

Smart, Creative and Entrepreneurial



Universitas
Esa Unggul

Data Warehouse

Munawar, PhD

Session 12

OLAP Operations and Query



Agenda

- OLAP Operation
- Data Visualization
- OLAP Query
- Q/A ?

Universitas
Esa Unggul



OLAP Operation

Universitas
Esa Unggul

DW Queries

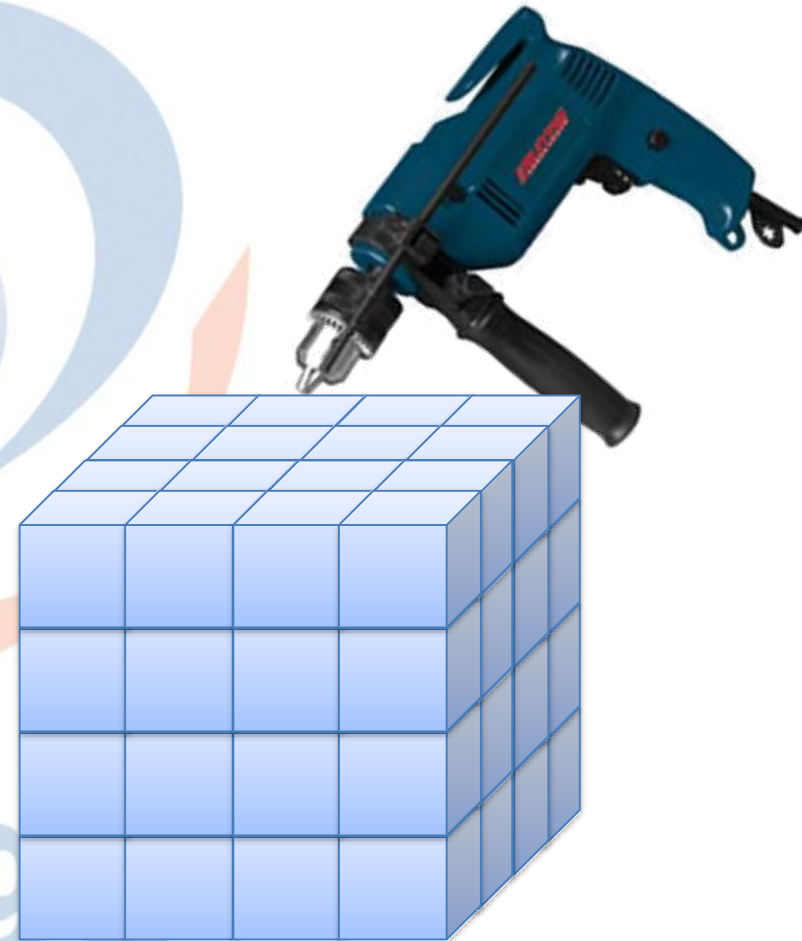
- DW queries are **big queries**
 - Imply a large portion of the data
 - Mostly read queries
- Redundancy a necessity
 - Materialized views, special-purpose indexes, de-normalized schemas
- Data is refreshed periodically
 - Daily or weekly
- Their purpose is to analyze data
 - OLAP (OnLine Analytical Processing)

DW Queries

- DW queries are **big queries**
 - Imply a large portion of the data
 - Mostly read queries
- Redundancy a necessity
 - Materialized views, special-purpose indexes, de-normalized schemas
- Data is refreshed periodically
 - Daily or weekly
- Their purpose is to analyze data
 - OLAP (OnLine Analytical Processing)

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



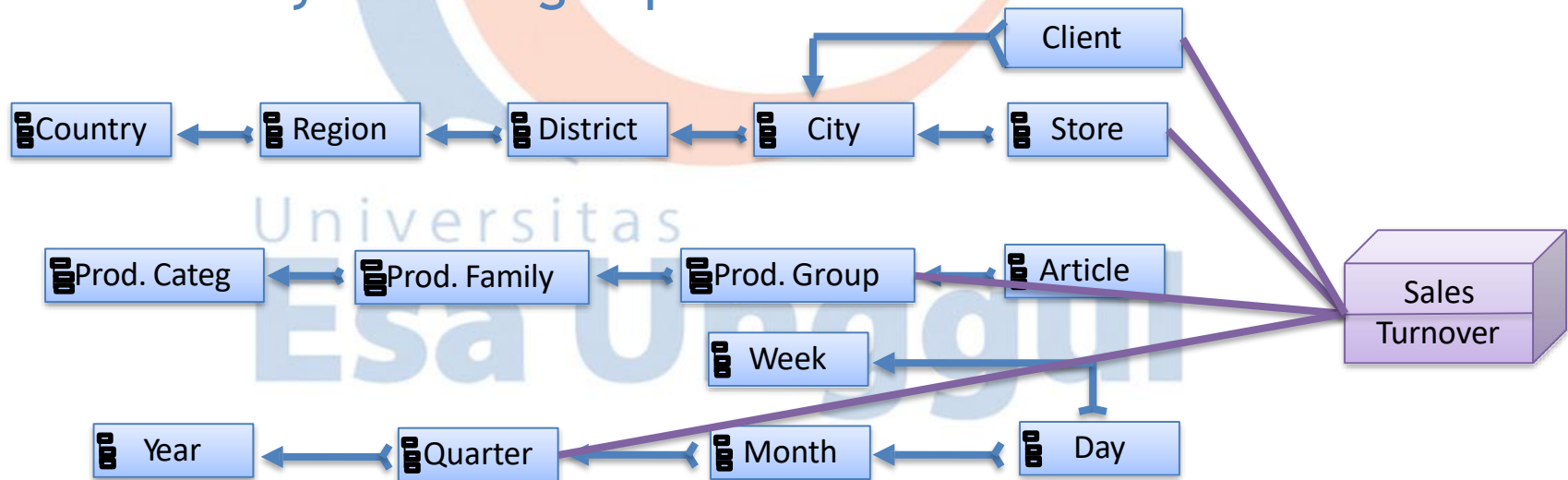
Roll-Up

- 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...

