2. Conceptual Design :

Entity Relationship (ER) Model

Database Design : 3 Steps

- **Conceptual Design** : **ER Model**
  - Describe data requirements of users
  - Both DBMS and H/W independent
  - High Level Conceptual Schema (= ER Diagram)

- **Logical Design**
  - Actual implementation using specific DBMS
  - DBMS dependent, but H/W independent
  - Logical (Low Level Conceptual) Schema

- **Physical Design**
  - Specify access paths, indexes, and file organizations
  - Both DBMS and H/W dependent
  - Internal Schema

Why Conceptual (ER) Design?

- Can not be handled by automatic tools :
  Database designer has full responsibility.
- Choice of target DBMS can be postponed:
- Change of DBMS does not affect Conceptual Schema
- Application requirements can change;
  - Can still be used as a staring point.
  - Reverse Engineering
- Different DBs can be compared in a uniform framework.
  - Heterogeneous Distributed databases
- ER Model is widely used tool for conceptual design

ER Model : Outlines

- ER Model Concepts
  - Entities and Attributes
  - Key Attributes
  - Relationships
  - Constraints
  - Recursive Relationship
  - Ternary Relationships
  - Weak Entity Types
  - Attributes in Relationship
- ER Design Guidelines
- ER Diagram for COMPANY

Example: COMPANY Database

- **Database Analysis Requirements : Company**

- Our company is organized into departments. Each department has a name, number and an only one employee who manages the department. (. . . . .)
- 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 *direct* supervisor of each employee. (. . . . .)
- Each employee has a number of dependents. For each dependent, we keep track of their name, sex, birth-date, and supported relationship to employee. (. . . . .)
- (etc.)

Entity and Attribute

- **Entity**
  - Entity is a thing (or object) existing in a world.
  - Examples: student, course,

- **Attribute**
  - Attributes are properties used to describe an entity.
  - For example:

- **Attribute Value**
  - An entity has a value for each of its attributes.
  - For example:

Types of Attributes (1)

- **Simple** Attribute
  - An entity has a single atomic value for the attribute.
  - For example:
- **Composite** Attribute
  - An attribute can be composed of other attributes.
  - For Example:
- **Multi-Valued** Attribute
  - An entity has several (> 1) values for the attribute.
  - For example:

Types of Attributes (2)

- **Complex** Attribute
  - Composite + Multi-valued
  - For example:
- **Derived** Attribute
  - Value of an attribute value is computed from other attribute(s)
  - For example:
- **Null** Attribute
  - Value of an attribute is 'unknown' or 'does not exist'
  - For example:

Entity Type and Key

- **Entity type**
  - A collection of similar entities (= sharing the same attributes)
  - For example:

- **Key**
  - A set of attributes to identify each entity uniquely.
  - Every values in a key attribute must be distinct.
  - An entity type may have more than one key attributes.

For example:

- EMPLOYEE의 key는?
- STUDENT의 key는?
- CAR의 key는?

Example: EMPLOYEE Entity Type

EMPLOYEE

(SSN, Name, BirthDate, Age, Phone)

(1234567, Bob, 6-1-1977, 36, 290-7218)

(2345678, Abe, 6-1-1966, 47, {290-7118, 390-7118}),

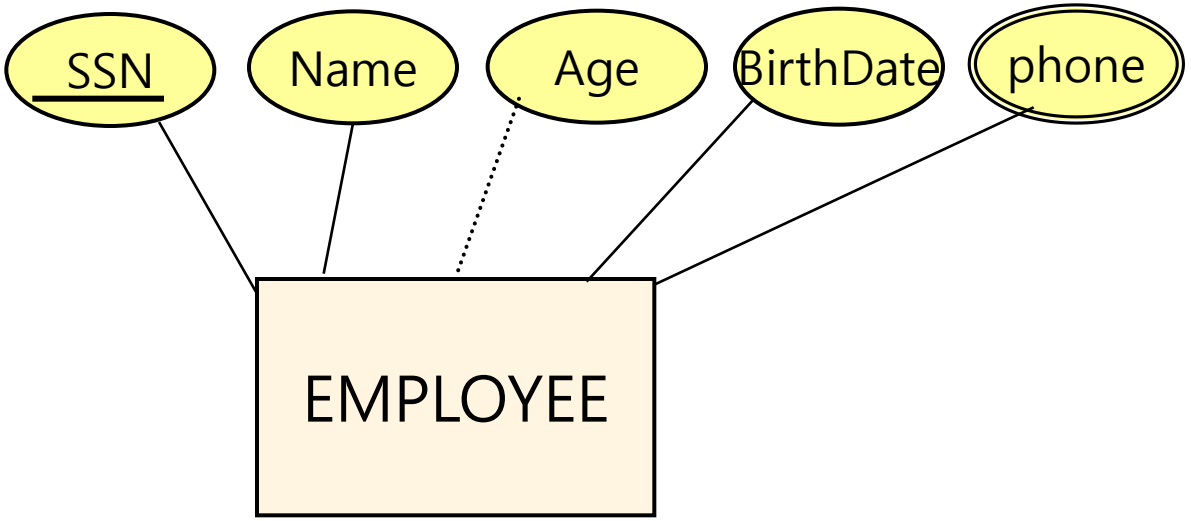
(3456789, Eve, 5-1-1957, 57, 290-7230)

.

.

.

ER Diagram : EMPLOYEE Entity Type



- SSN is a key attribute.
- Age is a derived attribute.
- phone is a multi-valued attribute.

Example: CAR Entity Type

CAR

Registration(Plate-Number, State), ID, Model, Year, Color

((ABC 123, TEXAS), TK629, Mustang, 1997, black)

((ABC 123, New York), WP9872, Sonata, 2002, blue)

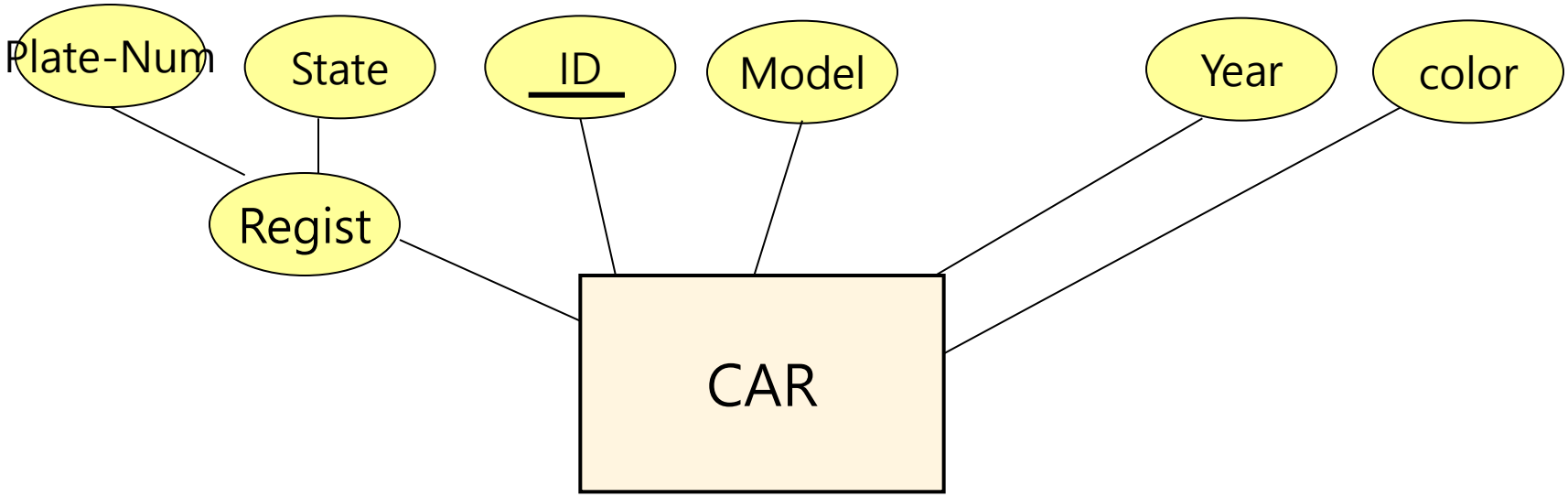
((VSY 720, TEXAS), TD729, Mercedes, 1, 2006, white)

.

.

.

ER Diagram : CAR Entity Type



- Registration is a composite attribute;
- ID is a key attribute;

Relationship (Type)

- **Relationship**
  - It connects (or relates) many entity types.

