
Knowage

Knowage Labs

Aug 30, 2022

Table of Contents

1	Before starting	3
1.1	Documentation structure	3
1.2	Target	3
1.3	About conventions	4
2	Installation Manual	5
2.1	Goal of the document	5
2.2	Requirements	5
2.2.1	Operating systems	5
2.2.2	Disk usage	6
2.2.3	Java environment	6
2.2.3.1	Linux	6
2.2.3.2	Windows	6
2.2.4	Application server	6
2.2.4.1	Tomcat 8.5	6
2.2.4.1.1	Linux	8
2.2.4.1.2	Windows	8
2.2.5	Database schema for metadata	8
2.2.6	Database schema for data	8
2.2.7	NodeJS requirements	9
2.2.7.1	CentOS	9
2.2.7.2	Ubuntu	9
2.2.8	Chromium requirements	10
2.3	Software release	10
2.4	Manual installation	11
2.4.1	Metadata database initialization	11
2.4.2	Using Tomcat	11
2.4.2.1	Dependencies	11
2.4.2.2	File system resources	11
2.4.2.3	Connection to metadata database	12
2.4.2.4	Cache database connection	12
2.4.2.5	Connection to business data	12
2.4.2.6	Environment variables definition	13
2.4.2.7	Changing the secret key for password encryption	14
2.4.2.8	Recommended configuration	14
2.4.2.9	Applications deploy	14

2.4.2.10	Thread pool definition	14
2.4.2.11	Advanced memory settings	14
2.4.2.12	Advanced Connector settings	15
2.4.3	Datasource link within the applications	15
2.4.4	Configuration of the metadata db dialect	15
2.4.5	Modification of the Quartz configuration	16
2.4.5.1	Clustering	16
2.4.6	Logging	17
2.4.7	Enable Java Security Manager	17
2.4.8	Installation of Chromium Cockpit Export script	17
2.5	How to upgrade to the latest version	18
2.5.1	Preliminary operations	18
2.5.2	Upgrade operations	18
2.6	Knowage CE Installer	20
2.6.1	Server-side requirements	20
2.6.1.1	Operating system	20
2.6.1.2	Java platform	20
2.6.1.3	Memory	20
2.6.1.4	Disk usage	20
2.6.1.5	Database	21
2.6.1.6	Application server	21
2.6.1.7	Proxy settings	21
2.6.2	Client-side requirements	21
2.6.2.1	Browser	21
2.6.2.2	Proxy settings	21
2.6.3	Launching	21
2.6.3.1	Windows	21
2.6.3.2	Linux/macOS	22
2.6.4	Managing Knowage CE	22
2.6.4.1	Windows	22
2.6.4.2	Windows (embedded MariaDB option)	22
2.6.4.3	Linux/macOS	22
2.7	R installation	23
2.8	Python Widget and DataSet	23
2.8.1	Install knowage-python webservice	25
2.8.2	Run knowage-python webservice	26
2.9	R Widget and DataSet	26
2.9.1	Install knowage-r webservice	26
2.9.2	Run knowage-r webservice	26
2.10	Configure Knowage to enable Python/R functionalities	27
2.11	DataMining Engine	27
2.11.1	JPY installation	27
2.12	CAS installation	29
2.12.1	Deploy of the CAS application	29
2.12.2	HTTPS certificate	29
2.12.3	Configuration of the HTTPS protocol for Tomcat	30
2.12.4	Knowage configuration	30
2.13	Advanced configuration	32
2.13.1	Thread manager	32
2.13.2	Cache parameters	33
2.13.3	Logging	33
2.13.4	Mail server	34
2.13.5	Maximum file size	35
2.13.6	Date format	36

2.13.7	Language	36
2.13.8	LDAP security connectors	36
2.13.9	Google Sign-In integration	38
2.13.10	Password constraints settings	39
2.13.11	Login security settings	40
2.13.12	Resource export folder cleaning settings	41
2.13.13	Import / Export	41
2.13.13.1	Users	41
3	Administration Manual	43
3.1	Knowage at a glance	43
3.1.1	Discovering Knowage	43
3.1.2	Modules	43
3.1.3	Layers	45
3.1.3.1	Delivery layer	45
3.1.3.2	Analytical layer	47
3.1.3.3	Data layer	47
3.1.4	What you can do with Knowage	47
3.1.4.1	Analytical and operational functionalities	48
3.1.4.2	Administrative tools and cross services	48
3.2	User Interface	48
3.2.1	Main menu	48
3.2.1.1	My Account menu	51
3.2.1.1.1	Modify	51
3.2.1.1.2	Delete account	51
3.2.1.1.3	Password change	51
3.3	Configure data sources	51
3.3.1	Connect to your data	51
3.3.2	Big Data and NoSQL	55
3.3.2.1	Hive	55
3.3.2.2	Spark SQL	56
3.3.2.3	Impala	57
3.3.2.4	MongoDB	57
3.3.2.5	Cassandra	58
3.4	Behavioural Model	58
3.4.1	Roles, users and attributes	59
3.4.1.1	User Management	62
3.4.1.1.1	User unlock	62
3.4.1.1.2	Roles settings	62
3.5	Analytical Model	64
3.5.1	Repository structure and rights	64
3.5.2	Menu configuration	66
3.5.2.1	Setting menu icon	66
3.6	Operational engines	67
3.6.1	ETL	67
3.6.1.1	KnowageTalendEngine	71
3.6.1.2	Job design	71
3.6.1.3	Template building	72
3.6.1.4	Creating the analytical document	73
3.6.1.5	Job execution	73
3.6.1.6	Job scheduling	73
3.6.2	External processes	73
3.6.2.1	Class definition	74
3.6.2.2	Template definition	75

3.7	Scheduler	76
3.7.1	Create an Activity	77
3.7.1.1	Save as snapshot	80
3.7.1.2	Save as file	81
3.7.1.3	Save as document	82
3.7.1.4	Send to Java class	82
3.7.1.5	Send mail	84
3.7.1.5.1	Static list	84
3.7.1.5.2	Dynamic list with mapping dataset	84
3.7.1.5.3	Dynamic List with script	86
3.7.2	Scheduling panel	86
3.7.3	Scheduler Monitor	87
3.8	Server Manager	87
3.8.1	Tenants Management	89
3.8.2	Template Management	89
3.8.3	Import/Export	91
3.8.3.1	Documents	91
3.8.3.2	Menu	95
3.8.3.3	Users	97
3.8.3.4	Catalogs	99
3.8.3.5	KPIs	100
3.8.3.6	Analytical Drivers	102
3.8.3.7	Glossary	104
3.8.3.8	Catalog	106
3.9	Server Settings	106
3.9.1	Configuration Management	106
3.9.2	Domain Management	108
3.9.3	Exporters Menu	109
3.9.4	Metadata	110
4	User Guide	113
4.1	Introduction	113
4.2	User Interface	113
4.2.1	Preliminary information	113
4.2.2	Main menu	114
4.2.3	Document Browser overview	117
4.3	Document Execution	119
4.3.1	Parameters management	119
4.3.2	Document Toolbar	119
4.3.2.1	Exporters	121
4.3.2.2	Business and structural metadata	121
4.3.2.3	Notes	122
4.3.2.4	Rate document	123
4.3.2.5	Scheduled Execution	123
4.4	User registration	123
5	Functionalities	127
5.1	Basic Data Access	127
5.1.1	Dataset	127
5.1.1.1	My dataset	128
5.1.1.2	CKAN integration	131
5.1.1.2.1	CKAN datasets access method	131
5.1.1.2.2	Export dataset	132
5.1.1.2.3	Save and handle dataset	132

