Relational Schema Design

Elmasri and Navathe 9.1 – 9.2

Outline

- ER to Relational Mapping
  - Steps 1 - 7

Step 1 – Regular Entity Types

- For each regular entity type E
  - create relation R that includes all simple attributes of E and simple component attributes of composite attributes of E
  - choose one of the key attributes of E as the primary key

Step 2 – Weak Entity Types

- For each weak entity type W with owner entity type E
  - create relation R including all simple attributes and simple component attributes of W
  - include primary key of E's corresponding relation as a foreign key in W
  - primary key of R is combination of foreign key from E and W's partial key

Step 3 –Binary 1:1 Relationship Types

- For each binary 1:1 relationship type R
  - identify relations corresponding to participating entities, say T and S
  - if there is total participation, make that S
  - include primary key of T as a foreign key in S
  - include all simple attributes and simple components of composites of R as attributes of S

Step 4
Regular 1:N Binary Relationship Types

- For each regular 1:N binary relationship type R
  - identify relations, say S and T, corresponding to participating entity types
  - if S is at “N side” of relationship type then include primary key of T in S
  - include any simple attributes or simple components of R as attributes of S

Step 5 – Binary M:N Relationship Types

- For each binary N:M relationship type K
  - create new relation R to represent K
  - if S and T are the relations corresponding to the participating entity types then include their primary keys as foreign keys in R
  - combination of foreign keys will be primary key of R
  - include any simple attributes or simple components of K as attributes of R

