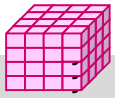
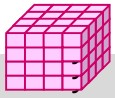
3. Mehrdimensionale Datenmodellierung und Operationen

- 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- und ROLLUP-Operator
  - SQL-Erweiterungen (Group By)**:** Cube, Rollup, Grouping Sets
  - RANK, WINDOW

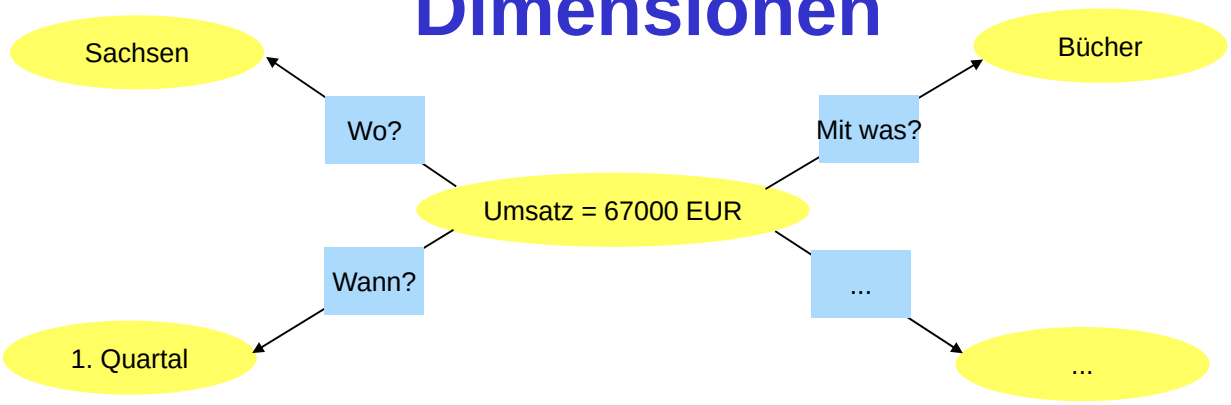


Kennzahlen

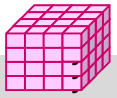
- Kennzahl ist numerische Größe mit konzentrierter Aussagekraft zur Diagnose, Überwachung und Steuerung eines Systems
  - auch: Fakten, Meßgrößen, Measures, Key Performance Indicators (KPI)
  - meist betriebswirtschaftliche Größen, z.B. Umsatz / Gewinn / Rentabilität
  - komplexe Beziehungen zwischen Kennzahlen möglich
  - KPIs oft aus einfacheren Kennzahlen abgeleitet: Umsatz pro Kunde, Liefertreue, Anlagenauslastung, ROI
- Kennzahlen besitzen beschreibende Attribute
  - z.B. Einheit, Wertebereich, Berechnungsvorschrift
- Arten von Kennzahlen
  - *Additive* bzw. *semi-additive* Kennzahlen: additive Aggregation bzgl. aller bzw. nur ausgewählter Dimensionen möglich
  - *nicht-additive* Kennzahlen (Bsp. Durchschnittswerte, Prozentanteile)



Dimensionen

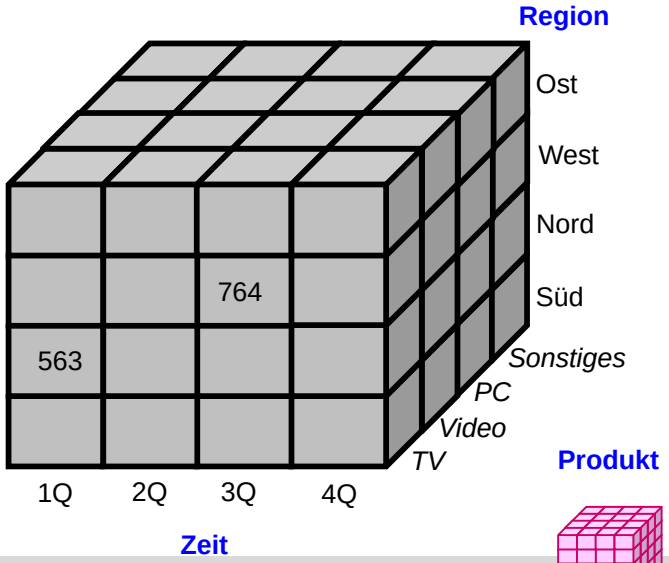


- Zahlenwert einer Kennzahl ist ohne semantischen Bezug nichtssagend
  - Dimensionen setzen Kennzahlen in Bezug zu Eigenschaften / sachlichen Kriterien
- *Dimension*: Datentyp, i.a. endlich (z.B. Aufzählung)
  - Beispiele: Menge aller Produkte, Regionen, Kunden, Zeitperioden etc.
  - *Dimensionselement*: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

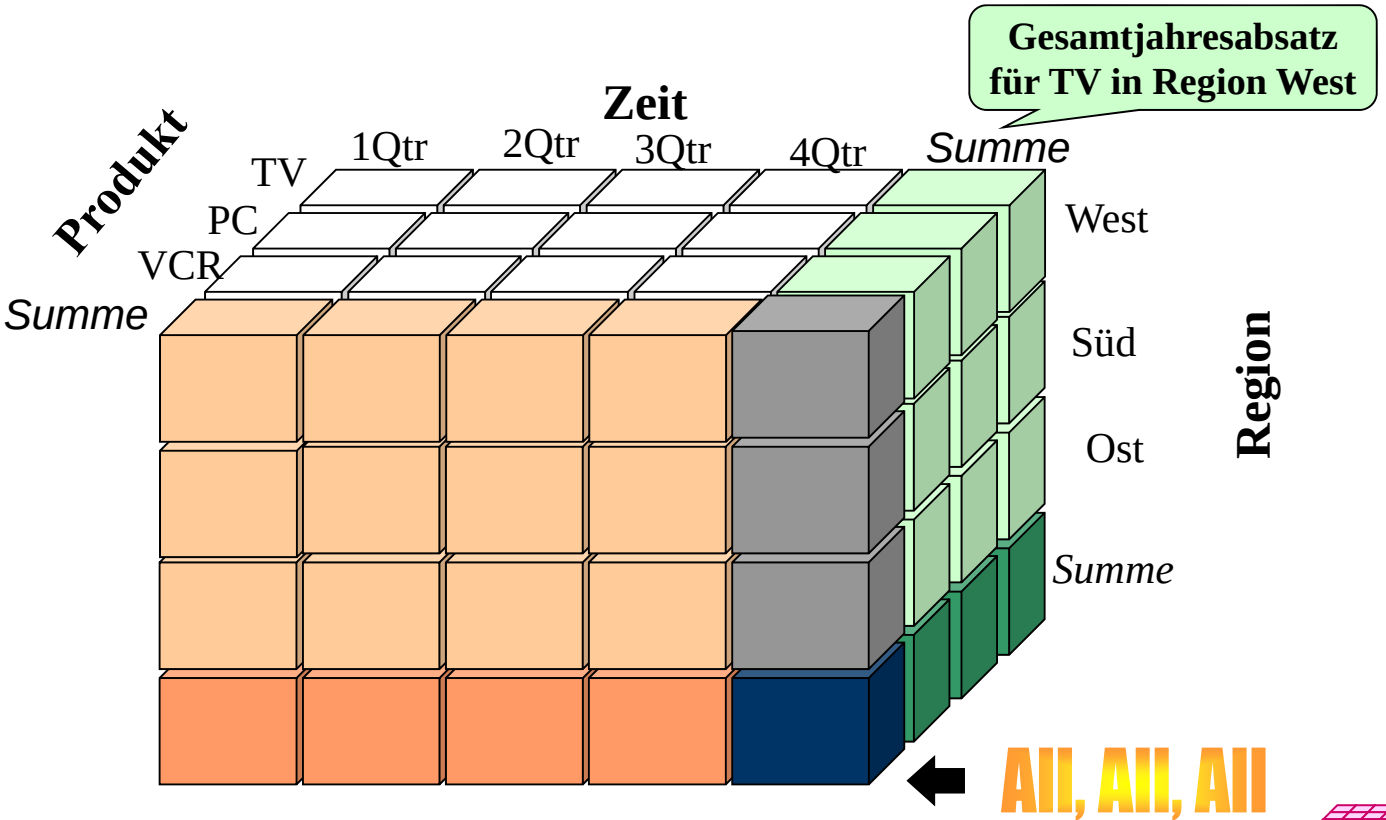


Data Cube

- Datenwürfel bzw. OLAP-Würfel (Cube), Data Cube
  - Dimensionen: Koordinaten
  - Kennzahlen: Zellen im Schnittpunkt der Koordinaten
- Cube bezüglich Dimensionen D_1, ...D_n 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, ...

