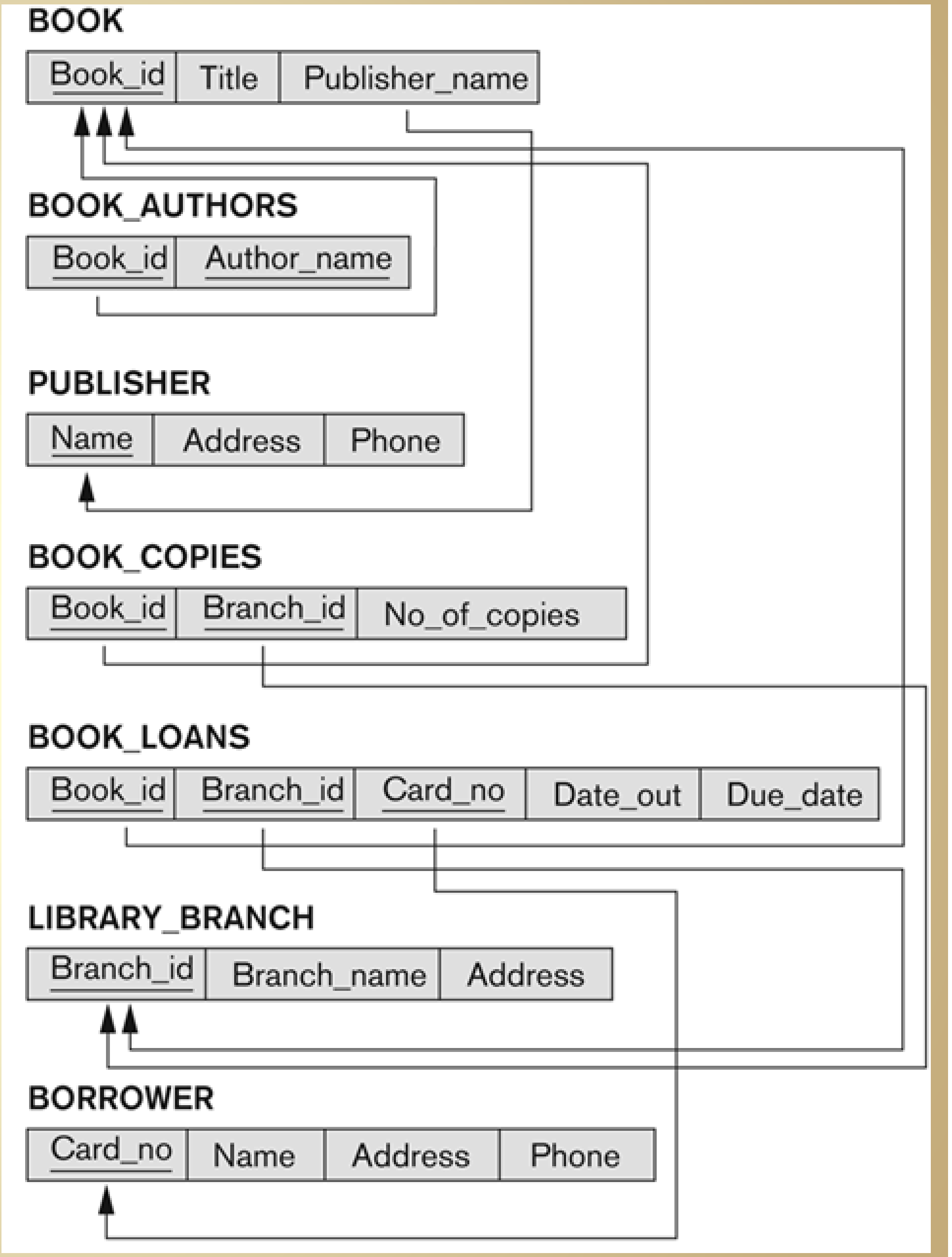
COMP163

Database Management Systems

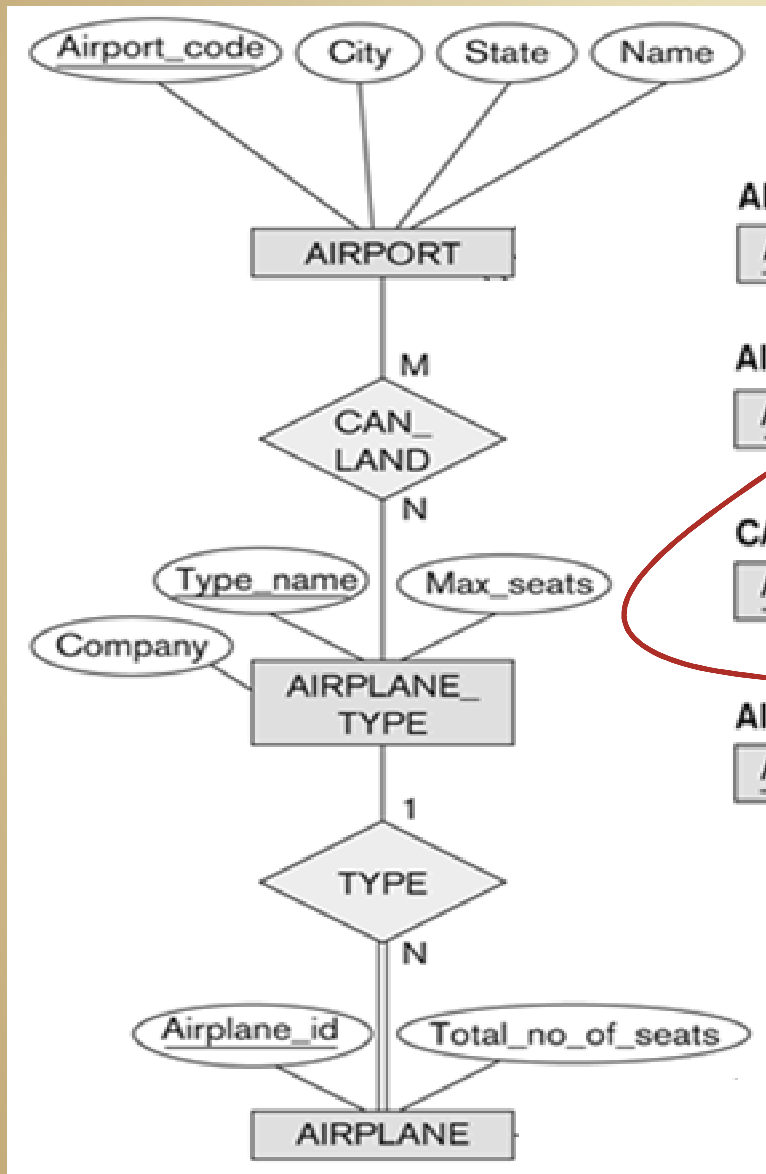
Lecture 5 – Chapter 9
ER to Relational Schema Translation

