Database Design Tutorial

SWEN 304

Trimester 2, 2017

Engineering and Computer Science

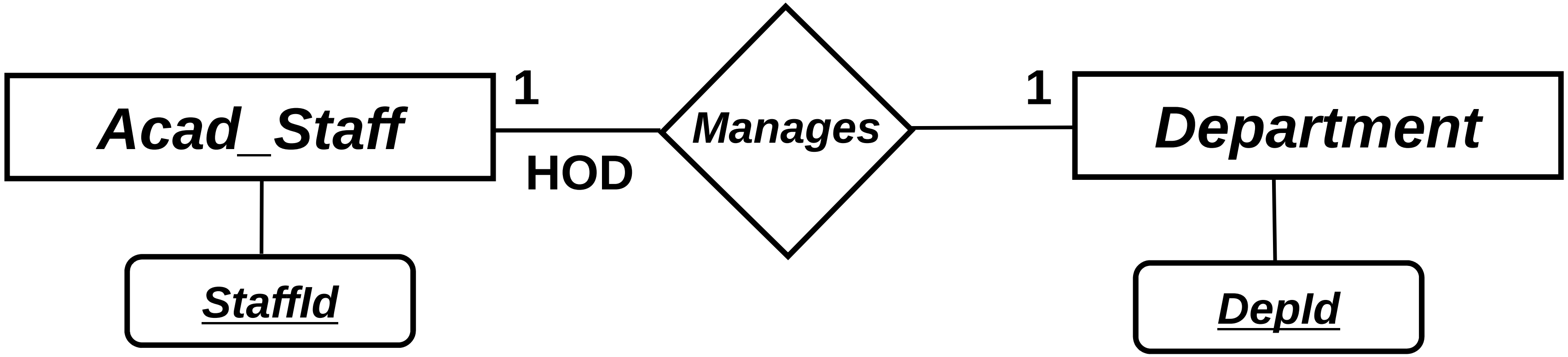
TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Outline

- Mapping
  - Relationship type with different cardinality ratios
  - Recursive relationships
  - More than one relationship
  - Ternary relationships
  - Superclass/subclass
  - Multivalued attributes

  of ER/EER model to the relational model

Cardinality 1:1, Both PCs are Partial (1)



- The best mapping solution:

  $AccadStaff\ (\ (\{StaffId, \ldots\},\ \{StaffId\}),$

  $Department\ (\{DeptId, \ldots\},\ \{DeptId\}),$

  Two Keys

  $Manages\ (\{StaffId,\ DeptId\},\ \{StaffId,\ DeptId\})$

  $Manages\ [StaffId] \subseteq Staff\ [StaffId]$

  $Manages\ [DeptId] \subseteq Department\ [DeptId]$

Cardinality 1:1, Both PCs are Partial (2)

Acad_Staff

StaffId	*StaffName*
007007	*James*
131313	*Susan*
010101	*Nigel*

Manages

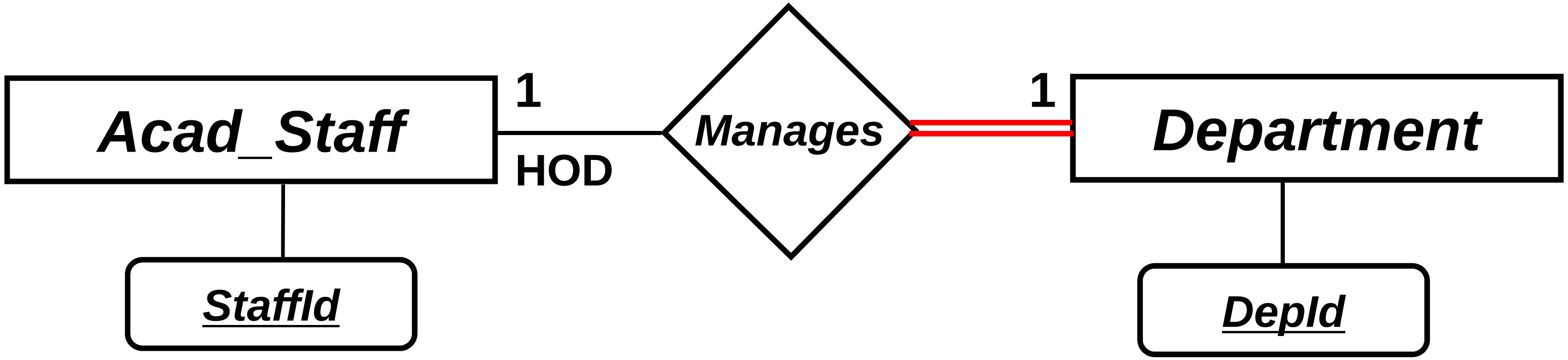
StaffId	*DeptId*
007007	*Comp*
131313	*Math*

Department

DeptD	*DeptName*
Comp	*Comp Sci*
Math	*Mathematics*
Stat	*Statistics*

Both StaffId and DeptId are keys. This way we secure the satisfaction of the cardinality ratio 1:1
A separate relationship table always implies partial participation constraints

Cardinality 1:1, Right PC Total, Left Not (1)



- Insert primary key *StaffId* of *Acad_Staff* into *Department* relation schema as the foreign key with

  Null (*Department*, *StaffId*) = *Not*

- So, solution is

  Acad_Staff ({*StaffId*,...}, {*StaffId* }),

  Department ({*StaffId*, ... *DeptId* }, {*StaffId*, *DeptId* })

  // two keys again, also all structural constraints reserved

  Department [*StaffId*] $\subseteq$ *Acad_Staff* [*StaffId*]

Cardinality 1:1, Right PC Total, Left Not (2)

Acad_Staff

<u>StaffId</u>	*StaffName*
007007	*James*
131313	*Susan*
010101	*Nigel*
555555	*Susan*
313131	*Paul*

