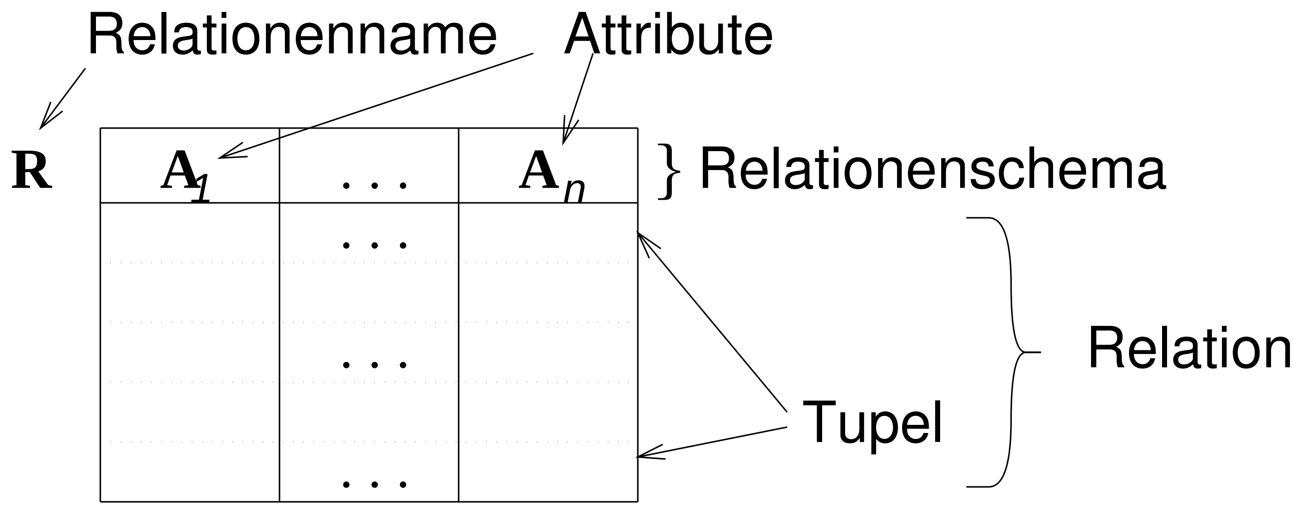
4. Datenbankmodelle für die Realisierung

- ➠ Relationenmodell
- ➠ Netzwerkmodell und hierarchisches Modell
- ➠ Objektorientierte Modelle
- ➠ Weitere Datenbankmodelle

Relationenmodell

Codd im Jahre 1970



Veranschaulichung eines Relationenschemas und einer Relation

Zwei Relationen zur Darstellung von `Personen`

Personen

PANr	Vorname	Nachname	PLZ	Ort	Straße	HNr	Geb.datum
4711	Andreas	Heuer	18209	DBR	BHS	15	31.10.1958
5588	Gunter	Saake	39106	MD	STS	55	05.10.1960
6834	Michael	Korn	39104	MD	BS	41	24.09.1974
7754	Andreas	Möller	18209	DBR	RS	31	25.02.1976
8832	Tamara	Jagellovsk	38106	BS	GS	12	11.11.1973
9912	Antje	Hellhof	18059	HRO	AES	21	04.04.1970
9999	Christa	Loeser	69121	HD	TS	38	10.05.1969

Pers_Telefon

PANr	Telefon
4711	038203-12230
4711	0381-498-3401
4711	0381-498-3427
5588	0391-345677
5588	0391-5592-3800
9999	06221-400177

Begriffe des Relationenmodells

Begriff	**Informale Bedeutung**
Attribut	Spalte einer Tabelle
Wertebereich	mögliche Werte eines Attributs (auch Domäne)
Attributwert	Element eines Wertebereichs
Relationenschema	Menge von Attributen
Relation	Menge von Zeilen einer Tabelle
Tupel	Zeile einer Tabelle
Datenbankschema	Menge von Relationenschemata
Datenbank	Menge von Relationen (Basisrelationen)

Begriffe des Relationenmodells II

Begriff	Informale Bedeutung
Schlüssel	minimale Menge von Attributen, deren Werte ein Tupel einer Tabelle eindeutig identifizieren
Primärschlüssel	ein beim Datenbankentwurf ausgezeichneter Schlüssel
Fremdschlüssel	Attributmenge, die in einer anderen Relation Schlüssel ist
Fremdschlüsselbedingung	alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf

Formalisierung Relationenmodell I

Attribute und Domänen

- $\mathcal{U}$ nichtleere, endliche Menge: *Universum*
- $A \in \mathcal{U}$: *Attribut*
- $\mathcal{D} = \{D_1, \ldots, D_m\}$ Menge endlicher, nichtleerer Mengen: jedes D_i: *Wertebereich* oder *Domäne*
- total definierte Funktion dom : $\mathcal{U} \longrightarrow \mathcal{D}$
- dom(A): *Domäne von* A
  $w \in$ dom(A): *Attributwert* für A

Formalisierung Relationenmodell II

Relationenschemata und Relationen

- $R \subseteq \mathcal{U}$: *Relationenschema*
- *Relation* r *über* $R = \{A_1, \ldots, A_n\}$ (kurz: $r(R)$) ist endliche Menge von Abbildungen $t : R \longrightarrow \bigcup_{i=1}^{m} D_i$, *Tupel* genannt
- Es gilt $t(A) \in \mathrm{dom}(A)$ ($t(A)$ Restriktion von t auf $A \in R$)
- für $X \subseteq R$ analog $t(X)$ *X-Wert* von t Menge aller Relationen über R: **REL**$(R) := \{r \mid r(R)\}$

Formalisierung Relationenmodell III

Datenbankschema und Datenbank

- Menge von Relationenschemata $S := \{R_1, \ldots, R_p\}$: *Datenbankschema*
- *Datenbank* über S: Menge von Relationen $d := \{r_1, \ldots, r_p\}$, wobei $r_i(R_i)$
- Datenbank d über S: $d(S)$
- Relation $r \in d$: *Basisrelation*

Unterschied zu der klassischen Definition einer Relation als Teilmenge des kartesischen Produktes

$$r_1 \subseteq \text{dom}(\texttt{PANr}) \times \text{dom}(\texttt{Vorname}) \times \text{dom}(\texttt{Nachname})$$

und

$$r_2 \subseteq \text{dom}(\texttt{PANr}) \times \text{dom}(\texttt{Nachname}) \times \text{dom}(\texttt{Vorname})$$

sind ungleich bei Definition mittels kartesischem Produkt!

