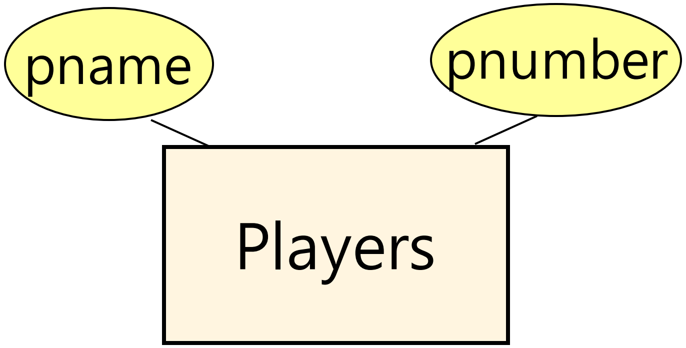
Weak Entity Types

- In real world, some entity type may not have its key.
- An entity type that does not have a key, is called a **weak** entity type.
- To identify weak entities uniquely, we must find its **owner (= strong)** entity type.
- Owner entity type has a weak relationship with weak entity type;
- Owner has always its own key.

Weak Entity Types : Example

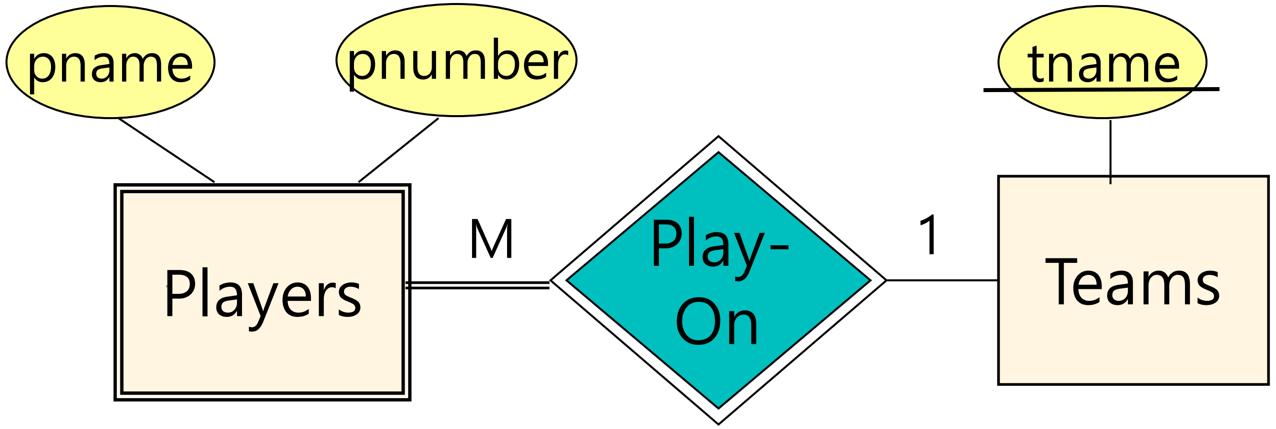


- 'pname' is almost a key for players, but there may be two with the same name.

- 'pnumber' is certainly a key within a same team. But, players on two different teams could have the same number.

- How to identify players uniquely?
  : Player들과 relationship을 갖는 Team들을 찾아 냄.
  이들 team들은 key가 존재하고 이를 owner라 함.

Weak Entity Types : Example



- "Players" (by double box) is a **weak** entity type.
- "Teams" is an **owner** (= identifying) entity type.
  : It has its own key "(team) name"
- "Plays-On" (by double diamond) is a **weak** relationship.

Weak Entity Type : Properties

(1) Weak entity type은 Owner와 **M : 1 (1 : 1)** 관계;

(2) Weak entity type은 항상 **total** participation;

(3) Weak entity type의 Key는?

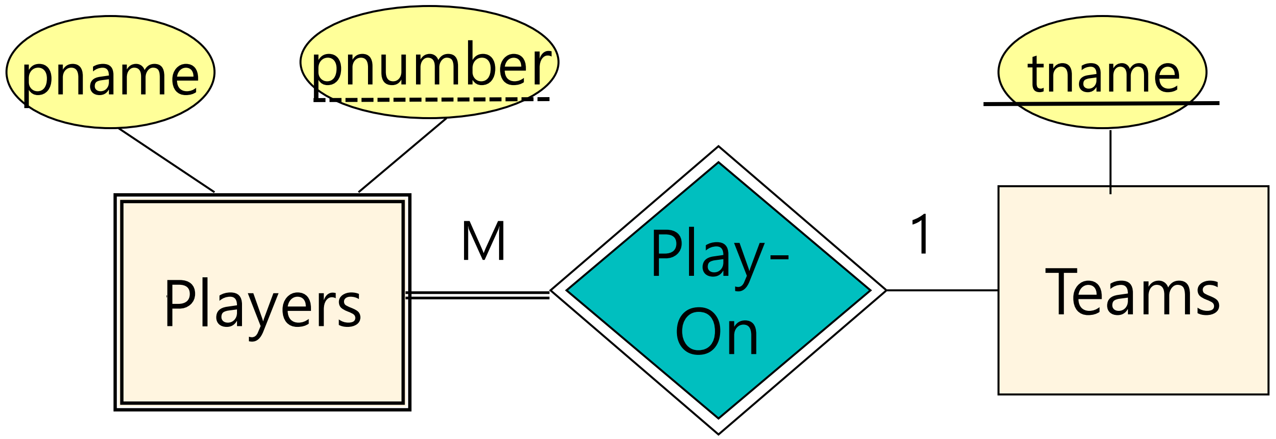
Key of **Owner** + **Partial Key** of **Weak entity type**

(**Partial key**는 owner key의 도움을 받아 weak entity들을 식별할 수 있는 일종의 부분 key를 의미)

(4) **Existence Dependency**

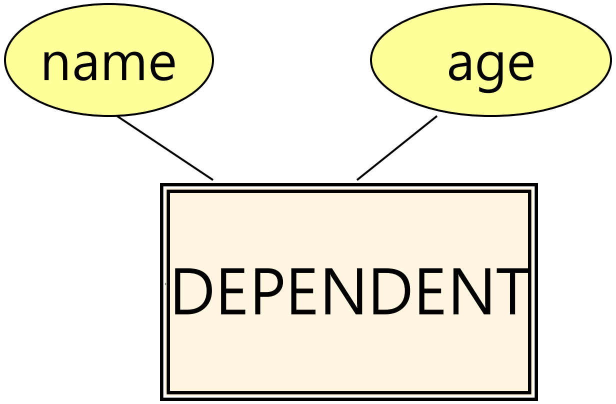
(Weak entity의 존재는 owner에 종속됨. 만약 어떤 owner entity가 DB에서 삭제되면, 이와 relationship 을 갖는 weak entity들 모두 역시 삭제되어야 함)

Weak Entity Types : Example



- “Teams” 의 key는?
- “Players” 의 partial key는?
- “Players” 의 key는?
- 만약 어떤 team이 해체 (즉 DB에서 삭제)된다면?

Weak Entity Types : Exercise



- 어느 회사 사원들의 가족(DEPENDENT)들임.
- 이들의 key를 찾을 수 없음.
- 위의 ER Diagram을 완성시켜라.

(Owner Entity type? 즉, 이 가족들을 부양하는 사원(EMPLOYEE))

Attributes on Relationship

- Sometimes, it is useful to attach an attribute to a relationship; Thus, a relationship can also have its own attributes;

- 일반적으로 m : n 관계를 갖는 relationship에서 요구됨.

