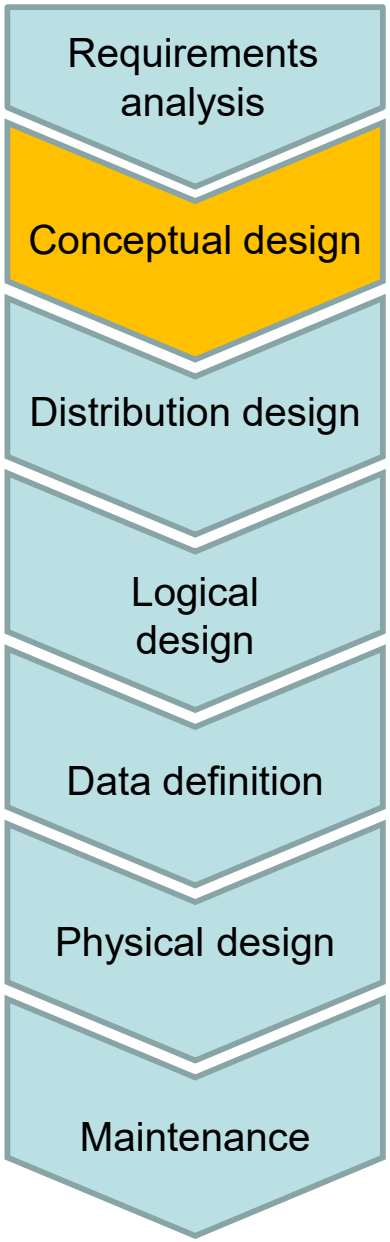
Technische Hochschule Rosenheim

Chapter 6 – Conceptual database design

Databases lectures

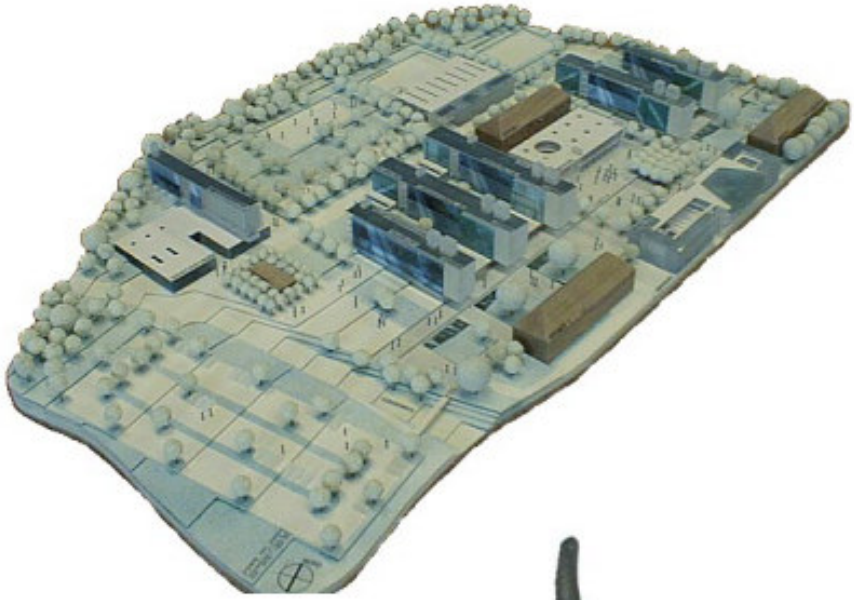
Prof. Dr Kai Höfig

Conceptual design – revision

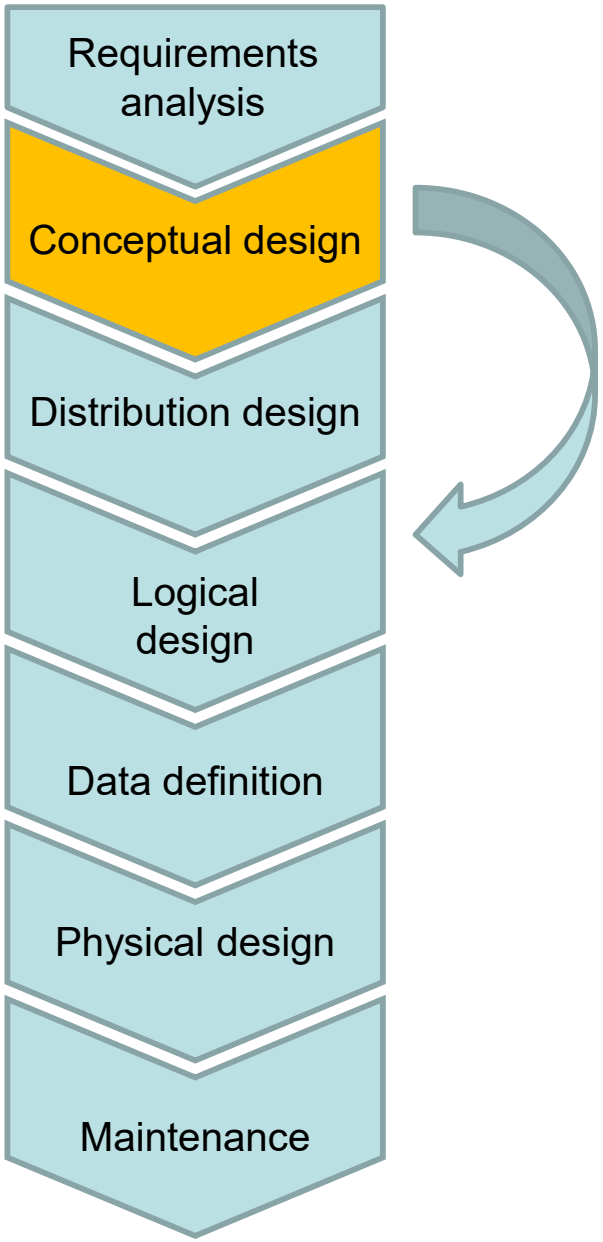


- Aim: initial formal description of the specialist problem, *regardless* of the system to be used later
- Linguistic devices: abstract (semantic) data model
- Method:
  - Modelling views, e.g. for different specialist departments
  - Analysis for conflicts
  - Integration of the views into an overall schema
- Result: conceptual data model, typically an ER or UML diagram
- Most important characteristics of design steps
  - Information preservation
    All data that could be saved in the previous model can also be saved in the new model
  - Consistency preservation
    Rules and restrictions in the previous model can also be ensured in the new model

Model and modelling

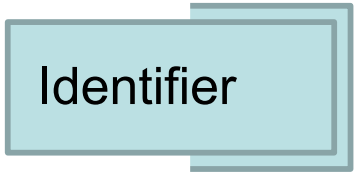


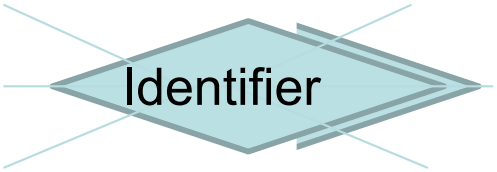
Entity-Relationship (ER) model



- The entity-relationship model allows us to graphically represent entities, attributes and relationships of data using simple language constructs (**syntax**)
- A relational database model can then be derived from this in a (partially) automated process (**semantics**)

Graphical notation and semantic meaning of the ER model

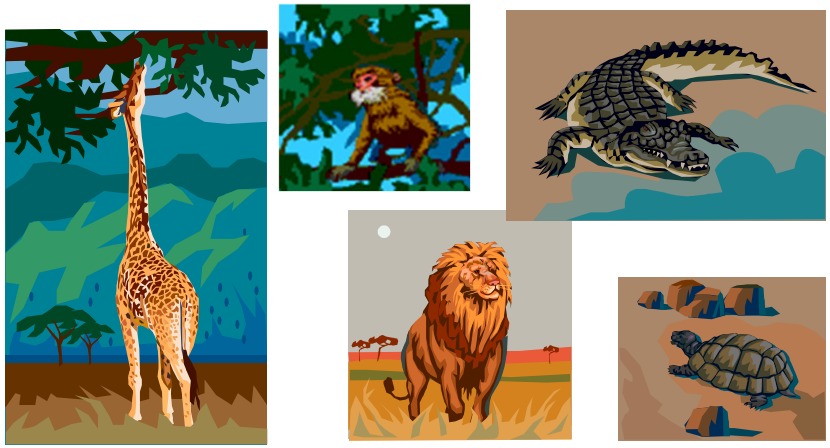




- Entity types map the objects in the real world. Relations are later derived from them.
  - A double border indicates a *weak entity*.
- Attributes are properties of entity types. They are later also attributes of the relations.
  - If it is a key attribute, the identifier is underlined.
  - A dashed border means that it is a calculated attribute.
- Relationships model the interrelationships between entity types. They can become relations or attributes in the relational model.
  - A double border indicates a *supporting relationship*.
- is-a relationships are used to model generalisations and specialisations. They collect attributes of an entity type in the relational model.

Example – the zoo DB

- You are tasked with creating a database for the administration of Rosenheim Zoo.



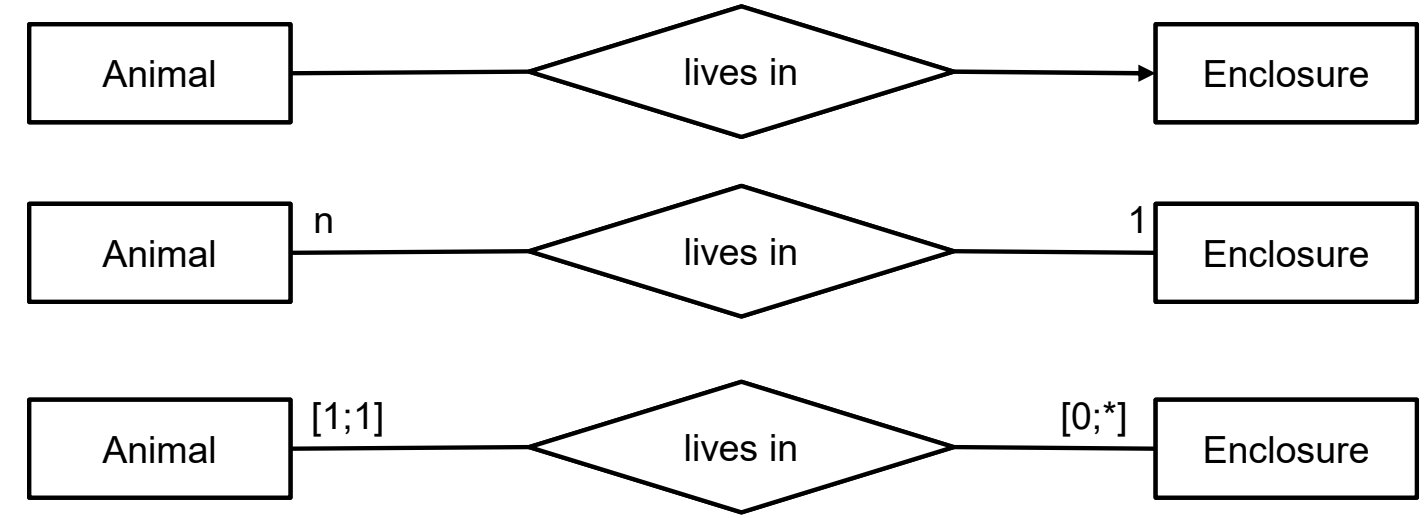
- You have conversations with Mr P, the zoo director, to find out the requirements for the database.

- Mr P:

Dear database professional, thank you for agreeing to help me!

Cardinalities of relationships: 1:n

Several animals live in each enclosure.



