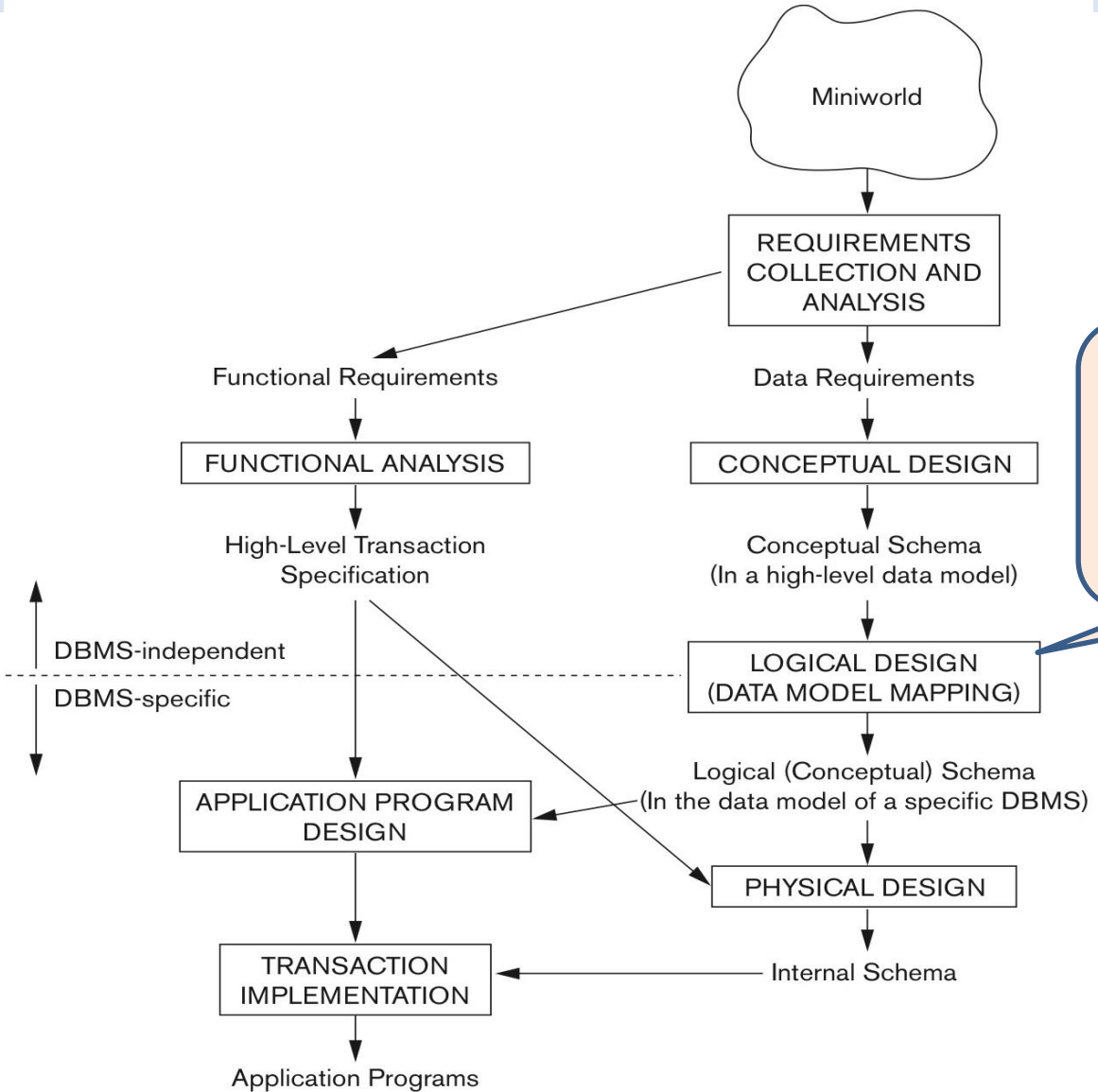
# CMPE 226 Database Systems
# ***ER-to-Relational Mapping***

Instructor: Kong Li

# Outline

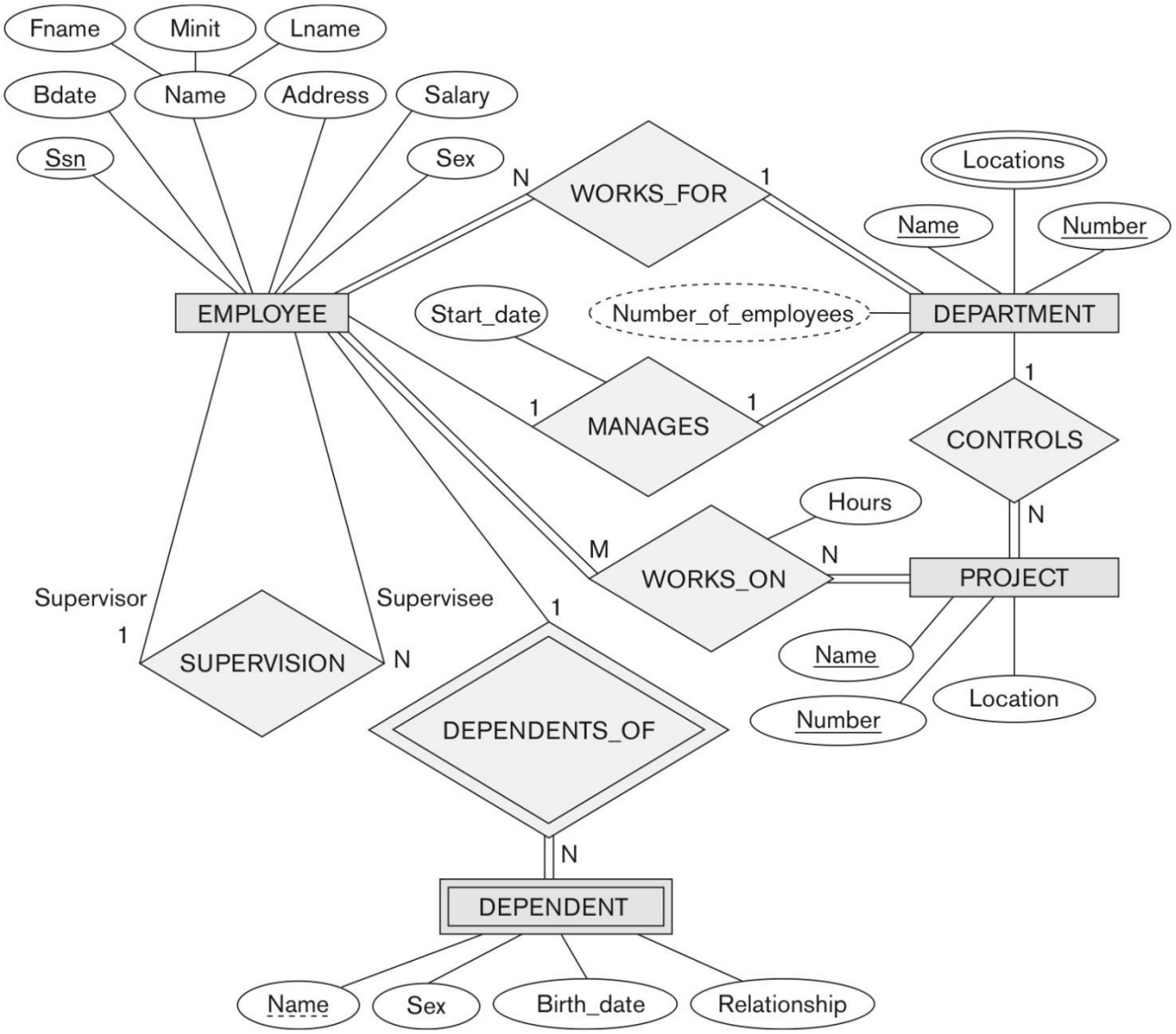
- ERD → Relational Database schema
  - Step 1 ~ 7
- EERD → Relational Database schema
  - Step 8 & 9
  - Ignore step 9