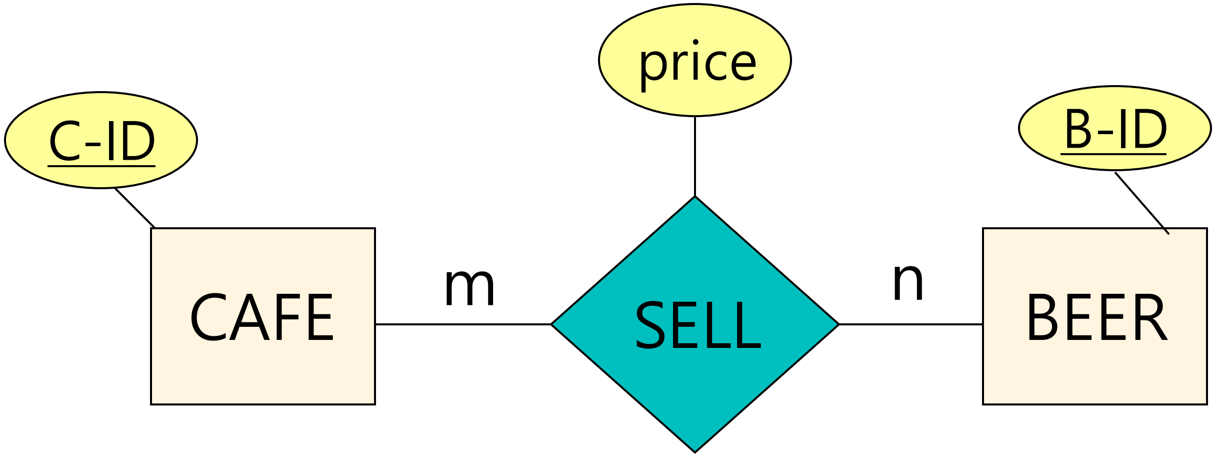
- 1 : n 혹은 1 : 1 relationship에서는 특별한 주의 요함.

- 1 : 1인 경우, relationship의 attribute를 양쪽 entity type들 중 어떤 쪽으로도 이동해도 상관없음;

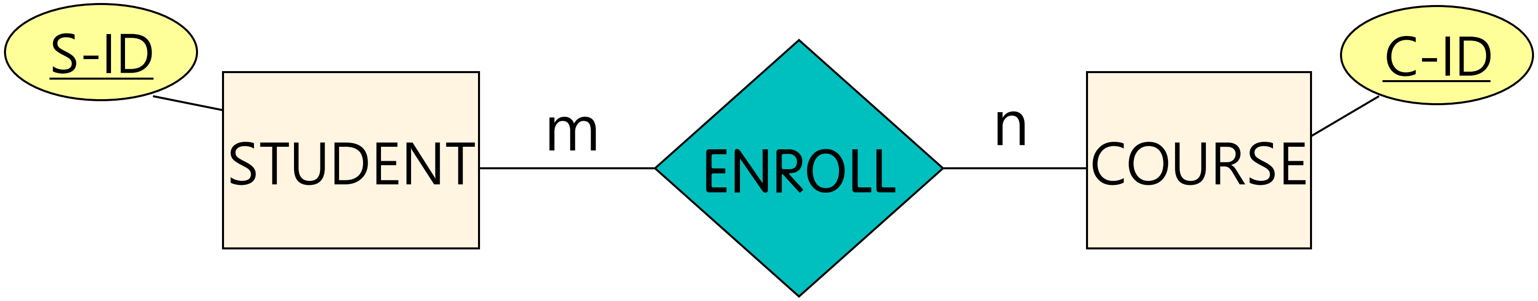
- 반면에 m : 1 인 경우에는 반드시 m-side의 entity type으로만 이동함.

Attributes on Relationship : Example

- 'Price' attribute is a function value of both the cafe and the beer, not of one alone.



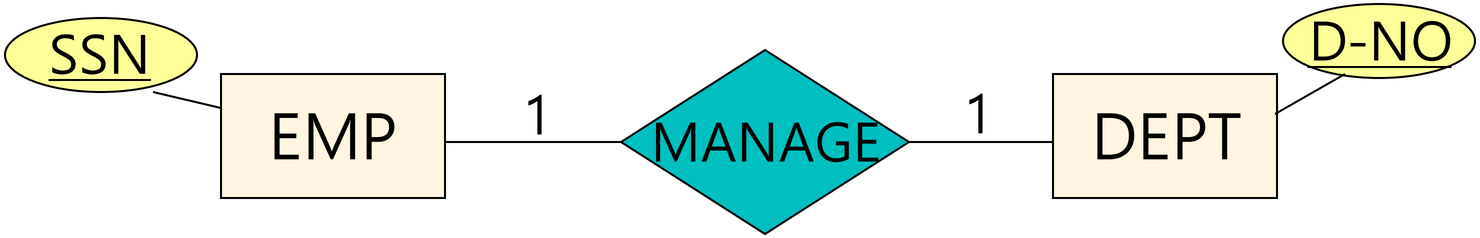
- We want add "grade" attribute; Where to attach?



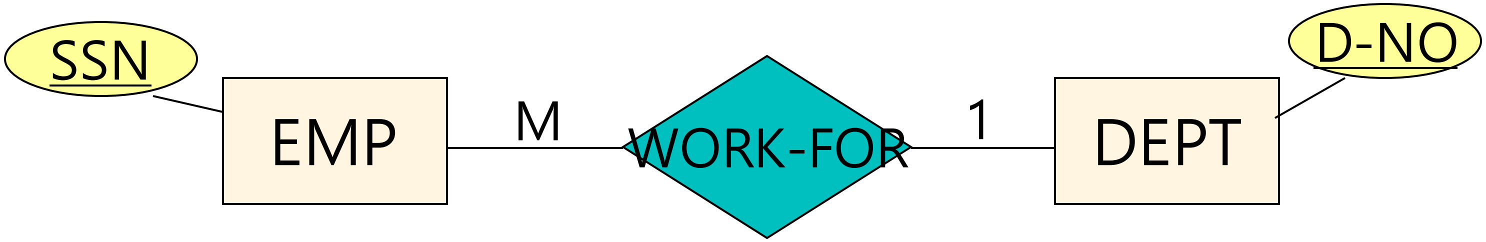
Attributes on Relationship : Example

- We want add “start-date” attribute;

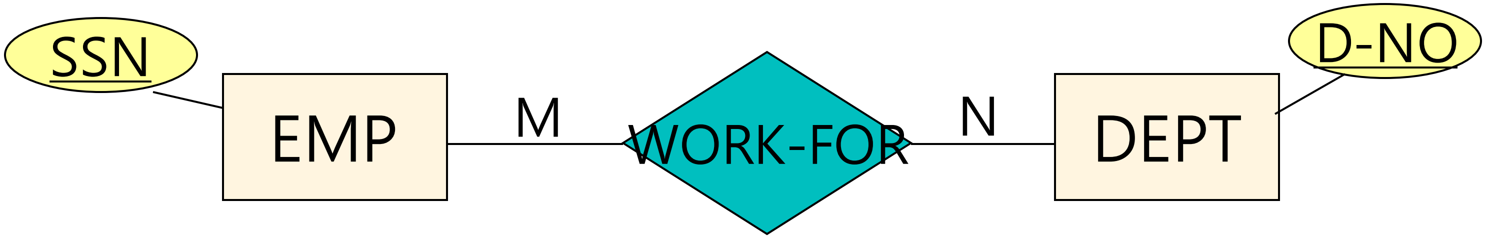
a)



b)



c)



Ternary Relationships

- Sometimes, we need a relationship that connects more than two entity types. (for example, ternary!)

- Consider the following requirements;

  (1) Entities : employees, beers, cafes

  (2) Relationship :

  "Employees only drink certain beers at certain cafes."

  For example;

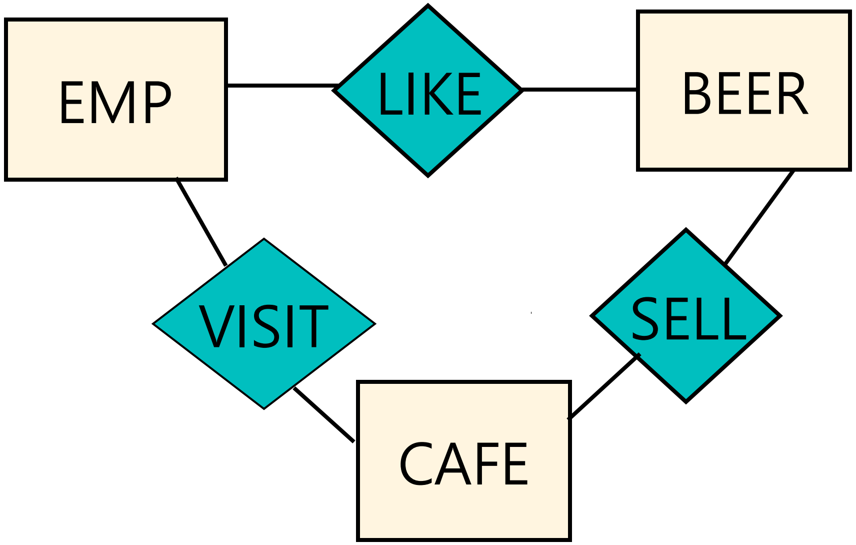
  - 'Joe' only drinks 'Bud' at 'Cheers'.
  - 'Bob' only drinks 'Guiness' at 'Warbar'.
  - 'Joe' only drinks 'IPA' at 'Cheers'.

