
Lecture 09

OLAP Operations & Queries

Summary – last week

Summary

- Partitioning: Horizontal or Vertical
 - Records used together are grouped together
- Joins: for DW it is sometimes better to perform cross product on dimensions first
- Materialized Views: we can't materialize everything
 - Static or Dynamic choice of what to materialize
 - The benefit cost function is decisive

This week

OLAP Operations & Queries

1. OLAP Operations
2. OLAP Queries



```
(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)



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: Response Time of seconds / few minutes

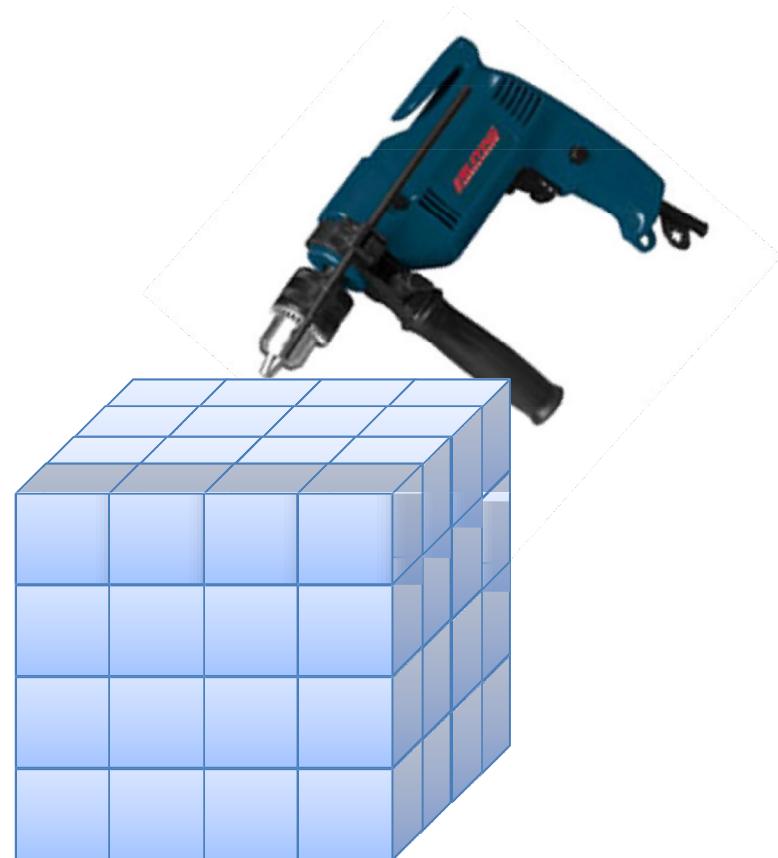


Typical Analytical Requests

- **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

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



Roll-up

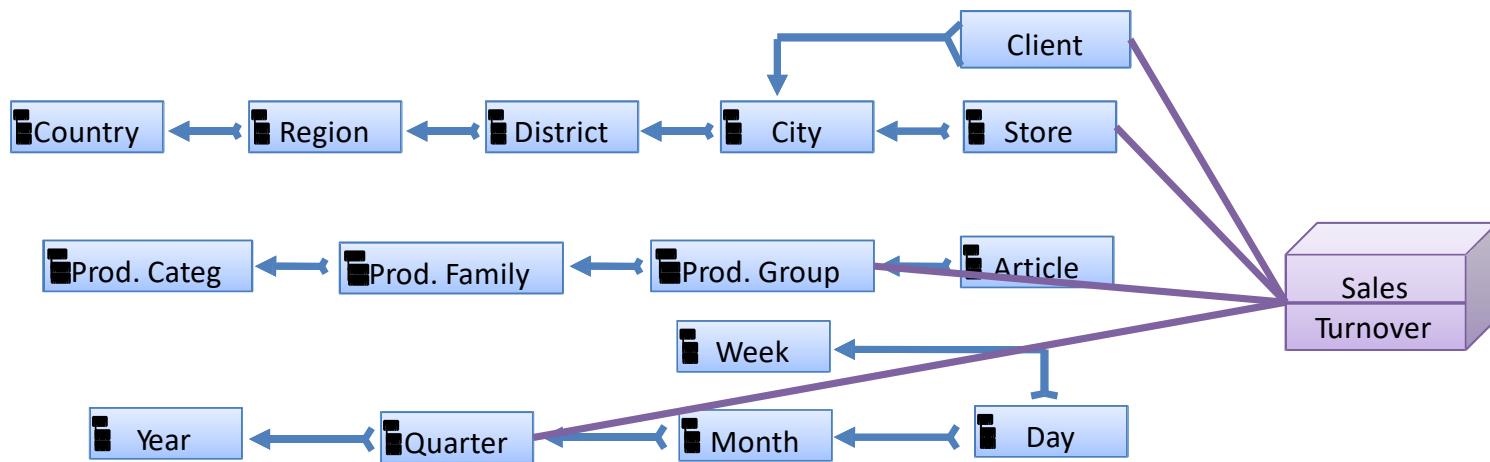
- **Roll-up (drill-up)**
 - Taking the current aggregation level of fact values and doing a **further aggregation**
