2 Datenmodelle

2.1 ER-Modell

Konzepte ER anhand Fluglinien-DB vergegenwärtigen; insbesondere Unterschied zwischen FLUG und ABFLUG und zwischen FLUGZEUG und FLUGZ.-MODELL; Unterschied zwischen funktionaler Beziehung und ist-Beziehung (Beschriftung immer ist)

Alternative Modellierungsmöglichkeiten in Fluglinien-DB

1. Pilot wird boolesches Attribute von Angestellter; es folgt: #Flugst für alle Angestellten (unnötige Angabe)

2. Flugz.Modell-Attribute werden Attribute von Flugzeug; es folgt: Hersteller und Modell# zu jedem Flugzeug (redundante Angabe)

3. Entity EINSATZ mit Personal#, Datum, Flug# statt Relationship eingesetzt_fuer

GRUNDSÄTZLICH: Einteilung Entity/Relationship/Attribut nicht zwingend; hängt von Entwurfsentscheidungen ab

---

ER-Modellierungskonzepte

1. Werte = darstellbare Daten; eingeteilt in fest gegebene Datentypen D1, D2, ...; z.B. integer, string, time, ...; verwendet für Attribute wie Name:string, Gehalt:integer

Interpretation (unabhängig von einem DB-Zustand):

Datentyp D $\mapsto$ Wertemenge $|D|$

z.B. integer $\mapsto \mathbb{Z}$ oder integer $\mapsto$ [-128,127] oder

integer $\mapsto \mathbb{Z} \cup \{\perp\}$ oder integer $\mapsto$ [-128,127] $\cup\{\perp\}$

---

2. Objekte (Entities); werden eingeteilt in frei definierbare Objekttypen bzw. Entitytypen E1, E2, ...

z.B. FLUGZEUG, ANGESTELLTER, BUCH

graphische Darstellung mit Rechtecken

Interpretation (in einem DB-Zustand σ):

Objekttyp E $\mapsto$

- $\mu(E)$ Menge der möglichen Objekte vom Typ E

- $\sigma(E)$ Menge der aktuellen Objekte von Typ E im Zustand σ

- wobei $\sigma(E) \subseteq \mu(E)$ und $\sigma(E)$ endlich

μ möglich; σ state (Zustand)

μ(PASSAGIER) $= \{p_1, p_2, ...\}$ $(p_i \mid i \in \mathbb{N})$

σ(PASSAGIER) $= \{p_{42}, p_{43}, p_{1001}\}$

---

3. Objekt-Attribute: A1, A2, ...; frei definierbar; z.B. Name:string, Flug#, Abfl.zeit:time, ...

graphische Notation: abgerundete Rechtecke; Angabe von Datentyp optional

Zu Attribut A gehört Objekttyp E und Datentyp D

Notation: A: E $\rightarrow$ D

z.B. Name: ANGESTELLTER $\rightarrow$ string

Flug# : FLUG $\rightarrow$ integer

Abfl.zeit : FLUG $\rightarrow$ time

...

Interpretation im DB-Zustand σ:

A : E $\rightarrow$ D interpretiert durch totale Funktion $\sigma(A) : \sigma(E) \rightarrow |D|$

z.B. $\sigma(Name) : \sigma(ANGESTELLTER) \rightarrow |string|$

Alternative Notationen für Objekttypen:

A1 : E $\rightarrow$ D1, ..., An : E $\rightarrow$ Dn

E(A1 : D1, ..., An : Dn)

E(A1, ..., An)

z.B. FLUG(Flug#:integer, Abfl.ort:string, Abfl.zeit:time, Ank.ort:string, Ank.zeit:time)

---

4. Beziehungen (Relationships):

Beziehungstypen R1, R2, ...

graphische Darstellung: Rauten

z.B. gebucht_fuer, kann_fliegen

zu einem n-stelligen Beziehungstyp R gehört Liste mit n Objekttypen E1, ..., En; n>=2; Ei=Ej ist erlaubt

Notation: R(E1, ..., En)

z.B. gebucht_fuer(PASSAGIER,ABFLUG)

lieferung(WARE,LIEFERANT,KUNDE) Obacht: mehr-stellige Beziehungen möglich

Interpretation im Zustand σ:

R(E1,...,En) $\mapsto$ Relation $\sigma(R)$ mit $\sigma(R) \subseteq \sigma(E1) \times ... \times \sigma(En)$

Obacht: Interpretation eines Relationships ist Relation, also eine Menge; ein mögliches Tupel kann in der Relation $\sigma(R)$ (a) nicht vorhanden sein oder (b) einmal vorhanden sein; ein Tupel kann aber nicht *mehrmals* in $\sigma(R)$ vorkommen; Beispiel dazu kommt unten mit dem Relationship AUSLEIHE

---

5. Beziehungsattribute:

analog zu Objekt-Attributen

A : R $\rightarrow$ D bzw. R(E1,...,En;A:D)

$\sigma(A) : \sigma(R) \rightarrow |D|$

z.B. gebucht_fuer(PASSAGIER,ABFLUG;Preis:real)

$\sigma(Preis) : \sigma(gebucht\_fuer) \rightarrow |real|$

mit $\sigma(gebucht\_fuer) \subseteq \sigma(PASSAGIER) \times \sigma(ABFLUG)$

für jedes Paar in $\sigma(gebucht\_fuer)$ wird Preis notiert

Beispiel: formale Interpretation des Bib-DB-Zustandes

$\mu(STUDENT) = \{s_i \mid i \in \mathbb{N}\}$

$\sigma(STUDENT) = \{s1, s2, s3\}$

$\sigma(BUCH) = \{b1, b2, b3\}$

...

$\sigma(SName) : \sigma(STUDENT) \rightarrow |string|$

$\sigma(SName) = \{(s1 \mapsto Meier), (s2 \mapsto Schulz), (s3 \mapsto Schmidt)\}$

(kurz für $\sigma(SName) = \{(s1 \mapsto 'Meier'), (s2 \mapsto 'Schulz'), (s3 \mapsto 'Schmidt')\}$)

...

$\sigma(AUSLEIHE) \subseteq \sigma(STUDENT) \times \sigma(BUCH)$

$\sigma(AUSLEIHE) = \{(s1, b1), (s2, b1), (s2, b2)\}$

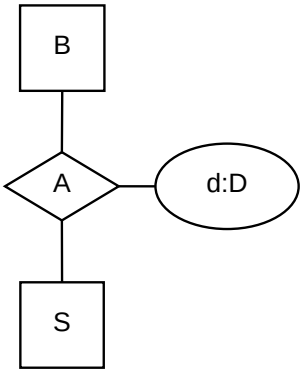
...

$\sigma(Datum) : \sigma(AUSLEIHE) \rightarrow |date|$

$\sigma(Datum) = \{((s1, b1) \mapsto 01.01.99), ((s2, b1) \mapsto 02.02.99), ((s2, b2) \mapsto 03.03.99)\}$

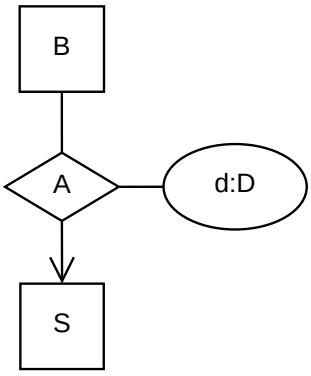
$\sigma(AUSLEIHE)$ ist Menge mit endlich vielen (STUDENT,BUCH)-Paaren; jedem Paar wird genau ein Datum zugeordnet; d.h. mit diesem ER-Schema können nicht mehrere Ausleihvorgänge des selben Studenten mit dem selben Buch beschrieben werden; wenn man das beschreiben will kann z.B. eine 3-stellige Beziehung verwenden oder aber ein mehrwertiges Attribut (z.B. Datum:Set(Date) oder Datum:List(Date)); im reinen ER-Modell gibt solche mehrwertigen Attribute nicht, sehr wohl aber im erweitereten ER-Modell und in UML

für gegebenes Buch und gegebenen Student maximal eine Ausleihe



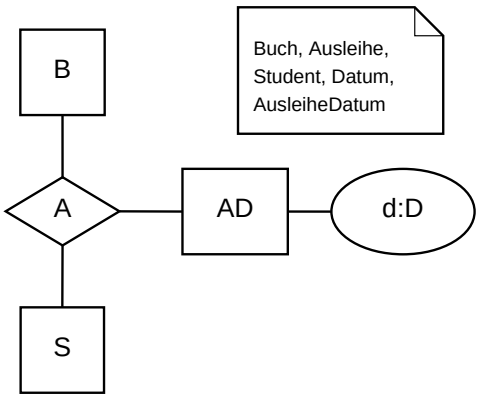
keine oder eine Ausleihe (b,s) mit b in sigma(B) und s in sigma(S); (b,s1) und (b,s2) erlaubt

für gegebenes Buch maximal nur eine Ausleihe



keine oder eine Ausleihe (b,s); (b,s1) und (b,s2) verboten

für gegebenes Buch und gegebenen Student auch mehrere Ausleihen



(b,s,ad1) und (b,s,ad2) erlaubt; (b,s1,...) und (b,s2,...) erlaubt

...

