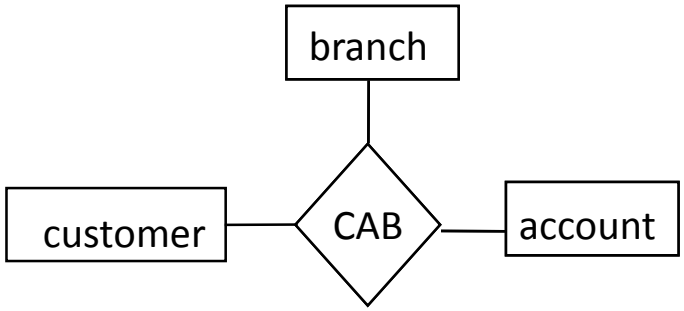
Customer	branch
C1	B1
C1	B2
C2	B1
C2	B2

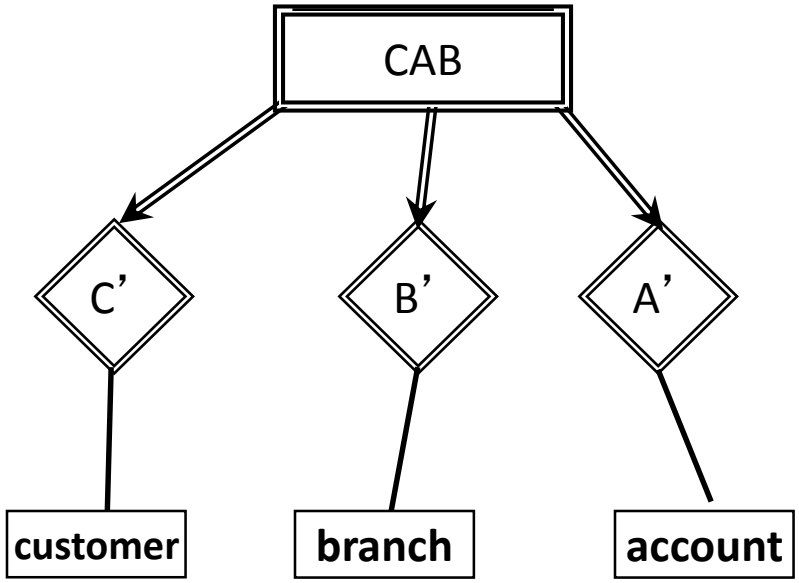
Account	branch
A1	B1
A1	B2
A2	B1
A2	B2

First Attempt: The relations CA, AB, and CB do not fully capture the information represented by CAB

- A multiway relationship can be converted to a collection of binary, one-to-many relationships.
- First Attempt: Can the CAB relationship set be represented using the ER diagram on the right?
  - These binary relationships AB, CA, and CB do not correctly capture the information represented by the ternary relationship.

Representing Ternary Relationship Using Binary Relationships

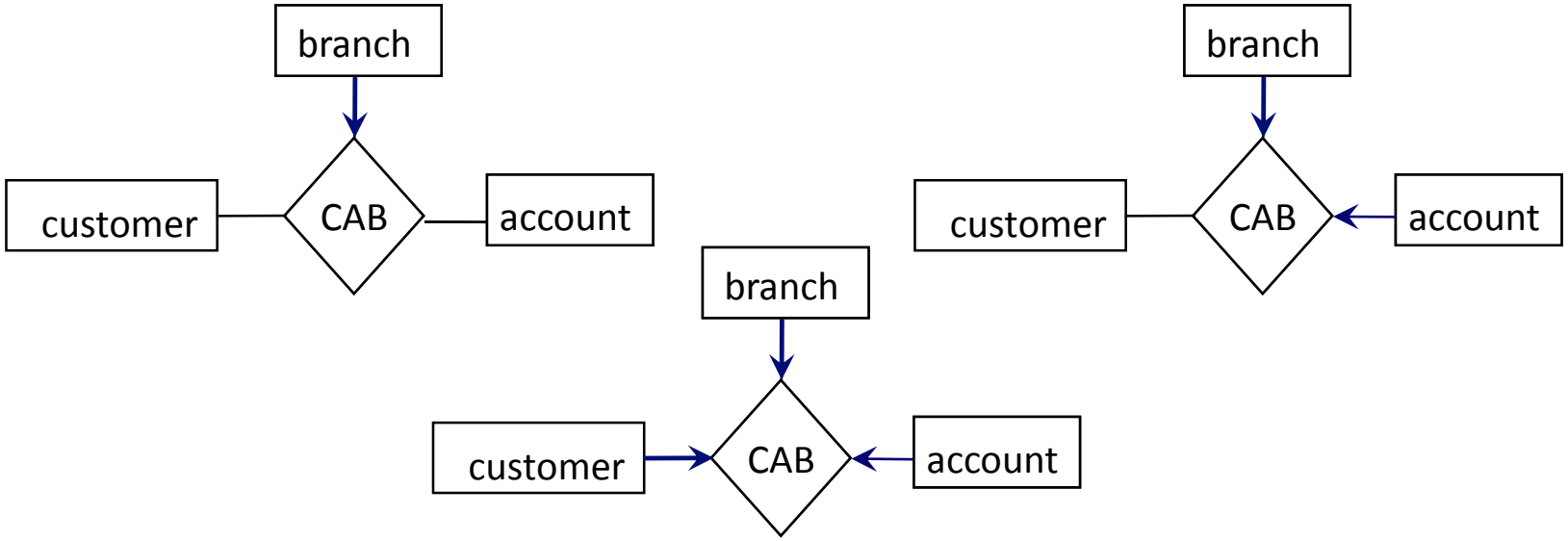




Second Attempt:

- Introduce a new entity set whose entities we may think of as the tuples of the multiway relationship
  - The CAB relationship is represented as a weak entity set that depends upon the customer, branch and account entity sets.
  - This schema using binary relationship fully captures the ternary relationship.