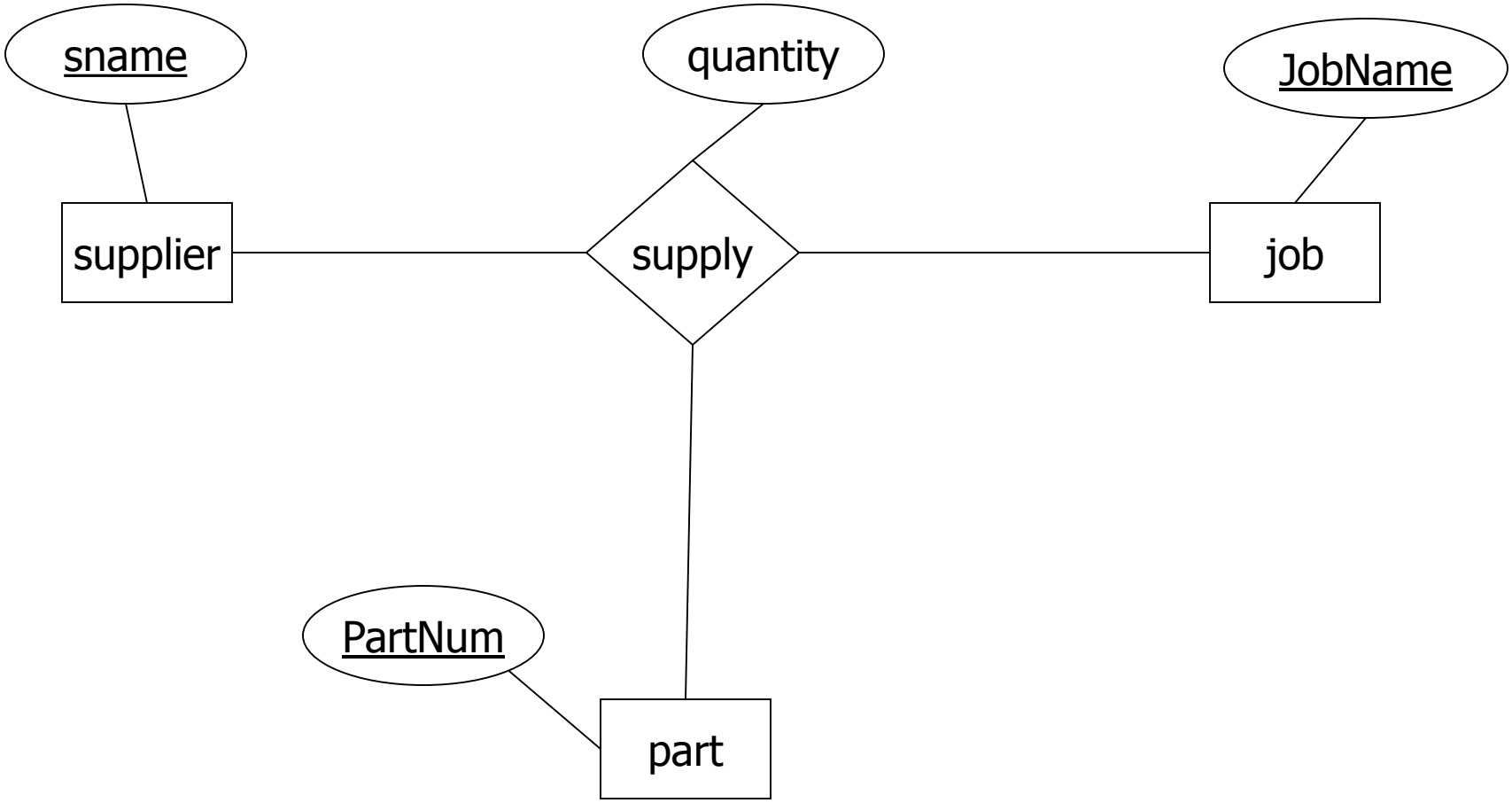
Step 6 – Multi-valued attributes

- For each multi-valued attribute A
  - create a relation R that contains the attribute A and the primary key of the relation S that corresponds to the entity type that contained A

Step 7: n-ary Relationship types

- For each n-ary relationship type L (n>2)
  - create a new relation R to represent L
  - include as foreign keys in R the primary keys of all relations that correspond to the entity types participating in L
