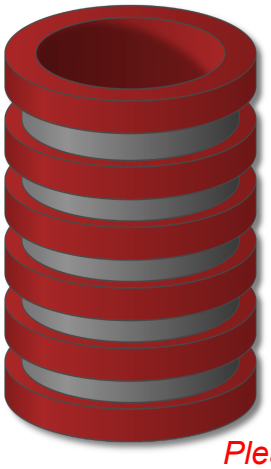
Announcements

- Due dates:
  - Group Project #2: Fri, Feb 21st
  - Individual HW #4: Mon, Feb 24th
- Additional XML slides
  - Lecture notes: [xml-sql.pdf]
  - … also linked from HW#4
- Today: Wrapping up OLAP
- Monday: President's Day
- Wednesday: Midterm



On-Line Analytical Processing (OLAP)

Demonstration

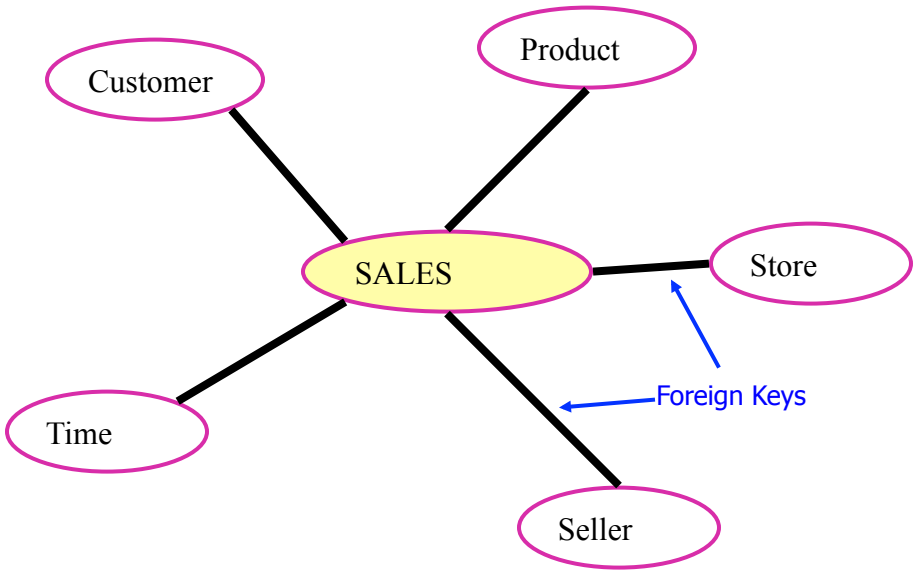
Please watch at db-class.org!

- *How many have done so already?*
- *What about exercises, quizzes, exams?*

More on Data Warehouses, and OLAP

Based on material from: Torsten Grust, Alfons Kemper

Star-Schema: Sales Data



Star-Schema

Sales					
Date	Store	Product	Count	Customer	SalesRep
25-Jul-00	Passau	1347	1	4711	825
...	...	...	...	...	...

- FACT tables (typically VERY LARGE, 1.000.000+ tuples)

Stores			
City	Country	State	...
Passau	D	Bayern	...
...	...	...	...

Customer			
CustId	Name	Age	...
4711	Kemper	43	...
...	...	...	...

Dimension tables (fairly small)

SalesRep					
PersonId	Name	Expertise	Manager	Age	...
825	Handyman	Electronics	119	23	...
...	...	...	...	...	...

Star-Schema (cont'd)

Time							
Date	Day	Month	Year	Quarter	Week	Weekday	Season
25-Jul-00	25	7	2000	3	30	Tuesday	Summer
...	...	...	...	...	...		
18-Dec-01	18	12	2001	4	52	Tuesday	Christmas
...	...	...	...	...	...	...	...