Representing Ternary Relationship Using Binary Relationships

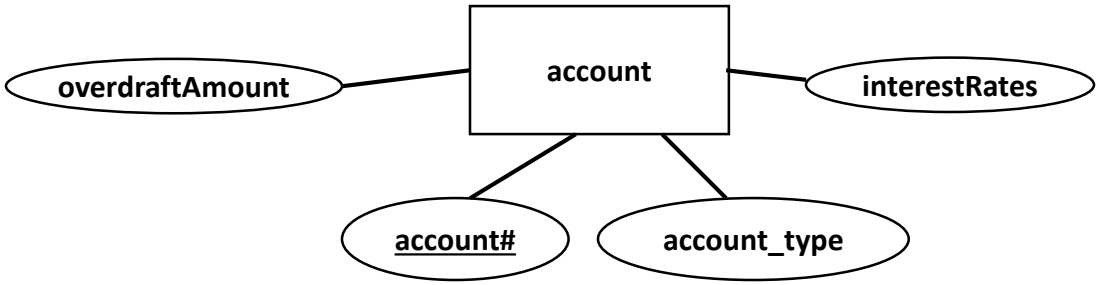


- Previous mapping technique works for many-many-many relationship.
- How to convert the many-many-1, many-1-1, 1-1-1 ternary relationships into binary relationships?
  - In general, it is always possible to convert any ternary (or multiway relationship) into a collection of binary relationships without losing information!!
  - However, the conversions can be quite complex and resulting unnatural schemas

Limitations of the Basic ER Model Studied So Far

How to represent different account types in the ER model:

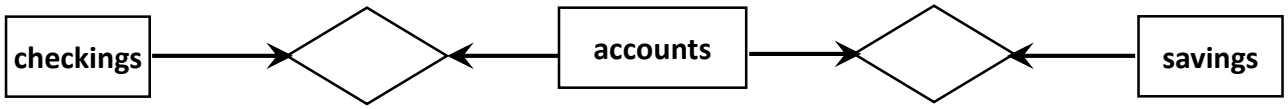
- **Possible Approach:**
  - Associate an attribute -- account-type with the accounts entity set
  - **Problems**:
    - different attributes may be associated with the account depending on its type
      - checking: overdraft amount
      - savings: interest rate
    - depending upon its type, savings and checking accounts may participate in different relationships.



Limitations of the Basic ER Model Studied So Far

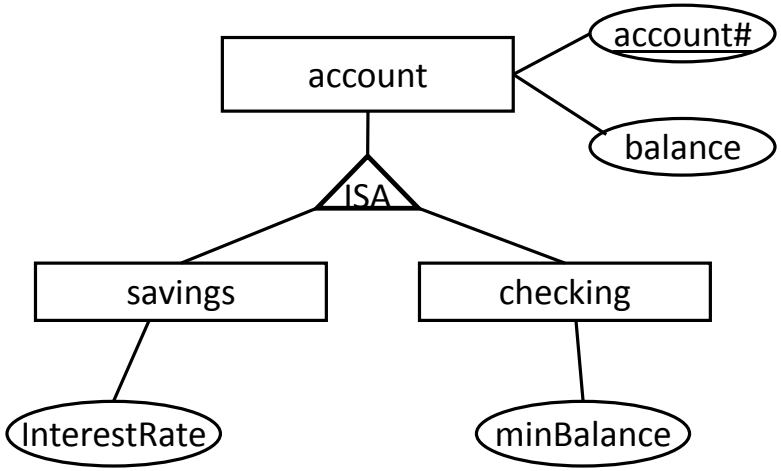
How to represent this in the ER model:

- **Approach 2:**
  - entity sets: checking, savings, and accounts.
  - relationships: 1-1 between checking and accounts, and 1-1 between savings and accounts
  - **Problems**:
    - *Not intuitive:* checking and savings are represented as entities different from accounts, even though they are accounts
    - *Redundancy of information:* info about accounts represented both in checking / savings as well as account entity set
    - *Potential Errors*: Same account could be erroneously associated with both checking as well as savings.