# ER-to-Relational Mapping

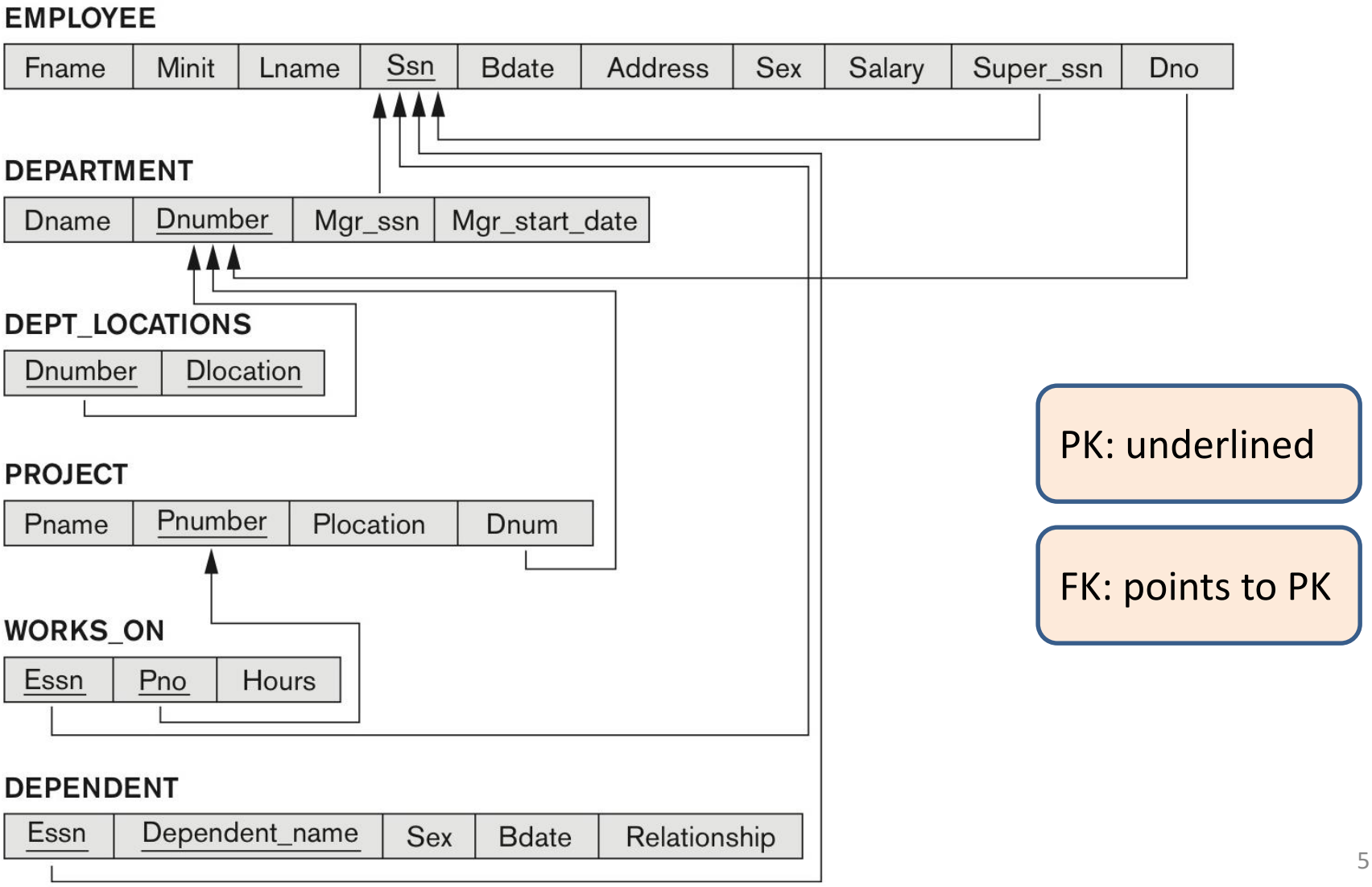



- ER-to-relational mapping: 7 steps
- EER-to-Relational mapping: 2 steps

# ER Diagram: COMPANY



# Result of mapping: COMPANY




PK: underlined

FK: points to PK

# Step 1: Regular Entity

- Create a relation *R* that include all simple attrs
  - PK: one of key attrs
  - *R* is called **entity relation**
    - Each tuple represents an entity instance

**EMPLOYEE**

| Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary |
|---|---|---|---|---|---|---|---|

**DEPARTMENT**

| Dname | Dnumber |
|---|---|

**PROJECT**

| Pname | Pnumber | Plocation |
|---|---|---|

# Step 2: Weak Entity

- Create a relation *R* that includes all simple attrs
  - PK: {PK of owner entity, partial key of weak entity}
    - PK of owner as FK attr of *R*
  - If weak entity type $E_2$'s owner is also weak entity type $E_1$, $E_1$ should be mapped before $E_2$

