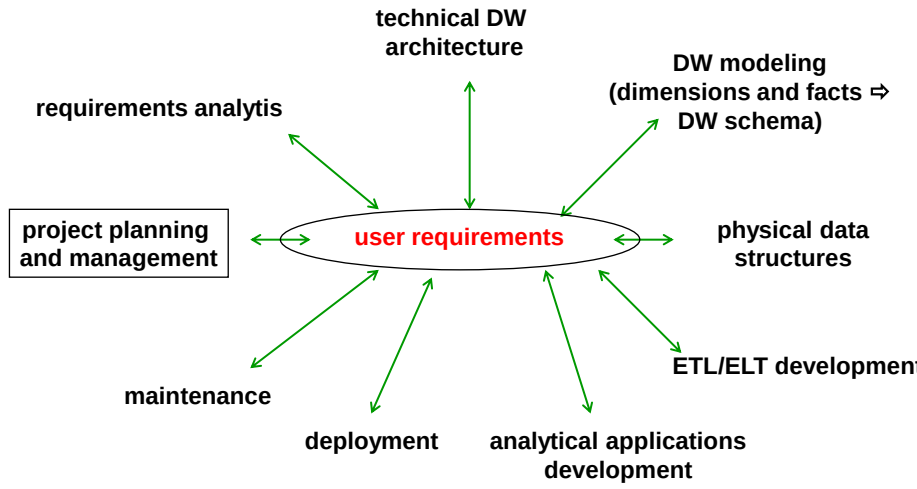
POZNAN UNIVERSITY OF TECHNOLOGY

Modeling Data Warehouse

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

Designing DW System
(R. Kimball's Method)

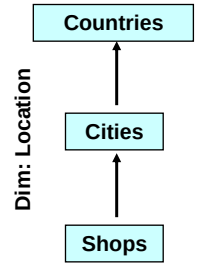


Models

- Conceptual
  - mutlidimensional data model (MD)
- Logical (implemenation)
  - relational →ROLAP
  - multidimensional → MOLAP
  - hybrid → HOLAP
- Slowly Changing Dimensions

Categories of Data

- Facts
  - data to be analyzed in a given context
    - sales, phone calls
    - characterized quantitatively by measures
    - number of intems sold, phone call duration time
- Dimensions
  - define the context of an analysis
    - chocolate sales (product) in Auchan (shop) in months (time)
  - typically composed of levels that form hierachies

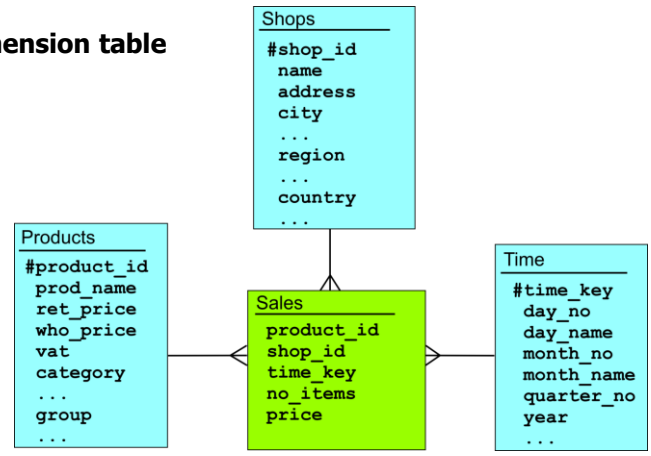


ROLAP

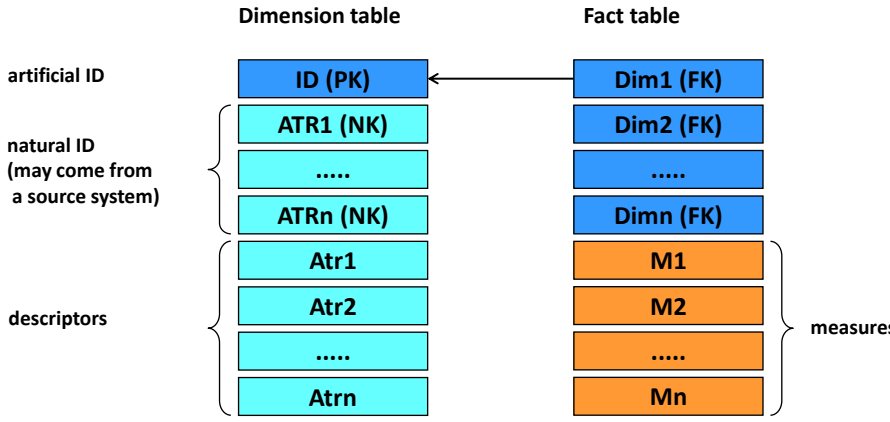
- **ROLAP**
  - **star schema**
  - **snowflake schema**
  - **fact constellation schema**
  - **starflake schema**

