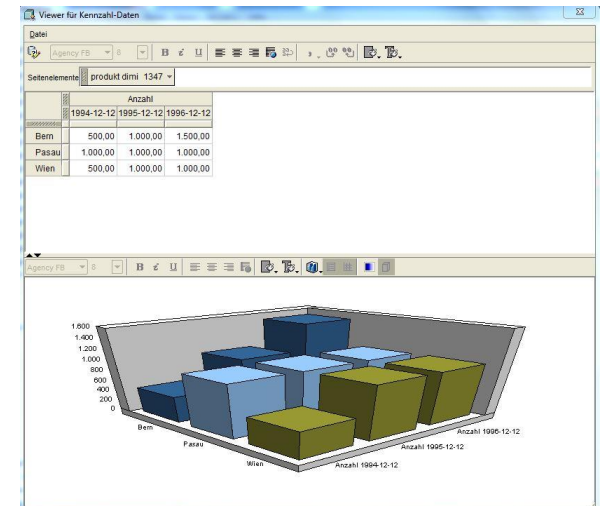
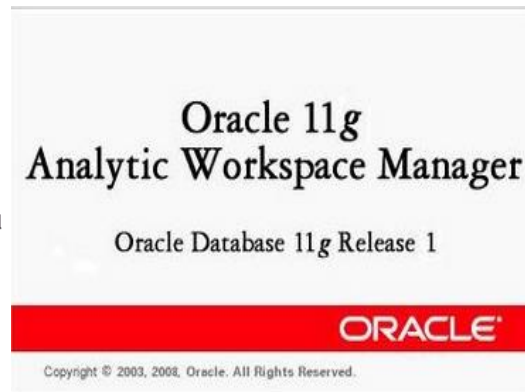
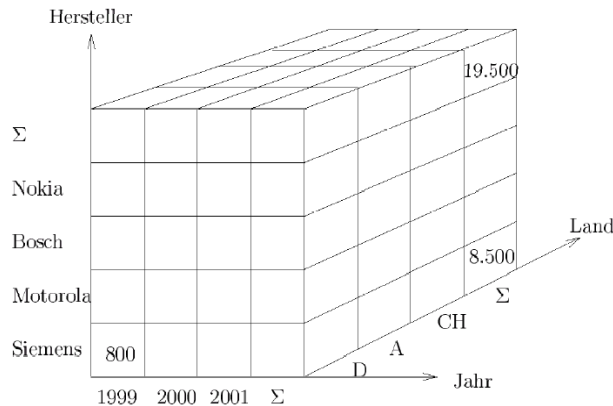
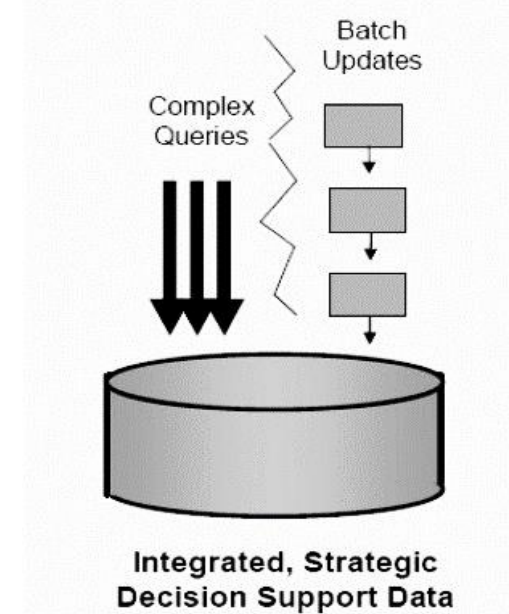
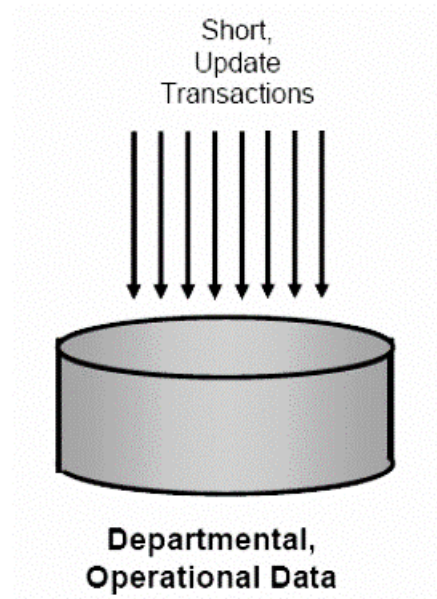


Datawarehouse

„Ein Data Warehouse ist eine physische Datenbank mit integrierter Sicht auf beliebige Daten zu Analysezwecken.“ (vgl. Bauer & Günzel, 2004)



Datawarehouse



Benutzertyp	Sachbearbeiter, operativ (OLTP)	Management, Planung (DWH/OLAP)
Benutzerzahl	Tausende	Tendenziell weniger
Antwortzeit	Sekunden	Sekunden bis Minuten, sogar Stunden bei komplexen Anfragen
Anwendung	Verwaltung, operatives Geschäft	Analyse, Entscheidungsunterstützung
Anfragetyp	Record-/tupelorientiert, vordefiniert	Multidimensional, aggregiert, ad-hoc
Transaktionstyp	Kurze Lese-/Schreiboperationen	Lange Leseoperationen
Datenverwaltungsziel	Transaktionskonsistenz (ACID)	Historisierung
Datenbankgröße	Megabytes bis Gigabytes	Gigabytes bis Terabytes
Optimierungsanforderung	Indexzugriff über Primärschlüssel	Full Table Scans

