Unit 4:
Relational Database Design

4.1. Database Design Fundamentals

4.2. Conceptual Design

4.3. Logical Design

Bases de Datos y Sistemas de información
DSIC Departamento de Sistemas Informáticos y Computación / Universidad Politécnica de Valencia

V. 16.11

Unit 4.3. Logical Design

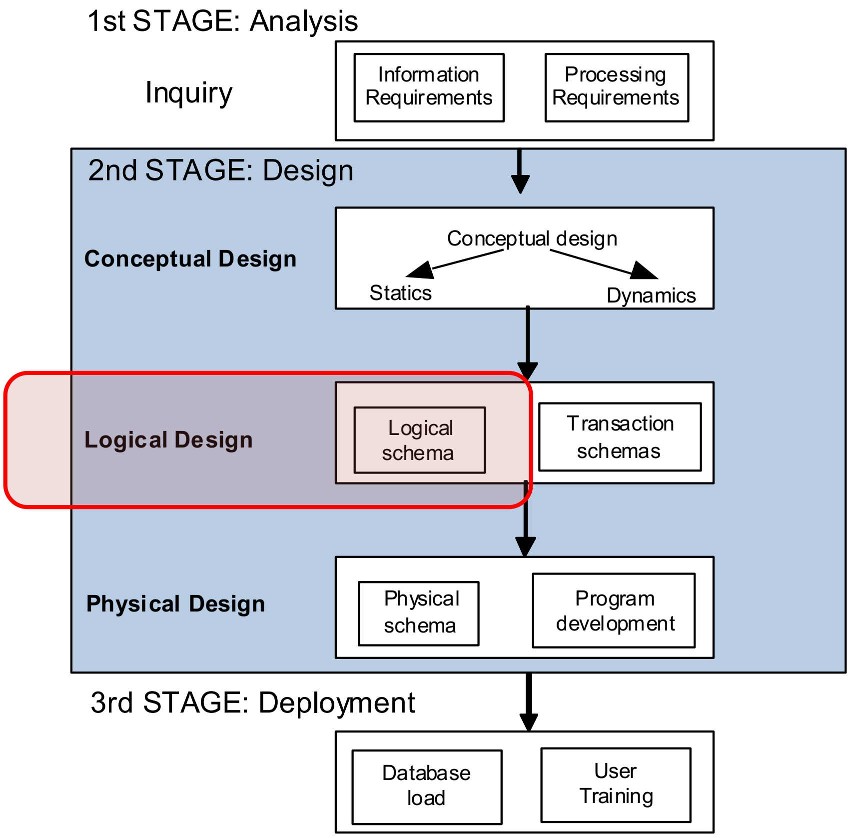
1. **Introduction**
2. Class Transformation
   - 2.1. Strong classes
   - 2.2. Weak classes
   - 2.3. Specialization
3. Association Transformation
   - 3.1. Non-reflexive associations
   - 3.2. Reflexive associations
   - 3.3. Association with link attributes
   - 3.4. Association within association (association classes)
4. Choosing directives for foreign keys
5. Examples
6. Introduction to Databases Normalization

1. Introduction

We can transform the ER-UML diagram into other formal models.

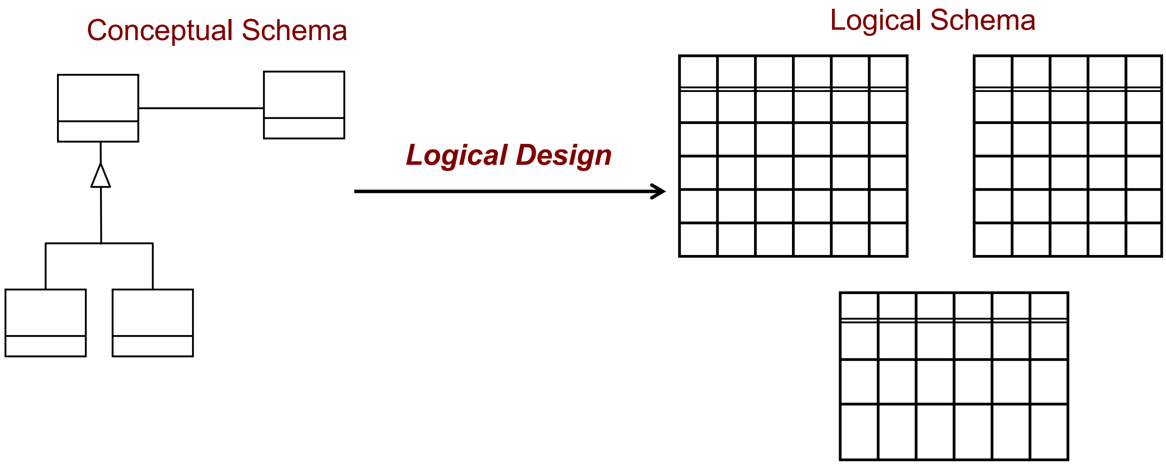
- In *software engineering*, this can lead to the definition of classes and attributes.
- In *databases*, we can transform the diagram into other database models, e.g. the relational data model.
  - This transformation is known as **logical design**.
  - The output will be the relational schema (seen in unit 2)

1. Introduction



Logical Design

Logical Design: Transformation of a conceptual schema, described using a data model (e.g. ER_UML) into another data model (e.g. relational model) which will be the one used by the Database Management System.



- We are going to apply transformations.
- Multiplicities, associations, and constraints are expressed by the use of PK, FK, NNV, and UNI constraints
- Some properties or constraints cannot be represented using these predefined constraints and we will have to add them to the list of general integrity constraints (implemented as assertions, triggers, or program constraints)
- When facing several design options:
  1. *Chose the resulting schema with the fewest general constraints.*
  2. *If the number of general constraints is similar, choose the solution with the fewest relations.*

Methodology to obtain the relational schema

I. Transform the classes into relations
   1. Strong classes
   2. Weak classes
   3. Specialized classes

II. Transform the associations according to their multiplicity
   - 0..1:0..*
   - 0..*:0..*
   - …

III. Those properties that can't be represented in the relational schema, will be expressed in a list of integrity constraints

Unit 4.3. Logical Design

2.1. Strong classes

A

a_0: {id}: t_a_0
a_1: {unique$_1$}:{0 ..1}:t_a_1
a_2: {1..1}:t_a_2
a_3: {0..1}:t_a_3
a_4: {1..*}:t_a_4
a_5: {0..*}:t_a_5
a_6: {0..1}:
    a_{61}:t_ a_{61}
    a_{62}:t_a_{62}

- "id" to PK
- "unique" to UNI
- "1..x" to NNV
- "x..*" to extra table and FK
  (A one-to-many multiplicity)
- "1..*" (also) to an extra IC.

A(a_0:t_a_0,a_1:t_a_1,a_2:t_a_2,a_3:t_a_3, a_{61}:t_a_{61}, a_{62}:t_a_{62})
  PK:{a_0}
  UNI:{a_1}
  NNV:{a_2}
A4(a_0:t_a_0,a_4:t_a_4)
  PK:{a_0,a_4}
  FK:{a_0}→A(a_0)
A5(a_0:t_a_0,a_5:t_a_5)
  PK:{a_0,a_5}
  FK:{a_0}→A(a_0)

A_4:{1..*}

IC1: Every value in the attribute *a_0* of *A* must appear in the attribute *a_0* of *A4*.

Example

Person

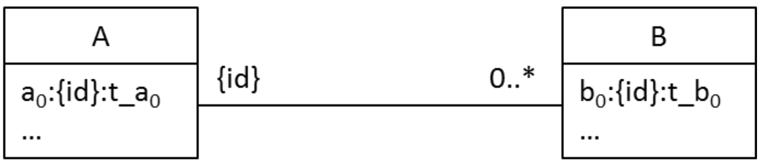
SSN:{id}: char
Passport:{unique}:{1..1}:char
Name:{1..1}:
    First: char
    Second: char
Age: {0..1}:int
Phone:{0..*}:char

Person(SSN: char, Passport: char, First_Name: char,
    Second_Name: char, Age: int,)
  PK:{SSN}
  UNI:{Passport}
  NNV:{Passport, First_name, Second_name}

Contacts(SSN: char, Phone:char)
  PK:{SSN, Phone}
  FK:{SSN}→ Person

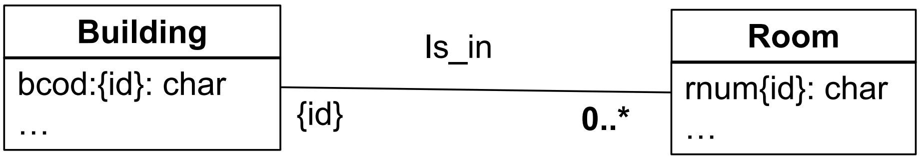
Unit 4.3. Logical Design

2.2. Weak classes

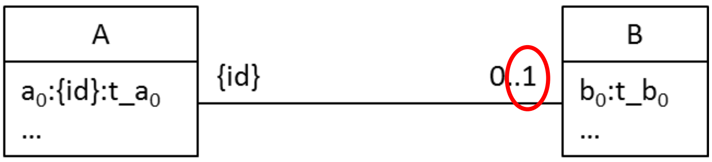


A(a_0:t_a_0,…)
    PK:{a_0}

B(b_0:t_b_0,a_0:t_a_0,…)
    PK:{a_0,b_0}
    FK:{a_0}→A(a_0)

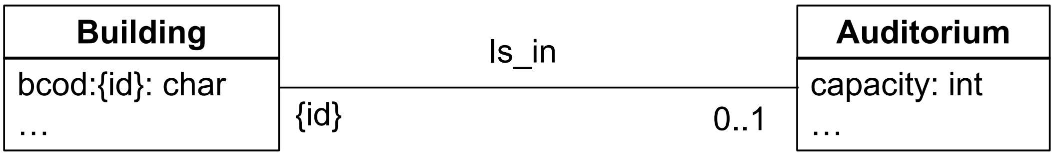



```
Building(bcod:char,…)
    PK:{bcod}

Room(rnum:char, bcod:char, …)
    PK:{rnum, bcod}
    FK:{bcod}→ Building
```

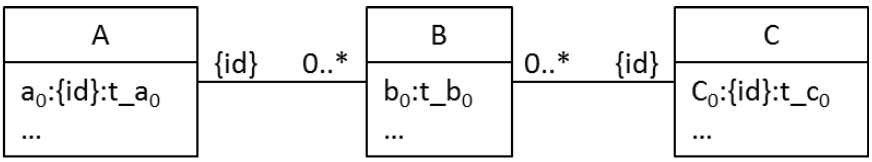
$A(a_0:t\_a_0,…)$
    PK:{a_0}

$B(b_0:t\_b_0,a_0:t\_a_0,…)$
    PK:{a_0}
    FK:{a_0}→$A(a_0)$



Building(bcod: char,…)
    PK:{bcod}

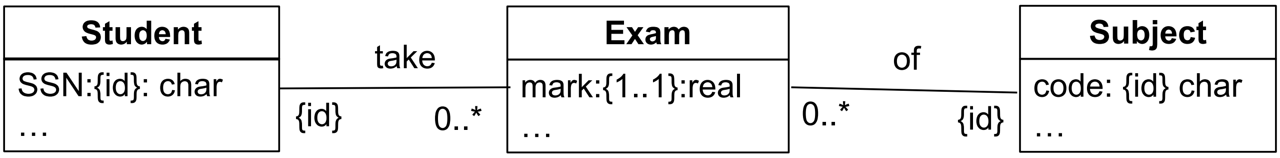
Auditorium(bcod: char, capacity: int, …)
    PK:{bcod}
    FK:{bcod}→ Building



A(a_0:t_a_0,…)
    PK:{a_0}

C(c_0:t_c_0,…)
    PK:{c_0}

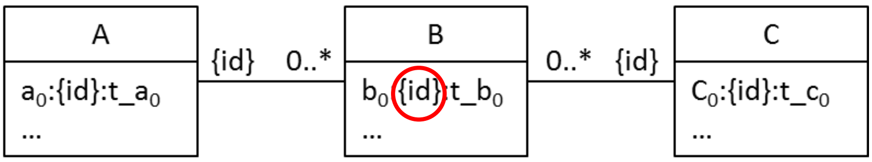
B(a_0:t_a_0,c_0:t_c_0,b_0:t_b_0,…)
    PK:{a_0,c_0}
    FK:{a_0}→A(a_0)
    FK:{c_0}→C(c_0)



Student(SSN:char,…)
    PK:{SSN}

Subject (code: char, …)
    PK:{code}

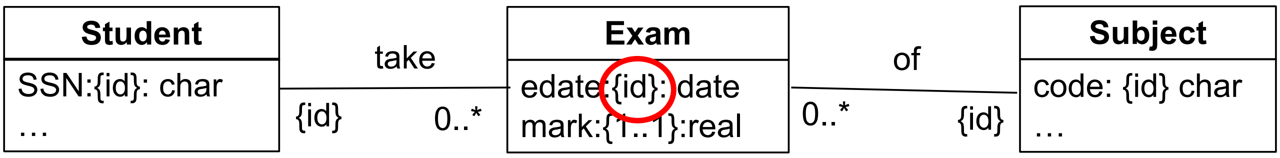
Exam (SSN: char, code: char, mark: real, …)
    PK:{SSN,code}
    FK:{SSN}→ Student
    FK:{code}→ Subject
    NNV:{mark}



A(a_0:t_a_0,…)
    PK:{a_0}

C(c_0:t_c_0,…)
    PK:{c_0}

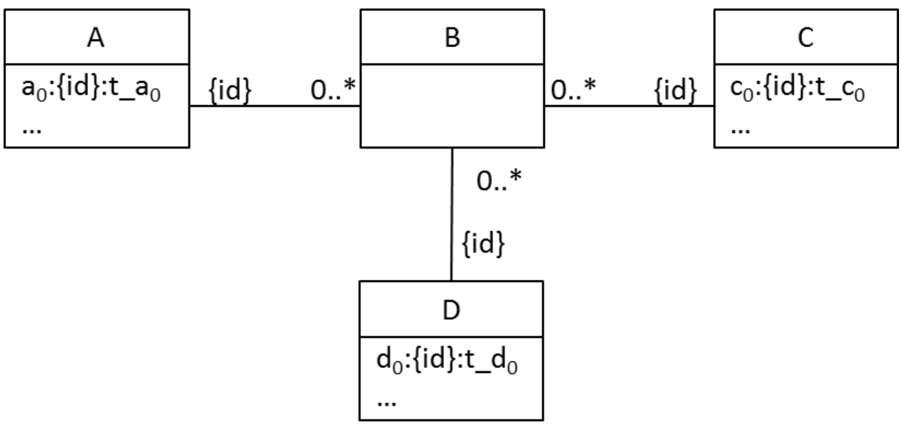
B(a_0:t_a_0,c_0:t_c_0,b_0:t_b_0,…)
    PK:{a_0,c_0,b_0}
    FK:{a_0}→A(a_0)
    FK:{c_0}→C(c_0)



Student(SSN: char,…)
PK:{SSN}

Subject (code: char, …)
PK:{code}

Exam (SSN: char, code: char, edate: date, mark: real, …)
PK:{SSN, code, edate}
FK:{SSN}→ Student
FK:{code}→ Subject
NNV:{mark}



$A(a_0 : t\_a_0, \ldots)$
PK: $\{a_0\}$

$C(c_0 : t\_c_0, \ldots)$
PK: $\{c_0\}$

D(d0:t_d0,…)
PK:{d0}

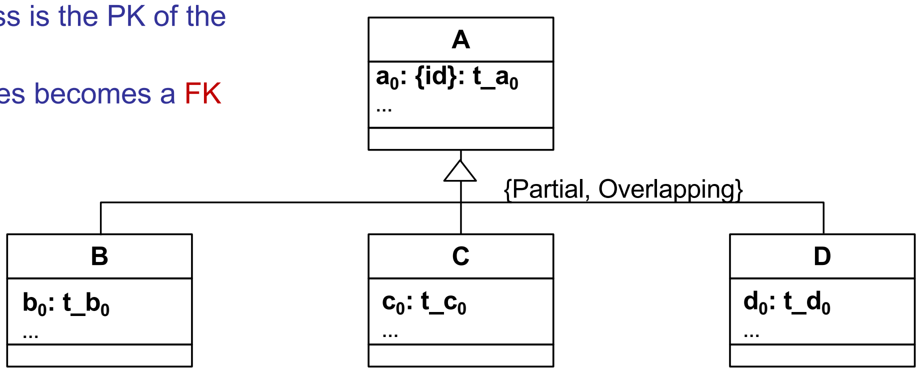
B(a0:t_a0,c0:t_c0,d0:t_d0)
PK: $\{a_0, c_0, d_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{c_0\} \rightarrow C(c_0)$
FK: $\{d_0\} \rightarrow D(d_0)$

Example:
A: Piece
C: Provider
D: Project
B: Supply

Unit 4.3. Logical Design

2.3. Specialization

- The PK of the superclass is the PK of the subclasses.
- The PK of the subclasses becomes a FK to the superclass