ROLAP - star schema

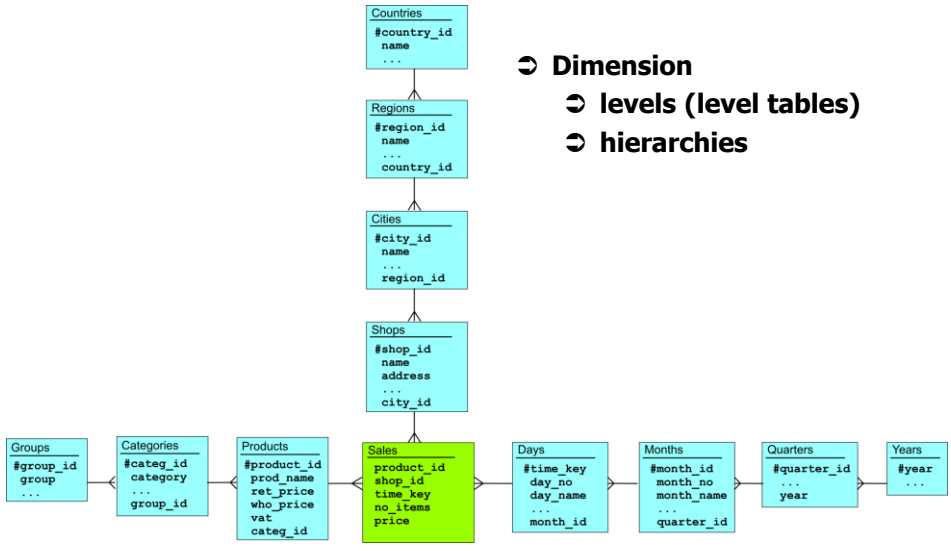
- **One fact table**
  - **measures**
- **At least one dimension table**



Structure of dimension

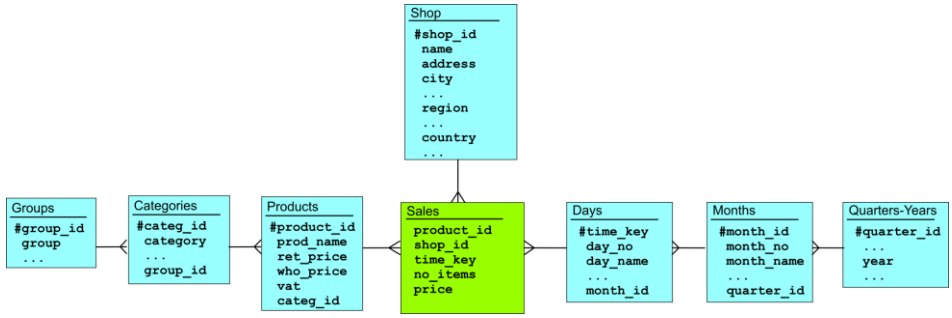


ROLAP - snowflake schema

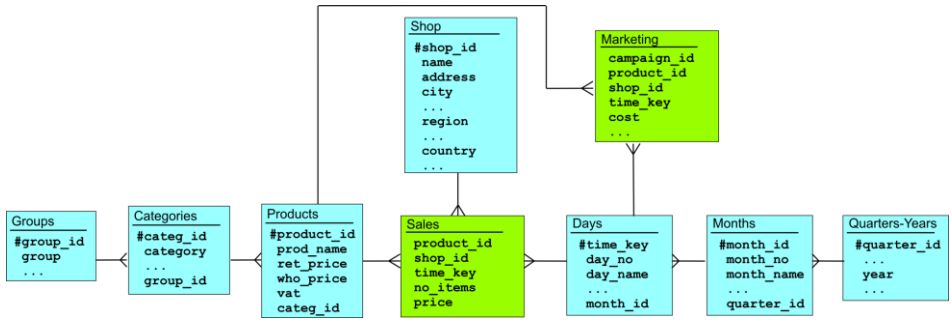


- Dimension
  - levels (level tables)
  - hierarchies

ROLAP - starflake schema



Schemat konstelacji faktów



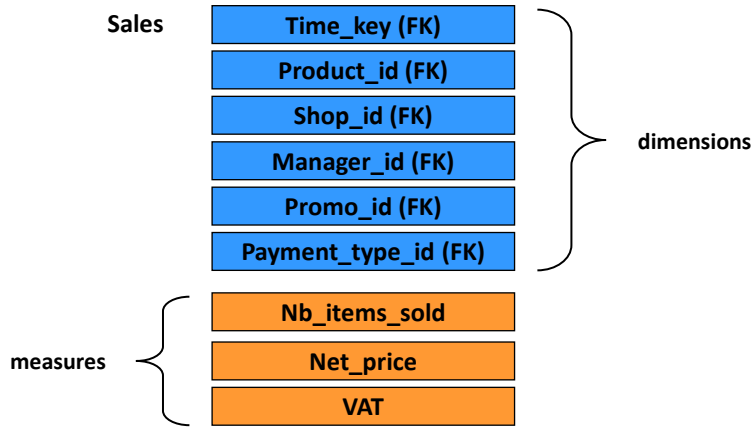
Which schema to use?

- Data redundancy
  - e.g., dimension TIME: 1sec granularity and time span of 10 years ⇨ 300 000 000 rows
    - telecom, physics
- Query performance

Which schema to use?

