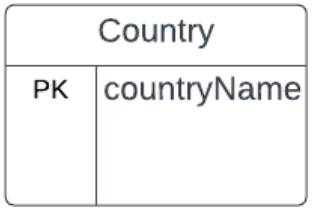
Modelling issues

course: Information Modelling

Relations with only identifying attributes

If we want a relation with all possible dates / names of country (where you can choose a valid date / country) then the **relation must be kept.**

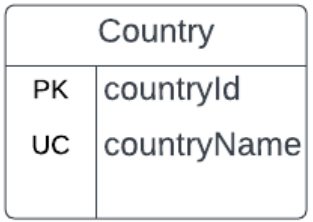
If we want a relation with the used dates / names of country then it's possible to **drop the relation** (dates / country names can be retrieved by querying the corresponding relations).



Relations with only identifying attributes

Nevertheless, it’s better that relation Country has a meaningless primary key. As country name should be unique add a uniqueness constraint.

The relation Calendar is clearly redundant.



Standard value and exceptional value

- All articles are delivered on the same date

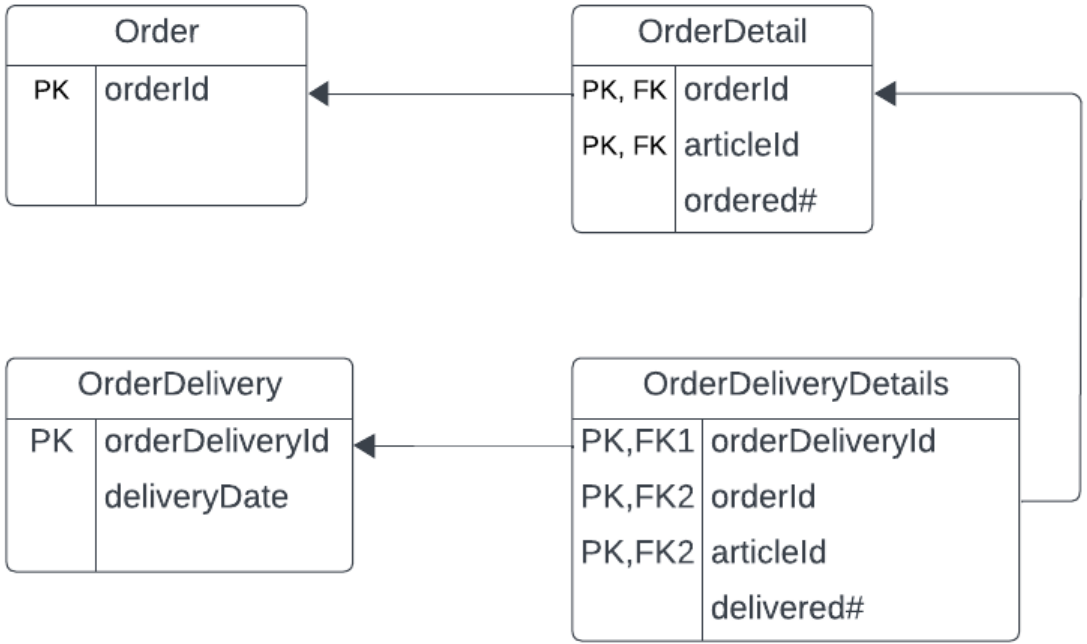
- Mostly all articles are delivered on the same date but exceptions occur. If the deliveryDate of an order detail is not null then the exceptional delivery date is in charge.

Standard value and exceptional value

- If it’s exceptional that articles of the same order are delivered on the same date, then a deliveryDate for each order detail might be a good idea

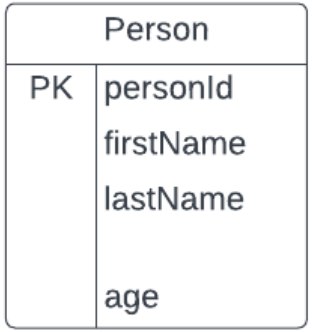
Standard value and exceptional value

- A better (conceptual) modelling can help too..