Unterschied zu der klassischen Definition einer Relation als Teilmenge des kartesischen Produktes: Beispiel

r_1

PANr	Vorname	Nachname
4711	Andreas	Heuer
5588	Gunter	Saake
6834	Michael	Korn

r_2

PANr	Nachname	Vorname
4711	Heuer	Andreas
5588	Saake	Gunter
6834	Korn	Michael

Relationen r_1 und r_2 bestehen aus Tupeln t_1, t_2, t_3 mit

t_1(PANr)=4711, t_1(Vorname)='Andreas', t_1(Nachname)='Heuer'
t_2(PANr)=5588, t_2(Vorname)='Gunter', t_2(Nachname)='Saake'
t_3(PANr)=6834, t_3(Vorname)='Michael', t_3(Nachname)='Korn'

Integritätsbedingungen

Identifizierende Attributmenge $K := \{B_1, \ldots, B_k\} \subseteq R$:

$$\forall t_1, t_2 \in r \; [t_1 \neq t_2 \quad \Longrightarrow \quad \exists B \in K : t_1(B) \neq t_2(B)].$$

- **Schlüssel**: ist minimale identifizierende Attributmenge
  $\{$`Vorname, Nachname, PLZ, Geburtsdatum`$\}$ und
  $\{$`PANr`$\}$ für `Personen`
  $\{$`PANr, Telefon`$\}$ für `Pers_Telefon`
- **Primattribut**: Element eines Schlüssels
- **Primärschlüssel**: ausgezeichneter Schlüssel
- **Fremdschlüssel**: $\leadsto \Box\ \Box$

Relationenalgebra

- Selektion

$$\sigma_{\texttt{Nachname}=\text{'Meyer'}}(r(\texttt{Personen})).$$

$$\sigma_{\texttt{Nachname}=\texttt{Vorname}}(r(\texttt{Personen})).$$

- Projektion

$$\pi_{\texttt{Vorname, PLZ}}(r(\texttt{Personen}))$$

Relationenalgebra II

- Verbund

$$\pi_{\texttt{Vorname,Nachname,Ort,Straße,Telefon}}(r(\texttt{Personen}) \bowtie r(\texttt{Pers\_Telefon}))$$

Vorname	Nachname	Ort	Straße	Telefon
Andreas	Heuer	DBR	BHS	038203-12230
Andreas	Heuer	DBR	BHS	0381-498-3401
Andreas	Heuer	DBR	BHS	0381-498-3427
Gunter	Saake	MD	STS	0391-345677
Gunter	Saake	MD	STS	0391-5592-3800
Christa	Loeser	HD	TS	06221-400177

Relationenalgebra III

- Mengenoperationen $\cap, \cup, -$
- Umbenennung $\beta_{\texttt{Wohnort} \leftarrow \texttt{Ort}}(r(\texttt{Personen}))$

  Formale Definition: $\leadsto$ □ □

Netzwerkmodell

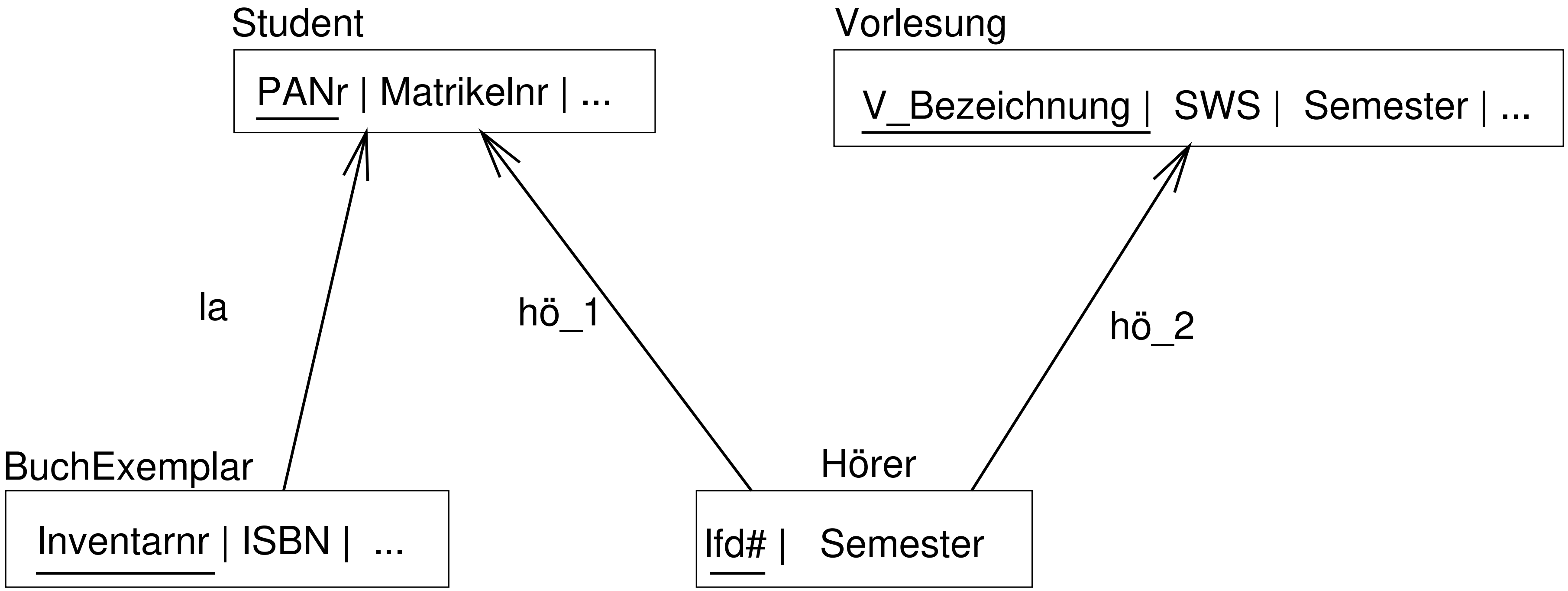
Netzwerkmodell: 1971 vom Normungsausschuß CODASYL-DBTG definiert