- **Relationship Type**
  - A collection of similar relationships .

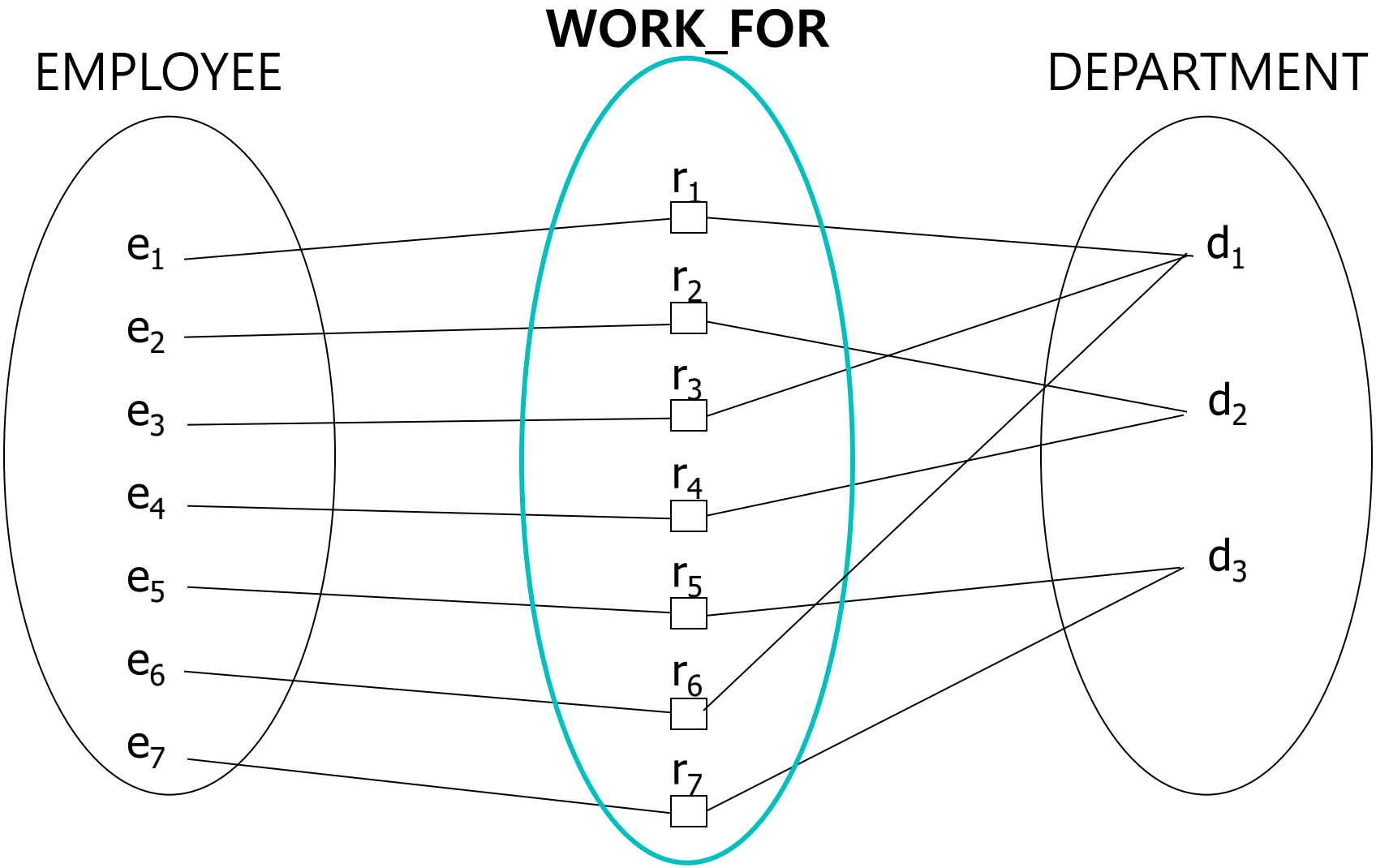
- **Degree** (of a relationship type)
  - the number of participating entity types in a relationship

