

# **21121350**

# **Database System**

## **Lecture 5: Entity-Relationship Model**

Lu Chen (陈璐)

College of Computer Science

Zhejiang University

Spring & Summer 2023

[luchen@zju.edu.cn](mailto:luchen@zju.edu.cn)/18868818726

# Steps of Database Design

- ❑ Requirement analysis
  - What data, applications, and operations needed.
- ❑ Conceptual database design
  - A high-level description of data, constraints using Entity-Relationship (E-R) model or a similar high level data model.
- ❑ Logical database design
  - Convert the conceptual design into a DB schema.
- ❑ Schema refinement
  - Normalization of relations: Check relational schema for redundancies and related anomalies.
- ❑ Physical database design
  - Indexing, query, clustering, and database tuning.
- ❑ Create and initialize the database & Security design
  - Load initial data, testing.
  - Identify different user groups and their roles.

# Outline

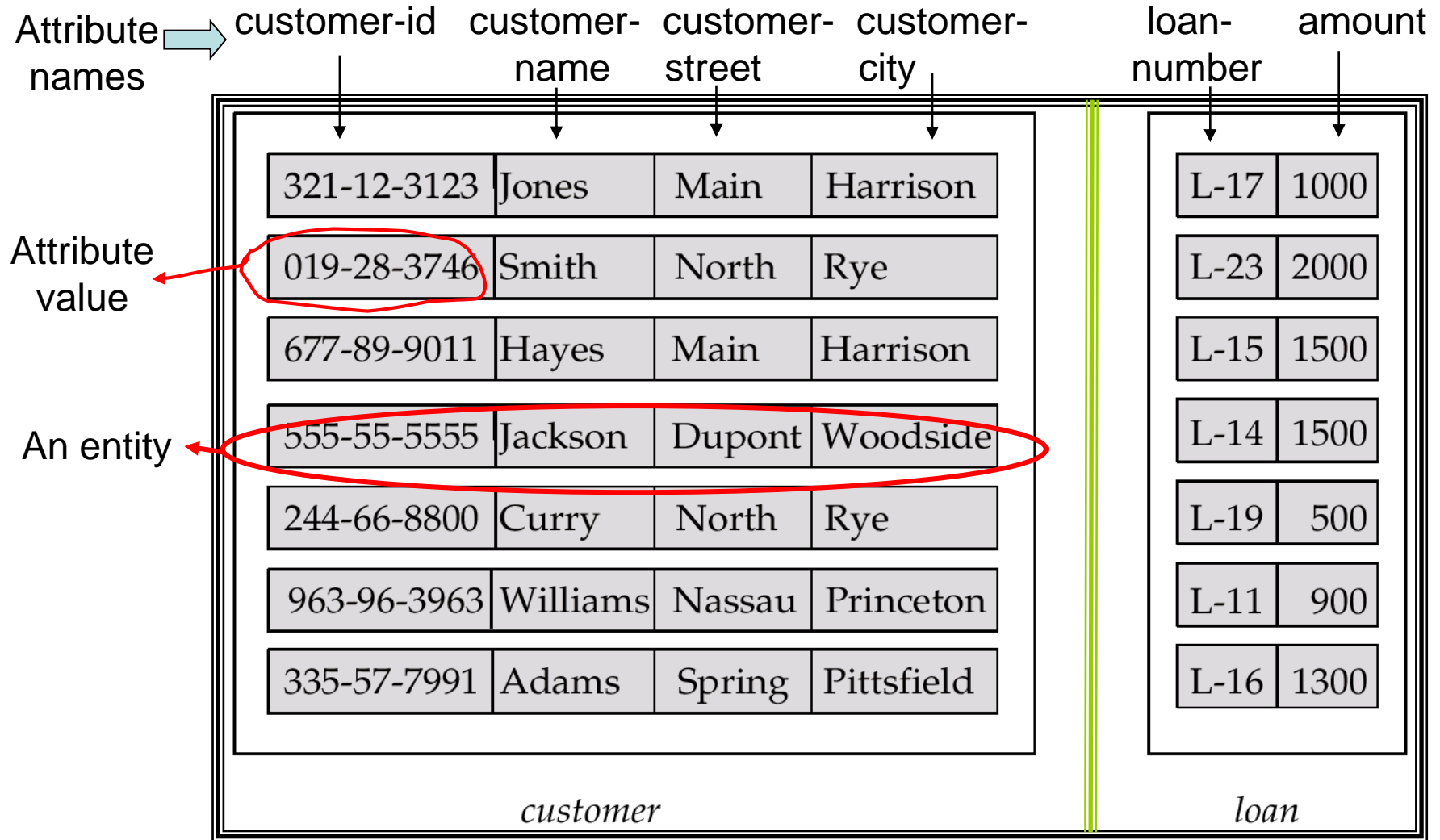
- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables



# Entity Sets

- ❑ The real world can be modeled as:
  - A collection of **entities** (实体)
  - **Relationships** (联系) among entities
- ❑ An **entity** is an object that exists and is distinguishable from other objects. --- **An entity may be concrete, or abstract.**
  - Example: specific *student*, *company*, *event*, *plant*.
- ❑ Entities have **attributes** (属性)
  - Example: student has *id*, *name*, *age*, *sex*, and *address*.
- ❑ An **entity set** is a set of entities of the **same type** that **share the same properties**.
  - Example: set of all students, companies, trees, holidays, customers, accounts, loans.
  - 一个实体集包含多个同类实体

# Example: Entity Sets *customer* and *loan*



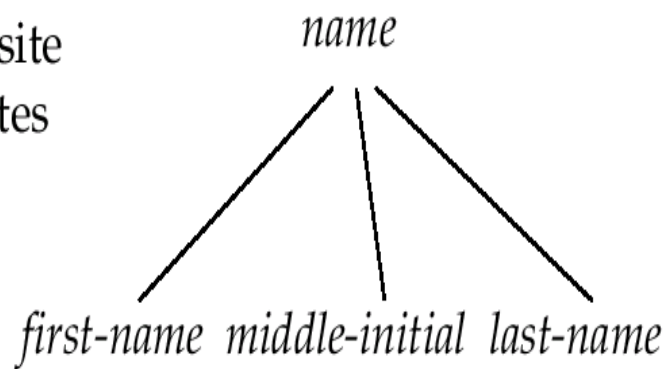
# Attributes

- ❑ An entity is represented by a set of attributes, that is **descriptive properties** possessed by all members of an entity set.
- ❑ **Domain** (域, value set) — the set of permitted values for each attribute.
- ❑ Attribute types:
  - **Simple** and **composite** attributes (简单和复合属性, 如sex, name).
  - **Single-valued** and **multi-valued** attributes(单值和多值属性).
    - E.g., multivalued attribute: phone-numbers (多个电话号码).
  - **Derived** attributes (派生属性).
    - Can be computed from other attributes, e.g., age, given date of birth.
    - versus **base attributes** or **stored attributes** (基属性, 存储属性).

# Composite Attributes

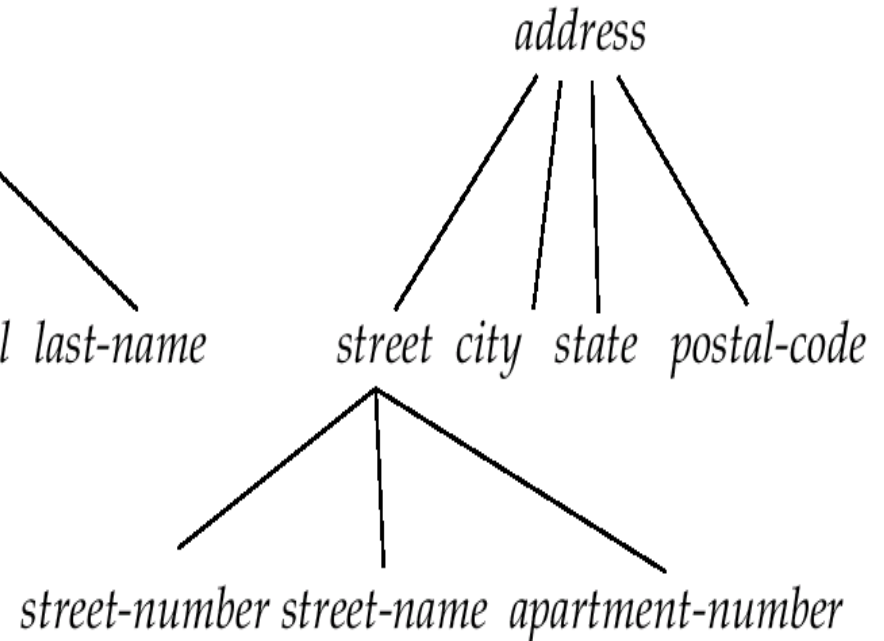
复合  
属性

Composite  
Attributes



分量  
属性

Component  
Attributes



# Outline

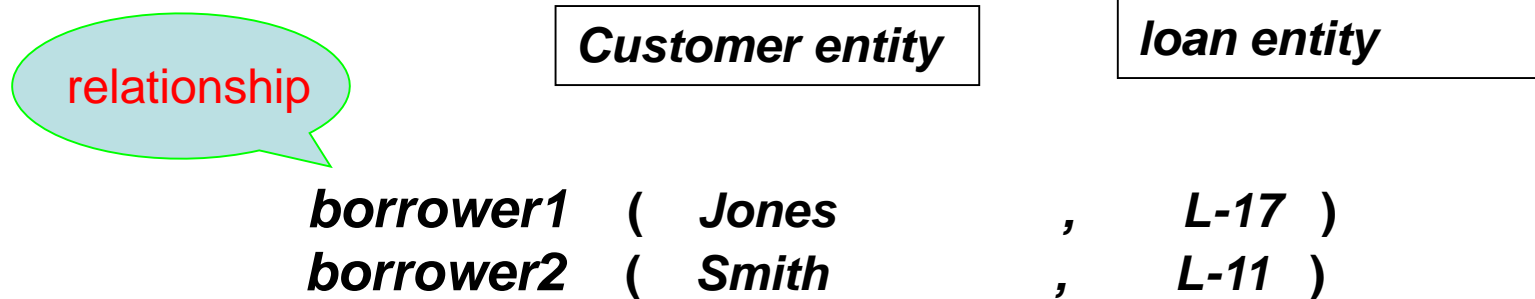
- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables





# Relationship Sets

- ❑ A relationship is an association among several entities (是二个或更多个不同类实体之间的关联).



- ❑ A relationship set is a set of relationship of the same type.
  - Example: *borrower* (*customer-name*, *loan-number*)
- ❑ 一个联系集包含多个同类联系 (或联系实例, relationship instance)
- ❑ 一个联系集表示二个或多个实体集之间的关联

# Relationship Sets (Cont.)

- Formally, a **relationship** set is a mathematical relation among  $n \geq 2$  entities, each taken from entity sets

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

where  $(e_1, e_2, \dots, e_n)$  is a relationship,  $E_i$  is an entity set.

- Example:  $(\text{Jones}, \text{L-17}) \in \text{borrower}$ , where  $\text{Jones} \in \text{customer}$  and  $\text{L-17} \in \text{loan}$

$$\left\{ \begin{array}{l} (3032111020, 1, 95) \\ (3032111021, 1, 90) \\ (3032111022, 2, 75) \\ \dots \dots \end{array} \right\} \in \text{student, course}$$

# Example 1

**Students**

<b>Sid</b>	Sname	Ssex	Sage	Specialty
3023001093	Tom	M	21	Cs
3011112340	Mary	F	20	Cs
3020621034	Jack	M	18	Cs
3020831035	Smith	M	19	Ma
3021131123	Alane	F	22	Is

relationship

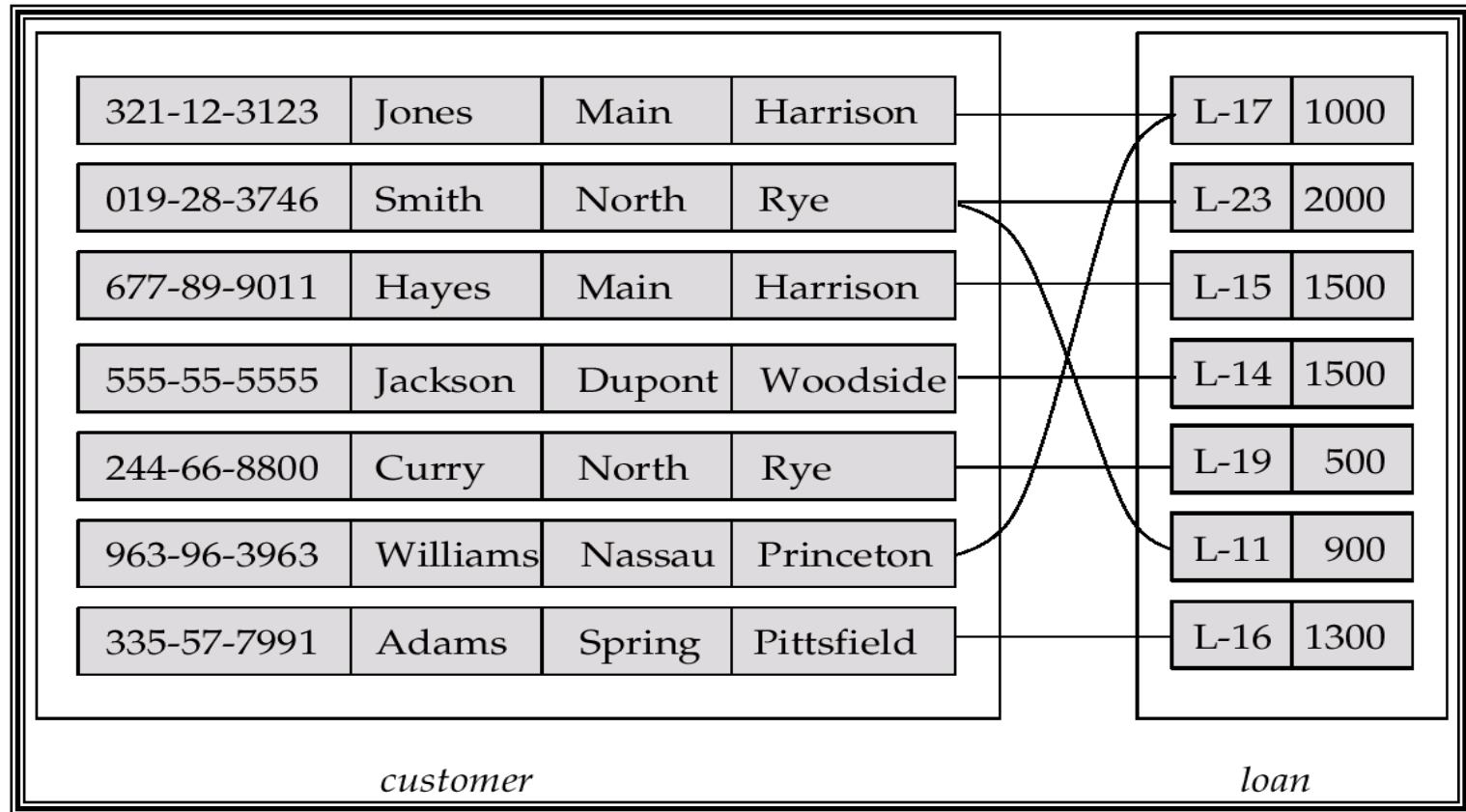
**Enrolled**

<b>sid</b>	<b>cid</b>	grade
3023001093	1	90
3023001093	2	85
3020621034	1	90
3020831035	1	75
3021131123	2	75

**Courses**

<b>cid</b>	Cname	credit
1	DB	4
2	OS	5
3	English	4
4	Math	4

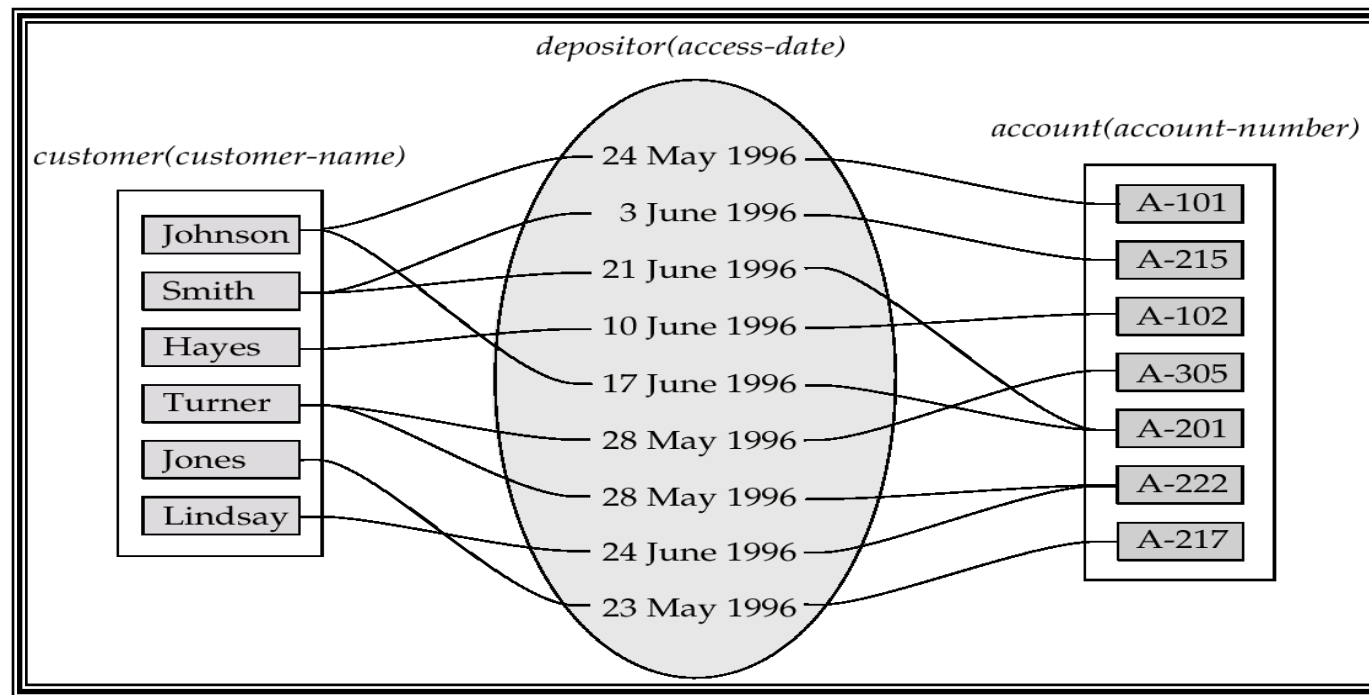
## Example 2: Relationship Set *borrower*



*Borrower(customer-name, loan-number)*

# Example 3: Relationship Set

- ❑ An **attribute** can also be the **property of a relationship set**.
- ❑ For instance, the depositor relationship set between entity sets customer and account may have the attribute **access-date**.