A(a_0:t_a_0,…)
  PK:{a_0}

C(a_0:t_a_0,c_0:t_c_0,…)
  PK:{a_0}
  FK:{a_0}→A(a_0)

B(a_0:t_a_0,b_0:t_b_0,…)
  PK:{a_0}
  FK:{a_0}→A(a_0)

D(a_0:t_a_0,d_0:t_d_0,…)
  PK:{a_0}
  FK:{a_0}→A(a_0)

IC $_{Total}$:

Every value which appears in the attribute a_0 of *A* must appear in the attribute a_0 of *B, C* or *D.*

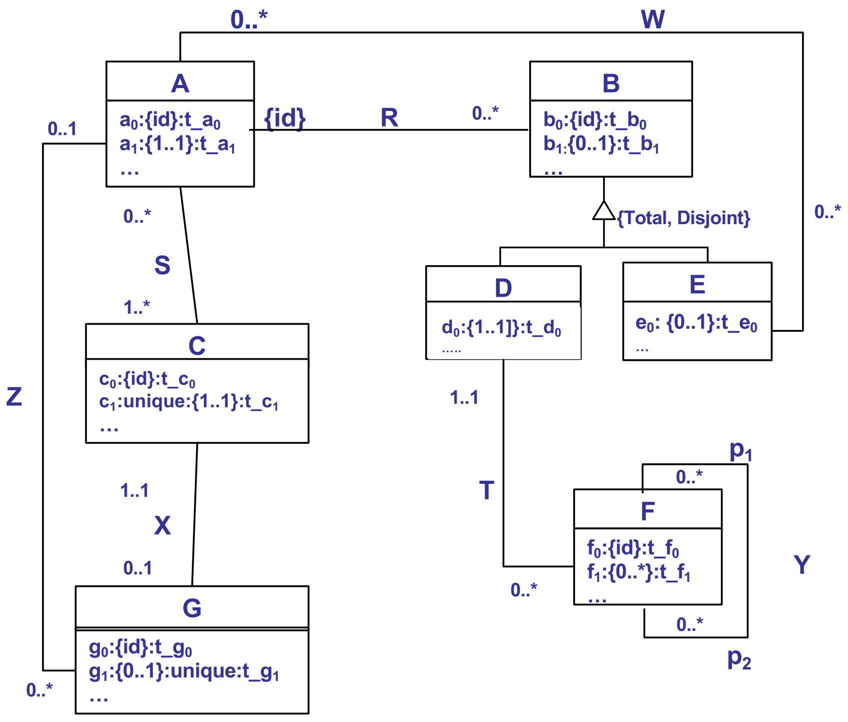
R I $_{Disjoint}$:

There cannot be the same value in the attribute a_0 of *B* and the attribute a_0 of *C*; nor for a_0 of *B* and a_0 of *D*; nor for a_0 of *C* and a_0 of *D.*

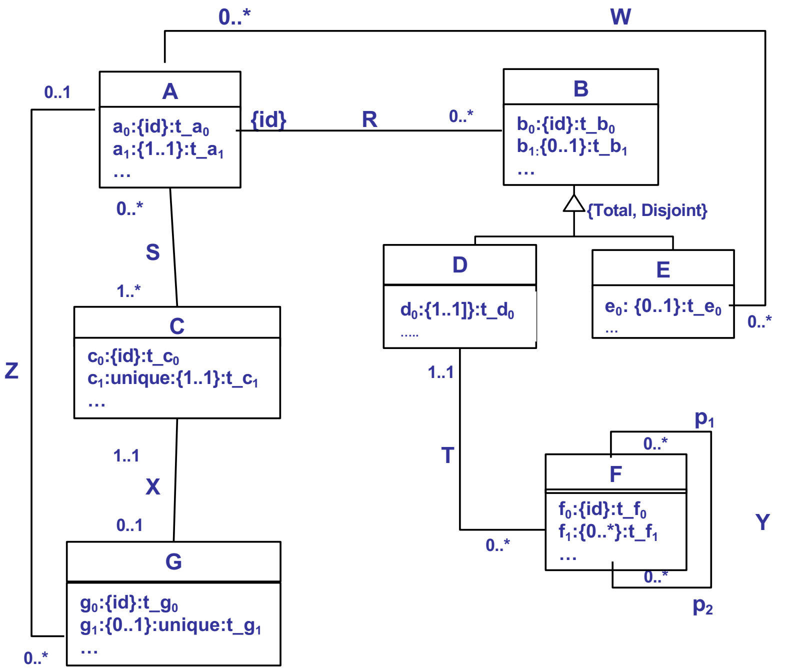
(alternative wording: A value a_0 of *A* cannot appear in more than one attribute a_0 of B, C or D).

```
A(a0:t_a0,…)
   PK:{a0}

B(a0:t_a0,b0:t_b0,…)
  PK:{a0}
  FK:{a0}→A(a0)

C(a0:t_a0,c0:t_c0,…)
   PK:{a0}
   FK:{a0}→A(a0)

D(a0:t_a0,d0:t_d0,…)
   PK:{a0}
   FK:{a0}→A(a0)
```


Exercise 1a: Transform the classes



Exercise 1a: Transform the classes



A(a_0:t_a_0, a_1:t_a_1,…)
PK:{a_0}
NNV:{a_1}

C(c_0:t_c_0, c_1:t_c_1,…)
PK:{c_0}
NNV:{c_1}
UNI:{c_1}

G(g_0:t_g_0, g_1:t_g_1,…)
PK:{g_0}
UNI:{g_1}

F(f_0:t_f_0,…)
PK:{f_0}

F1(f_0:t_f_0, f_1:t_f_1)
PK:{f_0, f_1}
FK:{f_0}→F(f_0)

B(a_0:t_a_0, b_0:t_b_0, b_1:t_b_1,…)
PK:{a_0, b_0}
FK:{a_0}→A(a_0)

D(a_0:t_a_0, b_0:t_b_0, d_0:t_d_0,…)
PK:{a_0, b_0}
NNV:{d_0}
FK:{a_0, b_0}→B(a_0, b_0)

E(a_0_E:t_a_0, b_0_E:t_b_0, e_0:t_e_0,…)
PK:{a_0, b_0}
FK:{a_0, b_0}→B(a_0, b_0)

+ IC1: Total: Every pair (a_0,b_0) of B must appear in a tuple of D or E

+ IC2: Disjoint: There cannot be a pair (a_0,b_0) appearing in D and E at the same time

Unit 4.3. Logical Design

Methodology to obtain the relational schema

I. Transform the classes into relations
   1. Strong classes
   2. Weak classes
   3. Specialized classes

II. Transform the associations according to their multiplicity:

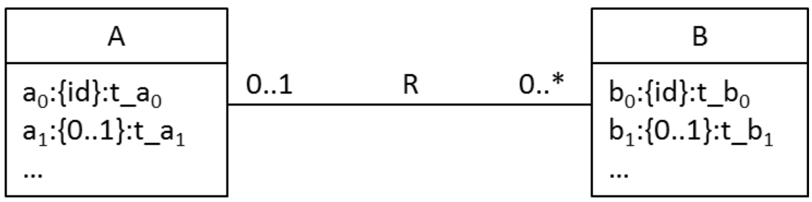
_..*:_..* ⟹ Add a new R

_..*:_..1
_..1:_..1 ⟹ 1..1: Do not add any R. Represent the association in the relation with the 1..1 multiplicity (Other existence constraint could be added to the system)

0..1: Does it have any link attribute?
   Yes: Add a new R
   No: Do not add any R. Represent the association in the relation with 0..1 multiplicity

III. Those properties that can't be represented in the relational schema, will be expressed in a list of integrity constraints

0..1 : 0..* Association

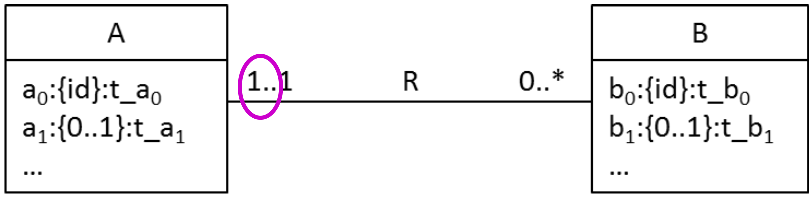


A(a_0:t_a_0,a_1:t_a_1,…)
  PK:{a_0}

B(b_0:t_b_0,b_1:t_b_1,…,a_0:t_a_0)
  PK:{b_0}
  FK:{a_0}→A(a_0)

Example:
  A: Person
  B: Car
  R: buys

1..1 : 0..* Association

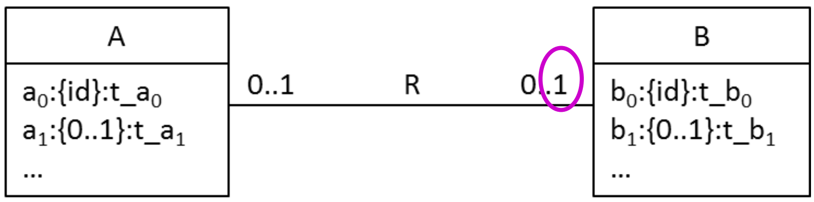


$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots, a_0:t\_a_0)$
  PK: $\{b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$
  NNV: $\{a_0\}$

Example:
A: Person
B: Car
R: owns

0..1 : 0..1 Association

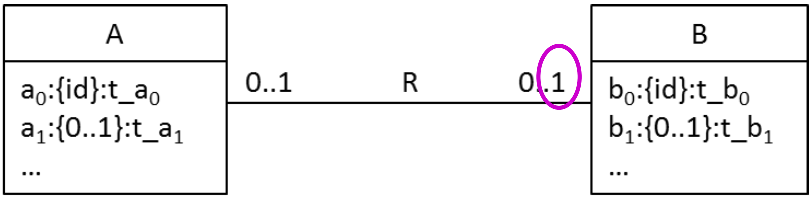


<u>Option 1</u>

$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots, a_0:t\_a_0)$
  PK: $\{b_0\}$
  UNI: $\{a_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$

0..1 : 0..1 Association



Option 2

$A(a_0:t\_a_0, a_1:t\_a_1, \dots, b_0:t\_b_0)$

$PK: \{a_0\}$

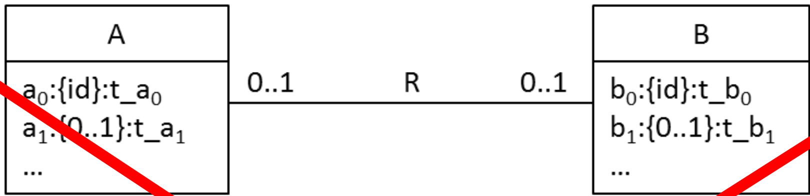
$UNI: \{b_0\}$

$FK: \{b_0\} \rightarrow B(b_0)$

$B(b_0:t\_b_0, b_1:t\_b_1, \dots)$

$PK: \{b_0\}$

0..1 : 0..1 Association



Option 3

$A(a_0:t\_a_0, a_1:t\_a_1, \dots)$

$PK: \{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \dots)$

$PK: \{b_0\}$

$R(b_0:t\_b_0, a_0:t\_a_0)$

$PK: \{b_0\}$

$UNI: \{a_0\}$

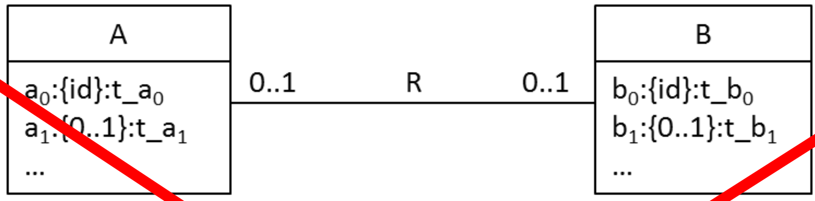
$NNV: \{a_0\}$

$FK: \{a_0\} \rightarrow A(a_0)$

$FK: \{b_0\} \rightarrow B(b_0)$

There are more relations:
It is worse

0..1 : 0..1 Association



Option 4

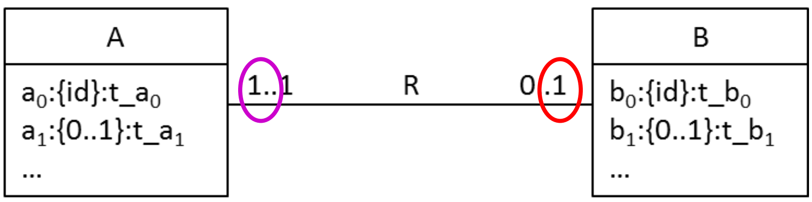
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
  PK: $\{b_0\}$

$R(b_0{:}t\_b_0, a_0{:}t\_a_0)$
  PK: $\{a_0\}$
  UNI: $\{b_0\}$
  NNV: $\{b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$
  FK: $\{b_0\} \rightarrow B(b_0)$

There are more relations:
It is worse

1..1 : 0..1 Association

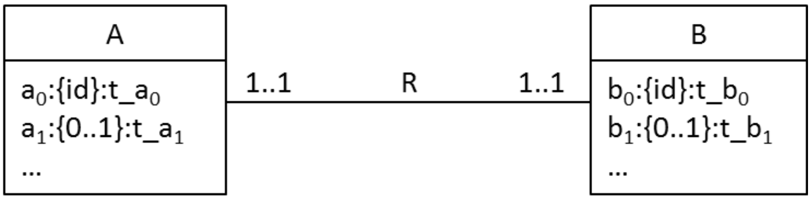


$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0)$
  PK: $\{b_0\}$
  UNI: $\{a_0\}$
  NNV: $\{a_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$

Example:
A: Passenger
B: Seat
(in a plane)

1..1 : 1..1 Association

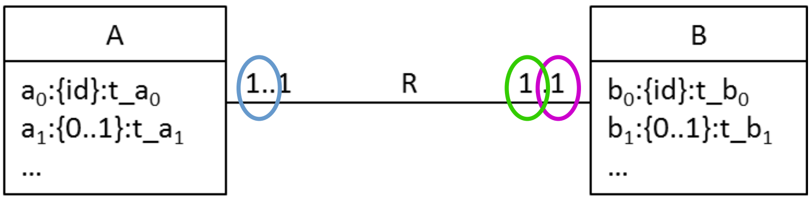


<u>Option 1</u>

A-B(a_0:t_a_0, a_1:t_a_1, …, b_0:t_b_0, b_1:t_b_1, …)
  PK: {a_0}
  UNI: {b_0}
  NNV: {b_0}

A-B objects are more complex to be manipulated

1..1 : 1..1 Association



<u>Option 2</u>

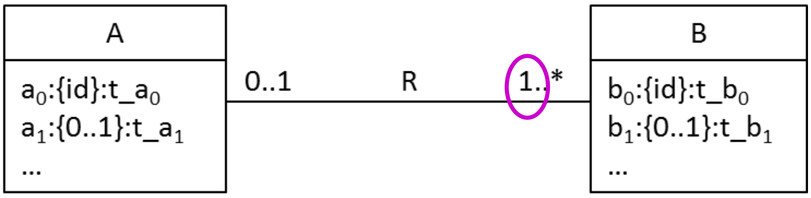
A(a_0:t_a_0, a_1:t_a_1, …)
  PK: {a_0}
  FK: {a_0}→B(a_0)

This FK is possible because a_0 in B has Uniqueness constraint

B(b_0:t_b_0, b_1:t_b_1, …, a_0:t_a_0)
  PK: {b_0}
  UNI: {a_0}
  NNV: {a_0}
  FK: {a_0}→ A(a_0)

Best option

0..1 : 1..* Association



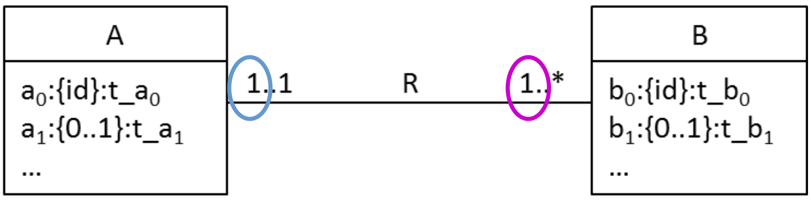
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0)$
  PK: $\{b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$

IC1: *Every value in a_0 of A must appear in a_0 of B.*

Example:
A: Company
B: Worker
R: has

1..1 : 1..* Association

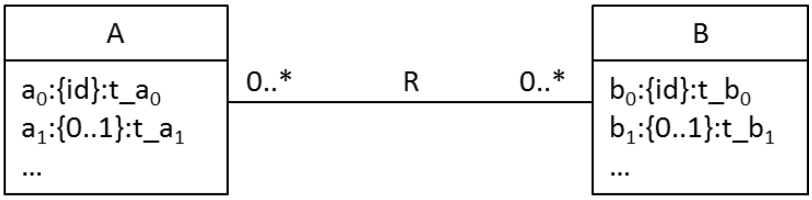


$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0)$
  PK: $\{b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$
  NNV: $\{a_0\}$

