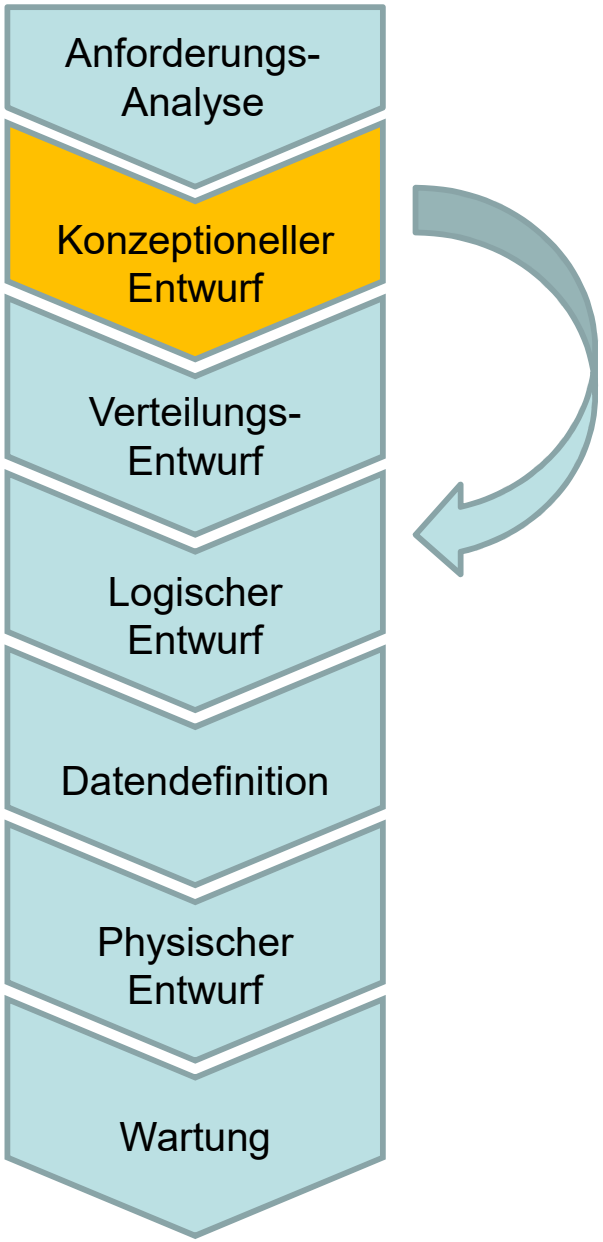
Technische Hochschule Rosenheim

Kapitel 6 – Konzeptioneller Datenbankentwurf

Vorlesung Datenbanken

Prof. Dr. Kai Höfig

Entity-Relationship (ER) Modell



- Das Entity-Relationship Modell erlaubt uns mittels einfacher Sprachkonstrukte Entitäten, Attribute und Beziehungen von Daten graphisch darzustellen (**Syntax**)
- Daraus lässt sich dann in einem (teil-)automatisierten Verfahren ein relationales Datenbankmodell ableiten (**Semantik**)

Graphische Notation und semantische Bedeutung des ER Modells

- Entity-Typen bilden die Objekte der realen Welt. Aus ihnen werden später Relationen abgeleitet.
  - Doppelte Umrandung kennzeichnet *weak entitiy*

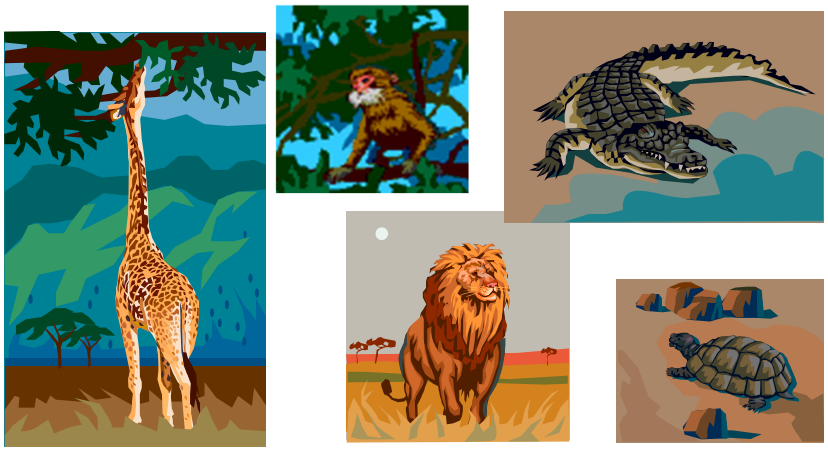
- Attribute sind Eigenschaften von Entity-Typen. Sie sind später auch Attribute der Relationen.
  - Handelt es sich um ein Schlüsselattribut, wird der Bezeichner unterstrichen.
  - Gestrichelte Umrandung bedeutet, dass es sich um ein berechnetes Attribut handelt.

- Relationships modellieren Beziehungen zwischen Entity-Typen. Sie können im Relationenmodell zu Relationen oder zu Attributen werden.
  - Doppelte Umrandung kennzeichnet *supporting relationship*

- Is-a Realtionships werden zur Modellierung von Generalisierungen und Spezialisierungen verwendet. Sie sammeln im Relationenmodell Attribute zu einem Entity-Typ

Beispiel – die Zoo-DB

- Sie sollen eine Datenbank für die Verwaltung des Rosenheimer Zoos erstellen.



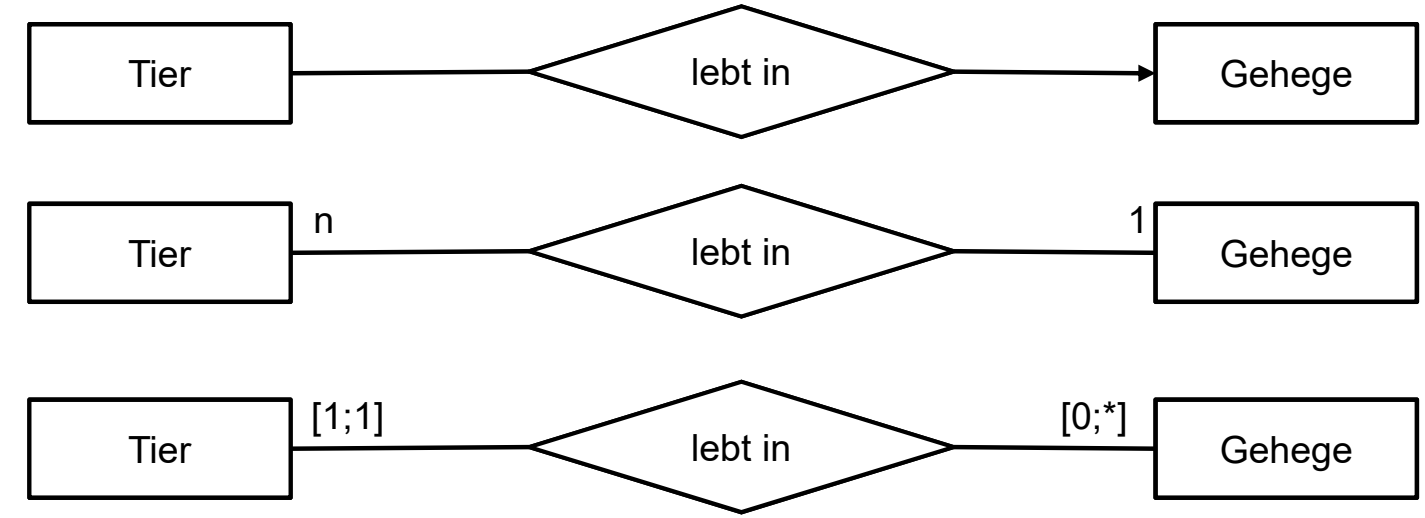
- Sie führen Gespräche mit Mr. P., dem Zoodirektor, um die Anforderungen an die Datenbank herauszufinden.

- Mr. P:

Hallo Datenbankprofi, vielen Dank, dass Sie mir helfen werden!

Kardinalität von Relationships: 1:n

In jedem Gehege leben mehrere Tiere.



1:n Relationship

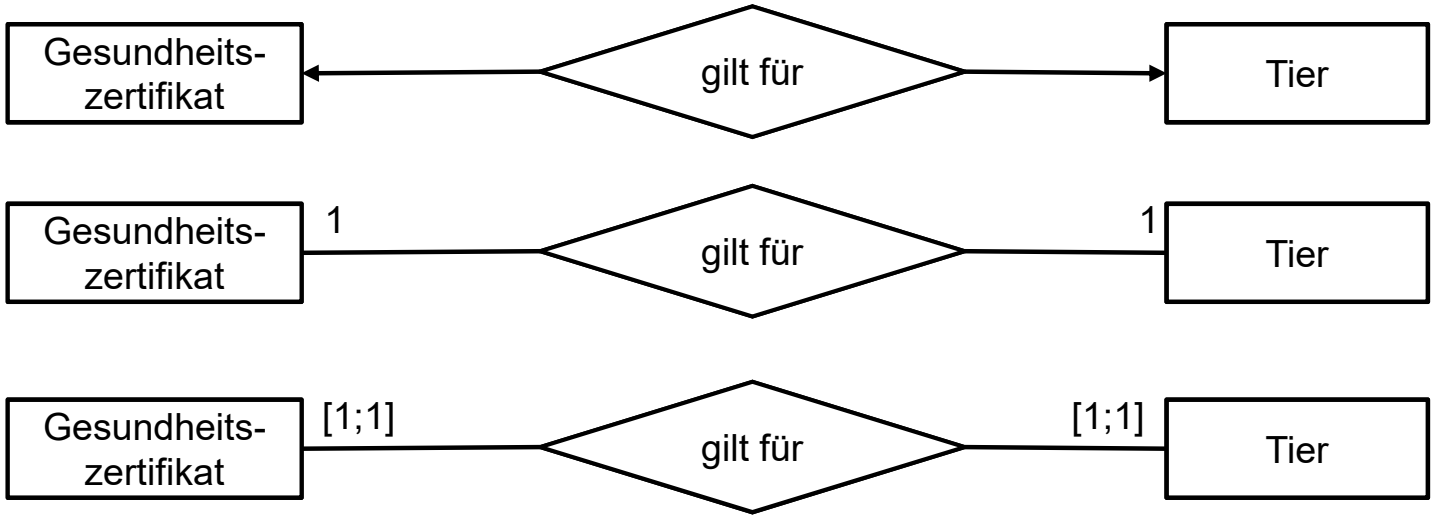
Jedem Entity e_1 vom Entity-Typ E_1 sind beliebig viele Entities E_2 zugeordnet, aber zu jedem Entity e_2 gibt es maximal ein e_1 aus E_1

Beispiele

- Lieferant liefert Produkt
- Mutter hat Kinder

Kardinalität von Relationships: 1:1

Jedes Tier bekommt vom Tierarzt ein eigenes Gesundheitszertifikat.



