数据库系统原理

Database System Principle

邵蓥侠

Email：shaoyx@bupt.edu.cn

北京邮电大学计算机学院

计算机应用技术中心

Part Two

Database Design

Chapter 6

Database Design and E-R Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML

Parts in Chapter 6

- Part 1. DB/DBS/DBAS design process (§ 6.1/6.10 + Supplementary)
  - DB design phases (§ 6.1)

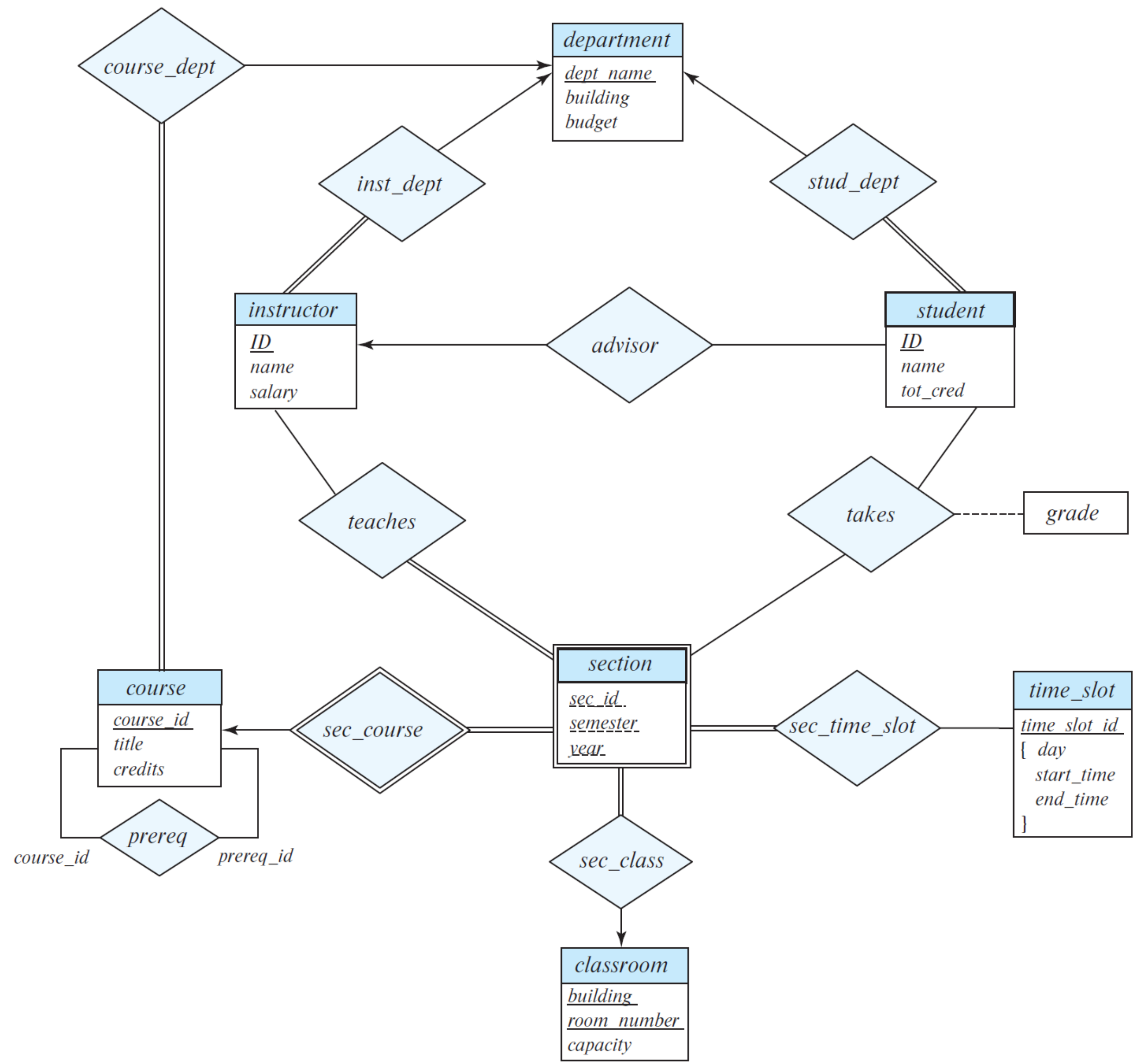
    requirement analysis, conceptual design, logical design, physical design
  - DBAS life-cycle model (生命周期模型)
- Part 2. The Entity-Relationship Model
  - *basic* E-R model
    - modeling elements (§ 6.2.1-6.2.3): entity sets, relationship sets, attributes
    - constraints (§ 6.3.1-6.3.3): mapping cardinality, participation constraint, keys
    - weak entity sets(§ 6.5.3)
    - removing redundant attributes in entity sets (§ 6.6)
    - E-R diagram(§ 6.2) and alternative notations(§ 6.10)

Parts in Chapter 6

- *extended* E-R Features (§ 6.8)
  - (§ 6.8.1-6.8.4) OO features in E-R model, i.e specialization, generalization, attributes inheritance, constraints on generalization
  - (§ 6.8.5) aggregation: relationship among relationships

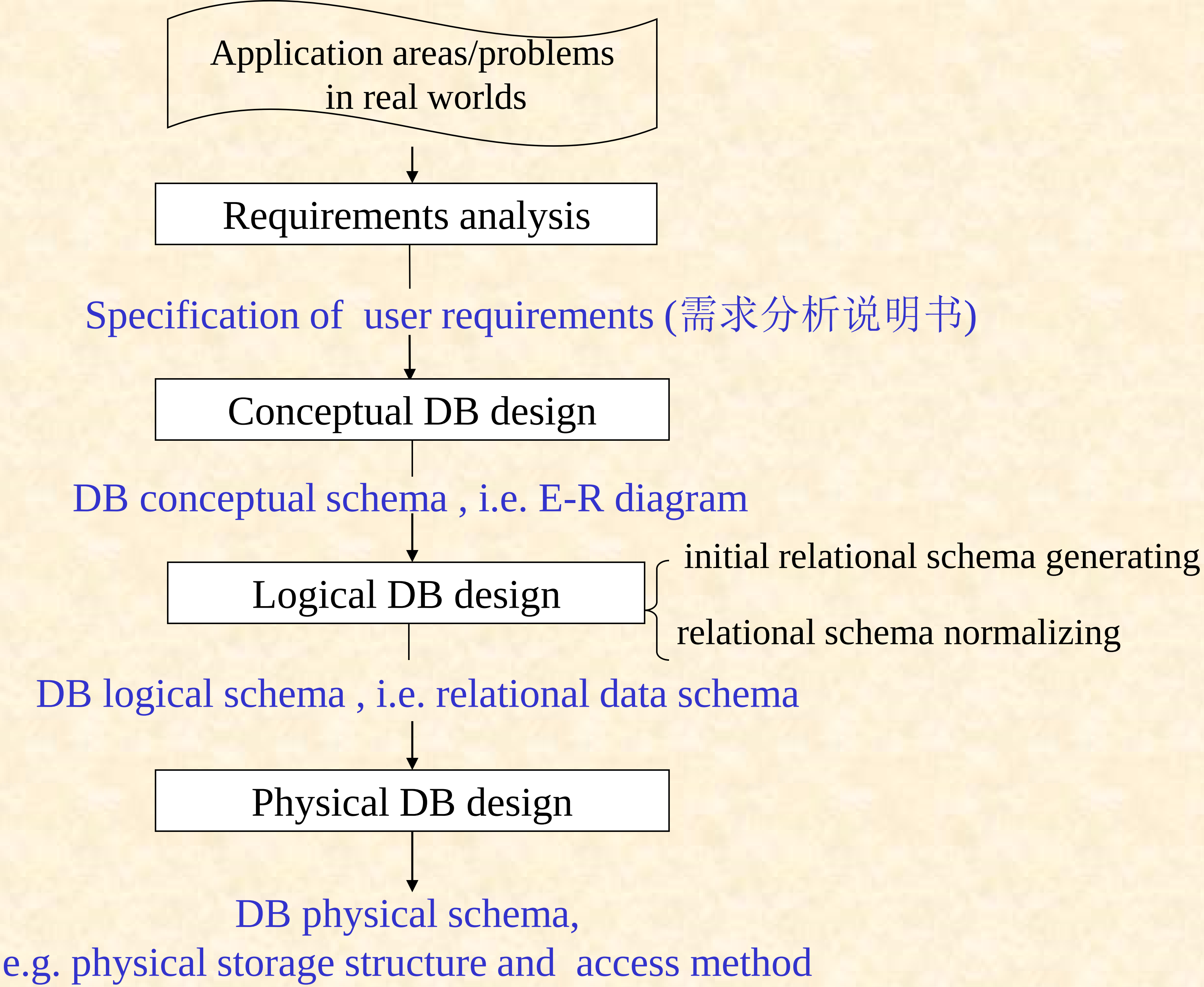
- Part 3. Reduction to Relational Schemas(§ 6.6, 6.8.6)
  - mapping elements in E-R model to that in relational models, i.e. conceptual schema → initial logical schema
- Part 4. E-R design issues (§ 6.9)
  - when applying E-R model to model the objects in real worlds, some issues (considerations and principles) should be addressed, to guarantee DBS effectiveness and efficiency for data

E-R Diagram for a University Enterprise



§ 6.1 Overview of the Design Process

- Database design consists of two sequential phases
  - analyzing of user requirements
    - *what data* should be stored in the database
    - what operations/transaction, such as *insert*, *delete*, *update* and *retrieve* are needed to conducted on these data
  - designing of DB schemas , in accordance with the *three-level of data abstract* (refer to Fig. 1.1)
    - conceptual design
    - logical design, at the logical level and view level
    - physical design, at the physical level
  - refer to Fig.1.0.1



Application areas/problems in real worlds

Requirements analysis

Specification of user requirements (需求分析说明书)

