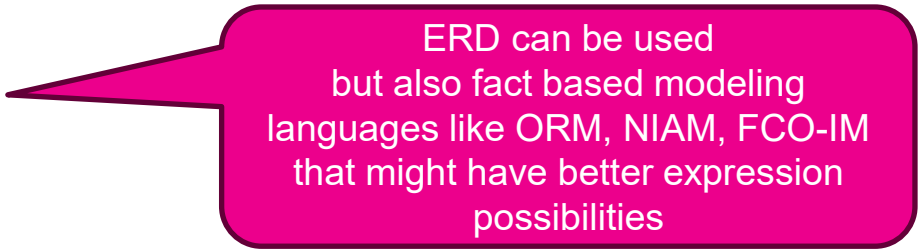
From ERD to relational model

course: Information Modelling

DESIGN

CONCEPTUAL

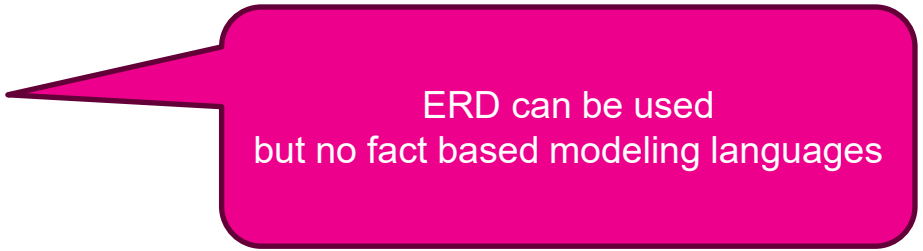
- Entity types
- Attributes
- Relationships



DATABASE DESIGN

LOGICAL

- Relations / Tables
- Attributes / Columns and keys
- Foreign key relationships
- General constraints
- ...



No implementation details

Mind the difference between relation and relationship

DATABASE DESIGN

TECHNICAL / PHYSICAL

- Tables with implementation choices
- Columns with definitions
- Foreign Key constraints
- Indexes
- ...

With implementation details

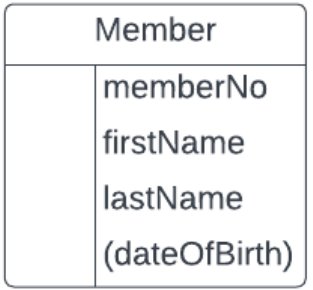
From conceptual to logical DB model

course: Information Modelling

FROM ENTITY TYPE TO RELATION

RELATION WITH ATTRIBUTES

- Notation of entity type can be used for relations, but add an additional first column to indicate some constraints like PK, FK, UC,...
- Relation name: same name as entity type
- Include all the simple (optional and non-optional) attributes
- Put optional attributes in branckets



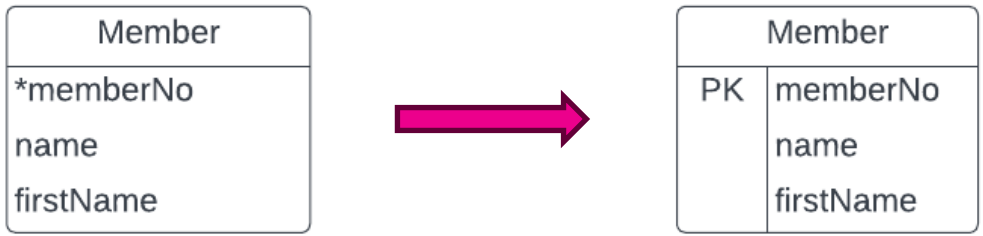
FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE

- Strong entity type
  - Primary key (PK)
    - Choose an identifying attribute or combination of those attributes
    - Prefer a single attribute
    - Prefer an attribute without a meaning
    - **In case of combination of attributes: no part of the key is redundant!**
  - Candidate keys: mark as uniqueness contraint (UC / UCx and x a number)

FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE



FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE

ClassRoom
*buildingNr
*floorNr
*corridorNr
*RoomNr
maxNrOfPersons

ClassRoom	
PK	buildingNr
PK	floorNr
PK	corridorNr
PK	RoomNr
	maxNrOfPersons

FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE

ClassRoom
*classRoomId$_1$
*buildingNr$_2$
*floorNr$_2$
*corridorNr$_2$
*RoomNr$_2$
maxNrOfPersons