- Typical size: 1,000 (3 years)

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

- Typical size: 10,000 (catalog)

Non-normalized dimension tables: Hierarchical classification

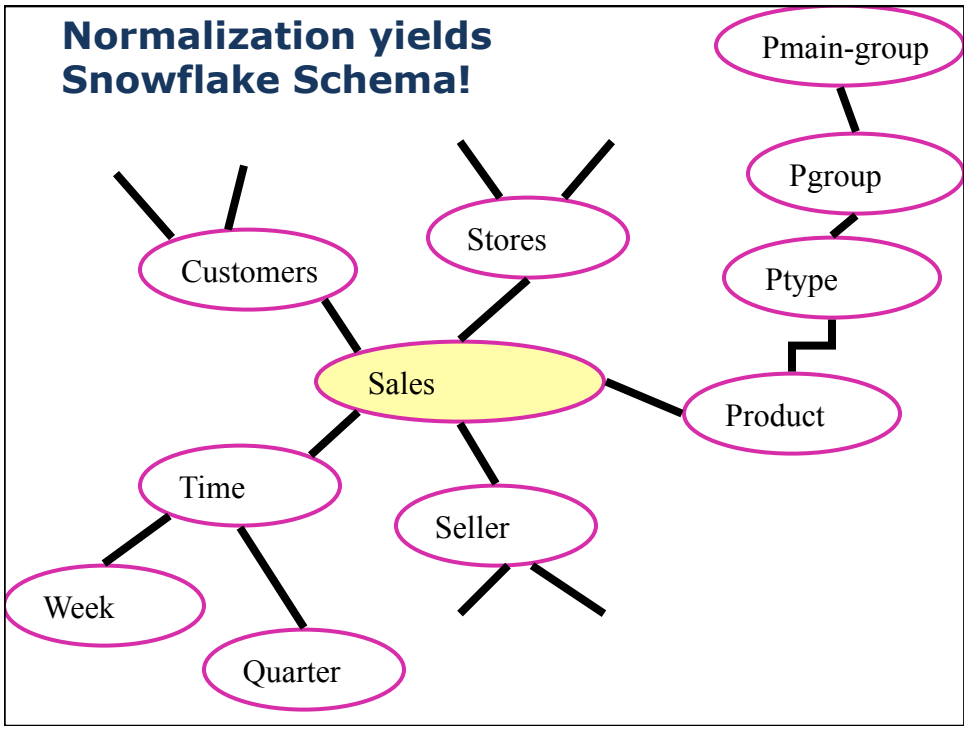
Time							
Date	Day	Month	Year	Quarter	Week	Weekday	Season
25-Jul-00	25	7	2000	3	30	Dienstag	Hochsommer
...	...	...	...	...	...		
18-Dec-01	18	12	2001	4	52	Dienstag	Weihnachten
...	...	...	...	...	...	...	...

FDs: Date → Month→ Quarter

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

ProduktNr → Produkttyp → Produktgruppe → Produkthauptgruppe

Normalization yields Snowflake Schema!



Typical Query over a Star Schema: Analysis

How many cell phones of which brand have young customers Ordered during Christmas 2011?

```
SELECT SUM(v.Anzahl), p.Hersteller
FROM   Verkäufe v, Filialen f, Produkte p, Zeit z, Kunden k
WHERE  z.Saison = 'Weihnachten' and
       z.Jahr = 2011 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 p.Hersteller;
```

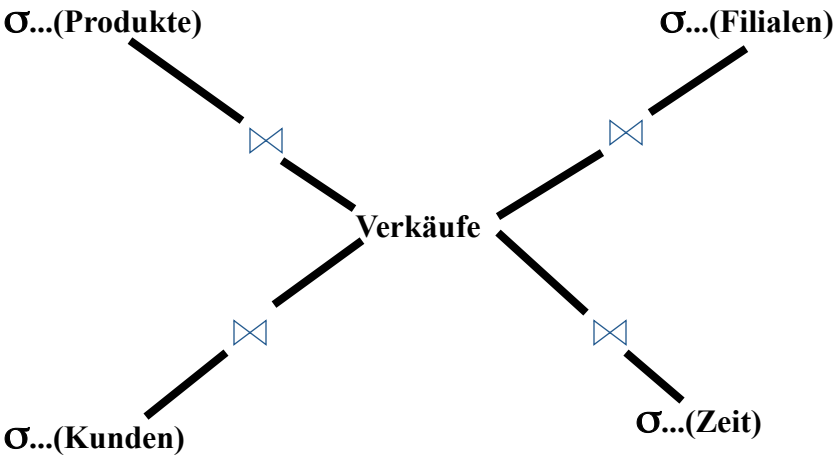
Aggregation

Filter on the dimensions

Grouping

(Star) Joins

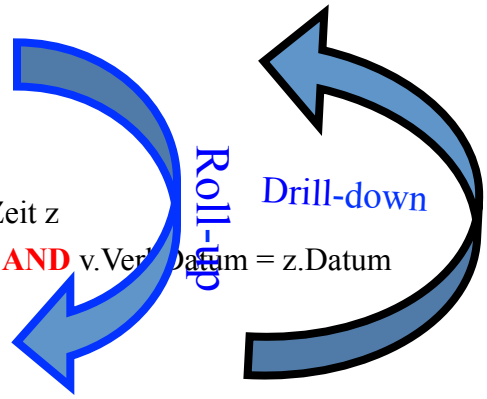
Algebra-Expression (*Star Join*)