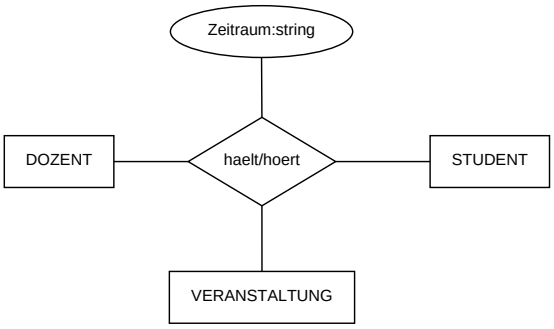
Weitere Beispiele für Beziehungstypen

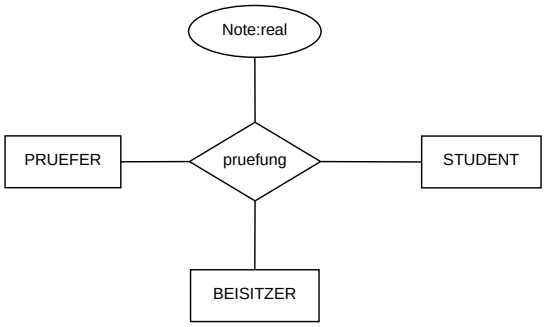
haelt/hoert(DOZENT,VERANSTALTUNG,STUDENT;Zeitraum:string)

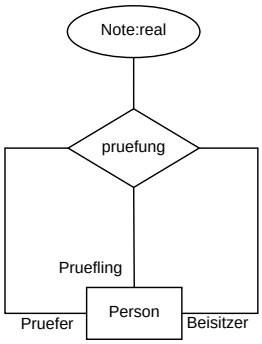
pruefung(PRUEFER,BEISITZER,STUDENT;Note:real)

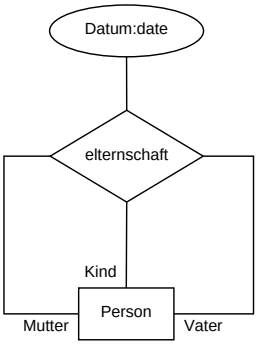
pruefung(Pruefer:PERSON,Pruefling:Person,Beisitzer:PERSON;Note:real)

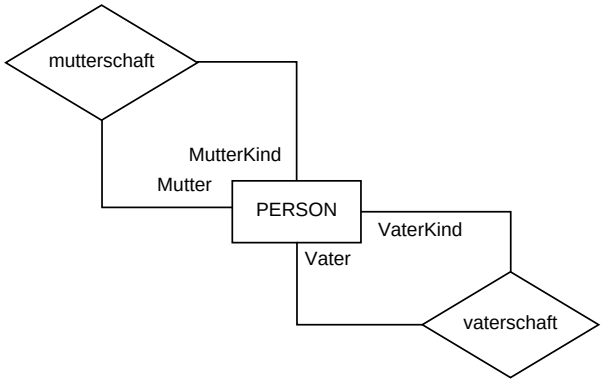
elternschaft(Mutter,Kind,Vater:PERSON;Datum:Date)

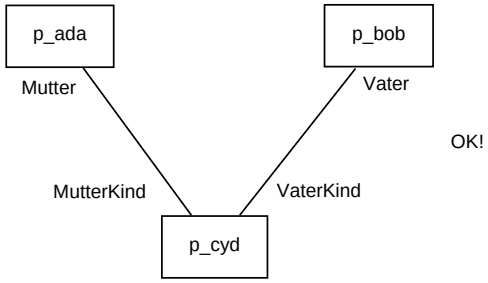


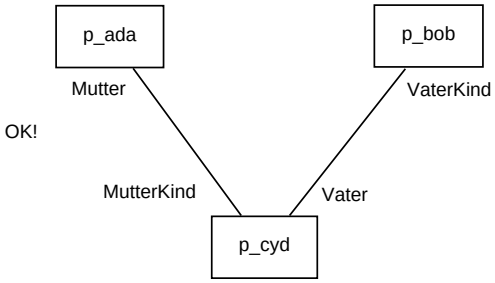




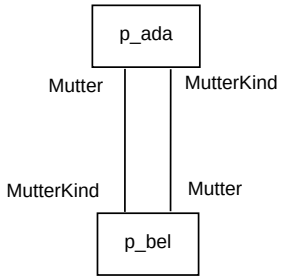








Nach ER-Schema erlaubt, aber kein gültiger Zustand der realen Welt!



mutterschaft(Mutter:PERSON,MutterKind:PERSON)

vaterschaft(Vater:PERSON,VaterKind:PERSON)

weiteres ER-Konzept: Rollennamen; obligat falls ein Entity mehrmals an einer Beziehung teilnimmt oder falls sonst keine eindeutige Navigation zwischen den Entities möglich ist; insbesondere wenn mehr als eine Beziehung zwischen 2 Entities besteht

R1(r:E1,s:E2), R2(t:E1,u:E2)

MuendlichePruefung(pruefer:PROF,pruefling:STUD)

Abschlussarbeit(betreuer:PROF,betreuter:STUD)

für PROF einstein: einstein.pruefling, einstein.betreuter

[oder pruefling(einstein), betreuter(einstein)]

Erinnerung: Integritätsbedingungen sind Teil von DB-Schemata (nicht nur die Strukturen)

obige Schemata erlauben (teilweise) noch merkwürdige DB-Zustände

Folgerung: Schema mit Integritätsbedingungen vervollständigen

---

Weitere ER-Modellierungs-Konzepte

funktionale Beziehung

z.B. Typ : FLUGZEUG $\rightarrow$ FLUGZ.-MODELL

z.B. verboten für R : E $\rightarrow$ F

$\sigma(E) = \{e1, e2\}$

$\sigma(F) = \{f1, f2\}$

$\sigma(R) = \{(e1, f1), (e1, f2), (e2, f2)\}$

z.B. $\sigma(Typ) = \{(flugz42, a320), (flugz42, b747), (flugz43, a340)\}$

Generalisierung E1 ist E2 (E1 ist spezieller als E2 oder 'E1<=E2')

graphische Darstellung: Pfeilspitze zum allgemeineren Typ

z.B. PILOT ist ANGESTELLTER

Attribut-Vererbung

im Beispiel: PILOT-Objekt kann als Objekt vom Typ ANGESTELLTER verwendet werden; also kann man jedes Attribut von ANGESTELLTER auch auf ein Objekt vom Typ PILOT anwenden

Attribute dürfen aber keine Namenskonflikte erzeugen; dann auch Mehrfach-Vererbung erlaubt

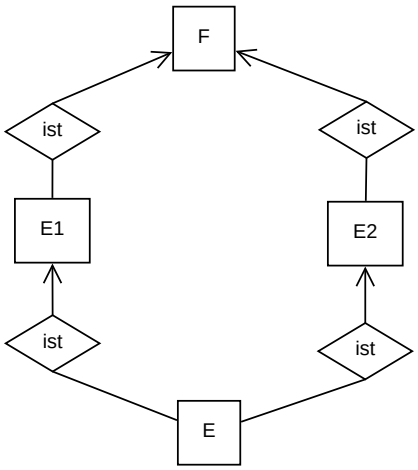
problematische Beispiele mit Mehrfach-Vererbung:

Beispiel 1: STUDENT ist ANGESTELLTER

STUDENT(..., Steuersatz, ...), ANGESTELLTER(..., Steuersatz, ...)

s sei STUDENT-Objekt; Steuersatz(s)?

