Chapter 6

Database Design

Ch6 Database Design

Logical Database Design

- also known as
  - Database Design
  - Database Modeling

Ch6 Database Design

- **Database design** is the process of producing a detailed **data model of a database**.
- This **logical data model** contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a **Data Definition Language**, which can then be used to create a database.
- A fully attributed data model contains detailed attributes for each entity.

- **how do we?**

  1) **analyze an enterprise**
  2) **list the data items for a database**
  3) **decide how to place these data items columns in relational tables**

- **For example:** **student-course database**
  - **<u>attributes of student:</u>** **sno, sname, dept, sage**
  - **<u>attributes of course:</u>** **cno, cname**
  - **<u>attribute of student&course:</u>** **grade**

- **we can put them all in the same table.**

  R(sno, sname, dept, sage, cno, cname, grade)

- **consider the SCG database (next slide), see any problems with that ?**

The SCG Database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

Problems

1) redundency（数据冗余）

2) abnormity of update（修改异常）

3) abnormity of delete（删除异常）

4) abnormity of insert（插入异常）

Ch6 Database Design

1) redundency（数据冗余）

- waste of disk space

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

2) abnormity of update（修改异常）

- waste of time
- user might get it wrong

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

3) abnormity of delete（删除异常）

- might lose some informations

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

3) abnormity of delete（删除异常）

- might lose some informations

需要删除学生:

“S0003, Zhang Yimou, CS, 17”

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	[illegible]	[illegible]	[illegible]	[illegible]	5
S0001	Wang [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S0001	Wang [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3
S0001	Wang Jian	CS	17	[illegible]	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

3) abnormity of delete（删除异常）

- might lose some informations

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS				5
S0001						
S0001						
S0001						
S0001						4
S0002	Chen Ying		19		BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

因此需要删除该学生所在的元组，结果会导致C107这门课程的信息也一起被删除。

3) abnormity of delete（删除异常）

- might lose some informations

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3

Relation R （删除后的结果关系）

4) abnormity of insert（插入异常）

- unsuccessful insert

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

Relation R

Sno	Sname	Dept	Sage
S0001	Wang Jian	CS	17
S0002	Chen Ying	MA	19
S0003	Zhang Yimou	CS	17

Relation S

Sno	Cno	Grade
S0001	C101	5
S0001	C102	5
S0001	C103	4
S0001	C105	3
S0001	C110	4
S0002	C103	3
S0002	C105	3
S0003	C107	4

Relation SC

Cno	Cname
C101	ABC
C102	ACD
C103	BBC
C105	AEF
C107	BHD
C110	BCF

Relation C

The SCG Database (another approach)

Contents

6.1 Introduction to E-R Concepts

- An **Entity-Relationship(ER) model** is an abstract way to describe a database.

- A design approach, called ***Entity-Relationship modelling***, is more intuitive, less mechanical, but basically leads to the same end design.

6.1 Introduction to E-R Concepts

- **Entity-Relationship Model**
  - **Proposed by Peter Chen (1976):**

    The Entity-Relationship Model: Toward a Unified View of Data

- **Peter Chen (Pin-shan Chen)**

6.1 Introduction to E-R Concepts

- **E-R Model**
  - **three fundamental data classification objects**
    - **entity**
    - **attribute**
    - **relationship**
- **the contents of this section**
  - **Entities, Attributes, and Simple E-R Diagrams**
  - **Transforming Entities and Attributes to Relations**
  - **Relationships among Entities**

6.1 Introduction to E-R Concepts

❑ Entities, Attributes, and Simple E-R Diagrams

- **Def. 6.1.1 Entity （实体）**
  - **An entity is a collection of distinguishable real-world objects with common properties.**

– **E.g., college registration database:**
  - **Students**
  - **Instructors**
  - **Class_rooms**
  - **Courses**
  - **Course_sections**
    - **different offerings of a single course, generally at different times by different instructors**

6.1 Introduction to E-R Concepts

- **Normally**
  - **an entity such is mapped to a relational table**
    - **represents a set of objects**
  - **each row is an entity occurrence, or entity instance**
    - **represents a particular object**

6.1 Introduction to E-R Concepts

❑Def. 6.1.2 Attribute （属性）

- ➢An attribute is a data item that describes a property of an entity or a relationship (defined below).

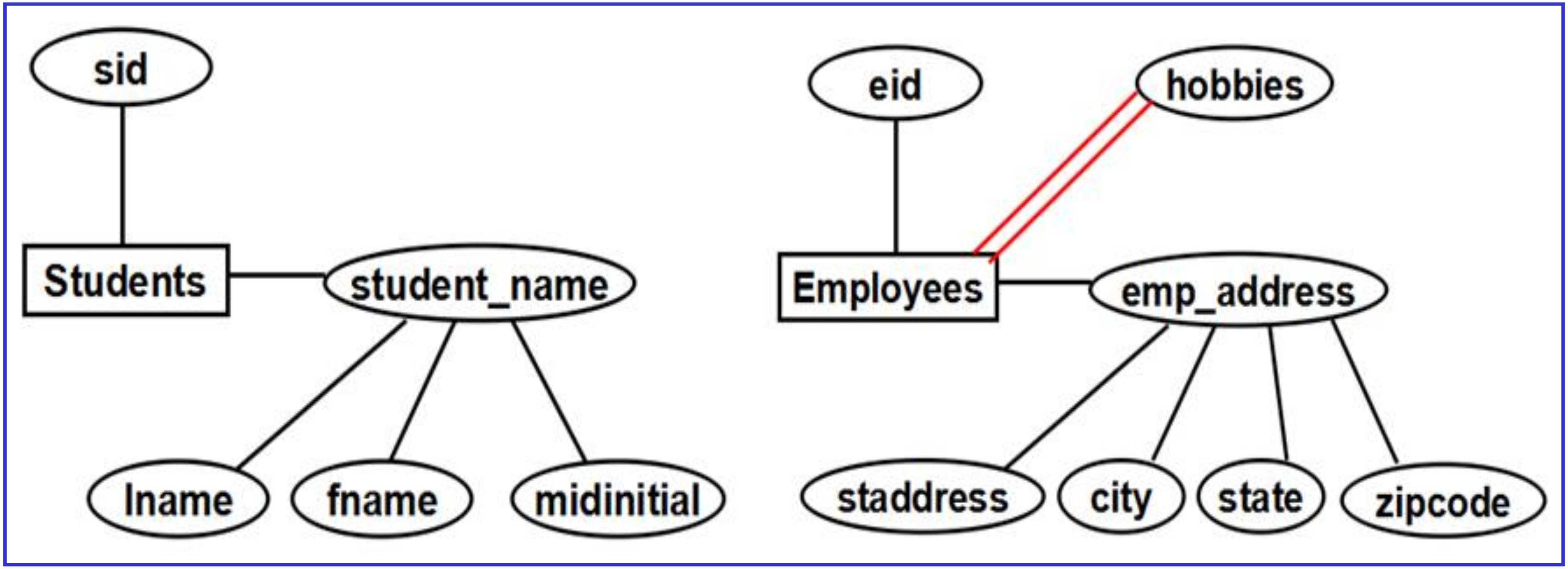


Figure 6.2 Example of E-R Diagrams with Entities and Attributes

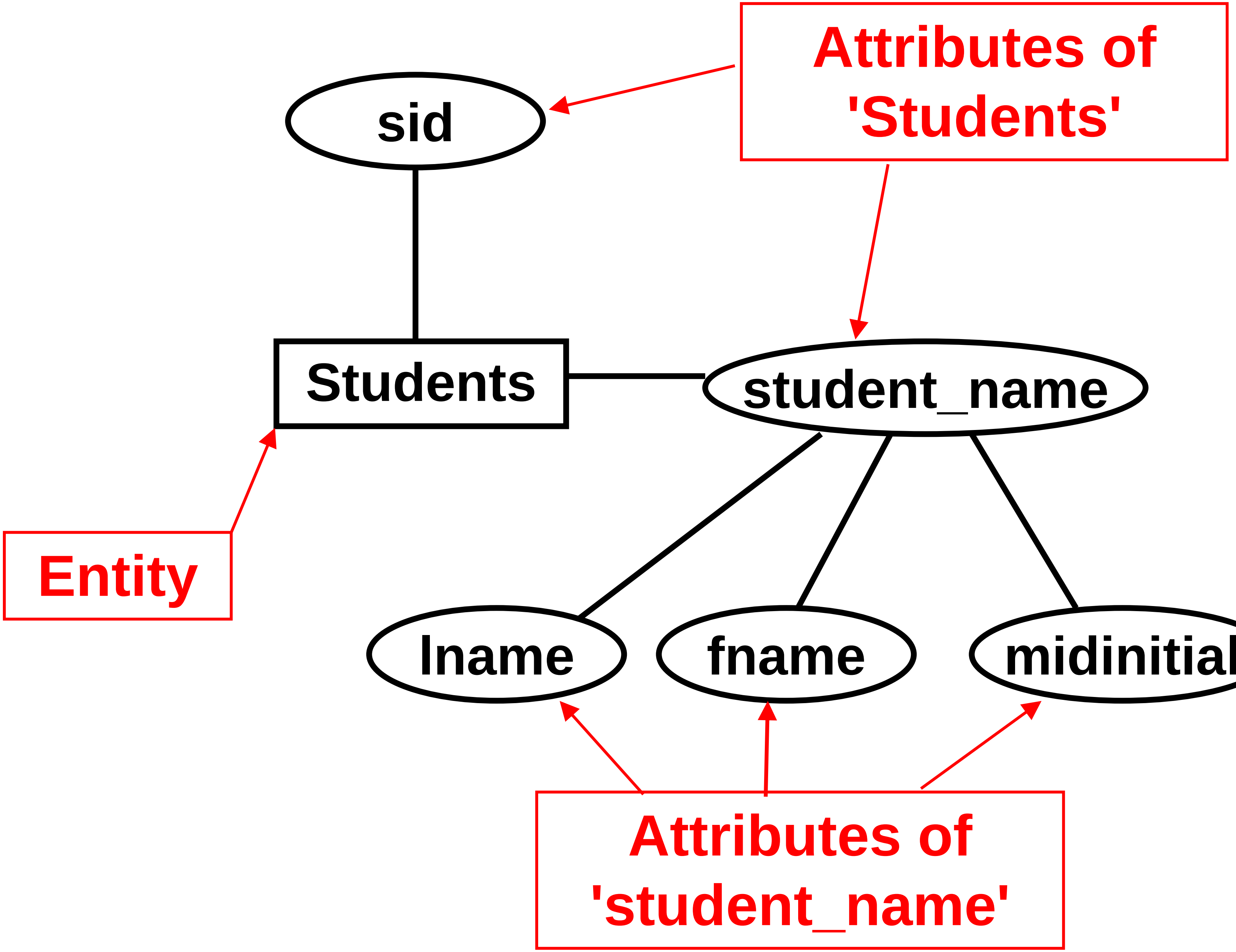


Figure 6.2 Example of E-R Diagrams with Entities and Attributes

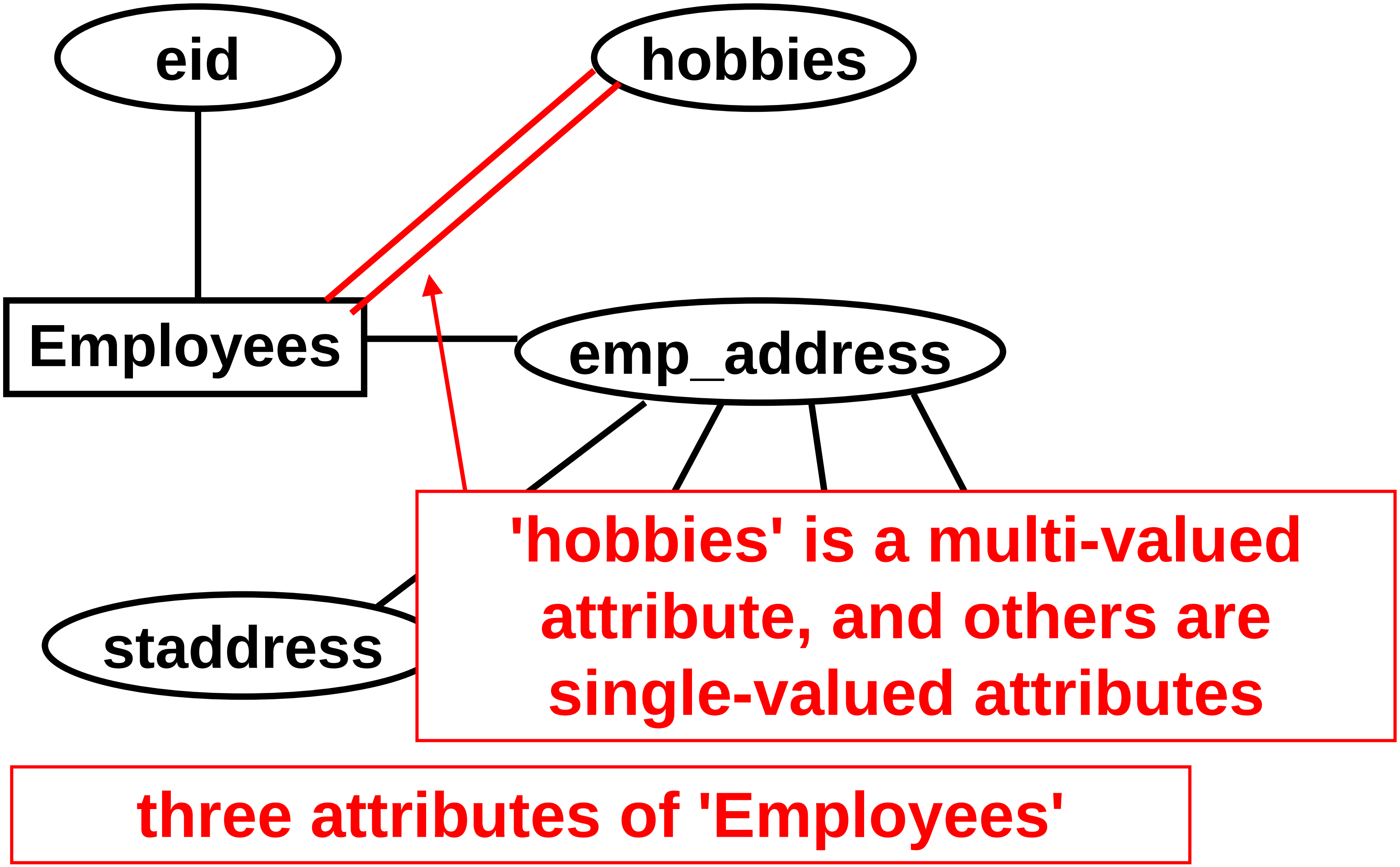
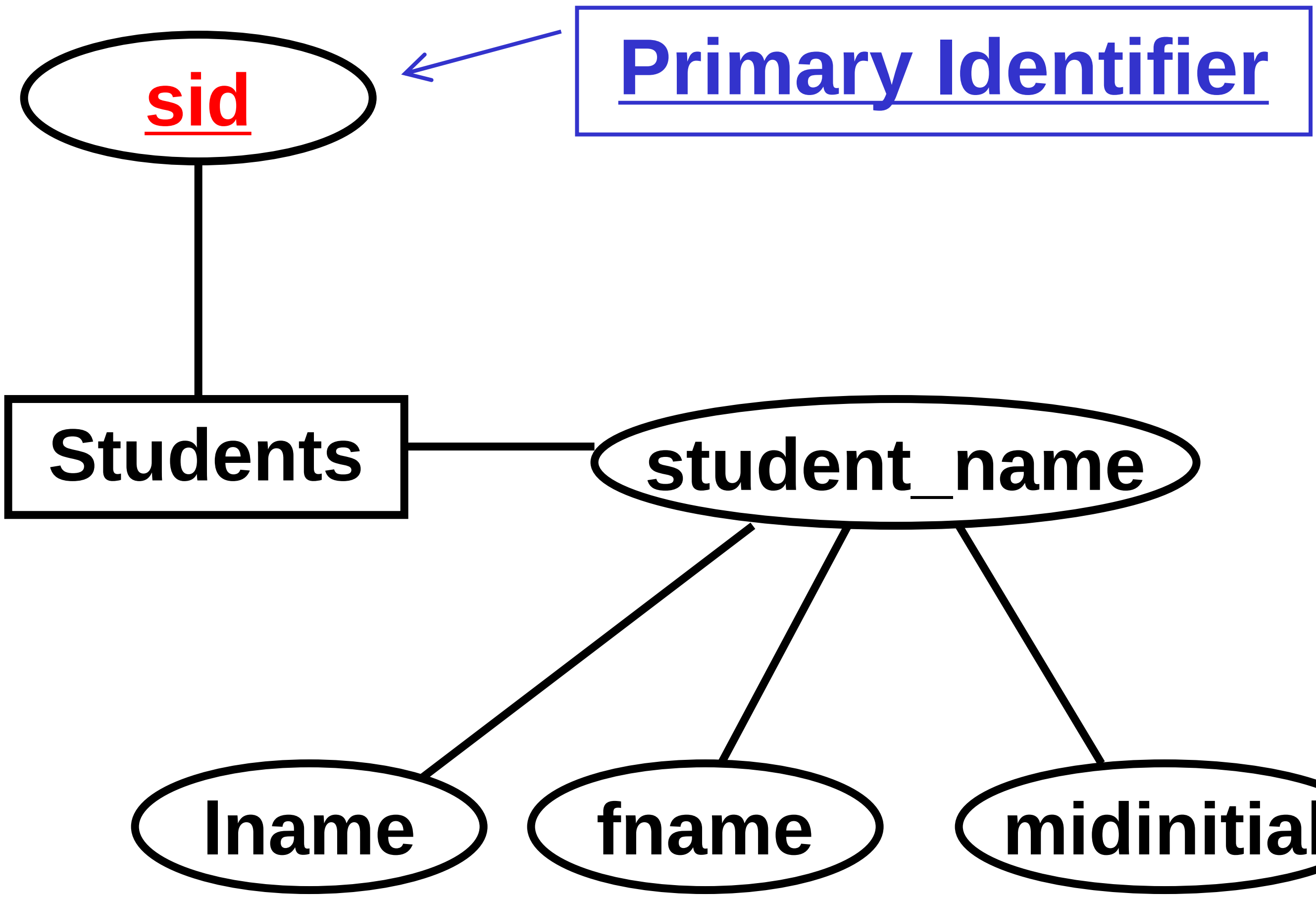


Figure 6.2 Example of E-R Diagrams with Entities and Attributes

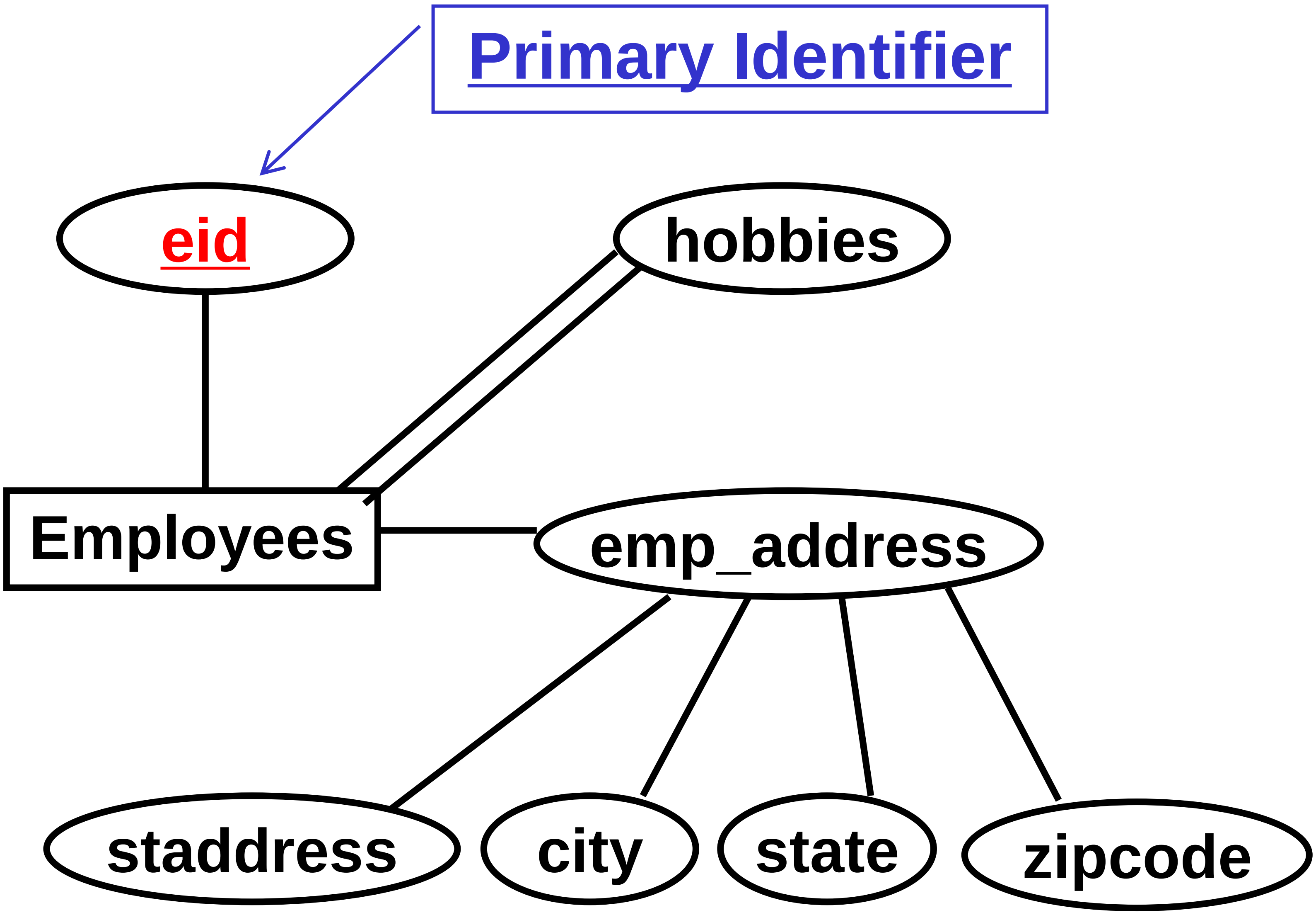
6.1 Introduction to E-R Concepts

- special terminology for special kinds of attributes (Figure 6.2, pg. 241)
  - identifier (candidate key，候选键)
    - An identifier is an attribute or set of attributes that uniquely identifies an entity instance.
  - primary identifier （主键）

6.1 Introduction to E-R Concepts



6.1 Introduction to E-R Concepts



6.1 Introduction to E-R Concepts

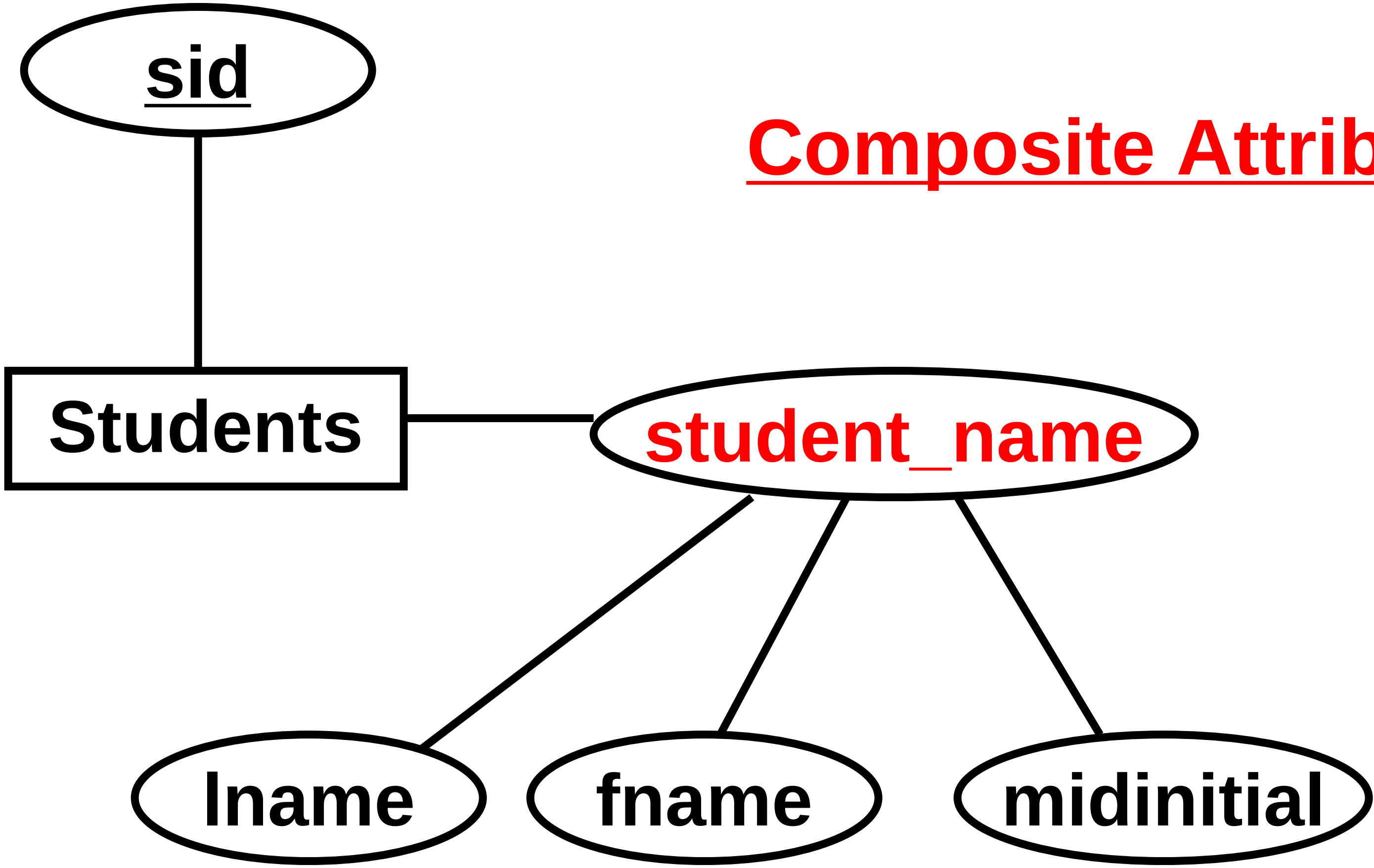
- **descriptor**
  - **A descriptor is a non-key attribute, descriptive.**
    - **Students.age**
    - **Employees.name**

