Lecture 15

OLAP Operations & Queries

Summary – last week

Optimization

1. Joins
2. MaterializedViews

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

This week

OLAP Operations

1. OLAP Operations

```
(SELECT lead_in, ROW_NUMBER()
        OVER (PARTITION BY zip_code
              ORDER BY lead_date)
        AS lead_link
 FROM Leads) AS L
FULL OUTER JOIN
(SELECT dealer_id, ROW_NUMBER()
        OVER (PARTITION BY zip_code
              ORDER BY dealer_priority DESC)
        AS dealer_link
 FROM Dealers) AS D
ON D.dealer_link = L.lead_link AND D.zip_code = L.zipcode;
```

DW Queries

- DW queries are **big** queries
  - Imply a large portion of the data
  - Read only queries
    - no Updates
- Redundancy a necessity
  - MaterializedViews, special-purpose indexes, de-normalized schemas
- Data is refreshed periodically
  - E.g., Daily or weekly
- Their purpose is to analyze data
  - OLAP (OnLineAnalytical Processing)

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

DW Queries (cont'd.)

- OLAP usage fields
  - Management Information
    - Sales per product group / area / year
  - Government
    - Population census
  - Scientific databases
    - Geo-, Bio-Informatics
  - Etc.
- Goal: ResponseTime of seconds / few minutes

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

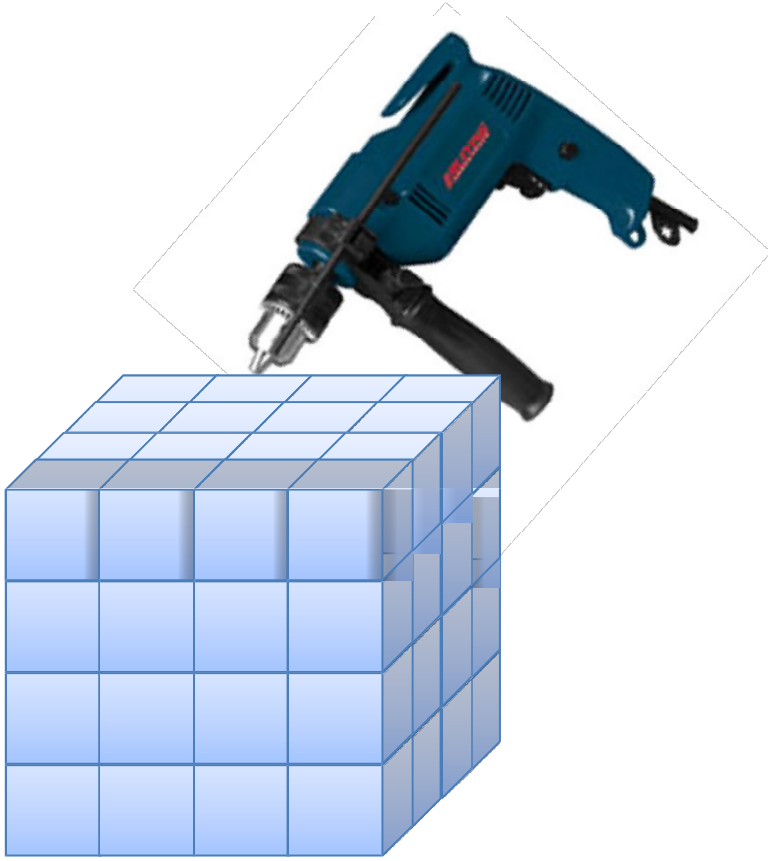
Typical Analytical Requests

- **Comparisons**
  - Show me the sales per region for **this year** and compare it to that of the **previous year** to identify discrepancies
- **Multidimensional ratios**
  - Show me the contribution to weekly profit made by **all items sold** in the **northeast stores between 1st of May and 7th of May**
- **Ranking and statistical profile**
  - Show me sales, profit and average call volume per day for my **10 most profitable** sales-people
- **Custom consolidation**
  - Show me the income statement **by quarter** for the last four quarters for my northeast region operations

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

OLAP Operations

- Typical OLAP operations
  - Roll-up
  - Drill-down
  - Slice and dice
  - Pivot (rotate)
- Other operations
  - Aggregate functions
  - Ranking and comparing
  - Drill-across
  - Drill-through



