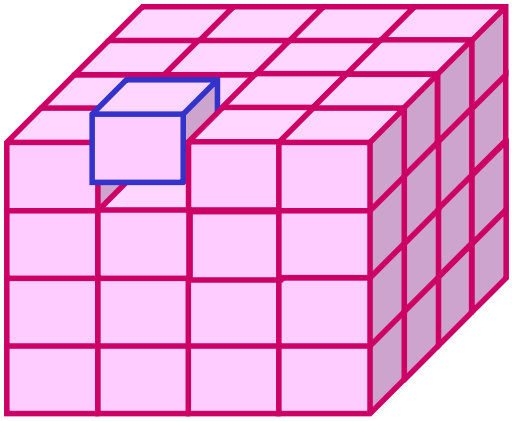
Data Warehousing

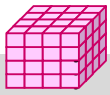
Kapitel 3: Mehrdimensionale Daten-modellierung und Operationen

Dr. Andreas Thor
Wintersemester 2009/10

Universität Leipzig
Institut für Informatik

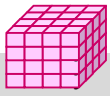
http://dbs.uni-leipzig.de





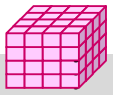
3. Mehrdimensionale Datenmodellierung und Operationen

- Grundlagen
  - Kennzahlen, Dimensionen, Cube
  - Cuboide / Aggregationsgitter
  - hierarchische Dimensionen / Konzepthierarchien
- Cube-Operationen
- Multi-dimensionale Speicherung (MOLAP)
- MDX
- Relationale Repräsentation mehrdimensionaler Daten (ROLAP)
  - Star-Schema
  - Varianten: Snowflake-, Galaxien-Schema
  - Anfragen: Star Join, Roll-Up, Drill-Down
  - Cube-Operator, Rollup-Operator, Grouping Sets

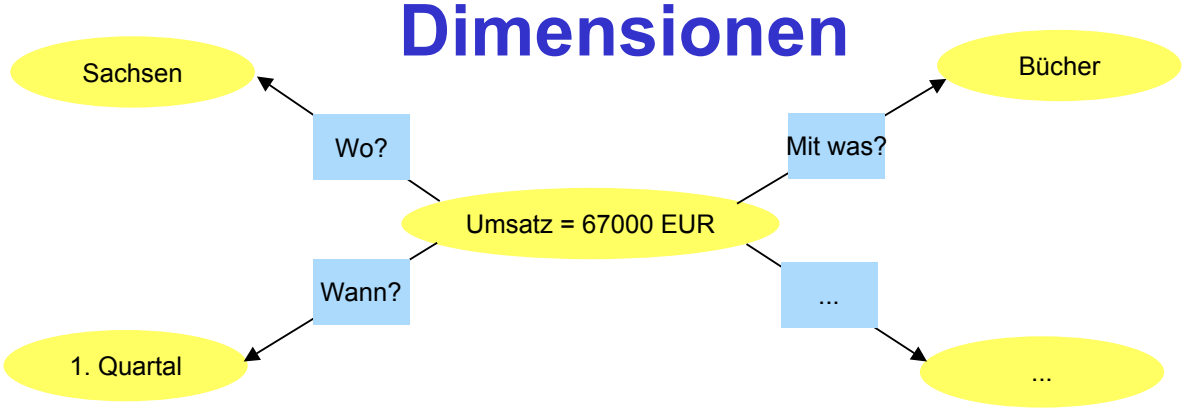


Kennzahlen

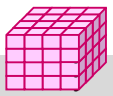
- auch: Fakten, Messgrößen, Measures, Measured Facts
- Kennzahl ist Größe mit konzentrierter Aussagekraft zur Diagnose, Überwachung und Steuerung eines Systems
  - meist betriebswirtschaftliche Größen, z.B. Umsatz / Gewinn / Rentabilität
  - komplexe Beziehungen zwischen Kennzahlen möglich
- Kennzahlen besitzen beschreibende Attribute
  - z.B. Einheit, Wertebereich, Berechnungsvorschrift
- Arten von Kennzahlen
  - *additive* Kennzahlen: additive Aggregation hinsichtlich aller Dimensionen möglich
  - *semi-additive* Kennzahlen: additive Aggregation nur hinsichtlich ausgewählter Dimensionen möglich
  - *nicht-additive* Kennzahlen (Bsp. Durchschnittswerte, Prozentanteile)



Dimensionen

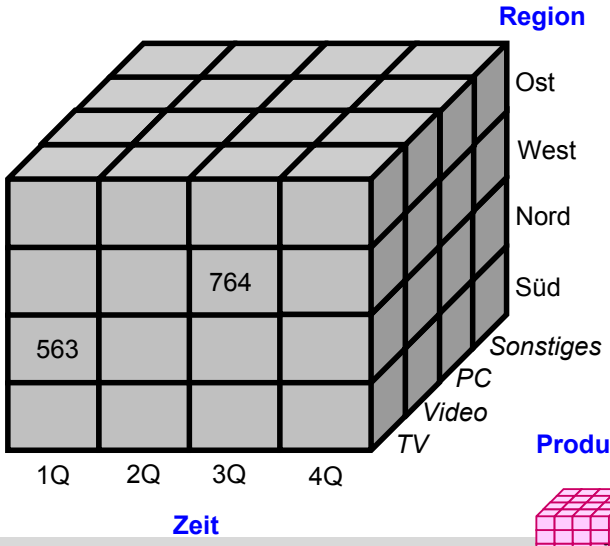


- Zahlenwert einer Kennzahl ohne semantischen Bezug nichtssagend
- Dimensionen setzen Kennzahlen in Bezug zu Eigenschaften / sachlichen Kriterien
- *Dimension*: Datentyp, i.a. endlich (z.B. Aufzählung)
  - Beispiele: Menge aller Produkte, Regionen, Kunden, Zeitperioden etc.
  - *Dimensionselement*: Element / Ausprägung / Wert zu einer Dimension
  - Attribute: *Klassifikations-/Kategorienattribute* (inkl. eines *Primärattributs*) sowie *„dimensionale Attribute“* (zusätzliche beschreibende Eigenschaften, z.B. Produktfarbe / Gewicht



Data Cube

- Datenwürfel bzw. OLAP-Würfel (Cube), Data Cube
  - Dimensionen: Koordinaten
  - Kennzahlen: Zellen im Schnittpunkt der Koordinaten
- Cube bezüglich Dimensionen D1, ...Dn und k Kennzahlen (Fakten):
  - W = { (d_1, ... d_n), (f_1, ... f_k), Dimensionselement d_i aus D_i, i= 1.. n, Kennzahlen f_j, j = 1..k)}
  - eindeutige Zellen-Adresse: (d_1, ... d_n)
  - Zellen-Inhalt: (f_1, ... f_k)
- n: Dimensionalität des Cube
- Alternative: k Cubes mit je einer Kennzahl pro Zelle (Multi-Cube)
- typischerweise 4 - 12 Dimensionen
  - Zeitdimension fast immer dabei
  - weitere Standarddimensionen: Produkt, Kunde, Verkäufer, Region, Lieferant, ...



Tabellen-Darstellung von Cubes

- direkte Umsetzung für 2 Dimensionen (2D-Sicht auf Produkt x Region)

Zeit = „Quartal1“

	Ost	Süd	West
Produkt 1	30	100	100
Produkt 2	40	110	88
Produkt 3	17	70	50

Zeit = „Quartal2“

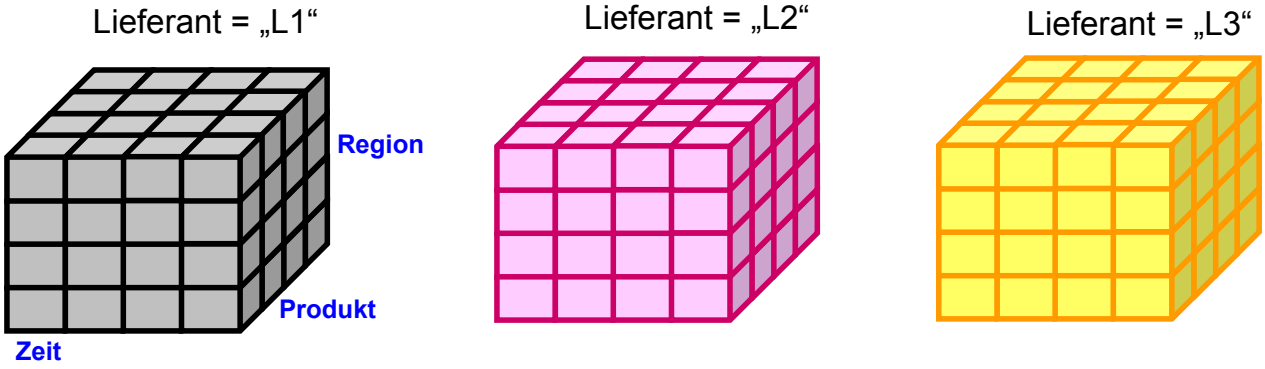
	Ost	Süd	West
Produkt 1	34	87	60
Produkt 2	32	80	103
Produkt 3	14	73	60

...