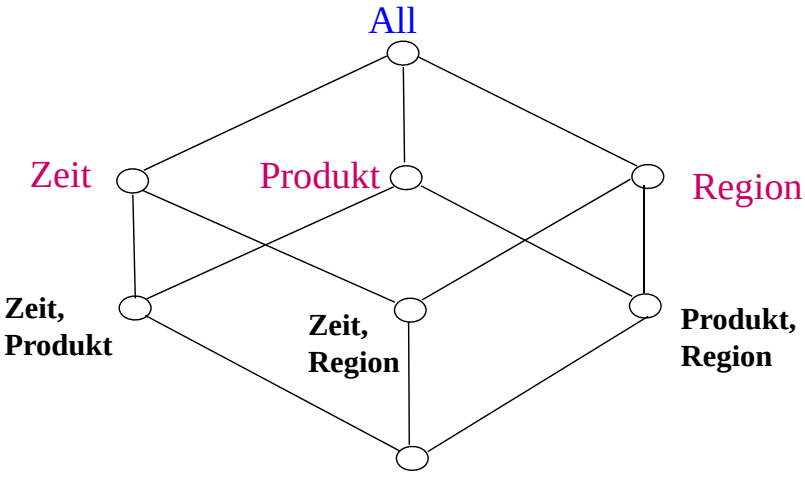


Data Cube: 3D-Beispiel mit Aggregation

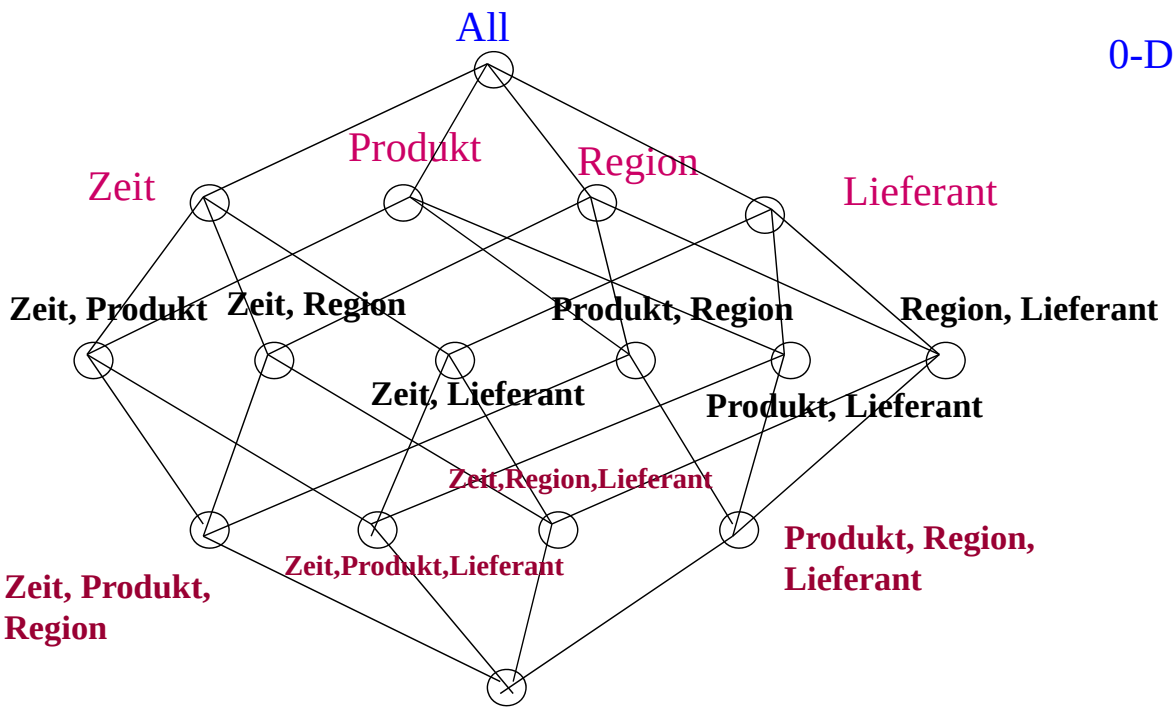


Cube mit Aggregationen

- Aggregationen von Kennzahlen für jede Dimension und Kombination von Dimensionen möglich -> Cuboide
  - Basis-Cuboid: N-dimensionaler Cube
  - Hieraus lassen sich Cuboiden geringerer Dimensionsanzahl ableiten -> Data Cube entspricht Verband (Lattice) von Cuboiden (**Aggregationsgitte**r)
  - N-dimensionaler Cube hat 2^N Cuboiden inkl. Basis-Cuboid (ohne Dimensionshierarchien)
  - Scheitel-Cuboid: 0-dimensionale Aggregation über alle Dimensionen



Cuboid-Verband für 4D Cube



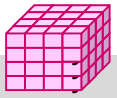
0-D(Scheitel) Cuboid

1-D Cuboide

2-D Cuboide

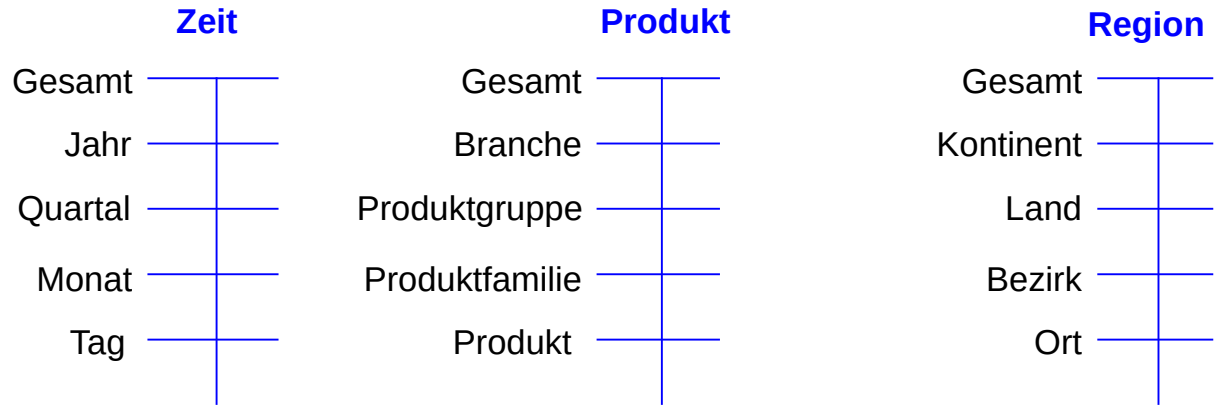
3-D Cuboide

4-D (Basis)-Cuboid

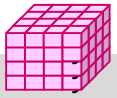


Dimensionshierarchien (Konzepthierarchien)