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

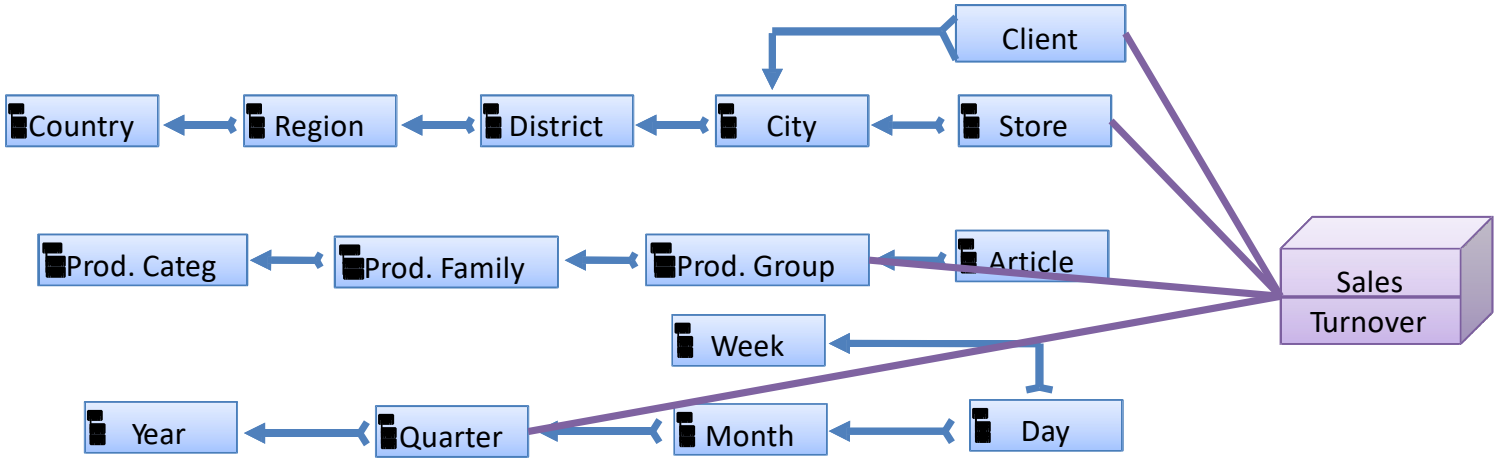
Roll-up

- **Roll-up (drill-up)**
  - Taking the current aggregation level of fact values and doing a **further aggregation**
  - **Summarize data** by
    - **Climbing up hierarchy** (hierarchical roll-up)
    - By **dimensional reduction**
    - Or by **a mix** of these 2 techniques
  - Used for obtaining an increased generalization
    - E.g., from Time.Week to Time.Year

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

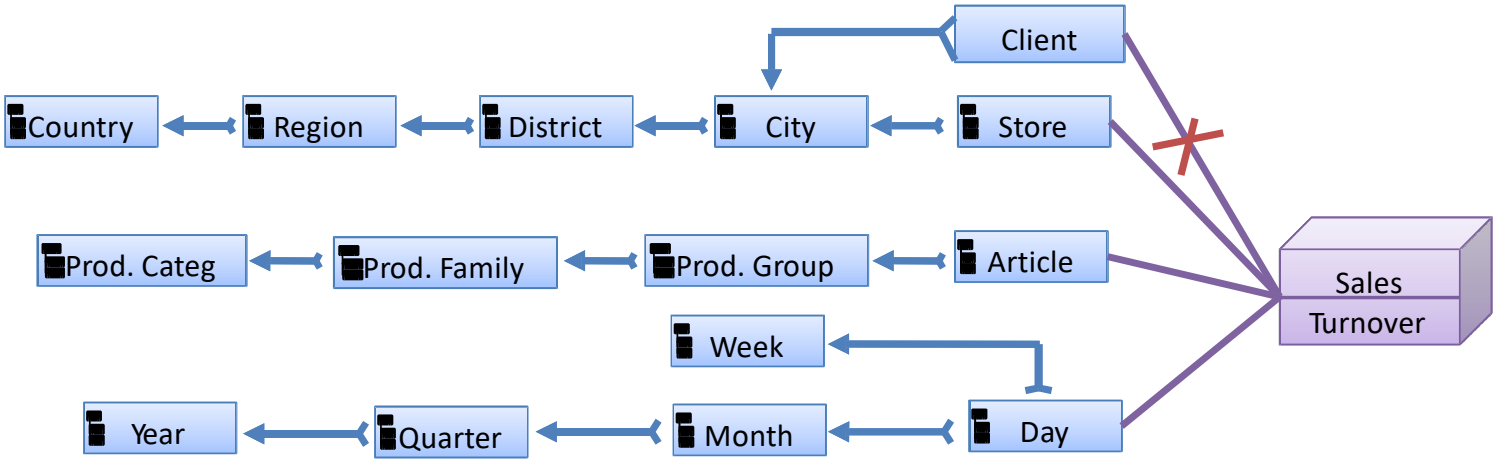
Roll-up (cont'd.)