ER-Modell	**Relationenmodell**	**Netzwerkmodell**
Entity	Tupel	logical record
Entity-Typ	Relationenschema	Record-Typ
Attribut	Attribut	Feld
binärer 1:n-Beziehungstyp	Relation	Link oder auch *Set-Typ*

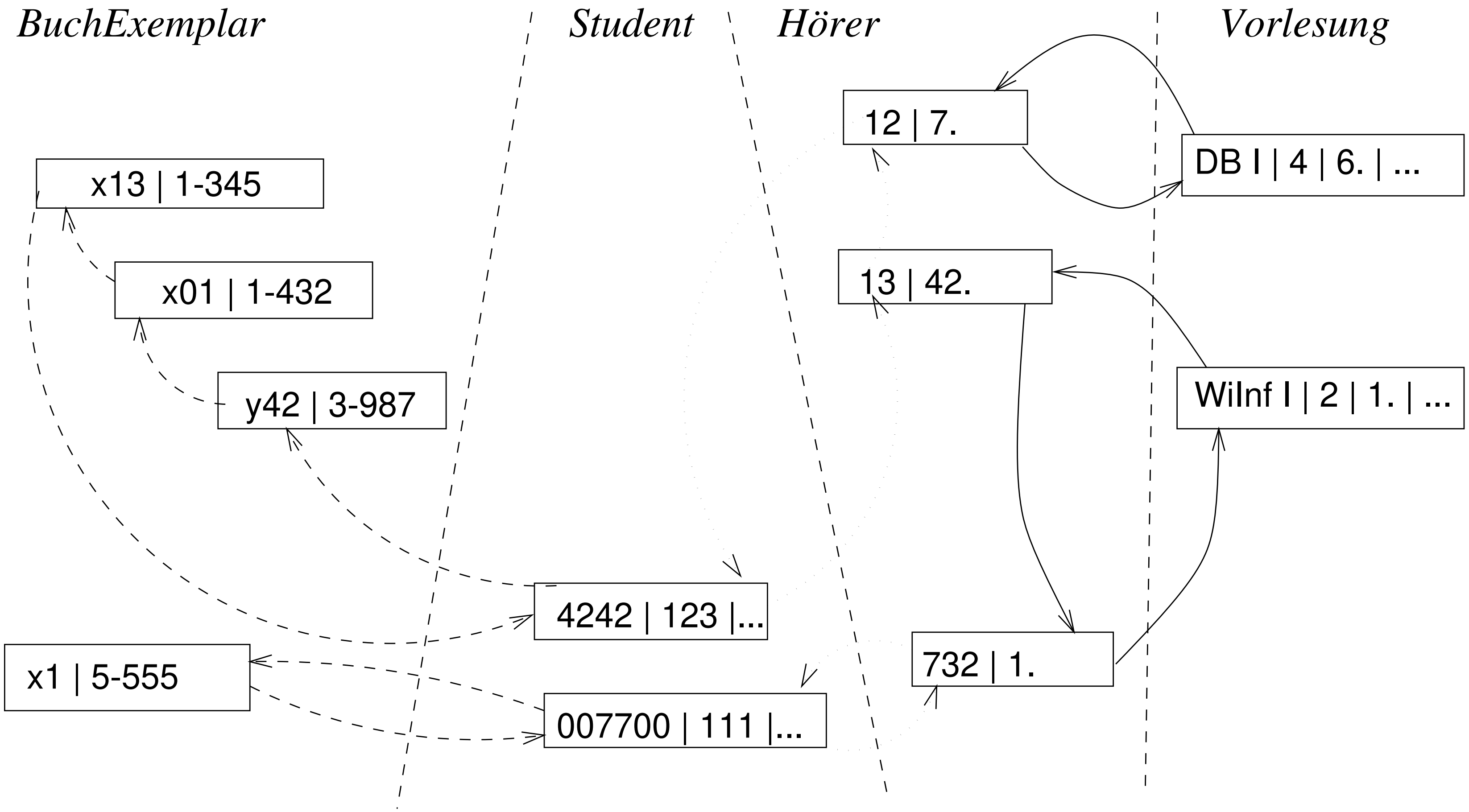
Schema im Netzwerkmodell

- *Netzwerkschema*: gerichteter Graph
  - Menge der Record-Typen als Knoten
  - Set-Typen als Kanten ((E_1, E_2) Kante, falls E_1 und E_2 in $n : 1$-Relationship)
- konkrete *Set-Ausprägung*: *Besitzer* (*Owner*) und *Teilnehmer* (*Members*)
- Netzwerkmodell entspricht ER-Modell mit Einschränkungen
  - alle Relationships binär
  - nur many-to-one Relationships erlaubt
  - Relationships haben keine Attribute
- Grund: leichtere Graphendarstellung, günstigere Implementierung