ClassRoom	
PK	classRoomId
UC	buildingNr
UC	floorNr
UC	corridorNr
UC	RoomNr
	maxNrOfPersons

FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE

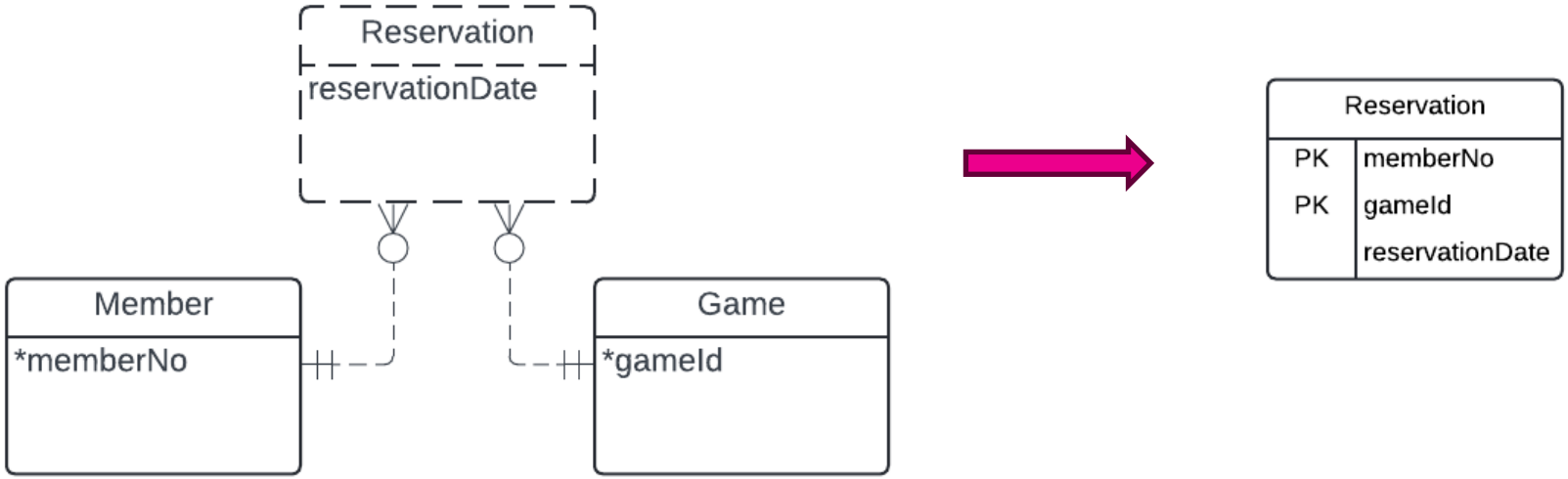
- Weak entity type
  - Primary key
    - Add **the primary key of all related strong entity types** (with a many-to-one relationship) **and the partial key** if existing as attribute and define it as key.
      In case that the combination is not identifying add **extra identifying attributes** to the key (should have been modelled at conceptual level).
      Remember: **no part of the key is redundant!**
  - Candidate key: mark as uniqueness contraint

FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE

FROM ENTITY TYPE TO RELATION

KEY ATTRIBUTE



FROM ENTITY TYPE TO RELATION

DERIVED ATTRIBUTE

- Is there an added value (e.g. performance) to store in table?
  - No
    - calculate in **code** (after fetching)
    - calculate in **query** (while fetching)
    - calculate with **stored procedure or view** (while fetching)
  - Yes
    - calculate in **database trigger** (on insert, on update, on delete) to guard consistency

FROM ENTITY TYPE TO RELATION

COMPOSITE ATTRIBUTE

- **Split** it in simple/atomic attributes
- Create an **extra relation** if it can be a concept, split in simple/atomic attributes and treat it as an extra entity type (e.g. address) with a one to many relationship with the source relation
- **Keep it composite** (e.g. date, json, ...) if the parts are not that important

FROM ENTITY TYPE TO RELATION

MULTI-VALUED ATTRIBUTE

- Create a **new relation**
- **Add that attribute** as an attribute of the new relation
- If that attribute is not suited to be part of the key, add an extra meaningless attribute
- Take the **primary key of the relation** where the multi-valued attribute belonged to and add it **as attribute** and make it part of the key of the new relation. Together with the other key attribute it will be the primary key.