1:1 Relationship

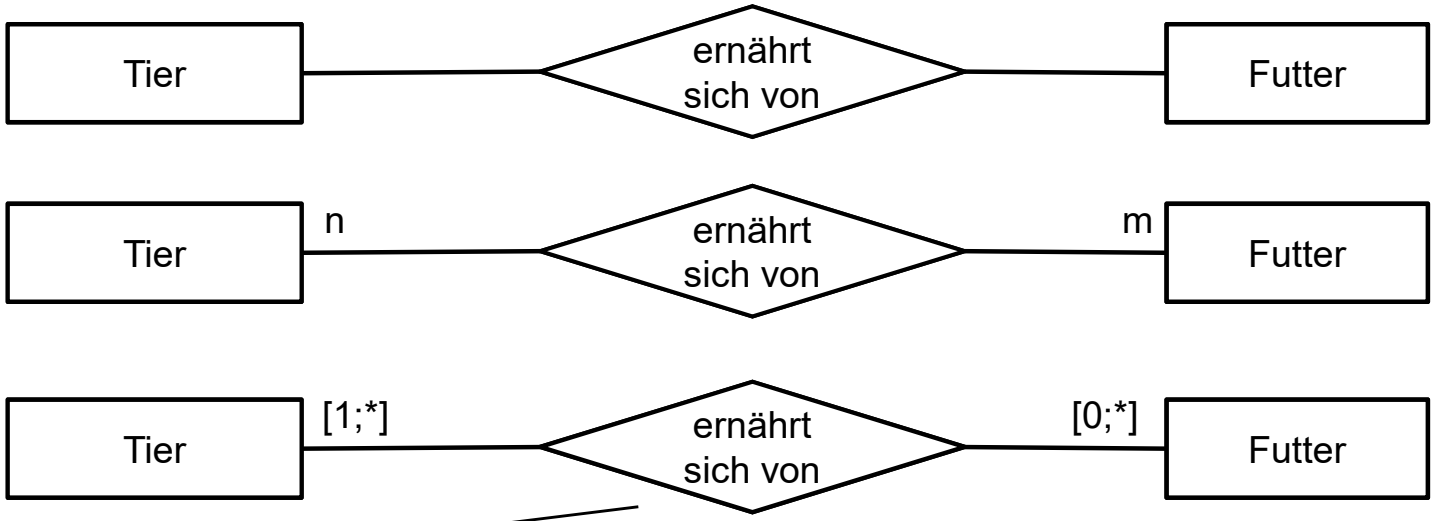
Jedem Entity e_1 vom Entity-Typ E_1 ist maximal ein Entity e_2 aus E_2 zugeordnet und umgekehrt

Beispiele

- Prospekt beschreibt Produkt,
- Mann ist verheiratet mit Frau

Kardinalität von Relationships: n:m

Wir haben verschiedene Futter für die Tiere, wobei ein Tier verschiedene Futter mag und ein Futter für mehrere Tiere gut ist.



n:m Relationship

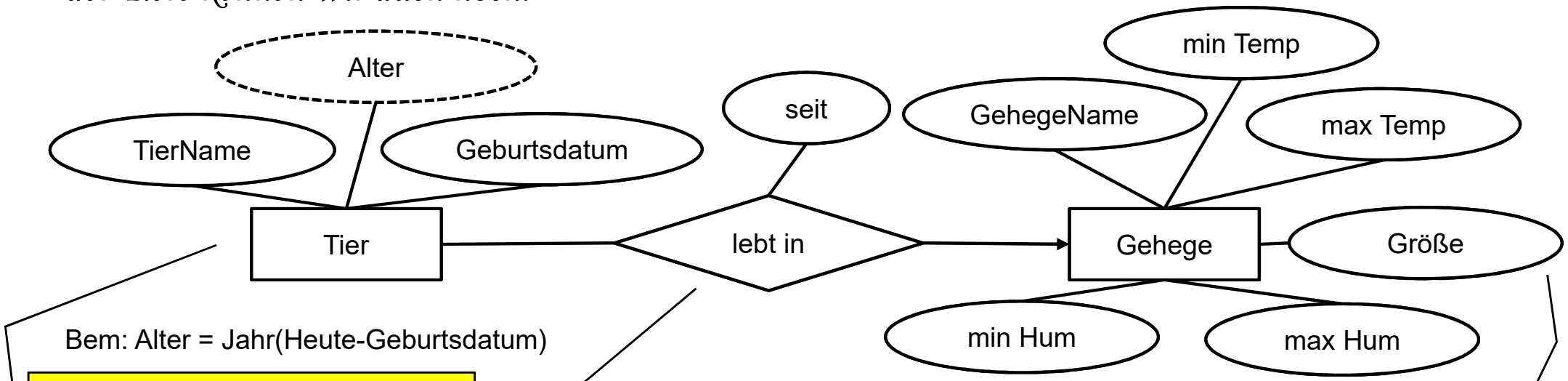
Jedem Entity e_1 vom Entity-Typ E_1 ist beliebig vielen Entities e_2 aus E_2 zugeordnet und umgekehrt

Beispiele

Bestellung umfasst Produkte

Entity-Typen und Relationships

Wir haben viele Tiere in dem Zoo, die leben in Gehegen. Tiere haben Namen und Gehege, und auch eine Größe. In jedem Gehege leben mehrere Tiere. Für jedes Gehege haben wir eine maximale Temperatur, eine minimale Temperatur, und dasselbe mit der Luftfeuchtigkeit. Das Alter der Tiere kennen wir auch noch.



Entity-Typ: Objekt der realen/Vorstellungswelt, über das Informationen zu speichern sind

Beispiele: **Tier**, **Gehege**; aber auch Informationen über Ereignisse, wie z.B. **Bestellungen**

Relationship: beschreibt Beziehung zwischen Entities

Beispiele: ein Tier **lebt** in einem Gehege, Bestellung wird **aufgegeben von** Kunde

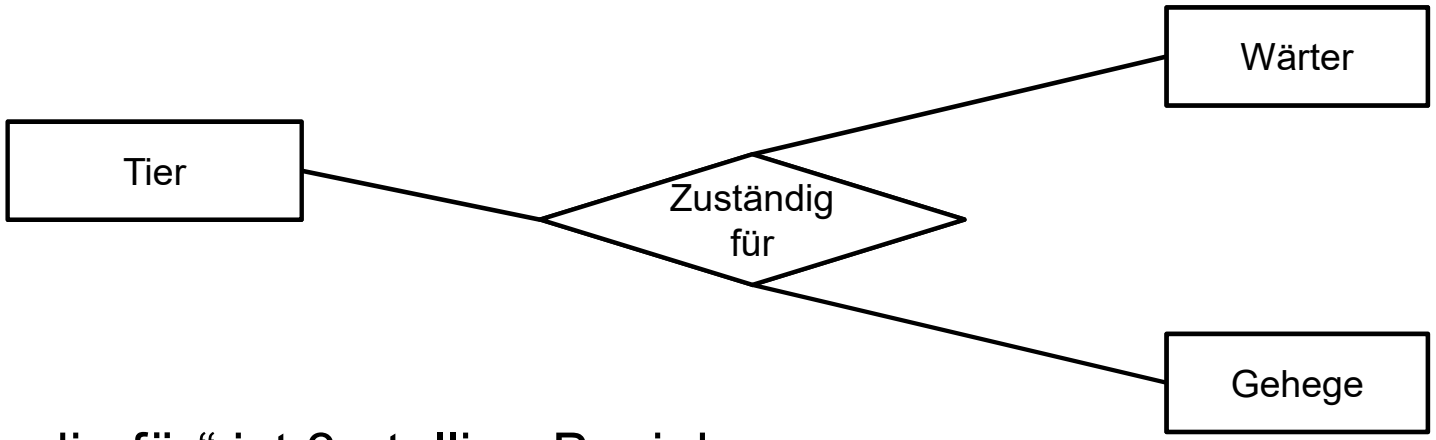
Attribut: repräsentiert eine Eigenschaft von Entity oder Beziehung

Beispiele: **Name** eines Tieres oder **Datum** einer Bestellung

Stelligkeit von Relationships

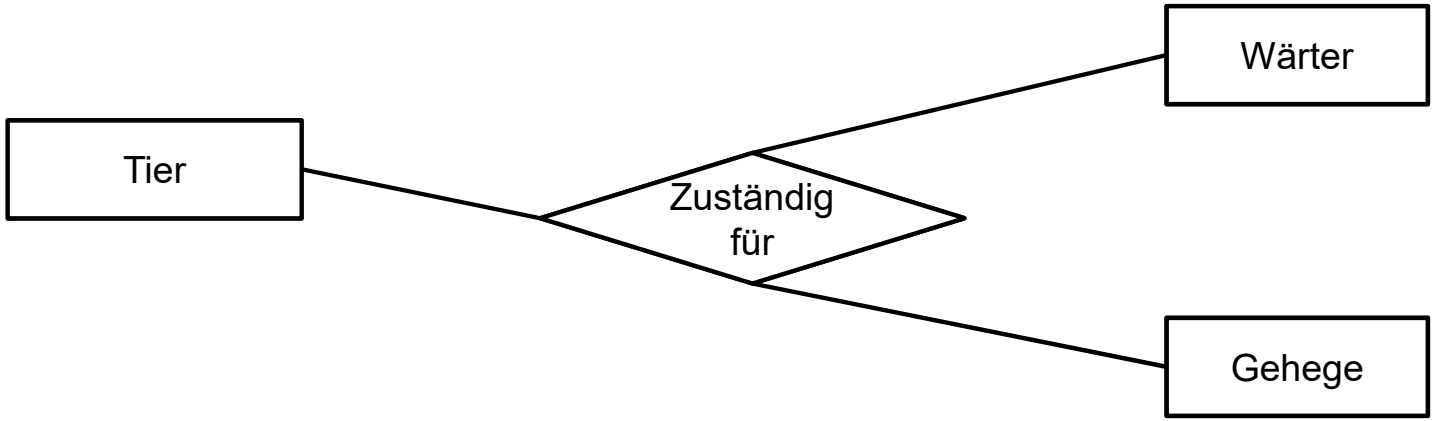
- Stelligkeit (oder Grad) eines Relationships
  - Anzahl der beteiligten Entity-Typen (häufig: binär)
  - Beispiel: Lieferant liefert Produkt, Tier lebt in Gehege

Jeder unserer Wärter ist für mehrere Tiere zuständig, die aber alle in einem Gehege leben.

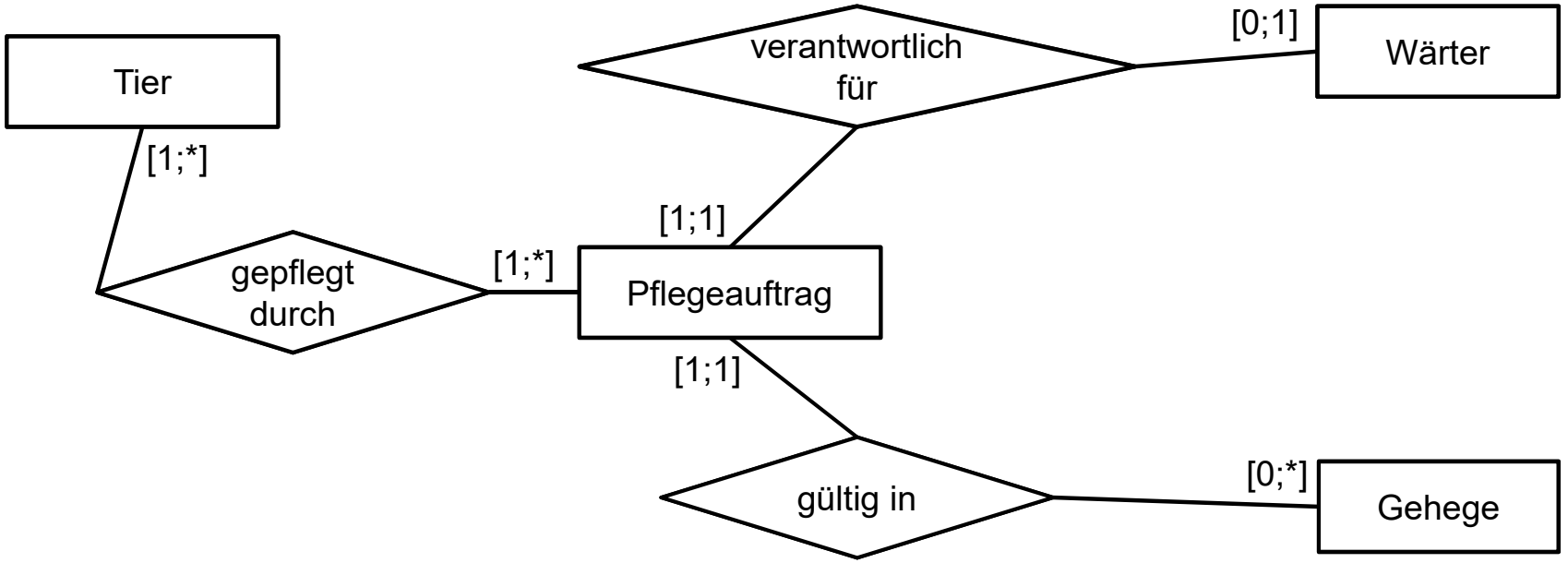


- „zuständig für“ ist 3-stellige Beziehung
  - Beachte: Kardinalitäten nicht eindeutig darstellbar!