IC1: *Every value in a_0 of A must appear in a_0 of B.*

0..* : 0..* Association



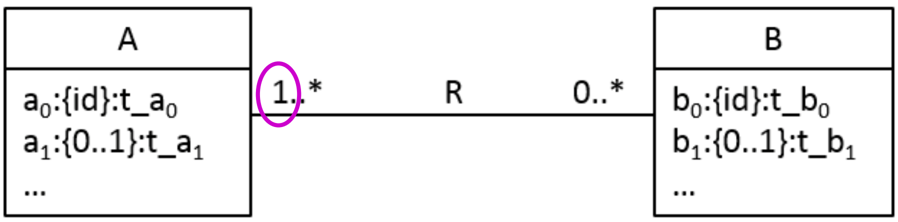
A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}

B(b_0:t_b_0,b_1:t_b_1,…)
PK:{b_0}

R(a_0:t_a_0,b_0:t_b_0)
PK:{a_0,b_0}
FK:{a_0}→A(a_0)
FK:{b_0}→B(b_0)

Example:
A: Book
B: Person
R: has_as_author

1..* : 0..* Association



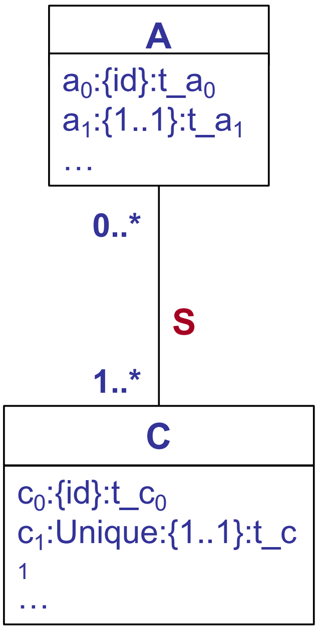
A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}

B(b_0:t_b_0,b_1:t_b_1,…)
PK:{b_0}

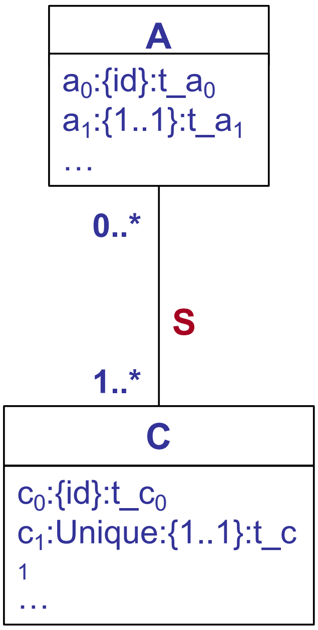
R(a_0:t_a_0,b_0:t_b_0)
PK:{a_0,b_0}
FK:{a_0}→A(a_0)
FK:{b_0}→B(b_0)

IC1: *Every value in b_0 of B must appear in b_0 of R.*

Exercise 2. Transform the association



Exercise 2. Transform the association



A(a_0:t_a_0, a_1:t_a_1,…)
PK:{a_0}
NNV:{a_1}

C(c_0:t_c_0, c_1:t_c_1,…)
PK:{c_0}
NNV:{c_1}
UNI:{c_1}

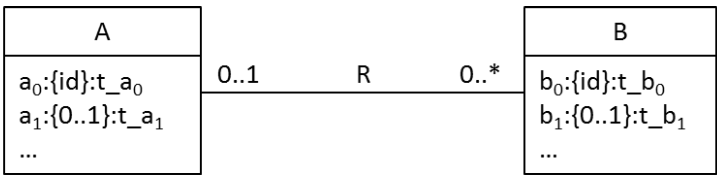
S(a_0:t_a_0, c_0:t_c_0)
PK:{a_0, c_0}
FK:{a_0}→A(a_0)
FK:{c_0}→C(c_0)

IC3: Existence constraint of A in S: Every value in *a_0 of A must appear in a_0 of S.*

Unit 4.3. Logical Design

0..1 : 0..* Reflexive association

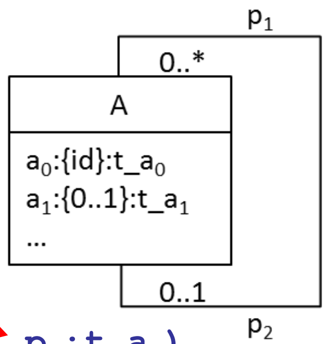
Reflexive associations are handle as any other binary association



a_0	a_1	…	a_0_p_2
1	b	…	
2	r	…	1
3	n	…	2
4	m	…	1

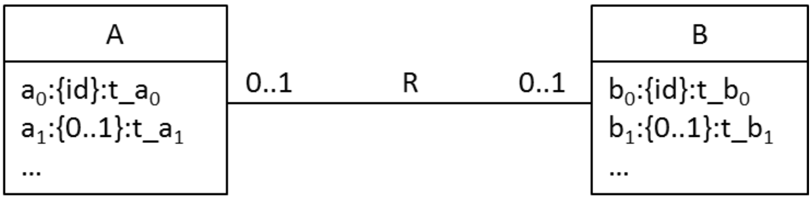
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0)$
  PK: $\{b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$



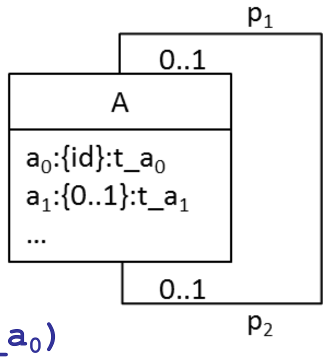
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots, a_0\_p_2{:}t\_a_0)$
  PK: $\{a_0\}$
  FK: $\{a_0\_p_2\} \rightarrow A(a_0)$

0..1 : 0..1 Reflexive association



A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}

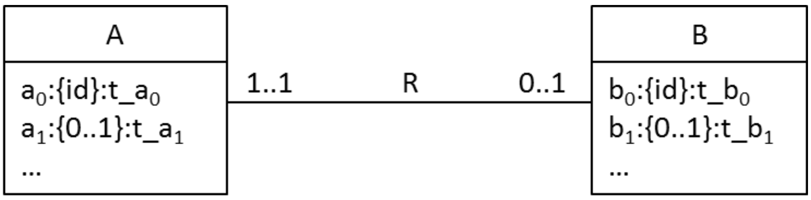
B(b_0:t_b_0,b_1:t_b_1,…,a_0:t_a_0)
PK:{b_0}
UNI:{a_0}
FK:{a_0}→A(a_0)



A(a_0:t_a_0,a_1:t_a_1,…,a_0_p_2:t_a_0)
PK:{a_0}
UNI:{a_0_p_2}
FK:{a_0_p_2}→A(a_0)

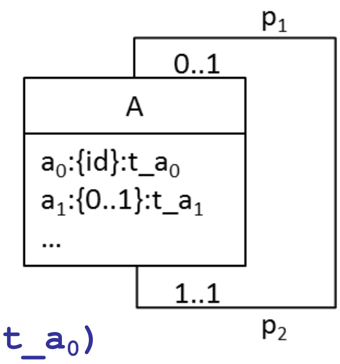
Example:
A: Person
R: husband_of

0..1 : 1..1 Reflexive association



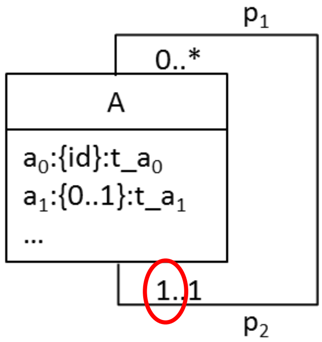
A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}

B(b_0:t_b_0,b_1:t_b_1,…,a_0:t_a_0)
PK:{b_0}
UNI:{a_0}
NNV:{a_0}
FK:{a_0}→A(a_0)



A(a_0:t_a_0,a_1:t_a_1,…,a_0_p_2:t_a_0)
PK:{a_0}
UNI:{a_0_p_2}
NNV:{a_0_p_2}
FK:{a_0_p_2}→A(a_0)

1..1 : 0..* Reflexive association



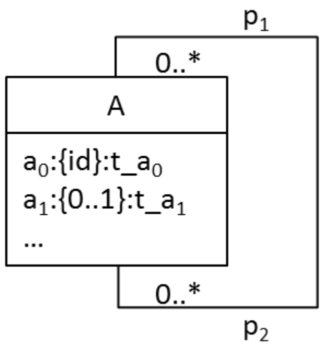
$A(a_0 : t\_a_0, a_1 : t\_a_1, \ldots, a_0\_p_2 : t\_a_0)$

$PK: \{a_0\}$

$FK: \{a_0\_p_2\} \rightarrow A(a_0)$

$NNV: \{a_0\_p_2\}$

0..* : 0..* Reflexive association



$A(a_0 : t\_a_0, a_1 : t\_a_1, \ldots)$

$PK: \{a_0\}$

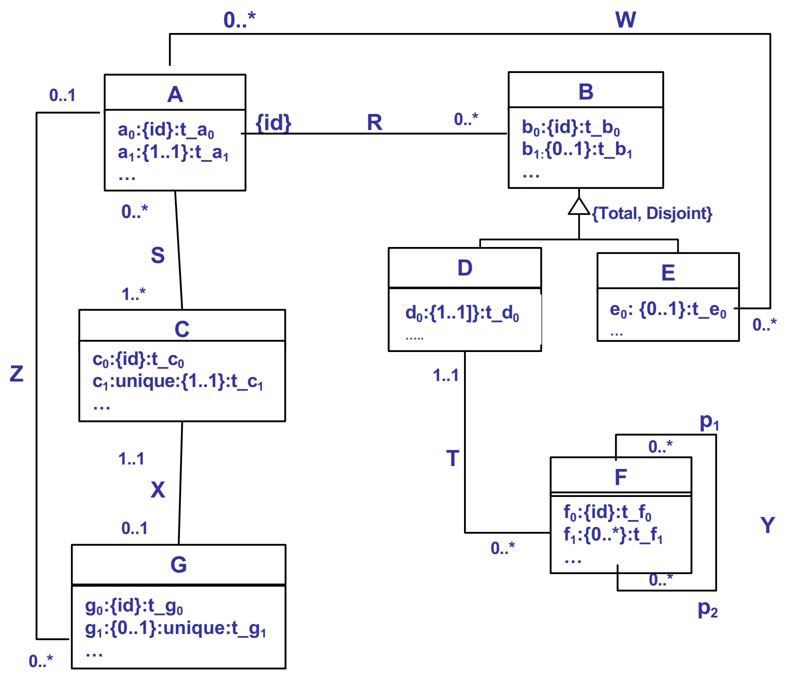
$R(a_0\_p_1 : t\_a_0, a_0\_p_2 : t\_a_0)$

$PK: \{a_0\_p_1, a_0\_p_2\}$

$FK: \{a_0\_p_1\} \rightarrow A(a_0)$

$FK: \{a_0\_p_2\} \rightarrow A(a_0)$

Exercise 1b. Transform the associations



A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}
NNV:{a_1}

C(c_0:t_c_0,c_1:t_c_1,…)
PK:{c_0}
NNV:{c_1}
UNI:{c_1}

G(g_0:t_g_0,g_1:t_g_1,…)
PK:{g_0}
UNI:{g_1}

F(f_0:t_f_0,…)
PK:{f_0}

F1(f_0:t_f_0,f_1:t_f_1)
PK:{f_0,f_1}
FK:{f_0}→F(f_0)

B(a_0:t_a_0,b_0:t_b_0,b_1:t_b_1,…)
PK:{a_0,b_0}
FK:{a_0}→A(a_0)

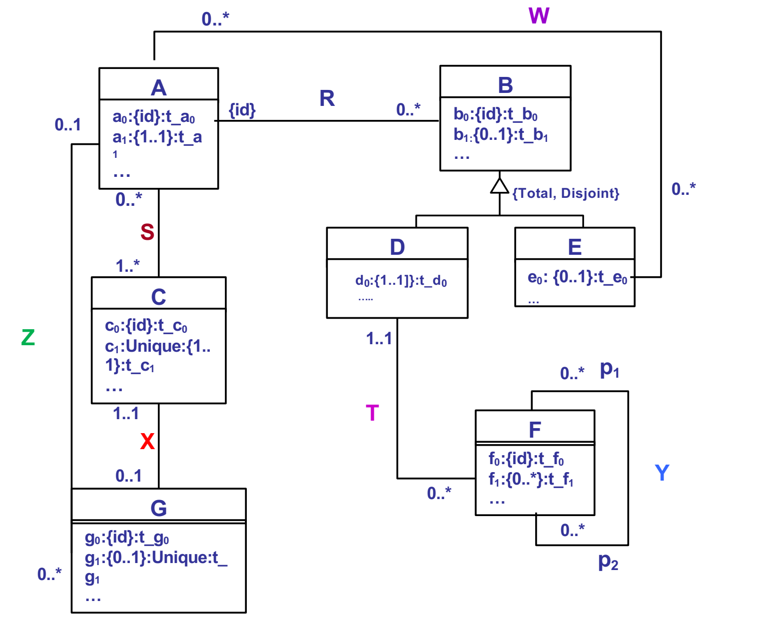
D(a_0:t_a_0,b_0:t_b_0,d_0:t_d_0,…)
PK:{a_0,b_0}
NNV:{d_0}
FK:{a_0,b_0}→B(a_0,b_0)

E(a_0_E:t_a_0,b_0_E:t_b_0,e_0:t_e_0,…)
PK:{a_0,b_0}
FK:{a_0,b_0}→B(a_0,b_0)

+ IC1: Total: Every pair (a_0,b_0) od B must appear in a tuple of D or E

+ IC2: Disjoint: There cannot be a pair (a_0,b_0) appearing in D and E at the same time

Exercise 1b. Transform the associations



S(a_0:t_a_0,c_0:t_c_0)
PK:{a_0,c_0}
FK:{a_0}→A(a_0)
FK:{c_0}→C(c_0)

+ IC3: Existence constraint of A in S: Every value in *a_0 of A must appear in a_0 de S.*

Y(f_0_p_1:t_f_0,f_0_p_2:t_f_0)
PK:{f_0_p_1,f_0_p_2}
FK:{f_0_p_1}→F(f_0)
FK:{f_0_p_2}→F(f_0)

W(a_0:t_a_0,a_0_E:t_a_0,b_0_E:t_b_0)
PK:{a_0,a_0_E,b_0_E}
FK:{a_0}→A(a_0)
FK:{a_0_E,b_0_E}→E(a_0_E,b_0_E)

A(a_0:t_a_0,a_1:t_a_1,…)
PK:{a_0}
NNV:{a_1}

C(c_0:t_c_0,c_1:t_c_1,…)
PK:{c_0}
NNV:{c_1}
UNI:{c_1}

G(g_0:t_g_0,g_1:t_g_1,…, c_0:t_c_0,a_0:t_a_0)
PK:{g_0}
UNI:{g_1}
NNV:{c_0}
UNI:{c_0}
FK:{c_0}→C(c_0)
FK:{a_0}→A(a_0)

F(f_0:t_f_0,…,a_0:t_a_0,b_0:t_b_0)
PK:{f_0}
FK:{a_0,b_0}→D(a_0,b_0)
NNV:{a_0,b_0}

F1(f_0:t_f_0,f_1:t_f_1)
PK:{f_0,f_1}
FK:{f_0}→F(f_0)

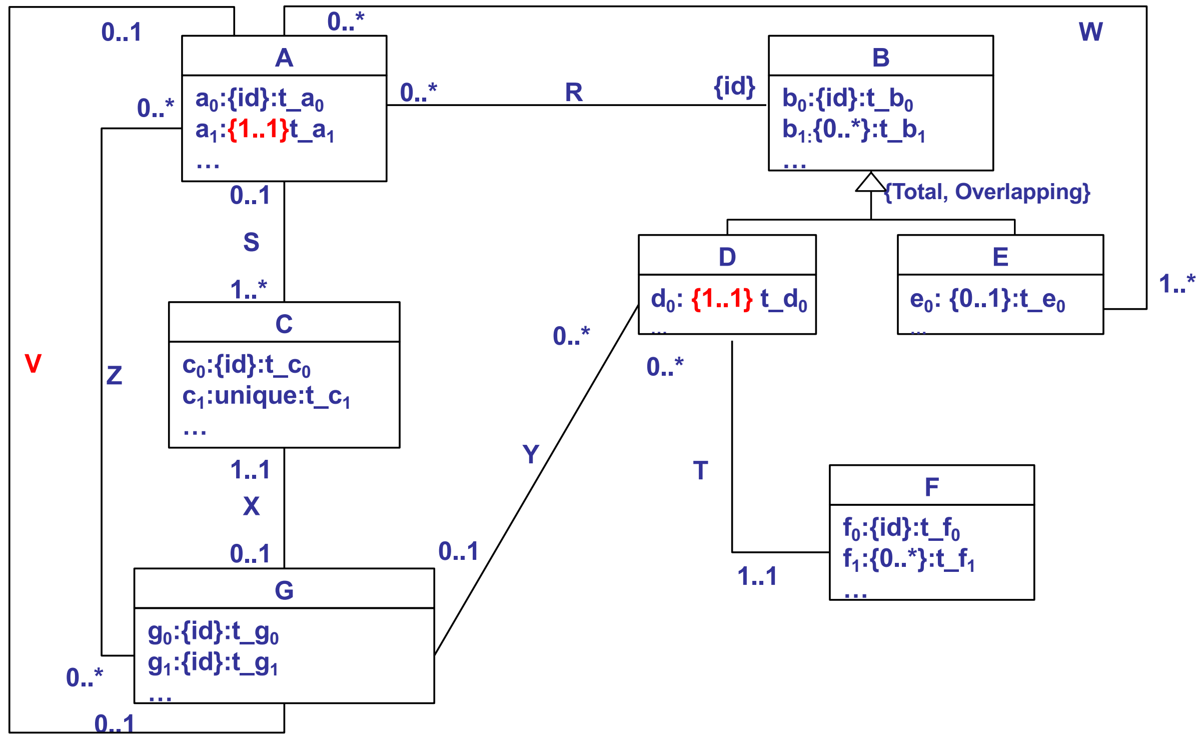
B(a_0:t_a_0,b_0:t_b_0,b_1:t_b_1,…)
PK:{a_0,b_0}
FK:{a_0}→A(a_0)

