# Advanced Data Management Technologies

## Unit 5 — Logical Design and DW Applications

J. Gamper

Free University of Bozen-Bolzano
Faculty of Computer Science
IDSE

*Acknowledgements: I am indebted to Michael Böhlen and Stefano Rizzi for providing me their slides, upon which these lecture notes are based.*

# Outline

# Outline

# Logical Design

- The **logical design** transforms the conceptual schema for a DM into a logical schema.
  - Choice of the type of logical schema, e.g., snowflake vs. star schema
  - Translation of conceptual schemata
  - Optimization (view materialization, fragmentation)

Conceptual Schema $\Longrightarrow$ Logical Design $\Longrightarrow$ Logical Schema

- Different principles from the one used in operational databases
  - data redundancy
  - denormalization of relations
- Frequently, DM design starts with a logical model.
- The logical model is based on the so-called multidimensional model

# The Multidimensional Model/1

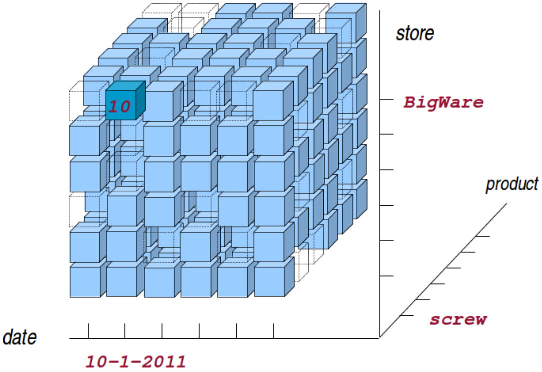
- The Multidimensional Model divides data into facts (with measures) and dimensions
- Facts
  - are the important entity, e.g.,a sale
  - have measures that can be aggregated, e.g., sales price
- Dimensions
  - describe facts
  - e.g., a sale has the dimensions Product, Store and Time

# The Multidimensional Model/2

- Multidimensional model is a logical model with one purpose: data analysis
- Better at that purpose than ERM since it has more built in "meaning":
  - What **is** important
  - What **describes** the important
  - What we want to optimize
  - Automatic aggregations → easy querying
- Less flexible and not suited for OLTP systems.
- Most popular data model for DW.
- Recognized and supported by OLAP/BI tools.
- Goal for dimensional modeling
  - Surround facts with as much context (dimensions and attributes) as possible;
  - Redundancy is ok in well-chosen places.
  - But you should not try to model all relationships (unlike ER/OO modeling!)

# MD Cubes/1

- Facts (data) "live" in a **multidimensional cube**.
- Example: Sales cube with 3 dimensions



# MD Cubes/2

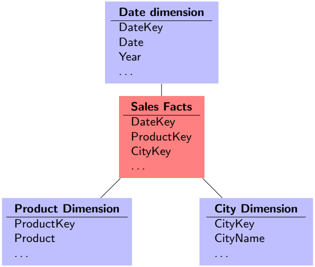
- A cube consists of cells.
  - A given combination of dimension values.
  - A cell can be empty (no data for this combination).
  - A sparse cube has many empty cells.
  - A dense cube has few empty cells.
  - Cubes become sparser for many/large dimensions.
- A cube may have many dimensions.
  - For more than 3 dimensions, the term hypercube is sometimes used.
  - Theoretically, there is no limit for the number of dimensions.
  - Typical cubes have 4-12 dimensions.
- Only 2-3 dimensions can be viewed at a time.
  - Dimensionality reduced by queries via projection/aggregation.

