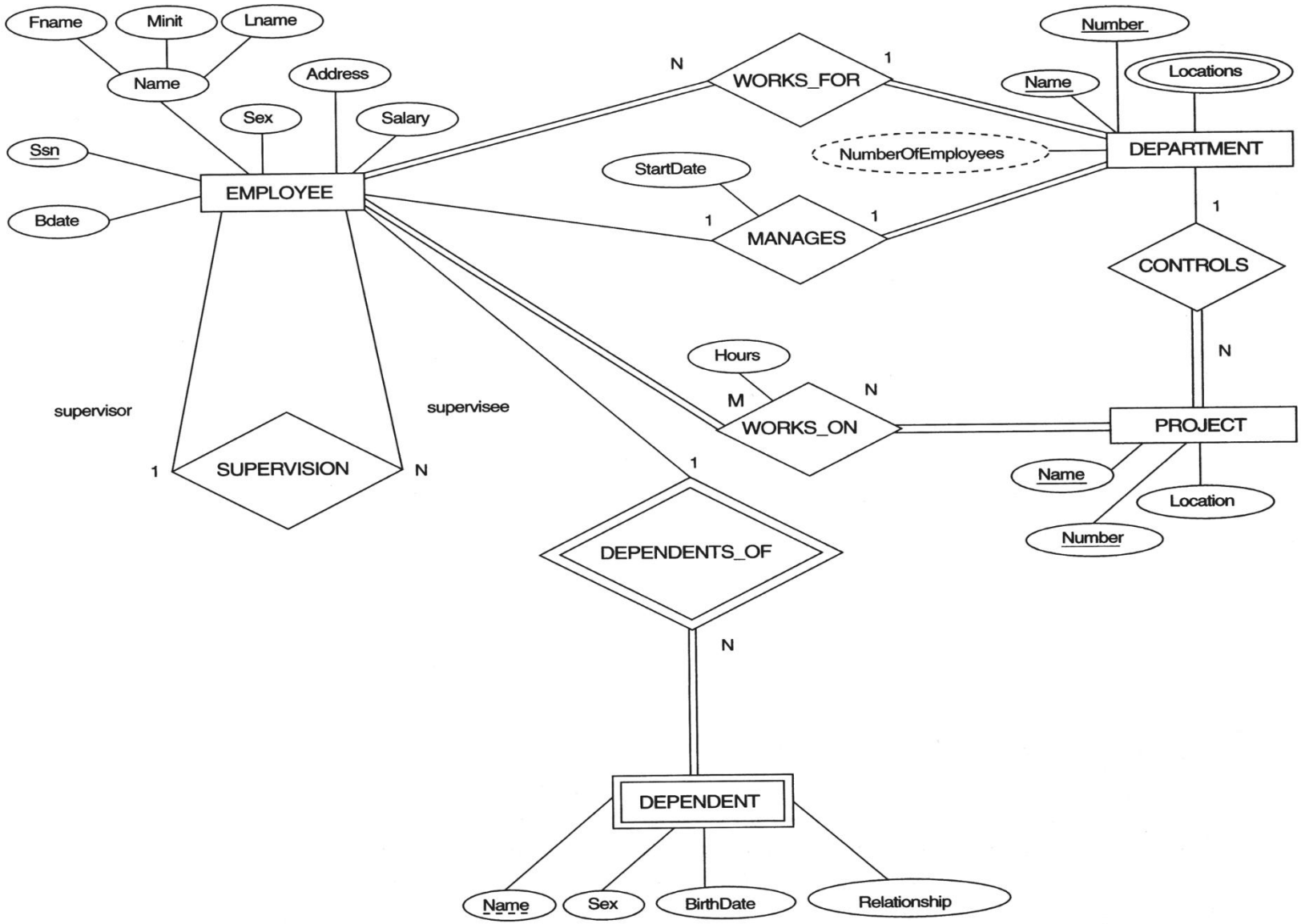
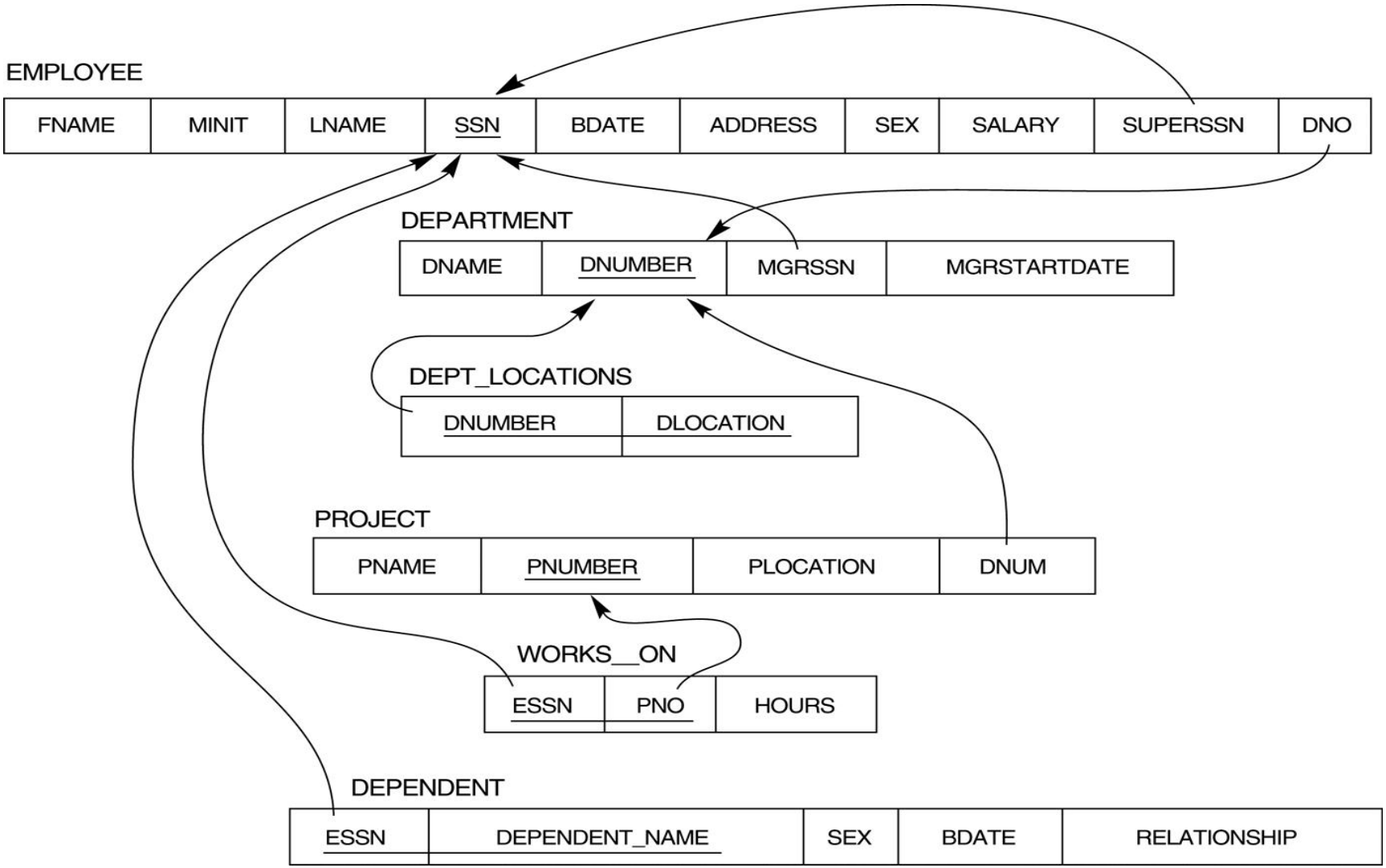
6. Relational Schema Design

ER to Relational Mapping

ER schema


Fname
Minit
Lname
Name
Address
Sex
Salary
Ssn
Bdate
EMPLOYEE
N
WORKS_FOR
1
Number
Name
Locations
DEPARTMENT
NumberOfEmployees
StartDate
1
MANAGES
1
1
CONTROLS
N
Hours
M
N
WORKS_ON
PROJECT
Name
Location
Number
supervisor
supervisee
1
SUPERVISION
N
1
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
BirthDate
Relationship

Mapping Result : Relational schema



What to Map?

- (Regular) Entity Types
- Weak Entity Types
- Binary 1 : M Relationships
- Binary M : N Relationships
- Binary 1 : 1 Relationships
- Recursive Relationships
- Multi-Valued Attributes
- Ternary Relationships
- Superclass/Subclass : IS-A Relationship

Mapping Guidelines

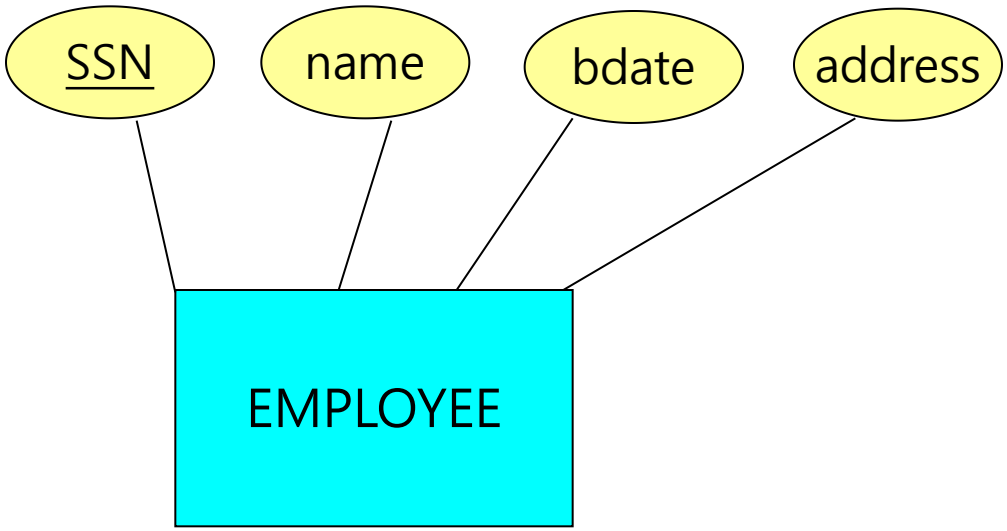
- Need to satisfy the following constraints;
  - 1 : M, M : N, 1 : 1 relationship
  - Total / Partial participation
  - Key Integrity
  - Referential Integrity
- Avoid many null values.
- Consider performance (= retrieval time).
- Avoid redundancy.