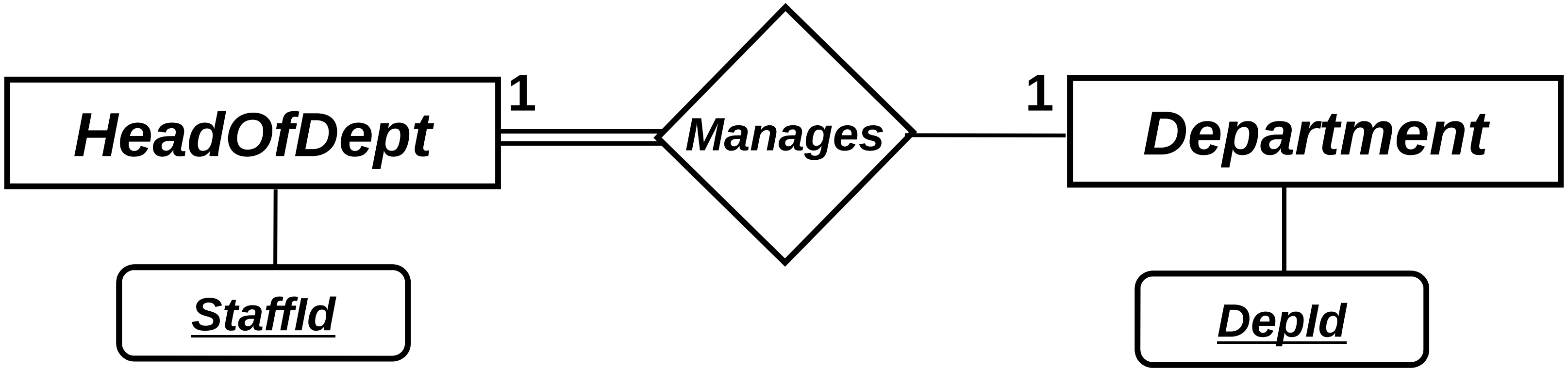
Department

<u>DeptD</u>	*DeptName*	*<u>StaffId</u>*
Comp	*Comp Sci*	*007007*
Math	*Mathematics*	*131313*
Stat	*Statistics*	*555555*

Can't have duplicates

Can't have duplicates

Cardinality 1:1, Left PC Total, Right Not



- Insert primary key *DepId* of *Department* as the foreign key into *HeadOfDept* relation schema, and put

  Null (*HeadOfDept*, *DeptId*) = *Not*

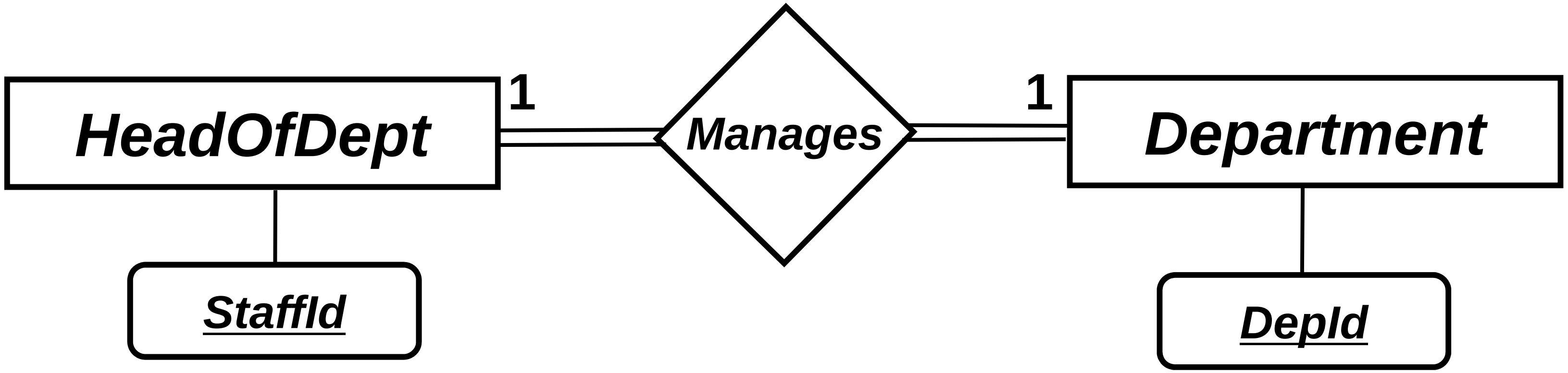
- So, solution is

  Department (<u>*DeptId*</u>,...)

  HeadOfDept(<u>*StaffId*</u>,..., <u>*DeptId*</u>),

  HeadOfDpt [*DeptId*] $\subseteq$ *Department* [*DeptId*]

Cardinality 1:1, Both PCs Total (1)



- Map the **two entity types into one relation schema** with:
  - appropriate name,
  - the set of attributes containing all simple, single valued attributes of both entity types and the relationship type,
  - the set of keys containing all the keys of both entity types
- So, solution would be
  - *Department_HeadOfDept* ({*DeptId*,..., *StaffId* }, {*DeptId*, *StaffId* })
    - All structural constraints preserved

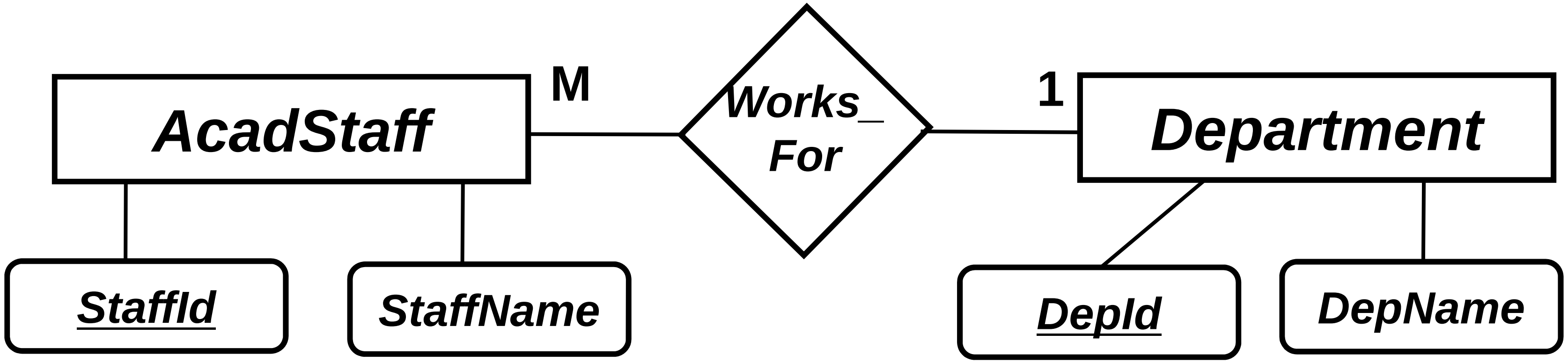
Cardinality 1:1, Both PCs Total (2)

