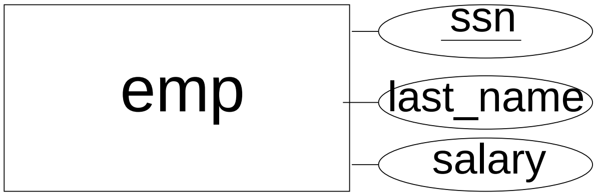
Converting an ER Schema

- Focus first on the entities, then the relationships, and finally the generalization hierarchies.
- Use the following guidelines...

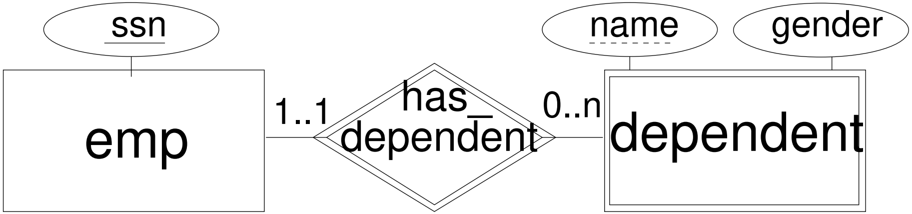
Strong Entity



- Create a relation;
  Copy the simple attributes and basic components of a composite attribute.
  Choose one of the key attributes as PK.

emp(<u>ssn</u>, last_name, salary)

Weak Entity plus identifying relationship

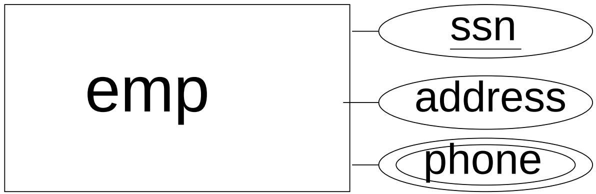


Create a relation
Copy attributes as before...
Add Owner's PK as FK.
Primary key is that FK plus partial key(s)

dependent(<u>ssn</u>, <u>name</u>, sex)

FK: ssn refers to emp.ssn.

- *Multi-valued Attribute* A of an entity E:



Create a separate relation.
Attributes: A plus PK of (the reln. corresp. to) E.

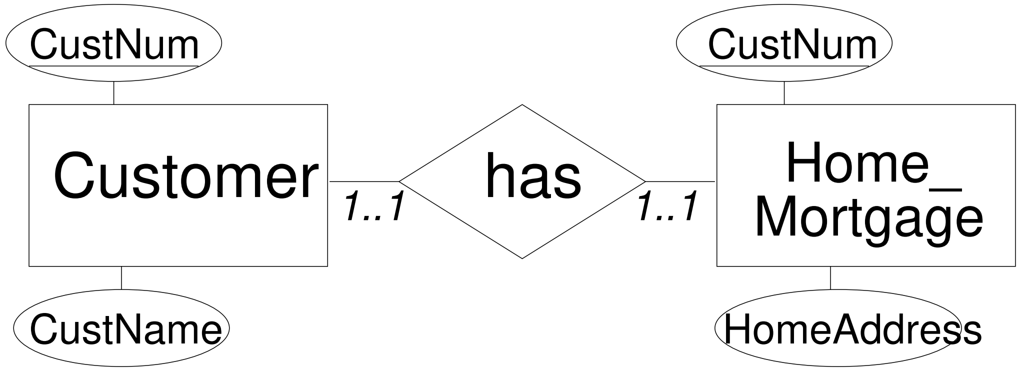
phone(essn, phone_number)

What about PK? ... and FK

1-1 Binary Relationship R between E_1 and E_2

- *Total* participation of *both* entities
  $\Rightarrow$ collapse into one relation?
  Yes, if they do not participate in other relationships or do participate in those that can benefit from such integration.
- *Same PK* for both $\Rightarrow$ choose *it* as *the* PK
- *Different PKs* $\Rightarrow$ choose one as PK