Ausweg: Attribut Steuersatz nur in einem Entitytyp definieren



Beispiel 2: E ist E1, E ist E2

E1(..., A:D, ...), E2(..., A:D, ...)

e sei E-Objekt; A(e)? A(e als E1-Objekt)? A(e als E2-Objekt)?

Ausweg: Attribute unterschiedlich benennen ODER

Attribut A nur in Entitytyp E zulassen ODER

Attribut A in muss in Entitytyp F definiert sein für den gilt E1 ist F und E2 ist F

Schlüssel

$\forall \underline{e}, \underline{e}' \in \sigma(E) : [\underline{e} \neq \underline{e}'] \Rightarrow [\exists i \in \{1, ..., k\} : \sigma(S_i)(\underline{e}) \neq \sigma(S_i)(\underline{e}')]$

$\Leftrightarrow$

$\forall \underline{e}, \underline{e}' \in \sigma(E) : [\forall i \in \{1, ..., k\} : \sigma(S_i)(\underline{e}) = \sigma(S_i)(\underline{e}')] \Rightarrow [\underline{e} = \underline{e}']$

Beobachtung: Gesamtattributmenge darf als Schlüssel gewählt werden

Anhand Folien vergenewärtigen: BUCH mit Schlüssel Doknr; DESKRIPTOR mit Schlüssel Wort; STUDENT mit Schlüssel Matnr; AUTOR mit Schlüssel AName; Schlüsselangabe ist im ER-Modell i.a. nicht zwingend notwendig (allerdings Voraussetzung für Übersetzung ins relationale Datenmodell)

Beispiel zu Schlüsselattributen

PERSON(<u>VName</u>,<u>NName</u>,Adresse) mit Schlüssel {VName,NName}

$\sigma(PERSON) = \{p1, p2, p3, p4\}$

$\sigma(VName/NName) : p1 \mapsto (Ada, Smith)$

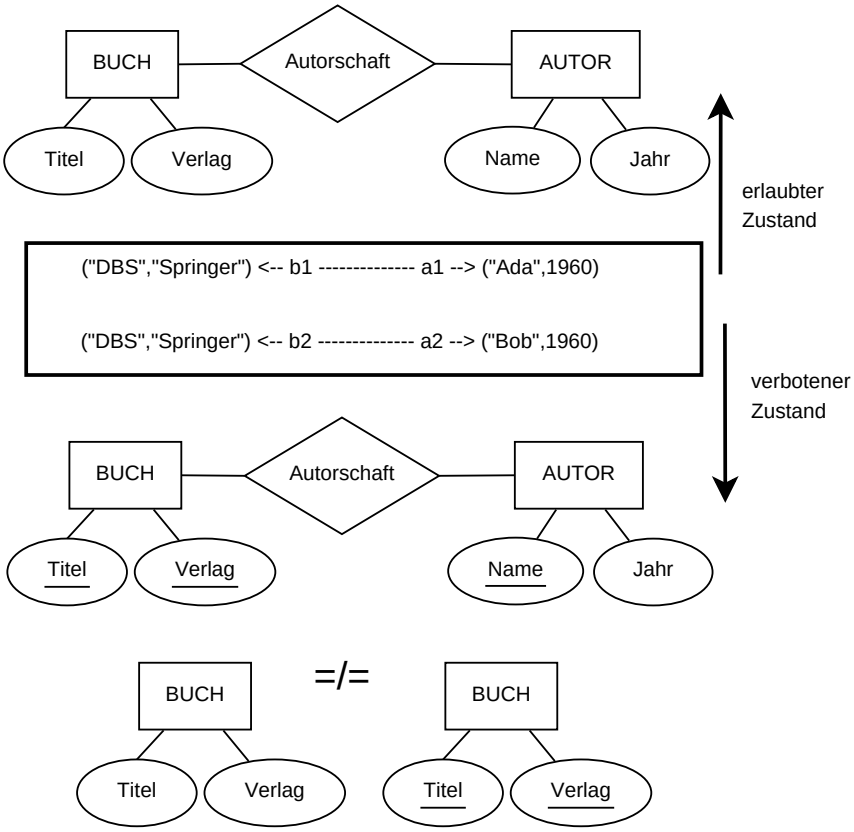
$\sigma(VName/NName) : p2 \mapsto (Bob, Smith)$

$\sigma(VName/NName) : p3 \mapsto (Ada, Jones)$

$\sigma(VName/NName) : p4 \mapsto (Bob, Smith)$

ist kein gültiger Zustand; aber '$\sigma - p4$' ist gültiger Zustand

---



ER-Schemata ohne Schlüssel: objektbasiert

Objekte können sich nicht nur durch Attribute, sondern auch durch Teilnahme an Beziehungen unterscheiden (eventuell können sich Objekte in einem Zustand nach außen nicht unterscheiden)

ER-Schemata mit Schlüsseln: wertebasiert

Objekte müssen sich durch Attribute unterscheiden; Festlegung der Schlüssel schränkt DB-Zustände ein

Beispiel: Schlüssel für Fluglinien-DB

Entitytyp	Schlüssel
PASSAGIER	Name
ANGESTELLTER	Personal#
FLUG	Flug#
PILOT	Personal#, vererbt von Angesteller
FLUGZ.-MODELL	Hersteller, Modell#
FLUGZEUG	Serien#
ABFLUG	?

ABFLUG: Schlüsselvergabe kann sinnvoll nur nach Aufnahme weiterer Attribute erfolgen; 2 Alternativen:

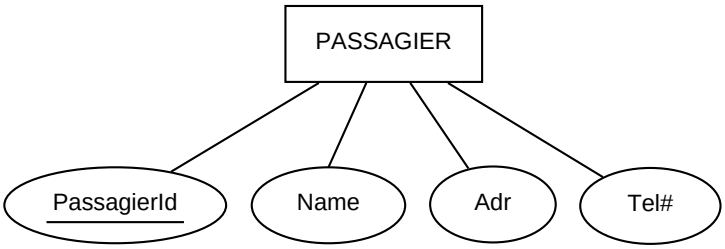
1. Flug# in ABFLUG aufnehmen und {Flug#,Datum} wird Schlüssel

Relationship Exempl.von ist dann unnötig, da Flug# in ABFLUG auf genau ein FLUG-Objekt verweist

2. AbflugId (mit Typ integer) aufnehmen und {AbflugId} wird Schlüssel

AbflugId = Zähler, der (z.B.) mit 1 initialisiert wird und bei der Erzeugung eines neuen ABFLUG-Objekts inkrementiert wird

Obacht: AbflugId ist kein Merkmal des modellierten Weltausschnitts; Merkmale (Attribute) des Weltausschnitts sind in der Regel geeigneter als Schlüsselattribute; Verwendung von Ids als Schlüssel birgt die Gefahr von Objektduplikaten



