KNOWAGE

Community Edition Manual

Contents

Chapter 1

About conventions

Some graphical conventions have been adopted to allow readers to easily identify special contents such as notes, summaries and essential information. All conventions are explained hereafter.

Read More

This icon refers to additional documentation, internal or external sources that may be useful for the reader.

Warning

This icon warns the reader about possible errors and issues using Knowage.

Advice

This icon provides best practices and suggestions.

Notable content

This icon highlights relevant content, to be drawn to the reader's attention.

The following fonts have been adopted, to easily identify special words and expressions:

- **`Menu`**, **`Menu items`** and **`static label`** refer to specific element of Knowage GUI;
- **Input field** is a label referencing input fields in Knowage GUI;
- `Code example` is a piece of code showing configuration patterns or parts of document template.

Chapter 2

End-User guide

This chapter focuses on Knowage user interface, providing detailed information on the Main Menu, the Document Browser and some general settings concerning analytical documents. First of all, a short introduction on profiling rules its provided.

2.1 User interface

Preliminary information

To begin with Knowage you need an authorized user and login into the portal.

Figure 2.1: Log in page.

These credential identifies you as user and are associated to your *role*.

In Knowage suite, roles represent categorizations of group of users, granting each user with different rights and visiblity criteria on documents and data, according to their business profile.

Supposing the reader has a role of type “End user”, a simple menu will be shown after the login.

Main menu

Knowage Menu gives you access to documents, data and all the functionalities that you are allowed to use. By default the menu button is at the left bottom corner of the home page: clik it to open the menu, as shown in Figure 2.2.

Figure 2.2: Home page

Knowage main menu consists in a set of icons associated with basic features. It will be open selecting the hamburger-like icon on the left part of the page and it is divided in two submenus: the general menu, which is collapsed, and the BI functionalities menu. In Table 3.5 and 3.4 the corresponding elements are summed up .

Icon	Name	Description
	Knowage user	Open a hidden menu with extra functionalities.
	Select role	Select the authentication role (available if you are associated to more than one role).
	Languages	Language options.
	Info	Infos about Knowage version.
	Log Out	Log out.

Table 2.1: Menu components - General menu.

In the following we describe all the features the main menu could contain, namely:

Icon	Name	Description
	Document browser	Show the archive folders and related documents.
	Workspace	Inquiry, navigate and create your data. Available only for KnowageBD and KnowageSI.
	Functions catalog	Access data mining functions.

Table 2.2: Menu components - BI functionalities menu.

Document Browser This is a standard functionality of Knowage Server. It enables you to access all available analytical documents, arranged in folders.

Workspace This is available only for KnowageBD and KnowageSI. Entering the Workspace you will find the sections: **`Recent`**, **`Documents`**, **`Data`** and **`Analysis`**. The "Recent" area shows the latest documents you were working on, while "Documents" contains the analytical documents the user asks to be archived on this area. This way the user has a more rapid and efficient way to retrieve the documents of his interest. The "Data" section is made up of the "Dataset" and the "Models" subsections. In the **`Dataset`** section, your datasets appear (see Figure 2.3) divided into four categories:

- **`MY DATA SET`**, containing all the datasets created by you,
- **`ENTERPRISE DATA SET`**, where you can find the datasets created by the developer and released to the users,
- **`SHARED DATA SET`**, containing datasets created by other users and shared with you,
- **`CKAN DATA SET`**, available only for KnowageBD, where you can search for open data among different portals and save the selected ones in your environment.

From here you can also modify your existing datasets or create new ones for instance uploading a CSV or XLS file.

In the **`Models`** one instead you have two tabs on the right side of the interface. The **`Business`** tab allows you to access models built up for you by the developer and inquiry them using the QbE interface. In the **`Federation definition`** tab you can create federation between one or more existing dataset using the specific GUI or just access the exiting ones as well.

Finally selecting **`My Analysis`** section, available only for KnowageBD and KnowageSI, you enter a new page where you can navigate and create your self-service analysis.

The **`Functions catalog`** lets the user enter the data mining functions a technical user has previuously developed.

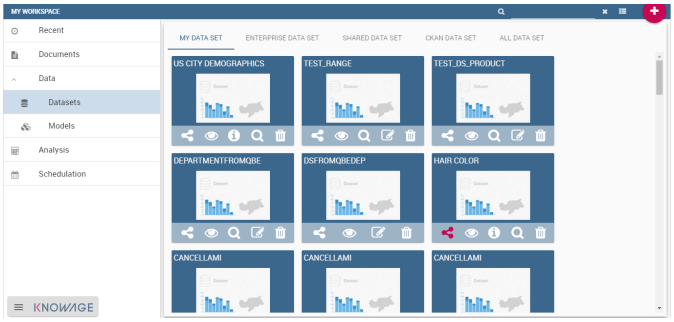


Figure 2.3: User Datasets.

The **`general menu`** is identified with the first icon in Table 3.5 and a label containing your user name. Opening the general menu you have the following extra buttons:

Select role If your user is associated with more than one role, Knowage requests you to specify the default role. You can select it when executing a document, or right after authentication by clicking on this icon and choosing a default role.

Languages Select the language of Knowage environment.

Info View the details of current Knowage version.

Document Browser overview

From BI functionalities Menu, select  to open the Document Browser.

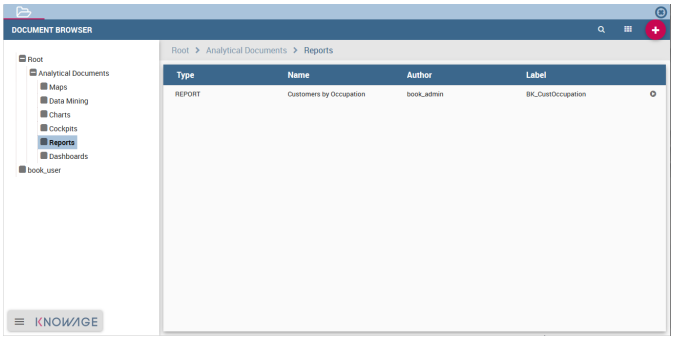


Figure 2.4: Document Browser.

By default the page is divided in two parts, as shown in Figure 2.4: in the left side there is the functionality tree representing the folder structure, while on the right you can see the list of all documents contained in the selected folder.

You can switch to the document preview view by clicking on grid icon in the top right corner, as shown in Figure 2.5.

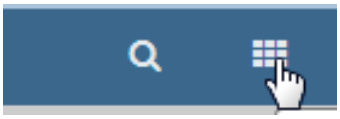


Figure 2.5: Changing documents view.

Each line shows the label, the name, the author and the type of the document, while the play button at the end of each row executes the document. Moreover, clicking on a line opens a side panel on the right of the page. Here you can see more metadata information such as the document description, the state and the creation date (see Figure 2.6).

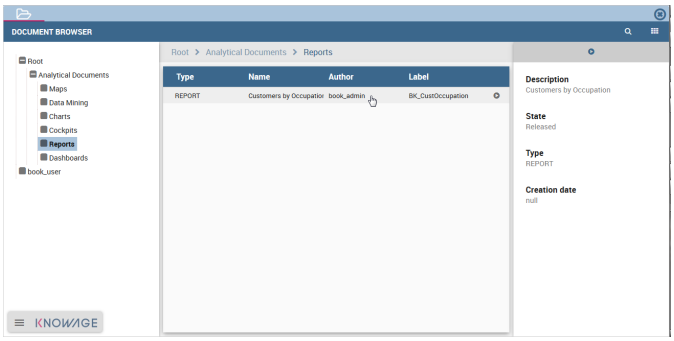


Figure 2.6: Documents details expanded.

At the top of this side panel you find the  button, the same one you see at the end of each document line. Click on it to execute the document.

2.2 Document Execution

In this section we describe all the features related to Knowage analytical documents, such as parameters management, printing, exporting and so on.

First of all, notice that once you execute a document from the document browser or from the menu, it is visualized full screen. In the first case, you can return to the document browser by clicking on the folder icon located at the top left, as shown in Figure 2.7.

Figure 2.7: Back to Document Browser.

Parameters management

Knowage documents may have associated parameters. If any, you will be asked to select the chosen parameter's values in a collapsible panel located at the top or on the right side of the page. If this is the case, choose the parameters values and then click the **Execute** button to run the document. In case there are only optional parameters or default values are already defined, the document is directly executed after the first click on its relative icon. Figure 2.8 shows an example of the parameters panel. Mandatory parameters are shown in bold together with an asterisk on the right, while optional parameters are normal shaped.

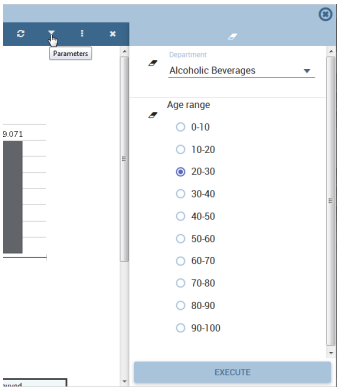


Figure 2.8: Parameters panel.

It is possible to show or hide the parameter panel by clicking on the filter button located in the document toolbar. With the **Reset** button at the top of the panel you can clear the form.

Furthermore, the parameter configuration can be saved for future use. This is particularly useful when the document includes several customized parameters. This feature is accessible from the toolbar located at the top right corner of the parameters panel.

- **Reset** inserted values for parameters;
- **Open saved** a window listing the **saved parameters**, so that you can select or modify them;
- **Save** the parameters. Here you can choose between two options: **Public** means visible to all the other users that share your role while **Private** means visible only to you.

Document Toolbar

All documents inside Knowage environment share the same toolbar with different features, see Figure 2.9. We provide first a short description and next a detailed explanation.

The button is to access the help online as defined in the Glossary and it is available only in KnowageSI.

Figure 2.9: Document Toolbar.

The refreshes the document.

The opens the parameters panel (see Figure 2.8) and it is visible only if there are parameters associated to the document.

The opens the contextual menu shown in Figure 2.10. We describe the main functionalities provided by this menu in the following.

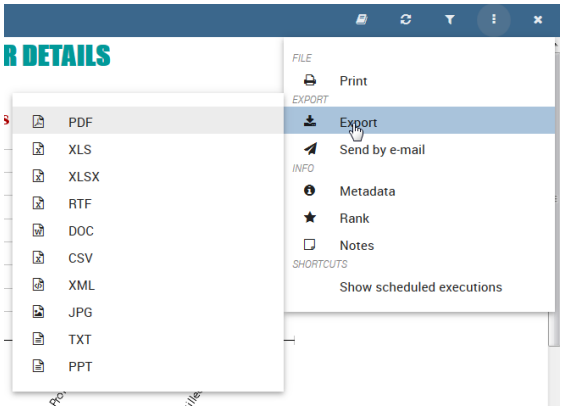


Figure 2.10: Toolbar contextual menu.

Exporters

Each Knowage document can be exported into several formats, depending on the options offered by the engine.

Clicking **`Export`** in the document toolbar, as shown in Figure 2.10, you will see the available formats for the current document. Select one and check the exported document.

2.3 Specific document guides

OLAP usage

OLAP tools enable users to analyse multidimensional data interactively from multiple perspectives. OLAP consists of three basic analytical operations: consolidation (roll-up), drill-down and slicing and dicing.

OLAP user manual step by step

We start our lecture on the OLAP engine by analysing an existing OLAP document.

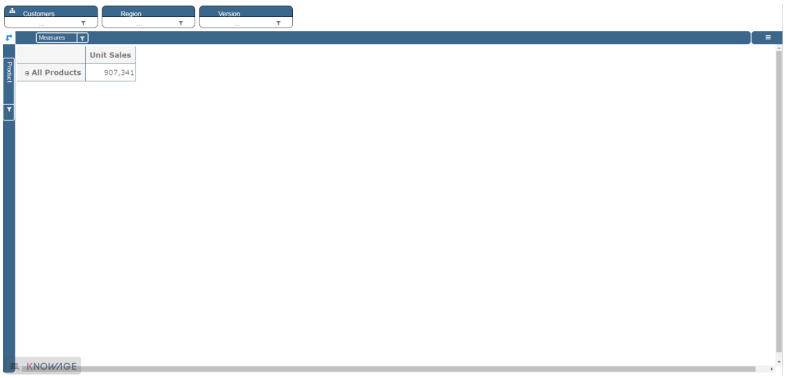


Figure 2.11: Exploring an existing OLAP document.

We will describe the main parts of the OLAP page in the following.

The filter panel

Once the document is launched we have a panel at the top of the dedicated area where filters are configured. We see in Figure 2.12 that the filter panel is made up of **`Filter cards`**. These represent hierarchies defined in OLAP schema positioned on filter axis.

Figure 2.12: The filter panel.

The filter cards

Filter cards are placed on the filter panel (Figure 2.13) and on column axis (Figure 2.14).

Figure 2.13: The filter card inside the filter panel.

Filter cards are used to:

- inform the user about available hierarchies defined in OLAP schema,

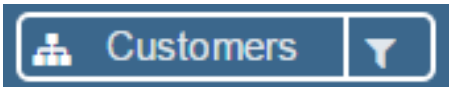


Figure 2.14: The filter cards on column axis.

- inform the user about hierarchy's name,
- perfom slices,
- choose different hierarchies,
- place hierarchies in different axes,
- filter visible members.

Considering Figure 2.15, we can see that a filter card is made up of:

- (a) an icon for launching hierarchy chooser dialog,
- (b) a hierarchy name,
- (c) icon for launching slicer chooser dialog,
- (d) choosed slicer name,
- (e) icon for launching filter dialog.

Figure 2.15: Features of a filter card.

Axes panel

The axes panel is used to:

- place hierarchy (represented with filter card) on columns or rows axis,
- position hierarchies in a particular order,
- swap axes.

Referring to Figure 2.16, the axes panel consists of the following items:

- (a) columns axis,
- (b) row axis,
- (c) filter cards,
- (d) icon for swap axes,
- (e) icon for hierarchy order.

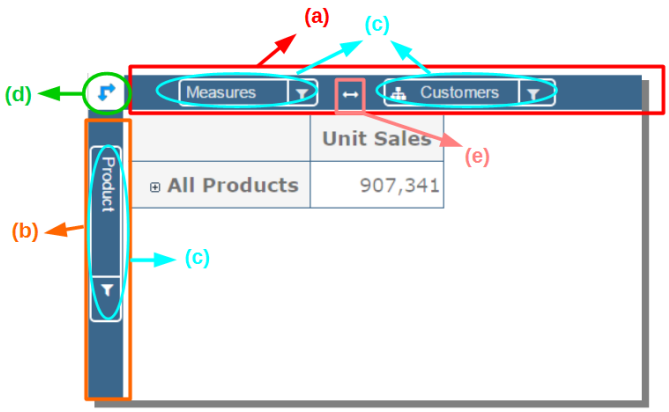


Figure 2.16: Axes panel features.

Pivot table

The Pivot table is the central part of the OLAP page. In Figure 2.17 is shown an example. Pivot

	Unit Sales															
	All Regions	Canada West	Central West é	Mexico Central	Mexico South	Marida	Merida	Mexico West	Acapulco	Acapulco	Store 1	Guadalajara	Guadalajara	No Region	North West
All Products	907,341	81,698	3,588	250,110	65,669	65,669	65,669	44,870	41,110	41,110	41,110	3,759	3,759		330,781
Drink	88,089	7,707	324	22,320	5,967	5,967	5,967	4,418	4,069	4,069	4,069	349	349		30,517
Alcoholic Beverages	23,132	2,184	115	6,391	1,586	1,586	1,586	1,320	1,220	1,220	1,220	101	101		8,261
Beverages	50,560	4,221	158	11,949	3,246	3,246	3,246	2,444	2,241	2,241	2,241	204	204		16,985
Dairy	14,397	1,301	51	3,979	1,135	1,135	1,135	653	609	609	609	44	44		5,271
Food	647,443	57,863	2,604	180,611	46,981	46,981	46,981	31,757	29,078	29,078	29,078	2,680	2,680		237,292
Baked Goods	26,508	2,411	106	7,376	1,938	1,938	1,938	1,336	1,237	1,237	1,237	99	99		9,708
Baking Goods	68,908	6,172	278	19,070	5,006	5,006	5,006	3,251	2,984	2,984	2,984	267	267		25,244
Breakfast Foods	11,753	984	53	3,228	844	844	844	503	439	439	439	64	64		4,303
Canned Foods	63,493	5,494	253	17,942	4,788	4,788	4,788	3,227	2,965	2,965	2,965	262	262		23,070
Canned Products	6,085	549	35	1,793	480	480	480	296	285	285	285	11	11		2,062
Dairy	42,967	3,837	191	11,774	3,067	3,067	3,067	2,094	1,896	1,896	1,896	198	198		15,880
Deli	40,692	3,515	150	11,397	2,970	2,970	2,970	1,901	1,720	1,720	1,720	181	181		15,252
Eggs	13,512	1,253	65	3,795	1,018	1,018	1,018	692	639	639	639	53	53		4,894
Frozen Foods	90,401	7,995	329	25,330	6,499	6,499	6,499	4,554	4,177	4,177	4,177	377	377		33,081

Figure 2.17: Pivot table.

table is used to:

- show data based on MDX query sent from the interface,
- drill down and drill up,

- drill through,
- show properties of a particular member,
- sort data,
- show calculated fields,
- show links to other documents (crossnavigation),
- write back cell values.

Referring to Figure 2.18, Pivot table consists of:

- (a) members of hierarchies placed on axes,
- (b) cells with data,
- (c) icons for drill down and drill up,
- (d) icons for sorting (if it's enabled),
- (e) icons for showing properties (if it's enabled and exists for a member),
- links for cross navigation (if it's enabled and exists for a member)

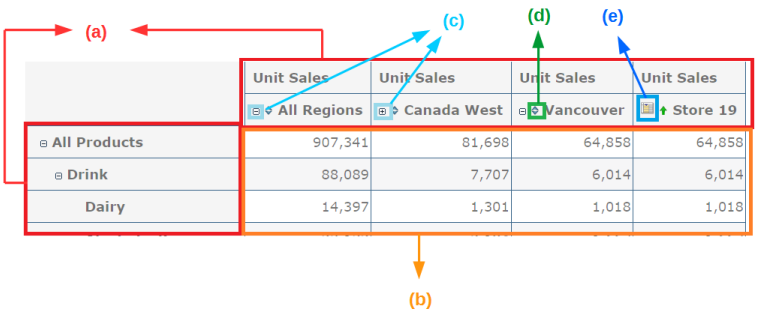


Figure 2.18: Pivot table features.

Side bar

You can open the side bar by clicking on the icon positioned on the top right side of the page (Figure 2.19). Side bar will be shown on the right side (Figure 2.20).

Side bar is used to:

- choose between different data representations,
- choose between different drill types,
- call dialogs and functionalities that effect the pivot table,

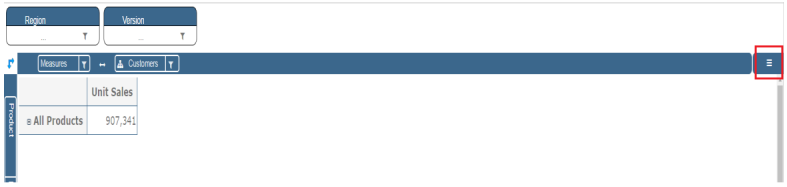


Figure 2.19: Open the side bar.

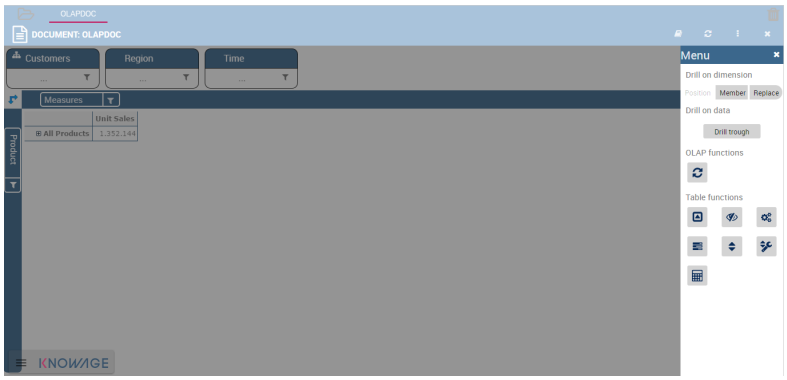


Figure 2.20: Side bar.

- get additional data based on loaded model.

The side bar shows the **Menu**, an area where options to inspect the OLAP layout are available. As highlighted in Figure 2.21, the Menu is divided in three subsection:

- (a) drill options,
- (b) OLAP functions,
- (c) table functions,
- what if.

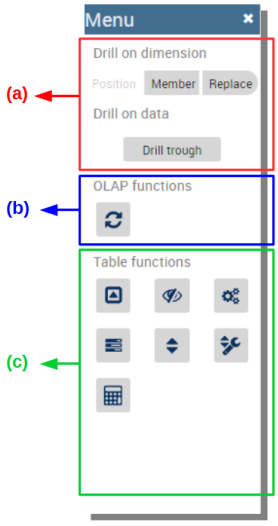


Figure 2.21: Side bar Menu.

In particular, referring to Figure 2.22, drill types consists of:

- (a) position,
- (b) member,
- (c) replace,
- (d) drill through.

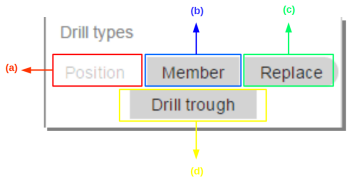


Figure 2.22: Drill types.

Referring to Figure 2.23, the OLAP functions consist of:

- (a) reload model,
- (b) show MDX,
- (c) send MDX.

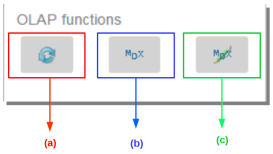


Figure 2.23: OLAP functions.

Referring to Figure 2.24, table functions consist of:

- (a) show parent members,
- (b) hide spans,
- (c) show properties,
- (d) suppress empty rows/columns,
- (e) enable compact properties,
- (f) enable sorting,
- (g) sorting settings,
- (h) calculated field wizard.

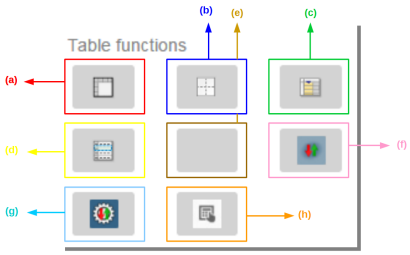


Figure 2.24: Table functions.

Functionalities

Placing hierarchies on axes

To move a hierarchy (filter card) from one axis to another the user should click and drag it to desired axis.

Let us suppose we want to move a hierarchy from the filter panel to the columns axis. The steps are summarized in Figure 2.25

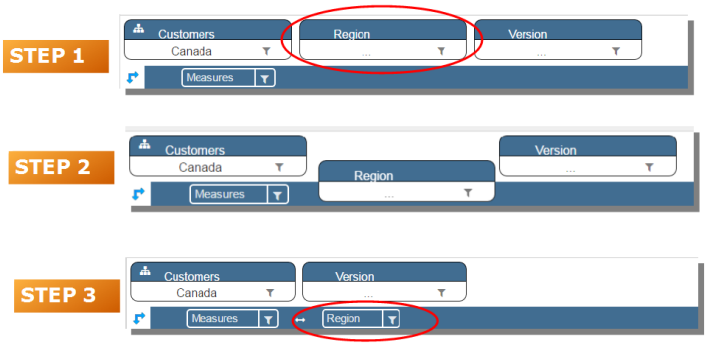


Figure 2.25: Move a hierarchy to the columns axis.

Vice versa, to move back the hierarchy from the columns axis to the filter panel the user must simply drag and drop the hierarchy from one place to the other as in Figure 2.26.

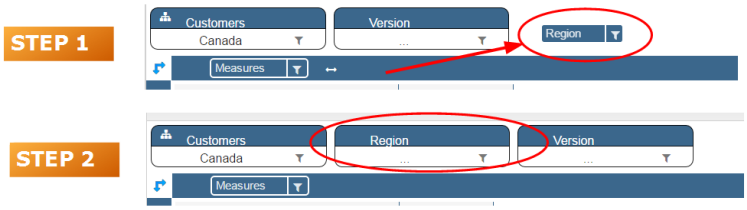


Figure 2.26: Move a hierarchy from the columns axis to the filter panel.

Similarly, a hierarchy can be moved from the filter panel to the rows axis simply dragging and dropping it from one place to the other.

Swaping axes

To swap axes the user should click on the icon . The user will get the outcome showed in Figure 2.27.

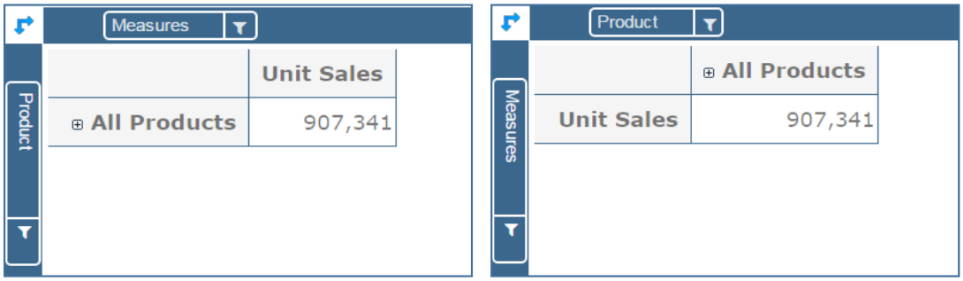


Figure 2.27: Swap axes.

Selecting different hierarchies on dimension

If an OLAP schema is defined, the user can choose different hierarchies of the same dimension. The icon for opening the dialog is positioned on the top left corner of the filter card (if the dimension has more than one hierarchy). Select the hierarchies icon underlined in Figure 2.28.

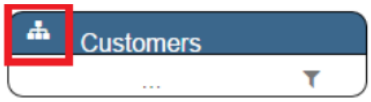


Figure 2.28: Hierarchies icon.

A pop up will be displayed. Figure 2.29 shows its characteristics. The window will present:

- (a) the dimension name,
- (b) name of selected hierarchies,
- (c) drop down list of available hierarchies,
- (d) save button,
- (e) cancel button.

After selecting the hierarchy and saving user's choice, that hierarchy will be used by the pivot table.

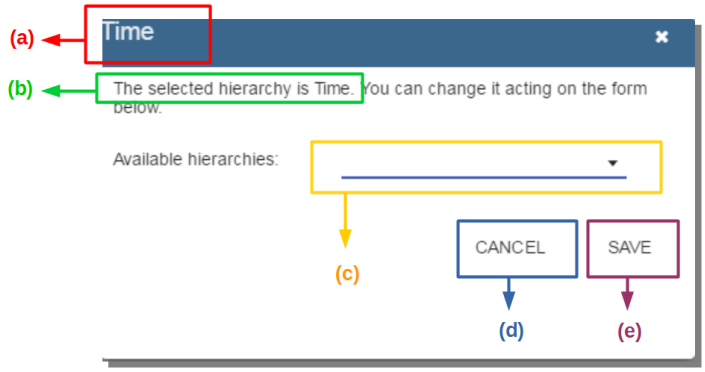


Figure 2.29: Hierarchies dialog pop up.

If the user opens the dialog window again he would see selected hieararchy and be able to change it if he needs to as shown in Figure 2.30.

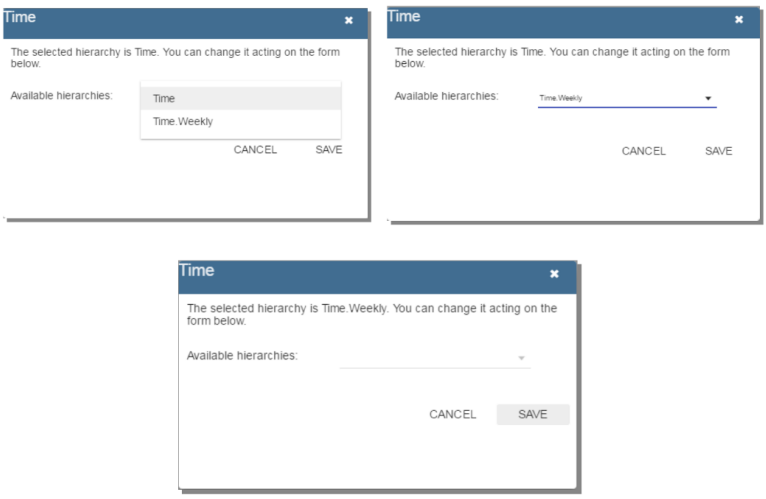


Figure 2.30: Changing the hierarchies.

We give an example of output when the hierarchy “Time” is selected in Figure 2.31 and hierarchy “Time Weekly” in Figure 2.32.

	Unit Sales															
	⊟ All Periods	⊟ 1997	⊟ Q1	⊟ February	1	2	3	4	5	6	7	8	9	10	11	12
⊞ All Products	907,341	397,544	110,531	34,208	1,458	1,656	497	819	1,459	2,386	1,597	1,359	1,783	1,354	1,591	862

Figure 2.31: Time hierachy: the table shows days in the month.

	Unit Sales								
	⊟ All Periods	⊟ 1997	⊟ Q1	⊟ February	6	7	8	9	10
⊞ All Products	907,341	397,544	110,531	34,208	1,458	9,774	9,734	8,147	5,094

Figure 2.32: Time Weekly hierachy: table shows weeks in the month.

Slicing

To perform slicing operation the user can choose a member to slice into by clicking on the icon positioned on the right bottom corner of the filter card as in Figure 2.15 (in the instance it is placed on the filter panel axis). When filter card is on columns or rows axis icon with the same simbol is used for filtering.

Referring to Figure 2.33, dialog for slicer choosing consists of:

- (a) a dimension name,
- (b) a search input field,
- (c) a search button,
- (d) a show/hide siblings checkbox,
- (e) a member tree,
- (f) a selected member icon,
- (g) a highlighted member (result of searching),
- (h) a save and a cancel buttons.

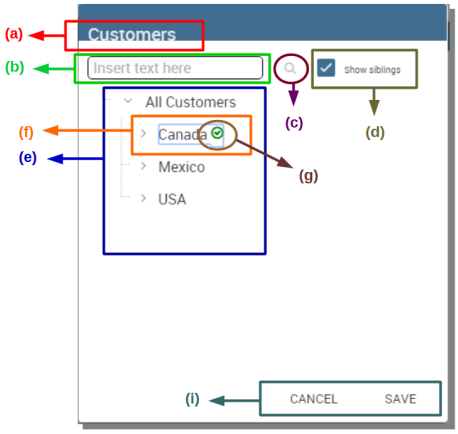


Figure 2.33: Dialog for slicer choosing.

In particular, it is possible to search for a member in three ways:

1. by browsing the member tree (Figure 2.34);

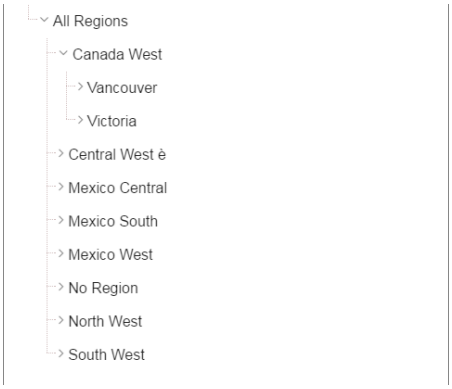


Figure 2.34: Browsing the member tree.

2. by typing member's name or it's part in the input field and clicking on the search button. The research will be possible if the user enters at least four letters. If the user wishes to include member's siblings to the research, the checkbox (Figure 2.33, (d)) needs to be checked (Figure 2.35);

3. after the first research, if the user types some other member's name before clicking on the search button, visible members whose names contains a entered text will be highlighted (Figure 2.36).

Once the selection has been saved, the users choice will affect the pivot table and the filter cards slicer name will rearrange.

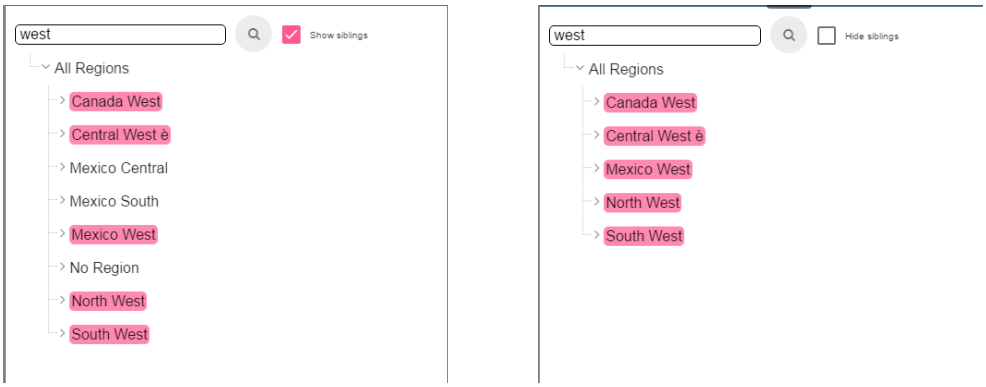


Figure 2.35: Using the research box.

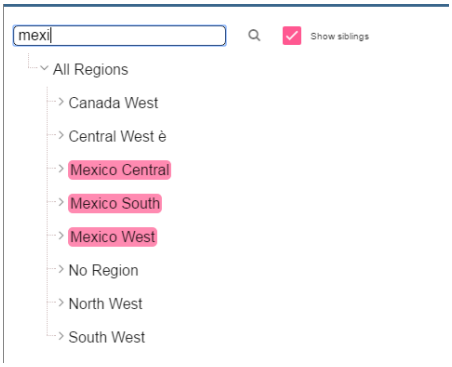


Figure 2.36: Using the research box after a first investigation.

Filtering

To filter dimension members in a pivot table, the user should click on a button (see Figure 2.15) located on the right side of dimension's filter card that is placed on one of the axes in the axes panel.

The procedure to search for a member using the filter dialog has no meaningful differences with the one described for the slicer chooser dialog. The pop up interface is the one showed in Figure 2.37. After selecting a member, the user should click on the save button. The pivot

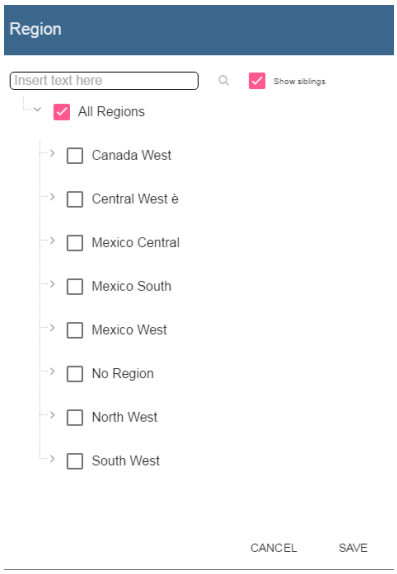


Figure 2.37: Filter dialog.

table will display the changements. Otherwise click on the cancel button to discard changes.

Drill down and drill up

User can choose between drill types by clicking on one of the three buttons in the drill types section of the side bar (Figure 2.20). There are three drill types. In the following we give some details on them.

1. **`Position`**: this is a default drill type. When selected, clicking on a drill down/drill up command will expand/collapse a pivot table with child members of a member with that particular command. See Figure 2.39.
2. **`Member`**: if the user wants to perform drill operation not only on one member but all members of the same name and level he needs to select member drill type. See Figure 2.40.

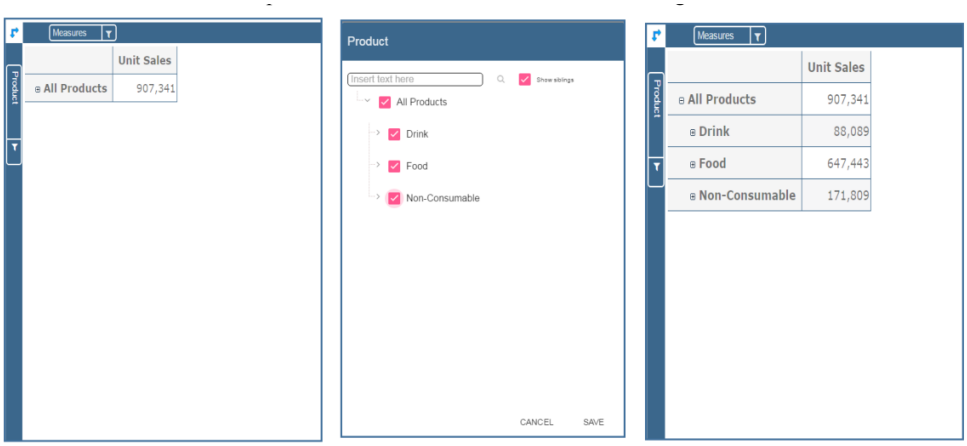


Figure 2.38: Filter effects on pivot table.

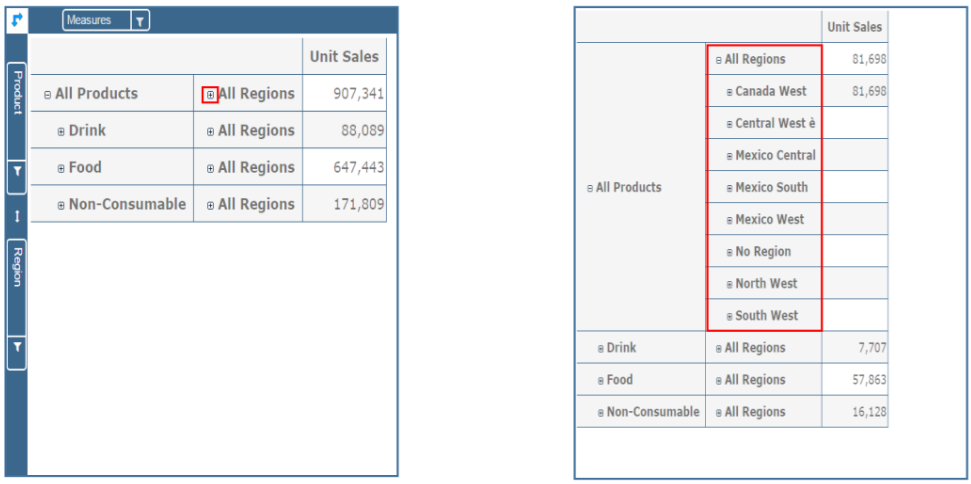


Figure 2.39: “Position” drill down.

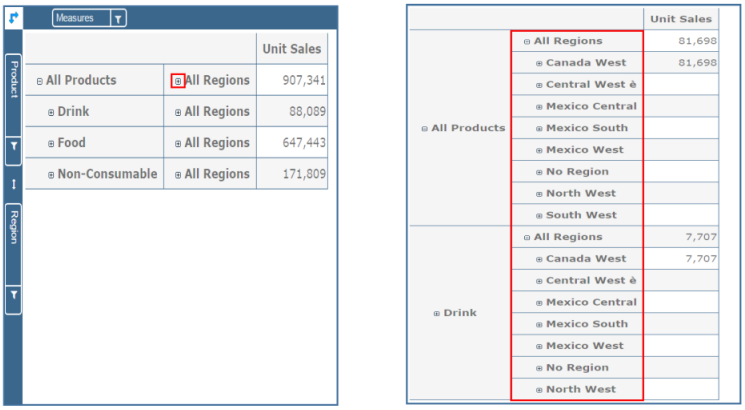


Figure 2.40: “Member” drill down.

3. **`Replace`**: for drill operation that replaces a parent member with his child members, the user needs to select replace drill type. To drill up the user should click on an icon next to dimension name on which he wants to perform operation. See Figure 2.41.

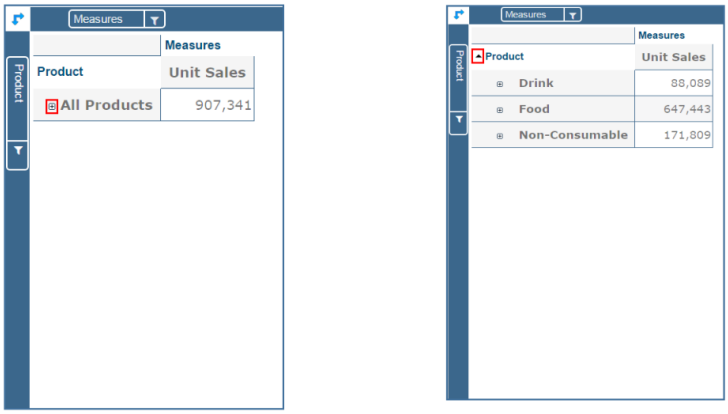


Figure 2.41: “Replace” drill down.

Drill through

To perform drill through operation the user needs first to select a cell, as in Figure 2.42, on which he wants the operation to be executed. Then clicking on the button for a drill through

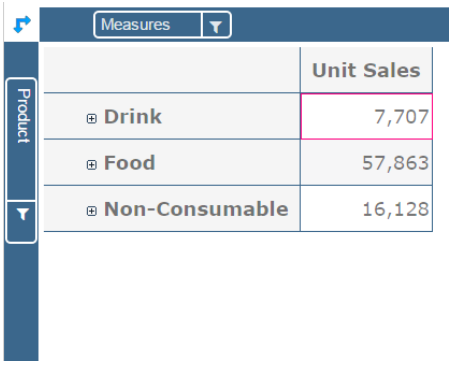


Figure 2.42: Drill thorugh option.

in the side bar, a dialog will open with results (this pop up could take some time to open).

In particular, referring to Figure 2.43, drill though dialog consists:

- (a) hierarchy menu,
- (b) table of values,
- (c) maximum rows drop down list,
- (d) pagination,
- (e) apply button,
- (f) export button,
- (g) cancel button.

The user must therefore select a cell, open the side bar and select the drill through item from the panel. A pop up will show up: here the user can choose the level of detail with which data will be displayed. The steps to follow are:

1. to click on hierarchy in hierarchy menu,
2. to check the checkbox of the level,
3. to click on the “Apply” button (after checking the checkbox, remember to click outside of the level list and then select apply).

The user can also select the maximum rows to load by choosing one of the options in the drop down list (see Figure 2.43, (c)). Finally, loaded data can be exported in csv format by clicking on the “Export” button.

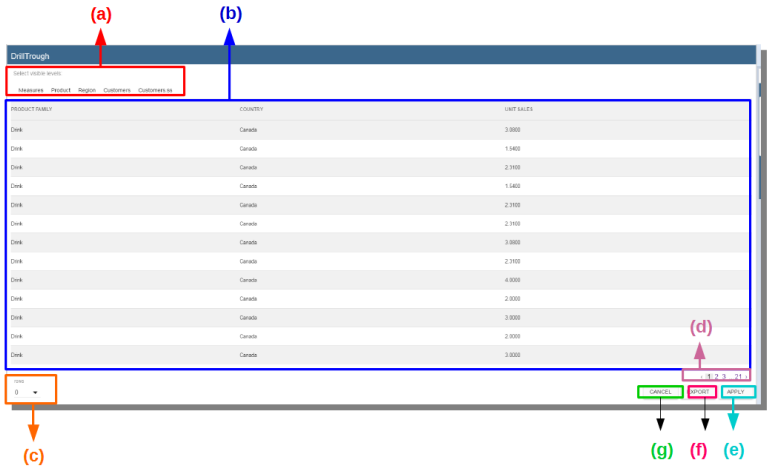


Figure 2.43: Drill thorugh window.

Refreshing model

To refresh a loaded model the user needs to click on the “Refresh” button available in the side bar panel. This action will clear the cash, load pivot table and the rest of data again.

Showing MDX

To show current mdx query user should click on show mdx button in the side bar. Figure 2.44 shows an example.

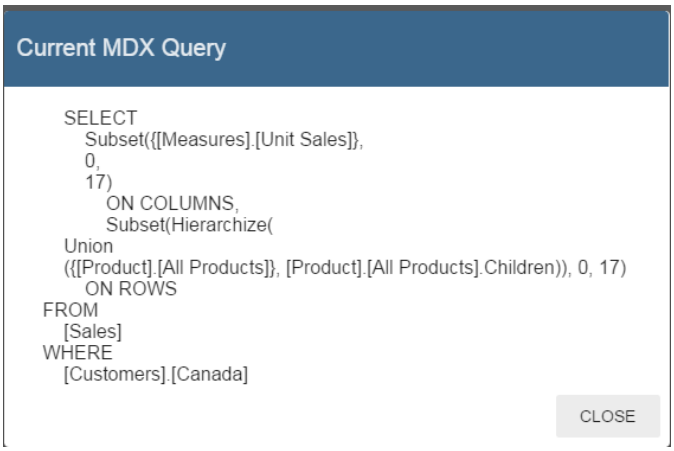


Figure 2.44: Showing MDX query example.

Sending MDX

If you want to execute an MDX query you need to:

- click on send MDX button in the sidebar,
- type a query in a text area of send MDX dialogs (Figure 2.45),
- click on the save button (Figure 2.45).

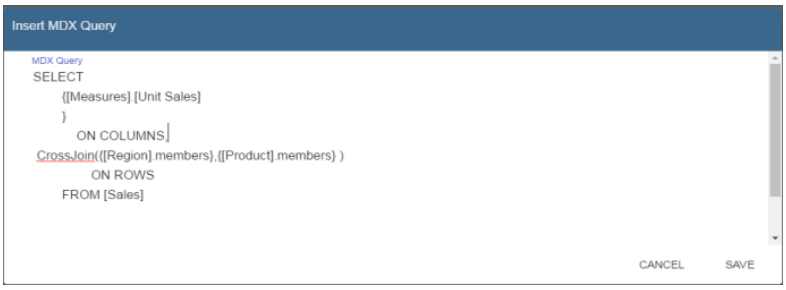


Figure 2.45: Sending MDX query example.

Result of the MDX query "should" appear in pivot table as in Figure 2.46. In fact, the user is responsable for entering *valid* MDX query.

		Unit Sales
⊞ All Regions	⊟ All Products	907,341
	⊟ Drink	88,089
	Alcoholic Beverages	23,132
	Beverages	50,560
	Dairy	14,397
	⊟ Food	647,443
	Baked Goods	26,508
	Baking Goods	68,908
	Breakfast Foods	11,753
	Canned Foods	63,493
	Canned Products	6,085
	Dairy	42,967
	Deli	40,692
	Eggs	13,512
	Frozen Foods	90,401
	Meat	6,122
	Produce	127,035

Figure 2.46: Sending MDX query example.

Showing parent members

If a user wants to see additional informations about members shown in the pivot table (for example: member's hierarchy, level or parent member) he should click on a show parent members button in the side bar panel. The result will be visible in the pivot table. An example is shown in Figure 2.47 and Figure 2.48.

	Unit Sales								
	⊖ All Regions	⊕ Canada West	⊕ Central West è	⊕ Mexico Central	⊕ Mexico South	⊕ Mexico West	⊕ No Region	⊕ North West	⊕ South West
⊖ All Products	907,341	81,698	3,588	250,110	65,669	44,870		330,781	130,626
⊖ Drink	88,089	7,707	324	22,320	5,967	4,418		30,517	16,838
Alcoholic Beverages	23,132	2,184	115	6,391	1,586	1,320		8,261	3,275
Beverages	50,560	4,221	158	11,949	3,246	2,444		16,985	11,556
Dairy	14,397	1,301	51	3,979	1,135	653		5,271	2,007
⊕ Food	647,443	57,863	2,604	180,611	46,981	31,757		237,292	90,335
⊕ Non-Consumable	171,809	16,128	660	47,179	12,721	8,694		62,973	23,453

Figure 2.47: Pivot table without the parent members mode.

			Measures								
			Unit Sales								
			Region								
Product			⊖ All Regions	All Regions							
(All)	Product Family	Product Department		⊕ Canada West	⊕ Central West è	⊕ Mexico Central	⊕ Mexico South	⊕ Mexico West	⊕ No Region	⊕ North West	⊕ South West
⊖ All Products			907,341	81,698	3,588	250,110	65,669	44,870		330,781	130,626
All Products	⊖ Drink		88,089	7,707	324	22,320	5,967	4,418		30,517	16,838
	Drink	Alcoholic Beverages	23,132	2,184	115	6,391	1,586	1,320		8,261	3,275
		Beverages	50,560	4,221	158	11,949	3,246	2,444		16,985	11,556
		Dairy	14,397	1,301	51	3,979	1,135	653		5,271	2,007
	⊕ Food		647,443	57,863	2,604	180,611	46,981	31,757		237,292	90,335
	⊕ Non-Consumable		171,809	16,128	660	47,179	12,721	8,694		62,973	23,453

Figure 2.48: Pivot table after the parent members selection.

Hiding/showing spans

To hide or show spans the user should click on show/hide spans button in the side bar. The result will be visible in pivot table as in Figure 2.49.

		Unit Sales
⊖ All Products	⊖ All Regions	907,341
	⊕ Canada West	81,698
	⊕ Central West è	3,588
	⊕ Mexico Central	250,110
	⊕ Mexico South	65,669
	⊕ Mexico West	44,870
	⊕ No Region	
	⊕ North West	330,781
	⊕ South West	130,626
⊕ Drink	⊕ All Regions	88,089
⊕ Food	⊕ All Regions	647,443
⊕ Non-Consumable	⊕ All Regions	171,809

		Unit Sales
⊖ All Products	⊖ All Regions	907,341
⊖ All Products	⊕ Canada West	81,698
⊖ All Products	⊕ Central West è	3,588
⊖ All Products	⊕ Mexico Central	250,110
⊖ All Products	⊕ Mexico South	65,669
⊖ All Products	⊕ Mexico West	44,870
⊖ All Products	⊕ No Region	
⊖ All Products	⊕ North West	330,781
⊖ All Products	⊕ South West	130,626
⊕ Drink	⊕ All Regions	88,089
⊕ Food	⊕ All Regions	647,443
⊕ Non-Consumable	⊕ All Regions	171,809

Figure 2.49: Hide/show spans.

Showing properties

In OLAP schema the XML member properties, if configured, could be represented in two possible ways:

1. as part of pivot table, as shown in Figure 2.50, where a property values are placed in rows and columns. To get these values, the user needs to click on show properties button in the side bar. Results will be shown in the pivot table;

⊖ All Regions	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊖ Canada West	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊖ Vancouver	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊖ Vancouver	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
Store 19	Deluxe Supermarket	Ruth	23112	16418	4016	2678	true	6644 Sudance Drive
⊕ Victoria	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ Central West è	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ Mexico Central	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ Mexico South	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ Mexico West	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ No Region	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ North West	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address
⊕ South West	Store Type	Store Manager	Store Sqft	Grocery Sqft	Frozen Sqft	Meat Sqft	Has coffee bar	Street address

Figure 2.50: Show properties.

2. in a pop up as compact properties. To enable compact properties user should click on enable compact properties button in the side bar. Next to a member who has propeties will be a command for calling a dialog for showing properties. Clicking on it the properties dialog will be open. Figure 2.51 shows an example.

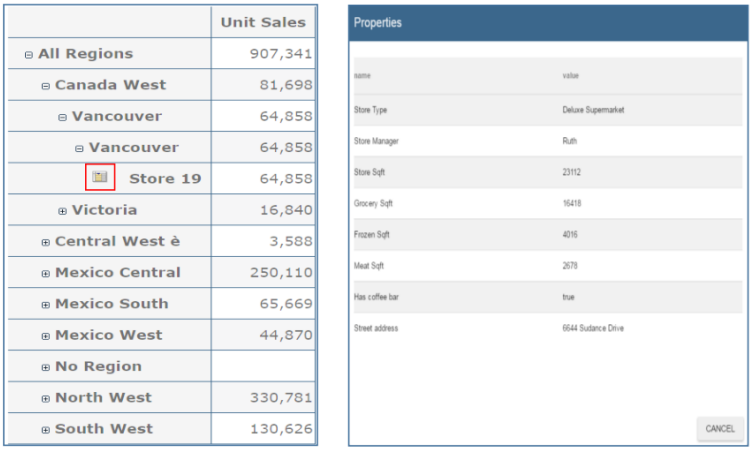


Figure 2.51: Show properties summarized in a pop up.

Suppressing empty colunms/rows

To hide the empty rows and/or colums, if any, from pivot table the user can click on the "Suppress empty rows/colums" button in the side bar panel. Results will be visible in the pivot table. An example is given in Figure 2.52.

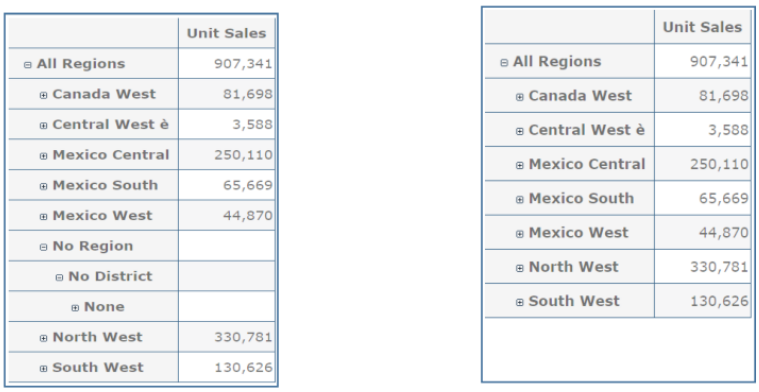


Figure 2.52: Suppressing empty colunms/rows.