6.1 Introduction to E-R Concepts

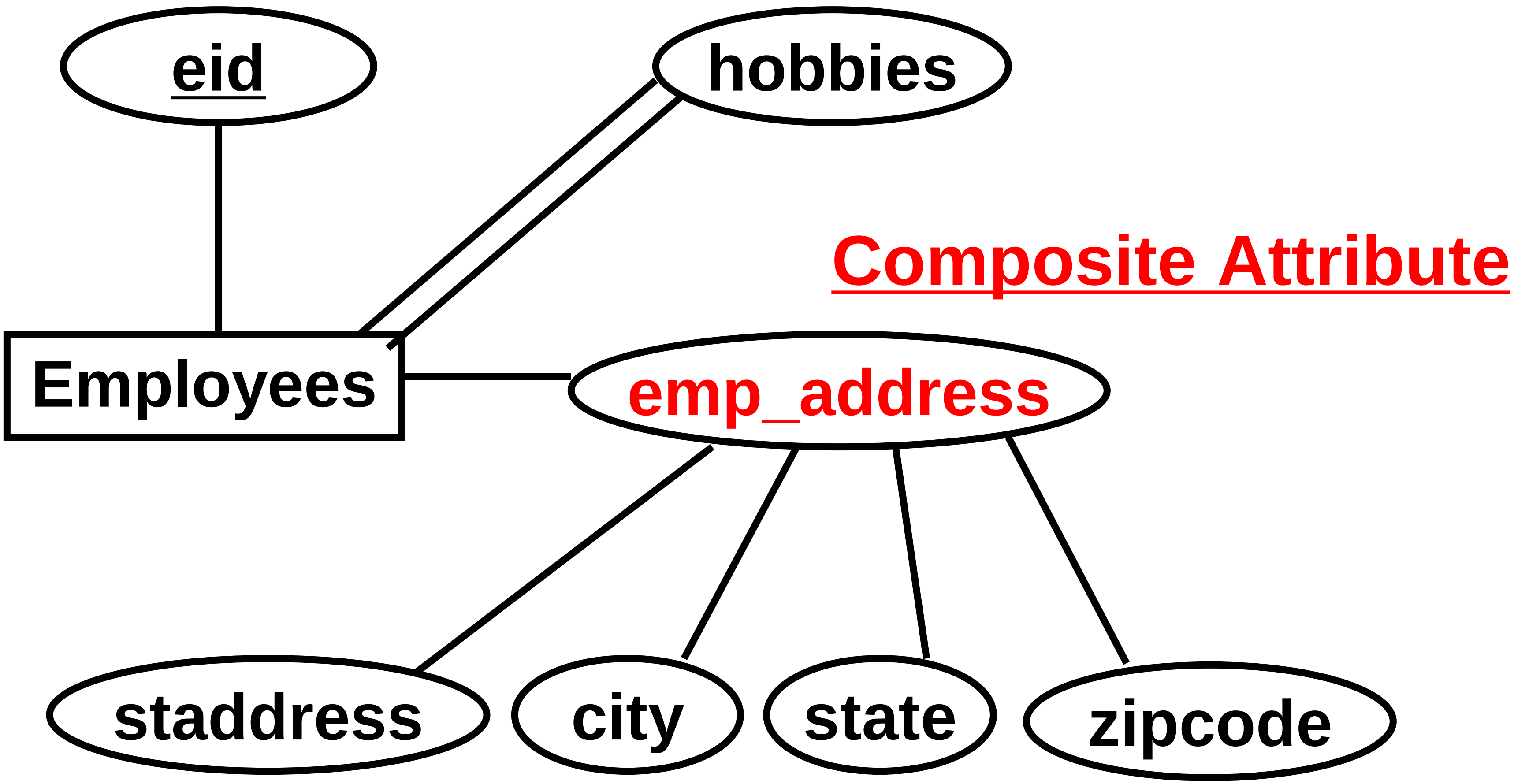
- **a composite attribute**
  - **a group of simple attributes that together describe a property.**
    - **Students.student_name**
    - **Employees.emp_address**

6.1 Introduction to E-R Concepts

Composite Attribute

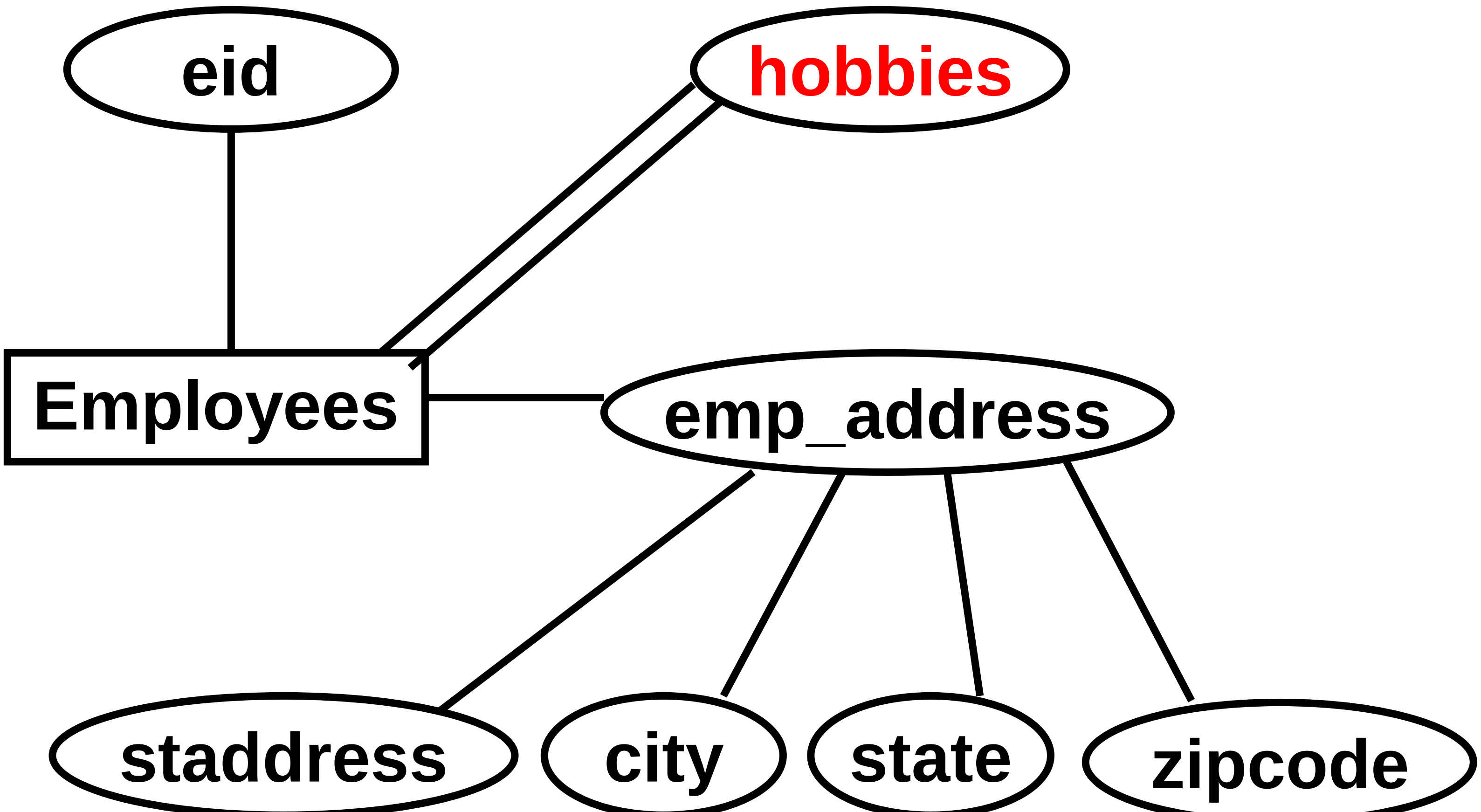


6.1 Introduction to E-R Concepts



6.1 Introduction to E-R Concepts

- **a multi-valued attribute**
  - **can take on multiple values for a single entity instance.**



E-R diagrams

- a entity
  - representing as a rectangle
- a single-valued attribute
  - representing as a oval
  - attached by a straight line to the entity.
- a composite attribute
  - is also in an oval attached directly to the entity
  - the simple attributes that make up the composite are attached to the composite oval.
- a multi-valued attribute
  - is attached by a double line to the entity it describes.

6.1 Introduction to E-R Concepts

- **Transforming Entities and Attributes to Relations**
  - **Transformation Rule 1**
    - **An entity is mapped to a single table. *The single-valued attributes of the Entity are mapped to columns* (composite attributes are mapped to multiple simple columns).**
    - **Entity occurrences become rows of the table.**
  - **Example 6.1.1, pg. 241**

Students(*<u>sid</u>*, lname, fname, midiaitia)

Employees(*<u>eid</u>*, staddress, city, state, zipcode)

6.1 Introduction to E-R Concepts

- **Transforming Entities and Attributes to Relations**
  - **Transformation Rule 2**
    - **A multi-valued attribute must be mapped to its own table.**
  - **Example 6.1.2, pg.242**

Employees(*<u>eid</u>*, staddress, city, state, zipcode)

hobbies(hobby, eid)

- **<u>No longer true in ORDBMS</u> !**

6.1 Introduction to E-R Concepts

❑ Relationships（联系）among Entities

- **Def. 6.1.3. Relationship (pg. 242).**
  - **Given an ordered list of m entities, E1, E2, . . . , Em, (where the same entity may occur more than once in the list)**
  - **a relationship R defines a rule of correspondence between the instances of these entities.**
  - **Specifically, R represents a set of m-tuples, a subset of the Cartesian product of entity instances.**

Figure 6.3: Examples of Relationships

Instructors teaches Course_sections

Figure 6.3: Examples of Relationships

Employees works_on Projects(percent of time)

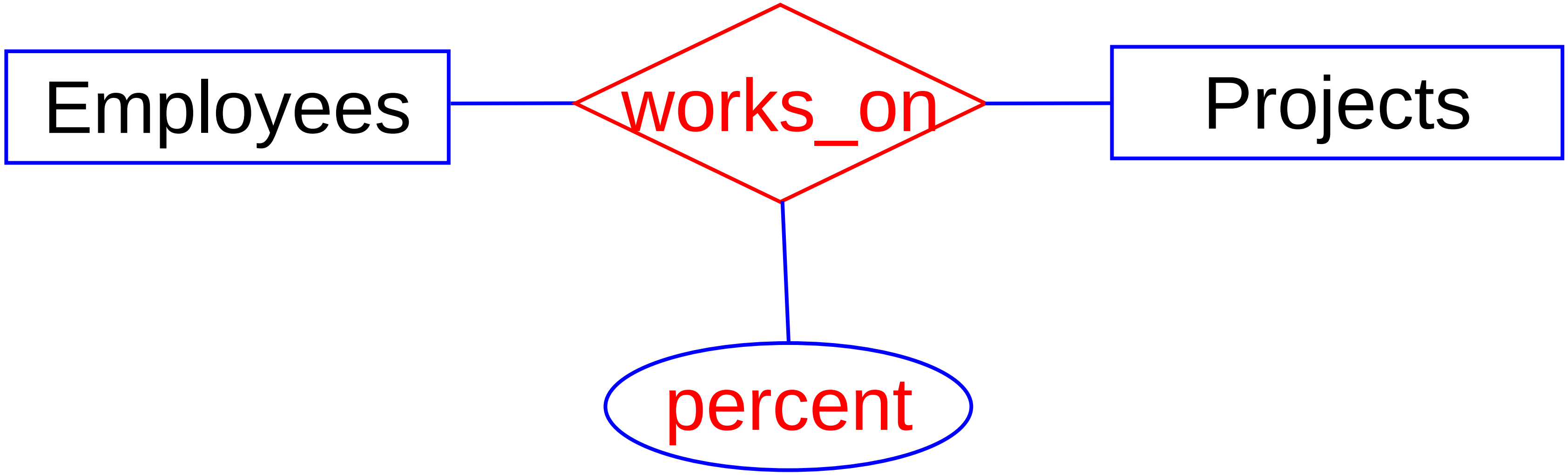
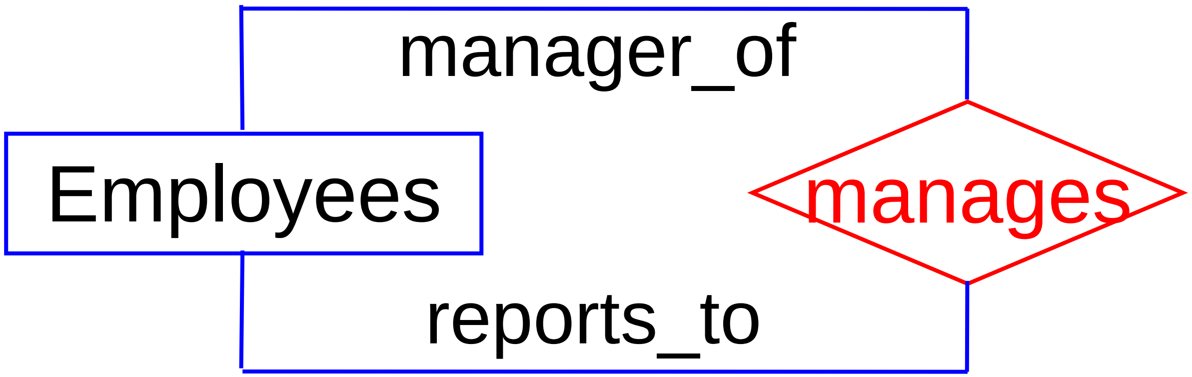


Figure 6.3: Examples of Relationships

Employees manages Employees



- **ring, or recursive relationship**

❑ Figure 6.3: Examples of Relationships

Instructors teaches Course_sections

Employees works_on Projects(percent of time)

Employees manages Employees

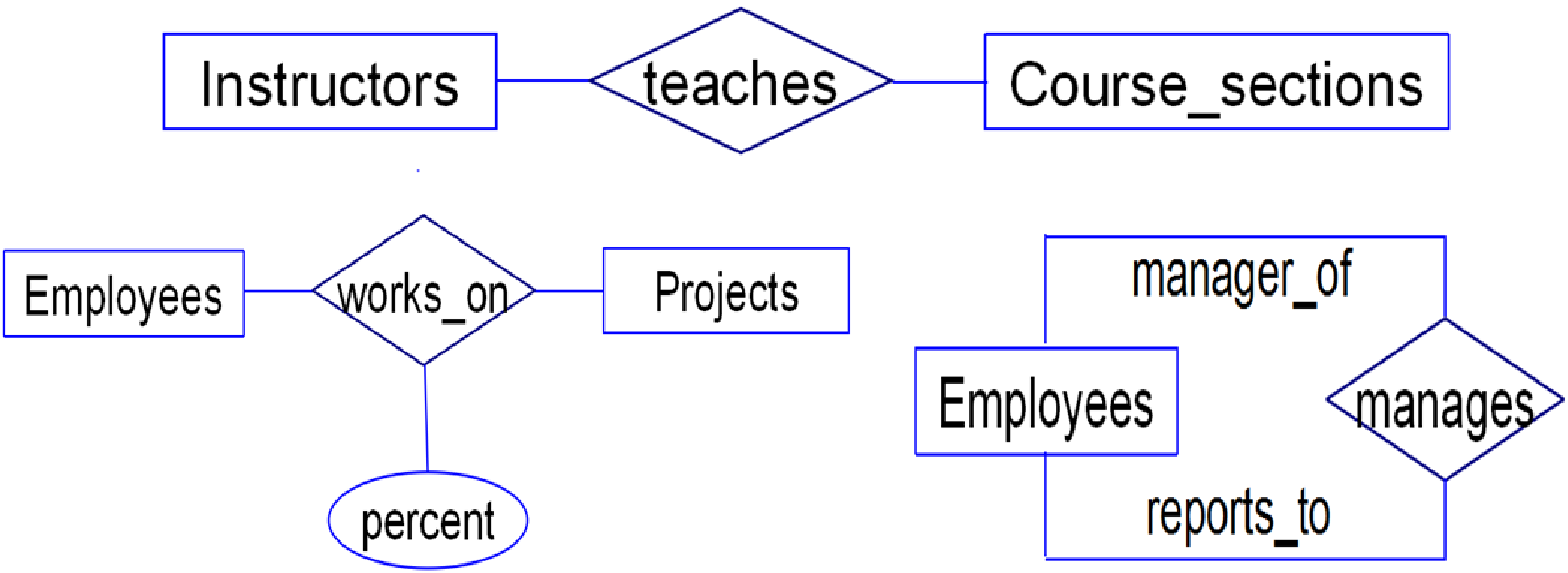


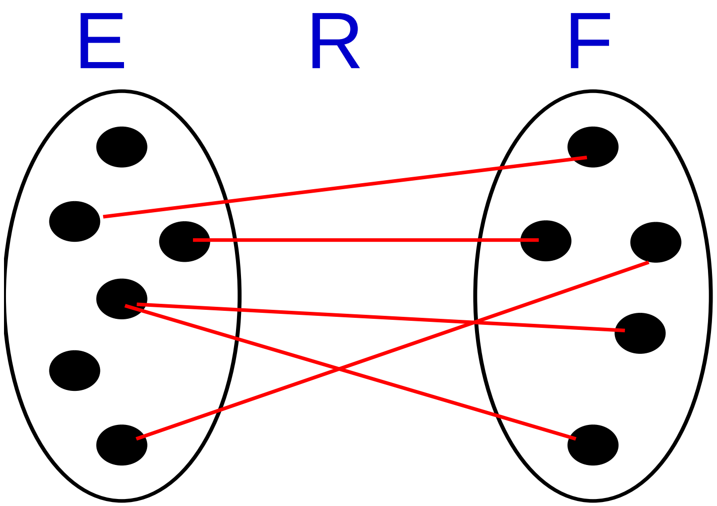
Figure 6.3:Examples of E-R Diagrams with Relationships

6.1 Introduction to E-R Concepts

- **Figure 6.3, pg. 243**
  - **Employees _works_on_ Projects, where _works_on_ is a relationship. _works_on_ has the connected attribute '_percent_'.**
  - **Note:**
    - **_percent_, associated with relationship, i.e., a value with each relationship instance.**
    - **The relationship instance represents a specific pairing of an Employees instance with a Projects instance;**
    - **percent represents the percent of time an employee instance works on that project.**

6.2 Further Details of E-R Diagrams

- **Cardinality of Entity Participation in a Relationship**
  - **Look at Figure 6.6.**
    - **Entities E and F, relationship R.**
    - **Lines between dots.**
      - **Dots are entity instances.**
      - **Lines are relationship instances.**

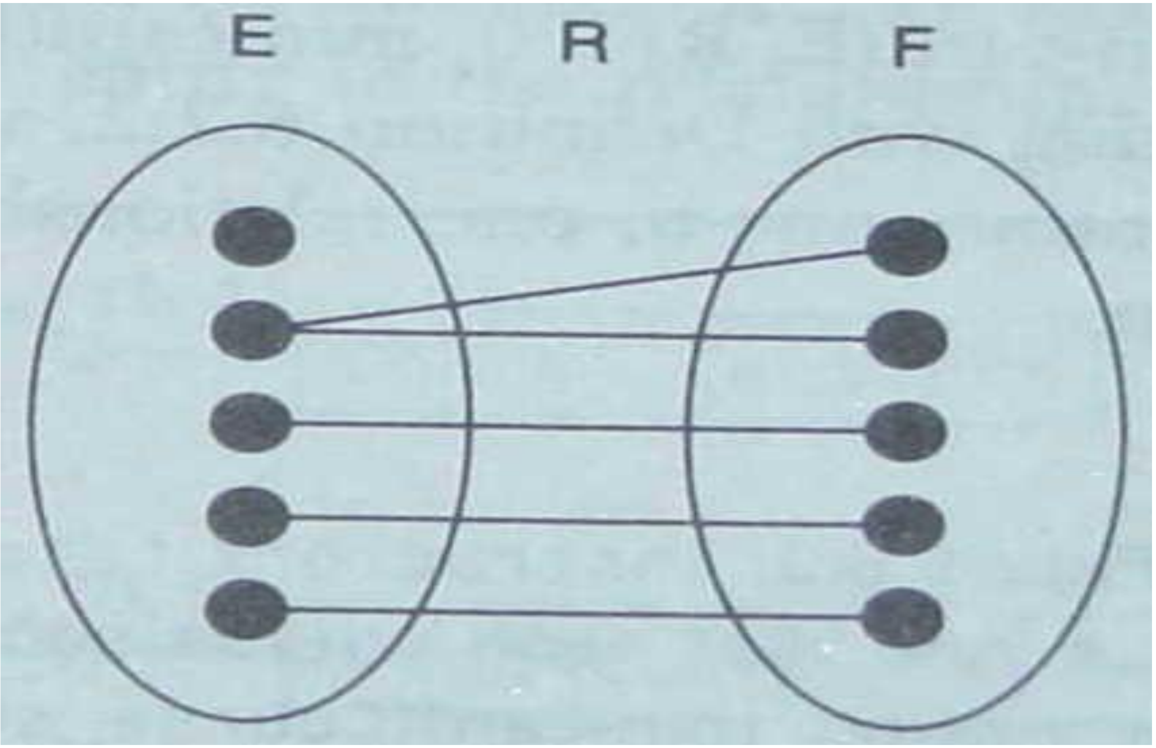


6.2 Further Details of E-R Diagrams

- If all dots in the entity E have AT MOST one line coming out, we say:
  - **max-card(E, R) = 1**
- If more than one line out is possible, we say:
  - **max-card(E, R) = N**

- If all dots in the entity E have AT LEAST one line coming out, we say:
  - **min-card(E, R) = 1**
- If some dots might not have a line coming out, we say:
  - **min-card(E, R) = 0**

6.2 Further Details of E-R Diagrams



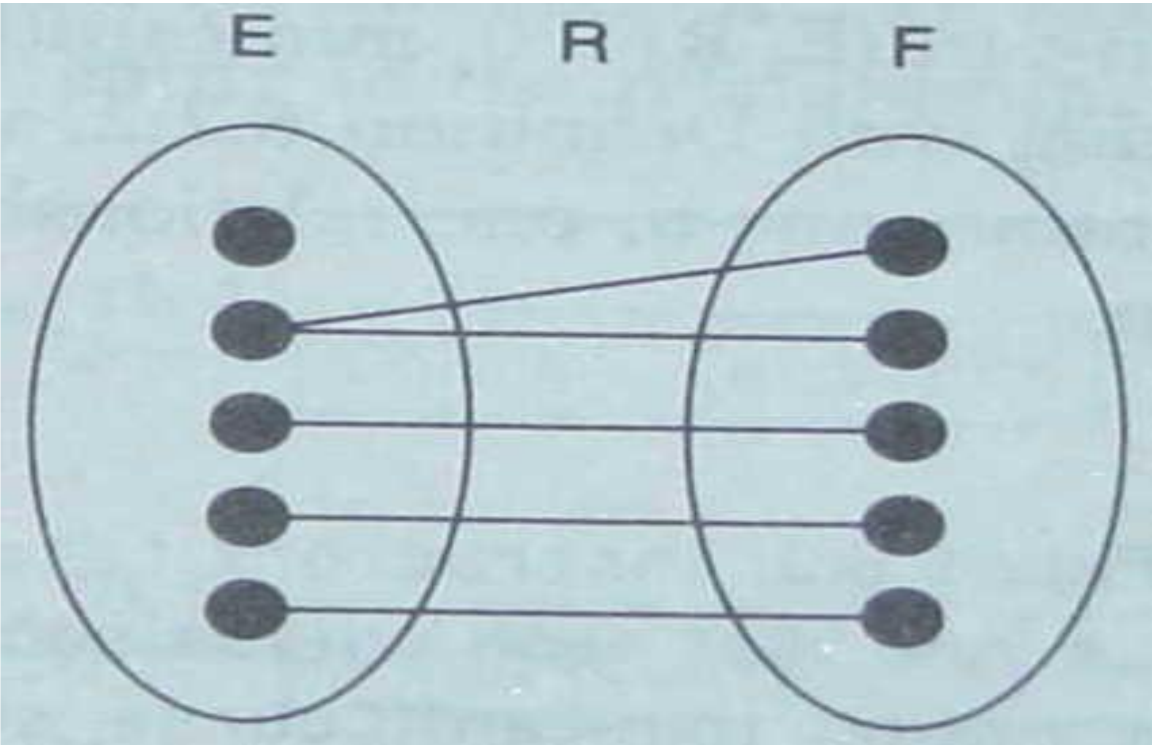
max-card(E, R) = ?

min-card(E, R) = ?

max-card(F, R) = ?

min-card(F, R) = ?