5.1.2	Models	132
5.1.3	Dataset federation	134
5.2	Advanced Data Access	134
5.2.1	My first dataset	134
5.2.1.1	New dataset creation	136
5.2.1.2	File Dataset	139
5.2.1.3	Query Dataset	139
5.2.1.4	Java Class Dataset	141
5.2.1.5	Script	141
5.2.1.6	QbE	141
5.2.1.7	Custom Dataset	143
5.2.1.7.1	Flat Dataset	143
5.2.1.7.2	Ckan	143
5.2.1.7.3	Federated	145
5.2.1.7.4	Rest	145
5.2.1.7.5	Python/R	149
5.2.1.7.6	Big Data - NoSQL	149
5.2.1.7.7	Solr Dataset	151
5.2.2	Parameters and profile attributes	153
5.2.3	Further operations on a dataset	155
5.2.3.1	Script option	155
5.2.3.2	Transformations	156
5.2.3.3	Dataset persistence	159
5.2.3.4	Preview	159
5.3	Behavioural Model	159
5.3.1	Roles, users and attributes	160
5.3.2	Analytical drivers	160
5.3.2.1	Creating a List Of Value	165
5.3.2.2	Parametrizing LOVs	170
5.3.2.3	Creating a validation rule	171
5.3.2.4	Creating an analytical driver	172
5.3.2.5	Creating an analytical driver for a spatial filter	177
5.3.2.6	Analytical driver's use modes	177
5.3.2.7	Behavioural Model Lineage	178
5.4	Analytical Document	180
5.4.1	Main concepts	180
5.4.1.1	Document types	180
5.4.2	Register an analytical document	182
5.4.2.1	Analytical documents on Server	182
5.4.2.1.1	Document lifecycle	186
5.4.2.1.2	Template Versioning	186
5.4.2.1.3	Document Visibility	187
5.4.3	Visibility rules	187
5.4.4	Association with analytical drivers	188
5.4.4.1	Associating a Spatial driver	189
5.4.4.2	Correlation between parameters	191
5.4.4.3	Correlation through LOV and drivers	193
5.4.4.4	Controlled visibility	193
5.4.5	Cross Navigation	195
5.4.5.1	Declaration of the output parameters	196
5.4.5.2	Cross navigation definition	196
5.5	Chart	199
5.5.1	My first Chart	200
5.5.1.1	Structure	204

5.5.1.2	Configuration	204
5.5.1.3	Advanced options	207
5.5.2	Chart types in detail	208
5.5.2.1	Traditional charts	208
5.5.2.2	Scatter chart	211
5.5.2.3	Sunburst chart	211
5.5.2.4	Wordcloud chart	215
5.5.2.5	Treemap chart	215
5.5.2.6	Parallel chart	216
5.5.2.7	Heatmap chart	217
5.5.2.8	Chord chart	220
5.5.2.9	Gauge chart	220
5.5.2.10	Bubble chart	221
5.5.3	A short comment on chart drill down	225
5.5.4	Stand alone charts	225
5.5.5	Cross Navigation	227
5.6	Cockpit	229
5.6.1	My first Cockpit	229
5.6.1.1	Text widget	232
5.6.1.2	Image widget	235
5.6.1.3	Chart widget	236
5.6.1.4	Table widget	236
5.6.1.5	Cross Table widget	246
5.6.1.6	Document section	253
5.6.1.7	Selection widget	253
5.6.1.8	Selector Widget	254
5.6.1.9	HTML Widget	257
5.6.1.10	Map Widget	260
5.6.1.11	Discovery Widget	266
5.6.1.12	Python/R Widget	269
5.6.1.13	Custom Chart Widget	272
5.6.1.14	Cross Navigation	284
5.6.1.15	Widget properties	286
5.6.2	General configuration	287
5.6.3	Data configuration	289
5.6.3.1	Source	289
5.6.3.1.1	Parametric sources management	289
5.6.3.2	Associations	291
5.6.3.3	Indexes	292
5.6.3.4	Frequency	293
5.6.3.5	Variables	293
5.6.3.6	Template	294
5.6.4	Selections	294
5.6.5	Clear cache	294
5.6.6	Save	294
5.6.7	Multisheet functionality	296
5.6.8	Export cockpit	296
5.7	Free Inquiry	298
5.7.1	My first Query By Example	298
5.7.1.1	Query design and execution	298
5.7.1.1.1	Datamart Schema	299
5.7.1.1.2	Calculated fields management	299
5.7.1.1.3	Query Editor	301
5.7.1.1.4	Selected Fields	301

	5.7.1.1.5	Filters	304
	5.7.1.1.6	Query Preview	306
	5.7.1.2	Advanced QbE functionalities	309
	5.7.1.2.1	Spatial fields usage	309
	5.7.1.2.2	Time functions for creating calculated fields	315
	5.7.1.2.2.1	Subqueries	320
	5.7.1.2.2.2	Parameters	320
5.8	Meta Web		321
5.8.1	Metamodel creation		321
5.8.2	Association with analytical drivers		322
5.8.2.1	Correlation between parameters		323
5.8.2.2	Correlation through LOV and drivers		326
5.8.2.3	Create an empty model		326
5.8.2.4	Editing the metamodel		328
5.8.2.5	Create a new relationship		330
5.8.2.6	SQL Filter		332
5.8.2.7	Create a new business class		332
5.8.2.8	Table property list		334
5.8.2.9	Column property list		334
5.8.2.10	Generate the datamart		337
5.8.2.11	Additional functions for business model		339
5.9	Registry		339
5.9.1	Registry features		339
5.9.1.1	JPivot Registry characteristics		343
5.9.2	Registry development		344
5.9.2.1	JPivot Registry instance		345
5.10	Create a Birt report		346
5.10.1	Developing a BIRT report		346
5.10.1.1	Adding parameters to reports		351
5.10.2	Cross Navigation for BIRT Reports		356
5.11	Jasper reporting		359
5.11.1	Document definition*		359
5.12	Location Intelligence		363
5.12.1	Basic concepts		363
5.12.2	More on GIS and Spatial Data*		364
5.12.2.1	Spatial Data		364
5.12.2.2	GIS		365
5.12.3	Analytical document execution		366
5.12.3.1	Extra functionalities		370
5.12.4	Template building with GIS designer		370
5.12.5	Designer sections		372
5.12.5.1	Layer section		372
5.12.5.2	Dataset join columns		372
5.12.5.3	Indicators		373
5.12.5.4	Filters		373
5.12.5.5	Map menu configuration		374
5.12.5.6	Layer filters		374
5.12.5.7	Edit map		375
5.12.6	GEOReport Engine*		375
5.12.7	Template building with GIS designer for technical user*		378
5.12.8	Cross navigation definition*		378
5.13	SVG document visualization		380
5.13.1	My first SVG Map or design		380
5.13.1.1	Technical activities		383

5.13.1.1.1	SVG Catalogue	383
5.13.1.1.2	SVG Format	383
5.13.1.1.3	Datasets definition	385
5.13.1.1.4	Template building	385
5.13.1.1.5	Advanced functionalities	391
5.14	Data Mining	391
5.14.1	Data Mining document interface	392
5.14.2	Functions Catalog	397
5.14.2.1	The General tab*	401
5.14.2.2	The Input tab*	401
5.14.2.3	The Script tab*	402
5.14.2.4	The Output tab*	406
5.14.3	Engine description	409
5.14.3.1	Data	409
5.14.3.2	Parametrization and customization	409
5.14.3.3	Outputs	409
5.14.4	My first data mining document*	409
5.14.5	Create a new function in the Function Catalog	411
5.15	Multidimensional Analysis	413
5.15.1	OLAP user manual step by step	414
5.15.1.1	The filter panel	414
5.15.1.2	The filter cards	414
5.15.1.3	Axes panel	416
5.15.1.4	Pivot table	416
5.15.1.5	Side bar	418
5.15.2	Functionalities	424
5.15.2.1	Placing hierarchies on axes	424
5.15.2.2	Swaping axes	424
5.15.2.3	Selecting different hierarchies on dimension	426
5.15.2.4	Slicing	427
5.15.2.5	Filtering	428
5.15.2.6	Drill down and drill up	428
5.15.2.7	Drill through	431
5.15.2.8	Refreshing model	436
5.15.2.9	Showing MDX	436
5.15.2.10	Sending MDX	436
5.15.2.11	Showing parent members	437
5.15.2.12	Hiding/showing spans	437
5.15.2.13	Showing properties	437
5.15.2.14	Suppressing empty columns/rows	437
5.15.2.15	Sorting	437
5.15.2.16	Calculated members and sets	442
5.15.3	Creation of an OLAP document*	448
5.15.3.1	About the engine	448
5.15.3.2	Development of an OLAP document	448
5.15.3.2.1	Schema modelling	449
5.15.3.2.1.1	Engine catalogue configuration	450
5.15.3.2.2	OLAP template building	450
5.15.3.2.3	Creating the analytical document	452
5.15.3.3	OLAP Designer*	452
5.15.3.3.1	Profiled access	455
5.15.3.4	Cross Navigation	459
5.15.3.4.1	Cross navigation on members	460
5.15.3.4.2	Cross navigation from a cell of the pivot table	460