- 3 Dimensionen: mehrere 2D-Tabellen bzw. geschachtelte Tabellen bzw. 3D-Cube

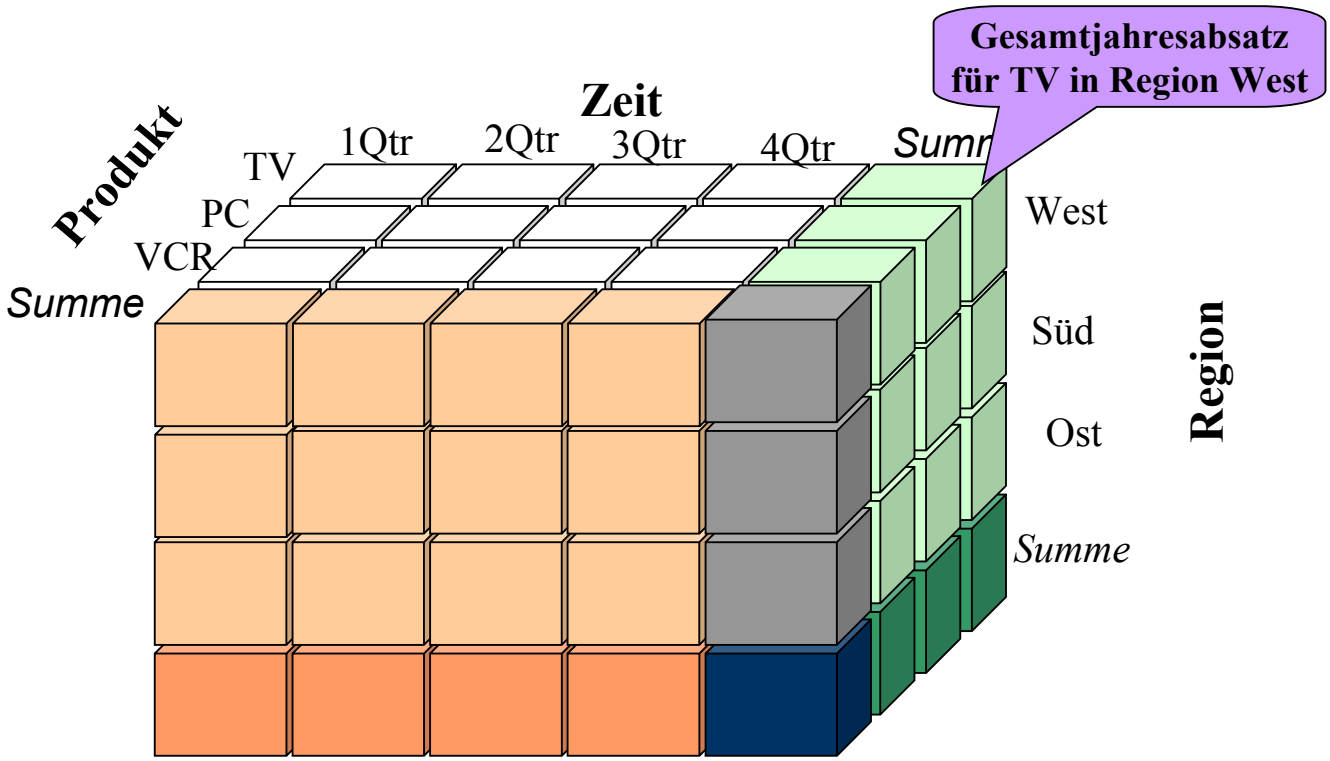
		Ost	Süd	West
Produkt 1	**Quartal 1**	30	100	100
	Quartal 2	34	87	60
Produkt 2	**Quartal 1**	40	110	88
	Quartal 2	32	80	103
Produkt 3	**Quartal 1**	17	70	50
	Quartal 2	14	73	60

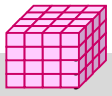
Cube-Darstellung



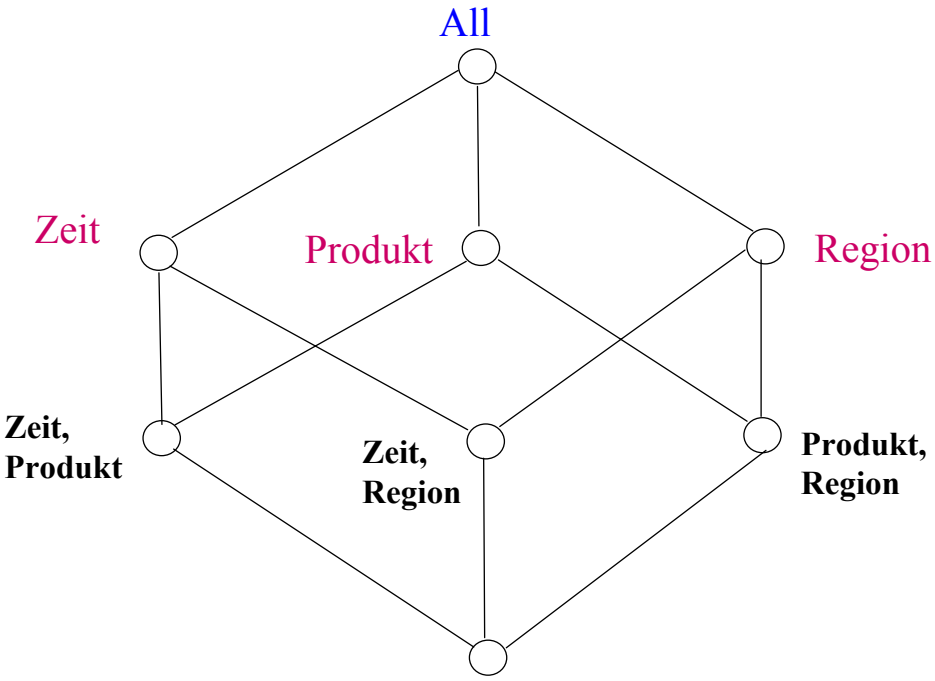
- 4D-Cube kann als Menge von 3D-Cubes dargestellt werden
- Aggregation: Aus N-dimensionalem Cube kann Menge von (N-1)-dimensionalen Sub-Cubes oder *Cuboiden („Quadern“)* abgeleitet werden
  - Basis-Cuboid: n-dimensionaler Cube
  - Scheitel-Cuboid: 0-dimensionale Aggregation über alle Dimensionen
  - aus Basis-Cuboid lassen sich Cuboiden geringerer Dimensionsanzahl ableiten -> Data Cube entspricht Verband (Lattice) von Cuboiden
  - N-dimensionaler Cube hat 2^N Cuboiden inkl. Basis-Cuboid (ohne Berücksichtigung von Dimensionshierarchien)

Data Cube: 3D-Beispiel mit Aggregation





Zugehörige Cuboiden (Aggregationsgitter)



