
Lecture 07

Data Warehouse Architecture (cont'd.)

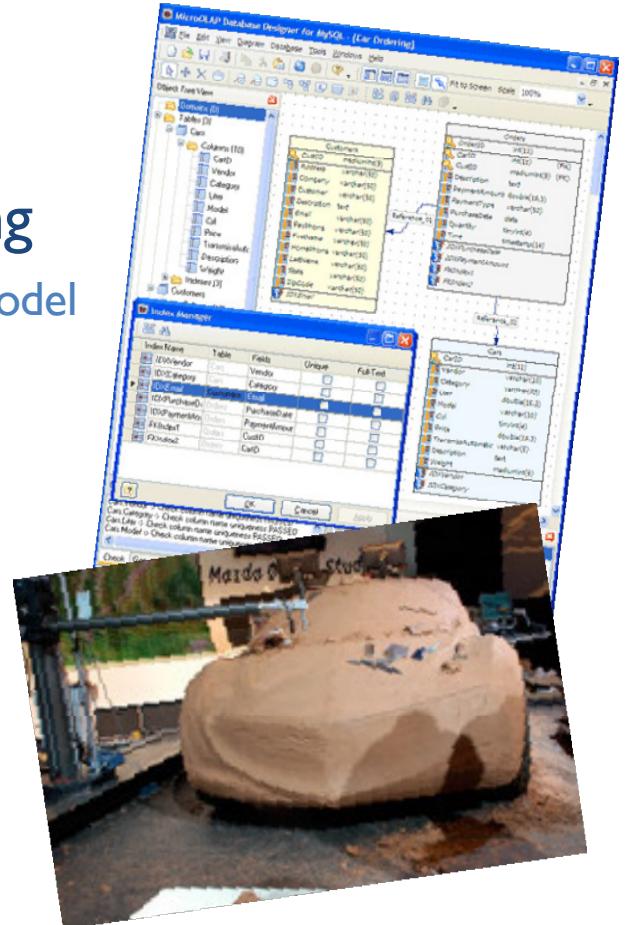
Logical Model

Summary – last week

- Last week:
 - Distributed DWH
 - Data Modeling – Conceptual Modeling
 - Multidimensional Entity Relationship (ME/R) Model

- This week:
 - Logical Model

Summary

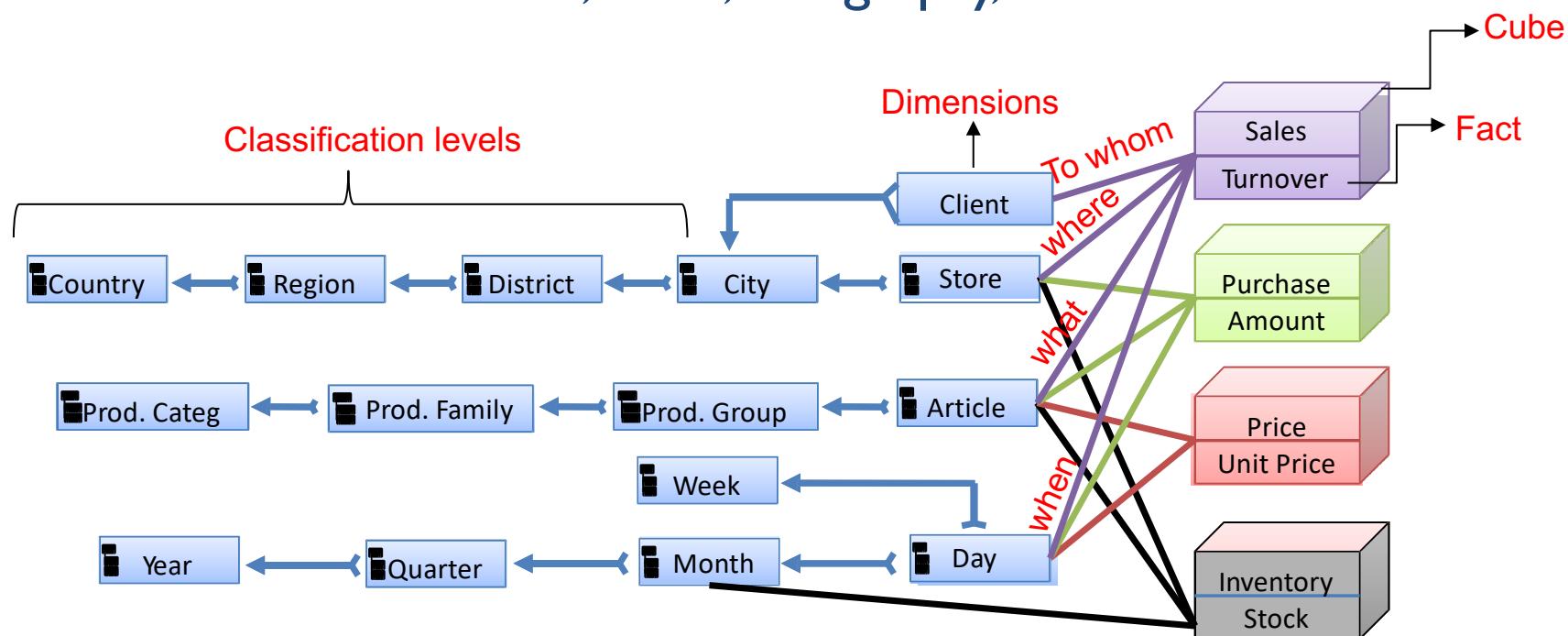


Logical Model

- Goal of the Logical Model
 - Confirm the **subject areas** e.g. sales or inventory
 - Create ‘real’ **facts** and **dimensions** from the subjects that we have identified. **Facts examples** are how much I sold? How much profit I got? While **dimensions examples** are when I did sell this item? or where I did sell this?
 - Establish the needed **granularity** for our dimensions e.g. time dimension can go for years → months → weeks → days etc.

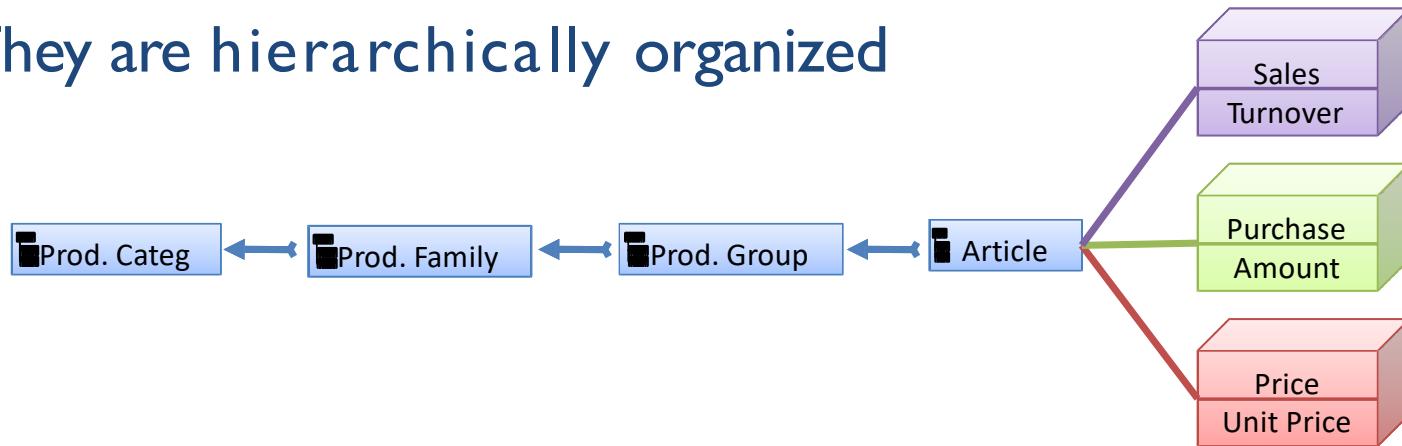
Logical Model (cont'd.)