# Outline

# Star Schema

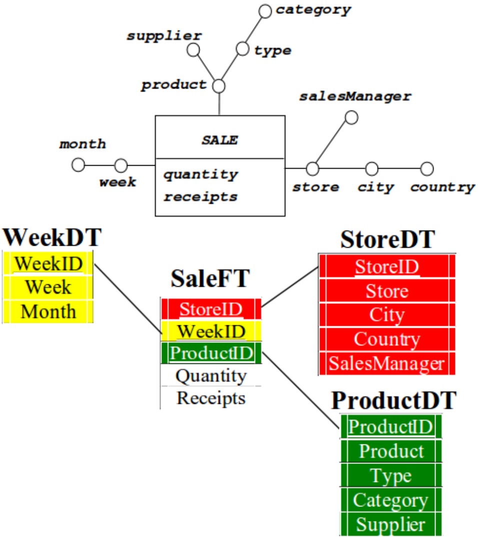
- A common approach to draw a multidimensional model (in relational systems) is the **star schema**, which consists of
  - a set of dimension tables, $DT_1, \ldots, DT_n$, with a primary key $k_i$ and dimensional attributes;
  - a fact table including measures and foreign keys $k_i$ to the dimensional tables.
- As we will see later, a star schema is a relational implementation of the multidimensional model.

**Example:** Star schema for sales facts with 3 dimensions



# Translating Conceptual Schema to Star Schema

- Create a fact table including all measures
- For each dimension, create a dimension table with a primary key and one column for each dimensional attribute
- Besides this simple rule, specific solutions are required for different advanced constructs of the DFM



# Instance of Star Schema

**ProductDT**

| ProductKey | Product | Type | Category |
|---|---|---|---|
| P1 | Bud | Beer | Beverage |
| P2 | Forst | Beer | Beverage |
| P3 | Warst | Beer | Beverage |

**DateDT**

| DateKey | Day | Month | Year |
|---|---|---|---|
| D1 | 25 | May | 2013 |
| D2 | 26 | May | 2013 |

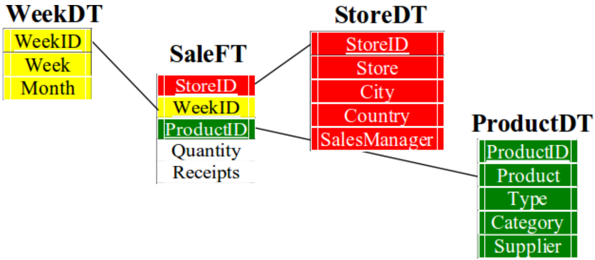
**SalesFT**

| ProductKey | StoreKey | DateKey | Quantity |
|---|---|---|---|
| P1 | S1 | D1 | 575 |

**StoreDT**

| StoreKey | Store | City | Country |
|---|---|---|---|
| S1 | Bilka | Aalborg | Denmark |
| S2 | Spar | Bolzano | Italy |

# OLAP Query on Star Schema




- Query: *Total quantity sold for each product type, week, and city, only for food products.*

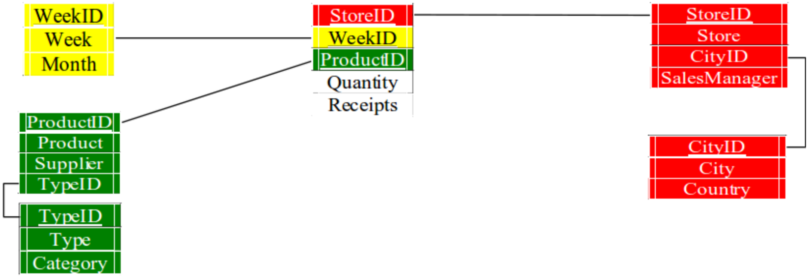
```
SELECT    City, Week, Type, SUM(Quantity)
FROM      WeekDT, StoreDT, ProductDT, SaleFT
WHERE     WeekDT.WeekID = SaleFT.WeekID AND
          StoreDT.StoreID = SaleFT.StoreID AND
          ProductDT.ProductID = SaleFT.ProductID AND
          ProductDT.Category = 'Food'
GROUP BY City, Week, Type;
```

# Star Schema

- PROS
  - Simple and easy overview → ease-of-use
  - Relatively flexible
  - Fact table is normalized
  - Dimension tables often relatively small
  - "Recognized" by many RDBMSes → good performance
- CONS
  - Hierarchies are "hidden" in the columns
  - Dimension tables are de-normalized

# Snowflake Schema

- The star schema can be optimized in terms of space if one or more dimensions are normalized → snowflake schema.



