

# Data Warehouse

***Martin Clement***

*Head of Analytics*

[Martin.Clement@atvantage.com](mailto:Martin.Clement@atvantage.com)

**ATVANTAGE**



| Data Warehouse, 2025

# OLAP & Multidimensional Models

ATVANTAGE



**TIMETOACT**  
**GROUP**

# What is OLAP?

- OLAP => **OnLine Analytical Processing**
- First OLAP database was Express, introduced in 1970
- Ted Codd, inventor of the relational database model, proposed 12 rules for OLAP in 1993 in a white paper for Arbor Essbase
  - ▶ Later modified to 18 features in 4 groups in 1995
- Criteria for OLAP systems like
  - ▶ Multi-dimensionality
  - ▶ Transparency
  - ▶ Constant response-times
  - ▶ Dynamic treatment of sparse matrices
  - ▶ Multi-user support
  - ▶ Flexible definition of reports
  - ▶ No limits on dimensions and hierarchy levels

# What is OLAP?

- Nigel Pendse of The OLAP Report, a prominent independent group, proposed a simpler definition in 2005:
  - ▶ **Fast Analysis of Shared Multidimensional Information**, or FASMI for short
    - **Fast** - *The system is targeted to deliver most responses to users within about five seconds.*
      - maximum response time for regular queries 5 seconds / complex queries 20 seconds
    - **Analysis** - *The system can cope with any business logic and statistical analysis that is relevant. The target user should be able to define ad hoc calculations quickly and easily.*
      - *intuitive analysis, queries may contain arbitrary computations*
    - **Shared** - *The system implements all the security requirements for confidentiality, including concurrent update locking if write access is needed.*
      - *Shared usage and access control*
    - **Multidimensional** - *The system must provide a multidimensional conceptual view of the data, including full support for hierarchies and multiple hierarchies, as this is certainly the most logical way to analyze businesses and organizations.*
      - *Multidimensional view on the data regardless of the underlying data model*
    - **Information** - *All of the data and derived information needed.*
      - Scalability, the response times do not depend on the size of the data
  - In both definitions, the key aspect is - **Multidimensionality**

# Recap: Multidimensional data model

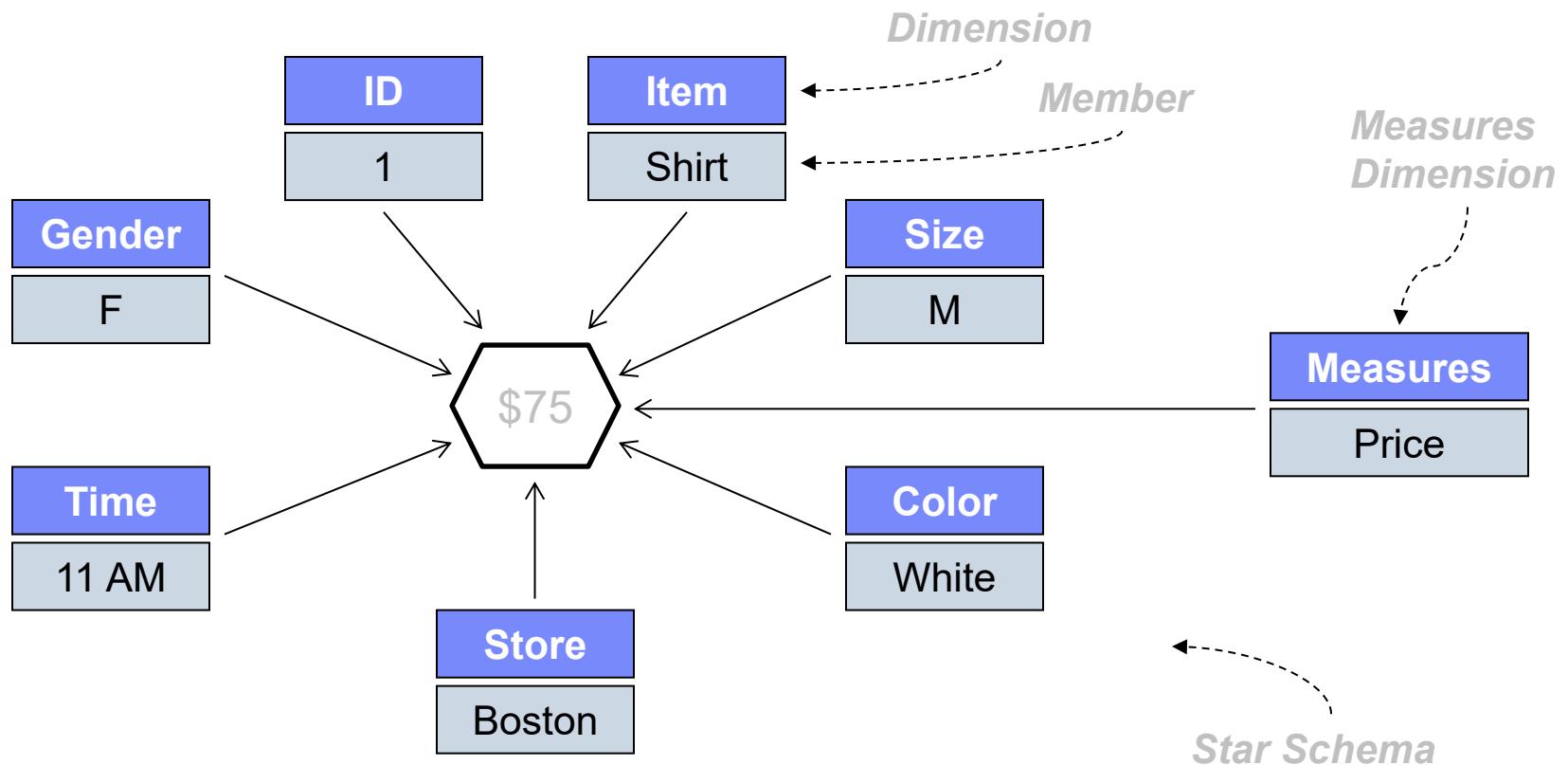
- Euclidean geometry:
  - ▶ Points are defined by coordinates on x-y-z axes
- OLAP:
  - ▶ Data points are defined by “coordinates” on 2 or more “axes”
    - Axes => **Dimensions**
    - Coordinates => **Members** or **Elements**
    - Data values => **Measures**
  - ▶ Data points are stored in a **Cube**
- Problem: We can visualize in 2 and 3 dimensions. How do we wrap our heads around more than 3 dimensions?

# Example – Relational view

ID	Item	Size	Color	Store	Time	Gender	Price
1	Shirt	M	White	Boston	11 AM	F	\$75
2	Pant	L	White	Boston	7 PM	M	\$50
3	Pant	S	Black	Ottawa	4 PM	F	\$90
4	Pant	L	Gray	Ottawa	8 PM	M	\$60
5	Shorts	L	Brown	Ottawa	6 PM	M	\$40

# Example – Multidimensional view

ID	Item	Size	Color	Store	Time	Gender	Price
1	Shirt	M	White	Boston	11 AM	F	\$75



# Basic visualization – Crosstab



# Aggregation (Consolidation)



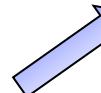
# Drill down / up

- **Drill down / drill up** provides the ability to view the data at different levels of detail.

ID	Size	Color	Store	Time	Gender
All	All	All	All	All	All

**Measures**

Item	Price
- All	\$315
Shirt	\$75
Pant	\$200
Shorts	\$40



# Pivot (Slice & Dice)

- **Slice & Dice** provides the ability to view the data from different angles.



# Stacking

ID	Size	Color	Store	Time	Gender
All	All	All	All	All	All



Store	Measures	
	Item	Price
- All	- All	\$315
	Shirt	\$75
	Pant	\$200
	Shorts	\$40
Boston	- All	\$125
	Shirt	\$75
	Pant	\$50
	Shorts	\$0
Ottawa	- All	\$190
	Shirt	\$0
	Pant	\$150
	Shorts	\$40

# Calculated member

- A **calculated member** is a member whose value is derived from other members, often as a result of a mathematical and/or logical calculation.

ID	Size	Color	Store	Time	Gender
All	All	All	All	All	All

Item	Measures		
	Price	Cost	{Profit}
- All	\$315	\$145	\$170
Shirt	\$75	\$15	\$60
Pant	\$200	\$120	\$80
Shorts	\$40	\$10	\$30

*Calculated member*  
 $\text{Profit} = \text{Price} - \text{Cost}$

# OLAP Level

- Definition: Group of members related to the same human concept. The members are distinct value (instance) of the concept being at the same level of aggregation (granularity) of the data.
- Level used to define human business concept, user want to analyze the data by.
  - ▶ Example: City level, Country level, Gender Level, Color level, Store level, etc.
- Similar to relational column:
  - ▶ OLAP level  $\approx$  Relational database column
  - ▶ OLAP member  $\approx$  distinct column value



Level

↓

Member [ Item ]

Member [ Size ]

Member [ Color ]

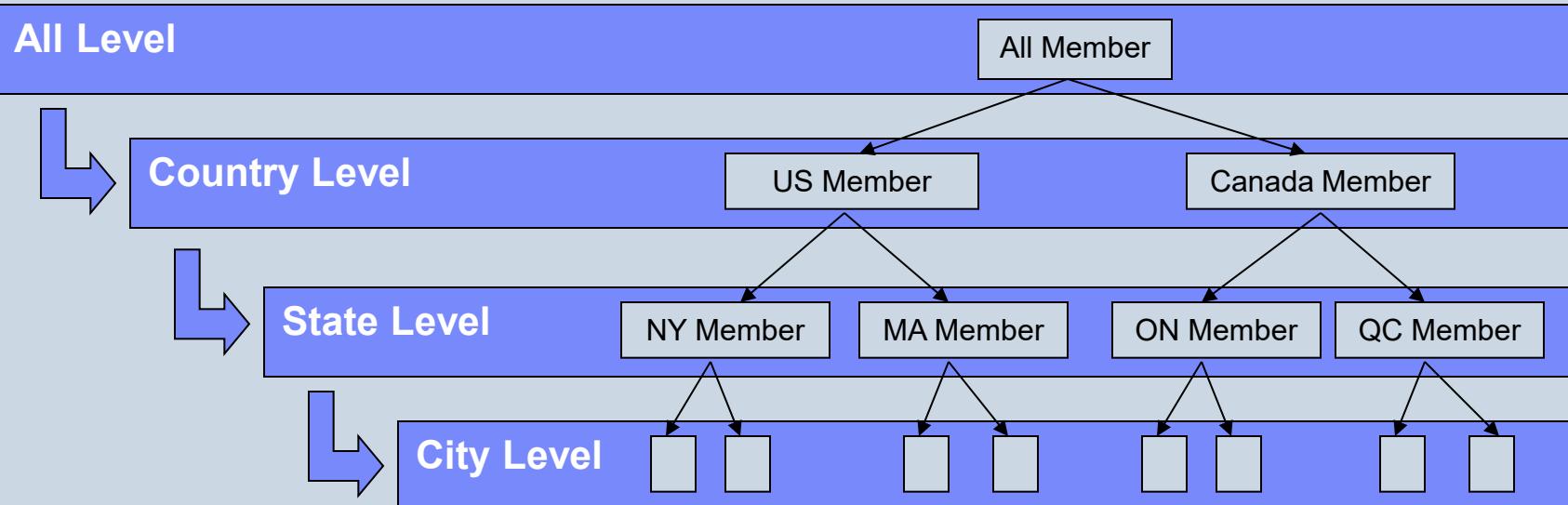
Member [ Store ]

