UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Unit 4:
Relational Database Design

4.1. Database Design Fundamentals

4.2. Conceptual Design

4.3. Logical Design

Unit 4.3. Logical Design

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

1st STAGE: Analysis

Inquiry

| Information Requirements | Processing Requirements |

2nd STAGE: Design

Conceptual Design

Conceptual design

Statics   Dynamics

Logical Design

Logical schema   Transaction schemas

Physical Design

Physical schema   Program development

3rd STAGE: Deployment

Database load   User Training

4
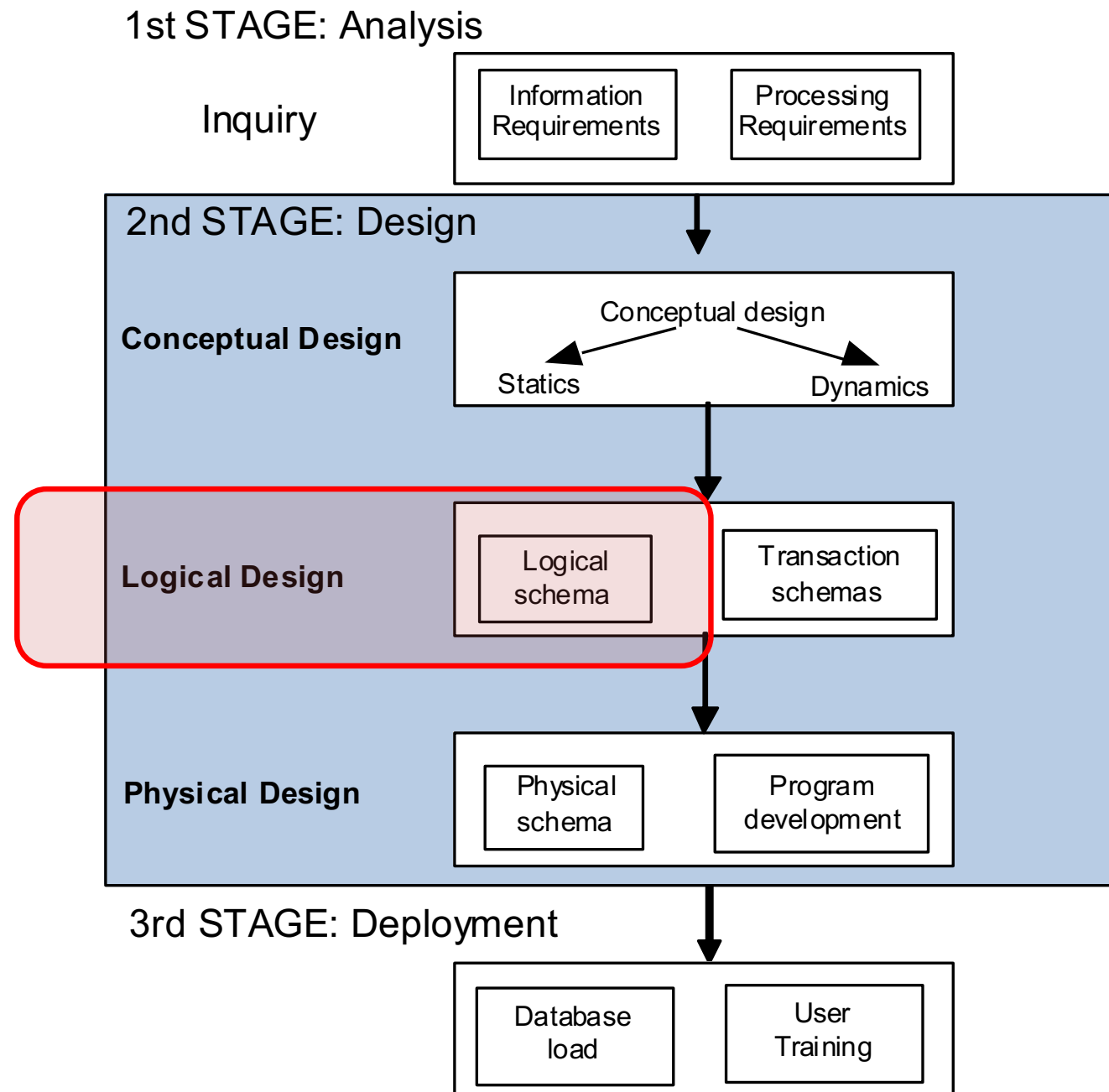
Logical Design

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.

Conceptual Schema

Logical Schema

Logical Design
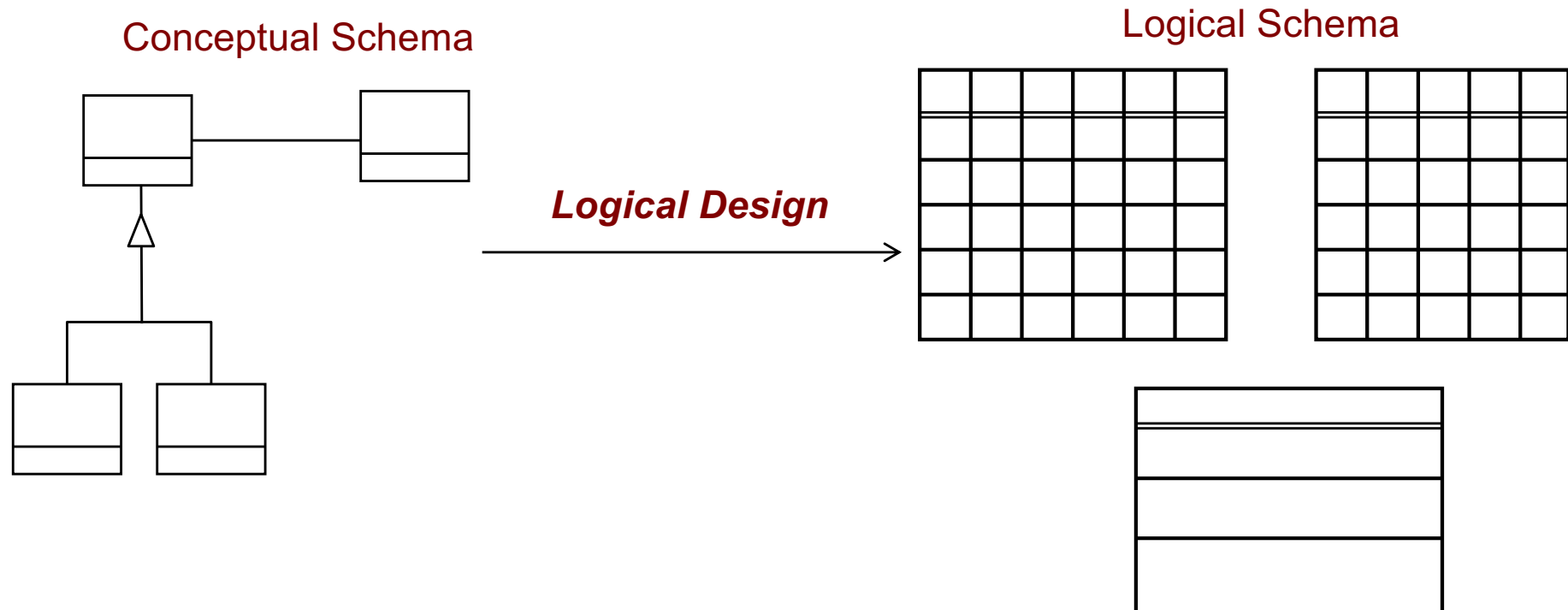
- We are going to apply transformations.
- Multiplicities, associations, and constraints are expressed by the use of PK, FK, NNV, and UNI constraints
- Some properties or constraints cannot be represented using these predefined constraints and we will have to add them to the list of general integrity 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 classes into relations

    1. Strong classes

    2. Weak classes

    3. Specialized classes

II. Transform 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 as 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}$\rightarrow$A(a_0)
A5(a_0:t_a_0,a_5:t_a_5)
  PK:{a_0,a_5}
  FK:{a_0}$\rightarrow$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}$\rightarrow$ Person

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

2.2. Weak classes


```
A(a₀:t_a₀,…)
    PK:{a₀}

B(b₀:t_b₀,a₀:t_a₀,…)
    PK:{a₀,b₀}

    FK:{a₀}→A(a₀)
```
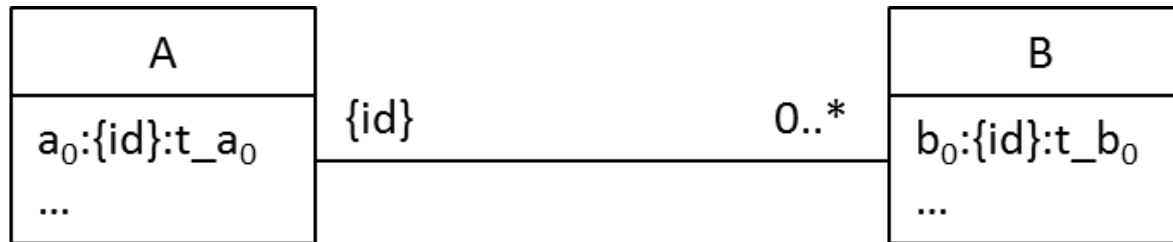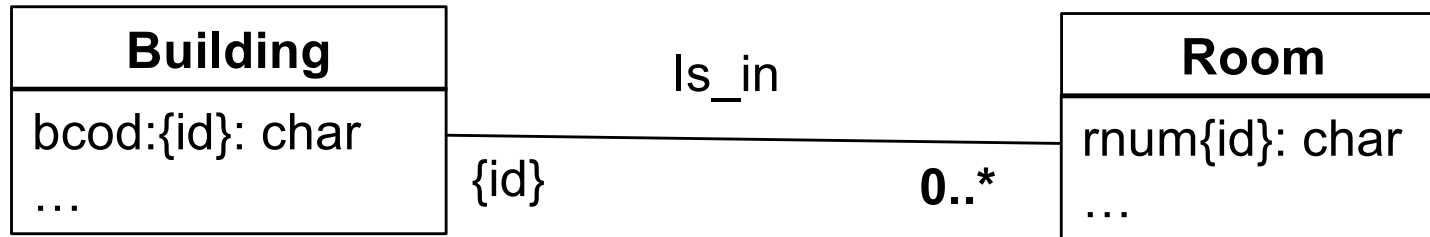
```
Building(bcod:char,…)
    PK:{bcod}

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

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

$B(\mathbf{b_0}:t\_b_0,\mathbf{a_0}:t\_a_0,\ldots)$
    $PK:\{a_0\}$
    $FK:\{a_0\}\rightarrow A(a_0)$
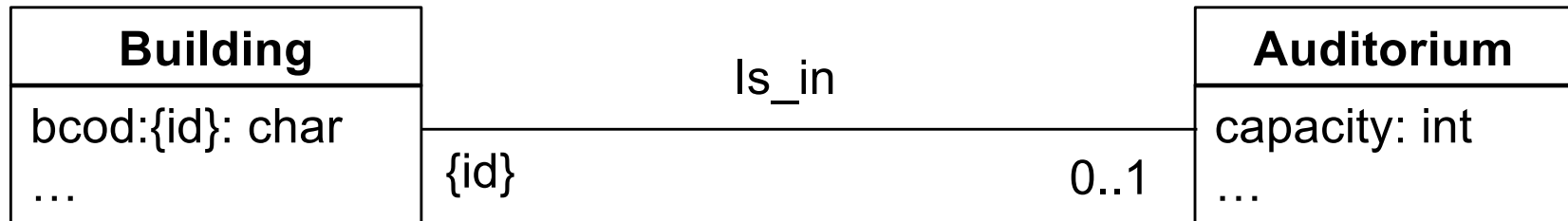
```
Building(bcod: char,…)
    PK:{bcod}

Auditorium(bcod: char, capacity: int, …)
    PK:{bcod}
    FK:{bcod}→ Building
```
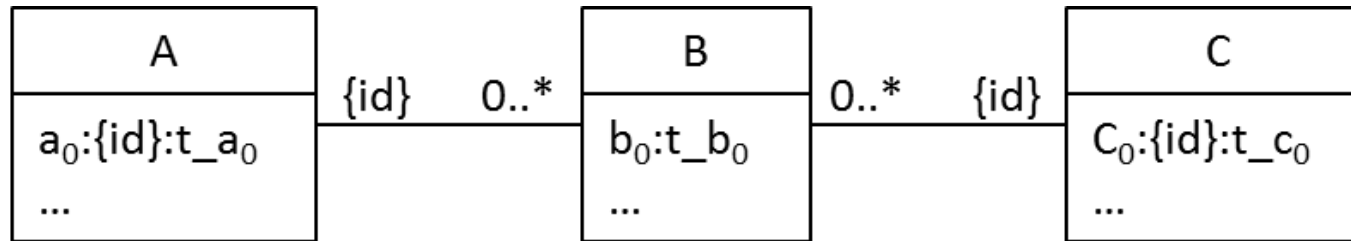
A($\mathbf{a_0}$:t_a$_0$,…)
    PK:{a$_0$}

C($\mathbf{c_0}$:t_c$_0$,…)
    PK:{c$_0$}

B($\mathbf{a_0}$:t_a$_0$, $\mathbf{c_0}$:t_c$_0$, $\mathbf{b_0}$:t_b$_0$,…)
    PK:{a$_0$,c$_0$}
    FK:{a$_0$}$\rightarrow$A(a$_0$)
    FK:{c$_0$}$\rightarrow$C(c$_0$)

Student		take	**Exam**		of	**Subject**
SSN:{id}: char			mark:{1..1}:real			code: {id} char
…	{id}	0..*	…	0..*	{id}	…
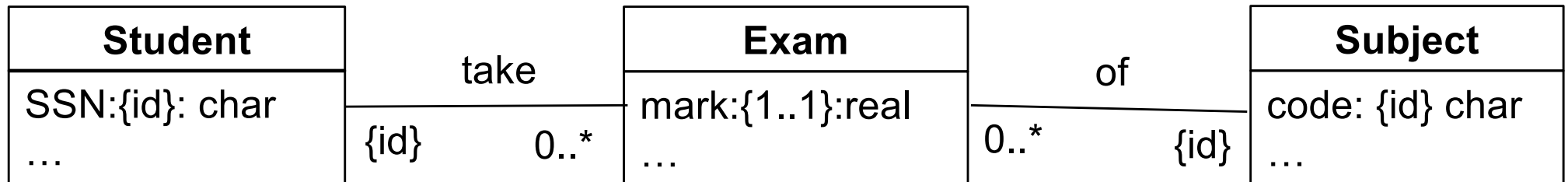
```
Student(SSN:char,…)
    PK:{SSN}

Subject (code: char, …)
    PK:{code}

Exam (SSN: char, code: char, mark: real, …)
    PK:{SSN,code}
    FK:{SSN}→ Student
    FK:{code}→ Subject
    NNV:{mark}
```
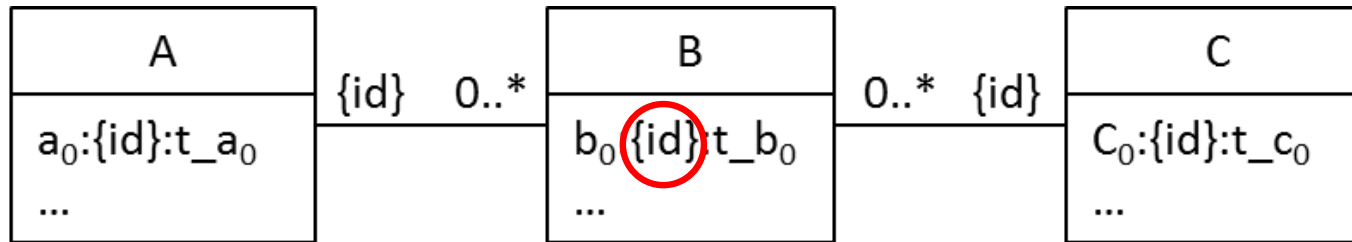
A($\mathbf{a_0}$:t_a$_0$,…)
    PK:{a$_0$}

C($\mathbf{c_0}$:t_c$_0$,…)
    PK:{c$_0$}

B($\mathbf{a_0}$:t_a$_0$, $\mathbf{c_0}$:t_c$_0$, $\mathbf{b_0}$:t_b$_0$,…)
    PK:{a$_0$, c$_0$, b$_0$}
    FK:{a$_0$}→A(a$_0$)
    FK:{c$_0$}→C(c$_0$)
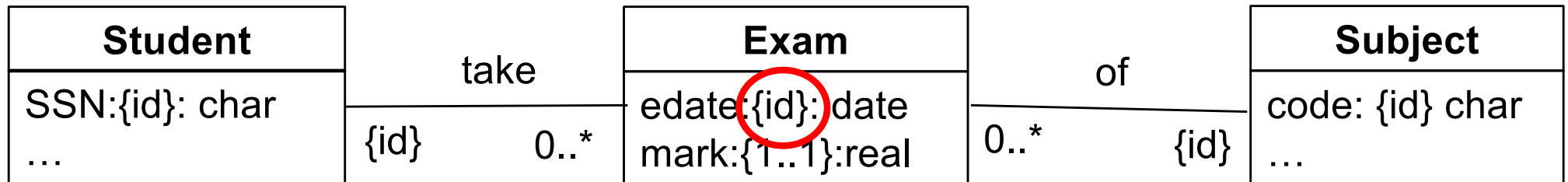
```
Student(SSN: char,…)
     PK:{SSN}