 - **Summarize data by**
 - Climbing up hierarchy (hierarchical roll-up)
 - By dimensional reduction
 - Or by a mix of these 2 techniques
 - Used for obtaining an increased generalization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

Roll-up (cont'd.)

- **Hierarchical roll-ups**

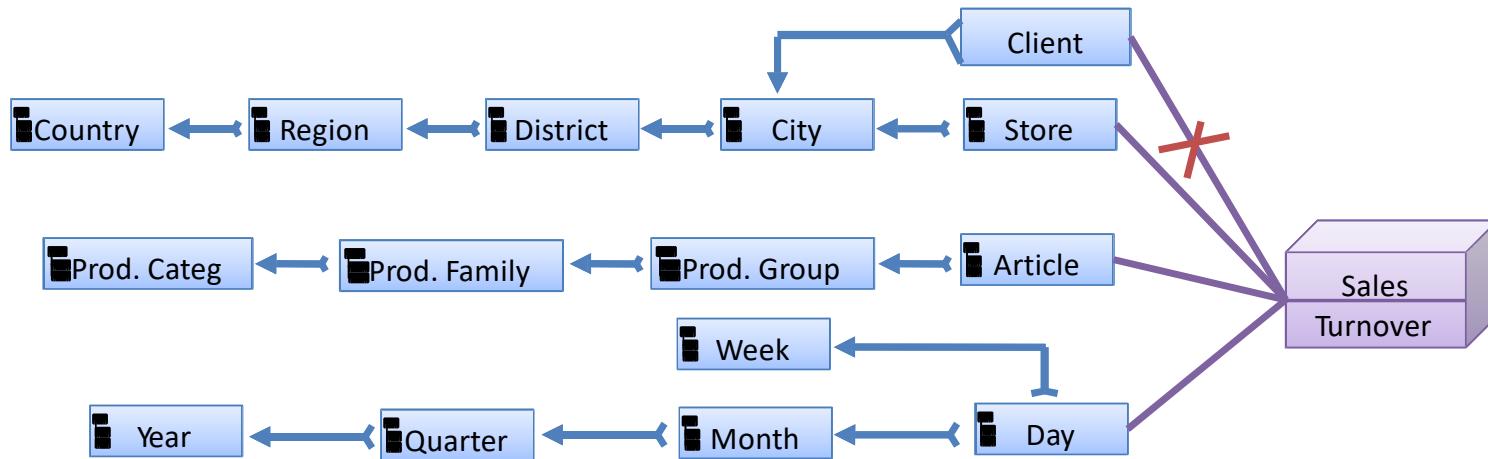
- Performed on the **fact table and some dimension tables** by **climbing up** the attribute hierarchies

- E.g., climbed the Time hierarchy to Quarter and Article hierarchy to Prod.group



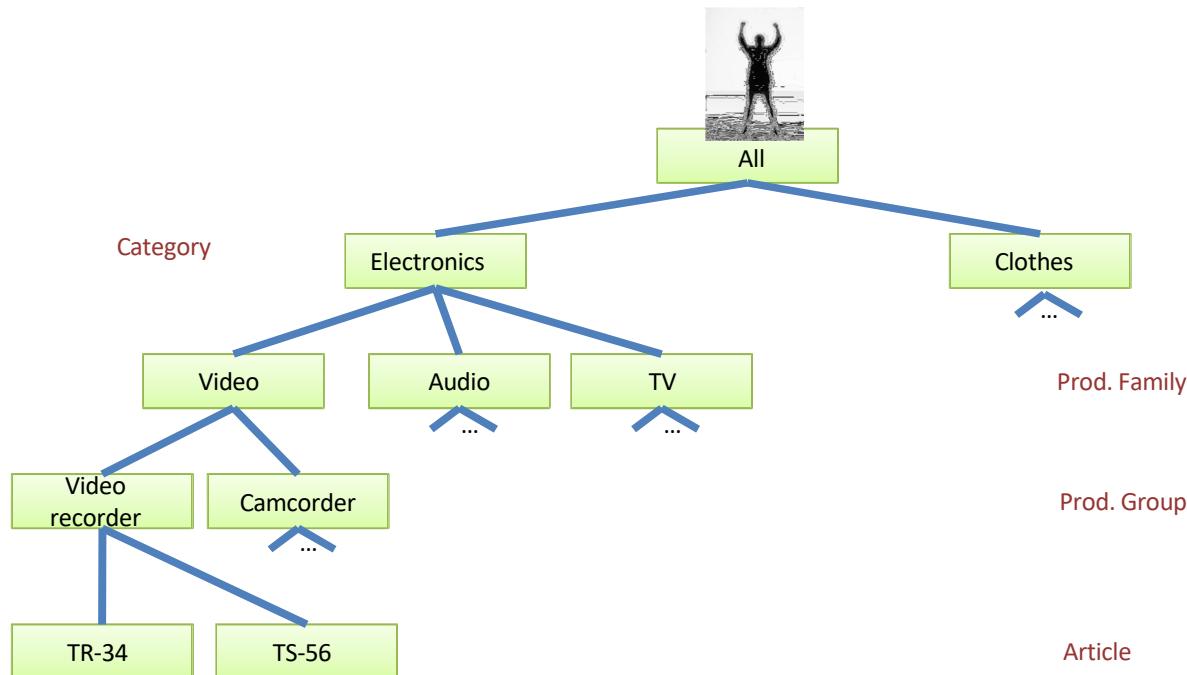
Roll-up (cont'd.)

- **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.)

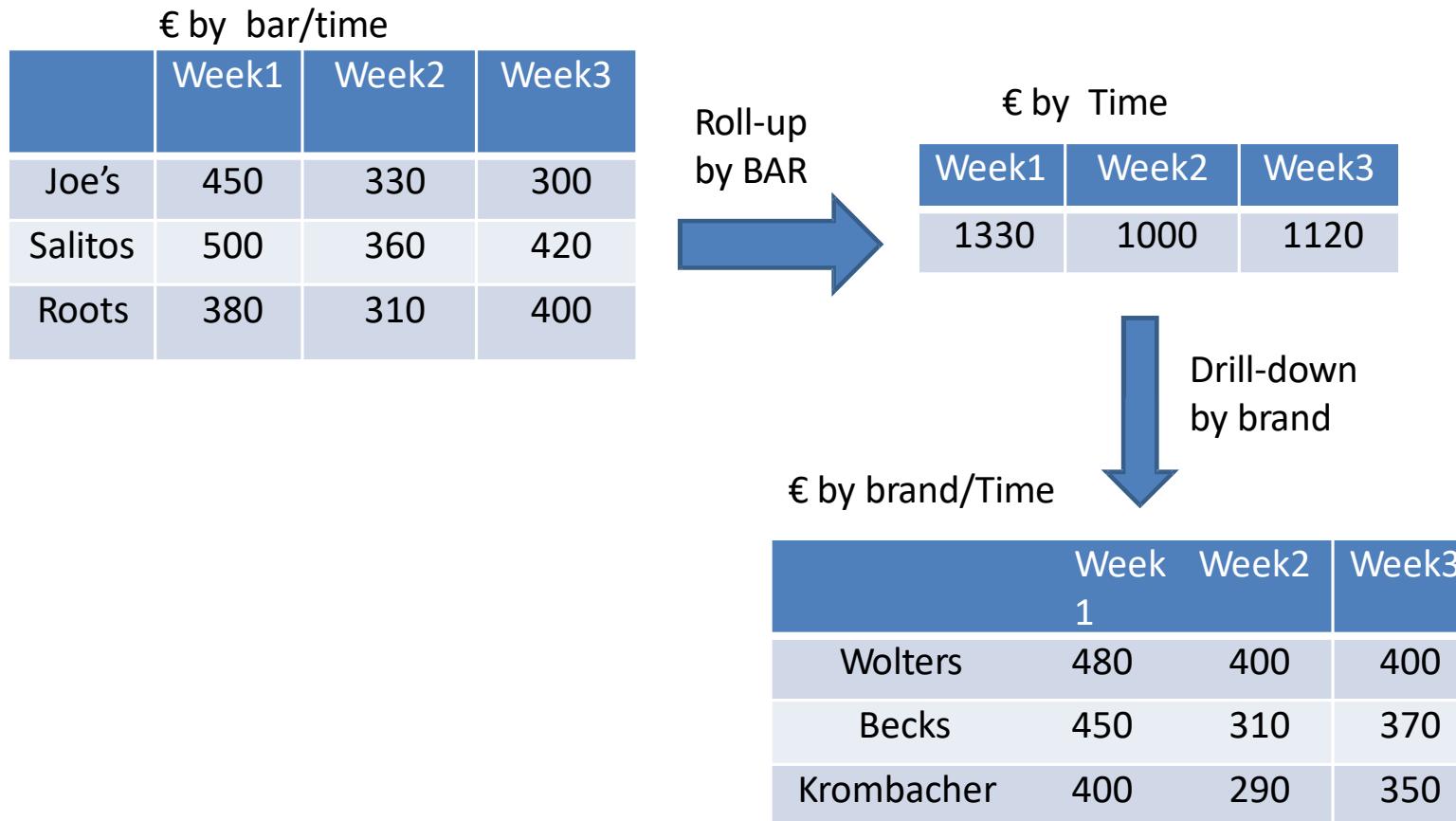
- Climbing above the top in hierarchical roll-up
 - In an ultimate case, **hierarchical roll-up** above the top level of an attribute hierarchy (attribute “ALL”) **can be viewed** as converting to a **dimensional roll-up**