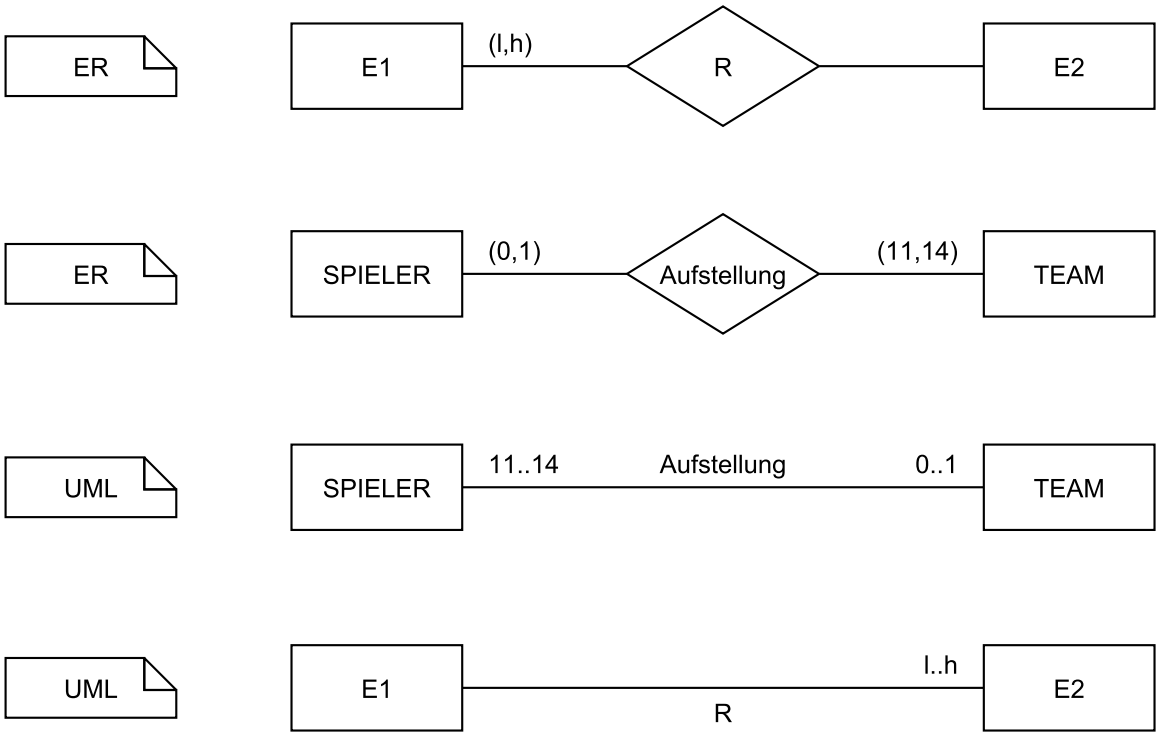
gültiger Zustand:
(42, "Ada White", "NY", 4711)
(43, "Ada White", "NY", 4711)

Weiteres Modellierungskonzept: Kardinalitäten

$l \in \mathbb{N}_0, h \in \mathbb{N} \cup \{*\}, l \leq h$ mit $l \leq *$ für $l \in \mathbb{N}_0$

Entity vom Typ E1 nimmt mindestens l-mal und höchstens h-mal an $\sigma(R)$ teil; l(ow), h(igh); Constraint für Teilnahmefähigkeit

$\forall e_1 \in \sigma(E1) : l \leq COUNT\{r \in \sigma(R) | r.E1 = e_1\} \leq h$



Entity vom Typ E1 steht mittels R mit mindestens l und höchstens h Entities vom Typ E2 in Verbindung; Constraint für Verbindungsfähigkeit

$\forall e_1 \in \sigma(E1) : l \leq COUNT\{e_2 \in \sigma(E2) | (e_1, e_2) \in \sigma(R)\} \leq h$

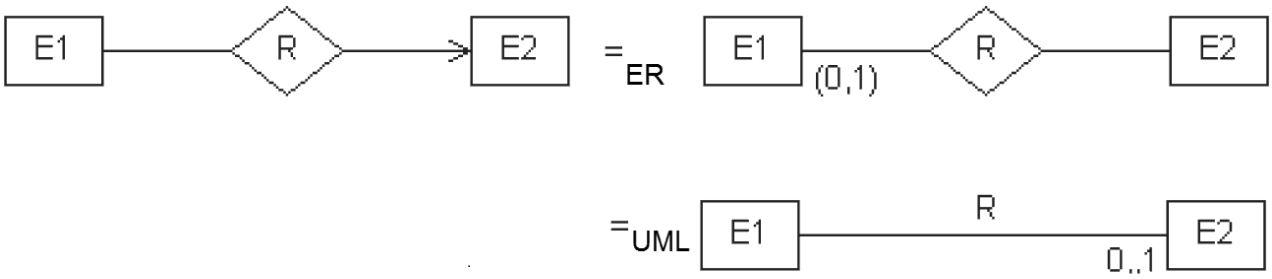
unterschiedliche Syntax für den gleichen Sachverhalt; Analogie: Programmiersprachen mit Bedingungen und Statements

blockorientierte Programmiersprache: if B then S1 else S2 endif; B $\rightarrow$ true/false

maschinennahe Programmiersprache: cond(B,S2,S1); B $\rightarrow$ 0/1; (false $\approx$ 0, true $\approx$ 1)

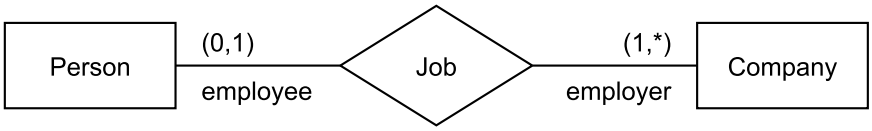
ER-Syntax: (l,h); UML-Syntax: l..h

ER-Default: (0,*); UML-Default: 0..*; d.h. keine Einschränkung; optional

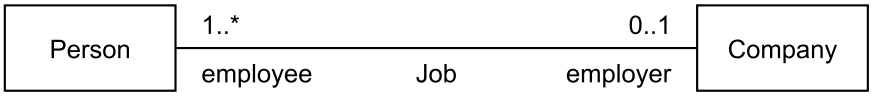


Vorsicht bei Untergrenze >0 :

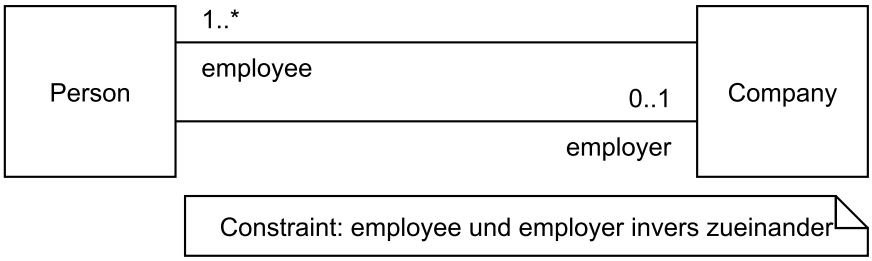
A



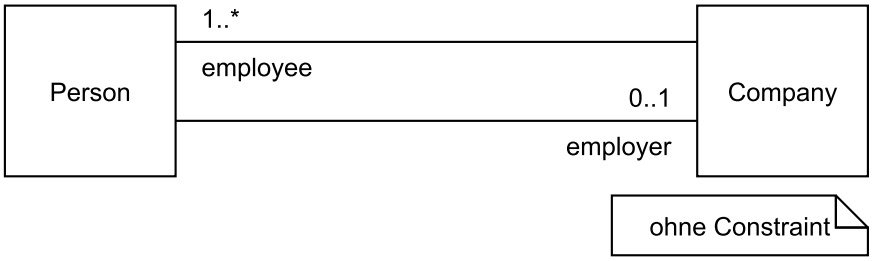
B



C



D



A, B, und C äquivalent; D unterschiedlich

2.2 Relationenmodell

Konzepte:

1. Werte wie im ER-Modell

2. Relationen

Relationenschema R(A1:D1, ..., An:Dn)

entspricht Entitytyp R mit Attributen Ai : R $\rightarrow$ Di; obligatorische Schlüsselangabe

{A1, ..., An} als Schlüssel erlaubt; Datentypen können auch weggelassen werden, wenn aus Kontext klar

Interpretation:

R $\mapsto$ endliche Relation $\sigma(R) \subseteq \mu(R) = |D1| \times ... \times |Dn|$

Tupel $t = (t1, ..., tn) \in \mu(R)$ entspricht einem möglichen Objekt vom Typ R

Relation $\sigma(R)$ entspricht der Menge der aktuellen Objekte vom Typ R

Ai induziert Funktion $\sigma(Ai) : \sigma(R) \rightarrow |Di|$; $t \mapsto ti$

Beobachtung: im Relationenmodell gibt es kein Konstrukt das Relationships direkt entspricht

---

Beispiel (Angabe von Schlüsseln wieder durch unterstreichen):

BUCHUNG(Name:string, Adresse:string, TelNr:digits, FlugNr:char(5), Datum:date, VonNach:string)

$\mu(BUCHUNG) = |string| \times |string| \times |digits| \times |char(5)| \times |date| \times |string|$

$= C^* \times C^* \times Z^* \times C^5 \times D \times C^*$ (weiter eingeschränkt durch Schlüsselbedingung)

mit $C = \{A, B, ..., a, b, ..., 0, ..., 9, ...\}$

$Z = \{0, ..., 9\}$

$D = \{w \in C^* | w \; korrektes \; Datum \; im \; Format \; TT.MM.JJ\}$

$\mu(BUCHUNG) \cong C^* \times C^5 \times D$ (wegen der gewählten Schlüssel)