# Instance of Snowflake Schema

**ProductTypeDT**

| TypeKey | Type | CategoryKey |
|---|---|---|
| T1 | Beer | Beverage |
| T2 | Wine | Beverage |

**MonthYearDT**

| MonthKey | Month | Year |
|---|---|---|
| M1 | May | 2013 |

**ProductDT**

| ProductKey | Product | TypeKey |
|---|---|---|
| P1 | Bud | T1 |
| P2 | Forst | T1 |
| P3 | Warst | T1 |

**DateDT**

| DateKey | Day | MonthKey |
|---|---|---|
| D1 | 25 | M1 |

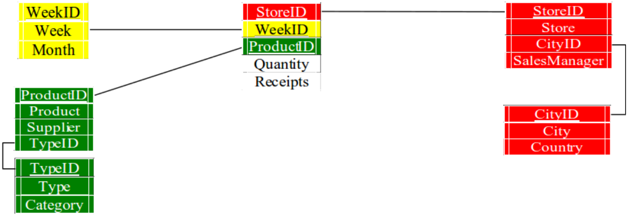
**SalesFT**

| ProductKey | StoreKey | DateKey | Quantity |
|---|---|---|---|
| P1 | S1 | D1 | 575 |

**StoreDT**

| StoreKey | Store | City | Country |
|---|---|---|---|
| S1 | Bilka | Aalborg | Denmark |
| S2 | Spar | Bolzano | Italy |

# OLAP Query on Snowflake Schema




- Query: *Total quantity sold for each product type, week, and city, only for food products.*

```
SELECT    City, Week, Type, SUM(Quantity)
FROM      WeekDT, StoreDT, ProductDT, CityDT, TypeDT, SaleFT
WHERE     WeekDT.WeekID = SaleFT.WeekID AND
          StoreDT.StoreID = SaleFT.StoreID AND
          ProductDT.ProductID = SaleFT.ProductID AND
          StoreDT.CityID = CityDT.CityID AND
          ProductDT.TypeID = TypeDT.TypeID AND
          ProductDT.Category = 'Food'
GROUP BY City, Week, Type;n
```

# Snow-flake Schema

- PROS
  - Hierarchies are made explicit/visible
  - Very flexible
  - Dimension tables use less space
    - However this is a minor saving
    - Disk space of dimensions is typically less than 5 percent of disk for DW
- CONS
  - Harder to use due to many joins
  - Worse performance
    - e.g., efficient bitmap indexes are not applicable

# Redundancy in DW

- Only very little redundancy in fact tables.
  - The same fact data (generally) only stored in one fact table.
- Redundancy is mostly in dimension tables.
  - Star dimension tables have redundant entries for the higher levels.
  - Redundancy problems?
    - Inconsistent data: the central load process helps with this.
    - Update time: the DW is optimized for querying, not updates.
    - Space use: dimension tables typically take up less than 5% of DW.
- So: controlled redundancy is good, up to a certain limit.

# Strengths

- Many-to-one relationship from fact to dimension
- Many-to-one relationships from lower to higher levels in the hierarchies
- Therefore, it is impossible to "count/sum wrong"

# Outline

# Dimensions/1

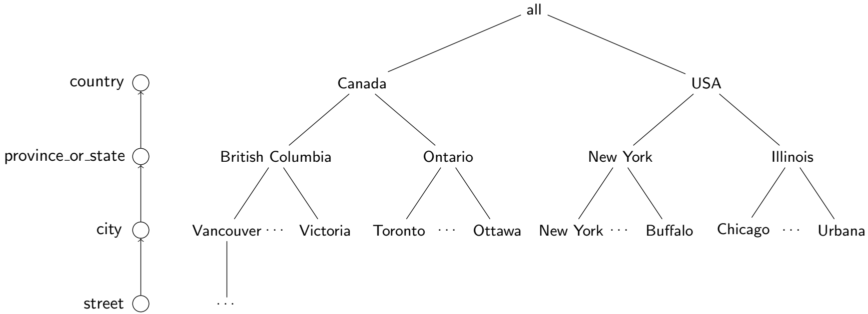
- **Dimensions** are the core of multidimensional databases.
  - Other types of data models do not explicitly support dimensions.
- Dimensions are used for the
  - selection of data;
  - grouping/aggregating data at the right level of detail.
- Dimensions consist of (discrete) dimension values
  - Product dimension has values “milk”, “cream”, . . .
  - Time dimension has values “01/10/2013”, “02/10/2013”, . . .
- Dimension values may have an ordering.
  - Used for comparing cube data across values,
    - e.g., percentage of sales increase compared with last month.
  - Especially used for Time dimension.