*depositor(customer-name, account-number, access-date)*

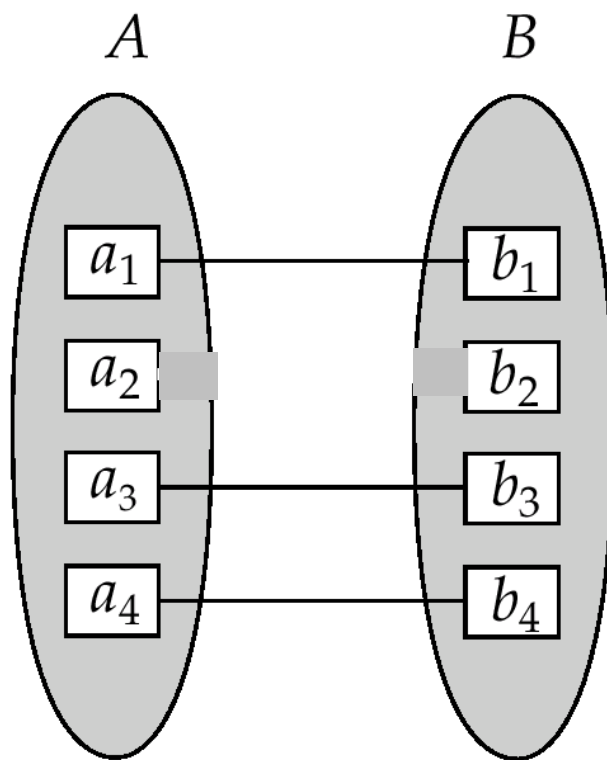
# Degree of a Relationship Set

- ❑ Refers to the number of entity sets that participate in a relationship set.
- ❑ Relationship sets that **involve two entity sets** are **binary** (or degree two).
- ❑ Relationship sets may **involve more than two entity sets**.
  - E.g., suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a **ternary relationship set** between entity sets (*employee, job, branch*)
- ❑ Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)

# Mapping Cardinalities

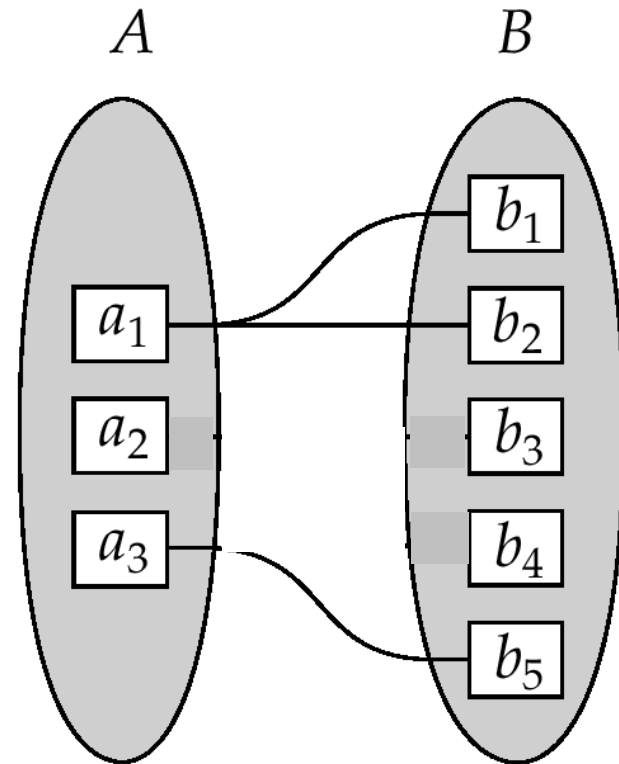
- ❑ Express the number of entities to which another entity can be associated via a relationship set. (一个联系集中，一个实体可以与另一类实体相联系的实体数目。其中数目是指最多一个还是多个)
- ❑ Most useful in describing **binary relationship sets**.
- ❑ For a **binary relationship set** the mapping cardinality must be one of the following types:
  - One to one (1 : 1), 如: 就任总统 (总统, 国家)
  - One to many (1 : n), 如: 分班情况 (班级, 学生)
  - Many to one (n : 1), 如: 就医 (病人, 医生)
  - Many to many (n : m), 如: 选课 (学生, 课程)

# Mapping Cardinalities (Cont.)



(a)

One to one



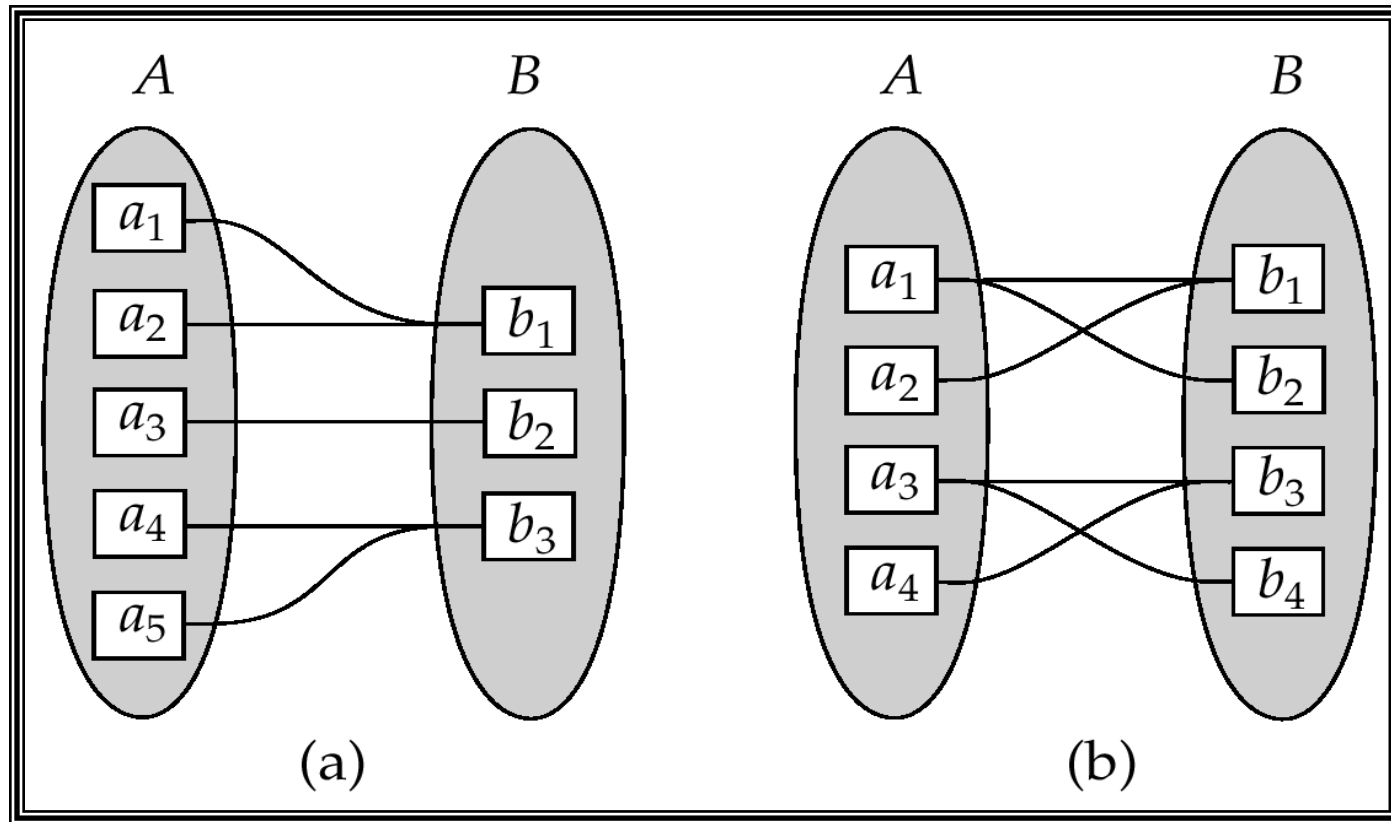
(b)

One to many

**Note:** Some elements in entity set A and B may not be mapped to any elements in the other set.



# Mapping Cardinalities (Cont.)



Many to one

Many to many

**Note:** Some elements in A and B may not be mapped to any elements in the other set.

# Outline

- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ **Keys**
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables



# Keys for Entity Sets

- ❑ A **super key** of an entity set is a set of one or more attributes whose values **uniquely determine each entity**.
- ❑ A **candidate key** of an entity set is **a minimal super key**.
  - *Customer-id* is a candidate key of *customer*.
  - *Loan-number* is a candidate key of *loan*.
- ❑ Although several candidate keys may exist, one of the candidate keys is selected to be the primary key.
- ❑ Example: *customer*(*cus-num*, *cus-name*, *cus-street*, *cus-city*)
  - **Candidate key**: *cus-num*
  - **Super key**: {*cus-num*}, {*cus-num*, *cus-name*}, {*cus-num*, ...}

# Keys for Relationship Sets

- ❑ The combination of primary keys of the participating entity sets forms a super key of a relationship set (参与一个联系集的各实体集的码的组合，构成该联系集的超码).
- ❑ (*customer-id, loan-number*) is the super key of *borrower*
  - (*sid, cid*) is the super key of *enrolled*
  - Note: this means a pair of entity can have at most one relationship in a particular relationship set.
- ❑ Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys (1:1, 1:n, m:n).
- ❑ Need to consider semantics of relationship set in selecting the primary key in case of more than one candidate key, e.g. 作为码的属性不能为空，值不应常变.

# Example 1

**Students**

<b>Sid</b>	Sname	Ssex	Sage	Specialty
3023001093	Tom	M	21	Cs
3011112340	Mary	F	20	Cs
3020621034	Jack	M	18	Cs
3020831035	Smith	M	19	Ma
3021131123	Alane	F	22	Is

relationship

**Enrolled**

<b>sid</b>	<b>cid</b>	grade
3023001093	1	90
3023001093	2	85
3020621034	1	90
3020831035	1	75
3021131123	2	75

**Courses**

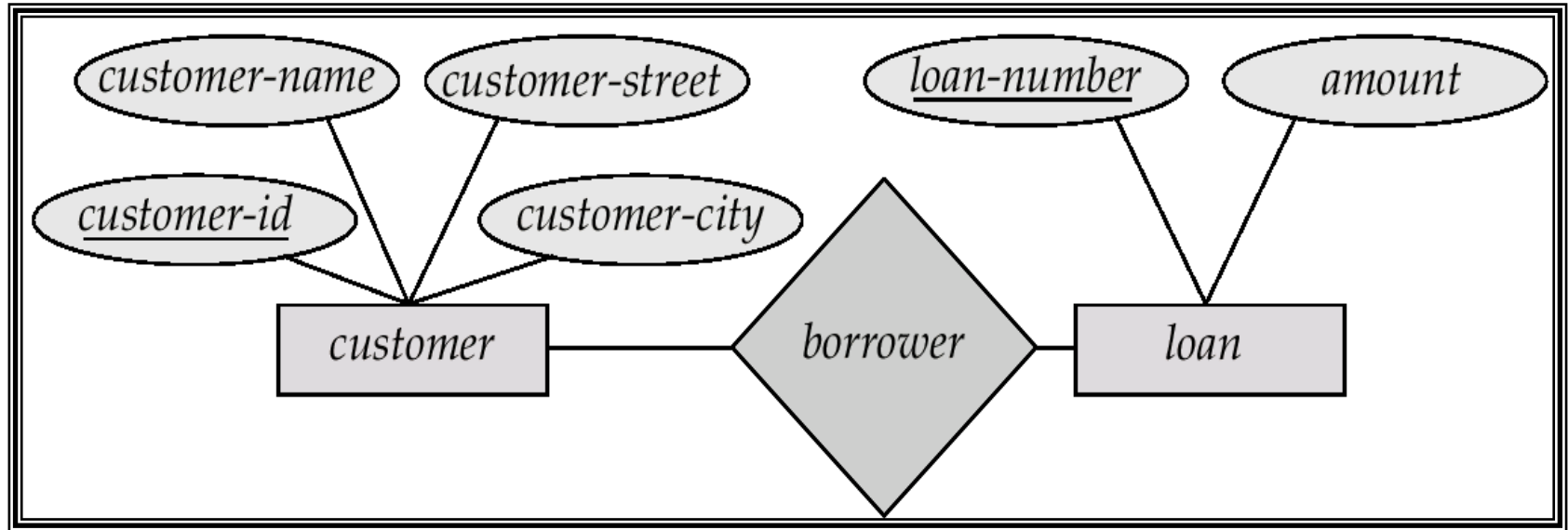
<b>cid</b>	Cname	credit
1	DB	4
2	OS	5
3	English	4
4	Math	4

# Outline

- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables

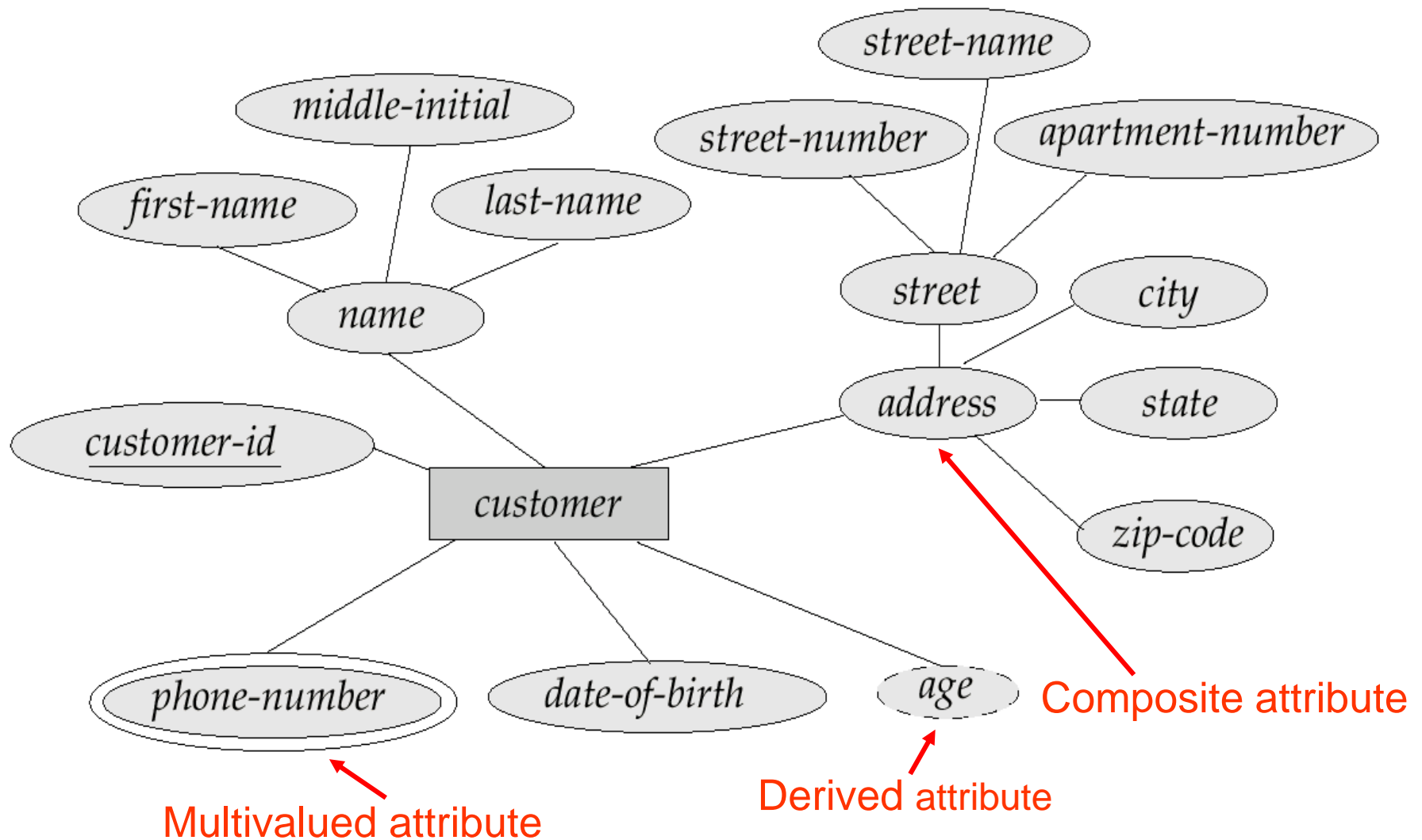


# E-R Diagrams



- ❑ 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).

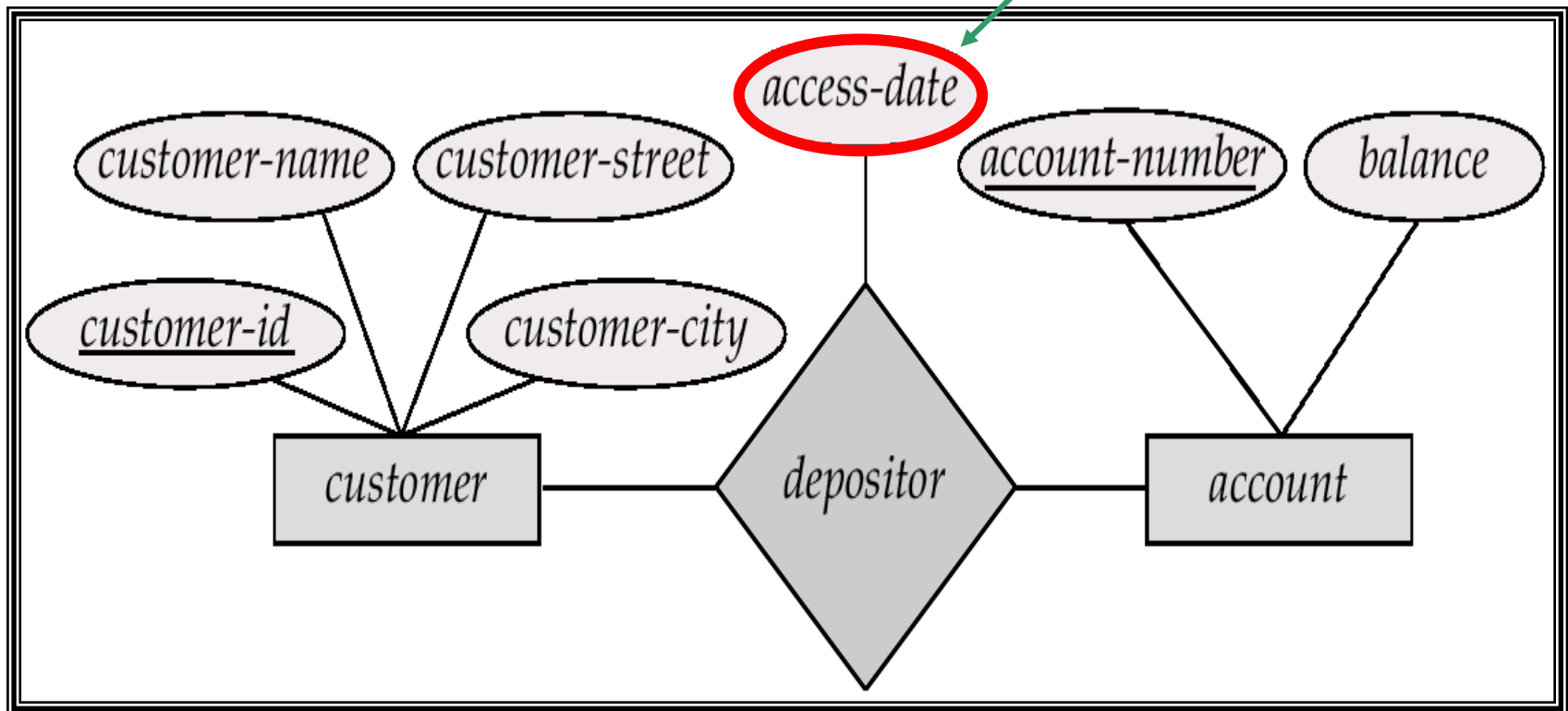
# E-R Diagram With Composite, Multivalued, and Derived Attributes





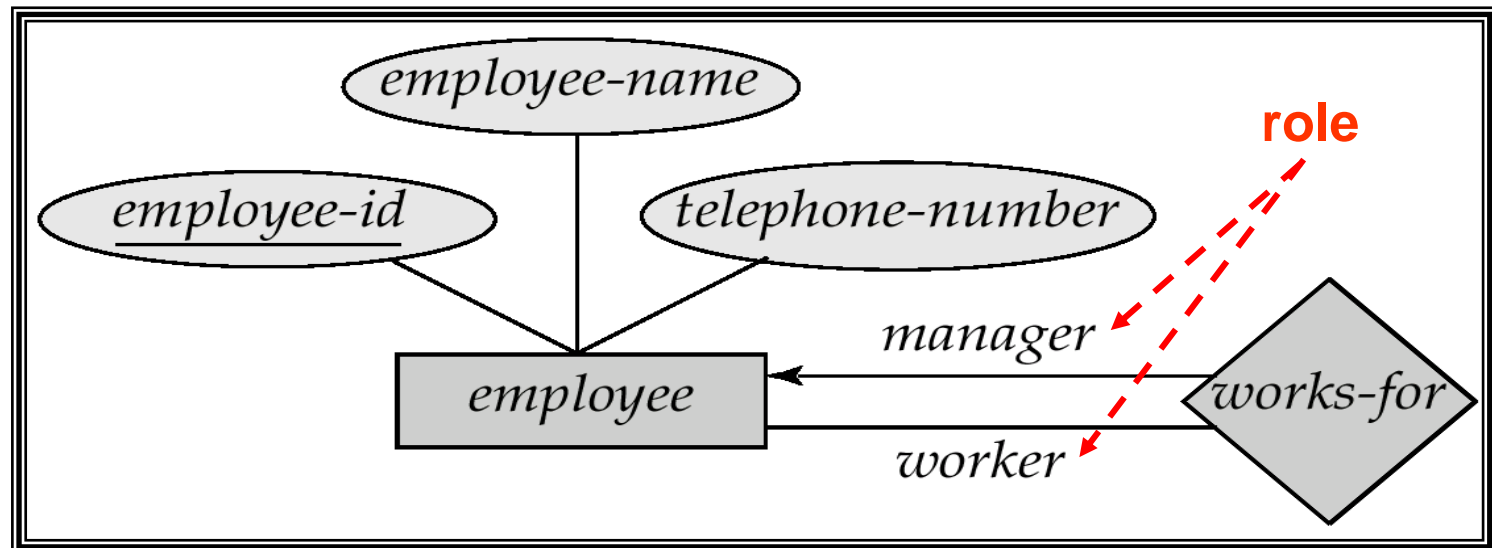
# Relationship Sets with Attributes

Descriptive attribute of the relationship



# E-R Diagrams

- ❑ Entity sets of a relationship need not be distinct, e.g., **Recursive relationship set** (自环联系集).
- ❑ **Role**: the function that an entity plays in a relationship, e.g., the labels “**manager**” and “**worker**” are called roles; they specify how employee entities interact via the works-for relationship set.
- ❑ Role labels are optional, and are used to clarify semantics of the relationship.

