POZNAN UNIVERSITY OF TECHNOLOGY

Modeling Data Warehouse

Part 2

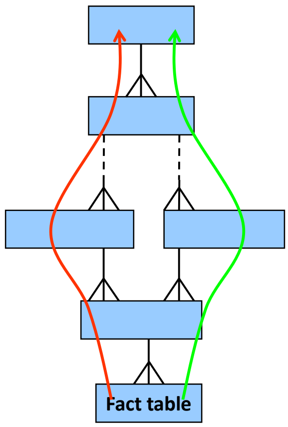
Robert Wrembel
Poznan University of Technology
Institute of Computing Science
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel

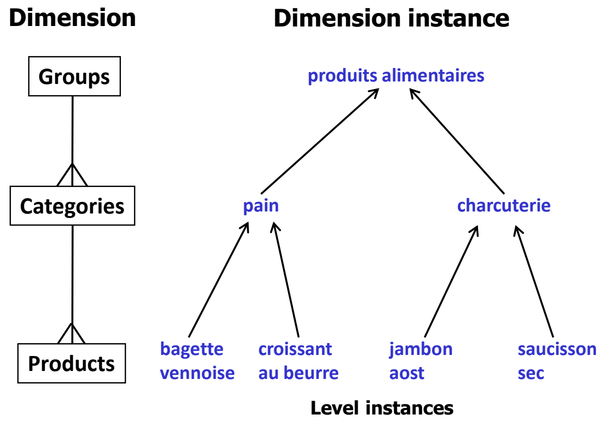
Outline

- **Roll-up and aggregability**
- **Taxonomy of dimension hierarchies**
- **Multidimenisional OLAP**
- **Updates to dimension instances → SCD**

Definitions

Aggregation path





Aggregability

- **Aggregability (summarizability)**
  - **to be able to aggregate measures on a higher level (City) in a dim. hierarchy based on aggregates from a lower level (Shop)**
- **Criteria of correct aggregability**
  - **disjointnes of level instances**
    - **1:M relationship between an upper and a lover level in a dim. hierarchy**
    - **a lower level instance is related to only one upper level instance**
  - **completeness**
    - **all level instances belong to a dimension instance**
    - **every lower level instance is related to an upper level instance**
  - **right aggregate function to compute**
    - **using an adequate aggregate function to a given type of a measure**

Types of measures

- **Addtitive** (flow, rate) ⇨ correct aggregation for **all** dimensions
  - e.g., nb of items sold ⇨ possible aggregation in dim. Time, Customer, Product, Shop, Supplier, ...
- **Semiadditive** (stock, level) ⇨ correct aggregation for **some** dimensions
  - e.g., nb of items in a store ⇨ possible aggregation in dim. (Store → City → Region)
  - aggregating in Time results in uninterpretable results
- **Nonadditive** (value-per-unit) ⇨ aggregated values are non-interpretable in **any** dimension
  - e.g., net price, exchange rate, opening tick for SUM

Types of agg. functions

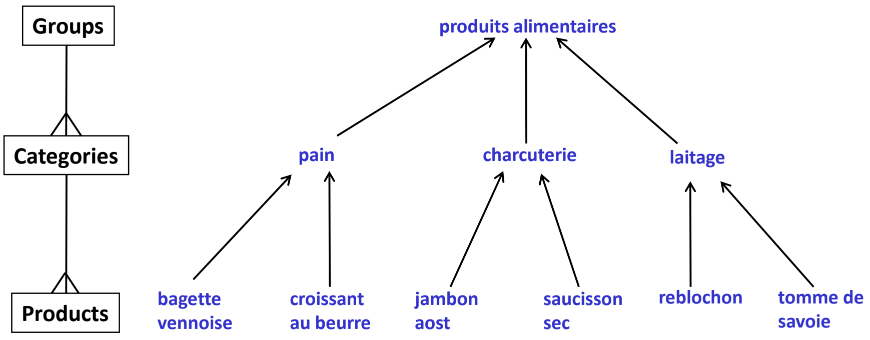
- **Distributive**
  - partial results of aggreg. funcion F on n subsets of set S can be aggregated into a result that is identical to applying F on the whole S
  - upper level aggregate can be computed from lower level aggregates (sales by cities aggregate to sales in a country)
  - count, min, max, sum
- **Algebraic**
  - computed using distributive functions
  - avg, stdev
  - given partial Avg1, Avg2, ...
    - Avg = (Avg1*n1 + Avg2*n2 + ...)/(n1+n2+ ...)
- **Holistic**
  - can be computed only on all elementary data
  - median

