Institut für Informationssysteme
Technische Universität Braunschweig

Data Warehousing & Mining Techniques

Wolf-Tilo Balke
Muhammad Usman
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de

Summary

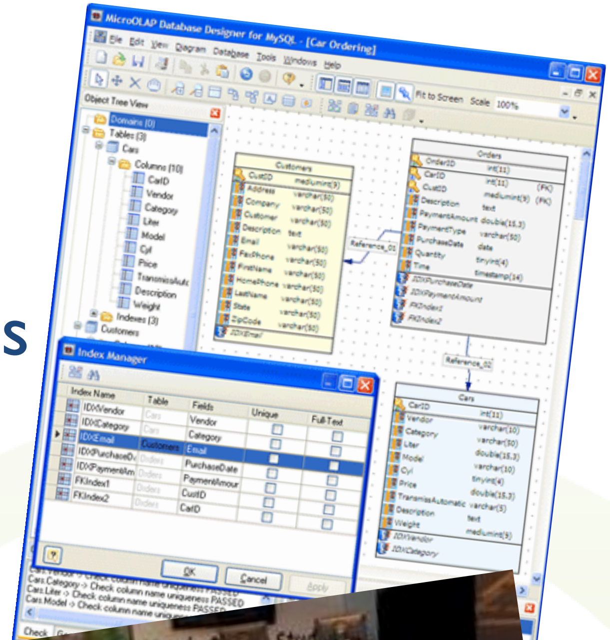
- Last Lecture:
  - Architectures: Three-Tier Architecture
  - Data Modeling in DW – multidimensional paradigm
    - Conceptual Modeling: ME/R and mUML
- This week:
  - Data Modeling (continued)

3. DW Modeling

3. Data Modeling

3.1 Logical Modeling:
Cubes, Dimensions, Hierarchies

3.2 Physical Modeling:
Array storage, Star, Snowflake

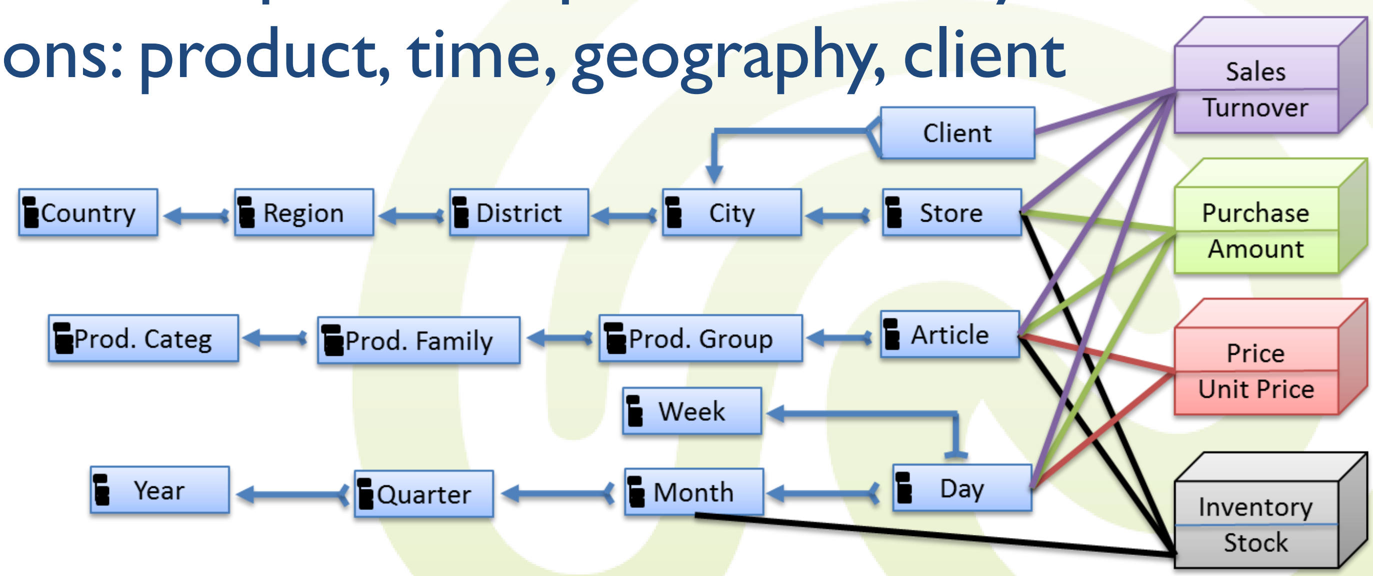


3.1 Logical Model

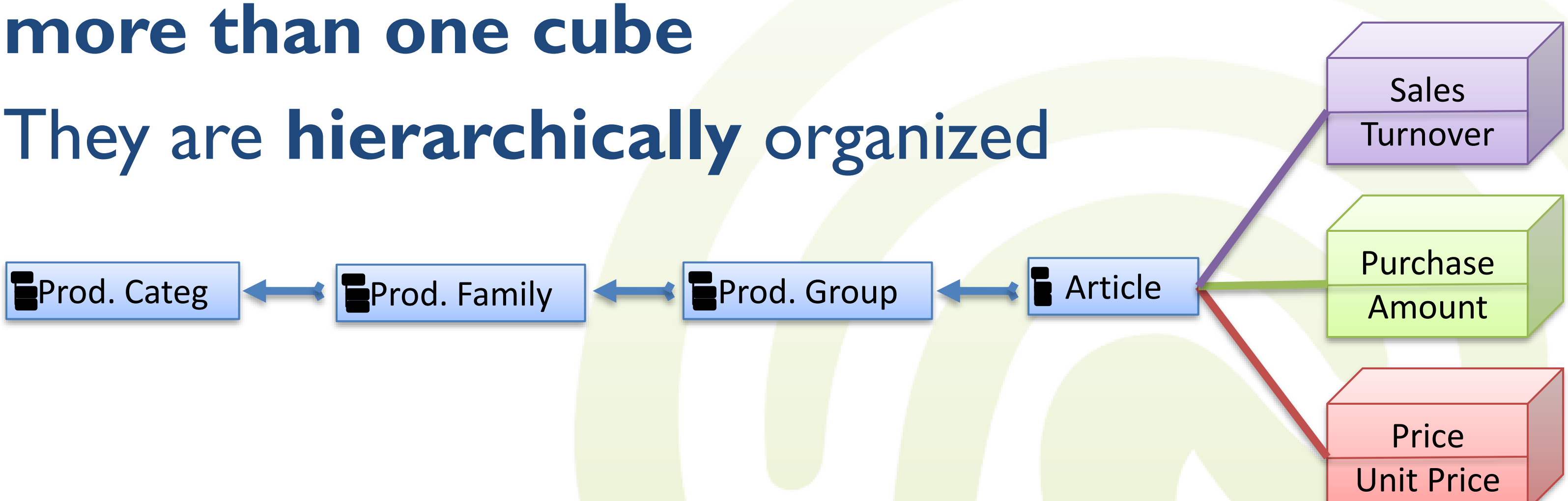
- Elements of the logical model
  - Dimensions and cubes
- Basic operations in the multidimensional paradigm
  - Cube -selection, -projection, -join
- Change support for the logical model

3.1 Logical Model

- **Goal of the Logical Model**
  - Refine the ‘real’ **facts** and **dimensions** of the subjects identified in the conceptual model
  - Establish the **granularity** for dimensions
  - E.g. cubes: sales, purchase, price, inventory dimensions: product, time, geography, client

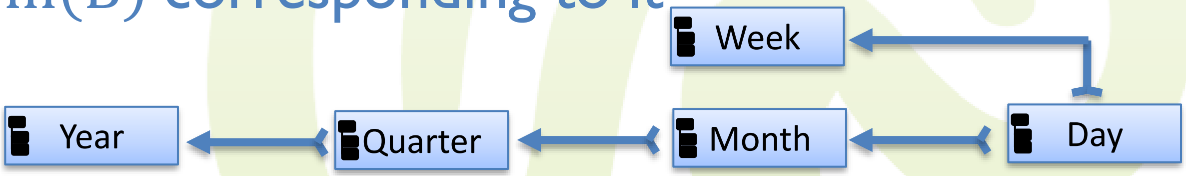


- **Dimensions are entities** chosen in the data model regarding some analysis purpose
  - Each dimension can be used to define **more than one cube**
  - They are **hierarchically** organized



3.1 Dimensions

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

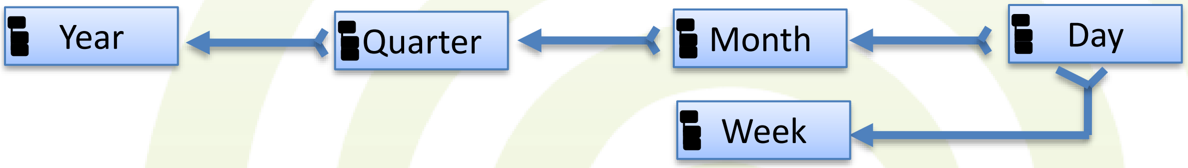


3.1 Dimensions

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

3.1 Dimensions

- 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



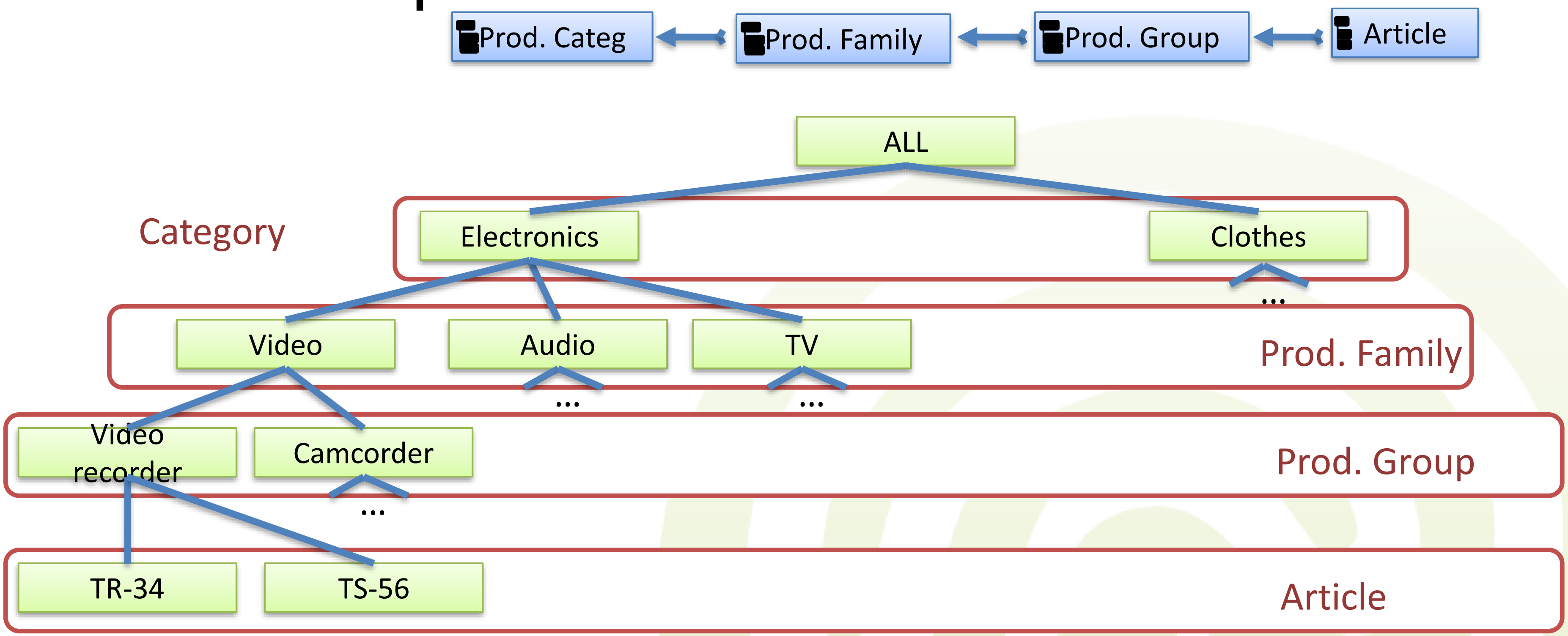
  - Here T.Day is the **smallest element**