  . . . .

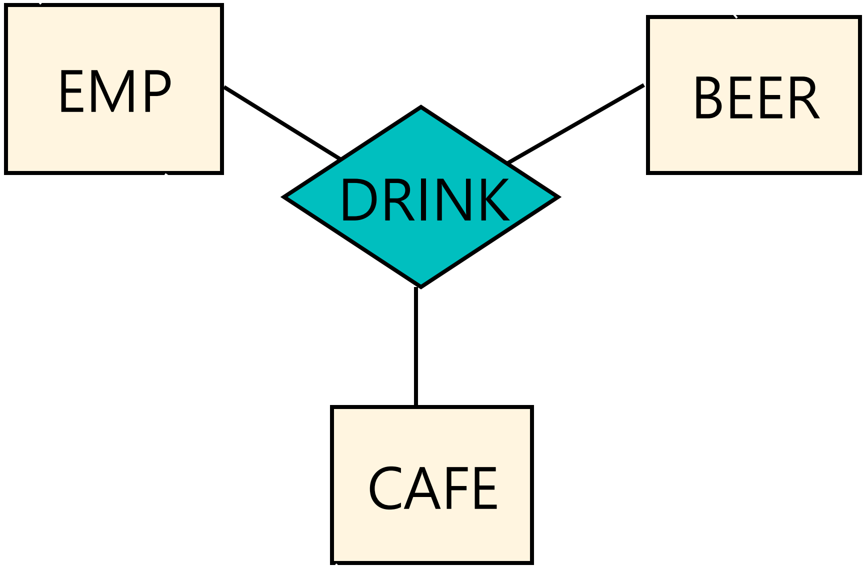
- 위의 요구사항을 정확히 ER model로 표현하라.

Two Possible ER Modeling

(a) Three Binary relationship

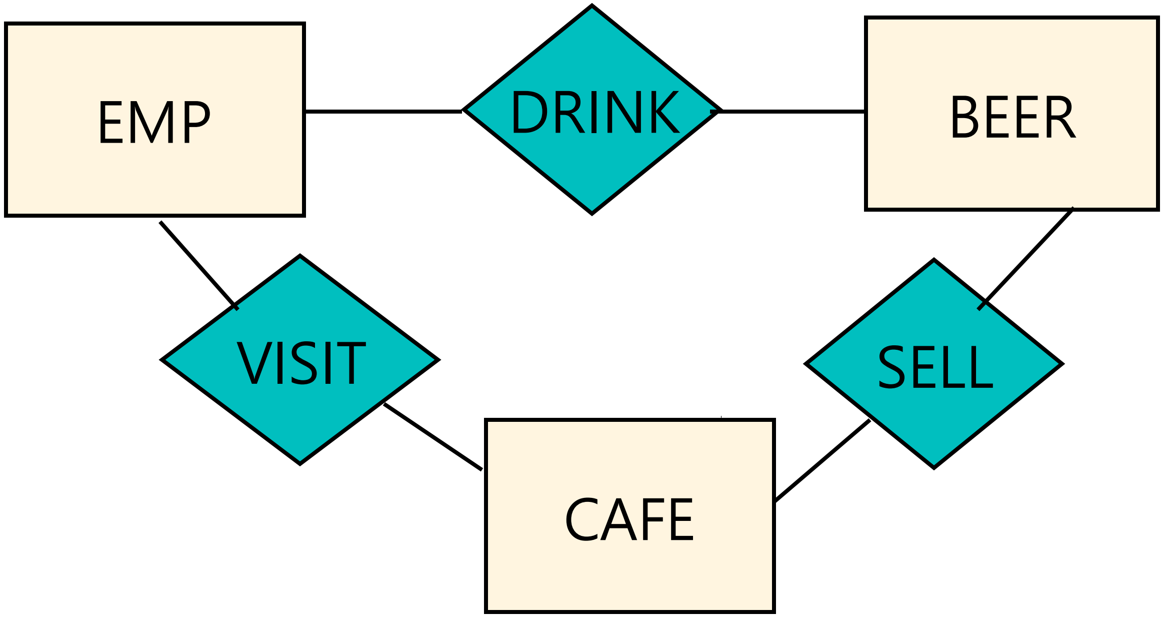


(b) Ternary relationship



- Which one seems correct? Answer will be (b)

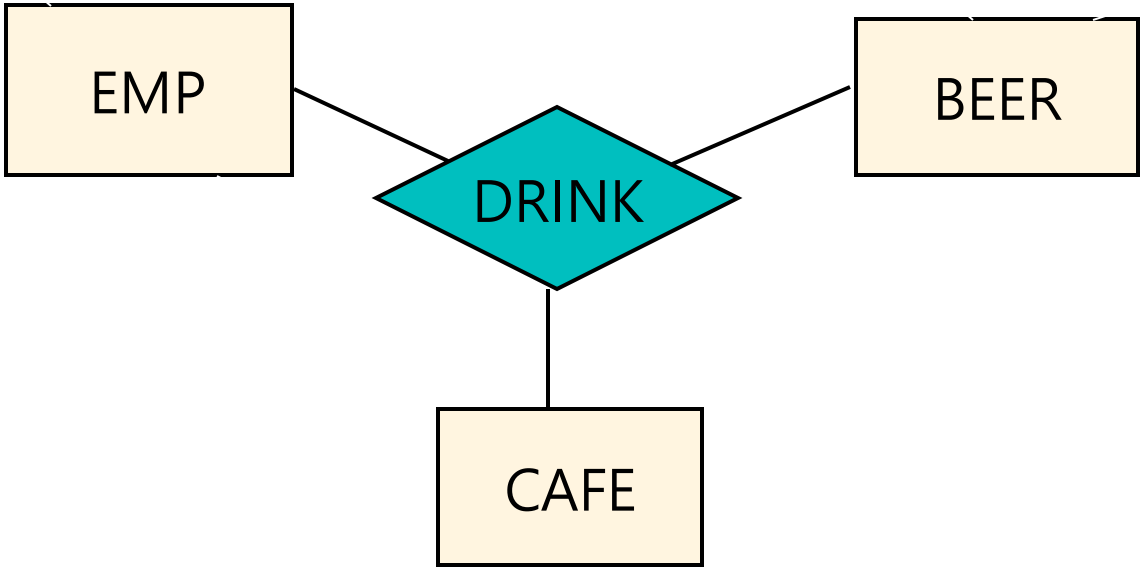
(a) Three Binary Relationships



- Employees DRINK certain beers (but which cafe?) : (e, b, ?)
- Cafes SELL certain beers (but to whom?) : (c, b, ?)
- Employees VISIT certain cafes (but which beer) : (e, c, ?)

- 3 개의 Binary 관계로는 요구사항을 정확히 표현하지 못 함.