  (1) Unary (= Recursive) Relationship : Degree 1

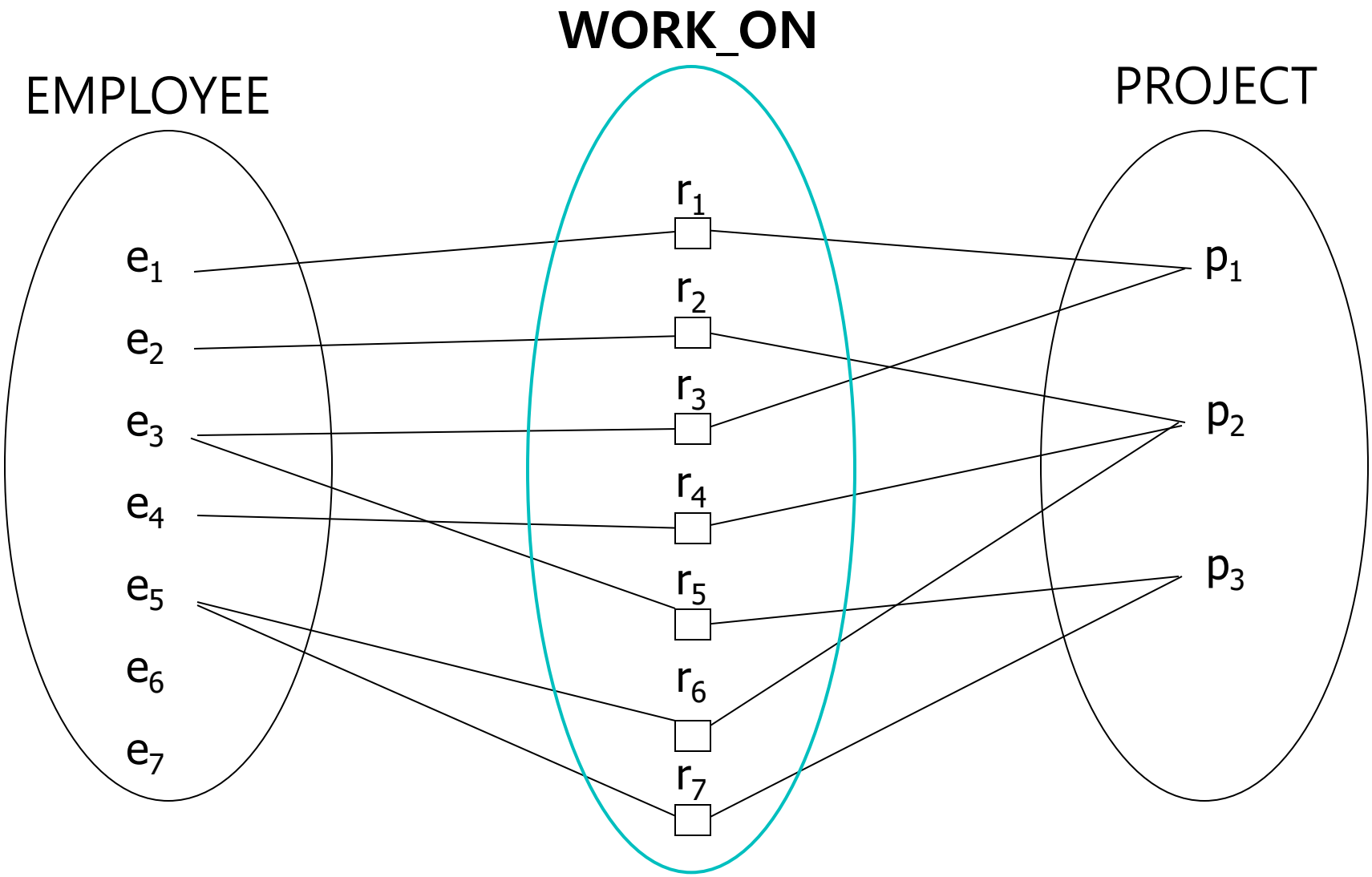
  (2) Binary Relationship : Degree 2

  (3) Ternary Relationship : Degree 3

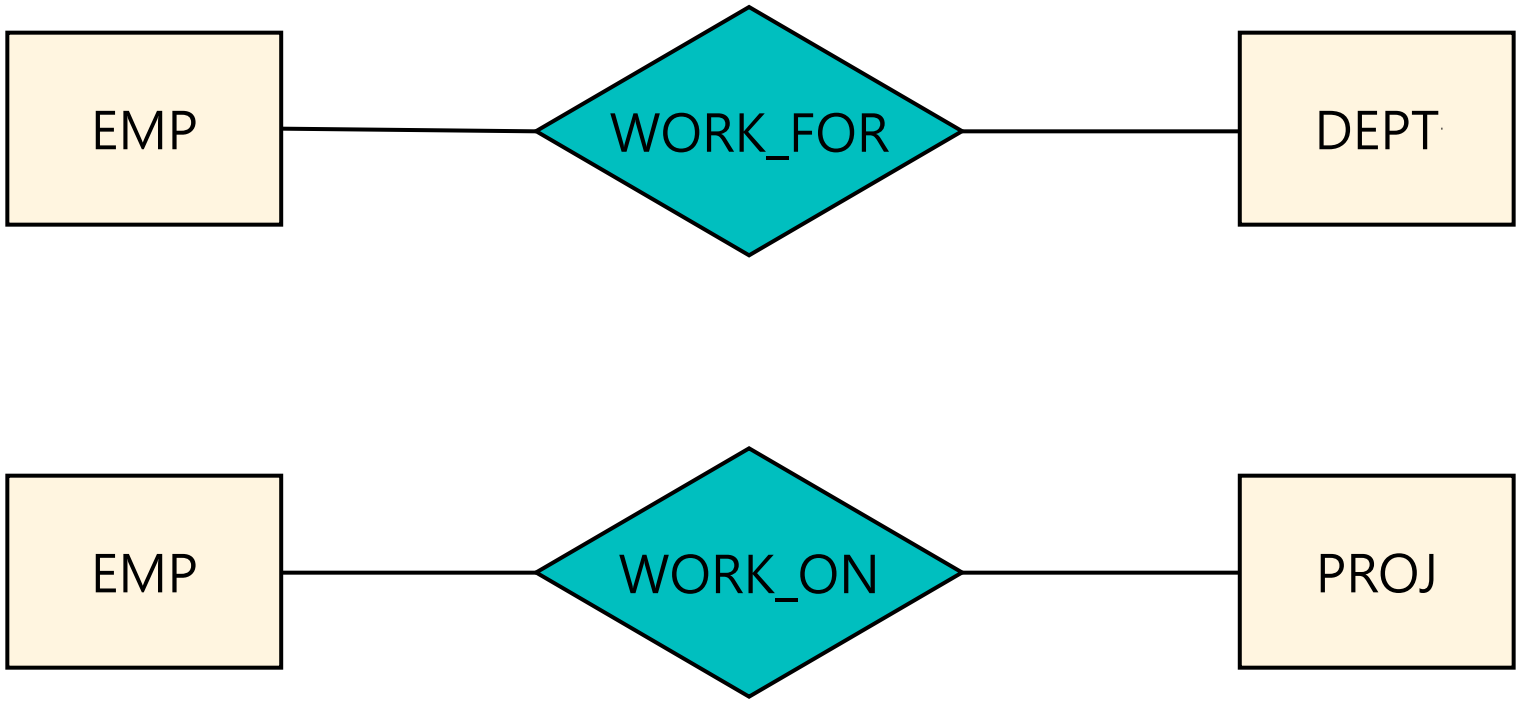
WORK_FOR Relationship Type



WORK_ON Relationship Type



ER Diagram : WORK_FOR and WORK_ON Relationship Type



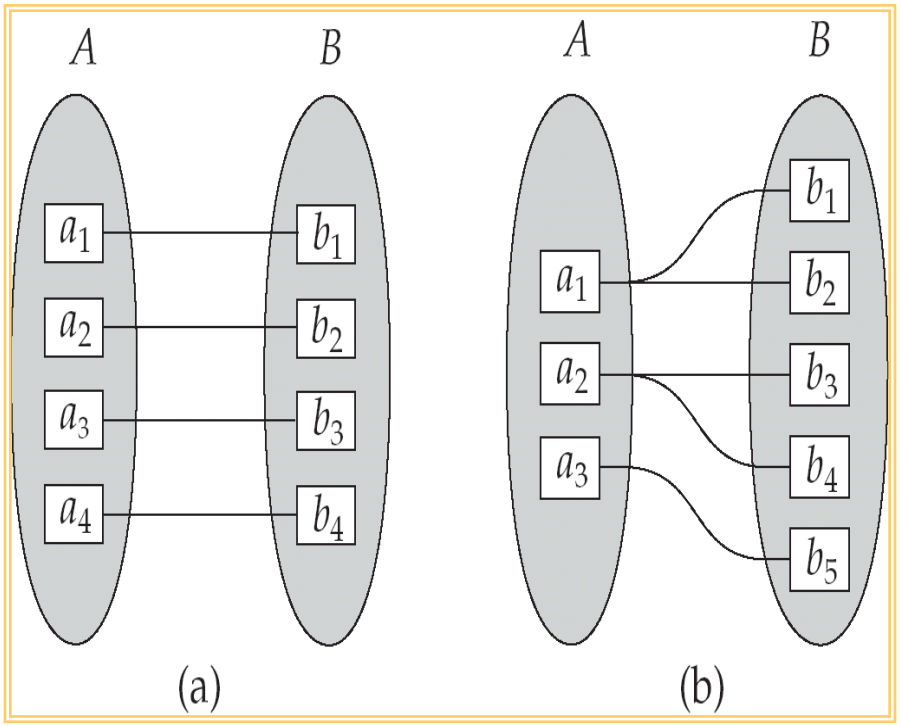
Constraints on Relationship(1)

- **Mapping** Constraints

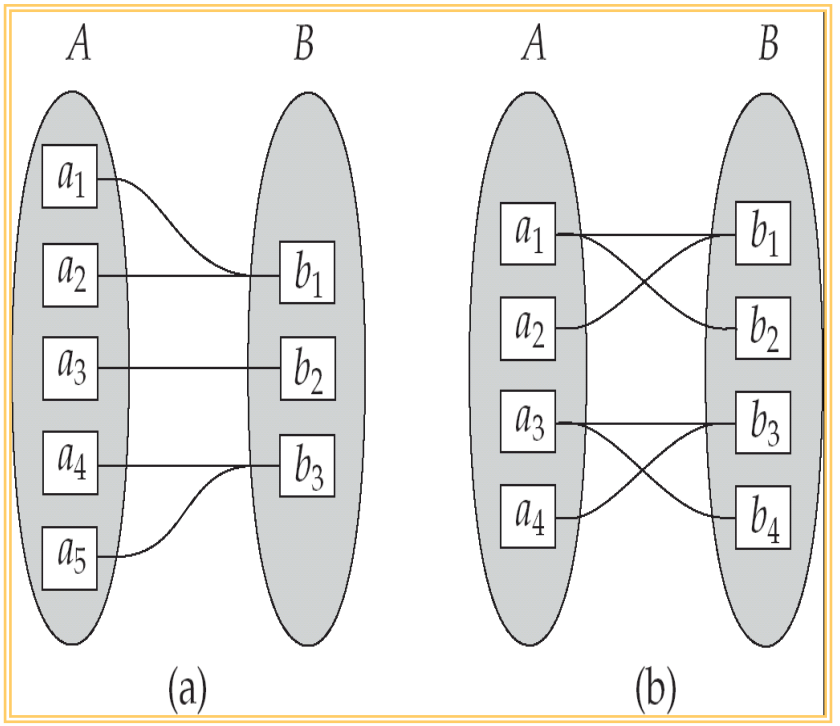
  : Express the number of entities to which another entity can be associated via a relationship set.