Conceptual DB design

DB conceptual schema , i.e. E-R diagram

Logical DB design

initial relational schema generating

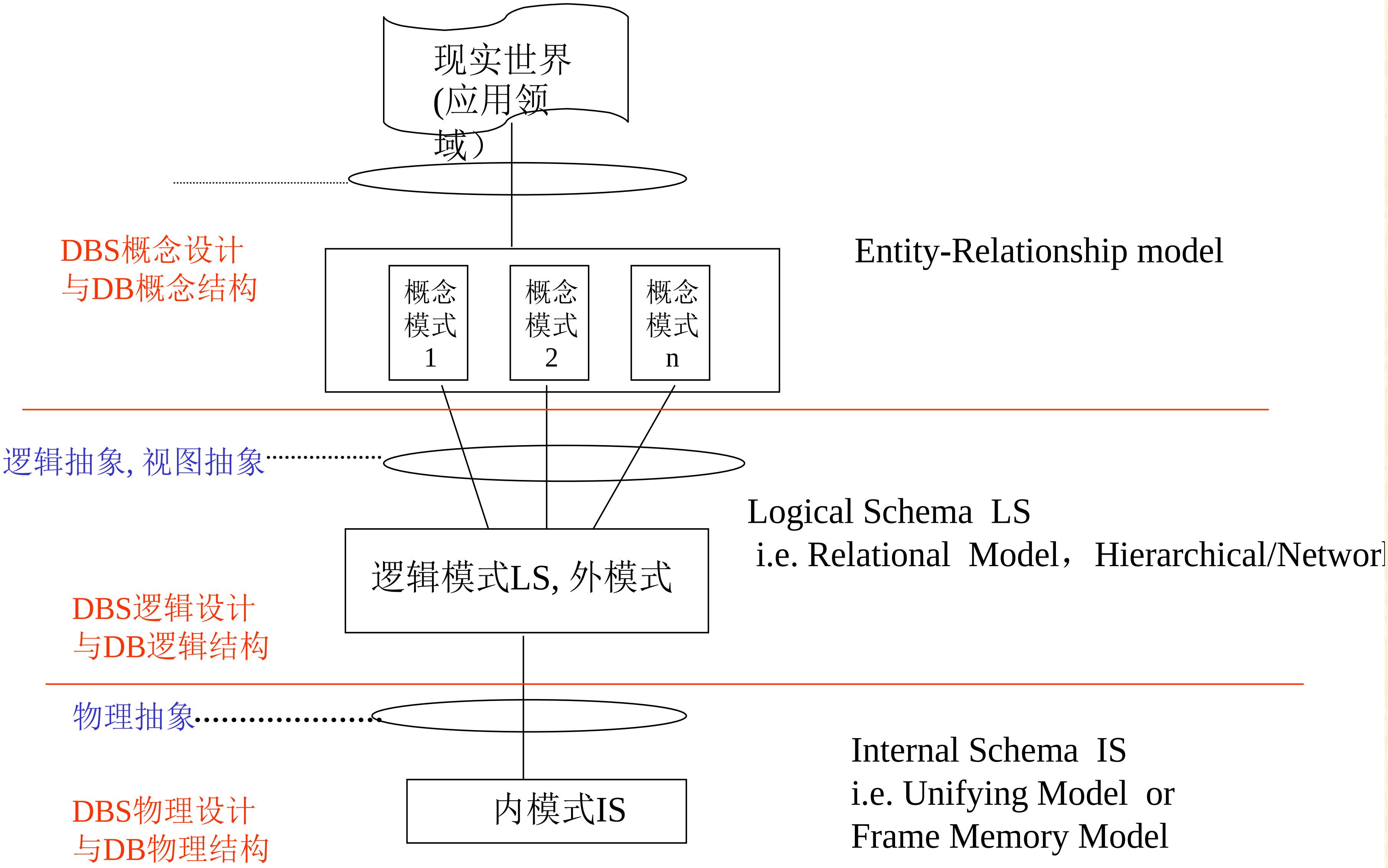
relational schema normalizing

DB logical schema , i.e. relational data schema

Physical DB design

DB physical schema,
e.g. physical storage structure and access method

Fig. 6.0.1 DB design phases



现实世界
(应用领
域）

DBS概念设计
与DB概念结构

概念
模式
1

概念
模式
2

概念
模式
n

Entity-Relationship model

逻辑抽象, 视图抽象

Logical Schema LS
i.e. Relational Model，Hierarchical/Networ

逻辑模式LS, 外模式

DBS逻辑设计
与DB逻辑结构

物理抽象

Internal Schema IS
i.e. Unifying Model or
Frame Memory Model

DBS物理设计
与DB物理结构

内模式IS

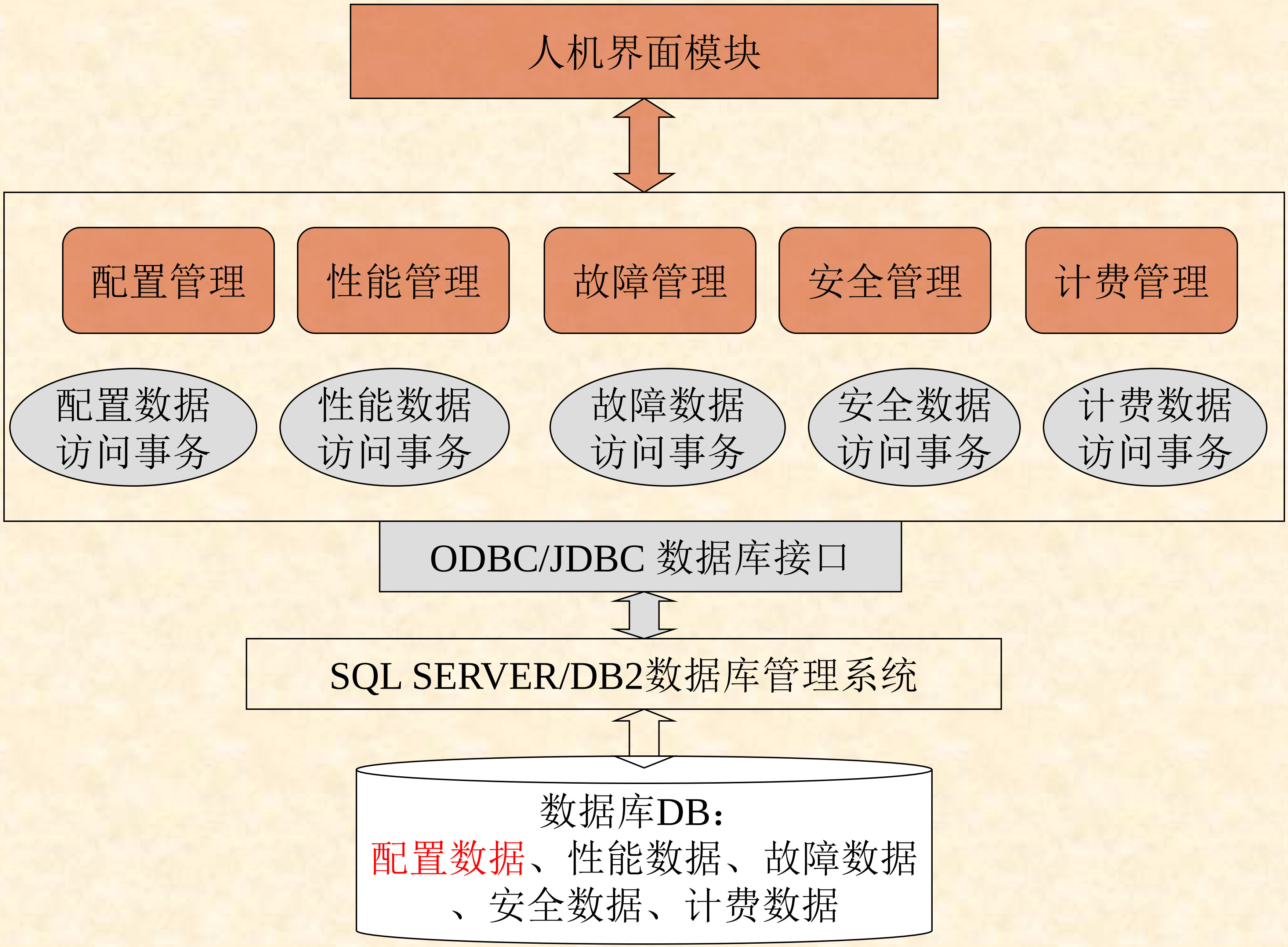
DBS设计与设计阶段： 数据模型—数据抽象—数据模式

Overview of the Design Process (cont.)

- 数据库应用系统DBAS设计
  - DB, DBMS, users, **application programs**
  - refer to Fig. 7.0.2
- DBAS设计
  - refer to Fig.7.0.3
  - 参照软件工程中软件开发瀑布模型原理，DBAS的生命周期由**项目规划、需求分析、系统设计、实现与部署、运行管理与维护**等5个基本活动组成
  - 根据DBAS的软件组成和各自功能，分为数据组织与存储设计、数据访问与处理设计、应用设计三条设计主线，分别用于设计数据库、数据库事务和应用程序

Overview of the Design Process (cont.)

