

# Datenbanken und SQL

## Kapitel 3

### Datenbankdesign – Teil 2: Entity-Relationship-Modell

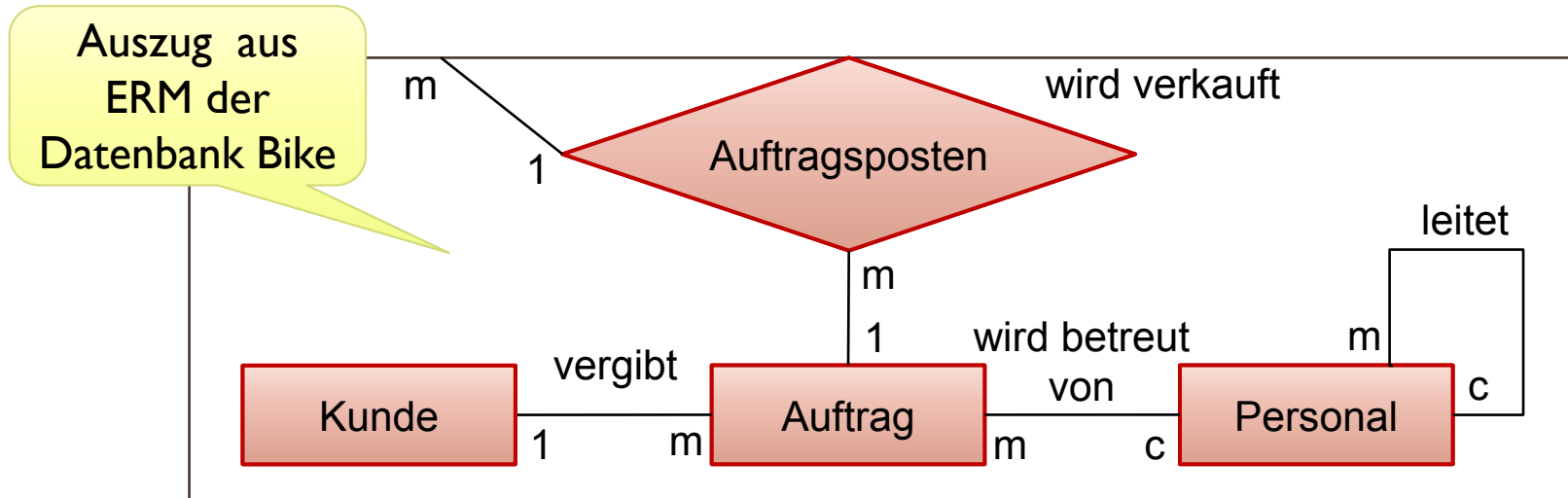
# Datenbankdesign

---

- ▶ **Entity-Relationship-Modell ERM**
  - ▶ Entitäten und ihre Eigenschaften
  - ▶ Beziehungen zwischen den Entitäten
  - ▶ Überführung der Entitäten in Relationen
  - ▶ Überführung der Beziehungen in Fremdschlüssel
  - ▶ Fremdschlüsseleigenschaften
  - ▶ Schwache Entitäten
  - ▶ Subtypen

# Entity-Relationship-Modell (ERM)

- ▶ Bisher:
  - ▶ Betrachten einzelner Relationen isoliert für sich
- ▶ Jetzt:
  - ▶ Betrachten des Zusammenspiels der Relationen



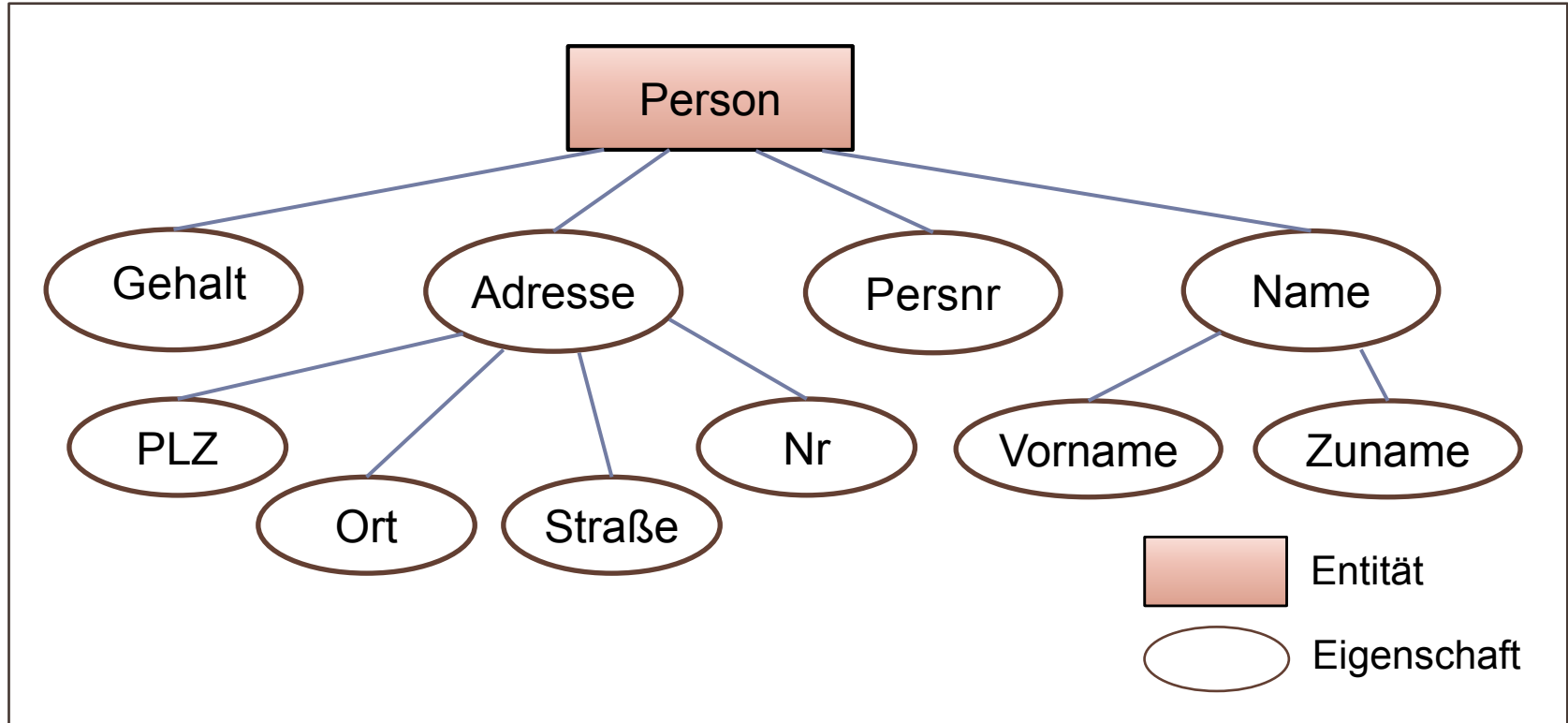
# Begriffe im Entity-Relationship-Modell