Gründe für die Trennung von OLTP- und DWH/OLAP-Systemen

- OLTP optimiert für kurze Transaktionen und bekannte Lastprofile
- Komplexe OLAP-Anfragen degradieren die Performance von Transaktionen des operationalen Betriebs
- Spezieller physischer und logischer Datenbankentwurf für multidimensionale Anfragen notwendig
- Transaktionseigenschaften (ACID) im DWH unwichtig, da nur Lesezugriff
- Es werden historische Daten benötigt, die in OLTP-Systemen typischerweise nicht vorliegen
- Konsolidierung (Integration, Bereinigung und Aggregation) von Daten aus verschiedenen heterogenen Datenquellen
- Datenredundanz (Speicherung vorberechneter Aggregate auf verschiedenen Ebenen) kann aus Performance-Gründen notwendig sein; ist im DWH kontrollierbar
- Sicherheit (z.B. Anonymisierung der Daten im DWH)

Datawarehouse

Data Warehousing mit ETL- und OLAP-Werkzeugen entwickelte sich Mitte der 90er Jahre

Die Daten werden a priori integriert und zur Analyse im Data Warehouse gespeichert

DSS/EIS können nun auf Data Warehouses mit Hilfe von Standard-OLAP-Werkzeugen zugreifen

Aber es gehört mehr zur Entscheidungsunterstützung als nur OLAP:

- In die Zukunft gerichtete Analysen (Prognosesysteme, Data Mining)
- Dokumente/unstrukturierte Daten

In den letzten Jahren hat der Begriff „Business Intelligence“ den Ausdruck OLAP weitestgehend abgelöst, er ist jedoch üblicherweise weiter gefasst (inkludiert Data Mining, fachliche Anwendungen, teils unstrukturierte Daten, Portale, ...)

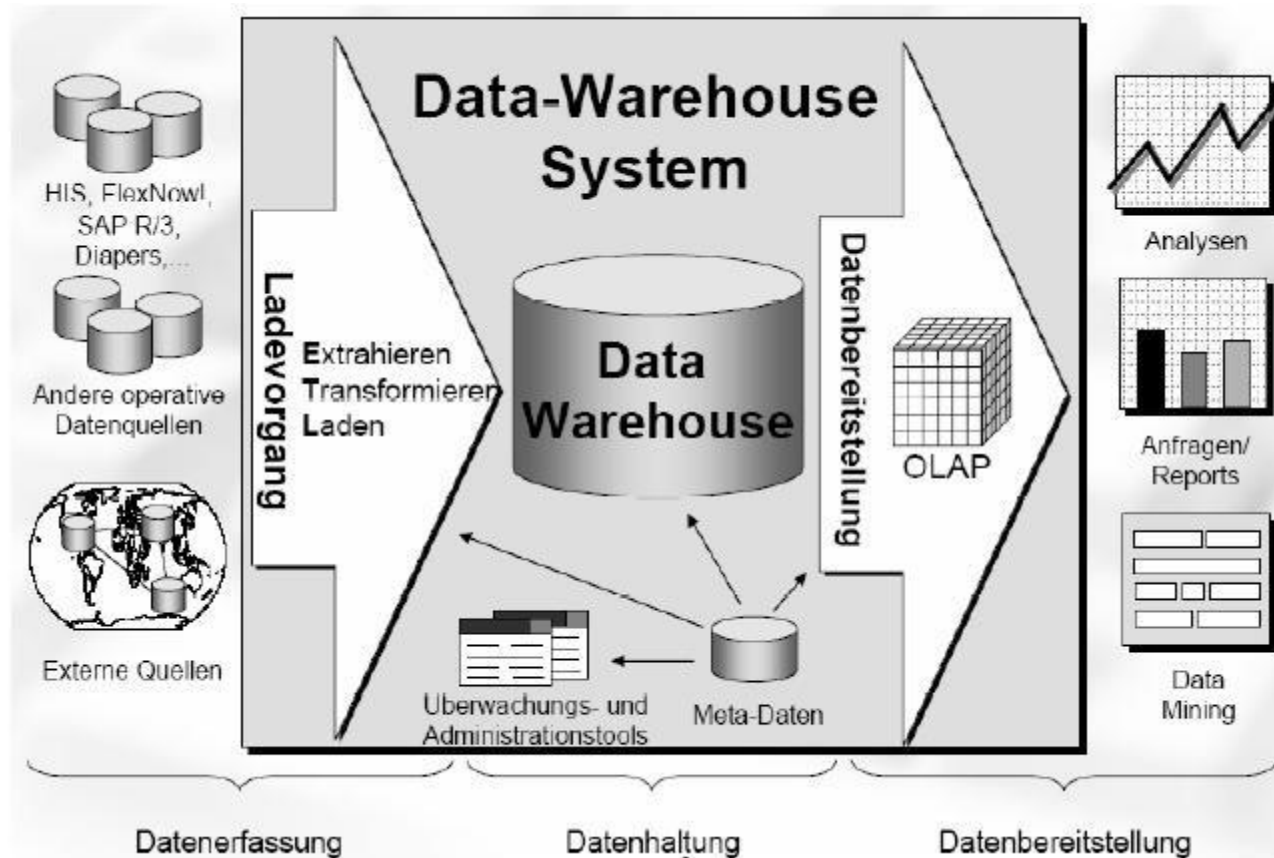
Die verschiedenen OLAP-Modelle

Ein Data Warehouse beruht im Normalfall auf einer relationalen Datenbank. Wie im vorherigen Abschnitt beschrieben werden die Daten daher in Tabellenform verwaltet. Um diese Daten auch mit den OLAP - Funktionalitäten auswerten zu können, müssen die Daten von der Tabellenstruktur in die „Würfel-Zellenstruktur“ transformiert werden. Die Art und Weise wie dies gemacht wird, bestimmt die OLAP-Architektur.

