H6

Designing Databases: Mapping a Conceptual into a Logical Design

Fundamentals of Databases

Alvaro A A Fernandes, SCS, UoM

[COMP23111 Handout 06 of 12]

Acknowledgements

- These slides are adaptations (mostly minor, but some major) of material authored and made available to instructors by **Ramez Elmasri and Shamkant B. Navathe** to accompany their textbook **Database Systems: Models, Languages, Design, and Application Programming, 6th (Global) Edition, Addison-Wesley Pearson, 2011, 978-0-13-214498-8**
- Copyright remains with them and the publishers, whom I thank.
- Some slides had input from Sandra Sampaio, whom I thank.
- All errors are my responsibility.

In Previous Handouts

- We learned how data models lead to a distinction between schemas and instances that enables a logical view of the data.
- We learned about the relational approach to logical data modelling.
- We learned about the relational algebra and SQL, both its DDL and DML capabilities and its querying constructs.
- We learned of the practical benefits of performing conceptual modelling before logical design and why the EER approach serves well this purpose.

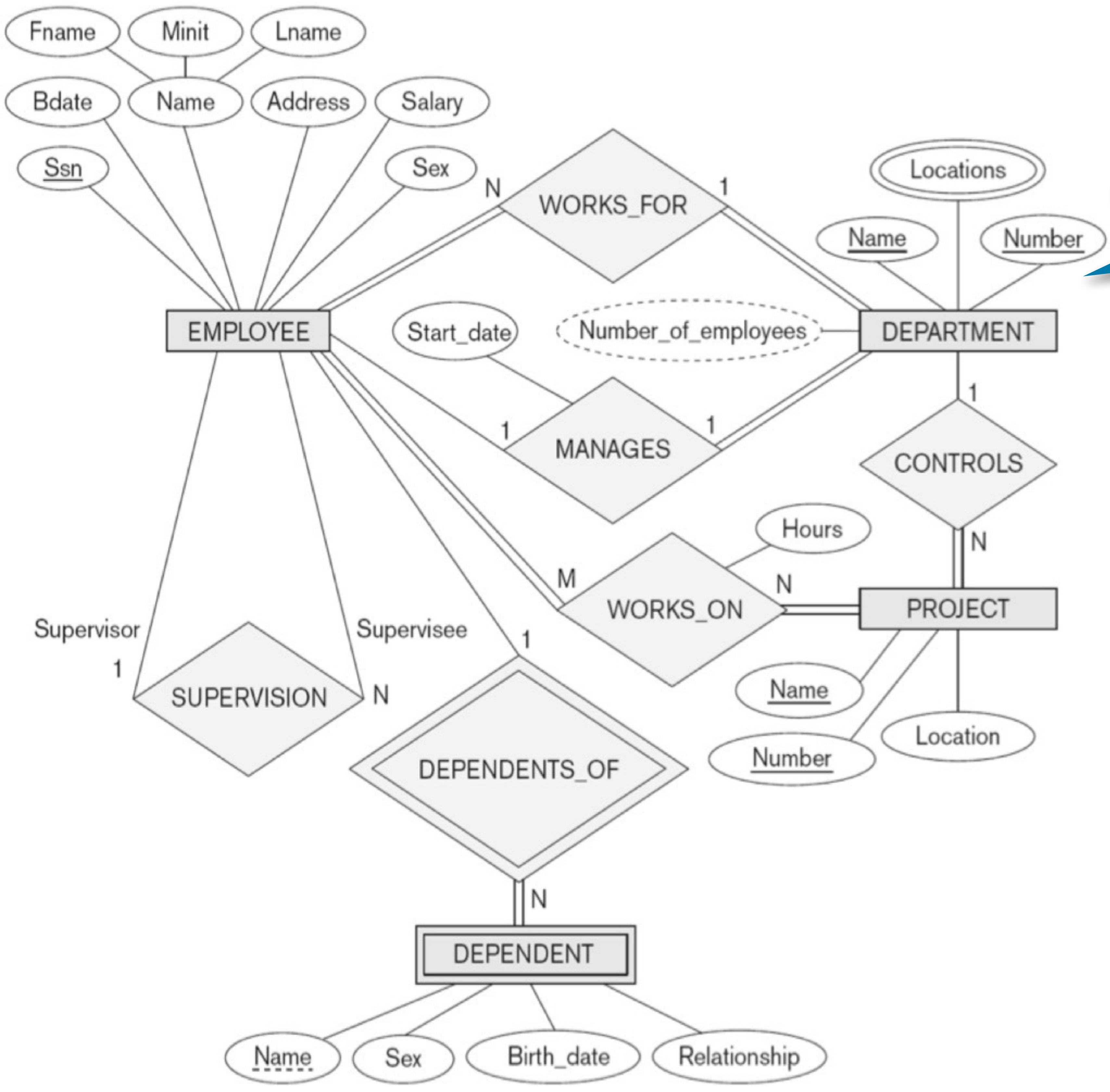
In This Handout

- We'll learn how a conceptual model can be mapped into a logical model that is capable of being implemented in a relational DBMS.

Relational Database Design through (E)ER-to-Relational Mapping

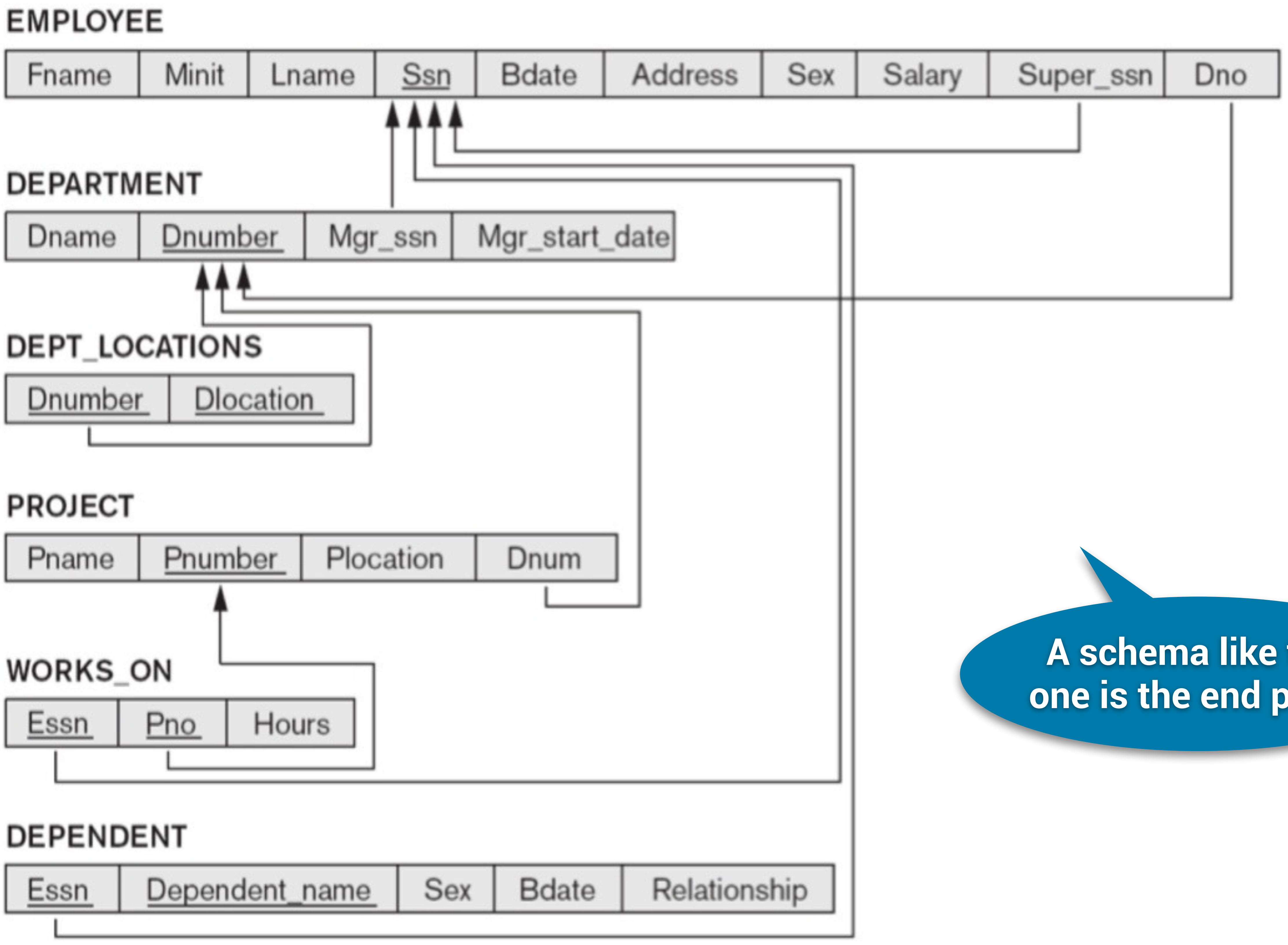
- The goal is to derive a relational schema (i.e., logical, implementable) from an (E)ER conceptual schema
- As we have seen, a conceptual schema (because it is more abstract and expressive) is better suited to serve as a bridge between the DB designer and the various kinds of DB users when the latter are expressing interactively and iteratively their data requirements to the former.
- There is a seven-step semi-formal procedure to convert an ER model into a relational schema.
- Then, two further steps suffice for converting an EER model into a relational schema.