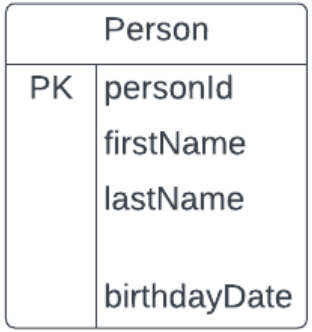


Time dependent attribute

- Value of attribute changes constantly over time (each day the age of some entities has to increase)

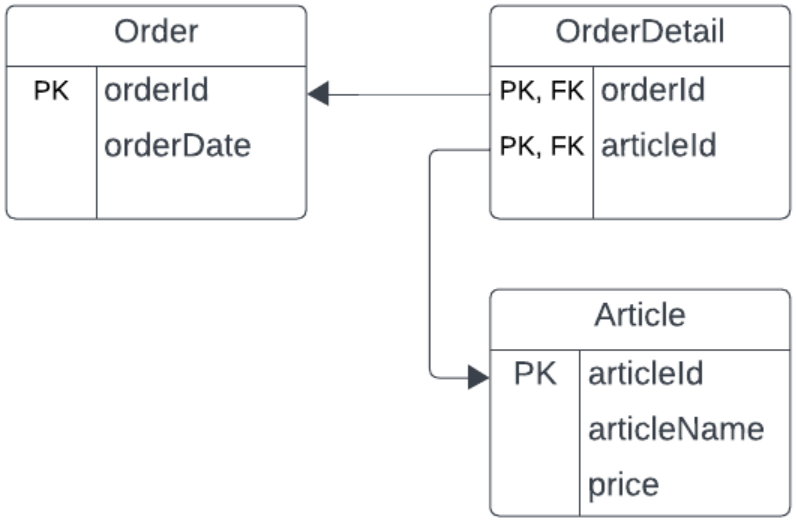


Choose an attribute, independent of the current date/time and calculate the value in the same way as derived attributes



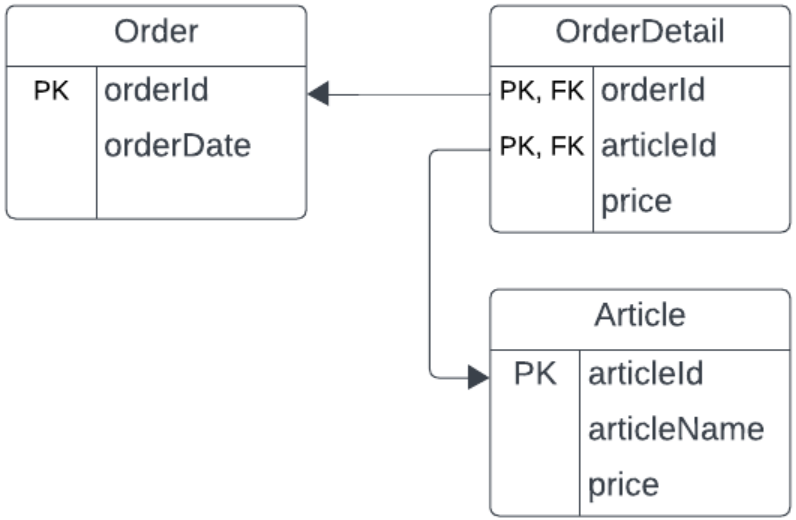
Time dependent attribute

- Price can change over time
  so if you collect the information about the price of an article, you might fetch an old price