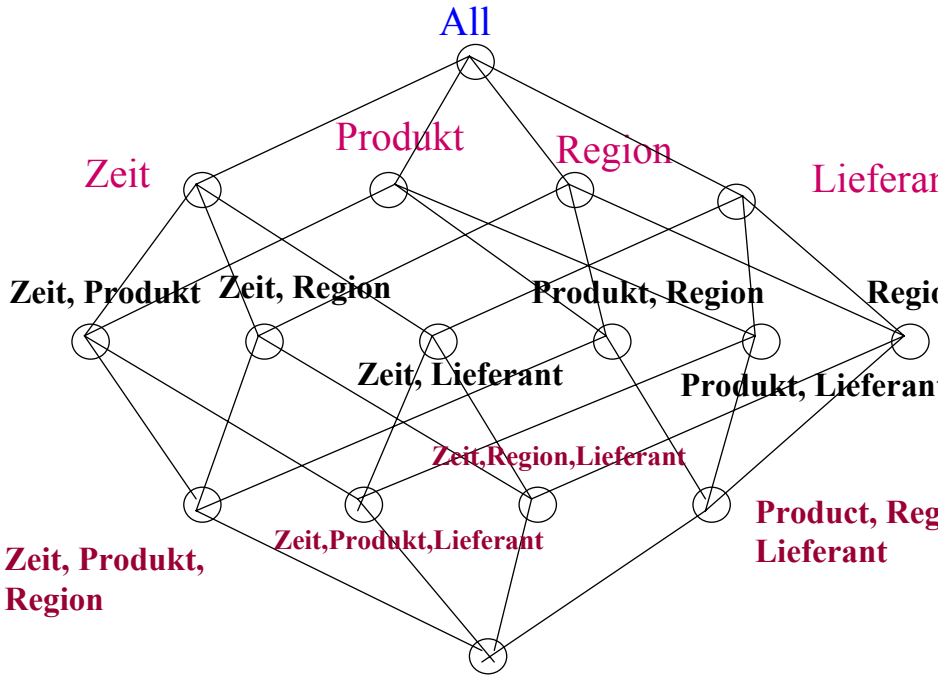
0-D (Scheitel) Cuboid

1-D Cuboiden

2-D Cuboiden

3-D (Basis)-Cuboid

Cube: Verband von Cuboiden



0-D(Scheitel) Cuboid

1-D Cuboiden

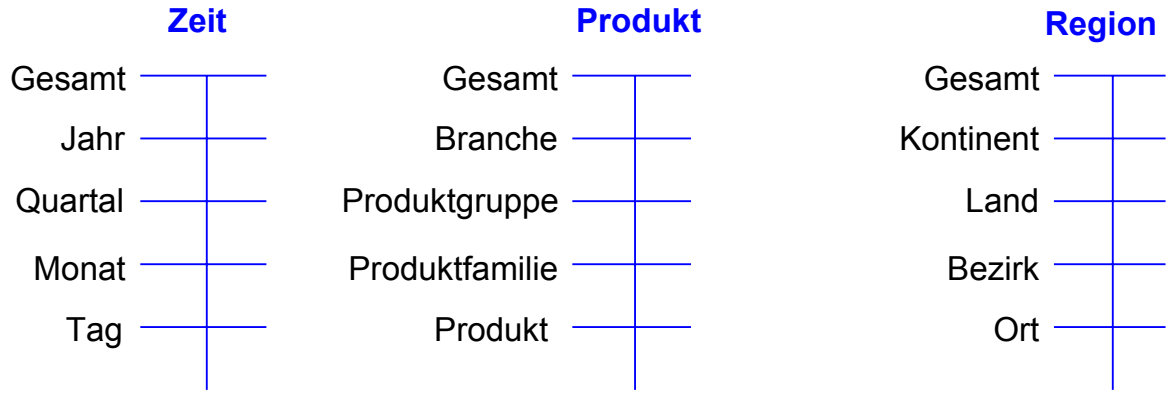
2-D Cuboiden

3-D Cuboiden

4-D (Basis)-Cuboid

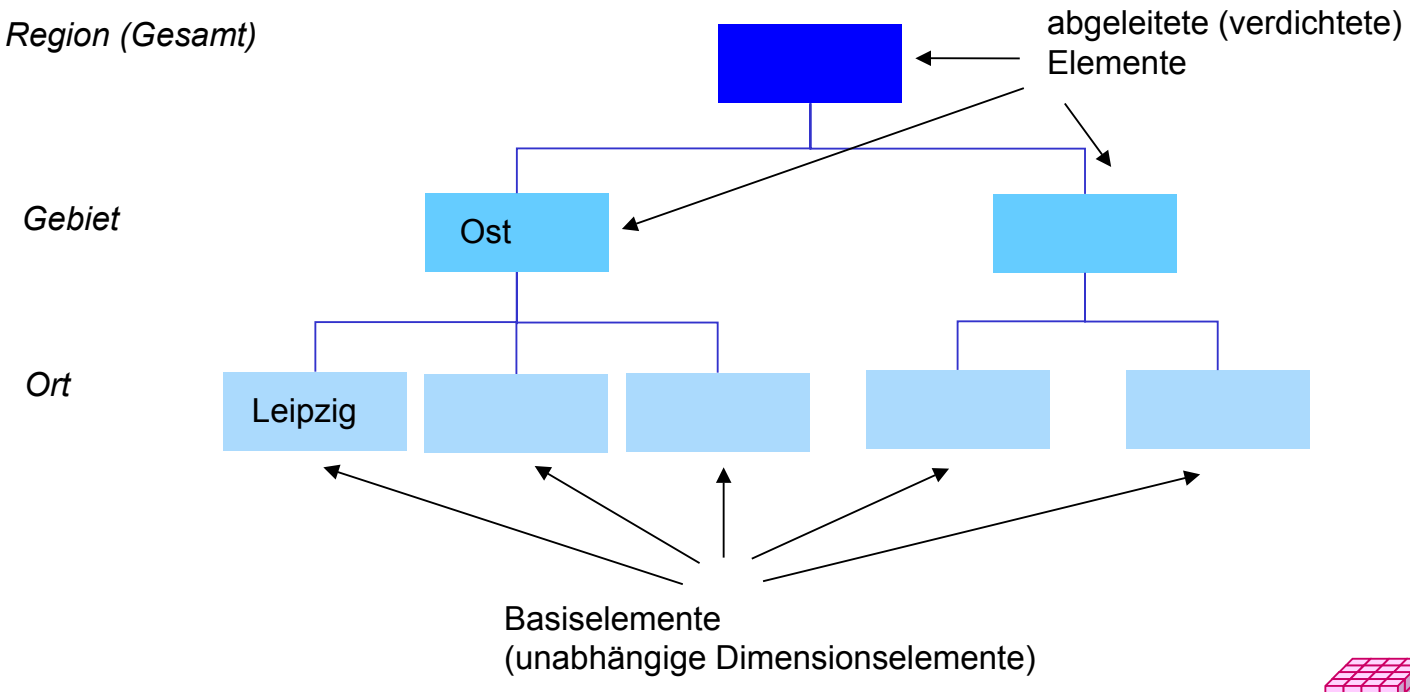
Dimensionshierarchien (Konzepthierarchien)

- häufig hierarchische Beziehungen zwischen Dimensionsobjekten
  - Top-Level pro Hierarchie für alle Dimensionselemente (Gesamt, Top, All)
  - *Primärattribut*: unterste (genaueste) Stufe
  - funktionale Abhängigkeiten zwischen Primärattribut und *Klassifikationsattributen* höherer Stufen
- Beispiele



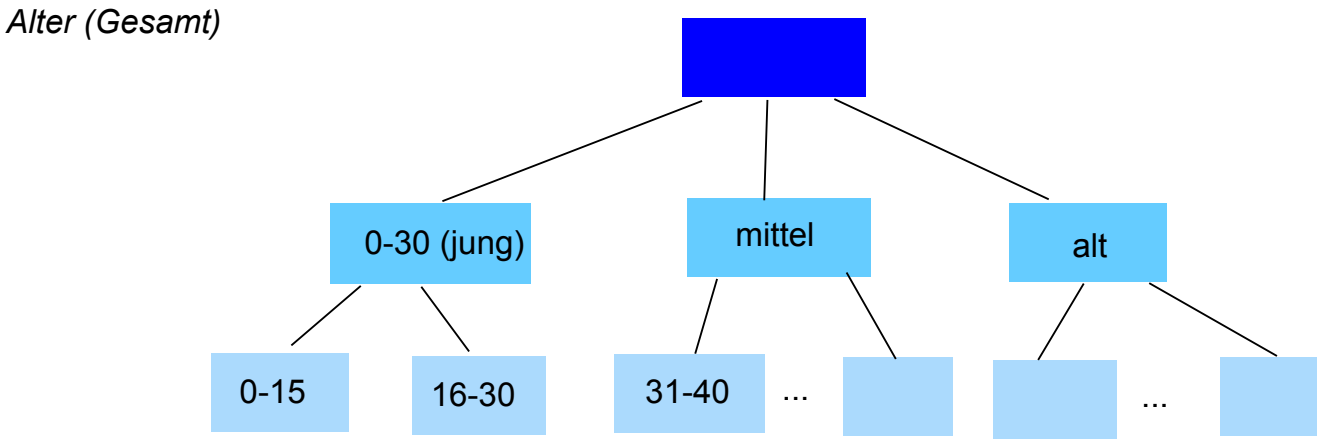
Beispiel einer Konzepthierarchie (Region)

- *einfache Hierarchie* (pro Element höchstens ein übergeordnetes Element) vs. *parallele Hierarchie* bzw. Halbordnung

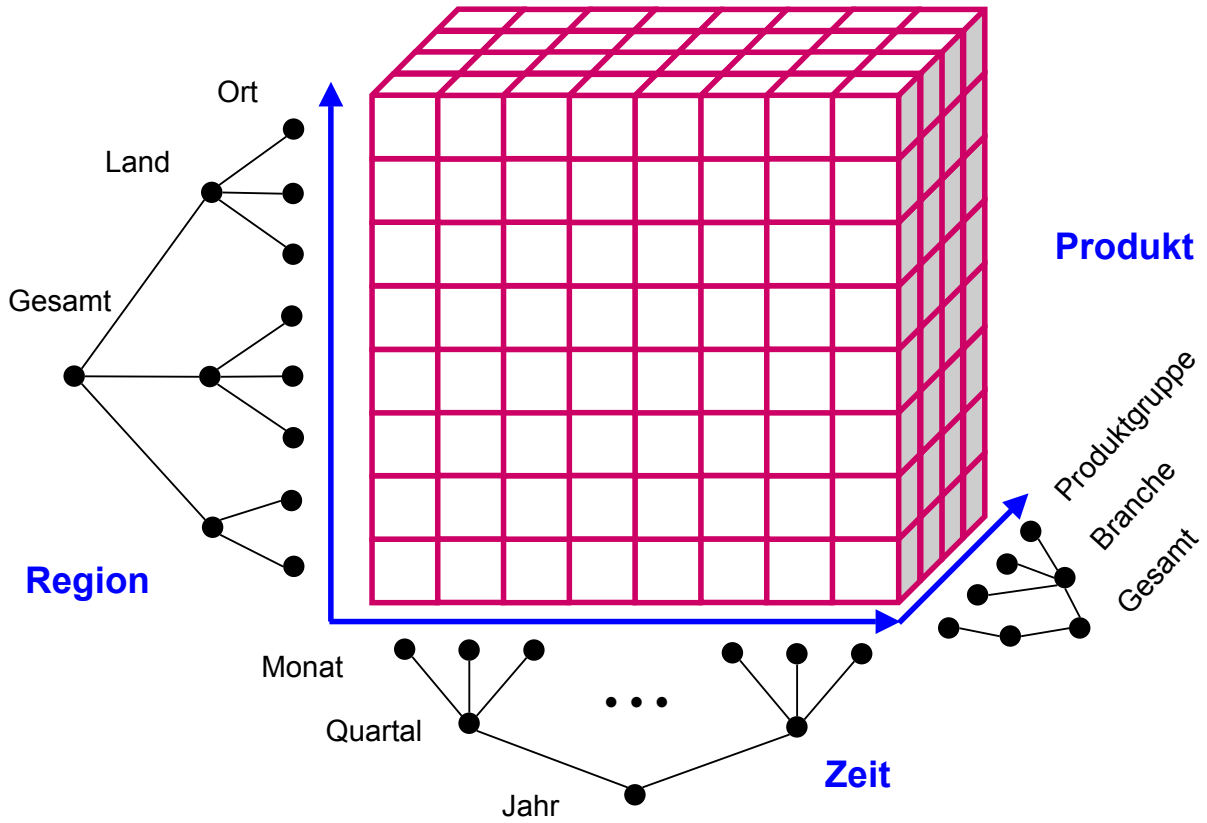


Konzepthierarchien (3)

- Hierarchien: auf Schemaebene meist durch Klassifikationsattribute und deren funktionalen Abhängigkeiten gegeben
- Variante: Hierarchiebildung durch Wertegruppierungen / Diskretisierungen („Set-grouping Hierarchies“)
  - können Auswertungen vereinfachen
  - günstige Einteilungen auf Basis vorhandener Werte teilweise automatisch berechenbar