6.2 Further Details of E-R Diagrams



max-card(E, R) = N

min-card(E, R) = 0

max-card(F, R) = 1

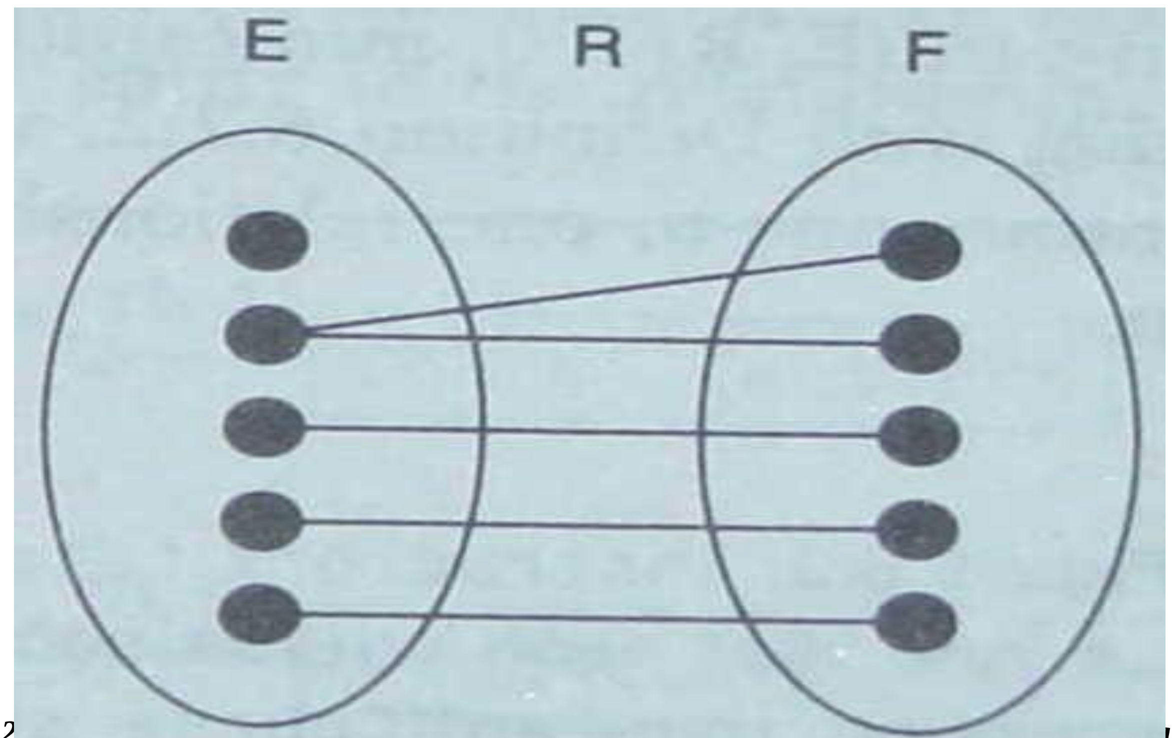
min-card(F, R) = 1

6.2 Further Details of E-R Diagrams

- **Def. 6.2.1 Card(E, R)**
  - **We combine these, by saying card(E, R) = (x, y) if:**

    min-card(E, R) = x and max-card(E, R) = y

    - **x is either 0 or 1, and y is either 1 or N**



card(E, R) = (0, N)

card(F, R) = (1, 1)

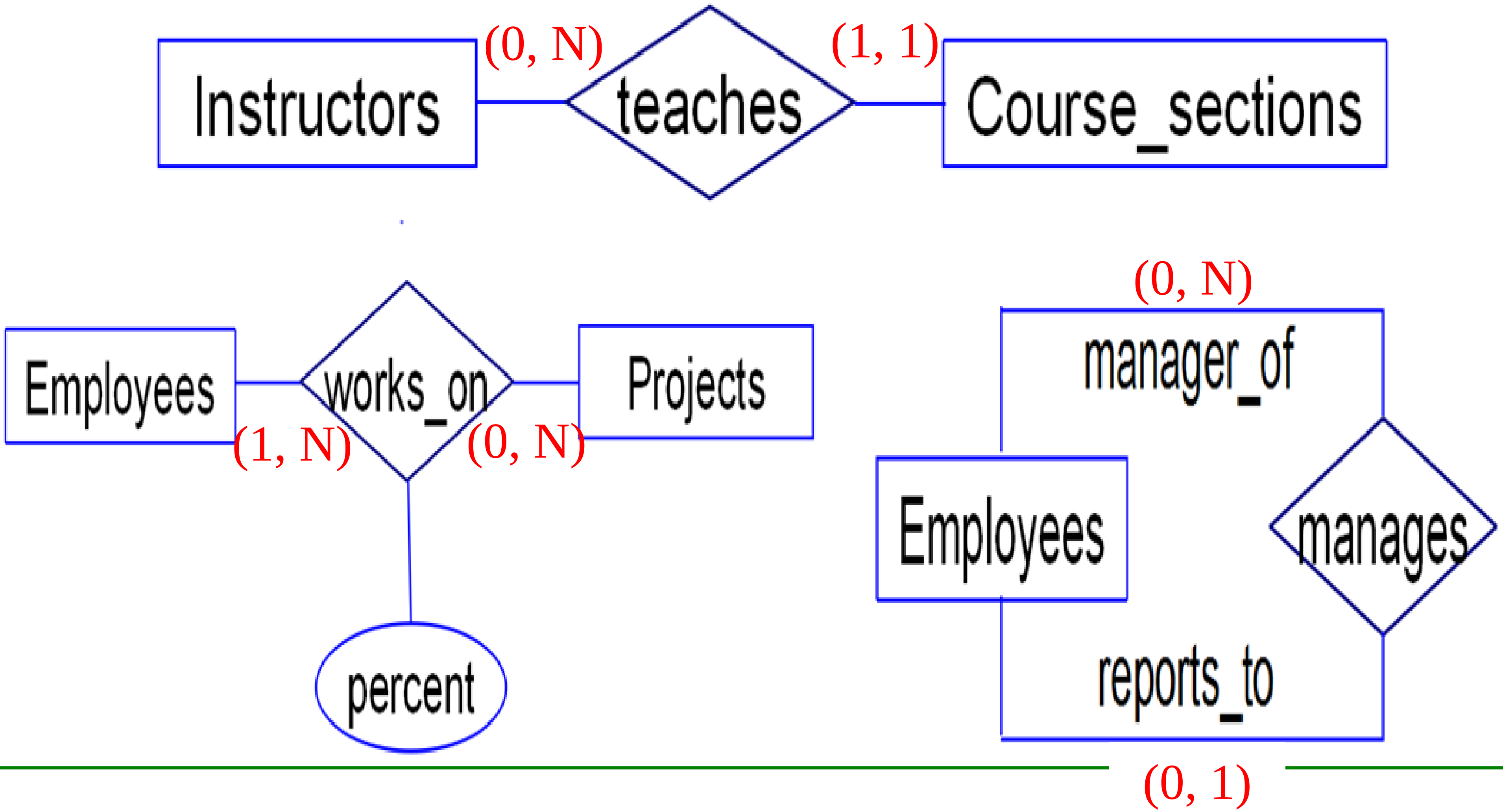


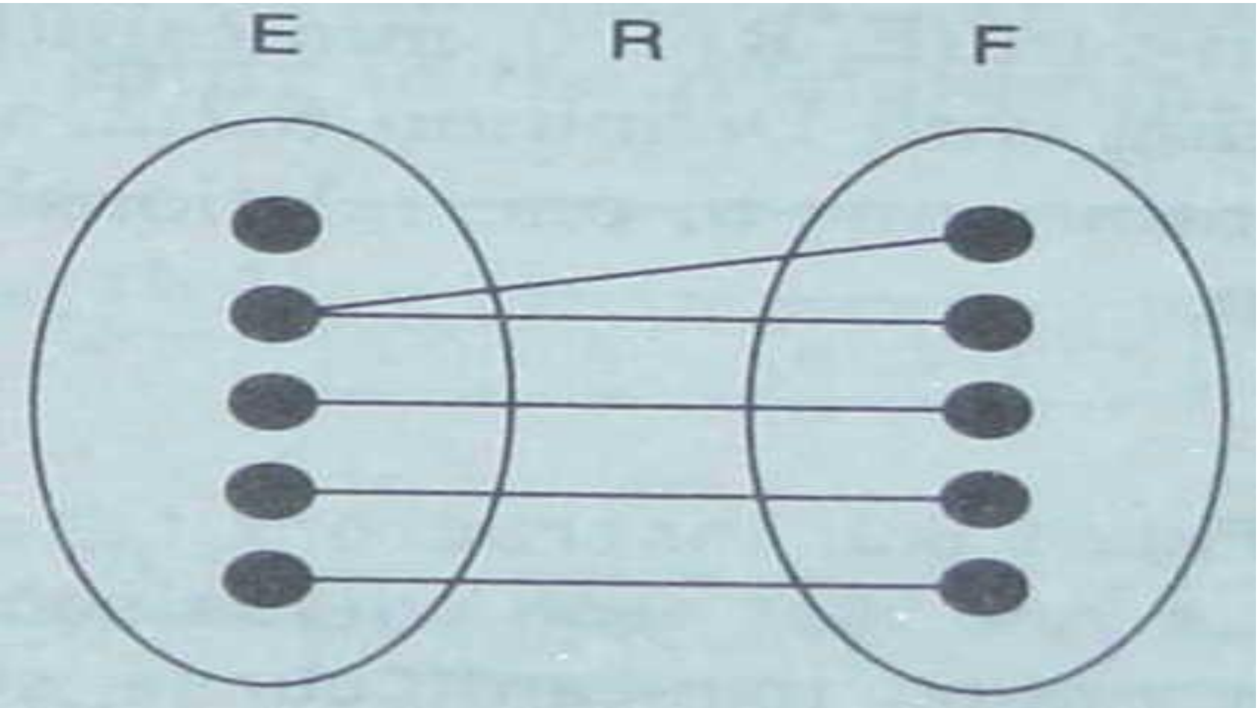
Figure 6.7: An E-R Diagrams with Labels(x,y) on Entity-Relationship Connections

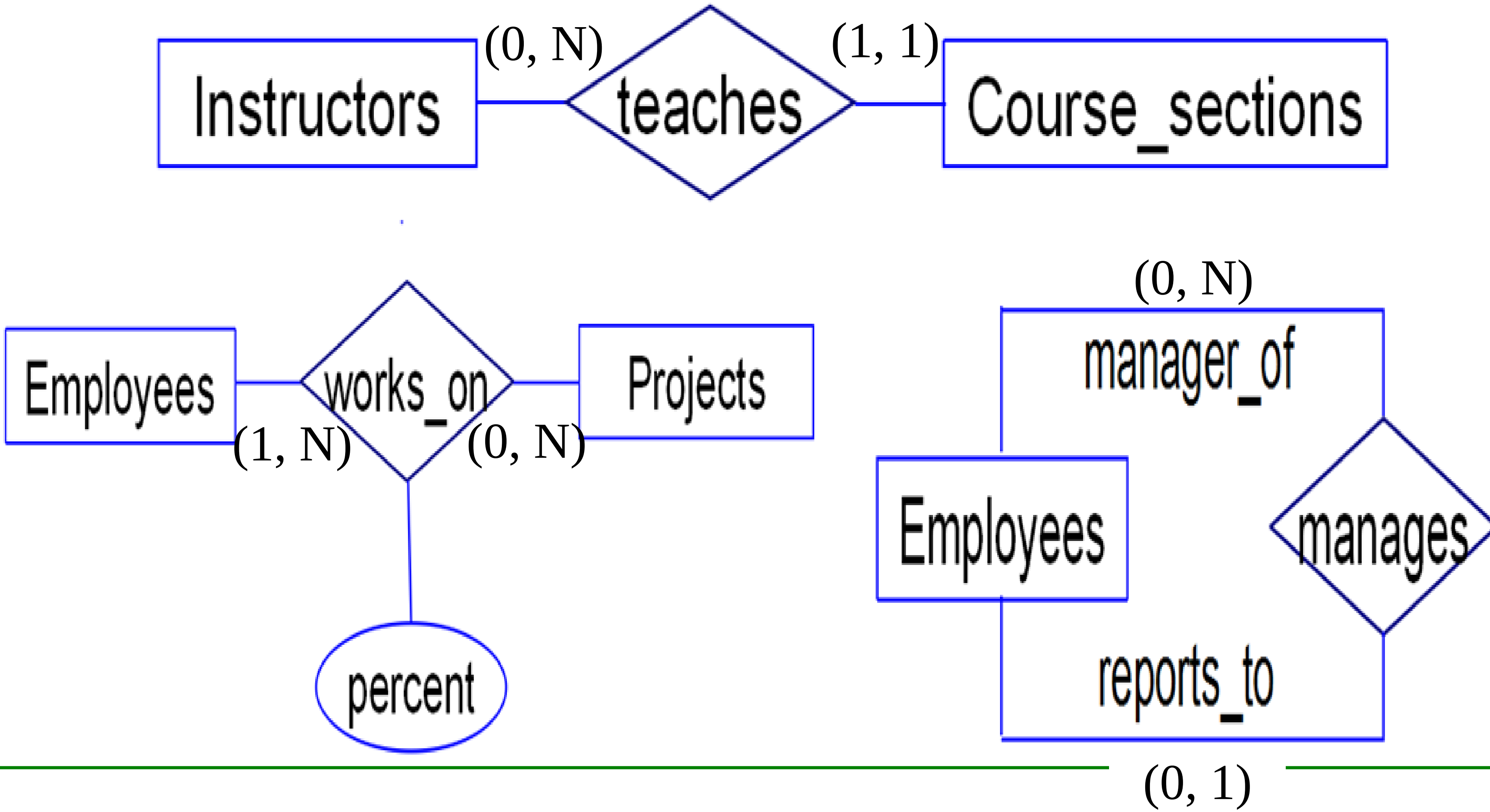
6.2 Further Details of E-R Diagrams

- **Employee1 in Emps_One is a manager of Employee2 in Emps_Two.**

6.2 Further Details of E-R Diagrams

- **Def 6.2.2**
  - if max-card(X, R) = 1 then X is said to have ***single-valued participation(单值参与)*** in the relationship R.
  - If max-card(X, R) = N, then X is said to be ***multi-valued participation（多值参与）*** in this relationship.



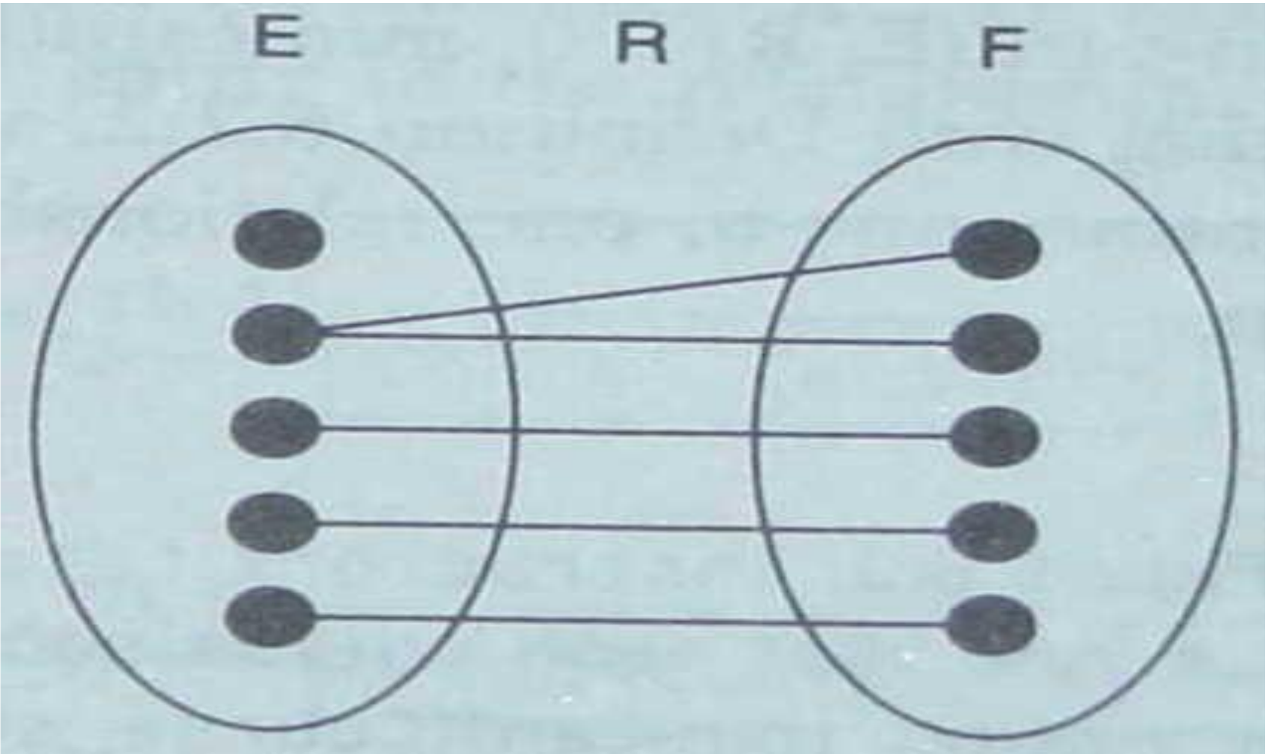


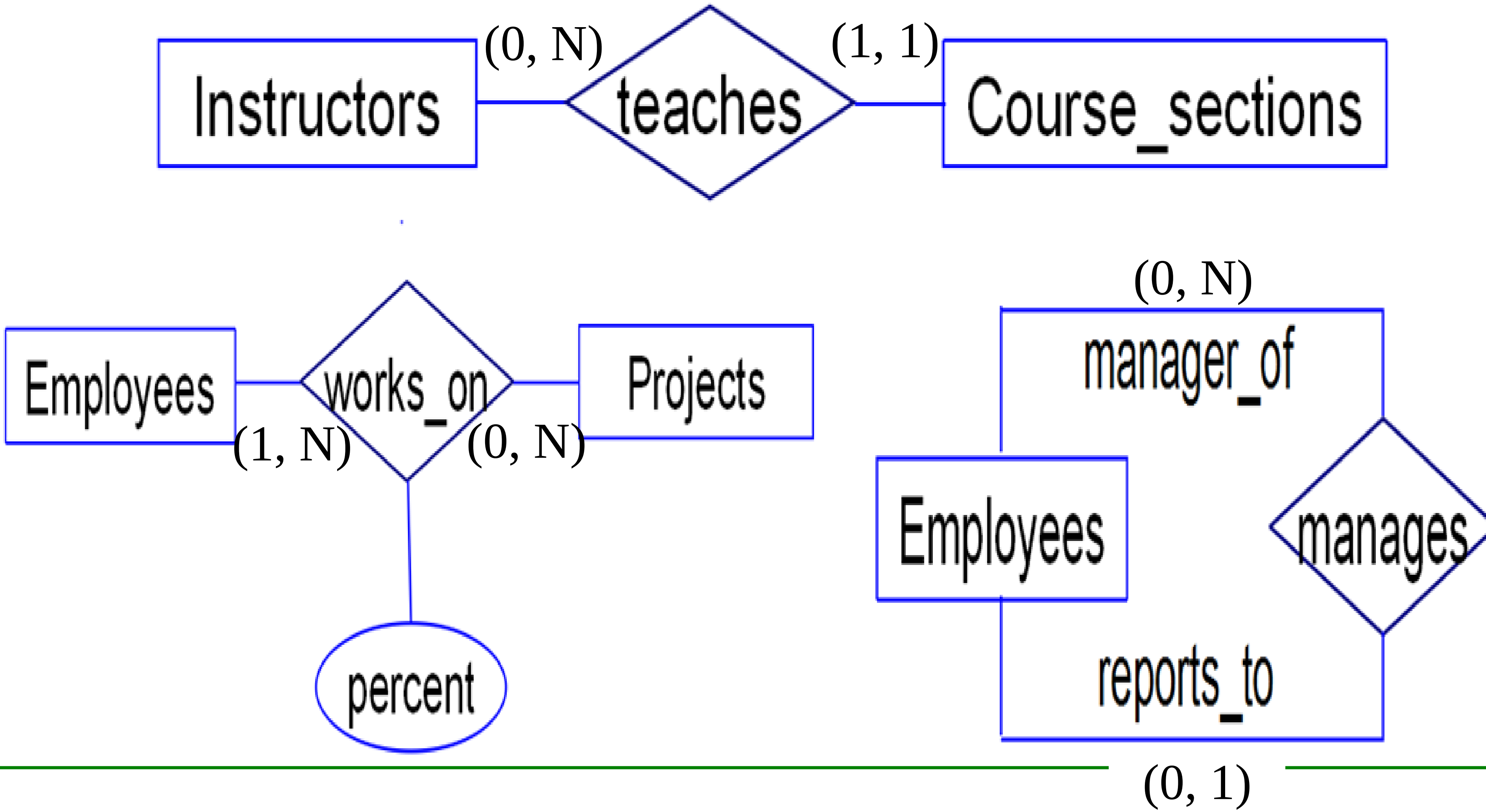
• *single-valued participation:*......

• *multi-valued participation:*......

6.2 Further Details of E-R Diagrams

- **Def 6.2.3**
  - **If min-card(X, R) = 1, X is said to have *mandatory participation（强制参与）* in the relationship R**
  - **if min-card(X, R) = 0, then *optional participation（可选参与）***

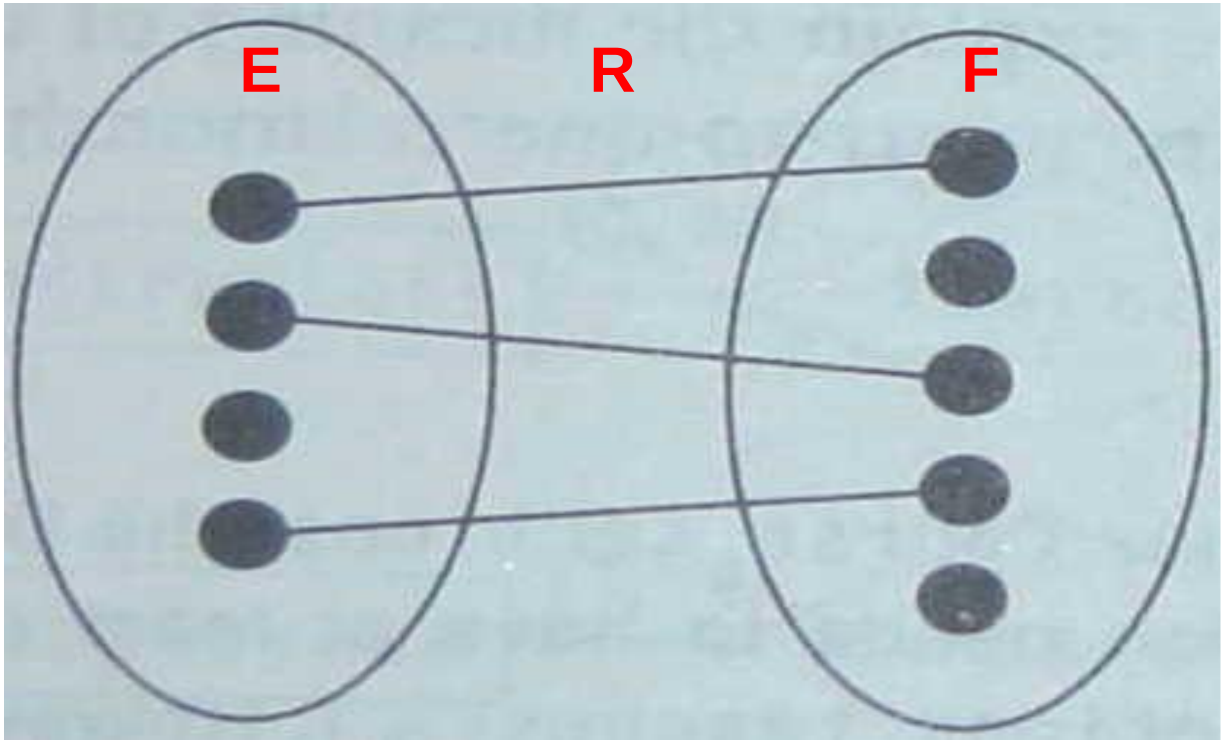




- *mandatory participation:*......
- *optional participation:*......

