Database System

北京交通大学软件学院

王方石 教授

E-mail: fshwang@bjtu.edu.cn

Chapter 5 Database Design

5.1 Database Development Lifecycle

5.2 Entity/Relationship Model

5.3 Enhanced Entity-Relationship Model

5.4 Rules of Mapping E-R Model to Relational Model

5.1 Database Development Lifecycle

1. **Requirements collection and analysis**
2. **Database Dessign**
   - **Conceptual Design**
   - **Logical Design**
   - Physical Design
3. Database Implementation
4. Testing
5. Operational maintenance

Database Design

- **Process of creating a design for a database that will support the enterprise's operations and objectives.**
- **The result of designing DB is data models.**

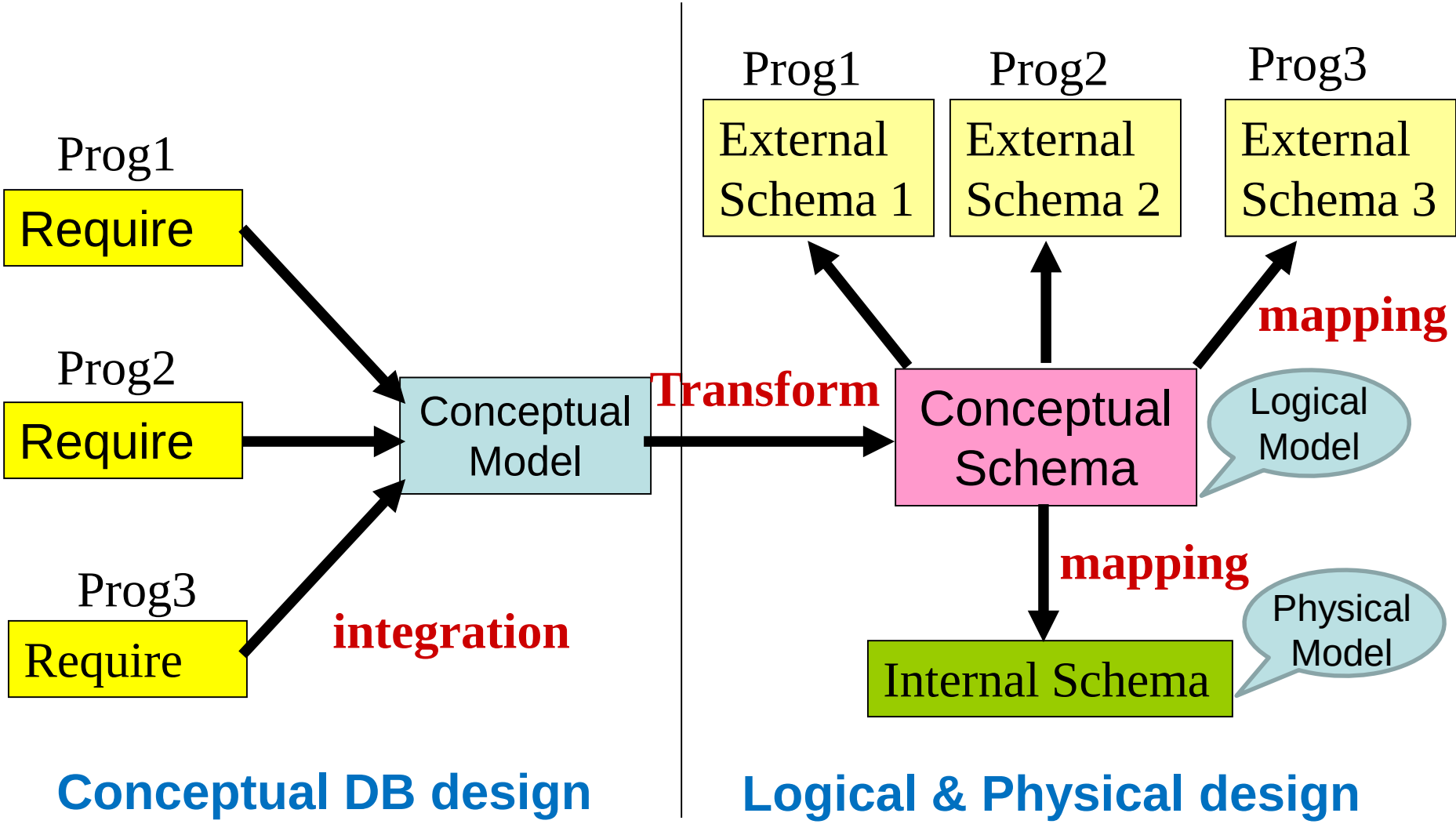
Database Design

- **Main purposes of data modeling include:**
  - **to assist in understanding the meaning (semantics) of the data;**
  - **to facilitate communication about the information requirements.**
- **Building data model requires answering questions about entities, relationships, and attributes.**

Three Phases of Database Design

- **Conceptual database design**
  - **Conceptual Models**
- **Logical database design**
  - **Logical models (relational)**
- **Physical database design**
  - **Physical Models (specific DBMS)**

Three Phases of Database Design



（1） Conceptual Database Design

It is the process of constructing a model of the information (called Conceptual Model) used in an enterprise, independent of DBMS and all physical considerations.

- **Conceptual Model is built using the information in users' requirements specification（用户需求规格说明）.**
- **Conceptual Model is the source of information for logical design phase.**

（2） Logical Database Design

- **It is the process of constructing a model of the information used in an enterprise based on a specific data model (e.g. relational), but independent of a particular DBMS（DB2 or Oracle） and other physical considerations.**
- **Conceptual data model is refined and mapped on to a logical data model.**
- **The logical data model is based on the target data model for the database (for example, the relational data model).**

（2）Logical Design

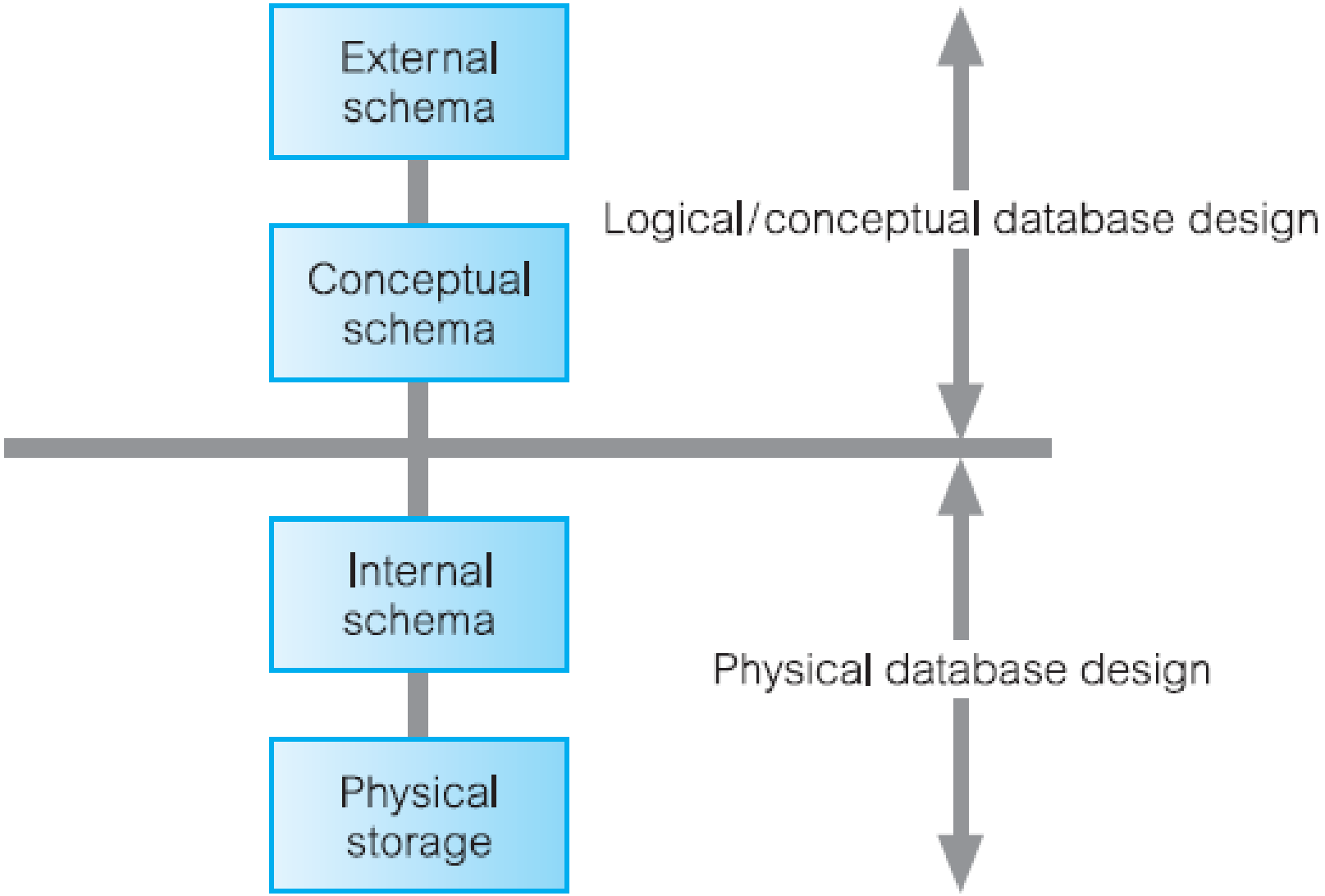
Translating ER model (diagram) to relational model

- **Entity set** -> **relation**.
- **Attributes** -> **attributes**.
- **Relationships** -> **relations** whose attributes are only:
  - The keys of the connected entity sets.
  - Attributes of the relationship itself.

（3） Physical Database Design

- **It is the process of producing a description of the database implementation on secondary storage.**
- **It is tailored to a specific DBMS.**
- **It describes the storage structures (存储结构) and access methods（存取方法） used to achieve efficient access to data.**

Data modeling and ANSI-SPARC architecture



5.2 Entity/Relationship Model
Purpose of E/R Model

- The E/R model allows us to sketch database schema designs.
- It includes some constraints, but not operations.
- Designs are pictures called *entity-relationship diagrams*.
- Later: convert E/R designs to relational DB designs (i.e. logical data model).

5.2.1 Concepts of the ER Model

- **Entity**
- **Attributes**
- **Entity Sets**
- **Attribute Domain**
- **Keys**
- **Relationship**
- **Relationship types**

Entity Sets

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An ***entity*** is an object that exists and is distinguishable from other objects.
  - Example: Physical existence (person, event)
    or Conceptual existence (sale, viewing)
- Entities have ***attributes***: property of (the entities of) an entity set.
  - Example: people have *names* and *addresses*
- An ***entity set*** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays

Attributes

- **Attribute**
  - **Property of an entity or a relationship type.**

  Example:

  staff ***(staffNo, lname, fname, position, salary)***

  branch ***(branchNo, branch-address)***

- **Attribute Domain**
  - **Set of allowable values for one or more attributes.**
  - **e.g. rooms [1..15]**