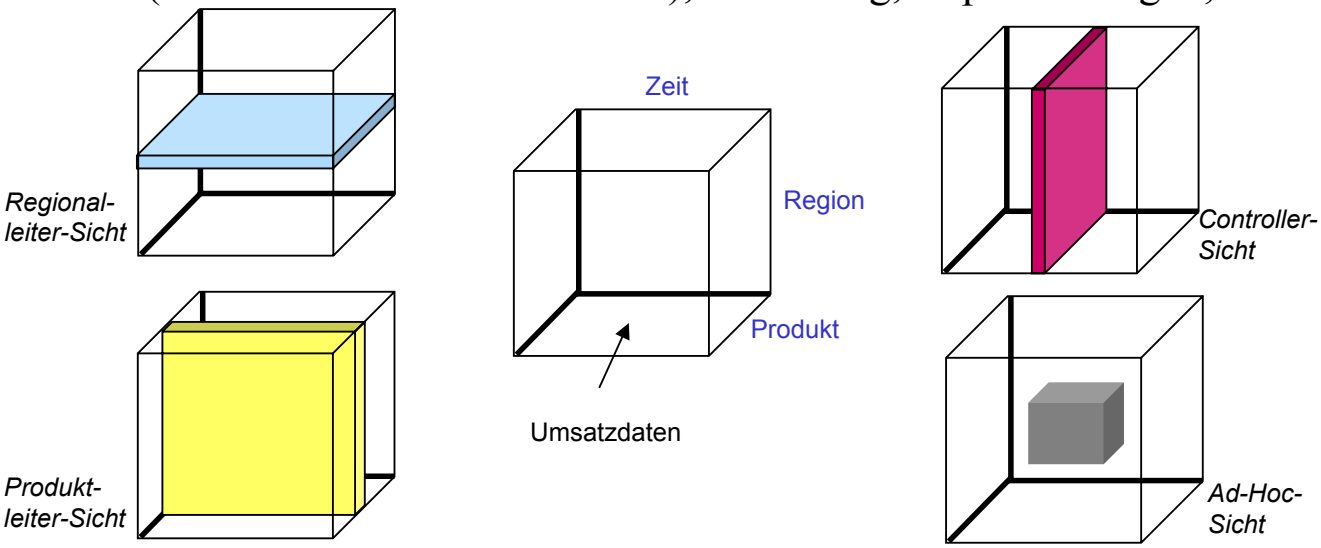


Cube mit hierarchischen Dimensionen

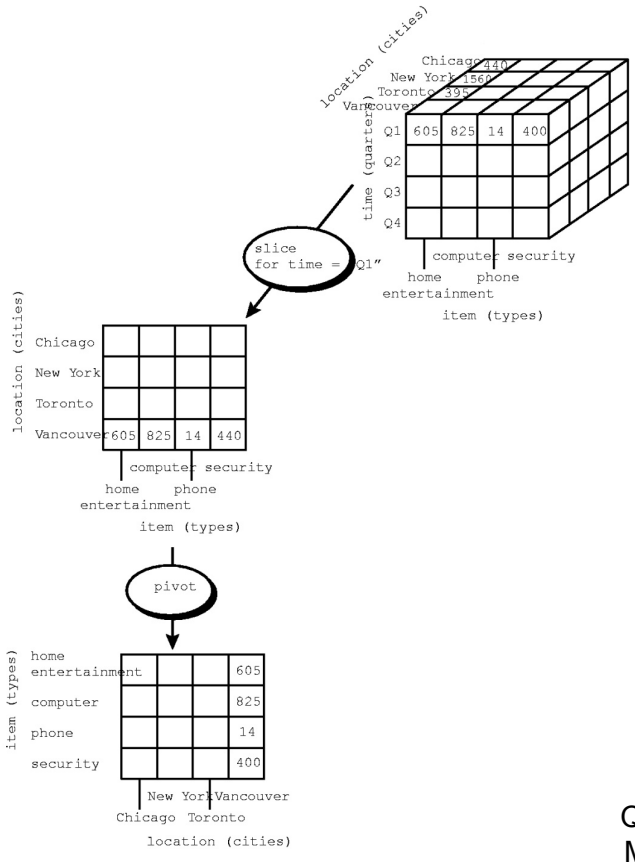


Operationen auf Cubes

- Slice: Herausschneiden von „Scheiben“ aus dem Würfel durch Einschränkung (Selektion) auf einer Dimension
  - Verringerung der Dimensionalität
- Dice: Herausschneiden einen „Teilwürfels“ durch Selektion auf mehreren Dimensionen
- unterschiedlichste mehrdimensionale Aggregationen / Gruppierungen
- Pivot (Austausch von Dimensionen), Sortierung, Top-n-Anfragen, ...

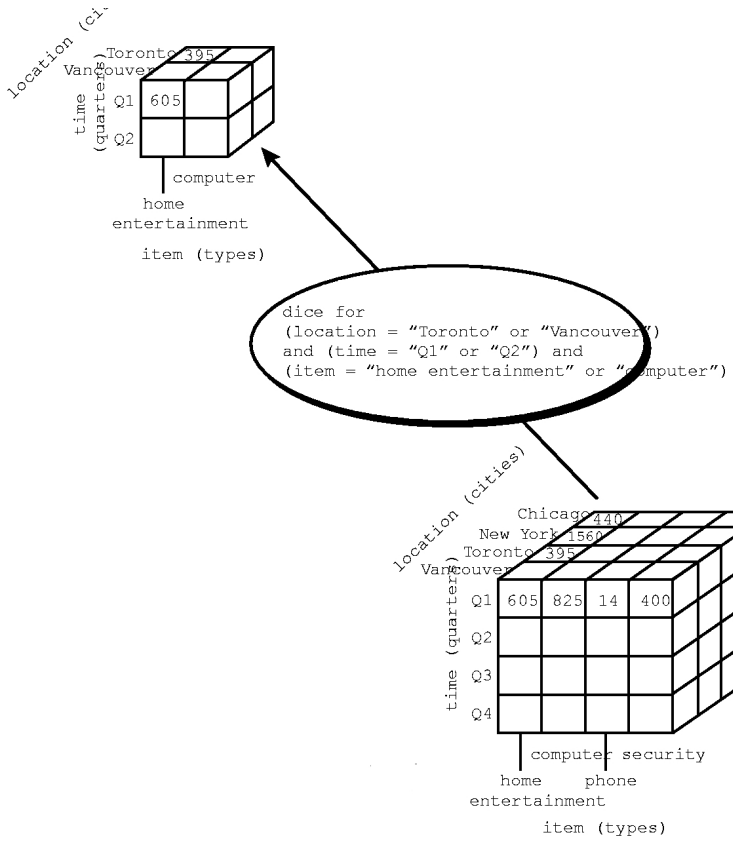


Beispiele: Slice / Pivot



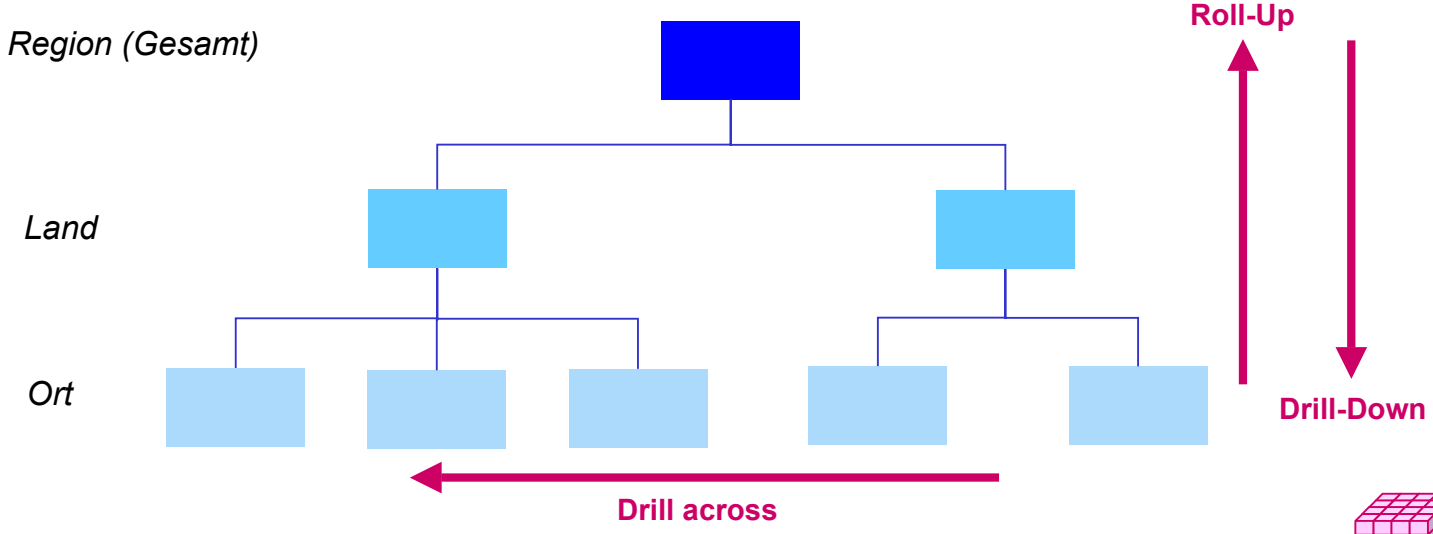
Quelle: J. Han, M. Kamber: Data Mining,Morgan Kaufmann, 2001

Beispiel: Dice

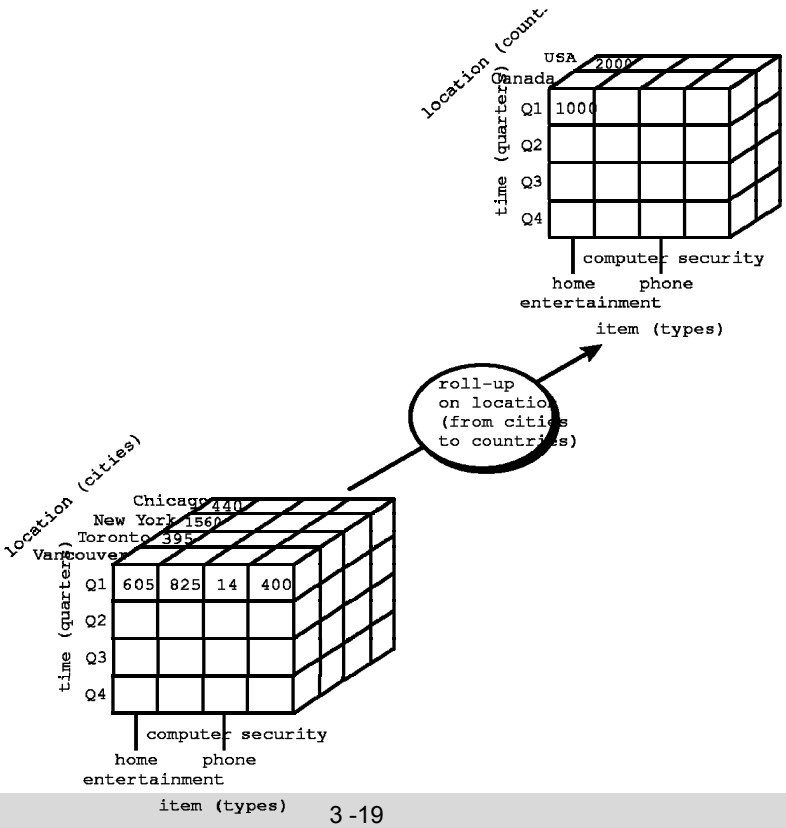


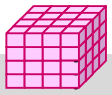
Navigation in Hierarchien

- Drill-Down
  - Navigation nach „unten“ in der Hierarchie
  - Erhöhung des Detailgrad: von verdichteten Daten zu weniger verdichteten/aggregierten Daten
- Roll-Up (Drill-Up)
  - Navigation nach „oben“ in der Hierarchie
  - von weniger verdichteten (aggregierten) Daten zu stärker verdichteten Daten