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}
```
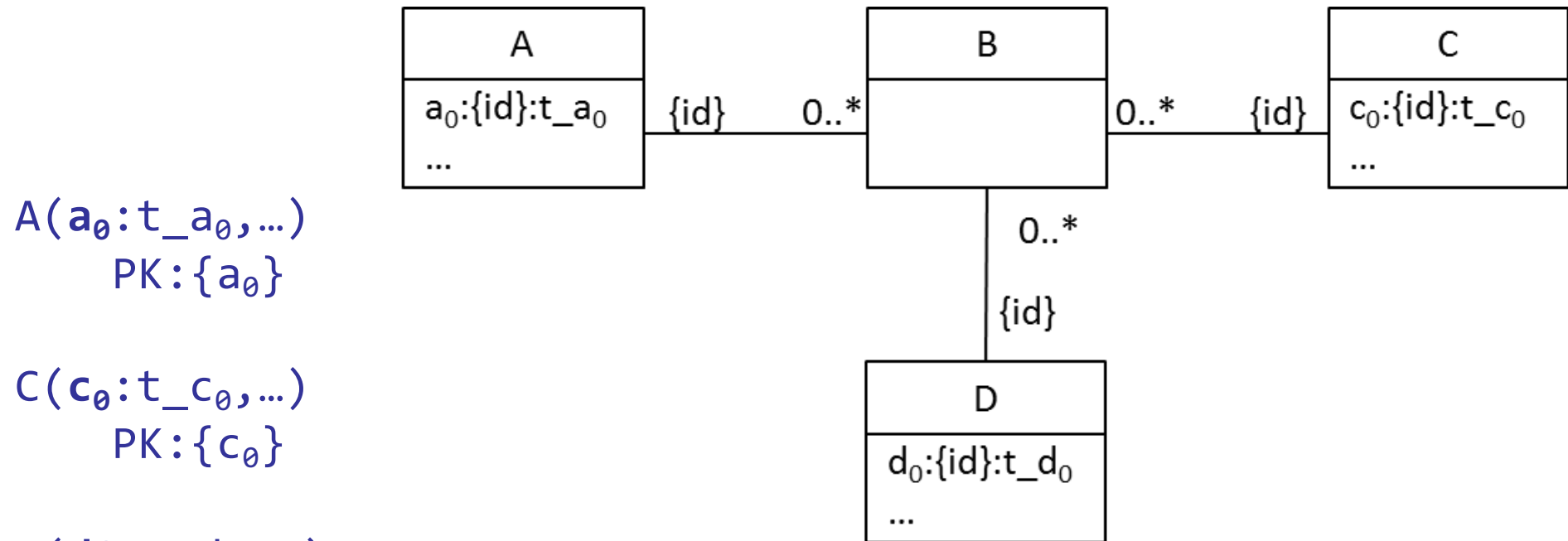
$A(a_0:t\_a_0,…)$
PK:$\{a_0\}$

$C(c_0:t\_c_0,…)$
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:$\{a_0\}{\to}A(a_0)$
FK:$\{c_0\}{\to}C(c_0)$
FK:$\{d_0\}{\to}D(d_0)$

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

20

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

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        │
        ├─────────────────┤
        │ a₀: {id}: t_a₀  │
        │ ...             │
        ├─────────────────┤
        │                 │
        └─────────────────┘
```

{Partial, Overlapping}

```
┌───────────┐   ┌───────────┐   ┌───────────┐
│     B     │   │     C     │   │     D     │
├───────────┤   ├───────────┤   ├───────────┤
│ b₀: t_b₀  │   │ c₀: t_c₀  │   │ d₀: t_d₀  │
│ ...       │   │ ...       │   │ ...       │
├───────────┤   ├───────────┤   ├───────────┤
│           │   │           │   │           │
└───────────┘   └───────────┘   └───────────┘
```
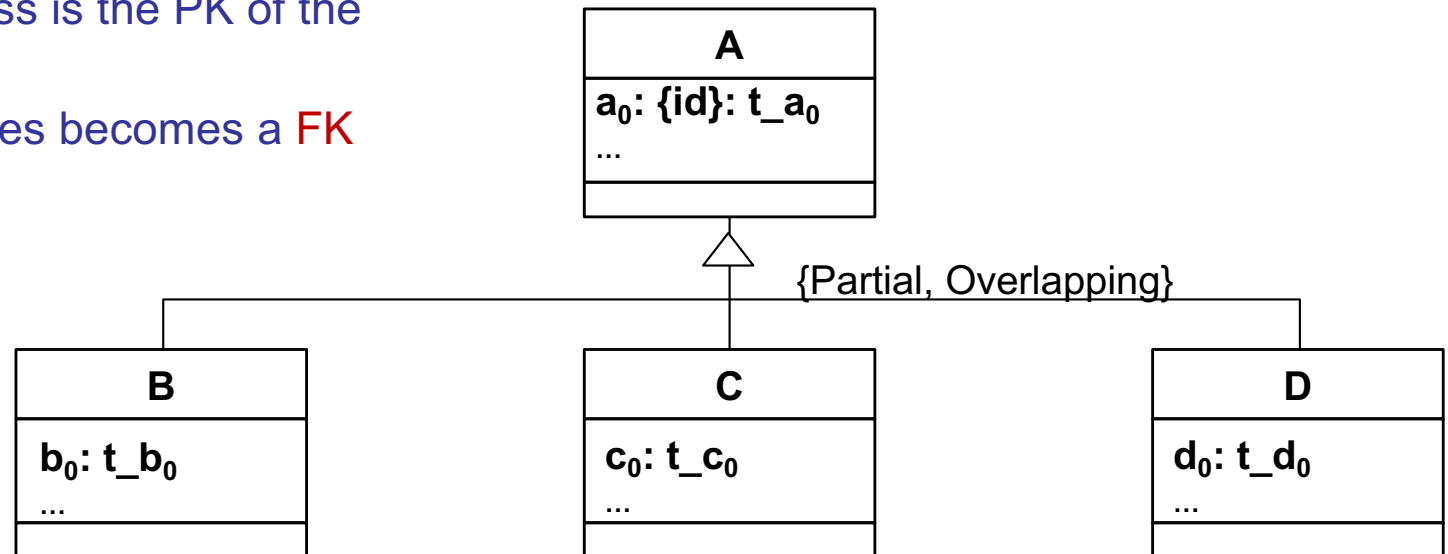
$A(a_0:t\_a_0, \dots)$
   $PK:\{a_0\}$

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

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

$D(a_0:t\_a_0,\ d_0:t\_d_0, \dots)$
   $PK:\{a_0\}$
   $FK:\{a_0\} \rightarrow A(a_0)$

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.*

IC 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.*

(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
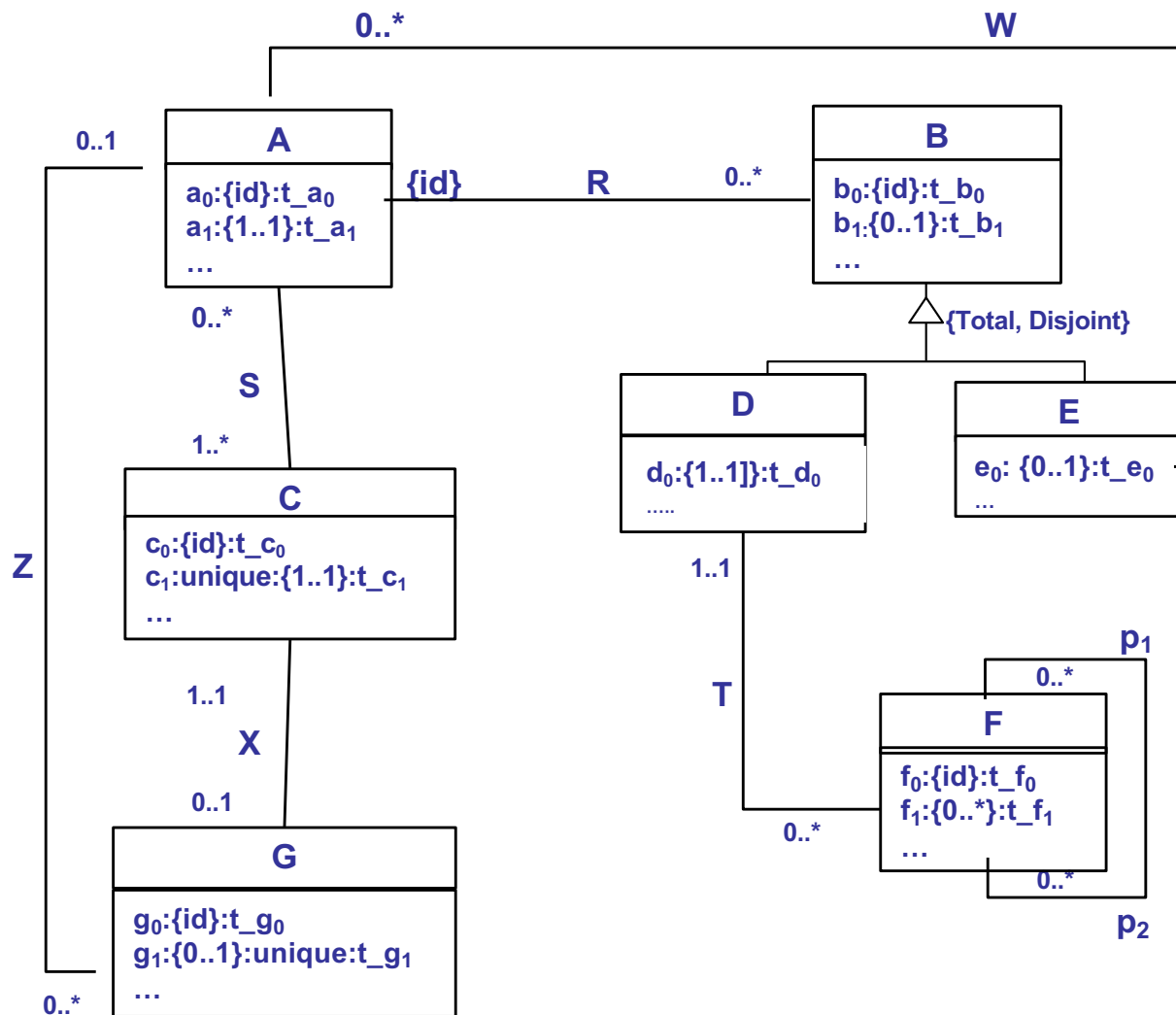
Unit 4.3. Logical Design

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

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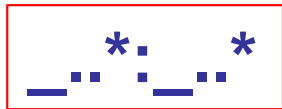
_..*._..*   ⟹   Add a new R
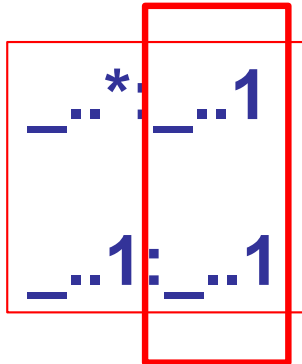
_..*._..1   ⟹   1..1: Do not add any R. Represent the association in the relation with the 1..1 multiplicity (Other existence constraint could be added to the system)

_..1._..1

        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
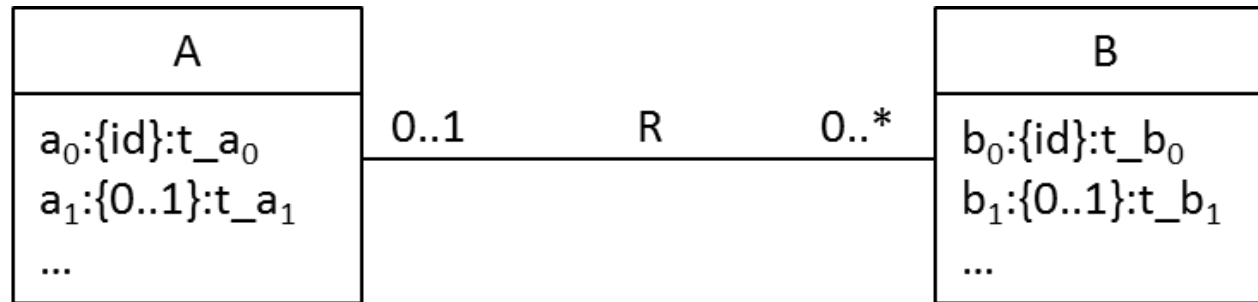
A(a$_0$:t_a$_0$,a$_1$:t_a$_1$,…)
  PK:{a$_0$}

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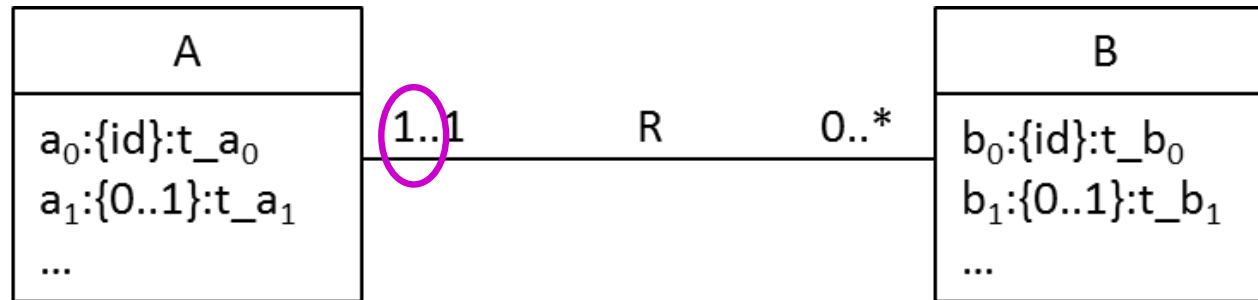
$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$
  $PK:\{a_0\}$

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

Example:
   A: Person
   B: Car
   R: owns

0..1 : 0..1 Association
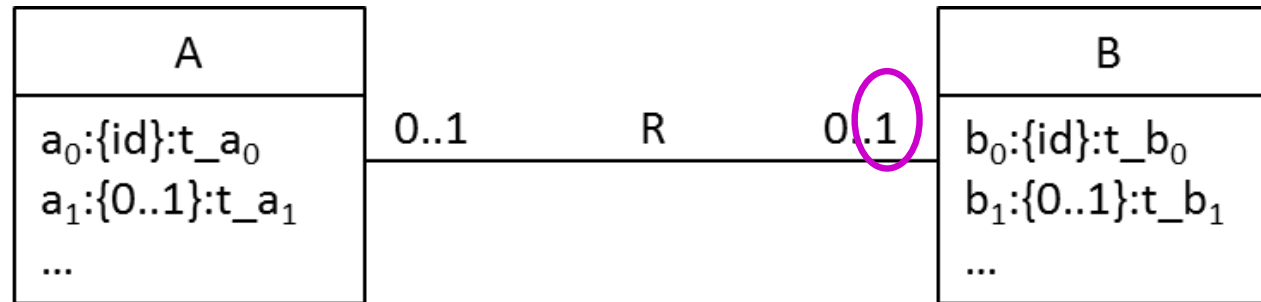
Option 1

```
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)
```

0..1 : 0..1 Association
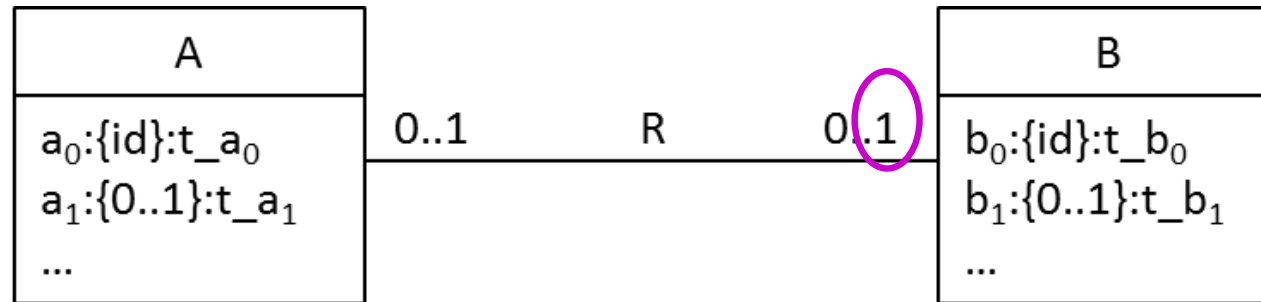
Option 2

$A(a_0:t\_a_0, a_1:t\_a_1, \ldots, b_0:t\_b_0)$
    $PK:\{a_0\}$
    $UNI:\{b_0\}$
    $FK:\{b_0\} \rightarrow B(b_0)$

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

0..1 : 0..1 Association

A		B
a_0:{id}:t_a$_0$	0..1 R 0..1	b_0:{id}:t_b$_0$
a_1:{0..1}:t_a$_1$		b_1:{0..1}:t_b$_1$
...		...
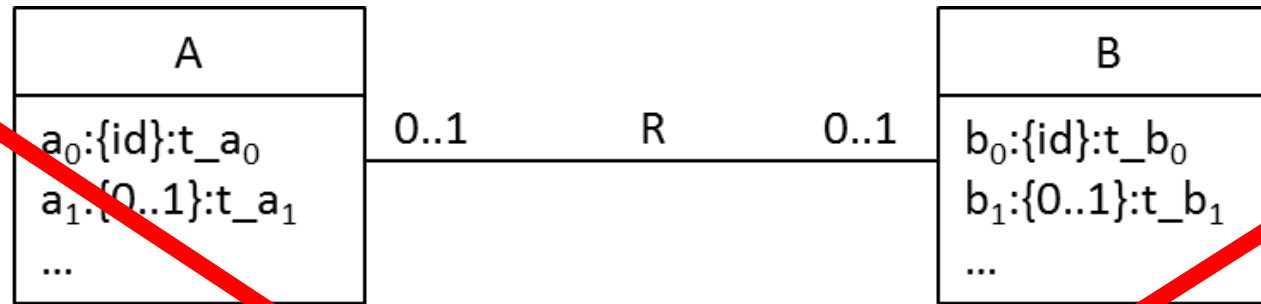
Option 3

```
A(a₀:t_a₀,a₁:t_a₁,…)
    PK:{a₀}


B(b₀:t_b₀,b₁:t_b₁,…)
    PK:{b₀}


R(b₀:t_b₀,a₀:t_a₀)
    PK:{b₀}
    UNI:{a₀}
    NNV:{a₀}
    FK:{a₀} → A(a₀)
    FK:{b₀} → B(b₀)
```

There are more relations:
It is worse

31

0..1 : 0..1 Association

A

a_0:{id}:t_a_0
a_1:{0..1}:t_a_1
…

0..1   R   0..1

B

b_0:{id}:t_b_0
b_1:{0..1}:t_b_1
…
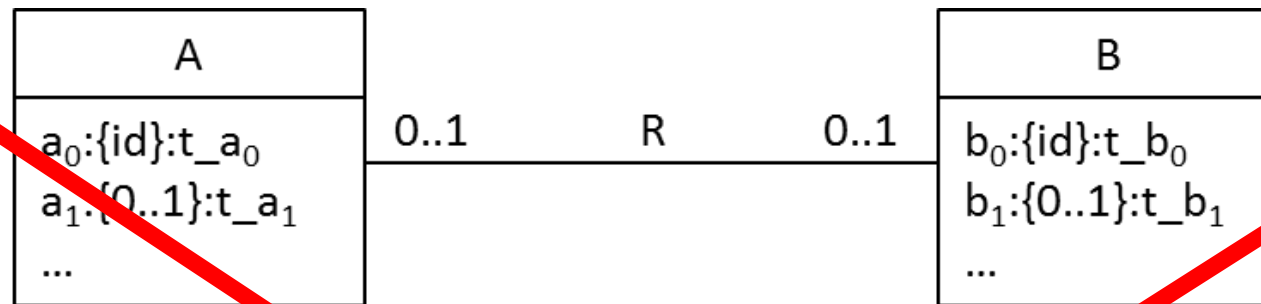
Option_4