1-1 Binary Relationship R between E_1 and E_2



Replace Customer(CustNum, CustName) and Home_Mortgage(CustNum, HomeAddress) by

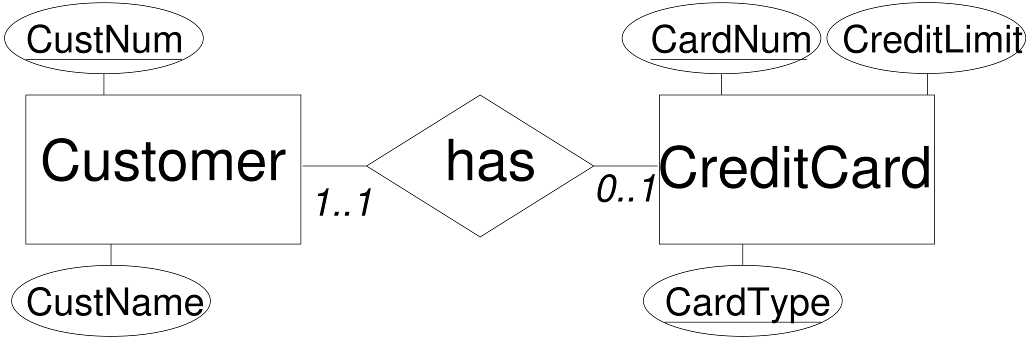
CustomerHomeMortgage(CustNum,
CustName, HomeAddress)

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

1-1 Binary Relationship R between E_1 and E_2

- Only one entity E_1 has partial participation
  - Let the other entity E_2 represent R
  - Attributes: Add PK of E_1 as FK plus simple attributes of R.

1-1 Binary Relationship R between E_1 and E_2



Customer(<u>CustNum</u>, CustName)

CreditCard(<u>CardNum, CardType,</u>

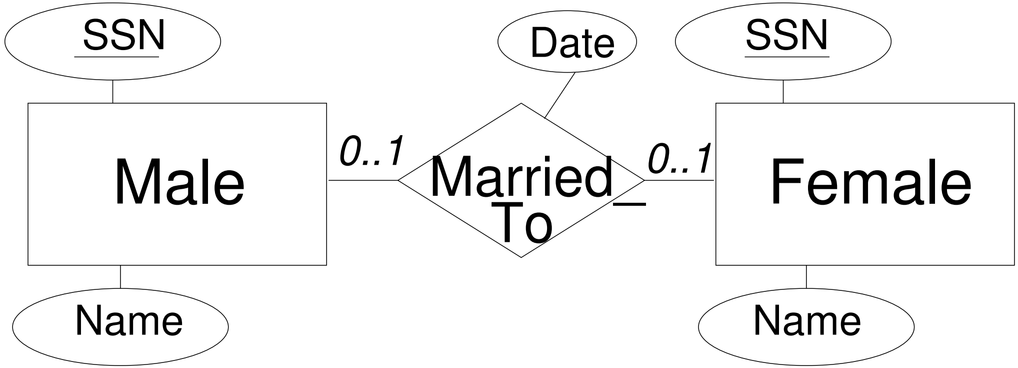
CreditLimit, CustNum)

FK: CustNum refers to Customer.CustNum.

1-1 Binary Relationship R between E_1 and E_2

- Both entities have partial participation
  - Copy PKs from both and choose one as the PK.

