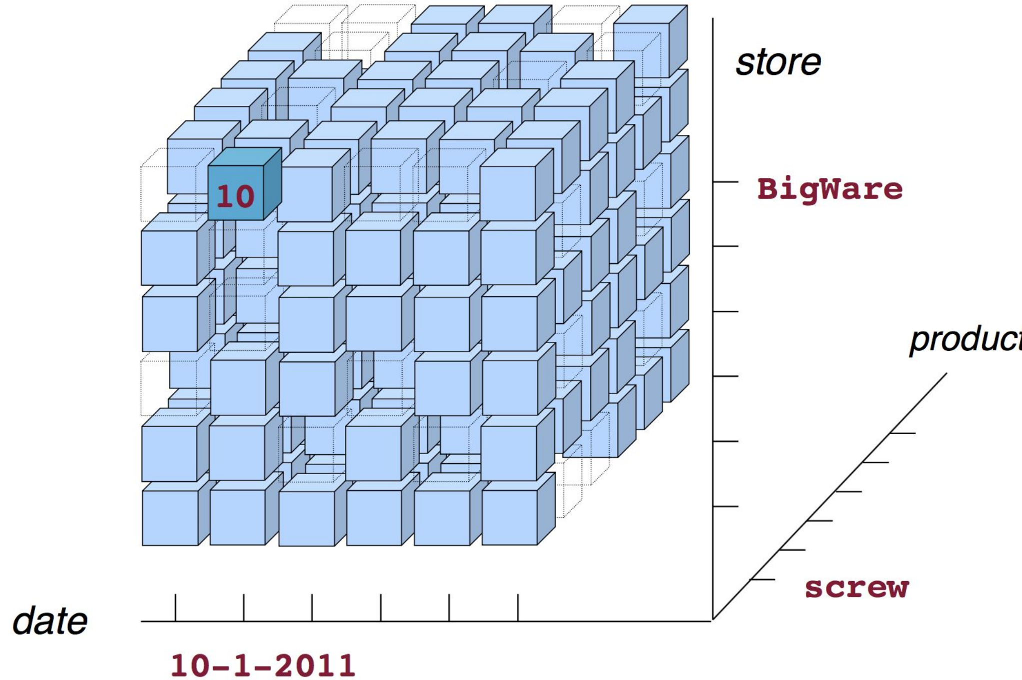
Business Intelligence for the Data Scientist

Giacomo Bergami



Querying a DataWarehouse: from theory…

I. A quick review. Using DFM as a Conceptual Design Language

Conceptual Design

- **Conceptual Models** enhance the communication between IT developers and clients (the final users): no IT knowledge is required to understand such visual language. It schematizes the Multidimensional view over the data.
  - An instance of a conceptual model is called **conceptual schema**, that is a concise description of the users' requirements.
- The **data cube** is the visual representation of an instance of the **conceptual schema**.
- Helps the definition of the Logical Data Warehouse level.

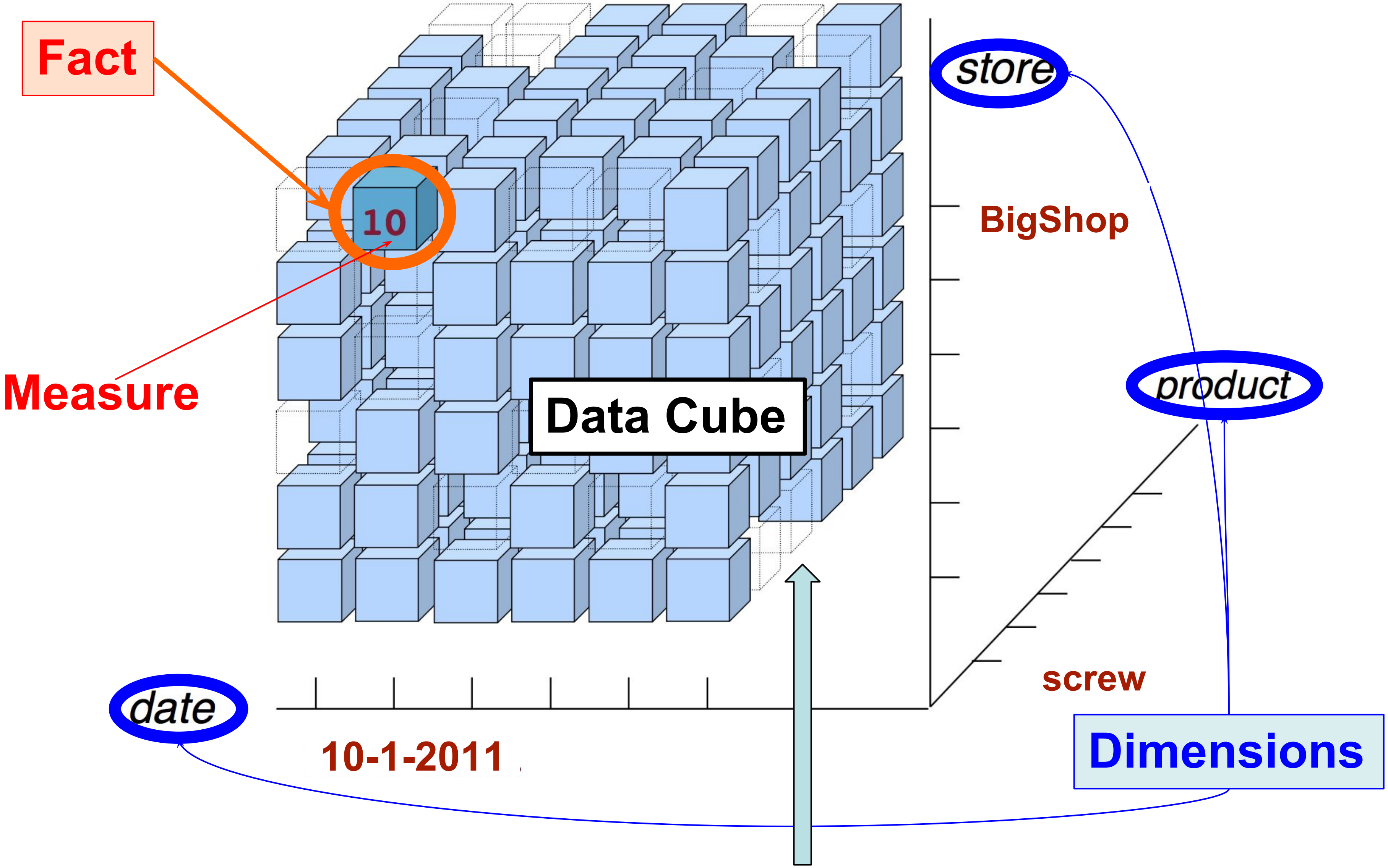
Data Cube: a Multidimensional View

Facts, Measures and Dimensions

- A **fact** is a concept relevant to decision-making processes. It typically models a set of events taking place within a company (e.g., sales, shipments, purchases). It is essential that a fact has dynamic properties or evolves in some way over time
- A **measure** is a numerical property of a fact and describes a quantitative fact aspect that is relevant to analysis (e.g., every sale is quantified by its receipts)
- A **dimension** is a fact property with a finite domain and describes an analysis coordinate of the fact. Typical dimensions for the sales fact are products, stores, and dates, and could be arranged in hierarchies.