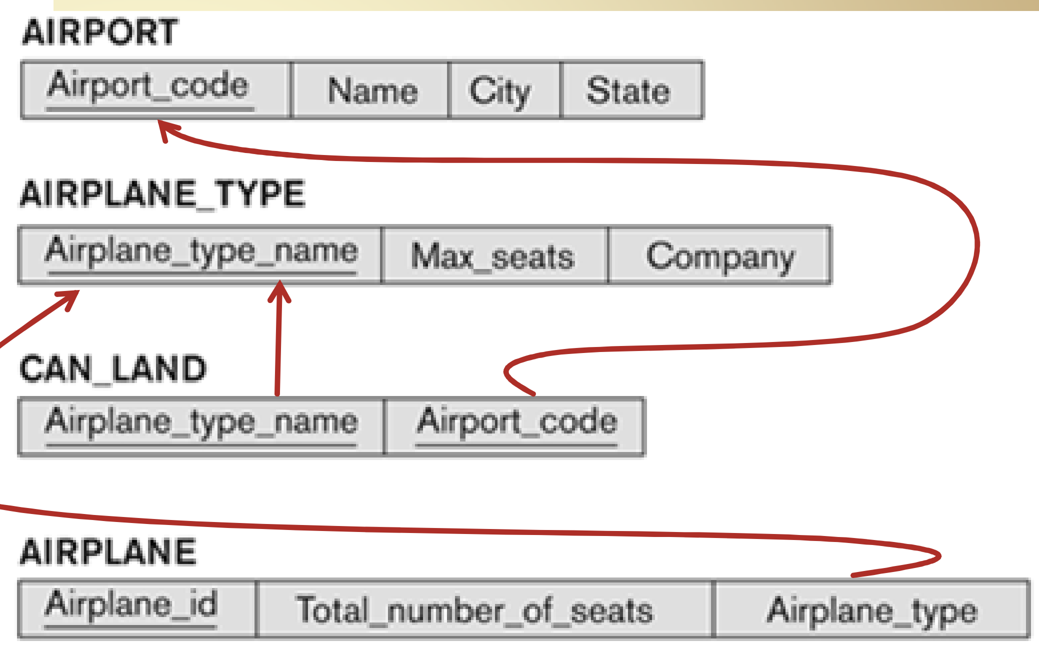
REVIEW

Reverse engineer this relational schema to find an equivalent ER schema.

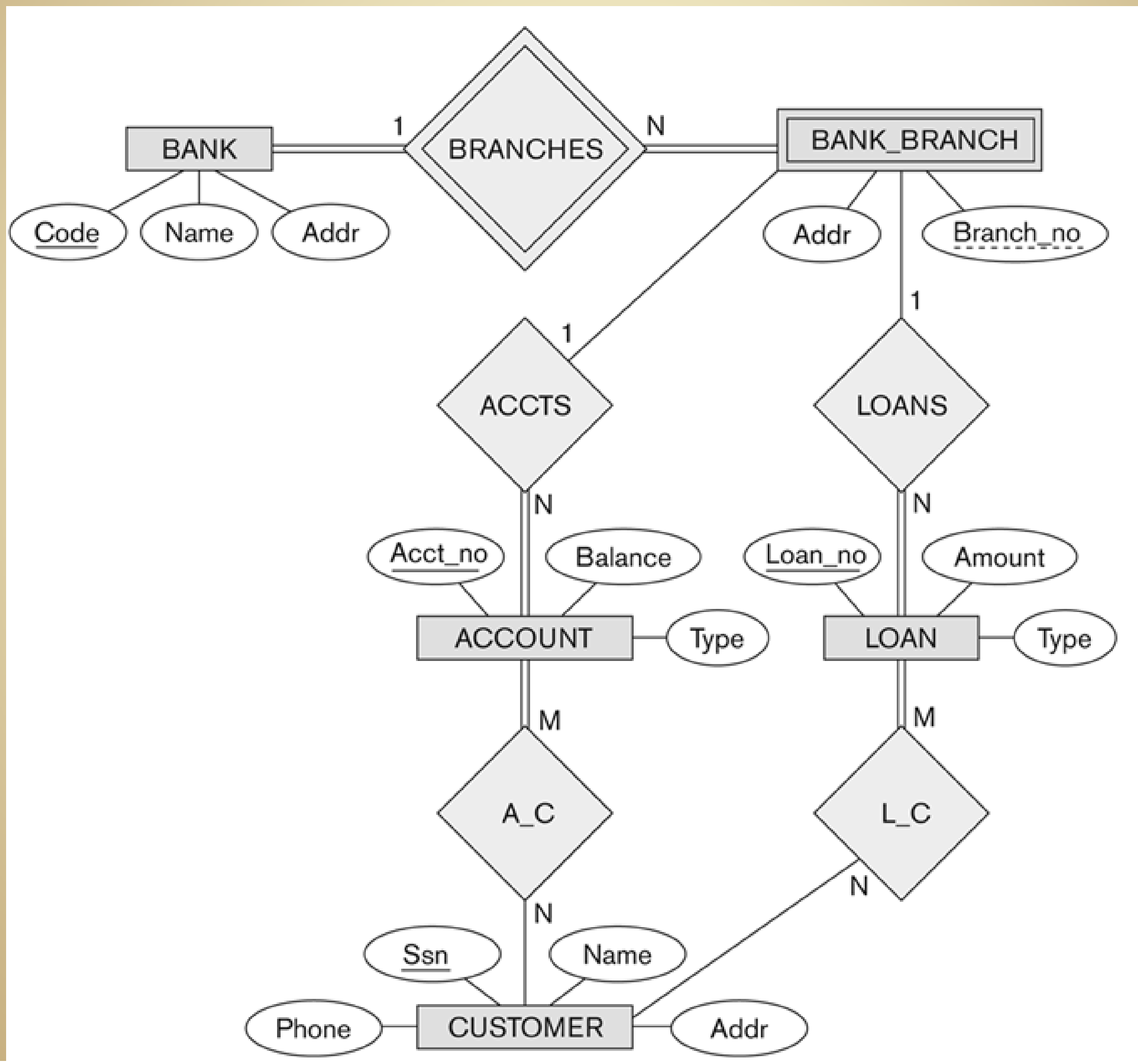


PREVIEW: ER to Relational



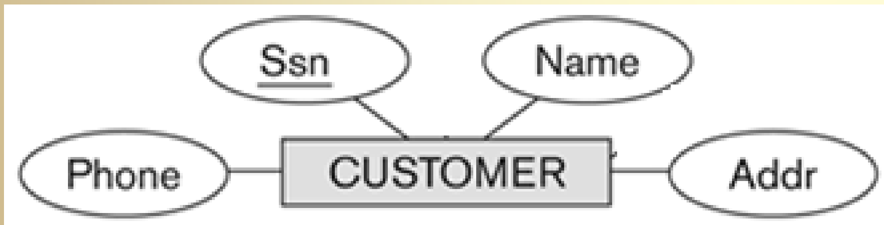


EER Bank Schema



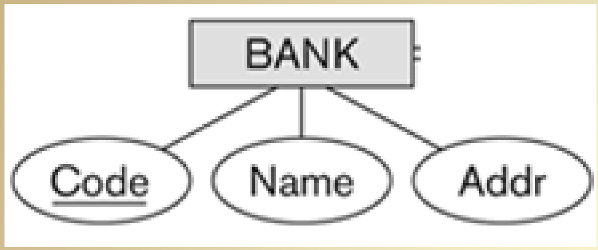
Step 1: Regular Entities

- Regular entity types become relations
  - include all simple attributes
  - include only components of compound attributes
  - keys become primary keys
  - if multiple keys (candidate keys) select a primary key

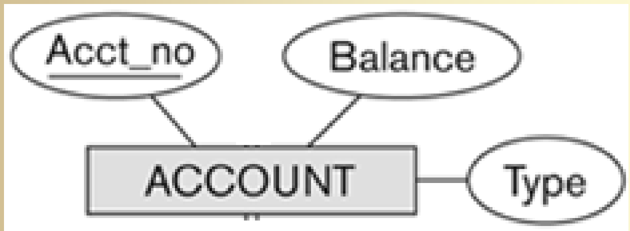


CUSTOMER(<u>Ssn</u>, Name, Addr, Phone)

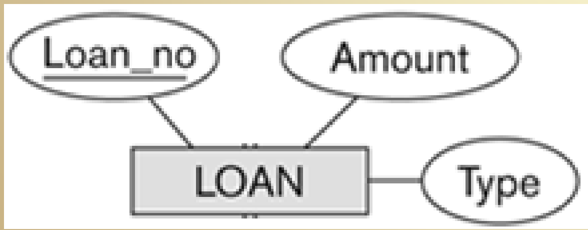
Step 1: Regular Entities



BANK(<u>Code</u>, Name, Addr)



ACCOUNT(<u>Acct_no</u>, Type, Balance)