- 根据数据库系统三级模式结构，DBAS设计阶段分为概念设计、逻辑设计、物理设计三个步骤，每一步设计内容涵盖了三条设计主线


人机界面模块

配置管理

性能管理

故障管理

安全管理

计费管理

配置数据访问事务

性能数据访问事务

故障数据访问事务

安全数据访问事务

计费数据访问事务

ODBC/JDBC 数据库接口

SQL SERVER/DB2数据库管理系统

数据库DB：

配置数据、性能数据、故障数据、安全数据、计费数据

Fig. 7.0.2 电信网管系统示意图

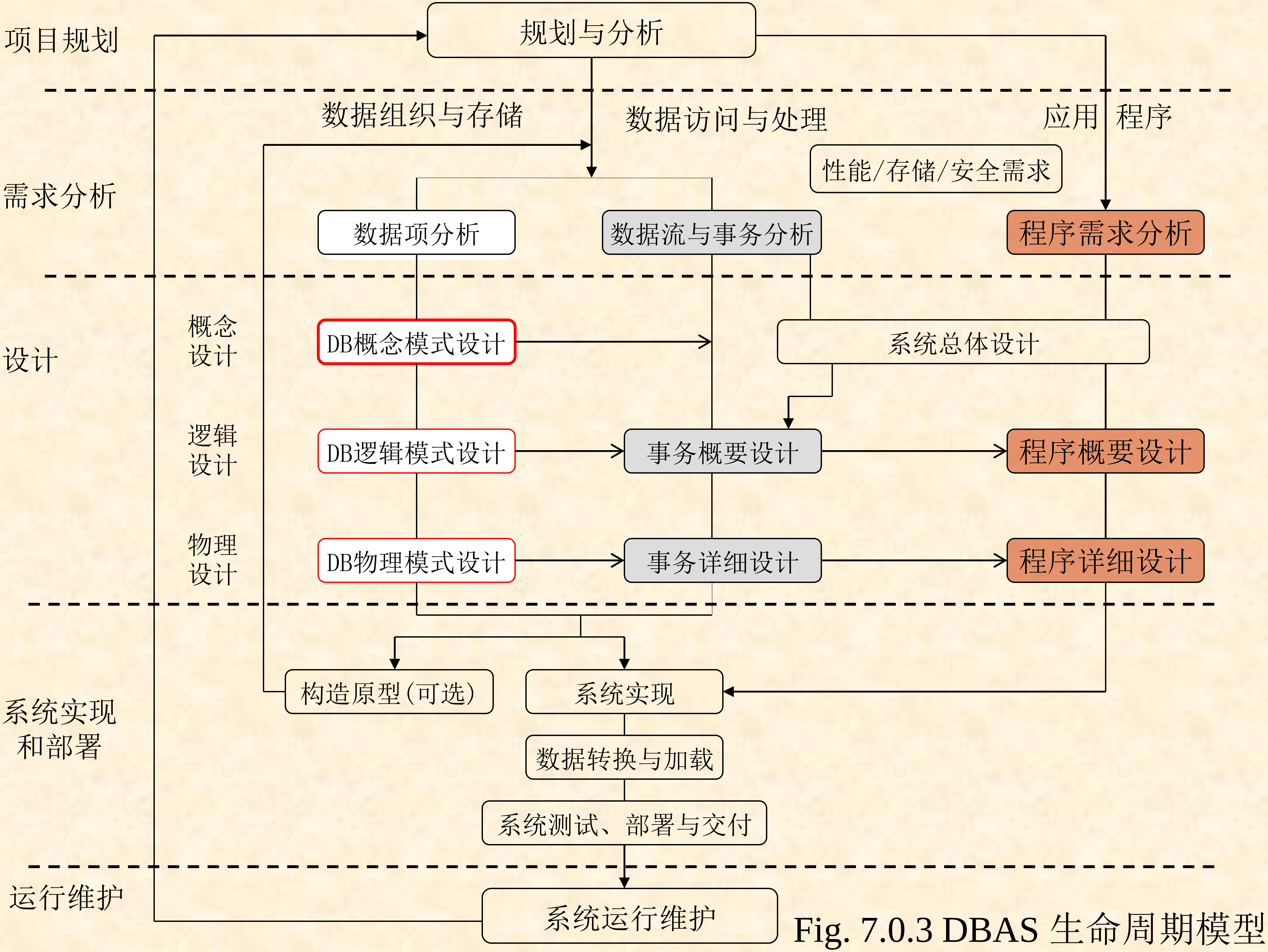
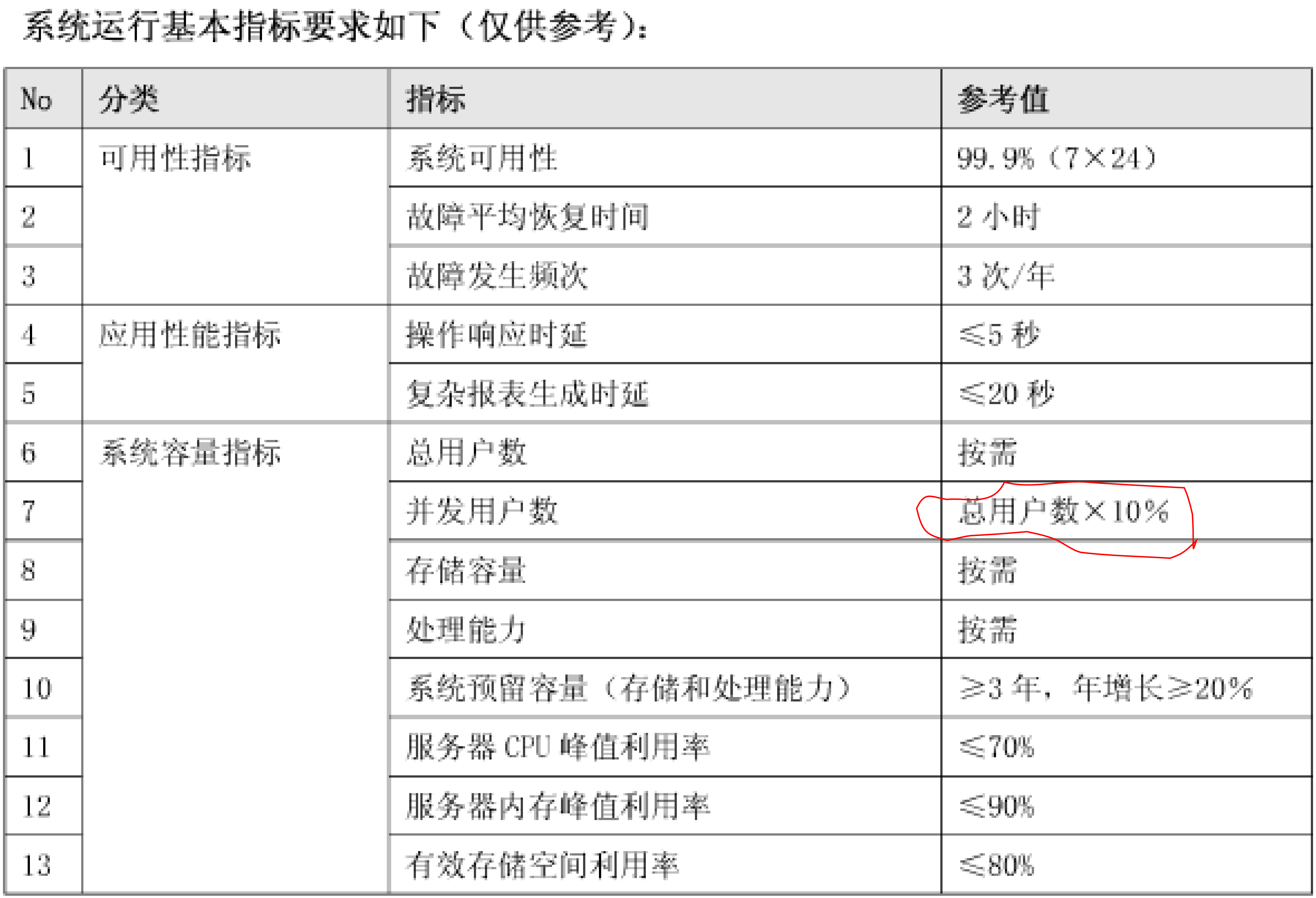


Fig. 7.0.3 DBAS 生命周期模型

中国移动网管支撑系统规范CMOSS2.0——性能指标要求



系统运行基本指标要求如下（仅供参考）：

No	分类	指标	参考值
1	可用性指标	系统可用性	99.9%（7×24）
2		故障平均恢复时间	2 小时
3		故障发生频次	3 次/年
4	应用性能指标	操作响应时延	≤5 秒
5		复杂报表生成时延	≤20 秒
6	系统容量指标	总用户数	按需
7		并发用户数	总用户数×10%
8		存储容量	按需
9		处理能力	按需
10		系统预留容量（存储和处理能力）	≥3 年，年增长≥20%
11		服务器 CPU 峰值利用率	≤70%
12		服务器内存峰值利用率	≤90%
13		有效存储空间利用率	≤80%

6.2 The Entity-Relationship Model

- The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.
- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model.
- The ER data model employs three basic concepts:
    - entity sets,
    - relationship sets,
    - attributes.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.

6.2.1 Entity Sets（实体集，实体型）

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- 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
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (*ID*, *name*, *street*, *city*, *salary*)
    course= (*course_id*, *title*, *credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.

Entity Sets (cont.)

- An entity consists of all values of its all attributes
  - e.g. Fig 6.2

- Entity-set

customer-set = {< ID-value, *name*-value, *street* - value,

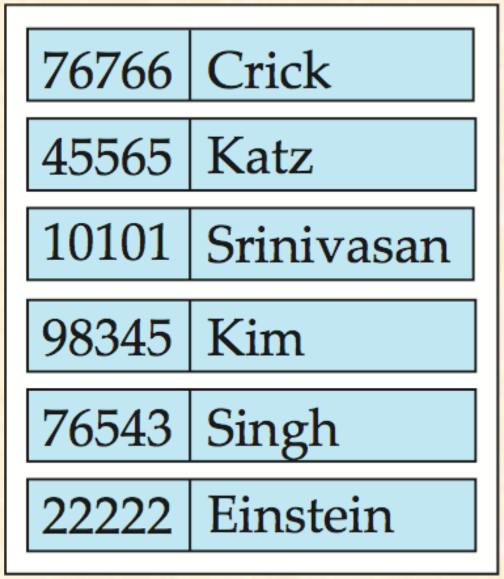
city -value>

| ID-value $\in D_1$, name-value $\in D_2$,

street -value $\in D_3$, city -value D_4}

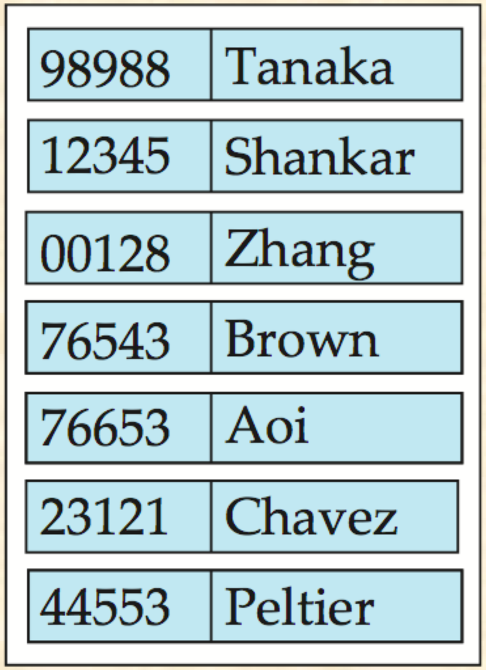
$\subseteq D_1 \times D_2 \times D_3 \times D_4$

Entity Sets



instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor



student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

§ 6.2.2 Relationship Sets

- A relationship is an association among several entities
  - e.g. in Fig. 6.2

Jone	*borrower*	L-17
customer	relationship	*loan*
entity set		*entity set*

- A Relationship set （联系集，联系型） is a set of relationship of the same type
  - *note:* more than one relationship set among the same entity sets

§ 6.2.2 Relationship Sets

- A **relationship** is an association among several entities

  Example:

