

Conceptual Design using the Entity Relationship (ER) Model

Juliana Freire

Some slides adapted from L. Delcambre, R. Ramakrishnan, and
Silberschatz, Korth and Sudarshan

Overview of Database Design

- **Conceptual design:**

- Analyze ‘problem’, define which information the database must hold and the relationships among the components of the information
- What are the **entities** and **relationships** and **attributes** in the enterprise?
- Use a *language* to specify design -- *ER Model is used for this*

Understand what users want from database

- **Schema Refinement:**

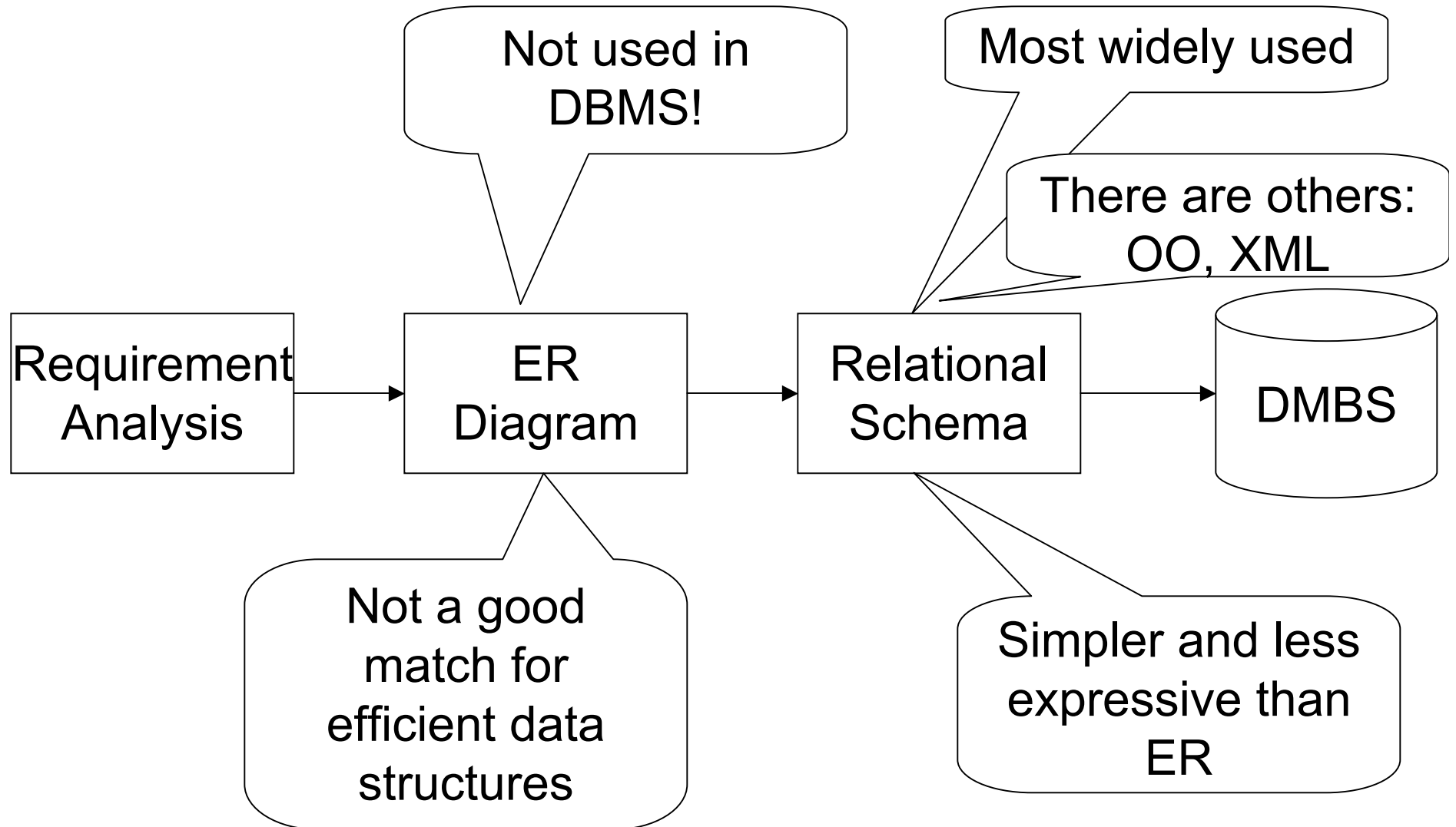
- ER diagram is converted into a relational schema
- Check relational schema for redundancies and relational anomalies – *Normalization*
- Input schema to DBMS – database comes to existence!

Simple yet precise description

- **Physical Database Design and Tuning:**

- Consider typical workloads and further refine the database design.

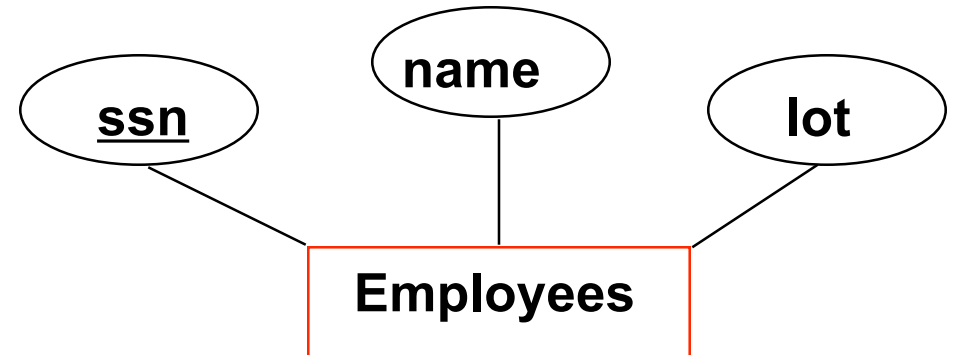
Overview of Database Design



Conceptual Design

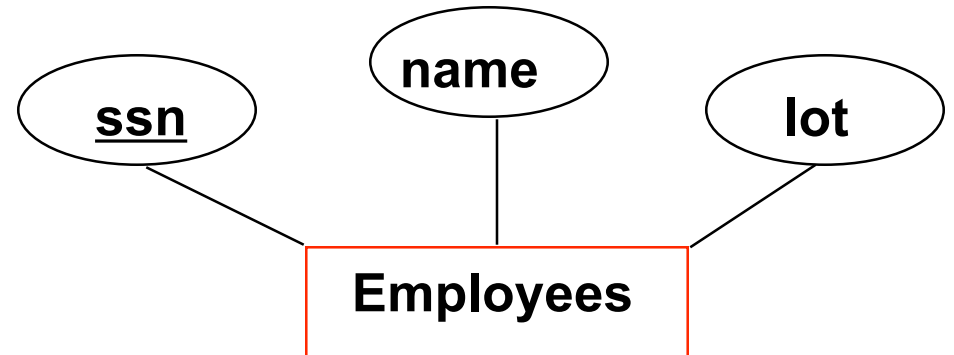
- *ER Model is used at this stage*
 - What are the *entities* and *relationships* among these entities in the enterprise?
 - What information about these entities and relationships should we store in the database?
 - What are the *integrity constraints* or *business rules* that hold?
 - A database `schema'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

ER Model - Entities



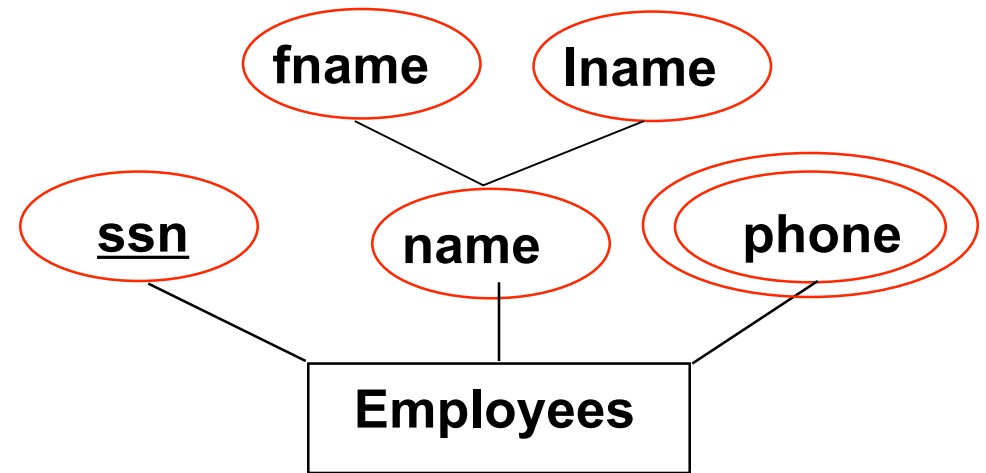
- **Entity:** Real-world object distinguishable from other objects
 - E.g., specific person, company, event
 - described (in DB) using a set of *attributes*
- **Entity Set:** A collection of similar entities that share the same properties. E.g., all employees
 - All entities in an entity set have the same set of attributes
 - E.g., set of all persons, companies, events
 - Each entity set has a *key*

Entities vs. Objects



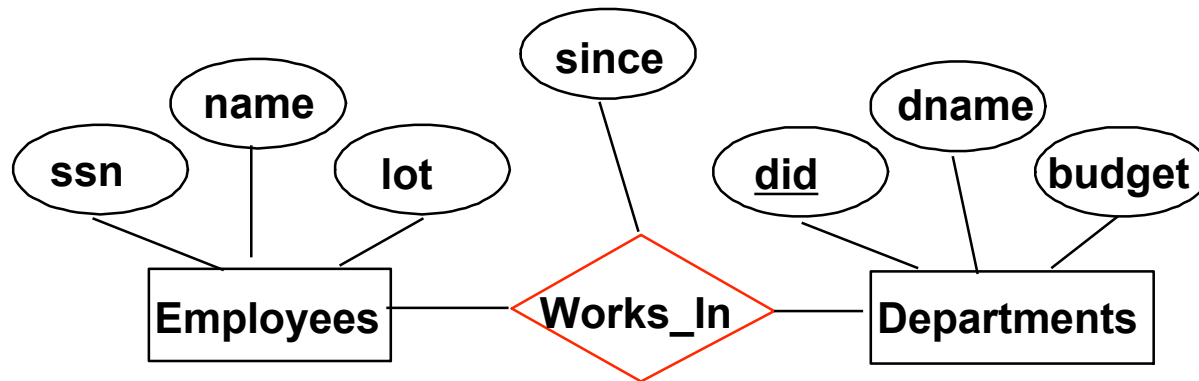
- **Entity** is similar to **object** in the sense of object-oriented programming
- **Entity set** similar to a **class of objects**
- Entities are static, no methods!