5.16	What-if analysis	461
5.16.1	Interface details	461
5.16.2	Meta-language description	463
5.16.3	What-if analysis implementation	466
5.16.3.1	Workflow description*	466
5.16.3.2	Schema definition*	467
5.16.3.3	Changes in the mondrian schema*	467
5.17	Glossary and data lineage	472
5.17.1	Glossary management	472
5.17.2	Glossary Usage	475
5.17.3	Help Online functionality	478
5.18	Key Performance Indicator	480
5.18.1	KPI development	480
5.18.2	Creation of a KPI document	495
5.19	Scorecard	498
5.19.1	Scorecard development	498
5.19.2	Creation of a Scorecard document	506
5.20	Alert	507
5.20.1	Alert implementation	510
5.21	News management	512
5.21.1	How to publish news	512
5.21.2	How the end user can read the news	515
5.22	Dataset Preview	516
5.22.1	Dataset preview structure	516
5.22.2	Dataset preview export	516
5.22.3	Inside workspace	518
5.22.4	Inside cockpit	518

Knowage is the professional open source suite for modern business analytics over traditional sources and big data systems. Knowage is the new brand for SpagoBI project: this new brand marks the value of the well-known open source business intelligence suite after significant functional and technological transformations and a new offering model. The suite is composed of several modules, each one conceived for a specific analytical domain. They can be used individually as complete solution for a certain task, or combined with one another to ensure full coverage of user' requirements.

The sub-products of the suite are:

- BD (big data), to analyse data stored on big data clusters or NoSQL databases
- SI (smart intelligence), the usual business intelligence on structured data, but more oriented to self-service capabilities and agile prototyping
- ER (enterprise reporting), to produce and distribute static reports
- LI (location intelligence), to relate business data with spatial or geographical information
- PM (performance management), to manage KPIs and organize scorecards
- PA (predictive analysis), for more advanced analyses

Knowage supports a modern vision of the data analytics, providing new self-service capabilities that give autonomy to the end-user, now able to build his own analysis and explore his own data space, also combining data that come from different sources. If this is your first contact with Knowage, it is highly recommended to have a look at the [website](#).

Knowage source code repository is available on GitHub! Visit the [repository](#).

1.1 Documentation structure

Knowage documentation is composed by four main areas:

- **Installation Manual;**
- **General User Manual;**
- **General Administrator Manual;**
- **Functionalities.**

Each one focuses on a specific aspect of the product: how to properly install it, how to generally approach the suite, which are the general tools an administrator has access to and which are the analytical potentialities of the platform.

1.2 Target

Each area is thought to be delivered to different users.

- **Installation Manual:** it is directed at technical users. It provides all the instructions needed to install and configure the Knowage suite. Here the essential steps to succeed with the standard installation on certified environments will be described, including the optional use of CAS as a SSO solution and the use of HTTPS protocol.
- **General User Manual:** it is directed at unskilled end users. It provides a first approach to Knowage interface and functionalities. It can be used as a first approach to Knowage. It focuses on all those elements which are shared among the products and involves the end user.
- **General Administrator Manual:** it is directed at Knowage administrator. It describes all the management and configuration tools shared by all Knowage products.
- **Functionalities:** it is directed at all kind of users. Here each user can find detailed instructions on how to develop different kind of analysis exploiting all BI capabilities of Knowage platform. Notice that when a functionality is strictly related to a specific module it will be notified.

1.3 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 here after.

Note: Read more

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

Warning: Warning

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

Hint: Advice

This icon provides best practices and suggestions.

Important: 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.

2.1 Goal of the document

The present document illustrates how to install and configure the Knowage suite. There will be described the essential steps to succeed with the standard installation on certified environments, which includes the optional use of CAS as a SSO solution and the use of HTTPS protocol.

2.2 Requirements

Before going into details on Knowage installation, it is necessary to check if certain requirements are satisfied. We start to distinguish between the certified environments and the compatible ones. The first are those where check tests take place. The latter are those environments technically compatibles but where integration tests are not executed.

2.2.1 Operating systems

The following Operating Systems (OS) are those ones which suit with Knowage platform.

Table 2.1: Certified environments

Certified Environments	
Operating System	Version
CentOS	7 64-bit
Windows	7 , 10

Table 2.2: Compatible environments

Compatible Environments	
Operating System	Version
RHEL Red Hat Enterprise	7
Ubuntu	18 LST
Windows server	2012, 2008

2.2.2 Disk usage

The Knowage installation requires 2 GB of available space on file system. This space does not include the space relative to the data and the metadata storage.

2.2.3 Java environment

The environment in which Knowage will be installed must include a JDK 1.8 installation. Be sure that the JDK component is successfully installed and that the environment variable `JAVA_HOME` is properly configured. The steps to configure it depend on the OS. Knowage 7 is compatible with Open JDK 1.8.

2.2.3.1 Linux

Define the `JAVA_HOME` variable inside the users' file `.bash_profile` used in the installation process

Listing 2.1: Instructions to set the `JAVA_HOME` variable for Linux environment.

```
1 export JAVA_HOME=<root path of the Java installation>
2 export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_60/
3 export PATH=$JAVA_HOME/bin:$PATH
```

2.2.3.2 Windows

Define the `JAVA_HOME` variable and `PATH` in the section “Environment variables” which can be reached from the “System”.

2.2.4 Application server

The following lists the supported application servers:

Table 2.3: Supported application servers

Support type	Application Server	Version
Certified	Apache Tomcat	8.5.57

For each application server installation please refer to its official documentation.

2.2.4.1 Tomcat 8.5

In the following we will refer to Tomcat installation folder as `TOMCAT_HOME`.

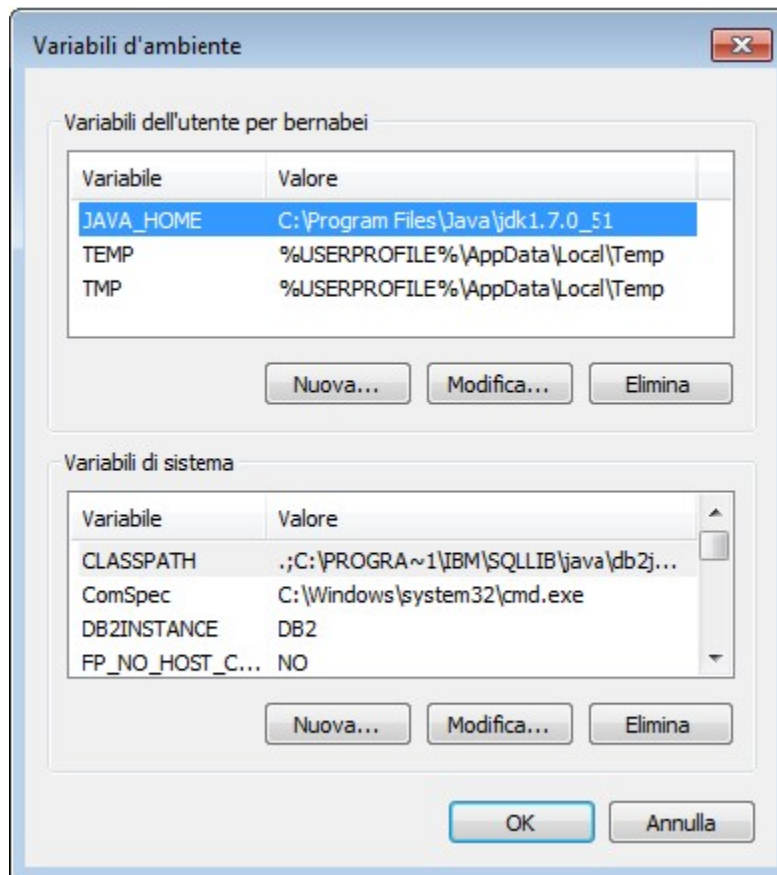


Fig. 2.1: Setting the path for the JAVA_HOME variable for Windows

2.2.4.1.1 Linux

It is recommended to create a proper user for the execution of Tomcat. We state the main steps to follow for this purpose.