- **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

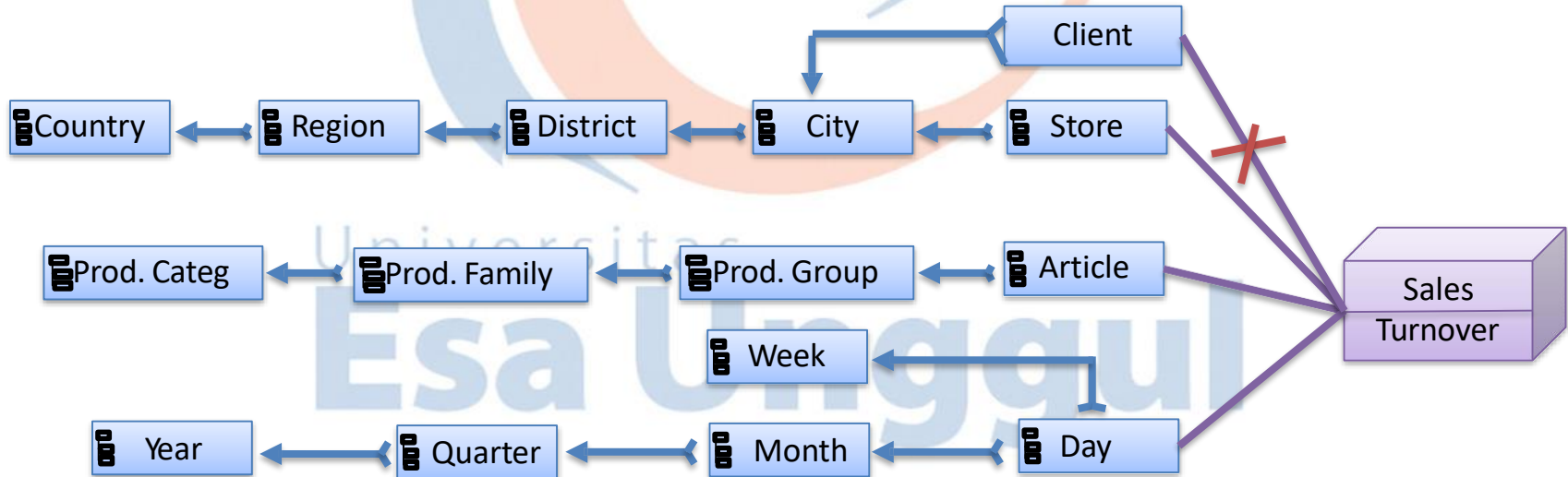


Roll-Up...

- **Dimensional roll-ups**

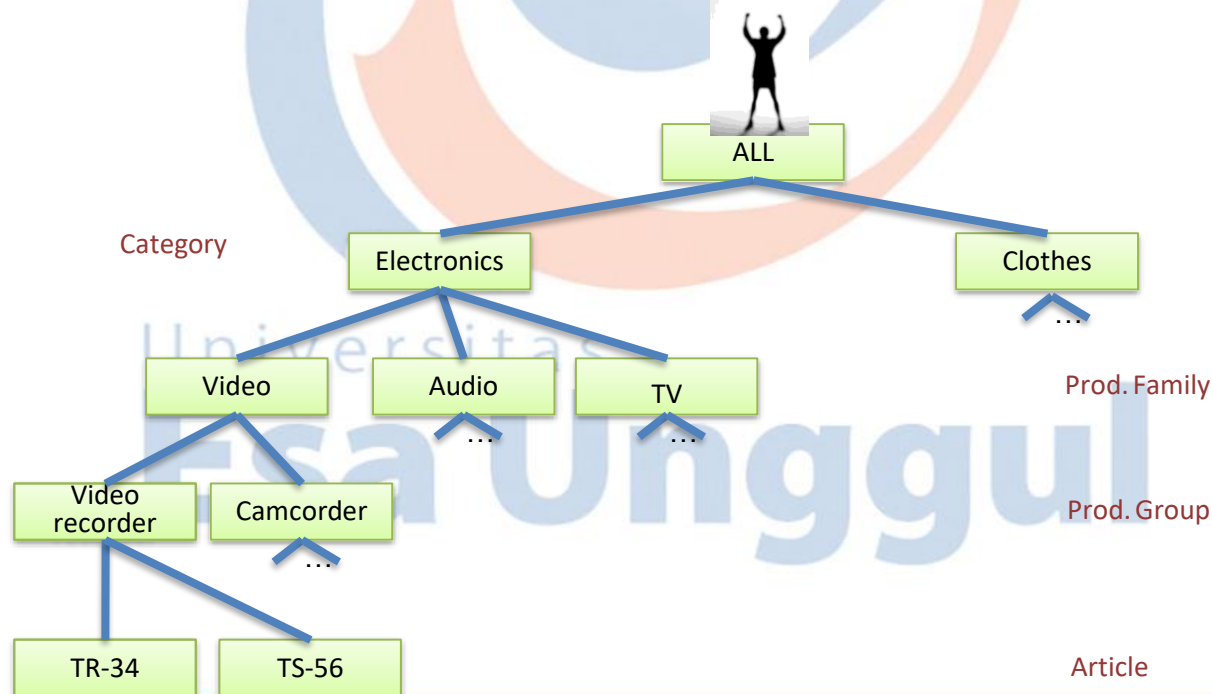
- Are done solely on the fact table by dropping one or more dimensions