  44553 (Peltier) *advisor* 22222 (Einstein)

  student entity relationship set *instructor* entity

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

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

  where $(e_1, e_2, \ldots, e_n)$ is a relationship

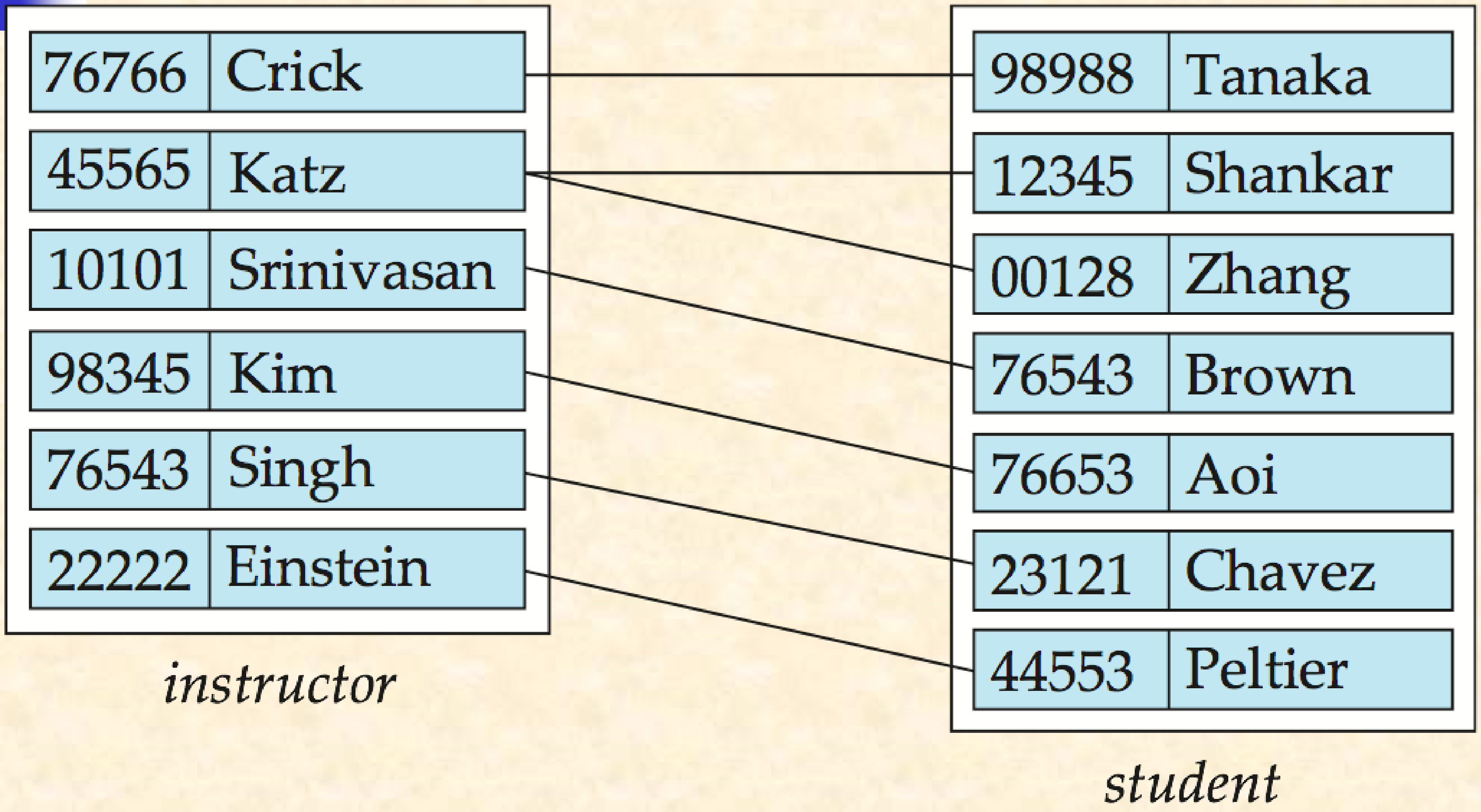
  - Example:

    (44553,22222) $\in$ *advisor*

Relationship Sets (cont.)

- The entity sets E_1, E_2, …, E_n participate in relationship set R
- A relationship instance in an E-R schema represents an association between the named entities in the real-world enterprise that is being modeled
  - e.g. in Fig. 6.2, the relationship instance between *Crick* and the *Tanaka*

Relationship Set *advisor*



76766 Crick
45565 Katz
10101 Srinivasan
98345 Kim
76543 Singh
22222 Einstein

instructor

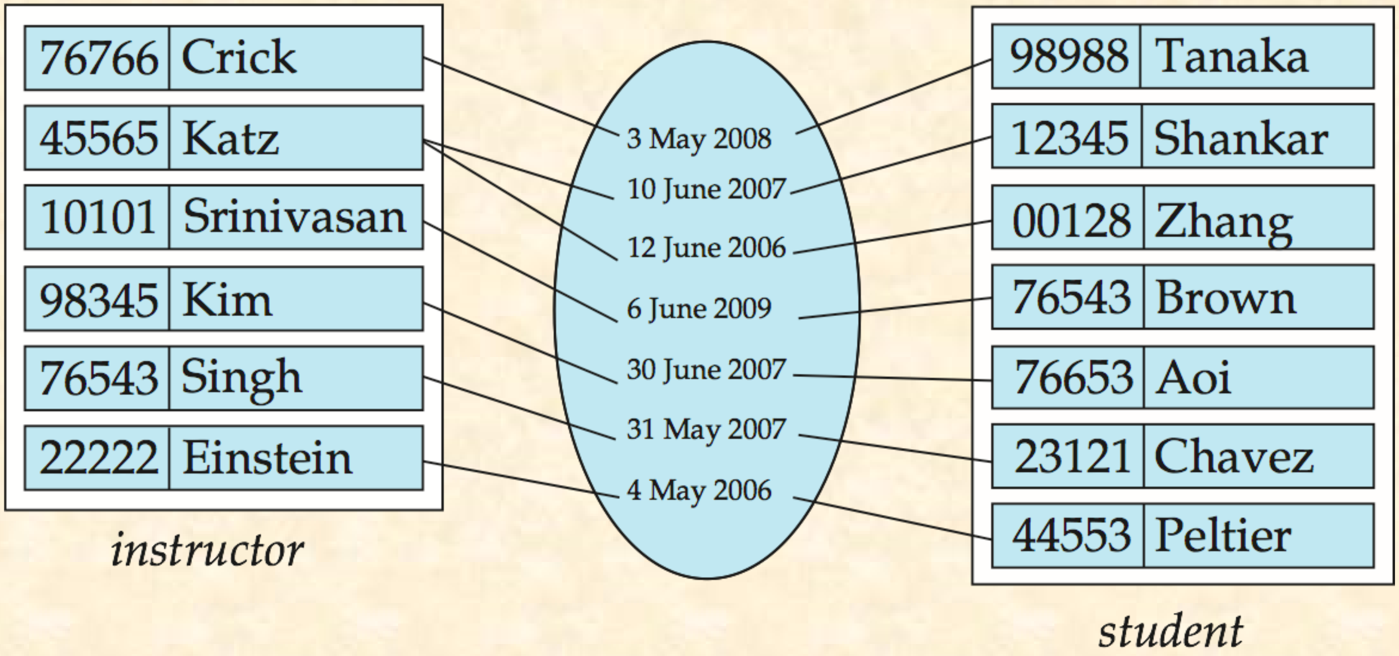
98988 Tanaka
12345 Shankar
00128 Zhang
76543 Brown
76653 Aoi
23121 Chavez
44553 Peltier

student

Fig.6.2

Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

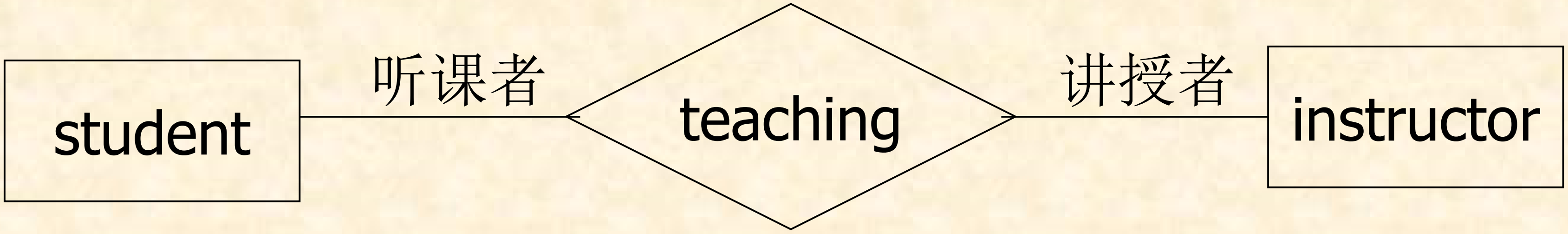
student

Degree of a Relationship Set

- binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*

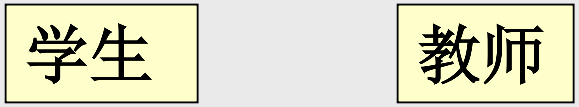
Relationship Sets (cont.)

- Role
  - the functions that an entity plays in a relationship is called that entity's role