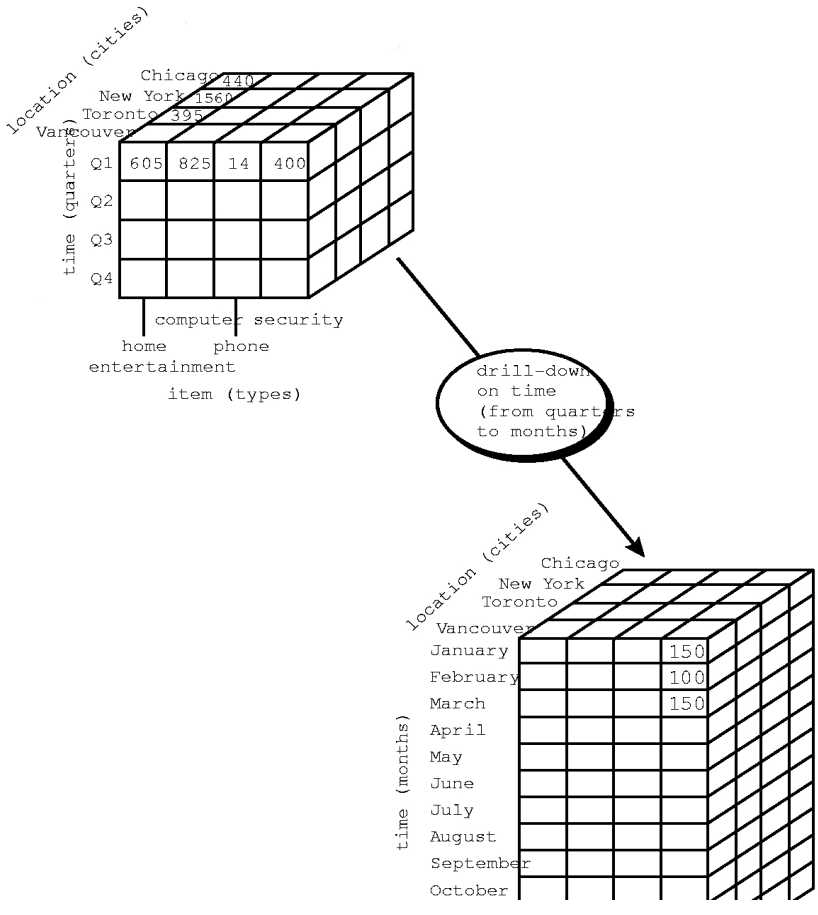


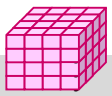
Roll-Up





Drill-Down



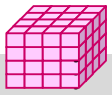


Drill-Down / Roll-Up (2D)

Produktgruppe	*Ost*	*Süd*	*Nord*	*West*
Elektronik	1800	1500	1450	2000
Spielwaren	500	1700	600	1500
Kleidung	1200	1200	400	1000

Drill-Down ↓ ↑ **Roll-Up**

Elektronik	*Ost*	*Süd*	*Nord*	*West*
TV	800			
DVD-Player	700			
Camcorder	300			

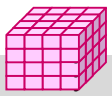


Aggregation: 2D-Beispiel

- Summenbildung

Produktgruppe	*Ost*	*Süd*	*Nord*	*West*	**Summe**
Elektronik	1800	1500	1450	2000	6750
Spielwaren	500	1700	600	1500	4300
Kleidung	1200	1200	400	1000	3800
Summe	3500	4400	2450	4500	14850

- Vorberechnung (Materialisierung) der Aggregationen zur schnellen Beantwortung von Aggregationsanfragen
- hoher Speicher- und Aktualisierungsaufwand (bei vielen Dimensionselementen) ermöglicht nur kleinen Teil benötigter Aggegationen vorzuberechnen

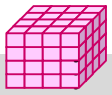


Größe der Cubes

- Größe der Basis-Cuboids
  - Anzahl der Zellen entspricht Produkt der Dimensionskardinalitäten D_i , i=1..n
  - Beispiel: 1.000 Tage, 100.000 Produkte, 1 Million Kunden
  - jede weitere Dimension, z.B. Region oder Verkäufer, führt zu einer Vervielfachung des Datenraumes
- Vorberechnung von (aggregierten) Cuboiden erhöht Speicherbedarf
- Größe eines hierarchisch aggregierten Cubes
  - Aggregierung für jedes Dimensionselement auf einer höheren Hierarchiestufe möglich
  - Kombinationsmöglichkeit mit jedem Element auf einer der Hierarchiestufen der anderen n-1 Dimensionen
- Anzahl Cuboiden bei n-dimensionalem Cube: $T = \prod_{i=1}^{n} (L_i + 1)$

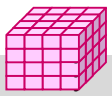
  L_i: #Ebenen von Dimension i (ohne Top-Level)

Zeit		Produkt		Kunde	
Gesamt	1	Gesamt	1	Gesamt	1
Quartal	12	Branche	50	Kundengruppe	10.000
Monat	36	Produktgruppe	5.000	Einzelkunde	1 Million
Tag	1000	Produkt	100.000		



Umsetzung des multi-dimensionalen Modells

- Aspekte
  - Speicherung der Daten
  - Formulierung / Ausführung der Operationen
- MOLAP: Direkte Speicherung in multi-dimensionalen Speicherungsstrukturen
  - Cube-Operationen einfach formulierbar und effizient ausführbar
  - begrenzte Skalierbarkeit auf große Datenmengen
- ROLAP: relationale Speicherung der Daten in Tabellen
  - effiziente Speicherung sehr großer Datenmengen
  - umständliche Anfrageformulierung
  - Standard-SQL nicht ausreichend (nur 1-dimensionale Gruppierung, ...)
- HOLAP: hybride Lösung
  - relationale Speicherung der Detail-Daten, multidimensionale Zugriffsschnittstelle
  - unterschiedliche Kombinationen mit multidimensionaler Speicherung / Auswertung von aggregierten Daten
- Vorberechnung von Aggregationen erforderlich für ausreichende Leistung



Multi-dimensionale Datenspeicherung

- Datenspeicherung in multi-dimensionaler Matrix
  - direkte Umsetzung der logischen Cube-Sicht
  - Vorab-Berechnung und Speicherung der Kennzahlen basierend auf dem Kreuzprodukt aller Wertebereiche der Dimensionen
  - schneller direkter Zugriff auf jede Kennzahl über Indexposition (x_1, x_2, ... x_n)

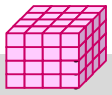
multi-dimensional (Kreuztabelle)

	Sachsen	Brandenburg	Thüringen
TV	100	150	200
DVD-Player	50	170	150
Camcorder	20	120	100

relational

Produkt	Region	Umsatz
TV	Brandenburg	150
Camcorder	Sachsen	20
...	...	...

Anfragen:
- Wie hoch ist der Umsatz für DVD-Player in Thüringen
- Wie hoch ist der Gesamtumsatz für Camcorder?



Multi-dimensionale Datenspeicherung (2)

- mehrdimensionale Speicherung führt oft zu dünn besetzten Matrizen
- Beispiel (Kundenumsätze nach Regionen)

multi-dimensional (2-dimenstional)

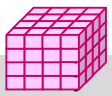
REGION

Kunde	*B*	*S*	*NRW*	*SH*	*BW*	*SA*	*MVP*	*HH*	*TH*
Kunde 1	100	-	-	-	-	-	-	-	-
Kunde 2	-	-	150	-	-	-	-	-	-
Kunde 3	-	-	-	-	200	-	-	-	-
Kunde 4	-	50	-	-	-	-	-	-	-
Kunde 5	-	-	-	170	-	-	-	-	-
Kunde 6	-	-	-	-	-	-	-	-	100
Kunde 7	-	-	-	-	-	20	-	-	-
Kunde 8	-	-	-	-	-	-	120	-	-
Kunde 9	-	-	-	-	-	-	-	100	-

relational

Kunden	Region	Umsatz
Kunde 1	B	100
Kunde 2	NRW	150
Kunde 3	BW	200
Kunde 4	S	50
Kunde 5	SH	170
Kunde 6	TH	100
Kunde 7	SA	20
Kunde 8	MVP	120
Kunde 9	HH	100

- vollständig besetzte Matrizen i.a. nur für höhere Dimensionsebenen
- Unterstützung dünn besetzter Matrizen erforderlich (Leistungseinbussen)
  - Zerlegung eines Cubes in Sub-Cubes („chunks“), die in Hauptspeicher passen
  - zweistufige Adressierung: Chunk-Id, Zelle innerhalb Chunk



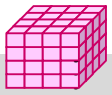
Sprachansatz MDX*

- MDX: MultiDimensional eXpressions
  - Microsoft-Spezifikation für Cube-Zugriffe / Queries im Rahmen von OLE DB for OLAP
  - an SQL angelehnt
  - Extraktion von aggregierten Sub-Cubes / Cuboiden aus Cubes
- Unterstützung durch Microsoft und zahlreiche Tool-Anbieter
- Hauptanweisung

```
SELECT    [<axis_specification> [, <axis_specification>...]]
FROM      [<cube_specification>]
[WHERE [<slicer_specification>]]
```

  - Axis_specification: Auszugebende Dimensionselemente
  - 5 vordefinierte Achsen: columns, rows, pages, chapters, and sections
  - Slicer: Auswahl der darzustellenden Werte

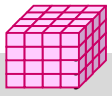
* http://msdn.microsoft.com/en-us/library/ms145506.aspx



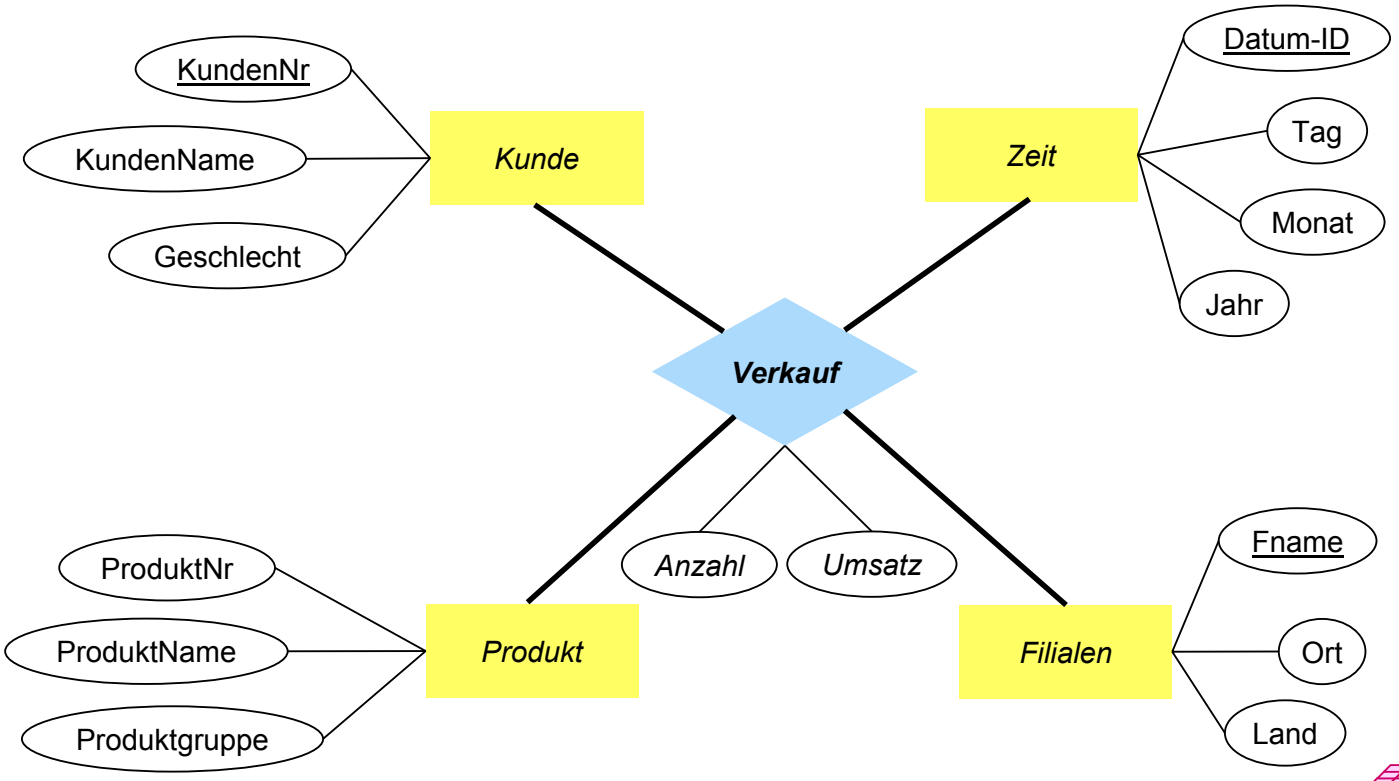
MDX: Beispiele

```
SELECT  Region.CHILDREN ON COLUMNS,
        Produkt.CHILDREN ON ROWS
FROM    Verkauf
WHERE   (Umsatz, Zeit.[2007])
```

```
SELECT  Measures.MEMBERS ON COLUMNS,
        TOPCOUNT(Filiale.Ort.MEMBERS, 10, Measures.Anzahl) ON ROWS
FROM    Verkauf
```

