Translations

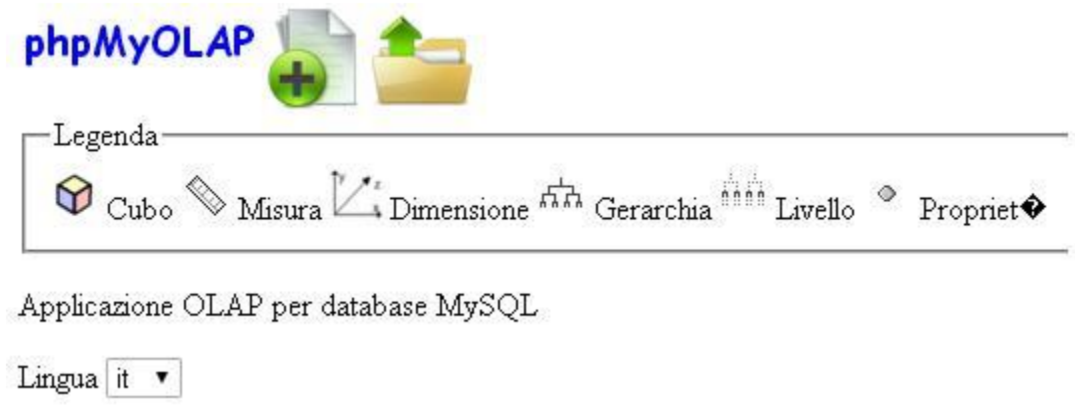
The phpMyOLAP application is multilingual because it allows you to add a file with the translations of all the used words and of all the expressions that are displayed in the GUI.
This file, with the extension .php, must be placed in the folder < *installation path* > / *lang* .

During configuration, it is possible to set the default language to be used. To do this, you need to edit the following lines of code in the *config.php file*

```
//default language file
$lang = "en.php";
```

selecting the .php file to use.

In any case, the user can always decide which language to use, among those available, during the interaction with the application. In fact, directly on the home page, there is a menu with which the user can choose the preferred language.



How to add a new language

The .php file with translations is technically an associative vector that is automatically included by the *controller* .
To add a new language in phpMyOLAP, you need to create a new .php file with the same name as the international code of the country, in which language you want to translate the texts (eg IT for Italy, EN for England, ...) . Therefore, for each element of the associative vector, the value corresponding to the text to be displayed must be set. It is recommended not to change the name of the carrier, which is `$message` , and not to change the keys of the carrier.
Eg,

```
$message["desc"]="Applicazione OLAP per database MySQL";
```

in Italian (content of the language file called it.php), becomes

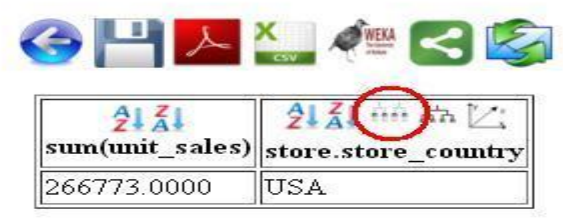
```
$message["desc"]="OLAP application for MySQL databases";
```

in English (in the file en.php).

If you create a translation file for a new language, you can create a fork on github / GitHub.

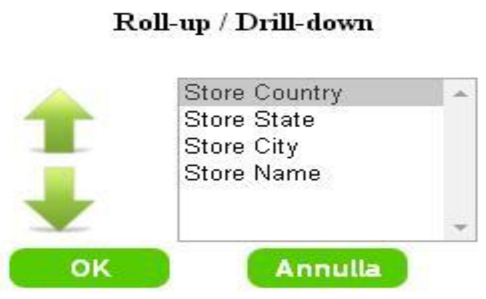
Roll-up Drill-down

Roll-up and Drill-down operations let you explore data based on hierarchical levels, as defined in the xml file containing the data warehouse metadata. After creating a report, a series of buttons are visible on the header of each column that represents an aggregation entity, or hierarchical level. One of these is the Roll-up and Drill-down button.



sum(unit_sales)	store.store_country
266773.0000	USA

Clicking on this button opens the pop-up window for selecting the aggregation level. In this way, it is possible to go down one or more levels, performing a Drill-down operation, or going up one or more levels, performing a Roll-up operation.