LOAN(<u>Loan_no</u>, Type, Amount)

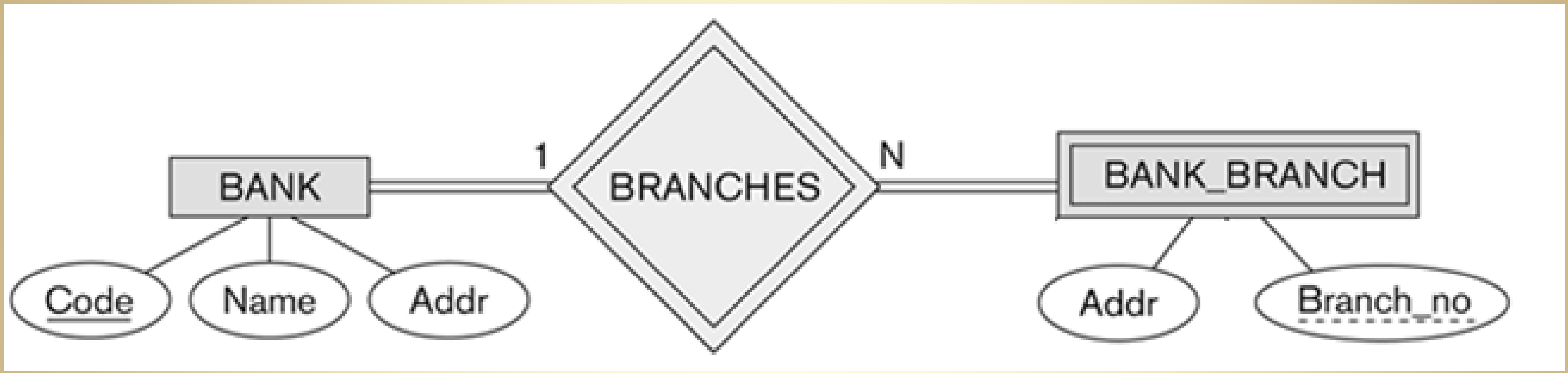
Step 2: Weak Entities

- Weak entity types become relations
  - include all simple attributes
  - include only components of compound attributes
  - create a primary key from partial key
    and key of owning entity type (through identifying relationship)
  - attributes acquired through identifying relationship
    become a foreign key*

* typically, deletions and insertions will be propagated through this foreign key

Step 2: Weak Entities

- Weak entity types become relations



BANK_BRANCH(<u>Bank_code</u>, <u>Branch_No</u>, Addr)

FK

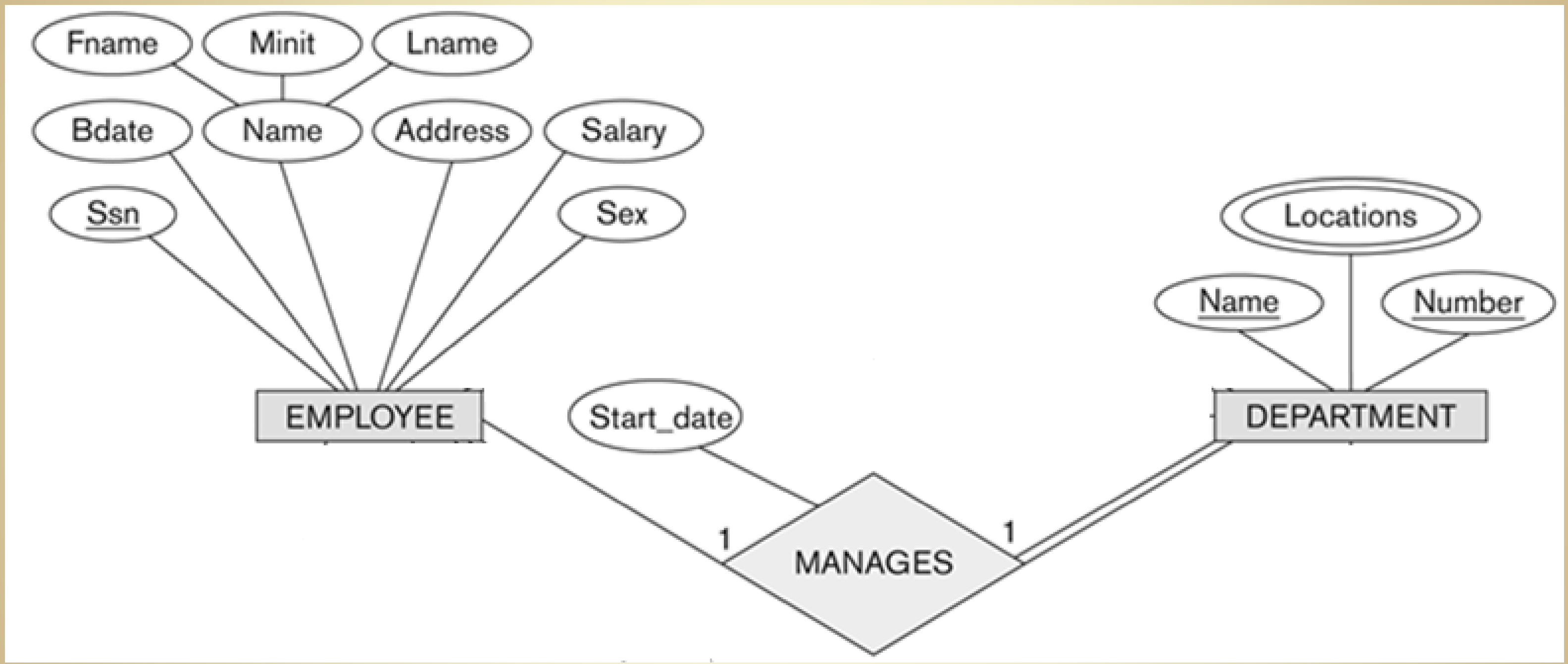
BANK(<u>Code</u>, Name, Addr)

Step 3: Binary 1:1 Relationships

- Approach 1: Foreign Key
  - Chose one of the related entity types to hold the relationship (chose one with total participation, if possible)
  - add FK to other relation
  - move all relationship attributes to this relation
  - *this approach is preferable, except as noted below*
- Approach 2: Merged Relation
  - combine the relations for the related entities into a single relation
  - *use only when both participations are total*
- Approach 3: Separate Relation
  - same as binary M:N relationship (see step 5)
  - *not generally a good option*

Step 3: Binary 1:1 Relationships

- Approach 1: Foreign Key



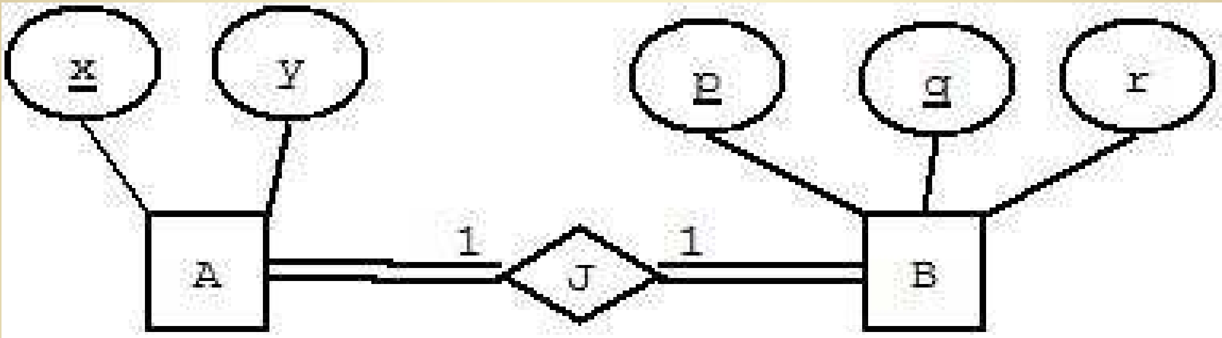
EMPLOYEE(<u>Ssn</u>, Name, …)