- **Hierarchical roll-ups**
  - Performed on the **fact table and some dimension tables** by **climbing up** the attribute hierarchies
    - E.g., climbed the Time hierarchy to Quarter and Article hierarchy to Prod. group



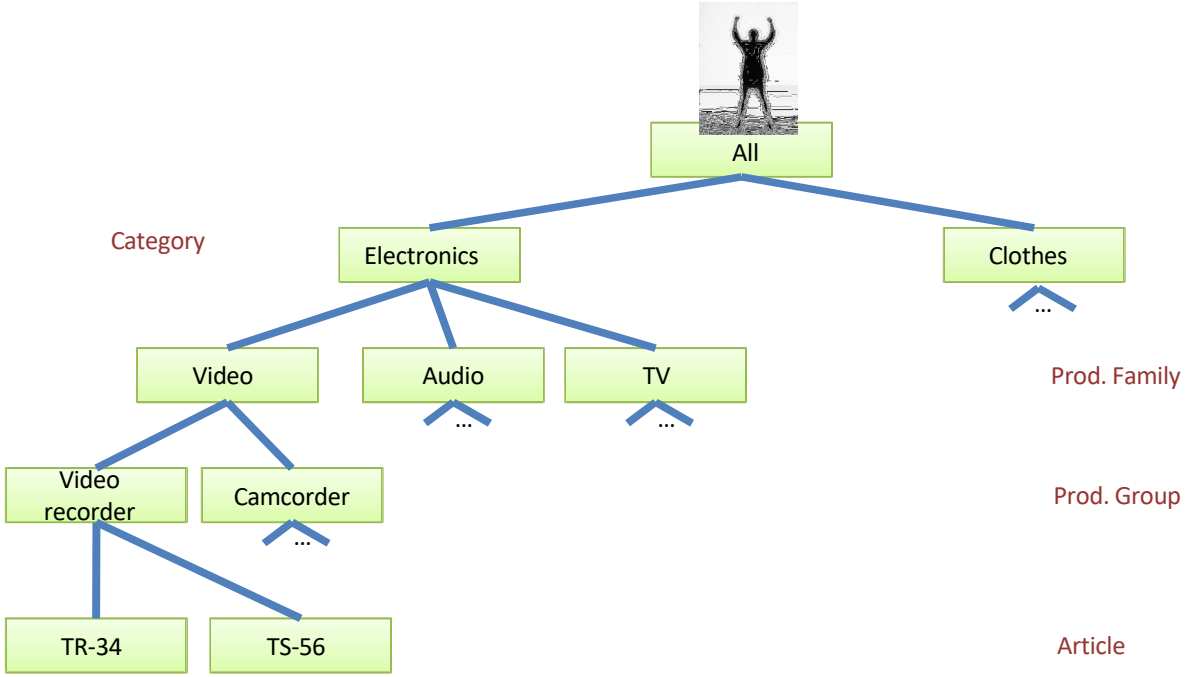
Roll-up (cont'd.)

- **Dimensional roll-ups**
  - Are done solely on the fact table by **dropping one or more dimensions**
    - E.g., drop the Client dimension



Roll-up (cont'd.)

- Climbing above the top in hierarchical roll-up
  - In an ultimate case, **hierarchical roll-up** above the top level of an attribute hierarchy (attribute "ALL") **can be viewed** as converting to a **dimensional roll-up**



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Drill-down

- Drill-down (roll-down)
  - **Reverse of Roll-up**
  - Represents a **de-aggregate** operation
    - From higher level of summary to lower level of summary – detailed data
  - Introducing new dimensions
  - Requires the **existence** of materialized **finer grained data**
    - You can't drill if you don't have the data