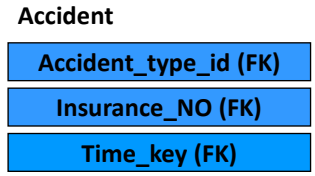
- Hints
  - attributes from different hier. levels frequently used together in roll-up ⇨ store them in one table
  - high level attributes rarely used ⇨ store them in separate high level tables

Fact table



Factless fact

➲ **Stores facts that typically represent events**



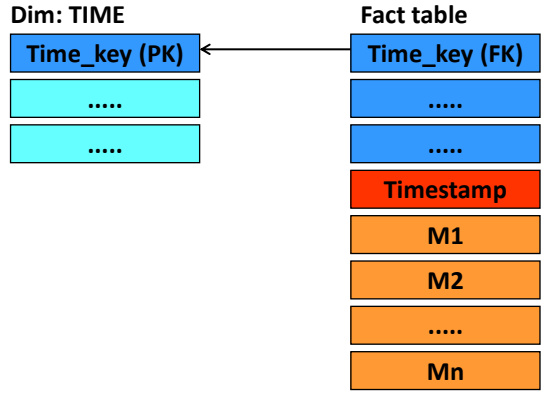
Dimension TIME

- Exists in all DW schemas
- Typical granularity - day
  - Time_key
    - artificial ID: values 1, 2, …, n
    - date-time type
    - timestamp type
    - numerical type: 11032008 (11-03-2008)

Date_id (PK)
Date
Day_name
Day_no_week
Day_no_month
Day_no_year
Last_day_month
Last_day_year
Week_no_year
Month_name
Month_no
Quarter
Year
Fiscal_day_no_week
Fiscal_day_no_month
Fiscal_day_no_year
FIscal_…
…
Holiday
Holiday_type
Shop_open_holiday
Weekend_day
…

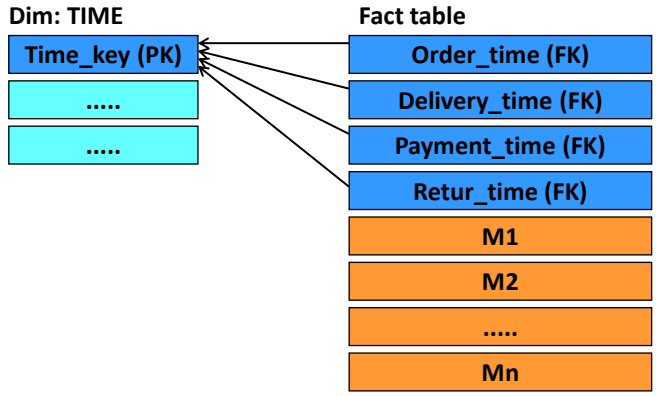
Dimension TIME

- Registering time with granulairty > days
  - timestamp in a fact table



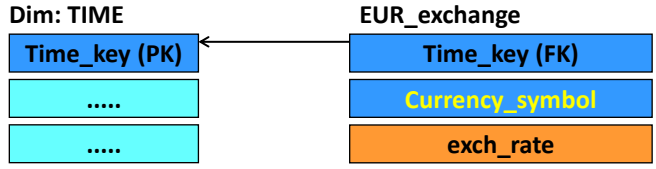
Dimension roles

- The same dimension may play different roles (give contexts) for a fact table
  - e.g., the TIME dimension



Dimension roles

- Fact dimension: a dimension in a fact table



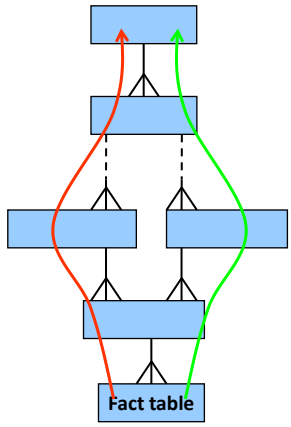
- Measure being a dimension
  - trip length → discretization of values: short, medium, long

Artificial IDs

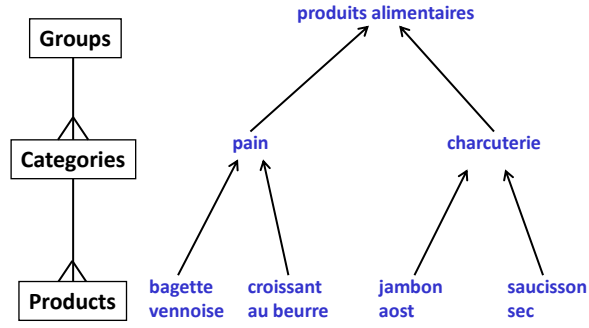
- By an ETL process
- ID types
  - numerical
    - efficiency in processing
    - chronology represented by values
    - no semantics
  - alphanumerical
    - less efficient in processing
    - may have semantics
    - longer than numerical
    - typically concatenation of natural key and timestamp

Definitions

- Aggregation path



- Level instance and dimension instance



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

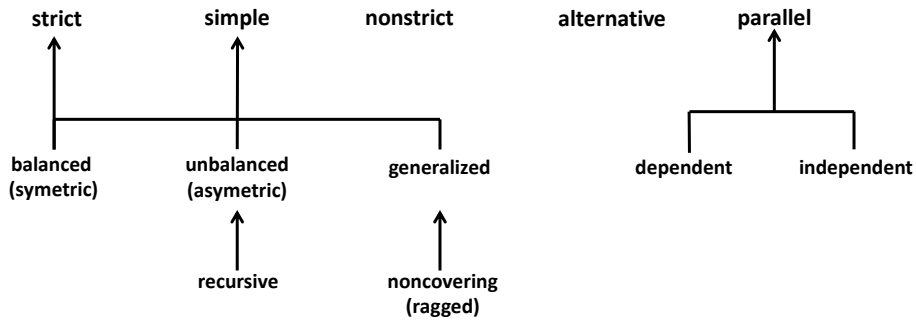
Types of measures

- 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 Timt 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 measures

- **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
- **Holistic**
  - can be computed only on all elementary data
  - median