FK

DEPARTMENT(<u>Name</u>, <u>Number</u>, Mgr, Mgr_start_date)

Step 3: Binary 1:1 Relationships

- Approach 2: Merged Relation



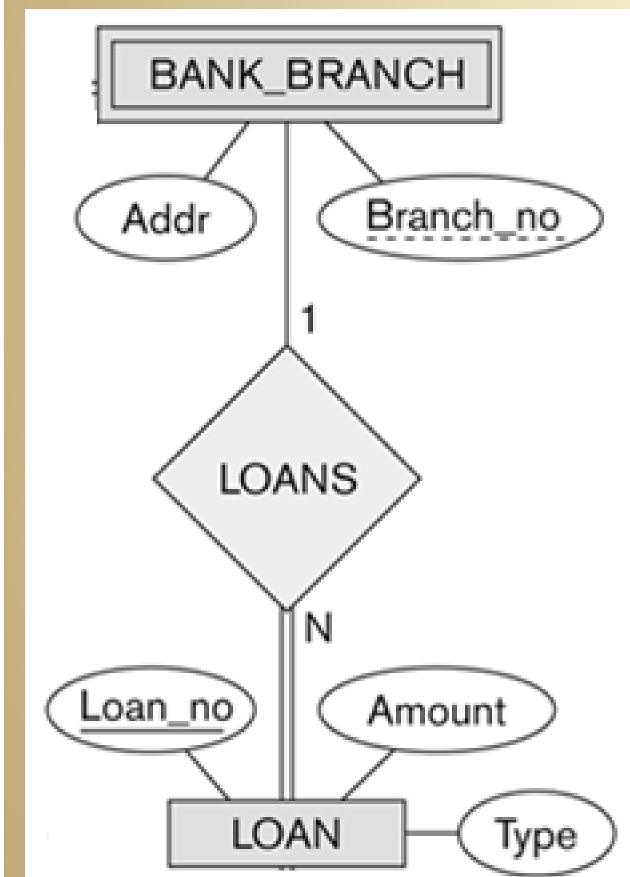
AJB(<u>x</u>, y, p, q, r)

or

AJB(x, y, <u>p</u>, <u>q</u>, r)

Step 4: Binary 1:N Relationships

- 1:N Relationships become foreign key at N side
  - any relationship attributes also go to N side

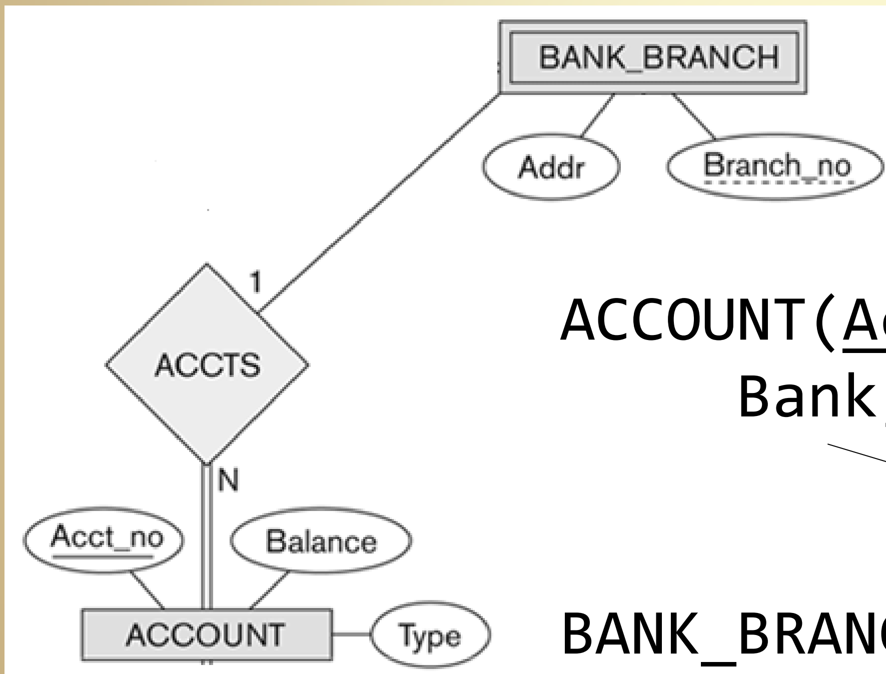


LOAN(Loan_no, Type, Amount,
Bank, Branch)

BANK_BRANCH(Bank_code, Branch_No,
Addr)

Step 4: Binary 1:N Relationships

- 1:N Relationships become foreign key at N side
  - any relationship attributes also go to N side

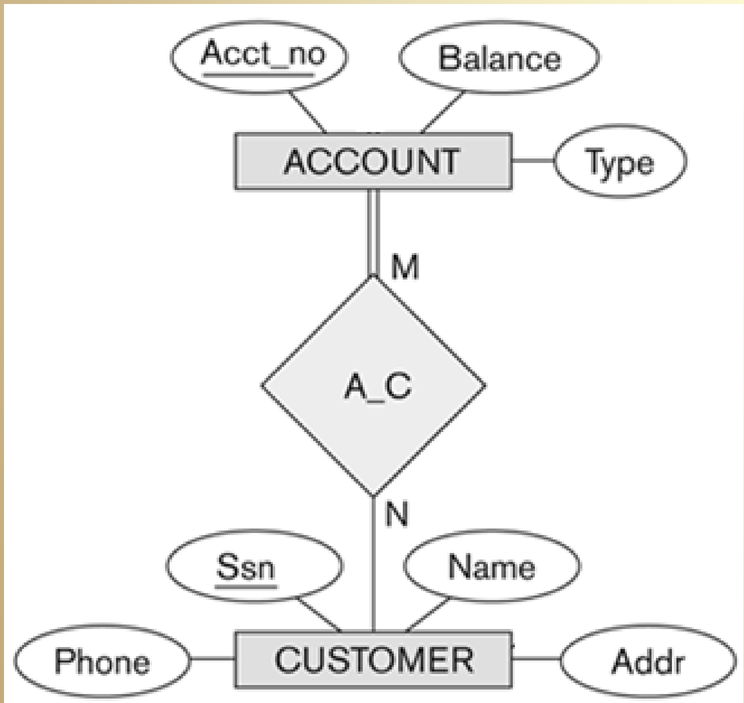


ACCOUNT(<u>Acct_no</u>, Type, Balance,
Bank, Branch)

BANK_BRANCH(<u>Bank_code</u>, <u>Branch_No</u>,
Addr)

Step 5: Binary M:N Relationships

- M:N Relationships must become a new relation
  - contains FKs to both related entities
  - combined FKs become PK for new relations
  - relationship attributes go in new relation



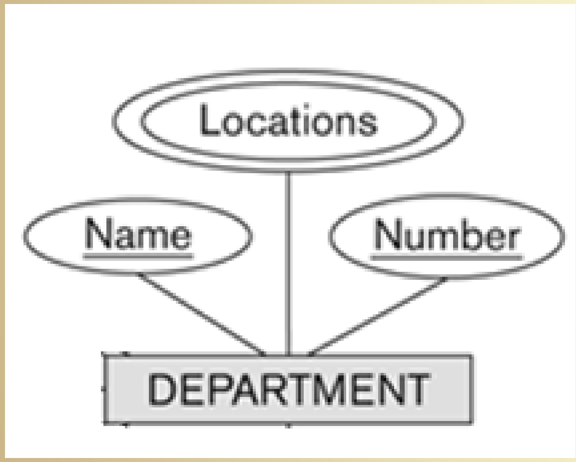

```
CUSTOMER(Ssn, Name, Addr,
              Phone)

      A_C(Acct, Cust)

ACCOUNT(Acct_no, Type,
    Balance, Bank, Branch)
```