Roll-up/Drill-Down

```
SELECT Jahr, Hersteller, SUM(v.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;

SELECT Jahr, SUM(v.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;
```



Analysis along different dimensions

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

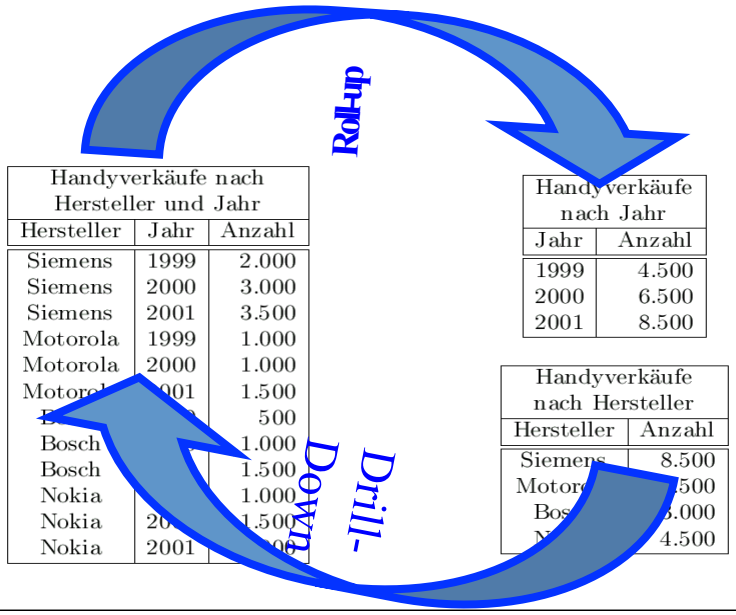
Handyverkäufe nach Jahr

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

Sales Analysis along different dimensions



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		500
Bosch		1.000
Bosch		1.500
Nokia		1.000
Nokia		1.500
Nokia	2001	

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

Data Cubes (*n*-dimensional)

In Decision-Support-Systems similar to spreadsheet-style (*cross tabulation*):

Hersteller \ Jahr	1999	2000	2001	Σ
Siemens	2.000	3.000	3.500	8.500
Motorola	1.000	1.000	1.500	3.500
Bosch	500	1.000	1.500	3.000
Nokia	1.000	1.500	2.000	4.500
Σ	4.500	6.500	8.500	19.500

This 2-dimensional ***data cube*** includes all results from the previous slide

Extreme case: Maximal Aggregation

No grouping (i.e., no **GROUP BY**-clause) ⇒ all rows form a single group, prior to aggregation:

```
SELECT SUM(Anzahl)
FROM   Verkäufe v, Produkte p
WHERE  v.Produkt = p.ProduktNr AND p.Produkttyp = 'Handy';
```

Materialization of Aggregates

```
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, 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,  NULL, SUM(v.Anzahl)
  FROM   Verkäufe v, Produkte p
  WHERE v.Produkt = p.ProduktNr  and  p.Produkttyp = 'Handy' );
```

Attributs = {Year, Maker}

⇒ subqueries for Group By Combinations

{ ∅, {Y,M}, {Y}, {M} }

Materialization of Aggregates

```
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, 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, NULL, SUM(v.Anzahl)
  FROM   Verkäufe v, Produkte p
  WHERE v.Produkt = p.ProduktNr  and  p.Produkttyp = 'Handy' );
```