Attributes

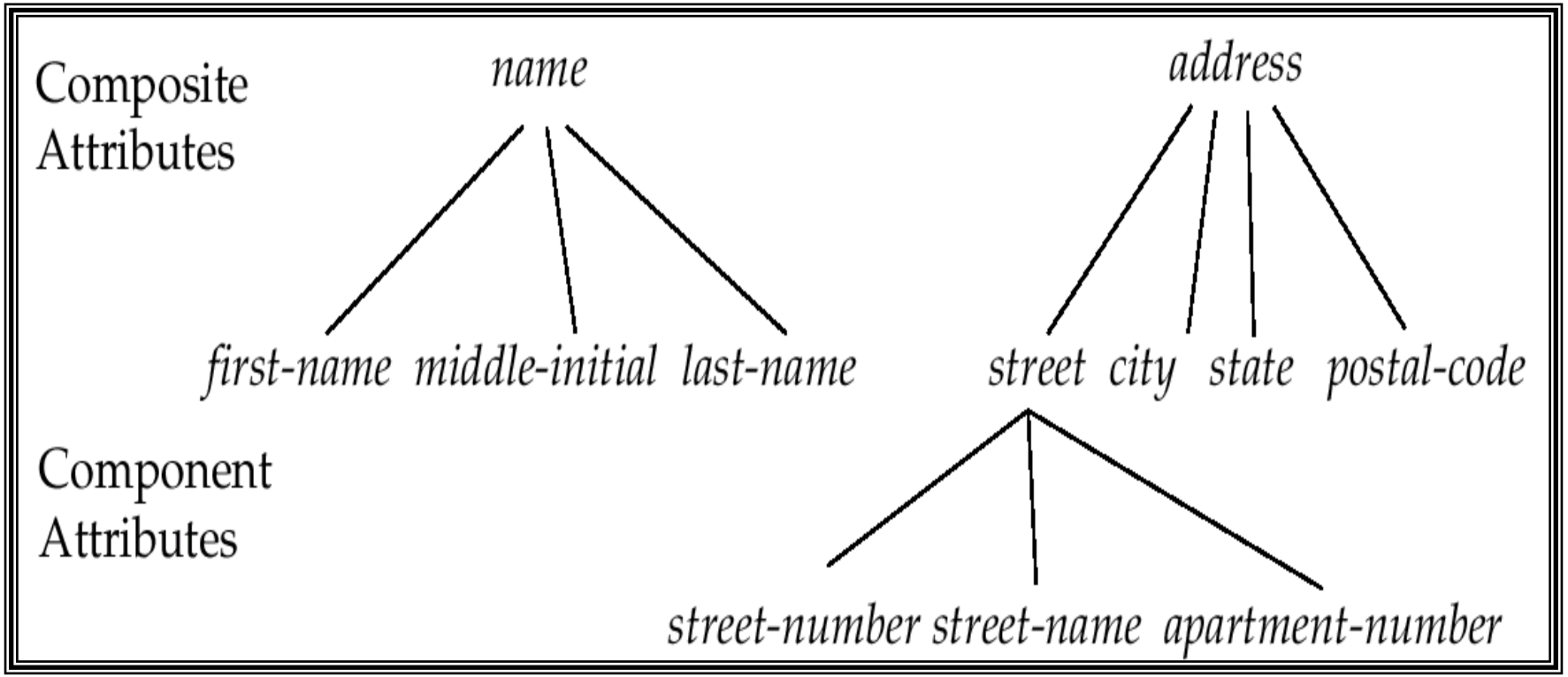
◆Simple Attribute

– **Attribute composed of a single component with an independent existence.**

◆Composite Attribute（复合属性）

– **Attribute composed of multiple components, each with an independent existence.**

Composite Attributes



Attributes

- **Single-valued Attribute**
  - **Attribute that holds a single value for each occurrence of an entity type.**
- **Multi-valued Attribute（多值属性）**
  - **Attribute that holds multiple values for each occurrence of an entity type.**
  - **E.g. multivalued attribute: *phone-numbers***

Attributes

◆ Derived Attribute（派生属性）

- **Attribute that represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity type.**
- **Can be computed from other attributes**

  e.g. student (sno, sname, DOB, *age*, NumOfCourse)

  sc (sno, cno, grade)

  ***age*, given date of birth**

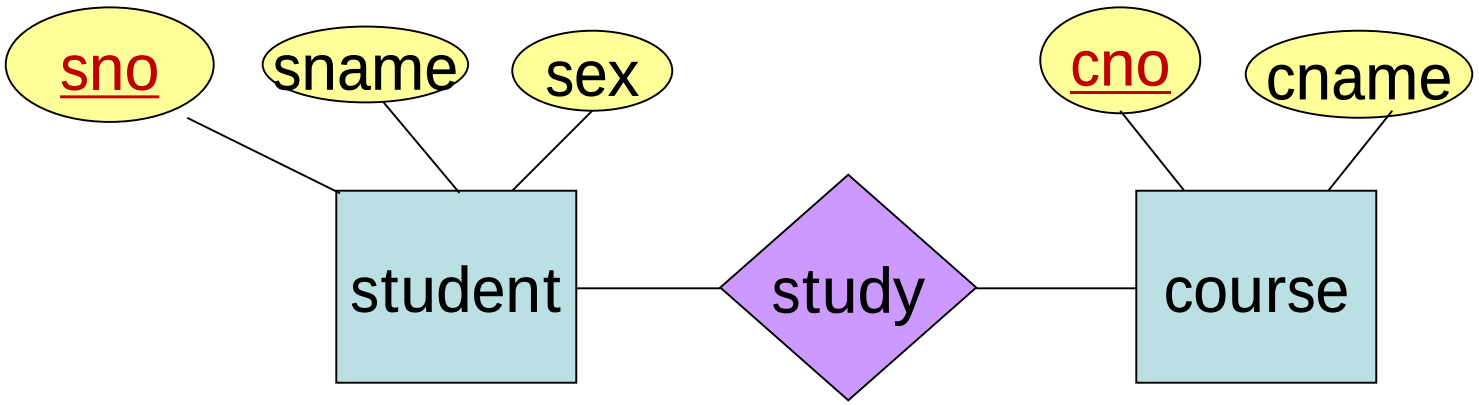
  NumOfCourse： the total number of learned courses.

Keys (关键字，码，键)

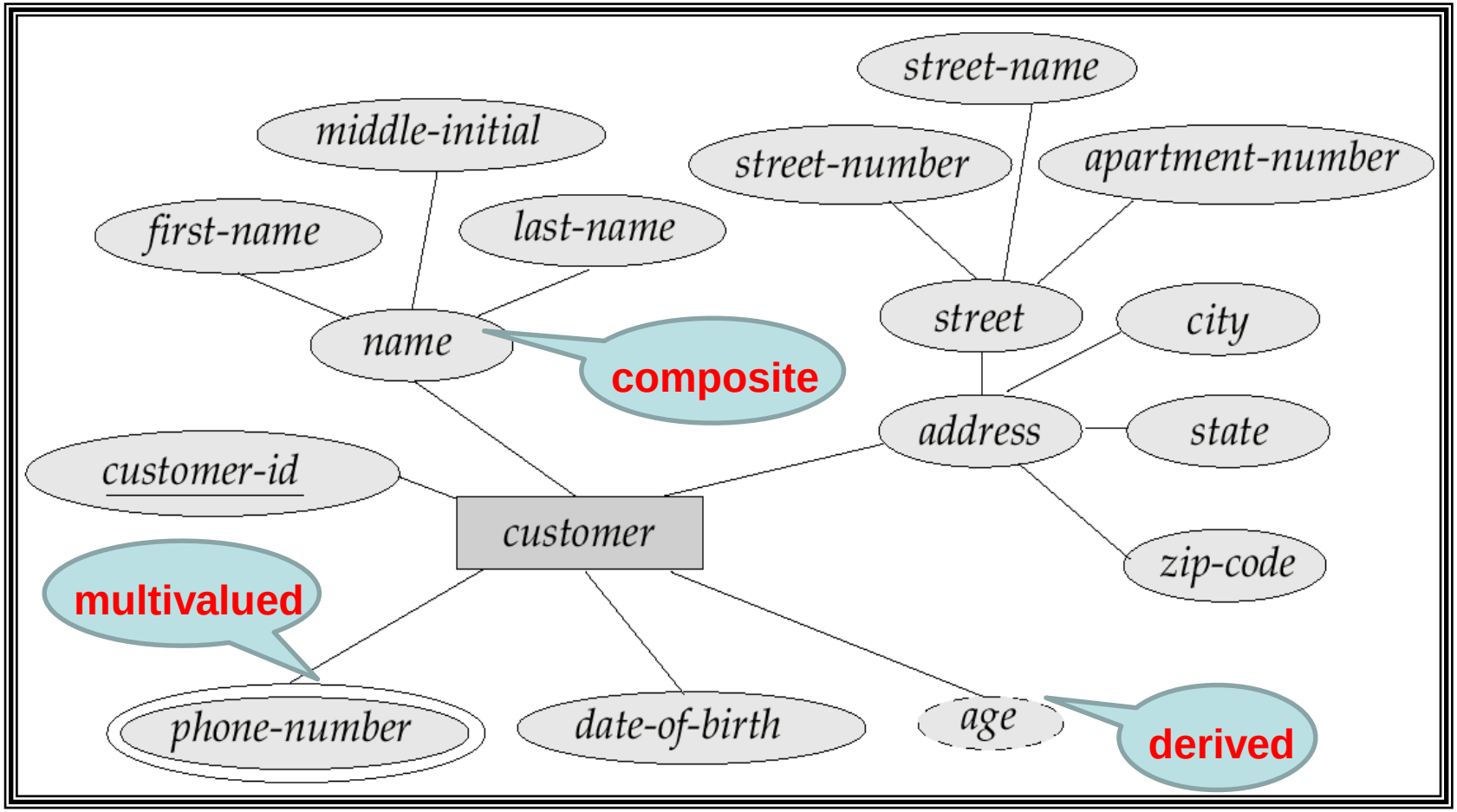
- **Candidate Key（CK，候选码）**
  - **Minimal set of attributes that uniquely identifies each occurrence of an entity type.**
- **Primary Key（PK，主码）**
  - **Candidate key selected to uniquely identify each occurrence of an entity type.**
- **Alternate Key（AK， 备用码）**
  - **Other Candidate key(s) except primary key**
- **Composite Key（复合候选码）**
  - **A candidate key that consists of two or more attributes.**
  - **e.g.** Advert (propertyNo, newspaperName, dateAdvert, cost)

E/R Diagrams

- In an entity-relationship diagram:
  - **Entity set** = rectangle.
  - **Attribute** = oval (**ellipse**), with a line to the rectangle representing its entity set.
  - **Primary Key**： being Underlined.
- A **relationship** connects two or more entity sets.
- It is represented by a diamond, with lines to each of the entity sets involved.



Entity With Composite, Multivalued, and Derived Attributes



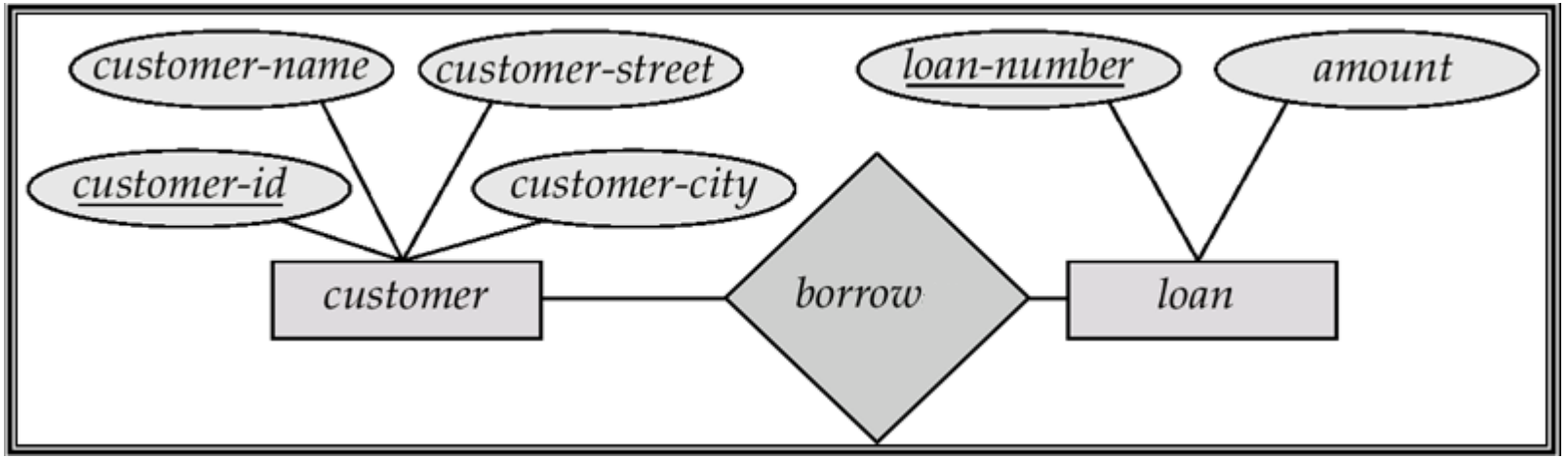
- **Ellipses** represent attributes
  - **Double ellipses** represent **multivalued** attributes.
  - **Dashed ellipses** denote **derived** attributes.
- **Underline** indicates primary key attributes.

