Moderne Betriebliche Anwendungen von Datenbanksystemen

Online Transaction Processing

Betriebswirtschaftliche Standard-Software (SAP R/3)

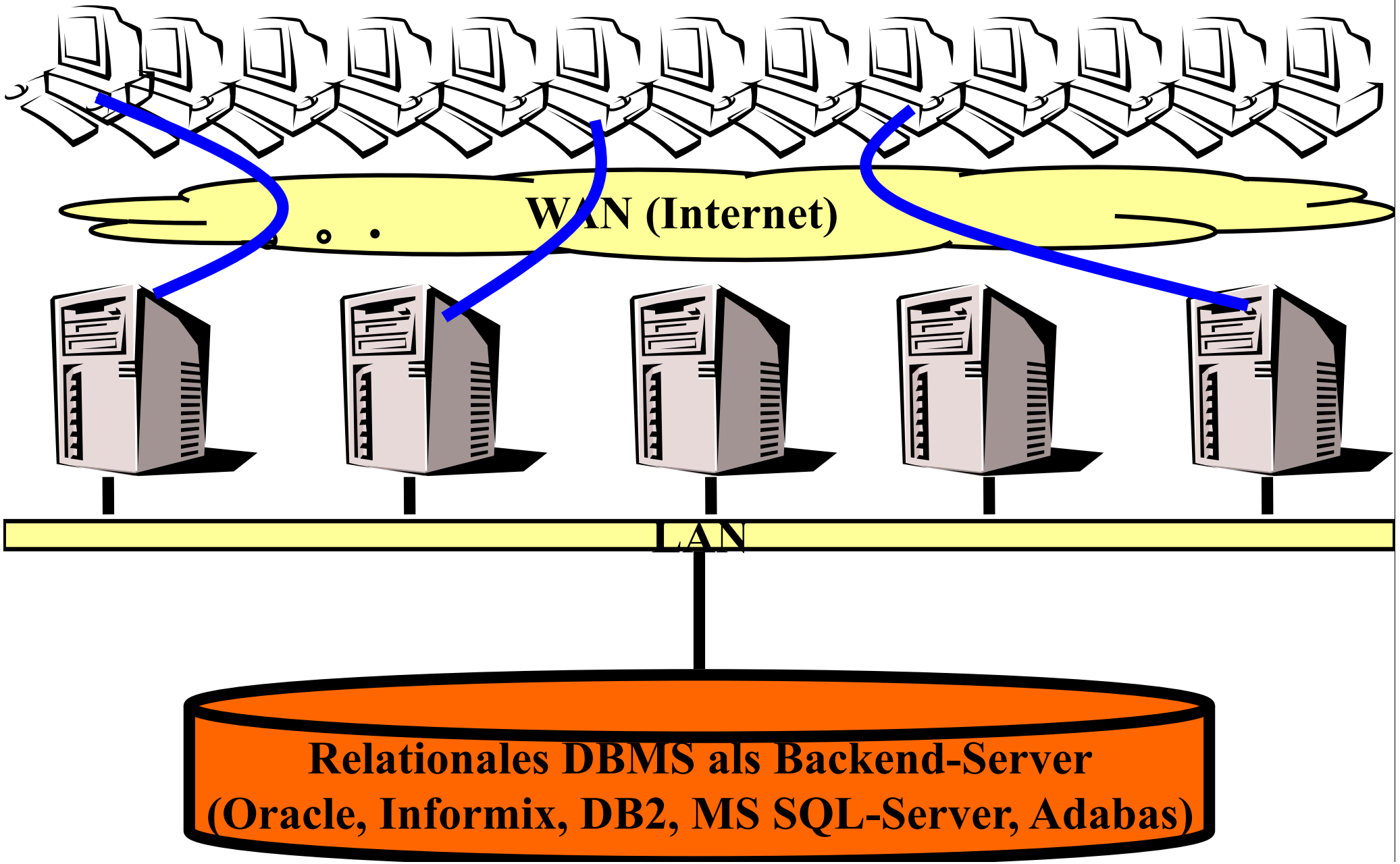
Data Warehouse-Anwendungen

Data Mining

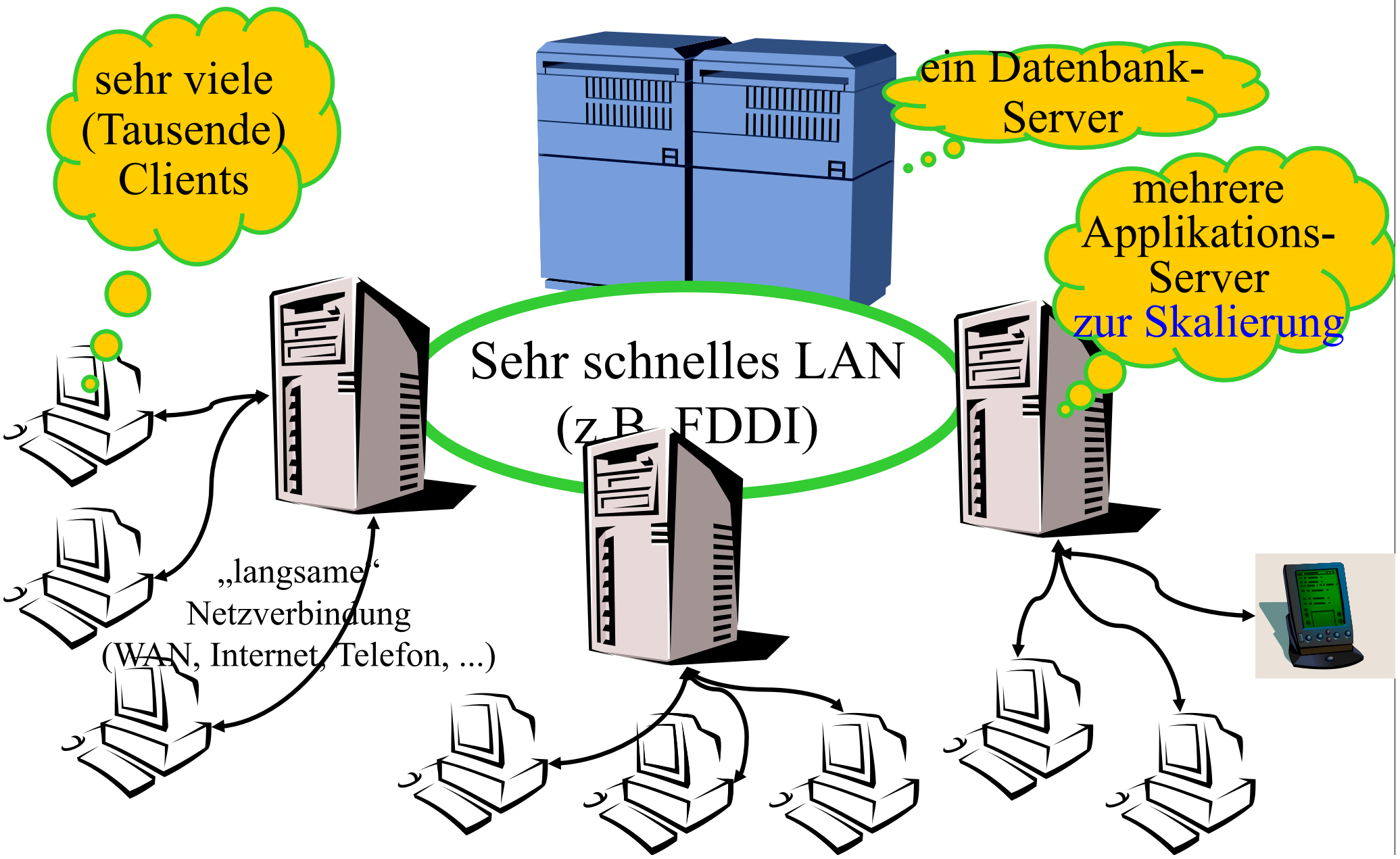
OLTP: Online Transaction Processing

- Beispiele
  - Flugbuchungssystem
  - Bestellungen in einem Handelsunternehmen
- Charakterisierung
  - Hoher Parallelitätsgrad
  - Viele (Tausende pro Sekunde) kurze Transaktionen
  - TAs bearbeiten nur ein kleines Datenvolumen
  - „mission-critical" für das Unternehmen
  - Hohe Verfügbarkeit muss gewährleistet sein
- Normalisierte Relationen (möglichst wenig Update-Kosten)
- Nur wenige Indexe (wegen Fortschreibungskosten)

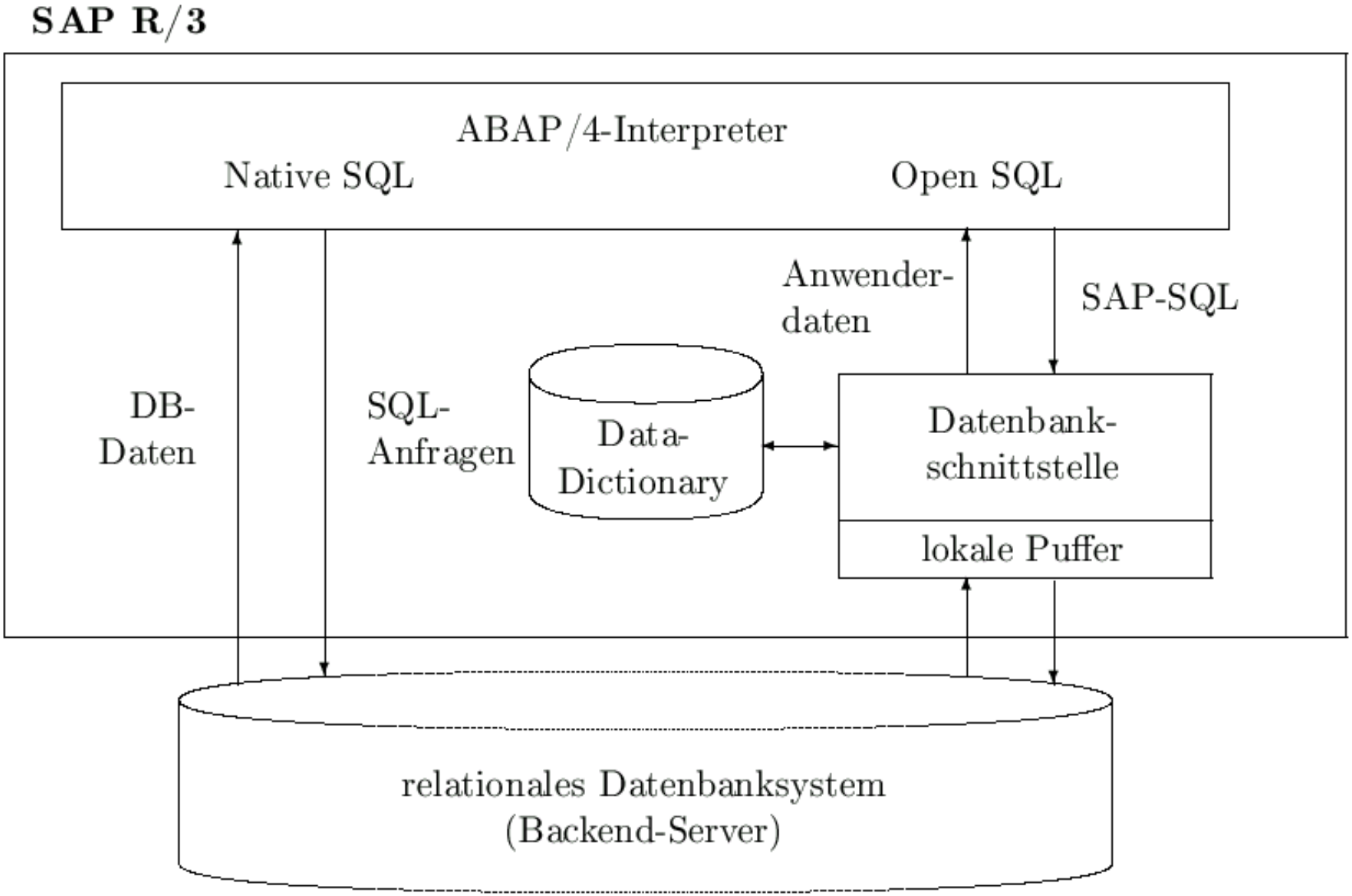
SAP R/3: Enterprise Resource Modelling (ERP-System)



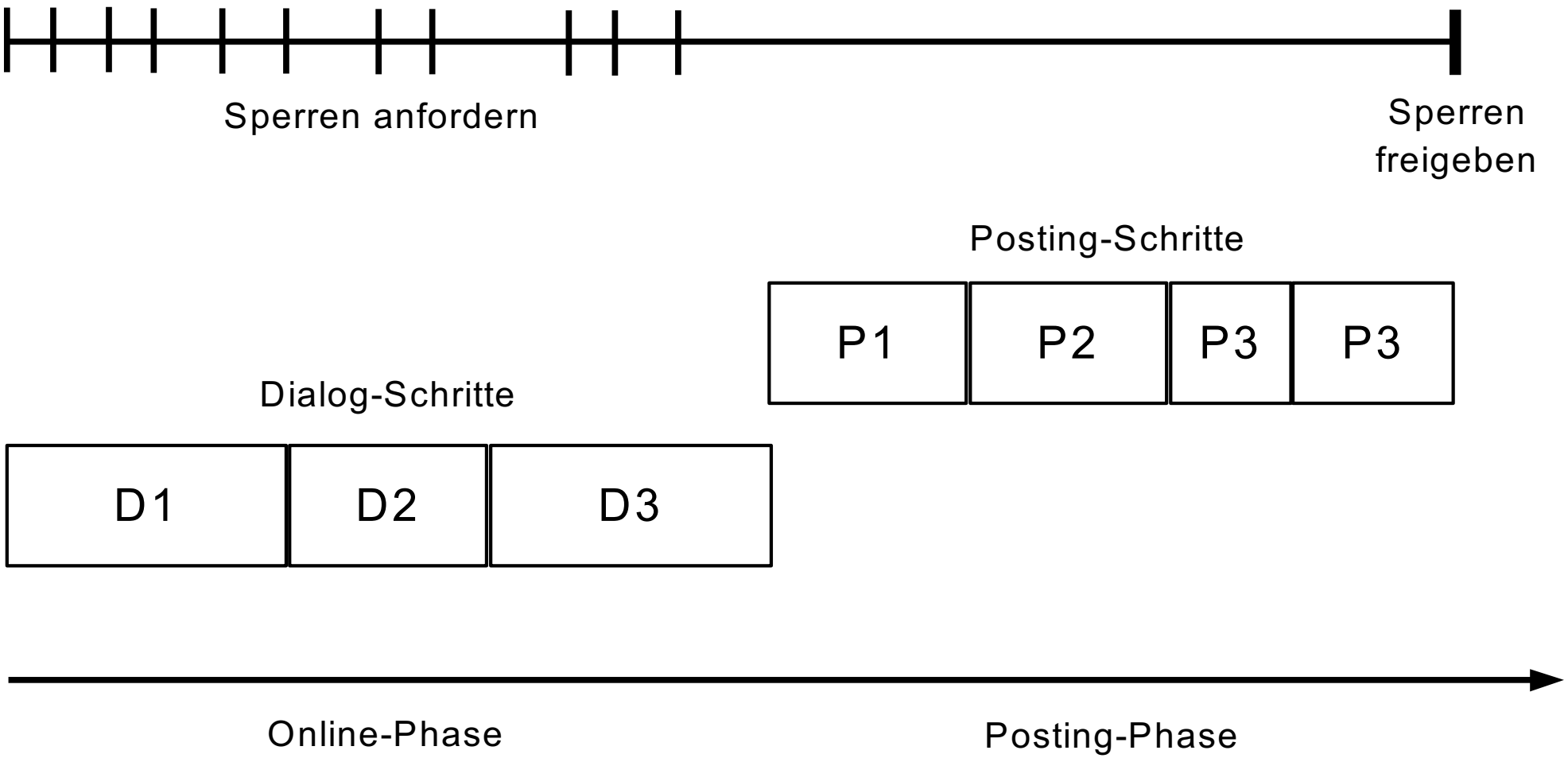
Dreistufige Client/Server-Architektur (3 Tier, SAP R/3)



Interne Architektur von SAP R/3



Transaktionsverarbeitung in SAP R/3



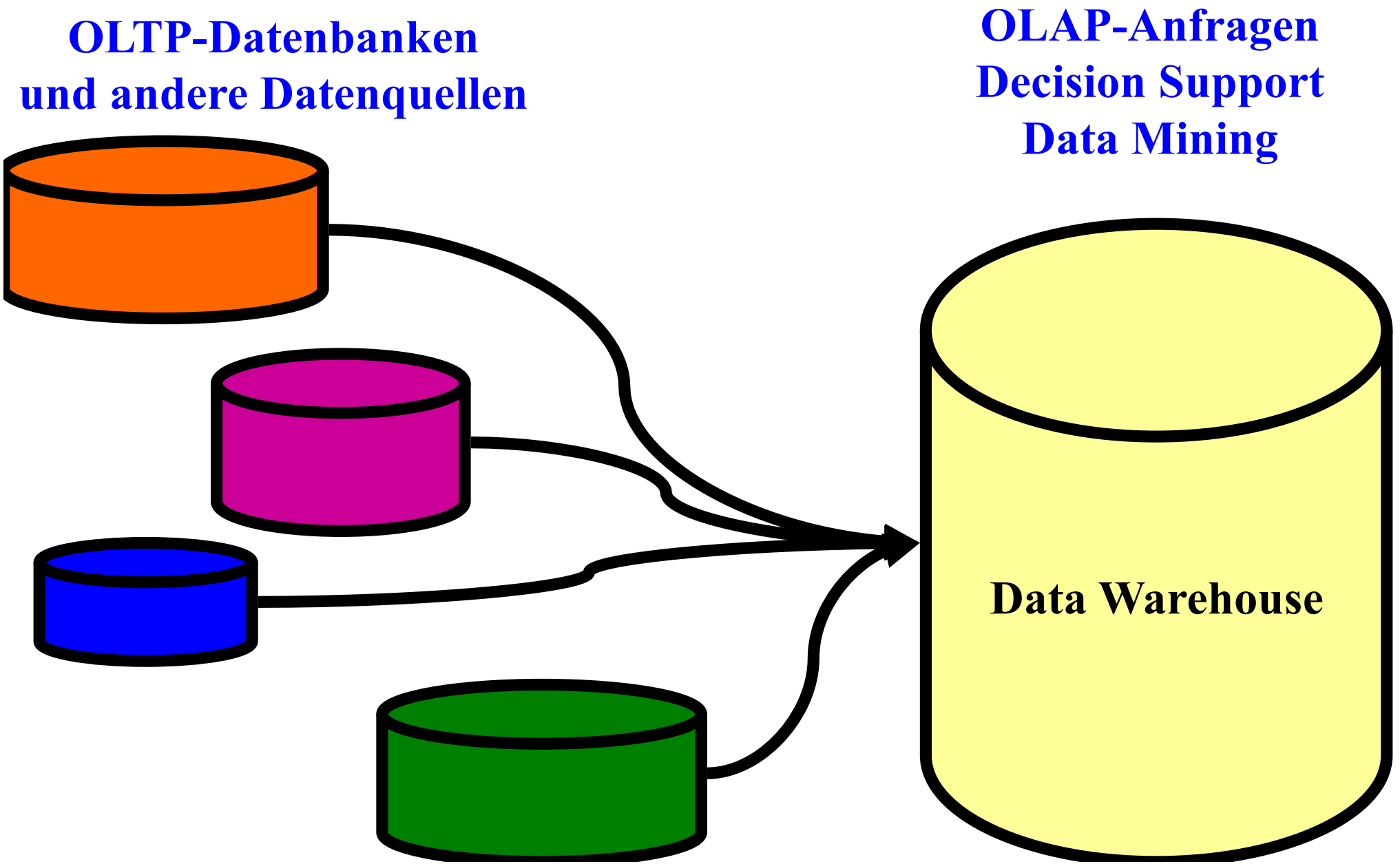
Data Warehouse-Anwendungen: OLAP~Online Analytical Processing

- Wie hat sich die Auslastung der Transatlantikflüge über die letzten zwei Jahre entwickelt?

oder

- Wie haben sich besondere offensive Marketingstrategien für bestimmte Produktlinien auf die Verkaufszahlen ausgewirkt?

Sammlung und periodische Auffrischung der Data Warehouse-Daten



Das Stern-Schema

Verkäufe					
VerkDatum	Filiale	Produkt	Anzahl	Kunde	Verkäufer
25-Jul-00	Passau	1347	1	4711	825
...	...	...	...	...	...

Filialen			
Filialenkennung	Land	Bezirk	...
Passau	D	Bayern	...
...	...	...	...

Kunden			
KundenNr	Name	wiealt	...
4711	Kemper	43	...
...	...	...	...