ER Model - Attributes



- Entity sets have associated **attributes** – properties of the entities in that set
 - E.g., employees have ssn, name, and multiple phone numbers
- *Domain* – the set of permitted values for each attribute, e.g., $18 < \text{age} < 65$
- Attribute types:
 - *Simple* attributes: e.g., ssn, fname, lname
 - *Composite* attributes: e.g., name
 - *Single-valued* (e.g., ssn) and *multi-valued* attributes, e.g., phone
 - *Derived* attributes, e.g., age, given date of birth

ER Model - Relationships



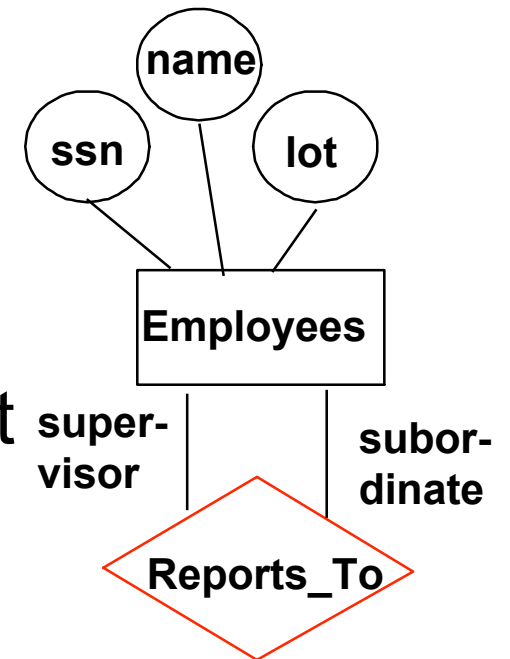
- **Relationship:** connection among two or more entity sets
 - E.g., the employee John works in Pharmacy department
- To create an instance of a relationship, we must indicate which employee and which department we want to have connected (for this relationship).
- We need the **key** value for an employee and the **key** value for a department, stored together, to represent the relationship

Keys

- A *super key* of an entity set is a set of one or more attributes whose values uniquely determine each entity.
 - SSN, name
- A *candidate key* of an entity set is a minimal super key
 - SSN is candidate key of *Employees*
 - *DId* is candidate key of *Departments*
- Although several candidate keys may exist, one of the candidate keys is selected to be the *primary key*

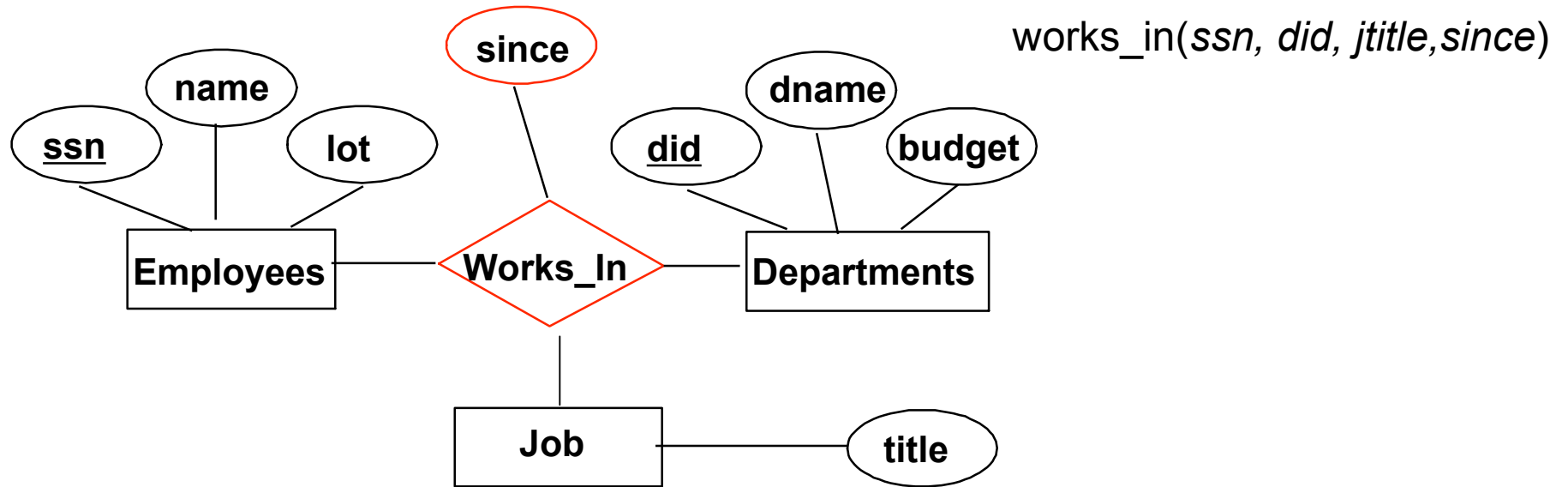
Roles in Relationships

- Same entity set can participate in different relationship sets, or in different “roles” in same set
- Draw as many lines from the relationship set to the entity set as the entity set appears in the relationship



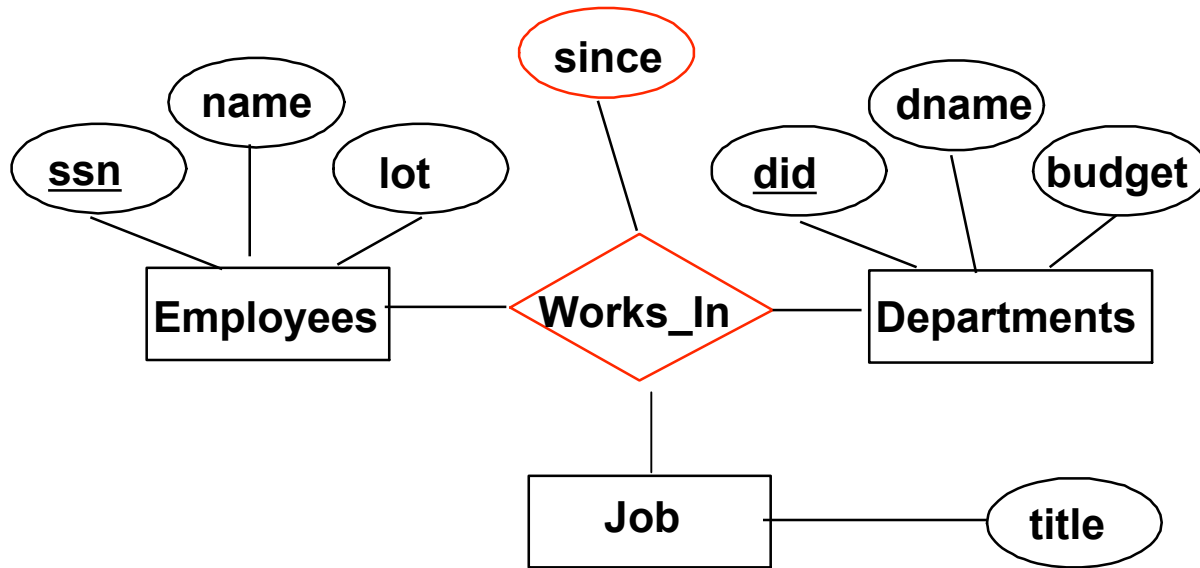
`reports_to(supervisor_ssn, subordinate_ssn)`

Multi-Way Relationships



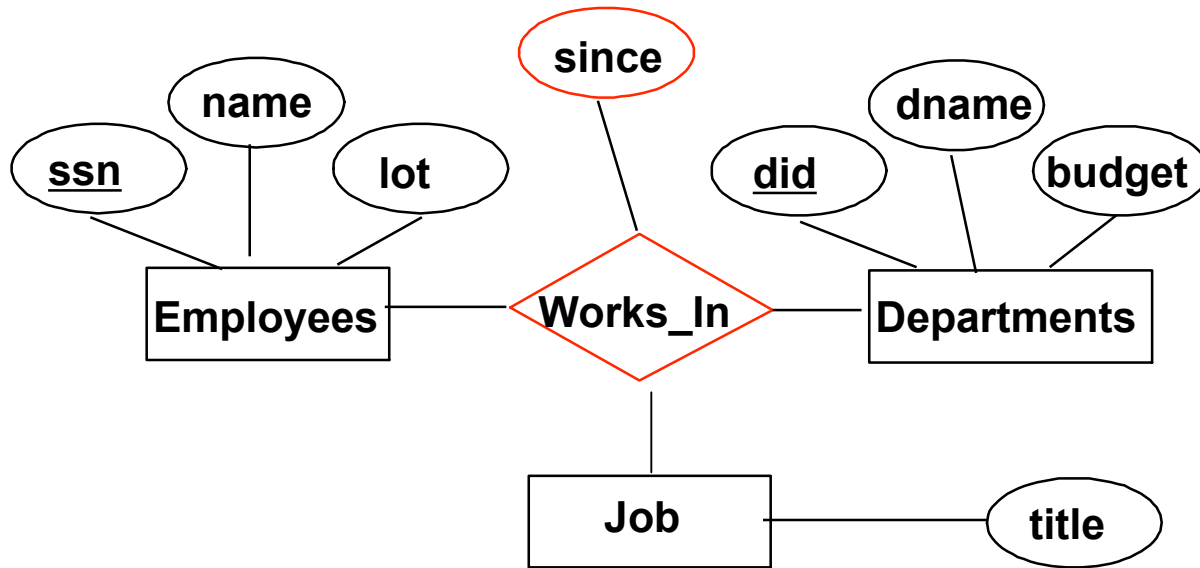
- Relationship **degree**: the number of entities involved
 - **binary** is the most common
 - E.g., Works_in is a **ternary** relationship: employees of the company may have jobs at multiple depts, with different jobs at different depts