Department

StaffId	*StaffName*	*<u>DepId</u>*	*DeptName*
007007	*James*	*Comp*	*Comp Sci*
131313	*Susan*	*Math*	*Mathematics*
555555	*Susan*	*Stat*	*Statistics*

Note: There are two candidate keys: *StaffId* and *DepId*

Cardinality Ration 1:M



AcadStaff

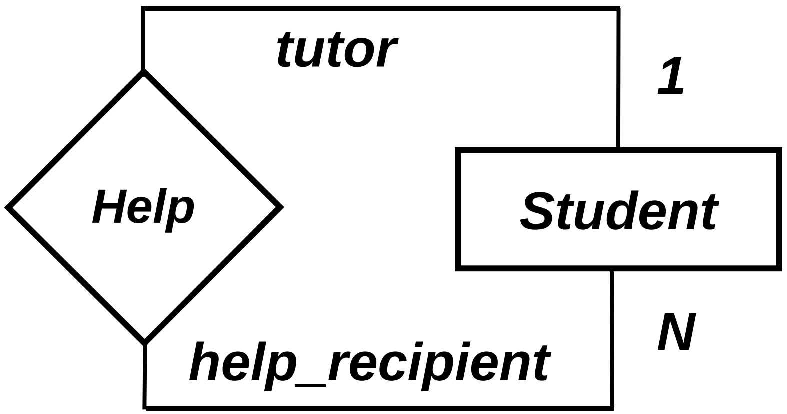
StaffId	StaffName	DepId
007007	James	Comp
131313	Susan	Comp
010101	Nigel	ω

Department

DeptD	DeptName
Comp	Comp Sci
Math	Mathematics
Stat	Statistics

AcadStaff [*DeptId*] $\subseteq$ *Department* [*DeptId*]

Mapping Recursive Relationship Type



Student

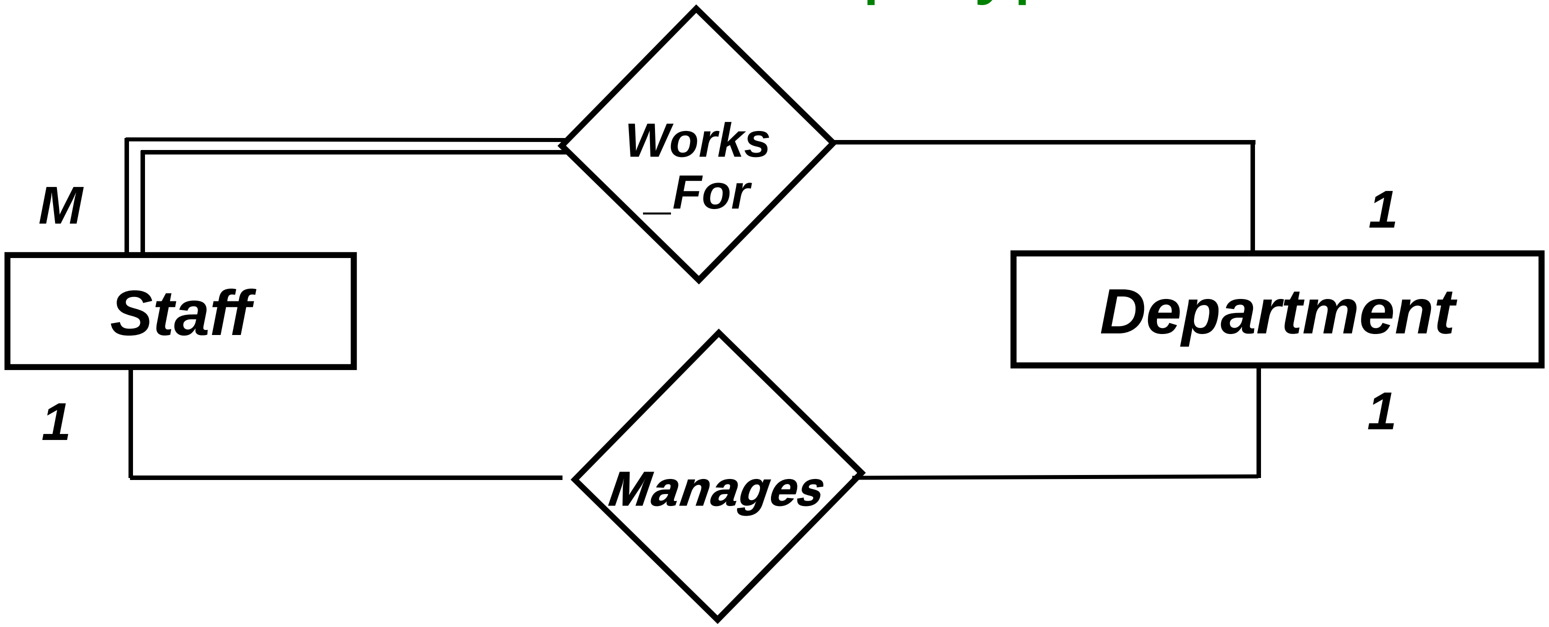
<u>StudentId</u>	Stud_Name	Tutor
007007	James Bond	ω
131313	Susan Smith	007007
505050	John Cecil	007007
123456	Amit Gandhi	654321
654321	Craig Anslow	ω

$$Dom(Tutor) \subseteq Dom(StudentId)$$

Referential Integrity:

$$Student\ [Tutor] \subseteq Student\ [StudentId]$$

More Than One Relationship Type



Staff

<u>StafId</u>	St_Name	DeptId
007	James	Comp
131	Mark	Math
505	Pavle	Comp

Manages

<u>HoD</u>	<u>DeptId</u>
007	Comp
131	Math

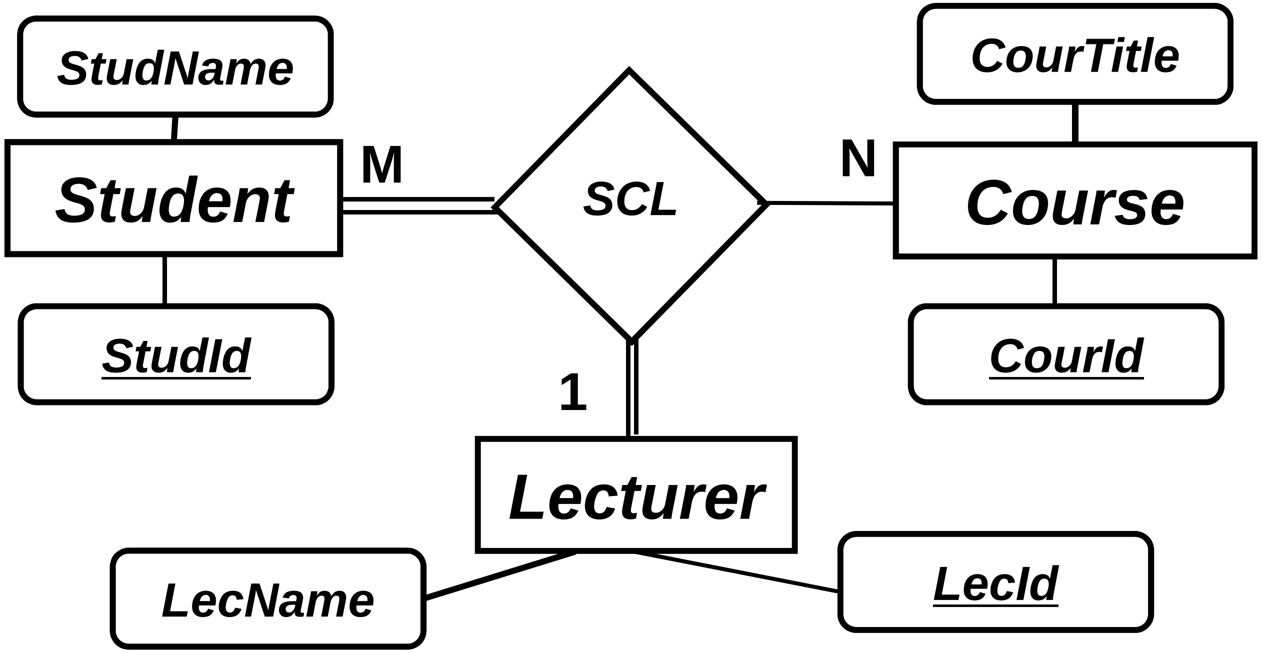
Department

<u>DeptId</u>	Dep_Name
Comp	Comp Science
Math	Mathematics
Stat	Statistics

Null (Staff, DeptId) = Not,   Dom(HoD) $\subseteq$ Dom(StafId)

Mapping (n-ary R, n > 2) - an example (1)

- Ternary relationship type: *SCL*



Student ({*StudId*, *StudName*}, {*StudId* }),
Lecturer ({*LecId*, *LecName*}, {*LecId* }),
Course ({*CourId*, *CourTitle*}, {*CourId* }),
SCL ({*StudId*, *LecId*, *CourId* }, {*StudId* + *CourId* }), *Null*(*SCL*, *LecId*) = *N*

Mapping (n-ary R, n > 2) - an example (2)

$S = \{$
$Student\ (\{StudId,\ StudName\},\ \{StudId\ \}),$
$Lecturer\ (\{LecId,\ LecName\},\ \{LecId\ \}),$
$Course\ (\{CourId,\ CourTitle\},\ \{CourId\ \}),$
$SCL\ (\{StudId,\ LecId,\ CourId\ \},\ \{StudId + CourId\ \}),$
$Null(SCL,\ LecId) = N$
$\}$

$IC = \{$
$SCL[StudId\ ] \subseteq Student\ [StudId\ ],$
$SCL[CourId\ ] \subseteq Course\ [CourId\ ],$
$SCL[LecId\ ] \subseteq Lecturer\ [LecId\ ],$
$Student\ [StudId\ ] \subseteq SCL\ [StudId\ ],$ // total participation constraint
$Lecturer\ [LecId\ ] \subseteq SCL\ [LecId\ ]$ // total participation constraint
$\}$

Mapping (n-ary R, n > 2) - an example (3)