Verkäufer					
VerkäuferNr	Name	Fachgebiet	Manager	wiealt	...
825	Handyman	Elektronik	119	23	...
...	...	...	...	...	...

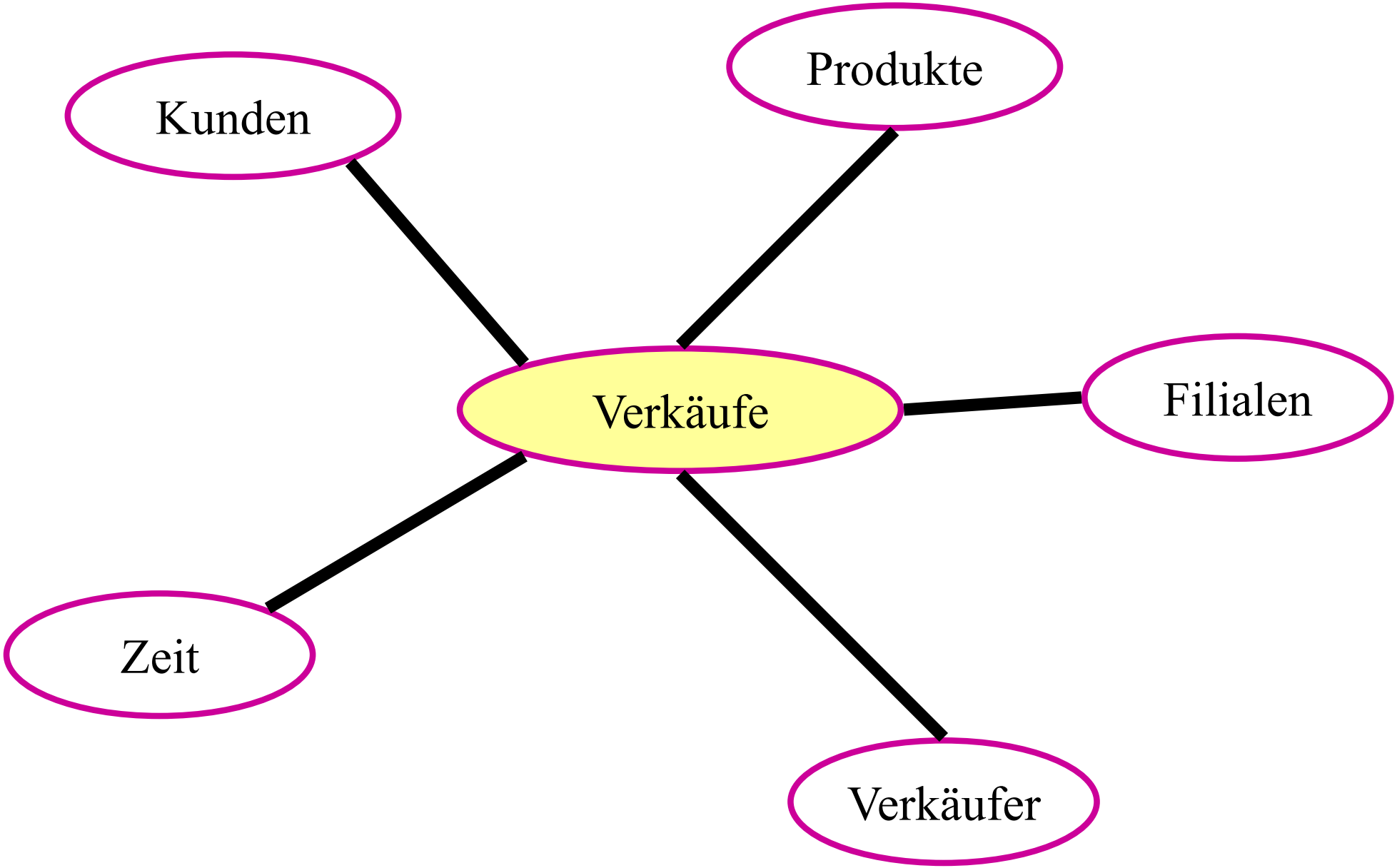
Zeit								
Datum	Tag	Monat	Jahr	Quartal	KW	Wochentag	Saison	...
...	...	...	...	...	...	...	...	...
25-Jul-00	25	Juli	2000	3	30	Dienstag	Hochsommer	...
...	...	...	...	...	...	...	...	...
18-Dec-01	18	Dezember	2001	4	52	Dienstag	Weihnachten	...
...	...	...	...	...	...	...	...	...

Produkte					
ProduktNr	Produkttyp	Produktgruppe	Produkthauptgruppe	Hersteller	...
1347	Handy	Mobiltelekom	Telekom	Siemens	...
...	...	...	...	...	...

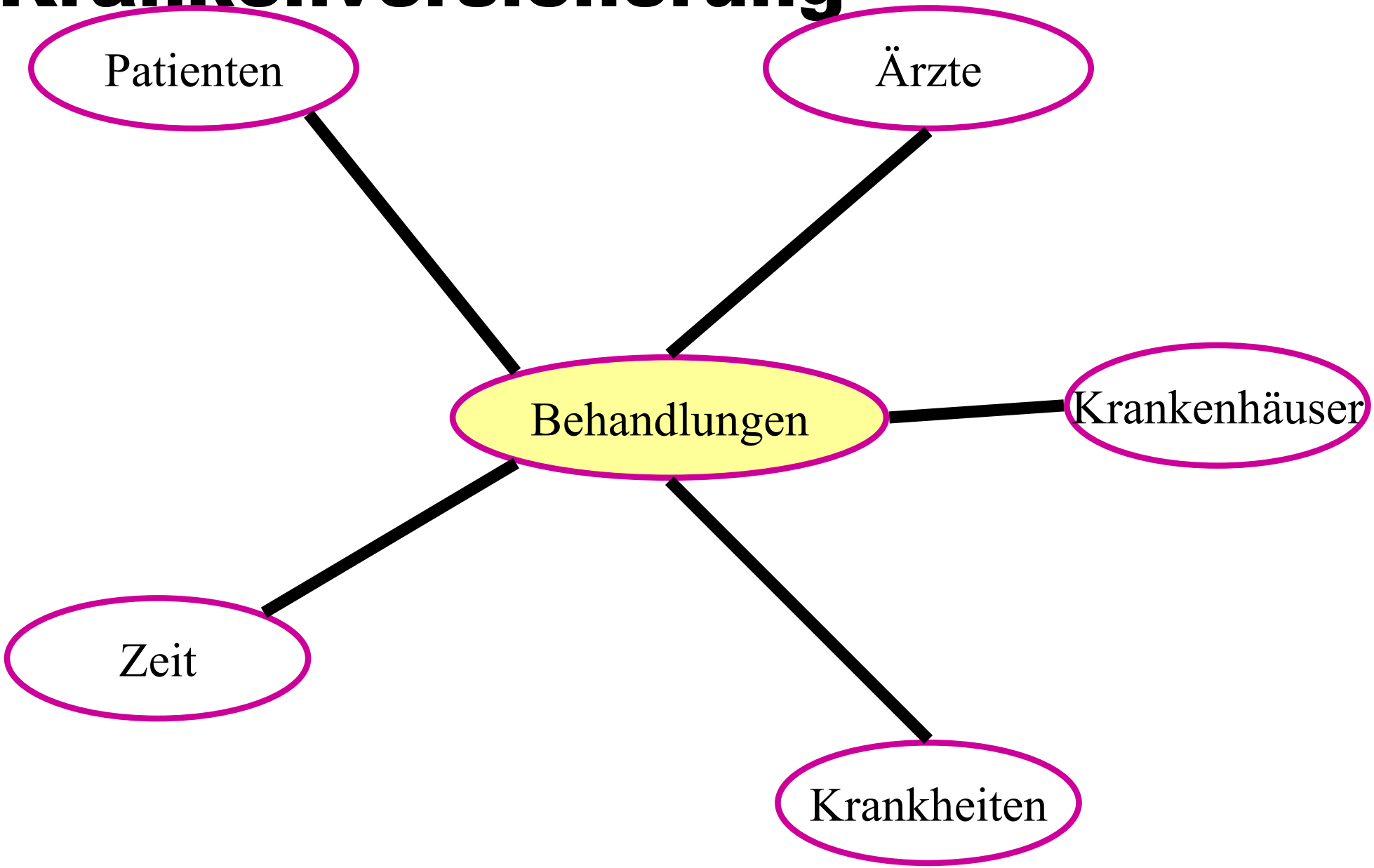
Stern-Schema bei Data Warehouse-Anwendungen

- Eine sehr große Faktentabelle
  - Alle Verkäufe der letzten drei Jahre
  - Alle Telefonate des letzten Jahres
  - Alle Flugreservierungen der letzten fünf Jahre
  - normalisiert
- Mehrere Dimensionstabellen
  - Zeit
  - Filialen
  - Kunden
  - Produkt
  - Oft nicht normalisiert

Das Stern-Schema: Handelsunternehmen



Das Stern-Schema: Krankenversicherung



Stern-Schema

Verkäufe					
VerkDatum	Filiale	Produkt	Anzahl	Kunde	Verkäufer
25-Jul-00	Passau	1347	1	4711	825
...	...	...	...	...	...

Faktentabelle (SEHR groß)

Filialen			
FilialenKennung	Land	Bezirk	...
Passau	D	Bayern	...
...	...	...	...

Kunden			
KundenNr	Name	wieAlt	...
4711	Kemper	43	...
...	...	...	...

Dimensionstabellen (relativ klein)

Verkäufer					
VerkäuferNr	Name	Fachgebiet	Manager	wieAlt	...
825	Handyman	Elektronik	119	23	...
...	...	...	...	...	...

Stern-Schema (cont'd)

Zeit							
Datum	Tag	Monat	Jahr	Quartal	KW	Wochentag	Saison
25-Jul-00	25	7	2000	3	30	Dienstag	Hochsommer
...	...	...	...	...	...		
18-Dec-01	18	12	2001	4	52	Dienstag	Weihnachten
...	...	...	...	...	...	...	...

Produkte					
ProduktNr	Produkttyp	Produktgruppe	Produkthauptgruppe	Hersteller	..
1347	Handy	Mobiltelekom	Telekom	Siemens	..
...	...	...	...	...	..

Nicht-normalisierte Dimensionstabellen: effizientere Anfrageauswertung

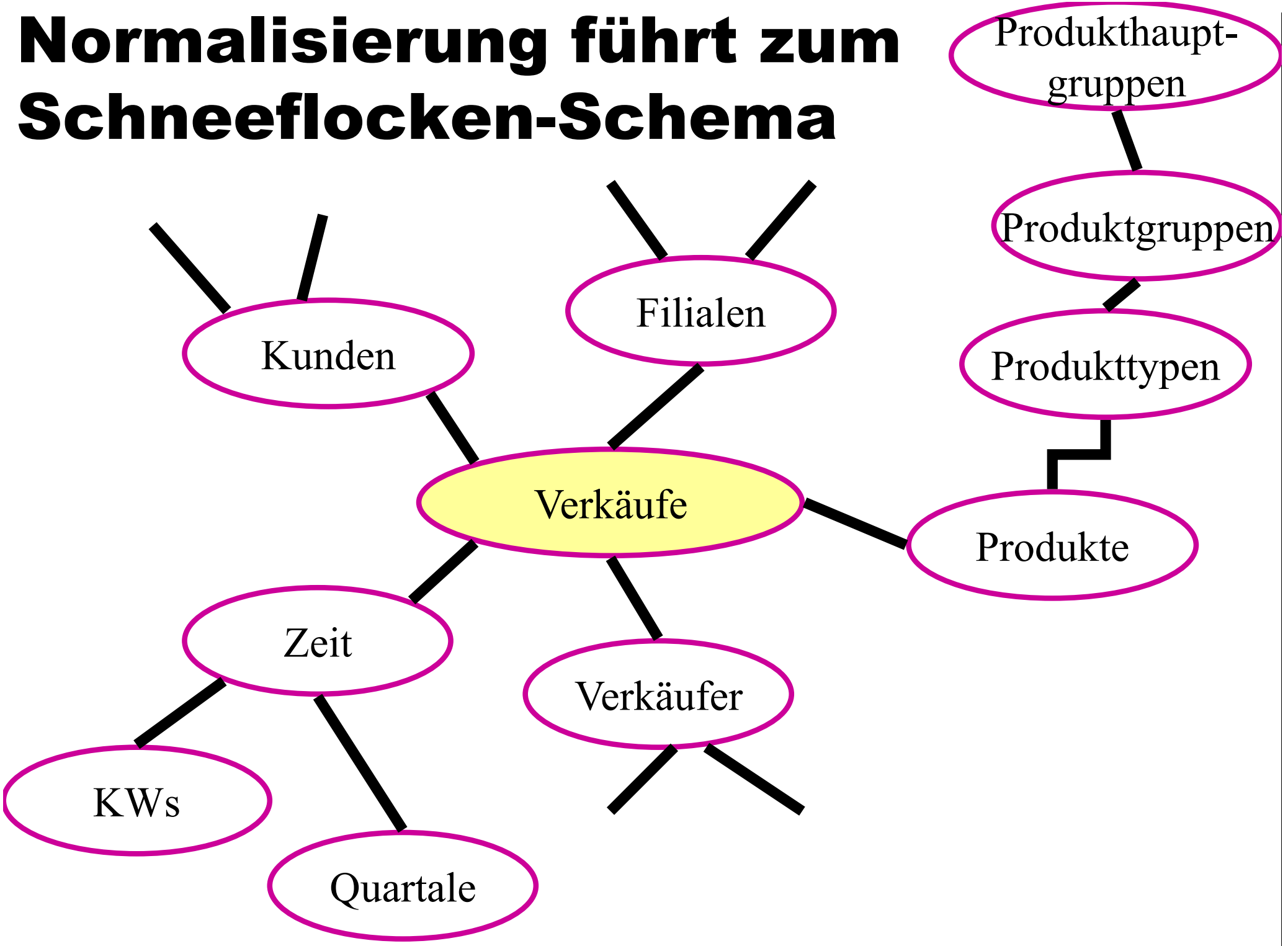
Zeit							
Datum	Tag	Monat	Jahr	Quartal	KW	Wochentag	Saison
25-Jul-00	25	7	2000	3	30	Dienstag	Hochsommer
…	…	…	…	…	…		
18-Dec-01	18	12	2001	4	52	Dienstag	Weihnachten
…	…	…	…	…	…	…	…

Datum → Monat → Quartal

Produkte					
ProduktNr	Produkttyp	Produktgruppe	Produkthauptgruppe	Hersteller	..
1347	Handy	Mobiltelekom	Telekom	Siemens	..
…	…	…	…	…	..

ProduktNr → Produkttyp → Produktgruppe → Produkthauptgruppe

Normalisierung führt zum Schneeflocken-Schema



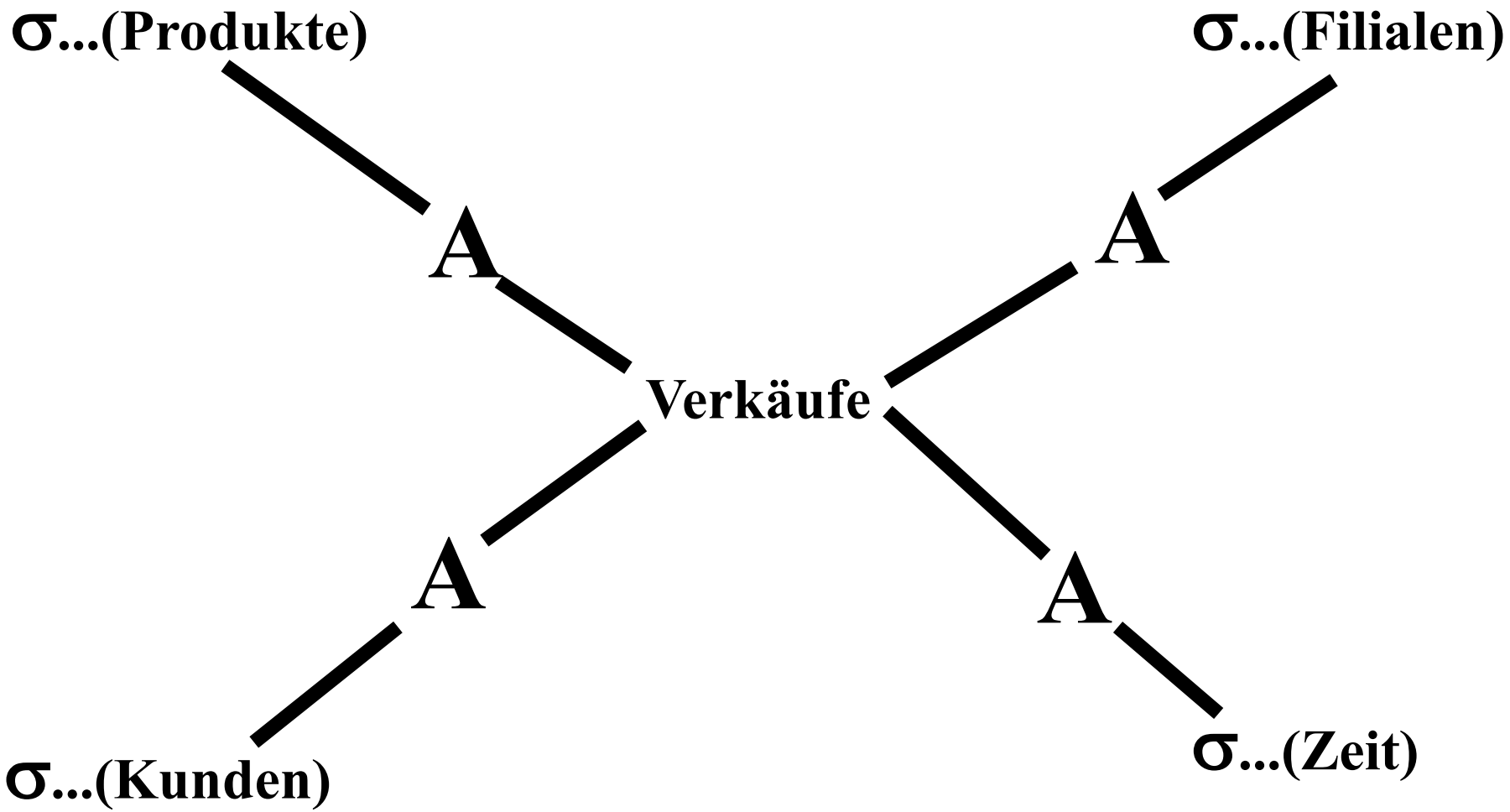
Anfragen im Sternschema

```
select sum(v.Anzahl), p.Hersteller
from Verkäufe v, Filialen f, Produkte p, Zeit z, Kunden k
where z.Saison = 'Weihnachten' and
      z.Jahr = 2001 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 p.Hersteller;
```

Einschränkung der Dimensionen

Join-Prädikate

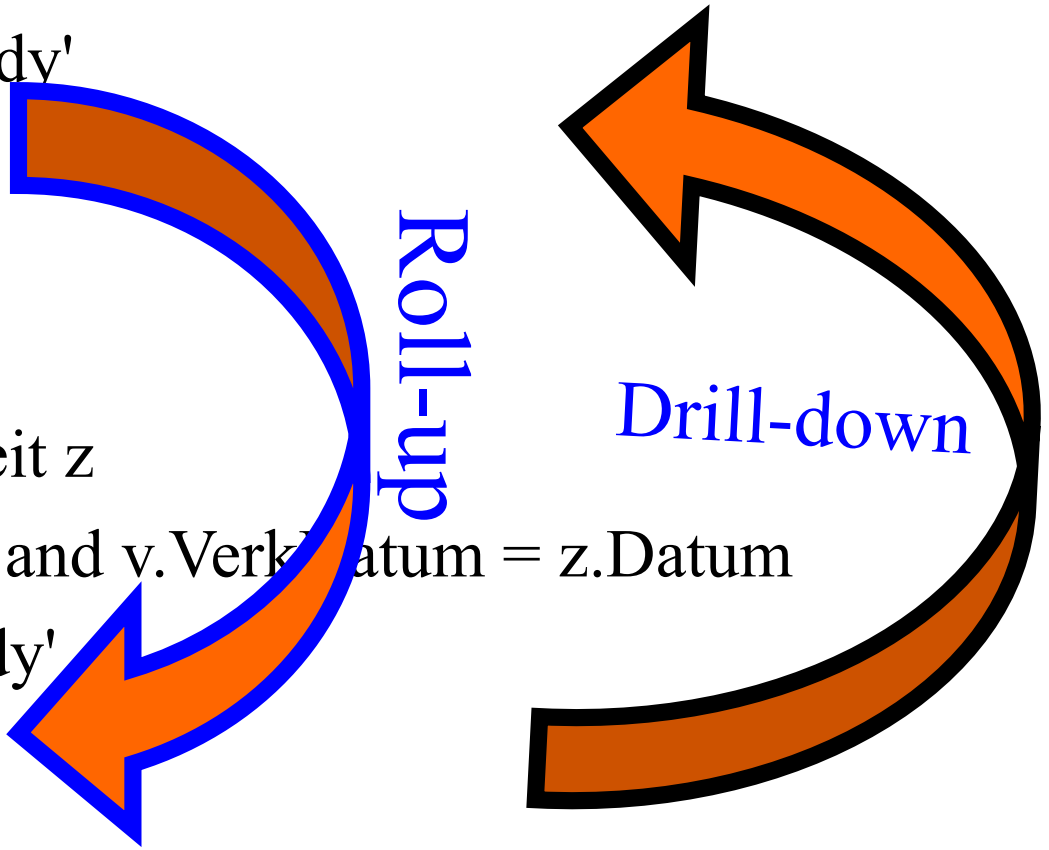
Algebra-Ausdruck



Roll-up/Drill-down-Anfragen

```
select Jahr, Hersteller, sum(Anzahl)
from Verkäufe v, Produkte p, Zeit z
where v.Produkt = p.ProduktNr and v.VerkDatum = z.Datum
        and p.Produkttyp = 'Handy'
group by p.Hersteller, z.Jahr;
```

```
select Jahr, sum(Anzahl)
from Verkäufe v, Produkte p, Zeit z
where v.Produkt = p.ProduktNr and v.VerkDatum = z.Datum
        and p.Produkttyp = 'Handy'
group by z.Jahr;
```