**DEPENDENT**

| Essn | Dependent_name | Sex | Bdate | Relationship |
|---|---|---|---|---|

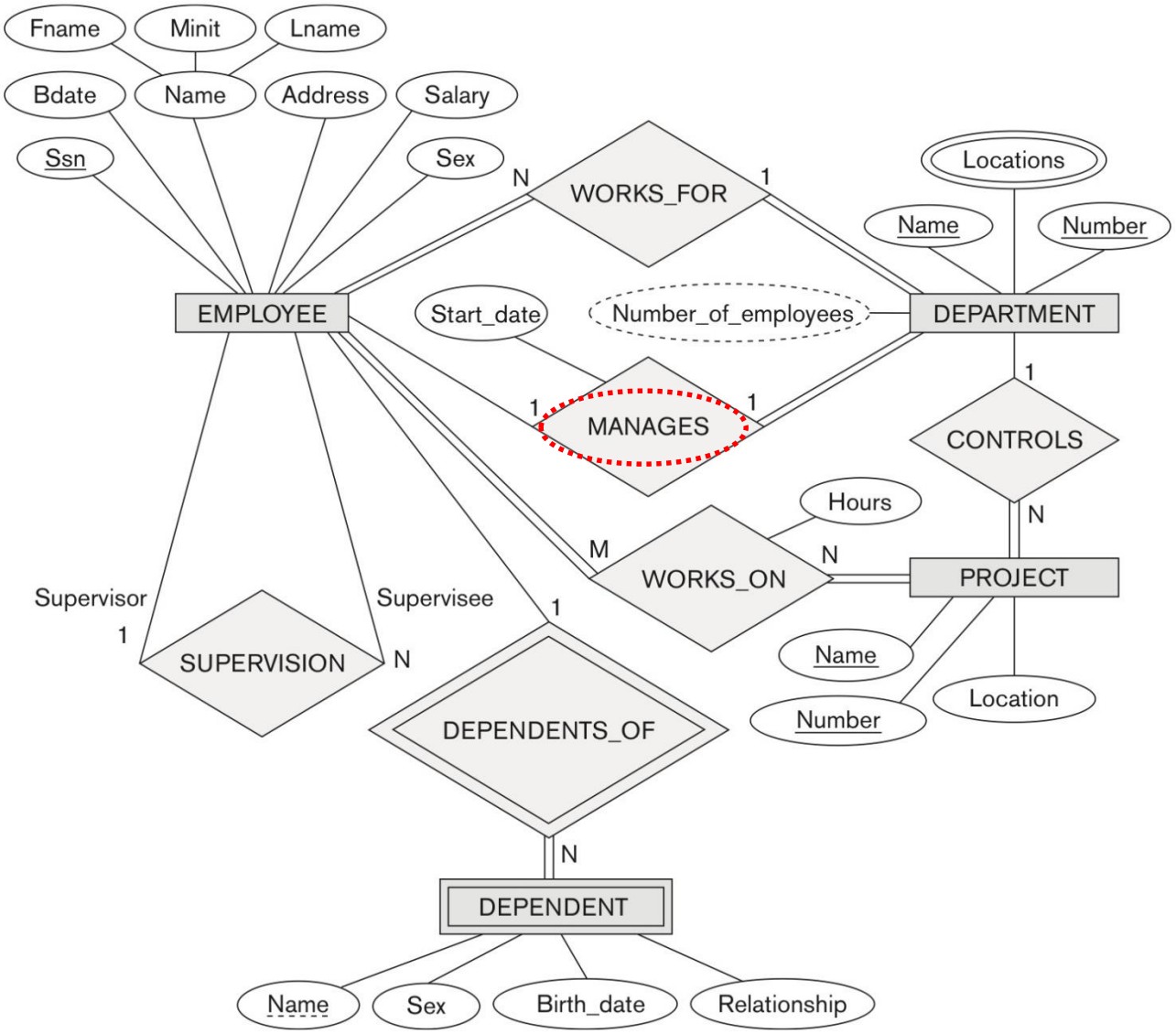
  - FK: CASCADE option for ON DELETE or ON UPDATE
- No need to map any identifying relationship