Entity Types

- For regular entity type **E**, create a relation **R** that includes all the simple attributes of **E**.
- Choose one of the keys of **E** as primary key (**PK**) for **R**.
- If the chosen key of **E** is composite, the set of simple attributes that form it will together form the **PK** of **R**.
- Each entity in **E** corresponds a row (tuple) in **R**
- Each attribute in **E** corresponds a column (attribute) in **R**

Entity Types

ER :



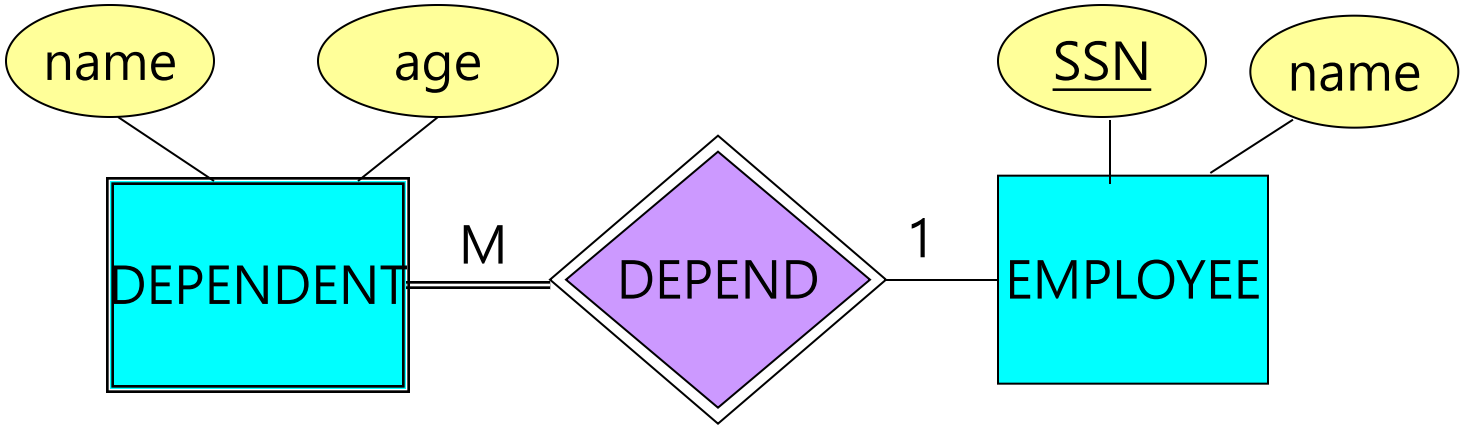
Relation : **EMPLOYEE (SSN, name, bdate, address)**

——— : PK

Weak Entity Types

- For weak entity type **W**, create a relation **R** and include all the attributes of **W**.
- Find **W**'s owner entity type **E**;
- Include as foreign key (**FK**) of **R** the **PK** of the owner **E**.
- **PK** of **R** is {**PK** of owner **E**, partial key of **R**}

Weak Entity Types: 예



- DEPENDENT is a weak type; EMPLOYEE is an owner type.

DEPENDENT (name, SSN, age)

EMPLOYEE (SSN, name)

——— : PK

- - - - - - - : FK

1 : M Relationship : Total/Total

Case 1 **:** Both sides are **'Total':**

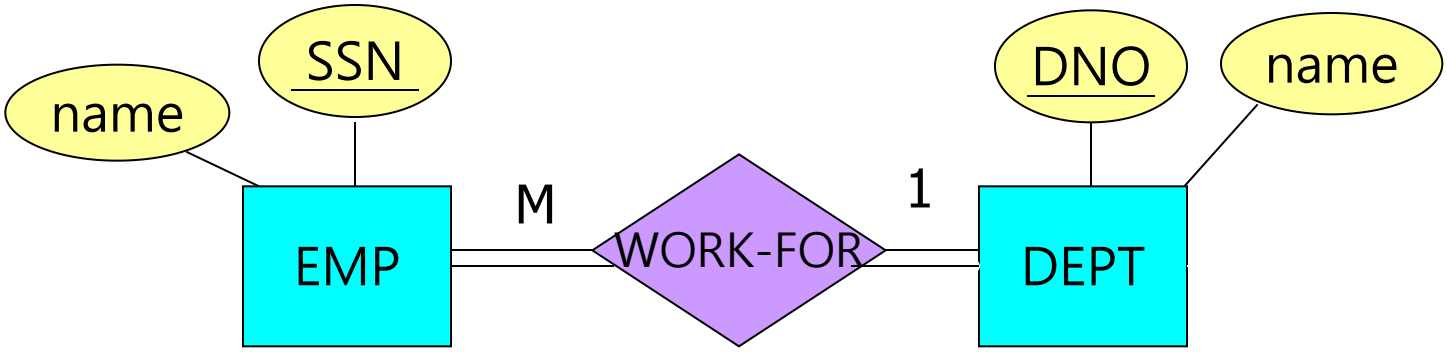
- For 1 : M relationship **R**, create a relation **S** representing the entity type at the "M-side" of the relationship.

- Include as **FK** in **S** the **PK** of the relation **T** that represents the entity type at the "1 – side".

  : Why? This is because we must satisfy key integrity;

- Include any simple attributes of the 1 : M relationship as attributes of **S**.

1 : M Relationship : Total/Total



EMP (SSN, name, DNO) ——— : PK

DEPT (DNO, name) - - - - - - : FK

- Include the **PK** 'DNO' of DEPT relation (at the 1-side) as **FK** in EMP relation (at the M-side)
- WORK-FOR exists between DNO (in EMP) and DNO (in DEPT).

1 : M Relationship : Partial/Partial

Case 2 : Both sides are **'Partial'**

- For 1 : M relationship **R**, create a **new** relation **S** to represent **R**.

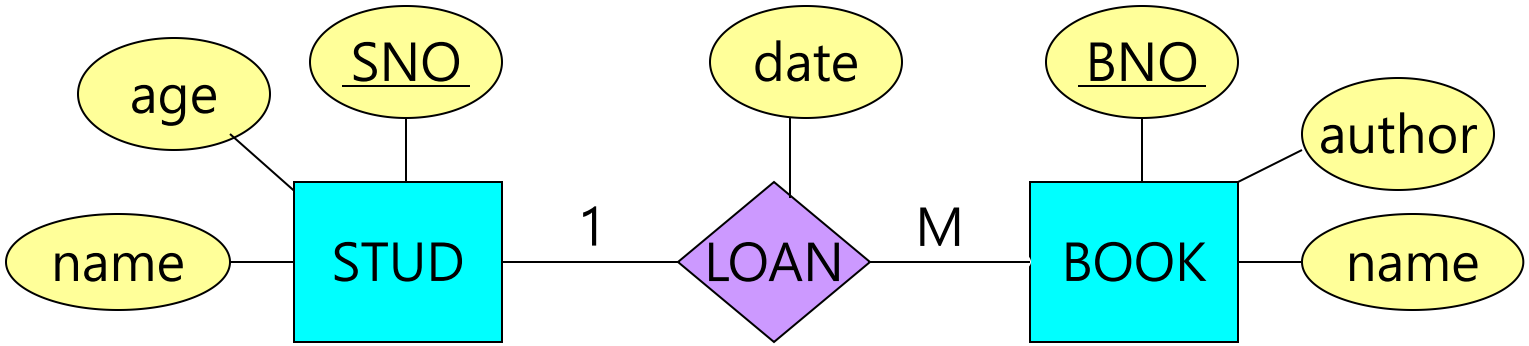
- Include as **FK** in **S** the **PK**s of each relations that represent the participating entity types;

  : Why? This is because we need to avoid many null values.)

- Also, include any simple attributes of the relationship type **R** as attributes of **S**.

1 : M Relationship : Partial/Partial

- Method A :



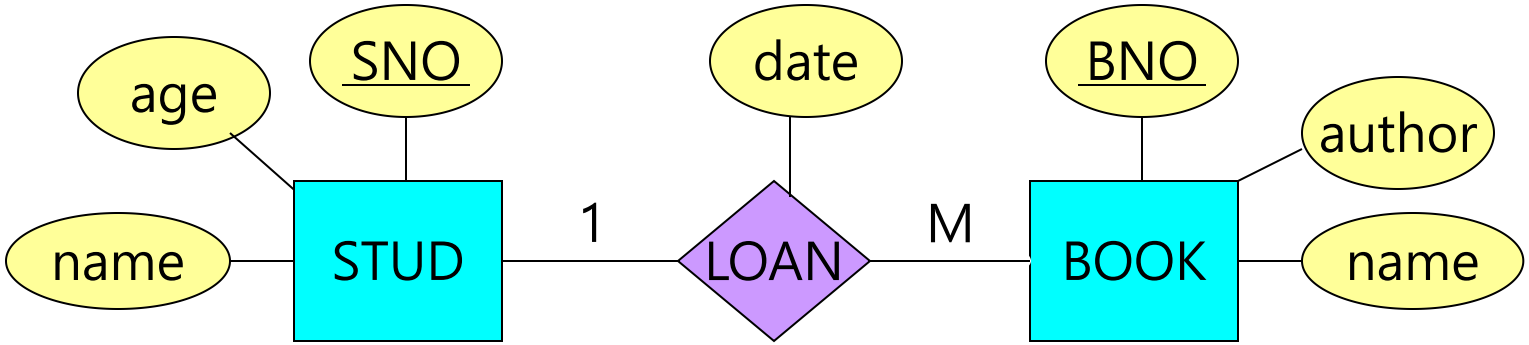
STUD (SNO, name, age)

BOOK (BNO, name, author, date, SNO)

- Problem : There may exist many null values in a BOOK relation; Why?