- E.g., drop the Client dimension



Roll-Up...

- 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

\$ by BAR/Time

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

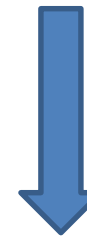
Roll-up
by BAR



\$ by Time

Week1	Week2	Week3
1330	1000	1120

Drill-down
by Brand



\$ by Brand/Time

	Week1	Week2	Week3
Wolters	480	400	400
Becks	450	310	370
Krombacher	400	290	350

Universitas
Esa Un

Slice

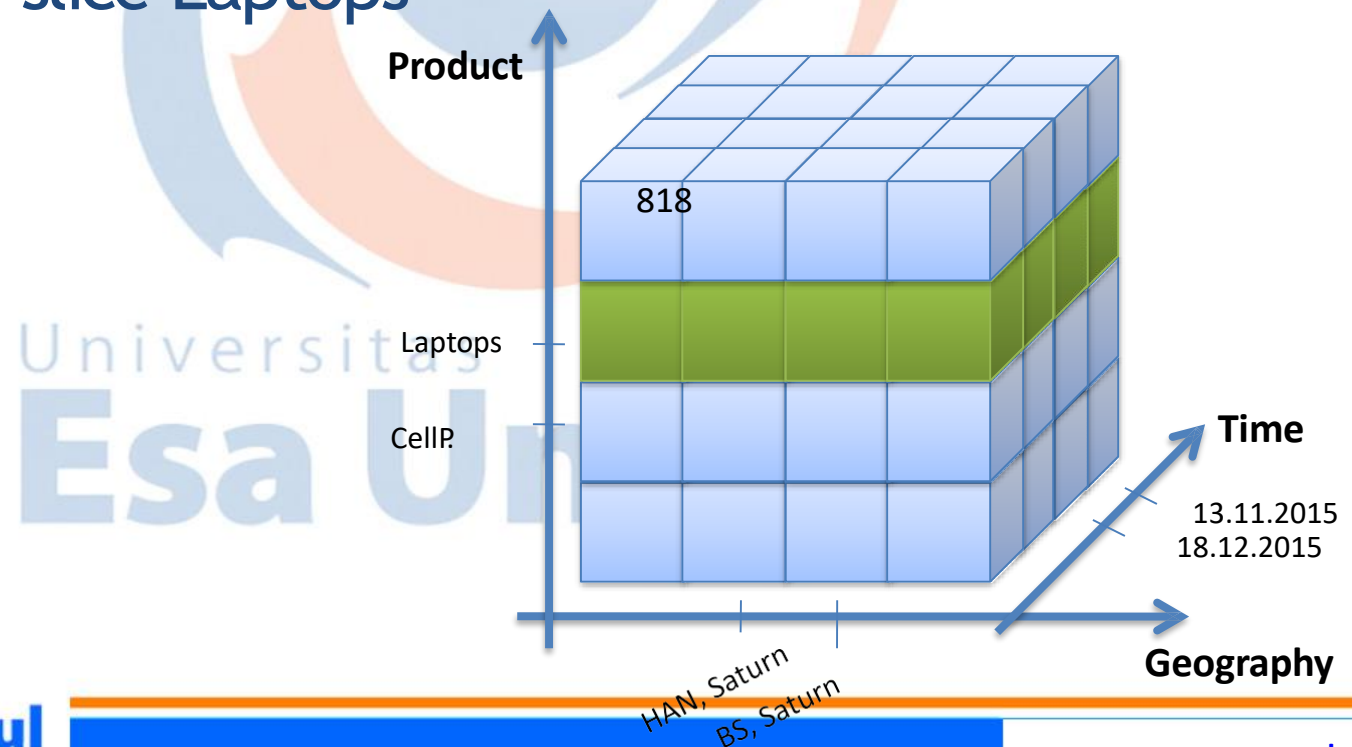
- Slice: a **subset** of the multi-dimensional array corresponding to a **single value** of one or more dimensions and **projection on the rest** of dimensions
 - E.g., project on Geo (store) and Time from values corresponding to Laptops in the product dimension

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

Universitas
Esa Unggul

Slice...