- 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

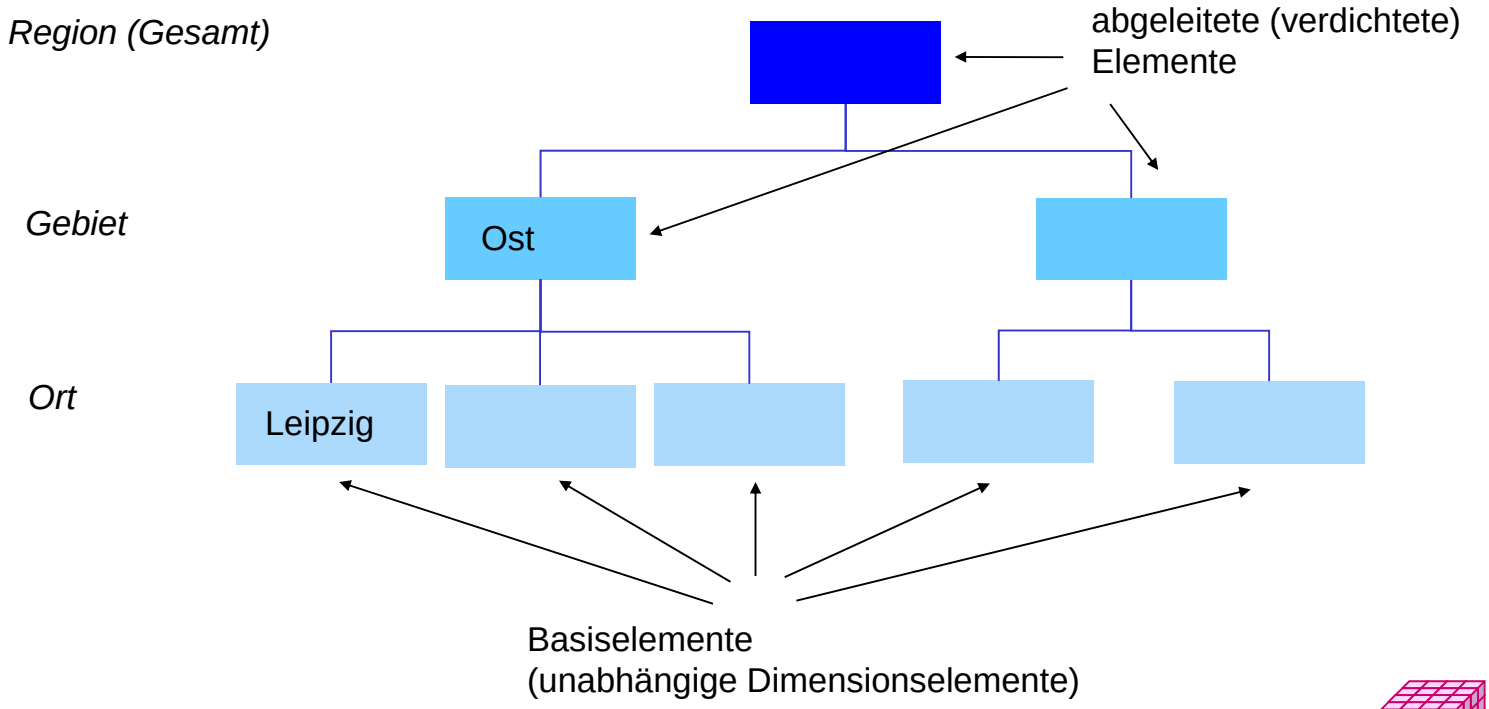


- Dimensionen können neben Klassifikations(Hierarchie)attributen noch beschreibende *dimensionale Attribute* aufweisen



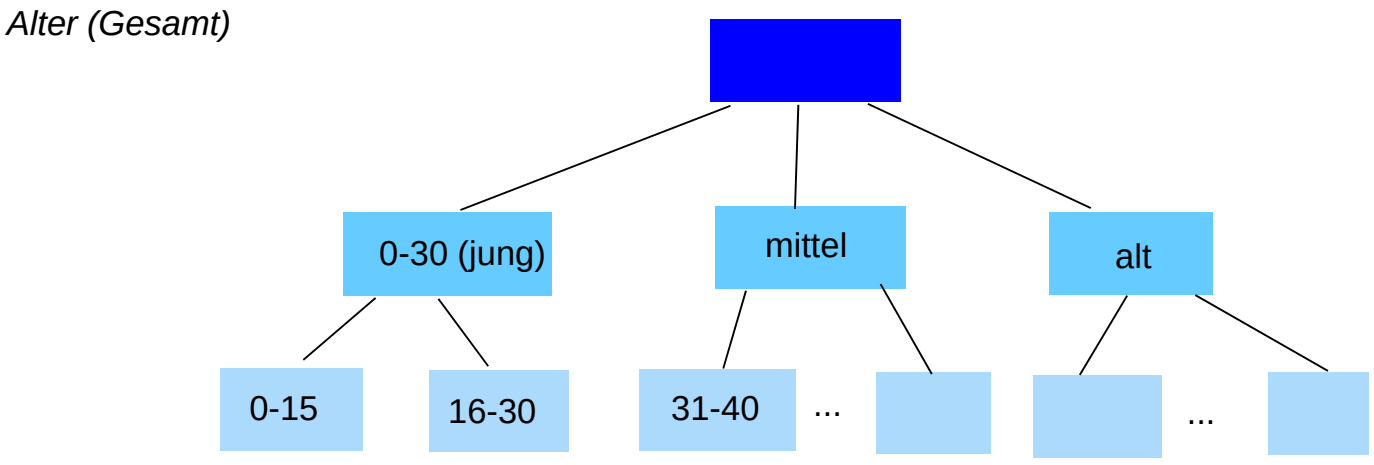
Beispiel einer Konzepthierarchie (Region)

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

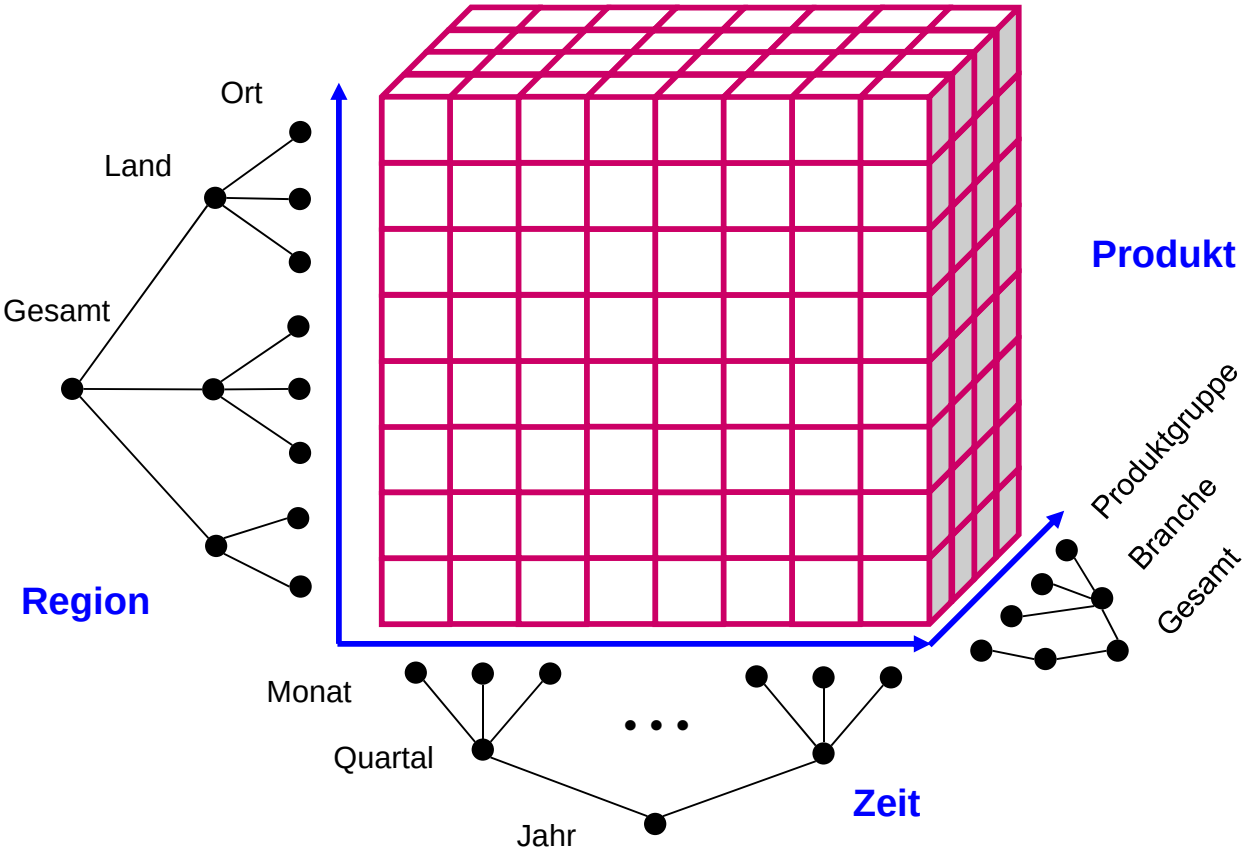


Konzepthierarchien (3)