(b) Ternary Relationship



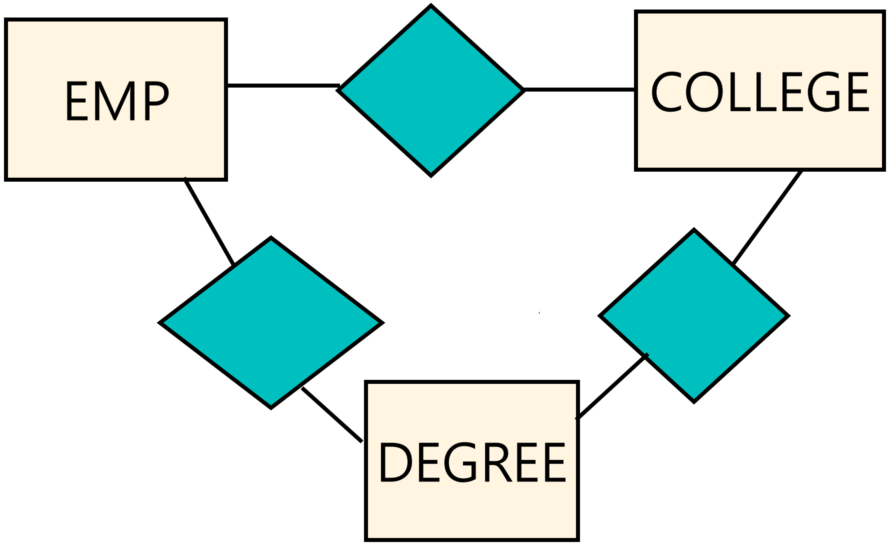
- Employees only drink certain beers at certain cafes : (e, b, c) (e2, b1, c2), (e3, b2, c1), (e5, b4, c5), . .

- In general, from 3 binary relationships (c, b), (e, b), and (e, c)

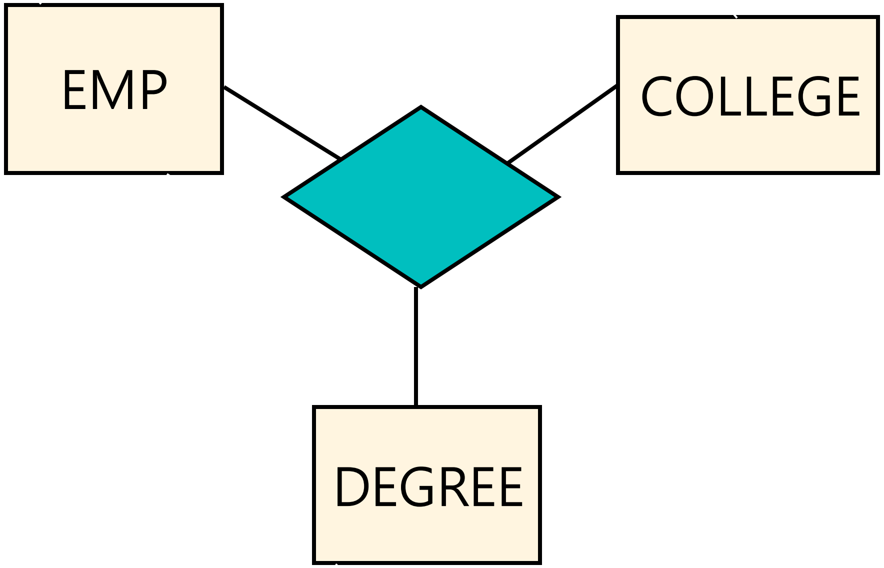
we can not infer a ternary relationship (e, b, c), but reverse is true.

Another Case

(a) Three Binary relationship



(b)Ternary relationship



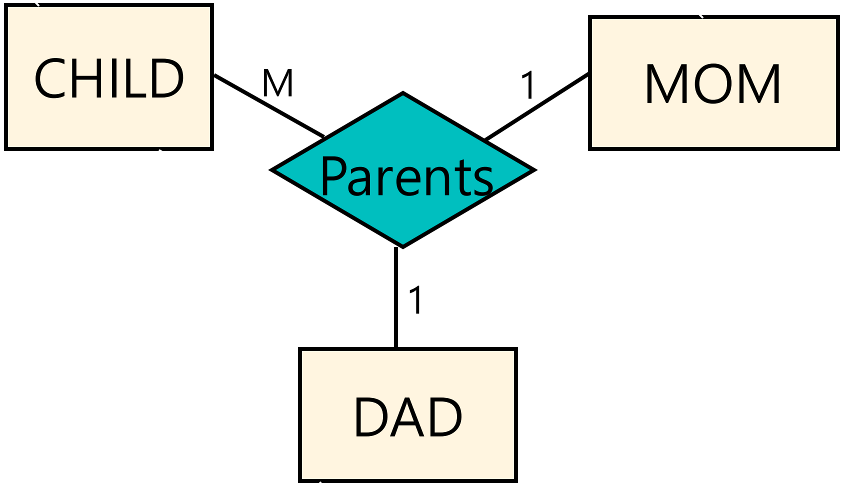
- Are they both represent the same information?

Binary vs. Ternary Relationships

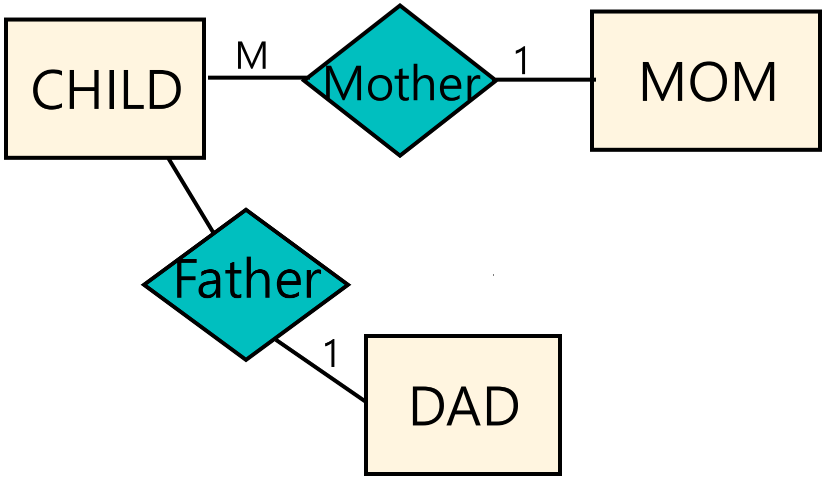
- Some database design does not permit ternary (or more) relationships.
  - Ternary relationship can be represented by several binary relationships;
  - It can be also represented as a weak entity type
  - Each entity in ternary relationship is identified by 3 owners with no partial key
- Every n-ary relationship can be represented by binary relationships.

Convert Ternary into Binary

(A) Ternary relationship



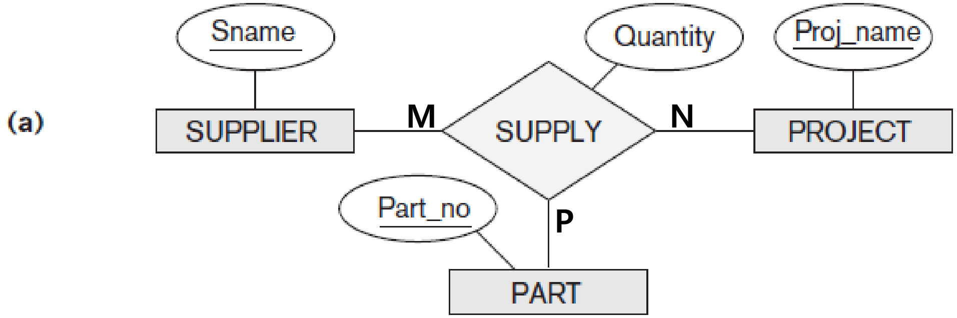
(B) Two Binary relationships



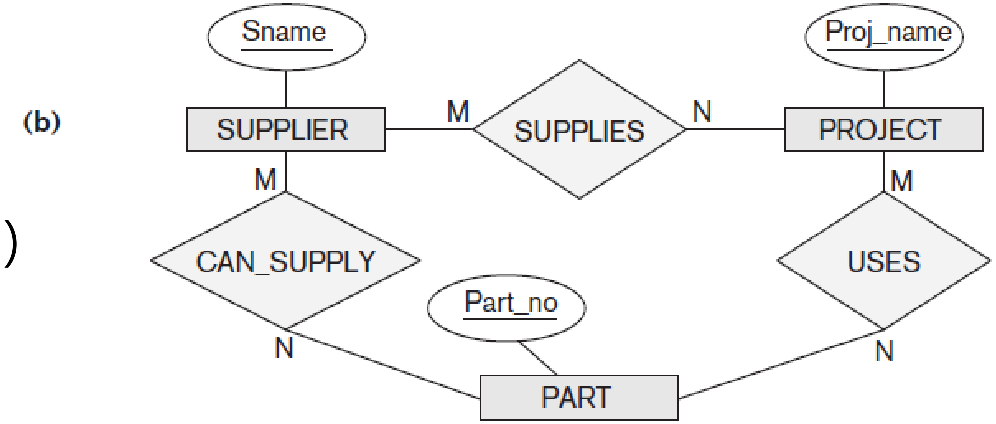
- A ternary relationship “Parents”, relating a child to his/her mom and dad, is naturally replaced by two binary relationships, “Father” and “Mother”.