Step 6: Multivalued Attributes

- Multivalued attributes must become new relations
  - FK to associated entity type
  - PK is whole relation

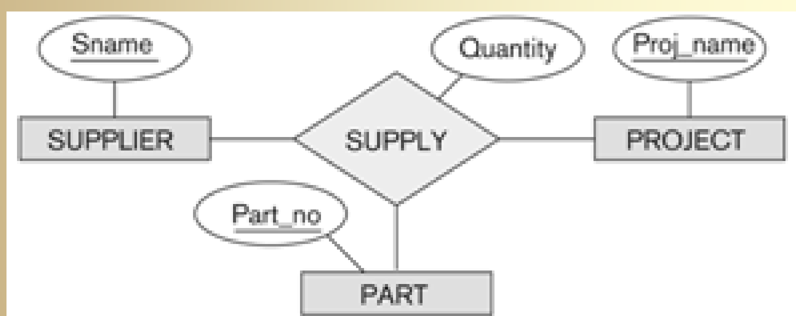


DEPARTMENT(<u>Name</u>, <u>Number</u>, Mgr, Mgr_start_date)

DEPT_LOCATIONS(<u>DName</u>, <u>Dno</u>, <u>Location</u>)

Step 7: N-ary Relationships

- Non-Binary Relationships become new relations
  - FKs to all participating entity types
  - Combine FKs to make a PK
    (exclude entities with max participation of 1)
  - Include any relationship attributes



SUPPLIER(SName)

PROJECT(Proj_name)

PART(Part_no)

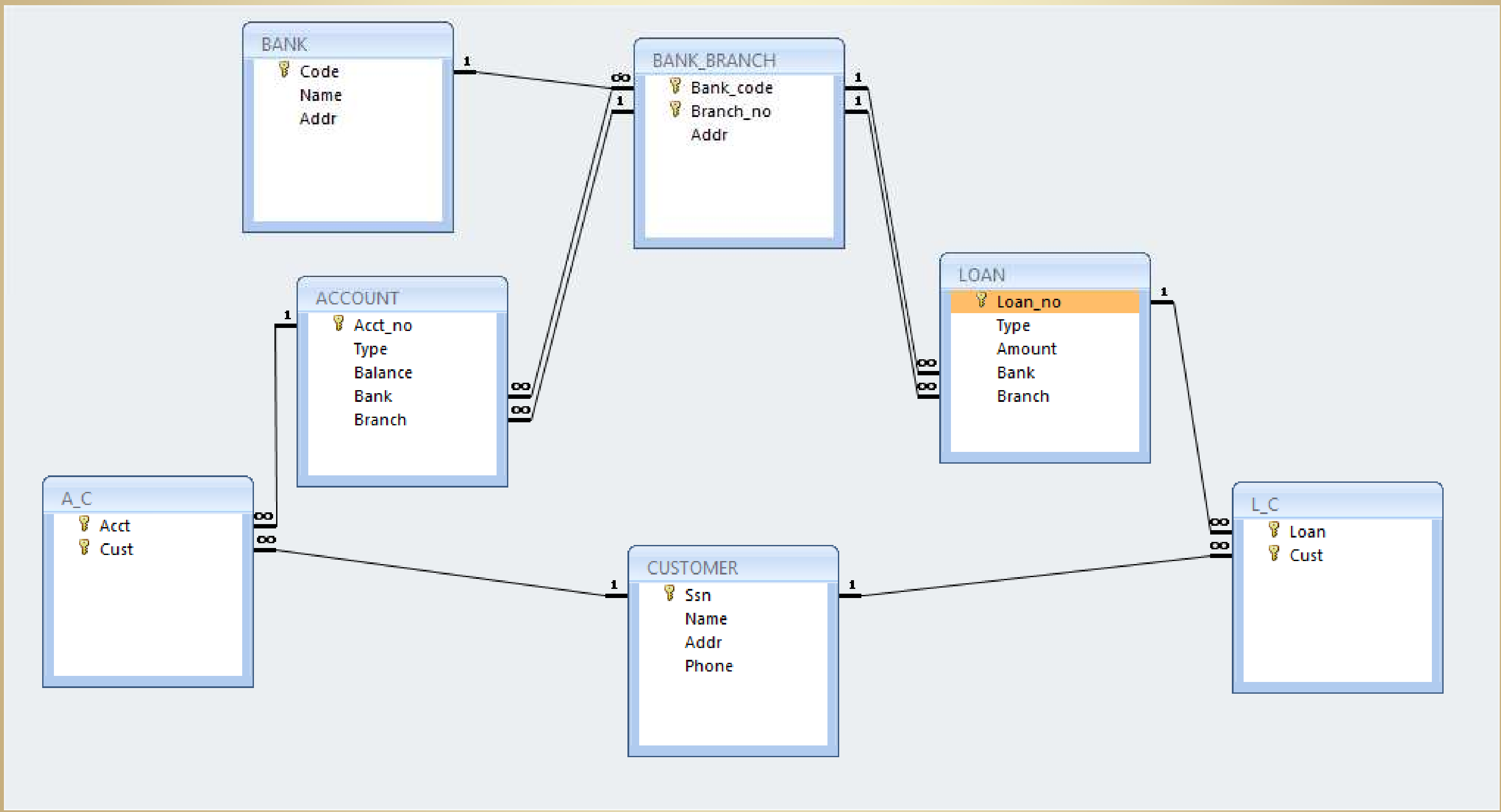
SUPPLY(SName, PName, Part, Quantity)

Completed Bank Schema

```
CUSTOMER(Ssn, Name, Addr, Phone)
BANK(Code, Name, Addr)
ACCOUNT(Acct_no, Type, Balance, Bank, Branch)
LOAN(Loan_no, Type, Amount, Bank, Branch)
BANK_BRANCH(Bank_code, Branch_No, Addr)
A_C(Acct, Cust)
L_C(Loan, Cust)

BANK_BRANCH(Bank_code) refers to BANK
LOAN(Bank,Branch) refers to BANK_BRANCH
ACCOUNT(Bank,Branch) refers to BANK_BRANCH
A_C(Acct) refers to ACCOUNT
A_C(Cust) refers to CUSTOMER
L_C(Loan) refers to LOAN
L_C(Cust) refers to CUSTOMER
```

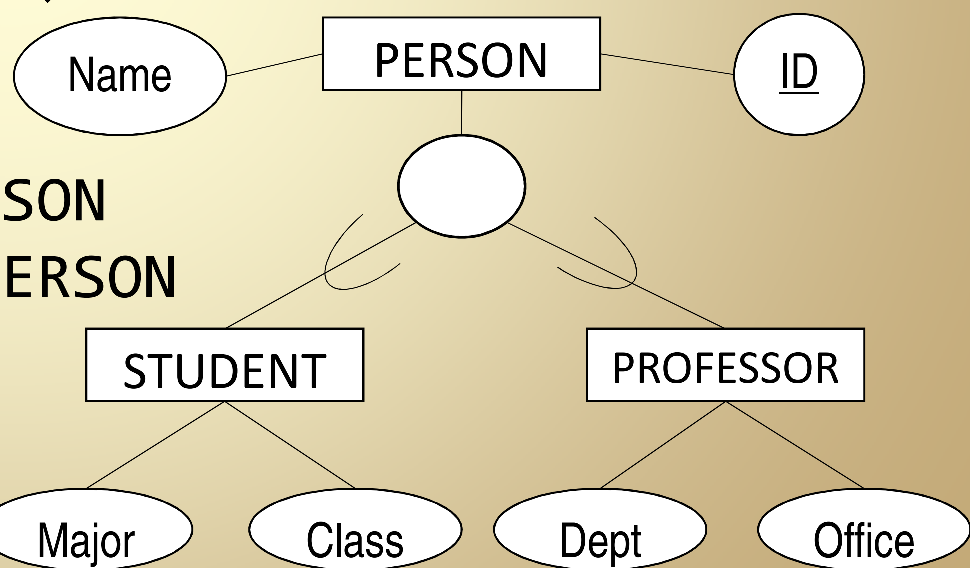
Bank Schema: MS Access



Step 8: Inheritance

- Option a: Each entity type becomes a relation
  - all have same PK (from superclass)
  - PKs in subclasses are FKs to superclass
  - *most general solution*