1:n relationship

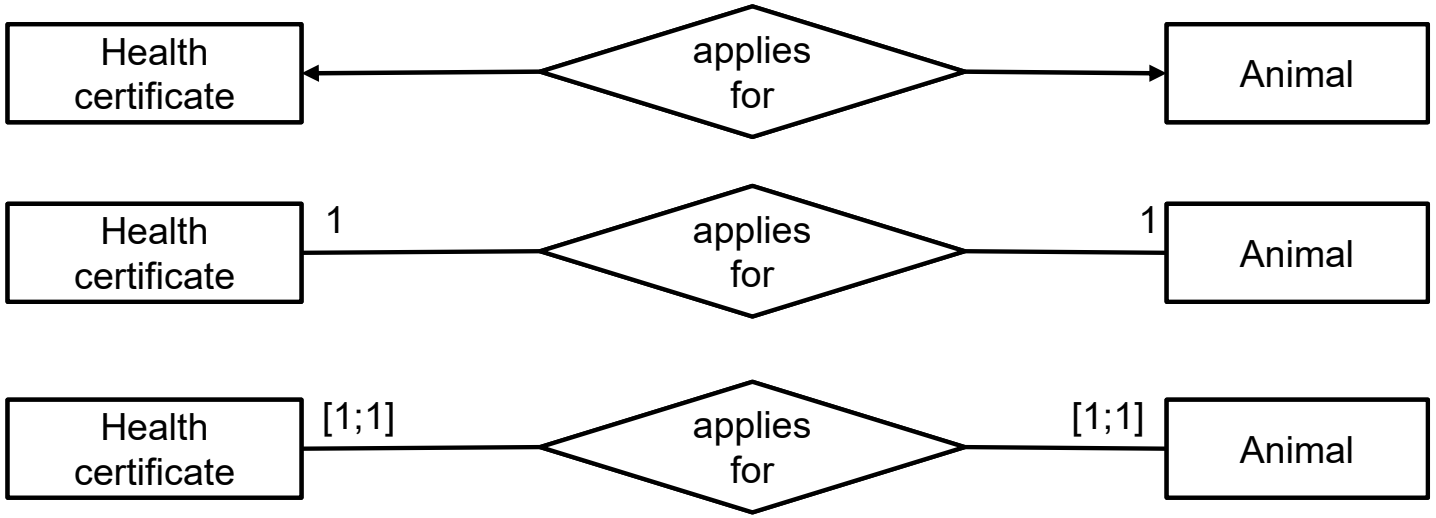
Every entity e_1 of the entity type E_1 can be assigned any number of entities E_2, but each entity e_2 can have a maximum of one e_1 from E_1

Examples

- Supplier delivers products
- Mother has children

Cardinalities of relationships: 1:1

Each animal receives its own health certificate from the vet.



1:1 relationship

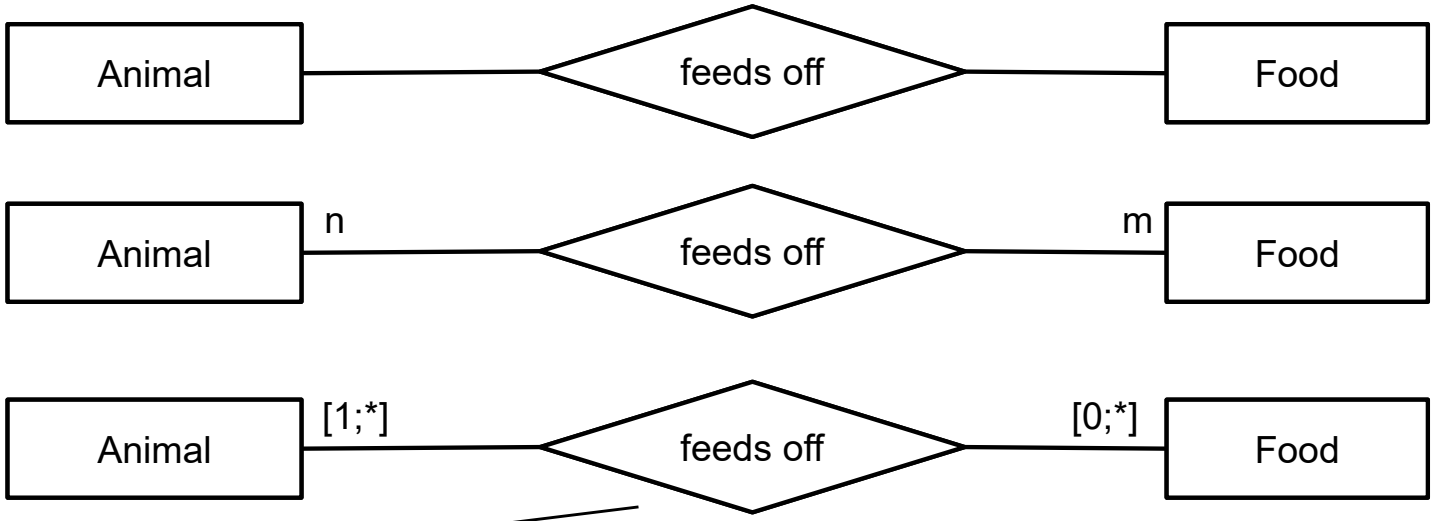
Every entity e_1 of the entity type E_1 has a maximum of one entity e_2 assigned from E_2 and vice versa

Examples

Brochure describes product,
Man is married to woman

Cardinalities of relationships: n:m

We have several types of food for the animals, whereby one animal likes different types of food and one type of food is good for more than one animal.



n:m relationship

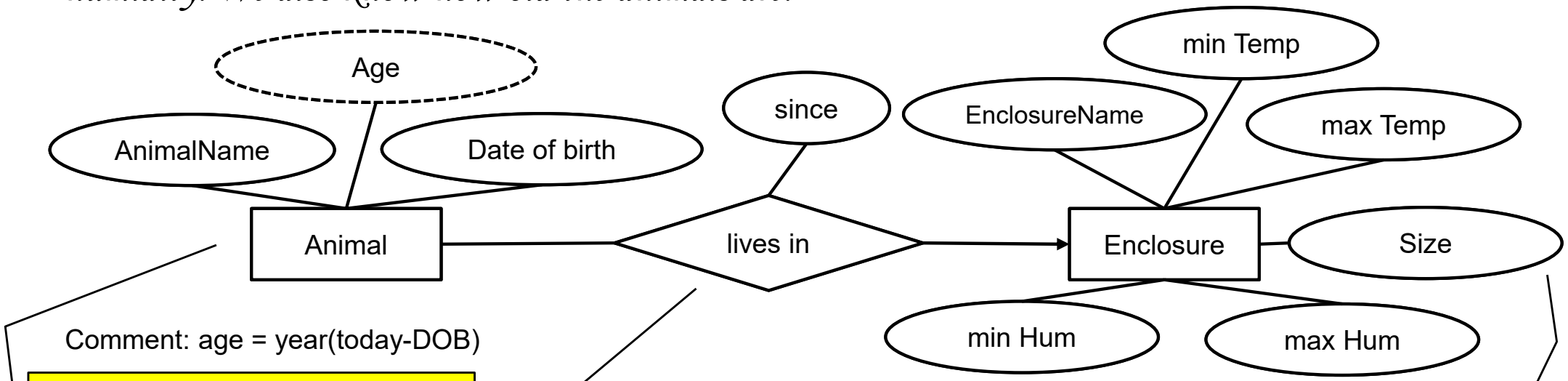
Every entity e_1 of the entity type E_1 can be assigned any number of entities e_2 from E_2 and vice versa

Examples

Orders include products

Entity types and relationships

We have lots of animals in the zoo, and they live in enclosures. The animals have names and enclosures and also a size. Several animals live in each enclosure. For each enclosure, we have a maximum temperature, a minimum temperature, a maximum humidity and a minimum humidity. We also know how old the animals are.



Entity type: object from the real/imaginary world about which information is to be stored

Examples: **Animal**, **Enclosure**; but also information about events, e.g. **Orders**

Relationship: describes the interrelationship between entities

Examples: an animal **lives** in an enclosure, an order is **placed by** a customer

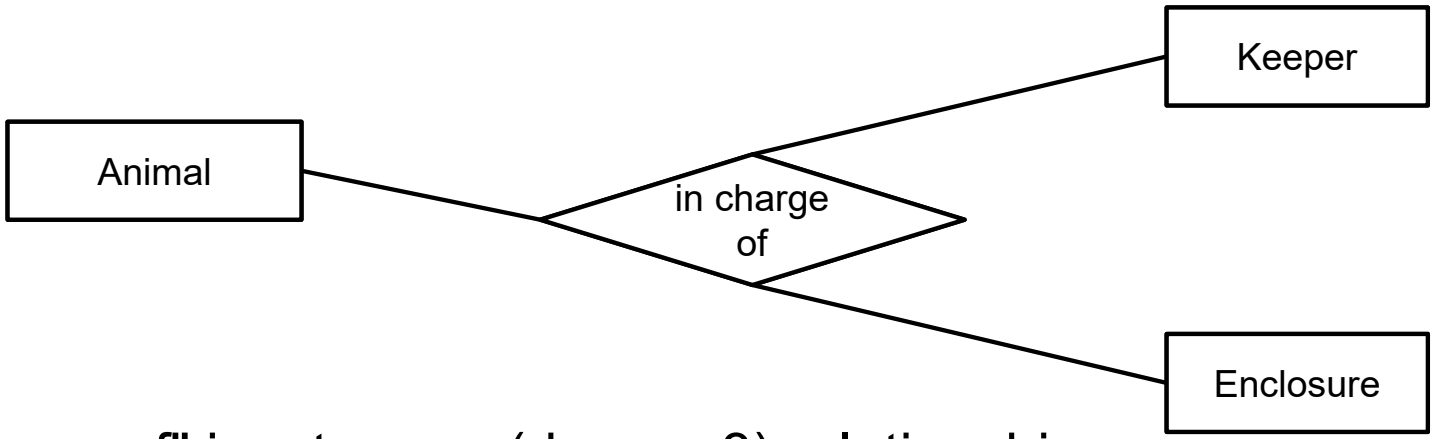
Attribute: represents a property of an entity or relationship

Examples: **name** of an animal or **date** of an order

Degrees of relationships

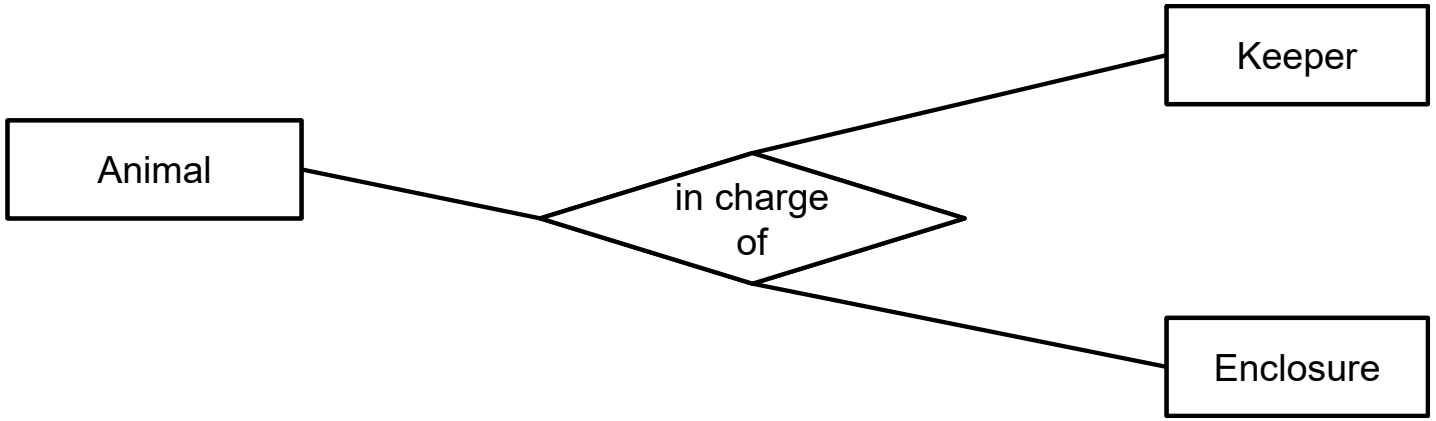
- Degree of a relationship
  - Number of entity types involved (often: binary)
  - Example: supplier delivers product, animal lives in enclosure

Every one of our keepers is in charge of several animals, but they all live in one enclosure.