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Roll-up drill-down example

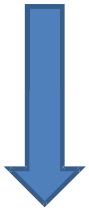
€ by bar/time

	Week1	Week2	Week3
Joe's	450	330	300
Salitos	500	360	420
Roots	380	310	400

Roll-up by BAR

€ by Time

Week1	Week2	Week3
1330	1000	1120



Drill-down by brand

€ by brand/Time

	Week 1	Week2	Week3
Wolters	480	400	400
Becks	450	310	370
Krombacher	400	290	350

Slice

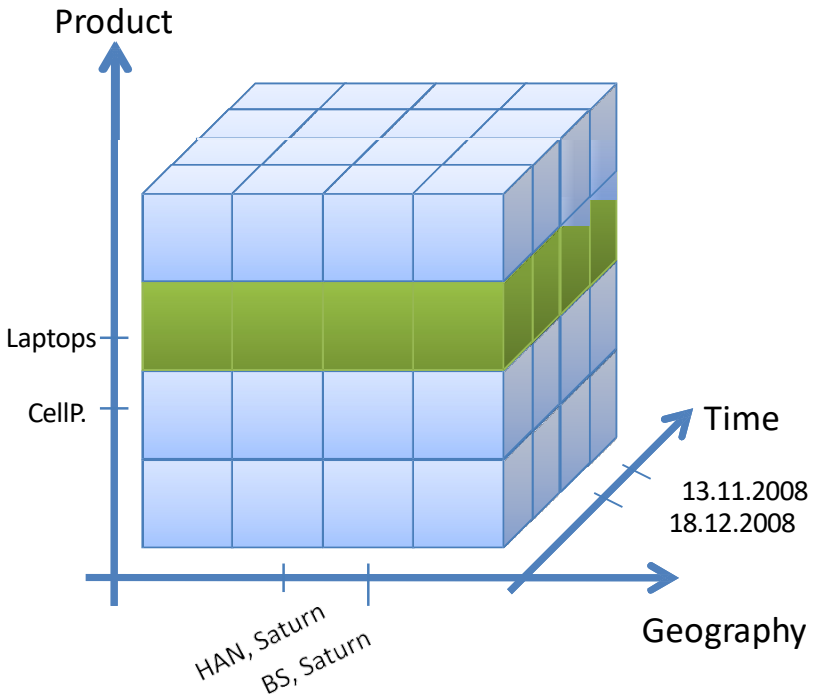
- Slice: a **subset** of the multi-dimensional array corresponding to **a single value** for one dimensions **and projecting on the rest** of dimensions
  - E.g., project on Geo (store) and Time from values corresponding to Laptops in the product dimension

$$\pi_{\text{StoreId, TimeId, Amount}} \left(\sigma_{\text{ArticleId = LaptopId}}(\text{Sales})\right)$$

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Slice (cont'd.)

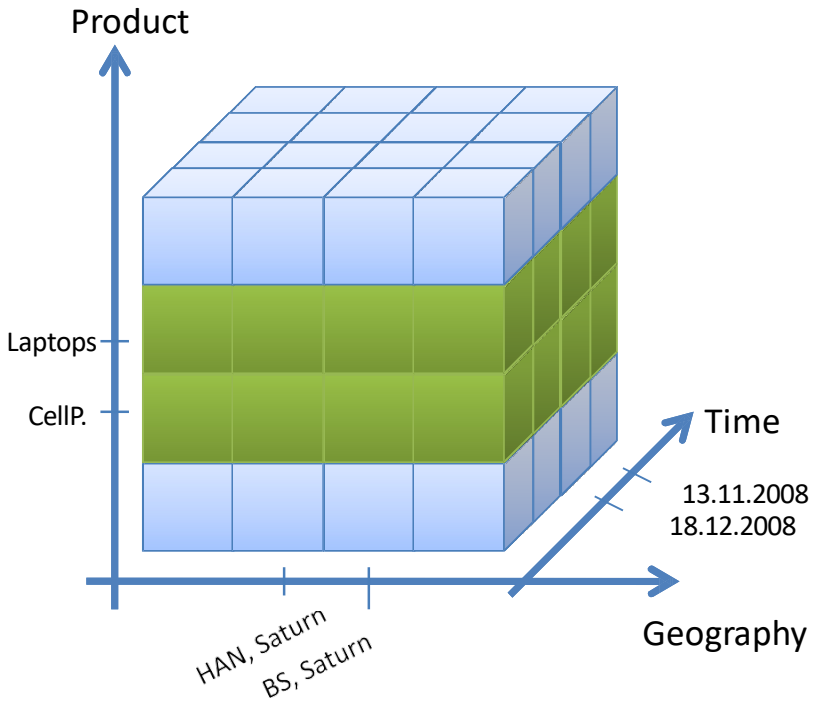
- Amounts to equality select condition
- **WHERE** clause in SQL
  - E.g., slice Laptops



Dice

- Dice: amounts to **range select condition** on one dimension, or to **equality select condition** on **more** than one dimension
  - E.g., Range SELECT