1 : M Relationship : Partial/Partial

- Method B :



STUD (SNO, name, age)

BOOK (BNO, name, author)

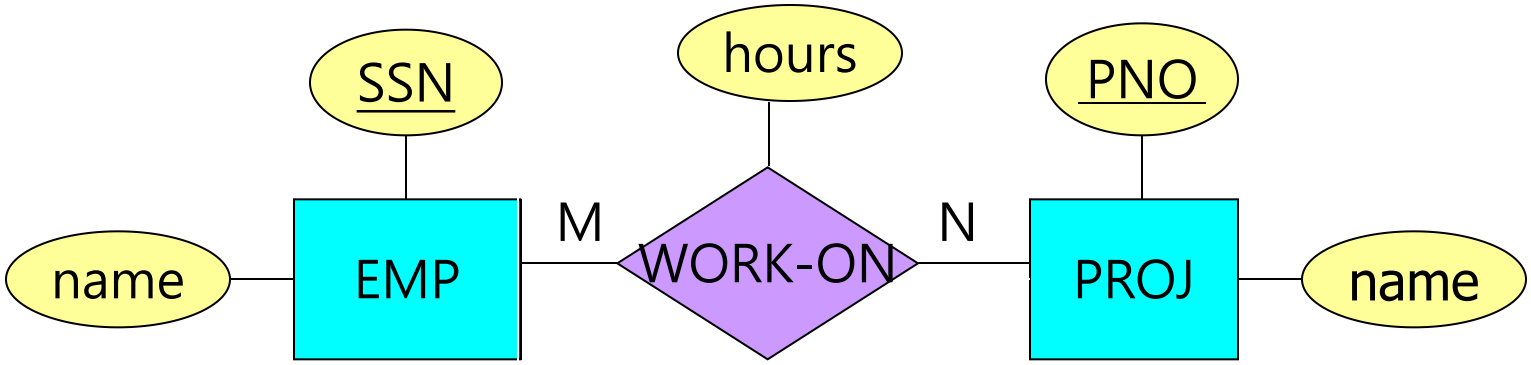
LOAN (BNO, SNO, date)

- Create a new relation LOAN, where each one of {SNO, BNO} is **FK**; BNO is **PK**; Why?
- No more null values! But, we need two join operations.

M : N Relationship

- For M : N relationship **R**, create a **new** relation **S** to represent **R**.
- Include as **FK** in **S** the **PK**s of each relations that represent the participating entity types;
- The combination of each **FK**s will form the **PK** of **S**.
- Also, include any simple attributes of the M : N relationship type as attributes of **S**.

M : N Relationship



EMP (SSN, name)

PROJ (PNO, name)

WORK-ON (SSN, PNO, hours)

- Create a new relation WORK_ON. Each **PK** of PROJ and EMP are included as **FK**s in WORK_ON.
- {SSN, PNO} is **PK** of WORK-ON relation.

1 : 1 Relationship

- For 1 : 1 relationship **R**, create the relations **S** and **T** that correspond to the entity types participating in **R**.

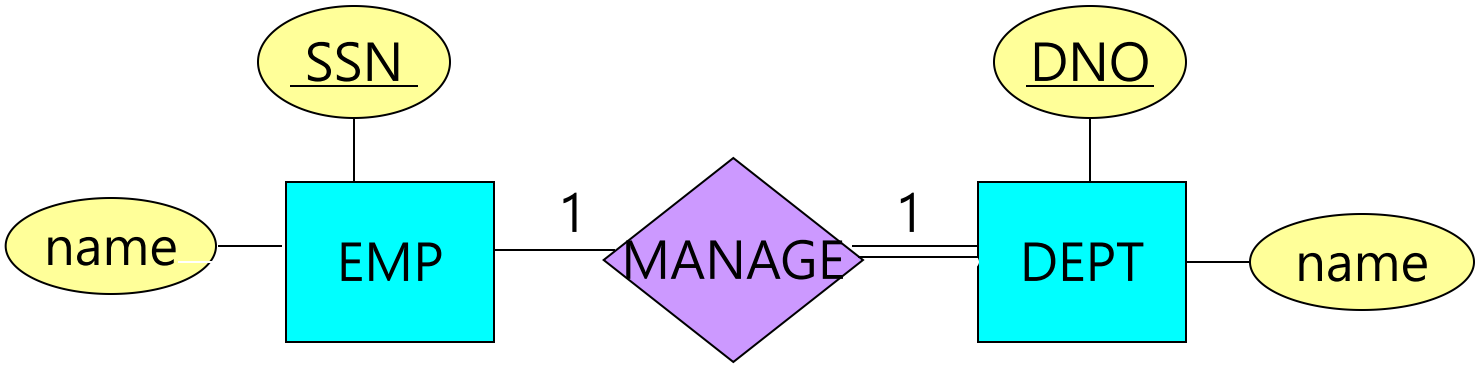
Case 1. **Foreign Key** : Two Relations
- The one side (say, **S**) is total and the other side (say, **T**) is partial.
- Include as **FK** in relation **S** the **PK** of relation **T**.

Case 2. **Merge relations** : Single Relation
- Both sides (say, **S** and **T**) are total.
- Merge two relations **S** and **T** and their relationship into a single relation.

Case 3. **Create a new relation** : Three Relations
- Both sides (say, **S** and **T**) are partial.
- Create a new relation **R** by including the **PK**s of the relations **S** and **T**.

1 : 1 Relationship

Case 1: The only one side (i.e., DEPT) is total:

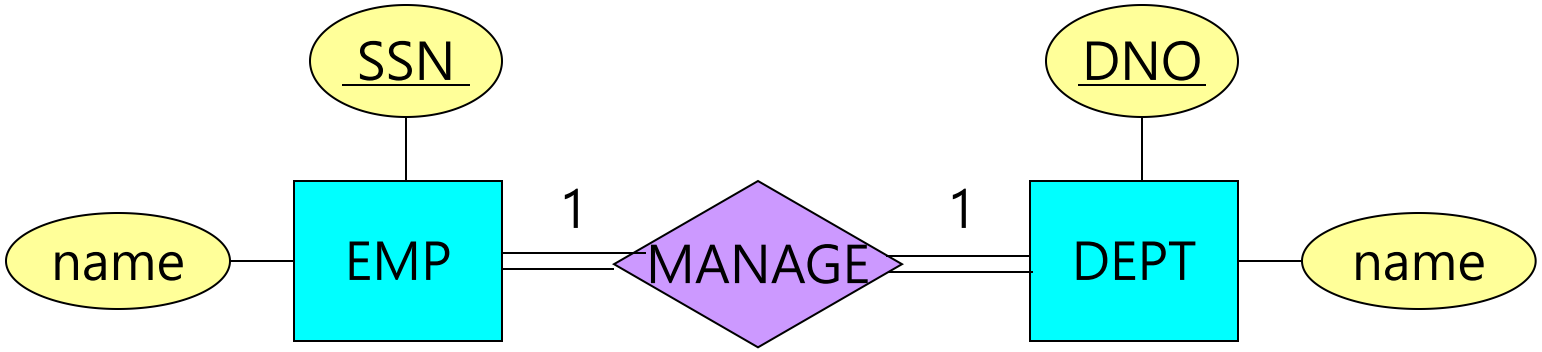


EMP (SSN, name)

DEPT (DNO, name, SSN)

1 : 1 Relationship

Case 2: The both sides are total:

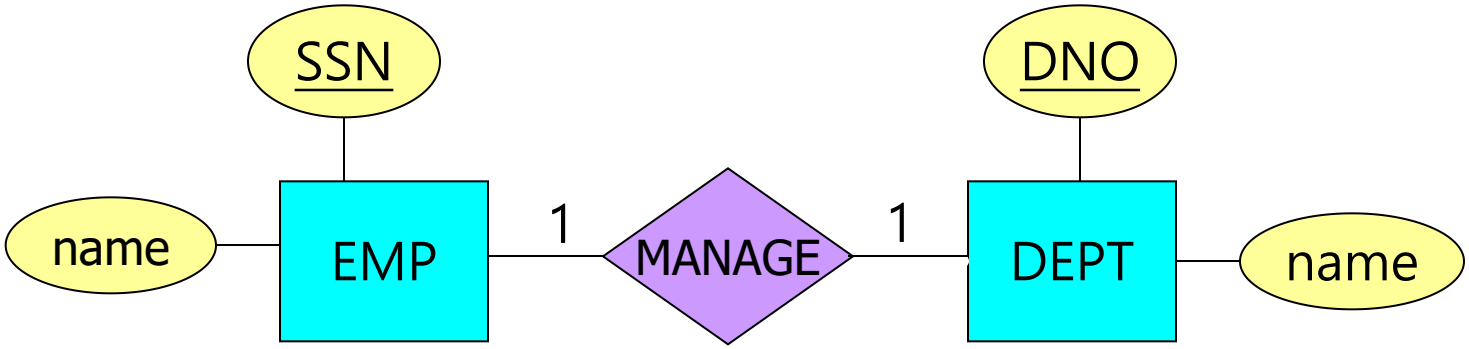


MANAGE (SSN, ename, DNO, dname)

- We merge into a single relation MANAGE.
- Either SSN or DNO (in MANAGE) is **PK**; No foreign key!
- In this case, we must rename 'name' as 'ename' and 'dname'.

1 : 1 Relationship

Case 3: The both sides are partial:



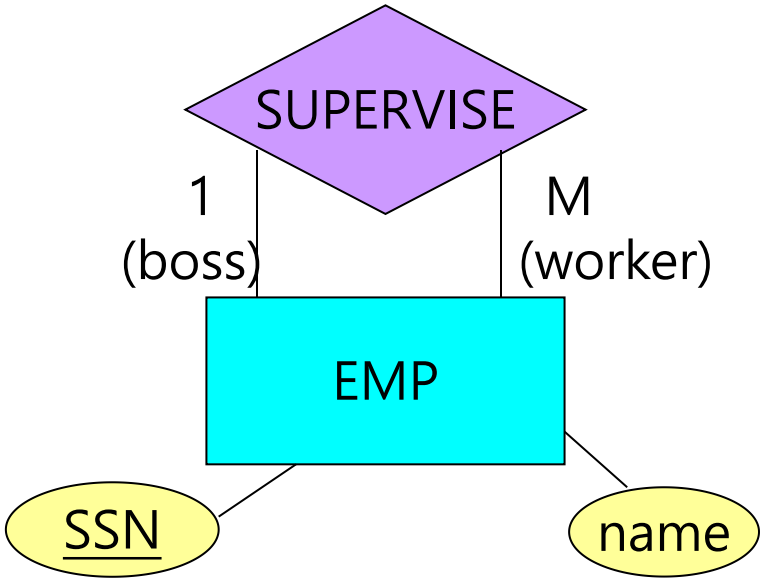
EMP (SSN, name)

DEPT (DNO, name)

MANAGE (SSN, DNO)

- Either SSN or DNO (in MANAGE) is **PK**.