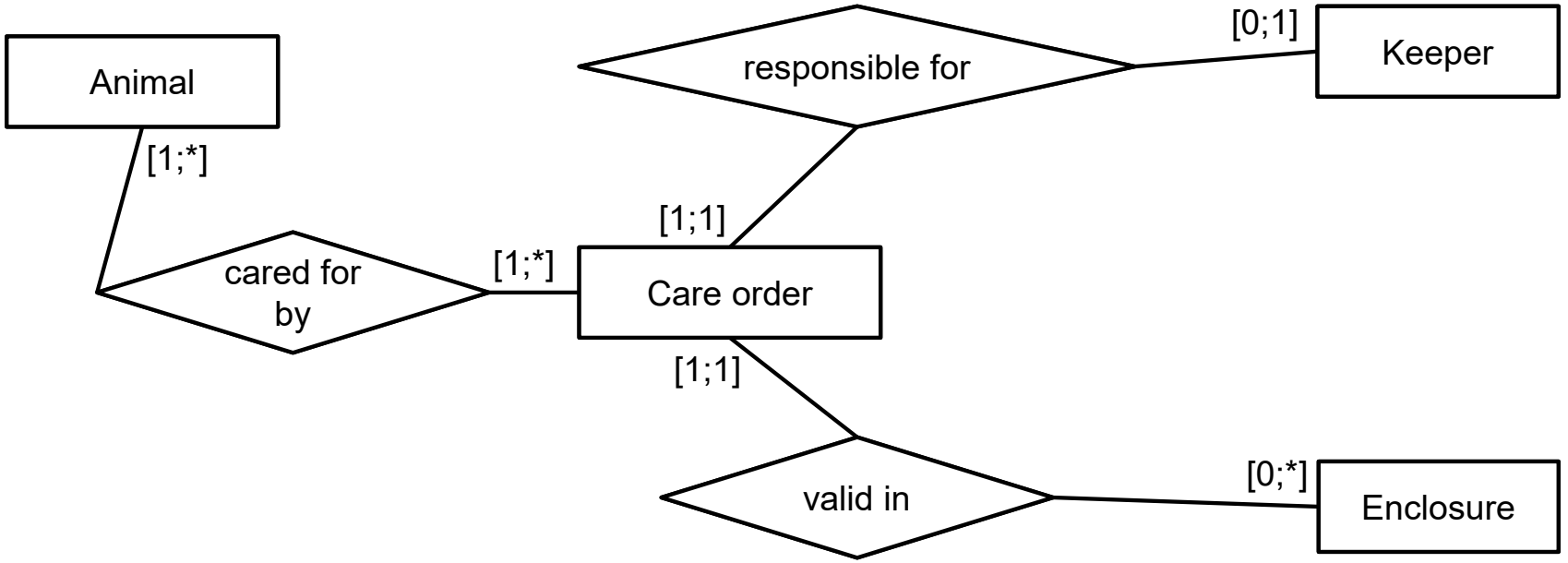


- "in charge of" is a ternary (degree 3) relationship
  - Note: cardinalities cannot be uniquely represented!

Modelling of n-ary (degree n) relationships using binary relationships

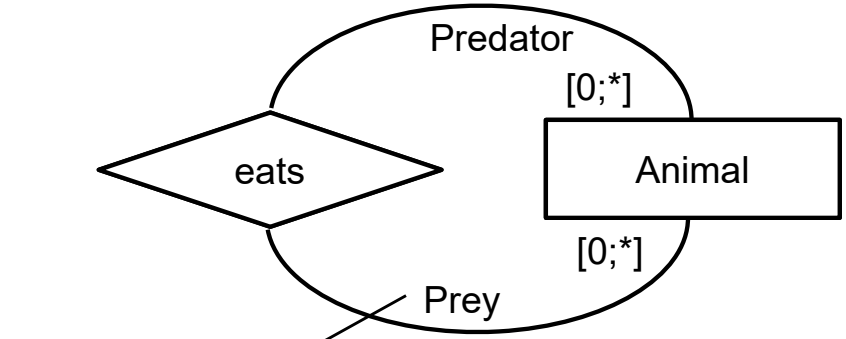


- Alternative modelling of the "in charge of" relationship



Roles in relationships

Oh yes, and we need to make sure that we don't put any animals into one enclosure that eat the others!

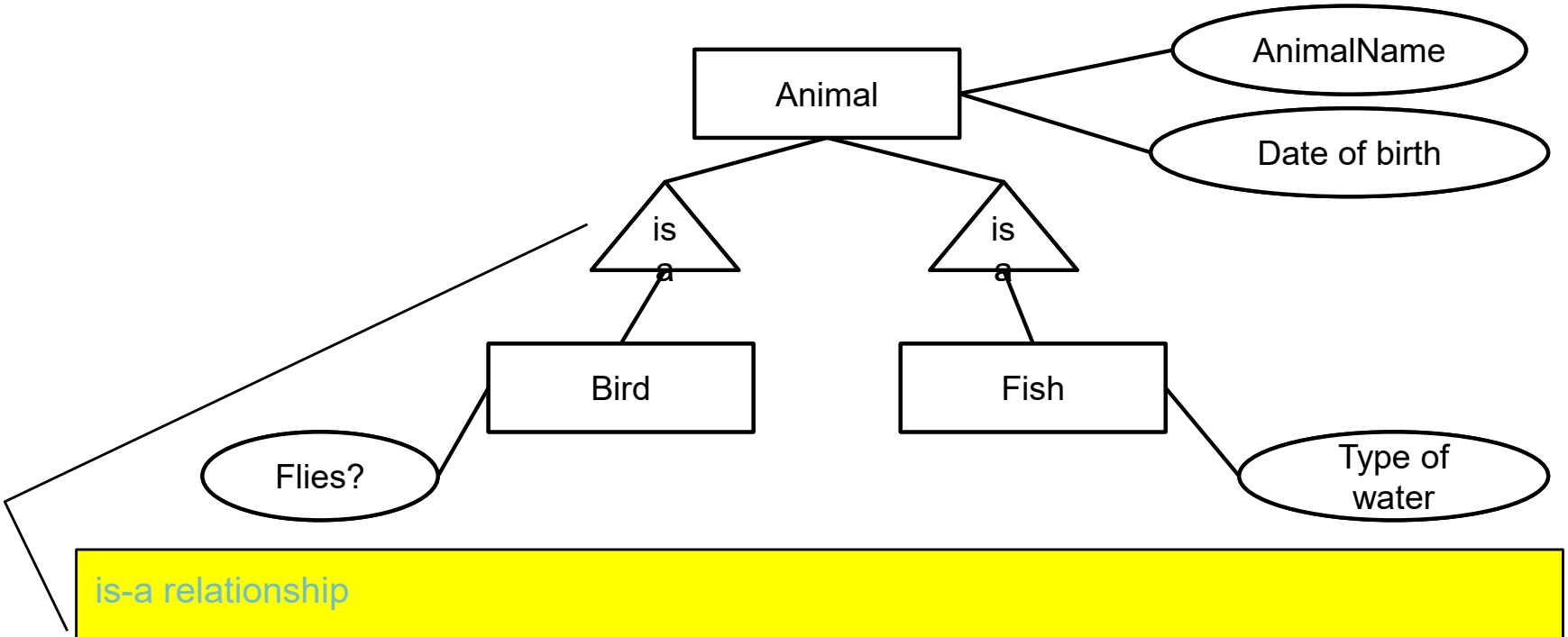


Roles

If an entity type occurs more than once in a relationship, we must assign role names to distinguish between the instances.

is-a relationships

We also need to make sure that take special care of birds and fish - fish need fresh or salt water, and some birds fly whereas others don't.

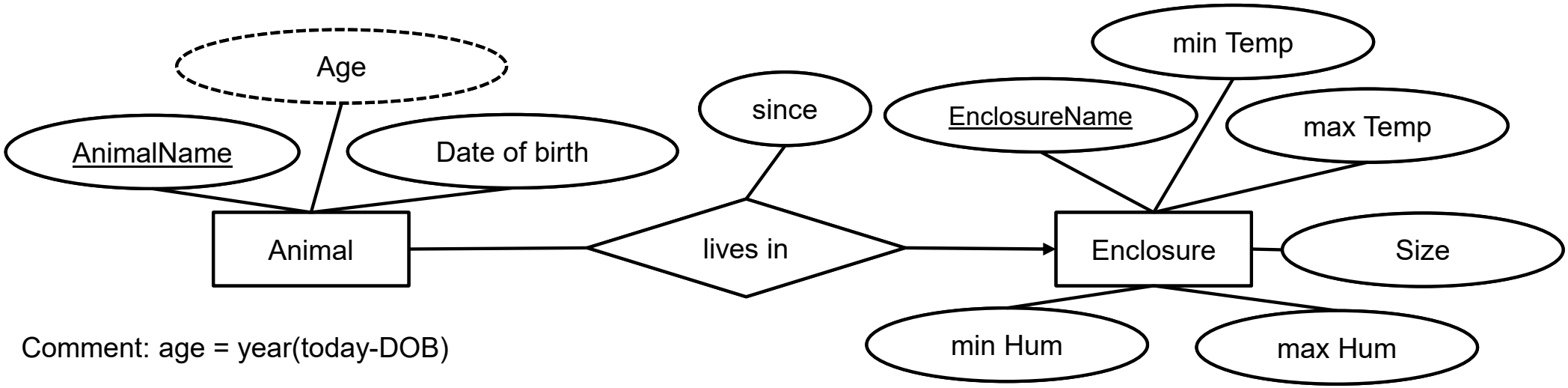


is-a relationship

- Corresponds to generalisation/specialisation
- Always 1:1 (therefore not marked separately)
- Attributes are "collected"
- With us: always a clear "father"

Keys in the ER model

- Keys = underline all attributes involved



Comment: age = year(today-DOB)

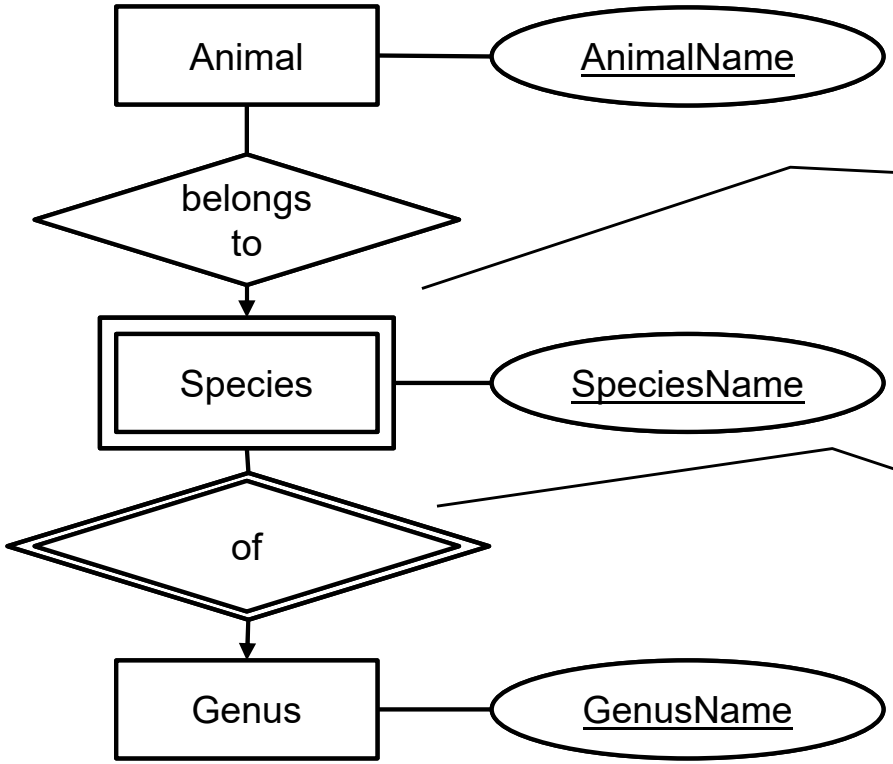
- Comments
  - Generally only primary keys are marked
  - For is-a relationships, the top entity type must contain all relevant attributes

Weak entity types (1)

Each animal belongs to a species, and each species belongs to a genus. Whereby the same species can belong to different genera, which is why only the combination of species and genus can accurately describe the animal.

>> "Sorry, I didn't understand that - could you give me an example?"