Student

<u>*StudId*</u>	*StName*
s_1	Susan
s_2	James
s_3	Anny

Course

<u>*CourId*</u>	*CoName*
c_1	Java
c_2	C++
c_3	DB Sys

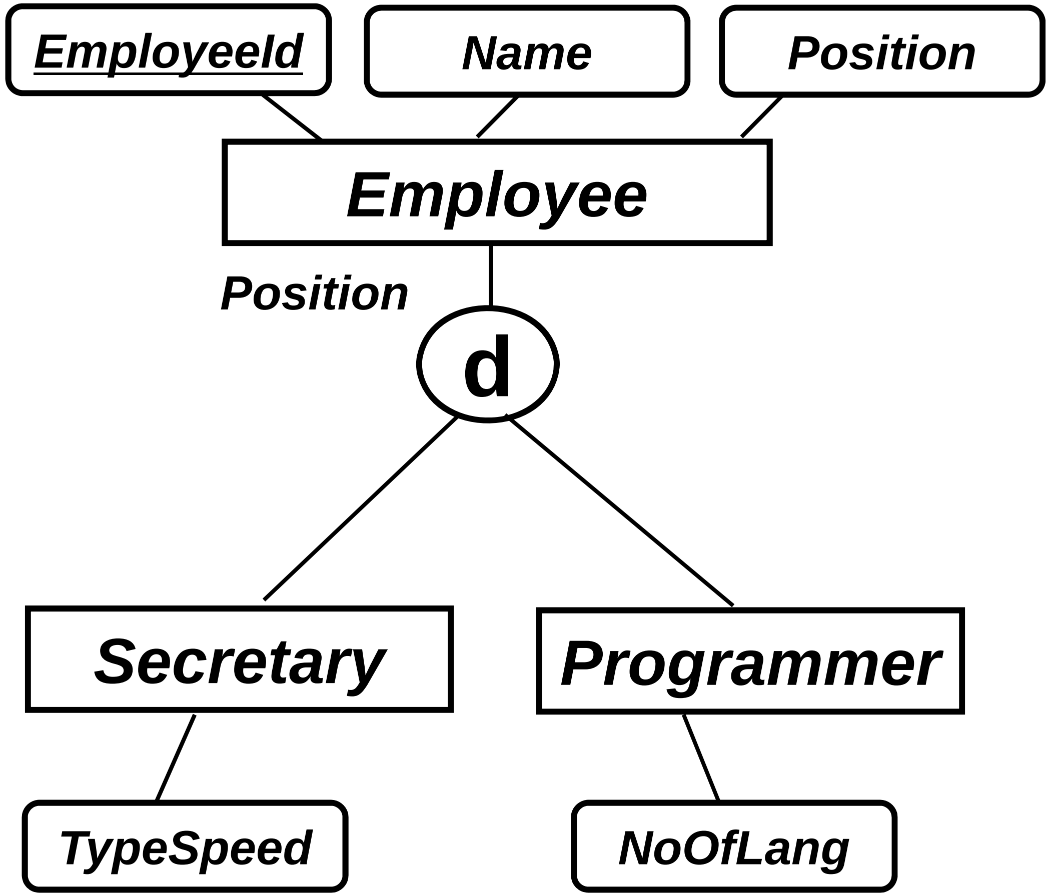
Lecturer

<u>*LecId*</u>	*LeName*
l_1	Pondy
l_2	Pavle

SCL

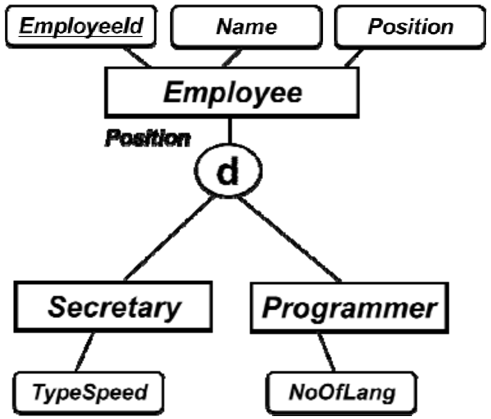
<u>*StudId*</u>	<u>*CourId*</u>	*LecId*
s_1	c_1	l_1
s_2	c_1	l_1
s_1	c_3	l_2
s_3	c_3	l_2

Map Superclass/Subclass Relationships



IS-A Hierarchy – Mapping Option 1 (1)

- Separate mapping of each construct – subclass is considered as a weak entity type



{ *Employee* (*EmpId*, *Name*, *Position*)
Secretary (*EmpId*, *TypeSpeed*)
Programmer (*EmpId*, *NoOfLang*) }

- If the classification is total

$(r(Secretary)[EmpId] \cup r(Programmer)[EmpId] = r(Emplyee)[EmpId]) \in IC$ (interrelation constraint)

- If the classification is disjoint

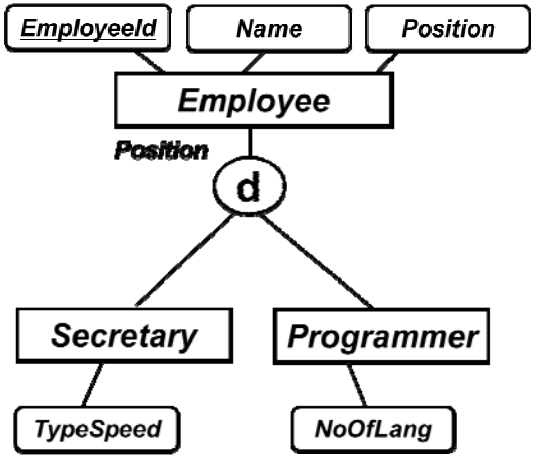
$(r(Secretary)[EmpId] \cap r(Programmer)[EmpId] = \emptyset) \in IC$

- Referential integrity constraints also needed

IS-A Hierarchy – Mapping Option 1 (2)

Employee

<u>EmpId</u>	Name	Position
007007	James	Programmer
131313	Susan	Secretary
010101	Nigel	Programmer
555555	Susan	Secretary
919191	Paul	Accountant



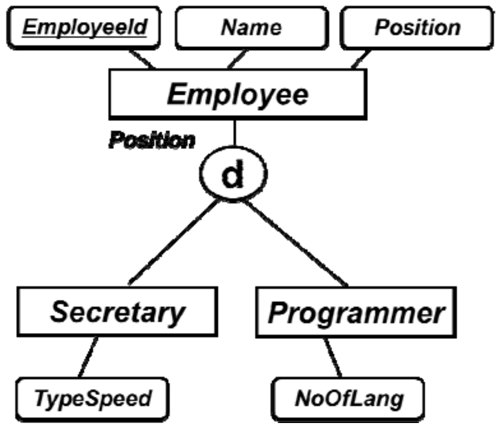
Secretary

<u>EmpId</u>	TypeSpeed
131313	A
555555	B

Programmer

<u>EmpId</u>	NoOfLang
007007	5
010101	3

IS-A Hierarchy–Mapping Options 2&3



- **Option 2**: Each subclass relation inherits attributes from superclass (attribute *Position* is not needed, since table names convey information about position)

  {*Secretary* (*<u>EmpId</u>, TypeSpeed, Name*)
  Programmer (*<u>EmpId</u>, NoOfLang*, *Name*)}