Relationship Types

- **Degree of a Relationship (联系的元数)**
  - **The number of participating entity types in relationship.**

- **different degree:**
  - **two is binary（二元联系）;**
  - **three is ternary （三元联系）;**
  - **four is quaternary （四元联系）.**

Binary Relationship called *borrow*

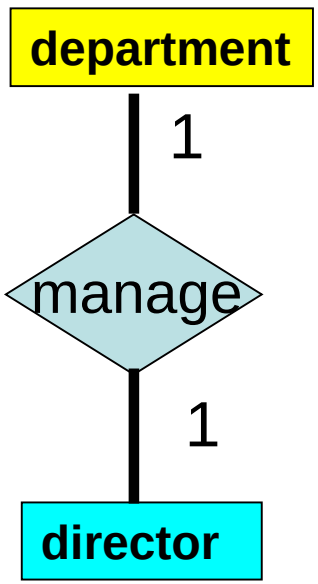


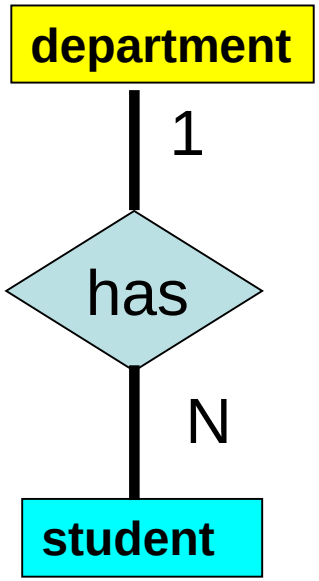
- **Rectangles** represent entity sets.
- **Diamonds** represent relationship sets.
- **Lines** link attributes to entity sets and entity sets to relationship sets.
- **Ellipses** represent attributes
  - **Double ellipses** represent multivalued attributes.
  - **Dashed ellipses** denote derived attributes.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

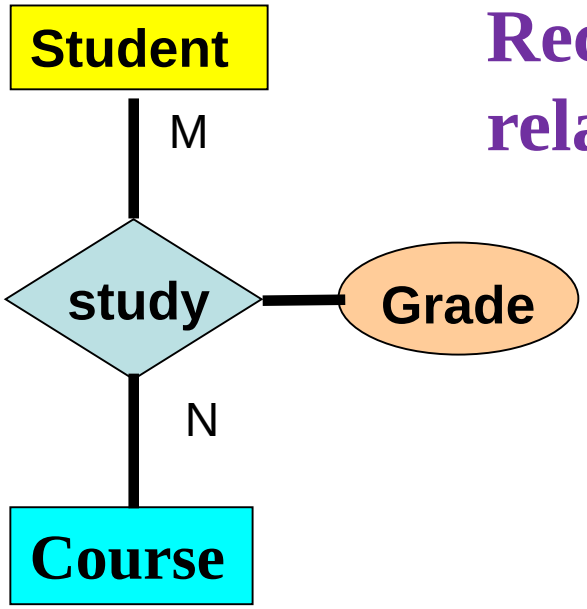
Three types of Binary relationship

- **The most common degree for relationships is binary.**
- **Binary relationships are generally referred to as being:**
  - **one-to-one (1:1)**
  - **one-to-many (1:*)**
  - **many-to-many (*:*)**

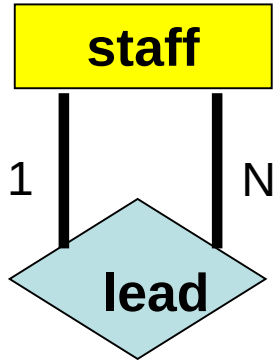
Three types of Binary relationship







Recursive(递归) relationship



1:1

1:N

M:N

1:N

5.2.2 ER Modeling

Design Steps

Step 1: Differentiate (划分) entity sets vs. attributes

Determine entity, attribute, relationship.

Step 2. Integrate sub ER diagrams

- Design ER diagram for each sub-system
- Eliminate the conflicts of all sub ER diagrams, and generate the initial overall ER diagram.
- Eliminate the redundancy, and generate the basic overall ER diagram.

5.2.2 ER Modeling

During step 1, consider the following issues:

(1) Use of entity sets vs. attributes

- Treat the data as attributes as soon as possible. Don't use an entity set when an attribute will do.
- As an attribute, it dose not need a further description, i.e. it can not contain other attributes.
- As an attribute, it can not have the relationship with other entities except the entity to which it belongs.
- The relationship between an attribute and its own entity can be only M:1.

5.2.2 ER Modeling

During step 1, consider the following issues:

(2) Use of entity sets vs. relationship sets

The possible guideline is to designate a **relationship set** to describe **an action** that occurs between entities.

(将实体集之间发生的动作描述为联系)

Design Issues

(3) 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.
- We focus on **Binary relationship.**

(4) Placement of relationship attributes

Review of Lecture 18

1. 为什么需要规范化理论？目的是什么？
2. 规范化的核心操作是什么？
3. 如何判断其是否为无损连接分解？
4. 什么是泛关系？
5. 什么样的关系模式最低是2NF？最低是3NF？
6. 在什么情况下，3NF可能不是BCNF？在什么情况下， 3NF一定是BCNF？
7. 1NF →2NF →3NF →BCNF，分别消除了什么不好的性质？
8. 数据库开发的生命周期分哪几个阶段？
9. 数据库设计分为哪几个阶段？其任务是什么？现在程序员只需要考虑哪级模式以上的设计？
10. 概念：简单属性、复合属性、单值属性、多值属性、派生属性、递归联系
11. 在设计ER模型时，确定实体、属性的原则？什么样的信息被设计成属性？

Design Techniques

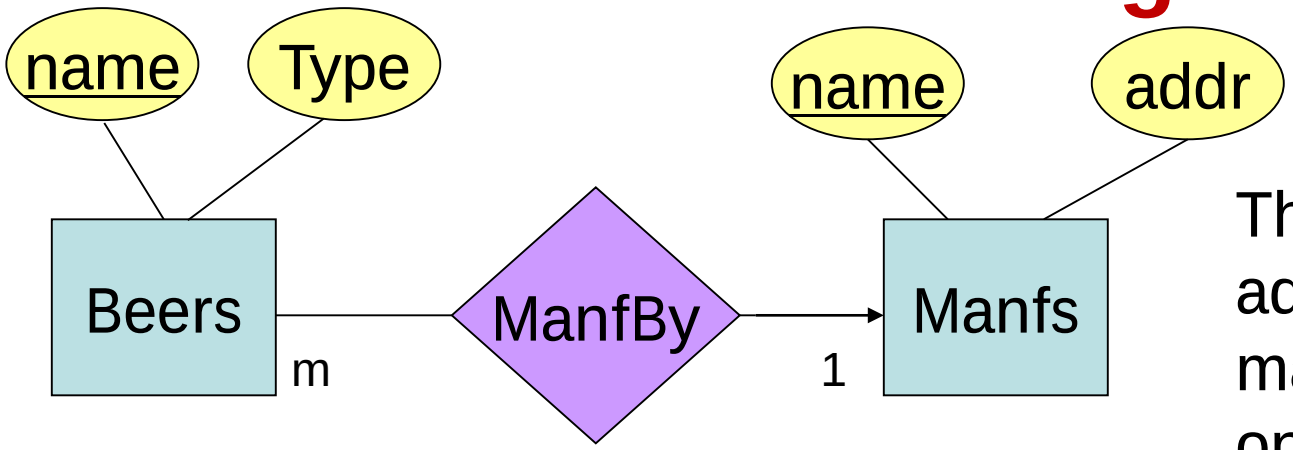
1. Avoid redundancy.
2. Don’t use an entity set when an attribute will do.

1. Avoiding Redundancy

- *Redundancy* = saying the same thing in two (or more) different ways.
- Wastes space and (more importantly) encourages inconsistency.
  - Two representations of the same fact become inconsistent if we change one and forget to change the other.
  - *Redundancy* recall anomalies due to FD's.

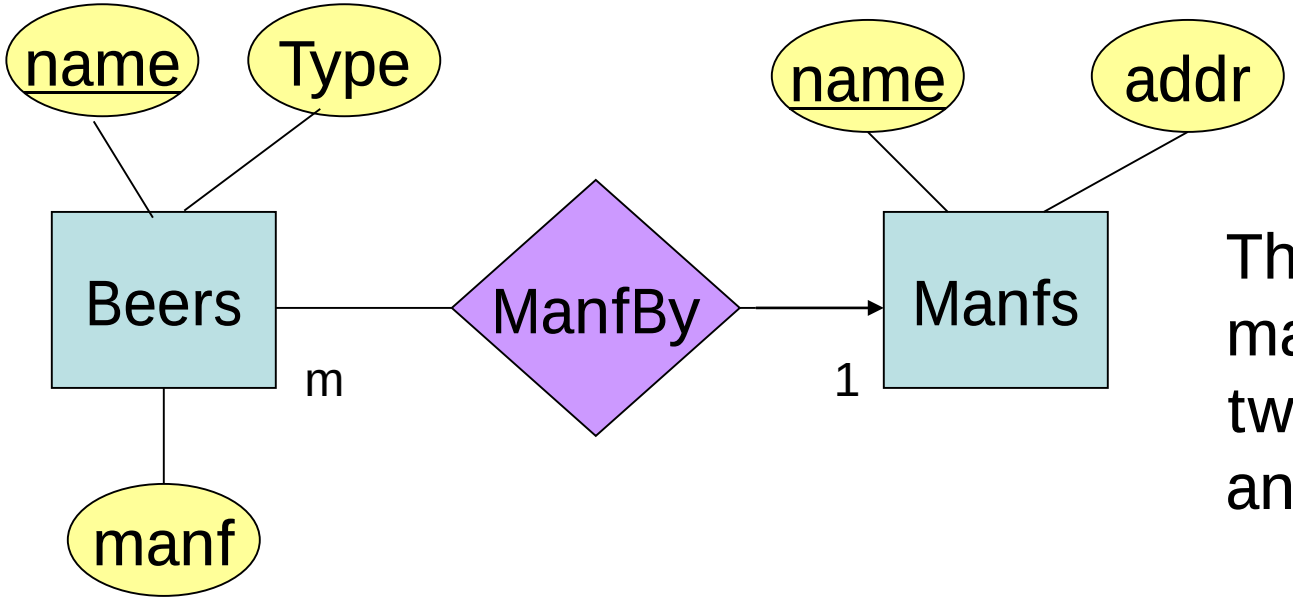
Example for avoiding redundancy

Good Design



This design gives the address of each manufacturer exactly once.

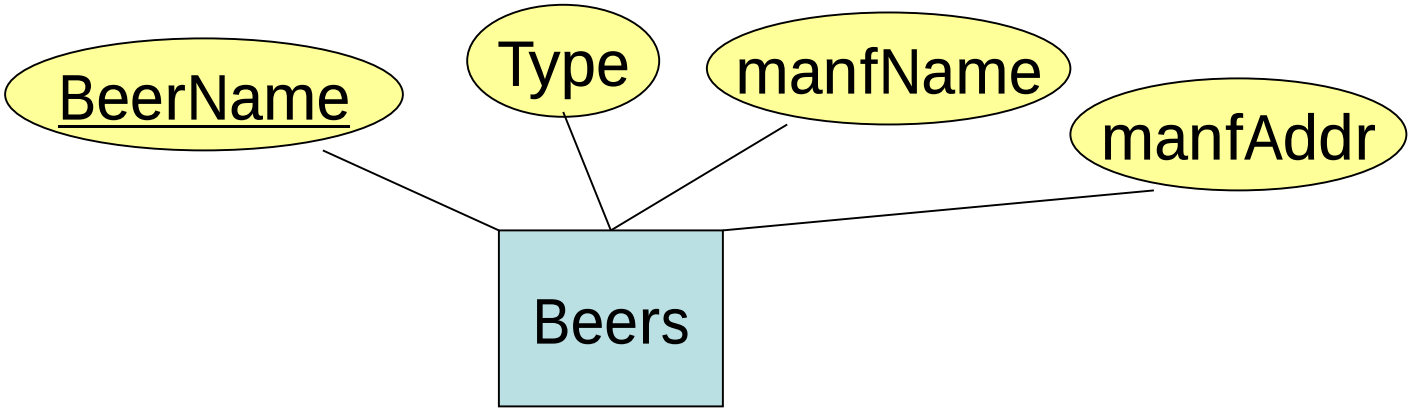
Bad Design



This design states the manufacturer of a beer twice: as an attribute and as a related entity.