Dimension hierarchies

Malinowski E., Zimanyi E.: Advanced Data Warehouse Design. From Conventional to Spatial and Temporal Applications. Springer Verlag, 2008

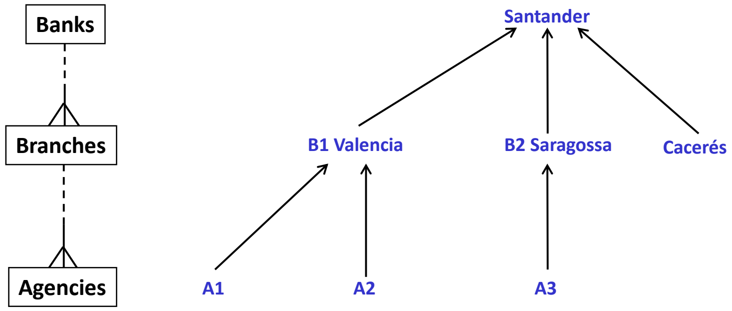
Balanced hierarchy

- **Includes only one aggregation path**
- **1:M relationship both-side obligatory between an upper level and a lower level**



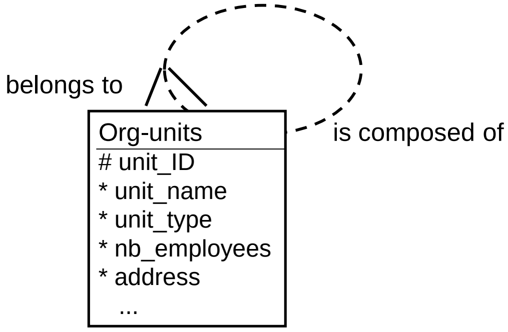
Unbalanced hierarchy

- Includes only one aggregation path
- 1:M relationship between an upper level and a lower level, obligatory only from a lower level



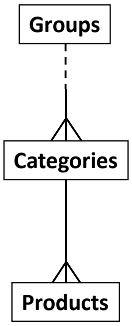
Unbalanced recursive hierarchy

- Special case



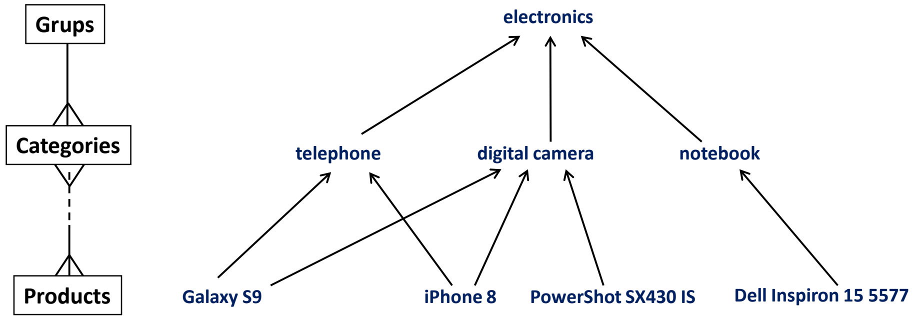
Strict hierarchy

- Dimension schema includes only 1:M relationships between an upper and a lower level
  - special cases: balanced and unbalanced



Non-strict hierarchy

- Dimension schema includes at least one M:N relationships between an upper and a lower level

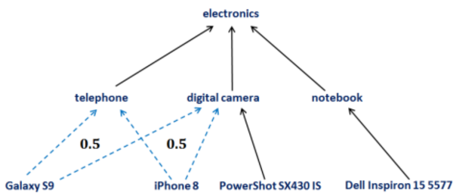


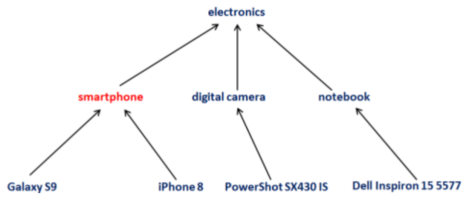
- Problem: aggregating twice the same product