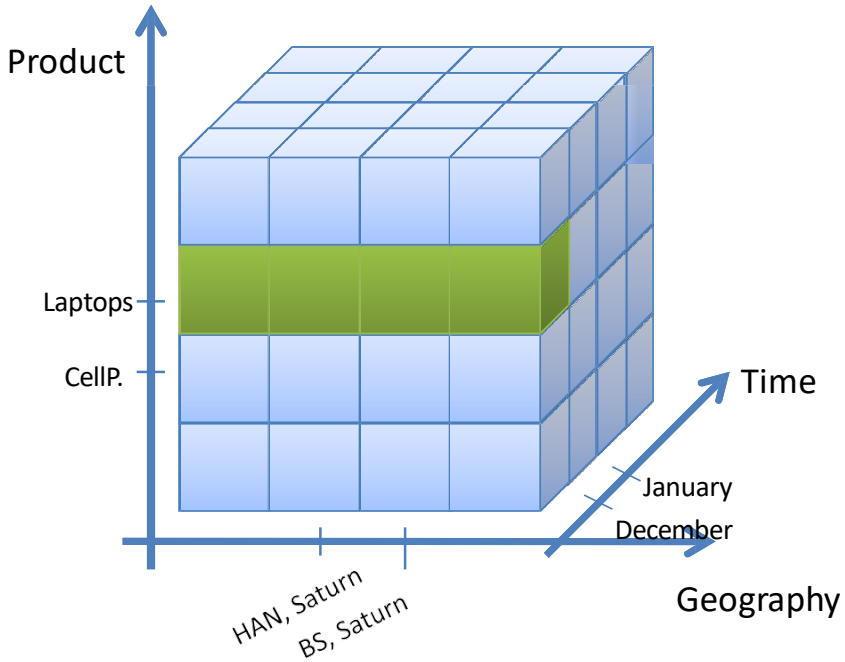
$$\pi_{\text{StoreId, TimeId, Amount}}(\sigma_{\text{ArticleId} \in \{\text{Laptop, CellP}\}}(\text{Sales}))$$



Dice (cont'd.)

- E.g., Equality SELECT on 2 dimensions Product and Time

$$\pi_{\text{StoreId, Amount}} (\sigma_{\text{ArticleId = Laptop} \wedge \text{MonthID = December}} (\text{Sales}))$$



Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Pivot

- Pivot (rotate): re-arranging data for viewing purposes
  - The simplest view of pivoting is that it selects two dimensions to **aggregate the measure**
    - The aggregated values are often displayed in a **grid** where each point in the (x, y) coordinate system corresponds to an aggregated value of the measure
    - The x and y coordinate values are the values of the selected two dimensions
  - The result of pivoting is also called **cross–tabulation**

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Pivot (cont'd.)

- Consider pivoting the following data

Location	
CityId	City
1	Well..
2	Nels..
3	Auck..

Sales			
CityId	PerId	TimId	Amnt
1	1	1	230
1	1	2	300
1	1	8	310
1	2	7	50
2	3	1	550
2	3	5	100
3	4	6	880
3	5	1	60
3	5	2	60
3	5	4	140

Time	
TimId	Day
1	Mon
2	Tue
3	Wed
4	Thu
5	Fri
6	Sat
7	San
8	Mon

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Pivot (cont'd.)

- Pivoting on City and Day

	Mon	Tue	Wed	Thu	Fri	Sat	San	SubTotal
Auckland	60	60	0	140	0	880	0	1140
Nelson	550	0	0	0	100	0	0	650
Wellington	540	300	0	0	0	0	50	890
SubTotal	1150	360	0	140	100	880	50	2680

	Auck	Nels	Well	SubTotal
Mon	60	550	540	1150
Tue	60	0	300	360
Wed	0	0	0	0
Thu	140	0	0	140
Fri	0	100	0	100
Sat	880	0	0	880
San	0	0	50	50
SubTotal	1140	650	890	2680

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Typical Analytical Requests

- OLAP operations are hard to express in query languages
  - Most analysts and decision makers won't enjoy it

```
SELECT  f.region, z.month, sum(a.price * a.volume)
FROM    Order a, Time z, PoS f
WHERE  a.pos = f.name AND a.date = z.date
GROUP BY f.region, z.month
```

