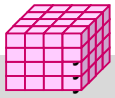
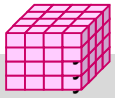
3. Mehrdimensionale Datenmodellierung und Operationen

- Grundlagen
  - Kennzahlen, Dimensionen, Cube
  - Cuboide / **Aggregationsgitter**
  - hierarchische Dimensionen / Konzepthierarchien
  - Cube-Operationen
- multi-dimensionale Speicherung (MOLAP)
  - MDX-Abfragen
- relationale Repräsentation mehrdimensionaler Daten (ROLAP)
  - Star-Schema
  - Varianten: Snowflake-, Galaxien-Schema
- Anfragen an relationale Data Warehouses
  - Star Join, Roll-Up, Drill-Down
  - SQL-Erweiterungen (Group By)**:** Cube, Rollup, Grouping Sets
  - RANK-, WINDOW-Queries

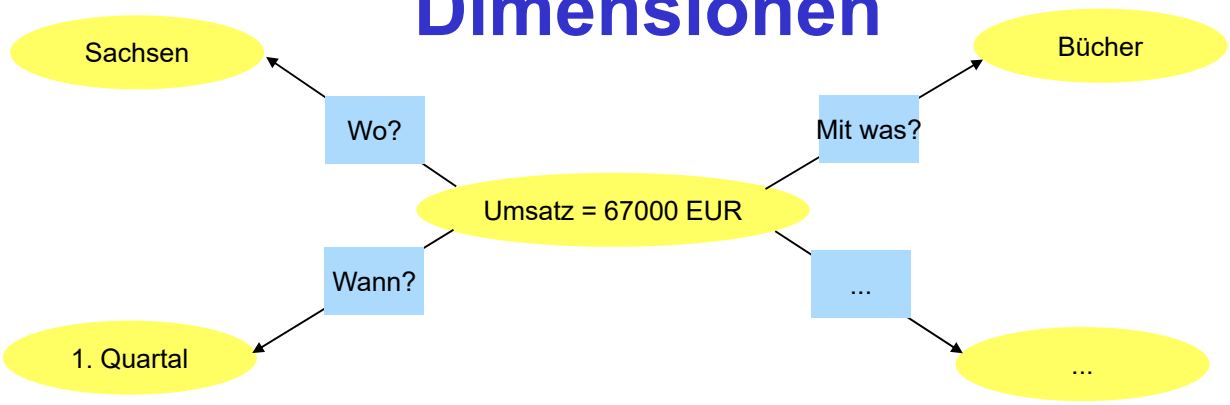


Kennzahlen

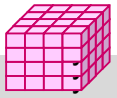
- Kennzahl ist numerische Größe mit konzentrierter Aussagekraft zur Diagnose, Überwachung und Steuerung eines Systems
  - auch: Fakten, Meßgrößen, Measures, Key Performance Indicators (KPI)
  - meist betriebswirtschaftliche Größen, z.B. Umsatz / Gewinn / Rentabilität
  - komplexe Beziehungen zwischen 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

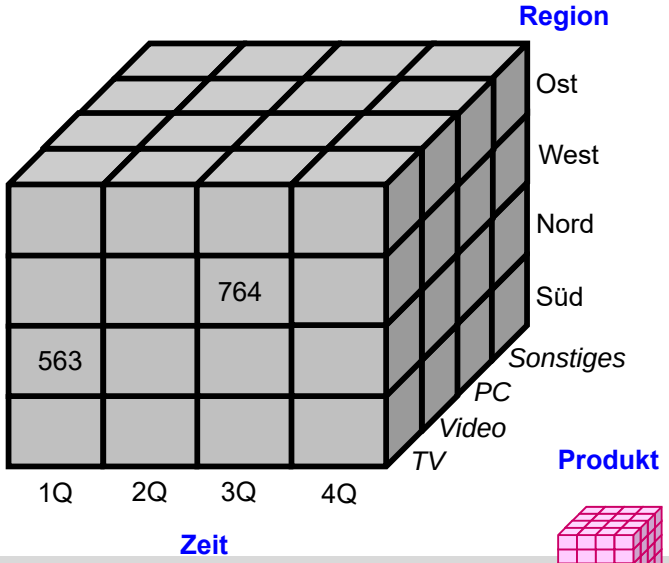


- Zahlenwert einer Kennzahl ist ohne semantischen Bezug nichtssagend
  - Dimensionen setzen Kennzahlen in Bezug zu Eigenschaften / sachlichen Kriterien
- *Dimension*: Datentyp, i.a. endlich (z.B. Aufzählung)
  - Beispiele: Menge aller Produkte, Regionen, Kunden, Zeitperioden etc.
  - *Dimensionselement*: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

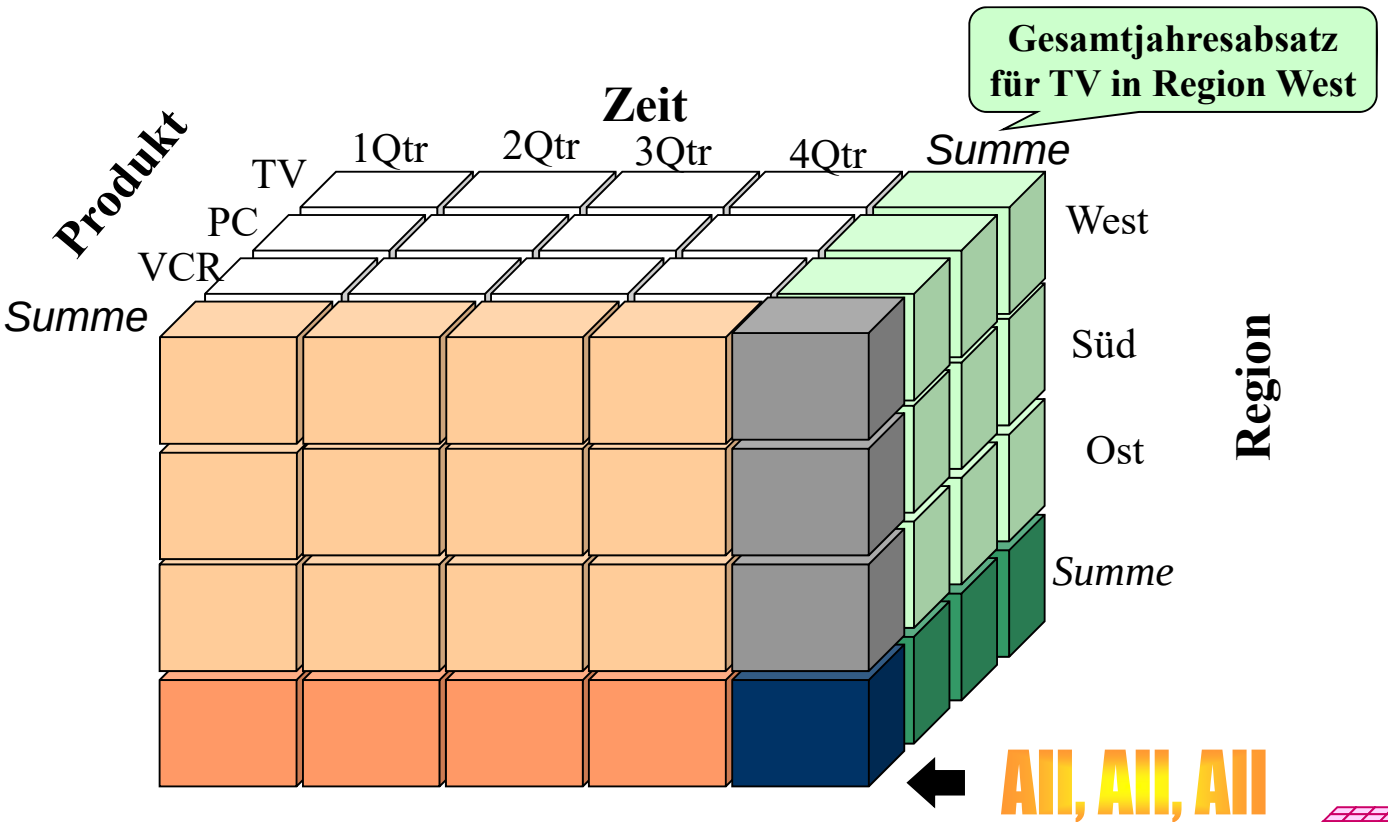


Data Cube

- 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, ...