$\sigma(BUCHNUG)$	Name	Adresse	TelNr	FlugNr	Datum	VonNach
t1	Ada	HB	3271	LH093	31.12.99	H-F
t2	Ada	HB	3271	LH094	31.12.99	F-M
t3	Ada	HB	3271	LH093	31.12.99	M-H

{t1,t2} und {t2,t3} gültiger Zustand; {t1,t2,t3} und {t1,t3} ungültiger Zustand, da t1 und t3 sich nicht in den Schlüsselattributen unterscheiden

---

Begriff: Relationales DB-Schema = Sammlung mehrerer Relationenschemata mit verschiedenen Namen

Grundoperationen für Änderungen

R(A1:D1, ..., An:Dn); $t, t1, t2 \in |D1| \times ... \times |Dn|$

ω :	DB-Zustände	(x Parameter)	$\rightarrow$	DB-Zustände
	σ_{alt}	...	$\mapsto$	σ_{neu}

1. ω = insert(R,t)

$\sigma_{neu}(R)$:= if Schlüsselbedingung bleibt erfüllt

then $\sigma_{alt}(R) \cup \{t\}$

else $\sigma_{alt}(R)$ + Warnung

2. ω = delete(R,t)

$\sigma_{neu}(R) := \sigma_{alt}(R) - \{t\}$ + Warnung falls $t \notin \sigma_{alt}(R)$

3. ω = change(R,t1,t2) = [delete(R,t1); insert(R,t2)]

Beispiel: ER-Schema $\rightarrow$ Relationales DB-Schema

nach 1.: BUCH(<u>Doknr</u>,Titel,Verlag,Ort,Jahr)

STUDENT(<u>Matnr</u>,SName,Fach,Sem#)

DESKRIPTOR(<u>Wort</u>)

AUTOR(<u>AName</u>)

nach 2.: AUSLEIHE(<u>Doknr</u>,<u>Matnr</u>,Datum)

BA(<u>Doknr</u>,<u>AName</u>)

BD(<u>Doknr</u>,<u>Wort</u>)

nach 3.: BA umbenennen zu AUTOREN

BD umbenennen zu DESKRIPTOREN

DESKRIPTOR, AUTOR streichen

zusätzliche Annahme hierfür: Jeder Deskriptor steht in Beziehung zu mindestens einem Buch; jeder Autor ist an mindestens einem Buch beteiligt; alle vorkommenden Deskriptoren (Werte für das Attribut Wort) tauchen dann in DESKRIPTOREN auf, alle Autoren (Werte für das Attribut AName) in AUTOREN

Ergebnis:

BUCH(Doknr,Titel,Verlag,Ort,Jahr)

STUDENT(Matnr,SName,Fach,Sem#)

AUSLEIHE(Doknr,Matnr,Datum)

AUTOREN(Doknr,AName)

DESKRIPTOREN(Doknr,Wort)

Erweiterung um referentielle Integritätsbedingungen: bei der Übersetzung von Relationshiptypen benötigt man Integritätsbedingungen (IBen), die sicherstellen, dass in den Relationship-Schemata auch nur Objekte vorkommen, die auch in den Entitytyp-Schemata vorkommen, sogenannte referentielle IBen

im Beispiel (hier nur Kurznotation, auch in SQL-Syntax darstellbar):

```
AUSLEIHE.[Doknr] -> BUCH.[Doknr]
AUSLEIHE.[Matnr] -> STUDENT.[Matnr]
AUTOREN.[Doknr] -> BUCH.[Doknr]
((gestrichen AUTOR: AUTOREN.[AName] -> AUTOR.[AName]))
DESKRIPTOREN.[Doknr] -> BUCH.[Doknr]
((gestrichen DESKRIPTOR: DESKRIPTOREN.[Wort] -> DESKRIPTOR.[Wort]))

Syntax: R1.[A1,A2] -> R2.[A3,A4]
        Vorausetzung {A3,A4} Schluessel in R2
```

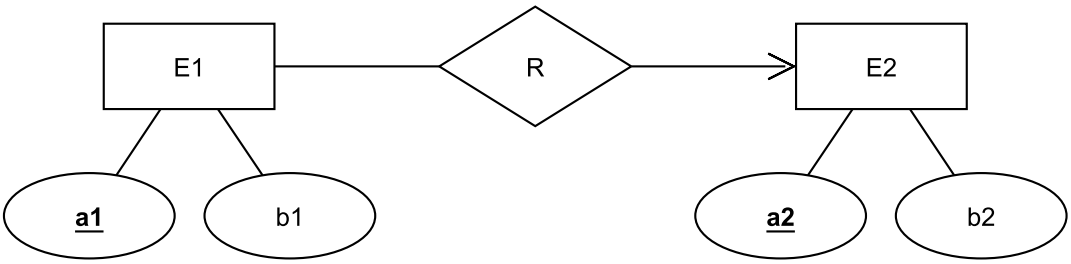
Erinnerung: Begriff "Schema"; relationales Schema, Relationenschema, relationales DB-Schema

z.B. BUCH(...) ist eine relationales Schema; z.B. STUDENT(...) ist ein Relationenschema; relationales Schema und Relationenschema sind Synonyme

{ BUCH, STUDENT, AUSLEIHE, AUTOREN, DESKRIPTOREN } ist ein relationales DB-Schema

---

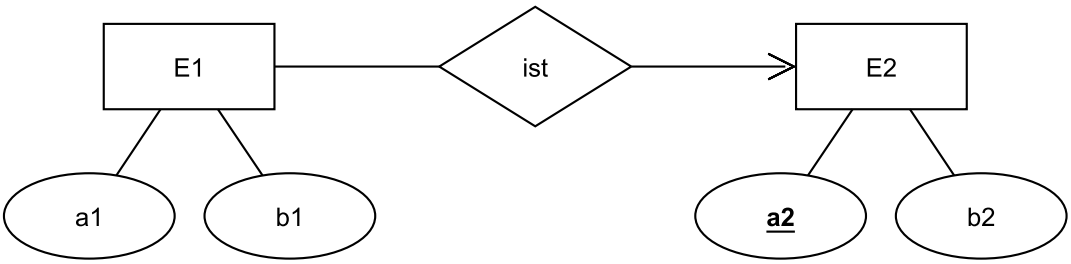
Behandlung von funktionalen Beziehungen und Generalisierungen



Relationenschemata:
E1(a1,b1,a2)
E2(a2,b2)
E1.[a2] → E2.[a2]

partielle funktionale Beziehungen durch NULL-Werte für a2 in E1 darstellen

FLUGZEUG(Serien#,Hersteller,Modell#)
FLUGZ.-MODELL(Hersteller,Modell#)
FLUGZEUG.[Hersteller,Modell#] → FLUGZ.-MODELL.[Hersteller,Modell#]



Relationenschemata:
E1(a1,b1,a2)
E2(a2,b2)
E1.[a2] → E2.[a2]

PILOT(#Flugst,Personal#)
ANGESTELLTER(Name,Adresse,Gehalt,Personal#)
PILOT.[Personal#] → ANGESTELLTER.[Personal#]

Übersetzung der restlichen Anteile des Fluglinien-DB-Schemas:

PASSAGIER(Name,Adresse,Tel#)

ABFLUG(Datum,Flug#)

gebucht_fuer(Name,Datum,Flug#,Preis)

PLUS: referentielle IBen, auch für andere Relationships

FLUG(Flug#,Abfl.ort,Ank.ort,Abfl.zeit,Ank.zeit)

kann_fliegen(Personal#,Hersteller,Modell#)

eingesetzt_fuer(Datum,Flug#,Personal#)

Andere Relationships und Generalisierung gehen ein in Entitytyp-Schemata:

Exempl.von $\mapsto$ ABFLUG

ist $\mapsto$ PILOT

TYP $\mapsto$ FLUGZEUG

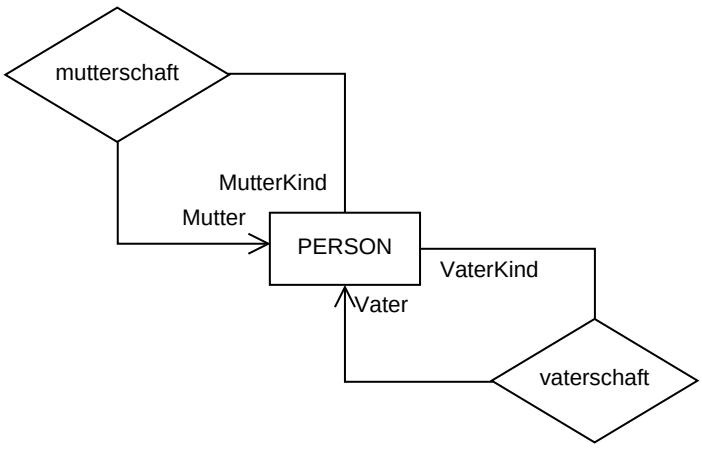
Beispiel: mutterschaft und vaterschaft

ER-Schema:

PERSON(VName,NName,Alter,Geschlecht)

mutterschaft(Mutter:PERSON,MutterKind:PERSON) funktional in Richtung Mutter

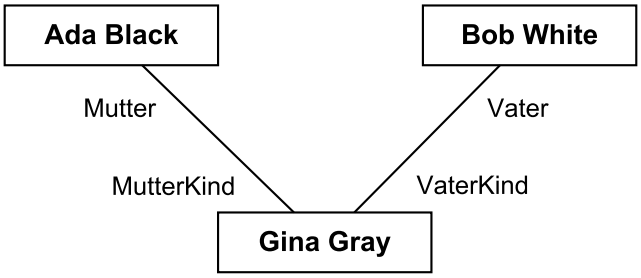
vaterschaft(Vater:PERSON,MutterKind:PERSON) funktional in Richtung Vater



relationales Schema:

PERSON(VName,NName,Alter,Geschlecht,
MVName,MNName,VVName,VNName)

PERSON.[MVName,MNName] → PERSON.[VName,NName]
PERSON.[VVName,VNName] → PERSON.[VName,NName]
IB *Mutter weiblich, Vater männlich* noch nicht gewährleistet



Person

	VName	NName	Alter	Geschlecht	MVName	MNName	VVName	VNName
	Ada	**Black**	**42**	**W**				
	Bob	**White**	**42**	**M**				
	Gina	**Gray**	**21**	**W**	**Ada**	**Black**	**Bob**	**White**

Beispiel: Erste Anfragen in SQL

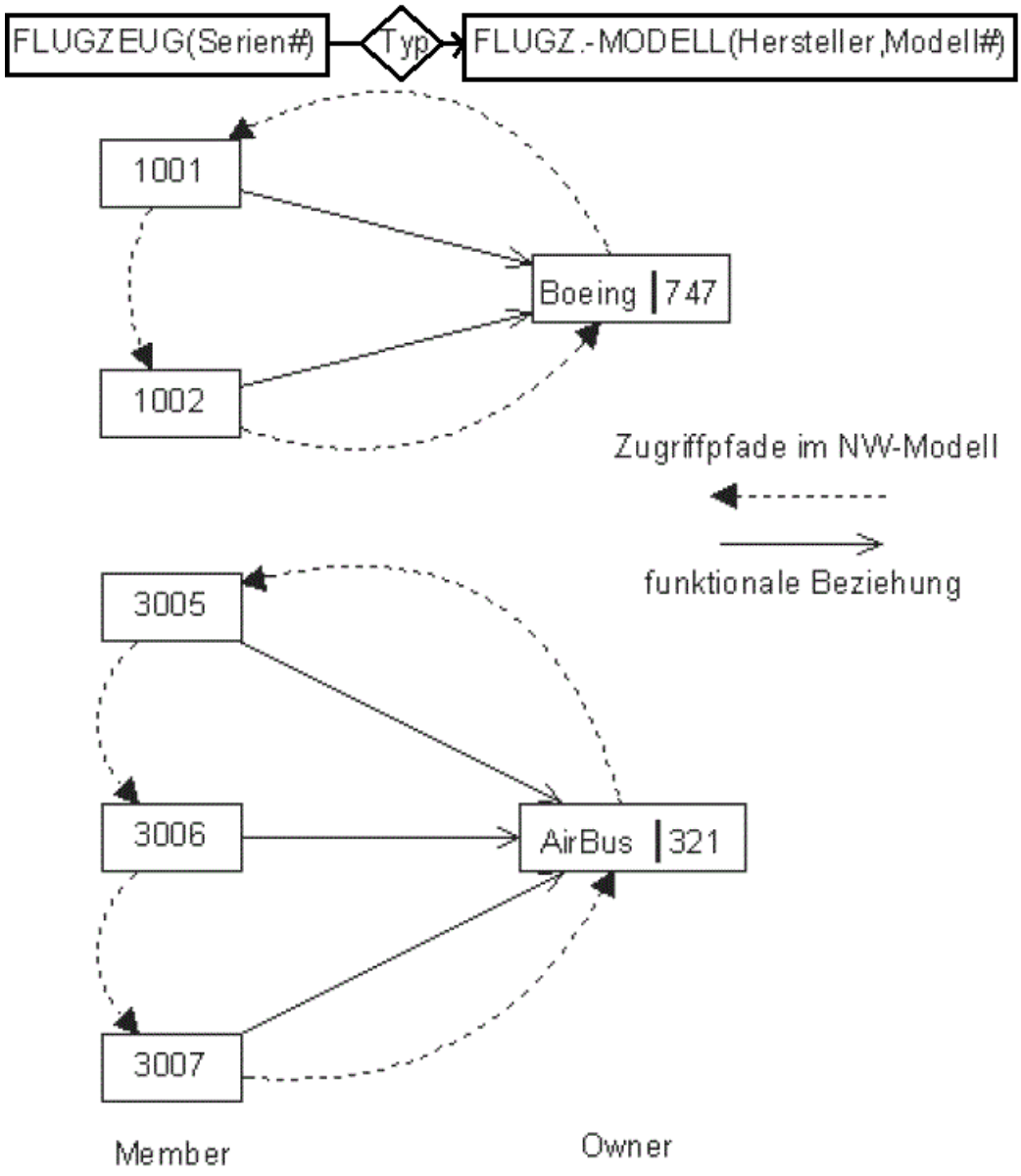
```
RELATIONENMODELL
select AUTOREN.AName
from   AUTOREN, AUSLEIHE, STUDENT
where  AUTOREN.Doknr = AUSLEIHE.Doknr and
       AUSLEIHE.Matnr = STUDENT.Matnr and
       STUDENT.SName = "Zimmermann"

ER-MODELL
select AName(a)
from   a in AUTOR -- a Variable zu Entity AUTOR
where  exists (b in BUCH, s in STUDENT)
         ( BA(b,a) and AUSLEIHE(b,s) and
           -- Parameterreihenfolge bei Relationship-Abfragen
           -- von oben nach unten, von links nach rechts
           SName(s)="Zimmermann" )
```