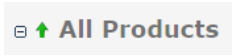
	Unit Sales						
	All Products	Drink	Alcoholic Beverages	Beverages	Dairy	Food	Non-Consumable
All Regions	907,341	88,089	23,132	50,560	14,397	647,443	171,809
Canada West	81,698	7,707	2,184	4,221	1,301	57,863	16,128
Vancouver	64,858	6,014	1,667	3,329	1,018	45,976	12,868
Victoria	16,840	1,692	517	892	283	11,887	3,260
Central West è	3,588	324	115	158	51	2,604	660
San Francisco	3,588	324	115	158	51	2,604	660
Mexico Central	250,110	22,320	6,391	11,949	3,979	180,611	47,179
Mexico South	65,669	5,967	1,586	3,246	1,135	46,981	12,721
Mexico West	44,870	4,418	1,320	2,444	653	31,757	8,694
No Region							
North West	330,781	30,517	8,261	16,985	5,271	237,292	62,973
South West	130,626	16,838	3,275	11,556	2,007	90,335	23,453

	Unit Sales			
	All Products	Drink	Food	Non-Consumable
All Regions	907,341	88,089	647,443	171,809
No Region				
Central West è	3,588	324	2,604	660
San Francisco	3,588	324	2,604	660
Mexico West	44,870	4,418	31,757	8,694
Mexico South	65,669	5,967	46,981	12,721
Canada West	81,698	7,707	57,863	16,128
Victoria	16,840	1,692	11,887	3,260
Vancouver	64,858	6,014	45,976	12,868
South West	130,626	16,838	90,335	23,453
Mexico Central	250,110	22,320	180,611	47,179
North West	330,781	30,517	237,292	62,973

Figure 2.53: Member sorting.

Sorting

To enable member sorting the user must click on the "Enable sorting" button in the side bar panel. The command for sorting will appear next to the member's name in the pivot table. In addition, the sorting command will show the members of "Measures" hieararchy or members that are crossjoined with them, as shown in Figure 2.53.

To sort members the user needs to click on the sorting command 

, available next to each member of the pivot table. Note that the sorting criteria is ascending at first launch. If the user clicks on the sorting icon, criteria will change to descending and the result will be shown in pivot table.

Note that to remove it, the user just have to click on the icon again. To change sorting mode user should click on sorting settings button in the side bar. Referring to Figure 2.54, dialog sorting settings consists:

- (a) sorting modes:
  - (b) basic (by default),
  - (c) breaking,
  - (d) count,
- (e) number input field for count mode definition,
- (f) save button.

Note that "breaking mode" means that the hierarchy will be broken.

If the user selects "Count sorting" mode the top or last 10 members will be shown by default in the pivot table. Furthermore, the user can also define a custom number of members that should be shown.

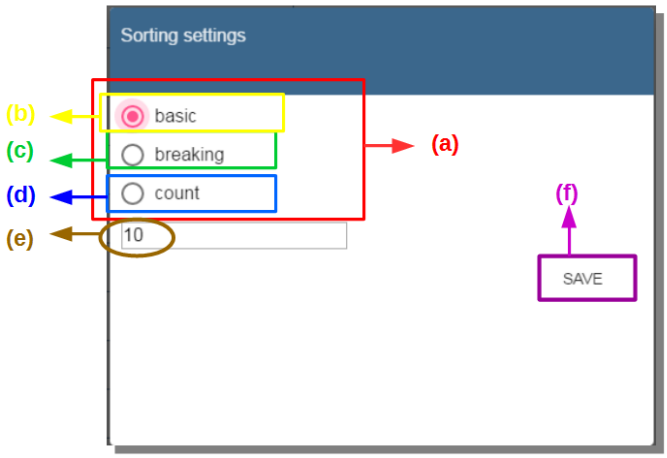


Figure 2.54: Sorting settings window.

Map usage

GIS document execution

Let's have a look on the user interface of Knowage Location Intelligence features.

In Figure 2.55 we provide an example of a BI analysis carried out thanks to map. In our example, the colour intensity of each state shown proportionally increases according to the value of the indicator selected. States who have no record connected are not coloured at all.

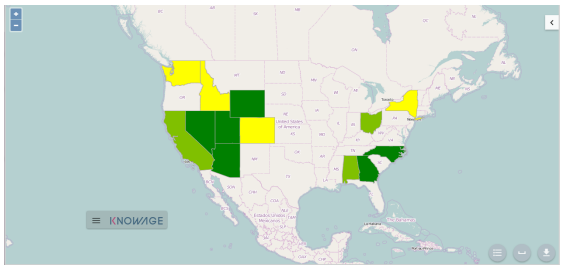


Figure 2.55: Example of GIS document. USA sales per store

Click on the arrow on the top right to open the Location Inteligence options panel. Here you can choose the **`Map Type`**, the indicators to be displayed on the map and you can enter filters.

The **`Map Type`** available are:

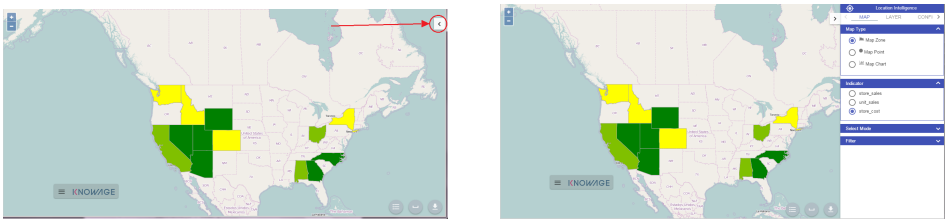


Figure 2.56: Arrow button (left) Location Inteligence options panel (right) .

- **`Map Zone`**: the different map zone are filled with different colour range according to the indicator values
- **`Map Point`**: the indicator values are displayed by points with differs on the radius. A bigger radius means a higher indicator's value.
- **`Map Chart`**: thanks to this visualization type you can compare more than one indicators simultaneously. Choose which indicators compare among the available ones. You have to mark them in the **`indicator`** panel area to visualize them. The charts appears on the map displaying the selected indicators' values.

These three typologies of data visualization on map are compared in Figure 2.57.

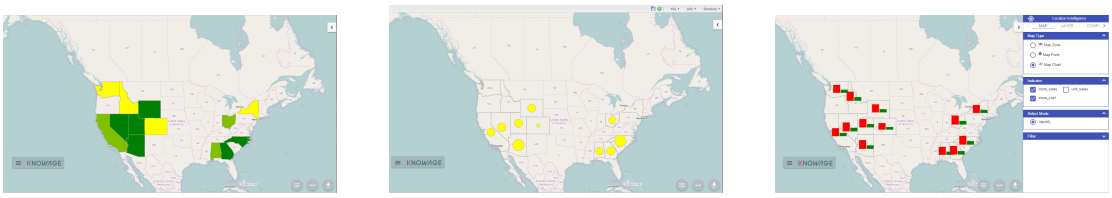


Figure 2.57: `Map Zone` (left) `Map Point` (center) and `Map Chart` (right).

Now you can add extra layers on the default one. Switch to the **`layer`** tab of the Location Inteligence options panel.

Here click on **`select form catalog`**, choose the layers you want to add. Mark them in the bottom part of the Location Intelligence area in the `Layer` box and the selected layer are displayed. These steps are shown in Figure 2.58. In our exmple we upload some waypoints, you can see the results obtained in Figure 2.59.

Now let's focus on **`Configuration`** tab of Location Inteligence panel option. Here you can set some extra configurations. Let's have a look them for each data visualization typology.

For the **`Map Zone`** you can set:

- **`Method`**: the available ones are quantiles or equal intervals. If you choose quantiles data are classified into a certain number of classes with an equal number of units in each classe. If you choose equal Intervals the value are divided in ranges for each classe, the

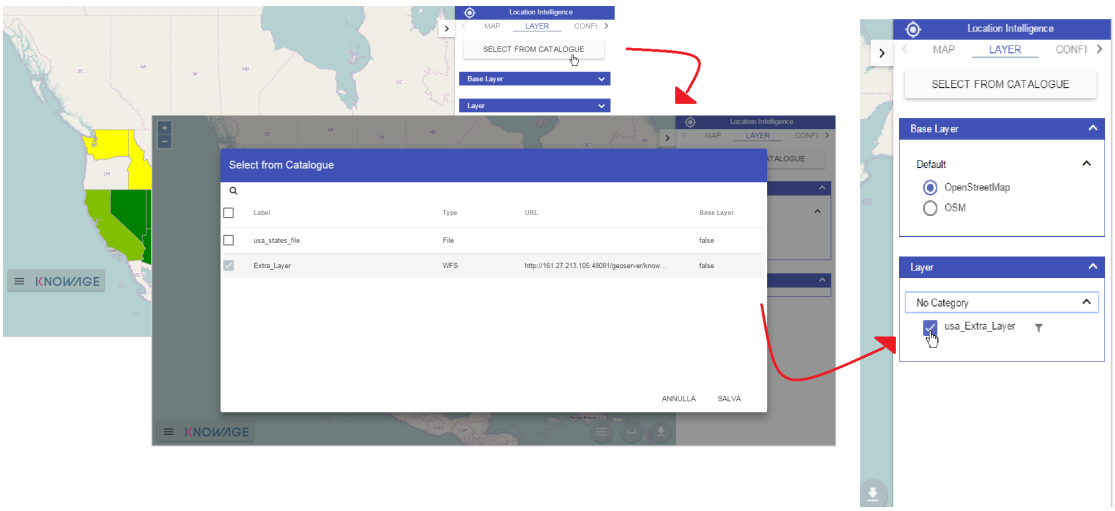


Figure 2.58: Steps for layer adding

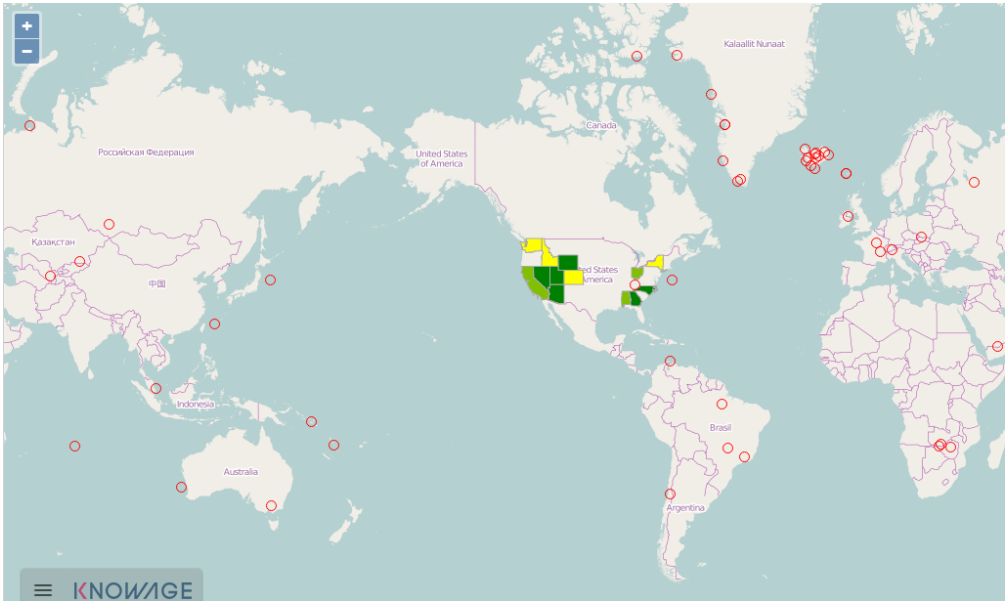


Figure 2.59: Map with two layers

classes are equal in size and their number can be set. The entire range of data values (max - min) is divided equally into however many classes have been chosen.

- **N°of classes**: the number of intervals in which data are subdivided.
- **Range colours**: You can choose the first and the last colour of the range. For both of them you can use a colour pixer by clicking on the coloured square. An example is provided in Figure 2.60.

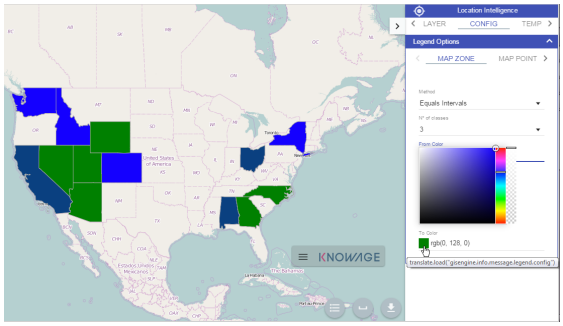


Figure 2.60: Map Zone extra configurations

For the **Map Point** you can set:

- **Colour**: the colour of the circle.
- **Min/Max value**: the minimum and the maximum circles radius.

For the **Map Chart** you can set the colour of each chart's bar.

The last tab of the panel is dedicate to the template preview, it is provided for advanced user who want to have an approach on generated code.

We can conclude our overview on Gis document describing the button located at the bottom right corner, you can see them underlined in Figure 2.61. From the left to the right this bottons can be used for: have a look at the legend, compute a measure of an area of the map and do the .pdf export of the map.

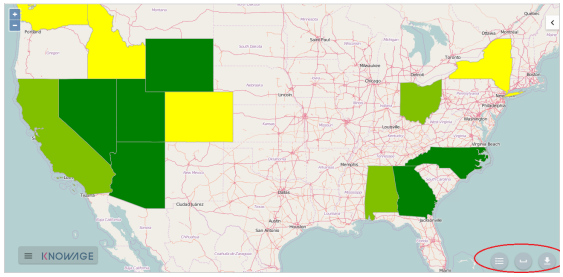


Figure 2.61: From the left to the right: Legend, Measure and Export bottom.

Extra functionalities

Let's come back to Location Layer main tab ad focus on the **`Select Mode`** area. If cross navigation has been set you find two options: **`identify`** and **`Cross navigation`**.

Select **`Cross Navigation`**, the **`Spatial Item`** tab appears. In this tab you can configure your selection. To make your selection hide CTRL key and choose the area on the map with the mouse. If you choose **`near`**, the features in the Km set are selected. If you choose **`intersect`**, the features which borders intersect your designed area If you choose **`inside`**, only the features completely inside your area of selection are considered for the cross navigation.

When selection is made, a box appears. In this box you find cross navigation information. The number of features selected and a botton to perform the cross navigation with the active selection.

Qbe usage

`QbE` (i.e., Query By Example) allows you to query (a subset of) a database through a high-level representation of the entities and relations. Its main characteristics are:

- it has a rich end user GUI;
- it allows to select attributes and set filters;
- it does not require any knowledge of data structures;
- it requires a semantic knowledge of data;
- it is useful every time the free inquiry on data is more important than their graphical layout;
- it leaves the management of results free;
- it supports export capabilities;
- it allows the repeatable execution of inquiries;
- it works on a data domain with limitations.

Building a QbE query does not require any technical knowledge, but data domain knowledge: technical aspects, such as creating filters, aggregation and ordering criteria, are managed by a user-friendly graphical interface.

In the following we discuss each step in detail, showing basic and advanced functionalities of the **`QbE Editor`**.

Query design and execution

You can open the QbE editor for querying a model directly from the **`Workspace`**> **`Models`** section, by clicking on the model icon.

In this paragraph we show how to build a simple query with the QbE editor.

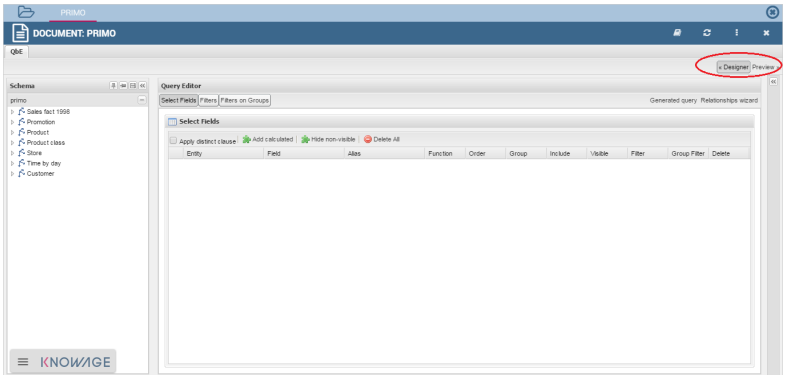


Figure 2.62: QbE editor.

As shown in Figure 2.62 the window of the QbE editor contains the **`Query designer`**.

Datamart Schema

Starting from the left panel, it shows the logic representation of the business model and the list of entities that can be inquired to generate the query. Entities and relationships are represented in a tree structure, with user-defined names. Fields can be dragged from here and dropped onto the editor area to insert them into a query.

There are two types of entities: *facts*, represented by a cube symbol (i.e., the `Sales fact 1998` entity) or *dimensions*, represented by a three-arrows symbol (i.e., the `Product` entity).

Each single entity is composed of a title, some attributes or measures and relationships with other entities. In particular, by exploding the content of an entity (i.e. `Sales fact 1998` as in Figure 2.63), you may encounter the following elements:

measure: it refers to fields associated with numeric data and additive data (e.g. number of sold items);

attribute: it refers to fields that can be associated to a category (e.g. product category);

relation: it refers to relationships or connections between two entities (e.g. relationship between the product sales fact and the product dimension).

Right clicking on an item in the tree, the contextual menu opens and shows some additional features:

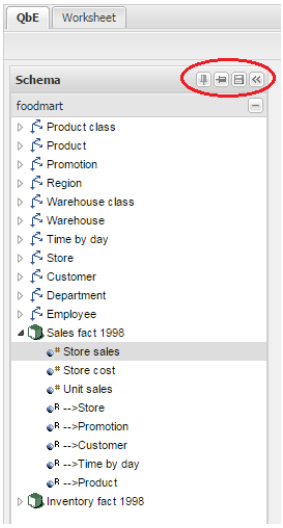


Figure 2.63: Datamart schema toolbar.

Add calculated field: to add a field that can be obtained via simple expressions combining existing fields. Clicking on the contextual menu item, the wizard opens. Here you can combine fields with arithmetic and date functions. When you create a calculated field, you can add it to the model by clicking the **Save** button located in the top right corner of the panel. In addition, they can be used in queries. Calculated fields may also be managed by expert users via advanced functionalities, which will be described at the end of this section.

Edit field: to rename a field.

Add/Edit Range: to add or manage a range of values of the selected attribute (details are provided below).

Remove calculated field: to remove a calculated field that was added before.

Let us see more in detail how to add calculated fields and ranges.

Calculated fields management

You can create new calculated fields either inside a query or in the model, that means in the damatart schema panel. Calculated fields defined in this second way can be saved for future use.

In order to define a new calculated field in the model, right click on the chosen entity and select **`Add calculated field`**. The wizard offers an editor in which you can define the calculated field.

To build a calculated field, you shall define:

- **Name**;
- **Type**: string, number or date;
- **Nature**: measure or attribute;
- **Formula**: you can click on the fields included in the item tree on the left (or drag and drop them) and build the formula.

An example is provided in Figure 2.64.

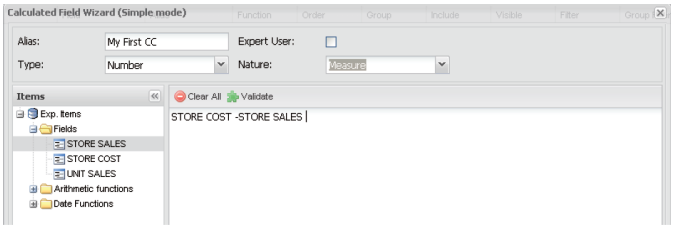


Figure 2.64: Calculated field wizard.

There are two types of calculated fields that you can add to the QbE query: *standard* and *expert*. The *standard* ones are SQL expressions that are injected into the query.

With the *expert* calculated fields (you should mark the **`Expert user`** box in the calculated fields wizard) you can build Groovy scripts, show images, add links. This second type of calculated field is computed after the query has been executed.

Query Editor

The central panel provides a query editor, including three different tabs:

- **`Select Fields`**, containing the list of columns to be returned by the query;
- **`Filters`**, containing filtering conditions on fields values;
- **`Filters on Groups`**, containing filtering conditions on aggretated measures.

Elements from the datamart schema on the left can be dragged and dropped onto the query editor tabs. If a whole entity is selected, all its attributes are dropped into the editor. Alternatively, you can drag and drop single entity fields, as said before. To remove an attribute from the query editor, just click on the dedicated icon in the delete column or select the corresponding row and press **`Delete`** on your keyboard.

The expert user can visualize the query matching his selections by clicking on the **`Generated query`** button at the top right corner of the panel. This way it is possible to check the SQL generated by the graphical interface.

Let us now see in detail the three functionalities, listed above, which split the query editor area in different sections.

Select Fields

This tab contains the list of columns to be returned by the query. To add a new attribute in this section, just click on a field in the schema panel tree or drag and drop it onto the query editor.

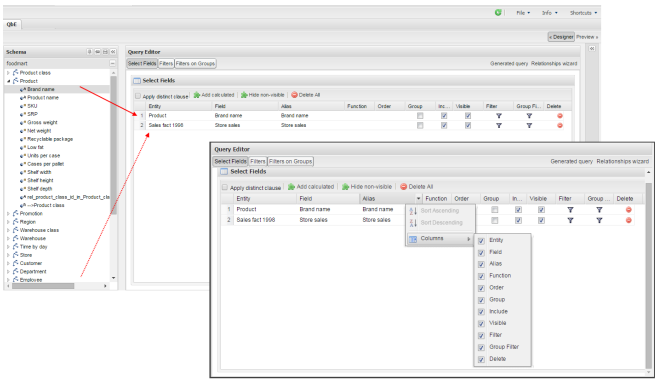


Figure 2.65: Select fields interface.

This panel is structured as a table: rows contain the attributes selected from the datamart schema, while columns include applicable functions as shown in Figure 2.65.

For each dropped item, the first two colums **Entity** and **Field** show the entity and the related attribute field respectively, and they are not editable.

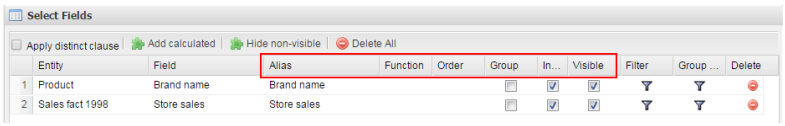


Figure 2.66: **`Select Fields`** panel options.

With the other columns, shown in Figure 2.66, it is possible to:

Alias: define aliases for fields: those aliases are shown as column headers in the result table;

Function: in case of aggregation, define the aggregation function (e.g., **`SUM`**, **`AVERAGE`**, ...) on the non-grouped items;

Order: define a sorting criteria: double click on the **`Order`** column to set the ordering criteria;

Group: in case of aggregations, define the attribute that you want to group on (if you know SQL syntax, these attributes are the ones you should place in the `GROUP BY` clause);

Include: indicate the column(s) to be included in the result (please notice that non-included attributes will not be returned by the query, but can be used in it, e.g. to apply grouping criteria);

Visible: indicate whether a column shall be visible in the result (hidden attributes are used and returned by the generated query, but are not shown in the result table);

Filter: add a filter criteria: clicking on this filter icon redirects you to the **`Filters`** tab;

Group Filter: add a filter on groups: clicking on this filter icon redirects you to the **`Filters on Groups`** tab;

Pay attention to grouping options: if you want to define an aggregation function on a field (like, for instance, the **`COUNT`** of the sold items), you shall tick the **Group** checkbox for all the other fields dragged in the **`Select Filters`** panel without an aggregation function defined, otherwise you will get an SQL exception. The possible grouping functions are shown in Figure 2.67.

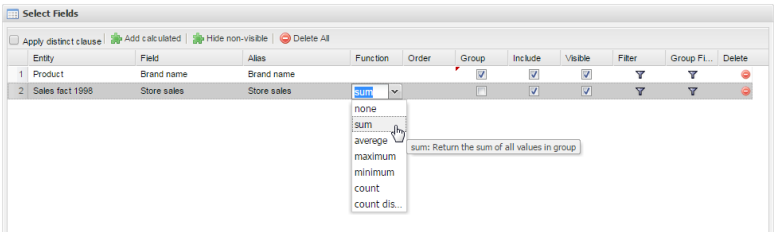


Figure 2.67: Aggregation functions.

When you drag attributes belonging to entities that are linked through a relationship path, the QbE automatically resolves relationships between attributes (implicit join).

Filters

The **`Filters`** panel allows you to define filter criteria (`WHERE` clause). Similarly to the select area, filters are structured as a table: here rows contain filters, while columns represent the elements of the filter.

There are three ways to create a filter:

- drag an attribute from the datamart schema to the **`Filters`** panel;
- click the filter symbol on the row of an attribute in the **`Select Fields`** panel;
- click the **`New`** button in the **`Filters`** panel.

To remove a filter from the query editor, select the left side of the row (multiple rows can be selected as well) and press the **`Delete`** button on your keyboard.

Filters are expressions of type:

Left operand + **Operator** + **Right operand**.

Once you have selected the left operand, you can configure the filter by using the proper setting values on columns. In particular:

- the **Filter Name** column contains the (editable) name of the filter while the **Filter Description** column contains an editable description;
- the **Left operand**, **Operator**, **Right operand** columns allow you to define filters according to the syntax defined above. Double clicking in the **Right operand** column, a lookup function is activated to facilitate selection of values; Figure 2.68 highlights the right operand selection.

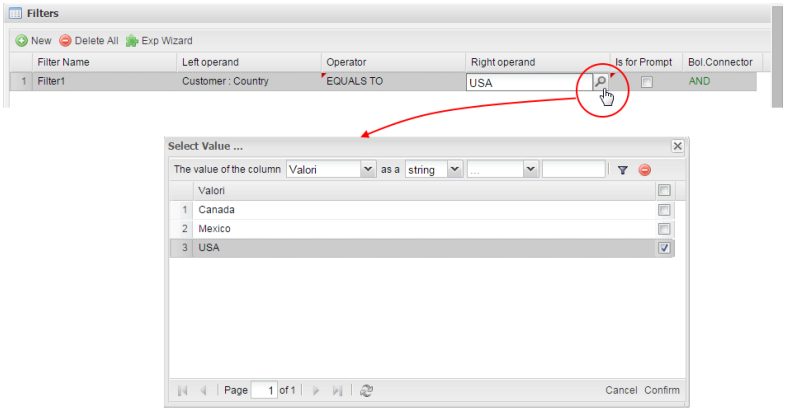


Figure 2.68: Filter lookup for right operand selection.

- the **Left Operand Type** and **Right Operand Type** columns define the types of operands;
- the **Is for Prompt** column should be checked in order to insert dinamically the value for the parameters at execution time;
- the **Boolean Connector** column shall be used to create the right expression when multiple filters are defined;

Not all available features of the editor panel are visible by default. To customize the editor appearance, double click on the arrow located on each column header and select **`Columns`**. As you can see from Figure 2.69, you can decide which columns you want to appear in the editor.

Note that more complex combinations of filters can be defined using the Expression Wizard, which you can find selecting the **`Exp Wizard`** icon.

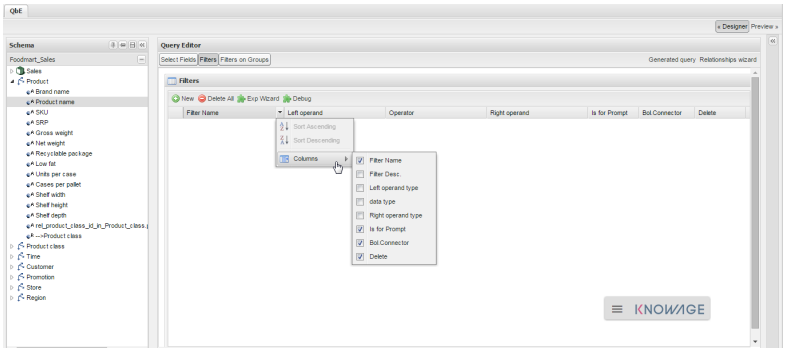


Figure 2.69: Filter editor customization.

Filters on Groups

By moving to the **`Filters on Group`** tab it is possible to define filters on aggretated measures.

Filters on groups are expressions of type:

Aggr. function + **Left operand** + **Operator** + ***[Aggr. function]*** + **Right operand**,

where the second ***[Aggr. function]*** is in this case optional. Example expessions could be, for instance, the filter "`sum(sales) > 10000`" or "`sum(sales) > sum(costs)`".

Once you have selected the left operand, you can configure the filter using the proper setting values on columns. Columns are the same as those of the **`Filters`** tab, that is the ones just described in the previous section. There are, however, additional columns related to grouping functions. In particular, the two columns named **`Function`** are visible close to the right and the left operand, respectively: define here the aggregation function to use on the left, or right, operand.

Query Preview

Once you are satisfied with your query or if you want to check the results, you can see the returned data by clicking the **`Preview`** button located in the top right corner of the panel. From there, you can go back to the **`Designer`** tab to modify the definition of the query.

In case you have started the QbE editor directly from a model (that is, you have clicked on a model icon in the **`My Data`** > **`Models`** section) from here you can also click the **`Save`** button located in the top right corner of the page to save your query as a new dataset, reachable later from the **`My Data`**> **`Dataset`** section. Please note that this operation saves the *definition* of your query and not the snapshot of the resulting data. This means that every time you re-execute the saved dataset, a query on the database is performed to recover the updated data.

We highlight that when the save button is selected, a pop up shows asking you to fill in the datails, split in three tabs:

- **Generic**, in this tab you set basic information for your dataset like its **Label**, **Name**, **Description** and **Scope**. The available values for the scope are **Public** and **Private**. If you choose **Public**, the dataset will be visible to all other users otherwise it won't.
- **Persistence**, you have the chance to persist your dataset, i.e., to write it on the default database. Making a dataset persistent may be useful in case dataset calculation takes a considerable amount of time. Instead of recalculating the dataset each time the documents using it are executed, the dataset is calculated once and then retrieved from a table to improve performance. You can also decide to schedule the persistence operation: this means that the data stored will be update according to the frequency defined in the **scheduling** options.

  Choose your scheduling option and save the dataset. Now the table where your data are stored will be persisted according to the settings provided.
- **Metadata** It recaps the metadata associated to the fields involved in your query.

2.4 Self-service data

This chapter describes extra features, i.e. available only in KnowageBD and KnowageSI products, to access data as end user.

A dataset is a way to read data from different sources and represents the portion of data used by various documents. Suppose you want to create a bar chart showing the sales trend for the current year; in this case you need to pass to the document the total sales amount for each month of the current year. You can create your own dataset uploading an XLS or a CSV file or use a dataset already defined. Knowage offers you also the chance to download open data from WEB thanks to CKAN integration. Moreover you can create your own and more complete dataset from different sources through the dataset federation. In the following we will describe all these functionalities.

As end user you can accede to the management data area by clicking on the **Workspace** icon from BI functionalities menu as shown in Figure 2.70 and entering the **Data** section of the window.

Figure 2.70: Access to **My Data** area

Consequently you have the subsections: **`Dataset`** and **`Models`**. Switch to **`Models`** to explore the models and the **`Dataset Federation`** area. Please note that **`Dataset Federation`** functionalities are available only in KnowageBD and KnowageSI.

Dataset

This area contains datasets classified in different ways. The datasets are classified as follows:

`My dataset`, here you find datasets created by your own files;

`Enterprise dataset`, the datasets in this area are the certified ones, created and shared by Knowage developers;

`Shared dataset`, these datasets have been shared by other users;

`CKAN dataset`, here you can download public datasets from web;

`All dataset`, here you find all the datasets listed.

In this area you can also create datasets uploading your own files.

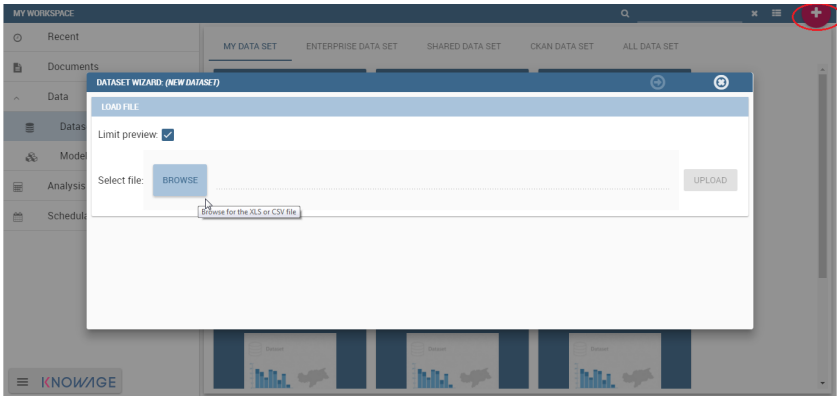


Figure 2.71: Dataset creation.

Click **`Create Dataset`** to open the dataset wizard. It guides you through dataset creation. You can choose between XLS or CSV file as in Figure 2.71. In the example shown in Figure 2.72, we upload an XLS file.

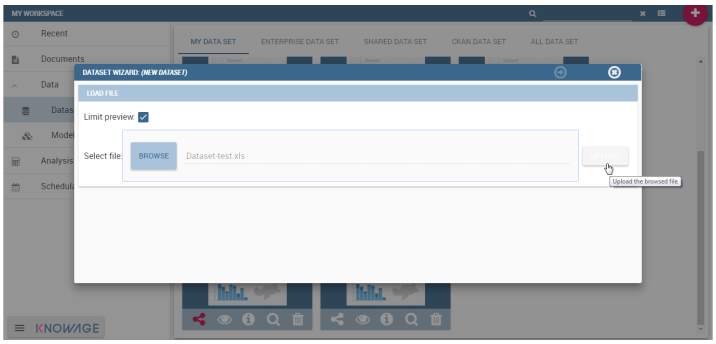


Figure 2.72: Uploading XLS for dataset.

The GUI leads the user to insert some information to configure the dataset. For instance to specify the number of rows to skip or to limit and which sheet (of the XLS file) to pick up values from (see Figure 2.73).

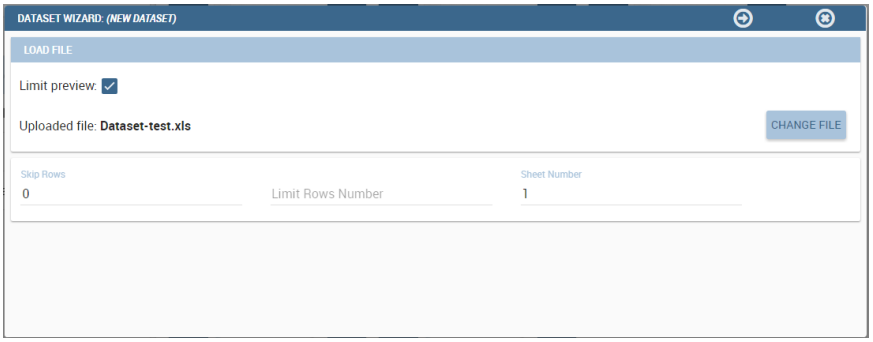


Figure 2.73: Configuration features.

Once you have uploaded the file, you can check and define the metadata (`measure` or `attribute`) of each column. To switch a `measure` to an **`attribute`** (or viceversa), click on **`Value`** column of the interested row field as shown in Figure 2.74.

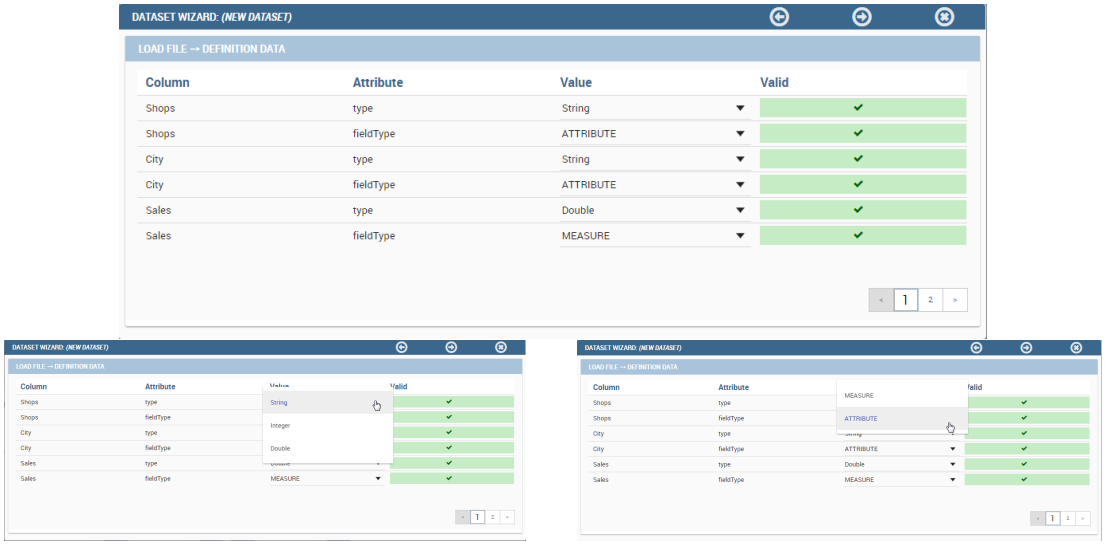


Figure 2.74: Change metadata.

Finally the user can inspect results through data preview. As last step the user is asked to insert a name to save the dataset as shown in Figure 2.75. Note that the user can decide to enable the persistance of dataset, which means that the server creates a snapshot of the extracted data and avoid to relaod it each time the dataset is revoked.

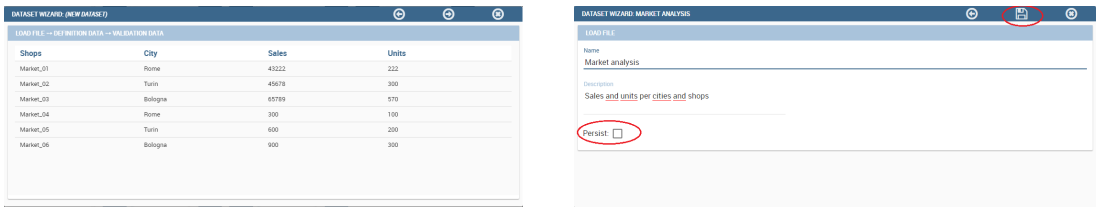


Figure 2.75: Saving dataset.

As we discussed previously, you find all created datasets under **My dataset** area. You can share/unshare them by clicking on the **share** icon (have a look at Figure 2.76). The colour of the icon changes from white to red when sharing is turned to active. A shared dataset is visible to all other users having your same role.

Note that dedicated area "**Shared Dataset**" contains all acquired datasets thanks to the sharing of other users.

Save and handle dataset

If you want to use a dataset not yet used, any action on it will start the metadata import wizard. You access it by clicking the magnifier icon. As first step, the user is demanded to insert some mandatory parameters to set the parser configuration.

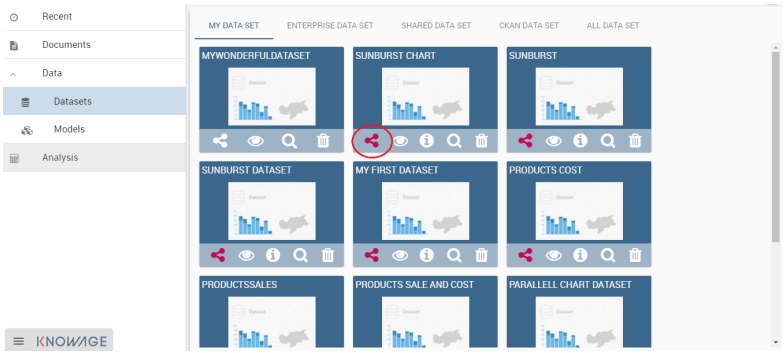


Figure 2.76: Share a dataset.

As second step of the importing wizard the user is asked to specify how the dataset will appear and to check metadata. Be careful to choose the proper data type (String, Integer, Double) and field type (Measure, Attribute). After that, click on **`Next`** to see the validation results, confirm and finalize dataset import. Once completed the dataset importation, the selected dataset will appear in the **`DataSet`** tab too. These actions just listed on the dataset change for downloded datasets. In particular you have the eye-shaped icon to refresh the dataset or change metadata by repeating the download process and the magnifier icon to inquire it throught the QbE interface.

2.5 Models

Here you find the models that the developer has built for you. You can query it using the QbE interface and create your own dataset from them. All details on how to use the QbE interface to perform free inquiries can be found in the dedicated chapter.

Dataset federation

Dataset federation is a functionality available only in KnowageBD and KnowageSI. It offers the possility to create a new dataset combinining two or more datasets, in case your role is allowed to. Let's explain it through a simple example. Suppose you have stored in a database you products information, i.e. sales, costs, promotions ecc.) and you find as open data the customers feedbacks on these products. If you contruct datasets on these resources sharing at least one column, then you can join them on the common column and improve your analysis.

Click on **`Create Federation`** to see all datasets available and choose the ones you want to federate. When you are done click **`Next`** and choose which columns the join have to be made on and click the plus icon to add it to the **`Association list`**. In our example in Figure 2.77 we choose `Product`.

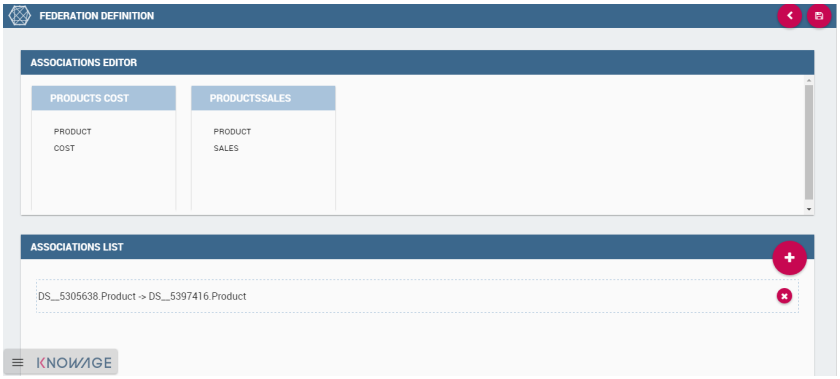


Figure 2.77: Federated dataset details.

Saving, this new federation is created in **`Federation definition`**. Open it by clicking the magnifier icon on the federation. In this way you open it with QbE tool. All details on how to use the QbE interface to perform free inquiries can be found in the dedicated chapter. You can create new datasets, save them and retrieve them from the **`Dataset`** section.

Chapter 3

Administrator guide

3.1 Administrative menu

This chapter focuses on Knowage administrative task, offered by the Main Menu.

Main menu

Knowage Menu gives you access to all its functionalities. By default you find the menu button at the left bottom corner of the home page, click it to open the menu, as shown in Figure 3.1. You can minimize it by clicking somewhere else outside the menu. In this way the menu button appears and you can reopen the menu according to your needs. You can move this button around the page by dragging and dropping it. Choose the position that best fits with your work.

Figure 3.1: Home page

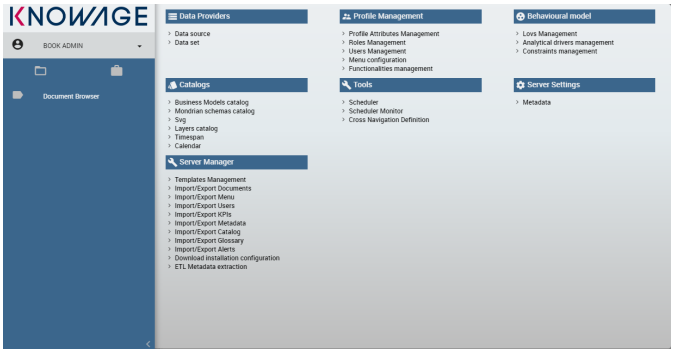


Figure 3.2: Menu

The administrator menu is divided in subpanels which maps the different managing areas:

`Data Providers` Here you can set and manage Data Sources.

`Profile Management` In this panel you can organize the users profilation, authorizations and attributes, but also organize the Analitycal model. It means you can create/manage Roles, Users and Attributes as well as configure the functionality tree and the menu, i.e. the list of quick access link to analytcal document or other resources provided to the users.

`Behavioural model` Here you manage all the Behavioural model, which means create analytical drivers and lov. In this area you can access the contraints configuration too.

`Catalogs` In this area you manage different catalogs, that may vary from product to product: the **`Business Model`** catalogues used for QbE,the **`Layer Catalogs`** for the creation of GIS analitycal documents and so on.

`Tools` In this area you can access the different scheduler options.

`Server Settings` In this panel you have access to all server settings configuration options, such as configuration or domain management.

`Server Manager` This is an optional package. It gives you access to different server functionalities, such as template management and all the import/export features.

3.2 Data source configuration

In order to connect to your data, you have to define a new data source connection. Defining a data source allows Knowage to access data transparently without the need to redefine the connection to the database in case some of its configuration properties change over time. Knowage manages two types of data source connections:

- direct JDBC connections, which are directly managed by Knowage;

Icon	Name	Management areas
	`Data Providers`	Data source settings.
	`Profile Management`	Profile Attributes, Roles, User and Menu configuration
	`Behavioural model`	Lovs, Analytical Driver and Contraints
	`Catalogs`	Business Models and Layers
	`Tools`	Scheduler
	`Server Settings`	User Data Properties, Configuration, Domains and Metadata
	`Server Manager`	Template manager and Import \Export options

Table 3.1: Menu components - Administrator Menu.

- connections retrived as JNDI resources, which are managed by the application server on which Knowage is working. This allows the application server to optimize data access, e.g., by defining connection pools.

The second type of connection is the one recommended in real projects.

To add a new connection, first add the relative JDBC driver to the folder `KnowageServer-<version>/lib` and restart Knowage. Then, login as administrator (**`biadmin`**/**`biadmin`** in the standard distribution) and select the **`Data source`** item from the **`Data provider`** panel in the administrator menu.

By clicking the **`Add`** button on the top right corner of the left panel, an empty form will be displayed on the right.

The detail page of each data source (on the right side as shown in Figure 3.3) includes the following properties:

Label Mandatory identifier of the data source.

Description Description of the data source.

Dialect The dialect used to access the database. Supported dialects are: Oracle, SQL Server, HSQL, MySQL, PostgreSQL, Ingres, DB2, AS400.

Read Only Available options are: **`Read Only`** and **`Read-and-write`**. In case the data source is defined as read-and-write, it can be used by Knowage to write temporary tables.

Write Default If a data source is set as **`Write Default`** then it is used by Knowage for writing temporary tables also coming from other **`Read Only`** data sources. Note that each

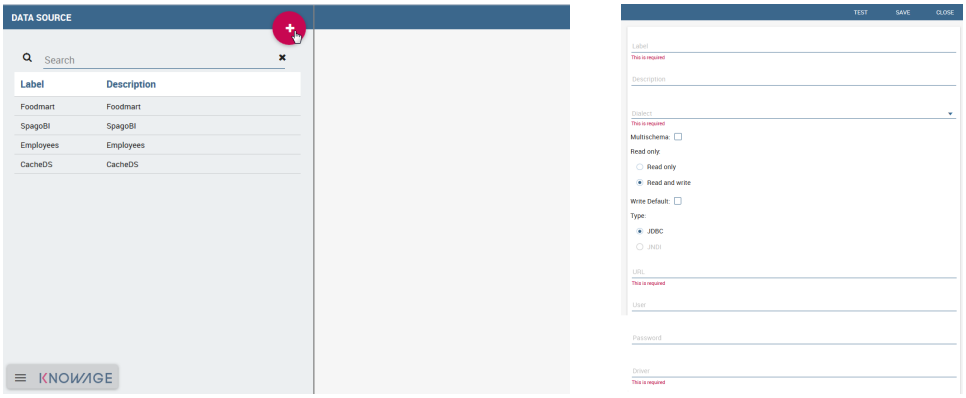


Figure 3.3: Left: Add a new data source, Right: Data source details.

Knowage installation can have only one **`Write Default`** data source.

Type The available options are **`JDBC`** or **`JNDI`**.

If you want to define a direct JDBC connection, then you have to also set the following fields:

URL Database URL. An example for MySQL databases is `jdbc:mysql://localhost:3306/foodmart_key`.

User Database username.

Password Database password.

Driver Driver class name. An example for MySQL databases is `com.mysql.jdbc.Driver`.

If instead you want to define a JNDI connection, fill in the following fields:

Multischema Available options are **`Yes`** or **`No`**. If **`Yes`**, the JNDI resource full name is calculated at runtime by appending a user's profile attribute (specified in the **`Multischema attribute`** field) to the JNDI base name defined in the `server.xml`, we suppose it has been told at the end of installation or during server configuration.

Schema attribute The name of the profile attribute that determines the schema name.

JNDI NAME It depends on the application server. For instance, for Tomcat 7 it has the format

`java:comp/env/jdbc/<resource_name>`. If the data source is multischema, then the string is `java:comp/env/jdbc/<prefix>`.

Once you have filled the form, you can test the new data source by clicking on the **`Test`** button at the top right corner of the page and then save it.

Now you are connected to your data and you can start a new Business Intelligence project with Knowage!

3.3 Data set configuration

In this section we suppose to log in as an admin user. In this case the dataset definition is no longer available under **My data** section. Otherwise the functionality is granted by the **Dataset** item under the **Data Providers** section of server menu, as highlighted in Figure 3.4. This area gives you the possibility to define datasets among a wide range of types. Moreover you can add parameters, define scope, manage metadata and perform advanced operation on datasets. While the datasets creation and management between user and admin change in favour to the latter, the **Models** and **Federation definitions** tabs available in **My data** section remain the same. For this reason in this chapter we are going to describe only the dataset creation and management.

My first dataset

As stated before, you can open the dataset graphical editor by selecting **Dataset** in **Data Provider** panel, as shown in Figure 3.4.

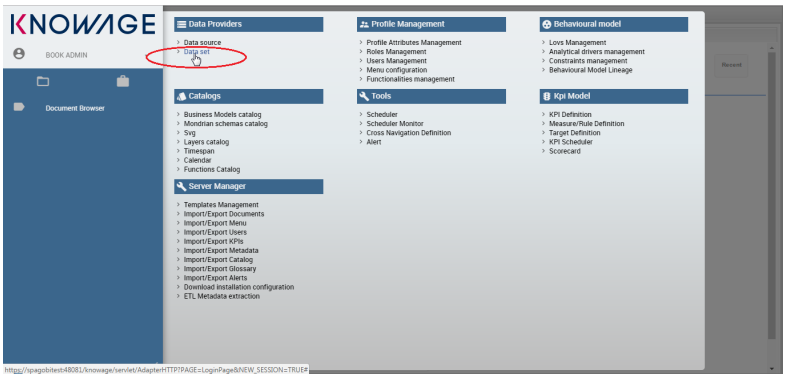


Figure 3.4: Access data set creation area.

A dataset acts as a data provider for analytical documents this is why many types are supported. Knowage manages several dataset types:

- File,
- Query,
- Java Class,
- Script (Groovy, Javascript, Embedded Javascript or ECMAScript),
- Qbe query over the metamodel (only in KnowageBD and KnowageSI),
- Custom,
- Flat,

- Ckan (only in KnowageBD and KnowageSI),
- Federated (only in KnowageBD and KnowageSI),
- REST,
- Big Data (only in KnowageBD and KnowagePM).

All types of dataset share some common operations, while others are specific to each of them. Find here below the needed steps to define a dataset:

1. choose a name and a unique label;
2. choose the type of dataset and the source, depending on the dataset type;
3. write the code defining the dataset;
4. associate parameters to the dataset, if any (optional);
5. apply transformations (optional);
6. test the dataset and save it.

Some of these steps depend on the specific type of dataset, as we will see.

New dataset creation

The dataset graphical editor is divided into two areas: the left one shows the list of all available datasets and the right one shows three tabs, each one corresponding to a specific type of editing operation on dataset.

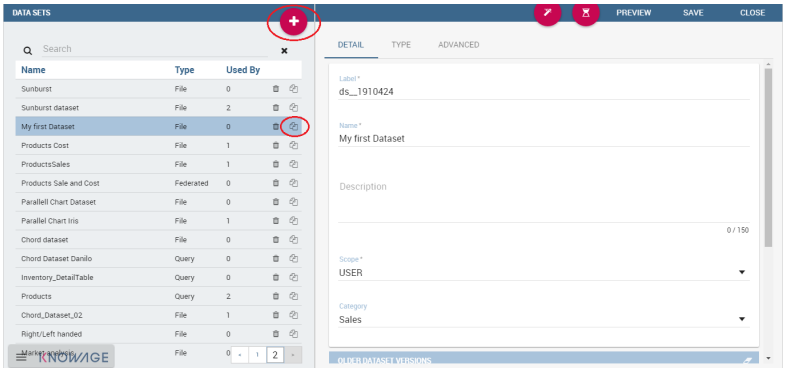


Figure 3.5: Dataset Panel.

Each item of the list in the left panel shows the dataset label (i.e., the dataset unique identifier), name and type, as well as the number of documents currently using it. To create a new dataset, click the **Add** icon  . If your dataset is similar to another existing dataset, you can click

the **`Clone`** icon . This will create a copy of the dataset, except for the label that must edit once again. All fields are pre-filled with values from the existing dataset but they can be modified and saved without affecting the original dataset.

To remove an existing dataset, click the small dustbin icon on the corresponding row of the dataset list.

Once you have clicked the **`Add`** button, you can fill in the dataset definition form. Each tab in the right panel corresponds to a step of the dataset definition process.

In the **`Detail`** tab you define the **Name**, the **Label** and an optional **Description** of the dataset (refer to Figure 3.5). In the lower part you can see a versioning system for the dataset, as shown in Figure 3.6. Each time that you edit and save a dataset, the older version will be archived and will remain accessible from the from the lower part of the detail panel.