An Example ER Diagram for a COMPANY DB



A model like this one is the starting point.

An Example Relational Schema for a COMPANY DB


EMPLOYEE
Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno
DEPARTMENT
Dname | Dnumber | Mgr_ssn | Mgr_start_date
DEPT_LOCATIONS
Dnumber | Dlocation
PROJECT
Pname | Pnumber | Plocation | Dnum
WORKS_ON
Essn | Pno | Hours
DEPENDENT
Essn | Dependent_name | Sex | Bdate | Relationship

A schema like this one is the end point.

Step 1: Mapping of Regular Entity Types

- For each regular (i.e., not weak) entity type E, create a relation R that includes all the simple attributes of E.
- Then include the leaf-level component attributes of a composite attribute.
- Then choose one of the keys in E as the primary key (PK) of R.
- If the chosen key is composite in E, then make the set of component attributes in it the composite PK of R.
- Then make the other keys in E secondary keys in R (i.e., assert them as unique in the DDL).
- Such relations are called **entity relations** and *each tuple represents an entity instance*.

After Step 1: Entity Relations

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary

DEPARTMENT

Dname	Dnumber

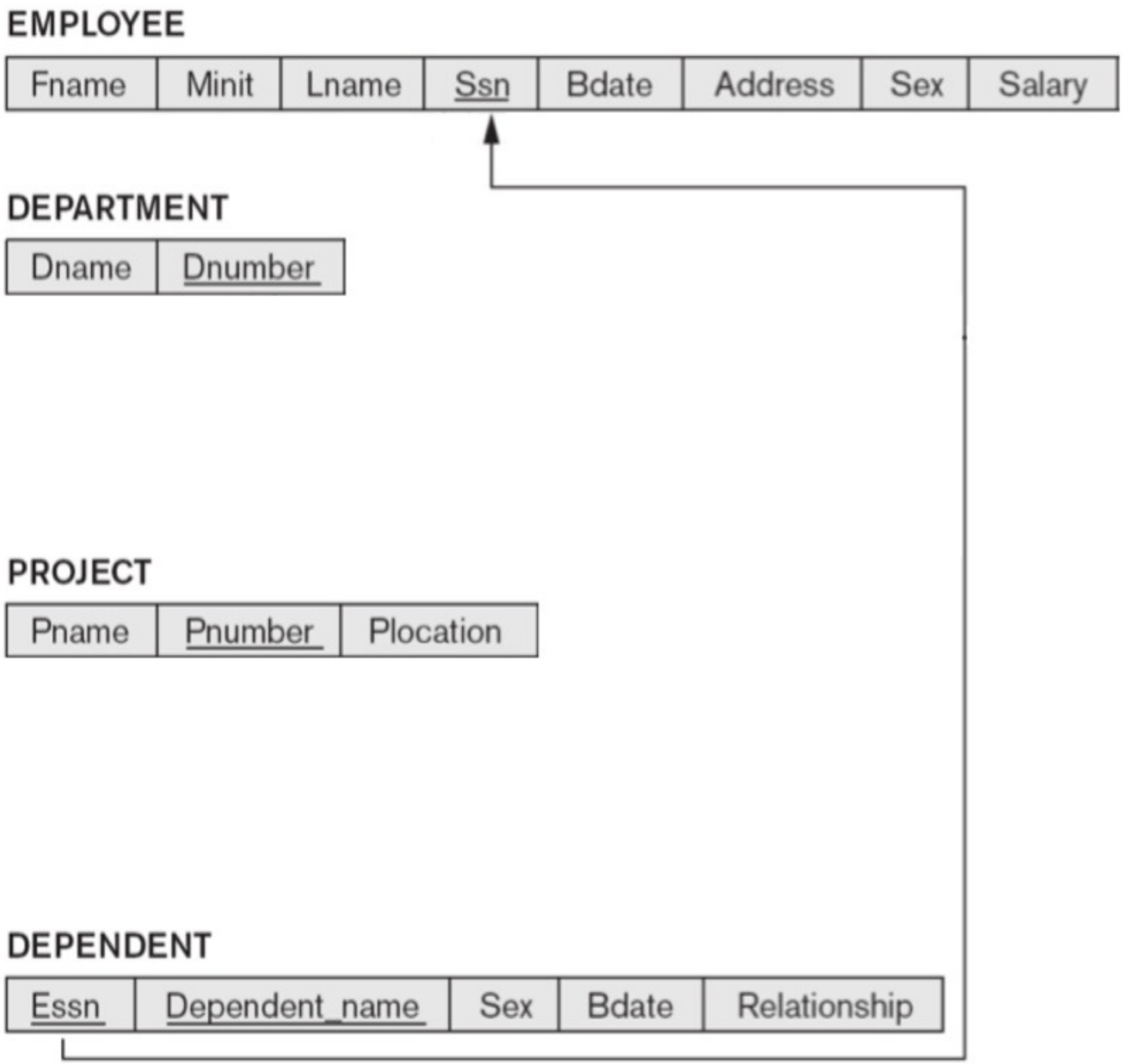
PROJECT

Pname	Pnumber	Plocation

Step 2: Mapping of Weak Entity Types

- For each weak entity type W with owner entity type E, create a relation R that includes all the simple attributes (or components of a composite attribute) of W as attributes of R.
- Then include the PK attributes of E as foreign key (FK) attributes of R.
- Then make the PK of R the composition of the PK of E and the weak/partial key of W.
- If the owner of W is another weak entity type W' with owner E, then map W' first and map W subsequently.

After Steps 1-2



Step 3: Mapping of Binary 1:1 Relationship Types

- For each binary 1:1 relationship type **R**, identify the relations S and T that correspond to the entity types participating in it.
- There are three possible approaches:
  - Foreign key approach (FKA)
  - Merged relationship approach (MRA)
  - Cross-reference or relationship relation approach (RRA)

Step 3: Mapping of Binary 1:1 Relationship Types

- The foreign key approach is very useful, in general.
- In **FKA**:
  - Choose one of the relations, say, S, and include the PK of T (the parent, or exporter, as it were) as FK in S (the child, or importer, as it were).
  - Then include all the simple (or component of composite) attributes of the relationship type as attributes in S.
  - It is better to chose an entity type with total participation in **R** for the role of S (i.e., as the PK importer).

Step 3: Mapping of Binary 1:1 Relationship Types

- In **MRA**:
  - Merge the two entity types and the relationship into a single relation.
  - This is possible when both participations are total, since the two relations will have the same number of tuples at all times.