3.1 Dimensions

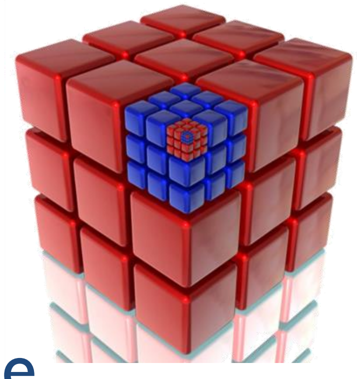
- **Classification hierarchies**
  - Let $D.K_0 \longrightarrow \ldots \longrightarrow D.K_k$ be a path in the classification schema of dimension D
  - A **classification hierarchy** concerning these path is a balanced tree which
    - Has as nodes $dom(D.K_0) \cup \ldots \cup dom(D.K_k) \cup \{ALL\}$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

3.1 Dimensions

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



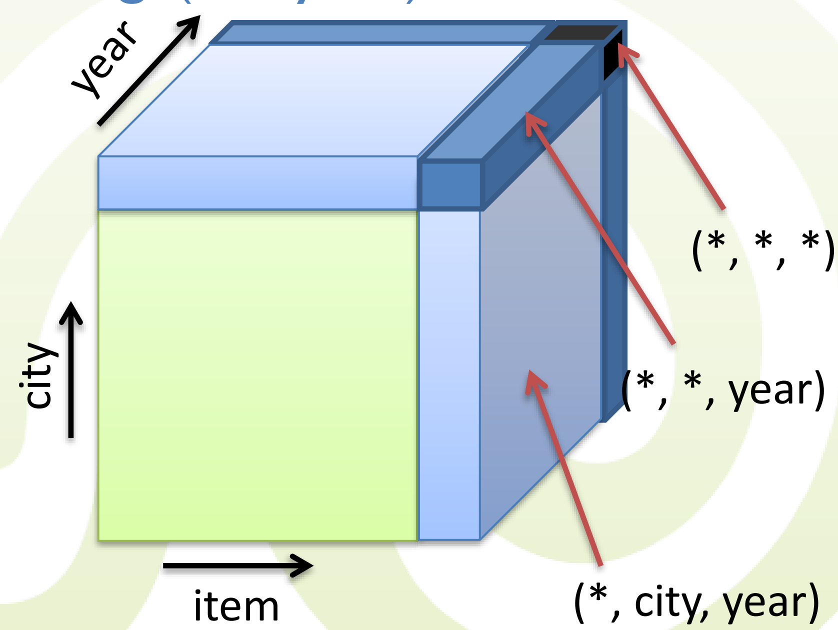
3.1 Cubes



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

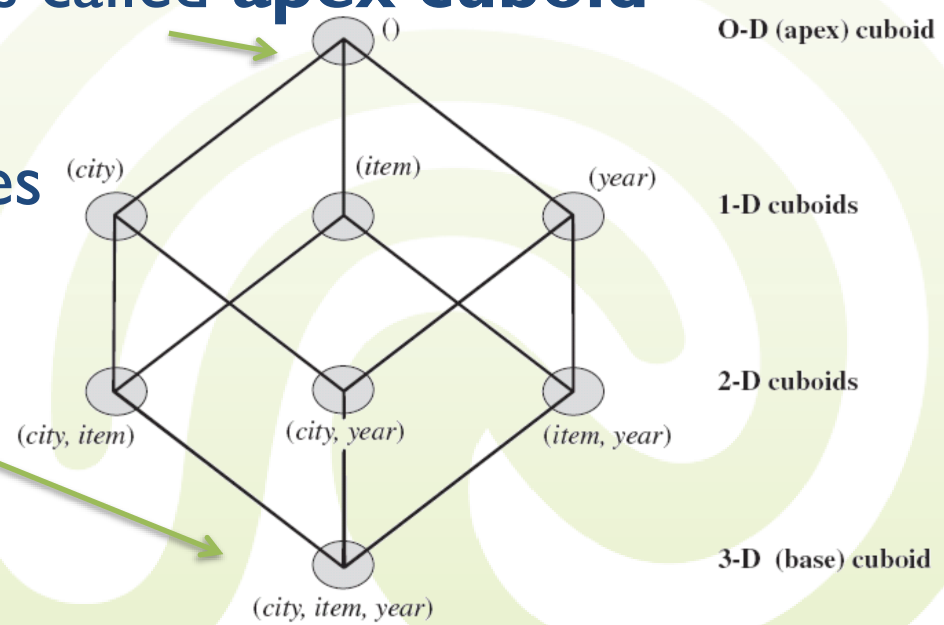
3.1 Cubes

- Aggregates are used for speeding up queries
  - For the 3-dim cube sales ((item, city, year), (turnover)) we have
    - 3 aggregates with 2 dimensions e.g. (*, city, year)
    - 3 aggregates with 1 dimension e.g. (*, *, year)
    - 1 aggregate with no dimension (*,*,*)



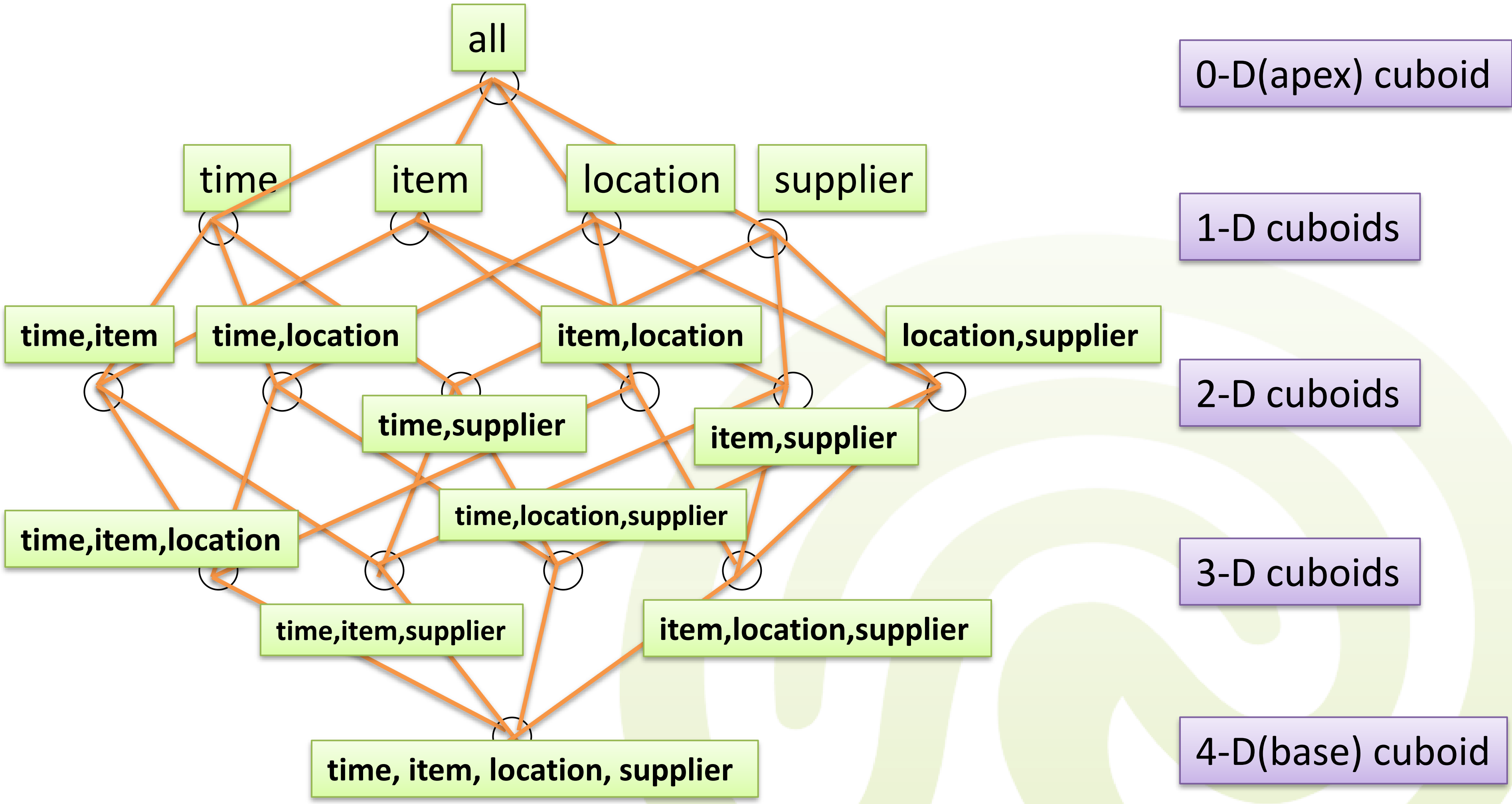
3.1 Cubes

- In the logical model cubes (also comprising the aggregates) are represented as a **lattice of cuboids**
  - The top most cuboid, the 0-D, which holds the highest level of summarization is called **apex cuboid**
  - The nD cube containing non-aggregated measures is called a **base cuboid**



3.1 Cubes

- But things can get complicated pretty fast (4 dim.)