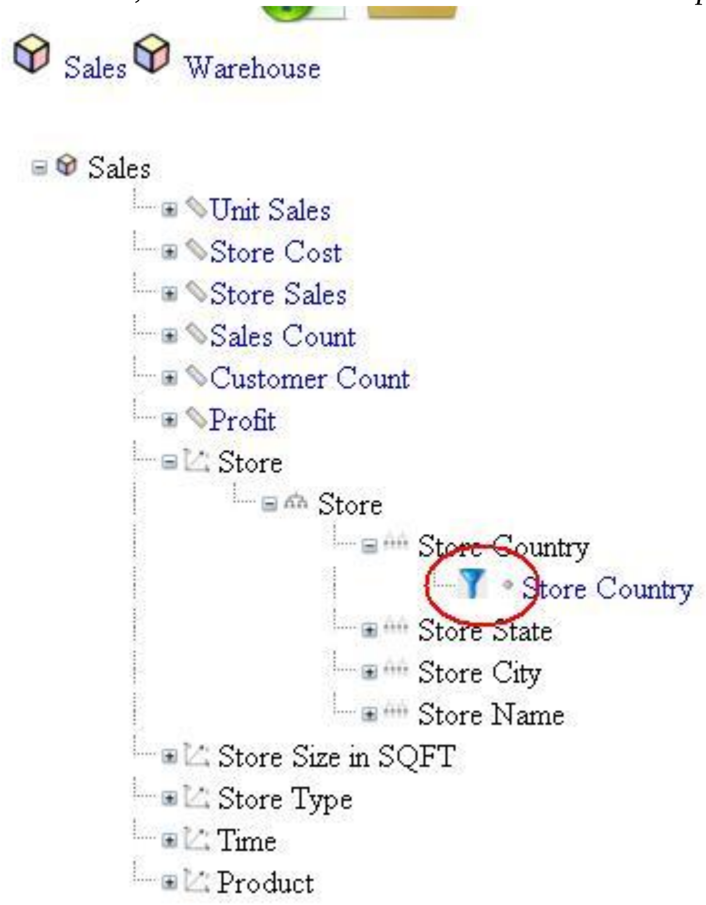
Further functions

The other two buttons allow, respectively, to change hierarchy and to change dimension. If the current dimension is made up of multiple hierarchies, you can automatically update the report by aggregating data based on the lowest level of the target hierarchy. This operation therefore allows the exploration of the data to continue, subsequently exploiting the roll-up operation in the new hierarchy.
For simplicity, while with the operations of Roll-up and Drill-down you move up and down in a hierarchy, with the operation of change hierarchy you move laterally in the dimension.
Similarly, thanks to the size change operation, it is possible to move sideways in the cube. In this way, the selected dimension is replaced by the destination size, and the report is automatically updated with the aggregated data according to the lowest level of the first hierarchy of the new dimension.

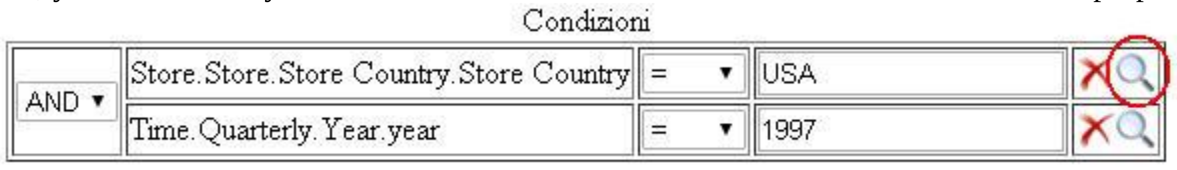
Slice-and-Dice

The Slice-and-Dice operation consists of setting selection conditions on the data, so as to cut out parts or portions of the cube. These selection conditions are usually set on the dimensions in order to set up the analysis coordinates. For example, this happens when you want to perform an analysis of the data, selecting a certain time interval or choosing a specific place.

During the exploration of the tree representing the multidimensional structure of the data warehouse, the Slice-and-Dice button is visible on the *properties* .



By clicking on this button, the property is added to the selection conditions. In the relative text field, you can manually set a value or select it from a set of values associated with that property.


Condizioni

AND	Store.Store.Store Country.Store Country	=	USA	
	Time.Quarterly.Year.year	=	1997	

The conditions can be linked to each other on the basis of the two main logical operators: AND and OR. The comparison operator can also be modified. The operators available are the classic ones for the comparison between numbers and strings, such as LIKE.