Entität	Ein eindeutig unterscheidbares Objekt, ein unterscheidbares Element
Eigenschaft	Ein Teil einer Entität, der die Entität beschreibt
Beziehung	Eine Entität, die zwei oder mehr Entitäten miteinander verknüpft
Subtyp	Eine Entität, die ein Teil einer anderen, umfassenderen Entität ist
Supertyp	Eine Entität, die Subtypen enthält
Schwache Entität	Entität, die von einer anderen Entität vollständig abhängig ist

# Beispiele zu den Begriffen

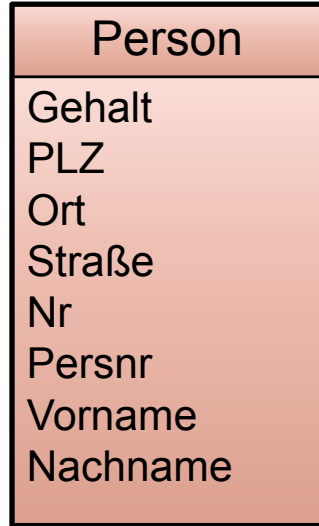
Begriff	Beispiele
Entität	Person, Werkzeug, Produkt, Rechnung
Eigenschaft	Name, Vorname, PLZ, Ort einer Person; Größe, Gewicht eines Werkzeugs; Preis eines Produkts, Rechnungsdatum
Beziehung	Die Entitäten Verkäufer und Produkt stehen miteinander in einer Beziehung: Der Verkäufer verkauft Produkte.
Subtyp	Die Entität Verkäufer ist ein Subtyp zur Entität Mitarbeiter
Supertyp	Die Entität Mitarbeiter ist ein Supertyp der Entität Verkäufer
Schwache Entität	Die Entität Arbeitszeit ist schwach gegenüber der Entität Mitarbeiter

# Beispiel: Entität Person



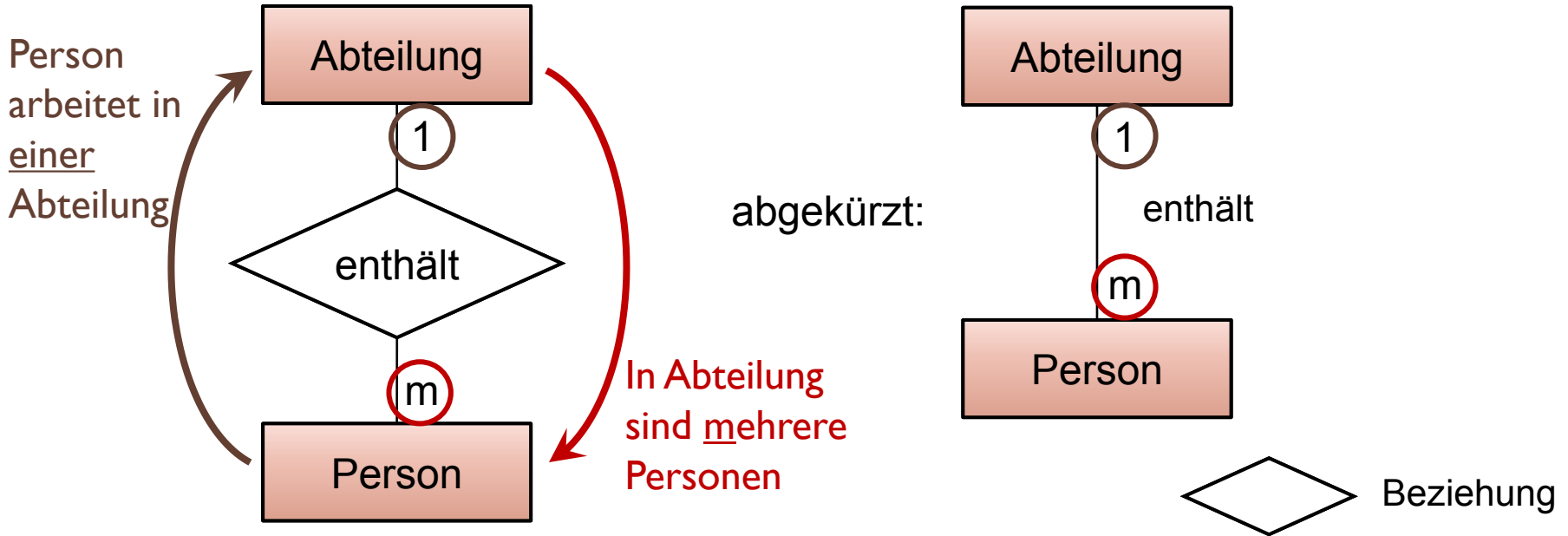
# Entität Person in „UML“-Notation

---



- ▶ In die Entität werden die Eigenschaften mit aufgenommen
- ▶ Manchmal werden auch Primärschlüssel, alternative Schlüssel und Fremdschlüssel gleich mit gekennzeichnet