If PK of T is (a,b),
then FK: S.(a,b) to T.(a,b);
separate FKs are not correct

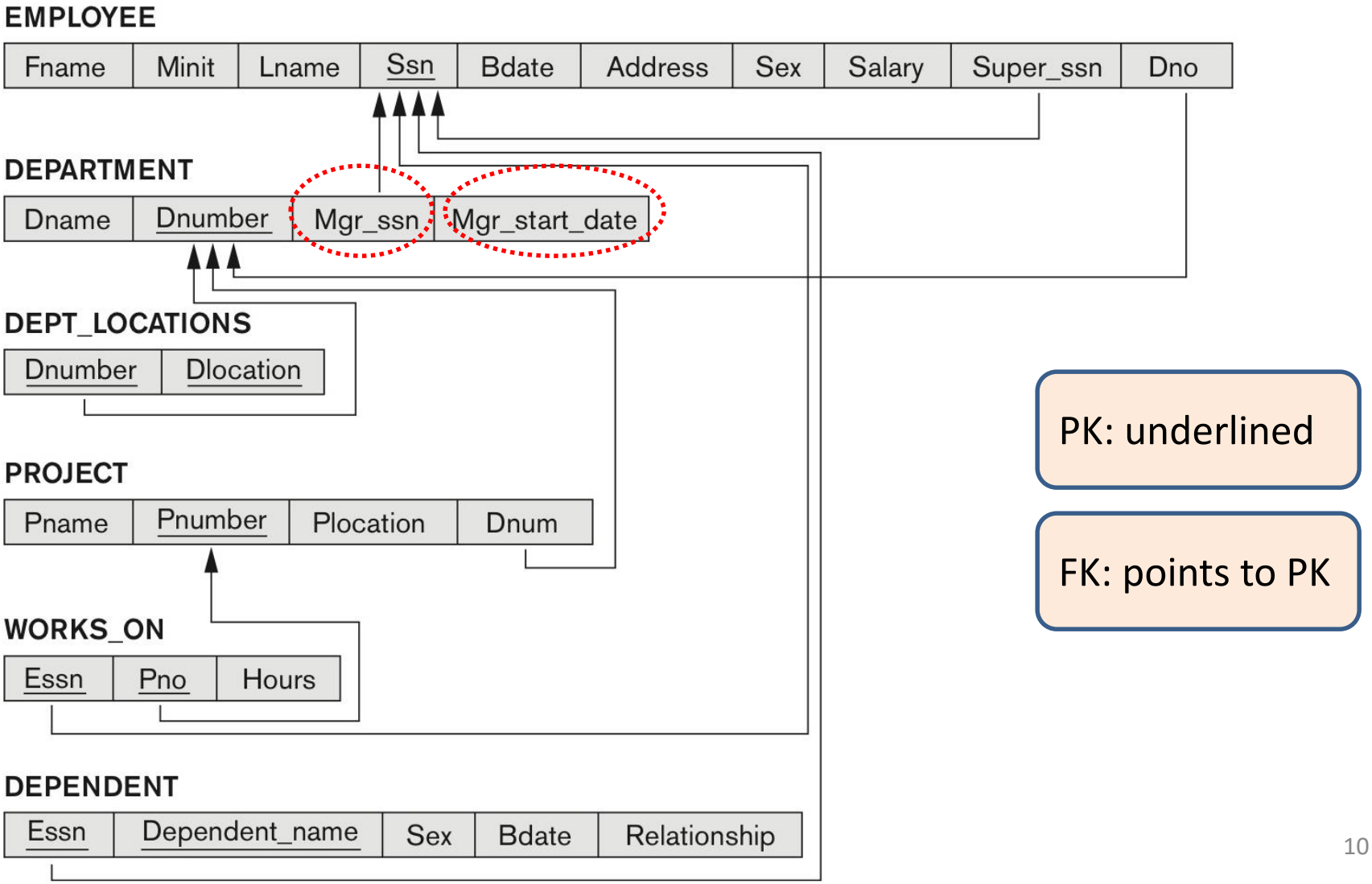
# Step 3: Binary 1:1 Relationship

- ***S – R – T***
  - *R*: binary 1:1 relationship type
  - *S* and *T*: entity types participating in *R*
- Approach #1: FK approach - recommended
  - Modify the relation *S* that is total participation in *R*
    - Include PK of *T* as FK in *S*
    - Include simple attrs of *R* in *S*
  - Ex: EMPLOYEE – manage - DEPARTMENT
    - DEPARTMENT.Mgr_ssn: FK to EMPLOYEE.Ssn
    - Start_date of relationship: DEPARTMENT.Mgr_start_date
  - Q: what if we choose *T*, not total participation?

# ER Diagram: COMPANY



# Result of mapping: COMPANY




PK: underlined

FK: points to PK

# Step 3: Binary 1:1 Relationship (cont'd)

- Approach #2: merged relation
  - If **both** *S* and *T* are total participation, merge *S*, *T*, *R* to one relation
    - Not recommended due to DB normalization (later)
- Approach #3: cross-reference or relationship relation
  - Create a **relationship relation** *R* (or lookup table)
    - Include PKs of *S* and *T*, as FKs in *R*
    - PK of *R*: one of the two FKs, and the other is unique
    - Cons: extra relation, and JOIN is needed for retrieval
  - Required for binary M:N relationships (later)

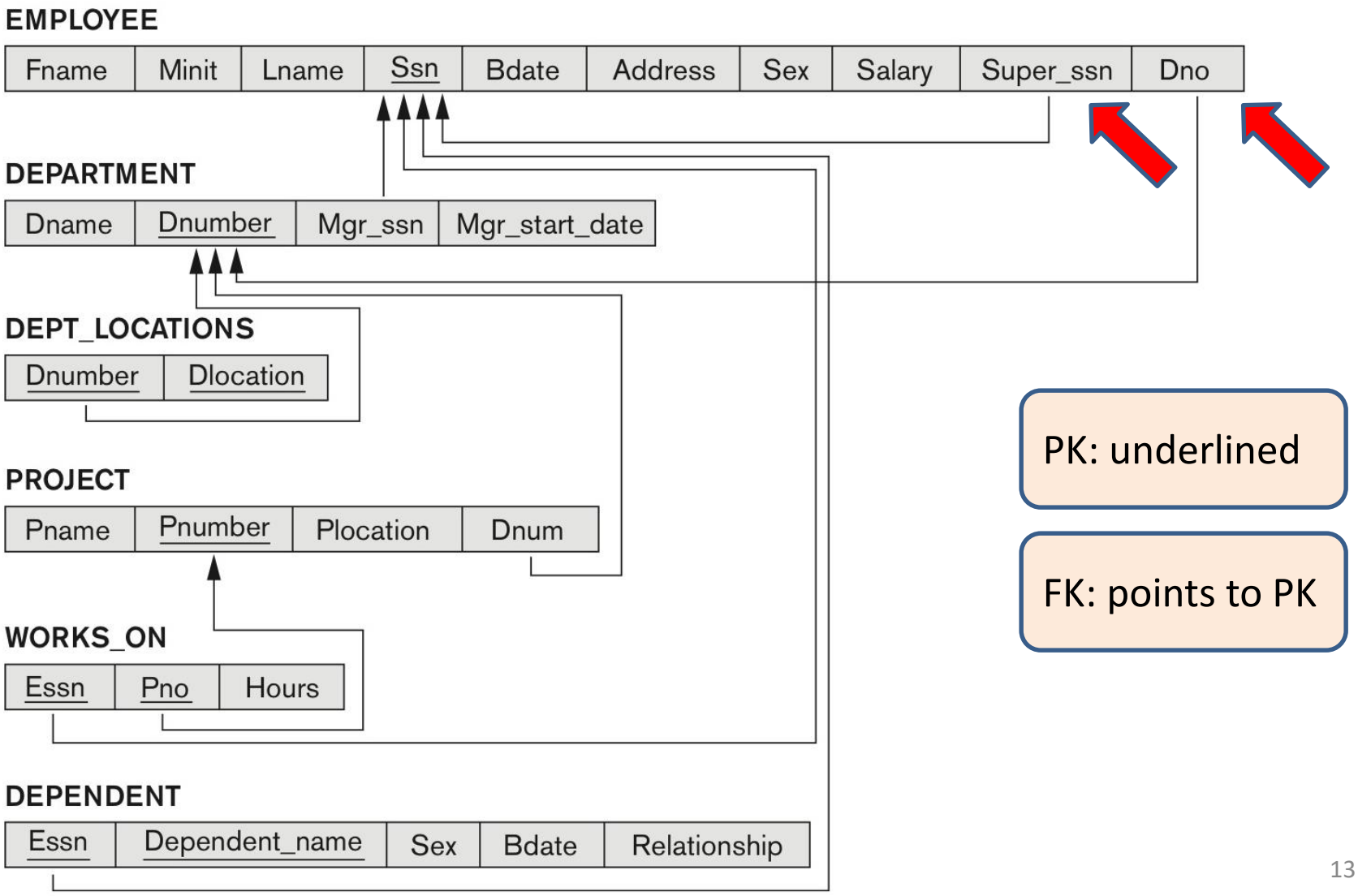
# Step 4: Binary 1:*N* Relationship

- ***S – R – T***
  - *R*: binary 1:N relationship type
  - *S*: entity type on N-side of *R, T*: entity types on 1-side of *R*
- Approach #1: modify the relation *S* (N-side of *R*)
  - Include PK of *T* as FK in *S*
  - Include simple attrs of *R* as attrs of *S*
  - Ex
    - WORKS_FOR: EMPLOYEE.Dno as FK to DEPARTMENT.Dnumber
    - CONTROLS: PROJECT.Dnum as FK to DEPARTMENT.Dnumber
    - SUPERVISION: EMPLOYEE.Super_ssn as FK to EMPLOYEE.Essn