- Note that both (A) and (B) represent the same information.

(a) Ternary relationship :



(b) Binary relationships :

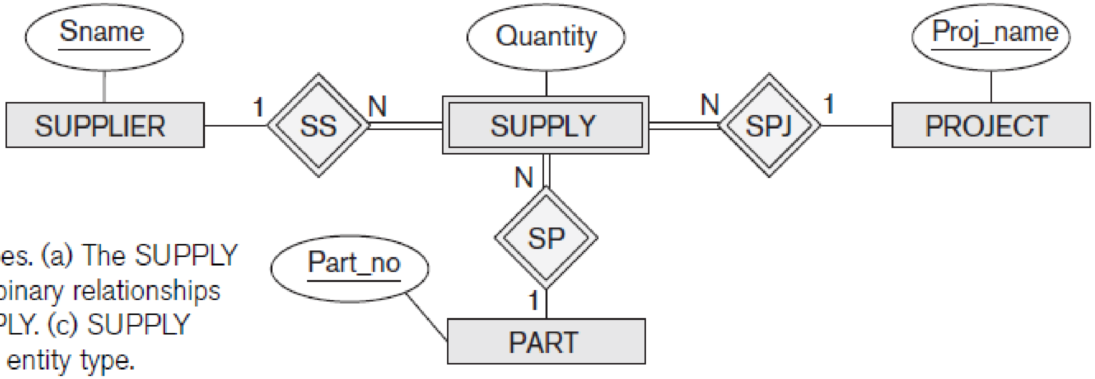


Note: (b) is not equal to (a)

(c) Binary relationships by using weak entity type :

(c)

- SUPPLY의 key = {Sname, Proj-name, Part-no}

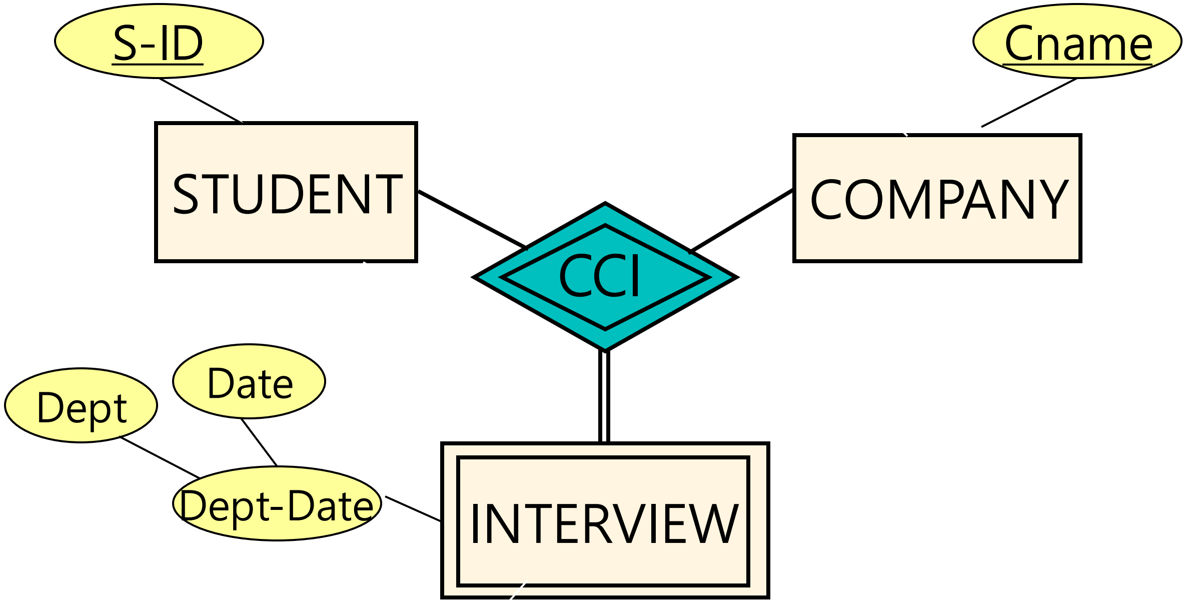


Note: (c) is equal to (a)

Figure 7.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Weak Entity with Ternary Relationship

- It is possible to have a weak entity type with a ternary relationship.



Weak entity type "INTERVIEW" with ternary relationship

- A student can have multiple interviews with the same company; For example, with different company departments, or separate dates.
- Here, "INTERVIEW" is represented as weak entity type; It has two owners; "STUDENT" and "COMPANY" with partial key "Dept-Date".
- "INTERVIEW" 의 key = {S-ID, Cname, {Dept, Date}}

A Few ER Design Guidelines

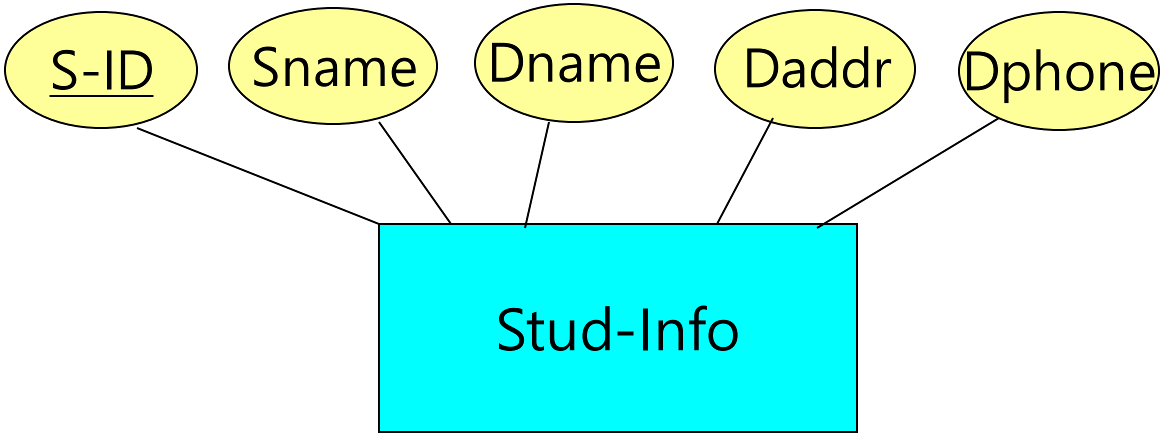
- Avoid Redundancy; Redundancy wastes space and occurs inconsistency.
  - Repeats of the same information may become inconsistent if we change one and forget to change the other.

- Do not use an entity type when an attribute will do; Entity type must satisfy at least one of the following conditions:
  - It is more than the name of something; it has at least one non-key attribute.

    or

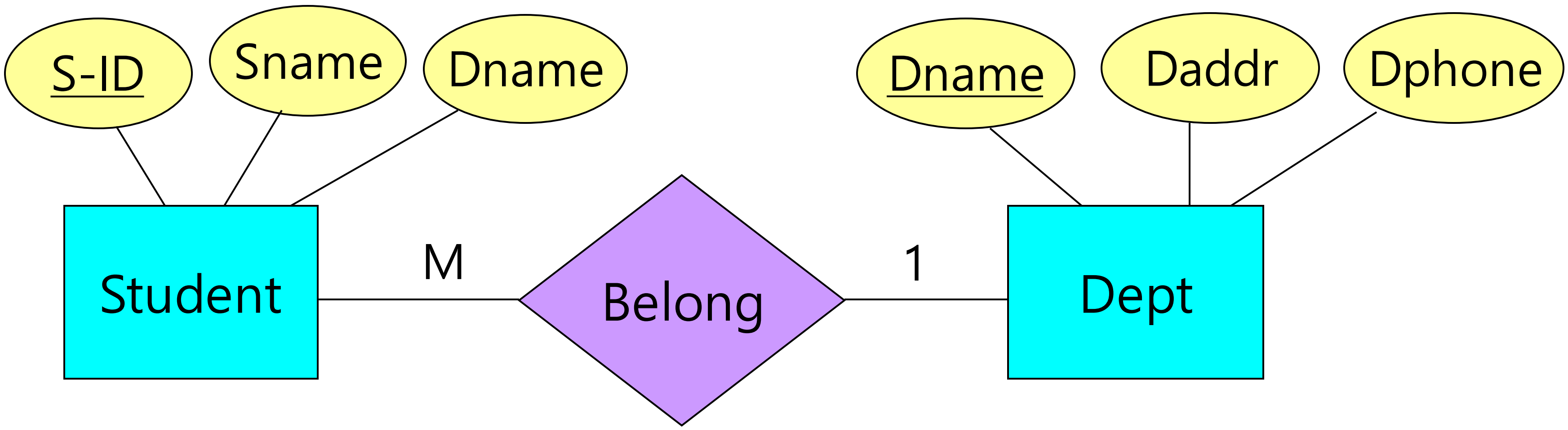
  - It is the "Many" side in a M : 1 relationship.