Time dependent attribute

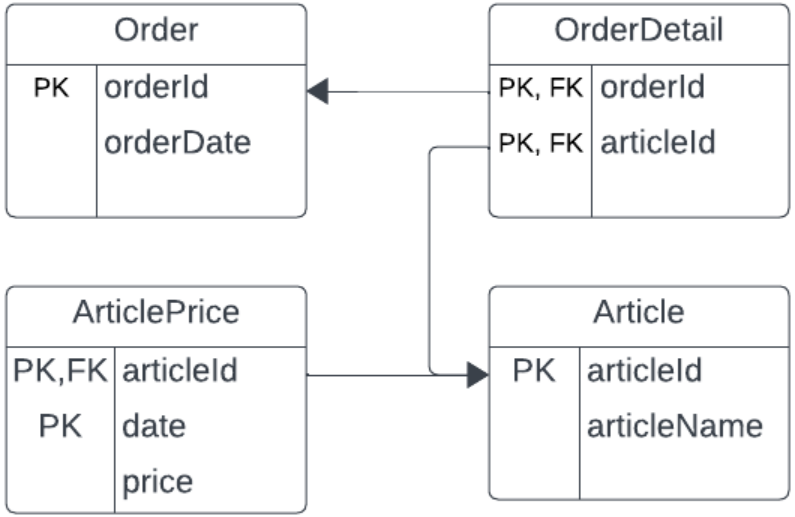
- You can duplicate the price in the order detail
  It's a must if price can be negotiated and differs for each order.



Time dependent attribute

- A better (conceptual) modelling can help too... keep a **history** of prices: **daily or interval** (date is the startdate and the price is valid until a new price is registered)

  In case of an interval the retrieval of the correct price can be programmed in a stored function.



Fixed number of attributes

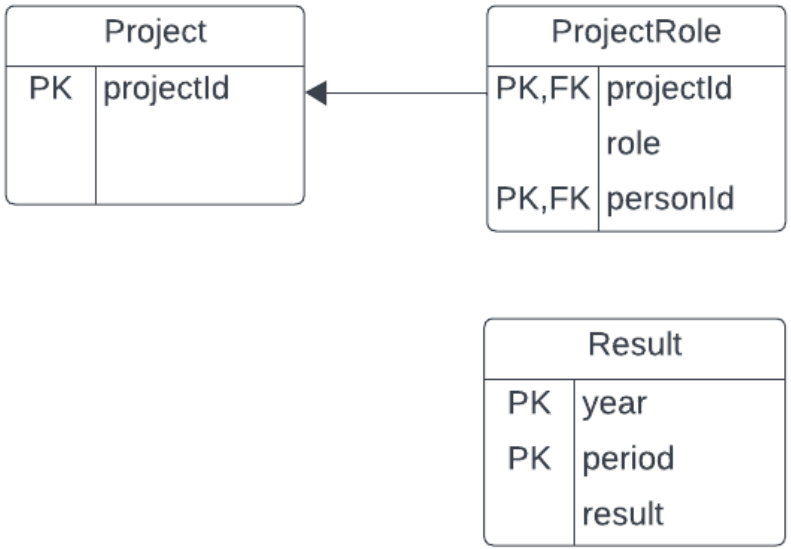
- Several attributes refer to the same domain and contains values with the same semantics, but have a different role/dimension.

Project	
PK	projectId
FK	head
FK	analist
FK	developer1
FK	developer2

Result	
PK	year
	q1_result
	q2_result
	q3_result
	q4_result

Fixed number of attributes

- Create an attribute for the role/dimension and find an attribute to replace the fixed number of attributes. Eventually a new relation is needed. The attribute (or another one) will be part of the key.



Normalisation

Functional dependency

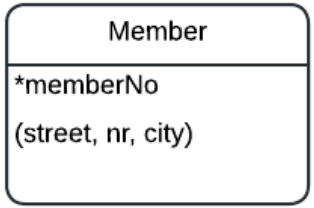
gameId -> name

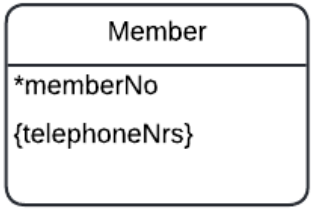
gameId is called **determinant** of the functional dependency