D(a_0:t_a_0,b_0:t_b_0,d_0:t_d_0,…)
PK:{a_0,b_0}
NNV:{d_0}
FK:{a_0,b_0}→B

E(a_0_E:t_a_0,b_0_E:t_b_0,e_0:t_e_0,…)
PK:{a_0,b_0}
FK:{a_0,b_0}→B

+ IC2: Disjoint (see previous slide)
+ IC1: Total (see previous slide)

Exercise 3



Exercise 3

(without directives)

Class transformation:

B(b_0: t_b_0, …)
PK: {b_0}

B_b1(b_0: t_b_0, b_1: t_b_1)
PK: {b_0, b_1}
FK: {b_0} → B

b_1 is multi-valued

A(a_0: t_a_0, a_1: t_a_1, …, b_0: t_b_0)
PK: {a_0, b_0}
FK: {b_0} → B
NNV: {a_1}

A is weak

D(b_0: t_b_0, d_0: t_d_0, …)
PK: {b_0}
FK: {b_0} → B
NNV: {d_0}

D specializes B

C(c_0: t_c_0, c_1: t_c_1, …)
PK: {c_0}
UNI: {c_1}

F(f_0: t_f_0, …)
PK: {f_0}

F_f1(f_0: t_f_0, f_1: t_f_1)
PK: {f_0, f_1}
FK: {f_0} → F

f_1 is multi-valued

G(g_0: t_g_0, g_1: t_g_1, …)
PK: {g_0, g_1}

E(b_0: t_b_0, e_0: t_e_0, …)
PK: {b_0}
FK: {b_0} → B

E specializes B

IC_{Total}: "Every value which appears in *b_0* of *B* must appear in *b_0* of *D or E*".

Exercise 3

(without directives)

Association transformation:

$B(b_0\text{: t\_b}_0, \ldots)$
PK: $\{b_0\}$
$B\_b1(b_0\text{: t\_b}_0, b_1\text{: t\_b}_1)$
PK: $\{b_0, b_1\}$
FK: $\{b_0\} \rightarrow B$
$A(a_0\text{: t\_a}_0, a_1\text{: t\_a}_1, \ldots, b_0\text{: t\_b}_0)$
PK: $\{a_0, b_0\}$
FK: $\{b_0\} \rightarrow B$
NNV: $\{a_1\}$
$D(b_0\text{: t\_b}_0, d_0\text{: t\_d}_0, \ldots, f_0\text{: t\_f}_0, g_0\text{: t\_g}_0, g_1\text{: t\_g}_1)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$
NNV: $\{d_0\}$
FK: $\{f_0\} \rightarrow F$
NNV: $\{f_0\}$
FK: $\{g_0, g_1\} \rightarrow G$
$E(b_0\text{: t\_b}_0, e_0\text{: t\_e}_0, \ldots)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$

T

Y

IC_{Total}: "Every value which appears in b_0 of *B* must appear in b_0 of *D or E*".

$C(c_0\text{: t\_c}_0, c_1\text{: t\_c}_1, \ldots, a_0\text{: t\_a}_0, b_0\text{: t\_b}_0)$
PK: $\{c_0\}$
UNI: $\{c_1\}$
FK: $\{a_0, b_0\} \rightarrow A(a_0, b_0)$
IC: All $\{a_0, b_0\}$ in A must be in C

S

$F(f_0\text{: t\_f}_0, \ldots)$
PK: $\{f_0\}$
$F\_f1(f_0\text{: t\_f}_0, f_1\text{: t\_f}_1)$
PK: $\{f_0, f_1\}$
FK: $\{f_0\} \rightarrow F(f_0)$
$G(g_0\text{: t\_g}_0, g_1\text{: t\_g}_1, \ldots, c_0\text{: t\_c}_0, a_0\text{: t\_a}_0, b_0\text{: t\_b}_0)$
PK: $\{g_0, g_1\}$
FK: $\{c_0\} \rightarrow C$
NNV: $\{c_0\}$ UNI: $\{c_0\}$
FK: $\{a_0, b_0\} \rightarrow A(a_0, b_0)$
UNI: $\{a_0, b_0\}$

X

V

(can be symmetrically solved in A)

$W(a_0\text{: t\_a}_0, b_{0A}\text{: t\_b}_0, b_{0E}\text{: t\_b}_0)$
PK: $\{a_0, b_{0A}, b_{0E}\}$
FK: $\{a_0, b_{0A}\} \rightarrow A(a_0, b_{0A})$ FK: $\{b_{0E}\} \rightarrow E(b_0)$

IC: For every $\{a_0, b_0\}$ in A there must be a corresponding $\{a_0, b_{0A}\}$ in W

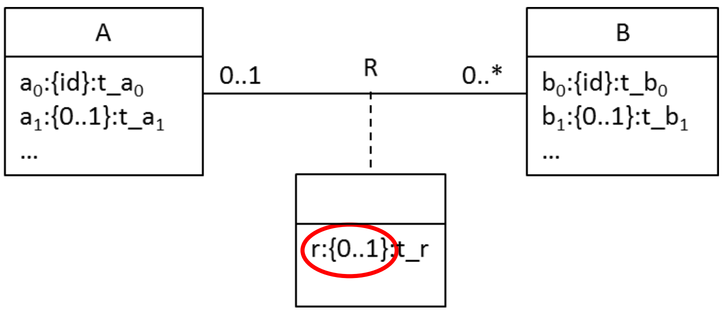
$Z(a_0\text{: t\_a}_0, b_0\text{: t\_b}_0, g_0\text{: t\_g}_0, g_1\text{: t\_g}_1)$
PK: $\{a_0, b_0, g_0, g_1\}$
FK: $\{a_0, b_0\} \rightarrow A(a_0, b_0)$
FK: $\{g_0, g_1\} \rightarrow G(g_0, g_1)$

Unit 4.3. Logical Design

Associations with link attributes

- The link attributes are added to the table where the association is represented.
- When link attributes are presented, the transformation schemas showed in the previous sections could be wrong:
  - Add integrity constraints (I. C.)
  - Add new tables

Associations with link attributes. Case 1

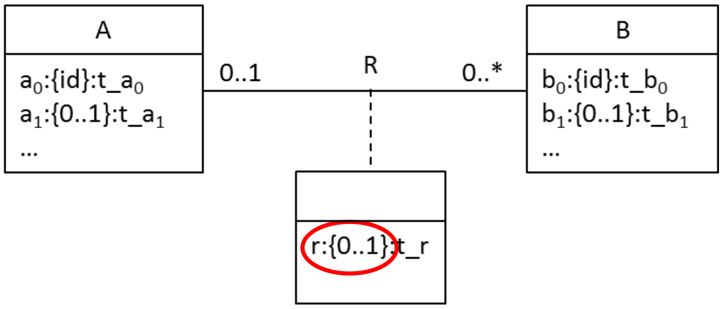


Example:
A: Company
B: Person
R: working_for
r: salary

$\mathbf{A(a_0:t\_a_0, a_1:t\_a_1, \ldots)}$
  $\mathbf{PK: \{a_0\}}$
$\mathbf{B(b_0:t\_b_0, b_1:t\_b_1, \ldots, a_0:t\_a_0, r:t\_r)}$
  $\mathbf{PK: \{b_0\}}$
  $\mathbf{FK: \{a_0\} \rightarrow A(a_0)}$

IC1: There can't be a tuple in B where a_0 is NULL and r is not null

Associations with link attributes. Case 1



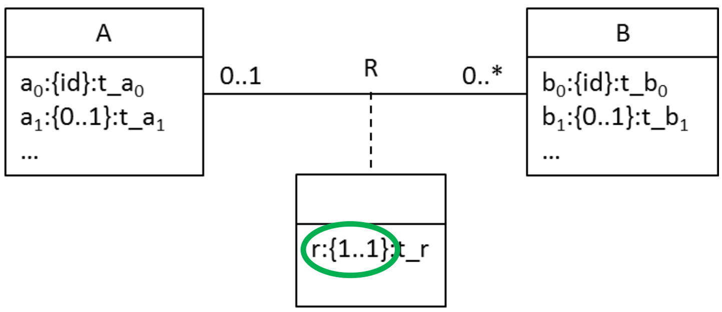
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
PK: $\{b_0\}$

$R(b_0{:}t\_b_0, a_0{:}t\_a_0, r{:}t\_r)$
PK: $\{b_0\}$
NNV: $\{a_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{b_0\} \rightarrow B(b_0)$

Better solution:
There is no general constraint

Associations with link attributes. Case 2

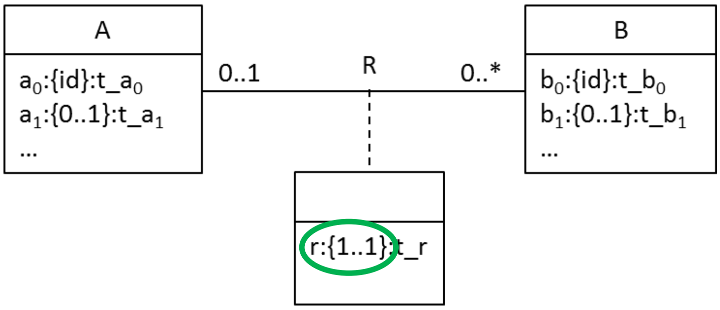


$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0, r{:}t\_r)$
PK: $\{b_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$

IC1: There can't exists a tuple in B where a_0 is null and r is not null, or vice versa.

Associations with link attributes. Case 2



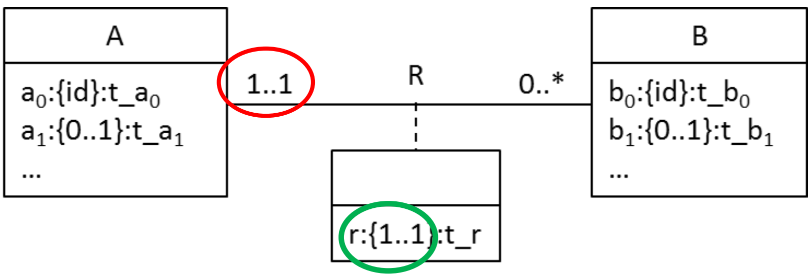
$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
PK: $\{b_0\}$

$R(b_0{:}t\_b_0, a_0{:}t\_a_0, r{:}t\_r)$
PK: $\{b_0\}$
NNV: $\{a_0\}$
NNV: $\{r\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{b_0\} \rightarrow B(b_0)$

Better solution
There is no integrity constraint.

Associations with link attributes. Case 3

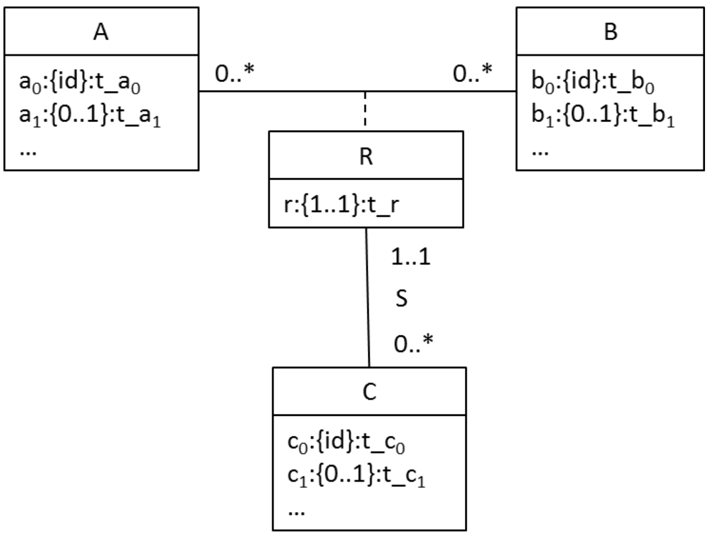


$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots, a_0{:}t\_a_0, r{:}t\_r)$
PK: $\{b_0\}$
NNV: $\{a_0\}$
NNV: $\{r\}$
FK: $\{a_0\} \rightarrow A$

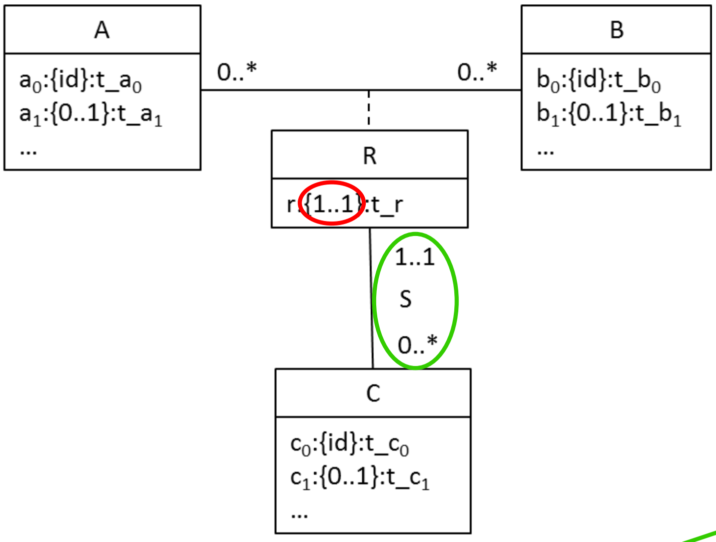
Unit 4.3. Logical Design

Association within association: association classes



1. Transform the association between A and B (following the previous transformation schemas)
2. Transform the other associations (S)
3. Check the correctness of the whole transformation

Association within association. Example 1



$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$

PK: $\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots)$

PK: $\{b_0\}$

$R(a_0:t\_a_0, b_0:t\_b_0, r:t\_r)$

PK: $\{a_0, b_0\}$

FK: $\{a_0\} \rightarrow A(a_0)$

FK: $\{b_0\} \rightarrow B(b_0)$

NNV: $\{r\}$

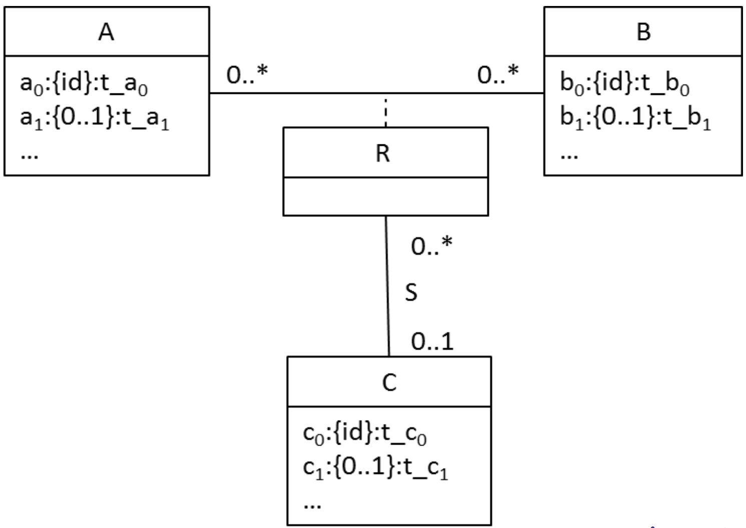
$C(c_0:t\_c_0, c_1:t\_c_1, \ldots, a_0:t\_a_0, b_0:t\_b_0)$

