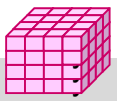
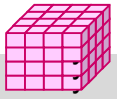
3. Mehrdimensionale Datenmodellierung und Operationen

- 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
  - Star Join
- mehrdimensionale Gruppierungen in SQL
  - Group-By-Erweiterungen**:** Cube, Rollup, Grouping Sets
- Auswertung von Zeitreihen/Satzfolgen mit SQL
  - RANK-, WINDOW-Queries

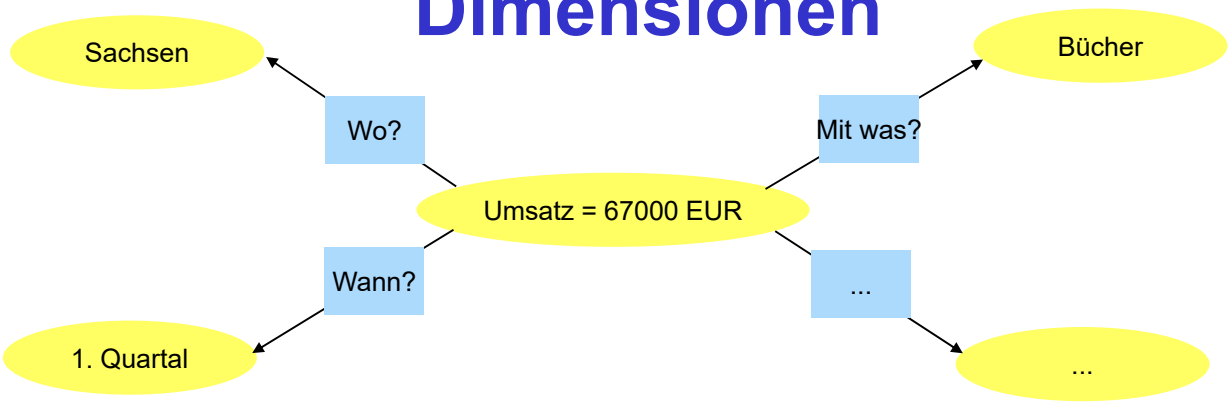


Kennzahlen

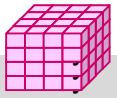
- Kennzahl ist numerische Größe mit konzentrierter Aussagekraft zur Diagnose, Überwachung und Steuerung eines Systems
  - auch: Fakten, Meßgrößen, Measures, Key Performance Indicators (KPI)
  - meist betriebswirtschaftliche Größen, z.B. Umsatz / Gewinn / Rentabilität
  - KPIs oft aus einfacheren Kennzahlen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



Dimensionen

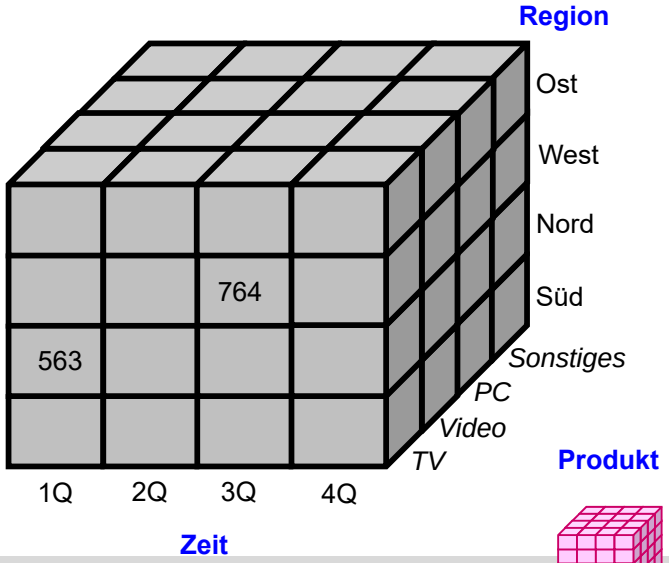


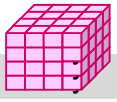
- Zahlenwert einer Kennzahl ist ohne semantischen Bezug nichtssagend
  - Dimensionen setzen Kennzahlen in Bezug zu Eigenschaften / sachlichen Kriterien
- *Dimension*: Datentyp, i.a. endlich (z.B. Aufzählung)
  - Beispiele: Menge aller Produkte, Regionen, Kunden, Zeitperioden etc.
  - *Dimensionselement*: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



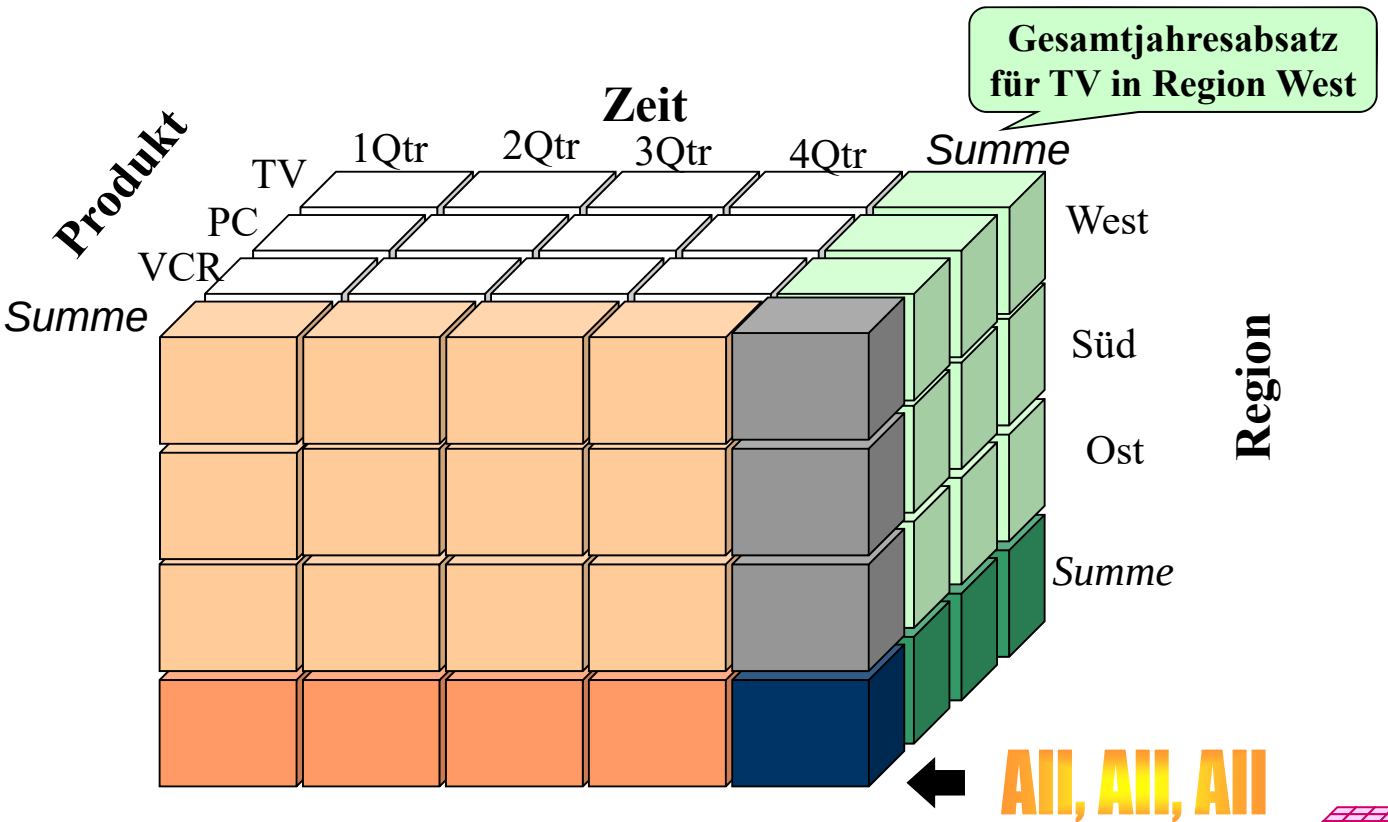
Data Cube

- 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, ...