- Create the Tomcat user.

```
1 useradd -m tomcat
2 passwd <password for the tomcat user>
```

- Install the Tomcat using the Tomcat user. Remeber to define the TOMCAT_HOME variable.

```
1 export TOMCAT_HOME=<path of the installation Tomcat root folder >
```

- Be sure that the Tomcat uses the JDK 1.8: usually the Tomcat settings are defined in the TOMCAT_HOME/bin/setenv.sh file, therefore if the TOMCAT_HOME/bin/setenv.sh file does not exit, the user must create it and insert it in the content as shown below. Note that CATALINA_PID contains the ID of the Tomcat process and it kills the process if needed.

```
1 export CATALINA_PID=<root folder of the Tomcat installation>/logs/tomcat-knowage.pid
2 export JAVA_HOME=<root folder of the JDK 1.8 installation>
```

- Modify the TOMCAT_HOME/bin/shutdown.sh file to force the shut down of the application in case of hanging:

```
1 exec "$PRGDIR"/"$EXECUTABLE" stop -f "$@"
```

2.2.4.1.2 Windows

It is recommended to install Tomcat as a service. Documentation is available at <https://tomcat.apache.org/tomcat-8.5-doc/windows-service-howto.html>.

2.2.5 Database schema for metadata

Knowage uses a schema to manage metadata, that is all those information required for its operation. These concern the configuration, the users and the analytical documents. It is possible to use the following DBMSs for the creation of this schema.

Table 2.4: Exploitable DBMSs for the metadata schema creation

Support Type	DBMS	Version
Certified	Oracle	8, 9, 10, 11, 12
Certified	MySQL	5.7, 8
Certified	PostgreSQL	8.2, 9.1, 12.3
Certified	MariaDB	10.1, 10.2, 10.3

Therefore, a schema must be available. It can be reached through the JDBC protocol by the Knowage installation server; such a schema will be called *metadata DB* in the following. Observe that Knowage includes all the DDL for table creation.

2.2.6 Database schema for data

A schema for data must be also available. It can be queried through Knowage and can be reached through the JDBC protocol by the Knowage installation server; such a schema will be called *data DB* in the following.

2.2.7 NodeJS requirements

Important: Enterprise Edition only

NodeJS is required only for Enterprise Edition.

Knowage includes some NodeJS scripts that need to be executed with NodeJS 8 or greater: see [NodeJS official documentation](#) for the installation process.

2.2.7.1 CentOS

In CentOS you need to erase older versions of NodeJS, if present:

Listing 2.2: Command to erase older versions of NodeJS

```
yum erase -y nodejs
```

Then you need to clear YUM cache and update all local packages:

Listing 2.3: Cache clearing and system updating

```
yum clean all
yum update -y
```

Next you can install the official repository of NodeJS:

Listing 2.4: Installation of the repository of NodeJS

```
curl -sL https://rpm.nodesource.com/setup_8.x | bash -
```

Important: If you are behind a corporate proxy, you would need to set `http_proxy` and/or `https_proxy`.

Finally you can install NodeJS:

Listing 2.5: Installation of NodeJS

```
yum install -y nodejs
```

2.2.7.2 Ubuntu

In Ubuntu you need to erase older versions of NodeJS, if present:

Listing 2.6: Command to erase older versions of NodeJS

```
apt-get remove nodejs
```

Then you need to clear APT cache and update all local packages:

Listing 2.7: Cache clearing and system updating

```
apt-get update
apt-get upgrade -y
```

Next you can install the official repository of NodeJS:

Listing 2.8: Installation of the repository of NodeJS

```
curl -sL https://deb.nodesource.com/setup_8.x | bash -
```

Important: If you are behind a corporate proxy, you would need to set `http_proxy` and/or `https_proxy`.

Finally you can install NodeJS:

Listing 2.9: Installation of NodeJS

```
apt-get install -y nodejs
```

2.2.8 Chromium requirements

Important: Enterprise Edition only

Chromium is required only for Enterprise Edition.

Knowage provides a distribution of Chromium for its functionalities but some other dependencies are needed. In Linux distribution you need to install following Chromium dependencies:

Listing 2.10: Installation of Chromium dependencies

```
# For CentOS
yum install -y at-spi2-atk cups-libs expat glib2 glibc.i686 glibc libcanberra-gtk3 libgcc libstdc++ libX11_
↳ libXScrnSaver minizip nspr nss-mdns nss-util nss policycoreutils-python policycoreutils zlib

# For Ubuntu
apt-get install -y libxss1 libasound2 libatk-bridge2.0-0 libatk1.0-0 libatspi2.0-0 libc6 libcairo2 libcups2_
↳ libdbus-1-3 libexpat1 libgcc1 libgdk-pixbuf2.0-0 libglib2.0-0 libnspr4 libnss3 libpango-1.0-0 libpangocairo-
↳ 1.0-0 libuuid1 libx11-6 libx11-xcb1 libxcb1 libxcomposite1 libxcursor1 libxdamage1 libxext6 libxf86devi
↳ libxi6 libxrandr2 libxrender1 libxtst6 bash
```

2.3 Software release

You can download Knowage through the following links:

- Community Edition: <http://www.knowage-suite.com/site/ce-download/>
- Enterprise Edition: <http://www.knowage-suite.com/portal> (registration required)

A typical release contains three elements:

- Installer, which gives you a step-by-step Knowage installation
- Web Application Archives (WARs) and DDL scripts (to populate the schema used by Knowage for its metadata)

Table 2.5: Supported databases for DDL scripts

Supported databases
MySQL / MariaDB
Oracle
Postgres

2.4 Manual installation

2.4.1 Metadata database initialization

Knowage requires a database schema to store its own metadata (definition of analyses, datasets and so on). For initializing such a schema, the user must execute the creation scripts provided for the DBMS in use. The package which includes the DDL will contain the following SQL files:

Listing 2.11: Scripts for metadata schema

```
XXX_create.sql
XXX_create_quartz_schema.sql
```

where XXX represents the DBMS type (as instance ORA stands for Oracle). The corresponding SQL files for deleting tables are also provided.

2.4.2 Using Tomcat

2.4.2.1 Dependencies

You must add required libraries into TOMCAT_HOME/lib folder:

- the JDBC connector for the metadata database with its dependencies (if any);
- the JDBC connector for the business data database with its dependencies (if any);
- the commonj library with its dependencies:
 - Apache Geronimo,
 - Concurrency JSR-166,
 - CommonJ.

Important: Enterprise Edition only

To enable the Import/Export capability, please also add the JDBC connector for [HyperSQLDB](#) , taking care of using version 1.8.0.2 .

2.4.2.2 File system resources

Create the folder TOMCAT_HOME/resources. Such a folder will contain some useful static resources and the indexes for the search engine used by Knowage.