Man unterscheidet zwischen

- relationalem OLAP (ROLAP)
- multidimensionalem OLAP (MOLAP)

Datawarehouse



- **ETL** = Extract, Transform, Load (Integrationsschicht)

Relationales OLAP-Modell

Beim relationalen OLAP werden die Daten erst während der Nutzung aus der relationalen Datenbank gelesen. Die multidimensionalen Strukturen werden mit Hilfe des Star- oder Snowflake-Schemas in einem hierfür erweiterten RDBMS abgelegt. Unter dem Begriff Star-Schema versteht man eine Datenbankstruktur mit einer Basistabelle, der sogenannten Fakt-Tabelle, um die sternenförmig die Referenztabellen (Dimensionstabellen) angeordnet werden.

Eine sehr große Faktentabelle

- Alle Verkäufe der letzten drei Jahre
- Alle Telefonate des letzten Jahres
- Alle Flugreservierungen der letzten fünf Jahre
- normalisiert

Mehrere Dimensionstabellen

- Zeit
- Filialen
- Kunden
- Produkt
- Oft nicht normalisiert

Stern-Schema (1 / 2)

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

Faktentabelle (SEHR groß)

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

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

Dimensionstabellen (relativ klein)

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

Stern-Schema (2 / 2)

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

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

Nicht-normalisierte Dimensionstabellen: effizientere Abfrageauswertung

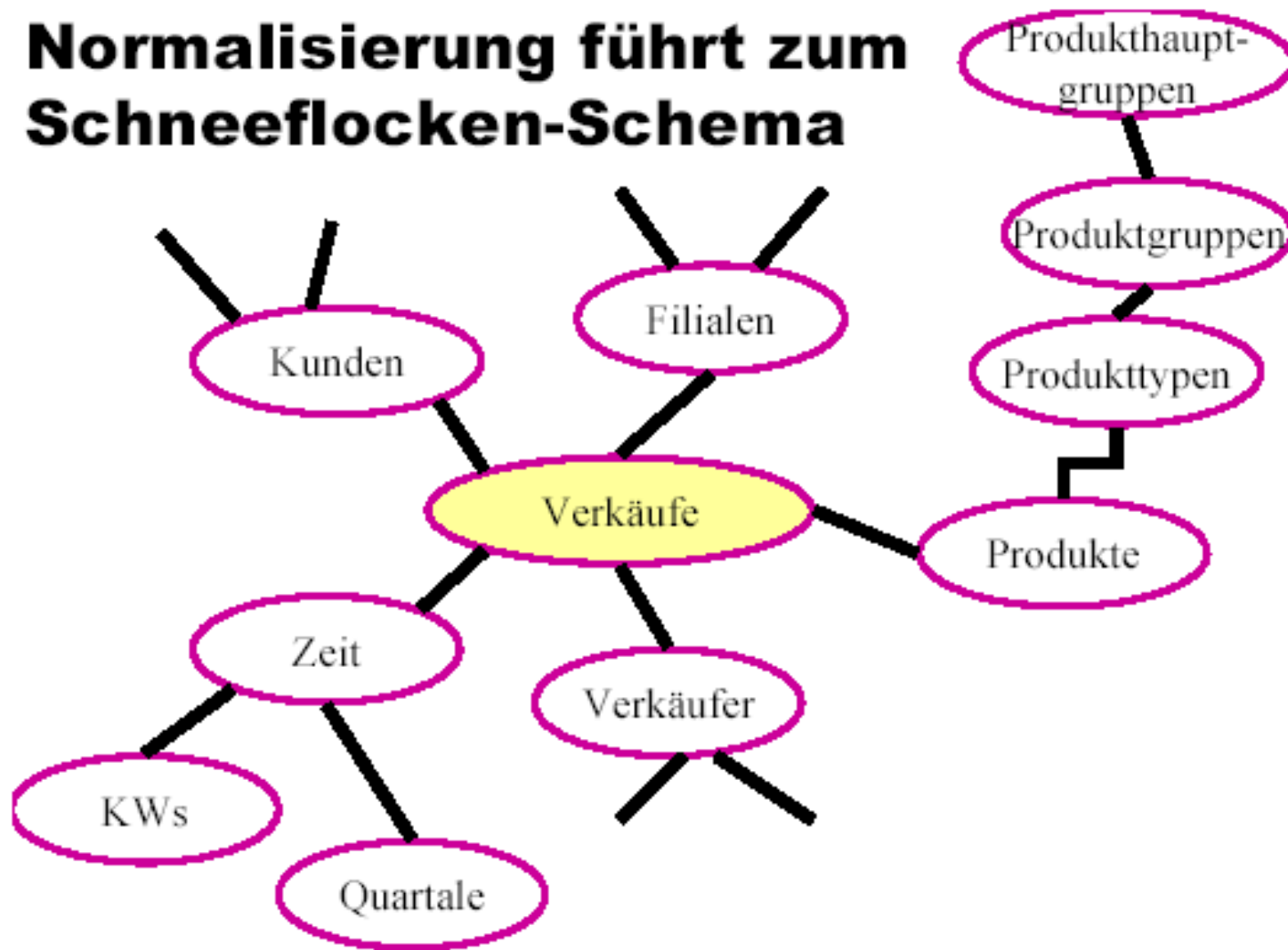
Zeit							
Datum	Tag	Monat	Jahr	Quartal	KW	Wochentag	Saison
25-Jul-00	25	7	2000	3	30	Dienstag	Hochsommer
...		
18-Dec-01	18	12	2001	4	52	Dienstag	Weihnachten
...