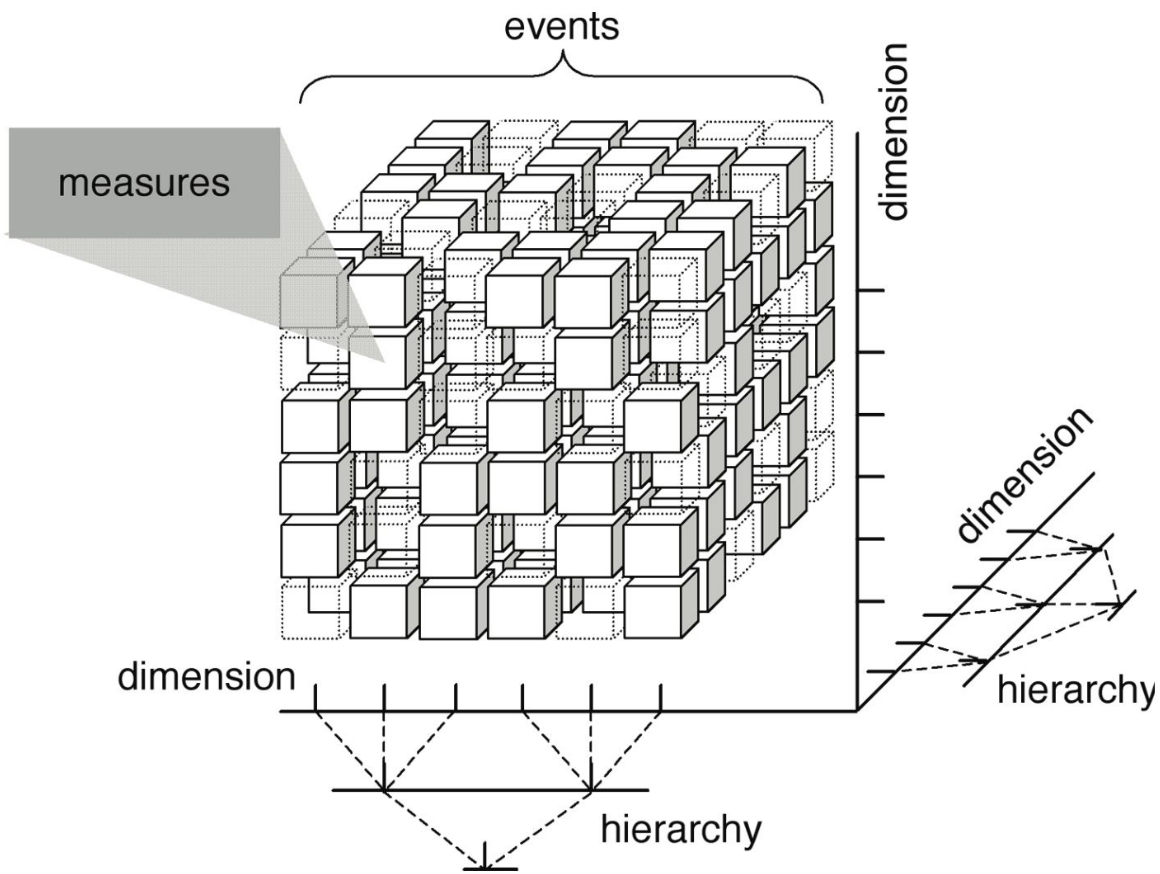
Data Cube: a Multidimensional View

Facts, Measures and Dimensions



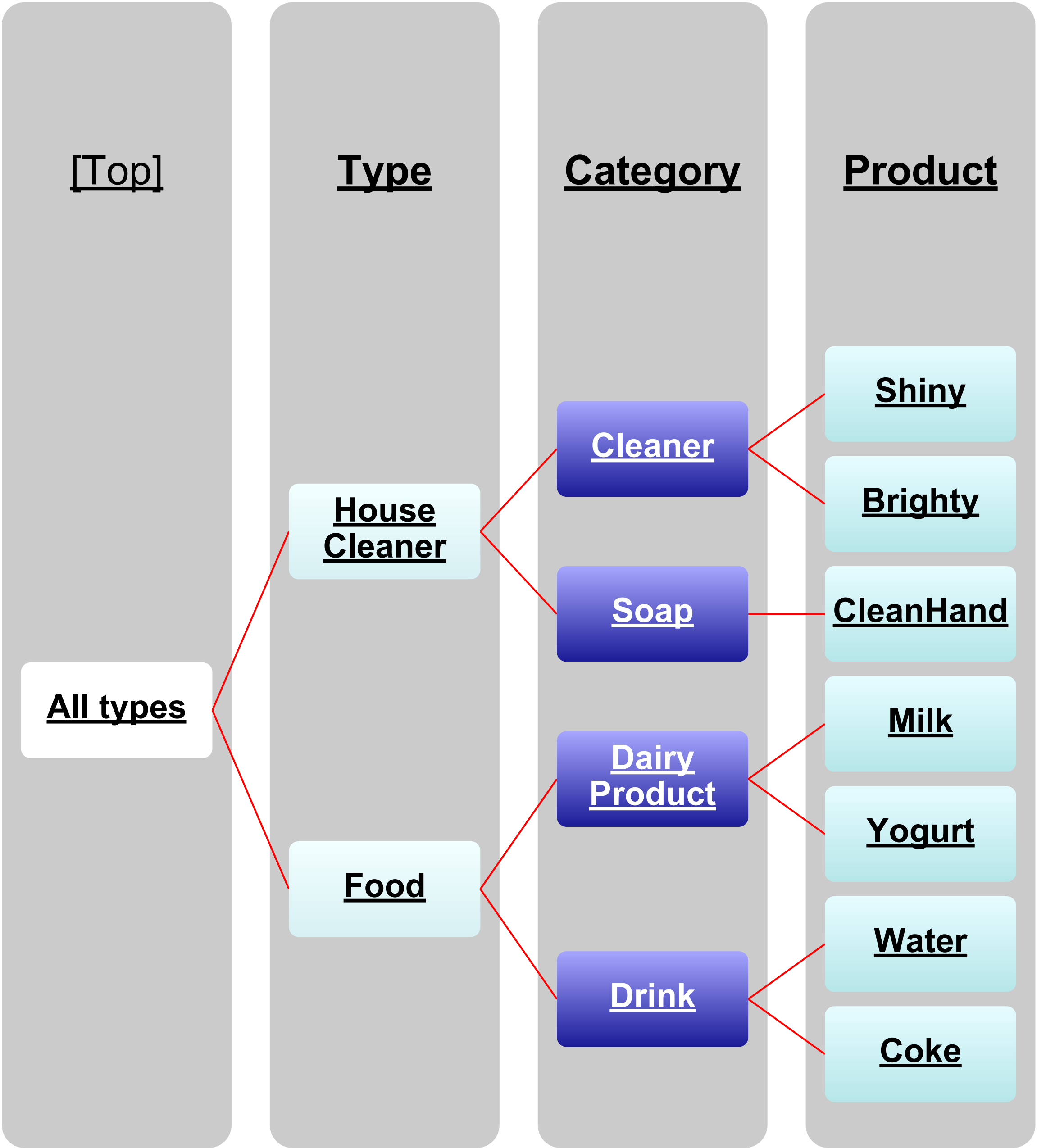
Please note that the data cube could be sparse

Dimensions and Hierarchies

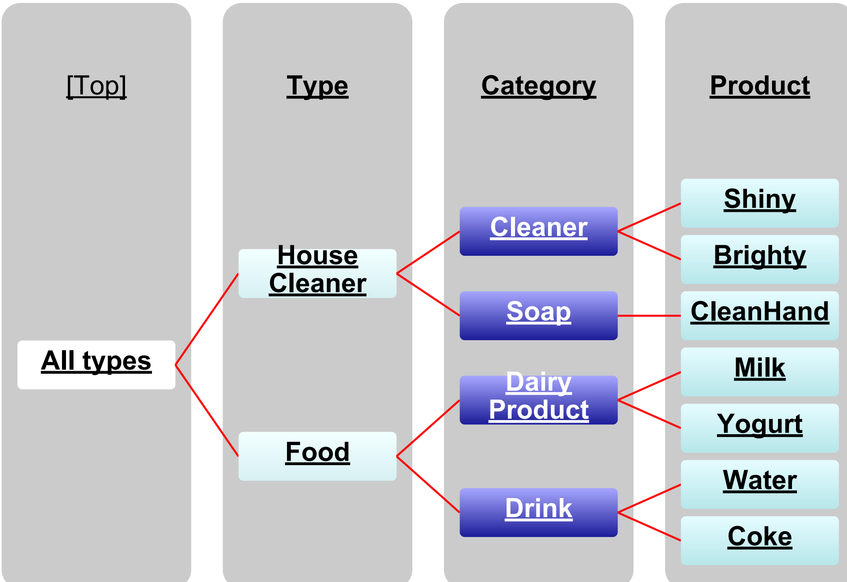


Alongside to each dimension we could build a hierarchy of dimensions: this could be useful to predict the value resulting from the aggregation along a specific dimension. All the measures will be aggregated accordingly to the choosen aggregation function.

Hierarchy: *Example*



Relational Hierarchies



Relational Implementation of the Product Hierarcy

ID	Product	Type	Category
0	Shiny	Cleaner	House Cleaner
1	Brighty	Cleaner	House Cleaner
2	CleanHand	Soap	House Cleaner
3	Milk	Dairy Product	Food
4	Yogurt	Dairy Product	Food
5	Water	Drink	Food
6	Coke	Drink	Food

A Conceptual Design Language: **DFM** (1/2)

- The **DFM** (Dimensional Fact Model) is a graphical conceptual model for both data warehouses and data marts.
  - provides effective support to conceptual design
  - makes communication possible between designers and final users with the goal of formalizing requirement specifications
  - enables early testing (i.e., before cubes are actually implemented)
  - builds a stable platform for logical design (independently of the target logical model)
  - provides clear and expressive design documentation

A Conceptual Design Language: **DFM** (2/2)

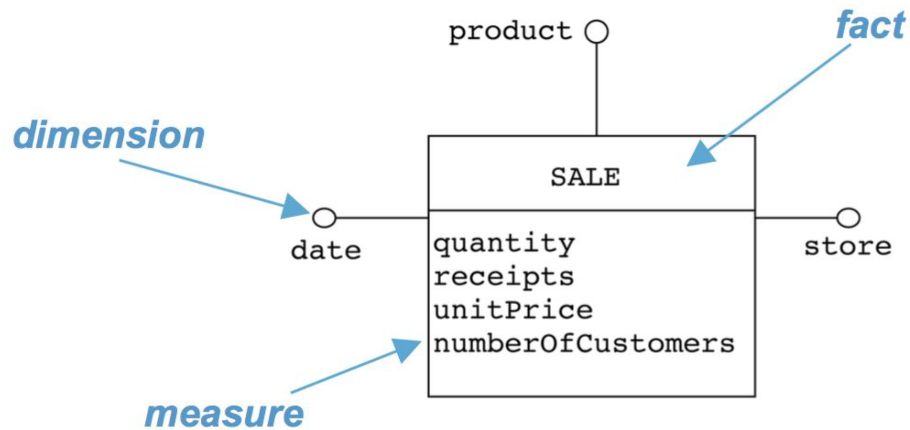
- The conceptual representation generated by the DFM consists of a set of fact schemata that basically model facts, measures, dimensions, and hierarchies
- The model is simpler than the best known **MultiDim** (see the text book), since information is condensed to the data cube elements.

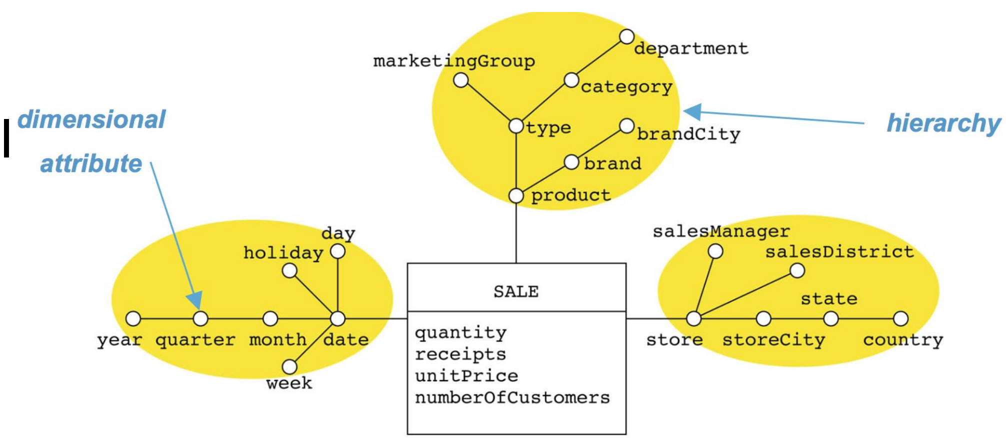
(Dimensional) Hierarchies

- **Hierarchies** describe the different possible abstraction levels of the facts' dimensions. We assume for simplicity's sake that our hierarchies are *taxonomies* (i.e. directed trees).
  - They are used to (dis-)aggregate facts.
  - Note that in the most general assumpion, a hierarchy could become an *ontology*.
- Each dimension on the DFM model can be the root of a hierarchy of so-called **attributes**. The arcs model *many-to-one associations* between the parent node and the children.

Using **DFM** for represent the Data Cube Schema

- Intuitively, each fact is described as a UML class, where the attributes are the measures of the fact itself.
- Each dimension is described as a part of a coordinate describing the fact. No details about the dimension's definiton are provided for a single fact view.
- The model could be extended in order to express a hierarchy level over the dimensions over which perform the roll ups and the drill downs.