Item	Size	Color	Store
Shirt	M	White	Boston
Shirt	L	White	Boston
Pant	S	Black	Ottawa
Pant	L	Gray	Ottawa
Shorts	L	Brown	Ottawa

# OLAP Hierarchy

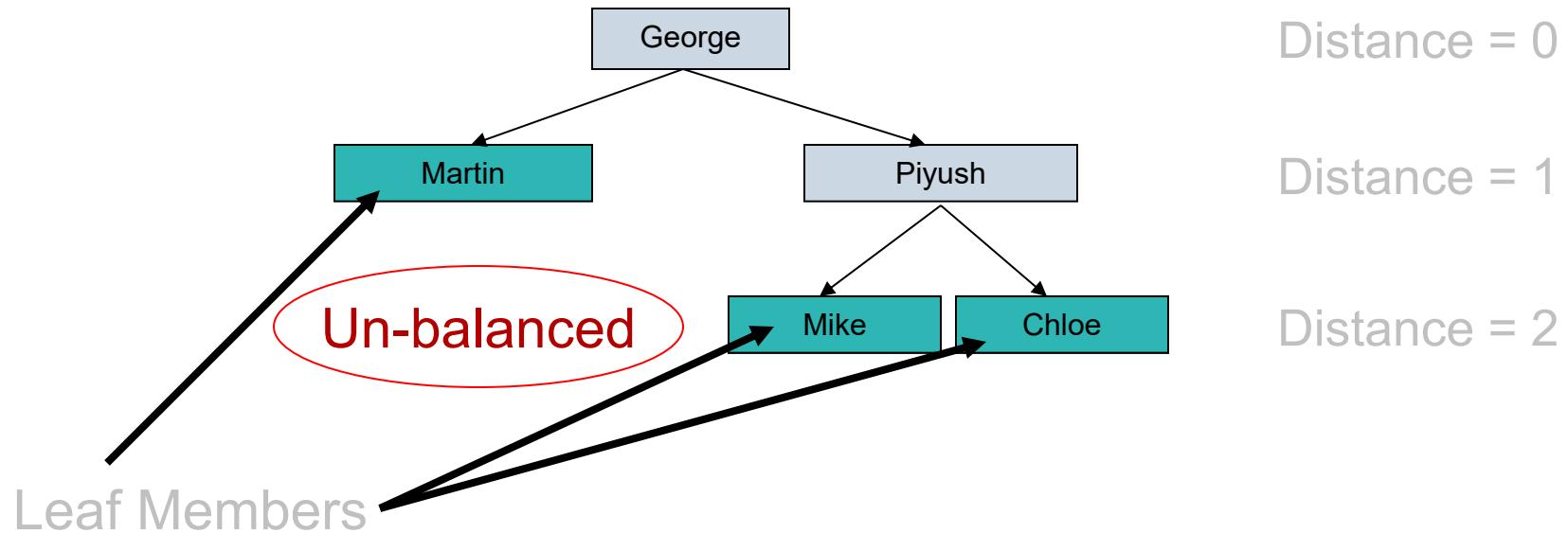
- Definition: Define strict composition and decomposition relationship between levels and members. The relationship is based on human (business) view of the world and natural relationships between those concepts (levels).
- Hierarchy define the member drill-down and drill-up behavior.

## Geography Hierarchy



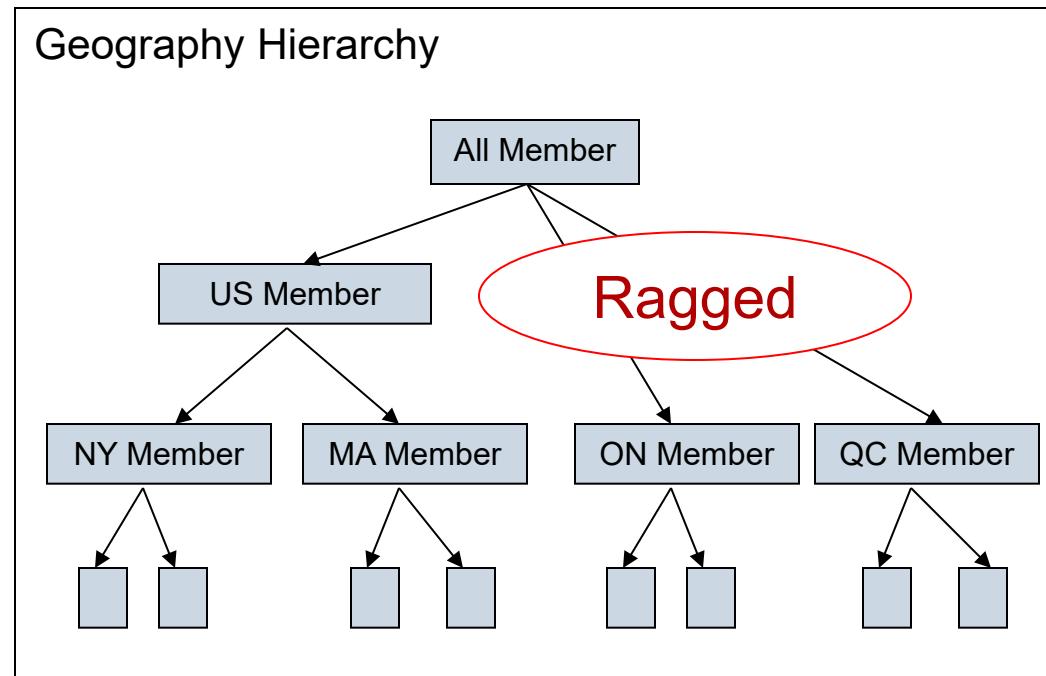
# OLAP Hierarchy – Un-Balanced

- Hierarchy is un-balanced when the leaf members are not all part of the same level or at the same distance from the root member
- Level based hierarchy mapped to relational database – Leaf member missing or if being null values (then suppressed, no placeholder used), i.e. no actual leaf member for some of the parent member at the next level



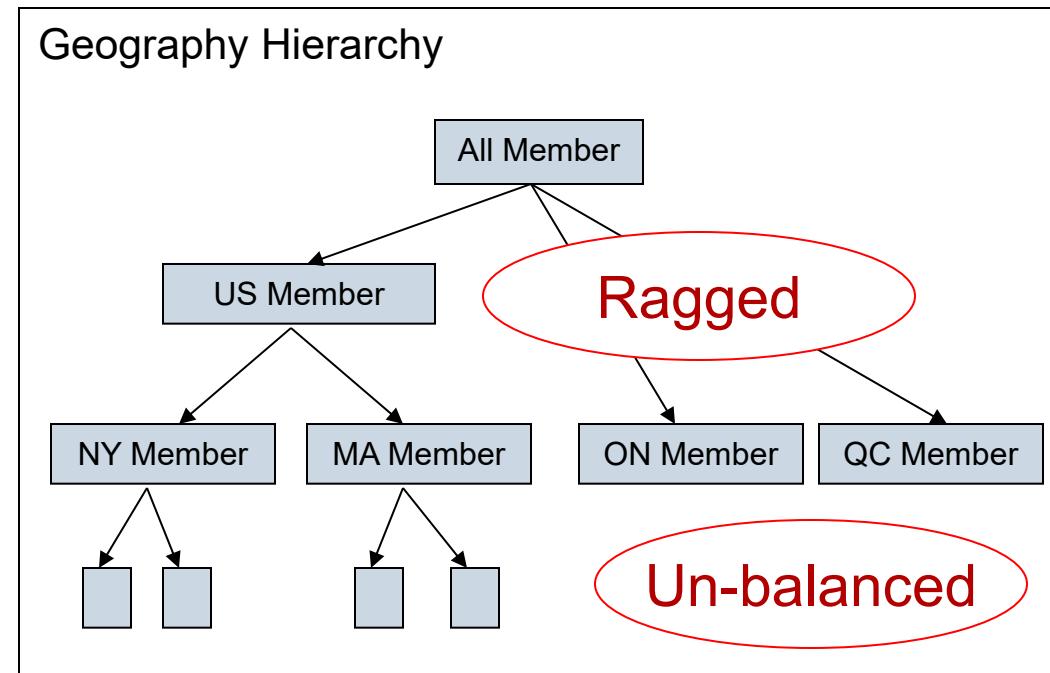
# OLAP Hierarchy - Ragged

- A ragged hierarchy is a hierarchy where there is a parent and child relation between members, but for member being not of consecutive level.
- Level based hierarchy mapped to relational database – Member missing or if being null value (then suppressed, no placeholder used), i.e. no actual member at a non-leaf level