  - If the classification is disjoint

    $r(Secretary)[EmpId] \cap r(Programmer)[EmpId] = \emptyset$

  - Appropriate if the classification is total and subclasses disjoint

- **Option 3**: All classes together are represented as one relation schema

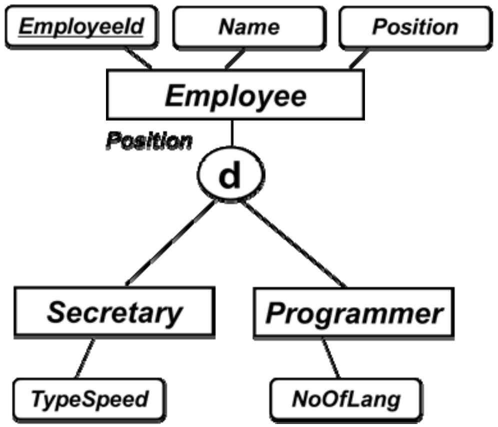
  {*Employee* (*<u>EmpId</u>, TypeSpeed, Name, NofLang* , position)}

  - Appropriate if the number of specific attributes is small

IS-A Hierarchy–Mapping Option 2

Secretary

<u>EmpId</u>	Name	TypeSpeed
131313	Susan	A
555555	Susan	B

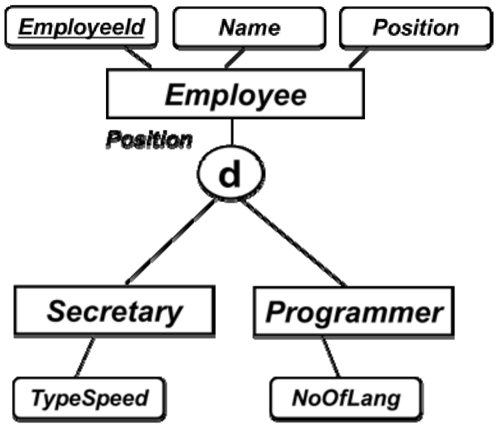


Programmer

<u>EmpId</u>	Name	NoOfLang
007007	James	5
010101	Nigel	3

What about Paul?

IS-A Hierarchy–Mapping Option 3



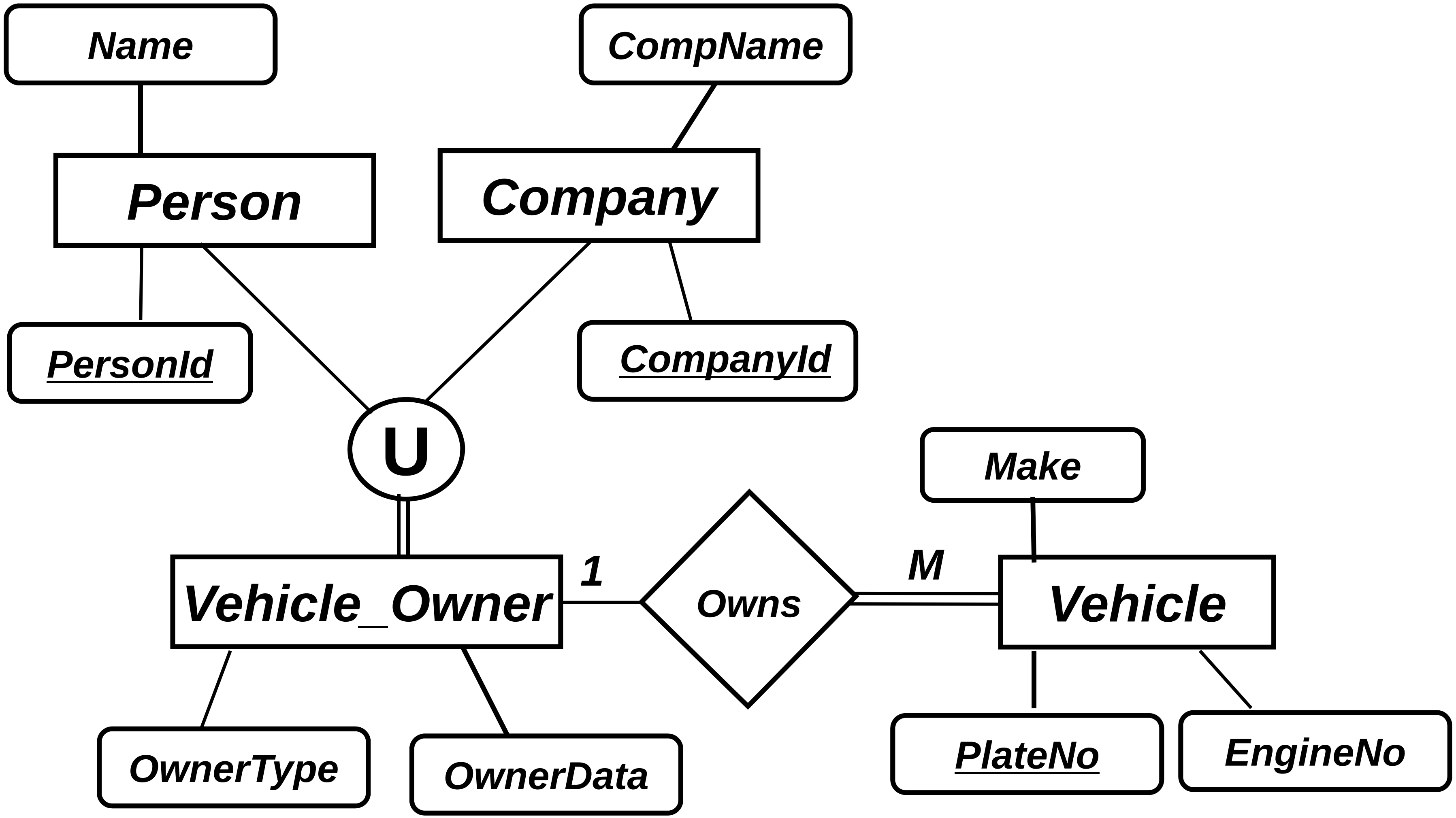
Employee

EmpId	Name	NoOfLang	TypeSpeed	Position
131313	Susan	ω	A	Secretary
555555	Susan	ω	B	Secretary
007007	James	5	ω	Programmer
010101	Nigel	3	ω	Programmer
919191	Paul	ω	ω	Accountant