PERSON(ID, Name)
STUDENT(ID, Major, Class)
PROFESSOR(ID, Dept, Office)

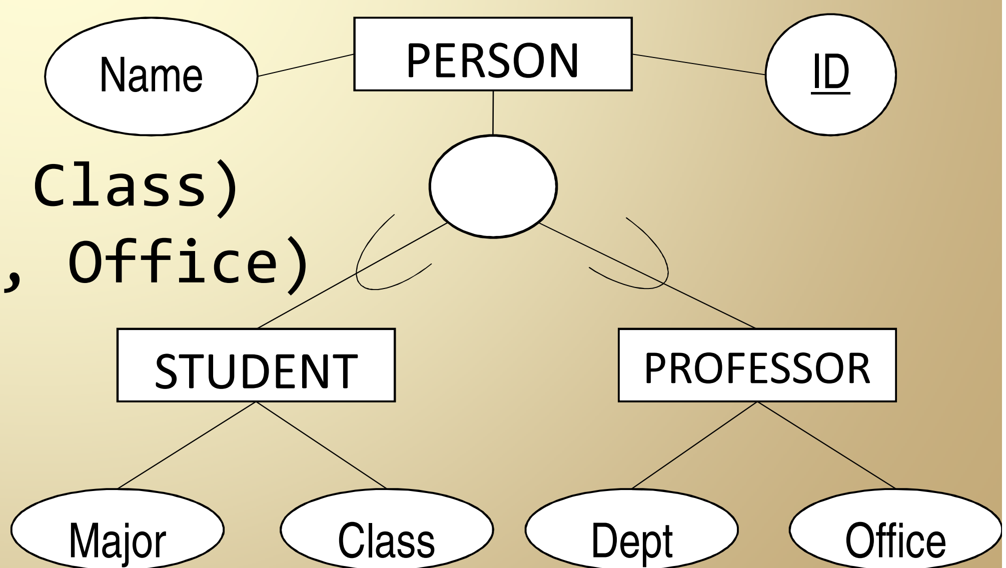


STUDENT(ID) refers to PERSON
PROFESSOR(ID) refers to PERSON

Step 8: Inheritance

- Option b: Each subclass becomes a relation
  - all have same PK (from superclass)
  - each relation gets all superclass attributes
  - *restriction: only works for covering inheritance*
  - *problem: need to join tables to find all PERSONs*

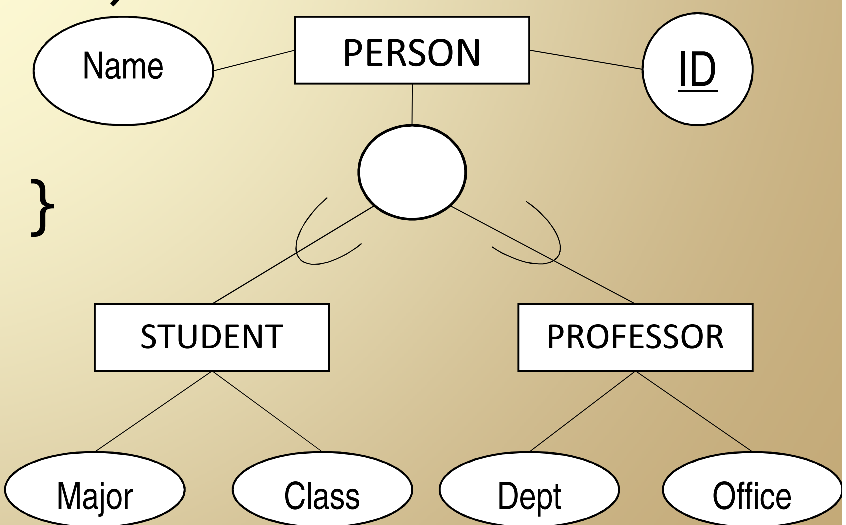
STUDENT(ID, Name, Major, Class)
PROFESSOR(ID, Name, Dept, Office)



Step 8: Inheritance

- Option c: Single relation with a type discriminator
  - PK from superclass
  - all attributes from all classes
  - *restriction: only works for disjoint inheritance*
  - *problem: lots of NULL values*

```
PERSON(ID, Classifier, Name,
     Major, Class, Dept, Office)
```



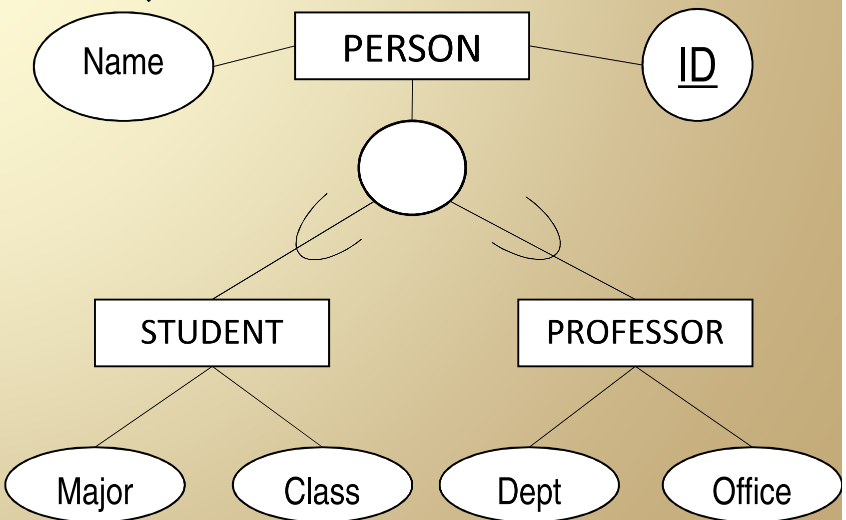

```
Classifier ϵ { ‘S’, ‘P’, ‘N’ }
```

Step 8: Inheritance

- Option d: Single relation with multiple discriminators
  - PK from superclass
  - all attributes from all classes
  - *works for overlapping inheritance*
  - *problem: lots of NULL values*

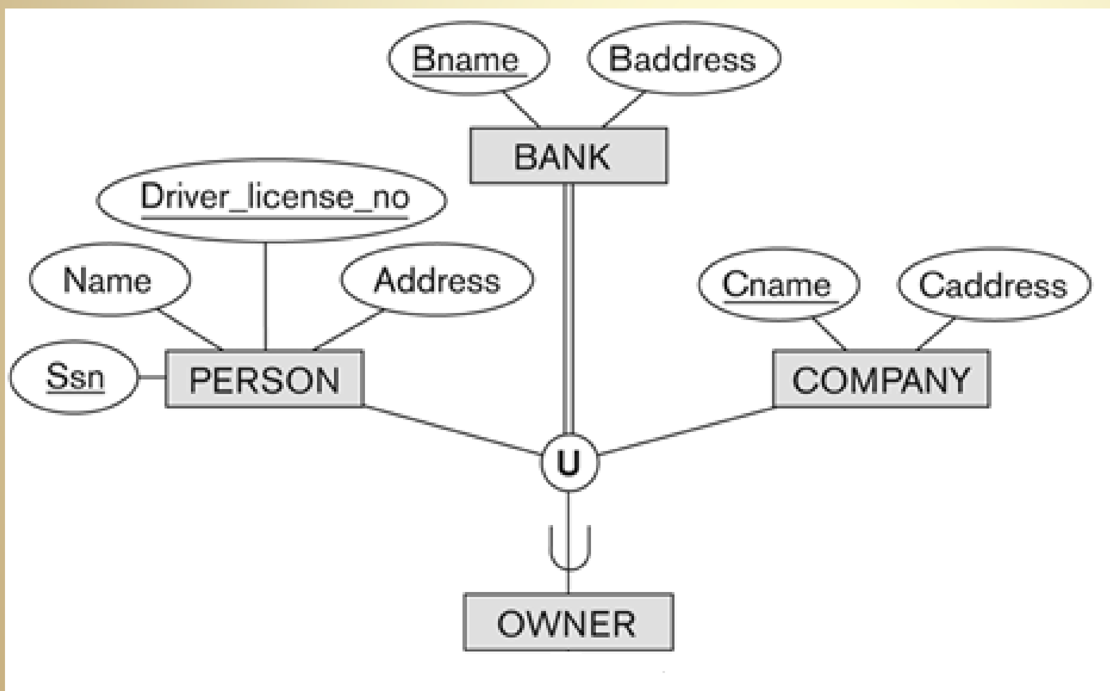
```
PERSON(ID, isStudent, isProfessor,
Name, Major, Class, Dept, Office)
```

```
dom(isStudent) = Boolean
dom(isProfessor) = Boolean
```