1-1 Binary Relationship R between E_1 and E_2

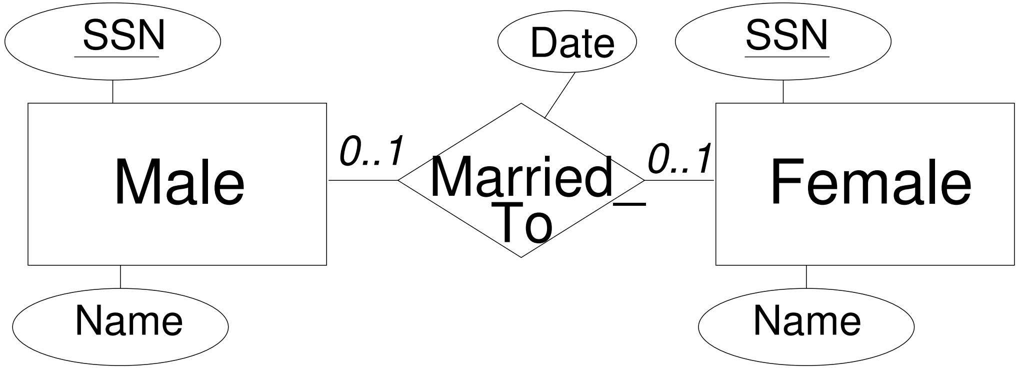


Male(<u>MaleSSN</u>, Name)

Female(<u>FemaleSSN</u>, Name)

Married_To(<u>MaleSSN</u>, FemaleSSN, Date)

Unique constraint on *FemaleSSN*



OR

Male(<u>MaleSSN</u>, Name)

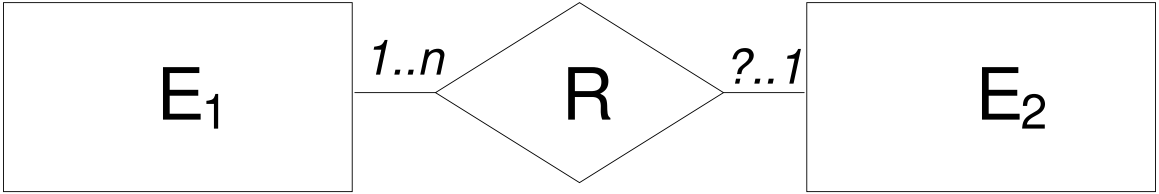
Female(<u>FemaleSSN</u>, Name)

Married_To(MaleSSN, <u>FemaleSSN</u>, Date)

Unique constraint on *MaleSSN*

Many-to-One Binary Relationship

(One-to-many / many-to-one)



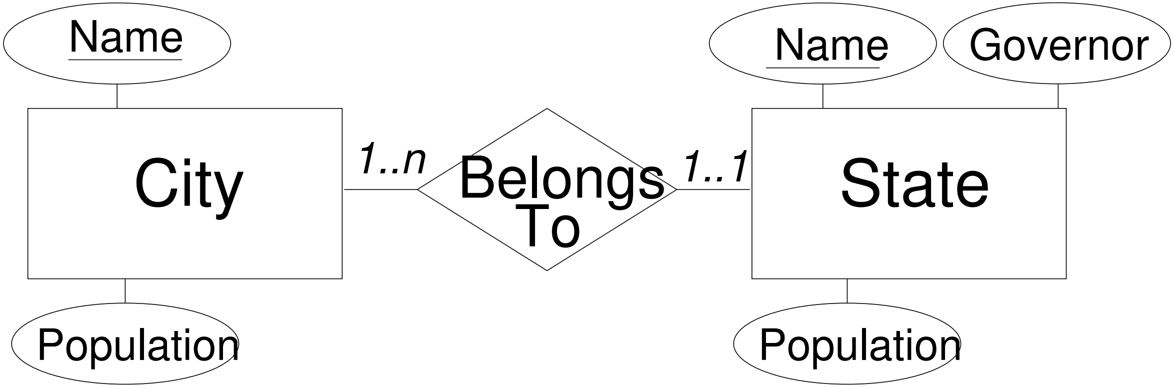
- We will consider two values of *?* :
  1 and 0, i.e.,
  E_1's participation is *total* and *partial* respectively.

NEW MEXICO TECH
SCIENCE • ENGINEERING • RESEARCH UNIVERSITY

Many-to-One Binary Relationship

1. E_1 has total participation
   - Let E_1 represent R:
   - Attributes: PK of E_2 as FK plus simple attributes of R.

Many-to-One Binary Relationship



State(StateName, Population, Governor)

City(CityName, Population, StateName)

FK: StateName refers to State.StateName

What about the min. cardinality of 1 for State?