Step 3: Mapping of Binary 1:1 Relationship Types

- In **RRA**:
  - Create a new, distinct relation R′ for capturing the cross-referencing between S and T.
  - Then include as FKs of R′ the PK of S and the PK of T.
  - Then choose one of them to be the PK of R′ and assert the other to be unique (i.e., a secondary key).
  - This relation is then called a **relationship relation** (or a lookup table).

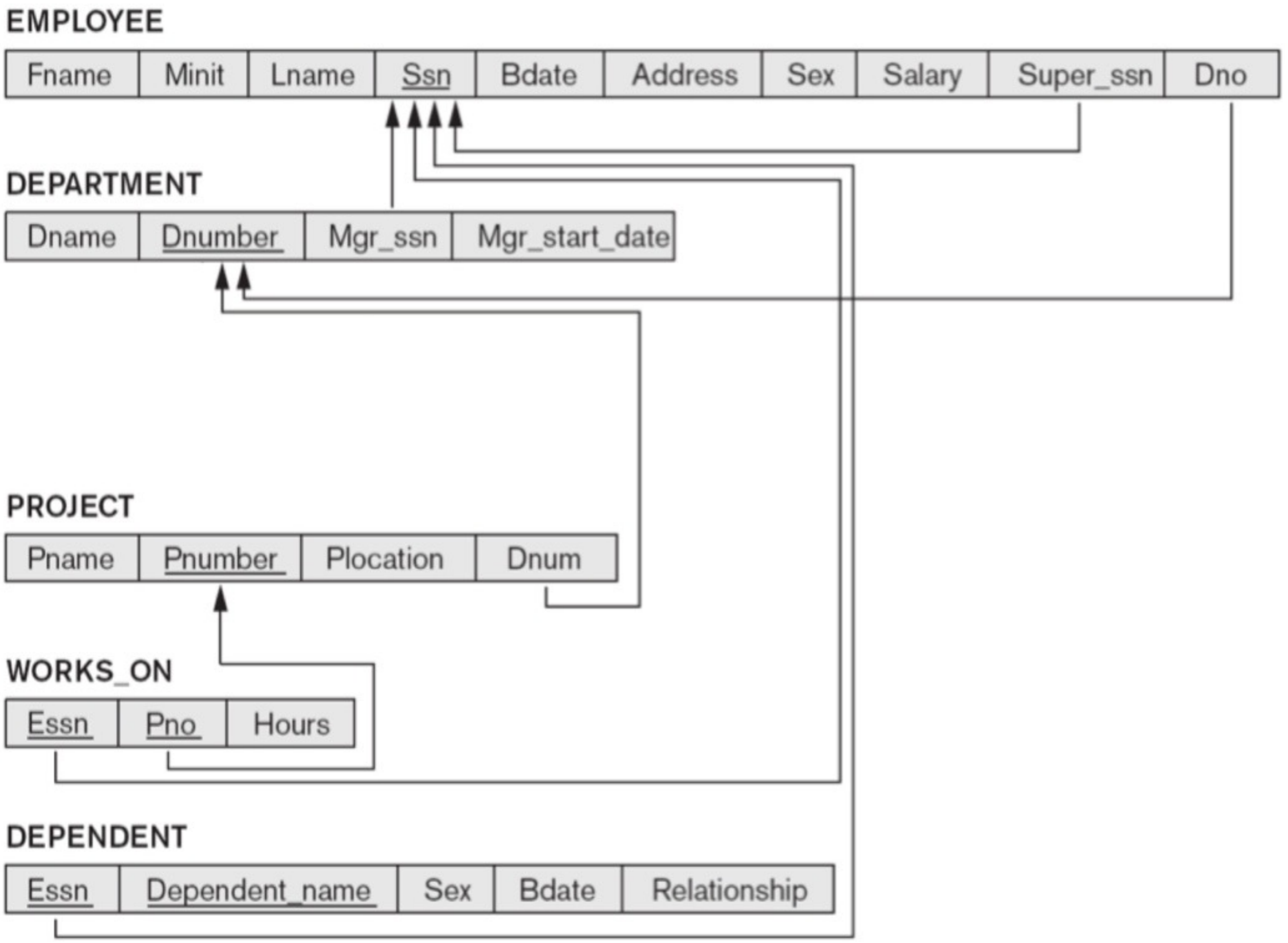
Step 4: Mapping of Binary 1:N Relationship Types

- For each regular binary 1:N relationship type **R** between entity types S and T, use FKA:
  - Identify the relation, say S, that stands at the N-side of R (i.e., is the child, or importer)
  - Then include the PK of T (i.e., the parent, or exporter) as FK in S
  - Then include the simple (or component of composite) attributes of **R** as attributes of S.
- Alternatively, use RRA.

Step 5: Mapping of Binary M:N Relationship Types

- For each binary M:N relationship type **R** between entity types S and T, use RRA:
  - Create a new relation R′.
  - Then include as FKs of R′, the PKs of S and T.
  - Then make the PK of R′ the composition of the PKs of S and T.
  - Then include any simple (or components of composite) attributes of **R** in R′.

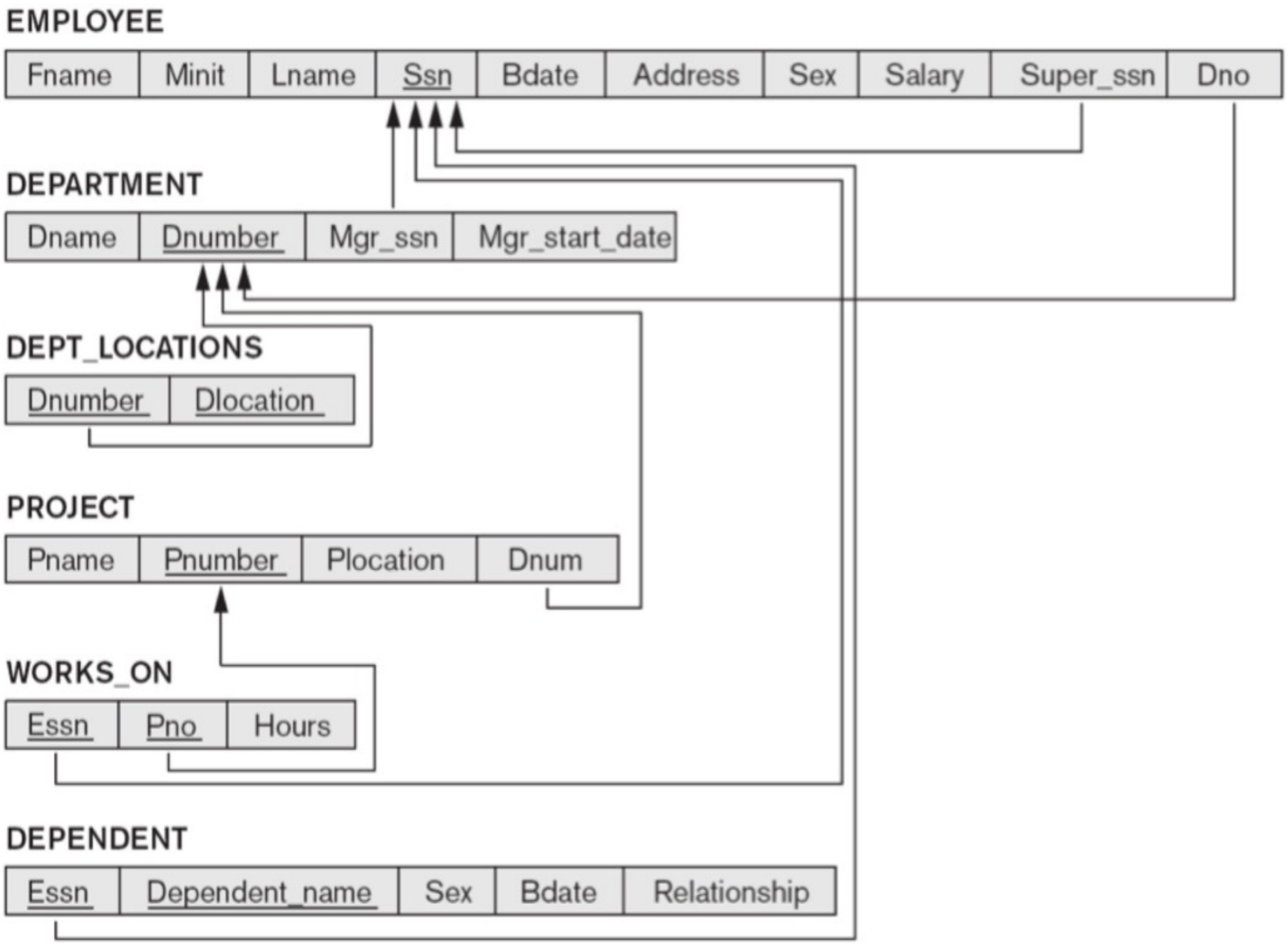
After Steps 1-5



Step 6: Mapping of Multivalued Attributes

- For each multivalued attribute A in an entity type S
  - Create a new relation A′ with the attribute A.
  - Then include the PK of S as an FK of A′.
  - Then make the PK of A′ the composition of A and the PK of S.
  - Then, if the multivalued attribute is composite, include its components.