Non-strict hierarchy

➲ Solutions

- distributing a measure value between categories
  - e.g., telephone: 50%, digital camera: 50%
- adding a new category or replacing existing one, e.g., smartphone
- transforming a non-strict into a strict hierarchy → assigning a product to one (main) category

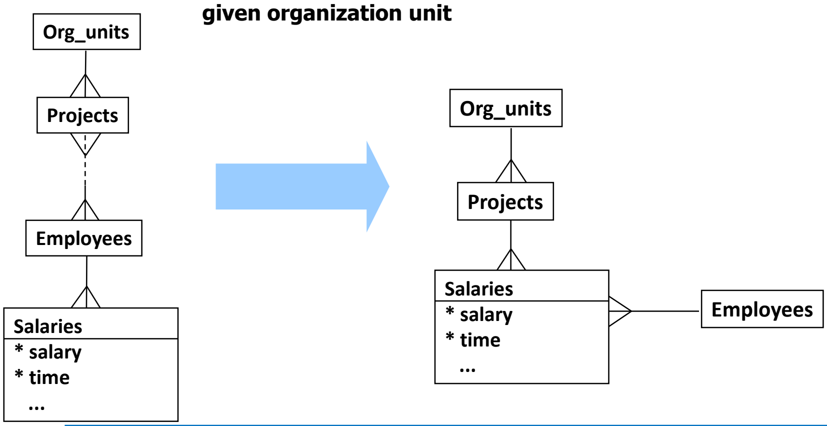




Non-strict hierarchy

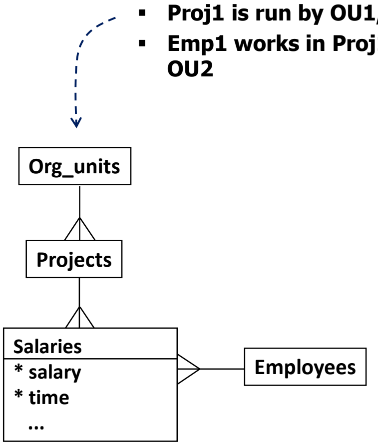
➲ Non-strict → strict transformation

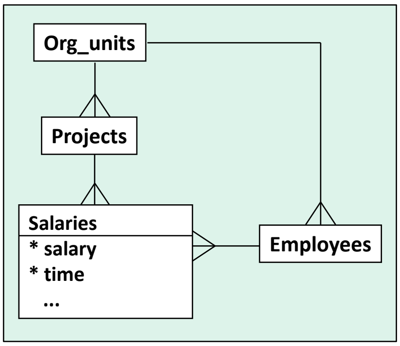
- we must know the distribution of a measure value between dimension level instances
  - e.g., how much an employee earns in a given project run by a given organization unit



Non-strict hierarchy

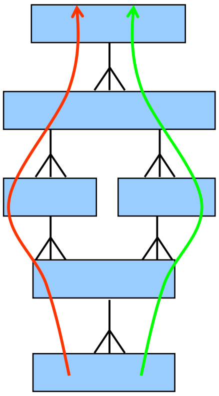
- A problem still exists
  - count employees by organization unit
    - Proj1 is run by OU1, Proj2 is run by OU2
    - Emp1 works in Proj1 and Proj2 → will be counted for OU1 and OU2





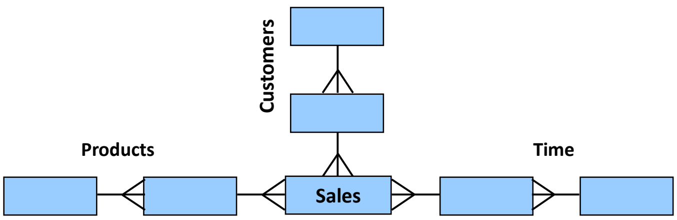
Alternative hierarchy

- Composed of hierarchies that share at least one level
- Aggregating through any path from facts to a shared level yields the same results at the shared level