If PK of T is (a,b),
then FK: S.(a,b) to T.(a,b);
separate FKs are not correct

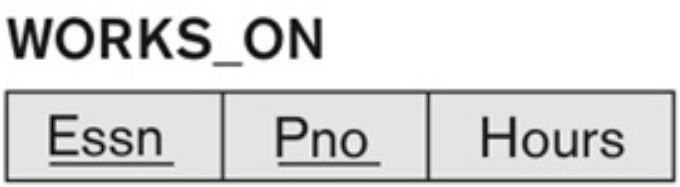
- Approach #2: create **relationship relation** *R*
  - Include PKs of *S* and *T*, as FKs in *R*
  - PK of *R*: PK of S (N side)

# Result of mapping: COMPANY



# Step 5: Binary *M*:*N* Relationship

- ***S – R – T***
- Create a new **relationship relation** *R*
  - Include PKs of *S* and *T* as FKs in *R*
    - FK propagate (CASCADE) option for ON UPDATE, ON DELETE
  - PK of *R* (composite PK): PK of *S*, PK of *T*
  - Include any simple attrs of *M*:*N* relationship type




If PK of T is (a,b),
then FK: S.(a,b) to T.(a,b);
separate FKs are not correct

  - 1:1 and 1:N relationships can be mapped to this

# Step 6: Multivalued Attributes

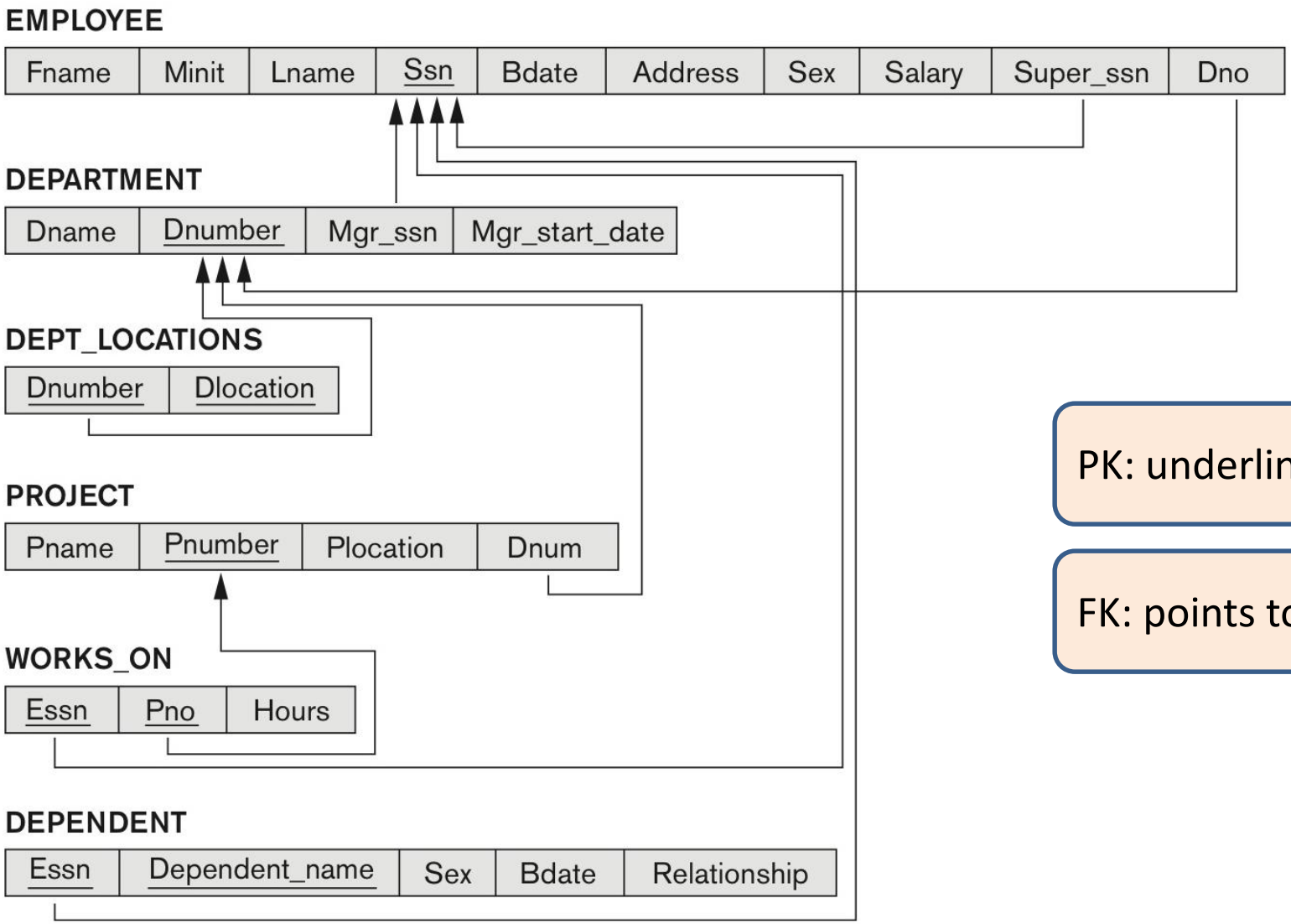
- Create a new relation *R* for multivalued attr *S.A*
  - Include attr of *A*, and PK of *S*
  - PK of *S* as FK in *R*
    - FK propagate (CASCADE) option for ON UPDATE, ON DELETE
  - PK of *R*: {*A*, PK of *S*}
  - If the multivalued attr is composite, include its simple components

**DEPT_LOCATIONS**

| Dnumber | Dlocation |
|---|---|

Q: what about a weak entity's multivalued attr? What is the proper FKs?