PK: $\{c_0\}$

FK: $\{a_0, b_0\} \rightarrow R(a_0, b_0)$

NNV: $\{a_0, b_0\}$

Association within association. Example 2



$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$

PK: $\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots)$

PK: $\{b_0\}$

$C(c_0:t\_c_0, c_1:t\_c_1, \ldots)$

PK: $\{c_0\}$

$R(a_0:t\_a_0, b_0:t\_b_0, c_0:t\_c_0)$

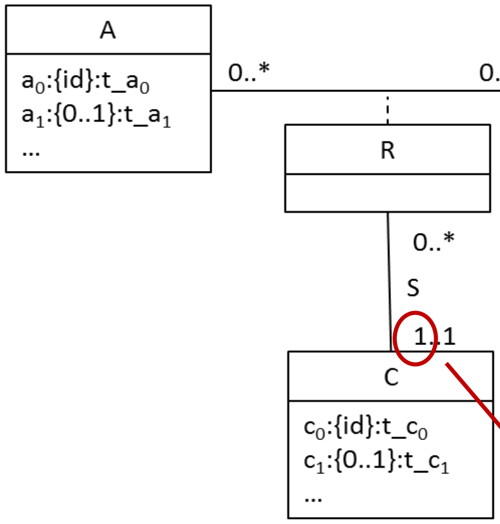
PK: $\{a_0, b_0\}$

FK: $\{a_0\} \rightarrow A(a_0)$

FK: $\{b_0\} \rightarrow B(b_0)$

FK: $\{c_0\} \rightarrow C(c_0)$

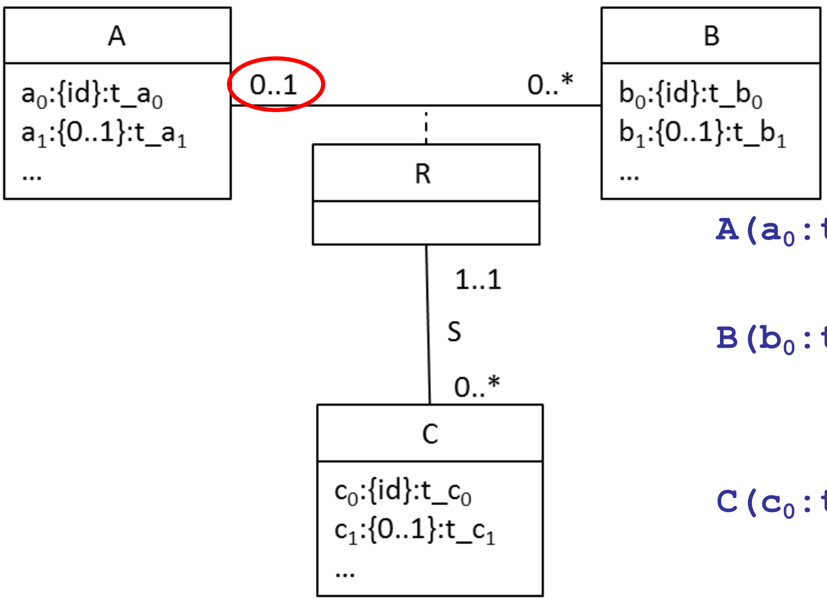
Association within association. Example 3



A(a_0:t_a_0,a_1:t_a_1,...)
PK:{a_0}
B(b_0:t_b_0,b_1:t_b_1,...)
PK:{b_0}
C(c_0:t_c_0,c_1:t_c_1,...)
PK:{c_0}

R(a_0:t_a_0,b_0:t_b_0,c_0:t_c_0)
PK:{a_0,b_0}
FK:{a_0} → A(a_0)
FK:{b_0} → B(b_0)
FK:{c_0} → C(c_0)
NNV:{c_0}

Association within association. Example 4

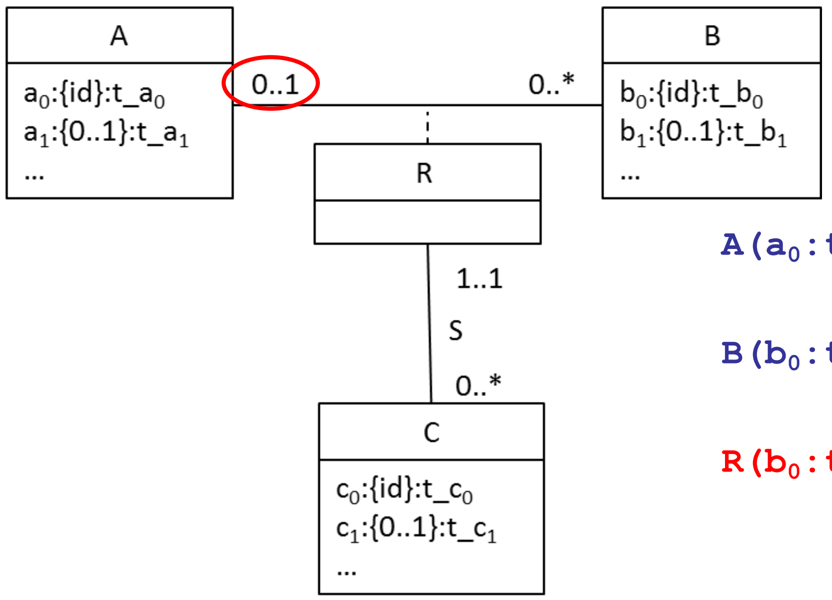


It can be improved

A(a_0:t_a_0,a_1:t_a_1,...)
PK:{a_0}
B(b_0:t_b_0,b_1:t_b_1,...,a_0:t_a_0)
PK:{b_0}
FK:{a_0}→ A(a_0)
C(c_0:t_c_0,c_1:t_c_1,...,b_0:t_b_0)
PK:{c_0}
FK:{b_0}→B
NNV:{b_0}

IC1: There can't exists a tuple in C associated with a tuple in B which is not associated with A

Association within association. Example 4



Better transformation

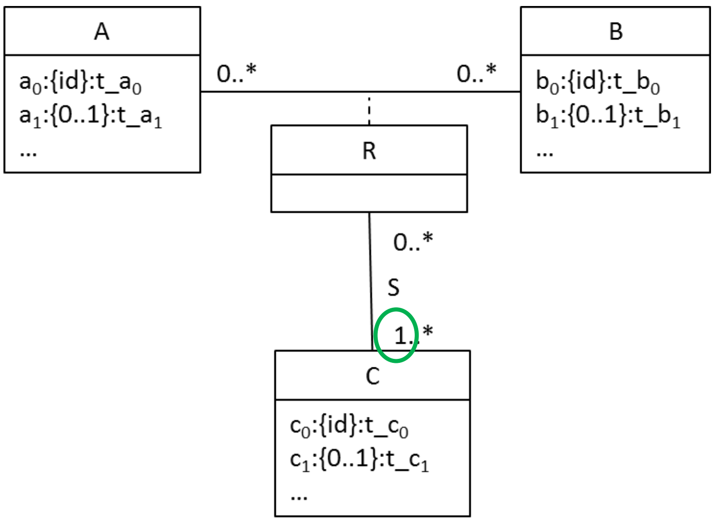
$A(a_0 : t\_a_0, a_1 : t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0 : t\_b_0, b_1 : t\_b_1, \ldots)$
PK: $\{b_0\}$

$R(b_0 : t\_b_0, a_0 : t\_a_0)$
PK: $\{b_0\}$
NNV: $\{a_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{b_0\} \rightarrow B(b_0)$

$C(c_0 : t\_c_0, c_1 : t\_c_1, \ldots, b_0 : t\_b_0)$
PK: $\{c_0\}$
FK: $\{b_0\} \rightarrow R(b_0)$
NNV: $\{b_0\}$

Association within association. Example 5



$A(a_0 : t\_a_0, a_1 : t\_a_1, \ldots)$
PK: $\{a_0\}$

$B(b_0 : t\_b_0, b_1 : t\_b_1, \ldots)$
PK: $\{b_0\}$

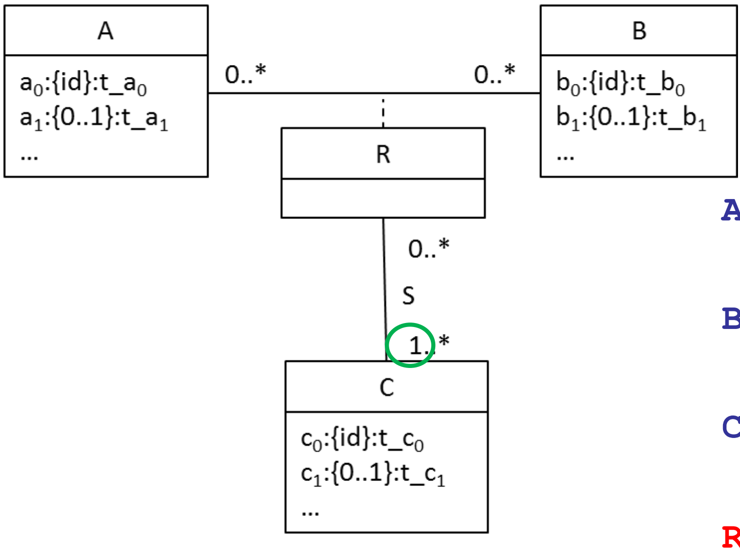
$C(c_0 : t\_c_0, c_1 : t\_c_1, \ldots)$
PK: {c0}

$R(a_0 : t\_a_0, b_0 : t\_b_0)$
PK: $\{a_0, b_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{b_0\} \rightarrow B(b_0)$

$S(a_0 : t\_a_0, b_0 : t\_b_0, c_0 : t\_c_0)$
PK: $\{a_0, b_0, c_0\}$
FK: $\{a_0, b_0\} \rightarrow R(a_0, b_0)$
FK: $\{c_0\} \rightarrow C(c_0)$

IC1: (Existence constraint of R in S) **There can't exists a pair (a_0, b_0) in R which is not in *S*.**

Association within association. Example 5

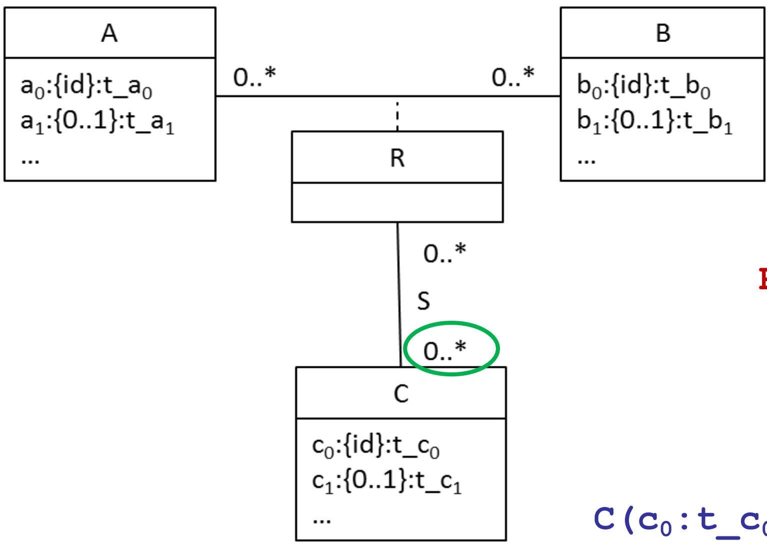


Better transformation

$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
  PK: $\{b_0\}$
$C(c_0{:}t\_c_0, c_1{:}t\_c_1, \ldots)$
  PK: $\{c_0\}$
$RyS(a_0{:}t\_a_0, b_0{:}t\_b_0, c_0{:}t\_c_0)$
  PK: $\{a_0, b_0, c_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$
  FK: $\{b_0\} \rightarrow B(b_0)$
This represents S → FK: $\{c_0\} \rightarrow C(c_0)$

NOTE: This transformation is not possible when R is part of other association

Association within association. Example 6



$A(a_0{:}t\_a_0, a_1{:}t\_a_1, \ldots)$
  PK: $\{a_0\}$
$B(b_0{:}t\_b_0, b_1{:}t\_b_1, \ldots)$
  PK: $\{b_0\}$
$R(a_0{:}t\_a_0, b_0{:}t\_b_0)$
  PK: $\{a_0, b_0\}$
  FK: $\{a_0\} \rightarrow A(a_0)$
  FK: $\{b_0\} \rightarrow B(b_0)$
$C(c_0{:}t\_c_0, c_1{:}t\_c_1, \ldots)$
  PK: $\{c_0\}$
$S(a_0{:}t\_a_0, b_0{:}t\_b_0, c_0{:}t\_c_0)$
  PK: $\{a_0, b_0, c_0\}$
  FK: $\{a_0, b_0\} \rightarrow R(a_0, b_0)$
  FK :$\{c_0\} \rightarrow (c_0)$

Unit 4.3. Logical Design

Choosing directives for foreign keys

We saw in Unit 1 that foreign keys can be defined with some directives that restore the consistency:

1. NO ACTION: The operation is not allowed if it violates the foreign key constraint.
2. DELETE/UPDATE SET TO NULLS: The value on the referring table is set to nulls.
3. DELETE/UPDATE CASCADE: The row/value on the referring table is deleted/updated.

- Choosing one of them depends on the problem at hand.
- We need to find the one that better fits the meaning of the association.
- Some DBMS do not implement some of these directives

Choosing directives for foreign keys

On UPDATE

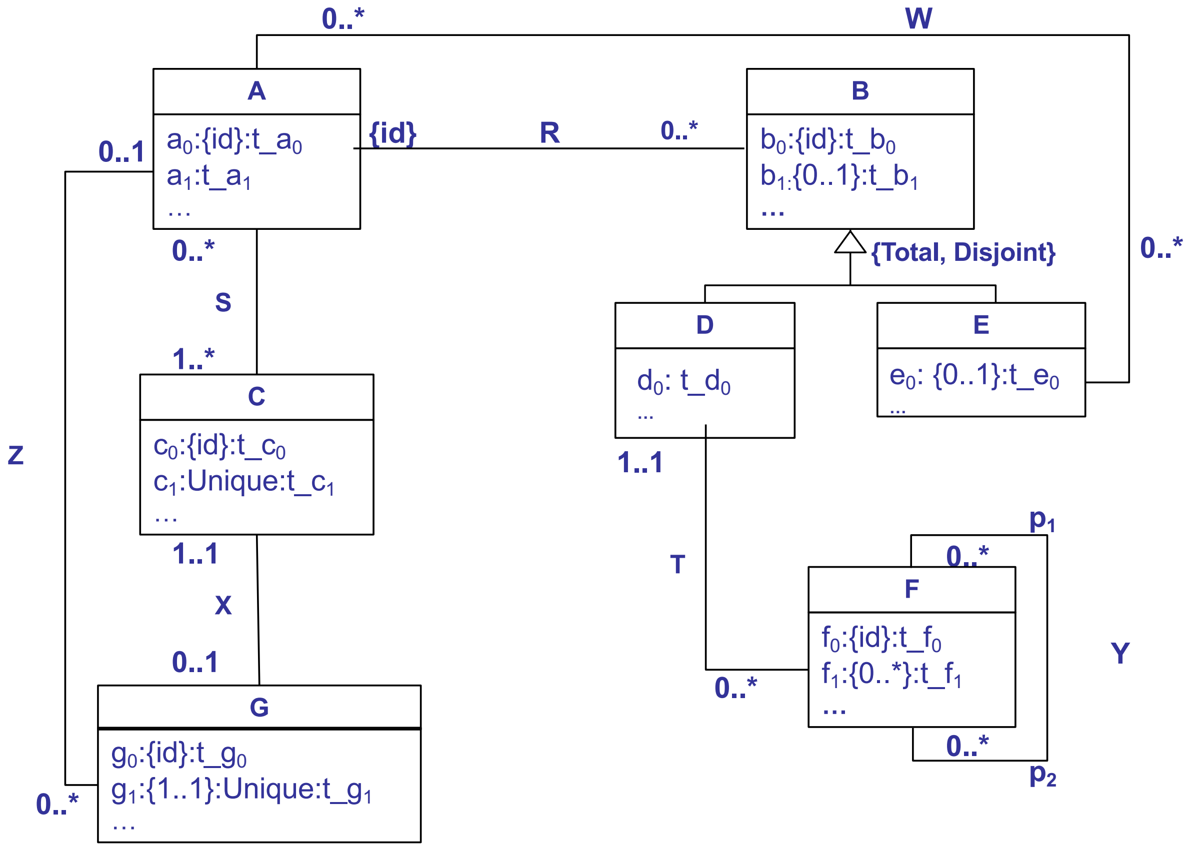
It is recommended, as a general rule, to use “ON UPDATE **CASCADE**”.

On DELETE

- If the value of the referring table has a NNV constraint, then SET TO NULL makes no sense (it is like NO ACTION).
- For multi-valued attributes {0/1…*}, which lead to an extra table, any deletion in the parent table will require a CASCADE deletion in the child table.
- If there is a 1 to 1 correspondence (e.g. existence), then it may be convenient to use **SET TO NULLS** or **CASCADE**, as the NO ACTION case may lead to the impossibility of deletion (without transactions).
- In specializations, SET TO NULLS cannot be done as the subclass depends on the primary key of the superclass.
- In *total* specializations, NO ACTION may be problematic, as the constraint ensuring totality will be violated.

