Universitas Esa Unggul

Smart, Creative and Entreprenurial

Data Warehouse

Munawar, PhD

Session 08

Data Warehouse Logical Design

Agenda

- Concept of Logical Design
- Logical Design Process
- Basic Operations
- The Way Data Are Stored

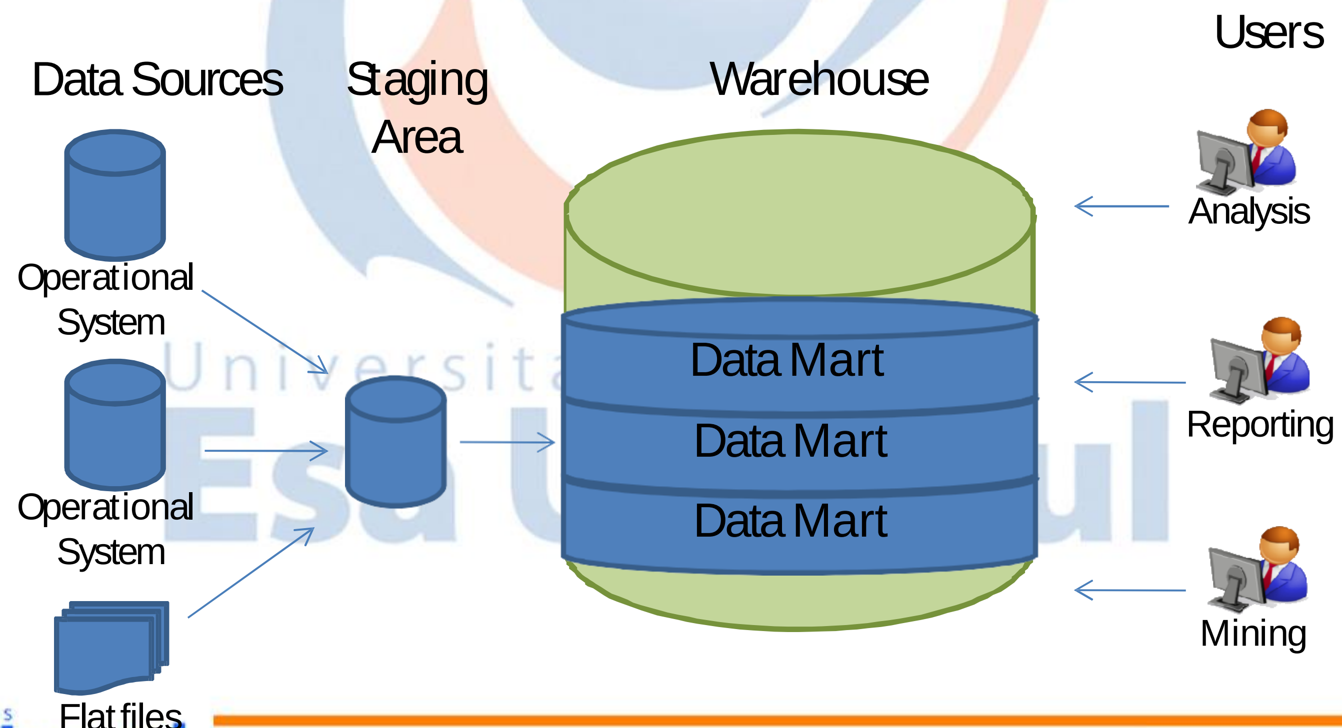
Concept of Logical Design

Intro to Logical Design

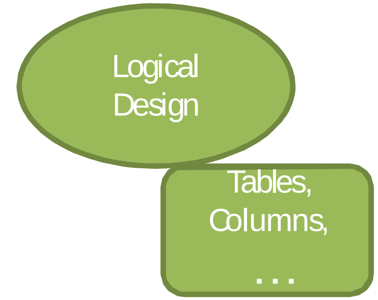
- Logical design is the most attractive step given that it presents tremendous benefits to system performance.
- It is intended to obtain conceptual schemata based on the data structure that will be applied by a DM or DW, with consideration for a number of constraints, particularly those related to disk space or query retrieval (Trujilo et al, 2000).
- Logical design is relevant in a relational OLAP (ROLAP) environment, and a logical model can be easily derived in a multidimensional OLAP (MOLAP) environment.

Logical Data Mart

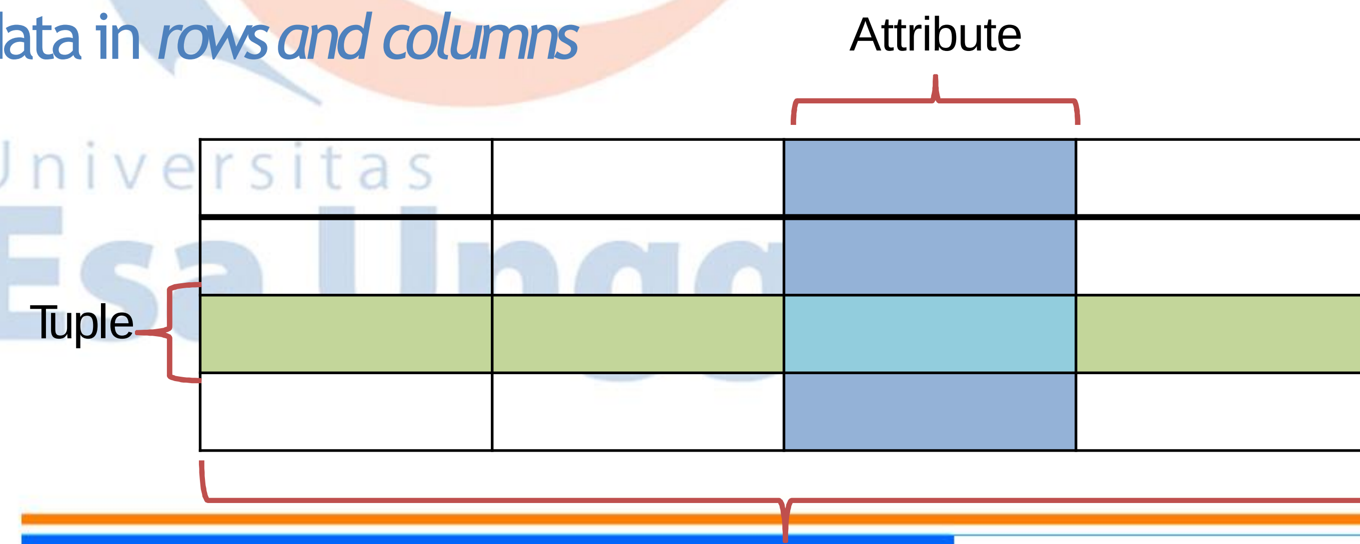
- Data Marts are *not* separate databases, but logical views of the D W
  - Integrated view of the enterprise