Recursive 1 : M Relationship

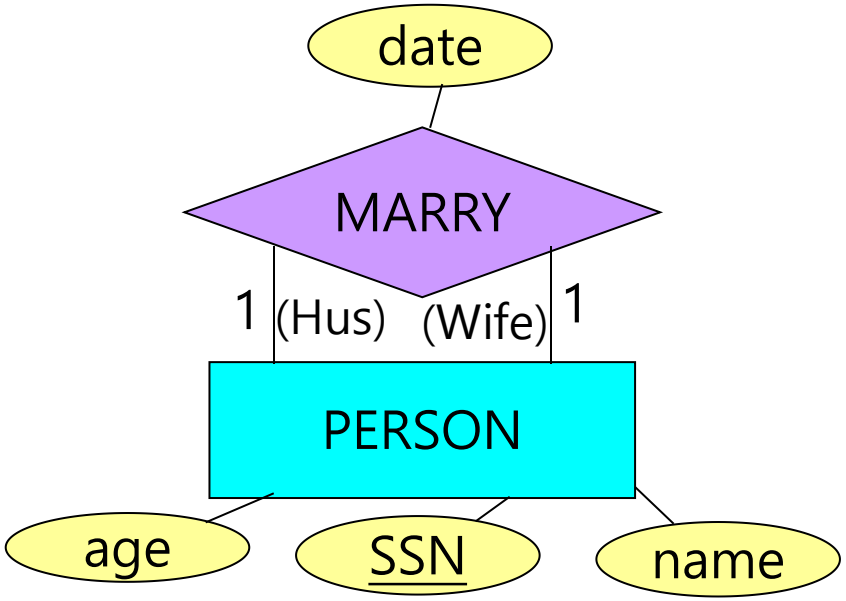


SSN	name	Super-SSN
11111	bob	33333
22222	joe	33333
33333	ann	44444
44444	dan	55555
55555	eve	null

EMP (SSN, name, Super-SSN)

- Most employees may have his/her boss.
- Include the **PK** 'SSN' (1-side) of EMP relation itself as **FK**.
- We must rename the FK 'SSN as "super-SSN.

Recursive 1 : 1 Relationship



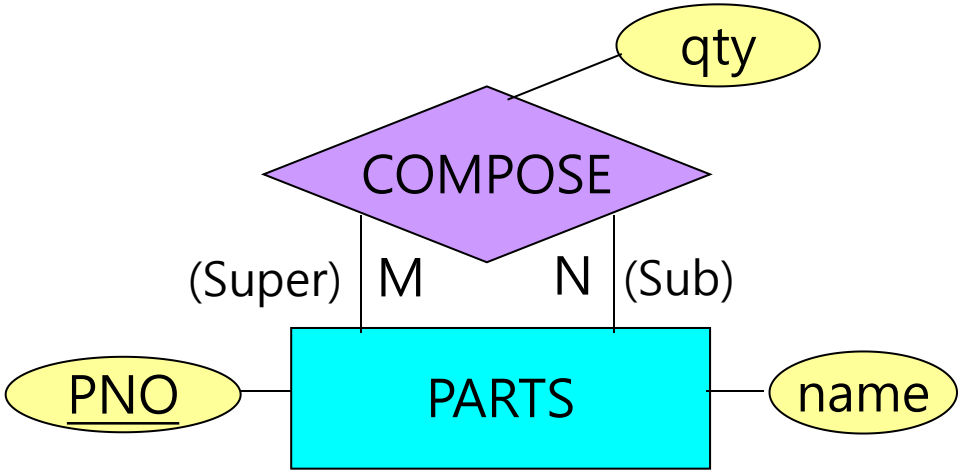
H-SSN	W-SSN	date
12345	34567	1/15/88
23456	45678	2/15/88
56789	67890	3/15/88

PERSON (SSN, name, age)

MARRY (H-SSN, W-SSN, date)

- Many persons may not marry; Create a new relation MARRY;
- Include both SSNs as FKs; Rename each SSN as H-SSN, W-SSN;
- PK is either {H-SSN} or {W-SSN}; What if remarriage allowed?

Recursive M : N Relationship



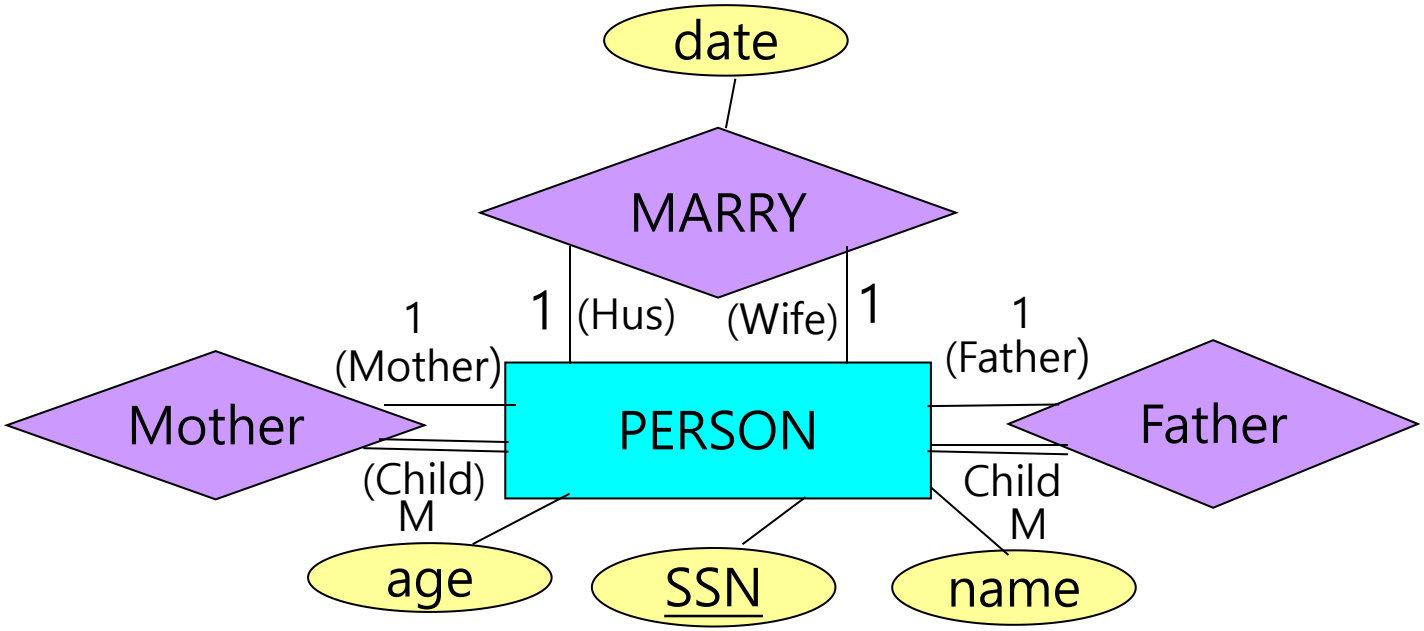
Sup-PNO	Sub-PNO	qty
A	B	15
A	C	20
B	D	15
B	E	15
F	C	10

PARTS (PNO, name)

COMPOSE (Sup-PNO, Sub-PNO, qty)

- Create a new relation COMPOSE;
- Include both PNO's as FK's; Rename PNO as Sup-PNO
- and Sub-PNO; PK is {Sup-PNO, Sub-PNO}

Recursive Relationship: Exercise



PERSON (SSN, name, age, F-SSN, M-SSN)

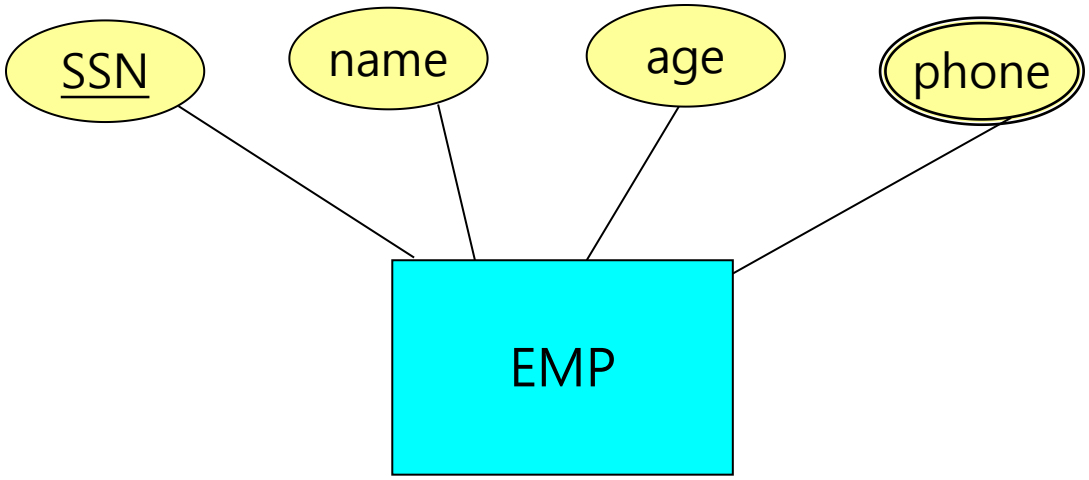
MARRY (H-SSN, W-SSN, date)

- 홍길동의 친할아버지의 이름을 구하라.
- 홍길동의 장인어른의 이름을 구하라.
- 홍길동의 외손자(들)의 이름을 구하라
- . . . . . .

Multi-Valued Attributes

- For each multi-valued attribute **A**, create a new relation **R**.
- **R** includes the attribute **A**, plus the **PK** **K** of the relation **S** that represents the entity type including **A**. Then, remove **A** from the relation **S**.
- The **PK** of **R** is {**A**, **K**} where K is **FK**.
- If the multi-valued attribute is composite, we include its simple components.

Multi-Valued Attributes



EMP (SSN, name, age)

EMP-PHONES (SSN, phone)

Ternary Relationship

- For each ternary relationship **R**, create a new relation **S**;
- Include as **FK** in **S** the **PK**s of each relations that represent the participating entity types.
- Also, include any simple attributes of ternary relationship (or simple components of composite attributes) as attributes of **S**.
- Relation **S**의 **PK**는 m : n : p relationship에서 m, n, p의 값에 따라 달리 설정됨.