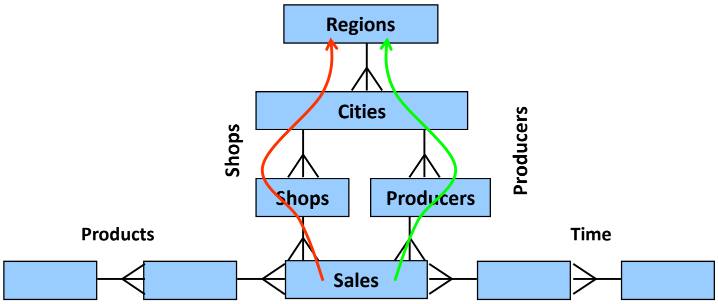
Parallel hierarchies

- **Parallel independent hierarchies ⇨ a standard case**
  - **hierarchies don't share levels**
  - **each hierarchy is an independent context of an analysis, e.g., Prodcts, Customers, Time**



Parallel hierarchies

- **Parallel dependent hierarchies**
  - **hierarchies share levels**
  - **each hierarchy offers a different context of analysis**



- **Sales→Producers→Cities: sales of products delivered by producers located in a given city**
- **Sales→Shops→Cities: sales by shops located in a given city**
- **both analyses yield different results**

MOLAP

Multidimensional Data Model — **conceptual**

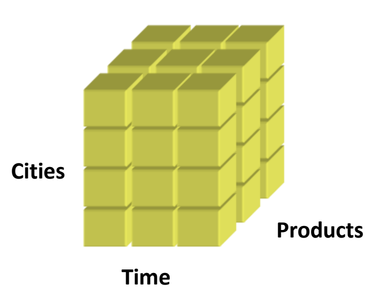
ROLAP
- star schema
- snowflake schema
- starflake schema
- fact constellation schema

MOLAP - Multidimensional OLAP

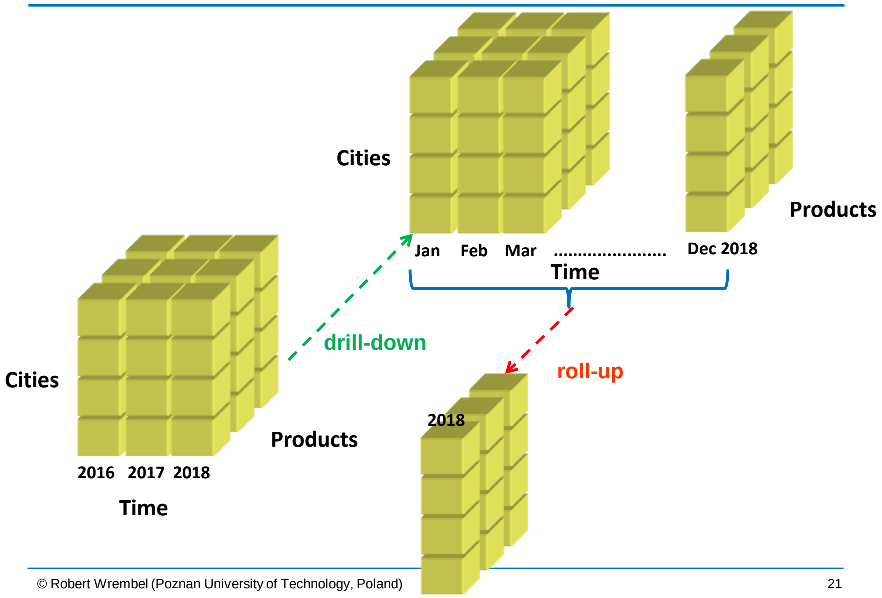
logical / implementation

MOLAP

- **Logical model: data cube (hypercube, md cube)**
- **Operations**
  - **drill-down / roll-up**
  - **slice, dice**
  - **rotate (pivot)**
  - **drill-accross**