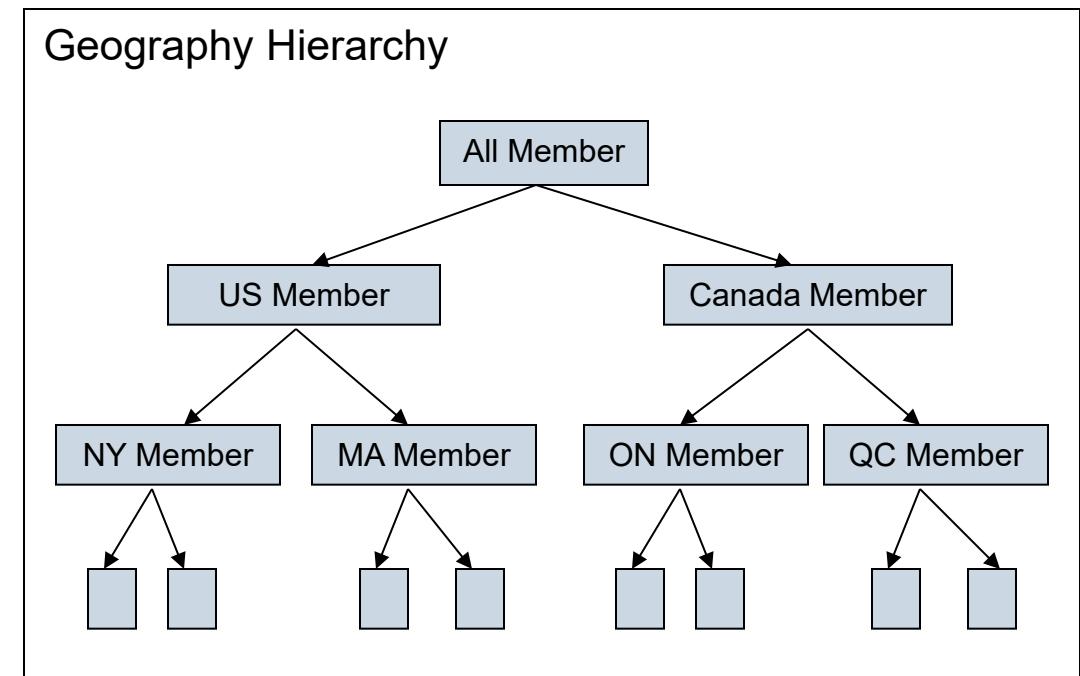
# OLAP Hierarchy – Ragged and Un-balanced

- A hierarchy could be un-balanced and ragged!



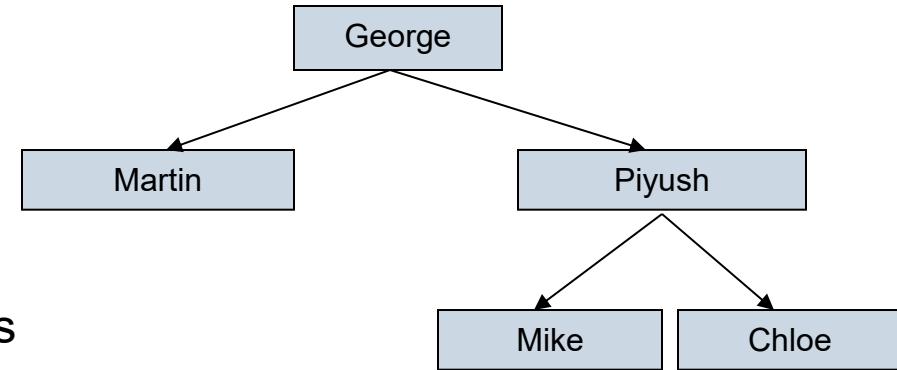
# OLAP Hierarchy – Parent-Child

- Hierarchy exist without level
- Member organized into a hierarchy, defining each parent/child relationship.
- Virtual level could be defined
  - ▶ Distance to the root
  - ▶ Distance to the leaf
  - ▶ Arbitrary grouping of members



# OLAP Hierarchy – Parent-Child – Relational database

- Parent-child hierarchy is similar to relational database table having recursive join
- Employee / manager relationship is common case

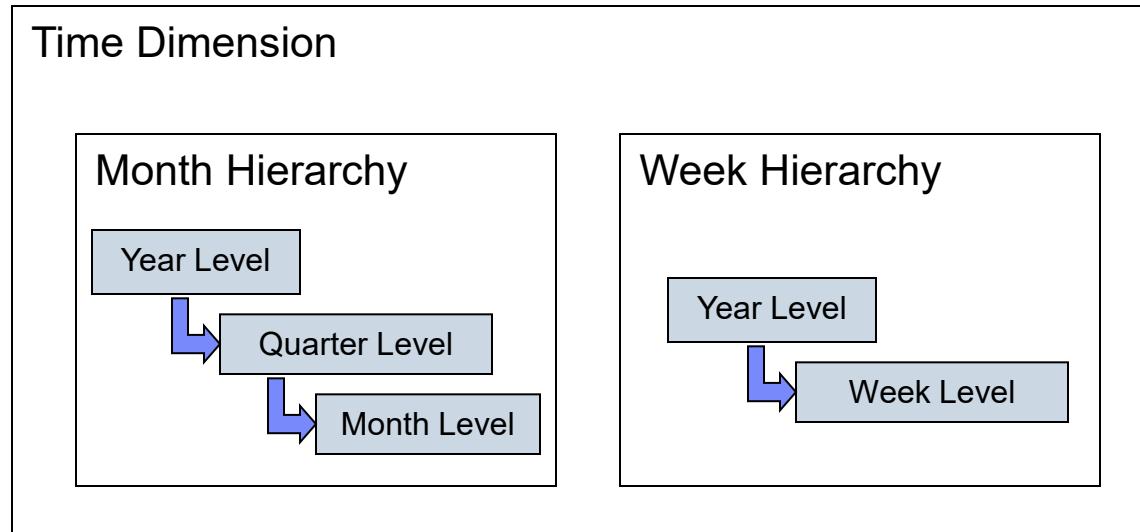


↓

Employee ID	Employee Name	Manager ID	Title
1	George	null	Director
2	Piyush	1	Manager
3	Martin	1	Architect
4	Mike	2	Developer
5	Chloe	2	Developer

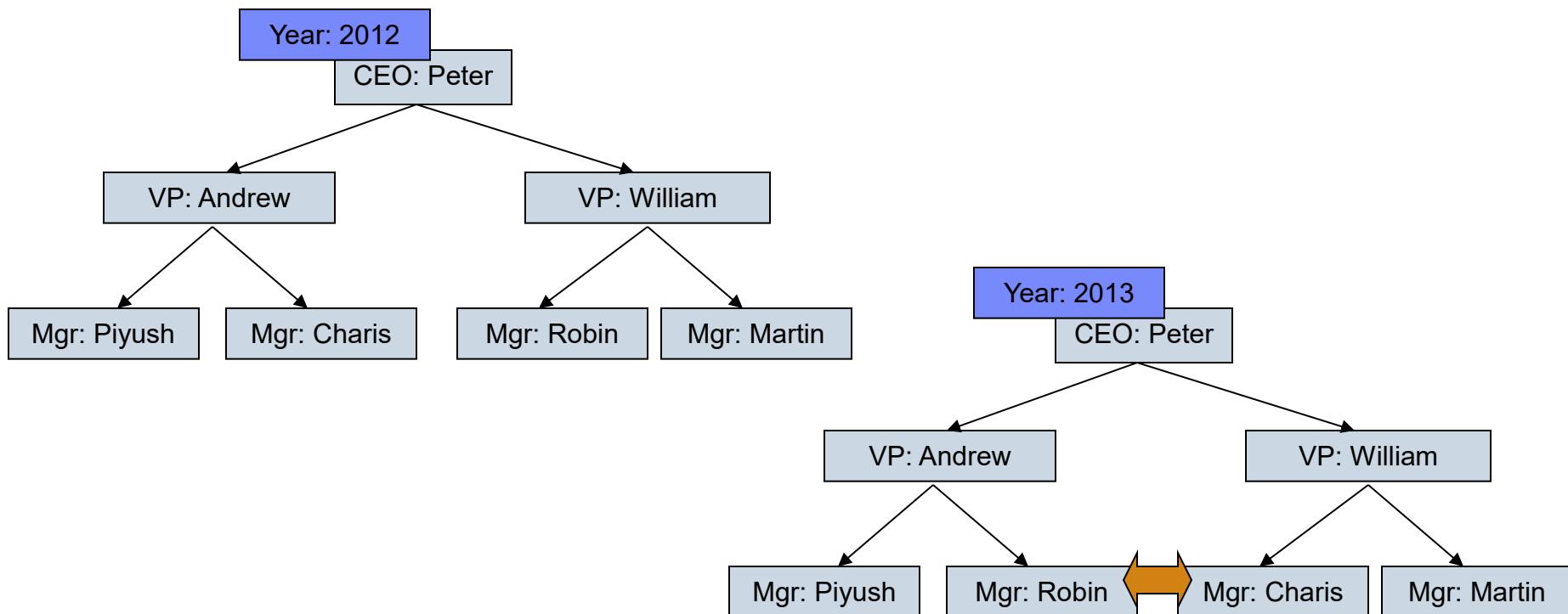
# Multiple Hierarchy

- Dimension could have more than one hierarchy
- It could be set of arbitrary hierarchies
- It could be hierarchies sharing the same levels, organized in different way
- Sometime, hierarchy must have the same leaf members (e.g. only different grouping)



# Slowly Changing Hierarchy

- Some hierarchy change over time
- For example, employee could report to different manager over different period of time
- The hierarchy changes is impacting the measure aggregation value



# Implementing Dimensions and Hierarchies

ATVANTAGE



**TIMETOACT**  
**GROUP**

# Dimensions

- Dimension types
  - ▶ Fundamental dimensions
    - Change the granularity of the data warehouse solution
    - Adding of a dimension changes the number of data records in the fact table
  - ▶ Describing dimensions
    - Contrary to fundamental dimensions
    - No influence on granularity
    - Contain additional attributes that belong together / do not match for another dimension

# Hierarchies

- Hierarchy data in one or more tables
  - ▶ the dimension table(s)
- Measures and dimension keys in one or more fact tables
  - ▶ Joins are complex with more than one fact table
  - ▶ Dimension key is on detail level of hierarchy
- Aggregated values for each hierarchy level
  - ▶ Pre-calculated and stored in the database
  - ▶ dynamically calculated, i.e. through built-in aggregation functions