# Dimensions/2

- Dimensions encode **hierarchies** with levels.
  - Typically 3-5 levels (of detail).
  - Dimension values are organized in a tree structure or lattice
    - Product: Product → Type → Category
    - Store: Store → Area → City → County
    - Time: Day → Month → Quarter → Year
  - Dimensions have a
    - bottom level: most detailed;
    - top level (ALL): most general.
  - **General rule:** dimensions should contain much information
    - Time dimensions may contain holiday, season, events, . . .
    - Good dimensions have 50-100 or more attributes/levels.

# Concept Hierarchy Example

- A Location dimension with attributes street, city, province_or_state, and country encodes implicitly the following hierarchy.



# Facts

- **Facts** represent the subject of the desired analysis.
  - The "important" in the business that should be analyzed.
- A fact is identified via its dimension values.
  - A fact is a non-empty cell.
  - Some models give facts an explicit identity.
- Generally, a fact should
  - be attached to exactly one value in each dimension;
  - only be attached to dimension values in the bottom levels,
    - e.g., if the lowest time granularity is day, for each fact the exact day should be specified;
    - some models do not require this.

# Different Types of Facts

- **Event** facts (transaction)
  - A fact for every business event, e.g., sale.
  - Event happened for a combination of dimension values and has measures.
- **"Fact-less"** facts
  - A fact per event, e.g., customer contact.
  - Has no numerical measures.
  - Event just happened for a combination of dimension values.
- **(Periodic) Snapshot** facts
  - Captures current status, e.g., inventory, sales of today.
  - A fact for every dimension combination for a given time interval.
- **Cumulative snapshot** facts
  - Captures cumulative status of a process up to now, e.g, sales order.
  - Typically several date stamps, which are updated as the process is completed, e.g., order date, shipping date, paying date.

- Every type of facts answers different questions.
- Event facts and snapshot facts are most frequent.

# Granularity

- **Granularity** of facts is important.
  - What does a single fact mean?
  - Determines the level of detail.
  - Given by the combination of bottom dimension levels
    - e.g., total sales per store per day per product.
- Has an impact on the number of facts, hence the scalability!
- Often the granularity is a single business transaction, e.g., sale.
- Sometimes the data is aggregated, e.g., total sales per store per day per product.
  - Aggregation might be necessary for scalability.
- Generally, transaction detail can be handled
  - Except perhaps huge clickstreams, etc.

# Measures

- **Measures** represent the fact property that users want to study and analyze,
  - e.g., total sales or average sales per day.
- A measure has two components
  - **Numerical value:** used to describe a fact/event, e.g., sales price, # of items in a transaction.
  - **Aggregation formula:** used for aggregating/combining a number of measure values into one, e.g., SUM, AVG, MAX.
- Single fact table rows/measures are (almost) never retrieved, but aggregations over millions of fact rows.

# Additivity of Measures

- A measure is called **additive along a dimension** if the SUM operator can be used to aggregate it along that dimension (hierarchy); otherwise it is **non-additive** along that dimension.
- Additivity is an important property of measures
  - Provides flexibility in aggregation and navigation.
  - Most frequently the case.
- Classification of measures based on additivity.

# Different Types of Measures/1

- **Additive** measures (flow measures):
  - Additive along all dimensions
  - Refer to a timeframe, at the end of which they are evaluated cumulatively
  - Typically the case for event facts,
    - e.g., the number of products sold in a day, monthly receipts, yearly number of births, gross profit per year, cost, etc.
- **Semi-additive** measures (level measures)
  - Additive only over some dimensions (typically non-temporal dimensions)
  - Are evaluated at particular times and often occur in snapshot facts
    - e.g., the number of products in inventory: non-additive across time
    - customer_count: additive across store, non-additive across product
    - the number of inhabitants in a city
- **Non-additive** measures (unit measures)
  - Additive over none of the dimensions
  - Are evaluated at particular times but are expressed in relative terms
    - e.g., product unit price, discount percentage, currency exchange: SUM makes no sense along any dimension, but AVG, MIN, MAX.