Step 9: Unions

- Union types become a new relation of surrogate keys
  - surrogate keys are added to all defining classes
  - attributes of the union type go in the new relation



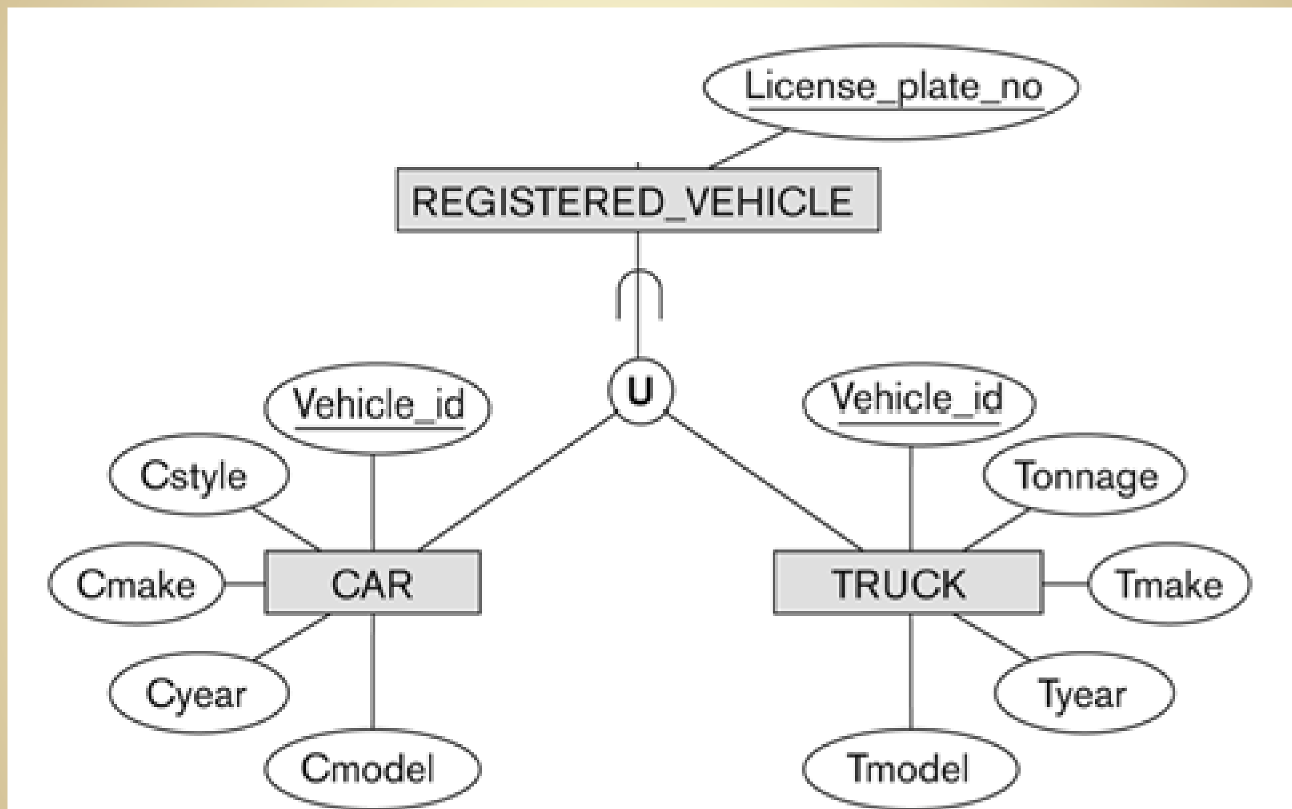
add surrogate key to OWNER

Step 9: Unions

PERSON(Driver_license_no, Ssn, Name,
          Address, Owner_id)
BANK(Bname, Baddress , Owner_id)
COMPANY(Cname, Caddress , Owner_id)
OWNER(ID)

PERSON(Owner_id) refers to OWNER
BANK(Owner_id) refers to OWNER
COMPANY(Owner_id) refers to OWNER

Step 9: Unions



add surrogate key to REGISTERED_VEHICLE

Step 9: Unions

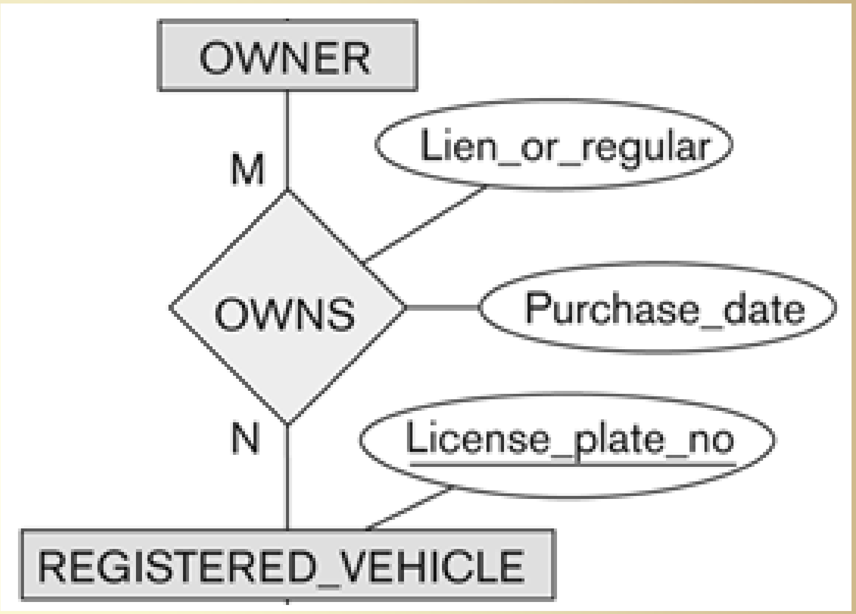
```
CAR(Vehicle_id, Cstyle, Cmake,
          Cyear, Cmodel)
TRUCK(Vehicle_id, Tonnage, Tmake,
          Tyear, Tmodel)
REGISTERED_VEHICLE(Vehicle_id, License_plate_no)
```

```
CAR(Vehicle_id) refers to REGISTERED_VEHICLE
TRUCK(Vehicle_id) refers to REGISTERED_VEHICLE
```

in this case, we don’t need to invent a surrogate key, since the domains of CAR keys and TRUCK keys are the same (and non-overlapping)