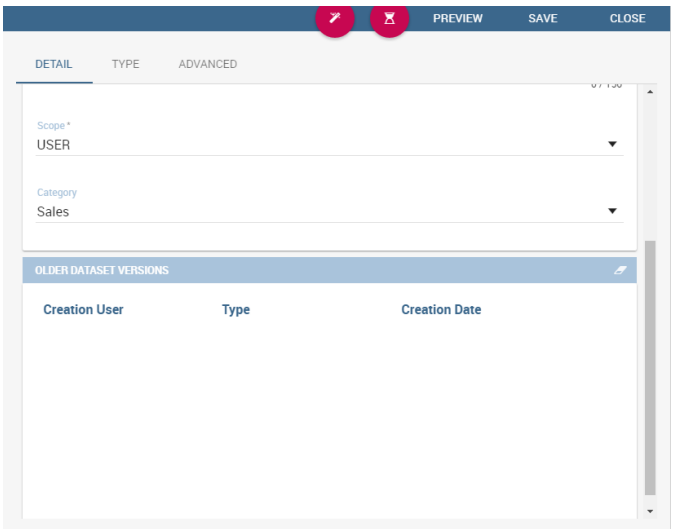


Figure 3.6: The dataset versioning.

In the **Scope**tab you can choose between the three options listed on the Table 3.2. Their combination allows the definition of fine-grained purpose datasets.

Dataset	Private	Public
User	Created from file (CSV, XLS) or from QbE (My Data) for personal use only.	Dataset created from file (CSV, XLS) or from QbE (My Data) and shared with other users.
Technical	Not applicable.	Dataset created by a BI developer to be used in one or more documents. Not visible to end users.
Enterprise	Not applicable.	Dataset of any type created by a technical user and certified by a trusted entity within the organization, and made available to all end users for reuse.

Table 3.2: Scope options.

You can also specify the **Category** of the dataset. This field is not mandatory but it can be used to categorize datasets in your BI project, so that you can easily recover them when performing searches.

In the **`Type`** tab you can define the type of dataset:you can write the code or upload an XLS file or call for a web service accordingly to the dataset type and add parameters to it, if any. You can see an example on the Figure 3.7.

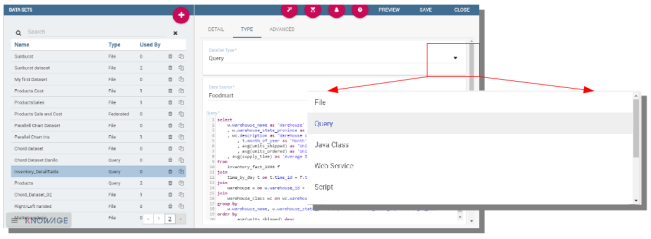


Figure 3.7: The dataset type definition.

In the **`Advanced`** tab, shown in Figure 3.8, you can apply the pivoting transformation to the dataset results if needed or decide to persist the dataset.

Once all those settings have been performed you can see a preview of the dataset results clicking on the **`Preview`** button available on the top right corner of the page. It is recommended to check preview to detect possible errors in the dataset code before associating it to a document.

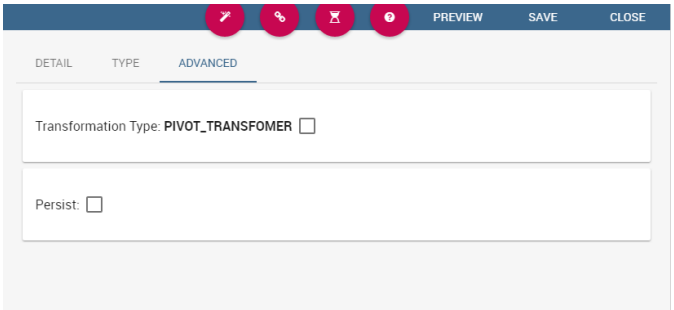


Figure 3.8: The dataset trasformation tab.

Note that the metadata can be manage by clicking on the icon  and use the same criterion described in Section 2.4. Otherwise use the icon  to save without associating any metadata.

Let us describe more deeply each type of dataset.

File Dataset

A dataset of type **File**, see Figure 3.9, reads data from an XLS or CSV file. To define a **`File Dataset`** select the **File** type, then upload the file by browsing in your personal folders and set the proper options for parsing it.

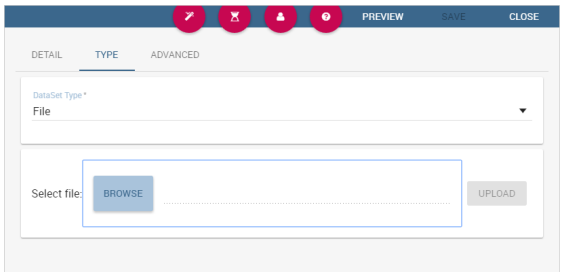


Figure 3.9: File Dataset.

Once you have uploaded the file, you can check and define the metadata (`measure` or `attribute`) of each column.

Query Dataset

Selecting the query option requires the BI developer to write an SQL statement to retrieve data.

The SQL dialect depends on the chosen data source. The SQL text must be written in the **Query** text area. Code 3.1 shows an example.

```
SELECT p.media_type as MEDIA, sum(s.store_sales) as SALES
FROM sales_fact_1998 s
JOIN  promotion p on s.promotion_id=p.promotion_id
GROUP BY p.media_type
```

Code 3.1: SQL query example.

It is also possible to dynamically change the original text of the query at runtime. This can be done by defining a script (Groovy or Javascript) and associating it to the query. Click on the **`Edit Script`** button (see Figure 3.10) and the script editor will open.

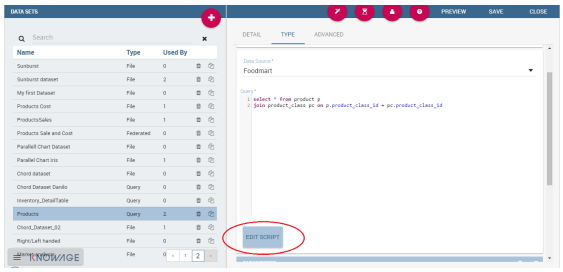


Figure 3.10: Script editing for dataset.

Here you can write the script. The base query is bounded to the execution context of the script (variable query) together with its parameters (variable parameters) and all the profile attributes of the user that executes the dataset (variable attributes).

In Code 3.2 we uses Javascript to dynamically modify the `FROM` clause of the original query according to the value of the parameter year selected at runtime by the user.

```
 if( parameters.get('year') == 1997 ) {
    query = query.replace(FROM sales_fact_1998,
    FROM sales_fact_1997);
}
else {
    query = query; // do nothing
}
```

Code 3.2: Query dataset's script example.

Java Class Dataset

Selecting a dataset of **`Java Class`** type allows the execution of complex data elaboration implemented by a Java class. The compiled class must be available at `\webapps\Knowage\WEB-INF\` classes with the proper package. The class defined by the developer must implement the interface `it.eng.spagobi.tools.dataset.bo.IJavaClassDataSet` and the methods implemented are:

- `public String getValues(Map profile, Map parameters);`
  This method provides the result set of the dataset using profile attributes and parameters. The String to return must be the XML result set representation of type:

```
<ROWS>
  <ROW value="value1" .../>
  <ROW value="value2" .../>
  ...
</ROWS>
```

- `public List getNamesOfProfileAttributeRequired();`
  This method provides the names of profile attributes used by this dataset implementation class. This is a utility method, used during dataset execution.

Script

If you select this option, the results of the dataset will be produced by a script. Therefore, the developer should write a script returning an XML string containing a list of values with the syntax shown below.

```
<ROWS>
  <ROW value="value1" .../>
  <ROW value="value2" .../>
  ...
</ROWS>
```

If the script returns a single value, this will be automatically encoded in the XML format above. The script must be written using Groovy or Javascript language. Knowage already provides some Groovy and Javascript functions returning the value of a single or multi-value profile attribute. These functions are explained in the information window that can be opened from the **`Dataset Type`** tab. New custom functions can be added in `predefinedGroovyScript.groovy` and `predefinedJavascript.js` files contained in the `KnowageUtils.jar` file.

QbE

This is available only in KnowageBD and KnowageSI.

The QbE dataset type option allows the definition of dataset results based on a query defined over a metamodel. To define a QbE dataset you need to select the **Data Source** and **Datamart** that you want to use. Once chosen your datamart you can click the lookup button of the **Open QbE** field and a pop up window will appear showing a QbE interface where you can define your query. Once saved, you can check the generated query thanks to the **View QbE Query**. All these features are exhibited in Figure 3.11.

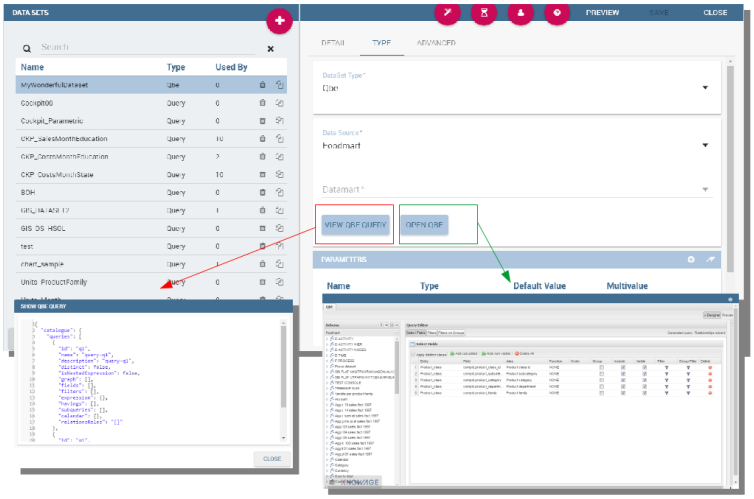


Figure 3.11: QbE Dataset.

Custom Dataset

Selecting a Custom dataset type allows the developer to execute complex data elaboration by a custom Java dataset implementation. There are two options:

- implement the `it.eng.spagobi.tools.dataset.bo.IDataSet` interface;
- extend the `it.eng.spagobi.tools.dataset.bo.AbstractCustomDataSet` class.

The methods executing the dataset that must be implemented are:

- `void loadData();`
- `void loadData(int offset, int fetchSize, int maxResults);`

Using the `AbstractCustomDataset` class allows the developer to access predefined utility methods, such as:

- `public void setParamsMap(Map paramsMap);`
- `public IDataSetTableDescriptor createTemporaryTable (String tableName, Connection connection);`
- `public IDataStore decode(IDataStore datastore);`
- `private void substituteCodeWithDescriptions(IDataStore datastore, Map<String, List<String> > codes, Map<String, List<String> > descriptions);`
- `private Map<String, List<String> > getCodes(IDataStore datastore).`

The full class name (package included) must be set on the Java class name field, while it is possible to add custom attributes for dataset execution and retrieve them via the following method of the IDataSet interface: `Map getProperties()`.

Flat Dataset

A flat dataset allows the retrieval of an entire table from a data source. In other words, it replaces a dummy query like "`select * from sales`" by automatically retrieving all rows in a table. To create a flat dataset, simply enter the table and the data source name, as shown in Figure 3.12.

Ckan

This is available only in KnowageBD and KnowageSI.

A Ckan dataset let you use open data as resource. You have to fill all the settings fields properly to let the dataset work successfully. Let's have a look on them:

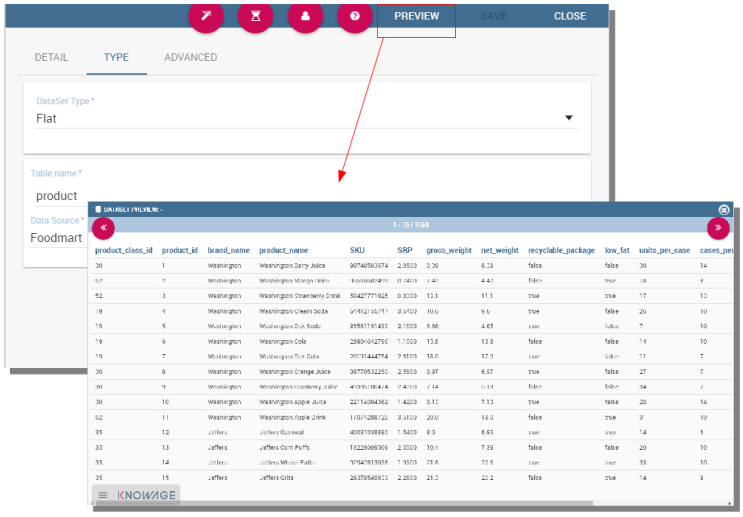


Figure 3.12: Flat Dataset.

- **`File Type`***: this field specifies the type of the file you want to import. Allowed ones are: CSV or XML;
- **`Delimiter Character`***: Here you have to insert the delimiter used in the file. Allowe values are: , ; \t |
- **`Quote Character`***: Allowed values for this field are: “ or ”;
- **`Encoding`**: Here you have to specify the encoding typology used. Allowed values are: UTF-8, UTF-16, windows-1252 , ASCII or ISO-8859-1;
- **`Skip rows`**: the number inserted stands for the rows not to be imported;
- **`Limit rows`**: it is the maximum number of rows to be imported. If you leave it blank all rows are uploaded;
- **`XLS numbers`**: it is the number of scheets to be imported;
- **`CKAN ID`***: here you have to insert the ID of the resource you are interested in. Look for it among the additional information in Ckan dataset webpage.
- **`CKAN url`***: it is the direct link to download the resources available on Ckan dataset webpage.

We marked with the * symbol the mandatory fields. We suggest to do a preview of your dataset before saving it to be sure everything have been correctly configured.

Federated

This is available only in KnowageBD and KnowageSI.

In this area you can only manage metadata, visibility and perform the advanced operation we are going to describe at the end of this section.

Instead, the creation of **`Federated`** done can be accessed from **`My data`** BI functionality under **`Federatation Definitions`**.

Rest

The REST dataset enables Knowage to retrieve data from external REST services. The developer of the dataset is free to define the body, method, headers and parameters of the request; then he has to specify how to read data from the service response using JSON Path expressions (at the moment no other ways to read data is available, therefore the REST service is presumed to return data in JSON format).

Let's make as example in order to understand how it works. Suppose an external REST service providing data from sensors, we want to retrieve values from prosumers electricity meters, a prosumer being a producer/consumer of electricity, and that the request body should be something like:

```
{   "entities": [
        {
            "isPattern": "true",
            "id": ".*",
            "type":"Meter"
        }       ]
 }
```

Code 3.3: Request body code.

querying for "Meter" entities, and that the JSON response is something like:

```
{
  "contextResponses": [
    {
      "contextElement": {
        "id": "pros6_Meter",
        "type": "Meter",
        "isPattern": "false",
        "attributes": [
          {
            "name": "atTime",
```

```
11            "type": "timestamp",
12            "value": "2015-07-21T14:49:46.968+0200"
13          },
14          {
15            "name": "downstreamActivePower",
16            "type": "double",
17            "value": "3.8"
18          },
19          {
20            "name": "prosumerId",
21            "type": "string",
22            "value": "pros3"
23          },
24          {
25            "name": "unitOfMeasurement",
26            "type": "string",
27            "value": "kW"
28          },
29          {
30            "name": "upstreamActivePower",
31            "type": "double",
32            "value": "3.97"
33          }
34        ]
35      },
36      "statusCode": {
37        "reasonPhrase": "OK",
38        "code": "200"
39      }
40    },
41    {
42      "contextElement": {
43        "id": "pros5_Meter",
44        "type": "Meter",
45        "isPattern": "false",
46        "attributes": [
47          {
48            "name": "atTime",
49            "type": "timestamp",
50            "value": "2015-08-09T20:29:45.698+0200"
51          },
52          {
53            "name": "downstreamActivePower",
```

```
            "type": "double",
            "value": "1.8"
          },
          {
            "name": "prosumerId",
            "type": "string",
            "value": "pros5"
          },
          {
            "name": "unitOfMeasurement",
            "type": "string",
            "value": "kW"
          },
          {
            "name": "upstreamActivePower",
            "type": "double",
            "value": "0"
          }
        ]
      },
      "statusCode": {
        "reasonPhrase": "OK",
        "code": "200"
      }
    }
  ]
}
```

Code 3.4: JSON response code.

In this example we have two **`Context Elements`** with the following attributes:

- **`atTime`** ;
- **`downstreamActivePower`**;
- **`prosumerId`**;
- **`unitOfMeasurement`**;
- **`upstreamActivePower`**.

Let's see how to define a Knowage dataset:

We specified

- the URL of the REST service;

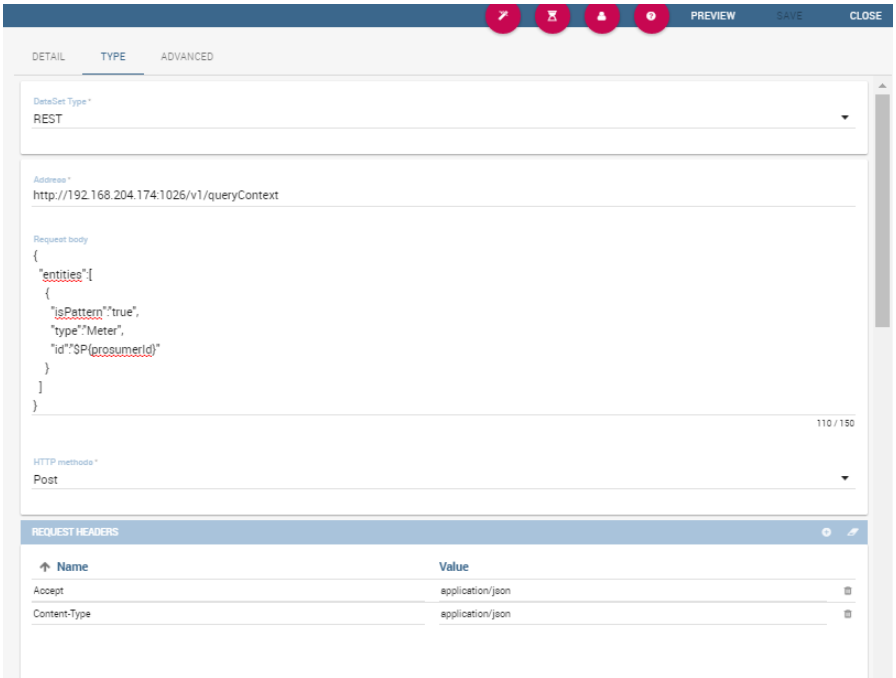


Figure 3.13: REST dataset interface.

- the request body;
- the request headers (in this example we ask the service for JSON data);
- the HTTP method;
- the JSONPath to retrieve the items (see below), i.e. the JSONPath where the items are stored;
- the JSONPaths to retrieve the attributes (see below), i.e. the JSONPaths useful to retrieve the attributes of the items we are looking for; those paths are relative to the "JSON Path items";
- offset, fetch size and max results parameters, in case the REST service has pagination.

Once followed the steps above the user obtains upstream/downstream active power for each prosumer.

`NGSI checkbox` is specific for NGSI REST calls: it permits easy the job when querying the Orion Context Broker (`https://github.com/telefonicaid/fiware-orion`) and to omit some of the REST fields (since the JSON format from NGSI s pecifications is fixed): you don't need to specify headers, JSONPath items, JSONPath attributes (all available attributes are fetched) and pagination parameters (offset and fetch size).

When checking the **`Use directly JSON attributes`** checkbox, yon can skip the definition of

the JSONPath attributes, since the JSON structure is presumed to be fixed as in the following example:

```
{
  "contextResponses": [
    {
      "prosumerId":"pros1",
      "downstreamActivePower":3.1,
      "upstreamActivePower":0.0
    },{
      "prosumerId":"pros2",
      "downstreamActivePower":0.5,
      "upstreamActivePower":2.4
     }
   ]
}
```

Code 3.5: Use directly JSON attributes.

then it will be enough to define only the **`JSON Path Items`** and check **`Use directly JSON Attributes`** without defining the attributes; the attributes will be retrieved automatically from the JSON object.

In the above examples, the JSON Path Items will be: `$.contextResponses[*]`

and the dataset result will look like:

prosumerId	downstreamActivePower	upstreamActivePower
pros1	3.1	0.0
pros2	0.5	2.4

Table 3.3: Dataset result

The REST dataset permits usage of profile attributes and parameters using the same syntax as for other dataset types: `$<profile attribute>` and `$P<parameter>`. You can use both of them as placeholders in every field: most likely you need to use them in REST service URL or on the request body. As an example, suppose you want to retrieve the value of just one prosumer that is specified by the "prosumerId" parameter, you have to set the request body as:

```
{
  "entities":[
    {
      "isPattern":"true",
```

```
      "type":"Meter",
      "id":"$P{prosumerId}"
    }
  ]
}
```

Code 3.6: Request body for prosumerId parameter.

Big Data

KnowageBD provides the possibility to define Big Data dataset as well as Big Data datsources. To set these kind of datasets the user just have to select the **`Query`** type and insert the code according to the dialect in use (that is accordingly to the datasource dialect).

For example, let's suppose we defined a Mongo datasource and want to create a dataset upon it. Therefore choose the "'Query type' dataset and, as we revealed in advance, choose the correct language: in this case JS instead of SQL. The script must respect some convention, in particular:

- the return value of the query must be assigned to a variable with name "query". For example

  `var query = db.store.find();`

- if the return value doesn't come from a query, for example it's a js variable, than it must be assigned to a variable with name **`sbiDatasetfixedResult`**. The result will be managed by Knowage accordingly to the type of the variable:

  - if it's a primitive type the resulting dataset contains only a columns with name "result" and value equal to the value of the variable sbiDatasetfixedResult;

  - if it's an object, the resulting dataset contains a column for each property of the object.

    For example, if we consider the query

    `sbiDatasetfixedResult = {a:2, b:3}`,

    the dataset is as shown in Table 3.4

a	b
2	3

Table 3.4: Dataset output.

  - if it's a list than the columns of the dataset are the union of the properties of all the objects contained in the list.

For istance, let's consider the query

```
sbiDatasetfixedResult = [{a:2, b:3},{a:2, c:3}]
```

the dataset is

a	b	c
2	3	
2		3

Table 3.5: Dataset output.

The result of a query in MongoDB can assume different shapes: Cursor, Document, List, fix value. Knowage can manage automatically the result of the query. The algorithm to understand how to manage the result is very simple.

- If in the query it finds the variable sbiDatasetfixedResult the result will be managed as described above.
- If in the query it finds a findOne the result will be managed as a single document.
- If in the query it finds an aggregate the result will be managed as an aggregation.
- In the ether cases the result will be managed as a Cursor.

It's possible to force the behaviour. In particular the result stored in the variable query, will be managed:

- as cursor if in the script exist a variable with value `LIST_DOCUMENTS_QUERY`.

  Example: `var retVal= "LIST_DOCUMENTS_QUERY“;`

- a document if in the script exist a variable with value `SINGLE_DOCUMENT_QUERY`.

  Example: `var retVal= "SINGLE_DOCUMENT_QUERY”.`

Similar techniques can be applied to the other languages. We leave the reader to examine the dialect related to each Big Data datasource.

3.4 Business Model

By clicking on **`Resources`** >**`Catalogs`** >**`Business Model catalog`** you can manage your Business Models.

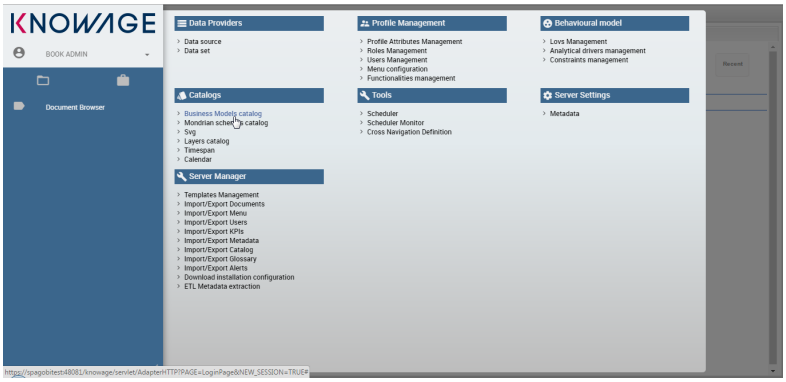


Figure 3.14: Business Model Catalog menu item on Server.

The page is divided in two areas. The left one contains the list of the existing Business Models. Here you can create a new model by clicking on the button **Add** located on top of the business models' catalog or delete an existing model by clicking on the dedicated button at the end of each row in the list. In the right side you find the details of the catalog selected from the list on the left. Here you can modify and save the details of the selected model. In addition the templates versioning is provided at the bottom side of this area.

Metamodel creation

In this section we go into details of how to build your own metamodel. Knowage allows to develop and manage metadata through the use of a web interface that is called **Meta Web**. We recall that dealing with metadata means to manage data that describe the structure of a relational model, namely to deal with the relationship between tables, table columns, keys and so on.

The Meta Web allows the user to access these information through the usage of a graphic interface and to easily combine, redefine and query them on an abstract model, guaranteeing the safety of the source model. In addition, we stress that the users can perform queries over data without the usage of a data query language.

Using the Meta Web application, it is possible to reverse the content of a database and manipulate this information creating a new model that can fit the user's needs. In this section will we see what are the steps need in order to create a metamodel and query it with the QBE.

To create a Metamodel enter the **Business Model Catalogue** and add a new model clicking on the "Plus" icon. Referring to Figure 3.15, you will be prompted to enter the following fields:

- Name (mandatory): Name of the model (cannot be changed after the save).
- Description: A longer description of your model.

- Category (mandatory): Select, from the ones avaiables, a category that the model belongs to.
- Data Source (mandatory): select the datasource that will be used to create your model (so the one that contains the tables that you need).

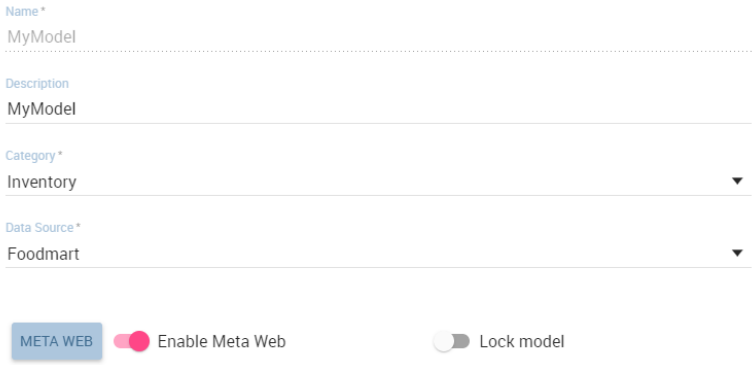


Figure 3.15: Setting the metamodel basic information.

After you have compiled these information, click on "Save" on the top right corner of the screen. Now click on the switch **`Enable Meta Web`** that will show up a button **`Meta Web`**, click on that and you are ready to design the model.

Create an empty model

The first time you enter the Meta Web, the interface (see Figure 3.16 will show you the available tables extracted from the selected data source.

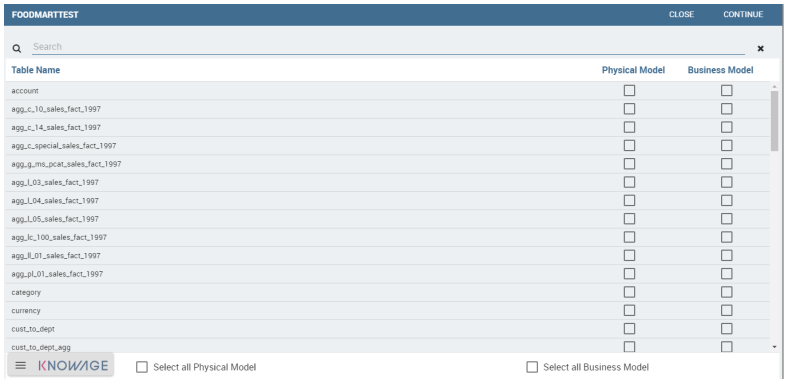


Figure 3.16: Metaweb interface.

For each table you can decide if you want to include it in your metamodel. More in detail a metamodel is divided in two model:

- **`Physical Model`**: it represents a "snapshot" of the database at the moment of the creation of you metamodel. The physical model contains a list of tables and information like columns and foreign keys retrieved from the database. The Physical Model cannot be modified but could be updated to reflect changes made on the database after the creation.
- **`Business Model`**: it is based on the physical model but let the user recombine some of his informations. For example is possible to create a Business Class that contains only some of the columns of a Physical Table and create new relationships between Business Classes that are not defined on the physical database.

If you choose to include a table only in the physical model is always possible to create a corresponding business class later during the editing. When you have finished to select the tables you can proceed to the editing clicking on the **`Continue`** button.

Editing the metamodel

The Meta Web Editor is divided in two main tabs "Business Model" and "Physical Model" corresponding to the related models. Clicking on one of this tab will change the view showing the elements of the specific model.

On the right portion of the interface you will see a tree like structure with the list of tables imported in the Physical Model (see Figure 3.17.

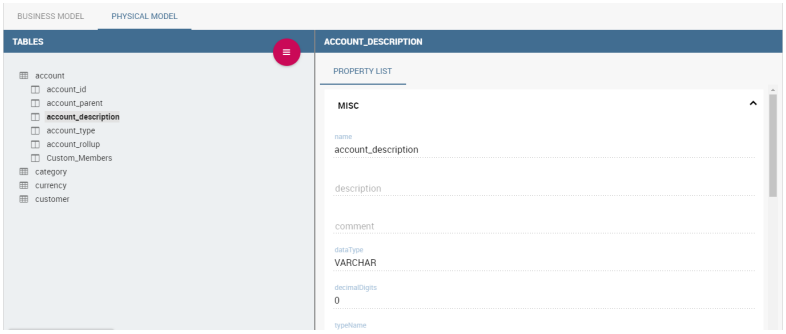


Figure 3.17: Physical Model Tab.

Clicking on the icon on the left of each Physical Table is possible to expand the corresponding node showing the columns contained. Clicking on each column name is possible to see on the right portion of the screen a list of properties like data type and length. Clicking on a table name is possible some generic informations and also the associated foreign keys defined The red circle with the sandwitch icon opens a cascading menu that will let you update the physical model connecting on the database and retrieve new information about the database structure.

Then the user can switch to the Business Model tab, shown in Figure 3.18.

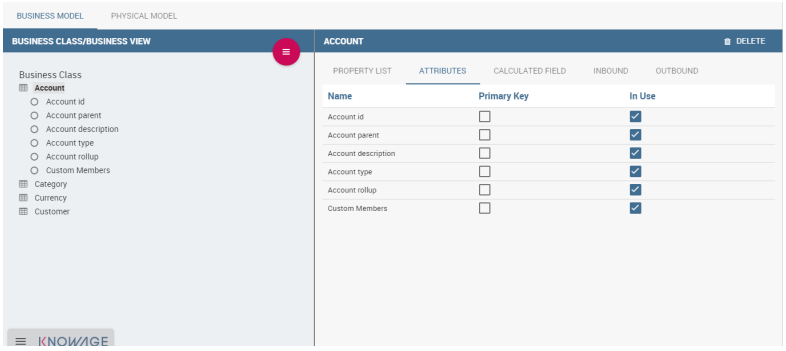


Figure 3.18: Physical Model Tab.

In this view you will see all the Business Class created at the first initialization and the ones added later. Also here the left part has the Business Classes represented in a tree structure, clicking on each name will show generic informations, the attributes (corresponding to columns), calculated fields and inbound and outbound relationships. In this view the sandwitch icon will let the user add other Business Classes (from the tables of the Physical Model) and Business View (a combination of more tables with a predefined join path).

Create a new business class

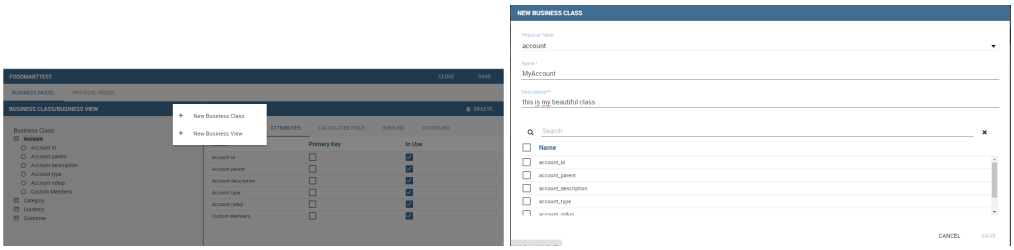


Figure 3.19: Create a new business class.

If you want to add some new business class after the first initialization is possible to do this using the sandwich icon in the Business Class tab (see Figure 3.19). Clicking on that will open a new dialog where is asked to the users to:

- select a Physical Table from the avaiable ones;
- insert a description for this new business class;
- select one or more columns.

Then click on save to add the business class.

Create a new relationship

In the Business Model is possible to define new relationships between Business Classes that are not inherited from the physical foreign keys. The Business Relationships are divided in two types:

- **`Inbound`**: relationships that have the selected Business Class as a target (so they are entering);
- **`Outbound`**: relationships that have the selected Business Class as a source (so the starts from).

To create a new relationship, just select the tab "Inbound" or "Outbound" after selecting one Business Class. Then click on the button "Add" and you will see a dialog.

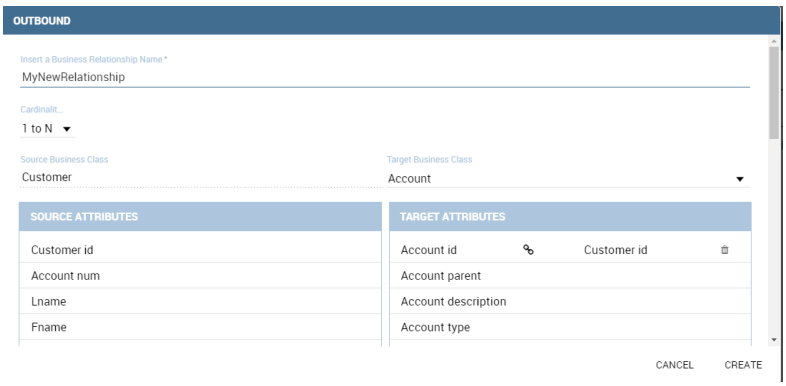


Figure 3.20: Setting the outbound relationship.

In Figure 3.20 the outbound relationship is shown. Here you have to :

- enter the business relationship name,
- select the cardinality of the relationship (1 to N is suggested),
- select the Source and Target Business Classes,
- Then is possible to drag and drop a Business attribute from the source Business Class to another Business attribute in the target Business Class. This will create a link between the two attributes.

When all these steps are accomplished, click on "Create".

Generate the datamart

After the editing of the metamodel click on "Save" on the Meta Web toolbar on the upper right corner. Now you have a metamodel that can be compiled and used to generate a datamart.

Now if you go back to the Business Model catalog you will see that near the "Meta Web" button there is a "Generate" button. Clicking on it, a dialog will open (see Figure 3.21).

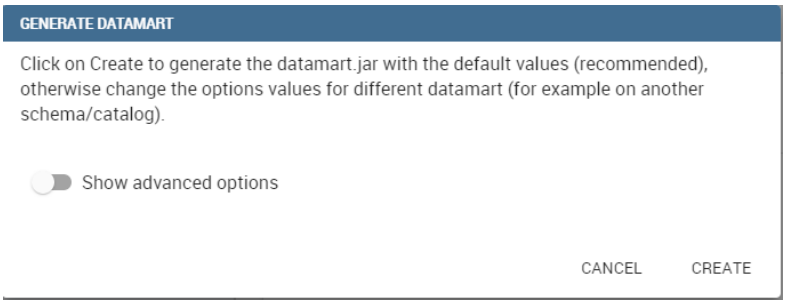


Figure 3.21: Generate datamart dialog.

If you just press "Create" the generation of the datamart begins otherwise clicking on the switch "Show Advacend options" is possible to change some details like schema/catalogue of the database used to query the metamodel. When the datamart is generated it will be possible to query the metamodel accessing it in the Workspace interface.

3.5 User and roles

Knowage users are defined by:

- identities,
- roles,
- profiles.

The *identity* of a user consists of all data used to identify that user, i.e., a username and a password, as well as a human readable full name.

The *profile* of a user consists of a set of properties called attributes, describing general information about the user, e.g., age and gender, but also domain-specific properties, such as the organizational unit to which he belongs. Some attributes, such as name and email, are defined by default in Knowage. Others can be added by the model administrator, as explained in the following sections.

The *role* of a user represents a categorization of a group of users. These roles may correspond to specific positions in the company, e.g., "general manager" or a "sales director", or to a position with respect to the BI project, e.g., "data administrator" and "BI developer". Different users may have the same role, as well as the same user may have multiple roles.

Knowage allows you to create several roles, according to your project needs. However, all roles must belong to a specific *role type*. A role type is a higher-level categorization used by Knowage, in order to map roles for the different features of the suite.

Role Type	Description	Standard User
ADMIN	General administrator. Manages all Knowage functionalities.	biadmin
MODEL_ADMIN	Model administrator. Manages the Behavioural Model and its associated functionalities.	bimodel
DEV_ROLE	Developer. Creates and modifies datasets and documents.	bidev
TEST_ROLE	Test user. Tests analytical documents.	bitest
USER	End user. Executes documents visible to him and creates ad-hoc reporting analysis.	biuser

Table 3.6: Knowage Role Types.

Authentication can be handled internally by Knowage or delegated to an external Single Sign-On (SSO) system.

Authentication Management

The choice of handling authentication internally or delegating it to an external SSO system typically depends on the presence of an authentication system already in place. If this is the case, Knowage can seamlessly integrate with the existing authentication infrastructure.

Once the user has logged in, his role is loaded. Roles are managed internally. In case the user is associated with multiple roles, he will be asked to choose one.

Alternatively, by clicking on the icon shown in Figure 3.22, he can select a default role that will be kept valid until he logs out.

The steps to create a user model follow:

- Create profile attributes;
- Create roles;
- Create users and associate attribute values and roles to them.

Knowage supports the management of user profiles and roles through the **`Profile Management`**

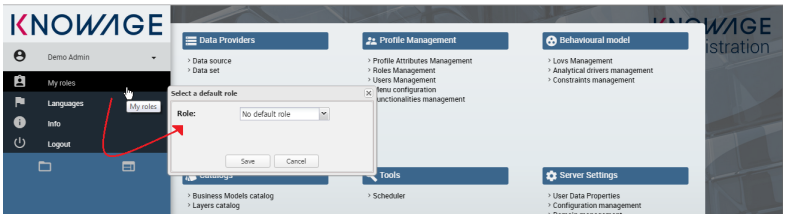


Figure 3.22: User roles in Knowage.

menu section. This menu is only visible to Knowage administrator and to the model administrator, since users and roles management is a critical operation that requires an appropriate level of responsibility.

The **`Profile Management`** menu section contains three sub-menu items:

- **Profile Attribute Management**: to define new profile attributes and manage the existing ones.
- **Role Management**: to create new roles and manage permissions for each role.
- **User Management**: to create users, manage their identities, assign values to their profile attributes and associate them with roles.

In the following, we show how the model administrator can define user profiles and roles using these functionalities. Remember that Knowage profile management can also be integrated with external profiling systems.

Clicking on **`Profile Attribute Management`**, the list of currently defined attributes is shown. To add a new attribute, click the **`Add button`**, as shown in Figure 3.23: a new row is added to the list, where you can insert the name and description of the new attribute. To delete an attribute, select the corresponding row and click **`Delete`**.

Attributes defined in this section will be available to all user profiles. It is not mandatory to assign a value to each attribute for each user, since profile attributes without values will not be considered in the definition of the user profile.

Figure 3.23: Profile attributes Management.

Once the attributes are defined, the model administrator can define roles, using the **`Role Management`** functionality. The role management tool is two-sided: on the left you can see the list of already defined roles. At the beginning of a project, only default roles are visible. To add a new role, click the **`Add`** button and move to the right panel. To delete a role, simply click the **`Delete`** button at the end of the role's row.

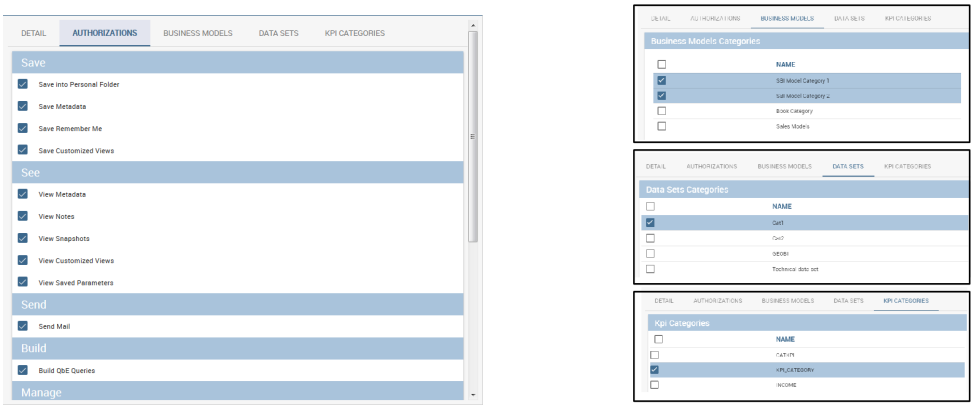


Figure 3.24: Roles Management.

Role Management

The behavioural model should be built taking into account the specificity of each organization and the needs of the BI project. Therefore, it is a good practice to define specific roles for the BI project and avoid using Knowage technical roles only.

In the right panel there are three tabs. The **`Detail`** tab allows the administrator to define role name and role type (mandatory). The role type regulates the visibility of that role based on the categorization described in Table 3.6. A code and a description can be added too, as shown in Figure 3.24.

The **`Authorizations`** tab allows you to assign permissions to each role. Rights are predefined and grouped into categories, as shown in Figure 3.24.

The **`Business Models`**, **`Data sets`** and **`KPI Categories`** tabs are is intended to assign specific categories to each role, in a way that each user can only see the business models, datasets or KPI that belong to the categories associated with his role.

The **`Business Models`** tab is available only for KnowageBD and Knowage SI, while the **`KPI Categories`** one is available only for KnowagePM. More details on business models and KPIs can be found in the corresponding chapters.

You can create new categories for business models and datasets using the **`Server Settings`** > **`Domain management`** menu item.

Last but not least, the **`User Management`** section includes a left panel that allows the administrator create and delete users, and a right panel that allows him to manage user details, roles and attributes (see Figure 3.25).

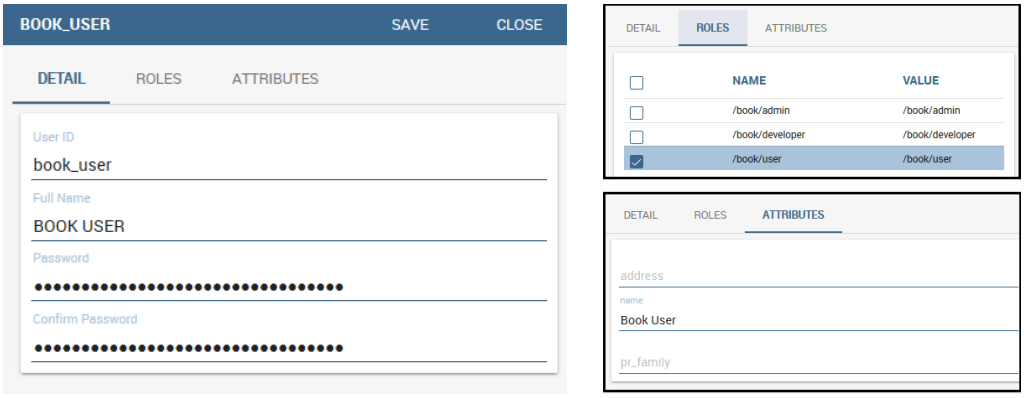


Figure 3.25: Users Management.

3.6 Behavioural Model

An analytical driver (hereafter simply driver) models a concept or a piece of data frequently used as a distinguishing criterion on the global data context. A driver highlights the concepts guiding the analysis, providing a unique representation of them and describing how they are shown and checked according to the end users' roles. When connected to analytical documents, a driver produces an explicit or implicit parameter used to filter data.

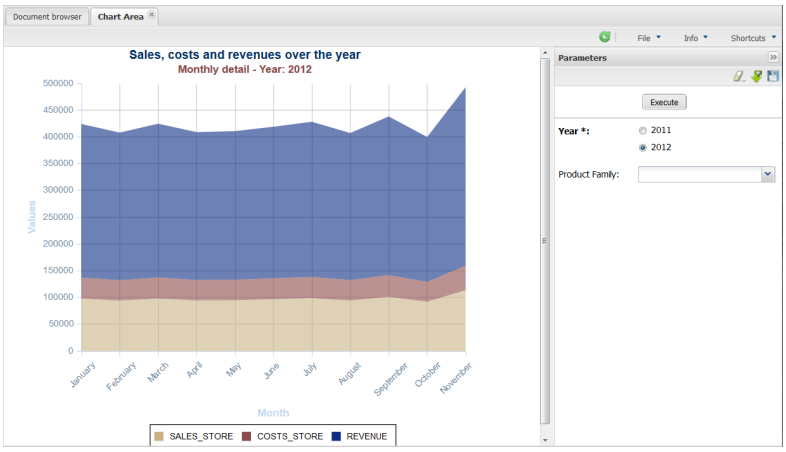


Figure 3.26: Parametric Report.

The Figure 3.26 represents a report with two parameters:

- the **Deparment**, a mandatory field, displayed as a combo box and with possible values `Alcoholic Beverages`, `Baked Goods`, `Baking Goods` and so on;
- the **Age Range**, a mandatory field, displayed as list of values and with possible values `0-10`, `10-20` and so on.

All these aspects are regulated by the analytical driver behind each parameter. In particular, each driver provides many *use modes*, defining:

Who is involved in a specific use mode, in terms of a list of end user roles, considering that a role can be associated to a single use mode only.

What data he can access and how they are presented to the end user for his potential selection. This information is provided by the so called *List of Value (LOV)*.

How to check the validity of the chosen values. This information is provided by the so called *Check*.

In other terms, each use mode refers to an initial visualization method and content (LOV), to one or more validation rules (check) and to one or more end user roles (roles). The logic of a driver is represented in Figure 3.27.

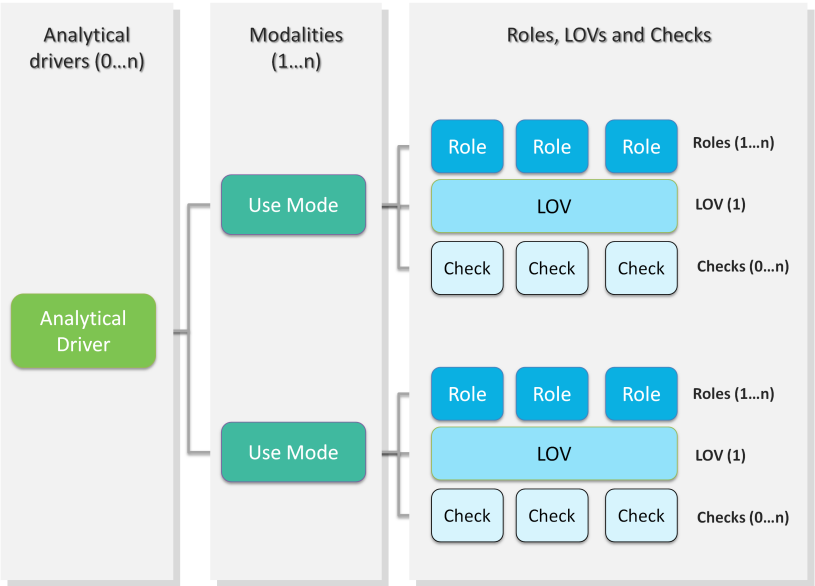


Figure 3.27: Analytical driver schema.

Let's consider the following example. We need to represent the concept of "product family". Since this is a common driver and discriminator for the enterprise analysis, an analytical driver will be coded, with all its behavioural rules, such as:

- if the user is a call center operator or a user that provides internal support, he can manually write the product family he wants to select. This value will be formally verified (it must be a text) and checked on the product family registry.
- if the user is a product brand director or an operative secretary, he can choose the value from a preloaded list of all the product families belonging to his brand. For this reason, the value does not need any check.

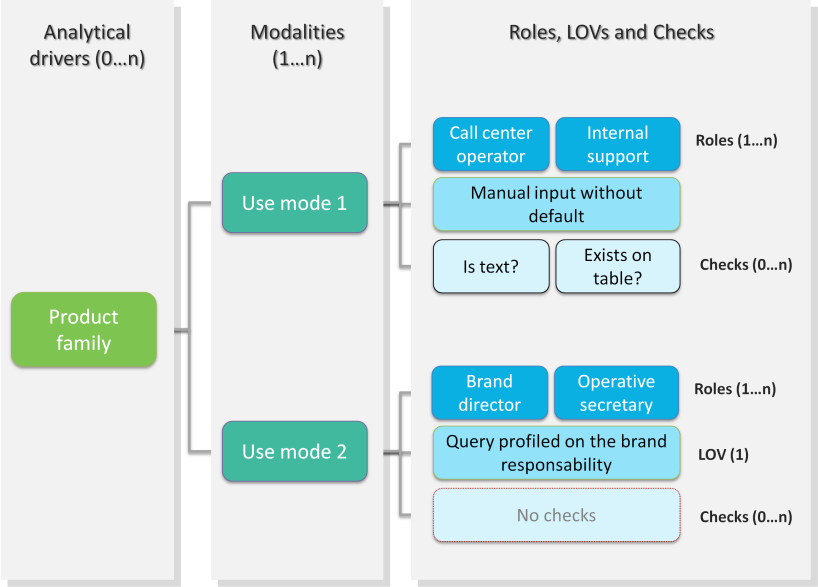


Figure 3.28: Analytical driver schema - Example.

Once defined, a driver can be related to many documents, driving their behaviour and filters in a common way. This way, a user who runs different documents that use the same drivers always receives the same parameter form, applying the same filters over shown data. In fact, when an authenticated user (with its roles and profile) runs an analytical document, its technical metadata are read, mainly in terms of document template and related drivers. Based on them, a customized page for the parameters input is produced, according to the driver logic for the end user role. The selected values are then validated and the final output comes to the user. Figure 3.29 shows this process.

Thanks to analytical drivers, a single document is able to cover the analytical demands of various categories of users, with noticeable advantages in terms of:

- reduction of the number of documents to be developed and maintained,
- consistency in the request for parameters,
- complexity reduction in the development of documents, thanks to the separation between security matters and massive development,
- simple maintenance of the security (visibility over data) over time, despite the increase of developed documents or added engines.

In the next paragraphs we explain how to create a new analytical driver together with its basic components.

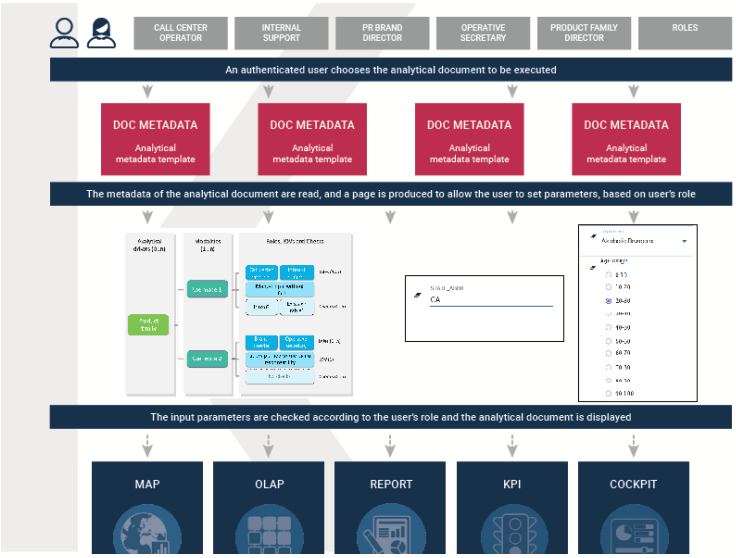


Figure 3.29: Overall process.

Creating a List Of Value

A *List Of Value* (LOV), is a collection of data organized in attribute-value fashion. For example, the LOV in Code 3.7 retrieves id, name and food family for a product.

```
{195, High Top Almonds, Food};
{522, Tell Tale Walnuts, Food};
{844, Very Good Soda, Drink};
```

Code 3.7: LOV example

There may be multiple attributes in a LOV, but only one of them is the core value that is actually used in the analytical driver. Other values have a descriptive function: they can be used to provide a human readable description of the LOV, as well as to store information used, for example, to correlate analytical drivers. In our example, the core value is the customer's id, while the others are additional data describing the customer. Knowage allows to create different types of LOV:

Query: SQL query to retrieve values from the database;

Script: Groovy or JavaScript to dynamically return values;

List of fixed values: Values are defined statically at LOV creation time;

Java objects: External object invoked by name that returns the list of values;

Dataset: Dataset already defined in Knowage Server that is used to retrieve values. Note that the dataset must not contain parameters, while profile attributes are allowed.