Bad Design

Merge the manufacturer table and beer table



- This design repeats the manufacturer's address once for each beer
- loses the address if there are temporarily no beers for a manufacturer.

2. Entity Sets Versus Attributes

An **entity** set should satisfy at least one of the following conditions:

- It is more than the name of something**. it has at least one non-primary-key attribute**.

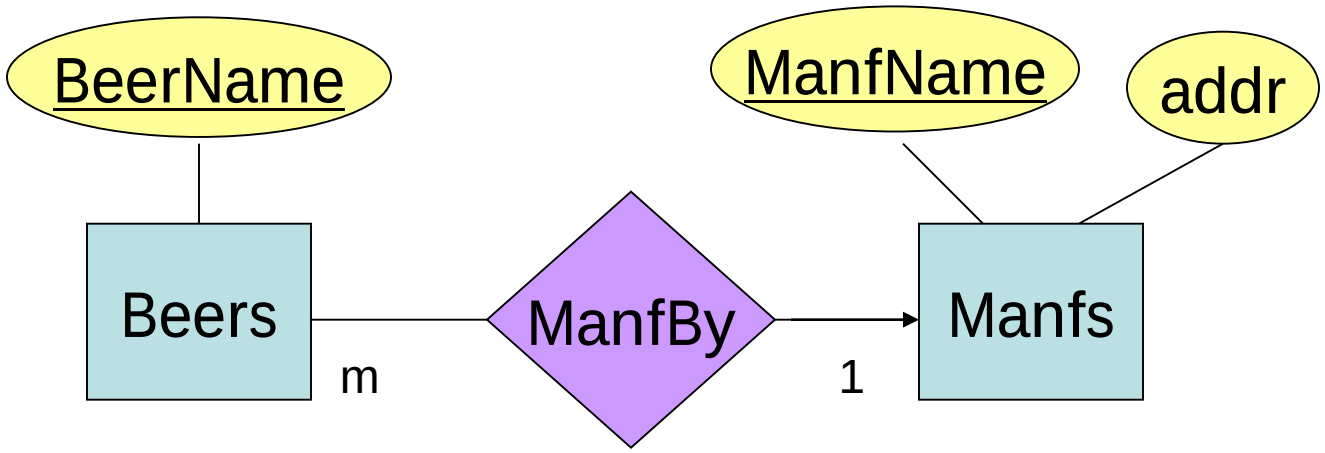
  实体，除了名称（PK），还应该有其他属性, 即实体至少应该有一个非主属性

or

- It is the "**many**" in a many-one or many-many relationship.

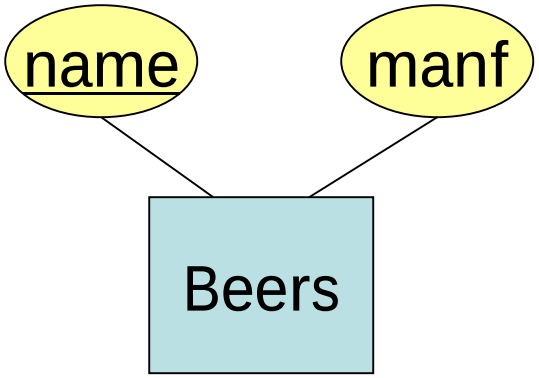
  或者，它是1：M 或 M:N 联系的多端（则它可以只包含一个主键，没有非主属性）

Good Design



- Manfs deserves to be an entity set because of the nonkey attribute addr.
- Beers deserves to be an entity set because it is the “many” of the one-many relationship ManfBy.

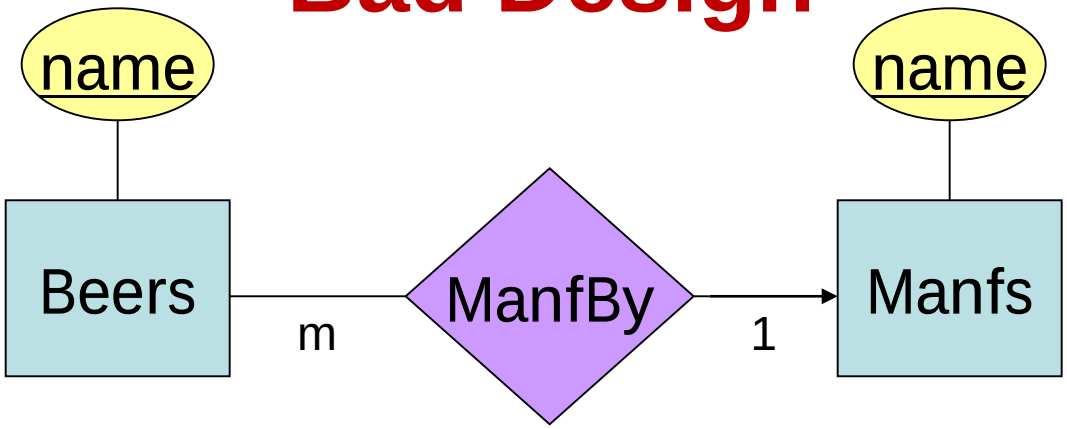
Example: If we need only manufacturer's name