Normalisation 1NF

- 1NF: all attributes should be atomic – all attributes should be functional dependent of the key

  no composite attributes /
  no multivalued attributes





Composite Attributes vs 1NF

Composite attribute

An attribute can hold a

composition of values

and each part has a meaning

1st Normal Form

Each attribute should contain

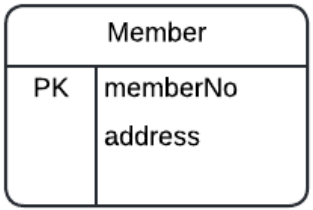
atomic values

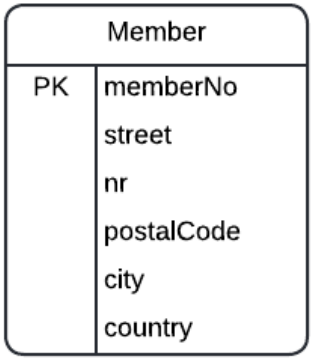
If you run across a composite attribute, this is a major hint that you need another entity

but you may use composite attributes sparsely

Composite Attributes vs 1NF

- split the composite attributes in the original relation





Sometimes the attributes will later be grouped into an new relation (occurs in 2NF or 3NF)

Multivalued Attributes vs 1NF

Multivalued attribute

An attribute can hold

multiple values

1st Normal Form

Each attribute should contain

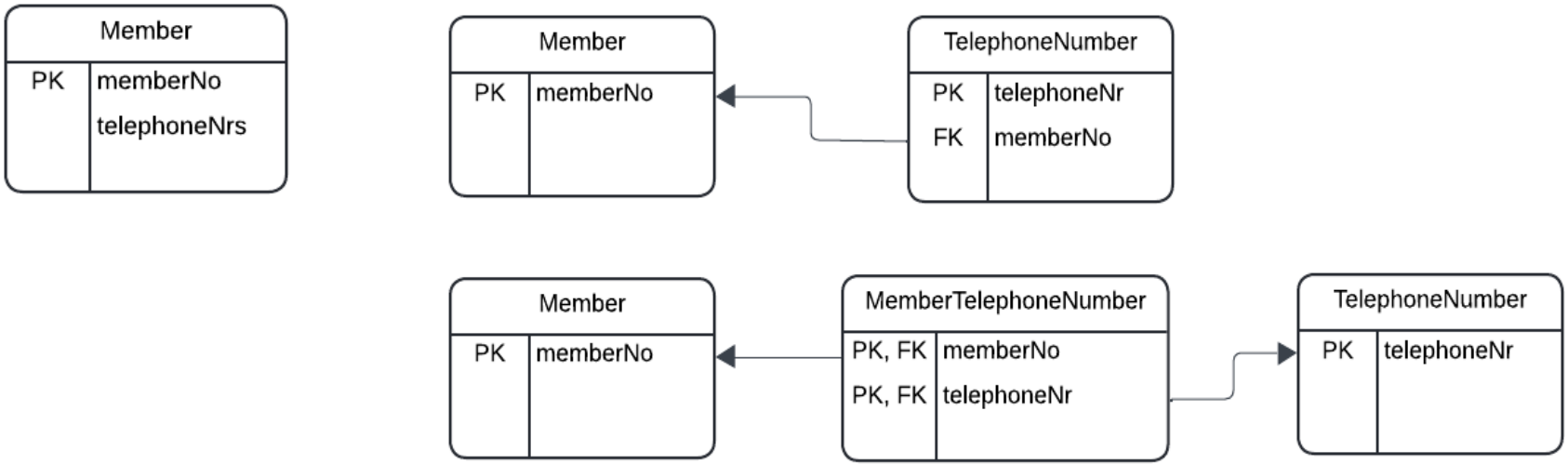
atomic values

If you run across a multivalued attribute, this is a major hint that you need an extra entity type

but you may use multivalued attributes sparsely

Normalisation 1NF

- Split the multivalued attributes into a new relation



Normalisation 2NF

- 2NF: 1NF + all non-key attributes should be **fully** functional dependent of the key(s) - **no partial dependencies**

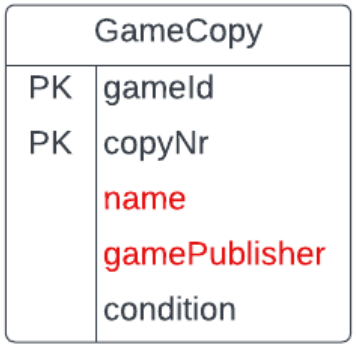
  gameId, copyNr -> name, gamePublisher, condition
  gameId -> name, gamePublisher
  (= partial dependency in GameCopy)