- 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

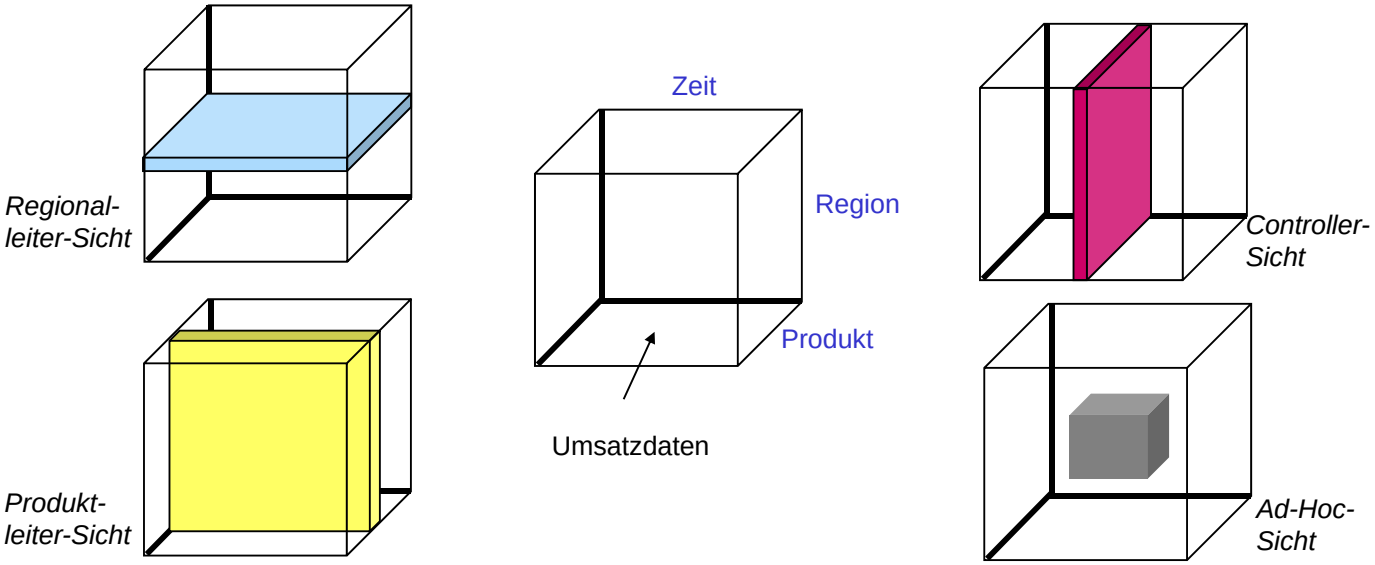


Cube mit hierarchischen Dimensionen

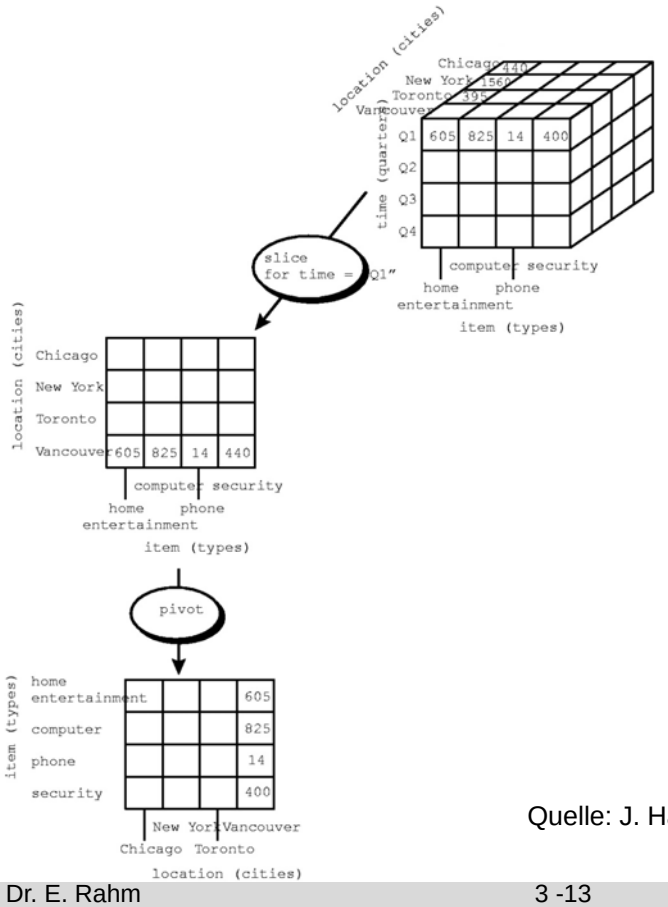


Operationen auf Cubes

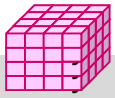
- Slice: Herausschneiden von „Scheiben“ aus dem Würfel durch Einschränkung (Selektion) auf einer Dimension
  - Verringerung der Dimensionalität
- Dice: Herausschneiden einen „Teilwürfels“ durch Selektion auf mehreren Dimensionen
- unterschiedlichste mehrdimensionale Aggregationen / Gruppierungen
- Pivot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