Attributes on Relationships



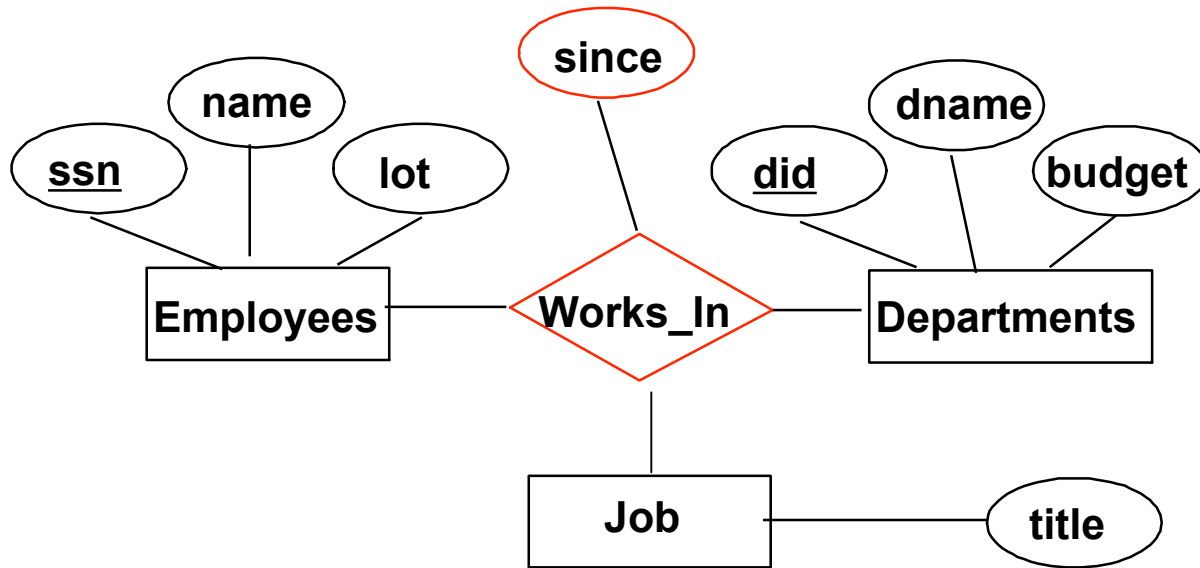
- A relationship set may have an attribute
- E.g., “since” records the date employee started a given job at a particular department
- *Can we instead place “since” in the Job entity?*
- *Or place “since” in the Employee entity?*

Challenge Questions



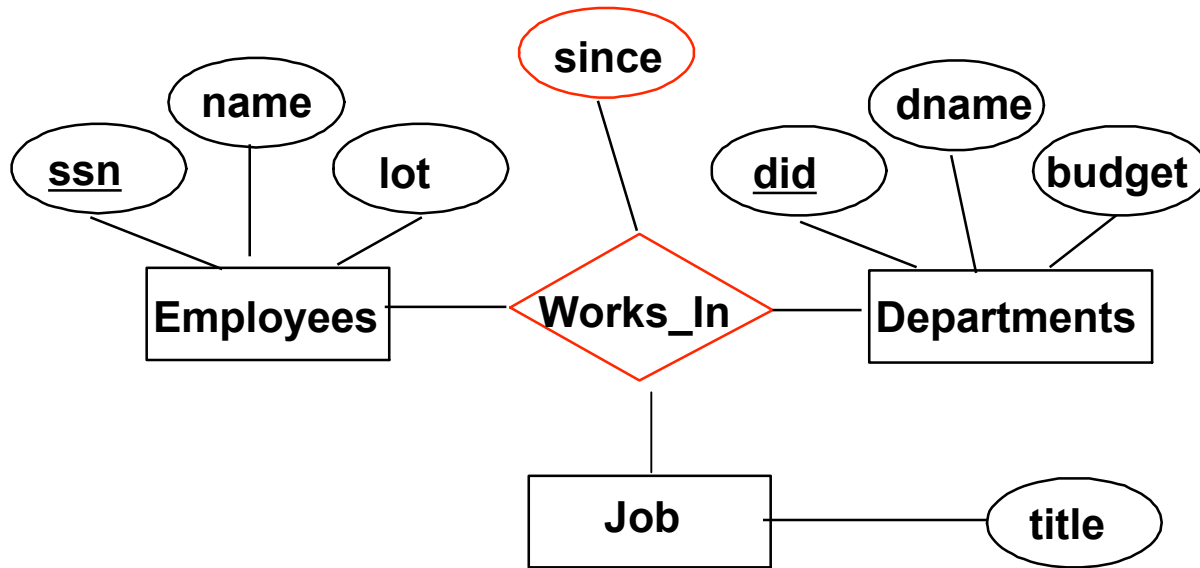
- *Can we instead place “since” in the Job entity?*
- *Or place “since” in the Employee entity?*

Challenge Questions



- *Can we instead place “since” in the Job entity?*
Job(mechanic, 14AUG2003)
Job(programmer, 4MAR1978)
- *Or place “since” in the Employee entity?*

Challenge Questions

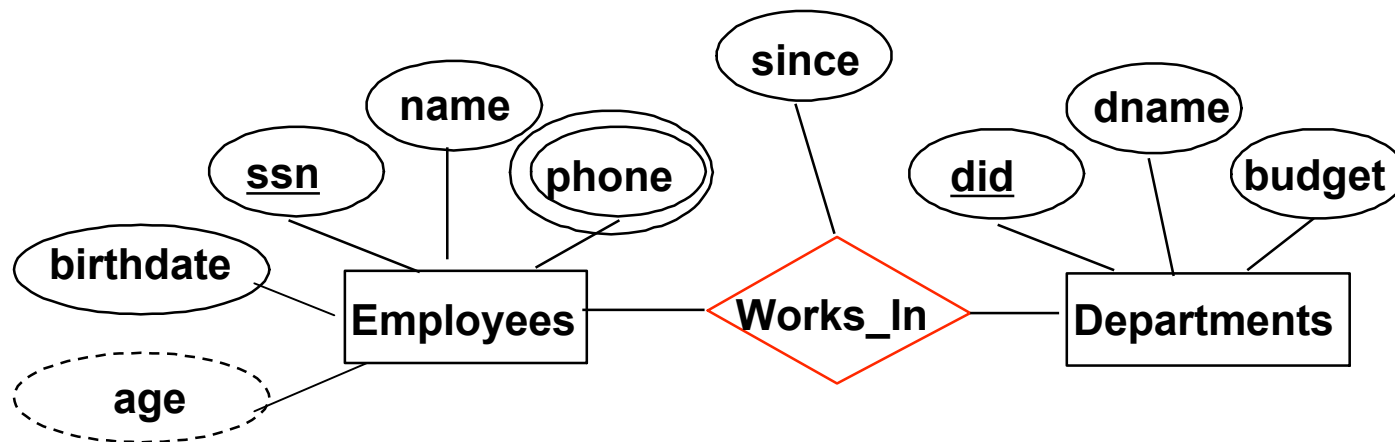


- *Can we instead place “since” in the Job entity?*
Job(mechanic, 14AUG2003)
Job(programmer, 4MAR1978)
- *Or place “since” in the Employee entity?*
Employee(123456789, 'John Doe', 23, 14AUG2003)

Relationships – more formally...

- **Relationship Set:** Collection of similar relationships
 - An n-ary relationship set R relates n entity sets $E_1 \dots E_n$:
 $\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$, (e_1, e_2, \dots, e_n) is a **relationship**
 - $(\text{John}, \text{Pharmacy}) \in \text{Works_in}$
 - $\text{Works_in}(\text{John}, \text{Pharmacy})$

E-R Diagrams



- **Rectangles** represent entity sets.
- **Diamonds** represent relationship sets.
- **Lines** link attributes to entity sets and entity sets to relationship sets.
- **Ellipses** represent attributes
 - **Double ellipses** represent multivalued attributes.
 - **Dashed ellipses** denote derived attributes.
- **Underline** indicates primary key attributes (will study later)

Instances of an E-R Diagram

- E-R diagrams describe the *schema* of a database
- A *database instance* contains data that follows the prescribed structure
- We have no data during the conceptual design, but *imagining the data exists* helps us to think about the design

Department

Did	Dname	Budget
D100	Pharmacy	120,000.00
D101	Cardiology	540,000.00
D102	HR	20,000.00

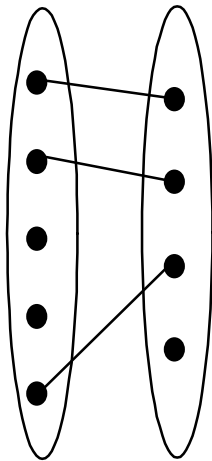
Works_in

Did	ssn	Since
D100	123-45-6789	1964
D100	222-33-4444	2005
D102	999-777-5555	1999

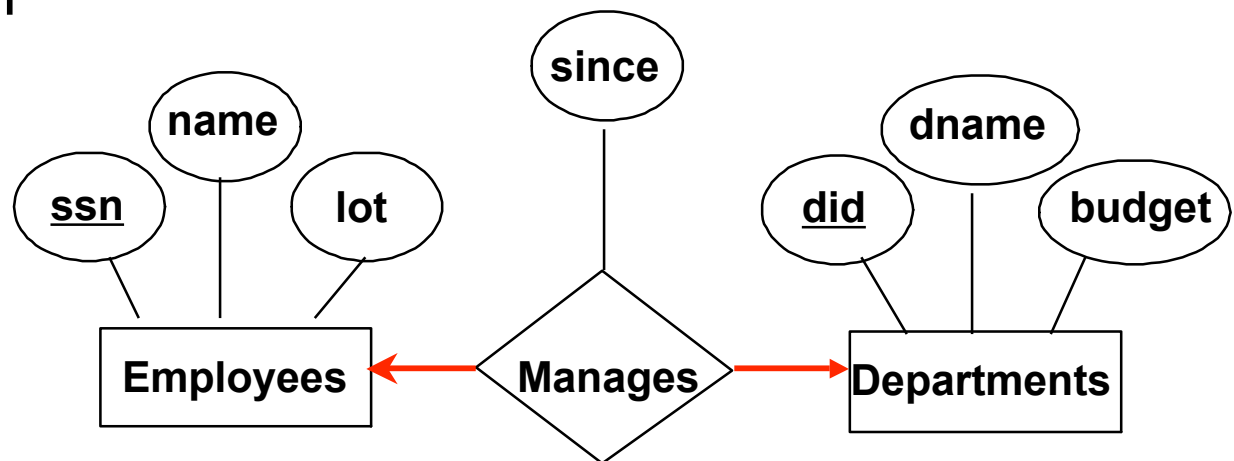
Multiplicity of Relationships

How many times must/may an entity instance participate?