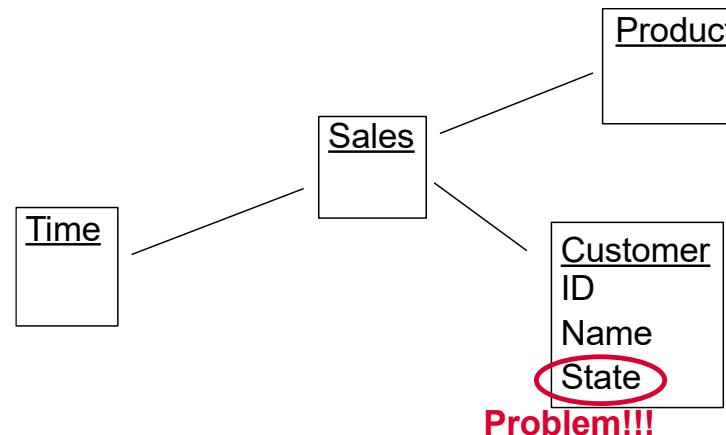
# Hierarchies

Key	Product_ID	Product_Name	Product_Group_ID	Product_Group_Name	Product_Family_ID	Product_Family_Name	... [further describing attributes]
8987	P001	Laptop L1	PG001	Laptop	PF001	PC	...
8988	P002	Laptop L2	PG001	Laptop	PF001	PC	...
8989	P003	Laptop L3	PG001	Laptop	PF001	PC	
8990	-	-	PG001	Laptop	PF001	PC	
8991	P004	Desktop D1	PG002	Desktop	PF001	PC	
8992	P005	Desktop D2	PG002	Desktop	PF001	PC	
	-	-	PG002	Desktop	PF001	PC	
	P006	Multimedia M1	PG003	Multimedia PC	PF001	PC	
	P007	Multimedia M2	PG003	Multimedia PC	PF001	PC	
	-	-	PG003	Multimedia PC	PF001	PC	
	-	-	-	-	PF001	PC	
	P008	Printer LP1	PG004	Laser Printer	PF002	Printer	
8999	P009	Printer LP1	PG004	Laser Printer	PF002	Printer	
9000	-	-	PG004	Laser Printer	PF002	Printer	
9001	P010	Printer DJ1	PG005	Deskjet	PF002	Printer	
9002	P011	Printer DJ2	PG005	Deskjet	PF002	Printer	
9003	-	-	PG005	Deskjet	PF002	Printer	
9004	-	-	-	-	PF002	Printer	

Key for  
product group  
Laptop

# Slowly changing dimensions

- Assumption until now:
  - ▶ static dimension tables
- However: changes in dimension tables are possible
  - ▶ new products are added or old ones delisted
  - ▶ the vendor of a product changes
    - IBM PCs → Lenovo PCs
  - ▶ a customer moves to another city
    - E.g., from Stuttgart to Cologne



# Slowly changing dimensions – Possible approaches

- Simply update record in dimension table
- Additional column for old/original value in dimension table
- Insert new record in the dimension table
- Versioning of dimension table records
- Versioning with time-attributes

# Simply update record in dimension table

- Advantage
  - ▶ Most simple solution
- Disadvantages
  - ▶ Different results for queries with a time range including the data of the change before and after the update
    - For instance the total revenue figures for Cologne and Stuttgart would change.
  - ▶ No change history is preserved.

# Simply update record in dimension table

- Example

Dimensionstabelle

	Kunde_ID	Kunde_Text	Kunde_Strasse	Kunde_Ort	Geschl	Kunde_Tel	LOADTIME
1.	1	Müller	Parkstrasse	Köln	M	1234	31DEC1999:23:03:08
	2	Meier	Schlossallee	München	W	4321	31DEC1999:23:03:08
	3	Schulz	Schillerstrasse	Berlin	W	5678	31DEC1999:23:03:08
	4	Schmitz	Berliner	Hamburg	M	8765	31DEC1999:23:03:08

	Kunde_ID	Kunde_Text	Kunde_Strasse	Kunde_Ort	Geschl	Kunde_Tel	LOADTIME
2.	1	Müller	Parkstrasse	Köln	M	1234	31DEC1999:23:03:08
	2	Meier	Schlossallee	München	W	4321	31DEC1999:23:03:08
	3	Schulz	Schillerstrasse	Berlin	W	5678	31DEC1999:23:03:08
	3	Schulz-Maier	Schillerstrasse	Berlin	W	5678	31JAN2000:23:05:04
	4	Schmitz	Berliner	Hamburg	M	8765	31DEC1999:23:03:08

	Kunde_ID	Kunde_Text	Kunde_Strasse	Kunde_Ort	Geschl	Kunde_Tel	LOADTIME
3.	1	Müller	Parkstrasse	Köln	M	1234	31DEC1999:23:03:08
	2	Meier	Schlossallee	München	W	4321	31DEC1999:23:03:08
	3	Schulz	Schillerstrasse	Berlin	W	5678	31DEC1999:23:03:08
	3	Schulz Maier	Schillerstrasse	Berlin	W	5678	31JAN2000:23:05:04
	3	Schulz-Maier	Goethestrasse	München	W	3333	28FEB2000:23:01:03
	4	Schmitz	Berliner	Hamburg	M	8765	31DEC1999:23:03:08

Quelle: Finger, R. (2002), Historisierungskonzepte, Vortrag im Rahmen der Seminarreihe „Data Warehouses und Data Marts – Effizienter Einsatz für das Controlling“, Frankfurt am Main 2002.

© Kemper, Mehanna, Unger: Business Intelligence, Vieweg 2004, ISBN 3-528-05802-1

# Additional column for old/original value in dimension table

- Update operation sets the old/original value as well
  - ▶ limited history information available
- Apart from that same advantages/disadvantages as simple update approach
  - ▶ Different results for queries with a time range including the data of the change before and after the update
    - For instance the total revenue figures for Cologne and Stuttgart would change.
  - ▶ No explicit change history

# Insert new record into the dimension table

- Is treated like a new dimension element
- Advantages
  - ▶ Simple operation
  - ▶ Old reports still return the same results
- Disadvantages
  - ▶ No relationship to old record
    - E.g., a customer that moves is treated as a new customer
    - Behavior cannot be observed over a longer period of time

# Versioning of dimension table records

- Insert new record in the dimension table
- Extension of the primary key of the dimension table by a version number
- Facts are referenced by the extended keys with the version number

**Produkt**

Data-Warehouse-Systeme; Architektur, Entwicklung, Anwendung; ISBN 3-932588-76-2

ANR_VNR	Artikel	Produktgruppe	Produktfamilie
1235-001	Quickphone 150	Singleband	Mobiltelefon
1235-002	Quickphone 150	Dualband	Mobiltelefon
1237-001	Quickphone 100	Singleband	Mobiltelefon
1239-001	Quickphone 200	Dualband	Mobiltelefon
:	:	:	:

**Faktentabelle**

ANR_VNR	Filial_ID	Zeit_ID	Verkäufe	Einkauf	Preis
1235-001	50013	02.03.1999	60	200	299,00
1237-001	50013	02.03.1999	31	150	599,00
1235-002	50013	05.03.1999	50	300	199,00
1237-001	50013	05.03.1999	20	120	99,00
1235-002	50013	06.03.1999	53	140	199,00
1239-002	50013	06.03.1999	35	134	99,00
124623-003	50013	07.03.1999	40	123	2499,00
:	:	:	:	:	:

# Versioning of dimension table records

- Example

Dimensionstabelle

Kunde_ID	Kunde_Text	Kunde_Strasse	Kunde_Ort	Geschl	Kunde_Tel	Curr	LOADTIME
1.01	Müller	Parkstrasse	Köln	M	1234	1	31DEC1999:23:03:08
2.01	Meier	Schlossallee	München	W	4321	1	31DEC1999:23:03:08
3.01	Schulz	Schillerstrasse	Berlin	W	5678	0	31DEC1999:23:03:08
3.02	Schulz-Maier	Schillerstrasse	Berlin	W	5678	0	31JAN2000:23:05:04
3.03	Schulz-Maier	Goethestrasse	München	W	3333	1	28FEB2000:23:01:03
4.01	Schmitz	Berliner	Hamburg	M	8765	1	31DEC1999:23:03:08

Welche Bestellmenge  
entfällt auf Schulz-Maier im  
Februar 2000?

Fakten-Tabelle

Kunde_ID	Bestell_ID	...	Bestellmenge	LOADTIME
1.01	1111111		1	31DEC1999:23:03:08
2.01	2222222		3	31DEC1999:23:03:08
3.01	3333333		4	31DEC1999:23:03:08
3.02	4444444		2	31JAN2000:23:05:04
3.03	5555555		4	28FEB2000:23:01:03
4.01	6666666		2	31DEC1999:23:03:08

Quelle: Finger, R. (2002), Historisierungskonzepte, Vortrag im Rahmen der Seminarreihe „Data Warehouses und Data Marts – Effizienter Einsatz für das Controlling“, Frankfurt am Main 2002.

© Kemper, Mehanna, Unger: Business Intelligence, Vieweg 2004, ISBN 3-528-05802-1

# Versioning of dimension table records

- Advantages
  - ▶ Old queries return the same results
  - ▶ Object identity preserved
    - Not a new dimension element
    - Only a new version of the dimension element
- Disadvantage
  - ▶ No explicit information on the time of the change

# Versioning with time-attributes

- Additional attributes for dimension table
  - ▶ Change-Timestamp
  - ▶ Valid from (timestamp)
  - ▶ Valid-to (timestamp)
- Valid-from column or version number is additional primary key of the dimension table
- Operations when updating
  - ▶ Update Valid-to column for currently valid dimension record and set it to to the new valid-from value
  - ▶ Insert new dimension record
  - ▶ Determine the facts in the fact table that are already related to the new dimension record and update the correspondig foreign key values by the primary keys of the new dimension record

# Versioning with time attributes

- Advantages
  - ▶ Old queries return the same results
  - ▶ Object identity preserved
    - Not a new dimension element
    - Only a new version of the dimension element
  - ▶ Full change history available
- Disadvantage
  - ▶ Complex update operation

# Versioning with time-attributes

- Example 1

Dimensionstabelle

1.	Kunde_ID	Kunde_Text	...	Kunde_Ort	gueltig_von	gueltig_bis	Curr	LOADTIME
	1.01	Müller		Köln	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	2.01	Meier		München	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	3.01	Schulz		Berlin	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	4.01	Schmitz		Hamburg	31DEC1999	31DEC9999	1	31DEC1999:23:03:08