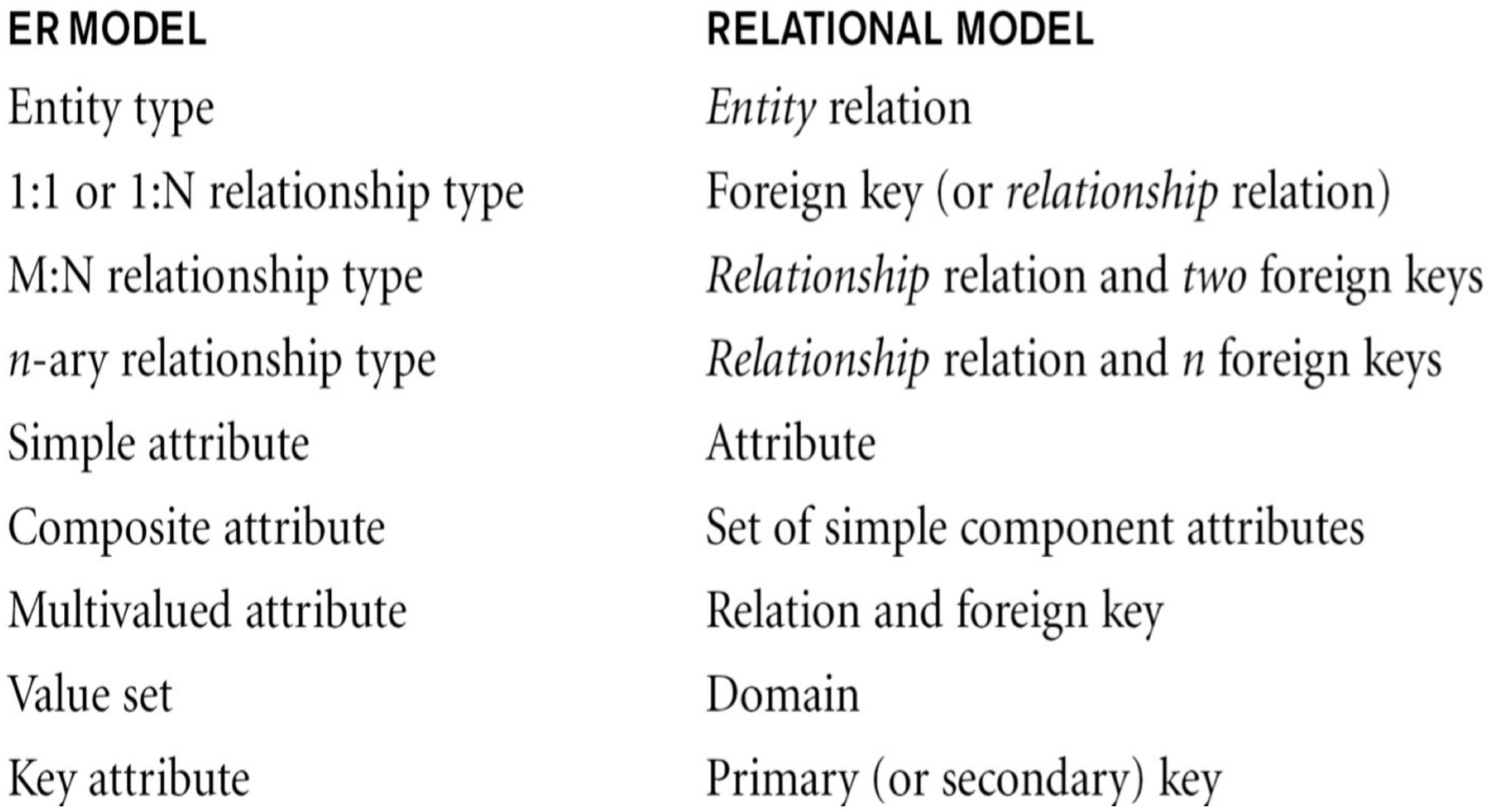
After Steps 1-6


EMPLOYEE
Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno
DEPARTMENT
Dname | Dnumber | Mgr_ssn | Mgr_start_date
DEPT_LOCATIONS
Dnumber | Dlocation
PROJECT
Pname | Pnumber | Plocation | Dnum
WORKS_ON
Essn | Pno | Hours
DEPENDENT
Essn | Dependent_name | Sex | Bdate | Relationship

Step 7: Mapping of N-ary Relationship Types

- For each n-ary, n>2, relationship type **R**:
  - Create a new relation R′ to represent **R**.
  - Then include the PKs of participating entity types as FKs in R′.
  - Then make the PK of R′ the concatenation of the PKs of participating entity types.
  - <u>But</u>, if any participating entity type X does so with a cardinality constraint of 1, then do not include the PK of X in the PK of R′.
  - Then include any simple (or components of composite) attributes of **R** in R′.