Modellierung mehrstelliger Relationships durch binäre Relationships

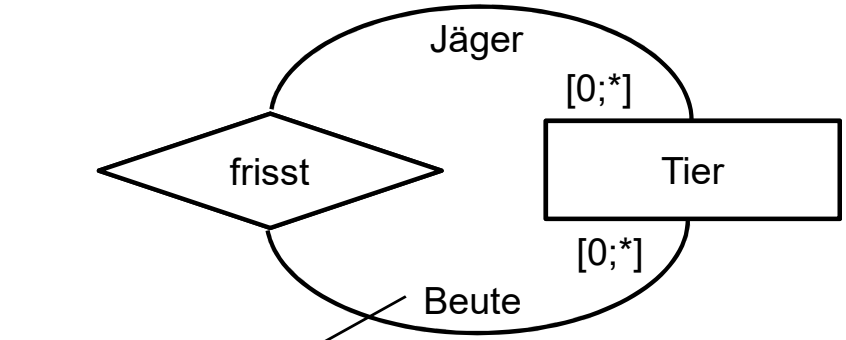


- Alternative Modellierung der „zuständig für“ Beziehung



Rollen in Relationships

Ach ja, wir müssen auch noch aufpassen, dass wir keine Tiere zusammen in ein Gehege sperren, wo das eine Tier das andere auffrisst!

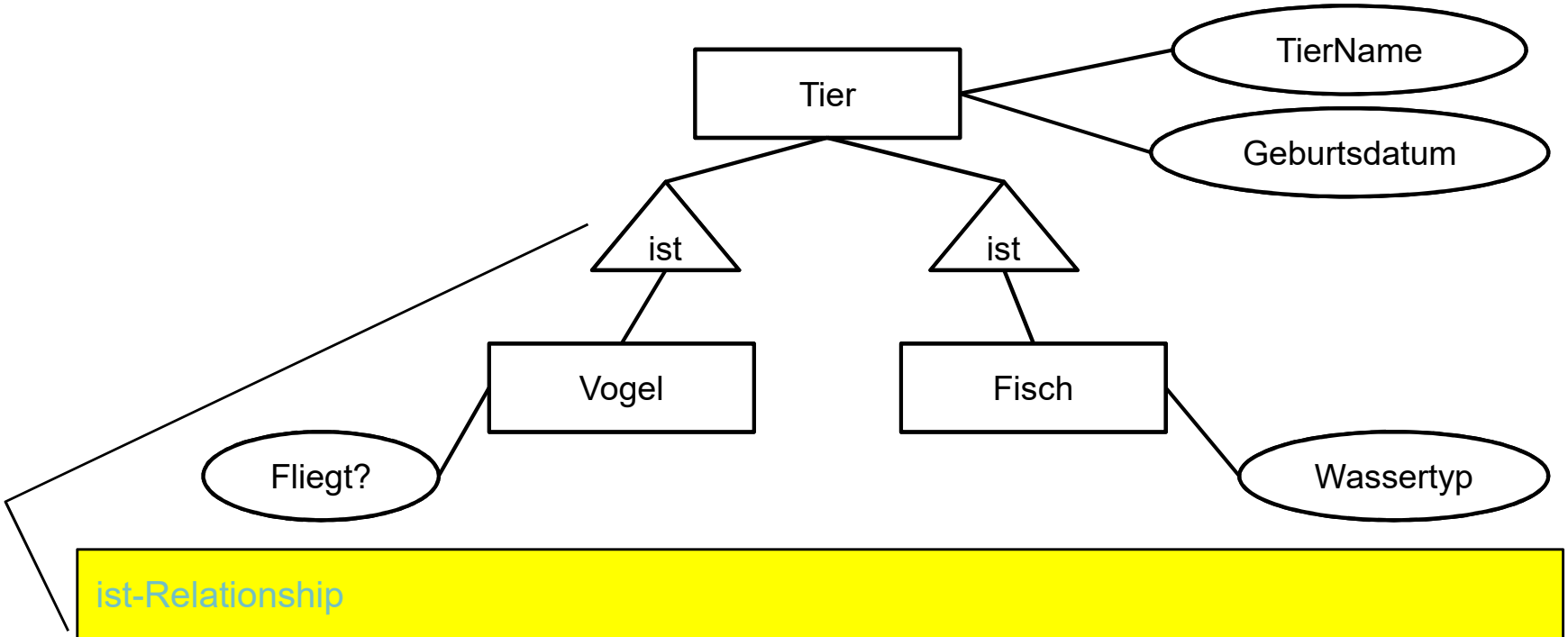


Rollen

Kommt ein Entity-Typ mehrfach in einem Relationship vor, müssen wir zur Unterscheidung der Vorkommen Rollennamen vergeben.

„ist“ (is-a) Relationships

Wir müssen auch noch aufpassen, dass wir Vögel und Fische speziell betrachten, Fische brauchen Süß- oder Salzwasser, und bei Vögeln ist wichtig ob sie fliegen können oder nicht.

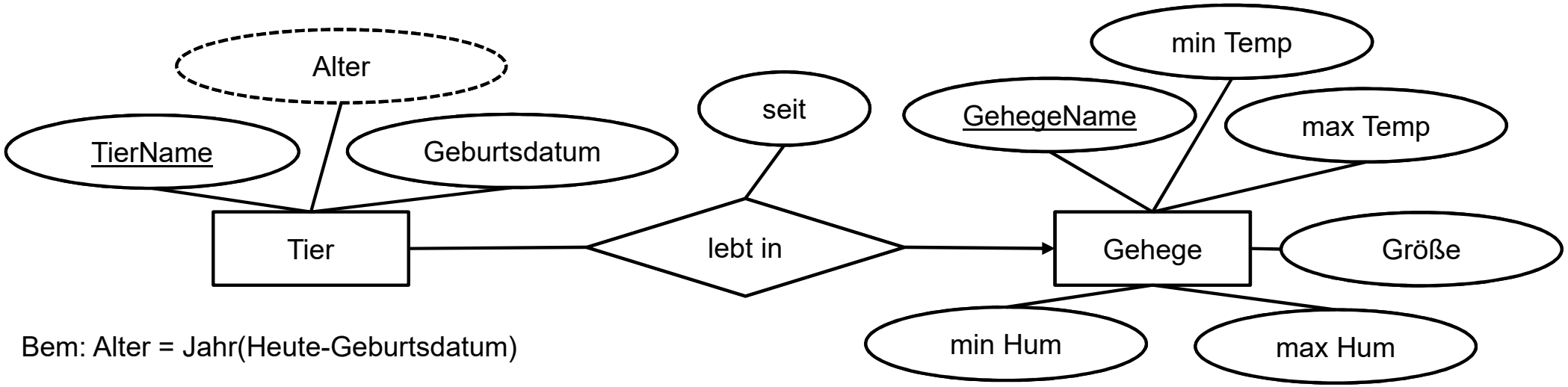


ist-Relationship

- Entspricht Generalisierung/Spezialisierung
- Immer 1:1 (daher nicht extra gekennzeichnet)
- Attribute werden „aufgesammelt“
- Bei uns: immer eindeutiger „Vater“

Schlüssel im ER Modell

- Schlüssel = Unterstreichen aller beteiligter Attribute



Bem: Alter = Jahr(Heute-Geburtsdatum)

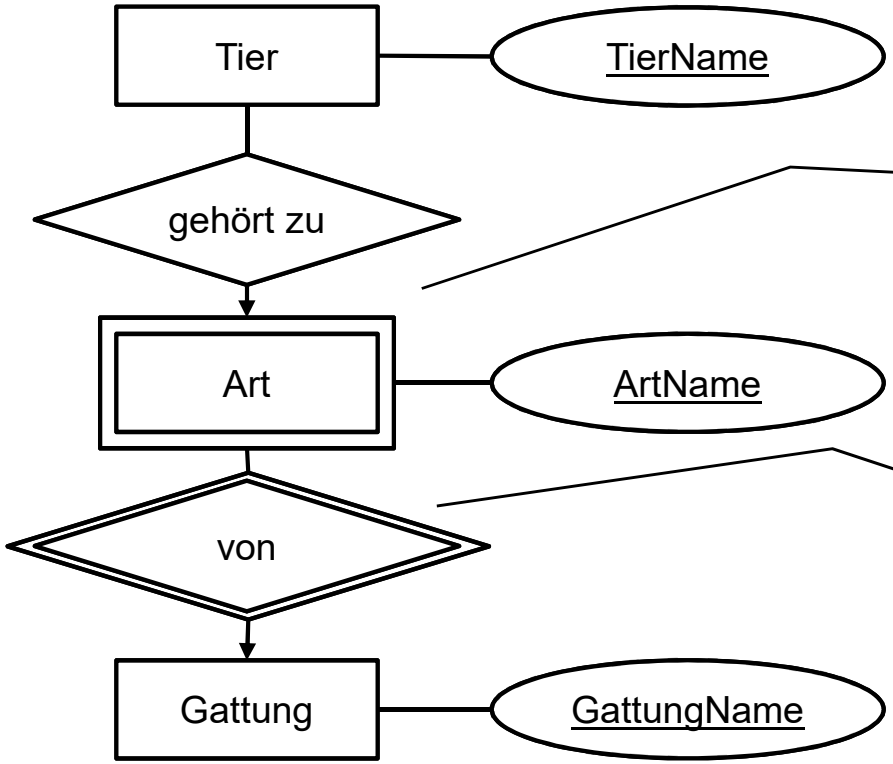
- Bemerkungen
  - i.allg. nur Primärschlüssel gekennzeichnet
  - Bei ist-Relationships muss der oberste Entity-Typ alle relevanten Attribute enthalten

Abhängige Entity-Typen („weak entity sets“) (1)

Jedes Tier gehört zu einer Tierart, und jede Tierart zu einer Gattung. Wobei die gleiche Art zu verschiedenen Gattungen gehören kann, deswegen beschreibt nur die Kombination von Art und Gattung das Tier genau.

>> „Wie Bitte, das verstehe ich nicht, können Sie mir ein Beispiel geben?“

Gerne: Cyriocosmus elegans ist eine Tarantel, Centruroides elegans ist ein Skorpion. Dabei ist “elegans” die Art, und “Cyriocosmus “ bzw. “Centruroides “ die Gattung.



Abhängige Entity-Typen (weak entity set)

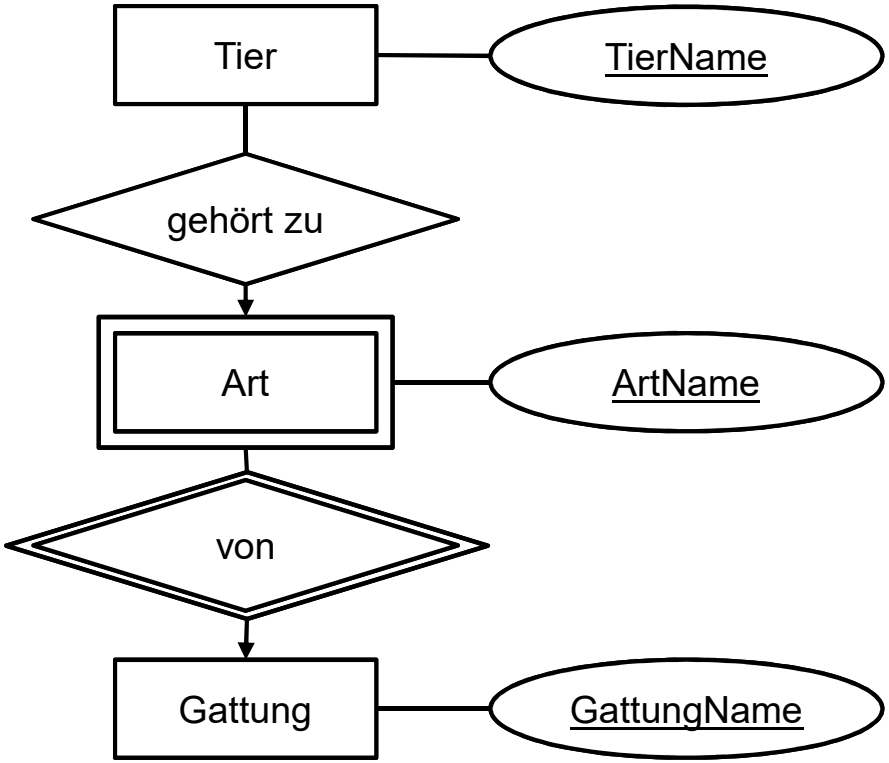
- Schlüssel ist (ArtName, GattungName)