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

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

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

There are more relations:
It is worse

1..1 : 0..1 Association
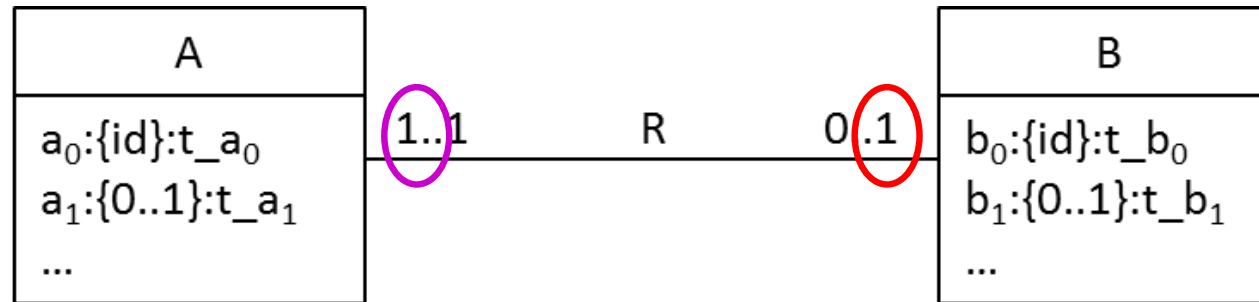

```
A(a₀:t_a₀,a₁:t_a₁,…)
    PK:{a₀}

B(b₀:t_b₀,b₁:t_b₁,…,a₀:t_a₀)
    PK:{b₀}
    UNI:{a₀}
    NNV:{a₀}
    FK:{a₀} → A(a₀)
```
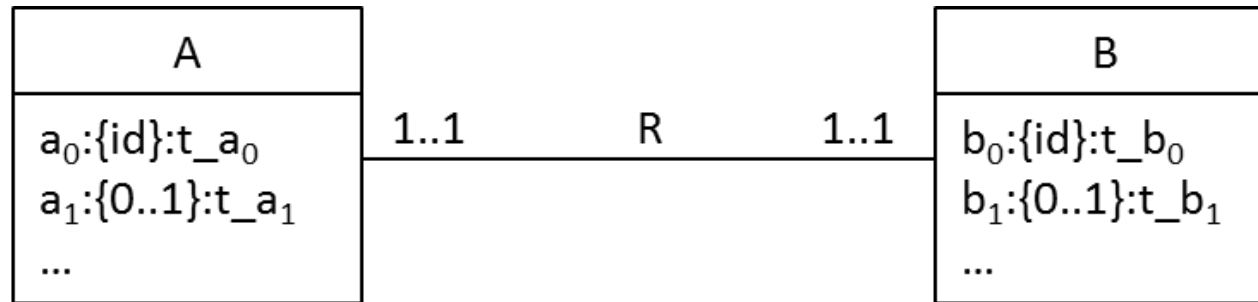
1..1 : 1..1 Association

Option 1

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
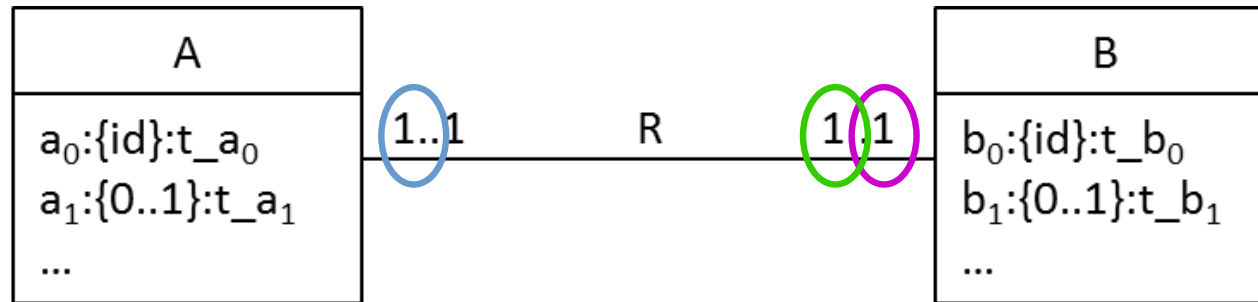
Option 2

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

This FK is possible because a_0 in B has Uniqueness constraint

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

Best option

0..1 : 1..* Association
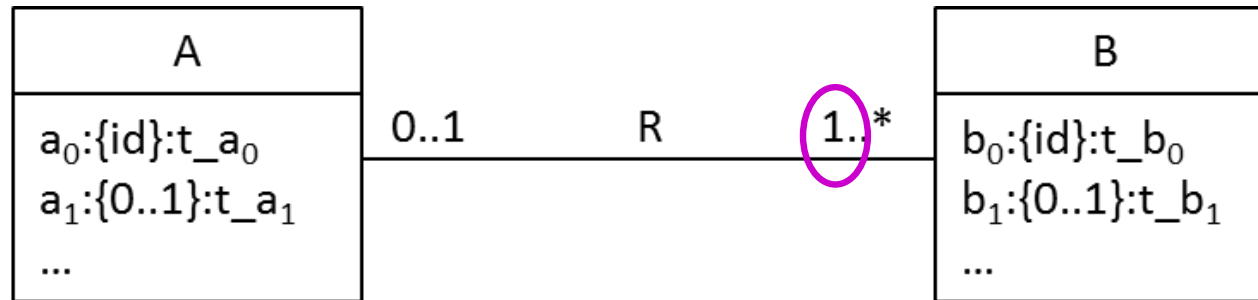
$A(a_0:t\_a_0, a_1:t\_a_1, …)$
  $PK:\{a_0\}$

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

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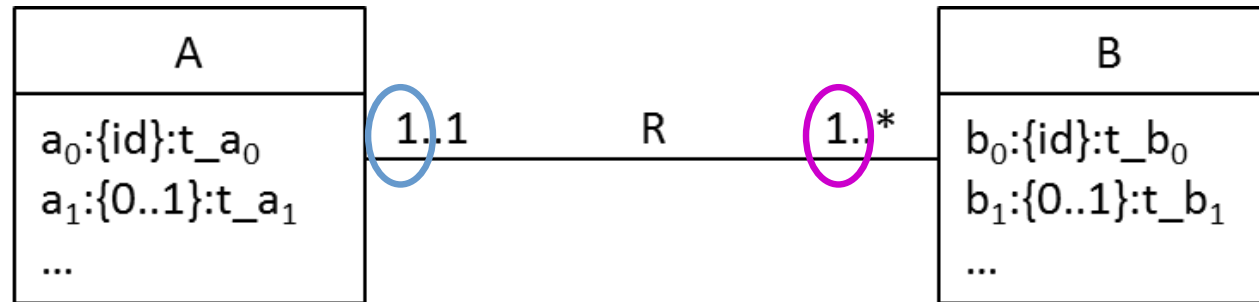
$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$
 $PK:\{a_0\}$