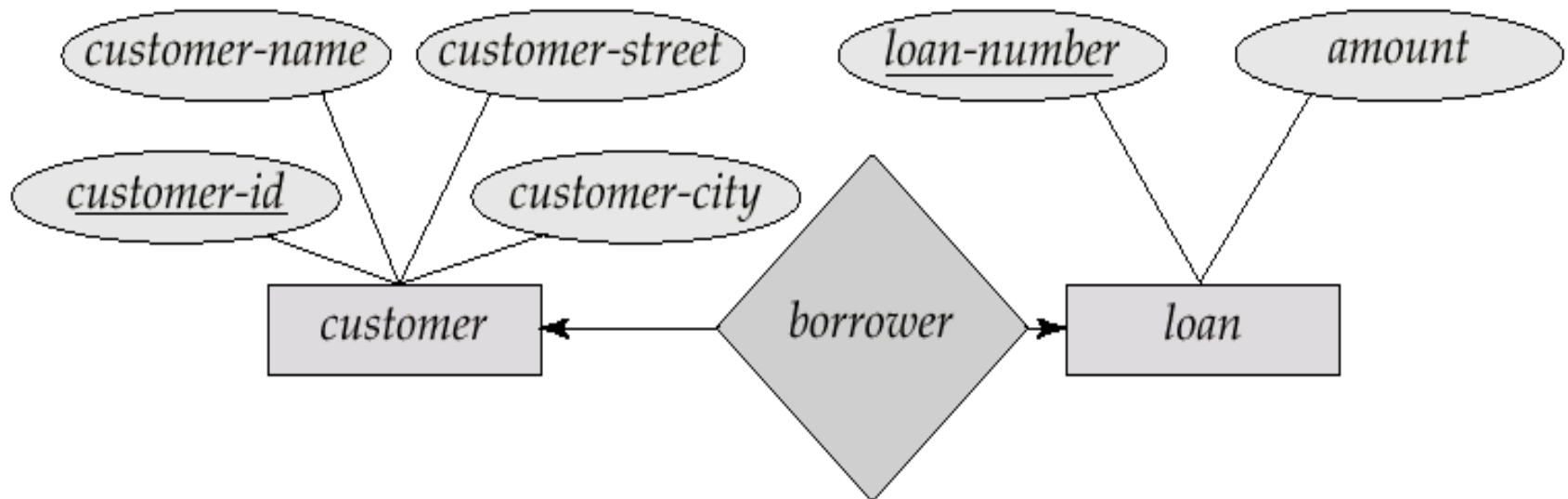


# Express the Cardinality Constraints

- We express cardinality constraints by drawing either a **directed line** ( $\rightarrow$ ), signifying “**one**”, or **an undirected line** ( $—$ ), signifying “**many**”, between the relationship set and the entity set.

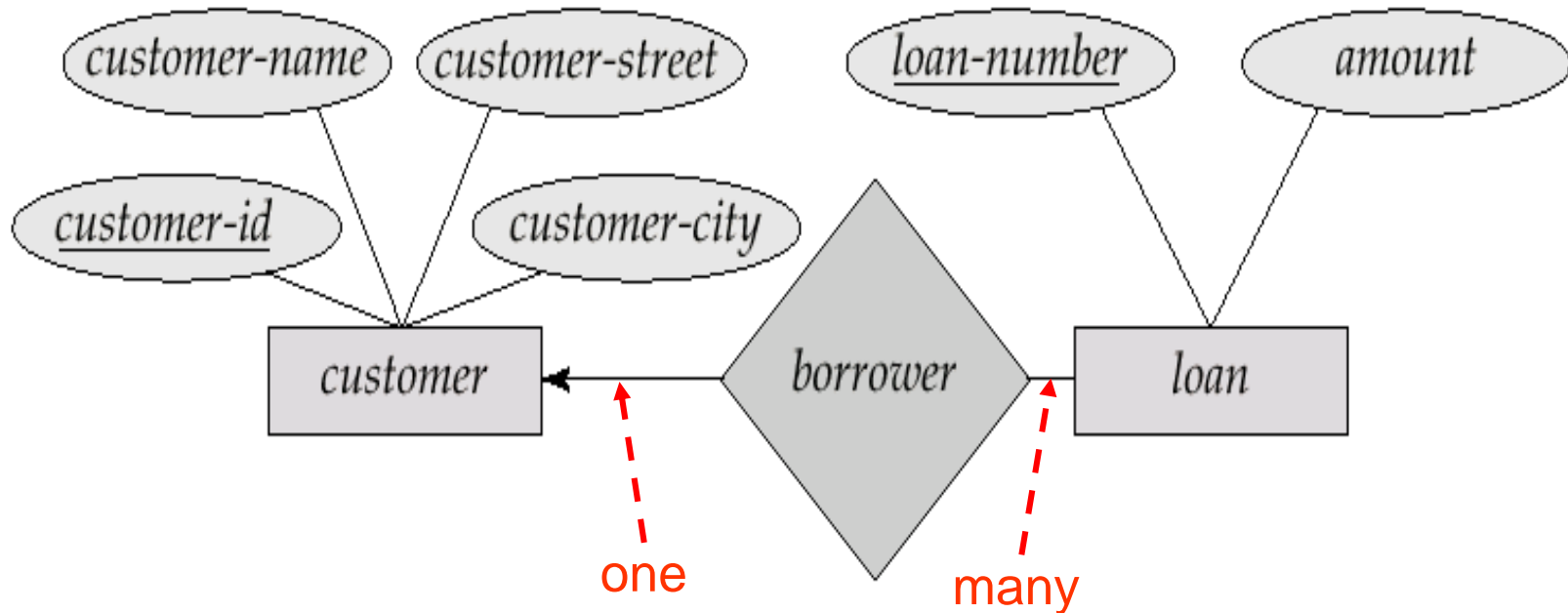
# One-To-One Relationship

- ❑ A *customer* is associated with **at most one loan** via the relationship *borrower*.
- ❑ A *loan* is associated with **at most one customer** via *borrower*.
- ❑ 注意: 一个联系集的类型取决于被表达对象的语义约束及设计者的意图.



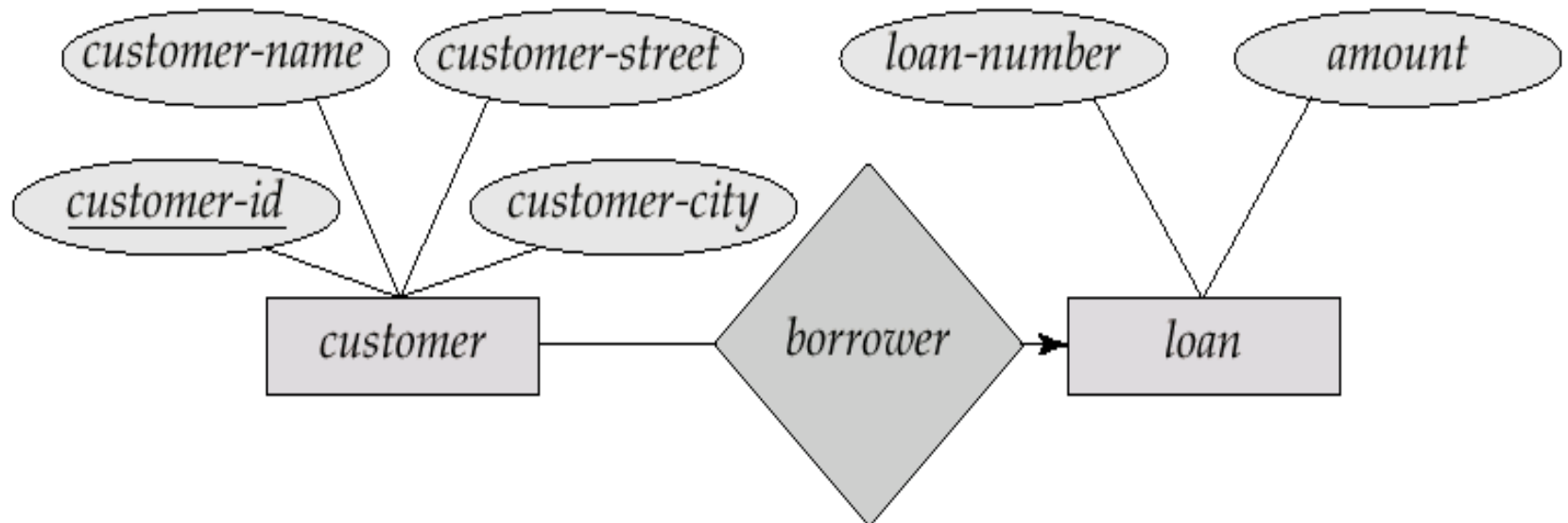
# One-To-Many Relationship

- ❑ A *loan* is associated with **at most one** *customer* via *borrower*.
- ❑ A *customer* is associated with **several** (including 0) *loans* via *borrower*.



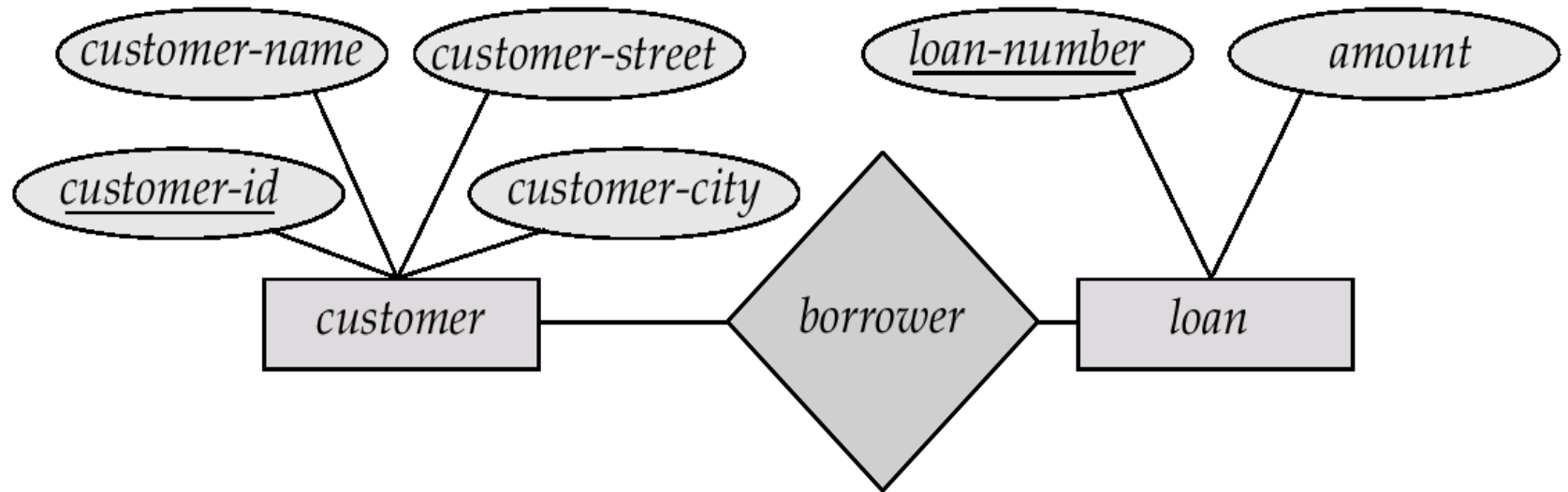
# Many-To-One Relationship

- ❑ A *loan* is associated with *several* (including 0) *customers* via *borrower*.
- ❑ A *customer* is associated with *at most one loan* via *borrower*.



# Many-To-Many Relationship

- ❑ A *customer* is associated with *several* (possibly 0) *loans* via *borrower*.
- ❑ A *loan* is associated with *several* (possibly 0) *customers* via *borrower*.



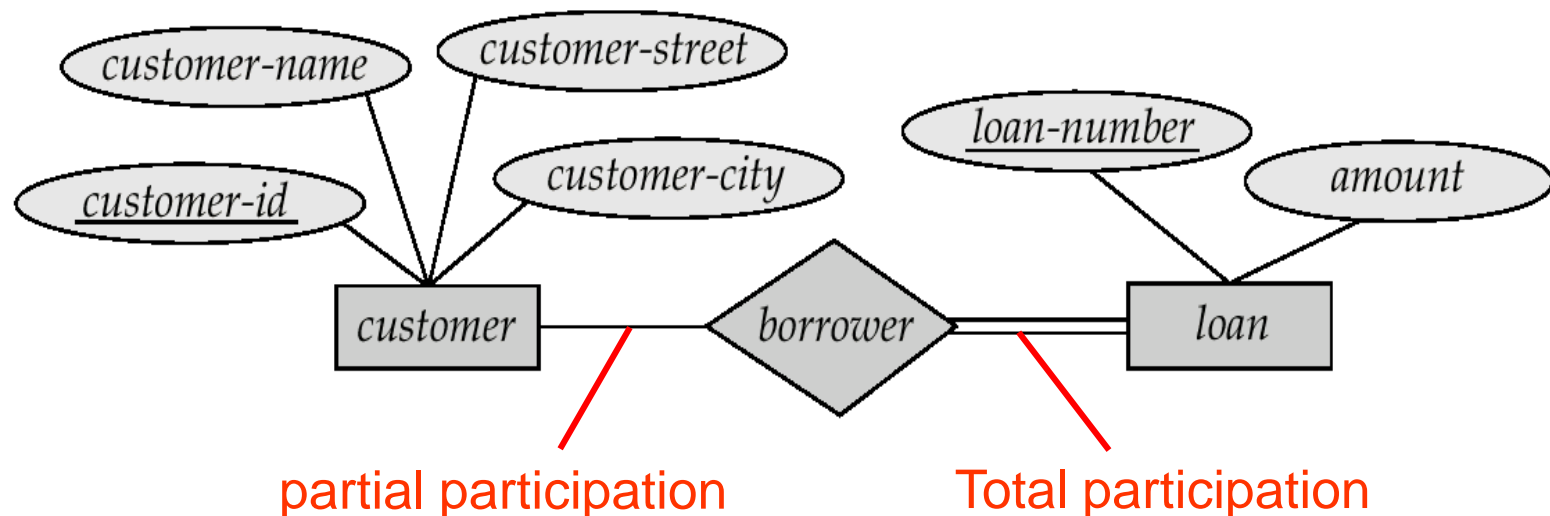
# Participation of an Entity Set in a Relationship Set

- ❑ **Total participation (全参与) (indicated by double line)**: every entity in the entity set participates in at least one relationship in the relationship set.
  - E.g., participation of *loan* in *borrower* is total.
  - Every loan must have a customer associated to it via borrower.
  
- ❑ **Partial participation (部分参与)**: some entities may not participate in any relationship in the relationship set.
  - E.g., participation of *customer* in *borrower* is partial.



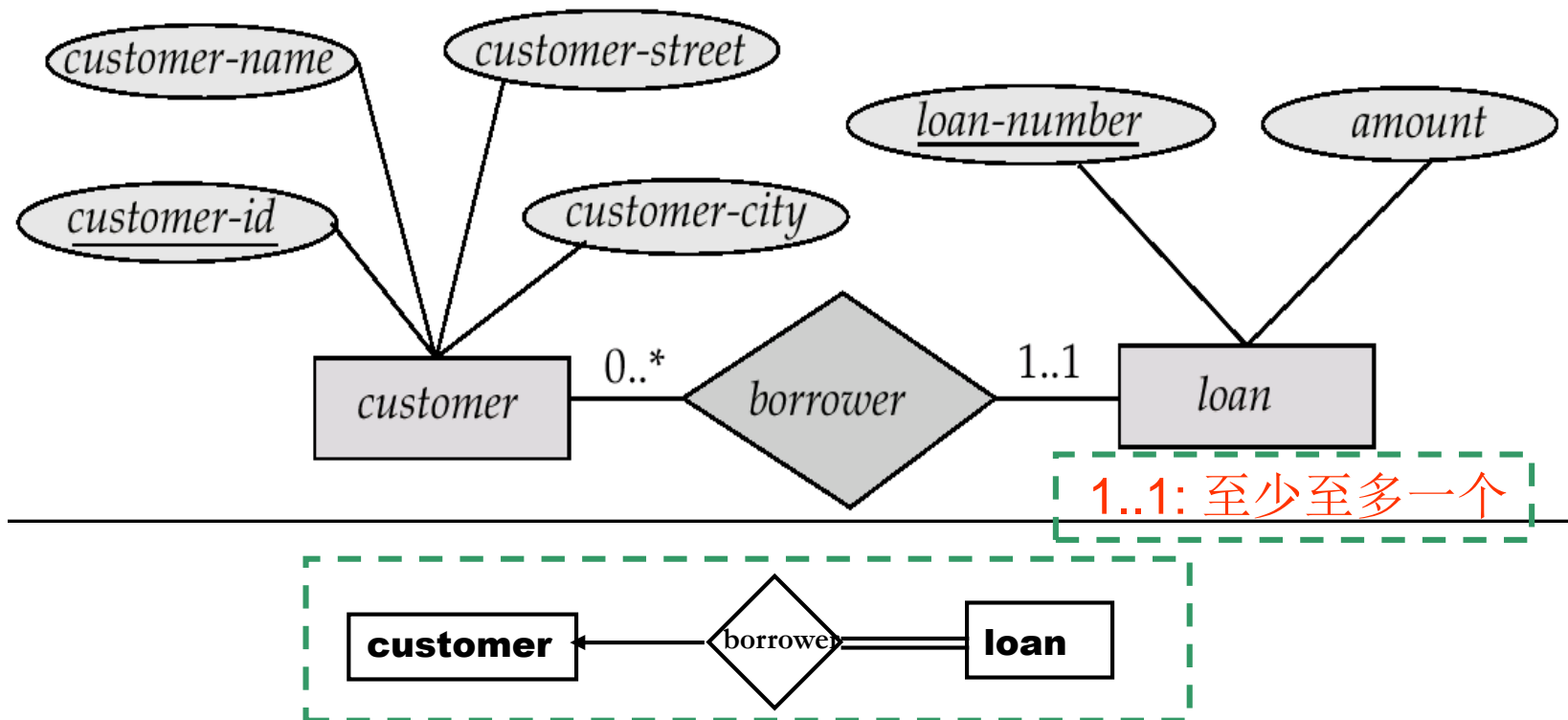
# Participation of an Entity Set in a Relationship Set (Cont.)

- ❑ 映射基数约束(Mapping cardinality constraints), 限定了一个实体与发生关联的另一端实体可能关联的数目上限。
- ❑ 全参与和部分参与约束, 则反映了一个实体参与关联的数目下限: 0次, 还是至少1次。



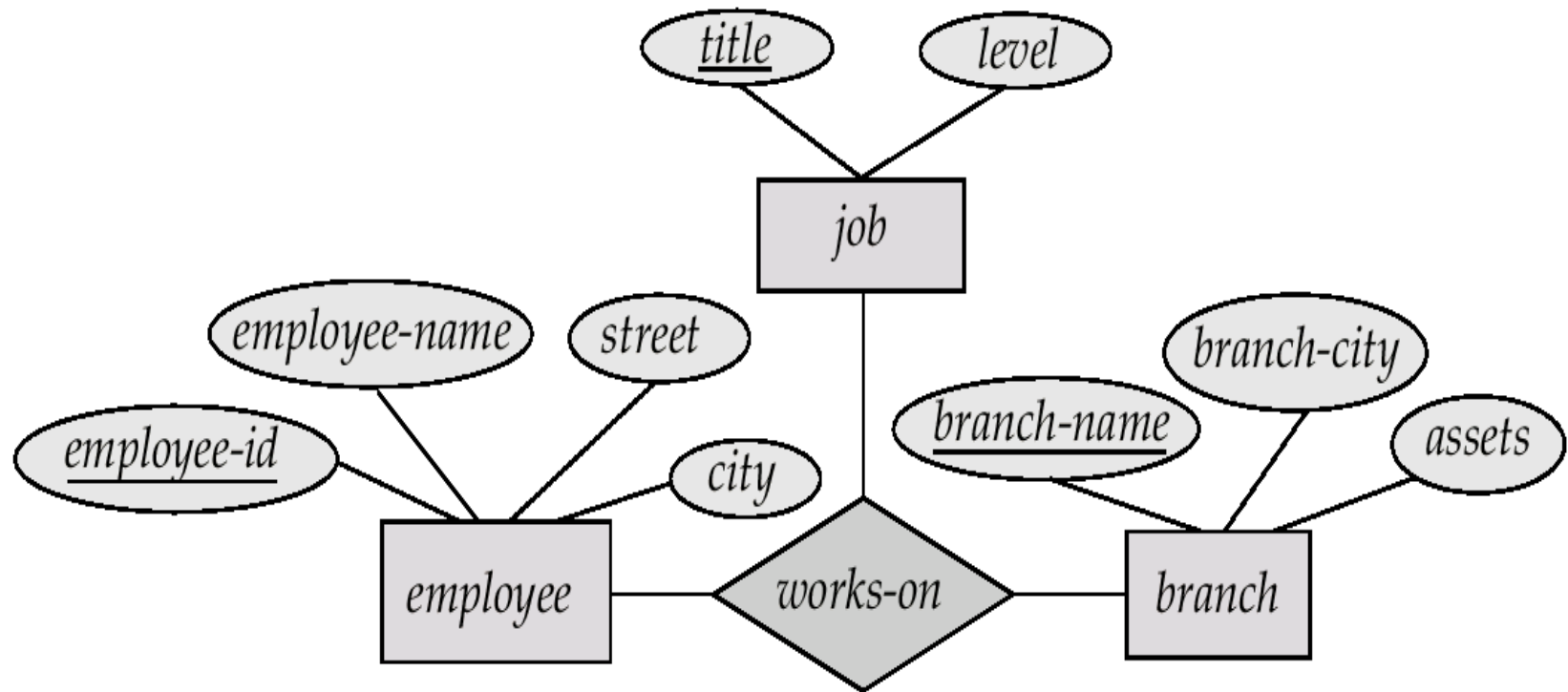
# Alternative Notation for relationship Constraints

- ❑ Alternative notation for **cardinality constraints** and **participation constraints**.
- ❑ E.g., 一个客户可以借多笔或0笔贷款，一笔贷款至少、至多属于一个客户。



# E-R Diagram with a Ternary Relationship

□ 一个银行职员在多个支行兼职，并承担不同类型的工作。



# Binary vs. Non-Binary Relationships

❑ **Some** relationships that appear to be non-binary may be better represented using binary relationships.