- **Logical structure** of the multidimensional model
 - Cubes: Sales, Purchase, Price, Inventory
 - Dimensions: Product, Time, Geography, Client



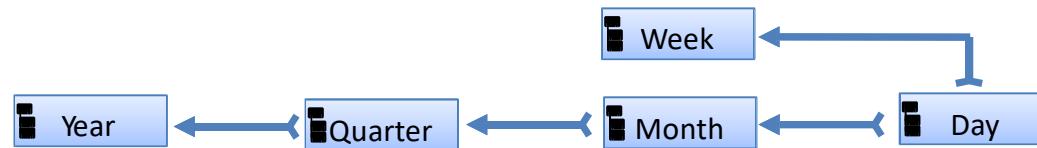
Dimensions

- **Dimensions are...**
analysis purpose chosen **entities**,
within the data model
 - One dimension can be used to define **more than one cube**
 - They are hierarchically organized



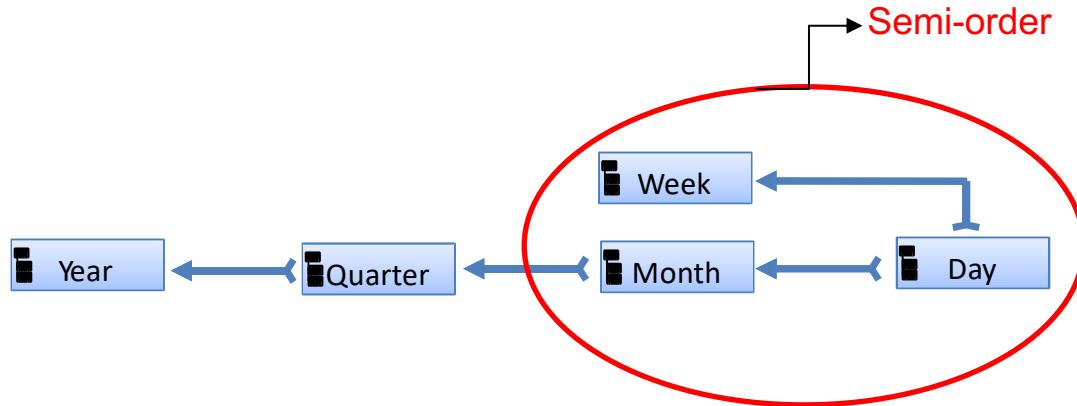
Dimensions (cont'd.)

- Dimension hierarchies are organized in **classification levels** (e.g., Day, Month, ...)
 - The dependencies between the classification levels are described by the **classification schema** through **functional dependencies**
 - An attribute B is functionally dependent on an attribute A, denoted $A \rightarrow B$, if for all $a \in \text{dom}(A)$ there exists exactly one $b \in \text{dom}(B)$ corresponding to it



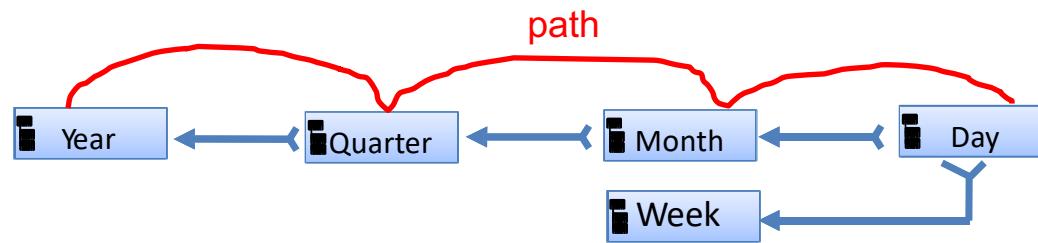
Dimensions (cont'd.)

- Classification schemas
 - The classification schema of a dimension D is a semi-ordered set of **classification levels** $(\{D.K_0, \dots, D.K_k\}, \rightarrow)$
 - With a **smallest element** $D.K_0$, i.e. there is no classification level with smaller granularity



Dimensions (cont'd.)

- A **fully-ordered** set of classification levels is called a **Path**
 - If we consider the **classification schema** of the time dimension, then we have the following paths
 - T.Day → T.Week
 - T.Day → T.Month → T.Quarter → T.Year



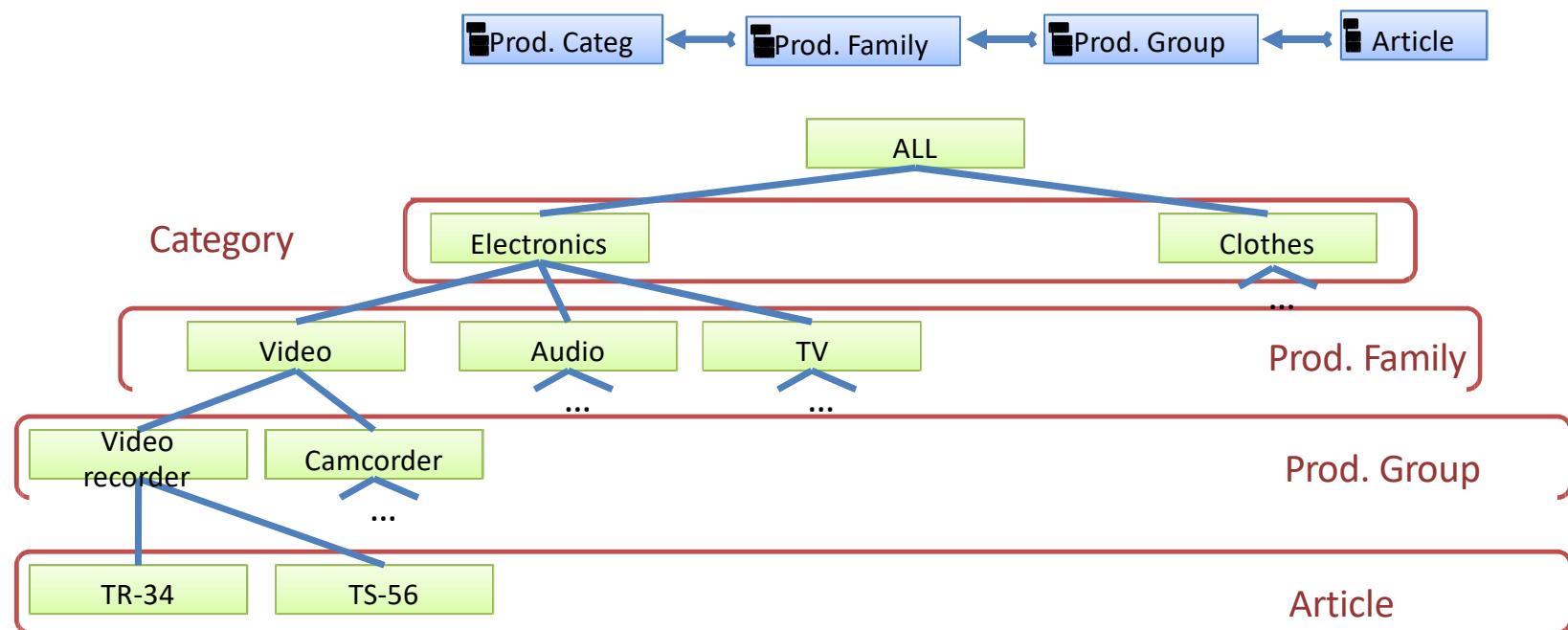
- Here T.Day is the **smallest element**