  - One to One (1 : 1)
  - Many to One (M : 1)
  - One to Many (M : 1)
  - Many to Many (M : N)

Mapping Constraints



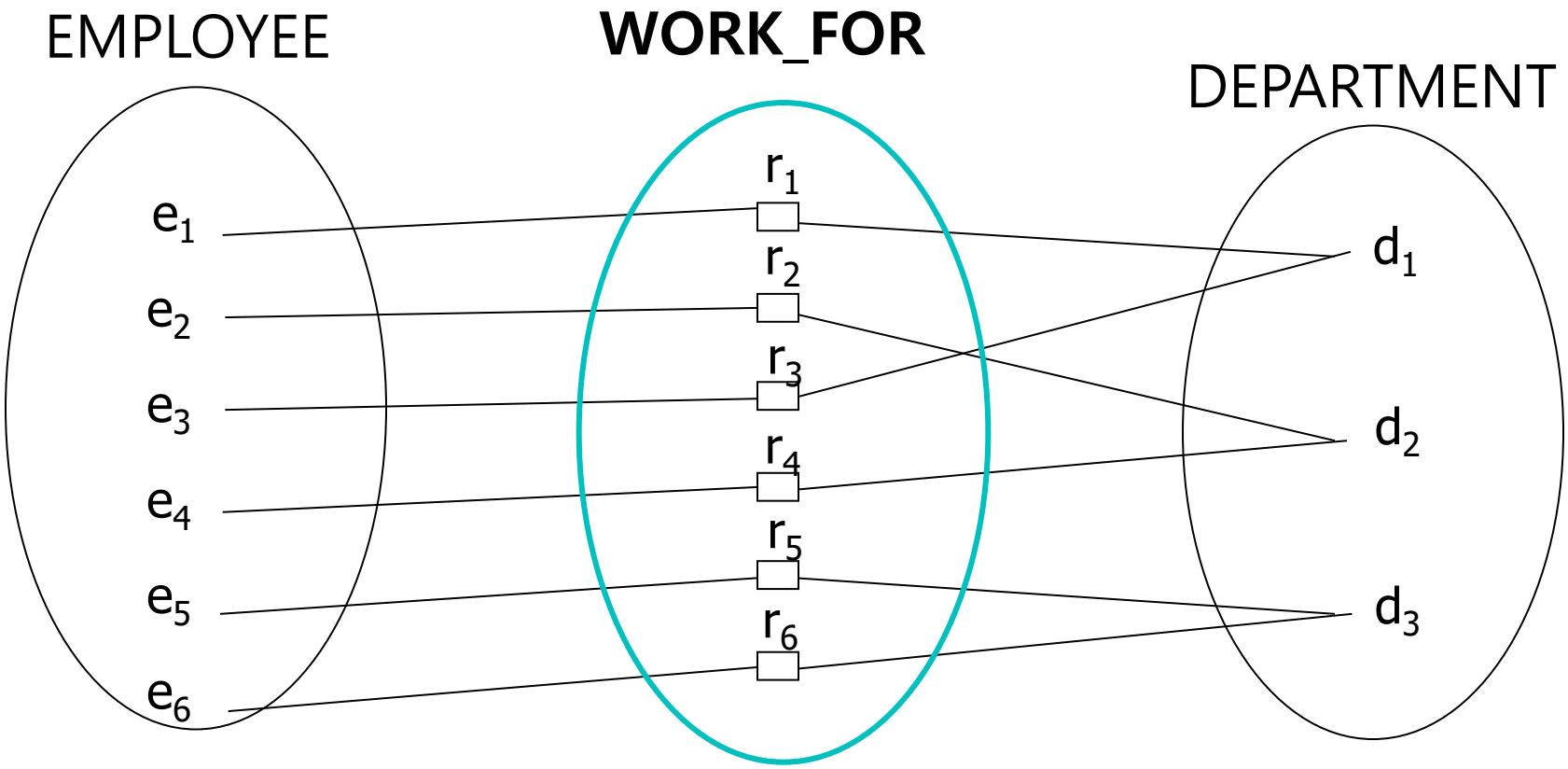
1 : 1 1 : M



M : 1 M : N

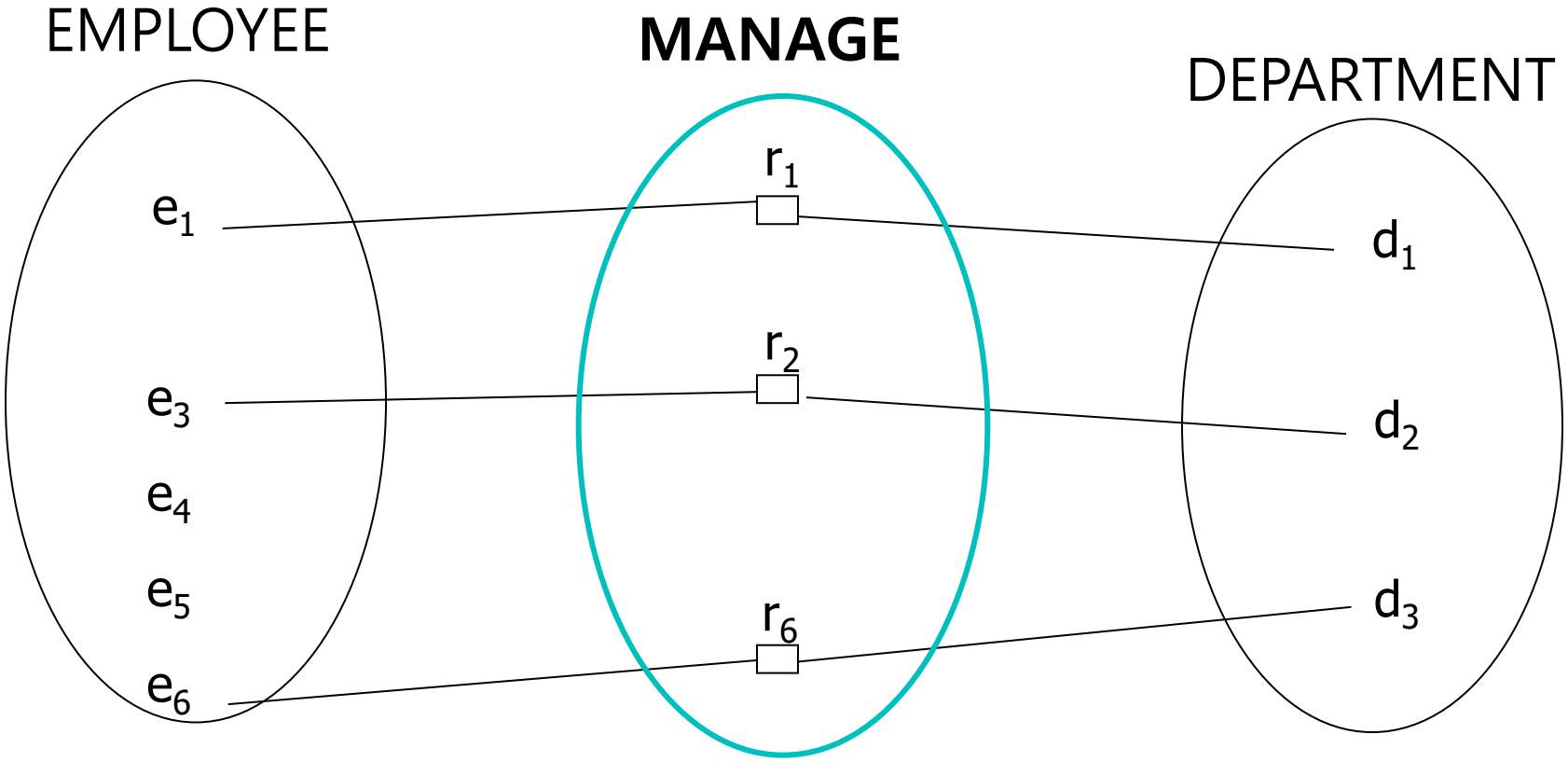
Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

WORK_FOR Relationship Type (M : 1)