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

 $NNV:\{a_0\}$

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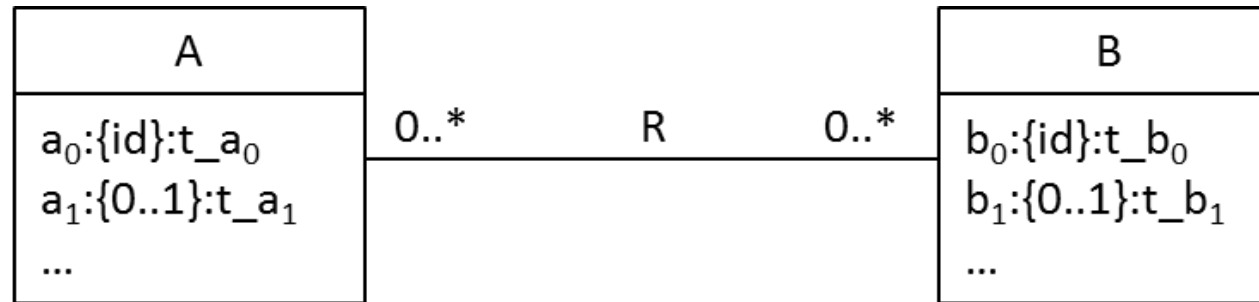
0..* : 0..* Association


```
A(a₀:t_a₀,a₁:t_a₁,…)
  PK:{a₀}

B(b₀:t_b₀,b₁:t_b₁,…)
  PK:{b₀}

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

Example:
A: Book
B: Person
R: has_as_author

1..* : 0..* Association
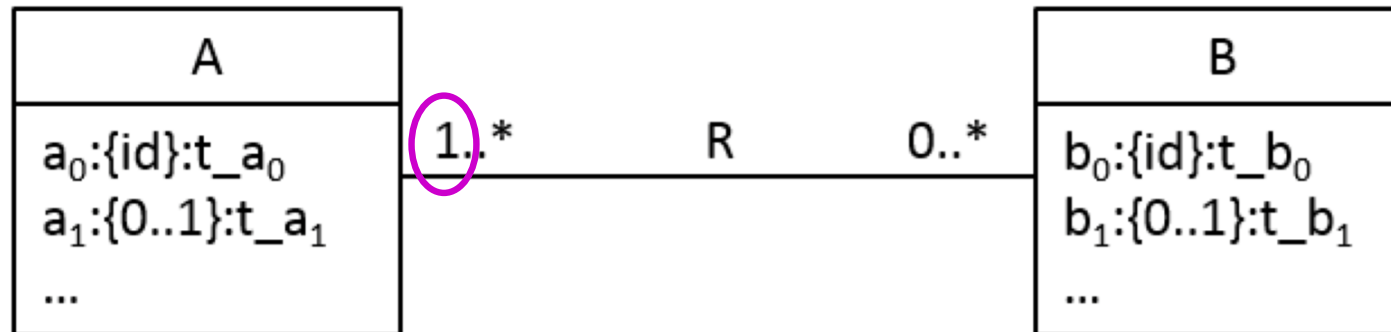
$$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$$
$$PK:\{a_0\}$$

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

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

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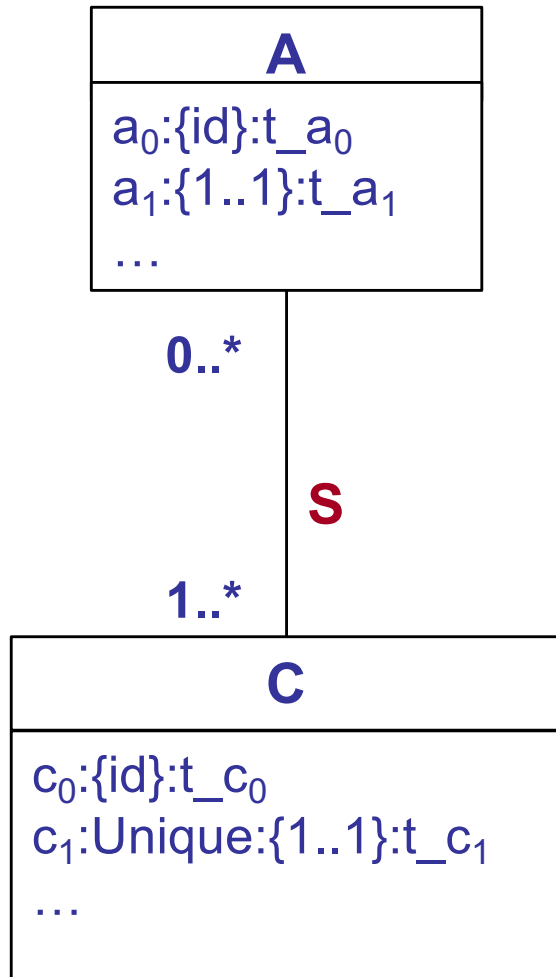
A

a_0:{id}:t_a_0
a_1:{1..1}:t_a_1
…

0..*

S

1..*

C

c_0:{id}:t_c_0
c_1:Unique:{1..1}:t_c_1
…

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

0..1 : 0..* Reflexive association

Reflexive associations are handle as any other binary association
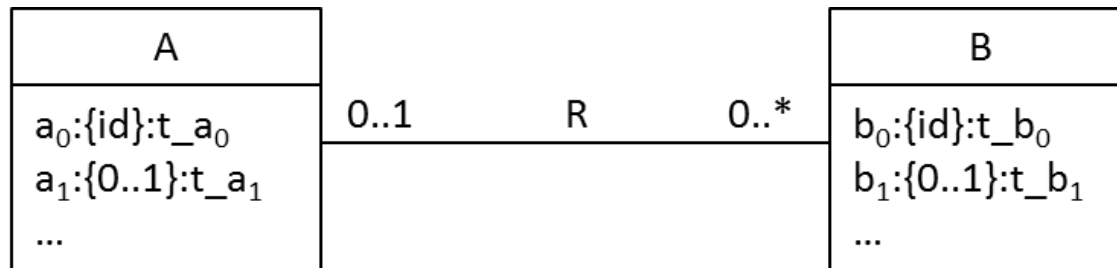
a_0	a_1	...	$a_0\_p_2$
1	b	...	
2	r	...	1
3	n	...	2
4	m	...	1

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

$B(b_0:t\_b_0, b_1:t\_b_1, …, a_0:t\_a_0)$
    $PK:\{b_0\}$
    $FK:\{a_0\} \rightarrow A(a_0)$
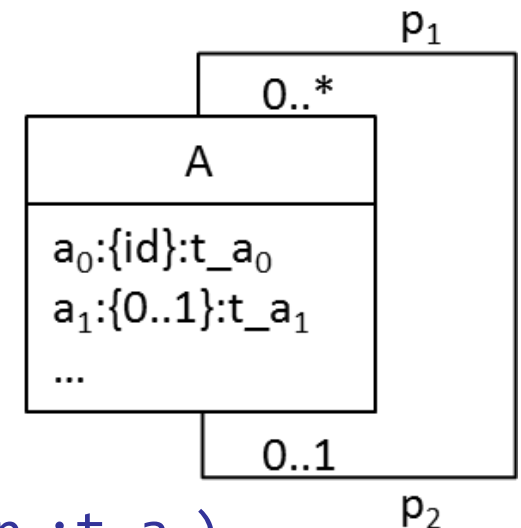
$A(a_0:t\_a_0, a_1:t\_a_1, …, a_0\_p_2:t\_a_0)$
    $PK:\{a_0\}$
    $FK:\{a_0\_p_2\} \rightarrow A(a_0)$

0..1 : 0..1 Reflexive association
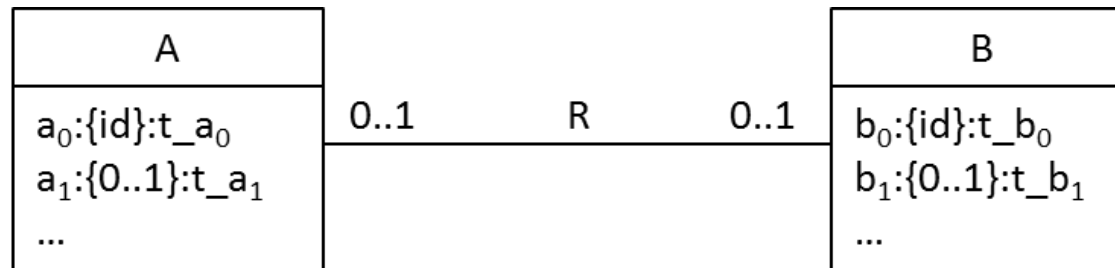
$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$
    $PK:\{a_0\}$

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

$A(a_0:t\_a_0, a_1:t\_a_1, \ldots, a_0\_p_2:t\_a_0)$
    $PK:\{a_0\}$
    $UNI:\{a_0\_p_2\}$
    $FK:\{a_0\_p_2\} \rightarrow A(a_0)$
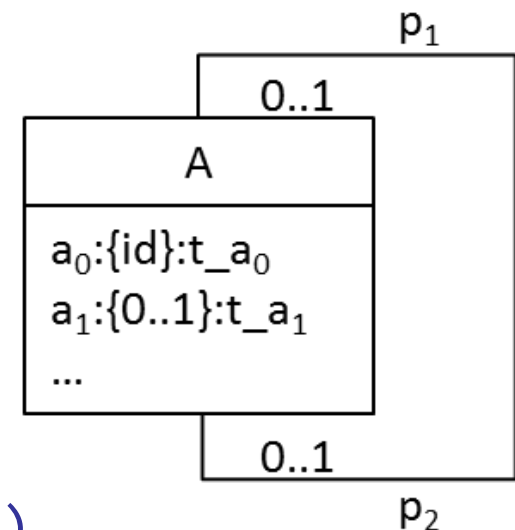
Example:
    A: Person
    R: husband_of

43

1..1 : 0..* Reflexive association
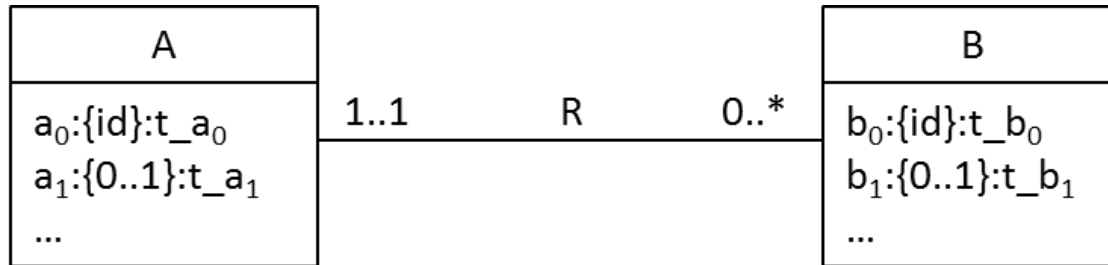
A

A
a_0:{id}:t_a_0
a_1:{0..1}:t_a_1
...

1..1   R   0..*

B
b_0:{id}:t_b_0
b_1:{0..1}:t_b_1
...

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

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

p_1

0..*

A
a_0:{id}:t_a_0
a_1:{0..1}:t_a_1
...

1..1

p_2
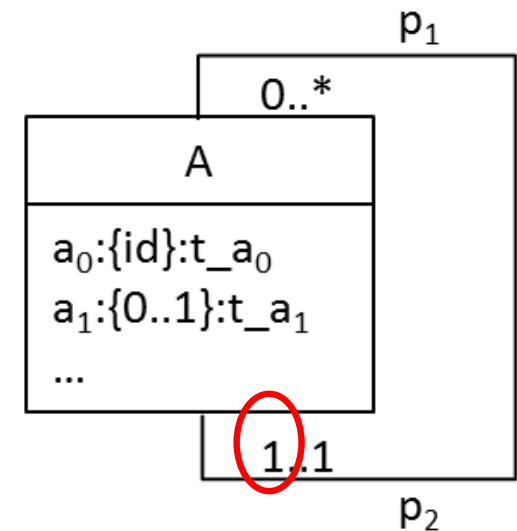
$A(a_0:t\_a_0, a_1:t\_a_1, ..., a_0\_p_2:t\_a_0)$
   $PK:\{a_0\}$
   $FK:\{a_0\_p_2\} \rightarrow A(a_0)$
   $NNV:\{a_0\_p_2\}$

0..1 : 1..1 Reflexive association
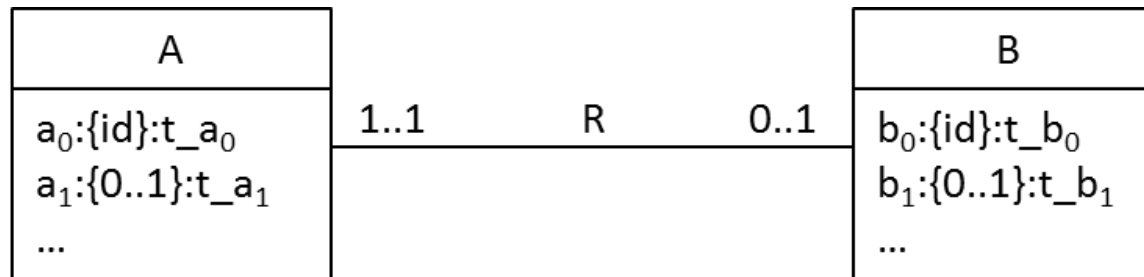
$A(a_0:t\_a_0, a_1:t\_a_1, \dots)$
    $PK:\{a_0\}$

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

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

45
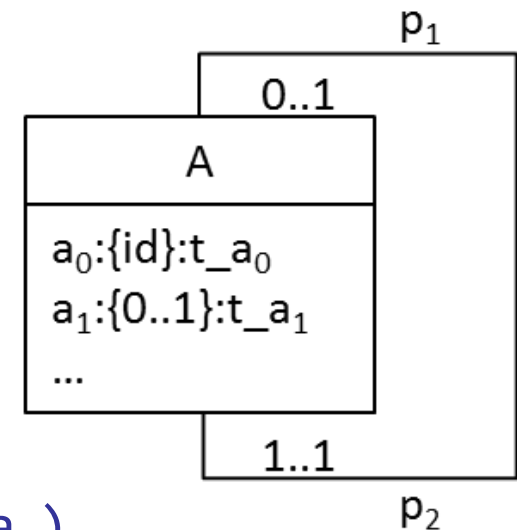
0..* : 0..* Reflexive association
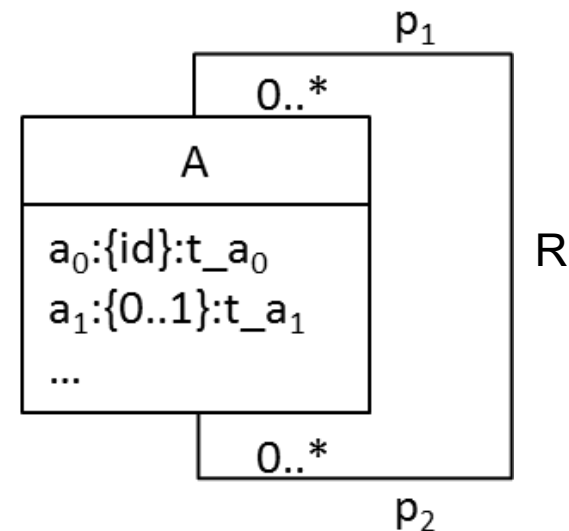
$A(a_0:t\_a_0, a_1:t\_a_1, \dots)$

    $PK:\{a_0\}$

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

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

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

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

Example:
    A: Country
    R: borders

Exercise 1b. Transform the associations
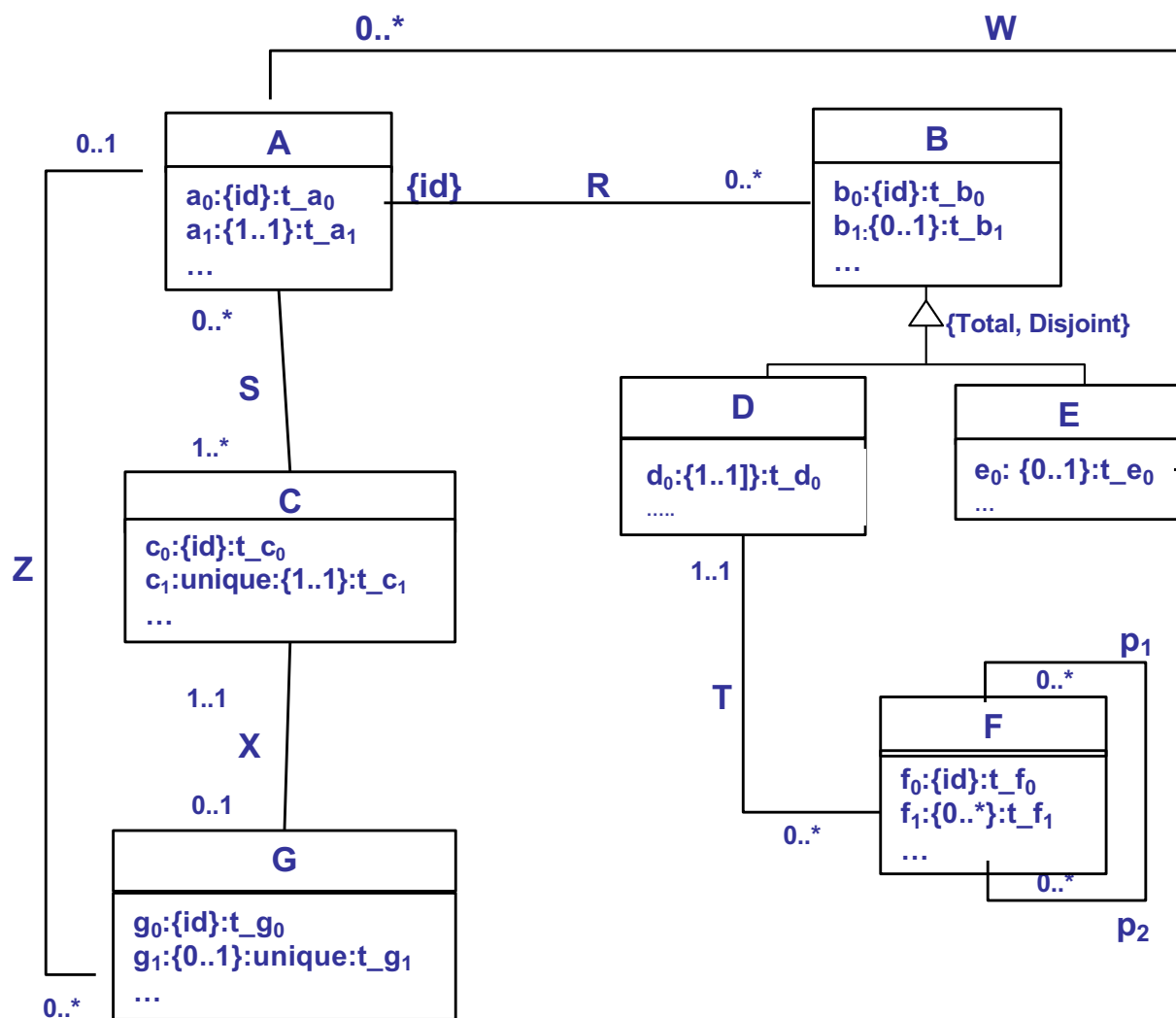
A(a_0:t_a$_0$, a_1:t_a$_1$, ...)
  PK:{a_0}
  NNV:{a_1}

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
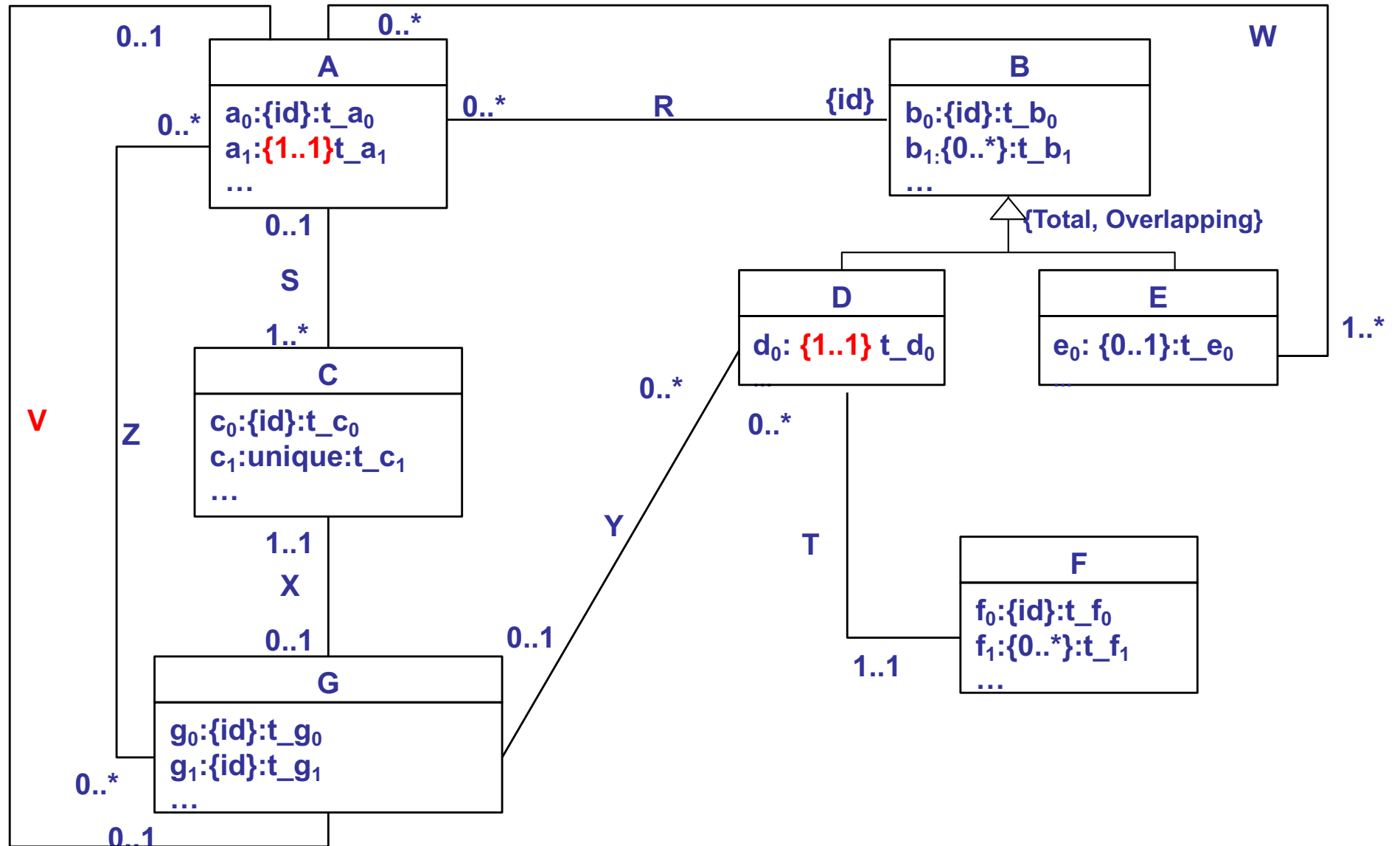
Exercise 3

Exercise 3

Class transformation:

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

$B\_b1(b_0: t\_b_0, b_1: t\_b_1)$ ← **b_1 is multi-valued**
PK: $\{b_0, b_1\}$
FK: $\{b_0\} \rightarrow B$

$A(a_0: t\_a_0, a_1: t\_a_1, \ldots, b_0: t\_b_0)$
PK: $\{a_0, b_0\}$
FK: $\{b_0\} \rightarrow B$ ← **A is weak**
NNV: $\{a_1\}$

$D(b_0: t\_b_0, d_0: t\_d_0, \ldots)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$ ← **D specializes B**
NNV: $\{d_0\}$

$E(b_0: t\_b_0, e_0: t\_e_0, \ldots)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$ ← **E specializes B**

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

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

$F\_f1(f_0: t\_f_0, f_1: t\_f_1)$
PK: $\{f_0, f_1\}$ ← **f_1 is multi-valued**
FK: $\{f_0\} \rightarrow F$

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

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

Exercise 3

Association 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
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}
D(b_0: t_b_0, d_0: t_d_0, …, f_0: t_f_0, g_0: t_g_0, g_1: t_g_1)
    PK: {b_0}
    FK: {b_0} → B
    NNV: {d_0}
    FK: {f_0} → F
    NNV: {f_0}
    FK: {g_0, g_1} → G   **T** **Y**
E(b_0: t_b_0, e_0: t_e_0, …)
    PK: {b_0}
    FK: {b_0} → B

IC$_{Total}$: "Every value which appears in b_0 of B must appear in b_0 of D or E".

C(c_0: t_c_0, c_1: t_c_1, …, a_0: t_a_0, b_0: t_b_0)
    PK: {c_0}
    UNI: {c_1}   **S**
    FK: {a_0, b_0} → A(a_0, b_0)
IC: All {a_0, b_0} in A must be in C
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_0)
G(g_0: t_g_0, g_1: t_g_1, …, c_0: t_c_0, a_0: t_a_0, b_0: t_b_0)
    PK: {g_0, g_1}
    FK: {c_0} → C
    NNV: {c_0}   UNI: {c_0} **X**
    FK: {a_0, b_0} → A(a_0, b_0)
    UNI: {a_0, b_0} **V**

    (can be symmetrically solved in A)

W(a_0: t_a_0, b_{0A}: t_b_0, b_{0E}: t_b_0)
    PK: {a_0, b_{0A}, b_{0E}}
    FK: {a_0, b_{0A}} → A(a_0, b_{0A})   FK: {b_{0E}} → 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: t_a_0, b_0: t_b_0, g_0: t_g_0, g_1: t_g_1)
    PK: {a_0, b_0, g_0, g_1}
    FK: {a_0, b_0} → A(a_0, b_0)
    FK: {g_0, g_1} → G(g_0, g_1)

51

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

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. Example 1
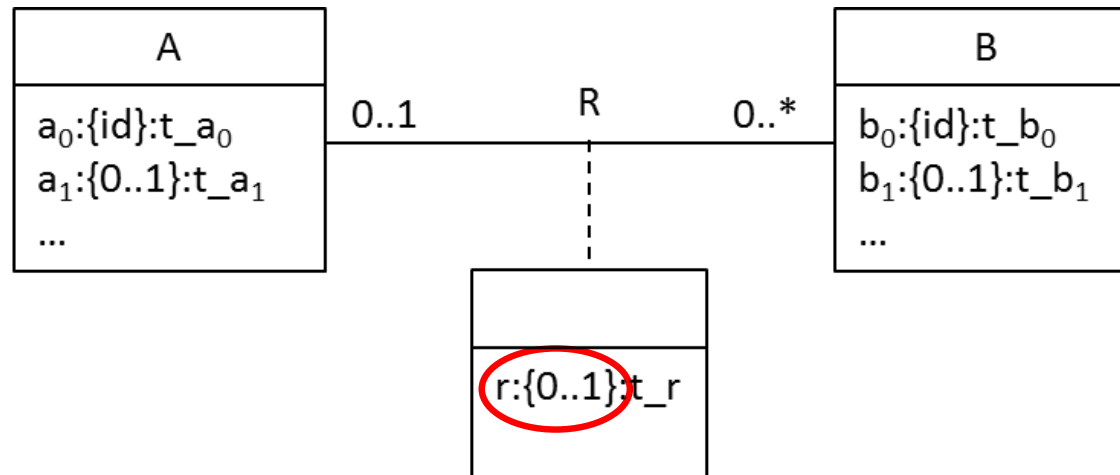
$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$

      $PK:\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots, a_0:t\_a_0, r:t\_r)$

      $PK:\{b_0\}$

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

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

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

Associations with link attributes. Example 1
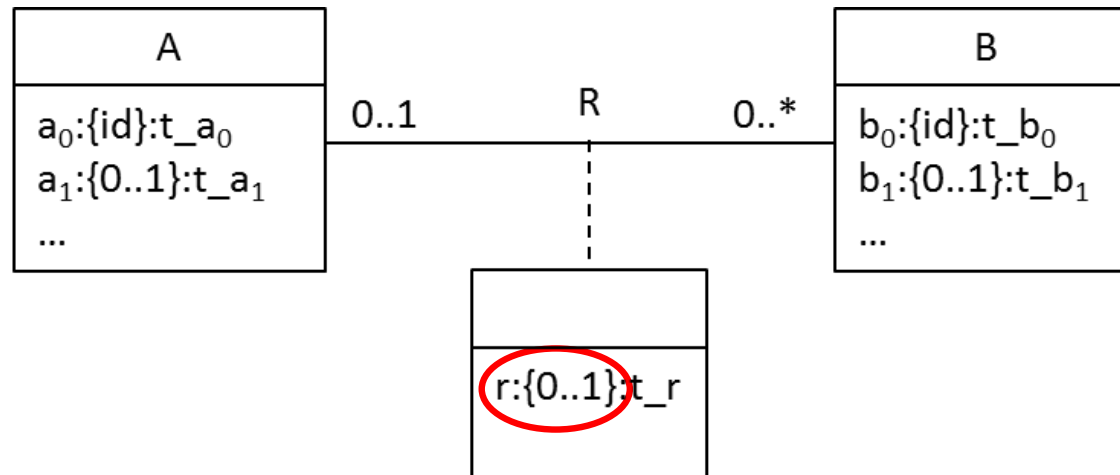
$A(a_0:t\_a_0,a_1:t\_a_1,…)$

      $PK:\{a_0\}$

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

      $PK:\{b_0\}$

      $NNV:\{a_0\}$

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

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

$B(b_0:t\_b_0,b_1:t\_b_1,…)$

      $PK:\{b_0\}$

Better solution:
There is no general constraint