Extended ER Features

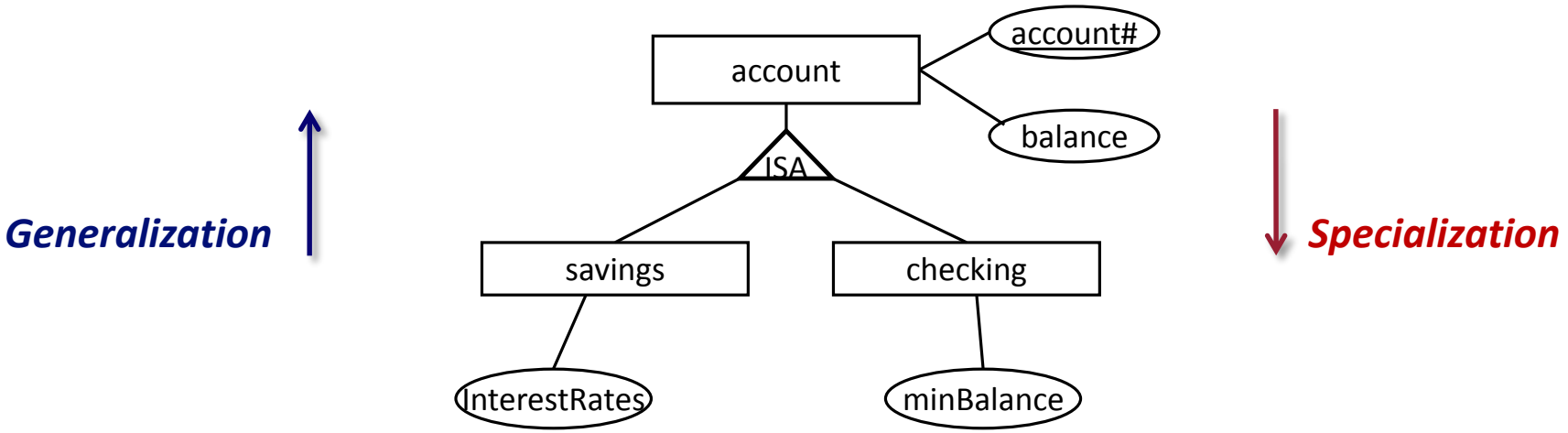
Subclass/Superclass Relationships



- "*savings*" and "*checking*" are **subclasses** of the account entity set.
- *"account"* is a **superclass** of *savings* and *checking* entity sets
- An entity in a subclass has to belong to superclass as well
  - every savings account is also an account
  - every checking account is also an account
- **Attribute and Relationship Inheritance**: As in OO languages, subclasses inherit all the attributes of the superclass. Similarly, subclasses inherit all relationships in which the superclass participates
- The **key of** entity set corresponding to the subclass is the same as the key for the superclass
- ISA relationship is one-to-one, although we don't draw two arrows on the two sides.

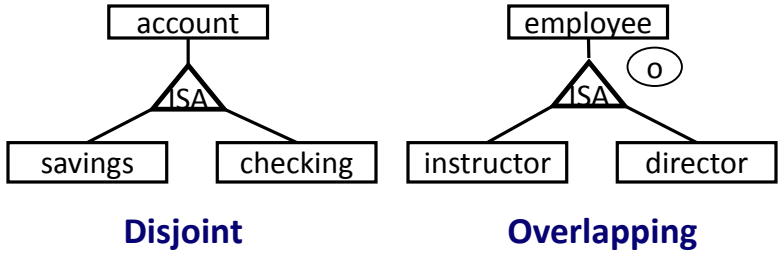
Specialization and Generalization

- ***Specialization***: process of classifying a class of objects into more specialized subclasses
  - E.g., during design, we begin with an account entity set. We then specialize the account set into different types of accounts.
- ***Generalization***: Reverse of specialization -- it is a process of synthesis of two or more (lower level) entity sets to produce a higher-level entity set.
  - E.g., during design, we have identified a car, a sports utility vehicle, and a truck. We generalize these classes to create an automobile entity set.



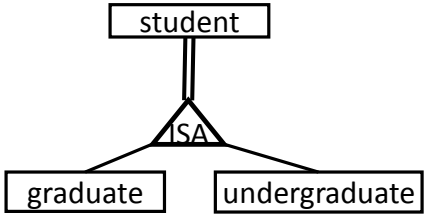
Types of Class/Subclass Relationships

- **Disjoint vs Overlapping**:
  - If the subclasses of the entity set do not overlap then it is disjoint
    - default
  - Else, overlapping
    - denoted by a 'o' next to ISA triangle