Dimensions (cont'd.)

- Classification hierarchies
 - Let $D.K_0 \rightarrow \dots \rightarrow D.K_k$ be a path in the classification schema of dimension D
 - A classification hierarchy concerning these path is a balanced tree which
 - Has as nodes $\text{dom}(D.K_0) \cup \dots \cup \text{dom}(D.K_k) \cup \{\text{ALL}\}$
 - And its edges respect the functional dependencies

Dimensions (cont'd.)

- **Example:** classification hierarchy from the path product dimension

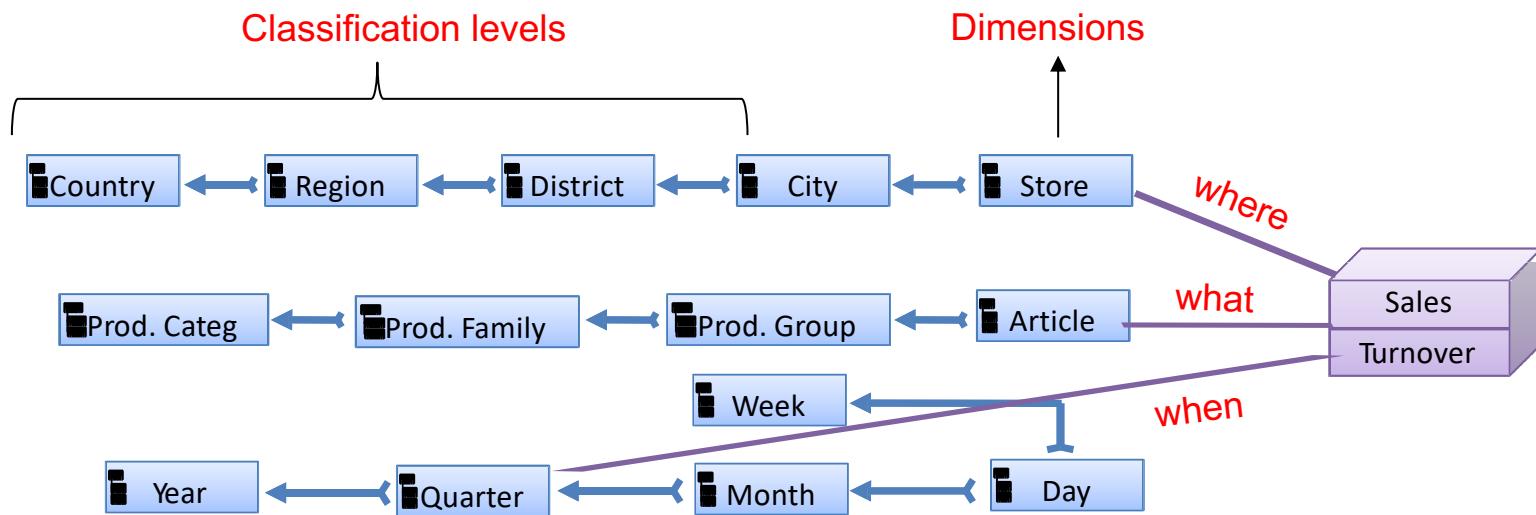


Cubes

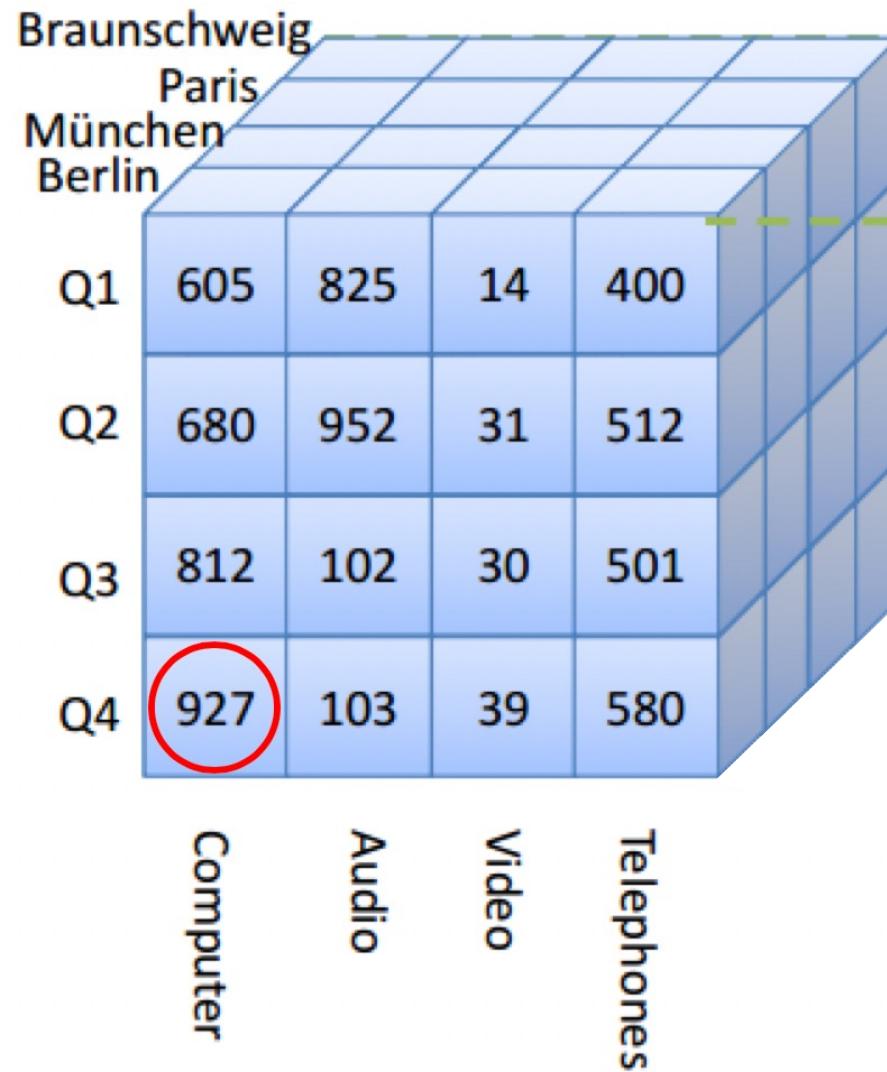
- **Cube** represent the basic unit of the multidimensional paradigm
 - They store one or more **measures** (e.g. the turnover for sale) in **raw or pre-aggregated form**
- More formally a cube C is a set of cube cells $C \subseteq \text{dom}(G) \times \text{dom}(M)$, where $G = (D_1.K_1, \dots, D_n.K_n)$ is a set of **granularities**, $M = (M_1, \dots, M_m)$ the set of **measures**
 - E.g. Sales((Store, Article, Quarter), (Turnover))

Cubes (cont'd.)