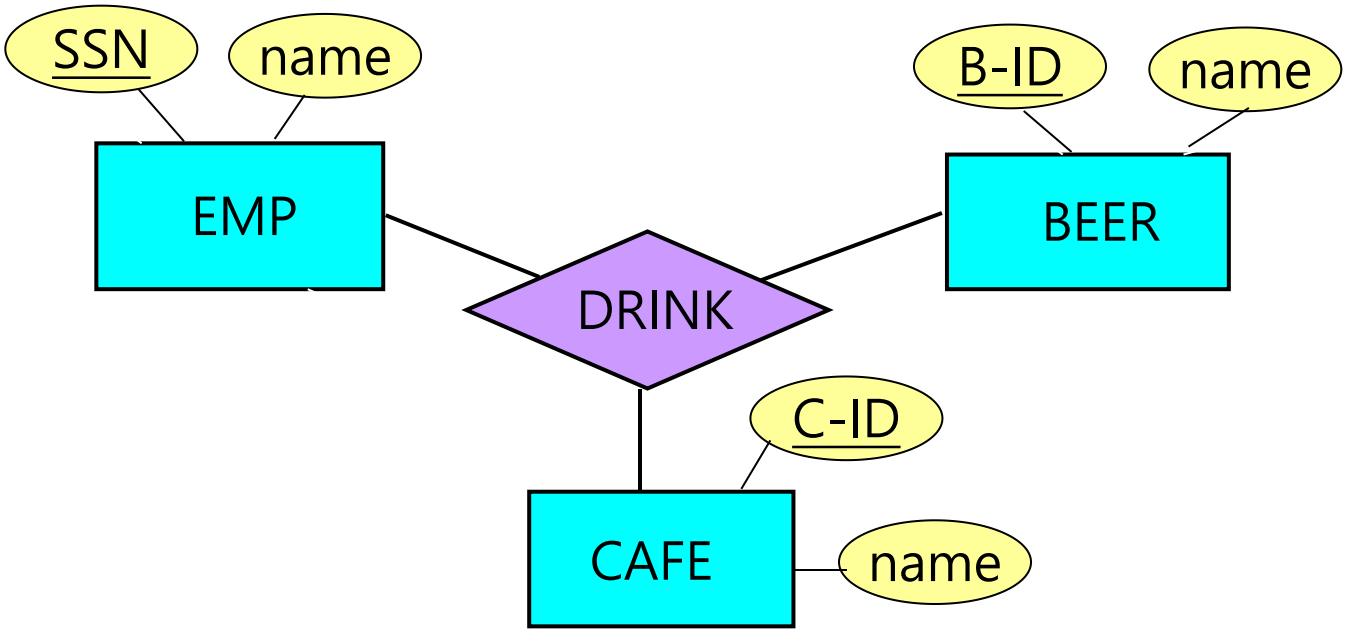
Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g. an arrow from *works_on* to *job* indicates each employee works on at most one job at any branch.
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean

    1. each *A* entity is associated with a unique entity from *B* and *C* or

    2. each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

Ternary Relationship : No Constraints



EMP (SSN, name)

BEER (B-ID, name)

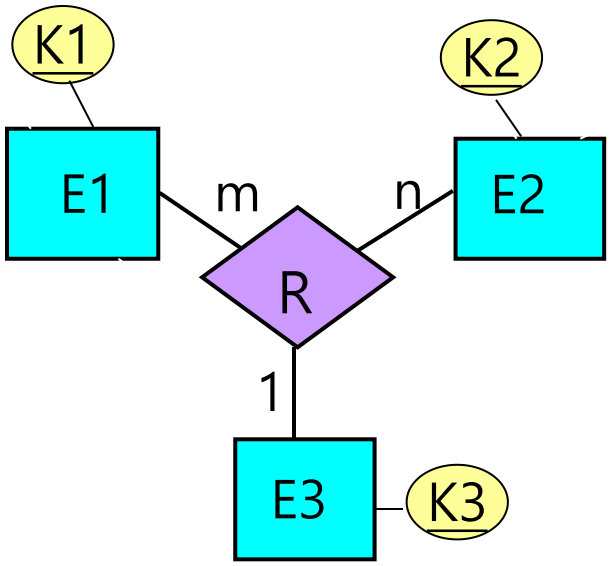
CAFE (C-ID, name)

DRINK (SSN, B-ID, C-ID, price)

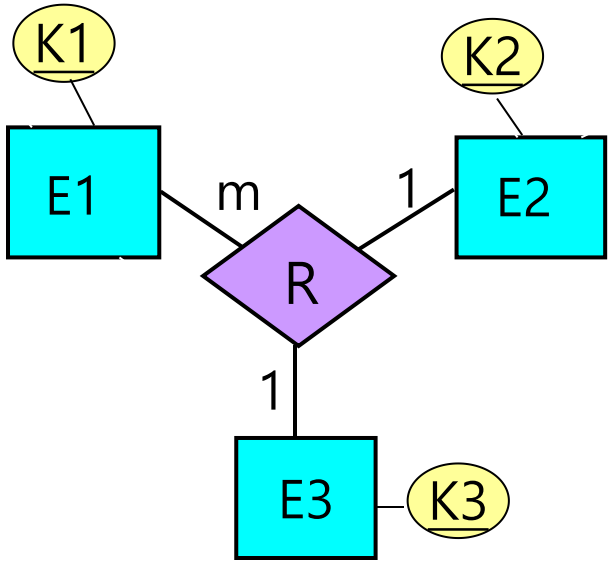
Ternary Relationship

- Find key(s) of each relationship R;

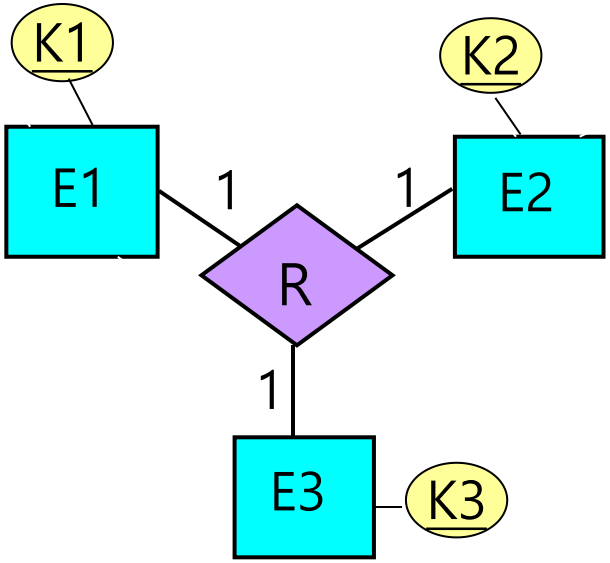
(A)



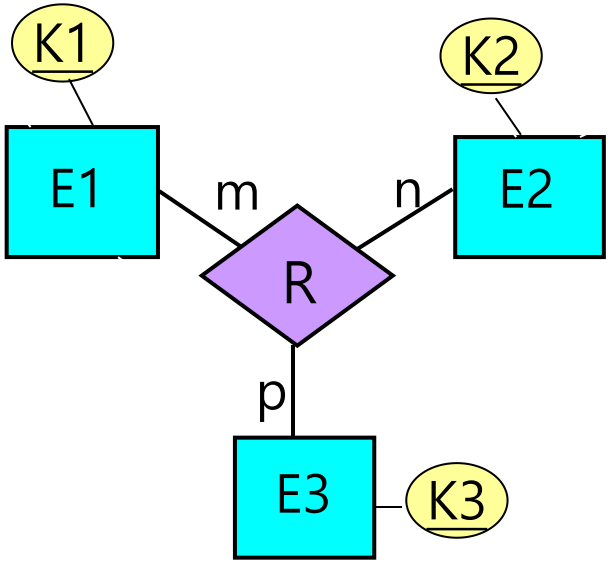
(B)



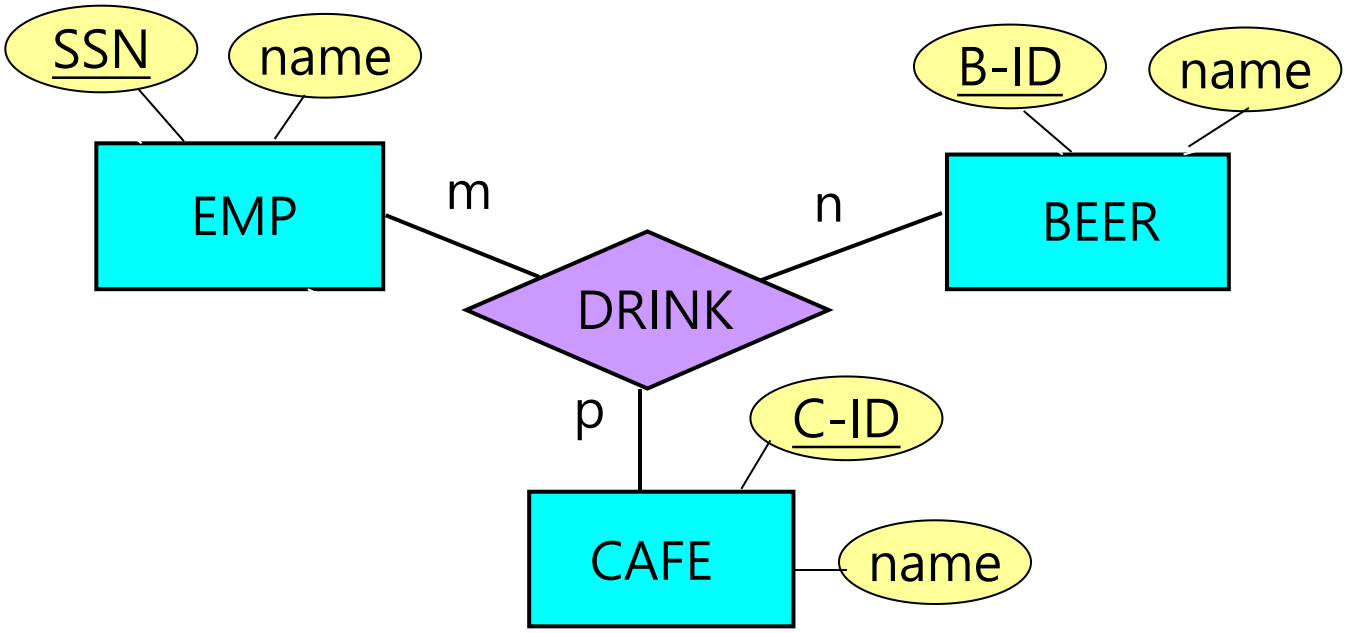
(C)



(D)