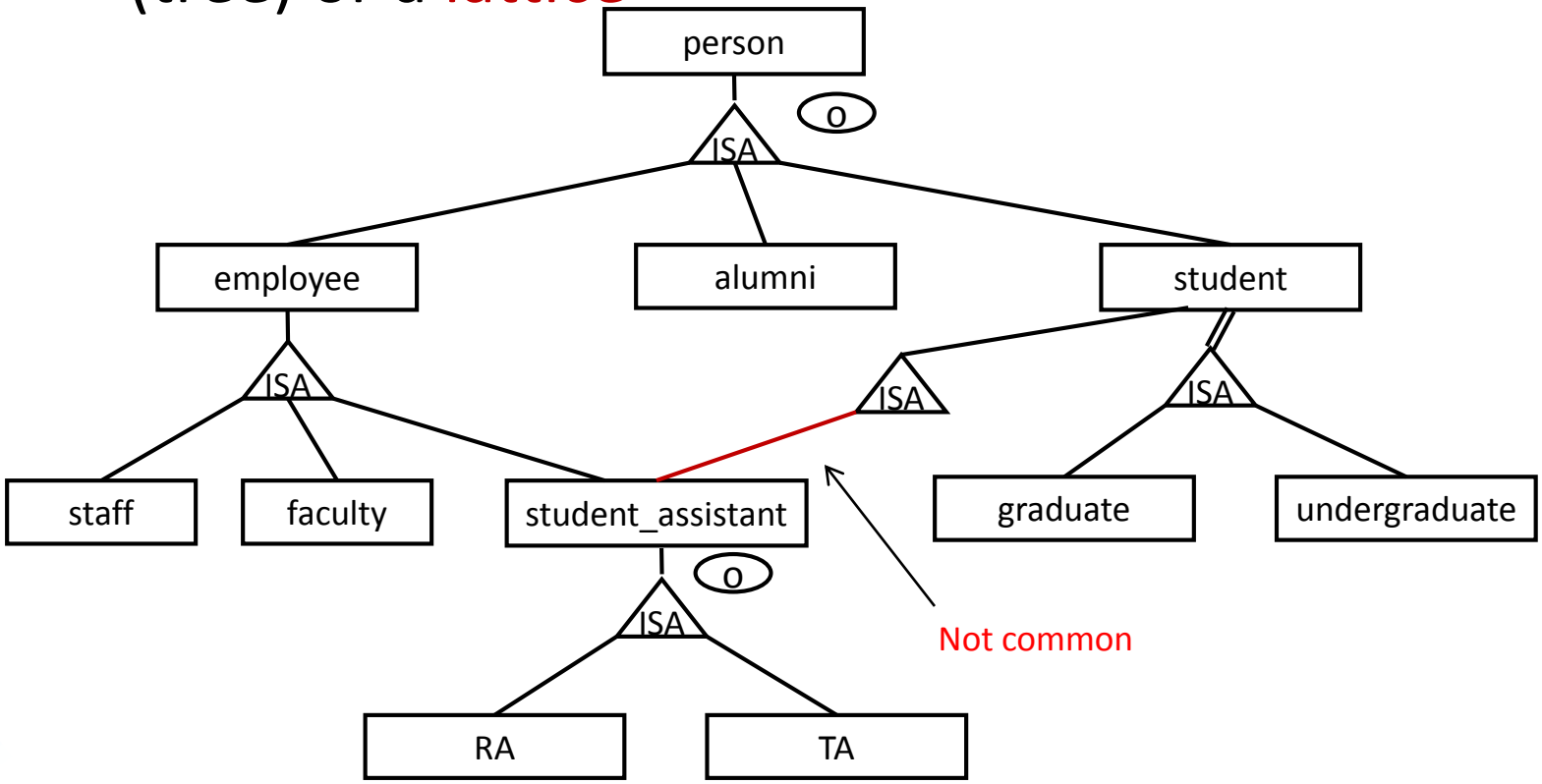
Disjoint **Overlapping**

- **Total vs Partial** ***(also called "covering")***:
  - If each entity in a superclass belongs to at least one of the subclasses, then total.
    - denoted by a double line from superclass to ISA triangle
  - Else, partial



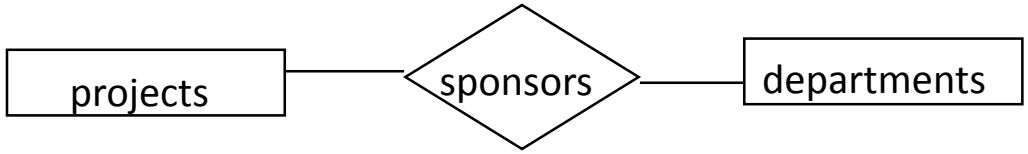
Superclass/Subclass Lattice

- Class/Subclass relationships might form a hierarchy (tree) or a lattice

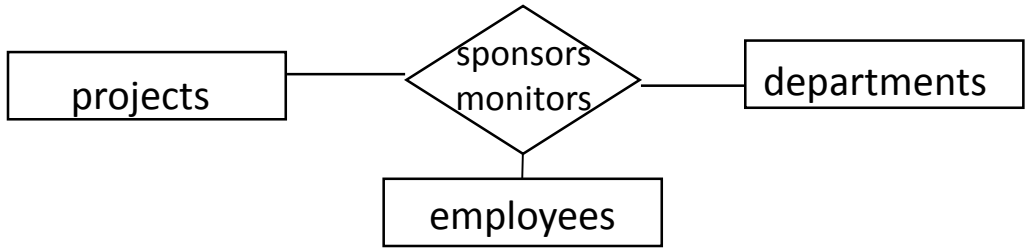


Back to Limitations of Basic ER Model

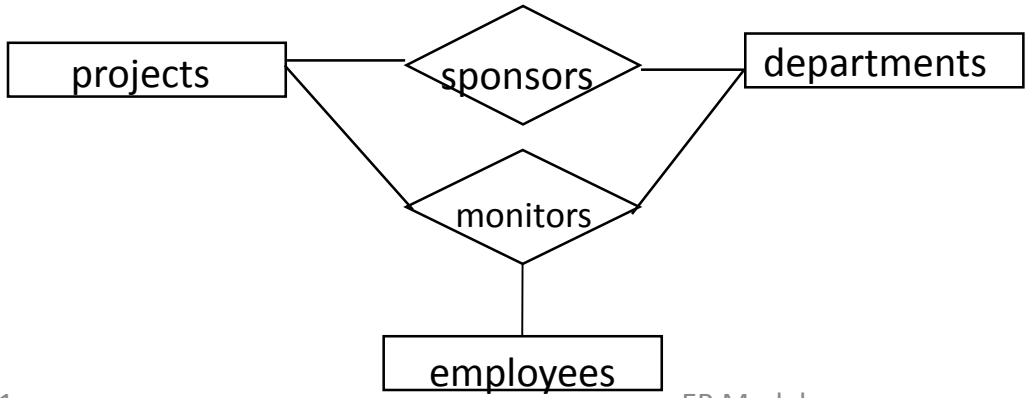
- Suppose projects are sponsored by one or more departments.



- A department that sponsors a project might assign an employee to monitor the sponsorship.



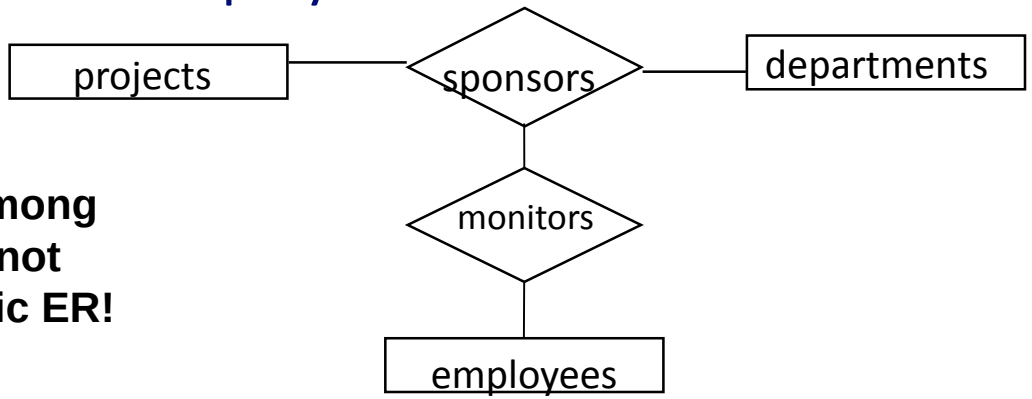
incorrect since it requires each sponsorship to have a monitoring employee



redundant relationships!