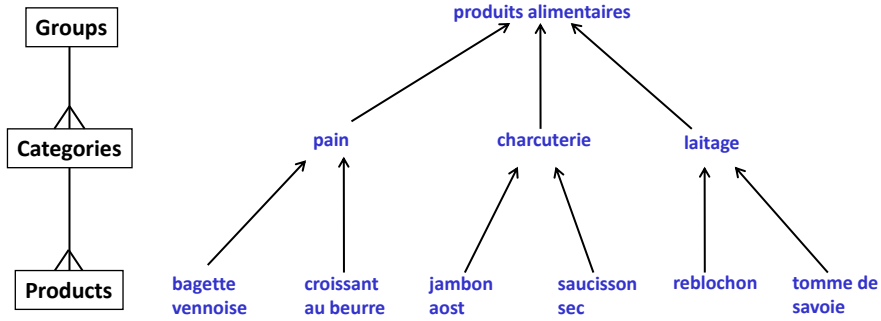
Dimension hierarchies



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

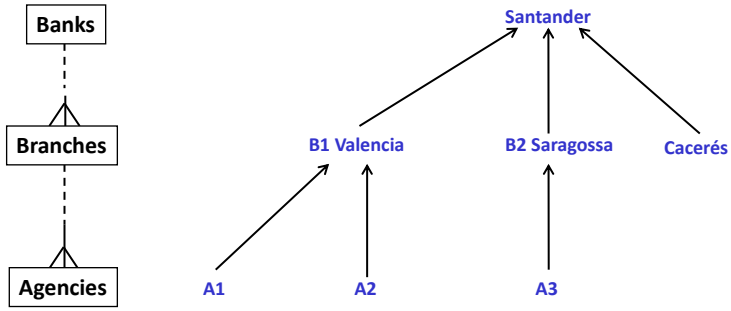
Balanced hierarchy

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



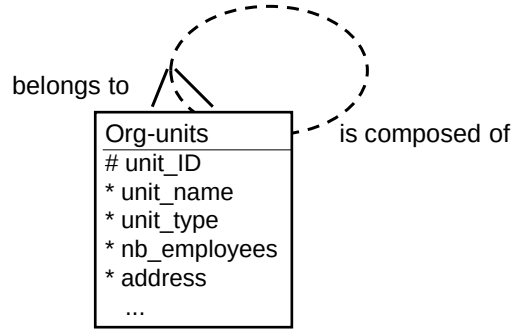
Unbalanced hierarchy

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



Unbalanced recursive hierarchy

- **Special case**

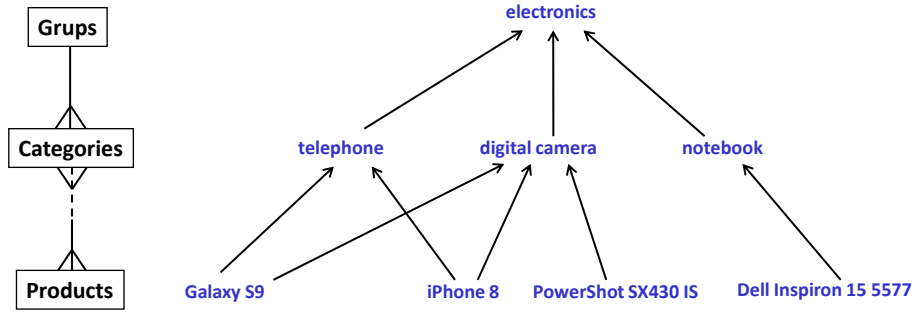


Strict hierarchy

- **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



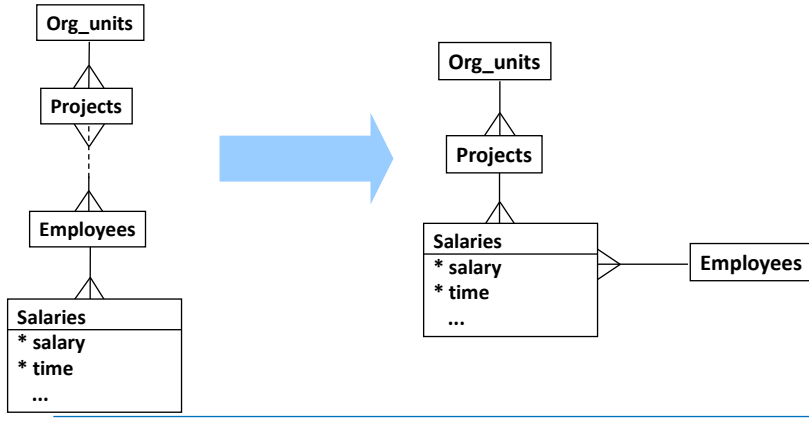
- Problem: aggregating twice the same product

Non-strict hierarchy

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

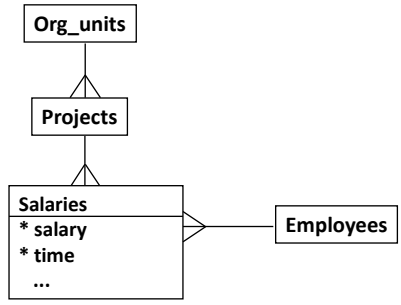
Non-strict hierarchy

- Non-strict → strict transformation
  - we must know the distribution of a measure value between categories
  - e.g., how much an employee earns in a given project