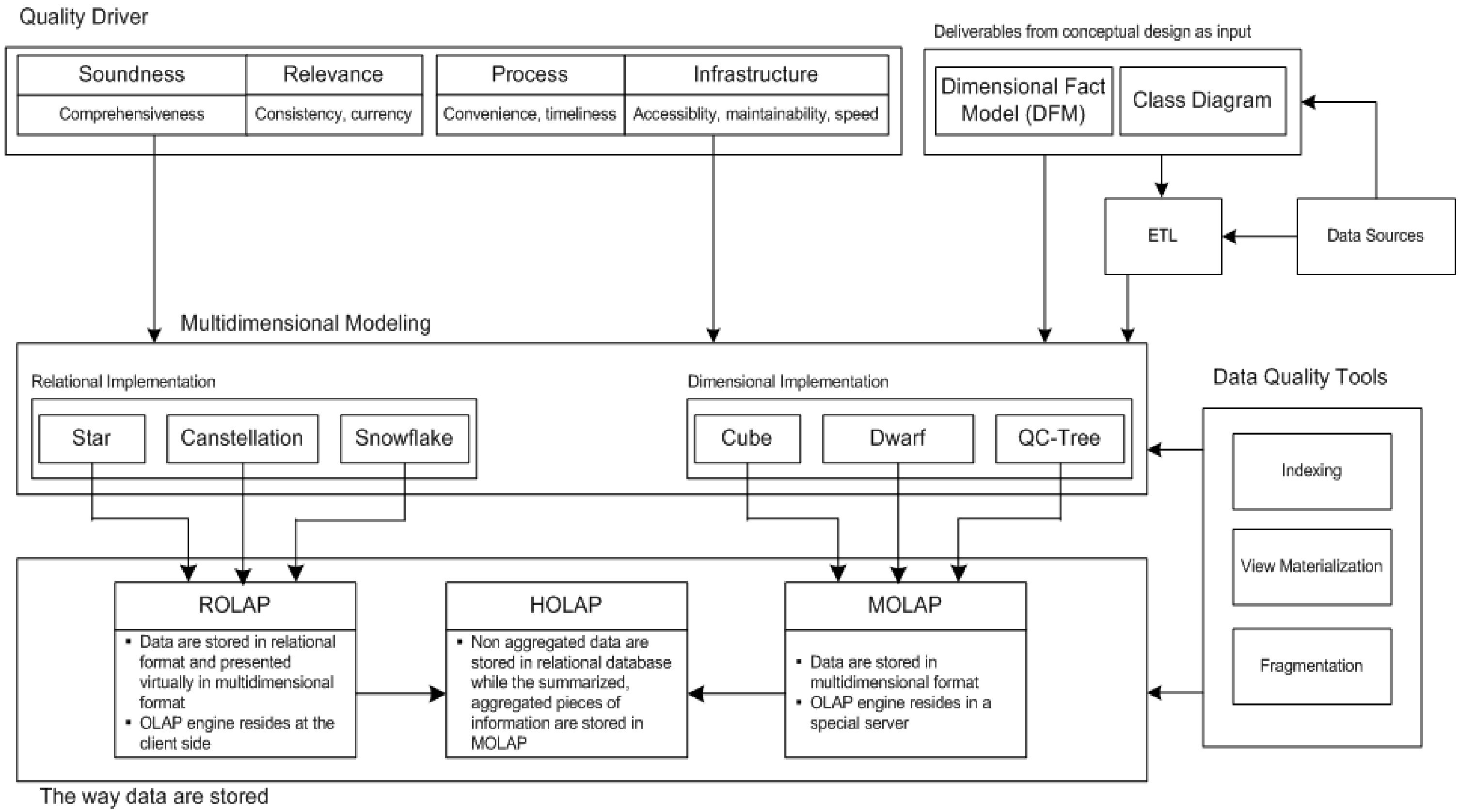
Logical Model



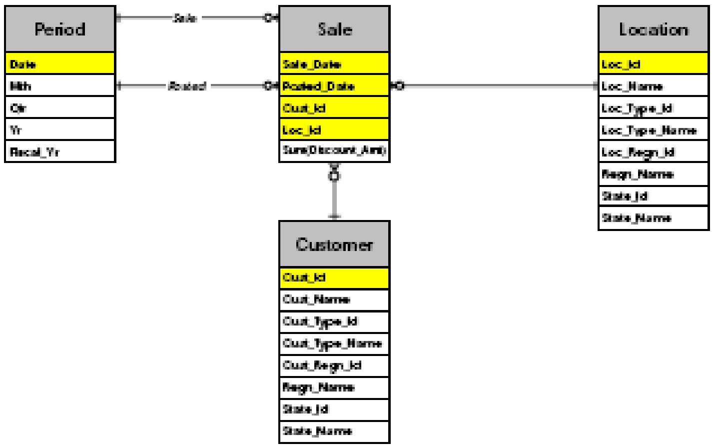
- Logical design **arranges data** into a logical structure
  - Which can be mapped into the **storage objects** supported by DBMS
    - In the case of RDB, the storage objects are *tables* which store data in *rows and columns*