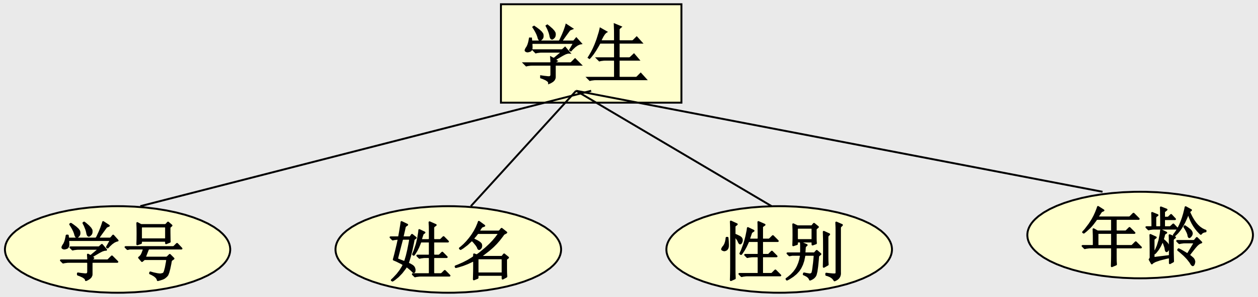


The Entity-Relationship Model

1） 用长方形表示实体集，长方形内写明实体集名。

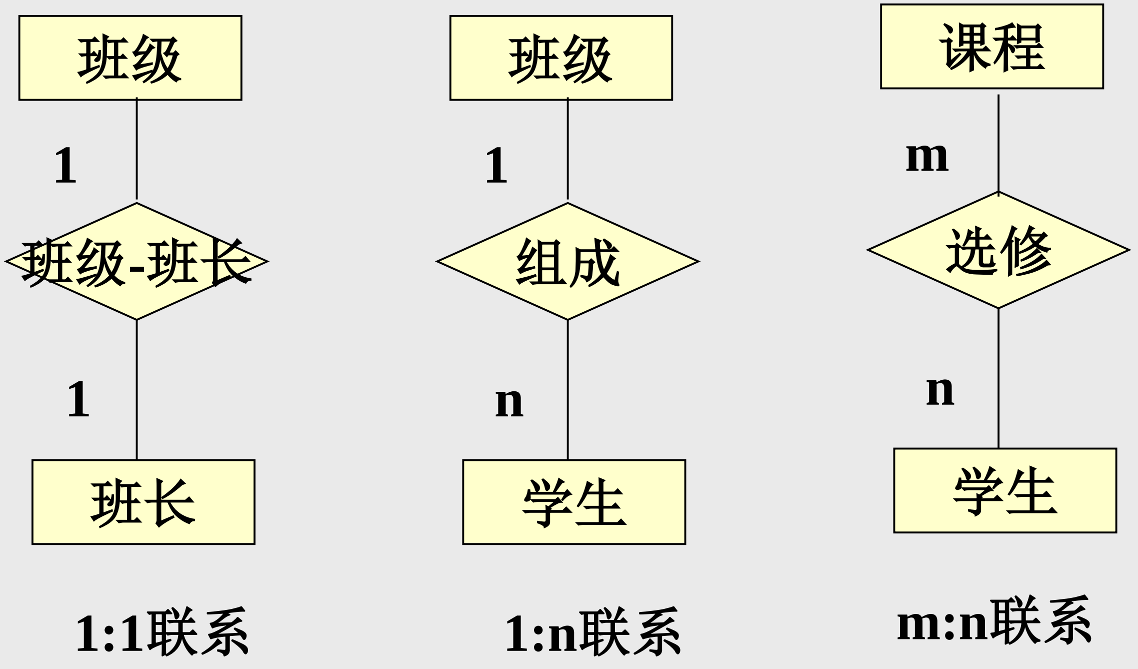


2） 用椭圆形表示实体集的属性，并用线段将其与相应的实体集连接起来。



The Entity-Relationship Model

3） 用菱形表示实体集间的联系，菱形内写上联系名，用线段分别与有关实体集连接起来，在线段旁标出联系的类型（1:1、1:n或m:n）。

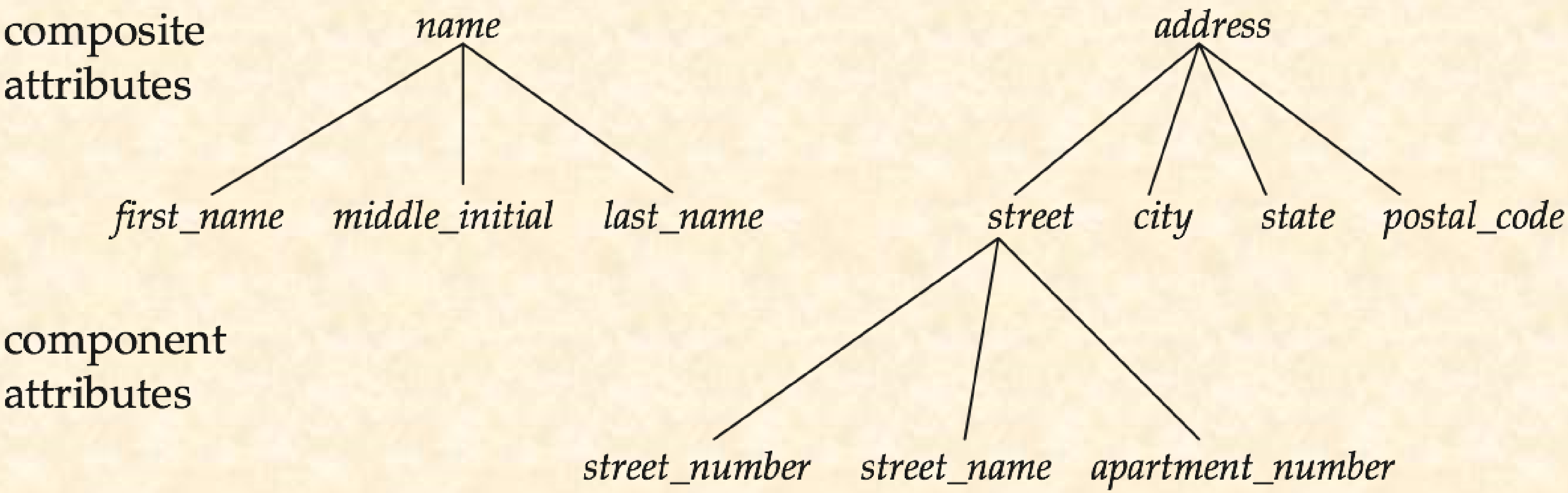


§ 6.3 Attributes

- The *domain* or *value set* of the attribute
  - the set of permitted values for the attribute

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

Composite Attributes



composite attributes

component attributes

name

first_name

middle_initial

last_name

address

street

city

state

postal_code

street_number

street_name

apartment_number

Attributes (cont.)

- Null value for an attribute means
  - the attribute “ not applicable “ for the entity, not existing
  - the value for the attribute exists, but is “unknown”

Redundant Attributes

- Suppose we have entity sets:
  - *instructor*, with attributes: *ID*, *name*, *dept_name*, *salary*
  - *department*, with attributes: *dept_name*, *building*, *budget*
- We model the fact that each instructor has an associated department using a relationship set *inst_dept*
- The attribute *dept_name* appears in both entity sets. Since it is the primary key for the entity set *department*, it replicates information present in the relationship and is therefore redundant in the entity set *instructor* and needs to be removed.
- BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later.

6.4 Mapping Cardinalities

- Mapping cardinalities
  - *semi-quantitatively* expressing the number of entities to which another entity can be associated via a relationship set
- For a binary relationship set ***R***, the mapping cardinality must be one of the following types: Fig.6.9, Fig.6.10, ***from A to B***
  - *one to one*
  - *one to many*
  - *many to one*
  - *many to many*

6.4 Mapping Cardinalities

- Mapping cardinalities
  - *semi-quantitatively* expressing the number of entities to which another entity can be associated via a relationship set
- For a binary relationship set ***R***, the mapping cardinality must be one of the following types: Fig.6.9, Fig.6.10, ***from A to B***
  - *one to one*
    - An entity in A is associated with *at most one* entity in B, and an entity in B is associated with *at most one* entity in A.
  - *one to many*
    - An entity in A is associated with *any number (zero or more)* of entities in B. An entity in B, however, can be associated with *at most one* entity in A.

–注意many-to-one和one-to-many 定义和方向!

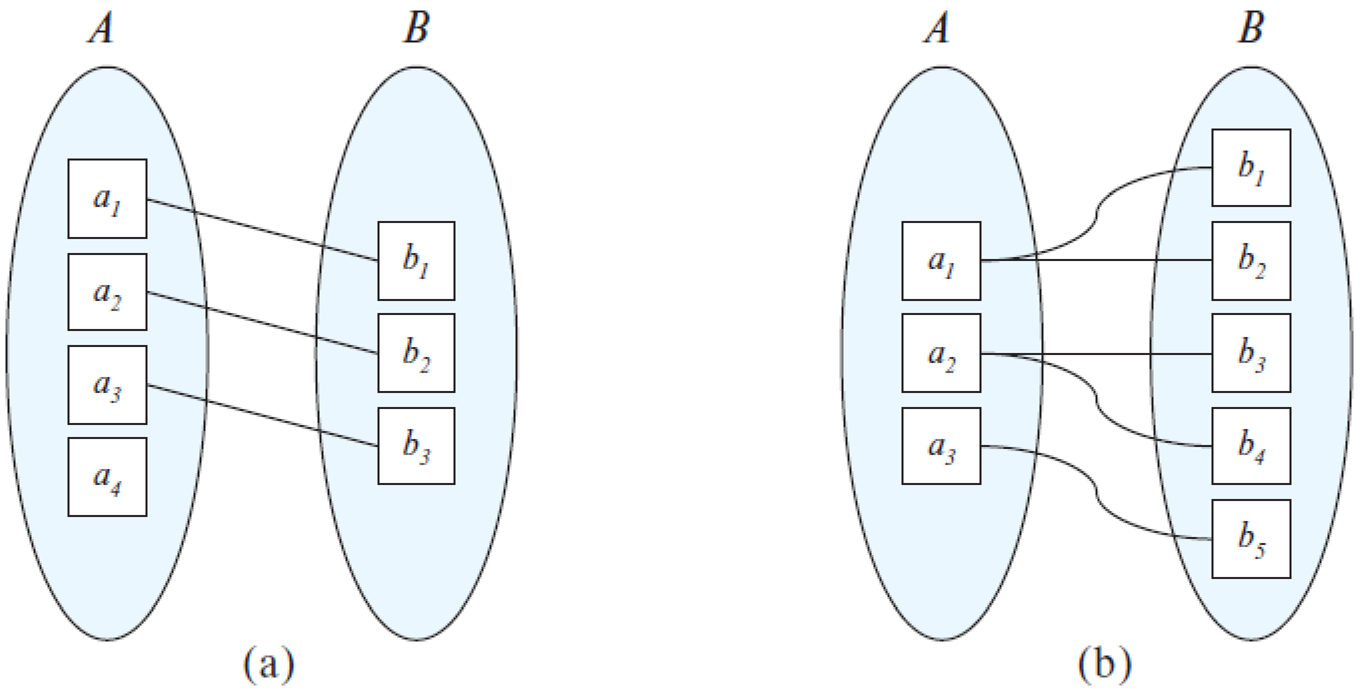


Figure 6.9 Mapping cardinalities. (a) One-to-one. (b) One-to-many.

Note: some elements in ***A*** or ***B*** may not be mapped to any elements in the other set

Fig. 6.9 Mapping cardinalities

6.4 Mapping Cardinalities

- Mapping cardinalities
  - *semi-quantitatively* expressing the number of entities to which another entity can be associated via a relationship set
- For a binary relationship set ***R***, the mapping cardinality must be one of the following types: Fig.6.9, Fig.6.10, ***from A to B***
  - *many to one*
    - An entity in A is associated with *at most one* entity in B. An entity in B, however, can be associated with *any number (zero or more)* of entities in A.
  - *many to many*
    - An entity in A is associated with *any number (zero or more)* of entities in B, and an entity in B is associated with *any number (zero or more)* of entities in A.