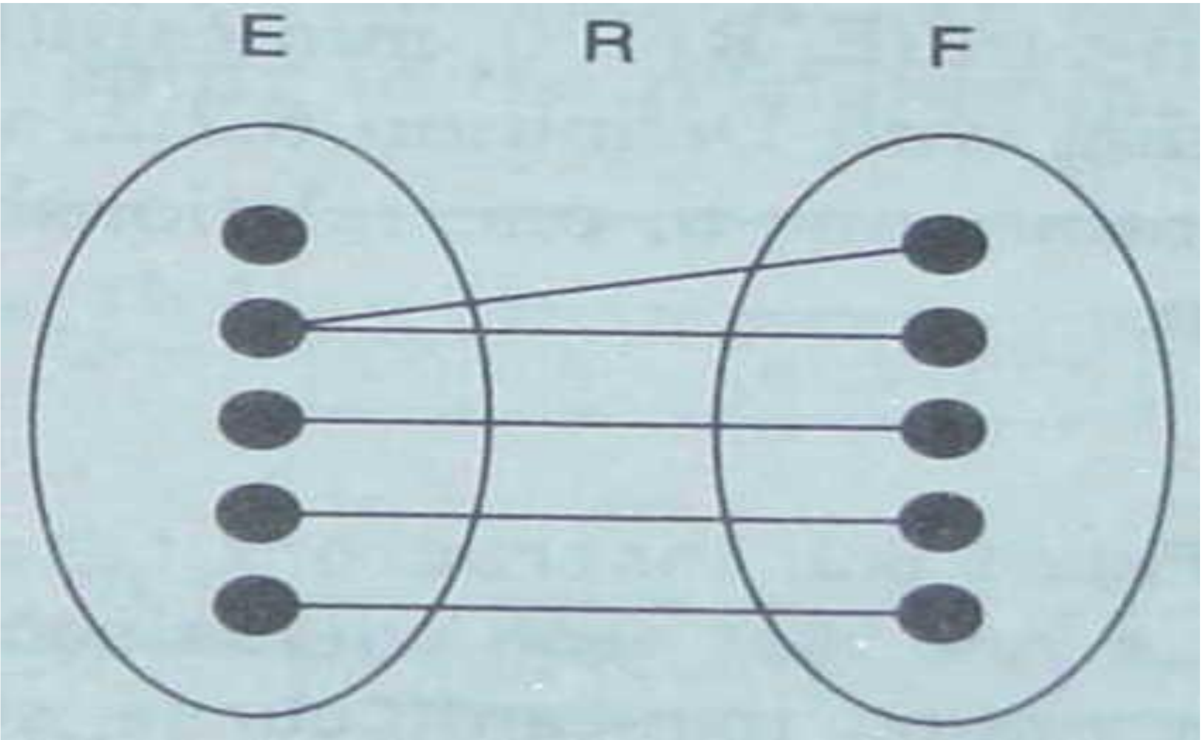
6.2 Further Details of E-R Diagrams

- **One-to-One, Many-to-Many, and Many-to-One Relationship (Figure 6.6)**
  - **One-to-One (1-1) relationship**
    - **both entities are single-valued in the relationship (max-card concept only)**



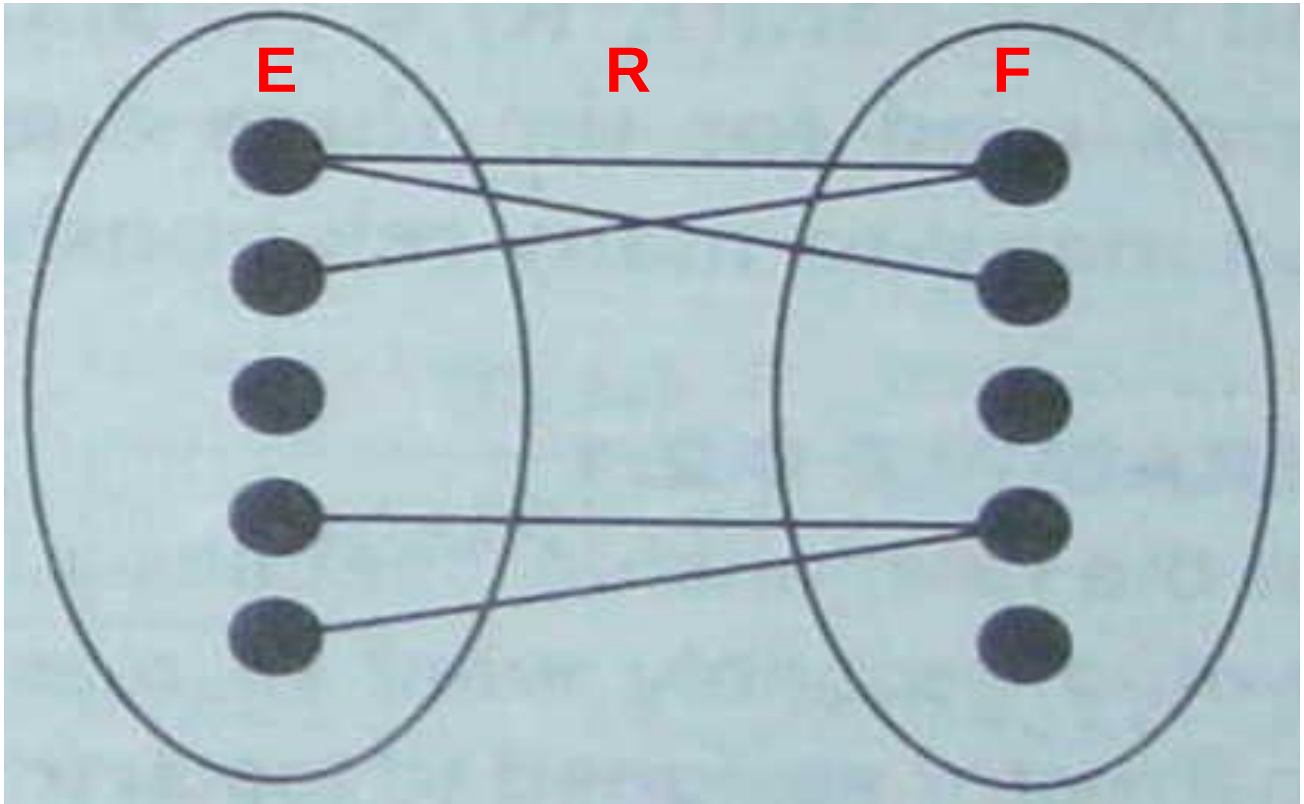
6.2 Further Details of E-R Diagrams

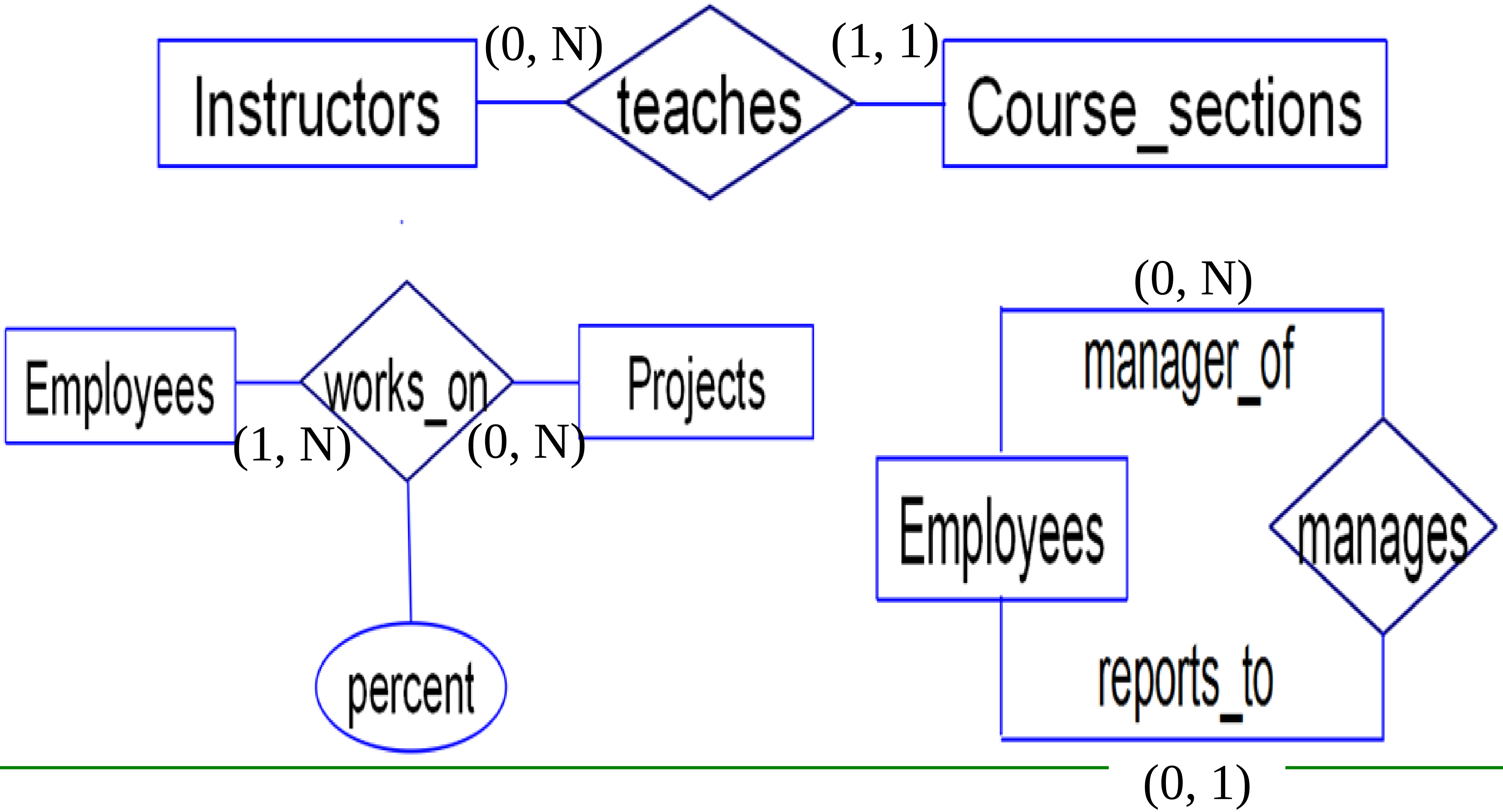
- **One-to-One, Many-to-Many, and Many-to-One Relationship (Figure 6.6)**
  - **Many-to-One (N-1)**
    - **one entity is multi-valued and one is single valued**



6.2 Further Details of E-R Diagrams

- **One-to-One, Many-to-Many, and Many-to-One Relationship (Figure 6.6)**
  - **Many-to-Many (N-N)**
    - **both entities are multi-valued**





- *one-to-one:*......
- *one-to-many:*......
- *many-to-many:*......

6.2 Further Details of E-R Diagrams

- **Transforming Binary Relationships（二元联系）to Relations**

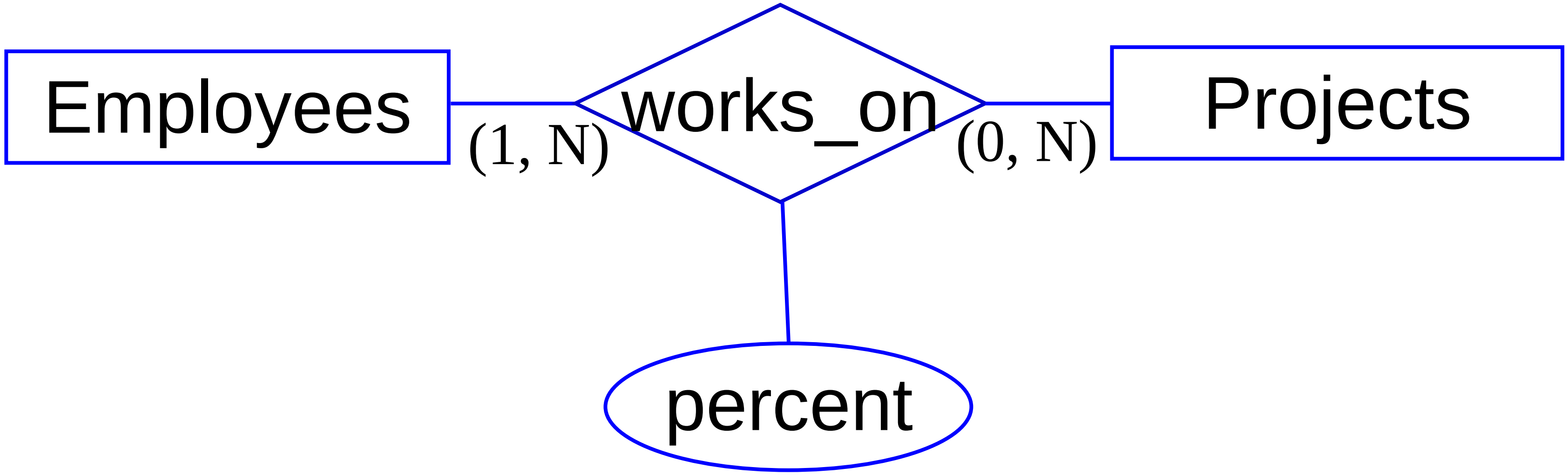
  - **Transformation Rule 3. N-N Relationships**
    - **When two entities E and F take part in a many-to-many binary relationship R, the relationship is mapped to a representative table T in the related relational database design.**

6.2 Further Details of E-R Diagrams

- **Transformation Rule 3. (cont.)**
  - **The table T contains columns for all attributes in the primary keys of both tables transformed from entities E and F.**
    - **this set of columns forms the primary key for the table T.**
  - **T also contains columns for all attributes attached to the relationship.**

6.2 Further Details of E-R Diagrams

- **Example 6.2.2**
  - **Employees(<u>eid</u>, straddr, city, ……)**
  - **Projects(<u>prid</u>, proj_name, due_date)**
  - **works_on(<u>eid, prid</u>, percent)**



6.2 Further Details of E-R Diagrams

- **Transformation Rule 4. N-1 Relationships**
  - **represent with foreign key in entity with single valued participation (the Many side).**
    - **Since max-card(F,R)=1, each row of T is related by a foreign key value to at most one instance of the entity E.**
  - **Example 6.2.3: teachers**
    - **Instructors(<u>insid</u>, lname, ……)**
    - **Course_sections(<u>secid</u>, *<u>insid</u>*, course, …)**

- **Transformation Rule 5&6. 1-1 Relationships**
  - **Optional on one side**
    - **Represent as two tables, foreign key column in one with mandatory participation: column defined to be NOT NULL.**
  - **Mandatory on both sides**
    - **never can break apart. It's appropriate to think of this as two entities in a single table.**

6.3 Additional E-R Concepts

- **Cardinality of Attributes**
  - **Def 6.3.1 (Figure 6.10, pg. 250)**
    - **(0, ?) means don't have to say not null (optional)**
      - **midinitial, emp_address**
    - **(1, ?) means do (mandatory)**
      - **sid, student_name, lname, fname, city, ...**
    - **(?, 1) single valued attribute**
      - **sid, eid**
    - **(?, N) multi-valued**
      - **hobbies**

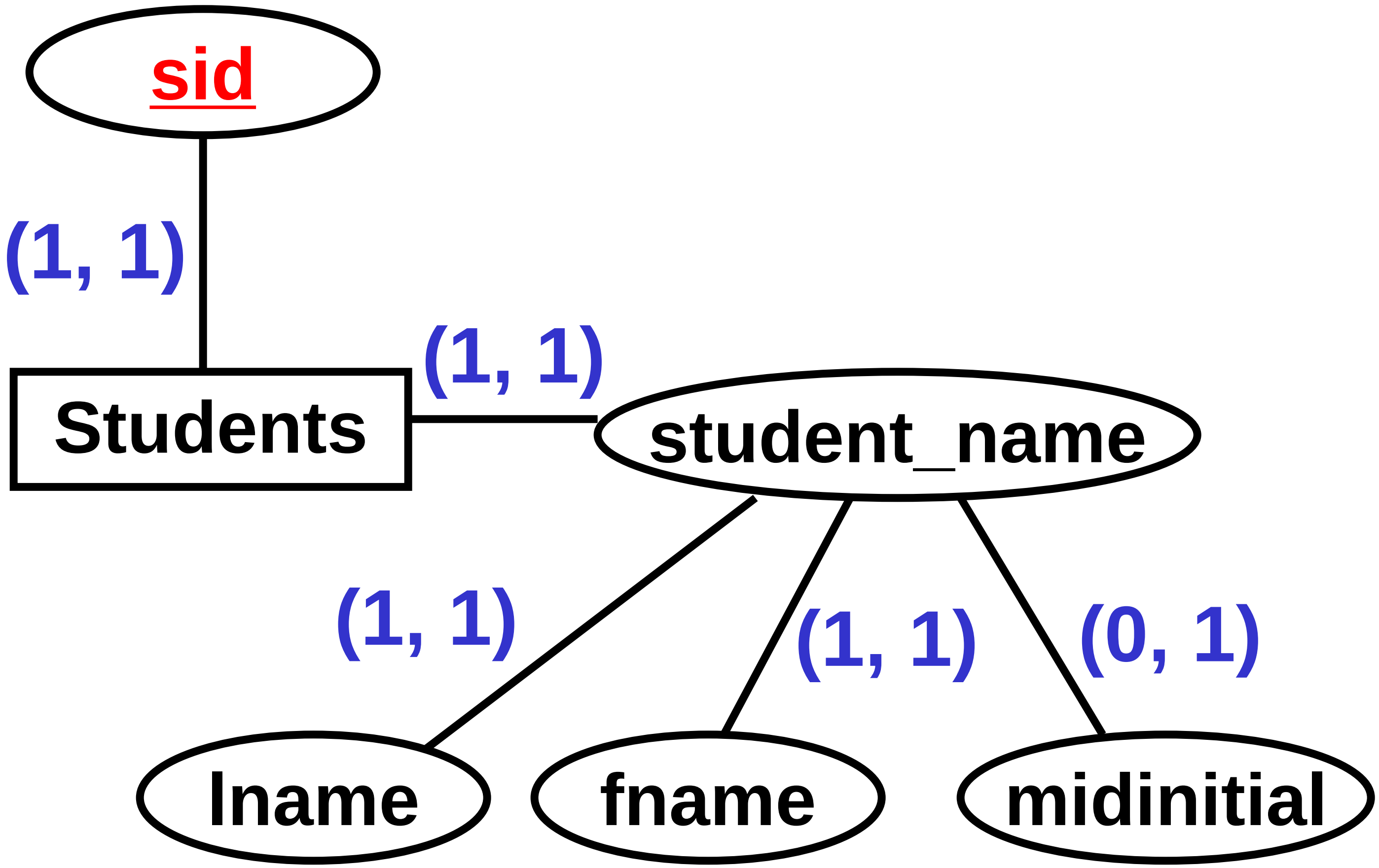


Figure 6.10 (1) Students

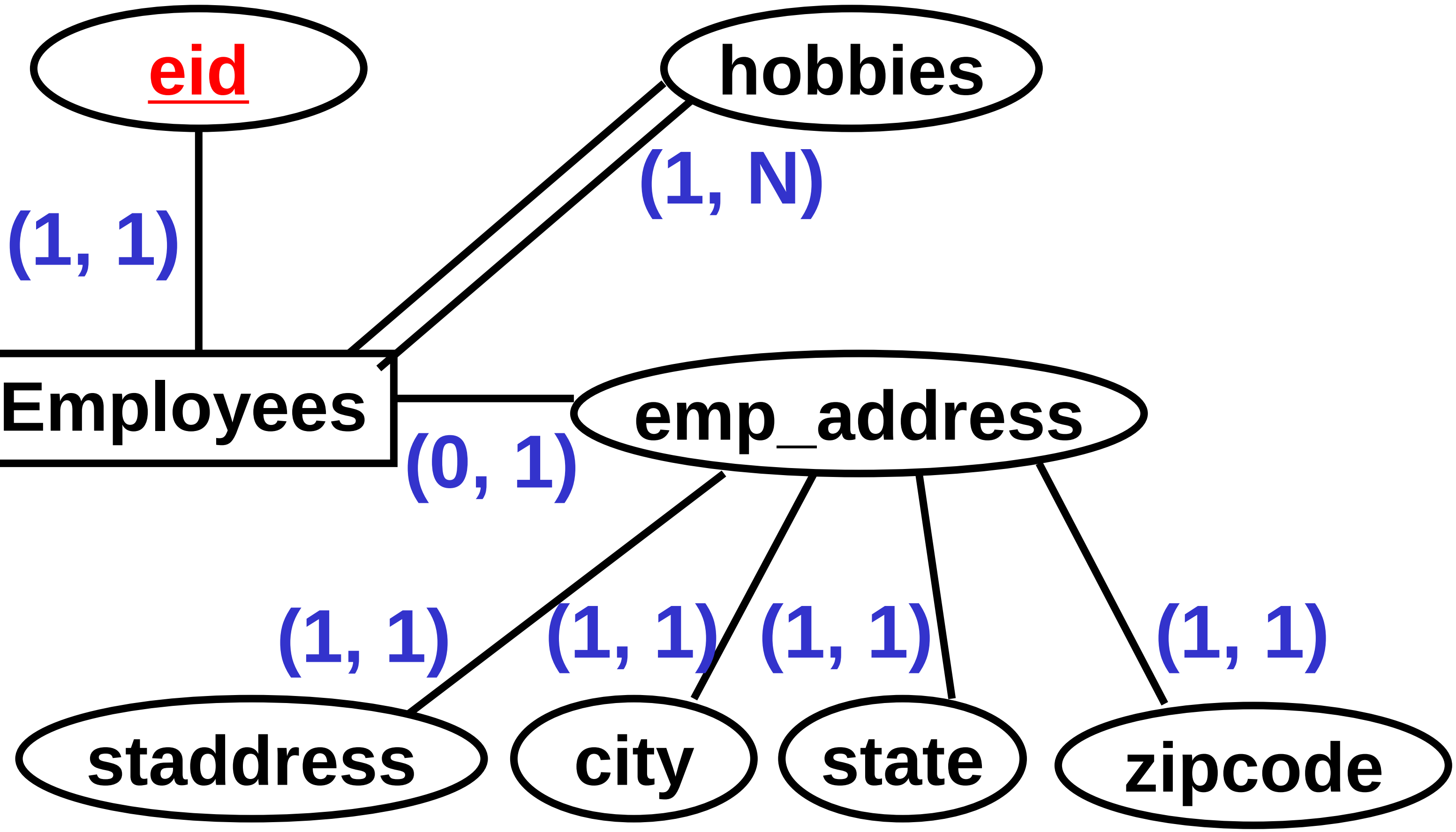
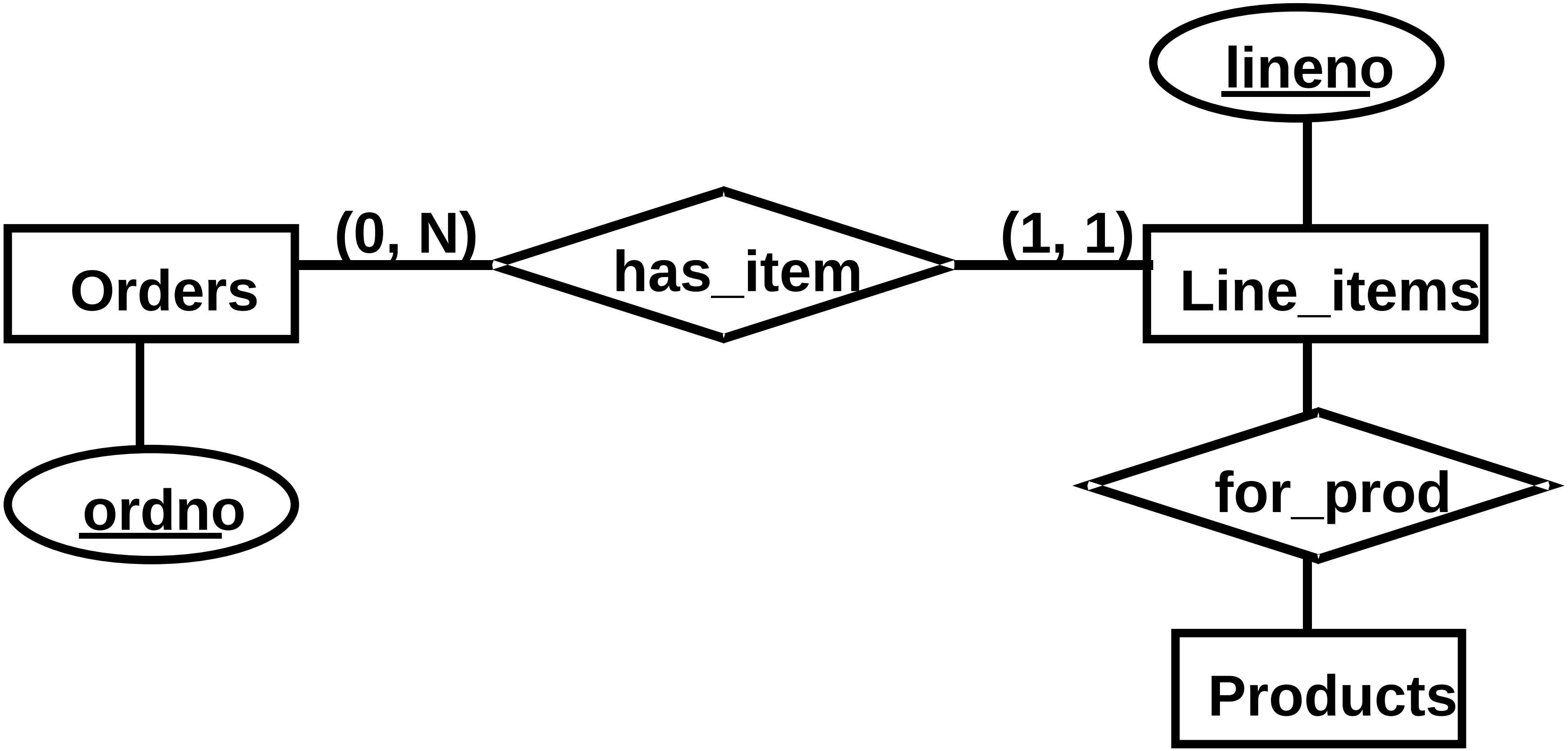


Figure 6.10 (2) Employees

6.3 Additional E-R Concepts

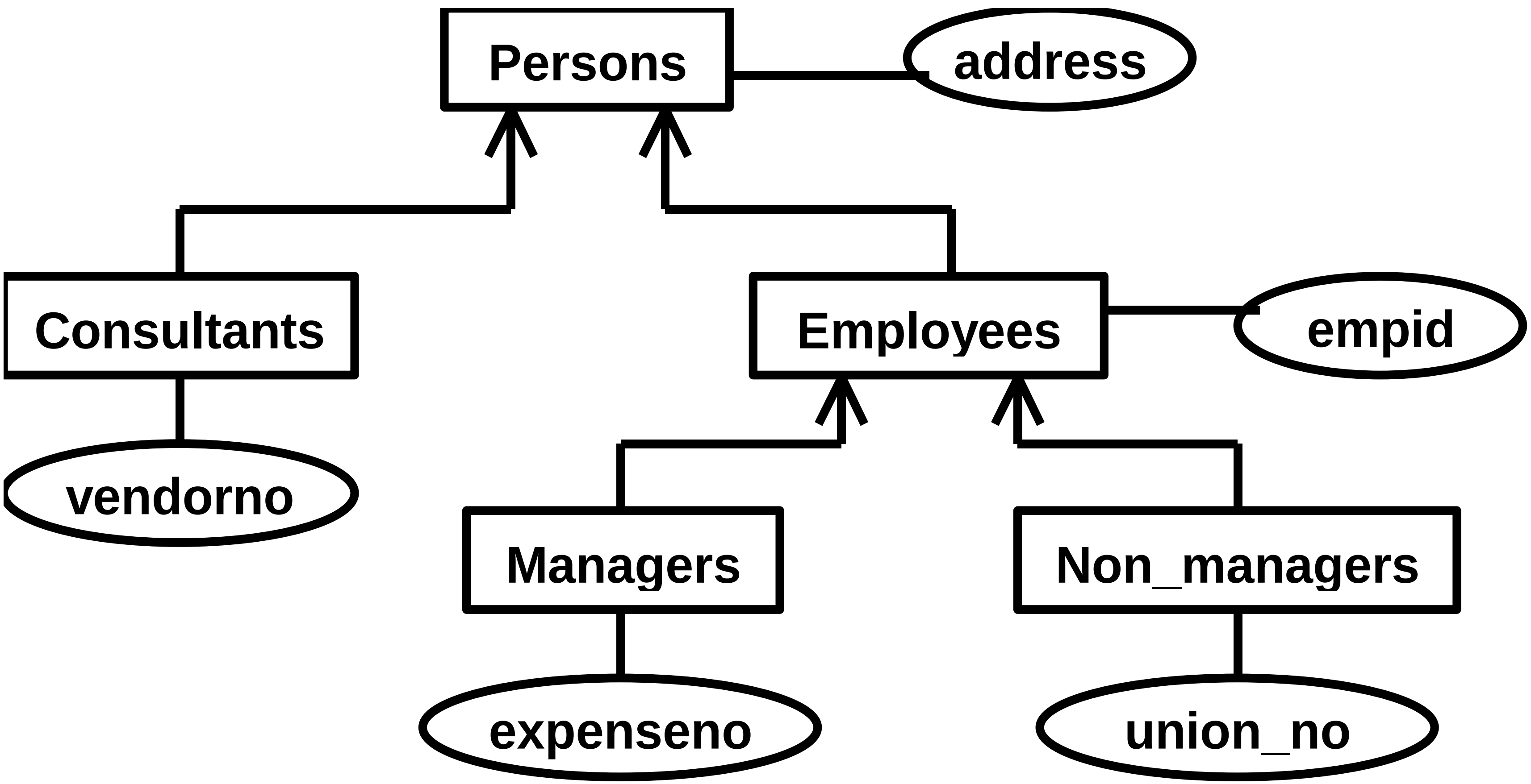
- **Weak Entities**
  - **Def 6.3.2**
    - **A weak entity is an entity whose occurrences are dependent for their existence, through a relationship R, on the occurrence of another (strong) entity.**
    - **Figure 6.11, pg. 251**