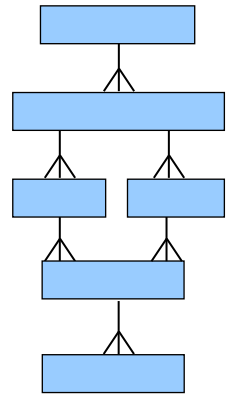
Non-strict hierarchy

- Problem
  - 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



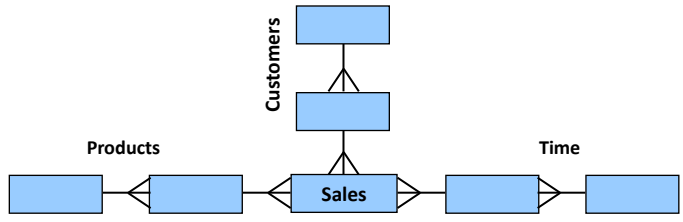
Alternative hierarchy

- 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



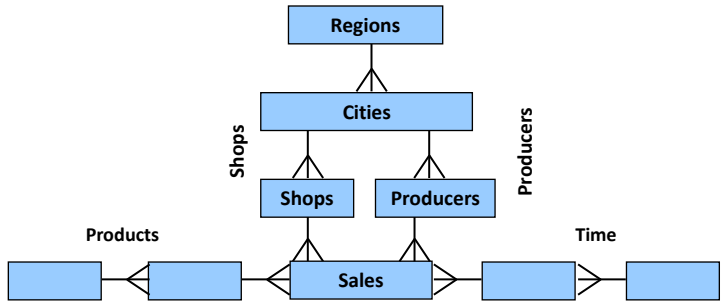
Parallel hierarchies

- 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

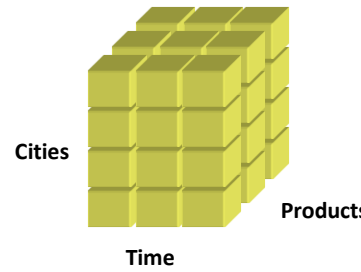
- Parallel dependent hierarchies
  - hierarchies share levels
  - each hierarchy is an independent context of an analysis



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

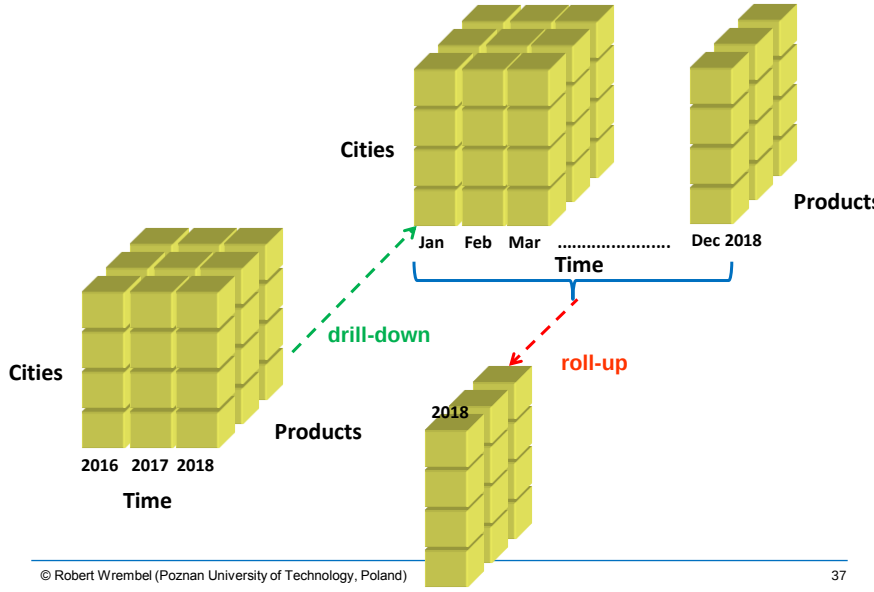
MOLAP

- Logical model: data cube (hypercube)
- Implementation
  - multidim-table
  - hash table (SQL Server)
  - BLOB (Oracle)
  - Quad tree
  - K-D tree
- Operations
  - drill-down / roll-up
  - slice, dice
  - rotate (pivot)
  - drill-accross
  - drill-through
- Language
  - MDX
  - OLAP DML



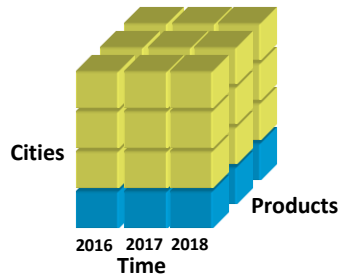
- Systems
  - MS Analysis Services
  - SAS Olap Server
  - Oracle Hyperion

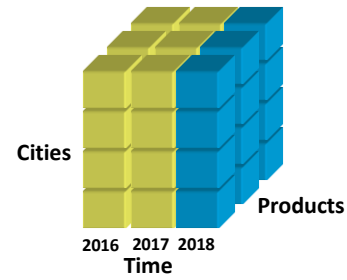
Drill-down / roll-up



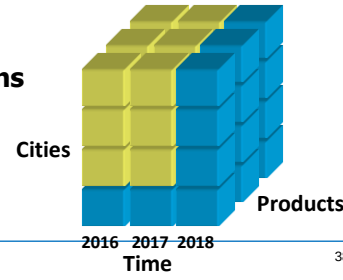
Slice, dice

➲ **Slice ⇨ selection on 1 dimension**

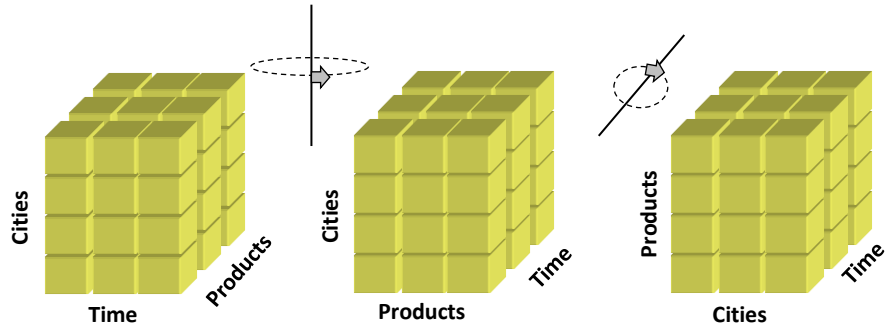




➲ **Dice ⇨ selection on N dimensions**



Rotate (pivot)