Many-to-One Binary Relationship

It has not been captured.
For it, we need an *Inclusion Dependency* constraint

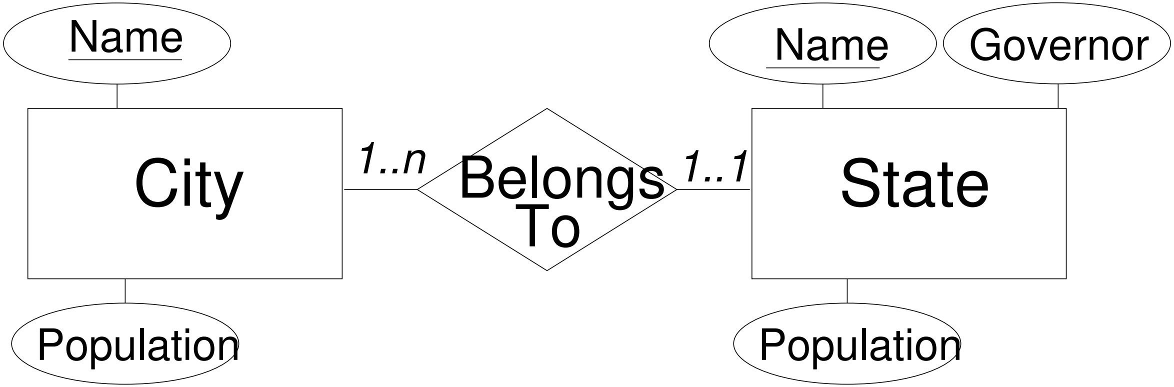
$$\text{State.StateName} < \text{City.StateName}$$

which translates into

$$\Pi_{StateName}\text{State} \subseteq \Pi_{StateName}\text{City}$$

Putting it all together...

Many-to-One Binary Relationship



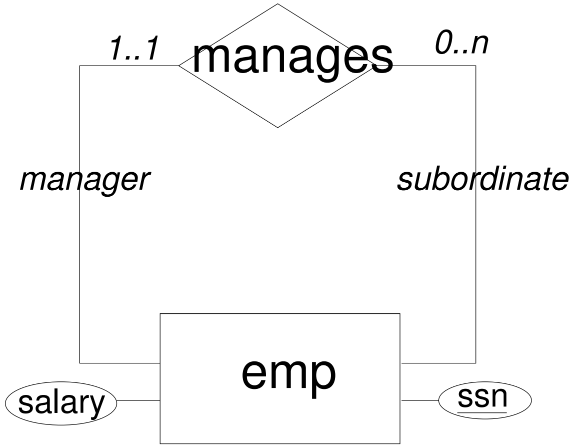
State(<u>StateName</u>, Population, Governor)

City(<u>CityName</u>, Population, StateName)

FK: StateName refers to State.StateName

Incl. Dep.: State.StateName $<$ City.StateName

Many-to-One Binary Relationship



emp(<u>ssn</u>, salary, mgrssn)