- **Figure 6.11**


lineno
(0, N)
(1, 1)
Orders
has_item
Line_items
for_prod
ordno
Products

6.3 Additional E-R Concepts

- **Generalization Hierarchies**
  - Figure 6.12, pg. 252



6.4 Case Study

- **Figure 6.13 & 6.14**
  - **E-R Design for a Simple Airline Reservation Database**
    - **Entity**

      Passengers **Flights**

      Seats **Gates**

    - **Relationship**
      - **Travels_On (Passengers, Flights)**
      - **Has_Seat (Flights, Seats)**
      - **Seat_Assign (Passengers, Seats)**
      - **Marshals (Flights, Gates)**

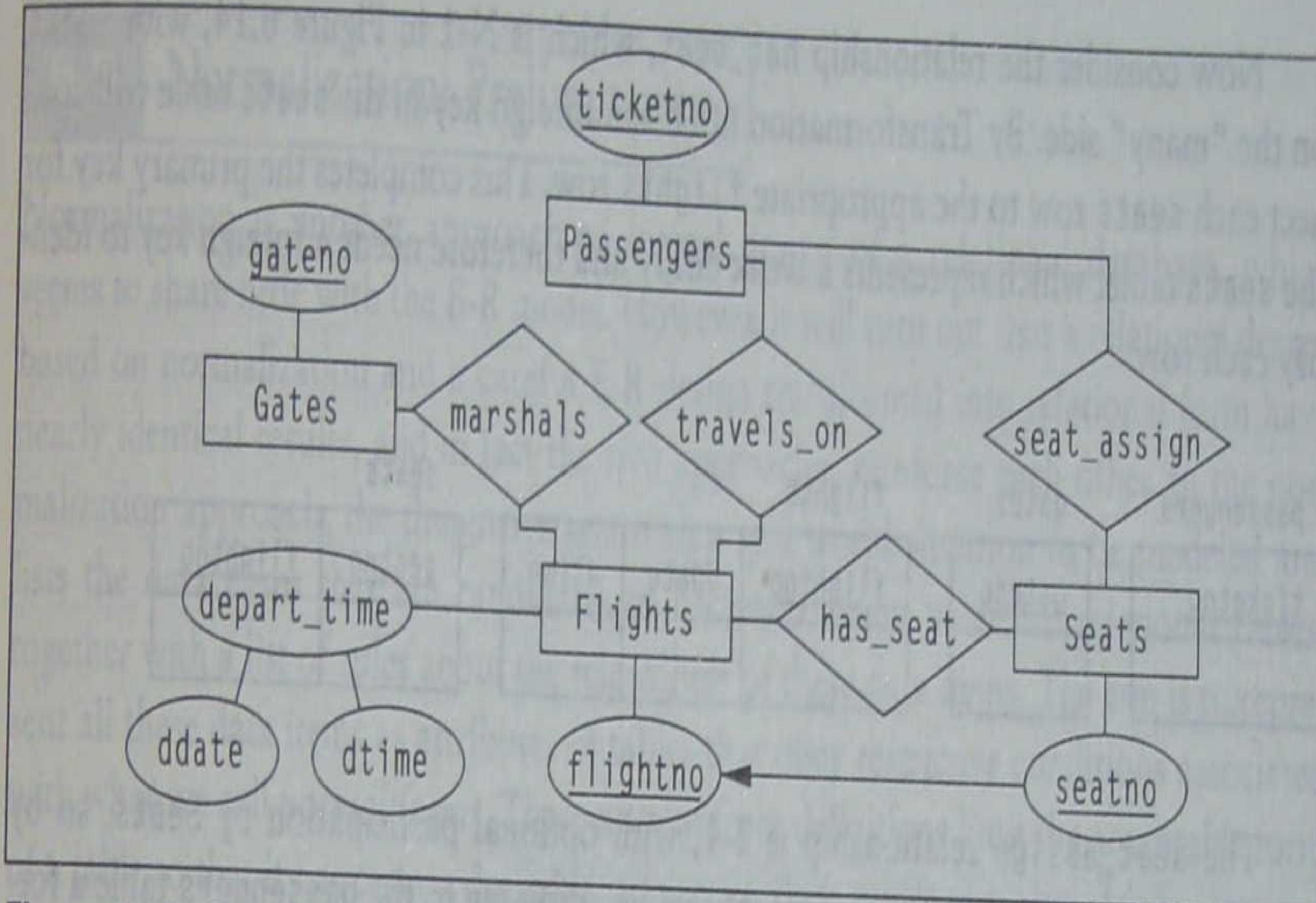


Figure 6.13 Early E-R Design for a Simple Airline Reservation Database

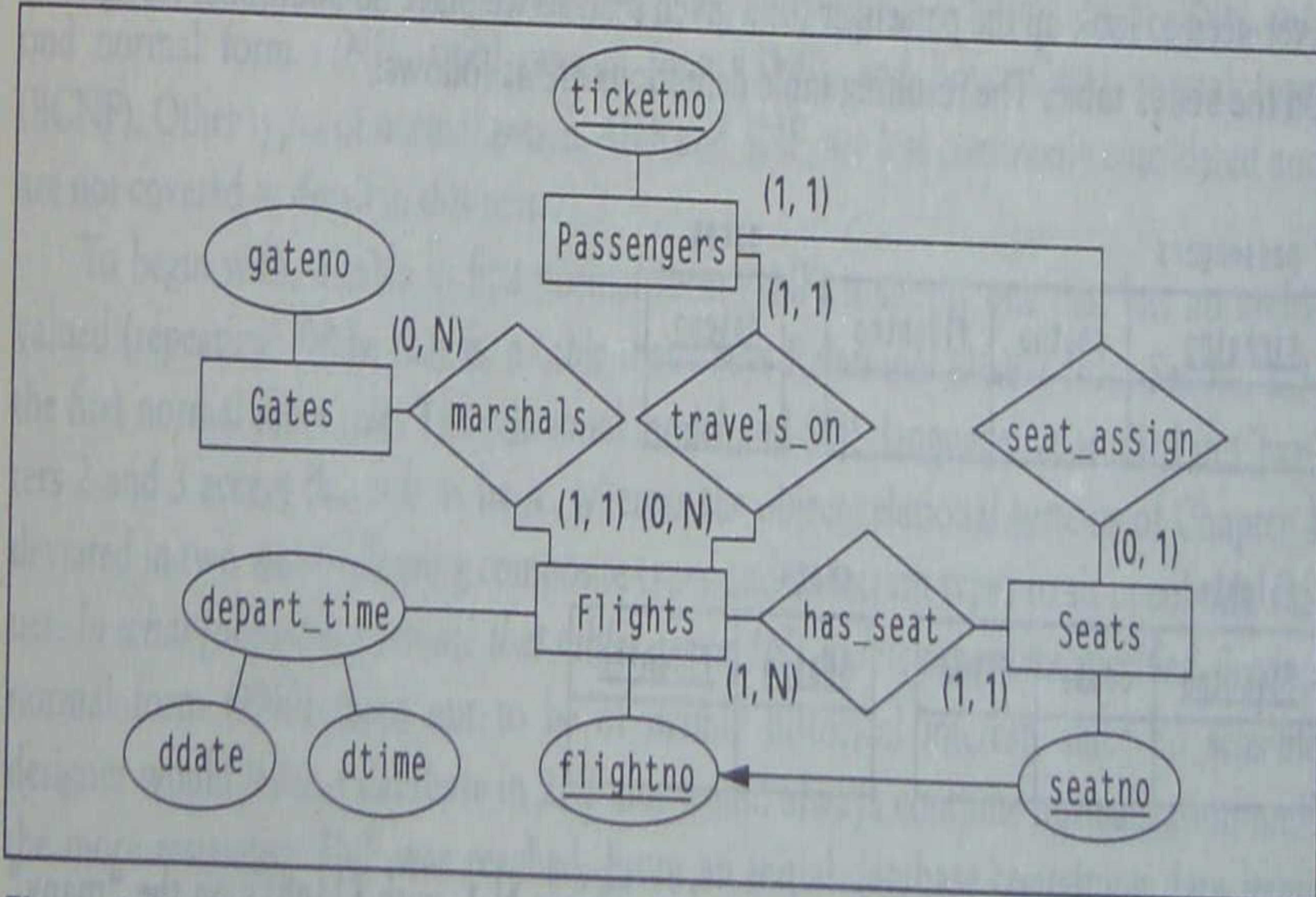


Figure 6.14 E-R Design with Cardinalities for a Simple Airline Reservation Database

6.4 Case Study

- 6.4.1 图书借阅管理
- 6.4.2 篮球联赛信息管理
- 6.4.3 聊天论坛管理
- 6.4.4 邮件信息管理

6.4 Case Study（例6.4.1）

设有一个图书借阅管理数据库，已知：图书的属性有书号（具有唯一性）、书名；读者的属性有借书证号（具有唯一性，每个读者只能有一个借书证号）、姓名、身份证号、住址、电话；出版社的属性有出版社名称（具有唯一性）、地址、联系电话。

其中：每本图书只能有一个出版社出版发行；每个读者可以同时借阅多本图书，也可以在不同时候借阅同一本图书；系统需要记录每本图书被借阅的借阅日期和归还日期。

请用E-R模型表示该数据库系统的概念模型，并将其转换成等价的关系模式。

模型设计

6.4 Case Study（例6.4.2）

- 假设需要设计一个用于NBA篮球比赛的数据库系统，需要记录的信息有：
  - 每个球员的球衣号码、姓名、身高、体重和位置
  - 每个球队的名称和主场使用的体育馆的名称
  - 每场比赛的比赛日期和比分
  - 其中：
    - 每个球员只能效力于一个球队
    - 比赛采用主客场多循环方式
  - 请用E-R模型表示该数据库系统的概念模型，并将其转换成等价的关系模式。

模型设计

6.4 Case Study（例6.4.3）

- 假设需要设计一个用于网络论坛聊天信息管理的数据库系统，需要记录的信息有：
  - 注册用户的用户名，**email**，电话，联系地址
  - 帖子的帖子**ID**，标题，内容
  - 每份帖子的发帖用户，帖子之间的回复关系

请用E-R模型表示该数据库系统的概念模型，并将其转换成等价的关系模式。

模型设计

6.4 Case Study（例6.4.4）

- 有一个邮件管理数据库，其信息如下：
  - 联系人：用户名，**email (关键字)**，电话，联系地址
  - 邮件：邮件**ID**，邮件标题，邮件内容，收信人集合，抄送人集合
  - 邮件之间的回复关系

请用E-R模型表示该数据库系统的概念模型，并将其转换成等价的关系模式。

模型设计

6.5 Normalization(规范化): Preliminaries

- Normal Form (NF，范式)

- A Running Example
  - Figure 6.15 (pg. 256)
    - Employee（emp_id,emp_name,emp_phone）
    - Department（dept_name,dept_phone,dept_mgrname）
    - Skill（skill_id,skill_name,skill_date,skill_lvl）

  - Def. 6.5.1 Update Anomaly
  - Def. 6.5.2 Delete Anomaly, Insert Anomaly

- Functional Dependency (FD, 函数依赖)
- Armstrong's Axioms (Armstrong公理)

- Closure of a Set of FDs (函数依赖集F的闭包)
- FD Set Cover (函数依赖集的覆盖)
- Minimal Cover (最小覆盖)

- Closure of a Set of Attributes (属性集的闭包)

6.6 Functional Dependencies

- **Functional Dependency (FD)**

- **Def. 6.6.1 A→B (A <u>functionally determines</u> B, or B is <u>functionally dependent</u> on A)**
  - **For any rows r_1 and r_2 in T,**

    if $r_1(A) = r_2(A)$ then $r_1(B) = r_2(B)$.

- **Example 6.6.1**

  emp_id, emp_name, emp_phone, dept_name

The SCG Database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	Wang Jian	CS	17	C101	ABC	5
S0001	Wang Jian	CS	17	C102	ACD	5
S0001	Wang Jian	CS	17	C103	BBC	4
S0001	Wang Jian	CS	17	C105	AEF	3
S0001	Wang Jian	CS	17	C110	BCF	4
S0002	Chen Ying	MA	19	C103	BBC	3
S0002	Chen Ying	MA	19	C105	AEF	3
S0003	Zhang Yimou	CS	17	C107	BHD	4

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname ?

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname

Sno → Dept ?

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname

Sno → Dept

Sno → Cno ?

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname

Sno → Dept

Sno → Cno ✗

Cno → Cname ?

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname

Sno → Dept

Sno → Cno ✗

Cno → Cname

Cno → Sno ?

The SCG database

Sno	Sname	Dept	Sage	Cno	Cname	Grade
S0001	WangJian	CS	17	C101	ABC	5
S0001	WangJian	CS	17	C102	ACD	5
S0001	WangJian	CS	17	C103	BBC	4
S0001	WangJian	CS	17	C105	AEF	3
S0001	WangJian	CS	17	C110	BCF	4
S0002	ChenYin	MA	19	C103	BBC	3
S0002	ChenYin	MA	19	C105	AEF	3
S0003	ZhangFei	CS	17	C107	BHD	4

Sno → Sname

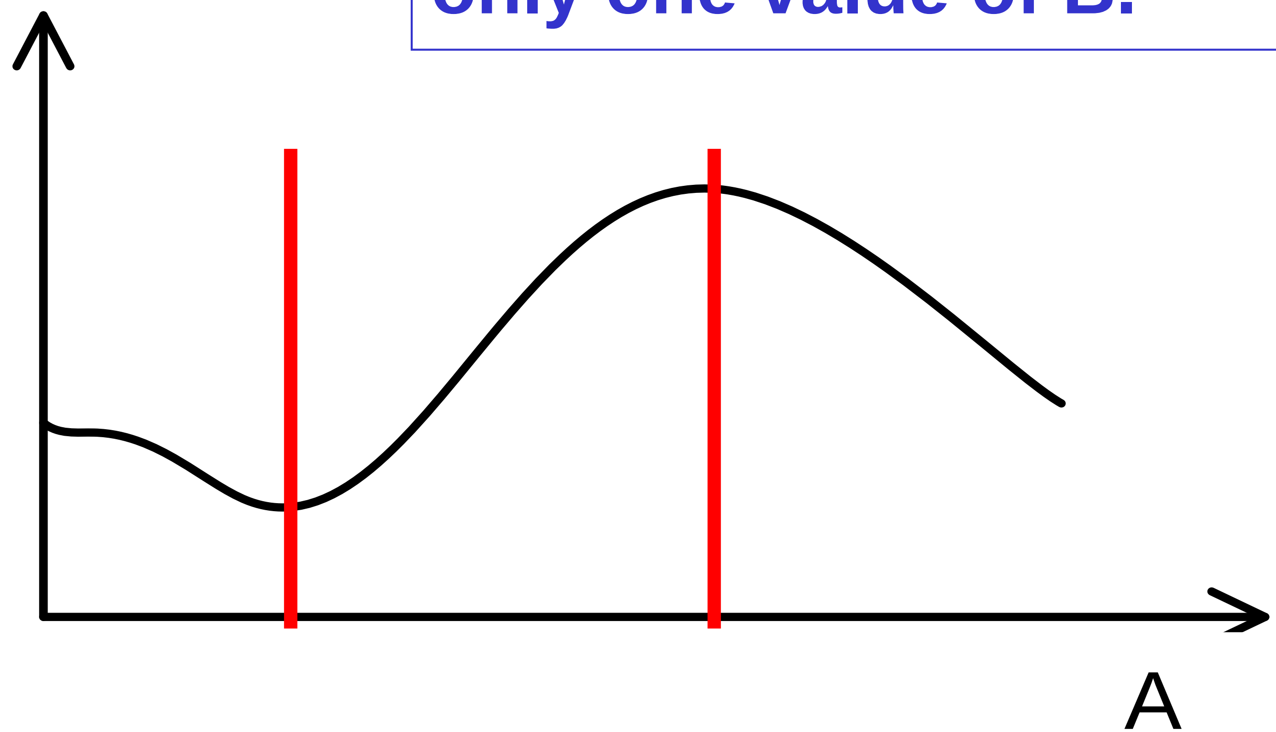
Sno → Dept

Sno → Cno ✗

Cno → Cname

Cno → Sno ✗

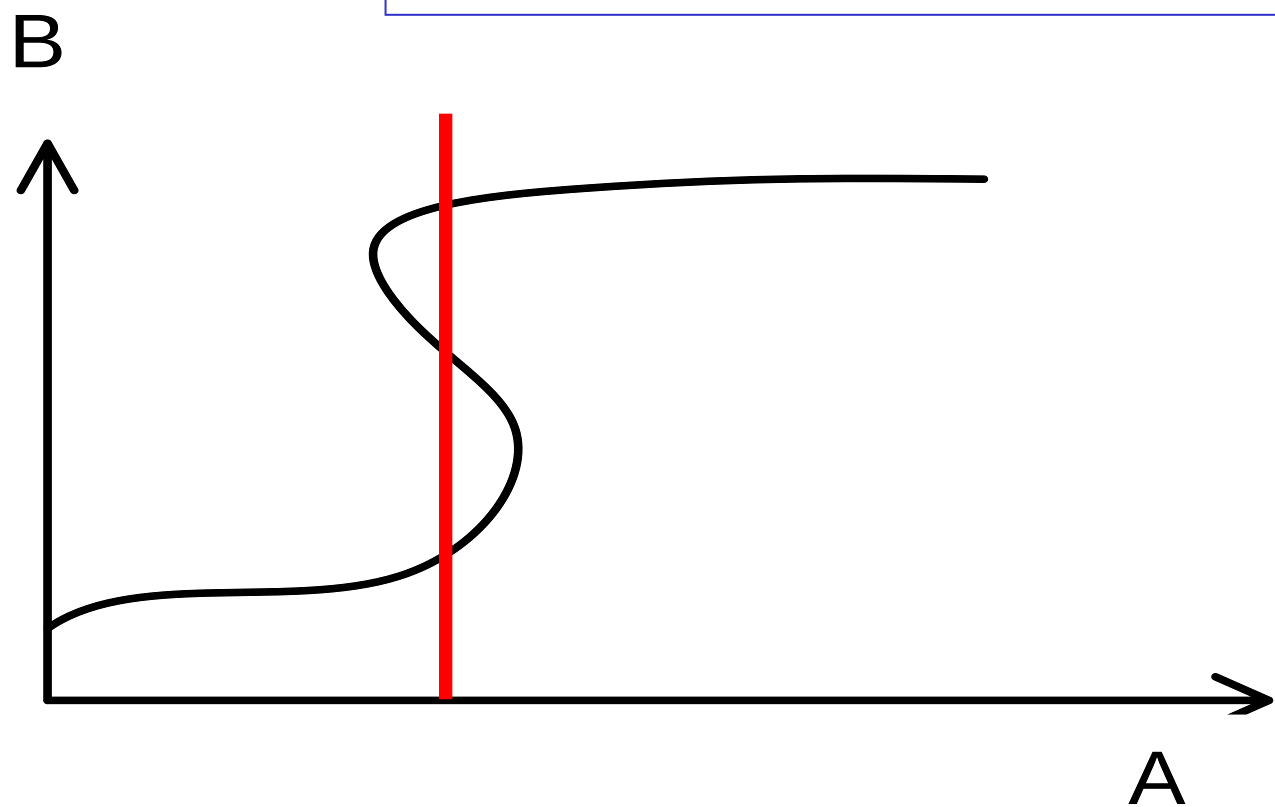
Each value of A corresponds to only one value of B.



A functionally determines B.

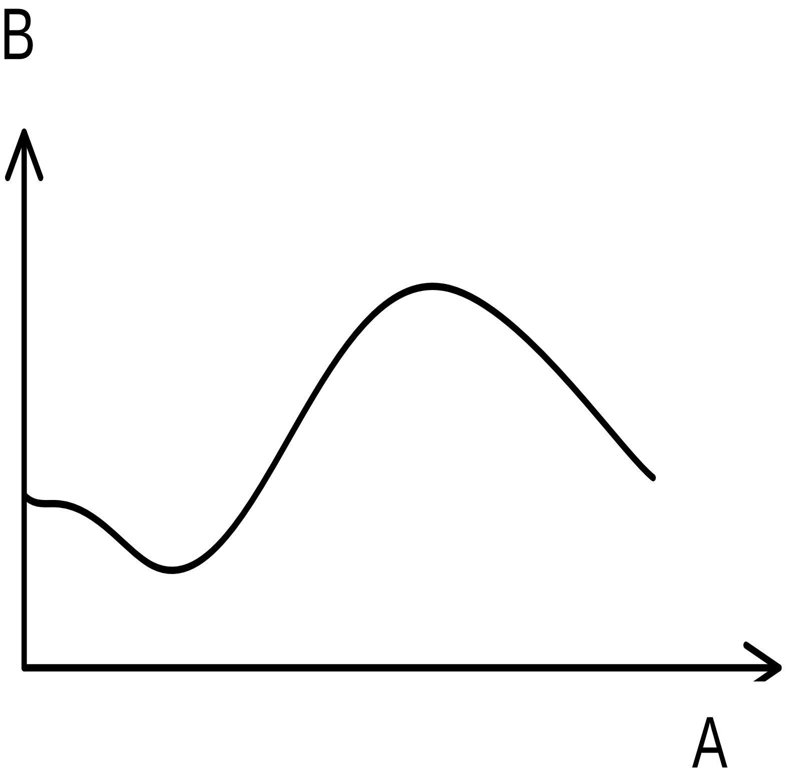
Figure 6.18(a) Graphical Depiction of Functional Dependency

Some values of A correspond to more than one value of B.

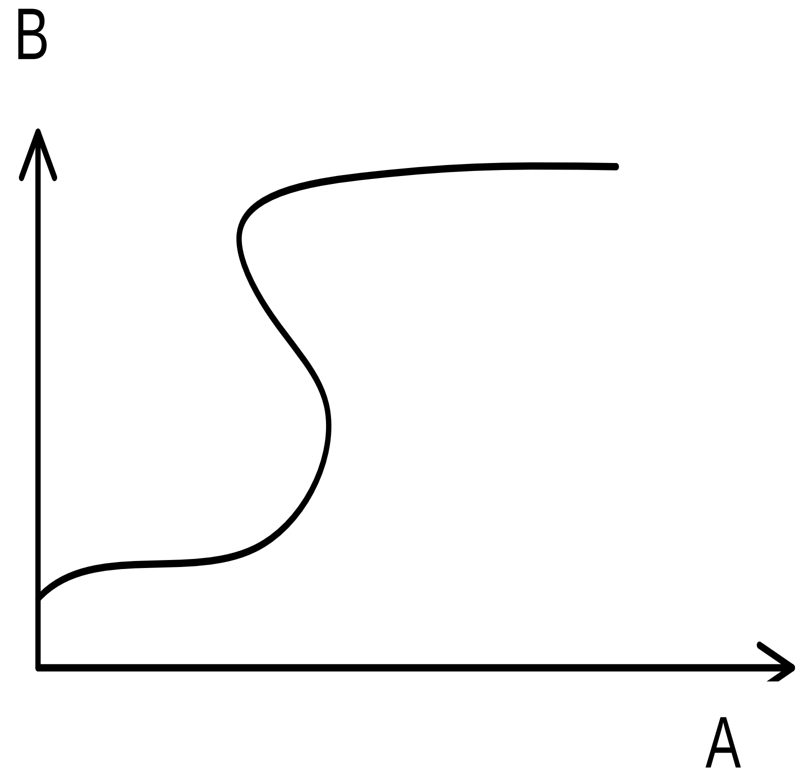


A does not functionally determine B.