- Each dept has *at most* one manager, and a manager can manage *at most* one department



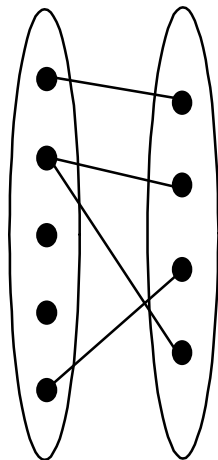
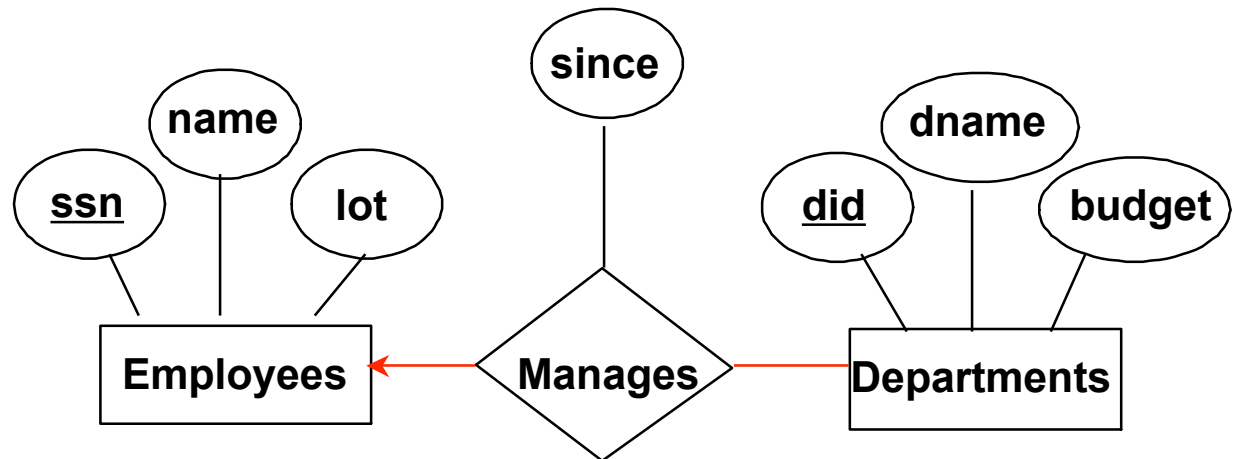
1-to-1



One-to-one relationship

Multiplicity of Relationships

- Each dept has *at most one* manager, but an employee can manage multiple departments



1-to Many

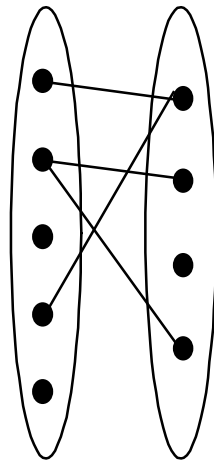
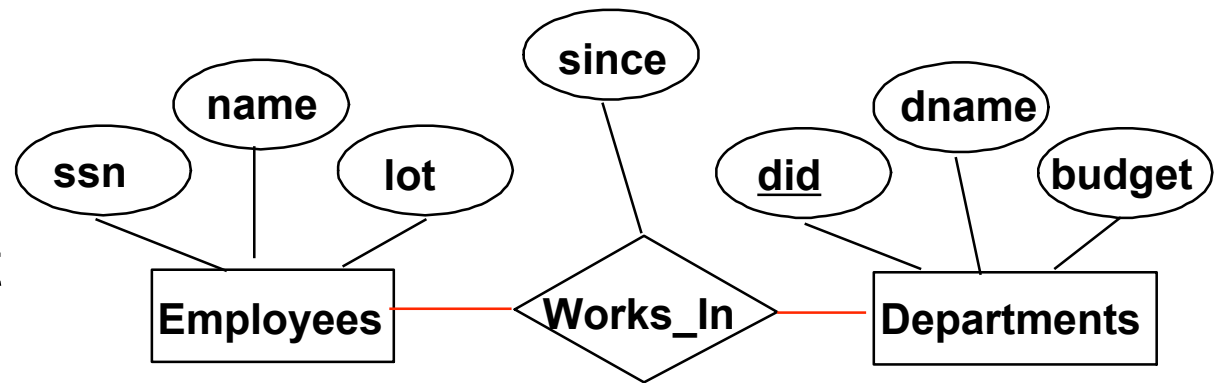
One-to-many relationship
(employees → departments)

Many-to-one relationship
(departments → employees)

Multiplicity of Relationships

How many times must/may an entity instance participate?

- An employee can **work in** many departments; a dept can have many employees.

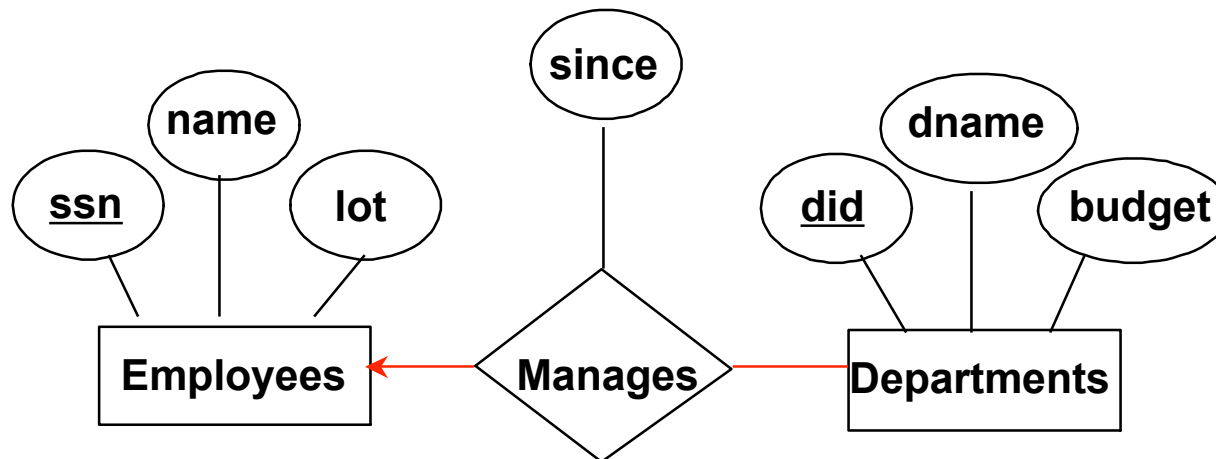


Many-to-Many

Many-to-many relationship

Semantics of the “arrow”

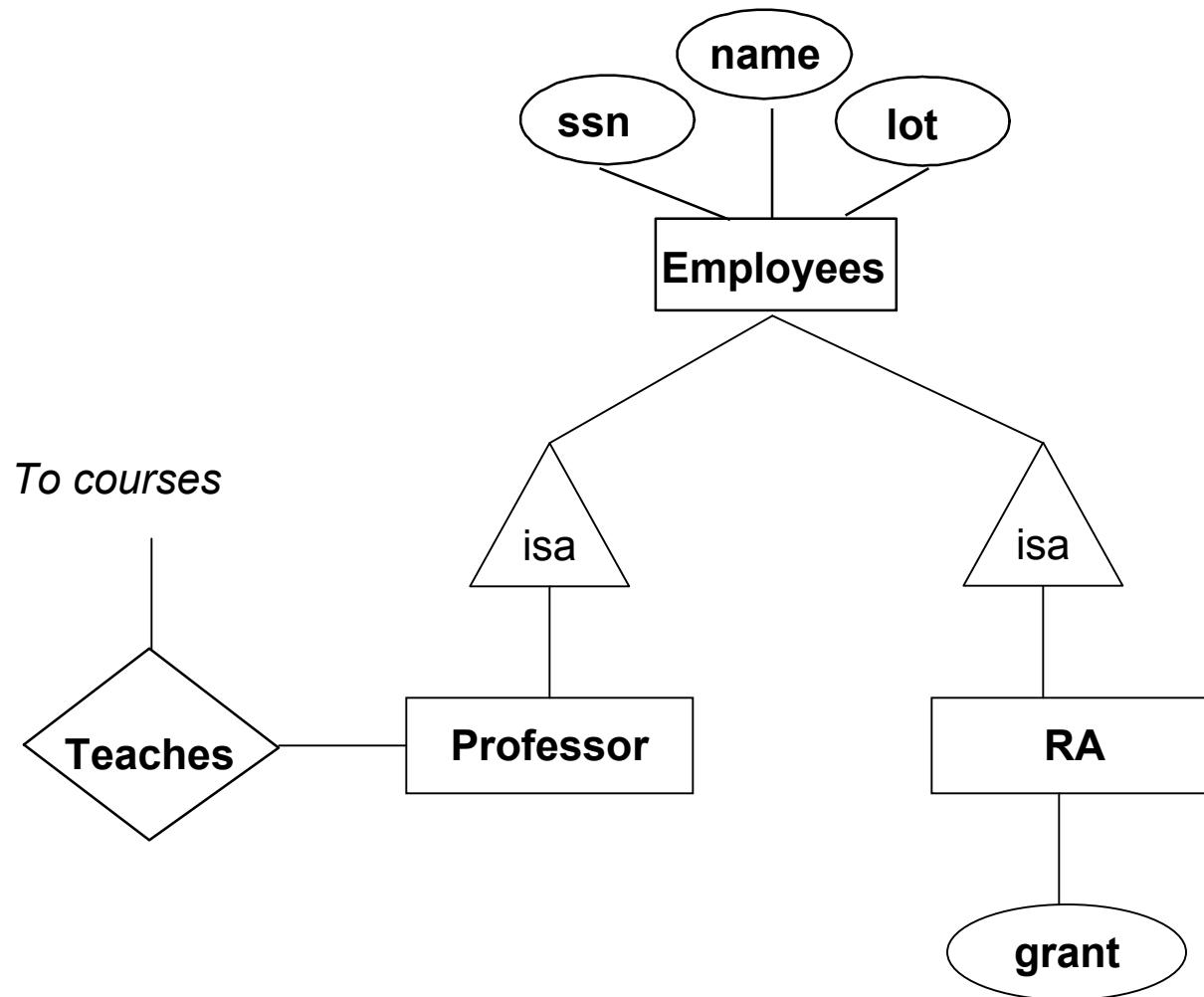
- Arrow means *at most one*
- It does not guarantee existence of an entity set pointed to
- E.g., there can be departments without a manager at a particular point in time



Specialization and ISA Relationship

- Model the case when certain entities have special properties not associated with all members of the entity set
 - Designate sub-groupings within an entity set that are distinctive from other entities in the set.
- Familiar notion to O-O programmers
 - Subclassing; subtyping
- More formally: entity subsets
 - *Undergrad* \subseteq *Student* in a university database
 - Cannot create a *Undergrad* who is not also a *Student*
 - However, there may be *Students* who are not *Undergrads*
- Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization: Example



Another Example of ISA

- Undergrad ISA Student; TA ISA student
- Undergrad and TA inherits all attributes of Student
- Can *Jones* be a *Undergrad* **and** a *TA*?