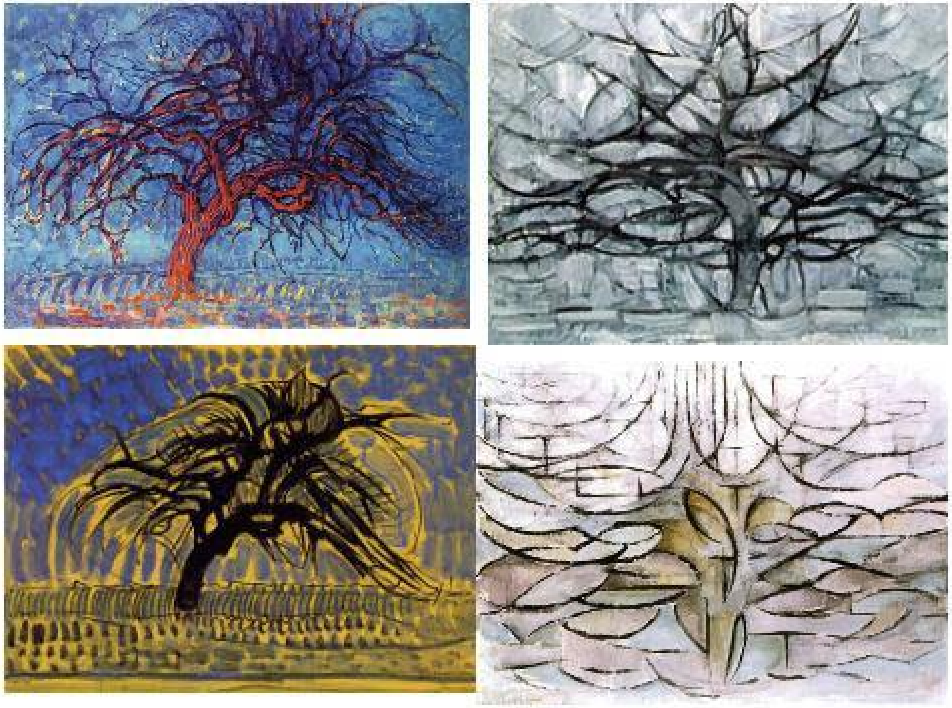


References (Ia)

- M. Golfarelli, S. Rizzi: "**Data Warehouse Design: Modern Principles and Methodologies**". McGraw-Hill, 2009.
- S. Rizzi: "**Conceptual Modeling Solutions for the Data Warehouse**". Idea Group Publishing, 2007.
  - https://fenix.tecnico.ulisboa.pt/downloadFile/3779571785339/DFM.pdf
- Jensen, Pedersen et al.: "**Multidimensional Databases and Data Warehousing**". Morgan and Claypool Publishers, 2010. (*Chapters 1&2*)

References (Ib)

- Further readings**:**
  - Martin Staudt et al. “**The Role of Metadata for Data Warehousing**”. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.39.7518
  - A. Vaisman, E. Zimányi: “Data Warehouse Systems: Design and Implementations”. Springer Verlag, 2014. (*Chapters 2,3,6*)



Piet Mondrian and his “bomen”

Creating the datawarehouse cube schema (ROLAP)

II. Schema Workbench

Mondrian Schema (Pentaho)

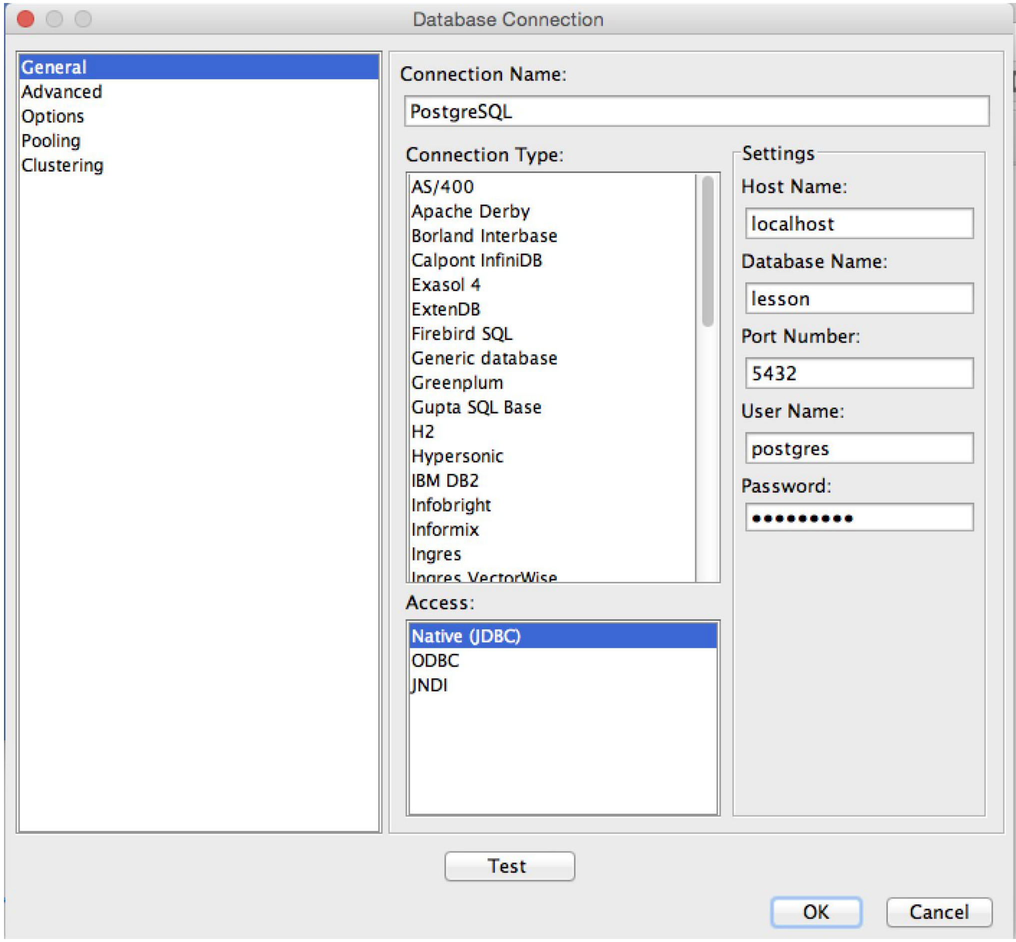
- We need a way to map the relational representation of our relational database into a multidimensional view (*ROLAP*).
- A **Mondrian Schema** defines a multi-dimensional database. It contains a logical model, consisting of cubes, hierarchies, and a mapping of this modle onto a physical model.
  - The hierarchies do not include dimensions' tree hierarchies, but only hierarchy on a line.
  - The *Mondrian Schema* is expressed as a XML file, that could be edited using the tool **Schema Workbench** (http://sourceforge.net/projects/mondrian/files/schema%20workbench/)
  - Could handle both *star* and *snowflake* schemas.

Connecting "Schema Workbench"

- Create the relational database:

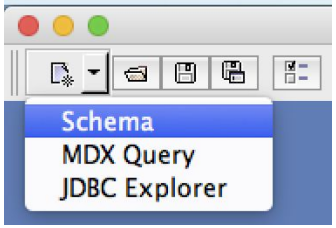
```
create database lesson
with owner = postgres
```

- Create a ProductHierarchy relation containing all the elements of the previous table
- Initialize "Schema Work" with the default settings (Option > Connection…)



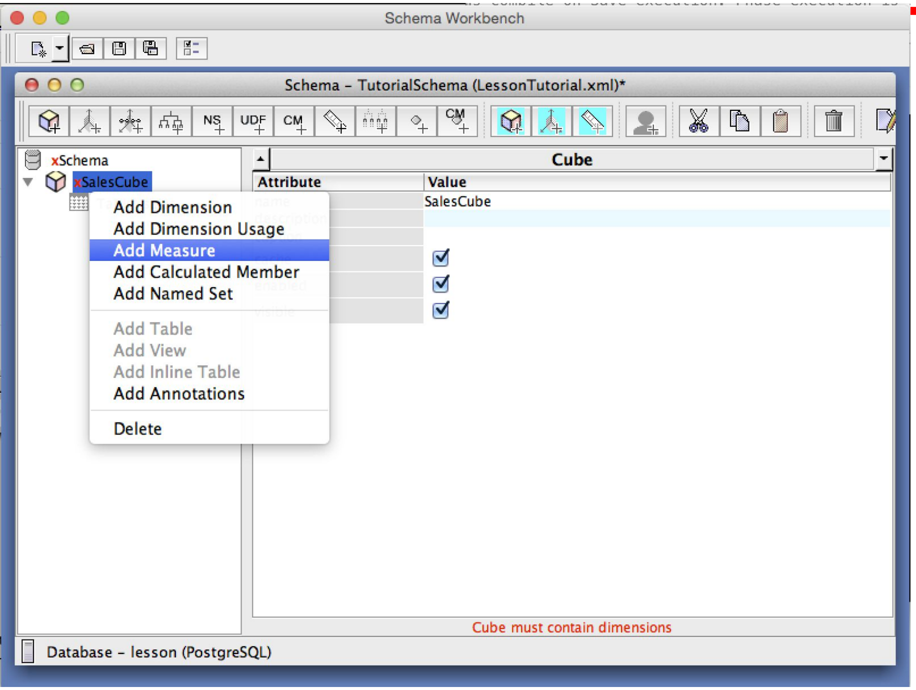
Creating the DataCube & Measures (1/2)

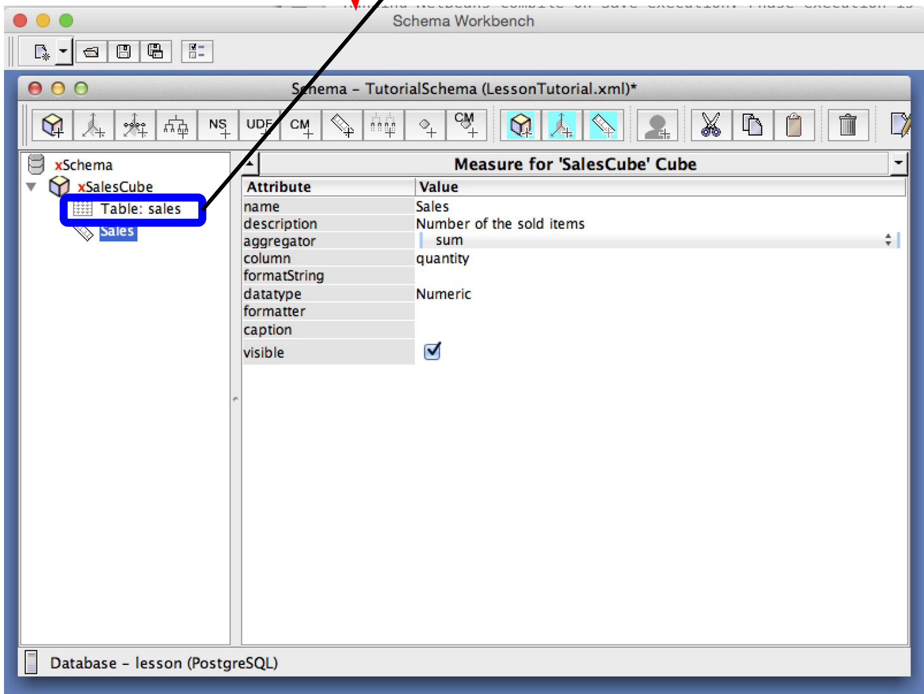
- Create a new Schema file
- Create the DataCube Sales and add a measure as the number of the sold items.