- The **coordinates** of a cell are the classification nodes from $\text{dom}(G)$ corresponding to the cell
 - Sales ((Store, Article, Quarter), (Turnover))
 - ...

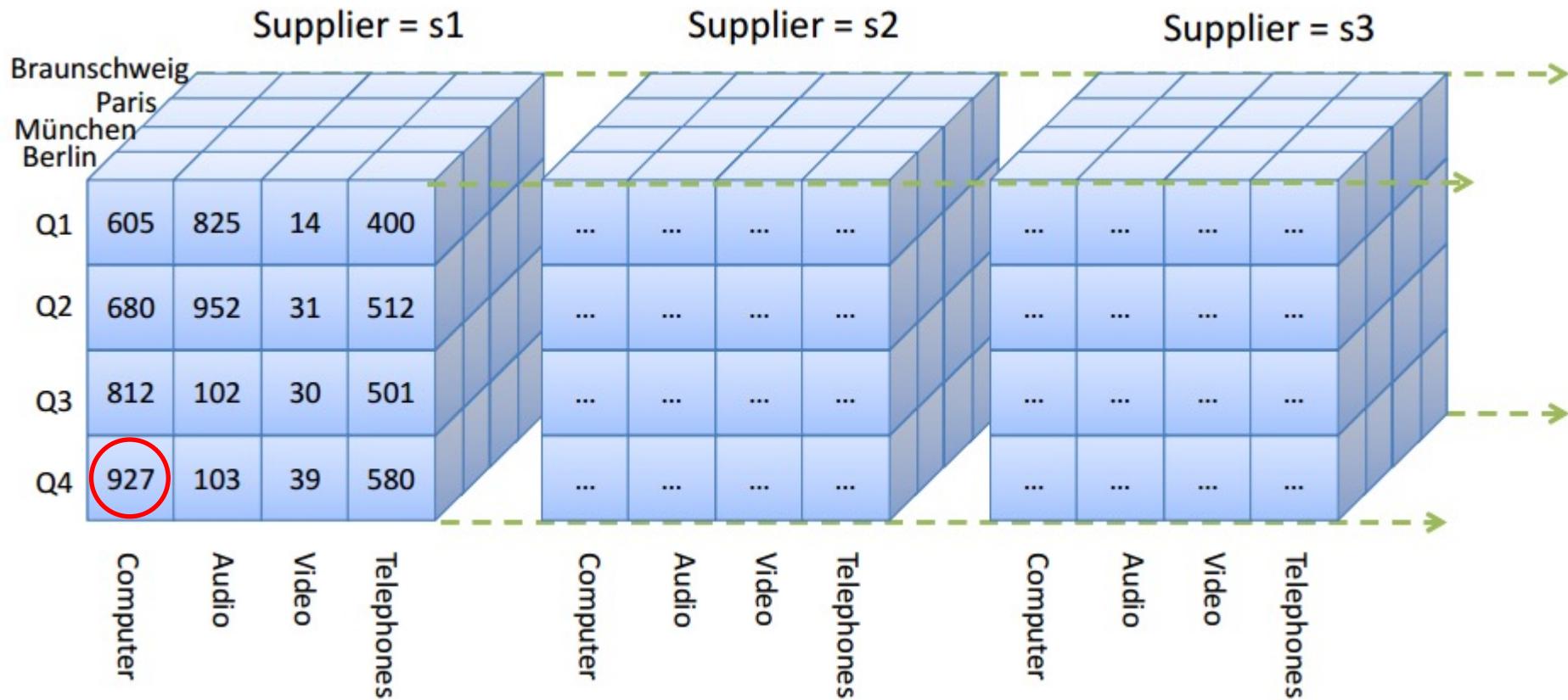


Cubes (cont'd.)



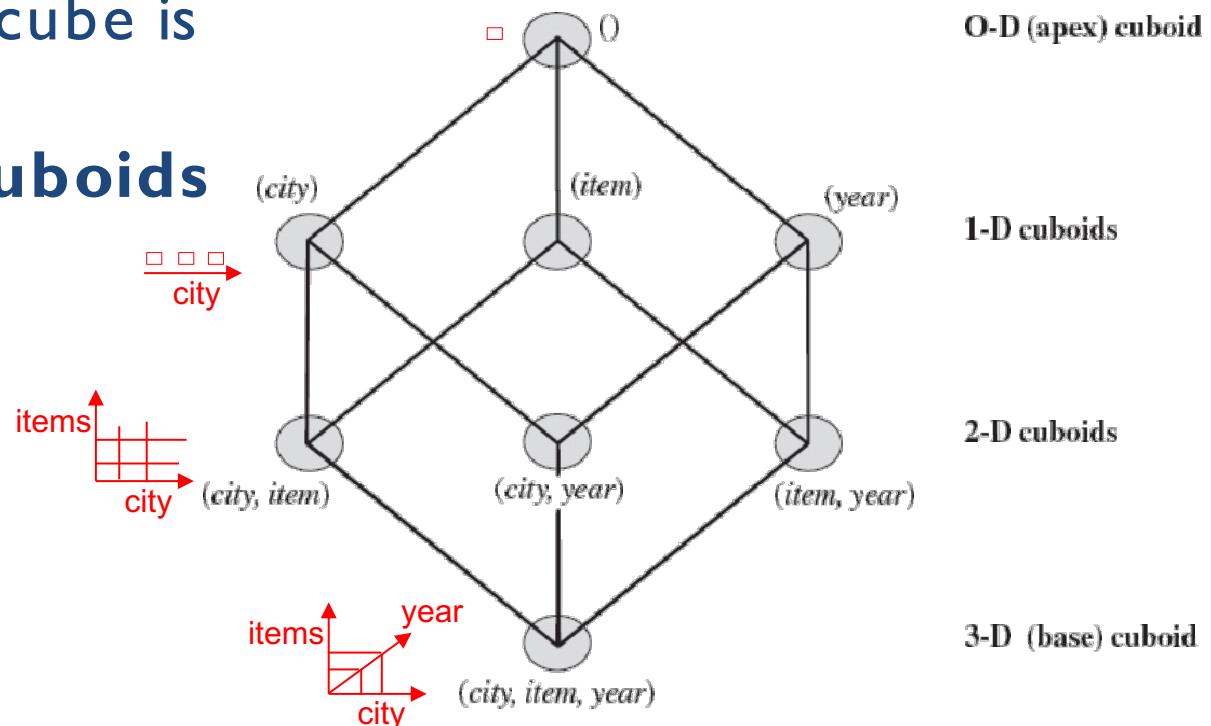
Cubes (cont'd.)

