Datenbanksysteme 2011

Kapitel 16:
Objektorientierte Datenbanken

Vorlesung vom 05.07.2011

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Schwächen relationaler Systeme

Buch: {[ISBN, Verlag, Titel, Autor, Version, Stichwort]}

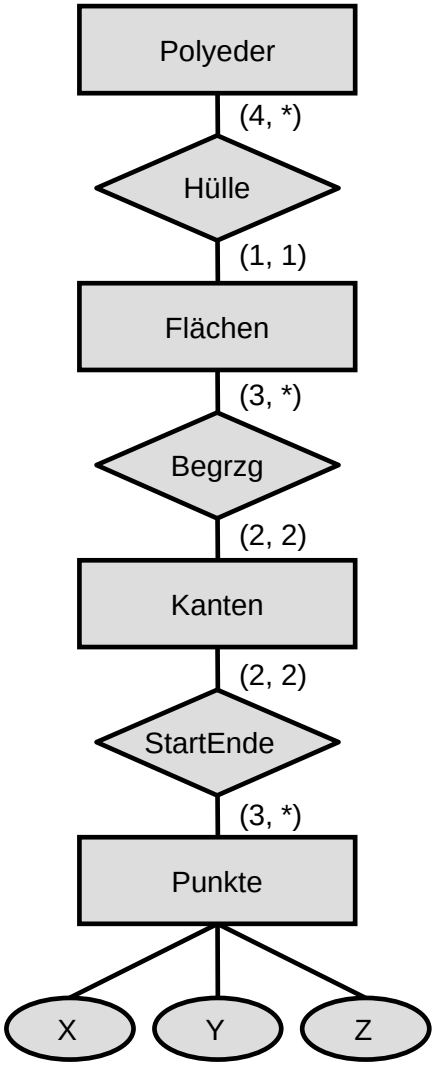
2 Autoren, 5 Versionen, 6 Stichworte

$\Rightarrow 2 \times 5 \times 6 = 60$ Einträge

Buch	:	{[ISBN, Titel, Verlag]}
Autor	:	{[ISBN, Name, Vorname]}
Version	:	{[ISBN, Auflage, Jahr]}
Stichwort	:	{[ISBN, Stichwort]}

Problem:
"Liste Bücher mit den Autoren Meier & Schmidt"

Modellierung von Polyedern



Polyeder

PolyID	Gewicht	Material	. . .
cubo#5	25.765	Eisen	. . .
tetra#7	37.985	Glas	. . .
. . .	. . .	. . .	. . .

Flächen

FlächenID	PolyID	Material
f1	cubo#5	. . .
f2	cubo#5	. . .
. . .	. . .	. . .
f6	cubo#5	. . .
f7	tetra#7	. . .

Kanten

KantenID	F1	F2	P1	P2
k1	f1	f4	p1	p4
k2	f1	f2	p2	p3
. . .	. . .	. . .	. . .	. . .

Punkte

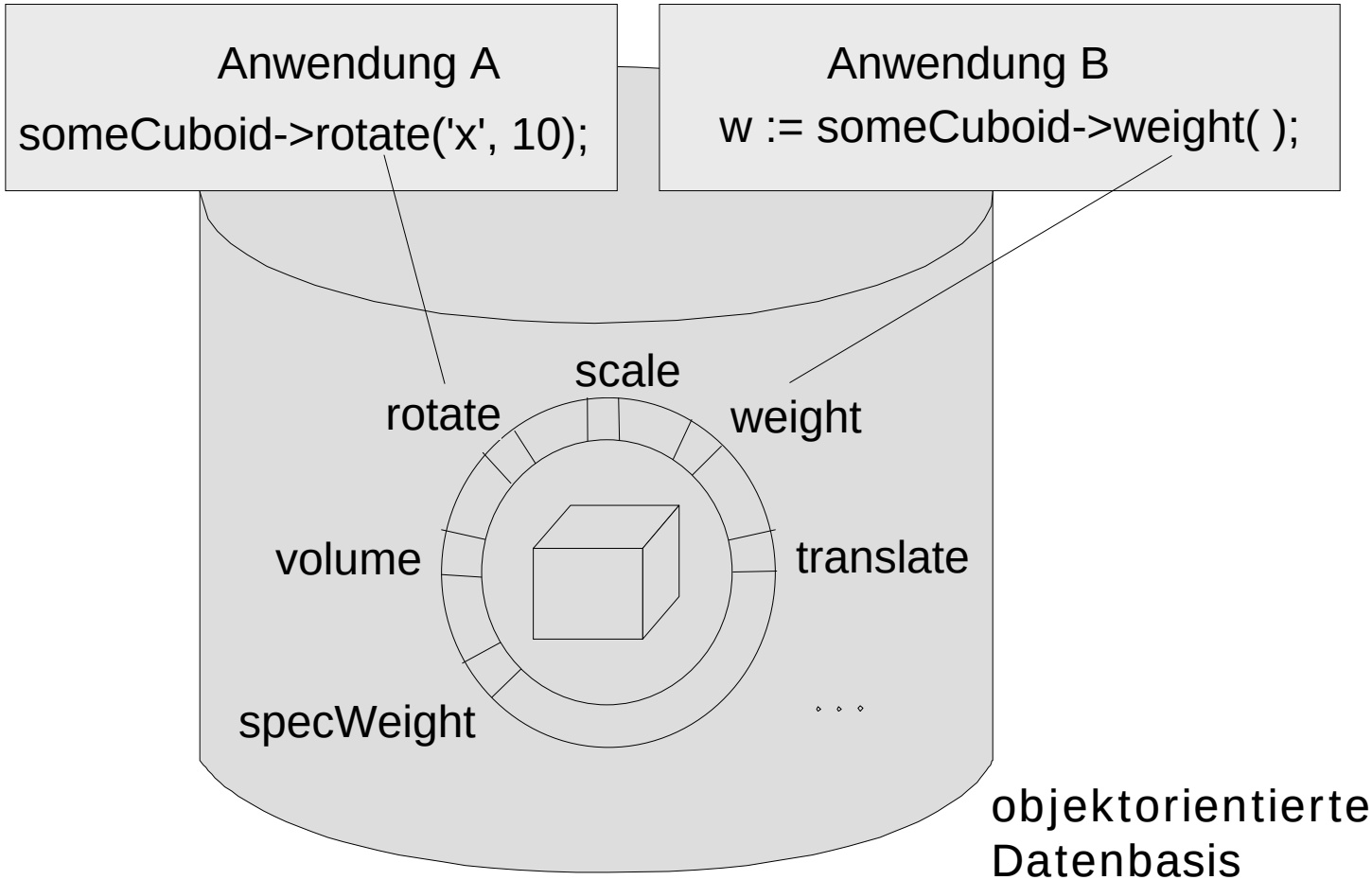
PunktID	X	Y	Z
p1	0.0	0.0	0.0
p2	1.0	0.0	0.0
. . .	. . .	. . .	. . .
p8	0.0	1.0	1.0
. . .	. . .	. . .	. . .