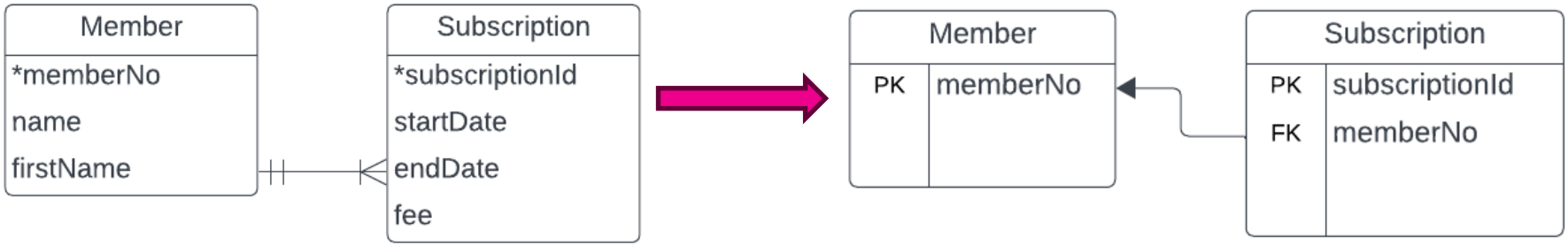
BINARY RELATIONSHIP

ONE-TO-MANY

- The entity type on the **one side** is the **parent** entity type
- The entity type on the **many side** is the **child** entity type
- **Add the primary key** of the parent relation **as attribute** of the child relation and create a foreign key relationship from the child to the parent. Mark the attributes with FKx - x as a number to avoid confusion if needed
- Draw an arrow from the foreign key to the primary key (cfr databases)

BINARY RELATIONSHIP

ONE-TO-MANY



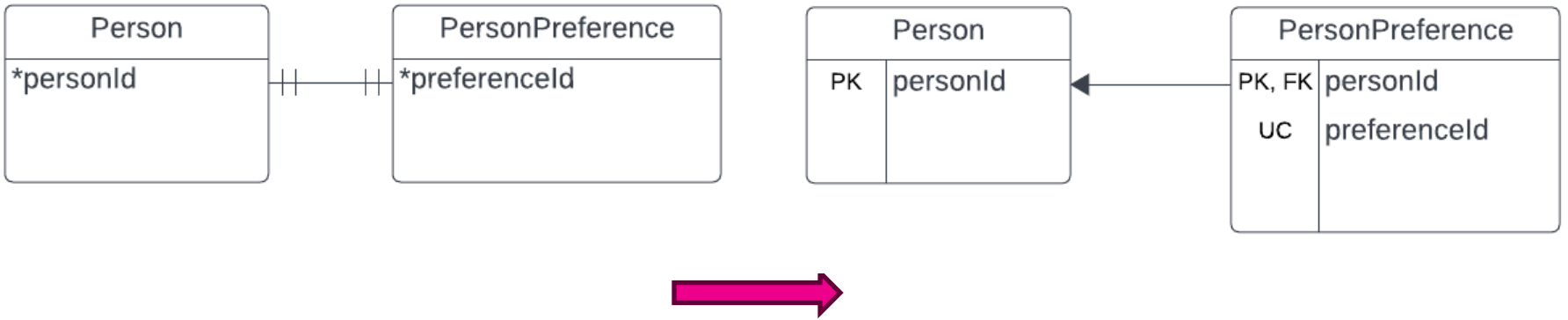
BINARY RELATIONSHIP

ONE-TO-ONE BOTH MANDATORY

- Mandatory participation on both sides of the relationship
  - **Keep separate relations** and choose the primary key of one of the relations as primary key in the other relation where the original will be changed as alternate key (or drop it if not important).
  - **Keep separate relations** and their primary keys but add in one of the relations the primary key as attribute and make it an alternate key
  - **Combine the attributes in one relation** and choose one of the keys as primary key and the other as alternate key (or drop it if not important)