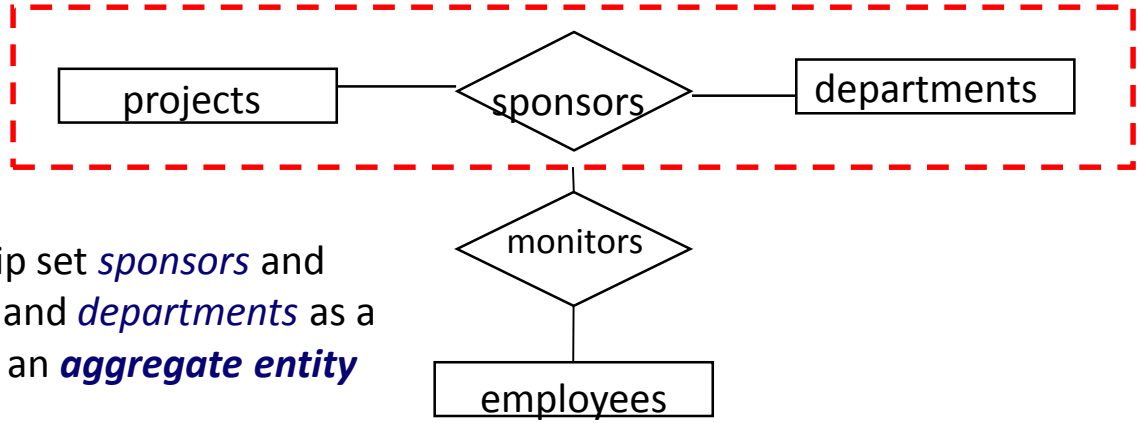
Limitations of Basic ER Model

- The "*monitors*" should be a relationship that associates "Sponsorship" to "employees"



relationships among relationships not permitted in basic ER!

Solution in extended ER : Aggregation

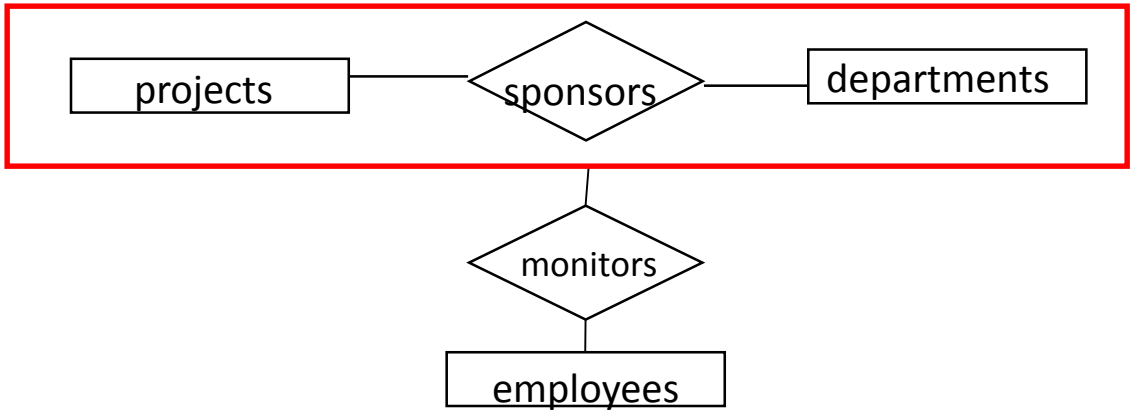


- Treat the relationship set *sponsors* and the entity sets *projects* and *departments* as a higher level entity set-- an ***aggregate entity set***

- Permit relationships between aggregate entity sets and other entity sets

Aggregation

- Without introducing redundancy, the following diagram represents:
  - A project is sponsored by a particular department
  - A sponsorship (project, department combination) may have an associated monitoring employee.



Review of ER Model

- Basic Model:
  - Entities : strong, weak
  - Attributes associated with entity sets and relationships
  - Relationships: binary, ternary, ...
  - Role of entity sets in a relationship
  - Constraints on entity set: domain constraints, key constraint
  - Constraint on relationships: key constraints, participation --total vs partial
- Extended Model:
  - Notion of superclass and subclass
  - Superclass/subclass relationships: disjoint vs overlapping, total vs partial
  - Notion of aggregation

Schema Design Issues

- **Observation:** There may be multiple ER schemas describing the same target database.
- Decisions that need to be made:
  - whether to use an attribute or entity set to represent an object
  - whether to model a concept as a relationship or an entity set
  - whether to use ternary relationship or a set of binary ones
  - whether to use a strong entity set of a weak entity set
  - whether using generalization/specializations is appropriate
  - whether using aggregates is appropriate
- Unfortunately, there are no straightforward answers to these questions
- No two design teams will come up with the same design.
- However, there are some simple design principles that should be followed during ER design.