Of course: Cyriocosmus elegans is a tarantula, Centruroides elegans is a scorpion. "elegans" is the species and "Cyriocosmus" or "Centruroides" is the genus.



Weak entity types

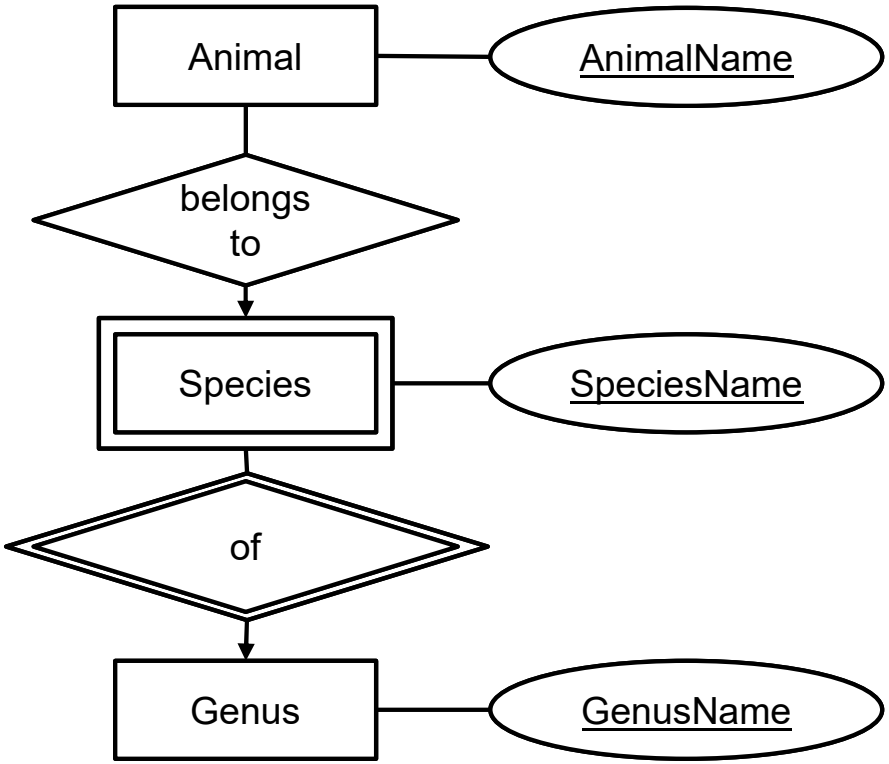
- Key is (SpeciesName, GenusName)

Supporting relationship

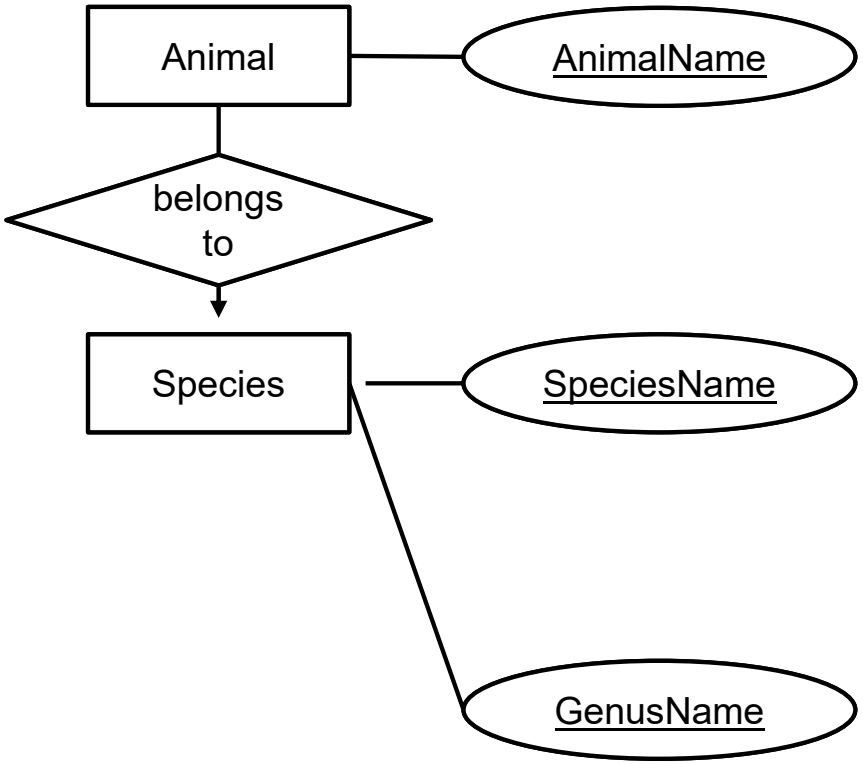
- Always 1:n
- In the associated entity type there are other key attributes of the weak entity type

Weak entity types (2)

Weak entity types should also not be overused. In principle, a large number of attributes can be moved out into separate relations. However, this only makes sense if additional data about the new entity is also stored. So if genus only consists of a single attribute, there is an alternative implementation:



Implementation with weak entity type



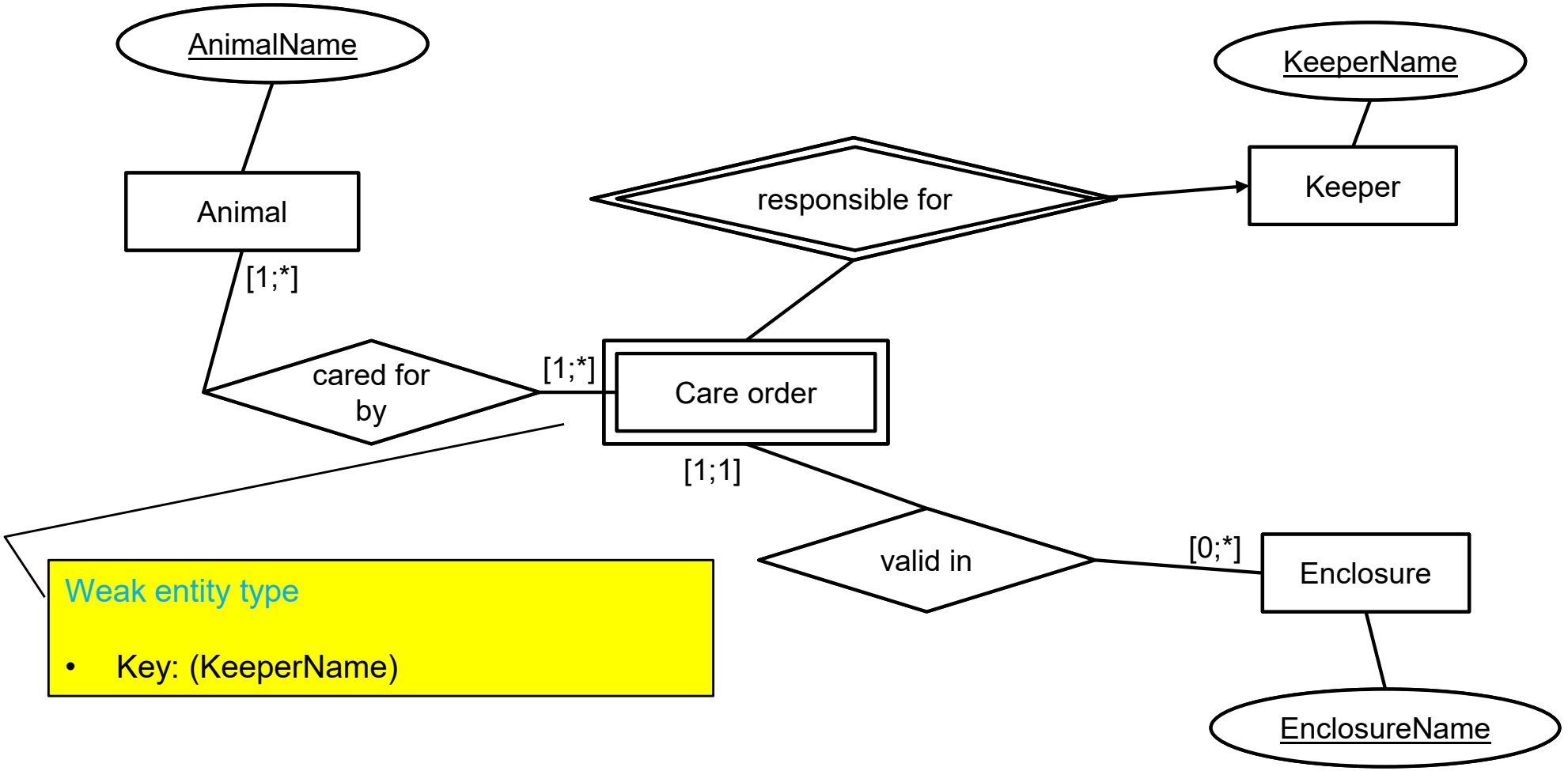
Alternative implementation as attribute

Weak entity types (3)

- Weak entity type: does not contain all attributes that are required for its key
  - Marked by double border
  - Some key attributes are contained in other entities
  - Therefore necessary: 1:n relationships ("supporting relationships", diamond with a double border) to these other entity types!
  - How do we recognise that an entity is weak? No key in its attributes!
- Key of a weak entity type
  - May have own attributes (underlined)
  - All keys of the entity types associated with supporting relationships (and recursively if they are also weak entity types!)
- Typically occur in
  - Hierarchies
    - Example: animal species / animal genus
  - Conversion of n-ary (degree n) relationships into binary relationships
    - Example of care order

Another example of a weak entity type

- Example of "care order" weak entity type



The multiplex cinema

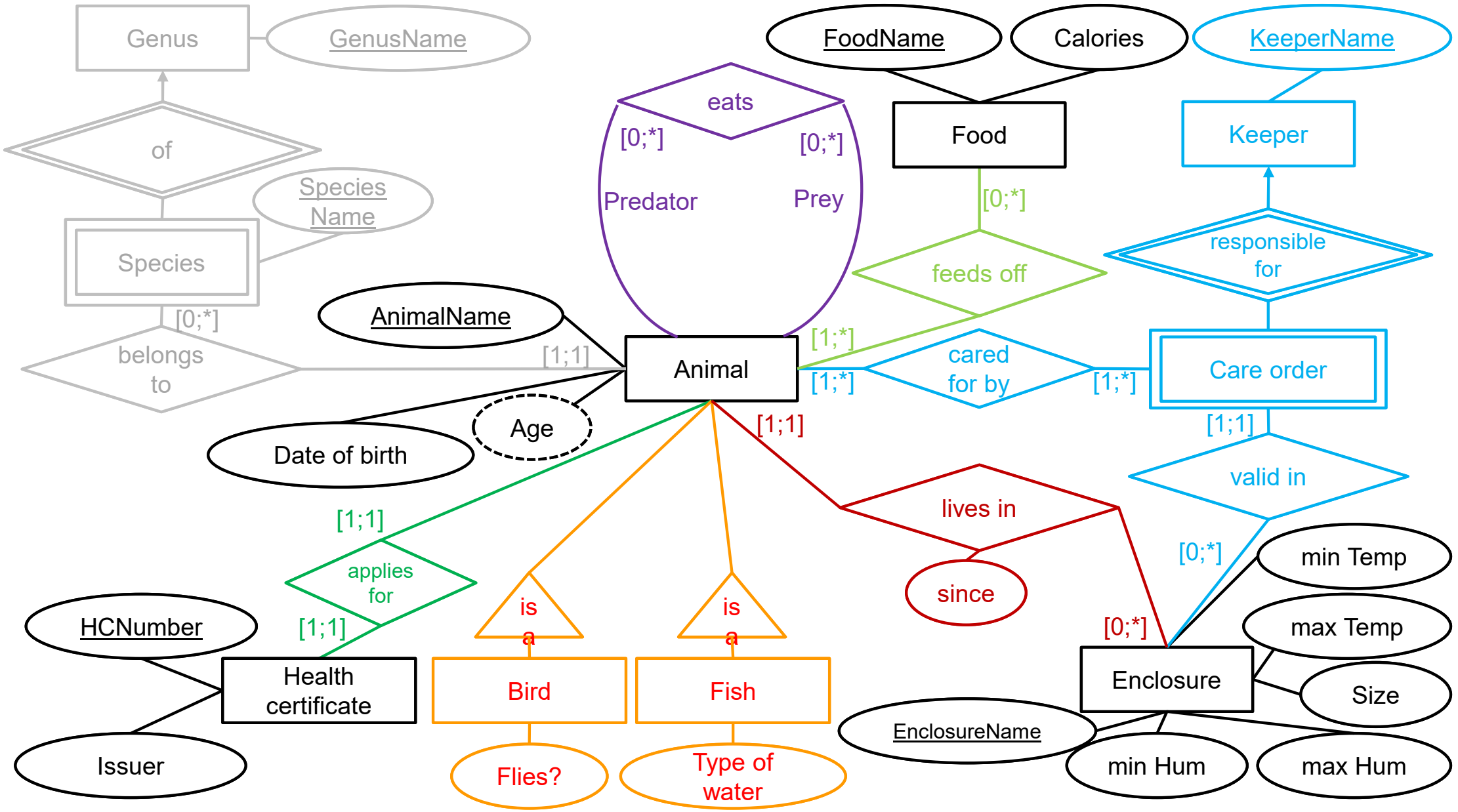
Let’s look at the operator of a multiplex cinema. The multiplex cinema has several auditoriums. Each auditorium has a name, a unique number and contains a number of seats. Each film has a name, was produced in a certain year in a certain language and has a unique code. A film showing takes place on a particular day at a particular time in an auditorium. Obviously only one film can be shown in an auditorium at any one time.