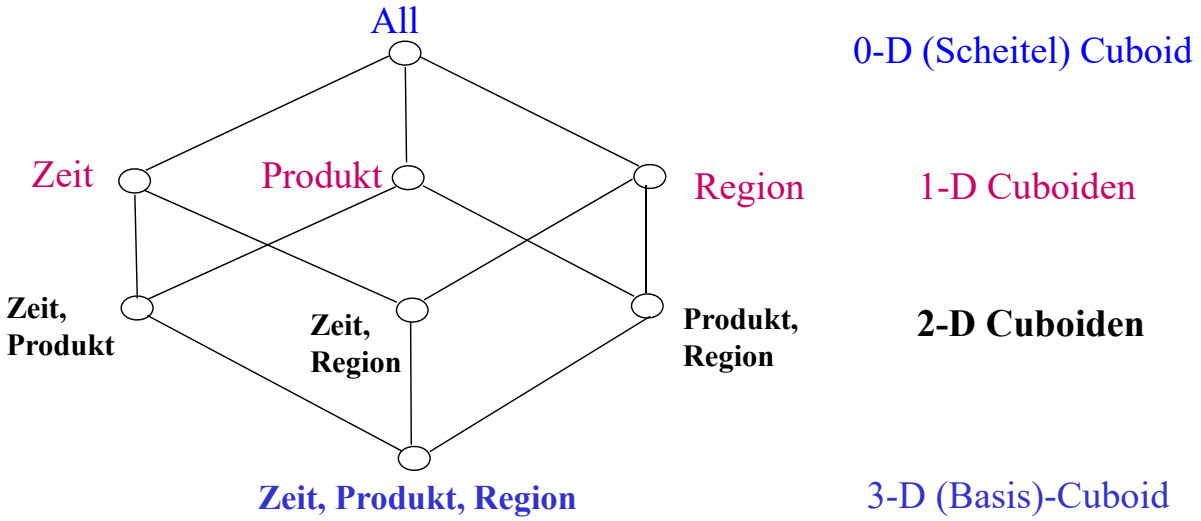


Data Cube: 3D-Beispiel mit Aggregation

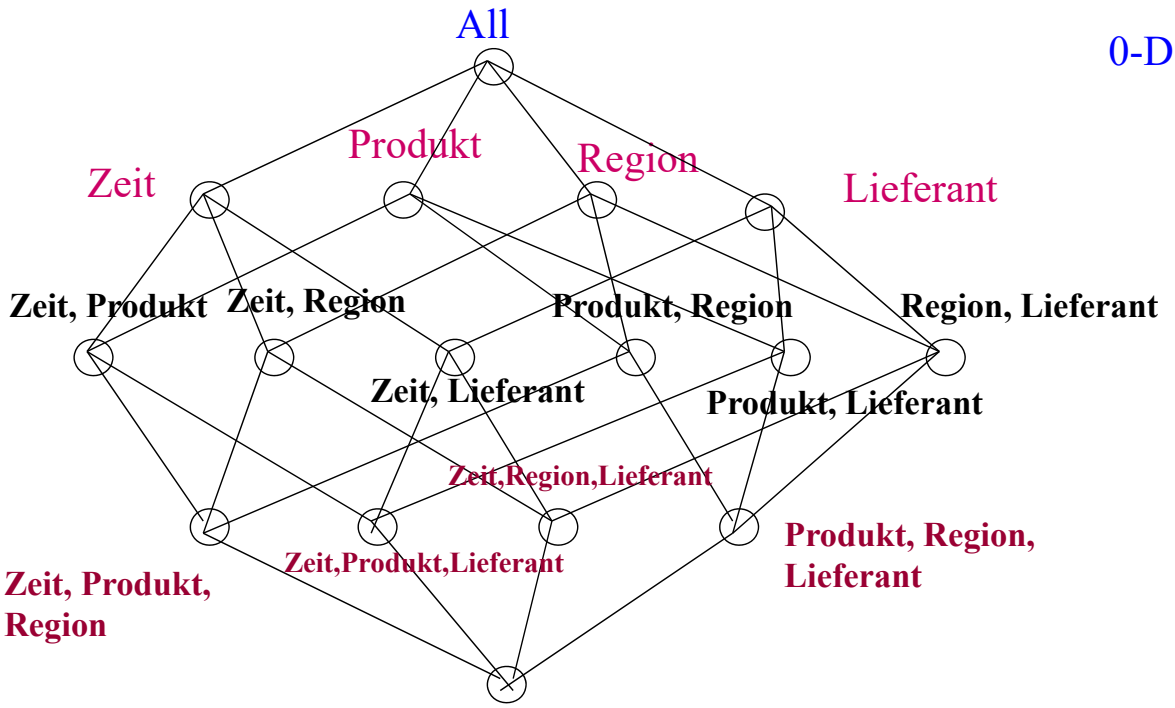


Cube mit Aggregationen

- 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



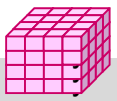
0-D(Scheitel) Cuboid

1-D Cuboide

2-D Cuboide

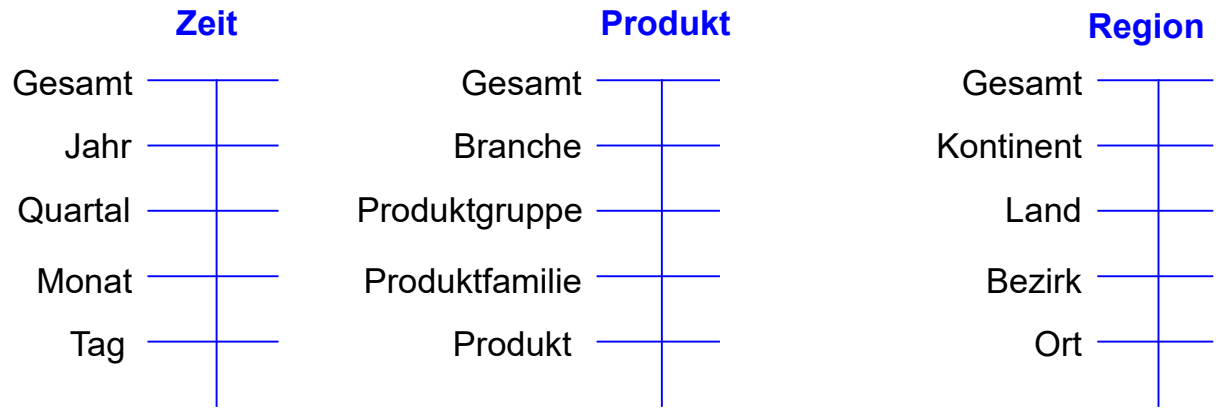
3-D Cuboide

4-D (Basis)-Cuboid

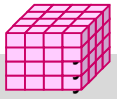


Dimensionshierarchien (Konzepthierarchien)

- 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

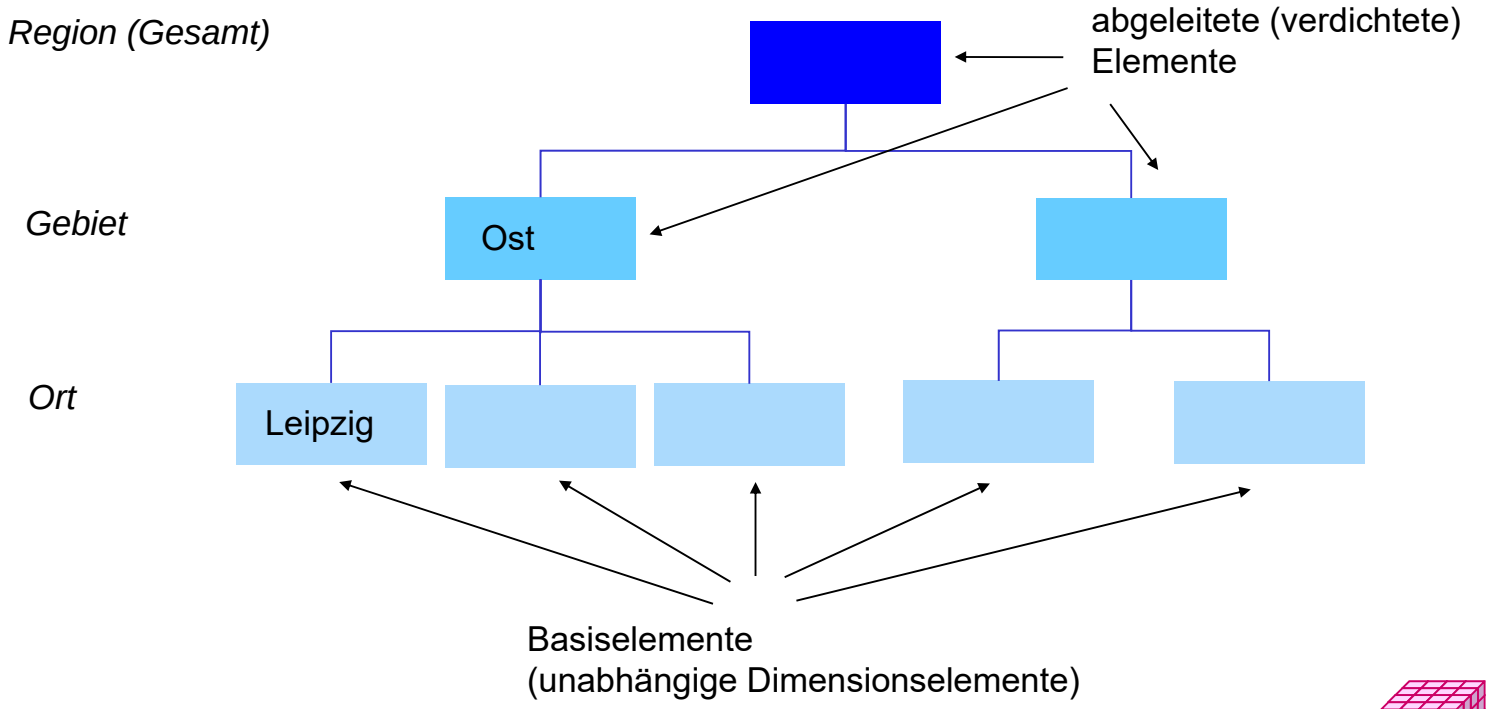


- Dimensionen können neben Klassifikations(Hierarchie)attributen noch beschreibende *dimensionale Attribute* aufweisen