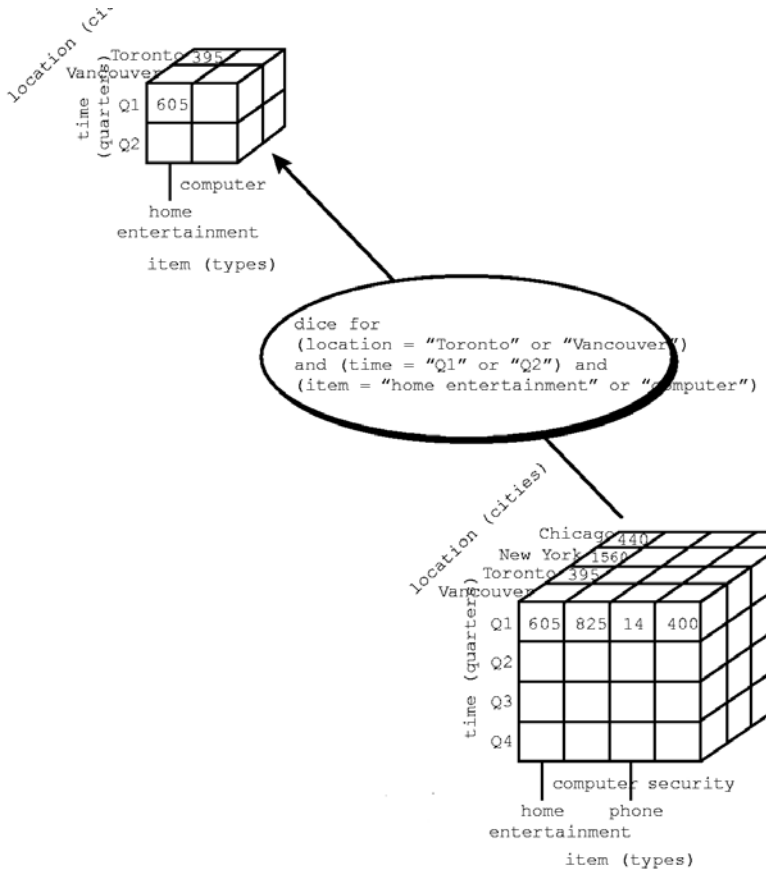
Beispiele: Slice / Pivot

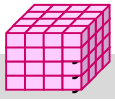


Quelle: J. Han, M. Kamber: Data Mining, Morgan Kaufmann



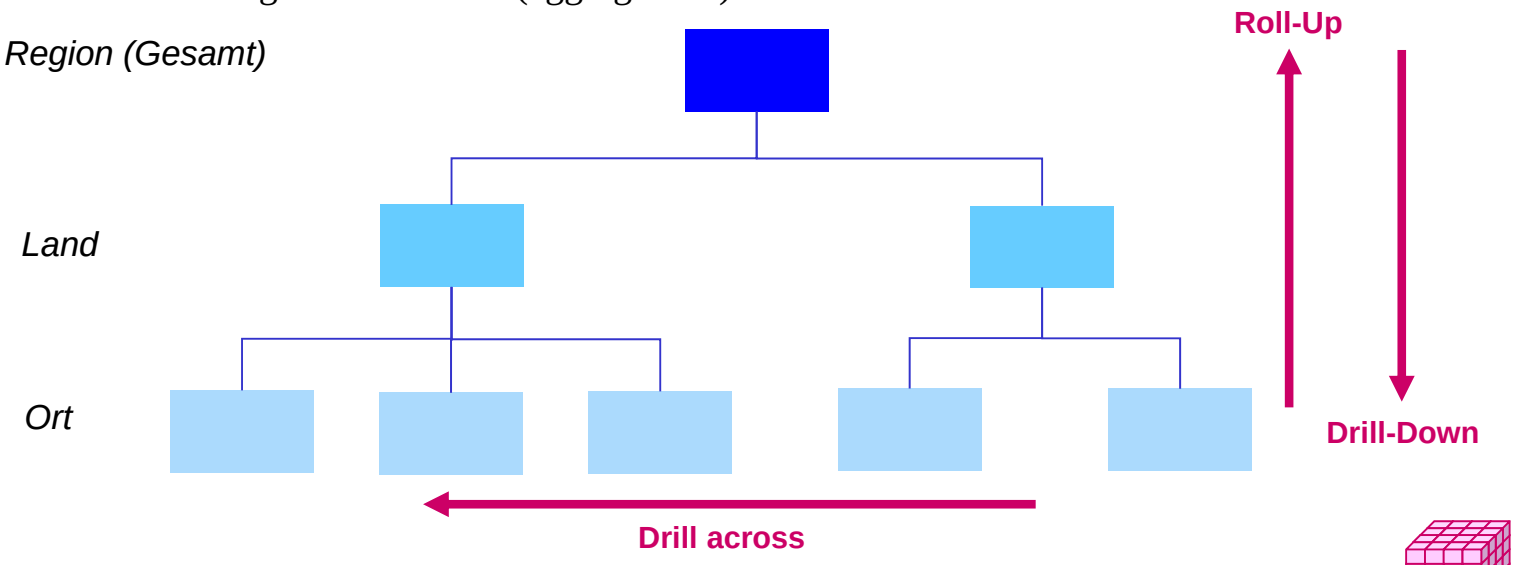
Beispiel: Dice



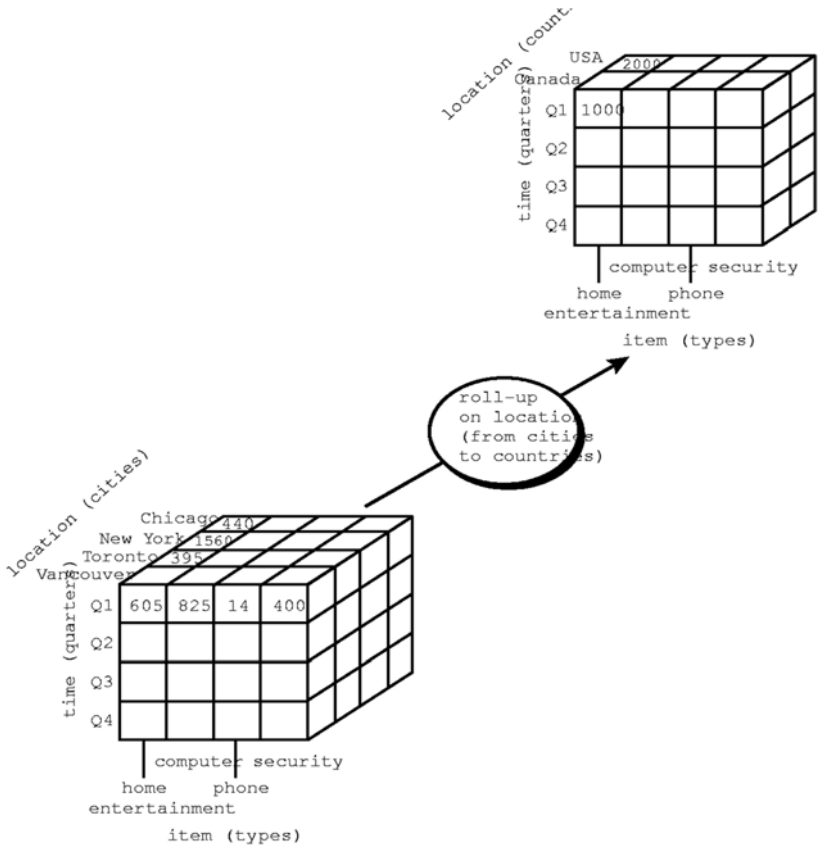


Navigation in Hierarchien

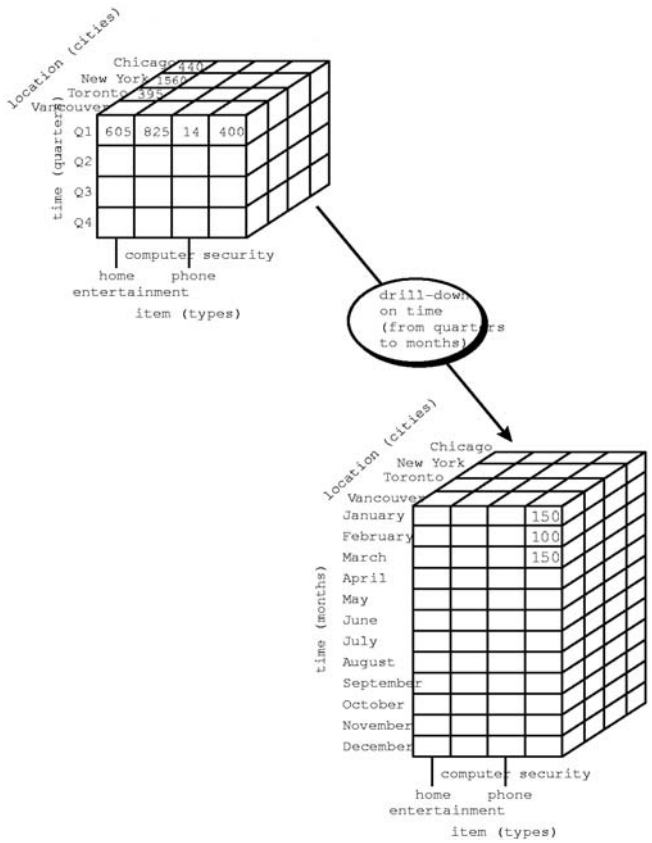
- Drill-Down
  - Navigation nach „unten“ in der Hierarchie
  - Erhöhung des Detailgrad: von verdichteten Daten zu weniger verdichteten/aggregierten Daten
- Roll-Up (Drill-Up)
  - Navigation nach „oben“ in der Hierarchie
  - von weniger verdichteten (aggregierten) Daten zu stärker verdichteten Daten



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**

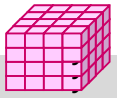
Elektronik	*Ost*	*Süd*	*Nord*	*West*
TV	800			
DVD-Player	600			
Camcorder	400			

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

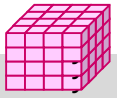


Größe der Cubes

- 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: $T = \prod_{i=1}^{n} (L_i + 1)$

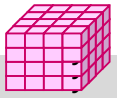
  L_i: #Ebenen von Dimension i (ohne Top-Level)

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)

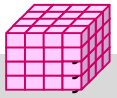
	Sachsen	Brandenburg	Thüringen
TV	100	150	200
DVD-Player	50	170	150
Camcorder	20	120	100

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)

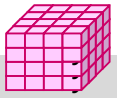
REGION

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



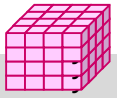
Sprachansatz MDX*

- MDX: MultiDimensional eXpressions
  - Microsoft-Spezifikation für Cube-Zugriffe / Queries
  - 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

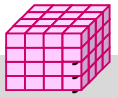
* http://msdn.microsoft.com/en-us/library/ms145506.aspx



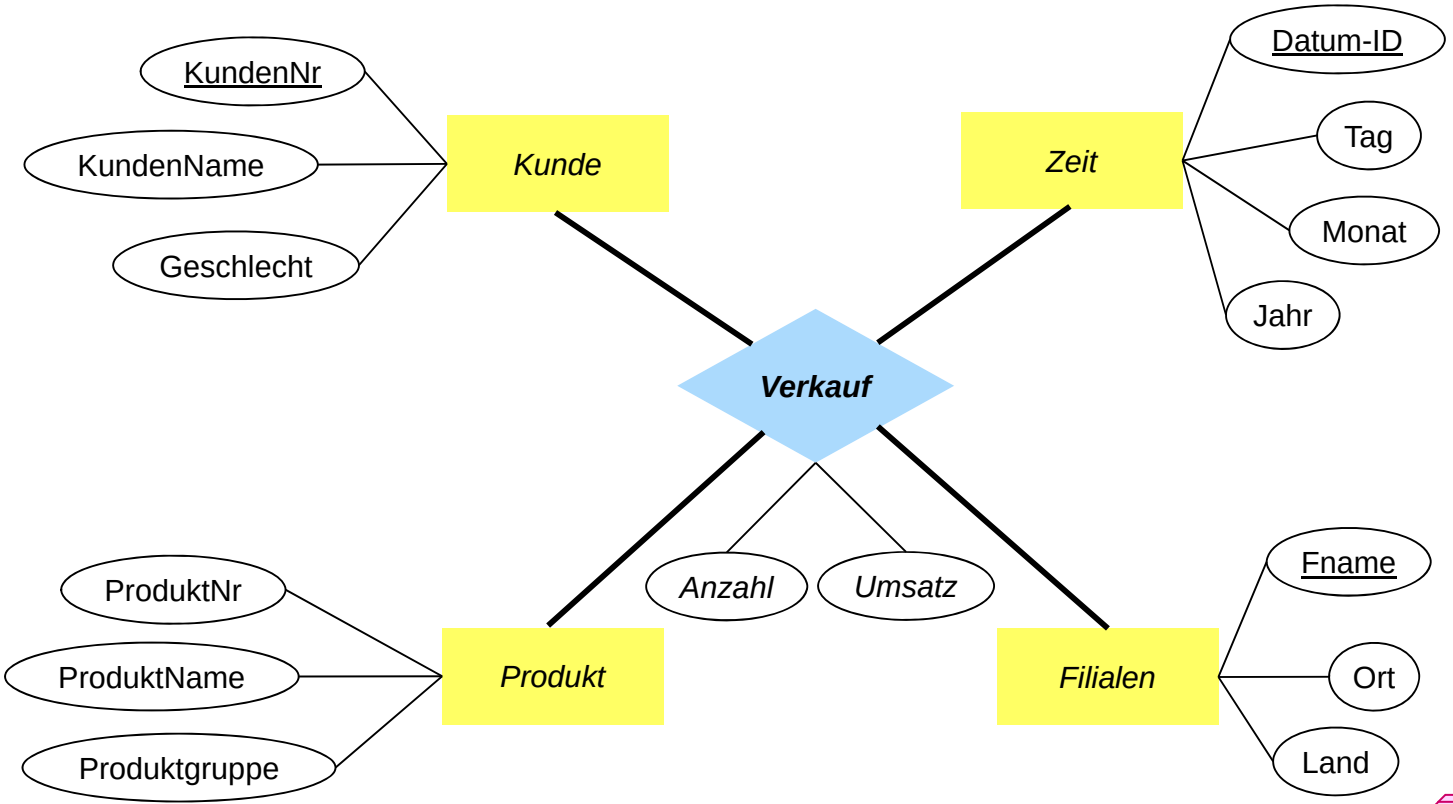
MDX: Beispiele

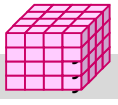
```
SELECT  Region.CHILDREN ON COLUMNS,
        Produkt.CHILDREN ON ROWS
FROM    Verkauf
WHERE   (Umsatz, Zeit.[2011])
```