Correspondence between ER and Relational



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

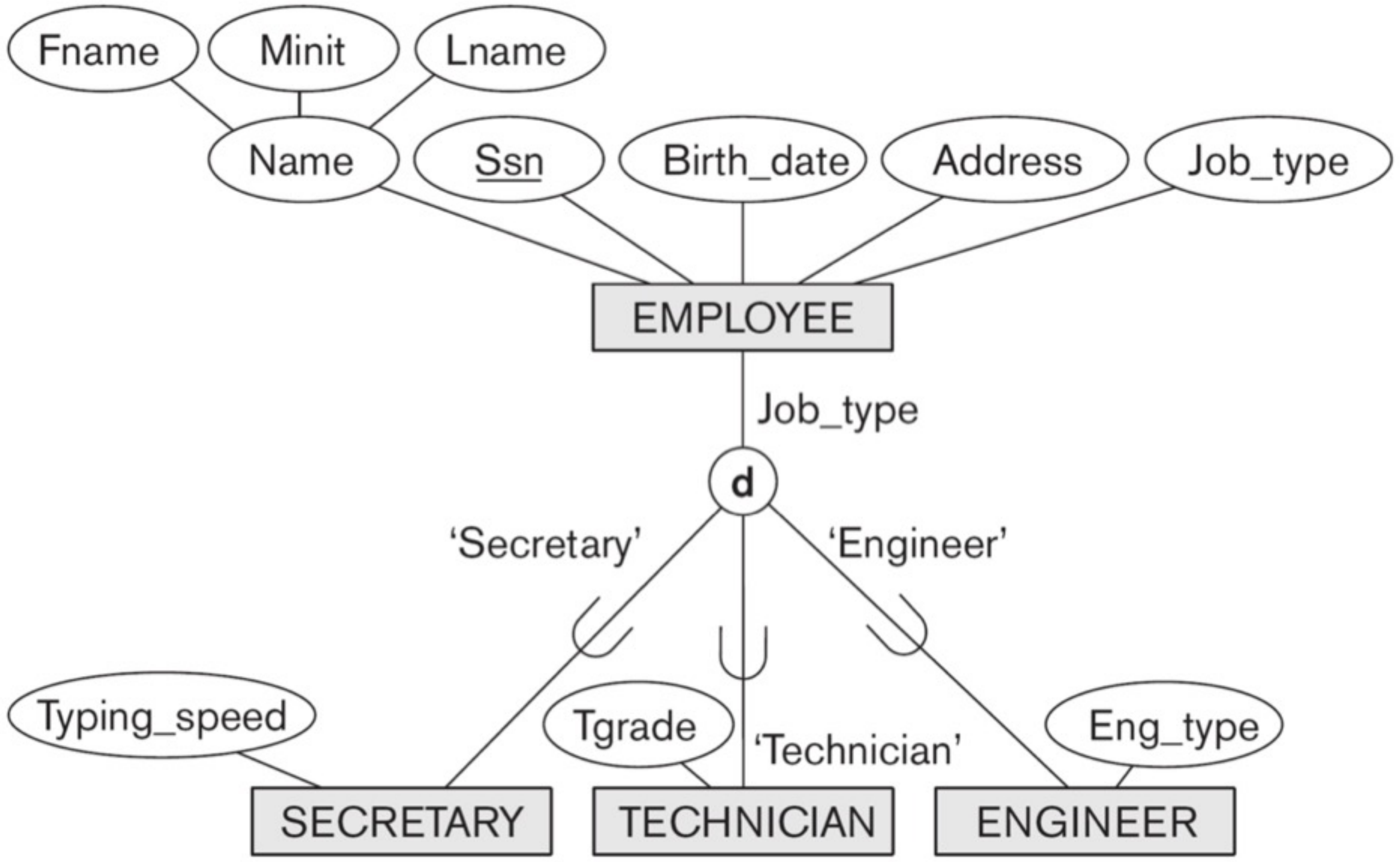
Major Contrasts

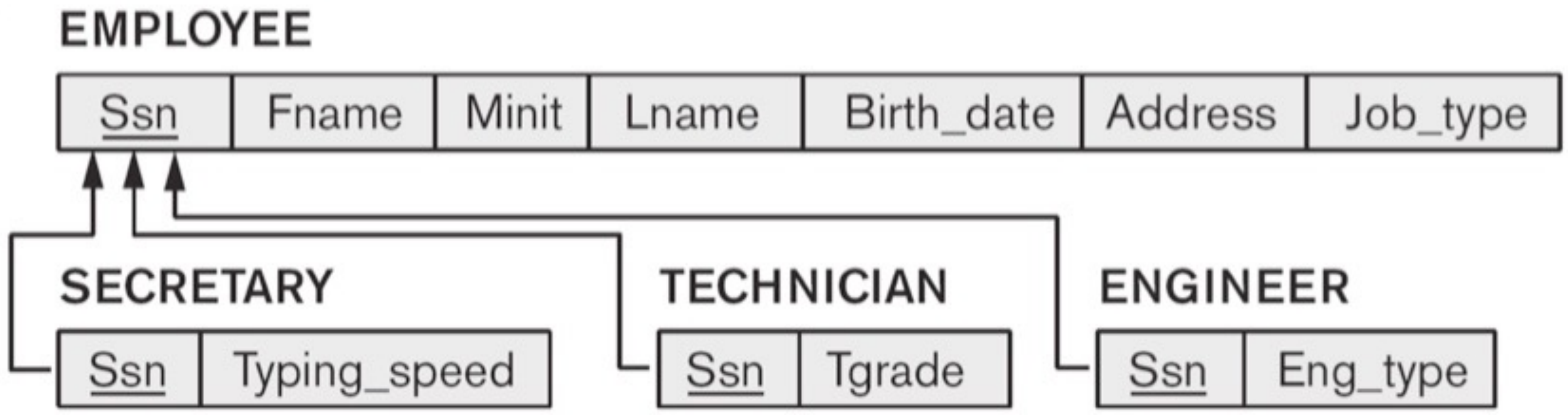
- In a relational schema, relationship types are not represented explicitly.
- Instead, they are represented by having two attributes A and B, where one is a primary key and the other a foreign key.
- Composite and multivalued attributes are decomposed in the relational schema .
- Derived attributes need triggers and procedures (studied later in the course) to be supported.
- Specialization and generalization are only indirectly supported (as we shall now explore).

Step 8: Options for Mapping Specialization or Generalization

- Option 8A: Using *multiple relations* (superclass and subclasses) works for any specialization (total or partial, disjoint or overlapping).

Step 8 Example: Option 8A





EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type

SECRETARY

Ssn	Typing_speed

TECHNICIAN

Ssn	Tgrade

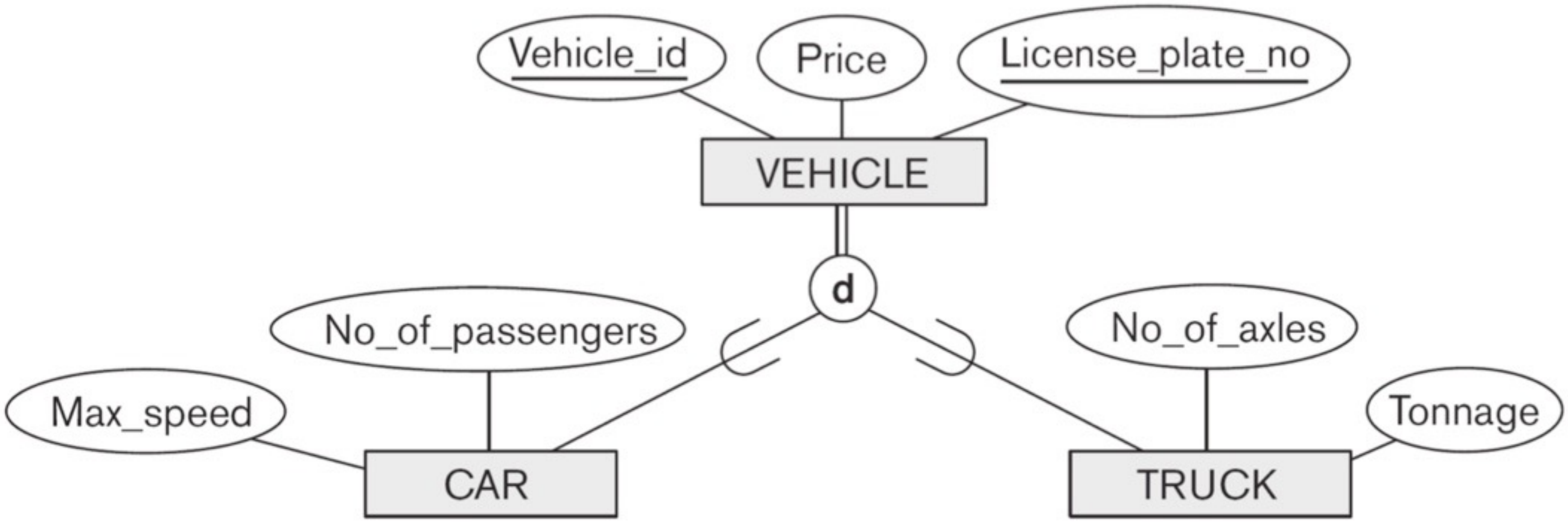
ENGINEER

Ssn	Eng_type

Step 8: Options for Mapping Specialization or Generalization

- Option 8B: Using *multiple relations on subclass relations* only works if subclasses are total.
- It is only recommended if the specialization has disjointedness constraint.
- Otherwise, the same entity may be duplicated in several relations.