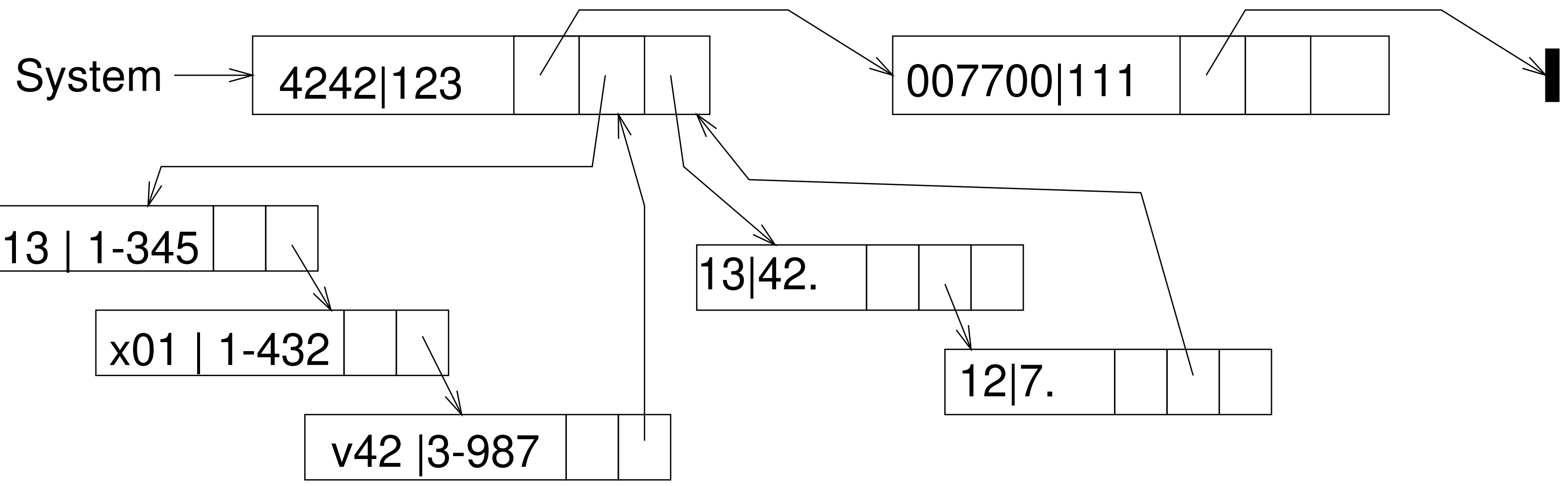
Beispielschema im NWM



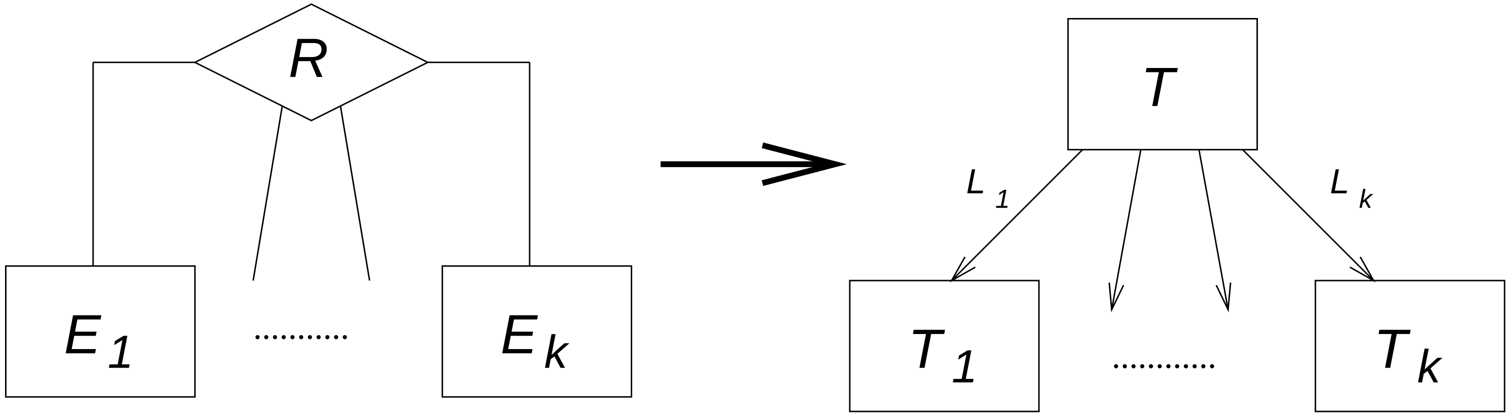
Beispielausprägung im Netzwerkmodell



Detaildarstellung einer Beispielausprägung



Simulation einer allgemeinen Relationship



Kett-Entity T

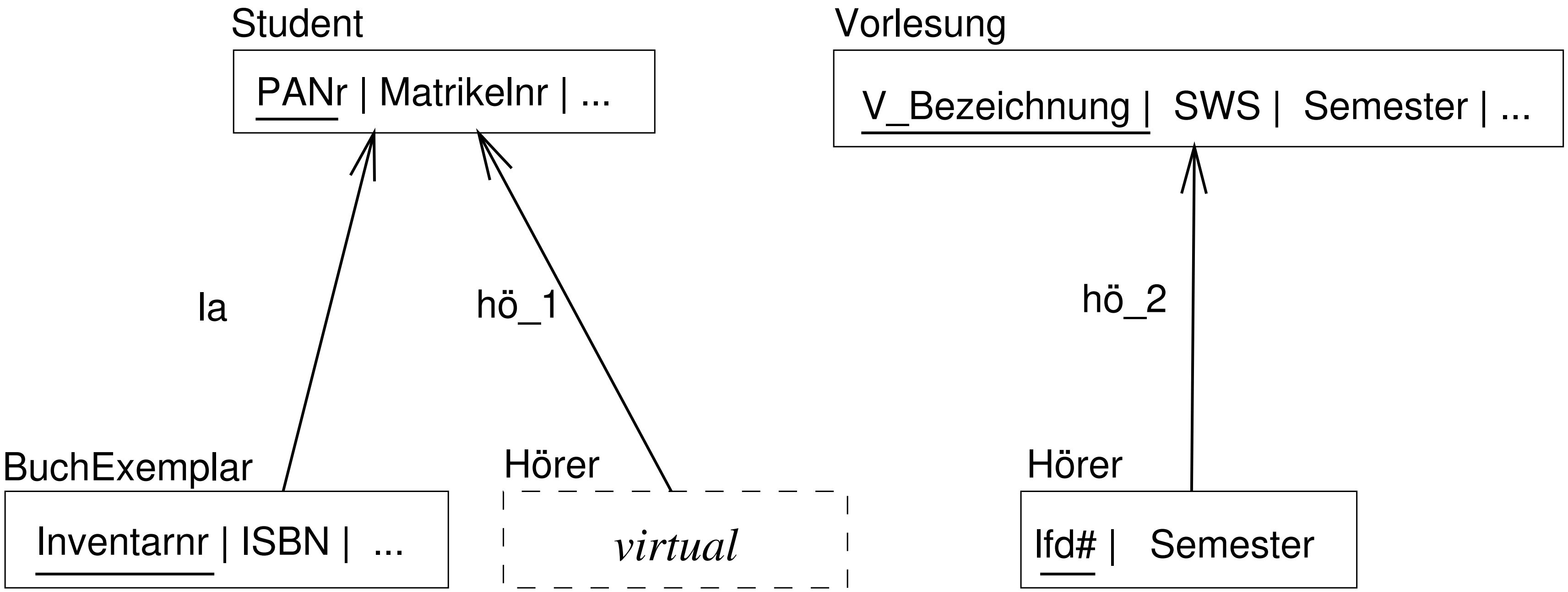
Hierarchisches Datenmodell

wie NWM, aber nur Hierarchien

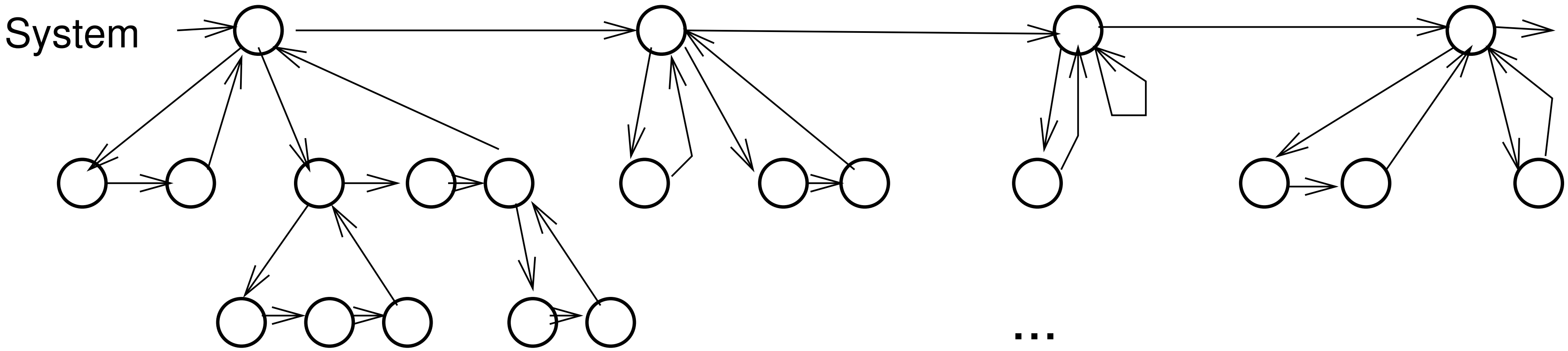
Einführung: IBM 1969 mit System IMS

- Eine *Hierarchie* ist ein Netzwerkschema, das ein Wald ist ('Menge von Bäumen').
- Eine reine Hierarchie kann keine allgemeinen Beziehungen darstellen, so daß sogenannte "virtual records" ('Zeiger') eingeführt werden, um die Baumstruktur zu durchbrechen.

Beispiel im hierarchischen Datenmodell



Skizze der Speicherstrukturen im hierarchischen Datenmodell



Erweiterte relationale und semantische Modelle

Geschachtelte Relationen: Das NF^2-Modell

> *NF^2-Relationen erlauben komplexe Attributwerte in dem Sinne, daß Attribute selbst wieder Relationen sein können.*