Ultimative Verdichtung

```
select sum(Anzahl)
from Verkäufe v, Produkte p
where v.Produkt = p.ProduktNr and p.Produkttyp = 'Handy';
```

Handyverkäufe nach Hersteller und Jahr		
Hersteller	Jahr	Anzahl
Siemens	1999	2.000
Siemens	2000	3.000
Siemens	2001	3.500
Motorola	1999	1.000
Motorola	2000	1.000
Motorola	2001	1.500
Bosch	1999	500
Bosch	2000	1.000
Bosch	2001	1.500
Nokia	1999	1.000
Nokia	2000	1.500
Nokia	2001	2.000

Handyverkäufe nach Jahr	
Jahr	Anzahl
1999	4.500
2000	6.500
2001	8.500

Handyverkäufe nach Hersteller	
Hersteller	Anzahl
Siemens	8.500
Motorola	3.500
Bosch	3.000
Nokia	4.500

Abb. 17.7: Analyse der Handyverkaufszahlen nach unterschiedlichen Dimensionen

Hersteller \ Jahr	1999	2000	2001	Σ
Siemens	2.000	3.000	3.500	8.500
Motorola	1.000	1.000	1.500	3.500
Bosch	500	1.000	1.500	3.000
Nokia	1.000	1.500	2.000	4.500
Σ	4.500	6.500	8.500	19.500

Abb. 17.8: Handyverkäufe nach Jahr und Hersteller

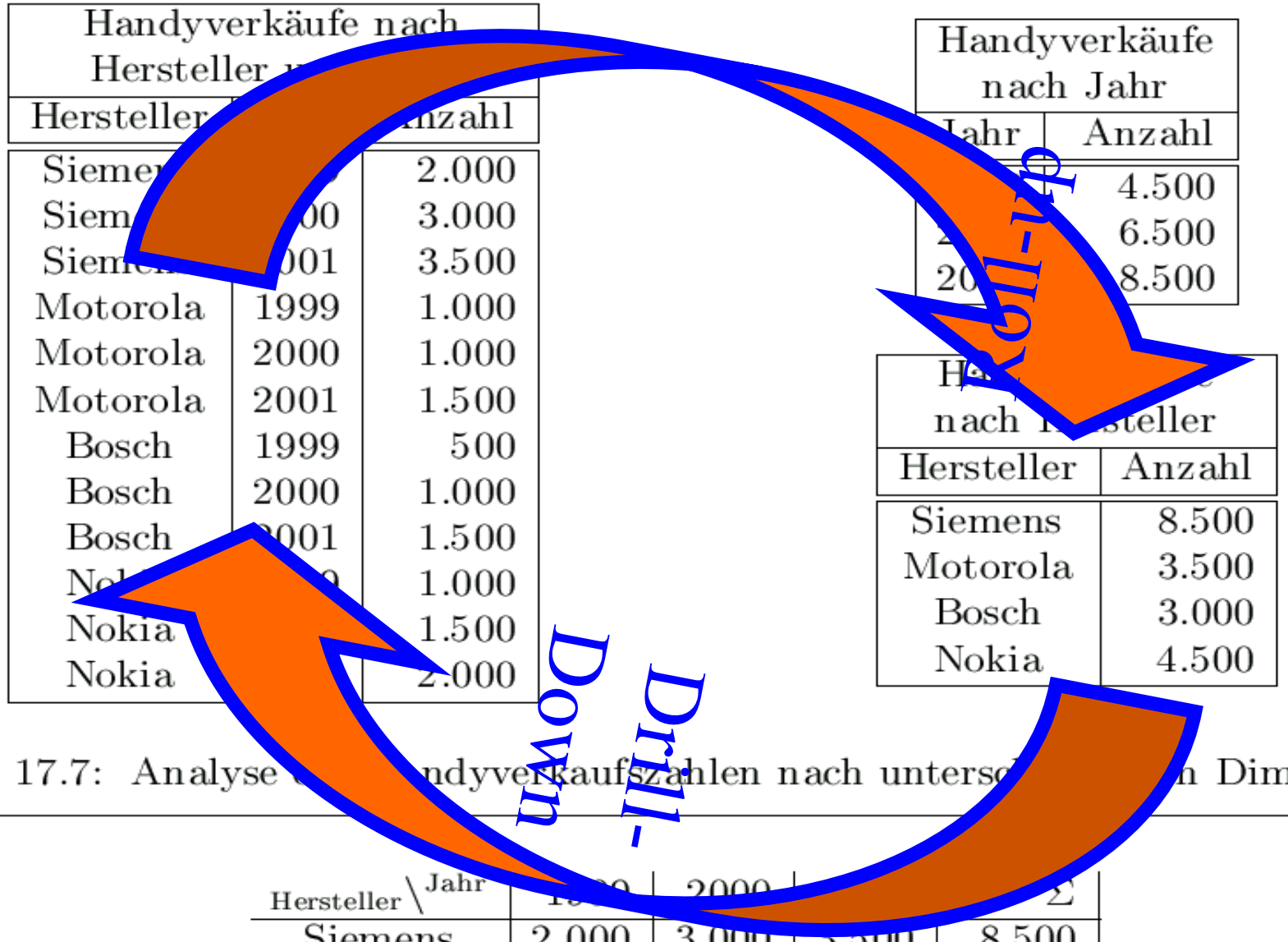
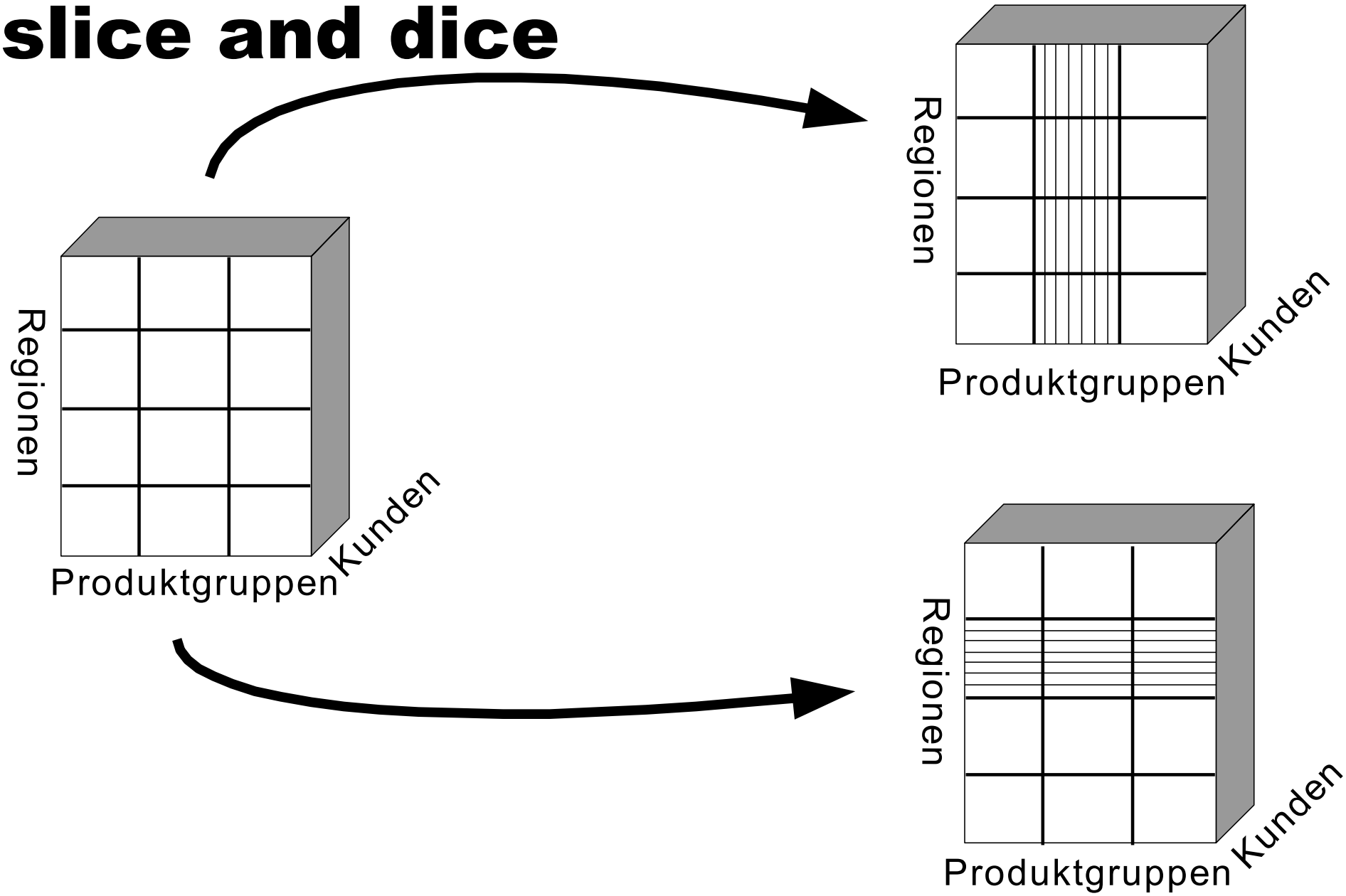


Abb. 17.7: Analyse [illegible]ndyverkaufszahlen nach unters[illegible]n Dimensionen

Hersteller \ Jahr	[illegible]	2000	[illegible]	Σ
Siemens	2.000	3.000	3.500	8.500
Motorola	1.000	1.000	1.500	3.500
Bosch	500	1.000	1.500	3.000
Nokia	1.000	1.500	2.000	4.500
Σ	4.500	6.500	8.500	19.500

Abb. 17.8: Handyverkäufe nach Jahr und Hersteller

Flexible Auswertungsmethoden: slice and dice



Materialisierung von Aggregaten

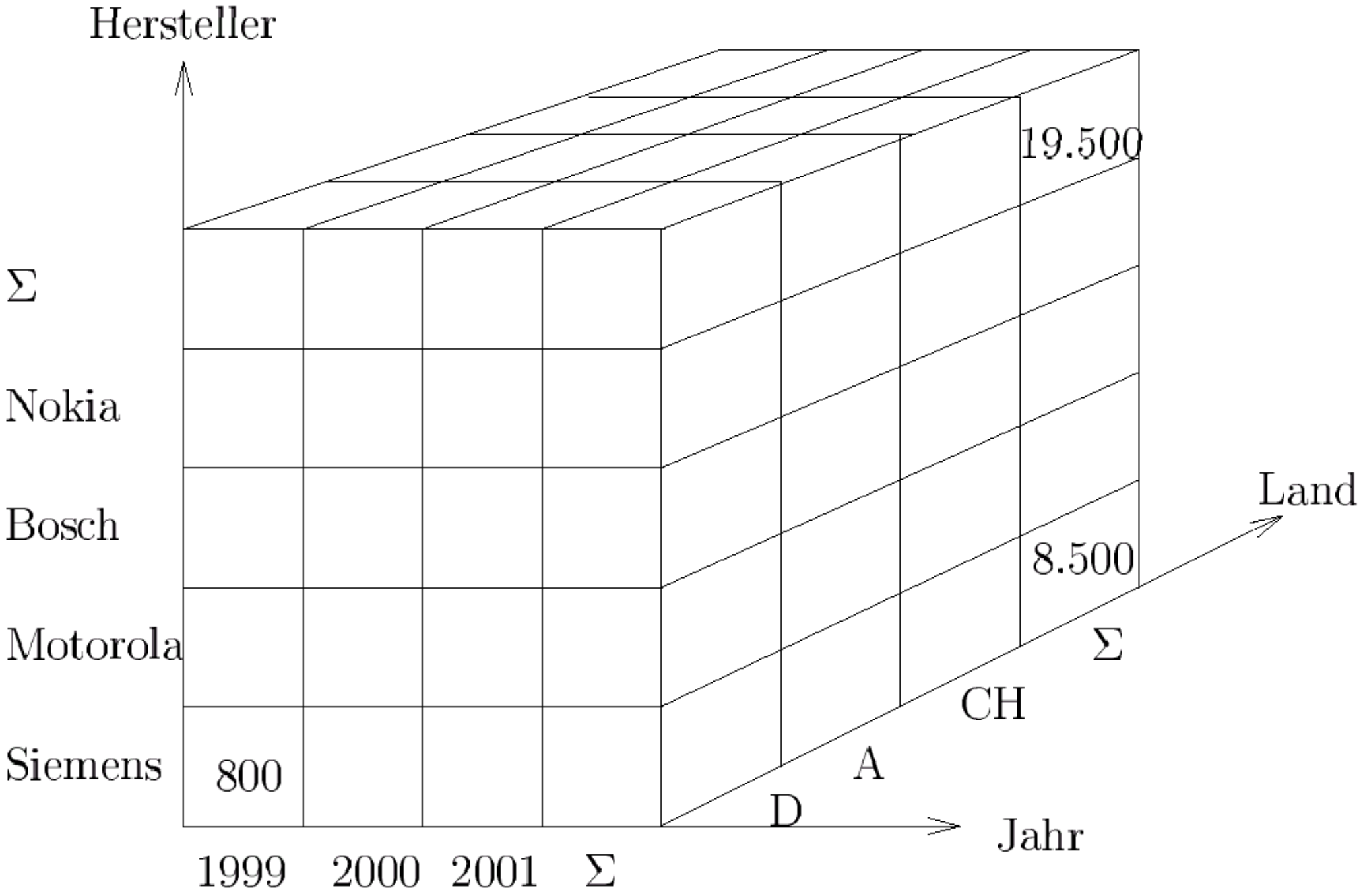
```
insert into Handy2DCube
( select  p.Hersteller, z.Jahr, sum(v.Anzahl)
   from Verkäufe v, Produkte p, Zeit z
   where v.Produkt = p.ProduktNr  and p.Produkttyp = 'Handy'
            and v.VerkDatum = z.Datum
   group by z.Jahr, p.Hersteller )       union
( select  p.Hersteller, to_number(null), sum(v.Anzahl)
  from Verkäufe v, Produkte p
  where v.Produkt = p.ProduktNr  and p.Produkttyp = 'Handy'
  group by p.Hersteller )                  union
( select null, z.Jahr, sum(v.Anzahl)
  from Verkäufe v, Produkte p, Zeit z
  where v.Produkt = p.ProduktNr  and p.Produkttyp = 'Handy'
           and v.VerkDatum = z.Datum
  group by z.Jahr )                        union
( select  null, to_number(null), sum(v.Anzahl)
  from Verkäufe v, Produkte p
  where v.Produkt = p.ProduktNr  and  p.Produkttyp = 'Handy' );
```

Relationale Struktur der Datenwürfel

Handy2DCube		
Hersteller	Jahr	Anzahl
Siemens	1999	2.000
Siemens	2000	3.000
Siemens	2001	3.500
Motorola	1999	1.000
Motorola	2000	1.000
Motorola	2001	1.500
Bosch	1999	500
Bosch	2000	1.000
Bosch	2001	1.500
Nokia	2000	1.000
Nokia	2001	1.500
Nokia	2001	2.000
null	1999	4.500
null	2000	6.500
null	2001	8.500
Siemens	**null**	8.500
Motorola	**null**	3.500
Bosch	**null**	3.000
Nokia	**null**	4.500
null	**null**	19.500

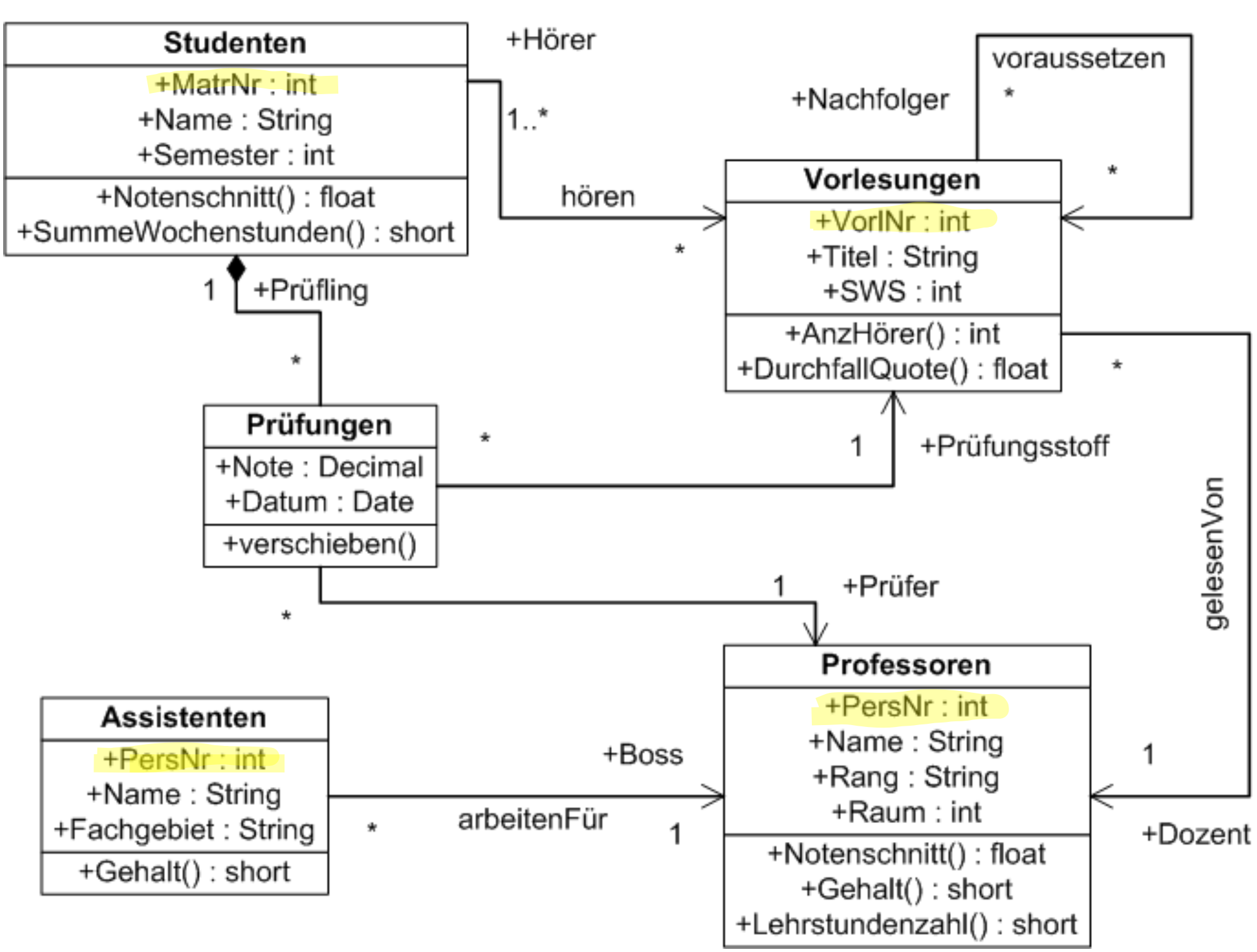
Handy3DCube			
Hersteller	Jahr	Land	Anzahl
Siemens	1999	D	800
Siemens	1999	A	600
Siemens	1999	CH	600
Siemens	2000	D	1.200
Siemens	2000	A	800
Siemens	2000	CH	1.000
Siemens	2001	D	1.400
...	...	...	...
Motorola	1999	D	400
Motorola	1999	A	300
Motorola	1999	CH	300
...	...	...	...
Bosch	...	...	...
...	...	...	...
null	1999	D	...
null	2000	D	...
...	...	...	...
Siemens	**null**	**null**	8.500
...	...	...	...
null	**null**	**null**	19.500