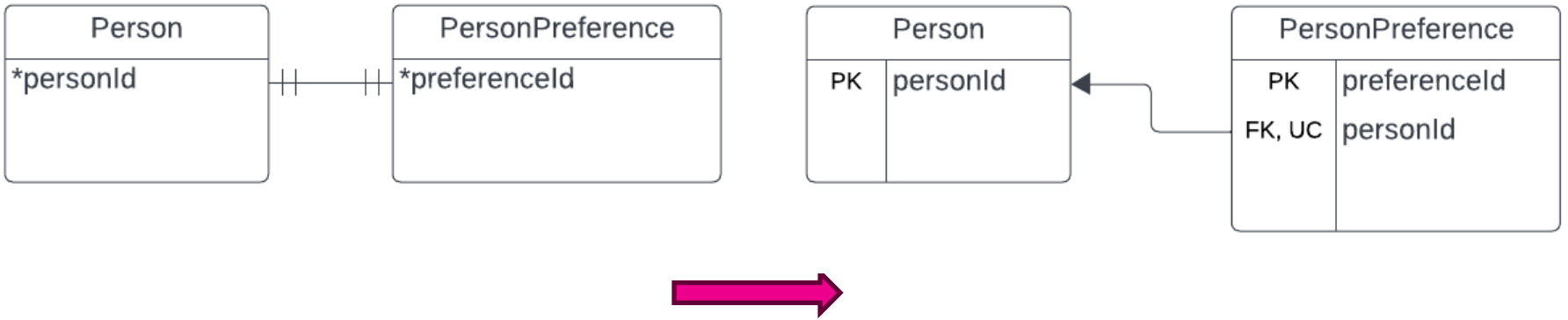
BINARY RELATIONSHIP

ONE-TO-ONE BOTH MANDATORY



BINARY RELATIONSHIP

ONE-TO-ONE BOTH MANDATORY



BINARY RELATIONSHIP

ONE-TO-ONE MANDATORY/OPTIONAL

- Mandatory participation on one side of the relationship
  - The entity type on the **mandatory side** is the **parent** entity type
  - The entity type on the **optional side** is the **child** entity type

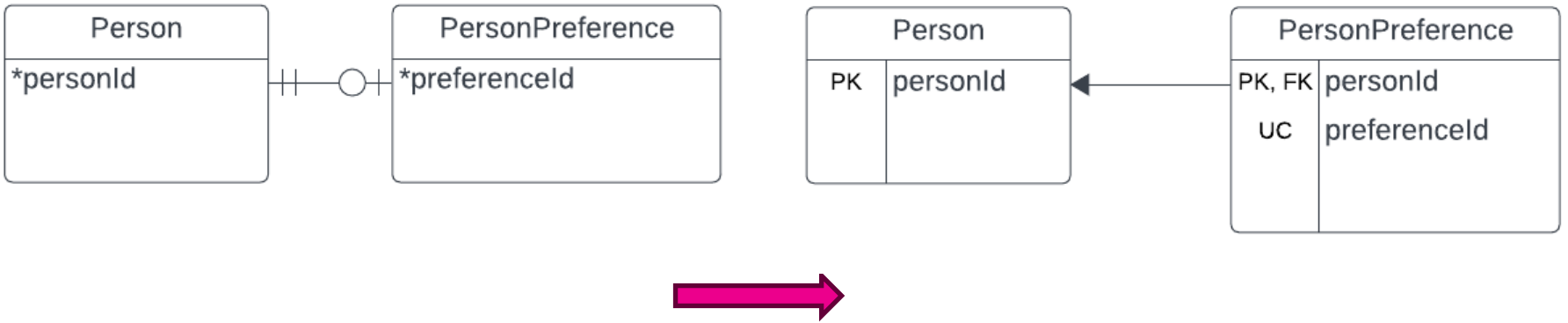
BINARY RELATIONSHIP

ONE-TO-ONE MANDATORY/OPTIONAL

- Mandatory participation on one side of the relationship
  - **Keep separate relations** and choose the primary key of the parent relation as primary key in the child relation where the original will be changed as alternate key (or drop it if not important)
  - **Keep separate relations** and their primary keys but add in the child relation the primary key as attribute and make it an alternate key
  - **Combine the attributes in one relation** and choose the key of the parent as primary key and the other as alternate key (or drop it if not important). Make the attributes of the client relation optional