Another Example of ISA

- Undergrad ISA Student; TA ISA student
- Undergrad and TA inherits all attributes of Student
- Can *Jones* be a *Undergrad* **and** a *TA*?
 - *Overlap*: Yes, by default
- Can we have an *Student* who is not a *Undergrad* or *TA*?

Another Example of ISA

- Undergrad ISA Student; TA ISA student
- Undergrad and TA inherits all attributes of Student
- Can *Jones* be a *Undergrad* **and** a *TA*?
 - *Overlap*: Yes, by default
- Can we have an *Student* who is not a *Undergrad* or *TA*?
 - *Non-totality*: Yes, by default

Modeling Constraints

- *Additional information about aspects of application we are modeling*
- **Keys** attributes or set of attributes that uniquely identify an entity within its set
 - E.g., employee's ssn
- **Referential integrity constraints** are requirements that a values referred to by some object actually exists in the database: prevent *dangling* pointers
 - E.g., department *must* have manager
- **Domain constraints** require that the value of an attribute must be drawn from a specific set of values
 - E.g., $0 < \text{age} < 120$
- **General constraints** are arbitrary assertions required to hold in database
 - E.g., a department can have at most 30 employees

Constraints in the Database

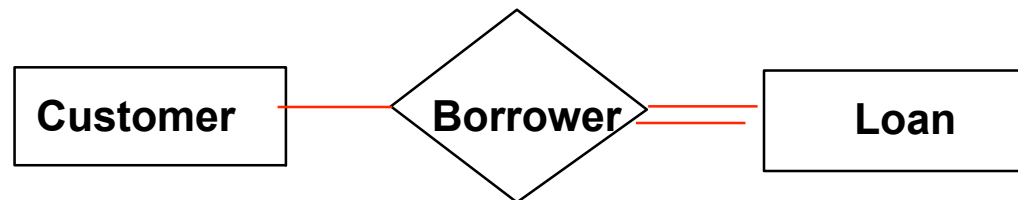
- Constraints are part of the *schema*
- Declared by the database designer along with the structural design
- Once a constraint is defined, **insertions or modifications to the database that violate the constraint are disallowed**
- We will learn more about constraints later!

Keys

- Every entity set must have a key
- A key can consist of more than one attribute
 - E.g., countryCode + areaCode + phoneNumber
- An entity set may have more than one possible key
 - E.g., ssn and employeeld
 - Customary to pick one key to be the *primary key*
- *Primary keys* are represented by underlining the corresponding attribute name(s)

Participation Constraints

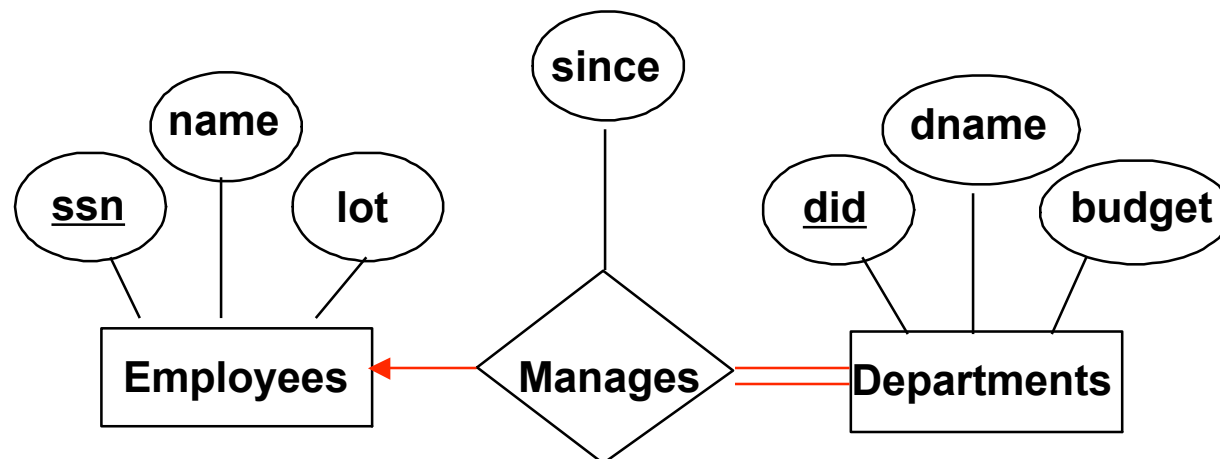
- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
 - participation of *loan* in *borrower* is total -- every loan must have a customer associated to it via borrower; a *loan* cannot exist without being in at least one *borrower* relationship
- **Partial participation**: some entities may not participate in any relationship in the relationship set
 - participation of *customer* in *borrower* is partial, some customers only have a checking account



Challenge Question

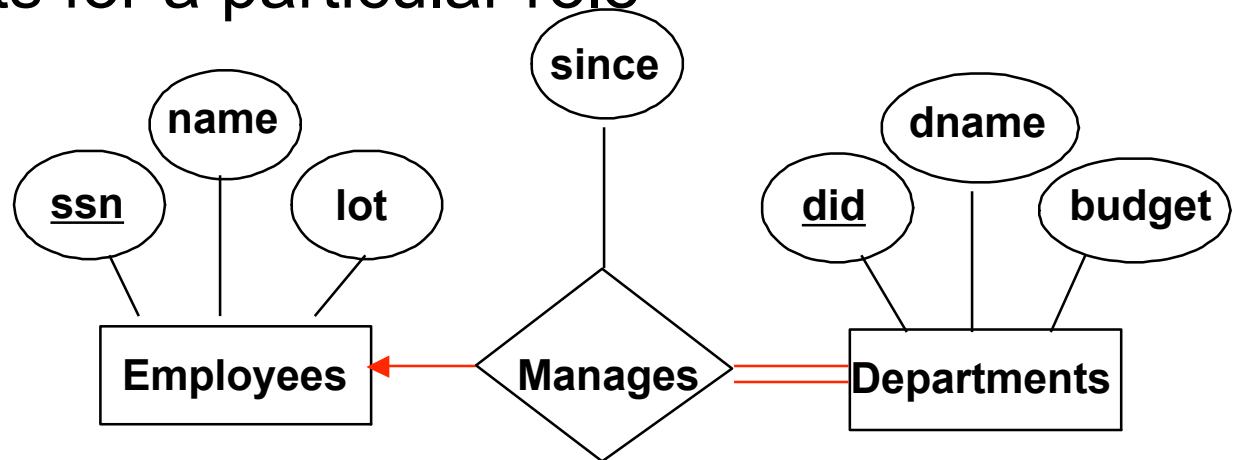
The many-to-one relationship *Manages* states that a department have *at most* one manager, it may have no manager.

What happens if *Departments* has total participation in *Manages*?



Referential Integrity

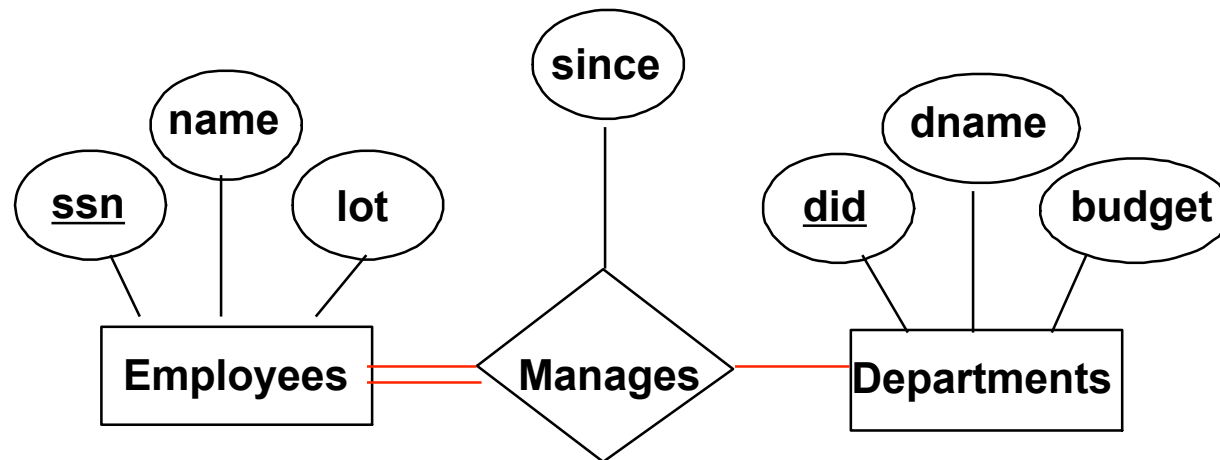
- The many-to-one relationship *Manages* states that a department has *at most* one manager, it may have no manager
- Combined with total participation asserts that *exactly one* value exists for a particular role



Forbid the deletion of an employee that manages a department, or delete both the employee and department!

Challenge Question

What happens if Employees has total participation in Manages?