# Different Types of Measures/2

| Measures | Temporal hierarchies | Nontemporal hierarchies |
|---|---|---|
| Additive (Flow) | **SUM**, AVG, MIN, MAX | **SUM**, AVG, MIN, MAX |
| Semi-additive (Level) | AVG, MIN, MAX | **SUM**, AVG, MIN, MAX |
| Non-additive (Unit) | AVG, MIN, MAX | AVG, MIN, MAX |

# Non-aggregable Measures

- A measure is called **non-aggregable along a dimension** if it cannot be aggregated along that dimension using any aggregation operator.

- **Example:** A measure `numberOfCustomers` with dimensions `product`, `store`, and `day` that is estimated from the number of receipts.
- Non-aggregable along product dimension, since a receipt is likely to contain several products.
  - many-to-many relationship between receipts and products (instead of many-to-one)
- Can be aggregated over store and date dimension.

# Outline

1. Multidimensional Model
2. Star and Snowflake Schema
3. Facts, Dimensions, and Measures
4. **DW Applications**
5. DW Implementation

# Reporting

- **Reporting** is for users who need a regular access to information in an almost static way.
  - e.g., local health authorities must send monthly reports to state offices.
- Report is defined by a query (multiple queries) and a layout (diagrams, histograms, etc.).

Cdc 8090

| VOLUMI, MIX E QUALITA' PRODUZIONE | FINO AL | 2008 | BUDGET PIANO | PESO | Limite inferiore | Limite superiore | 2009 PROIEZ. | SCOST. | MATURATO | BUDGET 2010 BUDGET | PESO | Limite inferiore | Limite superiore |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dimessi ordinari | 31.08. | 293 | 296 | 0% | 0 | 0 | 270 | -8,8% | 0% | 270 | | | |
| Trasferimento | 31.08. | 1 | 2 | | | | 0 | -100,0% | 0% | 0 | | | |
| gg di degenza | 31.08. | 990 | 939 | | | | 893 | 5,8% | 0% | 893 | | | |
| n. posti letto | 31.08. | 5 | 5 | | | | 5 | | 0% | [redacted] | | | |
| Accessi day hospital/surgery | 31.08. | 887 | 981 | 0% | 0 | 0 | 1.168 | 19,0% ! | 0% | 1.168 | | | |
| n. posti letto day hosp./surg. | 31.08. | 4 | 6 | | | | 6 | | 0% | 6 | | | |
| Totale attività per esterni | 31.08. | 45.670 | 46.986 | 0% | 0 | 0 | 44.039 | -6,3% | 0% | 44.039 | | | |
| Totale attività per interni | 30.09. | 566 | 548 | 0% | 0 | 0 | 559 | 1,9% | 0% | 559 | | | |
| Totale attività ricevuta | 30.06. | 2.561 | 0 | 0% | 0 | 0 | 2.502 | | 0% | 2.502 | | | |
| - di cui di laboratorio | 31.08. | 2.295 | 2.202 | 0% | 0 | 0 | 2.340 | 6,3% | 0% | 2.340 | | | |
| - di cui di radiologie | 31.08. | 185 | 171 | 0% | 0 | 0 | 102 | -40,4% | 0% | 102 | | | |
| n° prest. di lab. x dimessi ordinari | 31.08. | 7,83 | | | | | 6,67 | | 0% | 6,67 | 40% | | 6,67 |
| n° prest. di rad. x dimessi ordinari | 31.08. | 0,63 | | | | | 0,38 | | 0% | 0,38 | 30% | | 0,38 |
| COSTI ED EFFICIENZA | | | | | | | | | | | | | |
| Consumi beni sanitari | 30.09. | 486.304 | 411.792 | 50% | 0 | 432.382 | 501.299 | 21,7% ! | [redacted] | 501.299 | | | |
| PHT + H-CSP2 | 30.09. | | | | | | 0 | | 0% | 0 | | | |
| Consumi beni non sanitari | 30.09. | 3.648 | 3.728 | 0% | 0 | 0 | 3.264 | -11,9% | 0% | 3.284 | | | |
| altri costi | 30.09. | 4.459 | 3.831 | 0% | 0 | 0 | 6.524 | 70,3% ! | 0% | 6.524 | | | |
| Totale consumi | | 494.411 | 419.351 | 0% | 0 | 0 | 511.107 | 21,9% ! | 0% | 511.107 | | | |
| costi personale (non da pianificare) | 30.06. | 994.834 | | | | | 1.029.189 | | | | | | |
| unità personale | 30.09. | 5,75 | 7,00 | 0% | 0.00 | 0.00 | 6,26 | -10,6% | 0% | 6,44 | | | |
| presenza media | 31.08. | 5,16 | | | | | 6,44 | | | | | | |
| Tasso utilizzo letti | 31.08. | 54,10% | 51,31% | 0% | 0,00% | 0,00% | 54,20% | | 0% | 54,20% | | | |
| degenza media | 31.08. | 3,33 | 3,16 | 0% | 0.00 | 0.00 | 3,60 | | 0% | 3,60 | | | |
| tasso op | 30.06. | 70,76%. | 87,23% | 0% | 0,00% | 0,00% | 70,73% | | 0% | 70,73% | | | |
| % ricoveri di 1 giorno | 31.08. | 7,64% | 2,00% | 50% | 0,00% | 3,00% | 6,40% | | [redacted] | 6,40% | 30% | | 6,40% |
| peso medio drg | 30.06. | 0,68 | 0,81 | | | | 0,89 | | 0% | 0,89 | | | |
| mobilità provinciale passiva | 30.06. | 151.396 | 0 | 0% | 0 | 0 | 133.249 | | 0% | | | | |
| mobilità provinciale attiva | 30.06. | 680.010 | 0 | | | | 954.940 | | 0% | | | | |
| mobilità Innsbruck | 30.06. | 30.775 | 0 | | | | 24.448 | | 0% | | | | |
| servizio trasporti: n° pazienti | 31.08. | 32 | 0 | 0% | 0 | 0 | 21 | | 0% | | | | |
| servizio trasporti: € | 31.05. | 1.566 | 0 | | | | 999 | | 0% | | | | |
| Summe Gewichtung | | | | 100% | | | | | 0% | | | | |