  - OLAP clients allow operations to be performed through GUIs

QUARTER	Store Name:	PRODTYPE	Quantity:	Line Cost Of Goods Sold
Q2	Audio Expert	Digital	111,421	28,064,250.00
Q1	Audio Expert	Digital	105,983	25,092,678.00
Q1	eMart	Digital	108,221	24,990,368.00
Q2	eMart	Digital	115,102	24,971,512.00
Q1	eMart	Analog	97,128	21,152,262.00
Q2	eMart	Analog	74,737	16,789,403.00
Q1	Audio Expert	Analog	78,449	16,467,146.00
Q4	eMart	Digital	72,126	14,000,951.00
Q3	eMart	Digital	66,156	13,867,709.00
Q2	Audio Expert	Analog	57,944	11,868,758.00
Q3	Audio Expert	Digital	50,076	11,210,406.00
Q4	Audio Expert	Digital	53,275	11,190,923.00
Q1	TV City	Digital	41,307	10,128,967.00
Q4	eMart	Analog	39,515	9,383,389.00
Q3	eMart	Analog	36,306	8,308,647.00
Q2	TV City	Digital	29,627	6,732,303.00
Q2	AV VideoTown	Digital	27,377	5,928,507.00
Q4	Audio Expert	Analog	25,897	5,916,936.00

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

OLAP Query Languages

- Getting from OLAP operations to the data
  - As in the relational model, **through queries**
    - In OLTP we have SQL as the standard query language
  - However, OLAP operations are hard to express in SQL
  - There is **no standard** query language for OLAP
  - Choices are:
    - **SQL-99** for ROLAP
    - **MDX** (Multidimensional expressions) for both MOLAP and ROLAP – designed by Microsoft

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Typical Queries

- OLAP queries

  SELECT $d_1.x, d_2.y, d_3.z, \text{sum}(f.t_1), \text{avg}(f.t_2)$
  FROM Fact f, Dim1 d_1, Dim2 d_2, Dim3 d_3
  WHERE a < d_1.field < b AND d_2.field = c
  GROUP BY $d_1.x, d_2.y, d_3.z$;

- The idea is to
  - Select by Attributes of Dimensions
    - E.g., region = "Europe"
  - Group by Attributes of Dimensions
    - E.g., region, month, quarter
  - Aggregate on measures
    - E.g., sum(price * volume)

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

OLAP Query Languages

- SQL-99
  - Prepare SQL for OLAP queries
  - New SQL commands
    - GROUPING SETS
    - ROLLUP
    - CUBE
  - New aggregate functions e.g. median and mod
  - Queries of type“top k”

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

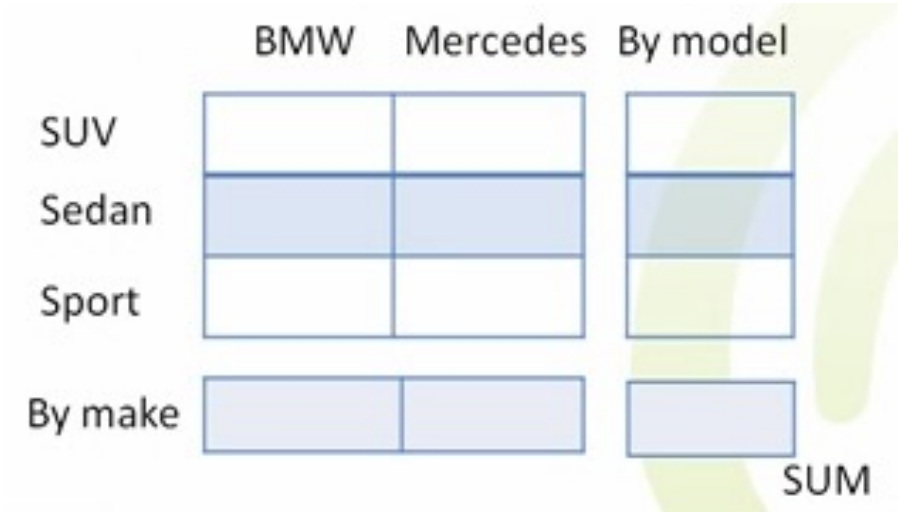
SQL-92

- Shortcomings of SQL/92 with regard to OLAP queries
  - Hard or impossible to express in SQL
    - Multiple aggregations
    - Comparisons (with aggregation)
    - Reporting features
  - Performance penalty
    - Poor execution of queries with many AND and OR conditions
  - Lack of support for statistical functions

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

SQL-92 (cont'd.)

- Multiple aggregations in SQL/92
  - Create a 2D spreadsheet that shows sum of sales by maker as well as car model
  - Each subtotal requires a separate aggregate query




```
SELECT model, make, sum(amt)
FROM sales GROUP BY model,
make
union
SELECT model, sum(amt) FROM
sales GROUP BY model
union
SELECT make, sum(amt) FROM
sales GROUP BY make
union
SELECT sum(amt) FROM sales
```

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

SQL-92 (cont'd.)