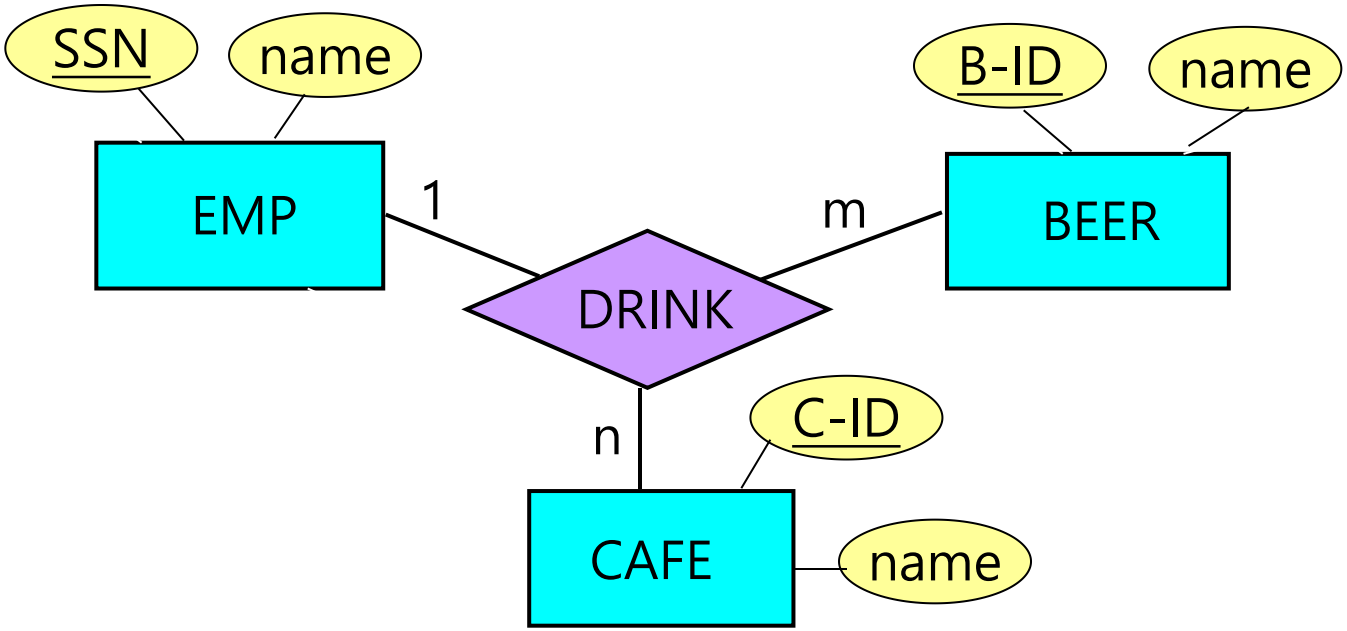


Ternary Relationship (m : n : p)



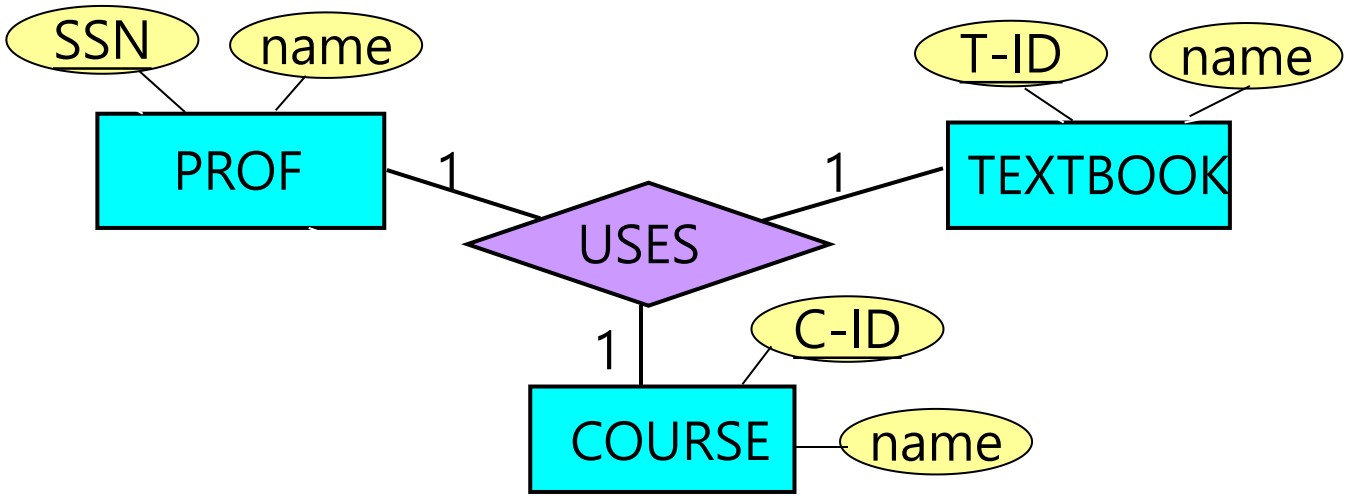
- Employees can drink different beers at any cafes;
- Each café has many employees drinking with many beers;
- **DRINK(SSN, B-ID, C-ID, price)**의 Key는?

Ternary Relationship (1 : m : n)



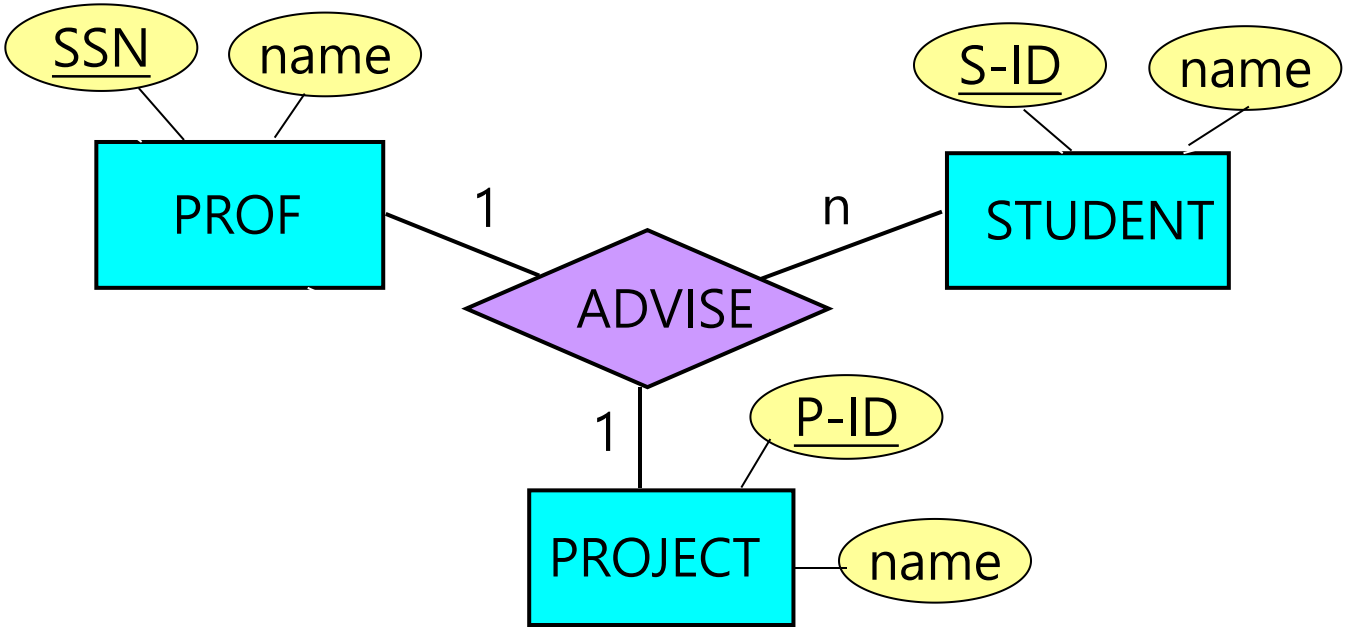
- For certain beer and cafe combination, only one employee exists; That means : only one employee drink a certain beer at a certain cafe.
- Each (b, c) combination uniquely determine a single employee.
- **DRINK(SSN, B-ID, C-ID, price)**의 Key는?

Ternary Relationship (1 : 1 : 1)



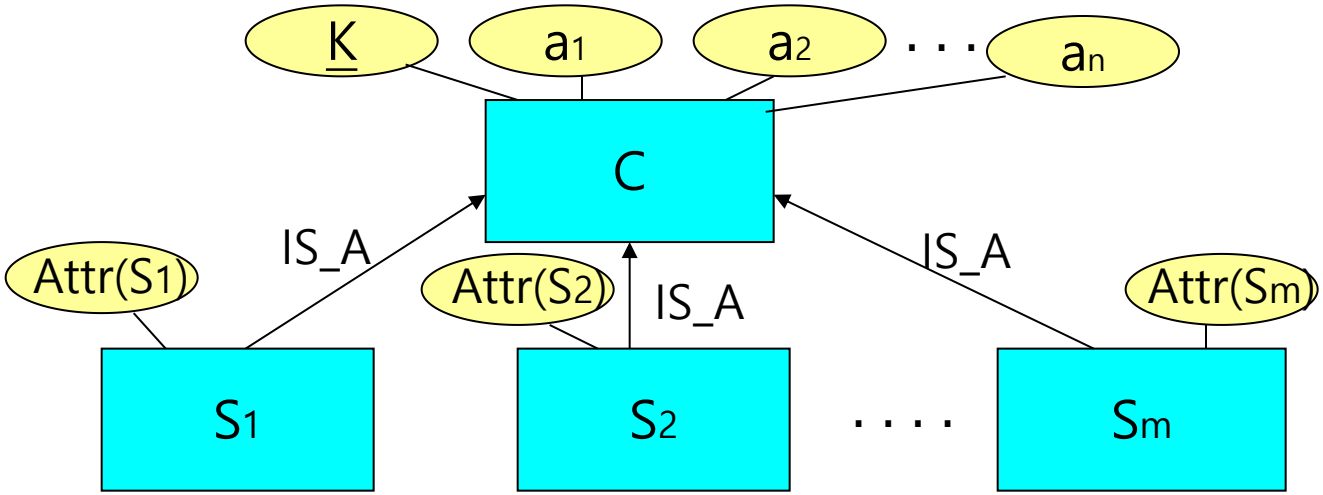
- For certain {professor, textbook} pair, only one course exists;
  (Only one course is offered for certain professor and certain textbook.)
- For certain {textbook, course}, only one professor exists;
  (Only one professor teaches for certain course and certain textbook.)
- For certain {professor, course}, only one textbook exists;
  ((Only one textbook is used for certain professor and certain course.)
- **USES (SSN, T-ID, C-ID, room)**의 Key는?

Ternary Relationship (1 : n : 1)



- What the above ERD means? Explain!
- **ADVISE (SSN, S-ID, P-ID)**의 Key는?

IS-A Relationship



- Superclass C 와 각 subclass $\{S_1, S_2, \ldots, S_m\}$ 간에 IS_A 관계가 존재.
- C의 attribute들을 $\{K, a_1, \ldots, a_n\}$, 각 subclass Si 에만 있는 고유의 attribute들을 Attr(Si)로 표기. (단, K는 C의 key)
- 위의 ER schema를 relation으로의 mapping에 3 가지 방식이 있음.