# OLAP/1

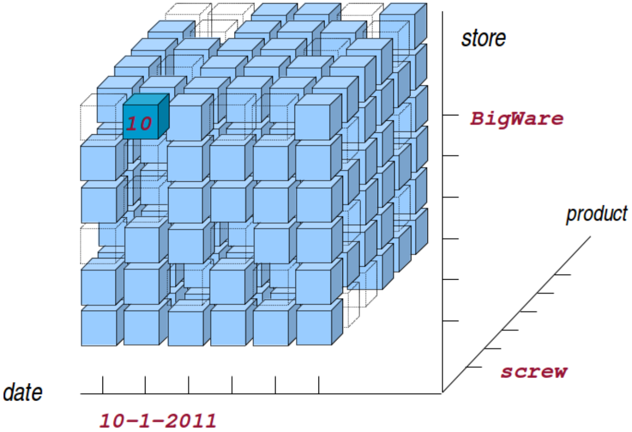
- **OLAP (Online Analytical Processing)** is the most popular way to exploit information in a DW.
- Provides more flexibility, especially when the analysis needs are not defined beforehand.
- Interactive way to explore data on the basis of the multidimensional model.
  - Each step is the result of the outcome of preceding steps.
- Each step of an analysis session applies an OLAP operator.
- OLAP tools typically use tables with multiple headers and colors to visualize multidimensional query results.