Good Design

there is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Bad Design

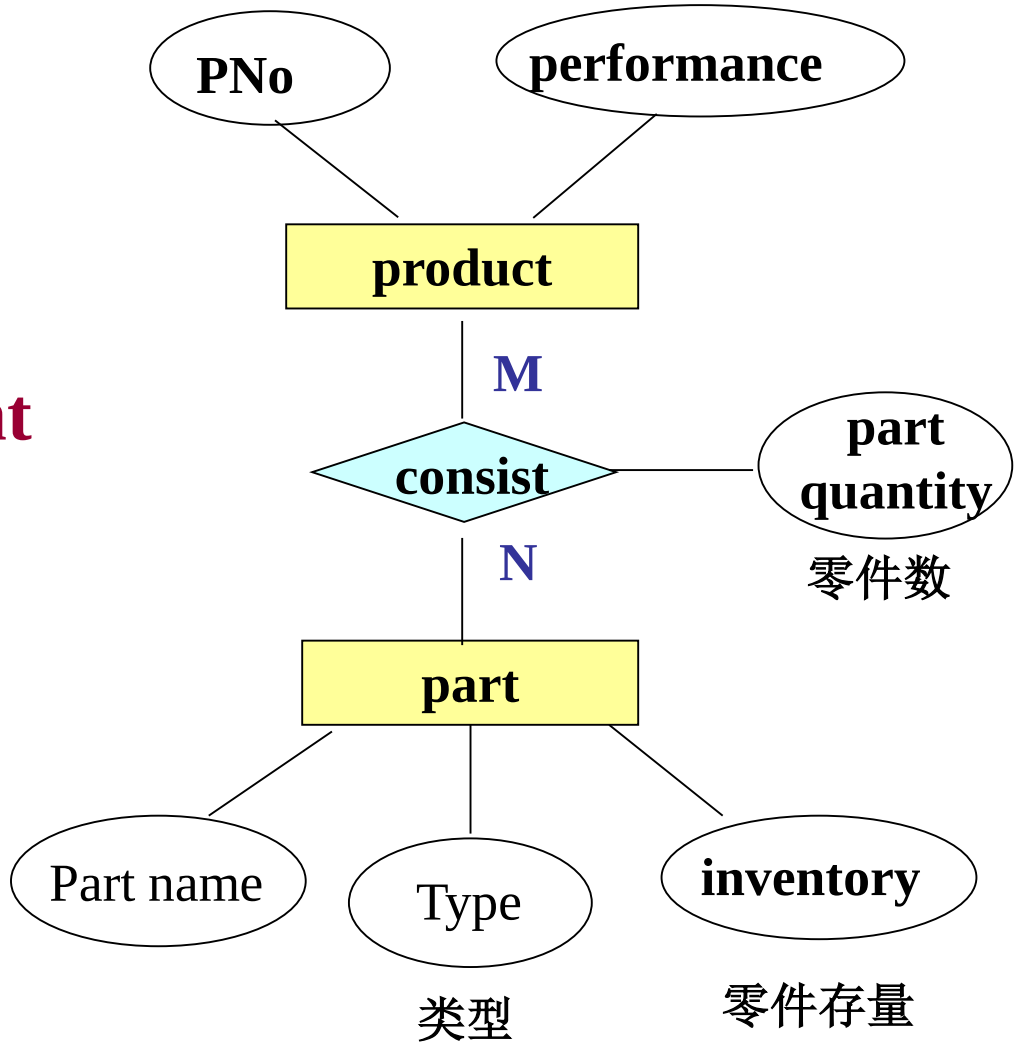


Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

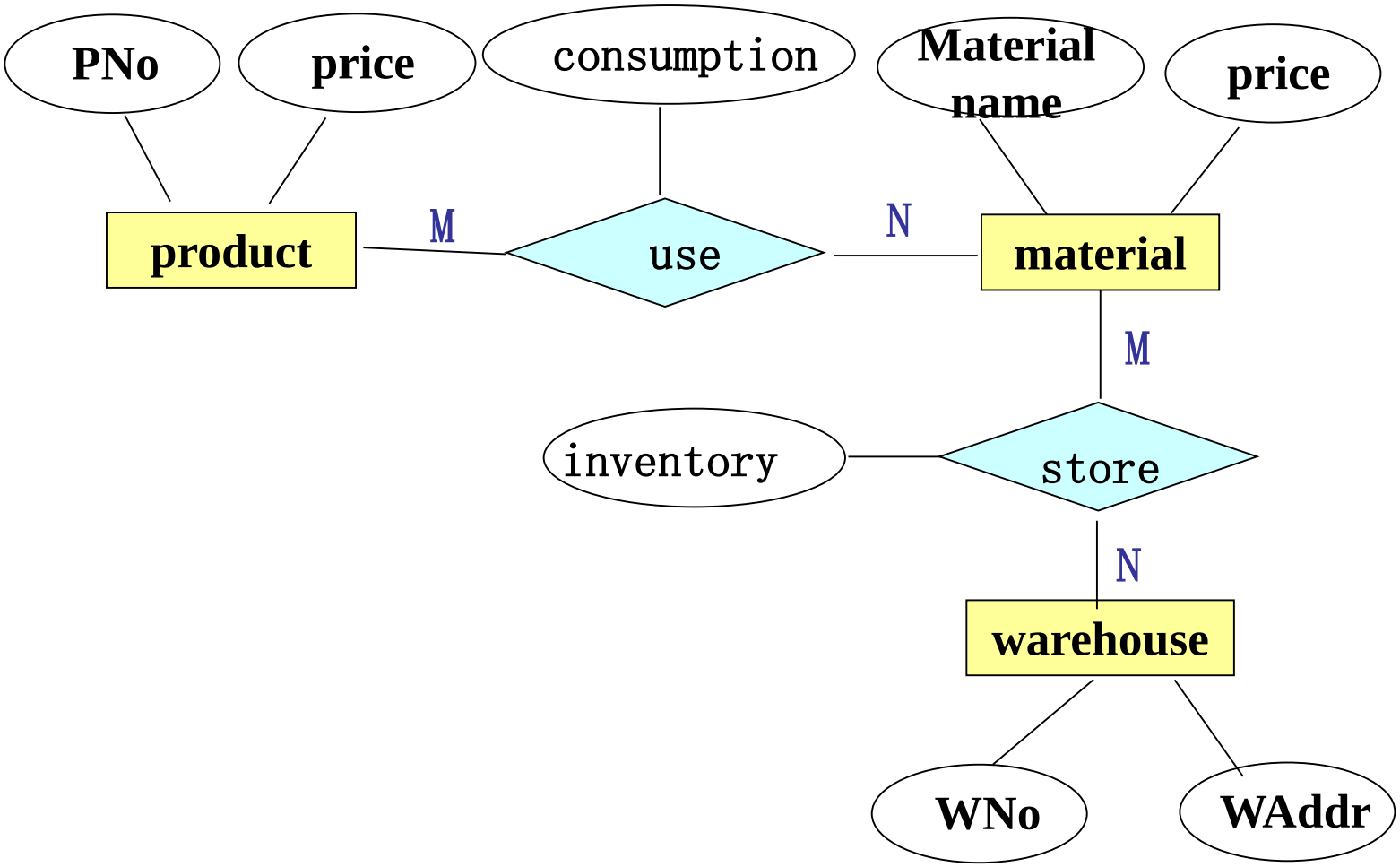
Step 2: Integrate sub ER diagrams

example：factory（1）department of technique
（2）department of supply

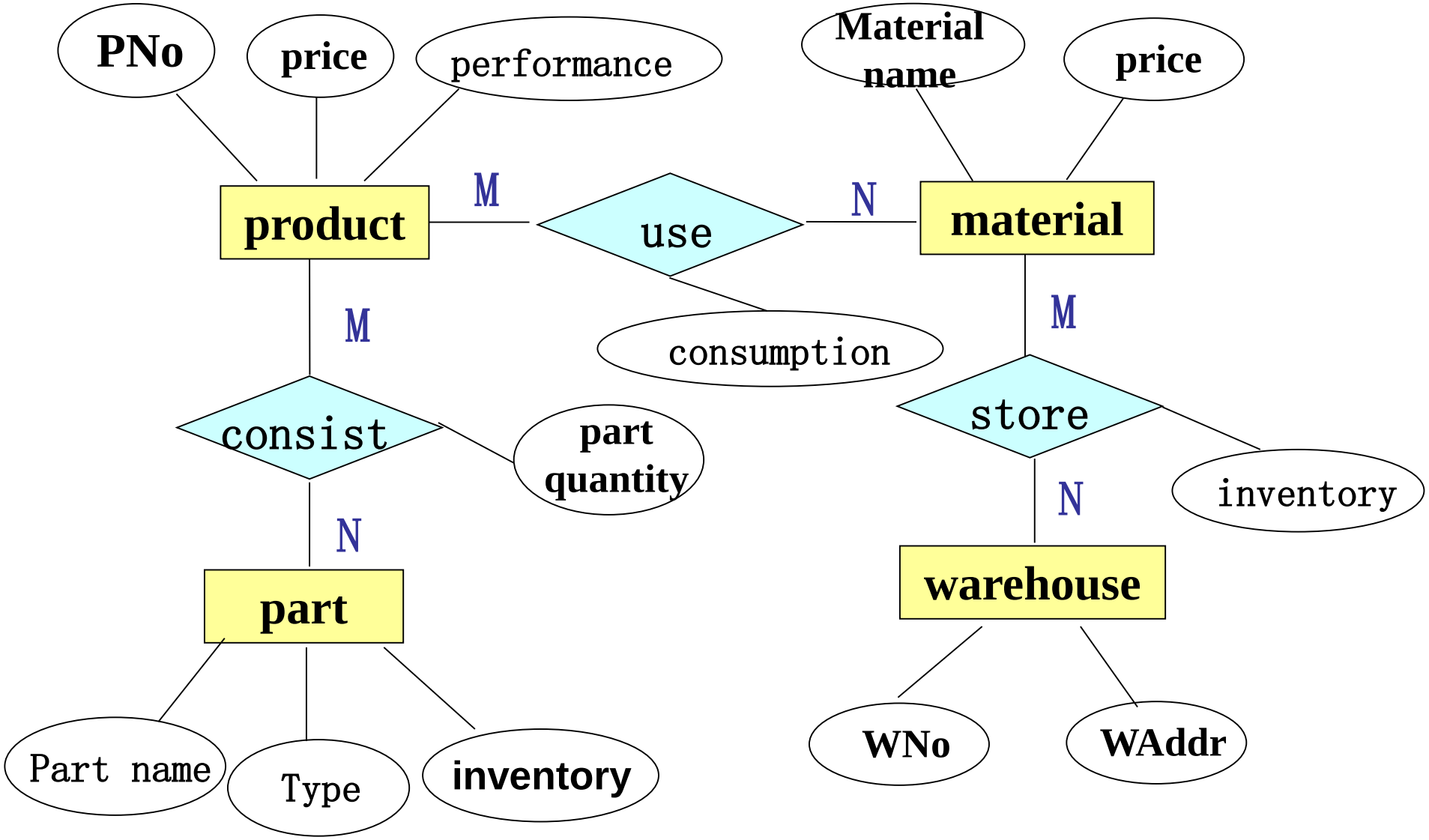
Sub ER diagram of technique department



Sub ER diagram of supply department



Integrated ER diagram：



5.3 Enhanced Entity-Relationship Model

- **Since 1980s there has been an increase in emergence of new database applications with more demanding requirements.**

- **Basic concepts of ER modeling are not sufficient to represent requirements of newer, more complex applications.**

- **Response is development of additional 'semantic' modeling concepts.**

The Enhanced Entity-Relationship Model

- **Semantic concepts are incorporated into the original ER model and called the Enhanced Entity-Relationship (EER) model.**

- **Examples of additional concepts of EER model are:**
  - **specialization / generalization（ Isa ）;**
  - **aggregation（聚合， 'has-a' or 'is-part-of'）;**
  - **composition（组合，a special form of aggregation）**

Specialization / Generalization

- **Specialization（特殊化，找差异）**
  - **Process of maximizing differences between members of an entity by identifying their distinguishing characteristics.**

- **Generalization (is subset of，概化、泛化，找共同点)**
  - **Process of minimizing differences between entities by identifying their common characteristics.**

Specialization /Generalization

- 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**.

Specialization /Generalization

The concepts of Specialization / Generalization is associated with special types of entities known as Superclasses and Subclasses, and the process of attribute inheritance.

- **Superclass**
  - **An entity type that includes one or more distinct subgroupings of its occurrences.**
- **Subclass**
  - **A distinct subgrouping of occurrences of an entity type.**

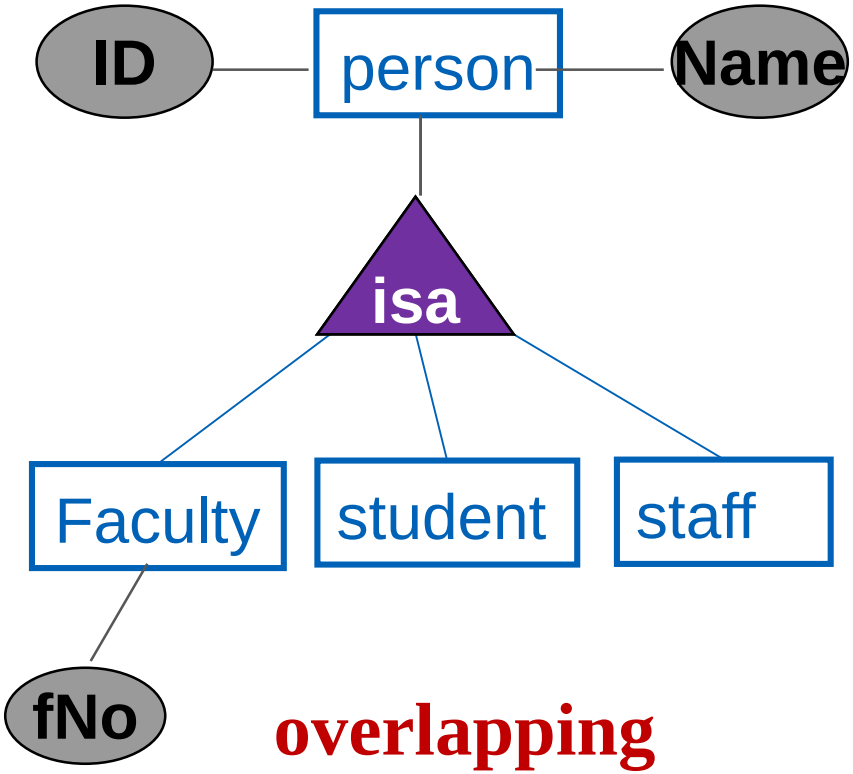
Superclass/subclass relationship is one-to-many (1:m).

Superclass / Subclass

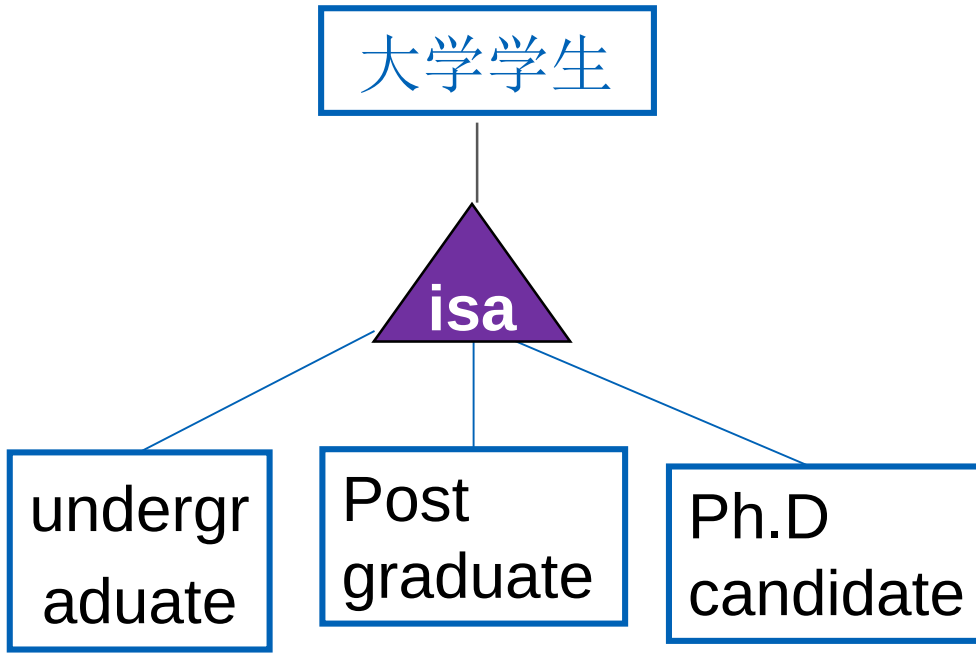
- **Attribute Inheritance**
  - **An entity in a subclass represents same ‘real world’ object as in superclass, and may possess subclass-specific attributes, as well as those associated with the superclass.**
- *Subclass* = special case = fewer entities = more properties.