How do we model a good ER diagram?

- Correctness
  - Map reality correctly
- Avoid redundancies
  - Otherwise we will get a non-BCNF schema later (→ anomalies)
- Simplicity
  - Do not use unnecessary entity types/relationships, etc.
- Use the correct relationships
  - Often we can use different ones, some of them may be redundant (→ anomalies)
- Use the correct elements
  - Attribute or entity types+relationship – attributes simpler, but only if no dependencies (FDs!) are created (→ anomalies)
- Choose meaningful names

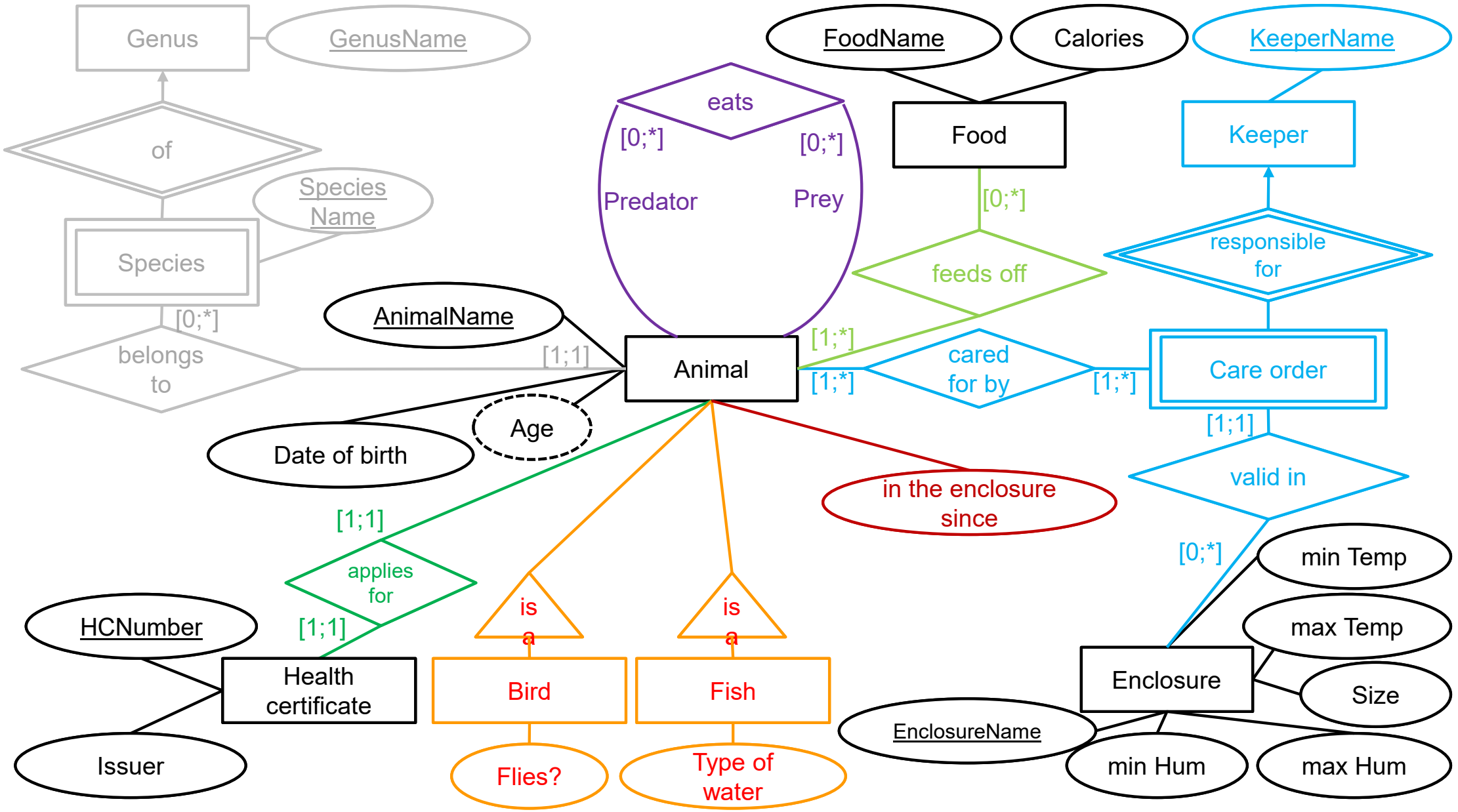
Summary of the zoo example

Comment: age = year(today-DOB)



Summary of the zoo example

Comment: age = year(today-DOB)



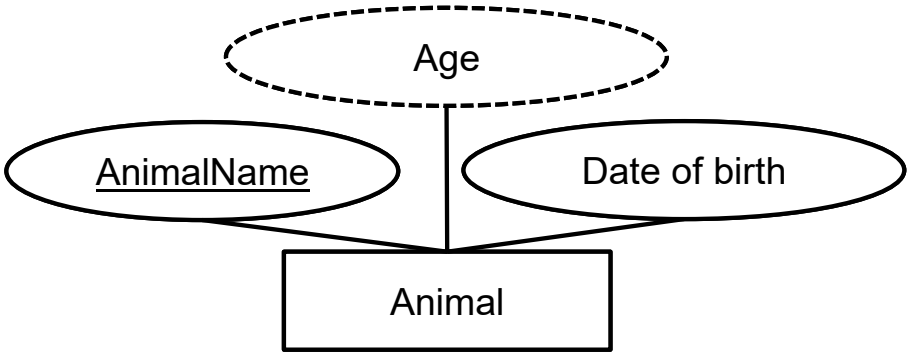
ER mapping

- First partial step of the logical database design
- Mapping of ER model to relational model
- In principle, it's simple:
  - Every entity type becomes a relation
  - Every relationship becomes a relation
- But
  - Merging of relations
  - Weak entity types
  - is-a relationships

Mapping of (simple) entity types

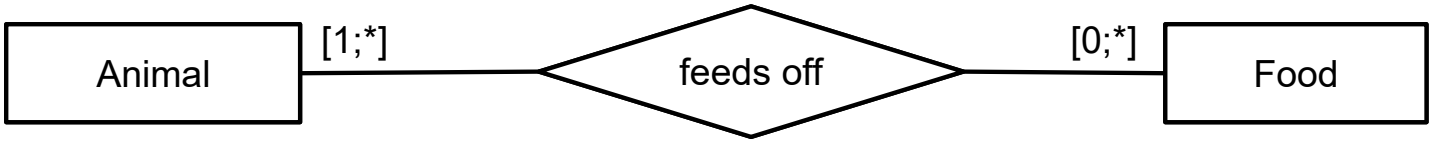
- Every entity type becomes a relation
  - Attributes: all non-computed attributes of the entity type
  - Key: key of the entity type
  - Foreign keys: none

- Example



ANIMAL = {AnimalName, Date_of_birth} with
K_{Animal} = {{AnimalName}}

Mapping of n:m relationships

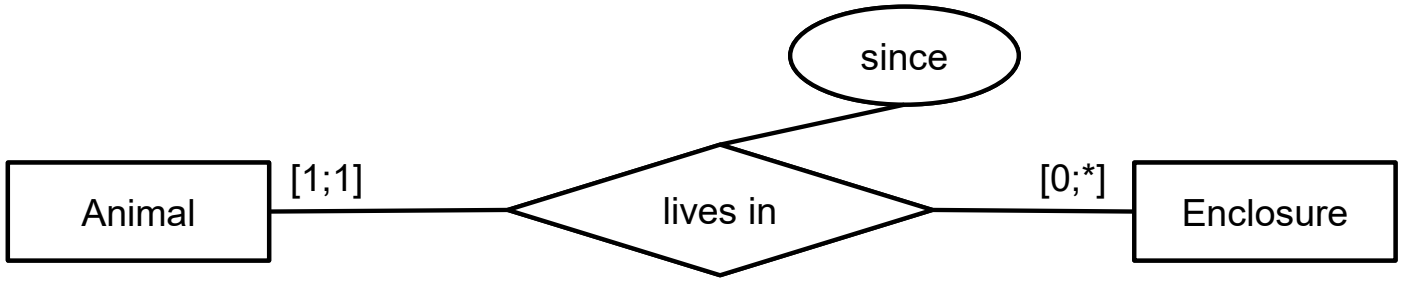


FEEDS_OFF = {AnimalName, FoodName} with

K_{FEEDS_OFF} = {{AnimalName, FoodName}}

{AnimalName} references ANIMAL(AnimalName)
{FoodName} references FOOD(FoodName)

Mapping of 1:n relationships

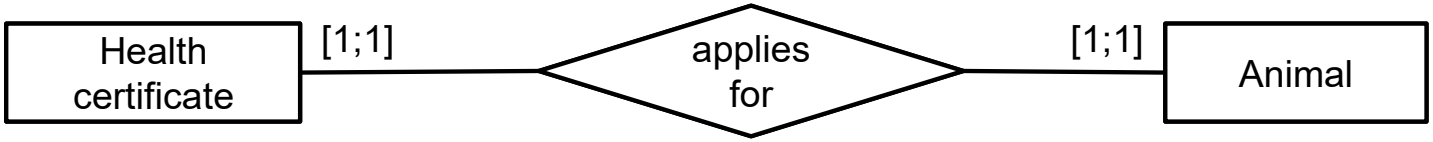


LIVES_IN = {AnimalName, EnclosureName, since} with

K_{LIVES_IN} = {{AnimalName}}

{AnimalName} references ANIMAL(AnimalName)
{EnclosureName} references ENCLOSURE(EnclosureName)

Mapping of 1:1 relationships



APPLIES_FOR = {HCNumber, AnimalName} with

$K_{APPLIES\_FOR}$ = {{HCNumber}, {AnimalName}}

{HCNumber} references HEALTH_CERTIFICATE(HCNumber)
{AnimalName} references ANIMAL(AnimalName)

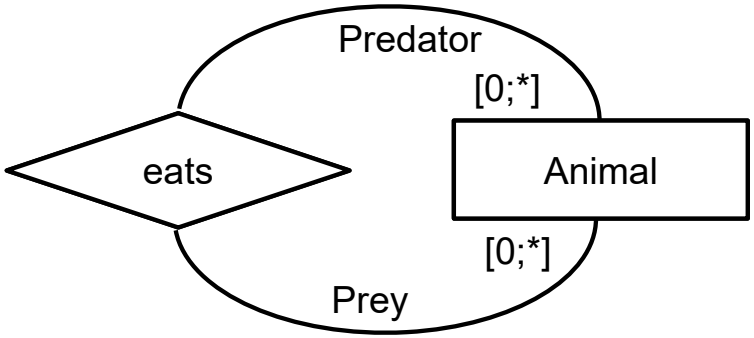
Mapping of relationships

- Every relationship becomes a relation
  - Attributes: keys of the entity types involved + non-computed attributes of the relationship
  - Foreign keys: attributes of the entity types involved reference their respective entity type relations