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



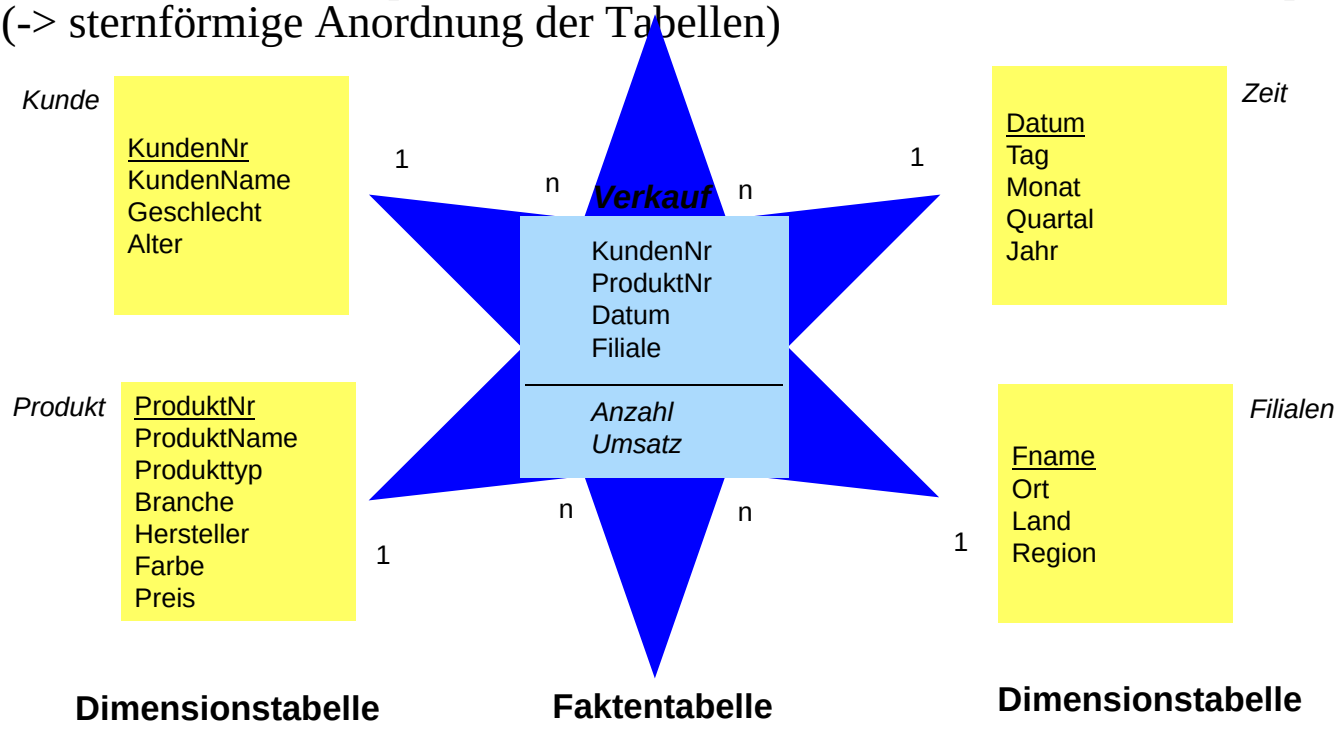
ER-Diagramm eines multi-dimensionalen Datenmodells





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 ...	26900 ...

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

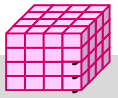
Kunde			
KundenNr	Name	Geschlecht	Alter
4711 ...	Weber ...	M ...	39 ...

Zeit					
Datum	Tag	Monat	Jahr	Quartal	...
7654 ...	25 ...	April ...	2012 ...	2 ...	...

Produkt					
ProduktNr	Produktname	Produkttyp	Hersteller	Farbe	Preis
1847 ...	Passat XY ...	Auto ...	VW ...	Blau ...	31999 ...

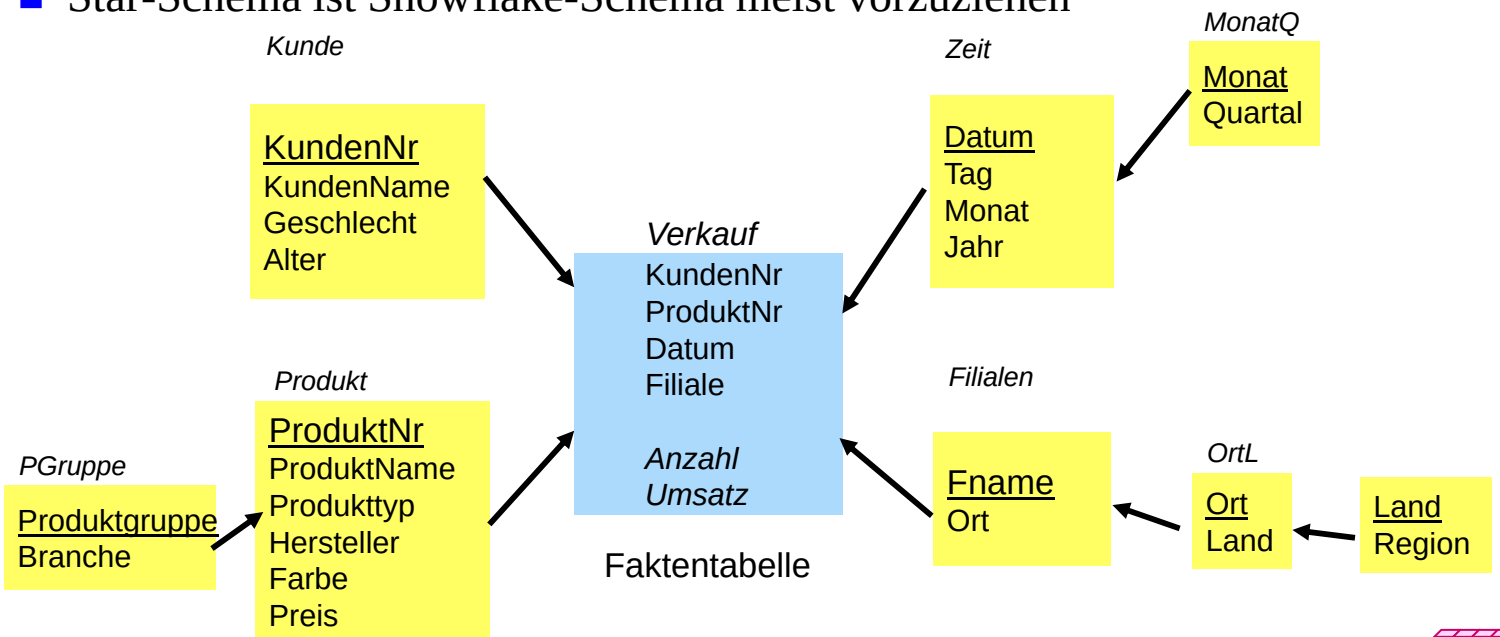
Star-Schema (2)

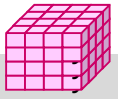
- Formale Definition: Star-Schema besteht aus einer Menge von Tabellen D_1, ...D_n, F mit
  - Dimensionstabellen D_i bestehend aus (i.a. künstlichen) Primärschlüssel d_i und Dimensionsattributen
  - Faktentabelle F bestehend aus Fremdschlüsseln d_1, ... d_n sowie Meßgrößen (Kennzahlen) als weiteren Attributen
  - 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