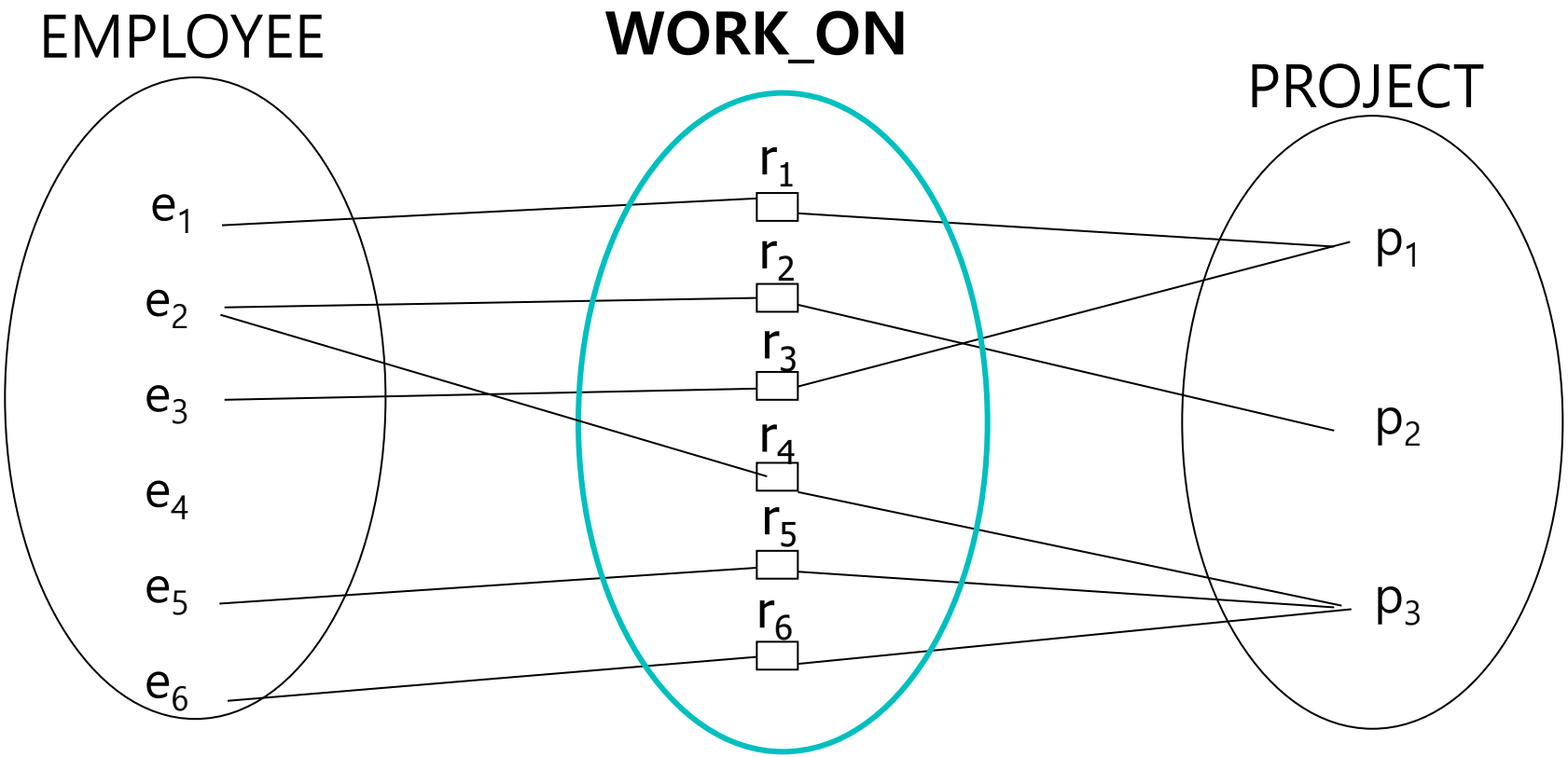
"Each employee works for at one department, but a department can have many employees for it."

MANAGE Relationship Type (1 : 1)



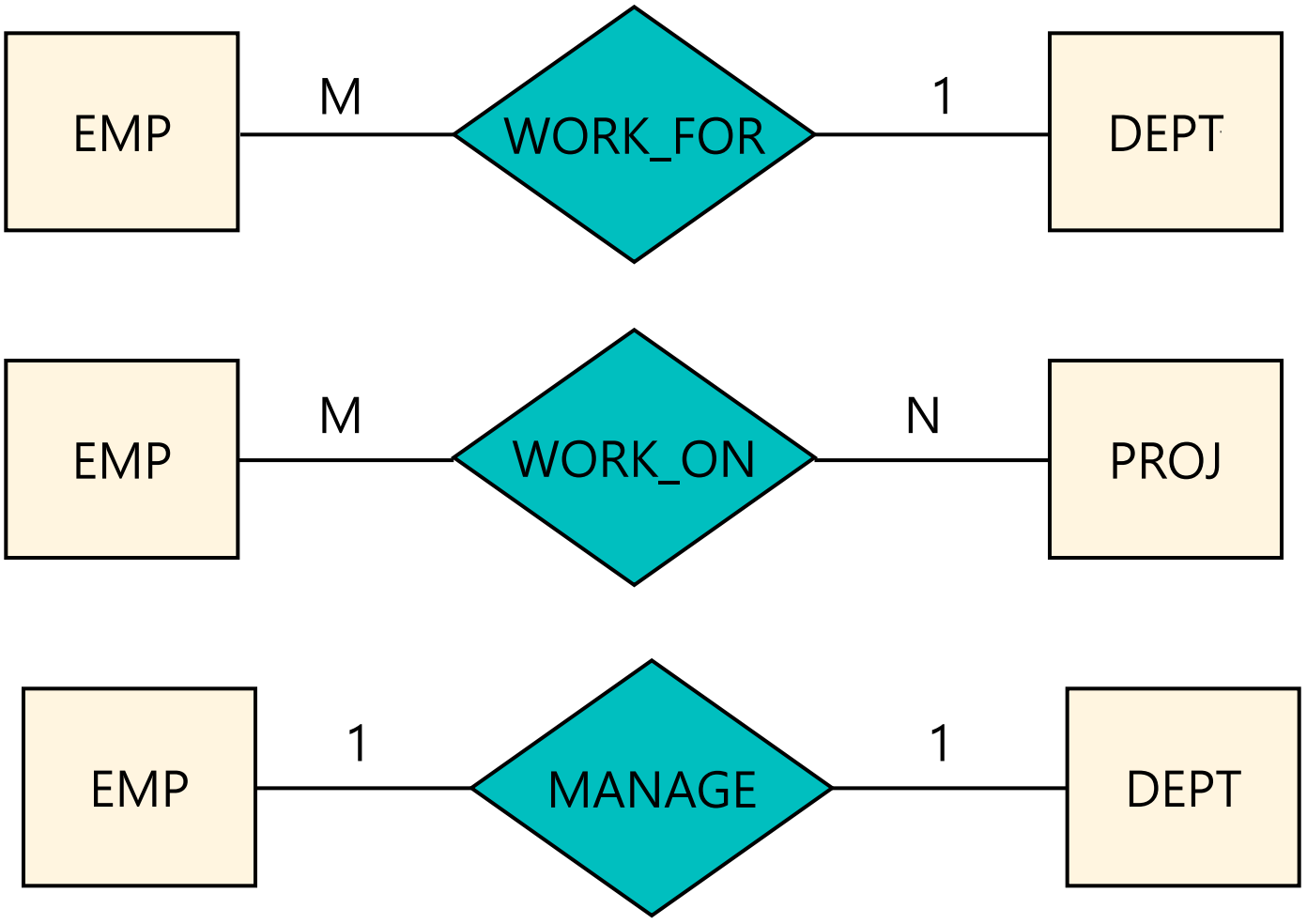
"Each employee works for at one department, and department has one employee for it."

WORK_ON Relationship Type (M : N)



"Each employee can work on many projects, and a project can have many working employees for it.

ER Diagram : Mapping Constraints



Constraints on Relationship(2)