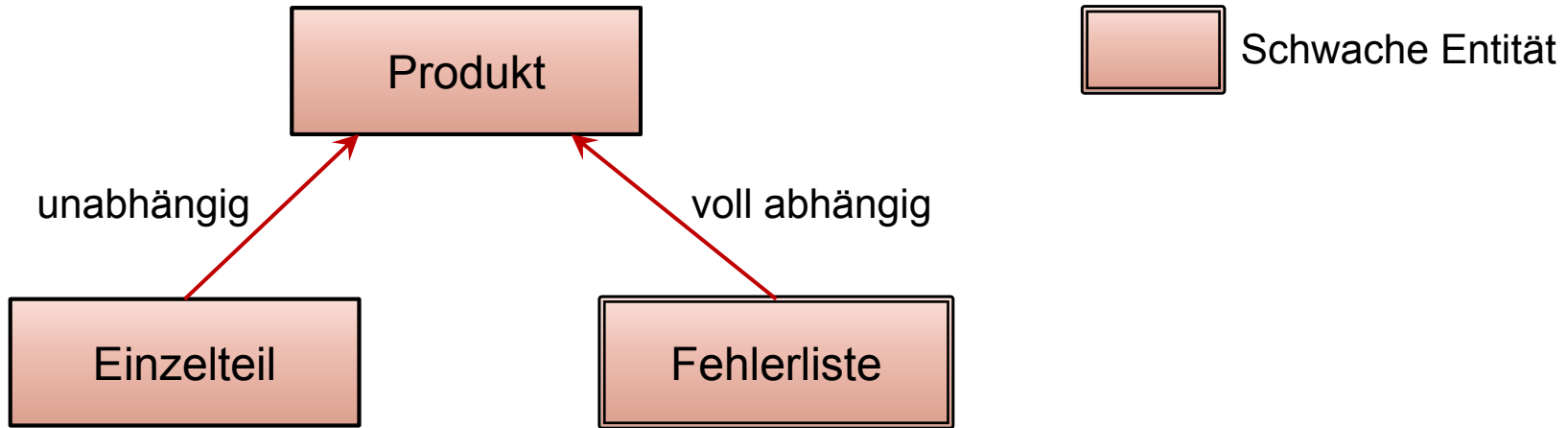
# Beispiel einer Beziehung



- ▶ In einer Abteilung arbeiten mehrere (m) Personen
- ▶ Eine Person arbeitet in genau einer (1) Abteilung



# Schwache Entität



## ▶ Einzelteil:

- ▶ Ist auf Lager, auch wenn Produkt nicht mehr ex.
- ▶ Unabhängig vom Produkt

## ▶ Produkt-Fehlerliste:

- ▶ Wertlos, wenn Produkt nicht mehr ex.
- ▶ Komplette abhängig vom Produkt

# Umsetzung der Entität Person in SQL

Person
Persnr
Vorname
Nachname
Gehalt
PLZ
Ort
Straße
Nr

```
CREATE TABLE Person
(   Persnr      INTEGER,
    PRIMARY KEY (Persnr),
    Vorname     CHARACTER(20),
    Nachname    CHARACTER(20) NOT NULL,
    Gehalt      NUMERIC (10, 2),
    PLZ         CHARACTER(5),
    Ort         CHARACTER(25),
    Strasse     CHARACTER(25),
    Nr          CHARACTER(4)
);
```

Ganzzahl

Primärschlüssel

Zeichenkette  
der Länge 20

Nachname muss  
angegeben werden

Gleitpunktzahl:  
10 Zeichen, davon  
2 Nachkommastellen

# Beziehungen (grobe Einteilung)

---

## ▶ 1 zu 1 Beziehung

- ▶ Ein Auto hat einen Motor
- ▶ Ein Motor ist in einem Auto

## ▶ m zu 1 Beziehung

- ▶ In einer Abteilung arbeiten mehrere Personen
- ▶ Eine Person ist einer Abteilung zugeordnet

## ▶ m zu n Beziehungen

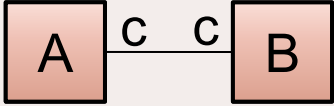
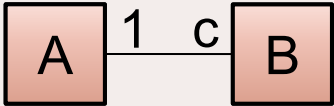
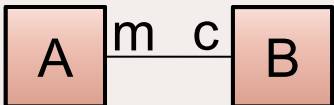
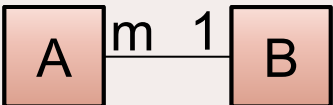
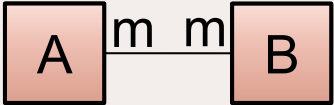
- ▶ Ein Verkäufer verkauft mehrere Produkte
- ▶ Ein Produkt wird von mehreren Verkäufern angeboten