Datum → Monat → Quartal

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

ProduktNr → Produkttyp → Produktgruppe → Produkthauptgruppe

Normalisierung führt zum Schneeflocken-Schema



Star Join

Das Sternschema führt bei typischen Abfragen zu sogenannten *Star Joins*:

Welche Handys (d.h. von welchen Herstellern) haben junge Kunden in den bayrischen Filialen zu Weihnachten 1996 gekauft ?

```
select sum(v.Anzahl), p.Hersteller
from Verkäufe v, Filialen f, Produkte p, Zeit z, Kunden k
where z.Saison = 'Weihnachten' and z.Jahr = 1996 and k.wiealt < 30
and p.Produkttyp = 'Handy' and f.Bezirk = 'Bayern'
and v.VerkDatum = z.Datum and v.Produkt = p.ProduktNr
and v.Filiale = f.Filialenkennung and v.Kunde = k.KundenNr
group by Hersteller;
```

OLAP-Operationen zur explorativen Navigation

Der Verdichtungsgrad bei einer SQL-Anfrage wird durch die **group by**-Klausel gesteuert. Werden mehr Attribute in die **group by**-Klausel aufgenommen, spricht man von einem *drill down*. Werden weniger Attribute in die **group by** Klausel aufgenommen, spricht man von einem *roll up*.

Wieviele Handys wurden von welchem Hersteller in welchem Jahr verkauft ?

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

Wieviele Handys wurden in welchem Jahr verkauft ?

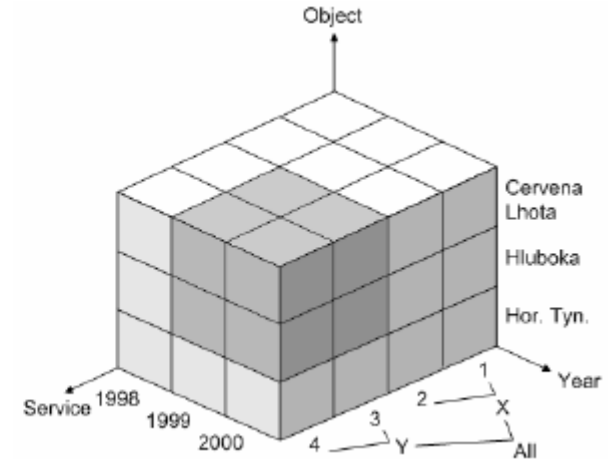
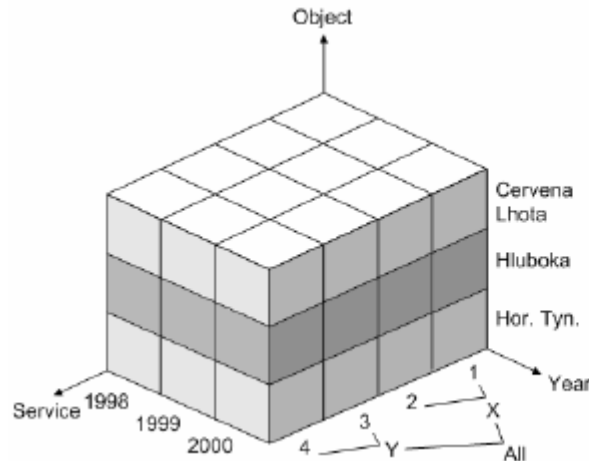
```
select Jahr, sum(Anzahl)
from   Verkäufe v, Produkte p, Zeit z
where  v.Produkt = p.ProduktNr
and    v.VerkDatum = z.Datum
and    p.Produkttyp = 'Handy'
group by z.Jahr;
```

Wieviele Handys wurden von welchem Hersteller verkauft ?

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

OLAP-Operationen zur explorativen Navigation

- Slice und Dice: Auswahl eines Teilwürfels (z.B. Beschränkung auf Objekt „Hluboka“)



- Drill-down und Roll-up: Auswahl des Detaillevels/ Aggregierungsgrades
- Split und Merge: erhöht bzw. verringert die Dimensionalität des Ergebniswürfels, z.B. je Objekt -> je Objekt und Jahr
- Drill-Across: weist betrachtete Kennzahlen in anderen Dimensionen aus
- Drill-Through: betrifft die Architektur des DWH, z.B. Wechsel der Datenquelle

Ergebnis SQL-Statments

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

Handyverkäufe nach Jahr	
Jahr	Anzahl
1999	4.500
2000	6.500
2001	8.500

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

Materialisierung von Aggregaten

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

Da es sehr zeitaufwendig ist, die Aggregation jedes Mal neu zu berechnen, empfiehlt es sich, sie zu materialisieren, d.h. die vorberechneten Aggregate verschiedener Detaillierungsgrade in einer Relation abzulegen. Es folgen einige SQL-Statements, welche die linke Tabelle der vorherigen Folie erzeugen. Mit dem **null**-Wert wird markiert, dass entlang dieser Dimension die Werte aggregiert wurden.