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

Roll-up drill-down example



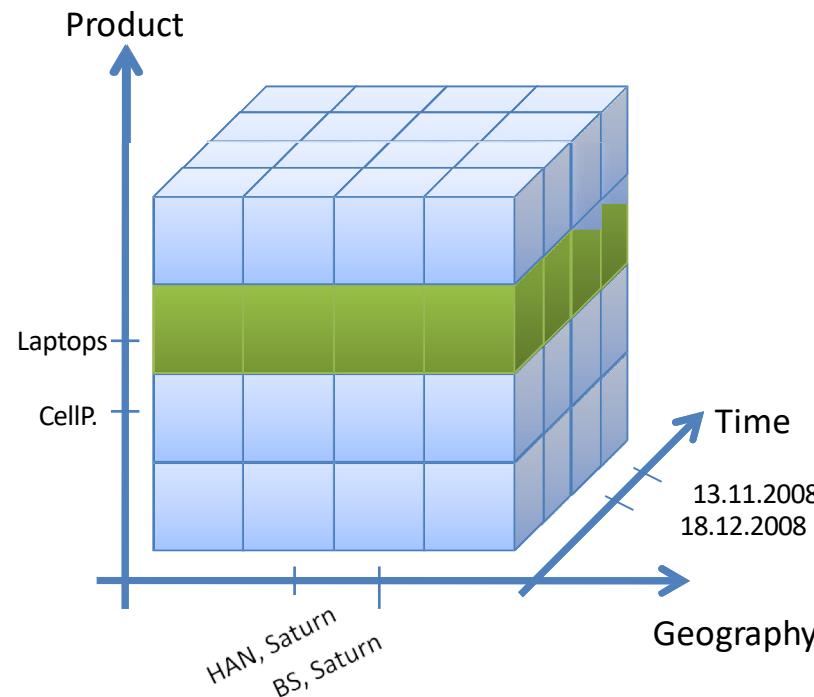
Slice

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

$$\pi_{\text{StoreId}, \text{Timeld}, \text{Ammount}} (\sigma_{\text{ArticleId} = \text{LaptopId}}(\text{Sales}))$$

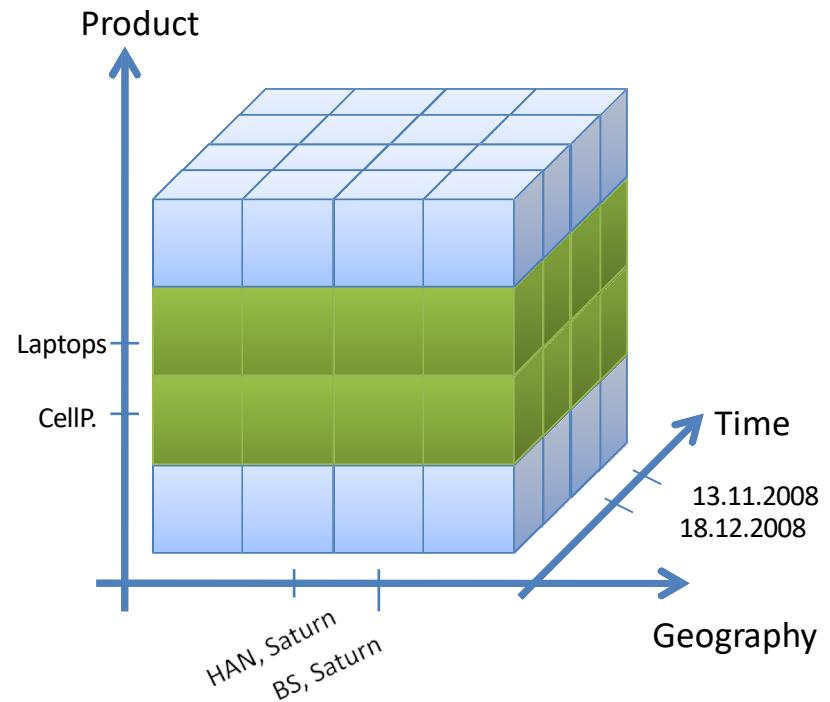
Slice (cont'd.)

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



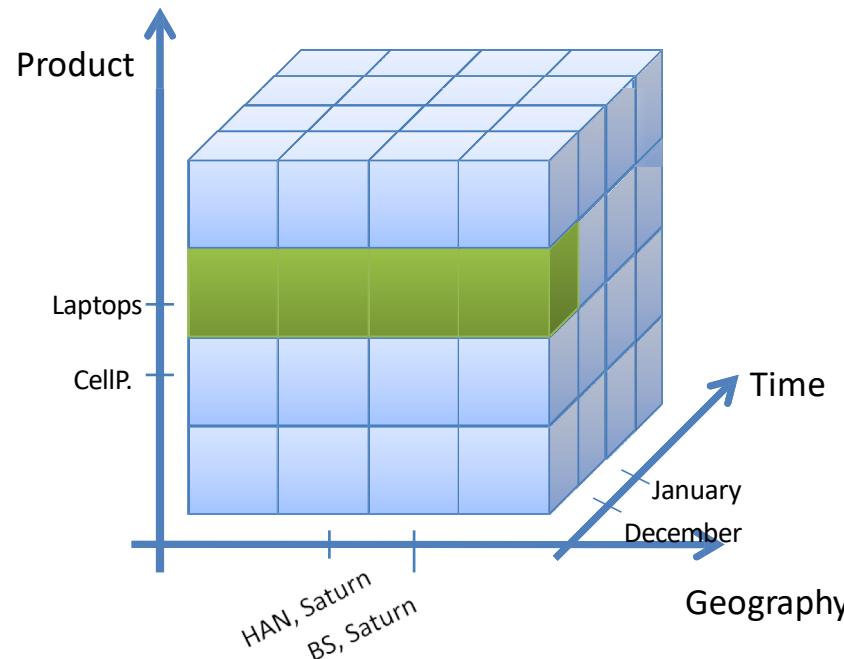
Dice

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

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


Dice (cont'd.)

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

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


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**

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

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

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
```