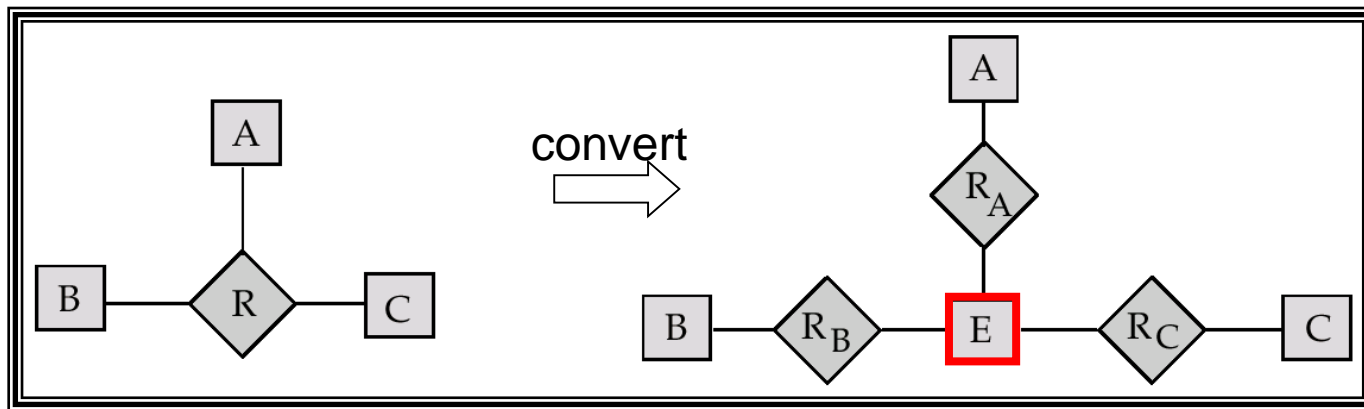
- E.g., a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*.

*parents(he, she, child) => father(he, child), mother(she, child)*

- Using two binary relationships allows partial information, e.g., only mother being know.
- But there are **some relationships** that are naturally non-binary, e.g., *works-on(employee, branch, job)*.

# Converting Non-Binary Relationships to Binary Form

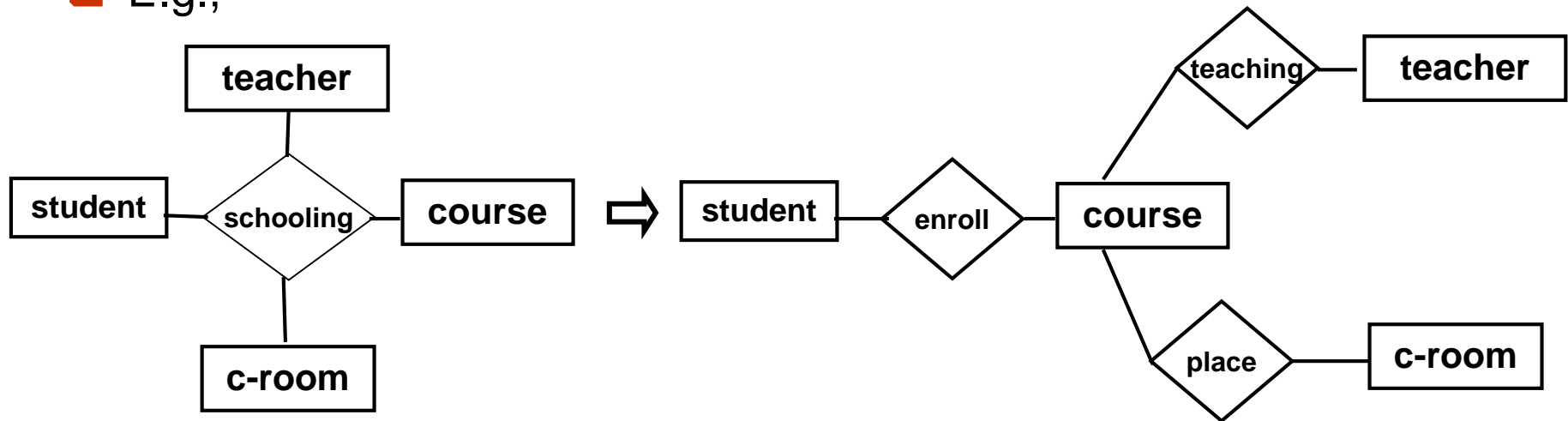
- ❑ In general, any non-binary relationship can be represented using binary relationships by creating an **artificial entity set**.
  - Replace non-binary relationship  $R$  between entity sets  $A$ ,  $B$ , and  $C$  by an entity set  $E$ , and three new relationship sets.
  - Create a special identifying attribute for  $E$ .
  - Add any attributes of  $R$  to  $E$ .
  - For each relationship  $(a_i, b_i, c_i)$  in  $R$ , create  $R_A, R_B, R_C$ .



# Converting Non-Binary Relationships to Binary Form (Cont.)

- ❑ In general, any non-binary relationship can be represented using binary relationships by creating an **artificial entity set**.
  - Replace non-binary relationship  $R$  between entity sets  $A$ ,  $B$ , and  $C$  by an entity set  $E$ , and three new relationship sets.
  - Create a special identifying attribute for  $E$ .
  - Add any attributes of  $R$  to  $E$ .
  - For each relationship  $(a_i, b_i, c_i)$  in  $R$ , create  $R_A, R_B, R_C$ .

❑ E.g.,



# Outline

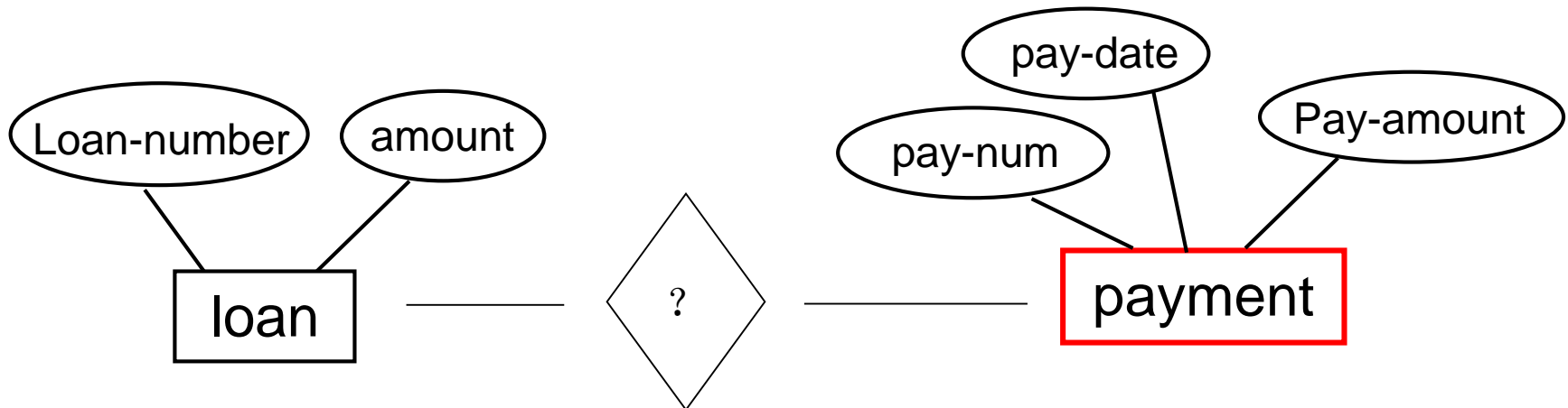
- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ **Weak Entity Sets**
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables



# Weak Entity Sets

❑ An entity set that does not have a primary key is referred to as a **weak entity set**.

- E.g., 还贷登记表 *payment*(*pay-num*, *pay-date*, *pay-amount*). 假设为了清楚起见, *pay-num* 按对应的每项贷款分别编号(都从1, 2, 3 ...开始), 这样, *pay-num* 就不是码, 并且该实体集没有码。故 *payment* 是弱实体集。 *pay-num* is discriminator or partial key (分辨符或部分码).
- C.f. weak entity set Vs. strong entity set.





# Weak Entity Sets (Cont.)

- ❑ The existence of a weak entity set **depends on** the existence of a identifying entity set or owner entity set (标识实体集或属主实体集).
  - E.g., *loan(loan-num, amount)* is an 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.
- ❑ The related relationship is called identifying relationship (标识性联系)
  - E.g., loan-payment

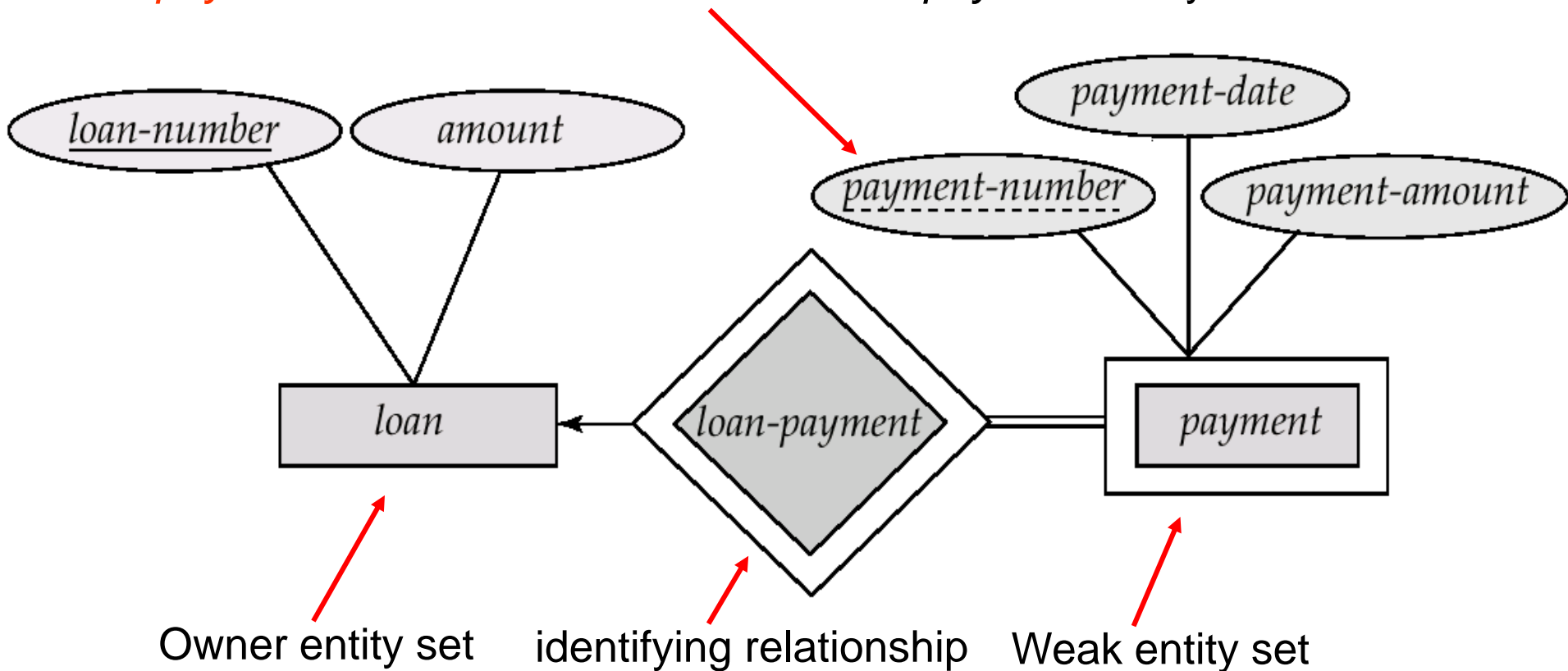
# Weak Entity Sets (Cont.)

- ❑ The **discriminator** or **partial key** (分辨符或部分码) of a weak entity set is the set of attributes that distinguishes among all those entities in a weak entity set that depend on one particular strong entity (e.g., payment-number).
- ❑ The **primary key of a weak entity set** is formed by the **primary key of the strong entity set** on which the weak entity set is existence dependent, **plus the weak entity set's discriminator**.

# Weak Entity Sets (Cont.)

Primary key for payment(loan-number, payment-number)

*payment-number* is discriminator of the *payment* entity set



# Weak Entity Sets (Cont.)

- ❑ Note: the primary key of the strong entity set is **not explicitly stored with the weak entity set**, since it is implicit in the identifying relationship.
- ❑ If *loan-number* were explicitly stored, *payment* could be made a strong entity, but then the relationship between *payment* and *loan* would be duplicated by an implicit relationship defined by the attribute *loan-number* common to *payment* and *loan*.

# Outline

- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ **Extended E-R Features**
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables



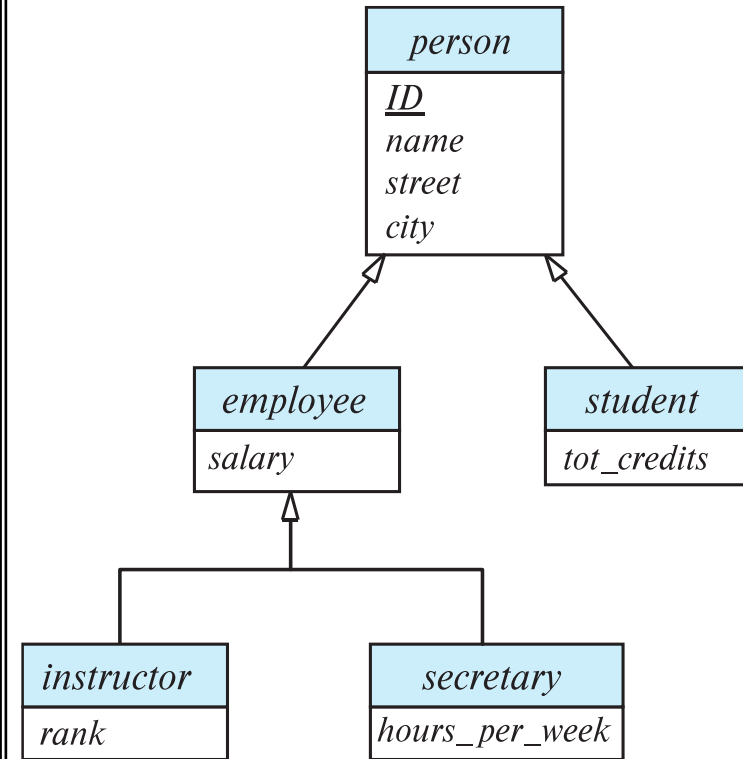
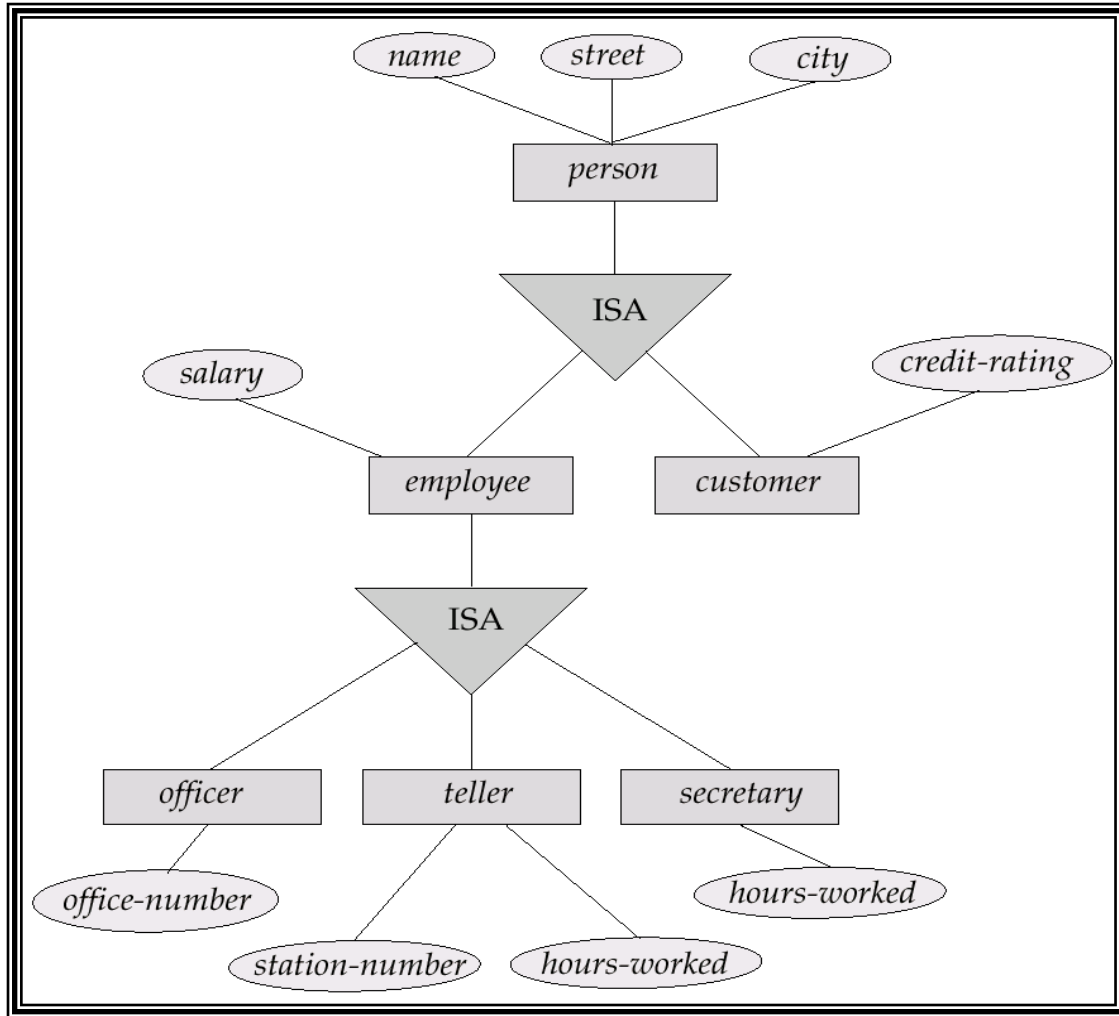
# Extended E-R features

## ❑ Stratum of the entity set

### ➤ Specialization (特殊化、具体化)

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- 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



# Extended E-R features (Cont.)

## ❑ Stratum of the entity set (Cont.)

### ➤ Generalization (泛化、普遍化)

- A **bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.