Relationale Struktur der Datenwürfel

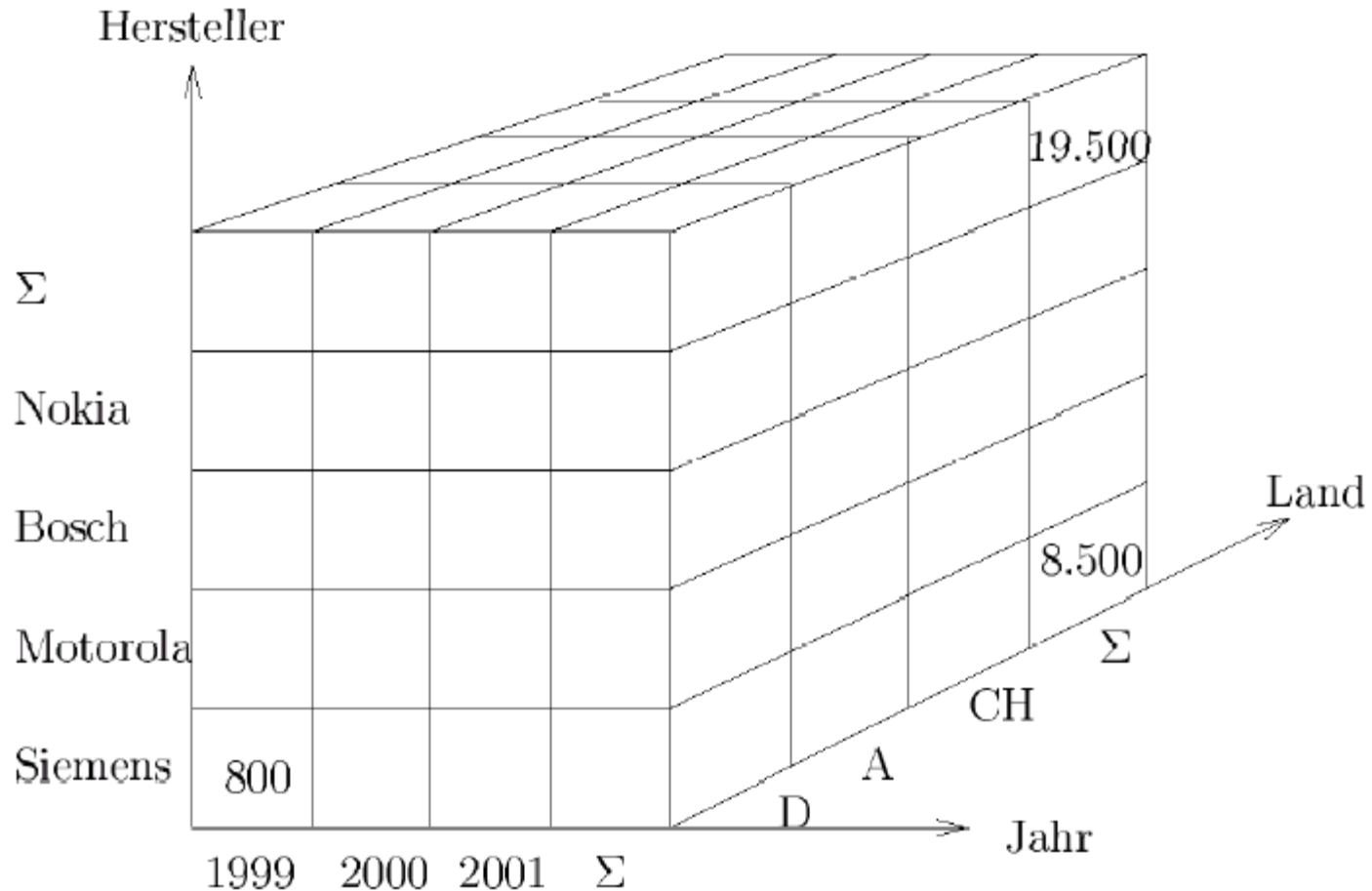
2hoch2

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

2hoch3

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

Würfeldarstellung Handy3DCube



CUBE-Operator

Offenbar ist es recht mühsam, diese Art von Anfragen zu formulieren, da bei n Dimensionen insgesamt 2^n Unteranfragen formuliert und mit **union** verbunden werden müssen. Außerdem sind solche Anfragen extrem zeitaufwendig auszuwerten, da jede Aggregation individuell berechnet wird, obwohl man viele Aggregate aus anderen (noch nicht so stark verdichteten) Aggregaten berechnen könnte.