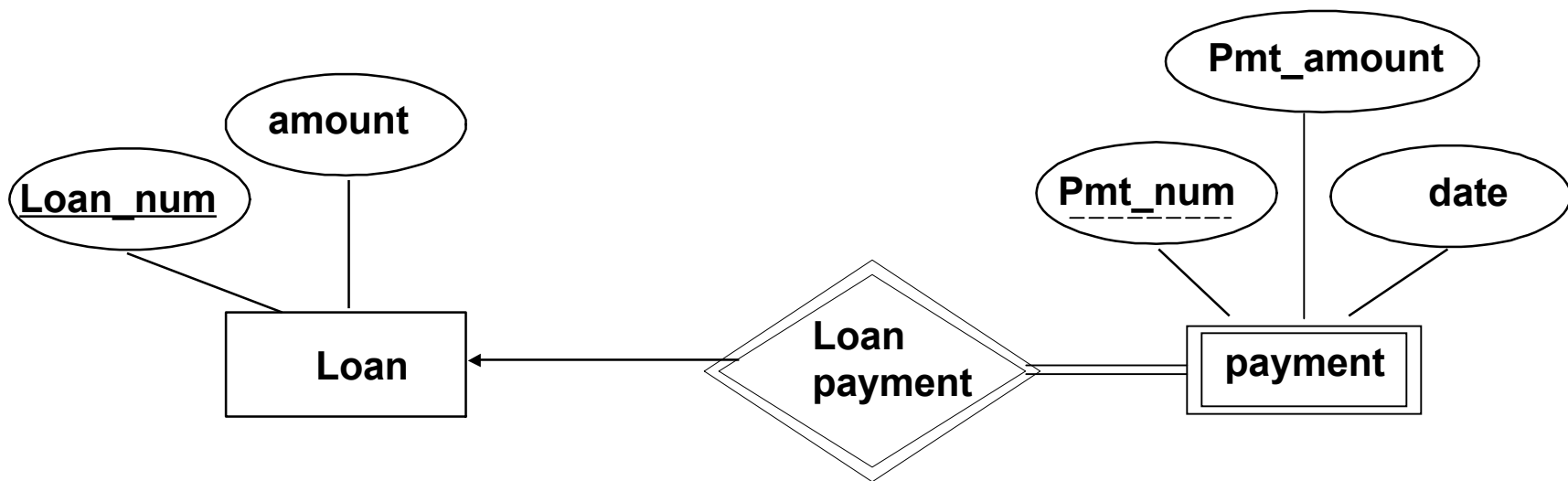


Weak Entity Sets

- Entity set that **does not have a primary key**
- Existence of a weak entity set depends on the existence of *a identifying entity set*
 - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
 - Identifying relationship depicted using a double diamond
- The *discriminator (or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set
- The *primary key* of a weak entity set is formed by:
primary key of the strong entity set plus the weak entity set's discriminator

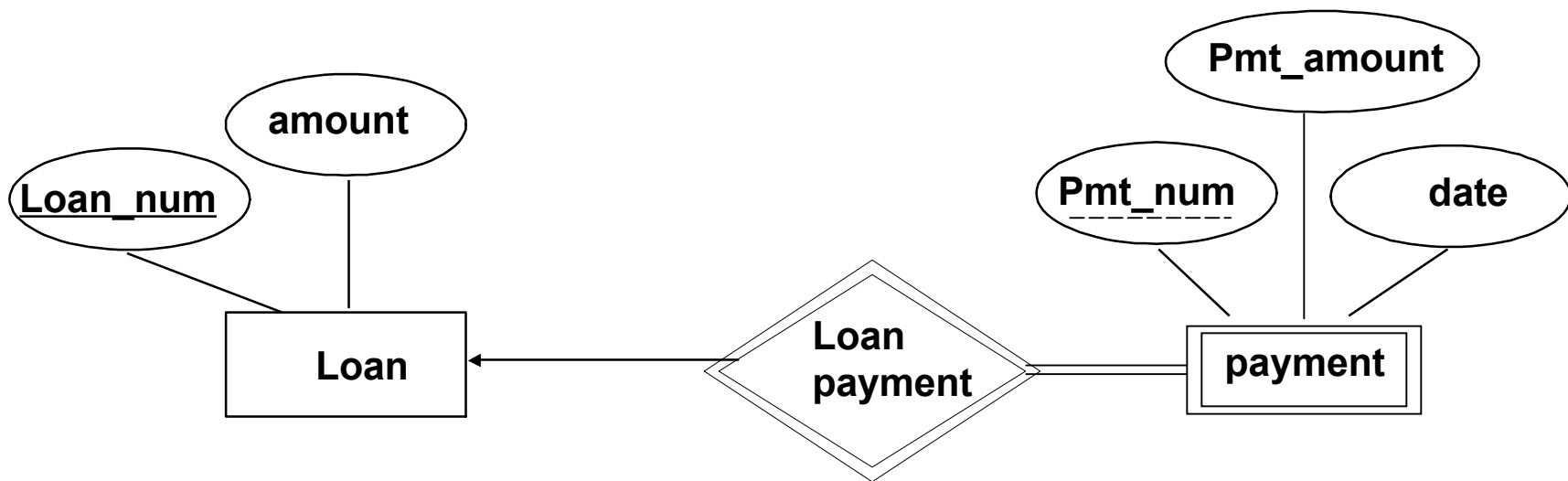
Weak Entity Sets - Example

- We depict a weak entity set by *double rectangles*
- We underline the discriminator of a weak entity set with a dashed line
- *payment-number* – discriminator of the *payment* entity set
- What is the primary key for *payment*?



Weak Entity Sets - Example

- We depict a weak entity set by *double rectangles*
- We underline the discriminator of a weak entity set with a dashed line
- *payment-number* – discriminator of the *payment* entity set
- What is the primary key for *payment*?
(*loan-number, payment-number*)



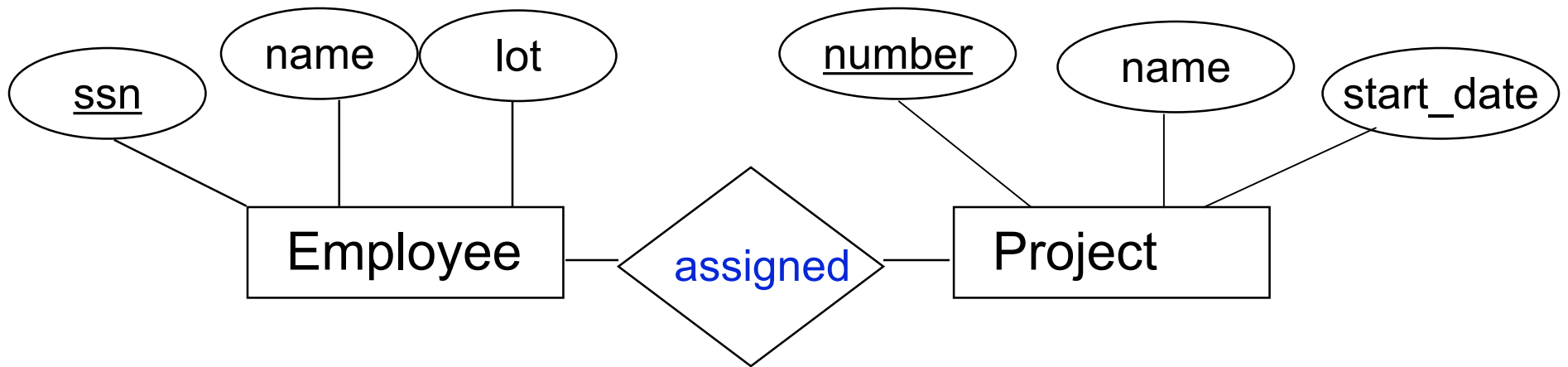
Strong vs. Weak Entity Sets

- Strong entity set:
 - Has sufficient attributes to form a primary key
- Weak entity set:
 - Lacks sufficient attributes to form a primary key
 - Hence, lacks sufficient attributes to form *any* key
- But every entity set needs a key; What to do?
 - Must *import* attributes from strong entity set(s)
 - A weak entity set member is subordinate to the dominant entity from strong entity set providing attributes to complete its key

Keys of Relationship Sets

- Suppose
 - R involves E_1, \dots, E_n
 - Primary key of E_i is $\text{primary-key}(E_i)$
 - R has no attributes
 - Attribute names are all unique
- Then
 - $\text{primary-key}(E_1) \cup \dots \cup \text{primary-key}(E_n)$
 - ... is a superkey of R
- Why talk about keys of relationship sets?
 - To enforce cardinality constraints
 - To prepare for tabular (relational) representation

Keys of Relationship Sets



What are the keys for *assigned*?

Design Issues

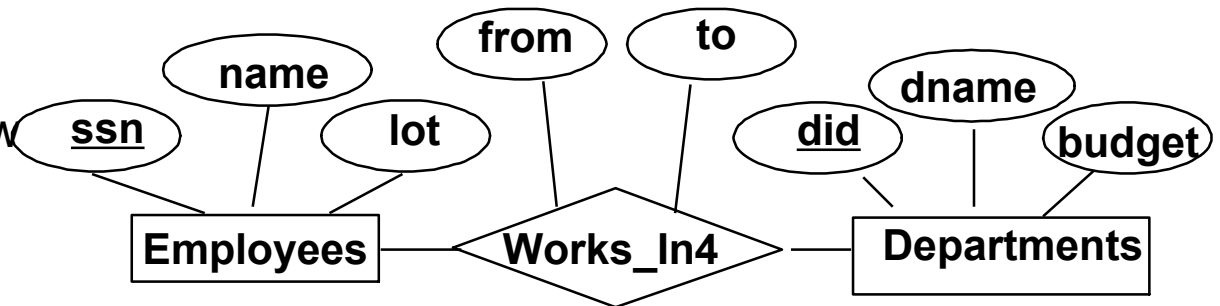
- Use of entity sets vs. attributes
 - Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question
 - *E.g., should Phone be an attribute of Employee or a separate entity?*
- Use of entity sets vs. relationship sets
 - Possible guideline is to designate a relationship set to describe an *action* that occurs between entities
- Binary versus n -ary relationship sets
 - Although it is possible to replace any nonbinary (n -ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a n -ary relationship set shows more clearly that several entities participate in a single relationship

Design Issues (cont.)

- Placement of relationship attributes
- The use of a strong or weak entity set
- The use of specialization contributes to modularity in the design

Entity vs. Attribute

- Works_In4 does not allow an employee to work in a department for two or more periods.

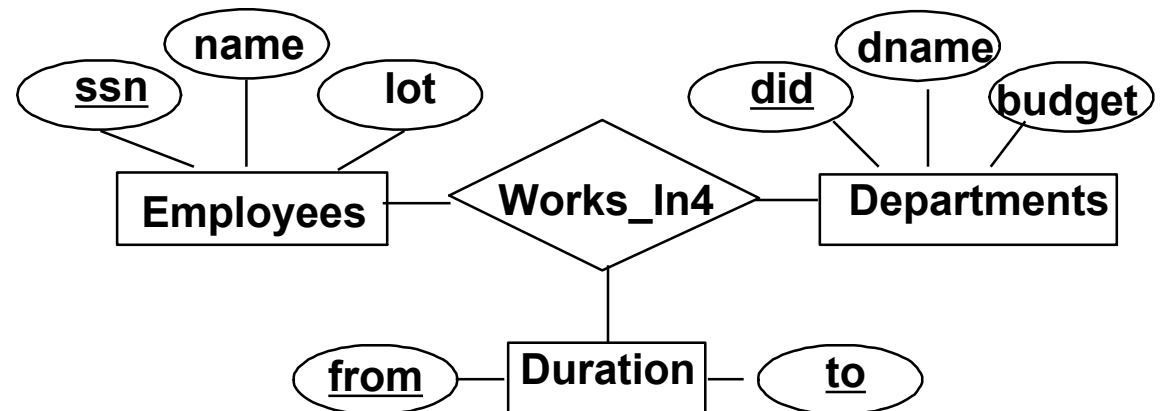


What if we want to record all possible periods an employee worked in a particular department?