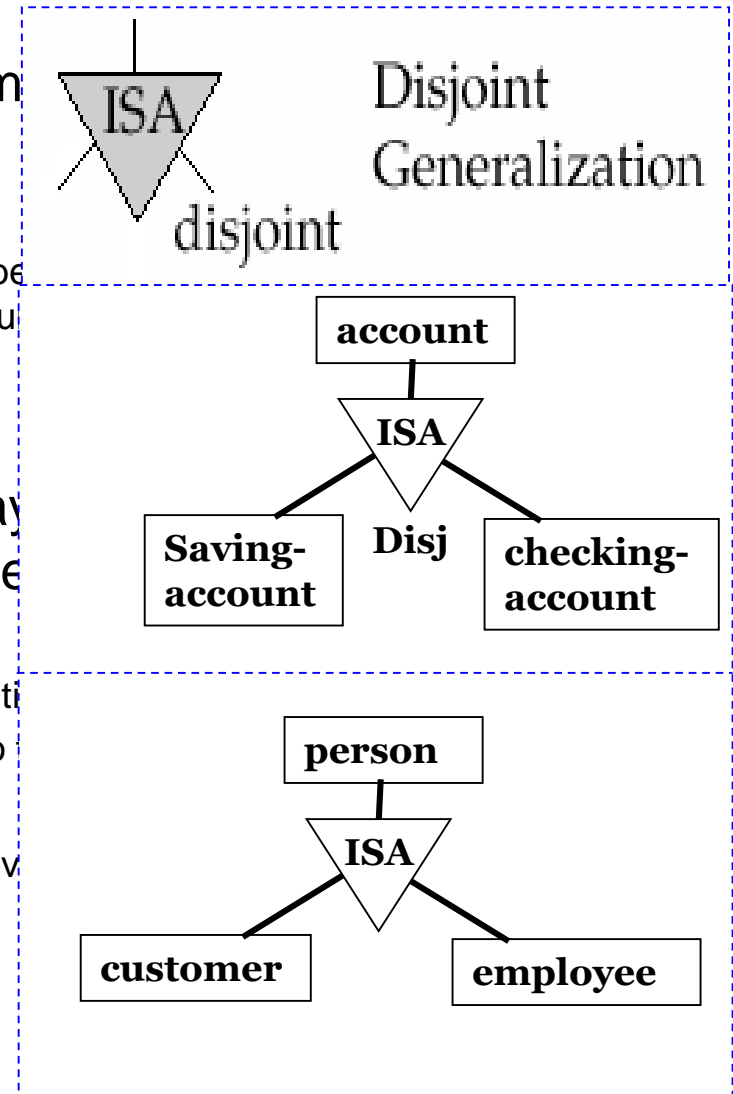
# Design Constraints on a Specialization / Generalization

## ❑ Constraint on which entities can be members of an entity set.

- Condition-defined (条件定义的)
  - E.g., (1) All customers over 65 years are members of *senior citizen ISA person*; (2) account to saving account
- User-defined, e.g., employee to teams

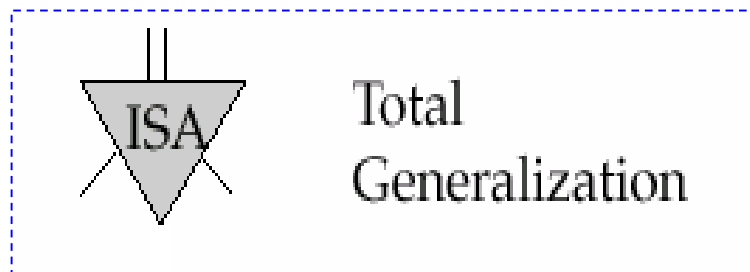
## ❑ Constraint on whether or not entities may belong to a lower-level entity set within a single generalization

- Disjoint (不相交)
  - An entity can belong to only one lower-level entity set
  - Noted in E-R diagram by writing *disjoint* next to the ISA triangle
- Overlapping (可重叠)
  - An entity can belong to more than one lower-level entity set
  - E.g.,

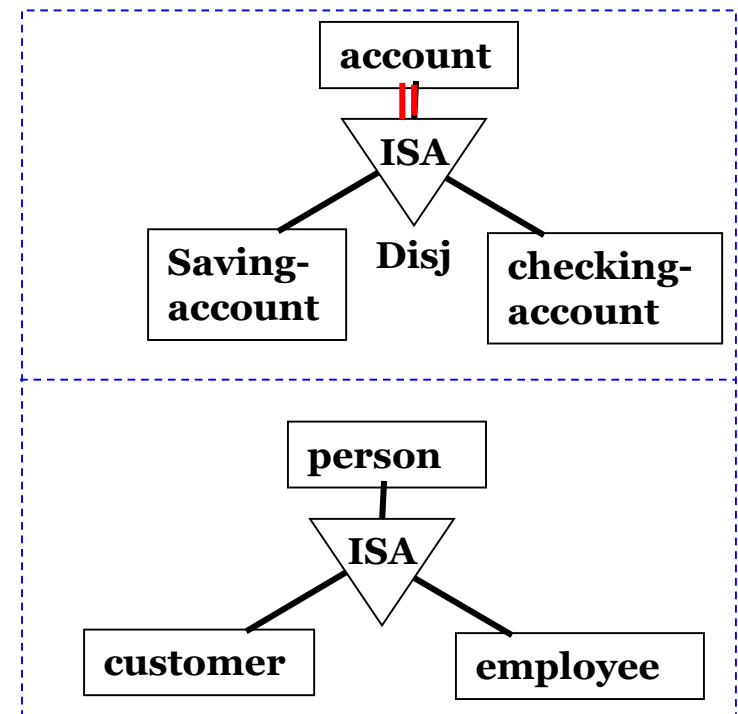


# Design Constraints on a Specialization / Generalization (Cont.)

- ❑ Completeness constraint (完全性约束) -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - Total: an entity must belong to one of the lower-level entity sets.
  - Partial: an entity need not belong to one of the lower-level entity sets.



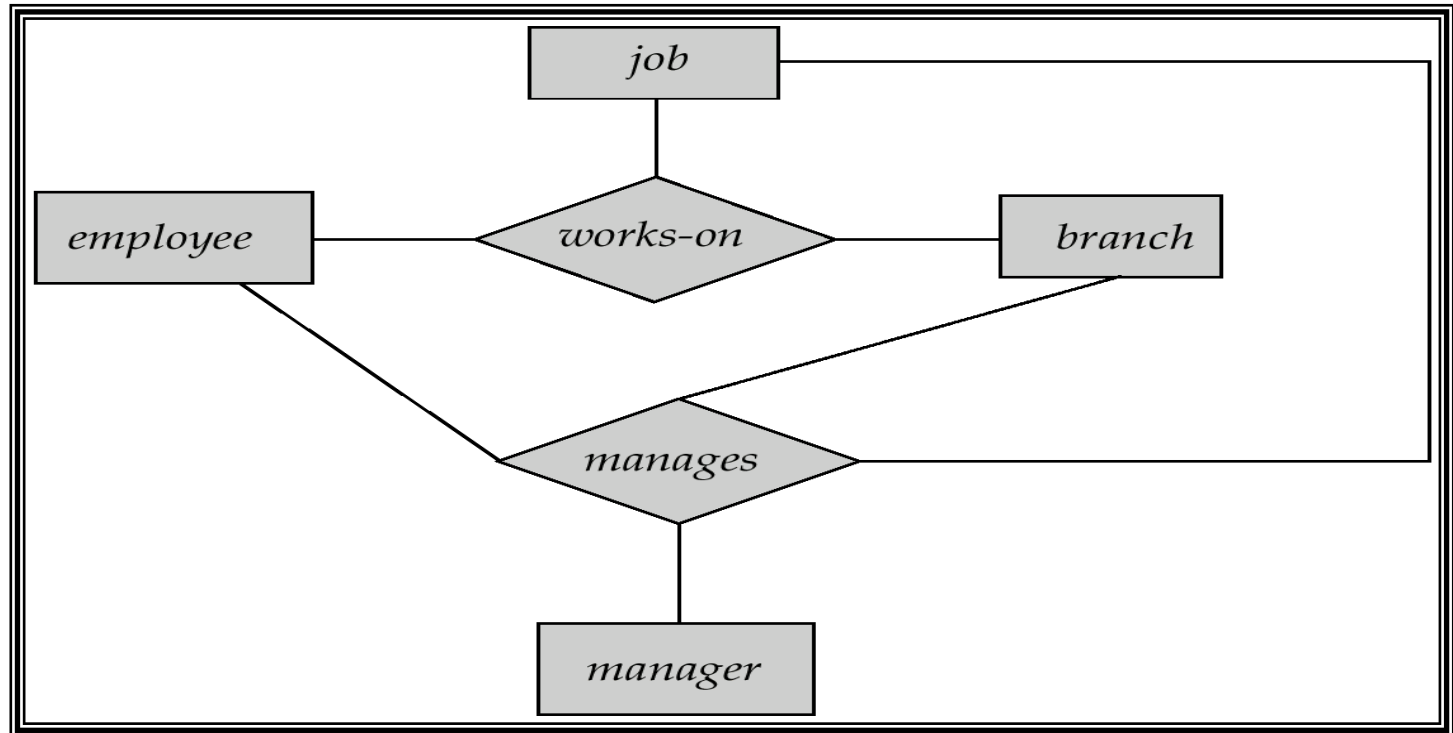
(完全泛化)



(部分泛化)

# Aggregation

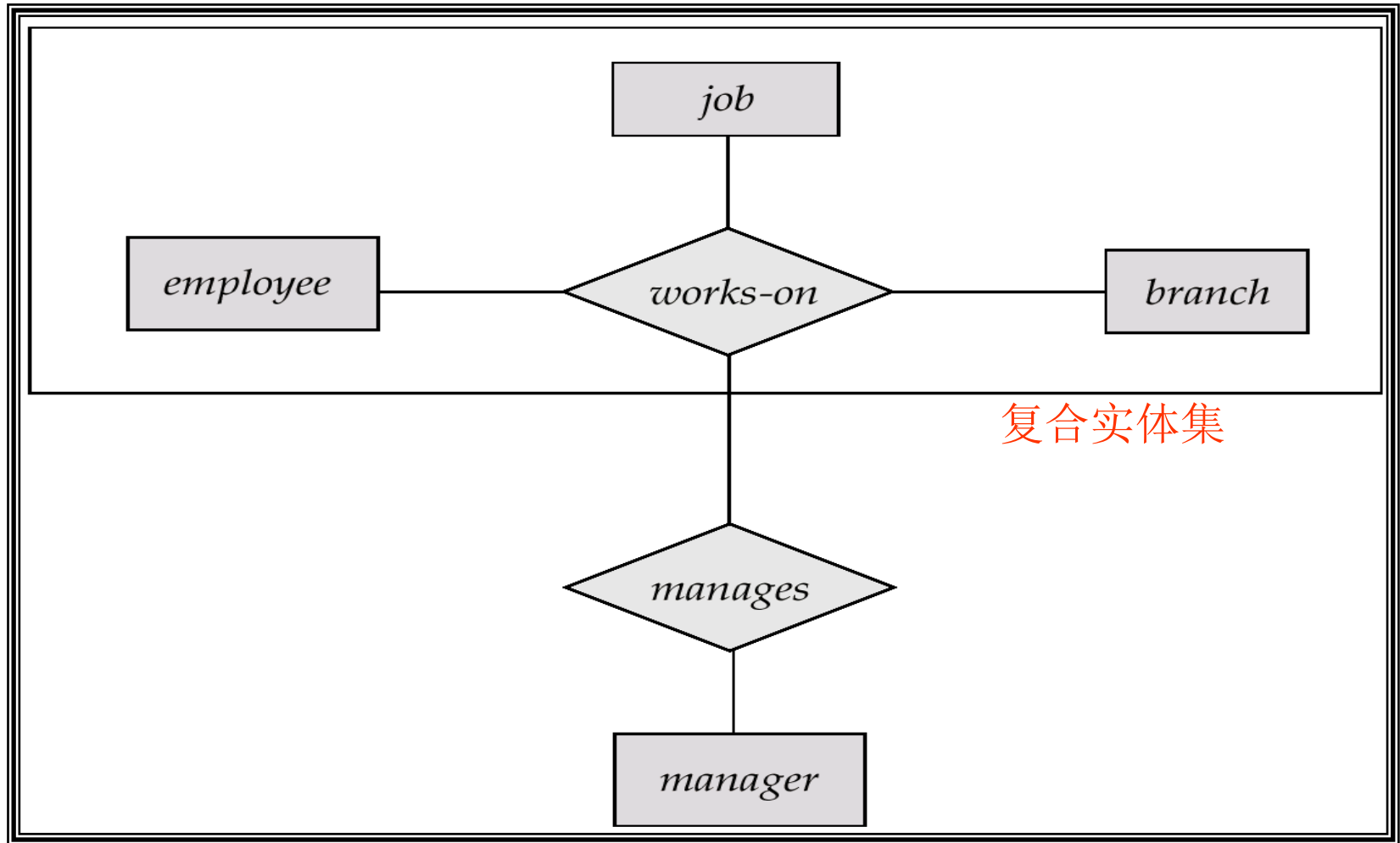
- ❑ How to express relationships among relationships ?
- ❑ Consider the ternary relationship *works-on*, which we saw earlier. Suppose we want to record managers for tasks performed by an employee at a branch.



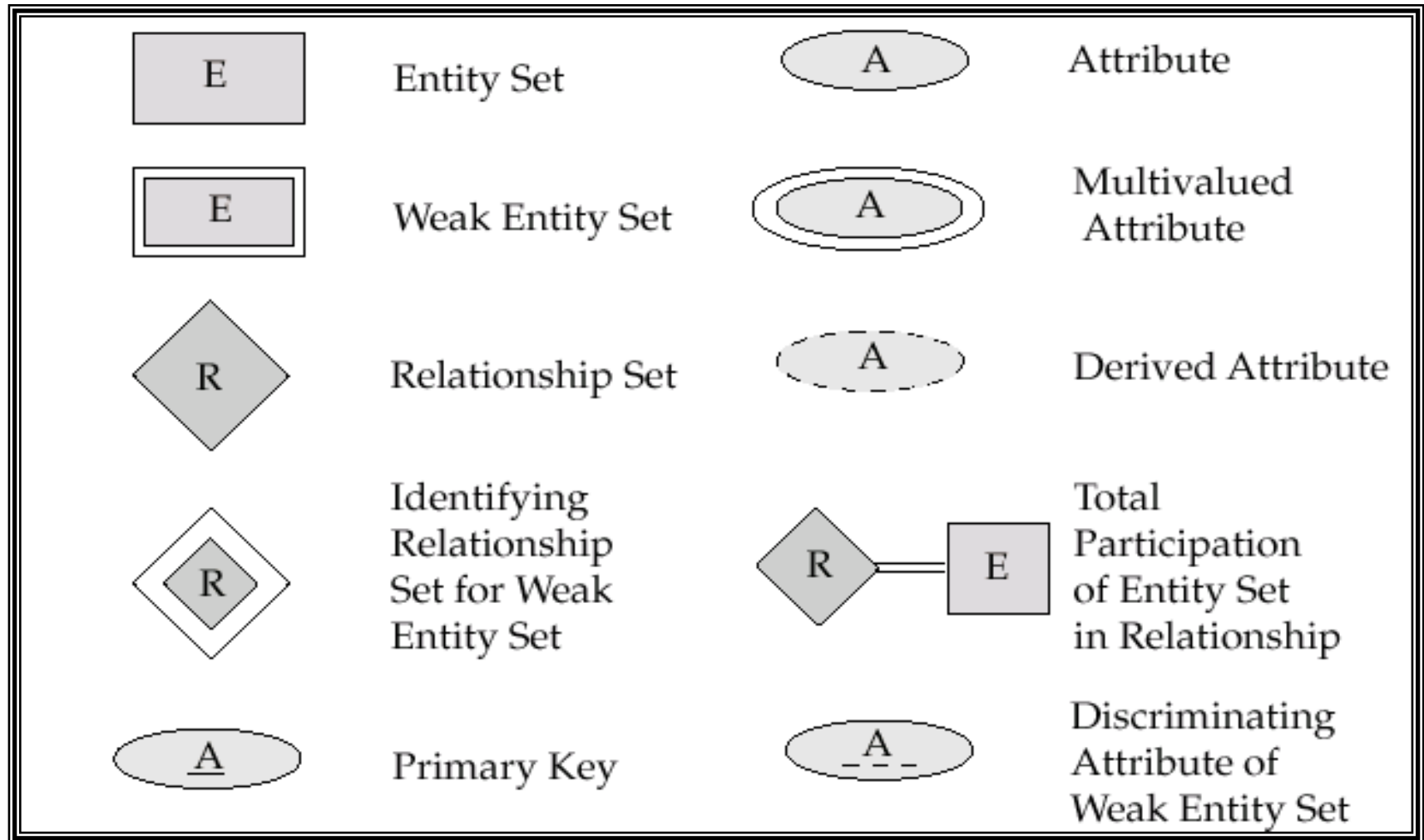
# Aggregation (Cont.)

- ❑ Relationship sets *works-on* and *manages* represent overlapping information.
  - Every *manages* relationship corresponds to a *works-on* relationship.
  - However, some *works-on* relationships may not correspond to any *manages* relationships.
    - So we can't discard the *works-on* relationship.
- ❑ Eliminate this redundancy via *aggregation*.
  - Treat relationship as an abstract entity.
  - Allows relationships between relationships.
  - Abstraction of relationship into new entity.
- ❑ Without introducing redundancy, the following diagram represents:
  - An employee works on a particular job at a particular branch.
  - An employee, branch, job combination may have an associated manager.

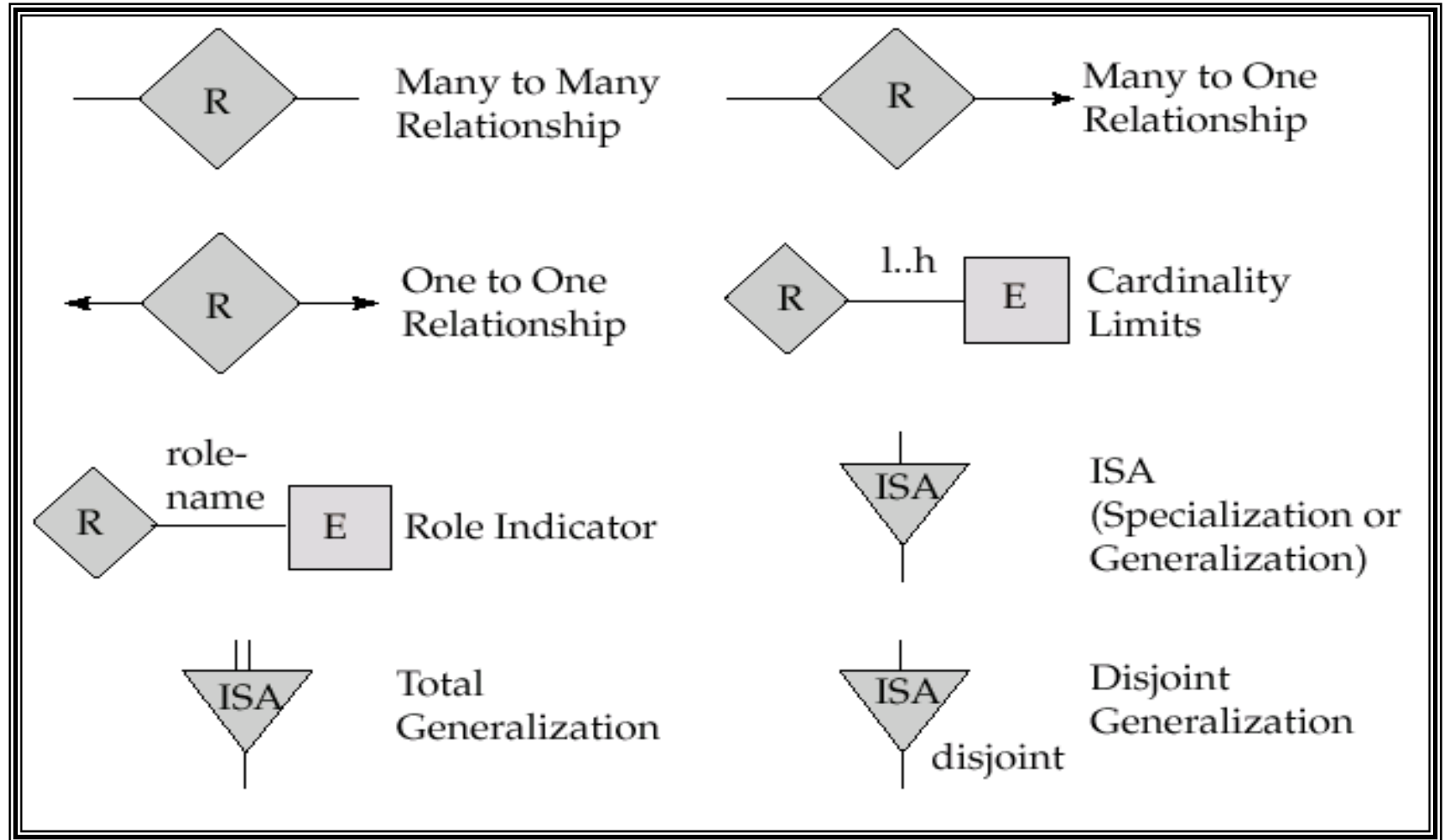
# E-R Diagram With Aggregation



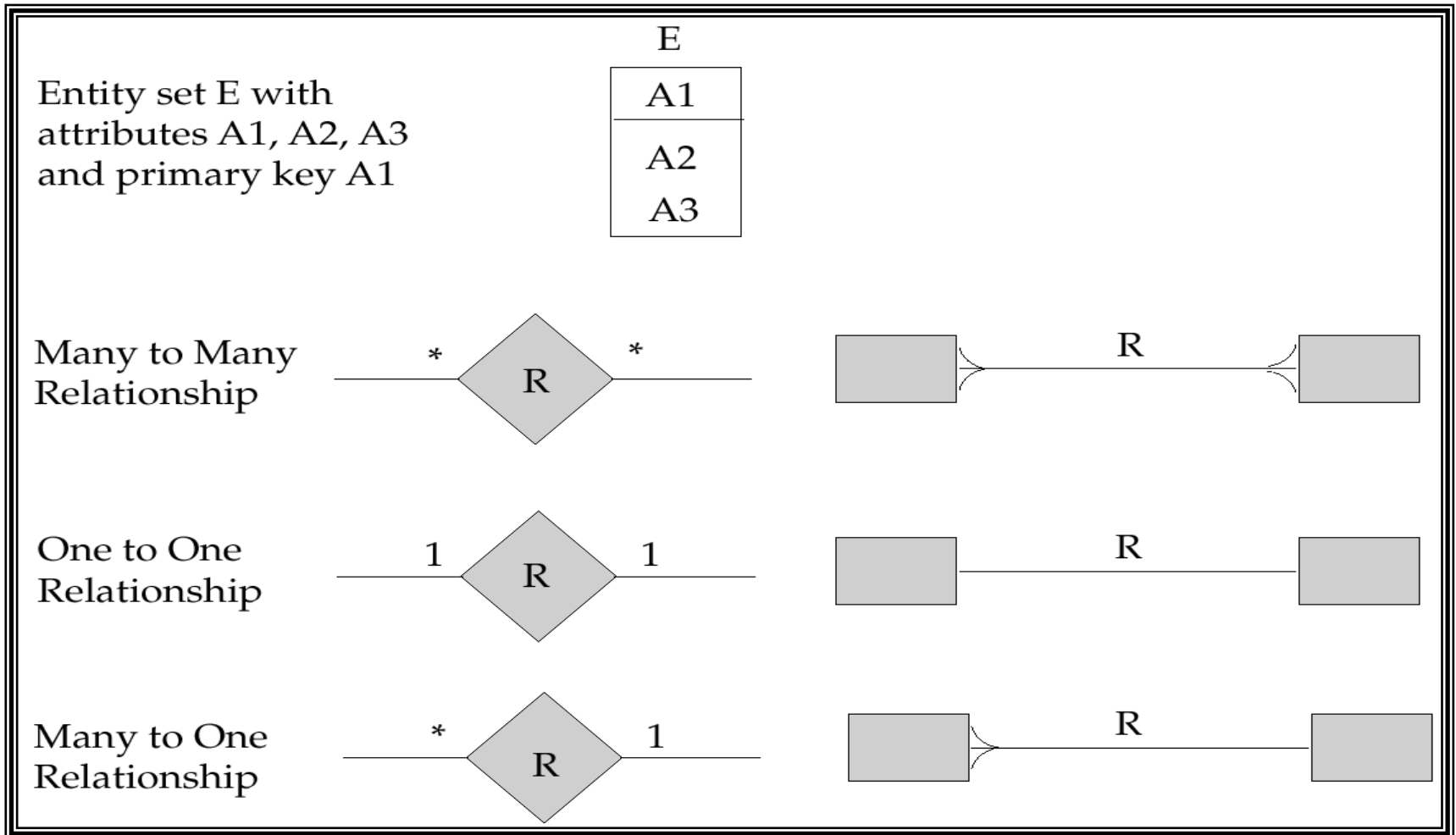
# Summary of Symbols Used in E-R Notation