Supporting Relationship

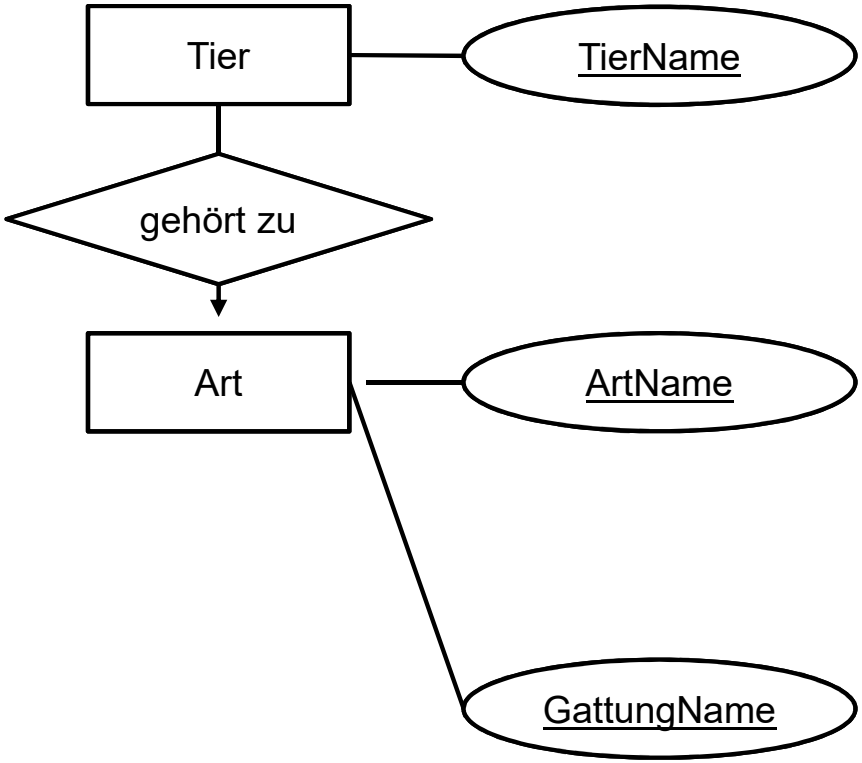
- Immer 1:n
- Im verbundenen Entity-Typ sind weitere Schlüsselattribute des abhängigen Entity-Typs

Abhängige Entity-Typen („weak entity sets“) (2)

Abhängige Entity-Typen sollten auch nicht überstrapaziert werden. Im Prinzip können sehr viele Attribute ausgelagert werden in eigene Relationen. Das macht aber nur Sinn, wenn auch zusätzliche Daten zu der neuen Entität gespeichert werden. Wenn also Gattung nur aus einem einzelnen Attribut besteht, gibt es auch eine alternative Implementierung:



Implementierung mit abhängigem Entity Typ



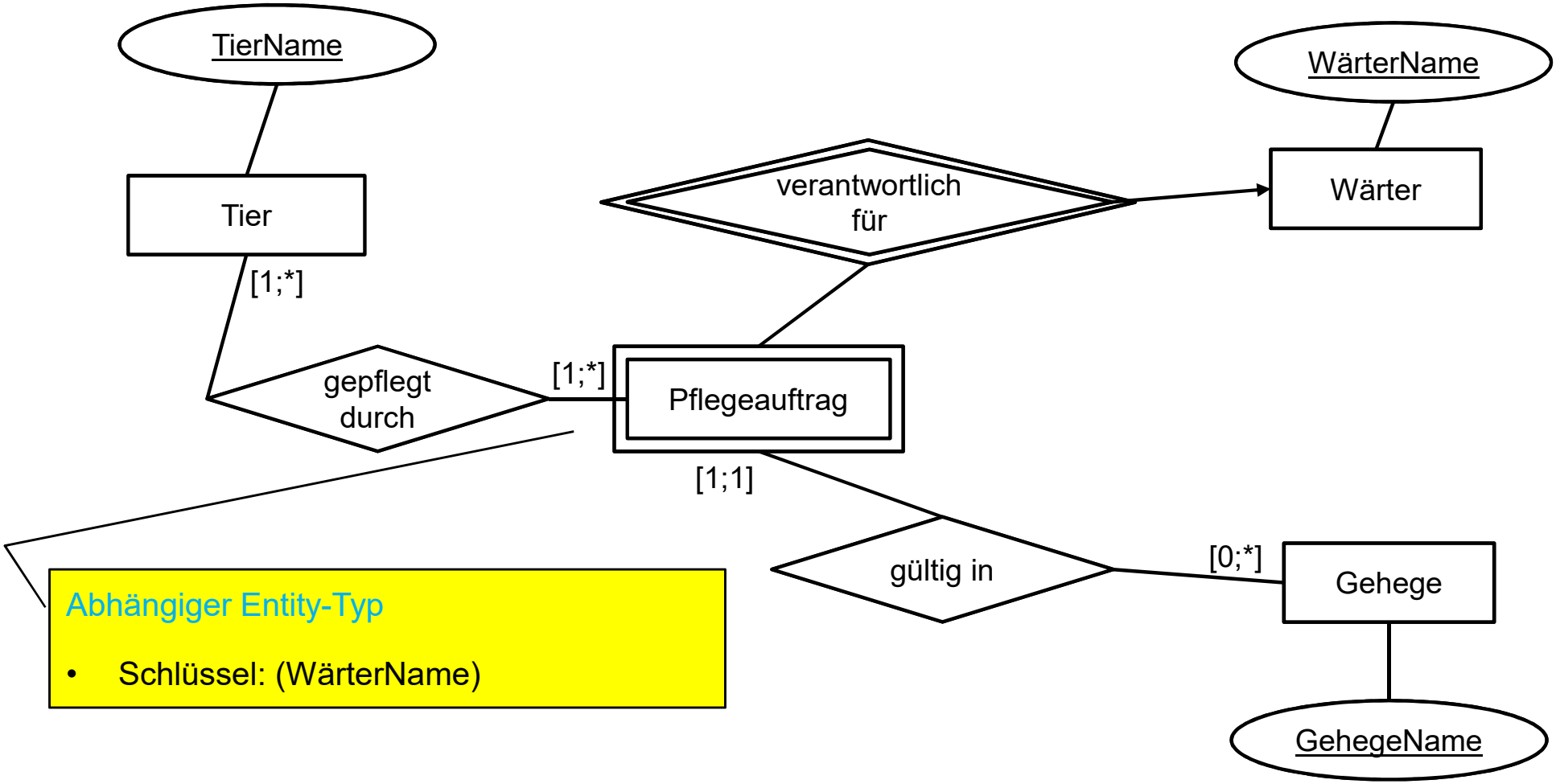
Alternative Implementierung als Attribut

Abhängige Entity-Typen („weak entity sets“) (3)

- Abhängiger Entity-Typ (Weak entity set): enthält nicht alle Attribute, die für seinen Schlüssel nötig sind
  - Durch doppelte Umrandung gekennzeichnet
  - Einige Schlüsselattribute sind in anderen Entitäten enthalten
  - Daher nötig: 1:n Relationships („supporting Relationships“, Rauten mit doppelter Umrandung) zu diesen anderen Entity-Typen!
  - Wie erkennt man, dass eine Entity weak ist? Kein Schlüssel in seinen Attributen!
- Schlüssel eines abhängigen Entity-Typs
  - Ev. eigenen Attribute (unterstrichen)
  - Alle Schlüssel der durch supporting Relationships verbundenen Entity-Typen (und das rekursiv, falls diese auch abhängige Entity-Typen sind!)
- Auftreten typischerweise in
  - Hierarchien
    - Beispiel: Tierart / Tiergattung
  - Umwandlung von mehrstelligen Relationships in binäre Relationships
    - Beispiel Pflegeauftrag

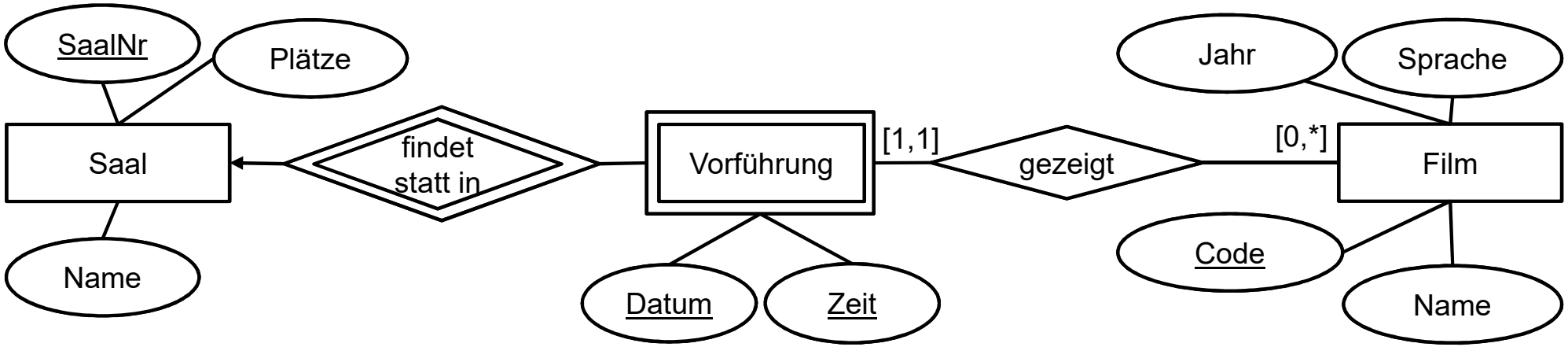
Weiteres Beispiel für einen abhängigen Entity-Typen

- Beispiel abhängiger Entity-Typ „Pflegeauftrag“



Das Multiplex-Kino

Betrachten Sie einen Multiplex-Kino Betreiber. Im Multiplex-Kino gibt es mehrere Säle. Jeder Saal hat einen Namen, eine eindeutige Nummer und enthält eine Anzahl Sitzplätze. Jeder Film hat einen Namen, wurde in einem Jahr in einer Sprache produziert und hat einen eindeutigen Code. Die Vorführung eines Films findet an einem bestimmten Tag zu einer bestimmten Uhrzeit in einem Saal statt. Zu jedem Zeitpunkt kann natürlich nur ein Film in einem Saal gezeigt werden.