Beispiel im NF^2-Modell

Fachbereich	Belegschaft			
	PANr	Nachname	Telefone	Gehalt
			Telefon	
Informatik	4711	Heuer	038203-12230 0381-498-3401 0381-498-3427	6000
	5588	Saake	0391-345677 0391-5592-3800	6000
	7754	Möller		550
	8832	Jagellovsk		2800
Mathematik	6834	Korn		750

PNF-Relationen

PNF: *Partitioned Normal Form*

Relationen in PNF haben auf jeder Stufe der Schachtelung einen flachen Schlüssel.

PNF-Relation:

A	D	
	B	C
1	2	3
	4	2
2	1	1
	4	1
3	1	1

Nicht PNF:

A	C
B	D
1	2
2	3
1	3
2	4

PNF-Relation und entschachtelte äquivalente Relation

A	D	
	B	C
1	2	3
	4	2
2	1	1
	4	1
3	1	1

A	B	C
1	2	3
1	4	2
2	1	1
2	4	1
3	1	1

Verallgemeinerte geschachtelte Relationen

Erweitertes NF^2-Modell: eNF^2-Modell

Beliebige Kombination der Typkonstruktoren

- **set of**: Mengenbildung
- **tuple of**: Tupelkonstruktion
- **list of**: Listen
- **bag of**: Multimengen

Typkonstruktoren können beliebig kombiniert werden, so daß etwa **set of bag of integer** erlaubt ist.