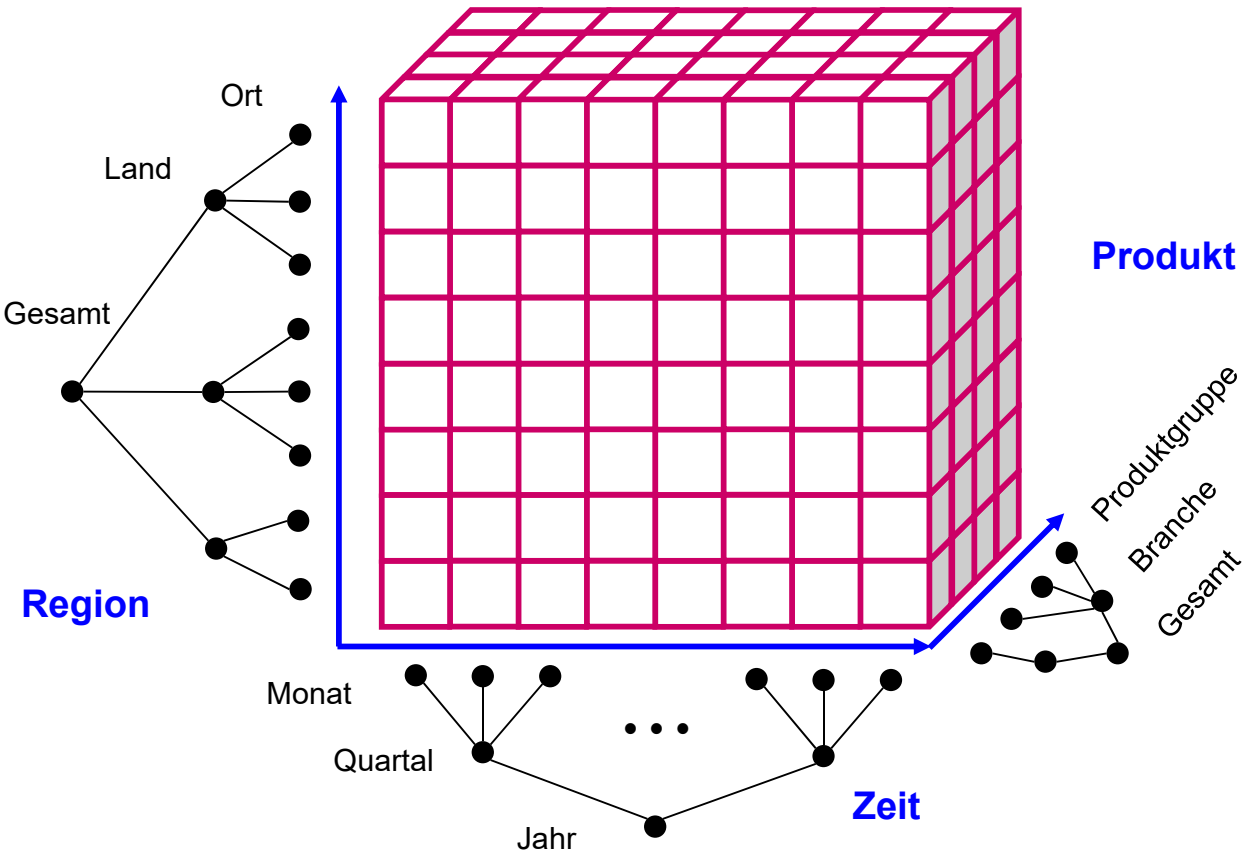


Beispiel einer Konzepthierarchie (Region)