Step 8 Example: Option 8B



CAR

Vehicle_id	License_plate_no	Price	Max_speed	No_of_passengers

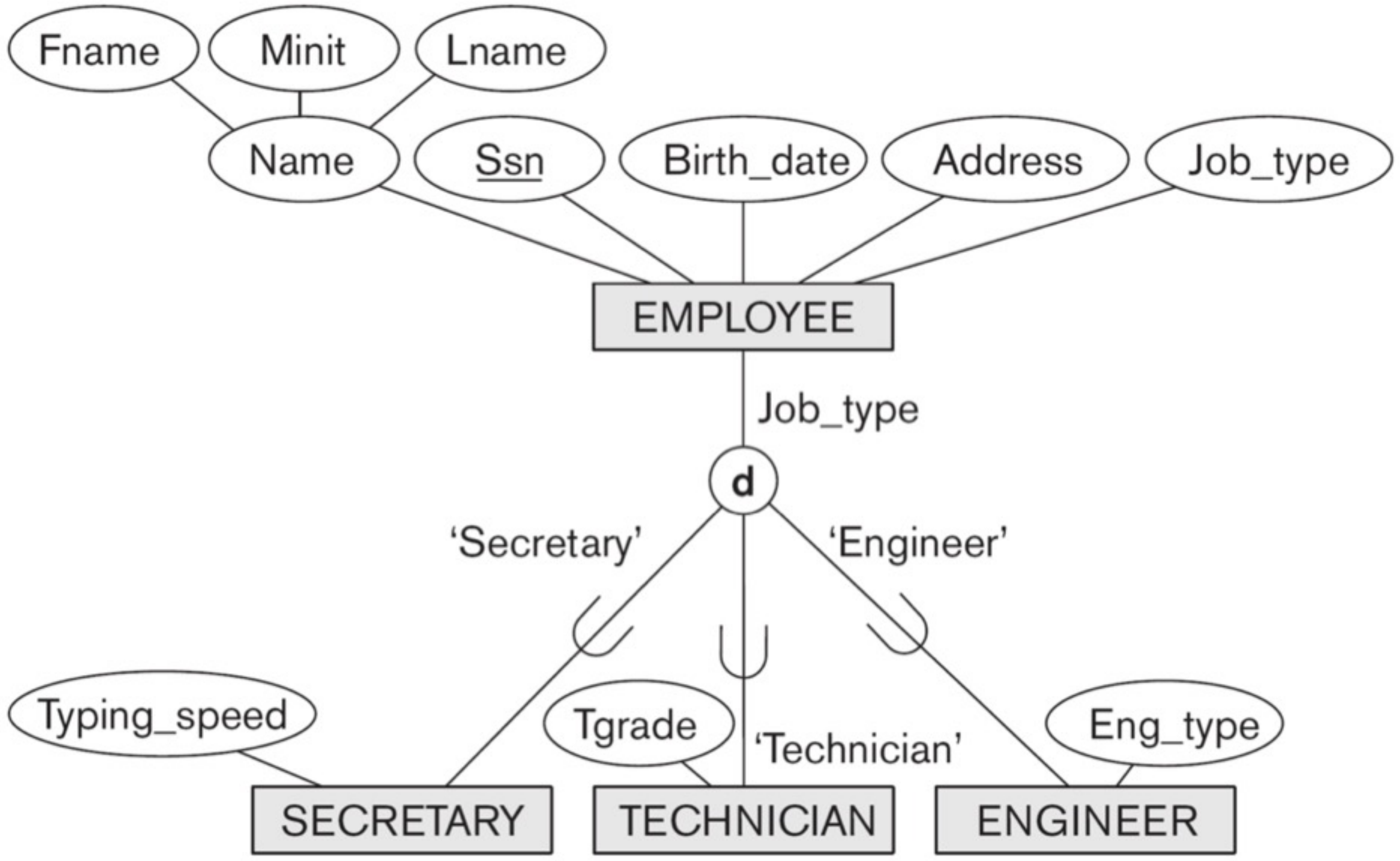
TRUCK

Vehicle_id	License_plate_no	Price	No_of_axles	Tonnage

Step 8: Options for Mapping Specialization or Generalization

- Option 8C: Using a *single relation with one so-called type (or discriminating) attribute* whose value in a tuple indicates which subclass the tuple belongs to.
- This works only if subclasses are disjoint and still has the potential for generating many NULL values if many specific attributes exist in the subclasses.

Step 8 Example: Option 8C



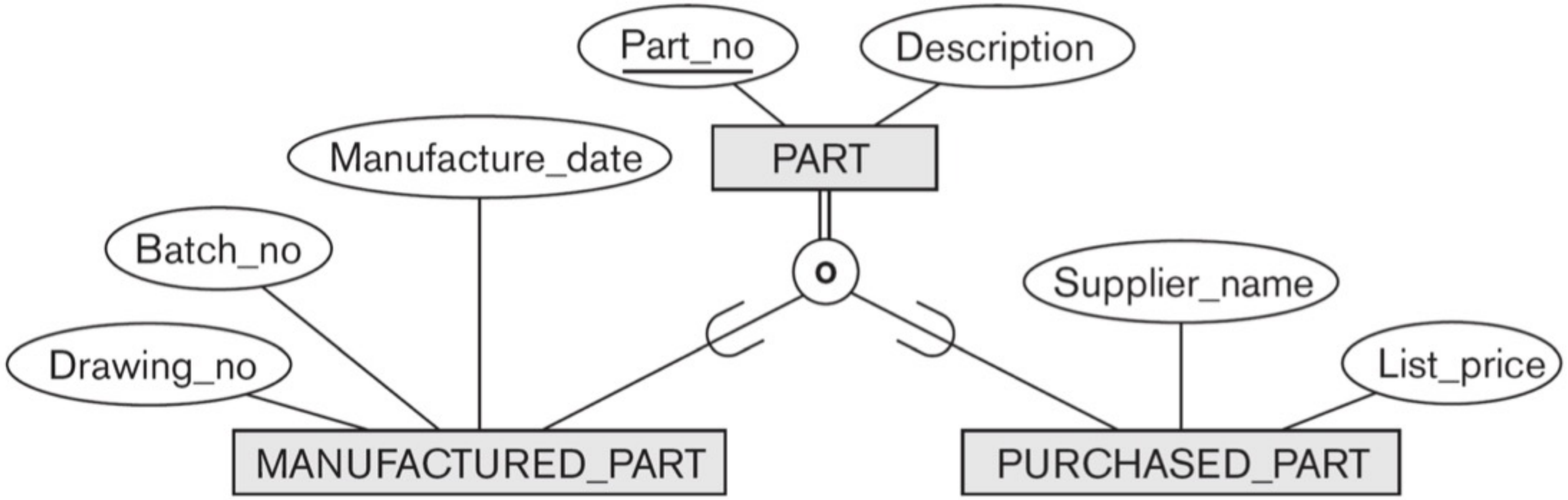
EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type

Step 8: Options for Mapping Specialization or Generalization

- Option 8D: Using a *single relation with multiple Boolean type attributes*, each of which indicates whether or not the tuple belongs to the corresponding subclass.
- This work when subclasses are overlapping and will also work for a disjoint specialization.