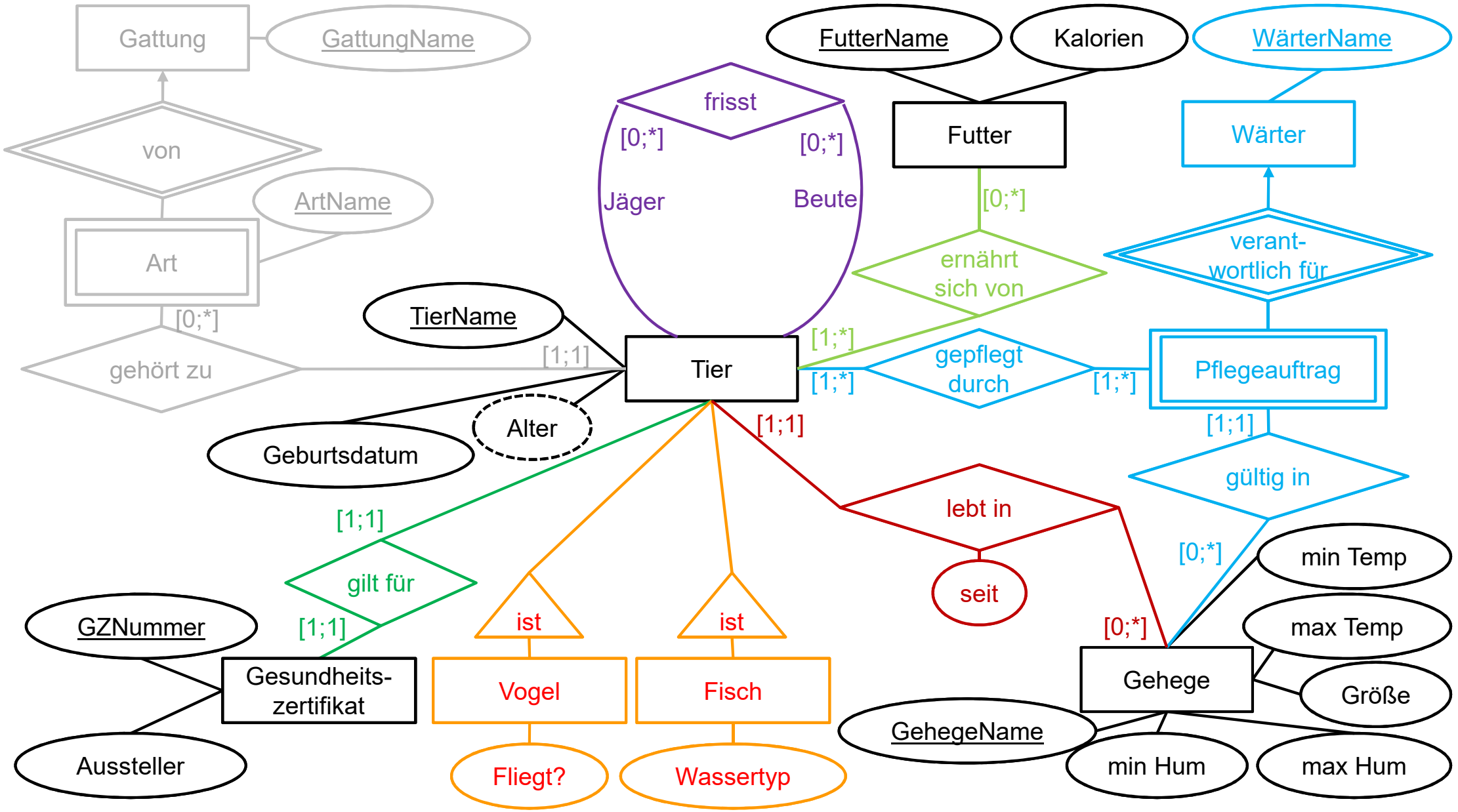


Wie modelliert man ein gutes ER Diagramm?

- Korrektheit
  - Realität richtig abbilden
- Redundanzen vermeiden
  - Sonst erhält man später ein nicht BCNF Schema (→ Anomalien)
- Einfachheit
  - Keine unnötigen Entity-Typen/Relationships etc. verwenden
- Die richtigen Relationships verwenden
  - Häufig kann man unterschiedliche verwenden, einige davon können redundant sein (→ Anomalien)
- Die richtigen Elemente verwenden
  - Attribute oder Entity-Typen+Relationship – Attribute einfacher, aber nur wenn keine Abhängigkeiten (FDs!) entstehen (→ Anomalien)
- Aussagekräftige Namen wählen

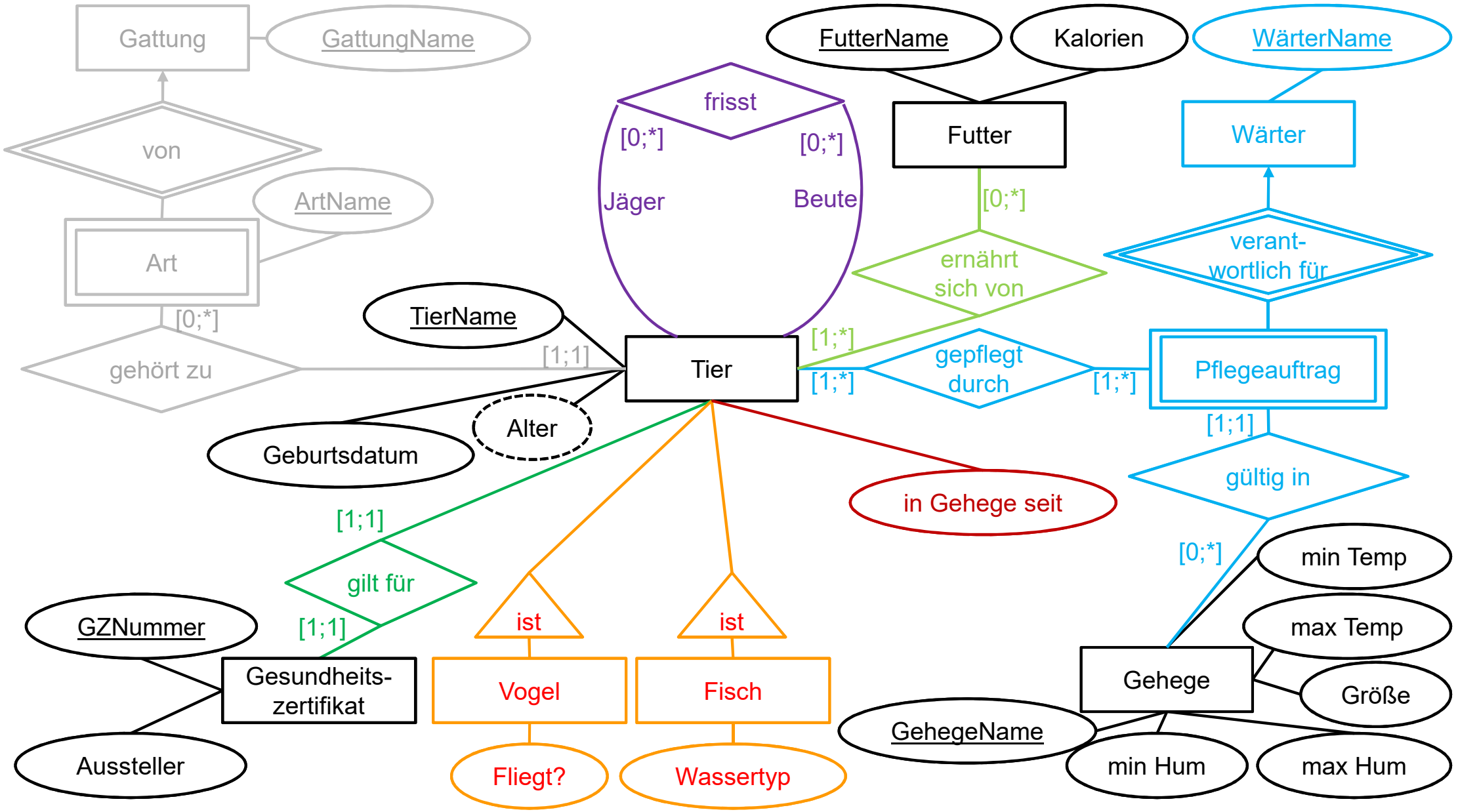
Zusammenfassung des Zoo-Beispiels

Bem: Alter = Jahr(Heute-Geburtsdatum)



Zusammenfassung des Zoo-Beispiels

Bem: Alter = Jahr(Heute-Geburtsdatum)



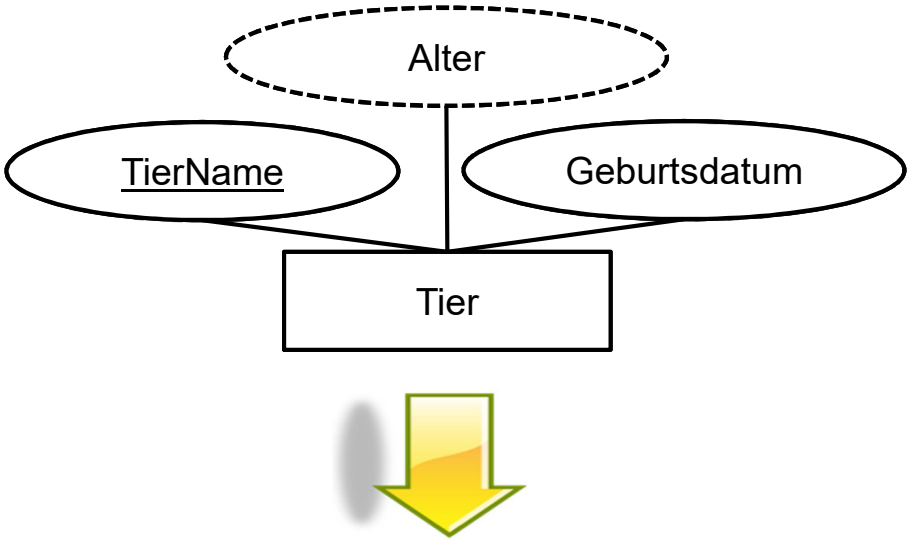
ER-Abbildung

- Erster Teilschritt des logischen Datenbankentwurfs
- Abbildung von ER-Modell auf Relationenmodell
- Im Grunde einfach:
  - Jeder Entity-Typ wird zu einer Relation
  - Jedes Relationship wird zu einer Relation
- Aber
  - Verschmelzen von Relationen
  - Abhängige Entity-Typen
  - Ist-Relationships

Abbildung von (einfachen) Entity-Typen

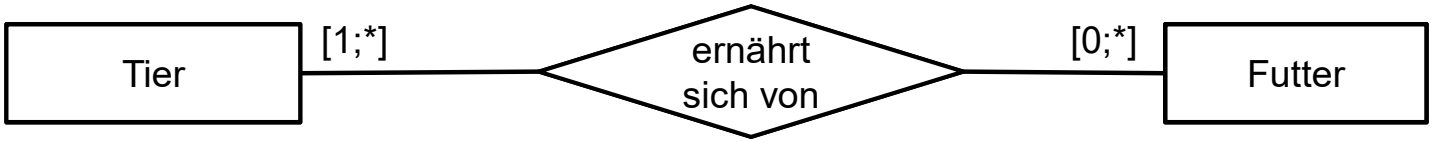
- Jeder Entity-Typ wird zu einer Relation
  - Attribute: alle nicht berechneten Attribute des Entity-Typs
  - Schlüssel: Schlüssel des Entity-Typ
  - Fremdschlüssel: keine

- Beispiel



TIER = {TierName, Geburtsdatum} mit
K_{Tier} = {{TierName}}

Abbildung von n:m- Relationships

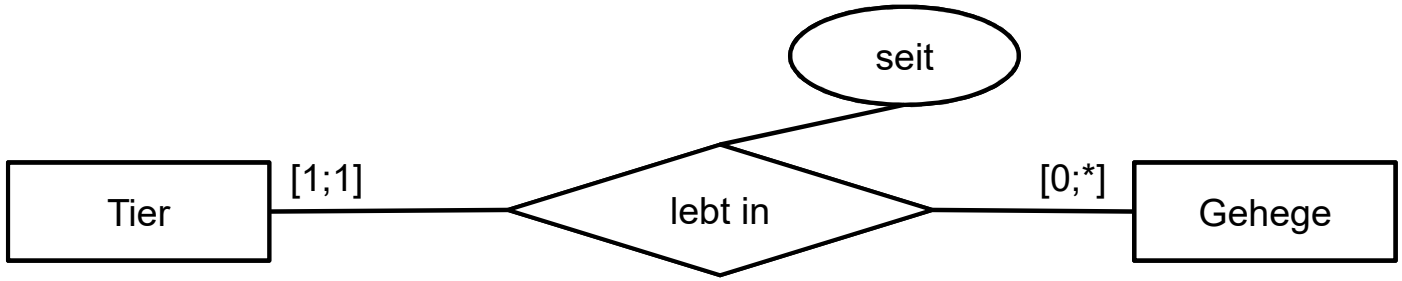


ERNÄHRT_SICH_VON = {TierName, FutterName} mit

$K_{ERNÄHRT\_SICH\_VON}$ = {{TierName, FutterName}}

{TierName} references TIER(TierName)
{FutterName} references FUTTER(FutterName)

Abbildung von 1:n- Relationships



LEBT_IN = {TierName, GehegeName, seit} mit

K_{LEBT_IN} = {{TierName}}

{TierName} references TIER(TierName)
{GehegeName} references GEHEGE(GehegeName)

Abbildung von 1:1- Relationships

$GILT\_FÜR = \{GZNummer, TierName\}$ mit

$K_{GILT\_FÜR} = \{\{GZNummer\}, \{TierName\}\}$

{GZNummer} references GESUNDHEITSZERTIFIKAT(GZNummer)
{TierName} references TIER(TierName)