Schwachpunkte

- Segmentierung
- Künstliche Schlüsselattribute
- Fehlendes Verhalten
- Externe Programmierschnittstelle erforderlich

Vorteile der objektorientierten Datenmodellierung

Struktur + Verhalten in einem Objekt-Typ integriert



ODMG

Object Database Management Group
http://www.odbms.org/

entwickelt Standards für

- objektorientiertes Datenmodell
- objektorientierte Abfragesprache OQL
- Schnittstellen zu C++, Smalltalk, Java

Eigenschaften von Objekten

Ein Objekt

- gehört zu einer Klasse
  entsteht durch Instanziierung
  ist Teil der Extension = alle Objekte

- hat Identität
  systemweit eindeutig, unveränderbar

- Wert = Zustand = Ausprägung

Identität

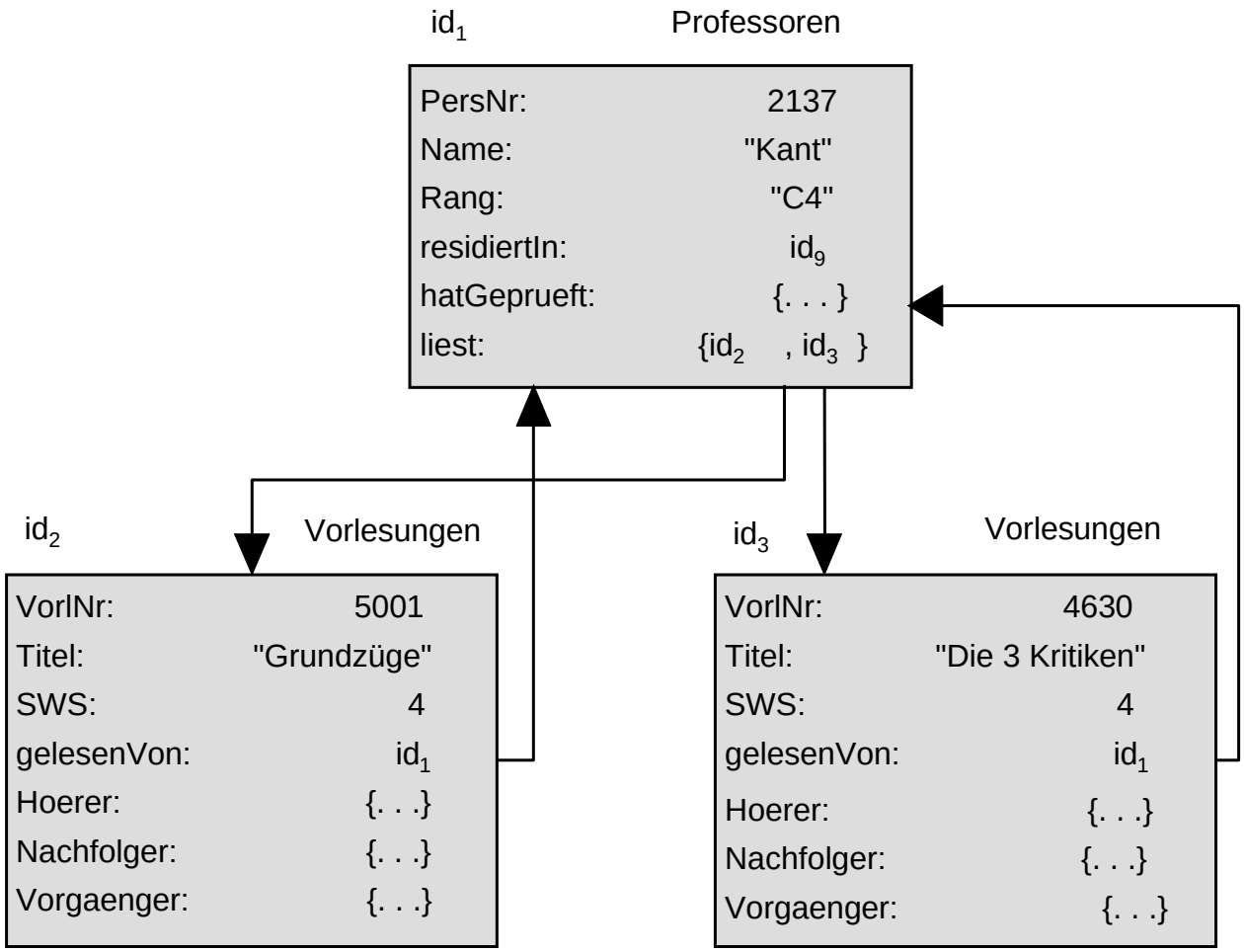
relational:

- Schlüssel
- vom Anwender erdacht

objektorientiert:

- Objektidentifikator OID
- vom System generiert
- unabhängig vom Zustand
- unabhängig vom Speicherort
- transient + persistent
- momentaner Speicherort = Tabelle[OID]

Objekte aus der Universitätswelt



Objekttypen-Definition

- Angaben zur Generalisierung und Spezialisierung
- Strukturbeschreibung
  mit Attributen und Beziehungen
- Verhaltensbeschreibung
  mit Operationen

Attribute

```
class Professoren {
  attribute long   PersNr;
  attribute string Name;
  attribute string Rang;
};
```

strukturierte Attribute

```
class Person {
  attribute string Name;
  attribute struct Datum {
    short Tag;
    short Monat;
    short Jahr;
  } GebDatum;
};
```

1:1 Beziehung

```
class Professoren {
  attribute long  PersNr;
  ...
  relationship Raeume residiertIn
                       inverse Raeume::beherbergt;
};

class Raeume {
  attribute long  RaumNr;
  attribute short Groesse;
  ...
  relationship Professoren beherbergt
                   inverse Professoren::residiertIn;
};
```