Figure 6.18(b) Graphical Depiction of Functional Dependency

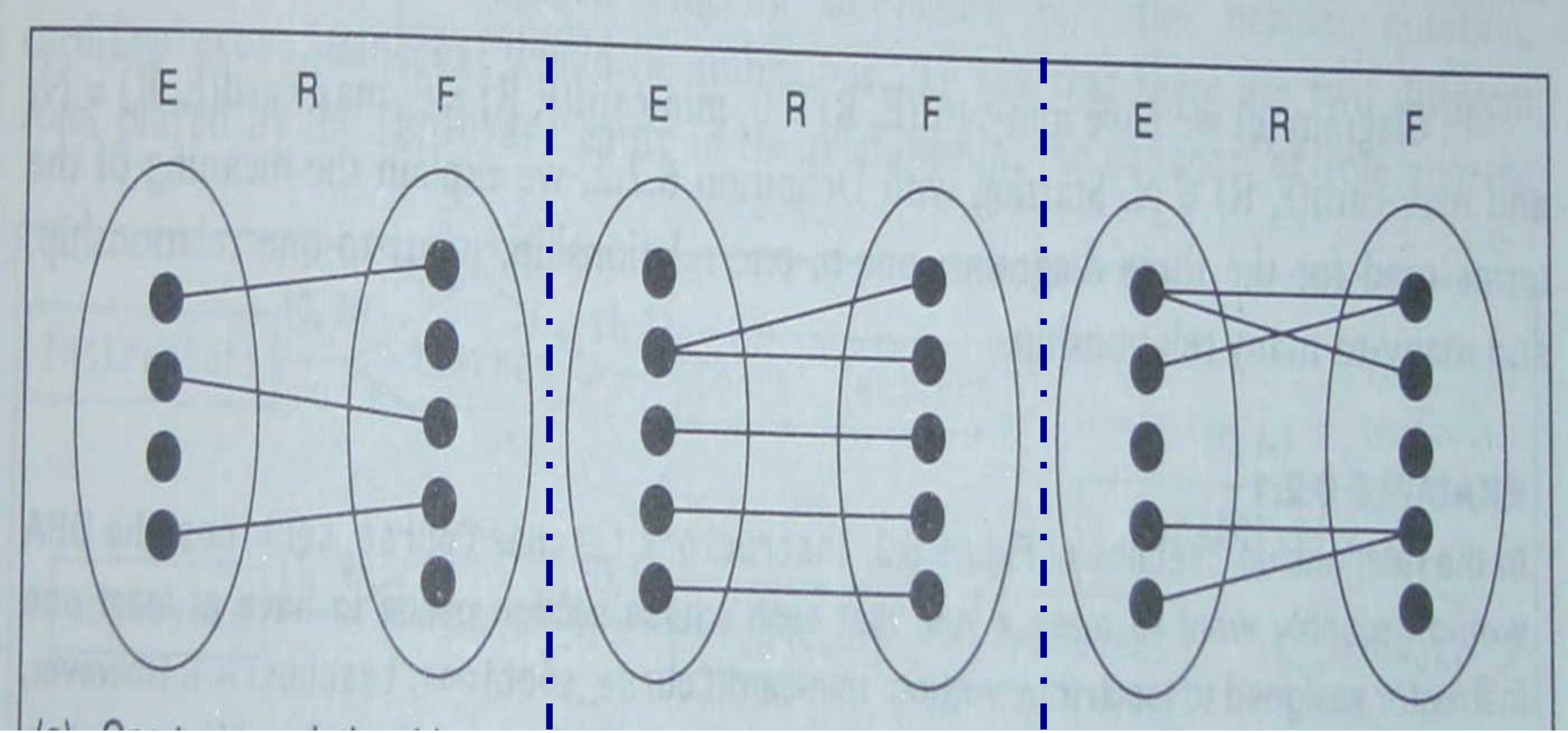


A functionally determines B. Each value of A corresponds to only one value of B.



A does not functionally determine B. Some values of A correspond to more than one value of B.

Figure 6.18 Graphical Depiction of Functional Dependency



E → F

F → E

F → E

(none)

- 我们借用前面的图6.6，假设这里的E和F为两个属性，连线表示：*在关系R中，E和F之间的取值对应关系*

6.6 Functional Dependencies

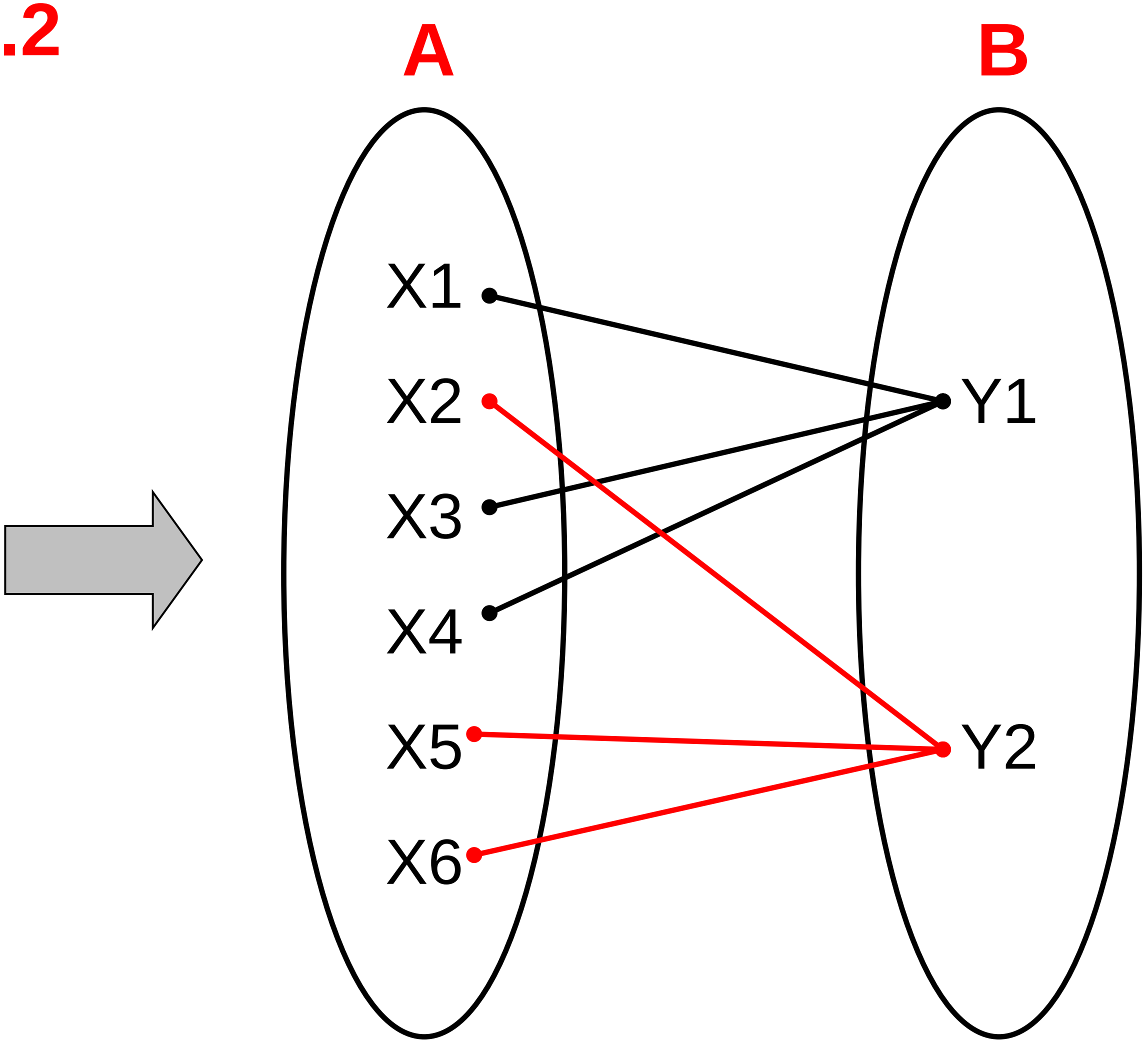
❑ Example 6.6.2

T1

A	B
x1	y1
x2	y2
x3	y1
x4	y1
x5	y2
x6	y2

A→B ?

B→A ?



A → B

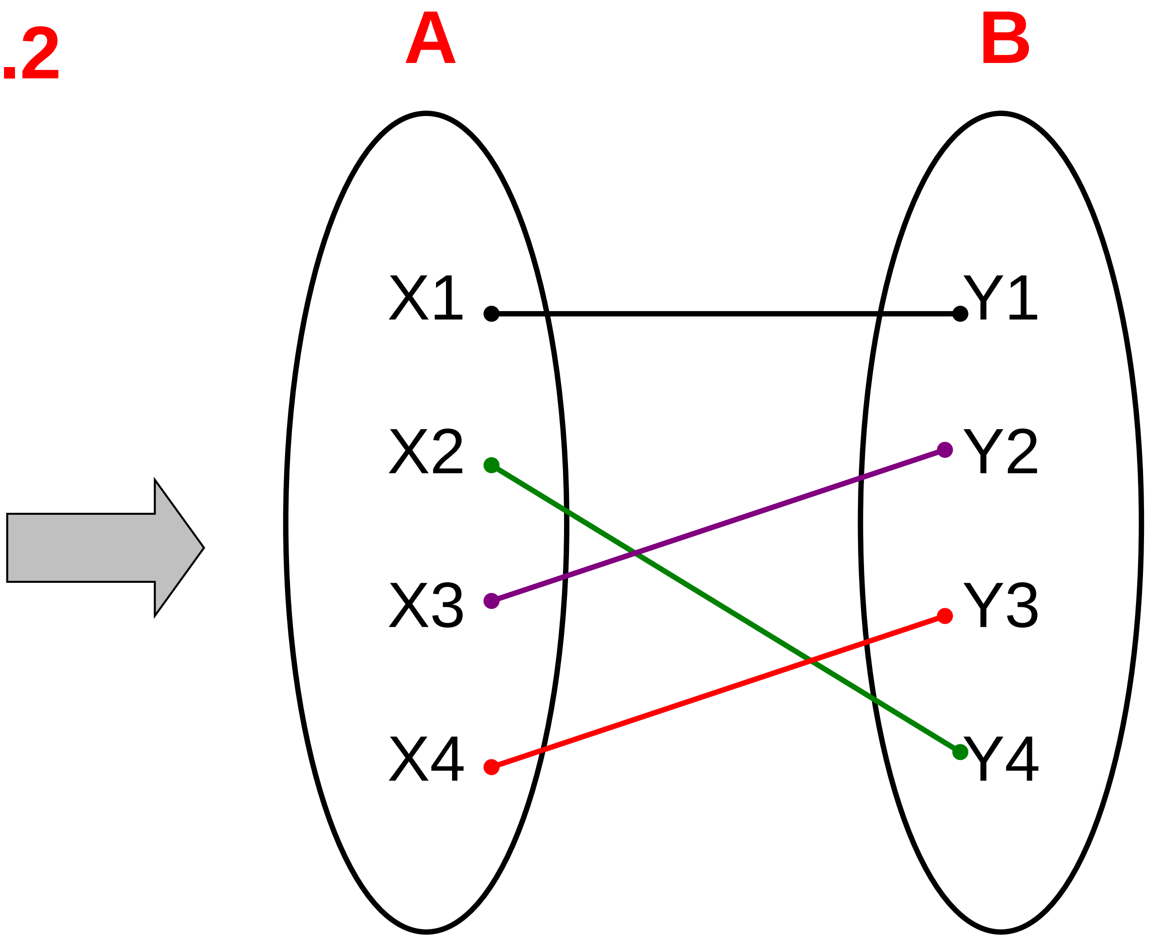
Example 6.6.2

T2

A	B
x1	y1
x2	y4
x3	y2
x4	y3

A→B ?

B→A ?



A → B

B → A

6.6 Functional Dependencies

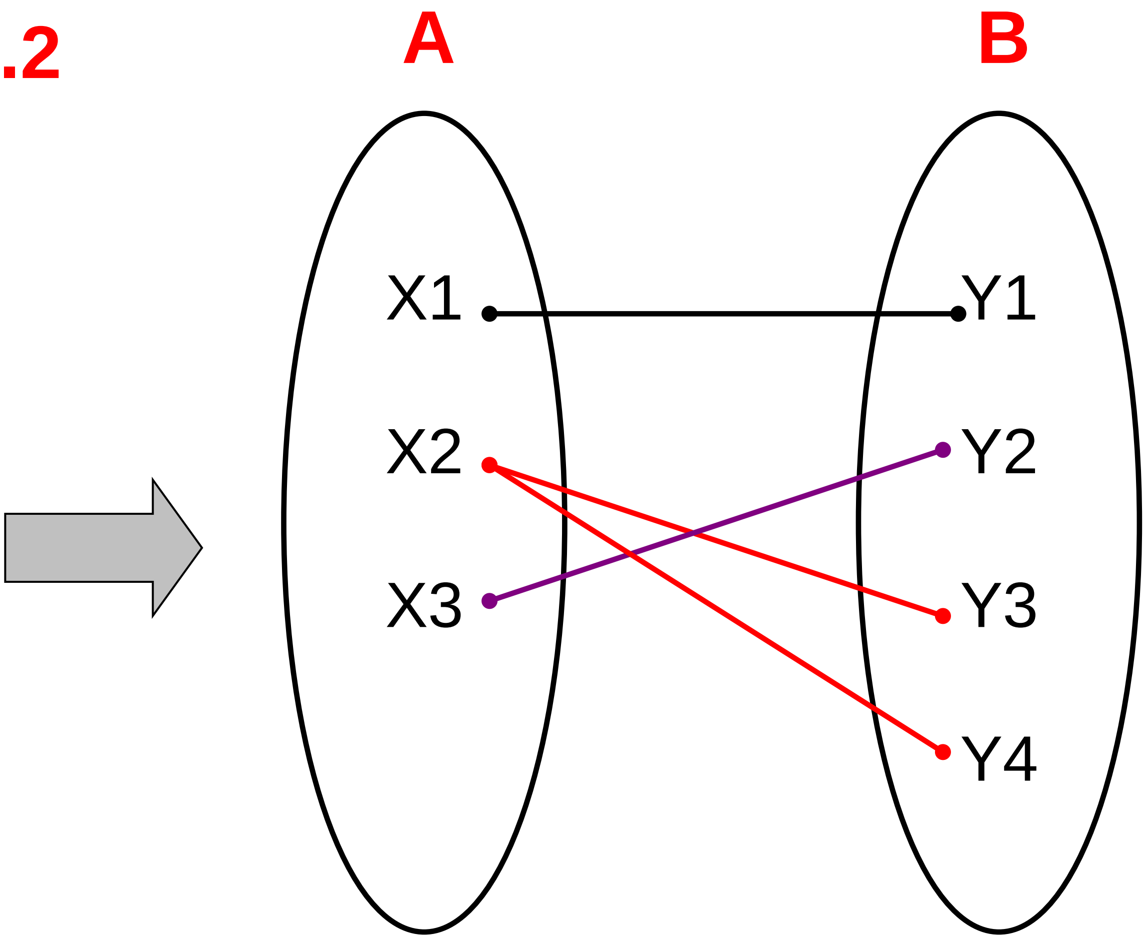
❑ **Example 6.6.2**

T3

A	B
x1	y1
x2	y3
x3	y2
x2	y4

A→B ?

B→A ?



B → A

6.6 Functional Dependencies

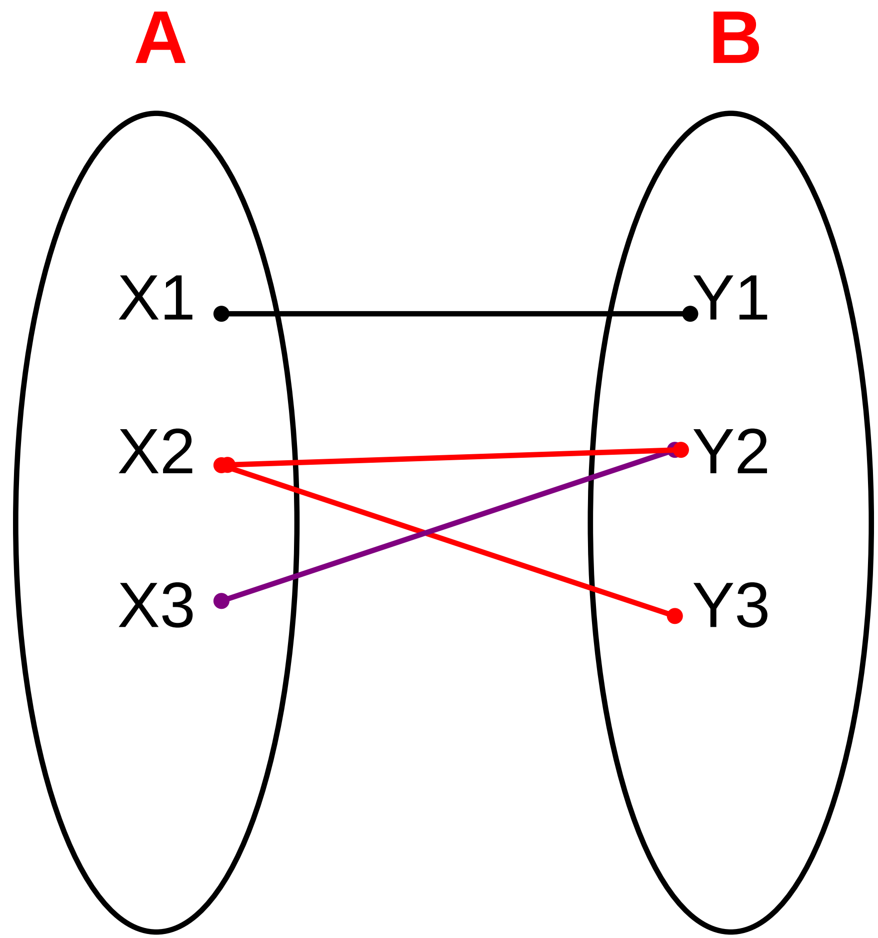
❑ Example 6.6.2

T4

A	B
x1	y1
x2	y2
x3	y2
x2	y3

A→B ?

B→A ?



(none)

6.6 Functional Dependencies

❑ Example 6.6.2

T1

A	B
x1	y1
x2	y2
x3	y1
x4	y1
x5	y2
x6	y2

A→B √

B→A ×

T2

A	B
x1	y1
x2	y4
x3	y2
x4	y3

A→B √

B→A √

T3

A	B
x1	y1
x2	y3
x3	y2
x2	y4

A→B ×

B→A √

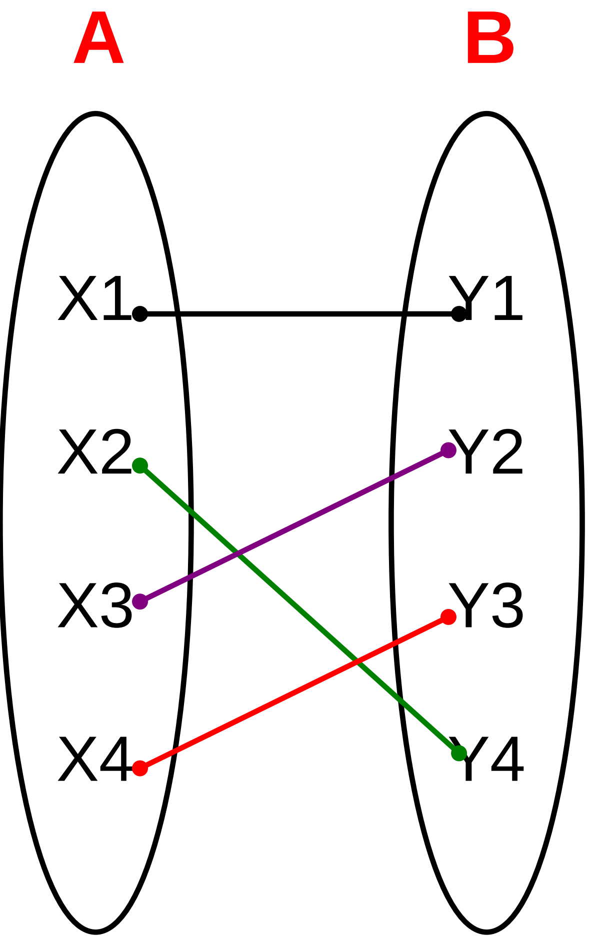
T4

A	B
x1	y1
x2	y2
x3	y2
x2	y3

A→B ×

B→A ×

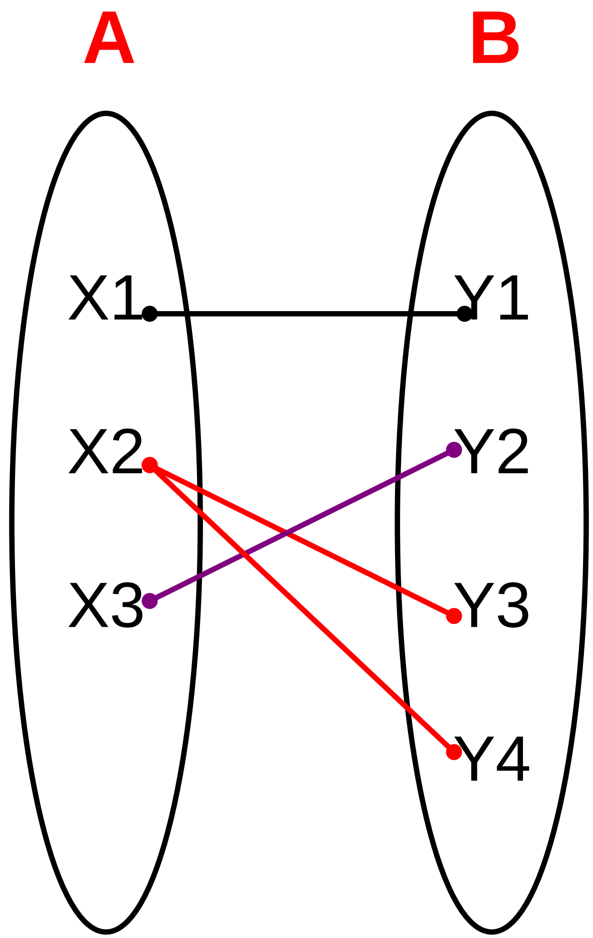
（一对一）



A → B

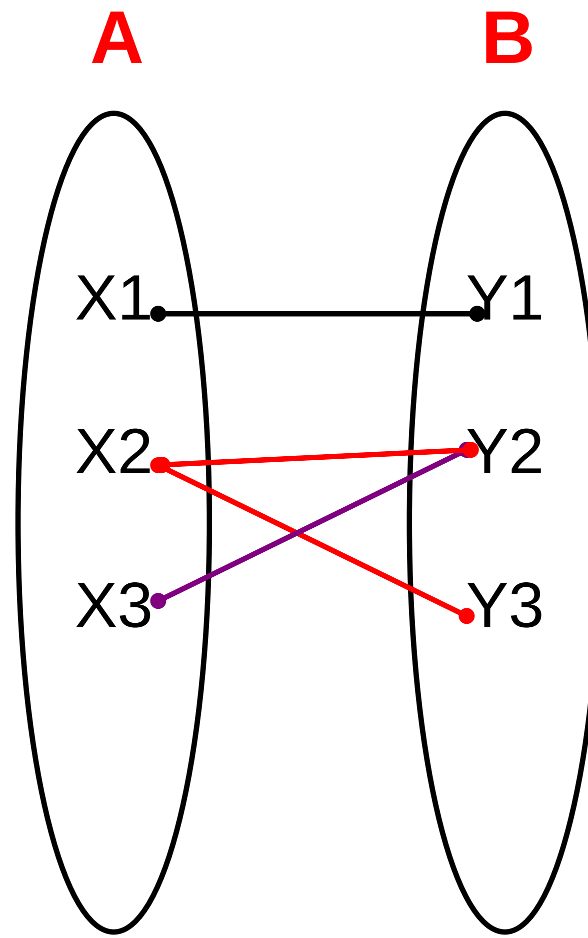
B → A

（一对多）



B → A

（多对多）



(none)

6.6 Functional Dependencies

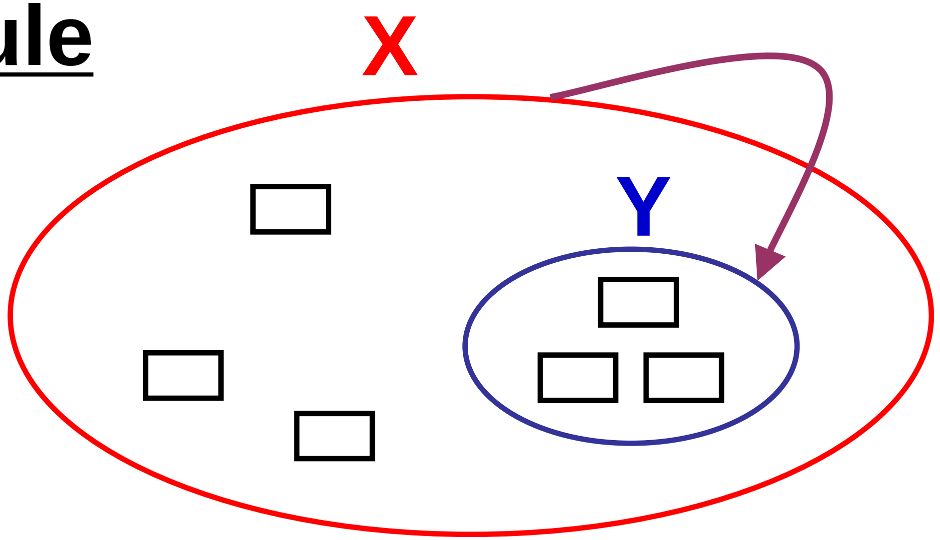
❑Armstrong's Axioms

- 从已知的一些函数依赖，可以推导出另外一些函数依赖，这就需要一系列推理规则。
- 函数依赖的推理规则最早出现在1974年W.W.Armstrong 的论文里，这些规则常被称作"Armstrong 公理"
- 最基本的推理规则只有3条，由这3条基本规则可以定义出若干条'扩充规则'

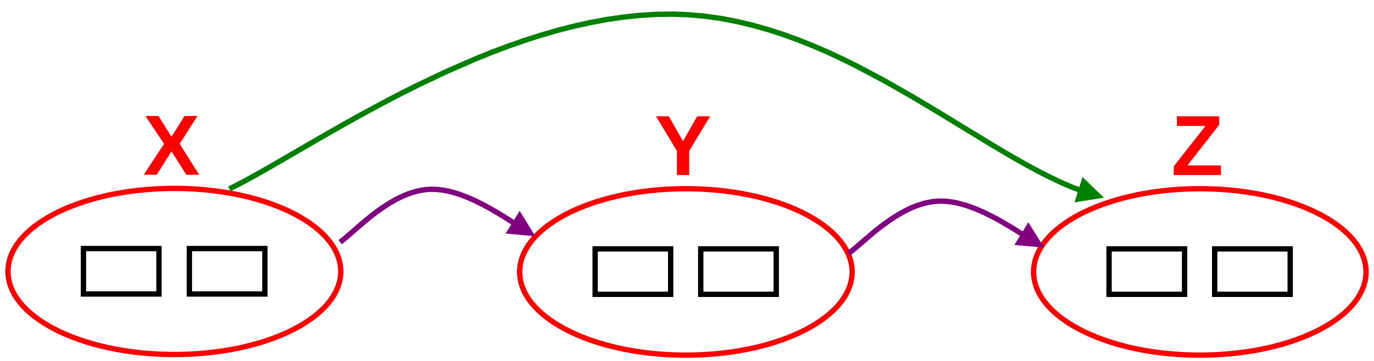
6.6 Functional Dependencies

- **Armstrong’s Axioms**
  - **Rule 1（自反规则）： Inclusion Rule**
    - If $Y \subseteq X$， then $X \rightarrow Y$
  - **Rule 2（传递规则）： Transitivity Rule**
    - If $X \rightarrow Y$ and $Y \rightarrow Z$， then $X \rightarrow Z$
  - **Rule 3（增广规则）： Augmentation rule**
    - If $X \rightarrow Y$, then $XZ \rightarrow YZ$
- **Figure 6.19 (pg. 262)**