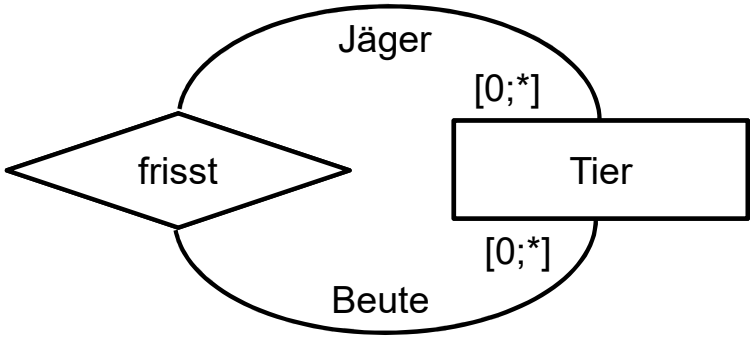
Abbildung von Relationships

- Jedes Relationship wird zu einer Relation
  - Attribute: Schlüssel der beteiligten Entity-Typen + nicht berechnete Attribute des Relationships
  - Fremdschlüssel: Attribute der beteiligten Entity-Typen referenzieren ihre jeweiligen Entity-Typ-Relationen

- Festlegung der Schlüssel der neuen Relationenschemata
  - **m:n-Beziehung**
    Primärschlüssel der beiden Entity-Typen **zusammen** ➔ **ein** Schlüssel
  - **1:n-Beziehung**
    Primärschlüssel der **n-Seite** ➔ Schlüssel im neuen Relationenschema
    - Bei der funktionalen Notation die Seite ohne Pfeilspitze
    - Bei der [min;max] Notation die Seite mit [1;1] oder [1;0]
  - **1:1-Beziehung**
    Primärschlüssel der beiden Entity-Typen ➔ **jeweils ein** Schlüssel im neuen Relationenschema, Primärschlüssel dann aus diesen Schlüsseln gewählt

Abbildung von rekursiven n:m Relationships

- Zusätzlich: Umbenennen der Attribute Beispiel

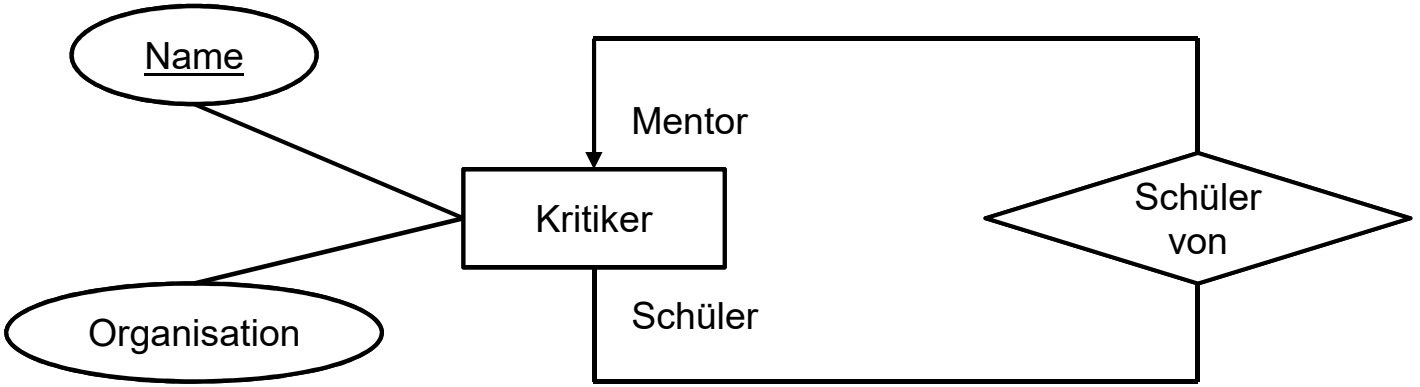


FRISST = {Jäger, Beute} mit

K_{FRISST} = {{Jäger, Beute}}

{Jäger} references TIER(TierName)
{Beute} references TIER(TierName)

Abbildung von rekursiven 1:n Relationships



KRITIKER = {Name, Organisation, Mentorname} mit

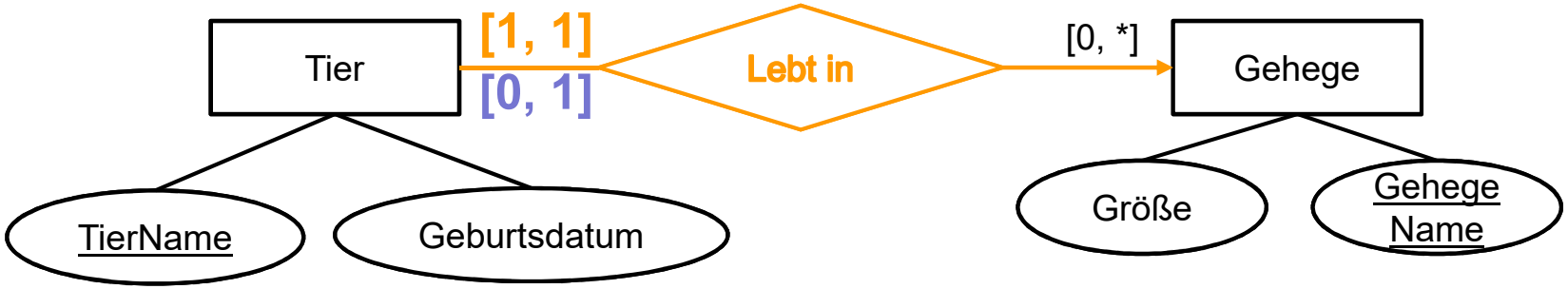
$K_{KRITIKER}$ = {{Name}}

{Mentorname} references KRITKER({Name})

Verschmelzen von Relationen

- Häufig sind die entstehende Relationenschemata nicht optimal
  - **Verschmelzen**: Relationen vereinigen, Schlüssel sinnvoll setzen, Fremdschlüssel (soweit noch sinnvoll) beibehalten
  - **m:n-Beziehung** - nicht verschmelzen
  - **1:n-Beziehung**
    Verschmelze mit der Entity-Typ-Relation der n Seite, Schlüssel beibehalten
    - Falls [0;1]: null-Werte für den Fremdschlüssel erlaubt
    - Falls [1;1]: „not null“ spezifizieren
  - **1:1-Beziehung**
    Verschmelze nicht optionale Seiten mit jeweiligen Entity-Typ-Relationen
    - Falls [0;1] - [0;1]: nicht verschmelzen
    - Falls [1;1] - [0;1]: immer: die [1;1] Seite verschmelzen; Schlüssel: von Entity-Typ-Relation; optional: auch [0;1] Seite unter Verwendung von null-Werten verschmelzen
    - Falls [1;1] - [1;1]: alle drei verschmelzen, Schlüssel: vom Relationship

Verschmelzen von 1:n Relationships



- **[1, 1]:** Verschmelze `TIER` und `LEBT_IN`, not null

  `TIER' = {TierName, Geburtsdatum, GehegeName}`

  $K_{Tier'}$ `= {{TierName}}`

  `{GehegeName} references GEHEGE({GehegeName})`

  `GehegeName not null`

- **[0, 1]:** Verschmelze `TIER` und `LEBT_IN`, null erlaubt

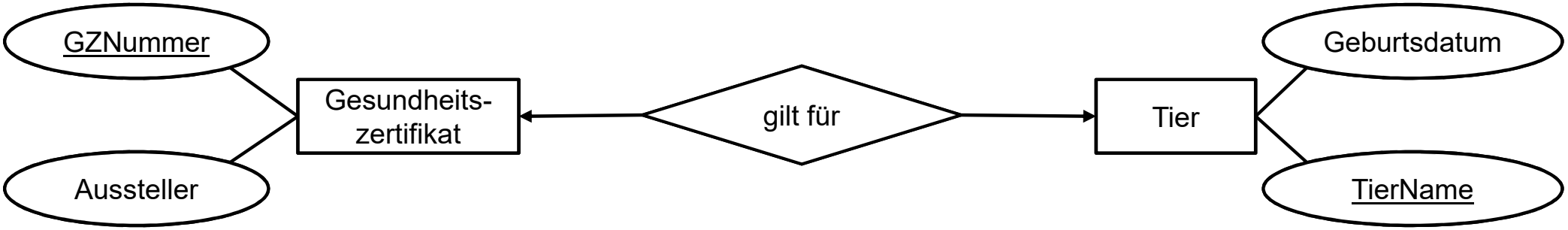
  `TIER' = {TierName, Geburtsdatum, GehegeName}`

  $K_{Tier'}$ `= {{TierName}}`

  `{GehegeName} references GEHEGE({GehegeName})`

  `GehegeName null` (wird i.allg. nicht extra aufgeführt)

Versuch: Verschmelzen von 1:1 Relationships (1)



- verschmolzene Relation:

TIER

TierName	Geburtsdatum	GZNummer	Aussteller
Leo	1.1.2000	42-007	H. Huber
Hilde	11.11.1999	42-009	M. Müller

- Tiere ohne Gesundheitszertifikat erfordern Nullwerte:

TIER

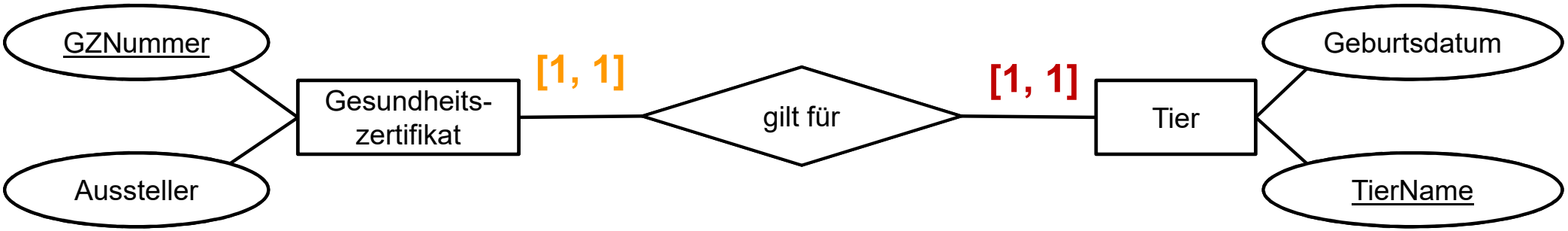
TierName	Geburtsdatum	GZNummer	Aussteller
Leo	1.1.2000	42-007	H. Huber
Hilde	11.11.1999	⊥	⊥

- Gesundheitszertifikate ohne Tiere führen zu weiteren Nullwerten:

TIER

TierName	Geburtsdatum	GZNummer	Aussteller
Leo	1.1.2000	42-007	H.Huber
Hilde	11.11.1999	⊥	⊥
⊥	⊥	42-003	S. Schmidt

Verschmelzen von 1:1 Relationships (2)

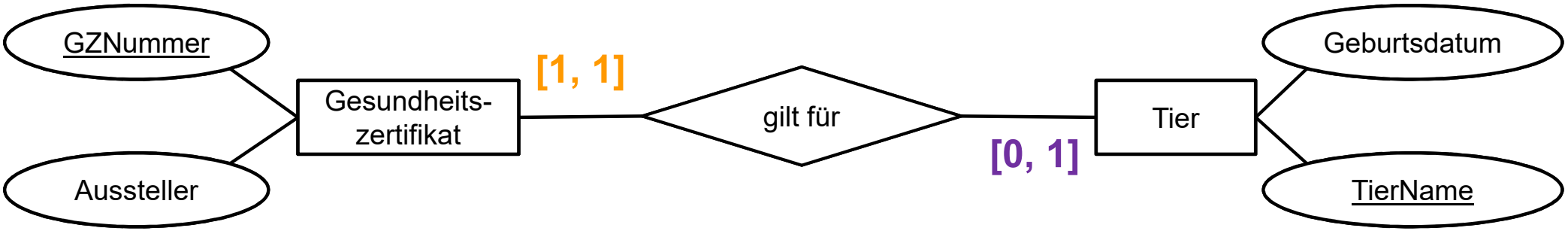


- **[1, 1]** auf **[1, 1]**: alle drei (`GESUNDHEITSZERTIFIKAT, GILT_FÜR, TIER`) verschmelzen möglich