# Summary of Symbols (Cont.)

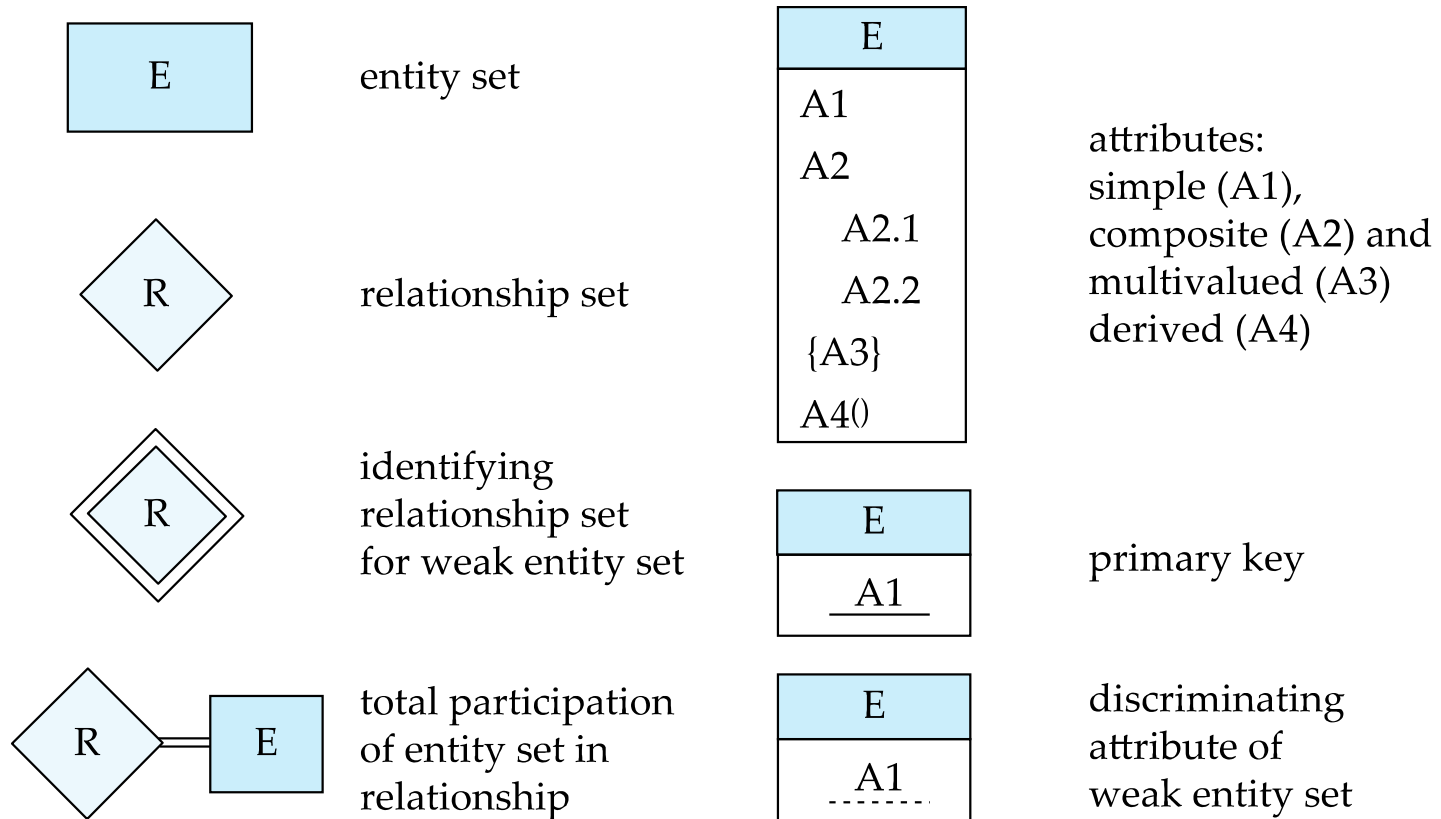


# Alternative E-R Notations

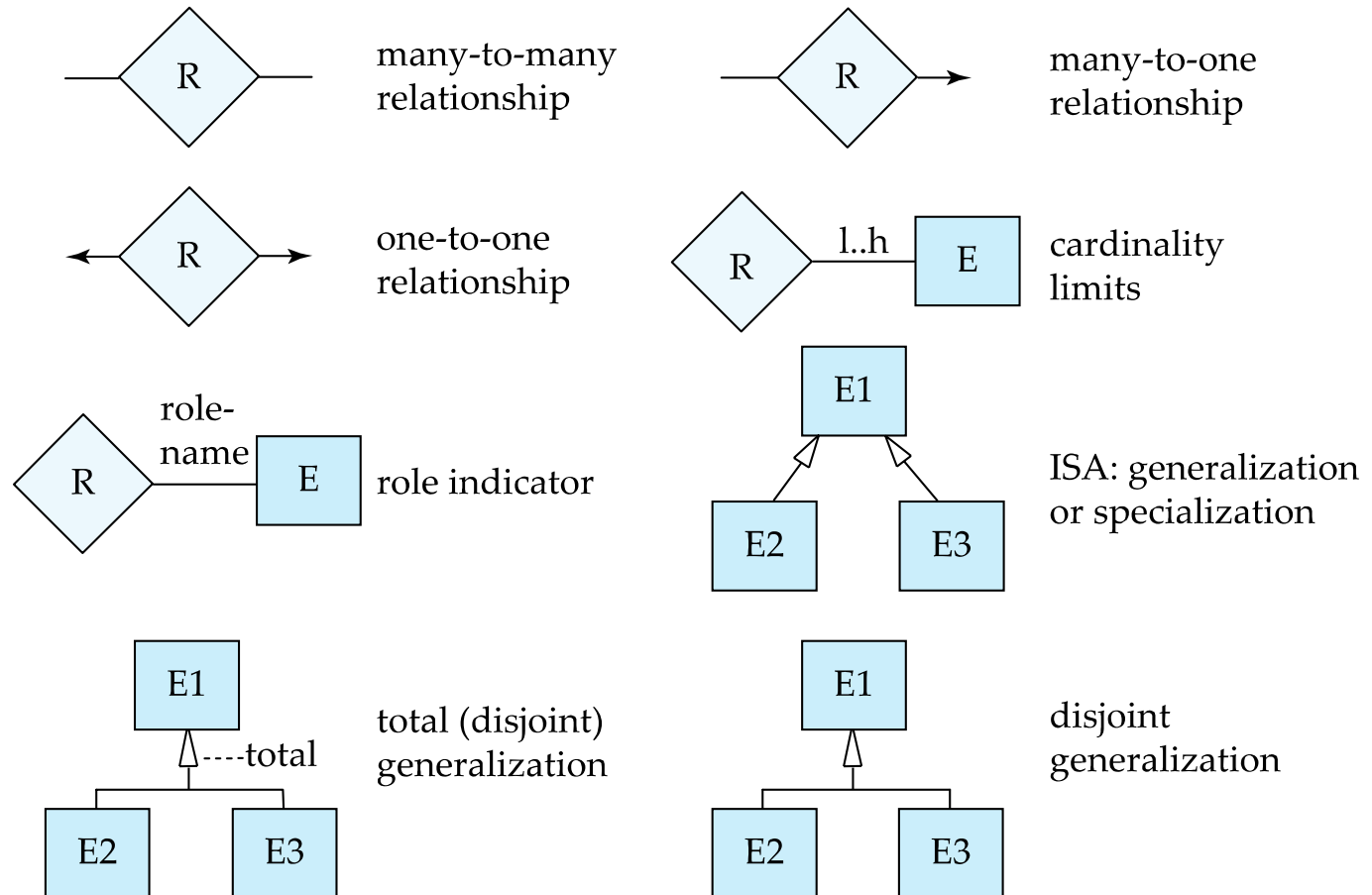




# Alternative E-R Notations



# Alternative E-R Notations



# Outline

- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ **Design of an E-R Database Schema**
- ❑ Reduction of an E-R Schema to Tables



# Design of an E-R Database Schema



## ❑ Requirement analysis

- What data, applications, and operations needed.

## ❑ Conceptual database design

- A high-level description of data, constraints using **E-R model** or a similar high level data model.

## ❑ Logical database design

- Convert the conceptual design into DB schema --- **tables**
- Schema refinement: **Normalization of relations** --- Check relational schema for redundancies and related anomalies.

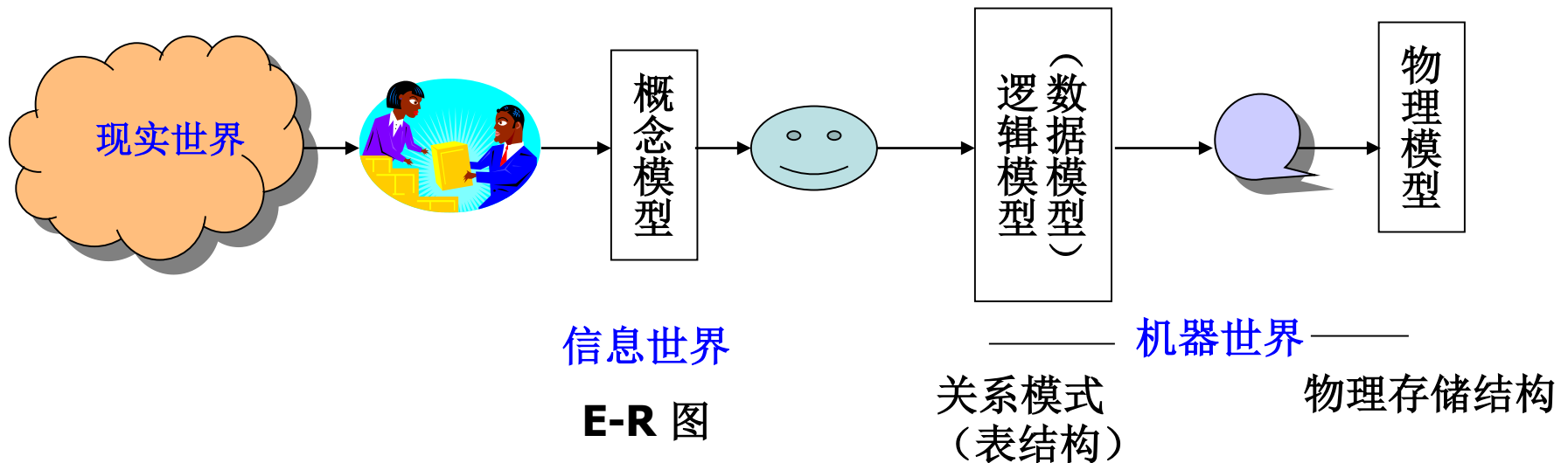
## ❑ Physical database design

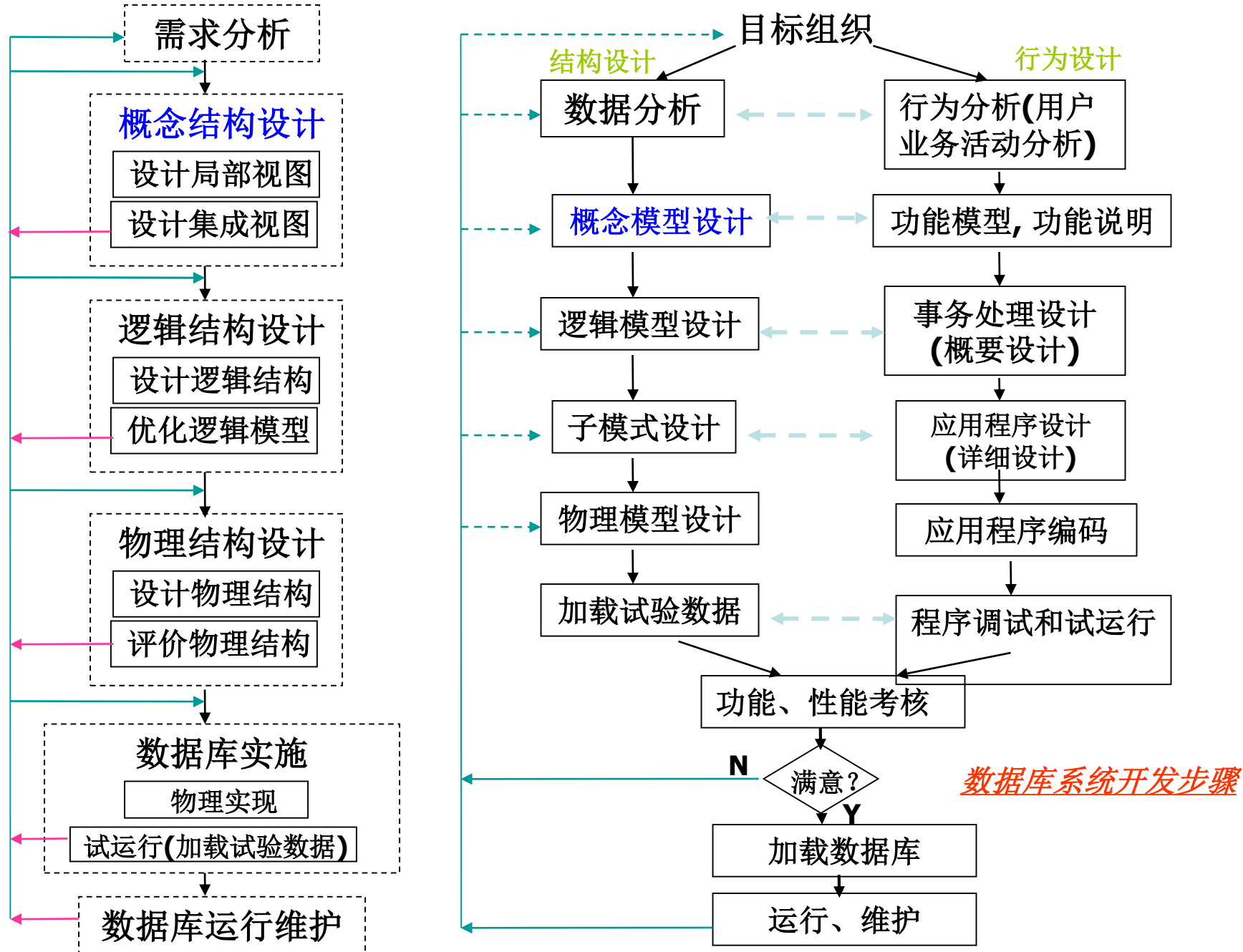
- Indexing, clustering and database tuning.

# Complementarity

## Steps of database design

### 数据库的设计步骤





# E-R Design Decisions

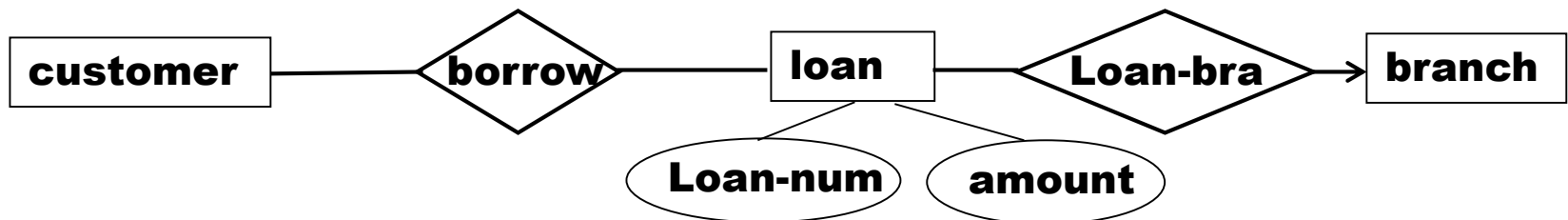
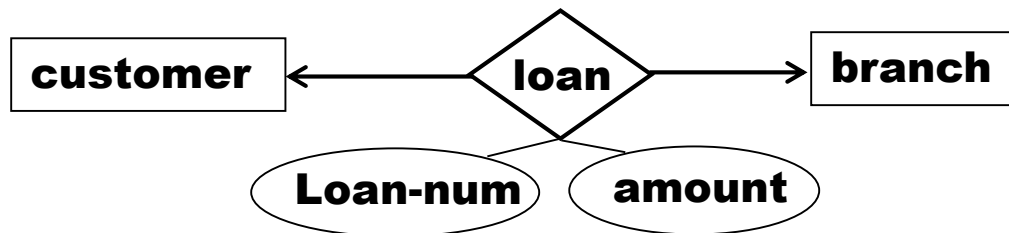
## ❑ (1) Use an **attribute** or **entity set** to represent an object?

- E.g. 1: *employee(emp-id, emp-name, ..., **phone**)*
- 优点: 简单; 但多个电话怎么处理? 电话的其他属性?
- E.g. 2: *employee(emp-id, emp-name, ...);*  
*phone(phone-num, location, type, color);*  
*emp-phone(emp-id, phone-num)*
- 若一个对象只对其名字及单值感兴趣, 则可作为属性, 如**性别**; 若一个对象除名字外, 本身还有其他属性需描述, 则该对象应定义为实体集。如**电话**, **部门**.
- 一个对象不能同时作为实体和属性.
- 一个**实体集**不能与另一实体集的**属性**相关联, 只能实体与实体相联系.

# E-R Design Decisions (Cont.)

## ❑ (2) Use it as an **entity set** or a **relationship set**?

- E.g., *enrolled*, *borrower*, *depositor*
- **Relationship set** --- to describe an **action** that occurs between entities (二个对象之间发生的动作 --- 用“**relationship set**”表示).
- The mapping cardinality will effect the matter.  
Considering the *branch*, *loan*, *customer*, ...





# E-R Design Decisions (Cont.)

- ❑ (3) Use it as an **attribute of an entity** or a **relationship?**, e.g.,  
*student*(*sid*, *name*, *sex*, *age*, ..., *supervisor-id*, *supervisor-name*,  
*supervisor-position*, ..., *class*, *monitor*)

- 要从对象的语义独立性和减少数据冗余方面考虑

- student*(*sid*, *name*, *sex*, *age*, ...);

- supervisor*(*sup-id*, *name*, *position*, ...);

- class*(*classno*, *specialty*, *monitor*, *stu-num*);

- stu-class*(*sid*, *classno*);

- stu-sup*(*sid*, *sup-id*, *from*, *to*);

# E-R Design Decisions (Cont.)

- ❑ (4) The use of a **ternary** or  **$n$ -ary relationship** versus a pair of **binary relationships**.
- ❑ \*(5) The use of **a strong** or **weak entity set**.
- ❑ \*(6) The use of **specialization/generalization** – contributes to modularity in the design (有助于模块化).
- ❑ \*(7) The use of **aggregation** – can group a part of E-R diagram into a single entity set, and treat it as a single unit without concern for the details of its internal structure.