Semantische Datenmodelle

Semantische Datenmodelle unterstützen weitere Abstraktionskonzepte wie Generalisierung und Spezialisierung

- *Funktionale Datenmodelle*: Alle Anwendungsobjekte werden mit Entity-Typen und Funktionen modelliert.
- *IFO-Modell*: IFO steht für IST-Beziehungen, Funktionen und Objekttypen.
- *SDM*: viele, teilweise auch redundante Konzepte zur Modellierung

Objektorientierte Modelle inkl. ODMG

Objektorientierte Datenbankmodelle bieten

- mehr Konzepte zur Darstellung der Struktur
  - *komplexe Werte*, die mit Typkonstruktoren wie **set of**, **tuple of** und **list of** beschrieben werden können,
  - *Objektidentität*, die gespeicherte Objekte von Werten, die sie besitzen, unterscheiden kann,
  - *Vererbung* von Attributen zwischen Objekttypen, die in einer IST-Beziehung stehen, sowie
- mehr Konzepte zur Darstellung objektspezifischer Operationen, etwa *Methoden* (legen Operationen fest, mit denen die Anwendungsdaten (nur) manipuliert werden dürfen)

  Beispiel: Studenten $\leadsto$ □ □

Modell nach Beeri

- Strukturteil
  - Typen und Typkonstruktoren
  - Objektidentität
  - Klassen
  - Strukturvererbung (oder Klassen- und Typhierarchie)
- Operationenteil
  - Anfrageoperationen
  - Änderungsoperationen

Modell nach Beeri II

- Höhere Konzepte
  - Metaklassen
  - Methoden
  - Vererbung und Overriding von Methoden
  - Einkapselung

Definition eines objektorientierten Datenbanksystems

Datenbanksystem, das

- auf einem objektorientierten Datenbankmodell mit Strukturteil, Operationenteil und höheren Konzepten basiert,
- auf der konzeptuellen Ebene durch neue Datentypen und neue Funktionen erweiterbar ist,
- weitere Datenbank-Eigenschaften besitzt (wie Persistenz, Speicherungsstrukturen und Zugriffspfade, Transaktionen und Concurrency-Control-Komponenten sowie Recovery-Mechanismen)
- und neben den Operationen des Operationenteils (Anfrage- und Datenmanipulationssprache) auch eine komplette Programmier-Umgebung beinhaltet.

Klassifikation von OODBS

Systeme (seit 1987, Manifesto 1989, ODMG-Industrie-Standard 1993)

- Erweiterung objektorientierter Programmiersprachen (OOPLs)
  - C++- oder SMALLTALK-Datenmodell (etwa GemStone, ObjectStore, POET, ...)
- Erweiterung relationaler Datenbanksysteme
  - Relationales Datenmodell $+$ Typkonstruktoren $+$ Objektidentität $+ \ldots$ (etwa DASDBS, AIM/P, POSTGRES, ...)
  - speziell: Objekt-relationale Datenbanksysteme (etwa Illustra, UniSQL, jetzt auch viele RDBS wie DB2)
- Neuentwicklungen
  - eigenes OO Datenmodell (etwa O_2, Itasca, OSCAR)

Strukturteil

- Typen und Typkonstruktoren
  - Standard-Datentypen wie INTEGER und STRING
  - Typkonstruktoren wie SET OF und TUPLE OF: kompliziertere Typen
- Objektidentität
  - vom System vergeben
  - eindeutig
  - unveränderbar
  - für den Benutzer unsichtbar

Strukturteil II

- Klassen
  - beschreiben Objekte mit ähnlichen Eigenschaften
  - Typ, Objektvorrat und Objektbehälter
  - Methoden
- Komponenten-Beziehungen bei Klassen (VERLAGE Komponente von BÜCHERN)

Strukturteil III

- Is-A-Beziehungen
  - *Klassenhierarchie*: Objektmenge der Unterklasse ist Teilmenge der Objektmenge der Oberklasse

    (STUDENTEN sind eine Teilmenge der PERSONEN)
  - *Typhierachie*: Typ der Unterklasse hat mehr Eigenschaften als Typ der Oberklasse

    (STUDENTEN haben neben den Eigenschaften von Personen auch noch MATRIKELNUMMER und STUDIENFACH)

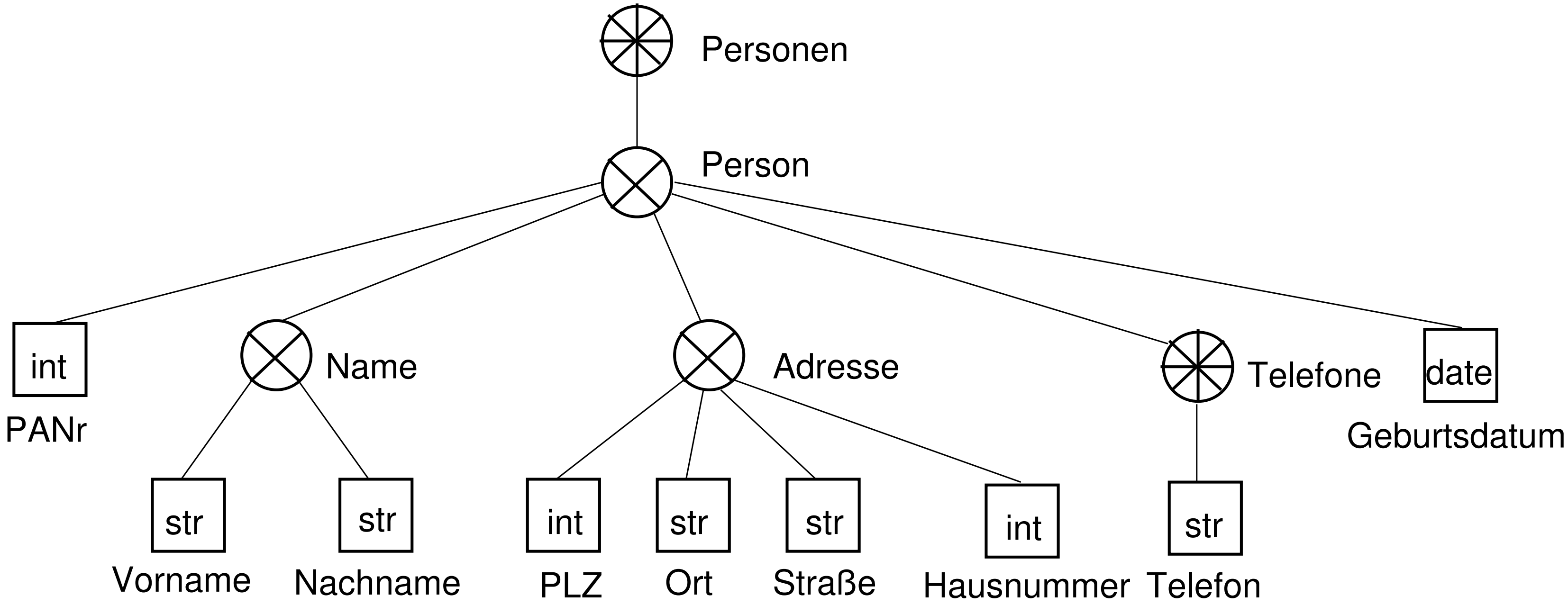
Definition eines Objekttyps

```
set of(tuple of(PANr: integer,
                Name: tuple of(Vorname: string,
                               Nachname: string),
                Adresse: tuple of(PLZ: integer,
                                  Ort: string,
                                  Straße: string,
                                  Hausnummer: integer),
                Telefone: set of(Telefon: string),
                Geburtsdatum: date))
```