Relational structure of the cube

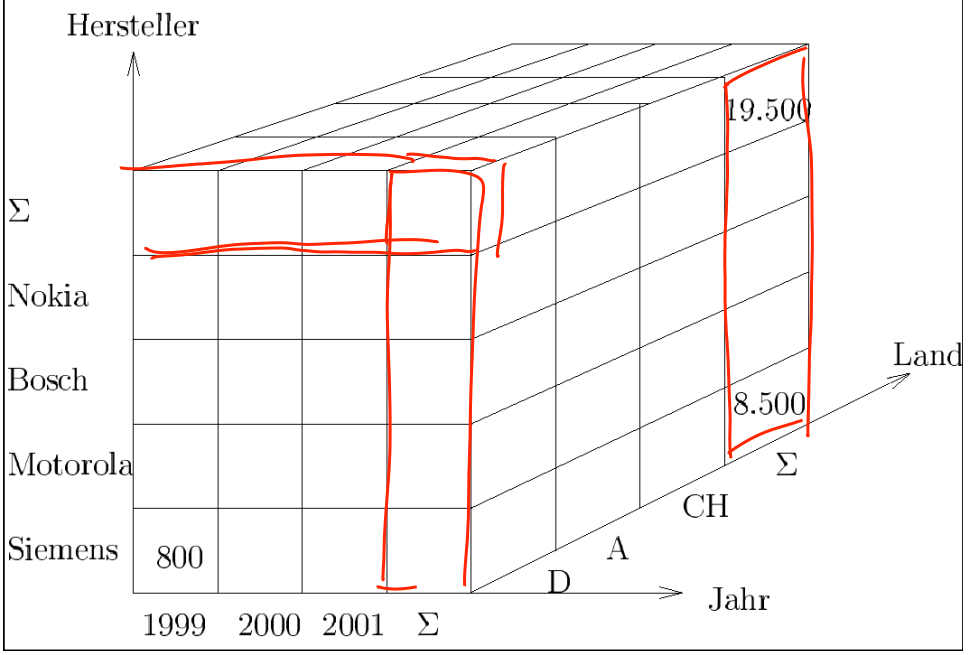
Handy2DCube

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

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

Cube representation



The CUBE-Operator (SQL:1999)

- Computing Aggregates leads to hard-to-optimize queries:
  - *n* Dimensions ⇒ 2^n subqueries, with UNION
  - But: aggregate can be computed hierarchically
- This CUBE-Operator replaces 2^3 subqueries:

```
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);
```

Year , Maker , Country

Reuse of Partial Aggregates

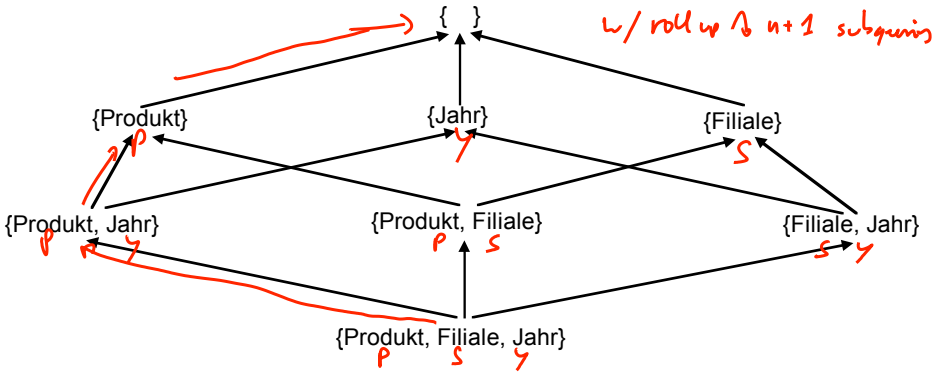
Assume the following aggregate has been materialized:

```
INSERT INTO VerkäufeProduktFilialeJahr
( SELECT    v.Produkt, v.Filiale, z.Jahr, SUM(v.Anzahl)
  FROM      Verkäufe v, Zeit z
  WHERE     v.VerkDatum = z.Datum
  GROUP BY  v.Produkt, v.Filiale, z.Jahr );
```

Then the following query can use the pre-computed aggregate (instead of the original fact table)

```
SELECT    v.Produkt, v.Filiale, SUM(v.Anzahl)
FROM      Verkäufe v  VerkäufeProduktFilialeJahr  v
GROUP BY  v.Produkt, v.Filiale
```

Materialisation-Hierarchy



- Subaggregates S can be **reused** for aggregation A if there is a path S → ... → A
- Lattice structure

WITH ROLLUP vs WITH CUBE

- ROLLUP (Year, Month, Day)
  - { Y, M, D }
  - { Y, M }
  - { Y }
  - { }
- CUBE (Year, Month, Day)
  - { Y, M, D }
  - { Y, M }
  - { Y, D }
  - { Y }
  - { M, D }
  - { M }
  - { D}
  - { }