2.4.2.3 Connection to metadata database

To define connection towards metadata database, edit the `TOMCAT_HOME/conf/server.xml` and add the information related to the metadata database inside the `GlobalNamingResources` tag. Specify: username, password, driver class name, JDBC URL and validation query (any valid query to be executed to validate connections). Connection's name must be `jdbc/knowage`:

```
1  <Resource auth="Container"
2      driverClassName="<JDBC driver>"
3      name="jdbc/knowage"
4      password="<password>"
5      type="javax.sql.DataSource"
6      url="<JDBC URL>"
7      username="<user name>"
8      validationQuery="<validation query>"
9      maxTotal="50"
10     maxIdle="50"
11     minIdle="10"
12     validationInterval="34000"
13     removeAbandoned="true"
14     removeAbandonedTimeout="3600"
15     logAbandoned="true"
16     testOnBorrow="true"
17     testWhileIdle="true"
18     timeBetweenEvictionRunsMillis="10000"
19     minEvictableIdleTimeMillis="60000" />
```

2.4.2.4 Cache database connection

In some scenarios (for example when defining a cockpit document on top of a file dataset), Knowage requires a database to be used as cache. It is highly recommended to create an empty database schema for this purpose. Then, you need to configure it inside `TOMCAT_HOME/conf/server.xml` as you did for metadata database. Feel free to type a name of your choice, in this example we used `jdbc/ds_cache`:

```
1  <Resource auth="Container"
2      driverClassName="<JDBC driver>"
3      name="jdbc/ds_cache"
4      password="<password>"
5      type="javax.sql.DataSource"
6      url="<JDBC URL>"
7      username="<user name>"
8      validationQuery="<validation query>"
9      maxTotal="50"
10     maxIdle="50"
11     minIdle="10"
12     validationInterval="34000"
13     removeAbandoned="true"
14     removeAbandonedTimeout="3600"
15     logAbandoned="true"
16     testOnBorrow="true"
17     testWhileIdle="true"
18     timeBetweenEvictionRunsMillis="10000"
19     minEvictableIdleTimeMillis="60000" />
```

2.4.2.5 Connection to business data

Edit the `TOMCAT_HOME/conf/server.xml` and add the information related to the database containing business data to be analysed by Knowage inside the `GlobalNamingResources` tag, specifying username, password, driver class name, URL and validation query. Feel free to type a name of your choice, in this example we used `jdbc/dwh`:

```

1 <Resource auth="Container"
2     driverClassName="com.mysql.jdbc.Driver"
3     name="jdbc/dwh"
4     password="password"
5     type="javax.sql.DataSource"
6     url="jdbc:mysql://localhost:3306/dwh"
7     username="root"
8     validationQuery="SELECT 1"
9     maxTotal="50"
10    maxIdle="50"
11    minIdle="10"
12    validationInterval="30000"
13    removeAbandoned="true"
14    removeAbandonedTimeout="3600"
15    logAbandoned="true"
16    testOnBorrow="true"
17    testWhileIdle="true"
18    timeBetweenEvictionRunsMillis="10000"
19    minEvictableIdleTimeMillis="60000"
20    factory="org.apache.tomcat.jdbc.pool.DataSourceFactory" />

```

2.4.2.6 Environment variables definition

Edit the file `TOMCAT_HOME/conf/server.xml` in Tomcat and add the following constants in the `GlobalNamingResources` tag, by setting the domain within the `host_url` value. That domain will be used by the browser to call Knowage server.

Such environment variables have the following meaning:

- `resource_path`: resources folder path,
- `sso_class`: SSO connector class name,
- `service_url`: backend services address, typically set to `http://localhost:8080/knowage`,
- `host_url`: frontend services address, the one the user types in his browser,
- `hmacKey`: secret key to generate JWT tokens used by the default security mechanism. You **must change** it, and **do not distribute** it. You can put any random alphanumeric string in it, and you can change it everytime you want, you just need to restart Tomcat to apply the change,
- `password_encryption_secret`: the complete path of a file to contain the password encryption secret. The file must contain random text of any length. This is a security configuration, so don't use short strings. For example, you can create a file and write text into it. **Do not distribute** it for any reason, create at least a backup copy of the file. **After the first start of Knowage, it will no longer be possible to change the secret key.** In case you lost this secret, look at the paragraph below to see how to update the passwords of existing users.

Important: Again we stress the point that the HMAC key must be a random string. Please DO NOT copy and paste it from this documentation, since this will compromise the security of the application.

Below you can see an example of configuration of the above variables in the `server.xml` file

```

1 <Environment name="resource_path" type="java.lang.String" value="${catalina.home}/resources"/>
2 <Environment name="sso_class" type="java.lang.String" value="it.eng.spagobi.services.common.JWTSSoService"/>
3 <Environment name="host_url" type="java.lang.String" value="http://localhost:8080"/>
4 <Environment name="service_url" type="java.lang.String" value="http://mydomain.com/knowage"/>
5 <Environment name="hmacKey" description="HMAC key" type="java.lang.String" value="... a random string ..."/>
6 <Environment name="password_encryption_secret" description="File for security encryption location"
7     type="java.lang.String" value="${catalina.home}/conf/knowage.secret"/>

```

2.4.2.7 Changing the secret key for password encryption

The password encryption secret key must be set during the installation and cannot be changed **anymore**, otherwise Knowage will no longer be able to authenticate already defined users. *In case the secret key is lost* you must create a new one, configure it into Knowage as described above and update passwords of existing users directly into Knowage metadata database (SBI_USER table). For this reason Knowage provides you a tool to get new encrypted values. This tool is a Java class that is shipped with the knowage-utils library; it accepts 2 input parameters:

- the complete path of the password encryption secret file;
- the password value in plaintext.

Below is an example of invoking the tool by command line using 'mypassword' as the plaintext password to be encrypted (of course TOMCAT_HOME must be replaced by the actual Tomcat base folder path).

```
1 java -cp "<TOMCAT_HOME>/webapps/knowage/WEB-INF/lib/knowage-utils-7.2.0.jar" it.eng.spagobi.security.utils.  
↪ PasswordEncryptionToolMain <TOMCAT_HOME>/conf/knowage.secret mypassword
```

This procedure must be repeated for all already existing users.

2.4.2.8 Recommended configuration

Edit TOMCAT_HOME/conf/setenv.sh (Linux) or TOMCAT_HOME/conf/setenv.bat (Windows) file in Tomcat by adding the following JVM arguments:

```
1 export JAVA_OPTS="$JAVA_OPTS -Dfile.encoding=UTF-8"  
2  
3 # We add -Duser.timezone=UTC to solve error when establishing connection to Oracle metadata database:  
4 # java.sql.SQLException: ORA-00604: error occurred at recursive SQL level 1  
5 # ORA-01882: timezone region not found  
6  
7 export JAVA_OPTS="$JAVA_OPTS -Duser.timezone=UTC"  
8  
9 export JAVA_OPTS="$JAVA_OPTS -Djava.awt.headless=true"  
10  
11 export JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -Djava.security.policy=$CATALINA_HOME/conf/catalina-  
↪ relaxed.policy"
```

2.4.2.9 Applications deploy

To deploy Knowage you have to copy all the WAR files inside the TOMCAT_HOME/webapps folder. Once the first start is ended each WAR file will be unzipped. It is also possible to unzip the WAR files manually using the unzip utility.

2.4.2.10 Thread pool definition

You must configure TOMCAT_HOME/conf/server.xml file and add the settings related to the pool of thread editing the GlobalNamingResources tag, as shown follow.

```
1 <Resource auth="Container" factory="de.myfoo.commonj.work.FooWorkManagerFactory" maxThreads="5" name="wm/  
↪ SpagoWorkManager" type="commonj.work.WorkManager"/>
```

2.4.2.11 Advanced memory settings

It is recommended to increase the memory dimension used by the application server. This can be done by adjusting some properties. The memory required by each application server depends on many factors: number of users, type of analyses, amount of handled data, etc. The minimum requirements are Xms1024m and Xmx2048m.

[LINUX] Insert at the beginning of the TOMCAT_HOME/bin/setenv.sh file this command:

```
export JAVA_OPTS="$JAVA_OPTS -Xms1024m -Xmx2048m -XX:MaxPermSize=512m"
```

[WIN] Insert at the beginning of the TOMCAT_HOME/bin/setenv.bat file this command:

```
set JAVA_OPTS= %JAVA_OPTS% -Xms1024m -Xmx2048m -XX:MaxPermSize=512m
```

2.4.2.12 Advanced Connector settings

Important: It is highly recommend to add URIEncoding="UTF-8" attribute to server.xml file connector tags in order to avoid special characters issues.

```
<Connector address="0.0.0.0" port="8009" protocol="AJP/1.3" maxPostSize="2097152000" redirectPort="8443"
URIEncoding="UTF-8" />
```

2.4.3 Datasource link within the applications

You would set up ResourceLink for JNDI datasource. To do so, you have to configure each knowage*/META-INF/context.xml and set the ResourceLink for each JNDI data source previously created. Inside the released packages two links are already defined:

- one for the jdbc/knowage resource, which the user must keep
- the other for the jdbc/foodmart, which should be renamed with jdbc/dwh.

```
<Context docBase="knowage-ee" path="/knowage" reloadable="true">
  <ResourceLink global="jdbc/dwh" name="jdbc/dwh" type="javax.sql.DataSource"/>
  <ResourceLink global="jdbc/knowage" name="jdbc/knowage" type="javax.sql.DataSource"/>
  <ResourceLink global="jdbc/ds_cache" name="jdbc/ds_cache" type="javax.sql.DataSource"/>
  <ResourceLink global="resource_path" name="resource_path" type="java.lang.String" />
  <ResourceLink global="sso_class" name="sso_class" type="java.lang.String" />
  <ResourceLink name="hmacKey" global="hmacKey" type="java.lang.String"/>
  <ResourceLink global="host_url" name="host_url" type="java.lang.String" />
  <ResourceLink global="service_url" name="service_url" type="java.lang.String"/>
  <ResourceLink global="wm/SpagoWorkManager" name="wm/SpagoWorkManager" type="commonj.work.WorkManager" />
  <ResourceLink global="password_encryption_secret" name="password_encryption_secret" type="java.lang.
String"/>
</Context>
```

Due to security reasons, configuration file containing some content to be used for encrypting must be created in Knowage environment. File must not be empty, but there's no minimum length. File can be put everywhere; path and name must be properly configured for each knowage*/META-INF/context.xml as shown below.