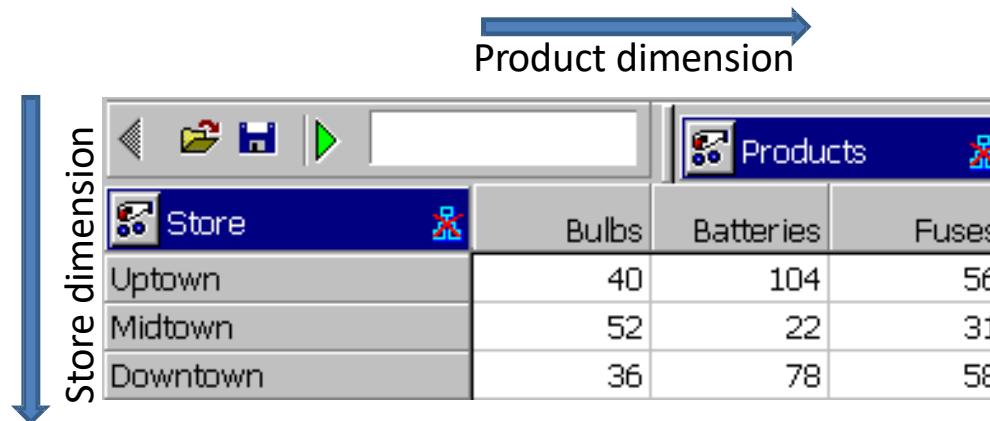


- OLAP clients allow operations to be performed through GUIs

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	

OLAP Data Visualization

- How do these operations look like for the user?
 - E.g., Crystal Decisions OLAP software
 - 2 dimensions ... is trivial
 - E.g., Products by Store



The screenshot shows a software interface for Crystal Decisions OLAP. At the top, there is a toolbar with icons for navigation and a search bar. To the right of the toolbar is a title bar labeled "Products". Below the title bar is a grid containing data. The first column is labeled "Store" and contains three rows: "Uptown", "Midtown", and "Downtown". The second column is labeled "Bulbs" and contains the values 40, 52, and 36 respectively. The third column is labeled "Batteries" and contains the values 104, 22, and 78 respectively. The fourth column is labeled "Fuses" and contains the values 56, 31, and 58 respectively. A blue arrow points from the left towards the "Store" column, labeled "Store dimension". A horizontal blue arrow points from the top towards the "Bulbs" column, labeled "Product dimension".

Store	Bulbs	Batteries	Fuses
Uptown	40	104	56
Midtown	52	22	31
Downtown	36	78	58



A SEAGATE COMPANY

OLAP Data Visualization (cont'd.)

- 3 dimensions
 - We can visualize sold quantity on 3 dimensions as layers

The diagram illustrates the visualization of sold quantity across three dimensions using OLAP cubes. Three arrows point from the text labels to the corresponding dimensions in the cube interface:

- A blue arrow labeled "Supplier dimension" points to the top row of the cube interface.
- A blue arrow labeled "Product dimension" points to the leftmost column of the cube interface.
- A blue arrow labeled "Store dimension" points to the rightmost column of the cube interface.

Products		Uptown	Midtown	Downtown	
Bulbs		40	52	36	0
Batteries		104	22	78	0
Fuses		56	31	58	



OLAP Data Visualization (cont'd.)