# Beziehungen (Besonderheiten)

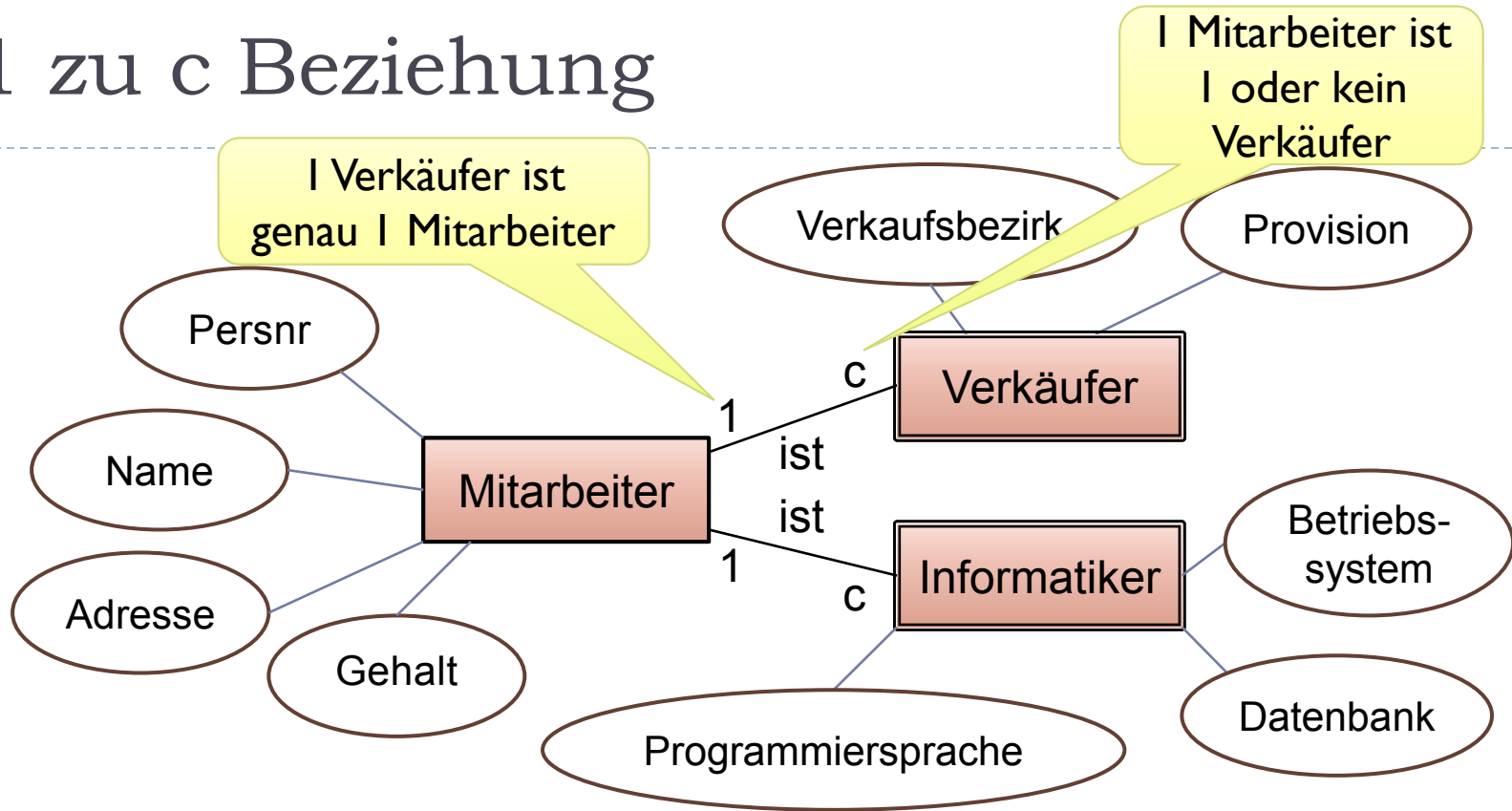
---

- ▶ **m:**
  - ▶ Mehrere kann sein: 0, 1, 2, ... 13, ... 103.517 usw.
  - ▶ m entspricht dem häufig verwendeten Sternsymbol ( \* )
- ▶ **n:**
  - ▶ Nur ein anderer Buchstabe für m
- ▶ **l:**
  - ▶ Eins kann sein: 0 oder 1 (Beispiel: Ein Motor ist nicht im KFZ!)
  - ▶ Unterscheidung ist wichtig in relationalen Datenbanken
  - ▶ Wir verwenden c für 0 oder 1, also  $c \in \{0, 1\}$
  - ▶ Wir verwenden 1, wenn der Wert 0 nicht vorkommen darf

# Mögliche Beziehungen

Beziehungen	c (0..1)	1	m
c (0..1)		Symmetrie!	Symmetrie!
1		Nicht möglich	Symmetrie!
m			

# 1 zu c Beziehung



- ▶ Detailinfos: ausgelagert in Subtypen Verkäufer, Informatiker
- ▶ Reduziert Redundanzen

# c zu c und 1 zu 1 Beziehungen

---

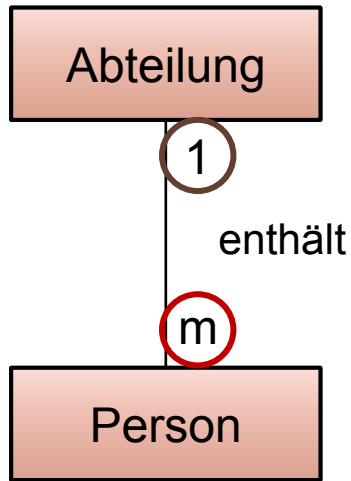
## ► c zu c:

- Sehr selten, etwa: Auto --- Motor
- Spezialfall von m zu c, zusätzlich: Fremdschlüssel ist eindeutig

## ► 1 zu 1:

- Erfordert, dass in beiden Entitäten immer je ein Eintrag existiert (ein Verweis muss ja gegenseitig existieren!)
- In relationalen Datenbanken erfolgt erst ein Eintrag der einen, dann ein Eintrag der anderen, also: 1 zu c Bedingung
- In relationalen Datenbanken treten diese Beziehungen also nicht auf (außer mittels komplexer Transaktionsmechanismen)

# m zu 1 Beziehung

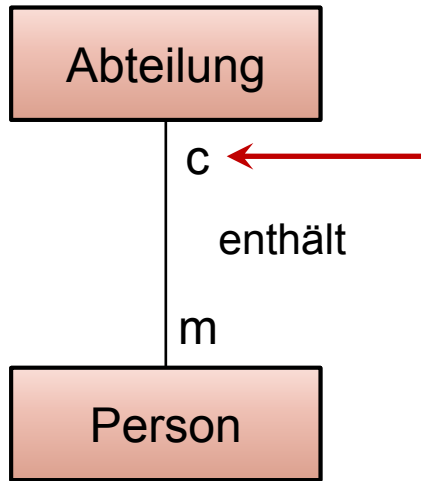


- ▶ In einer Abteilung arbeiten mehrere Personen
- ▶ Eine Person ist exakt einer Abteilung zugeordnet
- ▶ Wichtig:
  - ▶ Eine Person ist immer einer Abteilung zugewiesen
  - ▶ Aber: In einer Abteilung können vorübergehend auch keine Personen arbeiten
    - ▶ m lässt den Wert 0 zu!



# m zu c Beziehungen

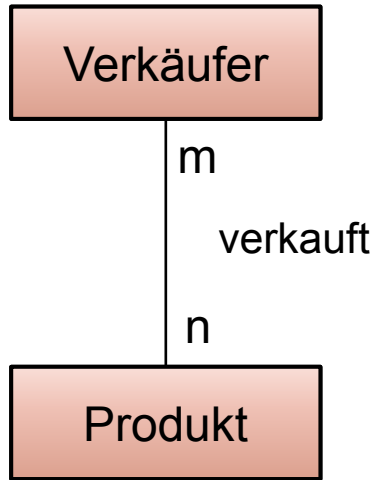
---



- ▶ In einer Abteilung arbeiten mehrere Personen
- ▶ Eine Person ist einer oder keiner Abteilung zugeordnet
  - ▶ Szenario:
  - ▶ Abteilung wird aufgelöst. Mitarbeiter gehören dann keiner Abteilung an und werden erst nach und nach anderen Abteilungen zugeordnet
  - ▶ Dies erfordert: m zu c !

# m zu n Beziehungen

---



- ▶ Ein Verkäufer verkauft mehrere Produkte
- ▶ Ein Produkt wird von mehreren Verkäufern verkauft
- ▶ Wichtig:
  - ▶ m zu n schließt ein:
  - ▶ Ein neuer Verkäufer hat noch nichts verkauft
  - ▶ Ein neues Produkt wurde noch nicht verkauft