Associations with link attributes. Example 2
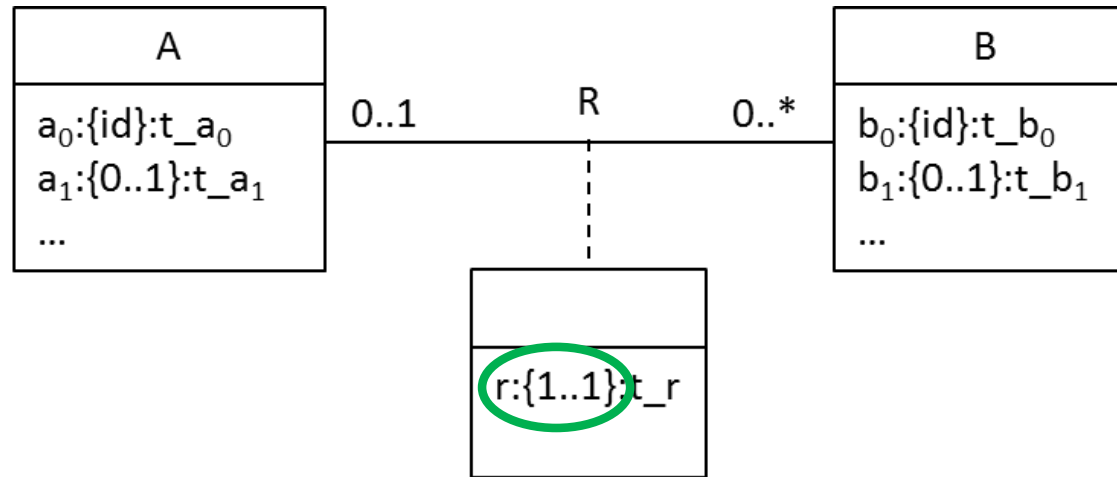
$A(a_0:t\_a_0,a_1:t\_a_1,\ldots)$

    $PK:\{a_0\}$

$B(b_0:t\_b_0,b_1:t\_b_1,\ldots,a_0:t\_a_0,r:t\_r)$

    $PK:\{b_0\}$

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

       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. Example 2
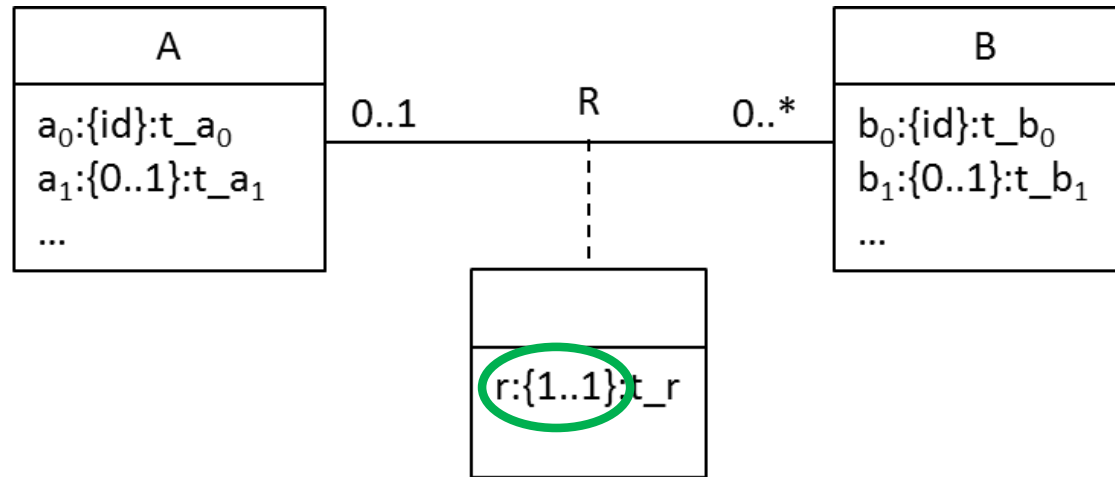
$$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$$
$$\quad PK:\{a_0\}$$

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

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

Better solution
There is no integrity constraint.

Associations with link attributes. Example 3
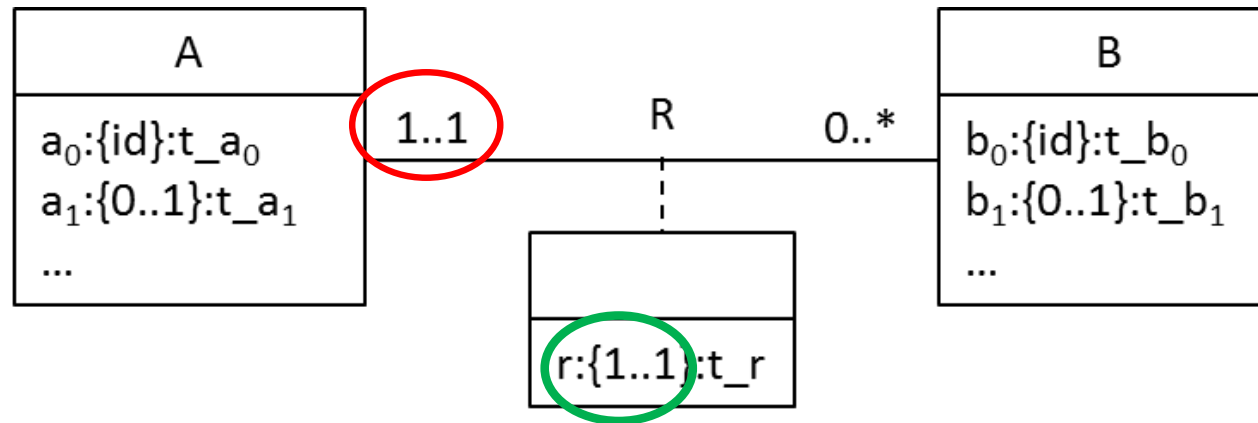
$A(a_0:t\_a_0, a_1:t\_a_1, \ldots)$

    $PK:\{a_0\}$

$B(b_0:t\_b_0, b_1:t\_b_1, \ldots, a_0:t\_a_0, r:t\_r)$

    $PK:\{b_0\}$

    $NNV:\{a_0\}$

    $NNV:\{r\}$

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

Unit 4.3. Logical Design

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

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

$$C(c_0:t\_c_0, c_1:t\_c_1, \ldots, a_0:t\_a_0, b_0:t\_b_0)$$
$$PK:\{c_0\}$$
$$FK:\{a_0, b_0\} \rightarrow R(a_0, b_0)$$
$$NNV:\{a_0, b_0\}$$

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

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

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

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

It can be improved

$A(a_0:t\_a_0,a_1:t\_a_1,…)$

$PK:\{a_0\}$

$B(b_0:t\_b_0,b_1:t\_b_1,…,a_0:t\_a_0)$

$PK:\{b_0\}$

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

$C(c_0:t\_c_0,c_1:t\_c_1,…,b_0:t\_b_0)$

$PK:\{c_0\}$

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

$NNV:\{b_0\}$

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

Better transformation

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

$PK:\{a_0\}$

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

$PK:\{b_0\}$

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

$PK:\{b_0\}$

$NNV:\{a_0\}$

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

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

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

$PK:\{c_0\}$

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

$NNV:\{b_0\}$

65

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

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

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

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

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

$\qquad PK:\{a_0\}$

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

$\qquad PK:\{b_0\}$

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

$\qquad PK:\{c0\}$

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

$\qquad PK:\{a_0,b_0\}$

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

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

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

$\qquad PK:\{a_0,b_0,c_0\}$

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

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

It can be improved

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 6
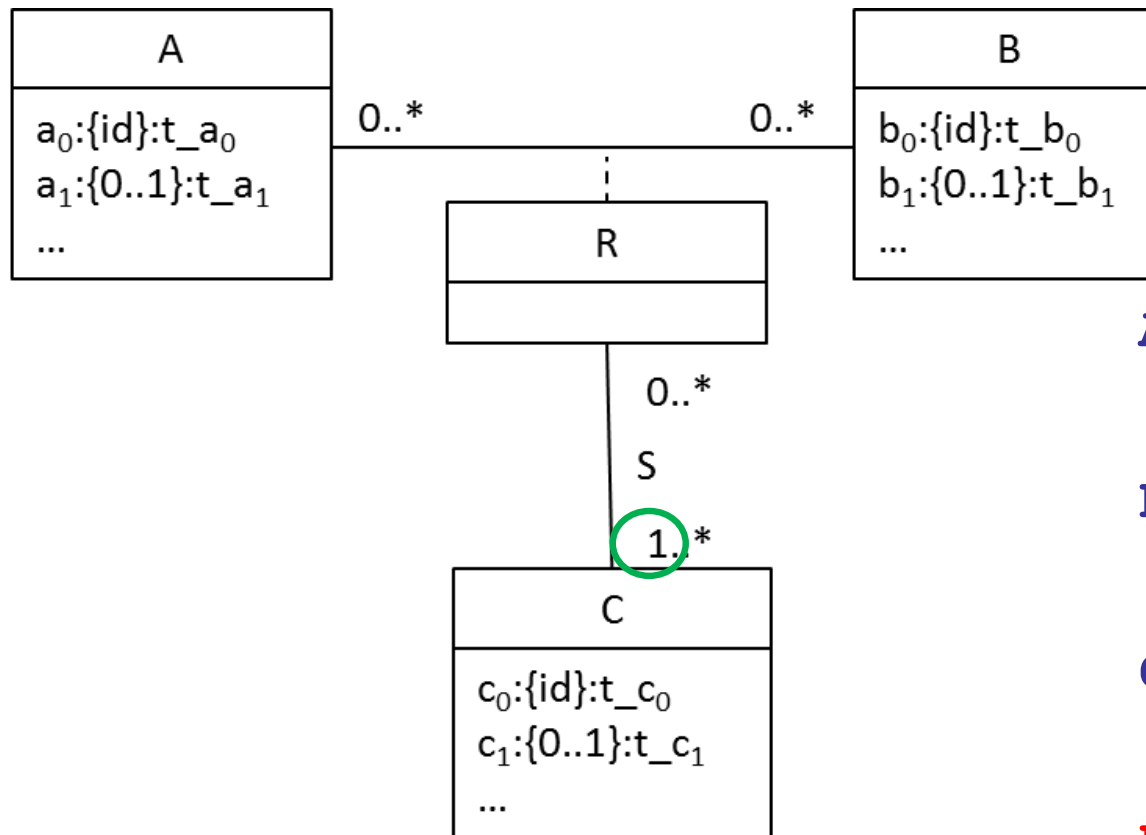
Better transformation

$A(a_0:t\_a_0,a_1:t\_a_1,…)$

$PK:\{a_0\}$

$B(b_0:t\_b_0,b_1:t\_b_1,…)$

$PK:\{b_0\}$

$C(c_0:t\_c_0,c_1:t\_c_1,…)$

$PK:\{c_0\}$

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

$PK:\{a_0,b_0,c_0\}$

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

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

This represents S $\longrightarrow$ $FK:\{c_0\}\rightarrow C(c_0)$

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

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

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 primary key 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

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
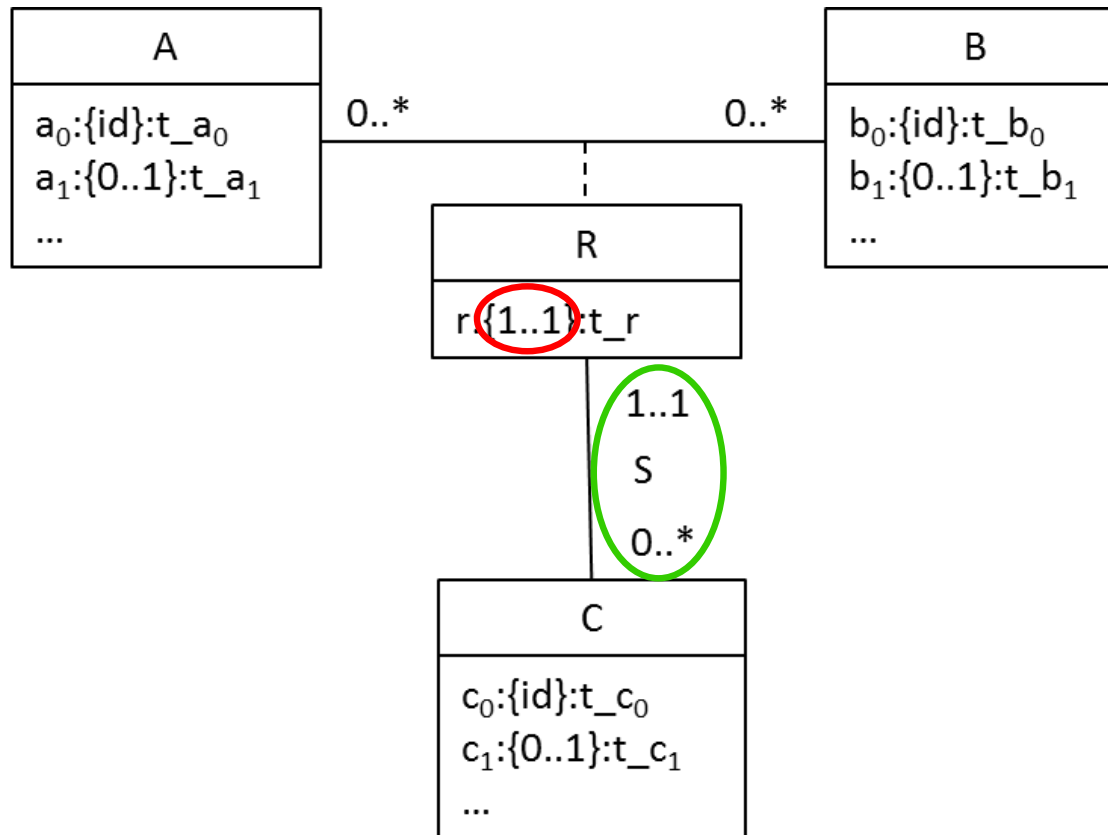
5. **Examples**

6. Introduction to Databases Normalization

A class diagram for Exercise 4.

The diagram contains the following classes and associations:

Class A
- a_0:{id}:t_a_0
- a_1:t_a_1
- ...

Class B
- b_0:{id}:t_b_0
- b_1:{0..1}:t_b_1
- ...

Class C
- c_0:{id}:t_c_0
- c_1:Unique:t_c_1
- ...

Class D
- d_0: t_d_0
- ...

Class E
- e_0: {0..1}:t_e_0
- ...

Class F
- f_0:{id}:t_f_0
- f_1:{0..*}:t_f_1
- ...

Class G
- g_0:{id}:t_g_0
- g_1:{1..1}:Unique:t_g_1
- ...

Associations:
- W: A 0..* — B 0..*
- R: A {id} — B 0..* (1..*)
- S: A 0..* — C 1..*
- X: C 1..1 — G 0..1
- Z: A 0..1 — G 0..*
- B ◁ {Total, Disjoint} D, E
- T: D 1..1 — F 0..*
- Y: E 0..* (W connection)
- p_1: F 0..*
- p_2: F 0..*

{Total, Disjoint}

73
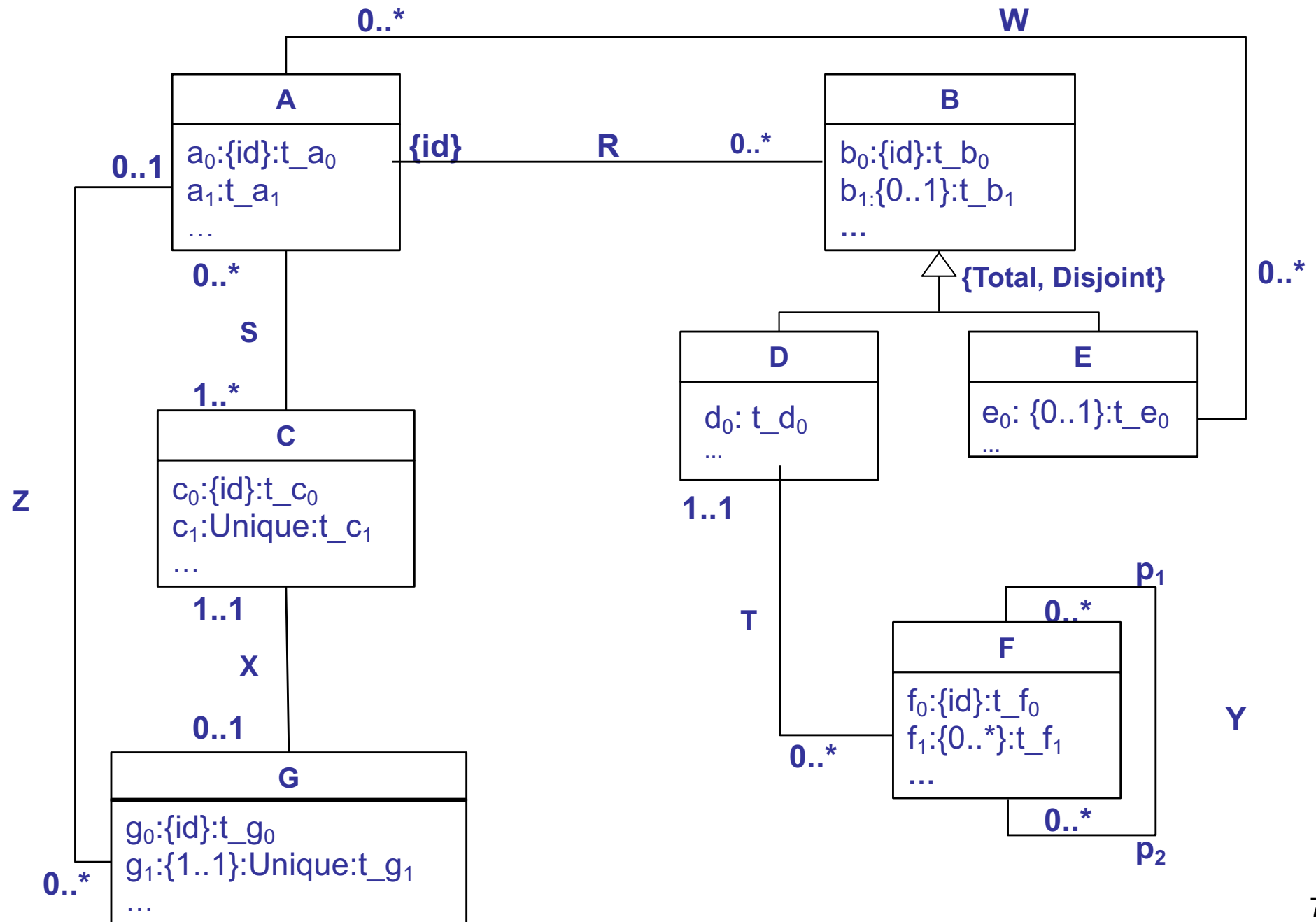
Exercise 4

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

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

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

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

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

$F(f_0: t\_f_0, \ldots)$
    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: $\{f_0\} \rightarrow F(f_0)$

$G(g_0: t\_g_0, g_1: t\_g_1, \ldots)$
    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".

74

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

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

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

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

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

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".

$S(a_0: t\_a_0, c_0: t\_c_0)$
  PK: $\{a_0, c_0\}$ On Opdate Cascade
  FK: $\{a_0\} \rightarrow A(a_0)$ On Delete Cascade
  FK: $\{c_0\} \rightarrow C(c_0)$
                                   On Opdate Cascade

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

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

$F\_f1(f_0: t\_f_0, f_1: t\_f_1)$
  PK: $\{f_0, f_1\}$ On Opdate Cascade
  FK: $\{f_0\} \rightarrow F(f_0)$ On Delete Cascade

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

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

$Y(f0p_1: t\_f_0, f0p_2: t\_f_0)$
  PK: $\{f0p_1, f0p_2\}$
  FK: $\{f0p_1\} \rightarrow F(f_0)$ On Opdate Cascade
  FK: $\{f0p_2\} \rightarrow F(f_0)$ On Opdate Cascade