–注意many-to-one和one-to-many 定义和方向！

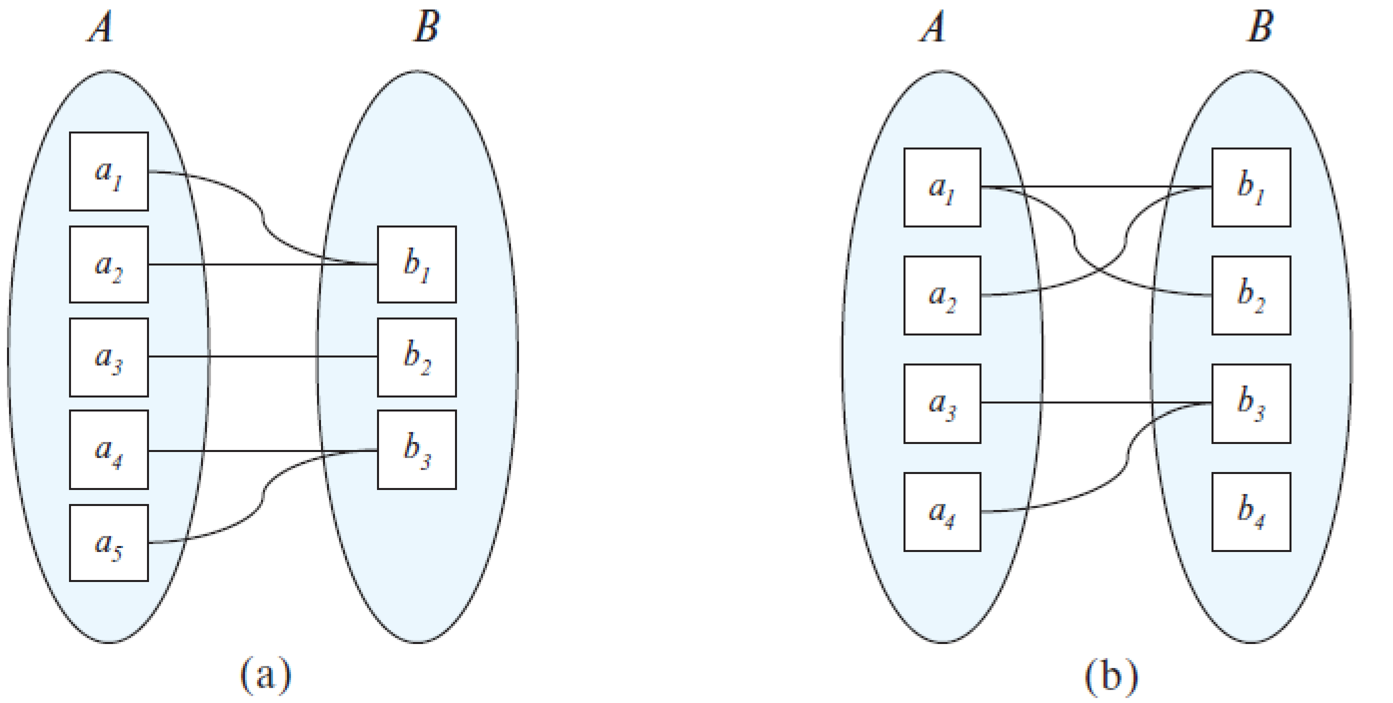


Figure 6.10 Mapping cardinalities. (a) Many-to-one. (b) Many-to-many.

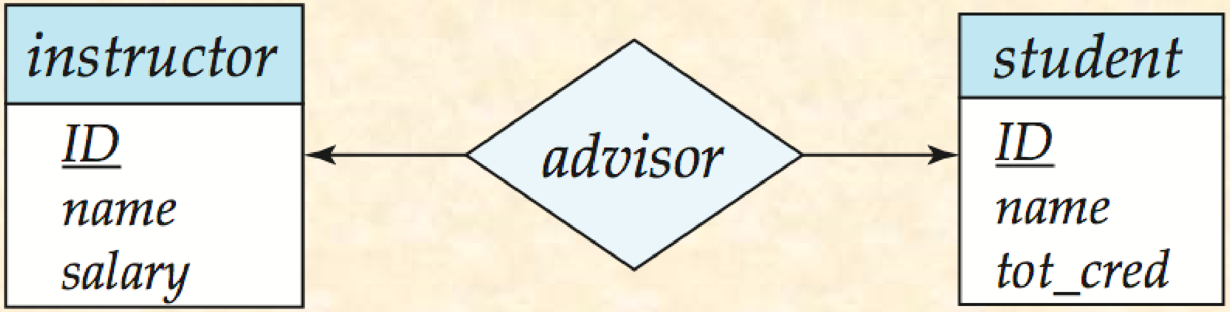
Note: Some elements in ***A*** or ***B*** may not be mapped to any elements in the other set

Fig. 6.10 Mapping cardinalities

Cardinality Constraints

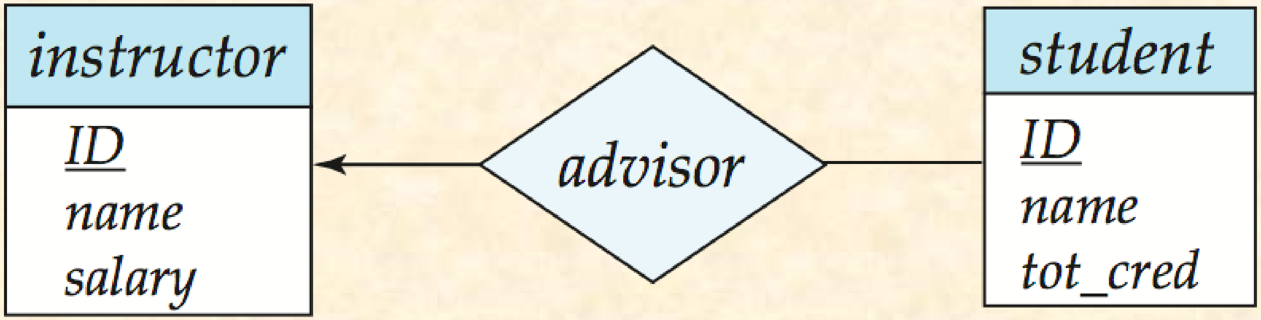
- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*



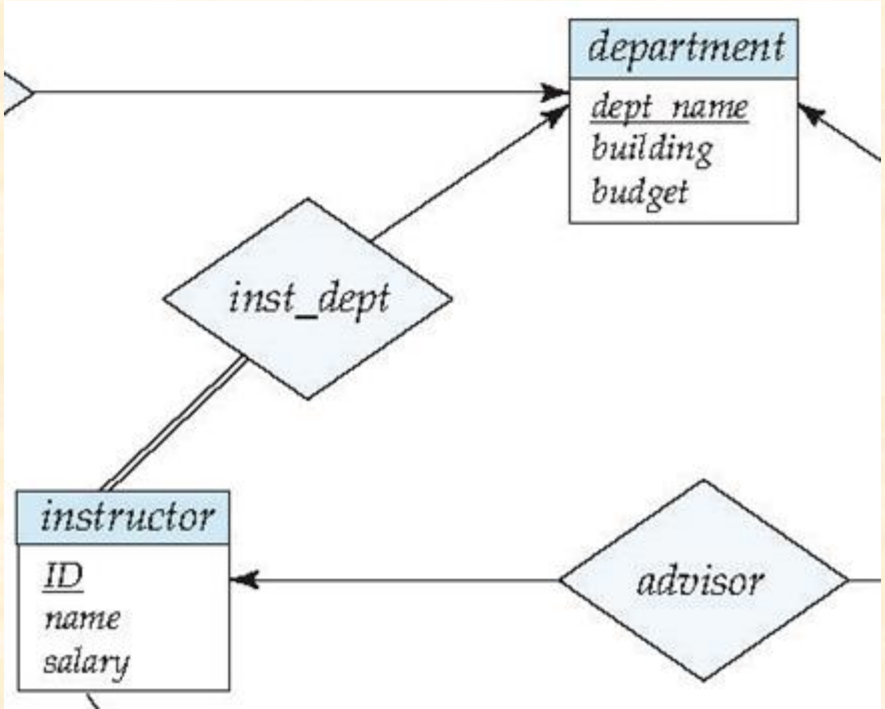
One-to-Many Relationship

- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



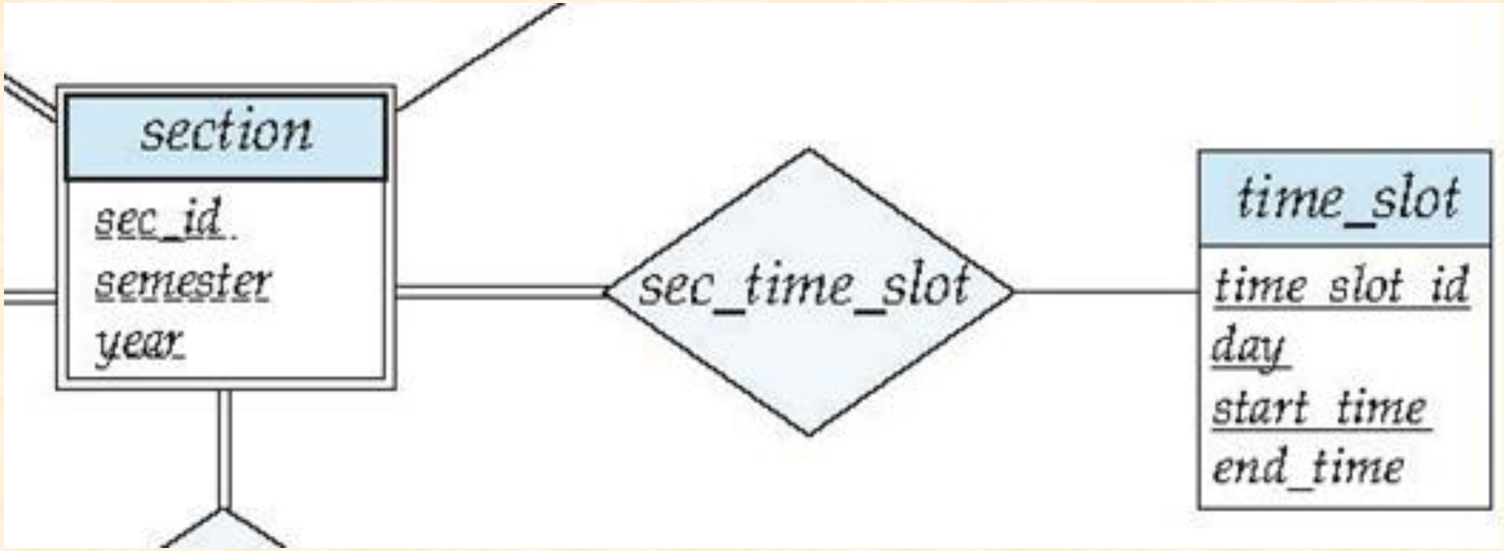
Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *department*,
  - an instructor is associated with at most one department via *inst_dept*,
  - and a department is associated with several (at least one) instructor via *inst_dept*