1:N-Beziehung

```
class Professoren {
  ...
  relationship set <Vorlesungen> liest
             inverse Vorlesungen::gelesenVon;
};

class Vorlesungen {
  ...
  relationship Professoren gelesenVon
              inverse Professoren::liest;
};
```

Binäre N:M-Beziehung

```
class Studenten {
  ...
  relationship set <Vorlesungen> hoert
            inverse Vorlesungen::Hoerer;
};


class Vorlesungen {
  ...
  relationship set <Studenten> Hoerer
                  inverse Studenten::hoert;
};
```

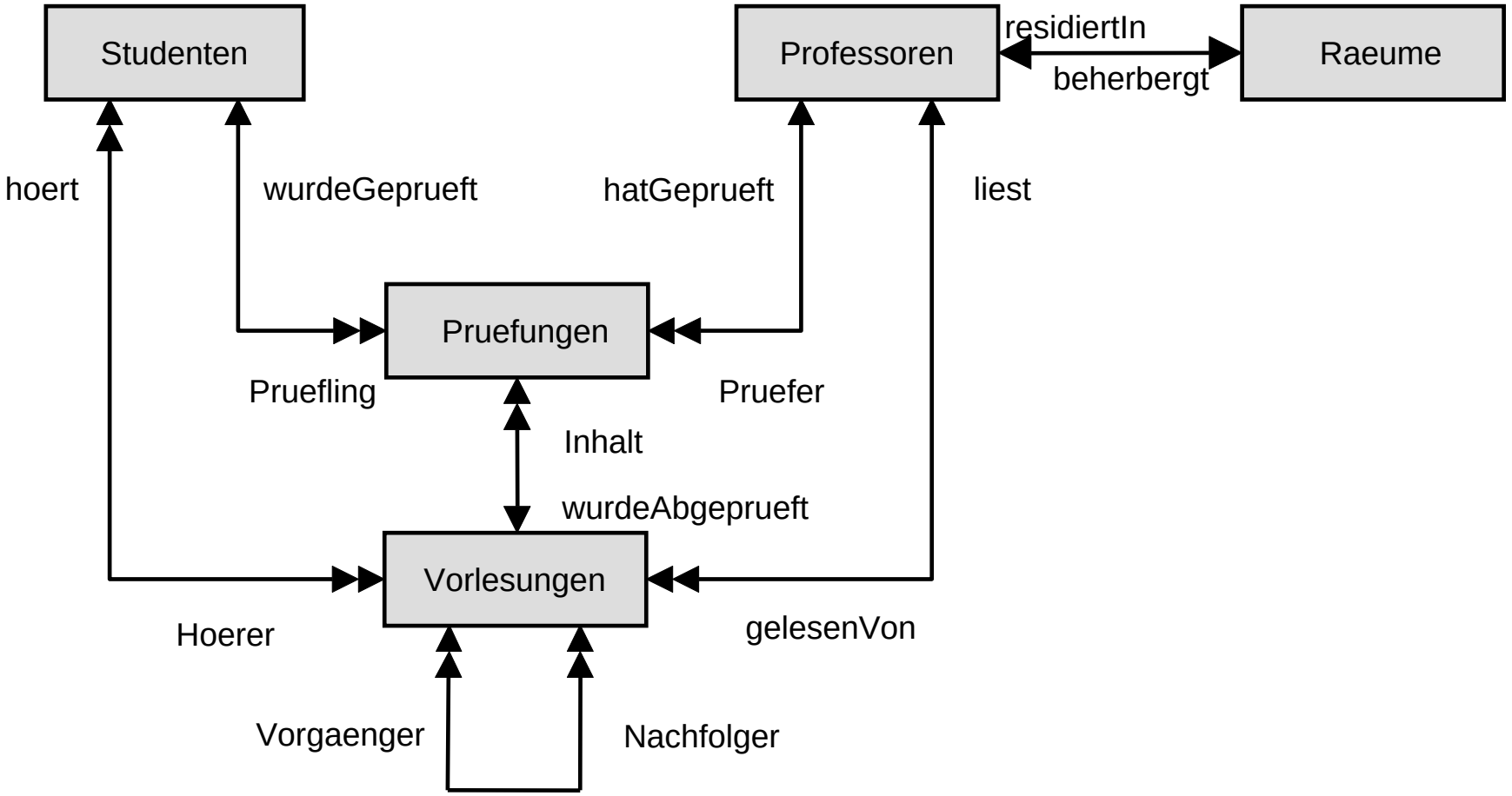
ternäre Beziehungen

Ternäre (oder *n* > *3* stellige) Beziehungen benötigen einen eigenständigen Objekttyp

pruefen : {[MatrNr, VorlNr, PersNr, Note]}

```
class Prüfungen {
  attribute    float       Note;
  relationship Professoren Prüfer
               inverse Professoren::hatGeprueft;
  relationship Studenten   Pruefling
               inverse Studenten::wurdeGeprueft;
  relationship Vorlesungen Inhalt
               inverse Vorlesungen::wurdeAbgeprueft;
};
```

Universität



Universität

```
class Pruefungen {
  attribute    float       Note;
  relationship Professoren Pruefer   inverse Professoren::hatgeprueft;
  relationship Studenten   Pruefling inverse Studenten::wurdegeprueft;
  relationship Vorlesungen Inhalt    inverse Vorlesungen::wurdeAbgeprueft;
};
class Professoren {
  attribute    long             PersNr;
  attribute    string           Name;
  attribute    string           Rang;
  relationship Raeume           residiertIn inverse Raeume::beherbergt;
  relationship set<Vorlesungen> liest       inverse Vorlesungen::gelesenVon;
  relationship set<Pruefungen>  hatgeprueft inverse Pruefungen::Pruefer;
};
class Vorlesungen {
  attribute    long             VorlNr;
  attribute    string           Titel;
  attribute    short            SWS;
  relationship Professoren      gelesenVon inverse Professoren::liest;
  relationship set<Studenten>   Hoerer     inverse Studenten::hoert;
  relationship set<Vorlesungen> Nachfolger inverse Vorlesungen::Vorgaenger;
  relationship set<Vorlesungen> Vorgaenger inverse Vorlesungen::Nachfolger;
  relationship set<Pruefungen>  wurdeAbgeprueft inverse Pruefungen::Inhalt;
};
class Studenten {
  attribute    long             MatrNr;
  attribute    string           Name;
  attribute    short            Semester;
  relationship set<Pruefungen>  wurdeGeprueft inverse Pruefungen::Pruefling;
  relationship set<Vorlesungen> hoert         inverse Vorlesungen::Hoerer;
};
```