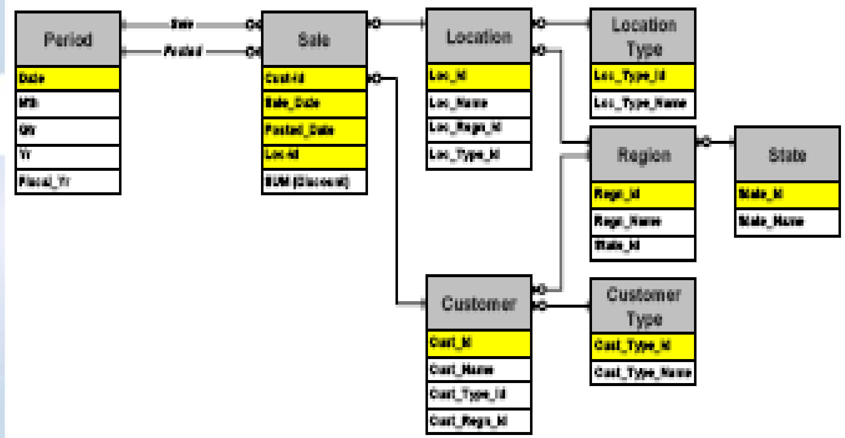
DW Logical Design



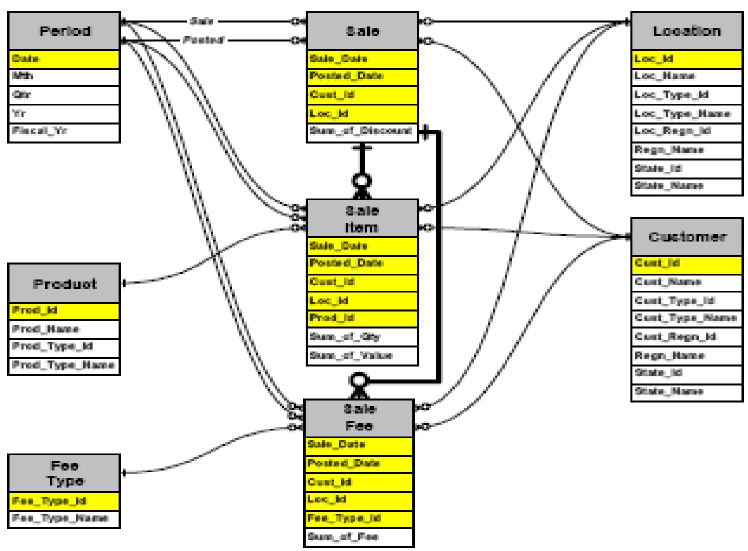
Relational Implementation



Star Schema



Snowflake Schema



Constellation Schema

Compariosn Between Relational Implementation

	Star Schema	Fact Constellation Schema	Snowflake Schema
Efficiency	High	High	Moderate
Usability	High	Moderate	Moderate
Reusability	Low	Low	High
Flexibility	High	High	Moderate
Redundancy	High	High	Low
Complexity	Low	Moderate	Moderate

Comparison…

DW schema	Advantages	Drawbacks
Star schema	• The simplest structure (Moody and Kortink, 2008) • Reduces number of tables and therefore enables optimisation (Basaran, 2005) • The number of relationships between the tables can be reduced (Basaran, 2005). • The number of joins needed in user queries can be reduced (Basaran, 2005). • Query performance can be accelerated.	• Very inflexible (Teklitz, 2000) • For every gigabyte of row data, a schema will require at least an additional gigabyte for aggregations (Teklitz, 2000) • Requires considerable development maintenance effort to manage schema-oriented DW (Teklitz, 2000)
Fact constellation schema	• Storage space can be saved through reusable dimension tables (Levene and Loizou, 2003).	• It may not be helpful for small organisations because of its complexity (Feng et al, 2004)
Snowflake schema	• Hierarchical structures of each dimension can be shown explicitly (Teklitz, 2000) • Intuitive and easy to understand (Arfaoui and Akaichi, 2012) • Data aggregation can be accommodated (Arfaoui and Akaichi, 2012) • Easy to extend through additional new attributes without inference with existing database programmes (Arfaoui and Akaichi, 2012).	• Increases unnecessary complexity (Teklitz, 2000) • Diminishes query performance (Teklitz, 2000)

Multidimensional Implementation

DW schema	Advantages	Drawbacks
Star schema	• The simplest structure (Moody and Kortink, 2008) • Reduces number of tables and therefore enables optimisation (Basaran, 2005) • The number of relationships between the tables can be reduced (Basaran, 2005). • The number of joins needed in user queries can be reduced (Basaran, 2005). • Query performance can be accelerated.	• Very inflexible (Teklitz, 2000) • For every gigabyte of row data, a schema will require at least an additional gigabyte for aggregations (Teklitz, 2000) • Requires considerable development maintenance effort to manage schema-oriented DW (Teklitz, 2000)
Fact constellation schema	• Storage space can be saved through reusable dimension tables (Levene and Loizou, 2003).	• It may not be helpful for small organisations because of its complexity (Feng et al, 2004)
Snowflake schema	• Hierarchical structures of each dimension can be shown explicitly (Teklitz, 2000) • Intuitive and easy to understand (Arfaoui and Akaichi, 2012) • Data aggregation can be accommodated (Arfaoui and Akaichi, 2012) • Easy to extend through additional new attributes without inference with existing database programmes (Arfaoui and Akaichi, 2012).	• Increases unnecessary complexity (Teklitz, 2000) • Diminishes query performance (Teklitz, 2000)

Optimization Techniques

- Index
- Materialized View
- Data Fragmentation (Aouiche, 2005)
- Without optimisation techniques, queries may take hours or days to execute because of the high complexity of queries that are related to a large number of joins with dimension tables

Indexing Technique

Indexing techniques	Advantages	Drawbacks
Bitmap indexing	• Widely used in DW environment • Response time can be minimised. • Storage needs can be minimised compared with other indexing techniques • Dramatic performance for a small amount of memory or CPU • Efficient maintenance	• Slow performance for high-cardinality column data. • More work is required if index is modified. • Concurrency occurs if any modification on bitmap indexes is inequitable.
Cluster indexing	• Performance can be optimised. • Good for range-based queries but needs sorted data	• Increasing sorting costs for unsorted data • Costly operation because the re-ordering of data is needed for data insertions (Davidson, 2008; Aizawa, 2002)
Hash-based indexing	• Large amounts of data can be minimised (Delmarco, 2006). • Average look-up cost can be minimised through hash function, bucket table size and internal data structures. • No key to be sorted • Best option for equality selections	• Leads to collusion • Range queries is unsupported • Leads to long chains in static hashing • Impossible for hash reversing

Fragmentation Technique

- Vertical → splits tables by column; one table is divided into two or more tables →
- Horizontal → splits tables by row; the tables are the same as those in the original, except that the rows are split → to minimise irrelevant data access → is designed for partitioning a relation into a set of smaller relations so that only one fragment is executed by many applications
- Hybrid → horizontal fragmentation followed by vertical fragmentation or vice versa

Aggregation

- Aggregates are needed in case of high load predictable queries exist. In such a case, faster response can be obtained by aggregates, as well be having results already stored in aggregates. Summary data must be applied only in critical condition.
- Based on practical experience, in case of small data in the fact table, no aggregation is needed in such case
- Commonly, selective aggregation is used, depending upon the requirements of organisation and often asked questions
- The possible total number of aggregations can be defined by simply multiplying the number of levels in each dimension hierarchy.