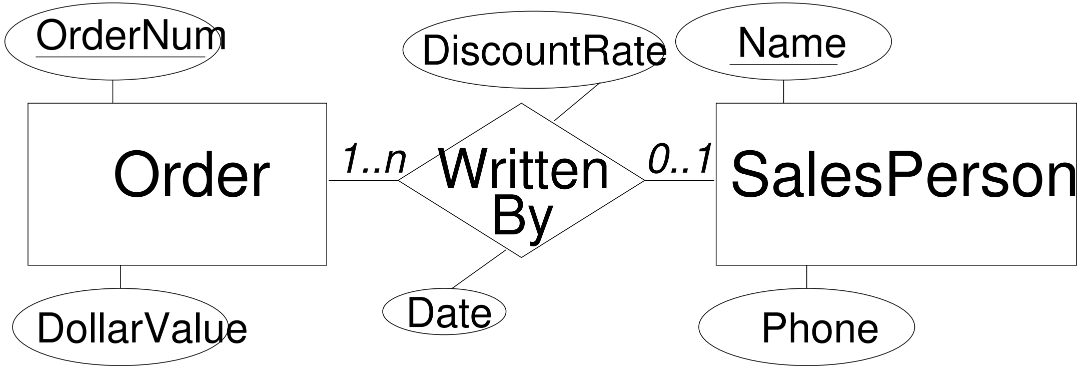
FK: mgrssn refers to emp.ssn

Many-to-One Binary Relationship

2. E_1 has partial participation

- Previous scheme $\Rightarrow$ null values are possible.
- These null values are not acceptable
  $\Rightarrow$ define a separate relation with PK from E_1.
  Add attributes of R

Many-to-One Binary Relationship



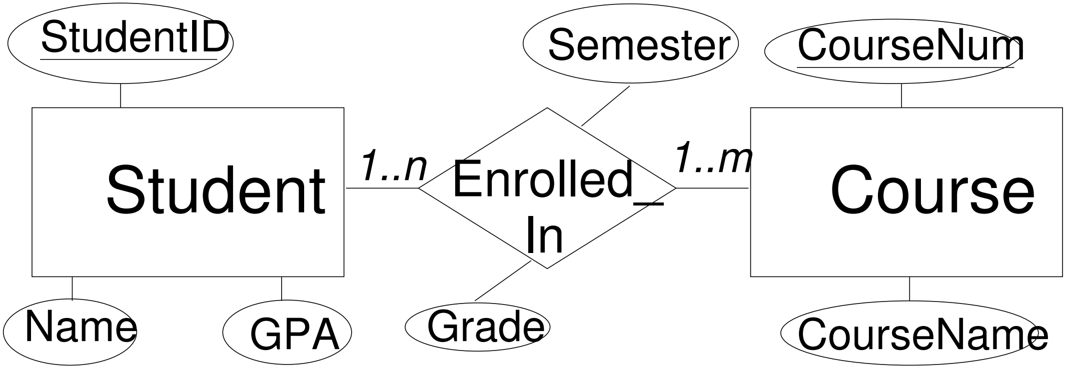
SalesPerson(<u>SalesPersonName</u>, Phone)
Order(<u>OrderNum</u>, DollarValue)
OrderWritten(<u>OrderNum</u>, SalesPersonName,
DiscountRate, Date)

SalesPerson.SalesPersonName $<$ OrderWritten.SalesPersonName
(Inclusion Dependency)

Many-to-Many Binary relationship

- Create a new relation.
  Copy PKs of both entities, each as FK.
  Together they form the PK.
  Add attributes of R.

Many-to-Many Binary relationship



Student(<u>StudentID</u>, Name, GPA)
Course(<u>CourseNum</u>, CourseName)
Enrolment(<u>StudentID, CourseNum,</u> Semester, Grade)

We also need 2 inclusion dependencies for the two min. cardinalities of 1...

Many-to-Many Binary relationship

Student(<u>StudentID</u>, Name, GPA)

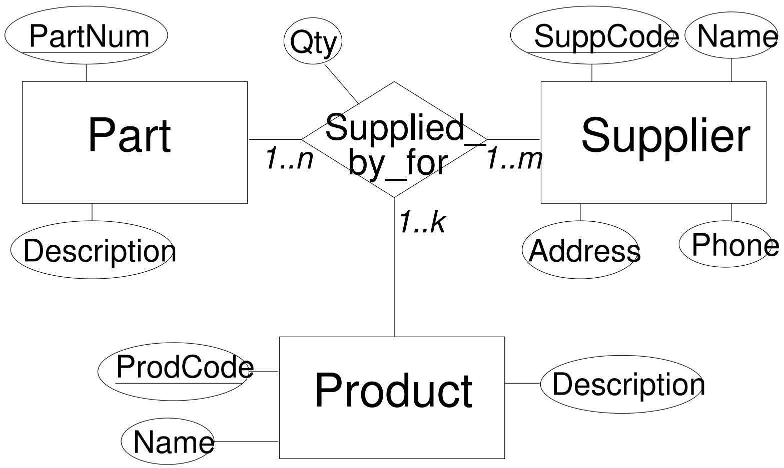
Course(<u>CourseNum</u>, CourseName)