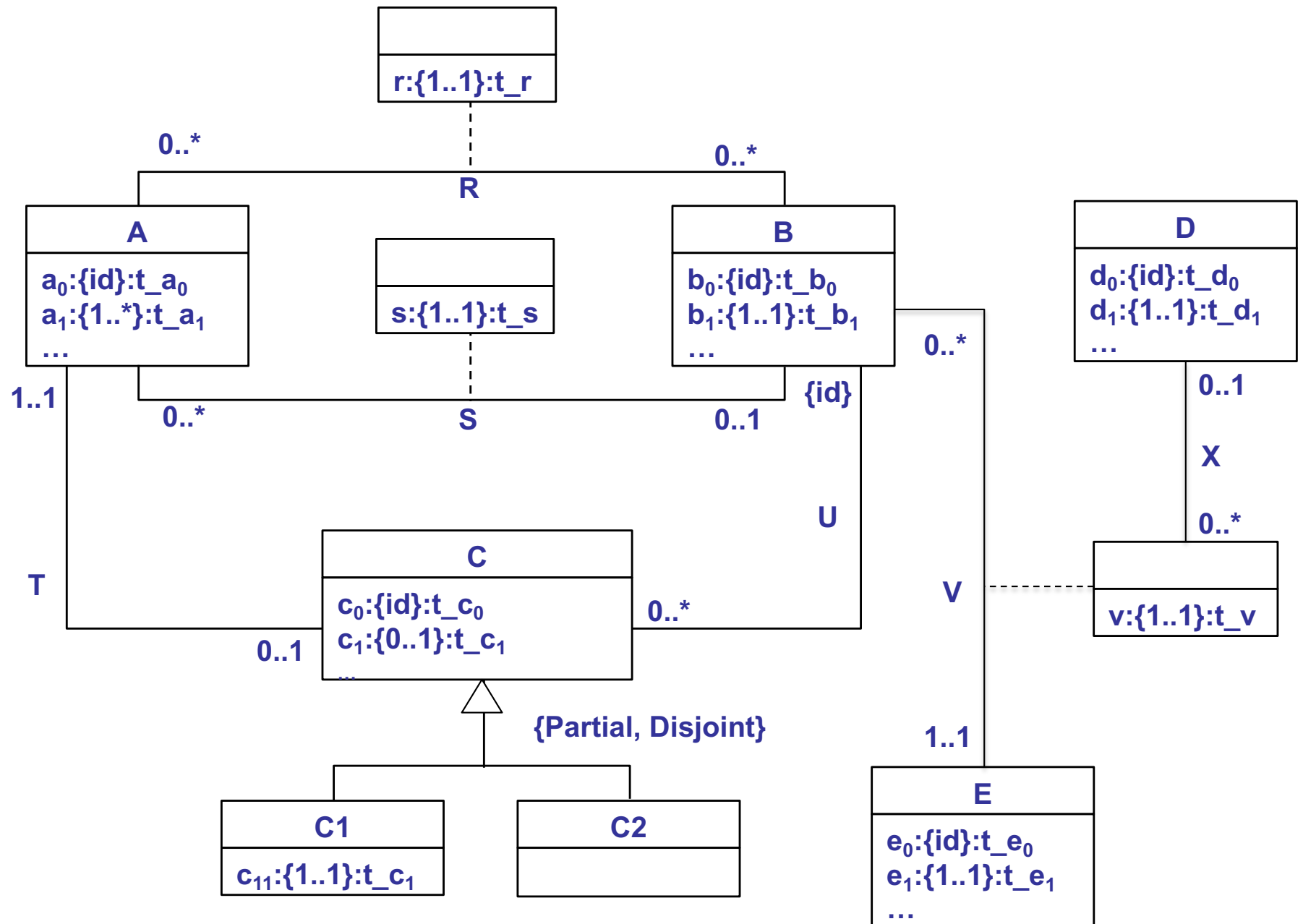
Exercise 5

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:$\{a_0\} \rightarrow$ 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:$\{c_0, b_0\}$
  FK:$\{b_0\} \rightarrow$ B

C1$(c_0: t\_c_0, b_0: t\_b_0, c_{11}: t\_c_{11})$
  PK:$\{c_0, b_0\}$
  FK:$\{c_0, b_0\} \rightarrow$ C
  NNV:$\{c_{11}\}$

C2$(c_0: t\_c_0, b_0: t\_b_0)$
  PK:$\{c_0, b_0\}$
  FK:$\{c_0, b_0\} \rightarrow$ C

r:$\{1..1\}$:t_r

0..*   R   0..*

A
a_0:$\{id\}$:t_a_0
a_1:$\{1..*\}$:t_a_1
…

s:$\{1..1\}$:t_
s

B
b_0:$\{id\}$:t_b_0
b_1:$\{1..1\}$:t_b_1
…

D
d_0:$\{id\}$:t_d_0
d_1:$\{1..1\}$:t_d_1
…

1..1   0..*   S   0..1 $\{id\}$ 0..*   0..1

T   **C**
c_0:$\{id\}$:t_c_0
c_1:$\{0..1\}$:t_c_1
…

U   V

X

0..1   0..*

v:$\{1..1\}$:t_
v

$\{$Partial, Disjoint$\}$

1..1

C1
c_{11}:$\{1..1\}$:t_c_1

C2

E
e_0:$\{id\}$:t_e_0
e_1:$\{1..1\}$:t_e_1
…
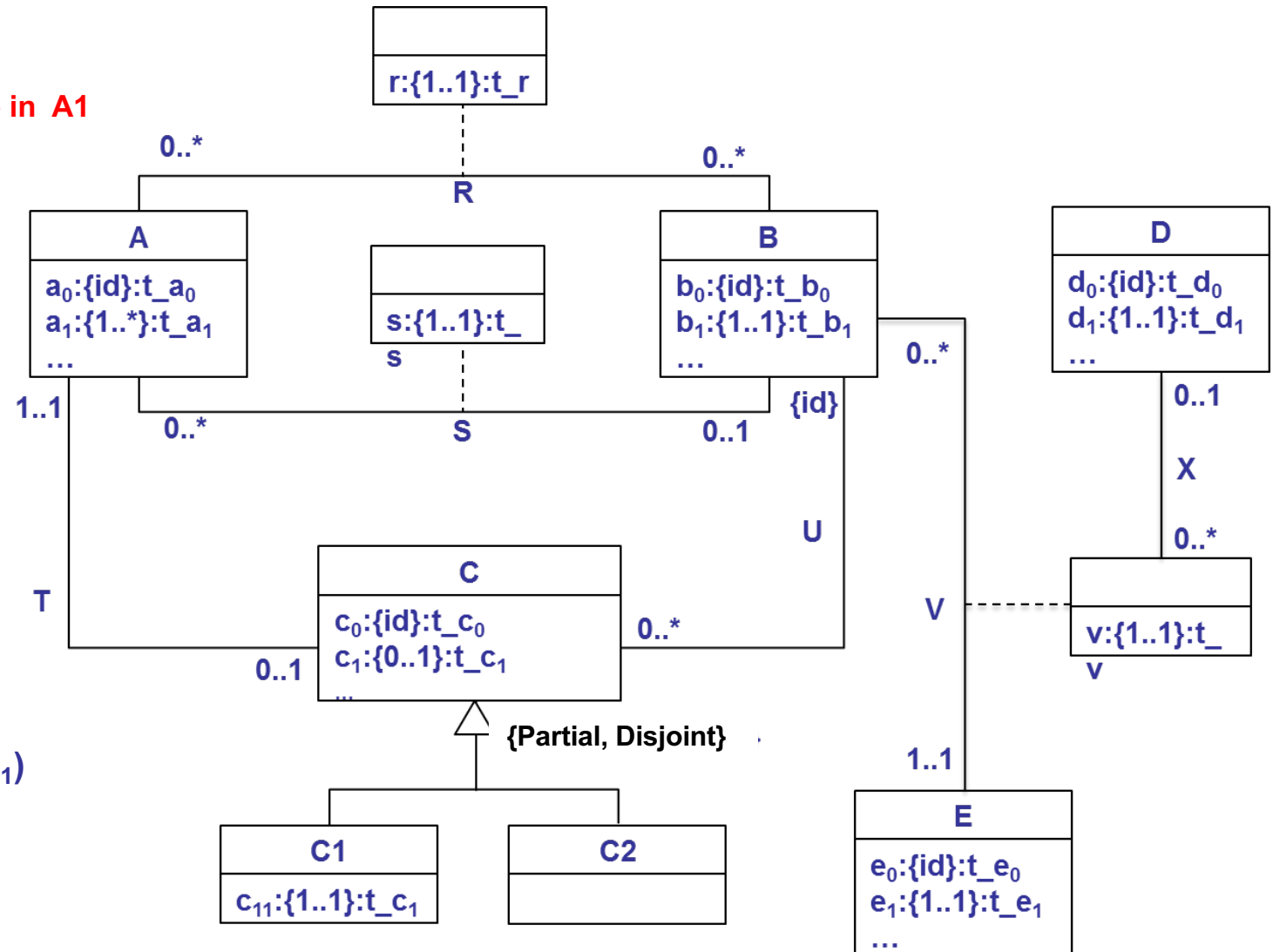
IC2-Disjoint: A value $\{c_0, b_0\}$ in C cannot appear in C1 and C2 at the same time.

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:$\{a_0\} \rightarrow$ 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:$\{e_0\} \rightarrow$ E
  NNV:$\{e_0\}$
  NNV:$\{v\}$
  FK:$\{d_0\} \rightarrow$ 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:$\{a_0\} \rightarrow$ A
  FK:$\{b_0\} \rightarrow$ B
  NNV:$\{r\}$

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

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:$\{a_0\} \rightarrow$ A
  FK:$\{b_0\} \rightarrow$ 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:$\{ c_0, b_0\} \rightarrow$ C
  NNV:$\{c_{11}\}$

C2 $(c_0: t\_c_0, b_0: t\_b_0)$
  PK:$\{c_0, b_0\}$
  FK:$\{ c_0, b_0\} \rightarrow$ C

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

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: $\{b_2\}$

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

B weak

b_1 multi-valued

$E(a_0{:}t\_a_0, b_0{:}t\_b_0, e_0{:}t\_e_0, e_1{:}t\_e_1)$

    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: $\{a_0\}\rightarrow$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: $\{a_0,\ b_0\}\rightarrow$B

C $(c_0{:}t\_c_0,\ c_1{:}t\_c_1,\ a_0{:}t\_a_0)$

    PK: $\{c_0\}$

    UNI: $\{c_1\}$

S

    FK: $\{a_0\}\rightarrow$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: $\{c_0\}\rightarrow$C

    NNV: $\{c_0\}$

K

    UNI: $\{c_0\}$

    FK: $\{a_0\}\rightarrow$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: $\{a_0,\ b_0\}\rightarrow$B

    NNV: $\{e_1\}$

Y

    FK: $\{d_0\}\rightarrow$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: $\{d_0\}$

    FK: $\{f_0\}\rightarrow$F

    NNV: $\{f_0\}$

T

    NNV: $\{t_0\}$

W $(a_0{:}t\_a_0,\ \mathbf{a_0\_E}{:}t\_a_0,\ \mathbf{b_0\_E}{:}t\_b_0)$

    PK: $\{a_0\_E,\ b_0\_E,\ a_0\}$

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

    FK: $\{\mathbf{a_0\_E,\ b_0\_E}\}\rightarrow$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.

81

A class diagram showing classes A, B, C, D, E, F, G, H, J with associations.

A
- a_0:{id}:t_a_0
- a_1:{0..1}t_a_1

B
- b_0:t_b_0
- b_1:{1..*}: t_b_1
- b_2:{1..1}: t_b_2

C
- c_0:{id}:t_c_0
- c_1:unique:t_c_1
- …

D
- d_0: t_d_0

E
- e_0: {0..1}:t_e_0

H
- h_0: {0..*}:t_h_0

J
- j_0:{1..1}: t_j_0

G
- g_0:{id}:t_g_0
- g_1:{id}:t_g_1
- g_2:{0..1}:t_g_2

F
- f_0:{id}:t_f_0
- f_1:{0..1}:t_f_1

Associations and labels: {id}, R, {0..1}, {Total, Disjoint}, V, S, 0..1, 1..*, 0..*, Y, T, X, Z, U, 1..1, W, {id}

Exercise 7

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

B weak

B_1 multi-valued

$B\_b1(a_0: t\_a_0, b_1: t\_b_1)$
    PK: $\{a_0, b_1\}$
    FK: $\{a_0\} \rightarrow B(a_0)$
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: $\{a_0\} \rightarrow B(a_0)$

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: $\{a_0\} \rightarrow B(a_0)$

E specializes B

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

H specializes B

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

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

83

$A(a_0: t\_a_0, a_1: t\_a_1)$
    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: $\{b_2\}$

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

$F(f_0: t\_f_0, f_1: t\_f_1,$ $\boxed{a_0D: t\_a_0}$, $\boxed{a_0H: t\_a_0}$)
    PK: $\{f_0\}$
    $\boxed{\text{FK: } \{a_0D\} \to D(a_0)}$
    RI: "Every value a_0 in D must be in F"
    $\boxed{\text{PK: } \{a_0H\} \to H(a_0) \quad \text{NNV: } \{a_0H\}}$
$G(g_0: t\_g_0, g_1: t\_g_1, g_2: t\_g_2,$ $\boxed{c_0: t\_c_0}$)
    PK: $\{g_0, g_1\}$
    $\boxed{\text{UNI: } \{c_0\} \quad \text{NNV: } \{c_0\} \\ \text{FK: } \{c_0\} \to C(c_0)}$
$C(c_0: t\_c_0, c_1: t\_c_1, …,$ $\boxed{a_0: t\_a_0}$)
    PK: $\{c_0\}$
    UNI: $\{c_1\}$
    $\boxed{\text{FK: } \{a_0\} \to A \\ \text{RI: Every value } a_0 \text{ in A must be in C"}}$
$J(j_0: t\_j_0, a_0: t\_a_0, g_0: t\_g_0, g_1: t\_g_1)$
    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: $\{f_0\} \to F$

T   U   X   S

$B\_b1(a_0: t\_a_0, b_1: t\_b_1)$
    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

A

a_0:{id}:t_a_0
a_1:{1..*}:t_a_1
...

{id} — R — 0..* — **B**

b_0:{id}:t_b_0
b_1:{1..1}:t_b_1
...

{Total, Disjoint }

D

d_0: {único}:{0..1}:t_d_0
...

E

e_0: {0..1}:t_e_0
...

s_1 — 0..*

C

c_0:{id}:t_c_0
c_1:{único}:{0..1}:t_c_1
...

S

s_2 — 0..*

1..1

X

0..1

G

g_0:{id}:t_g_0
g_1:{id}:t_g_1
...

0..*

Z

z_0:{1..1}:t_z_0

0..1

1..1

T

t_0: {0..1}: t_t_0

0..*

F

f_0:{id}:t_f_0
f_1:{1..1}:t_f_1
...

85

Exercise 8

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

$B(b_0:t\_b_0, b_1:t\_b_1, a_0:t\_a_0)$
   PK: $\{b_0, a_0\}$
   FK: $\{a_0\} \rightarrow A$ B weak
   NNV: $\{b_1\}$ a_1 multi-valued

$A1(a_0:t\_a_0, a_1:t\_a_1)$
   PK: $\{a_0, a_1\}$
   FK: $\{a_0, a_1\} \rightarrow A$

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

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

D & E specializes B

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

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

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

All a_0 in A must be in a_0 of A1

RI_{Total}: ""Every $\{a_0, b_0\}$ in B, must also be in $\{a_0, b_0\}$ of D or E".
$RI_{Disjoint}$: "A pair $\{a_0, b_0\}$ in B cannot be in8 more than one tuple of D or E".

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