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

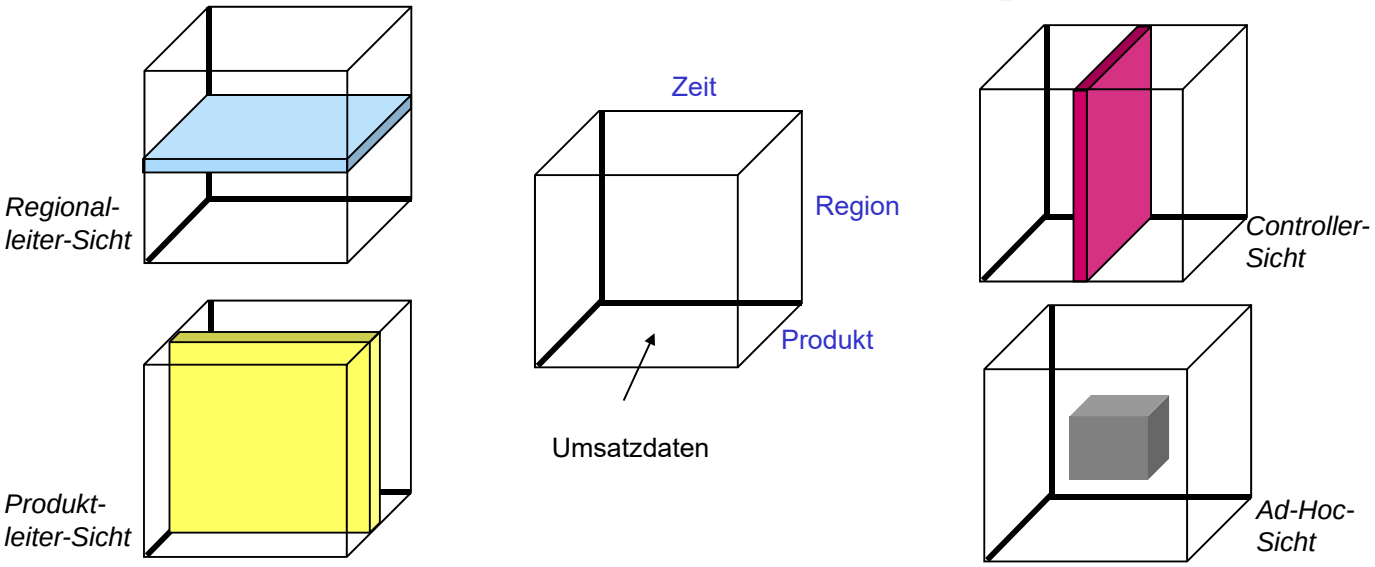


Cube mit hierarchischen Dimensionen

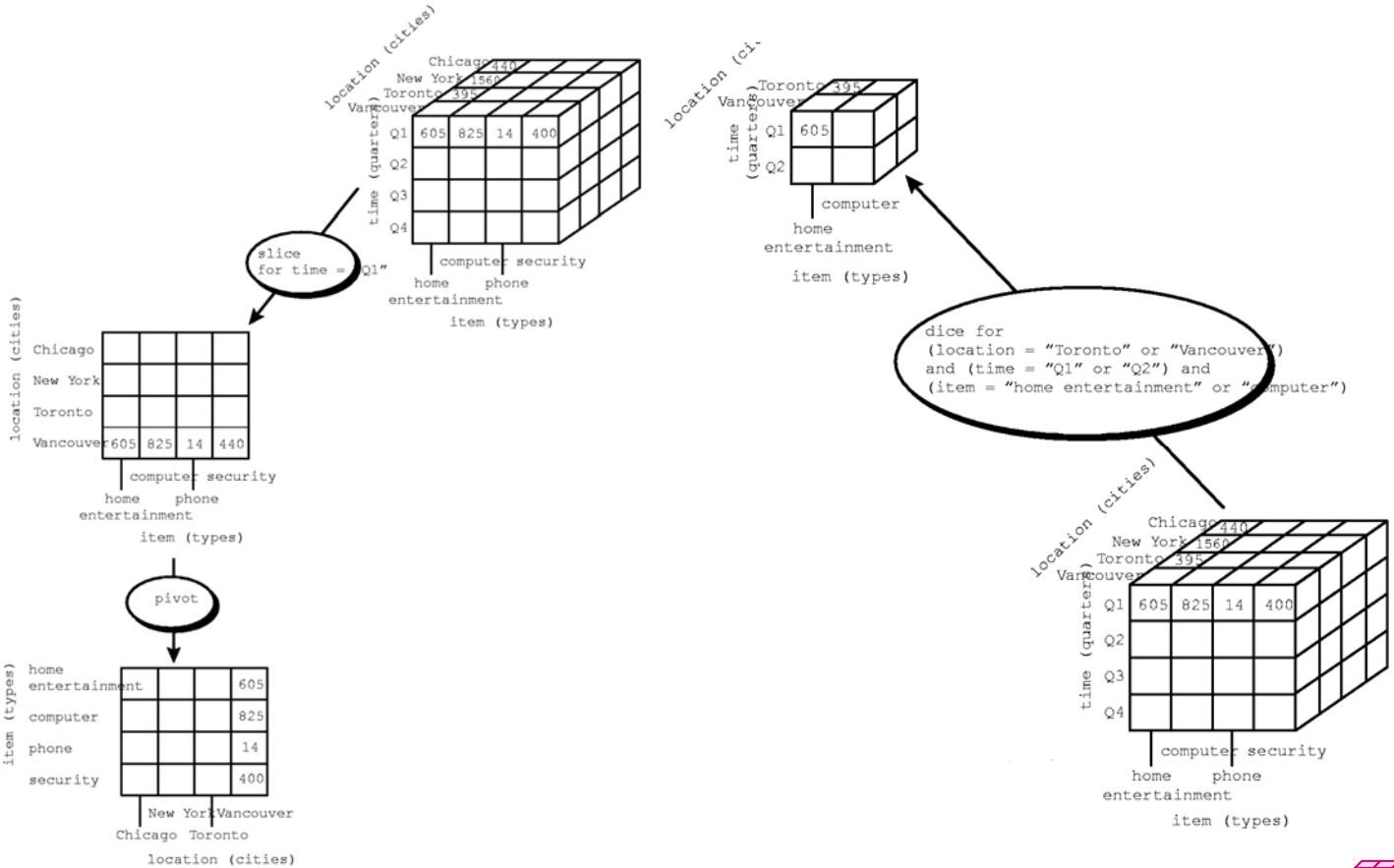


Operationen auf Cubes

- 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, …



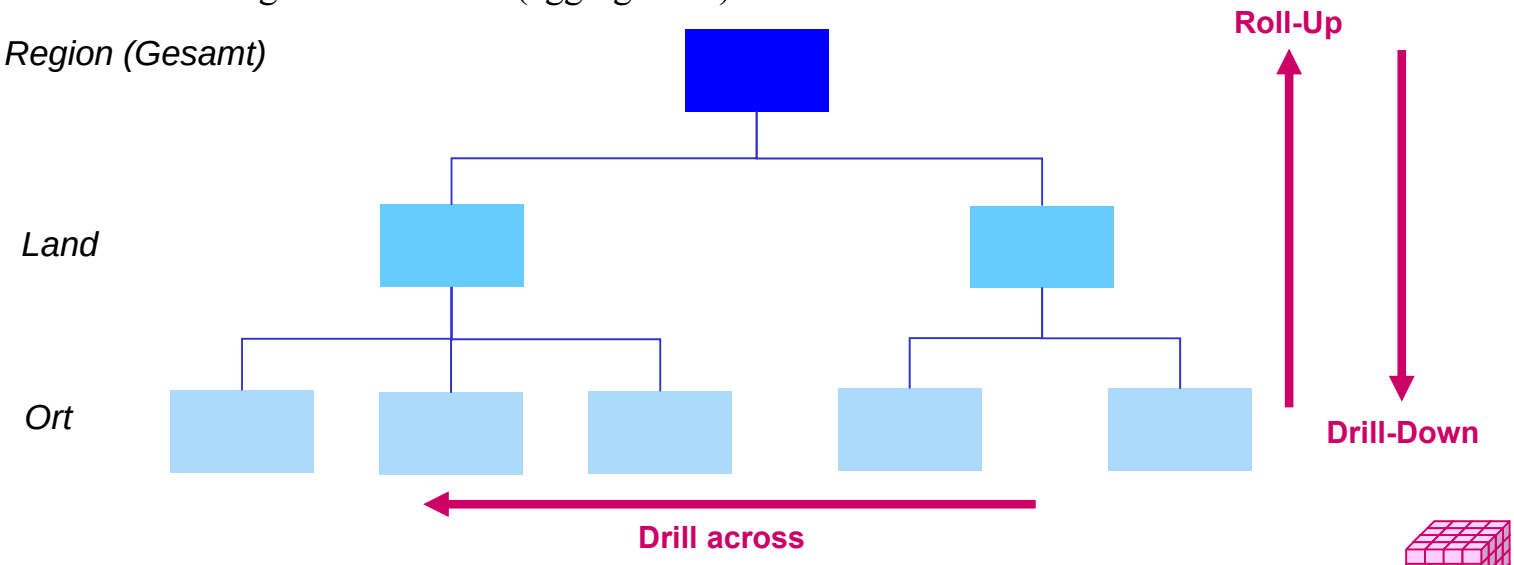
Beispiele



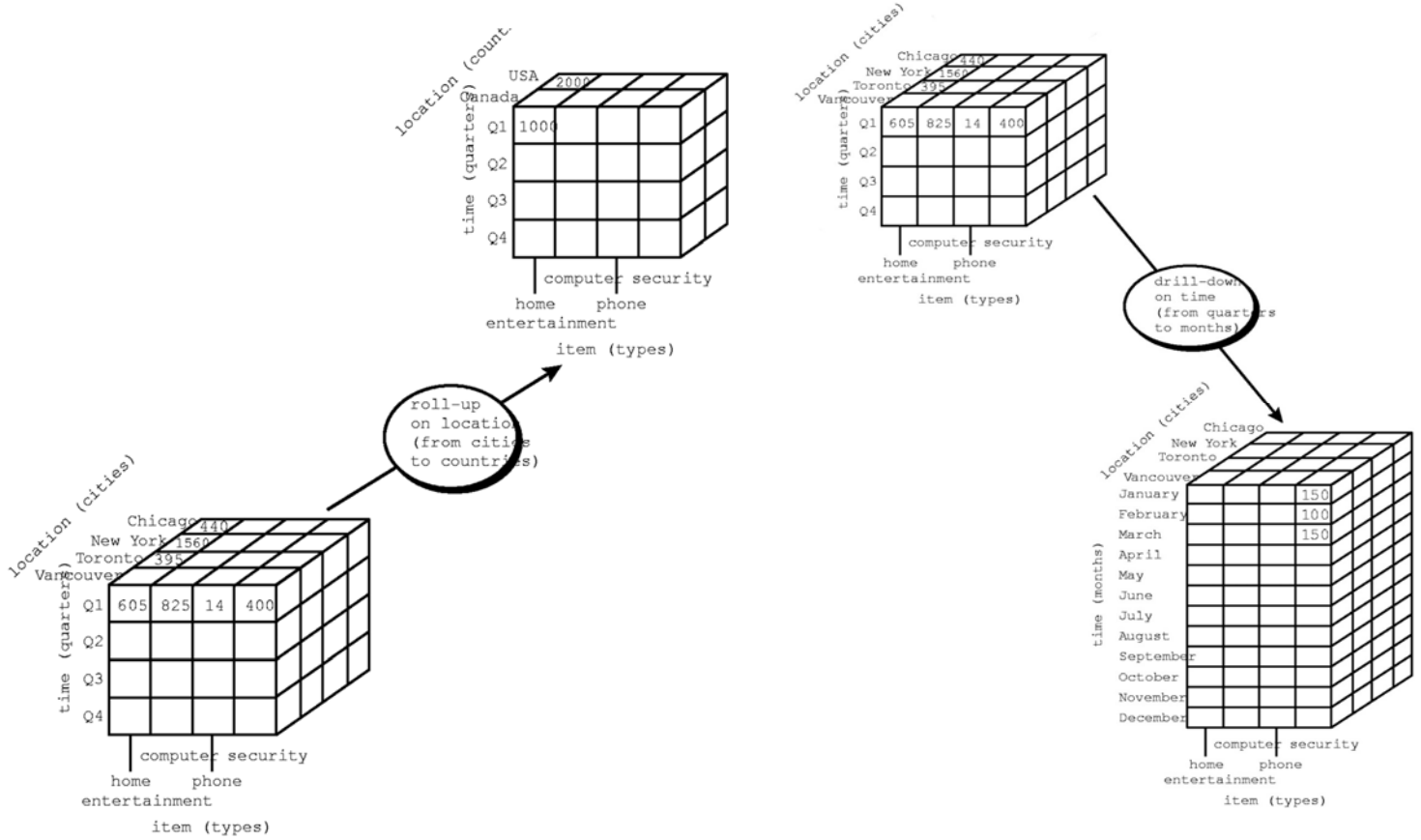
Quelle: J. Han, M. Kamber: Data Mining, Morgan Kaufmann

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



Beispiele Roll-Up, Drill-Down



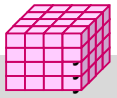
Quelle: J. Han, M. Kamber: Data Mining, Morgan Kaufmann

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			

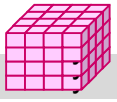


Aggregation: 2D-Beispiel

- 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

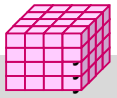


Größe der Cubes

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

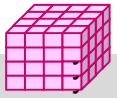
  L_i: #Ebenen von Dimension i (ohne Top-Level)

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)

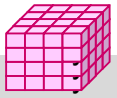
multi-dimensional (Kreuztabelle)

	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)

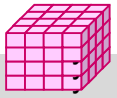
REGION

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



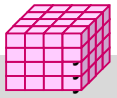
Sprachansatz MDX*

- 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

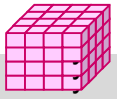
* http://msdn.microsoft.com/en-us/library/ms145506.aspx