# Result of mapping: COMPANY

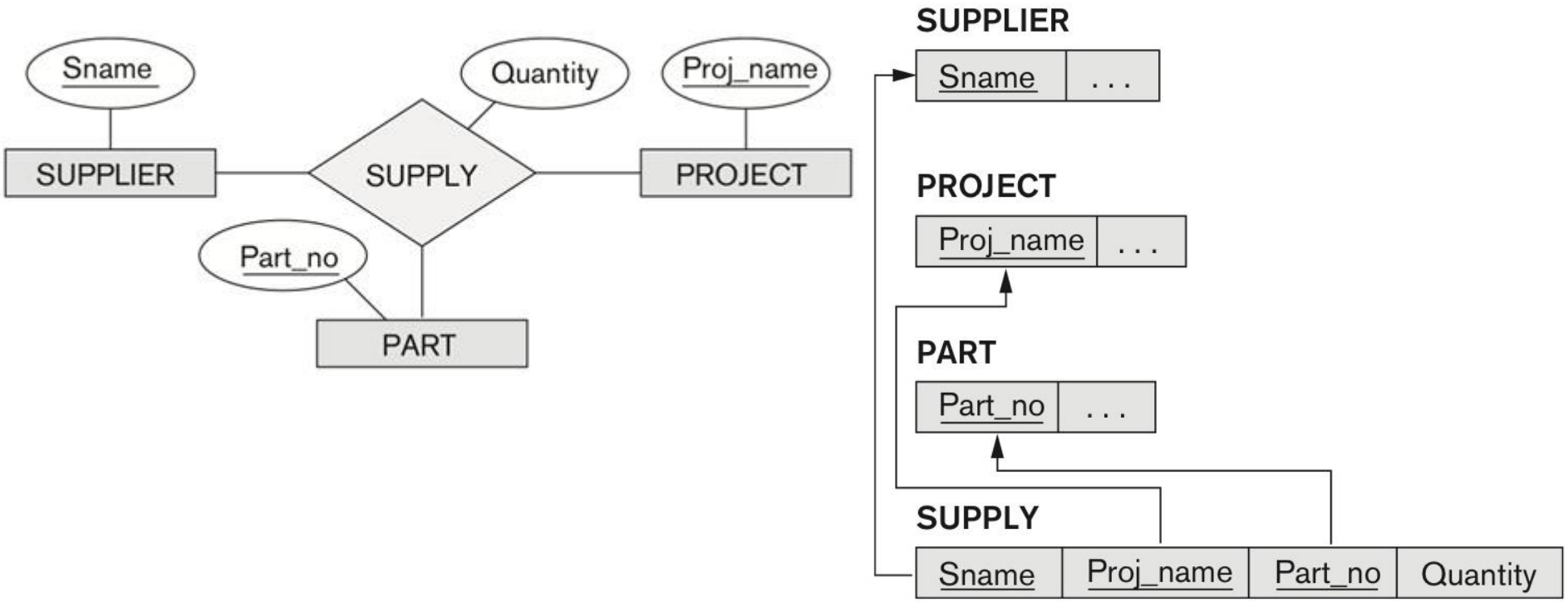



PK: underlined

FK: points to PK

# Step 7: *N*-ary Relationship Types

- Create a new relation *S* for each *n*-ary relationship *R*
  - Include PKs of participating entity types as FKs in *S*
  - PK of *S*: {PKs of participating entity types}
  - Include any simple attrs as attrs



# Summary of Mapping

**Table 9.1** Correspondence between ER and Relational Models

| ER MODEL | RELATIONAL MODEL |
|---|---|
| Entity type | *Entity* relation |
| 1:1 or 1:N relationship type | Foreign key (or *relationship* relation) |
| M:N relationship type | *Relationship* relation and *two* foreign keys |
| *n*-ary relationship type | *Relationship* relation and *n* foreign keys |
| Simple attribute | Attribute |
| Composite attribute | Set of simple component attributes |
| Multivalued attribute | Relation and foreign key |
| Value set | Domain |
| Key attribute | Primary (or secondary) key |

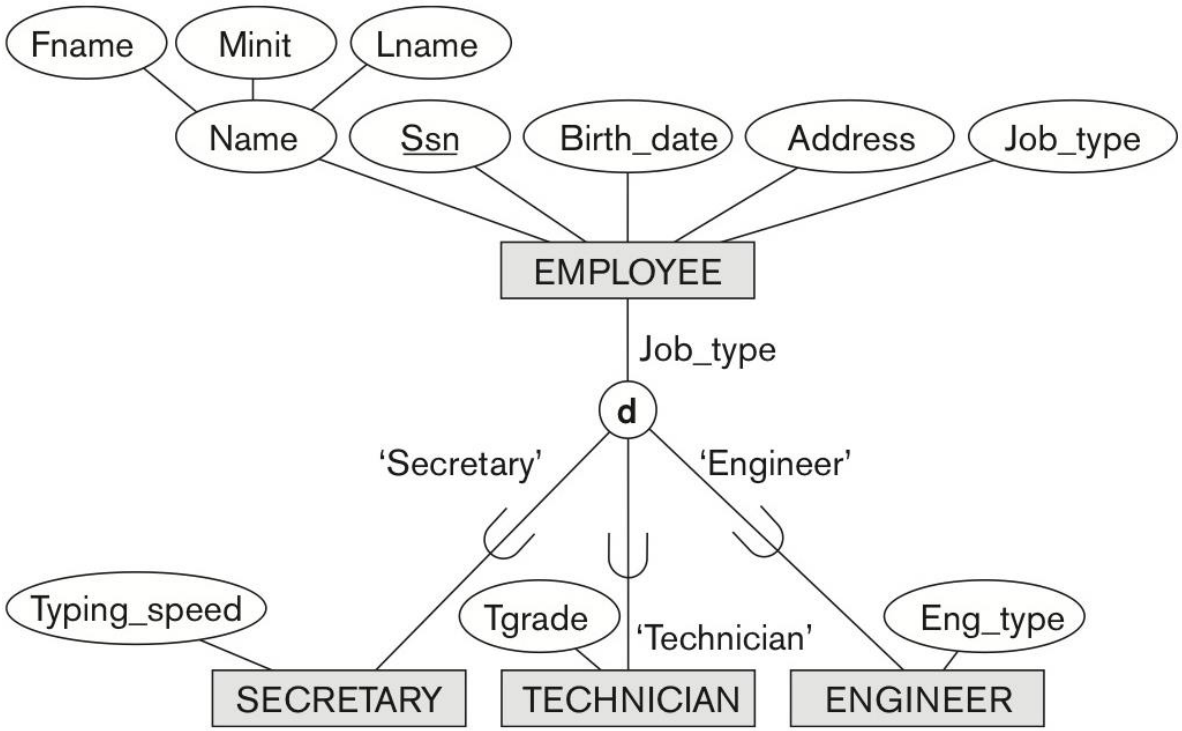
- Relationship types in relational schema
  - Relationship types may *not* be represented explicitly
  - Represented by (1) new attr, or (2) new relation
  - Data retrieval: EQUIJOIN or natural join