- 4 dimensions (supplier, city, quarter, product)



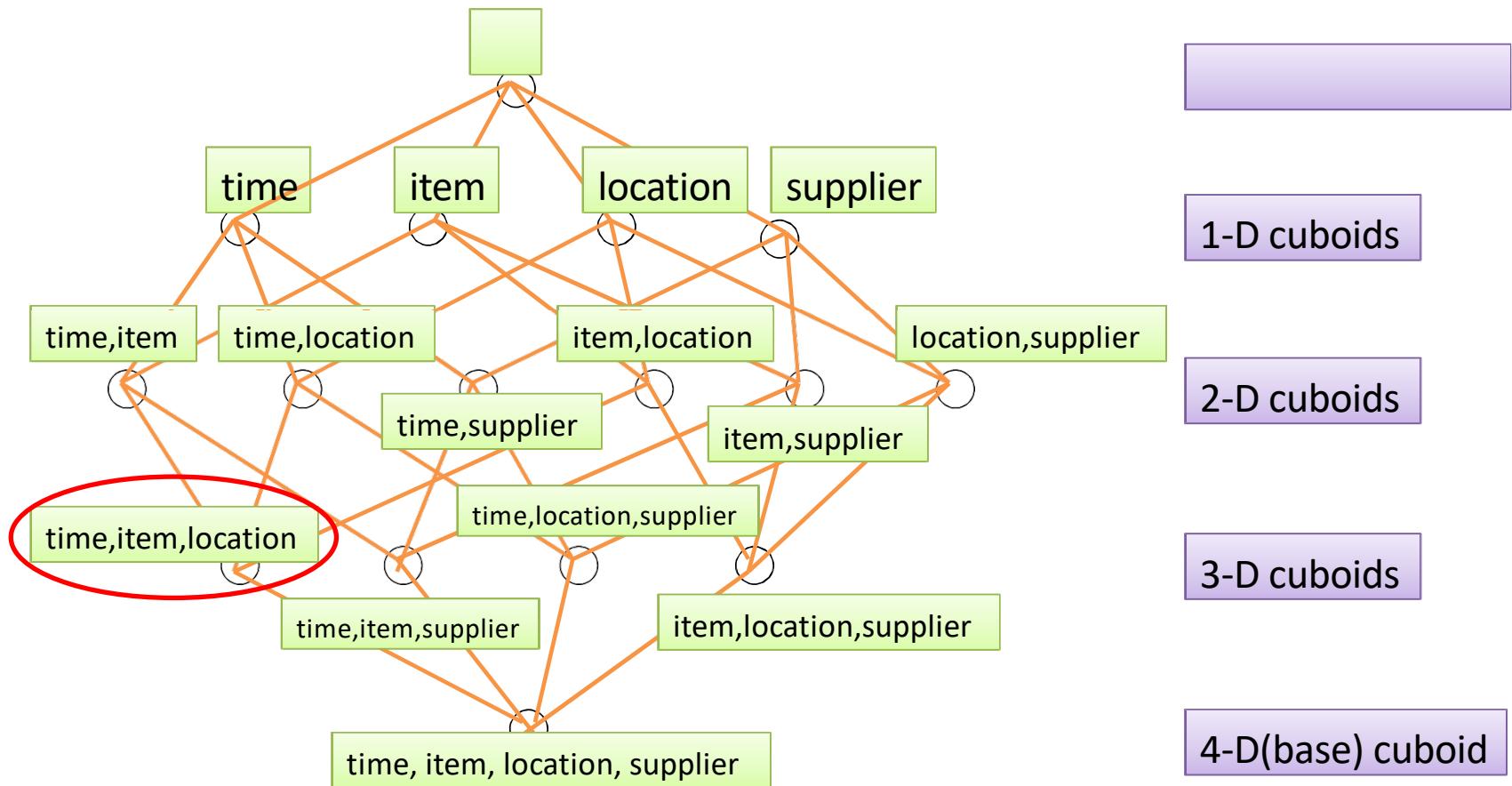
Cubes (cont'd.)

- We can now imagine **n-dimensional cubes**
 - n-D cube is called a **base cuboid**
 - The top most cuboid, the 0-D, which holds the highest level of summarization is called **apex cuboid**
 - The full data cube is formed by the **lattice of cuboids**



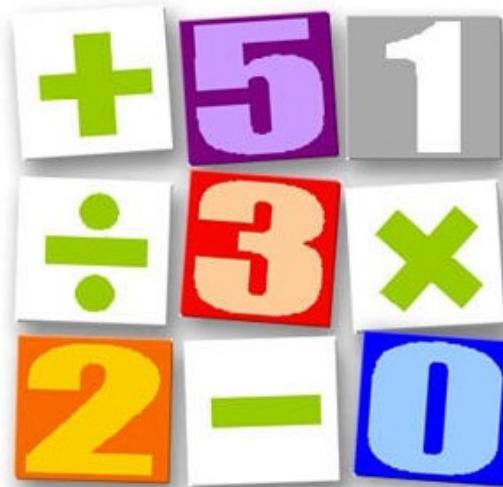
Cubes (cont'd.)

- But things can get complicated pretty fast



Basic Operations

- Basic operations of the multidimensional model on the logical level
 - Selection
 - Projection
 - Cube join
 - Sum
 - Aggregation

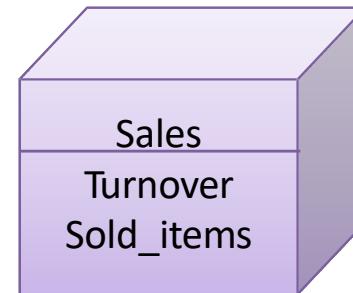


Basic Operations (cont'd.)

- Multidimensional Selection
 - The **selection** on a cube $C((D_1.K_1, \dots, D_g.K_g), (M_1, \dots, M_m))$ through a predicate P , is defined as $\sigma_P(C) = \{z \in C : P(z)\}$, if all variables in P are either:
 - Classification levels K , which functionally depend on a classification level in the granularity of K , i.e. $D_i.K_i \rightarrow K$
 - Measures from (M_1, \dots, M_m)
 - E.g. $\sigma_{P\text{Prod_group}=\text{"Video"}}(\text{Sales})$

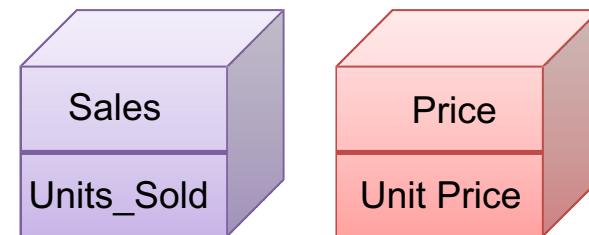
Basic Operations (cont'd.)