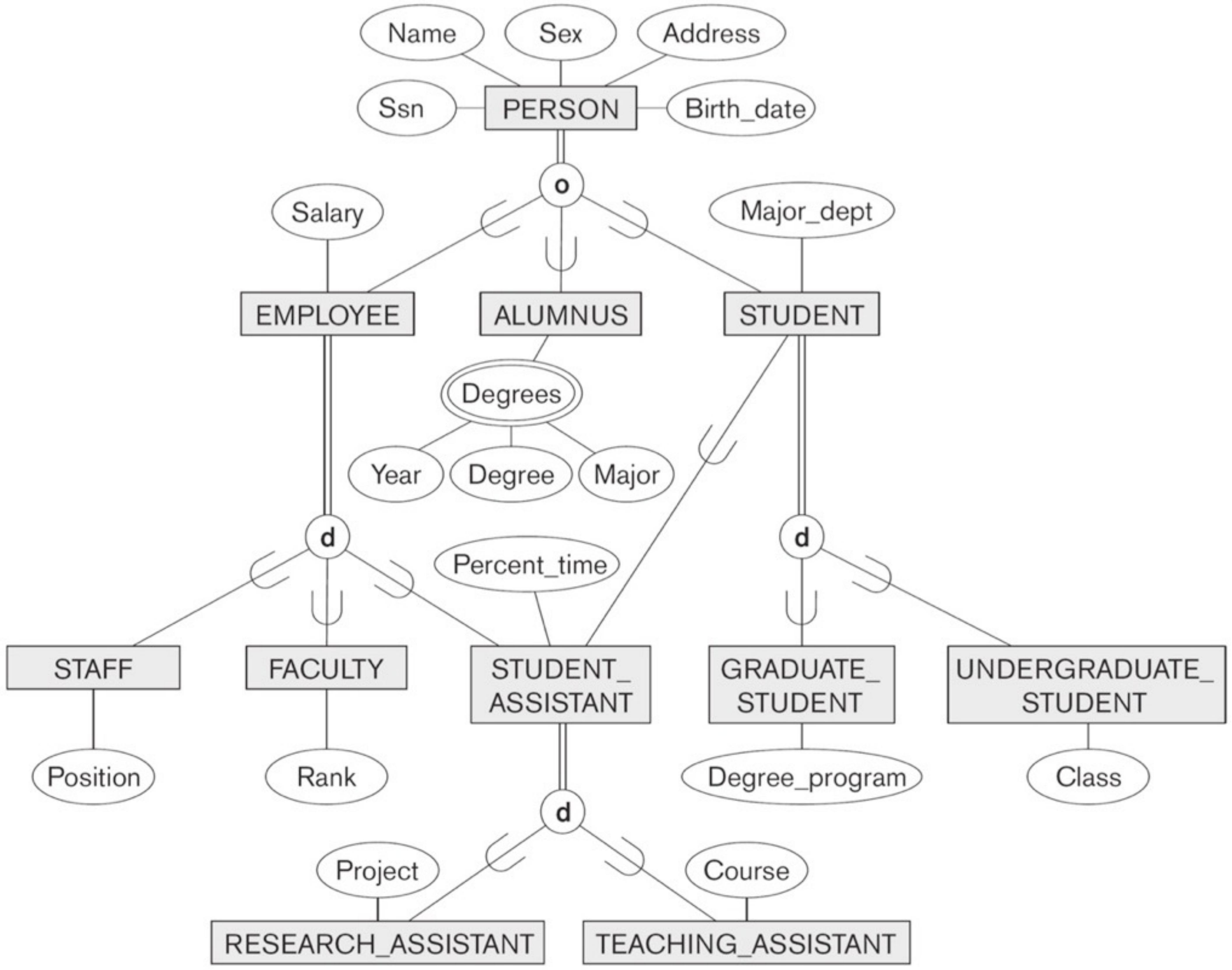
Step 8 Example: Option 8D



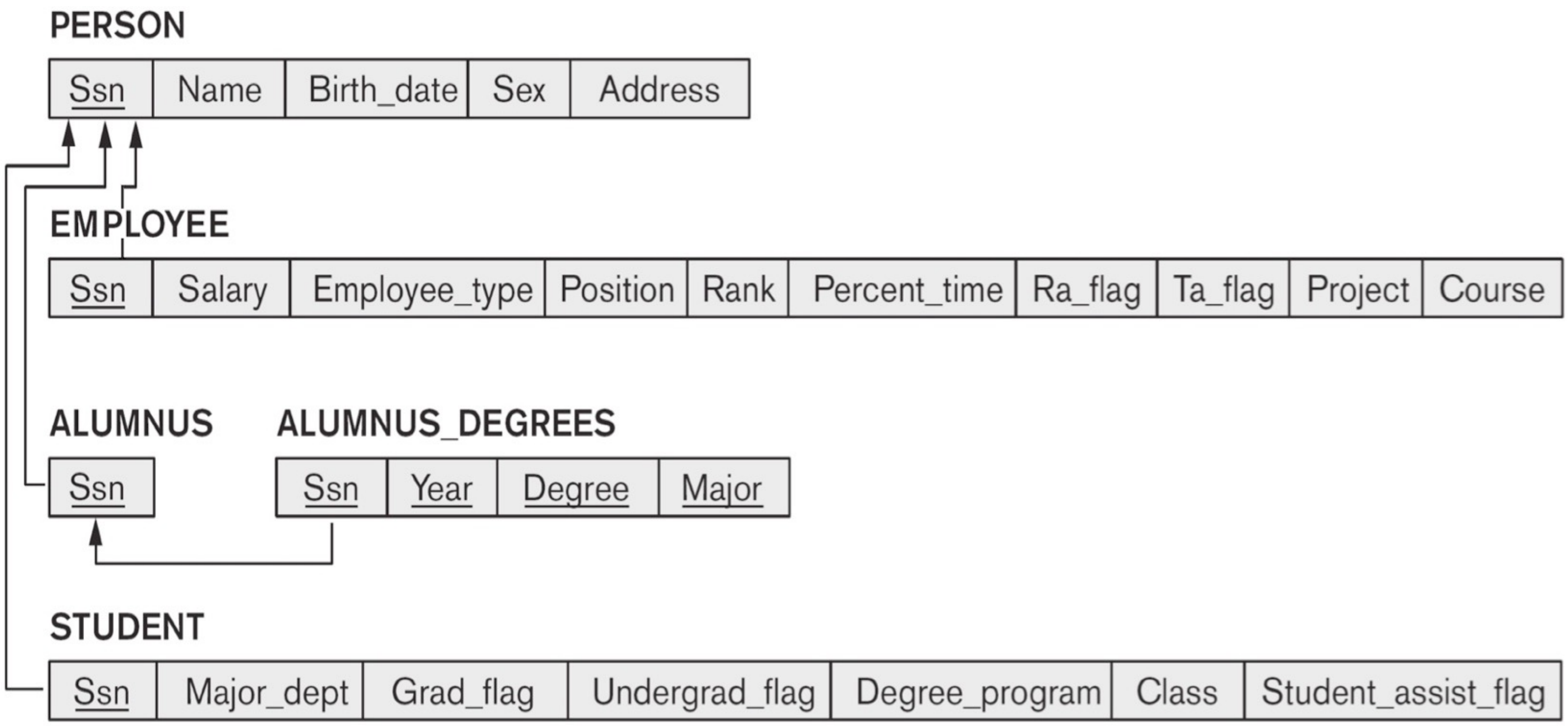
PART

Part_no	Description	Mflag	Drawing_no	Manufacture_date	Batch_no	Pflag	Supplier_name	List_price

A Specialization Lattice with Multiple Inheritance



Mapping the Lattice Using Various Options


PERSON
Ssn | Name | Birth_date | Sex | Address
EMPLOYEE
Ssn | Salary | Employee_type | Position | Rank | Percent_time | Ra_flag | Ta_flag | Project | Course
ALUMNUS
Ssn
ALUMNUS_DEGREES
Ssn | Year | Degree | Major
STUDENT
Ssn | Major_dept | Grad_flag | Undergrad_flag | Degree_program | Class | Student_assist_flag