2.	Kunde_ID	Kunde_Text	...	Kunde_Ort	gueltig_von	gueltig_bis	Curr	LOADTIME
	1.01	Müller		Köln	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	2.01	Meier		München	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	3.01	Schulz		Berlin	31DEC1999	14JAN2000	0	31DEC1999:23:03:08
	3.02	Schulz-Maier		Berlin	15JAN2000	31DEC9999	1	31JAN2000:23:05:04
	4.01	Schmitz		Hamburg	31DEC1999	31DEC9999	1	31DEC1999:23:03:08

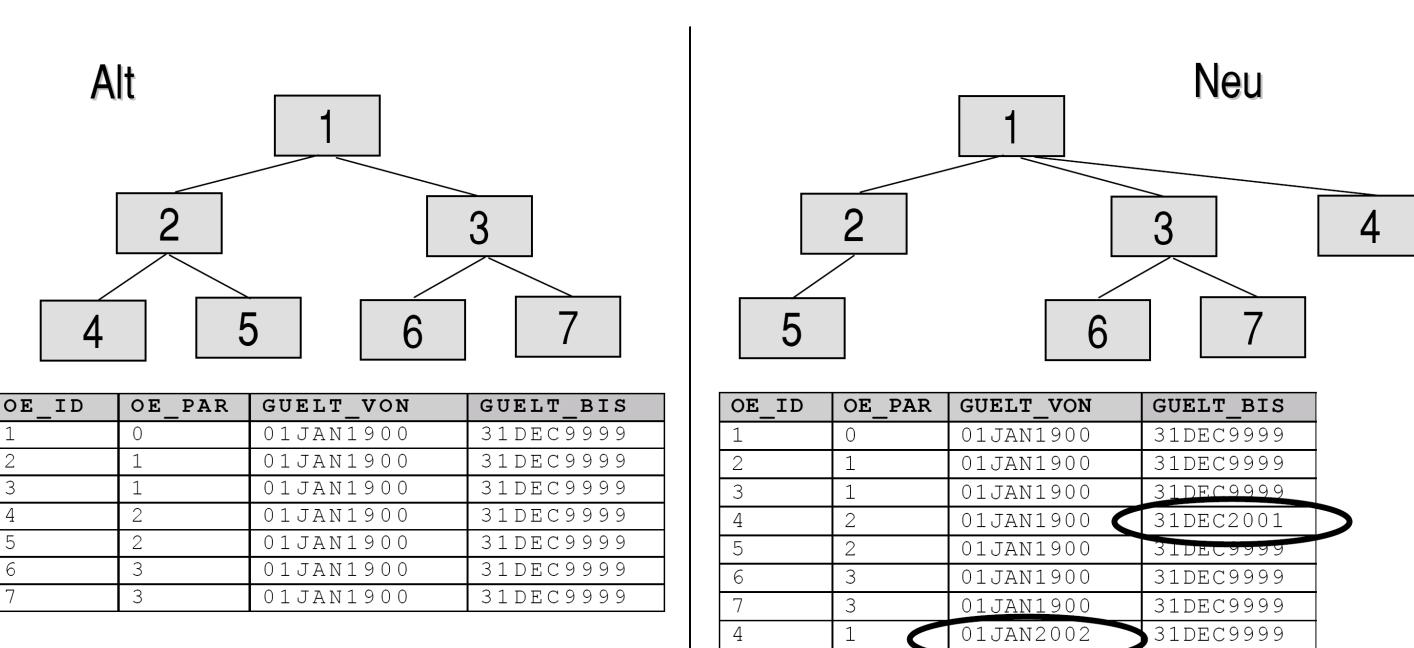
3.	Kunde_ID	Kunde_Text	...	Kunde_Ort	gueltig_von	gueltig_bis	Curr	LOADTIME
	1.01	Müller		Köln	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	2.01	Meier		München	31DEC1999	31DEC9999	1	31DEC1999:23:03:08
	3.01	Schulz		Berlin	31DEC1999	14JAN2000	0	31DEC1999:23:03:08
	3.02	Schulz-Maier		Berlin	15JAN2000	19FEB2000	0	31JAN2000:23:05:04
	3.03	Schulz-Maier		München	20FEB2000	31DEC9999	1	28FEB2000:23:01:03
	4.01	Schmitz		Hamburg	31DEC1999	31DEC9999	1	31DEC1999:23:03:08

Quelle: Finger, R. (2002), Historisierungskonzepte, Vortrag im Rahmen der Seminarreihe „Data Warehouses und Data Marts – Effizienter Einsatz für das Controlling“, Frankfurt am Main 2002.

© Kemper, Mehanna, Unger: Business Intelligence, Vieweg 2004, ISBN 3-528-05802-1

# Versioning with time-attributes

- Example 2



Quelle: Finger, R. (2002). Historisierungskonzepte, Vortrag im Rahmen der Seminarreihe „Data Warehouses und Data Marts – Effizienter Einsatz für das Controlling“. Frankfurt am Main 2002.

© Kemper, Mehanna, Unger: Business Intelligence, Vieweg 2004, ISBN 3-528-05802-1

# Aggregations

- Types of facts
  - ▶ Additive:
    - Can be summed up through all of the dimensions in the fact table.
    - Example: Retail data warehouse:
      - Dimensions: time, location, customer and product
      - Measure: sales amount
  - ▶ Semi-Additive:
    - Can be summed up for some of the dimensions in the fact table only.
      - Example: Banking data warehouse
        - Dimensions: time, account
        - Measure: current balance
  - ▶ Non-Additive:
    - Cannot be summed up for any of the dimensions present in the fact table.
      - Example: Retail data warehouse:
        - Dimensions: time, location, customer and product
        - Measure: profit margin
    - Can be computed from additive or semi-additive facts

# Aggregation function

- Have to be defined for each measure and dimension
- Sum is the most frequently used
- Other possible aggregation function
  - ▶ Count, Count Distinct
  - ▶ Average
  - ▶ Min, Max
  - ▶ ... Complex formula

| Data Warehouse, 2025

# Querying OLAP systems

ATVANTAGE



**TIMETOACT**  
**GROUP**

# MDX – OLAP Query Language

- MDX - Multidimensional Expressions
  - ▶ Developed by Microsoft, de-facto industry standard
  - ▶ SQL like syntax, language used to query OLAP databases
    - SELECT => specifies dimensions and members to be returned
    - FROM => specifies the source cube of the query
    - WHERE => restricts returned data to specific dimension and member criteria
  - ▶ Additional language elements
    - Scalar – data type „string“ or „number“
    - Dimension
    - Hierarchy
    - Level
    - Member
    - Tuple
    - Set

# MDX – Key concepts

- The result of an MDX query is itself a cube
  - ▶ The edges or dimensions in the result cube are called **axes**
  - ▶ The first three axes are called **columns**, **rows**, and **pages**
  - ▶ Columns and rows are often referred to as “axis dimensions”, pages as “slicer dimension”
- **Tuples** list dimensions and members to identify individual cells
  - ▶ If all dimensions are not specified, the tuple identifies a slice of the cube
  - ▶ Example:  
`( [Item].[Shirt], [Store].[Boston], [Measures].[Price] )`
- **Sets** are ordered collections of tuples
  - ▶ Example:  
`{ [Measures].[Price], [Measures].[Cost] }`
- **Calculated members** are members whose values are specified by an expression and are calculated at run time
  - ▶ Example:  
`[Measures].[Price] - [Measures].[Cost]`

# MDX – Sample Query

```
SELECT
    { [Measures].[Store Sales] } ON COLUMNS,
    { [Date].[2002], [Date].[2003] } ON ROWS

FROM Sales

WHERE ( [Store].[USA].[CA] )
```

This query defines the following result set information:

- The SELECT clause sets the query axes as the Store Sales Amount member of the Measures dimension, and the 2002 and 2003 members of the Date dimension.
- The FROM clause indicates that the data source is the Sales cube.
- The WHERE clause defines the "slicer axis" as the California member of the Store dimension.

# MDX – Simple Example

WITH

```
MEMBER [Measures].[Profit] AS '[Measures].[Price] - [Measures].[Cost]'
```

SELECT

```
{ ([Measures].[Price]), ([Measures].[Cost]), ([Measures].[Profit]) } on columns,
```

```
{ [Store].[Boston], [Store].[Ottawa] } on rows
```

FROM SalesCube

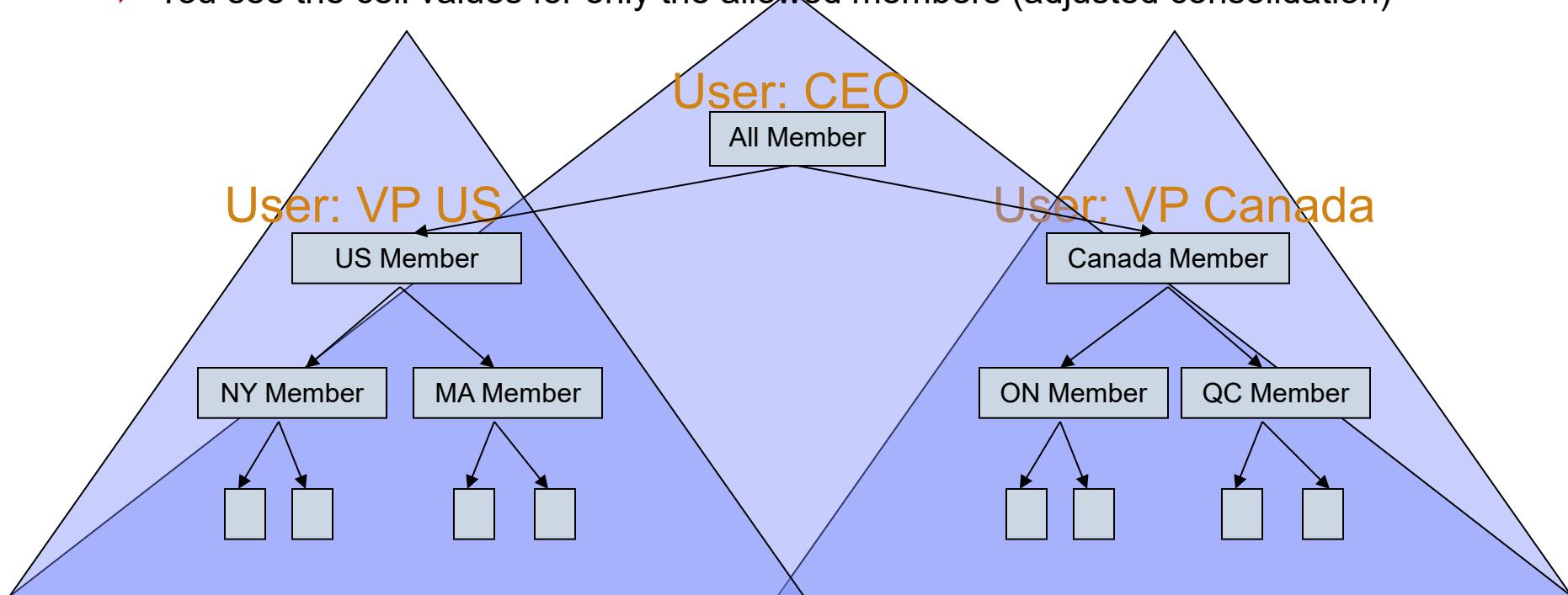
```
WHERE { [Item].[Pant] }
```

**Item: Pant**

Store	Measures		
	Price	Cost	Profit
Boston	\$50	\$40	\$10
Ottawa	\$150	\$80	\$70

# Security

- Security defined by members
- Typically hierarchy base. A user allowed to see a member and all its descendants
- OLAP database could apply security in different way
  - ▶ You see the cell values for the allowed members
  - ▶ You see the cell values for only the allowed members (adjusted consolidation)