- Multidimensional projection
 - The **projection** of a function of a measure $F(M)$ of cube C is defined as
 - $m_{F(M)}(C) = \{ (g, F(m)) \in \text{dom}(G) \times \text{dom}(F(M)): (g, m) \in C \}$
 - E.g. projection $m_{\text{turnover, sold_items}}(\text{Sales})$



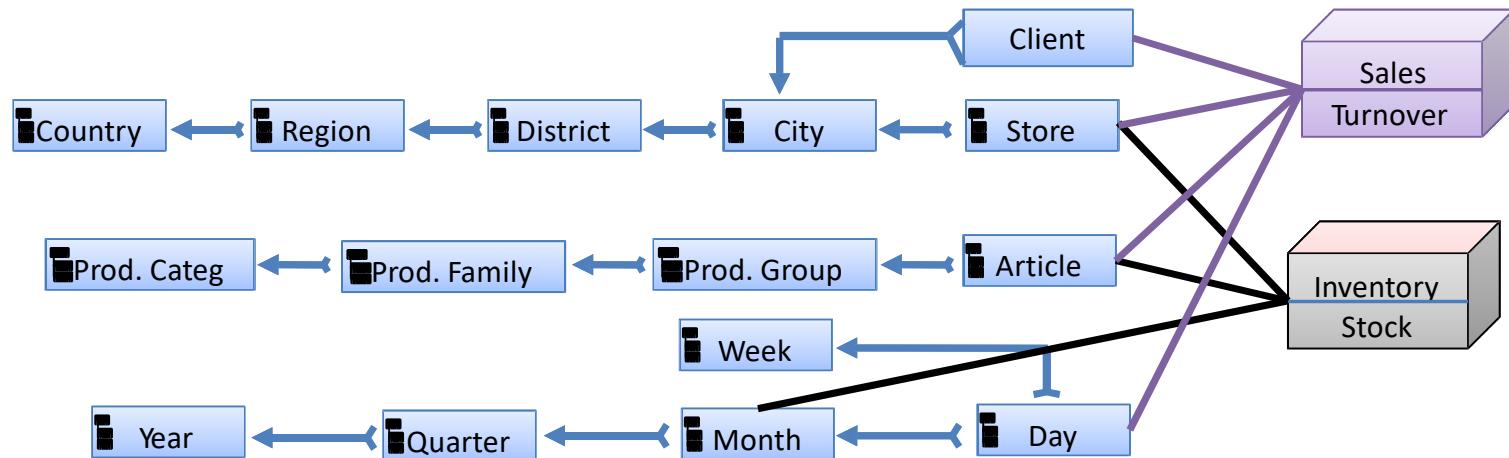
Basic Operations (cont'd.)

- Join operations between cubes is usual
 - E.g. if turnover would not be provided, it could be calculated with the help of the unit price from the price cube
- 2 cubes $C_1(G_1, M_1)$ and $C_2(G_2, M_2)$ can only be joined, if they have the same granularity ($G_1 = G_2 = G$)
 - $C_1 \bowtie C_2 = C(G, M_1 \cup M_2)$



Basic Operations (cont'd.)

- When the granularities are different, but we still need to join the cubes, **aggregation** has to be performed
 - E.g., Sales \bowtie Inventory: aggregate Sales((Day, Article, Store, Client)) to Sales((Month, Article, Store, Client))



Basic Operations (cont'd.)

- **Aggregation:** most important operation for OLAP operations
- Aggregation functions
 - Build a single values from set of value, e.g. in SQL: SUM, AVG, Count, Min, Max
 - Example: $\text{SUM}_{(\text{P.Product_group}, \text{G.City}, \text{TMonth})}(\text{Sales})$

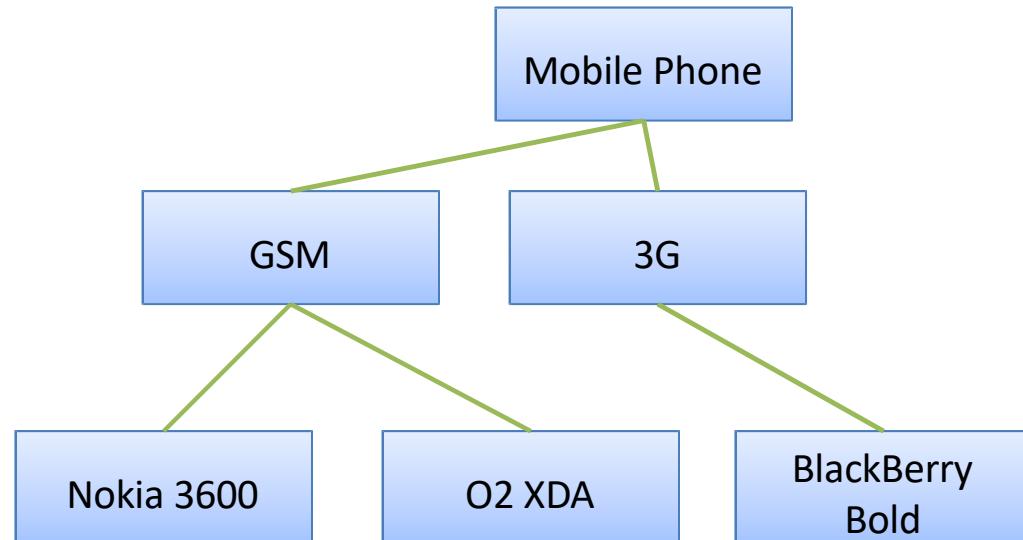
Change Support

- Classification schema, cube schema, classification hierarchy are all designed in the building phase and **considered as fix**
 - Practice has proven otherwise DW grow old, too
 - Changes are strongly connected to the time factor
 - This lead to the time validity of these concepts
- Reasons for schema modification
 - New requirements
 - Modification of the data source