Mapping Categories

- A category is the subclass of the union of two or more superclasses
- A category is mapped to one relation schema with:
  - The same name as the category type,
  - All the single valued attributes of the category (including the attribute *CategoryType*), and
  - An artificial attribute, so called surrogate key
- The relationship between category relation schema and superclass relation schemas is accomplished by inserting the surrogate key in each superclass relation schema

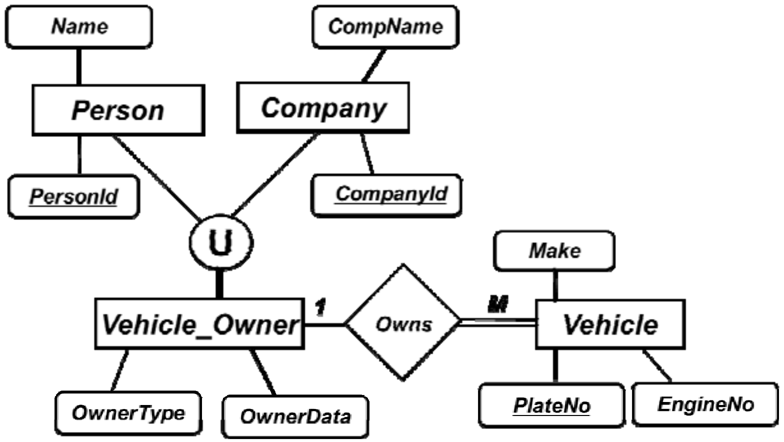
Mapping Categories – Example (1)



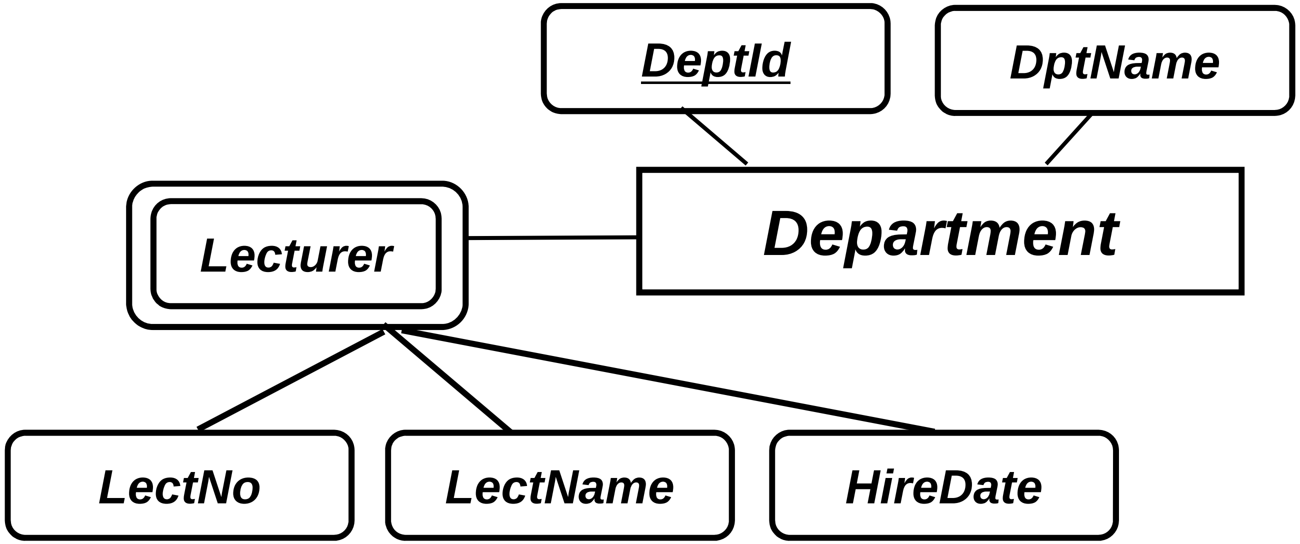
Mapping Categories – Example (2)

{*Vehicle_Owner* (*<u>OwnerId</u>*, *OwnerData*, *OwnerType*),
Person (*<u>PersonId</u>*, *Name*, *OwnerId*),
Company (*<u>CompanyId</u>*, *CompName*, *OwnerId*),
Veicle (*<u>PlateNo</u>*, *EngineNo*, *Make*, *OwnerId*)}

- *OwnerId* is a surrogate key



Mapping a Multivalued Attribute (1)