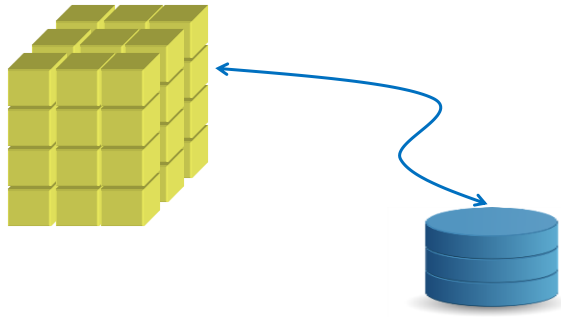


Drill-across

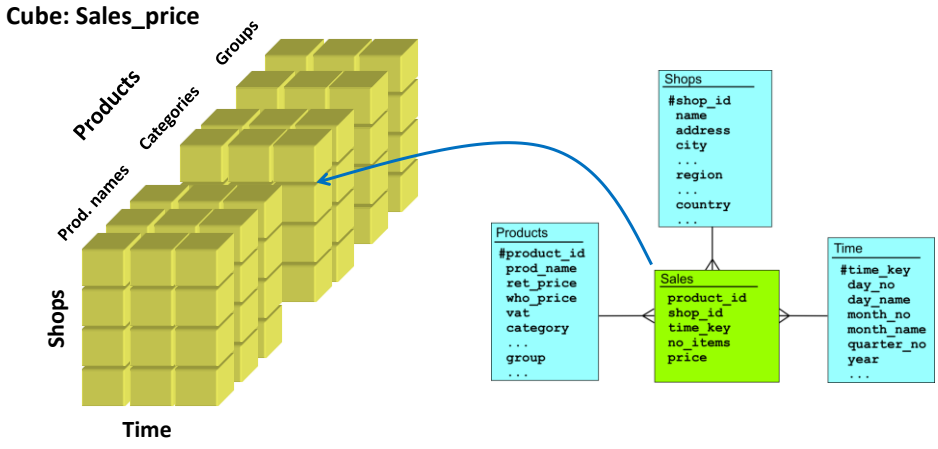
➲ **Analyzing data from 2 or more cubes ⇨ the cubes must have at least one common dimension**

Drill-through

➲ **Accessing a central DW to get more detailed data**



ROLAP vs. MOLAP



HOLAP

- Detailed data, aggregated data (fine grained) → ROLAP
- Topic-wise data, aggregated data (fine grained to coarse) → MOLAP
- Central HD + data marts → HOLAP

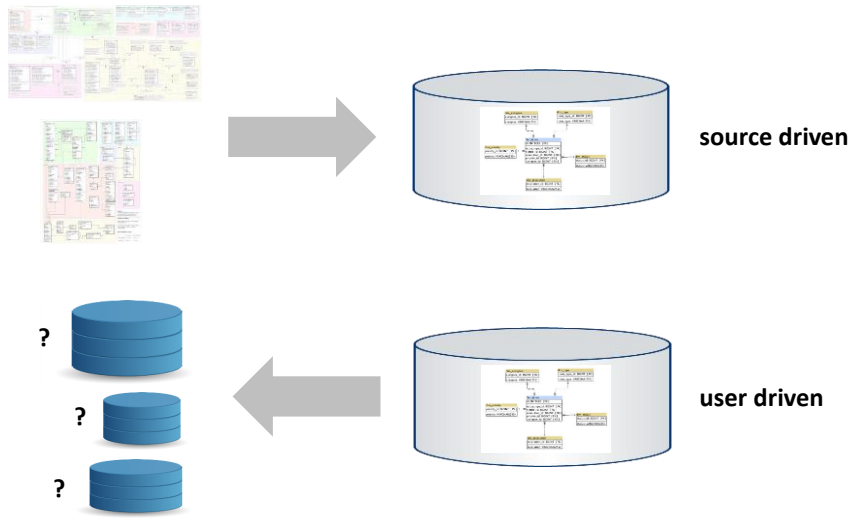
DW modeling: some remarks

- Identify facts → key types of transactions
  - commerce: sales transactions
  - banks: financial operations on accounts
  - stock exchange: sell/buy quotations
  - insurance: buying a policy, damage payment
  - telecom: phone calls
- Identify key dimensions
- Design a fact table
- Design dimension table(s)

DW modeling: some remarks

- Fact data: how detailed?
  - storing every single product purchase record
  - storing the value of a whole basket
  - derived attributes
    - net, vat, gross ⇔ net, vat, and dynamically computed gross
  - storing only necessary attributes
    - dim table Customer with $8*10^6$ records
    - each customer makes daily 2 phone calls
    - one year time span in a fact table
    - decreasing the length of each fact row by 10B → size decrease by 54GB