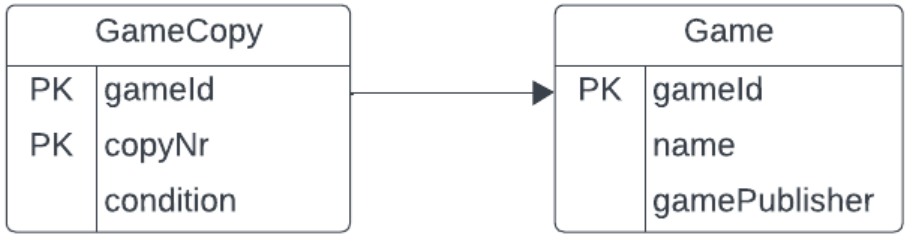
GameCopy	
PK	gameId
PK	copyNr
	name
	gamePublisher
	condition

Normalisation 2NF

- Split the attributes into a new relation with the partial key as key

gameId, copyNr -> condition
gameId -> name, gamePublisher

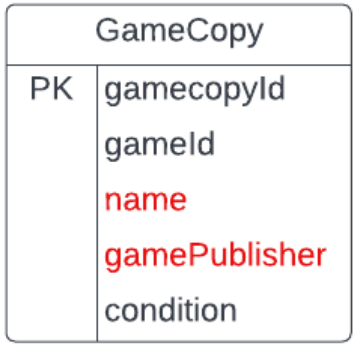




Normalisation 3NF

- 3NF: 1NF + all non-key attributes should be only dependent of the key(s) - **no transitive dependencies**

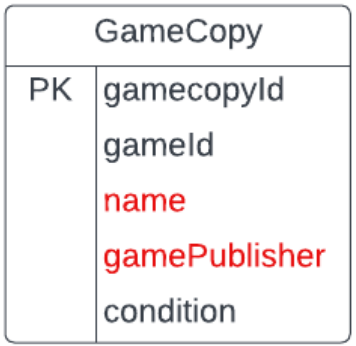
gamecopyId -> gameId, name, gamePublisher, condition
gameId -> name, gamePublisher
(= transitive dependency in GameCopy)

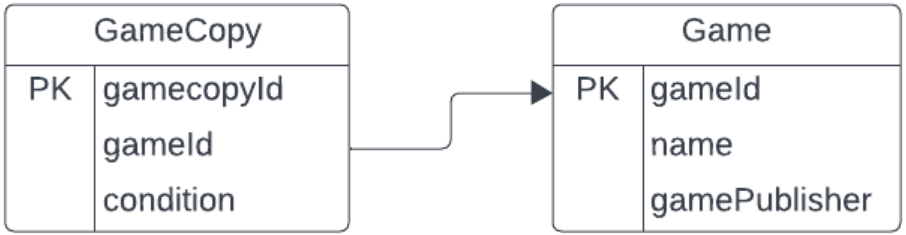


Normalisation 3NF

- Split the attributes into a new relation with the transitive dependency attribute as key

gamecopyId -> gameId, condition
gameId -> name, gamePublisher





Normalisation BCNF

- BCNF: 1NF + every determinant should be a key

  It's a rare situation, but some relations are in 3NF but not in BCNF. This can occur if the relation has overlapping keys and a part of a key is functional dependant on another part of a key

  All keys should be fully dependent of only keys - **no partial dependencies between keys**

Normalisation BCNF

- Let’s analyse a situation of client interviews with the following rules
  - ↪ A client can have only one interview a day
  - ↪ A room can only be booked once on a day and specific timeslot (no double bookings)
  - ↪ A staff member can only attend one interview at once on a day and timeslot
  - ↪ A staff member is available in the same room the whole day

ClientInterview	
PK	clientNo
PK,UC1,UC2	interviewDate
UC1,UC2	interviewTime
UC1	staffNo
UC2	roomNo

Normalisation BCNF

- **Functional dependancies:**
  clientNo, interviewDate -> interviewTime, staffNo, roomNo
  interviewDate, interviewTime, staffNo -> clientNo, roomNo
  interviewDate, interviewTime, roomNo -> clientNo, staffNo

  interviewDate, staffNo -> roomNo
  interviewDate, staffNo is not a key!