3.1 Basic Operations

- **Basic operations** of the multidimensional paradigm at logical level
  - Selection
  - Projection
  - Cube join
  - Aggregation

3.1 Basic Operations

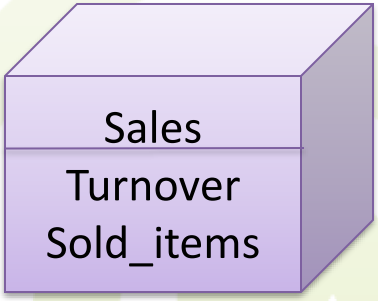
- Multidimensional Selection
  - The **selection** on a cube $C((D_1.K_1, \ldots, D_g.K_g), (M_1, \ldots, M_m))$ with a predicate P, is defined as $\sigma_P(C) = \{z \in C{:}P(z)\}$, if all variables in P are either:
    - Classification levels K , which functionally depend on a classification level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

3.1 Basic Operations

- Multidimensional projection
  - The **projection** of a function of some measure $F(M)$ of cube C is defined as

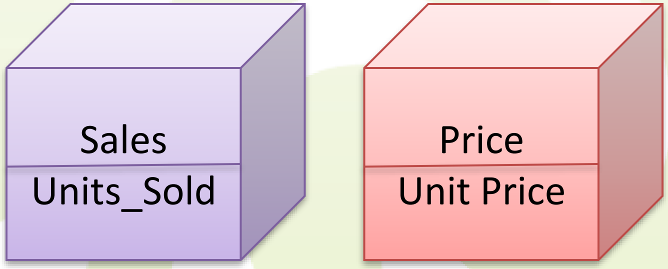
    $$\pi_{F(M)}(C) = \{ (g,F(m)) \in dom(G) \text{ x } dom(F(M)): (g,m) \in C\}$$

  - E.g. $\pi_{turnover,\ sold\_items}(Sales)$



3.1 Basic Operations

- **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** $(G_1 = G_2 = G)$
  - $C_1 \bowtie C_2 = C(G, M_1 \cup M_2)$

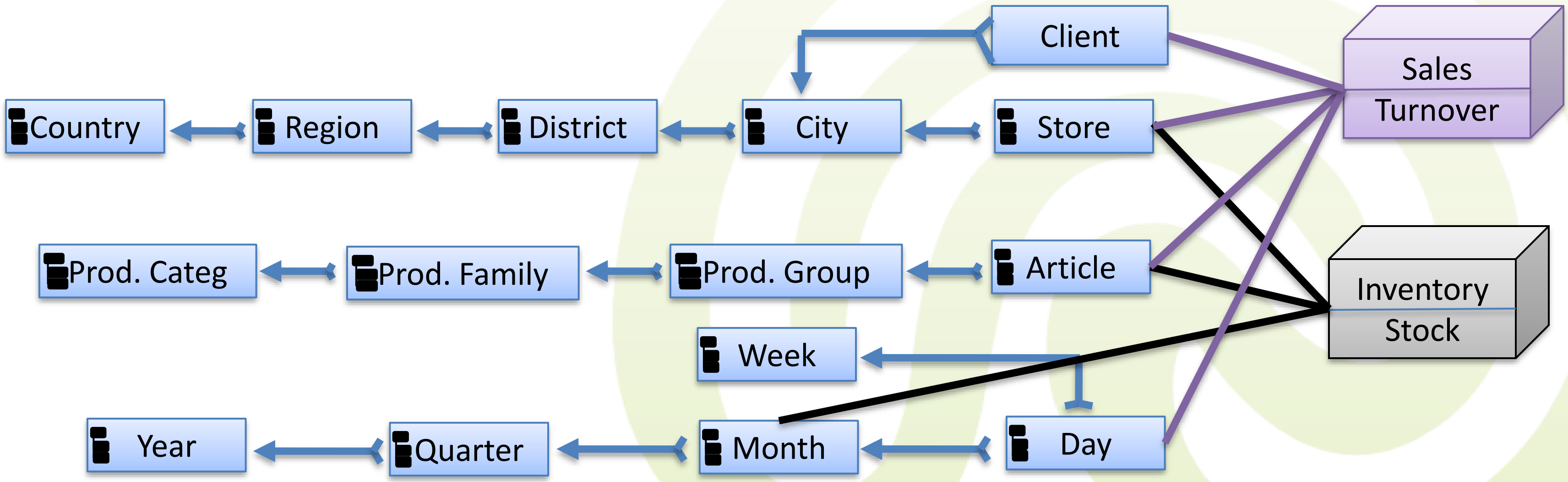


3.1 Basic Operations

- **Comparing granularities**
  - A granularity $G=\{D_1.K_1, \ldots, D_g.K_g\}$ is *finer* than $G'=\{D_1'.K_1', \ldots, D_h'.K_h'\}$, if and only if for each $D_j'.K_j' \in G'\ \exists\ D_i.K_i \in G$ where $D_i.K_i \rightarrow D_j'.K_j'$

3.1 Basic Operations

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



3.1 Basic Operations

- **Aggregation** is the most important operation for OLAP
- Aggregation functions
  - Compute a single value from some set of values, e.g. in SQL: SUM, AVG, Count, …
  - Example: $\mathrm{SUM}_{(\mathrm{P.Product\_group,\ G.City,\ T.Month})}(\mathrm{Sales})$

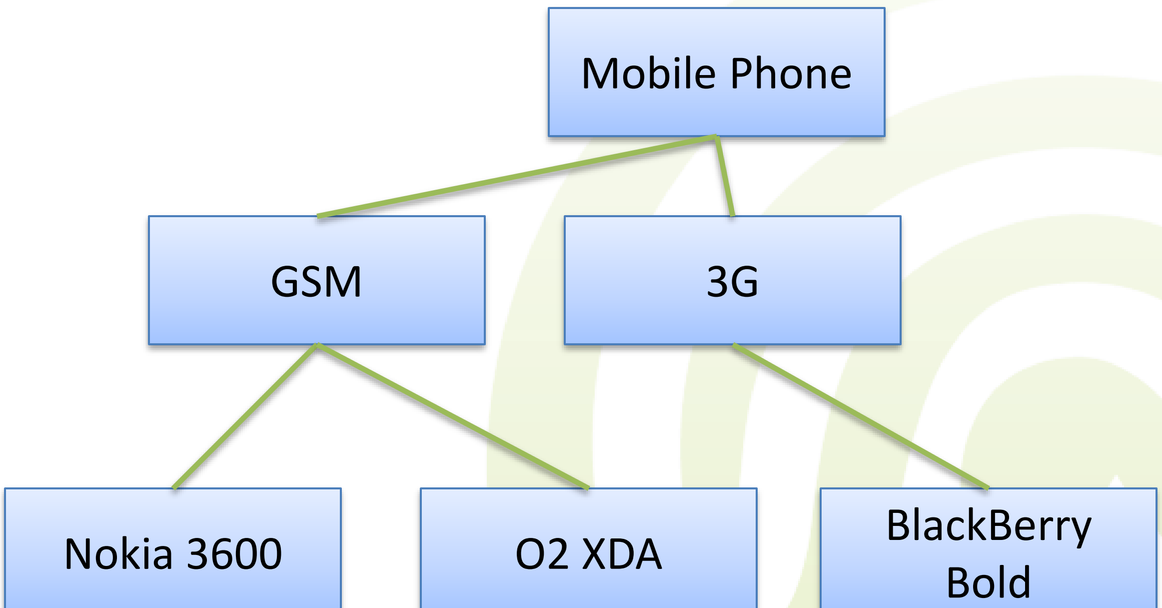
3.1 Change support

Detour

- Classification hierarchy, classification schema, cube schema are all designed in the building phase and **considered as fixed**
  - Practice has proven different
  - DW grow old, too
- Reasons for classification hierarchy and schema **modifications**
  - New requirements
  - **Data evolution**

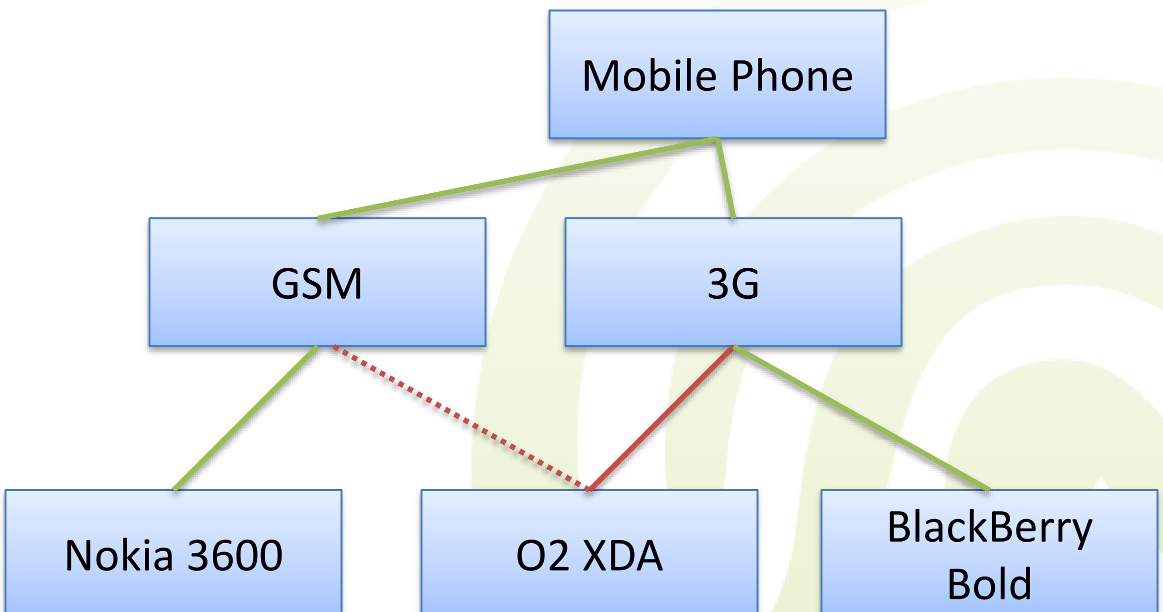
3.1 Classification Hierarchy Detour

- E.g. **Saturn** sells lots of electronics
  - Assume they feed data to their DW since 2003
  - Example of a simple classification hierarchy of data until 01.07.2008, for mobile phones only:



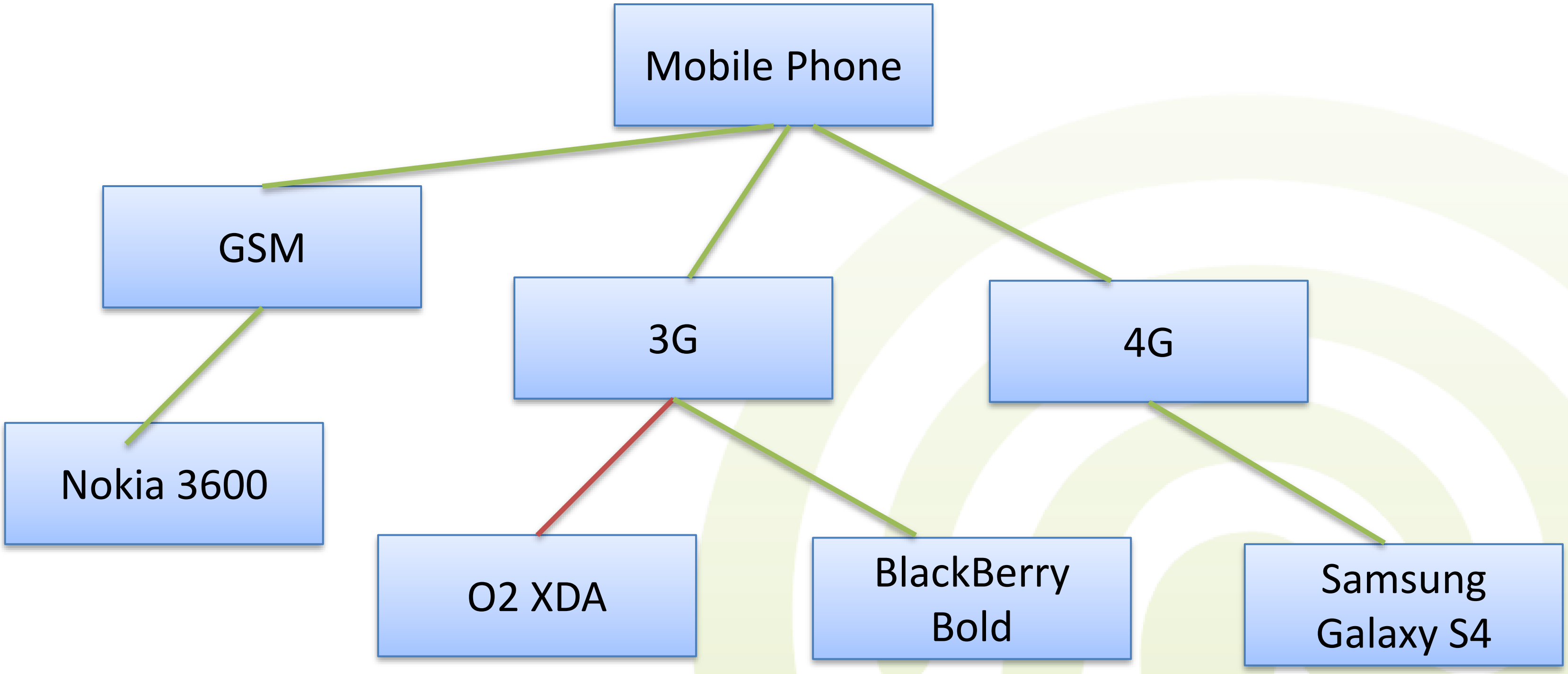
3.1 Classification Hierarchy Detour

- After 01.07.2008 **3G** becomes hip and affordable and many phone makers start migrating towards **3G capable phones**
  – O2 made its XDA 3G capable

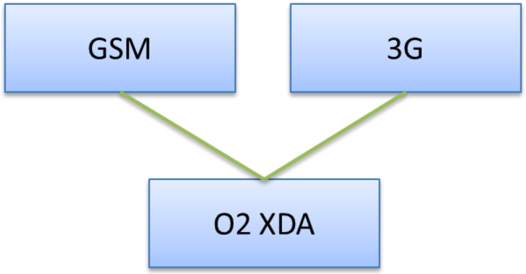


3.1 Classification Hierarchy Detour

- After 01.04.2011 phone makers already develop **4G capable phones**



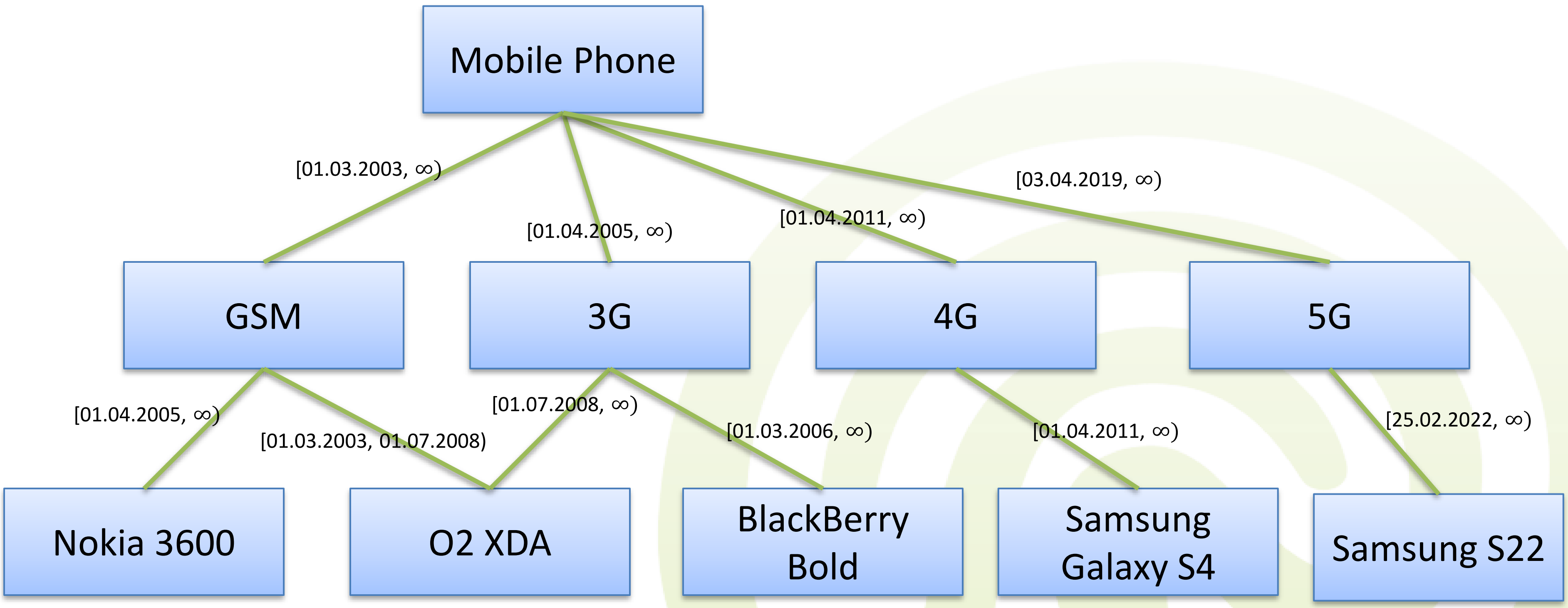
3.1 Classification Hierarchy Detour



- Problem: Sales volume for GSM products can be problematic
  - According to the most actual schema, O2 XDA belongs to the 3G category
  - No O2XDA GSM only device will account for the GSM sales volume
- Solution: **trace the evolution** of the data
  - **Versioning system** of the classification hierarchy with **validity timestamps**

3.1 Classification Hierarchy Detour

- **Annotated change data**



3.1 Classification Hierarchy

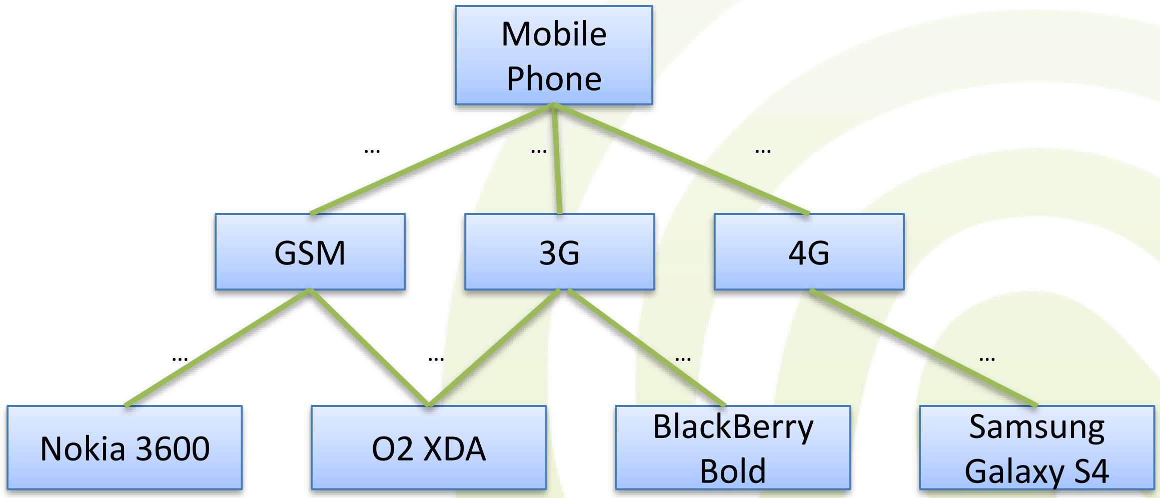
Detour

- The tree can be stored as metadata as a **validity matrix**
  - Rows are parent nodes and columns are child nodes

	GSM	3G	4G	Nokia 3600	O2 XDA	Berry Bold	Samsung Galaxy S4
Mobile phone	[01.03.2003, ∞)	[01.04.2005, ∞)	[01.04.2011, ∞)				
GSM				[01.04.2005, ∞)	[01.03.2003, 01.07.2008)		
3G					[01.07.2008, ∞)	[01.03.2006, ∞)	
4G							[01.04.2011, ∞)
Nokia 3600							
O2 XDA							
Berry Bold							