Works_in4(s123,d1,f1,t1)

Works_in4(s123,d1,f2,t2)

Works_in4(s123,d2,from,to)

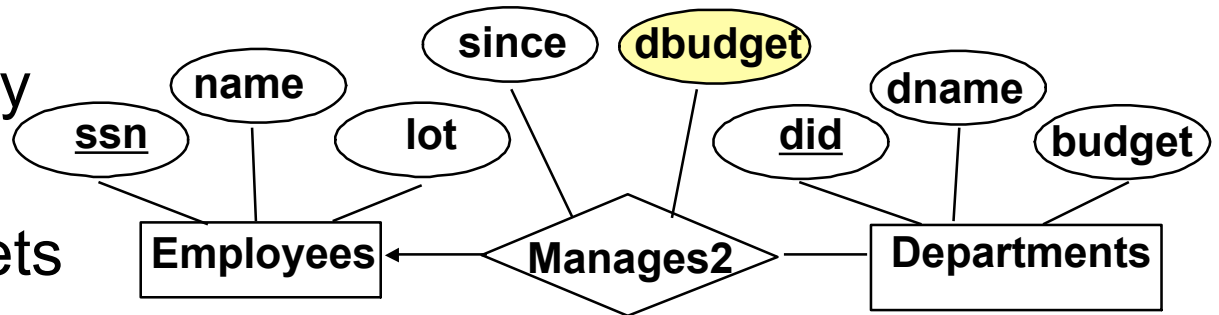


- Introduce a new entity set “Duration”, to record *several values of the descriptive attributes for each instance of this relationship.*

Entity vs. Relationship

manages2(ssn,did,since,dbudget)

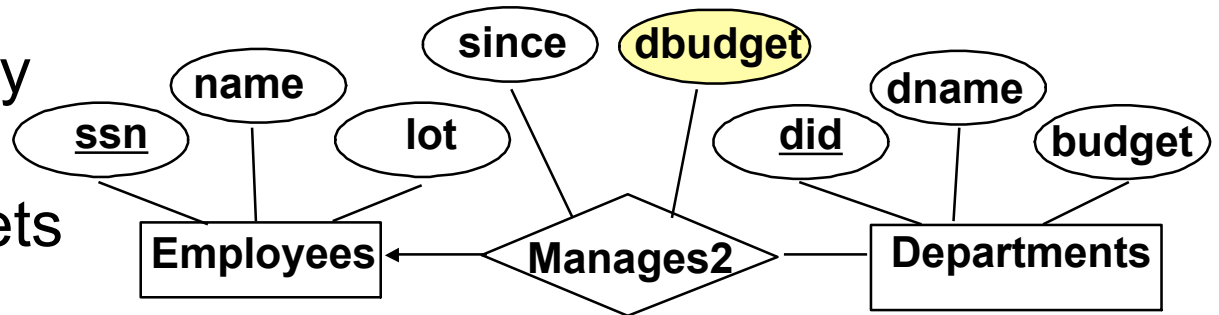
- Manager gets a separate discretionary budget for each dept
- What if a manager gets a discretionary budget that covers *all* managed depts?



Entity vs. Relationship

manages2(ssn,did,since,dbudget)

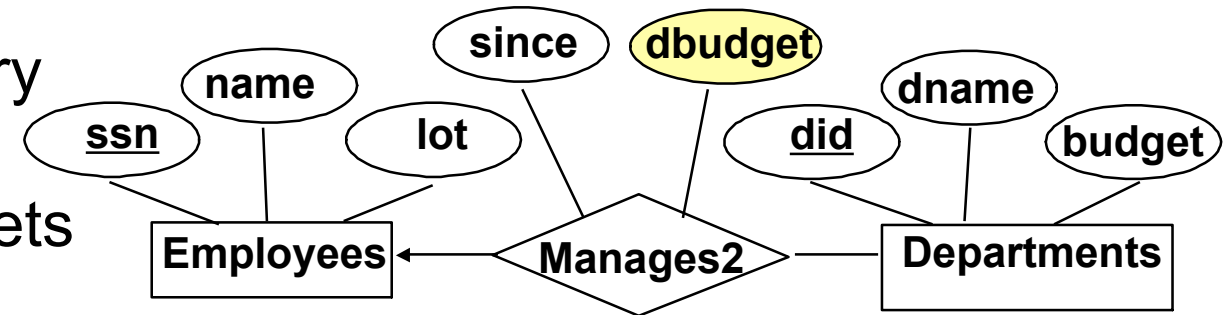
- Manager gets a separate discretionary budget for each dept
- What if a manager gets a discretionary budget that covers *all* managed depts?
 - **Redundancy:** *dbudget* stored for each dept managed by manager.
 - **Misleading:** Suggests *dbudget* associated with department-mgr combination.



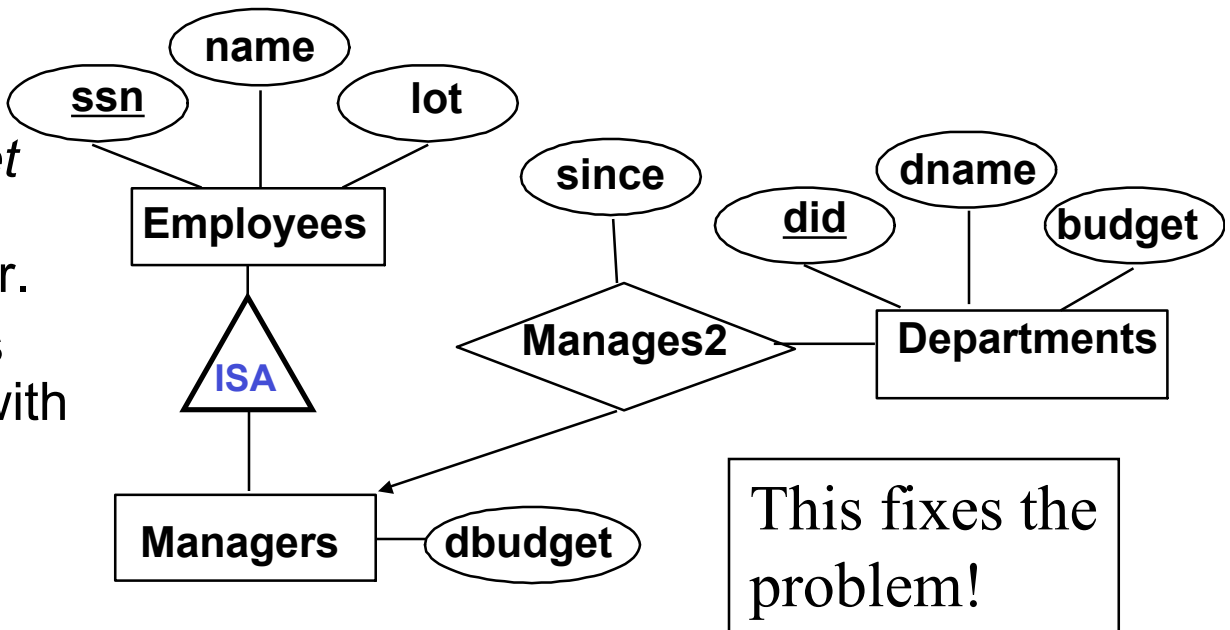
Entity vs. Relationship

manages2(ssn,did,since,dbudget)

- Manager gets a separate discretionary budget for each dept
- What if a manager gets a discretionary budget that covers *all* managed depts?



- **Redundancy:** *dbudget* stored for each dept managed by manager.
- **Misleading:** Suggests *dbudget* associated with department-mgr combination.



Design Principles

What makes a design good or bad?

- Design should be faithful to specifications
- Avoid redundancy – **more on normalization later!**
- Keep it simple
 - Avoid creating unnecessary entities/relationships
- Pick the right kind of element (see examples “Entity vs. Relationship” and “Entity vs. Attribute”)
 - Rule of thumb: if *thing* has more info than just its name, make it an entity

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*
- Note: There are many variations on ER model

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: e.g., *key constraints*, and *participation constraints*. 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.

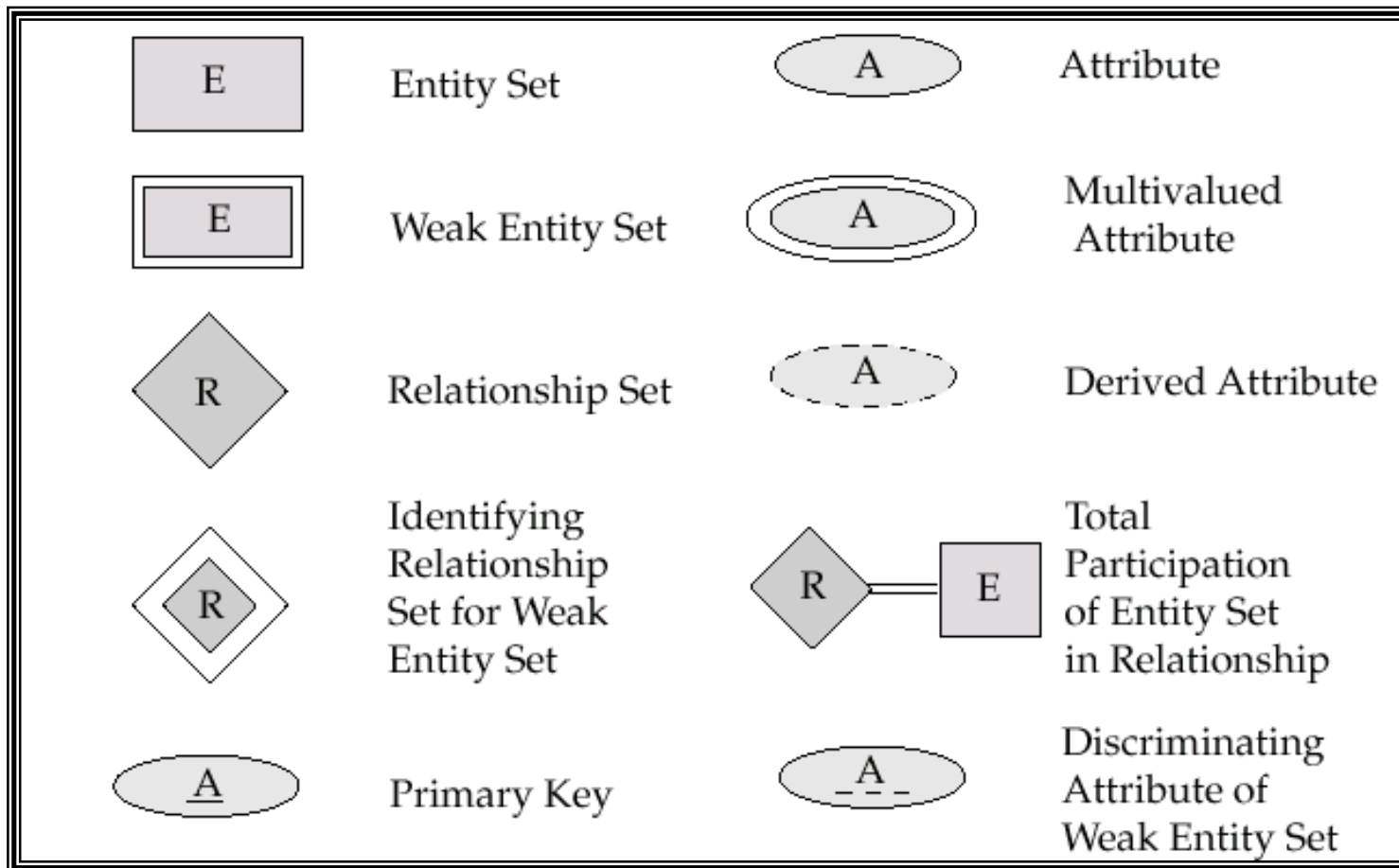
More about this later!