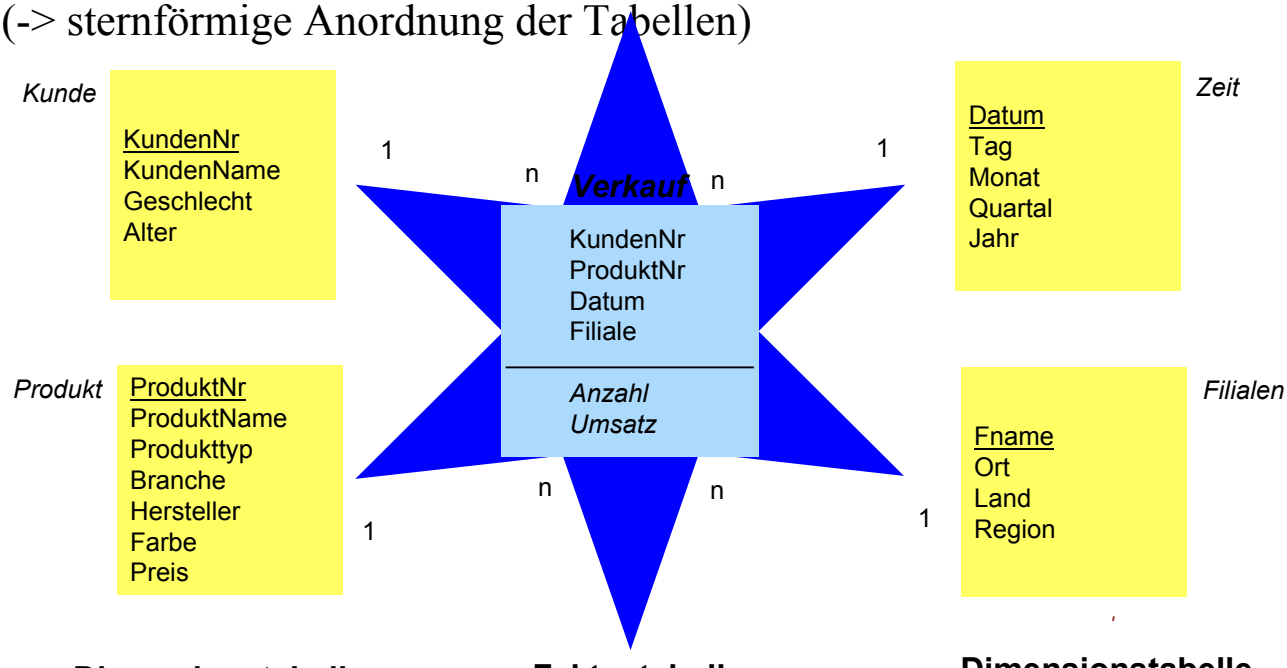


ER-Diagramm eines multi-dimensionalen Datenmodells



Relationale Speicherung: Star-Schema

- Faktentabelle bildet Zentrum des Star-Schemas und enthält die Detail-Daten mit den zu analysierenden Kennzahlen
- 1 Dimensionstabelle pro Dimension, die nur mit Faktentabelle verknüpft ist (-> sternförmige Anordnung der Tabellen)



Beispielausprägungen

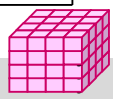
Verkauf					
Datum	Filiale	ProduktNr	KundenNr	Anzahl	Umsatz
7654	Leipzig4	1847	4711	1	24000
...	...	...	...	...	...

Filialen			
FName	Ort	Land	Region
Leipzig4	Leipzig	Sachsen	Ost
...	...	...	...

Kunde			
KundenNr	Name	Geschlecht	Alter
4711	Weber	M	39
...	...	...	...

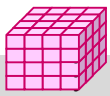
Zeit					
Datum	Tag	Monat	Jahr	Quartal	...
7654	25	April	2005	2	...
...	...	...	...	...	

Produkt					
ProduktNr	Produktname	Produkttyp	Hersteller	Farbe	Preis
1847	Passat XY	Auto	VW	Blau	27999
...	...	...	...	...	...



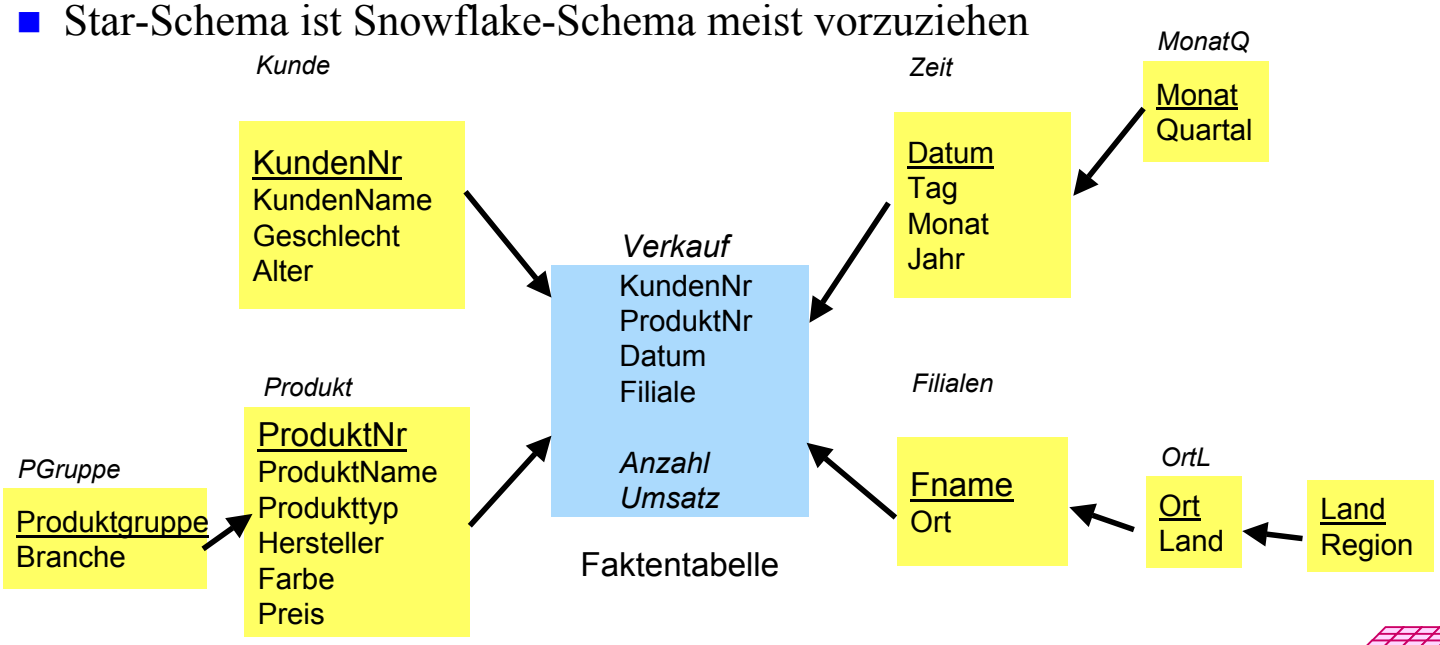
Star-Schema (2)

- Formale Definition: Star-Schema besteht aus einer Menge von Tabellen D_1, ...D_n, F mit
  - Dimensionstabellen D_i bestehend aus (i.a. künstlichen) Primärschlüssel d_i und Dimensionsattributen
  - Faktentabelle F bestehend aus Fremdschlüsseln d_1, ... d_n sowie Meßgrößen (Kennzahlen) als weiteren Attributen
  - Die Dimensionstabellen sind i.a. denormalisiert, d.h. nicht in dritter Normalform
- Beobachtungen
  - Anzahl der Datensätze in Faktentabelle entspricht Anzahl der belegten Zellen einer multi-dimensionalen Matrix
  - leere Dimensionskombinationen unproblematisch, da nur relevante Kombinationen in der Faktentabelle auftreten.
  - dennoch oft riesige Faktentabellen
  - Dimensionstabellen vergleichsweise klein, teilweise jedoch auch umfangreich (Kunden, Artikel etc.)