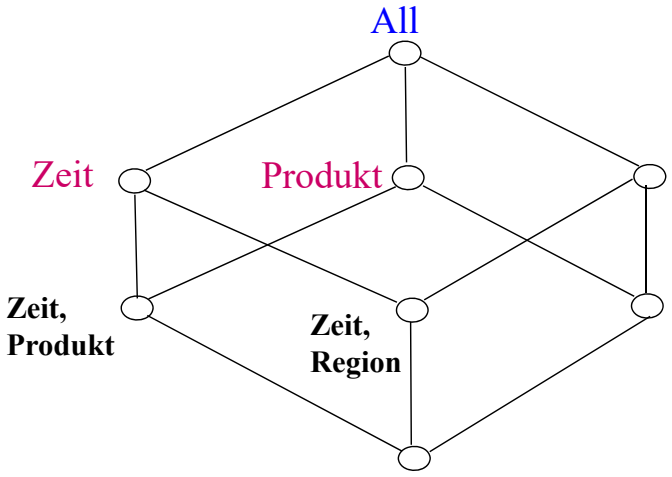


Data Cube: 3D-Beispiel mit Aggregation



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



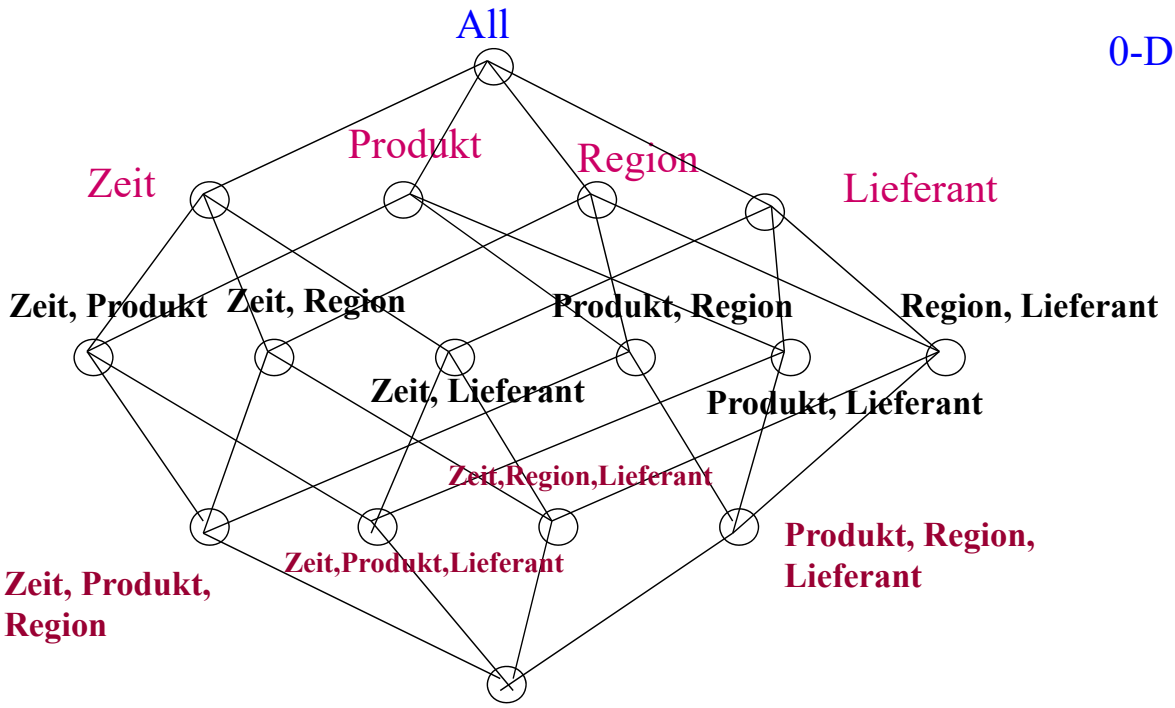
0-D (Scheitel) Cuboid

1-D Cuboide

2-D Cuboide

3-D (Basis)-Cuboid

Cuboid-Verband für 4D Cube



0-D(Scheitel) Cuboid

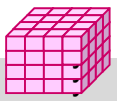
1-D Cuboide

2-D Cuboide

3-D Cuboide

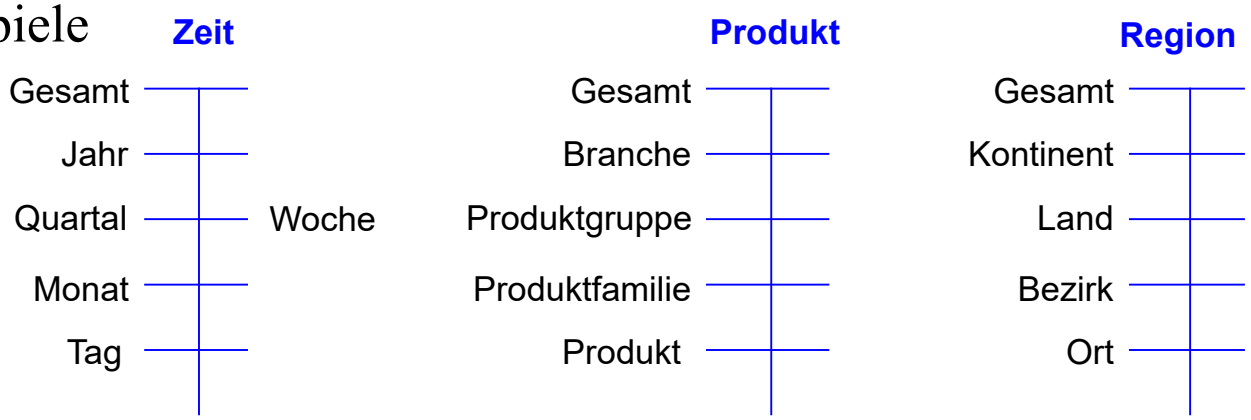
Zeit, Produkt, Region, Lieferant

4-D (Basis)-Cuboid

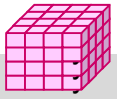


Dimensionshierarchien (Konzepthierarchien)

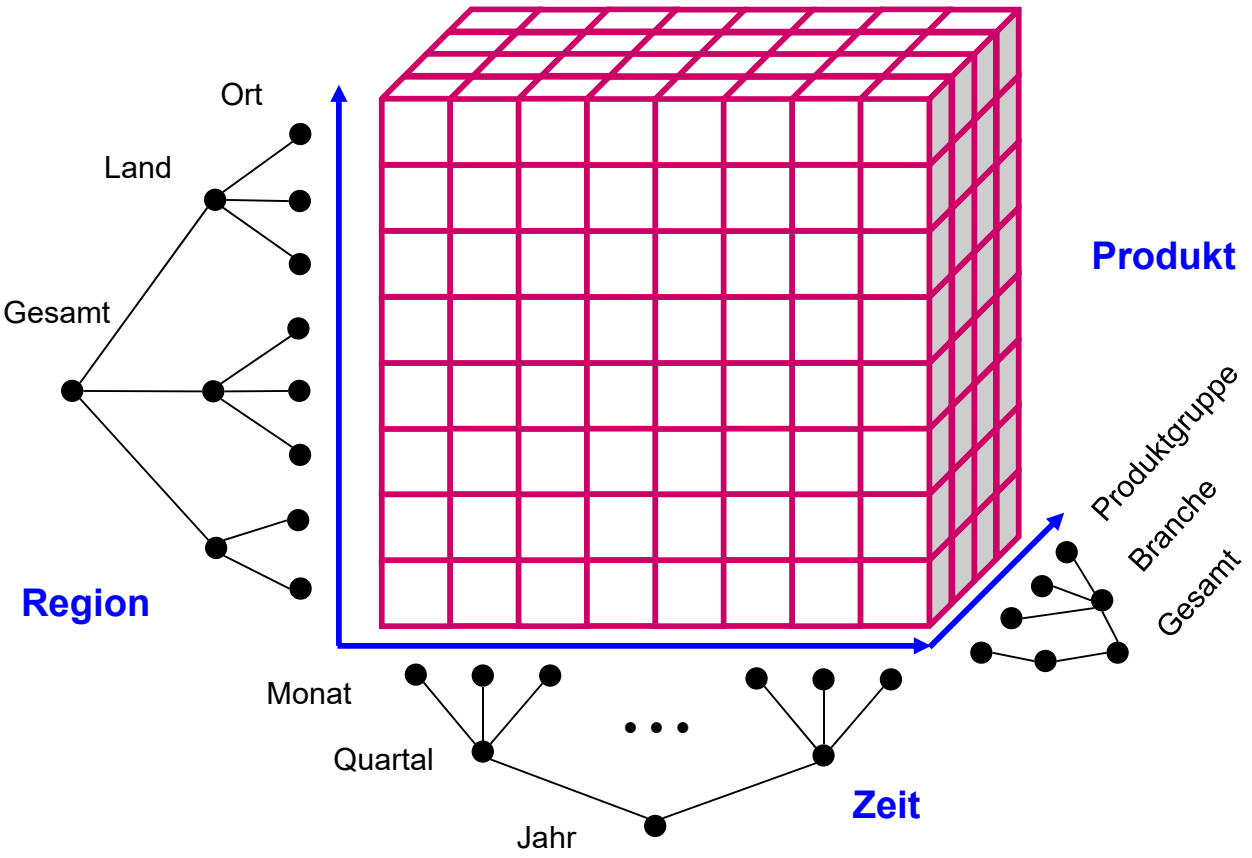
- häufig hierarchische Beziehungen zwischen Dimensionsobjekten
  - Top-Level pro Hierarchie für alle Dimensionselemente (Gesamt, Top, All)
  - *Primärattribut*: unterste (genaueste) Stufe
  - funktionale Abhängigkeiten zwischen Primärattribut und *Klassifikationsattributen* höherer Stufen
  - *einfache Hierarchie* (pro Element höchstens ein übergeordnetes Element) vs. *parallele Hierarchie* bzw. Halbordnung
- Beispiele