- **Participation** Constraints

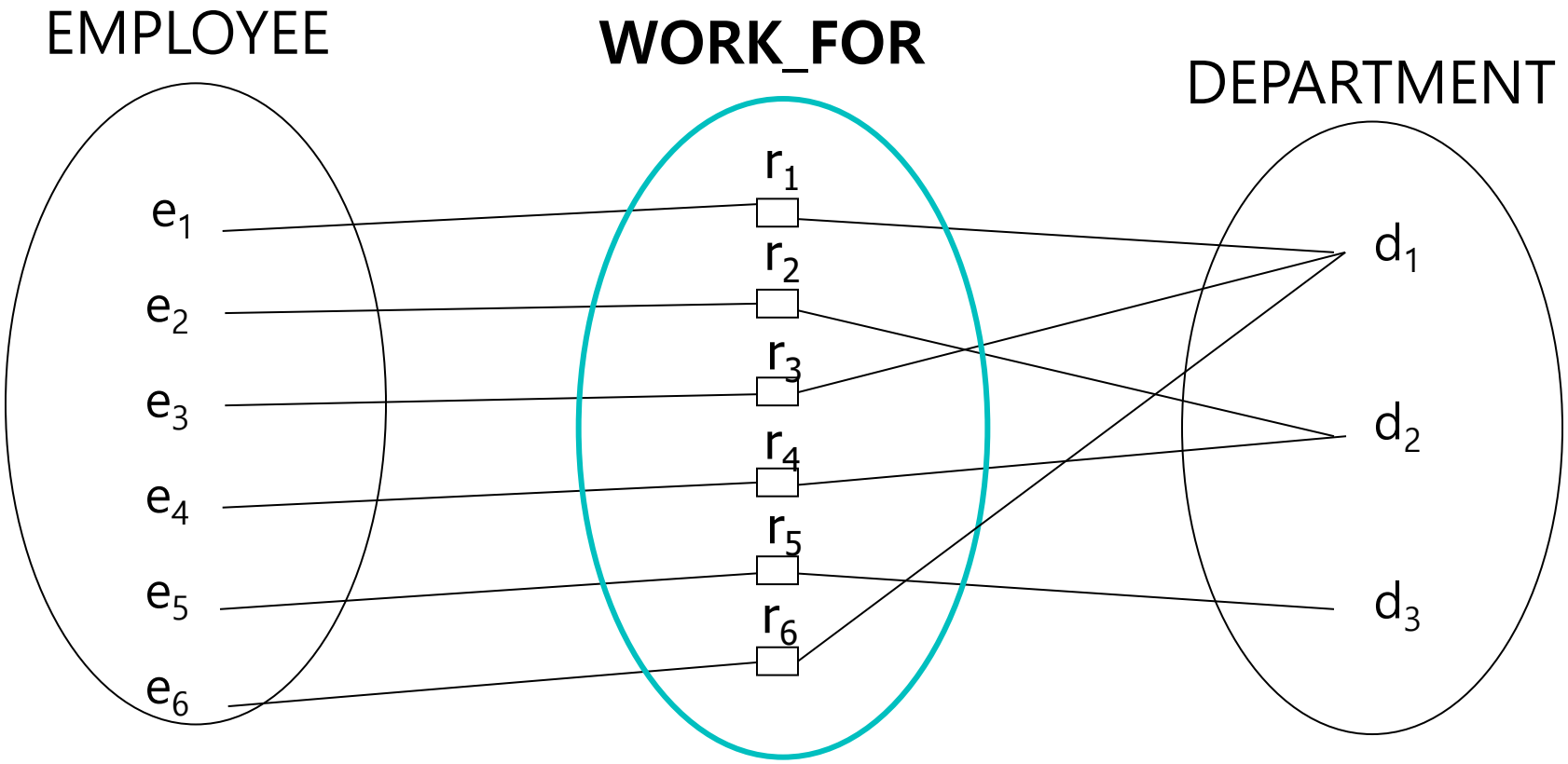
  - **Total (= Mandatory)**

    - Every entity must participate (in a relationship).
    - It is not allowed that there are any entities not participating in relationship.

  - **Partial (= Optional)**

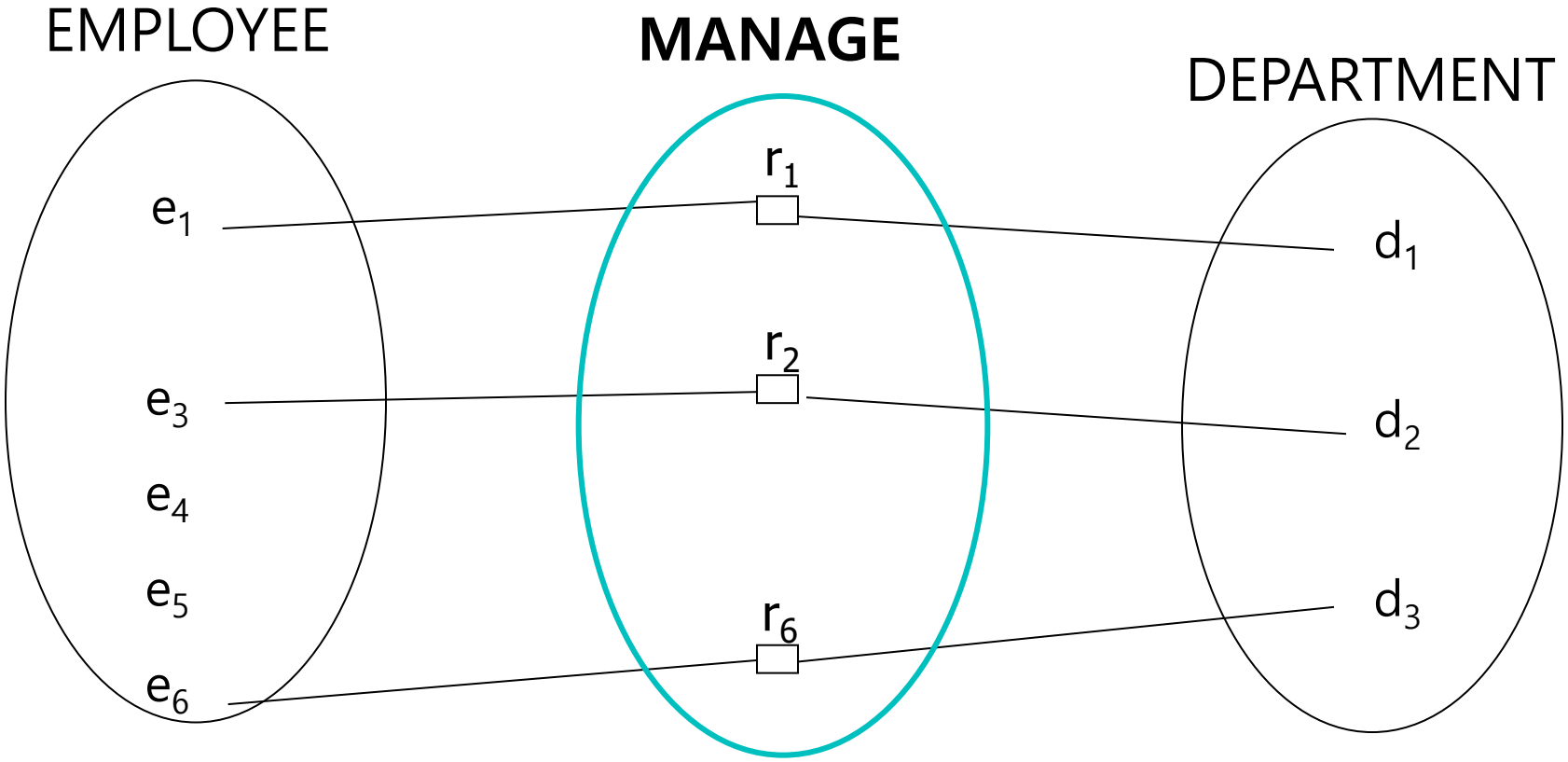
    - Some entities may not participate.

WORK_FOR Relationship Type (Total : Total)



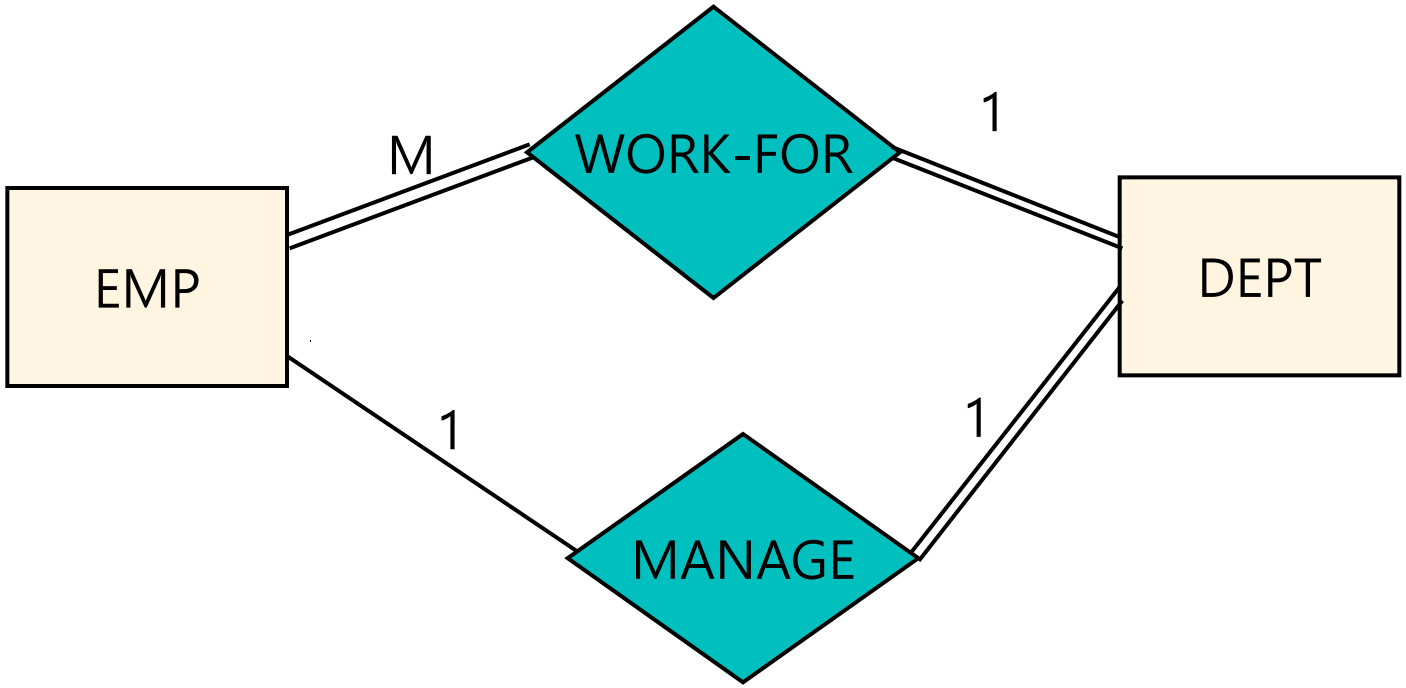
"Every employee must work for some department, and every department must have working employee for it."