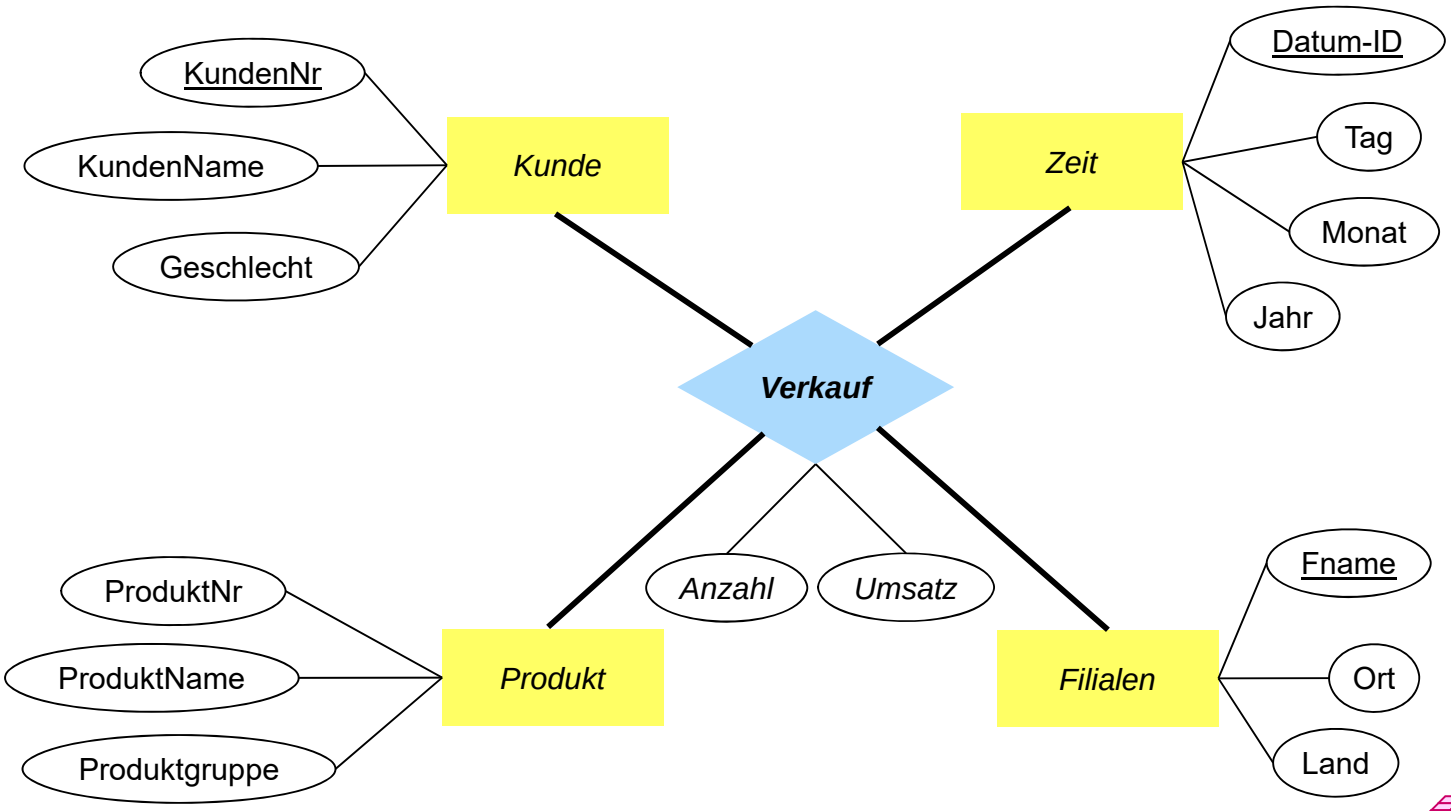
MDX: Beispiele

```
SELECT  Region.CHILDREN ON COLUMNS,
        Produkt.CHILDREN ON ROWS
FROM    Verkauf
WHERE   (Umsatz, Zeit.[2015])
```

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

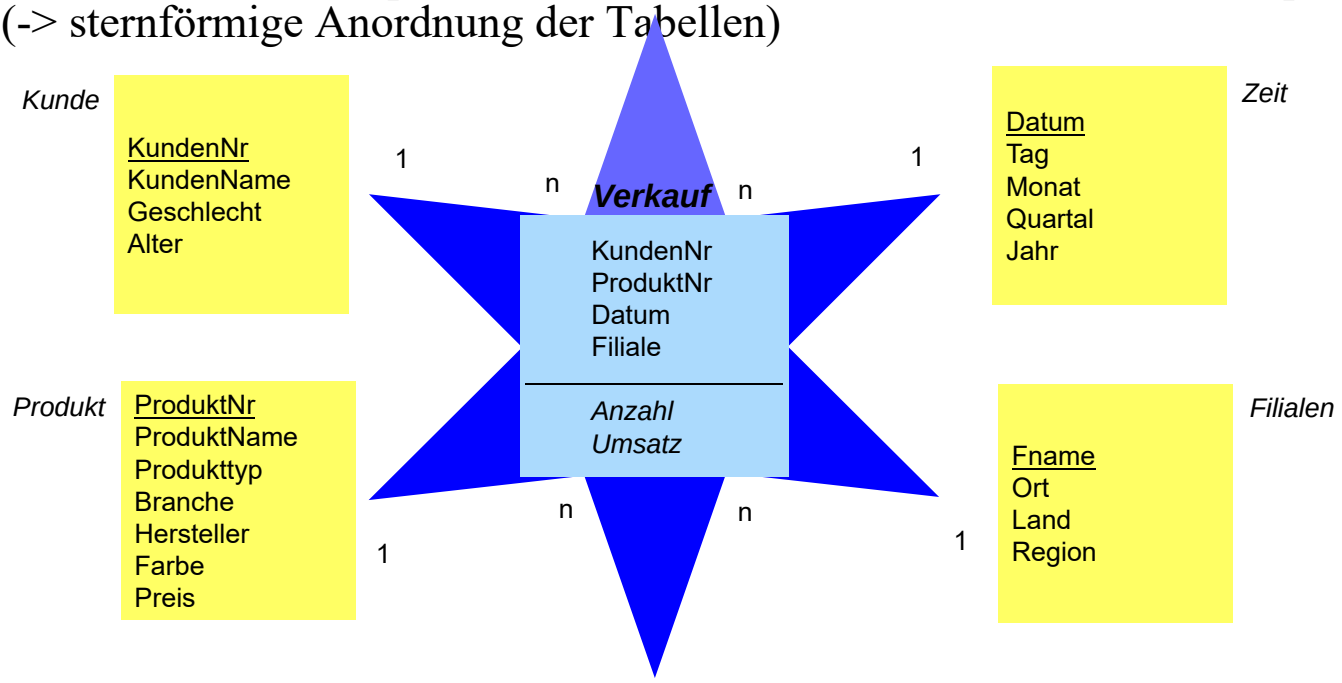


ER-Diagramm eines multi-dimensionalen Datenmodells



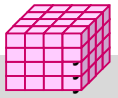
Relationale Speicherung: Star-Schema

- 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)



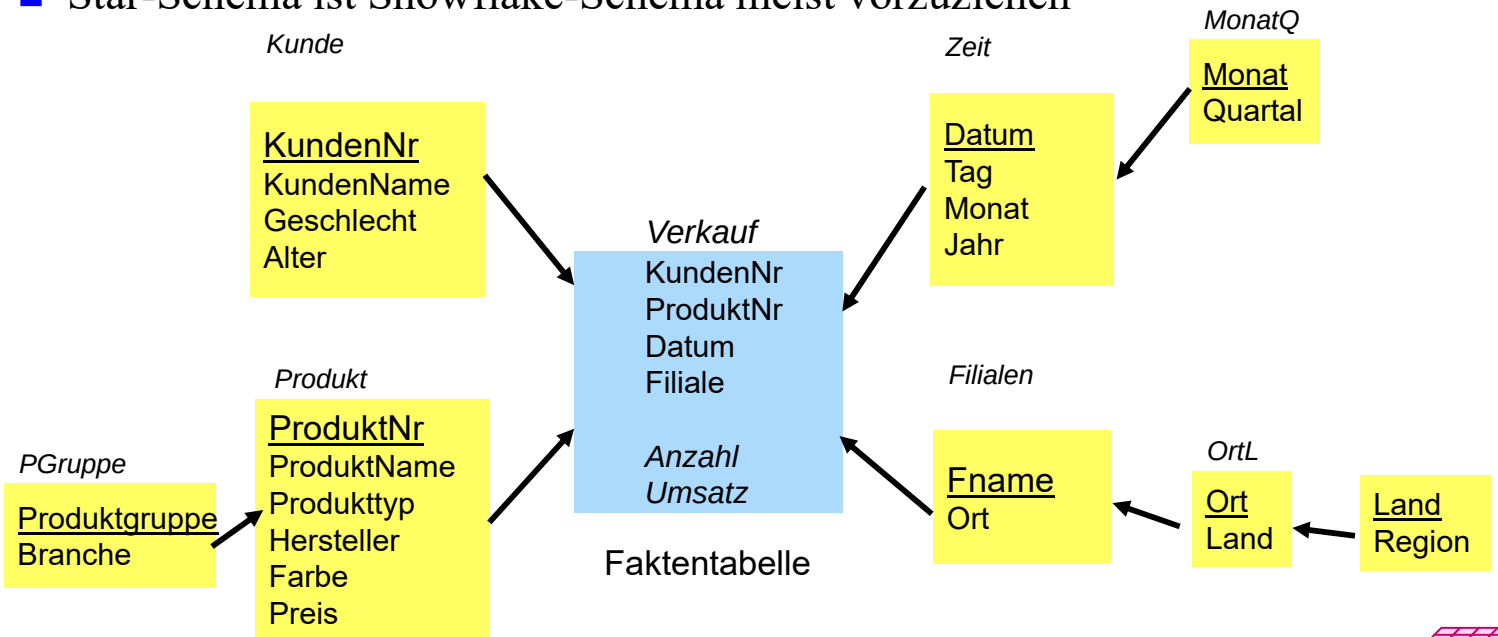
Star-Schema (2)