Extensionen und Schlüssel

```
class Studenten (extent AlleStudenten key MatrNr) {
  attribute    long              MatrNr;
  attribute    string            Name;
  attribute    short             Semester;

  relationship set<Vorlesungen> hoert
                  inverse Vorlesungen::Hoerer;

  relationship set<Pruefungen>  wurdeGeprueft
                  inverse Pruefungen::Pruefling;
};
```

Operationen

- Objekt erzeugen mit Konstruktor
- Zustand erfragen mit Observer
- Zustand verändern mit Mutator
- Objekt zerstören mit Destruktor

Modellierung des Verhaltens

```
class Professoren {
  exception hatNochNichtGeprueft { };
  exception schonHoechsteStufe   { };
  ...
  float wieHartAlsPruefer() raises (hatNochNichtgeprueft);
  void befoerdert() raises (schonHoechsteStufe);
};
```

in C++:

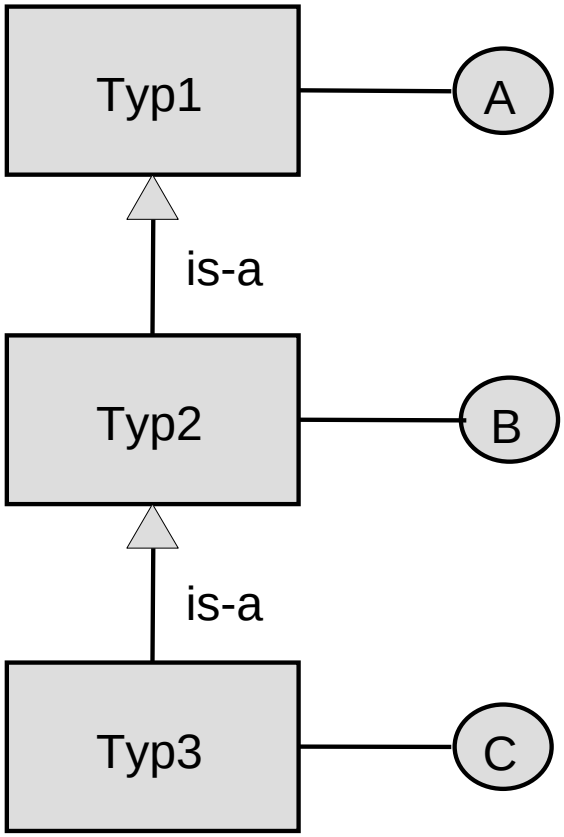
```
meinLieblingsProf->befoerdert();
```

in OQL:

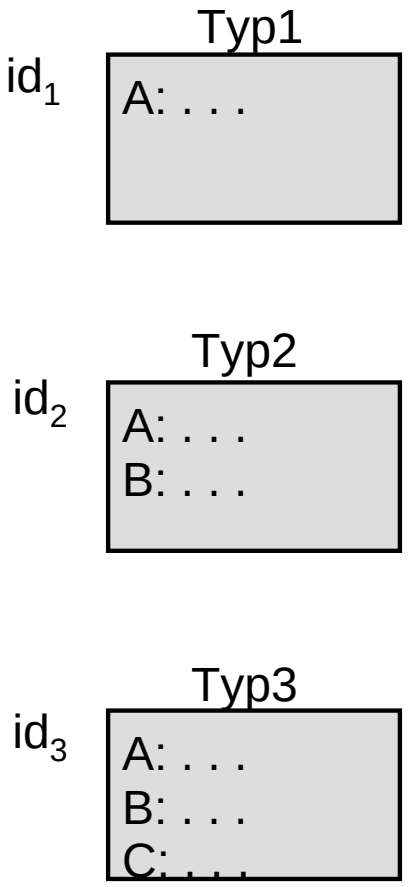
```
select p.wieHartAlsPruefer()
from p in AlleProfessoren
where p.name = "Kant";
```