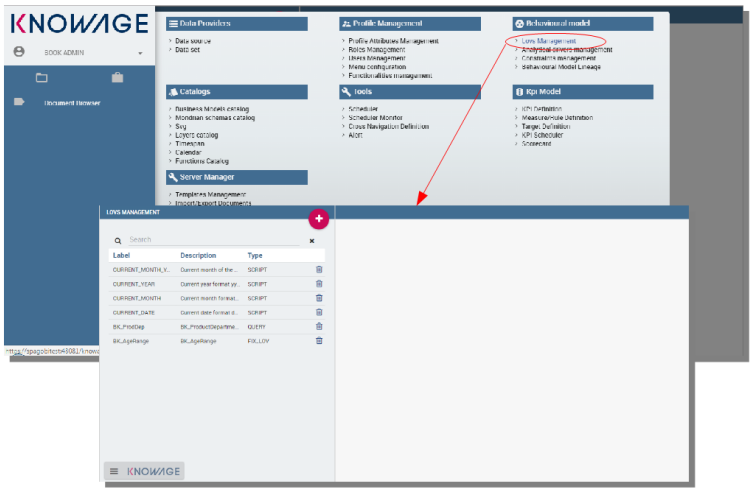


Figure 3.30: LOV list.

To create and manage LOVs, select **`Behavioural Model`** > **`Lovs Management`** from the developer menu. The entire list of available LOVs appears, as shown in Figure 3.30. For each LOV, the list shows the label, description and type; to see the details of a LOV the user must simply select it and they will appear in the right half of the page. On the contrary, to delete one dataset click on the icon available at the end of the row. Notice that you cannot delete a LOV if a driver is currently using it.

To create a new LOV, click on the icon at the top right corner of the page. The LOV creation interface will open, where you can set label, name and description, choose the LOV type and define its values accordingly, as highlighted in Figure 3.31.

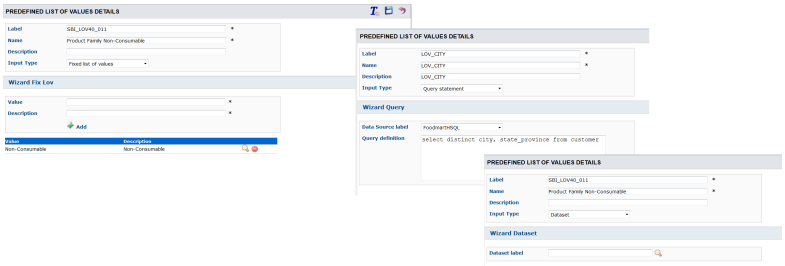


Figure 3.31: LOV Creation interface.

Once completed the form, click on **`Preview`** button to enable the **`Test`** button. Notice that you cannot save the LOV without testing it, since this allows to detect errors before the LOV is actually used in a driver and associated to a document. After testing, you will be able to define

which column is the actual value of the LOV, i.e., which value will be passed to the analytical driver using this LOV. Only *one* column can be the **value** attribute and only *one* column can be chosen as **Descriptive** attribute, while the others can be **visible**. Figure 3.32 and Figure 3.33 exhibit an example. Columns that are not visible can be used for correlating drivers.

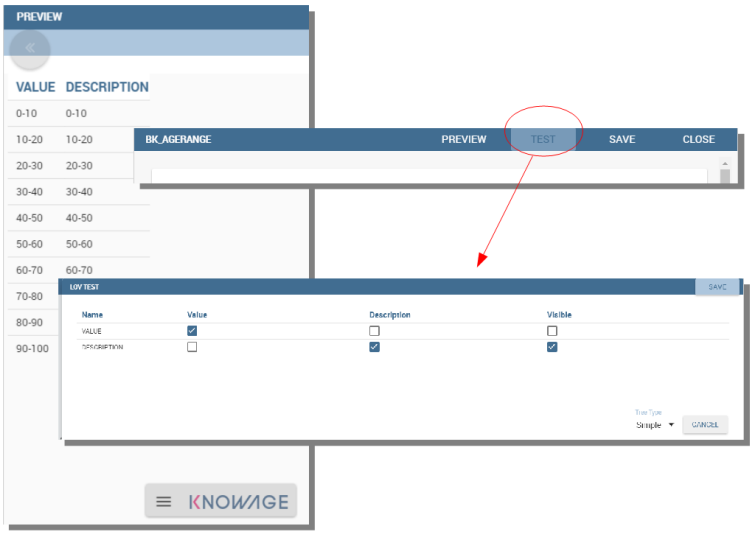


Figure 3.32: Preview and Test of the LOV.

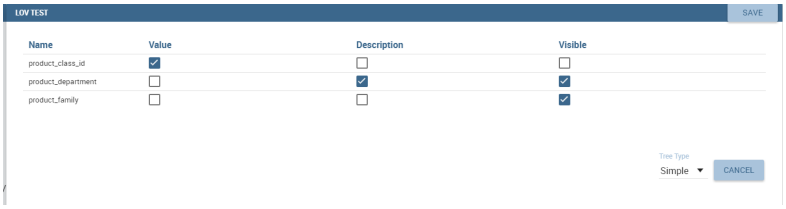


Figure 3.33: Preview and Test of the LOV.

Correlating analytical drivers

Drivers can be correlated so that the value of the first driver is used as a parameter to select values in the second.

We stress that the visibility of specific fields serve to improved human readability when applying filters to documents handled by third users. Moreover it is possible to choose (refer to Figure 3.34) between **`simple`**, **`tree`** and **`tree with selectable internal nodes`** typology of LOV. The last two are hierarchical and let the user visualize the parameters together with their logical tree structure.

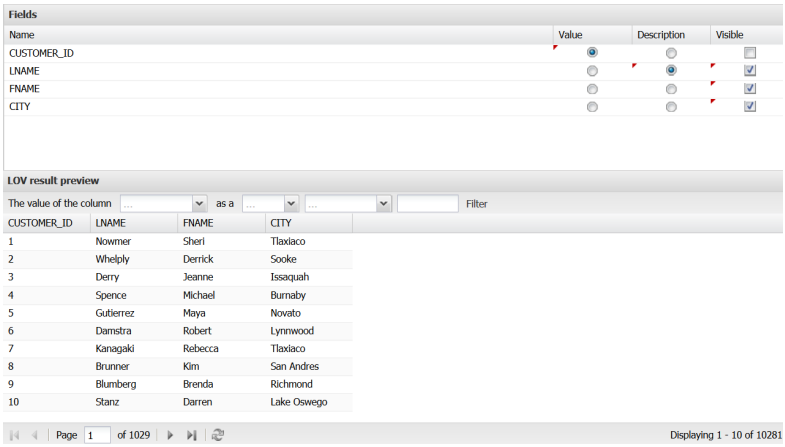


Figure 3.34: Hierarchical LOV definition.

Parametrizing LOVs

Suppose that you need to retrieve a list of values representing all brand names of your products. Then you can use a Query LOV like in code 3.8.

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
FROM PRODUCT
```

Code 3.8: Query LOV example

This is suitable for end users like the general manager who need to see all brands for every product family. Suppose now that another end user is, for example, the food manager. He should not see every brand name, but only those related to the `Food` product family. This could be done using user's profile attributes.

In particular, all query except the *List of fixed values* type can be parameterized using profile attributes. This means that, at LOV execution time, the value of the attribute in the user's profile is assigned to a placeholder in the LOV query/script. Suppose that, in our example, the food manager user has the profile attribute `pr_family` equal to `Food`. You can write this second Query LOV using the placeholder with the standar syntax `${profile_attribute_name}`, as shown in Code 3.9.

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
FROM PRODUCT
WHERE C.PRODUCT_FAMILY = '${pr_family}'
```

Code 3.9: Parametric query

Then, at LOV execution time, for the user food manager the query becomes as shown in Code

3.10 and hence the corresponding LOV will return only the brand names related to the `Food` product family.

```
SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
FROM PRODUCT
WHERE C.PRODUCT_FAMILY = 'Food'
```

Code 3.10: Runtime placeholder substitute

This means that if you are the food manager and your user has the profile attribute `pr_family=Food`, then you will see only the brand related to the food family as a result of this LOV; while if you are the drink manager and your user has consequently the profile attribute `pr_family=Drink`, you will see only the brand related to drink family products.

Note that an information button and a profile attribute button are available (see Figure 3.35) to guide user in writing the code properly, using the syntax correctly and typing the right profile attribute name.

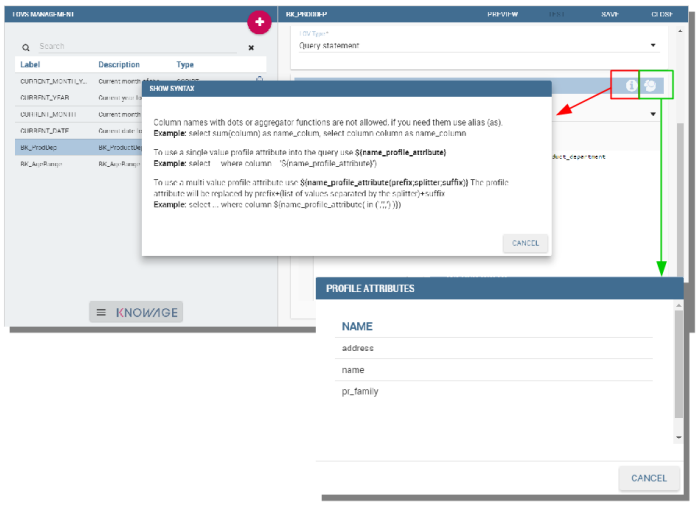


Figure 3.35: Assistance in retrieving syntax and profile attributes.

Creating a validation rule

Knowage supports the validation of the document's input parameters via validation rules, which open the window in Figure 3.36. Validation rules can be defined in **`Behavioural model`** > **`Constraints Management`**. A validation rule checks parameter values as given by LOVs to verify that they comply with the defined constraints.

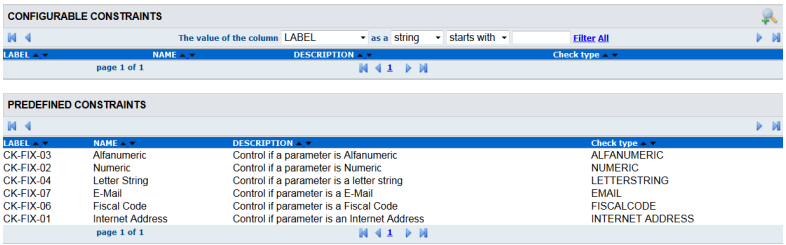


Figure 3.36: Contraints Management.

Knowage default checks are:

Alfanumeric: it checks if the parameter is alfanumeric;

Numeric: it checks if the parameter is numeric;

Letter String: it checks if the parameter is a letter string;

E-Mail: it checks if the parameter is an e-mail;

Fiscal Code: it checks if the parameter has the correct syntax of a fiscal code;

Internet Address: it checks if the parameter is an internet address.

If the administrator needs to create additional validation rules, he can click on to open the rule creation interface. Here he can define a customized validation rule using the available check options:

Date: here you can set a costumized format type of date;

Regular Expression: to set a regular expression validation rule;

Max/Min Length: it lets you set the maximum and/or minimum character parameters length;

Range: to set a range the parameters value has to satisfy;

Decimal: to set a maximal decimal places for the parameters.

Creating an analytical driver

As explained at the beginning of this section, analytical drivers use information about users, their roles and profiles to filter data returned by their associated LOVs. Users, roles and profiles must have been already defined in the project context so that they are available to the driver.

To create a driver, select **Behavioural Model** > **Analytical Drivers Management** from the developer menu. Here, you will see the entire list of available drivers. For each driver, the list shows unique label, description and type. To explore details the user must just select one

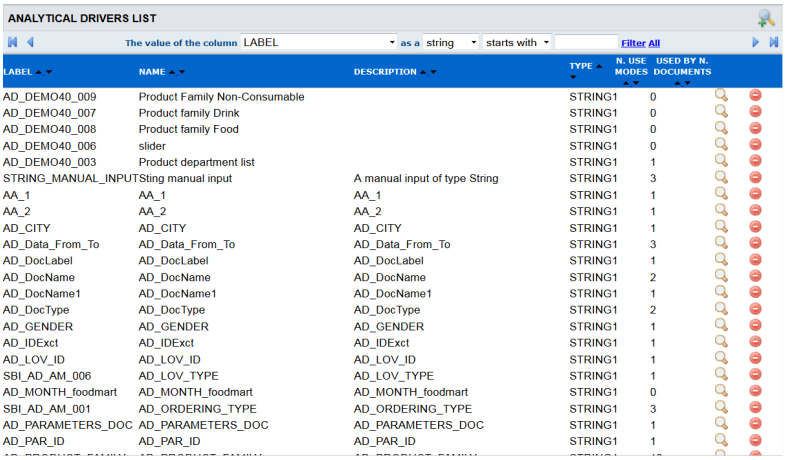


Figure 3.37: Analytical Driver Management.

menu item from the list and they will appear in the half right side, as shown in Figure 3.37. Otherwise to delete one analytical driver the user must use the icon available at the end of each row of the list. Notice that you cannot delete a driver if a document is currently using it.

To create a new driver, click on at the top right corner. The driver creation interface will open. At first execution only the upper part of the window is visible, as shown in Figure 3.38. The upper part is the **`Detail`** section, where you can set the label, name and description. Choose the type between **Date**, **String** or **Number** depending on the type of expected data. Select **Functional** or **Temporal** if the driver is used by an end user or a scheduler, respectively. A click on the save botton, enabled as soon as the form is filled in, will save the driver and let the section below appear.

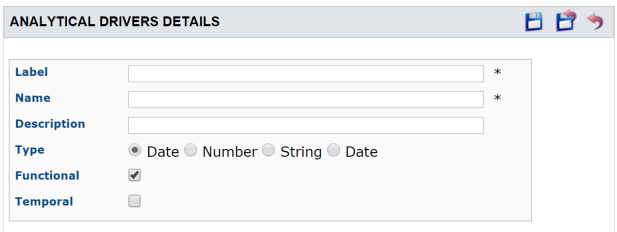


Figure 3.38: Driver creation.

In the **Analytical Driver Use Mode Details** section, one or more LOVs are linked to the current driver, as well as roles and checks are assigned via the so-called *use modes*.

To associate LOVs to the driver, switch to the "Analytical Driver Use Mode Details" tab. Here the user must set label and name of that specific use mode, the kind of input among **`LOV input`**, **`Manual input`** and **`Map input`**, as shown in Figure 3.39.

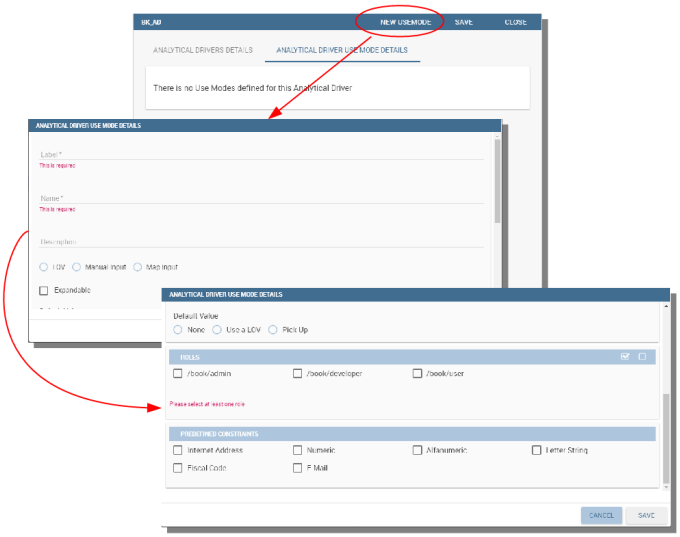


Figure 3.39: Detail panel of LOV creation, second step.

The first type allows the user to pick values from a previously defined LOV. When selecting this option the interface spread out the configuration panel where the user is asked to select a LOV from the list and a **`Modality`**. The latter defines how values are selectable at document execution. In fact the user can choose among:

List values selection: the filter will look like a lookup table;

Pop up: the filter will look like a lookup table;

Slider: the user can drag the slider to choose the parameter;

Tree: made for hierarchical LOV, lets the users navigate the parameters in a hierarchical way;

Combo Box values selection: the filter will look like a drop down menu.

The second kind of input expects the user to type manually the value. Otherwise the third opens a map from which the user must select one or more regions accordingly to the layer property. When selecting this option the interface spread out the configuration panel where the user is asked to choose a layer and the layer property. More details are furnished in next Sections 3.6 for this kind of input .

Moreover the user can add default values (namely values that will be passed to the document at its first execution) using the dedicated area. Here it is possible to pick default values from another LOV or to pick the first or the latter value of the current LOV (if the LOV input type was selected).

At the bottom of the page the user must associate roles to the “use mode”. This action is mandatory. The user connects the user’s roles that he/she wants to be allowed to see a certain

list of values or certain regions or be able to type values at his/her convenience.

Therefore, since an admin user can decide to separate values according to the other users' roles, the analytical driver definition allows to configure different use mode. We can also set validation checks if needed. Then it is sufficient to save each use mode and click on **`new usemode`** to set a new one. We repeat the same procedure for all the use modes. Each use mode is represented in a separate tab. We will go deeper into this at the end of the section.

All the selections can be multi-valued, but note that this option has to be set directly on the document detail during analytical driver association.

Creating an analytical driver for a spatial filter

In previous section we explained how to configure a driver and how it can be linked to different kind of inputs. In this part we linger on the possibility to define a spatial analytical driver. Referring to Figure 3.40, we notice that for setting the geographical driver we must select the **`map input`** option: here, expanding the combobox you choose the layer on which the filter will act. It is then necessary that the layer has been previously created and uploaded into Knowage **`Layers catalog`**. Then it is mandatory to specify the property name of the geometry in use using the manual text box just below. Remember that the property name must be exactly the same, therefore respect the upper and the lowercase of the string.

Figure 3.40: Spatial analytical driver settings.

These few steps will implent the spatial analytical driver to be associated to a document and be used to set a spatial filter.

Analytical driver's use modes

Sometimes the same analytical driver (i.e., the same concept, like the concept of product brand) should display different values according to the user that is executing it.

Suppose you have a report on sales and costs like the one in Figure 3.26 and you want to add to it the possibility to filter also on product brands. If you load the report as the general manager, you should choose between all the possible product brands in the corresponding parameter. If instead you load it as, for instance, the food manager, then you should be able to filter only on product brands related to the `Food` familiy.

In order to do this, let us focus again on the definition of the LOV and check that the already defined use mode `All Brands` is associated to the correct role `general_manager`. Here you can add a second tab, called for instance `Profiled_Brands`, and associate it to the role `product_manager`. This is because the food manager user has `product_manager` role with profile attribute `pr_family = Food`.

Finally, we choose the second LOV created, the one returning only those brands that belong to a specific family (see the code example in section Parametrizing LOVs). The family is selected by checking the value of the family attribute in the user profile.

Notice that here you can also choose a different type of display mode for the LOV. In other terms, different use modes correspond not only to different LOVs, but also to (possibly) different display mode (pop-up windows, combobox, ...). For instance, you can select a combobox display mode for the `All Brands` use mode and the pop up window display mode for the `Profiled_Brands` use mode.

Once you have saved the LOV, just log out from Knowage and log in with a different user role, i.e. as a general manager, food manager and drink manager. Executing your report on sales and costs you can now notice the differences on the values and on the display mode of the `Product Brand` parameters according to the different users. Notice that, for food manager and drink manager, the parameters are always displayed as a pop-up window, while for the general manager also the display mode of the parameter varies.

3.7 Visibility rules

The *analytical model* let you organize analytical documents in hierarchies and folders. Let's have a look on its structure.

Repository structure and rights

Knowage adopts a folder structure to organize analytical documents in hierarchies and folders. This structure is called the **Functionalities tree** and is accessible via **`Profile Management`** > **`Functionalities management`**.

There are two main reasons for organizing documents into folders and hierarchies: to facilitate their search and accessibility, and to effectively manage visibility on documents according to user roles.

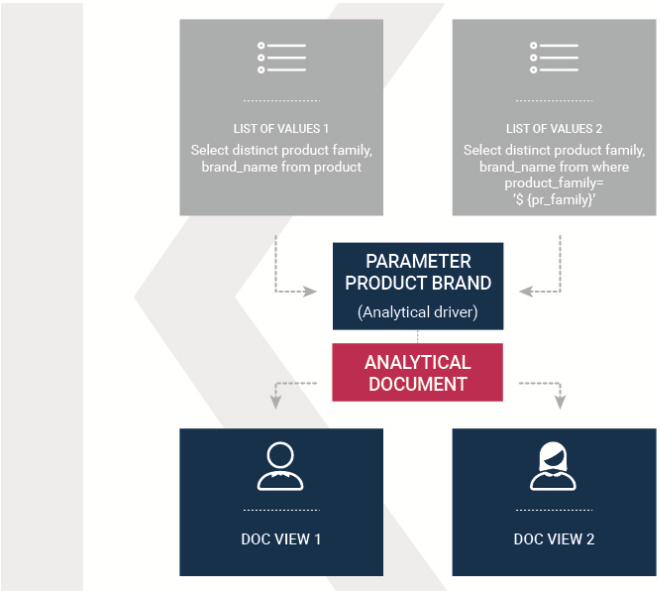


Figure 3.41: Behavioural Model Schema.

By default *permissions are set at folder level*. This guarantees that a user can not see anything outside that folder (unless he has permissions on other folders as well). It is also possible to further restrict the visibility scope of a user by associating rights to specific values of the profile attributes.

Besides visibility limitations inherited by the containing folders, the developer can add further restrictions to a single document.

To create a new folder, select **`Profile Management`**> **`Functionalities Management`**. The functionality tree is shown. Clicking on an item you can select one of the following options:

- **Insert**: to add a new child to the tree. Select this to create a new folder and go to the next step.
- **Detail**: to see details of an item.
- **Erase**: to delete an item. This option is available only if the folder does not have any children nodes in the tree.
- **Move up**: to move the item up into the hierarchy.
- **Move down**: to move the item down into the hierarchy.

Once you select **Insert**, the functionality details opens (see Figure 3.42). Enter a label and name for the new folder (functionality). In the table, assign permissions to roles.

There are four types of permission:

- **Development**: to create, edit and delete analytical documents;

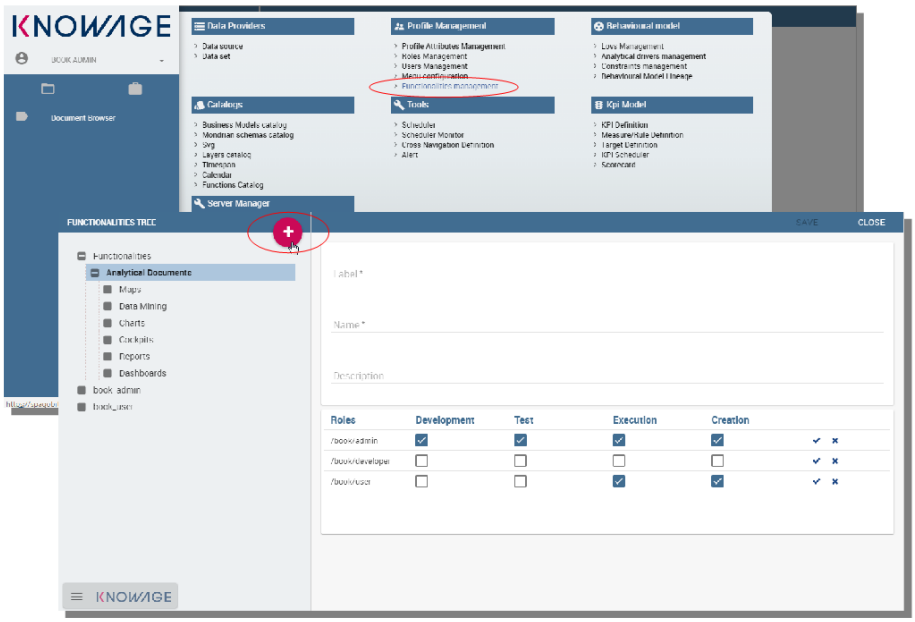


Figure 3.42: Create a new folder and assign permissions.

- **Test**: to execute the document and modify its status from `test` to `released`;
- **Execution**: to execute the document;
- **Creation**: to create ad-hoc reporting documents like worksheets and cockpits (for the end user).

To assign permissions to roles, check the related boxes. Each user with that role will have permissions on that folder, except in case of specific restrictions on the single document.

Permission Inheritance

A subfolder inherits the permissions assigned to the parent folder. While it is possible to further restrict inherited permissions, the opposite is not allowed: rights cannot be extended going down the hierarchy.

Menu configuration

Knowage allows the definition of a menu for the end user.

This menu will be displayed in the left bar of Knowage homepage, under the Knowage icon. It is possible to associate to each node a static page, a document, a functionality (as a folder) or nothing (empty node). Every node can be associated to different roles. This menu structure can be created and modified exclusively by the administrator in the **Tools** area. To access the Menu configuration area, go to **Profile Management** > **Menu Configuration** from the Main Menu.

It sufficient to click on the "Plus" of the Menu Configuration page to add a new folder to the **Menu Tree**. When selecting one node af the tree and clicking on the "Plus" icon, the user can add a new folder as a child of the former one, as shown in Figure 3.43.

Let us describe the different fields of the form.

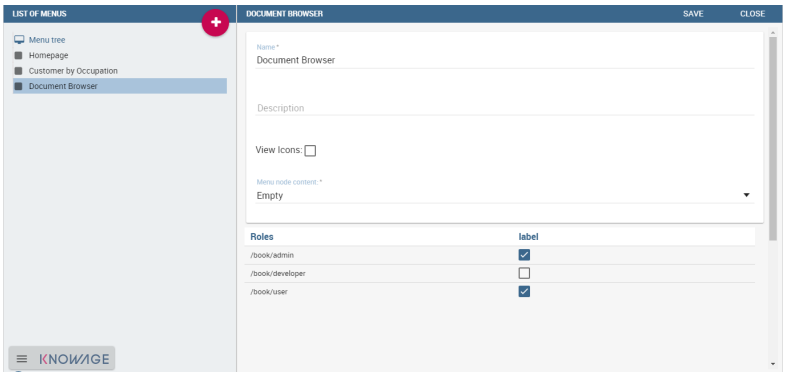


Figure 3.43: Menu tree actions.

In general you can:

- define a name and a description for the menu item; the name is what appears on the menu and it is a mandatory field;
- choose whether to view or not an icon near the menu node (for example, when there is a document associated);
- define the content of the menu item;
- choose the roles eligible to see that particular menu item. Only users associated with the selected roles will see this menu item.

Observe that when one inserts a menu item as a child, this will inherit the general details of a menu node.

There are four types of menu item content: empty, document, static page and functionality, as shown in Figure 3.44. The **empty** content type corresponds to a blank page, and it is typically chosen for father nodes.

The **document** content type runs directly a document. For this type you have to choose, see Figure 3.44, a related document through the lookup button and define the list of parameters

in the standard url (i.e.: `&par1=val1&par2=val2&...`). You can also choose to hide the toolbar or the slider panel.

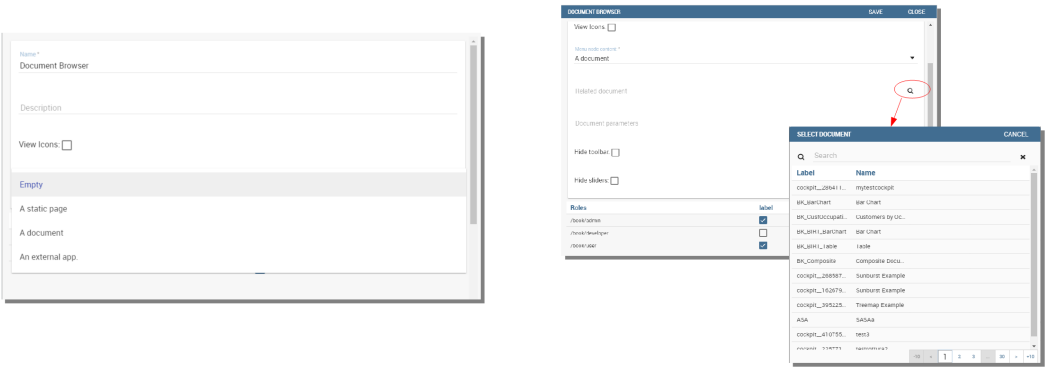


Figure 3.44: **`Empty`** (left) and **`document`** (right) content type.

The **`static page`** content type shows a static HTML page. In this case, the administrator must define the static page that he wants to load. The HTML page combo is loaded with all HTML pages found in a folder called **`static-menu`** that must be located under the path defined in the system variable called `Knowage_resource_path`.

Finally, the **`external application`** content type, see Figure 3.45, runs a URL address.

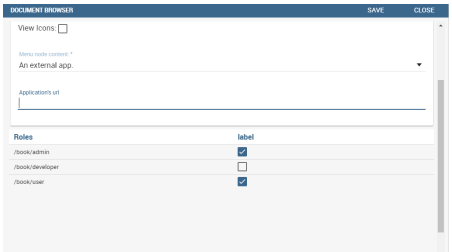


Figure 3.45: External application content type.

3.8 Server Settings

In the **`Server Manager`** menu panel you find some management functionalities, as Figure 3.46 shows.

Configuration Management

By clicking on the **`Server Settings`** > **`Configuration Management`**, you can manage many configuration elements. For example here you can set default language as well as mail settings.

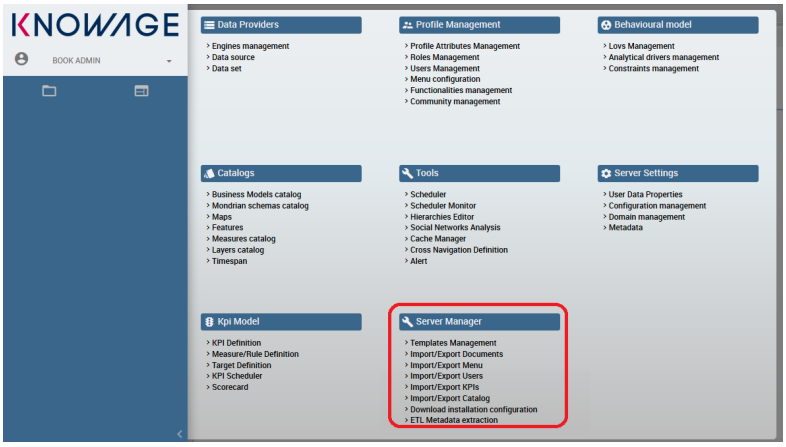


Figure 3.46: Server Manager Panel

Start typing `DEFAULT` in the search form, as shown in Figure 3.47, to filter among available items and find what you are interested it.

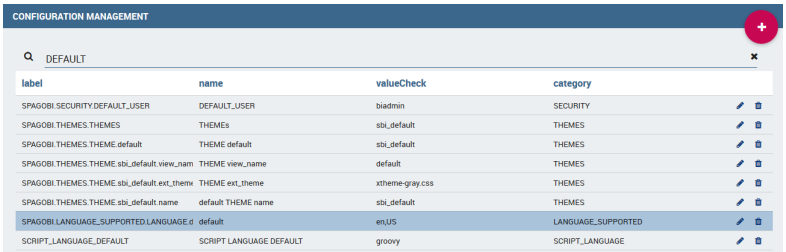


Figure 3.47: Configuration categories list.

We provide an example to let you understand the usage of the interface. Suppose you want to set italian as default language. Select the row with `SPAGOBI.LANGUAGE_SUPPORTED.LANGUAGE.default` as label and click the pencil icon at the end of the row to edit the element. Insert `it,IT` as **`Value Check`** as click **`Save`**.

You can view available languages and their code (**`Value Check column`**) in the row `SPAGOBI.LANGUAGE_SUPPORTED.LANGUAGES`.

Domain Management

By clicking on **`Domains Management`** item menu, you can manage categories. In Figure 3.48 we show Domain Management editor. You can add for example new categories for a business model, for a dataset and for all domain you can see in the **`Domain name column`**.

We provide an example to describe how it works. Suppose you want to add a new category among the datasets, named "Costs". In Figure 3.49 you can see the categories already existing.

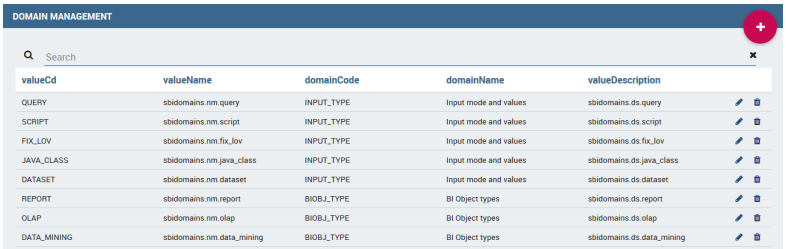


Figure 3.48: Domain management editor.

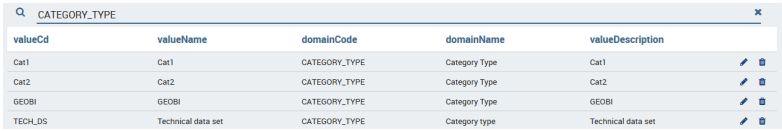


Figure 3.49: Business Model Categories already existing.

Click on the plus red button in the top right corner and by default a new page opens with the form you need to fill in. An example is shown in Figure 3.50. Fill the columns as follow:

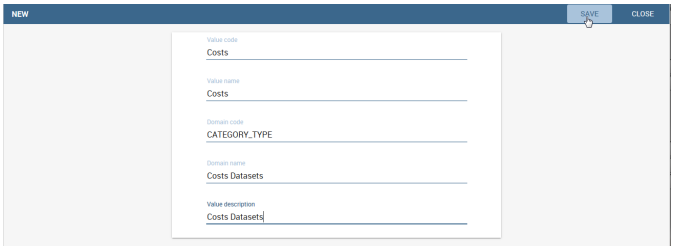


Figure 3.50: Form to be filled to create a new category.

- **`Value code`**: Costs
- **`Value name_image`**: Costs
- **`Domain code`**: CATEGORY_TYPE
- **`Domain name_image`**: Costs Datasets
- **`Value description`**: Costs Datasets

All the values except the **`Domain code`** to you. the last are mandatory for correct configuration. Now Click on Save. You have successfully create your new dataset category.

You can also modify an existing domain by selecting its dedicated row and clicking the edit button.

Chapter 4

Developer guide

4.1 Main concept

The creation and management of analytical documents in Knowage involves different elements:

Template. The template defines the standard layout of a document, including specific information on its appearance and the way contents should be displayed. For each analytical document the history of templates is maintained.

Dataset. Each document is associated to one or more datasets. The dataset provides actual data that will be represented according to the defined template. That is to say, the dataset provides the actual content of the analysis and the template is in charge of giving it a meaningful structure.

Data source. In order to retrieve data via the dataset, a source of information must be defined. Depending on the type of document, the data source may be associated to the document either directly or implicitly (via the dataset).

Parameters. Parameters allow the connection between the document and analytical drivers associated to it. In other words, at document execution time, each driver generates a value that is assigned to the corresponding parameter.

These elements are then combined inside each document engine, in order to produce different analytical documents. This process generates an HTML output, which can be accessed and navigated using a web browser. Other output formats are supported, including XLS, CSV, PDF, XML.

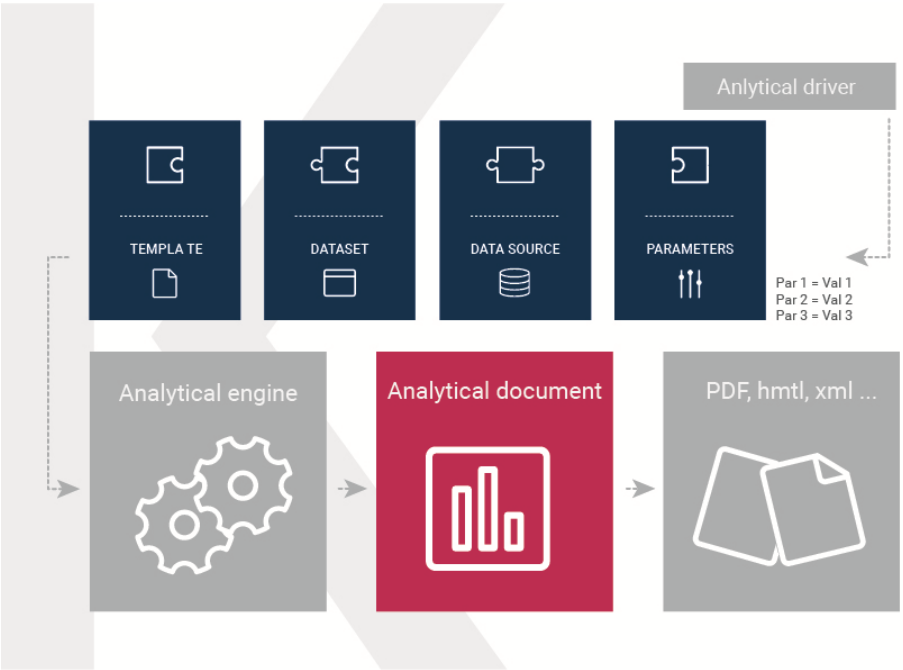


Figure 4.1: Components of Knowage analytical document.

4.2 Analytical document

Regardless of their type, all analytical documents interact with some generic components (such as cross-services) and with the specific engine that generate them (e.g. Report engine, OLAP engine). Therefore, all analytical documents are managed in the same way in terms of:

- document storage and versioning;
- document life cycle, based on a specific approval process including different status (development, test, released, suspended);
- multiple positioning on the repository and indirectly first visibility level;
- rules to restrict document visibility to some end user profiles;
- management of analytical drivers;
- multi-format export logic;

In the next sections we describe in detail how to create and manage analytical documents in Knowage.

4.3 Register an analytical document

There are two different ways to create a new document on Knowage Server. If you are developing a report usign Knowage Report Designer [1], simply click on **`Deploy`** and the document window will open with pre-filled options.

> **Deploy a document from Knowage Report Designer**
>
> Knowage Report Designer is the tool that allows to design and upload reports onto Knowage Server. Please refer to the dedicated section for full details and examples.

The second option is to manually create the document on the Server, using the relative web designer.

Analytical documents on Server

First of all click on **`Document Development`** from the BI functionalities menu, as shown in 4.2.

Figure 4.2: Documents Development button.

By default the page is divided into two parts, as shown in Figure 4.3: in the left side there is the functionality tree representing the folder structure, while on the right you can see the list of all documents contained in the selected folder.

[1] The Knowage Report Designer can be used to create report with Birt or composite documents only

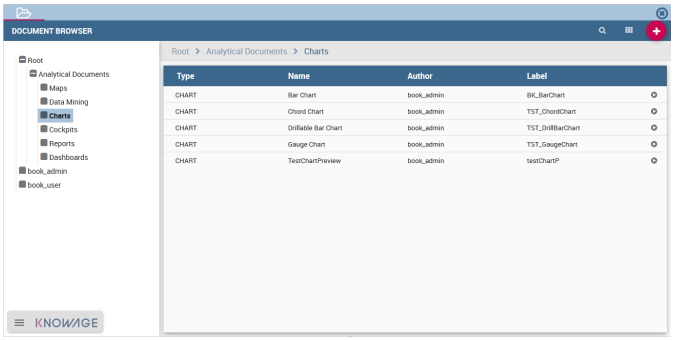


Figure 4.3: Documents Development section.

You can switch to the document preview view by clicking on grid icon in the top right corner, as shown in Figure 4.4.

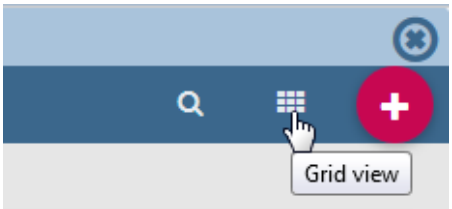


Figure 4.4: Changing documents view.

Each line shows the label, the name, the author and the type of the document, while the play button at the end of each row executes the document. Moreover, clicking on a line opens a side panel on the right of the page. Here you can see more metadata information such as the document description, the state and the creation date.

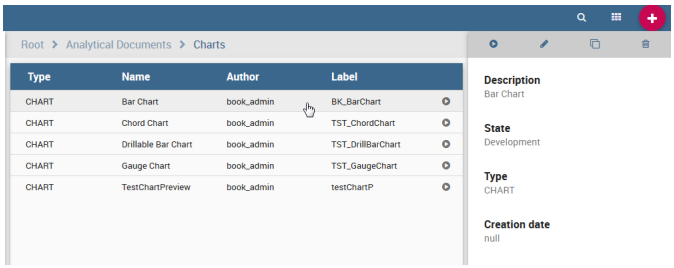


Figure 4.5: Side panel.

At the top of this side panel you find four button:

execute the document;

access document details;

clone the item;

erase the document.

Figure 4.6 shows the detail panel of a document. On the left, document details are shown, including **name**, **type**, **dataset** and **state**. On the right, you can alternatively see either the history of document templates or the functionality tree and the document position. If you want to copy or move a document from a folder into another, check or uncheck the corresponding folders (see Figure 4.10).

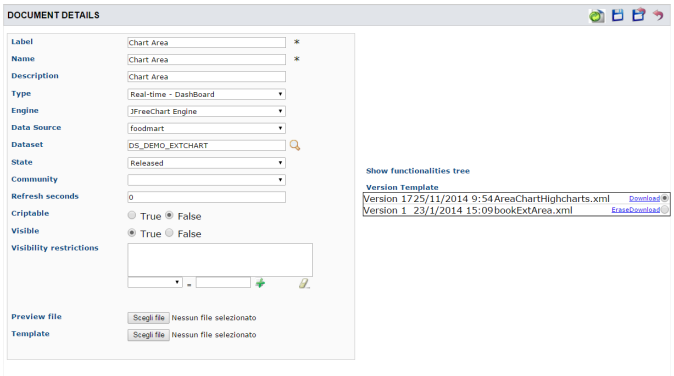


Figure 4.6: Detail panel of Knowage analytical document.

In order to create a new document you need to click on the red plus button (shown in Figure 4.4) in the top right corner of the **`Document Development`** page. The different types of documents that you can create are: **`Geo-Referenced Analysis`**, **`Cockpit`** and **`Generic Document`**. Please note that not all of them are available in all Knowage products.

To create a new generic document click the above-mentioned button and select **`Generic Document`**. You will be shown a window like the one in Figure 4.6 but with empty fields, in order to allow you to define the document details.

First of all, choose a **label**, a **name** and a **description**. It is important to point out that the **label** is the unique identifier of the document in Knowage Server. Then, select the type of document and the appropriate engine from the drop down menus, according to the document you are developing (see Figure 4.7).

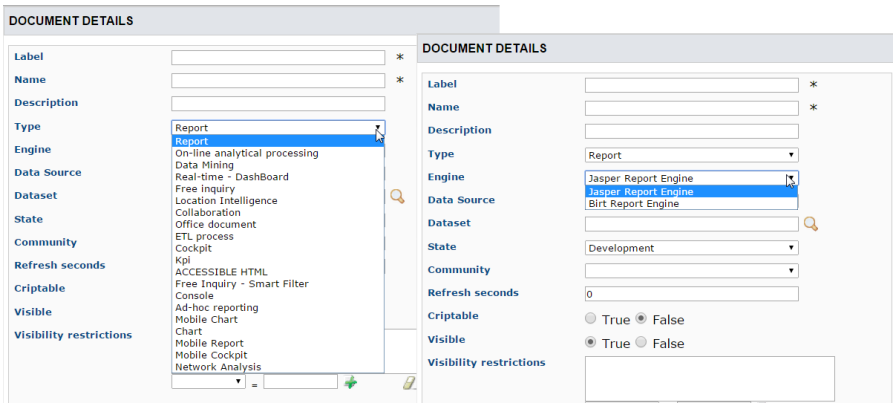


Figure 4.7: Select Type and Engine for a new document.

Now you have to select the dataset and data source that will feed your document with data, see Figure 4.8. Both should have already been defined in the corresponding sections for Knowage

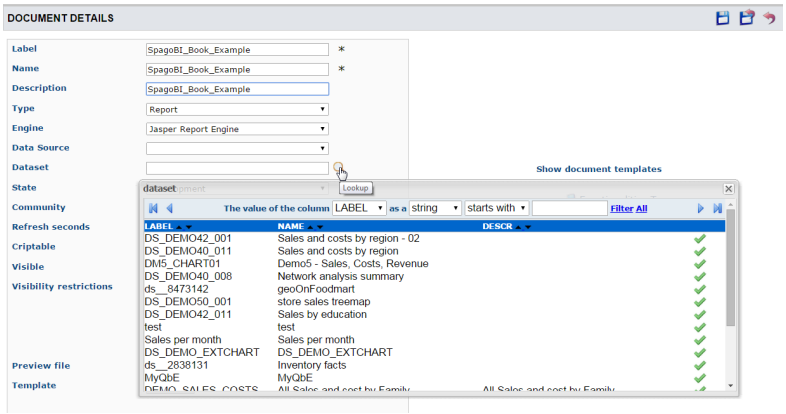


Figure 4.8: Selecting a dataset for the document.

to show them in the available options of the menus. Select the data source from the drop down menu. Then click on the green icon ✔ and select the dataset from the lookup window.

Note that some types of document do not require the definition of a dataset at this point because they use embedded datasets. Depending on the type, it may also be necessary to select the data source.

It is advisable to regularly save the document in this process, by clicking the related icon at the top right corner of the window.

Document lifecycle

The next step is to choose the status of the document using the **State** drop down menu. At any time in fact, each document is associated to a state, which will typically change over time following the development of the project. Those states are:

- development;
- test;
- released;
- suspended.

Upon creation, the document is by default in development state. Any time you upload a new template or make changes to the document, it is recommended that the state is updated so as to reflect its actual development state.

The main reason for this is that the state of the document has an impact on its accessibility. As discussed in the behavioural model, Knowage defines role types (administrator, developer, tester, user). States are compatible with the corresponding role type. Administrators can change the state of documents at any time. Developers can not access only the documents with test state. Testers can not see documents in development or suspended state. Users can execute only documents in released state. Note that a tester may change the state of a document from test back to development.

Template Versioning

Knowage Server supports versioning of uploaded templates, as shown in Figure 4.9. To view

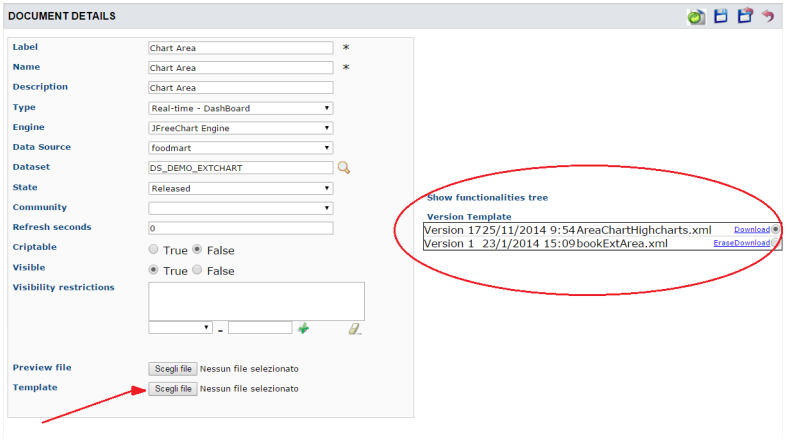


Figure 4.9: Template versioning for analytical documents.

them in the document detail window, click on **Show document templates** in the right panel. All

templates are saved with their date and name, and can be easily restored or deleted. To restore a template, choose it in the list by clicking on the selector, then remember to save: the new template will be uploaded. Using the same list you can download or delete a template.

Document Visibility

After having defined all details, you need to choose where the analytical document shall be saved in the functionality tree. This choice has an impact on the visibility of the document. Since folders in the functionality tree are subject to different access policies, which can be set when creating the node, then each document saved in that folder will inherit permissions accordingly.

Repository structure and rights

The **`Functionalities tree`** is Knowage document repository. It a is managed by administrator, who is in charge to profile user visibility too.

Note that the same document can be saved in different points of the functionality tree. This allows the administrator to make the document accessible to multiple roles based on visibility rules defined for the containing folder(s). To save your document in the repository, switch the perspective on the right panel by clicking on **`Show functionalities tree`**. This operation is needed only if you moved to the template history view. Here you can choose where you wish to save the document, by ticking the corresponding folder in the tree. If you wish to save it at multiple locations, tick all of them before saving. Each user having access to the containing folder will see the document.

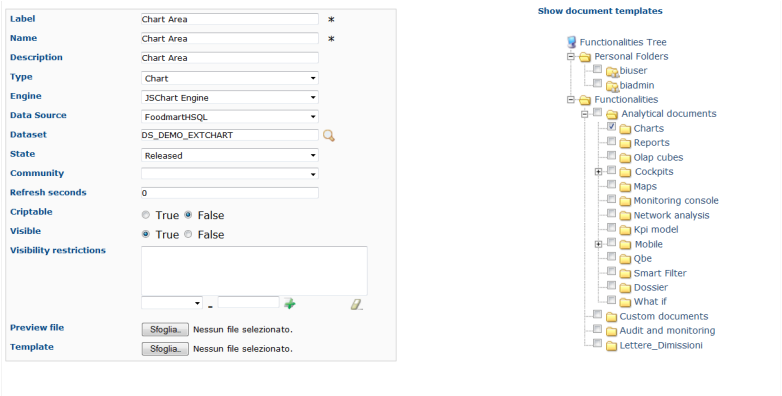


Figure 4.10: Functionality Tree, document saving settings.

4.4 Visibility rules

In addition to the standard mechanism supported by the functionalities tree, it is possible to further customize access to a document based on user profile attributes. This allows administrators to rule access to documents at a very fine-grained level, beyond simple repository-based policies.

This can be done by editing conditions in the **Visibility** section of the detail panel. To add a new condition pick a profile attribute from the drop down menu, assign it a value, then click on . This will add a new condition that must be verified to allow a user to access the document. In the same way you can add further conditions, and possibly remove all of them by clicking on the eraser.

4.5 Association with analytical drivers

We have already discussed the role of analytical drivers and how they are connected to analytical documents via parameters. In this section we will show how to practically define this association.

We assume that the document template and datasets are correctly set in terms of parameter definition. In particular, they should have been correctly referenced with their URL.

To add a new parameter, you can start editing the tab in the lower part of the document detail panel, see Figure 4.11.

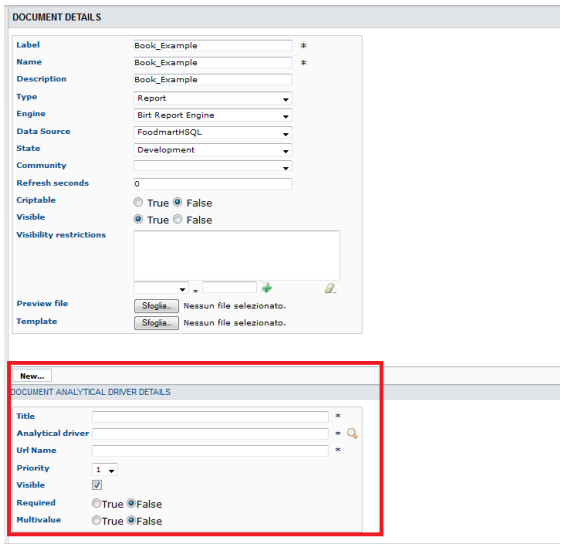
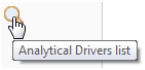


Figure 4.11: Association with analytical driver panel.

Choose a human readable name for the title. Then click on the lookup icon 

to choose the driver you wish to associate to the document. This will open the driver lookup window, where you can select the driver by clicking on the green icon . You can also inspect or delete a driver from here.

Once you have selected the driver, you should write the **`exact URL`** of the corresponding parameter. Then set the different features associated to the driver: you can set its visibility and decide if it is mandatory and multivalue. By default the parameter is visible, not mandatory and not multivalue.

If you want the document not to be visible to end users, untick the **Visible** checkbox. Note that the parameter will still exist and receive values from the associated driver. However, this will be hidden and the end user will not be able to choose any value for this parameter.

If you want to set it as a mandatory parameter just click on **`true`**. In this case, no default value is set. The end user will be asked to choose the value of the parameter before opening the document.

Similarly to set a parameter as multivalue click on **`true`**, in this way the user can perform multiple selections on among its values.

Remember to save each time you have completed the definition of a parameter before adding a new one. To add further parameters, click on the **`New`** icon. Repeat the same procedure how many times you wish. At this point you may wish to change the order of parameters (i.e., how they are presented to the user). To do so, modify the **Priority** number. In the following we will see some special operations that can be performed on drivers associated to a document.

Correlation between parameters

In the context of a document, two different parameters may be connected to each other: this means that the possible values of a parameter are limited by the value(s) of another parameter.

This feature can be useful when two (or more) parameters are logically related. For example, suppose to have a parameter for all the possible countries and another one for all the possible cities.

If the user selects a region, it is meaningless to show him all cities: he should only be enabled to choose among the cities in the selected region.

In general, to configure a correlation within a document you should make sure that the LOV associated with the parent parameter and the one associated to the child parameter share at least one column. This column defines which value from the parent parameter will be applied to the child, in order to constrain the results.

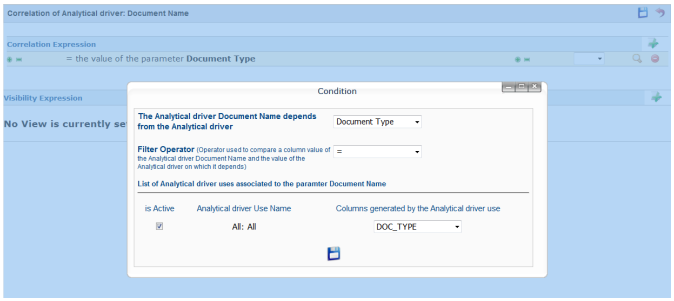


Figure 4.12: Definition of the correlation.