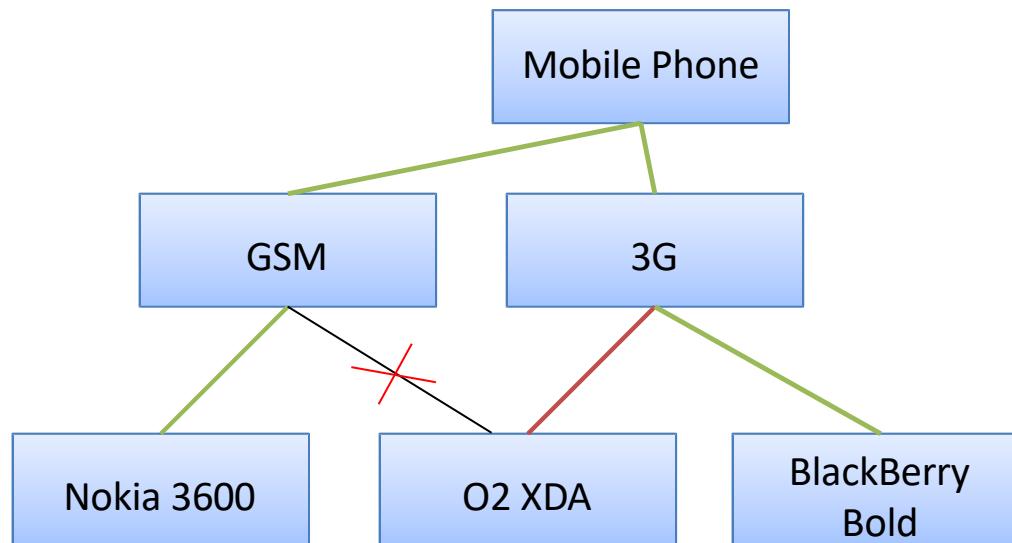
Classification Hierarchy

- E.g. Saturn sells a lot of electronics
 - Lets consider mobile phones
 - They built their DW on 01.03.2003
 - A classification hierarchy of their data until 01.07.2008 could look like this:



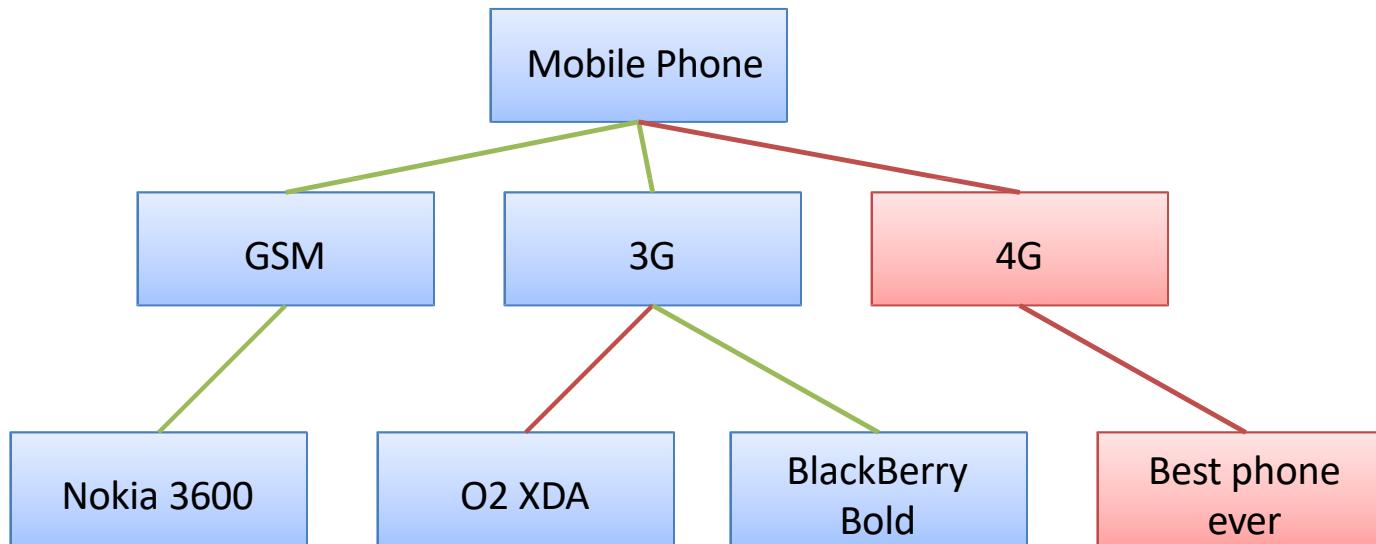
Classification Hierarchy (cont'd.)

- After 01.07.2008 3G becomes hip and affordable and many phone makers start migrating towards 3G capable phones
 - Lets say O2 makes its XDA 3G capable



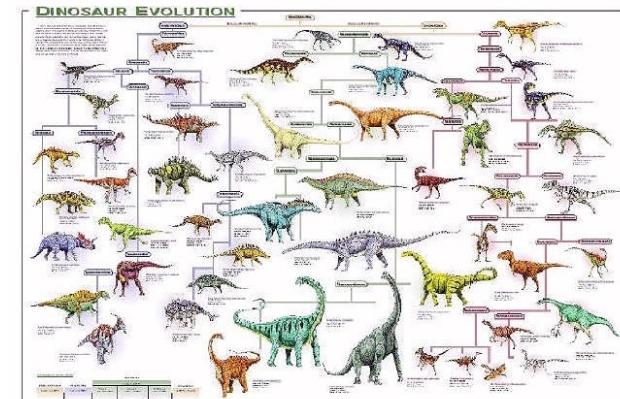
Classification Hierarchy (cont'd.)

- After 01.04.2010 phone makers already develop 4G capable phones



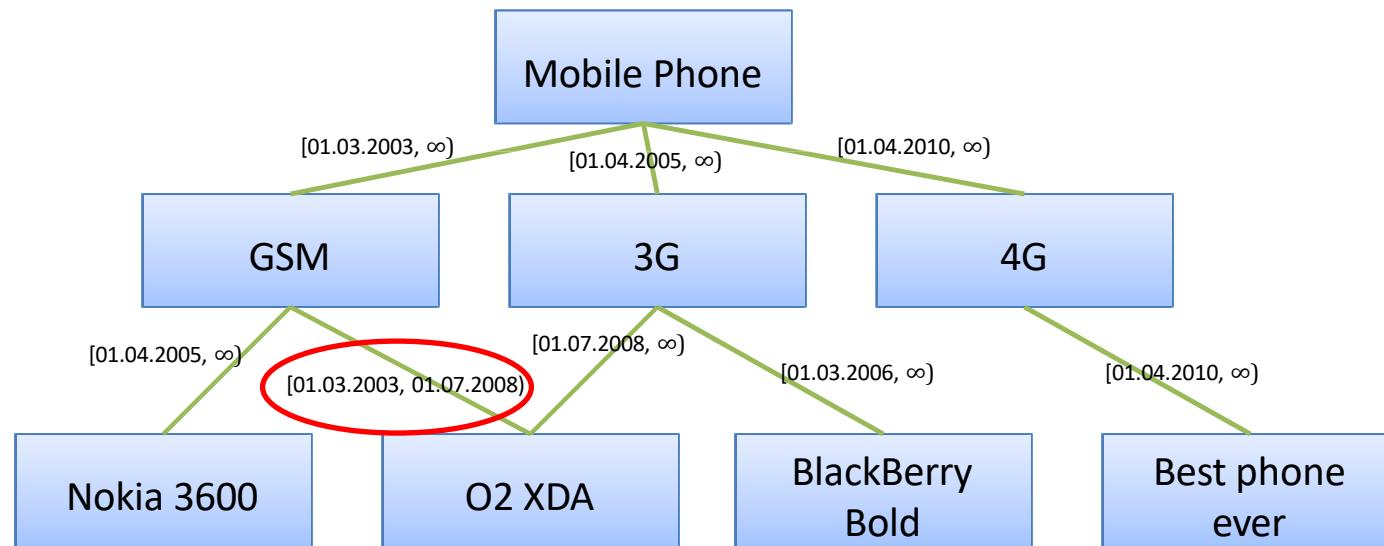
Classification Hierarchy (cont'd.)