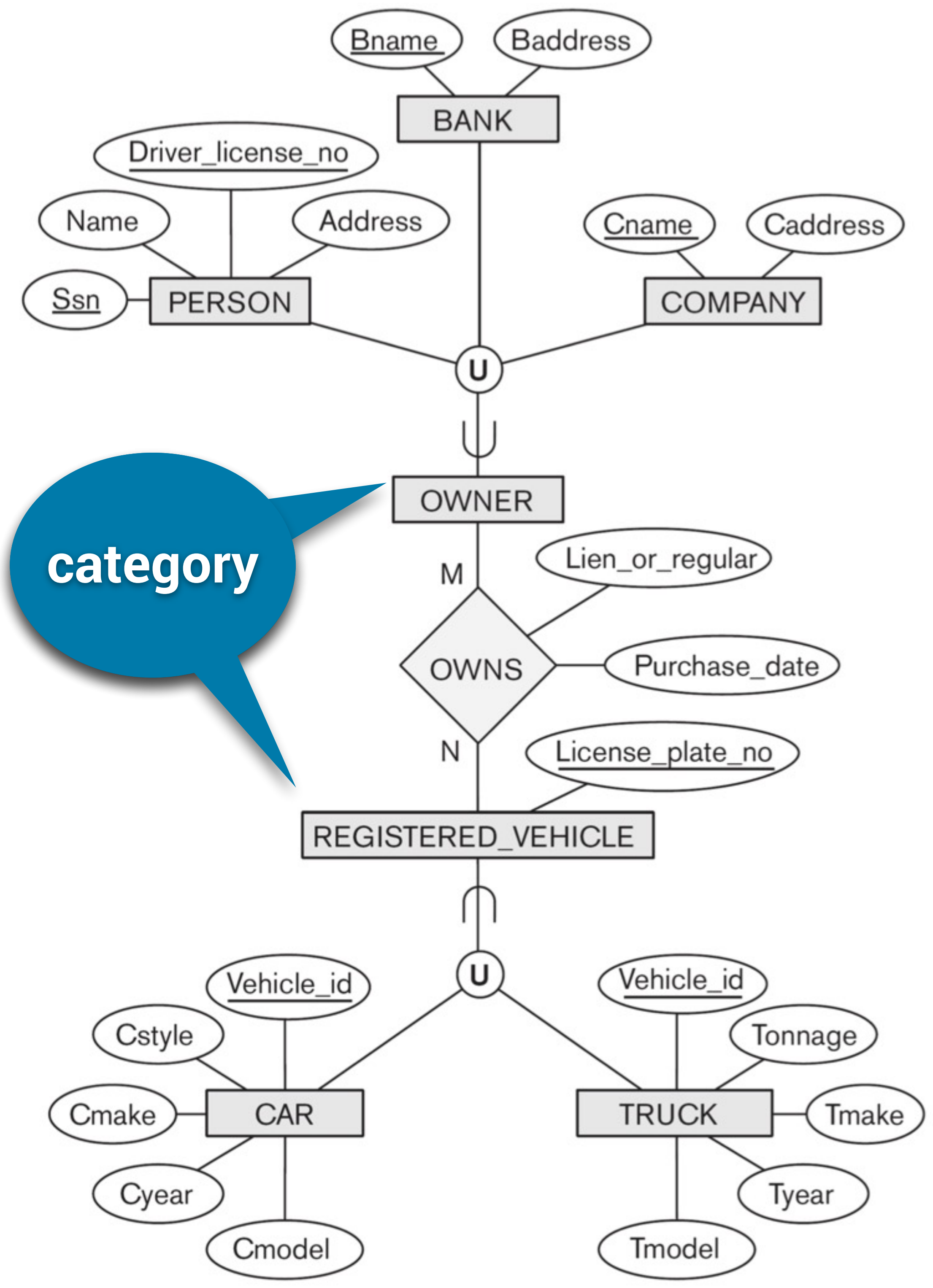
Step 9: Mapping of Union Types (Categories)

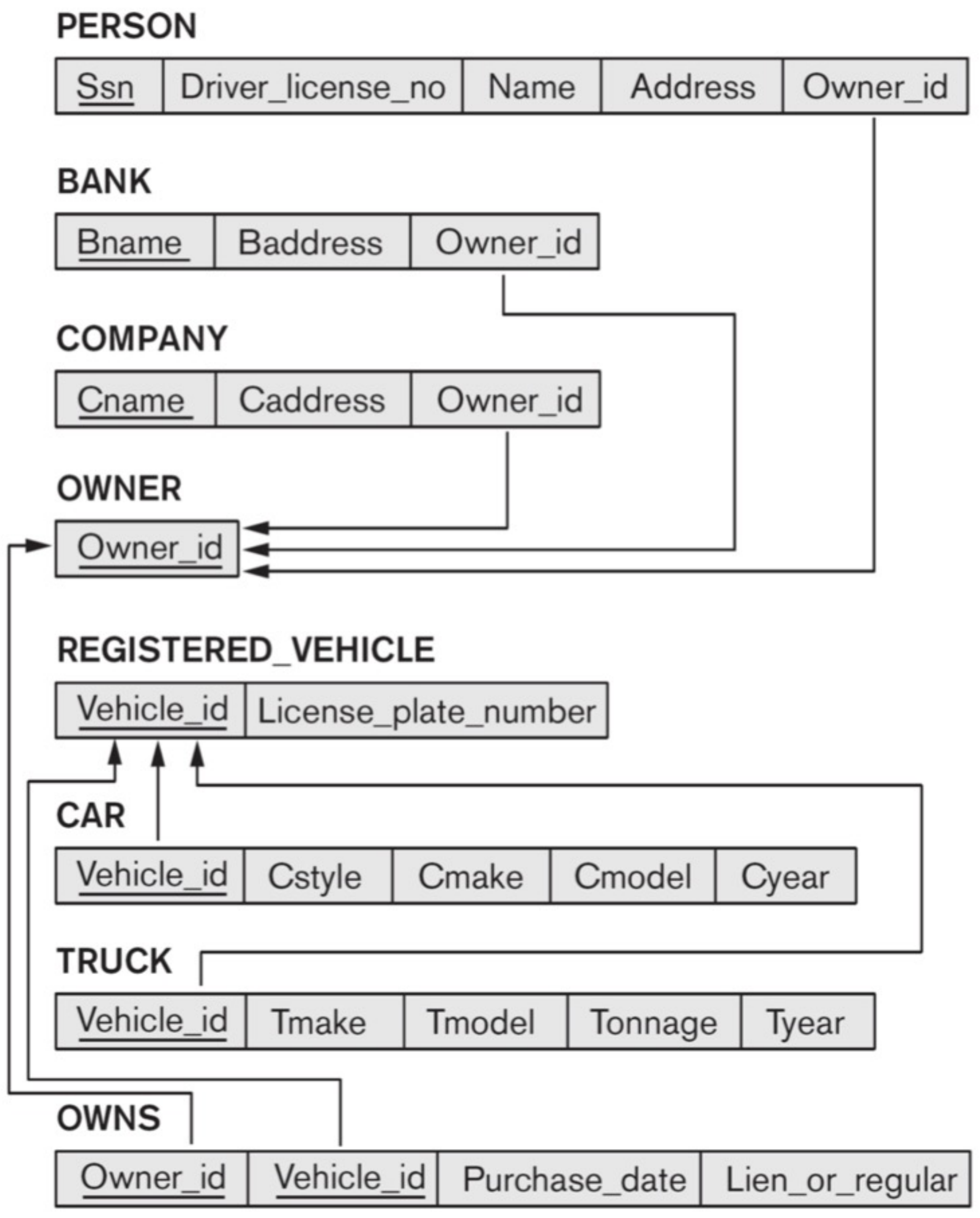
- A category (or union type) is a subclass of the union of two or more superclasses that have different keys because they can be of different entity types.
- If the defining superclasses have different keys, the custom is to specify a new key attribute, called a **surrogate key**.

Step 9: Mapping of Union Types (Categories)

- For a category C, create a relation C′ that includes all the attributes of C.
- Then make the PK of C′ the surrogate key of C.
- Then include the PK of C′ as FK in the relations corresponding to each of the superclasses of C.
- Then, optionally, add a type (or discriminating) attribute to C′ to indicate the entity type (i.e., what is the superclass) of each tuple.
- Note that of a particular entity in a superclass is not a member of C, it would have a NULL surrogate key, i.e., it does not appear as a tuple in C′.

Step 9 Example: Mapping Categories





PERSON

Ssn	Driver_license_no	Name	Address	Owner_id

BANK

Bname	Baddress	Owner_id

COMPANY

Cname	Caddress	Owner_id

OWNER

Owner_id

REGISTERED_VEHICLE

Vehicle_id	License_plate_number

CAR

Vehicle_id	Cstyle	Cmake	Cmodel	Cyear

TRUCK

Vehicle_id	Tmake	Tmodel	Tonnage	Tyear

OWNS

Owner_id	Vehicle_id	Purchase_date	Lien_or_regular

In The Next Handout

- We'll learn how to analyze, evaluate and improve a logical model.
- For that we will study the theory of normalization, based on functional dependencies.