`TIER' = {TierName, Geburtsdatum, GZNummer, Aussteller}`

$K_{TIER'}$ = {{TierName}, {GZNummer}}

Verschmelzen von 1:1 Relationships (3)



- **[1, 1]** auf **[0, 1]**: `GESUNDHEITSZERTIFIKAT` und `GILT_FÜR` verschmelzen möglich

  `GESUNDHEITSZERTIFIKAT' = {GZNummer, Aussteller, TierName}`

  $K_{GESUNDHEITSZERTIFIKAT'} = \{\{GZNummer\}\}$

  `{TierName} referenziert TIER({TierName})`

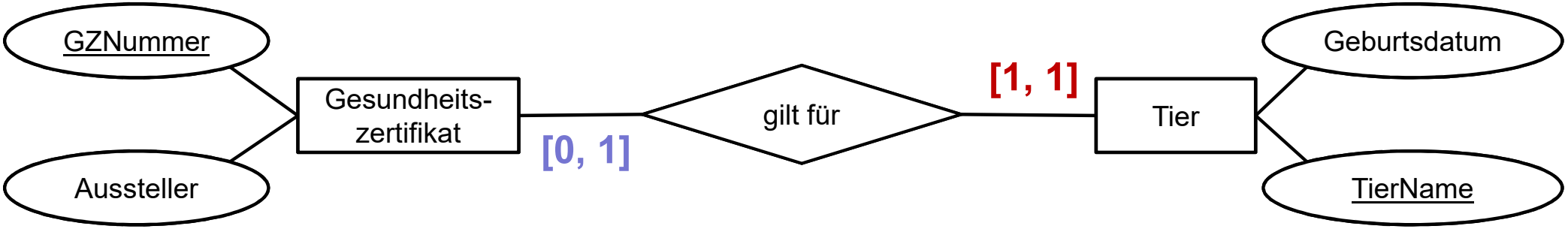
  `(TierName not null)`

  Optional:

  `TIER' = {TierName, Geburtsdatum, GZNummer, Aussteller}`

  $K_{TIER'} = \{\{GZNummer\}\}$; {TierName} unique, null

Verschmelzen von 1:1 Relationships (4)



- **[0, 1]** auf **[1, 1]**: `TIER` und `GILT_FÜR` verschmelzen möglich

  `TIER' = {TierName, Geburtsdatum, GZNummer}`

  $K_{TIER'}$ = {{TierName}}

  `{GZNummer} referenziert GESUNDHEITSZERTIFIKAT({GZNummer})`

  `(GZNummer not null)`

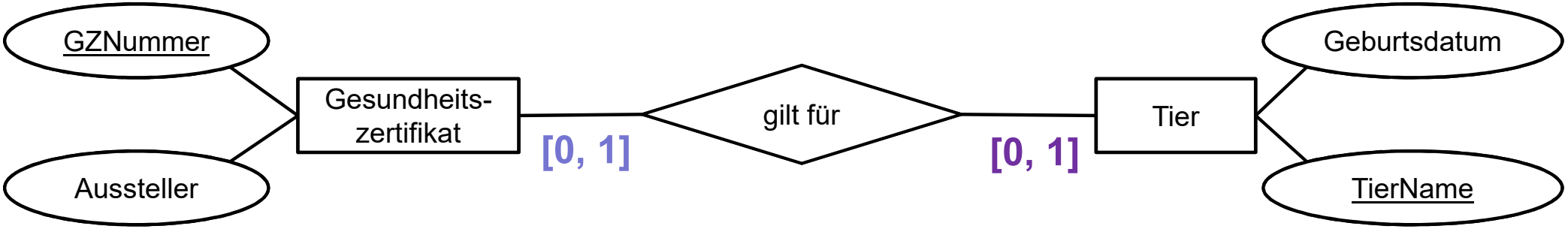
  Optional:

  `TIER'' = {TierName, Geburtsdatum, GZNummer, Aussteller}`

  $K_{TIER''}$ = {{TierName}}

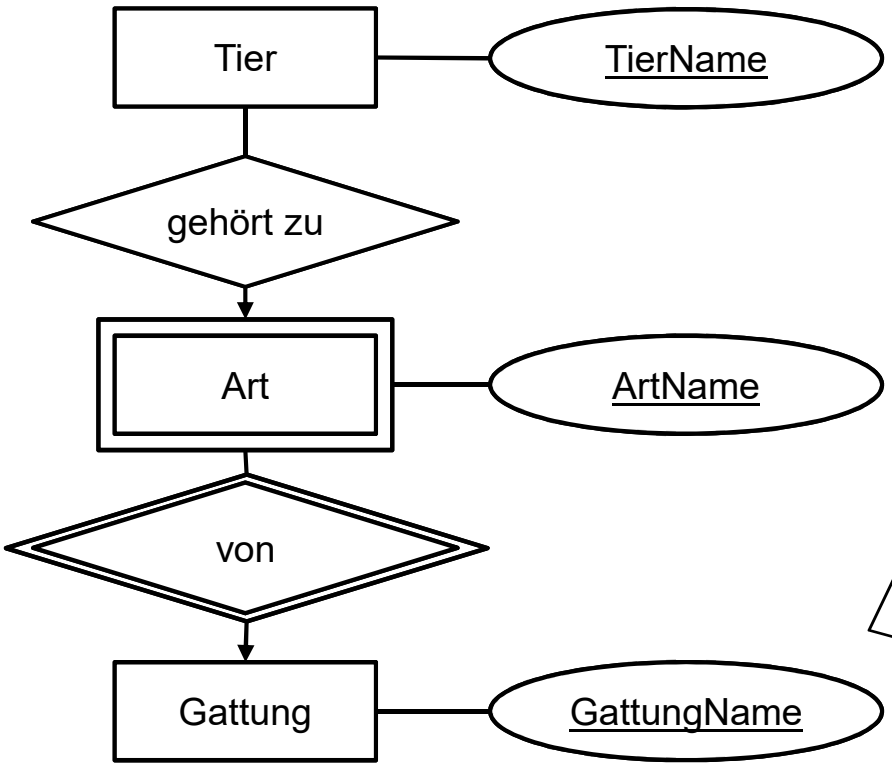
  `{GZNummer} unique, null`

Verschmelzen von 1:1 Relationships (5)



- **[0, 1]** auf **[0, 1]**: nicht verschmelzen

Abbildung von abhängigen Entity-Typen (weak entity sets)



$Gattung = \{GattungName\}$ mit
$K_{Gattung} = \{\{GattungName\}\}$

$ART = \{ArtName, GattungName\}$ mit
$K_{ART} = \{\{ArtName, GattungName\}\}$

{GattungName} references GATTUNG({GattungName})

- Attribut `GattungName` in `ART` ist Fremdschlüssel zur Relation `GATTUNG`
- Für „von“ wird **KEINE** Relation erzeugt!

Allgemein

1) Keine Relationen für Supporting Relationships
2) Relation des abhängigen Entity-Typs enthält eigene Attribute + alle Schlüssel der Entity-Typen auf die die Supporting Relationships zeigen

Abbildung von ist-Relationships

- Drei mögliche Abbildungen
  - ER Style
  - OO Style
  - null-Value Style

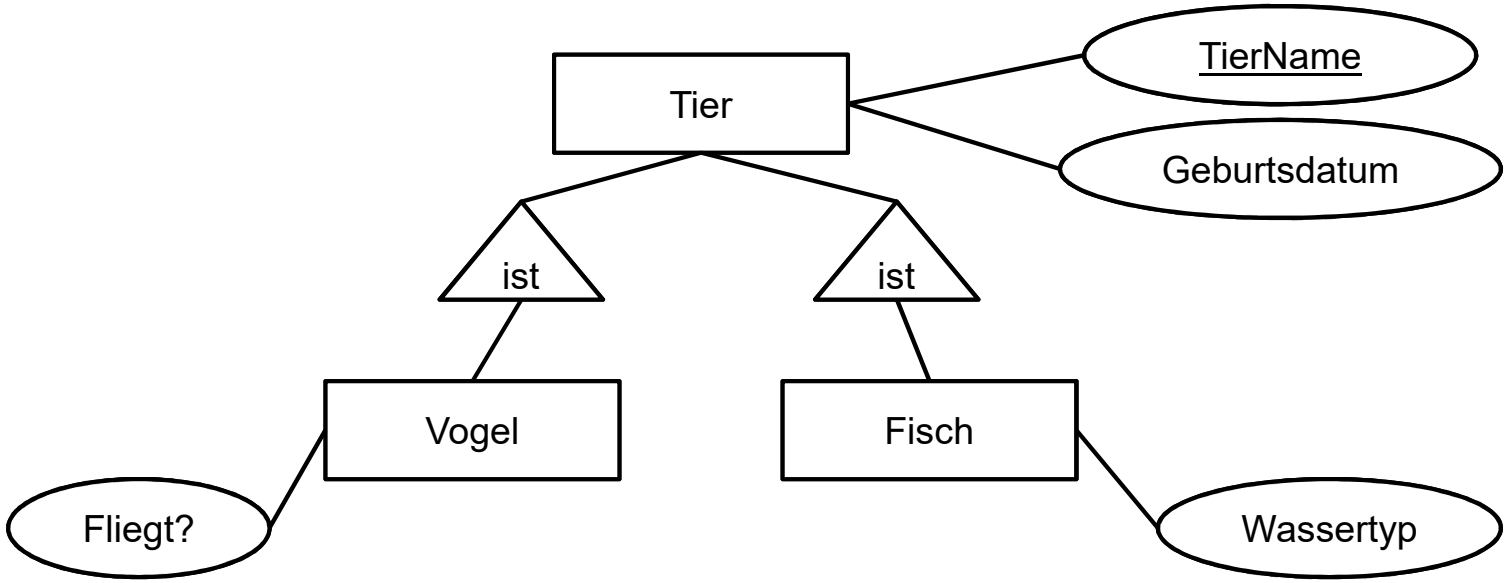
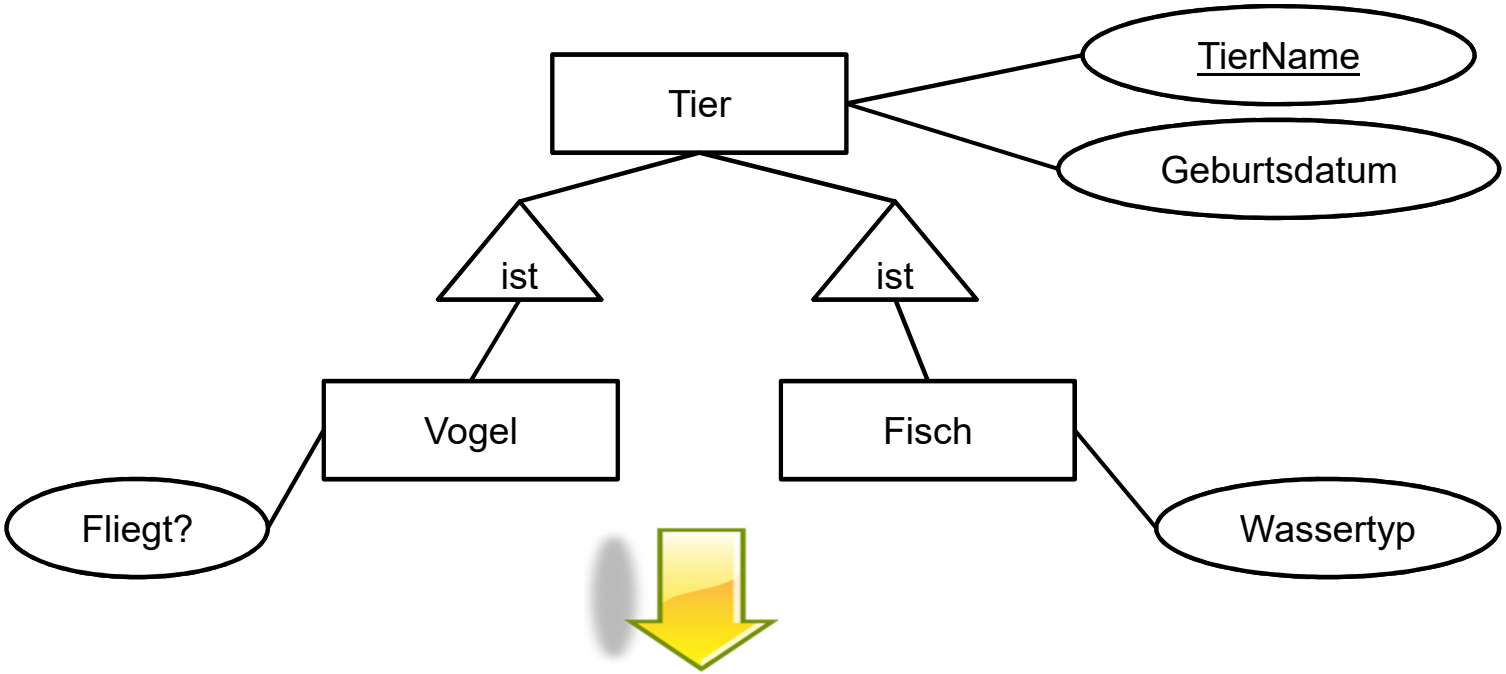