- It is important to **trace the evolution** of the data
 - It can explain which data was available at which moment in time
 - Such a **versioning system** of the classification hierarchy can be performed by constructing a **validity matrix**
 - When is something valid?
 - Use **timestamps** to mark it!



Classification Hierarchy (cont'd.)

- Annotated Change data



Classification Hierarchy (cont'd.)

- The tree can be stored as dimension metadata
 - The storage form is a **validity matrix**
 - Rows are parent nodes
 - Columns are child nodes

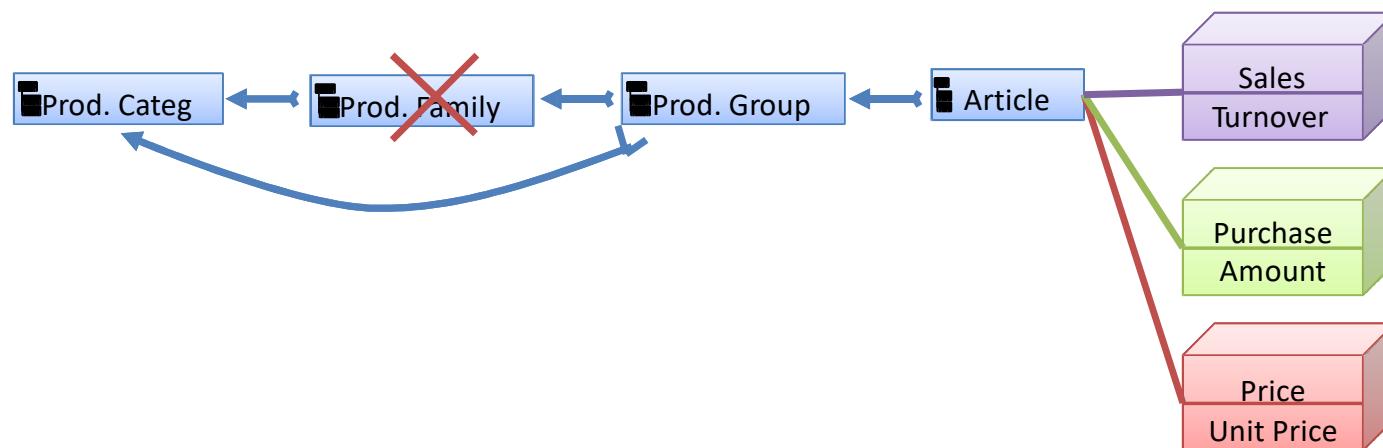
	GSM	3G	4G	Nokia 3600	O2 XDA	Berry Bold	Best phone
Mobile phone	[01.03.2003, ∞)	[01.04.2005, ∞)	[01.04.2010, ∞)				
GSM				[01.04.2005, ∞)	[01.03.2003, 01.07.2008) [01.07.2008, ∞)		
3G						[01.03.2006, ∞)	
4G							[01.04.2010, ∞)
Nokia 3600							
O2 XDA							
Berry Bold							
Best phone							

Schema Modification

- Improper modification of a schema (deleting a dimension) can lead to
 - Data loss
 - Inconsistencies
 - Data is incorrectly aggregated or adapted
- Proper schema modification is complex but
 - It brings flexibility for the end user
 - The possibility to ask “As Is vs. AsWas” queries and so on
- Alternatives
 - Schema evolution
 - Schema versioning

Schema Modification (cont'd.)

- Schema evolution
 - Modifications can be performed **without data loss**
 - It involves schema modification and **data adaptation** to the new schema
 - This data adaptation process is called **Instance adaptation**



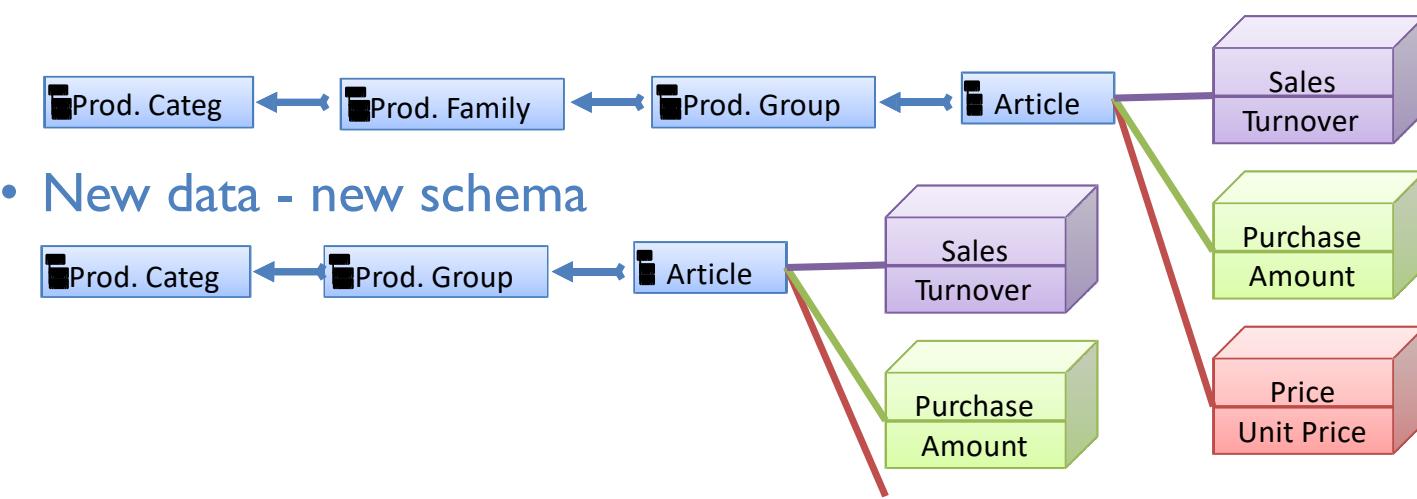
Schema Modification (cont'd.)

Detour

- Schema evolution
 - Advantage
 - Faster to execute queries in DW with many schema modifications
 - Disadvantages
 - It limits the end user flexibility to query based on the past schemas
 - Only actual schema based queries are supported

Schema Modification (cont'd.)

- Schema versioning
 - Also no data loss
 - All the data corresponding to all the schemas are always available
 - After a schema modification the data is held in their belonging schema
 - Old data - old schema



Schema Modification (cont'd.)

- Schema versioning
 - Advantages
 - Allows higher flexibility, e.g., “As Is vs. As Was”, etc. queries
 - Disadvantages
 - Adaptation of the data to the queried schema is done on the spot
 - This results in longer query run time



Summary

Summary

- Logical Model
 - Cubes, Dimensions, Hierarchies, Classification Levels

Next Lecture

- Physical Model
 - Relational Implementation through:
 - Star schema: improves query performance for often-used data
 - Snowflake schema: reduce the size of the dimension tables