To set the correlation, click the tab of the child parameter and displaying the details click on the correlation button .

Here you can add a new correlation rule by clicking on . Here you need to define:

- the parent parameter;
- the type of logical operator, in order to compare values of the parent parameter with values of the child parameter;
- the column, generated by the child parameter, whose value will be compared with the value of the same column in the parent parameter.

If a parameter depends on multiple parent parameters, you can define multiple correlations. Create the needed correlations and choose how they are logically connected (via AND / OR operators) as shown in Figure 4.13.

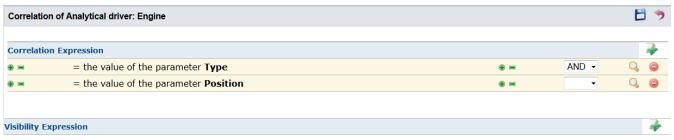


Figure 4.13: Multiple correlations.

If needed, you can insert or remove parenthesis at the extremes of each line clicking on the two green plus and minus icons.

Once defined the correlation, the child parameters will display the labels during the runtime in italics.

Controlled visibility

Another type of relation between parameters is supported by Knowage. It is possible to define values of a parent parameter that force the hiding or showing of a child parameter in the

parameters mask. Note that in the first case, the child parameter is hidden by default, while in the second case the parameter is shown by default.

To set a visibility expression, click always on the correlation button in the detail tab of the desired parameter, but then click on the plus icon in the **`Visibility Expression`** area. In the graphical editor you can define visibility rules similarly to correlation ones, as shown in Figure 4.14.

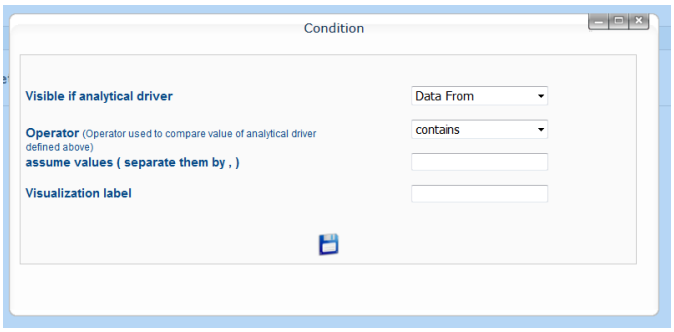


Figure 4.14: Visibilty expressions.

4.6 Cross Navigation

A powerful feature of Knowage analytical documents is the *cross navigation*, i.e., the ability to navigate documents in a browser-like fashion following logical data flows. Although cross-navigation is uniformly provided on all documents executed in Knowage Server, each type of document has its own modality to set the link pointing to another document.

Notice that the pointer can reference any Knowage document, regardless of the source document. For example, a BIRT report can point to a chart, a console, a geo or any other analytical document.

In Knowage there are two main typologies of cross navigation: *internal* and *external*.

Internal cross navigation updates one or more areas of a document by clicking on a series, a text, an image or, in general, on a selected element of the document.

External cross navigation opens another document by clicking on an element of the main document, allowing in this way the definition of a "navigation path" throughout analytical documents (usually, from very general and aggregated information down to the more detailed and specific information)). Indeed, you can add cross navigation also to a document reached by cross navigation. This can be helpful to go deeper into an analysis, since each cross navigation step could be a deeper visualization of the data displayed in the starting document.

It is obviously possible to associate more than one cross navigation to a single document. It means that by clicking on different elements of the same document the user can be directed

to different documents.

In this chapter we will examine in depth how to set output/input parameters on documents and, consequently, how to activate the cross navigation.

The first step is to define the parameters of the target document. These do not necessarly coincide with all the filters applied to the document. Please refer to Section 3.6 for more detail on how to manage parameters and their association to documents.

Therefore it is required to state which parameters among the ones associated to the target document are going to be involved in the navigation. Parameters coming out from the source document are said **`output parameters`** while the ones that receive values through the association (with the source document) are said **`input parameters`**. By the way, when declaring the parameters they will be called equally **`output parameters`** at first, since there is no criterion to distinguish output from input before the navigation is configured.

The definition of the output parameters is performed using the **`Manage output parameters`** button (see Figure 4.10) but it differs from document to document, according to its type. We will describe these differences in detail in each dedicated chapter, here we explain the common steps.

Declaration of the output parameters

Enter the **`Document details`** of the document of interest. Then click on **`Manage output parameters`** and the **`Output parameters`** dialog will open.

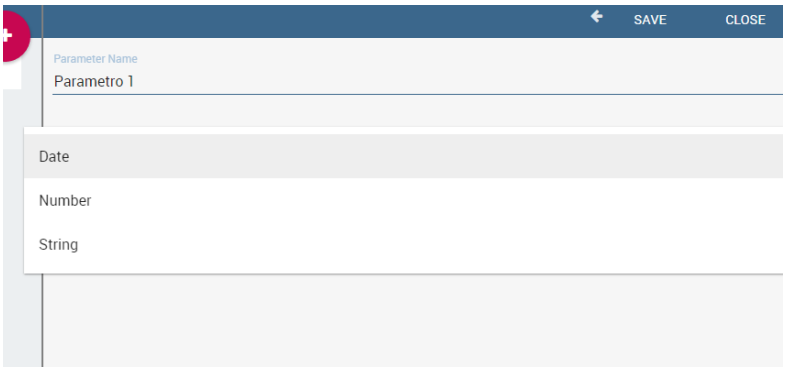


Figure 4.15: Setting an output parameter.

Here you have to state which parameters are going to be used as output parameters. If, for instance, you select the `Date` type (Figure 4.15), it is possible to choose the format in which your date has been coded. The default value is related to the locationing defined in (**`Menu`**>**`Languages`**).

Cross navigation definition

Finally you need to select the **`Cross Navigation Definition`** item from the menu to configure the cross navigation. Figure 4.16 shows the cross navigation definition window.

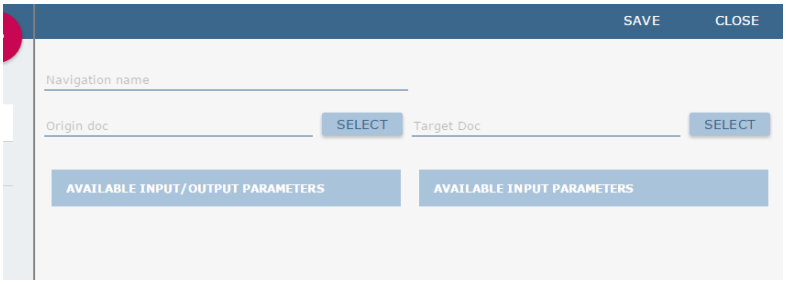


Figure 4.16: Cross navigation GUI.

It is required to give a name to the navigation; then select the document from which to start the navigation and the target document. The selecting of a document will cause the loading of input/output parameters related to the starting document in the left column and of the possible input parameters of the target document in the right column.

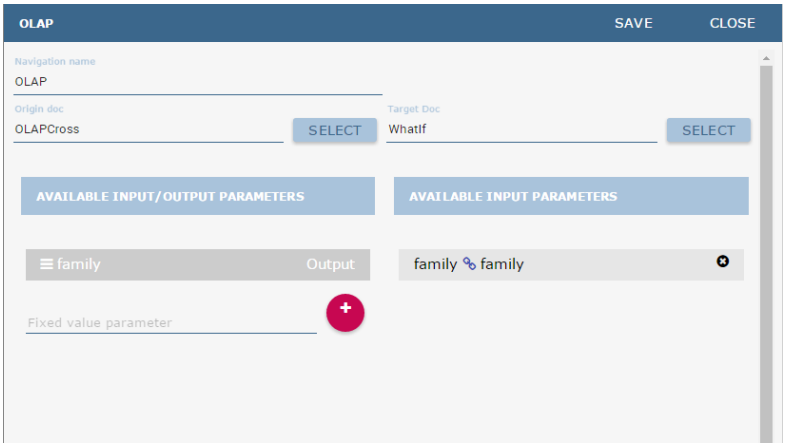


Figure 4.17: Setting the cross navigation through the menu item.

It is possible to configure the associations betweeen input/output parameters by simply dragging and dropping a parameter from the left column on another of the right column.

Once set, the association is highlighted as in Figure 4.19.

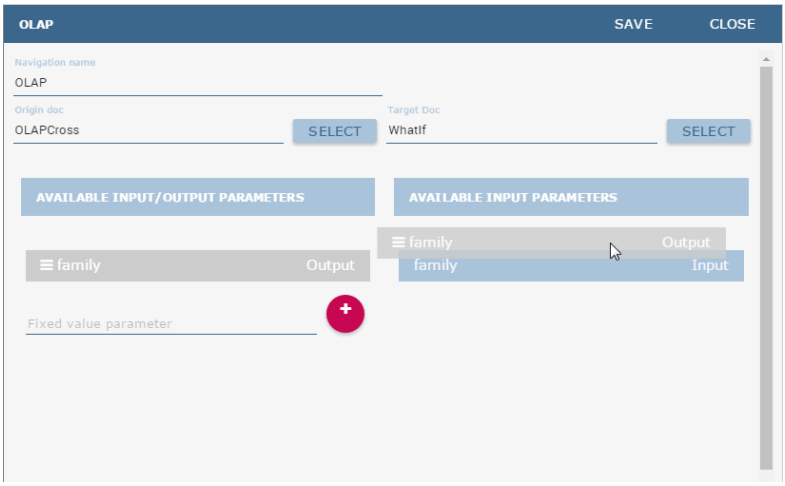


Figure 4.18: Relating parameters.

Figure 4.19: Association betweeen parameters.

To assign fixed values to target parameters it is necessary to edit first the box labeled **`Fixed value parameter`** and click on the "plus" icon. Then the value can be associated as fixed value of the one or more target parameters. Remember to click on the **`Ok`** button to save the cross navigation just set.

4.7 My first Chart

Charts are the most adopted method in presenting BI data since they allow an immediate perception of a phenomenon and are easily understandable. Focused on a visual impression more than a punctual lecture of values, they are specially suited to show trends and comparisons.

For these reasons, charts gain a pervasive level of usage and can be used by anyone to perform both synthetic and detailed analysis.

Knowage provides a chart engine to create several types of charts, including:

- Bar Chart
- Line Chart
- Pie Chart

- Sunburst Chart
- Wordcloud Chart
- Treemap Chart
- Parallel Chart
- Radar Chart
- Scatter Chart
- Heatmap Chart
- Chord Chart
- Gauge Chart

Once you enter the Knowage environment as a final user, enter the **`Analysis`** area under the **`Workspace`** menu item, click on the **`Create Analysis`** icon and choose **`Cockpits`**. Please note that this operation is available only in KnowageBD and KnowageSI. With KnowagePM only the developer can create charts document. He can open the chart designer as explained in Section 4.7.

Once opened, the cockpit interface is an empty page with a toolbar containing different options (see Figure 4.20).

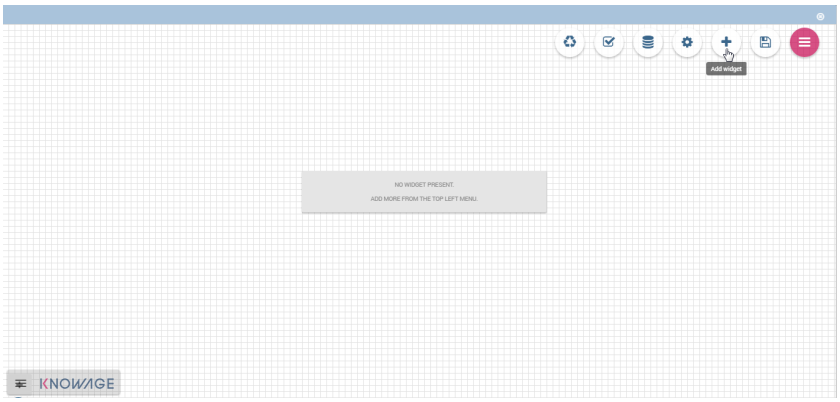


Figure 4.20: Add a chart to a cockpit.

Cockpit

The **`Cockpit Engine`** allows the user to self-build interactive cockpits through an intuitive and dynamic interface. Read more in Chapter 4.8.

Clicking on the **Add Widget** icon, you will be asked to choose among some available widgets. Pick out the **Chart** one and let's now go into details on how to build a chart from the scratch. The designer editor is divided into four principal tabs: **Dataset**, **Chart Engine Designer**, **Style**, **Cross** and **Filters**. As soon as the user clicks on the "Add Chart" button, he/she enters the "Dataset" tab editor. Here the user must select, using the "little plus" icon placed just aside the combobox line, one dataset. Then the user must switch to the "Chart Engine Designer" tab and choose a chart type among the available ones, as shown in Figure 4.21.

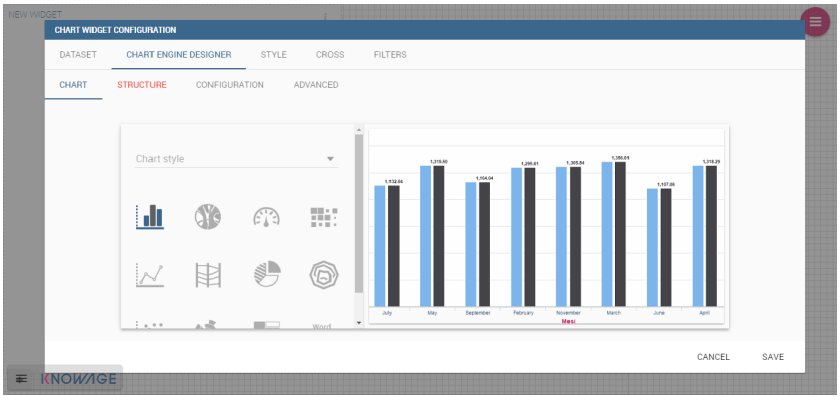


Figure 4.21: Chart editor.

After choosing the appropriate chart type you must go into the **Structure** page. Here it's possible to select the measures and the attributes chosen for the chart.

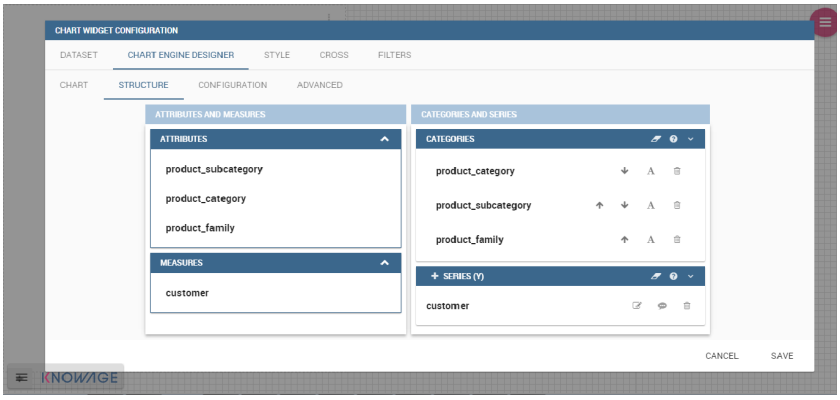


Figure 4.22: Chart structure.

Clicking on the **Configuration** page you will found seven different blocks as you can see in Figure 4.23

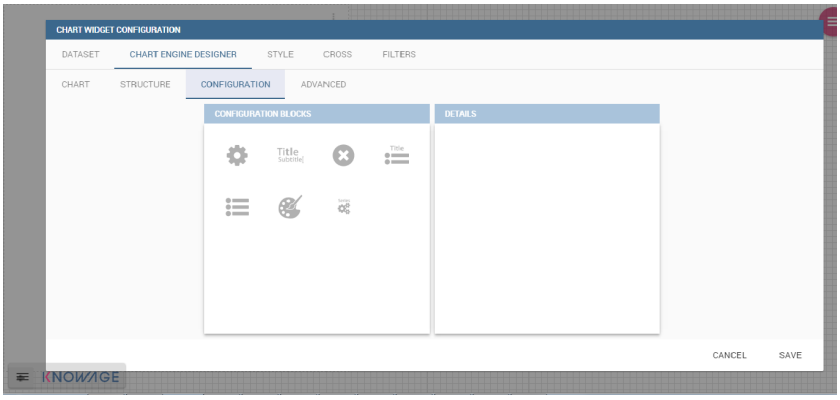


Figure 4.23: Chart configuration.

In detail this blocks concern:

- **`Generic Details`**, as the orientation of the chart, the family and the size font.
- **`Title and Subtitle details`**
- **`No data message`** where it is possible to put a message where the data are not founded.
- **`Legend Title`**
- **`Legend Items`**
- **`Color Palette`**
- **`Advanced Series Configuration`**

These seven blocks are the most common, but for some chart types may change.

The last section is the **`Advanced`** where expert users find advanced features. Here the user can see the representation of the generated json, which is showed as hierarchical tree, the values of the parameters set in the previous steps and configure advanced options.

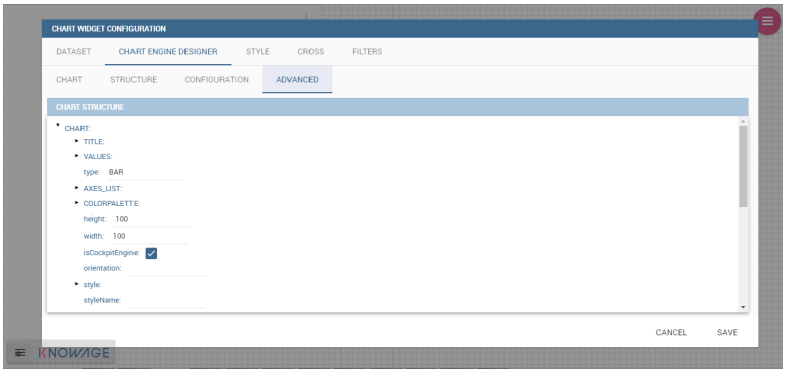


Figure 4.24: Chart Advanced Features.

In the next subsections, the available functionalities of the Structure and the Configuration tabs are described in a more specific way.

Structure

The first step to create a chart is to be sure of what kind of chart you want to select. Anyway you can change type of chart afterwards, but it may cause a loss in the settings already defined. Then, click on the "Structure" tab of the designer to choose the measures and the attributes. In fact, the tab shows a two axes panel. The horizontal axis indicates the X-axis where you must choose one or more attributes. As well, the left axis is the Y-axis and here you must choose measures. You can also insert manually the axis title for both the X and the Y axis if the chart is configured to have axis titles.

In this section it's possible to customize the labels of the axis, title and grid style clicking on different buttons. With the arrow button, on the top of the Y-axis and X-axis, it's possible to choose the axis configuration detail, the axis title configuration, the major and minor grid configuration (just for Y-axis) and ordering column (just for X-axis). With the pencil button opens a window on the right with the series configuration details where it's possible to choos the aggregation way, the order type of the series, if the data will be shown e so on. Finally, with the strip cartoon button you can choose the features of the tooltip (font color, text alignment, ecc). If the chart in place does not allow the customization of the axes the specific button will be disabled or not visible. Figure 4.25, Figure 4.26 and Figure 4.27 will show in detail the three buttons above explained:

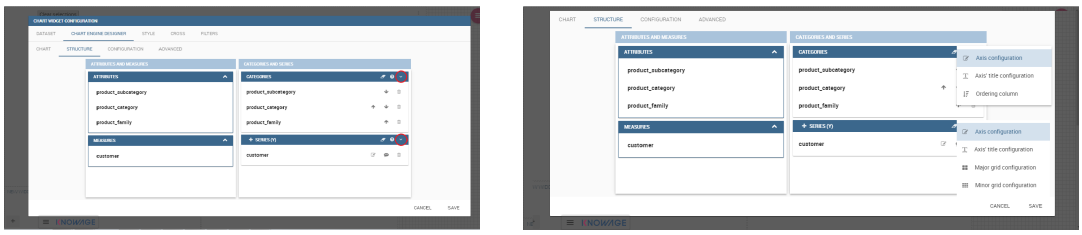


Figure 4.25: From left to right: (a) Generic configuration axis (the specific arrow). (b) Generic configuration axis.

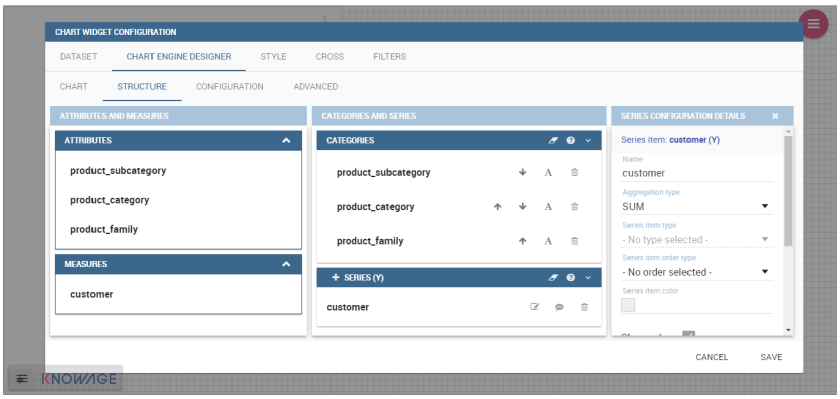


Figure 4.26: Series style configuration.

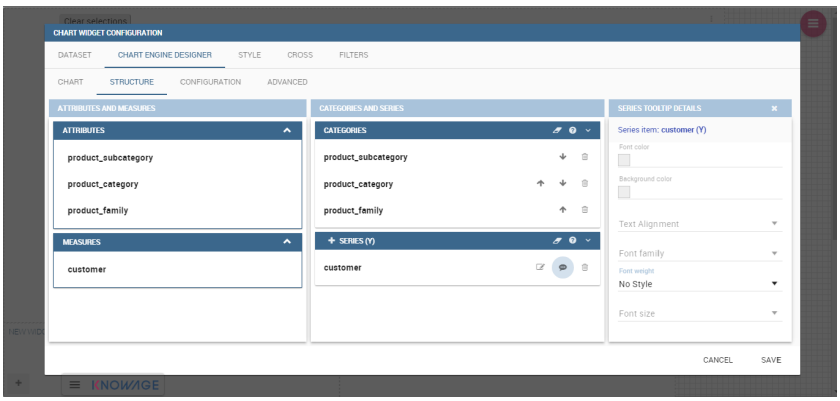


Figure 4.27: Series tooltip details.

Configuration

The **`Configuration`** section contains options to define the generic style of the chart. Here you can set the dimensions of the chart,the background color, insert the title and subtitle and define their style, choose the series palette, add and configure the legend. The listed options are an example of what you can configure in the tab.

Note that for the color palette details you can use one already in the list or you can choose any color inserting the hex color code with the hashtag symbol. This is a very useful feature to customize the output.

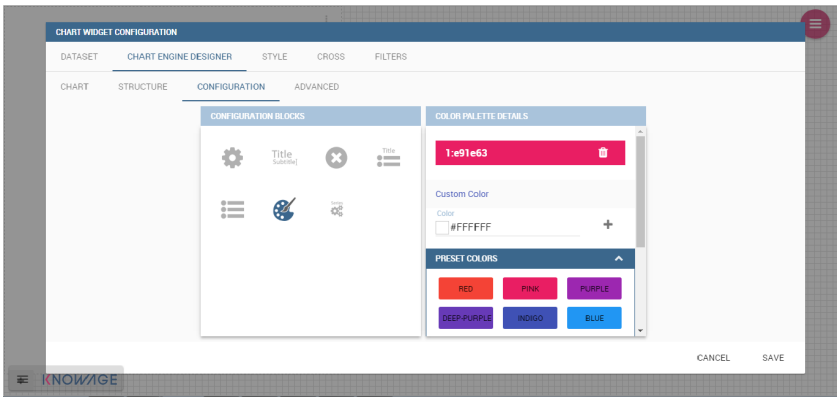


Figure 4.28: Color box editing.

Indeed, the options available in this tab change according to the chart selected enabling different configurations.

Stand alone charts

The previous chapters were dedicated to the end user approaching the Knowage Chart engine. We stressed how the final user must pass through the Cockpit interface to develop graphs. We want now spend some words about the developer experience. Indeed, if you are a technical user you can also create a chart as a stand alone document.

Once you enter the Knowage environment with developer credentials, open the technical menu directly into the **`Documents Development`** area, as shown in Figure 4.29.

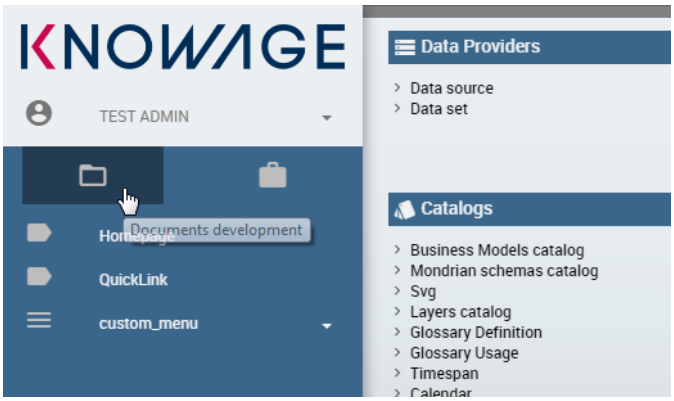


Figure 4.29: Documents Development.

Then click on the "Plus" icon of the **`Create Document`** feature and select **`Generic Document`**.

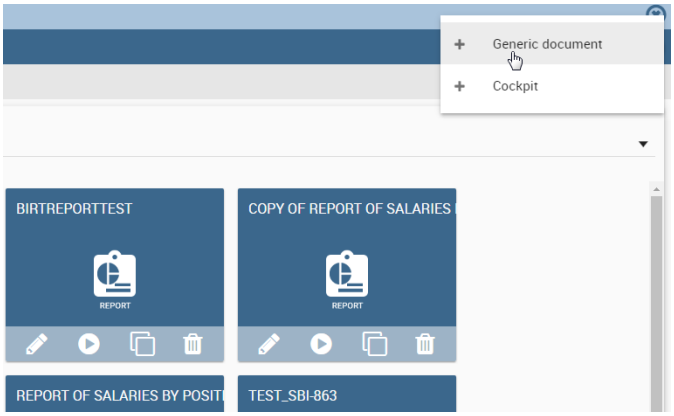


Figure 4.30: Create a new document.

You will be asked to fill in the form. We give an example in Figure 4.31 The fields marked with an asterisk are mandatory. Select the **Chart** type and engine. Choose the dataset with which

you want to manage your analysis. Use the magnifier to choose among the available datasets. Remember to pick out in which folder you want your chart to be stored (see Figure 4.32) and finally save.

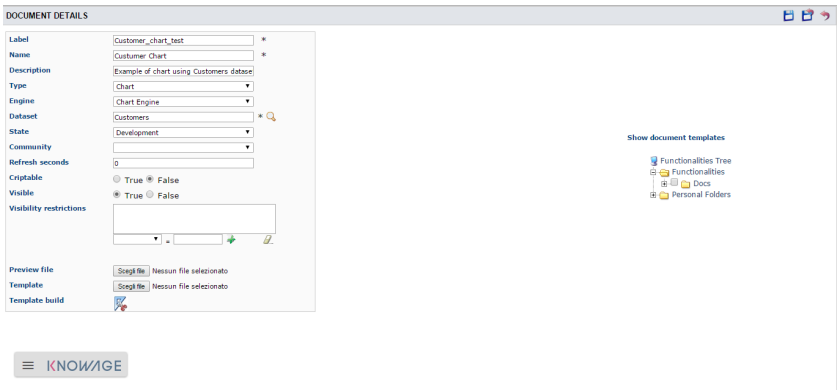


Figure 4.31: Document Details.

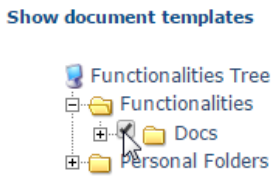


Figure 4.32: Select the folder in which you want your chart to be saved.

A new template can be generated through the editor clicking on **`Template build`** as showed in Figure 4.33 or a template previously created can be uploaded.

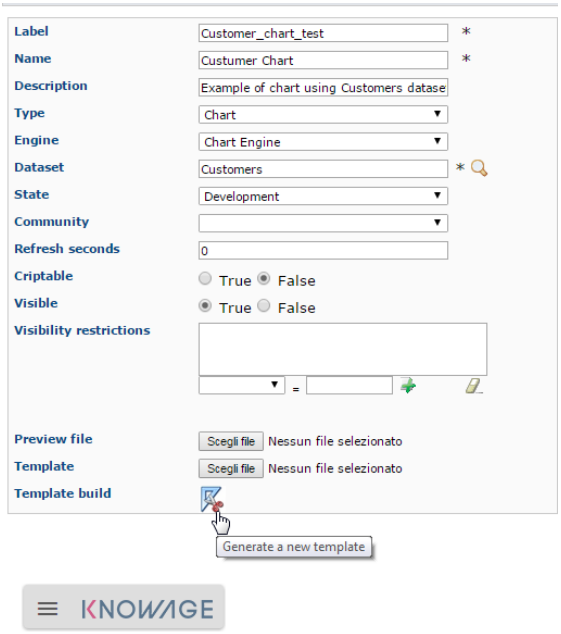


Figure 4.33: Template build.

If you choose to implement the new Chart through the Template Build feature, the steps to follow are exactly the same of those seen for the final user. In fact, once you click on the Template Build icon, you are redirected to the Chart designer. In this case, by the way, another functionality is enabled, the Cross Navigation.

4.8 My first Cockpit

You can create your new Cockpit from the **`Analysis`** area of the **`Workspace`** by clicking on the "Plus" icon and selecting **`Cockpits`** if you enter Knowage Server as final user, while you can enter the document browser and start a new cockpit using the "Plus" icon if you enter Knowage Server as admin.

Reaching the cockpit designer

We stress that the cockpit interface is reached by the final user and the administrator following two different paths.

Let us see how to build a cockpit and how the interface is displayed within the server. Once

opened, the cockpit interface is an empty page with a toolbar containing different options described in Table 4.1.

Icon	Name	Function
	`Cockpit menu`	Configuration menu of Cockpit.
	`Add widget`	It opens a window where you can create a new chart or table, add texts, images or Knowage documents.
	`General configuration`	It opens the window where you set the general cockpit options (name, label, show menu, etc.) and widget style (header, titles, borders, etc.).
	`Data configuration`	It opens a window where you can manage the dataset, the association between datasets and the refresh frequency.
	`Selections`	It adds a widget that manages selections.
	`Clear Cache`	It cleans temporary data.
	`Save as`	It opens the window to save the cockpit document as a new document.

Table 4.1: Cockpit editor toolbar.

By clicking the button **`Add Widget`** you can add a widget containing a **`Text`**, an **`Image`**, a **`Chart`**, a **`Table`**, a **`Static Pivot Table`** or a **`Document`** to your cockpit, as shown in Figure 4.34.

Figure 4.34: Widget Type.

In the following we go into details of each available widget.

Text widget

By clicking the button Text Widget you can add text to your cockpit. As already seen the widget editor opens and it is divided in three tabs: the **`Text editor`**, the **`Style`** and the **`Dataset`** tab.

On the "Text editor" tab you can type the text desiderated in center panel and customize it. On the right side you see the elements available to create a dynamic value. it contains the functions and fields of the dataset or datasets that you have selected in the third tab. To add a function to a measure first select the desired function and then the field of numeric type. We provide an example in the Figure 4.35.

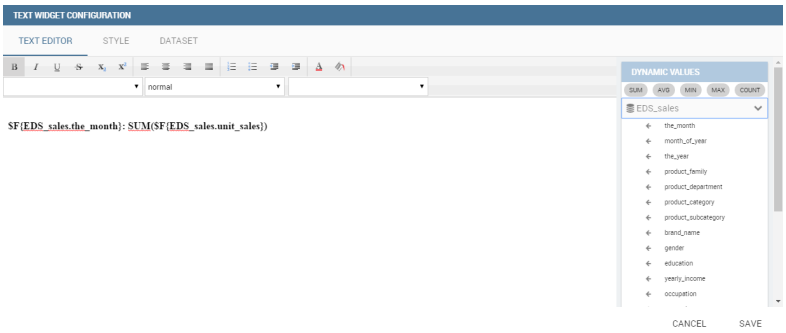


Figure 4.35: Text editor of text widget configuration.

On the "Style" tab you can customize the text widget. We have provided all details about this tab in the Table widget.

On the “Dataset” tab you can add more dataset to be used in the dynamic value.

On the right side there are two tabs, the **`Generic Configuration`** tab and the **`Custom Congifuration`** tab. Once you have finished, click save and your text widget will be created inside the cockpit.

Image widget

By clicking the button **`Image Widget`** you can add images to your cockpit. As already seen the widget editor opens and it is divided in three sections.

On the **`Gallery`** tab you can upload an image, delete it or select one from the gallery. Refer to the Figure 4.36.

Figure 4.36: Gallery tab of Image Widget Configuration.

On the **`Style`** tab you can configure the style of your image widget with the different options offered by this tab. Many of them are defined in the table widget that you will find later.

On the **`Cross`** tab you can define navigation to another document, as shown in Figure 4.37. For this purpose, you must activate **`Enable cross navigation`** flag and select the destination document through the list of cross navigation definition. This last flag is optional. If you select a cross navigation definition, when you launch the cross navigation it will go to the document of arrival directly. If the cross navigation definition is not defined, then when you launch the cross navigation will be shown a pop up with the list of cross navigation definition that exist for this cockpit.

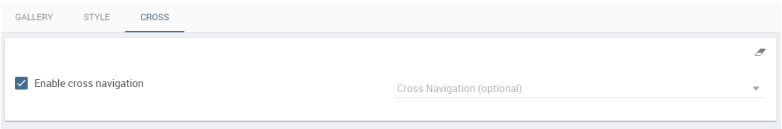


Figure 4.37: Cross tab of Image Widget Configuration.

Chart widget

Charts are an essential representation of data, Knowage let you use many different charts type and configure them according to your needs. We have provided all details about charts type and configuration in Chapter 4.7.

Table widget

The **`Widget table configuration`** opens and it guides you through the steps to configure the widget. The pop up opens showing the **`column`** tab, as you can see from Figure 4.38. In details, it is mandatory to select a dataset using the combobox (only if one or more datasets have been loaded using the **`Data Configuration`** feature) or clicking on the icon + available just aside the combobox line. You can page the table specifying the number of rows per sheet. Consequently the user can set columns properties.

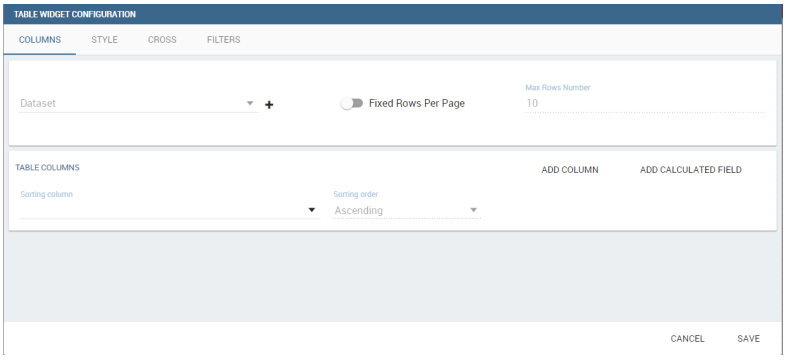


Figure 4.38: Table configuration window.

In fact, the column area is divided into two parts: on the left side you have columns ordering, on the right the user has the button to add a new column or a calcutated field. As soon as yhe dataset is selected, you can indicate the sorting column or modal selection column. The modal selection serves to specify which value will be passed to other widgets (if interaction is enabled) when clicking on the cell at document execution time. You can specify this field by selecting a value from the combobox. In the same way, you indicate the sorting column and the order type that steers the rows ordering. You can select the field and the order from the the dedicated comboboxes.

As beforehand mentioned, it is possible to add new columns. In order to add a new column you have to click on the **`Add Column`** icon on the top right of the second box. Once opened you can select one or more columns. When you have finished selecting the desired columns you can click on save button and your new columns will appear in the field list. Refer to Figure 4.39.

Likewise, to add a calculated field you have to click on the **`Add Calculated field`** icon next to add column icon. Once opened the Calculated Field Wizard you have to type an alias for

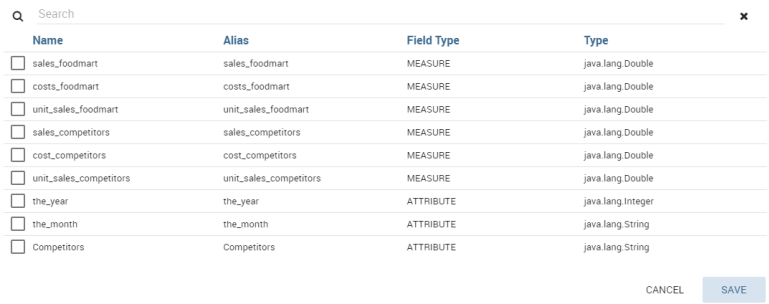


Figure 4.39: Add a new column.

your calculated field in the dedicated area at the top corner of the wizard. Then you can choose from the Items Tree the fields and the arithmetic function you want to use for building your expression. In the middle you can see the expression you have built. If you prefer you can create or modify the expression manually directly in the panel which is editable. When you are satisfied with your expression you can click on save button and your calculated field appears in the field list. We provide an example in the Figure 4.40.

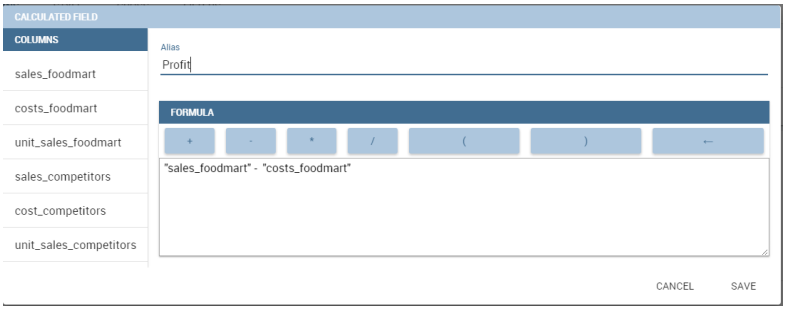


Figure 4.40: Add a calculated field.

At the very bottom of the window, you can see the dataset fields listed and you also can sort columns displayed in the table, insert a column alias and customize it by adding font and style configurations using the brush shaped icon, as you can see from Figure 4.41. Here you can find configuration features like the column size, max cell characters, hide on mobile option, etc.

Note that here you can indicate the column type and the aggregation. To add an aggregation to a column you must control the type of data that column has. An aggregation can only be added if the column value is of "number" type . The different aggregation functions are: *none (you also can not add any aggregation function)*, *Sum*, *Average*, *Maximum*, *Minimum*, *Count* and *Count distinct*.

The **`Style`** tab is where you can customize the table by using the different options of style. It is divided into eight parts:

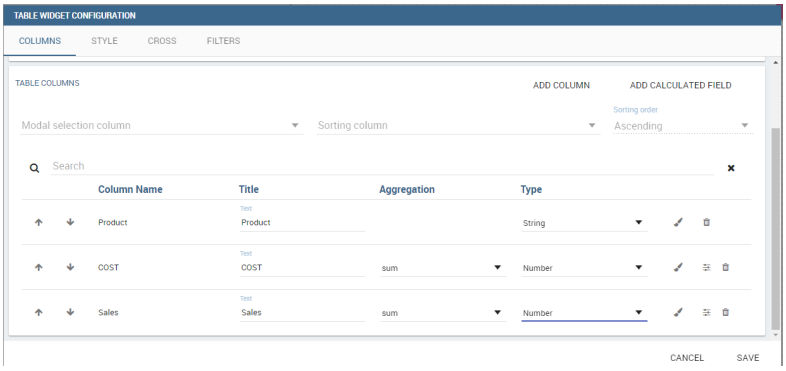


Figure 4.41: Column settings.

- In the **Summary** section you can show the total of the column and customize it using font and style configurations. Refer to Figure 4.42).

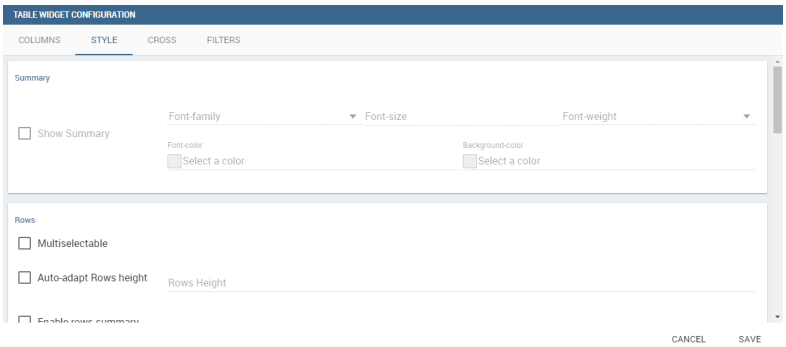


Figure 4.42: Summary section of the Style tab.

- In the **Rows** section you can set the table rows to be adapted in automatic or select a fixed height. You can also show the total of rows. While the multiselectable option allows you to select multiple values and pass them to other cockpit widgets or other external documents. Refer to Figure 4.43.

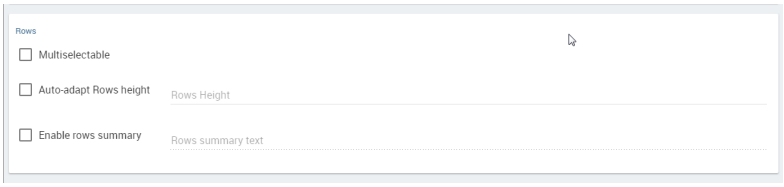


Figure 4.43: Rows section of the Style tab.

- In the **Grid** section you can add borders to the table and add color to alternate rows. In this section you can find different options to customize them. Refer to figure below.

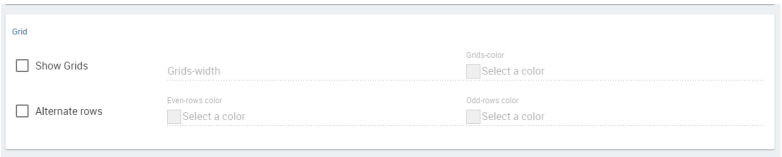


Figure 4.44: Grid section of the Style tab.

- In the **`Header`** style section you find the different options of style for the table header. Refer to Figure 4.45.

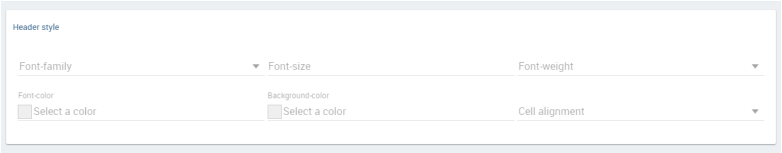


Figure 4.45: Header style section of the Style tab.

- In the **`Borders`** section you can add a border to the widget and customize it by using the colors, thickness and style. Refer to Figure 4.46.

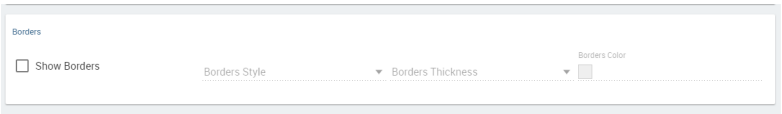


Figure 4.46: Borders section of the Style tab.

- In the **`Titles`** section you can add the titles to the widget and modify the font size and weight. In this section you can also change the height of the widget title. Refer to Figure 4.47.
- In the **`Shadows`** section you can add the shadows in the widget. Refer to Figure 4.48.
- In the **`Background`** color section you can set the background color of the widget. Refer to Figure 4.49.

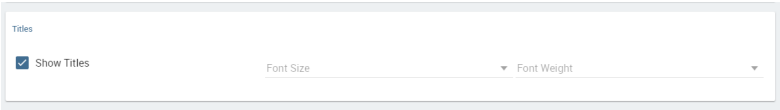


Figure 4.47: Titles section of the Style tab.

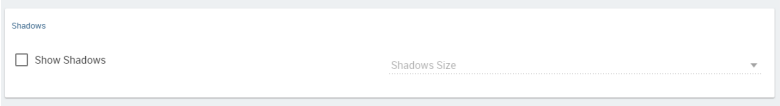


Figure 4.48: Shadows section of the Style tab.

Once the table style settings have been implemented you can switch to the next tab. The "Cross" tab is where the navigation to other documents is defined. It is visible to final users but yet only configurable by a technical user (like an administrator).

Cross navigation only for technical users.

Due to the fact that parameters can only be managed by a technical user the cross navigation cannot be implemented by the final user.

Referring to Figure 4.50, to add a cross navigation to the cockpit you must:

- activate the cross navigation flag;
- activate cross Enable on all row flag, if you want to be able to click on all the columns of the table;
- select the column whose value will be passed through output parameter to the document of arrival;
- select the output parameter that will pass the value to the document of arrival. This parameter type are defined in the document detail of the cockpit;
- select the destination document through the list of cross navigation definition. It is optional. If the Cross navigation is not selected then when you click to launch the cross navigation, a pop up will be open with all the cross navigations defined for that cockpit. If you select the Cross navigation and you click to launch the cross navigation, then it will go to the document of arrival directly.

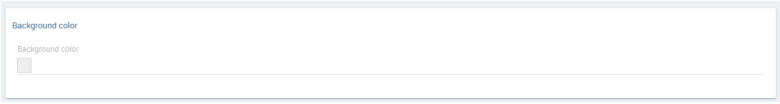


Figure 4.49: Background color section of the Style tab.

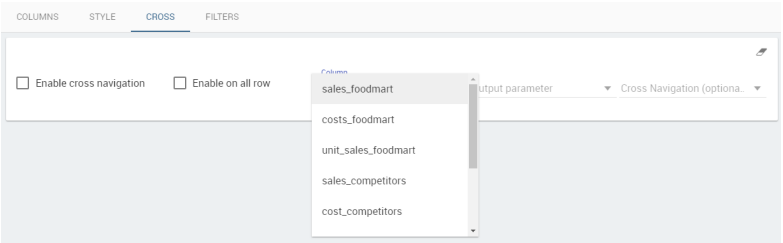


Figure 4.50: Cross tab of the table widget configuration.

Finally, the "Filters" tab is where you can filter the table results by adding a limit to the rows or a conditions in the columns. Figure 4.51 shows an example of how to set the limit rows and the conditions in the columns:

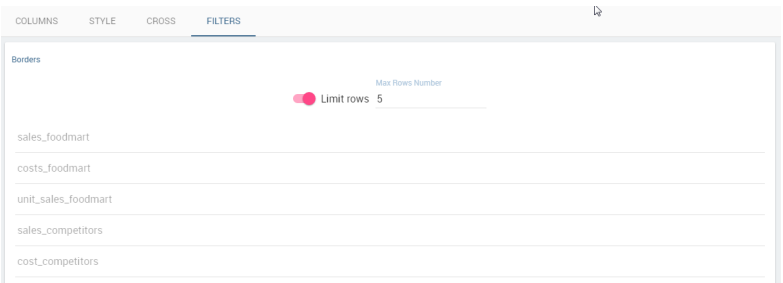


Figure 4.51: Filters tab of the table widget configuration.

Once you have finished setting the differente configuration options of the table widget, then just click on "Save" and your new widget is displayed inside the cockpit.

Pivot Table widget

Similar configurations are available also for the Pivot Table. In this data visualization option, you will have only three side tabs: **`Dataset`** tab, **`Configuration`** tab and **`Style`** tab, as you can see from Figure 4.52.

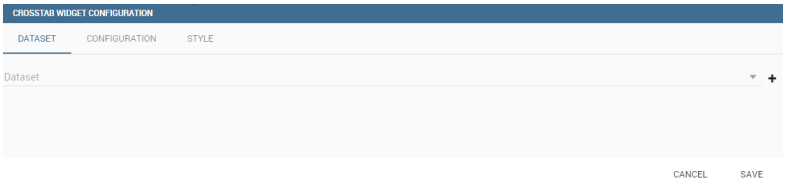


Figure 4.52: Dataset section of the crosstab widget configuration.

Using the "Dataset" tab the user can add the dataset to take values from. Consequently, it is necessary to select the fields you wish to appear as columns, those as row and measures to

be exhibited in the pivot table. See Figure 4.53. Remember to set column and row fields as attributes, while measure fields as numbers.

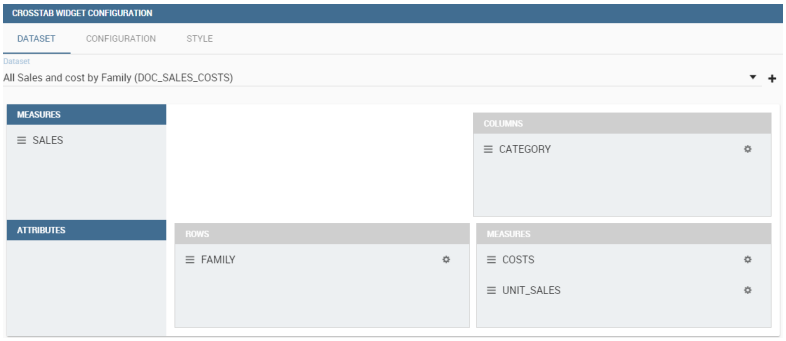


Figure 4.53: Selecting columns, rows and measures of the crosstab.

Once the dataset has been properly configured, you cna proceed to the "Configuration" tab. The latter is made up of three sections: **`General`**, **`On rows`** and **`On columns`**, as Figure 4.54 shows.

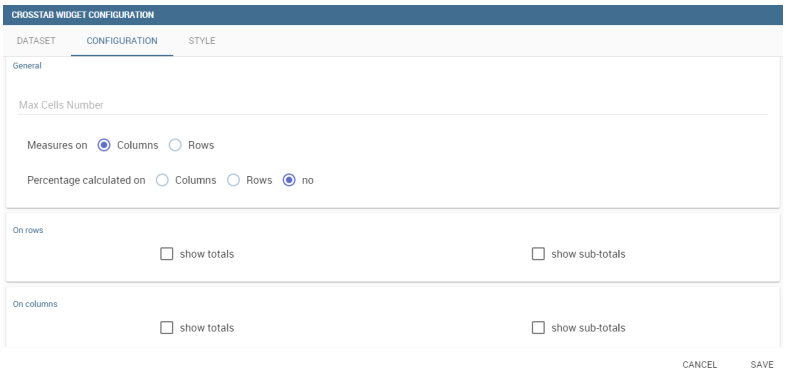


Figure 4.54: Configuration tab interface.

In the "General" section you can set the following features:

- Define the maximum cell number to show;
- decide to hook measures to columns or rows;
- decide to show percentages of measures related to columns or rows.

Thanks to the "On rows" feature, you can easily compute totals or subtotals on rows. Figure 4.55 exhibit an example.

Otherwise, thanks to the "On columns" feature, you can easily compute totals or subtotals on columns. Figure 4.56 exhibit an example.

	gender												
	F						M						Total
	occupation						occupation						Total
	Clerical	Management	Manual	Professional	Skilled Manual	SubTotal	Clerical	Management	Manual	Professional	Skilled Manual	SubTotal	Total
product family	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales
Drink	1,328.00	11,848.00	17,696.00	26,187.00	21,191.00	78,250.00	1,270.00	10,247.00	19,566.00	23,637.00	20,418.00	75,138.00	153,388.00
Food	9,817.00	97,784.00	150,429.00	216,781.00	171,188.00	645,999.00	11,982.00	86,671.00	160,463.00	198,064.00	173,250.00	630,430.00	1,276,429...
Non-Consumable	2,872.00	25,584.00	38,205.00	57,796.00	45,271.00	169,728.00	3,378.00	23,505.00	43,803.00	52,820.00	45,636.00	169,142.00	338,870.00

Figure 4.55: Computing totals and/or subtotals on rows.

		month_of_year											
		the_month	the_month	the_month	the_month	the_month	the_month	the_month	the_month	the_month	the_month	the_month	the_month
		January	February	March	April	May	June	July	August	September	October	November	December
gender	product family	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales
F	Drink	6,008.00	5,931.00	6,586.00	5,950.00	6,194.00	6,468.00	6,890.00	5,926.00	6,907.00	5,941.00	7,440.00	8,009.00
	Food	53,303.00	50,148.00	54,783.00	51,018.00	51,447.00	50,352.00	55,768.00	51,517.00	54,231.00	48,437.00	59,720.00	65,275.00
	Non-Consumable	13,449.00	13,572.00	14,453.00	13,253.00	13,490.00	13,063.00	14,488.00	13,632.00	14,032.00	13,512.00	15,768.00	17,016.00
	SubTotal	72,760.00	69,651.00	75,822.00	70,221.00	71,131.00	69,883.00	77,146.00	71,075.00	75,170.00	67,890.00	82,928.00	90,300.00
M	Drink	6,400.00	6,069.00	6,195.00	5,795.00	5,868.00	6,136.00	5,972.00	5,991.00	6,260.00	5,843.00	7,146.00	7,463.00
	Food	50,846.00	50,357.00	52,360.00	49,271.00	49,477.00	52,678.00	51,837.00	50,604.00	50,471.00	47,932.00	61,425.00	63,172.00
	Non-Consumable	13,953.00	12,838.00	14,216.00	12,749.00	13,546.00	13,980.00	14,576.00	13,570.00	13,639.00	13,408.00	16,089.00	16,578.00
	SubTotal	71,199.00	69,264.00	72,771.00	67,815.00	68,891.00	72,794.00	72,385.00	70,165.00	70,370.00	67,183.00	84,660.00	87,213.00
Total		143,959.00	138,915.00	148,593.00	138,036.00	140,022.00	142,677.00	149,531.00	141,240.00	145,540.00	135,073.00	167,588.00	177,513.00

Figure 4.56: Computing totals and/or subtotals on columns.