Vererbung

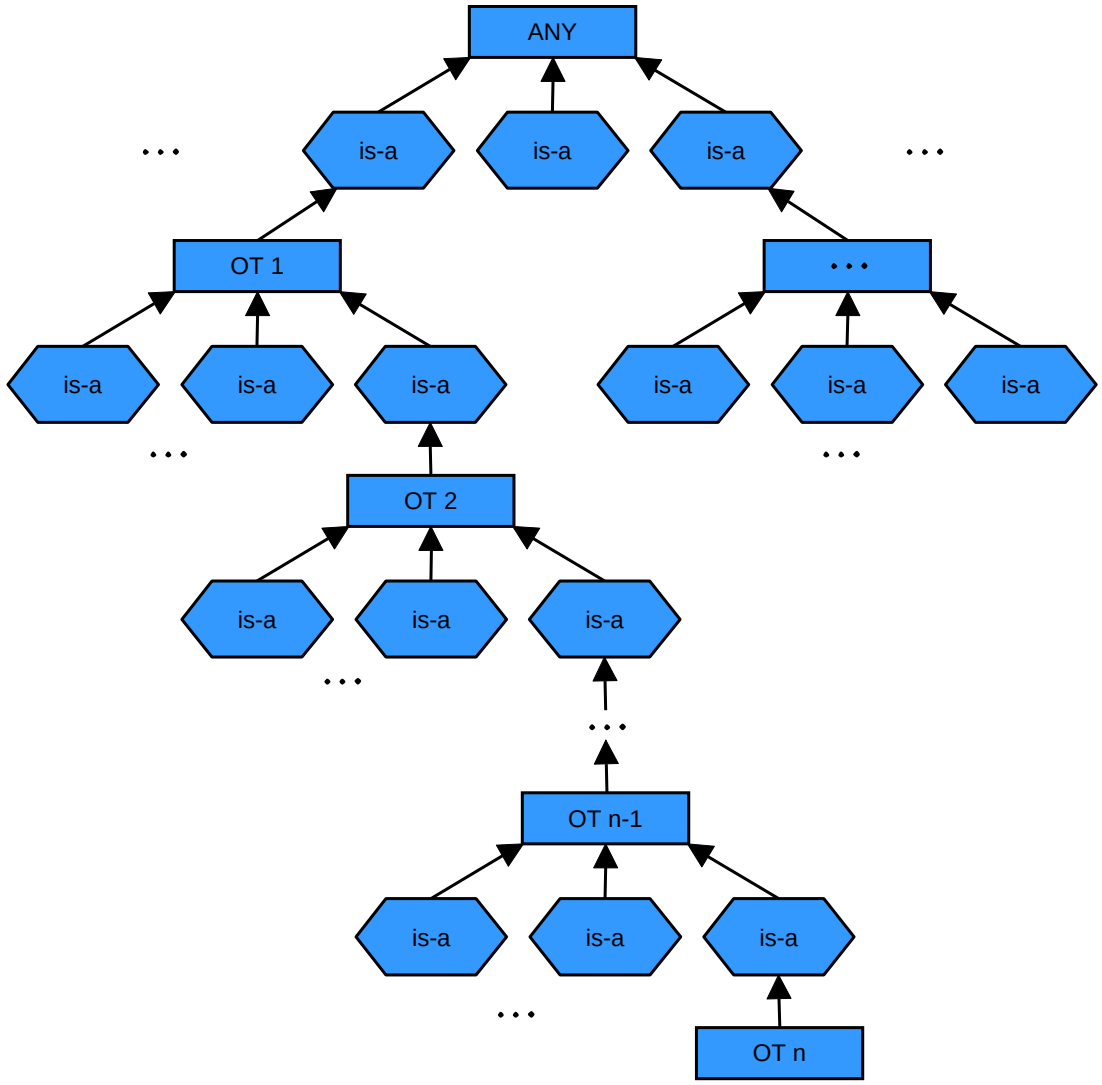
Objekttypen



Instanzen



Typhierarchie bei einfacher Vererbung



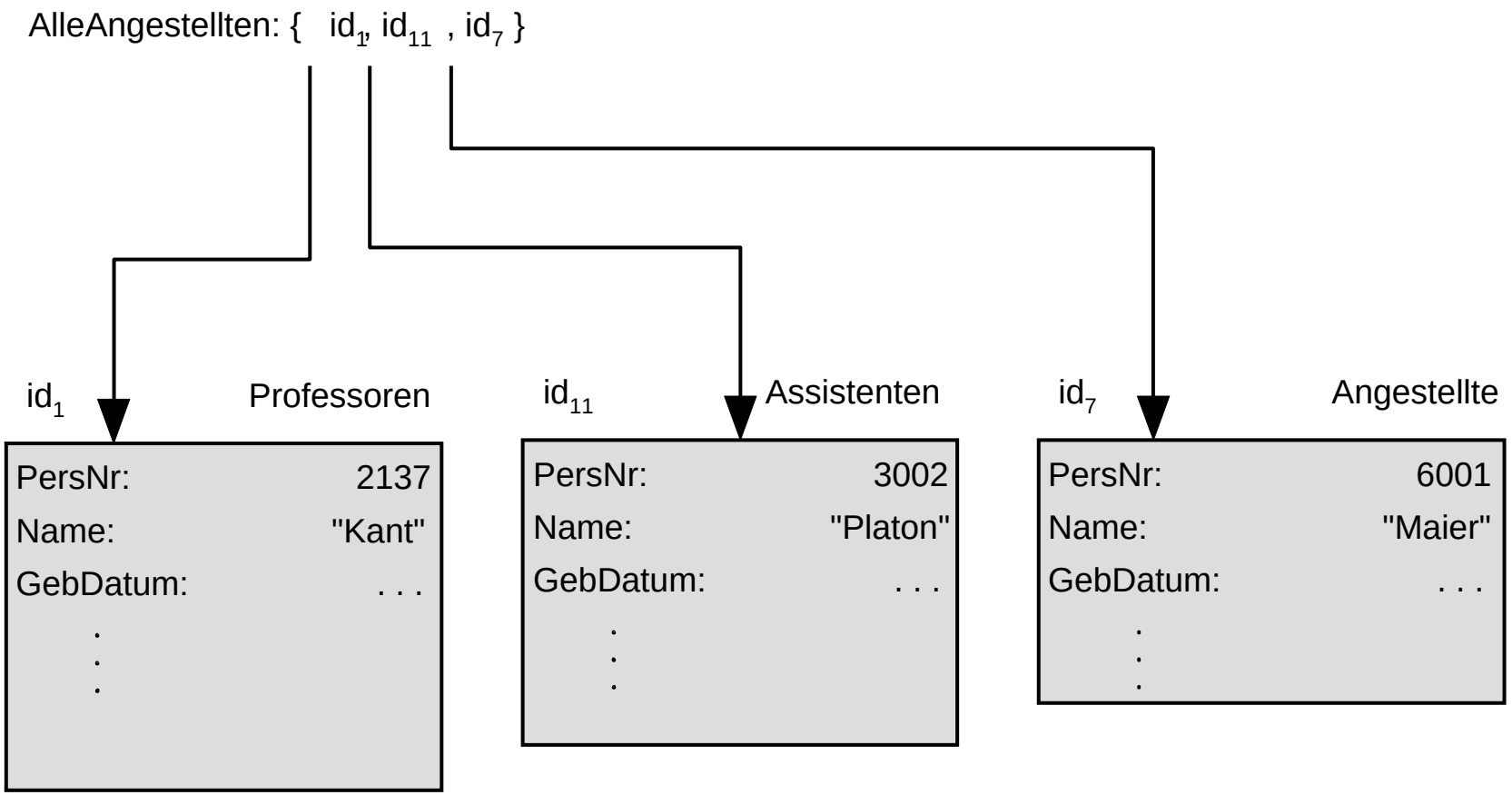
Vererbung

```
class Angestellte (extent AlleAngestellte) {
  attribute long PersNr;
  attribute string Name;
  attribute date GebDatum;
  short Alter();
  long Gehalt();
};

class Assistenten extends Angestellte (extent AlleAssistenten) {
  attribute string Fachgebiet;
};

class Professoren extends Angestellte (extent AlleProfessoren) {
  attribute string Rang;
  relationship Raeume residiertIn inverse Raeume::beherbergt;
  relationship set(Vorlesungen) liest inverse Vorlesungen::gelesenVon;
  relationship set(Pruefungen) hatgeprueft inverse Pruefungen::Pruefer;
};
```