Designing DW schema



Source-driven

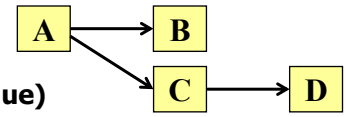
- **Semi-automatic discovery of a DW schema**
  - Song I.-Y., Khare R., Dai B.: SAMSTAR: a semi-automated lexical method for generating star schemas from an entity-relationship diagram. DOLAP, 2007
  - Romero O., Abello A.: Automating Multidimensional Design from Ontologies. DOLAP, 2007
  - Phipps D., Davis K.: Automating Data Warehouse Conceptual Schema Design and Evaluation. DMDW, 2002
  - Jensen M, Holmgren R., Pedersen T.B.: Discovering Multidimensional Structure in Relational Data. DAWAK, 2004

SAMSTAR

- **SAMSTAR**
  - **output: ER DW schema (generated semi-automatically)**
  - **input: ERD of a source system**
- **Steps**
  - **find fiacts and direct dimension levels (automat.)**
  - **find transitive levels (automat.)**
  - **select facts (human)**
  - **select dimensions for selected facts (automat.)**

SAMSTAR

- **Rules for selecting facts and dim. levels**
  - **entities at cardinality of M ⇨ fact candidates**
  - **entities at cardinality of 1 ⇨ level candidates**
- **Direct and indirect dim. level**



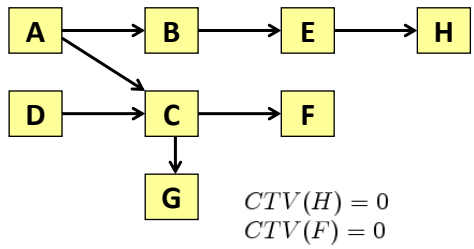
- **CTV (Connection Topology Value)**
  - **sum of weighted direct and indirect levels (transitive)**
  - **the weight of a direct level > the weight of an indirect level**
  - **CTV is computed for each entity in a source schema**

$$CTV(e) = 1 \cdot w_{direct} + w_{indirect} \cdot \sum_{i=1}^{n} CTV(i)$$

SAMSTAR

- **Computing CTV**

weights: $w_{direct} = 1; w_{indirect} = 0.8$



$$CTV(H) = 0$$
$$CTV(F) = 0$$
$$CTV(G) = 0$$
$$CTV(E) = 1 * 1 + 0.8 * CTV(H) = 1$$
$$CTV(B) = 1 * 1 + 0.8 * CTV(E) = 1.8$$
$$CTV(C) = 1 * 2 + 0.8(CTV(G) + CTV(F)) = 2$$
$$CTV(D) = 1 * 1 + 0.8(CTV(C)) = 2.6$$
$$CTV(A) = 1 * 2 + 0.8 * (CVT(B) + CTV(C)) = 5.04$$

SAMSTAR

- Entities for which CTV > ϑ represent facts

$$\vartheta = \frac{\sum_{i=1}^{n} CTV(i)}{n} + k \cdot StDev$$

  - k coefficient whose value is given by a designer; the greater its value the less candidates for fact entities (in practice 1.5 - 1.75)
- Other entites are candidates for dimension
- WordNet for discovering synonyms of entity names

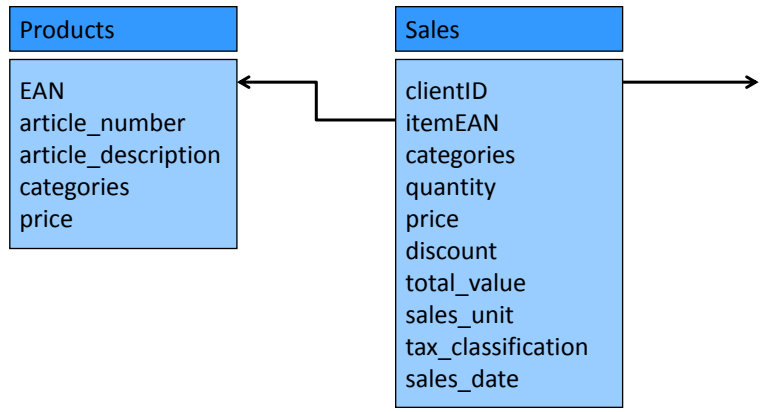
SAMSTAR

- Dim. candidates:
  - entities representing direct levels (at the 1 side of a relationship, where M connects to a fact entity)
  - entities representing indirect levels
- Designing dimensions
  - using WordNet for synonyms
  - using Dimensional Design Pattern (DDP) for defining a dimension structure
    - DDP → a collection of over 100 patterns of typical dimensions
  - adding the TIME dimension from DDP
- Manual acceptance or modification of the result schema

Real case

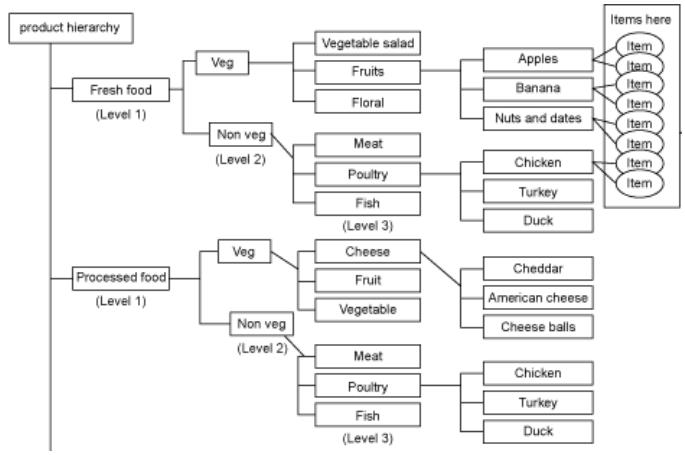
- **Retail chain (real data)**
  - **a total number of products in stock, including one-time-sales and seasonal ones: 15000**
  - **a total number of regular products (always in stock): at least 1400**
  - **maximum rate of a value change, e.g., a price: once per day**
  - **minimum rate of a value change,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