Switching to the "Style" tab you can find the general style settings available for the crosstab.

- **`Crosstab Font General Options`** where font and font size are set (Figure 4.57);

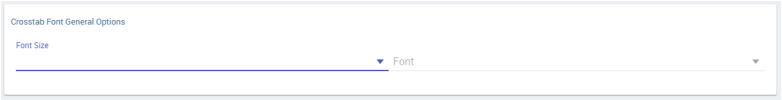


Figure 4.57: General style options for crosstab.

- **`Crosstab Headers Font Options`** where you can configure, as you can see from Figure 4.58, the header style settings as color, background, font, etc.
- **`Measures Font Options`** where you can configure several style options for measures, such as color, background, font size, etc. Figure 4.59 shows the window outlook.

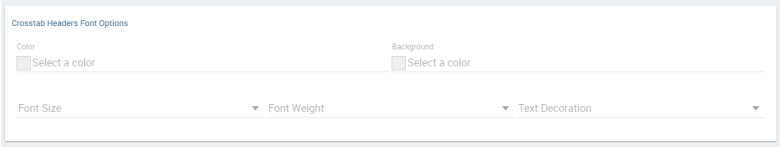


Figure 4.58: Crosstab Headers Font Options for crosstab.

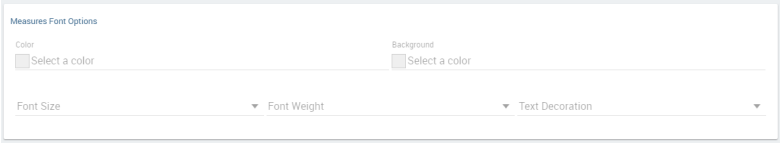


Figure 4.59: Measures Font Options for crosstab.

- Using the **`Grid`** section you can mark (or not) grid borders, decide for border style, thickness and color. You can also alternate row indicating different colors (refer to Figure 4.60).

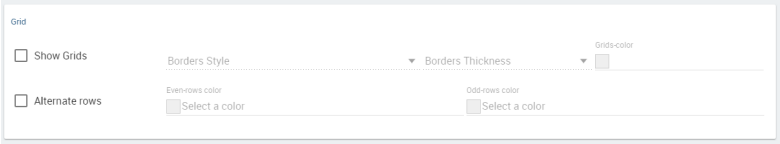


Figure 4.60: Grid Options for crosstab.

- In the **`Measures Headers`** section you cna configure different style option for measure headers, such as color, background, font size, etc. You can refer to Figure 4.61.
- In the **`Total`** section you can set color and background of totals (if any). You can refer to Figure 4.62.
- In the **`Subtotal`** section you can set color and background of subtotals (if any). You can refer to Figure 4.63.
- In the **`Borders`** section you can add borders to widgets and customize them using different styles. You can refer to Figure 4.64 to have a spark on this feature.
- In the **`Titles`** section you can add titles to widget and customize them using different styles. You can refer to Figure 4.65 to have a spark on this feature.
- In the **`Shadows`** section you can add a shadow to widget layout and indicate its measure. You can refer to Figure 4.66 to have a spark on this feature.
- In the **`Background color`** section you can color the widget background at convenience. You can refer to Figure 4.67 to have a spark on this feature.

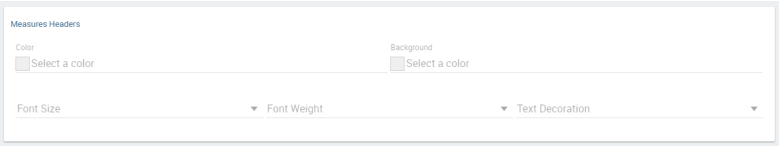


Figure 4.61: Measures Headers Option for crosstab.

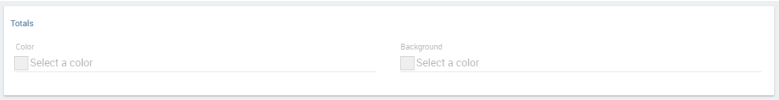


Figure 4.62: Color settings for Totals.

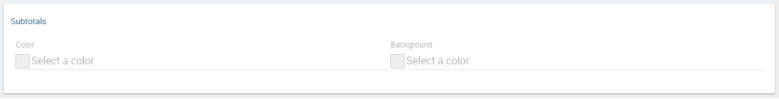


Figure 4.63: Color settings for Subtotals.

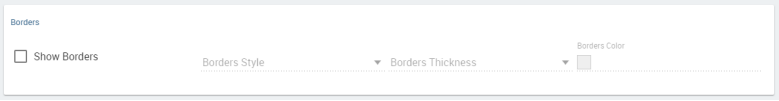


Figure 4.64: Border settings.

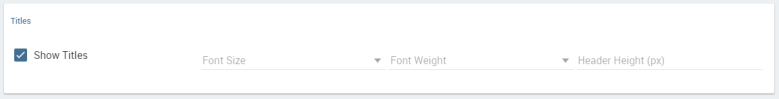


Figure 4.65: Title settings.

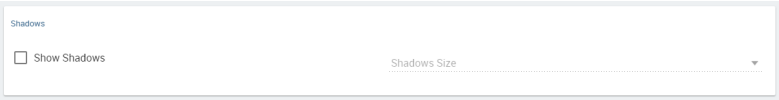


Figure 4.66: Shadow settings.

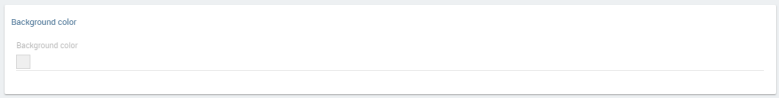


Figure 4.67: Shadow settings.

Once some or all (at least the mandatory) of the above mentioned setting features have been set you can save and the widget will be inserted into the cockpit area.

Document section

The Dcoument widget allows to add an external document into the cockpit area. This widget supports documents like reports, graphs, maps, etc.

The Document Widget configuration is divided into two parts: **`Custom`** tab and **`Style`** tab as you can see from Figure 4.68.

Nel Custom tab is the place where the document is uploaded while the Style tab is where all style options are set.

Figure 4.68: Custom tab of the Document widget.

Selection widget

This widget is related to the association concept so in this subsection we give information on how to activate it while we refer to the dedicated Section 4.8 for details on its functioning.

To enable the Selection widget, which means the possibility to have all associations listed and acccessible on a widget, the user have to simply open the "Selection" feature throught the "Add widget" functionality and configure the demanded options. Figure 4.69 shows the "Selection widget configuration" interface.

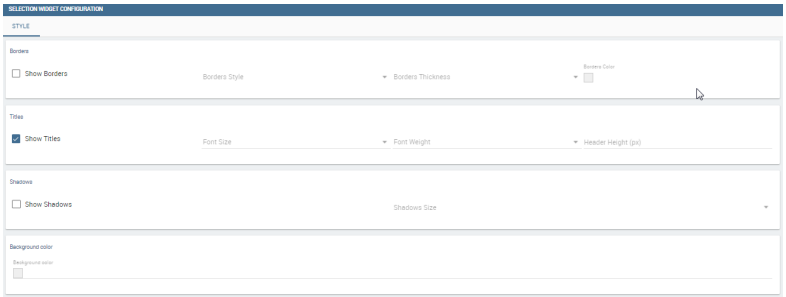


Figure 4.69: Selection widget configuration.

Then click on save and the Selection widget will be inserted into the cockpit area. Figure 4.70 reveals in advance the widget layout.

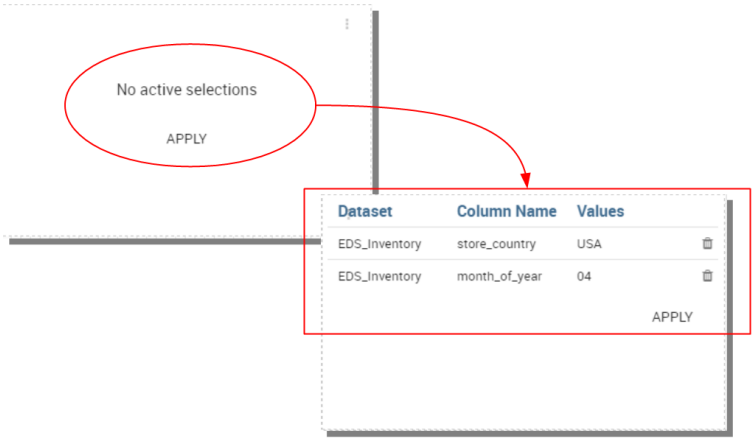


Figure 4.70: Selection widget outlook.

Widget properties

Once one or more (above mentioned) widgets have been implemented, the user has some more options exploring the icon available at the right top corner of the widget itself, as Figure 4.71 highlights.

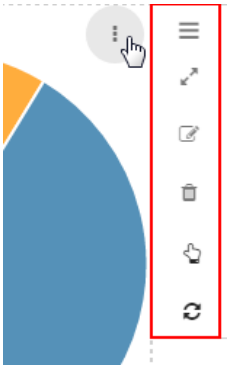


Figure 4.71: Widget properties.

Here the user can:

- move the widget in the cockpit area at convenience;
- modify its dimention;
- delete it;

- activate the on-click interaction of the widget with the other ones;
- activate the updating of widget data due to the interaction with other widgets.

General configuration

This option allows the user to manage all cockpit general settings that we are going to describe through images of the interface. Clicking on the **`General configuration`** button the window in FIgure 4.72 opens. This contains the **`General Settings`** tab and the **`Widget Style`** tab.

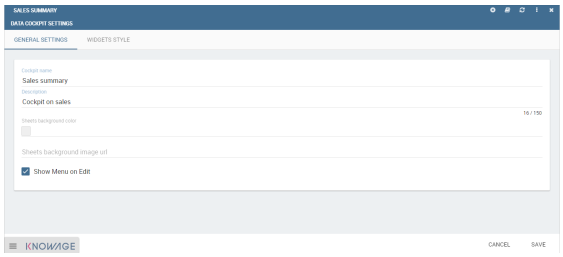


Figure 4.72: General configuration window.

Editing the fields of the first tab you can add or change the name and/or the description of your cockpit; moreover here you can choose the sheet color and decide to enable the menu when the document runs in display mode. The second tab (Figure 4.73 allows to configure some style options of the cockpit, like borders, shadows, titles and background color.

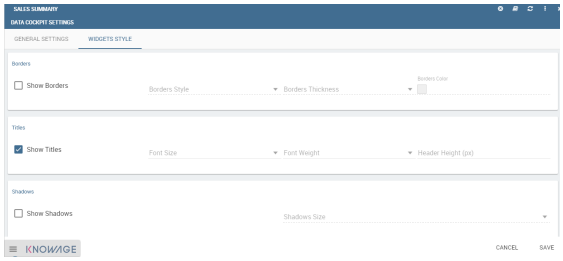


Figure 4.73: Widget style tab.

Data configuration

This feature manages the data storage and usage. In fact, here there is the possiblity to save data in cache, create associations between datasets, schedule the (data) refresh frequency and so on. Referring to Figure 4.74, the feature is implemented through several tabs: the **`Source`** tab, the **`Associations`** tab, the **`Frequency`** and the **`Template`** tab.

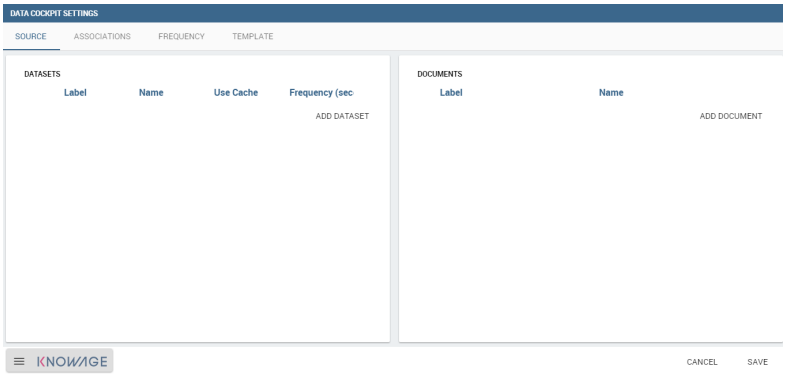


Figure 4.74: Data configuration window.

Source

The Source tab is split into two areas. On the left side the user can find the list of those dataset that are currently used by the cockpit. Here it is possible to add new dataset that will be passed to widgets. In other words, datasets inserted in this area will be listed in the dataset combobox of widgets like the Table, the Pivot Table and the Chart one. Note that the user can delete datasets as well.

Parametric sources management

If the user is adding a parametric dataset the window will exhibit them in an expandable box right below. It is also mandatory to give default values or to associate proper drivers to the document to secure its correct execution. By the way, a final user has no access to parametric dataset and he/she cannot handle analytical drivers, therefore **`parametric sources can be managed only by an admin user`**. We stress that the user must also type the driver name in the field box as highlighted in Figure 4.75. You can type it manually or use the look up just aside the parameter line.

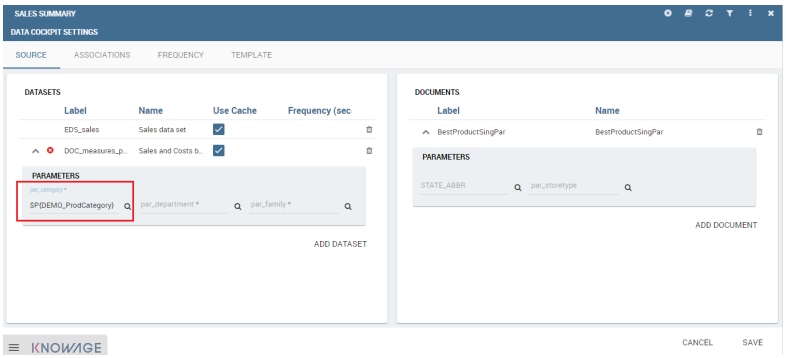


Figure 4.75: Dataset management.

On the right side of the window the user finds the list of external documents that can be added to the cockpit (through Document widgets), or as well as for the dataset case, of documents that are already in use in (previously set) Document widgets. In the occurance of parametric documents, parameter boxes are shown below. Note that it is mandatory to link them to analytical drivers (previuosly hooked to the document) or be assigned a fixed (default) value.

Associations

If your goal is to show data from a single dataset, it is not necessary to define any association. *Associations should be set within the designer when widgets are built on different datasets.* Associations can be set with the elements: dataset columns, dataset parameters and document parameters. Note that to implement an association the user must have at least one column. We show some examples in the following.

Figure 4.76 shows the association between two datasets. In this case the user must detect one field from the first dataset, the same field (in terms of values) in the other one. The relation will appear right below. Click on the save button to confirm the association. If the associations relie on multiple columns the user must add them one by one.

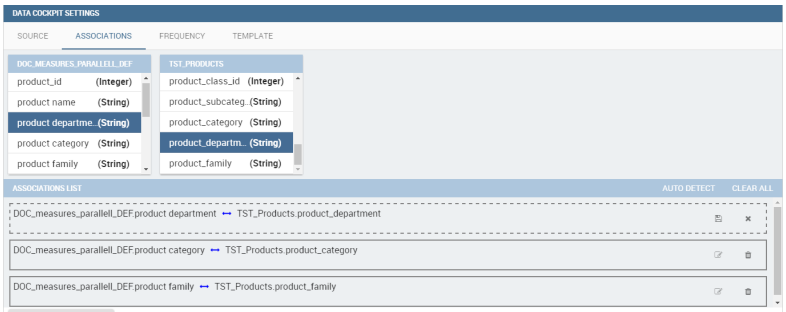


Figure 4.76: Associations between dataset columns.

The same procedure can be done in the case of dataset columns and dataset parameters, as shown in Figure 4.77.

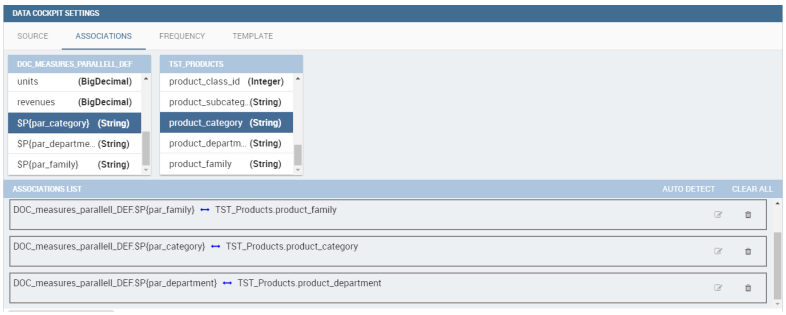


Figure 4.77: Associations between dataset column and dataset parameter.

Another example is furnished in Figure 4.78. Here the association is perfomed between a dataset column and document parameter.

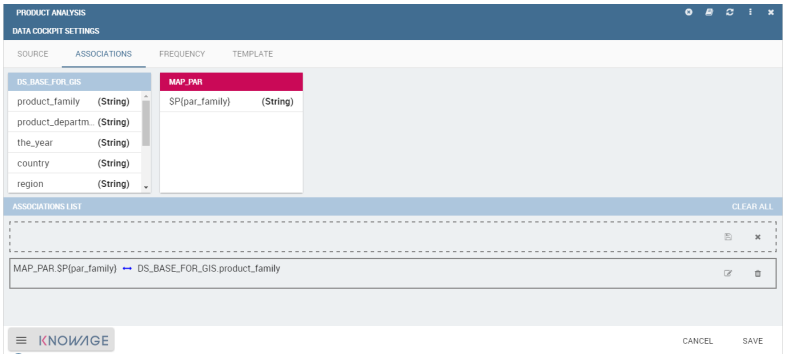


Figure 4.78: Associations between dataset column and document parameter.

Once you have defined the associations, as soon as you refresh one widget, all related widgets are refreshed simultaneously while data update.

Template

In this tab the user can find the json code (at the current stage of the work) which composes the template. Figure 4.79 shows an example.

Figure 4.79: Template example.

Selections

Adding the **`Selections`** to your widgets, namely the possibility to reload all widget data according to selection made throught the click on a specific item of the cockpit (cell value, chart bar, etc.), you can check which selections are active on your cockpit at anytime thanks to the **`Selection`** functionality. In Section 4.8 we already described how to add the "Selection" widget inside the cockpit area. If the user do not wish for the widget to stay visible, selections can still be accessed and managed through the menu configuration bar. Clicking on the "Selection" menu icon you can enter the "Selections" window. Here all selections and associations are listed, as shown in Figure 4.80. The "Delete" button is available just aside each row to let the

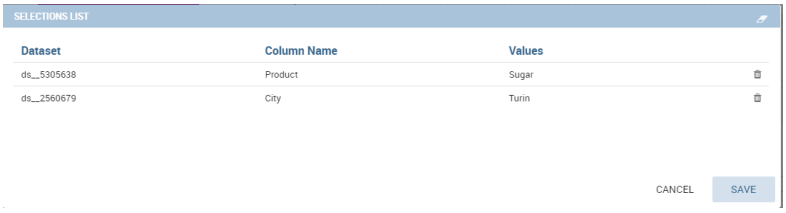


Figure 4.80: Selection window.

user to remove that specific selections. Click on the "Cancel" button to exit the window.

Clear cache

The **`Clear cache`** button lets you update the data shown in your widget to the ones in your database. When you create your widget and associate your datafields, a photo of data is made and stored in temporary tables. This means that your cockpit will display the same data at each execution until you clean the chace by clicking on the dedicated button and execute the document again. Now your data are refreshed and updated to the one contained in your database at last execution time. As discussed before this button is available also in "Read only" modality.

Save

You can save the cockpit by clicking on the save button in the right-top corner. The document will be saved in the personal folder (technical users) or in the **`My Analysis`** section.

We remember that it is possible to share the new cockpit with other users clicking on the dedicated icon. You can also choose in which folder, among the ones visible to your role, to place your shared document.

4.9 My first Report

Knowage Report Designer is the development environment based on Eclipse. It allows the developer to design and modify static reports. This module supports the developer while designing reports as well as during the installation and testing processes, directly on Knowage Server.

To install Knowage Report Designer, first check that your environment satisfies the following prerequisites:

- it has a JDK 1.7.x already installed;
- it has the `JAVA_HOME` variable already set;

- it has a certified operative system (Windows, Linux, Unix are generally accepted). The list of certified environments mainly depends on the ones supported by Eclipse. Please, refer to the Knowage Studio release notes to find out the Eclipse version included in each released package.

If so, the second step is to download the suitable package from the OW2 forge: http://forge.ow2.org/projects/knowage.

At this point, you just have to unzip the downloaded package under a folder, having a `KnowageReportDesigner_<version>_<distribution>_<date>` subfolder. Here you can find and run a **`KnowageReportDesigner.exe`** or **`Knowage.sh`** script to start and select the preferred workspace.

Connection between Server and Studio

Once Knowage Studio has been configured, connect it to Knowage Server. This will be the deployment environment to provide end-users with all certified objects. The main steps to set the connection are:

- select the Knowage perspective;
- create a Knowage project;
- set Knowage Server connections.

These points will be described in the following.

At first execution of Knowage Studio, a welcome page appears.

The first step is to create a Knowage project by selecting the Knowage icon of Figure 4.81 from the toolbar located at the top of the page.

Figure 4.81: New Knowage project icon.

Once the project has been created, a standard folder tree appears. The predefined folders are:

Business Analysis: it contains all developed documents (reports, charts, cockpits, etc.) that can be uploaded into or downloaded from Knowage Server. This folder can be structured with subfolders, to freely organize ones own documents;

Private folders: they contain the users personal or project documentation. This folder can be freely managed by the user;

Resources: they contain all technical resources used in the project (i.e., the reference to a Knowage Server).

Now you can configure references to one or more Knowage Servers. In other words, each user can work for many projects that can be:

- hosted on different servers
- hosted on the same server with different user accounts.

To define a server connection, click on the **`New Server`** item of the **`Server`** contextual menu.

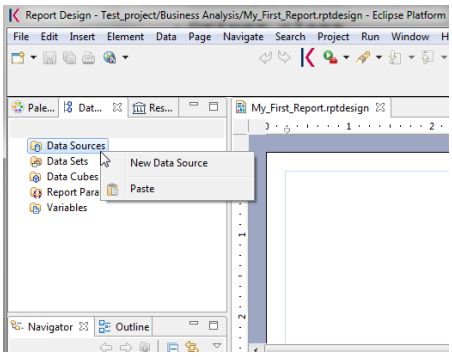


Figure 4.82: New Data Source connection.

A form will ask you for:

- **Server name:** a logical name to identify the server. The name is used on the local workspace only and has no relation with the physical one.
- **Url:** the http url where the server is hosted and reachable.
- **User:** the user who authenticates on Knowage Server, setting his access rights in terms of what kind of operations he can do (upload and download a model or a data set) and what parts of the Server repository he can access.
- **Password:** the users password.
- **Active:** a flag that indicates the active server. It is particularly useful when the user is working with multiple servers. The active server indicates that every upload and download operation refers to this Knowage Server instance.

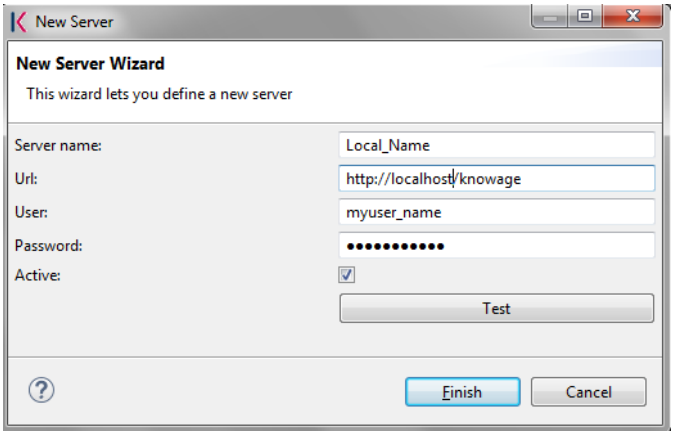


Figure 4.83: Server configuration wizard.

Connection to Knowage Server

If something in your network configuration has been changed from your first run of Knowage Studio, the connection test of Knowage Studio to the Server could fail. Most often this problem is due to the proxy settings in your Eclipse environment. If this is not the case, try to run Knowage Studio from the command line with the clean option (**`Knowage.exe`** clean) to reset working settings.

At this point, Knowage Studio is ready to work!

Metadata definition

Each Knowage document (e.g., report, olap, chart, cockpit, etc.), has its own technical metadata stored in Knowage internal repository. The most relevant technical metadata describing document structure, content and behaviour are:

- *Template*, which defines the document layout;
- *Data set*, which defines how data of each document should be read;
- *Analytical drivers*, which hook the template parameters to the graphical interface (at runtime), managing also the right form for parameters.

Data set definition

Each document type has its own way to define how to get data from an internal data source, accordingly to a data set definition. This allows the document to directly access the RDBMS, through the SQL loading script, which can be encoded within the template or externally (i.e., stored as Knowage Server resource), but without any abstraction from data sources.

Developing a BIRT report

To create a new document right-click on the **`Business Analysis`** folder and, to start, choose between report and dashboard. In Figure 4.84 we will choose **`Report with Birt`** and leave the other option to the next chapter.

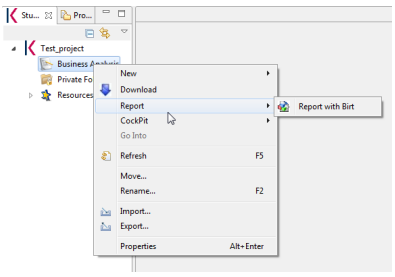


Figure 4.84: New document creation.

Once the document is designed, it is stored as a local file, marked out with an icon and a specific file extension:

- **`.sbidoccomp:`** document templates for dashboard that use the ComposedDocument engine;
- **`.rptdesign:`** document template for reports that use the BIRT engine.

In our case, we will get a .rptdesign file. A double click on one of these files allows to open the document template, with its related graphical editor.

The design and deployment of a BIRT report includes the following steps:

- create the empty document;
- switch to the report designer perspective;
- create the data source;
- create the dataset;
- design the report via the graphical interface;
- deploy the report on the server.

To create a new BIRT report, as just anticipated, right click on the **Business Analysis** folder and select **Report** > **Report with BIRT**. This will open an editor where you can choose the name of your document. The new document will be created under the **Business Analysis** folder.

Double click on it to open the editor. At this point, you are still working in the Knowage perspective. To design the report, switch to the actual BIRT designer perspective. Click on the perspective icon of the Eclipse editor and select the Report Designer among the available perspectives, as showed in Figure 4.85.

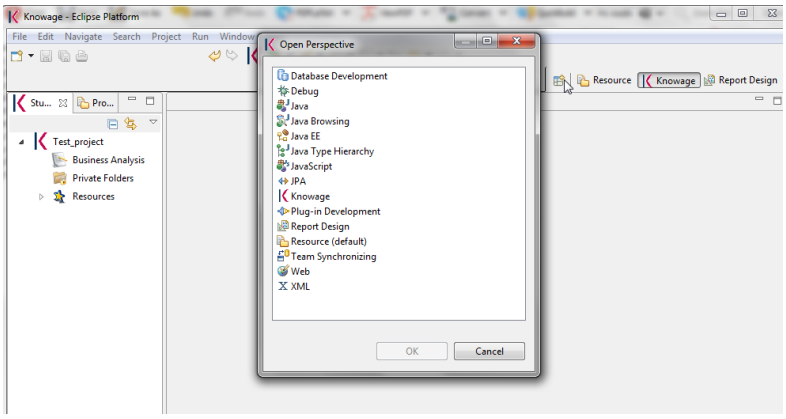


Figure 4.85: Change perspective.

The next steps are the creation of a datasource and of a dataset. As previously described in the section Dataset Definition, Knowage Studio allows the development of analytical documents using either internal or external datasets. In this specific example, we will show how to create a report with an internal dataset. First of all, in case of an internal dataset, define a **JDBC Data Source**. Right click on the **Data Source** item and select the corresponding data source. A pop up editor will open, prompting you the connection settings:

- **Driver class**
- **Database URL**
- **Username** and **password**

Note that these configuration parameters will be used by the Studio to connect to the database and let the report to be executed locally (i.e., within the Studio). Make sure that the database set in the Server share the same schema of that defined in the Studio.

Since you are setting a local reference to a database inside the report, remember to set an additional information: this will enable Knowage Server to correctly execute the report, by connecting to the data source referenced within the server and not inside the report. Basically you need to tell the server to override the data source configuration. Therefore, add a parameter to the report, called `connectionName`, right-clicking on the "Report Parameters" menu item and selecting "New Parameter". Fill in the form as suggested in Figure 4.86.

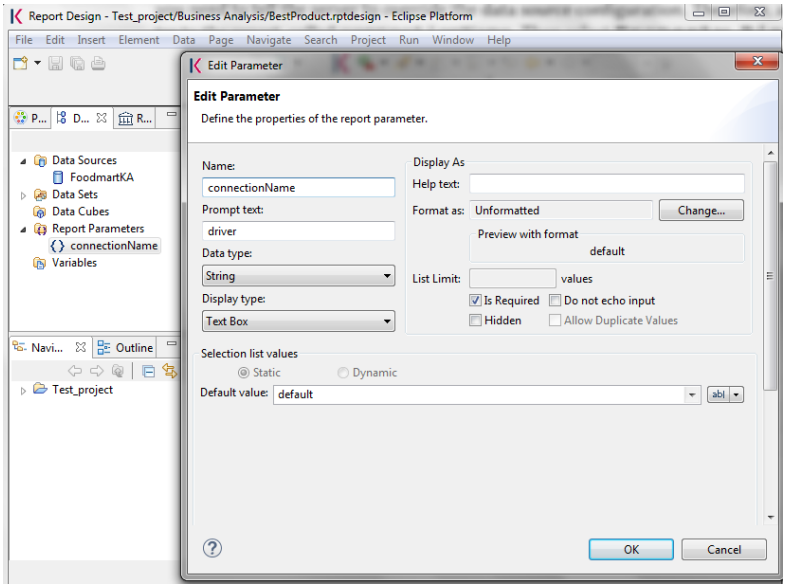


Figure 4.86: Adding connectionName Parameter.

Then go to **`Property Binding`** in the Data Source editor and set the property `JNDI URL` to the value of the `connectionName` parameter, as shown in Figure 4.87.

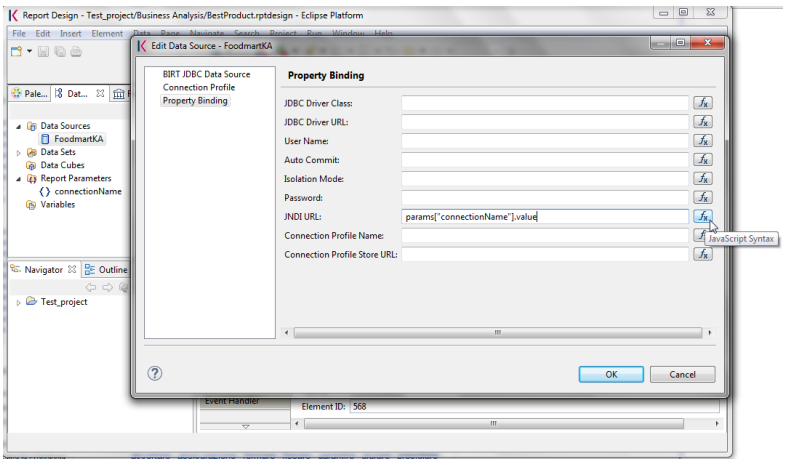


Figure 4.87: Setting the connectionName parameter in the Data Source editor

JNDI URL

Do not forget to define the connectionName parameter in your BIRT report and set the JNDI URL accordingly. Without these settings your BIRT report may be unable to access data once it is deployed on the server. In addition, if database and connection properties change, you need to change the connection properties only in Knowage server.

Once the data source has been configured, you can proceed with the creation of a dataset. Therefore, right-click on the **`Data Set`** item and select **`New Data Set`**. In the next window, select the data source, the type of query and give a name to the dataset, as exhibited in Figure 4.88. The scope of this name is limited to your report, because we are defining an internal dataset.

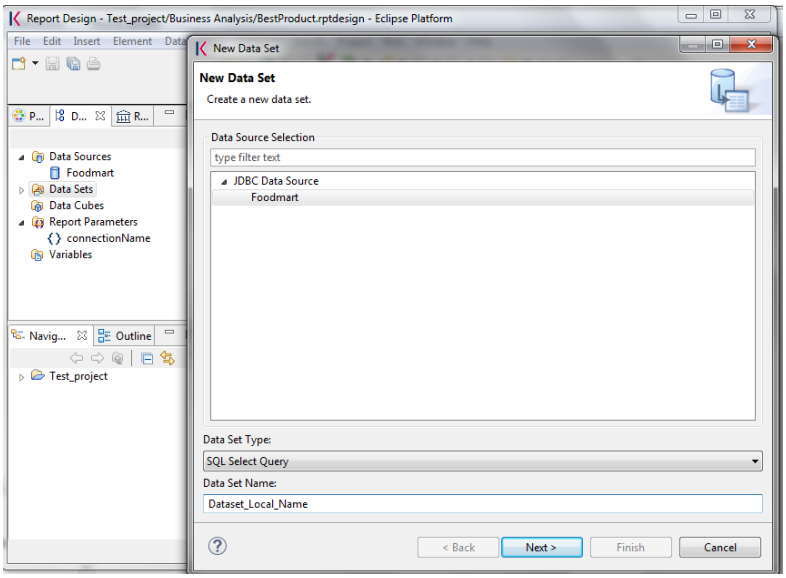


Figure 4.88: Dataset definition.

Now you can define your dataset by writing the SQL query in the editor and testing the results (see Figure 4.89). At any time, you can modify the dataset by clicking on it, which will re-open the query editor.

Let us design a very simple report, which contains a table showing the data from the defined dataset. The easiest way to create a table from a dataset is to drag & drop the dataset from the tree menu into the editor area.

The most generic way, which applies to all graphical elements, consists in switching to the **`Palette`** menu on the left panel, keeping the designer in the central panel. Drag and drop the

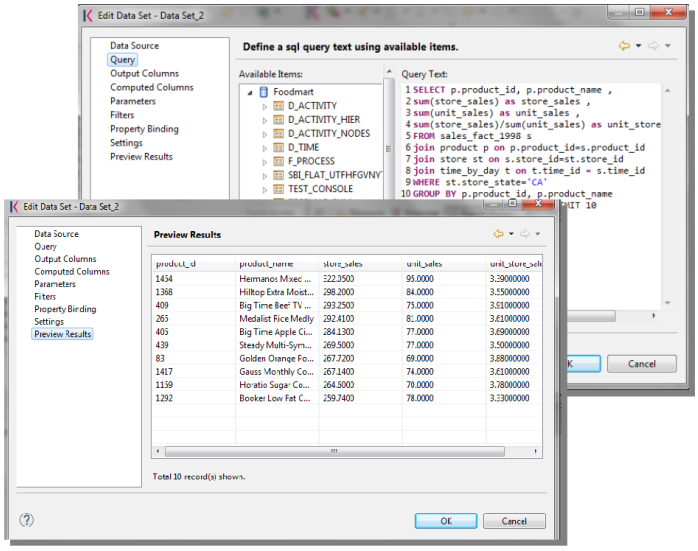


Figure 4.89: Dataset editor, with preview.

table into the editor area. Consider that this can be done with all other elements listed in the Palette. At this point, you can edit the table (as well as any other graphical element on the report) using the **`Property Editor`** tab below the editor area.

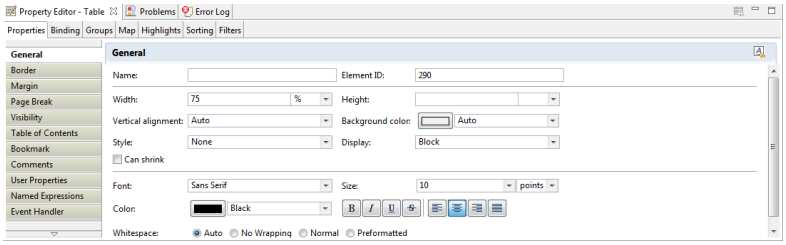


Figure 4.90: BIRT Property Editor.

While developing a report, it is particularly useful to test it regularly. To this end, click on the **`Preview`** tab below the editor area. To revert back to the editor, just click on the **`Layout`** tab. In the **`Master Page`** tab, you can set the dimensions and layout of the report; the **`Script`** tab supports advanced scripting functionalities; finally, the **`XML Source`** tab shows the editable source code of your report.

While developing a report, it is particularly useful to test it regularly. To this end, click on the Preview tab below the editor area. To revert back to the editor, just click on the Layout tab. In the Master Page tab, you can set the dimensions and layout of the report; the Script tab supports advanced scripting functionalities; finally, the XML Source tab shows the editable source code of your report.

Once your report is done, you can deploy it on Knowage Server.

Deploy on Knowage Server

Please refer to the section Download and Deploy in this chapter to find out more on report deployment.

The BIRT report designer allows the creation of complex reports, with different graphical elements such as cross tabs, charts, images and different text areas. In this section we do not provide any details on graphical development but we focus on specific aspects of Knowage BIRT Report Engine.

BIRT Designer

For a detailed explanation of report design, please refer to BIRT documentation at www.eclipse.org/birt/

Using an external Data Set

In the afore-described example, we built a report using an internal dataset, i.e., a dataset defined within the report. This has two main implications. First, the dataset is not visible outside the report execution: for example, it cannot be directly reused by other reports. Second, an internal dataset is always defined as a SQL query and it cannot take advantage of Knowage business model abstraction. For these reasons, Knowage allows the definition of external datasets in reports. An external dataset is defined in Knowage Server and, as a consequence, it is visible to all documents on the server (i.e., it can be used by any of them, if properly linked to the document). External datasets can either be SQL datasets or QbE datasets, that is, datasets defined by queries over a business model.

An external dataset can be included into any BIRT report by downloading it from a Knowage Server. Specifically:

- define a Knowage Server datasource;
- download a dataset from the Knowage Server datasource.

We always start by right-clicking on the **`Data Source`** item. Select **`Knowage Server Data Source`** and set the appropriate input configuration:

- **`Server URL`**
- **`Username`** and **`password`** used to log into the Server (e.g., biadmin).

After filling in the configuration fields, test the connection and save it. The new data source will appear in the left tree menu. Instead of connecting to a database via a JDBC driver, connect

to the server as the source of data. Obviously, the actual data source and dataset must have previously been defined on the Server.

To select the dataset, click on **`New Data Set`** as above, but this time select the **`Knowage Data Source`** that you have just defined. Now, instead of choosing a new name for the dataset, insert the correct label of the dataset that you want to import from the Server. If the label is correct, the dataset will be imported in the report by clicking on **`Finish`**. Notice that the imported dataset may be a SQL or a QbE one. Since both types of datasets are stored in the same repository by Knowage Server, we are enabled to use any BM query in the development of a report.

Use of BM queries in report development.

The ideal use of a business model is to define queries over the BM via Knowage Meta, deploy them on Knowage Server and reuse them on Knowage Studio as external datasets.

Adding parameters to reports

Most times reports show data analysis that depend on variable parameters, such as time, place, type. Knowage Studio allows the designer to add parameters to a report and link them to analytical drivers defined in Knowage Server.

To use these parameters, you first need to add them to your report. Right-click on **`Report Parameters`** in the tree panel and select **`New Parameter`**. Here you can set the data type and choose a name for your parameter.

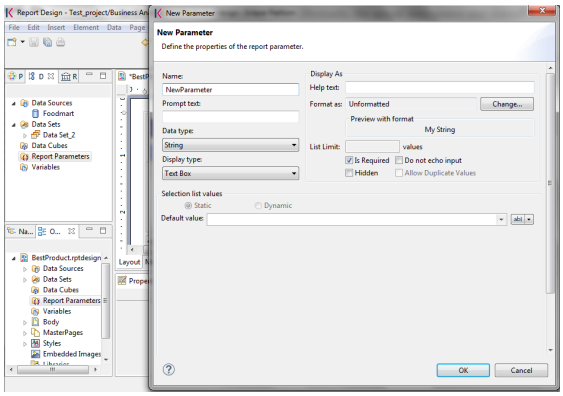


Figure 4.91: Creation of a new parameter in a BIRT report.

Parameters URI

Be careful when assigning a name to a parameter inside a report. This name must correspond to the parameters URI when you deploy the document on Knowage Server.

Once you have defined all parameters, open the (or create a new) dataset. Parameters are identified by a question mark **?** . For each **?** that you insert in your query, you must set the corresponding link in the **Parameters** tab: this will allow parameters substitution at report execution time. Note that you must set a link for each question mark as shown in Figure 4.92, even if the same parameter occurs multiple times in the same query.

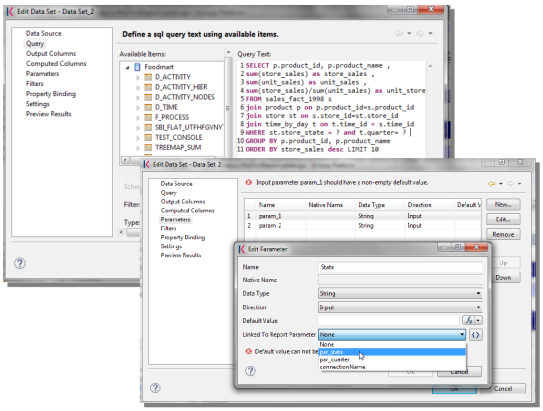


Figure 4.92: Insert parameters into the dataset definition.

Parameters can also be used within some graphical elements, such as dynamic text, with the following syntax:

```
params[name_of_parameter].value
```

Code 4.1: Parameters syntax

Transfer reports from Studio to Server and vice versa

We saw that developers can use Knowage Studio deployment service to easily register the report with its template on Knowage Server. Alternatively, any valid BIRT template (developed with or without Knowage Studio) can be directly uploaded in Knowage Server using the web interface for document management.

Download and deploy

To modify an already deployed document, first download the related template from the Knowage Server repository.

Right-click on the **Business Analysis** folder or on one of its subfolders. In the contextual menu, select the **Download** option. At this point, the functionality tree appears, allowing you to choose the documents to be downloaded.

These documents will be available in the local folder that you have previously selected. Document details (i.e., label, description, state, engine and parameters) are stored as metadata in the local repository. Metadata can be refreshed from the Server by clicking on the **Refresh** button in the **Knowage** > **Document Metadata** tab of the **Properties** section. To open Properties, right-click on the document item and select **Properties**.

In a similar way, after a document update, the Deploy option of the same menu sends the new template to the Server, ready for use.

Another possible situation is when the designer creates a new template from scratch and deploys it on the Server. At first deploy, a link between the template and a document on the Server is created. It will last until the document on the Server is deleted or its label is modified. In those cases, you will need to re-deploy the template from the Studio.

To deploy a template, right-click and select **Deploy**. You will be prompted a form for basic metadata on the new document. Required and/or pre-filled input data may change according to the document type. However, they usually include:

- **Label:** free label as short code;
- **Name:** name of the document;
- **Description:** long description;
- **Type:** document type (report, chart, cockpit, etc.);
- **Data Set:** the already deployed data set for documents that use external ones;
- **Data source:** the reference to the data source that will be used on Knowage Server for documents that have an internal data set, in order to work with official source instead of local or working RDBMS;
- **State:** the initial state of the document (development, test, released, suspended) according to their life cycle management policy;
- **Refresh seconds:** the automatic refresh time;
- **Position:** the folder in the remote Knowage Server repository where documents are deployed, indirectly setting who can use it and its first authorization level.

Analytical documents

The described form sets basic metadata, generally managed as technical metadata on Knowage Server.

These document details are stored as metadata in the local repository and used to register it in the central repository of the Server as well. To look at their local values, select the **Properties** item from the document contextual menu and choose **Knowage**.

Directly from there, local metadata can be refreshed anytime on the active server, by simply pressing the **Refresh Metadata on active server** button.

Cross Navigation for BIRT Reports

A powerful feature of Knowage analytical documents is cross-navigation, i.e., the ability to navigate documents in a browser-like fashion following logical data flows. Although cross-navigation is uniformly provided on all documents executed in Knowage Server, each type of document has its own modality to set the link pointing to another document.

Notice that the pointer can reference any Knowage document, regardless of the source document. For example, a BIRT report can point to a chart, a console, a geo or any other analytical document.

In Knowage there are two main typologies of cross navigation: *internal* and *external*.

Internal cross navigation updates one or more areas of a document by clicking on a series, a text, an image or — in general — on a selected element of the document.

External cross navigation opens another document by clicking on an element of the main document, allowing in this way the definition of a "navigation path" throughout analytical documents (usually, from very general and aggregated information down to the more detailed and specific information)). Indeed, you can add cross navigation also to a document reached by cross navigation. This can be helpful to go deeper into an analysis, since each cross navigation step could be a deeper visualization of the data displayed in the starting document.

It is obviously possible to associate more than one cross navigation to a single document. It means that by clicking on different elements of the same document the user can be directed to different documents.

To allow the external cross-navigation in a BIRT report, you need to add a hyperlink to the element you want to be clickable using the **Properties** tab of the Knowage Studio. Most report elements can host a hyperlink. For example, let us add a hyperlink to a cell in the table.

Click on the table cell and select the **Hyperlink** item in the **Properties** tab. By clicking on Edit, the hyperlink editor will open and show three input fields:

- **`Location:`** write here the URI,
- **`Target:`** select Self,
- **`Tool Tip.`** write the text you wish to appear on the link,

as showed in the following Figure 4.93.

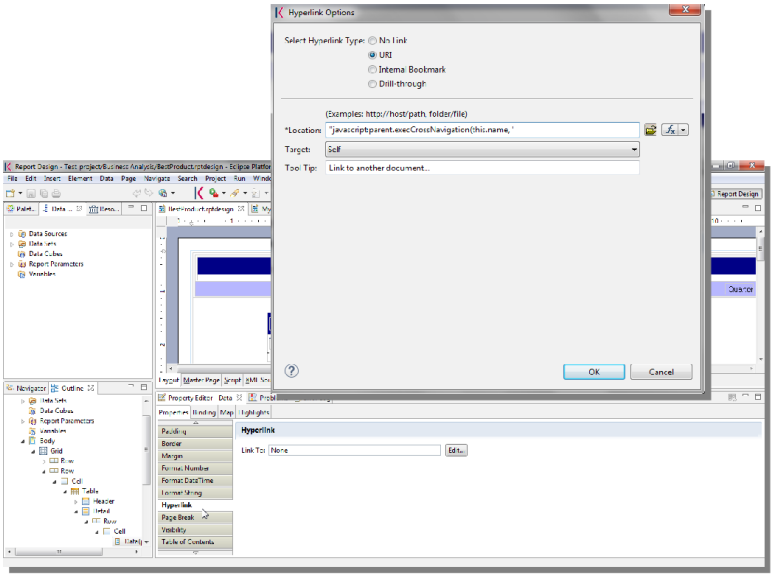


Figure 4.93: Hyperlink editor.

To edit the Location, click on the right drop down button and select the JavaScript syntax. This will open BIRT JavaScript editor. Here you must write down the javascript function "`javascript:parent.execExternalCrossNavigation`" passing JSON arguments like ParName: string, null and string.

In Code 4.2 we give an idea of how the syntax should be like:

```
javascript:parent.execExternalCrossNavigation(+
{OUT_PAR:'+params["par_period].value+"'"+
,'OUT_STRING:'+string_text+"'"+
,'OUT_NUM:+numberX+
,'OUT_ManualSTRING:'foo'"+
,'OUT_ARRAY:['A','B','5']}"+
,null,+
'Cross_Navigation_Name');

```

Code 4.2: Cross Navigation syntax

Type the right cross navigation name

It is important to underline that the "`Cross_Navigation_Name`" of Code 4.2 is the cross navigation name related to the document and set using the "Cross Navigation Definition" feature we described in Section 4.6.

It will be necessary to type the right cross navigation name related to the document as defined using the "Tool" settings of Knowage server and to define those parameters (`OUT_PAR`, `OUT_STRING`, etc.) as output parameters in the deployed document on the Server (see Section 4.6).

Note that the syntax of the string is fixed, while you need to assign values to the parameters that will be passed to the destination document. The JavaScript editor helps you to insert dataset column bindings, as shown in Figure 4.94, and report parameters automatically.

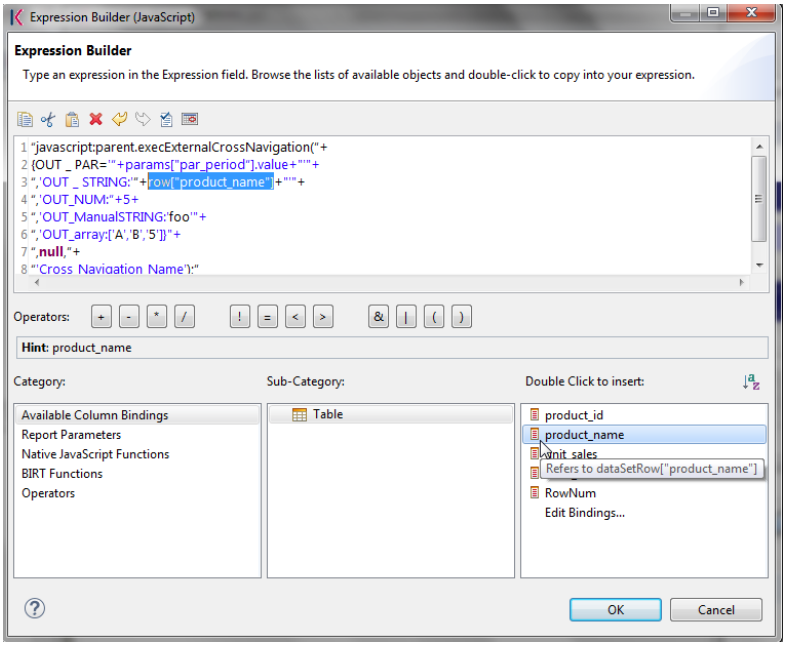


Figure 4.94: Column bindings.

To manage multi-value parameters is enough to list all values between brackets separating them with commas, as reported in Code 4.2. More specifically, the array must contain values of the same type. For example:

```
OUT_SeveralNames:['Michael','Paul','Sophia']
```

or

```
OUT_SeveralNames:[5,9,31938].

```

Finally, it is possible to set a sort of "multi"-cross navigation if for example the exit document is related to more than one document through the Cross Navigation Definition. Let suppose that the source document goes to a target document and the name of the navigation is "CrossNav1" and simultaneously the source document goes to a second target document and the name of the navigation is "CrossNav2". If in the JavaScript function of Code 4.2 the "`Cross_Navigation_Name`" is left empty as in Code 4.3, when the user clicks on the object for which the navigation has been enabled a pop up opens asking for the user to choose between the "CrossNav1" navigation or the "CrossNav2" one. This procedure allows the user to have a more than one possible navigation starting from the same object.

```
javascript:parent.execExternalCrossNavigation(+
{OUT_PAR:'+params["par_period].value+"'"+
,'OUT_STRING:'+string_text+"'"+
,'OUT_NUM:+numberX+
,'OUT_ManualSTRING:'foo'"+
,'OUT_ARRAY:['A','B','5']}"+
,null,+
');

```

Code 4.3: Cross Navigation syntax

4.10 My first OLAP

Multidimensional analysis allows the hierarchical inquiry of numerical measures over predefined dimensions. In this section we want to go into details of how a technical user can create an OLAP document.

Development of an OLAP document

The creation of an OLAP analytical document requires the following steps:

1. schema modelling;
2. catalogue configuration;
3. OLAP cube template building;
4. analytical document creation.