Aggregation can be stored

- As new field in existing fact table.
- As a new field in existing fact table, aggregations face the following issues:
  - ✓Issue of double counting.
  - ✓Aggregation can be seen by the user.
- As new fact table
- As new fact table, the following benefits can be obtained compared with previous method:
  - ✓Double counting issue resolved.
  - ✓Aggregation unseen by the user.
  - ✓Easily updated in future without any problem to tables.
  - ✓Field size of aggregation does not affect the size of the field for the basic data.

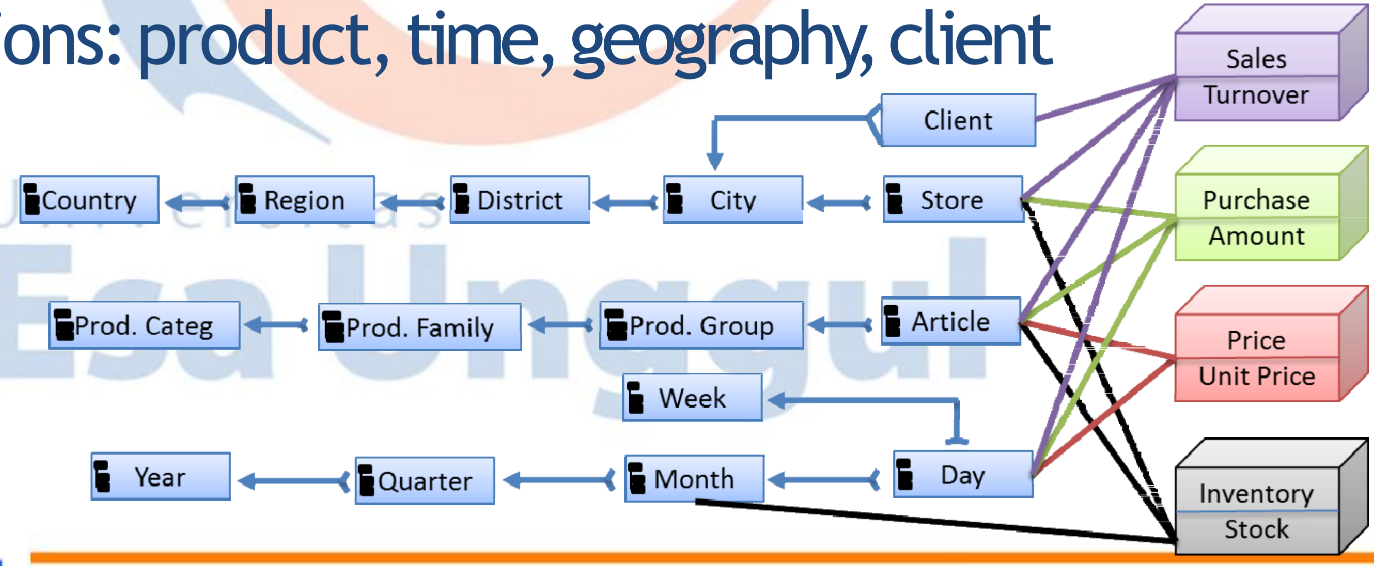
Typical Query on DW

(1) Roll-up: aggregate fact attributes to view data at a higher level of abstraction.
(2) Drill-down: disaggregate fact attributes in order to introduce further details.
(3) Drill-cross: relate and compare distinct facts.
(4) Slice-and-dice: select and project facts so as to reduce their dimensionality

Logical Design Process

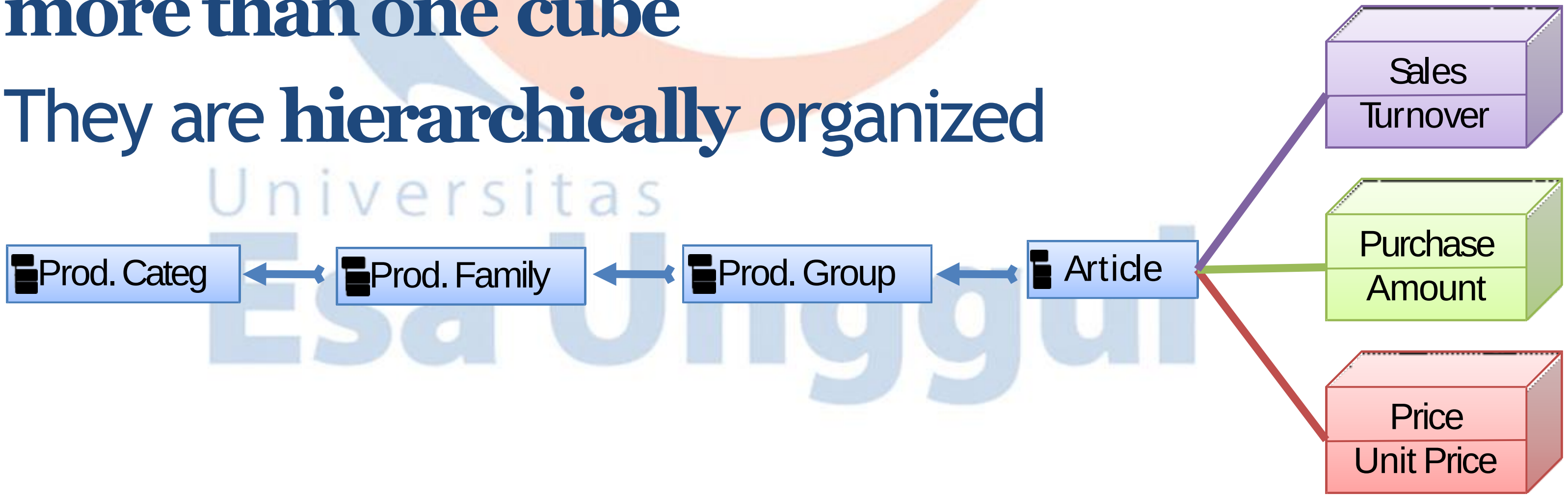
Logical Model…

- **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

- **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

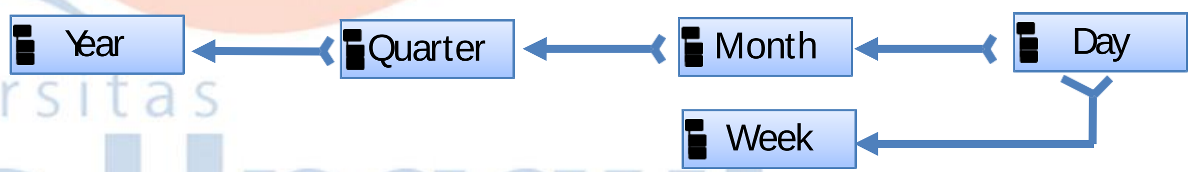


Dimensions…

- 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 $A \longrightarrow B$, if for all $a \in dom(A)$ there exists exactly one $b \in dom(B)$ corresponding to it

Dimensions...