- Comparisons in SQL/92
  - This year's sales vs. last year's sales for each product
    - Requires a self-join

    - CREATE VIEW v_sales AS SELECT prod_id, year, sum(qty) AS sale_sum FROM sales GROUP BY prod_id, year;
    - SELECT cur.prod_id, cur.year, cur.sale_sum, last.year, last.sale_sum FROM v_sales cur, v_sales last WHERE cur.year = (last.year+1) AND cur.prod_id = last.prod_id;

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

SQL-92 (cont'd.)

- Reporting features in SQL/92
  - Too complex to express
    - RANK (top k) and NTILE ("top X%" of all products)
    - Median
    - Running total, moving average, cumulative totals
  - E.g., moving average over a 3 day window of total sales for each product
    - CREATE OR REPLACEVIEW v_sales AS SELECT prod_id, time_id, sum(qty) AS sale_sum FROM sales GROUP BY prod_id, time_id;
    - SELECT end.time, avg(start.sale_sum) FROM v_sales start, v_sales endWHERE end.time >= start.time AND end.time <= start.time + 2 GROUP BY end.time;

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

SQL-99

- Grouping operators
  - Extensions to the GROUP BY operator
    - GROUPING SET
    - ROLLUP
    - CUBE

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Grouping Operators

- GROUPING SET
  - **Efficiently** replaces the series of **UNIONed** queries
    - SELECT dept_name, COUNT(*) FROM personnel
      GROUP BY dept_name
      UNION ALL
      SELECT job_title, COUNT(*) FROM personnel
      GROUP BY job_title;

    - Can be re-written as:
      SELECT dept_name, job_title, COUNT(*) FROM Personnel
      GROUP BY GROUPING SET (dept_name, job_title);

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Roll-up

- Roll-up: produces a result set that contains subtotal rows in addition to regular grouped rows
  - GROUP BY ROLLUP (a, b, c) is equivalent to GROUP BY GROUPING SETS
  - (a, b, c), (a, b), (a), ()
  - N elements of the ROLLUP translate to (N+1) grouping sets
  - **Order** is significant to ROLLUP!
    - GROUP BY ROLLUP (c, b, a) is equivalent with grouping sets of (c, b, a), (c, b), (c), ()

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Roll-up (cont'd.)

- Roll-up operation, e.g.,:
  - SELECT year, brand, SUM(qty) FROM sales GROUP BY ROLLUP(year, brand);

Year	Brand	SUM(qty)	
2008	Mercedes	250	(year, brand)
2008	BMW	300	
2008	VW	450	
2008		1000	(year)
2009	Mercedes	50	(year, brand)
…	…	…	
2009		400	(year)
		1400	(ALL)

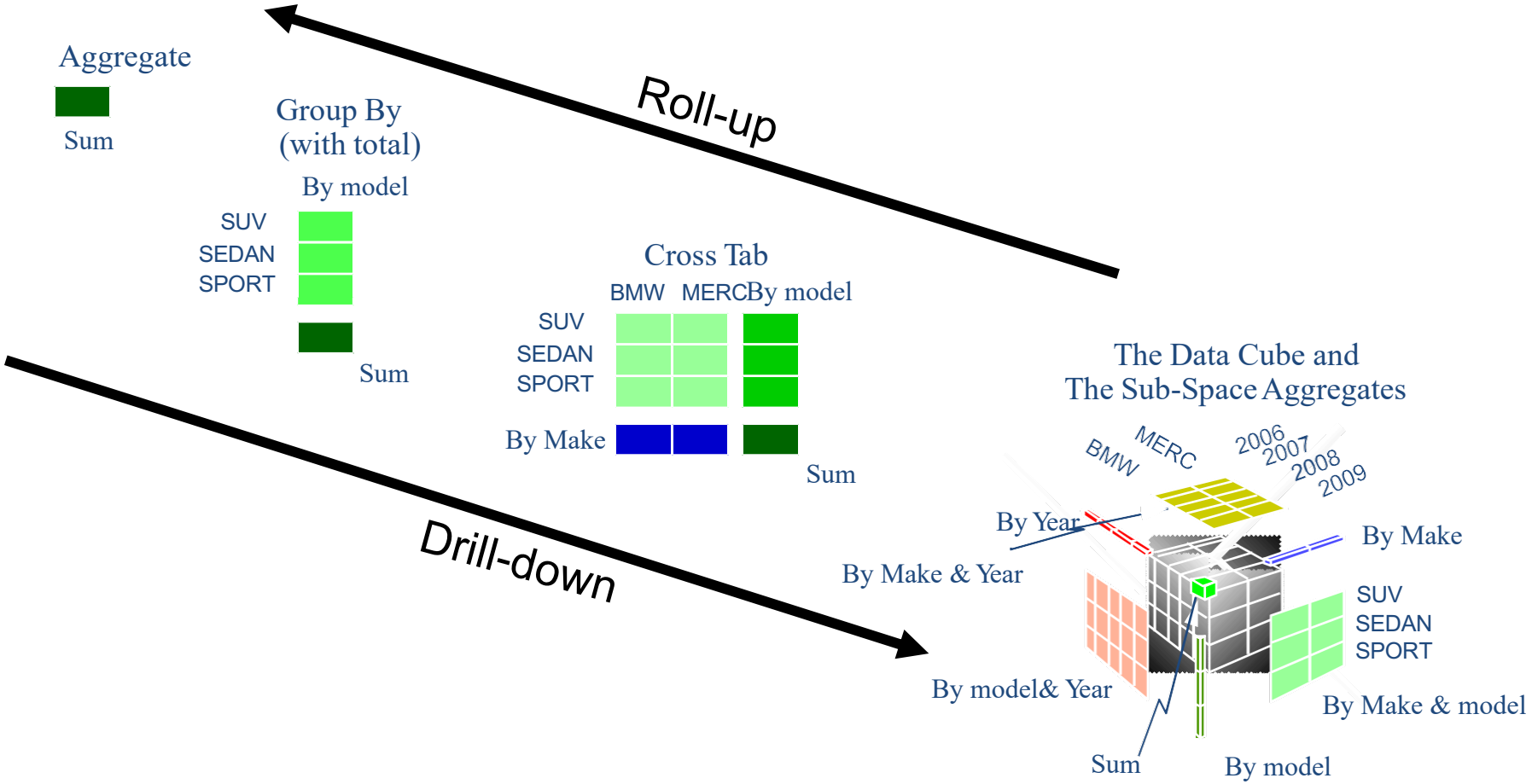
Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Grouping Operators (cont'd.)