Redundancy : Very Bad Design



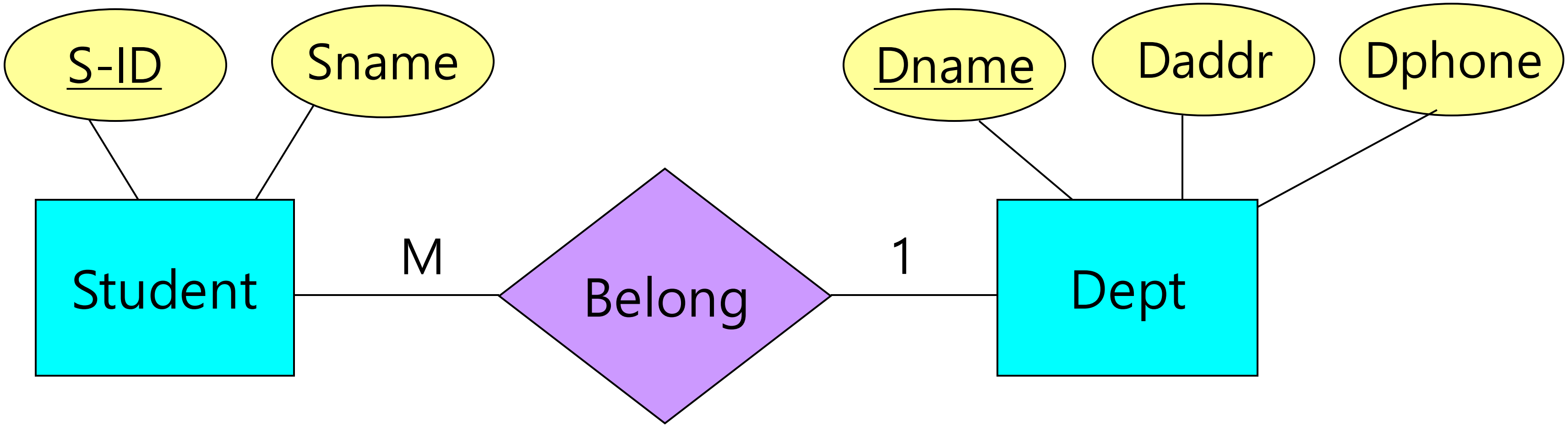
- The same department's information (i.e., Daddr, Dphone) is repeated many times.
- If we want to update some department's information?
- If we want to delete some department's information?
- If we want to add the new department's information later?

Redundancy : Bad Design



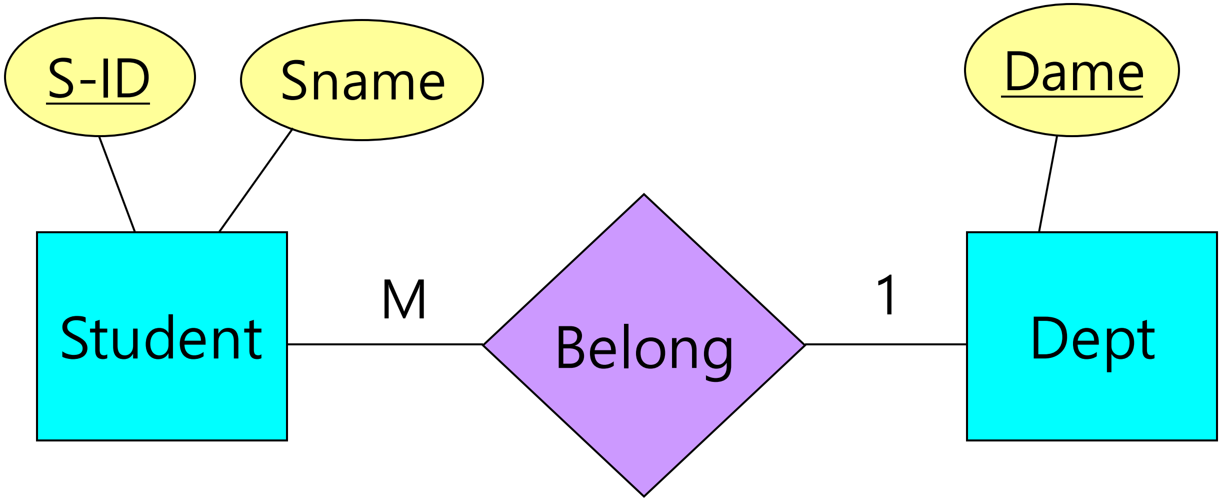
- Better! But this design still repeats the Dname of a department twice: as an attribute and as a related entity.

No Redundancy : Good Design



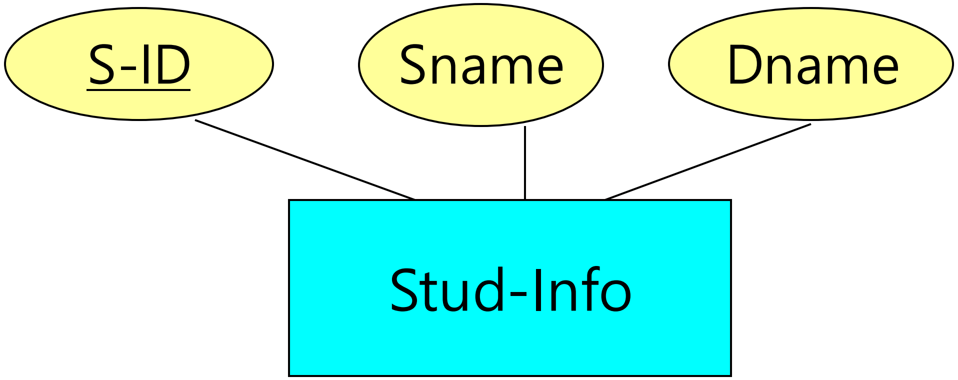
- This design represents the information of each department only once.
- If we want to update some department's information?
- If we want to delete some department's information?
- If we want to add the new department's information later?

Entity vs Attribute : Bad



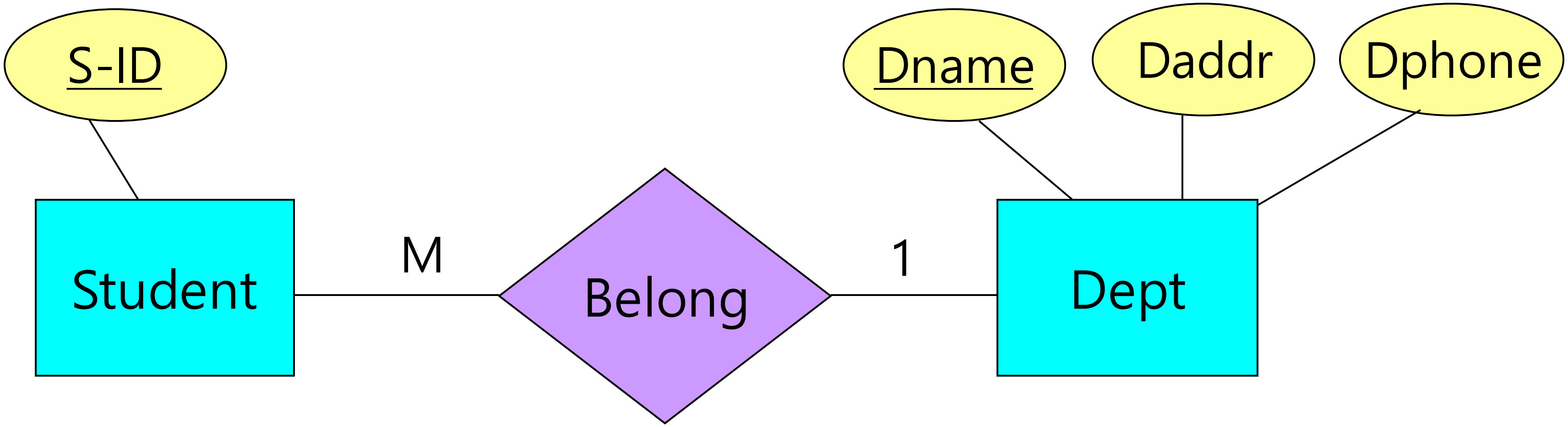
- The department is nothing but a name, and is not at the "Many" side of this relationship, it should not be an entity type.