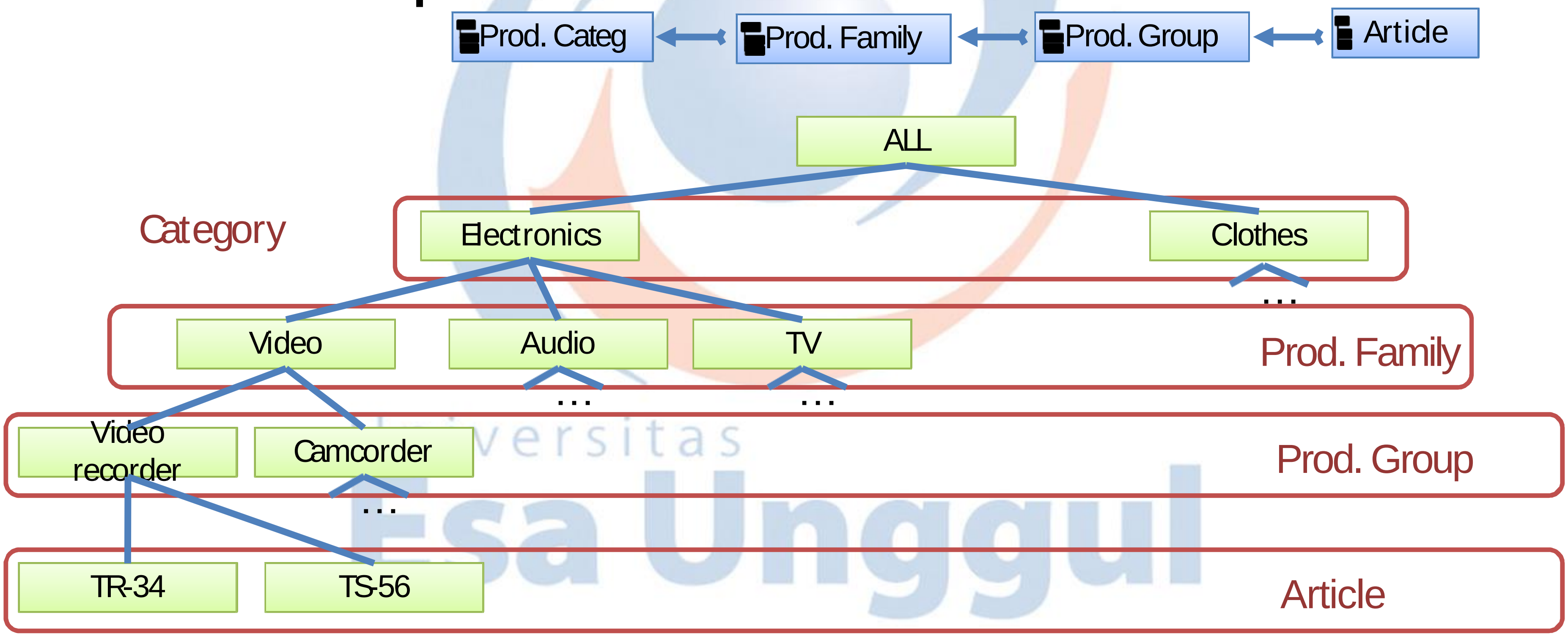
- A **fully-ordered** set of classification levels is called a **Path**
  - If we consider the **classification schema** of the time dimension, then we have the following paths
    - T.Day → T.Week
    - T.Day → T.Month → T.Quarter → T.Year



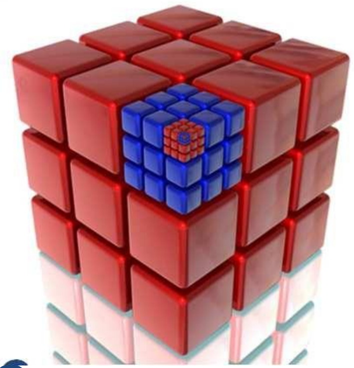
  - Here T.Day is the **smallest element**

Dimensions…

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



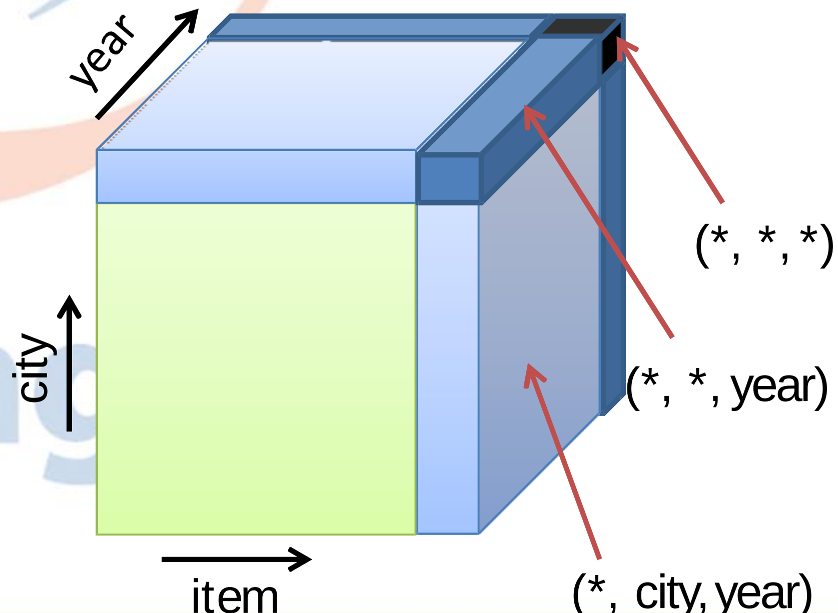
Cubes



- 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))