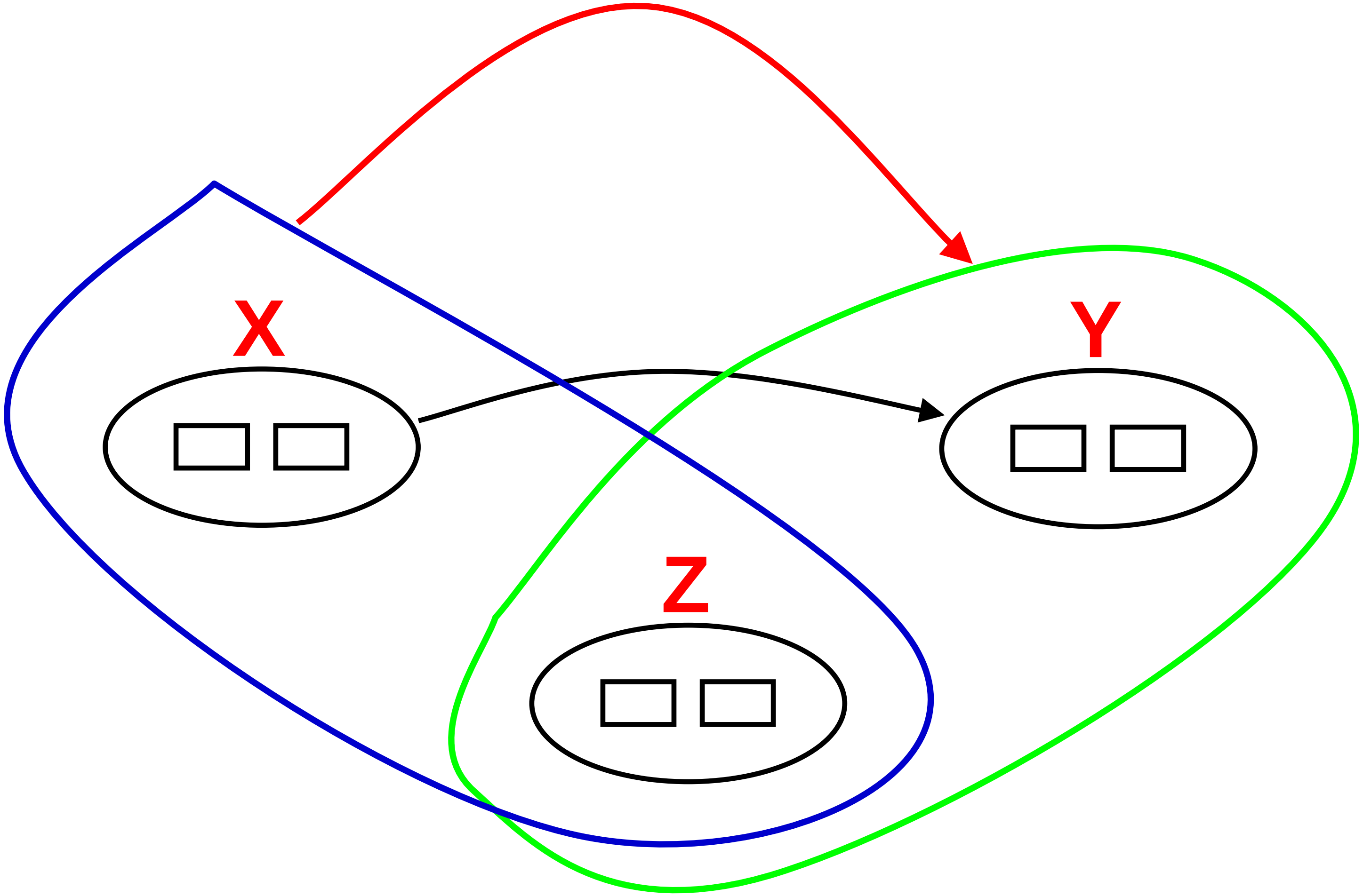
1. Inclusion rule



2. Transitivity rule



3. Augmentation rule



6.6 Functional Dependencies

❑Rule 1: Inclusion Rule

If $Y \subseteq X$，then $X \rightarrow Y$

Proof：

- 设t_1, t_2是关系R中的任意两个元组($t_1 \in R$, $t_2 \in R$)，且它们在属性集X上的值相等($t_1[X] = t_2[X]$)
- 由于Y是X的子集，即$X \supseteq Y$
- 因此必有$t_1[Y] = t_2[Y]$

Rule 2: Transitivity Rule

If $X \rightarrow Y$ and $Y \rightarrow Z$， then $X \rightarrow Z$

Proof:

- 设 $t_1 \in R$, $t_2 \in R$, 如果 $t_1[X] = t_2[X]$ ………………….(1)
- 由(1)及$X \rightarrow Y$得：$t_1[Y] = t_2[Y]$ ……………………… (2)
- 由(2)及$Y \rightarrow Z$得：$t_1[Z] = t_2[Z]$

❑Rule 3: Augmentation rule

If $X \rightarrow Y$, then $XZ \rightarrow YZ$

Proof：

- 设 $t_1 \in R$, $t_2 \in R$, 如果 $t_1[XZ] = t_2[XZ]$, 则：

  $t_1[X] = t_2[X]$ ……………………………(1)

  $t_1[Z] = t_2[Z]$ ……………………………(2)

- 由(1)及$X \rightarrow Y$得： $t_1[Y] = t_2[Y]$ …………(3)

- 由(2)及(3)得：$t_1[YZ] = t_2[YZ]$

6.6 Functional Dependencies

- **Theorem 6.6.8 Some Implications of Armstrong's（pg. 263）**
  - Rule 4: Union Rule （合并规则）

    If X → Y and X → Z， then X → YZ
  - Rule 5: Decomposition Rule （分解规则）

    If X → YZ， then X → Y， and X → Z
  - Rule 6: Pseudotransitivity Rule （伪传递规则）

    If X → Y， and WY → Z， then XW → Z
  - Rule 7: Set accumulation rule （聚积规则）

    If X → YZ and Z → W， then X → YZW

6.6 Functional Dependencies

- **Rule 4: Union Rule**

  If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

- **Proof:**
  1) **We have (a) $X \rightarrow Y$ and (b) $X \rightarrow Z$.**
  2) **By Armstrong's Augmentation rule and (a), we have (c) $XX \rightarrow XY$. But XX is X UNION X = X, so (c) can be rewritten (d) $X \rightarrow XY$.**
  3) **Now by (b) and augmentation, we have (e) $XY \rightarrow YZ$.**
  4) **And by (d) and (e) and transitivity, we have $X \rightarrow YZ$, the desired result.**

6.6 Functional Dependencies

- **Rule 5: Decomposition Rule**

  If $X \rightarrow YZ$, then $X \rightarrow Y$, and $X \rightarrow Z$

- **Proof:**

  1) **We have (a) $X \rightarrow YZ$.**

  2) **By Armstrong's Inclusion Rule, we have (b) $YZ \rightarrow Y$ and (c) $YZ \rightarrow Z$.**

  3) **By (a) and (b) and Armstrong's Transitivity Rule, we have $X \rightarrow Y$.**

  4) **By (a) and (c) and Armstrong's Transitivity Rule, we have $X \rightarrow Z$.**

- **Rule 6: Pseudotransitivity Rule**

  If $X \rightarrow Y$, and $WY \rightarrow Z$, then $XW \rightarrow Z$

- **Proof:**

  1) We have (a) $X \rightarrow Y$ and (b) $WY \rightarrow Z$.
  2) By Armstrong's Augmentation Rule and (a), we have (c) $WX \rightarrow WY$.
  3) By (c) and (b) and Armstrong's Transitivity Rule, we have (d) $WX \rightarrow Z$.
  4) But $WX = XW$, so (d) can be rewritten $XW \rightarrow Z$, the desired result.

Rule 7: Set Accumulation Rule

If $X \rightarrow YZ$ and $Z \rightarrow W$, then $X \rightarrow YZW$

Proof:

1) We have (a) $X \rightarrow YZ$ and (b) $Z \rightarrow W$.
2) By Armstrong's Augmentation Rule and (b), we have (c) $YZ\,Z \rightarrow YZ\,W$.
3) But $YZ\,Z = (Y \cup Z) \cup Z = Y \cup (Z \cup Z) = Y \cup Z = YZ$, so (c) can be rewritten (d) $YZ \rightarrow YZW$.
4) By (a) and (d) and Armstrong's Transitivity Rule, we have $X \rightarrow YZW$, the desired result.

6.6 Functional Dependencies

- **Example 6.6.4: relation T**

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

- **Find FDs in relation T**

6.6 Functional Dependencies

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

- 首先考虑决定因素和依赖因素都是单个属性的情况：

$A \rightarrow B$ √
$A \rightarrow C$ ✗
$A \rightarrow D$ ✗

$B \rightarrow A$ ✗
$B \rightarrow C$ ✗
$B \rightarrow D$ ✗

$C \rightarrow A$ ✗
$C \rightarrow B$ √
$C \rightarrow D$ ✗

$D \rightarrow A$ √
$D \rightarrow B$ √
$D \rightarrow C$ √

6.6 Functional Dependencies

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

- 也可以按照如下顺序考虑可能存在的函数依赖情况：

$A \rightarrow B$ √　　$B \rightarrow A$ ✗　　$C \rightarrow A$ ✗　　$D \rightarrow A$ √

$A \rightarrow C$ ✗　　$B \rightarrow C$ ✗　　$C \rightarrow B$ √　　$D \rightarrow B$ √

$A \rightarrow D$ ✗　　$B \rightarrow D$ ✗　　$C \rightarrow D$ ✗　　$D \rightarrow C$ √

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

A → B　　　C → B　　　D → ABC

其次，再考虑决定因素是多个属性的情况：

1) 在FD的左边不需要考虑含有属性 D 的情况，why ?

2) 在FD的左边不需要考虑含有属性 B 的情况，why ?

因此只需要考虑下述的FD是否成立：

AC → B ?　　　AC → D ?

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

$A \rightarrow B$ $C \rightarrow B$

$D \rightarrow A$ $D \rightarrow B$ $D \rightarrow C$

$AC \rightarrow B$? $AC \rightarrow D$?

- 在上述的FD关系中，我们不用考虑 $AC \rightarrow B$，why ?
- 因此，最后只需要考虑下面的一个FD是否可能成立？

$AC \rightarrow D$ √

6.6 Functional Dependencies

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d2
a2	b1	c1	d3
a2	b1	c3	d4

- 该关系上可能存在的函数依赖：

  $A \rightarrow B$ $C \rightarrow B$

  $D \rightarrow A\ B\ C$

  $AC \rightarrow D$

a) 形如1)和2)这两种情况的函数依赖，都属于是可能成立的，并且可以从已写出的这六个函数依赖中推导出来。

b) 在情况3)中，可以用已写出的这六个函数依赖来证明它是否成立。

- 该关系上可能存在的函数依赖:

$A \rightarrow B$　　　　$C \rightarrow B$　　　　$AC \rightarrow D$

$D \rightarrow A\ B\ C$

- 思考问题：为什么没有写出
  1) 左边含有属性D的其它的那些可能的FD？
  2) 右边为单个属性B的其它的那些可能的FD？
  3) 右边为多个属性的那些可能的FD？

Content of next

- Closure of a Set of FDs (函数依赖集F的闭包)
- FD Set Cover (函数依赖集的覆盖)
- Equivalence of two sets of FDS (函数依赖集的等价)

- Closure of a Set of Attributes (属性集的闭包)
  - Algorithm 6.6.12
- Minimal Cover (最小覆盖)
  - Algorithm 6.6.13

6.6 Functional Dependencies

- **Def. 6.6.9 Closure of a Set of FDs (函数依赖集F的闭包，记为 F^+)**
  - **Given a set F of FDs on attributes of a table T, we define the CLOSURE of F, symbolized by F^+, to be the set of all FDs implied by F.**
    - **There are two FDs (a) and (b) in F. By Armstrong's Axioms and (a) and (b), we can get a new FD (c), then (c) is a FD implied by F.**
    - **If FD (d) is a trivial dependency ($A \rightarrow A$, $B \rightarrow B$, etc.), then (d) is a FD implied by F.**

6.6 Functional Dependencies

- **Example 6.6.5**

  F = { A→B, B→C, C→D, D→E, E→F, F→G, G→H }

  1) all FDs in F are element of F^+
  2) by Armstrong's inclusion rule, A→A, B→B, AB→B ...... are element of F^+
  3) by Armstrong's transitivity rule, A→C, A→D, ... are element of F^+
  4) by Armstrong's augmentation rule, AD→BD, ABD→BCD, ... are element of F^+
  5) by Armstrong's union rule, A→AB, A→BC, A→ABC, ..., A→ABCDEFGH, ... are element of F^+
  6) ......

Another Example

6.6 Functional Dependencies

- **Def. 6.6.10** FD Set Cover **(函数依赖集的覆盖)**
  - A set F of FDs on a table T is said to COVER another set G of FDs on T if the set G can be derived by implication rules from the set F, i.e., if $G \subseteq F^+$.

- 函数依赖集的等价
  - If F covers G and G covers F, we say the two sets of FDs are equivalent, $F \equiv G$.

6.6 Functional Dependencies

- **Example 6.6.6: Consider the two sets of FDs on the set of attributes {A,B,C,D,E}**

  $$F = \{ B \rightarrow CD,\ AD \rightarrow E,\ B \rightarrow A \}$$

  $$G = \{ B \rightarrow CDE,\ B \rightarrow ABC,\ AD \rightarrow E \}$$

  - **F covers G ?**
  - **G covers F ?**

6.6 Functional Dependencies

- **Ex. 6.6.6**

  F: (f_1) $B \rightarrow CD$ (f_2) $AD \rightarrow E$ (f_3) $B \rightarrow A$

  G: (g_1) $B \rightarrow CDE$ (g_2) $B \rightarrow ABC$ (g_3) $AD \rightarrow E$ }

F covers G ?

- **g_1 can be derived from the set F ?**
  1) **by f_1 and f_3 and union rule, we have ① $B \rightarrow ACD$**
  2) **by f_2 and augmentation rule, we have ② $CDAD \rightarrow CDE$, and can be rewritten ③ $ACD \rightarrow CDE$**
  3) **by ① and ③ and transitivity rule, we have g_1**

Ex. 6.6.6

F: (f_1) $B \rightarrow CD$ (f_2) $AD \rightarrow E$ (f_3) $B \rightarrow A$

G: (g_1) $B \rightarrow CDE$ (g_2) $B \rightarrow ABC$ (g_3) $AD \rightarrow E$ }

F covers G ?

- g_2 can be derived from the set F ?

1) by f_1 and decomposition rule, we have ① $B \rightarrow C$
2) by ① and f_3 and union rule, we have ② $B \rightarrow AC$
3) by ② and augmentation rule, we have g_2 $B \rightarrow ABC$

6.6 Functional Dependencies

- **Ex. 6.6.6**

  F: $(f_1)\ B \rightarrow CD$ $(f_2)\ AD \rightarrow E$ $(f_3)\ B \rightarrow A$

  G: $(g_1)\ B \rightarrow CDE$ $(g_2)\ B \rightarrow ABC$ $(g_3)\ AD \rightarrow E$ }

F covers G ?

- g_3 is an element of the set F.

6.6 Functional Dependencies

❑ Ex. 6.6.6

F: (f_1) $B \rightarrow CD$ (f_2) $AD \rightarrow E$ (f_3) $B \rightarrow A$

G: (g_1) $B \rightarrow CDE$ (g_2) $B \rightarrow ABC$ (g_3) $AD \rightarrow E$ }

❑ **G covers F ?**

- f_1 can be derived from the set G ?
  1) by g_1 and decomposition rule, we have f_1 $B \rightarrow CD$
- f_2 is an element of the set G.
- f_3 can be derived from the set G ?
  1) by g_2 and decomposition rule, we have f_3 $B \rightarrow A$

Def. 6.6.11 Closure of a Set of Attributes （属性集的闭包）

- Given a set X of attributes in a table T and a set F of FDs on T, we define the CLOSURE of the set X (under F), denoted by X^+ or X^+_F, as the largest set of attributes Y such that $X \rightarrow Y$ is in F^+.

$$X^+_F = \{ A \mid X \rightarrow A \in F^+ \}$$

algorithm 6.6.12

a) $X^+ := X$;

b) repeat {

$\quad$ $oldX^+ := X^+$;

$\quad$ for each $Y \rightarrow Z$ in F {

$\quad\quad$ if $Y \subseteq X^+$ then $X^+ := X^+ \cup Z$;

$\quad$ }

} until ($oldX^+ = X^+$)

Example 6.6.7: F = { (f$_1$) B→CD,   (f$_2$) AD→E, (f$_3$) B→A }, compute $\{B\}^+$?

- set X = $\{B\}^+$ = { B }
- first loop

1) the left side of f$_1$ is a subset of $\{B\}^+$, then $\{B\}^+ = \{B\}^+$ union {C,D}={B,C,D}

2) the left side of f$_2$ isn’t a subset of $\{B\}^+$, then f$_2$ does not apply at this time

3) the left side of f$_3$ is a subset of $\{B\}^+$, then $\{B\}^+ = \{B\}^+$ union {A} = {A,B,C,D}

4) X $\neq$ $\{B\}^+$, go to step b)

- second loop
  1) $X = \{B\}^+ = \{ A,B,C,D \}$
  2) skip the FDs that have been applied
  3) the left side of f_2 is a subset of $\{B\}^+$, then $\{B\}^+ = \{B\}^+$ union $\{E\} = \{A,B,C,D,E\}$
  4) $X \neq \{B\}^+$, go to step b)
- third loop
  1) $X = \{B\}^+ = \{A,B,C,D,E\}$
  2) loop through FDs in F again
  3) end with $X = \{B\}^+$
- return $\{B\}^+$

❑Def. 6.6.9 Closure of a Set of FDs

$F^+ = \{ X \rightarrow A \mid X \rightarrow A$ can be derived from $F \}$

❑Def. 6.6.10 FD Set Cover

F cover G **iff** " each $X \rightarrow A$ of G can be derived from F " $(G \subseteq F^+)$

❑Def. FD Set Equivalent

F covers G and G covers F

❑Def. 6.6.11 Closure of a Set of Attributes

$X_F^+ = \{ A \mid X \rightarrow A$ can be derived from F $(X \rightarrow A \in F^+) \}$

6.6 Functional Dependencies

- **Algorithm 6.6.13 <u>Minimal Cover</u> (最小覆盖)**
  - **a minimal set M of FDs that covers a given set F of FDs.**

a) 没有冗余(inessential)的函数依赖

b) 每一个函数依赖的左边都没有多余的属性

- **step 1**: From the set F of FDs, we create an equivalent set H of FDs, with only single attributes on the right side.

- **step 2:** From the set H of FDs, successively remove individual FDs that are inessential in H.

- **step 3:** From the set H of FDs, successively replace individual FDs with FDs that have a smaller number of attributes on the left-hand side, as long as the result does not change H^+.

- **step 4:** From the remaining set of FDs, gather all FDs with equal left-hand sides and use the union rule to create an equivalent set of FDs M where all left-hand sides are unique.

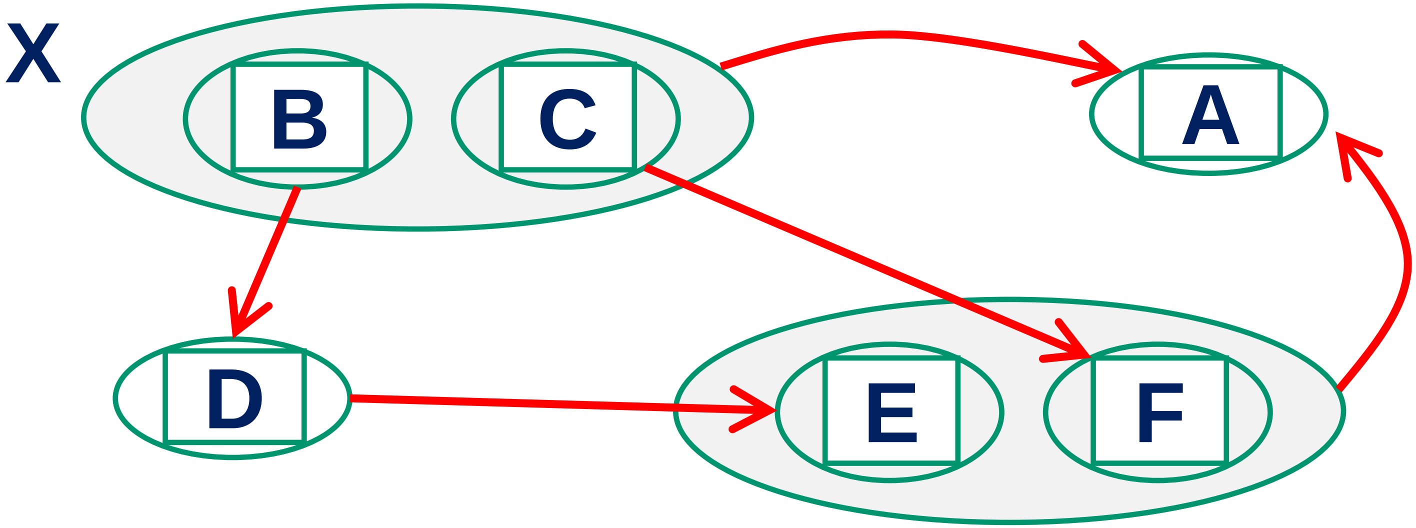


Figure 6.20 Example of an Inessential FD: X→A

- **step 2: From the set H of FDs, successively remove individual FDs that are inessential in H.**
  - **An FD X→Y is inessential in a set H of FDs, if X→Y can be removed from H, with result J, so that $H^+=J^+$, or H=J.**
  - **That is, removal of the FD from H has no effect on H^+.**

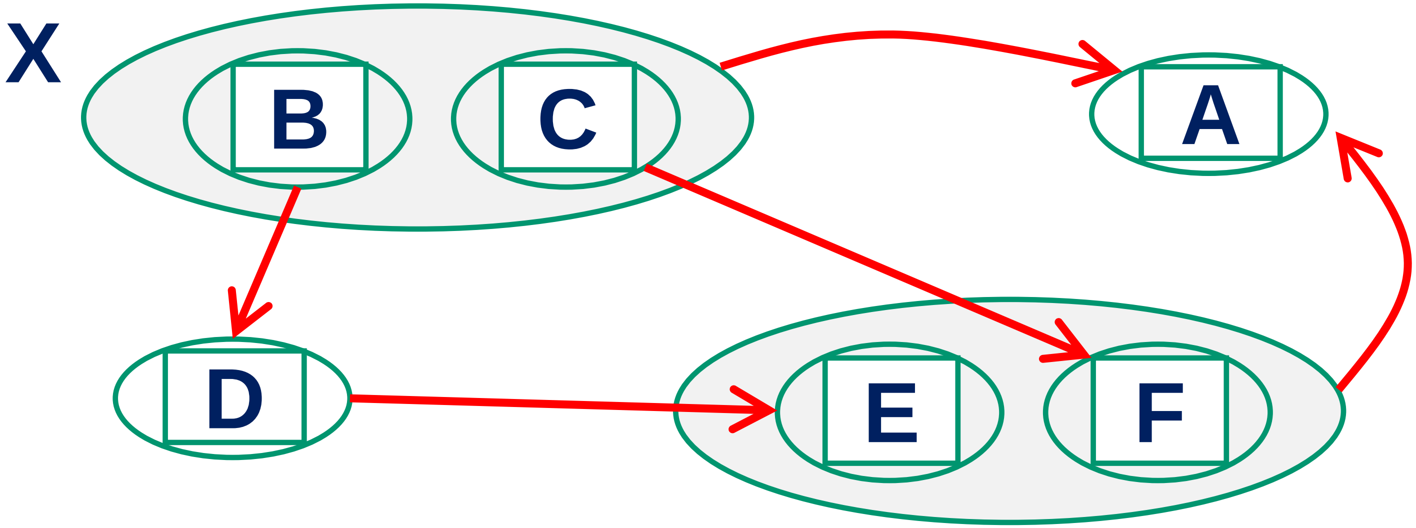


Figure 6.20 Example of an Inessential FD: X→A

- **Let:**

  F = {B→D, D→E, C→F, BC→A, EF→A}

  H = F – {BC→A} = {B→D, D→E, C→F, EF→A}

- **BC→A is inessential in F because of $F^+=H^+$**

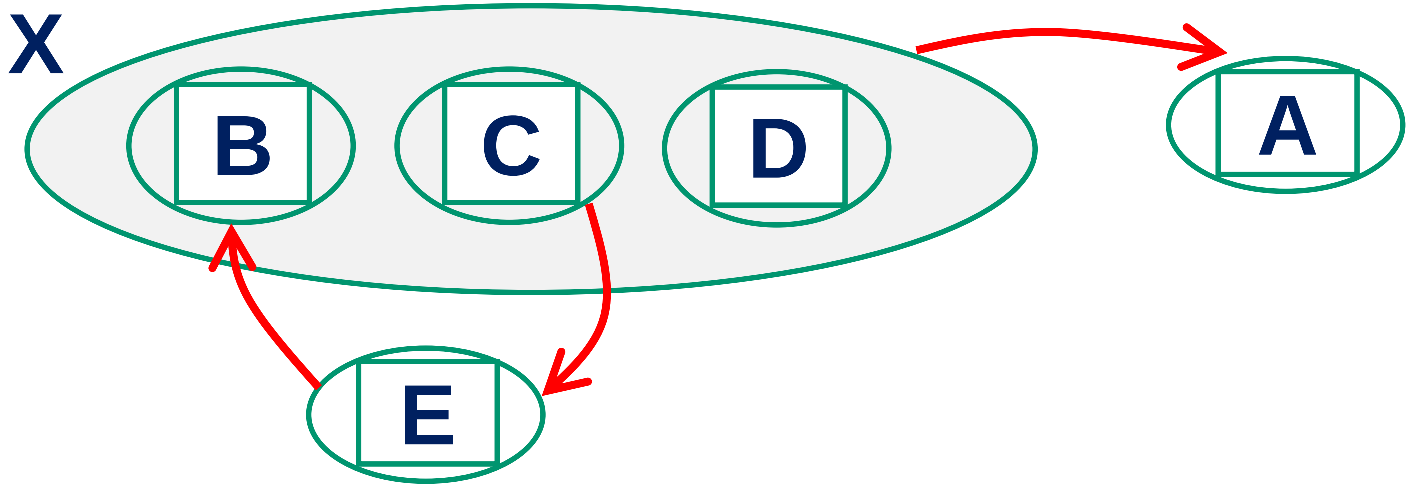


Figure 6.21 Example of an FD: X→A where B can be removed

- **step 3: From the set H of FDs, successively replace individual FDs with FDs that have a smaller number of attributes on the left-hand side, as long as the result does not change H^+.**

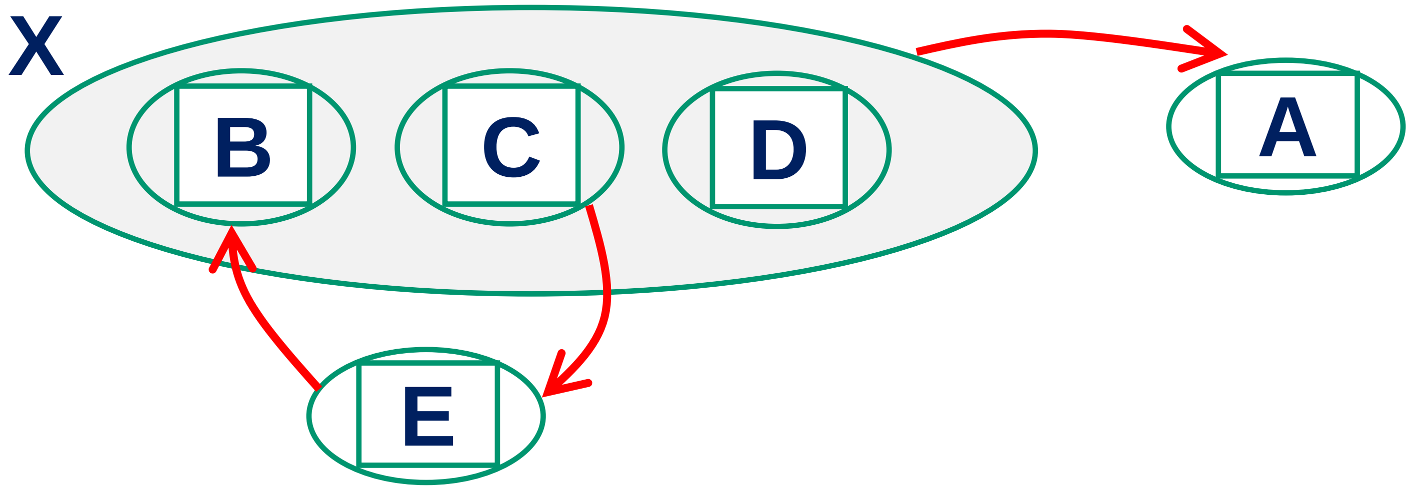


Figure 6.21 Example of an FD: X→A where B can be removed

- **Let: F = {C→E, E→B, BCD→A}**
- **We say B can be removed from BCD→A (*replace BCD→A by CD→A*) because:**
  - **Let: H = { C→E, E→B, CD→A }**
  - **We have: $F^+ = H^+$**

6.6 Functional Dependencies

- **step 4**: From the remaining set of FDs, gather all FDs with equal left-hand sides and use the union rule to create an equivalent set of FDs M where all left-hand sides are unique.