# Step 8: Specialization / Generalization

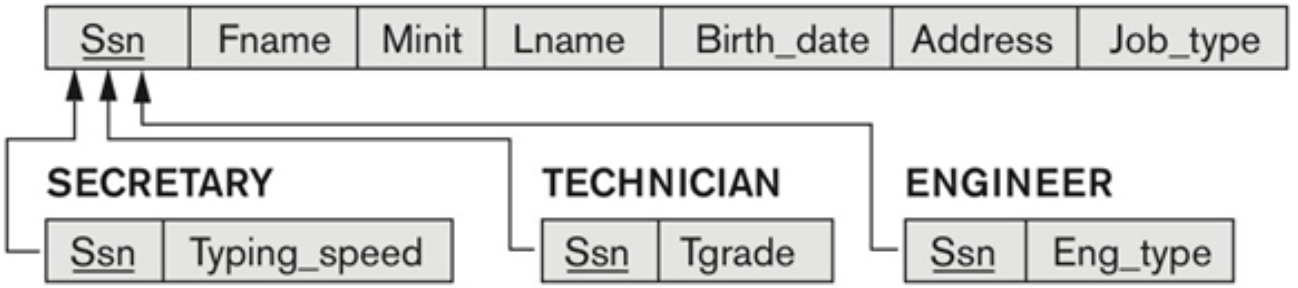
- Superclass C($\underline{k}$, $a_1$, …, $a_n$), and m subclasses {$S_1$, …, $S_m$}

- Option 8A: Multiple relations — superclass and subclasses
  - Create a new relation L for C
    - Attrs(L) = {$\underline{k}$, $a_1$, …, $a_n$}
    - PK(L)=k
  - Create a new relation $L_i$ for each $S_i$
    - Attrs($L_i$) = {k} ∪ Attrs($S_i$)
    - PK($L_i$) = k
  - Good for any specialization (total or partial, disjoint or overlapping)

# Option 8A




**(a) EMPLOYEE**

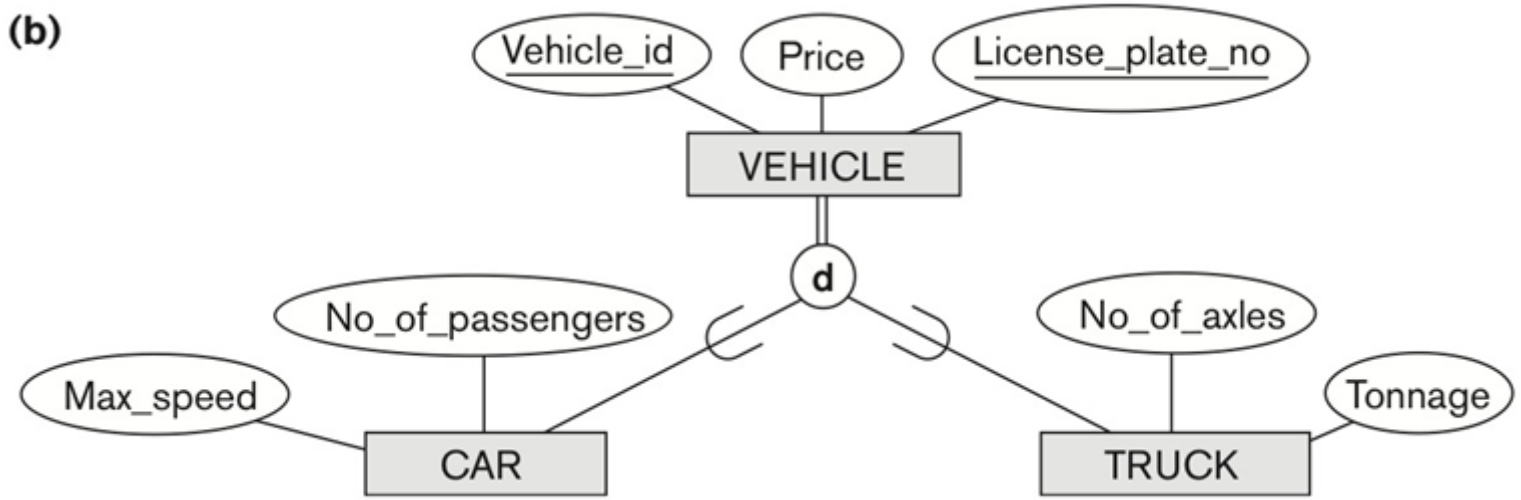


# Step 8: Specialization / Generalization (cont'd)

- Superclass C($\underline{k}$, $a_1$, …, $a_n$), and m subclasses {$S_1$, …, $S_m$}

- Option 8B: Multiple relations—subclass relations only
  - Create a new relation $L_i$ for each $S_i$
    - Attrs($L_i$) = Attrs($S_i$) ∪ {k, $a_1$, …, $a_n$}
    - PK($L_i$) = k
  - Good for
    - Subclasses are total (each entity in superclass must belong to at least one of the subclasses), **AND**
    - Specialization is disjoint
      - If overlapping, the same entity may be duplicated in several relations

# Option 8B




**(b) CAR**

| Vehicle_id | License_plate_no | Price | Max_speed | No_of_passengers |
|---|---|---|---|---|

**TRUCK**

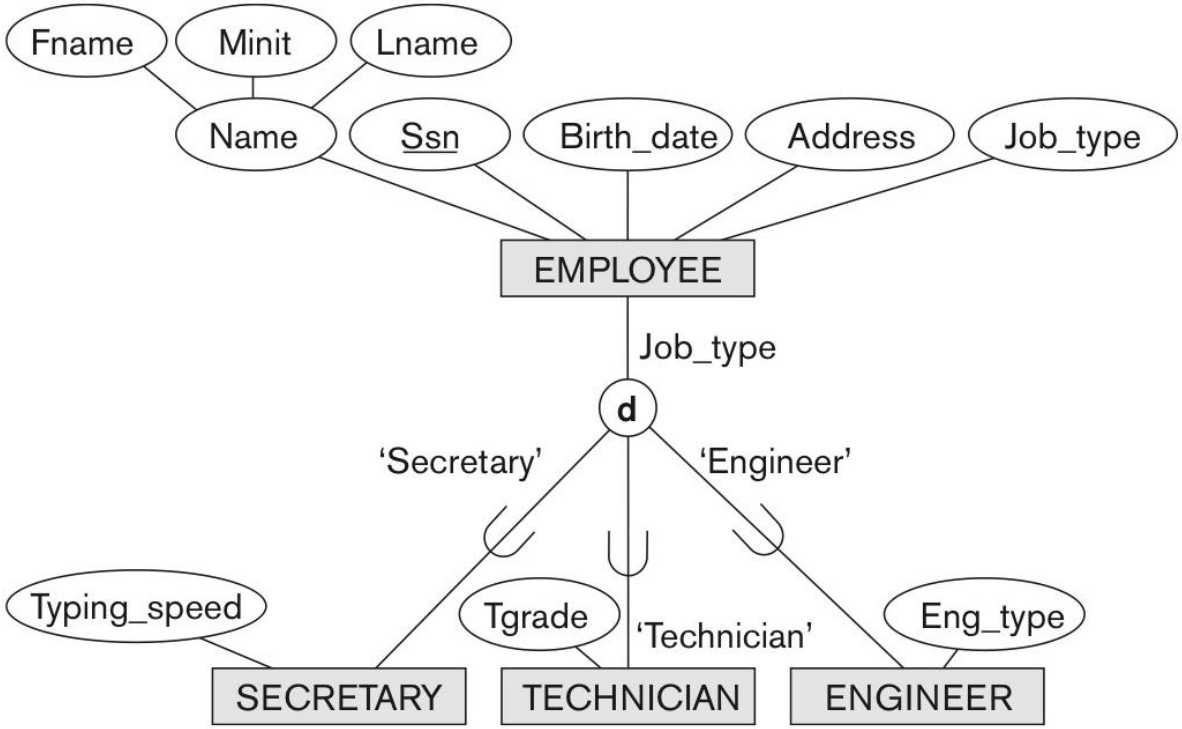
| Vehicle_id | License_plate_no | Price | No_of_axles | Tonnage |
|---|---|---|---|---|