$A1(a_0{:}t\_a_0, a_1{:}t\_a_1)$
  PK:$\{a0, a1\}$
  FK:$\{a0\} \rightarrow A$

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

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

$G(g_0{:}t\_g_0, g_1{:}t\_g_1\ c_0{:}t\_c_0)$
  PK: $\{g0, g1\}$
  FK: $\{c0\}{\rightarrow}C$
  NNV: $\{c0\}$
  Uni: $\{c0\}$

$S(c0\_s1{:}t\_c0, c0\_s2{:}t\_c0)$
  PK:$\{c0\_s1, c0\_s2\}$
  FK:$\{c0\_s1\} \rightarrow C(c0)$
  FK:$\{c0\_s2\} \rightarrow C(c0)$

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

$F(f_0{:}t\_f_0, f_1{:}t\_f_1, a_0{:}t\_a_0, b_0{:}t\_b_0, t_0{:}t\_t_0)$
  PK: $\{f_0\}$
  NNV: $\{f_1\}$
  FK: $\{a_0, b_0\}{\rightarrow}E$
  NNV: $\{a_0, b_0\}$
  Uni: $\{a_0, b_0\}$

$D(a_0{:}t\_a_0, b_0{:}t\_b_0, d_0{:}t\_d_0)$
  PK: $\{a0, b0\}$
  FK: $\{a0, b0\}{\rightarrow}B$
  Uni: $\{d0\}$

$Z(a0{:}t\_a0, b0{:}t\_b0, g0{:}t\_g0, g1{:}t\_g1, z0{:}t\_z0)$
  PK$\{a0, b0\}$
  NNV:$\{z0\}$
  FK:$\{g0, g1\} \rightarrow G$
  FK:$\{a0, b0\} \rightarrow D$
  NNV:$\{g0, g1\}$

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**

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 study 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[1]"*) 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

[1]X is the determinant attribute.

90

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 }
…

91

Full functional dependency

An Functional Dependency X $\to$ 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 } $\to$ { name }

{ ISBN } $\to$ { title }

{ ssn, ISBN } $\to$ { euros }

Functional dependencies Diagram

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

In a Functional Dependency Diagram each attribute is shown in a rectangle, and an arrow indicates the direction of the dependency between two attributes.

We are going to represent only full functional dependencies.

Example 3:
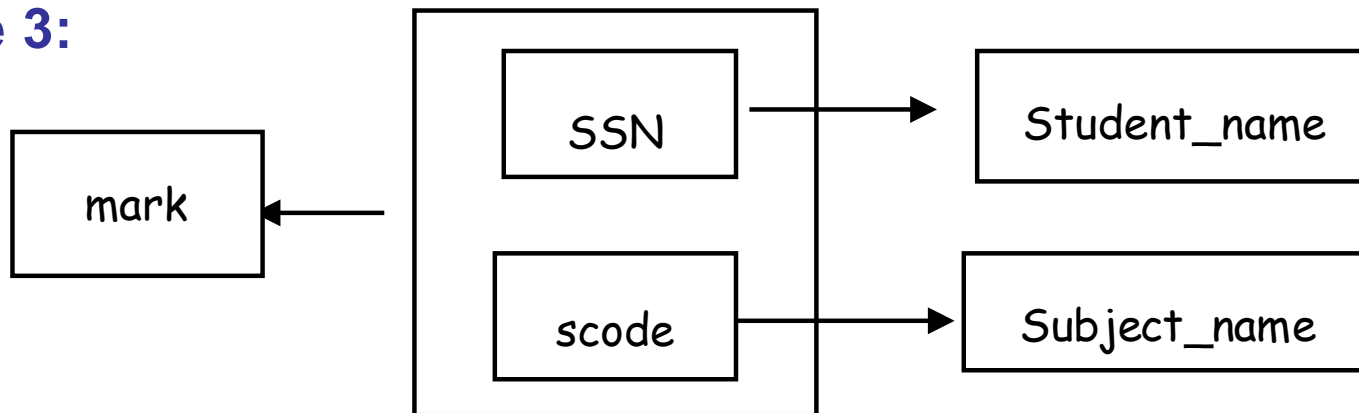
Key of a relation

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} $\rightarrow$ { name }
{ssn, ISBN} $\rightarrow$ { title }
{ssn, ISBN} $\rightarrow$ { euros }
{ssn, ISBN} $\rightarrow$ { 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:

Set

Record

Provider   PK: {vcod}

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:

⇩
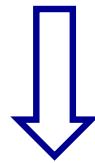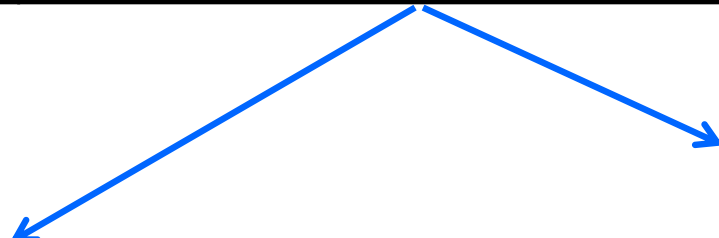
1. <u>Remove</u> the attribute from the relation and <u>define a new relation</u> with the attribute and the primary key of R.

2. Analyze the functional dependencies of the new relation to define its <u>primary key</u>.

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

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} $\rightarrow$ 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
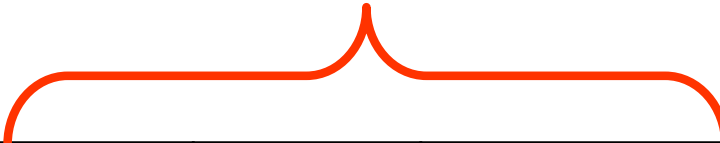
R has a composite attribute (a record).

⇩

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} $\rightarrow$ 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, scode}

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
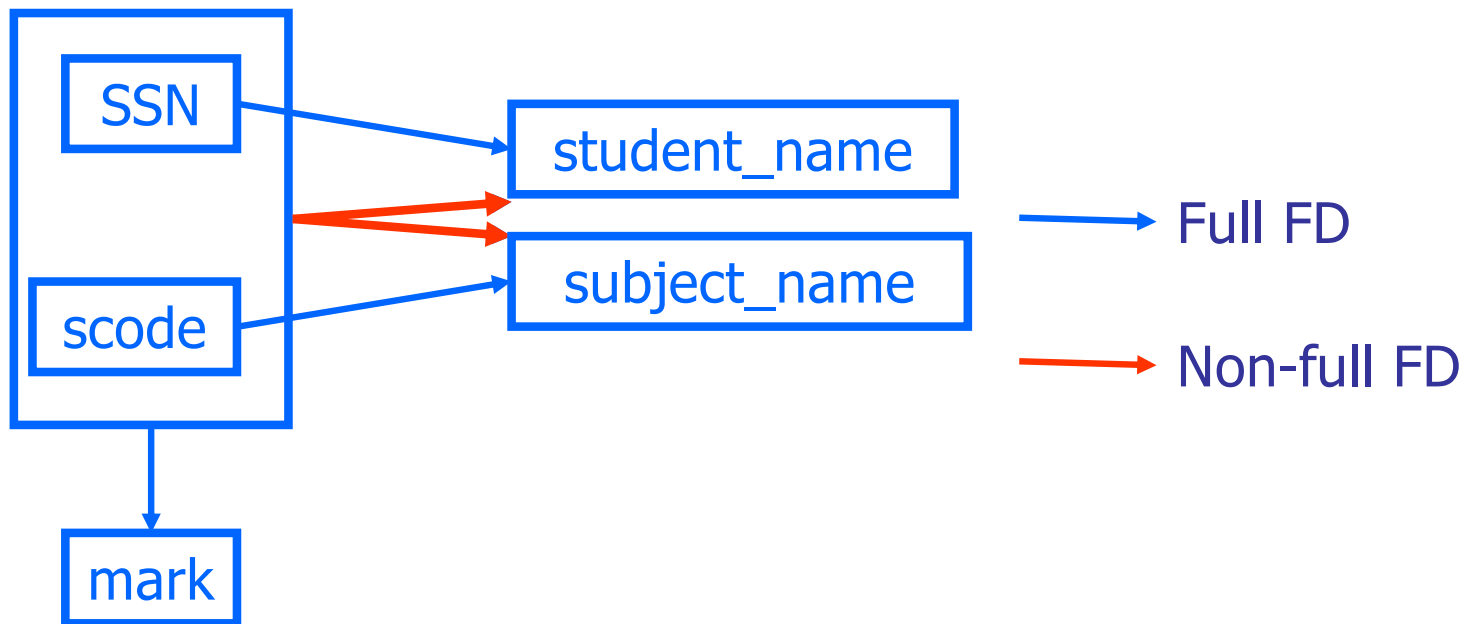
2nd NF Transformation
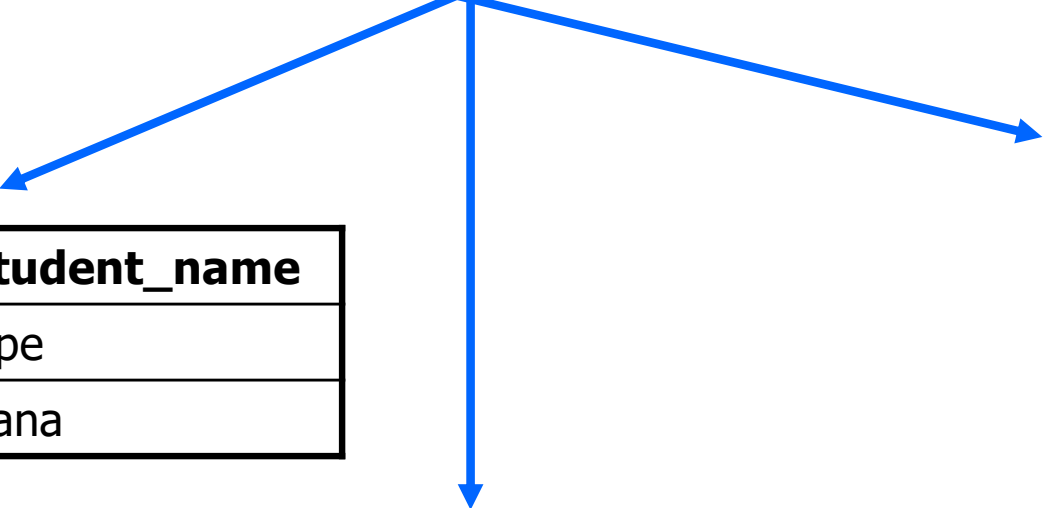
The primary key has more than one attribute and there is some non-prime attribute which does not fully functionally depend on the primary key

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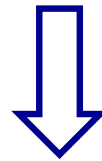
Divide the relation in several relations to remove the not fully functional dependency

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

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

scode	subject_name
DBD	Diseño de BD
BDA	Bases de Datos

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} $\rightarrow$ Student

FK: {scode} $\rightarrow$ 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} $\rightarrow$ {name}

       {ISBN} $\rightarrow$ {title}

2nd NF:

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

    PK: {ssn, ISBN}   FK:{ssn} -> Author   FK:{ISBN} -> Book

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

$A = \{A_1, A_2, \ldots, A_n\}$ is the set of attributes of R,

If $\{A_i\} \rightarrow \{A_j\}$ and $\{A_j\} \rightarrow \{A_k\}$ then $\{A_i\} \rightarrow \{A_k\}$ 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}

FD between non-prime attributes
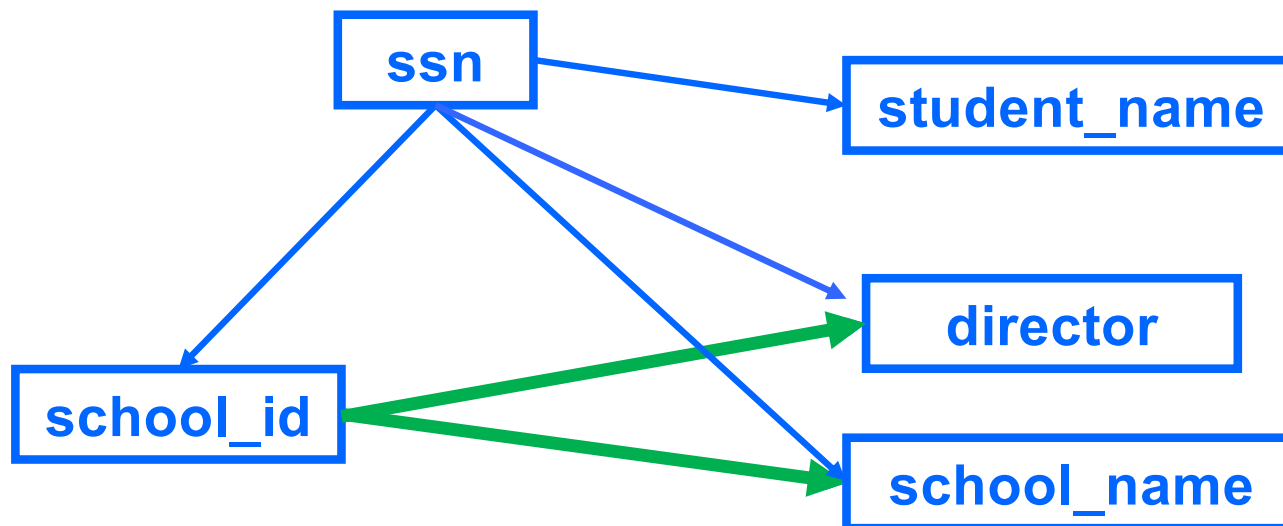
3rd NF Transformation

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

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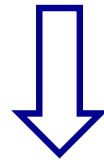
Remove the dependent attribute and create a new table with it and the determinant attribute. The PK of the new table will be the determinant attribute

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} $\rightarrow$ 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_number}
  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**:
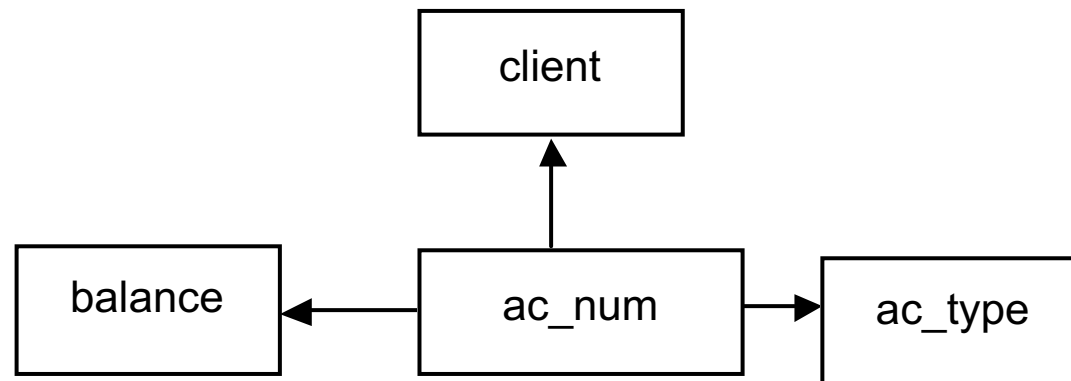
3rd NF. Example 6

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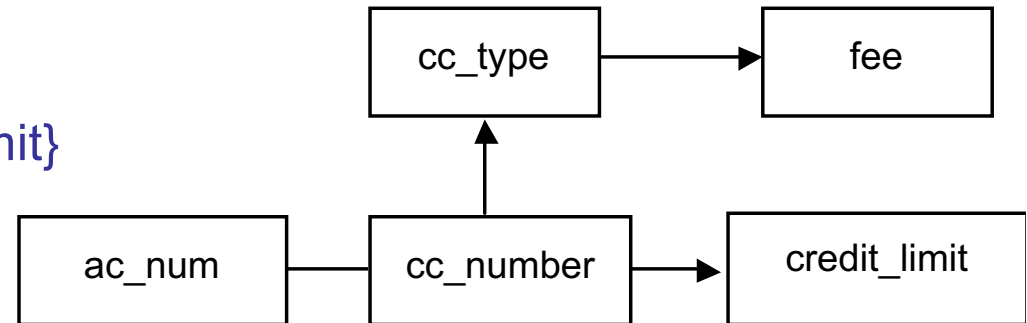
It is not in 3NF, because there is a FD between non-prime attributes

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}

$\{G\} \rightarrow \{E\}$ $\{G\} \rightarrow \{F\}$ $\{A\} \rightarrow \{D\}$ $\{A\} \rightarrow \{G\}$

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

2FN

$\{G\} \rightarrow \{E\}$ $\{G\} \rightarrow \{F\}$ **{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.

116

Exercise N1

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

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

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

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} -> R31
  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.

117

Exercise N2

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}

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

Exercise N2

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

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

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

- 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 pairs of values of {A.C} 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.

Exercise N3

Consider the following relational schema:

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

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

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