ClientInterview	
PK	clientNo
PK,UC1,UC2	interviewDate
UC1,UC2	interviewTime
UC1	staffNo
UC2	roomNo

Normalisation BCNF

- Split the attributes into a new relation with the determinant of the partial dependency as key

We loose UC2!
interviewDate, interviewTime, roomNo

On a date and time a room can be double booked!

Interview	
PK	clientNo
PK,UC1	interviewDate
UC1	interviewTime
UC1	staffNo

StafRoom	
PK	staffNo
PK	interviewDate
	roomNo

Normalisation BCNF

- Split the attributes into a new relation with the determinant of the partial dependency as key

Solve with view joining the tables and uniqueness constraint (not in every dbms available) or solve with database trigger

Interview	
PK	clientNo
PK,UC1	interviewDate
UC1	interviewTime
UC1	staffNo

StafRoom	
PK	staffNo
PK	interviewDate
	roomNo

Normalisation 4F

– 4NF: 1NF + no **multi-valued dependencies** between key parts

Example:
teachers give courses and use materials

CourseOrganisation	
PK	teacherId
PK	classId
PK	materialId

OR

CourseMaterial	
PK	courseId
PK	materialId

CourseTeacher	
PK	courseId
PK	teacherId

?

Normalisation 4F

Is the used material in every same course equal?
Yes? Then there is a **redundancy** based on the **multi-valued dependencies**

courseId ->-> materialId and **courseId ->-> teacherId**

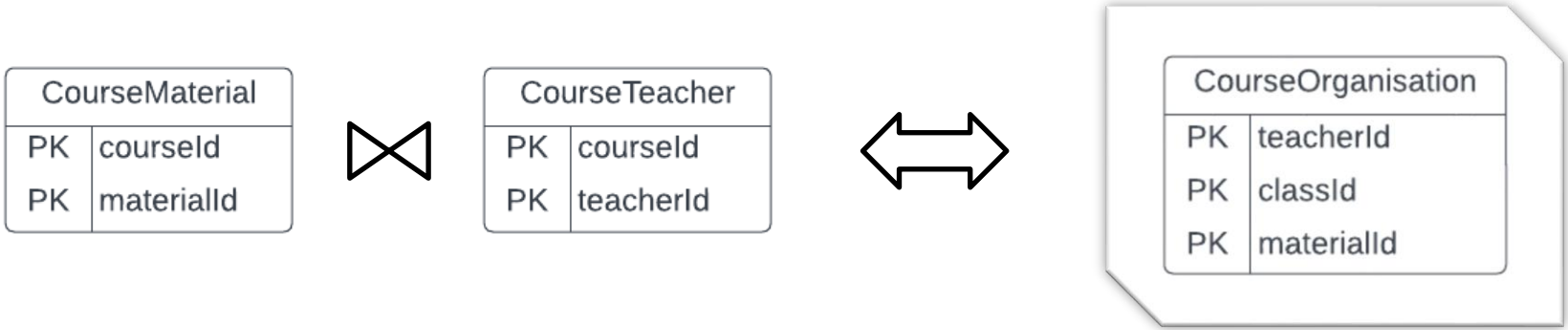
and the relation **should be splitted** based on the multi-valued dependencies

CourseMaterial	
PK	courseId
PK	materialId

CourseTeacher	
PK	courseId
PK	teacherId

Normalisation 4F

The original relation can be reconstructed through a join



This is called a **lossless decomposition**

Normalisation 4F

Course	Teacher	Materials
DB	Roels	db-book
DB	Roels	w3schools
DB	Sourie	db-book
DB	Sourie	w3schools
Java	Vlummens	j4profs
Java	De Wael	j4profs
…	…	…