| Data Warehouse, 2025

# Implementation of different OLAP systems



# Different OLAP Systems

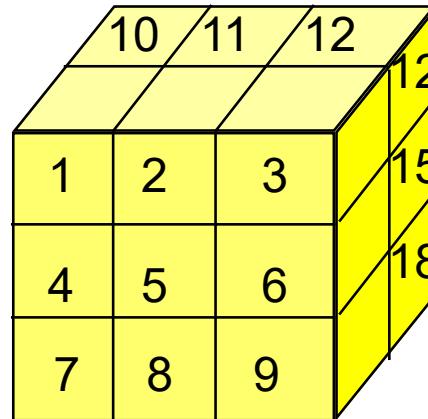
- MOLAP - Multidimensional OLAP
  - ▶ multidimensional structure on physical and logical level
    - multi-dimensional databases
- ROLAP – Relational OLAP:
  - ▶ relational database for better scalability
- HOLAP – Hybrid OLAP
  - ▶ uses multi-dimensional as well as relational databases
- DOLAP - Desktop OLAP
  - ▶ OLAP on a stand-alone desktop

# MOLAP - Multidimensional Databases

- A database specially designed to handle the organization of data in multiple dimensions
- Holds data cells in blocks that constitute a virtual cube
- Optimized to handle large amounts of numeric data
  - ▶ Index of descriptive names held separately from blocks of numeric data
  - ▶ Aggregated totals often precalculated
  - ▶ Not intended for textual data
- Commercial products
  - ▶ Oracle Essbase
  - ▶ IBM Planning Analytics / TM1

# Multidimensional storage

- Linearization of the cells in a cube into a one-dimensional array
- Depends on the order of dimensions
- Memory amount: #(dim1) x #(dim2) x ... x #(dimN) x cell-size
  - Depends on the cardinality of the dimensions only, not on the number of facts
- Example:
  - ▶ The numbers in the cube cells indicate the position in the array



# Example

- Cube with 3 dimensions
  - ▶ Product – 4 values – p1, p2, p3, p4
  - ▶ Store – 3 values – s1, s2, s3
  - ▶ Time (year) – 2 values - y1, y2
- Number of cells in the cube:  $4 \times 3 \times 2 = 24$
- Assignment of the cells in the array

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

# Example

- Sales in year y2

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

# Example

- Sales of store s1 in year y2

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

# Example

- Sales of product p2 in year y1

p1,s1,y1	p2,s1,y1	p3,s1,y1	p4,s1,y1	p1,s2,y1	p2,s2,y1	p3,s2,y1	p4,s2,y1
1	2	3	4	5	6	7	8
p1,s3,y1	p2,s3,y1	p3,s3,y1	p4,s3,y1	p1,s1,y2	p2,s1,y2	p3,s1,y2	p4,s1,y2
9	10	11	12	13	14	15	16
p1,s2,y2	p2,s2,y2	p3,s2,y2	p4,s2,y2	p1,s3,y2	p2,s3,y2	p3,s3,y2	p4,s3,y2
17	18	19	20	21	22	23	24

# Multidimensional storage

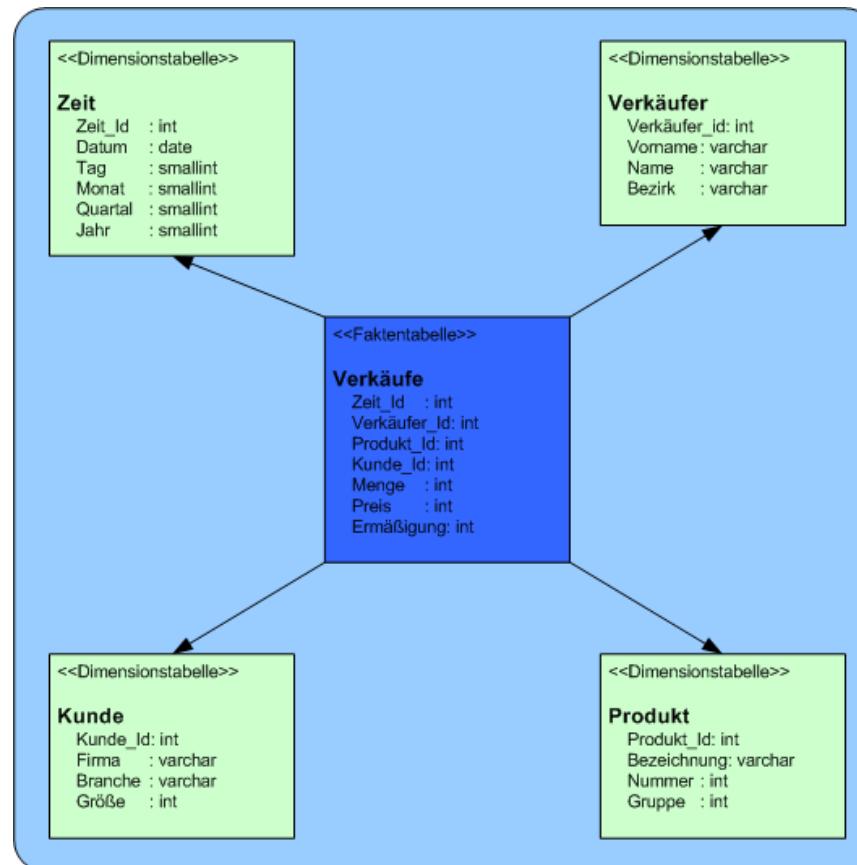
- Advantages
  - ▶ Efficient access to cube cells through simple index computations
  - ▶ Data relevant for one query can be in a small subrange of the array
    - Efficient processing of a query
- Disadvantages
  - ▶ Memory amount does not depend on the density of the cube
    - Null values occupy as well a cell in the cube
    - Solution:
      - Specific storage mechanism for sparse data
  - ▶ Storage array has to be recomputed if dimensions change
    - E.g.; a new product is added to product dimension

# ROLAP

- Physical data structure: relational tables
  - ▶ Advantage: can use well-engineered, reliable and high-performance database systems and query languages
- Special table structure
  - ▶ Star / Snowflake Schema
    - Dimension tables
    - Fact table consisting of
      - foreign keys to dimension tables
      - values for the measures
  - ▶ Memory amount depends mainly on the number of facts
- Aggregated totals are computed dynamically in general
  - ▶ Long response times
- Commerical products
  - ▶ MicroStrategy
  - ▶ IBM Cognos
  - ▶ Google Looker

# Star Schema

- Example from German Wikipedia



# ROLAP Enhancements

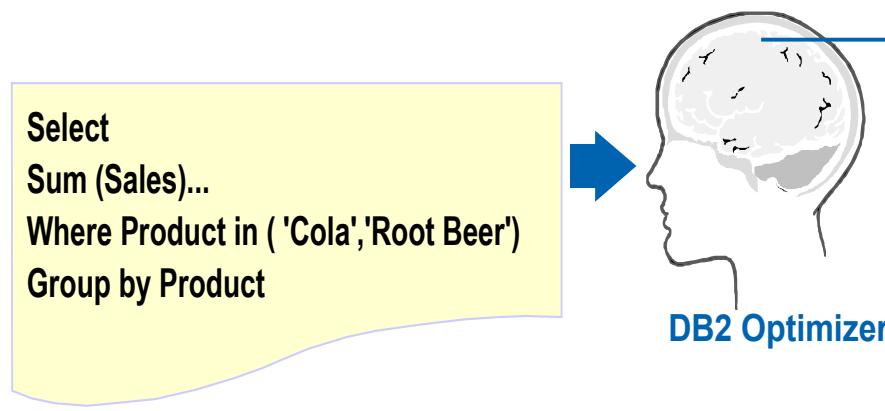
- Problems with storage of aggregated totals
  - ▶ Updates/Inserts are more complex
    - Aggregated values
      - have to be recomputed or
      - they become invalid
  - ▶ Query processing becomes more complex
    - SQL query corresponding to an OLAP request depends on which aggregated totals are precomputed
      - For instance, for computing the total revenue of a year it suffices to sum up
        - the values of every month if the total revenue per month has been precomputed
        - the values of every day if the total revenue per day has been precomputed

# ROLAP Enhancements

- Materialized Views / Query Tables
  - ▶ The DBMS takes charge of solving these problems
    - The user defines views containing aggregated values for certain hierarchy levels.
    - These views are materialized as tables.
      - Update options
        - immediate
        - deferred
    - When performing a query against a cube the DB optimizer takes advantage of these materialized views, i.e., no special queries have to be written for this by a user or application program.

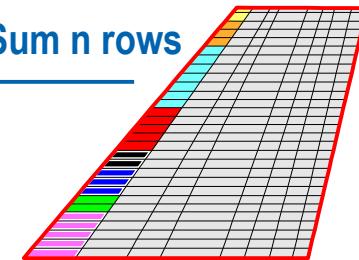
# Materialized Query Tables (MQTs) in IBM DB2

- Tables of Pre-Summarized Data
- Cost-Based DB2 Optimizer Routing
- Full and Partial Matching
- **Transparent** to Applications / Users



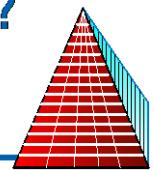
Base data?