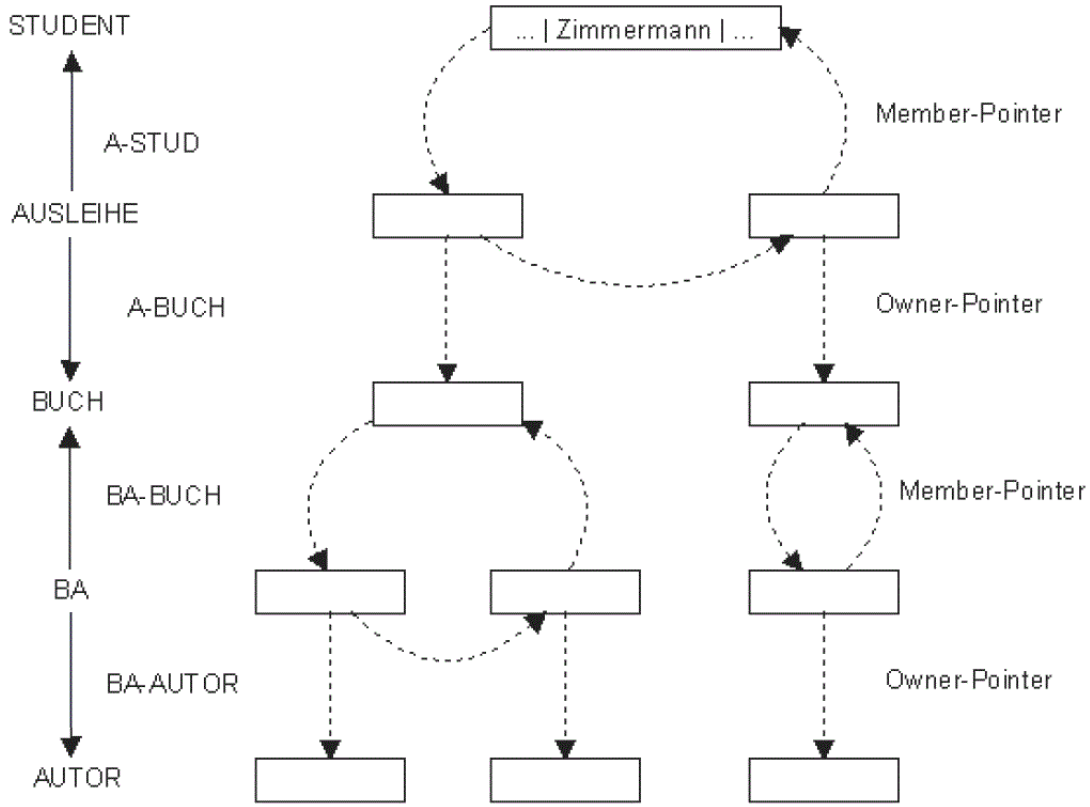
```
select AName(AUTOR(ba))
from   ba in BA -- ba Variable zu Relationship BA
where  exists (aus in AUSLEIHE)
         ( BUCH(ba)=BUCH(aus) and
           SName(STUDENT(aus))="Zimmermann" )
```

Beobachtung: ER-Anfrage kommt mit 2 Gleich-Abfragen aus, Relationenmodell braucht 3 Gleich-Abfragen; Grund: ER-Abfrage verwendet Variablen zu Relationshiptypen