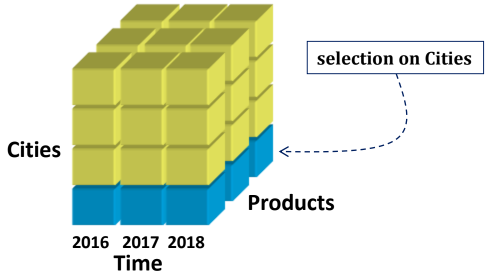


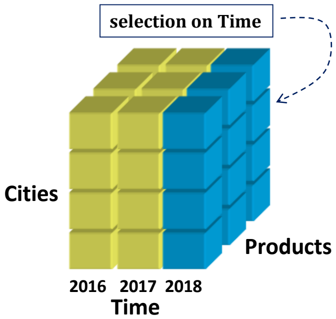
Drill-down / roll-up



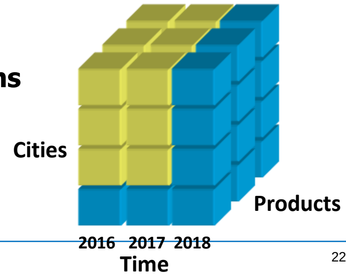
Slice, dice

- **Slice ⇨ selection on 1 dimension**

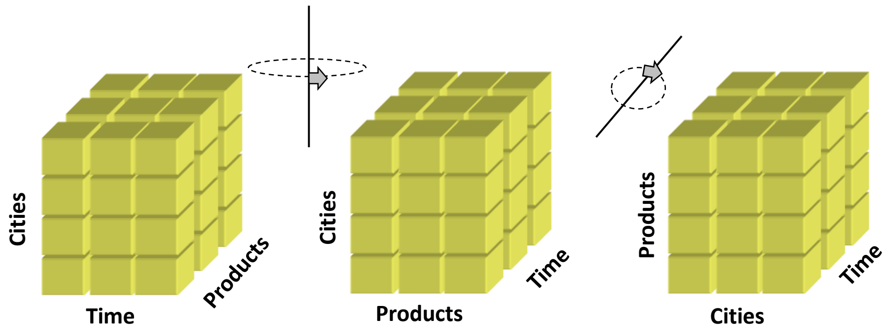




- **Dice ⇨ selection on N dimensions**



Rotate (pivot)



Drill-across

- Analyzing data from 2 or more cubes ⇨ the cubes must have at least one common dimension

MOLAP

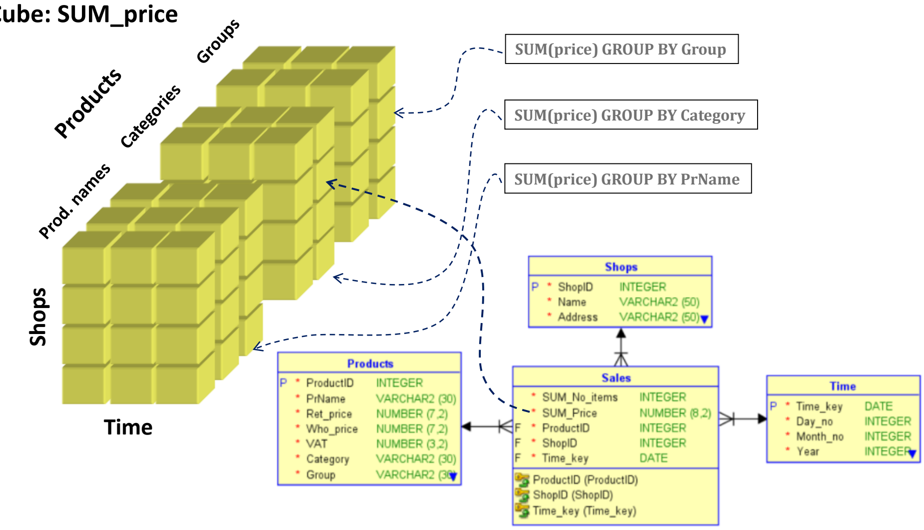
- **MOLAP language**
  - **MDX - MultiDimensional Expressions**
- **Systems supporting MDX**
  - **MS Analysis Services**
  - **SAS OLAP Server**
  - **Oracle Hyperion**
- **Other languages**
  - **OLAP DML**
    - **still in Oracle 12c**
  - **DAX - Data Analysis Expressions**
    - **Power BI, Analysis Services, and Power Pivot in Excel**
    - **library of functions and operators to build formulas and expressions**