6.6 Functional Dependencies

计算过程

Example 1

- Suppose X = { a, b, c, d } and
  F = { $a \to b$, $b\,c \to d$, $a\,c \to d$ }
- Give the minimal cover M for the set F of FDs.

Example 2

- Suppose Y = { a, b, c, d } and
  G = { $a \to a\,c$, $b \to a\,b\,c$, $d \to a\,b\,c$ }
- Give the minimal cover N for the set G of FDs.

6.6 Functional Dependencies

- **Example 6.6.8 Construct the minimal cover cover M for the set F of FDs.**
  - **F:**

    1) $A\ B\ D \rightarrow A\ C$
    2) $C \rightarrow B\ E$
    3) $A\ D \rightarrow B\ F$
    4) $B \rightarrow E$

计算过程

Content of next

- The process of normalization
  - Decompositions of table

- Lossless Decomposition & Lossy Decomposition (无损分解)
  - Theorem 6.7.3 & 6.7.4

- FD Preserved (依赖保持)

6.7 Lossless Decompositions

- The process of normalization
  - decompose a table into two or more small tables
    - projecting onto two or more subsets of columns that cover all columns and have some columns in common.
  - but it doesn't always work when join back that keep all information of original table.
    - Always get ALL rows back, but
    - might get MORE
      - see example 6.7.1 (pg. 272, next slide)

6.7 Lossless Decompositions

- example 6.7.1

ABC ≠ AB join BC

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c4

AB

A	B
a1	100
a2	200
a3	300
a4	200

BC

B	C
100	c1
200	c2
300	c3
200	c4

AB join BC

A	B	C
a1	100	c1
a2	200	c2
a2	200	c4
a3	300	c3
a4	200	c2
a4	200	c4

6.7 Lossless Decompositions

Def. 6.7.1 Lossless Decomposition (无损性分解)

lossless-join decomposition

For any table T with an associated set of functional dependencies F, *a decomposition* of T into k tables is a set of tables $\{T_1, T_2, ..., T_k\}$, with two properties:

1) **for every table T_i in the set, Head(T_i) is a proper subset of Head(T);**

2) **$Head(T) = Head(T_1) \cup Head(T_2) \cup ...... \cup Head(T_k)$**

6.7 Lossless Decompositions

- **Def. 6.7.1 (cont.)**
  - **Given any specific content of T, the rows of T are projected onto the columns of each T_i as a result of the decomposition.**

  - **A decomposition of a table T with an associated set F of FDs is said to be *a lossless decomposition* if, for any possible future content of T, the FDs in F guarantee that the following relationship will hold:**

$$T \equiv T_1 \text{ join } T_2 \text{ join } \ldots \text{ join } T_k$$

6.7 Lossless Decompositions

- **Lossy Decomposition**
  - a decomposition of T is $\{T_1, T_2, ..., T_k\}$
  - We join the tables of the decomposition, we might get back other rows that were not originally present, so

$$T \subset T_1 \text{ join } T_2 \text{ join } ... \text{ join } T_k$$

Ex 6.7.1 A Lossy Decomposition

(A Loss-Join Decomposition)

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c4

AB

A	B
a1	100
a2	200
a3	300
a4	200

BC

B	C
100	c1
200	c2
300	c3
200	c4

AB join BC

A	B	C
a1	100	c1
a2	200	c2
a2	200	c4
a3	300	c3
a4	200	c2
a4	200	c4

❑Ex 6.7.2 A Different Content for Table ABC

ABC = AB join BC

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3

AB

A	B
a1	100
a2	200
a3	300

BC

B	C
100	c1
200	c2
300	c3

AB join BC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3

6.7 Lossless Decompositions

- **Def. 6.7.2** A database schema is the set of headings of all tables in a database, together with the set of all FDs that the designer wishes to hold on the join of those tables.

- **Ex 6.7.3** Table ABC with a FD: $B \rightarrow C$
  - Assume the table content of ABC is (right).

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3

6.7 Lossless Decompositions

❑ Ex 6.7.3 Table ABC with a FD: B→C

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c4

– If we tried to insert a row (a4, 200, c4), this insert would fail. Why?

6.7 Lossless Decompositions

- **Ex 6.7.3 Table ABC with a FD: B→C**

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c2

– but, we can insert a row (a4, 200, c2) to this table.

❑Ex 6.7.3 Table ABC with a FD: B→C

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c2

AB

A	B
a1	100
a2	200
a3	300
a4	200

BC

B	C
100	c1
200	c2
300	c3

AB join BC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c2

ABC ≡ AB join BC, why?

6.7 Lossless Decompositions

- **Theorem 6.7.4.** Given a table T with a set F of FDs valid on T, then a decomposition of T into two tables $\{T_1, T_2\}$ is *a lossless decomposition* if one of the following functional dependencies is implied by F:
  1) $Head(T_1) \cap Head(T_2) \rightarrow Head(T_1)$
  2) $Head(T_1) \cap Head(T_2) \rightarrow Head(T_2)$

- **PROOF. (pg. 275)**
  – Assume $Head(T) = \{ X, Y, Z \}$, $Head(T_1) = \{ X, Y \}$, $Head(T_2) = \{ Y, Z \}$, then
    $Head(T_1) \cap Head(T_2) = \{ Y \}$

6.7 Lossless Decompositions

- **Ex 6.7.4: In Example 6.7.3**

ABC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c2

AB

A	B
a1	100
a2	200
a3	300
a4	200

BC

B	C
100	c1
200	c2
300	c3

AB join BC

A	B	C
a1	100	c1
a2	200	c2
a3	300	c3
a4	200	c2

6.7 Lossless Decompositions

- **Ex 6.7.5 CUSTORDS**

- **Ex 6.7.6 Lossless Join Decomposition with Multiple Tables: $T \Rightarrow \{ T_1, T_2, ..., T_k \}$**
  - **we can demonstrate losslessness by using the two-table result in a recursive manner.**
    - **$(((T_1 \text{ join } T_2) \text{ join } T_3) \ldots \text{ join } T_k)$**

6.7 Lossless Decompositions

- **Ex.** **Give a decomposition of table T(A,B,C) with a set F of FDs:** $\rho = \{T_1, T_2\}$
  Is it *a lossless decomposition* ?

  1) $F = \{ A \rightarrow B \}$, $T_1(A, B)$, $T_2(A, C)$

  2) $F = \{ A \rightarrow C, B \rightarrow C \}$, $T_1(A, B)$, $T_2(A, C)$

  3) $F = \{ A \rightarrow B \}$, $T_1(A, B)$, $T_2(B, C)$

  4) $F = \{ A \rightarrow B, B \rightarrow C \}$, $T_1(A, C)$, $T_2(B, C)$

6.7 Lossless Decompositions

- **Ex.** Give a decomposition of table T(A,B,C,D) with a set F of FDs {$A \to B$, $B \to C$, $A \to D$, $D \to C$}:

  T_1 (A,B) T_2 (A,C) T_3 (A,D)

  - Is it *a lossless decomposition* ?

– i.e.

- T_1 and T_2 is a lossless decomposition ?
- ($T_1 \infty T_2$) and T_3 is a lossless decomposition ?

6.8 Normal Forms

- **emp_info (emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname, skill_id, skill_name, skill_date, skill_lvl)**

emp_id → {emp_name, emp_phone, dept_name}

dept_name → {dept_phone, dept_mgrname}

skill_id → skill_name

{emp_id, skill_id} → {skill_date, skill_lvl}

Figure 6.22 & 6.23

6.8 Normal Forms

emps(emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname)

emp_id→{emp_name, emp_phone, dept_name}

dept_name→{dept_phone, dept_mgrname}

skills(emp_id,skill_id,skill_name,skill_date,skill_lvl)

skill_id→skill_name

{emp_id, skill_id} → {skill_date, skill_lvl}

Figure 6.24

6.8 Normal Forms

- **Proposition 6.8.1**
  - **The key for the emp_info table is the attribute set (emp_id, skill_id)**
    - **This is also the key for the skills table**
    - **the emps table has a key consisting of the single attribute emp_id**

- **PROOF.**
  - **By Theorem 6.7.3.**

6.8 Normal Forms

- **Proposition 6.8.2**
  - **The factorization of the emp_info table into the emps table and skills table is a true lossless decomposition.**

- **PROOF.**
  - **By the theorem 6.7.4.**

6.8 Normal Forms

- Figure 6.26
  - emps(emp_id, emp_name, emp_phone, dept_name)
  - depts(dept_name, dept_phone, dept_mgrname)
  - emp_skills(emp_id, skill_id, skill_date, skill_lvl)
  - skills(skill_id, skill_name)

- Ex 6.8.2
  - Figure 6.26
  - Figure 6.24
  - Figure 6.25

❑ Def. 6.8.3 FD Preserved (依赖保持性)

- **Given a database schema with a universal table T and a set of functional dependencies F, let $\{T_1, T_2, \ldots\ldots, T_k\}$ be a lossless decomposition of T.**
- **Then an FD $X \rightarrow Y$ of F is said to be *preserved* in the decomposition of T, or alternatively the decomposition of T preserved the FD $X \rightarrow Y$, if for some table T_i of the decomposition, $X \cup Y \subseteq Head(T_i)$.**
- **When this is the case, we also say that the FD $X \rightarrow Y$ is *preserved* in T_i or that it lies in T_i or is in T_i.**

Def. 6.8.3 依赖保持性

- 设关系模式R上的函数依赖集为F，将关系模式R分解为{T_1, T_2, ……,T_k }这k个子关系模式，从函数依赖集F中可以推导出的在子关系模式T_i上所存在的函数依赖集为F_i (i=1,2,…,k)

- 如果函数依赖集F和（$F_1 \cup F_2 \cup \ldots \cup F_k$）是相互等价的，即$F^+ = (F_1 \cup F_2 \cup \ldots \cup F_k)^+$，则我们称该分解是具有依赖保持性的

Content of next

- **Superkey & Key**
  - **Algorithm to Find Candidate Key**
  - **PRIME ATTRIBUTE (主属性)**
  - **NON-PRIME ATTRIBUTE (非主属性)**
- **Normal Forms:**
  - **2NF, 3NF, BCNF**
- **Algorithm 6.8.8**

- **Theorem 6.7.3.** **Given a table T with a set of FDs F and a set of attributes X in Head(T)**

X is a superkey of T iff

X functionally determines all attributes in T ($X \rightarrow Head(T)$ or $X^{+}_{F} = Head(T)$)

6.8 Normal Forms

An Algorithm to Find Candidate Key

- Given a table T with a set F of FDs

```
1.  set K := Head(T) ;
2.  for each attribute A in K
    {
        compute (K – A)_F^+ ;
        if  (K – A)_F^+ contains all the attributes in T, then
        {
            set K := K – { A } ;
        }
    }
```

6.8 Normal Forms

- **Find candidate key for this table R.**

1) R (A, B, C, D),   F: { $B \rightarrow D$, $AB \rightarrow C$ }

2) R (A, B, C),   F: { $A \rightarrow B$, $B \rightarrow A$, $A \rightarrow C$ }

3) R (A, B, C, D),   F: { $A \rightarrow C$, $CD \rightarrow B$ }

6.8 Normal Forms

1) R (A, B, C, D), F: { B→D, AB→C }

解：K = { A, B, C, D }

a) $\because \{K-A\}^+ = \{B,C,D\}^+ = \{B,C,D\} \neq U$

b) $\because \{K-B\}^+ = \{A,C,D\}^+ = \{A,C,D\} \neq U$

c) $\because \{K-C\}^+ = \{A,B,D\}^+ = \{A,B,D, C\} = U$

$\therefore K = K-C = \{A,B,D\}$

d) $\because \{K-D\}^+ = \{A,B\}^+ = \{A,B, D, C\} = U$

$\therefore K = K-D = \{A,B\}$

e) return K.

2) R (A, B, C), F: { A→B, B→A, A→C };

解1： K = { A, B, C }

$\because \{K-A\}^+ = \{B,C,A\} = U \quad \therefore K = K-A = \{B,C\}$

$\because \{K-B\}^+ = \{C\} \neq U$

$\because \{K-C\}^+ = \{B,A,C\} = U \quad \therefore K = K-C = \{B\}$

return { B }

2) R (A, B, C), F: { A→B, B→A, A→C };

解2：K = { A, B, C }

$\because \{K-B\}^+ = \{A,C,B\} = U \quad \therefore K = K-B = \{A,C\}$

$\because \{K-A\}^+ = \{C\} \neq U$

$\because \{K-C\}^+ = \{A,B,C\} = U \quad \therefore K = K-C = \{A\}$

return {A}

6.8 Normal Forms

- **Def. 6.8.5 A PRIME ATTRIBUTE**
  - **A prime attribute of a table T is any attribute that is part of a key for that table**
    - **not necessarily a primary key**

- **Def. A NON-PRIME ATTRIBUTE**

6.8 Normal Forms

Schema	Key	Prime attributes	Non-prime attributes	BCNF?
R (A, B, C, D) { B→D, AB→C }	?	?	?	?
R (A, B, C) { A→B, B→A, A→C }	?	?	?	?
R (A, B, C, D) { A→C, CD→B }	?	?	?	?

6.8 Normal Forms

Schema	Key	Prime attributes	Non-prime attributes	BCNF?
R (A, B, C, D) { B→D, AB→C }	AB	?	?	
R (A, B, C) { A→B, B→A, A→C }	A B	?	?	
R (A, B, C, D) { A→C, CD→B }	AD	?	?	

6.8 Normal Forms

Schema	Key	Prime attributes	Non-prime attributes	BCNF?
R (A, B, C, D) { B→D, AB→C }	AB	A, B	C, D	?
R (A, B, C) { A→B, B→A, A→C }	A B	A, B	C	?
R (A, B, C, D) { A→C, CD→B }	AD	A, D	B, C	?

6.8 Normal Forms

Schema	Key	Prime attributes	Non-prime attributes	BCNF?
R (A, B, C, D) { B→D, AB→C }	AB	A, B	C, D	No
R (A, B, C) { A→B, B→A, A→C }	A B	A, B	C	Yes
R (A, B, C, D) { A→C, CD→B }	AD	A, D	B, C	No

6.8 Normal Forms

- **Normal Forms:**
  - **BCNF**
  - **2NF**
  - **3NF**

- **Algorithm 6.8.8**

6.8 Normal Forms

- **Def. 6.8.4. Boyce-Codd Normal Form (BCNF)**
  - **A table T in a database schema with FD set F is in BCNF iff**
    - **for any FD $X \rightarrow A$ in F^+ that lies in T *(all attributes of X and A in T)*, A is a single attribute not in X, then**

      X must be a superkey for T

6.8 Normal Forms

- **emps(emp_id,emp_name,emp_phone,dept_name)**

  emp_id → {emp_name, emp_phone, dept_name}

- **depts(dept_name, dept_phone, dept_mgrname)**

  dept_name → {dept_phone, dept_mgrname}

- **emp_skills(emp_id, skill_id, skill_date, skill_lvl)**

  {emp_id, skill_id} → {skill_date, skill_lvl}

- **skills(skill_id, skill_name)**

  skill_id → skill_name

Figure 6.26

❑ Def. 6.8.6. Third Normal Form (3NF).

- A table T in a database schema with FD set F is in 3NF iff,
  - for any FD $X \rightarrow A$ implied by F that lies in T, if A is a single non-prime attribute not in X, then X must be a superkey for T.

- **BCNF和3NF定义的对比**
- **BCNF**
  - for any FD $X \rightarrow A$ in F^+ that lies in T (all attributes of X and A in T), A is <u>a single attribute</u> not in X, then X must be a superkey for T
- <u>**3NF**</u>
  - for any FD $X \rightarrow A$ implied by F that lies in T, if A is <u>a single non-prime attribute</u> not in X, then X must be a superkey for T.

6.8 Normal Forms

- **Example 6.8.6 (Figure 6.29)**
  - Each of the tables in this schema is in BCNF, and therefore in 3NF.

- **Example 6.8.7 (Figure 6.28)**
  - This table is in 3NF but not in BCNF.

Theorem: If table T is BCNF, then T is 3NF.

6.8 Normal Forms

- **Example 6.8.8** Head(T) = {A B C D}, and FD set F as follows: F = {$AB \rightarrow CD$, $D \rightarrow B$}

1) Find candidate key for table T
2) Table T is 3NF ?
3) Table T is BCNF ?

- **Def. 6.8.7. Second Normal Form (2NF)**
  - **A table T with FD set F is in 2NF iff:**
    - **for any X $\rightarrow$ A implied by F that lies in T, where A is a single non-prime attribute not in X, then X is not properly contained in any key of T.**

- **Ex 6.8.9**
  - **Figure 6.25**

3NF和2NF定义的对比

3NF

- for any X→A implied by F that lies in T, if A is a single non-prime attribute not in X, then X must be a superkey for T.

2NF

- for any X → A implied by F that lies in T, if A is a single non-prime attribute not in X, then X is not properly contained in any key of T.

6.8 Normal Forms

- **emps(emp_id,emp_name,emp_phone,dept_name, dept_phone, dept_mgrname)**

  emp_id → {emp_name, emp_phone, dept_name}

  dept_name → {dept_phone, dept_mgrname}

- **emp_skills(emp_id, skill_id, skill_date, skill_lvl)**

  {emp_id, skill_id} → {skill_date, skill_lvl}

- **skills(skill_id, skill_name)**

  skill_id → skill_name

Figure 6.25

Example of Normal Forms

- **Relations**

  emp_info(emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname, skill_id, skill_name, skill_date, skill_lvl)

- **Functionally Dependents**

  emp_id→{emp_name, emp_phone, dept_name}

  dept_name→{dept_phone, dept_mgrname}

  skill_id→skill_name

  {emp_id, skill_id}→{skill_date, skill_lvl}

Is 2NF ?

- **Relations**

  emp_info (emp_id, emp_name, emp_phone,
  dept_name, dept_phone, dept_mgrname,
  skill_id, skill_name, skill_date, skill_lvl)

- **Functionally Dependents**

  emp_id $\rightarrow$ {emp_name, emp_phone, dept_name}

  dept_name $\rightarrow$ {dept_phone, dept_mgrname}

  skill_id $\rightarrow$ skill_name

  {emp_id, skill_id} $\rightarrow$ {skill_date, skill_lvl}

Figure 6.23

Figure 6.24 Is 2NF ?

- **Relations 1**

  <u>**Emps**</u> **(emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname)**

- **Functionally Dependents in Relations 1**

  emp_id → {emp_name, emp_phone, dept_name}

  dept_name → {dept_phone, dept_mgrname}

- **Relations 2**

  <u>**Skills**</u> **(emp_id, skill_id, skill_name, skill_date, skill_lvl)**

- **Functionally Dependents in Relations 2**

  skill_id → skill_name

  {emp_id, skill_id} → {skill_date, skill_lvl}

Figure 6.25 Is 2NF ?

- **Relations 1**

Emps (emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname)

emp_id → {emp_name, emp_phone, dept_name}

dept_name → {dept_phone, dept_mgrname}

- **Relations 2**

Emp_Skills (emp_id, skill_id, skill_date, skill_lvl)

{emp_id, skill_id} → {skill_date, skill_lvl}

- **Relations 3**

Skills (skill_id, skill_name)

skill_id → skill_name

Figure 6.25 Is 3NF ?

- **Relations 1**

  <u>Emps</u> (emp_id, emp_name, emp_phone,
  dept_name, dept_phone, dept_mgrname)

  emp_id → {emp_name, emp_phone, dept_name}

  dept_name → {dept_phone, dept_mgrname}

- **Relations 2**

  <u>Emp_Skills</u> (emp_id, skill_id, skill_date, skill_lvl)

  {emp_id, skill_id} → {skill_date, skill_lvl}

- **Relations 3**

  <u>Skills</u> (skill_id, skill_name)

  skill_id → skill_name

Figure 6.26 Is 3NF ?

- **emps(emp_id, emp_name, emp_phone, dept_name)**

  emp_id → {emp_name, emp_phone, dept_name}

- **depts(dept_name, dept_phone, dept_mgrname)**

  dept_name → {dept_phone, dept_mgrname}

- **emp_skills(emp_id, skill_id, skill_date, skill_lvl)**

  {emp_id, skill_id} → {skill_date, skill_lvl}

- **skills(skill_id, skill_name)**

  skill_id → skill_name

Figure 6.26 Is BCNF ?

- **emps(emp_id, emp_name, emp_phone, dept_name)**

  emp_id → {emp_name, emp_phone, dept_name}

- **depts(dept_name, dept_phone, dept_mgrname)**

  dept_name → {dept_phone, dept_mgrname}

- **emp_skills(emp_id, skill_id, skill_date, skill_lvl)**

  {emp_id, skill_id} → {skill_date, skill_lvl}

- **skills(skill_id, skill_name)**

  skill_id → skill_name

- **Algorithm 6.8.8: An Algorithm to Achieve Well-Behaved 3NF Decomposition**
  - This algorithm, given a universal table T and set F of FDs, generates a lossless join decomposition of T that is in 3NF and preserves all FDs of F.
  - The output is a set S of headings (sets of attributes) for tables in the final database schema.

6.8 Normal Forms

- **Algorithm 6.8.8**
  1. replace F with minimal cover of F;
  2. $S = \Phi$;
  3. for all $X \rightarrow Y$ in F

     if, for all $Z \in S$, $X \cup Y \not\subseteq Z$

     then $S = S \cup \text{Heading}(X \cup Y)$

     end for
  4. If, for all candidate keys K for T:

     for all $Z \in S$, $K \not\subseteq Z$

     then choose a candidate key K and set $S = S \cup \text{Heading}(K)$

Example of Normal Forms

- **Relations & Functionally Dependents**

  emp_info(emp_id, emp_name, emp_phone, dept_name, dept_phone, dept_mgrname, skill_id, skill_name, skill_date, skill_lvl)

{

emp_id→{emp_name, emp_phone, dept_name}

dept_name→{dept_phone, dept_mgrname}

skill_id→skill_name

{emp_id, skill_id}→{skill_date, skill_lvl}

}

Figure 6.26 3NF

❑ Relations & Functionally Dependents

Emps(emp_id,emp_name,emp_phone,dept_name)

emp_id $\rightarrow$ {emp_name, emp_phone, dept_name}

Depts (dept_name, dept_phone, dept_mgrname)

dept_name $\rightarrow$ {dept_phone, dept_mgrname}

Emp_Skills (emp_id, skill_id, skill_date, skill_lvl)

{emp_id, skill_id} $\rightarrow$ {skill_date, skill_lvl}

Skills (skill_id, skill_name)

skill_id $\rightarrow$ skill_name