Enrolment(<u>StudentID, CourseNum,</u> Semester, Grade)

Student.StudentID $<$ Enrolment.StudentID

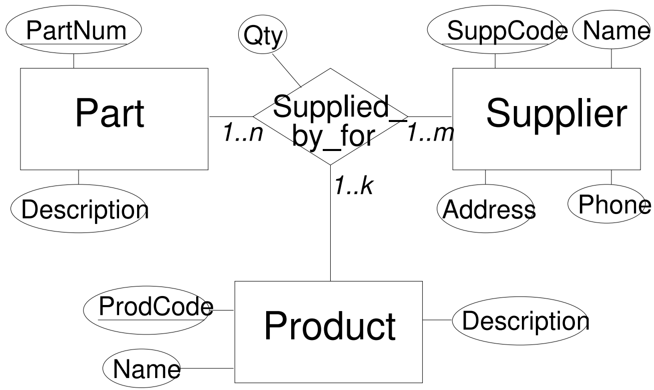
Course.CourseNum $<$ Enrolment.CourseNum

N-ary Relationship ($N > 2$)



- Follow the strategy of Many-to-Many relationships.

N-ary Relationship ($N > 2$)



Part(PartNum, Description)

Supplier(SuppCode, Name, Address, Phone)

Product(ProdCode, Name, Description)

Supply(PartNum, ProdCode, SuppCode, Qty)

Plus 3 inclusion dependencies ...

N-ary Relationship ($N > 2$)

Part(<u>PartNum</u>, Description)

Supplier(<u>SuppCode</u>, Name, Address, Phone)

Product(<u>ProdCode</u>, Name, Description)

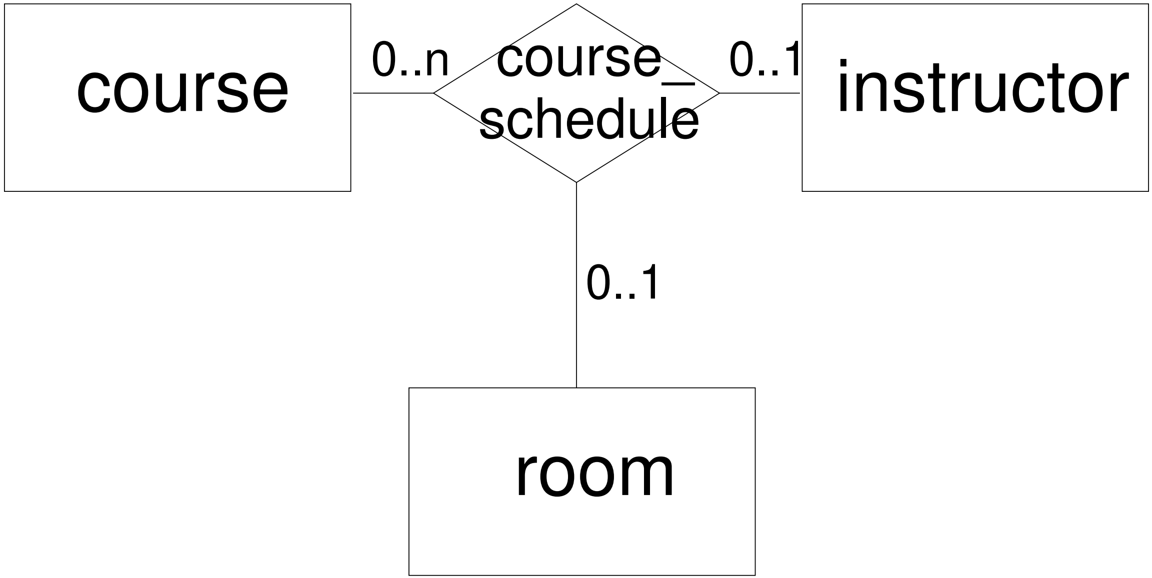
Supply(<u>PartNum, ProdCode, SuppCode</u>, Qty)