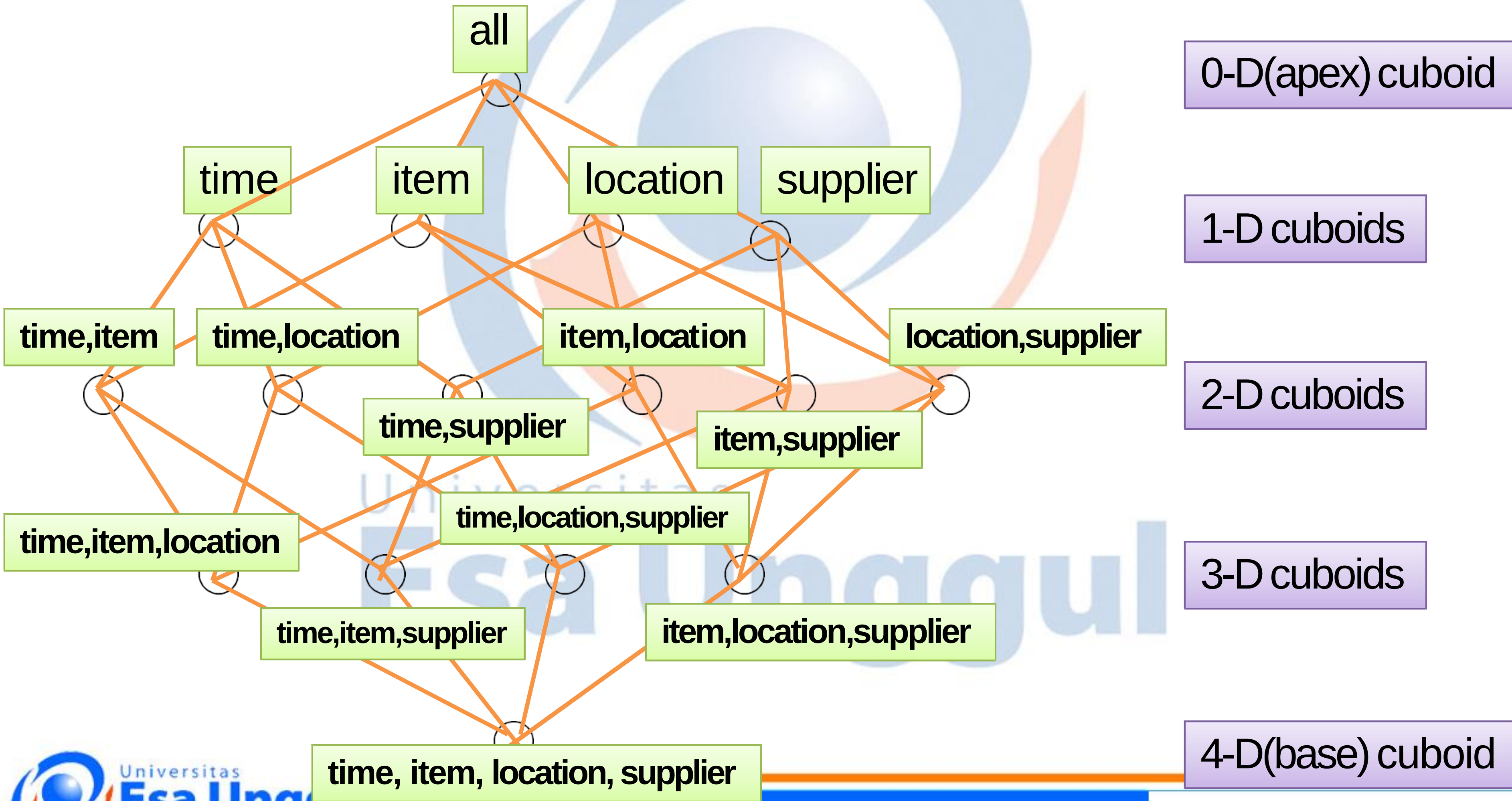
Cubes…

- 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 (*,*,*)



Cubes…

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



Basic Operation

Basic Operations

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

Selection

- 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. $\sigma_{P.Prod\_group=\text{“Video”}}(Sales)$

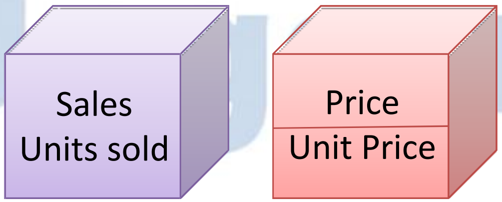
Projection

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

    $n_{F(M)}(C) = \{ (g,F(m)) \in dom(G) \times dom(F(M)): (g,m) \in C\}$
  - E.g. $n_{turnover,\ sold\_items}(Sales)$

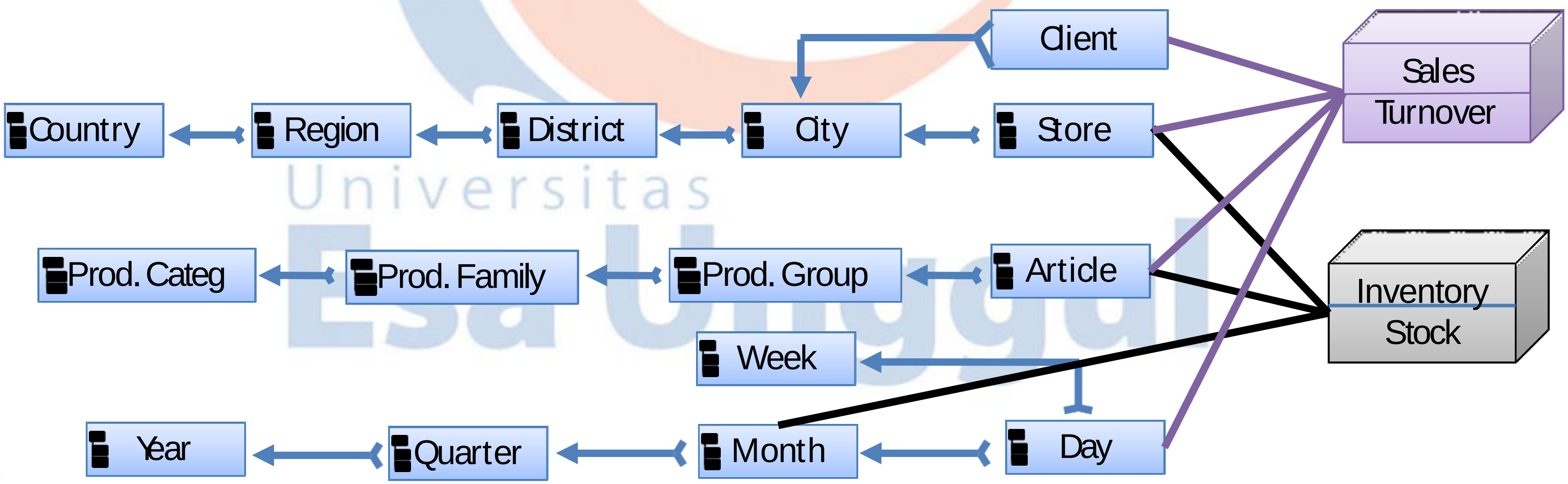
Join Operations

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



Granularities

- 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))