Many-to-Many Relationship

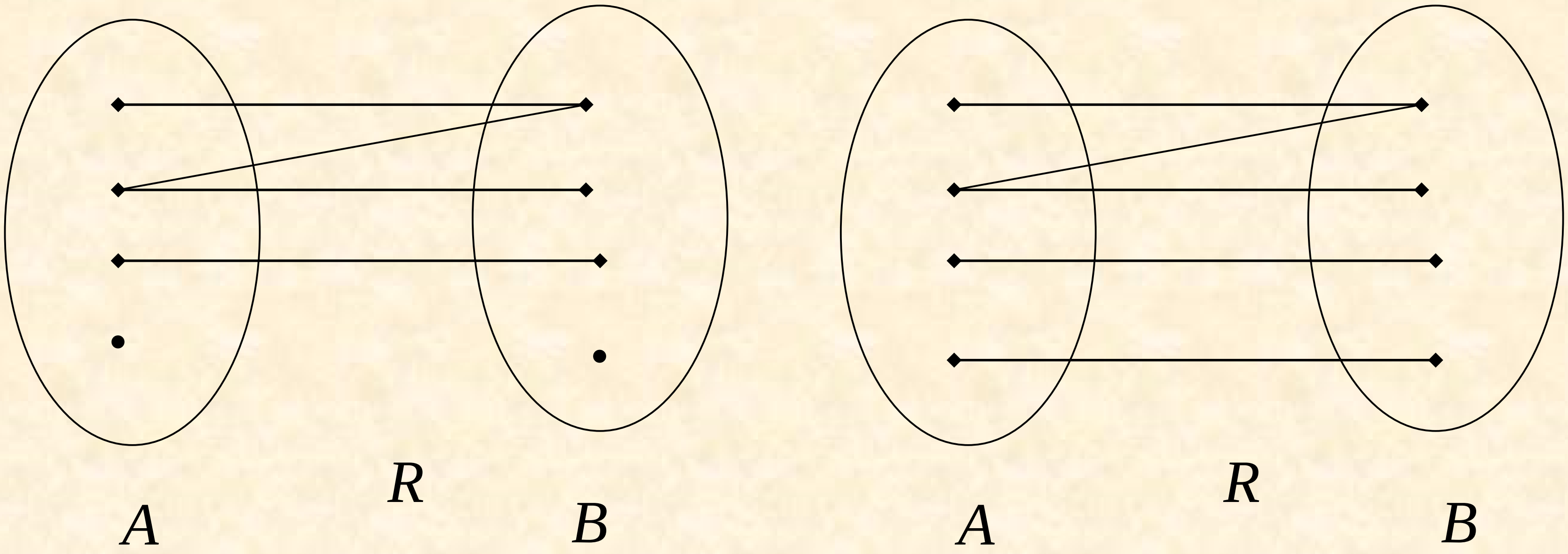
- A *section* is associated with several (possibly 0) timeslots via *sec_time_slot*
- A *timeslot* is associated with several (possibly 0) sections via *sec_time_slot*



Participate Constraints

- The participation of an entity *E* in a relationship *R* is *total*, if
  - every entity in *E* participates in at least one relationship in *R*
  - e.g. participation of *instructor* in *inst_dept* is total
    - refer to Fig.6.15 ▷
    - every *instrctor* must have a *department* associated to it via *inst_dept*
- The participation of an entity *E* in a relationship *R* is *partial*, if
  - some entities in *E* may not participate in any relationship in *R*
  - e.g. participation of *student* in *advisor* is partial, because some students maybe have no advisors
    - refer to Fig.6.15▷

Participate Constraints (cont.)



(a) partial participation

(b) total participation

Fig. 7.0.4 Total/participation participation

Cardinality Limits for Participation

- Cardinality limits (参与的基数界限) are used to express *quantitative* constraints on participation
- *E.g.* ***instructor***, ***student***, ***advisor***
  - 每个*student*最少有1个指导*instructor*，最多也只有1个指导*instructor*
  - 每个instructor最多可以指导多个*student*, 最少可以指导0个*student*

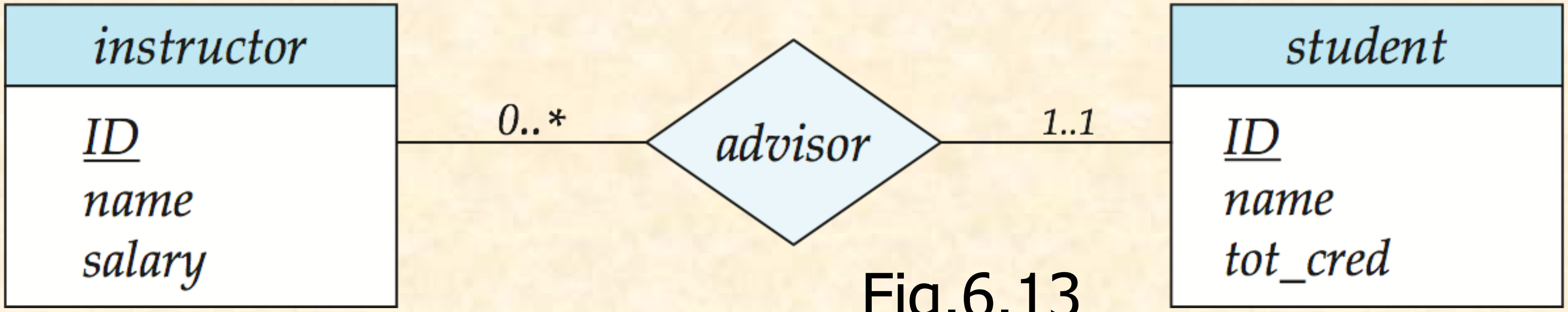
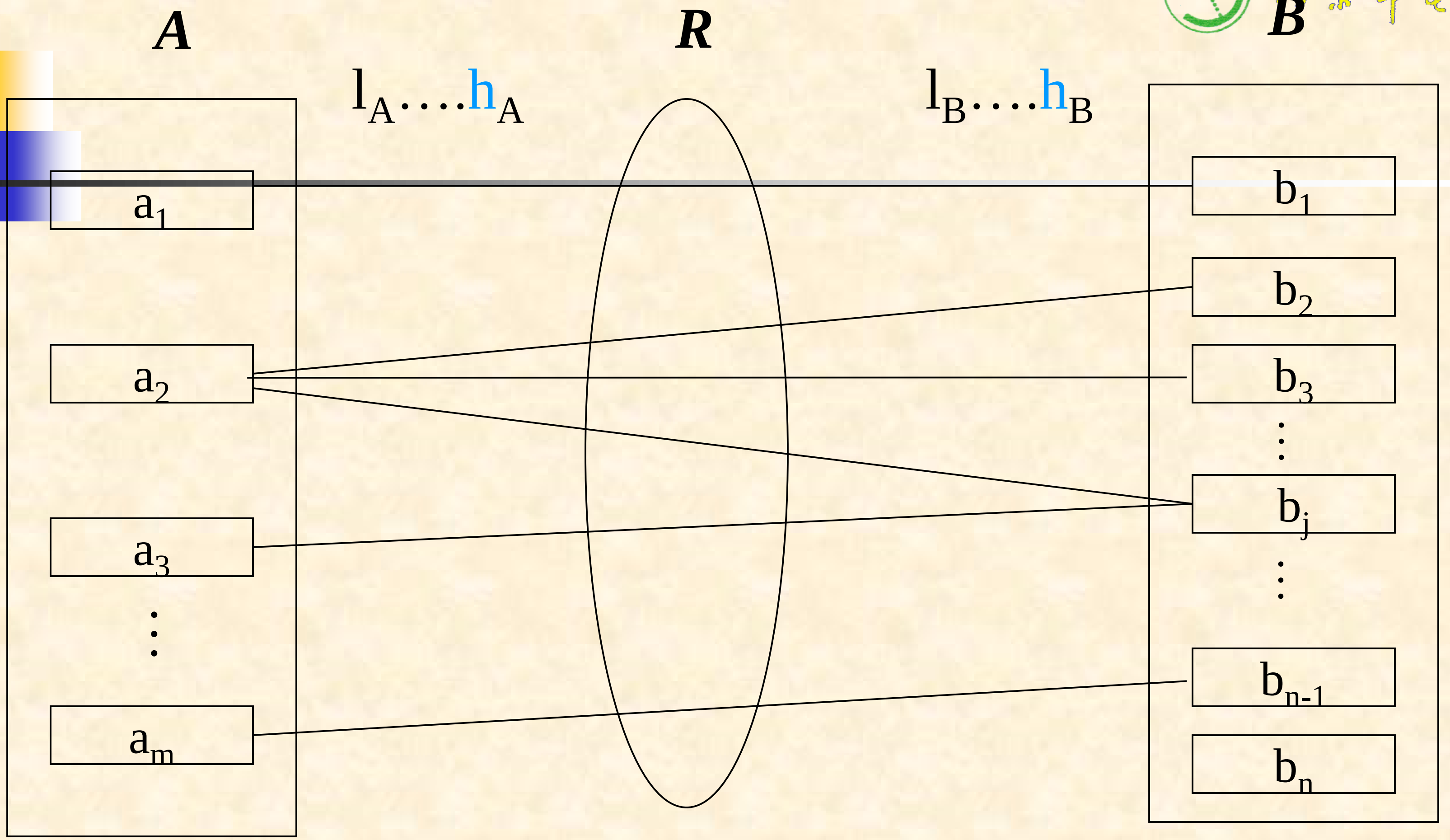


Fig.6.13

Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Cardinality Limits for Participation

- /* 设联系***R***关联了entity sets ***A***和***B***, 为*定量地*描述***A***参与***R***的total/partial participation 和***A***中的entity与***B***中的entity的mapping cardinality, 引入实体参与联系的cardinality limits
- *A*参与*R*的基数下界l_A和上界h_A , refer to Fig.7.0.5
  - *A*中的每个实体*a*通过*R*关联了最少l_A个、最多h_A个*B*中实体*b*
    - l_A : 对*A*中的每个实体*a*, *B*中至少有l_A个实体*b*通过*R*与其对应/关联
    - h_A:对*A*中的每个实体*a*, *B*中至多有h_A个实体*b*通过*R*与其对应/关联



$l_A=1, h_A=3$ $l_B=0, h_B=2$

Fig.7.0.5 Illustration for cardinality limits

Cardinality Limits for Participation (cont.)