Part.PartNum < Supply.PartNum

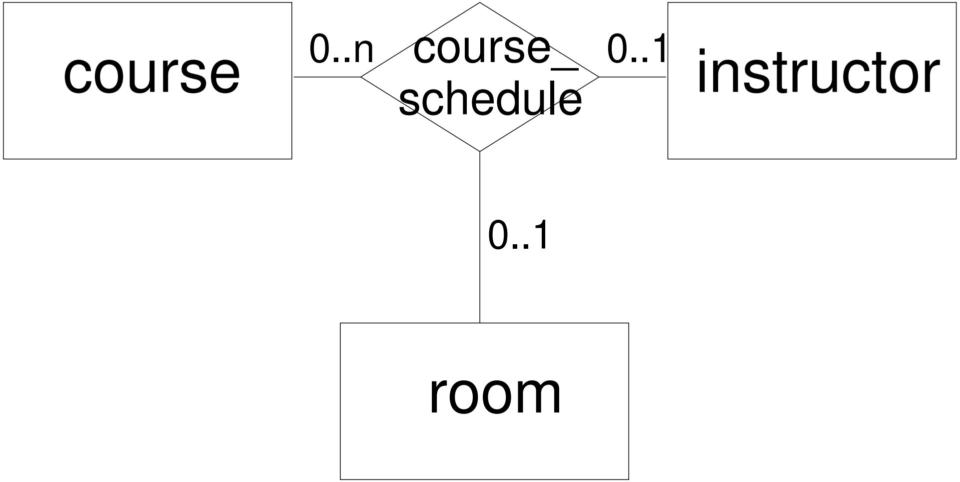
Product.ProdCode < Supply.ProdCode

Supplier.SuppCode < Supply.SuppCode

Special case 1

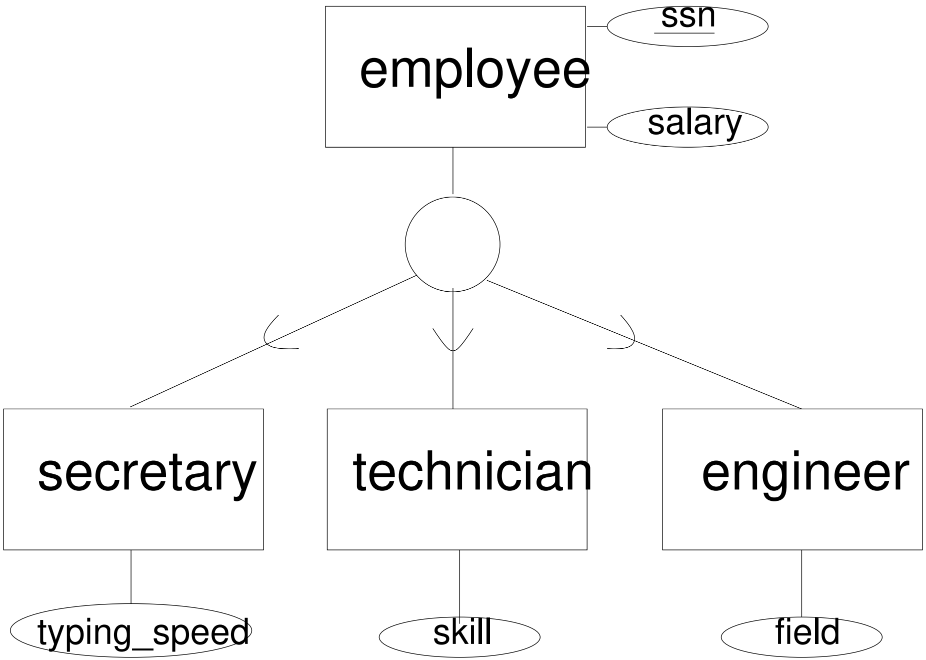


Special case 2



Additional constraint: every course is taught by exactly one instructor in exactly one room.