# Step 8: Specialization / Generalization (cont'd)

- Superclass C($\underline{k}$, $a_1$, …, $a_n$), and m subclasses {$S_1$, …, $S_m$}

- Option 8C: Single relation with one type attr
  - Create a single relation L
    - Attrs(L) = {k, $a_1$, …, $a_n$} ∪ Attrs($S_1$) ∪ … ∪ Attrs($S_m$) ∪ {t}
    - T: type attr, indicating the subclass to which the tuple belong
  - Good for disjoint subclasses
    - Potential for generating many NULL values if many specific attrs exist in the subclasses

# Option 8C




**(c) EMPLOYEE**

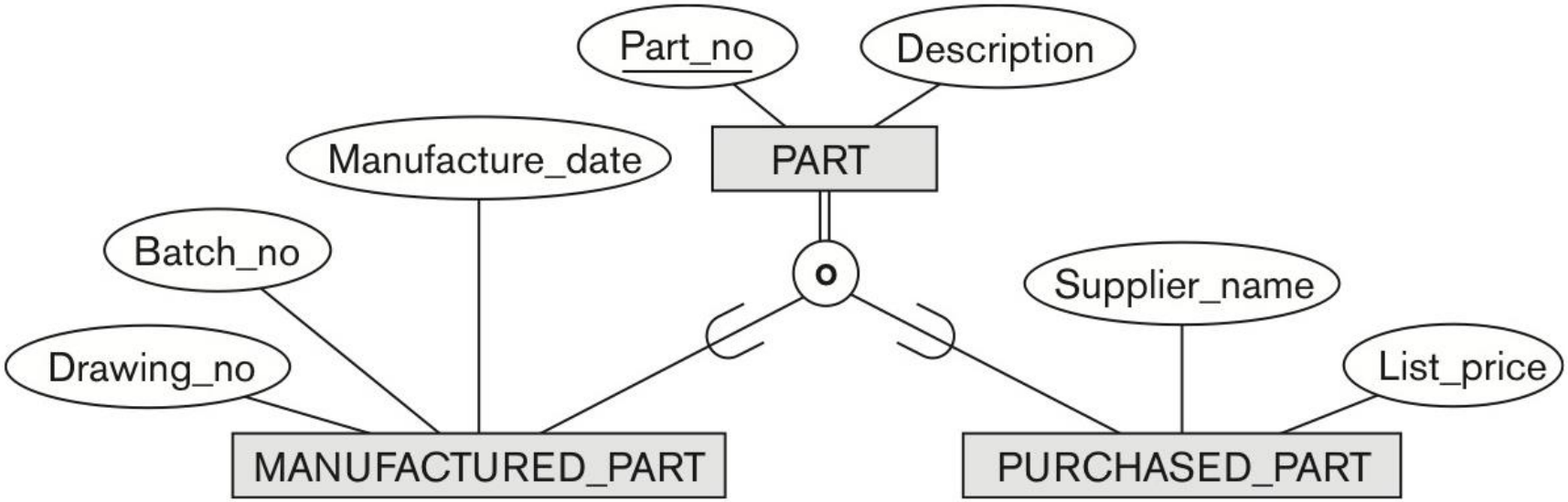
| Ssn | Fname | Minit | Lname | Birth_date | Address | Job_type | Typing_speed | Tgrade | Eng_type |
|---|---|---|---|---|---|---|---|---|---|

# Step 8: Mapping Specialization or Generalization (cont'd)

- Superclass C($\underline{k}$, $a_1$, …, $a_n$), and m subclasses {$S_1$, …, $S_m$}

- Option 8D: Single relation with multiple type attrs
  - Create a single relation L
    - Attrs(L) = {k, $a_1$, …, $a_n$} ∪ Attrs($S_1$) ∪ … ∪ Attrs($S_m$) ∪ {$t_1$, …, $t_m$}
    - PK(L) = k
    - $t_i$: boolean type attr, true if a tuple belong to subclass $S_i$
  - Good for
    - subclasses are overlapping  or disjoint

# Option 8D




**(d) PART**

| Part_no | Description | Mflag | Drawing_no | Manufacture_date | Batch_no | Pflag | Supplier_name | List_price |
|---|---|---|---|---|---|---|---|---|

# Table mapping exercises

- A partial key is a composite attr
- A multi-valued attr of a weak entity
- A multi-valued attr of a M:N relationship

# If drawing is not possible (e.g., online exam)

- Employee(ssn, fname, lname, super_ssn, dno)
  - PK: ssn
  - FK: (super_ssn) -> Employee(ssn)
  - FK: (dno) -> Department(dnumber)
- Department(dnumber, dname)
  - PK: dnumber
- Dependent(essn, fname, sex)
  - PK: (essn, fname)
  - FK: (essn) -> Employee(ssn)

# Summary

- Schema diagrams
- Mapping
  - Entity
  - Weak entity
  - 1:1 relationship
  - 1:N relationship
  - M:N relationship
  - Multi-value attr
  - N-ary relationship
  - Specialization, generalization

# Self Exercise

- Earlier HW: ERD to table mapping
- 7/E: Exercise 9.4, 9.5, 9.6, 9.8
- 6/E: Exercise 9.4, 9.5, 9.6, 9.8