- 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

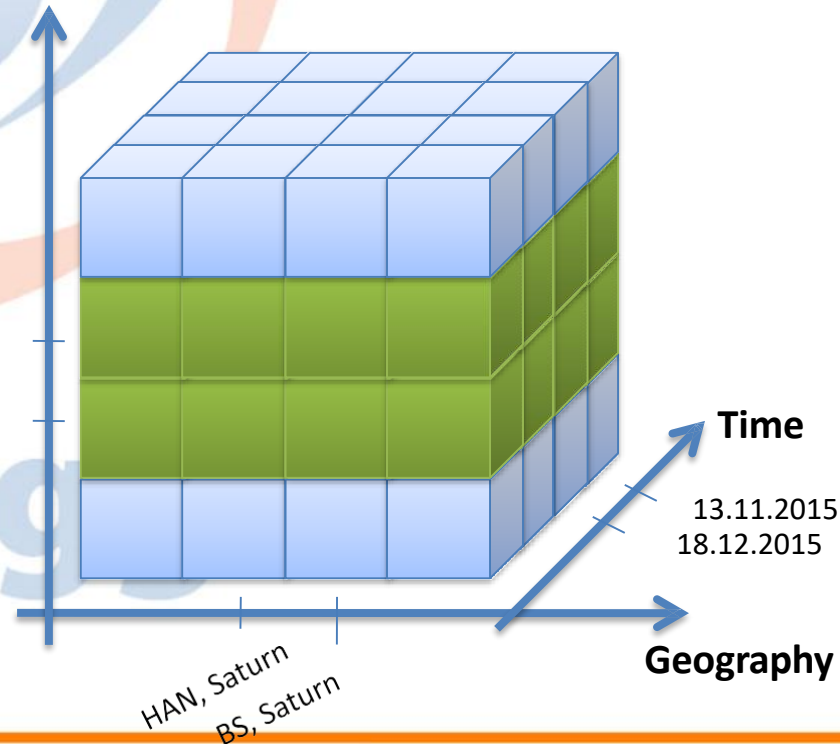
$\pi_{\text{StoreId, TimeId, Amount}} (\sigma_{\text{ArticleId}} \in \{\text{Laptop, CellP}\}(\text{Sales}))$

Universitas
Esa Unggul

Product

Laptops

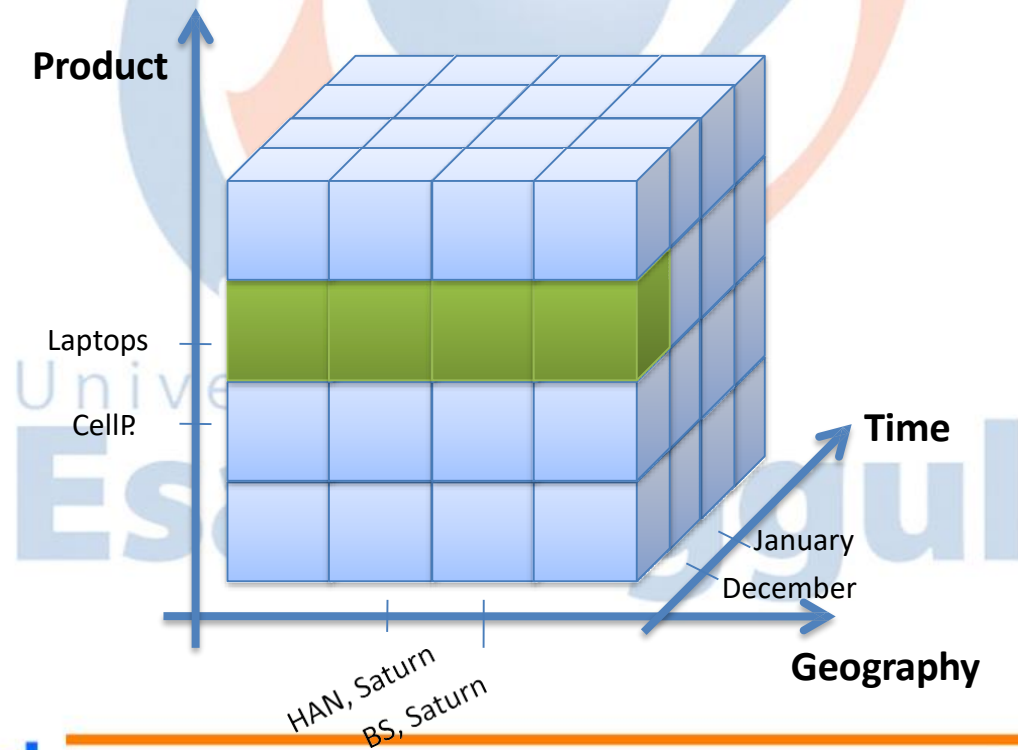
CellP.



Dice...

- E.g. equality SELECT on 2 dimensions Product and Time

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

- Consider pivoting the following data

Location	
CityId	City
1	Braunschweig
2	Hannover
3	Hamburg

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

Universitas
Esa U

Pivot...

- Pivoting on City and Day

	Mon	Tue	Wed	Thu	Fri	Sat	San	SubTotal
Hamburg	60	60	0	140	0	880	0	1140
Hannover	550	0	0	0	100	0	0	650
Braunschweig	540	300	0	0	0	0	50	890
SubTotal	1150	360	0	140	100	880	50	2680

	Hamburg	Hannover	Braunschweig	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