Important: Context update

The modification of these files will be effective as soon as the web application is reloaded or the application server is restarted.

2.4.4 Configuration of the metadata db dialect

Important: This step is not mandatory anymore

Knowage is now able to autonomously determine following configuration. That said, the following is not mandatory anymore however a user can force it to specific value.

Verify that the right dialect has been set inside `hibernate.cfg.xml` files. We list all the possible dialects that can be used:

```
1 <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>,  
2 <property name="hibernate.dialect">org.hibernate.dialect.PostgreSQLDialect</property>  
3 <property name="hibernate.dialect">org.hibernate.dialect.Oracle9Dialect</property>
```

You have to configure these following Hibernate configuration files and set the chosen dialect:

```
1 knowagekpiengine/WEB-INF/classes/hibernate.cfg.xml  
2 knowagegeoreportengine/WEB-INF/classes/hibernate.cfg.xml  
3 knowage/WEB-INF/classes/hsqldb/hibernate.cfg.xml  
4 knowage/WEB-INF/classes/hibernate.cfg.xml  
5 knowagesvgviewerengine/WEB-INF/classes/hibernate.cfg.xml  
6 knowagemeta/WEB-INF/classes/hibernate.cfg.xml  
7 knowagecockpitengine/WEB-INF/classes/hibernate.cfg.xml  
8 knowagedataminingengine/WEB-INF/classes/hibernate.cfg.xml
```

Important: Context update

The modification of these files will be effective as soon as the web application is reloaded or the application server is restarted.

2.4.5 Modification of the Quartz configuration

Important: This step is not mandatory anymore

Knowage is now able to autonomously determine following configuration. That said, the following is not mandatory anymore however a user can force it to specific value.

The scheduler is configured in `knowage/WEB-INF/classes/quartz.properties`. It is essential to enhance in this file the property `org.quartz.jobStore.driverDelegateClass` with the right value, according to the metadata database in use. Following the possible values:

```
1 # Hsqldb delegate class  
2 #org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.HSQLDBDelegate  
3 # Mysql delegate class org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.StdJDBCDelegate  
4 # Postgres delegate class  
5 #org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.PostgreSQLDelegate  
6 # Oracle delegate class  
7 #org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobstore.oracle.OracleDelegate
```

2.4.5.1 Clustering

When Knowage is installed in cluster with several nodes, it is necessary to activate the Cluster modality, adding these parameters to the `knowage/WEB-INF/classes/quartz.properties` file of every involved machines:

```
1 org.quartz.jobStore.isClustered = true  
2 org.quartz.jobStore.clusterCheckinInterval = 20000  
3 org.quartz.scheduler.instanceId = AUTO  
4 org.quartz.scheduler.instanceName = RHECMClusteredSchedule
```

2.4.6 Logging

It is necessary to set up a folder where Knowage and its analytical engines can store their respective log files. From now on, we will call LOG_DIR such folder and LOG_DIR_PATH the path that leads to it. This path is configured in file log4j.properties located inside the \WEB-INF\classes\ available in each web application. Shortly, to configure the Knowage log folder the user must execute the following steps:

- create the LOG_DIR folder on all cluster nodes on which it is intended to deploy Knowage Server and/or one of its analytical engines. The LOG_DIR_PATH string must be the same for every node;
- **[LINUX]** verify that Knowage has write permissions on this folder; set the property log4j.appender.knowage.File inside the WEB-INF/classes/log4j.properties to LOG_DIR_PATH/knowage.log;
- set the property log4j.appender.knowageXXXXXEngine.File inside the WEB-INF/classes/log4j.properties file of each engine to LOG_DIR_PATH/knowageXXXXXEngine.log;
- only for the Birt Engine, to set the property logDirectory inside the WEB-INF/classes/BirtLogConfig.properties file of the knowagebirtreportengine application to LOG_DIR_PATH.

2.4.7 Enable Java Security Manager

In Knowage, a user can create datasets, LOVs, etc.. with script languages like JavaScript. That introduces a security concern where a malicious user can execute code that can break the entire system. Java allows a system administrator to enable a [Security Manager](#) and to create a sandbox to limit privileges around the code that execute the script.

The Security Manager can be enabled with the following steps:

- Write a Security Policy for the Security Manager;
- Enable the Security Manager in the JVM.

The Security Policy is a text file read by a Security Manager that specifies all the privileges that a JVM can give to Java code: Tomcat has already a default policy in the file TOMCAT_HOME/conf/catalina.policy but is too much strict for Knowage code that needs to write multiple logs, make network connection and execute external applications. Knowage is already secured and can use a more relaxed policy like:

Listing 2.12: Complete path of the script

```
grant {
    permission java.security.AllPermission;
};
```

This policy can be saved to TOMCAT_HOME/conf/knowage-default.policy.

To enable the Security Manager a system administrator have to add some options to the Java JVM:

[LINUX] Insert at the end of the TOMCAT_HOME/bin/setenv.sh file this command:

```
export JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -Djava.security.policy=$CATALINA_HOME/conf/knowage-
default.policy"
```

[WIN] Insert at the end of the TOMCAT_HOME/bin/setenv.bat file this command:

```
set JAVA_OPTS= %JAVA_OPTS% -Djava.security.manager -Djava.security.policy=%CATALINA_HOME%\conf\knowage-default.
policy
```

2.4.8 Installation of Chromium Cockpit Export script

Important: Enterprise Edition only

Chromium Cockpit Export script is only available for Enterprise Edition.

Extract archive `knowage-cockpit-export-installation.zip` to `/opt/knowage`:

Listing 2.13: Complete path of the script

```
/opt/knowage/cockpit-export/cockpit-export.js
```

For alternatives path you have to fix `internal.nodejs.chromium.export.path` in Knowage Configuration Management.

2.5 How to upgrade to the latest version

This section describes the main steps to manually update an existing Knowage installation, on top of the certified Apache Tomcat server, to the latest available version.

Pay attention to the fact that Knowage versions' names adhere to the Semantic Versioning 2.0.0.

In the following we will refer to Tomcat installation folder as `TOMCAT_HOME`.

The upgrade of the following Knowage components is generally needed:

- the applications that reside within `TOMCAT_HOME/webapps` folder: all `knowage*.war` files (`knowage.war`, `knowagebirtreportengine.war`, `knowagecockpitengine.war`, ...)
- the Knowage metadata database, where information about analyses, datasets, etc ... is stored.

The latter component must be upgraded in case you are moving from a different major or minor version (for example: from 6.4.x to 7.2.y, or from 7.1.x to 7.2.y), but there is no need in case you are upgrading to a new patch release of the same major/minor family (for example: from 7.2.0 to 7.2.6, or from 7.2.6 to 7.2.13).

2.5.1 Preliminary operations

Before starting upgrade procedure, you have to:

- download latest packages from the OW2 repository: base location is <https://release.ow2.org/knowage/>, you'll find a folder for each version, each folder contains: Applications with the relevant war files, and Database scripts with the SQL scripts (distributed as zip files) to upgrade Knowage metadata database for supported RDBMS;
- make a backup copy of the old web applications that reside within: `TOMCAT_HOME/webapps`;
- make a backup copy of Knowage metadata database.