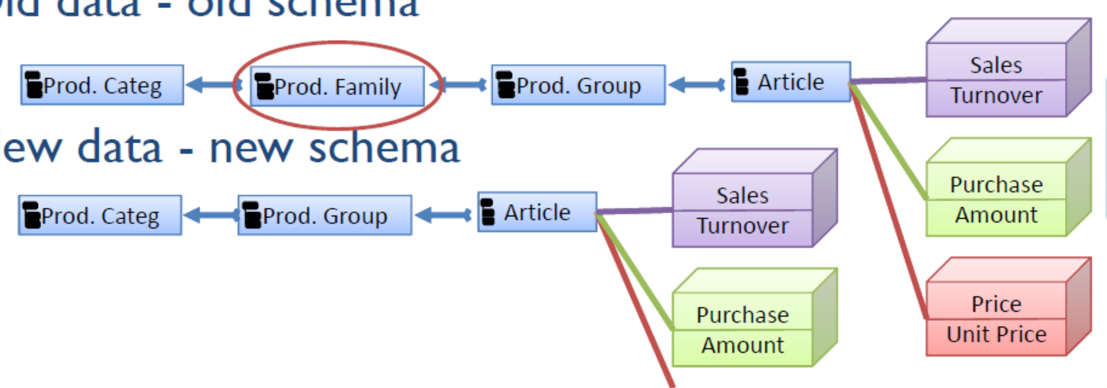


Aggregation

- **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: $SUM_{(P.Product\_group,\ G.City,\ T.Month)}(Sales)$

Schema Versioning

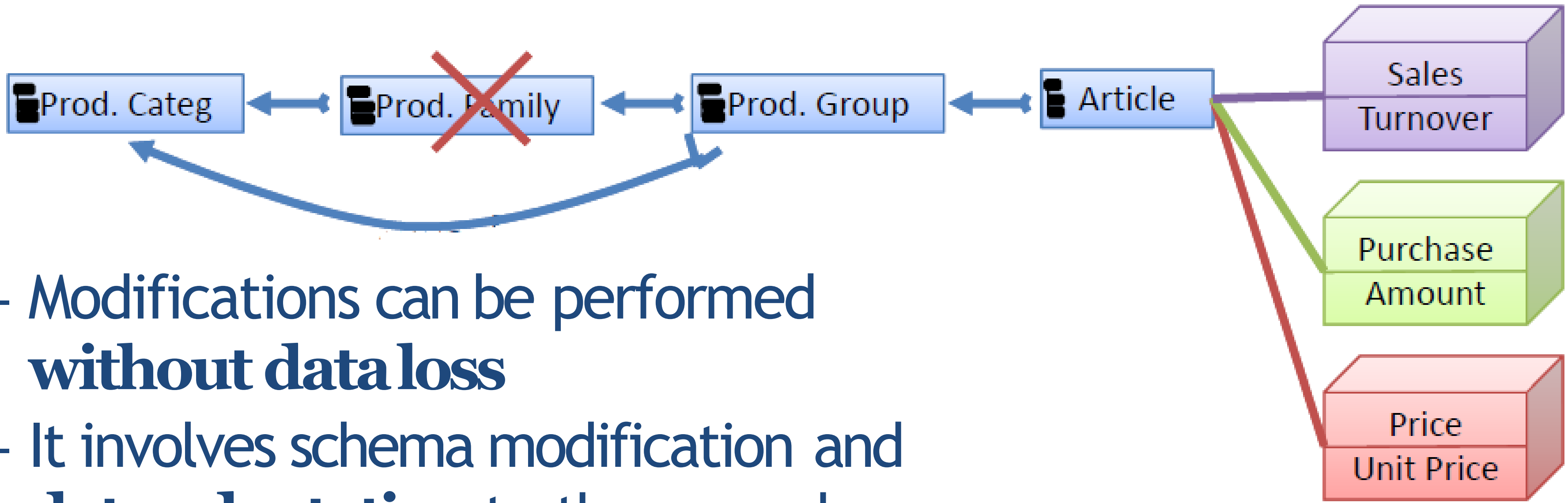
- **No data loss**
- All the data corresponding to all the schemas are always available
- After a schema modification the data is held in their belonging schema
  - Old data - old schema
  - New data - new schema



Schema Versioning…

- 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

Schema Versioning…



- 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

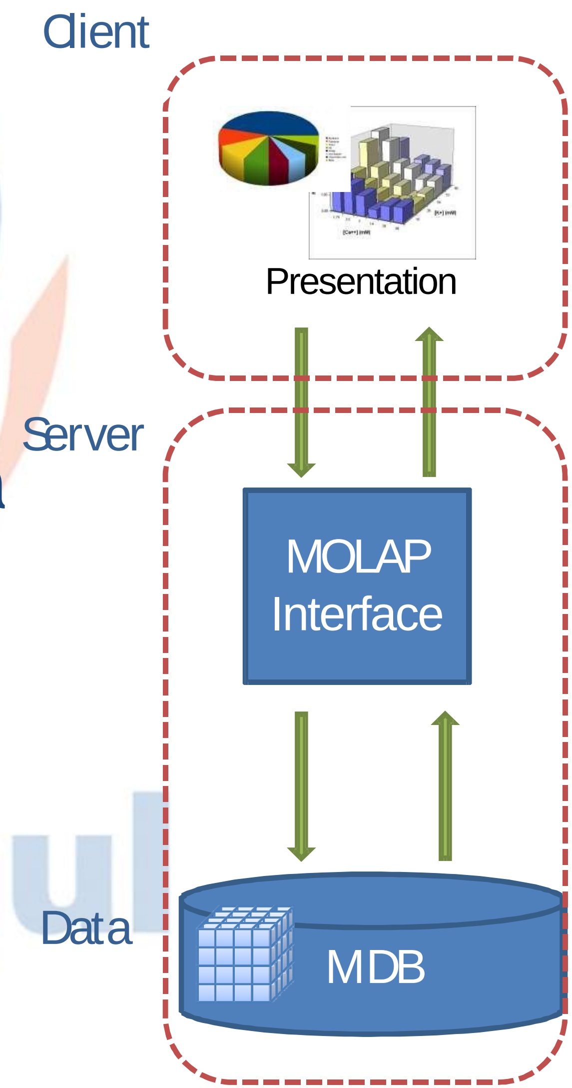
The Way Data are Stored

The Way Data are Stored

- The way data are stored:
  - MOLAP (Multidimensional OLAP)
  - ROLAP (Relational OLAP)
  - HOLAP (Hybrid OLAP)
  - DOLAP (Desktop OLAP)