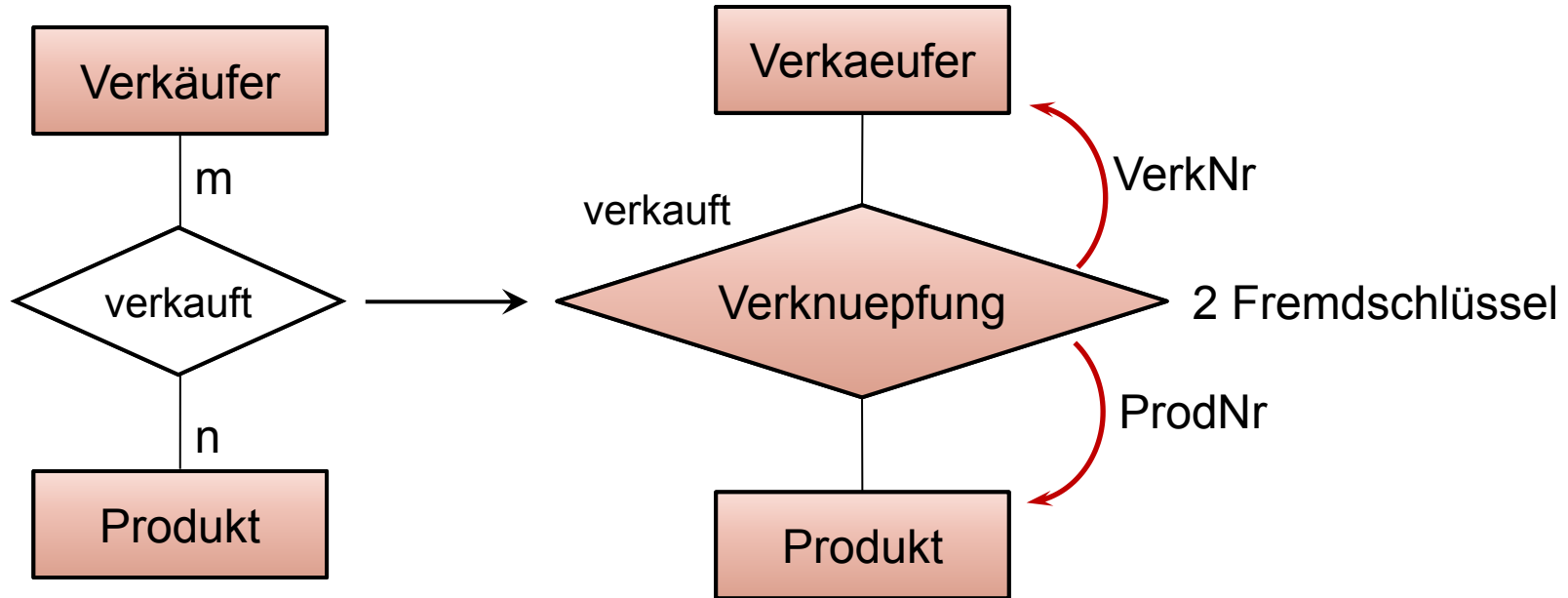
# Beispiele (1)

Beziehung	Bemerkung
KFZ-Halter $\longleftrightarrow$ KFZ <i>c zu m</i>	<ul style="list-style-type: none"><li>  Halter kann mehrere KFZ anmelden</li><li>  KFZ ist auf maximal einen Halter zugelassen</li></ul>
Student $\longleftrightarrow$ Vorlesung <i>m zu n</i>	<ul style="list-style-type: none"><li>  Student besucht mehrere Vorlesungen</li><li>  Vorlesung belegen mehrere Studenten</li></ul>
Kunde $\longleftrightarrow$ Bestellung <i>1 zu m</i>	<ul style="list-style-type: none"><li>  Kunde gibt mehrere Bestellungen auf</li><li>  Bestellung gehört zu genau einem Kunden</li></ul>
Bewohner $\longleftrightarrow$ Haus <i>m zu 1</i>	<ul style="list-style-type: none"><li>  Bewohner wohnt in einem Haus</li><li>In   Haus wohnen mehrere Bewohner</li></ul>

# Beispiele (2)

Beziehung	Bemerkung
Park $\longleftrightarrow$ Baum <i>1 zu m</i>	In 1 Park wachsen mehrere Bäume 1 bestimmter Baum steht in einem Park
Park $\longleftrightarrow$ Baumart <i>m zu n</i>	In 1 Park wachsen mehrere Baumarten 1 Baumart wächst in mehreren Parks
KFZ $\longleftrightarrow$ Motor <i>c zu c</i>	1 KFZ besitzt maximal einen Verbrennungsmotor 1 Motor wird in maximal einem KFZ eingebaut
KFZ-Typ $\longleftrightarrow$ Motortyp <i>m zu n</i>	1 KFZ-Typ besitzt mehrere Motorvarianten 1 Motortyp wird in mehreren KFZ-Typen verbaut
Leiter $\longleftrightarrow$ Abteilung <i>c zu 1</i>	1 Abteilungsleiter leitet genau eine Abteilung 1 Abteilung besitzt maximal einen Abteilungsleiter

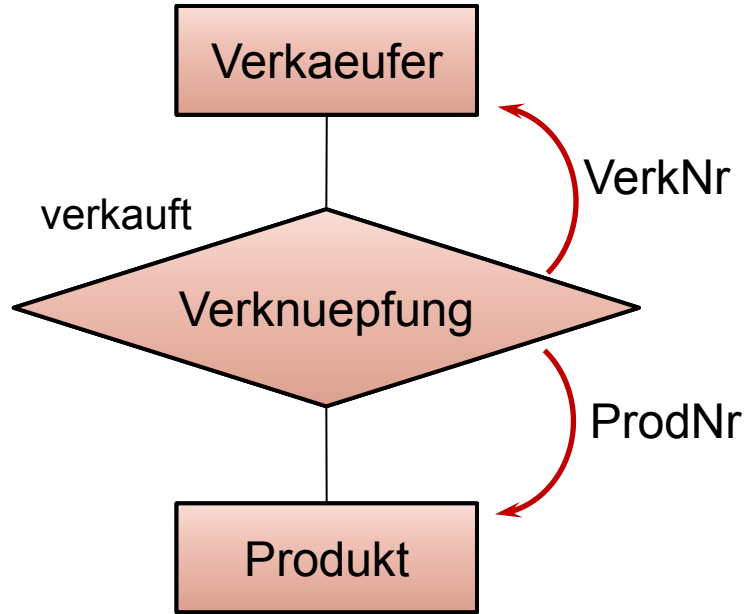
# m zu n Beziehungen: Realisierung (1)



- ▶ Eigene Relation erforderlich
- ▶ Relation enthält 2 Fremdschlüssel
- ▶ Die 2 Fremdschlüssel sind Schlüsselkandidat

◊ Beziehungsrelation

# m zu n Beziehungen: Realisierung (2)



```
CREATE TABLE Verknuepfung  
( VerkNr CHARACTER(4)  
  REFERENCES Verkaeufer,  
  ProdNr CHARACTER(4)  
  REFERENCES Produkt,  
  Umsatz INTEGER,  
  PRIMARY KEY (VerkNr, ProdNr)  
);
```

Fremdschlüssel

Fremdschlüssel

Primärschlüssel

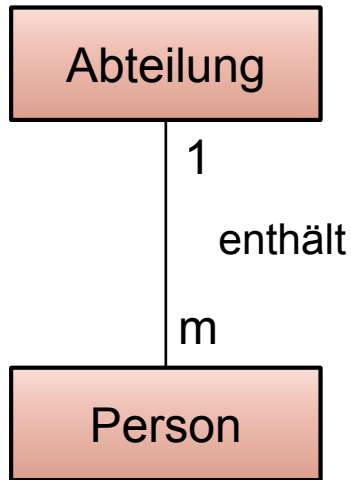
# Definition (Beziehungsrelation)

---

- ▶ Seien  $k$  Relationen mit  $k > 1$  gegeben. Eine Relation  $R$  heißt Beziehungsrelation, wenn sie diese  $k$  Relationen wie folgt miteinander verbindet:
  - $R$  enthält  $k$  Fremdschlüssel mit  $k > 1$ , die je auf genau eine der  $k$  gegebenen Relationen verweisen.
  - Die  $k$  Fremdschlüssel bilden zusammen einen Schlüsselkandidaten.
- Wichtig:
  - Jede Relation mit obigen Eigenschaften ist also eine Beziehungsrelation!