Adding the measure

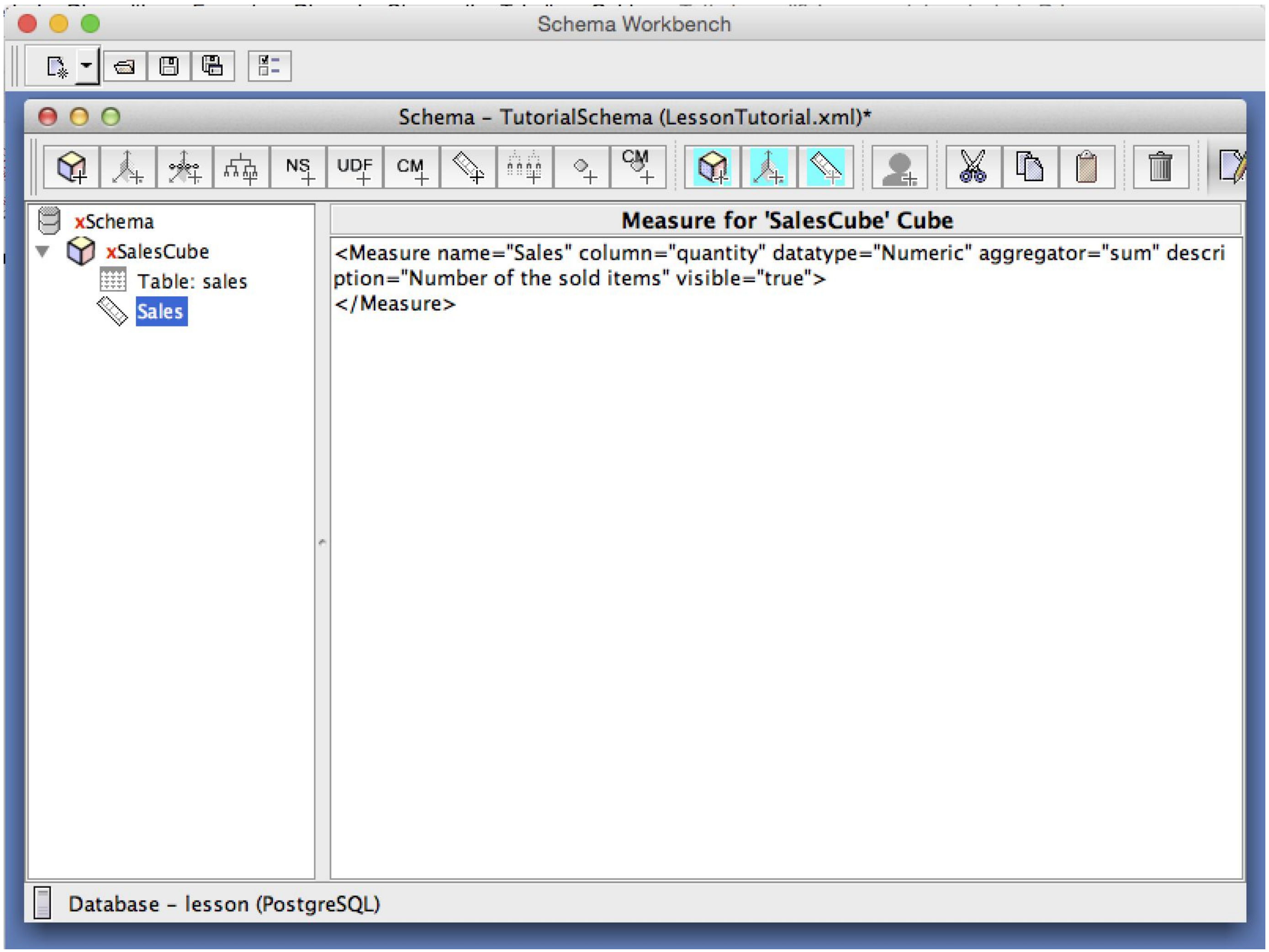
The cube's table refers to the **fact table**





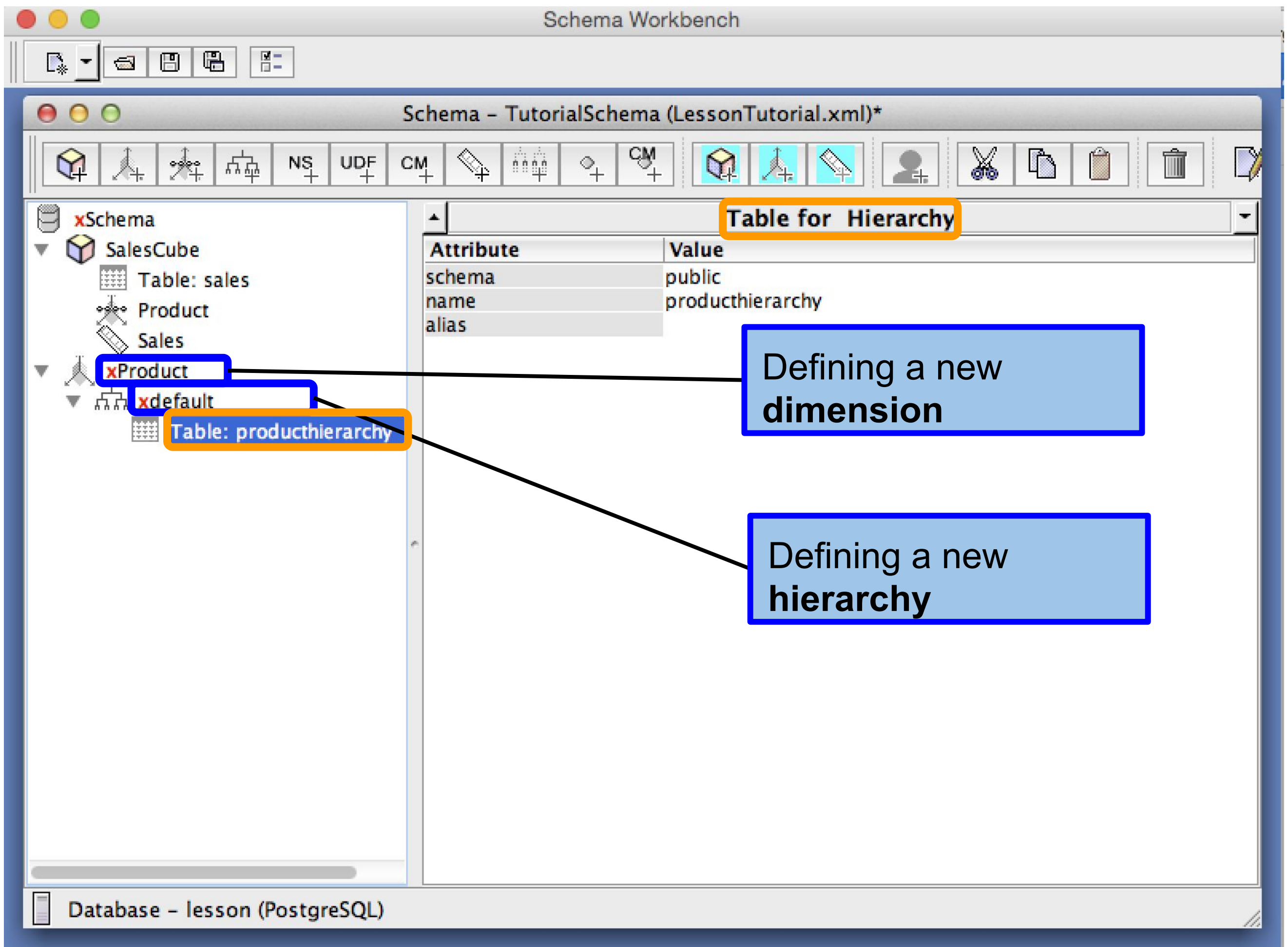
Measures (2/2)

- The description of the Measure in Mondrian Schema is the following one (View > View XML):



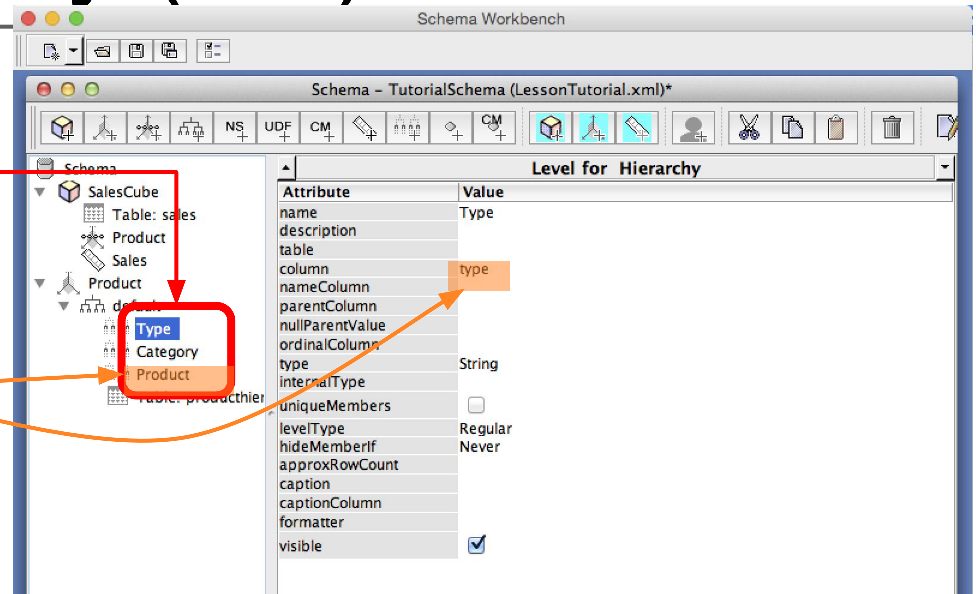
Hierarchy (1/3)

- The description of the Measure in Mondrian requires to point out the dimension table **ProductHierarchy**

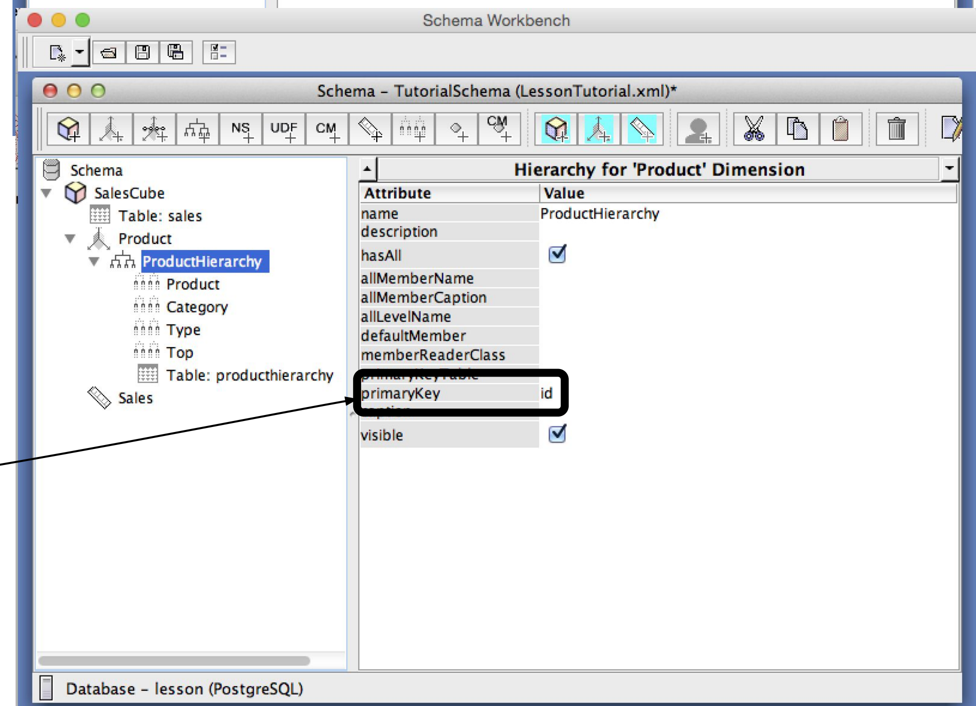


Hierarchy (2/3)