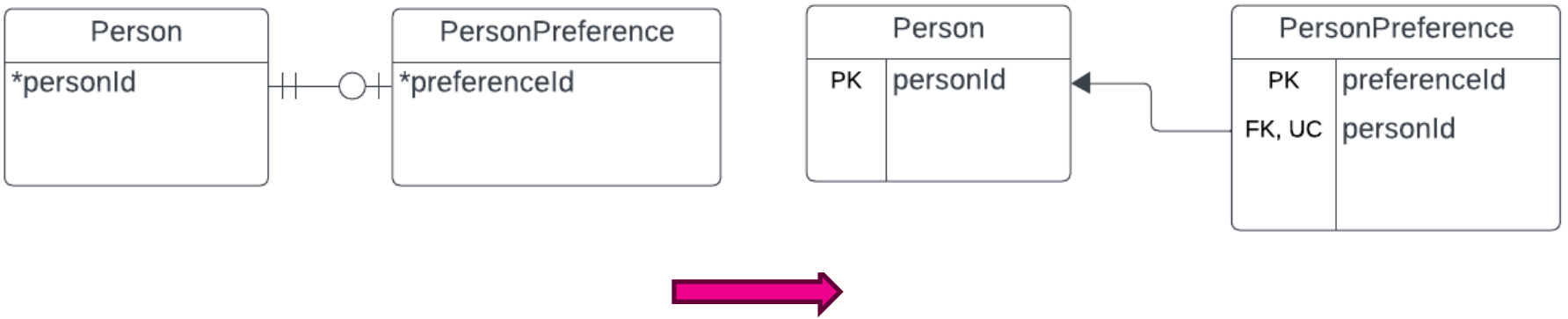
BINARY RELATIONSHIP

ONE-TO-ONE MANDATORY/OPTIONAL



BINARY RELATIONSHIP

ONE-TO-ONE MANDATORY/OPTIONAL



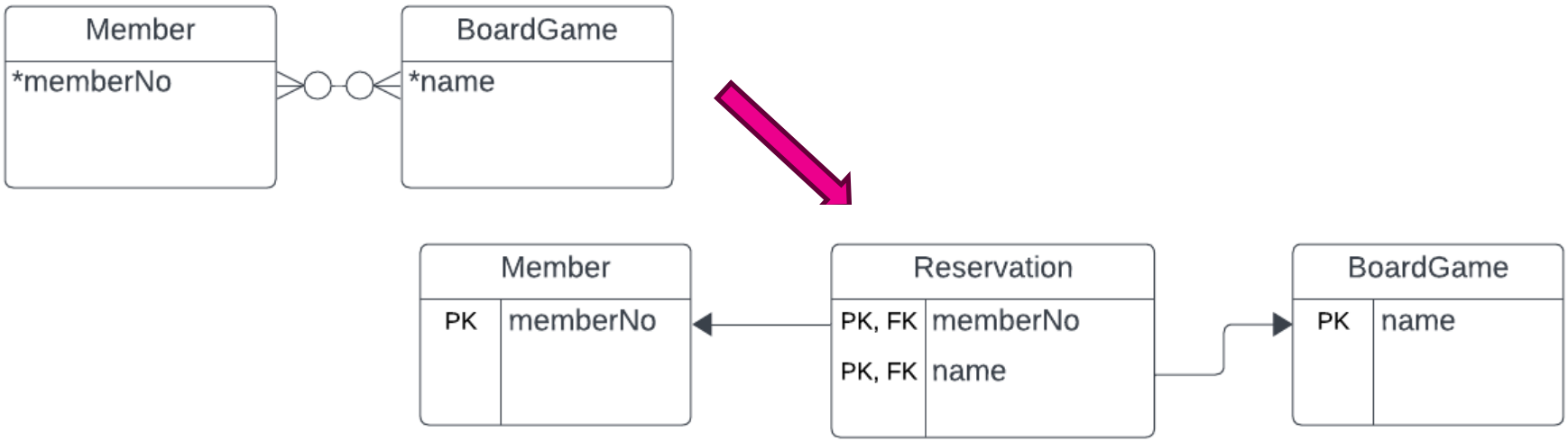
BINARY RELATIONSHIP

MANY-TO-MANY

- **Create a new relation**
- Add eventual attributes to the relation (normally in crow's foot notation the relationship should have been already upgraded as an entity type)
- Add the primary key of both related relations as attribute and make the combination the primary key of the relation