Schema modelling

Mondrian is a ROLAP tool. As such, it maps OLAP structures, such as cubes, dimensions and attributes directly on tables and columns of a relational data base via XML-based files, called Mondrian schemas. Mondrian schemas are treated by Knowage as resources and organized into catalogues. Hereafter, an example of Mondrian schema in Code 4.4:

```
 <?xml version="1.0"?>
<Schema name="FoodMart">

<!-- Shared dimensions -->
<Dimension name="Customers">
  <Hierarchy hasAll="true" allMemberName="All Customers" primaryKey="
    customer_id">
    <Table name="customer"/>
    <Level name="Country" column="country" uniqueMembers="true"/>
    <Level name="State Province" column="state_province" uniqueMembers="true"/>
    <Level name="City" column="city" uniqueMembers="false"/>
  </Hierarchy>
  ...
</Dimension>
...
<!-- Cubes -->
<Cube name="Sales">
  <Table name="sales_fact_1998"/>
  <DimensionUsage name="Customers" source="Customers" foreignKey="customer_id"
    />
  ...
  <!-- Private dimensions -->
  <Dimension name="Promotion Media" foreignKey="promotion_id">
    <Hierarchy hasAll="true" allMemberName="All Media" primaryKey="promotion_id
    ">
      <Table name="promotion"/>
      <Level name="Media Type" column="media_type" uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>
  ...
  <!-- basic measures-->
  <Measure name="Unit Sales" column="unit_sales" aggregator="sum" formatString=
    "#,###.00"/>
  <Measure name="Store Cost" column="store_cost" aggregator="sum" formatString=
    "#,###.00"/>
```

```
31    <Measure name="Store Sales" column="store_sales" aggregator="sum"
        formatString="#,###.00"/>
32    ...
33    <!-- derived measures-->
34    <CalculatedMember name="Profit" dimension="Measures">
35      <Formula>
36        [Measures].[Store Sales] - [Measures].[Store Cost]
37      </Formula>
38      <CalculatedMemberProperty name="format_string" value="$#,##0.00"/>
39    </CalculatedMember>
40  </Cube>
41  ...
42  </Schema>
```

Code 4.4: Mondrian schema example

Each mapping file contains one schema only, as well as multiple dimensions and cubes. Cubes include multiple dimensions and measures. Dimensions include multiple hierarchies and levels. Measures can be either primitive, i.e., bound to single columns of the fact table, or calculated, i.e., derived from calculation formulas that are defined in the schema. The schema also contains links between the elements of the OLAP model and the entities of the physical model: for example, <table> sets a link between a cube and its dimensions, while the attributes primaryKey and foreignKey reference integrity constraints of the star schema.

Mondrian

For a detailed explanation of Mondrian schemas, please refer to the documentation available at the official project webpage: http://mondrian.pentaho.com/

Engine catalogue configuration

To reference an OLAP cube, first insert the corresponding Mondrian schema into the catalogue of schemas managed by the engine. In order to do this, go to **Catalogs > Mondrian schemas catalog**. Here you can define the new schema uploading you XML schema file and choosing **Name** and **Description**. When creating a new OLAP template, you will choose among the available cubes defined in the registered schemas.

Note that the **Lock** option forbids other technical users to modify settings.

OLAP document development

To create a new OLAP document, create a new document in the **Document Development** area and select **Online analytical processing** as **Type**. Then you can choose the available engines. In this case we have only the **OLAP engine**.

Once the cube has been created, you need to build a template which maps the cube to the analytical document. Knowage Server is endowed of an efficient OLAP designer which avoid the user to edit manually the XML-based template. We will therefore describe here all features of this functionality.

The user needs to have a functioning Modrian schema to start the work with. Select **Mondrian Schemas Catalog** to check the available Mondrian schemas on server. The page as the one in Figure 4.95 will open.

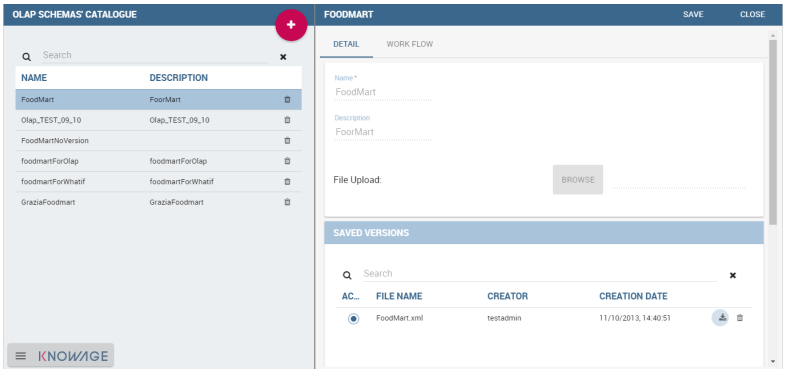


Figure 4.95: Schema Mondrian from catalog.

Then we start entering the **Document Browser** and clicking on the "Plus" icon at the top right corner of the page. Fill in the mandatory boxes as Label and Name of the document, select the On-line Analytica Process Type of document and the What-if Engine (we stress that the What-if engine is available only for who have purchased the Knowage SI package). Remember to save to move to the next step: open the Template Build. The latter can be opened clicking on the editor icon  and it is available at the bottom of the document detail page.

The action opens a first page asking for the kind of template. Here we choose the Mondrian one. Consequently you will be asked to choose the Mondrian Schema and after that to select a cube. Figure 4.96 sums up these three steps. Following the example just given in Figure 4.96 you will enter a page like that of Figure 4.97.

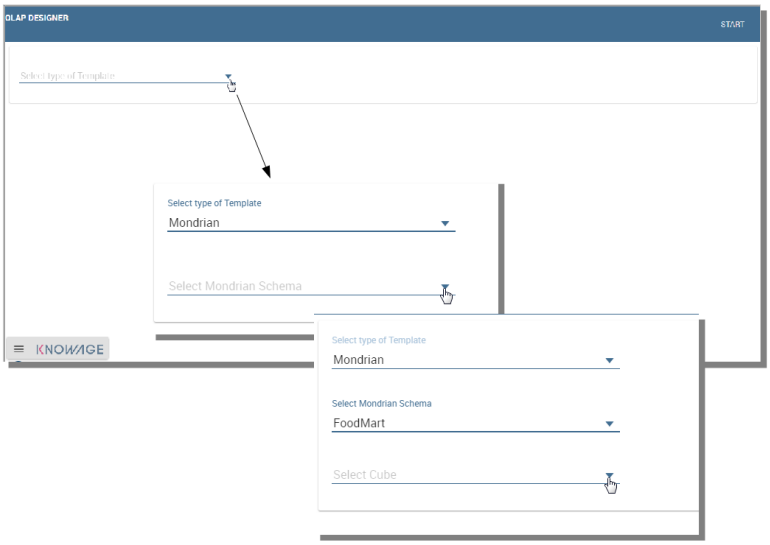


Figure 4.96: OLAP core configuration.

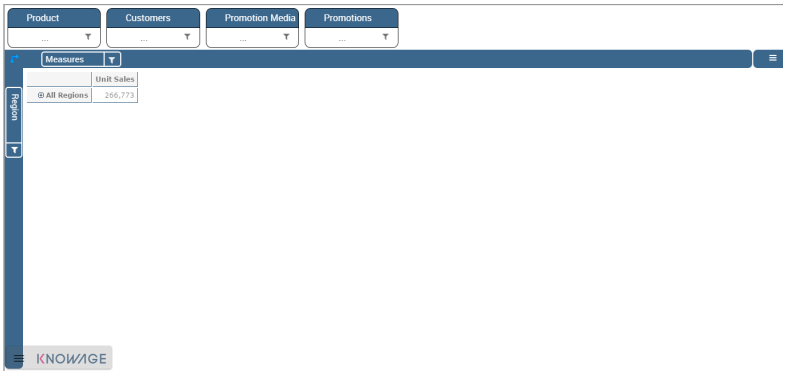


Figure 4.97: Defining OLAP template.

Once entered the page the user can freely set the fields as filter panels or as filter cards, according to requirements. Refer to Chapter 2.3 to review the terminology. Make your selection and you can already save the template as shown in Figure 4.98. You can notice that the side panel contains some features (Figure 4.99):

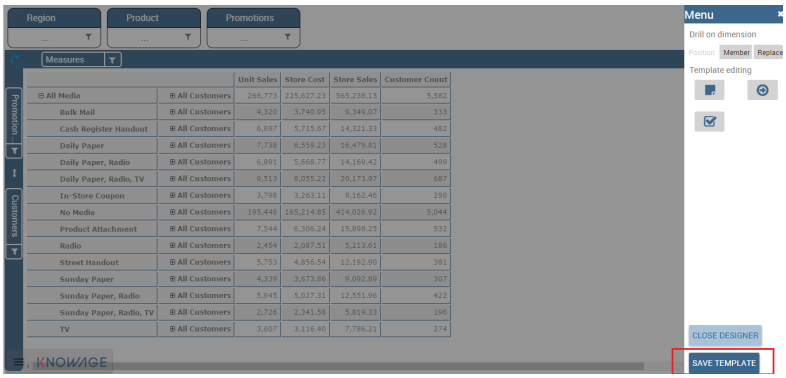


Figure 4.98: Defining OLAP template.

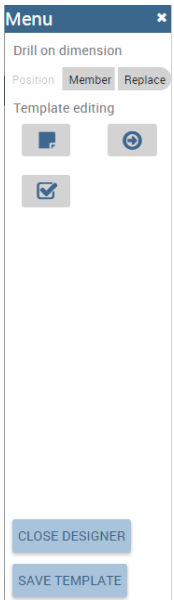
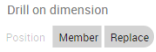


Figure 4.99: Side panel features for the OLAP Designer.

- 

to set the drill on Position, Member or Replace;
-  to configure the scenario;
-  to define the cross navigation;
-  to configure buttons visibility.

Refer to Section 2.3 to recall the action of the different drills. To select between them will affect the navigation of the OLAP outputs by users. Instead the scenario is used to allow the end-user to edit or not the records contained in the OLAP table. The user is first asked to select the cube in order to get the measures that the admin lets the end-user the permission to edit and modify. Referring to Figure 4.101, an admin user must simply check the measures using the wizard. At the bottom of the page there is also the possibility to add a parameter that can be used by the end-user when editing the measure, for example if one has a frequent multiplication factor that changes accordingly to the user's needs, the end-user can use that factor to edit measures and ask the admin to update it periodically.

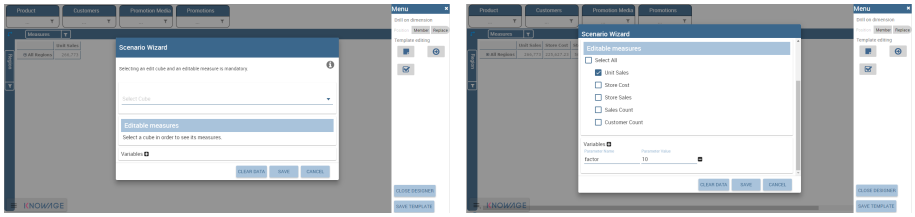


Figure 4.100: Wizard to configure the scenario.

Once one cross navigation has been set you keep on adding as many as required. Just open the wizard and click on the "Add" button at the top right corner.

Note that the parameter name will be used to configure the (external) cross navigation. In fact, to properly set the cross navigation the the user must access the "Cross Navigation Definition" functionalities available in Knowage Server. Here, you will use the parameter just set as output parameter.

As shown in Figure 4.102, the buttons visibility serves to decide which permissions are granted to the end-user. Some features can only be let visible while the admin can also grant the selection for others.

Once the configuration is done click on the **`Save template`** button and on the **`Close designer`** button to exit template. As Figure 4.99 highlights, these two buttons are available at the bottom of the side panel.

The admin can develop the OLAP document using also the OLAP engine. In this case the OLAP designer will lack of the scenario configuration since in this case the end-user must not have the grants for editing the records. So in this instance the "Configure scenario" button is not

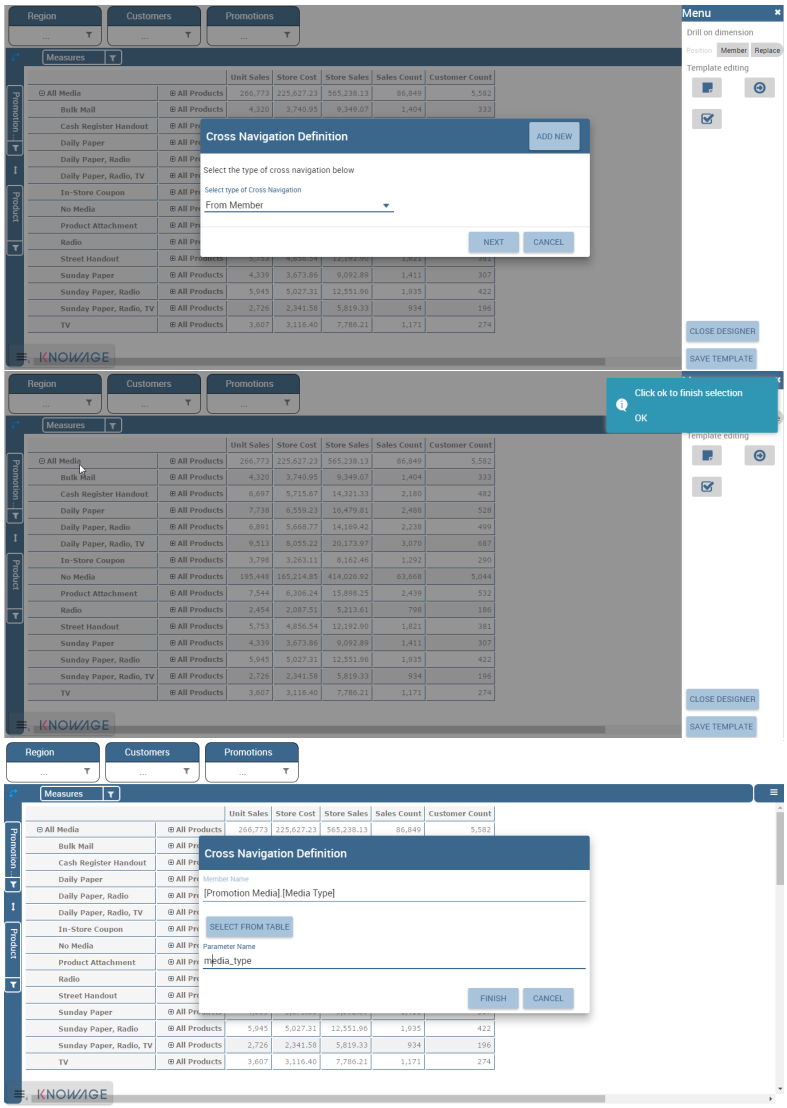


Figure 4.101: Cross navigation definition.

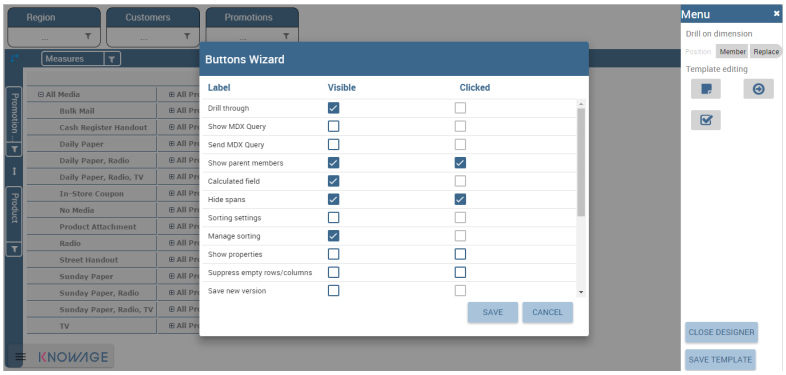


Figure 4.102: Wizard to configure the scenario.

available at all. For the other two options the instructions are right the same as the What-if engine.

Once all configurations are set the user must save the work. The OLAP will be displayed with the typical cube icon as shown in Figure 4.103.

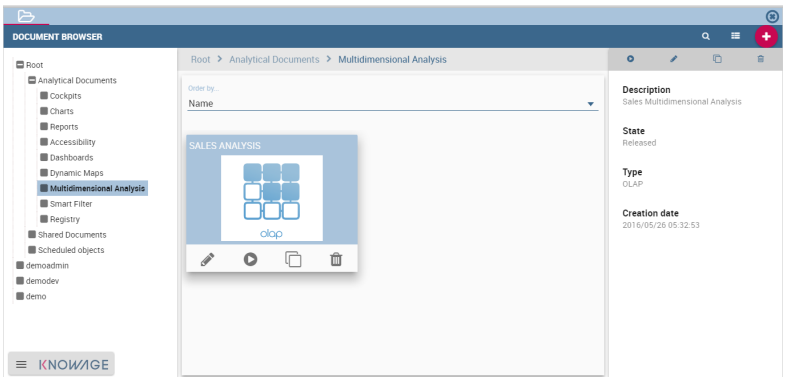


Figure 4.103: OLAP document on server.

Profiled access

Knowage provides two different ways to regulate data visibility based on user profiles. Data visibility can be profiled at the level of:

- the OLAP query;
- the OLAP cube.

These two approaches are conceptually different. In the first case, filtering policies apply to the query: this means that the cube has all data visible and each query can be filtered according to different criteria.

In the second case filtering policies directly apply to the cube: therefore, the cube itself is filtered and all queries over that cube share the same data visibility criteria.

To set the data profiling at the query level, you should edit the `DATA-ACCESS` section in the OLAP query template. The available access levels are:

None. No member of this dimension will be accessible in the query.

Custom. Access is granted to some members of the dimension only.

All. All members of the dimension are accessible.

In Code 4.5 you can see an example where the visibility rule is applied to members of the dimension `[Product].[All Products].[${family}]`. Here `${family}` represents an attribute in the user profile. The rule states that each user will be allowed to see members of the product dimension having the family attribute value equal to the value of the profile attribute for that user. For example, if the attribute has value `Food` for a user, he will be allowed to see only products whose category is `Food`. If the attribute has no value for a user, then he will be allowed to see all categories of the product.

```
<?xml version="1.0" encoding="UTF-8"?>
<olap>
...
<!-- data profiling -->
<DATA-ACCESS>
  <granted-dimensions>
    <dimension name="Product" grantSource="ProfileAttributes">
      <rules access="custom" topLevel="" bottomLevel="" rollupPolicy="PARTIAL">
        <members>
          <MEMBER name="[Product].[AllProducts].[${family}]" access="all"/>
        </members>
      </rules>
    </dimension>
  </granted-dimensions>
</DATA-ACCESS>
```

```
</olap>
```

Code 4.5: Query level profilation example.

The `topLevel` and `bottomLevel` attributes state the range of values that the user can see when navigating the cube (e.g., drilling down). If they are left blank, the user can access all hierarchy levels with no limitation.

The `rollupPolicy` attribute determines how Mondrian computes a member's total if the current role cannot see all children of that member. There are three possible values:

Full. The total for that member includes all children. This is the default policy if you do not specify the attribute value.

Partial. The total for that member includes accessible children only.

Hidden. If no child is inaccessible, the total is hidden.

As said above, the second profiling modality takes place at the cube level. In this case you need to set the filter, which is based on an attribute in the users profile (like the OLAP query case), within the Mondrian schema. This is the result of the fact that data profiling is performed on the cube directly, not on the query. Here you don't need to define a rollup policy because the filter acts directly the data retrieval logics on the Mondrian Server. So the user can only see the data that have been retrieved by the server according to the filter, with no further option.

```
<?xml version="1.0"?>
<Schema name="FoodMartProfiled">
....
<Cube name="Sales_profiled">
  <Table name="sales_fact_1998"/>
  ...
  <!-- profiled dimension -->
  <Dimension name="Product" foreignKey="product_id">
    <Hierarchy hasAll="true" allMemberName="All Products" primaryKey="
    product_id">
      <View alias="Product">
        <SQL dialect="generic">
          SELECT
            pc.product_family as product_family,
            p.product_id as product_id,
            p.product_name as product_name,
            p.brand_name as brand_name,
            pc.product_subcategory as product_subcategory,
            pc.product_category as product_category,
            pc.product_department as product_department
```

```
20          FROM
21            product as p
22            JOIN product_class as pc ON p.product_class_id = pc.
    product_class_id
23          WHERE
24            and pc.product_family = '${family}'
25        </SQL>
26      </View>
27    <Level name="Product Family" column="product_family" uniqueMembers="false"
    />
28    <Level name="Product Department" column="product_department" uniqueMembers=
    "false"/>
29    <Level name="Product Category" column="product_category" uniqueMembers="
    false"/>
30    <Level name="Product Subcategory" column="product_subcategory"
    uniqueMembers="false"/>
31    <Level name="Brand Name" column="brand_name" uniqueMembers="false"/>
32    <Level name="Product Name" column="product_name" uniqueMembers="true"/>
33    </Hierarchy>
34  </Dimension>
35 </Cube>
36 ...
37 </Schema>
```

Code 4.6: Cube level profilation example.

In the above example, the filter is implemented within the SQL query that defines the dimension using the usual syntax and `pr.product_family = '${family}'`.

4.11 My first KPI

KPI stands for Key Performance Indicator. It denotes a set of metrics, usually derived from simple measures, which allow managers to take a snapshot on key aspects of their business. The main characteristics of KPIs follow:

- summary indicators,
- complex calculations,
- thresholds supporting results evaluation,
- reference target goals,
- easy to use but requiring more specific skills to design it,

- association with alarms,
- not necessarily used for real-time analysis,
- may refer to a specific time frame.

For these reasons, KPIs are always defined by expert analysts and then used to analyze performances through synthetic views that select and outline only meaningful information.

Knowage allows the configuration of a KPI document thanks to a specific **KPI engine**. The critical value (or values) can be computed and visualized through the functionalities available in the 'KPI model' section of Knowage menu area (see Figure 4.104).

KPI development

We introduce the KPI tool by splitting the topic in steps. We briefly sum up here the arguments that we will cover in the following sections in order to have a general overview of the KPI implemetation.

- **Measure definition**: it is necessary to define first the measures and attributes, eventually with parameters, to compute the critical value of interest.
- **KPI definition**: here you compute the requested value through a formula using the measures and attributes set in previuos step and configure thresholds.
- **Target**: it is possible to monitor the value of one or more KPIs comparing it to an additional fixed value.
- **Schedulation**: you can schedule the execution of one or more KPIs, eventually filtered by conditions.
- **Document creation**: finally, you develop the KPI document.

Therefore, we go into further details.

Measure definition. The first step is to create a new measure or rule. Select **Measure/Rule Definition** from the contextual menu of the main page, as shown in Figure 4.104.

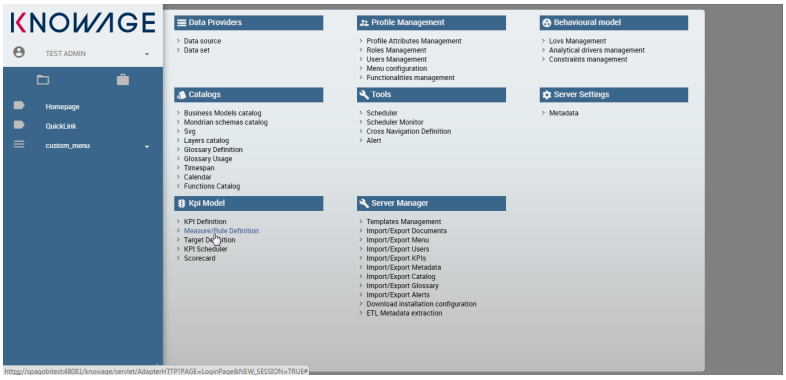


Figure 4.104: Measure/Rule Definition menu item.

Click on the "Plus" icon to set a new measure/rule. A query editor page is opened. Note that once the data source is selected, pressing simultaneously the CRTL key and the space bar opens a contextual menu containing the available datasource columns and the database keywords. Refer to Figure 4.105 to have an example.

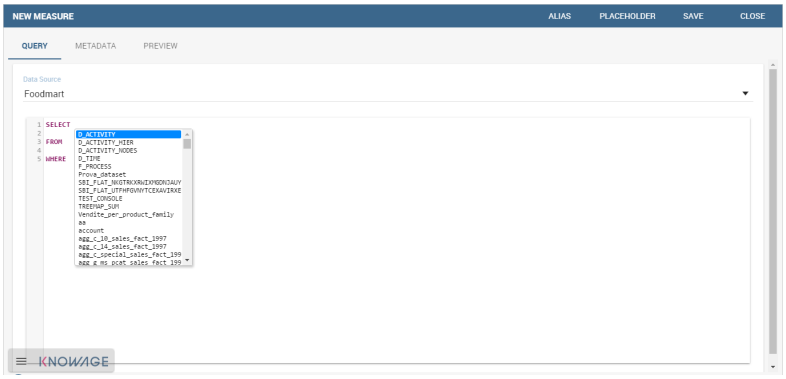


Figure 4.105: Editing the query when defining a KPI.

Each rule is based on a query to which you can add placeholders. A placeholder represents a filter that you can add to your rule and that can be useful for profiling tasks. It is possible to assign value to a placeholder while configuring the schedulation of the KPI (this procedure will be further examined in "Document creation" paragraph). The syntax to use in queries for setting a placeholder is `columnName=@placeholderName`, as the example in Figure 4.106 shows.

Data Source

Foodmart

```
1  SELECT
2  p.brand_name
3   ,p.product_name
4  , store_country
5  ,store_city
6  , sum(store_sales) as store_sales4
7  , sum(store_cost) as store_costs4
8  , sum(unit_sales) as unit_sales4
9  FROM
10 sales_fact_1997 s
11 join product p on p.product_id=s.product_id
12 join store st on s.store_id=st.store_id
13 join time_by_day t on t.time_id = s.time_id
14 join product_class pc on pc.product_class_id=p.product_class_id
15 WHERE
16 store_country=@STORETST
17 GROUP BY
18 p.brand_name
19  ,p.product_name
20 , store_country
21 ,store_city
22 ORDER BY
23 store_sales desc
```

Figure 4.106: Setting placeholder in query definition.

Generally the rule such a query can return one or more measures and possibly an attribute. An example is given in Figure 4.107.

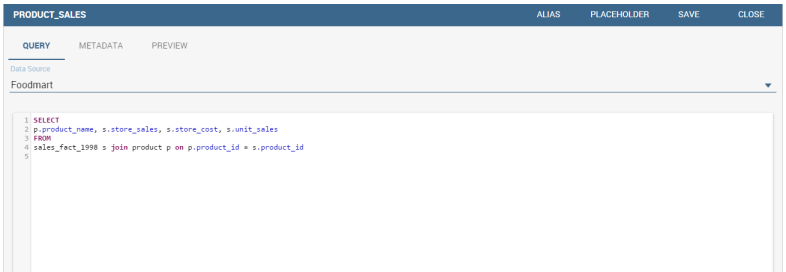


Figure 4.107: Query definition.

A typology (measure, attribute and temporal attribute) and a category can be assigned to each fields returned by the query using the **Metadata** tab as highlighted in Figure 4.108. The typology is required to associate a type to each field returned by the query. In particular, if the field is a temporal one, it is mandatory to specify to which level you want it to be considered, that is if it corresponds to a day, a month, a year, a century or a millennium. For measures and attribute it is possible to assign also a category to easily look them up in a second moment.

We say in advance that, it is important to distinguish these metedata categories from the required field "Category" that occurs while saving the KPI definition (see Figure 4.121). In

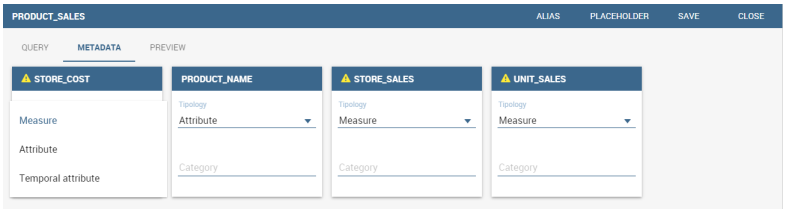


Figure 4.108: Metadata settings.

fact, the category assigned when saving the KPI definition will be added (if it doesn't exist) in the "KPI categories" list, used to profile KPIs on roles (see Figure 4.109).

Do not mistake metadata category with the KPI category

The categories defined in the metadata tab of the "Measure definition" functionality are not the same categories selected in the tab area of the "Roles management" functionality (see Figure 4.109). The fist serve to classify the metadata while the second serve the profiling issue.

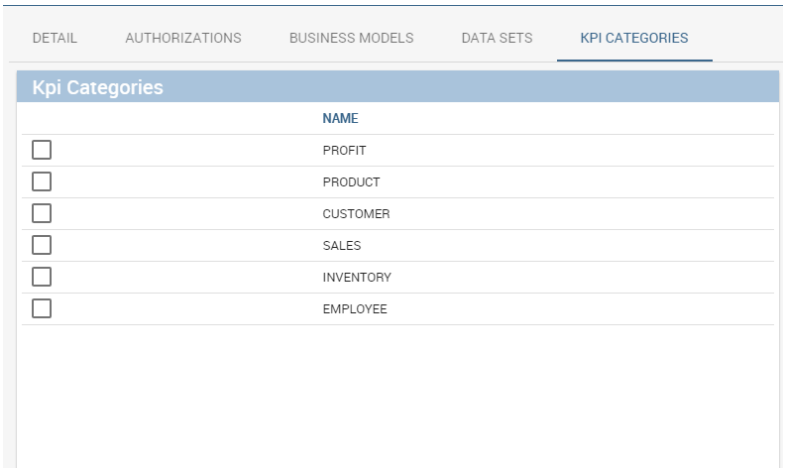


Figure 4.109: KPI category.

As we told, a proper categorization exists for the aggregations of type temporal. In fact, when associating "temporal attribute" as metadata typology, the technical user must indicate the hierarchy level of the data: day, month or year. You can see an example in Figure 4.110. Note that the field set as temporal type must contains numbers (therefore string types are not allowed). For example, if one wants to set a field as "month", such a field must contain {01,02,03,...,12} that will be considered as {January, February, March,...,December}.

The **`Preview`** tab allows you to check the query execution and have a look on a part of the

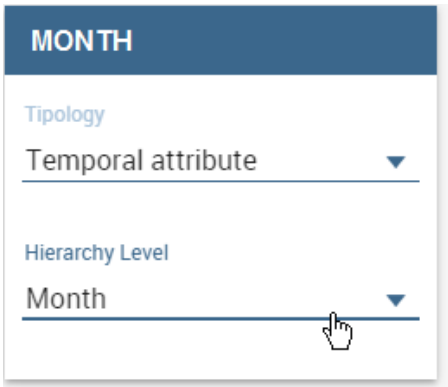


Figure 4.110: Hierarchy level for temporal attributes.

result set.

Let's now examine extra features available on the right top corner. There you can find the following tab:

- **`Alias`**: as in Figure 4.111, you can see the aliases defined in other rules; note that only the aliases of those colums saved as attribute are stored and showed. This is useful in order to avoid aliases already in use when defining a new rule. Indeed an alias can not be saved twice even if contained in different rules.

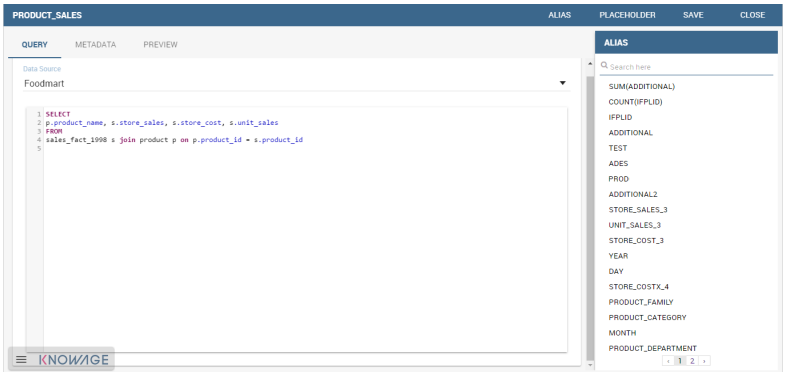


Figure 4.111: Checking aliases.

- **`Placeholder`**: here you can check the existing placeholders. These are set in the query you're editing or in other ones. Figure 4.112 gives an example.
- **`Save`**: to save the query and other settings just configured.
- **`Close`**: to exit the rule configuration window.

Figure 4.112: Setting placeholders in a query.

KPI definition. Select the **KPI definition** item from the contextual menu of the main page of Knowage, as shown in Figure 4.113. Click on the "Plus" icon to configure a new KPI.

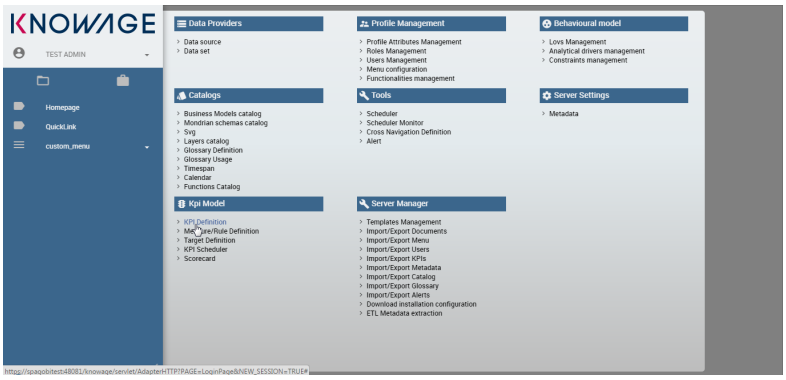


Figure 4.113: Configure a new KPI.

The window opens a first tag, entitled **Formula** (see Figure 4.114), where you must type the formula to enable calculations.

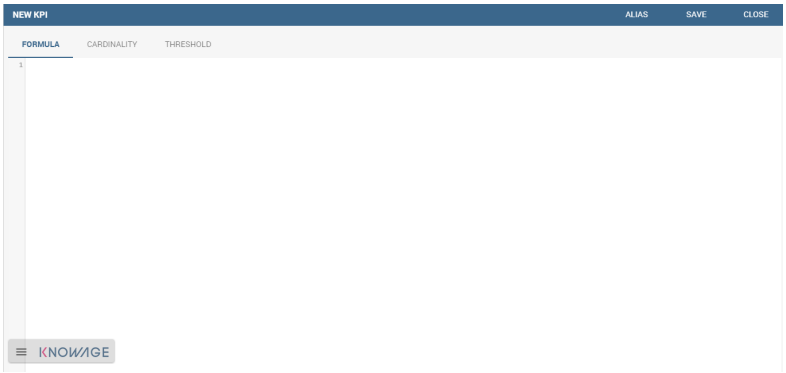


Figure 4.114: Formula definition tab.

Press CTRL key and space bar simultaneously to access all measures defined in the rules, as shown in Figure 4.115. Once a measure is selected, you need to choose which function must act on it. This can be done by clicking on the $f()$ that surrounds the chosen measure. See Figure 4.116.

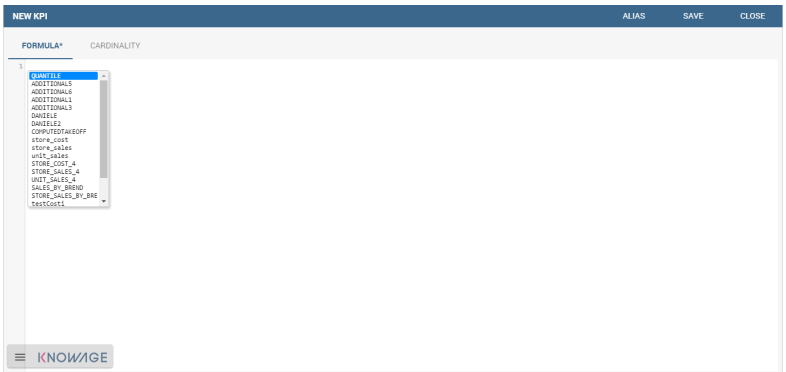


Figure 4.115: Press crtl and space to get measures.

Clicking on the $f()$ the interface opens a pop up where you can select which function apply to the measure, see Figure 4.117. Once the selection is made the formula will be autofilled with the proper sintax and you can go on editing it.

Once a complete formula (an example is given in Figure 4.118) has been inserted you can move to the next tab.

The **`Cardinality`** tab allows you to define the granularity level (namely the grouping level) for the attributes of the defined measures.

Referring to the example in Figure 4.119, selecting (with a check) both measures for the attribute `product_name`, the KPI measures are computed, grouped on each `product_name`; otherwise no grouping will be done.

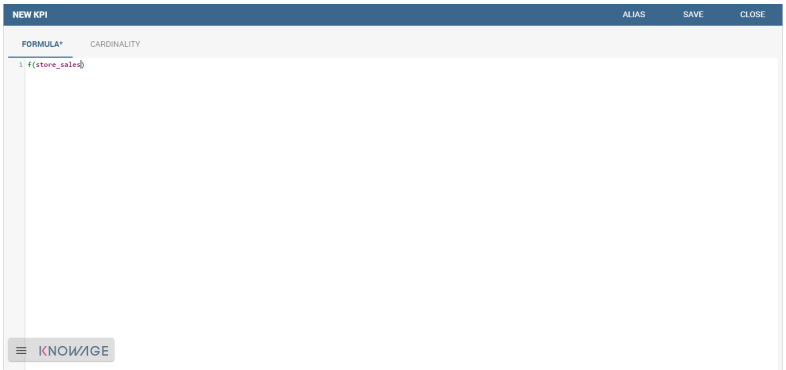


Figure 4.116: Formula syntax.

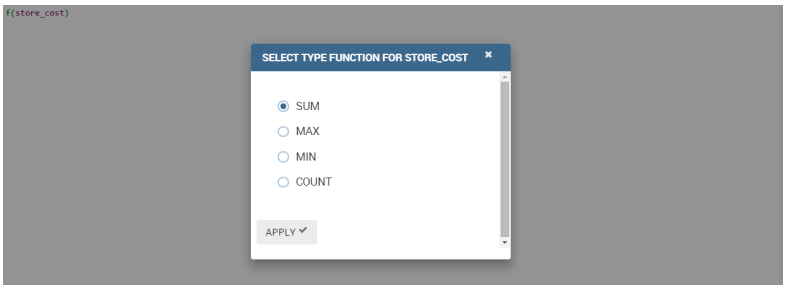


Figure 4.117: Available functions.

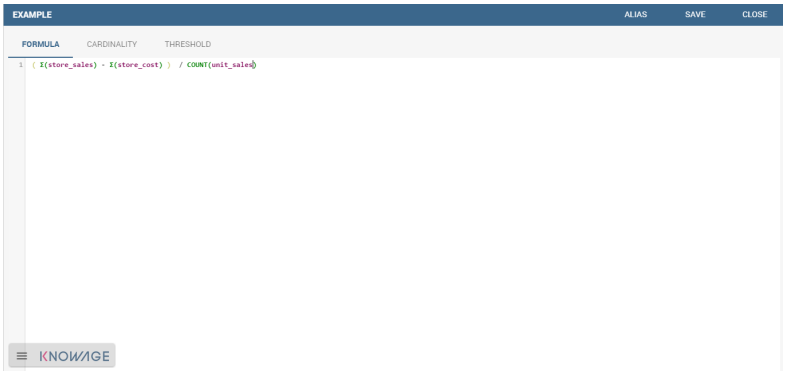


Figure 4.118: Complete formula example.

Figure 4.119: Cardinality settings example.

Limit values can be set using the **Threshold** tab (Figure 4.120). It is mandatory to set at least one threshold otherwise the KPI cannot be saved. You can choose a threshold already defined clicking on "Threshold" list or create a new one.

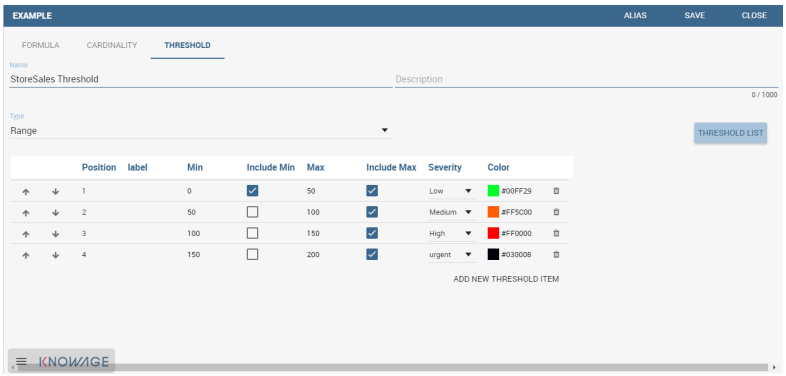


Figure 4.120: Setting thresholds.

To insert a new threshold it is mandatory to insert a name and assign a type, while the description is optional. Clicking on **`Add new threshold`** item a new item appears. It is necessary to define the **`Position`**, **`Label`**, **`Minimum`** and **`Maximum`** values. It is possible to choose if to include the minimum and maximum values in the value slot. The **`Severity`** is used to link colors to their meaning and make the thresholds readable by other technical users. Note that the color can be defined through the RGB code, the hexadecimal code or choosing it from the panel. Remember to save once all thresholds have been set.

"Standard" colors for thresholds

We call **`standard colors`** for thresholds the ones listed below (in terms of hexadecimals):

- green: #00FF00,
- yellow: #FFFF00,
- red: #FF0000.

Finally the user must save the KPI definition clicking on the "Save" button, available at the right top corner of the page. Once the user clicks on the "Save" button, the "Add KPI associations" wizard opens, as you can see from Figure 4.121. Here, it is mandatory to assign a name to the KPI. In addition, the user can set the KPI category so that only users whose roles have the permmissions to this specific category can access the KPI. Remember that it is possible to assign permissions over KPI when defining roles using the "Roles management" functionality available on Knowage main page. Furthermore, the user can check or uncheck the "**`Enable Versioning`**" button if he/she wishes to keep track of the rules/measures/targets that generate the KPI response at each KPI execution.

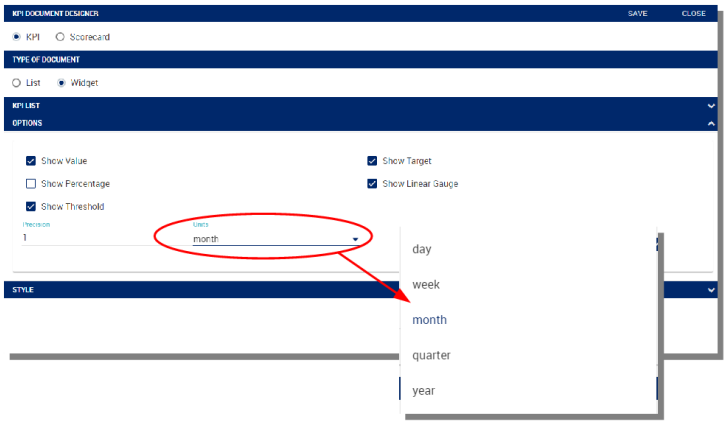


Figure 4.121: Save the KPI definition and set category.

Target. This step is not mandatory. Enter the **Target Definition** menu item as shown in Figure 4.122.

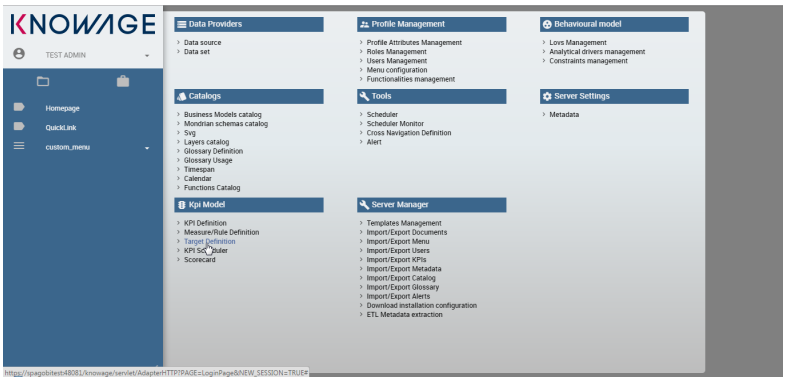


Figure 4.122: Target Definition menu item.

Clicking on the "Plus" icon you can add a new target (Figure 4.123).

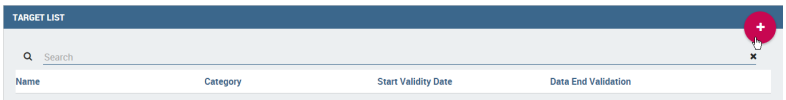


Figure 4.123: Add a new target.

The definition of a new target requires to type a name, a validity start date/end date and the association to at least one target. It is possibile to associate a target clicking on the item **Add KPI** and selecting the KPI of interest. Once the association is set, the "Value" box gets editable and you can insert the value you wish to send to the selected KPI. An example is given in Figure 4.124. In the KPI visualization phase, a red bold thick will be displayed on the indicated value (see Figure 4.125).

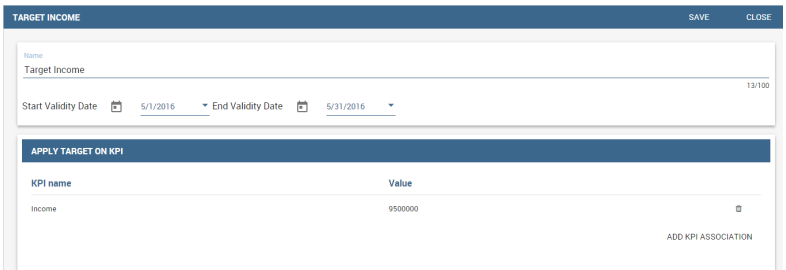


Figure 4.124: KPI target association.

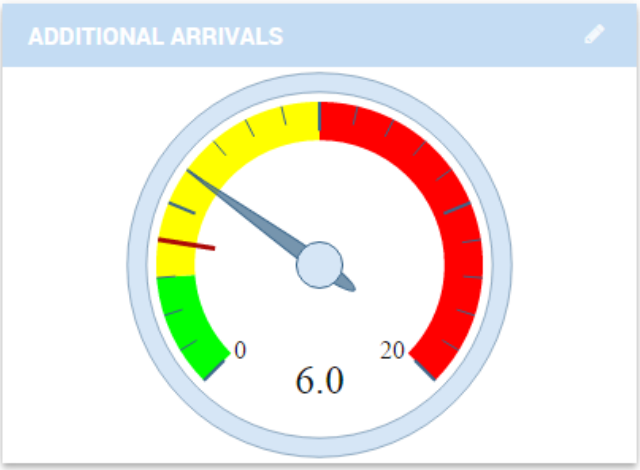


Figure 4.125: Target mark in KPI scale of values.

Note that once targets are set, the window in Figure 4.123 gets populated with a list. Note that here the category serves as description and only to order the records.

`Schedulation.` Once the KPI has been defined, it is necessary to schedule them to proceed with the creation of an analytical document. For this purpose, click on the **`KPI Scheduler`** from the contextual menu that you can see in Figure 4.126.

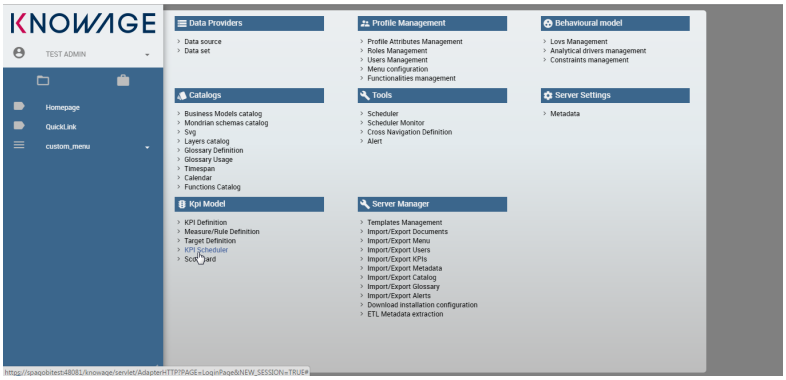


Figure 4.126: KPI Scheduler menu item.

As for the other interfaces it is enough to click on the "Plus" icon to create a new schedulation. The new schedulation window presents several tabs.

- **`KPI`**: it is possible to associate one or more KPI to the schedulation clicking on "Add KPI Association".
- **`Filters`**: here you assign values to the filters (if configured) associated to the schedu-

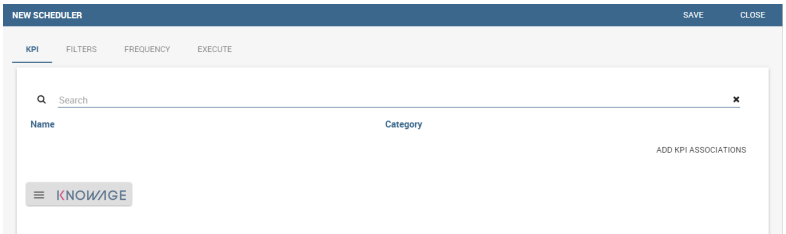


Figure 4.127: KPI tab window.

lation. Note that it is possibile to assign values to the filters by means of a LOV, a fixed list of values or a temporal function. In case the LOV option is chosen, remember that the LOV must return one unique value. This choice can be useful for profiling tasks.

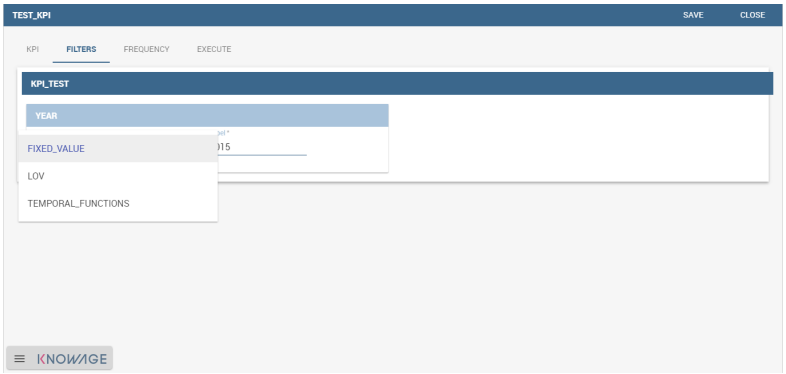


Figure 4.128: Filters options.

- **`Frequency`**: referring to Figure 4.129, here is the place where the schedulation time interval (start and end date) can be set together with its frequency.

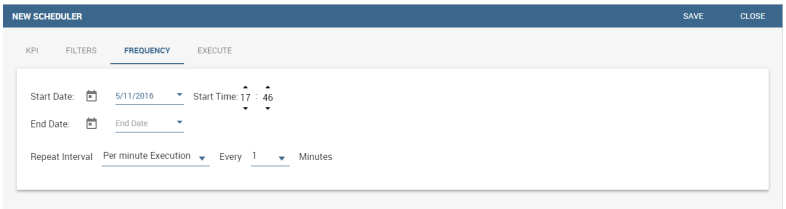


Figure 4.129: Frequency tab window.

- **`Execute`**: here you can select the execution type. The available options distinguish between the storing and the removal of old logged data. In fact, selecting **`Insert and update`** the scheduler compute the current (accordingly to the frequency choice) KPI values and store them in proper tables without deleting the old measurements and all error log text files are available right beneath. While selecting **`Delete and insert`** the

previous data are deleted.

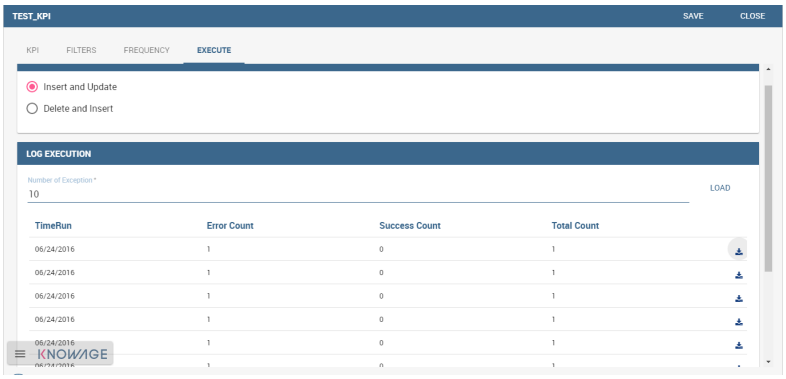


Figure 4.130: Execute tab window.

In Figure 4.131 we sum up the example case we have referred to since now.

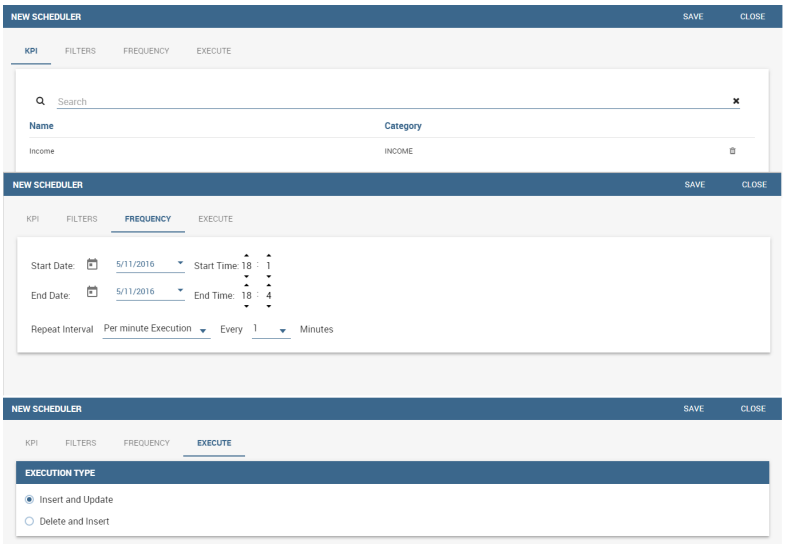


Figure 4.131: Overview of the KPI case.

Once the schedulation is completed click on the “Save” button. Remember to give a name to the schedulation as in Figure 4.132.

Creation of a KPI document

`Document creation.` Now the schedulation has been set and it is possible to visualize the results. We need at this point to create a new analytical document of type KPI and that uses the KPI engine (Figure 4.133). Then we save.

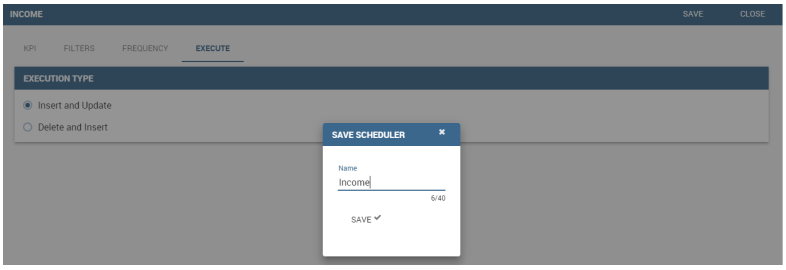


Figure 4.132: Overview of the KPI case.