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

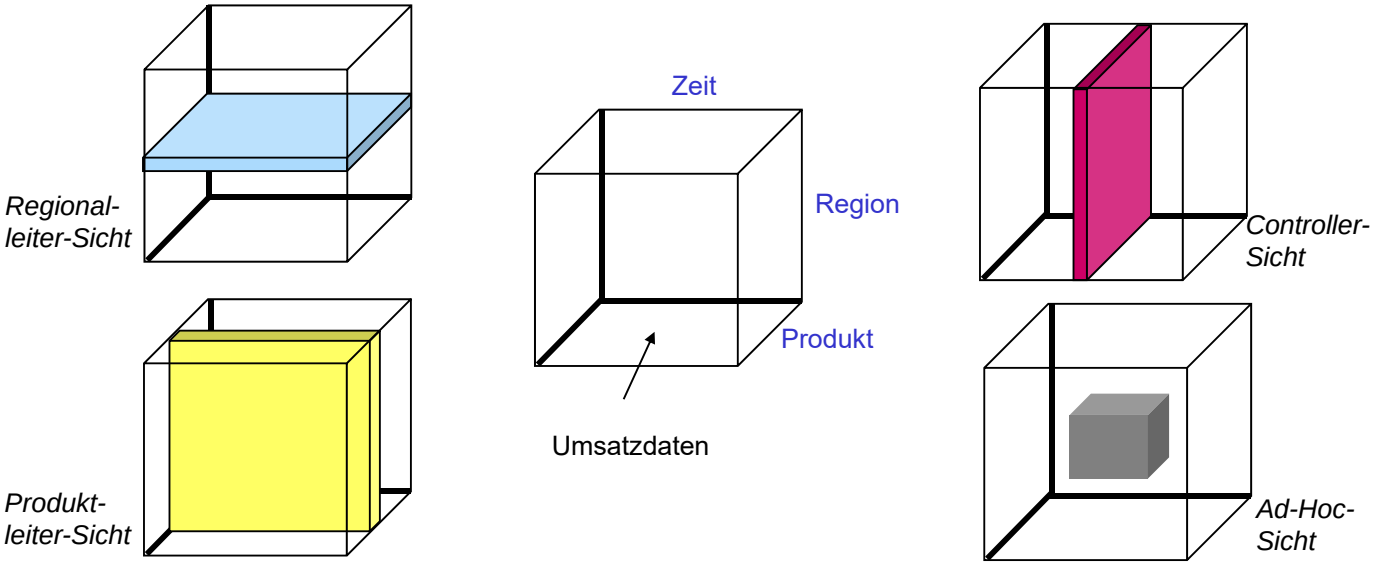


Cube mit hierarchischen Dimensionen

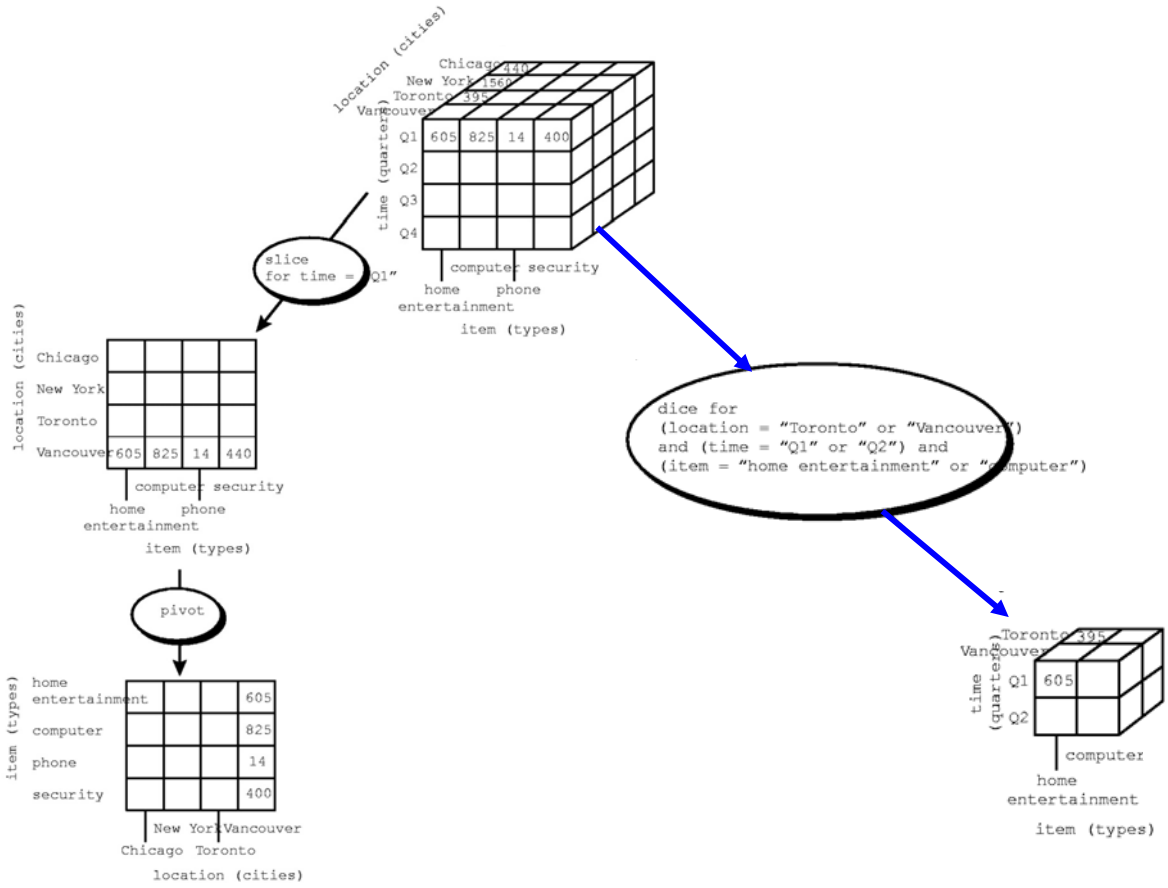


Operationen auf Cubes

- 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, ...