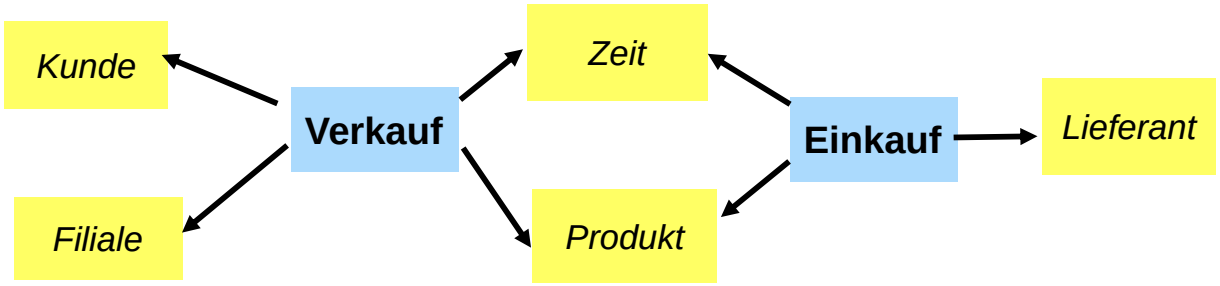


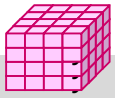


Galaxien-Schema

- Data Warehouses benötigen meist mehrere Faktentabellen

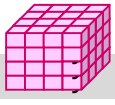
  -> Multi-Star-Schema (Galaxien-Schema, „Fact Constellation Schema“)
- gemeinsame Nutzung von Dimensionstabellen
- Speicherung vorberechneter Aggregate
  - separate Faktentabelle
  - im Rahmen der Faktentabelle mit Detail-Daten





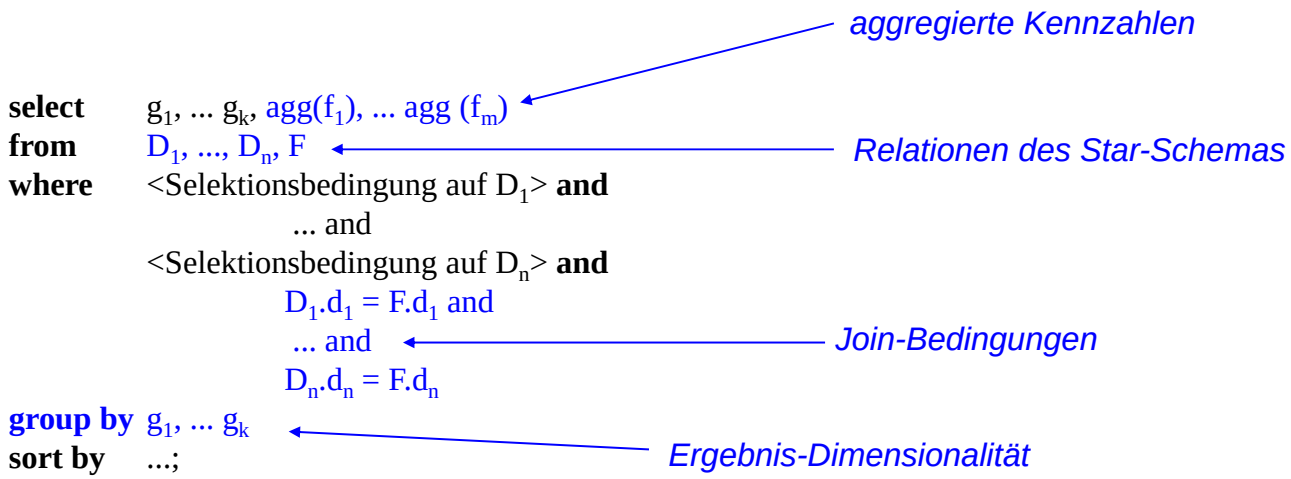
Behandlung von Änderungen in Dimensionen

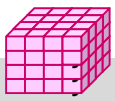
- Änderungsarten
  - neue Dimensionselemente (z.B. neue Produktversion)
  - Änderung von Werten zu einem Dimensionselement (z.B. neuer Familienstand/Wohnort von Kunden)
  - neue Hierarchiestufe einer Dimension
  - neue Dimension
- Behandlung auf Schema-Ebene (Schema-Evolution) oder Tupel-Ebene
- Änderung 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



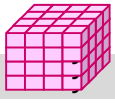


Beispiel eines Star-Join

- 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
2012	745
2013	710
2011	650



Mehrdimensionale Aggregationen mit 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;
```

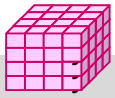
Hersteller	Jahr	*Anzahl*
VW	2011	*2.000*
VW	2012	*3.000*
VW	2013	*3.500*
Opel	2011	*1.000*
...	...	...
BMW	2013	*1.500*
Ford	2011	*1.000*
Ford	2012	*1.500*
Ford	2013	*2.000*

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

Hersteller	*Anzahl*
VW	*8.500*
Opel	*3.500*
Ford	*4.500*
BMW	*3.000*

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

Anzahl
19.500



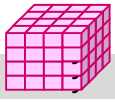
Relationale Speicherung aggregierter Werte

- Kreuztabelle (Crosstab-Darstellung)

Jahr / Hersteller	2011	2012	2013	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

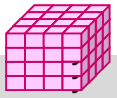
- relationale Darstellung (2D-Cube)

Hersteller	**Jahr**	***Anzahl***
VW	2011	*2.000*
VW	2012	*3.000*
VW	2013	*3.500*
Opel	2011	*1.000*
Opel	2012	*1.000*
Opel	2013	*1.500*
BMW	2011	*500*
BMW	2012	*1.000*
BMW	2013	*1.500*
Ford	2011	*1.000*
Ford	2012	*1.500*
Ford	2013	*2.000*
VW	**ALL**	*8.500*
Opel	**ALL**	*3.500*
BMW	**ALL**	*3.000*
Ford	**ALL**	*4.500*
ALL	2011	*4.500*
ALL	2012	*6.500*
ALL	2013	*8.500*
ALL	**ALL**	*19.500*



Materialisierung von Aggregaten

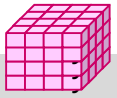
```
create table Auto2DCube (Hersteller varchar (20), Jahr integer, Anzahl integer);

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);
```

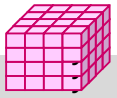


3D-Cube in relationaler Form

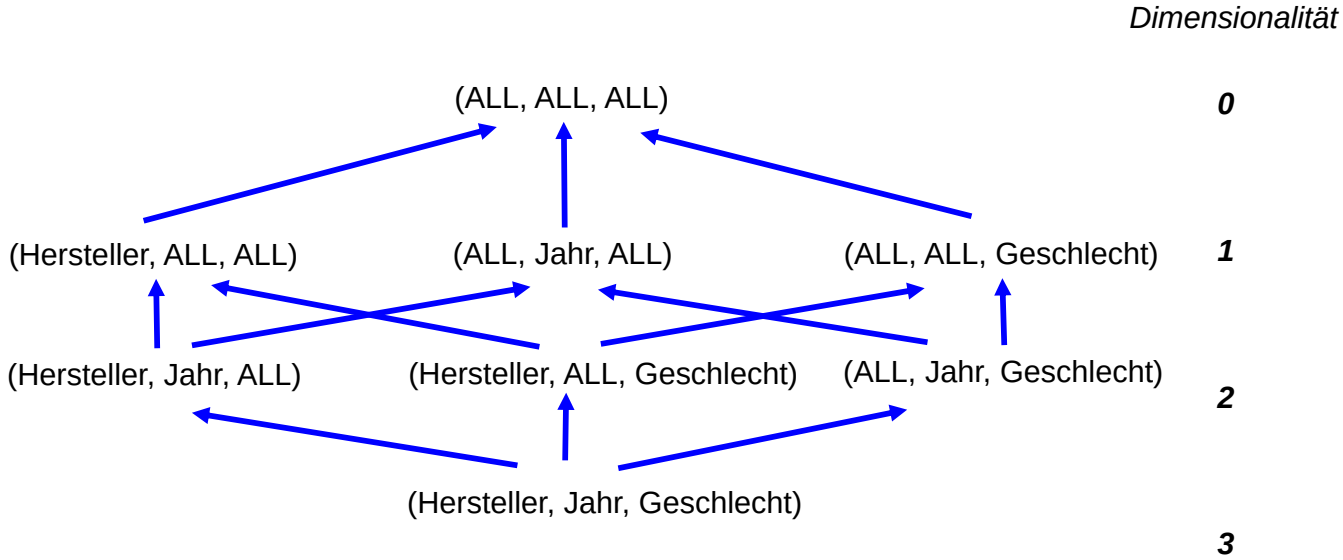
Hersteller	Jahr	Geschl	Anzahl
VW	2011	m	1300
VW	2011	w	700
VW	2012	m	1900
VW	2012	w	1100
VW	2013	m	2300
...	...	...	...
Opel	2011	m	800
Opel	2011	w	200
...	...	...	...
BMW	...	...	...
...	...	...	...

CUBE

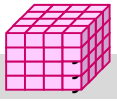
Hersteller	Jahr	Geschl	Anzahl
VW	2011	m	1300
VW	2011	w	700
...	...	...	...
VW	2011	ALL	2.000
...	...	ALL	...
Ford	2013	ALL	2.000
VW	ALL	m	5.400
...	...	...	...
Ford	ALL	w	...
ALL	2011	m	...
...	...		
VW	ALL	ALL	8.500
...	...		
ALL	2011	ALL	...
...			
ALL	ALL	m	...
...			
ALL	ALL	ALL	19.500



Cube-Aggregatgitter



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

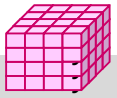


ROLLUP-Operator

- 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 ()

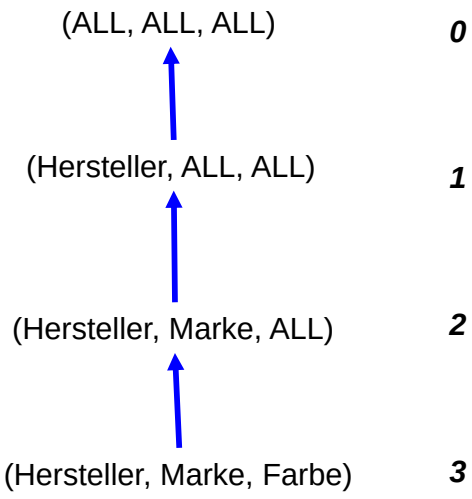
  liefert nur die Cuboide

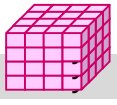
  a1, a2, ... , an-1, ALL, f (),
  ...
  a1, ALL, ... , ALL, f (),
  ALL, ALL, ... , ALL, f ()
- Reihenfolge der Attribute relevant!



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);



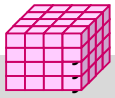


ROLLUP-Beispiel

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>

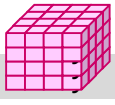
  leere Spezifikationsliste ( ) möglich: Aggregation über gesamte Tabelle

- 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

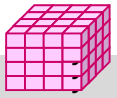


Statistische Funktionen in SQL

- Varianz
  - VAR_POP (bzgl gesamter Population/Eingabepartition)
  - VAR_SAMP (berücksichtigt Bessel-Korrektur)
- Standardabweichung; STDDEV_POP, STDEV_SAMP
- Kovarianz: COVAR_POP, COVAR_SAMP
- Korrelationskoeffizient: CORR