Entity vs. Attribute

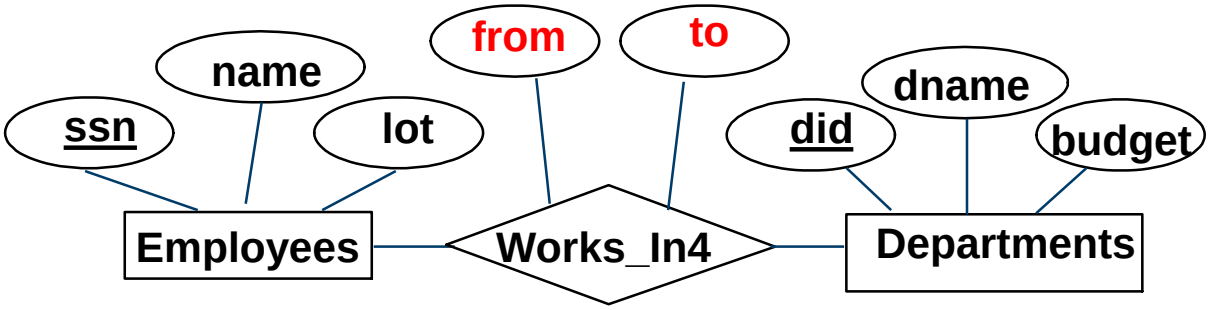
Example-1

- Should "*dependent*" be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of dependent information, and the semantics of the data:
  - If we have several dependents per employee, *"dependent"* must be an entity (since attributes cannot be set-valued).
  - If "*dependent*" has it's own attributes, "*dependent*" must be modeled as an entity (since attribute values are atomic).

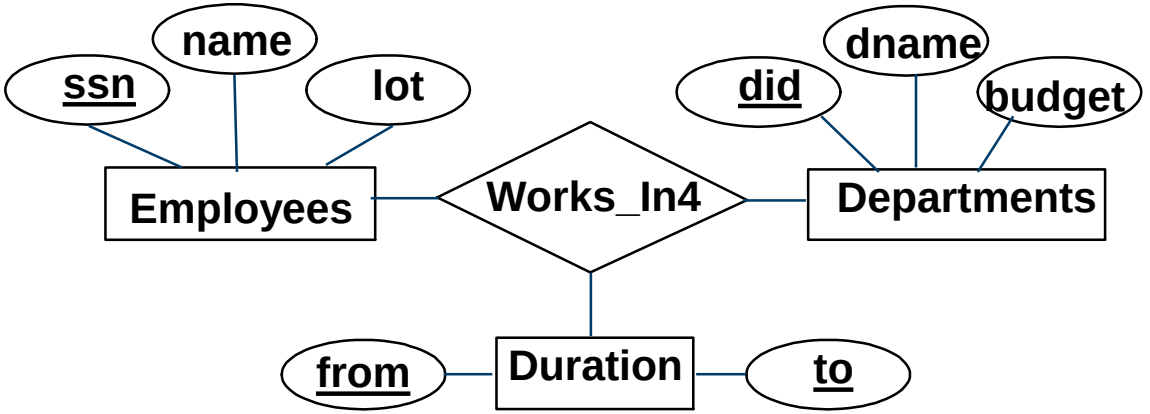
Entity vs. Attribute

Example-2:

- **Works_In4** does not allow an employee to work in a department for more than one period.



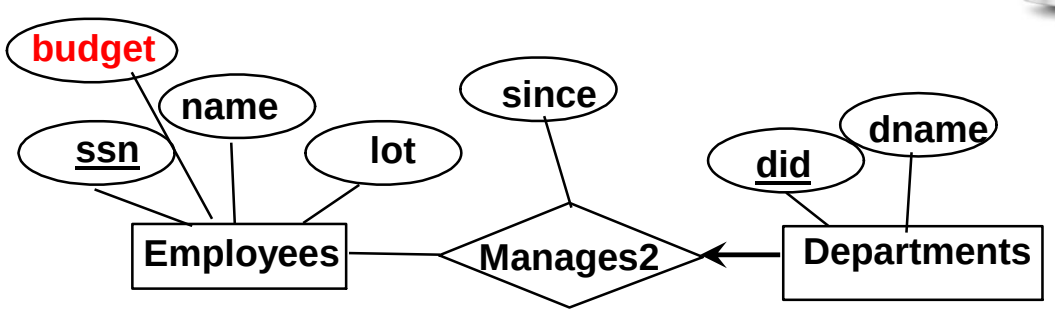
- We want to record *several values of the descriptive attributes for each instance of this relationship.* Accomplished by introducing new entity set, "**Duration**".

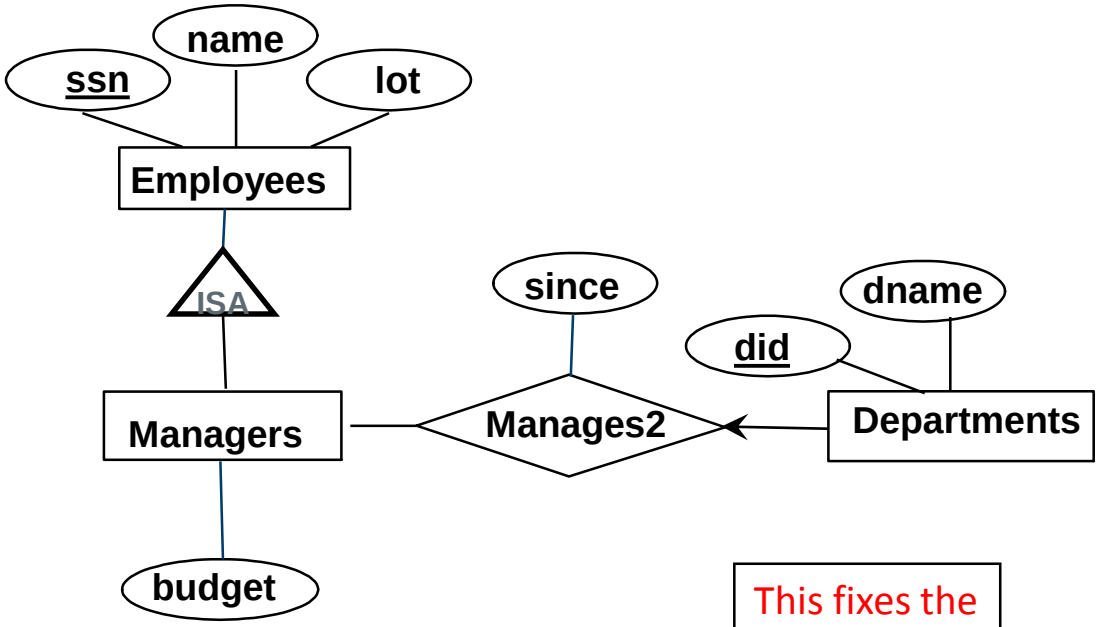


Subclass Relationship

Example-3:

- Assume budgets are associated with managers, not departments.
- 1st model: Not all employees are managers so most employees won't have budgets.