# m zu 1 Beziehung: Realisierung

- ▶ In der m Beziehung wird ein Fremdschlüssel hinzugefügt



```
CREATE TABLE Person
(  PersNr      INTEGER,
    Name        CHARACTER (25),
    ...
    Abteilungsnr  INTEGER NOT NULL
                  REFERENCES Abteilung,
    PRIMARY KEY (Persnr)
);
```

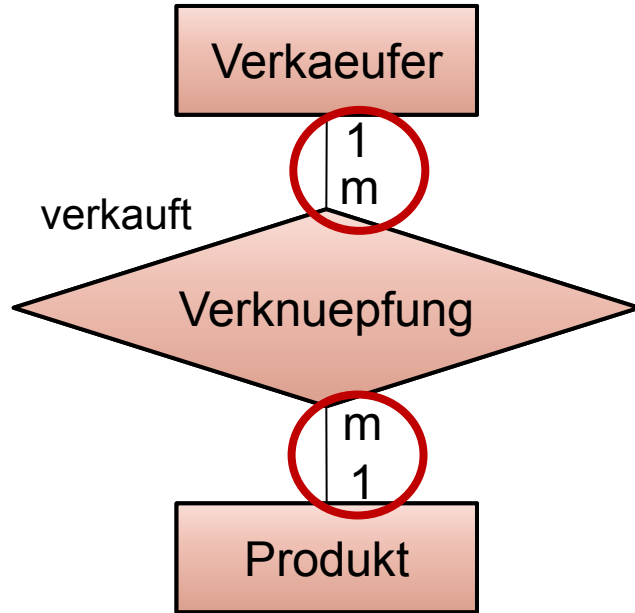
Wegen m zu 1

Fremdschlüssel

Primärschlüssel



# Einschub: m zu n = Zwei m zu 1



```
CREATE TABLE Verknuepfung
( VerkNr  CHARACTER(4)
  REFERENCES Verkaeuer,
  ProdNr  CHARACTER(4)
  REFERENCES Produkt,
  Umsatz  INTEGER,
  PRIMARY KEY (VerkNr, ProdNr)
);
```

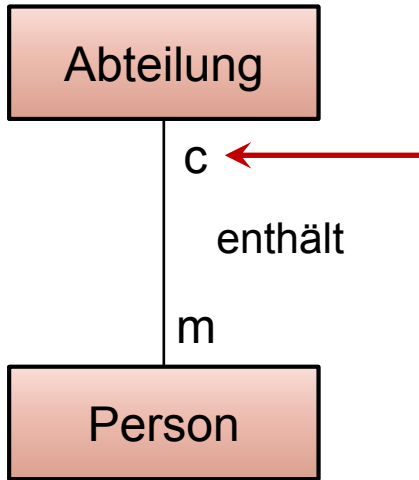
Fremdschlüssel

Fremdschlüssel

Kein NOT NULL, da Teil  
des Primärschlüssels

# m zu c Beziehung: Realisierung

- ▶ Wie m zu 1, allerdings sind Nullwerte erlaubt



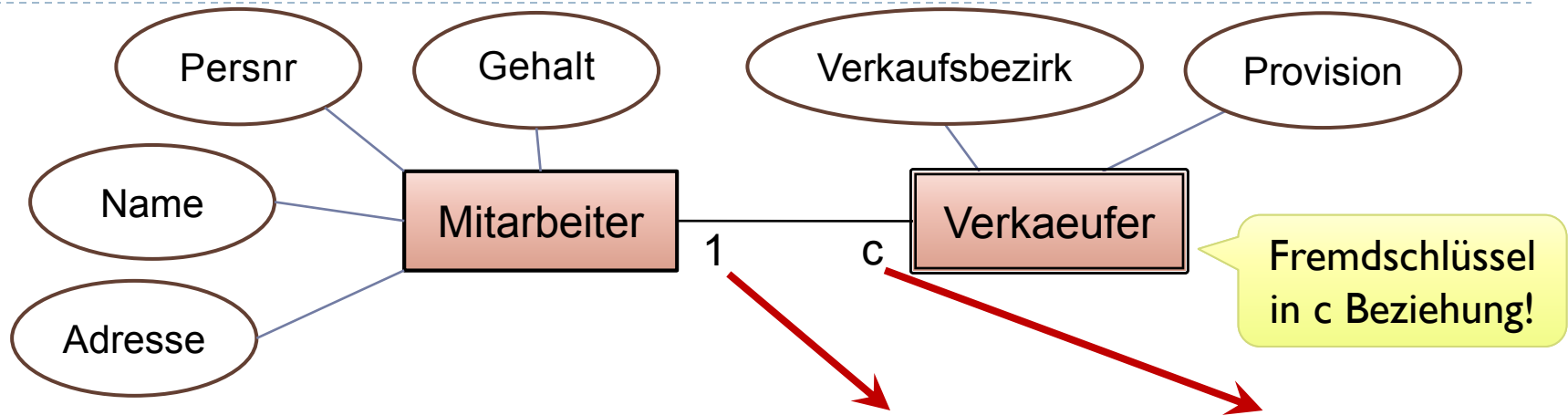
```
CREATE TABLE Person
(  PersNr      INTEGER,
   Name        CHARACTER (25),
   ...
   Abteilungsnr INTEGER
   REFERENCES Abteilung,
   PRIMARY KEY (Persnr)
);
```