Der **cube**-Operator

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

analytic workspace manager

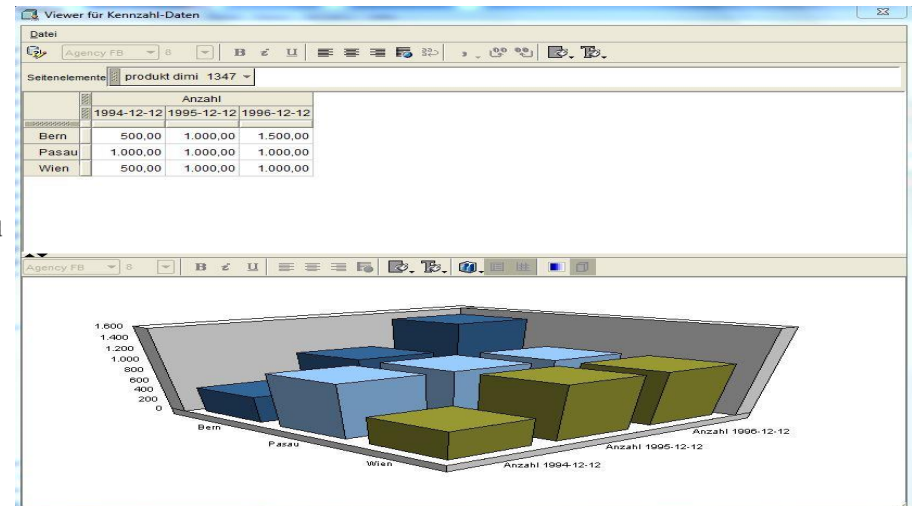
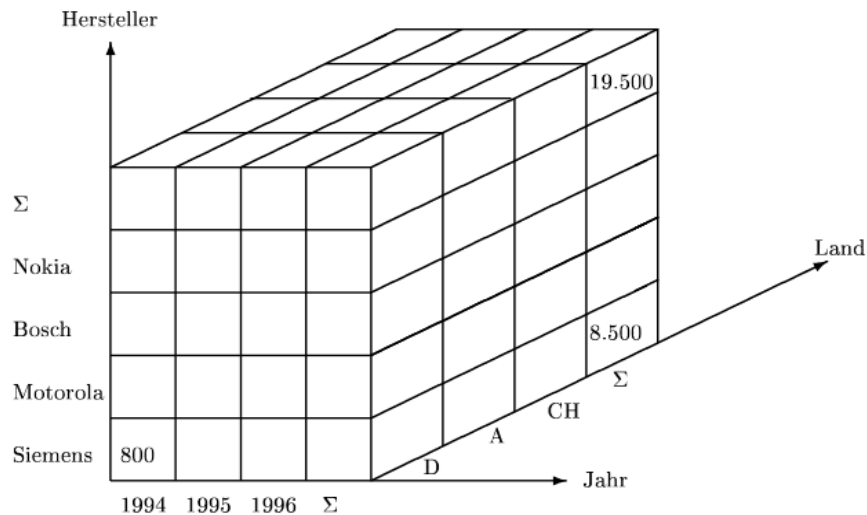
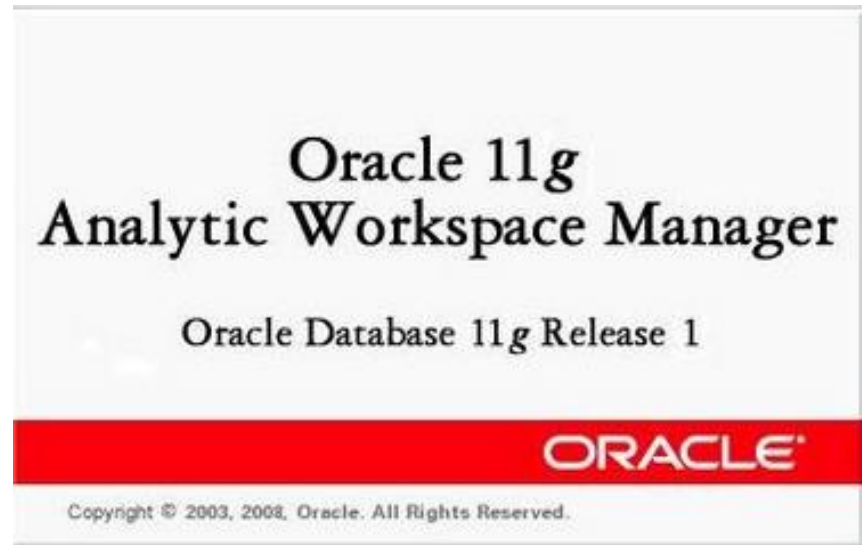
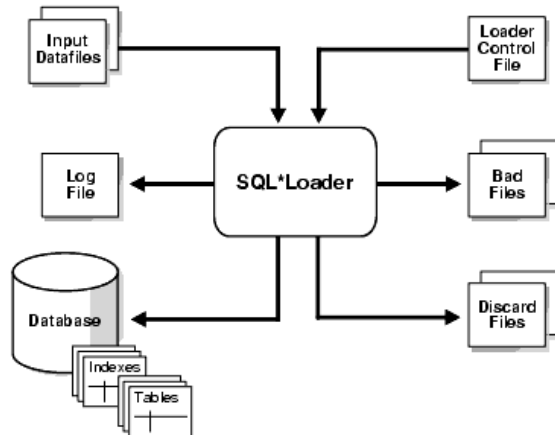








Abbildung 17.7: Würfeldarstellung der Handyverkaufszahlen nach Jahr, Hersteller und Land

analytic workspace manager

TUMFART_OLAP.VERKAEUFE	
VERKDATUM	DATE
PRODUKT	VARCHAR2 (100 BYTE)
ANZAHL	NUMBER (4)
KUNDE	NUMBER (4)
VERKAEUFER	NUMBER (4)
FILIALE	VARCHAR2 (100 BYTE)

TUMFART_OLAP.FILIALE	
P *	FILIALENKENNUNG VARCHAR2 (100 BYTE)
	LAND VARCHAR2 (100 BYTE)
	FILIALENKENNUNG_PRIMARY_KEY (FILIALENKENNUNG)
	FILIALENKENNUNG_PRIMARY_KEY (FILIALENKENNUNG)

TUMFART_OLAP.PRODUKTE	
P *	PRODUKTNR VARCHAR2 (100 BYTE)
	PRODUKTTYP VARCHAR2 (100 BYTE)
	HERSTELLER VARCHAR2 (100 BYTE)
	PRODUKTENR_PRIMARY_KEY (PRODUKTNR)
	PRODUKTENR_PRIMARY_KEY (PRODUKTNR)

TUMFART_OLAP.ZEIT	
P *	DATUM DATE
	TAG NUMBER (2)
	MONAT NUMBER (2)
	JAHR NUMBER (4)
	QUARTAL NUMBER (2)
	SAISON VARCHAR2 (100 BYTE)
	DATUM_PRIMARY_KEY (DATUM)
	DATUM_PRIMARY_KEY (DATUM)

analytic workspace manager

The screenshot displays the 'Analytic Workspace Manager' interface. On the left, a tree view shows the hierarchy: 'Datenbanken' > 'delphi (tumfart_olap) - OLAP 11g' > 'Schemas' > 'TUMFART_OLAP' > 'Analytische Workspaces' > 'AW_TUMFART' > 'AW_TUMFART_VERKAUF (im R...' > 'Dimensionen' > 'ZEIT_DIMI'. The 'ZEIT_DIMI' dimension is selected, and its properties are shown on the right.