Superclass / Subclass

- **Superclass may contain overlapping or distinct subclasses.**
- **Not all members of a superclass need be a member of a subclass.**（并不是超类中每一个成员都必须是子类中的成员。）



overlapping

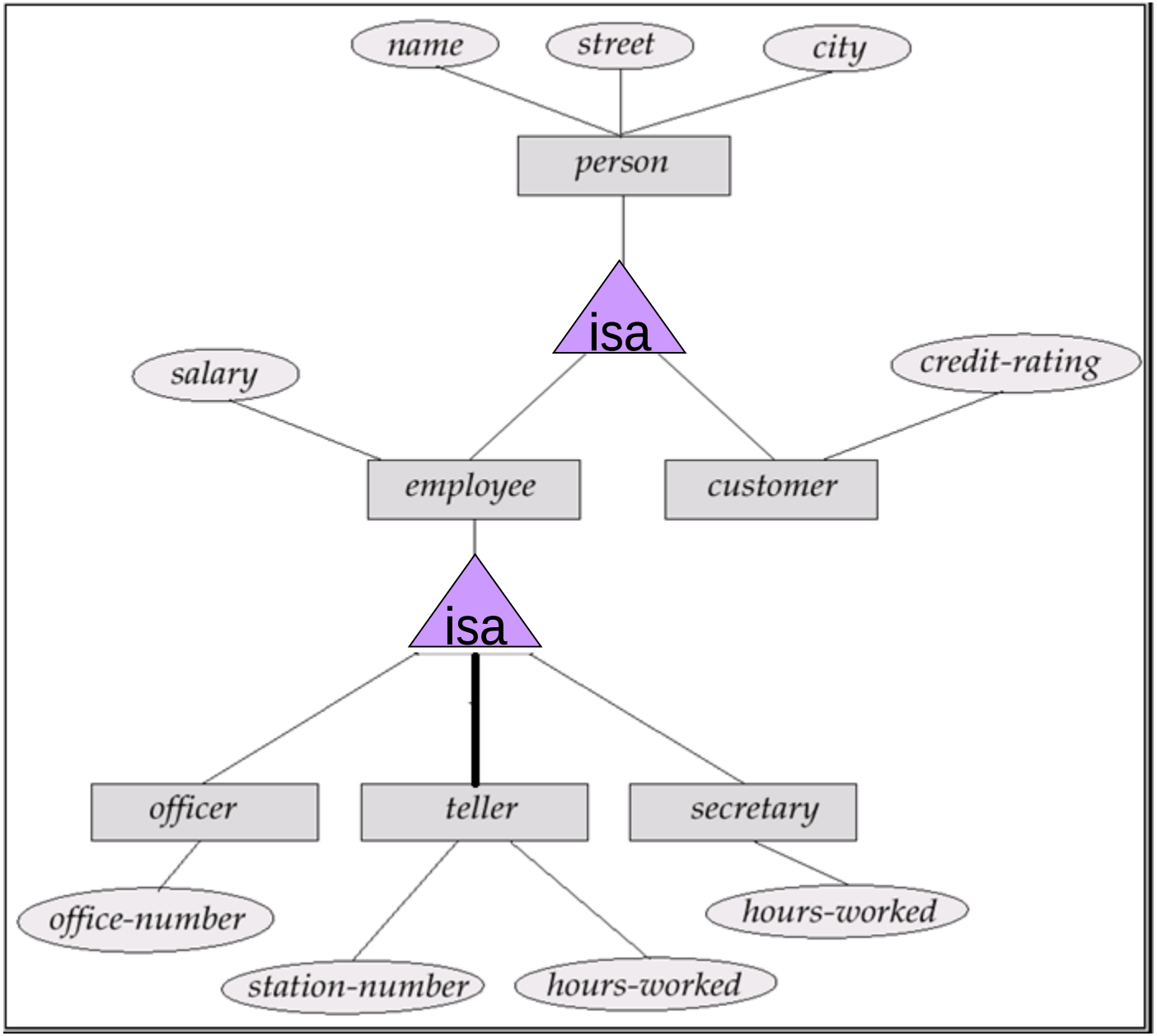


distinct

Assume subclasses form a tree.

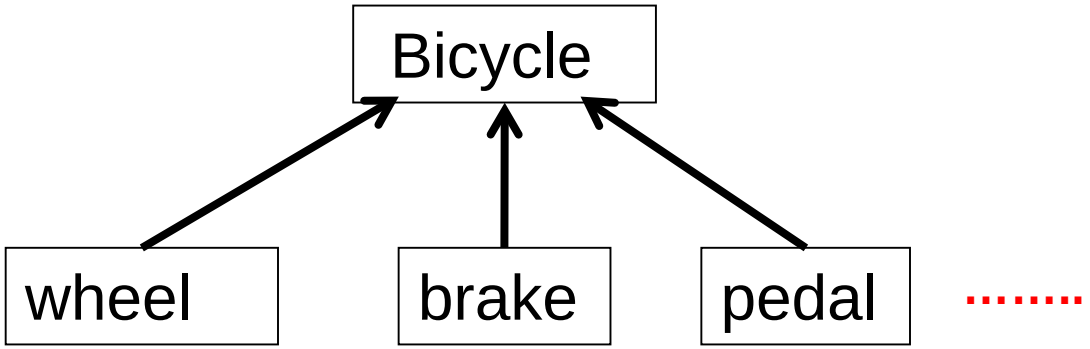
Isa triangles indicate the subclass relationship.
Point to the superclass.

Specialization Example

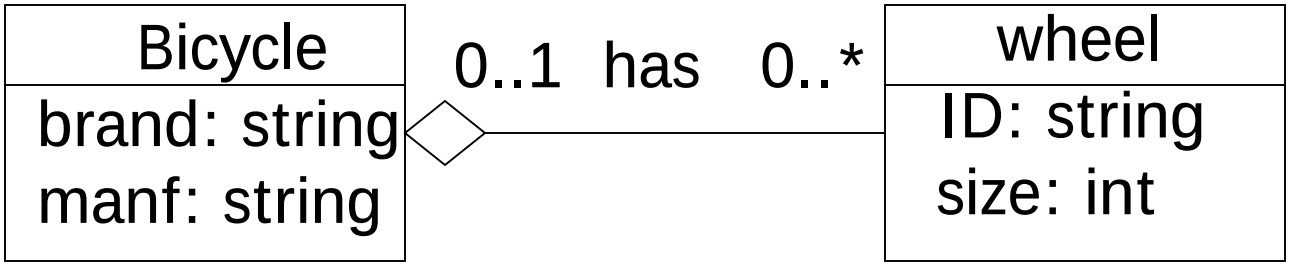


Aggregation (聚合)

- **Represents a ‘has-a’ or ‘is-part-of’ relationship between entity types, where one represents the ‘whole’ and the other ‘the part’.**



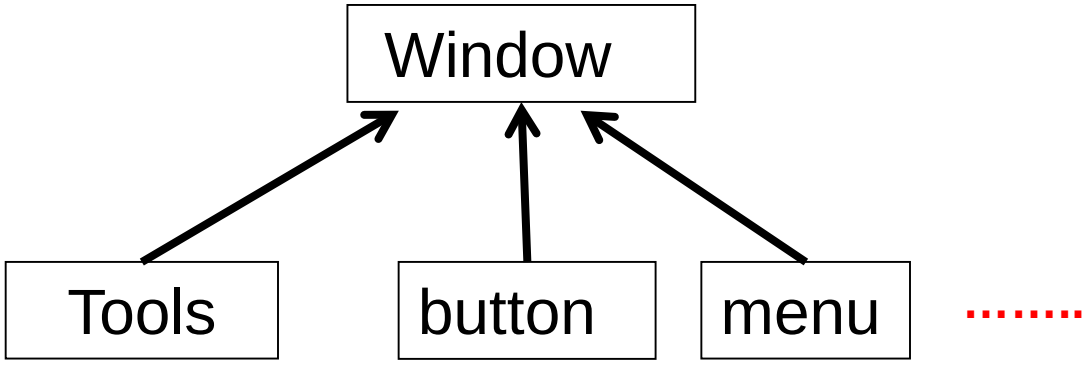
Example of Aggregation



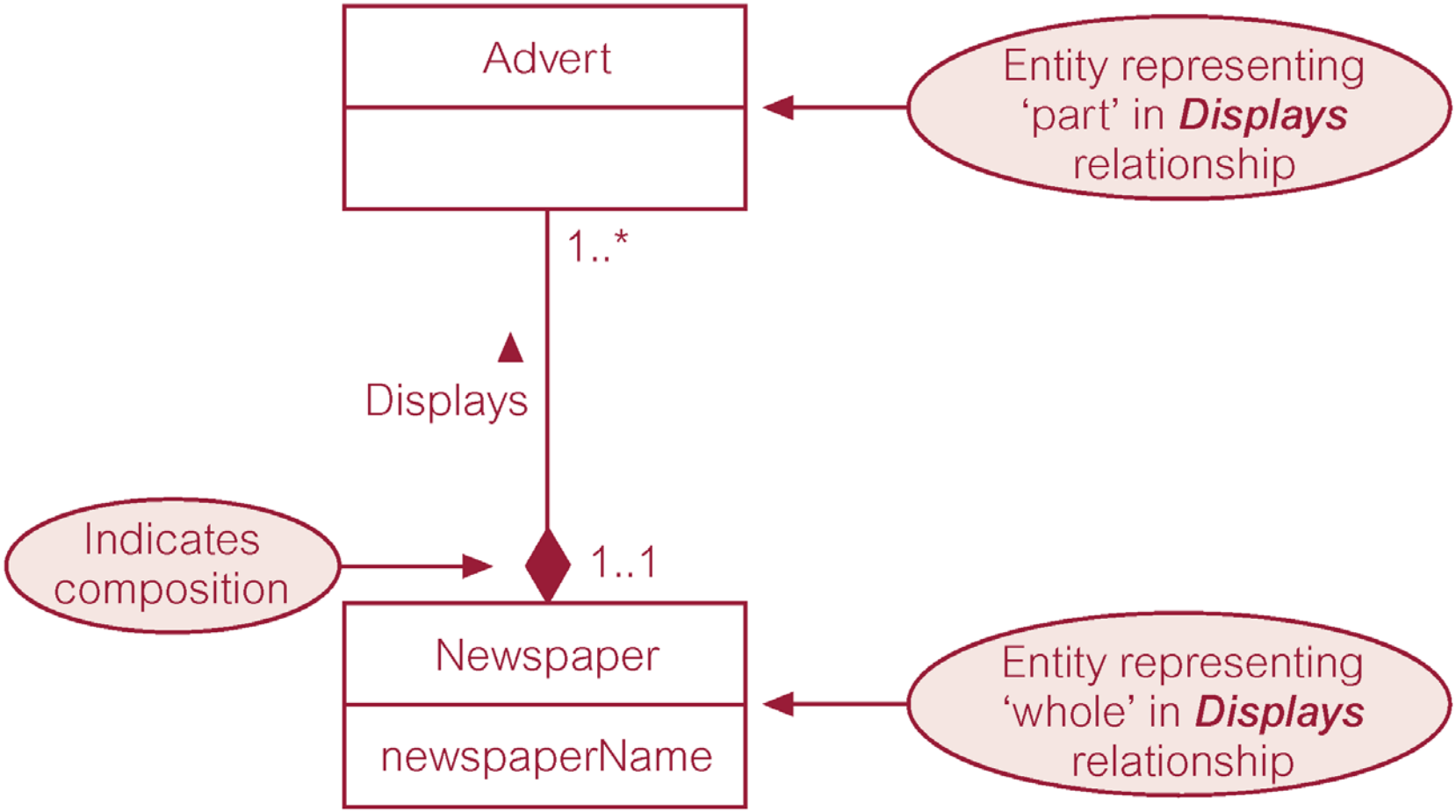
Represented by a diamond ◊ at the end of the connecting line, at the "owner" side.

Composition

- **It is a specific form of aggregation that represents an association between entities, where there is a strong ownership and coincidental lifetime (相同的生存期) between the 'whole' and the 'part'.**



Example of Composition



Represented by solid/filled-in diamond ♦ at owner.

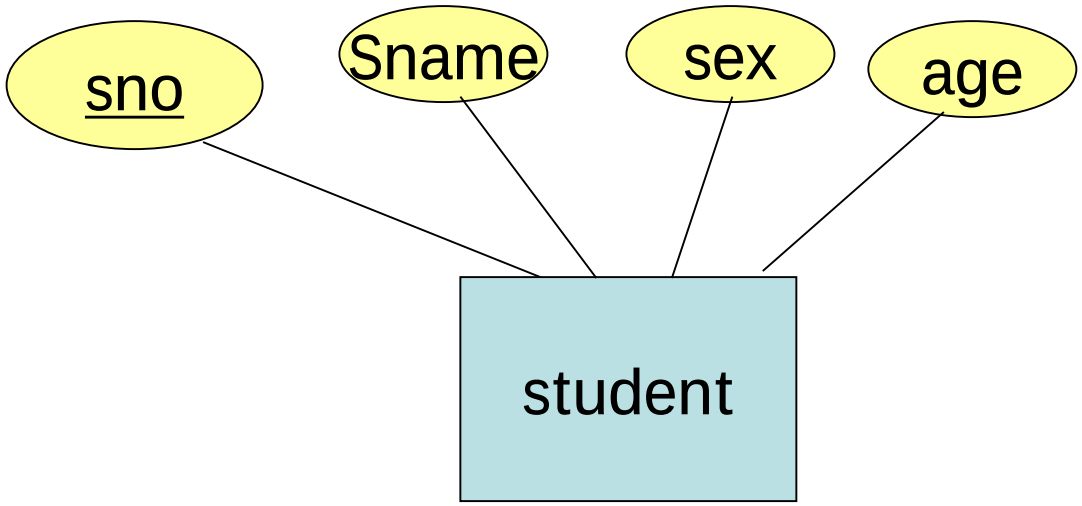
5.4 Rules of Mapping E-R model to tables

1. ***How to represent entities***

   For each entity in the ER model, create a table that includes all the entity's simple attributes.