Kein NOT NULL

Fremdschlüssel

Primärschlüssel

# 1 zu c Beziehung: Realisierung



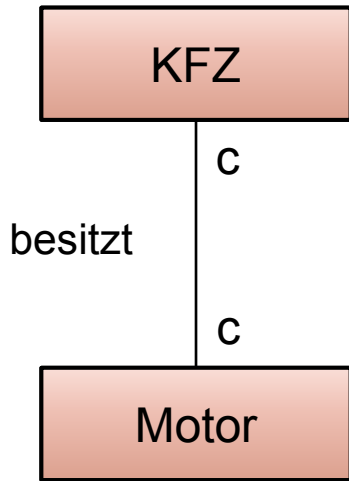
- ▶ Fremdschlüssel in Verkaeuer erfordert NOT NULL und Eindeutigkeit
- ▶ Der Primärschlüssel ist meist auch Fremdschlüssel (bei Subtypen!)

```
CREATE TABLE Verkaeuer
( PersNr INTEGER REFERENCES Mitarbeiter,
  PRIMARY KEY (PersNr),
  ... );
```

Primärschlüssel

Fremdschlüssel

# c zu c Beziehung: Realisierung



- ▶ Wie m zu c Beziehung, allerdings ist Fremdschlüssel zusätzlich eindeutig
- ▶ Empfehlung: Die „umfassendere“ Entität enthält den Fremdschlüssel

```
CREATE TABLE KFZ
( KFZNr    INTEGER,
  PRIMARY KEY (KFZNr),
  MotorNr  INTEGER REFERENCES Motor,
  UNIQUE (MotorNr),
  ...
);
```

Eindeutig!

Fremdschlüssel

# Zusammenfassung zur Realisierung

Beziehung	Überführung in Relationen und Fremdschlüssel
m zu n	Erfordert Beziehungsrelation mit zwei m zu 1 oder m zu c Beziehungen
m zu c	Hinzufügen eines Fremdschlüssels zur m Relation
m zu 1	Wie m zu c! Zusätzlich: Fremdschlüssel ist NOT NULL
c zu c	Wie m zu c! Zusätzlich: Fremdschlüssel ist UNIQUE
c zu 1	Wie c zu c! Zusätzlich: Fremdschlüssel ist NOT NULL Meist ist Fremdschlüssel der Primärschlüssel

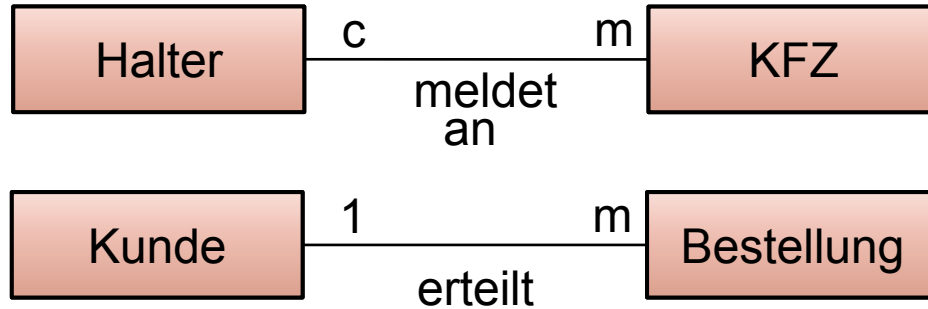
# Fremdschlüsseleigenschaften

---

- (1) Darf ein Fremdschlüsselwert leer bleiben, also Null-Werte enthalten?
- (2) Darf ein Tupel gelöscht werden, auf den sich ein Fremdschlüssel bezieht?
  - ▶ Wie sollte die Datenbank reagieren?
- (3) Darf ein Tupel geändert werden, auf den sich ein Fremdschlüssel bezieht?
  - ▶ Wie sollte die Datenbank reagieren?

# Frage 1: Nullwerte erlaubt?

---



- ▶ **ERM gibt die Antwort vor!**
  - ▶ Bei c Beziehung: Nullwerte sind immer zuzulassen
  - ▶ Bei 1 Beziehung: Nullwerte sind immer verboten
- ▶ **Zu beachten:**
  - ▶ Für Primärschlüssel gilt immer **NOT NULL**

# Frage 2: Löschen eines Tupel

---

## ▶ Fremdschlüsselbedingungen in SQL

- ▶ ON DELETE NO ACTION
- ▶ ON DELETE SET NULL
- ▶ ON DELETE CASCADE

## ▶ Funktionsweise

- ▶ Wird ein Tupel gelöscht, auf den ein Fremdschlüssel mit obiger Bedingung verweist, dann
  - ▶ wird das Löschen dieses Tupel verhindert (Nichtstun, No Action)
  - ▶ wird der darauf verweisende Fremdschlüssel auf Null gesetzt (Set Null)
  - ▶ wird auch das den Fremdschlüssel enthaltene Tupel gelöscht (Cascade)



# Frage 3: Ändern des Primärschlüssels

---

- ▶ Analog wird das Ändern eines Primärschlüssel behandelt:
  - ▶ ON UPDATE NO ACTION
  - ▶ ON UPDATE SET NULL
  - ▶ ON UPDATE CASCADE
- ▶ Funktionsweise
  - ▶ Wird ein Primärschlüsselwert geändert, auf den ein Fremdschlüssel mit obiger Bedingung verweist, dann
    - ▶ wird das Ändern dieses Tupel verhindert (Nichtstun, No Action)
    - ▶ wird der darauf verweisende Fremdschlüssel auf Null gesetzt (Set Null)
    - ▶ wird der Fremdschlüsselwert mit geändert (Cascade)

# Hinweise zu Frage 2 und 3

---

- ▶ m zu l und c zu l Beziehungen:
  - ▶ ON DELETE SET NULL ist nicht erlaubt!
  - ▶ ON UPDATE SET NULL ist nicht erlaubt!
- ▶ Es gelten die Hinweise zum kaskadierenden Löschen aus Kapitel 2
- ▶ Wir fügen zu jedem Fremdschlüssel eine ON DELETE Eigenschaft hinzu
- ▶ Wir fügen zu jedem Fremdschlüssel eine ON UPDATE Eigenschaft hinzu
- ▶ ON UPDATE CASCADE wird grundsätzlich empfohlen

# Relation Verknuepfung (vollständig)

---

CREATE TABLE Verknuepfung

( VerkNr CHARACTER(4) REFERENCES Verkaeuer  
ON DELETE NO ACTION  
ON UPDATE CASCADE,

ProdNr CHARACTER(4) REFERENCES Produkt  
ON DELETE NO ACTION  
ON UPDATE CASCADE,

Umsatz INTEGER,

PRIMARY KEY (VerkNr, ProdNr)

);

# Relation KFZ (vollständig)

---

CREATE TABLE KFZ

( KFZNr     INTEGER,

MotorNr    INTEGER    REFERENCES Motor  
                         ON DELETE SET NULL  
                         ON UPDATE CASCADE,