Typ der Dimensionstabellen: Star-Schema

ZEIT_DIMI	Quellspalte
Hierarchien	
DATUM_CHANNEL	
DATUM	
Element	TUMFART_OLAP.ZEIT.DATUM
LONG_DESCRIPTION	TUMFART_OLAP.ZEIT.DATUM
SHORT_DESCRIPTION	TUMFART_OLAP.ZEIT.DATUM
END_DATE	TUMFART_OLAP.ZEIT.DATUM
TIME_SPAN	100

Zeitdimension

analytic workspace manager

The screenshot displays the Analytic Workspace Manager interface. On the left, a tree view shows the database structure: **Datenbanken** > **delphi (tumfart_olap) - OLAP 11g** > **Schemas** > **TUMFART_OLAP** > **Analytische Workspaces** > **AW_TUMFART** > **AW_TUMFART_VERKAUF (im R...)** > **Dimensionen** > **FILIALE_DIMI**. The **FILIALE_DIMI** dimension is selected, showing its components: **Ebenen** (containing **FILIALENKENNUNG**), **Hierarchien** (containing **FILIALEN_CHANNEL**), **Attribute** (containing **LONG_DESCRIPTION** and **SHORT_DESCRIPTION**), **Zuordnungen**, **Views**, and **Datensicherheit**.

On the right, the **Typ der Dimensionstabellen:** dropdown is set to **Star-Schema**. Below it, a table lists the dimension's properties:

FILIALE_DIMI	Quellspalte
<input type="checkbox"/> Hierarchien	
<input type="checkbox"/> FILIALEN_CHANNEL	
<input type="checkbox"/> FILIALENKENNUNG	
Element	TUMFART_OLAP.FILIALE.FILIALENKENNUNG
LONG_DESCRIPTION	TUMFART_OLAP.FILIALE.FILIALENKENNUNG
SHORT_DESCRIPTION	TUMFART_OLAP.FILIALE.FILIALENKENNUNG

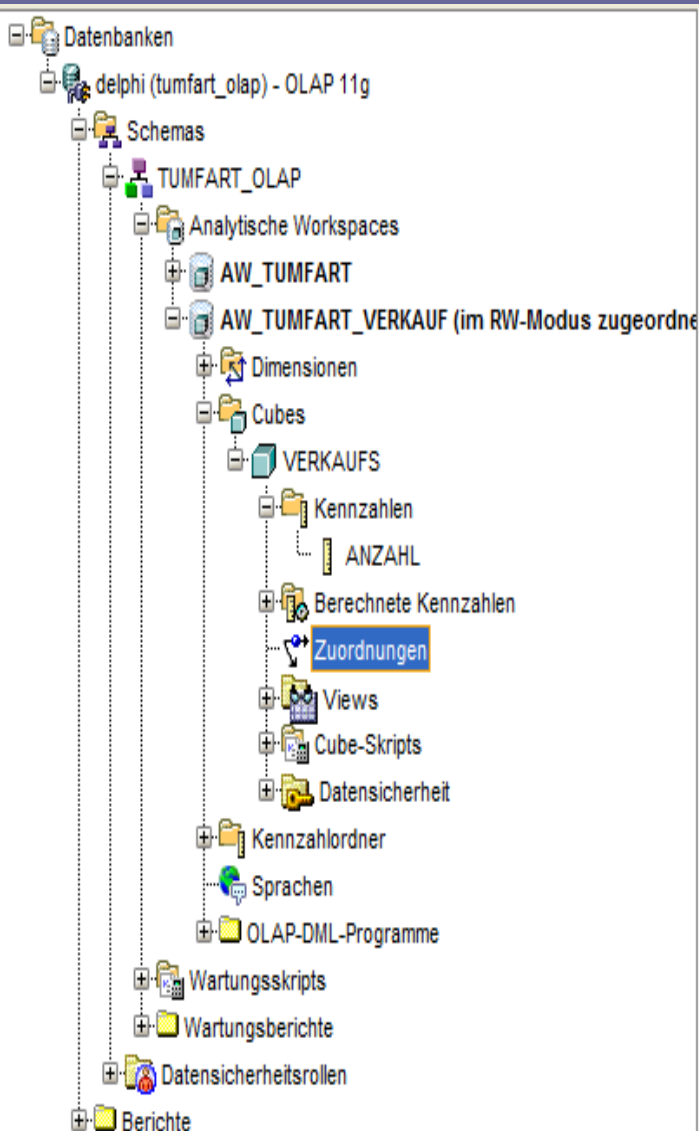
Filialedimension

analytic workspace manager

Produktdimension

PRODUKT_DIMI	Quellspalte
Hierarchien	
PRODUKTNR_CHANNEL	
PRODUKTNR	
Element	TUMFART_OLAP.PRODUKTE.PRODUKTNR
LONG_DESCRIPTION	TUMFART_OLAP.PRODUKTE.PRODUKTNR
SHORT_DESCRIPTION	TUMFART_OLAP.PRODUKTE.PRODUKTNR