- More dimensions are difficult to represent
 - If we introduce Time dimension, a data cell could be represented by its 4 dimensions as follows:
 - Philips from Supplier dimension
 - Batteries from Products dimension
 - Uptown from Store dimension
 - And Monday from Time dimension

OLAP Data Visualization (cont'd.)

- OLAP products represent 3 and more dimensional data reducing it to a 2D layout
 - By picking values of the dimensions which can not be displayed
 - E.g., Display the number sold of **Products** by any of the **Stores** on **Monday**

Store	Bulbs	Batteries	Fuses
Uptown	40	104	56
Midtown	52	22	31
Downtown	36	78	58

OLAP Data Visualization (cont'd.)

- Another way is by **nesting** on the same axis

The screenshot shows a 3D cube visualization interface. The vertical axis (depth) is labeled 'Products' and contains three categories: 'Bulbs', 'Batteries', and 'Fuses'. The horizontal axis (width) is labeled 'Store' and has three categories: 'Uptown', 'Midtown', and 'Downtown'. The depth axis (height) is labeled 'Measure' and has two categories: 'Revenue' and 'Cost'. The data values are displayed in the cells of the cube. For example, in the Uptown store, Revenue for Bulbs is 250, Cost for Bulbs is 115, Revenue for Batteries is 416, Cost for Batteries is 255, and so on for Fuses and other stores.

		Products		
		Bulbs	Batteries	Fuses
Store	Measure	250	416	160
	Cost	115	255	95
Midtown	Revenue	325	88	140
	Cost	145	55	80
Downtown	Revenue	225	312	90
	Cost	155	180	75

Days		Monday

OLAP Data Visualization (cont'd.)

- OLAP reporting has to be very flexible
 - The IBM way of an OLAP web based report

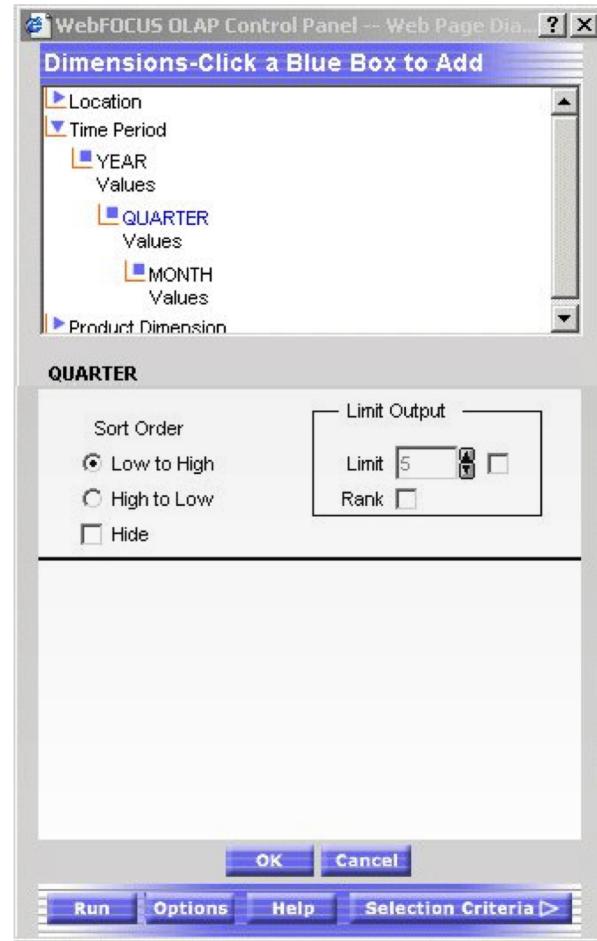
The screenshot shows the WebFOCUS OLAP Control Panel interface. At the top, there's a menu bar with File, Edit, View, Favorites, Tools, and Help. Below the menu is a toolbar with various icons. The address bar shows the URL http://localhost/cgi-bin/ibi_cgi/webapi.dll. The main area is divided into several sections:

- Dimensions:** A panel on the right containing three dropdown menus: Location, Time Period, and Product Dimension.
- Measures:** A panel at the bottom right containing checkboxes for QUARTER, Store Name, and PRODTYPE, all of which are checked.
- Drill Across:** A panel on the right with buttons for Shift Up, Remove, Shift Down, and Pivot.
- Drill Down:** A panel on the left with buttons for Shift Up, Remove, Shift Down, and Pivot.
- Data Grid:** The central area displays a grid of data. The columns are labeled QUARTER, Store Name, PRODTYPE, and Quantity. The rows show data for Q1, AV VideoTown, Audio Expert, City Video, Consumer Merchandise, and TV City across Analog and Digital product types.

QUARTER	Store Name:	PRODTYPE	Quantity:
Q1	AV VideoTown	Analog	18,449
		Digital	22,206
	Audio Expert	Analog	78,449
		Digital	105,983
	City Video	Analog	6,287
		Digital	7,196
	Consumer Merchandise	Analog	6,980
		Digital	14,957
	TV City	Analog	19,077