2.5.2 Upgrade operations

To upgrade Knowage installation follow these steps:

- stop Apache Tomcat service;
- upgrade the Knowage metadata database by executing the SQL scripts. Each SQL script is conceived for upgrading a Knowage <major.minor> version into the next one, therefore you need to execute all scripts between your current <major.minor> version to the latest one. As an example, suppose RDBMS is Oracle, your current

Knowage version is 6.0.x and you want to upgrade into Knowage 6.3.x, then you need to execute the following scripts:

```
1 ORA_upgradescript_6.0_to_6.1.sql
2 ORA_upgradescript_6.1_to_6.2.sql
3 ORA_upgradescript_6.2_to_6.3.sql
```

Note:

Moving into a newest patch version within the same <major.minor> family In case you are moving into a newest patch version that belongs to the same <major.minor> family (for example from Knowage 7.2.0 to 7.2.6, or from 7.2.6 to 7.2.13) there is no need to execute any SQL script.

- move all knowage*.war files and all the knowage* directories from TOMCAT_HOME/webapps into a backup directory;
- delete the following directories: TOMCAT_HOME/temp and TOMCAT_HOME/work;
- copy and paste all the new knowage*.war packages within TOMCAT_HOME/webapps directory;
- in case you are upgrading from a version prior to 7.2.0, create a file containing the password encryption secret (just a random text of any length). This is a security configuration, so don't use short strings. **Do not distribute** it for any reason. Then update TOMCAT_HOME/conf/server.xml file by creating a new environment variable as shown below and set the value property as the complete path of the file with password encryption secret:

```
<Environment name="password_encryption_secret" description="File for security encryption location"
type="java.lang.String" value="${catalina.home}/conf/knowage.secret"/>
```

At this point, in some cases you need to deal with some configuration files, in particular when you modified the following files within the previous Knowage installation, then you need to restore those changes (after having unzipped the war files):

- context files TOMCAT_HOME/webapps/knowage*/META-INF/context.xml: they contain links to resources such as datasource connections and environment variables; in case you modified them in order to add a new data-source, you need to restore the changes and check if links to environment variables defined in TOMCAT_HOME/conf/server.xml are all there. In case you defined contexts with relevant files inside TOMCAT_HOME/conf/Catalina/localhost and you are upgrading from a version prior to 7.2.0, then you need to add the link to the password_encryption_secret variable, since that was introduced by 7.2.0 version;
- Hibernate files: they contain the metadata database Hibernate dialect (the hibernate.dialect property): since version 7.2.0, Knowage is able to autodetect the dialect by itself but, in case you customized it to a value other than org.hibernate.dialect.MySQLDialect or org.hibernate.dialect.PostgreSQLDialect or org.hibernate.dialect.Oracle9Dialect, you have to restore your change: this is the list of Hibernate files to be checked:

```
TOMCAT_HOME/webapps/knowage/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagecockpitengine/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagedataminingengine/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagegeoreportengine/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagekpiengine/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagemeta/WEB-INF/classes/hibernate.cfg.xml
TOMCAT_HOME/webapps/knowagesvgviewerengine/WEB-INF/classes/hibernate.cfg.xml
```

- Quartz configuration file for metadata database dialect and for cluster configuration (in case of any cluster): again, since version 7.2.0, Knowage is able to autodetect the dialect by itself but, in case you customized the org.quartz.jobStore.driverDelegateClass property inside TOMCAT_HOME/webapps/knowage/WEB-INF/classes/quartz.properties to a value other than org.quartz.impl.jdbcjobstore.StdJDBCDelegate or org.quartz.impl.jdbcjobstore.PostgreSQLDelegate or org.quartz.impl.jdbcjobstore.oracle.OracleDelegate, you have to restore your change. Regarding clus-

ter configuration, by default it is not enabled on released packages therefore you need to restore it in case you have a clustered installation: add these lines in TOMCAT_HOME/webapps/knowage/WEB-INF/classes/quartz.properties (or restore them from the backup copy):

```
org.quartz.jobStore.isClustered = true
org.quartz.jobStore.clusterCheckinInterval = 20000
org.quartz.scheduler.instanceId = AUTO
org.quartz.scheduler.instanceName = RHECMClusteredSchedule
```

Important: Since Knowage 7.2.0, the security level was highly increased. For this reason, users are requested to log in and change their password as a first step after upgrading.

To admin users: it is recommended to check which users didn't change the password and tell them to do it as soon as possible. Run the following query on the Knowage metadata database to extract the list of users who are still using the previous password encryption mechanism:

```
select * from SBI_USER where password like '#SHA#%' order by user_id;
```

2.6 Knowage CE Installer

Knowage CE installer is an application which steers the user to the installation and the first configuration of the product.

Knowage CE is a web application, meaning it runs centrally on a server, and users interact with it through web browsers from any computer on the same network. Knowage CE Installer lets you easily configure your server.

2.6.1 Server-side requirements

2.6.1.1 Operating system

Knowage CE Installer runs on **Windows**, **Linux** and **macOS** operating systems.

2.6.1.2 Java platform

Knowage CE Installer requires:

- JDK 1.8
- JAVA_HOME environment variable

2.6.1.3 Memory

Knowage CE requires **3GB** of available RAM. This configuration is enough for most evaluation purposes.

2.6.1.4 Disk usage

Knowage CE requires **2GB** of free space on file system. Optional embedded MariaDB Server 10.2 requires **4GB** of free space on file system.

2.6.1.5 Database

Knowage CE Installer requires one of the following **external databases**:

- MySQL Server 5.7 already installed
- MariaDB Server 10.2 already installed

A user with sufficient **permissions to create schemas** must be provided. Knowage CE Installer connects to database using a JDBC driver via TCP/IP connection.

If you are using MySQL Server 5.7 we suggest to set following configuration in file `my.ini`:

- `innodb_buffer_pool_size` = 2G (adjust value here, 50%-70% of total RAM)
- `innodb_log_file_size` = 500M

Knowage CE Installer includes also the option to use one of the following **embedded databases**:

- MariaDB Server 10.2 for Windows 64 bit

Please note that embedded database option is not available for macOS.

2.6.1.6 Application server

Knowage CE Installer provides Apache Tomcat 7 out of the box. Don't worry about pre-installing any application server.

2.6.1.7 Proxy settings

If proxy is enabled please add property `http.nonProxyHosts` to JVM properties after completing installation, modifying file `<installation_directory>\Knowage-Server-CE\bin\setenv.bat` on Windows or `<installation_directory>/Knowage-Server-CE/bin/setenv.sh` on Linux/macOS.

```
1 -Dhttp.nonProxyHosts=localhost
```

2.6.2 Client-side requirements

2.6.2.1 Browser

Enable your browser to execute JavaScript.

2.6.2.2 Proxy settings

If proxy is enabled please add hostname to proxy's ignore list.

2.6.3 Launching

2.6.3.1 Windows

Important: The installer has to be run as administrator.

2.6.3.2 Linux/macOS

Extract the installer SH file typing the command in shell:

```
1 unzip Knowage-7_0_0-CE-Installer-Unix-20191022.zip
```

Warning: On macOS the default app used to open ZIP files may fail to extract the installer ZIP file.

Enable the execute permission on the file, typing the command in shell:

```
1 chmod +x Knowage-7_0_0-CE-Installer-Unix-20191022.sh
```

Knowage CE installer can run in GUI or console mode.

- **GUI mode** is available only if a desktop environment is available. Run installer in GUI mode typing the command in shell:

```
1 ./Knowage-7_0_0-CE-Installer-Unix-20191022.sh
```

- **Console mode** is always available and let complete installation using shell. Run installer in Console mode typing the command in shell:

```
1 ./Knowage-7_0_0-CE-Installer-Unix-20191022.sh -c
```

2.6.4 Managing Knowage CE

After completing installation, you can start/stop Knowage CE using desktop links, start menu entries or following shell commands.