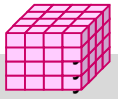
- formale Definition: Star-Schema besteht aus einer Menge von Tabellen D_1, ...D_n, F mit
  - Dimensionstabellen D_i bestehend aus (i.a. künstlichen) Primärschlüssel d_i und Dimensionsattributen
  - Faktentabelle F bestehend aus Fremdschlüsseln d_1, ... d_n sowie Meßgrößen (Kennzahlen) als weiteren Attributen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

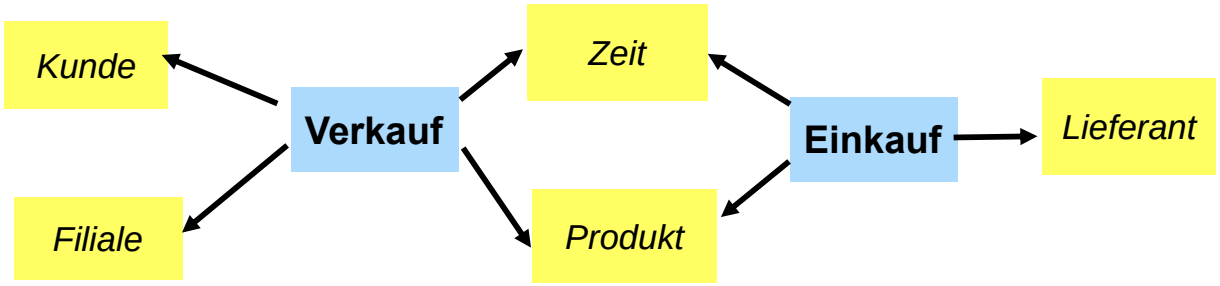


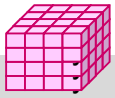


Galaxien-Schema

- 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





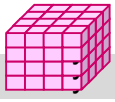
Übungsaufgabe : Warehouse-Entwurf

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.
- für Teilnehmer (Kunden) werden die üblichen Personenmerkmale für Analysezwecke erfasst, insbesondere Alter, Geschlecht und Beruf.

Dimensionen:

Faktentabelle:



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

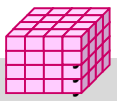
```
select      g1, ... gk, agg(f1), ... agg (fm)
from   D1, ..., Dn, F
where  <Selektionsbedingung auf D1> and
              ... and
       <Selektionsbedingung auf Dn> and
              D1.d1 = F.d1 and
               ... and
              Dn.dn = F.dn
group by    g1, ... gk
sort by     ...;
```

aggregierte Kennzahlen

Relationen des Star-Schemas

Join-Bedingungen

Ergebnis-Dimensionalität

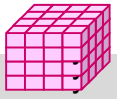


Beispiel eines Star-Join

- 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
2016	745
2017	710
2015	650



Mehrdimensionale Aggregationen mit Group-By

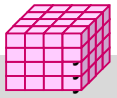
- Attributanzahl in **group by**-Klausel bestimmt Dimensionalität

```
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;

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';
```

Hersteller	Jahr	*Anzahl*
VW	2015	*2.000*
VW	2016	*3.000*
VW	2017	*3.500*
Opel	2015	*1.000*
...	...	...
BMW	2017	*1.500*
Ford	2015	*1.000*
Ford	2016	*1.500*
Ford	2017	*2.000*

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

Anzahl
19.500



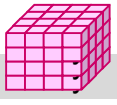
Relationale Speicherung aggregierter Werte

- Kreuztabelle (Crosstab-Darstellung)

Jahr / Hersteller	2015	2016	2017	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)

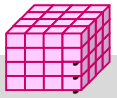
Hersteller	**Jahr**	***Anzahl***
VW	2015	*2.000*
VW	2016	*3.000*
VW	2017	*3.500*
Opel	2015	*1.000*
Opel	2016	*1.000*
Opel	2017	*1.500*
BMW	2015	*500*
BMW	2016	*1.000*
BMW	2017	*1.500*
Ford	2015	*1.000*
Ford	2016	*1.500*
Ford	2017	*2.000*
VW	**ALL**	*8.500*
Opel	**ALL**	*3.500*
BMW	**ALL**	*3.000*
Ford	**ALL**	*4.500*
ALL	2015	*4.500*
ALL	2016	*6.500*
ALL	2017	*8.500*
ALL	**ALL**	*19.500*



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, PostgresSQL (ab V9.5), …
- erspart mehrfache Berechnung der Aggregationen
  - entspricht Union von 2^n Aggregationen (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);
```

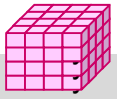


3D-Cube in relationaler Form

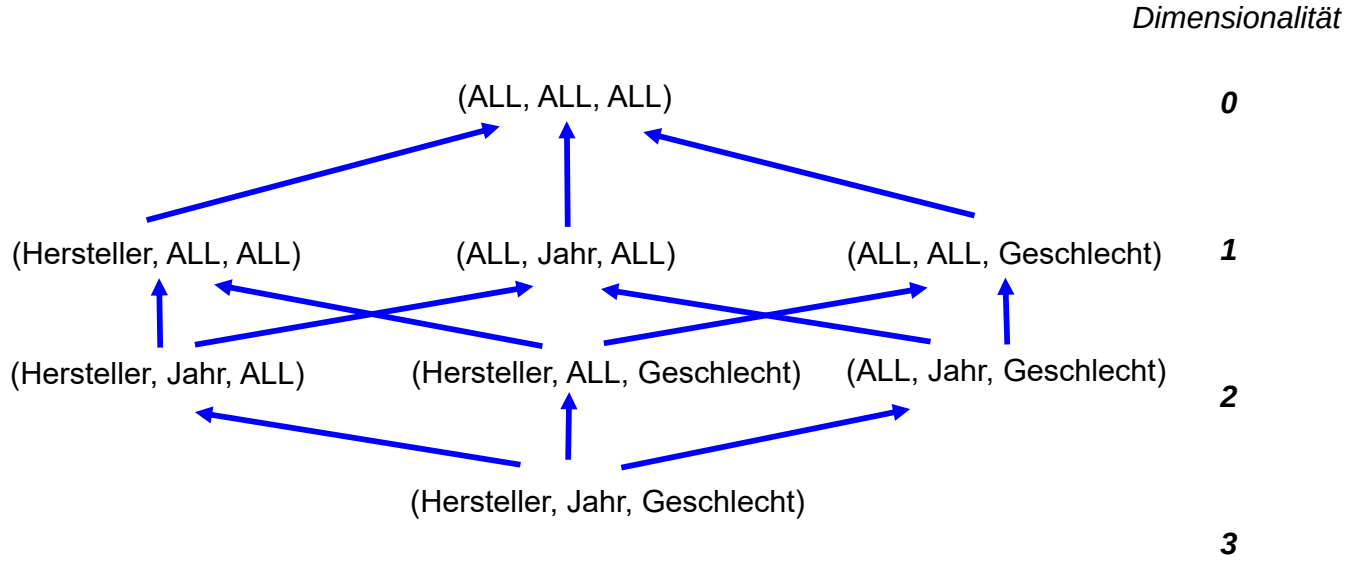
Hersteller	Jahr	Geschl	Anzahl
VW	2015	m	1300
VW	2015	w	700
VW	2016	m	1900
VW	2016	w	1100
VW	2017	m	2300
...	...	...	...
Opel	2015	m	800
Opel	2015	w	200
...	...	...	...
BMW	...	...	...
...	...	...	...

CUBE

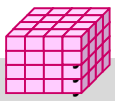
Hersteller	Jahr	Geschl	Anzahl
VW	2015	m	1300
VW	2015	w	700
...	...	...	...
VW	2015	ALL	2.000
...	...	ALL	...
Ford	2017	ALL	2.000
VW	ALL	m	5.400
...	...	...	...
Ford	ALL	w	...
ALL	2015	m	...
...	...		
VW	ALL	ALL	8.500
...	...		
ALL	2015	ALL	...
...			
ALL	ALL	m	...
...			
ALL	ALL	ALL	19.500



Cube-Aggregatgitter



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

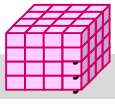


ROLLUP-Operator

- 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 ()

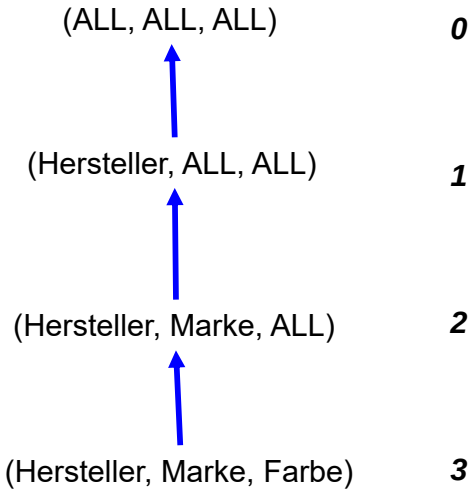
  liefert nur die Cuboide

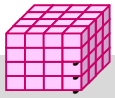
  a1, a2, ... , an-1, ALL, f (),
  ...
  a1, ALL, ... , ALL, f (),
  ALL, ALL, ... , ALL, f ()
- Reihenfolge der Attribute relevant!



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);
```

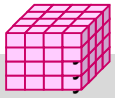




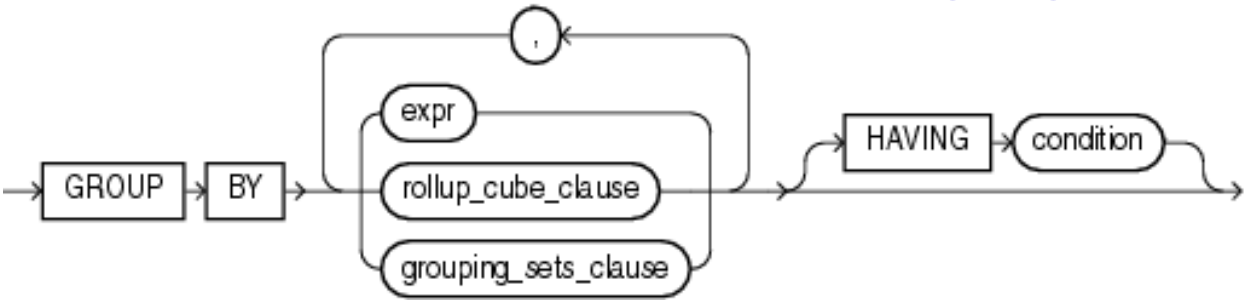
ROLLUP-Beispiel

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
...	...		