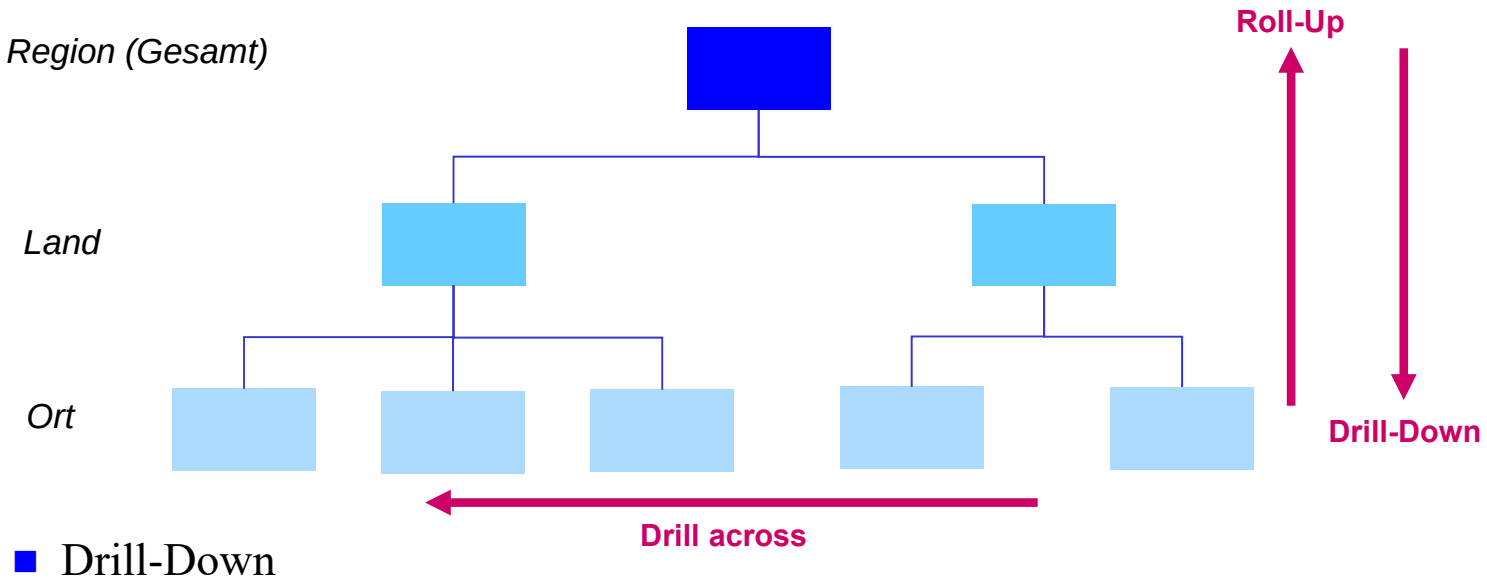


Beispiele



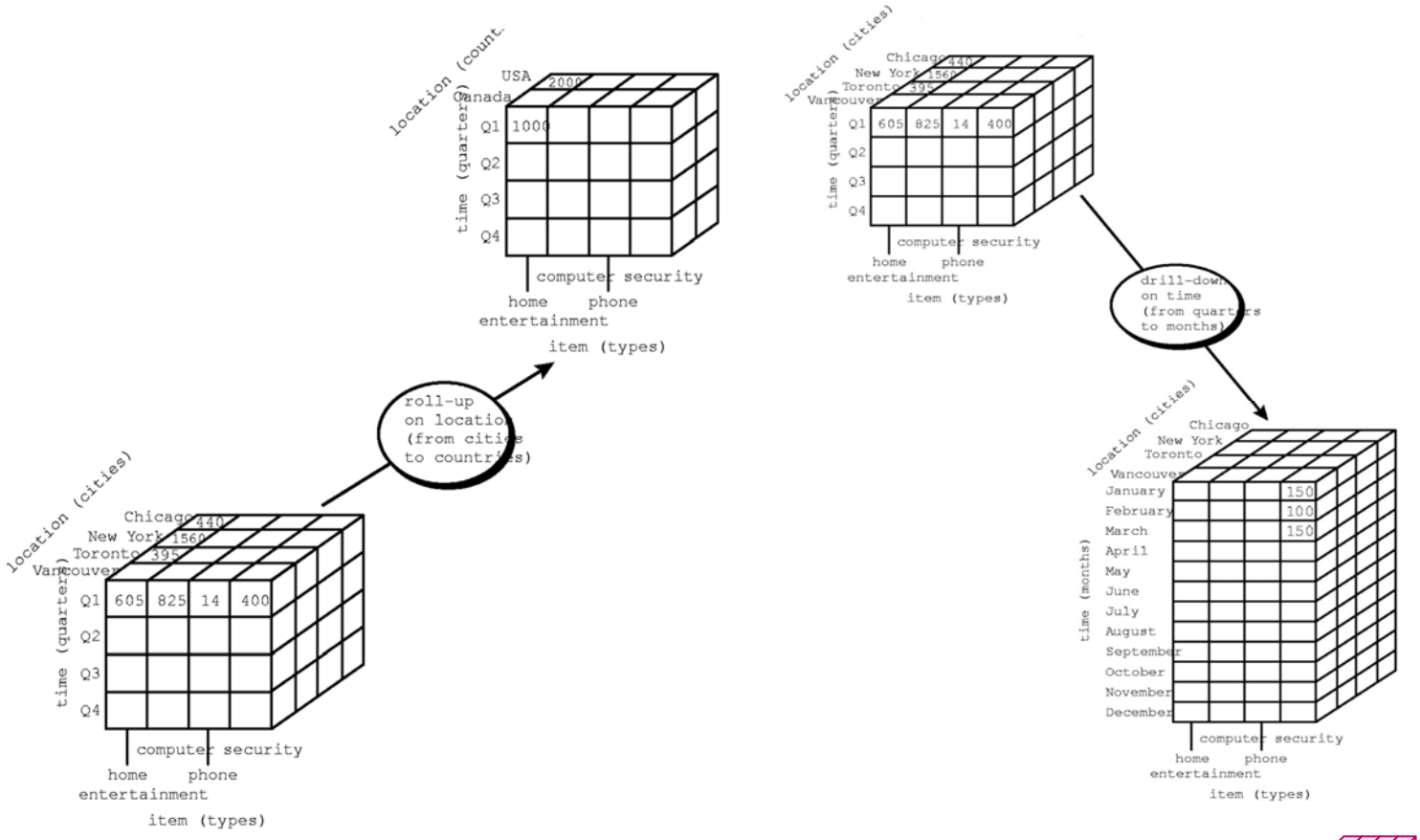
Quelle: J. Han, M. Kamber: Data Mining, Morgan Kaufmann

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

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

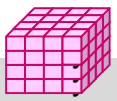
- Vorberechnung (Materialisierung) der Aggregationen zur schnellen Beantwortung von Aggregationsanfragen
- hoher Speicher- und Aktualisierungsaufwand bei vielen Dimensionen und vielen Dimensionselementen
  - i.a. kann nur kleiner Teil benötigter Aggregationen vorberechnet werden

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 -> **10^{14} Kombinationen** bei nur 3 Dimensionen
  - 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
  - für jedes Dimensionselement ist Aggregierung auf einer höheren Hierarchiestufe möglich
  - Kombinationsmöglichkeit mit jedem Element auf einer der Hierarchiestufen der anderen n-1 Dimensionen
- Anzahl Cuboiden bei n-dimensionalem Cube: $T = \prod_{i=1}^{n} (L_i + 1)$

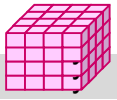
  L_i: #Ebenen von Dimension i (ohne Top-Level)

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

- betrifft Speicherung der Daten und Formulierung / Ausführung der Operationen
- MOLAP: direkte Speicherung in multi-dimensionalen Speicherungsstrukturen
  - Cube-Operationen einfach formulierbar und effizient ausführbar
  - begrenzte Skalierbarkeit auf große Datenmengen da riesige Anzahl von Zellen
- ROLAP: relationale Speicherung der Daten in Tabellen
  - effiziente Speicherung sehr großer Datenmengen
  - eher umständliche Anfrageformulierung mit SQL
  - 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
- (teilweise) Vorberechnung von Aggregationen wesentlich für gute Leistung



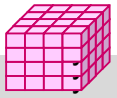
Multi-dimensionale Datenspeicherung

- 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
Smartphone	100	150	200
TV	50	170	150
Camcorder	20	120	100

Anfragen:
- wie hoch ist der Umsatz für TV in Thüringen
- wie hoch ist der Gesamtumsatz für Camcorder?



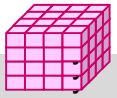
Multi-dimensionale Datenspeicherung (2)

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

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	-

- 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



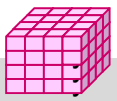
Sprachansatz MDX*

- 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

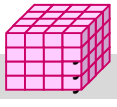
* http://msdn.microsoft.com/en-us/library/ms145506.aspx