Würfeldarstellung



Der cube-Operator

```
select  p.Hersteller, z.Jahr, f.Land, sum(v.Anzahl)
from Verkäufe v, Produkte p, Zeit z, Filialen f
where v.Produkt = p.ProduktNr   and p.Produkttyp = 'Handy'
        and v.VerkDatum = z.Datum and v.Filiale = f.Filialenkennung
group by cube (z.Jahr, p.Hersteller, f.Land);
```


Studenten
+MatrNr : int
+Name : String
+Semester : int
+Notenschnitt() : float
+SummeWochenstunden() : short
+Hörer
1..*
hören
*
+Nachfolger
voraussetzen
*
*
Vorlesungen
+VorlNr : int
+Titel : String
+SWS : int
+AnzHörer() : int
+DurchfallQuote() : float
1
+Prüfling
*
Prüfungen
+Note : Decimal
+Datum : Date
+verschieben()
*
1
+Prüfungsstoff
*
gelesenVon
1
+Prüfer
*
Professoren
+PersNr : int
+Name : String
+Rang : String
+Raum : int
+Notenschnitt() : float
+Gehalt() : short
+Lehrstundenzahl() : short
1
+Dozent
Assistenten
+PersNr : int
+Name : String
+Fachgebiet : String
+Gehalt() : short
+Boss
*
arbeitenFür
1

Professoren

PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten

MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen

VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen

Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören

MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistenten

PerslNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen

MatrNr	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2

Wiederverwendung von Teil-Aggregaten

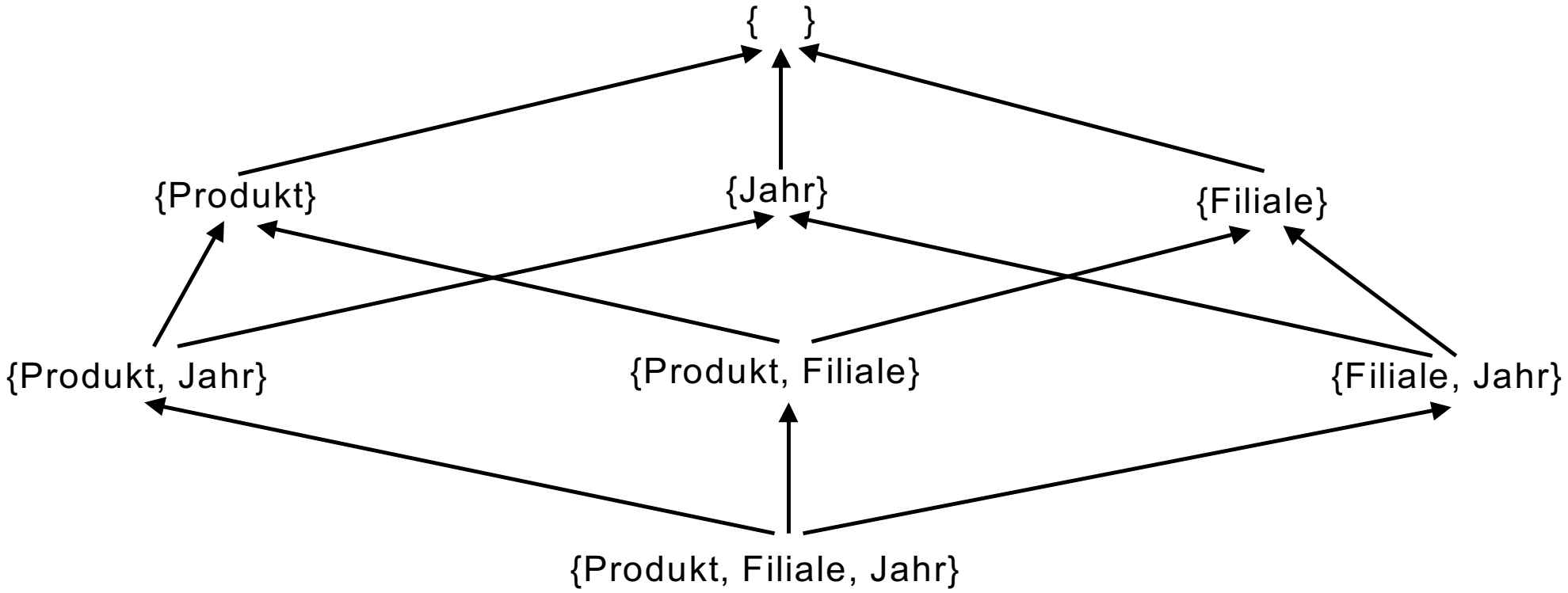
```
insert into VerkäufeProduktFilialeJahr
( select v.Produkt, v.Filiale, z.Jahr, sum(v.Anzahl)
  from Verkäufe v, Zeit z
  where v.VerkDatum = z.Datum
  group by v.Produkt, v.Filiale, z.Jahr );
```

```
select v.Produkt, v.Filiale, sum(v.Anzahl)
from Verkäufe v
group by v.Produkt, v.Filiale
```

Wiederverwendung von Teil-Aggregaten

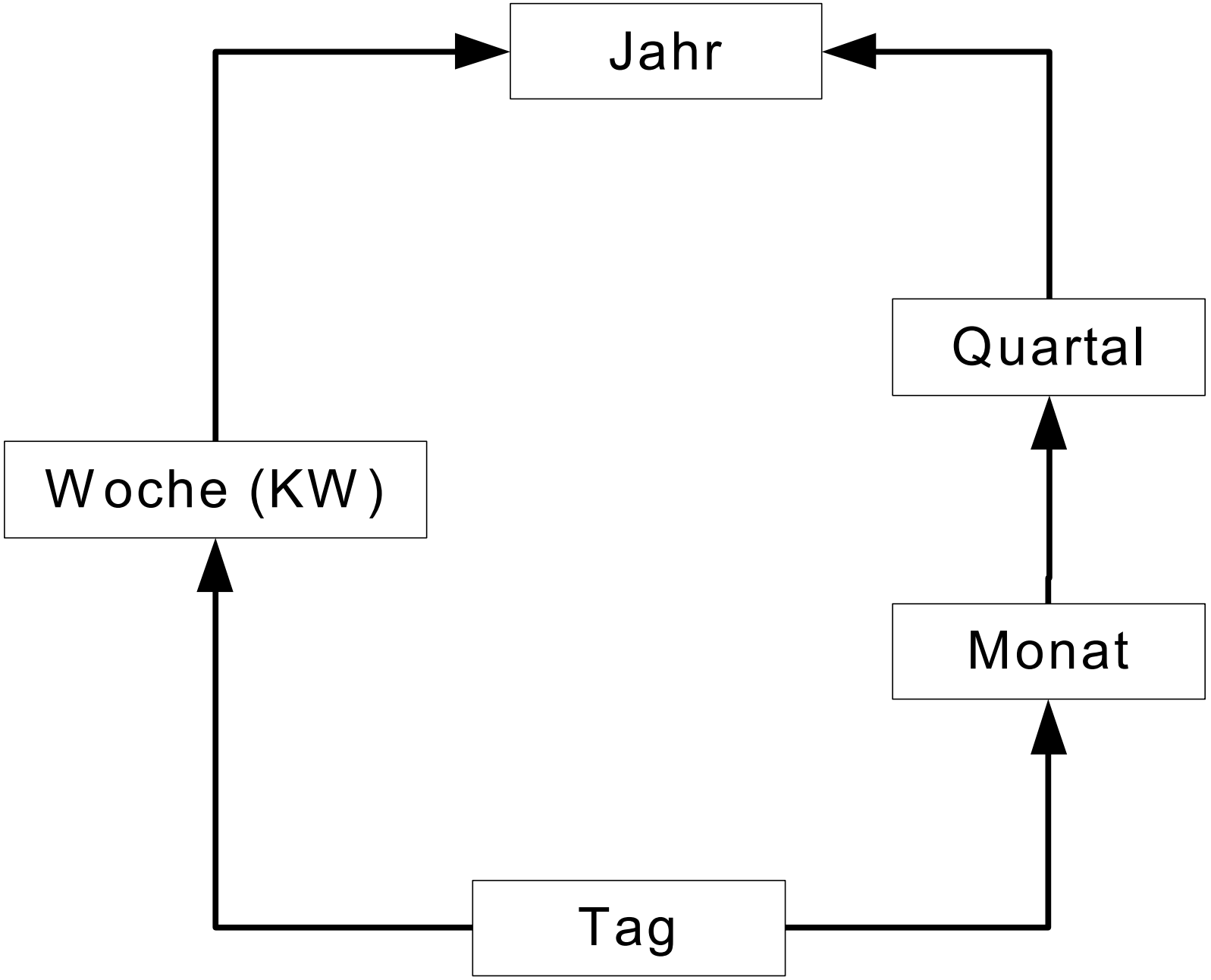
```
select v.Produkt, v.Filiale, sum(v.Anzahl)
from VerkäufeProduktFilialeJahr v
group by v.Produkt, v.Filiale

select v.Produkt, z.Jahr, sum(v.Anzahl)
from Verkäufe v, Zeit z
where v.VerkDatum = z.Datum
group by v.Produkt, z.Jahr
```

Die Materialisierungs-Hierarchie



- Teilaggregate T sind für eine Aggregation A wiederverwendbar wenn es einen gerichteten Pfad von T nach A gibt
- Also T → ..... → A
- Man nennt diese Materialisierungshierarchie auch einen Verband (Engl. *Lattice*)

Die Zeit-Hierarchie



Bitmap-Indexe

...	w_{18}	w_{19}	...
	18	**19**	
	0	0	
	1	0	
	1	0	
	0	0	
	0	1	
	...	...	

Kunden				
KundenNr	Name	wiealt	Geschlecht	...
007	Bond	43	m	...
4013	Mini	18	w	...
4315	Mickey	18	m	...
4711	Kemper	43	m	...
5913	Twiggy	19	w	...
...	...	...	...	...

G_m	G_w
m	**w**
1	0
0	1
1	0
1	0
0	1
...	...

- Optimierung durch Komprimierung der Bitmaps
- Ausnutzung der dünnen Besetzung
  - Runlength-compression
    - Grundidee: speichere jeweils die Länge der Nullfolgen zwischen zwei Einsen
  - Mehrmodus-Komprimierung:
    - bei langen Null/Einsfolgen speichere deren Länge
    - Sonst speichere das Bitmuster

Beispiel-Anfrage und Auswertung

```
select k.Name, ...
from Kunden k
where k.Geschlecht = 'w' and
        k.wiealt between 18 and 19;
```

$$(w_{18} \vee w_{19}) \wedge G_w$$

Bitmap-Operationen

w_{18}		w_{19}		$w_{18} \vee w_{19}$		G_w		$(w_{18} \vee w_{19}) \wedge G_w$
18		**19**		**18 .. 19**		**w**		**Ergebnis**
0		0		0		0		0
1	$\vee$	0	$=$	1	$\wedge$	1	$=$	**1**
1		0		1		0		0
0		0		0		0		0
0		1		1		1		**1**
...		...		...		...		...

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

Join-Index	
TID-V	TID-K
i	*II*
ii	*I*
iii	*II*
iv	*II*
v	*I*
vi	*II*
...	...

Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

bb. 17.17: Klassischer Join-Index

Bitmap-Join-Index

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

J_I	J_{II}	J_{III}	
0	1	...	
1	0	...	
0	1	...	...
0	1	...	
1	0	...	
0	1	...	
...	...	...	

Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

Join-Index	
TID-V	TID-K
i	*II*
ii	*I*
iii	*II*
iv	*II*
v	*I*
vi	*II*
...	...

Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

bb. 17.17: Klassischer Join-Index

Bitmap-Join-Index

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

J_I	J_{II}	J_{III}
0	1	...
1	0	...
0	1	...
0	1	...
1	0	...
0	1	...
...	...	...

...

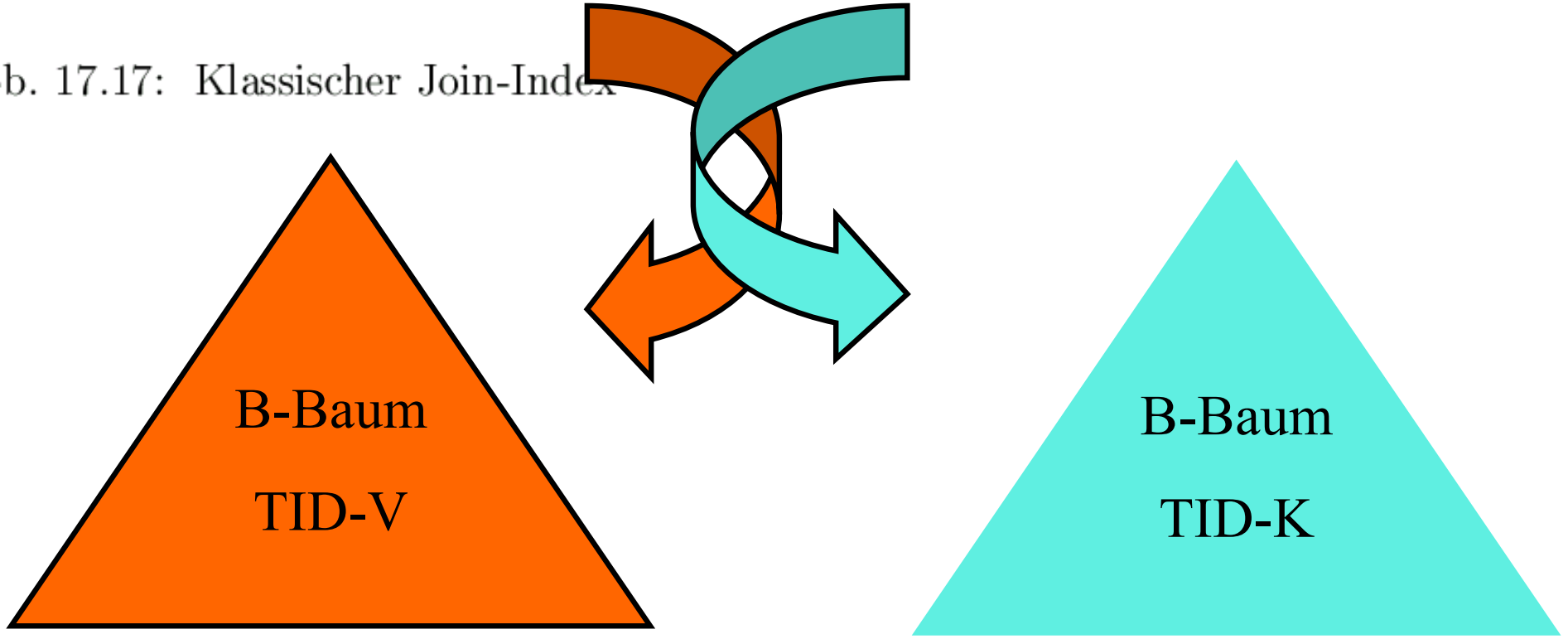
Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

Join-Index	
TID-V	TID-K
i	*II*
ii	*I*
iii	*II*
iv	*II*
v	*I*
vi	*II*
...	...

Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

bb. 17.17: Klassischer Join-Index



(i,II)(ii,I)(iii,II)(iv,II)(v,I)(vi,II)

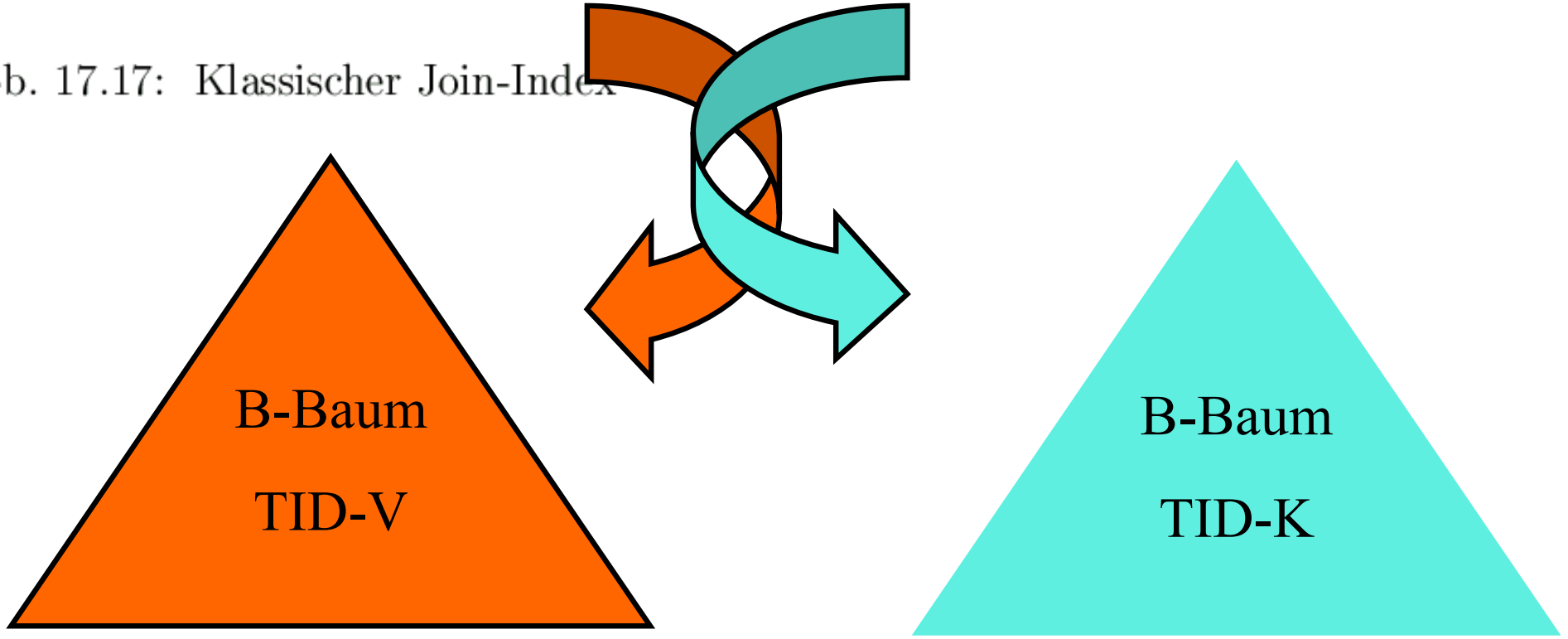
(I,i)(I,v)(II,i)(II,iii)(II,iv)(II,vi)

Verkäufe		
TID	...	KundenNr
i	...	007
ii	...	4711
iii	...	007
iv	...	007
v	...	4711
vi	...	007
...	...	...

Join-Index	
TID-V	TID-K
i	*II*
ii	*I*
iii	*II*
iv	*II*
v	*I*
vi	*II*
...	...