Summary of ER (Contd.)

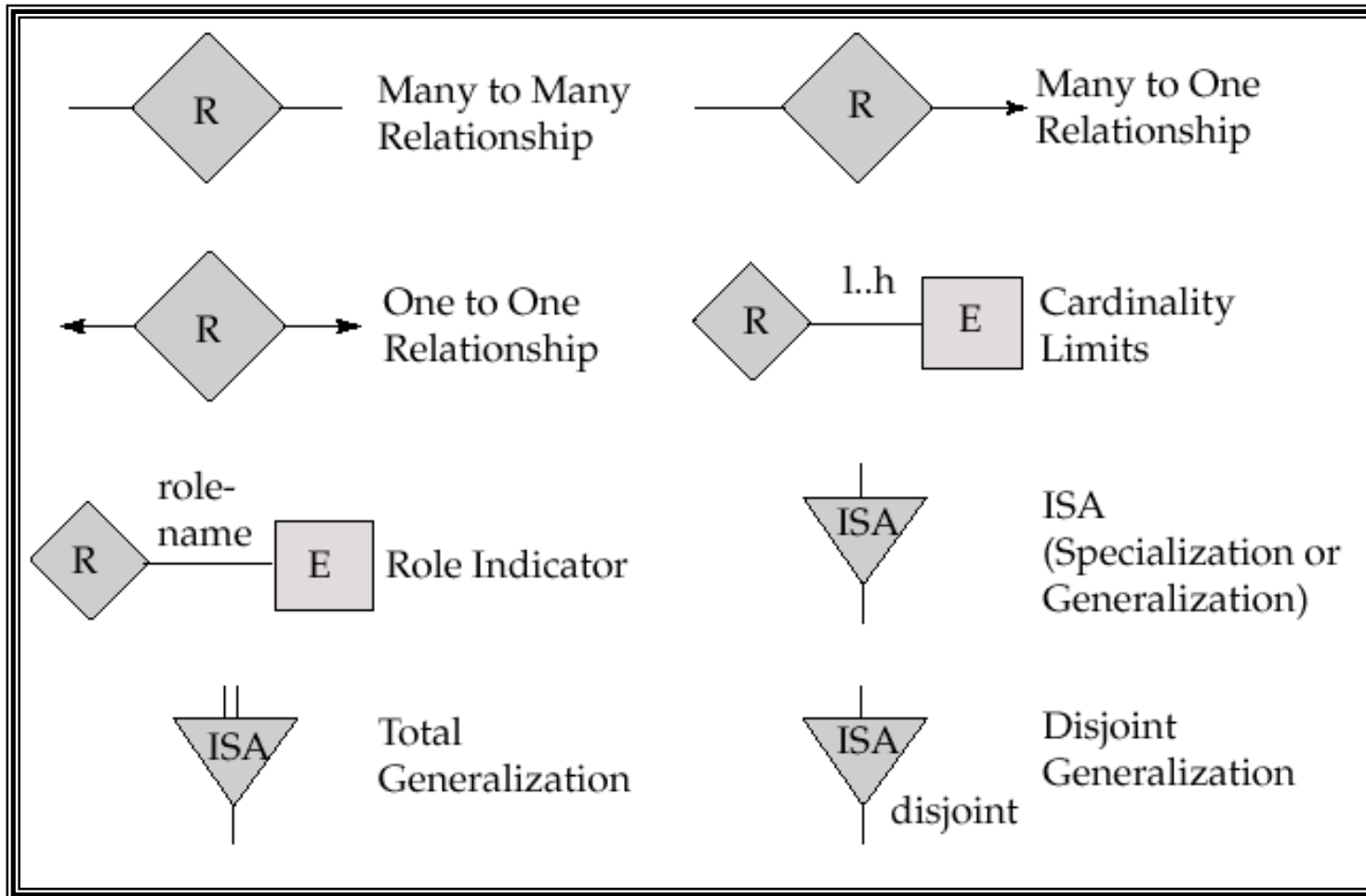
- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. Functional dependency information and normalization techniques are especially useful.

More about this later!

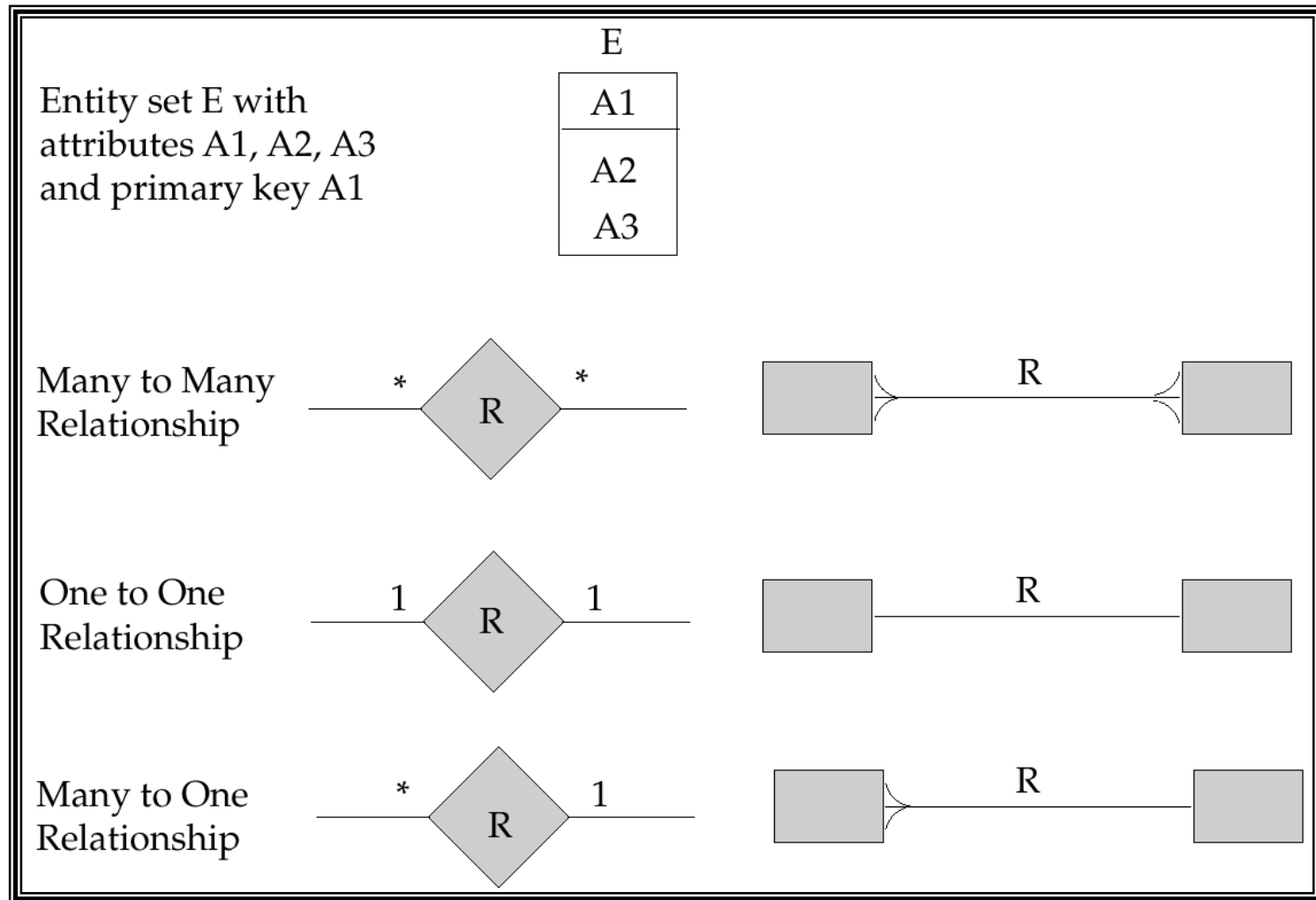
Summary of Symbols Used in E-R Notation



Summary of Symbols (Cont.)



Alternative E-R Notations



Example*

Excerpt from specifications for some project management software

A project is an endeavor undertaken to create a product or service. An organization pursues many projects at once. Every project has a definite beginning and a definite end. A project manager oversees all aspects of the project from its start to end. A large project might have more than one manager over its life, but any project has only one manager at any time.

Projects with many managers over their life are often subjects of conversation around the water cooler.

The project manager breaks down the project in to a list of component tasks--linear, not hierarchical--with regards to the project requirements (a project really starts when any of its tasks starts and ends when all its tasks are complete). Each task has an associated category such as design, development, and testing. It is not uncommon for the project manager to appoint a resource (called a lead) to monitor tasks in each of these categories. The project manager maintains a list of leads on the project and their contact information for immediate consultation.

A goal is established by/for individual resources to help achieve an organizational goal. A goal is established with regards to the duties, functions, responsibilities, skills and talents of available resources. A goal is the smallest component of management. It is associated with only one resource and may be related to only one task of one project. A task on the other hand may be associated with more than one resource and it may be broken down into one or more goals.

A resource is someone responsible for a goal. A resource has a manager, who is also a resource. A resource has a department, which serves as the 'home' of the resource. Project managers and leads are just resources playing those special roles in relation to a project between certain dates.