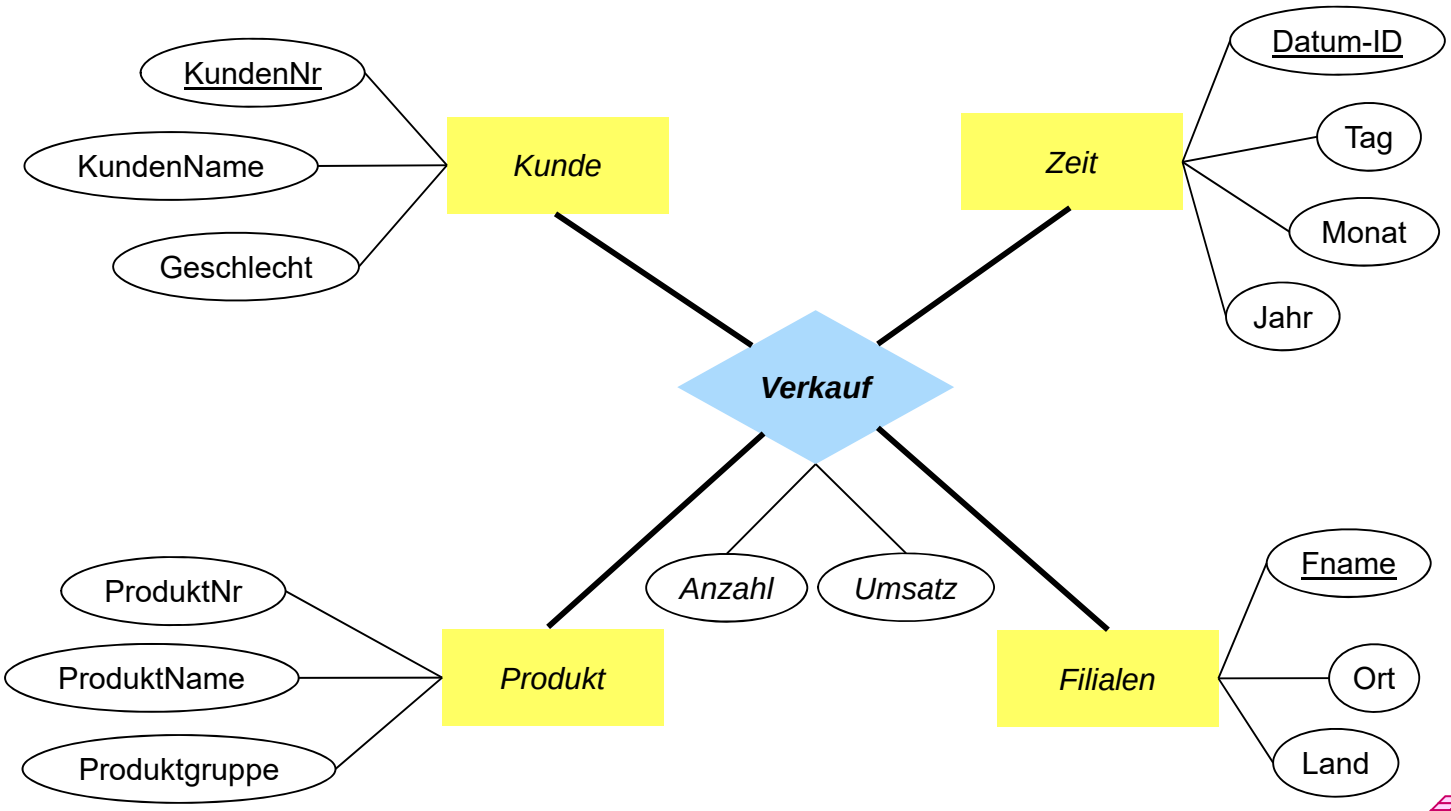
MDX: Beispiele

```
SELECT  Region.CHILDREN ON COLUMNS,
        Produkt.CHILDREN ON ROWS
FROM    Verkauf
WHERE   (Umsatz, Zeit.[2019])
```

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

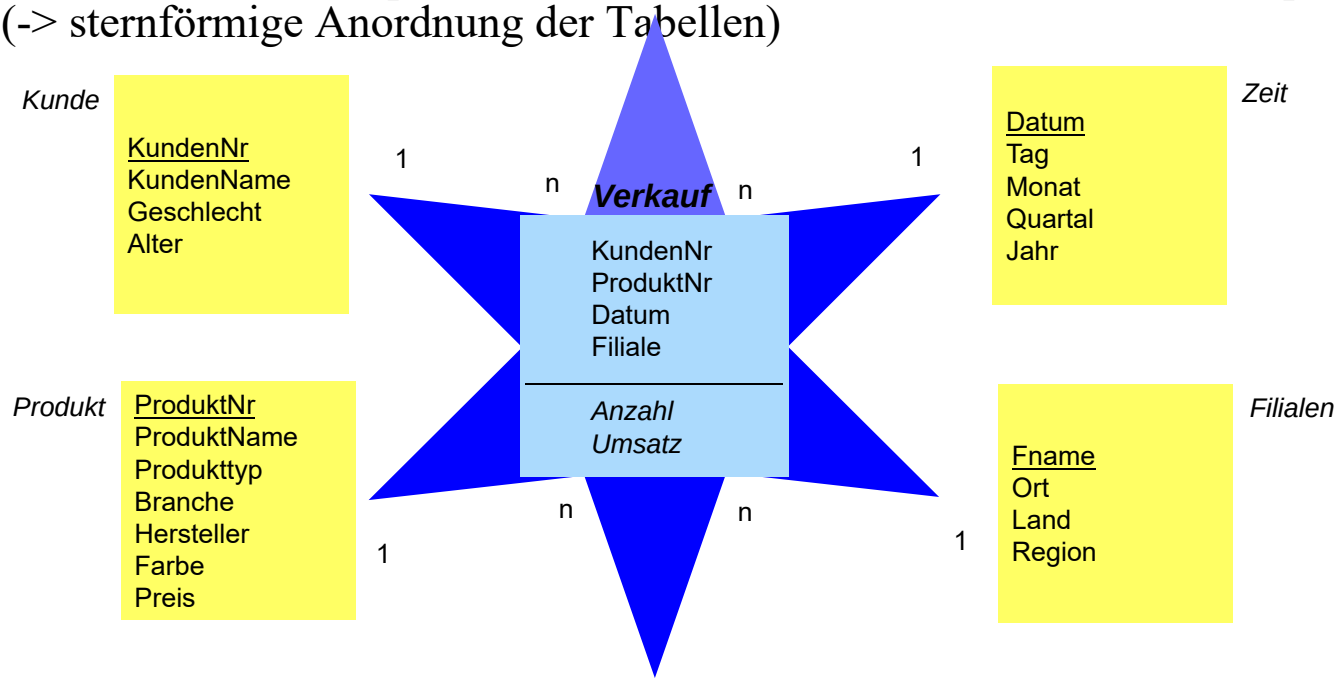


ER-Diagramm eines multi-dimensionalen Datenmodells



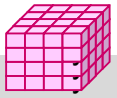
Relationale Speicherung: Star-Schema

- 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)



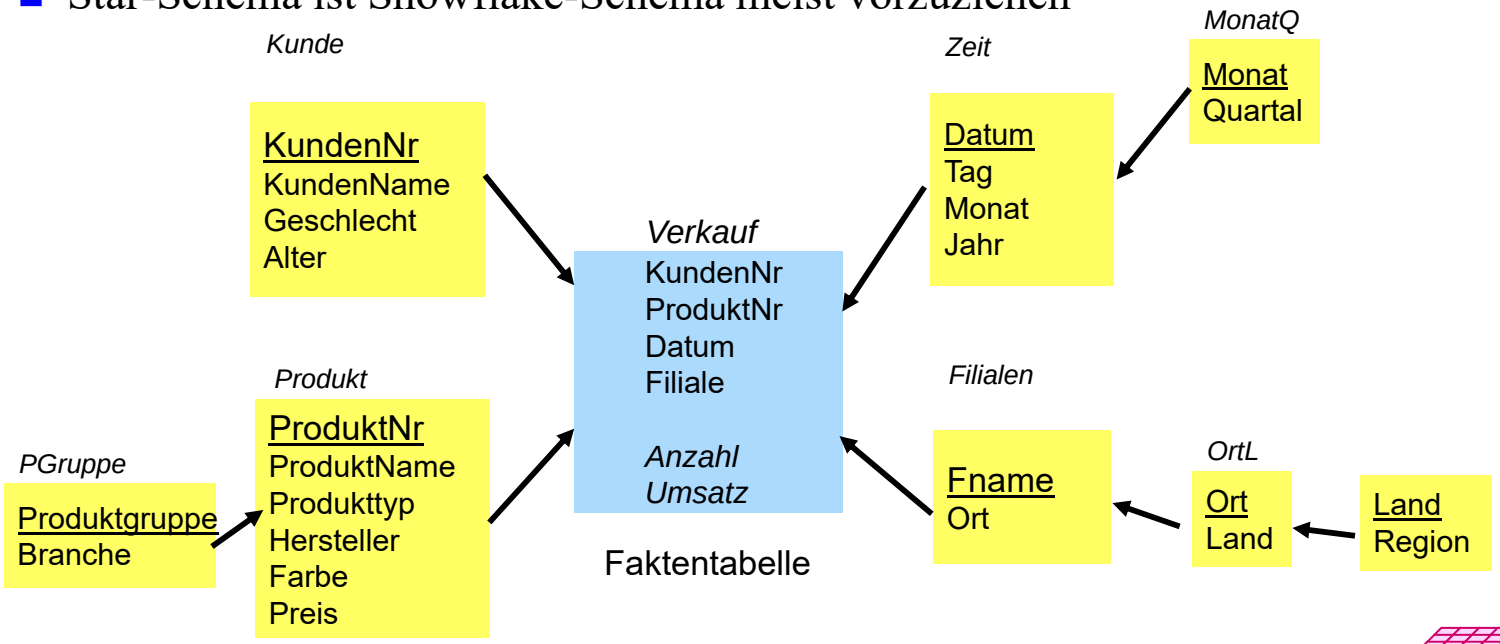
Star-Schema (2)