BINARY RELATIONSHIP

MANY-TO-MANY



N-ARY RELATIONSHIP

MANY-TO-MANY

- Create a relation
- Add eventual attributes to the relation (normally in crow's foot notation the relationship should have been already upgraded as an entity type)
- Add the primary key of all related relations as attribute and make the combination the primary key of the relation
- **These relations can have normalisation problems! We will see this later.**

SUPERTYPE / SUBTYPE TO RELATION - DIFFERENT MAPPINGS

SUPERTYPE / SUBTYPE

- **Create one relation for the supertype and one relation per subtype (SP+#SB)**
- Create one relation for the supertype with all attributes of the subtypes (SP)
- Create one relation for the supertype and one relation for all the attributes of all subtypes (SP+SB)
- Create one relation per subtype and add the attributes of the supertype in each relation (#SB)

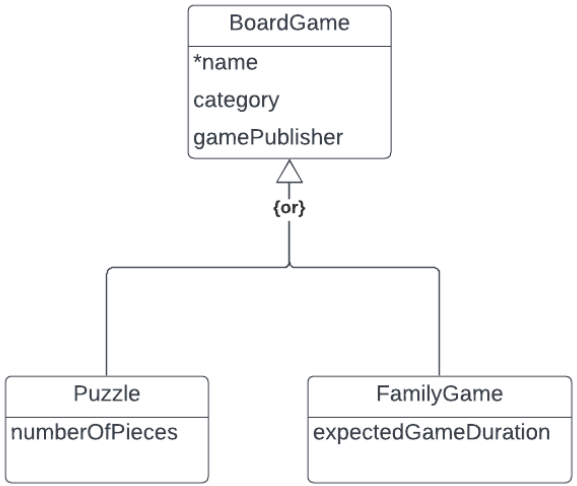
SUPERTYPE / SUBTYPE TO RELATION - DIFFERENT MAPPINGS

SUPERTYPE / SUBTYPE

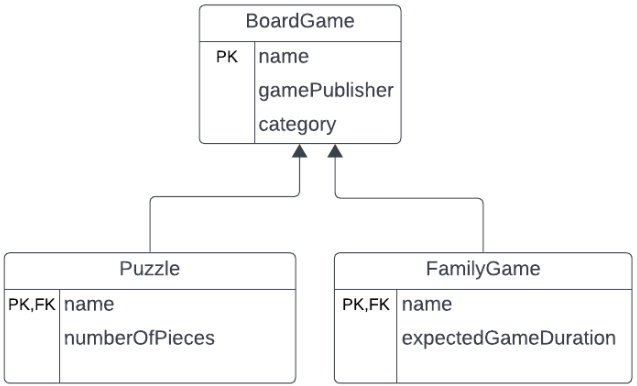
The following examples assume that there is only 1 category which leads to disjoint subtypes.

SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE

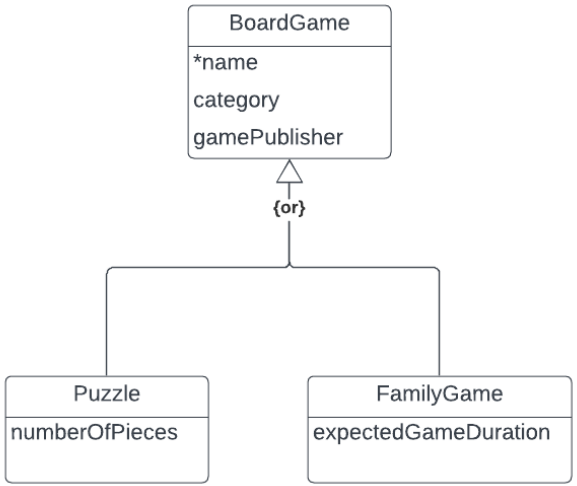


supertype
+
#subtypes
(SP+#SB)



SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE

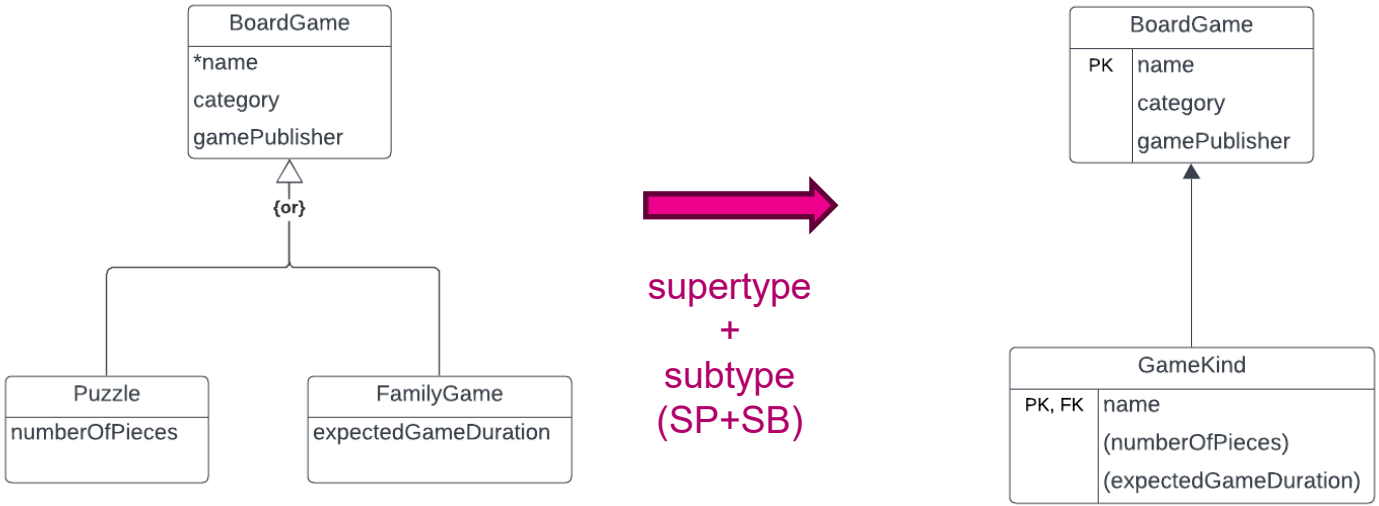


Supertype
(SP)

BoardGame	
PK	name
	gamePublisher
	category
	(numberOfPieces)
	(expectedGameDuration)

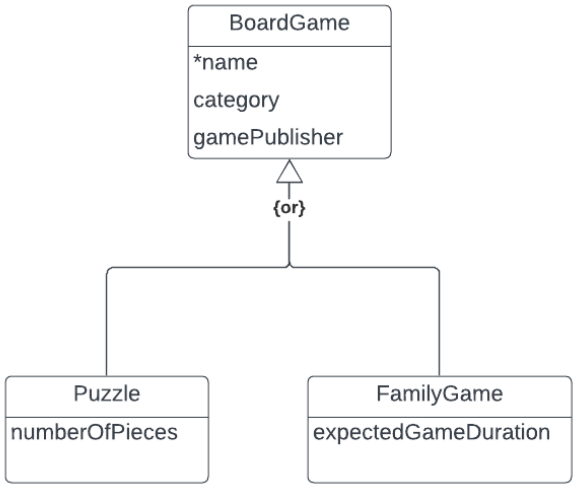
SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE



SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE



#subtypes
(#SB)