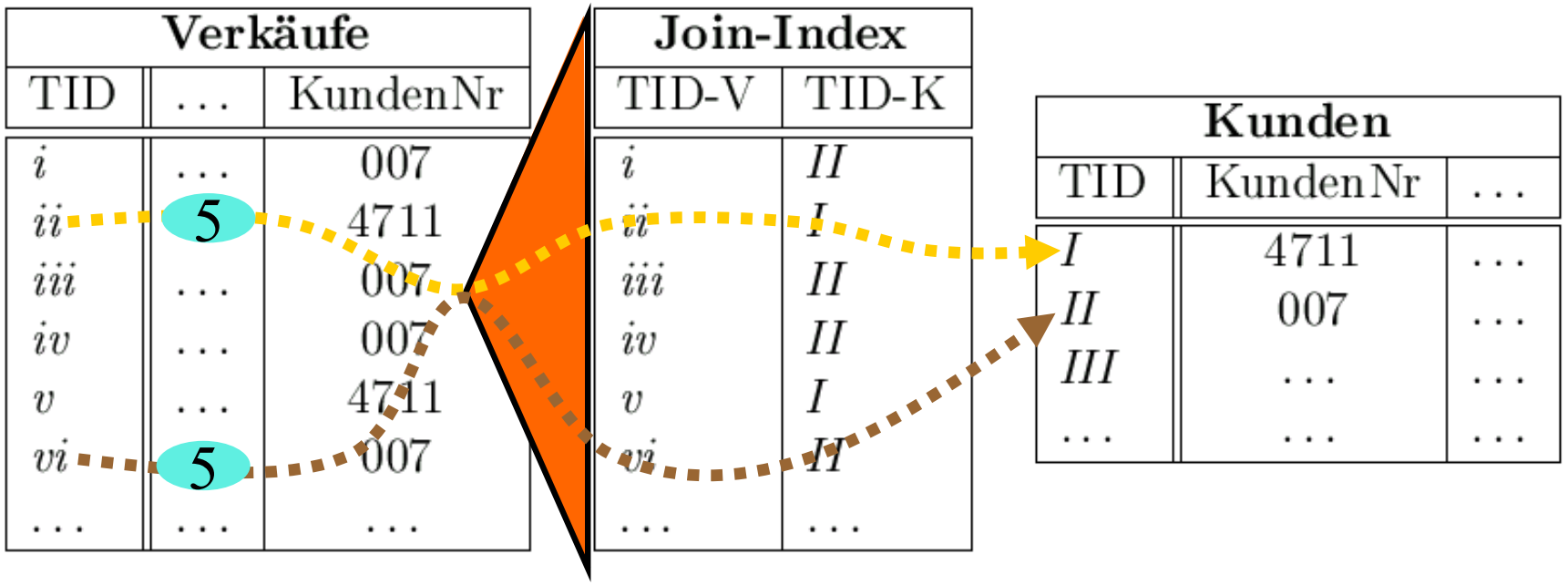
Kunden		
TID	KundenNr	...
I	4711	...
II	007	...
III	...	...
...	...	...

bb. 17.17: Klassischer Join-Index

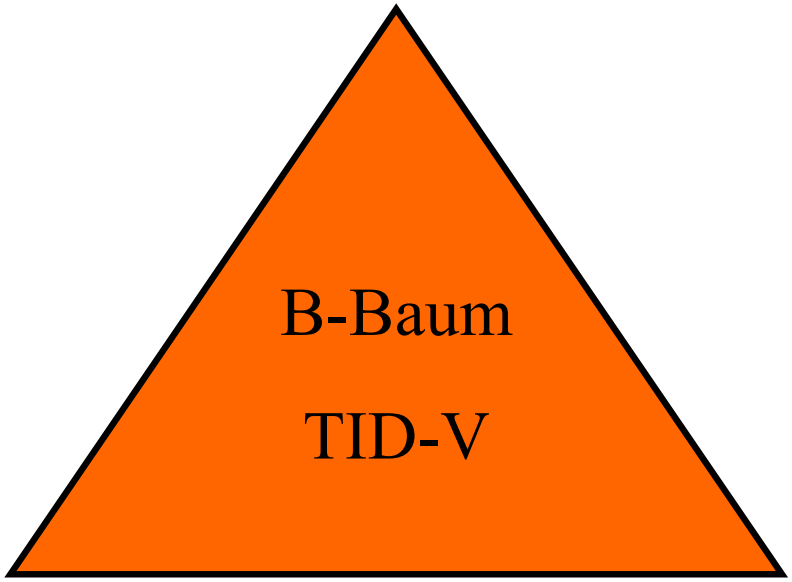


(i,II)(ii,I)(iii,II)(iv,II)(v,I)(vi,II)

(I,i)(I,v)(II,i)(II,iii)(II,iv)(II,vi)

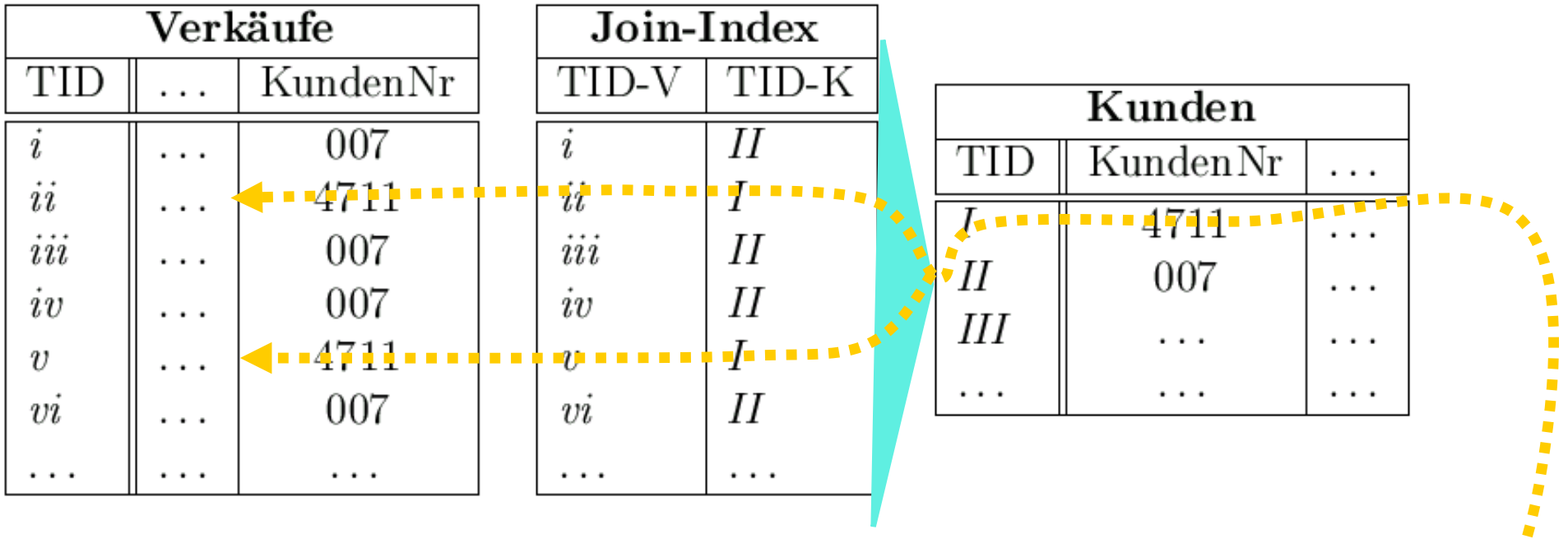


bb. 17.17: Klassischer Join-Index

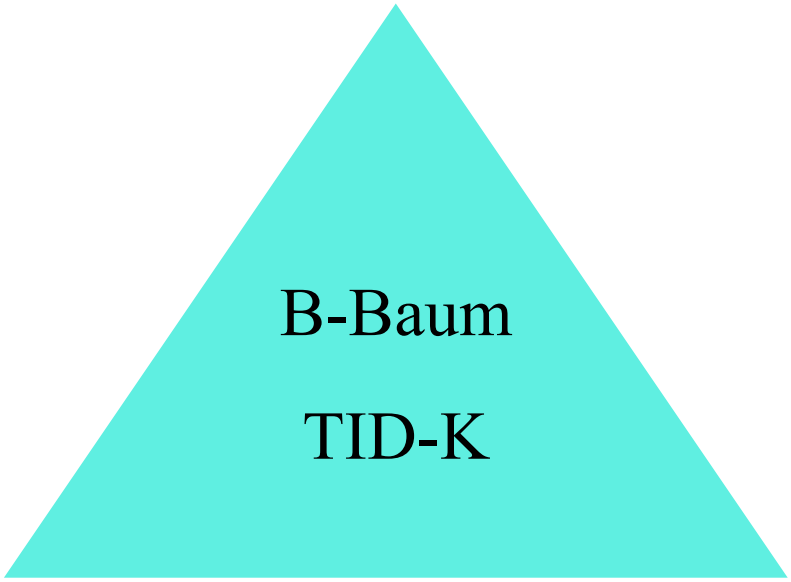


(i,II)(ii,I)(iii,II)(iv,II)(v,I)(vi,II)

```
Select k.*
From Verkäufe v, Kunden k
Where v.ProduktID = 5 And
      v.KundenNr = k.KundenNr
```



bb. 17.17: Klassischer Join-Index



Select v.*
From Verkäufe v, Kunden k
Where k.KundenNr = 4711 and
v.KundenNr = k.KundenNr

(I,i)(I,v)(II,i)(II,iii)(II,iv)(II,vi)...

Verkäufe					
VerkDatum	Filiale	Produkt	Anzahl	Kunde	Verkäufer
25-Jul-00	Passau	1347	1	4711	825
...	...	...	...	...	...

Filialen			
Filialenkennung	Land	Bezirk	...
Passau	D	Bayern	...
...	...	...	...

Kunden			
KundenNr	Name	wiealt	...
4711	Kemper	43	...
...	...	...	...

Verkäufer					
VerkäuferNr	Name	Fachgebiet	Manager	wiealt	...
825	Handyman	Elektronik	119	23	...
...	...	...	...	...	...

Zeit								
Datum	Tag	Monat	Jahr	Quartal	KW	Wochentag	Saison	...
...	...	...	...	...	...	...	...	...
25-Jul-00	25	Juli	2000	3	30	Dienstag	Hochsommer	...
...	...	...	...	...	...	...	...	...
18-Dec-01	18	Dezember	2001	4	52	Dienstag	Weihnachten	...
...	...	...	...	...	...	...	...	...

Produkte					
ProduktNr	Produkttyp	Produktgruppe	Produkthauptgruppe	Hersteller	...
1347	Handy	Mobiltelekom	Telekom	Siemens	...
...	...	...	...	...	...

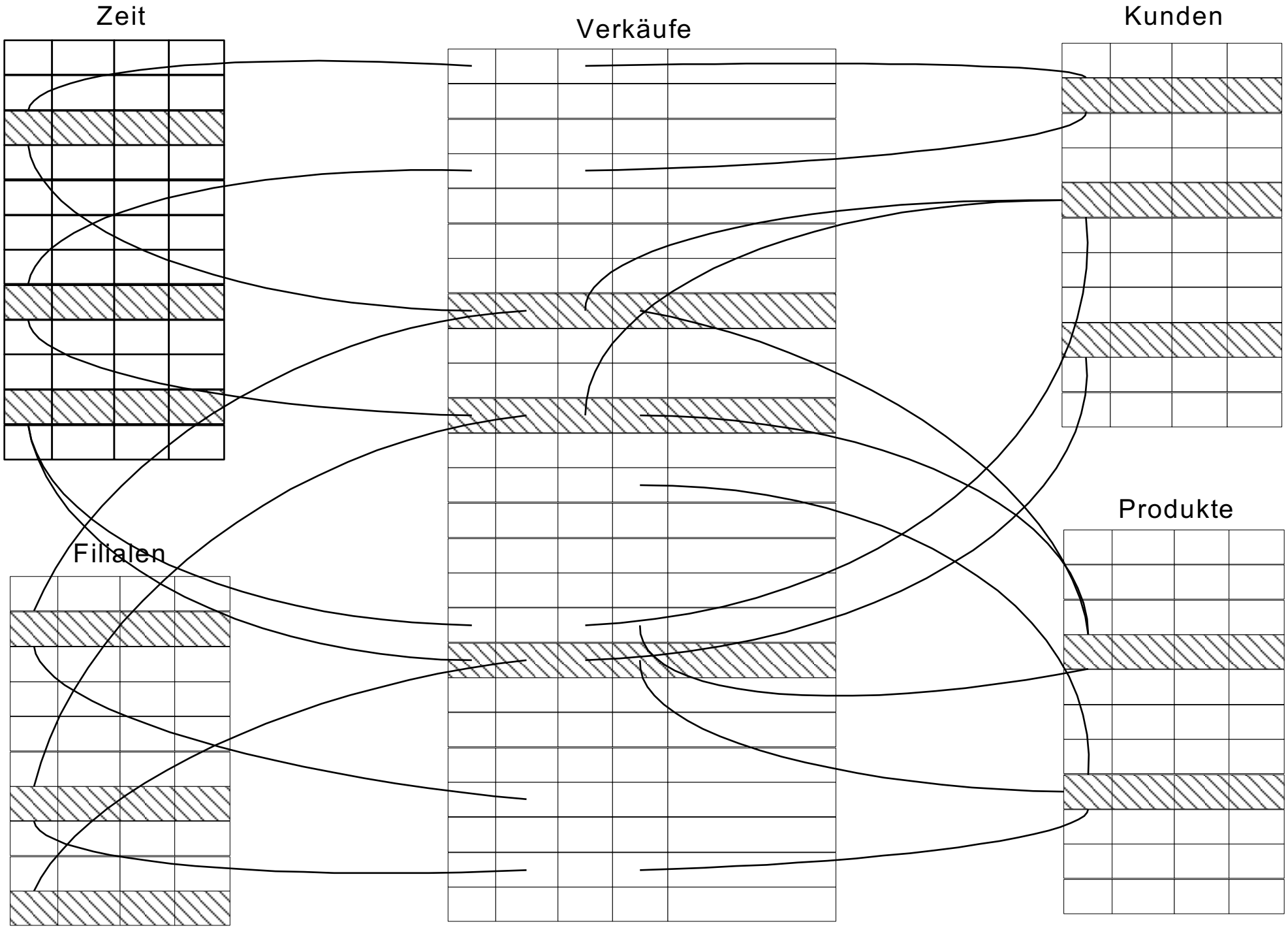
Beispielanfrage auf dem Sternschema: Stern-Verbund -- Star Join

```
select sum(v.Anzahl), p.Hersteller
from Verkäufe v, Filialen f, Produkte p, Zeit z, Kunden k
where z.Saison = 'Weihnachten' and
      z.Jahr = 2001 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 p.Hersteller;
```

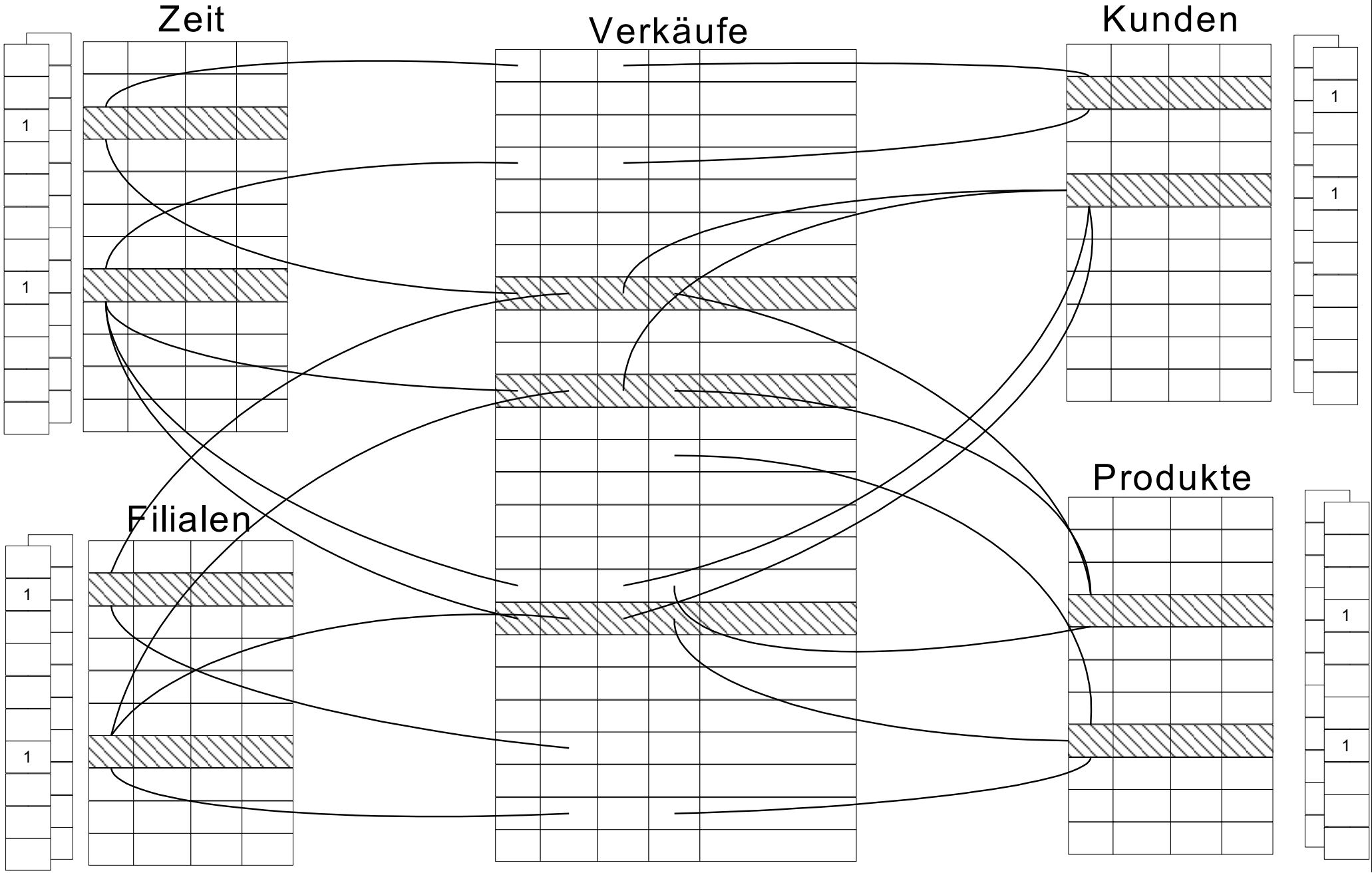
Einschränkung der Dimensionen

Join-Prädikate

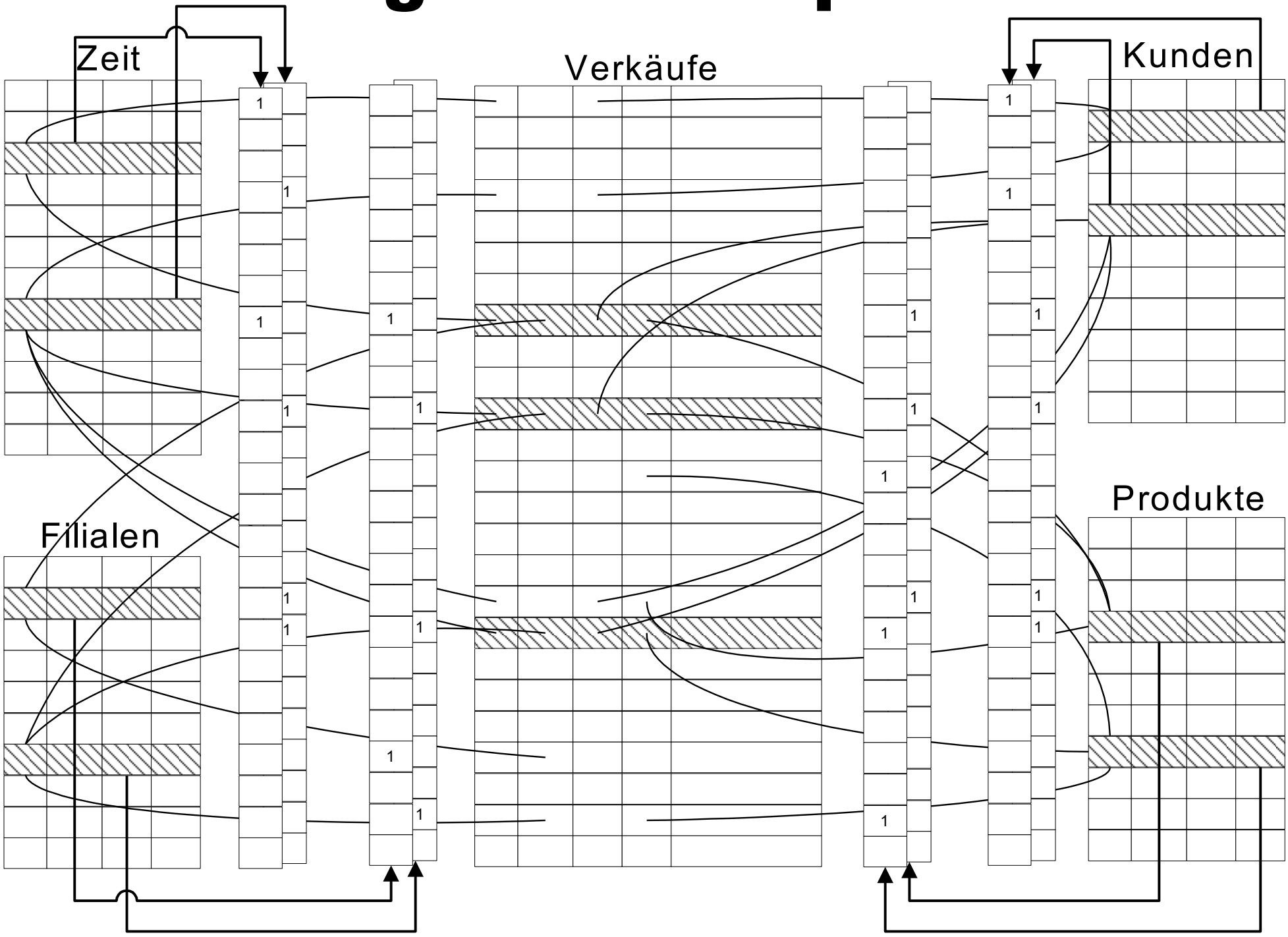
Illustration des Star Join



Bitmap-Indexe für die Dimensions-Selektion



Ausnutzung der Bitmap-Join-Indexe



Eine weitere Join-Methode: DiagJoin

- Für 1:N-Beziehungen
- Daten sind zeitlich geballt (clustered)
- Beispiel
  - Order
  - Lineitem
  - Order A Lineitem
  - Die Lineitems (Bestellpositionen) einer Order (Bestellung) kommen zeitlich kurz hintereinander
- Grundidee des DiagJoins besteht darin, synchron über die beiden Relationen zu laufen
- Die Orders werden in einem Fenster gehalten

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
…	…	…	…

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
…	…	…	…

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
…	…	…	…

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
…	…	…	…

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
4711	5	Quirl	…
…	…	…	…

DiagJoin

Order	
Customer	Order#
Kemper	4711
Maier	5645
Müller	7765
Hummer	9876
Kaller	9965
Lola	3452
Junker	1232
…	…

LineItem			
Order#	Position	Produkt	Preis
4711	1	PC	…
5645	1	Laptop	…
4711	2	Drucker	…
4711	3	Toner	…
5645	2	Hub	…
7765	1	Fax	…
4711	4	Papier	…
5645	3	Handy	…
7765	2	Mixer	…
9876	1	Handy	…
4711	5	Quirl	…
…	…	…	…

Muss zwischengespeichert werden und „nachbearbeitet“ werden.