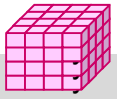
- formale Definition: Star-Schema besteht aus einer Menge von Tabellen D_1, ...D_n, F mit
  - Dimensionstabellen D_i bestehend aus (i.a. künstlichen) Primärschlüssel d_i und Dimensionsattributen
  - Faktentabelle F bestehend aus Fremdschlüsseln d_1, ... d_n sowie Meßgrößen (Kennzahlen) als weiteren Attributen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

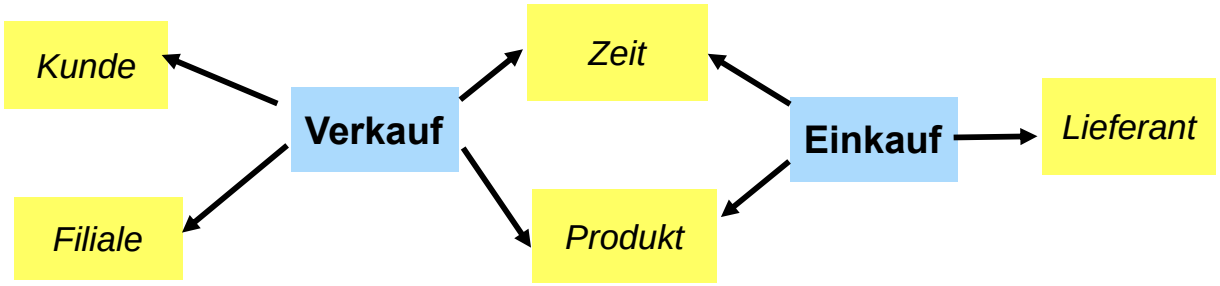




Galaxien-Schema

- 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 zur Analyse einer global auftretenden Viruserkrankung

- es sollen tagesweise die nachgewiesenen Infektionen und aufgetretenen Todesfälle zusammen mit dem Ort des Auftretens/Todes erfasst werden
- pro Infektion/Todesfall sollen anonyme Personenmerkmale wie Alter und Geschlecht erfasst werden
- Auswertungen über Infektionshäufigkeiten und Anzahl von Todesfällen sollen auch für unterschiedliche Zeiträume (Tag, Monate, Jahre) sowie für Länder und deren Kontinente möglich sein. Pro Land sollen auch die relativen Häufigkeiten bezogen auf je 1 Million Einwohner berechnet werden können.

Dimensionen:

Faktentabelle(n):

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

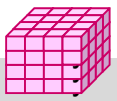
```
select      g1, ... gk, agg(f1), ... agg (fm)
from   D1, ..., Dn, F
where  <Selektionsbedingung auf D1> and
              ... and
       <Selektionsbedingung auf Dn> and
              D1.d1 = F.d1 and
               ... and
              Dn.dn = F.dn
group by    g1, ... gk
sort by     ...;
```

aggregierte Kennzahlen

Relationen des Star-Schemas

Join-Bedingungen

Ergebnis-Dimensionalität

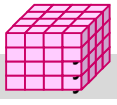


Beispiel eines Star-Join

- 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
2018	745
2019	710
2017	650



Mehrdimensionale Aggregationen mit Group-By

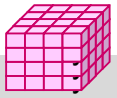
- Attributanzahl in **group by**-Klausel bestimmt Dimensionalität

```
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	2017	*2.000*
VW	2018	*3.000*
VW	2019	*3.500*
Opel	2017	*1.000*
...	...	...
BMW	2019	*1.500*
Ford	2017	*1.000*
Ford	2018	*1.500*
Ford	2019	*2.000*

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

Anzahl
19.500



Beispiel Covid-19-Daten

Download zB https://opendata.ecdc.europa.eu/covid19/casedistribution/csv

CSV-Import in relationale DB (MySql, PostgresSQL …)

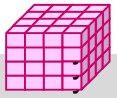
day	month	year	country	cases	deaths	pop	cont
27	4	2020	Finland	101	4	5518050	Europe
27	4	2020	France	461	242	66987244	Europe
27	4	2020	French_Poly...	0	0	277679	Oceania
27	4	2020	Gabon	0	0	2119275	Africa
27	4	2020	Gambia	0	0	2280102	Africa
27	4	2020	Georgia	30	1	3731000	Europe
27	4	2020	Germany	1018	110	82927922	Europe
27	4	2020	Ghana	271	1	29767108	Africa
27	4	2020	Gibraltar	5	0	33718	Europe
27	4	2020	Greece	0	0	10727668	Europe

Group-by continent (1-dim. Aggregation)

cont	cases	deaths
Africa	36,698	1,591
America	1,293,576	74,591
Asia	493,162	17,990
Europe	1,288,636	132,543
Oceania	8,129	116
Other	696	7

Group-by continent + month (drill-down, 2-dimens.)

cont	month	cases	deaths
Africa	2	3	0
Africa	3	5,122	166
Africa	4	31,573	1,425
America	2	76	0
America	3	188,658	3,687
America	4	1,104,842	70,904
Asia	2	73,468	2,679
Asia	3	85,873	3,990
Asia	4	333,821	11,321
Europe	2	1,106	23
Europe	3	407,028	27,125
Europe	4	880,502	105,395
Oceania	2	19	0
Oceania	3	5,298	21
Oceania	4	2,812	95

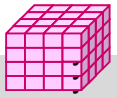


Cube-Operator

- SQL- CUBE-Operator für n-dimensionale Gruppierung und Aggregation
  - Syntax: *Group By CUBE (D_1, D_2, ... D_n)*
  - n-dimensionale Gruppierung/Aggregation + alle Gruppierungen geringerer Dimensionaltät; **2^n Aggregationen bei n Attributen** (4 bei n=2, 8 bei n=3 etc.)
  - bei niedriger Dimensionalität fasst ALL alle Werte einer Dimension zusammen
  - implementiert in MS SQL-Server, DB2, Oracle, PostgresSQL (ab V9.5), …
- Beispiele Auto-Verkauf (2D-Cube)

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

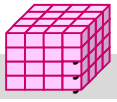
Hersteller	Jahr	*Anzahl*
VW	2017	*2.000*
VW	2018	*3.000*
VW	2019	*3.500*
Opel	2017	*1.000*
Opel	2018	*1.000*
Opel	2019	*1.500*
BMW	2017	*500*
BMW	2018	*1.000*
BMW	2019	*1.500*
Ford	2017	*1.000*
Ford	2018	*1.500*
Ford	2019	*2.000*
VW	**ALL**	*8.500*
Opel	**ALL**	*3.500*
BMW	**ALL**	*3.000*
Ford	**ALL**	*4.500*
ALL	2017	*4.500*
ALL	2018	*6.500*
ALL	2019	*8.500*
ALL	**ALL**	*19.500*



3D-Cube

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

Hersteller	Jahr	Geschlecht	Anzahl
VW	2017	m	1300
VW	2017	w	700
...	...	...	...
VW	2017	ALL	2.000
...	...	ALL	...
Ford	2019	ALL	2.000
VW	ALL	m	5.400
...	...	...	...
Ford	ALL	w	...
ALL	2017	m	...
...	...		
VW	ALL	ALL	8.500
...	...		
ALL	2017	ALL	...
...			
ALL	ALL	m	...
...			
ALL	ALL	ALL	19.500

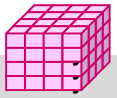


ROLLUP-Operator

- 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, da i.d.R. funktionale Abhängigkeiten (Land -> Kontinent, Datum -> Monat …)
- ROLLUP-Operator: intra-dimensionale Aggregierung
  - Syntax: *Group By ROLLUP (D_1, D_2, ... D_n)*
  - liefert nur die n+1 Gruppierungen

$d_1, d_2, \ldots, d_{n-1}, d_n, f()$, (Aggregationsfunktion f)
$d_1, d_2, \ldots, d_{n-1}$, ALL, f (),
...
d_1, ALL, ... , ALL, f (),
ALL, ALL, ... , ALL, f ()

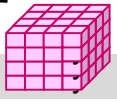
- Reihenfolge der Attribute relevant!



ROLLUP-Operator: Beispiel

```
select p.Hersteller as Hersteller,
  p. Marke as Marke, p.Farbe as
  Farbe, sum(v.Anzahl) as 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);
```

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

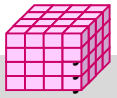


Rollup-Beispiel Covid-19-Daten

- MySQL-Anfrage (leicht abweichende Syntax)