# Example: Design of Banking Database

## ❑ Get system requirements

- Branch, customer, employee, loan
- Account: *saving account*, *checking account*

## ❑ Entity sets design

- *Branch*(branch-name, branch-city, assets)
- *Customer*(cust-id, cust-name, cust-street, cust-city)
- *Employee*(emp-id, emp-name, phone, salary)
- *Account*(acc-number, balance),  
*Saving-acc*(interest-rate),  
*Checking-acc*(overdraft-amount)
- *Loan*(loan-number, amount)  
*Payment*(pay-number, pay-date, pay-amount)

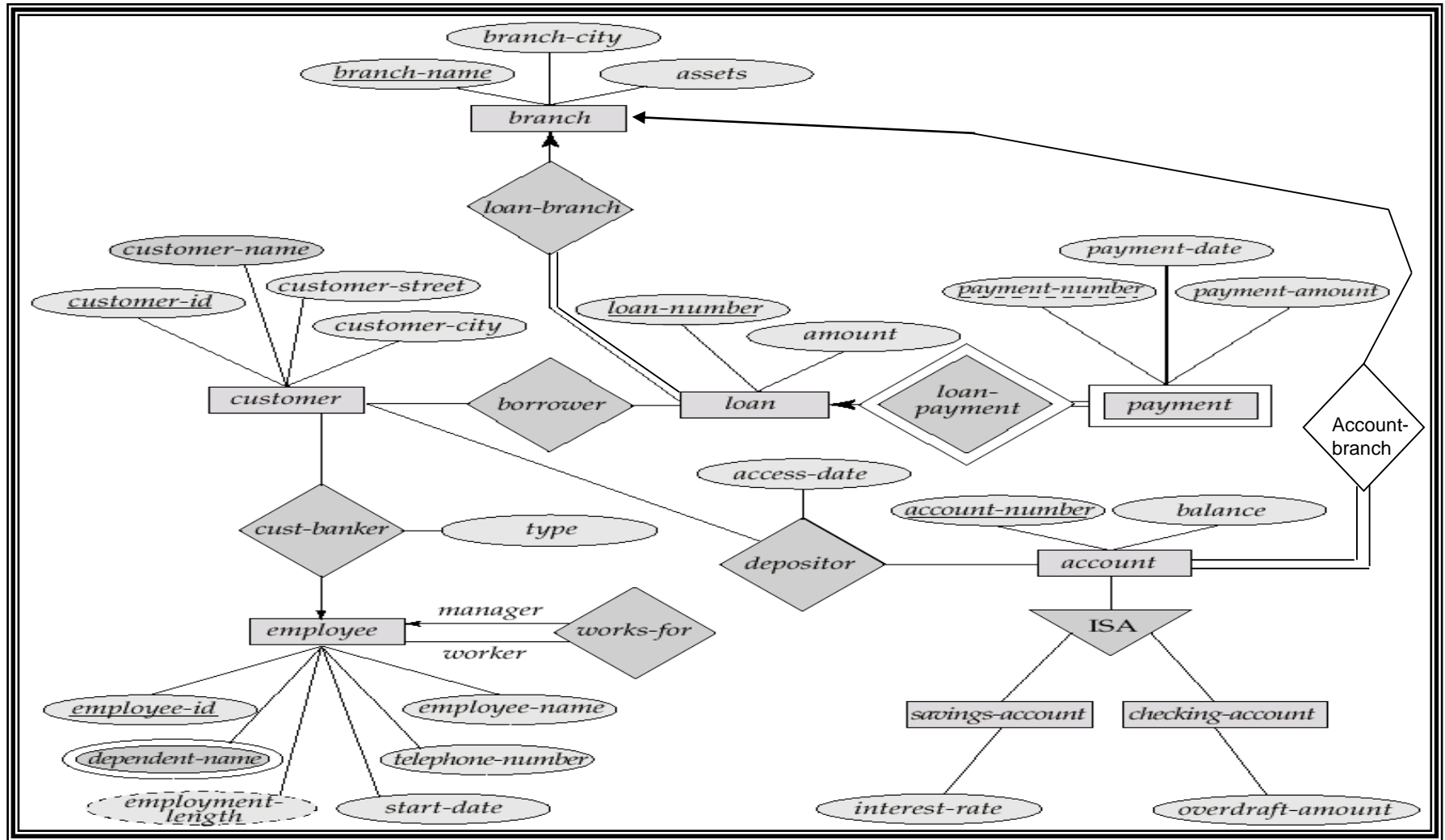
# Example: Design of Banking Database (Cont.)

## ❑ Relationship sets design

- **Borrower**: customer ---- loan
- **Depositor**: account ---- customer (account-date)
- **Loan-branch**: loan → branch
- **Account-branch**: account → branch
- **Cust-banker**: customer → employee (type)
- **Works-for**: manager ← worker

## ❑ E-R diagram

# E-R Diagram for a Banking Enterprise



# Outline

- ❑ Entity Sets
- ❑ Relationship Sets
- ❑ Keys
- ❑ E-R Diagram
- ❑ Weak Entity Sets
- ❑ Extended E-R Features
- ❑ Design of an E-R Database Schema
- ❑ Reduction of an E-R Schema to Tables



# Reduction of an E-R Schema to Tables

- ❑ Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.
- ❑ A database which conforms to an E-R diagram can be represented by a collection of tables.
- ❑ For each entity set and relationship set, there is a unique table which is assigned the name of the corresponding entity set or relationship set.

# Representing Entity Sets as Tables

- ❑ A strong entity set  $\Rightarrow$  to a table with the same attributes.
- ❑ *customer(customer-id, cust-name, cust-street, cust-city);*
- ❑ *branch(branch-name, branch-city, assets);*
- ❑ *account(account-number, balance);*
- ❑ *loan(loan-number, amount);*
- ❑ *employee(employee-id, employee-name, phone, start-date);*



# Entity Set with Composite Attributes

- ❑ Composite attributes are flattened out by creating a separate attribute for each component attribute.
  - E.g., given entity set *customer* with composite attribute *name* with component attributes *first-name* and *last-name*. The table corresponding to the entity set has two attributes:

*name.first-name* and *name.last-name*  
or *first-name* and *last-name*

*customer*(*customer-id*, *first-name*, *last-name*, *cust-street*, *cust-city*)

# Entity Set with Multivalued Attributes

❑ A multivalued attribute  $M$  of an entity  $E$  is represented by a separate table  $EM$ :

- E.g., multivalued attribute *dependent-names of employee* is represented by a table
- *employee(emp-id, ename, sex, age, dependent-names)*



- *employee(emp-id, ename, sex, age),*
- *employee-dependent-names(emp-id, dependent-name)*
- Each value of the multivalued attribute maps to a separate row of the table  $EM$ 
  - E.g., an employee entity with primary key *John* and dependents *Johnson* and *Johndoter* maps to two rows:  
(*John*, *Johnson*) and (*John*, *Johndoter*) in relation *employee-dependent-names*.

# Representing Weak Entity Sets

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set.

<i>loan-number</i>	<i>payment-number</i>	<i>payment-date</i>	<i>payment-amount</i>
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

*payment*(*loan-number*, *payment-number*, *payment-date*, *payment-amount*)

# Representing Relationship Sets as Tables

- ❑ A relationship set is represented as a table with columns for the primary keys of the two participating entity sets, (which are foreign keys here) and any descriptive attributes of the relationship set itself.
- ❑ (a) tables for many to many relationship sets:  
*borrower(customer-id, loan-number);*  
*depositor(customer-id, account-number, access-date)*

The two attributes corresponding to the two entity sets become the primary key of the table.

*borrower*

<i>customer-id</i>	<i>loan-number</i>
019-28-3746	L-11
019-28-3746	L-23
244-66-8800	L-93
321-12-3123	L-17
335-57-7991	L-16
555-55-5555	L-14
677-89-9011	L-15
963-96-3963	L-17

# Representing Relationship Sets as Tables (Cont.)

- ❑ (b) tables for many to one relationship sets:

*loan-branch(loan-number, branch-name);*

*account-branch(account-number, branch-name);*

*cust-banker(customer-id, employee-id, type);*

*works-for(employee-id, manager);*

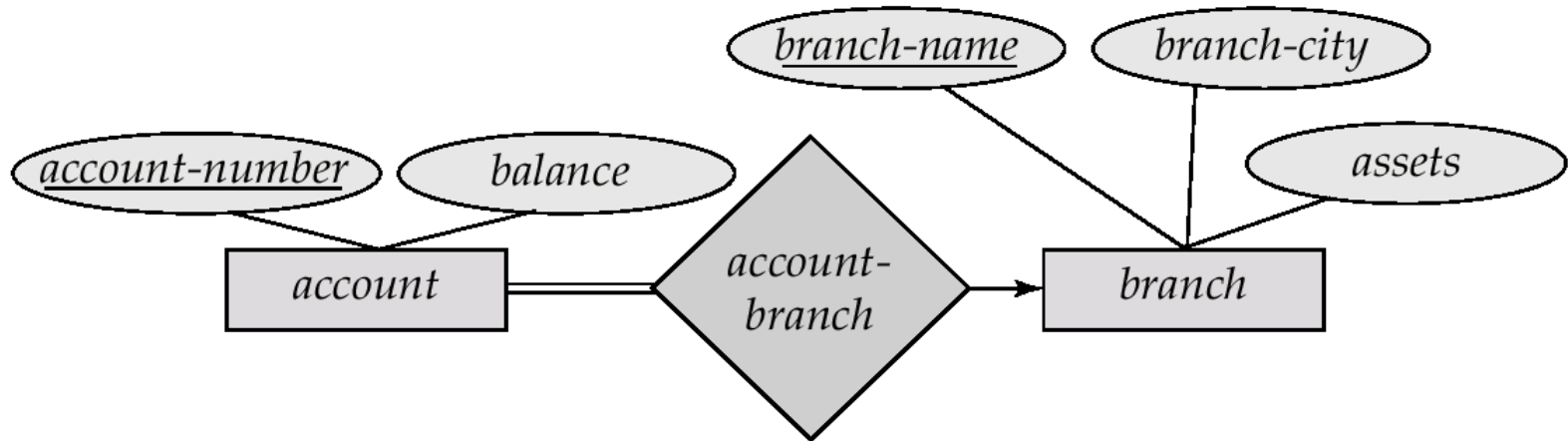
*loan-payment(loan-number, payment-number)*

The attribute corresponding to the entity set on the “many” side becomes the primary key of the table.

- ❑ (c) tables for one to one relationship sets: like (b)

# Redundancy of Tables

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the one side (对1:n联系, 可将“联系”所对应的表, 合并到对应“多”端实体的表中).



*account*(*account-number*, *balance*); *branch*(*branch-name*, *branch-city*, *assets*);

*account-branch*(*account-number*, *branch-name*)


*account*(*account-number*, *branch-name*, *balance*)

# Redundancy of Tables (Cont.)

- ❑ If participation is **partial** on the many side, replacing a table by an extra attribute in the relation corresponding to the “many” side could result in **null values**.
  - E.g., *cust-banker(customer-id, employee-id, type)*; 但有的customer没有banker, 则合并之后得:  
*Customer(customer-id, cust-name, cust-street, cust-city, banker-id, type)* , 导致 *Customer* 中有些元组的 *banker-id*、*type* 为 null)。
- ❑ For **one-to-one relationship sets**, **either side** can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets.

 **account-branch(account-number, branch-name);**  
**account(account-number, balance);**

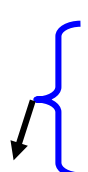
---

 **loan-branch(loan-number, branch-name);**  
**loan(loan-number, amount);**  
**loan(loan-number, branch-name, amount);**

---

 **cust-banker(customer-id, employee-id, type);**  
**customer(customer-id, customer-name, customer-street,  
customer-city);**

---

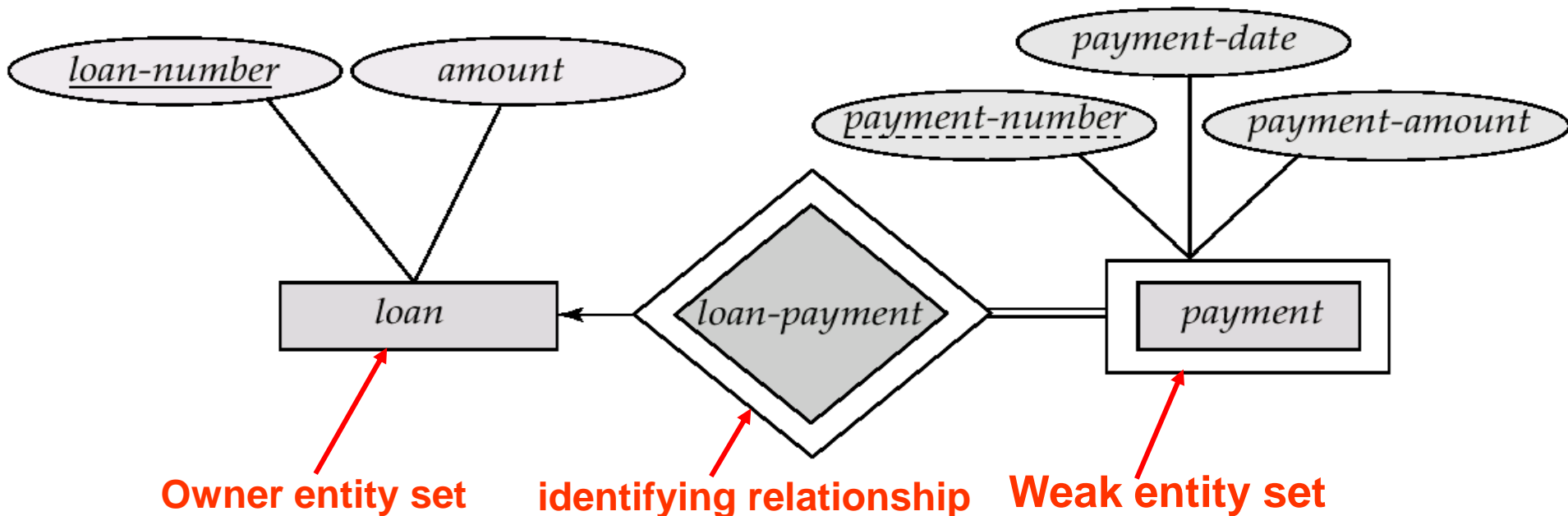
 **works-for(employee-id, manager);**  
**employee(employee-id, employee-name, phone, start-  
date);**  
**employee(employee-id, employee-name, phone, start-date,  
manager);**

---



# Redundancy of Tables (Cont.)

- ❑ (b) The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is **redundant** (联系弱实体集及其标识性实体集的联系集对应的表是冗余的，即对应**identifying relationship**的表是多余的。).
  - E.g., The payment table already contains the information that would appear in the loan-payment table (i.e., the columns loan-number and payment-number).



Payment(loan-number, payment-number, payment-date, paymenty-amount)



<i>loan-number</i>	<i>payment-number</i>	<i>payment-date</i>	<i>payment-amount</i>
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

Loan-payment (loan-number, payment-number )

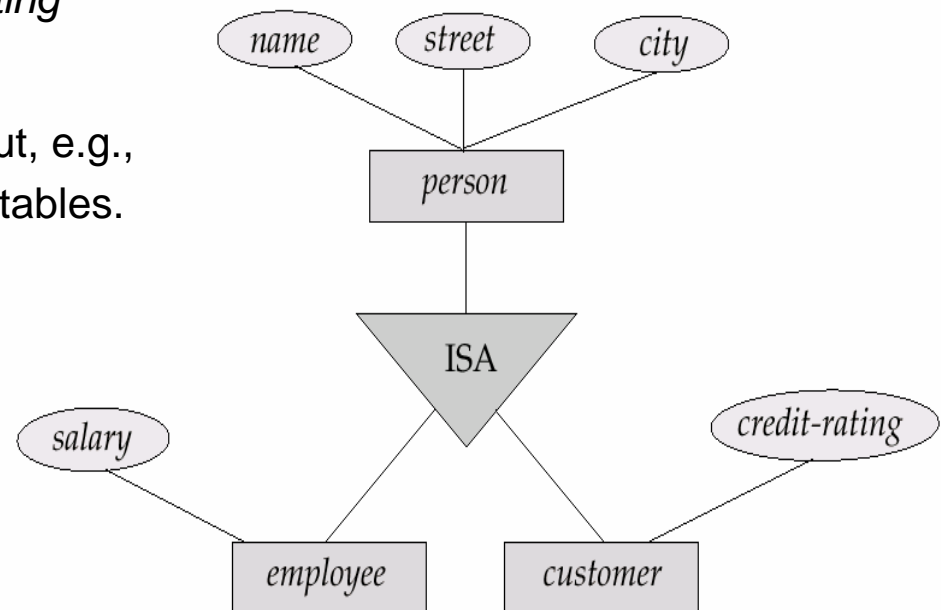
# Representing Specialization as Tables

## ❑ Method 1:

- Form a table for the higher level entity.
- Form a table for each lower level entity set, including primary key of higher level entity set and local attributes.

table	table attributes
person	<u>person-id</u> , name, street, city
customer	<u>person-id</u> , credit-rating
employee	<u>person-id</u> , salary

- **Drawback:** getting information about, e.g., *employee*, requires accessing two tables.



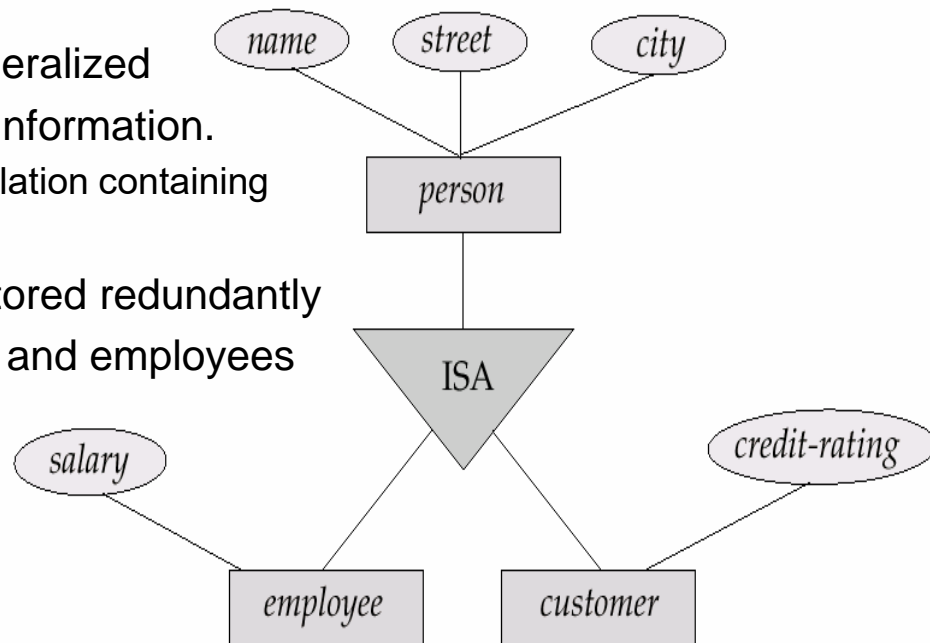
# Representing Specialization as Tables (Cont.)