  - the primary key of R is the combination of all these foreign keys
  - include any simple attributes or simple components of L as attributes of R

N-ary example



Solution to n-ary example

Tables that already exist:

Supplier (sname, …)

Job (jname, …)

Part (partNum, …)

Add new Table:

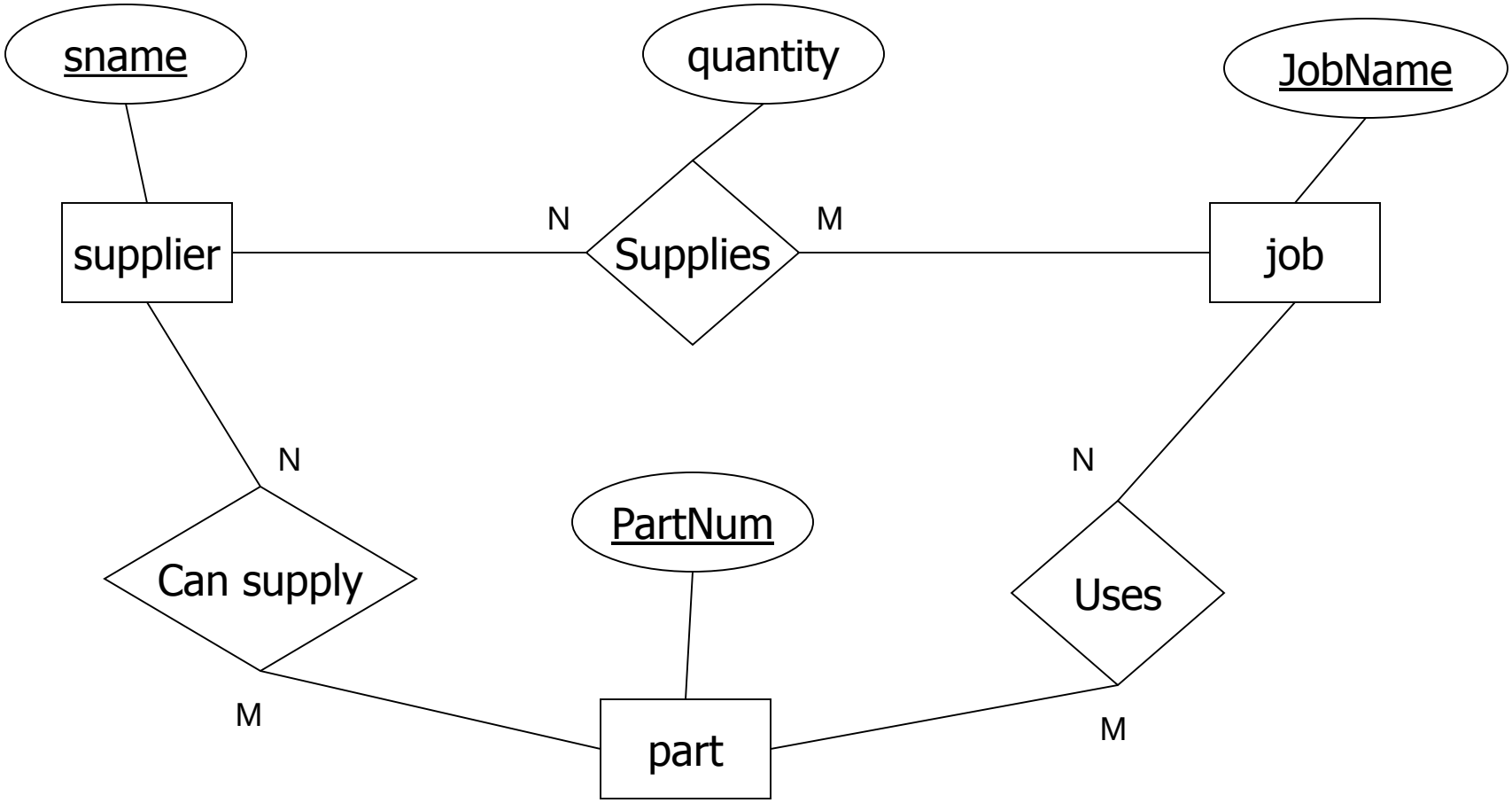
Supply (sname, jname, partNum, quantity, …)