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

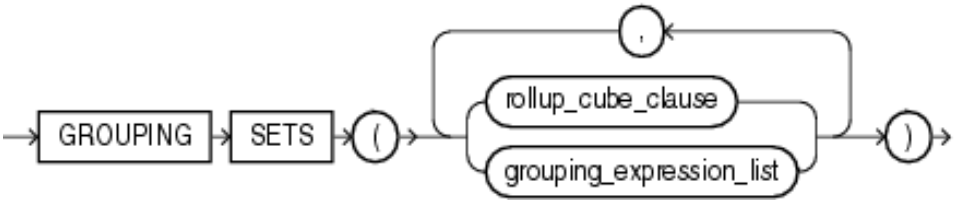


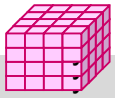
SQL: erweiterte Group-By Syntax



rollup_cube_clause:

grouping_sets_clause:





Grouping Sets

- mehrere Gruppierungen pro Anfrage

  GROUP BY GROUPING SETS (<Gruppenspezifikationsliste>)
  Gruppenspezifikation: (<Gruppenspezifikationsliste>) |
  CUBE <Gruppenspezifikationsliste> |
  ROLLUP <Gruppenspezifikationsliste>

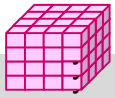
  leere Spezifikationsliste ( ) möglich: Aggregation über gesamte Tabelle

- 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*

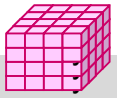
- CUBE, ROLLUP, herkömmliches Group-By entsprechen speziellen Grouping-Sets



Übungsaufgabe

- Bestimmen Sie für die gezeigte Tabelle *Goals* das Ergebnis folgender SQL-Anfragen:
  - 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),());

Spieler	Saison	*Anzahl*
Gomez	2012	*26*
Gomez	2013	*11*
Gomez	2017	*16*
Müller	2014	*13*
Müller	2015	*13*
Müller	2016	*20*
Aubameyang	2015	*16*
Aubameyang	2016	*25*
Aubameyang	2017	*31*
Lewandowski	2015	*17*
Lewandowski	2016	*30*
Lewandowski	2017	*30*
Werner	2017	*21*

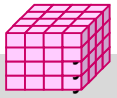


Statistische Funktionen in SQL

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

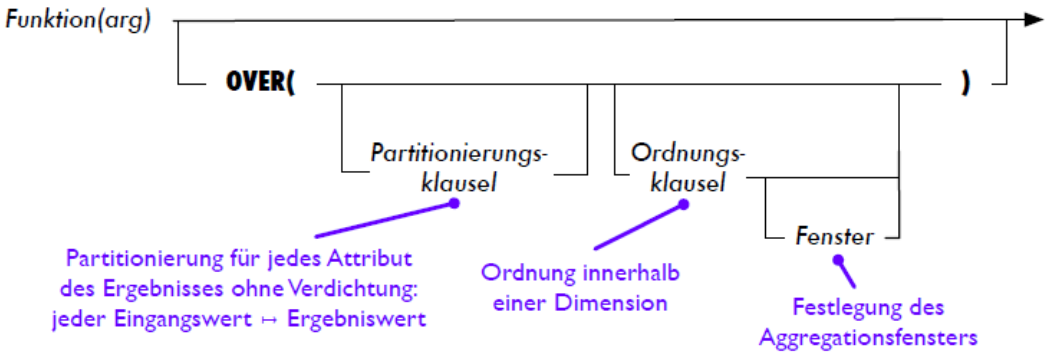
```
select p.Produkttyp, CORR (v.Verkaufspreis, p.Einkaufspreis)
from Verkauf v, Produkt p
where v.ProduktNr = p. ProduktNr
group by p.Produkttyp
```

- 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)

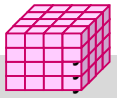


Rank/Windowing-Funktionen

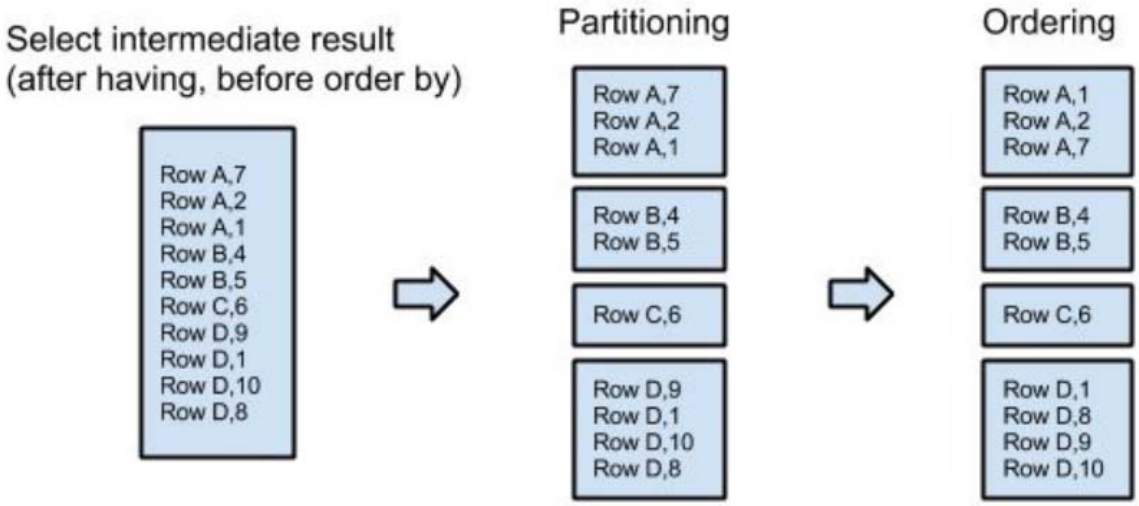
- standardisiert seit SQL:1999 / 2003
- erweiterte Analysemöglichkeiten
  - Berechnung von Rangfolgen / Top-N
  - kumulierte Häufigkeiten/Anteile (z.B. bezüglich eines Jahres/Monats)
  - Vergleiche, z.B. Monatsumsatz gegenüber gleitendem 3-Monatsdurchschnitt …
- **OVER**-Prädikat in Select-Klausel oder **WINDOW**-Klausel auf Datensequenzen



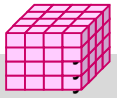
  - Funktionen: SUM, AVG, RANK, DENSE_RANK …
  - PARTITION BY: Zerlegung in mehrere Datensequenzen/ströme (optional)
  - ORDER BY: Sortierreigenfolge pro Datensequenz (optional)
    - Fensterangabe bei ORDER BY: tupelweise (ROWS) oder wertebereichsweise (RANGE) Einschränkung für Aggregationsfenster (optional)



Window-Verarbeitung



https://blog.matters.tech/sql-window-functions-basics-e9a9fa17ce7e



Rank-Funktion

Tabelle AVerkauf (Hersteller, Jahr, Anzahl)

```
select Hersteller, Anzahl,
  rank() over (order by Anzahl desc)
                    as Rang,
  dense_rank() over (order by Anzahl desc)
                    as DRang
from AVerkauf
where  Jahr=2015
order by Anzahl desc, Hersteller
```

alternative Formulierung mit WINDOW-Klausel

```
select Hersteller, Anzahl,
   rank() over w as Rang,
   dense_rank() over w as DRang
from AVerkauf
where  Jahr=2015
order by Anzahl desc, Hersteller
window w as (order by Anzahl desc)
```

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-Beispiel (LOTS-Datenbank)

SQL Anfrage

```
select titel, jahr, row_number() over w as lfdnr,
       rank() over w as Rang, dense_rank() over w as DRang
from buch
window w as (order by jahr)
```

zeige Datensätze 1 - 20 (4.877 insgesamt)