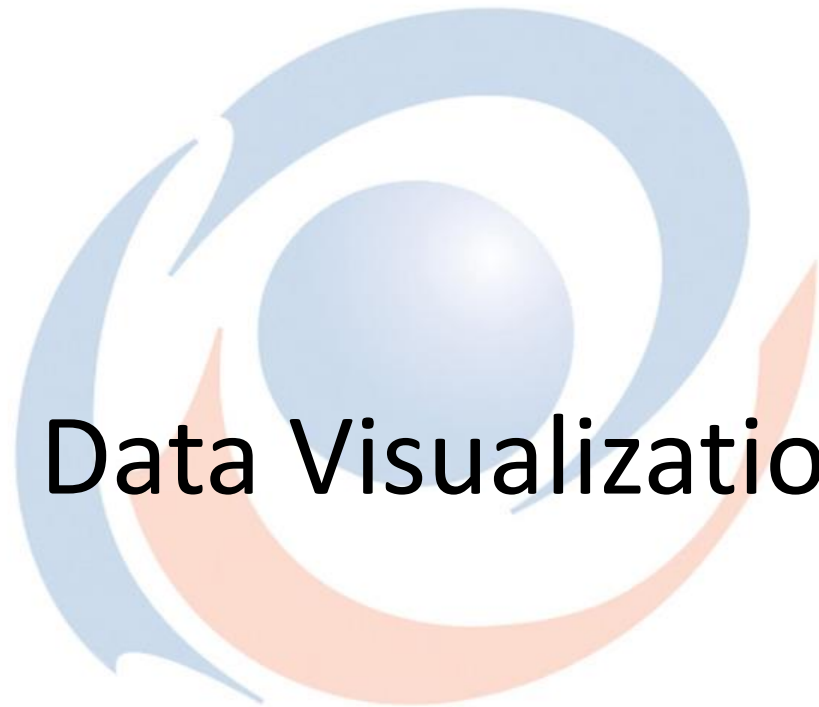
– Most analysts and decision makers

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:	PROTOTYPE	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	■



Data Visualization

Universitas
Esa Unggul

OLAP Data Visualization

- How do these operations look like for the user?
 - E.g. Crystal Decisions later bought by SAP and integrated into Business Objects
 - 2 dimensions ... is trivial
 - E.g. Products by Store

Product dimension →

Store dimension ↓

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

OLAP Data Visualization...

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

The screenshot shows an OLAP interface with a table of data. The table is nested on the 'Store' axis. The columns represent 'Products' (Bulbs, Batteries, Fuses) and the rows represent 'Measure' (Revenue, Cost) for each 'Store' (Uptown, Midtown, Downtown). The 'Days' axis is set to 'Monday'.

Store	Measure	Bulbs	Batteries	Fuses
Uptown	Revenue	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

OLAP Data Visualization...

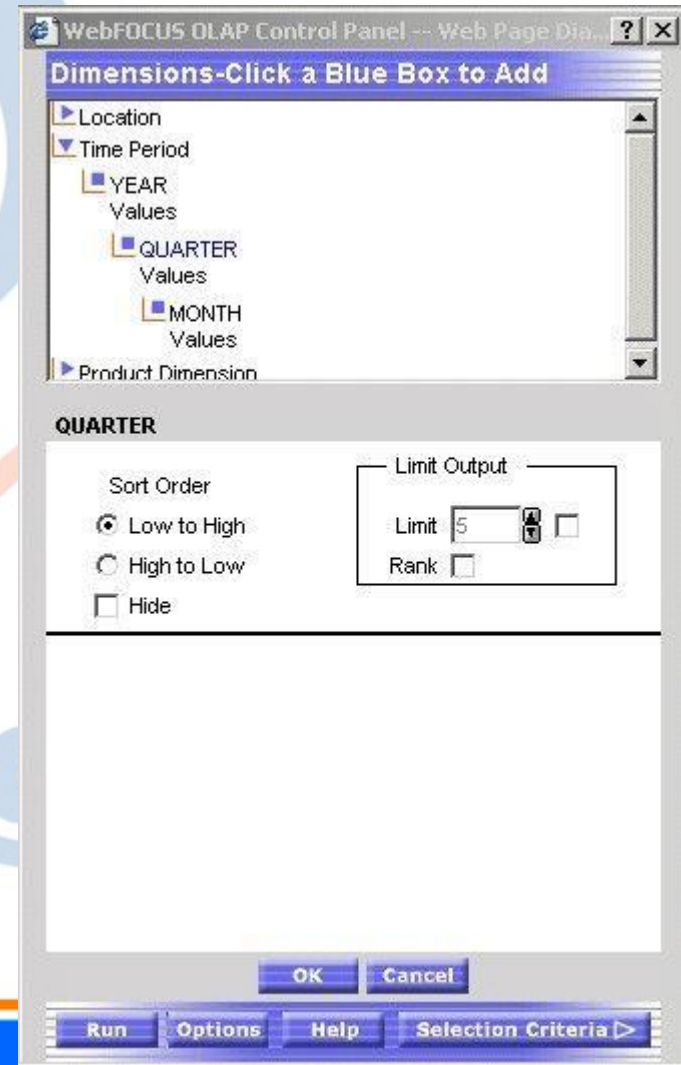
- OLAP reporting has to be very flexible
 - The IBM Infosphere - OLAP web based report

The screenshot displays a web browser window with the address `http://localhost/cgi-bin/ibi CGI/webapi.dll`. The main content area shows a data table with columns: QUARTER, Store Name, PRODTYPE, and Quantity. The data is filtered for Q1, with rows for various store names and product types. On the right side, there are several control panels: 'Dimensions' (Location, Time Period, Product Dimension), 'Drill Across' (Shift Up, Shift Down, Remove, Pivot), 'Drill Down' (QUARTER, Store Name, PRODTYPE), and 'Measures' (Quantity, Line Cost Of Goods). The 'Measures' panel has checkboxes for 'Quantity' and 'Line Cost Of Goods', both of which are checked. At the bottom, there are buttons for 'Run', 'Options', 'Help', and 'Selection Criteria'.