- Defining the Hierarchy as as set of levels.
- Each level refers to a specific column of the description table. The first level should be the top level of the hierarchy

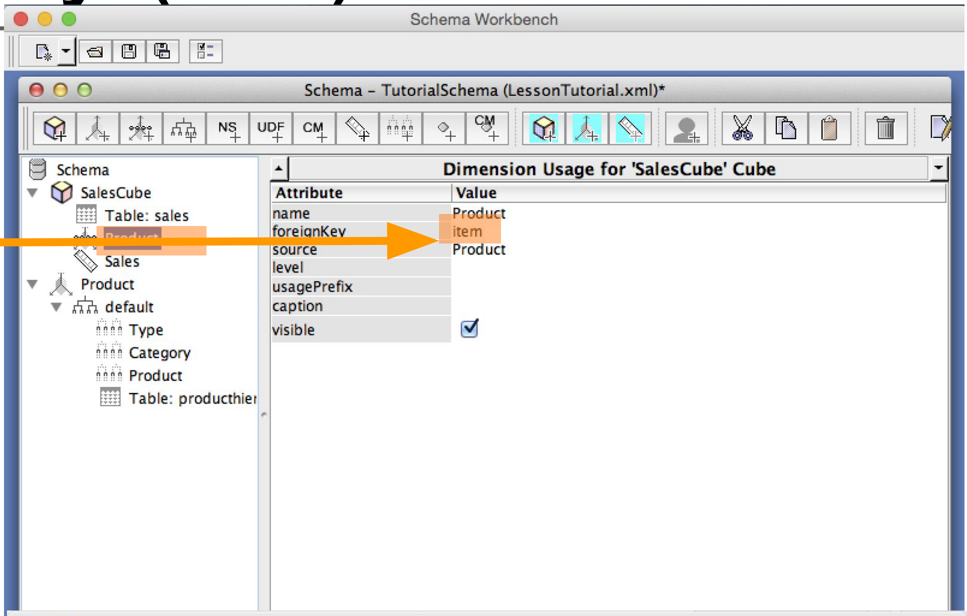


- In the detailment of the products' hierarchy, specify which is the **primary key** of the table

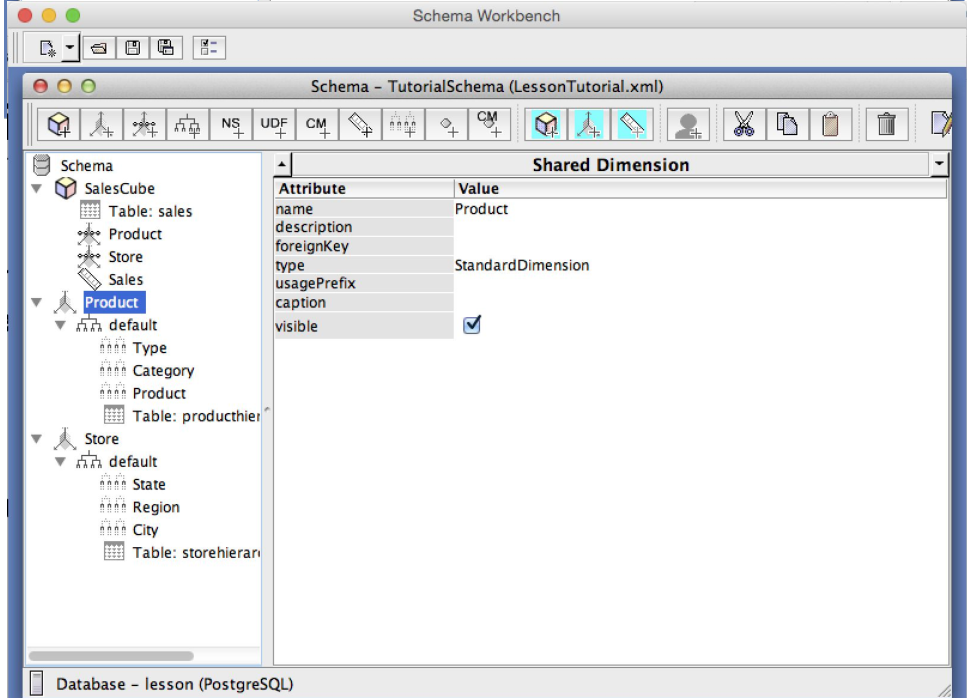


Hierarchy (3/3)

- For each dimension, specify which is the corresponding foreign key in the fact table.



- Do the same thing for the **Stores** dimension



Modifying the generated XML (1/2)

- Open the generated XML file with a text editor. Remove the circled elements.

```xml
<Schema name="TutorialSchema">
  <Dimension type="StandardDimension" visible="true" highCardinality="false" name="Product">
    <Hierarchy visible="true" hasAll="true" primaryKey="id">
      <Table name="producthierarchy" schema="public">
      </Table>
      <Level name="Type" visible="true" column="type" type="String" uniqueMembers="false" levelType="Regular" hideMemberIf="Never">
      </Level>
      <Level name="Category" visible="true" column="category" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
      </Level>
      <Level name="Product" visible="true" column="product" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
      </Level>
    </Hierarchy>
  </Dimension>
  <Dimension type="StandardDimension" visible="true" highCardinality="false" name="Store">
    <Hierarchy visible="true" hasAll="true" primaryKey="id">
      <Table name="storehierarchy" schema="public">
      </Table>
      <Level name="State" visible="true" column="country" type="String" uniqueMembers="false" levelType="Regular" hideMemberIf="Never">
      </Level>
      <Level name="Region" visible="true" column="region" type="String" uniqueMembers="true" levelType="Regular" hideMemberIf="Never">
      </Level>
      <Level name="City" visible="true" column="city" type="String" uniqueMembers="false" levelType="Regular" hideMemberIf="Never">
      </Level>
    </Hierarchy>
  </Dimension>
  <Cube name="SalesCube" visible="true" cache="true" enabled="true">
    <Table name="sales" schema="public">
    </Table>
    <DimensionUsage source="Product" name="Product" visible="true" foreignKey="item" highCardinality="false">
    </DimensionUsage>
    <DimensionUsage source="Store" name="Store" visible="true" foreignKey="store" highCardinality="false">
    </DimensionUsage>
    <Measure name="Sales" column="quantity" datatype="Numeric" aggregator="sum" description="Number of the sold items" visible="true">
    </Measure>
  </Cube>
```

Modifying the generated XML (2/2)

- This is the edited XML file, ready to be used by JasperServer. We also show the equivalent DFM view

```xml
<Schema name="TutorialSchema">
  <Dimension type="StandardDimension" visible="true" name="Product">
    <Hierarchy visible="true" hasAll="true" primaryKey="id">
      <Table name="producthierarchy" schema="public" />
      <Level name="Type" visible="true" column="type" type="String" uniqueMembers="false" />
      <Level name="Category" visible="true" column="category" type="String" uniqueMembers="true" />
      <Level name="Product" visible="true" column="product" type="String" uniqueMembers="true" />
    </Hierarchy>
  </Dimension>
  <Dimension type="StandardDimension" visible="true" name="Store">
    <Hierarchy visible="true" hasAll="true" primaryKey="id">
      <Table name="storehierarchy" schema="public" />
      <Level name="State" visible="true" column="country" type="String" uniqueMembers="false" />
      <Level name="Region" visible="true" column="region" type="String" uniqueMembers="true" />
      <Level name="City" visible="true" column="city" type="String" uniqueMembers="false" />
    </Hierarchy>
  </Dimension>
  <Cube name="SalesCube" visible="true" cache="true" enabled="true">
    <Table name="sales" schema="public" />
    <DimensionUsage source="Product" name="Product" visible="true" foreignKey="item" />
    <DimensionUsage source="Store" name="Store" visible="true" foreignKey="store" />
    <Measure name="Sales" column="quantity" datatype="Numeric" aggregator="sum" description="Number of the sold items" visible="true">
    </Measure>
  </Cube>
</Schema>
```

References (II)

- VV. AA.: “**Mondrian 3.0.4 Techical Guide: Developing OLAP Solutions with mondrian/JasperAnalysis**”. March 2009.

… to practice: querying a DataWarehouse

III. MultiDimensional eXpressions

Multi**D**imensional e**X**pressions

- **MDX** stands for *Multidimensional Expressions*.
- It is a widely supported SQL-like OLAP language for querying multidimensional databases. It directly manipulates the multidimensional cube, by setting which are the dimensions and the measures to be considered.
- MDX was first introduced as part of the OLE DB for OLAP specification in 1997 from Microsoft.
- Now several tools (JasperServer, Microsoft Server, Pentaho, …) could handle MDX Queries. We will show JasperServer at the end of the BI lectures.

MDX Types (1/2)

- <u>Levels</u> and <u>hierarchies</u> are expressed through a dot notation:

  `[Product].[Type].[Category].[Product]`

- Through the same notation, even <u>elements</u> could be expressed:

  `[Product].[Food].[Dairy Product].[Milk]`

- MDX uses the hierarchy formulation to surf the elements and to select them. Each fact is identified by a <u>tuple</u>:

  `([Product].[All Products],`

  `[Store].[All Stores])`

MDX Types (2/2)

- Tuples with the same dimensionality could be collected with <u>sets</u>.

```
{([Product].[Food],[Store].[All Stores]),
 ([Product].[House Cleaner],[Store].[All Stores])}
```

- Data could be expressed in strings and in scalars. MDX deals even with temporal representations, and in this case specific aggregation and selection functions could be used.