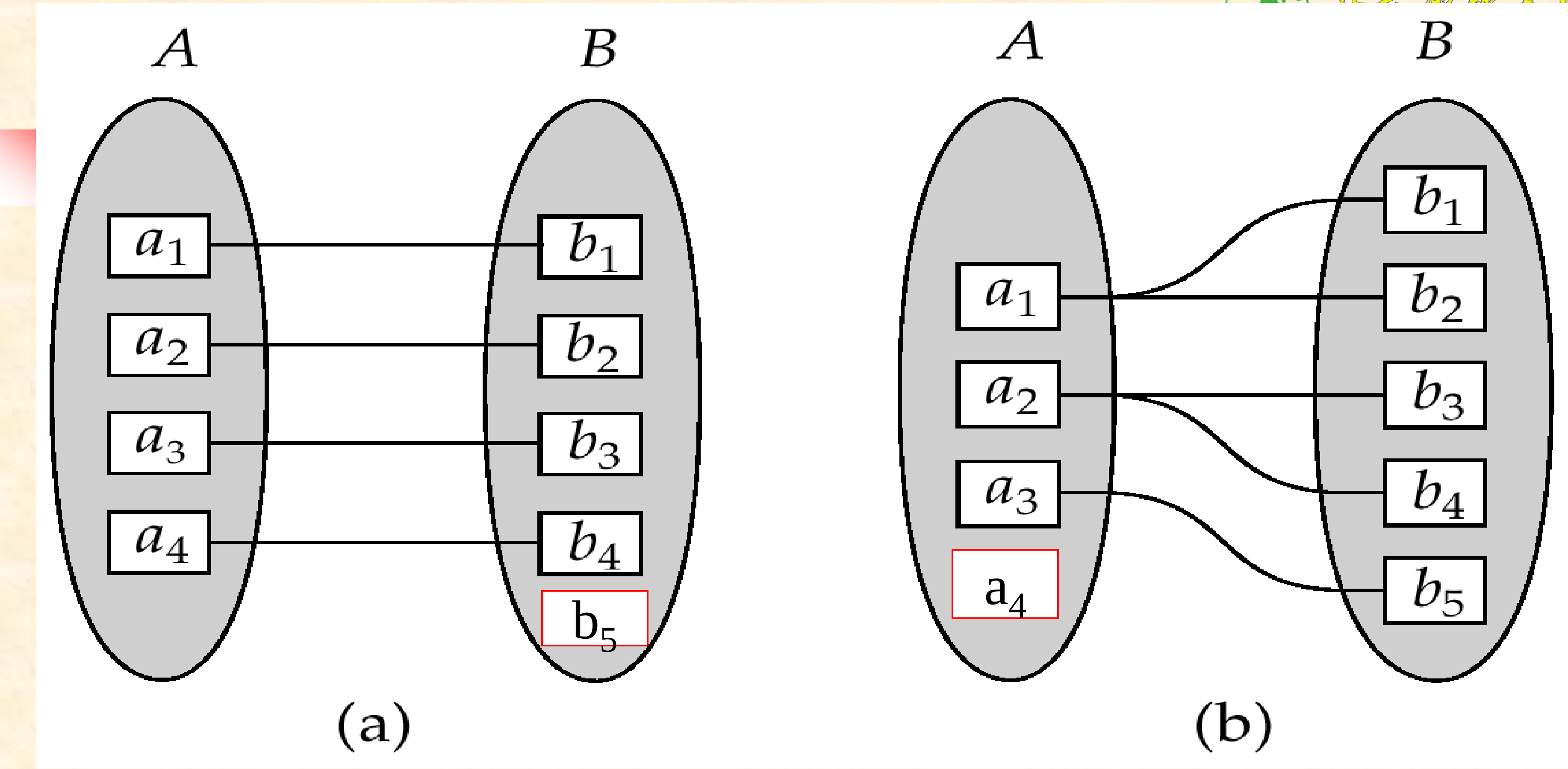
- Note
  - A maximum value of * indicates no limit
  - ***in some textbooks*, < l_A ,h_A > is put at the side of the entity B**
- Cardinality limits vs total/partial participation
  - A minimum value l_A =0: ***A*** is partial participation of ***R***
  - A minimum value l_A > 0: ***A*** is total participation of ***R*,** equivalent to *double line*

Cardinality Limits for Participation (cont.)

- Cardinality limits vs mapping cardinality
  - 设联系*R*关联了实体集*A*和*B*, 利用*A*的**基数界限**$< l_A , h_A >$中的h_A 、*B*的**基数界限**$< l_B , h_B >$中的h_B , 可推导出联系*R*的映射基约束
  - $< h_B , h_A >$ 表示了联系*R*的从*A*到*B*的映射基约束 !!!!
  - e.g. in Fig.7.10 ▷ , considering mapping cardinality form *instructor* to *student*
    - mapping cardinality form *instructor* to *student an* depends on $<h_{\text{student}}, h_{\text{instructor}}>$
    - for student, h_{student}=1, for instructor, $h_{\mathit{instructor}}$ =*
    - so, $<h_{\text{student}}, h_{\text{instructor}}>$ = <1, *>, and is *one to many*

Cardinality Limits for Participation (cont.)

- The associations between cardinality limits and the mapping cardinality are classified as follows, and illustrated by Fig.7.0.6



From *A* to *B*, one to one (1:1),

$< l_A , h_A > = <1, 1>$,

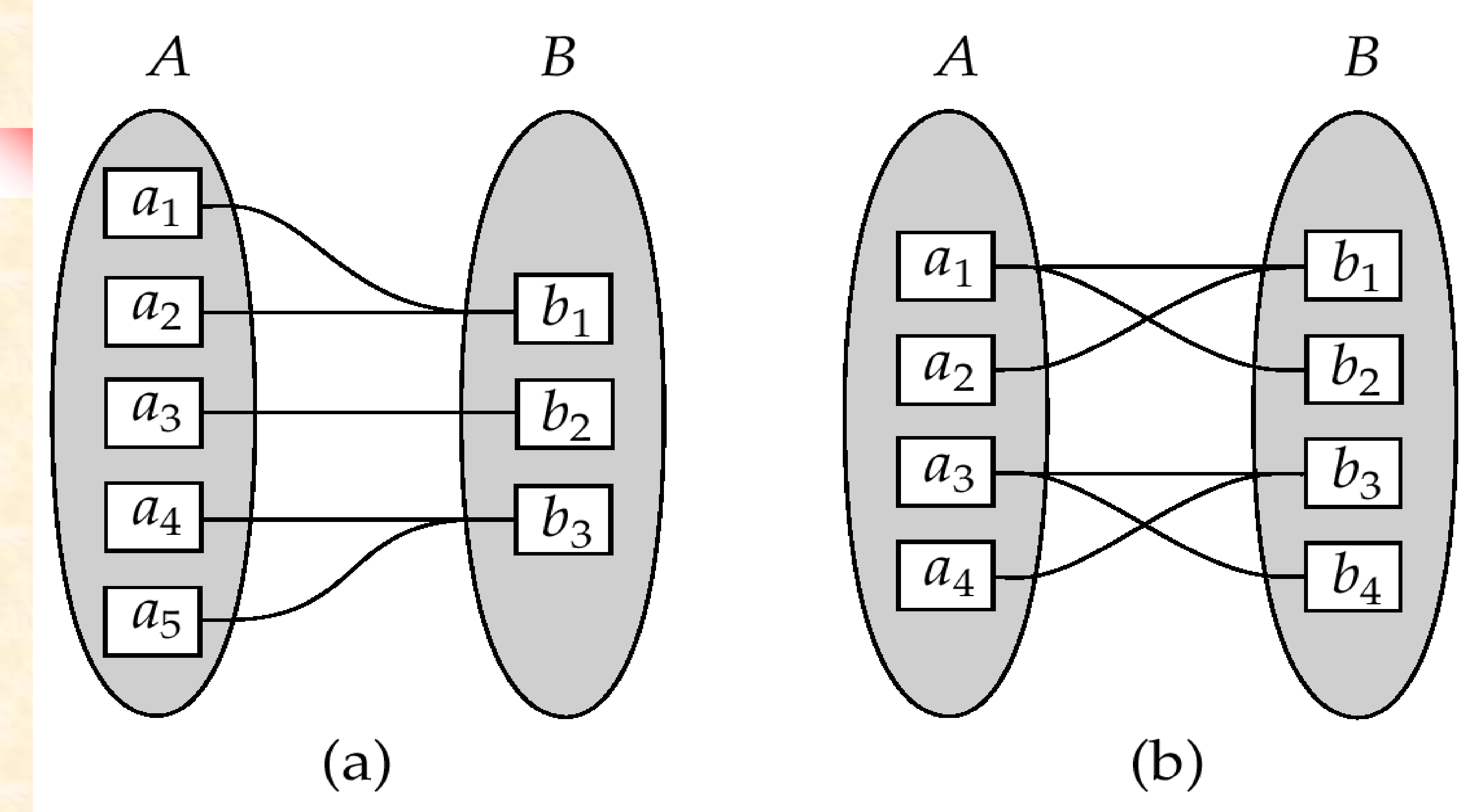
$< l_B , h_B > = <0, 1>$;

From *A* to *B*, one to many (1:2),

$< l_A , h_A > = <0, 2>$,

$< l_B , h_B > = <1, 1>$

Fig.7.0.6-1 Associations between cardinality limits and mapping cardinality



Form *A* to *B*, many to one (2:1),

$< l_A , h_A > = <1, 1>$,

$< l_B , h_B > = <1, 2>$

Form *A* to *B*, many to many (2:2)

$< l_A , h_A > = <1, 2>$

$< l_B , h_B > = <1, 2>$

6.5 Keys

- **Key** is a set *of attributes* (of a *entity set* or *relationship set*)

  , in which there are *one or more* attributes
  - the values of these attributes in one entity can be used to uniquely distinguish this entity from others , or
  - the values of these attributes in one relationship are used to uniquely identify the relationship
- Keys include
  - superkey (超键), candidate key (候选键), primary key (主键)

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 in the entity set
  - e.g. {*instructor_id* , *instructor_name*}
  - the super key may contain extraneous attributes
    - e.g *instructor_name*

- A candidate key is the minimal super key
  - *non-redundant* super key
    - e.g. *instructor-id* is the candidate key of *instructor*

Keys For Entity Sets (cont.)

- The *primary key* is a candidate key chosen by the database designer as the principal means of identifying entities within an entity set
  - although several candidate keys may exist, one of the candidate keys is selected to be the primary key
  - need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key

构造关系表时，如果有多个候选键，最好选取数值型（int, float）候选键作为关系表主键，便于提高基于主键的查询速度

—不要选字符串型属性，如varchar、datetime

—e.g. studentname, instructorName

Keys for Relationship Sets

- Keys for relationship sets ***R*** on entities E_1, E_2, …, E_n
  - $R = \{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, \ldots, e_n \in E_n\}$
    $\subseteq E_1 \times E_2 \times \ldots \times E_n$

    ， how to uniquely distinguish each relationship instances $\{(e_1, e_2, \ldots, e_n)$?
  - ***R*** is the combination of E_1, E_2, …, E_n, each E_i can be uniquely distinguished by primary_key(E_i), $1 \leq i \leq n$, so the set of all attributes in primary_key(E_1), primary_key(E_2), …, primary_key(E_n) can be used to recognize $(e_1, e_2, \ldots, e_n)$

Keys for Relationship Sets (cont.)

- The **super**_key for *R*
  - primary_key(E_1) ∪ primary_key(E_2) ….∪ primary_key(E_n)
  - e.g. in Fig.6.3, (*InstructorID*, *StudentIDr*) is the super key of *advisor* ▷
  - *note*
    - if the attribute names of primary-keys are not unique, the attributes with the same names should be renamed
- The candidate keys for *R*
  - minimal, non-redundant super keys

Keys for Relationship Sets (cont.)

- The ***candidate*** keys or the ***primary*** key for a binary relationship set ***R*** among entity sets ***A*** and ***B*** can be decided as follows, in accordance with the mapping cardinality of ***R***
  - ***R*** is *many-to-many*, ▷

    primary_key(R) = primary_key(A) ∪ primary_key(B)
  - ***R*** is *many-to-one from A to B*, !!

    primary_key(R) = primary_key(A)
  - ***R*** is *one-to-many*, !! ▷

    primary_key(R) = primary_key(B)

Keys for Relationship Sets (cont.)

- ***R*** is *one-to-one*, ▷

    primary_key(*R*) = primary_key(*A*)

    or: primary_key(*R*) = primary_key(*B*)