   For composite attributes, **include only the simple attributes** that make up the composite attribute in the table. For example, for the composite **address** attribute, you would include its simple attributes **street, city, state, and zipCode**.

Entity Set -> Relation



Relation: student (sno, sname, sex, age)

Rules of Mapping E-R model to tables

2. *How to represent relationships*

- The relationship that an entity has with another entity is represented by the primary key/foreign key mechanism. (用主键/外键表示联系)
- In deciding where to *post (or place) the* foreign key attribute(s), you must first identify the 'parent' and 'child' entities involved in the relationship.
- The parent entity refers to the entity that posts a copy of its primary key into the table that represents the child entity, to act as the foreign key.

Rules of Mapping E-R model to tables

2. *How to represent relationships*

We consider the identification of parent/child entities for different types of relationships and for multi-valued attributes.

(a) one-to-one (1:1) binary relationships;

(b) one-to-many (1:*) binary relationships;

(c) one-to-many (1:*) recursive relationships

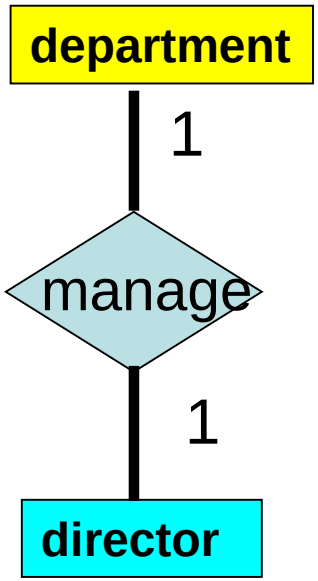
(d) many-to-many (*:*) binary relationships;

(a) One-to-one (1:1) binary relationships

- More frequently searched entity (**child**)
- Less frequently searched entity (**parent**)

As described above, a copy of the primary key of the parent entity is placed in the table representing the child entity as a foreign key.

1:1 Relationship -> Relation



Faculty (FacultyNo, SN, AGE, SEX, **DNO**)

department (DNO, DName, Tel, BuildingNo,

FacultyNo_of_director)

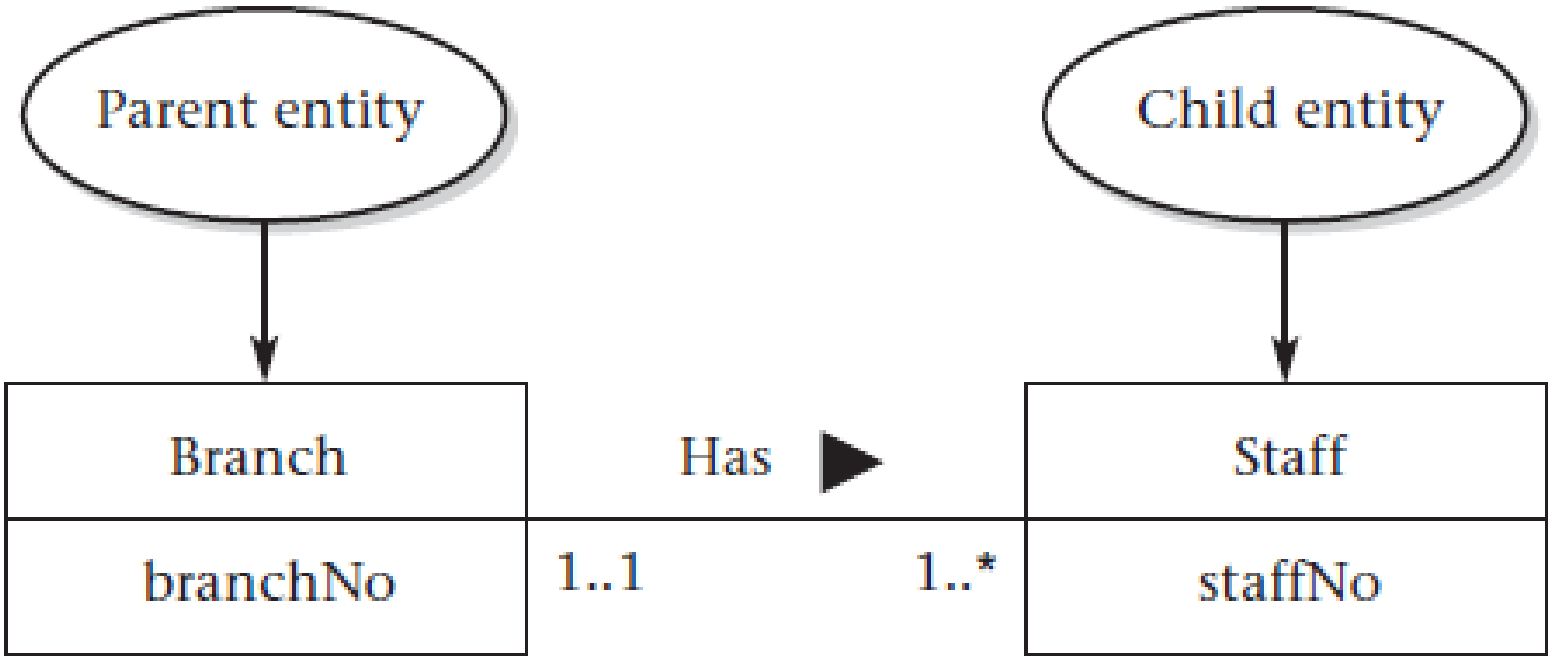
No need to construct a entity set for *director*

(b) One-to-many (1:*) binary relationships

For each 1:* binary relationship, the entity on the 'one side' of the relationship is designated as the parent entity and the entity on the 'many side' is designated as the child entity.

To represent this relationship, a copy of the primary key of the parent entity is placed into the table representing the child entity, to act as a foreign key.

One-to-many (1:*) binary relationships

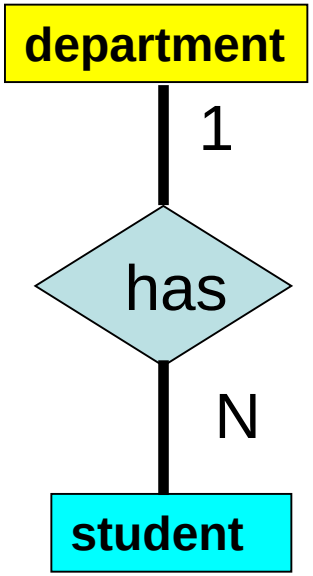


Branch (branchNo, street, city, state, zipCode)

Staff (staffNo, name, position, salary, branchNo)

Foreign Key branchNo references Branch(branchNo)

One to Many Relationship -> Relation

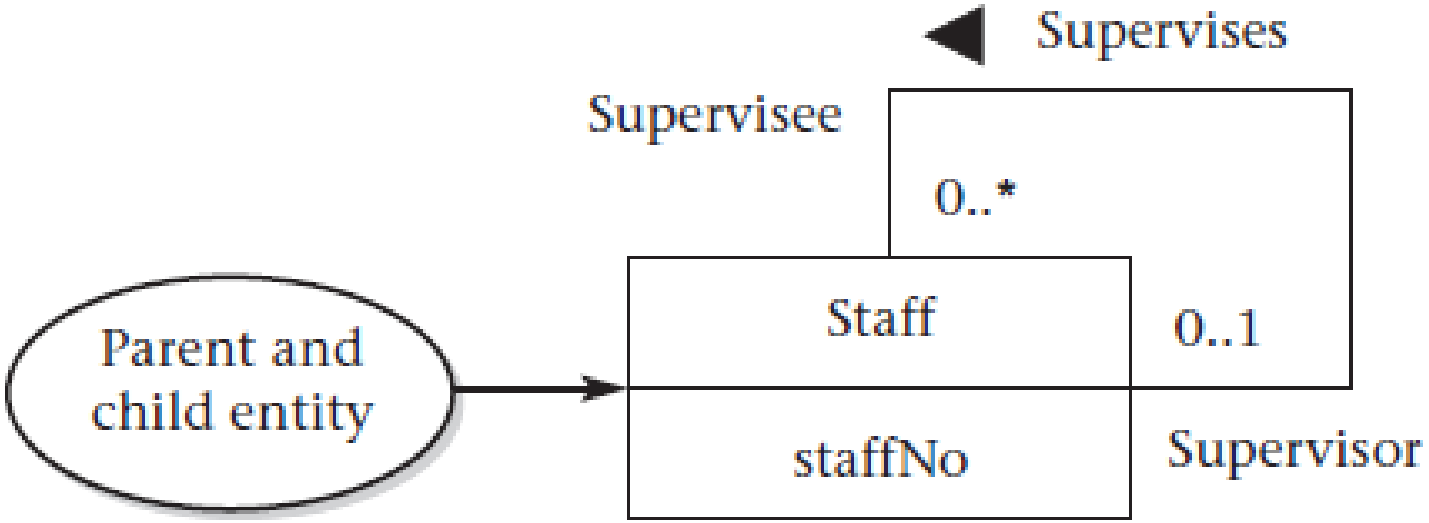


department (DNO, DName, Tel, BuildingNo)

Student (SNO, SN, AGE, SEX, **DNO**)

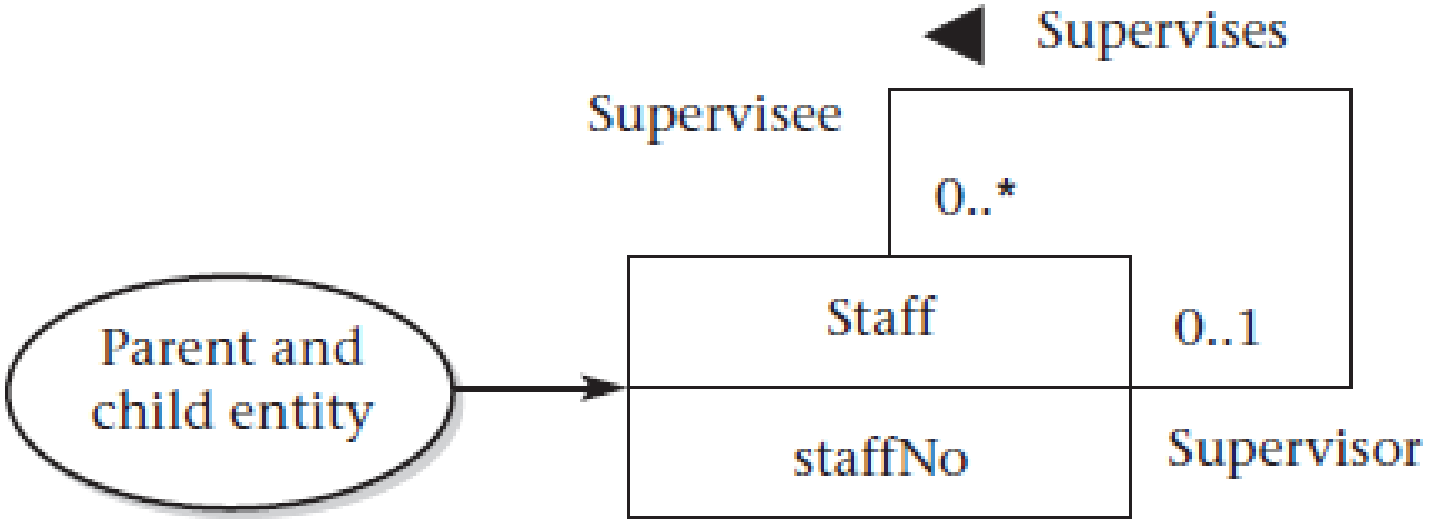
(c) One-to-Many (1:*) recursive relationships

The representation of a 1:* recursive relationship is shown in the following figure. There is a 1:* recursive relationship Staff *Supervises Staff.* In this case, **both the parent and the child entity is Staff**.