3.1 Classification Hierarchy Detour

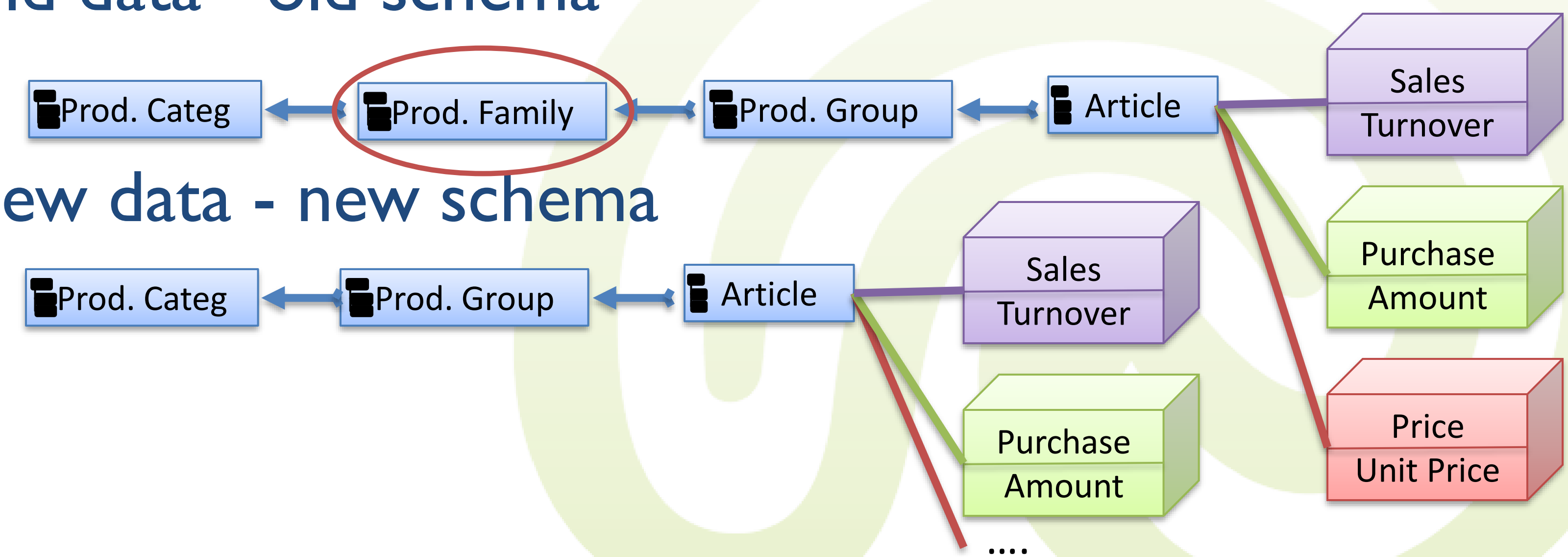
- Flexibility gain: Having the validity information, queries like **as is versus as was** are possible
  - Even if in the latest classification hierarchy GSM products would not be provided anymore one can still compare sales for **O2XDA as GSM vs. 3G**



3.1 Schema Versioning

Detour

- **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
  - New data - new schema



3.1 Schema Versioning

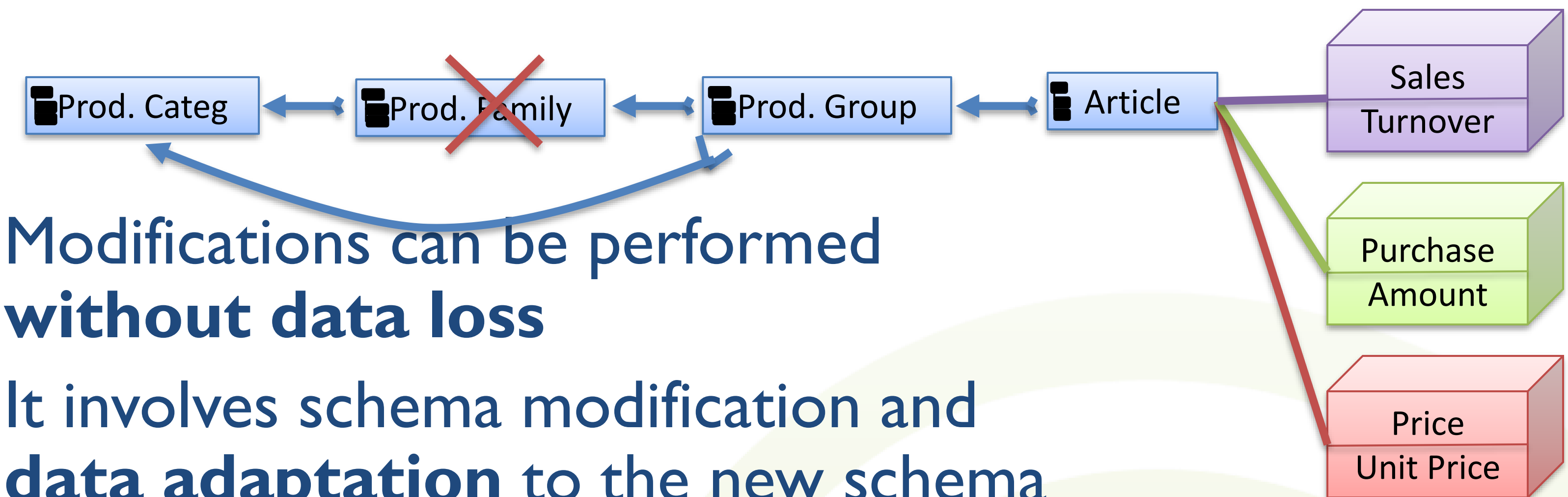
Detour

- Advantages
  - Allows higher flexibility e.g. querying for the product family for old data
- Disadvantages
  - Adaptation of the data to the queried schema is done **on the spot**
  - This results in longer query run time

3.1 Schema Modification

Detour

- Schema **evolution**



  - Modifications can be performed **without data loss**
  - It involves schema modification and **data adaptation** to the new schema
  - Advantage: Faster to execute queries for DW with many schema modifications
    - Because all data is prepared for the current and single schema
  - Disadvantage: It limits user flexibility - only queries based on the actual schema are supported

3.2 Physical Model

- Defining the **physical structures**
  - Define the actual storage architecture
  - Decide on how the data is to be accessed and how it is arranged

  - Performance tuning strategies (next lecture)
    - Indexing
    - Partitioning
    - Materialization

3.2 Physical Model

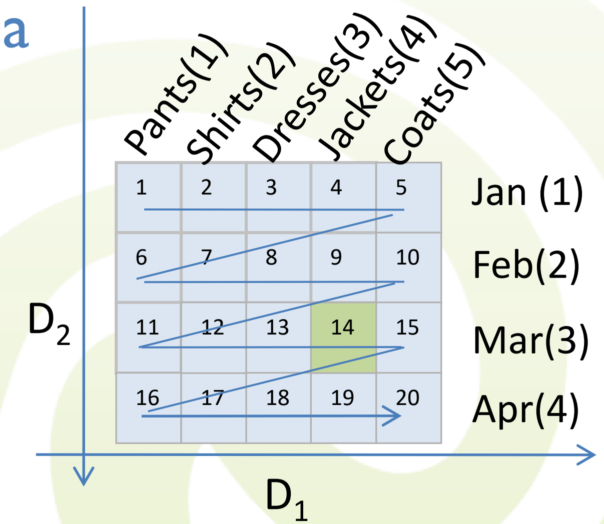
- The data in the DW is stored according to the multidimensional paradigm
  - The obvious multidimensional storage model is directly encoding matrices
- Relational DB vendors, in the market place saw the opportunity and adapted their systems
  - Special schemas respecting the multidimensional paradigm

3.2 Multidimensional Model

- The basic data structure for multidimensional data storage is the **array**
- The elementary data structures are the cubes and the dimensions
  - $C=((D_1, \ldots, D_n), (M_1, \ldots, M_m))$
- The storage of matrices is intuitive as arrays of arrays i.e. physically **linearized**

3.2 Linearization

- Linearization example: $2\mathrm{D}$ cube $|\mathrm{D}_1| = 5$, $|\mathrm{D}_2| = 4$, cube cells = 20
  - Query: Jackets sold in March?
    - Measure stored in cube cell $\mathrm{D}_1[4], \mathrm{D}_2[3]$
    - The $2\mathrm{D}$ cube is physically stored as a linear array, so $\mathrm{D}_1[4], \mathrm{D}_2[3]$ becomes array cell 14
      - $(\mathrm{Index}(\mathrm{D}_2) - 1) * |\mathrm{D}_1| + \mathrm{Index}(\mathrm{D}_1)$
      - Linearized Index $= 2 * 5 + 4 = 14$



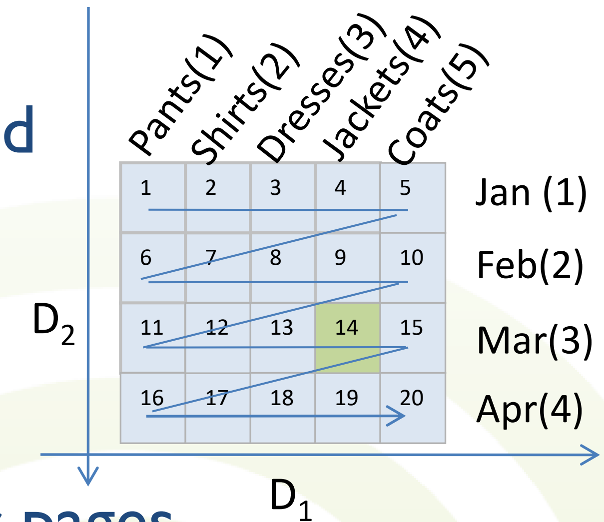
3.2 Linearization

- Generalization:
  - Given a cube $C=((D_1, D_2, \ldots, D_n), (M_1{:}Type_1, M_2{:}Type_2, \ldots, M_m{:}Type_m))$, the index of a cube cell z with coordinates $(x_1, x_2, \ldots, x_n)$ can be linearized as follows:
    - $Index(z) = x_1 + (x_2 - 1) * |D_1| + (x_3 - 1) * |D_1| * |D_2| + \ldots + (x_n - 1) * |D_1| * \ldots * |D_{n-1}| =$

      $= 1 + \sum_{i=1}^{n} ((x_i - 1) * \prod_{j=1}^{i-1} |D_j|)$

3.2 Problems in Array-Storage

- Influence of the **order of the dimensions** in the cube definition
  - In the cube the cells of D_2 are ordered one beneath the other e.g., sales of all pants involves a **column** in the cube
  - After linearization, the information is **spread** among several data blocks or pages
  - If we consider a data block to hold 5 cells, a query over all **products sold in January** can be answered with just 1 block read, but a query of all sold pants, involves reading 4 blocks



3.2 Problems in Array-Storage

- Solution: use **caching techniques**
  - But…caching and swapping is performed by the operating system, too
  - MDBMS has to manage its caches such that the OS doesn't perform any damaging swaps