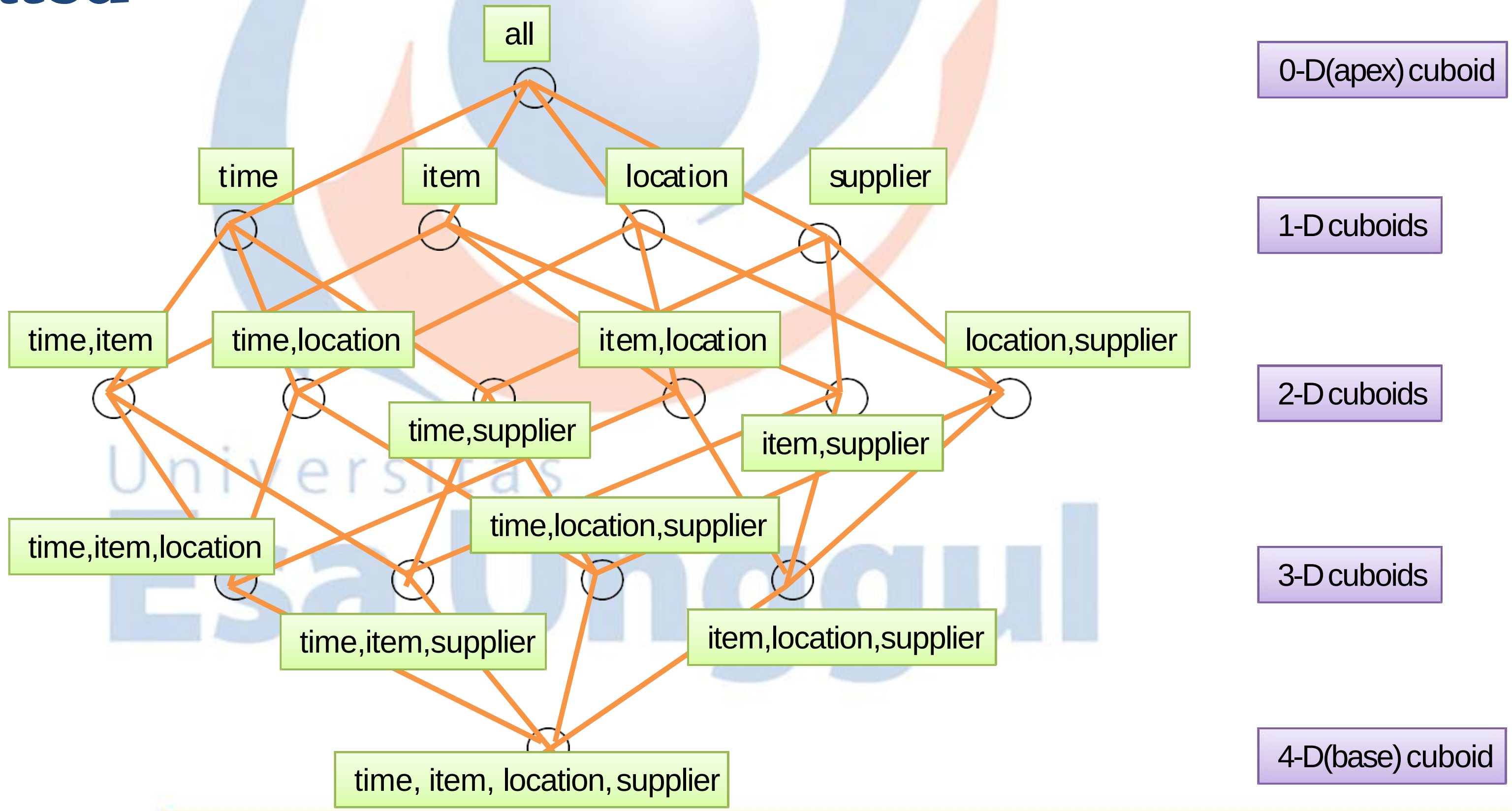
MOLAP

- **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)



MOLAP

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

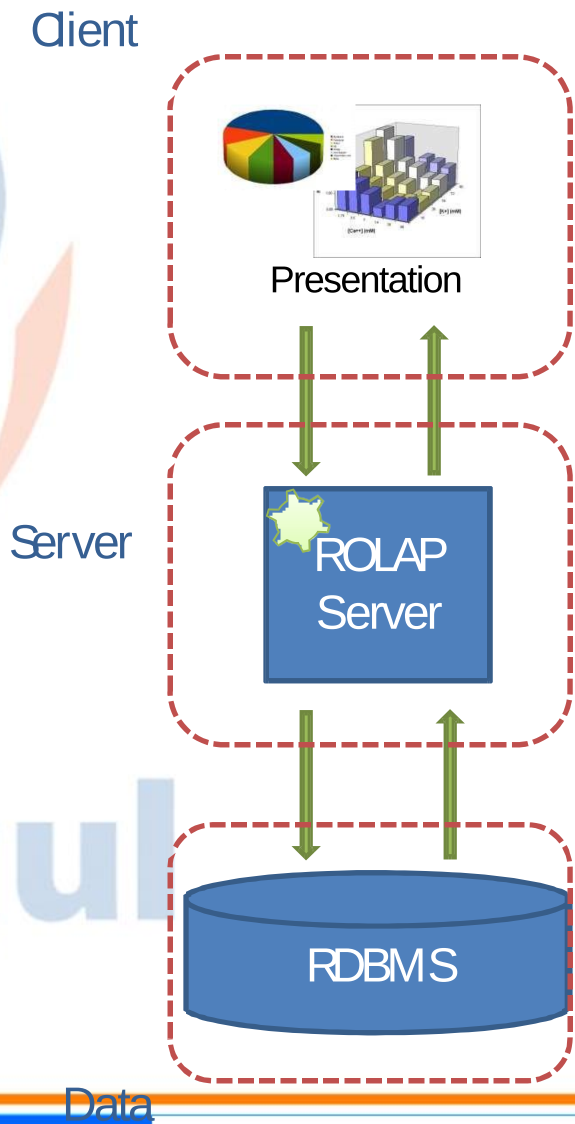


MOLAP

- 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 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), Microsoft Analysis Service, Palo (open source)

ROLAP

- 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**



ROLAP

- 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)

ROLAP vs MOLAP

- 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?**

HOLAP

- **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

DOLAP

- **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

Thank You…