# OLAP/2

- Two kinds of OLAP operators/queries:
  - Aggregation operators summarize fact data, e.g., with SUM.
  - Navigation operators allow to examine data from different viewpoints and detail levels.
- Analysis starts at some level, e.g. (Quarter, Product).
  - Roll Up: less detail, e.g., Quarter → Year
  - Drill Down: more detail, e.g., Quarter → Month
  - Slice/Dice: selection, e.g., Year=1999
  - Drill Across: "join" on common dimensions

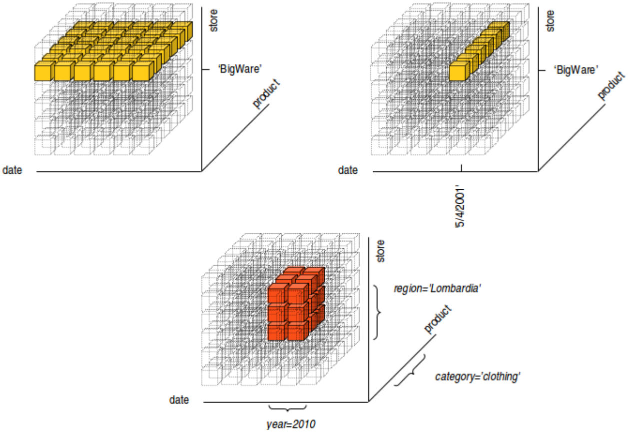
# OLAP Example/1

- Sales Cube



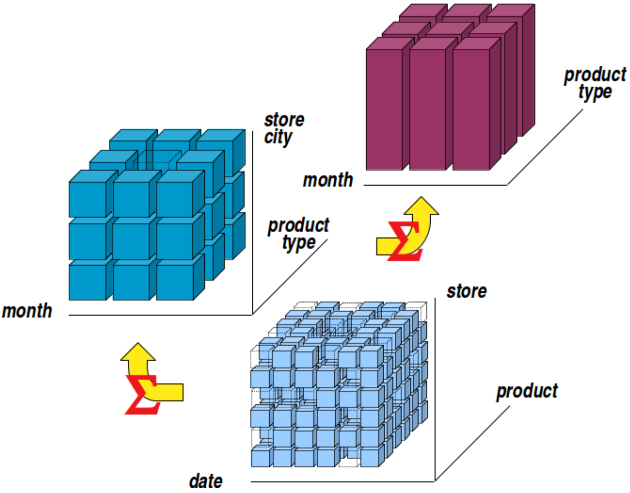
# OLAP Example/2

- Slicing and Dicing: select specific (ranges of) values for dimension attributes



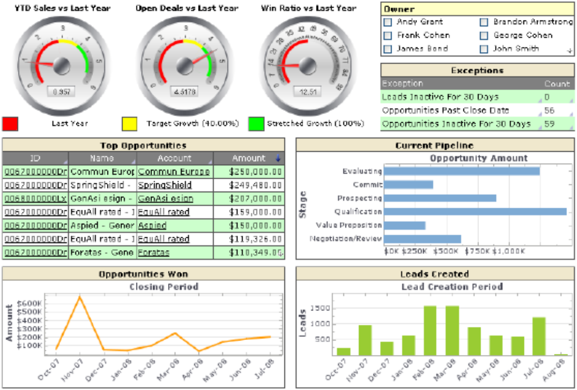
# OLAP Example/3

- Aggregation



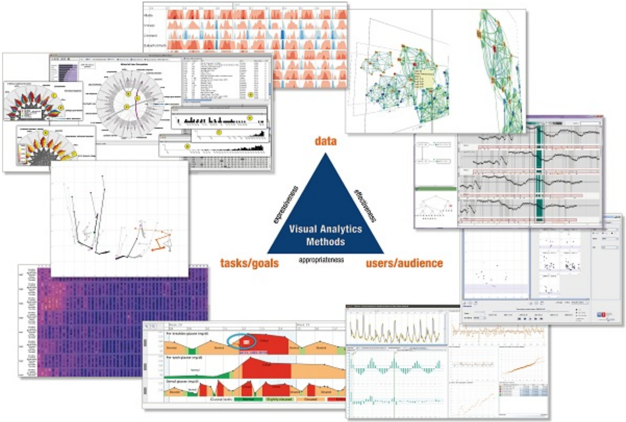
# Dashboards

- **Dashboards** display a limited amount of information in a easy-to-read graphical format.
- Frequently used by senior managers who need a quick overview of the most significant changes,
  - e.g., real-time overview of trends.
- Not useful for complex and detailed analysis.