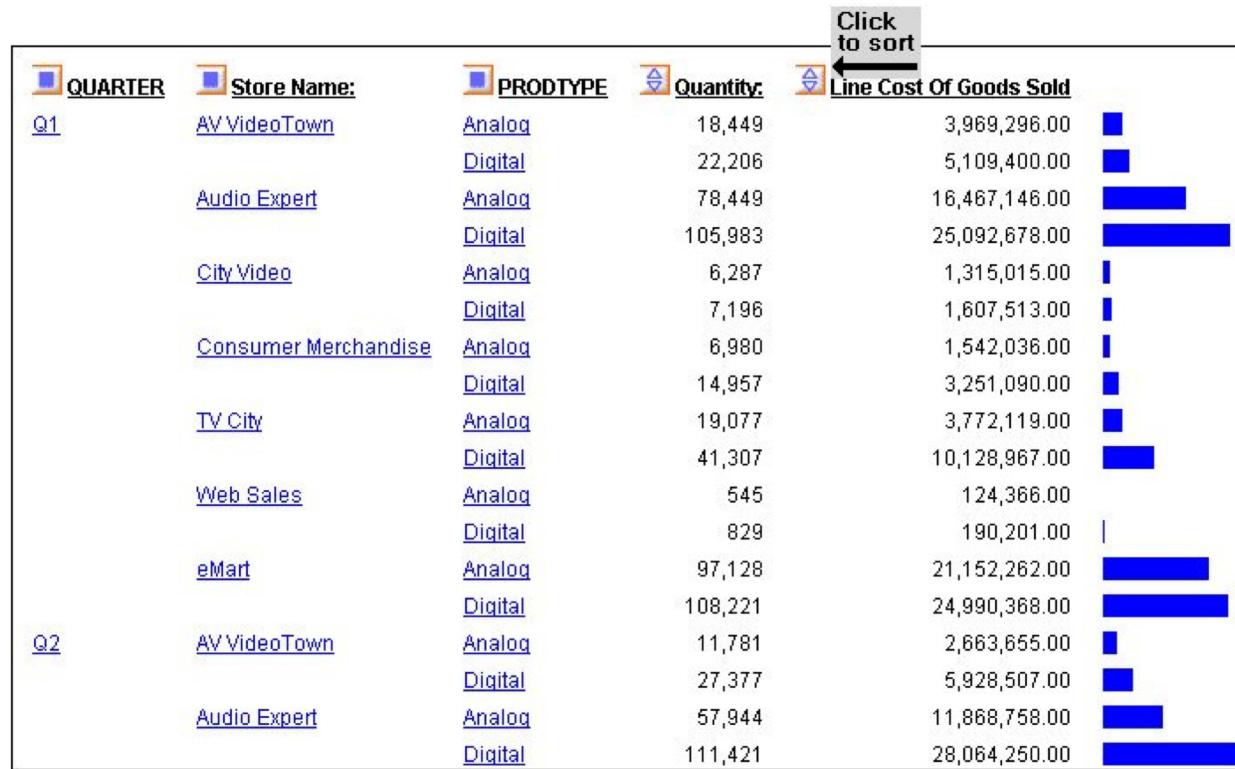
OLAP Data Visualization (cont'd.)

- **Drill-down** operation
 - Can be performed easy by going down on the hierarchy and choosing the granularity



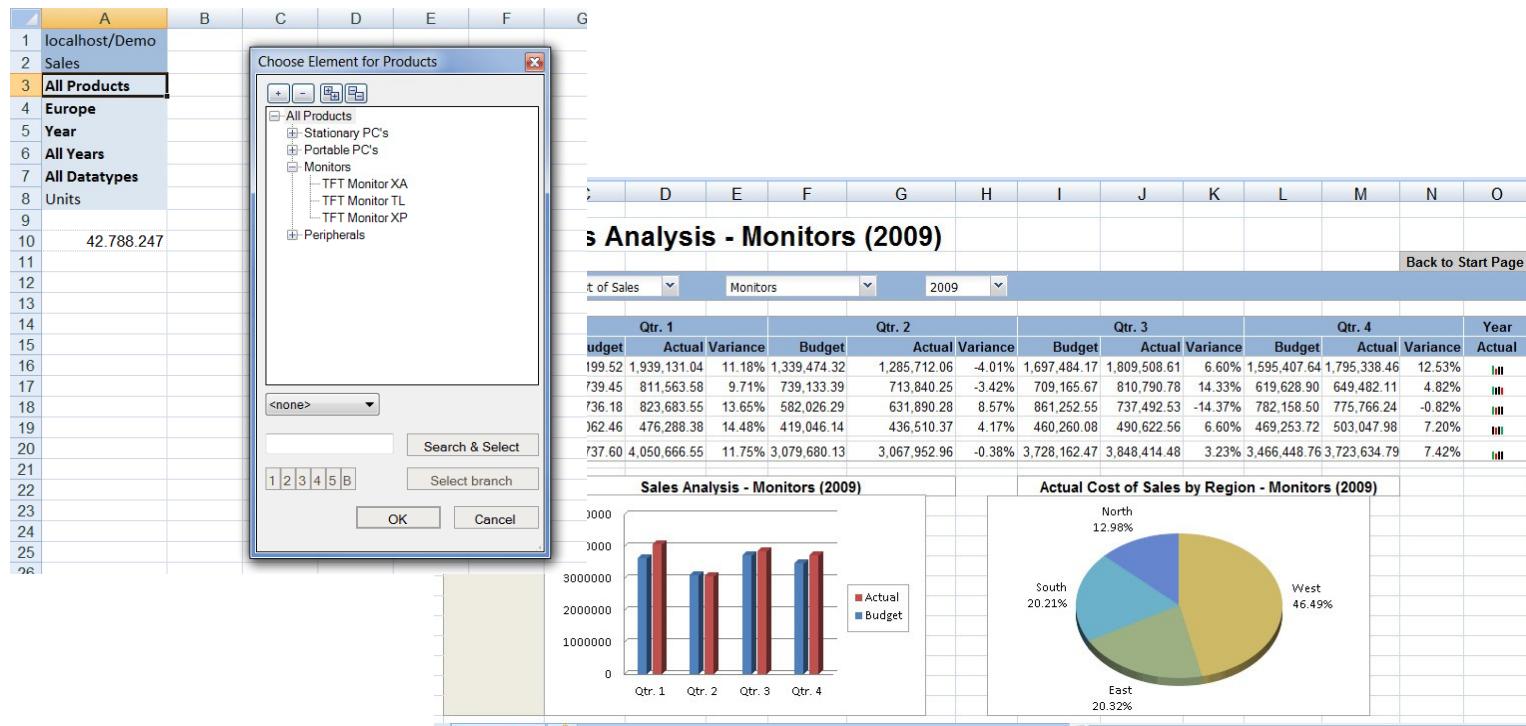
OLAP Data Visualization (cont'd.)