```
SELECT cont, country, sum(cases) as cases, sum(deaths) as deaths,
FORMAT(((SUM(`cov-cases`.`deaths`) * 1000000) / `cov-cases`.`pop`),1) AS `DeathsPerMillion` FROM covid.`cov-cases`
group by cont, country with rollup;
```

cont	country	cases	deaths	DeathsPerMillion
Europe	Portugal	25056	989	96.2
Europe	Romania	12240	717	36.8
Europe	Russia	106498	1073	7.4
Europe	Serbia	9009	179	25.6
Europe	Slovakia	1396	23	4.2
Europe	Slovenia	1429	91	44.0
Europe	Spain	213435	24543	525.3
Europe	Sweden	21092	2586	253.9
Europe	Switzerland	29503	1422	167.0
Europe	Ukraine	9866	250	5.6
Europe	United_Kingdom	171253	26771	402.6
Europe	NULL	1302143	134421	
Oceania	Australia	6762	92	3.7
Oceania	New_Zealand	1132	19	3.9
Oceania	Papua_New_Guinea	8	0	0.0
Oceania	NULL	7902	111	
NULL	NULL	3201470	232231	



Grouping Sets

- mehrere Gruppierungen pro Anfrage

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

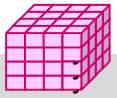
  leere Spezifikationsliste ( ) möglich: Aggregation über gesamte Tabelle

- 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));
```

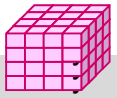
Hersteller	Farbe	*Anzahl*
VW	**ALL**	*8500*
Opel	**ALL**	*3500*
ALL	blau	*3100*
ALL	rot	*6200*
ALL	weiß	*2700*

- CUBE, ROLLUP, herkömmliches Group-By entsprechen speziellen Grouping-Sets
  - GROUP BY A,B -> GROUP BY GROUPING SETS (
  - GROUP BY ROLLUP (A,B) -> GROUP BY GROUPING SETS (
  - GROUP BY CUBE (A,B) -> GROUP BY GROUPING SETS (

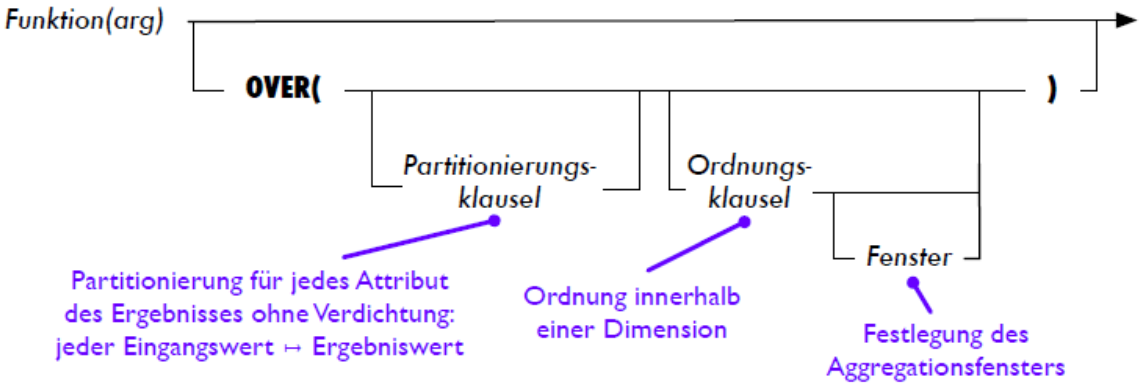


Zeitreihen

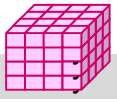
- viele Data-Warehouse-Fakten entsprechen Zeitreihen
  - Sequenz von Sätzen mit Zeit/Datumsangabe
  - Bsp.: Produktverkäufe, Erkrankungsfälle, Temperaturmesswerte, etc.
- Notwendigkeit Auswertungen auf bestimmte Zeitbereiche etc. zu begrenzen
- SQL kann daher Auswertung auf Fenster/Ausschnitte von Satzsequenzen ausführen
  - **OVER**-Prädikat in Select-Klausel oder **WINDOW**-Klausel
- unterstützt erweiterte Analysemöglichkeiten auf Sequenzen
  - Ranking: Berechnung von Rangfolgen / Top-N
  - kumulierte Häufigkeiten/Anteile (z.B. bezüglich eines Jahres/Monats)
  - fortgeschrittene Vergleiche, z.B.
    - Tagesinfektionen gegenüber gleitenden Wochendurchschnitt,
    - Monatsumsatz gegenüber gleitendem 3-Monatsdurchschnitt …
  - Unterstützung statistischer Funktionen wie Varianz (Funktionen `VAR_POP, VAR_SAMP`), Standardabweichung (`STDEV_POP, …`) etc



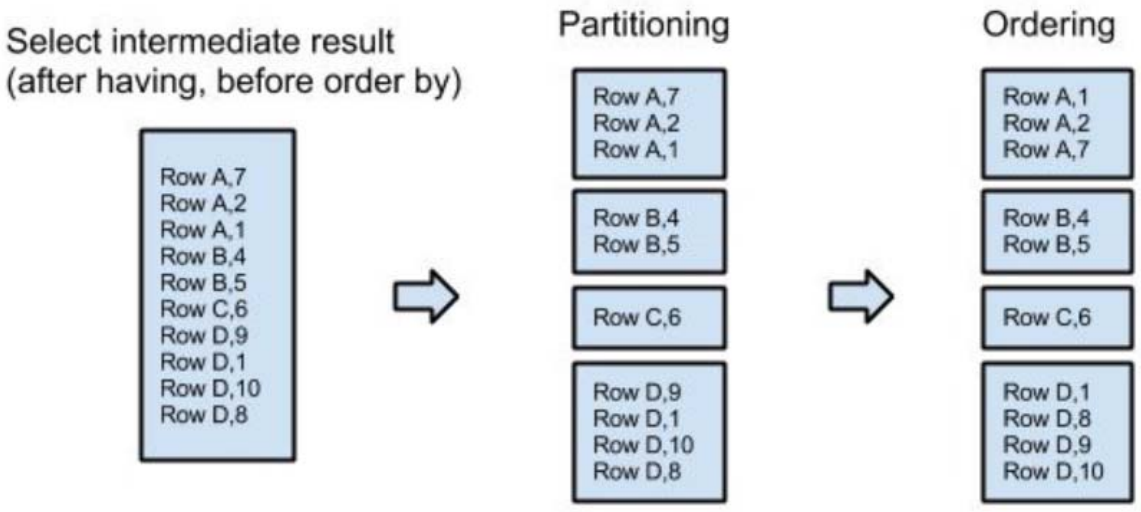
OVER-Prädikat



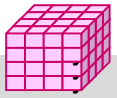
- Funktionen: `SUM, AVG,` **`RANK, DENSE_RANK,`** …
  - Berechnung bezüglich durch OVER spezifiziertes Intervall
- PARTITION BY: Zerlegung in mehrere Datensequenzen/ströme (optional)
- ORDER BY: Sortierreigenfolge pro Datensequenz (optional)
  - optionale Fensterangabe bei ORDER BY:
    tupelweise (ROWS) oder wertebereichsweise (RANGE) Einschränkung für Aggregationsfenster



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=2017
order by Anzahl desc, Hersteller
```

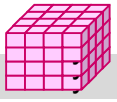
alternative Formulierung mit WINDOW-Klausel

```
select Hersteller, Anzahl,
   rank() over w as Rang,
   dense_rank() over w as DRang
from AVerkauf
where  Jahr=2017
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

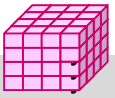
40



Rank-Beispiel (Covid-Datenbank)

```
SELECT country, sum(cases) as cases, sum(deaths) as deaths,
rank() over w1    as RankCases, rank() over w2    as RankDeaths
FROM covid.`cov-cases`
group by country
window w1 as (order by sum(cases) desc), w2 as  (order by sum(deaths) desc)
Limit 12
```

country	cases	deaths	RankCases	RankDeaths
United_States_of_America	1069826	63006	1	1
Italy	205463	27967	3	2
United_Kingdom	171253	26771	4	3
Spain	213435	24543	2	4
France	129581	24376	6	5
Belgium	48519	7594	13	6
Germany	159119	6288	5	7
Iran	94640	6028	9	8
Brazil	85380	5901	10	9
Netherlands	39316	4795	14	10
China	83956	4637	11	11
Canada	53236	3184	12	12



Rank-Funktion (2)

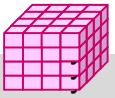
- partitionsweises Ranking

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

- weitere Ranking-Funktionen
  - **row_number():** Position des Satzes in Sequenz
  - **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