Entity vs Attribute : Good



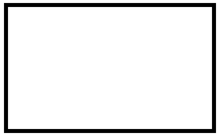
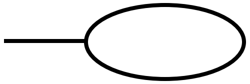
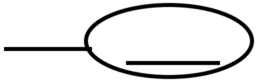
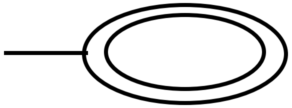
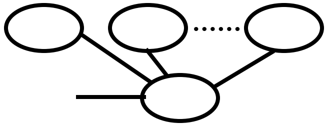
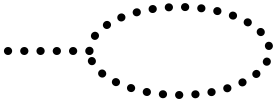
- There is no need to make the department an entity type, because we record nothing about departments besides their name.

Entity vs Attribute : Good

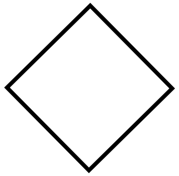
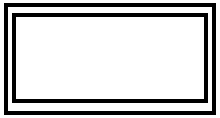
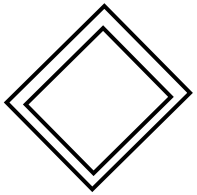
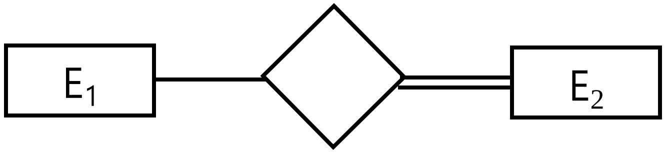
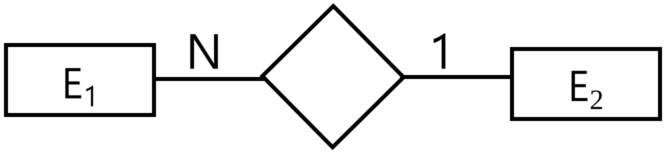
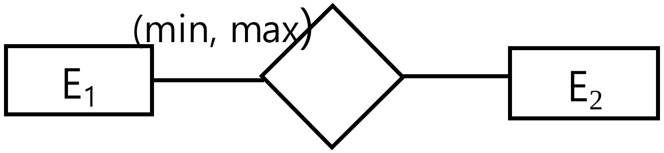


- Department deserves to be an entity type because of the non-key attributes (i.e., Daadr, Dphone).
- Student deserves to be an entity type because it is the "Many" of the M : 1 relationship Belong.

Notation for ER Diagram (1)

Symbol	Meaning
	Entity Type
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

Notation for ER Diagram (2)

Symbol	Meaning
	Relationship Type
	Weak Entity Type
	Weak Relationship Type
	Total/Partial Participation
	N : 1 Mapping
	(Min, Max)

Example: COMPANY Database

- Identifying **Entities, Relationships, Attributes,** and **Constraints:**

- Our company is organized into **departments**. Each department has a name, number and an *only one* **employee** who ***manages*** the department. We keep track of the start date of the department manager.

- Each department ***controls*** *many* **projects**. Each project is controlled by *only one* department. Each project has a name, number, location.

- We store each **employee**'s ssn, address, salary, sex, and birth-date. Each employee ***works for*** one department but may ***work on*** *several* projects. We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the ***direct supervisor*** of each employee.

- Each employee may have a number of **dependents**. For each dependent, we keep track of their name, sex, birth-date, and ***supported*** relationship to employee. (etc.)