Normalisation 4F

Course	Teacher
DB	Roels
DB	Sourie
Java	Vlummens
Java	De Wael

Course	Material
DB	db-book
DB	w3schools
Java	j4profs

Normalisation 4F

Course	Teacher	Materials
DB	Roels	db-book
DB	Roels	w3schools
DB	Sourie	db-book
DB	Sourie	w3schools
Java	Vlummens	j4profs
Java	De Wael	j4profs
…	…	…

Normalisation 4F

Is the used material in every same course equal?
No? Then there is **no redundancy** and **the multi-valued dependencies**

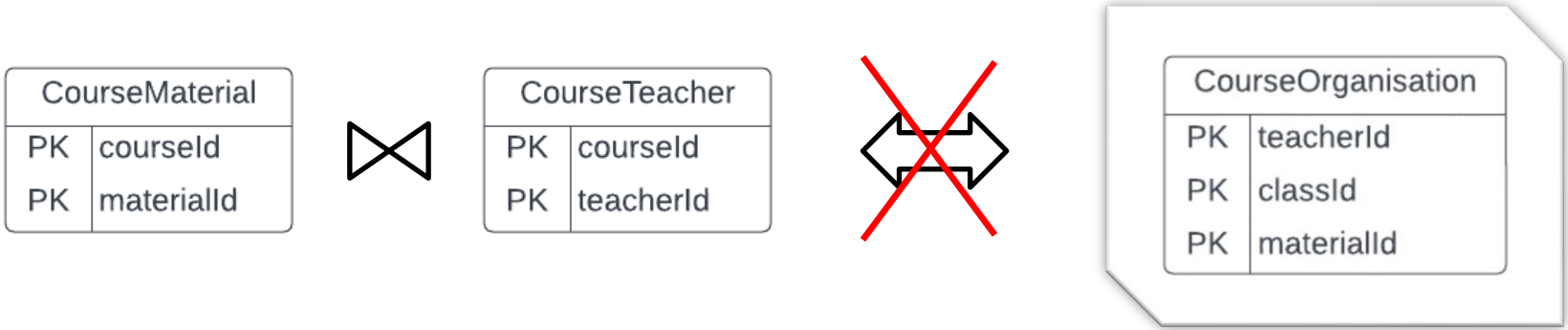
courseId ->-> materialId and **courseId ->-> teacherId**

don’t exist and the relation **should not be splitted**

CourseOrganisation	
PK	teacherId
PK	classId
PK	materialId

Normalisation 4F

The original relation can not be reconstructed through a join because of loss of information



This is called a **lossy decomposition**

Normalisation 4F

Course	Teacher	Materials
DB	Roels	db-book
DB	Roels	w3schools
DB	Sourie	owncourse
DB	Sourie	w3schools
Java	Vlummens	j4profs
…	…	…