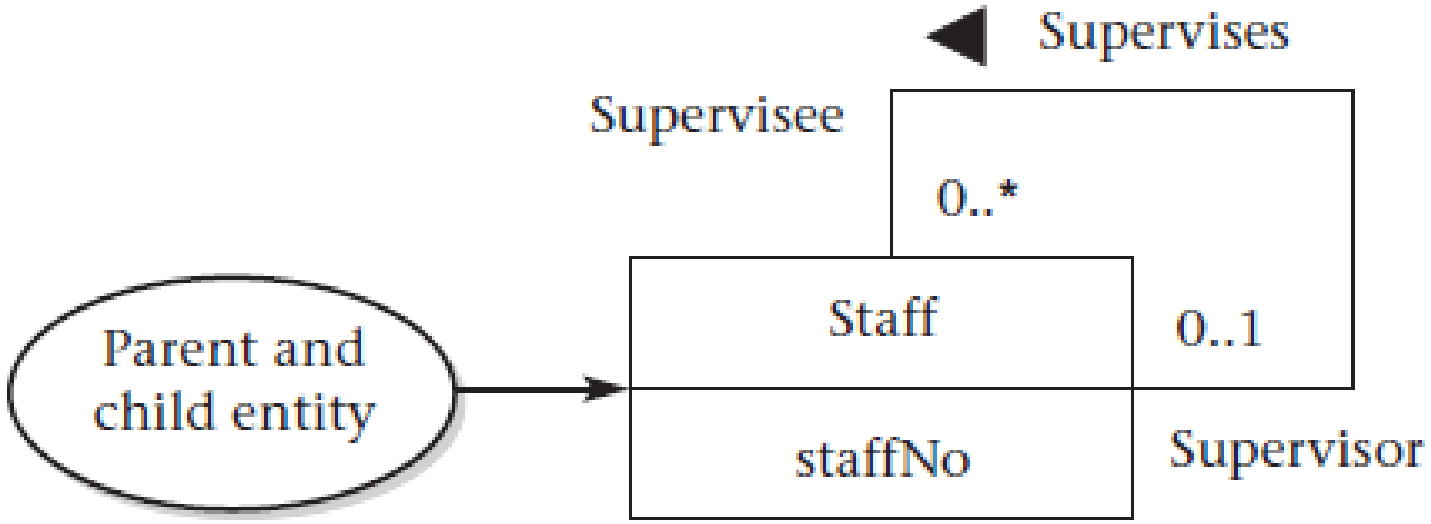


(c) One-to-Many (1:*) recursive relationships

Following the rule given above, you represent the *Supervises relationship by posting a copy of the primary* key of the Staff (parent) entity, staffNo, to the Staff (child) table, creating a second copy of this column to act as the foreign key. This copy of the column is renamed *supervisorStaffNo* to indicate its purpose.



One-to-Many (1:*) recursive relationships



Staff (staffNo, name, position, salary, branchNo, supervisorStaffNo)

Primary Key staffNo

Foreign Key branchNo references Branch(branchNo)

Foreign Key supervisorStaffNo references Staff(staffNo)

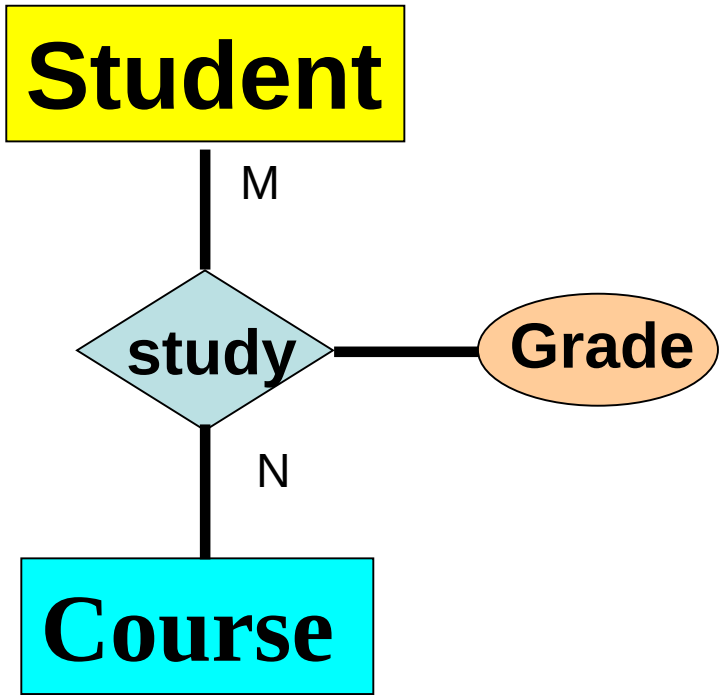
(d) Many-to-many (*:*) binary relationships

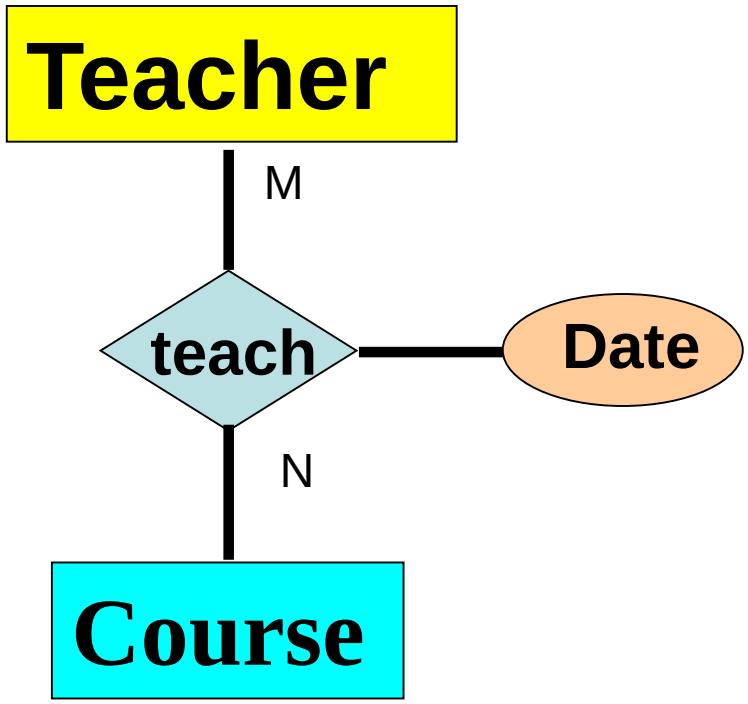
For each *:* binary relationship**, create a table to represent the relationship** and include any attributes that are part of the relationship.

We post a copy of the primary key attribute(s) of the entities that participate in the relationship into the new table, to act as foreign keys.

One or both of the foreign keys will also **form the primary key** of the new table, possibly in combination with some of the attributes of the relationship.

M:N





C(<u>Cno</u>, CN, CREDIT)

S(<u>Sno</u>, SN, AGE, SEX)

SC(<u>Sno,Cno</u>,G)

T(<u>Tno</u>, TN, AGE)

C(<u>Cno</u>, CN, CREDIT)

TC(<u>Tno, Cno</u>, DATE)

3. Schema Refinement

- Schema refinement: the modification of a schema to improve its design
  - Tables have **simple meaning**
  - Database has **less duplication** of information
  - Database has **fewer null** values
  - Database has **good performance**
- **Normalization** is the main approach for schema refinement
  - Need to understand functional dependency and keys first

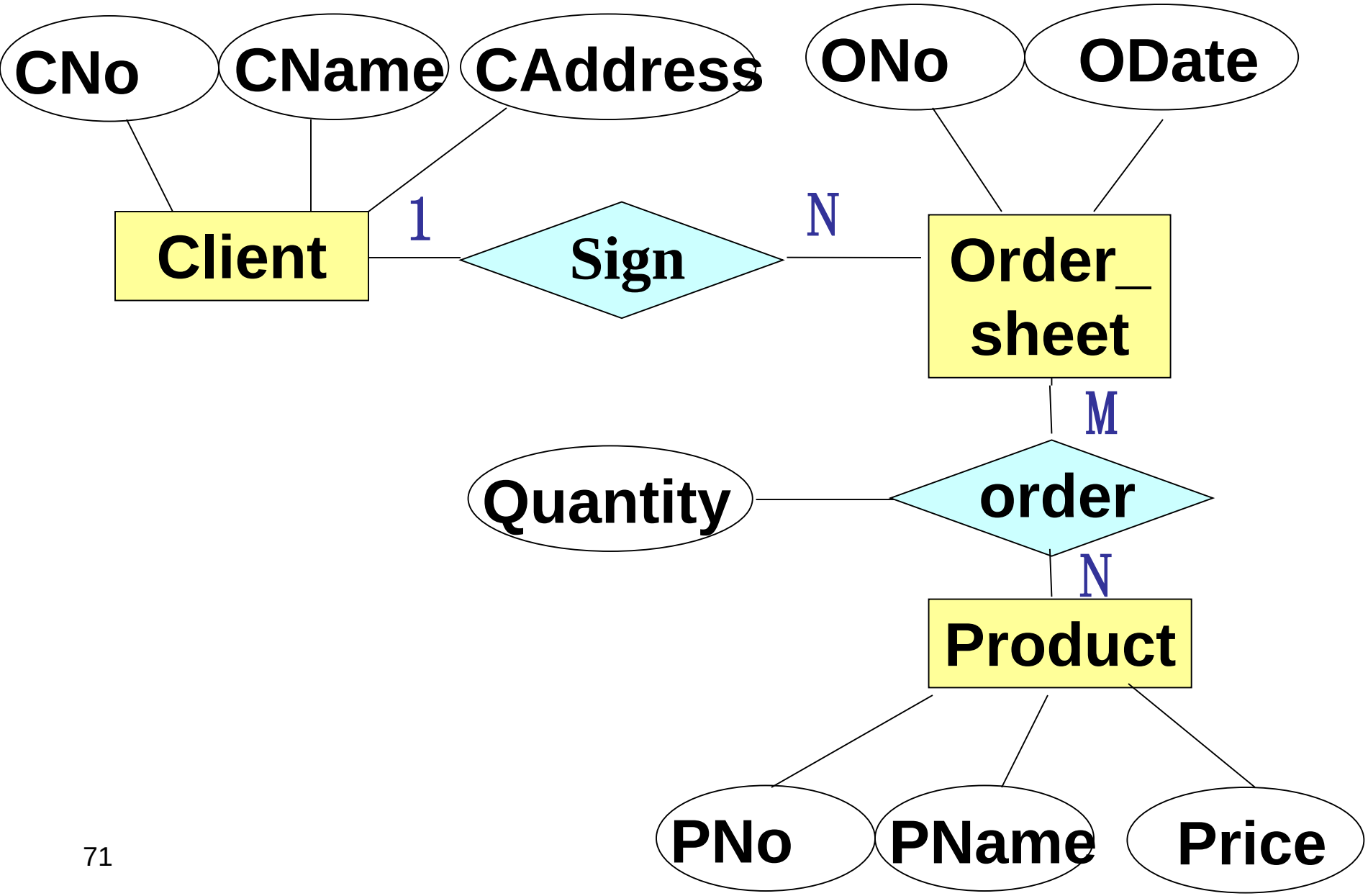
Example 1:

Suppose an organization needs to manage the following information: order number, client number, client name, client address, product number, product name, product price, ordered product quantity(Quantity), order date. One client can have many order sheets and there can be many products ordered in one order sheet.

(1)Please draw E-R model

(2)Transform the above E-R model to relational data model (logical model).

E-R Diagram：


CNo
CName
CAddress
ONo
ODate
Client
1
Sign
N
Order_
sheet
M
Quantity
order
N
Product
PNo
PName
Price

Relational Data Model

Client（CNo，CName，CAdrress）

Product（PNo，PName，Price）

Order_Sheet（ONo，CNo，ODate）

Order（ONo，PNo，Quantity）