- Determining the keys of the new relational schemas
  - **m:n relationship**
    Primary keys of the two entity types **together** ➔ **one** key
  - **1:n relationship**
    Primary key of the **n side** ➔ key in the new relational schema
    - In the functional notation, the side without arrowhead
    - In the [min;max] notation, the side with [1;1] or [1;0]
  - **1:1 relationship**
    Primary keys of the two entity types ➔ **each have one** key in the new relational schema, primary keys then selected from these keys

Mapping of recursive n:m relationships

- In addition: rename the attribute example

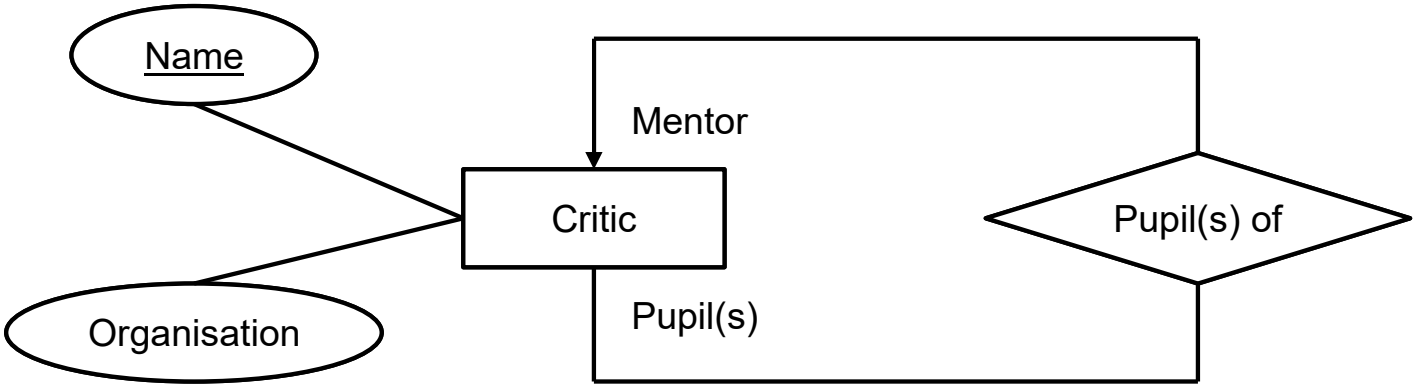


EATS = {Predator, Prey} with

K_{EATS} = {{Predator, Prey}}

{Predator} references ANIMAL(AnimalName)
{Prey} references ANIMAL(AnimalName)

Mapping of recursive 1:n relationships



CRITIC = {Name, Organisation, Mentor_name} with

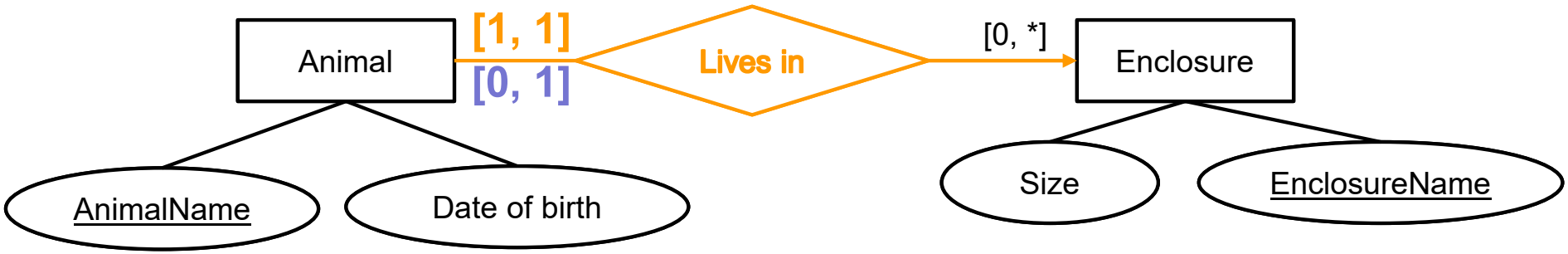
K_{CRITIC} = {{Name}}

{Mentor_name} references CRITIC({Name})

Merging of relations

- Often the resulting relational schemas are not optimal
  - **Merging**: combine relations, set keys sensibly, keep foreign keys (if still meaningful)
  - **m:n relationship** - do not merge
  - **1:n relationship**
    Merge with the entity type relation of the n side
    (or with the 1 side in the min/max notation), keep the keys
    - If [0;1]: null values for the foreign key allowed
    - If [1;1]: specify "not null"
  - **1:1 relationship**
    Merge non-optional sides with respective entity type relations
    - If [0;1] - [0;1]: do not merge
    - If [1;1] - [0;1]: always: merge the [1;1] side; keys: from entity type relation; optional: also merge [0;1] side using null values
    - If [1;1] - [1;1]: merge all three, keys: from relationship

Merging of 1:n relationships



- **[1, 1]:** Merge `ANIMAL` and `LIVES_IN`, not null

  `ANIMAL' = {AnimalName, Date_of_birth, EnclosureName}`

  $K_{ANIMAL'}$ `= {{AnimalName}}`

  `{EnclosureName} references ENCLOSURE({EnclosureName})`

  `EnclosureName not null`

- **[0, 1]:** Merge `ANIMAL` and `LIVES_IN`, null allowed

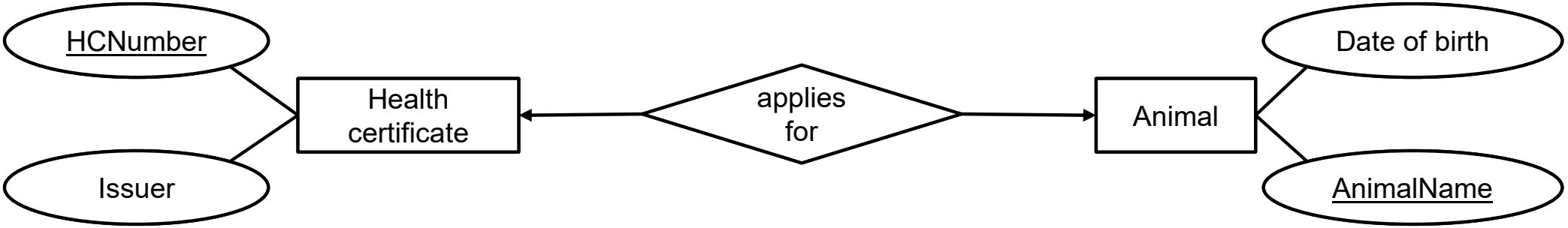
  `ANIMAL' = {AnimalName, Date_of_birth, EnclosureName}`

  $K_{ANIMAL'}$ `= {{AnimalName}}`

  `{EnclosureName} references ENCLOSURE({EnclosureName})`

  `EnclosureName null` (usually not specifically stated)

Attempt: Merging of 1: 1 relationships (1)



- Merged relation:

ANIMAL

AnimalName	Date_of_birth	HCNumber	Issuer
Leo	1.1.2000	42-007	H. Huber
Hilde	11.11.1999	42-009	M. Müller

- Animals without a health certificate require null values:

ANIMAL

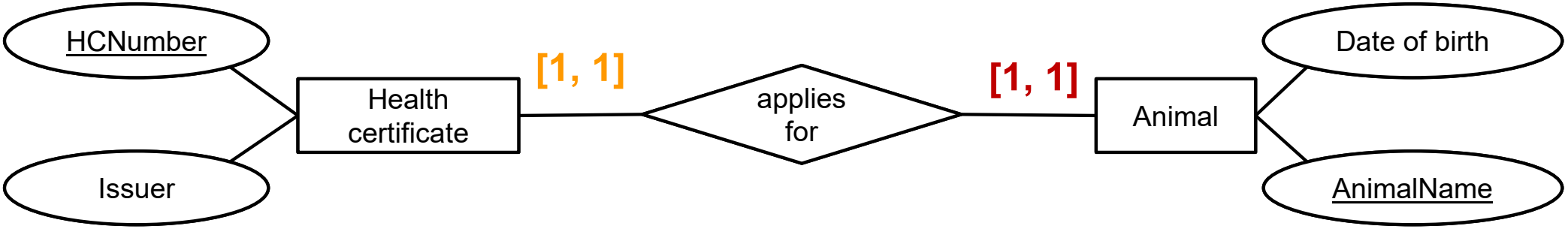
AnimalName	Date_of_birth	HCNumber	Issuer
Leo	1.1.2000	42-007	H. Huber
Hilde	11.11.1999	⊥	⊥

- Health certificates without animals lead to further null values:

ANIMAL

AnimalName	Date_of_birth	HCNumber	Issuer
Leo	1.1.2000	42-007	H.Huber
Hilde	11.11.1999	⊥	⊥
⊥	⊥	42-003	S. Schmidt

Merging of 1: 1 relationships (2)

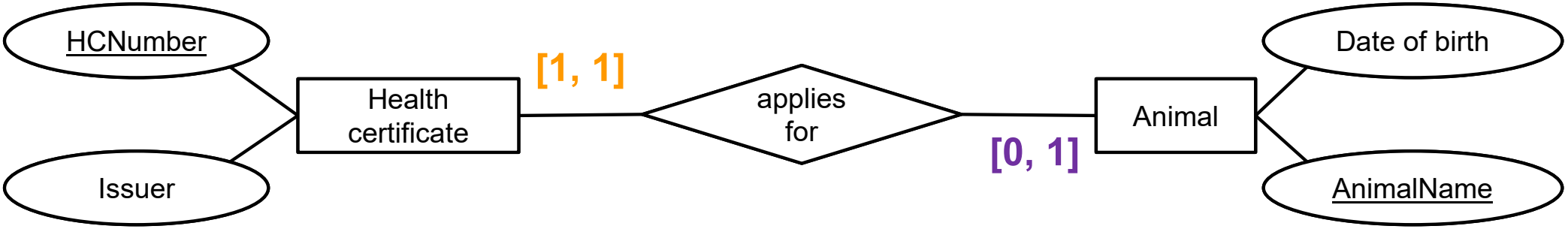


- **[1, 1]** to **[1, 1]**: merging of all three (`HEALTH_CERTIFICATE`, `APPLIES_FOR`, `ANIMAL`) possible

  `ANIMAL' = {AnimalName, Date_of_birth, HCNumber, Issuer}`

  $K_{ANIMAL'} = \{\{AnimalName\}, \{HCNumber\}\}$

Merging of 1: 1 relationships (3)



- **[1, 1]** to **[0, 1]**: HEALTH_CERTIFICATE and APPLIES_FOR merge possible

  HEALTH_CERTIFICATE‘ = {HCNumber, Issuer, AnimalName}

  $K_{HEALTH\_CERTIFICATE'}$ = {{HCNumber}}

  {AnimalName} references ANIMAL({AnimalName})

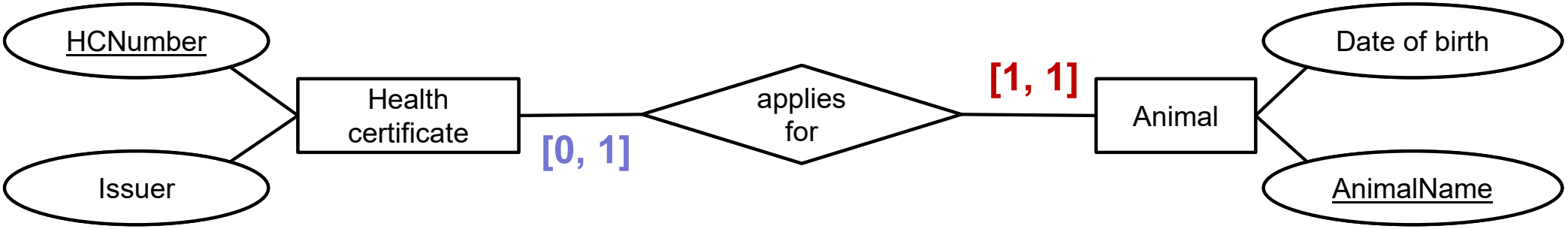
  (AnimalName not null)