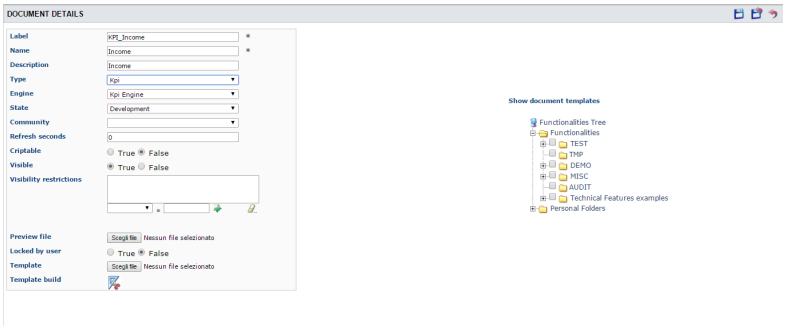


Figure 4.133: Overview of the KPI case.

Click on the **`Template build`** icon to develop the template. Here you can choose between KPI and Scorecard. In the KPI case it is possible to choose between the two following type of document, as you can see in Figure 4.138.

- **`List`**: with this option it is possible to add several KPI that will be shown in the same page with a default visualization shown in Figure 4.142.
- **`Widget`**: with this option it is always possible to add several KPI that will be shown in the same page but in this case you will also be asked to select its visualization: Speedometer (Figure 4.140) or KPI Card (Figure 4.139); then the minimum value and the maximum value that the KPI can assume and if you want to add a prefix or a suffix (for example the unit of measure of the value) to the showed value.

Then practically you must add the KPI association using the KPI List area of the interface. As you can see from Figure 4.134 here you select the KPI after clicking on the "ADD KPI ASSOCIATION" link. The latter opens a wizard that allows to pick up a multiple choice of the KPIs. Once chosen, you need to specify all empty fields of the form, like "Category", "View as" and so on (refer to Figure 4.134). Note that the "View as" field is were you can decide if the widget will be a Speedometer or a KPI Card.

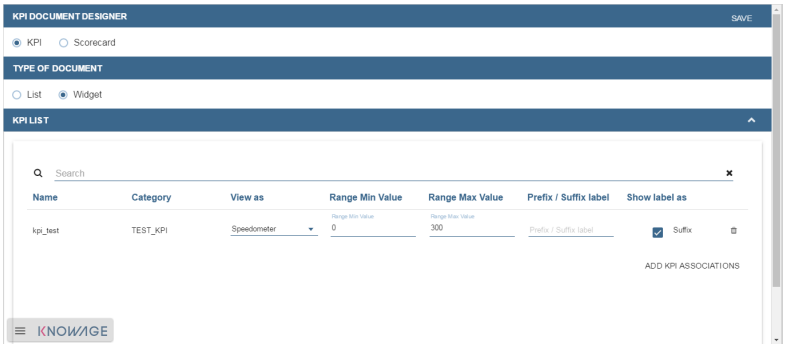


Figure 4.134: Setting the KPI associations using the dedicated area.

Moreover, you can set the other properties of the KPI document using the **`Options`** and the **`Style`** areas (Figure 4.135).

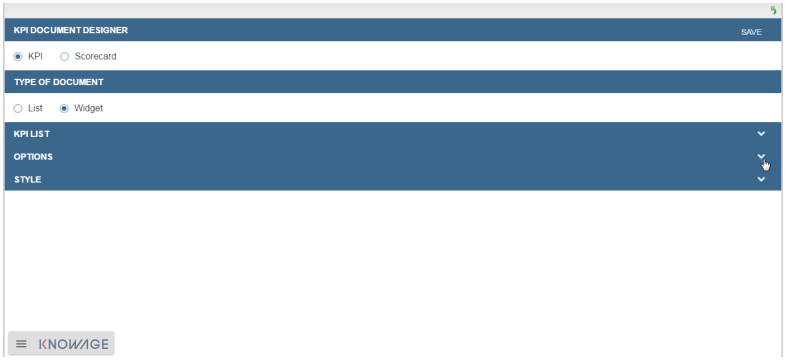


Figure 4.135: Areas of the Template Build for KPI.

In particular, it is possible to steer the time granularity used by the KPI engine to improve the performances. For this purpose, in the “Options” area (Figure 4.136) the user is invited to indicate the level of aggregation choosing among “day”, “week”, “month”, “quarter”, “year”.

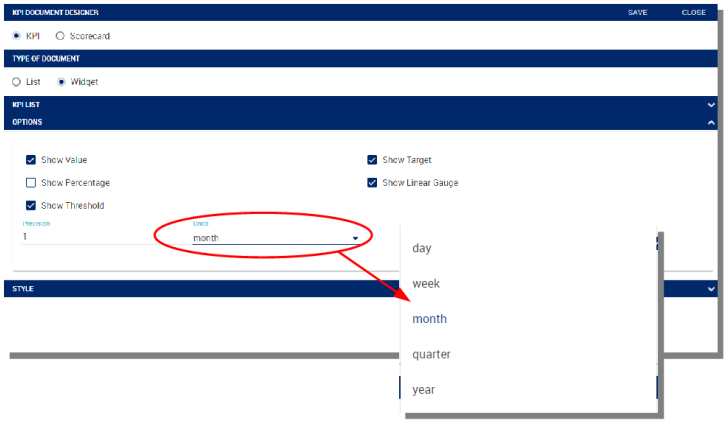


Figure 4.136: Choose the time granularity.

Finally in the “Style” area the user can customize the size of the widget, the font, the color and size of texts.

Figure 4.137: Style settings.

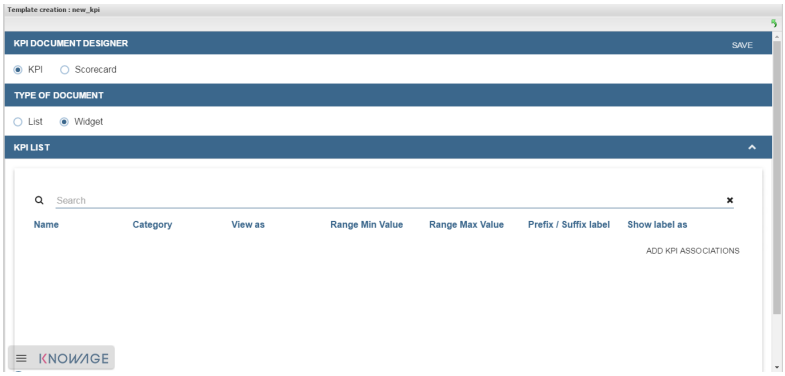


Figure 4.138: KPI association.

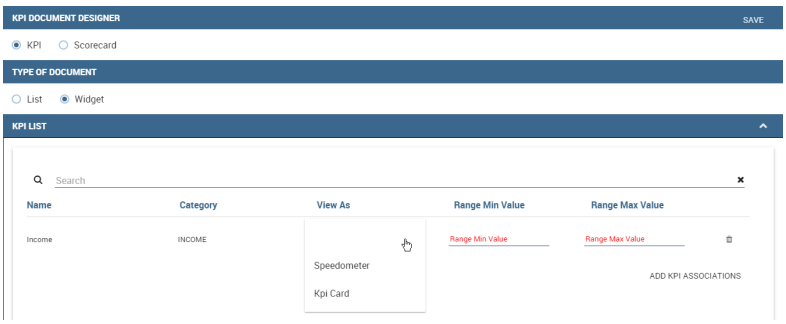


Figure 4.139: Widget document.

Then save and run the document. Examples are shown in Figure 4.140, Figure 4.141 and Figure 4.142.

In case the document contains grouping functions upon them, it is necessary to add the proper analytical drivers. Refer to Section 4.5 to get more information about how to associate an analytical driver to a document (and therefore to a KPI document). We stress that the URL of the analytical driver *must* coincide with the *attribute aliases* on which you have defined the grouping.

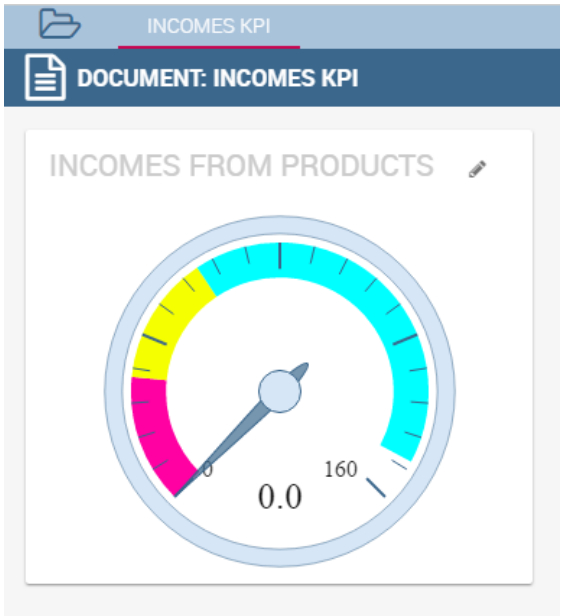


Figure 4.140: KPI Speedometer.

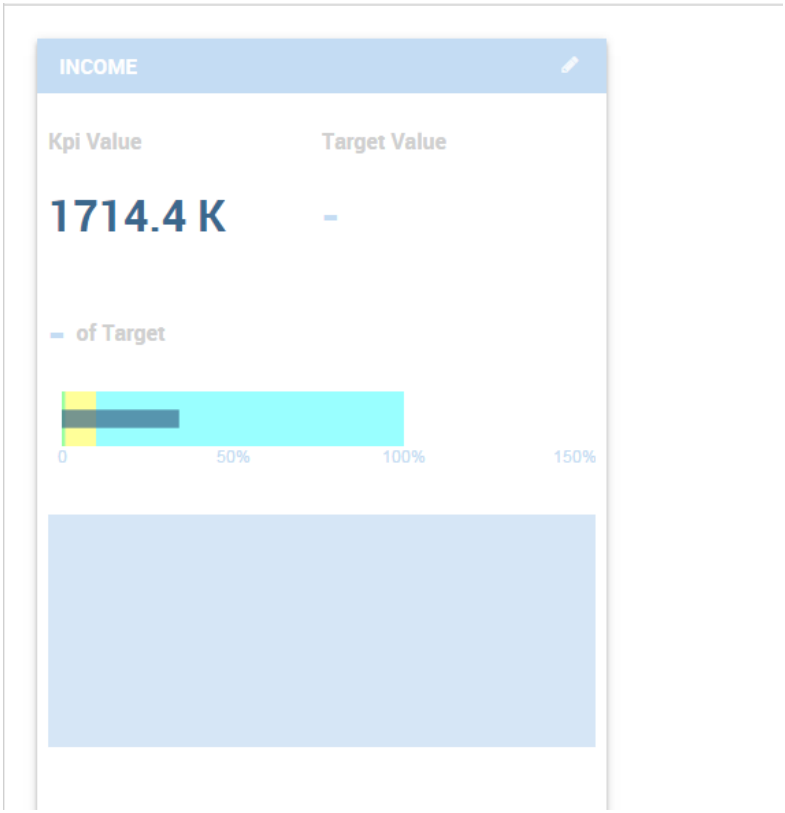


Figure 4.141: KPI Card.

Figure 4.142: KPI List.

4.12 My first Map

GIS engine document templates can now be built using GIS designer. Designer is available from administrator document detail page and also for end users workspace. The creation process and designer sections are described in the text below.

A GIS document can be created by a final user from workspace area of Knowage Server. Follow My Workspace » Analysis and click on the "Plus" icon available at the top right corner of the page and launch a new **`Geo-referenced analysis`**.

Figure 4.143: Start a new Geo-referenced analysis.

When the designer is opened there is option to choose dataset for joining spatial data and business data. When the dataset is selected the Dataset join columns and indicators sections will appear. By default dataset is not chosen and there is interface to create map without business data

Designer sections

Map name

Map name represents the name of the map, required in all the cases. In Figure 4.145 an example of name editing.

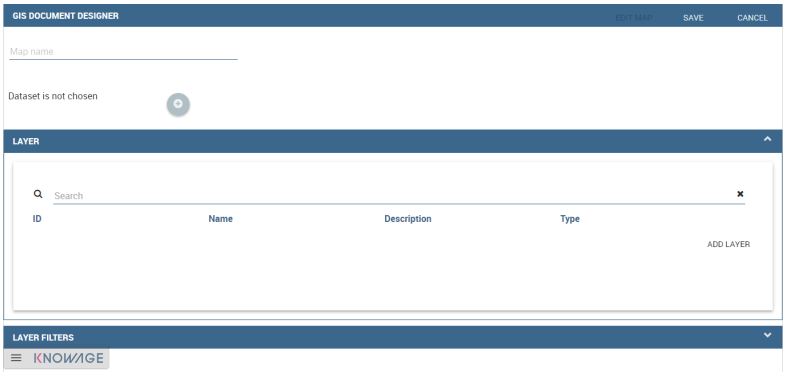


Figure 4.144: Gis document designer window.

Figure 4.145: Map name interface.

Layer section

Definition of the target layer is configurable in layer section. If the dataset is selected one of the available layers is chosen from list of layers catalogs. Button change layer (Figure 4.146) opens a pop up with a list of all available layer catalogs (Figure 4.147). Selecting one item from the list and clicking save the selected item will be chosen for template.

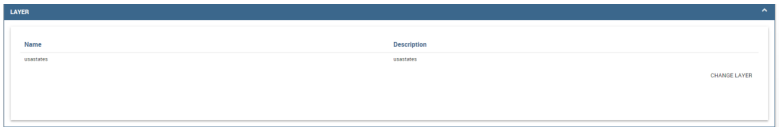


Figure 4.146: Target layer definition.

In case when there is no dataset multiple layers can be selected as in Figure 4.148.

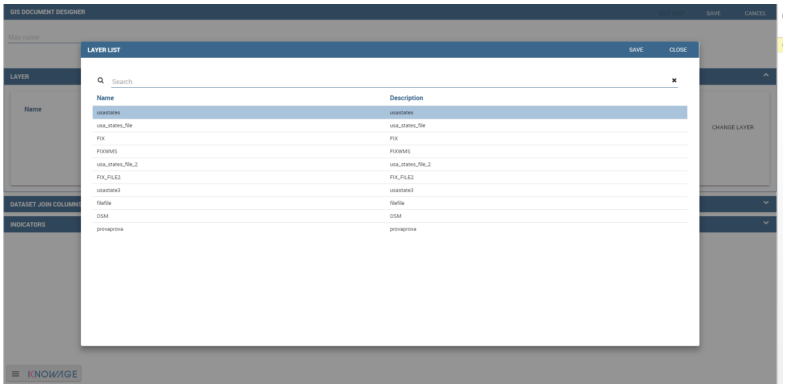


Figure 4.147: List of available layer catalogs.

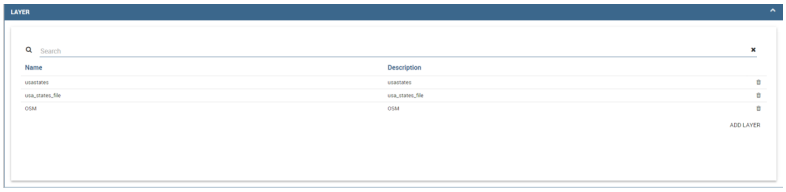


Figure 4.148: Multiple selection of available layers.

Dataset join columns

Dataset join columns section is for configuring joining spatial data and business data. This section is only present when the dataset is selected for the document. Designer data structure for joining is represented by the pairs of dataset columns and corresponding layer columns. Clicking on add join column that you can see in Figure4.149 new empty pair appears. Dataset join column can be selected from columns on selected dataset by choosing an option from combo box. Layer join column should be added as a free text by editing corresponding table column.

Figure 4.149: Dataset join columns interface.

Indicators

Measures definition is configurable by adding indicators. The interface is shown in Figure 4.150 This section is only present when dataset is chosen for the document. Indicators are

represented by pairs of measure field from selected dataset and corresponding label that will be used on map. Clicking on add indicators creates empty pair. Measure filed should be selected by picking one option from combo box that contains measure fields from selected dataset. Label should be inserted as free text by editing corresponding table column.

Figure 4.150: Indicators interface.

Filters

Using the filtering dedicated area of Figure 4.151 you define which dataset attributes can be used to filter the geometry. Each filter element is defined by an array (e.g. name : "`store_country`", label :"COUNTRY"). The first value (name : "`store_country`") is the name of the attribute as it is displayed among the properties. The second one label: "COUNTRY" is the label which will be displayed to the user. This section is only present when dataset is chosen for the document. Clicking on add filter creates empty pair. Label field should be selected by picking one option from combobox that contains attribute fields from selected dataset. Label should be inserted as free text by editing corresponding table column.

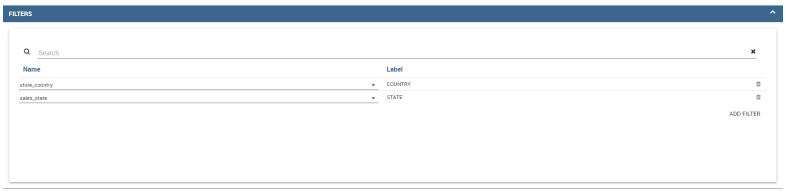


Figure 4.151: Filters interface.

Layer filters

Here, as you can see from Figure 4.152, you define which target layer attributes can be used to filter the geometry. *This section is only present when there is no dataset.* Add filters button opens pop up where you can choose all available filters of the selected layers. Figure 4.153 gives an example.

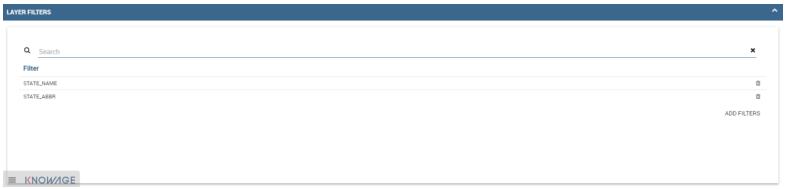


Figure 4.152: Layer filters interface interface.

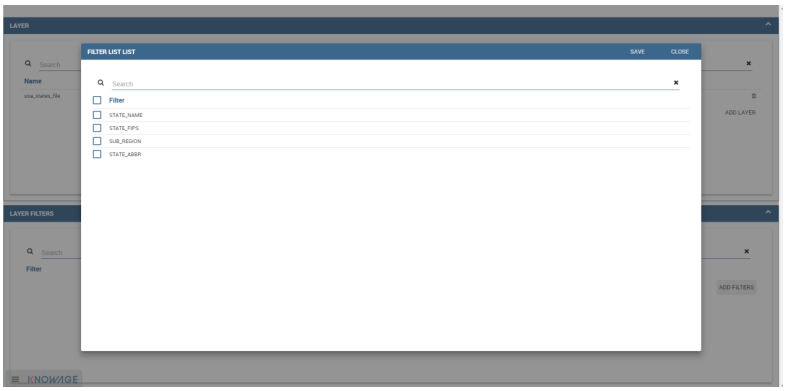


Figure 4.153: List of available filters.

Edit map

When all required fields are filled basic template can be saved. From workspace user is first asked to enter name and description of new created document as in Figure 4.154. When the template is saved successfuly EDIT MAP button is enabled in the right part of the main toolbar. Clicking the edit map button will open created map. An example is given in Figure 4.155. In edit mode you are able to save all custom setting made on map.

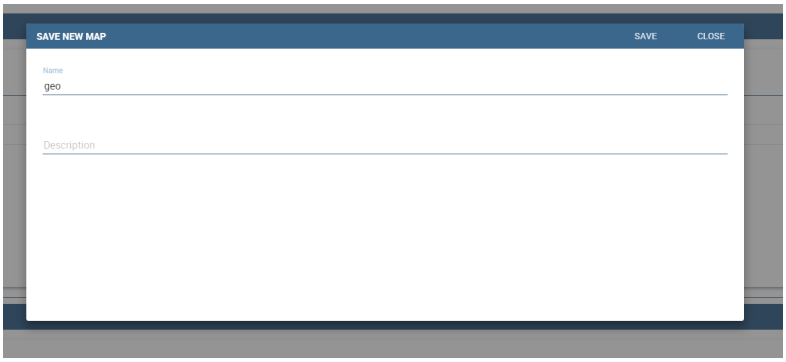


Figure 4.154: interface for name and description of new geo document for end user.

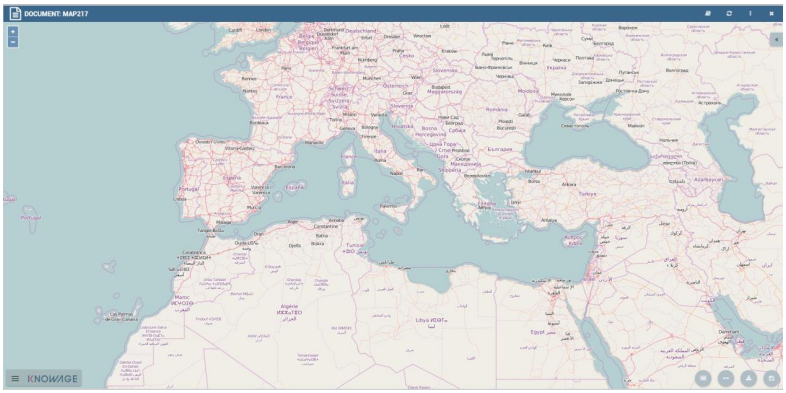


Figure 4.155: Map in edit mode with save template available.

4.13 My first Network

What is a network? It's a set of nodes linked by relations. So it's a 3-tuple where the entities are:

- Source node;
- Destination node;
- Relation.

In Knowage a document is the composition of a template and the data.

- In this case data are a dataset with at least the three columns described above.
- The template is the mapping of the columns of the dataset on graph properties (column → source node, column → destination node,)

Template

Knowage supports different types of network definitions. If you have your network defined in GRAPHML or XGMML notation you can upload that file as template of the document and that's it. After this operation you can open your network using Knowage.

On the other hand if you want to create a dynamic network that gets data from your datasource you should create a dynamic document. In this case the structure of a template is typed in Code 4.7:

```
<NET>

    <NETWOK_DEFINITION>
       Definition of the network with mapping and rendering options
```

```

        <options>
                general rendering options: tooltip, shape of the network,
    interaction buttons,...
        </options>

        <dataset_mapping_LIST>
                mapping between dataset columns and network properties: data,
    node shapes, node colors,....
        </dataset_mapping_LIST>
      </NETWOK_DEFINITION>

        <info>
        info of the document: left tab content
        </info>
        <drill>
          Cross navigation configuration
        </drill>

</NET>
```

Code 4.7: Template structure.

In the following we explain shortly the meaning of the Code 4.7.

The **NETWOK_DEFINITION** contains the definition of the network: nodes, edges, shapes, colours It has two children:

- **Options**: it contains the general options of the network. There are 2 type of options:
  - Network options. These are options that drive the rendering of all the network. For example where to put the navigation commands or the shape of the network (circular, radial,). You can find the list of available properties here http://cytoscapeweb.cytoscape.org/documentation/visualization and the list of layouts here http://cytoscapeweb.cytoscape.org/documentation/layout
  - Edge/nodes properties. General visual properties for the nodes and edges. The syntax for these settings is in next Code 4.8:

```
            <options>
                  <visual_style>
                          <nodes>
                                  Nodes properties
                          </nodes>
                          <edges
                                  Edges properties
```

```
                        </edges>
                </visual_style>
            </options>

```

Code 4.8: Syntax for edge/nodes properties.

Tooltip is a special Edge/node property. The tooltip contains a set of proerty/value couple and the syntax is typed in the next Code 4.9:

```
<nodes (or edge)>
  <tooltip_LIST>
  <tooltip property="OBJ PROPERTY" text="PROPERTY LABEL TEXT"/>
  </tooltip_LIST>
</nodes (or edge)>
```

Code 4.9: Syntax for tooltip,an edge/nodes property.

Where OBJ PROPERTY property is the name of the property (for example id) and PROPERTY LABEL TEXT is the text youll see as label of the property in the tooltip. You can find the list of available properties here: http://cytoscapeweb.cytoscape.org/documentation/visual_style

- **Dataset_mapping_LIST**: this section maps the columns of the dataset on properties of the graph. This is done with the tag `dataset_mapping`. There are 3 possibilities:
  - Map a column of the dataset on a property of the graph and the syntax is showed in Code 4.13:

```
<dataset_mapping element="source" column="sourceId" property="id"/>
```

    Where:

    * element: is the element where we want to apply the property. It can be source, target (for nodes) and edge;
    * property: the property of the network object we wan to set;
    * column: the label of the dataset column we want to map.

    The list of available node and edge properties is here http://cytoscapeweb.cytoscape.org/documentation/elements

  - Set a fixed value to a property. The syntax is showed in Code 4.13.

```
<dataset_mapping element="source" value="#caabff" property="color"/>
```

Where:

  * value is the fixed value of the property we want to set.

- **info**: contains some text/html that can help the user understanding the network. Since the syntax of the template is XML if you want to insert HTML you should envelop it into a CDATA tag. For example refer to Code 4.13:

```
      <![CDATA[
        .......
        ]]>

```

- **drill**: is used to link the network to another document. The structure of the tag is showed in Code 4.13

```
     <DRILL document="LINKED_DOCUMENT ">
        <PARAM name="PAR_NAME" type="TYPE" property =PROPERTY/>
     </DRILL>

```

Where:

  - DOCUMENT: is the label of the destination document;
  - PAR_NAME: is the destination document parameter label;
  - TYPE: parameter type
    * ABSOLUTE/RELATIVE,
    * EDGE: the parameter will get an edge property value,
    * NODE: the parameter will get an node property value;
  - PROPERTY: property of the object (node/edge) to bind to parameter.

An example

Lets try to create a network that shows where the customers of Mexico usually go shopping.

Data Here, in Code 4.10 the query on the foodmart demo data:

```
  SELECT s.store_city store
        ,c.city customer
        ,c.city customer_city
```

```
        ,count(*) number_sales
        ,((length(s.store_city) * 7) + 10) textlenght
        ,CONCAT (s.store_city,'-',c.city) rel_id
  FROM sales_fact_1998 sf
  JOIN customer c ON (c.customer_id = sf.customer_id)
  JOIN store s ON (s.store_id = sf.store_id)
  WHERE c.country = 'Mexico'
  GROUP BY store
        ,customer
        ,rel_id

```

Code 4.10: Foodmart demo data.

Now we can collect all these information and build our first network template. In our example the nodes are the cities and the relations represent where the customer of a city go to shop. Code 4.11 shows a simply template for this document:

```
<NET>
   <NETWOK_DEFINITION>
        <options pan_Zoom_Control_Position="topLeft">
        </options>

        <dataset_mapping_LIST>
          <dataset_mapping element="source" column="customer" property="id"/>
          <dataset_mapping element="target" column="store" property="id"/>
          <dataset_mapping element="edge" column="rel_id" property="id"/>
          </dataset_mapping_LIST>
   </NETWOK_DEFINITION>

</NET>

```

Code 4.11: Template for foodmart demo.

Now we try to make the graph "nicer". We want to:

- see the name of the cities,
- see the number of sales of customers coming from city A to shop in city B,
- add some image as background of the nodes

The template will look like Code 4.12:

```
<NET>
```

```
<NETWOK_DEFINITION>
        <options edgeLabelsVisible="true" pan_Zoom_Control_Position="topLeft" nodeTooltipsEnabled="true" layout="Circle">
         <visual_style>
                <edges directed="true">
                <label>
                <passthrough_Mapper  attrName="number_sales"/>
                </label>
                </edges>
         </visual_style>
         </options>

        <dataset_mapping_LIST>
                <dataset_mapping element="source" column="customer" property="id"/>
                <dataset_mapping element="source" property="size" value="50"/>
                <dataset_mapping element="source" column="customer_city" property="label"/>
                <dataset_mapping element="source" property="image" value="../img/city2.png"/>
                <dataset_mapping element="source" property="labelFontSize" value="12"/>
                <dataset_mapping element="source" property="labelFontWeight" value="bold"/>
                <dataset_mapping element="target" column="store" property="id"/>
                <dataset_mapping element="target" property="labelFontWeight" value="bold"/>
                <dataset_mapping element="target" property="labelFontSize" value="12"/>
                <dataset_mapping element="edge" column="rel_id" property="id"/>
                <dataset_mapping element="edge" column="number_sales" property="number_sales"/>
                <dataset_mapping element="edge" value="ARROW" property="sourceArrowShape"/>

        </dataset_mapping_LIST>
        </NETWOK_DEFINITION>

</NET>
```

Code 4.12: Improved template for foodmart demo.

Remark: The path ../img/city2.png is relative to the context of the web application, so it refers to the folder img inside the web application knowagenetworkengine

Finally, the result is showed in next Figure 4.156:

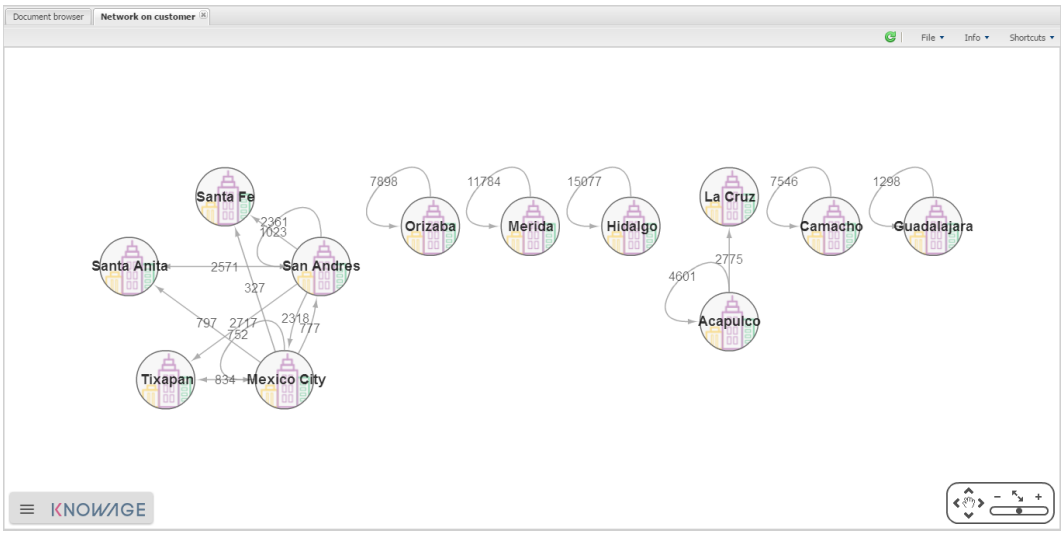


Figure 4.156: Network for foodmart demo example.

4.14 My first SVG Map or design

The SVG Viewer Engine is a tool that lets you develop documents based on the SVG, acronym for Scalable Vector Graphics, format. It permits to show different business information directly on each area, and permits the drill action to other more detailed SVG files using a logical hierarchy. This viewer is divided into two sections:

- a panel with many dynamic details such measures, layers and legend plus an optional section with specific information about the active document,
- the svg document.

To give an example, we can imagine to visualize throught an SVG the USA map. At first we can show data at the "Regions" level and then throught the click / drill - show the same or other information at the States "level". We give an example of map document produced with the SVG engine in Figure 4.157 and Figure 4.158.

Like other Knowage documents type there is a set of activities managed by the technical users and others used by the final users. These last ones are specifically about consulting.

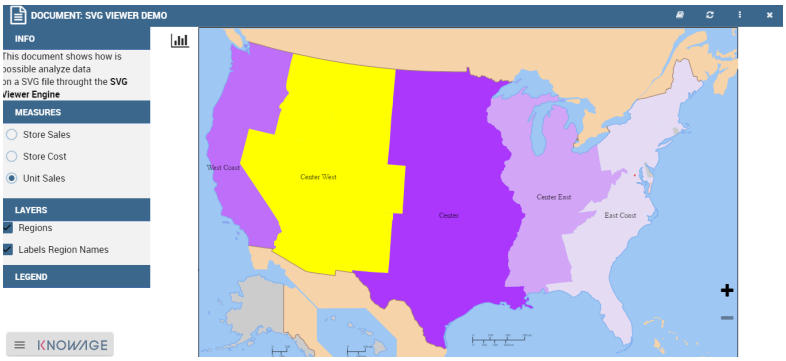


Figure 4.157: SVG document example at the USA Regions level.

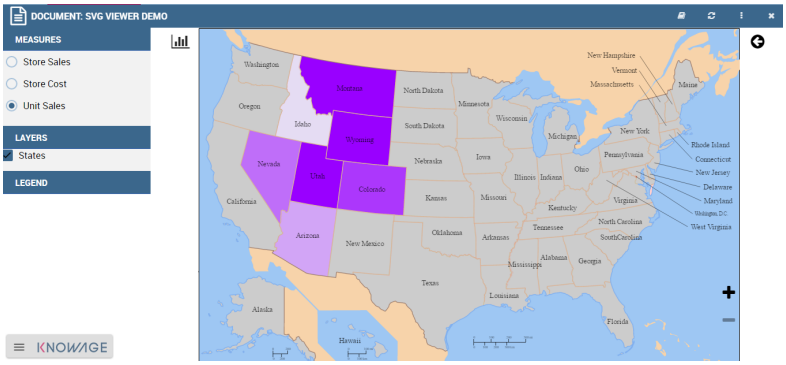


Figure 4.158: SVG document example at the States level after the selection of the “Center West” Region.

Technical activities

First of all, a technical user needs to configure the logical hierarchy of the SVG and to define datasets with the business data he/she wishes to show. Finally he/she must type the document template. We will give details about these points in the following sections.

SVG Catalogue

The first activity that you need to do as administrator is to find or create an SVG file. Any file saved in SVG format is a text file in XML format. As a consequence, they can easily be queried, indexed, enriched with scripts and, if necessary, zipped. The SVG final output could represent everything: geographical areas (like USA in the previous example in Figure 4.157), concepts (like the item production steps) and so on.

SVG Format

The Scalable Vector Graphics, SVG, refers to an XML-based encoding format, used to describe two dimensional vector graphical objects. SVG is an open standard, defined by the World Wide Web Consortium (W3C), which released its first version in 1999. Read more at http://www.w3.org/Graphics/SVG/.

Not all graphical objects of an SVG can be thematized. Using the SVG grouping operator <g>, the developer can create one or more subsets of graphical objects and specify which groups should be subject to thematization. Each group has an unique name, corresponding to the value of the id attribute of the <g> tag (e.g. <g id="regions">). Considering that, graphical objects grouped in an SVG file are usually homogeneous elements (in other words, they model a same category of objects: regions, towns, streets, etc.), we can consider these groups as layers and the objects can be considered as features.

Once obtained the SVG file, you should register it into Knowage SVG catalogue.

The Svg catalogue contains all SVG that can be used with this engine throught specific hierarchies. In this context a hierarchy is a definition of three concepts:

- the hierarchy itself,
- the level,
- the member.

These three information are used from the system to recover the correct SVG into the catalogue.

As you can see in Figure 4.159, you must insert a name and an optional description of the new SVG component, then you need to specify a logical hierarchy's label, its number of the level and a logical name for the member that it represents. At last you need to upload the SVG

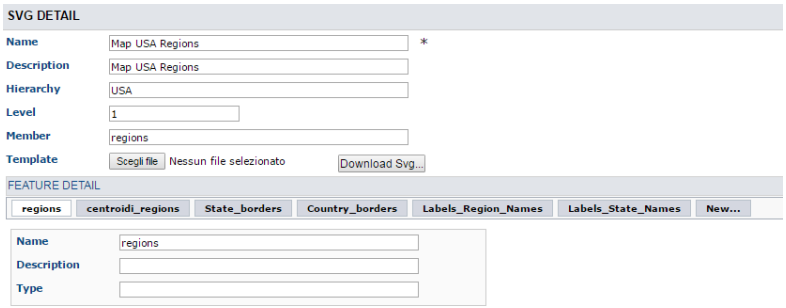


Figure 4.159: Entering the hierarchy details.

file. When this configuration will be saved, the system will read the SVG content and for each group (or tag <g>) will be created a layer. All layers will be shown into the "Feature Detail" section (read only section).

In this first example in Figure 4.159 we defined an SVG component for the USA regions specifying that it's the first level (in other words it's the first SVG of the "USA" hierarchy).

The second level (the more detailed SVG) is about the USA states and it's defined like the next example in Figure 4.160:

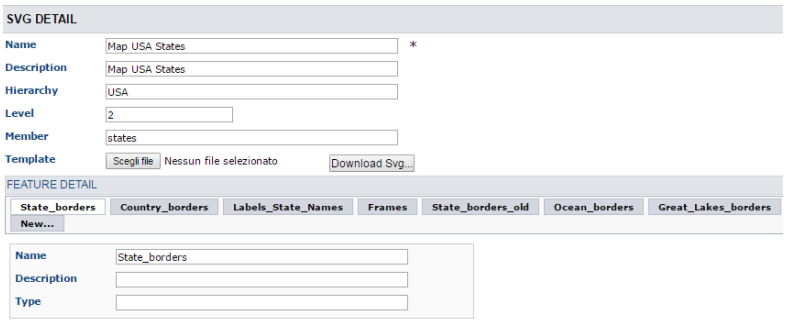


Figure 4.160: Entering the hierarchy details.

As you can see the principal differences between these configurations are only about the level content and the member label. This means that both will be used in the same hierarchy's context and that from the "Regions" SVG will be possible to drill on the "States" SVG. Anyway it is not mandatory to define more than one level: it depends from each project implementation.

Datasets definition

After that all SVG was loaded, you must define a dataset (one for each level) that you want to use for getting and showing business information from your DWH. Here in Figure 4.161 a dataset of our example:

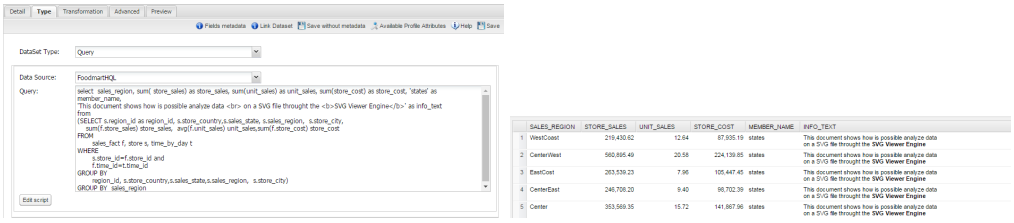


Figure 4.161: Left. Dataset behind the SVG document. Right. Dataset preview.

Template building

The template allows the SVG viewer to properly join business data (Knowage dataset) and spatial data (SVG included in the catalog), in order to produce the analytical documents.

At the moment there is not yet a designer to create a template for this engine, anyway, it's an XML file very simple to define.

An example is in Code 4.14.

```
<?xml version="1.0" encoding="UTF-8"?>
 <MAP>
   <DATAMART_PROVIDER>
     <HIERARCHY name="USA">
        <MEMBER name ="regions" measure_dataset = "ds_regions" level="1" >
        <MEMBER name ="states" measure_dataset = "ds_states" level="2" >
     <HIERARCHY>
   <DATAMART_PROVIDER>
 <MAP>

```

Code 4.13: Example of SVG code for template file.

Basically, it's necessary to specify the hierarchy that we want to use, as well as its members (remember that with member we are considering a specific SVG).

We recap in Table 4.2 the meaning of the main tag.

After, we need to define each member and first of all we can note that is composed by three sections: METADATA, LAYERS and MEASURE, as in Code 4.14:

Tag	Property	Note
HIERARCHY	name	Mandatory. The name of the hierarchy that we want use. The name must match to an existing hierarchy into the SVG catalogue.
MEMBER	name	Mandatory. The name of the member that we want use. The name must match to an existing member for the hierarchy specified into the SVG catalogue. Is too possibile get its value dinamically throught an analytical driver by using the standard syntax `$P<driver_url>`
MEMBER	`measure_dataset`	Mandatory. The label of the dataset defined in Knowage Dataset configuration.
MEMBER	level	Mandatory. The number of the level. This value must match the level property into the catalogue for the hierarchy and the member specified.

Table 4.2: Recap of tag properties and function.

```
<MEMBER name ="regions" measure_dataset = "ds_regions" level="1" >
<METADATA>
<LAYERS>
<MEASURES default_kpi="UNIT_SALES">
<MEMBER>

```

Code 4.14: Example of SVG code for template file.

Let us see each of them in more depth.

- **METADATA.** This is the section where we define the dataset metadata, in fact, each COLUMN tag defines the dataset columns that we want to use as attribute, as measure (used for thematize the SVG) or other technical meaning usefull for the engine.

```
<METADATA>
<COLUMN TYPE="geoid" column_id="sales_region" />
<COLUMN TYPE="measure" column_id="store_sales" />
<COLUMN TYPE="measure" column_id="store_costs" />
<COLUMN TYPE="measure" column_id="unit_sales" />
<COLUMN TYPE="drillid" column_id="member_name" />
<COLUMN TYPE="info" column_id="info_text" />

```

Code 4.15: Example of SVG code for template file.

Once again we give some details on metadata in next Table 4.6.

- **LAYERS.** In this section we define all layers that we want to enable in the document. Each layer will be shown into the detail panel "Layers section" as you can see in Figure 4.162 and could be actived or disactived directly by an action of the the final user. At least one layer must be defined.

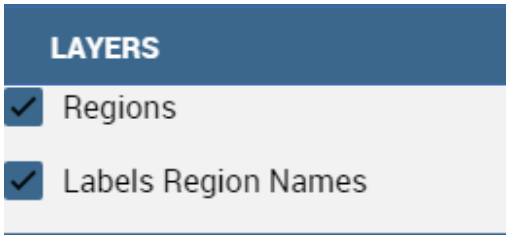


Figure 4.162: Available layers set by a technical user.

```
<LAYERS>
<LAYER name="regions" description="Regions" selected="true" />
```

Tag	Property	Note
COLUMN	TYPE	Mandatory. The type of the specific column. Possible values are: – **`geoid`**: mandatory. The engine uses this column to join the dataset records and the corresponding features in the svg. Also, it's the default value passed within the drill action to the svg of lower level (alternatively to the drillid property). – **`measure`**: mandatory. Defines the column like measure. All measures defined in this section will be shown into the detail panel (Measure section). – **`drillid`**: optional. Defines the alternative value to pass within the drill action to the next svg – **`parentid`**: optional. Defines the column that the system need to use for get correctly data linked to the parent value selected. – **`crosstype`**: optional. Defines the column that set the cross navigation type. Possible values are "cross" for external navigation or "drill" for internal navigation. If the single element returns null the link will be disabled – **`visibility`**: optional. Defines the column that throught a boolean value (string with "true" / "false") guides the visibility of each svg element. – **`label`**: optional. Defines the column with dynamic label to show on each svg element. – **`info`**: optional. Defines the column that contain a static detail to show on the Info section into the detail panel.
COLUMN	`column_id`	The dataset label that we want to use in according to the previous type setting.

Table 4.3: Recap of column tag properties and function.

```
    <LAYER name="Labels_Regions_Name" description="Labels_Regions_Name"
    selected="false" />
    <LAYERS>

```

Code 4.16: Code relative to the LAYER setting.

Tag	Property	Note
LAYER	name	Mandatory. The layer name. Mandatory. It must exists into the SVG document/catalogue (as tag <g).
LAYER	Description	Mandatory. The label that you want show into the detail panel.

Table 4.4: Recap of layer tag properties and function.

- **MEASURES** Measures are all the business values (KPI) that the user want to monitor throught this document type. Each measure defined in this section will be shown into the detail panel ("Measures" section) with a specific thematization and could be enabled or disabled directly by an action of the the final user. When the measure is active all its values are shown onto the SVG and each area has a specific tonality of the color in according to the threshold definition and its real value. All thresholds range are visualized into the "Legend" section of the detail panel as highlight in Figure 4.163. Is possibile to choose the thematization logic that it could be as quantile, percentage, uniform or static. Next, we'll see both definitions (see Thresholds details).Remember, that at least one measure must be defined.

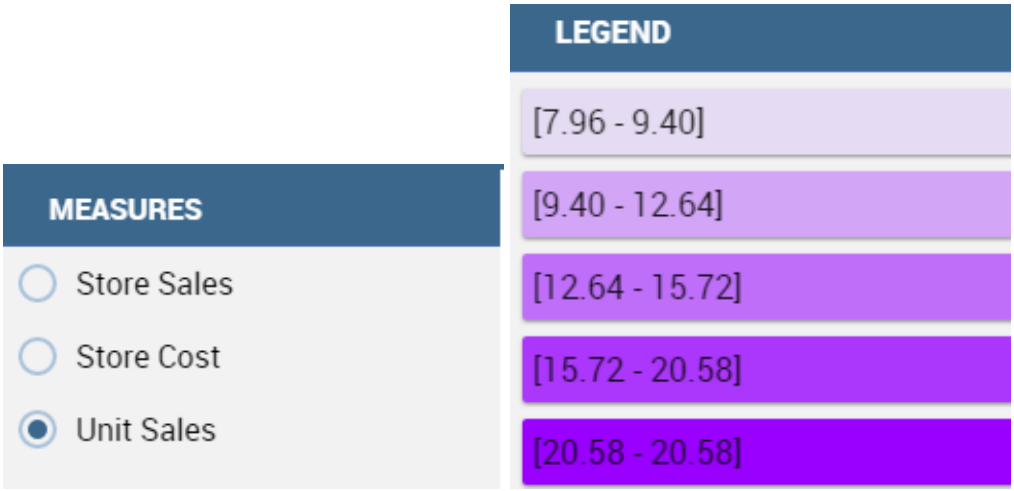


Figure 4.163: Left. Measure panel. Right. Legend panel.

```
<MEASURES default_kpi="UNIT_SALES">
  <KPI column_id="STORE_SALES" description="Store Sales" >
    <TRESHOLDS type="quantile" lb_value="0" ub_value="none" >
    <PARAM name="GROUPS_NUMBER" value="5" />
    </TRESHOLDS>
    <COLOURS type="grad" outbound_colour="#FFFFFF" null_values_color="#
    CCCCCC" >
    <PARAM name="BASE_COLOR" value="#009900" />
    <!--<PARAM name="opacity" value="0.5" />-->
    </COLOURS>
  </KPI>

  <KPI column_id="STORE_COST" description="Store Cost" >
  <KPI column_id="UNIT_SALES" description="Unit Sales" >
<MEASURE>

```

Code 4.17: Code for setting the KPI into SVG document.

We report Table 4.7 for further details on THRESHOLDS and COLOURS tag. This table includes the heuristics supporting value interval partition into a finite number of sub-intervals (type attribute of the THRESHOLDS tag).

While the following Table 4.8 defines the heuristics supporting color definition for each value sub-interval (type attribute of the COLOURS tag).

Sometimes users need to color the map and, at the same time, to continue to see the underlying objects, through a transparency effect (e.g. a raster image). In this case, specify the opacity parameter in order to properly regulate the transparency level of colors (1 = no transparency; 0 = invisible).

Now, after the template definiton, you can create it into Knowage. Remember that it must be a “Location Intelligence” document type with the engine “SVG Viewer Engine”.

Advanced functionalities

Other the default drill navigation that you have if for the document are defined more than one member, is it possible to cross versus other Knowage documents. To enable this feature, is necessary to set the enableExternalCross property for the MEMBER tag. Here an example in Code 4.18:

```
<MEMBER name="states" measure_dataset="ds_states"  level="2"
    enableExternalCross="true">
```

Tag	Property	Note
MEASURES	`default_kpi`	Mandatory. Defines the default kpi or the kpi that we wont enable at the beginning, when we start the document execution. Its value must exist into the METADATA section as measuere type.
KPI	`column_id`	Mandatory. The `column_id` property of the measure that you are defining. Its value must exist into the METADATA section as measuere type.
KPI	Description	Mandatory. The label that you want show into the detail panel.
THRESHOLDS	type	Mandatory. The type of logic to use to define the thematization. It could be: – **`quantile`**: it partitions the interval into N quintiles. – **`perc`**: it partitions the interval into subintervals whose extent represents a specific fraction of the overall interval extent. – **`uniform`**: it partitions the interval into N subintervals of a same extent. – **`static`**: it partitions the interval into smaller fixed-size subintervals, statically defined by the RANGE parameter
THRESHOLDS	`lb_value`	Mandatory. The lower value outside of which no value is considered.
THRESHOLDS	`ub_value`	Mandatory. The upper value outside of which no value is considered.
PARAM	name	Mandatory. Specify the parameter value necessaty to define correctly the temhatization. Its value depends by the threshold type. This attribite could be present more than once.
PARAM	value	Mandatory. It's the parameter name value.
PARAM	label	Optional. Specify the static labels for the legend when thresholds type is "static".
PARAM	value	Optional. It's the parameter label value.
COLOURS	type	Mandatory. Specify the logic type for defining colors range. It could be: – **`static`**: it assigns each sub-interval a specific color that is statically defined. – **grad**: it assigns each sub-interval a specific color that is dynamically calculated through a gradient function.
COLOURS	`outboud_color`	Mandatory. Defines the color to use when the value for the specific area is outbound of the maximum range.

Table 4.5: Recap of layer tag properties and function.

Tag	Property	Note
COLOURS	null_values_color	Mandatory. Defines the colour to use when the value for the specific area is null.
PARAM	namv	See the PARAM | name property specified for the THRESHOLD tag.
PARAM	nalue	See the PARAM | value property specified for the THRESHOLD tag.

Table 4.6: Recap of layer tag properties and function.

Tag	Property	Note
type	static	It partitions the interval into smaller fixed-size subintervals, statically defined by the RANGE parameter <TRESHOLDS type="static" lb_value="0" ub_value="none" > <PARAM name="range" value="0,256,512,1024" /> <PARAM name="label" value="Low,Medium,High,Max" /> </TRESHOLDS>
type	quantile	it partitions the interval into N quintiles. The exact amount of quintiles to be created is defined by the GROUPS_NUMBER parameter: <TRESHOLDS type="quantile" lb_value="0" ub_value="none" > <PARAM name="GROUPS_NUMBER" value="5" /> </TRESHOLDS>
type	perc	it partitions the interval into subintervals whose extent represents a specific fraction of the overall interval extent. The extent of each single subinterval is defined by the RANGE parameter. <TRESHOLDS type="perc" lb_value="0" ub_value="none" > <PARAM name="range" value="30,20,30,20" /> </TRESHOLDS>
type	uniform	it partitions the interval into N subintervals of a same extent. The exact number of sub-intervals is defined by the GROUPS_NUMBER parameter. <TRESHOLDS type="uniform" lb_value="0" ub_value="none" > <PARAM name="GROUPS_NUMBER" value="4" /> </TRESHOLDS>

Table 4.7: Recap of layer tag properties and function.

Tag	Property	Note
type	static	Static: it assigns each sub-interval a specific color that is statically defined, through the RANGE parameter `<COLOURS type="static" null_values_color="#FFFFFF" > <PARAM name="range" value="#CCD6E3,#6699FF,#4a7aaf,#283B64" /> </COLOURS>`
type	grad	Gradient : it assigns each sub-interval a specific color that is dynamically calculated through a gradient function, which progressively scales the base color intensity. This is defined through the `BASE_COLOR` parameter `<COLOURS type="grad" outbound_colour="#CCCCCC" null_values_color="#FFFFFF" > <PARAM name="BASE_COLOUR" value="#3333CC" /> </COLOURS>`

Table 4.8: Recap of layer tag properties and function.

```
2
```

Code 4.18: Code for enabling external cross navigation.

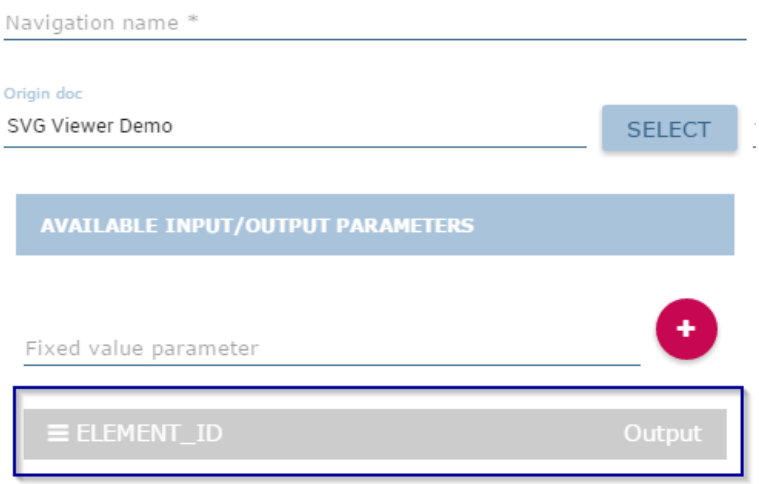


Figure 4.164: Using the Cross Navigation definition to link to external documents.

With this setting, you are able to create a “Cross Navigation Definition” with the standard Knowage functionality, where for default you will find the `element_id` as outuput parameter

as shown in Figure 4.164. It means that the identifer of the area selected is able to be passed. Other default output parameters are **`Hierarchy`**, **`Member`** and **`Level`**.

In a cross navigation it is also possible to pass the dataset column values. It is only necessary that a technical user prepares specific output parameters, setting the name like the alias of the dataset columns.

ENGINEERING

Knowage
Engineering Group