MANAGE Relationship Type (Partial : Total)



"Not every employee needs to manage a department, but every department must have some manager for it.

ER Diagram : Total/Partial Constraints



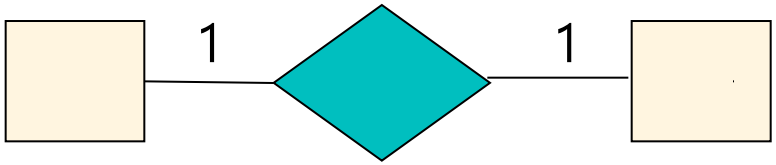
Total : ═══

Partial : ———

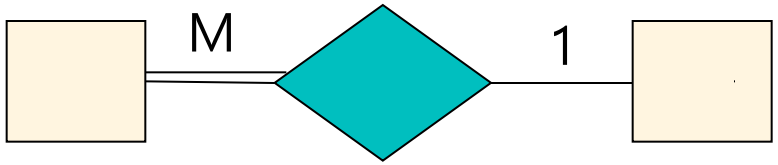
Exercises : Relationship

- Show examples of each of the following relationships;

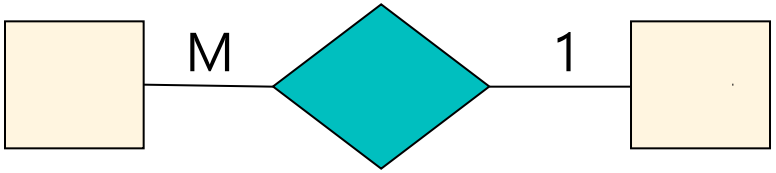
(a)



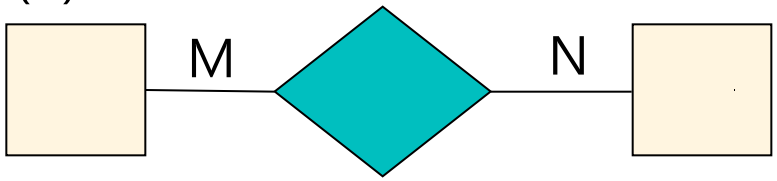
(b)



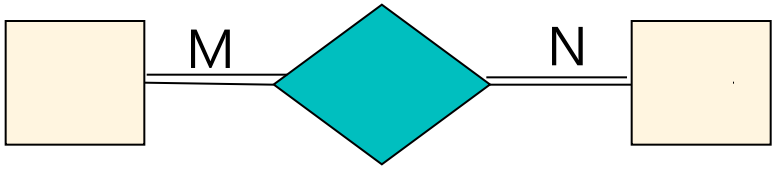
(c)



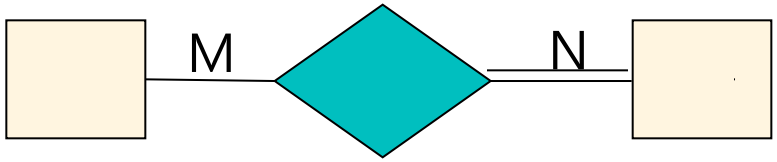
(d)



(e)



(f)

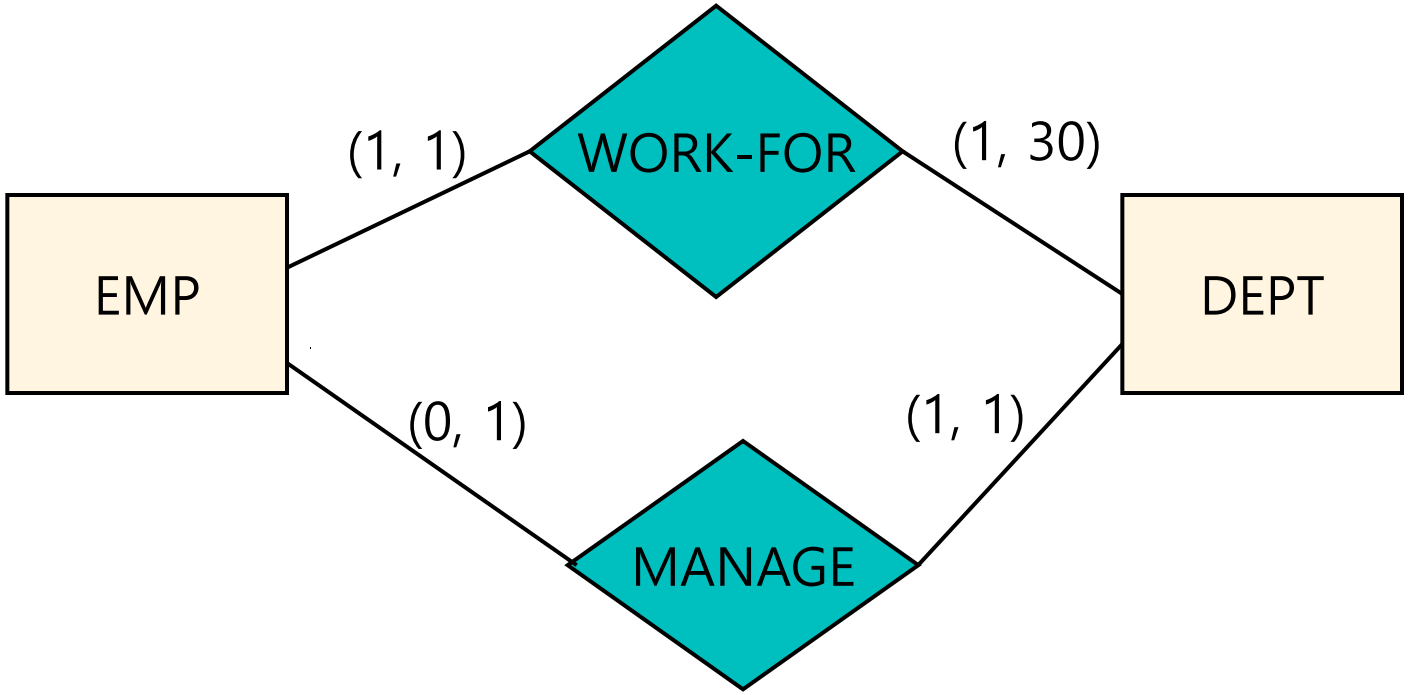


Constraints on Relationship(3)

- **(Min, Max)** Constraints

  - Assign (**min**, **max**) to each entity type.
  - **min** means "at least"; **max** means "at most"
  - **min = 0** means **partial**
  - **min > 0** means **total**
  - **min** ≤ **max**; **min** ≥ 0; n ≥ **max** ≥ 1