Zeige: 20 Datensätze, beginnend ab 21

titel	jahr	lfdnr	rang	drang
Elektromechanische Schaltungen und Schaltgeräte : eine Einführung in Theorie und Berechnung	1956	1	1	1
Mathematische Gesetze der Logik : Bd. I, Vorlesungen über Aussagenlogik	1960	2	2	2
Elektronische Zählschaltungen : eine Einführung in ihre Wirkungsweise und Technik	1961	3	3	3
Digitale Rechenanlagen : Grundlagen, Schaltungstechnik, Arbeitsweise, Betriebssicherheit	1961	4	3	3
Transistor logic circuits	1961	5	3	3
Matrizen und ihre technischen Anwendungen	1961	6	3	3
Grundlagen der Struktursynthese von Relaisschaltungen : mit 25 Tabellen	1962	7	7	4
Formale Logik	1962	8	7	4
A survey of mathematical logic	1962	9	7	4
Foundations of mathematical logic	1963	10	10	5
Einführung in die Anwendung moderner Rechenautomaten	1963	11	10	5
Digital computer design : logic, circuitry, and synthesis	1963	12	10	5
Foundations of mathematical logic	1963	13	10	5
Einführung in die mathematische Logik	1963	14	10	5
Boolesche Algebra und ihre Anwendungen	1964	15	15	6

Rank-Funktion (2)

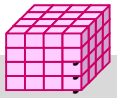
- partitionsweises Ranking

```
select Hersteller, Jahr, Anzahl,  rank() over
        (partition by Jahr order by Anzahl desc) as Rang,
from AVerkauf
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
```



Partitionsweises Ranking: Beispiel

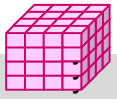
SQL Anfrage

```
select titel, preis, verlagsid, rank() over (partition by verlagsid order by preis desc) as VRang
from buch  where preis is not null
order by 2 desc
```

zeige Datensätze 1 - 20 (3.210 insgesamt)

Zeige: 20 Datensätze, beginnend ab 21 >

titel	preis	verlagsid	vrang
Computer - Neue Flügel des Geistes ? : die Evolution computergestützter Technik, Wissenschaft, Kultur und Philosophie	448.00	34	1
Microsoft Windows NT : Version 3.5 ; Expertenwissen zu Windows NT Workstation und Windows NT Server	399.00	44	1
Elektronisches Wörterbuch der Handels-/Finanz-/Rechtssprache : Deutsch, Englisch, Französisch ; Office Dic Bedienungshandbuch	398.00	428	1
Effizienter DB-Einsatz von ADABAS	378.10	2	1
The handbook of human factors	350.00	154	1
Digital image processing	322.00	154	2
Encyclopedia of software engineering	300.70	154	3
Grundlagen : [technische Informationen und Tools für den Support-Spezialisten]	299.00		1
Die relationale Datenbanksprache SQL : elektronische Weiterbildung unter WINDOWS [der Fernuniversität Hagen]	298.00	1	1
Handbuch der Universitäten und Fachhochschulen : Bundesrepublik Deutschland, Österreich, Schweiz	298.00	131	1
Concurrent engineering : tools and technologies for mechanical system design	298.00	6	1
From discourse to logic : introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory	291.20	173	1
Multimedia information storage and management	290.50	173	2
Visuomotor coordination : amphibians, comparisons, models and robots	285.00	257	1



Beispiel 2 (2 teuerste Bücher pro Verlag)

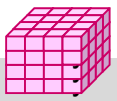
SQL Anfrage

```
select titel, name, preis
from (select titel, name, preis, rank() over (partition by verlagsid order by preis desc) rang from buch natural join
verlag where preis is not null) tmp
where rang <=2
```

zeige Datensätze 1 - 20 (488 insgesamt)

Zeige: 20 Datensätze, beginnend ab 21

titel	name	preis
Die relationale Datenbanksprache SQL : elektronische Weiterbildung unter WINDOWS [der Fernuniversität Hagen]	Addison Wesley	298.00
Multiparadigm programming in Leda	Addison Wesley	195.00
Effizienter DB-Einsatz von ADABAS	Vieweg	378.10
QM-Handbuch der Softwareentwicklung : Muster und Leitfaden nach DIN ISO 9001	Vieweg	248.00
Mikroprozessortechnik : mit Übungen und Testfragen	Vogel	69.00
C und Assembler in der Systemprogrammierung	Vogel	69.00
Einführung in die Datensicherheit : Probleme und Lösungen	Vogel	69.00
PC-Labor : technisch-physikalische Experimente mit Sensoren und PC-Software für Messwerterfassung, Analyse und Grafik	Markt & Technik	129.00
Windows 95 - das Kompendium	Markt & Technik	99.95
Softwareprüfung und Qualitätssicherung : das Handbuch zur Prüfung von Softwareerzeugnissen nach DIN ISO/IEC 12119	Oldenbourg	198.00
Lexikon Informatik und Datenverarbeitung	Oldenbourg	148.00
Concurrent engineering : tools and technologies for mechanical system design	Springer	298.00
Robots and biological systems: towards a new bionics?	Springer	268.00
Encyclopedia of graphics file formats	O'Reilly	119.00
PEXlib programming manual	O'Reilly	83.00



Aggregatberechnung auf Windows

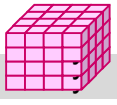
- Nutzung von SUM, AVG etc. in Verbindung mit OVER
- Anwendungsbeispiel für Tabelle *sales (date, value)*

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
```

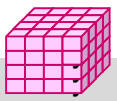


Beispieldaten

sales

date	value
1.2.2015	500
1.2.2015	300
5.7.2015	200
2.3.2016	400
3.4.2016	100
9.6.2017	500

date	day_sum	all_sum	anteil
1.2.2015			
5.7.2015			
2.3.2016			
3.4.2016			
9.6.2017			



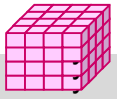
Bsp.: Kumulierte Summe

- 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
```



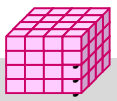
Lots-Beispiel

```
SELECT titel, jahr, preis, sum(preis) over (partition by jahr order by jahr) as jahresbetrag, sum(preis) over (order by jahr,buchid) as kumulierteSumme
FROM buch where preis is not null
```

zeige Datensätze 1 - 20 (3.210 insgesamt)

Zeige: 20 Datensätze, beginnend ab 21

titel	jahr	preis	jahresbetrag	kumuliertesumme
Elektromechanische Schaltungen und Schaltgeräte : eine Einführung in Theorie und Berechnung	1956	24.00	24.00	24.00
Informationstheorie : ein Einführung	1974	19.90	19.90	43.90
Prädikatenkalkül der ersten Stufe	1975	19.80	19.80	63.70
Transductions and context-free languages	1979	44.00	44.00	107.70
Derivative Wortbildung der deutschen Gegenwartssprache und ihre algorithmische Analyse	1980	64.00	64.00	171.70
Duden - Wann schreibt man groß, wann schreibt man klein : Regeln und ausführliches Wörterverzeichnis	1981	14.80	112.80	186.50
Natürlichsprachliche Argumentation in Dialogsystemen : KI-Verfahren zur Rekonstruktion und Erklärung approximativer Inferenzprozesse	1981	28.50	112.80	215.00
The science of programming	1981	50.00	112.80	265.00
Prädikatenlogik höherer Stufe	1981	19.50	112.80	284.50
Duden - Komma, Punkt und alle anderen Satzzeichen : mit umfangreicher Beispielsammlung	1982	12.80	32.60	297.30
Aussagenkalkül	1982	19.80	32.60	317.10
Algebraische Spezifikation : eine Einführung	1983	49.00	484.10	366.10
Notes on introductory combinatorics	1983	72.00	484.10	438.10
Very large data bases : 9th International Conference on Very Large Data Bases, Florence, Italy, October 31 - November 2, 1983 ; proceedings	1983	111.00	484.10	549.10
Duden - Wie sagt man anderswo? : landschaftliche Unterschiede im deutschen Sprachgebrauch	1983	12.80	484.10	561.90
Kryptographie : eine Einführung in die Methoden und Verfahren der geheimen Nachrichtenübermittlung	1983	24.80	484.10	586.70
Degrees of unsolvability : local and global theory	1983	168.00	484.10	754.70
Data structures and network algorithms	1983	26.70	484.10	781.40
Semantik II	1983	19.80	484.10	801.20
Natural language communication with pictorial information systems	1984	98.00	243.00	899.20



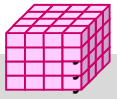
Windowing mit expliziter Fensterangabe

- Beispiel Moving Average für Tabelle *sales (date, value)*

  Berechne pro Datum durchschnittlichen 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)



Beispiel Moving Average

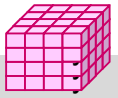
SQL Anfrage

```
select jahr, to_char(Dpreis, '999.99') Dpreis,
       to_char((avg(DPreis) over w),'999.99') as MovingAvg
from (select jahr, avg(preis) from buch  where preis is not null group by jahr) as DP(jahr,Dpreis)
window w as  (order by jahr rows between 1 preceding and 1 following)
order by 1
```

zeige Datensätze 1 - 20 (27 insgesamt)

Zeige: 20 Datensätze, beginnend ab 21

jahr	dpreis	movingavg
1956	24.00	21.95
1974	19.90	21.23
1975	19.80	27.90
1979	44.00	42.60
1980	64.00	45.40
1981	28.20	36.17
1982	16.30	35.00
1983	60.51	45.85
1984	60.75	64.87
1985	73.36	71.36
1986	79.96	76.06
1987	74.86	76.70
1988	75.29	74.18
1989	72.38	73.17
1990	71.83	73.78
1991	77.12	75.18
1992	76.58	75.99
1993	74.28	75.23
1994	74.84	74.68
1995	74.91	74.82



Explizites Windowing (2)

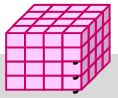
- partitionsweises Windowing

  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
```



Zusammenfassung

- Einfachheit des mehrdimensionalen 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 …