Export and sharing

The export function allows you to extract the data of the currently displayed report and save it in another format. The currently supported formats are: pdf, csv , Weka . Obviously, you can save the report directly in the application and then view it later. All those who have access to the application will be able to access the saved reports.

In addition to exporting and saving, you can send a link via email or share it on the most popular social networks.

Metadata file

The metadata file is an XML file located in the < *installation path* > / *schema* folder and contains the description of the multidimensional model of the data warehouse. In particular, it contains the names of the cubes and their measurements and dimensions, the latter organized into hierarchical levels, where each level has a set of properties. Furthermore, it describes the relationships between the multidimensional model and the relational model. Thanks to these metadata, it is possible to describe both *snowflake* patterns and *star* patterns.
The metadata file is used both during the definition of the reports, for the exploration of the multidimensional schema of the data warehouse by means of a tree structure, and during the creation of the report, for the automatic generation of queries.

First of all, it is necessary to give a name to the scheme.
```
<Schema name="FoodMart">
```

Cubes

Cubes are objects contained in the schema. Below is an example of the *Sales* cube in the FoodMart database used by Mondrian.

You must supply the cube *name* in the *name* property.
```
<Cube name="Sales" defaultMeasure="Unit Sales">
```
Immediately after, the name of the corresponding fact table is added.
```
<Table name="sales_fact_1997"></Table>
```
Then, you add the names of the dimensions with

- the relative names of the dimension tables in the *source* property
- the relative names of the foreign keys that connect the fact table to the dimension table in the *foreignKey* property

```
<DimensionUsage name="Store" source="Store" foreignKey="store_id"/>
<DimensionUsage name="Store Size in SQFT" source="Store Size in SQFT"
foreignKey="store_id"/>
<DimensionUsage name="Store Type" source="Store Type"
foreignKey="store_id"/><DimensionUsage name="Time" source="Time"
foreignKey="time_id"/>
<DimensionUsage name="Product" source="Product" foreignKey="product_id"/>
```
Then we move on to the definition of the measures, providing the name of the measure and the fact table column that contains the data.
```
<Measure name="Unit Sales" column="unit_sales" aggregator="sum"
formatString="Standard"/>
<Measure name="Store Cost" column="store_cost" aggregator="sum"
formatString="#,###.00"/>
<Measure name="Store Sales" column="store_sales" aggregator="sum"
formatString="#,###.00"/>
<Measure name="Sales Count" column="product_id" aggregator="count"
formatString="#,###"/>
<Measure name="Customer Count" column="customer_id" aggregator="distinct-
count" formatString="#,###"/>
<Measure name="Promotion Sales" aggregator="sum" formatString="#,###.00"/>
```
Any derivative measures must be inserted separately, providing, in addition to the name, the formula for calculating it.
```
<CalculatedMember name="Profit" dimension="Measures">
<Formula expression='store_sales+10' alias='profit'/>
<CalculatedMemberProperty name="FORMAT_STRING" value="$#,##0.00"/>
</CalculatedMember>
</Cube>
```

dimensions

Each dimension must be identified by a unique name.
```
<Dimension name="Warehouse">
```
A dimension can possess one or more hierarchies. Of each hierarchy, it is necessary to describe the name of the dimension table and its primary key.
```
<Hierarchy hasAll="true" name="Warehouse" primaryKey="warehouse_id"
primaryKeyTable="warehouse">
```
If the dimension corresponds to a star schema, then the hierarchy will consist of only one dimension table.
```
<Table name="warehouse"/>
```
For each hierarchical level, it is necessary to provide the name and set of its properties,
In the case of a star schema, you must indicate in the *column* property the name of the column corresponding to the level identifier.
```
<Level name="City" column="warehouse_city" uniqueMembers="false">
```
Each property must be associated with a dimension table column that contains the data.
```
<Property name="Warehouse City" column="warehouse_city"/>
</Level>
<Level name="Warehouse Name" column="warehouse_name" uniqueMembers="true">
<Property name="Warehouse Name" column="warehouse_name"/>
</Level>
</Hierarchy>
</Dimension>
```

As a further example, the case of a dimension corresponding to a snowflake pattern is given below. In this case, the *Product* dimension has a hierarchy consisting of two tables that must be joined.

```
<Dimension name="Product">
<Hierarchy hasAll="true" name="Product" primaryKey="product_id"
primaryKeyTable="product">
<Join leftKey="product_class_id" rightKey="product_class_id">
<Table name="product"/>
<Table name="product_class"/>
</Join>
...
</Dimension>
```