Unit 4.3. Logical Design

Exercise 4



Exercise 4

A(a_0: t_a_0, a_1: t_a_1, …)
PK: {a_0}

B(b_0: t_b_0, b_1: t_b_1, …, a_0: t_a_0)
PK: {a_0, b_0}
FK: {a_0} → A(a_0)

B weak

C(c_0: t_c_0, c_1: t_c_1, …)
PK: {c_0}
UNI: {c_1}

D(a_0: t_a_0, b_0: t_b_0, d_0: t_d_0, …)
PK: {a_0, b_0}
FK: {a_0, b_0} → B(a_0, b_0)

D subclass of B

E(a_0: t_a_0, b_0: t_b_0, e_0: t_e_0, …)
PK: {a_0, b_0}
FK: {a_0, b_0} → B(a_0, b_0)

E subclass of B

F(f_0: t_f_0, …)
PK: {f_0}

f_1 multi-valued

F_f1(f_0: t_f_0, f_1: t_f_1)
PK: {f_0, f_1}
FK: {f_0} → F(f_0)

G(g_0: t_g_0, g_1: t_g_1, …)
PK: {g_0}
UNI: {g_1}
NNV: {g_1}

IC_{Total}: "Every pair a_0, b_0 in *B, must also be in D or E".*

$IC_{Disjoint}$ "A pair a_0, b_0 cannot be in one tuple of D and E".

$A(a_0\text{: } t\_a_0,\ a_1\text{: } t\_a_1,\ \ldots)$
PK: $\{a_0\}$

$B(b_0\text{: } t\_b_0,\ b_1\text{: } t\_b_1,\ \ldots,\ a_0\text{: } t\_a_0)$
FK: $\{a_0, b_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$

$C(c_0\text{: } t\_c_0,\ c_1\text{: } t\_c_1,\ \ldots)$
PK: $\{c_0\}$
UNI: $\{c_1\}$

$D(a_0\text{: } t\_a_0,\ b_0\text{: } t\_b_0,\ d_0\text{: } t\_d_0,\ \ldots)$
PK: $\{a_0, b_0\}$
FK: $\{a_0, b_0\} \rightarrow B(a_0, b_0)$

$E(a_0\text{: } t\_a_0,\ b_0\text{: } t\_b_0,\ e_0\text{: } t\_e_0,\ \ldots)$
PK: $\{a_0, b_0\}$
FK: $\{a_0, b_0\} \rightarrow B(a_0, b_0)$

IC_{Total}: "Every pair a_0, b_0 in *B, must also be in D or E".*

$IC_{Disjoint}$ "A pair a_0, b_0 cannot be in more than one tuple of D or E".

$S(a_0\text{: } t\_a_0,\ c_0\text{: } t\_c_0)$
PK: $\{a_0, c_0\}$
FK: $\{a_0\} \rightarrow A(a_0)$
FK: $\{c_0\} \rightarrow C(c_0)$

IC: "Every a_0 in A must be in S"

$F(f_0\text{: } t\_f_0,\ \ldots,\ a_0\text{: } t\_a_0\ ,\ b_0\text{: } t\_b_0)$
PK: $\{f_0\}$
FK: $\{a_0, b_0\} \rightarrow D(a_0, b_0)$
NNV: $\{a_0, b_0\}$
T

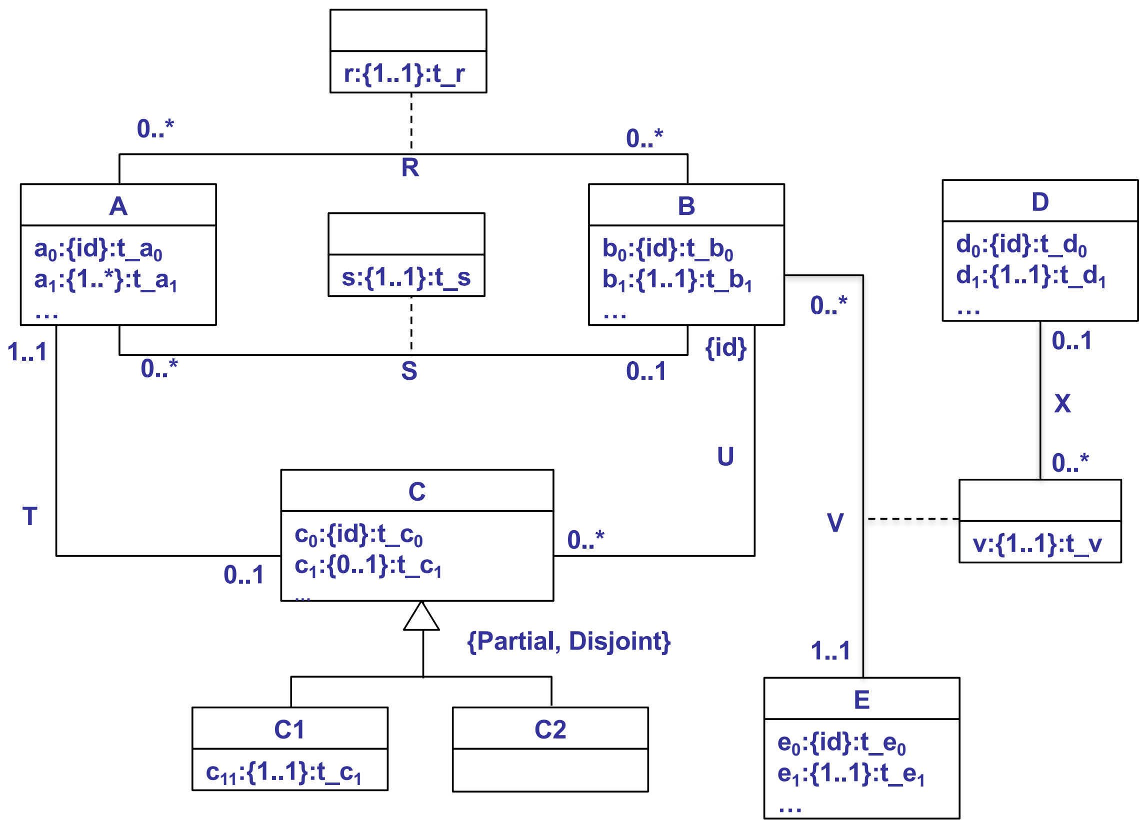
$F\_f1(f_0\text{: } t\_f_0,\ f_1\text{: } t\_f_1)$
PK: $\{f_0, f_1\}$
FK: $\{f_0\} \rightarrow F(f_0)$

$G(g_0\text{: } t\_g_0,\ g_1\text{: } t\_g_1,\ \ldots\ c_0\text{: } t\_c_0,\ a_0\text{: } t\_a_0)$
PK: $\{g_0\}$
UNI: $\{g_1\}$
NNV: $\{g_1\}$
UNI: $\{c_0\}$
NNV: $\{c_0\}$
FK: $\{c_0\} \rightarrow C(c_0)$
FK: $\{a_0\} \rightarrow A(a_0)$
X
Z

$W(a_{0A}\text{: } t\_a_0,\ a_{0B}\text{: } t\_a_0,\ b_0\text{: } t\_b_0)$
PK: $\{a_{0A}, a_{0B}, b_0\}$
FK: $\{a_{0A}\} \rightarrow A(a_0)$
FK: $\{a_{0B}, b_0\} \rightarrow E(a_0, b_0)$

$Y(f0p_1\text{: } t\_f_0,\ f0p_2\text{: } t\_f_0)$
PK: $\{f0p_1, f0p_2\}$
FK: $\{f0p_1\} \rightarrow F(f_0)$
FK: $\{f0p_2\} \rightarrow F(f_0)$

Exercise 5



Exercise 5 Class transformation

A (a_0:t_a_0, …)
PK:{a_0}

A1 (a_0: t_a_0, a_1: t_a_1)
PK:{a_0, a_1}
CA:{a_0} → A

IC1: Every value a_0 in A must be in A1

D (d_0:t_d_0, d_1:t_d_1 …)
PK:{d_0}
NNV:{d_1}

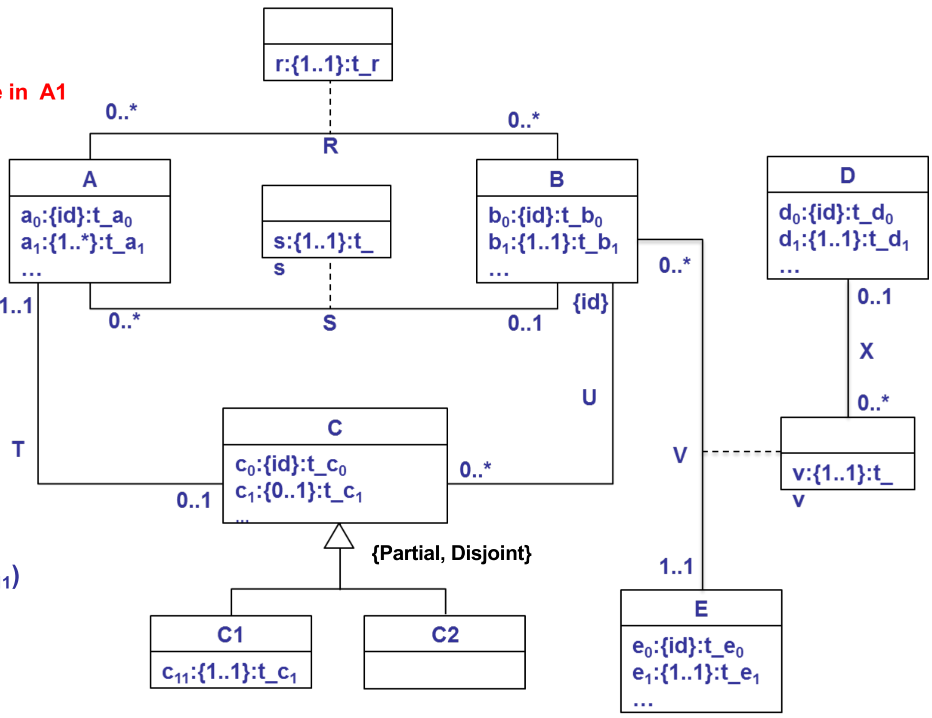
E (e_0:t_e_0, e_1:t_e_1 …)
PK:{e_0}
NNV:{e_1}

B (b_0:t_b_0, b_1:t_b_1, …,)
PK:{b_0}
NNV:{b_1}

C(c_0: t_c_0, c_1: t_c_1, b_0: t_b_0)
PK:{c_0, b_0}
CA:{b_0} → B

C1(c_0: t_c_0, b_0: t_b_0, c_{11}: t_c_{11})
PK:{c_0, b_0}
CA:{c_0, b_0} → C
NNV:{c_{11}}

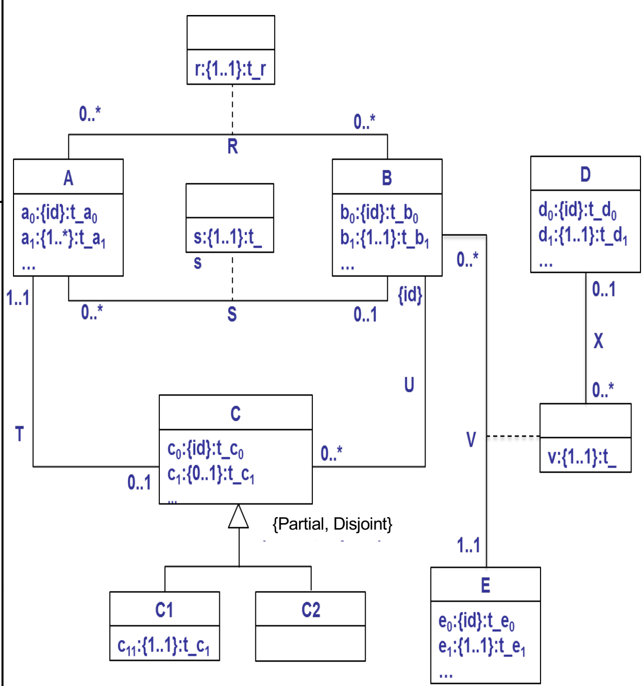
C2(c_0: t_c_0, b_0: t_b_0)
PK:{c_0, b_0}
CA:{c_0, b_0} → C



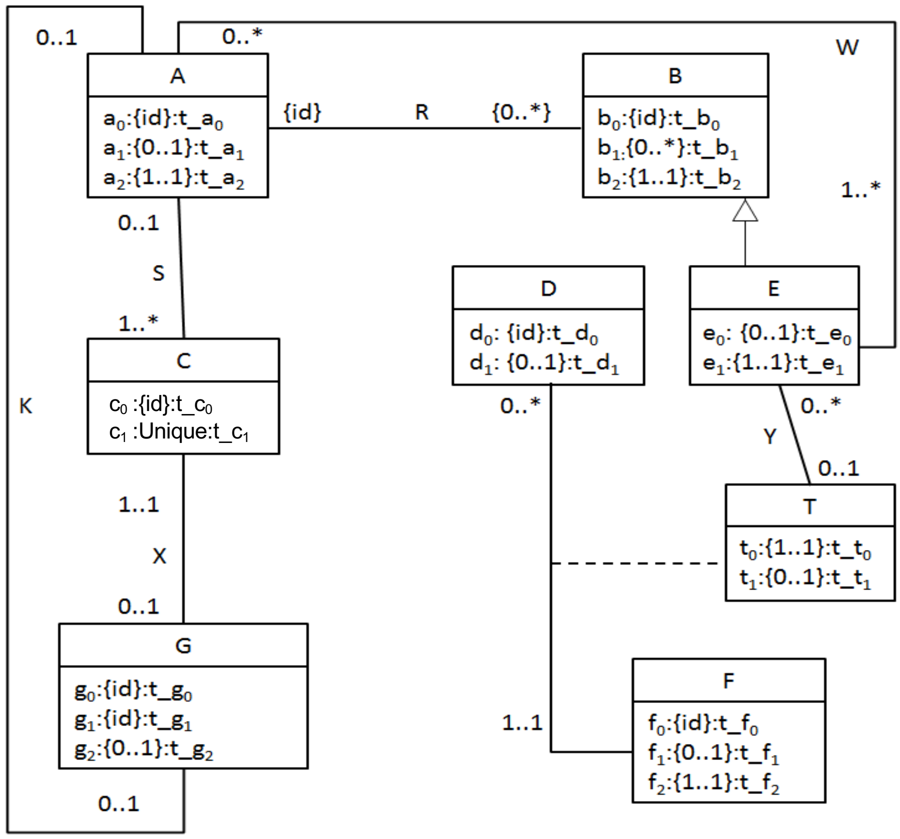
IC2-Disjoint: A value {c_0, b_0} in C cannot appear in C1 and C2 at the same time.

Exercise 5 Association transformation

A (a_0:t_a_0, …) PK:{a_0} **A1** (a_0: t_a_0, a_1: t_a_1) PK:{a_0, a_1} FK:{a_0}→A IC1: Every value a_0 in A must be in A1	**B** (b_0:t_b_0, b_1:t_b_1, …, e_0:t_e_0, v:t_v, d_0:t_d_0) PK:{b_0} NNV:{b_1} FK:{e_0}→ E NNV:{e_0} NNV:{v} FK:{d_0} → D V X
D (d_0:t_d_0, d_1:t_d_1 …) PK:{d_0} NNV:{d_1} **E** (e_0:t_e_0, e_1:t_e_1 …) PK:{e_0} NNV:{e_1} **R** (a_0: t_a_0, b_0: t_b_0, r:t_r) PK:{a_0, b_0} FK:{a_0}→A FK:{b_0}→B NNV:{r}	**C** (c_0: t_c_0, c_1: t_c_1, a_0: t_a_0, b_0: t_b_0) PK:{c_0, b_0} FK:{a_0} → A FK:{b_0}→B NNV:{a_0} UNI: {a_0} T **C1** (c_0: t_c_0, b_0: t_b_0, c_{11}: t_c_{11}) PK:{c_0, b_0} FK:{ c_0, b_0} → C NNV:{c_{11}}
S (a_0: t_a_0, b_0: t_b_0, s:t_s) PK:{a_0 } FK:{a_0} → A FK:{b_0} → B NNV:{s} NNV:{b_0}	**C2** (c_0: t_c_0, b_0: t_b_0) PK:{c_0, b_0} FK:{ c_0, b_0} → C IC-Disjoint: A value of {c_0, b_0} in C cannot appear in C1 and C2 at the same time.