- Trends Visualization
 - With the help of charts

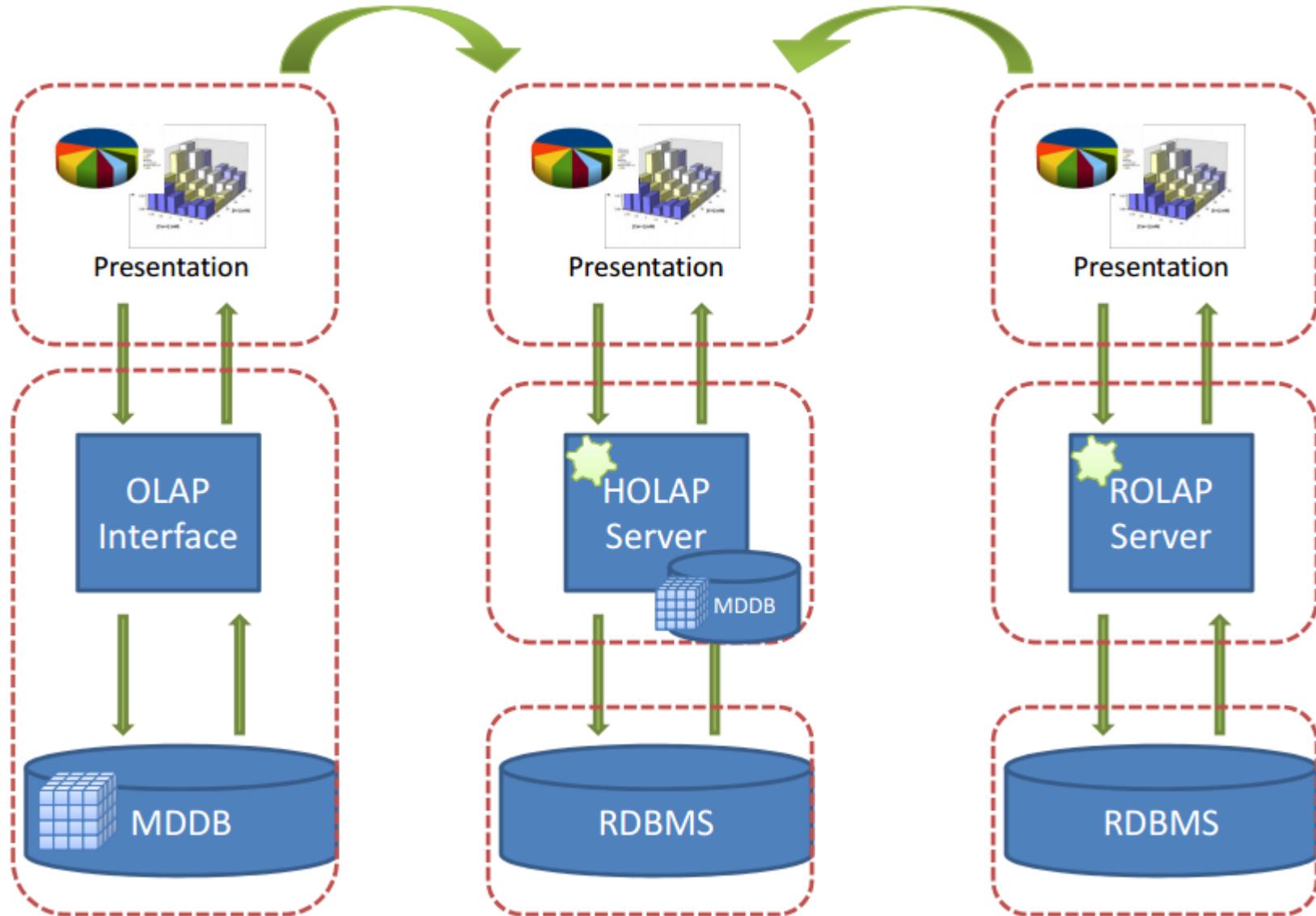


OLAP Data Visualization (cont'd.)

- Palo Technologies, integrated into Excel
 - Cubes are defined in a Web interface



How does OLAP work?



How does OLAP work?

- OLAP systems
 - Client/server architecture
 - The client displays reports and allows interaction with the end user to perform the **OLAP operations** and other custom queries – client can be fat or thin.
 - The server is responsible for providing the requested data. How? It depends on whether it is MOLAP, ROLAP, HOLAP, etc.

How does OLAP work?

- OLAP server
 - High-capacity, multi-user **data manipulation engine** specifically designed to **support and operate on multidimensional** data structures
 - It is optimized for
 - Fast, flexible calculation and transformation of raw data based on formulaic relationships



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

Typical Queries

- OLAP queries

```
SELECT d1.x,d2.y,d3.z,sum(f.t1),avg(f.t2)
  FROM Fact f,Dim1 d1,Dim2 d2,Dim3 d3
 WHERE a < d1.field < b AND d2.field = c
 GROUP BY d1.x,d2.y,d3.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)



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”

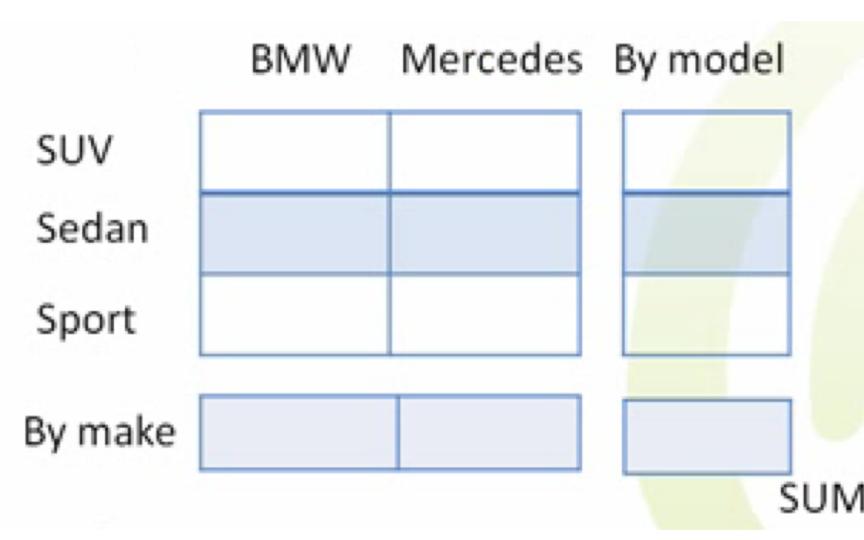


SQL-92

- 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

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
```

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;`

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 end WHERE end.time >= start.time AND end.time <= start.time + 2 GROUP BY end.time;`

SQL-99

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



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);`

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), ()

Roll-up (cont'd.)