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...

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



Universitas
Esa Unggul

OLAP Data Visualization...

- Trends Visualization
 - With the help of charts

QUARTER	Store Name:	PRODTYPE	Quantity:	Line Cost Of Goods Sold
Q1	AV VideoTown	Analog	18,449	3,969,296.00
		Digital	22,206	5,109,400.00
	Audio Expert	Analog	78,449	16,467,146.00
		Digital	105,983	25,092,678.00
	City Video	Analog	6,287	1,315,015.00
		Digital	7,196	1,607,513.00
	Consumer Merchandise	Analog	6,980	1,542,036.00
		Digital	14,957	3,251,090.00
	TV City	Analog	19,077	3,772,119.00
		Digital	41,307	10,128,967.00
	Web Sales	Analog	545	124,366.00
		Digital	829	190,201.00
	eMart	Analog	97,128	21,152,262.00
		Digital	108,221	24,990,368.00
Q2	AV VideoTown	Analog	11,781	2,663,655.00
		Digital	27,377	5,928,507.00
	Audio Expert	Analog	57,944	11,868,758.00
		Digital	111,421	28,064,250.00

OLAP Data Visualization...

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

Sales Analysis - Monitors (2009)

	Qtr. 1			Qtr. 2			Qtr. 3			Qtr. 4			Year
	Budget	Actual	Variance	Budget	Actual	Variance	Budget	Actual	Variance	Budget	Actual	Variance	
199.52	1,939,131.04	11.18%	1,339,474.32	1,285,712.06	-4.01%	1,697,484.17	1,809,508.61	6.60%	1,595,407.64	1,795,338.46	12.53%	█	
739.45	811,563.58	9.71%	739,133.39	713,840.25	-3.42%	709,165.67	810,790.78	14.33%	619,628.90	649,482.11	4.82%	█	
736.18	823,683.55	13.65%	582,026.29	631,890.28	8.57%	861,252.55	737,492.53	-14.37%	782,158.50	775,766.24	-0.82%	█	
762.46	476,288.38	14.48%	419,046.14	436,510.37	4.17%	460,260.08	490,622.56	6.60%	469,253.72	503,047.98	7.20%	█	
737.60	4,050,666.55	11.75%	3,079,680.13	3,067,952.96	-0.38%	3,728,162.47	3,848,414.48	3.23%	3,466,448.76	3,723,634.79	7.42%	█	

Sales Analysis - Monitors (2009)

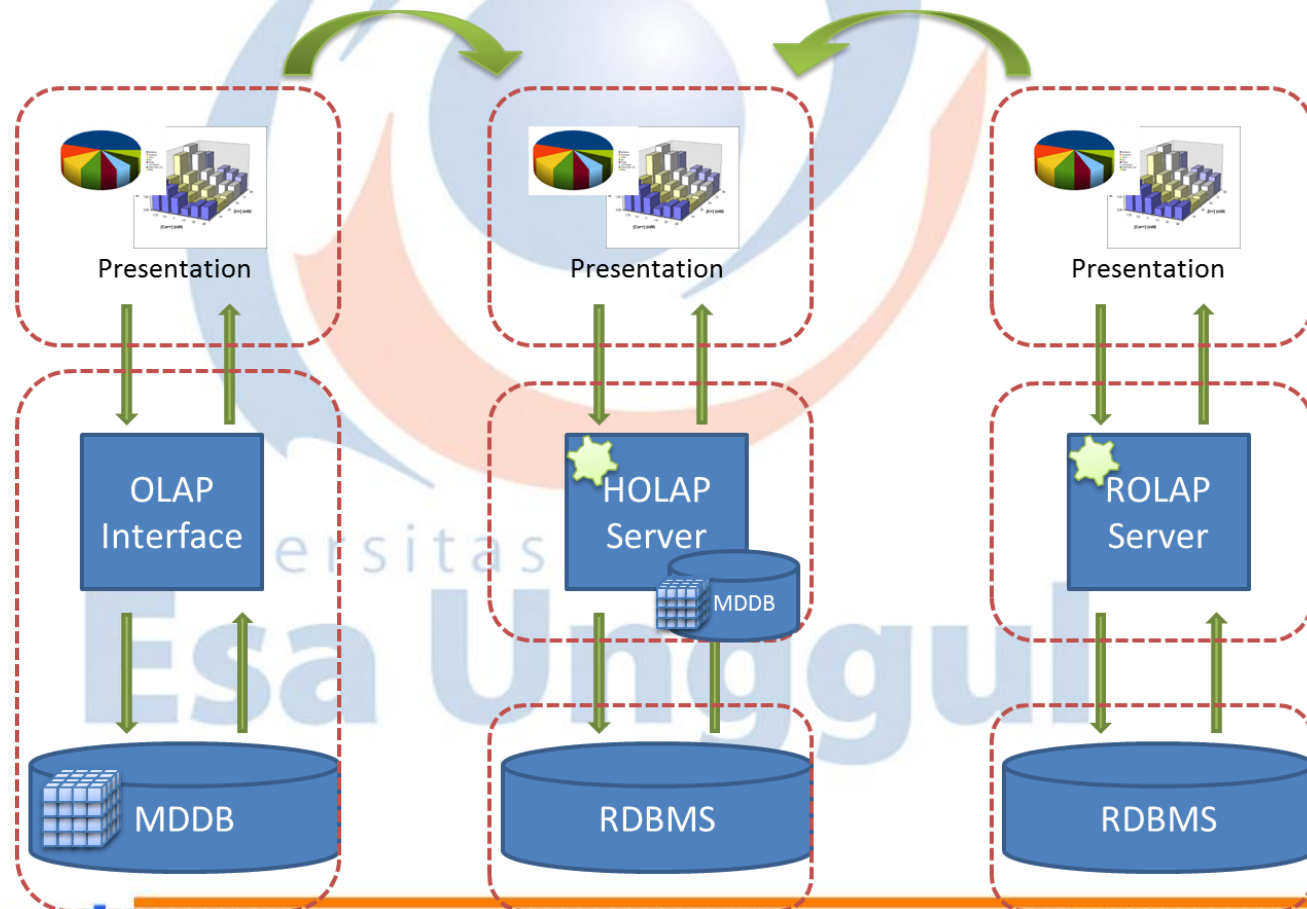
Quarter	Actual	Budget
Qtr. 1	~1,939,131	~1,339,474
Qtr. 2	~811,564	~739,133
Qtr. 3	~823,684	~582,026
Qtr. 4	~476,288	~419,046