ER Diagram : (Min, Max) Constraints

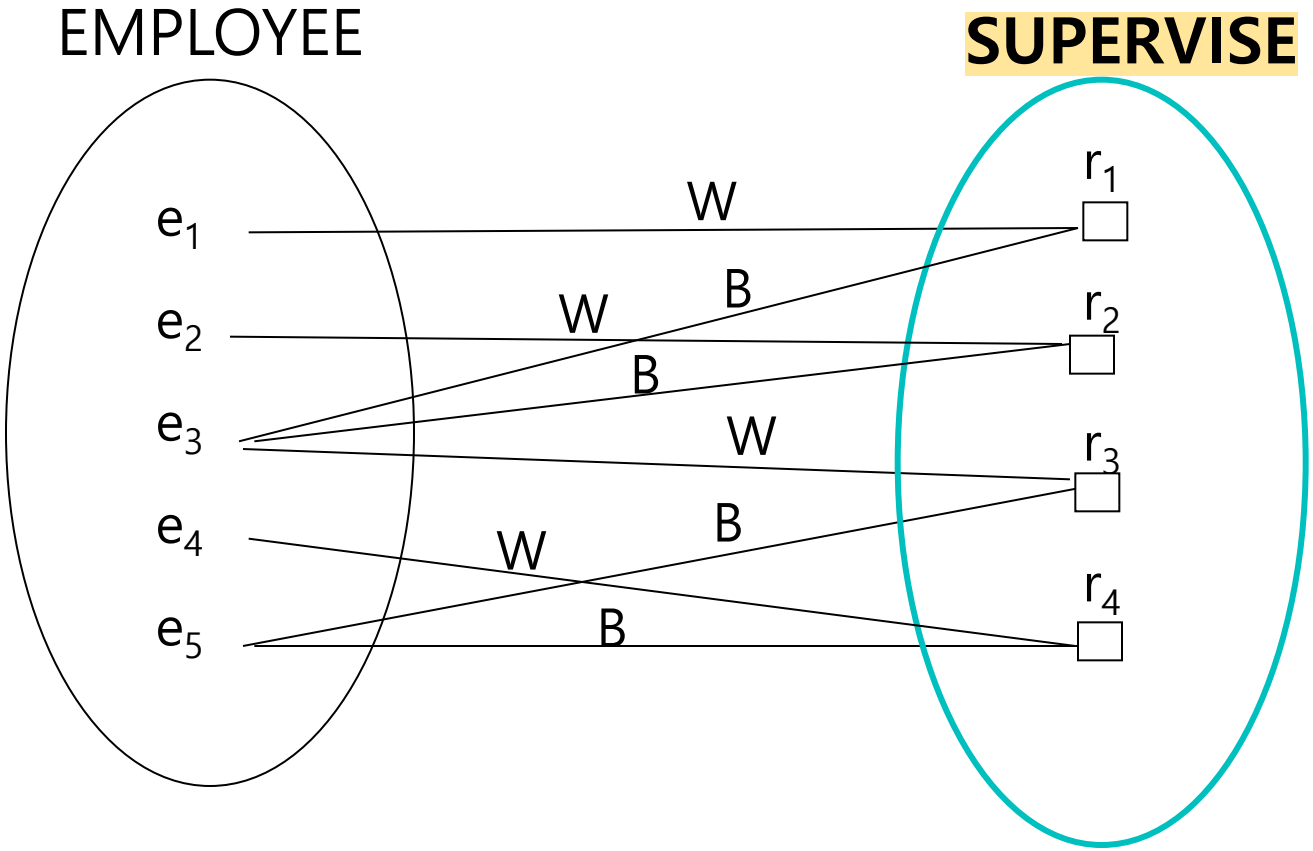


- "Each department must have *exactly one* manager and an employee can manage *at most one* department"
- "Each employee must work for *exactly one* department, but a department must have *maximum 30* employees"

Recursive Relationship

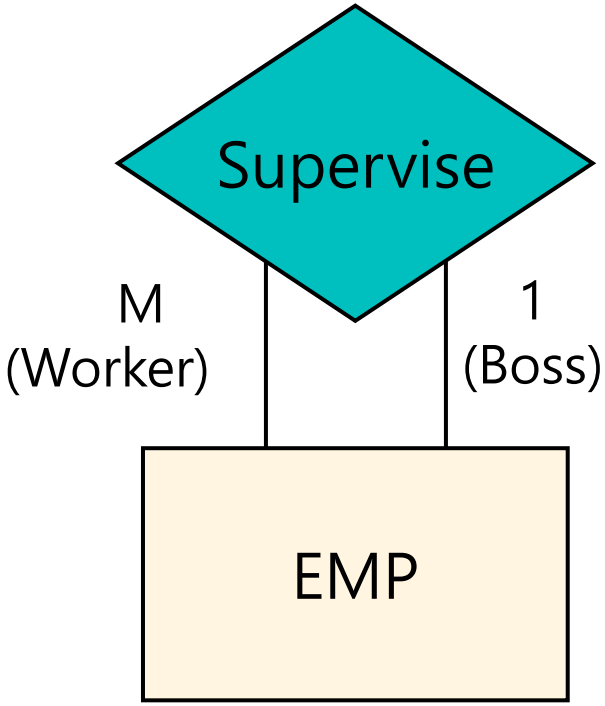
- Relationship with degree 1 is called **recursive**.
- Both participating entity types are the same.
- For example :
  - 사람과 사람들간의 관계
  - 학생과 학생들간의 관계
  - 과목과 과목들 간의 관계
  - (컴퓨터) 부품과 부품들 간의 관계
- 참여하는 양쪽의 동일한 entity type들을 구분하기 위해 서로 맡은 다른 **role** (역할)들이 필요함.
- ER diagram 그릴 때 role을 표기함.

Recursive Relationship : SUPERVISE



- role W : Supervisee (Worker)
- role B : Supervisor (Boss)

Recursive Relationship
1 : M

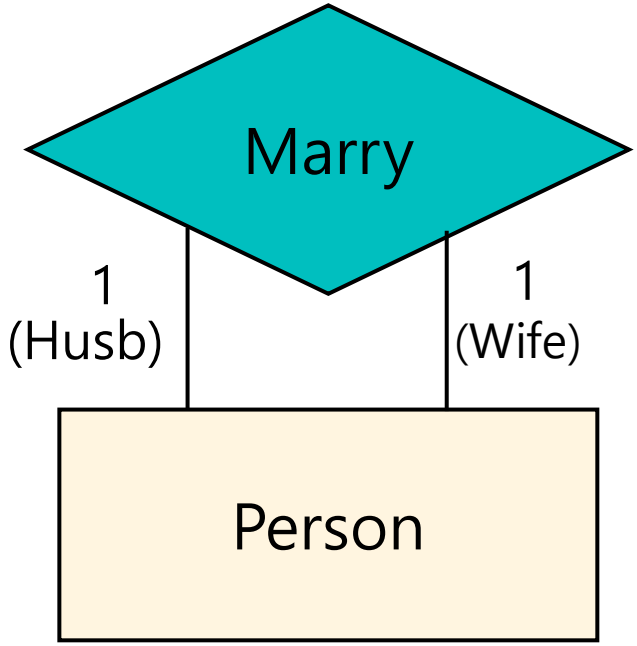


Example :

worker	boss
bob	joe
abe	joe
joe	ann
ann	eve

- "Each boss can have many workers" (1 : M)
- "Each worker can have only one boss" (M : 1)

Recursive Relationship
1 : 1

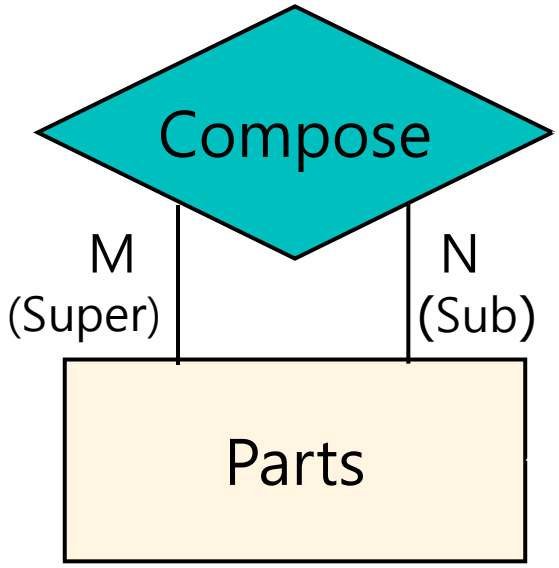


Example :

hus	wife
adam	eve
tarjan	jane
dick	jane
pete	jolie

- "Each person can have only one wife" (1 : 1)
- "Each person can have only one husband" (1 : 1)

Recursive Relationship
M : N



Example :

Super	Sub
A	B
A	C
B	D
B	E
F	C

- "A part consists of many different subparts" (1 : M)
- "A part can be subparts of many different parts" (N : 1)