MDX: Defining the Cube (1/2)

- The MDX queries could be used to define the initial cube over which we will next perform our operations:

```
[WITH <new measures or new sets>]
SELECT <measures> ON COLUMNS,
       <dimensions> ON ROWS
FROM <cube>
[WHERE <slicing>]
```

- As an example, we will use a data cube that has the following DFM diagram:

MDX: Defining the Cube (2/2)

- The simplest query allows to select all the members of a specific dimension

```
select [Measures].Members ON COLUMNS,
  [Store].Members ON ROWS
from [SalesCube]
```

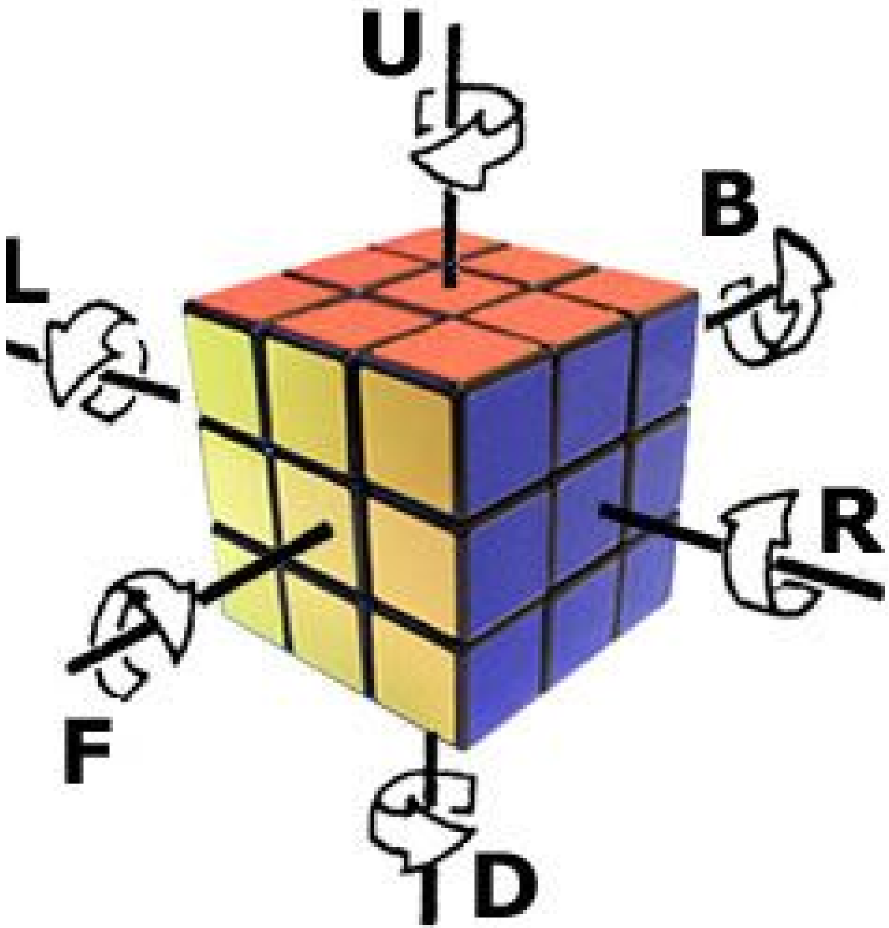
Selects all the possible measures

Expands the Store hierarchy towards the leaves

Dimensions				Measures	
Store	State	Region	City	Sales	Fact Count
All Stores				728	33
All Stores	Italy			728	33
	Italy	Emilia-Romagna		268	14
		Emilia-Romagna	Bologna	154	7
			Ferrara	114	7
		Lazio		216	7
		Lazio	Roma	216	7
		Veneto		244	12
		Veneto	Padova	122	6
			Rovigo	122	6

OLAP Operators: **pivot**

- **Pivot** allows an analyst to rotate the cube in space to see its various faces from different perspectives.



MDX: Pivoting

- Changes the order in which the rows appear inside the tuple

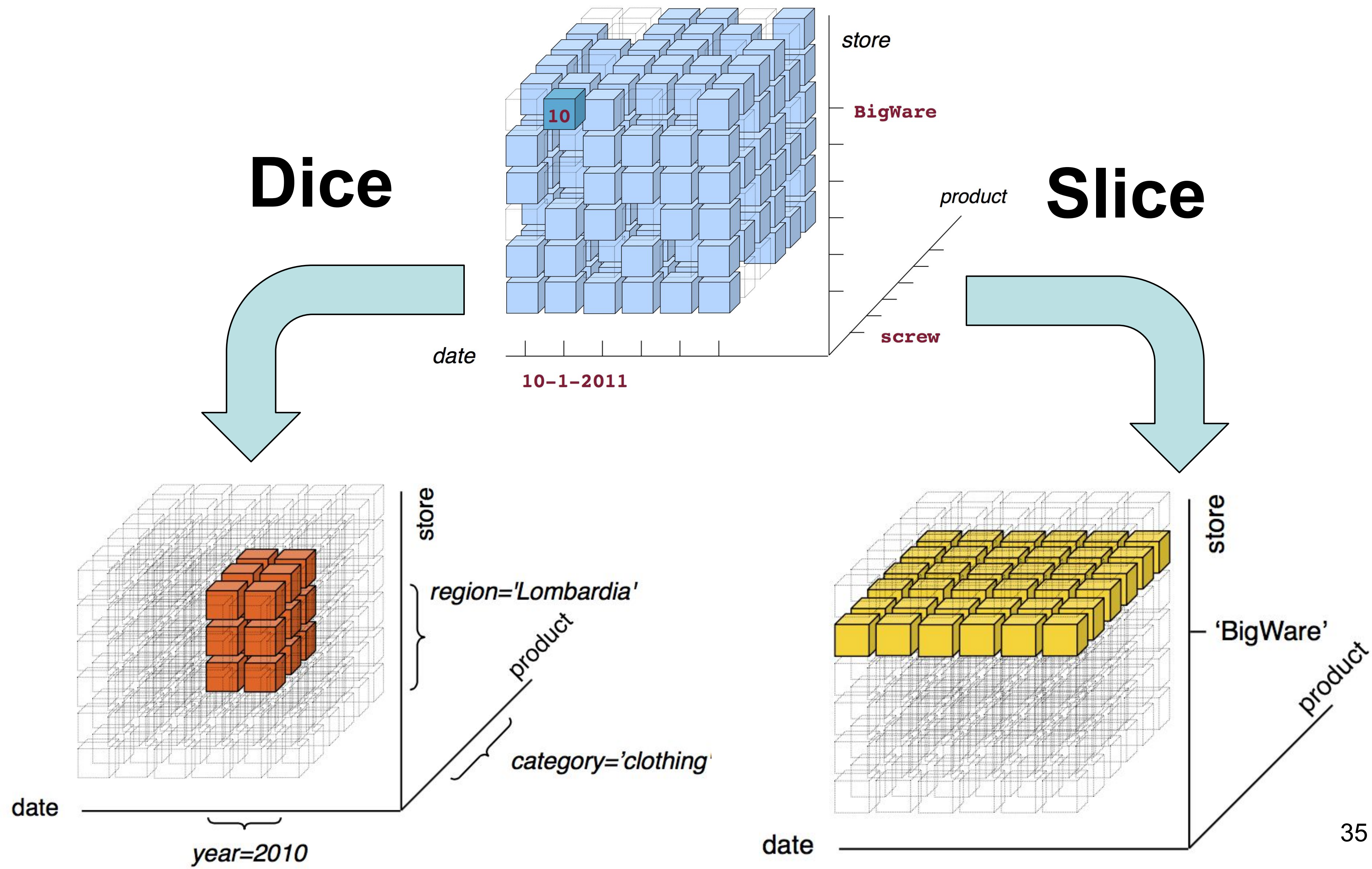
```
select {[Measures].[Sales]} ON COLUMNS,
  ([Store].[All Stores],
   [Product].[All Products]) ON ROWS
from [SalesCube]

select {[Measures].[Sales]} ON COLUMNS,
  ([Product].[All Products],
   [Store].[All Stores]) ON ROWS
from [SalesCube]
```

OLAP Operators: **slice & dice** (1/2)

- **Slice** picks a rectangular subset of a cube by choosing a single value for one of its dimensions.
- Allows dimensionality reduction.

- **Dice**: The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions.
- Provides a subset of the data keeping the dimensions' size inhalterated.

OLAP Operators: **slice & dice** (2/2)



MDX: Slicing

- As the *OLAP Slice*, the element that is selected will not be showed as a dimension.

```
select [Measures].[Sales] ON COLUMNS,
  [Store].[All Stores] ON ROWS
from [SalesCube]
where [Product].[Food].[Dairy Product].[Milk]
```

Selects all the stores: compressed hierarchy

Selects the Milk as a product

Dimensions				Measures	
Store	State	Region	City	Sales	Fact Count
All Stores				728	33
All Stores	Italy			728	33
	Italy	Emilia-Romagna		268	14
		Emilia-Romagna	Bologna	154	7
			Ferrara	114	7
		Lazio		216	7
		Lazio	Roma	216	7
		Veneto		244	12
		Veneto	Padova	122	6
			Rovigo	122	6

MDX: Dicing

- Dicing is carried out similarly to Drill-Down: specific dimensions and values are selected

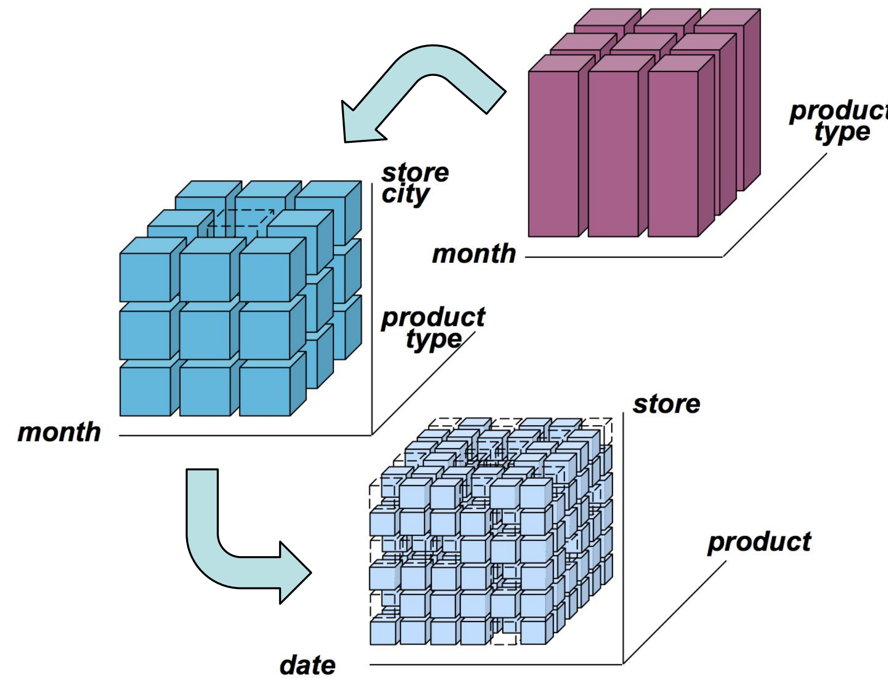
```
select {[Measures].[Sales]} ON COLUMNS,
([Store].[Italy].[Emilia-Romagna],
 [Product].[Food]
) ON ROWS
from [SalesCube]
```

Dimensions					Measures
Store	State	Region	Product	Type	Sales
All Stores	Italy	Emilia-Romagna	All Products	Food	184

Filter:

OLAP Operators: **drill-down**

- **Drill-down**: Allows the user to navigate among levels of data ranging from the most summarized to the most detailed (dis-aggregates the data)
- Has an inverse operation: **roll-up**



MDX: Drill-Down (1/2)