analytic workspace manager

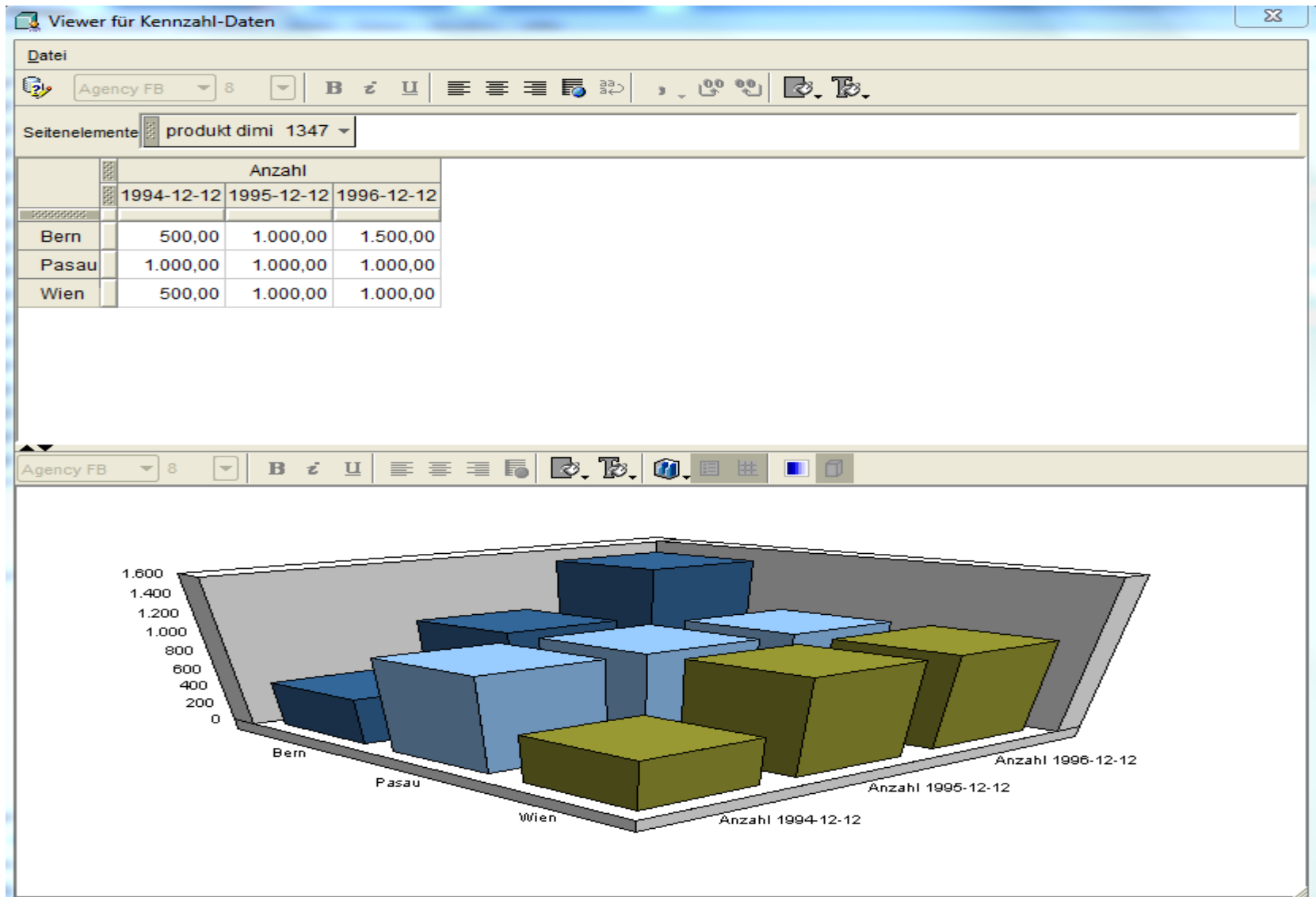


The screenshot shows the configuration table for the 'VERKAUFS' cube. The table has two columns: 'VERKAUFS' and 'Quellspalte'. The rows are organized into sections: 'Kennzahlen' (containing 'ANZAHL'), 'Dimensionen' (containing 'ZEIT_DIMI' and 'FILIALE_DIMI'), and 'PRODUKT_DIMI'. Each dimension row is followed by a 'Join-Bedingung' row. There is also a 'Filter' row at the bottom.

VERKAUFS	Quellspalte
Kennzahlen	
ANZAHL	TUMFART_OLAP.VERKAEUFE.ANZAHL
Dimensionen	
ZEIT_DIMI	
DATUM $\Sigma \uparrow$	TUMFART_OLAP.ZEIT.DATUM
Join-Bedingung	TUMFART_OLAP.VERKAEUFE.VERKDATUM = TUMFART_OLAP.ZEIT.DATUM
FILIALE_DIMI	
FILIALENKENNUNG $\Sigma \uparrow$	TUMFART_OLAP.FILIALE.FILIALENKENNUNG
Join-Bedingung	TUMFART_OLAP.VERKAEUFE.FILIALE = TUMFART_OLAP.FILIALE.FILIALENKENNUNG
PRODUKT_DIMI	
PRODUKTNR $\Sigma \uparrow$	TUMFART_OLAP.PRODUKTE.PRODUKTNR
Join-Bedingung	TUMFART_OLAP.VERKAEUFE.PRODUKT = TUMFART_OLAP.PRODUKTE.PRODUKTNR
Filter	

Cube

analytic workspace manager - Ergebnis



Weiteres CUBE-Operator Beispiel

```
select deptno,job,sum(sal)
from emp
group by cube(deptno,job) order by deptno;
```

Bildet alle möglichen Kombinationen von
Zwischensummen inklusive die
Gesamtsumme.

ZB.: Gehaltssumme je Job und je Abteilung,
Gehaltssumme je Job + Gehaltssumme je
Abteilung, Gesamtgehaltssumme

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10		8750
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
20		10875
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
30		9400
	ANALYST	6000
	CLERK	4150
	MANAGER	8275
	PRESIDENT	5000
	SALESMAN	5600
		29025

18 Zeilen ausgewählt.

Weitere Decision-Support Anfragen

- Rollup
- GROUPING
- GROUPING SETS
- ROW_NUMBER()
- RANK
- STOP AFTER
- DENSE_RANK
- WITH_BUCKET
- CROSS-JOIN
- NATURAL-JOIN
- EQUI-JOIN mit USING-Klausel
- JOIN-Prädikate bei der ON-Klausel

Rollup

```
select deptno, job, sum(sal)
from emp
group by rollup (deptno,job);
```

Bildet sortierte Zwischensummen auf jeder Ebene (zb.: Gehaltssumme je Abteilung und je Job) und zusätzlich die Summe je 1. Ebene ohne die 2. Ebene zu berücksichtigen (hier: Gehaltssumme je Abteilung) bis zur Gesamtsumme (zb.: Gesamtgehaltssumme).

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10		8750
20	CLERK	1900
20	ANALYST	6000
20	MANAGER	2975
20		10875
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
30		9400
		29025

13 Zeilen ausgewählt.