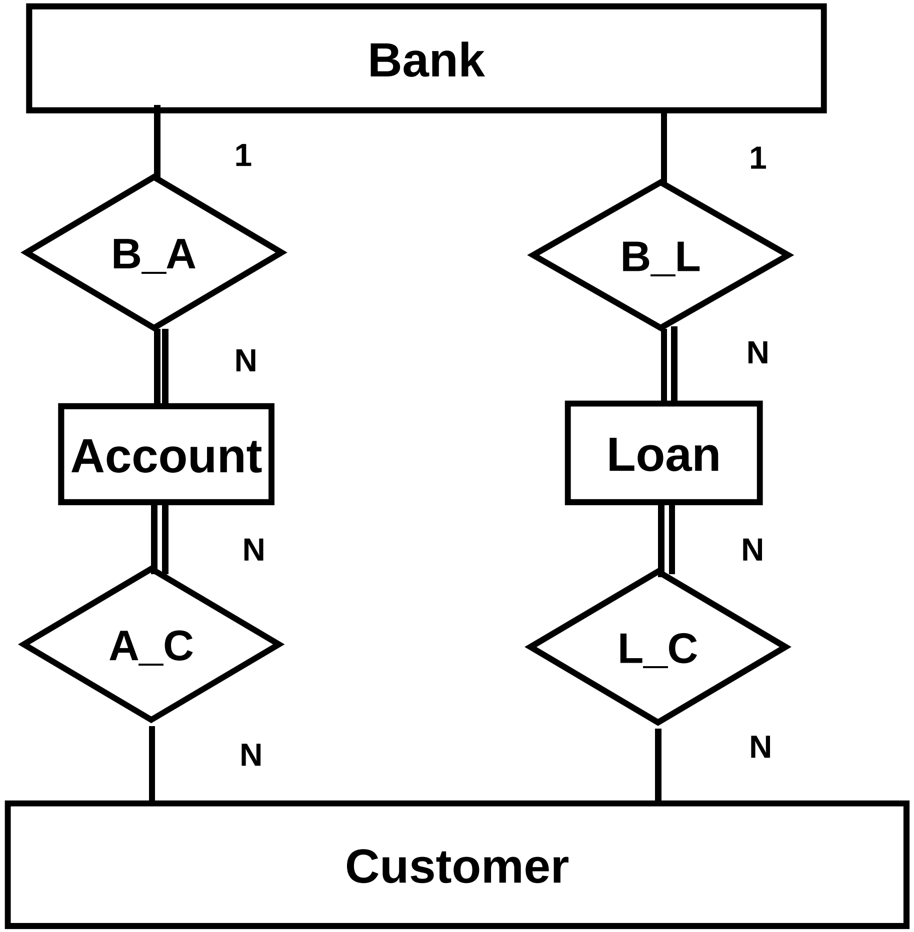
Mapping Multivalued Attributes (2)

Lecturer

DeptId	*LectNo*	*LectName*	*HireDate*
Comp	1	Ray	14.02.00.
Comp	2	Pavle	14.07.00.
Comp	3	Ewan	01.01.91.
Math	1	Colin	14.02.00.
Stat	1	John	01.07.01.

- Instead of specifying *Lecturer* as a multivalued attribute, it were possible to define it as a weak entity type. The effect would be the same

Example: Assignment 4 Question 4 (2005)



BankInfo

Bank(<u>BankId</u>, BankName)

Account(<u>AccountNo</u>, Balance)

Loan(<u>LoanNo</u>, Amount)

Customer(<u>CustomerId</u>, Name)

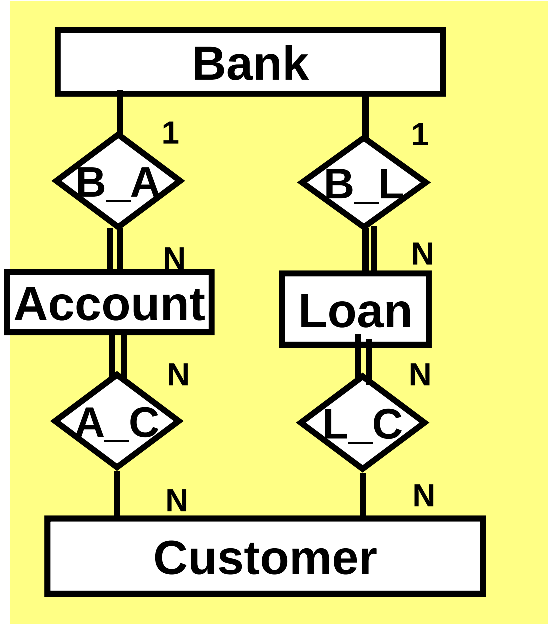
B_A(Bank, Account)

B_L(Bank, Loan)

A_C(Account, Customer)

L_C(Loan, Customer)

Example: Set of Relation Schemes



- Map conceptual schema *BankInfo* to

  a set of relation schemas *S*

  S = {

  Bank({BankId, BankName}, {BankId})

  Account({AccountNo, Balance, BankId}, {AccountNo})

  Loan({LoanNo, Amount, BankId}, {LoanNo})

  A_C({AccountNo, CustomerId}, {AccountNo+CustomerId})

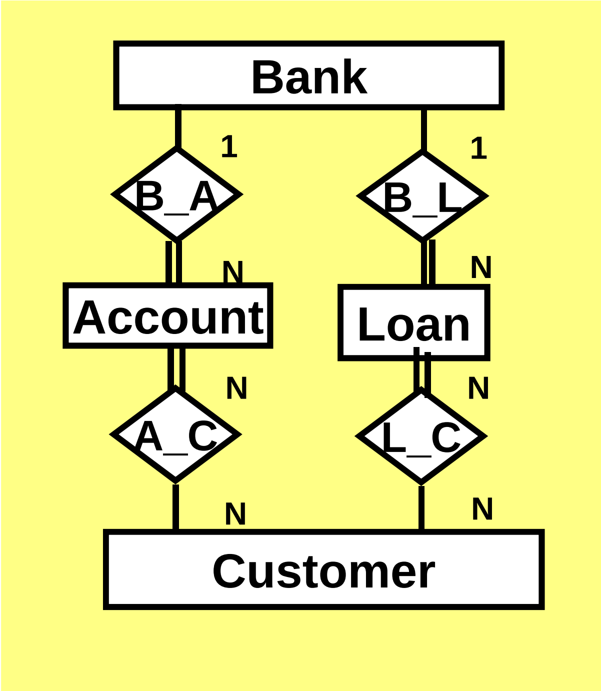
  L_C({LoanNo, CustomerId}, {LoanNo + CustomerId})

  Customer({CustomerId, Name}, {CustomerId})

  }

Example: Set of Referential Integrity Constraints

- State the referential integrity constraints that have a NOT NULL foreign key:
  - Account[BankId] $\subseteq$ Bank[BankId]
  - Loan[BankId] $\subseteq$ Bank[BankId]
  - Null(Account, BankId) =Not
  - Null(Loan, BankId) = Not



- Other referential integrity constraints
  - L_C[LoanNo] $\subseteq$ Loan[LoanNo]
  - L_C[CustomerId] $\subseteq$ Customer[CustomerId]
  - A_C[AccountNo] $\subseteq$Account[AccountNo]
  - A_C[CustomerId] $\subseteq$ Customer[CustomerId]