## ❑ Method 2:

- Form a table for each entity set with all local and inherited attributes.

table	table attributes
<i>person</i>	<u><i>person-id</i></u> , <i>name</i> , <i>street</i> , <i>city</i>
<i>customer</i>	<u><i>person-id</i></u> , <i>name</i> , <i>street</i> , <i>city</i> , <i>credit-rating</i>
<i>employee</i>	<u><i>person-id</i></u> , <i>name</i> , <i>street</i> , <i>city</i> , <i>salary</i>

- If specialization is **total**, table for generalized entity (*person*) not required to store information.
  - *person* can be defined as a “view” relation containing union of specialization tables
- **Drawback:** *street* and *city* may be stored redundantly for persons who are both customers and employees



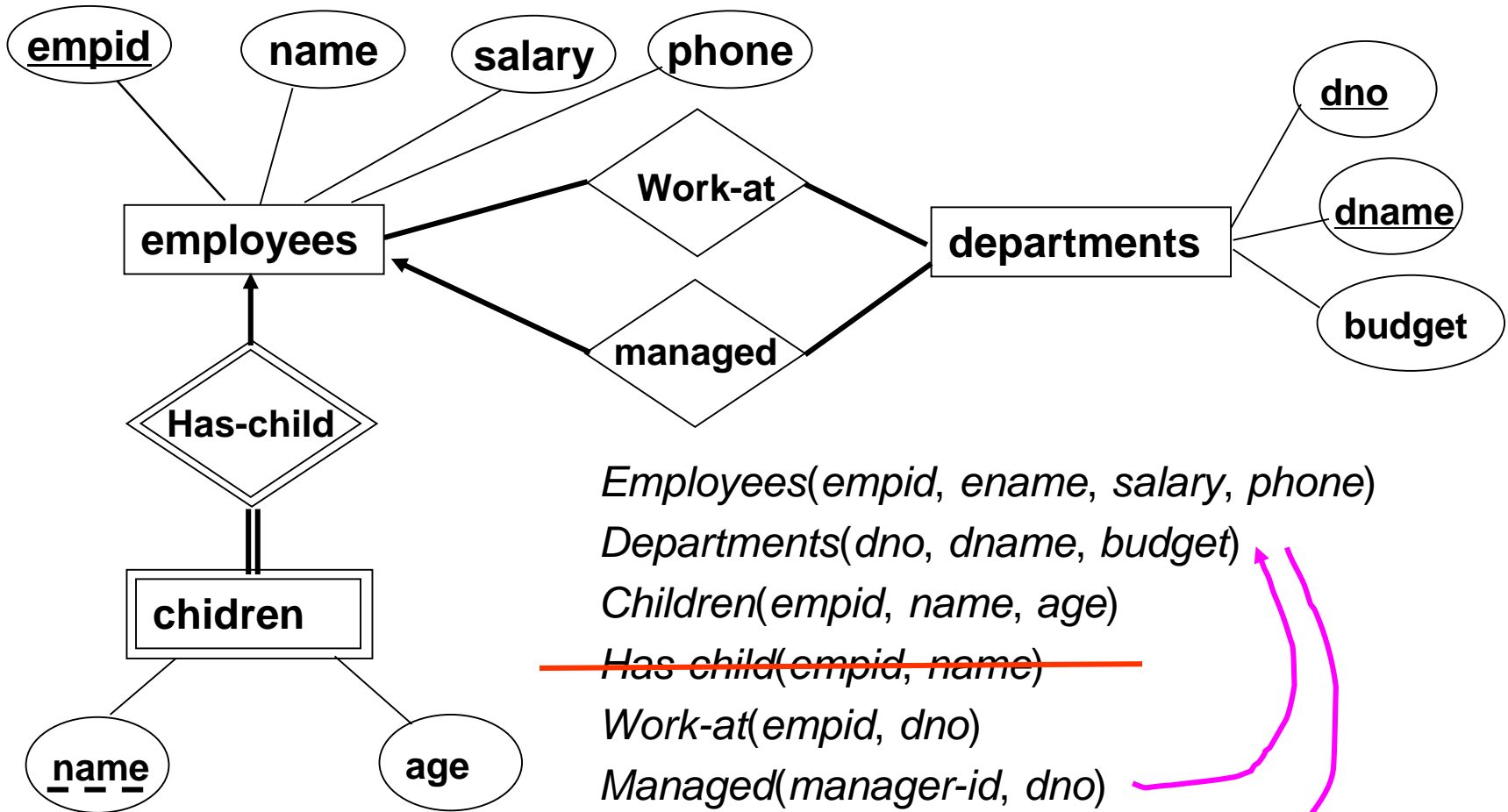
# Banking Database

- ❑ *branch(branch-name, branch-city, assets)*
- ❑ *customer(customer-name, customer-street, customer-city, banker-id)*
- ❑ *account(account-number, branch-name, balance)*
- ❑ *loan(loan-number, branch-name, amount)*
- ❑ *depositor(customer-name, account-number)*
- ❑ *borrower(customer-name, loan-number)*
- ❑ *Employee(employee-id, employee-name, phone, start-date, manager)*
- ❑ *Payment(loan-number, payment-number, payment-date, paymenty-amount)*

## Example 2

- ❑ A company database needs to store information about *employees* (identified by *empid*, with *name*, *salary* and *phone* as attributes); *departments* (identified by *dno*, with *dname* and *budget* as attributes) and *children* of the *employee* (with *name* and *age* as attributes). Employees work in departments; each department is *managed* by an employee; **a child must be identified uniquely by name when parent** (who is an employee; assume that only *one parent* works for the company) **is known**. We are not interested in information about a child once the parent leaves the company. Draw an E-R diagram that captures this information, and then transform it into relational schema.

## Example 2 (Cont.)



*Employees(empid, ename, salary, phone)*

*Departments(dno, dname, budget)*

*Children(empid, name, age)*

~~*Has child(empid, name)*~~

*Work-at(empid, dno)*

*Managed(manager-id, dno)*

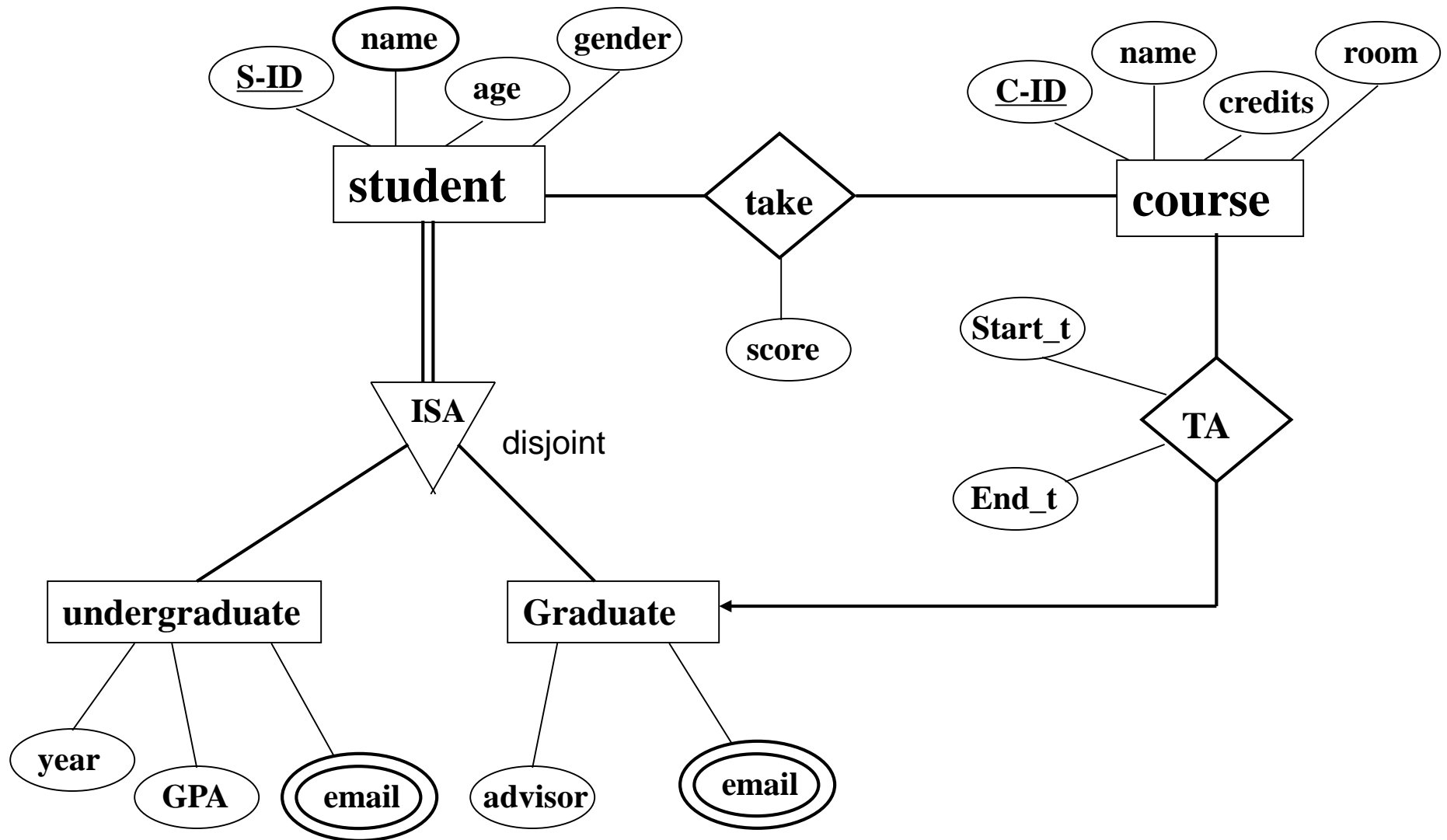
*Departments(dno, dname, budget, **manager-id**)*

## Example 3

- ❑ **Students** take **courses**. Each **student** has a student ID number, name, age, and gender. Each **course** has a course ID number, name, credits and room number. Each student is either an undergraduate or graduate student. For each undergraduate student, we want to record also his/her year, GPA, and (possibly multiple) email addresses. For each graduate student, we want to record his/her advisor name, and also his/her multiple email addresses. In addition, each course is assigned to exactly one graduate student, who acts as the TA (助教) of the course. For this TA role we want to record the start and end times(e.g., start at 11/02/28 and ends at 11/06/28). Please draw an E-R diagram that captures these information.



## Example 3 (Cont.)

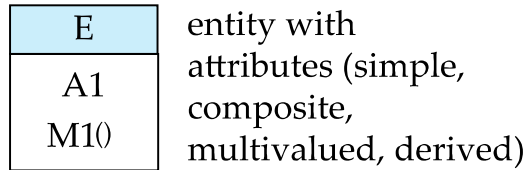


# UML

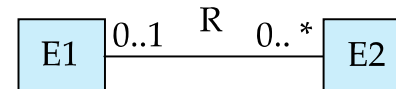
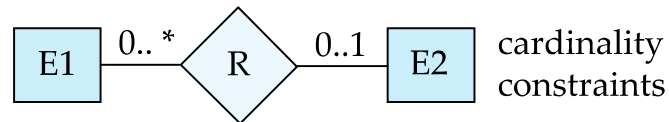
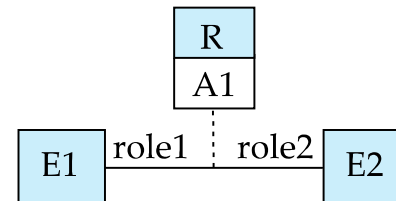
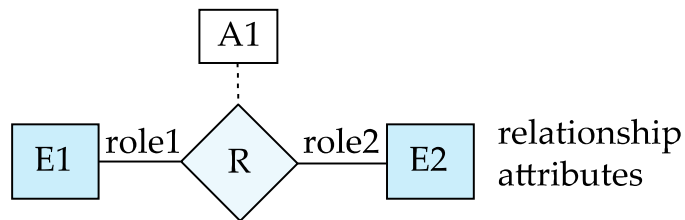
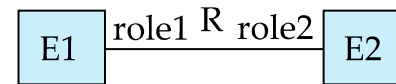
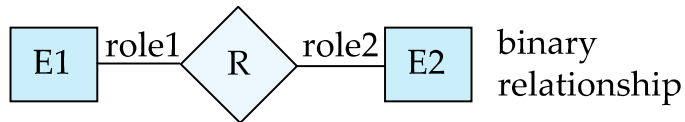
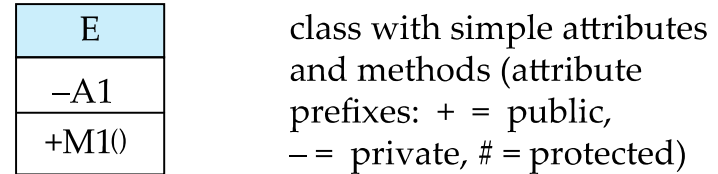
- ❑ **UML:** Unified Modeling Language
- ❑ UML has many components to graphically model different aspects of an entire software system
- ❑ UML Class Diagrams correspond to E-R Diagram, but several differences.

# ER vs. UML Class Diagrams

## ER Diagram Notation



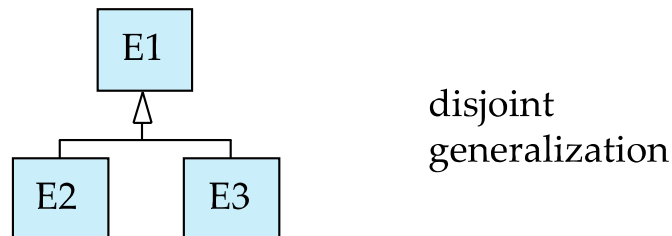
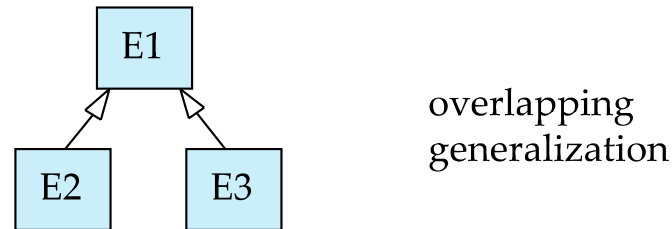
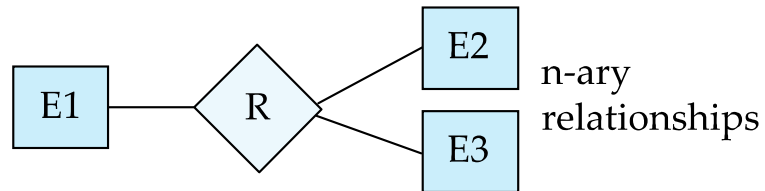
## Equivalent in UML



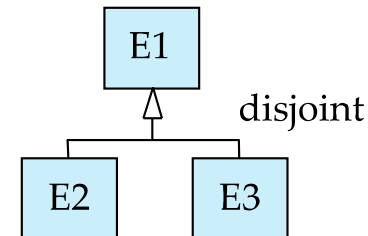
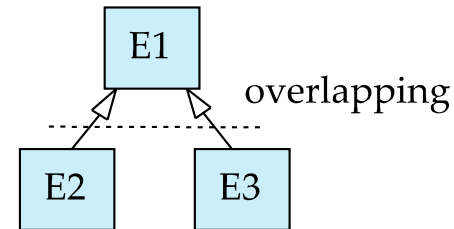
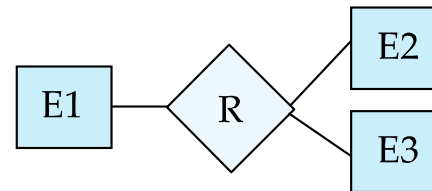
\* Note reversal of position in cardinality constraint depiction

# ER vs. UML Class Diagrams

## ER Diagram Notation



## Equivalent in UML



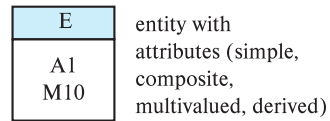
\* Generalization can use merged or separate arrows independent of disjoint/overlapping

# UML Class Diagrams (Cont.)

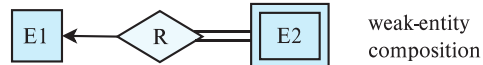
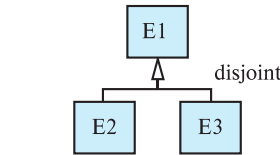
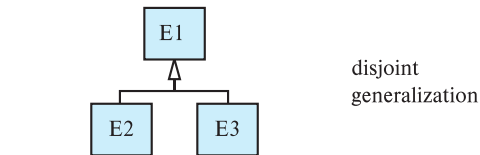
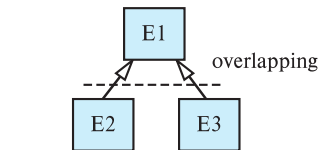
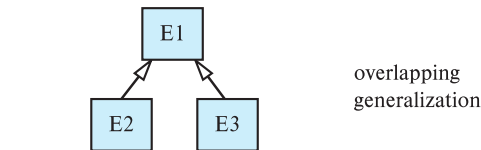
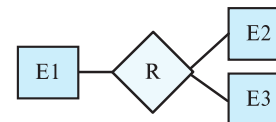
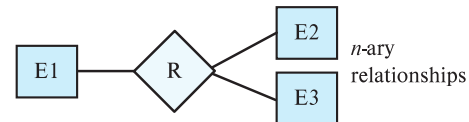
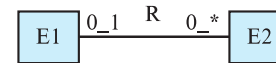
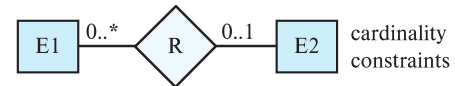
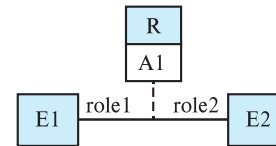
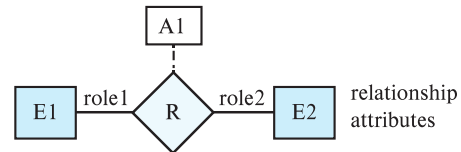
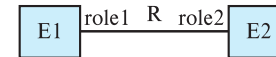
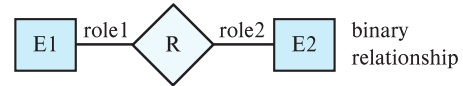
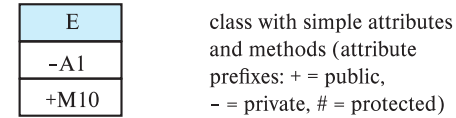
- ❑ Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- ❑ The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- ❑ The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.

# ER vs. UML Class Diagrams

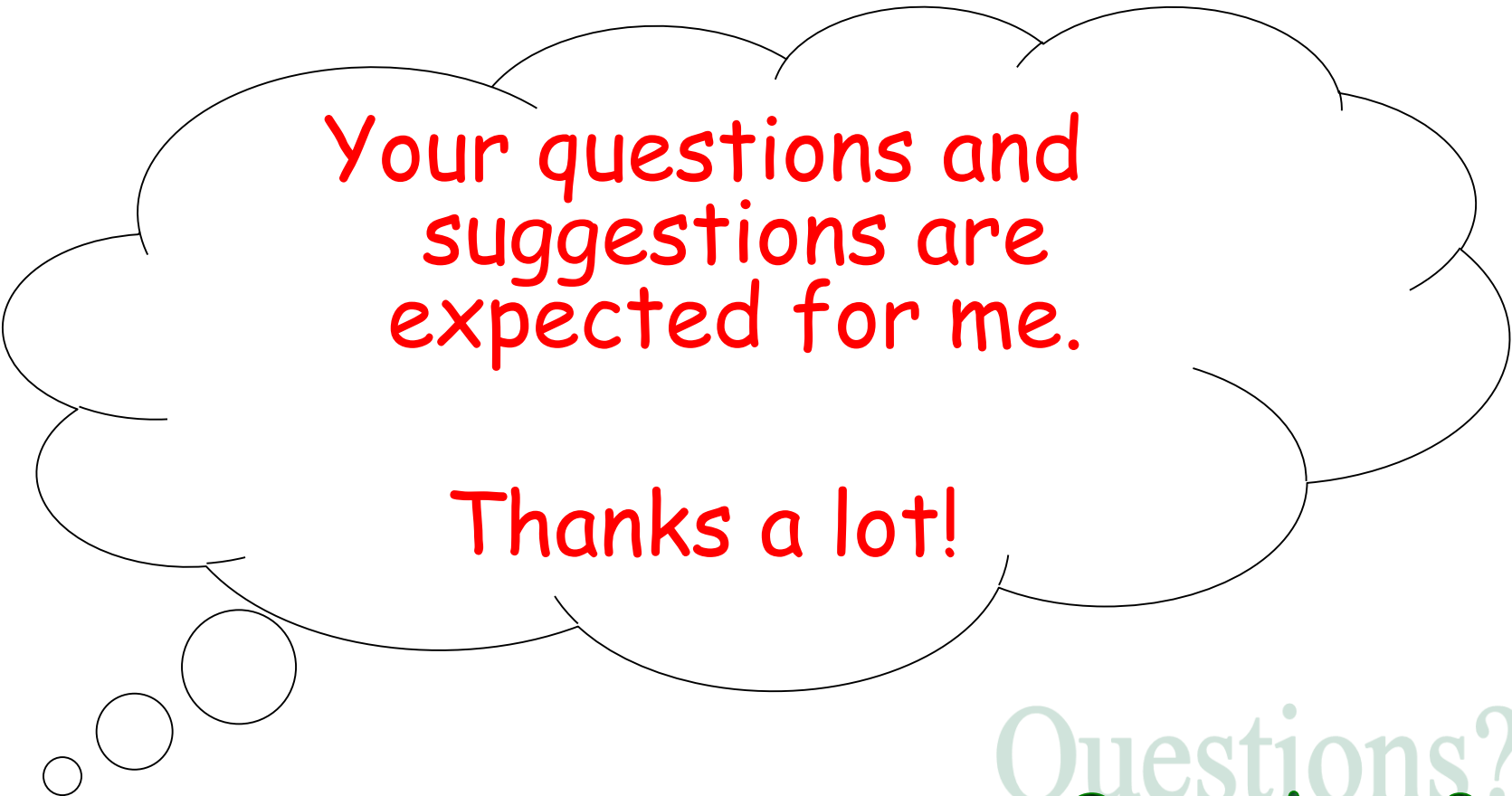
## ER Diagram Notation



## Equivalent in UML



# Q & A



Your questions and  
suggestions are  
expected for me.

Thanks a lot!

Questions?  
Questions?

Exercises: 6.1, 6.2, 6.21, 6.22, and 6.24 (see  
Pages 294-299)