- In order to expand a specific element of the hierarchy, we have to use *crossjoins* between dimensions

```
select {[Measures].[Sales]} ON COLUMNS,
   Crossjoin({[Store].[All Stores]}, [Product].
[Type].MEMBERS) ON ROWS
from [SalesCube]
```

same as *

Dimensions			Measures
Store	Product	Type	Sales
All Stores	All Products	Food	473
		House Cleaner	255

Filter:

MDX: Drill-Down (1/2)

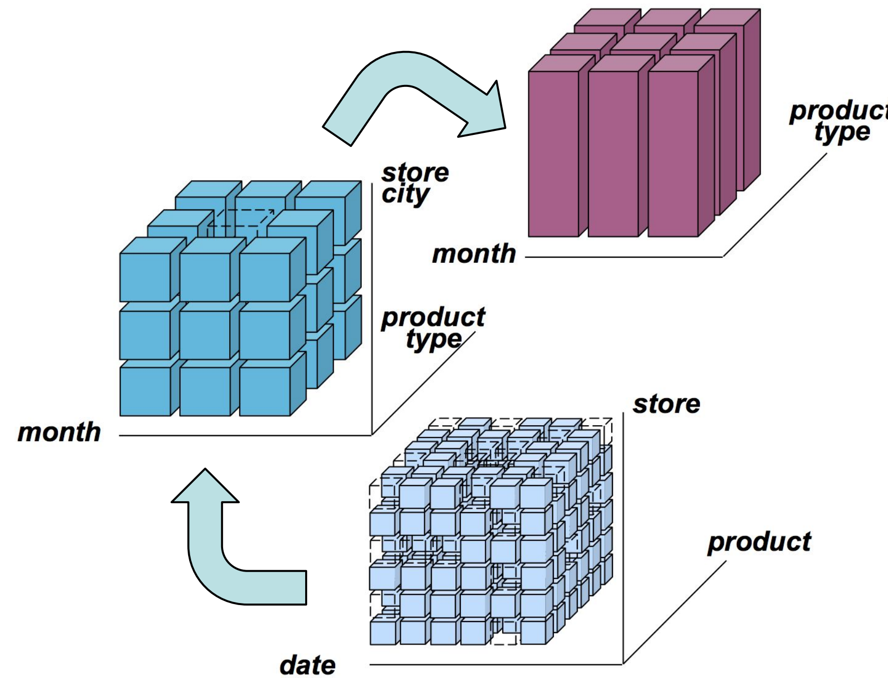
- In this case, even *unions* and *set differences* are used in order to drill down specific elements inside the hierarchy.

```
select {[Measures].[Sales]} ON COLUMNS,
  Union (
    Except(([Store].[Italy].Children *
              [Product].[All Products]),
             ([Store].[Italy].[Lazio] *
               [Product].[All Products]),
    ([Store].[Italy].[Lazio].Children *
     [Product].[All Products])
   ) ON ROWS
from [SalesCube]
```

All Stores	Italy	Emilia-Romagna		All Products	268
		Veneto		All Products	244
		Lazio	Roma	All Products	216

OLAP Operators: **roll-up**

- **Roll-Up**: aggregates the data along a dimension. The rule might computine operations along a hierarchy or apply a formula (e.g. "profit = sales - expenses”)
- Has an inverse operation: **drill-down**



MDX: Roll-Up (1/2)

- Similarly to the Drill-Down process, you simply have to specify which level of the hierarchy are interesting to view.
- The Mondrian Schema already makes explicit how the measures should be aggregated while traversing the hierarchy. The aggregation operations are automatically performed during the *ROWS* selection for each *COLUMN*.
- Further attributes could be added to the cube at runtime. We want to obtain for each element of the hierarchy the ratio between the son’s and the father’s measure “Sales”. (see next slides).

MDX: measures at run time (1/2)

```
WITH
MEMBER [Measures].[Count Ratio To Parent] AS
    IIF( ([Measures].[Sales],
          [Store].[All Stores].CurrentMember.Parent) = 0,
        NULL,
        [Measures].[Sales] /
        ([Measures].[Sales],
         [Store].[All Stores].CurrentMember.Parent)
       )
, FORMAT_STRING = "Percent"

SELECT {[Measures].[Sales], [Measures].[Count Ratio To Parent]}
ON COLUMNS
, {DESCENDANTS([Store].[All Stores], 1), [Store].[All Stores]
} ON ROWS
FROM [SalesCube]
```

New measure added alongside with the MDX query

MDX: measures at run time (1/2)

New measure added alongside with the MDX query

(Tutti)	State	Region	Sales	Count Ratio To Parent
All Stores			728	Infinity%
All Stores	Italy		728	100,00%
	Italy	Emilia-Romagna	268	36,81%
		Lazio	216	29,67%
		Veneto	244	33,52%

References (III)

- A. Vaisman, E. Zimányi: "**Data Warehouse Systems: Design and Implementations**". Springer Verlag, 2014. (*Chapters 6*)
- G. Spofford et al.: "**MDX Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase, 2nd Edition**". Wiley, 2006. (*Chapter 1*)

Creating a multidimensional view of a relational database

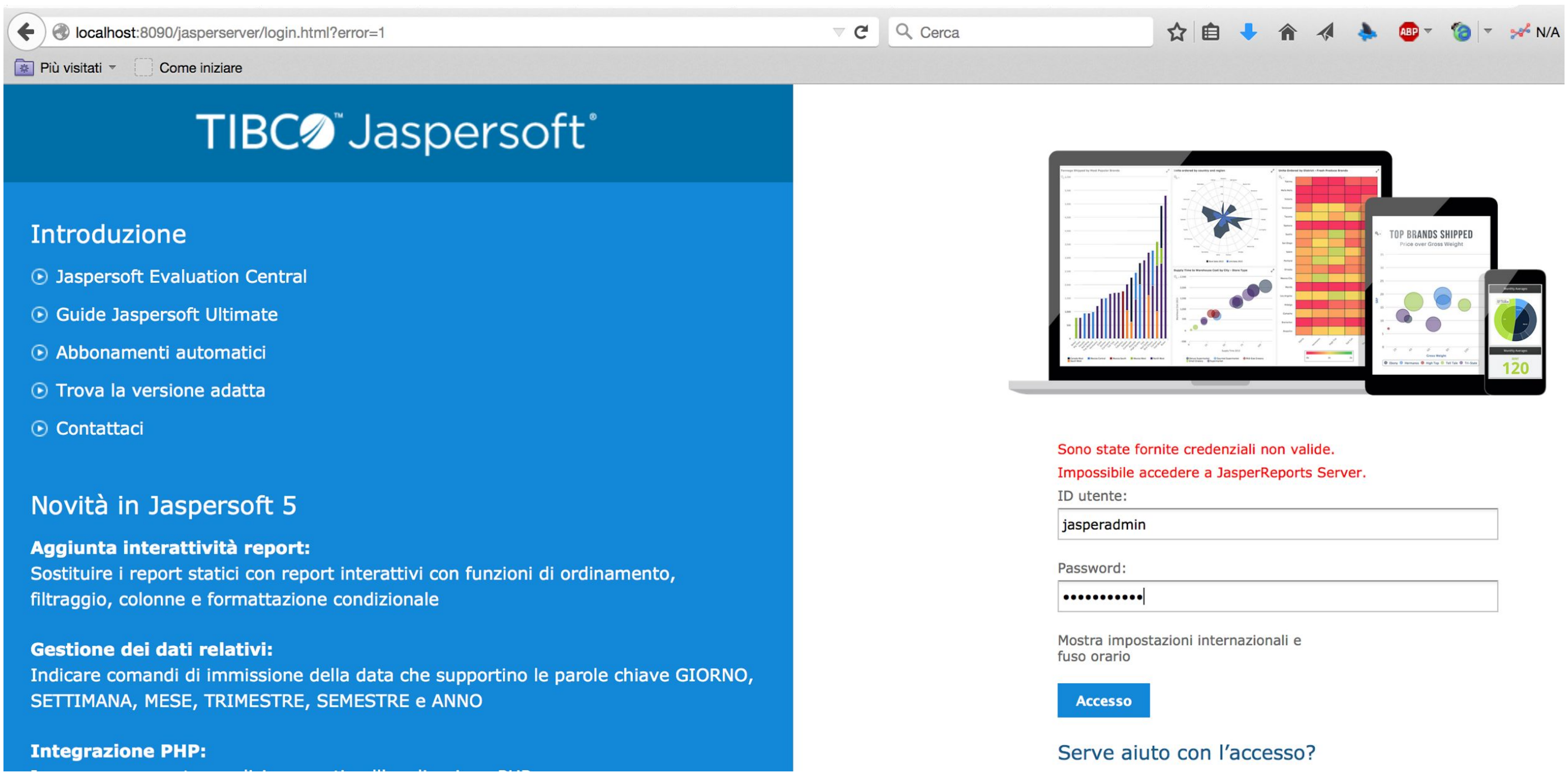
IV. JasperServer (ROLAP)

JasperServer (6.1)

- JaseperServer is a tool for directly interacting with the data cube via a tabular representation and a Histogram view of the measures.
- Jaspersoft provides other useful tools for making reports or ETLs.
- Uses (e.g.) a Mondrian Schema for defining the ROLAP mapping.
- NOTE: since both Schema Workbench and JasperServer need to access to the same database (PostgreSQL, MySQL…), it is desirable to use the system's default database.