Option A : Single Super/Multiple Subs

- Create a relation **L** for superclass **C** with

  Attr(**L**) = {K, a_1, . . . , a_n} (단, PK(**L**) = K)

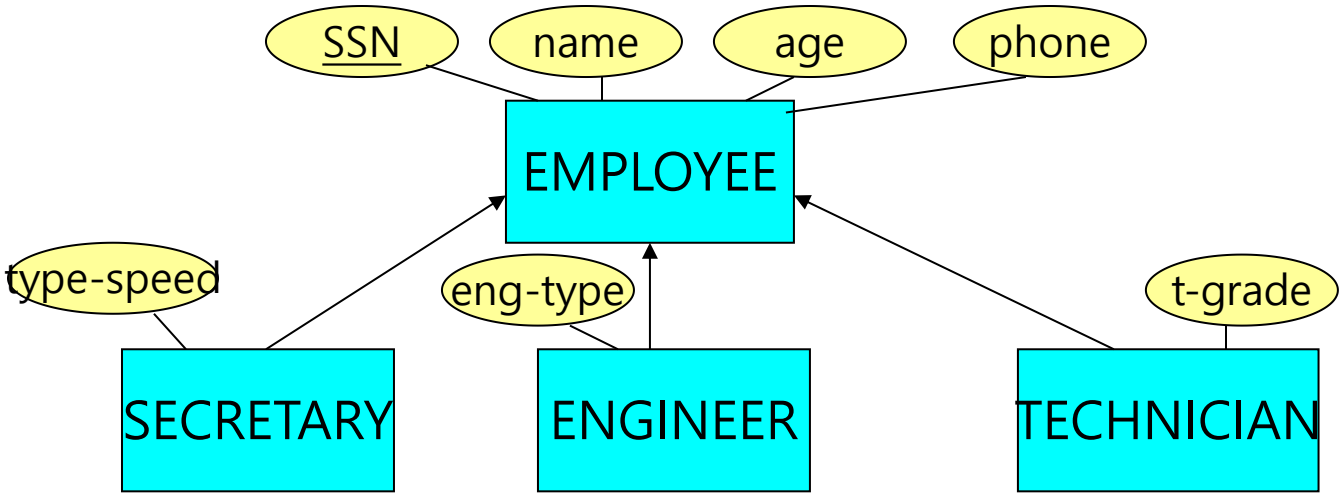
- Create a relation L_i for each subclass S_i with

  Attr(L_i) = {K} U Attr(S_i) (단, PK(L_i) = K)

- 각 subclass relation L_i는 superclass relation **L**과 NATURAL JOIN을 통해 attribute들을 inherit함.

- 이 option은 (Total , Partial, Disjoint , Overlap) 4 가지 조합의 제약조건에 모두 사용될 수 있음.

Option A : Single Super/Multiple Subs



EMPLOYEE (SSN, name, age, phone)

SECRETARY (SSN, type-speed)

ENIGINEER (SSN, eng-type)

TECHNICIAN (SSN, t-grade)

Option B : Multiple Subs Only

- Create a relation L_i for each subclass S_i with

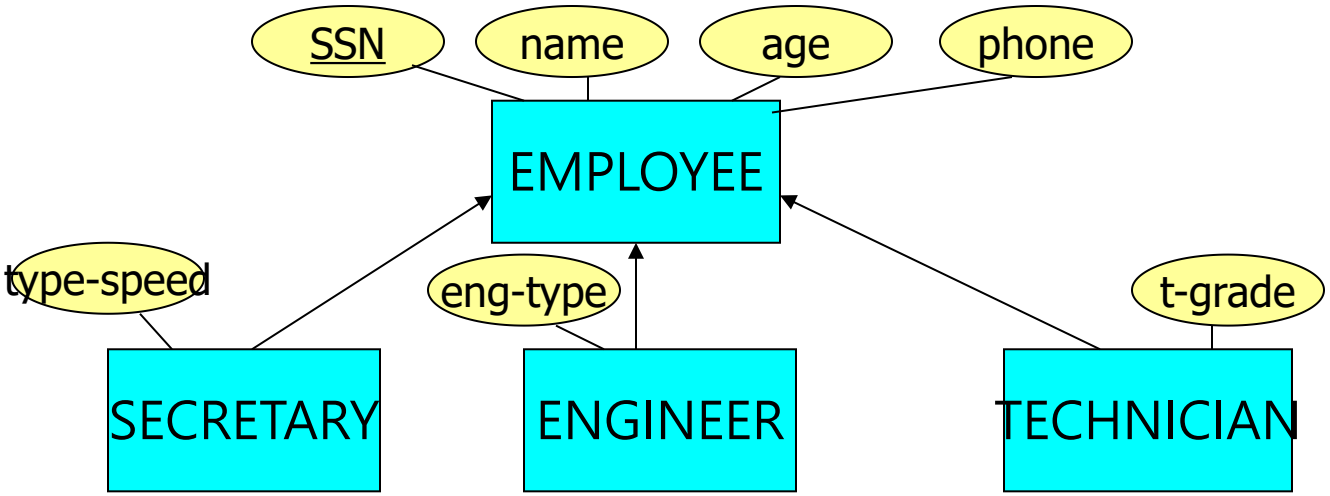
  Attr(L_i) = Attr(S_i) U {K, a_1, . . . , a_n} (단, PK(L_i) = K)

- Subclass들만 존재; Super의 attribute들은 이미 inherit 했음.

- Superclass C는 각 L_i relation들을 OUTER UNION하면 복구됨.

- 이 option은 total /disjoint 제약조건에만 사용될 수 있음.
  (만약 partial 참여인 경우, 어떠한 subclass에도 속하지 않은 entity의 정보는 유실됨.)

Option B : Multiple Subs Only



SECRETARY (SSN, name, age, phone, type-speed)

ENIGINEER (SSN, name, age, phone, eng-type)

TECHNICIAN (SSN, name, age, phone, t-grade)

Option C : Single Super Only

- Create a single relation **L** with

$$Attr(\mathbf{L}) = \{K, a_1, \ldots, a_n\} \cup Attr(\mathbf{S_1}) \cup, \ldots, \cup Attr(\mathbf{S_m})$$

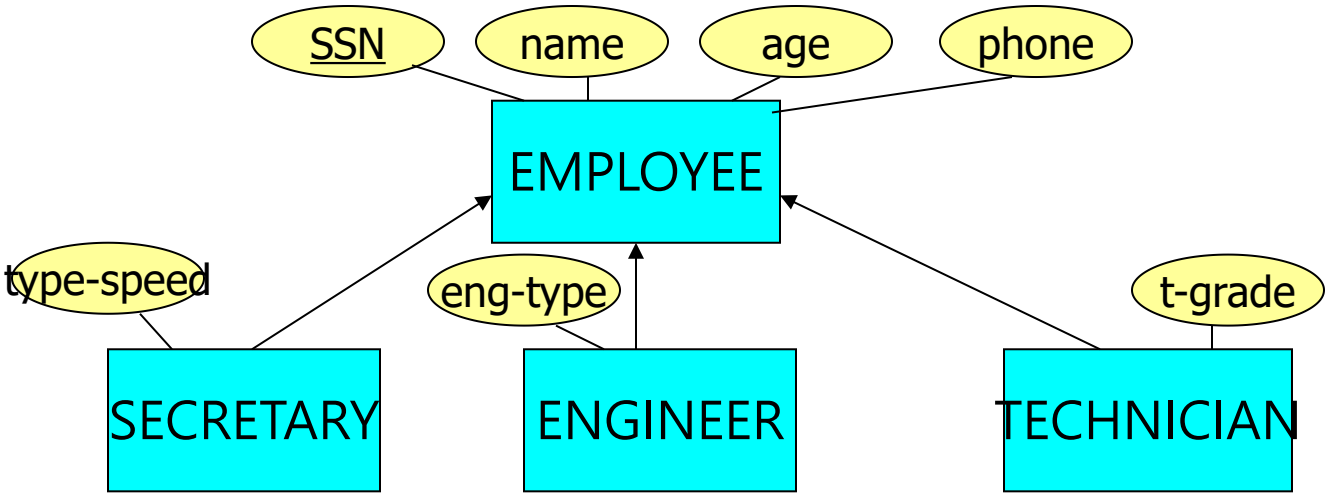
- 각 subclass는 relation **L**에 SELECT 연산으로 복구됨.

  (JOIN / UNION이 필요 없으므로 검색 시간이 효율적)

- Null 값들이 많이 발생; 특히 각 subclass가 그 자신만의 고유의 attribute들이 많은 경우.