Snowflake-Schema

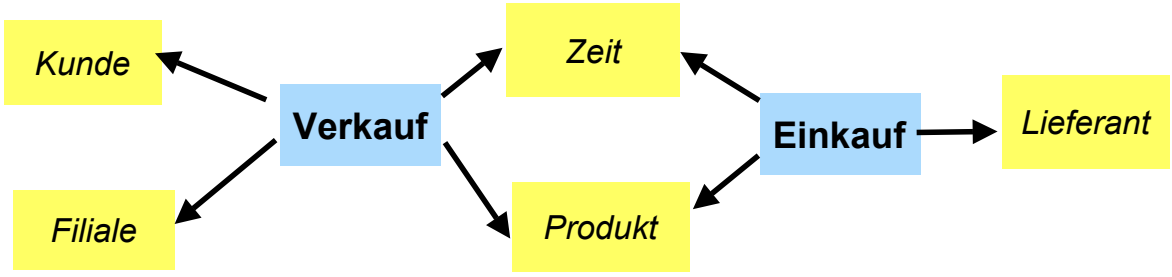
- explizite Repräsentation der Dimensionshierarchien
- normalisierte Dimensionstabellen
  - leicht geringere Redundanz, geringerer Änderungsaufwand
  - erhöhte Zugriffskosten (höherer Join-Aufwand)
- Star-Schema ist Snowflake-Schema meist vorzuziehen



Galaxien-Schema

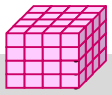
- Data Warehouses benötigen meist mehrere Faktentabellen

  -> Multi-Star-Schema (Galaxien-Schema, „Fact Constellation Schema“)
- gemeinsame Nutzung von Dimensionstabellen
- Speicherung vorberechneter Aggregate
  - separate Faktentabelle
  - im Rahmen der Faktentabelle mit Detail-Daten



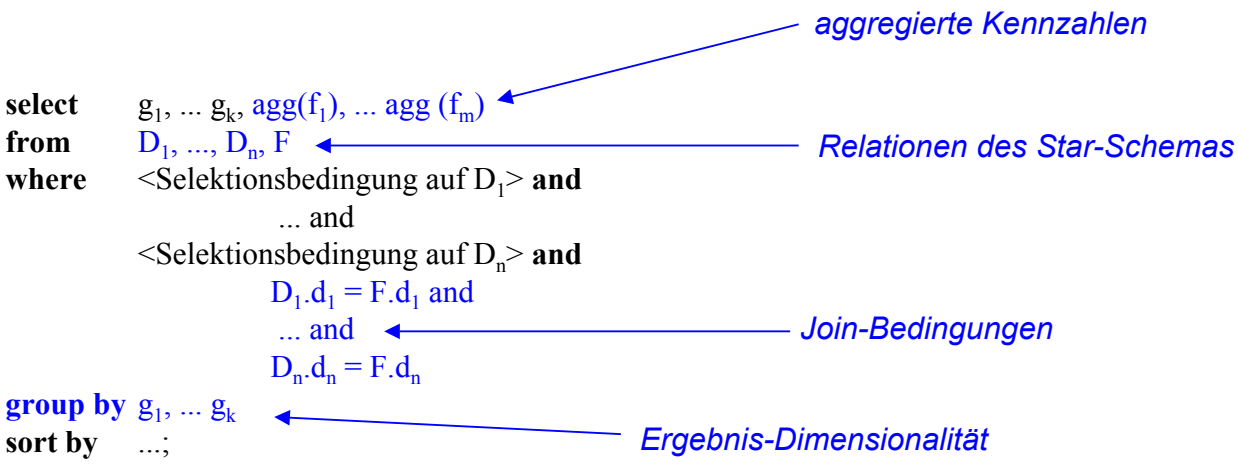
Behandlung von Änderungen in Dimensionen

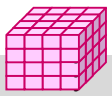
- Änderungsarten
  - neue Dimensionselemente (z.B. neue Produktversion)
  - Änderung von Werten zu einem Dimensionselement (z.B. neuer Familienstand/Wohnort von Kunden)
  - neue Hierarchiestufe einer Dimension
  - neue Dimension
- Behandlung auf Schema-Ebene (Schema-Evolution) oder Tupel-Ebene
- Änderung von Dimensionselementen
  - Lösung 1: Überschreiben der alten Werte (Auswertungen für ältere Zeiträume sind verfälscht)
  - Lösung 2: Versionierung von Dimensionselementen auf Tupel-Ebene, z.B. erweiterte Schlüsselwerte
  - Lösung 3: Versionierung auf Schema-Ebene (Neue Zeitattribute für Gültigkeitszeit, Änderungszeit)



Anfragen auf dem Star-Schema

- Star-Join
  - sternförmiger Join der (relevanten) Dimensionstabellen mit der Faktentabelle
  - Einschränkung der Dimensionen
  - Verdichtung der Kennzahlen durch Gruppierung und Aggregation
- Allgemeine Form



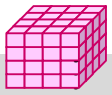


Beispiel eines Star-Join

- In welchen Jahren wurden von weiblichen Kunden in Sachsen im 1. Quartal die meisten Autos gekauft?

```
select    z.Jahr as Jahr, sum (v.Anzahl) as Gesamtzahl
from      Filialen f, Produkt p, Zeit z, Kunden k, Verkauf v
where     z.Quartal = 1 and k.Geschlecht = 'W' and
          p.Produkttyp = 'Auto' and f.Land = 'Sachsen' and
          v.Datum = z.Datum and v.ProduktNr = p.ProduktNr and
          v.Filiale = f. FName and v.KundenNr = k.KundenNr
group by  z.Jahr
order by  Gesamtzahl Descending;
```

Jahr	Gesamtzahl
2004	745
2005	710
2003	650



Mehrdimensionale Aggregationen: Group-By

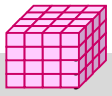
- Attributanzahl in **group by**-Klausel bestimmt Dimensionalität

```
select Hersteller, Jahr, sum (Anzahl) as Anzahl
from Verkauf v, Produkt p, Zeit z
where v.ProduktNr = p.ProduktNr and
v.Datum= z.Datum  and p.Produkttyp = 'Auto'
group by Hersteller, Jahr;
```

Hersteller	Jahr	*Anzahl*
VW	2003	*2.000*
VW	2004	*3.000*
VW	2005	*3.500*
Opel	2003	*1.000*
Opel	2004	*1.000*
Opel	2005	*1.500*
BMW	2003	*500*
BMW	2004	*1.000*
BMW	2005	*1.500*
Ford	2003	*1.000*
Ford	2004	*1.500*
Ford	2005	*2.000*

```
select Hersteller, sum (Anzahl) as Anzahl
from Verkauf v, Produkt p
where v. Produkt = p. ProduktNr and
and p. Produkttyp = 'Auto'
group by Hersteller;
```

```
select sum (Anzahl) as Anzahl
from Verkauf v, Produkt p
where v. Produkt = p. ProduktNr and
p. Produkttyp = 'Auto';
```



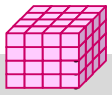
Relationale Speicherung aggregierter Werte

- Kreuztabelle (Crosstab-Darstellung)

Jahr Hersteller	2003	2004	2005	Σ
VW	2.000	3.000	3.500	8.500
Opel	1.000	1.000	1.500	3.500
BMW	500	1.000	1.500	3.000
Ford	1.000	1.500	2.000	4.500
Σ	4.500	6.500	8.500	19.500

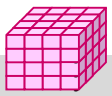
- relationale Darstellung (2D-Cube)

Hersteller	Jahr	*Anzahl*
VW	2003	*2.000*
VW	2004	*3.000*
VW	2005	*3.500*
Opel	2003	*1.000*
Opel	2004	*1.000*
Opel	2005	*1.500*
BMW	2003	*500*
BMW	2004	*1.000*
BMW	2005	*1.500*
Ford	2003	*1.000*
Ford	2004	*1.500*
Ford	2005	*2.000*
VW	**ALL**	*8.500*
Opel	**ALL**	*3.500*
BMW	**ALL**	*3.000*
Ford	**ALL**	*4.500*
ALL	2003	*4.500*
ALL	2004	*6.500*
ALL	2005	*8.500*
ALL	**ALL**	*19.500*



Materialisierung von Aggregaten

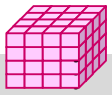
```
create table Auto2DCube (Hersteller varchar (20), Jahr integer, Anzahl integer);

insert into Auto2DCube
    (select p.Hersteller, z.Jahr, sum (v. Anzahl)
     from Verkauf v, Produkt p, Zeit z
     where v.ProduktNr = p.ProduktNr and p.Produkttyp = 'Auto'  and v.Datum = z.Datum
     group by z.Jahr, p.Hersteller)
 union
   (select p.Hersteller, ALL, sum (v.Anzahl)
    from Verkauf v, Produkt p
    where v.ProduktNr = p.ProduktNr and p.Produkttyp = 'Auto'
   group by p. Hersteller)
 union
   (select ALL, z. Jahr, sum (v.Anzahl)
    from Verkauf v, Produkt p, Zeit p
    where v.ProduktNr = p ProduktNr and p.Produkttyp = 'Auto' and v.Datum = z.Datum
    group by z. Jahr)
 union
    (select ALL, ALL, sum (v.Anzahl)
     from Verkauf v, Produkt p
     where v.ProduktNr = p. ProduktNr and p.Produkttyp = 'Auto');
```



Cube-Operator

- SQL-Erweiterung um CUBE-Operator für n-dimensionale Gruppierung und Aggregation
  - Syntax: *Group By CUBE (D_1, D_2, ... D_n)*
  - generiert als Ergebnis eine Tabelle mit aggregierten Ergebnissen (ALL-Tupel)
  - implementiert in MS SQL-Server, DB2, Oracle
- erspart mehrfache Berechnung der Aggregationen
  - erspart 2^n **union**-Anfragen (bei n Attributen in der **group by**-Klausel / n Dimensionen)
  - einfache Formulierung von Anfragen
  - effiziente Berechenbarkeit durch DBS (Wiederverwendung von Zwischenergebnisse)
- Beispiel

select p. Hersteller, z. Jahr, k.Geschlecht, **sum** (v. Anzahl)
from Verkauf v, Produkt p, Zeit z, Kunde k
where v.ProduktNr = p. ProduktNr
and p.Produkttyp = 'Auto' **and** v.Datum = z.Datum
group by cube (p.Hersteller, z.Jahr, k.Geschlecht);