Abbildung von ist-Relationships: ER Style

- ER Style: Eine Relation pro Entity-Typ

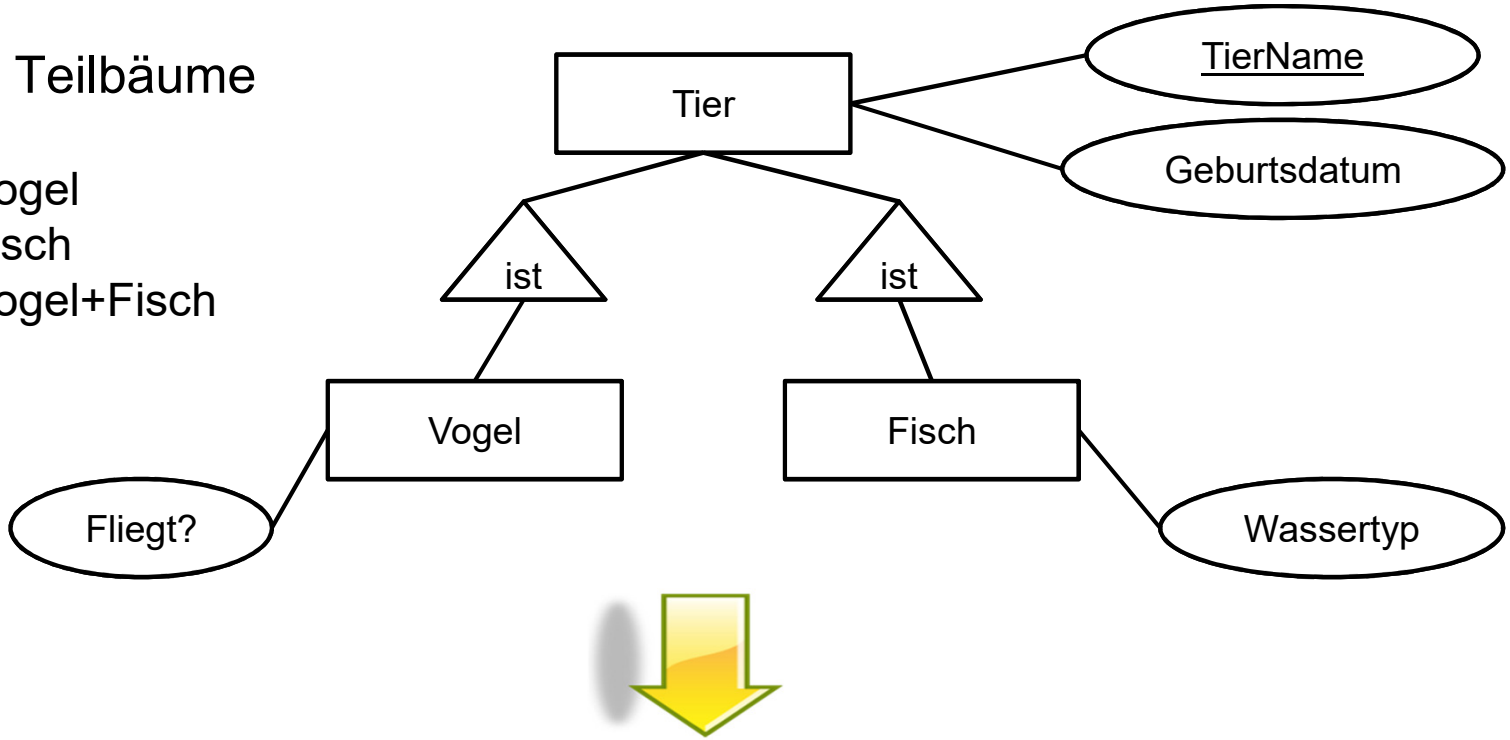



```
TIER = {TierName, Geburtsdatum}
VOGEL = {TierName, fliegt}
FISCH = {TierName, Wassertyp}
```

Beachte:
Jeder Fisch/Jeder Vogel hat zusätzliches Tupel in TIER!

Abbildung von ist-Relationships: OO Style

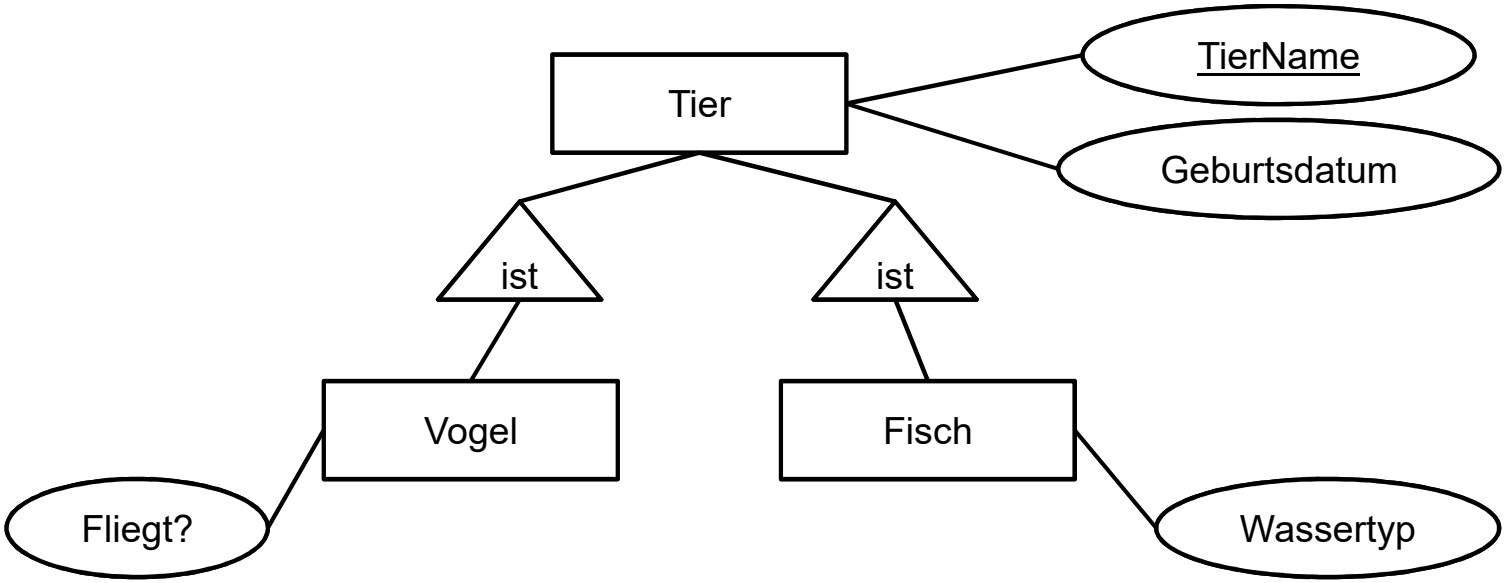
- OO Style: Eine Relation pro Teilbaum
- Beispiel: 4 Teilbäume
  - Tier
  - Tier+Vogel
  - Tier+Fisch
  - Tier+Vogel+Fisch




```
TIER   = {TierName, Geburtsdatum}
TIERV  = {TierName, Geburtsdatum, fliegt}
TIERF  = {TierName, Geburtsdatum, Wassertyp}
TIERVF = {TierName, Geburtsdatum, fliegt , Wassertyp}
```

Abbildung von ist-Relationships: null-Value Style

- Null-Value Style: Nur eine Relation, null für n/a Attribute




```
TIER   = {TierName, Geburtsdatum, fliegt , Wassertyp}
```

- Jeder Fisch hat `null` in `fliegt`
- Jeder Vogel hat `null` in `Wassertyp`
- Weder Fisch noch Vogel hat `null` in `fliegt` und `Wassertyp`

Abbildung von ist-Relationships - Diskussion

- Jede Abbildungsart hat Vor- und Nachteile
  - Weniger Relationen generell besser (weniger Joins nötig)
    - Null-Value Style gut
    - OO Style schlecht da sehr viele Relationen
    - ER ok
  - Weniger Platzbedarf generell besser
    - OO nur ein Tupel mit genau den passenden Attributen, also sehr gut
    - null-Value nur ein Tupel aber das ist lang
    - ER viele Tupel, aber nur Schlüssel wird wiederholt
  - Schnelle Query-Ausführung
    - Kommt sehr auf das Query an!

➔ Kein klarer Sieger, kommt auf die Anwendung an!

Übersicht über die Transformationen

ER-Konzept	wird abgebildet auf relationales Konzept
Entity-Typ E_i Attribute von E_i Primärschlüssel P_i	Relationenschema R_i Attribute von R_i Primärschlüssel P_i
Beziehungstyp dessen Attribute 1 : n 1 : 1 m : n	Relationenschema Attribute: P_1, P_2 weitere Attribute P_2 wird Primärschlüssel der Beziehung P_1 und P_2 werden Schlüssel der Beziehung $P_1 \cup P_2$ wird Primärschlüssel der Beziehung
IST-Beziehung	R_1 erhält zusätzlichen Schlüssel P_2

- mit
  - E_1, E_2 : an Beziehung beteiligte Entity-Typen,
  - P_1, P_2 : deren Primärschlüssel,
  - 1 : n-Beziehung: E_2 ist n-Seite,
  - IST-Beziehung: E_1 ist speziellerer Entity-Typ

Praktisches Vorgehen beim Design einer Datenbank

- Wir kennen nun 2 Möglichkeiten, eine Datenbank zu designen:
  1) Aus einer Mengen von Attributen und einer Menge von FDs direkt mittels Normalisierung (BCNF, 3NF / Dekomposition, Synthese)
  2) Mittels eines ER-Modells und (semi-)automatischer Transformation in ein Relationales Modell (RM)

  ➔ Frage: Welches Vorgehen ist nun „das richtige“?

- Typisches (empfohlenes) Vorgehen
  - Aufstellen eines ER-Modells
  - Transformation in ein Relationales Modell
  - Wenn das ER-Modell gut ist, ist das erzeugte RM meist automatisch in BCNF!
  - (Automatisiertes) Prüfen, ob das RM in BCNF (oder zumindest 3NF) ist
  - Wenn nicht: Hinweis auf einen Design-Fehler im ER, daher: zurück zum ER Modell und dieses Verbessern (sehr selten: Beheben der Redundanzen im RM)

  ➔ Normalformentheorie und Normalisierung ist ein wichtiges Hilfsmittel um ein gutes ER Modell zu designen!

Zusammenfassung

- ER-Modell
  - Entity-Typen
  - Relationships (Kardinalitäten: n:m, 1:n, 1:1)
  - Mehrstellige Relationships und Umsetzung in binäre
  - Rekursive Relationships
  - Abhängige Relationships
  - Ist-Relationship
- Gute ER-Modelle
- Transformation eines ER-Modells in ein Relationales Modell
  - Entity-Typen, Relationships
  - Verschmelzen
  - Abbildung abhängiger Relationships
  - Abbldung von ist-Relationships