```
select p.Produkttyp, CORR (p.Verkaufspreis, p.Einkaufspreis)
from Verkauf v, Produkt p
where v.ProduktNr = p. ProduktNr
group by p.Produkttyp
```

- Lineare Regressionsanalysen
  - REGR_SLOPE (Anstieg Regressionsgerade)
  - REGR_COUNT (Anzahl berücksicht. Wertepaare ungleich NULL)
  - REGR_R2 (Regr.koeffizienz)
  - REGR_AVGX, REGR_AVGY (Mittelwerte der X/Y-Parameter)

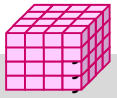


Rank/Windowing-Funktionen

- Standardisiert seit SQL:1999 / 2003
- Erweiterte Analysemöglichkeiten
  - Berechnung von Rangfolgen / Top-N
  - Kumulierte Häufigkeiten/Anteile (z.B. bezüglich eines Jahres/Monats)
  - Vergleiche, z.B. Monatsumsatz gegenüber gleitendem 3-Monatsdurchschnitt …
- Nutzung eines OVER-Prädikats in Select-Klausel zur Bezugnahme auf Datenstrom /Sequenz

```
WFctType(expr) OVER (WPartitioning WOrdering Wframe)
```

  - WFctType: SUM, AVG, RANK, DENSE_RANK …
  - Wpartitioning: optionale PARTITION-Klausel
  - WOrdering: optionale ORDER BY-Klausel
  - Wframe: optionale tupelweise (ROWS) oder wertebereichsweise (RANGE) Einschränkung für Aggregationsfenster in Verbindung mit ORDER BY
- Alternative WINDOW-Klausel

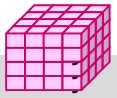


Rank-Funktion

```
select Hersteller, Anzahl,
    rank() over (order by Anzahl desc) as Rang,
    dense_rank() over (order by Anzahl desc) as DRang
from verkauf natural join produkt natural join zeit
where  jahr=2011
order by Anzahl desc, Hersteller
```

Hersteller	Anzahl	Rang	DRang
VW	2000	1	1
Ford	1000	2	2
Opel	1000	2	2
BMW	500	4	3

DENSE_RANK überspringt keine Nummer



Rank-Funktion (2)

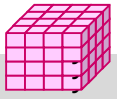
- Partitionsweises Ranking

```
select Hersteller, Anzahl,  rank() over
        (partition by Jahr order by Anzahl desc) as Rang,
from verkauf natural join produkt natural join zeit
order by Jahr, Rang
```

- Weitere Ranking-Funktionen
  - **percent_rank():** relativer Anteil pro Partition (zwischen 0 und 1)
  - **percentile_cont(p), percentile_disc(p):** Perzentile (Prozentränge) für kontinuierliche bzw. gleichmäßige Verteilung der Attributwerte innerhalb einer Gruppe (WITHIN GROUP- und ORDER BY-Klauseln)

Beispiel: Median der Verkaufspreise pro Hersteller

```
select Hersteller, percentile_disc(0.5) within group (order by Verkaufspreis)
        over (partition by Hersteller)  as Preismedian,
from verkauf natural join produkt
```



Aggregatberechnung auf Windows

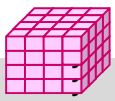
- Nutzung von SUM, AVG etc. in Verbindung mit OVER
- Anwendungsbeispiel für Tabelle *sales (date, value)*

Summe der Verkäufe pro Tag sowie Anteil an Gesamtsumme

```
select date, sum(value) as day_sum,
   sum(value) over () as all_sum,
  100.0*day_sum/all_sum as anteil
from sales
group by date
```

Summe der Verkäufe eines Tages im Verhältnis zu Verkäufen des Jahres

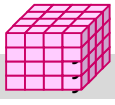
```
select date, sum(value) as day_sum,
   sum(value) over (partition by year(date)) as year_sum,
  100.0*day_sum/year_sum as janteil
from sales
group by date
```



Beispieldaten

date	value
1.2.2011	500
1.2.2011	300
5.7.2011	200
2.3.2012	400
3.4.2012	100
9.6.2013	500

date	day_sum	all_sum	anteil
1.2.2011			
5.7.2011			
2.3.2012			
3.4.2012			
9.6.2013			



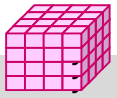
Bsp.: Kumulierte Summe

- Aggregatfunktion vor OVER aggregiert bei Order By vom ersten bis zum aktuellen Tupel
- Nutzbar zur Berechnung einer kumulierten Summe

Beispiel für Tabelle *sales (date, value)*

Summe der Verkäufe pro Tag sowie die kumulierten Gesamtverkäufe nach Tagen und die kumulierten Verkäufe im jeweiligen Jahr nach Tagen sortiert

```
select date, sum(value) AS day_sum,
  sum(value) over (order by date) as cum_sum,
  sum(value) over (partition by year(date) order by date) as cumy_sum
from sales
group by day
order by day
```

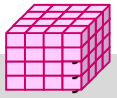


Windowing mit expliziter Fensterangabe

- Beispiel Moving Average für Tabelle *sales (date, value)*
  Berechne pro Datum durchscnittlichen Umsatz für diesen Tag, den vorhergehenden Tag sowie den nächsten Tag

```
select date, avg(value) over
        (order by date between rows 1 preceding and 1 following)
 from sales
```

- Weitere dynamische Windows-Spezifikationen
  - **rows unbounded preceding** (alle Vorgänger inkl. aktuellem Tupel)
  - **rows unbounded following** (alle Nachfolger inkl. aktuellem Tupel)
  - **rows between 2 preceding and 2 following**
    (Fenster von 5 Sätzen, zB 5 Tage, Monate etc.)
  - **rows 3 following** (aktueller Satz und maximal 3 nachfolgende Sätze)
  - **range between interval '10' day preceding and current row**
    (wertebasierter Bereich)



Explizites Windowing (2)

- Partitionsweises Windowing

  Tabelle *transaction* (*account-number, date-time, amount*)

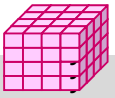
  Amount ist positiv für Zubuchung, negativ für Abbuchung

  Bestimme Kontostand (kumulierte Summe) pro Konto nach jeder Kontobewegung

```
select account-number, date-time,
    sum (amount) over
        (partition by account-number   order by date-time
         rows unbounded preceding ) as balance
from transaction
order by account-number, date-time
```

- Alternative Formulierung mit WINDOW-Klausel

```
select account-number, date-time,   sum (amount) over  w  as balance
from transaction
order by account-number, date-time
window w as (partition by account-number   order by date-time
             rows unbounded preceding )
```



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
  - 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, GROUPING SETS, Rank/Windowing-Funktionen …