# Visual Analytics

- **Visual Analytics** is about analytical reasoning supported by interactive visual interfaces.
- Graphical presentation of complex result.
- Color, size, and form help to give a better overview.



# Data Mining

- **Data mining** is automatic knowledge discovery.
- Has its roots in AI and statistics.
- Different tasks:
  - Classification
    - Partition data into pre-defined classes.
  - Clustering
    - Partition data into groups such that the similarity within individual groups ist greatest and the similarity between groups is smallest.
  - Affinity grouping/associations
    - Find associations/dependencies between data.
    - Rules: $A \rightarrow B(c\%, s\%)$: if $A$ occurs, $B$ occurs with confidence $c$ and support $s$.
  - Prediction
    - Predict/estimate unknown value based on similar cases.
- Important to choose the granularity for mining.
  - Too small granularity gives no good results (shirt brand, ...).

# Outline

# Relational OLAP (ROLAP)

- Data/Cube is stored in relational tables.
  - Fact table stores facts.
    - One column for each measure and dimension.
  - Dimension tables store dimensions.
  - SQL is used for querying.
- PROS
  - Leverages investments in relational technology.
  - Huge amount of literature and broad experience with RDBMSs.
  - Scalable to billions of facts.
  - Flexible design and easier to change.
  - New techniques adapted from MOLAP.
    - Indexes (e.g., bitmap), materialized views, special handling of star schemas.
- CONS
  - Storage use often 3-4 times higher than in MOLAP.
  - Higher response times due to joins.

# Relational OLAP (ROLAP) Schemas

- One completely de-normalized table
  - Bad: inflexibility, storage use, bad performance, slow update.
- Star schema
  - One fact table
  - De-normalized dimension tables
  - One column per level/attribute
- Snowflake schema
  - Dimensions are normalized
  - One dimension table per level
  - Each dimension table has integer key, level name, and one column per attribute

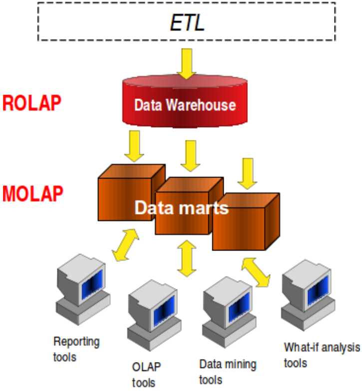
# Multidimensional OLAP (MOLAP)

- Data/cube is stored in special multidimensional data structures.
  - Arrays with positional access.
- PROS
  - Less storage use ("foreign keys" are not stored).
  - Multidimensional operations can be performed without complex and costly joins.
  - Faster query response times.
- CONS
  - Up till now not so good scalability.
  - Less flexible, e.g., cube must be re-computed when design changes.
  - Does not reuse an existing investment (but often bundled with RDBMS).
  - "New technology", not an open technology.
  - No standards yet available, very specific optimizations are used.

# Hybrid OLAP (HOLAP)

- ROLAP and MOLAP elements are combined into a single architecture.
  - Aggregates stored in multidimensional structures (MOLAP)
  - Detail data stored in relational tables (ROLAP)

- PROS
  - Scalable and fast.
  - Largest amount of data and sparse subcubes are stored in RDBMS.
  - Dense subcubes of aggregated data (DMs) are stored in multidimensional structures.
    - Most frequently needed by the user.
- CONS
  - Complexity



# Summary

- Logical design transforms the conceptual model into a logical model
- Multidimensional model is de facto standard logical model.
  - Consists of dimensions, facts, and measures
  - Facts are the important entities, dimensions describe the important entities/facts.
  - Data lives in multidimensional cubes.
- In relational systems, the multidimensional model is materialized as star or snowflake schema: 1 fact table and several dimension tables.
- Different fact types:
  - event facts, fact-less facts, snapshot facts, cumulative snapshot facts.
- Additivity is an important property of measures.
  - Additive measures, semi-additive measures, non-additive measures.
- Different DW applications: Reporting, OLAP, dashboards, visual analytics, and data mining.
- Different DW implementations
  - ROLAP
  - MOLAP
  - HOLAP