- **Cube operator**: contains all the subtotal rows of a Roll-up and in addition **cross-tabulation rows**
  - Can also be thought as a series of GROUPING SETs
  - All permutations of the cubed grouping expressions are computed along with the grand total
    - N elements of a CUBE translate to 2^N grouping sets:
      - GROUP BY CUBE (a,b,c) is equivalent to
        GROUP BY GROUPING SETS(a,b,c) (a,b) (a,c) (b,c) (a) (b) (c) ()

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Cube Operators



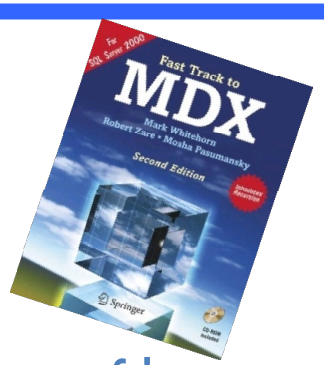
Cube Operators (cont'd.)

- E.g., CUBE operator
  - SELECT year, brand, SUM(qty) FROM sales GROUP BY CUBE (year, brand);

Year	Brand	SUM(qty)	
2008	Mercedes	250	(year, brand)
2008	BMW	300	
2008	VW	450	
2008		1000	(year)
2009	Mercedes	50	(year, brand)
…	…	…	
2009		400	(year)
	Mercedes	300	(brand)
	BMW	350	
	VW	750	
		1400	(ALL)

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

MDX



- MDX (MultiDimensional eXpressions)
  - Developed by Microsoft
    - Not really brilliant
    - But adopted by major OLAP providers due to Microsoft's market leader position
  - Used in
    - OLE (Object Linking and Embedding) DB for OLAP (ODBO) with API support
    - XML for Analysis (XMLA): specification of web services for OLAP
  - For ROLAP to support MDX, it is usually translated into SQL

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

MDX (cont'd.)

- Similar to SQL syntax

```
SELECT {Deutschland, Niedersachsen, Bayern, Frankfurt} ON COLUMNS,
        {Qtr1.CHILDREN, Qtr2, Qtr3} ON ROWS
        FROM SalesCube
        WHERE (Measures.Sales, Time.[2008], Products.[All Products]);
```

  - SELECT
    - axes dimensions, on columns and rows
  - FROM
    - Data source cube specification
    - If joined, data cubes must share dimensions
  - WHERE
    - Slicer - restricts the data area

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Summary

- OLAP Operations:
  - Roll-up: hierarchical, dimensional
  - Drill-down: You can't drill if you don't have the data
  - Slice, dice, Pivot

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides

Next Lecture

- Near-real-time Data Warehouse

Acknowledgment - Thanks to Wolf-Tilo Balke and Silviu Homoceanu - TU Braunschweig for the slides