Anforderungen an den DiagJoin

- 1:N Beziehung
- Die „1"-er Tupel sind in etwa derseleben Reihenfolge gespoeichert worden wie die „N"-er Tupel
- Die Tupel werden in der „time-of-creation"-Reihenfolge wieder von der Platte gelesen (full table scan)
- Die referentielle Integrität muss gewährleistet sein
- Das Fenster muss so groß sein, dass kaum Tupel nachbearbeitet werden müssen
- Nachbearbeitung bedeutet
  - Tupel auf dem Hintergrundspeicher speichern
  - Den zugehörigen Joinpartner via Index auffinden
  - Also ist ein Index auf Order.Order# hierfür notwendig
    - Nicht für die erste Phase des DiagJoins

Weitere Decision-Support Anfrage-Typen

- Top N-Anfragen
  - Ich will nur die N besten Treffer erhalten und ncht alle 5 Millionen
  - Muss bei der Anfrageoptimierung berücksichtigt werden
- Online Aggregation
  - Man berechnet das Ergebnis approximativ
  - Je länger die Anfrage läuft, desto genauer wird das Ergebnis

Top N-Anfragen

Select A.*
From Angestellte A, Abteilungen abt
Where A.Abteilung = abt.AbteilungsNr and abt.Ort = Passau
Order by A.Gehalt
Stop after 20

Top N-Anfragen

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

WohnLage	
Ort	Lage
Grünwald	München-Süd
Unterföhring	München-Nord
Ismaning	München-Nord
Garching	München-Nord
Bogenhausen	München-City
Nymphenburg	München-City

```
select m.Ort, m.Miete + k.Beitrag as Kosten
from Mietspiegel m, Kindergarten k
where m.Ort = k.Ort
order by Kosten
fetch first 1 rows only
```

Ranking in DB2

```
with KostenVergleich as (select m.Ort, m.Miete, k.Beitrag
                         from Mietspiegel m, Kindergarten k
                         where m.Ort=k.Ort)

select r.Ort, r.Rang
from (select v.Ort,
             RANK() over (order by v.Miete + 7 * v.Beitrag) as Rang
      from KostenVergleich v) as r
where r.Rang <= 3
```

Wir erhalten folgendes Ergebnis:

Top-3-Ergebnis	
Ort	Rang
Grünwald	1
Unterföhring	1
Bogenhausen	3
Ismaning	3


Hier klicken Blick ins Buch!
DIRK STERMANN
6 österreicher unter den ersten 5
ROMAN EINER ENTPIEFKENISIERUNG
Ullstein

Window Funktionen in SQL

```
select Ort, Zeit, Wert, abs(Wert − (avg(Wert) over w)) / (stddev(Wert) over w)
from Messungen
window w as (
   partition by Ort
   order by Zeit
   range between 5 preceding and 5 following)
```

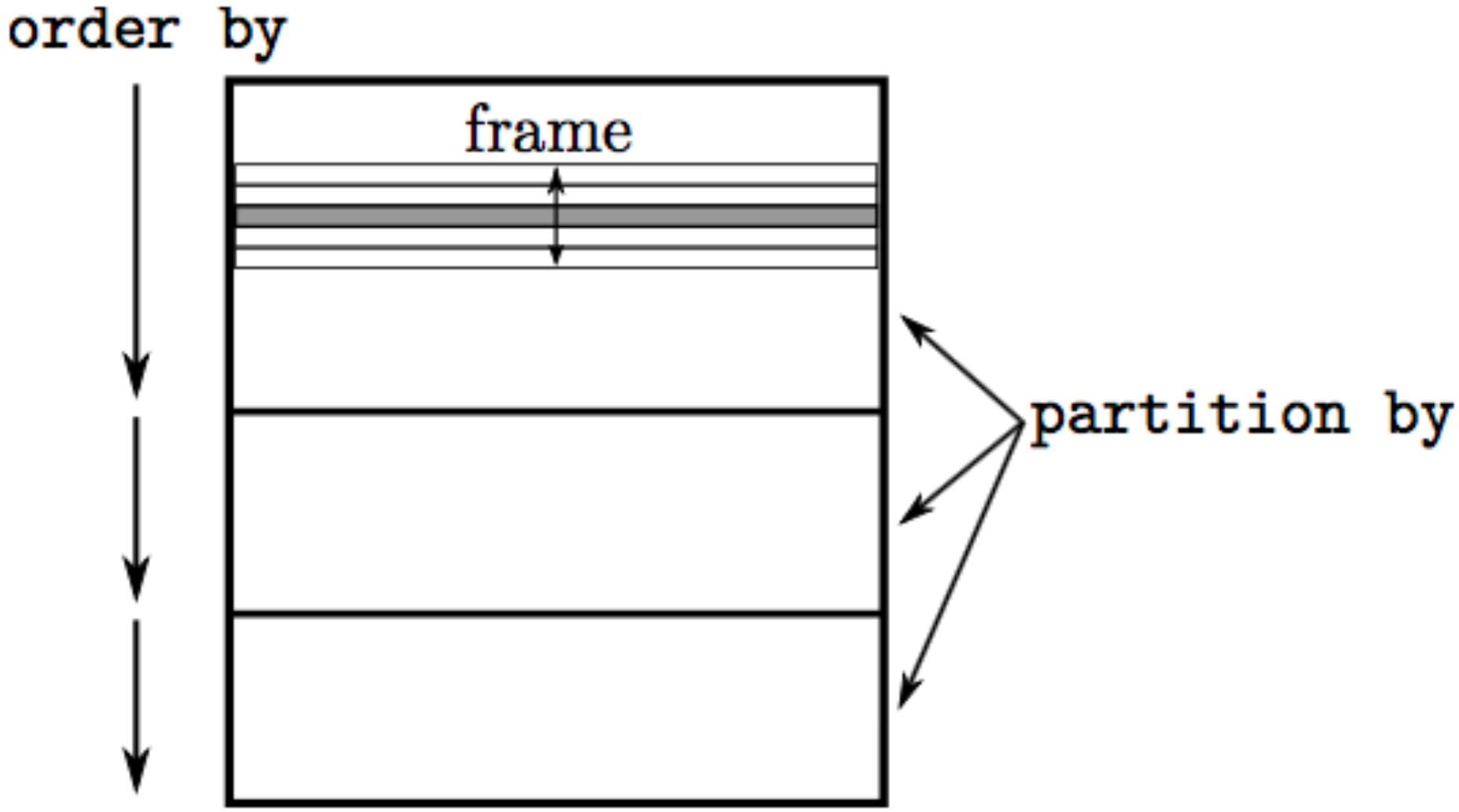
Komplexe Anfrage

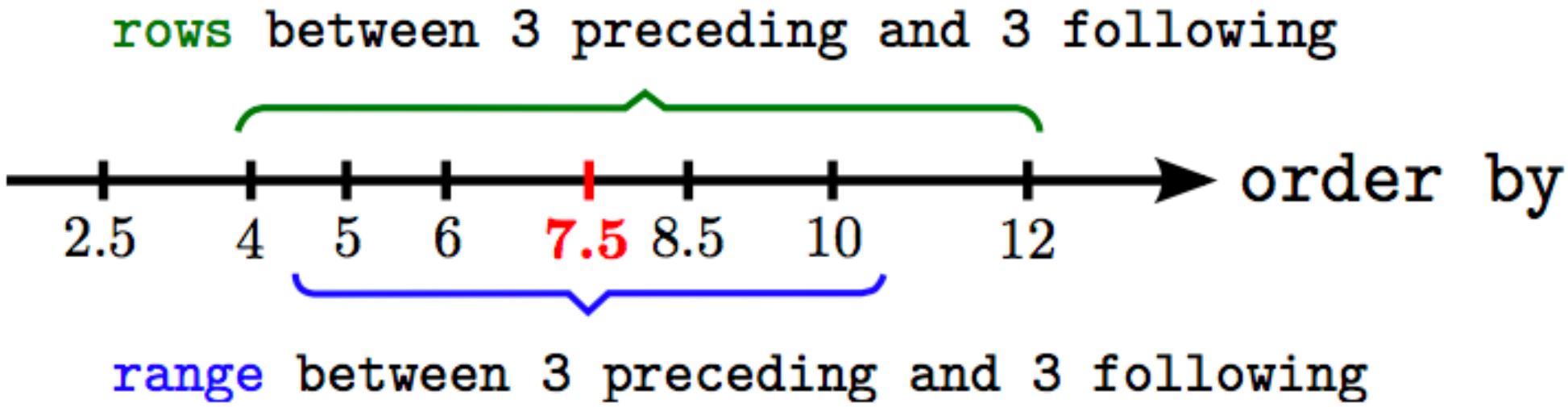
```
select Ort, Zeit, Wert, abs(Wert−
  (select avg(Wert)
   from Messungen m2
   where m2.Zeit between m.Zeit−5 and m.Zeit+5
         and m.Ort = m2.Ort))
/ (select stddev(Wert)
   from Messungen m3
   where m3.Zeit between m.Zeit−5 and m.Zeit+5
         and m.Ort = m3.Ort)
from Messungen m
```

Lag: Vorhergehendes Tupel im Frame

```
select Zeit, Wert,
       ((Wert − lag(Wert) over w) /
        (Zeit − lag(Zeit) over w)) as Änderungsrate
from Messungen
window w as (order by Zeit)
```

Zusammenhang: Frames / Partition / Sortierung




rows between 3 preceding and 3 following
order by
2.5
4
5
6
7.5
8.5
10
12
range between 3 preceding and 3 following

Medaillengewinner (in schön;-)

```
select Name, (case RangPlatz
                    when 1 then 'Gold'
                    when 2 then 'Silber'
                    else 'Bronze' end)
from (select Name, rank() over w as RangPlatz
      from Resultate
      window w as (order by Punkte desc))
where RangPlatz <= 3
```

Frame Begrenzungen

Neben den **preceding** und **following**-Spezifikationen können Frame-Begrenzungen auch wie folgt angegeben werden:

- **current row**: Hiermit kann das aktuelle Tupel inklusive seiner Peers angegeben werden. Im **range**-Modus werden zusätzlich alle Peers Teil des Frames.
- **unbounded preceding**: Der Frame enthält alle dem aktuellen Tupel vorausgehenden Tupel der Partition.
- **unbounded following**: Der Frame endet erst beim letzten Tupel der Partition.

```
select KundenID, Bestelldatum, sum(Preis) over
    (partition by KundenID,
                  extract(month from Bestelldatum),
                  extract(year from Bestelldatum)
     order by Bestelldatum
     range between unbounded preceding and current row)
from Verkäufe;
```

Explizite Windows-Klausur → Wiederverwendbarkeit

```
select min(Wert) over w1, max(Wert) over w1,
       min(Wert) over w2, max(Wert) over w2
from Messungen
window
  w1 as (order by Zeit
            range between 5 preceding and 5 following),
  w2 as (order by Zeit
            range between 3 preceding and 3 following)
```

Ranking innerhalb von Untergruppen

```
with KostenLageVergleich as (
   select m.Ort, m.Miete, k.Beitrag, l.Lage
   from Mietspiegel m, Kindergarten k, WohnLage l
   where m.Ort = k.Ort and k.Ort = l.Ort

select k.Ort, k.Lage, k.Miete + 3 * k.Beitrag as Kosten,
        rank() over (partition by k.Lage
                     order by k.Miete + 3 * k.Beitrag asc) as LageRang
from KostenLageVergleich k
order by k.Lage, LageRang
```

Die Partitionierung der (virtuellen) Relation *KostenLageVergleich* erfogt gemä des Attributs *Lage*. Als Ergebnis der Anfrage erhalten wir folgende Relation:

Ergebnis			
Ort	Lage	Kosten	LageRang
Bogenhausen	München-City	1900	1
Nymphenburg	München-City	2400	2
Unterföhring	München-Nord	1000	1
Ismaning	München-Nord	1500	2
Garching	München-Nord	1550	3
Grünwald	München-Süd	1400	1

Threshold-Algorithmus zur Auswertung von Top-n-Anfragen (3)

Mietspiegel	
Ort	Miete
Garching	**800**
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	**-100**
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis: Phase 1	
Ort	m.Miete + k.Beitrag
Threshold	**700**
Garching	1050
Grünwald	1600

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	**900**

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	**0**

Threshold-Algorithmus zur Auswertung von Top-n-Anfragen (3)

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	**900**
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	**0**
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis: Phase 2	
Ort	m.Miete + k.Beitrag
Threshold	**900**
Unterföhring	1000
Garching	1050
Ismaning	1100
Grünwald	1600

Threshold-Algorithmus zur Auswertung von Top_n-Anfragen

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	**1000**
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	**100**
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis: Phase 3	
Ort	m.Miete + k.Beitrag
Unterföhring	1000
Garching	1050
Ismaning	1100
Threshold	**1100**
Grünwald	1600
Bogenhausen	1700

No-Random-Access-Algorithmus

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis	
Ort	Kosten
Garching	700↗
Grünwald	700↗

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis	
Ort	Kosten
Garching	800↗
Grünwald	800↗
Unterföhring	900↗
Ismaning	900↗

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200

Zwischenergebnis	
Ort	Kosten
Garching	900↗
Grünwald	900↗
Unterföhring	**1000**√
Ismaning	1000↗

No-Random-Access-Algorithmus

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

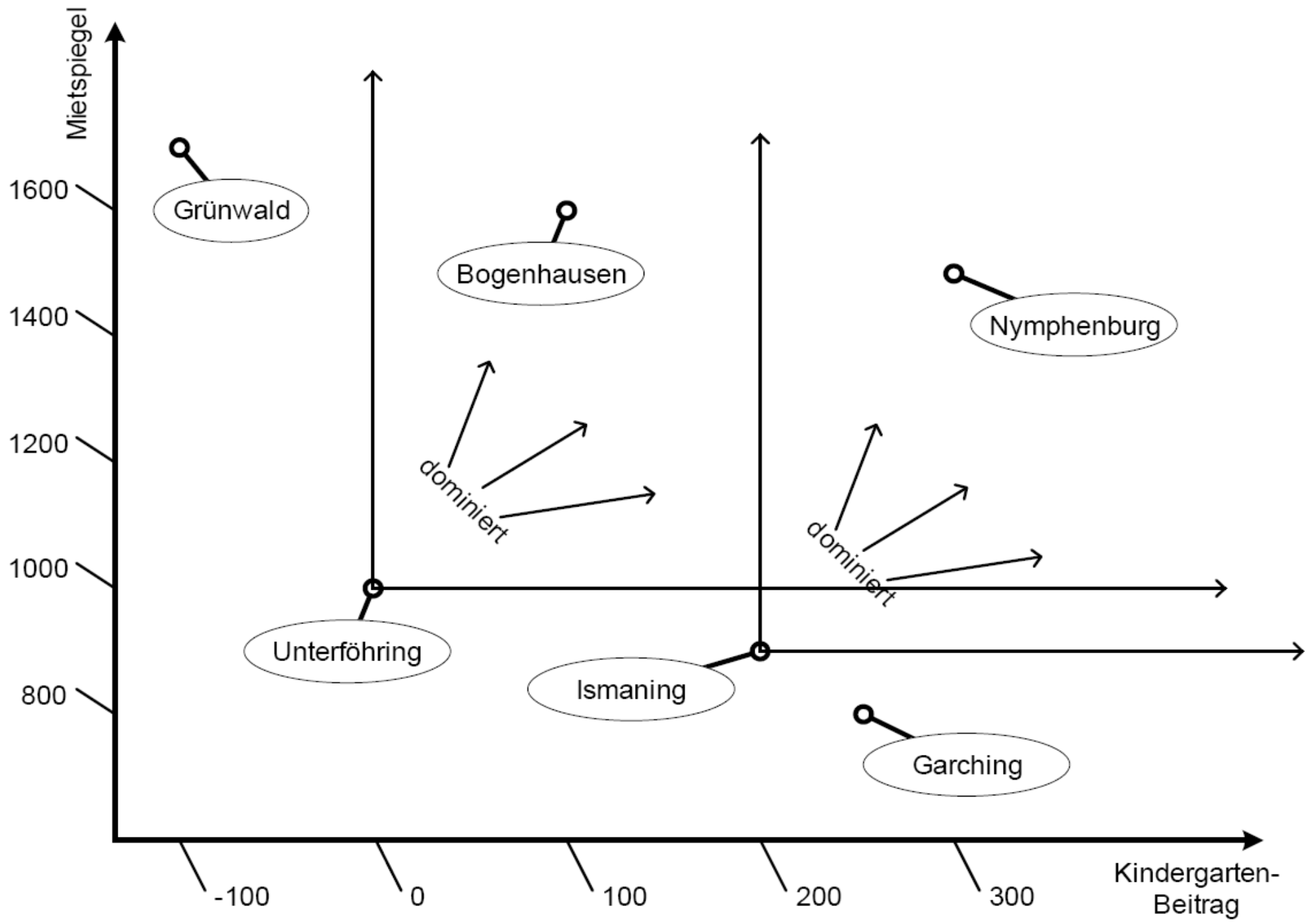
Zwischenergebnis	
Ort	Kosten
Garching	900↗
Grünwald	900↗
Unterföhring	**1000**√
Ismaning	1000↗
Bogenhausen	1100↗

Mietspiegel	
Ort	Miete
Garching	800
Ismaning	900
Unterföhring	1000
Nymphenburg	1500
Bogenhausen	1600
Grünwald	1700

Kindergarten	
Ort	Beitrag
Grünwald	-100
Unterföhring	0
Bogenhausen	100
Ismaning	200
Garching	250
Nymphenburg	300

Zwischenergebnis	
Ort	Kosten
Unterföhring	**1000**√
Garching	1000↗
Ismaning	**1100**√
Grünwald	1400↗
Bogenhausen	1600↗
Nymphenburg	1700↗

Skyline / Pareto-Optimum



Skyline in SQL

```
with KostenVergleich as (select m.Ort, m.Miete, k.Beitrag
                         from Mietspiegel m, Kindergarten k
                         where m.Ort=k.Ort)

select k.Ort
from KostenVergleich k
skyline of k.Miete min, k.Beitrag min
```

Skyline in Standard-SQL

```
select k.Ort
from KostenVergleich k
where not exists
        (select * from KostenVergleich dom
         where dom.Miete <= k.Miete and dom.Beitrag <= k.Beitrag and
                 (dom.Miete < k.Miete or dom.Beitrag < k.Beitrag))
```

Online-Aggregation

Select abt.Ort, avg(A.Gehalt)
From Angestellte A, Abteilungen abt
Where A.Abteilung = abt.AbteilungsNr
Group by abt.Ort