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

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

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

R21 (A: char, D: char)
  PK: {A}   NNV: {D}

R22 (B: char, C: int, E: int, F: char)
  PK: {B,C}   NNV: {E, F}

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

122

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

R21 (**A**: char, **D**: char)
PK: {A}   NNV: {D}

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

R22 (**B**: char, **C**: int, **E**: int, **F:** char)
PK: {B,C}   NNV: {E, F}

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

R1 (A: char, B: char, C: int, G: int)
PK: {A, B, C}   NNV: {G}
FK:{A} → R21(A)   FK:{B,C} → R22(B,C)
FK:{G} → R32(A)

R31 (E: int, F: char)
PK: {E}   NNV: {F}

R21 (A: char, D: char)
PK: {A}   NNV: {D}

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

R22 (B: char, C: int, E: int)
PK: {B,C}   NNV: {E}
FK:{E} → R31(E)

All the values of {A} in R21 are also in R1.
All the pairs of values of {B,C} in R22 are also in R1.
All the values of {E} in R31 must be in {E} of R22
All the values of {G} in R32 must be in {G} of R1

Exercise N4

Consider the following relational schema:
R (**A**: *int*, **B**: *char*, **C**: *char*, **D**: *set of integers*, **E**: *int*, **F**: *text*, **G**: *int*, **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).

{B} → {G}   {A, B} → {F}   {G} → {H}

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

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

R (A: int, B: char, C: char, E: int)
  PK: {A, B, C}
  FK: {A, B} → R3
  FK: {B} → R2
  NNV: {E}

R1 (A: int, B: char, C: char, D: int)
  PK: {A, B, C, D}
  FK: {A, B, C} → R

R2 (B: char, G: int)
  PK: {B}
  FK: {G} → R22
  NNV: {G}

R3 (A: int, B: char, F: text)
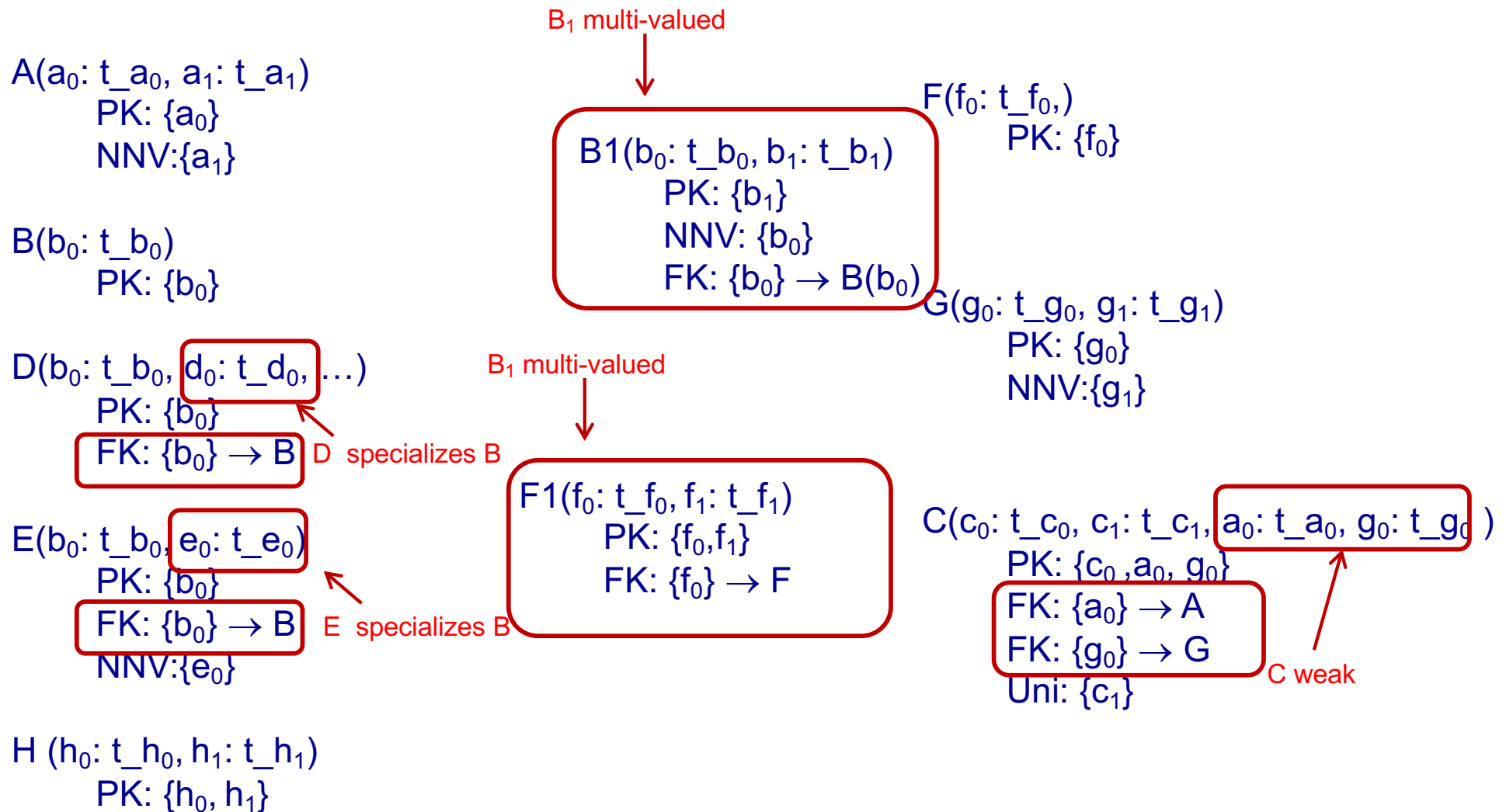 PK: {A,B}
  NNV: {F}

R22 (G: int, H: int)
  PK: {G}
  NNV: {H}

Exercise 9
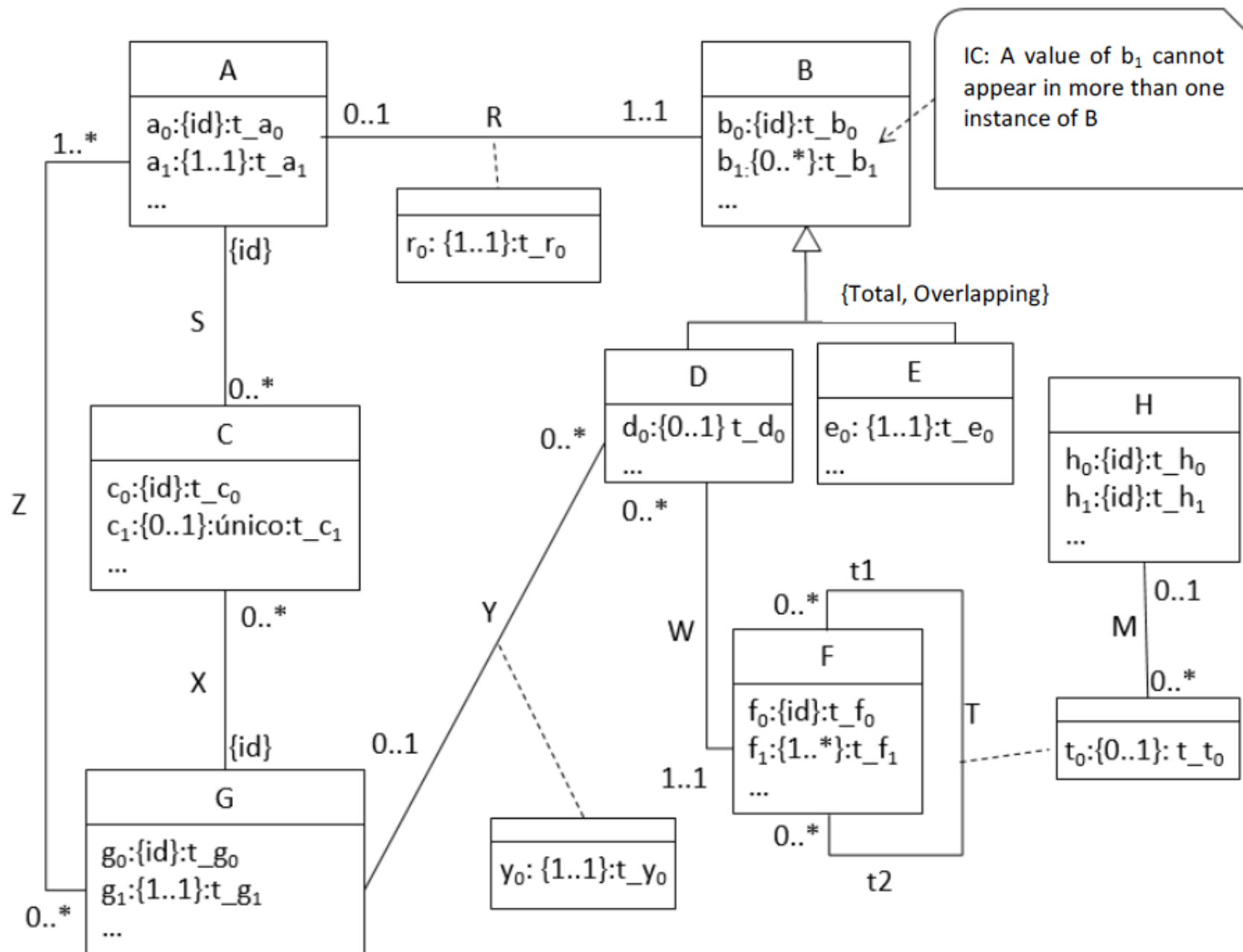
B1 multi-valued

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

$B(b_0: t\_b_0)$
　　　PK: $\{b_0\}$

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

$E(b_0: t\_b_0, e_0: t\_e_0)$
　　　PK: $\{b_0\}$
　　　FK: $\{b_0\} \rightarrow B$　　E specializes B
　　　NNV: $\{e_0\}$

$H(h_0: t\_h_0, h_1: t\_h_1)$
　　　PK: $\{h_0, h_1\}$

$B1(b_0: t\_b_0, b_1: t\_b_1)$
　　　PK: $\{b_1\}$
　　　NNV: $\{b_0\}$
　　　FK: $\{b_0\} \rightarrow B(b_0)$

B1 multi-valued

$F1(f_0: t\_f_0, f_1: t\_f_1)$
　　　PK: $\{f_0, f_1\}$
　　　FK: $\{f_0\} \rightarrow F$

$F(f_0: t\_f_0,)$
　　　PK: $\{f_0\}$

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

$C(c_0: t\_c_0, c_1: t\_c_1, a_0: t\_a_0, g_0: t\_g_0)$
　　　PK: $\{c_0, a_0, g_0\}$
　　　FK: $\{a_0\} \rightarrow A$
　　　FK: $\{g_0\} \rightarrow G$　　C weak
　　　Uni: $\{c_1\}$

RI$_{Total}$: ""Every b_0 in *B,* must also be in b_0 of D or E".

RImin_f1: Every value in f_0 of F must appear in f_0 of F1.

127

$A(a_0: t\_a_0, a_1: t\_a_1, b_0: t\_b_0, r_0: t\_b_0)$
PK: $\{a_0\}$
NNV:$\{a_1, b_0, r_0\}$
FK: $\{b_0\} \rightarrow B$
UNI:$\{b0\}$

R

$B1(b_0: t\_b_0, b_1: t\_b_1)$
PK: $\{b_1\}$
NNV: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$

$Z(a_0: t\_a_0, g_o: t\_g_0)$
PK: $\{a_0, g_0\}$
FK: $\{a_0\} \rightarrow A$
FK: $\{g_0\} \rightarrow G$

$C(c_0: t\_c_0, c_1: t\_c_1, a_0: t\_a_0, g_0: t\_g_0)$
PK: $\{c_0, a_0, g_0\}$
FK: $\{a_0\} \rightarrow A$
FK: $\{g_0\} \rightarrow G$
Uni: $\{c_1\}$

$B(b_0: t\_b_0)$
PK: $\{b_0\}$

$D(b_0: t\_b_0, d_0: t\_d_0, f_0: t\_f_0)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$
FK: $\{f_0\} \rightarrow F$
NNV:$\{f_0\}$

W

$T(f_{0\_t1}: t\_f_0, f_{o\_t2}: t\_f_0, t_o: t\_t_0, h_o: t\_h_0, h_1: t\_h_1)$
PK: $\{f_{0\_t1}, f_{o\_t2}\}$
FK: $\{f_{0\_t1}\} \rightarrow F(f_0)$
FK: $\{f_{0\_t0}\} \rightarrow F(f_0)$
FK: $\{h_{\_t0}, h_{\_t0}\} \rightarrow H(h_0, h_1)$

M

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

$E(b_0: t\_b_0, e_0: t\_e_0)$
PK: $\{b_0\}$
FK: $\{b_0\} \rightarrow B$
NNV:$\{e_0\}$

$F1(f_0: t\_f_0, f_1: t\_f_1)$
PK: $\{f_0, f_1\}$
FK: $\{f_0\} \rightarrow F$

$Y(g_0:t\_g_0, b_0:t\_b_0, y_0:t\_y_0)$
PK:$\{b_0\}$
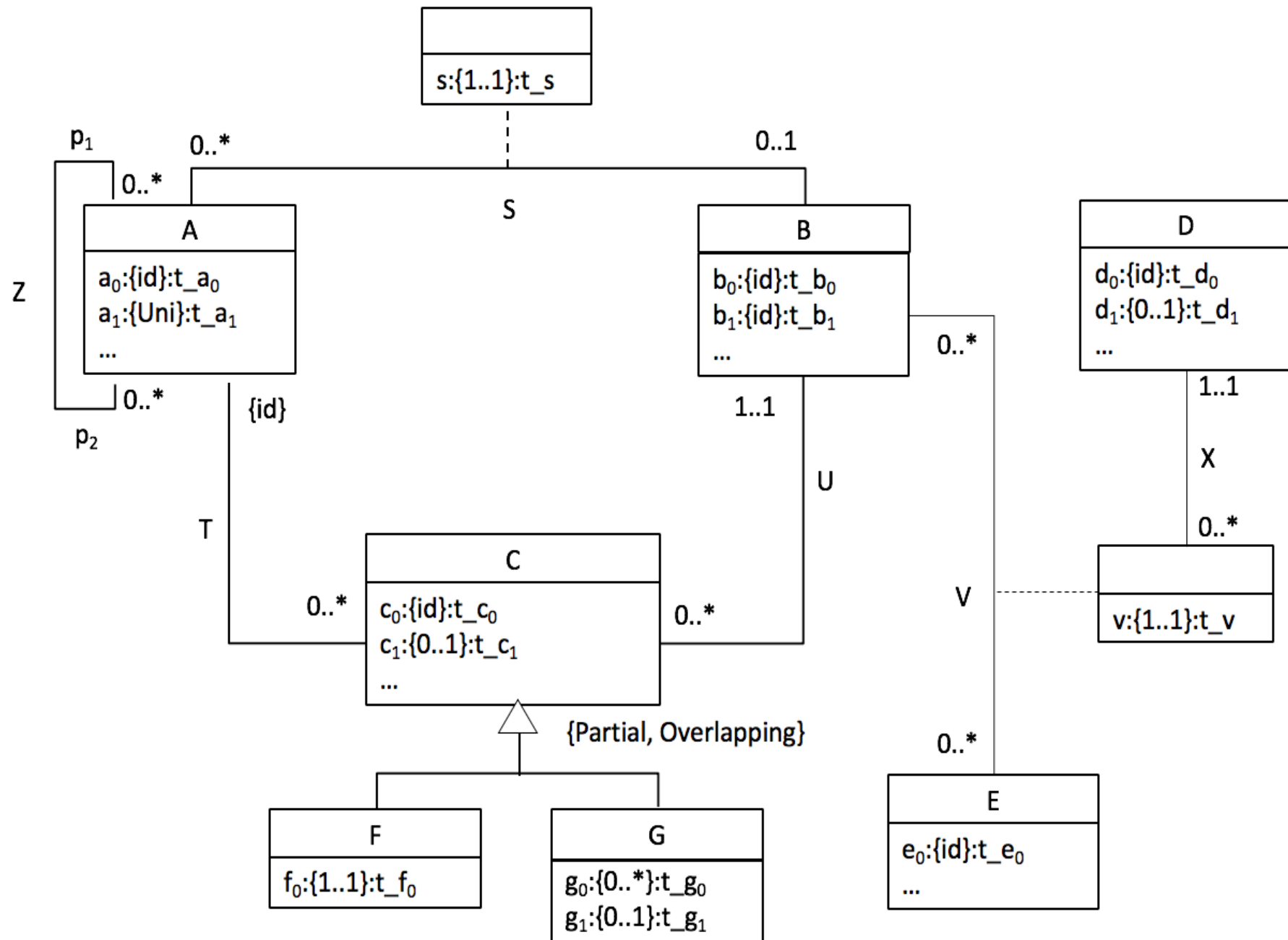FK:$\{g_0\} \rightarrow G$
FK:$\{b_0\} \rightarrow D$
NNV:$\{g_0, y_0\}$

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

$H (h_0: t\_h_0, h_1: t\_h_1)$
PK: $\{h_0, h_1\}$

RI$_{Total}$: ""Every b_0 in *B,* must also be in b_0 of D or E"   **RImin_f1:** Every value in f_0 of F must appear in f_0 of F1.
RImin_G_in_Z: Every value in g_0 of G must appear in g_0 of Z.

A (a_0:t_a_0, a_1:t_a_1 ,…

    PK:{a_0}

    Uni:{a_1}

B (b_0:t_b_0, b_1:t_b_1 ,…

    PK:{b_0, b_1}

D (d_0:t_d_0, d_1:t_d_1 ,…

    PK:{d_0}

E (e_0:t_e_0 ,…

    PK:{e_0}

F (c_0: t_c_0, a_0: t_a_0 , f_0: t_f_0

    PK:{c_0, a_0}

    FK:{c_0, a_0} $\rightarrow$ C(c_0, a_0)

    NNV:{f_0}

G (c_0: t_c_0, a_0: t_a_0 , g_1: t_g_1

    PK:{c_0, a_0}

    FK:{c_0, a_0} $\rightarrow$ C(c_0, a_0)

G0 (c_0: t_c_0, a_0: t_a_0 , g_0: t_g_0

    PK:{c_0, a_0 , g_0}

    FK:{c_0, a_0} $\rightarrow$ G(c_0, a_0)

C (c_0: t_c_0, a_0: t_a_0 , c_1: t_c_1,

    PK:{c_0, a_0}

    FK:{a_0} $\rightarrow$ A

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

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

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

E (e_0:t_e_0 ,…)
   PK:{e_0}

F (c_0: t_c_0, a_0: t_a_0 , f_0: t_f_0)
   PK:{c_0, a_0}
   FK:{c_0, a_0} $\rightarrow$ C(c_0, a_0)
   NNV:{f_0}

G (c_0: t_c_0, a_0: t_a_0 , g_1: t_g_1)
   PK:{c_0, a_0}
   FK:{c_0, a_0} $\rightarrow$ C(c_0, a_0)

G0 (c_0: t_c_0, a_0: t_a_0 , g_0: t_g_0)
   PK:{c_0, a_0 , g_0}
   FK:{c_0, a_0} $\rightarrow$ G(c_0, a_0)

Z (a_0_p1:t_a_0, a_0_p2:t_a_0)
   PK:{a_0_p1, a_0_p2}
   FK:{a_0_p1} $\rightarrow$ A(a_0)
   FK:{a_0 _p2} $\rightarrow$ A(a_0)

C (c_0: t_c_0, a_0: t_a_0 , c_1: t_c_1, b_0:t_b_0 b_1:t_b_1 ,…)
   PK:{c_0, a_0}
   NNV:{b_0, b_1}
   FK:{a_0} $\rightarrow$ A
   FK:{b_0,b_1} $\rightarrow$ B(b_0, b_1)   **U**

S (a_0: t_a_0, b_0: t_b_0 , b_1: t_b_1 , s: t_s)
   PK:{a_0}
   NNV:{b_0, b_1, s}
   FK:{a_0} $\rightarrow$ A
   FK:{b_0 ,b_1} $\rightarrow$ B(b_0, b_1)

V (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, b_1, e_0}   **V**
   NNV(d_0, v)
   FK:{b_0,b_1} $\rightarrow$ B(B_0, b_1)
   FK:{e_0} $\rightarrow$ E   **X**
   FK:{d_0} $\rightarrow$ D

131