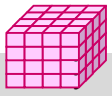


3D-Cube in relationaler Form

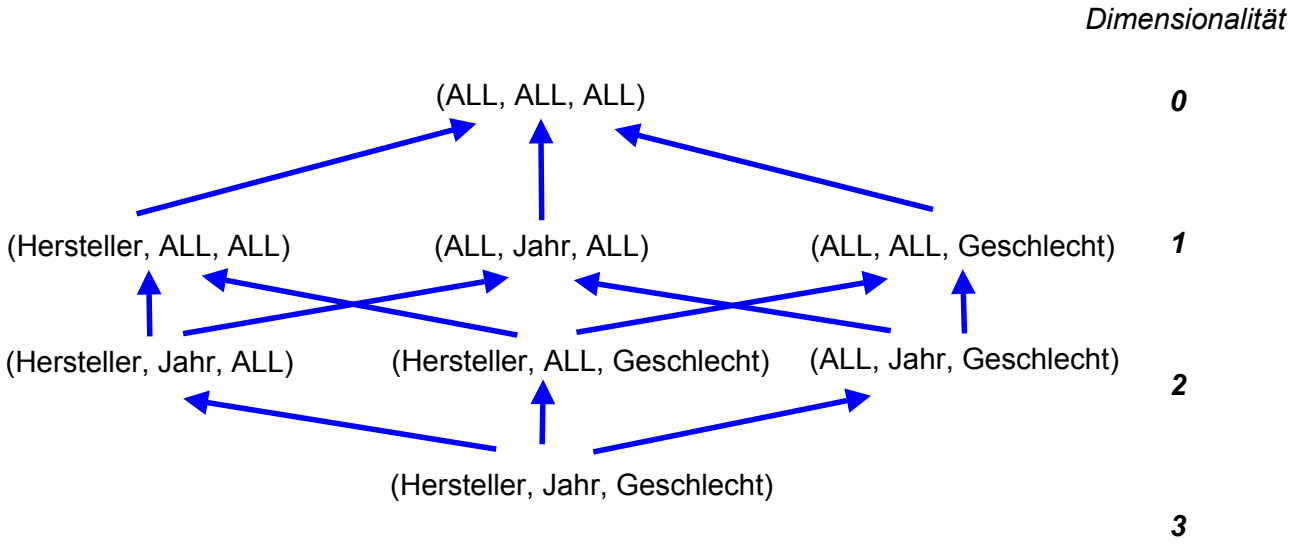
Hersteller	Jahr	Geschl	Anzahl
VW	2003	m	1300
VW	2003	w	700
VW	2004	m	1900
VW	2004	w	1100
VW	2005	m	2300
...	...	...	...
Opel	2003	m	800
Opel	2003	w	200
...	...	...	...
BMW	...	...	...
...	...	...	...

CUBE

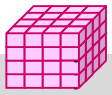
Hersteller	Jahr	Geschl	Anzahl
VW	2003	m	1300
VW	2003	w	700
...	...	...	...
VW	2003	ALL	2.000
...	...	ALL	...
Ford	2005	ALL	2.000
VW	ALL	m	5.500
...	...	...	...
Ford	ALL	w	...
ALL	2001	m	...
...	...		
VW	ALL	ALL	8.500
...	...		
ALL	2001	ALL	...
...			
ALL	ALL	m	...
...			
ALL	ALL	ALL	19.500



Cube-Aggregatgitter



- niedrig-dimensionale Aggregate / Cuboiden können aus höher-dimensionalen abgeleitet werden
- Materialisierung / Caching häufiger benutzter Aggregate ermöglicht Anfrageoptimierung

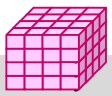


ROLLUP-Operator

- CUBE-Operator: inter-dimensionale Gruppierung / Aggregierung
  - generiert Aggregate für alle 2^n Kombinationsmöglichkeiten bei n Dimensionen
  - zu aufwendig für Roll-Up / Drill-Down innerhalb einer Dimension
- ROLLUP-Operator: intra-dimensionale Aggregierung
- ROLLUP zu a1, a2, ... , an, f ()

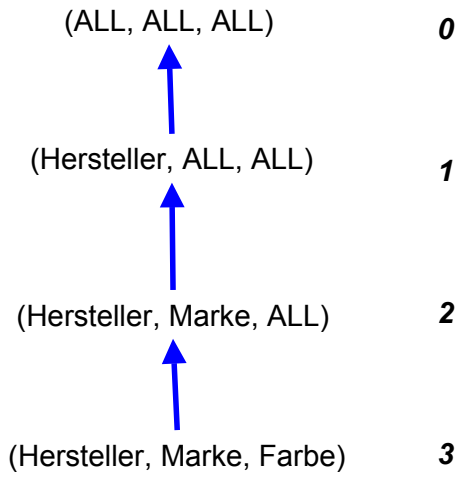
  liefert nur die Cuboide

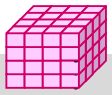
  a1, a2, ... , an-1, ALL, f (),
  ...
  a1, ALL, ... , ALL, f (),
  ALL, ALL, ... , ALL, f ()
- Reihenfolge der Attribute relevant!



ROLLUP-Operator: Beispiel

```
select p. Hersteller, p. Marke, p.Farbe, sum (v. Anzahl)
from Verkauf v, Produkt p
where v.ProduktNr = p. ProduktNr
   and p.Hersteller in („VW“,“Opel“)
group by rollup (p.Hersteller, p.Marke, p.Farbe);
```



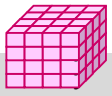


ROLLUP-Beispiel

Hersteller	Marke	Farbe	Anzahl
VW	Passat	rot	800
VW	Passat	weiß	600
VW	Passat	blau	600
VW	Golf	rot	1.200
VW	Golf	weiß	800
VW	Golf	blau	1.000
VW	...	rot	1.400
...	...	...	...
Opel	Vectra	rot	400
Opel	Vectra	weiß	300
Opel	Vectra	blau	300
...	...		

ROLLUP

Hersteller	Marke	Farbe	Anzahl
VW	Passat	rot	800
VW	Passat	weiß	600
VW	Passat	blau	600
VW	Golf	rot	1.200
VW	Golf	weiß	800
VW	Golf	blau	1.000
VW	...	rot	1.400
...	...	...	...
Opel	Vectra	rot	400
Opel	Vectra	weiß	300
Opel	Vectra	blau	300
...	...		
VW	Passat	ALL	2.000
VW	Golf	ALL	3.000
VW	...	ALL	3.500
Opel	Vectra	ALL	1.600
Opel	...	ALL	...
VW	ALL	ALL	8.500
Opel	ALL	ALL	3.500
ALL	ALL	ALL	12.000



Grouping Sets

- mehrere Gruppierungen pro Anfrage

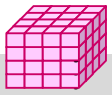
  GROUP BY GROUPING SETS (<Gruppenspezifikationsliste>)
  Gruppenspezifikation: (<Gruppenspezifikationsliste>) |
  CUBE <Gruppenspezifikationsliste> |
  ROLLUP <Gruppenspezifikationsliste>
  leere Spezifikationsliste ( ) möglich: Aggregation über gesamte Tabelle

- Beispiel

```
select p. Hersteller, p.Farbe, sum (v. Anzahl)
from Verkauf v, Produkt p
where v.ProduktNr = p. ProduktNr and
    p.Hersteller in („VW“,“Opel“)
group by grouping sets ((p.Hersteller), (p.Farbe));
```

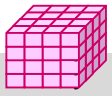
Hersteller	Farbe	*Anzahl*
VW	**ALL**	*8500*
Opel	**ALL**	*3500*
ALL	blau	*3100*
ALL	rot	*6200*
ALL	weiß	*2700*

- CUBE, ROLLUP, herkömmliches Group-By entsprechen speziellen Grouping-Sets



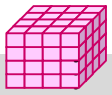
Einzelschritte beim Entwurf eines multi-dimensionalen Schemas

- Welche Geschäftsprozesse sollen modelliert und analysiert werden?
- Festlegung der Kennzahlen
  - Wo kommen sie her?
  - Granularität der Fakten. Welche OLAP-Genauigkeit ist notwendig?
- Bestimmung der Dimensionen
  - Gemeinsame Eigenschaften der Kennzahlen
  - Spezifikation der Dimensionsattribute
  - Konstante vs. sich ändernde Dimensionsattribute
  - Etablierung / Verwendung einer einheitlichen Terminologie
- Physische Design-Entscheidungen
  - Architektur (ROLAP/MOLAP/HOLAP)
  - vorzuberechnende Aggregationen
  - Speicherbedarf ermitteln
- Festlegung der Dauer der Historie, Behandlung alter Daten
- Aktualisierungsfrequenz bezüglich der Quellsysteme



Zusammenfassung

- Einfachheit des multi-dimensionalen Modellierungsansatzes wesentlich für Erfolg von Data Warehousing
  - Cube-Repräsentation mit Kennzahlen und hierarchischen Dimensionen
  - Operationen: Slice and Dice, Roll-Up, Drill-Down, ...
- Multidimensionale Speicherung
  - Problem dünn besetzter Matrizen
  - primär für aggregierte Daten relevant, weniger zur Verwaltung von Detail-Fakten
- Relationale Speicherung auf Basis von Star-Schemas
  - Unterstützung großer Datenmengen, Skalierbarkeit
  - neue Anforderungen bezüglich effizienter Verarbeitung von Star-Joins, mehrdimensionale Gruppierung und Aggregation ...
- Vorberechnung aggregierter Daten wesentlich für ausreichende Leistung
- Sprachansätze
  - MDX-Anweisungen für Cubes
  - SQL-Erweiterungen: CUBE-, ROLLUP-Operator



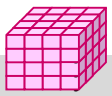
Übungsaufgabe: Warehouse-Entwurf

a) Erstellen Sie ein Star-Schema für ein großes deutsches Telefonunternehmen.

- Es soll Auswertungen über Anrufhäufigkeiten, generierte Umsätze und Dauer der Gespräche für die einzelnen Tarifarten über unterschiedliche Zeiten (Tageszeiten, Wochentage, Monate, Jahre) ermöglichen.
- Die Teilnehmer werden über ihre Adressen bzw. Telefonnummern Orten sowie Bundesländern zugeordnet. Es werden die üblichen Personenmerkmale für Analysezwecke erfasst, insbesondere Alter, Geschlecht und Beruf.

b) Schätzen Sie den Speicherbedarf für eine Aufzeichnungsdauer von 3 Jahren, 40 Millionen Teilnehmern und durchschnittlich 10 Gesprächen pro Tag und Teilnehmer.

c) Wie lautet für das Schema aus a) die SQL-Anfrage zur Bestimmung des Umsatzes aller sächsischen Ferngespräche am Abend nach Tarif AKTIV++ für jeden Monat im Jahr 2007?



Übungsaufgabe 2

- Bestimmen Sie für die gezeigte Tabelle *Goals* das Ergebnis folgender SQL-Anfragen:

Spieler	Saison	*Anzahl*
Elber	1999	*13*
Elber	2000	*14*
Elber	2001	*15*
Scholl	1997	*5*
Scholl	1998	*9*
Scholl	1999	*4*
Scholl	2000	*6*
Scholl	2001	*9*

  - Select Spieler, Saison,
    Sum (Anzahl) as Tore
    From Goals
    GROUP BY ROLLUP (Spieler, Saison);

  - Select Spieler, Saison,
    Sum (Anzahl) as Tore
    From Goals
    GROUP BY CUBE (Spieler, Saison);

  - Select Spieler, Saison, Sum (Anzahl) as Tore
    From Goals
    GROUP BY GROUPING SETS ((Spieler), (Saison),());