Data Mining

Klassifikation

Assoziationsregeln

Clustering

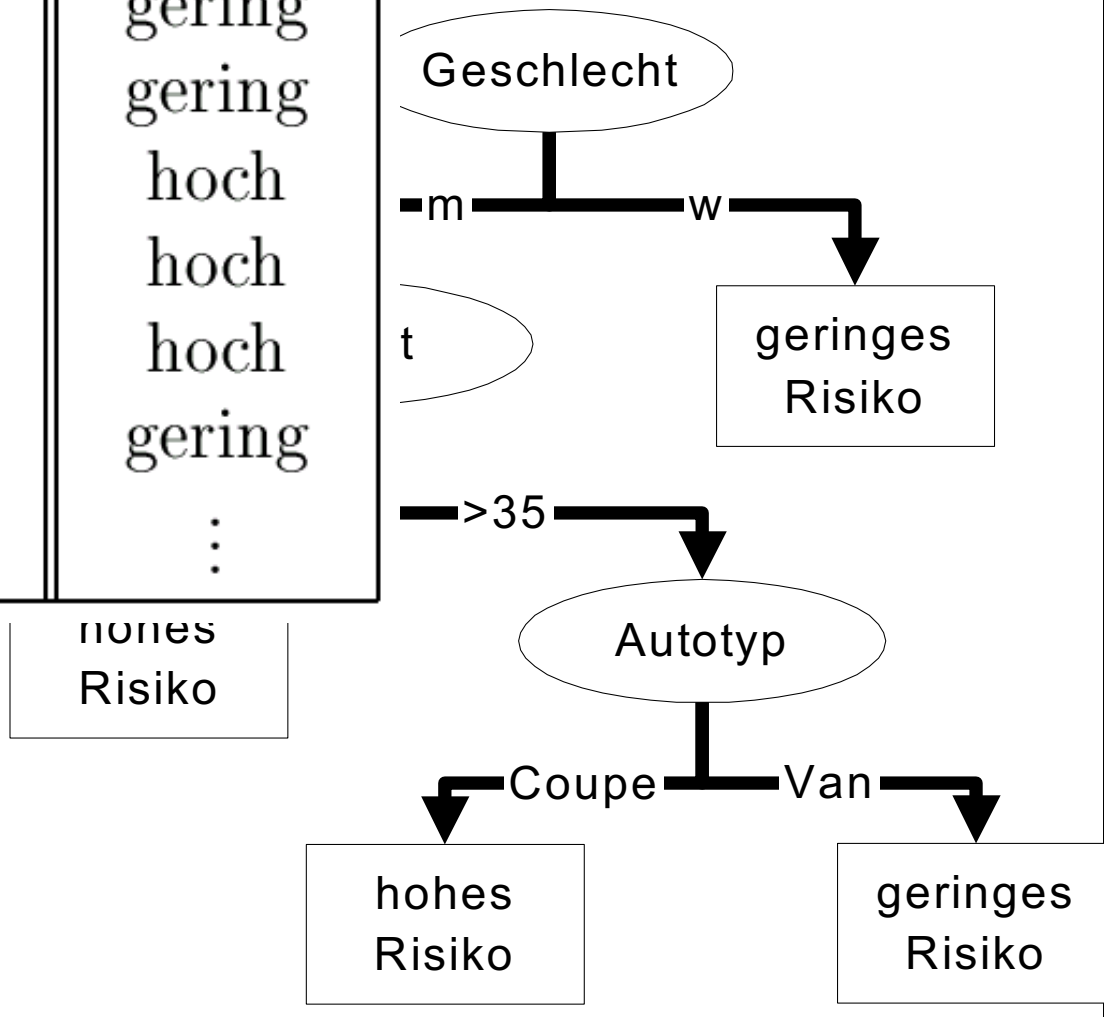
Klassifikationsregeln

- Vorhersageattribute
  - V1, V2, ..., Vn
- Vorhergesagtes Attribut A
- Klassifikationsregel
  - P1(V1) $\wedge$ P2(V2) $\wedge$ ... $\wedge$ Pn(Vn) ➔ A = c
  - Prädikate P1, P2, .., Pn
  - Konstante c
- Beispielregel

(wieAlt>35) $\wedge$ (Geschlecht =`m´) $\wedge$ (Autotyp=`Coupé´) ➔ (Risiko=´hoch´)

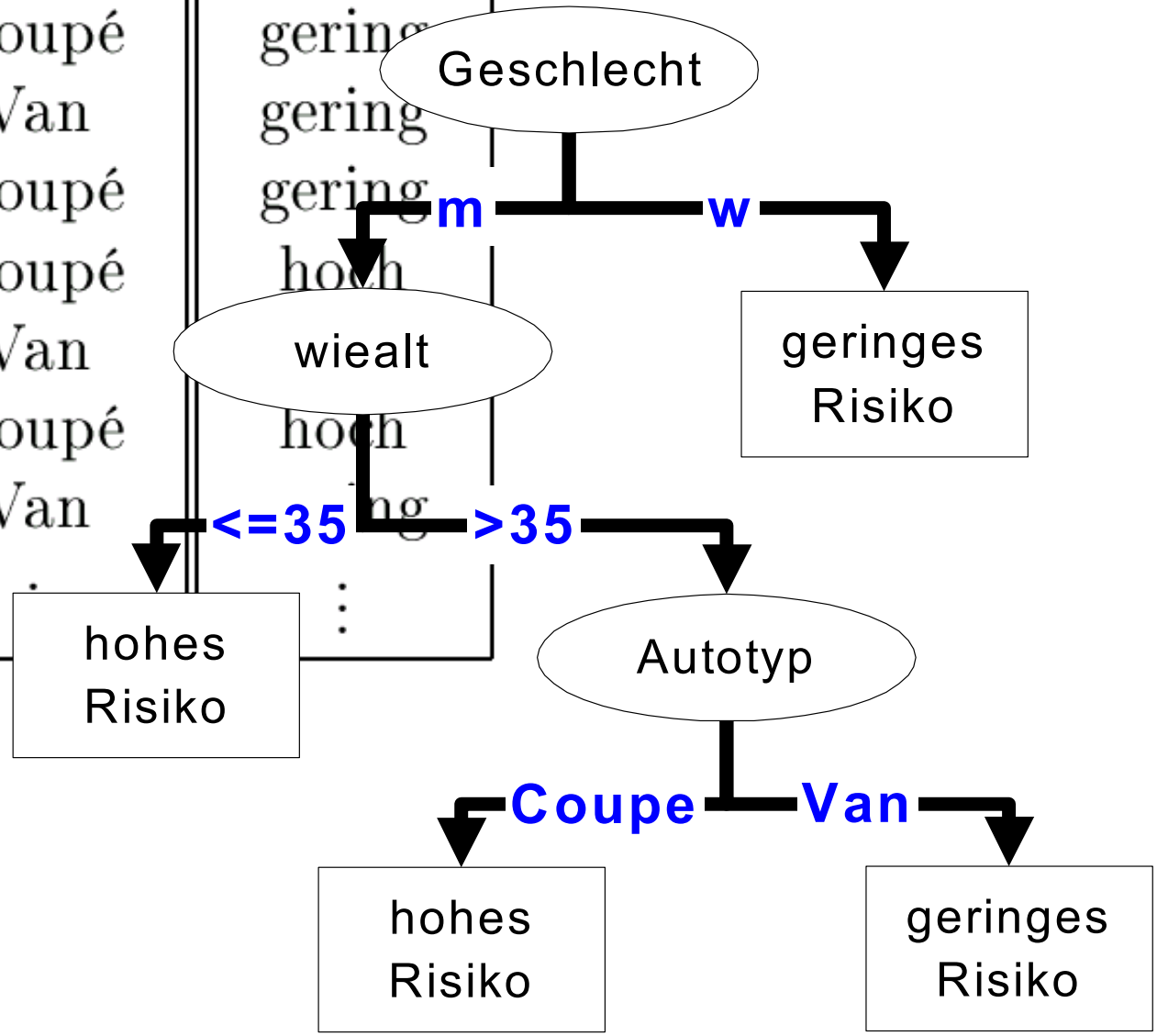
Klassifikations/Entscheidungsbaum

Schadenshöhe			
wiealt	Geschlecht	Autotyp	Schäden
45	w	Van	gering
18	w	Coupé	gering
22	w	Van	gering
38	w	Coupé	gering
19	m	Coupé	hoch
24	m	Van	hoch
40	m	Coupé	hoch
40	m	Van	gering
⋮	⋮	⋮	⋮



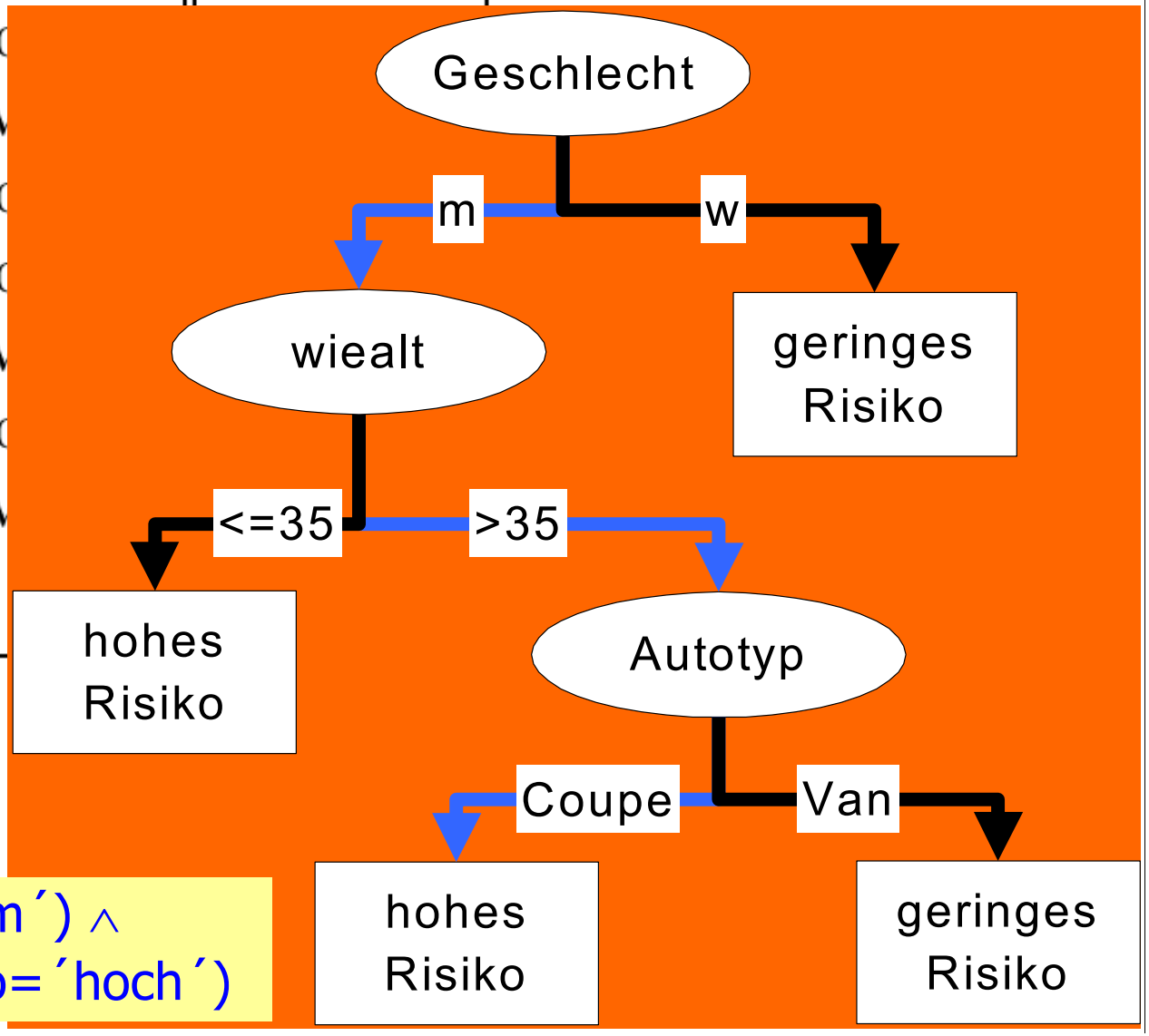
Klassifikations/Entscheidungsbaum

Schadenshöhe			
wiealt	Geschlecht	Autotyp	Schäden
45	w	Van	gering
18	w	Coupé	gering
22	w	Van	gering
38	w	Coupé	gering
19	m	Coupé	hoch
24	m	Van	gering
40	m	Coupé	hoch
40	m	Van	gering
⋮	⋮	⋮	⋮



Klassifikations/Entscheidungsbaum

Schadenshöhe			
wiealt	Geschlecht	Autotyp	Schäden
45	w	Van	gering
18	w	Co	
22	w	V	
38	w	Co	
19	m	Co	
24	m	V	
40	m	Co	
40	m	V	
⋮	⋮		



(wieAlt>35) ∧ (Geschlecht =`m´) ∧
(Autotyp=`Coupé´) ➔ (Risiko=´hoch´)

Wie werden Entscheidungs/ Klassifikationsbäume erstellt

- Trainingsmenge
  - Große Zahl von Datensätzen, die in der Vergangenheit gesammelt wurden
  - Sie dient als Grundlage für die Vorhersage von „neu ankommenden" Objekten
  - Beispiel: neuer Versicherungskunde wird gemäß dem Verhalten seiner „Artgenossen" eingestuft
- Rekursives Partitionieren
  - Fange mit einem Attribut an und spalte die Tupelmenge
  - Jede dieser Teilmengen wird rekursiv weiter partitiniert
  - Bis nur noch gleichartige Objekte in der jeweiligen Partition sind

Top-Down Klassifikationsbaum-Aufbau

- Eingabe: Knoten n, Partition D, Zerlegungsmethode S
- Ausgabe: Klassifikationsbaum für D, Wurzel n

- BuildTree(n,D,S)
  - Wende S auf D an und finde die richtige Zerlegung
  - Wenn eine gute Partitionierung gefunden ist
    - Kreiere zwei Kinder n1 und n2
    - Partitioniere D in D1 und D2
    - BuildTree(n1,D1,S)
    - BuildTree(n2,D2,S)

Assoziationsregeln

- Beispielregel
  - Wenn jemand einen PC kauft, dann kauft er/sie auch einen Drucker
- Confidence
  - Dieser Wert legt fest, bei welchem Prozentsatz der Datenmenge, bei der die Voraussetzung (linke Seite) erfüllt ist, die Regel (rechte Seite) auch erfüllt ist.
  - Eine Confidence von 80% für unsere Beispielregel sagt aus, dass vier Fünftel der Leute, die einen PC gekauft haben, auch einen Drucker dazu gekauft haben.
- Support
  - Dieser Wert legt fest, wieviele Datensätze überhaupt gefunden wurden, um die Gültigkeit der Regel zu verifizieren.
  - Bei einem Support von 1% wäre also jeder Hundertste Verkauf ein PC zusammen mit einem Drucker.

VerkaufsTransaktionen	
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

Verkaufstransaktionen Warenkörbe

- Finde alle Assoziationsregeln L ➔ R
  - mit einem Support größer als minsupp

    und
  - einer Confidence von mindestens minconf
- Dazu sucht man zunächst die sogenannten frequent itemsets, also Produktmengen, die in mindestens minsupp der Einkaufswägen/ Transaktionen enthalten sind
- Der A Priori-Algorithmus basiert auf der Erkenntnis, dass alle Teilmengen eines FI auch FIs sein müssen

A Priori Algorithmus

für alle Produkte

überprüfe ob es ein frequent itemset ist, also in mindestens minsupp Einkaufswägen enthalten ist

k:=1

iteriere solange

für jeden *frequent itemset* I_k mit k Produkten

generiere alle *itemsets* I_{k+1} mit k+1 Produkten und $I_k \subset I_{k+1}$

lies alle Einkäufe einmal (sequentieller Scan auf der Datenbank) und überprüfe, welche der (k+1)-elementigen *itemset*-Kandidaten mindestens minsupp mal vorkommen

k:=k+1

bis keine neuen frequent itemsets gefunden werden

A Priori-Algorithmus

VerkaufsTransaktionen	
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

Minsupp=3

Disqua-lifiziert

Zwischenergebnisse	
FI-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}	

A Priori-Algorithmus

VerkaufsTransaktionen	
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

Zwischenergebnisse	
FI-Kandidat	Anzahl
{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}	2
{Drucker, Papier, Toner}	3
{Drucker, PC, Toner}	2
{Papier, PC, Toner}	2

Ableitung von Assoziationsregeln aus den *frequent itemsets*

- Betrachte jeden FI mit hinreichen viel *support*
- Bilde alle nicht-leeren Teilmengen L ⊂ FI und untersuche die Regel
  - L ➔ FI – L
  - Die Confidence dieser Regel berechnet sich als
    - Condicence(L ➔ FI – L) = support(FI) / support(L)
    - Wenn die Confidence ausreicht, also > minconf ist, behalte diese Regel

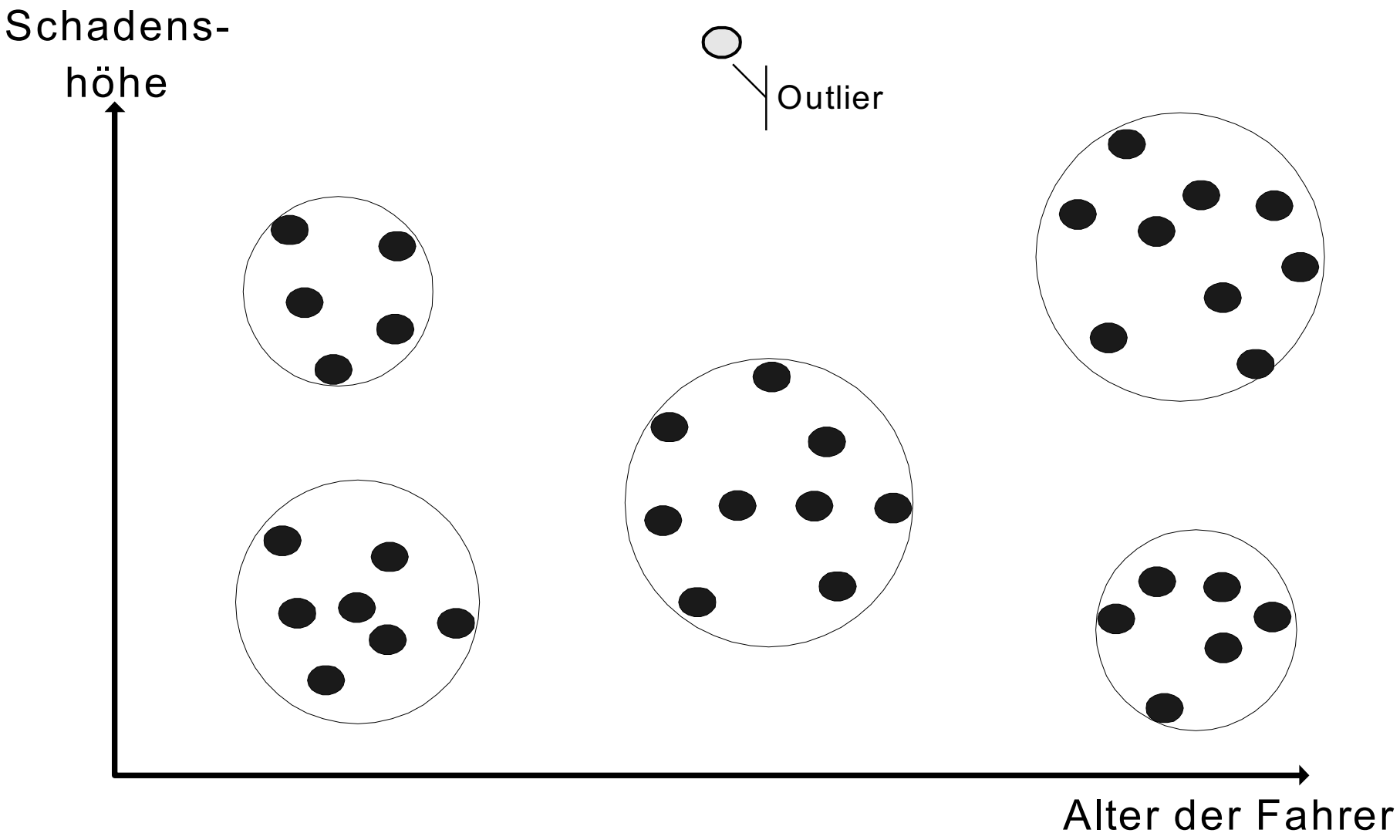
- Betrachte FI = {Drucker, Papier, Toner}
  - Support = 3
- Regel: {Drucker} ➔ {Papier, Toner}
  - Confidence = S({Drucker, Papier, Toner}) / S({Drucker})
    = (3/5) / (4/5)
    = ¾ = 75 %

Erhöhung der Confidence

- Vergrößern der linken Seite (dadurch Verkleinern der rechten Seite) führt zur Erhöhung der Confidence
  - Formal: $L \subset L^+$, $\mathbf{R} \subset \mathbf{R}^-$
  - Confidence(L➔R) <= C(L^+ ➔$\mathbf{R}^-$)

- Beispiel-Regel: {Drucker} ➔ {Papier, Toner}
  - Confidence = S({Drucker, Papier, Toner}) / S({Drucker})
    = (3/5) / (4/5)
    = ¾ = 75%
- Beispiel-Regel: {Drucker,Papier} ➔ {Toner}
  - Conf. = S({Drucker, Papier, Toner}) / S({Drucker,Papier})
    = (3/5) / (3/5)
    = 1 = 100%

Clustering



Clustering-Algorithmus

- Greedy Heuristik
- Lese sequentiell alle Datensätze
- Für den nächsten Datensatz r bestimme
  - Für alle bisher existierenden Cluster denjenigen c, dessen Zentrum den kürzesten Abstand zu r hat
  - Wenn distance(r,center(c)) <= epsilon
    - Füger r in c ein
  - Anderenfalls lege einen neuen Cluster c` an, der zunächst nur r enthält
- Funktioniert solange ganz gut, wie die Cluster in den Hauptspeicher passen

K-means Algorithmus

- Minimiere Q, also die Summe der Abstände der Datenpunkte x zum Mittelpunkt „ihres“ Clusters

$$Q = \sum_{i=1}^{k} \sum_{x_j \in S_i} ||x_j - \mu_i||^2$$

Algorithmus ... Im Detail

1. **Initialisierung:** Wähle zufällig k Mittelwerte $\mu_1^{(1)}, \ldots, \mu_k^{(1)}$ aus dem Datensatz aus. Diese bilden die initialen Mittelwerte der Cluster der ersten Iteration.