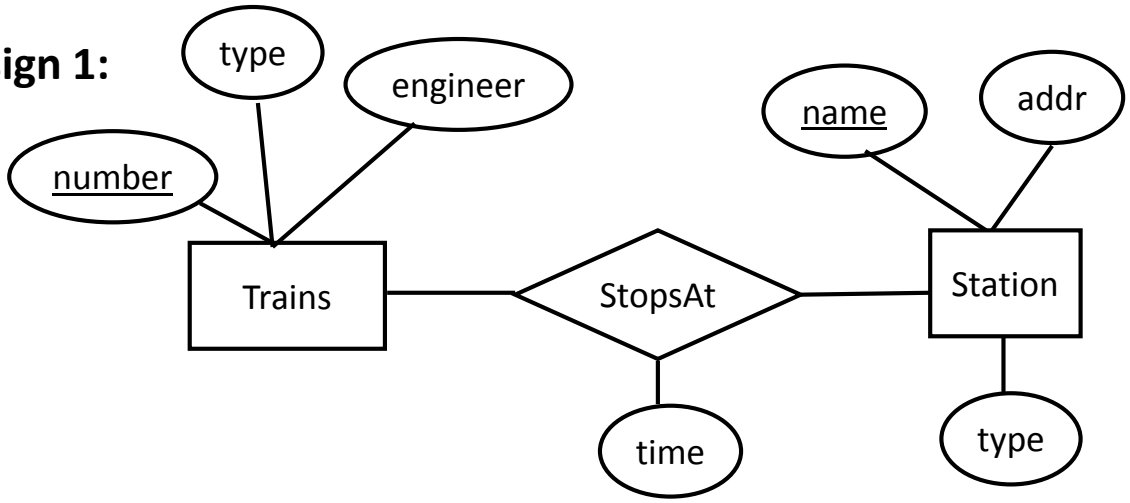


This fixes the problem!

ER Design Example-1:

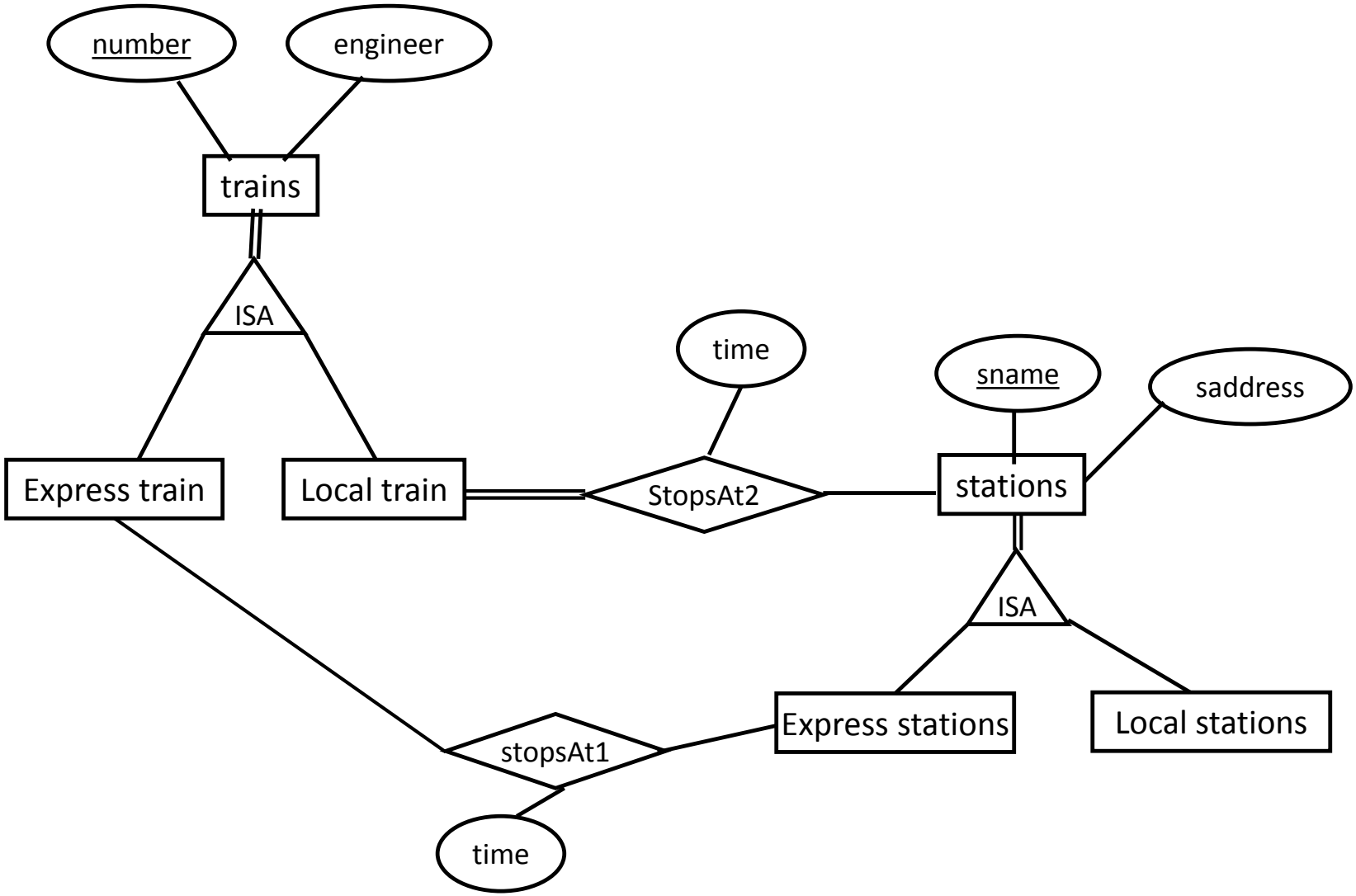
- We wish to design a database consistent with the following facts.
  1. A train has a unique number and has an engineer.
  2. Trains are either local trains or express trains, but never both.
  3. Stations are either express stops or local stops, but never both.
  4. A station has a name (assumed unique) and an address.
  5. All local trains stop at all stations.
  6. For each train and each station the train stops at, there is a time.
  7. Express trains stop only at express stations.