Generalization Hierarchies

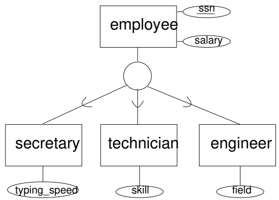


Problem

- Generalization Hierarchy must be eliminated.
- How?

Approach 1: E/R Style

- Keep all the entities:



emp(<u>ssn</u>, salary)
secy(<u>secy_ssn</u>, typing_speed) FK: secy_ssn refers to emp.ssn
tech(<u>tech_ssn</u>, skill) FK: tech_ssn refers to emp.ssn
engr(<u>engr_ssn</u>, field) FK: engr_ssn refers to emp.ssn

Approach 1: E/R Style

Also need disjointness constraints:

R.A disjoint from S.B

translates into

$$\Pi_A(R) \cap \Pi_B(S) = \emptyset$$

Approach 1: E/R Style

emp(<u>ssn</u>, salary)

secy(<u>secy_ssn</u>, typing_speed) FK: secy_ssn refers to emp.ssn

tech(<u>tech_ssn</u>, skill) FK: tech_ssn refers to emp.ssn

engr(<u>engr_ssn</u>, field) FK: engr_ssn refers to emp.ssn

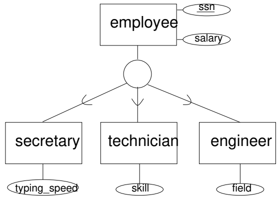
secy.secy_ssn disjoint from tech.tech_ssn
tech.tech_ssn disjoint from engr.engr_ssn
engr.engr_ssn disjoint from secy.secy_ssn

Approach 2: OO style

- A relation per possible subtree of the hierarchy

Approach 2a

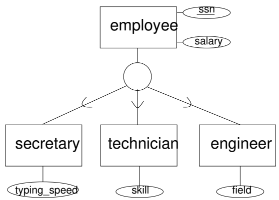
- When is the following appropriate?



secy(<u>secy_ssn</u>, salary, typing_speed)
tech(<u>tech_ssn</u>, salary, skill)
engr(<u>engr_ssn</u>, salary, field)

secy.secy_ssn disjoint from tech.tech_ssn
tech.tech_ssn disjoint from engr.engr_ssn
engr.engr_ssn disjoint from secy.secy_ssn

- When is the following appropriate?

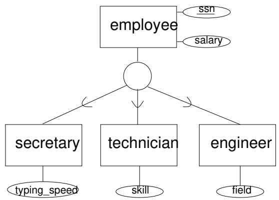


other_emp(<u>otherssn</u>, salary)
secy(<u>secy_ssn</u>, salary, typing_speed)
tech(<u>tech_ssn</u>, salary, skill)
engr(<u>engr_ssn</u>, salary, field)
Plus 6 disjointness constraints...

Approach 2c...

- How many subtrees can there be?

- Collapse into a single relation:

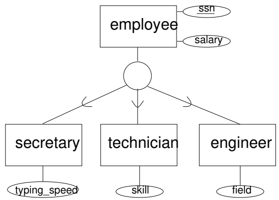


emp(<u>ssn</u>, salary, kind_of,
typing_speed, skill, field)

where kind_of$\in \{0, 1, 2, 3, \text{NULL}\}$

Approach 4

- Collapse into a single wide relation:



emp(<u>ssn</u>, salary, typing_speed, skill,
field, is_secy, is_tech, is_engr)
where is_xxxx: boolean

Given: a relationship R between superclasses S_1, S_2.

Implementation has a table for S_1?	Implementation has a table for S_2?	(Recursively) replace R by a copy between
Yes	Yes	N/A
Yes	No	S_1 and each child of S_2
No	Yes	S_2 and each child of S_1
No	No	each child of S_1 with each child of S_2

Question: Total coverage?