  Optional:

  ANIMAL‘ = {AnimalName, Date_of_birth, HCNumber, Issuer}

  $K_{ANIMAL'}$ = {{AnimalName}}; {HCNumber} unique, null

Merging of 1: 1 relationships (4)



- **[0, 1]** to **[1, 1]**: ANIMAL and APPLIES_FOR merge possible

  ANIMAL' = {AnimalName, Date_of_birth, HCNumber}

  $K_{ANIMAL'}$ = {{AnimalName}}

  {HCNumber} references HEALTH_CERTIFICATE({HCNumber})

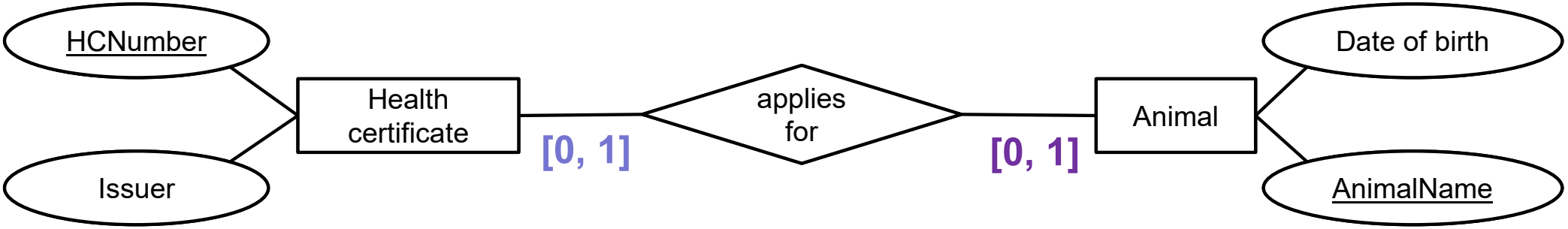
  (HCNumber not null)

  Optional:

  ANIMAL'' = {AnimalName, Date_of_birth, HCNumber, Issuer}

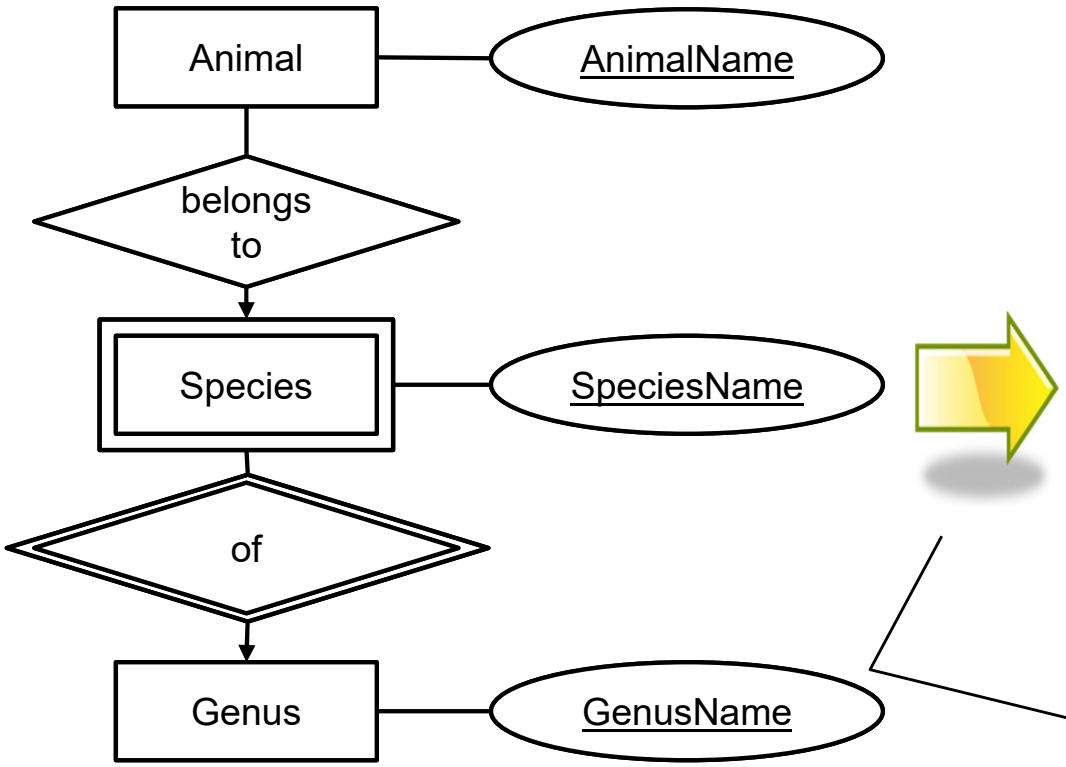
  $K_{ANIMAL''}$ = {{HCNumber}; {AnimalName} unique, null

Merging of 1: 1 relationships (5)



- [0, 1] to [0, 1]: do not merge

Mapping of weak entity types



Genus = {GenusName} with
K_{Genus} = {{GenusName}}

SPECIES = {SpeciesName, GenusName} with
$K_{SPECIES}$ = {{SpeciesName, GenusName}}

{GenusName} references GENUS({GenusName})

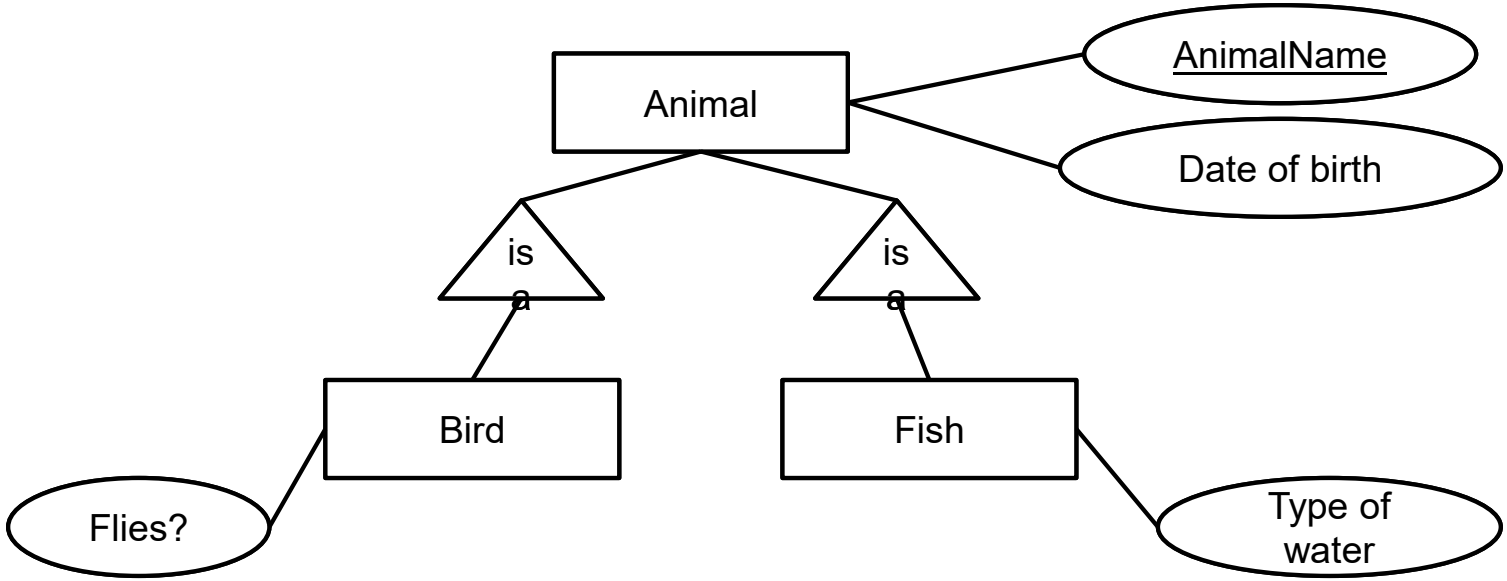
- Attribute GenusName in SPECIES is foreign key for relation GENUS
- **NO** relation is created for "of"!

General

1) No relations for supporting relationships
2) Relation of the weak entity type contains own attributes + all keys of entity types pointed to by the supporting relationships