sname is FK -> Supplier (sname)

jname is FK -> Job (jname)

partNum is FK -> Part (partNum)

Binary version of n-ary example



Solution to binary version of n-ary

Supplier (sname, …)

Job (jname, …)

Part (partNum, …)

Supplies (sname, jname)

Can_supply (sname, partNum)

Uses (partNum, jname)

Compare the two solutions…

N-ary Solution:

Supply (sname, jname, partNum, quantity, …)

Binary Solution:

Supplies (sname, jname) (supplier can supply to a job)

Can_supply (sname, partNum) (supplier can supply a part)

Uses (partNum, jname) (job uses a part)

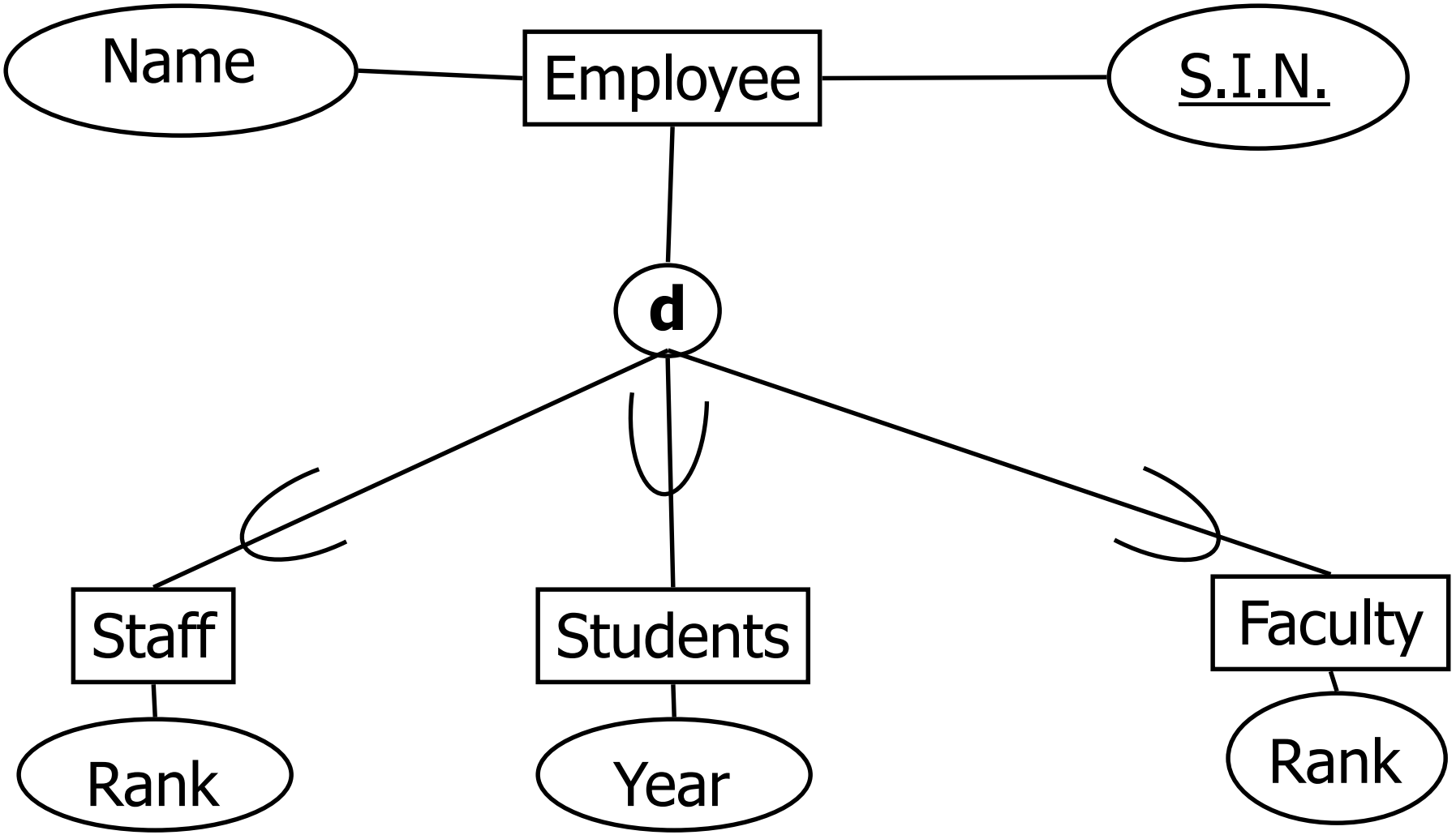
(s,j), (s,p), (p,j) != (s, p, j)

Step 8 - Specializations

- For each specialization with subclasses **{S_1 , S_2 , …, S_m }** and superclass **C** where attributes of **C** are **{k, a_1 , a_2 , …, a_n }** and **k** is the primary key

- Convert using one of 4 options:

Example for 8a-c



Step 8 – Option 8a

- create relation **L** for **C** where

  Attr(L) = {k, a_1 , a_2 , …, a_n} and

  PK(L) = k
- create relation **L_i** for each **S_i** where **Attr(L_i) = {k}** ∪ **{attributes of S_i}** and **PK(L_i) = k**