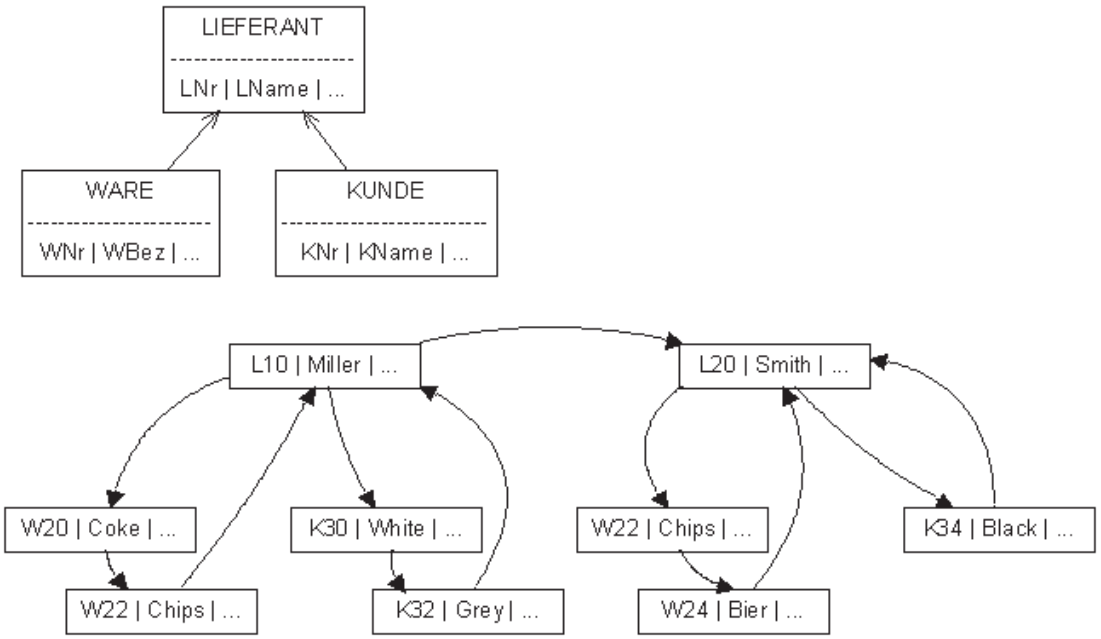
---

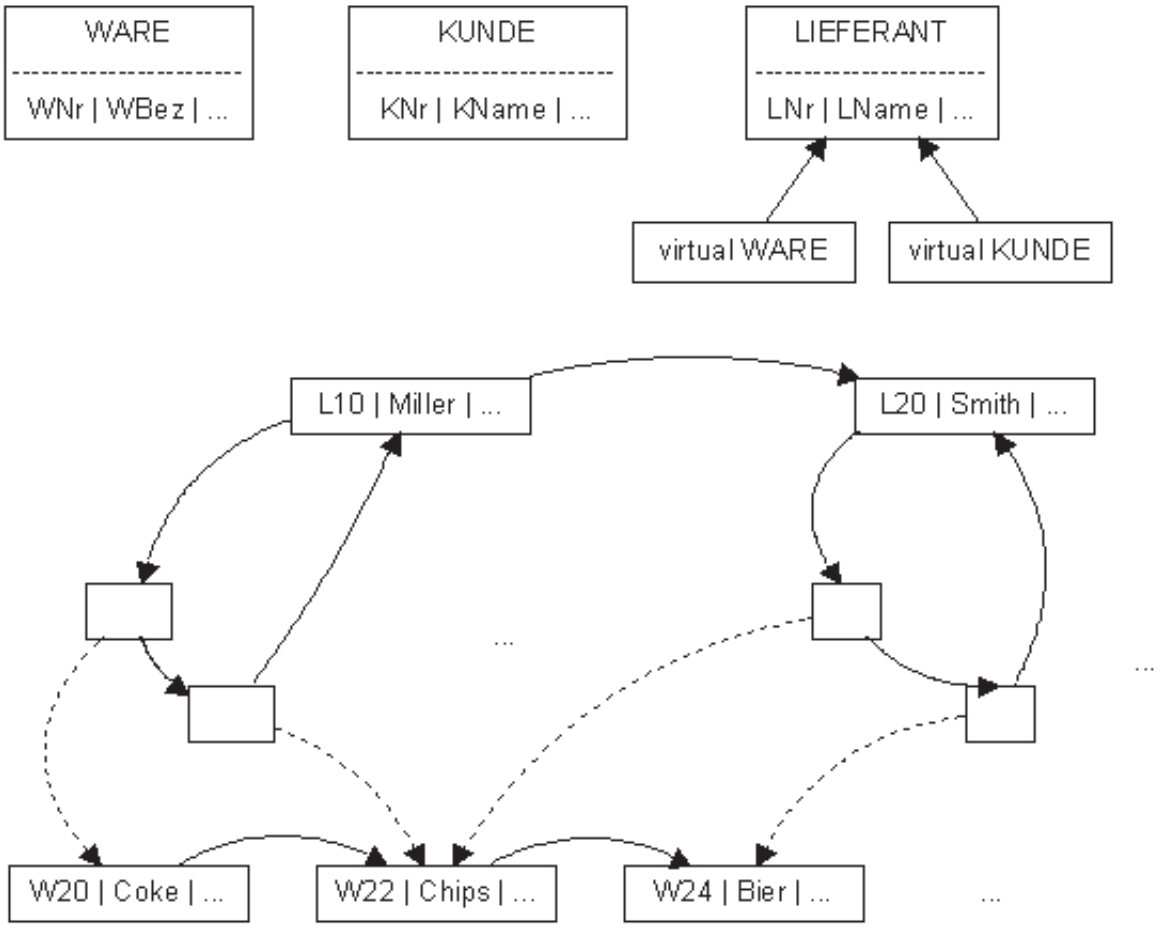
2.3 Netzwerk-Modell

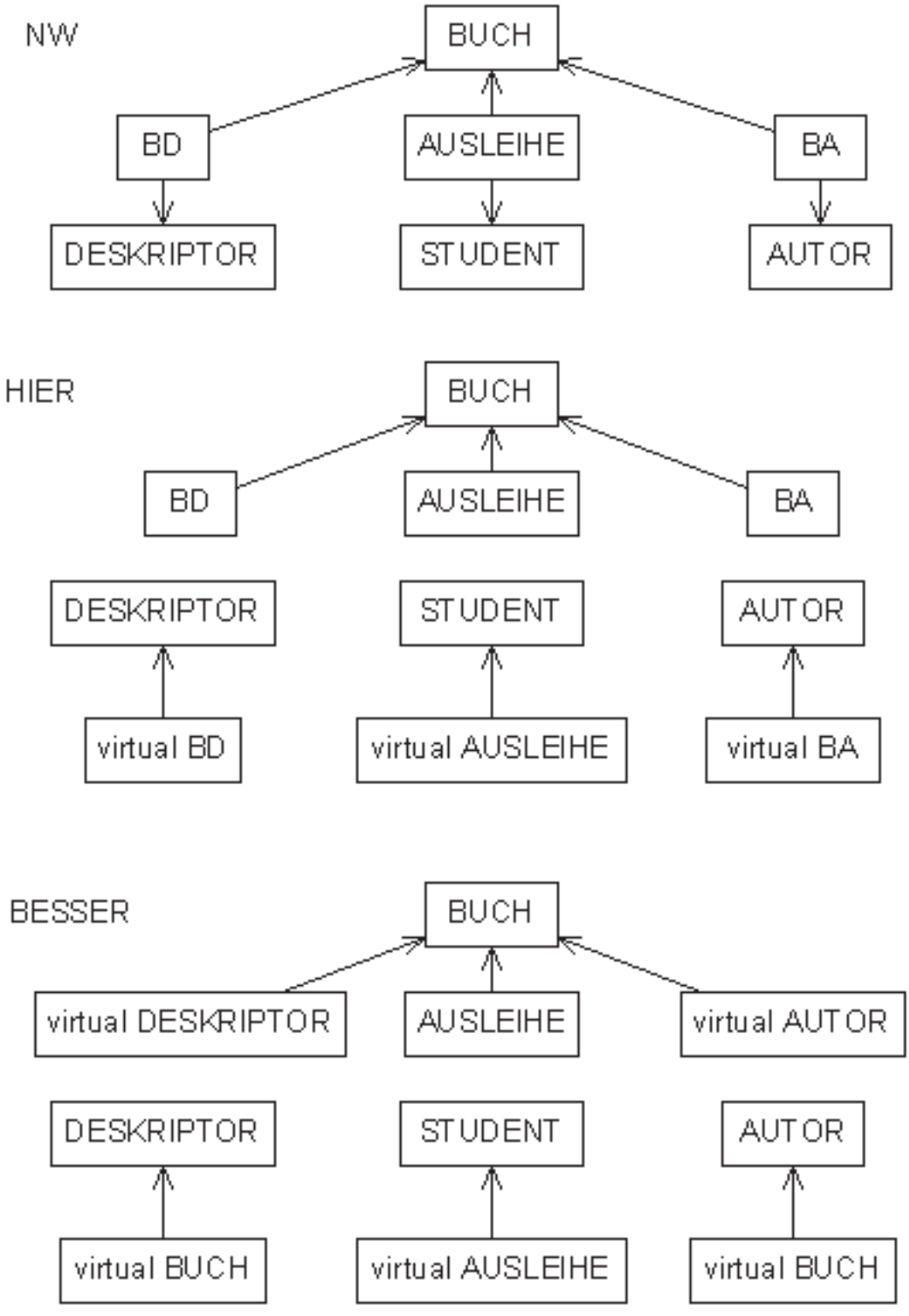




2.4 Hierarchisches Modell




WARE
WNr | WBez | ...
KUNDE
KNr | KName | ...
LIEFERANT
LNr | LName | ...
virtual WARE
virtual KUNDE
L10 | Miller | ...
L20 | Smith | ...
...
...
W20 | Coke | ...
W22 | Chips | ...
W24 | Bier | ...
...

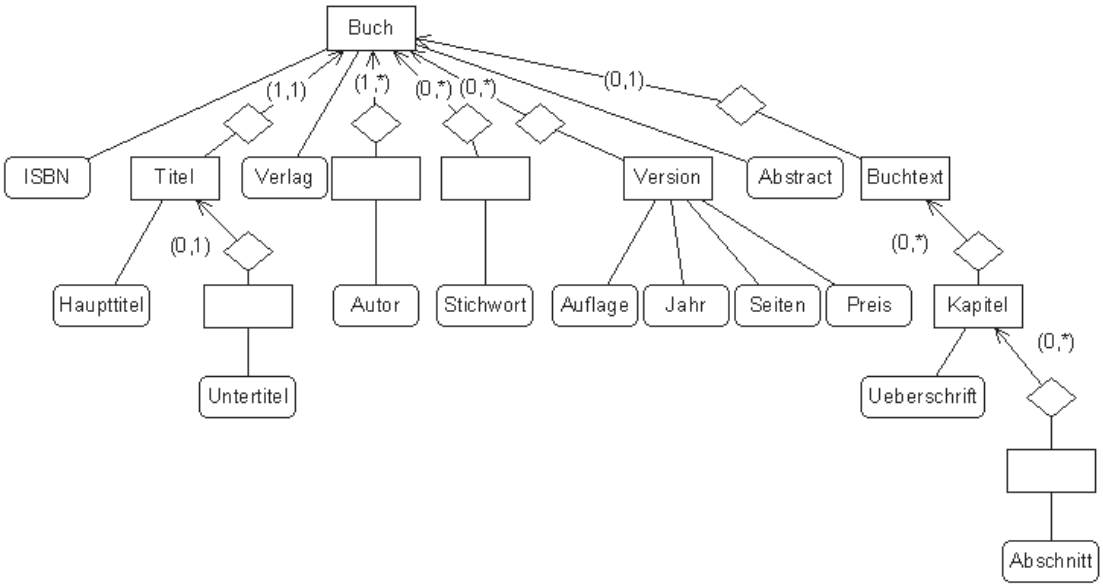


2.5 Objektorientierte Datenmodelle

(keine Ergänzungen)

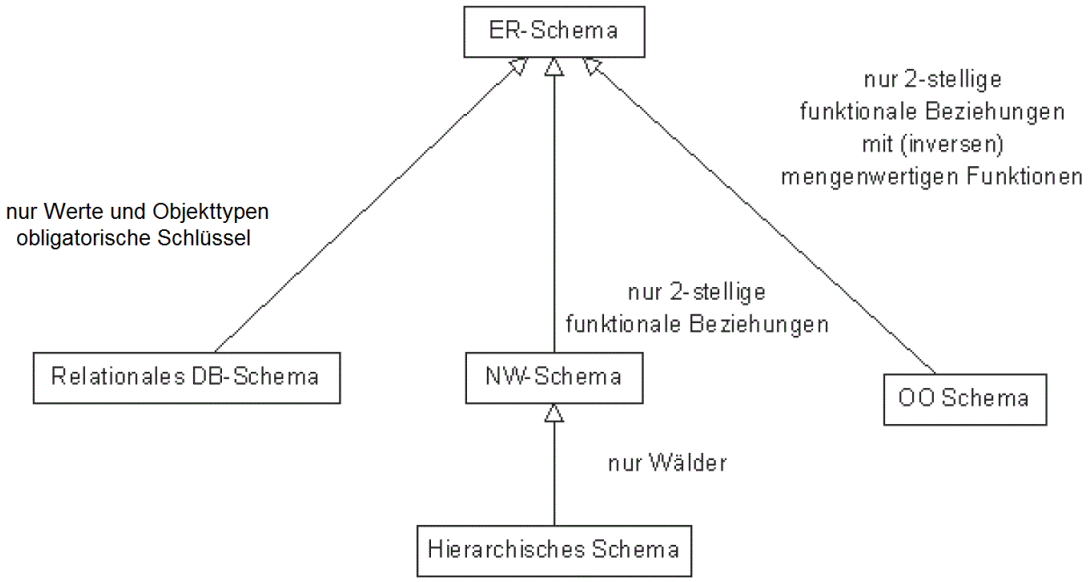
2.6 Semistrukturierte Datenmodelle

Korrespondenzen (ER - XML): `(0,1) = ?` `(1,*) = +` `(0,*) = *`



ER-Schema gibt Reihenfolge nicht wieder; Reihenfolge in XML präsent

2.7 Vergleich der Datenmodelle



Delta: ER versus OO - Verhaltenbeschreibungen in OO