3.2 Problems in Array-Storage

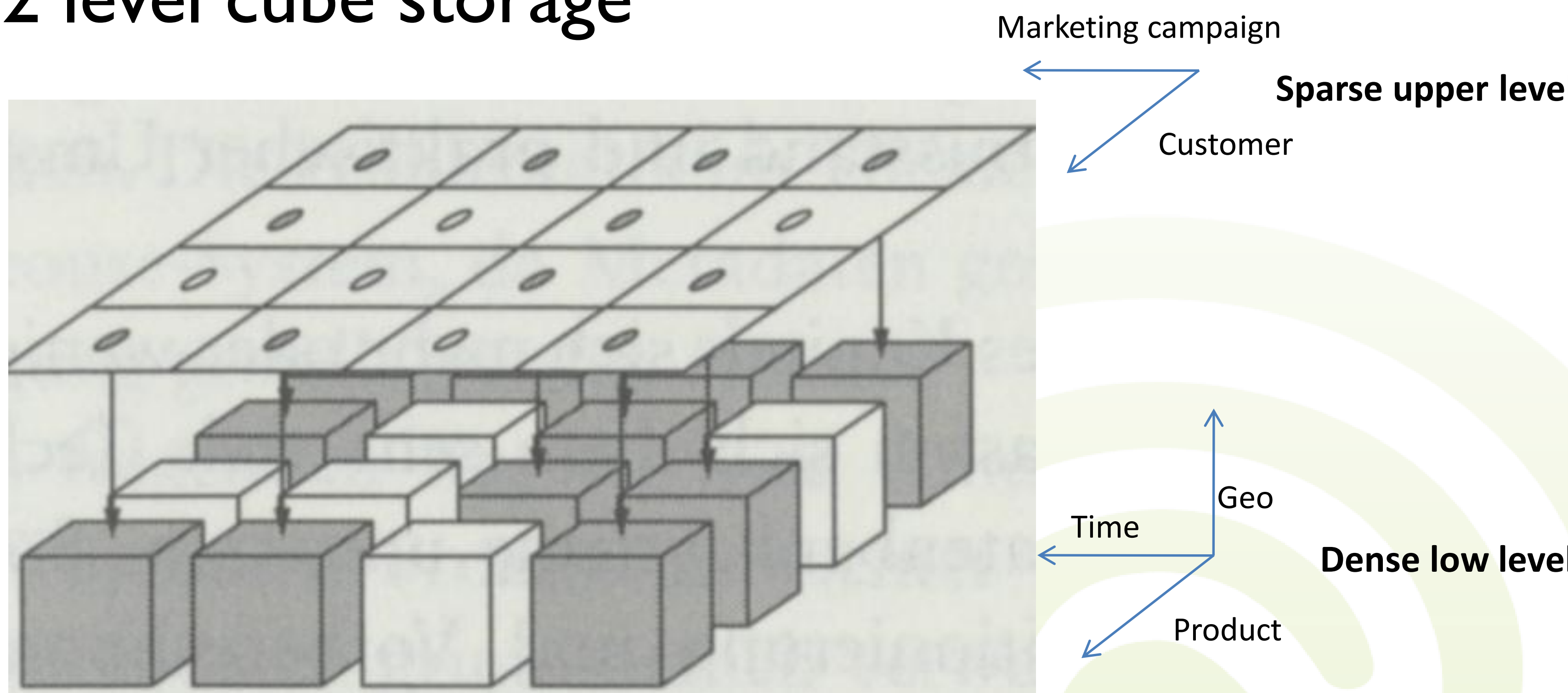
- Storage of **dense cubes**
  - If cubes are dense, array storage is quite efficient. However, operations suffer due to the large cubes
    - Loading huge matrixes in memory is not good
  - **Solution**: store dense cubes not linearly but **on 2 levels**
    - The first contains **indexes** and the second the **data cells** stored in blocks
    - Optimization procedures like **indexes** (trees, bitmaps), **physical partitioning**, and **compression** (run-length-encoding) can be used

3.2 Problems in Array-Storage

- Storage of **sparse cubes**
  - **All** the cells of a cube, **including empty ones**, have to be stored
  - Sparseness leads to data being stored in several physical blocks or pages
    - The query speed is affected by the large number of block accesses on the secondary memory
  - **Solution**:
    - **Do not store empty blocks or pages** but adapt the index structure
    - **2 level data structure:** upper layer holds all possible combinations of the sparse dimensions, lower layer holds dense dimensions

3.2 Problems in Array-Storage

- 2 level cube storage



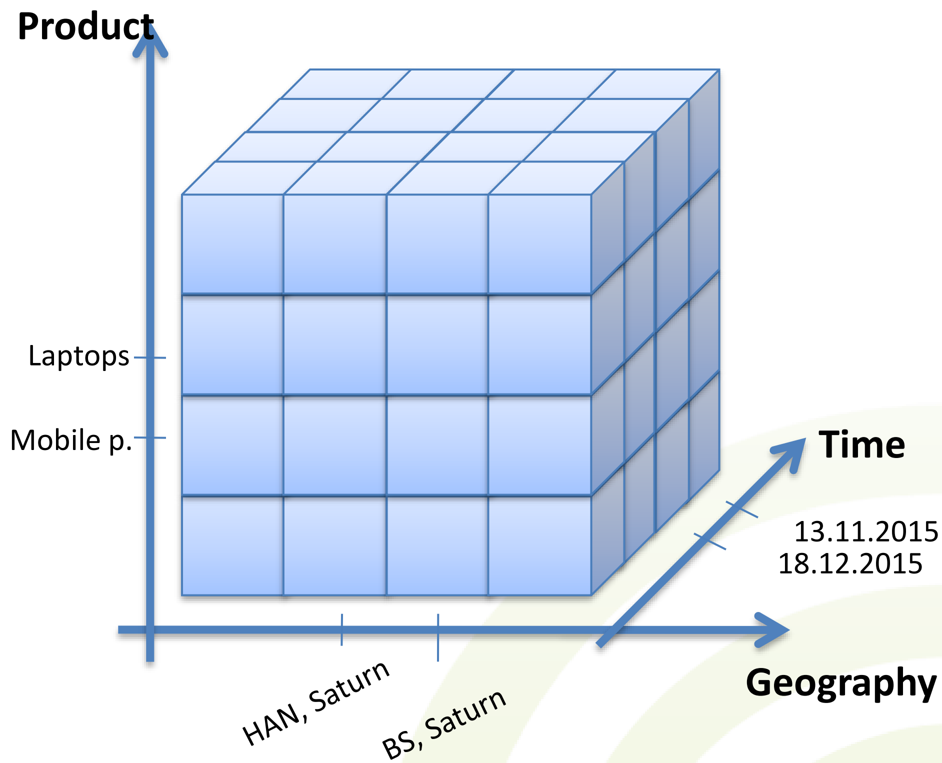
3.2 Physical Model

- **Relational** model, goals:
  - As low loss of semantically knowledge as possible e.g., classification hierarchies
  - The translation from multidimensional queries must be efficient
  - The RDBMS should be able to run the translated queries efficiently
  - The maintenance of the present tables should be easy and fast e.g., when loading new data

3.2 Relational Model

- Going from **multidimensional** to **relational**
  - **Representations** for cubes, dimensions, classification hierarchies and attributes
  - **Implementation** of cubes without the classification hierarchies is easy
    - A table can be seen as a *cube*
    - A column of a table can be considered as a dimension mapping
    - A tuple in the table represents a cell in the cube
    - If we interpret only a part of the columns as dimensions we can use the rest as measures
    - The resulting table is called a **fact table**

3.2 Relational Model



Article	Store	Day	Sales
Laptops	Hannover, Saturn	13.11.2015	6
Mobile Phones	Hannover Saturn	18.12.2015	24
Laptops	Braunschweig Saturn	18.12.2015	3

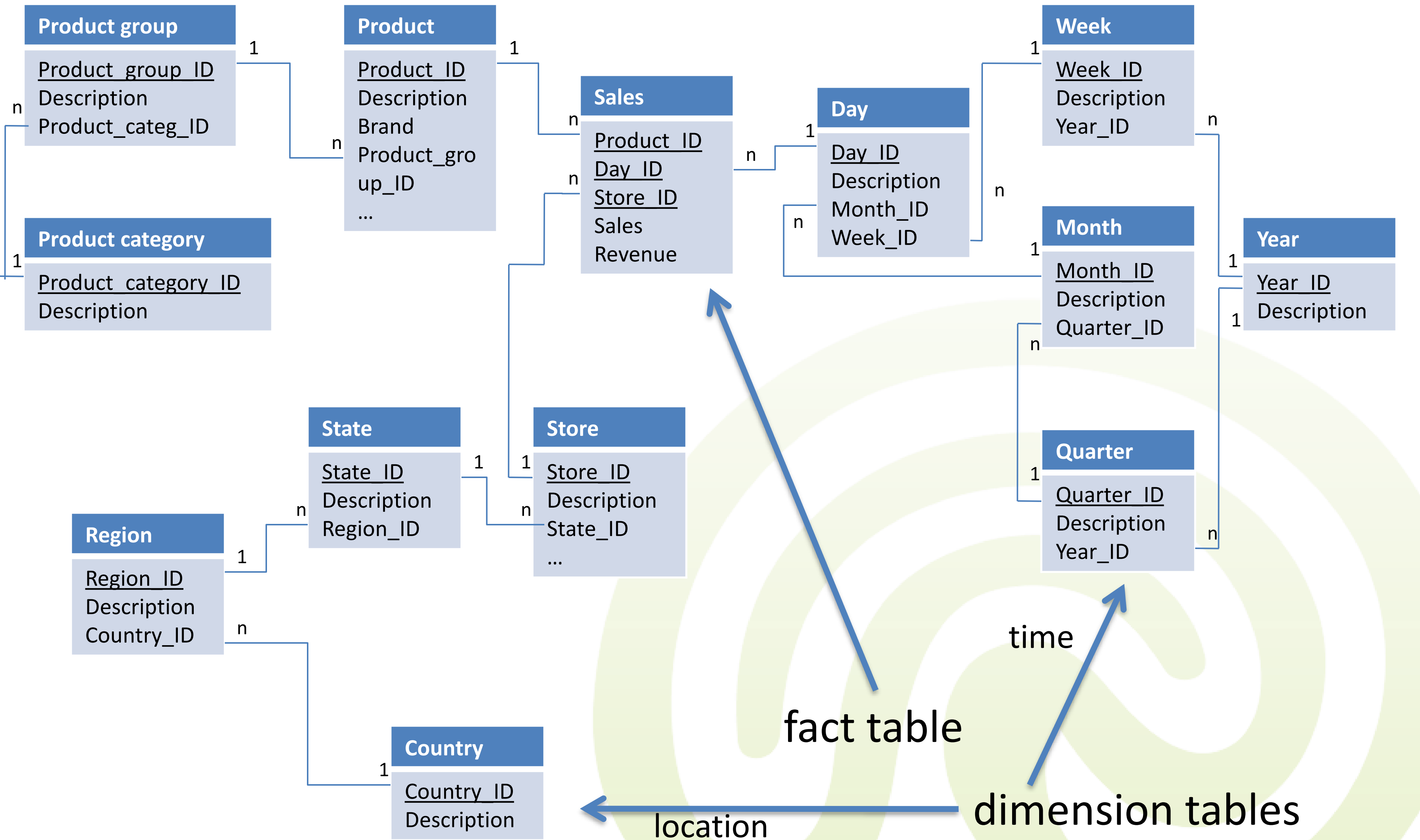
3.2 Relational Model

- **Snowflake-schema**
  - Simple idea: use a table for each classification level
    - This table includes the ID of the classification level and other attributes
    - 2 neighbor classification levels are connected by 1:n connections e.g., from n Days to 1 Month
    - The measures of a cube are maintained in a fact table
    - Besides measures, there are also the foreign key IDs for the smallest classification levels