Extension AlleAngestellten




```
select sum(a.Gehalt()) from a in AlleAngestellten
```

OQL: select

Liste alle C4-Professoren (als Objekte):

```
select p
from p in AlleProfessoren
where p.Rang = "C4";
```

Liste Name und Rang aller C4-Professoren (als Werte):

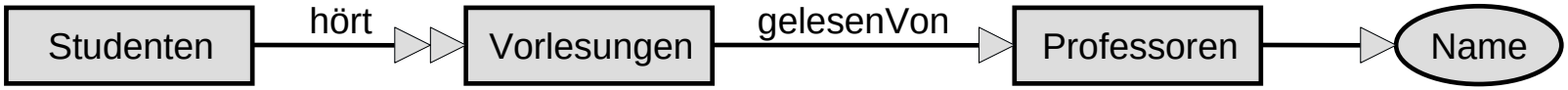
```
select p.Name, p.Rang
from p in AlleProfessoren
where p.Rang = "C4";
```

Liste Name und Rang aller C4-Professoren (als Tupel):

```
select struct (n: p.Name, r: p.Rang)
from p in AlleProfessoren
where p.Rang = "C4";
```

OQL: Pfadausdrücke

Welche Studenten hören Vorlesungen von Sokrates ?




```
select s.Name
from s in AlleStudenten, v in s.hoert
where v.gelesenVon.Name = 'Sokrates'
```