Step 9: Unions

OWNS relation uses the surrogate keys

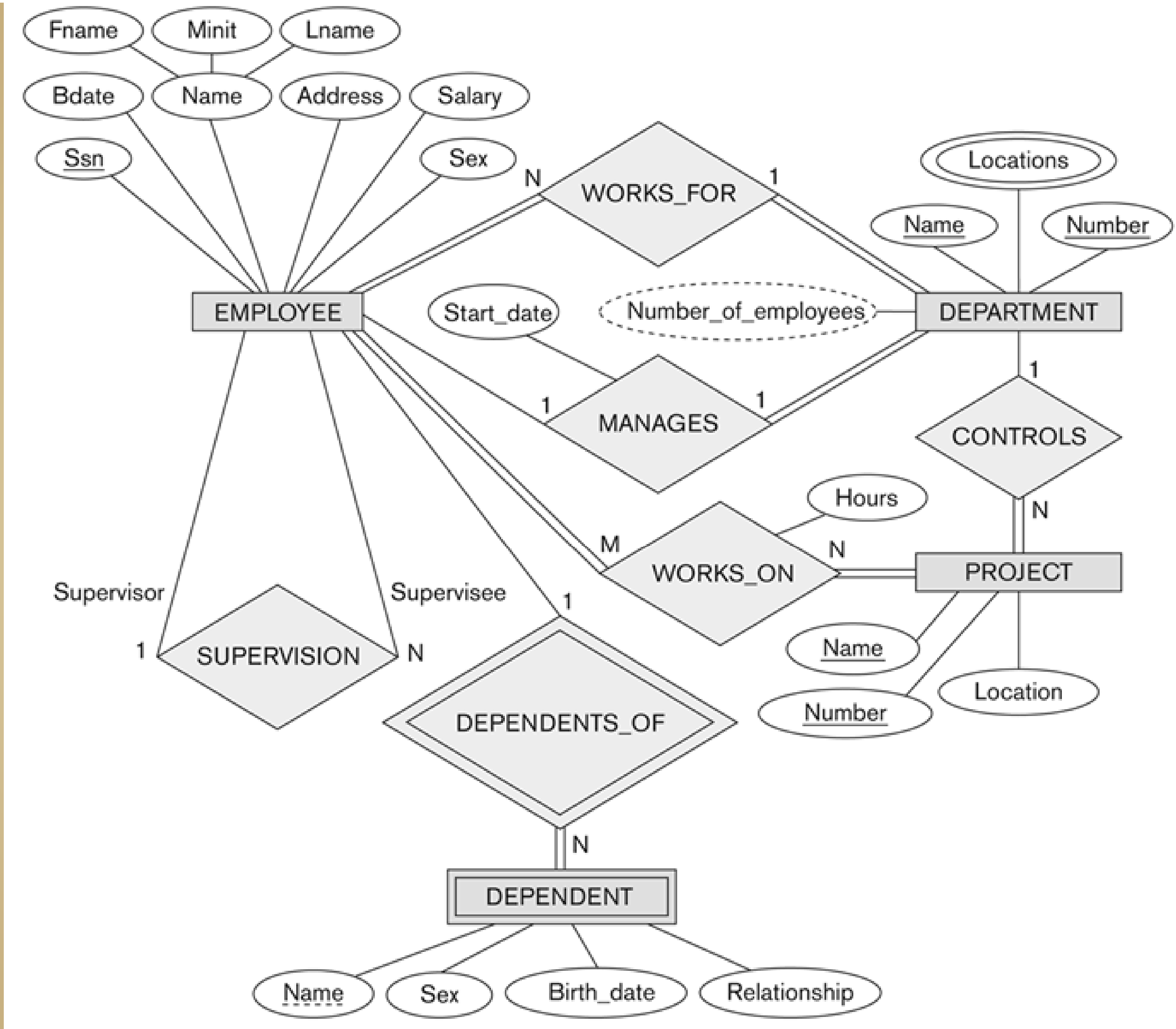


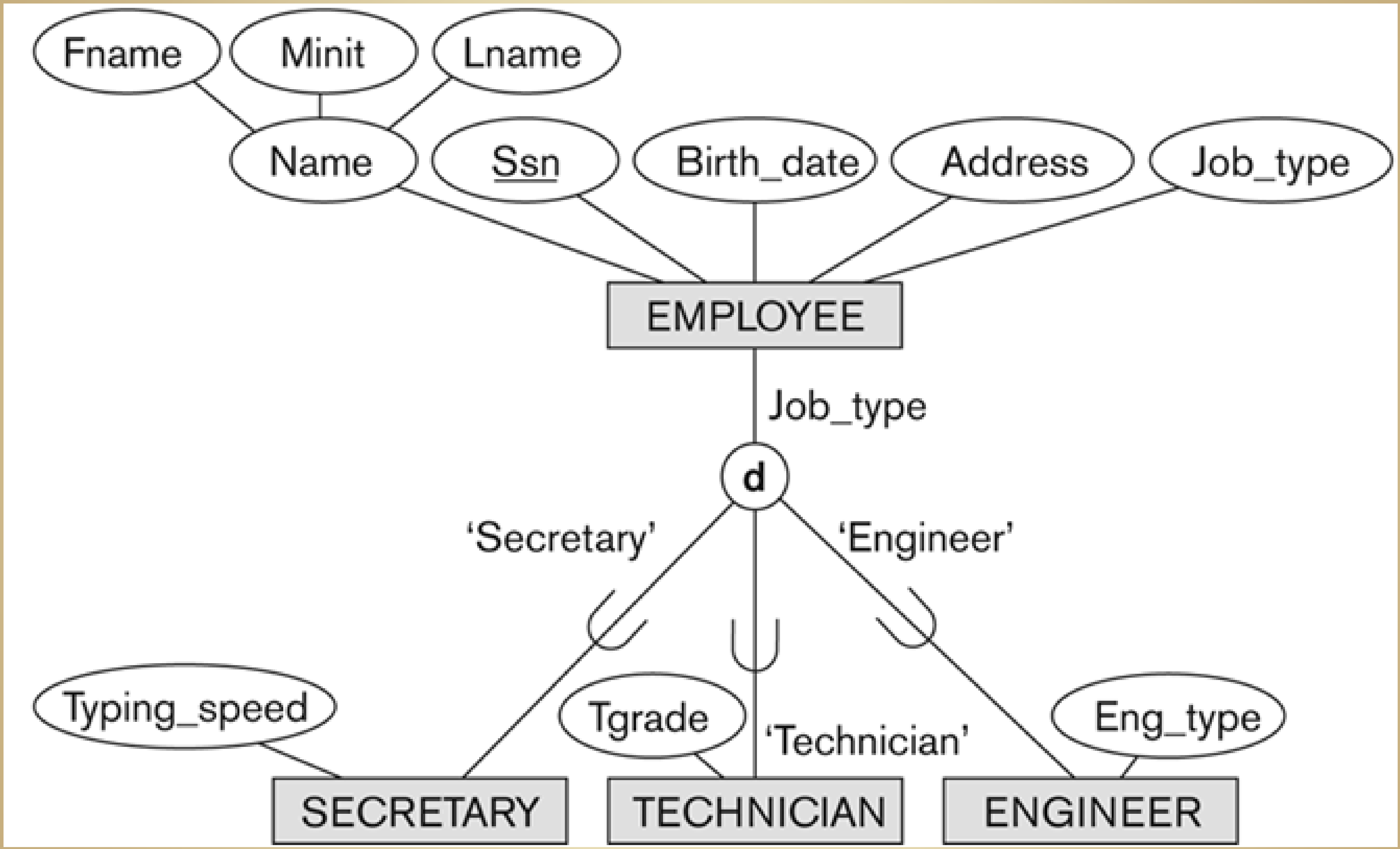

```
OWNS(Owner_id, Vehicle_id,
     Purchase_date, Lien_or_regular)

OWNS(Owner_id) refers to OWNER
OWNS(Vehicle_id) refers to REGISTERED_VEHICLE
```

EXERCISES

- Create relational schema from the following:


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Number
Location
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship


Fname
Minit
Lname
Name
Ssn
Birth_date
Address
Job_type
EMPLOYEE
Job_type
d
'Secretary'
'Engineer'
Typing_speed
Tgrade
'Technician'
Eng_type
SECRETARY
TECHNICIAN
ENGINEER