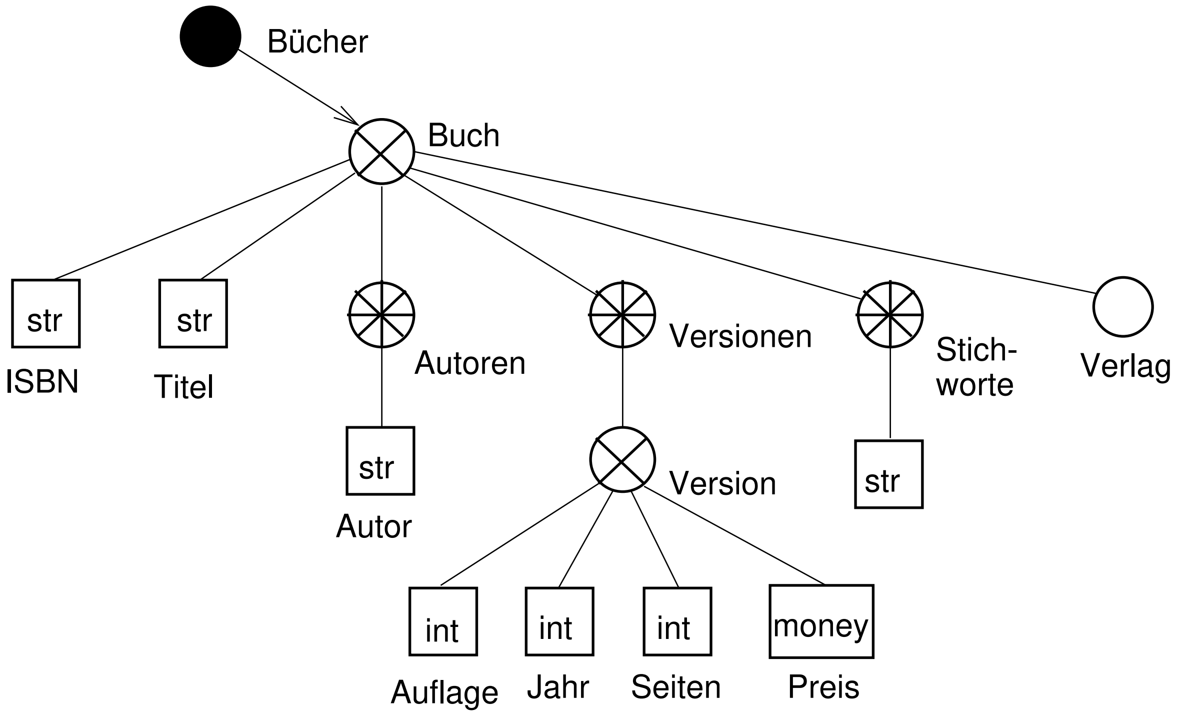
Symbole der graphischen Notation

abstrakte Klasse	Typkonstruktion
freie Klasse	Zustandsfunktion
Tupelkonstruktion	Is-A Beziehung
Mengenkonstruktion	Subtyp-Beziehung
Standard-Datentyp x	