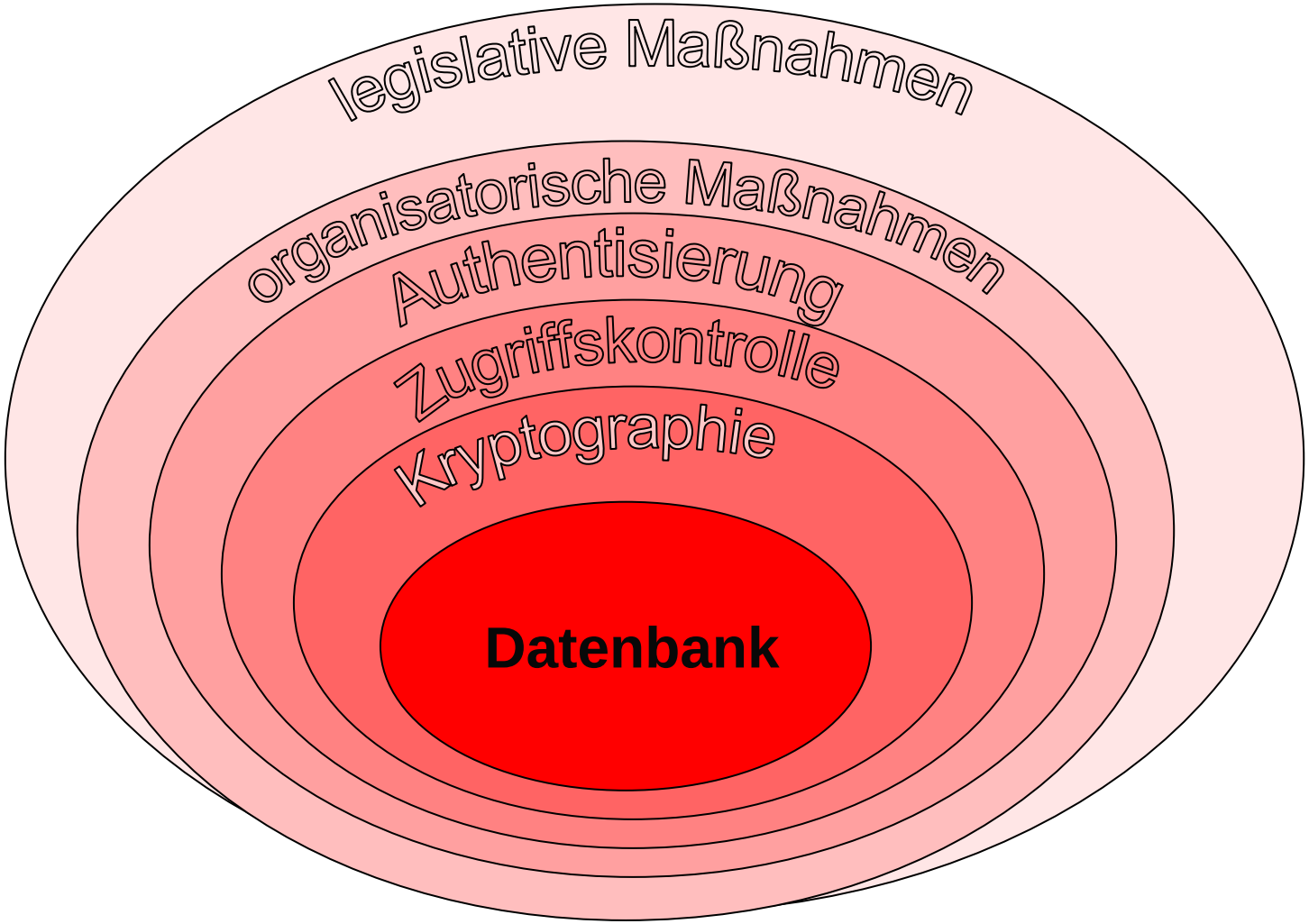
Datenbanksysteme 2011

Kapitel 17:
Sicherheit

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

Datenschutz




```
grant select on Professoren to Erika with grant option
```

Datenbanksysteme 2011

Kapitel 18:
Data Warehouse

Oliver Vornberger

Institut für Informatik
Universität Osnabrück

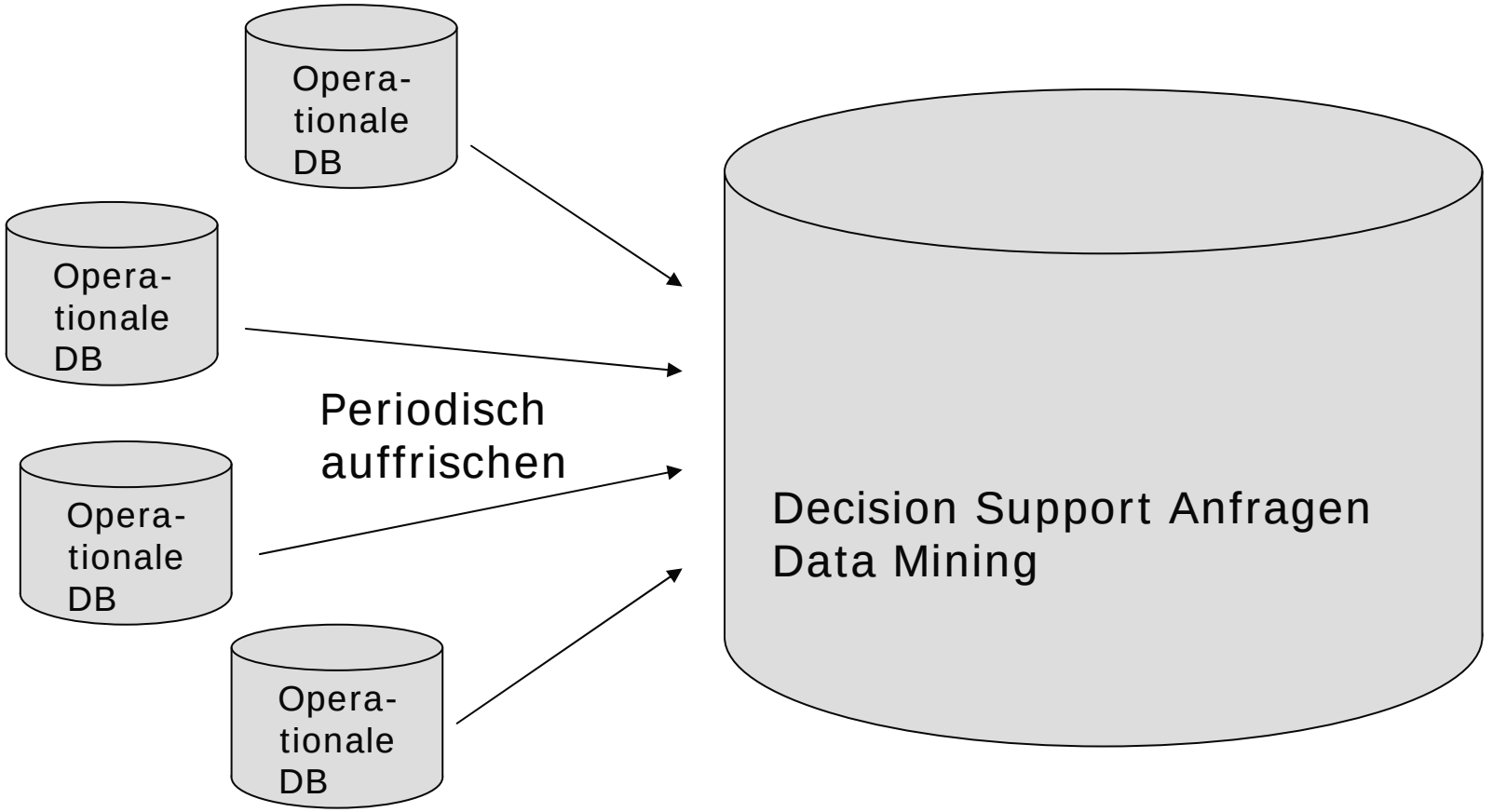
OLTP versus OLAP

online transaction processing

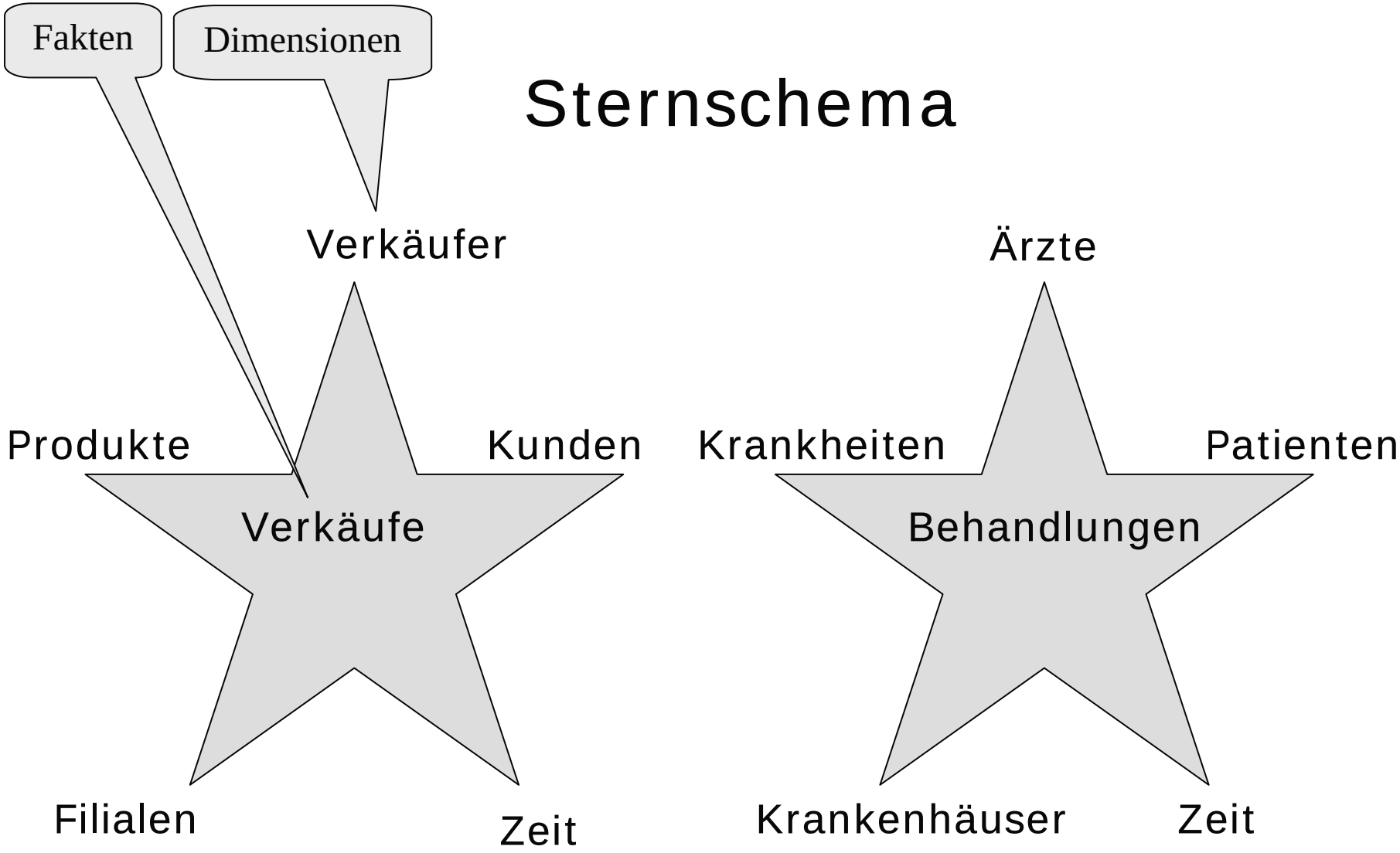
OLTP

online analytical processing

OLAP



Sternschema



Star Join

Welche Handys (d.h. von welchen Herstellern)
haben junge Kunden in den bayrischen Filialen
zu Weihnachten 1996 gekauft ?

```
select p.Hersteller, sum(v.Anzahl) as Anzahl
from Verkäufe v, Filialen f, Produkte p, Zeit z, Kunden k
where z.Saison = 'Weihnachten'
and z.Jahr       = 1996
and k.wiealt     < 30
and p.Produkttyp = 'Handy'
and f.Bezirk     = 'Bayern'
and v.VerkDatum  = z.Datum
and v.Produkt    = p.ProduktNr
and v.Filiale    = f.Filialenkennung
and v.Kunde      = k.KundenNr
group by Hersteller;
```

MySQL: WITH ROLLUP

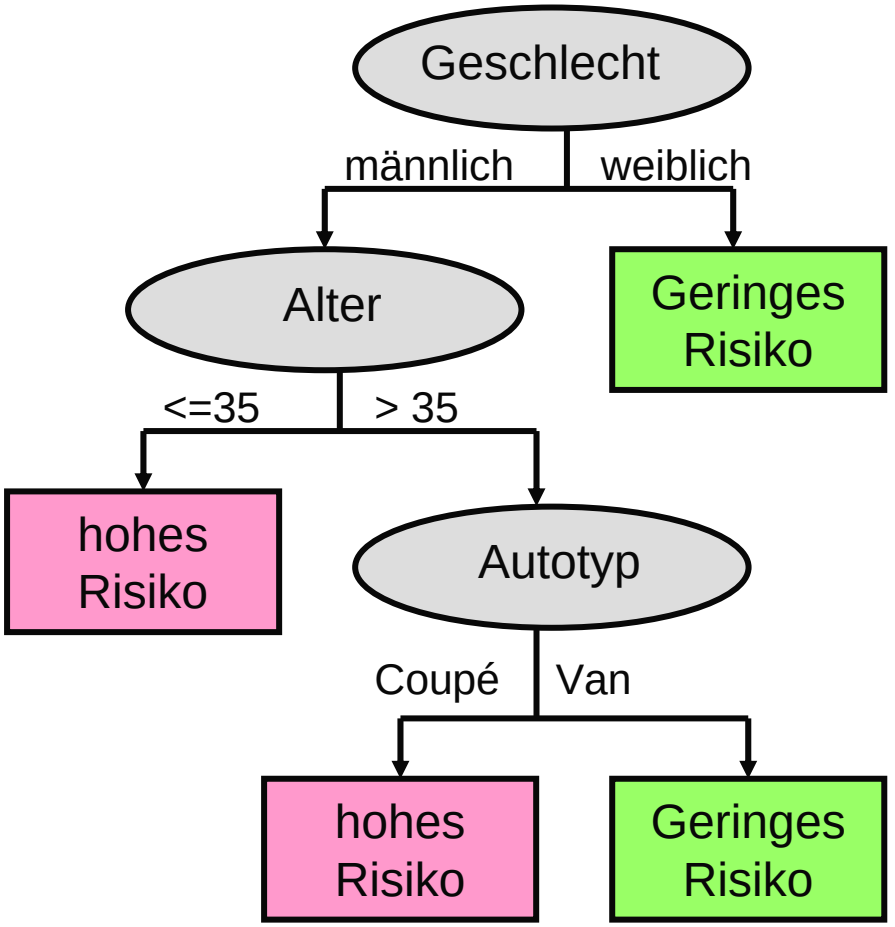
```
select Hersteller, Jahr, sum(Umsatz) as Umsatz
from Handys
group by Hersteller, Jahr
with Rollup
```