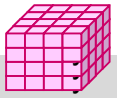
```
select Hersteller, percentile_disc(0.5) within
                     group (order by Verkaufspreis)
       over (partition by Hersteller)  as Preismedian,
from verkauf natural join produkt
```



Partitionsweises Ranking: Covid-Beispiel

```
select `cov-cases`.`country` AS `country`,`cov-cases`.`cont` AS `cont`,
sum(`cov-cases`.`cases`) AS `cases`, rank() OVER `w1` AS `GlobalRank`,
format((percent_rank() over w1),3) as PercentRank,
rank() OVER (PARTITION BY `cov-cases`.`cont` ORDER BY sum(`cov-cases`.`cases`) desc ) AS `ContRank`
from `covid`.`cov-cases` group by `cov-cases`.`country`
window `w1` AS (ORDER BY sum(`cov-cases`.`cases`) desc )
order by 4
```

country	cont	cases	GlobalRank	PercentRank	ContRank
United_States_of_America	America	1069826	1	0.000	1
Spain	Europe	213435	2	0.005	1
Italy	Europe	205463	3	0.010	2
United_Kingdom	Europe	171253	4	0.015	3
Germany	Europe	159119	5	0.020	4
France	Europe	129581	6	0.025	5
Turkey	Asia	120204	7	0.030	1
Russia	Europe	106498	8	0.035	6
Iran	Asia	94640	9	0.040	2
Brazil	America	85380	10	0.045	2
China	Asia	83956	11	0.050	3
Canada	America	53236	12	0.054	3
Belgium	Europe	48519	13	0.059	7
Netherlands	Europe	39316	14	0.064	8
Peru	America	36976	15	0.069	4
India	Asia	35043	16	0.074	4
Switzerland	Europe	29503	17	0.079	9
Portugal	Europe	25056	18	0.084	10
Ecuador	America	24934	19	0.089	5
Saudi_Arabia	Asia	22753	20	0.094	5



Aggregatberechnung auf Windows

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

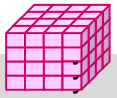
Summe der Verkäufe pro Tag sowie Anteil an Gesamtsumme

```
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
```

sales

date	value
1.2.2017	500
1.2.2017	300
5.7.2017	200
2.3.2018	400
3.4.2018	100
9.6.2019	500

date	day_sum	all_sum	anteil
1.2.2017	800	2000	40
5.7.2017	200	2000	10
2.3.2018	400	2000	20
3.4.2018	100	2000	5
9.6.2019	500	2000	25



Aggregate (2)

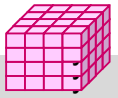
Summe der Verkäufe eines Tages im Verhältnis zu Verkäufen des Jahres

```
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
```

sales

date	value
1.2.2017	500
1.2.2017	300
5.7.2017	200
2.3.2018	400
3.4.2018	100
9.6.2019	500

date	day_sum	year_sum	janteil
1.2.2017	800	1000	80
5.7.2017	200	1000	20
2.3.2018	400	500	80
3.4.2018	100	500	20
9.6.2019	500	500	100



Bsp.: Kumulierte Summe

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

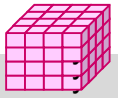
Summe der Verkäufe pro Tag sowie die kumulierten Gesamtverkäufe nach Tagen und die kumulierten Verkäufe im jeweiligen Jahr nach Tagen sortiert

```
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 date
order by date
```

sales

date	value
1.2.2017	500
1.2.2017	300
5.7.2017	200
2.3.2018	400
3.4.2018	100
9.6.2019	500

date	day_sum	cum_sum	cumy_sum
1.2.2017	800	800	800
5.7.2017	200	1000	1000
2.3.2018	400	1400	400
3.4.2018	100	1500	500
9.6.2019	500	2000	500



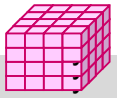
Covid-Beispiel Kum. Summen

Tagesfälle sowie kumulierte Summen insgesamt und pro Monat

```
SELECT day,month, cases, sum(cases) over w1 as cum_sum, sum(cases) over w2 as cum_msum
FROM covid.`cov-cases` where country="Germany" and cases<>0
window w1 as (order by month,day),
w2 as (partition by month order by month,day);
```

day	month	cases	cum_sum	cum_msum
28	1	1	1	1
29	1	3	4	4
31	1	1	5	5
1	2	2	7	2
2	2	1	8	3
3	2	1	9	4
4	2	2	11	6
7	2	1	12	7
8	2	1	13	8
12	2	2	15	10
26	2	2	17	12
27	2	4	21	16
28	2	26	47	42
29	2	10	57	52
1	3	54	111	54
2	3	18	129	72
3	3	28	157	100
4	3	39	196	139
5	3	66	262	205

day	month	cases	cum_sum	cum_msum
28	3	6294	48582	48525
29	3	3965	52547	52490
30	3	4751	57298	57241
31	3	4615	61913	61856
1	4	5453	67366	5453
2	4	6156	73522	11609
3	4	6174	79696	17783
4	4	6082	85778	23865
5	4	5936	91714	29801
6	4	3677	95391	33478
7	4	3834	99225	37312
8	4	4003	103228	41315
9	4	4974	108202	46289
10	4	5323	113525	51612
11	4	4133	117658	55745
12	4	2821	120479	58566
13	4	2537	123016	61103
14	4	2082	125098	63185
15	4	2486	127584	65671
16	4	2866	130450	68537
17	4	3380	133830	71917
18	4	3609	137439	75526
19	4	2458	139897	77984
20	4	1775	141672	79759
21	4	1785	143457	81544
22	4	2237	145694	83781
23	4	2352	148046	86133
24	4	2337	150383	88470



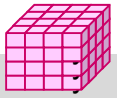
Windowing mit expliziter Fensterangabe

- 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 rows between 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)

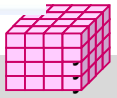


Beispiel Moving Average (Covid)

Infektionen pro Tag vs. Tagesdurchschnitt der letzten Woche

```
SELECT day,month, cases, avg (cases) over w1 as avgCasesWeek
FROM covid.`cov-cases` where country="Germany"
window w1 as (order by month,day rows 6 preceding);
```

day	month	Cases	avgCasesWeek
5	4	5,936	5,595
6	4	3,677	5,442
7	4	3,834	5,330
8	4	4,003	5,123
9	4	4,974	4,954
10	4	5,323	4,833
11	4	4,133	4,554
12	4	2,821	4,109
13	4	2,537	3,946
14	4	2,082	3,696
15	4	2,486	3,479
16	4	2,866	3,178
17	4	3,380	2,901
18	4	3,609	2,826
19	4	2,458	2,774
20	4	1,775	2,665
21	4	1,785	2,623
22	4	2,237	2,587
23	4	2,352	2,514
24	4	2,337	2,365
25	4	2,055	2,143
26	4	1,737	2,040
27	4	1,018	1,932
28	4	1,144	1,840
29	4	1,304	1,707
30	4	1,478	1,582



Explizites Windowing (2)

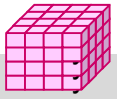
- partitionsweises Windowing

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

  Amount ist positiv für Zubuchung, negativ für Abbuchung

Bestimme Kontostand (kumulierte Summe) pro Konto nach jeder Kontobewegung

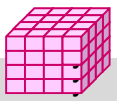
```
select account-number, date-time,
       sum (amount) over
        (partition by account-number   order by date-time
          rows unbounded preceding ) as balance
from transaction
order by account-number, date-time
```



Covid-Beispiel

```
select country, day, month, cases, sum(cases) OVER w1  AS cum_sum,
from cov-cases where cases <> 0
window w1 AS (partition by country
              ORDER BY month,day rows unbounded preceding)
```

country	day	month	cases	cum_sum
Germany	27	4	1018	155193
Germany	28	4	1144	156337
Germany	29	4	1304	157641
Germany	30	4	1478	159119
Italy	31	1	3	3
Italy	22	2	14	17
Italy	23	2	62	79
Italy	24	2	53	132
Italy	25	2	97	229
Italy	26	2	93	322
Italy	27	2	78	400
Italy	28	2	250	650
Italy	29	2	238	888
Italy	1	3	240	1128
Italy	2	3	561	1689
Italy	3	3	347	2036
Italy	4	3	466	2502
Italy	5	3	587	3089
Italy	6	3	769	3858
Italy	7	3	778	4636
Italy	8	3	1247	5883
Italy	9	3	1492	7375
Italy	10	3	1797	9172
Italy	11	3	977	10149



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 für Cubes bzw. SQL-Erweiterungen
  - Mehrdimensionale Gruppierung/Aggregation: CUBE-, ROLLUP, GROUPING SETS
  - Windowing-Funktionen für Zeitreihen / Datenströme