Actual Cost of Sales by Region - Monitors (2009)

Region	Percentage
North	12.98%
South	20.21%
East	20.32%
West	46.49%

From Presentation to Data

- Remember the D W architecture?



From Presentation to Data...

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

Universitas
Esa Unggul

OLAP Server

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

Universitas
Esa Unggul

OLAP Server...

- OLAP server may either
 - **Physically stage** the processed multidimensional information to deliver consistent and rapid response times to end users (**MOLAP**)
 - Store data in **relational databases** and **simulate multidimensionality** with special schemas (**ROLAP**)
 - Or offer a choice of both (**HOLAP**)

From Presentation to Data

- 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



OLAP Query

Universitas
Esa Unggul

Typical OLAP Query

- 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 queries in SQL

```
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;
```

OLAP Query Language

- SQL-99
 - Prepare SQL for OLAP queries
 - New SQL commands
 - GROUPING SETS
 - ROLLUP
 - CUBE
 - New aggregate functions
 - 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 advanced statistical functions

SQL-92

- 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 aggregation query

	BMW	Mercedes	By model
SUV			
Sedan			
Sport			
By maker			
			SUM

```

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

SQL-92

- 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

- 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 REPLACE VIEW 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;

OLAP Function

- **Moving-averages, Percentiles, Ranks** are all hard to compute with SQL-92
 - Involves multiple self joins of the fact table
- SQL-99 introduced the **window clause** for creating dynamical windows of data

Universitas
Esa Unggul

Window Clause

- The window clause specifies an **action to perform over a set of rows**
 - 3 sub-clauses: **Partitioning, ordering and aggregation grouping**
 - **<aggregate function> OVER ([PARTITION BY <column list>] ORDER BY <sort column list> [<aggregation grouping>])**
 - **SELECT ... AVG(sales) OVER (PARTITION BY region ORDER BY month ASC ROWS 2 PRECEDING) AS SMA3...**

(moving average of 3 rows)

NTILE

- Not a part of the SQL99 standard, but adopted by major vendors
- NTILE splits a set into equal groups
 - It **divides** an ordered partition into **buckets** and assigns a bucket number to each row in the partition
 - Buckets are calculated so that each bucket has **exactly the same number of rows** assigned to it or at most 1 row more than the others

NTILE...

- `SELECT ... NTILE(3) OVER (ORDER BY sales)`
`NT_3 FROM ...`

CHANNEL	CALENDAR	SALES	NT_3
Direct sales	02.2016	10,000	1
Direct sales	03.2016	9,000	1
Internet	02.2016	6,000	2
Internet	03.2016	6,000	2
Partners	03.2016	4,000	3

- `NTILE(4)` - quartile
- `NTILE(100)` - percentage

SQL-99

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



Universitas
Esa U

Grouping Operator

- **GROUPING SET**

- **Efficiently** replaces the series of UNIONed queries

- `SELECT dept_name, CAST(NULL AS CHAR(10)) AS job_title, COUNT(*) FROM personnel`

`GROUP BY dept_name`

`UNION ALL`

- `SELECT CAST(NULL AS CHAR(8)) AS dept_name, 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);`

Grouping Sets

- The issue of **NULL** values
 - The new grouping functions generate NULL values at the subtotal levels
 - How do we tell the difference between “generated NULLs” and “real NULLs” from the data itself?
 - The **GROUPING** function call returns 0 for NULL in the data and 1 for generated NULL

Year	Brand	SUM(qty)	
2016	Real NULL	250	} (year, brand)
2016	BMW	300	
2016	VW	450	
2016	Gen. Null	1000	} (year)

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 Roll-up operation translate to (N+1) grouping sets
 - **Order** is significant for Roll-up!
 - GROUP BY ROLLUP (c, b, a) is equivalent with grouping sets of (c, b, a), (c, b), (c), ()

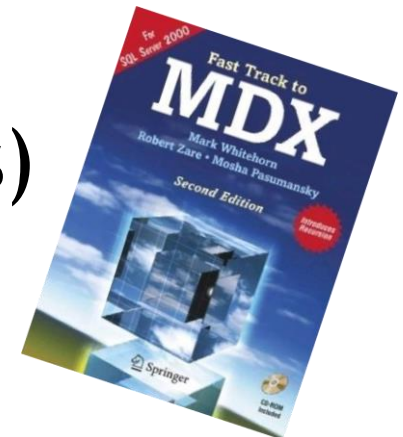
Roll-Up...

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

Year	Brand	SUM(qty)	
2015	Mercedes	250	
2015	BMW	300	(year, brand)
2015	VW	450	
2015	NULL	1000	(year)
2016	Mercedes	50	(year, brand)
...	
2016	NULL	400	(year)
NULL	NULL	1400	(ALL)

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 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 Basics

- **Similar to SQL syntax**