Hersteller	Jahr	Umsatz
Bosch	1994	500
Bosch	1995	1000
Bosch	1996	1500
Bosch	*(null)*	3000
Motorola	1994	1000
Motorola	1995	1000
Motorola	1996	1500
Motorola	*(null)*	3500
Nokia	1994	1000
Nokia	1995	1500
Nokia	1996	2000
Nokia	*(null)*	4500
Siemens	1994	2000
Siemens	1995	3000
Siemens	1996	3500
Siemens	*(null)*	8500
(null)	*(null)*	19500

MySQL WorkBench

Beispiel für Klassifikation

Alter	Geschlecht	Autotyp	Schaden
45	**w**	**Van**	**gering**
18	**w**	**Coupé**	**gering**
22	**w**	**Van**	**gering**
19	**m**	**Coupé**	**hoch**
38	**w**	**Coupé**	**gering**
24	**m**	**Van**	**Gering**
40	**m**	**Coupé**	**hoch**
40	**m**	**Van**	**gering**
…	…	…	…



Beispiel für Frequent Item Set

TransID	Produkt
111	Drucker
111	Papier
111	PC
111	Toner
222	PC
222	Scanner
333	Drucker
333	Papier
333	Toner
444	Drucker
444	PC
555	Drucker
555	Papier
555	PC
555	Scanner
555	Toner

Frequent Itemset-Kandidat	Anzahl
{Drucker}	4
{Papier}	3
{PC}	4
{Scanner}	2
{Toner}	3
{Drucker, Papier}	3
{Drucker, PC}	3
{Drucker, Scanner}	
{Drucker, Toner}	3
{Papier, PC}	2
{Papier, Scanner}	
{Papier, Toner}	3
{PC, Scanner}	
{PC, Toner}	2
{Scanner, Toner}	
{Drucker, Papier, PC}	
{Drucker, Papier, Toner}	3
{Drucker, PC, Toner}	
{Papier, PC, Toner}	