Normalisation 4F

Course	Teacher
DB	Roels
DB	Sourie
Java	Vlummens
Java	De Wael
…	…

⋈

Course	Material
DB	db-book
DB	w3schools
DB	owncourse
Java	j4profs
…	…

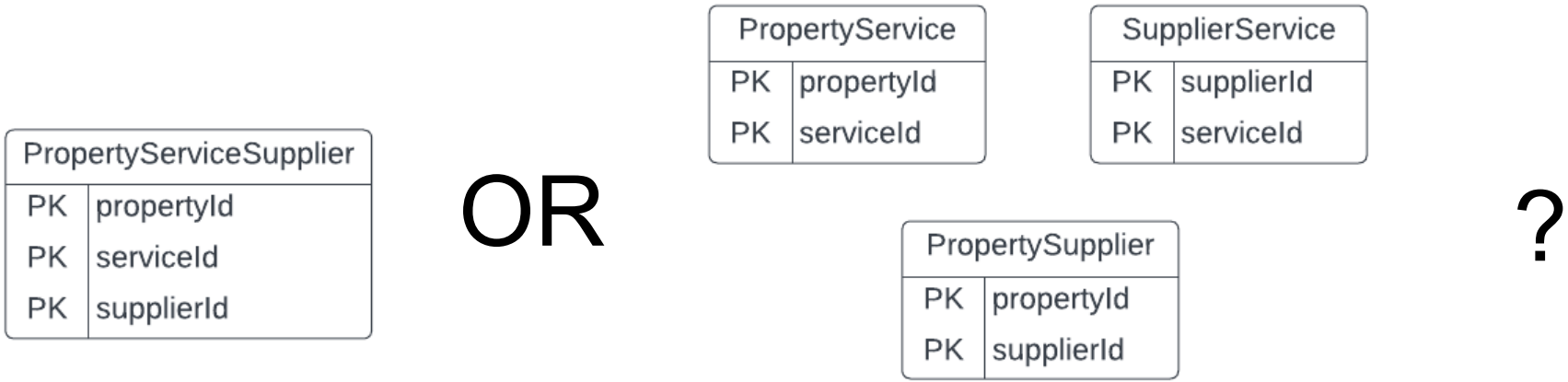
Normalisation 4F

Course	Teacher	Materials
DB	Roels	db-book
DB	Roels	w3schools
DB	Roels	owncourse
DB	Sourie	db-book
DB	Sourie	owncourse
DB	Sourie	w3schools
Java	Vlummens	j4profs
	…	

Normalisation 5F

– 5NF: 1NF + all relations in any join-dependency contain one of the keys

Example:
properties need services, services need suppliers and suppliers serve properties



Normalisation 5F

Is there a constraint?

Yes?

If a property required a service

And a supplier supplies the service

And a supplier supplies for the propery

Then supplier supplies service for property

Then a **join-dependency** exist and the relation can be **lossless decomposed**

Normalisation 5F

Property	Service	Supplier
House1	Electricity	Bt
House1	Electricity	U4s
House1	Elevator	U4s
House1	Plumbing	Bt
House2	Electricity	Tc
House3	Elevator	U4s
…	…	…

Normalisation 5F

Property	Service
House1	Electricity
House1	Elevator
House1	Plumbing
House2	Electricity
House3	Elevator
…	…

⋈

Supplier	Service
Bt	Electricity
Bt	Plumbing
Tc	Electricity
U4s	Electricity
U4s	Elevator
…	…

⋈

Property	Supplier
House1	Bt
House1	U4s
House2	Tc
House3	U4s
…	…

Normalisation 5F

PropertyService join with SupplierService join with PropertySupplier last join deletes non-existing row

Property	Service	Supplier
House1	Electricity	Bt
House1	Electricity	U4s
House1	Electricity	Tc
House1	Elevator	U4s
House1	Plumbing	Bt
House2	Electricity	Tc
House3	Elevator	U4s
…	…	…

Normalisation 5F

A **join-dependency** exists and not all relations in that dependency have the key, more specifically none of them.

The relation is not in 5NF so should be splitted.

After the split each of the 3 relations is in 5NF because there is no join-dependency left.

Normalisation 5F

Is there a constraint?

No?

So every supplier can support any service at a property

Then a join-dependency doesn’t exist and the relation

can’t be lossless decomposed

Normalisation 5F

Property	Service	Supplier
House1	Electricity	Bt
House1	Electricity	U4s
House1	Elevator	Tc
House1	Plumbing	Bt
House2	Electricity	Tc
House3	Elevator	U4s
…	…	…

Normalisation 5F

Property	Service
House1	Electricity
House1	Elevator
House1	Plumbing
House2	Electricity
House3	Elevator
…	…

⋈

Supplier	Service
Bt	Electricity
Bt	Plumbing
Tc	Electricity
Tc	Elevator
U4s	Electricity
U4s	Elevator
…	…

⋈

Property	Supplier
House1	Bt
House1	Tc
House1	U4s
House2	Tc
House3	U4s
…	…

Normalisation 5F

Property	Service	Supplier
House1	Electricity	Bt
House1	Electricity	Tc
House1	Electricity	U4s
House1	Elevator	Tc
House1	Elevator	U4s
House1	Plumbing	Bt
House2	Electricity	Tc
House3	Elevator	U4s
…	…	…

Normalisation 5F

A **join-dependency** doens’t exist so the relation is in 5NF.

Modelling issues

course: Information Modelling