MOLAP

- **Cube implementation**
  - **multidim-table**
  - **hash table (SQL Server)**
  - **BLOB (Oracle)**
  - **Quad tree**
  - **K-D tree**

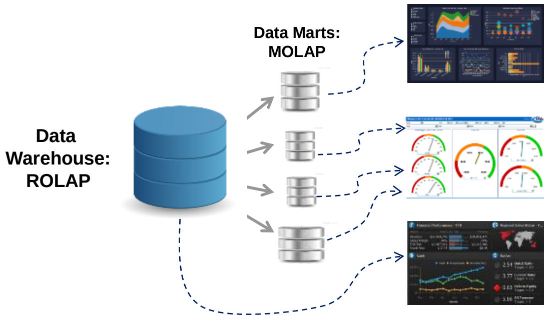
ROLAP vs. MOLAP

Cube: SUM_price



HOLAP

- ➲ **Detailed data, aggregated data → ROLAP**
- ➲ **Topic-wise data, aggregated data → MOLAP**
- ➲ **Central HD + data marts → HOLAP**

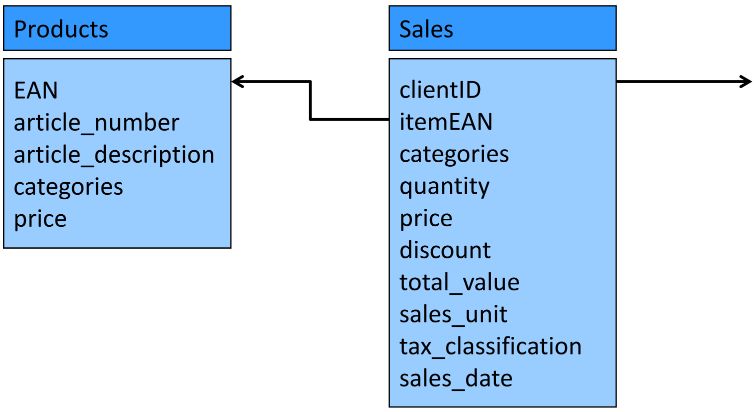


Real case from sales

- **Supermarket chain (real data)**
  - **products & sales**
    - **a total number of products in stock, including one-time-sales and seasonal ones: 15000**
    - **a total number of regular products (always in stock): at least 1400**
    - **a number of individual items sold per year: at least 15 300 000 000**
  - **data changes**
    - **maximum rate of a value change, e.g., price: once per day**
    - **minimum rate of a value change, e.g., product dimension instance: once per 6 months**
    - **average rate of a value change: once every 2 weeks**
  - **data size**
    - **maximum length of a record describing a product (dim): 82B**
    - **maximum length of a record describing a sold item (fact): 64B**

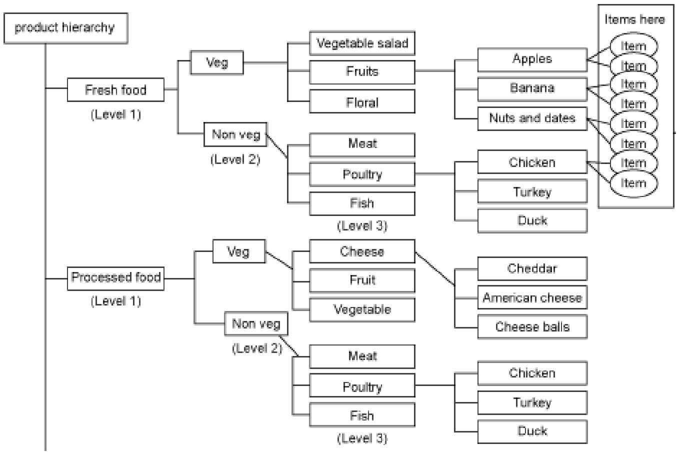
Real case from sales

- **An example dimension Products**



Real case from sales

➲ An example **instance of dimension Products**



Real case: data change

➲ Dimension instances change in time

➲ **Case 1: value change**

- correcting a value error
- price change
- **price** can change every day as the result of:
  - promotion campaign
  - entering a particular holiday period (e.g., prices of chocolates for Christmas and Easter)
  - reacting to price changes of competitors
  - approaching 'best before date'
  - ...