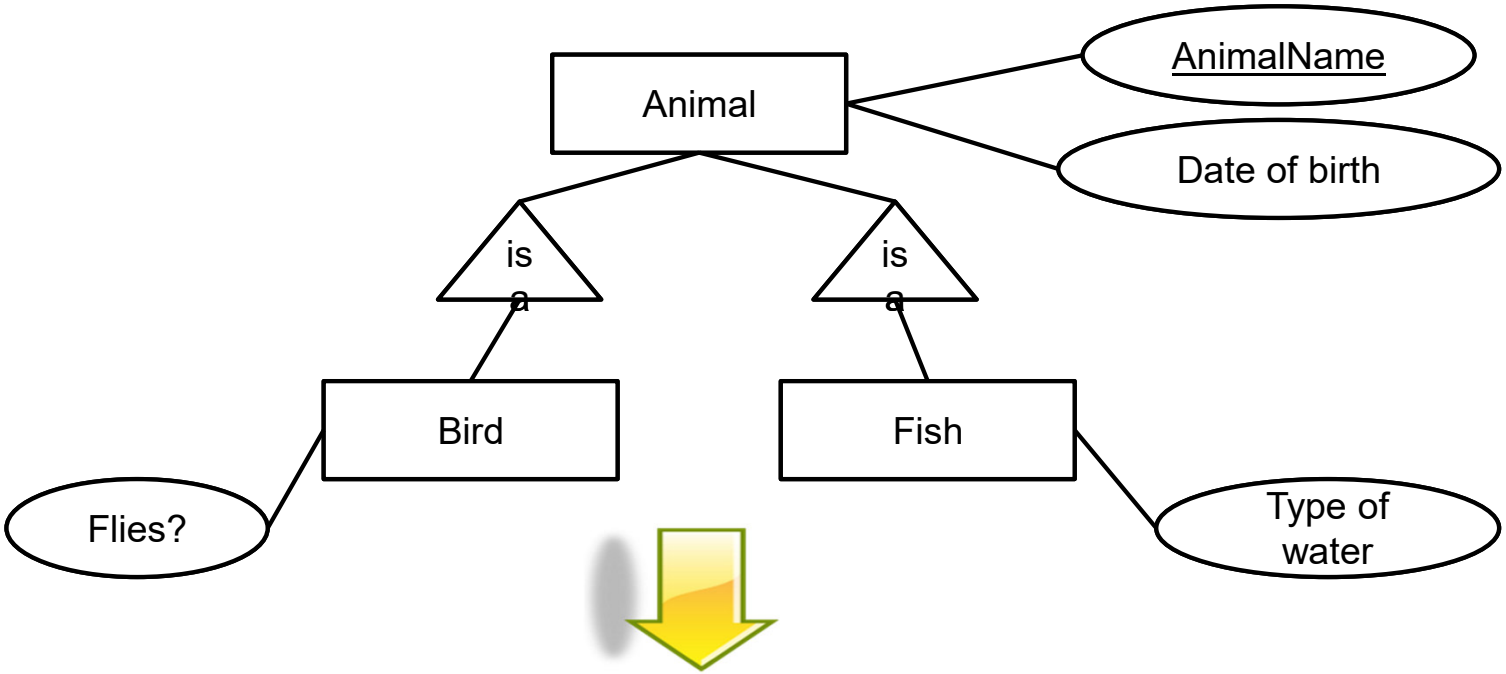
Mapping of is-a relationships

- Three possible mappings
  - ER style
  - OO style
  - null-value style



Mapping of is-a relationships: ER style

- ER style: one relation per entity type

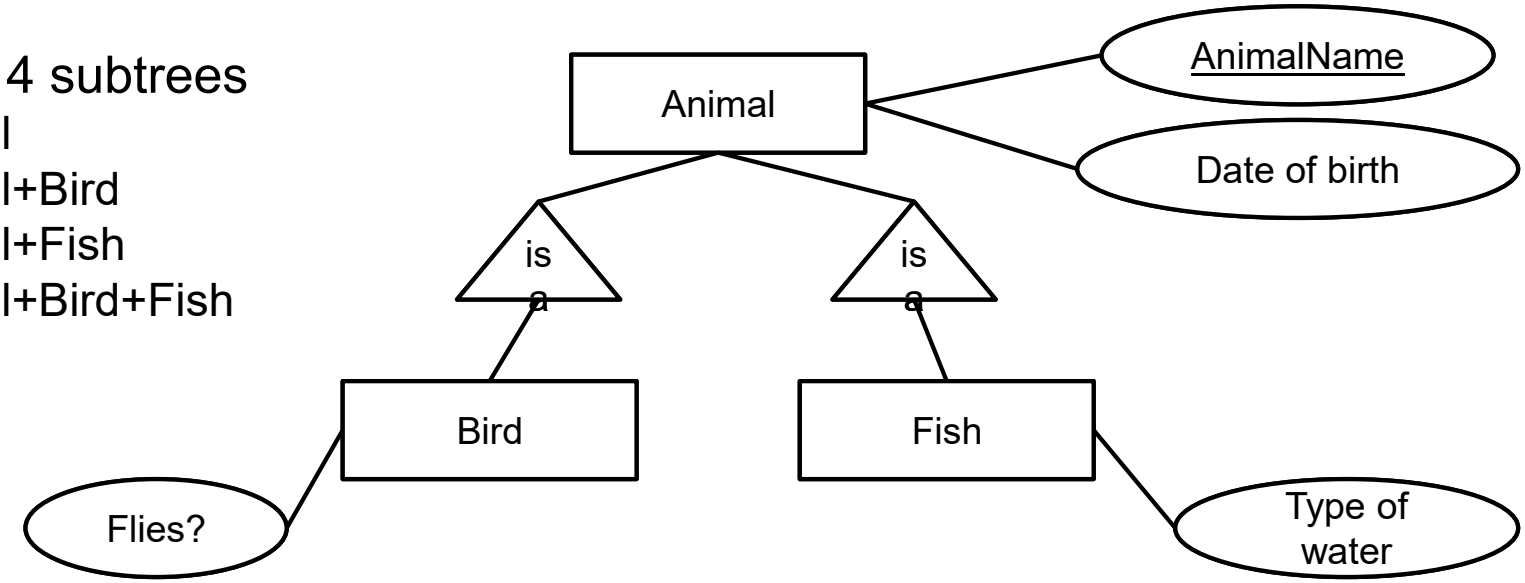



```
ANIMAL = {AnimalName, Date_of_birth}
BIRD = {AnimalName, flies}
FISH = {AnimalName, Type_of_water}
```

Note:
Every fish/bird has an extra tuple in ANIMAL!

Mapping of is-a relationships: OO style

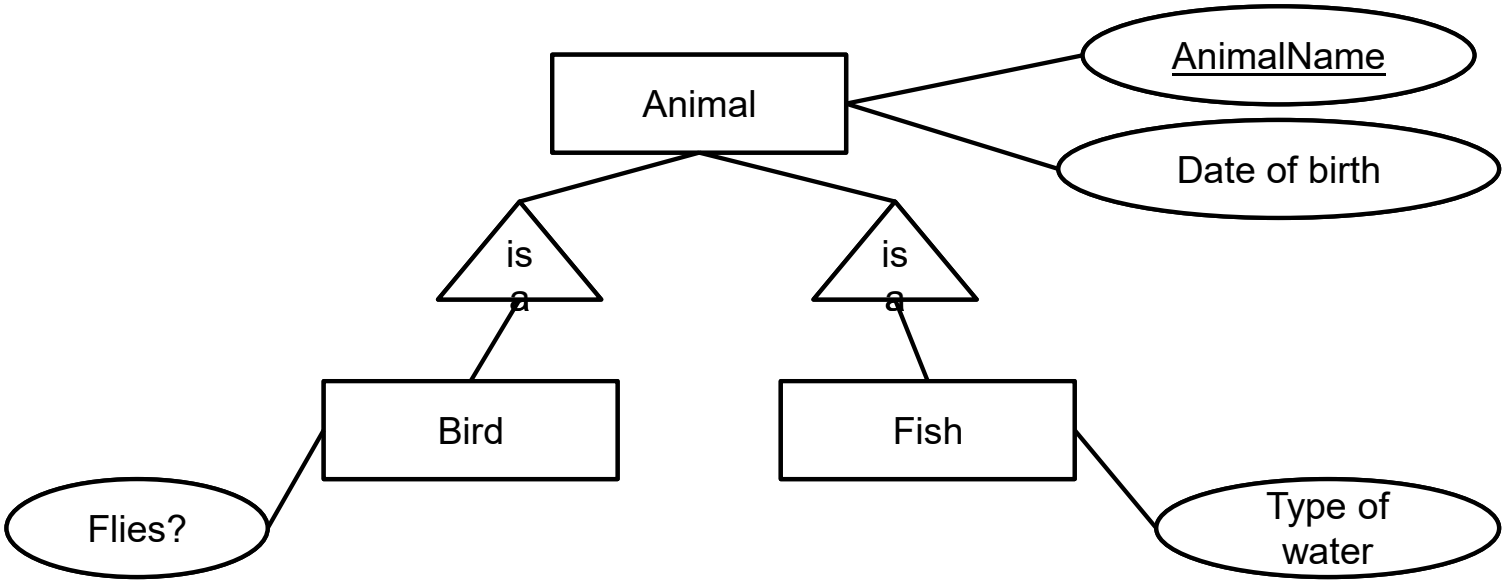
- OO style: one relation per subtree
- Example: 4 subtrees
  - Animal
  - Animal+Bird
  - Animal+Fish
  - Animal+Bird+Fish




```
ANIMAL   = {AnimalName, Date_of_birth}
ANIMALB  = {AnimalName, Date_of_birth, flies}
ANIMALF  = {AnimalName, Date_of_birth, Type_of_water}
ANIMALBF = {AnimalName, Date_of_birth, flies , Type_of_water}
```

Mapping of is-a relationships: null-value style

- null-value style: only one relation, null for n/a attributes




```
ANIMAL   = {AnimalName, Date_of_birth, flies , Type_of_water}
```

- Every fish has `null` in `flies`
- Every bird has `null` in `Type_of_water`
- Neither fish nor bird has `null` in both `flies` and `Type_of_water`

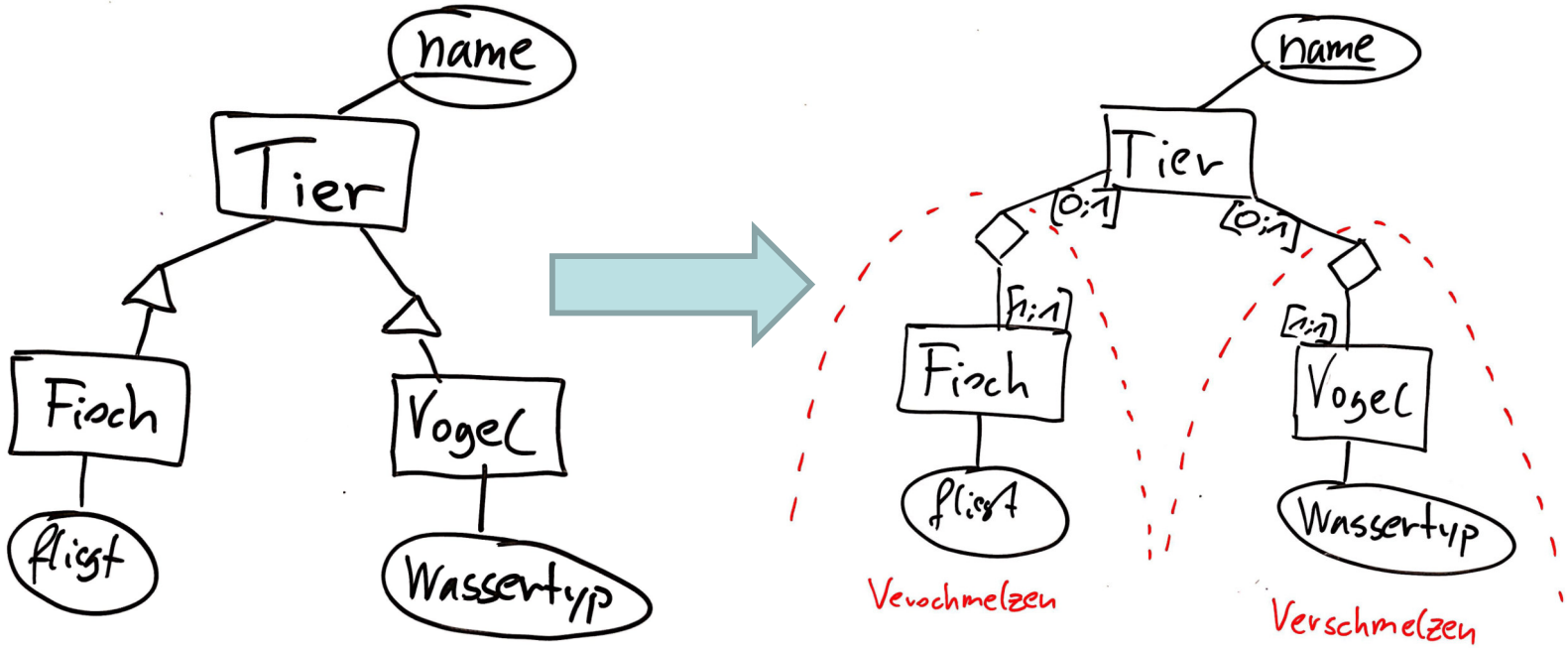
Mapping of is-a relationships - discussion

- Every type of mapping has advantages and disadvantages
  - Fewer relations are generally better (fewer joins required)
    - null-value style is good
    - OO style is bad as there are very many relations
    - ER is OK
  - Less space required is generally better
    - OO with only one tuple with exactly the right attributes is thus very good
    - null-value with only one tuple, but that is long
    - ER with many tuples, but only the key is repeated
  - Fast query execution
    - It depends very much on the query!

➔ No clear winner, depends on the application!

An alternative way to map *is-a* relationships

- *is-a* relationships can be derived by
  - null-value style: a big table with everything in it
  - OO style: one table per combination
  - ER style: one “boss” table and additionally one for each type
- You can also arrive at ER mapping via relationships and merging:



Overview of the transformations

ER concept	is mapped to relational concept
Entity type E_i Attributes of E_i Primary key P_i	Relational schema R_i Attributes of R_i Primary key P_i
Relationship type its attributes 1 : n 1 : 1 m : n	Relational schema Attributes: P_1, P_2 more attributes P_2 becomes primary key of the relationship P_1 and P_2 become keys of the relationship $P_1 \cup P_2$ becomes primary key of the relationship
is-a relationship	R_1 gets additional key P_2

- with
  - E_1, E_2 : entity types involved in relationship,
  - P_1, P_2 : their primary keys,
  - 1 : n relationship: E_2 is n side,
  - is-a relationship: E_1 is a more special entity type

Practical approach to database design

- We now know 2 ways to design a database:
  1) From a set of attributes and a set of FDs directly using normalisation (BCNF, 3NF / decomposition, synthesis)
  2) Using an ER model and (semi-)automatic transformation into a relational model (RM)

  ➔ Question: which approach is "the right one"?

- Typical (recommended) approach
  - Prepare an ER model
  - Transform into a relational model
  - If the ER model is good, the RM produced is usually automatically in BCNF!
  - (Automated) check whether the RM is in BCNF (or at least in 3NF)
  - If not: indication of a design flaw in the ER, therefore: go back to the ER model and improve this (very rarely: elimination of the redundancies in the RM)

  ➔ Normal form theory and normalisation is an important tool for designing a good ER model!

Summary

- ER model
  - Entity types
  - Relationships (cardinalities: n:m, 1:n, 1:1)
  - n-ary (degree n) relationships and conversion into binary relationships
  - Recursive relationships
  - Weak (non-identifying) relationships
  - is-a relationship
- Good ER models
- Transformation of an ER model into a relational model
  - Entity types, relationships
  - Merging
  - Mapping of weak (non-identifying) relationships
  - Mapping of is-a relationships