Puzzle	
PK	name
	gamePublisher
	numberOfPieces

FamilyGame	
PK	name
	gamePublisher
	expectedGameDuration

SUPERTYPE / SUBTYPE

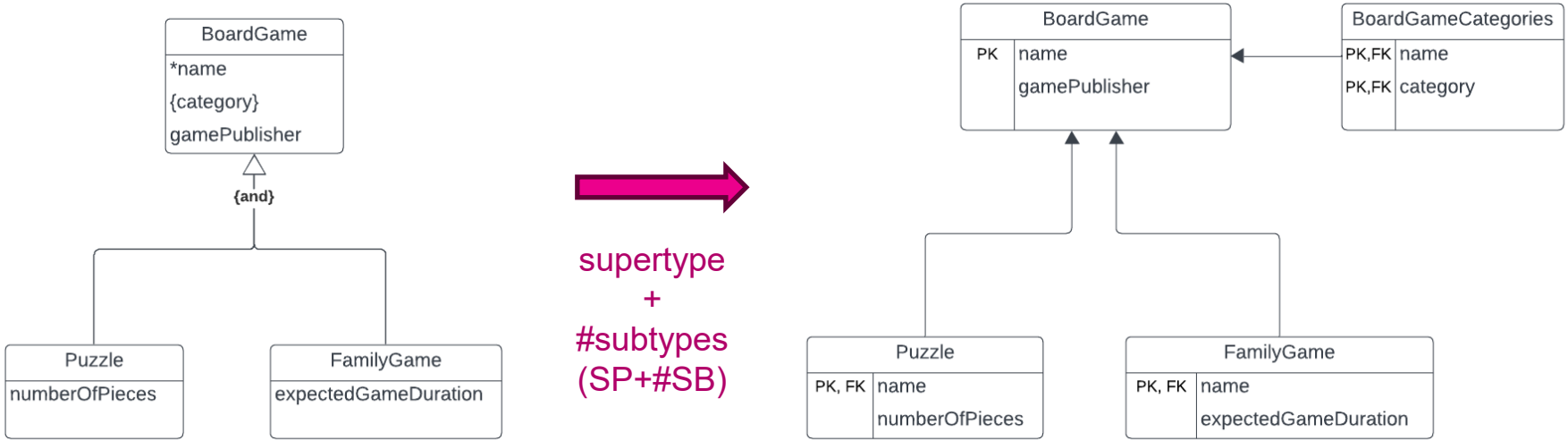
In case more than 1 category is possible (category as multi-valued attribute) there are non-disjoint subtypes, the same solutions apply but

- You need a table for the categories for the products (preferred solution) or
- You need a flag to indicate which categories are applicable

We give one of the four solutions and apply both mappings.

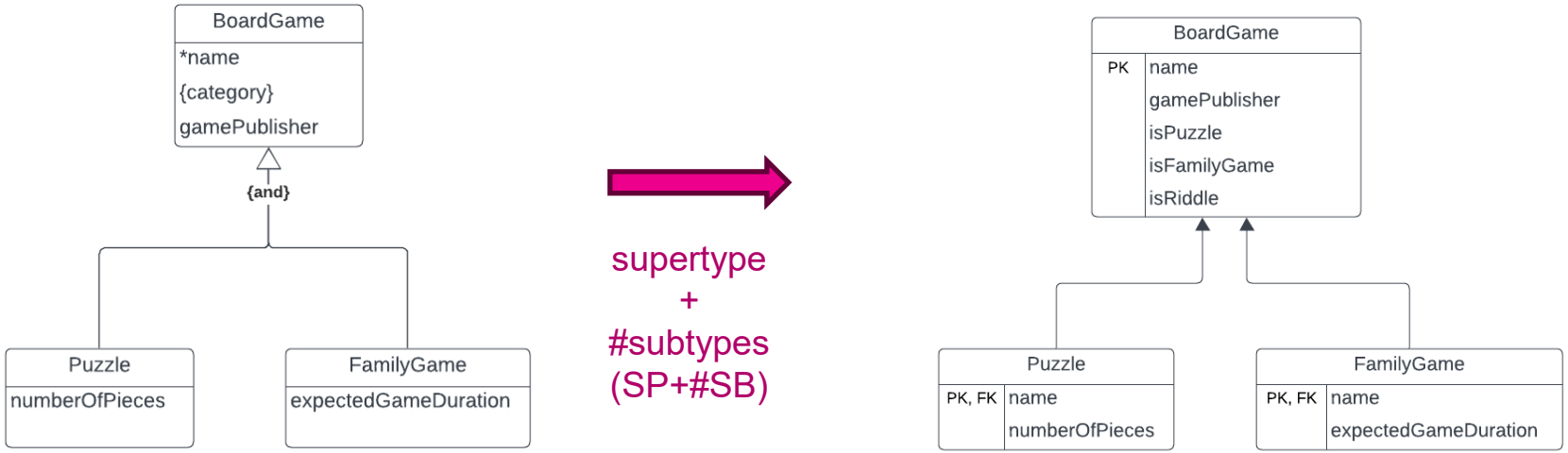
SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE



SUPERTYPE / SUBTYPE TO RELATION

SUPERTYPE / SUBTYPE



From logical to physical DB model

@course databases

course: Information Modelling

LOGICAL RELATIONAL MODEL TO PHYSICAL DATASE MODEL

NAMING CONVENTIONS

- Table names plural Member -> members or MEMBERS
  Underscore in case of combination of words
  Column names same

- ...

From ERD to relational model

course: Information Modelling