Fetch & Sum n rows



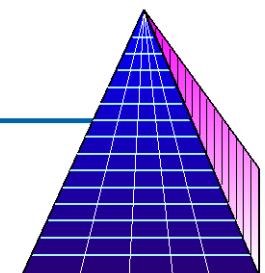
Partial Aggregate ?

Fetch & Sum



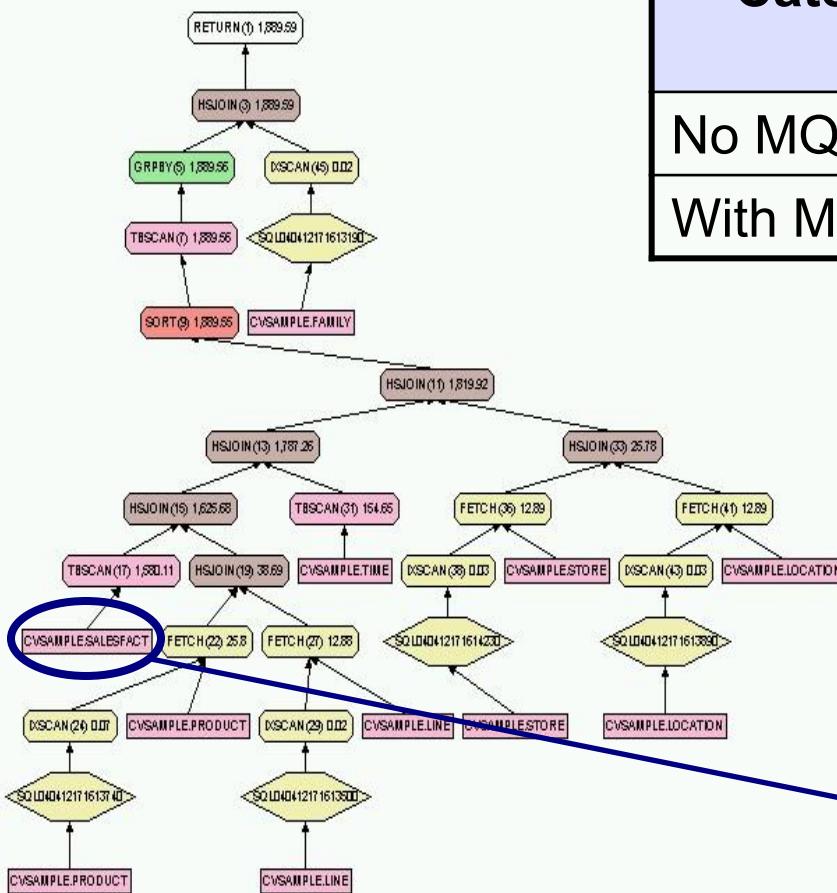
Full Aggregate ?

Simple Fetch



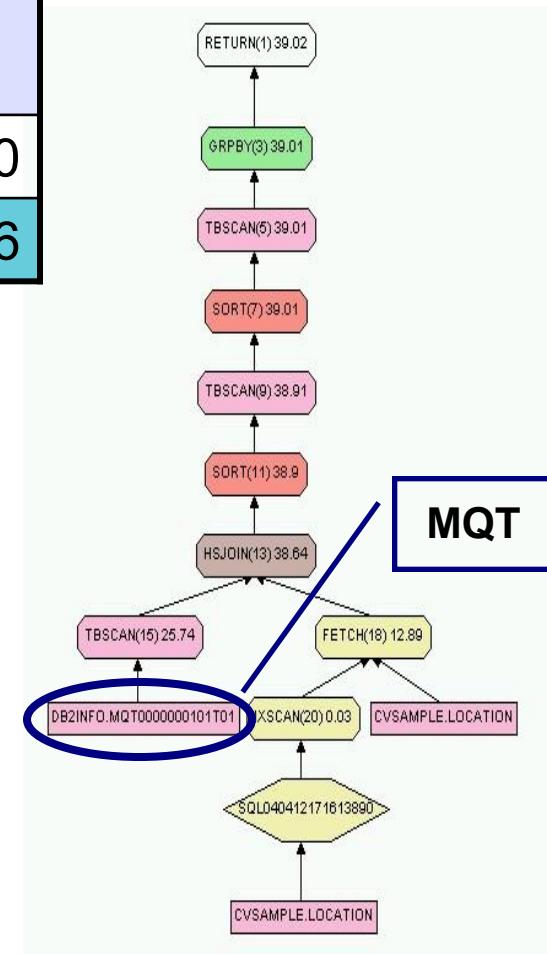
# Query Plan: Without and With MQT

No MQT



Category	Query Cost
No MQT	1889.5900
With MQT	39.0196

With MQT



Fact  
table

# MOLAP - ROLAP

	<b>MOLAP</b>	<b>ROLAP</b>
Database type	Multidimensional	Relational
Data storage	Special storage for cube data	Star schema - special multidimensional, relational data model
Size	100 s of Gigabytes	10s of Terabytes
Advantages	<ul style="list-style-type: none"> <li>■ special database products optimized for multidimensional analysis</li> <li>■ short response times</li> <li>■ suitable storage schema and query processing for multidimensional data</li> </ul>	<ul style="list-style-type: none"> <li>■ can use existing, well-established DBMS</li> <li>■ easy data import, update</li> <li>■ user access, backup, security mechanisms from DBMS can be used</li> </ul>

# MOLAP – ROLAP

	MOLAP	ROLAP
Disadvantages	<ul style="list-style-type: none"><li>■ problems with sparsity (ratio occupied / not occupied cells)</li><li>■ limited data volume</li><li>■ cube data accessible only</li><li>■ expensive update operation</li></ul>	<ul style="list-style-type: none"><li>■ complex SQL queries for processing OLAP requests → long response times</li></ul>

# HOLAP – Hybrid OLAP

- Combines the advantages of ROLAP and MOLAP
- Relational DBMS for storage of sparse, historic data
  - ▶ Data of highest granularity level
- Multidimensional DBMS for efficient storage of dense data cubes
  - ▶ Multidimensional cache for aggregated totals
- Complex architecture and maintenance processes
- No uniform OLAP query processing

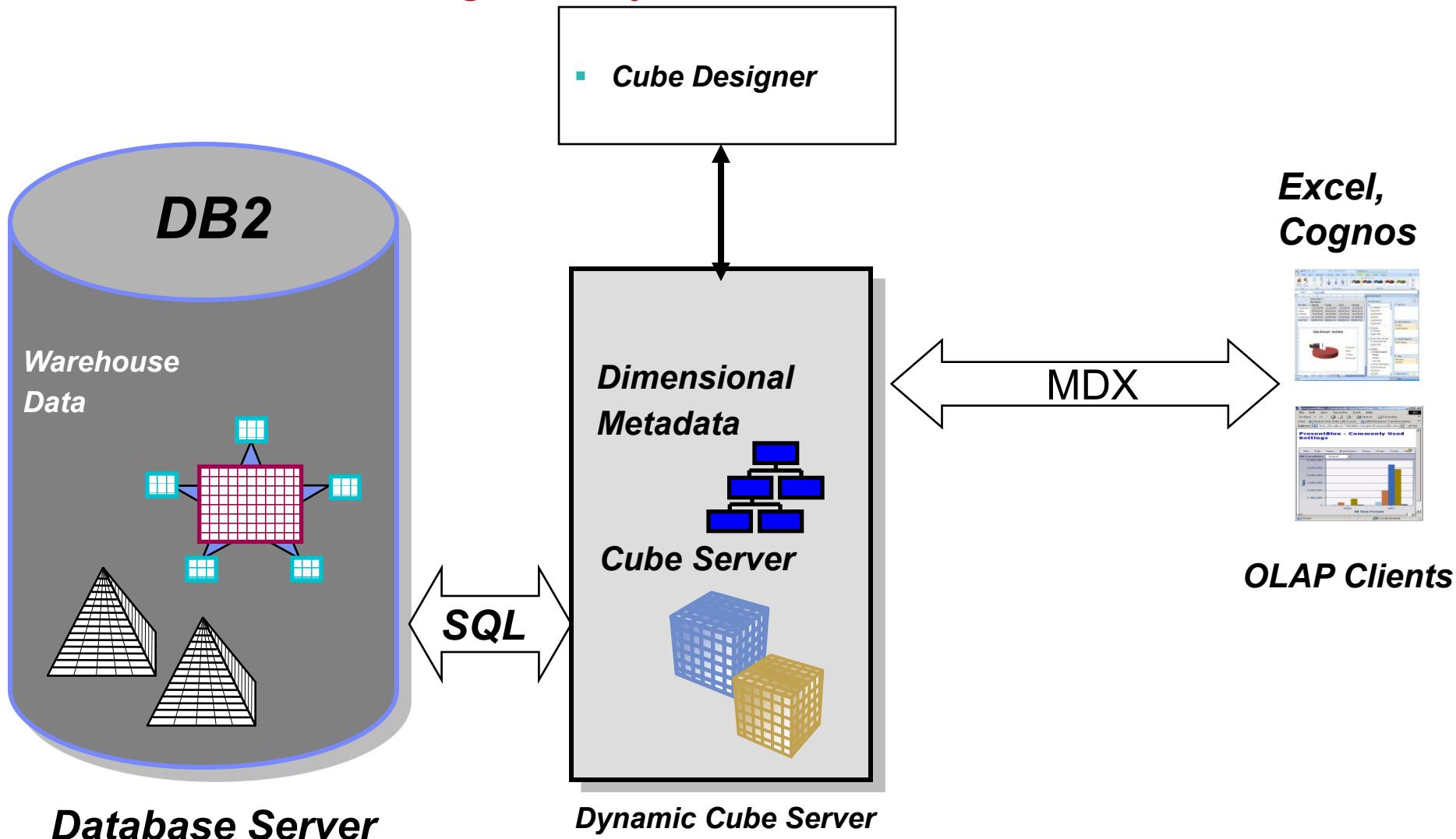
# DOLAP – Desktop OLAP

- Desktop solution for presentation of results with OLAP processor
- Data (and DB server) locally on each desktop
  - ▶ high redundancy
- Simple security concept
- Remote analysis possible
- Low purchase price
- Possibly high load on desktop
  - ▶ long response times, not very efficient
- Danger of "island solutions"
- Users can explore only a part of the data (due to limited storage capacity of desktop)
  - ▶ danger of wrong interpretation of data