Exercise 6



Exercise 6

A(a_0:t_a_0, a_1:t_a_1, a_2:t_a_2)
PK: {a_0}
NNV: {a_2}
B(b_0:t_b_0, b_2:t_b_2, a_0:t_a_0)
PK: {b_0, a_0}
FK: {a_0}$\rightarrow$A
NNV: {b_2}
B weak
b_1 multi-valued
B1(a_0:t_a_0, b_0:t_b_0, b_1:t_b_1)
PK: {a_0, b_0, b_1}
FK: {a_0, b_0}$\rightarrow$B
C(c_0:t_c_0, c_1:t_c_1)
PK: {c_0}
UNI: {c_1}
G(g_0:t_g_0, g_1:t_g_1, g_2:t_g_2)
PK: {g_0, g_1}

E specializes B

E(a_0:t_a_0, b_0:t_b_0, e_0:t_e_0, e_1:t_e_1)
PK: {a_0, b_0}
FK: {a_0, b_0}$\rightarrow$B
NNV: {e_1}
F(f_0:t_f_0, f_1:t_f_1, f_2:t_f_2)
PK: {f_0}
NNV: {f_2}
D(d_0:t_d_0, d_1:t_d_1)
PK: {d_0}

A (a_0:t_a_0, a_1:t_a_1, a_2:t_a_2)
PK: {a_0}
NNV: {a_2}

B (b_0:t_b_0, a_0:t_a_0, b_2:t_b_2)
PK: {b_0, a_0}
NNV: {b_2}
FK: {a_0}→A

B1 (a_0:t_a_0, b_0:t_b_0, b_1:t_b_1)
PK: {a_0, b_0, b_1}
FK: {a_0, b_0}→B

C (c_0:t_c_0, c_1:t_c_1, a_0:t_a_0)
S
PK: {c_0}
UNI: {c_1}
FK: {a_0}→A

G (g_0:t_g_0, g_1:t_g_1, g_2:t_g_2, c_0:t_c_0, a_0:t_a_0)
PK: {g_0, g_1}
X
FK: {c_0}→C
NNV: {c_0}
UNI: {c_0}
K
FK: {a_0}→A
UNI: {a_0}

E(a_0:t_a_0, b_0:t_b_0, e_0:t_e_0, e_1:t_e_1, d_0:t_d_0)
PK: {a_0, b_0}
FK: {a_0, b_0}→B
NNV: {e_1}
Y
FK: {d_0}→D

F (f_0:t_f_0, f_1:t_f_1, f_2:t_f_2)
PK: {f_0}
NNV: {f_2}

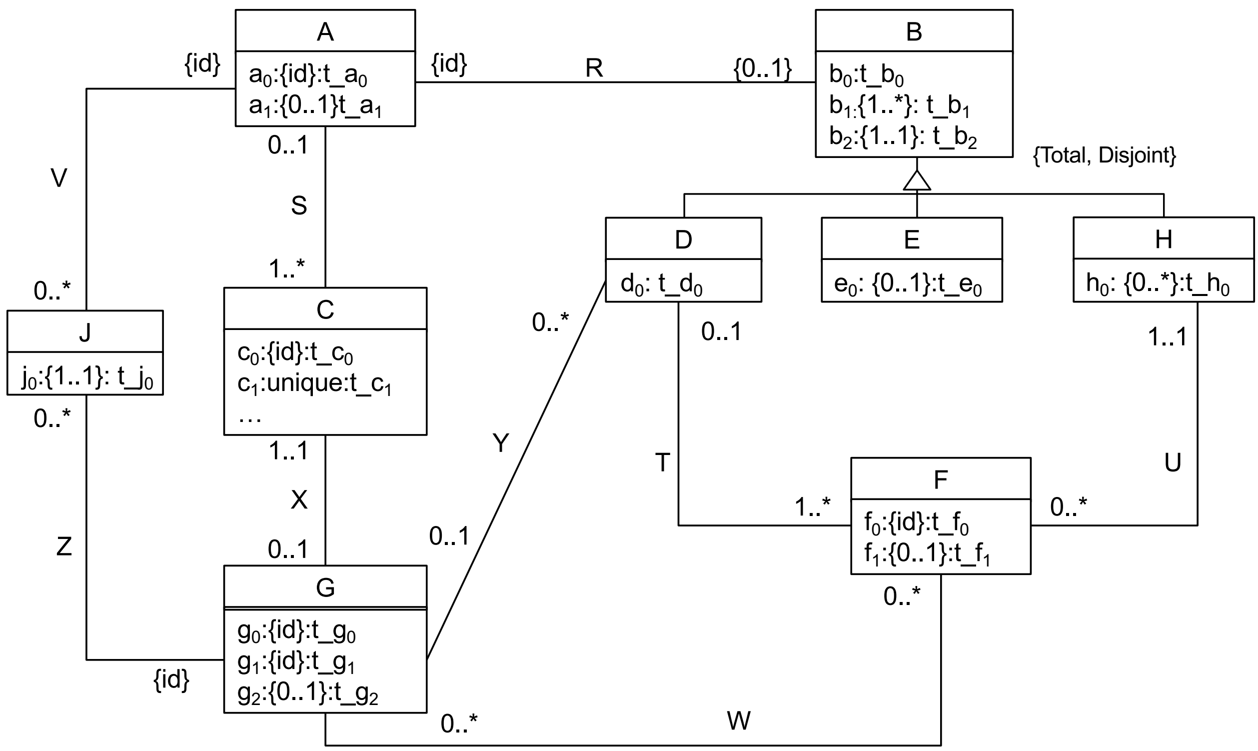
D (d_0:t_d_0, d_1:t_d_1, f_0:t_f_0, t_0:t_t_0, t_1:t_t_1)
PK: {d_0}
T
FK: {f_0}→F
NNV: {f_0}
NNV: {t_0}

W (a_0:t_a_0, **a_0_E**:t_a_0, **b_0_E**:t_b_0)
PK: {a_0_E, b_0_E, a_0}
FK: {a_0}→A
FK: {**a_0_E, b_0_E**}→E

Integrity Constraints:

1. Every value of a_0 of A must appear in a_0 of W at least once.
2. Every value of a_0 of A must appear in a_0 of C at least once.

Exercise 7



Logical design Exercise 7

B_1 multi-valued

$A(a_0: t\_a_0,\ a_1: t\_a_1)$
PK: $\{a_0\}$
$B(b_0: t\_b_0,\ b_2: t\_b_2,\ a_0: t\_a_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to A$
NNV: $\{b_2\}$

B weak

$B\_b1(a_0: t\_a_0,\ b_1: t\_b_1)$
PK: $\{a_0, b_1\}$
FK: $\{a_0\} \to B(a_0)$
RI: Every a_0 in B has at least one a_0 in B_b1

$F(f_0: t\_f_0,\ f_1: t\_f_1)$
PK: $\{f_0\}$

$G(g_0: t\_g_0,\ g_1: t\_g_1,\ g_2: t\_g_2)$
PK: $\{g_0, g_1\}$

$D(a_0: t\_a_0,\ d_0: t\_d_0,\ \ldots)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$

D specializes B

$C(c_0: t\_c_0,\ c_1: t\_c_1,\ \ldots)$
PK: $\{c_0\}$
Uni $\{c_1\}$

$E(a_0: t\_a_0,\ e_0: t\_e_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$

E specializes B

H specializesB

$H(a_0: t\_a_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$

$H\_h0(a_0: t\_a_0,\ h_0: t\_h_0)$
PK: $\{a_0, h_0\}$
FK: $\{a_0\} \to H(a_0)$

$J(j_0: t\_j_0,\ a_0: t\_a_0,\ g_0: t\_g_0,\ g_1: t\_g_1)$
PK: $\{a_0, g_0, g_1\}$
FK: $\{a_0\} \to A$
FK: $\{g_0, g_1\} \to G(g_0, g_1)$
NNV: $\{j_0\}$

J weak

RI_{Total}: ""Every a_0 in *B*, must also be in a_0 of D, E, or H".
$RI_{Disjoint}$: "A value a_0,in B cannot be in more than one tuple of D, E, or H".

h_0 multi-valued

$A(a_0: t\_a_0,\ a_1: t\_a_1)$
PK: $\{a_0\}$
$B(b_0: t\_b_0,\ b_2: t\_b_2,\ a_0: t\_a_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to A$
NNV: $\{b_2\}$

$D(a_0: t\_a_0,\ d_0: t\_d_0,\ g_0: t\_g_0,\ g_1: t\_g_1\ \ldots)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$
FK: $\{g_0, g_1\} \to G(g_0, g_1)$

Y

$E(a_0: t\_a_0,\ e_0: t\_e_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$
$H(a_0: t\_a_0)$
PK: $\{a_0\}$
FK: $\{a_0\} \to B(a_0)$

RI_{Total}: ""Every a_0 in *B*, must also be in a_0 of D, E, or H".
$RI_{Disjoint}$: "A value a_0,in B cannot be in more than one tuple of D, E, or H".

$H\_h0(a_0: t\_a_0,\ h_0: t\_h_0)$
PK: $\{a_0, h_0\}$
FK: $\{a_0\} \to H(a_0)$

$F(f_0: t\_f_0,\ f_1: t\_f_1,\ a_0D: t\_a_0,\ a_0H: t\_a_0)$
PK: $\{f_0\}$
FK: $\{a_0D\} \to D(a_0)$
RI: "Every value a_0 in D must be in F"
PK: $\{a_0H\} \to H(a_0)$ NNV: $\{a_0H\}$

T

U

$G(g_0: t\_g_0,\ g_1: t\_g_1,\ g_2: t\_g_2,\ c_0: t\_c_0)$
PK: $\{g_0, g_1\}$
UNI: $\{c_0\}$ NNV: $\{c_0\}$
FK: $\{c_0\} \to C(c_0)$

X

$C(c_0: t\_c_0,\ c_1: t\_c_1,\ \ldots,\ a_0: t\_a_0)$
PK: $\{c_0\}$
UNI: $\{c_1\}$
FK: $\{a_0\} \to A$
RI: Every value a_0 in A must be in C"

S

$J(j_0: t\_j_0,\ a_0: t\_a_0,\ g_0: t\_g_0,\ g_1: t\_g_1)$
PK: $\{a_0, g_0, g_1\}$
FK: $\{a_0\} \to A$
FK: $\{g_0, g_1\} \to G(g_0, g_1)$
NNV: $\{j_0\}$
$W(g_0: t\_g_0,\ g_1: t\_g_1,\ f_0: t\_f_0)$
PK: $\{g_0, g_1, f_0\}$
FK: $\{g_0, g_1\} \to G(g_0, g_1)$
FK: $\{f_0\} \to F$

$B\_b1(a_0: t\_a_0,\ b_1: t\_b_1)$
PK: $\{a_0, b_1\}$
FK: $\{a_0\} \to B(a_0)$
RI: Every a_0 in B has at least one a_0 in B_b1

Unit 4.3. Logical Design

6. Introduction to Database normalization

Normalization

Technique for producing a set of relations with desirable properties dividing some relations into other smaller ones.

Some problems solved by normalization

- Some attributes might be redundant, because of **functional dependencies**, which may be direct or transitive.
- Bad choices of the primary key among the **candidate keys**

There are several **normal forms**: 1NF, 2NF, 3NF, BCNF, 4NF, 5NF, …
but we will only see 1NF, 2NF and 3NF.

Functional dependencies

A **functional dependency** (FD) between two sets of attributes X and Y of a relation R, where X <> Y, denoted X→Y (*"X determines Y", or "Y functionally depend on X"*) specifies the following constraint in the real world:

Given two tuples t1 and t2 of R, if the value of X for t1 and t2 are equal then the values of Y for t1 and t2 are also equal (each value of X is associated with exactly one value of Y).

Example 1:

R (*name*: char, *street*: char, *zip_code*: char, *city*: char)

zip_code → city

If we know the *zip_code*, we can infer the city.

This redundancy may lead to inconsistences

Examples of functional dependencies

Example 2:

Wrote (*ssn*: char(4), *name*: varchar(50), *ISBN*: char(25), *title*: varchar(50), *euros*: float)

PK: {ssn, ISBN}

meaning that the writer with id *"ssn"* and name *"name"* has written a book with *"ISBN"* and *"title"* and has received *"euros"* as royalties.

Some functional dependencies:

{ ssn } → { name }
{ ssn, ISBN } → { name }
{ ssn, title } → { name }
{ ssn, euros } → { name }
{ ssn, ISBN, title } → { name }
{ ssn, ISBN, euros } → { name }
{ ssn, ISBN, title, euros } → { name }
{ ISBN } → { title }
{ ISBN, ssn } → { title }
{ ISBN, euros } → { title }
{ ISBN, name } → { title }
{ ISBN, ssn, name } → { title }
{ ISBN, ssn, euros } → { title }
{ ISBN, name, euros } → { title }
{ ssn, ISBN } → { euros }
{ ssn, ISBN, name } → { euros }
{ ssn, ISBN, title } → { euros }
{ ssn, ISBN, name, title } → { euros }
{ ssn, ISBN } → { title, name }
…

Full functional dependency

An Functional Dependency X →Y is a **full functional dependency (FFD)** if removal of any attribute A_i from X means that the dependency does not hold any more. That is, $\forall A_i$ / $A_i \in X$, Y doesn't functionally depend on (X – $\{A_i\}$).

Example 2:

Wrote (*ssn*: char(4), *name*: varchar(50), *ISBN*: char(25), *title*: varchar(50), *euros*: float)

PK: {ssn, ISBN}

Set of **full functional dependencies**:

{ ssn } → { name }
{ ISBN } → { title }
{ ssn, ISBN } → { euros }

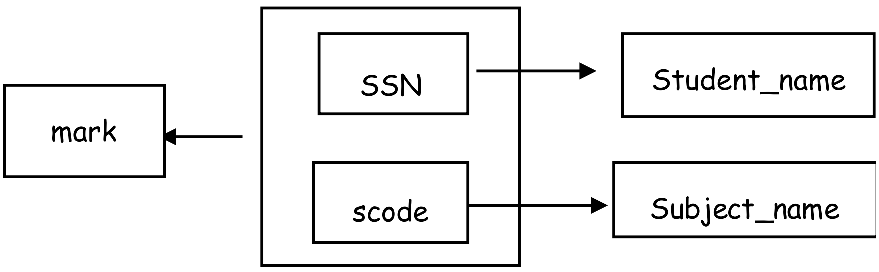
Functional dependencies Diagram

A set of Functional Dependencies for a data model can be documented in a **Functional Dependency Diagram**

In a Functional Dependency Diagram each attribute is shown in a rectangle with an arrow indicating the direction of the dependency.

We are going to represent only full functional dependencies.

Example 3:



Key of a relation

Key of a relation

Set of attributes that is PK or has uniqueness constraint.

For every key in a relation, every attribute subset depends on that key.

Prime attribute

Any attribute that belongs to some key of R.

Example 2:

Wrote (*ssn*: char(4), *name*: varchar(50), *ISBN*: char(25), *title*: varchar(50), *euros*: float)
PK: {ssn, ISBN}

{ssn, ISBN} → { name }
{ssn, ISBN} → { title }
{ssn, ISBN} → { euros }
{ssn, ISBN} → { title, name }

1st Normal Form

A relation is in 1NF if all its attributes are atomic (scalar, i.e. simple and indivisible).

Problem of relations which are not in 1NF:

- We must use operators for complex data: lists, sets, records…

Example 4:

Provider PK: {vcod}

Set → telephone; Record → address

vcod	name	telephone	address
V1	Pepe	(96 3233258, 964 523844, 979 568987, 987 456123)	Paz 7, Valencia
V2	Juan	(96 3852741, 910 147258)	Eolo 3, Castellón
V3	Eva	(987 456 312)	F. Lorca 2, Utiel

1st NF Transformation: Multi-valued attribute

R has an attribute which is a set of values:

⇩

1. Remove the attribute from the relation and define a new relation with the attribute and the primary key of R.

2. Analyze the functional dependencies of the new relation to define its primary key.

1st NF. Example 4

Supplier

vcod	name	telephone	address
V1	Pepe	(96 3233258, 964 523844, 979 568987, 987 456123)	Paz 7, Valencia
V2	Juan	(96 3852741, 910 147258)	Eolo 3, Castellón
V3	Eva	(987 456 312)	F. Lorca 2, Utiel

vcod	name	address
V1	Pepe	Paz 7, Valencia
V2	Juan	Eolo 3, Castellón
V3	Eva	F. Lorca 2, Utiel

vcod	telephone
V1	96 3233258
V2	96 3852741
V3	987 456 312
V1	964 523844
V1	979 568987
V1	987 456123
V2	910 147258

PK?

1st NF. Example 4

Supplier(vcod, name, telephone, address)
PK: {vcod}

Supplier(vcod, name, address)
PK: {vcod}

Phonebook (vcod, telephone)
PK: {telephone} ← If a telephone cannot be shared
FK: {vcod} → Supplier
NNV: {vcod}

1st NF. Example 4

Supplier (vcod, name, telephone, address)
PK: {vcod}

Supplier (vcod, name, address)
PK: {vcod}

Phonebook (vcod, telephone)
PK: {telephone, vcod} ← If a telephone can be shared
FK: {vcod} → Supplier

1st NF Transformation: Composite attribute

R has a composite attribute (a record).