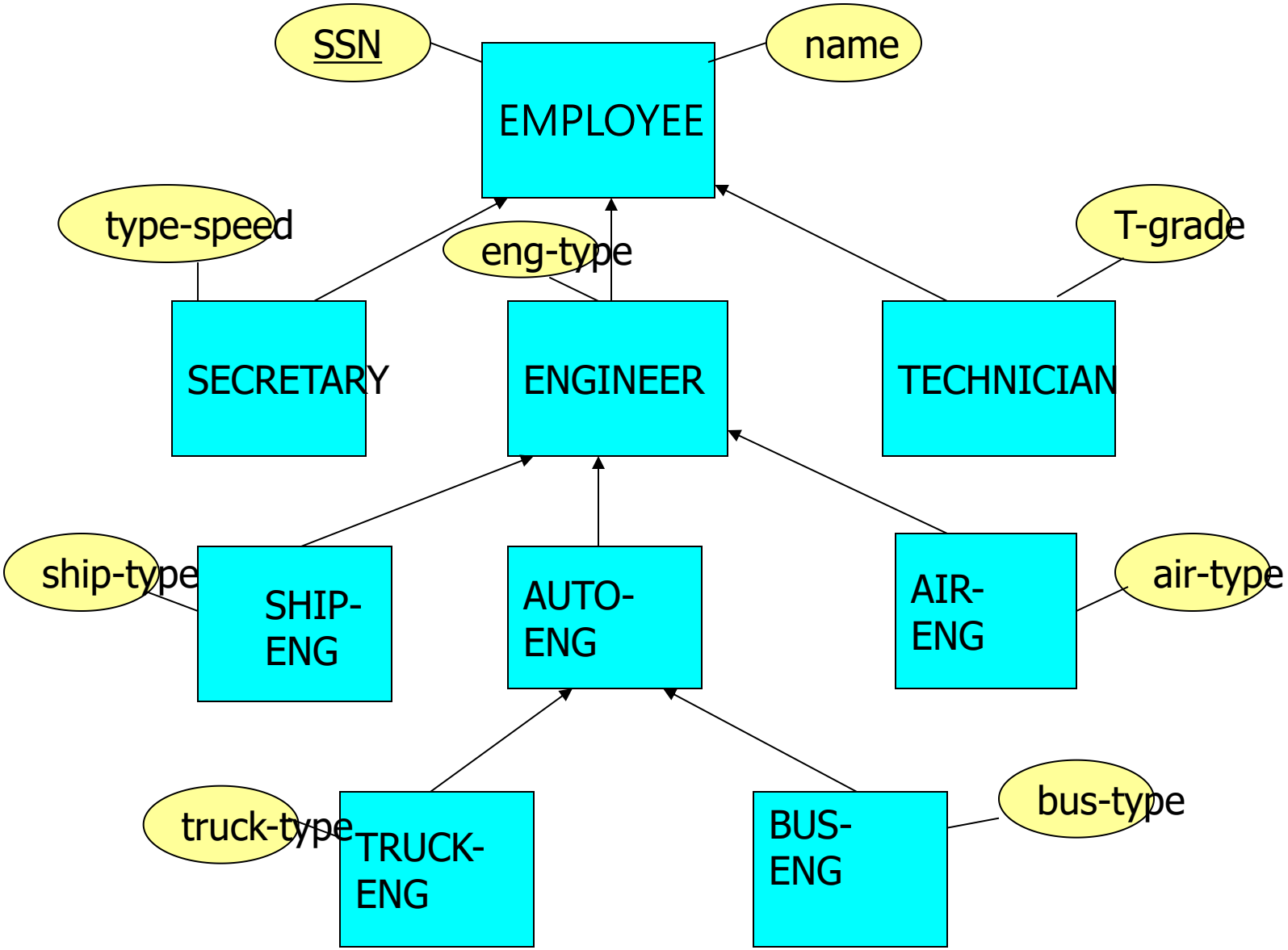
- 이 option은 disjoint 제약조건에만 사용되므로 매우 제한적임.

Option C : Single Super Only



EMPLOYEE(SSN, name, age, phone, type-speed, eng-type, t-grade)

Show Relational Schema : Exercise



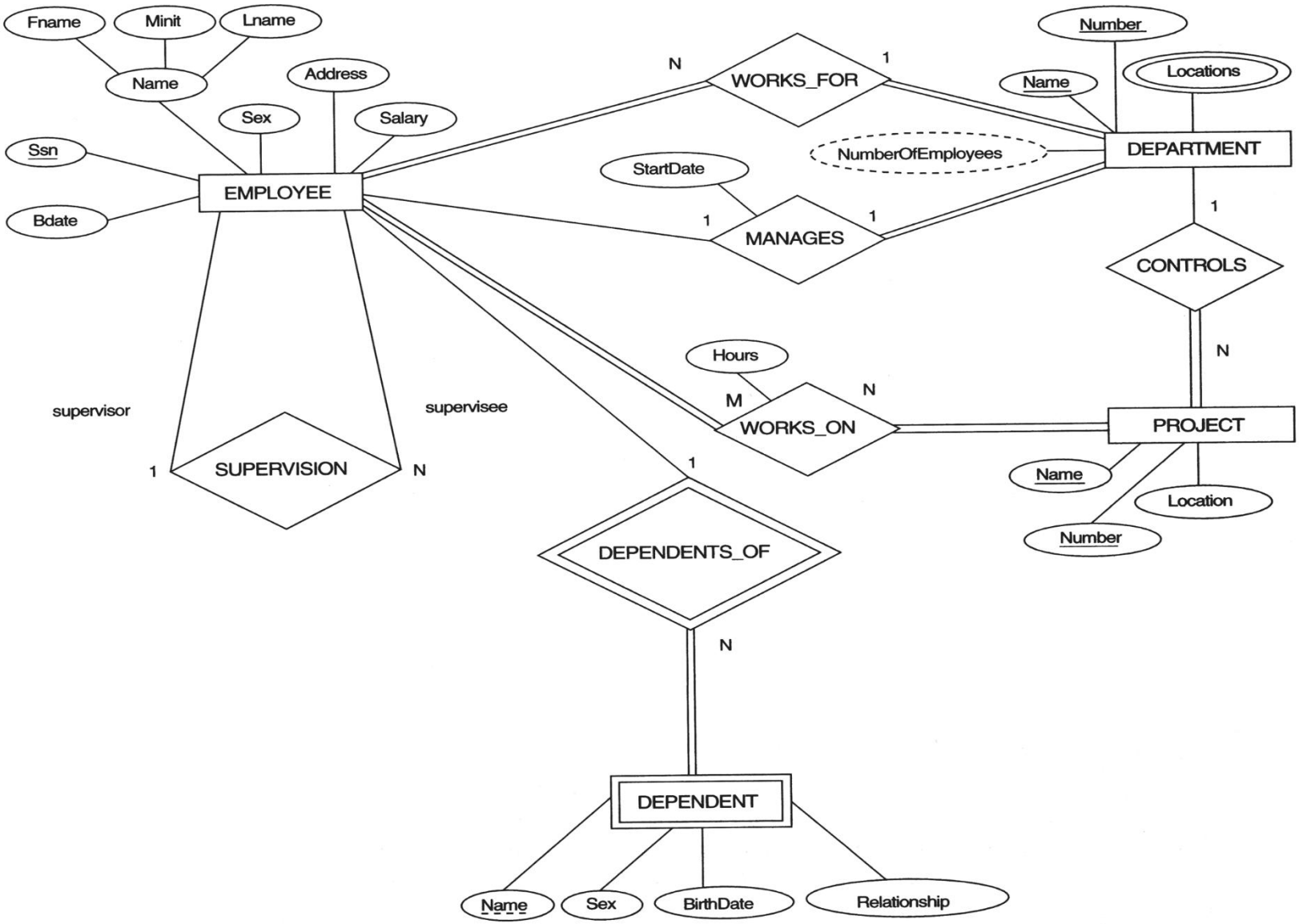
Summary of Mappings

- Entity Type
- Weak Entity Type
- Multi-valued attribute
- Composite Attribute
- 1 : N or 1 : 1 Relationship
- M : N Relationship
- Recursive Relationship
- Ternary Relationship
- IS_A Relationship

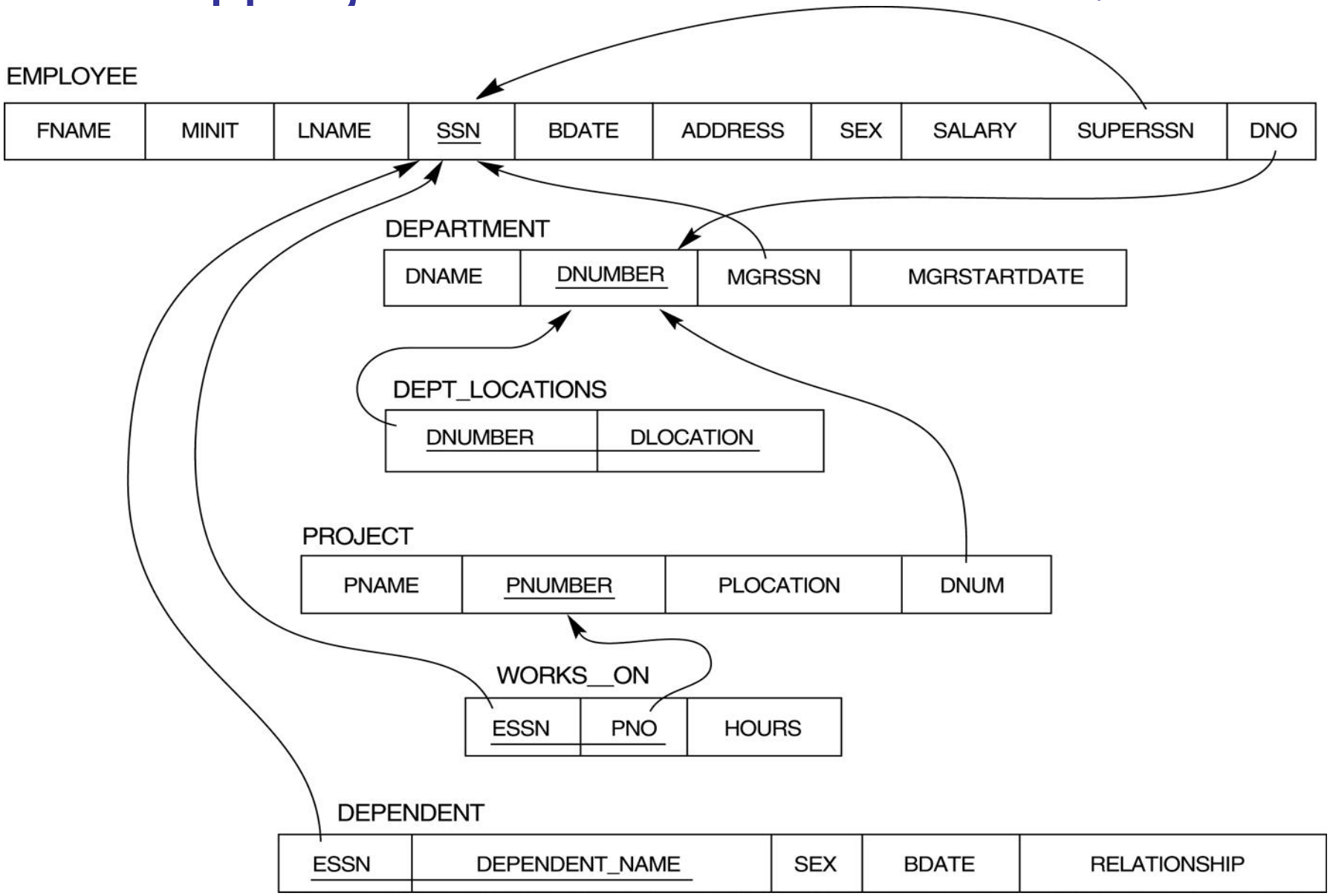
고려 사항

- ✓ Total 혹은 Partial?
- ✓ Foreign Key 설정?
- ✓ Primary Key 설정?
- ✓ Name 변경?
- ✓ Null value 발생?
- ✓ 검색 시간(= Join 회수 등)?

Show Relational Schema : Exercise



Mapping Result : Relational schema



Show Relational Schema : Exercise