# OLAP Engines

- Middleware between
  - ▶ Reporting/BI Frontend tool and
  - ▶ (relational or multidimensional) data store
- Provide a logical multidimensional view on OLAP cubes independantly of their storage scheme
- Holds OLAP metadata (dimensions, hierarchies, measures, ...)
- Usually support MDX through corresponding application programming interfaces
  - ▶ ODBO - OLE DB for OLAP
  - ▶ XMLA – XML for Analysis

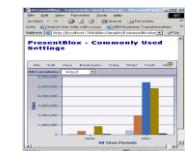
# ROLAP – IBM Cognos Dynamic Cubes Architecture



**Database Server**

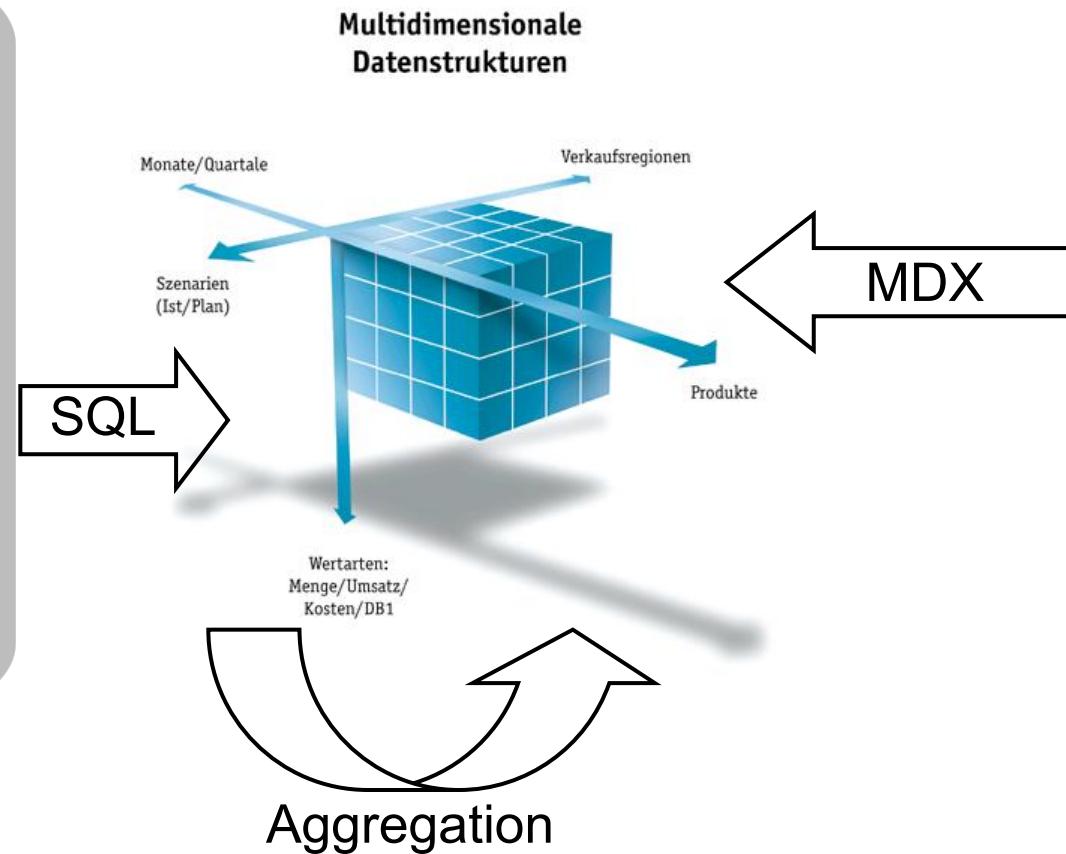
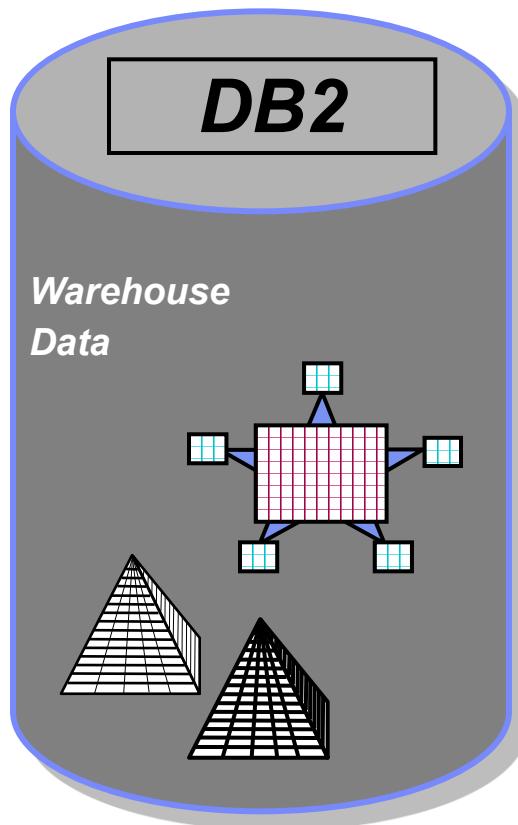
**Dynamic Cube Server**

**Excel,  
Cognos**



**OLAP Clients**

# MOLAP – IBM Cognos TM1



**Excel,  
Cognos**



# Writeback

- Some OLAP systems allow users to interactively change data values in cells
- Effect of data changes on aggregations and calculations can usually be seen in real-time
- Key feature for budgeting/planning and what-if applications
- Writeback may be:
  - ▶ Simple data entry:
    - e.g. user types into a leaf cell
  - ▶ Spreading/break-back
    - e.g. user changes a consolidated value, and the change is propagated to leaf cells
    - User specifies how the value is to be propagated
      - For example, split equally between all children, split proportionally to current values, etc.
    - Constraints may be added by “holding” certain cell values
      - For example, when a new value is entered for the full year, propagate that to the months but do not change (i.e., hold) the value for June

# OLAP – Strengths and Weaknesses

## ▪ Strengths

- ▶ Fast:
  - Interactive exploration of the data
  - Quickly render dashboard or report involving multiple complex elements
- ▶ Analysis:
  - Easy to add business calculations
  - Easy to add common business functionality (i.e. top 10 customers, time series, etc.)
  - Write back and what if analysis
- ▶ Dimension, hierarchy and measure: Provide common way to see business information

## ▪ Weaknesses

- ▶ Hard to deal with data stream or real time data – latency on cube cell values updates
- ▶ Very large models or applications requiring a very large number of intermediate aggregations may need to rely on pre-calculation for performance, which may lead to data explosion
- ▶ ROLAP and HOLAP try to minimize the cube size explosion, but still hard to deal with large dimensions
- ▶ Require different IT skill set to manage than traditional relational databases

# Reporting Model vs OLAP Model

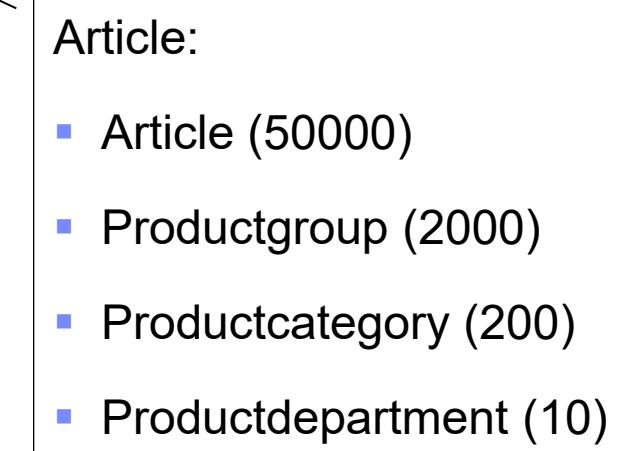
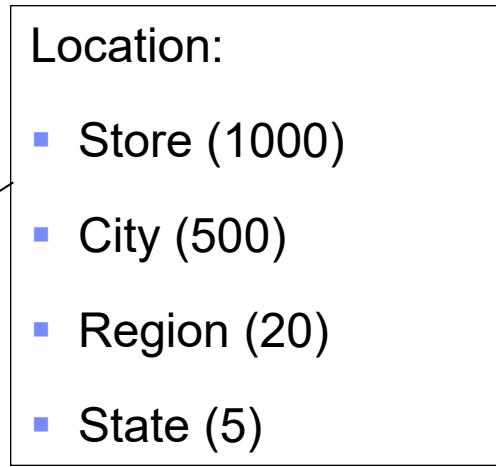
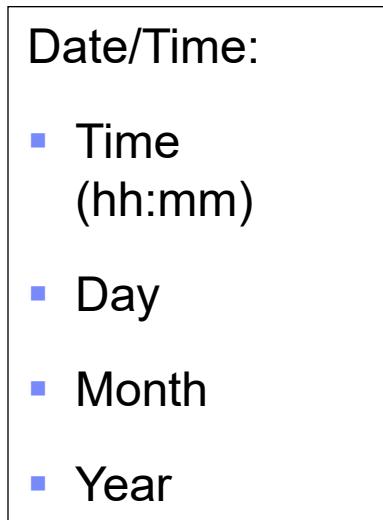
Reporting Model	OLAP Model
Column or item	Level and property
Group or folder	Dimension
Navigation path	Hierarchy
Fact	Measure
Value	Member and cell value
Detail filter	Set or member selection

# Exercise 1

- List benefits and drawbacks of ROLAP and MOLAP from
  - ▶ an end-user
  - ▶ an IT-manager
- perspective.

## Exercise 2

- The following is a data model used by a supermarket chain to analyze their business:



## Exercise 2

- With each transaction, an average of 20 different articles are bought.
- The data warehouse collects sales transactions data over 2 years.
- There are 1000 stores with 2000 transactions per store and day.
- Questions:
  1. How many records are stored in the fact table?
  2. What are the columns of the ROLAP fact table?
  3. What is the size of the cube (number of cells) that stores the aggregated values
    - at the most detailed level?
    - on day, article and store level?
  4. Compute the respective cube sizes for the other 3 (higher) hierarchy levels (Month, Year, City, Region, State, Productgroup, Productcategory, Productdepartment).