Step 8 – Option 8b

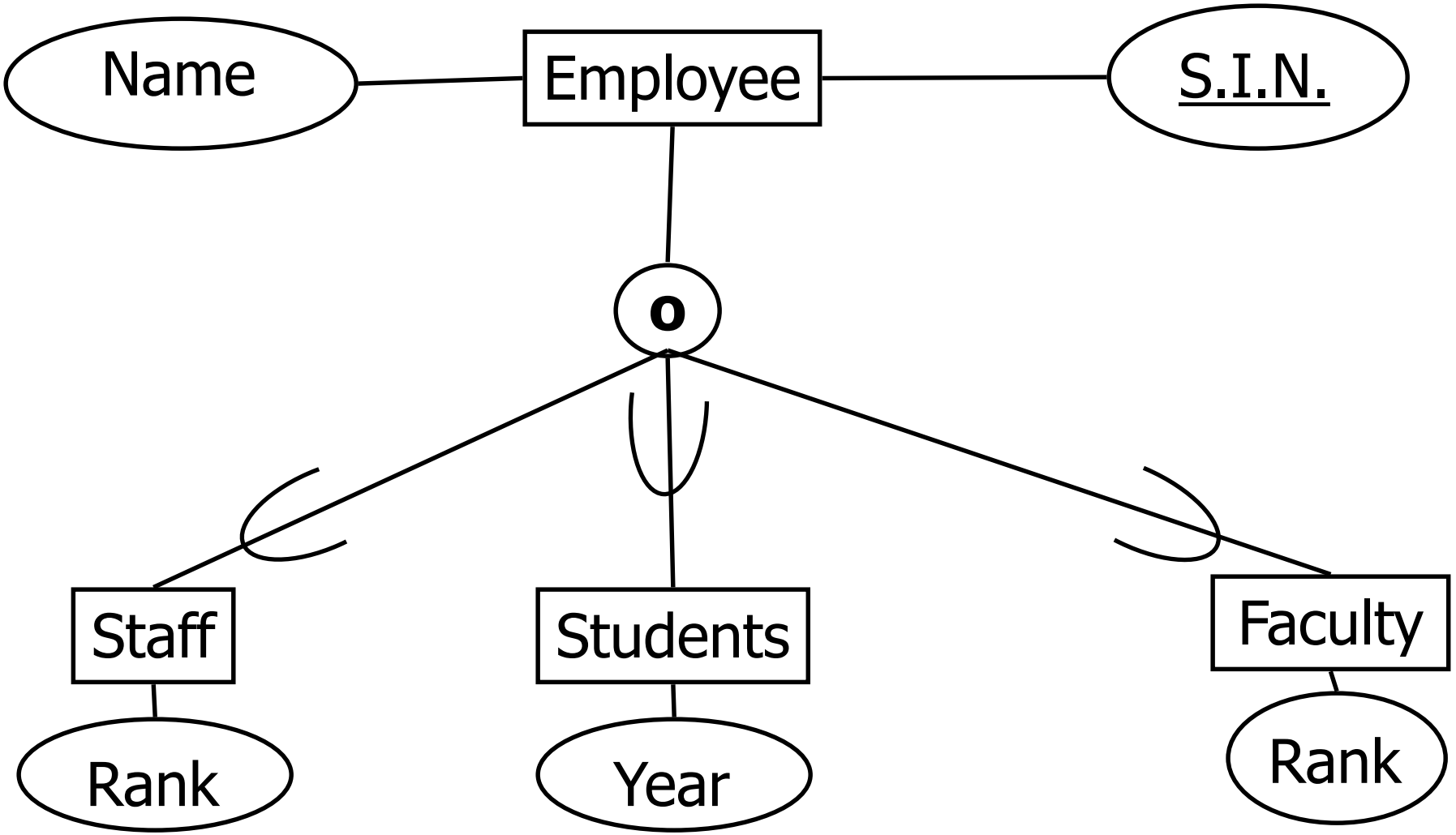
- create relation $\mathbf{L_i}$ for each $\mathbf{S_i}$ where $\mathbf{Attr(L_i) = \{k, a_1 , a_2 , \ldots, a_n\}}$ $\cup$ **{attributes of S_i}** and $\mathbf{PK(L_i) = k}$

Step 8 – Option 8c

- for cases where subclasses are disjoint
- create relation **L** where **Attrs(L) = {k, a_1 , a_2 , …, a_n}** U **{attributes of S_1}** U **{attributes of S_2}** U **…** U **{attributes of S_m}** U **{t}** and

  PK(L) = k, where **t** is a type (or discriminating) attribute

Example for 8d



Step 8 – Option 8d

- for cases where subclasses are overlapping
- create relation **L** where **Attrs(L) = {k, a_1 , a_2 , …, a_n}** U **{attributes of S_1}** U **{attributes of S_2}** U **…** U **{attributes of S_m}** U **{t_1, t_2, …, t_m}** and **PK(L) =** k, where **t_i** is a boolean attribute indicating whether a tuple belongs to S_i