3.2 Snowflake Schema

- **Snowflake?**
  - The **facts/measures** are in the center
  - The **dimensions** spread out in each direction and branch out with their granularity

3.2 Snowflake Example

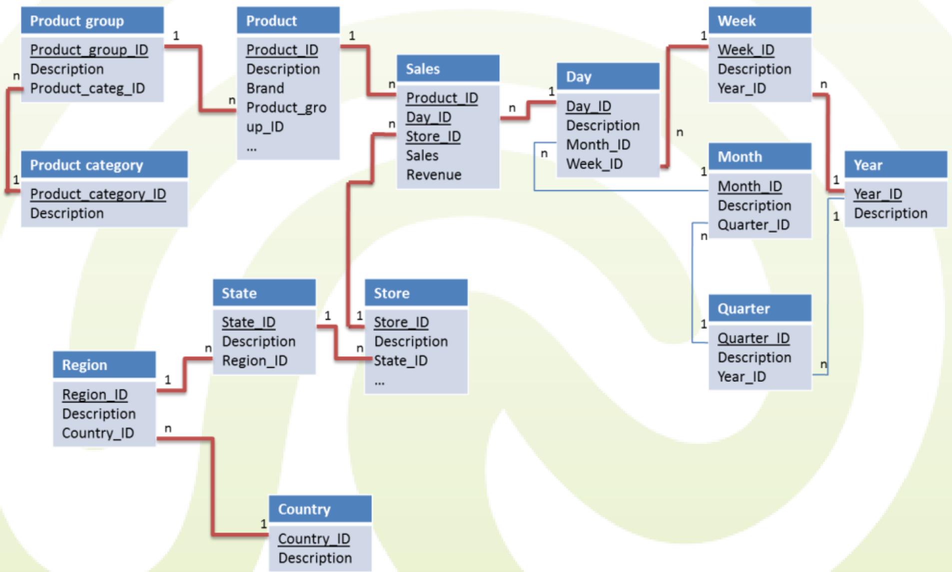


3.2 Snowflake Schema

- Advantage:
  – With a snowflake schema the **size of the dimension tables** will be reduced and queries will run faster
  – Easier to maintain (avoid redundancy)
  – Allows for more flexible querying with complex dimensions with many classification levels

3.2 Snowflake Schema

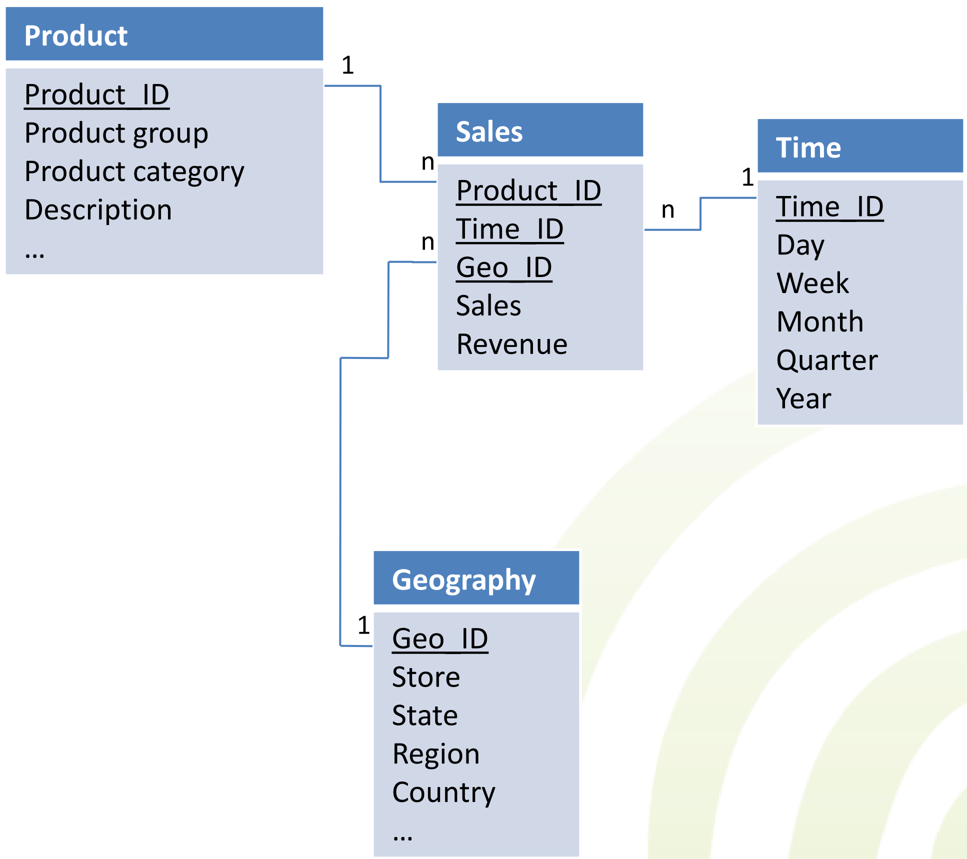
- Disadvantages
  - If **fact tables** are responsible for **90%** of the storage requirements then normalizing the dimensions can reduce the performance of the DW because it leads to a **large number of tables**
    - E.g. join between product categ. country and year have to be performed at query time



3.2 Relational Model

- **Star schema**
  - Basic idea: use a **denormalized** schema for all the dimensions
    - A **star schema** can be obtained from the snowflake schema through the denormalization of the tables belonging to a dimension

3.2 Star Schema - Example



3.2 Star Schema

- **Advantages**
  - Improves query performance for often-used data
  - Less tables and simple structure
  - Efficient query processing with regard to dimension joining
- **Disadvantages**
  - In some cases, high overhead of redundant data
  - Representing many-to-many relationships?

3.2 Snowflake vs. Star

- **Snowflake** vs. **Star**

  - The structure of the classifications are expressed in table schemas
  - The fact and dimension tables are normalized

  - The entire classification is expressed in just one table
  - The fact table is normalized while in the dimension tables the normalization is broken
    - This leads to redundancy of information in the dimension tables

3.2 Examples

- Snowflake

Product_ID	Description	Brand	Prod_group_ID
10	E71	Nokia	4
11	PS-42A	Samsung	2
12	5800	Nokia	4
	Bold	Berry	4

Prod_group_ID	Description	Prod_categ_ID
2	TV	11
4	Mobile Pho..	11

Prod_categ_ID	Description
11	Electronics

- Star

Product_ ID	Description	…	Prod. group	Prod. categ
10	E71	…	Mobile Ph..	Electronics
11	PS-42A	…	TV	Electronics
12	5800		Mobile Ph..	Electronics
13	Bold		Mobile Ph..	Electronics

3.2 Snowflake to Star

- When should we go from snowflake to star? **Heuristics**-based decision
  - When typical queries relate to coarser granularity (like product category)
  - When the volume of data in the dimension tables is relatively low compared to the fact table
  - When modifications on the classifications are rare compared to insertion of fact data

3.2 Do we have a winner?

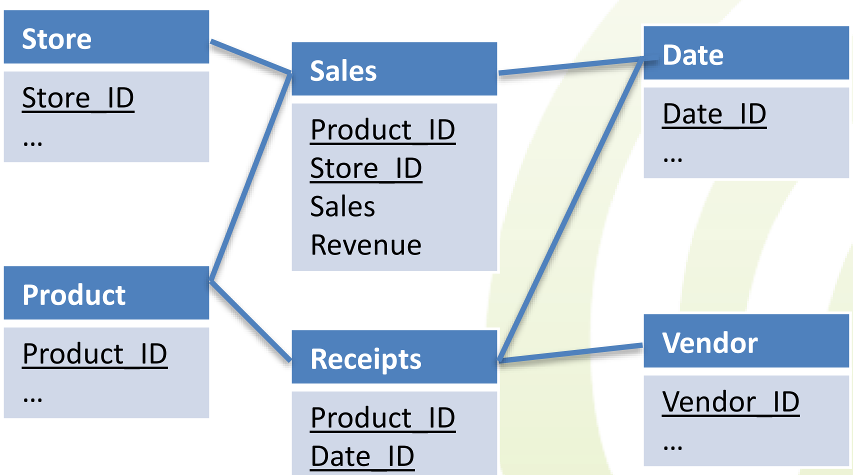
- Snowflake or Star?
  - It depends on the necessity
    - Fast query processing or efficient space usage
  - However, most of the time a **mixed form** is used
    - The **Starflake schema:** some dimensions stay normalized corresponding to the snowflake schema, while others are denormalized according to the star schema

3.2 Our forces combined

- The **Starflake schema:** which dimensions to normalize?
  - **Frequency** of the modifications: if the dimensions change often, normalization leads to better results
  - **Amount** of dimension elements: the bigger the dimension tables, the more space normalization saves
  - **Number of classification levels** in a dimension: more classification levels introduce more redundancy in the star schema
  - **Materialization of aggregates** for the dimension levels: if the aggregates are materialized, a normalization of the dimension can bring better response time

3.2 More Schemas

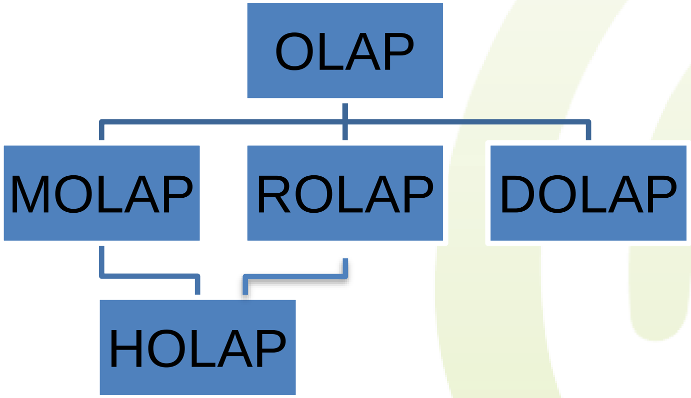
- **Galaxies**
  - In practice it is possible to have more **measures** described by different **dimensions**
    - Thus, more **fact tables**