⇩

remove the attribute and add a new attribute for each member of the composite attribute

Example 1

vcod	name	address
V1	Pepe	Paz 7, Valencia
V2	Juan	Eolo 3, Castellón
V3	Eva	F. Lorca 2, Utiel

vcod	name	street	number	city
V1	Pepe	Paz	7	Valencia
V2	Juan	Eolo	3	Castellón
V3	Eva	F. Lorca	2	Utiel

1st NF. Example 4

Supplier (vcod, name, telephone, address)
PK: {vcod}

Supplier (vcod, name, street, number, city)
PK: {vcod}

Phonebook (vcod, telephone)
PK: {telephone, vcod}
FK: {vcod} → Supplier

2nd NF

A relation R is in 2NF if it is in 1NF and all non-prime attributes have a full functional dependency on all the keys of *R*.

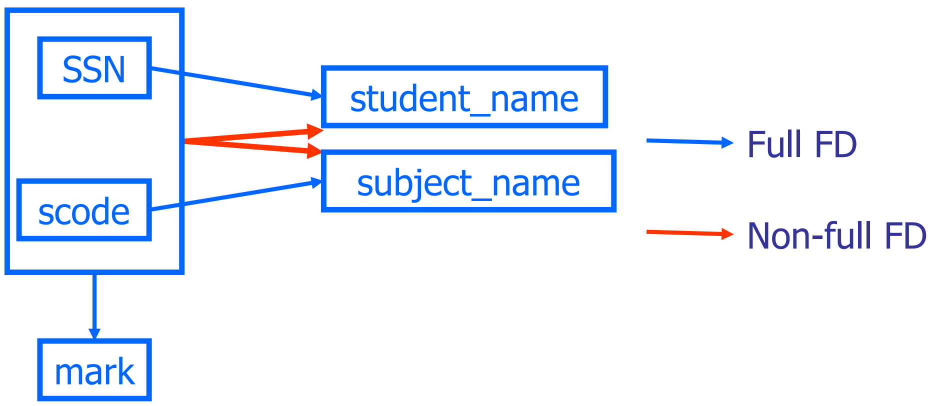
Problems of relations which are not in 2NF:

- Redundancy.
- It is more difficult to insert, delete, and update

2nd NF. Example 3

PK: {SSN, subject_code}

SSN	student_name	scode	subject_name	mark
1	Pepe	DBD	Diseño de BD	6
1	Pepe	BDA	Bases de Datos	7
2	Juana	DBD	Diseño de BD	7
2	Juana	BDA	Bases de Datos	5



2nd NF Transformation

The primary key has more than one attribute and there is some non-prime attribute which does not fully functionally depend on the primary key

⇩

Divide the relation in several relations to remove the not fully functional dependency

2nd NF. Example 3

SSN	student_name	scode	subject_name	mark
1	Pepe	DBD	Diseño de BD	6
1	Pepe	BDA	Bases de Datos	7
2	Juana	DBD	Diseño de BD	7
2	Juana	BDA	Bases de Datos	5

SSN	student_name
1	Pepe
2	Juana

scode	subject_name
DBD	Diseño de BD
BDA	Bases de Datos

SSN	scode	mark
1	DBD	6
2	BDA	7
1	DBD	7
2	BDA	5

2nd NF. Example 3

Exam (SSN, scode, student_name, subject_name, mark)

PK: {SSN, scode}

Student (SSN, student_name)

PK: {SSN}

Subject (scode, subject_name)

PK: {scode}

Exam (SSN, scode, mark)

PK: {SSN, scode}

FK: {SSN} → Student

FK: {scode} → Subject

2nd NF. Example 2

Wrote (*ssn*: char(4), *name*: varchar(50), *ISBN*: char(25), *title*: varchar(50), *euros*: float)

PK: {ssn, ISBN}

Partial dependencies on the PK:

{ssn} → {name}

{ISBN} → {title}

2nd NF:

Wrote (*ssn*: char(4), *ISBN*: char(25), *euros*: float)

PK: {ssn, ISBN}

Author (*ssn*: char(4), *name*: varchar(50))

PK: {ssn}

Book (*ISBN*: char(25), *title*: varchar(50))

PK: {ISBN}

3rd NF

A relation R is in 3NF if it is in 2NF and there are no functional dependencies between any non-prime attribute.

Problems of relations which are not in 3NF:

- Redundancy.
- It is more difficult to insert, delete, and update

Transitive dependency

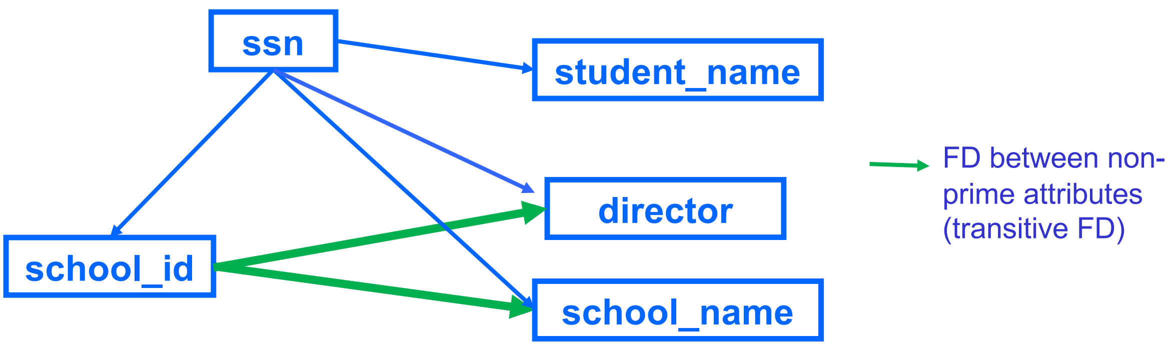
A = {A_1, A_2, …, A_n} is the set of attributes of R,

If {A_i} -> {A_j} and {A_j} -> {A_k} then {A_i} -> {A_k} is a **transitive dependency** (A_k is transitively dependent on A_i via A_j)

3rd NF. Example 5

ssn	student_name	school_id	school_name	director
1	Olga	ETSINF	Escuela de Informática	Pepe
2	Juana	ETSINF	Escuela de Informática	Pepe
3	Ana	ED	Escuela de Diseño	Eva
4	Juan	ED	Facultad de Diseño	Eva

PK: {ssn}



3rd NF Transformation

If there is at least a pair of non-prime attributes which are dependent

⇩

Remove the dependent attribute and create a new table with it and the determinant attribute. The PK of the new table will be de determinant attribute

3rd NF. Example 5

Student

PK: {ssn}

snn	student_name	school_id	school_name	director
1	Olga	ETSINF	Escuela de Informática	Pepe
2	Juana	ETSINF	Escuela de Diseño	Pepe
3	Ana	ED	Escuela de Diseño	Eva
4	Juan	ED	Escuela de Diseño	Eva

Student

snn	student_name	school_id
1	Olga	ED
2	Juana	ETSINF
3	Ana	ED
4	Juan	ED

School

school_id	school_name	director
ED	Escuela de Diseño	Pepe
ED	Escuela de Diseño	Eva

3rd NF. Example 5

Student (ssn, student_name, school_id, school_name, director)
PK: {ssn}

Student (ssn, student_name, school_id)
PK: {ssn}
FK: {school_id} → School

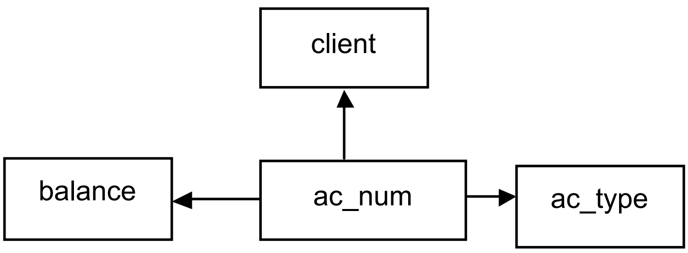
School (school_id, school_name, director)
PK: {school_id}

3rd NF. Example 6

Account (client: integer**, ac_num**: varchar(15), **ac_type**: varchar(10),
balance: real**)**
PK: {ac_num}
NNV: {client, ac_type, balance}

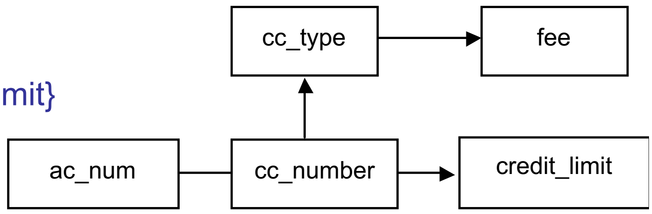
Ccard (ac_num: varchar(15), **cc_number**: varchar(15), **cc_type**: varchar(10),
fee: real, **credit_limit**: real**)**
PK: {cc_num}
FK: {ac_num} → Account
NNV: { ac_num, cc_type, fee, credit_limit}

In the **Account** table all the attributes functionally depend on the PK and there is no dependencies between non-prime attributes, so it is in **3NF**:



3ª Forma normal. Ejemplo

Ccard (ac_num: varchar(15), **cc_number**: varchar(15), **cc_type**: varchar(10), **fee**: real, **credit_limit**: real**)**
PK: {cc_number}
FK: {ac_num} → Account
NNV: { ac_num, cc_type, fee, credit_limit}



It is not in 3NF, because there is a FD between non-prime attributes

Ccard (ac_num: varchar(15), **cc_number**: varchar(15), **cc_type**: varchar(10), **fee**: real**)**
PK: {cc_number}
FK: {ac_num} → Account
NNV: { ac_num, cc_type, credit_limit}
FK: {cc_type} → Ccard_type

Ccard_type (cc_type: varchar(10), **fee**: real**)**
PK: {cc_type}
NNV: {fee}

Exercise N1

Consider the following relational schema:

R (**A**: integer, **B**: varchar, **C**: integer, **D**: varchar, **E**: varchar, **F**: varchar, **G**: varchar)
PK: {A, B}
NNV: {C, D, E, F, G}

and the following functional dependencies

{G} → {E} {G} → {F} {A} → {D} {A} → {G}

Transform the schema to obtain a set of relations in 3NF

Exercise N1

R (**A**: integer, **B**: varchar, **C**: varchar, **D**: varchar, **E**: varchar, **F**: varchar, **G**: varchar)

PK: {A, B}
NNV: {C, D, E, F, G}

{G} → {E} {G} → {F} {A} → {D} {A} → {G}

Transitive dependencies:
If {A} -> {G} and {G} -> {E} {A} -> {E}
If {A} -> {G} and {G} -> {F} {A} -> {F}

2FN

{G} → {E} {G} → {F} **{A} → {D}** **{A} → {G}** **{A} -> {E}** **{A} -> {F}**

R1(A: integer, B: varchar, C: integer)
PK: {A, B}
FK:{A} -> R21
NNV: {C}

R21 (A: integer, D: varchar, G: varchar, E: varchar, F: varchar)
PK: {A}
NNV: {D,G, E, F}

All the values of {A} in R21 are also in R1.

Exercise N1

R21 (**A**: integer, **D**: varchar, **G**: varchar, **E**: varchar, **F**: varchar)
PK: {A}
NNV: {D,G, E, F}

All the values of {A} in R21 are also in R1.

R1(**A**: integer, **B**: varchar, **C**: integer)
PK: {A, B}
FK:{A} -> R21
NNV: {C}

3FN

{G} → {E} **{G} → {F}** {A} → {D} {A} → {G} {A} -> {E} {A} -> {F}

R1(*A*: integer, *B*: varchar, *C*: varchar)
PK: {A, B}
FK:{A} -> R21
NNV: {C}

R21 (*A*: integer, *D*: varchar, *G*: varchar)
PK: {A}
FK:{G} -> R22
NNV: {D,G}

R31 (*G*: integer, *E*: varchar, *F*: varchar)
PK: {G}
NNV: {E, F}

All the values of {A} in R21 are also in R1.

All the values of {G} in R31 are also in R21.

Exercise N2

Consider the following relational schema:

R (**A**: integer, **B**: varchar, **C**: integer, **D**: varchar, **E**: varchar, **F**: varchar,
G: varchar, **H**: varchar)
PK: {A, B}
NNV: {C, D, E, F, G, H}

From the dependencies shown below, transform the relation to a set of relations in third normal form (3NF).

{A} → {C}   {B} → {F}   {F} → {G}   {F} → {H}

Exercise N2 2FN

R (**A**: integer, **B**: varchar, **C**: integer, **D**: varchar, **E**: varchar, **F**: varchar,
G: varchar, **H**: varchar)
PK: {A, B}   NNV: {C, D, E, F, G, H}

{A} → {C}   {B} → {F}   {F} → {G}   {F} → {H}

Transitive dependencies: If {B} -> {F} and {F} -> {G} {B} -> {G}
If {B} -> {F} and {F} -> {H} {B} -> {H}

{F} → {G}   {F} → {H}   **{A} → {C}**   **{B} → {F}**   **{B} -> {G}**   **{B} -> {H}**

R1(A: integer, B: varchar, D: varchar, E: varchar)
PK: {A, B}   NNV: {D, E}
FK:{A} -> R21   FK:{B} -> R22

R21 (A: integer, C: integer)
PK: {A}   NNV: {C}

R22 (B: varchar, F: varchar, G: varchar, H: varchar)
PK: {B}   NNV: {F,G, H}

All the values of {A} in R21 are also in R1.
All the values of {B} in R22 are also in R1.

Exercise N2

3FN

R1(**A**: integer, **B**: varchar, **D**: varchar, E: varchar)
PK: {A, B} NNV: {D, E}
FK:{A} -> R21 FK:{A} -> R21

R21 (**A**: integer, **C**: integer)
PK: {A} NNV: {C}

R22 (**B**: varchar, **F**: varchar, **G**: varchar, **H**: varchar)
PK: {B} NNV: {F,G, H}

All the values of {A} in R21 are also in R1.
All the values of {B} in R22 are also in R1.

{F} → {G} **{F} → {H}** {A} → {C} {B} → {F} {B} -> {G} {B} -> {H}

R1(A: integer, B: varchar, D: varchar, E: varchar)
PK: {A, B} NNV: {D, E}
FK: {A} -> R21 FK: {B} -> R22

R21 (A: integer, C: integer)
PK: {A} NNV: {C}

R22 (B: varchar, F: varchar)
PK: {B} NNV: {F }
FK: {F} -> R23

R31 (F: varchar, G: varchar, H: varchar)
PK: {F} NNV: {G, H}

All the values of {A} in R21 are also in R1.
All the values of {B} in R22 are also in R1.
All the values of {F} in R31 are also in R22.

Exercise N3

Consider the following relational schema:

R (**A**: char, **B**: int, **C**: int, **D**: char, **E**: int, **F**: int, **G**: char, **H**: int)
PK: {A, B, C}
NNV: {D, E, F, G, H}

From the dependencies shown below, transform the relation to a set of relations in third normal form (3NF).

{A,C} → {E} {B} → {D} {B} → {G} {E} → {H} {D} → {F}

Exercise N3 — 2FN

R (**A**: char, **B**: int, **C**: int, **D**: char, **E**: int, **F**: int, **G**: char, **H**: int)
PK: {A, B, C} NNV: {D, E, F, G, H}

{A,C} → {E} {B} → {D} {B} → {G} {E} → {H} {D} → {F}

Transitive dependencies: If {A,C} -> {E} and {E} -> {H} {A,C} -> {H}
If {B} -> {D} and {D} -> {F} {B} -> {F}

{A,C} → {E} {A,C} → {H} {B} → {D} {B} → {G} {B} → {F} {E} → {H} {D} → {F}

R1 (A: char, B: int, C: int)
PK: {A, B, C}
FK:{A,C} → R21(A,C) FK:{B} → R22(B)

R21 (A: int, C: int, E: int, H: int)
PK: {A, C} NNV: {E, H}

R22 (B: char, D: char, F: int, G: char)
PK: {B} NNV: {D, F, G}

All the values of pairs {A.C} in R21 are also in R1.
All the values of {B} in R22 are also in R1.

Exercise N3 — 3FN

R1 (**A**: char, **B**: int, **C**: int)
PK: {A, B, C}
FK:{A,C} → R21(A,C) FK:{B} → R22(B)

R21 (**A**: int, **C**: int, **E**: int, **H**: int)
PK: {A, C} NNV: {E, H}

R22 (**B**: char, **D**: char, **F**: int, **G**: char)
PK: {B} NNV: {D, F, G}

- All the values of pairs {A.C} in R21 are also in R1.
- All the values of {B} in R22 are also in R1

{A,C} → {E} {A,C} → {H} {B} → {D} {B} → {G} {B} → {F} **{E} → {H} {D} → {F}**

R1 (A: char, B: int, C: int,)
PK: {A, B, C}
FK: {A, C} -> R21(A,C) FK: {B} -> R22(B)

R21 (A: int, C: int, E: int)
PK: {A, C} NNV: {E} FK: {E} -> R31(E)

R22 (B: char, D: char, G: char)
PK: {B} NNV: {D G }
FK: {D} -> R32(D)

R31 (E: int, H: int)
PK: {E} NNV: {H}

R32 (D: char, F: int)
PK: {D} NNV: {F }

All the values of {A} in R21 are also in R1
All the values of {B} in R22 are also in R1
All the values of {E} in R31 are also in R21
All the values of {D} in R32 are also in R22.