Graphische Definition eines Objekttyps



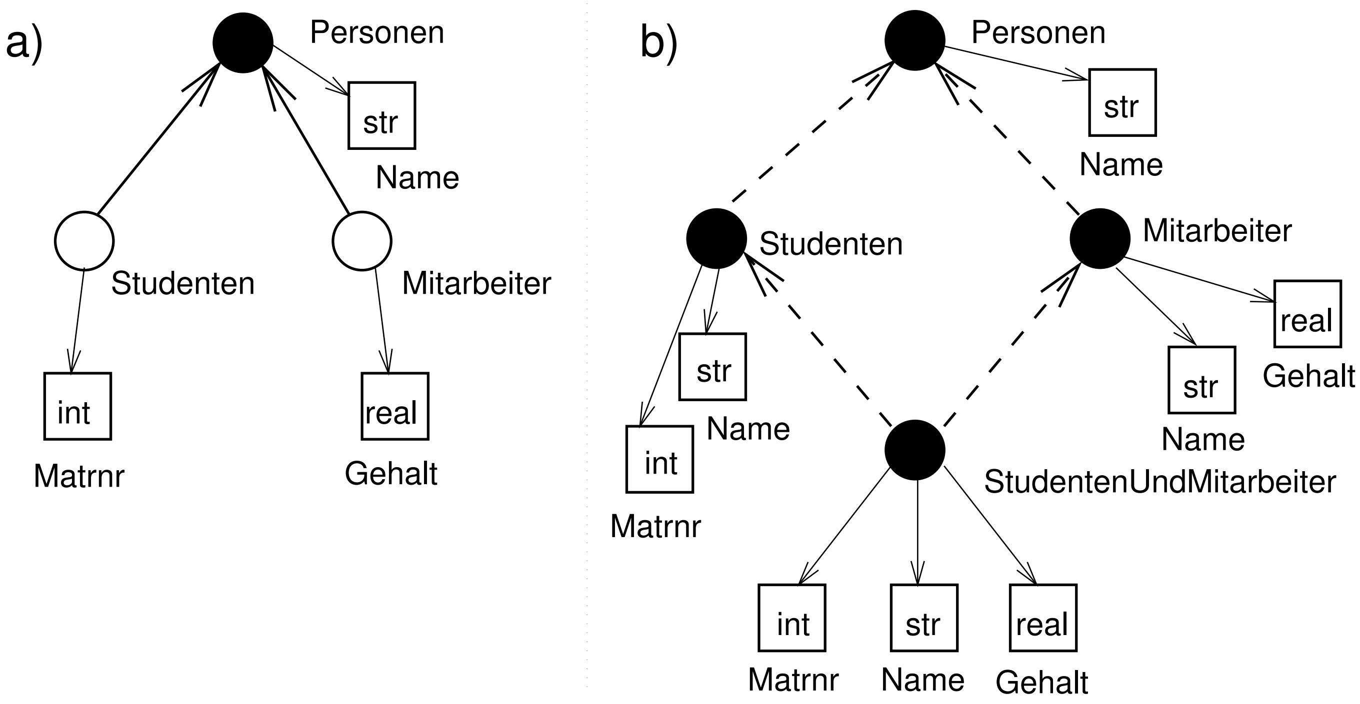
Graphische Deklaration einer Klasse



Beispiel Objektrelation

Bücher	ISBN	Titel	Verlag	Autoren	Stichworte	...
				Autor	Stichwort	
α_1	3-89319-175-5	DB2	β_1	Vossen Witt	RDB	...
α_2	0-8053-1753-3	Princ. of DBS	β_2	Elmasri Navathe	RDB Lehrbuch ER	... ...
...	...	...	...	...	...	...

Klassenhierarchie a) versus Typhierarchie b)



Klassendeklarationen im O_2-Modell I

```
class Personen
    type tuple(PANr: integer,
               Name: tuple(Vorname: string,
                           Nachname: string),
               Adresse: tuple(PLZ: integer,
                              Ort: string,
                              Straße: string,
                              Hausnummer: integer),
               Telefone: set(Telefon: string),
               Geburtsdatum: date)
```

Klassendeklarationen im O_2-Modell II

```
class Studenten inherits Personen
      type tuple(Matrikelnummer: integer,
                 Studienfach: string,
                 Vater: Personen,
                 Mutter: Personen,
                 Zeugnis: set(tuple(Fach: string,
                                    Note: real)))
```

Operationen

- mindestens die Möglichkeiten wie in Relationenalgebra / SQL
- *relationale Semantik:* Extraktion von Werten aus Zuständen von Objekten
  $\leadsto$ geschachtelte Relationen
- *objekterzeugende Semantik:* Erzeugung neuer Objekte als Anfrageergebnis mit Zuständen, die von vorhandenen Objekten extrahiert wurden
  $\leadsto$ Ergebnis ist eine dynamisch erzeugte Klasse
- *objekterhaltende Semantik:* Auswahl der in der Datenbank vorkommenden Objekte mit neuen Zuständen
  $\leadsto$ Ergebnis ist dynamisch erzeugte Ober- / Unterklasse

Operationen II

schwach ausgeprägt bei OOPL-Erweiterungen

- Standard-Methoden auf `COLLECTION`-Klassen
  (Selektionen mit sehr einfachen Selektionsprädikaten)
- “OSQL” mit relationaler Semantik (nicht so mächtig wie Standard-SQL)

Höhere Konzepte

- objekt- oder klassenspezifische Operationen
- werden wie Eigenschaften von Ober- zu Unterklassen vererbt
- Implementierung einer Methode kann bei Vererbung noch verändert werden (Overriding)
- System wählt selbständig zur Laufzeit passende Implementierung (dynamisches Binden)

Klassendeklarationen im O_2-Modell III

```
class Studenten inherits Personen
    type tuple(Matrikelnummer: integer,
               Studienfach: string,
               Vater: Personen,
               Mutter: Personen,
               Zeugnis: set(tuple(Fach: string,
                                  Note: real)))
    method Zur_Verfügung: money
```

Methodendeklaration im O_2-Modell

method body `Zur Verfügung`: *real* **in class** `Studenten`

```
{ return ( self → Vater → Zur_Verfügung
            + self → Mutter → Zur_Verfügung)
            ∗ 0.1 }
```

Der ODMG-Standard

Die Struktur des Standards ist viergeteilt:

- *Objektmodell* beschreibt Begriffe und semantische Festlegungen des OO Datenmodells (stark C++-lastig)
- *Datenbanksprachen* ODL (Object Definition Language) und OQL (Object Query Language): mögliche Schnittstelle zur Datendefinition und -manipulation
- *Spracheinbettungen* (oder *Bindings*) für C++, Java und SMALLTALK
- Bezug zur OMG, zu CORBA und zur ANSI-C++-Version

Beispiele zu ODL und OQL folgen in späteren Kapiteln.

Einordnung sonstiger Datenbankmodelle

- *GOOD* (Graph-Oriented Object Database Model): Ecken eines Graphen als Werte, Objekte und Typen
- *Klassenlose Datenmodelle*: Statt Klassen Objekte als *Prototypen*
- Wissensrepräsentation / *Feature-Terme*: Feature-Terme in einer Subsumptionshierarchie
  - Lilog-Datenmodell, *F-Logic*
- *Komplex-Objekt-Datenmodelle* im Ingenieurbereich: Unterstützung von hierarchischen *Ist-Teil-Von-Beziehungen*
  - Molekül-Atom-Datenmodell
  - *STEP-Modell* mit der Datenbeschreibungssprache *EXPRESS*