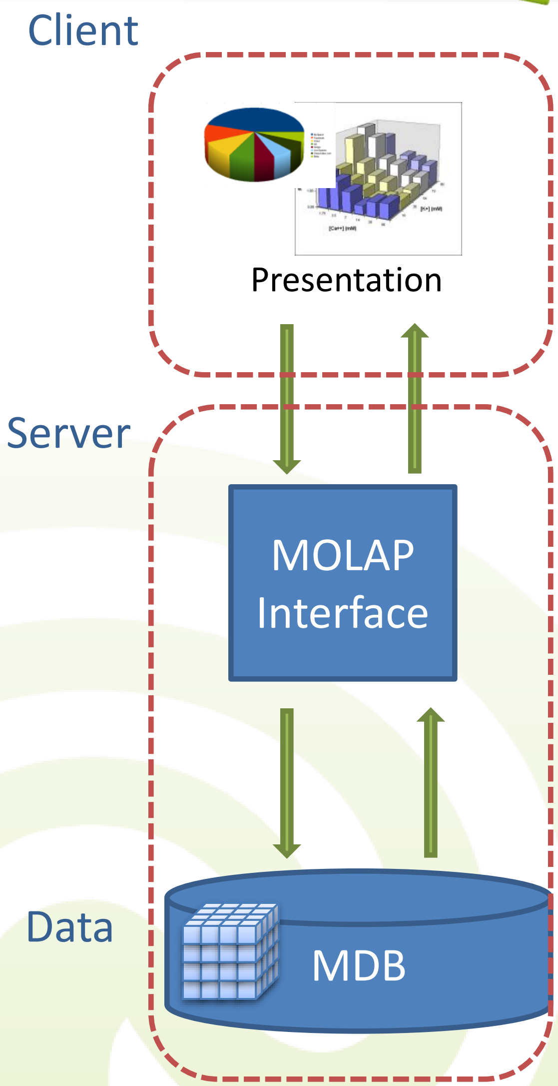
3.2 Physical Models

Detour

- Based on the physical model used:
  - MOLAP (Multidimensional OLAP)
  - ROLAP (Relational OLAP)
  - HOLAP (Hybrid OLAP)
  - DOLAP (Desktop OLAP)



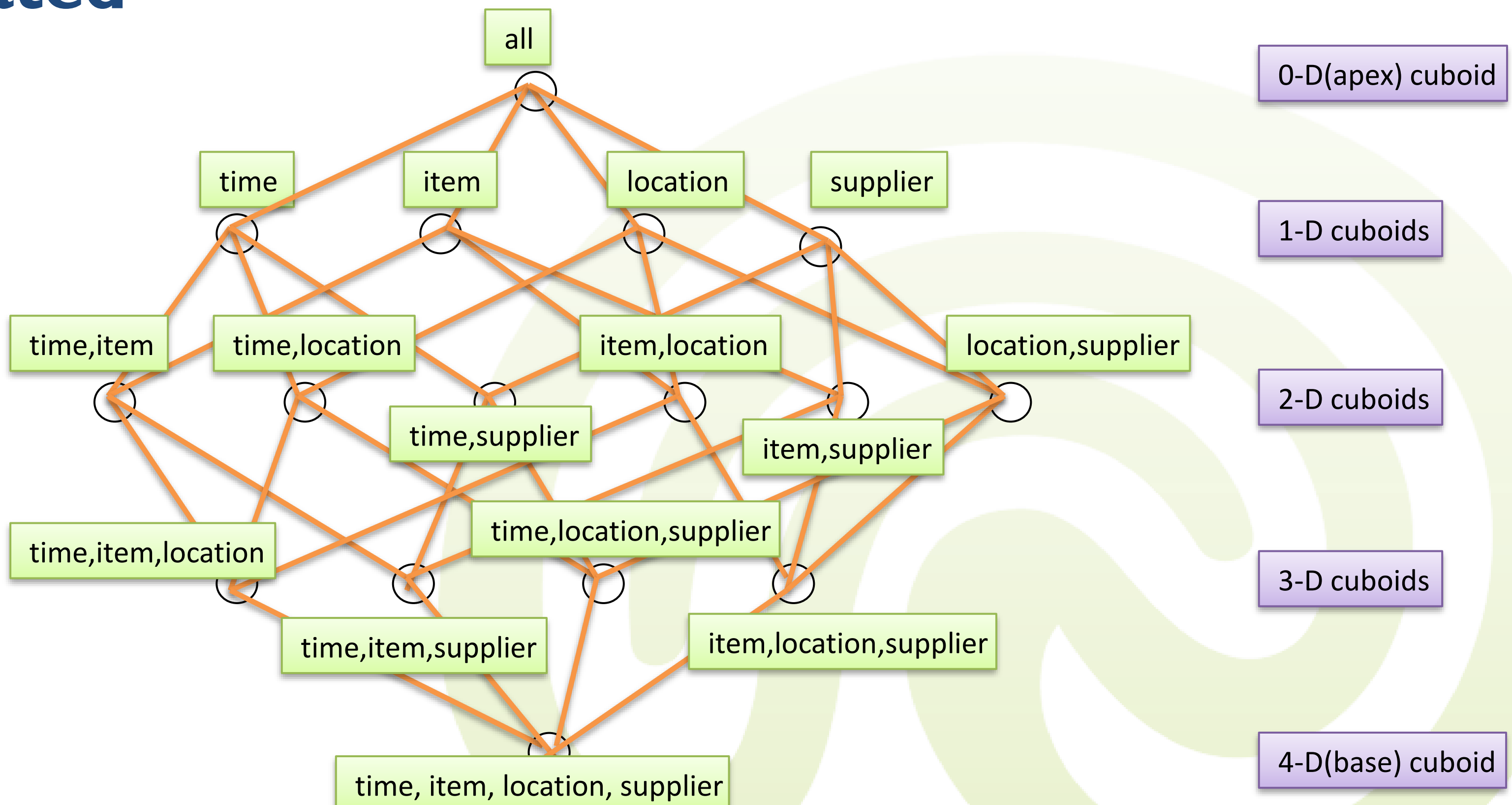
- **MOLAP**
  - Presentation layer provides the multidimensional view
  - The MOLAP server stores data in a multidimensional structure
    - The computation (pre-aggregation) occurs in this layer during the **loading step** (not at query)



3.2 MOLAP

Detour

- Advantage: excellent performance
  - **All values are pre-generated when the cube is created**



3.2 MOLAP

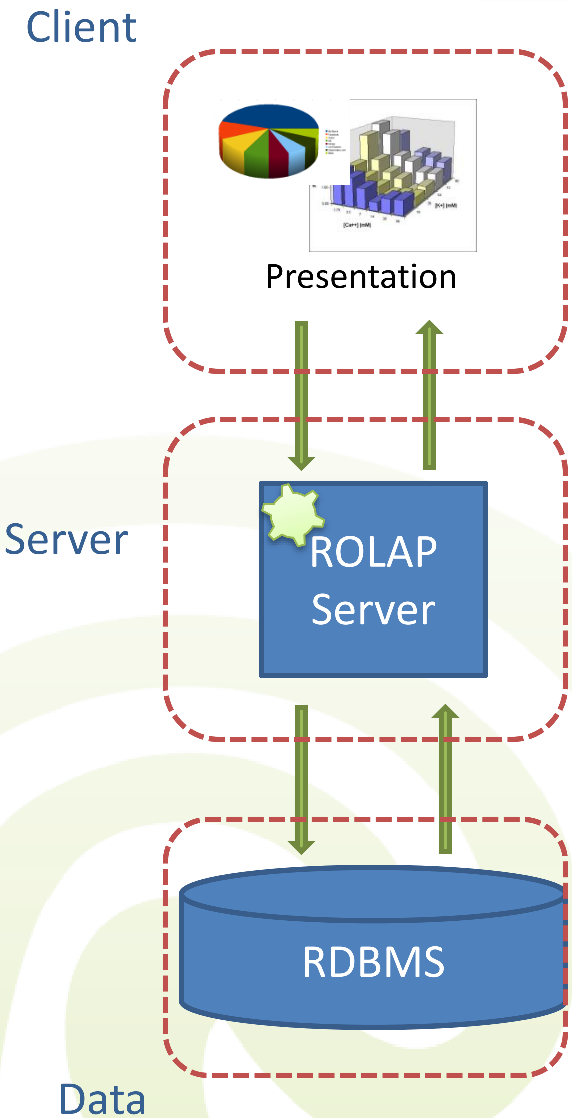
Detour

- Disadvantages
  - Enormous amount of overhead
    - An input file of 200 MB can expand to 5 GB with aggregates
  - Limited amount of data it can handle
    - Cubes can be derived from a large amount of data, but usually only **summary level information** are be included in the cube
  - Requires additional investment
    - Cube technology is **often proprietary**
- Products:
  - Cognos (IBM), Essbase (Oracle Analytics Cloud), Microsoft Analysis Service, Palo (Open source)

3.2 Physical Models

Detour

- ROLAP
  - Presentation layer provides the multidimensional view
  - The ROLAP Server generates SQL queries, from the OLAP requests, to query the RDBMS
  - Data is stored in **RDBs**



3.2 ROLAP

Detour

- Special schema design: e.g., **star, snowflake**
- Special indexes: e.g., bitmap, R-Trees
- Advantages
  - Proven technology (relational model, DBMS)
  - Can handle large amounts of data (VLDBs)
- Disadvantages
  - Limited SQL functionalities
- Products
  - Microsoft Analysis Service, Siebel Analytics (now Oracle BI), Micro Strategy, Mondrian (open source)

3.2 ROLAP vs. MOLAP

Detour

- Based on OLAP needs…

OLAP needs		MOLAP	ROLAP
User Benefits	Multidimensional View	√	√
	Excellent Performance	√	-
	Real-Time Data Access	-	√
	High Data Capacity	-	√
MIS Benefits	Easy Development	√	-
	Low Structure Maintenance	-	√
	Low Aggregate Maintenance	√	-

… MOLAP and ROLAP complement each other

- **Why not combine them?**

3.2 Physical Models

Detour

- **HOLAP:** Best of both worlds
- Split the data between MOLAP and ROLAP
  - Vertical partitioning
    - Aggregations are stored in MOLAP for **fast query performance**,
    - Detailed data in ROLAP to **optimize time of cube processing** (loading the data from the OLTP)
  - Horizontal partitioning
    - HOLAP stores some slice of data, usually the more recent one (i.e. sliced by Time dimension) in MOLAP for fast query performance
    - Older data in ROLAP

3.2 Physical Models

Detour

- **DOLAP:** Developed as extension to the production system reports
  - Downloads a small hypercube from a central point (data mart or DW)
  - Performs multidimensional analysis while disconnected from the data source
  - **Computation is performed at the client side**
  - Requires little investment
  - It lacks the ability to manage large data sets

Summary

Summary

- Logical Model
  - Dimensions, Hierarchies, Classification Levels and Cubes
- Physical Level
  - Array based storage
    - How to perform linearization
    - Problems:
      - Order of dimensions – solution: caching
      - Dense Cubes, Sparse Cubes - solution: 2 level storage
  - MOLAP, ROLAP, HOLAP

Summary

Summary

- Physical Level
  - Relational Implementation through:
    - Star schema: improves query performance for often-used data
      - Less tables and simple structure
      - Efficient query processing with regard to dimensions
      - In some cases, high overhead of redundant data
    - Snowflake schema: reduce the size of the dimension tables
      - However, through dimension normalization - large number of tables

Next lecture

- DW Optimization / Indexes
  - Bitmap indexes
  - Tree based indexes