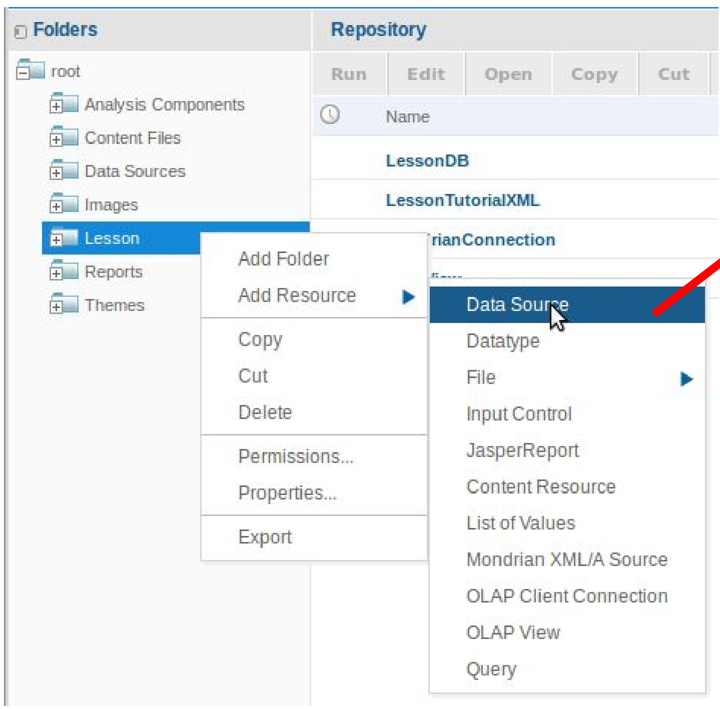
Log In

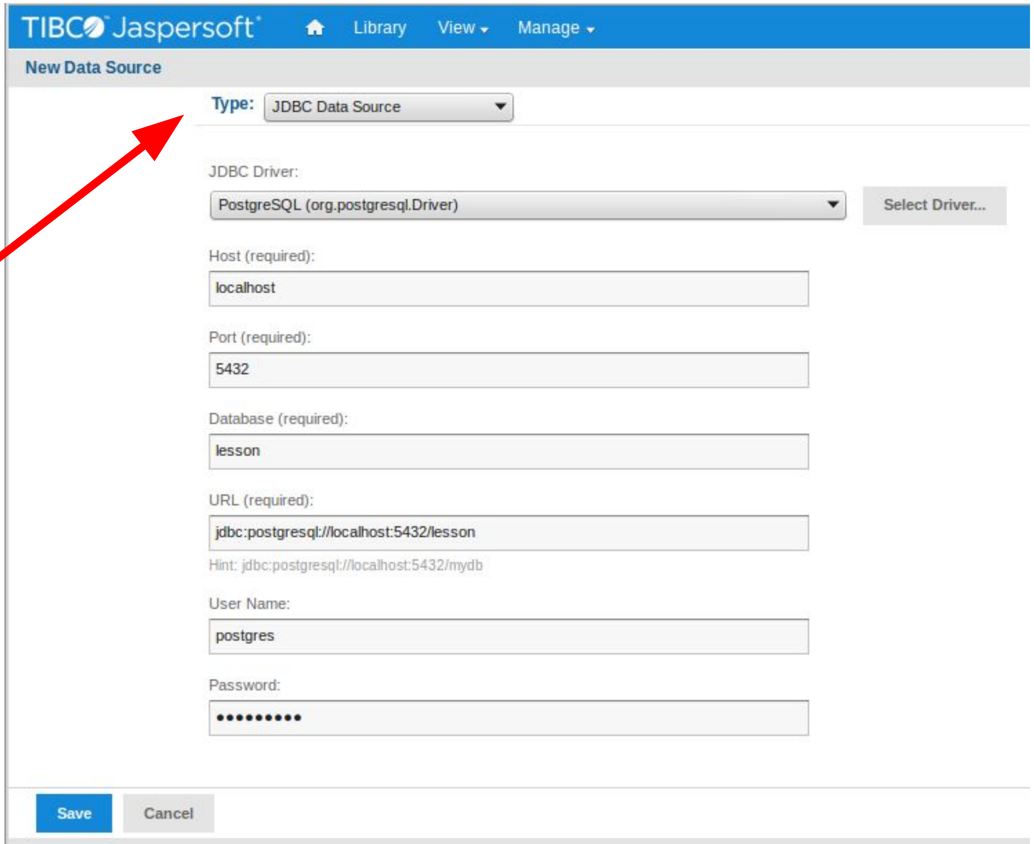
- Start the JasperServer and log-in as the administrator (user: *jasperadmin* - pw: *jasperadmin*)



Datasource

- Create a new Datasource, and then set the parameters for the database as in “Schema Workbench”. Afterwards, the database is added among the sources.

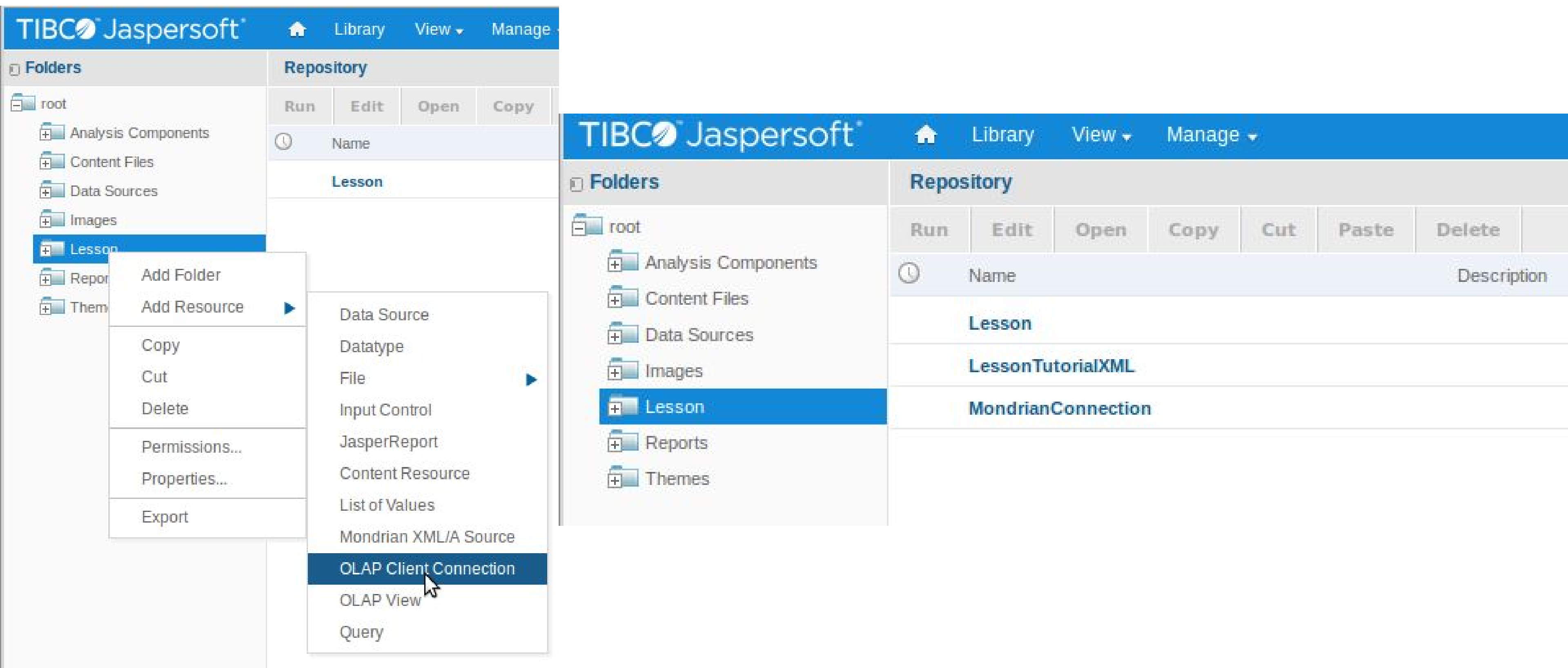




Connecting to the Datasource

- Create a client OLAP connection through the XML generated with the previous tool (LessonTutorialXML). Hereby the connection should be a “Mondrian” connection > “MondrianConnection”

- Use the “Lesson” datasource



OLAP view through MDX

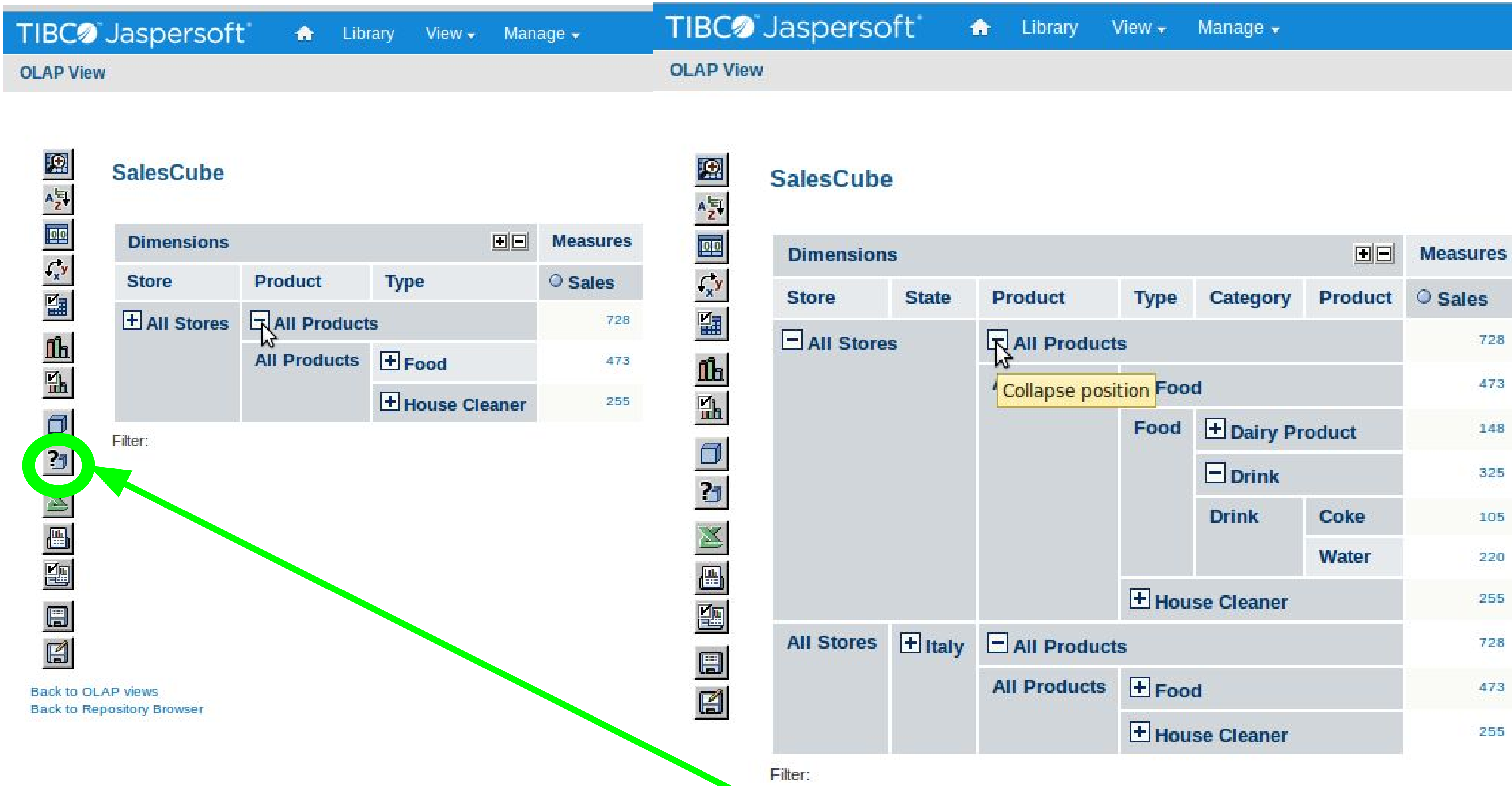
- The creation of an "**OLAP view**" requires to define the CUBE through an MDX query. In our case, a query that could show us all the expanded data is the following one:

The columns contain the measures (we could select each time which measure we would like to see)

In this case we choose to show the expanded hierarchy, hence we would like to see the set of all the possible combinations

```
select
{[Measures].[Sales]} on columns,
{([Store].[All Stores], [Product].[All
Products])} ON rows
from SalesCube
```

OLAP View



Allows to show the generated query while operating on the datacube tabular representation

References (IV)

- VV. AA.: “**TIBCO Jaspersoft® OLAP User Guide. Release 6.1**”. TIBCO Jaspersoft, 2015.