Design 1:



Does not capture the constraints that express trains only stop only at express stations and local trains stop at all local stations

Design 2: Better Design

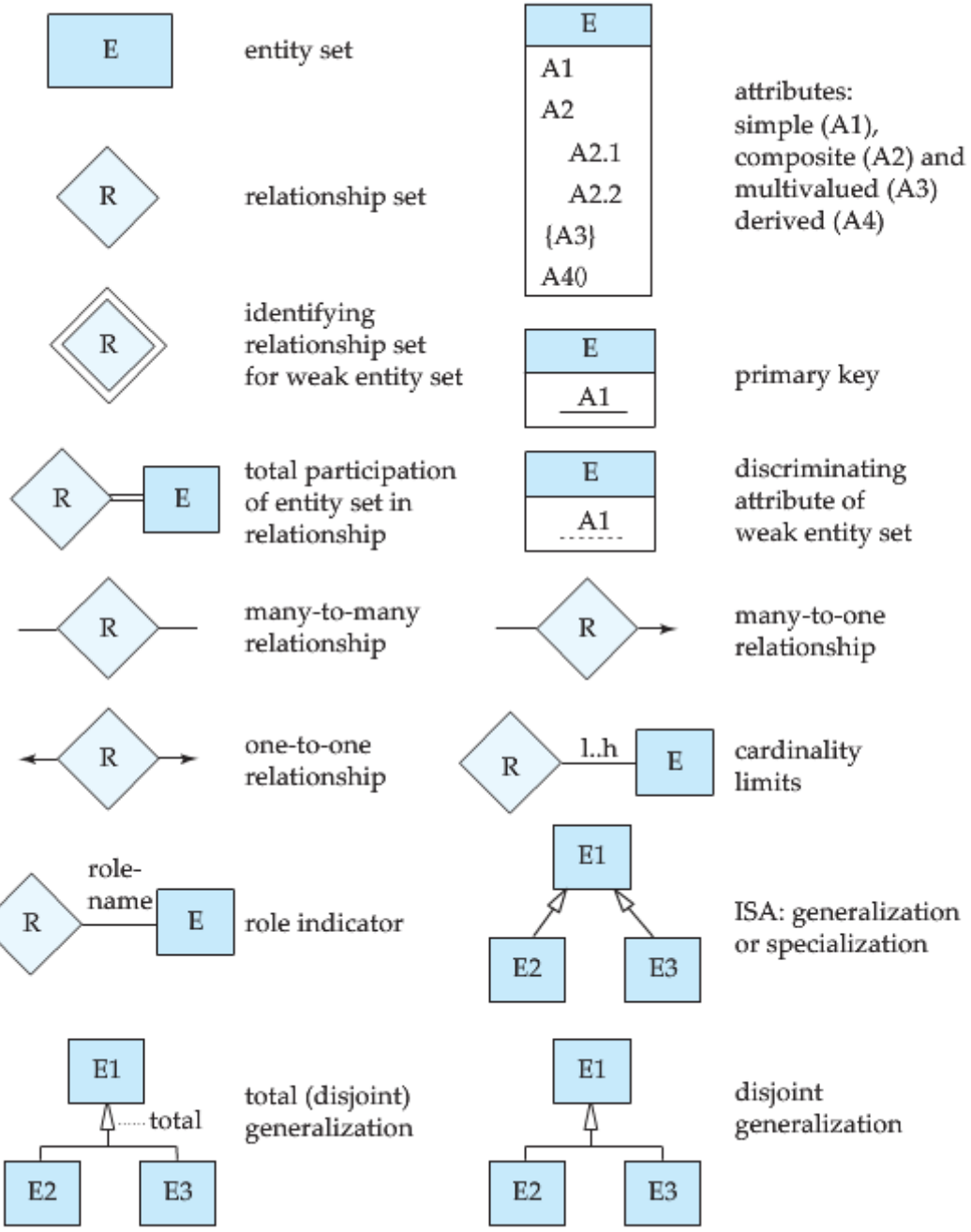


ER Design Example-2:

Notown Records would like to store information about musicians who perform on its albums (as well as other company data) in a database

1. Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address. No phone has more than one address, but musicians that share an address might have different phones.
2. Each instrument used in songs recorded at Notown has a name (e.g., guitar, synthesizer, flute), a musical key (e.g., C, B-flat, E-flat), and a instrument number which is unique for each instrument name.
3. Each album recorded on the Notown label has a unique identification number, a title, and a copyright date.
4. Each song recorded at Notown has a title and an author.
5. Each musician may play several instruments, and a given instrument may be played by several musicians.
6. Each album has a number of songs on it, but no song may appear on more than one album. The song titles in an album is assumed to be unique.
7. Each song is performed by one or more musicians, and a musician may perform a number of songs.
8. Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

Textbook notation: Symbols used in ER diagrams



Alternative ER notations