PRIMARY KEY (KFZNr),

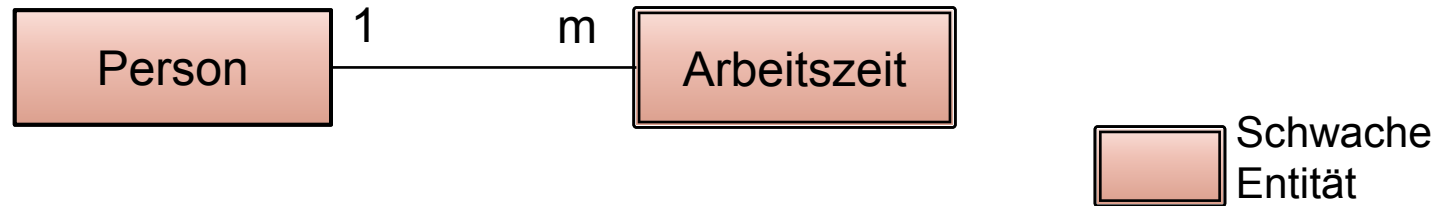
UNIQUE (MotorNr),

... );

Falls Motor kaputt und entsorgt wird, so wird automatisch die MotorNr auf NULL gesetzt!

# Schwache Entität

---



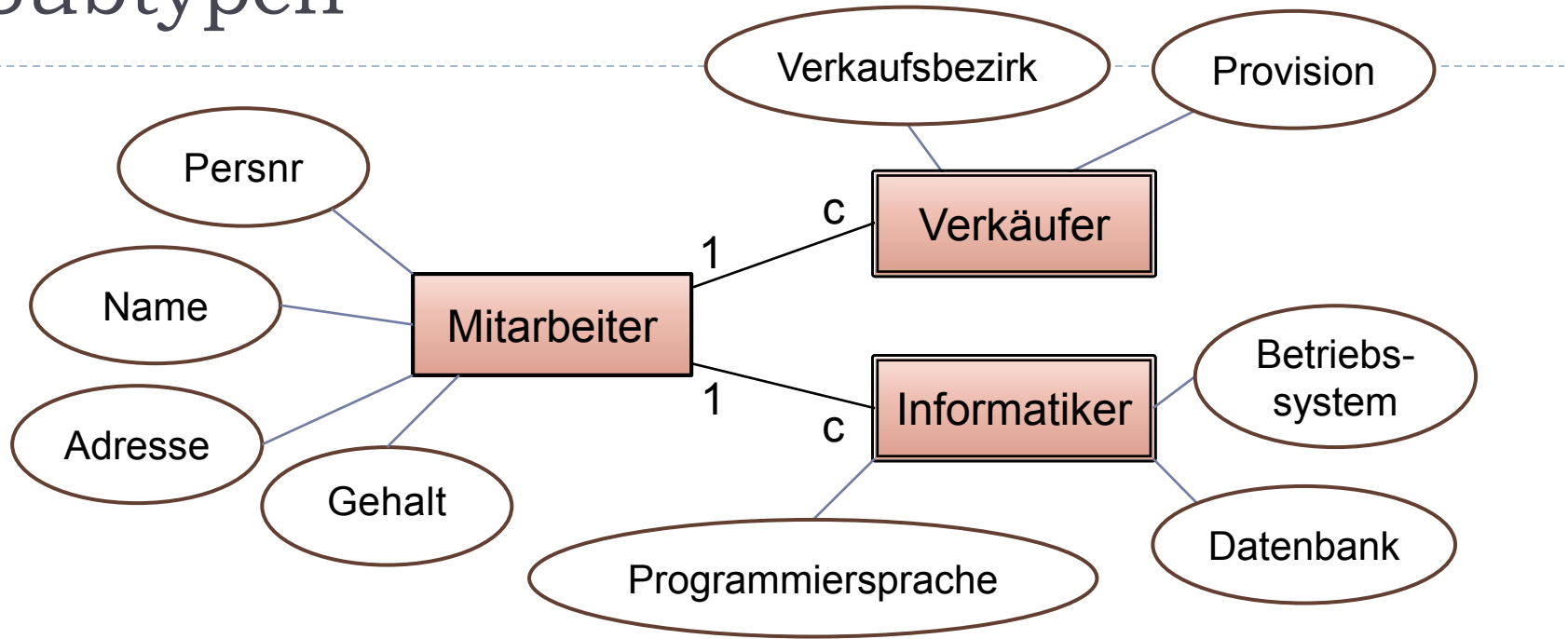
- ▶ Arbeitszeit ist vollständig abhängig von der Person
- ▶ Also: Arbeitszeit ist schwache Entität gegenüber Person
- ▶ Für schwache Entitäten gilt generell:
  - ▶ NOT NULL
  - ▶ ON DELETE CASCADE
  - ▶ ON UPDATE CASCADE

# Definition (Schwache Entität)

---

- ▶ Eine Entität heißt schwach, wenn für die dazugehörige Relation R gilt:
  - R enthält genau einen Fremdschlüssel mit den drei Eigenschaften Not Null, On Delete Cascade und On Update Cascade.
  - Auf R verweist kein Fremdschlüssel.
- ▶ Eine Relation, auf die Fremdschlüssel verweisen, ist nicht schwach!

# Subtypen



- ▶ Verkäufer und Informatiker sind Subtypen zu Mitarbeiter
- ▶ Subtypen sind schwach und enthalten Zusatzinformationen
- ▶ Es liegen 1 zu c Beziehungen vor!

# Subtyp Verkäufer

---

CREATE TABLE Verkäufer

( PersNr INTEGER REFERENCES Mitarbeiter  
ON DELETE CASCADE  
ON UPDATE CASCADE,

PRIMARY KEY (PersNr),

... );

Primärschlüssel ist  
auch Fremdschlüssel

NOT NULL  
ON DELETE CASCADE  
ON UPDATE CASCADE  
Keine weiteren Fremdschlüssel  
☐ schwache Entität!



# Zusammenfassung

---

- ▶ Bestimmen aller Entitäten mit ihren Eigenschaften.
- ▶ Ermittlung der Beziehungen zwischen den einzelnen Entitäten.
- ▶ Überführung der Entitäten in Relationen und der Eigenschaften in die Attribute dieser Relationen.
- ▶ Normalformen beachten!
- ▶ Überführung der  $m$  zu  $n$  Beziehungen in Beziehungsrelationen und aller anderen Beziehungen in Fremdschlüssel.
- ▶ Ermittlung der Eigenschaften der Fremdschlüssel. Hier helfen Stichworte wie *schwache Entität* und *Subtyp* weiter.