Real case: data change

➲ **Case 2: reference to sales units**

- **a given product, say yoghurt A, is sold in different package units**
  - **in January yoghurt A was sold in 2-packs**
  - **in February → in 4-packs**
  - **in March → in 6-packs**
  - **in April → in 8-packs**
- **in each of these sales periods, a database registers the number of units sold (i.e., n-packs), and not individual yoghurt jars**

Real case: data change

➲ **Case 3: dimension instance change**

➲ **Example 1: the instance of dimension Product changes its hierarchical structure as the result of:**

- **reclassifying products to other categories**
  - **milk → diary → food changed to milk → non vege → food**
  - **electronics → 7% VAT changed to electronics → 22% VAT**
- **removing some level instances**
- **new products offered**

➲ **Example 2: administrative changes to a country**

- **changing territorial organization of a country (Poland 1999; 49 → 16 voivodeships)**
- **East Germany + West Germany**
- **Yugoslavia split**
- **Czech-Slovakia split**

SCD: introduction

- The value of a **descriptor attribute** of a level instance changes in time
- The **hierarchy** of a dimension instance changes
- Both types of changes represent an update to an attribute value
- Need to record the history of such changes
  - 2020: croissant 2.0 EUR
  - 2019: croissant 1.5 EUR
  - 2018: croissant 1.0 EUR
  - `SUM(nb_items*price) where year between 2018 and 2020`
- Such changes are called **Slowly Changing Dimensions**
  - less frequent than changes in the fact table (mainly record inserts)

SCD: introduction

- R. Kimball proposed 7 SCD techniques to record the history of dimension attributes value changes
  - denoted as SCD1-SCD7
  - **3** basic and **4** extended

- R. Kimball, M. Ross: The Data Warehouse Toolkit, 3rd Edition, Wiley, 2013
- Slowly Changing Dimensions: http://www.kimballgroup.com/2013/02/design-tip-152-slowly-changing-dimension-types-0-4-5-6-7/

SCD1: Overwriting

- No history is kept
- The previous attribute value is overwritten by a new one
- Typically, this technique is used for correcting the erroneous data e.g., correcting spelling errors

before update

CustomerID	Name	City	Country
1313	Robert	Poznań	Poland

after update

CustomerID	Name	City	Country
1313	Robert	Bacelona	Spain

SCD2: Record versions

- For every change in the attribute value of a record, a new version of the record is created
- Each record version gets assigned VersionID
- A pair of timestamps is used to identify the validity period of versions

VersionID	CustomerID	Name	City	Country	Valid_from	Valid_until	Current
1	1313	Robert	Poznań	Poland	01-07-1995	31-08-2019	0
2	1313	Robert	Barcelona	Spain	01-09-2019		1

- Full history of changes
- If changes are frequent a table size may grow fast
- Performance may become a concern
- More complex ETL process

SCD3: Attribute versions

- **Typically two versions per attribute are maintained**
  - **the current**
  - **the previous**

simple variant

CustomerID	Name	City_new	Country_new	City_old	Country_old
1313	Robert	Barcelona	Spain	Poznań	Poland

extended variant

CustomerID	Name	City_new	Country_new	Valid_from	City_old	Country_old	Valid_from
1313	Robert	Barcelona	Spain	01-09-2019	Poznań	Poland	01-07-1995

SCD3: Attribute versions

- **Only two versions may be stored**
- **Extended model → storing n versions of an attribute**
- **Problem → a db designer has to know in advance for which attributes versions need to be stored**
- **More complex ETL process**

SCD4: Variant 1

- Original dimension table stores current record → like SCD1
- Additional table stores record versions → like SCD2
- PK: VersionID+CustomerID

current record

VersionID	CustomerID	Name	City	Country
2	1313	Robert	Bacelona	Spain

Fact table

versions

VersionID	CustomerID	Name	City	Country	Valid_from	Valid_until	Current
1	1313	Robert	Poznań	Poland	01-07-1995	31-08-2019	0
2	1313	Robert	Barcelona	Spain	01-09-2019		1

SCD4: Variant 2

- Separates static attributes from volatile attributes
- This approach can also be used to separate frequently queried attributes from rarely queried ones

may be implemented
as SCD2 or SCD3