ER Diagram : COMPANY Databases

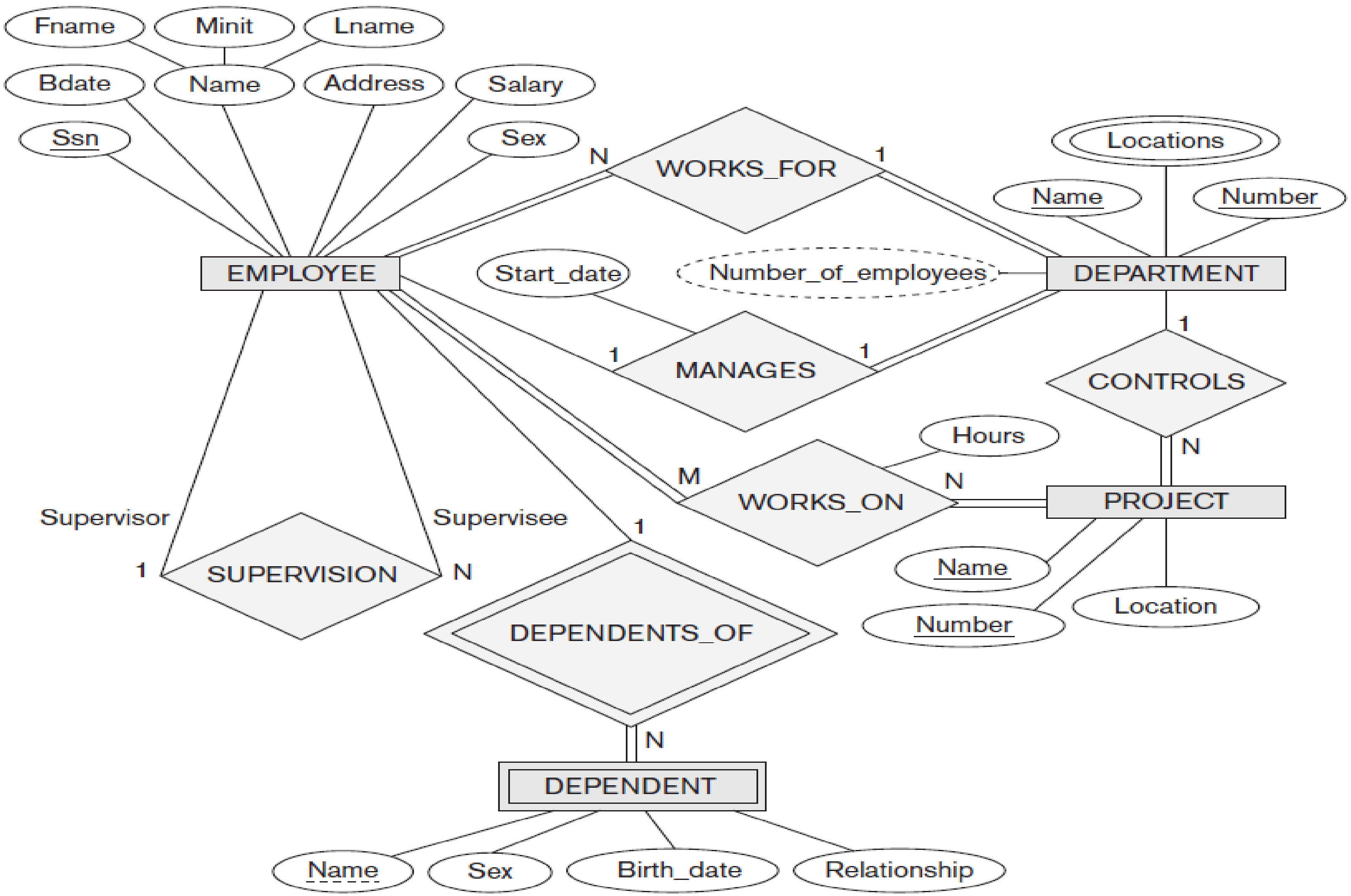


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

ER Diagram using (Min, Max) : COMPANY Databases

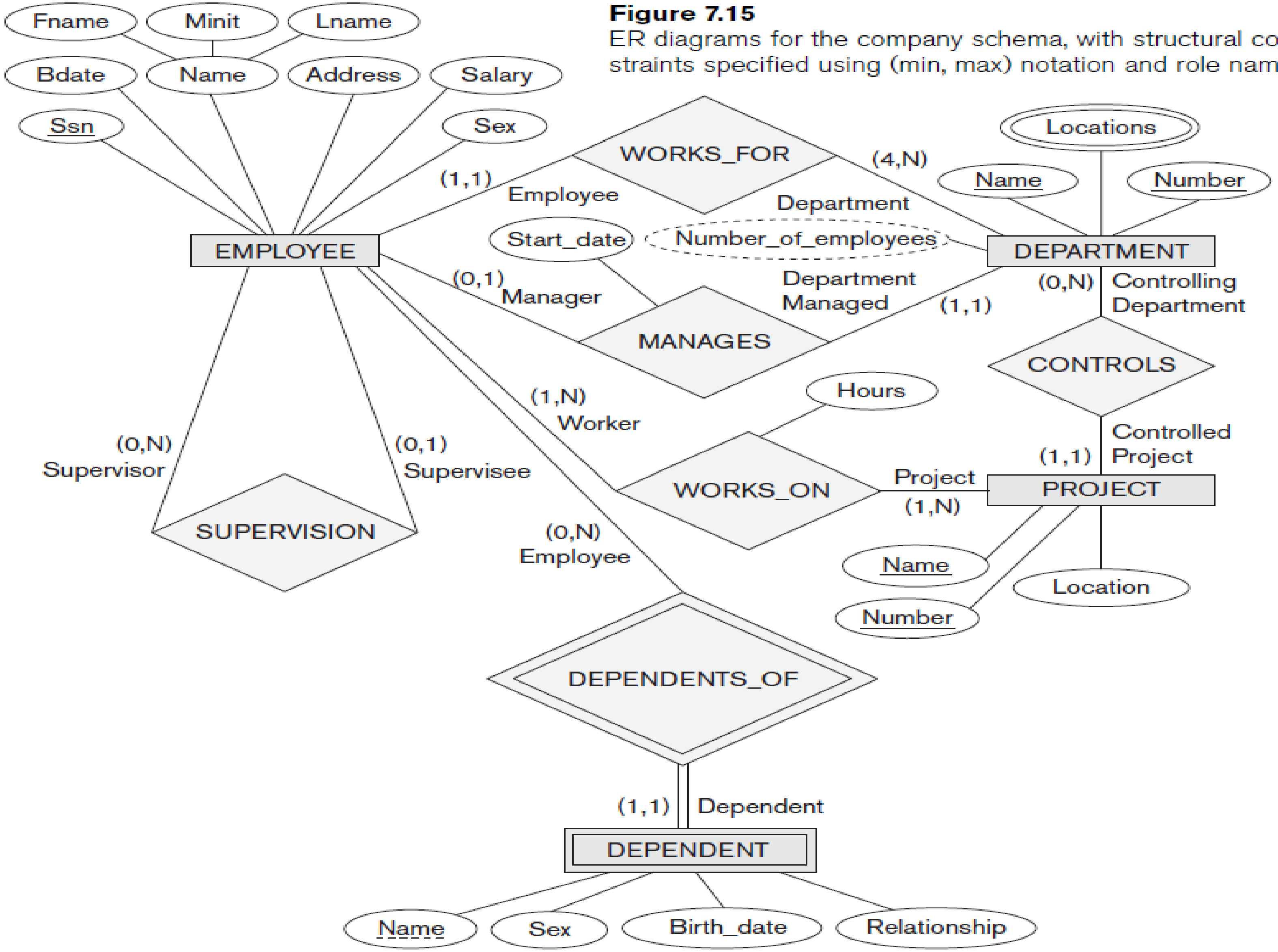
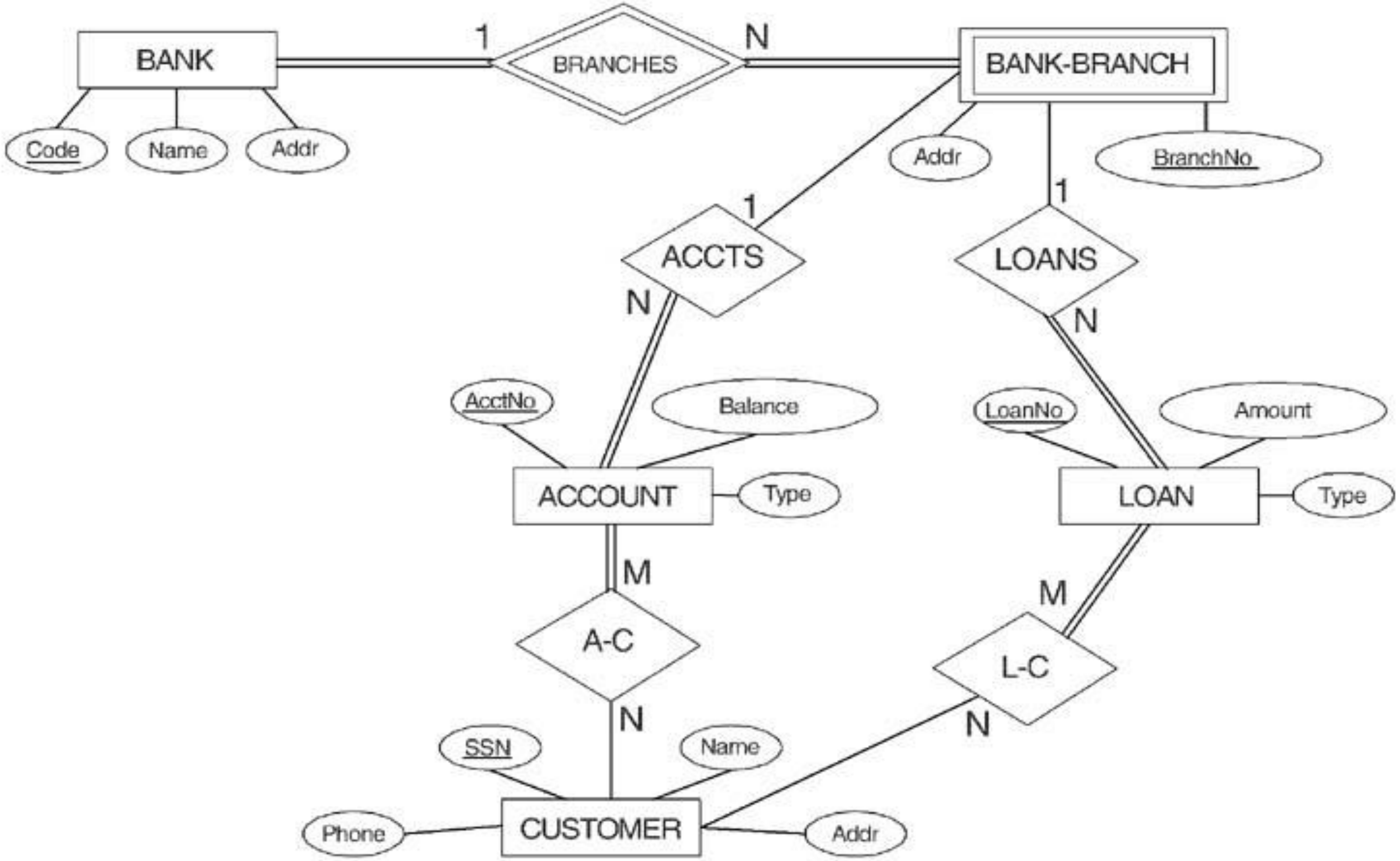


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

ER Diagram : BANK Databases


BANK
Code
Name
Addr
1
BRANCHES
N
BANK-BRANCH
Addr
BranchNo
1
ACCTS
N
1
LOANS
N
AcctNo
Balance
ACCOUNT
Type
LoanNo
Amount
LOAN
Type
M
A-C
N
M
L-C
N
SSN
Name
Phone
CUSTOMER
Addr