2. **Iteriere für** $t = 1$ **bis** T (oder solange sich noch Änderungen an den Clustern ergeben):

   (a) **Zuordnung zu Clustern:** Jedes Datenobjekt x_j wird jetzt dem Cluster $S_i^{(t)}$ zugeordnet, zu dessen Mittelwert $\mu_i^{(t)}$ es am nächsten liegt. Das heisst, nach diesen Zuordnungen enthält jedes Cluster $S_i^{(t)}$ für $1 \leq i \leq k$ nur Datenelemente, für die gilt:

   $$S_i^{(t)} = \{x_j : ||x_j - \mu_i^{(t)}||^2 \leq ||x_j - \mu_{i'}^{(t)}||^2 \text{ für alle } i' = 1, \ldots, k\}$$

   Falls ein Datenelement x_j sich für mehrere Cluster qualifiziert, wird es dennoch nur einem (bspw. zufällig gewählten) Cluster $S_i^{(t)}$ zugewiesen.

   (b) **Neuberechnung der Mittelwerte:** Jetzt werden für alle im t-ten Schritt berechneten Cluster die neuen Mittelwerte $\mu_i^{(t+1)}$ berechnet:

   $$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Beispiel-System: Microsoft® SQL Server 2000™ Analysis Services

```
CREATE MINING MODEL [MemberCards] (
    [customer Id] LONG KEY ,
    [Yearly Income] TEXT DISCRETE ,
    [Member Card Type] TEXT DISCRETE PREDICT,
    [Marital Status] TEXT DISCRETE )
USING Microsoft_Decision_Trees
```

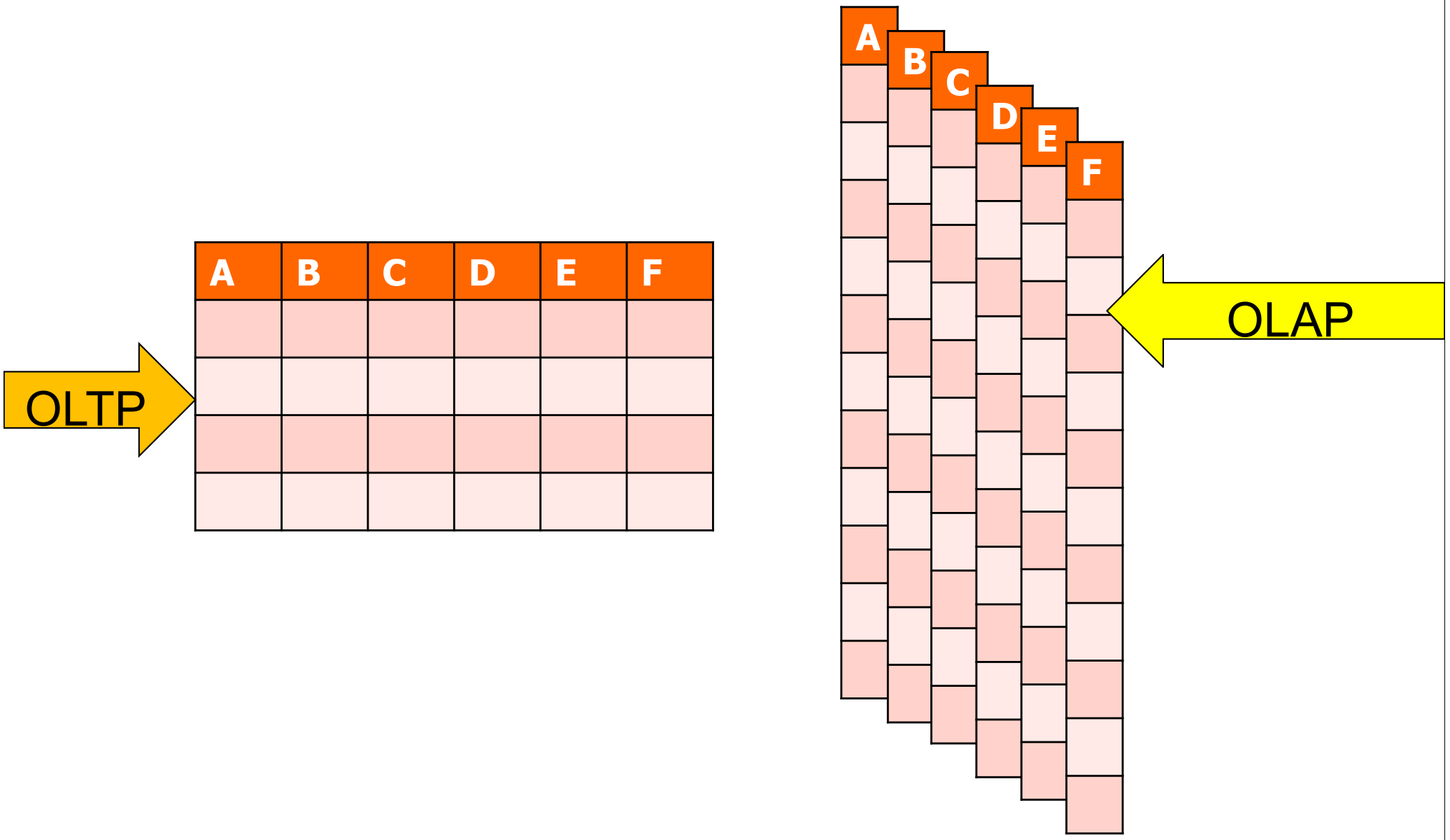
Beispiel-System: Microsoft® SQL Server 2000™ Analysis Services

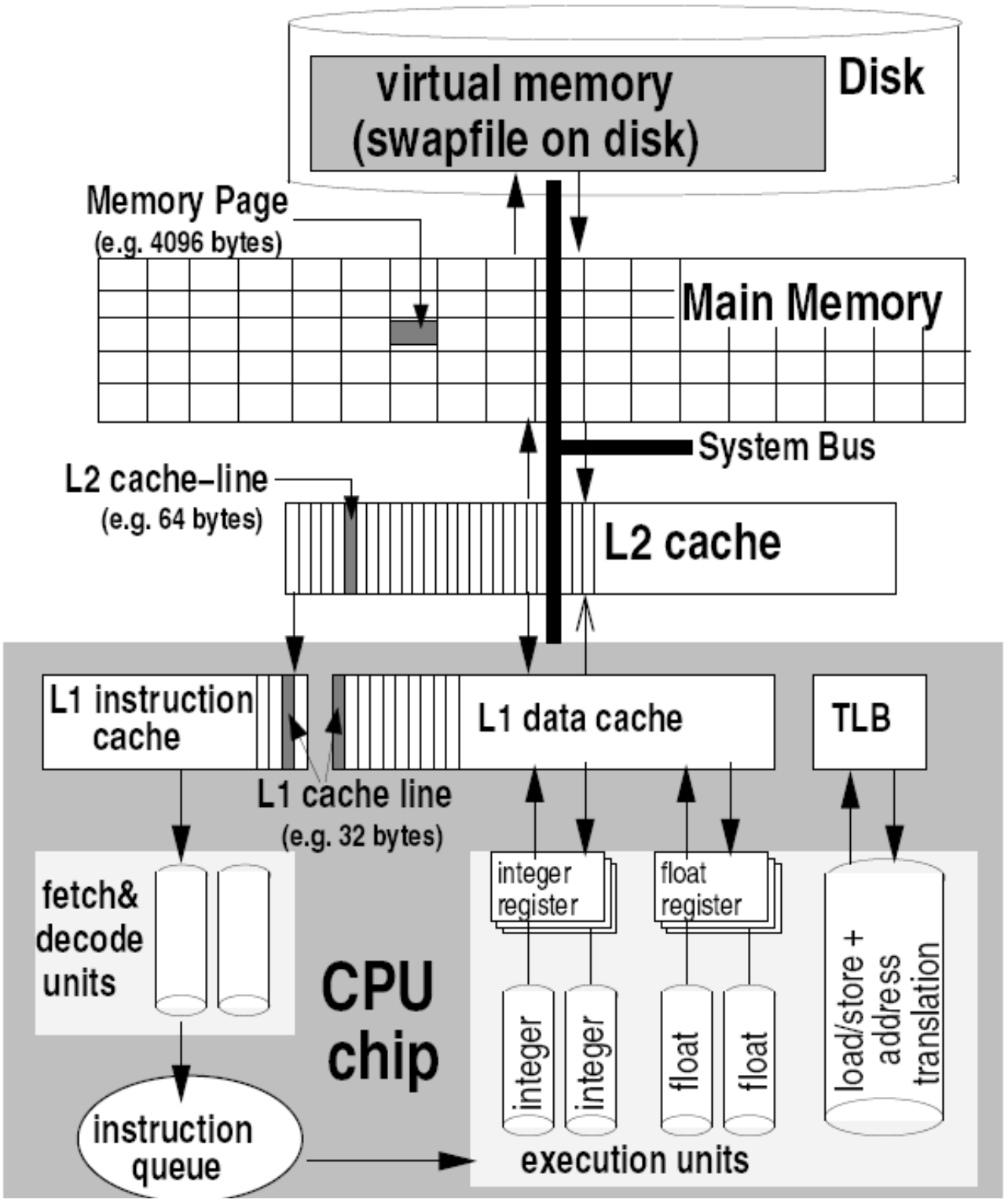
```
CREATE MINING MODEL [MemberCards] (
    [customer Id] LONG KEY ,
    [Yearly Income] TEXT DISCRETE ,
    [Member Card Type] TEXT DISCRETE PREDICT,
    [Marital Status] TEXT DISCRETE )
USING Microsoft_Decision_Trees
```

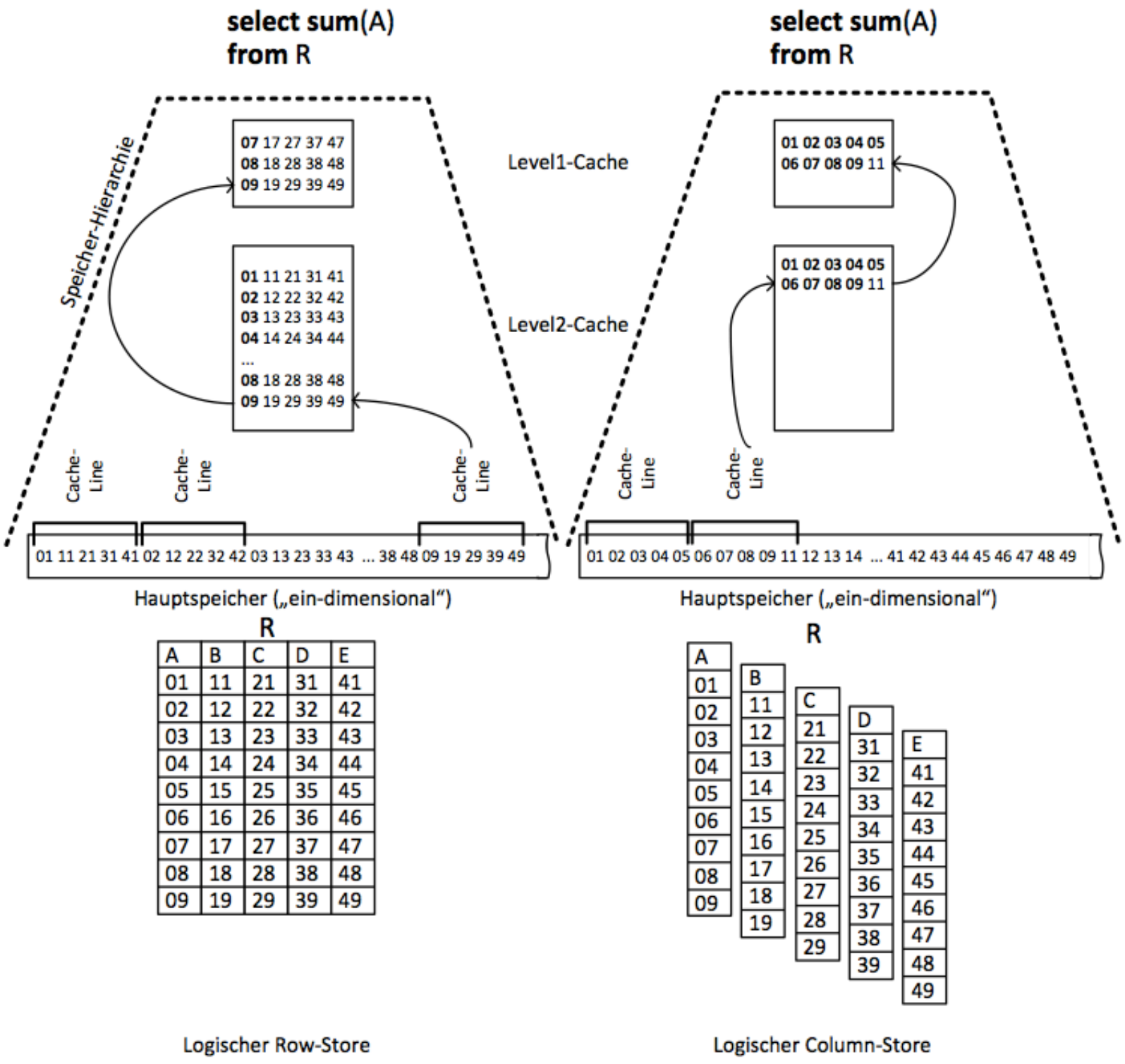
Mining Model definieren ➔ Trainieren ➔ in Anfragen nutzen:

```
SELECT [MemberCards].[Member Card Type]
FROM [Member Cards] NATURAL PREDICTION JOIN
     (SELECT 35000 AS [Yearly Income],
                  'single' AS [Marital Status]) as MoeglicheKunden
```

DW-Architektur: Row Store vs Column Store




virtual memory
(swapfile on disk)
Disk
Memory Page
(e.g. 4096 bytes)
Main Memory
System Bus
L2 cache–line
(e.g. 64 bytes)
L2 cache
L1 instruction
cache
L1 data cache
TLB
L1 cache line
(e.g. 32 bytes)
fetch&
decode
units
CPU
chip
integer
register
float
register
load/store +
address
translation
integer
integer
float
float
instruction
queue
execution units


select sum(A)
from R
Speicher-Hierarchie
07 17 27 37 47
08 18 28 38 48
09 19 29 39 49
Level1-Cache
01 11 21 31 41
02 12 22 32 42
03 13 23 33 43
04 14 24 34 44
...
08 18 28 38 48
09 19 29 39 49
Level2-Cache
Cache-Line
Cache-Line
Cache-Line
01 11 21 31 41 02 12 22 32 42 03 13 23 33 43 ... 38 48 09 19 29 39 49
Hauptspeicher („ein-dimensional“)
R
A B C D E
01 11 21 31 41
02 12 22 32 42
03 13 23 33 43
04 14 24 34 44
05 15 25 35 45
06 16 26 36 46
07 17 27 37 47
08 18 28 38 48
09 19 29 39 49
Logischer Row-Store
select sum(A)
from R
01 02 03 04 05
06 07 08 09 11
01 02 03 04 05
06 07 08 09 11
Cache-Line
Cache-Line
01 02 03 04 05 06 07 08 09 11 12 13 14 ... 41 42 43 44 45 46 47 48 49
Hauptspeicher („ein-dimensional“)
R
A
01
02
03
04
05
06
07
08
09
B
11
12
13
14
15
16
17
18
19
C
21
22
23
24
25
26
27
28
29
D
31
32
33
34
35
36
37
38
39
E
41
42
43
44
45
46
47
48
49
Logischer Column-Store

Row Store versus Column Store

Verkäufe				
Produkt	Kunde	Preis	Filiale	...
Handy	Kemper	345	Schwabing	...
Radio	Mickey	123	Bogenhausen	...
Handy	Minnie	233	Schwabing	...
Kühlschrank	Urmel	240	Augsburg	...
Beamer	Bond	740	London	...
Handy	Lucie	321	Bogenhausen	...

Row Store versus Column Store

Produkt	
ID	Produkt
0	Handy
1	Radio
2	Handy
3	Kühlschrank
4	Beamer
5	Handy

Kunde	
ID	Kunde
0	Kemper
1	Mickey
2	Minnie
3	Urmel
4	Bond
5	Lucie

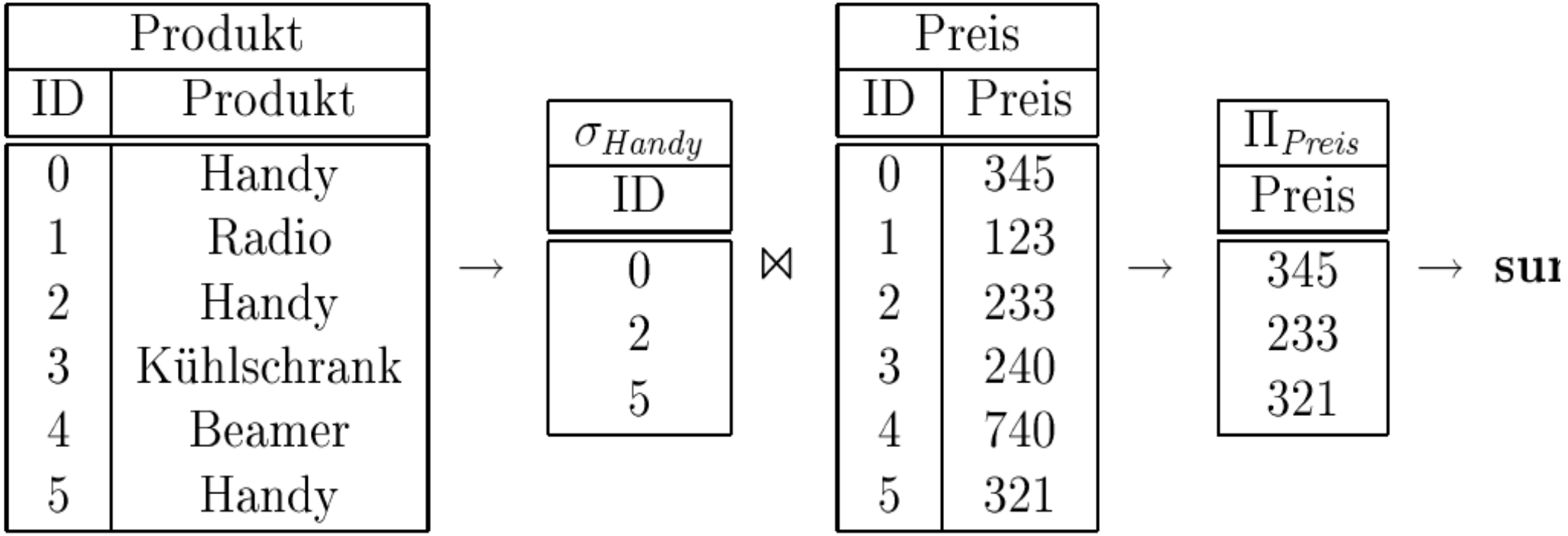
Preis	
ID	Preis
0	345
1	123
2	233
3	240
4	740
5	321

Filiale	
ID	Filiale
0	Schwabing
1	Bogenhausen
2	Schwabing
3	Augsburg
4	London
5	Bogenhausen

Anfragebearbeitung

select sum(Preis)
from Verkäufe
where Produkt = 'Handy'

Die schrittweise Abarbeitung dieser Anfrage ist nachfolgend illustriert



Komprimierung

Dictionary	
ID	Wort
0	Augsburg
1	Beamer
2	Bogenhausen
3	Handy
4	Kühlschrank
5	London
6	Radio
7	Schwabing
...	...

Produkt	
ID	Produkt
0	3
1	6
2	3
3	4
4	1
5	3

Filiale	
ID	Filiale
0	7
1	2
2	7
3	0
4	5
5	2