Real case



Real case

➲ **An example instance of dimension Product**



SCD: Introduction

- **Dimension instances change in time**
- **The change could be for**
  - **correcting an error**
    - **e.g., misspelling of a customer's name**
  - **real change in the business world**
    - **e.g., changing product price**
- **Such changes are called Slowly Changing Dimensions, as they are less frequent than changes of facts**

Real case: data change

- Case 1: value change
  - price change
  - it 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
    - reaching a 'best before date'
    - ...

Real case: data change

- Case 2: reference to units of sales
  - 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
  - the instance of dimension Product changes its hierarchical structure as the result of:
  - reclassifying products to other categories
    - milk → diary → food changed to milk → liquids → food
    - electronics → 7% VAT changed to electronics → 22% VAT
  - removing some instances of levels 2, 3, or 4 (typically one instance level is removed)

Real case: data change

- Other cases: dimension instance change
  - administrative changes to a country
    - voivodships in Poland
    - East Germany + West Germany
    - Yugoslavia split
    - Czech-Slovakia split
- New products or services offered

SCD: Introduction

- **The value of an attribute of a dimension table changes in time ⇨ need to record the history of changes**
- **R. Kimball proposed seven (SCD1-SCD7, 3 basic and 4 extended) techniques to record the history of dimension attributes' value changes**
  - 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/

Type 1: Overwrite the old value

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

Type 1: Overwrite the old value

- **Consider that the home city of employee 'Ahmed' is changed from 'Poznan' to 'Brussels'**

EmpID	EmpName	EmpCity
111	John	Poznan
222	Doe	Wroclaw
333	Ahmed	Poznan

Employee table before change

EmpID	EmpName	EmpCity
111	John	Poznan
222	Doe	Wroclaw
333	Ahmed	Brussels

Employee table after change

Type 1: Overwrite the old value

- **No history of the overwritten field is maintained which may be unacceptable in some systems e.g., financial systems**
- **Aggregates on the overwritten field must be recomputed**
- **All distributed copies of the overwritten dimension must be updated simultaneously**

Type 2: Add new dimension record

- **For every change in the attribute value of a dimension member, a new dimension record is added**
- **Each record version is identified by an artificial primary key called a surrogate key**
- **A pair of timestamps is used to identify the validity period of the record version**

Type 2: Add new dimension record

- **Consider employee 'Ahmed' changed his city from Poznan to Brussels in November 2015**

EmpKey	EmpID	EmpName	EmpCity	FromDate	ToDate
1	111	John	Poznan	01/01/2015	31/12/9999
2	222	Doe	Wroclaw	01/01/2015	31/12/9999
3	333	Ahmed	Poznan	01/01/2015	31/10/2015
4	333	Ahmed	Brussels	01/11/2015	31/12/9999

Employee table using Type 2 Changes

- **EmpKey is a surrogate key**
- **FromDate and ToDate together represent the validity period of the record version**
- **ToDate value of the current record is set to a date in distant future, i.e. 31/12/9999**

Type 2: Add new dimension record

- **It keeps the full track of history, but**
- **If the changes are frequent the table size may grow very fast**
- **Storage and performance may become a concern**
- **Complicate ETL processes**

Type 3: Add a new field

- **There are two columns for a particular attribute of interest to represent it's historic (prior) and current values**
- **Recall our example of address change**

EmpID	EmpName	CurrentCity	PreviousCity	EffectiveDate
111	John	Poznan	Poznan	01/02/2015
222	Doe	Wroclaw	Poznan	01/04/2015
333	Ahmed	Brussels	Poznan	01/11/2015

Employee dimension using type 3 changes

Type 3: Add a new field

- **Type 3 technique does not provide the complete history**
  - **if the address changes again, the information that the employee has lived in 'Poznan' will be lost**
- **Upon every change, the values in the current and history columns are overwritten which provokes all the caveats in the case of Type 1 changes**

SCDs: Summary

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change	Facts associated with the attribute's original value
Type 1	Overwrite attribute value	Facts associated with the attribute's current value
Type 2	Add new dimension row for each member with new value	Facts associate with the attribute value in effect when facts occurred
Type 3	Add new column to preserve attribute's current and historic value	Facts associate with both current and historic attribute value
Type 4	Add mini-dimension	Facts associate with rapidly changing attributes when facts occurred

SCDs: Summary

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 5	Add type 4 mini-dimension along with overwritten type 1 mini-dimension key in the base dimension	Facts associate with the attribute's value at the time facts occurred, plus rapidly changing attributes' current value
Type 6	Add type 1 attribute to type 2 base dimension and overwrite all prior values on change	Facts associate with the attribute's value at the time facts occurred, plus the current value
Type 7	Add type 2 dimension row with new values plus the view limited to the current rows and/or attribute values	Facts associate with the attribute's value at the time facts occurred, plus the current value