```
SELECT {Germany, Niedersachsen, Bayern, Frankfurt} ON COLUMNS,  
       {Qtr1.CHILDREN, Qtr2, Qtr3} ON ROWS  
FROM SalesCube  
WHERE (Measures.Sales, Time.[2015], Products.[All Products]);
```

- **SELECT**

- Dimensions, on columns and rows

- **FROM**

- Data source cube specification
- If joined, data cubes must share dimensions

- **WHERE**

- **Slicer** - restricts the data area
- Specifies the measures to return

MDX Basic Elements

- Lists: Enumeration of elementary nodes from different classification levels
 - E.g. {Germany, Niedersachsen, [Frankfurt am Main], USA}
- Generated elements: Methods which lead to new sets of the classification levels
 - Germany.CHILDREN generates: {Niedersachsen, Bayern,...}
 - Niedersachsen.PARENT generates Germany
 - Time.Quarter.MEMBERS generates all the elements of the classification level
- Functional generation of sets
 - DESCENDENT(USA, Cities): children of the provided classification level *Cities*
 - GENERATE ({USA, France}, DESCENDANTS(Geography.CURRENT, Cities)): enumerates all the cities in USA and France

MDX Basics

- **Sets nesting** combines individual coordinates to reduce dimensionality

```

— SELECT CROSSJOIN({Germany, Sachsen, Hannover, BS}{Ikea, [H&M-Möbel]})
  ON COLUMNS,
  {Qtr1.CHILDREN, Qtr2} ON ROWS
FROM salesCube
WHERE (Measure.Sales, Time.[2015], Products.[All Products]);

```

	Germany		Sachsen		Hannover		BS	
	Ikea	H&M-Möbel	Ikea	H&M-Möbel	Ikea	H&M-Möbel	Ikea	H&M-Möbel
Jan 14								
Feb 14								
Mar 14								
Qtr2								

sales numbers

MDX Basics

- Relative selection
 - Uses the **order** in the **dimensional structures**
 - Time.[2015].LastChild :last quarter of 2015
 - [2015].NextMember :{[2016]}
 - [2015].[Qtr4].Nov.Lead(2) :Jan 2016
 - [2010]:[2015] represents [2010], ..,[2015]
- Methods for hierarchy information extraction
 - Germany.LEVEL :country
 - Time.LEVELS(1) :Year
- Brackets
 - {}: Sets, e.g. {Hannover, BS, John}
 - []: text interpretation of numbers, empty spaces between words or other symbols
 - E.g. [2015], [Frankfurt am Main], [H&M]
 - (): tuple e.g. WHERE (Measure.Sales, Time.[2015], Products.[All Products])

MDX Basics...

- **Special functions and filters**

- Special functions TOPCOUNT(), TOPPERCENT(), TOPSUM()

- e.g. top 5 areas of Germany by turnover on rows

```
SELECT {Time.CHILDREN} ON COLUMNS,  
       {TOPCOUNT(Germany.CHILDREN, 5, Sales.turnover)} ON ROWS  
FROM salesCube  
WHERE (Measure.Sales, Time.[2018]);
```

- FILTER function, e.g. areas of Germany with increased turnover for 2018 compared to 2017

```
SELECT FILTER(Germany.CHILDREN, ([2018], Turnover) > ([2017], Turnover))  
       ON COLUMNS, Quarters.MEMBERS ON ROWS  
FROM salesCube  
WHERE (Measure.Sales, Time.[2018], Products.Electronics);
```

MDX Basics...

- **Time series**

- Set Value Expressions e.g., choosing time intervals
 - PERIODSTODATE(Quarter, [15-Nov-2018]):
returns 1.10.-15.11.2018
- Member Value Expressions e.g. pre-periods
 - PARALLELPERIOD(Year, 3, [Sep-2018]): returns [Sep-2015]
- Numerical functions: covariance, correlation, linear regression

mdXML

- **XMLA** (XML for Analysis)
 - Most recent attempt at a standardized API for OLAP
 - Allows client applications to talk to multi-dimensional data sources
 - In XMLA, mdXML is a MDX wrapper for XML
 - Underlying technologies
 - XML, SOAP, HTTP
 - Service primitives
 - DISCOVER
 - Retrieve information about available data sources, data schemas, server infos...
 - EXECUTE
 - Transmission of a query and the corresponding conclusion

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



Thank You...

Universitas
Esa Unggul



Universitas
Esa Unggul