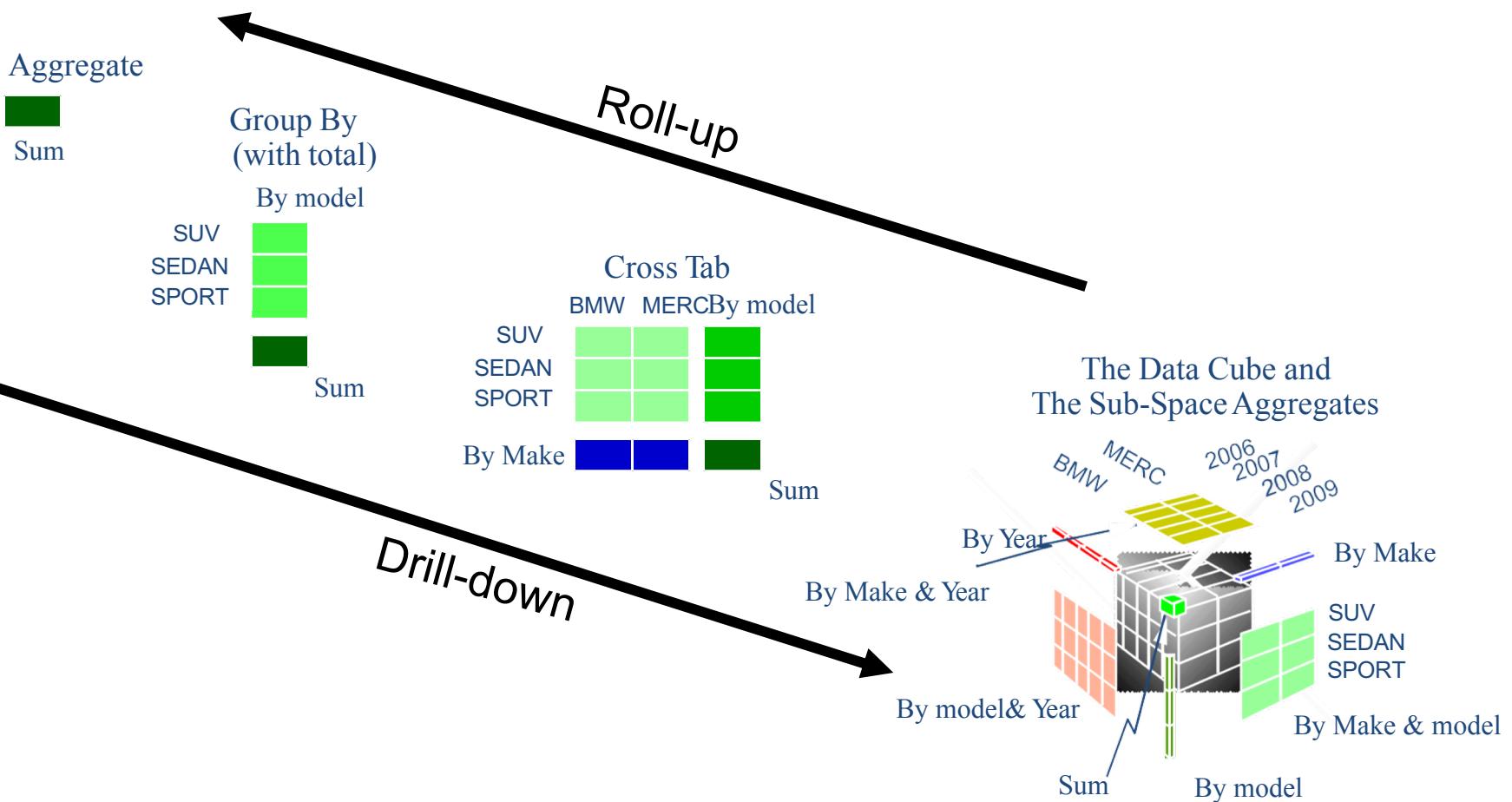
- Roll-up operation, e.g.:
 - `SELECT year,brand,SUM(qty) FROM sales GROUP BY ROLLUP(year, brand);`

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

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) ()

Cube Operators



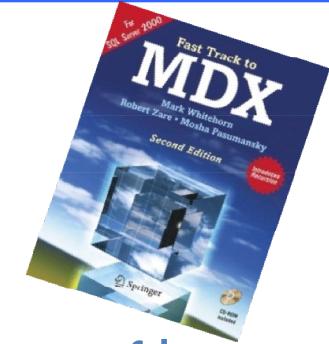
Cube Operators (cont'd.)

- 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	(year)
2008	VW	450	(year, brand)
2008		1000	(year)
2009	Mercedes	50	(year, brand)
...	(year)
2009		400	(brand)
	Mercedes	300	(ALL)
	BMW	350	
	VW	750	
		1400	

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



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

Summary

Summary

- OLAP Operations:
 - Roll-up:hierarchical, dimensional
 - Drill-down: You can't drill if you don't have the data
 - Slice,dice,Pivot
- Operations affect data through query languages
OLAP Query languages:SQL 99, MDX
 - SQL99: Grouping Set, Roll-up, Cube operators
 - MDX: Similar to SQL, used especially MOLAP solutions,in ROLAP it is mapped to SQL

Next Lecture

- Building the DW

The DW Project

Data Extract/Transform/Load (ETL)

Metadata