2.6.4.1 Windows

- Start Knowage CE using <installation directory>\Knowage-Server-CE\bin\startup.bat
- Stop Knowage CE using <installation directory>\Knowage-Server-CE\bin\shutdown.bat

2.6.4.2 Windows (embedded MariaDB option)

- Start Knowage CE using <installation directory>\Knowage-Server-CE\bin\knowage_startup.bat
- Stop Knowage CE using <installation directory>\Knowage-Server-CE\bin\knowage_shutdown.bat

2.6.4.3 Linux/macOS

- Start Knowage CE using <installation directory>/Knowage-Server-CE/bin/startup.sh
- Stop Knowage CE using <installation directory>/Knowage-Server-CE/bin/shutdown.sh

2.7 R installation

It is required the installation of the following components for the correct operation of the data mining engine:

- R
- R Studio
- rJava

[LINUX] The first two components, needed for the functionality of the Knowage data mining engine, has to be installed through the `rpm` comand, and, the third, through the RStudio GUI. Once retrieved the RPM file, open the folder and launch the comands:

Listing 2.14: Commands for the rpm file

```

1 rpm -Uvh ./R-3.2.2-1.el6.x86_64.rpm
2 yum install --nogpgcheck rstudio-server-rhel-0.99.486-x86_64.rpm
3 cp ./RStudio/rJava_0.9-8.tar.gz $TOMCAT_HOME
4 chown tomcat $TOMCAT_HOME/rJava_0.98.tar.gz
5 chown -R tomcat.root /usr/lib64/R/library && chmod -R 775 /usr/lib64/R/library
6 chown -R tomcat.root /usr/share/doc/R-3.2.2 && chmod -R 775 /usr/share/doc/R

```

Typing the address `http://server_ipormachine_name:8787/` in the browser, the user gets on screen the page showed below:

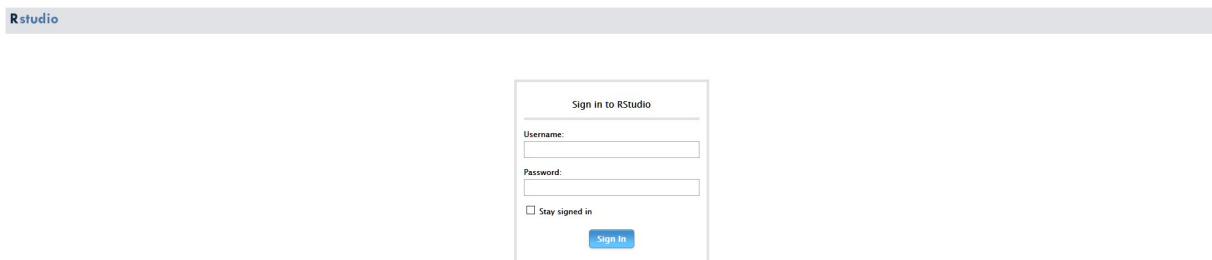


Fig. 2.2: Sign in RStudio.

Then log in with the user credentials used for the Tomcat 7 installation: `tomcat / <tomcat_user_password>`.

The following images show the sequence of steps the user will encounter:

Meanwhile the package is installed, remember to answer NO when asked to create a personal library in the user home (that can be found under `$HOME/RStudio/log`). This way, rJava will be installed in the directory `/usr/lib64/R/library/rJava`.

Finally, edit the `TOMCAT_HOME/bin/setenv.sh` adding the following commands:

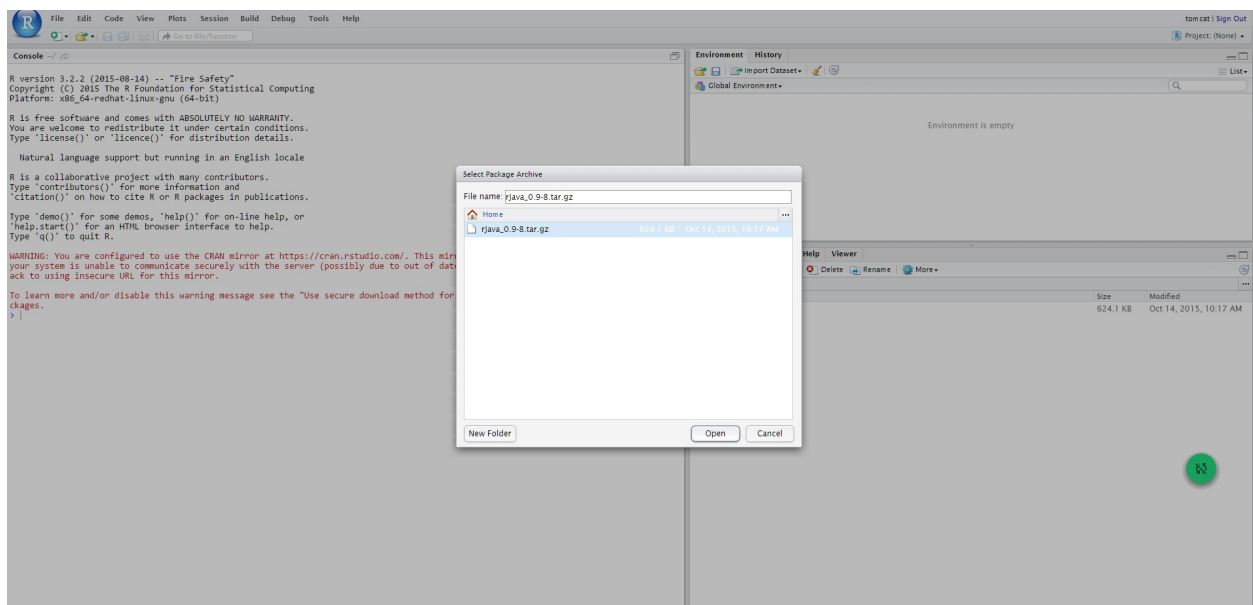
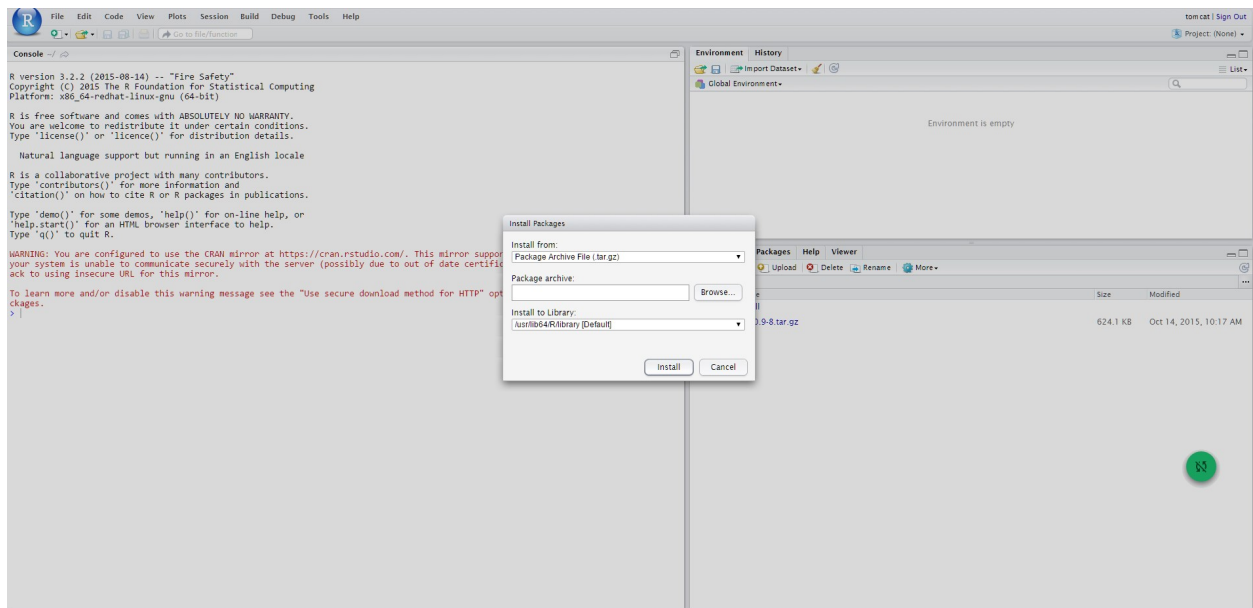
```

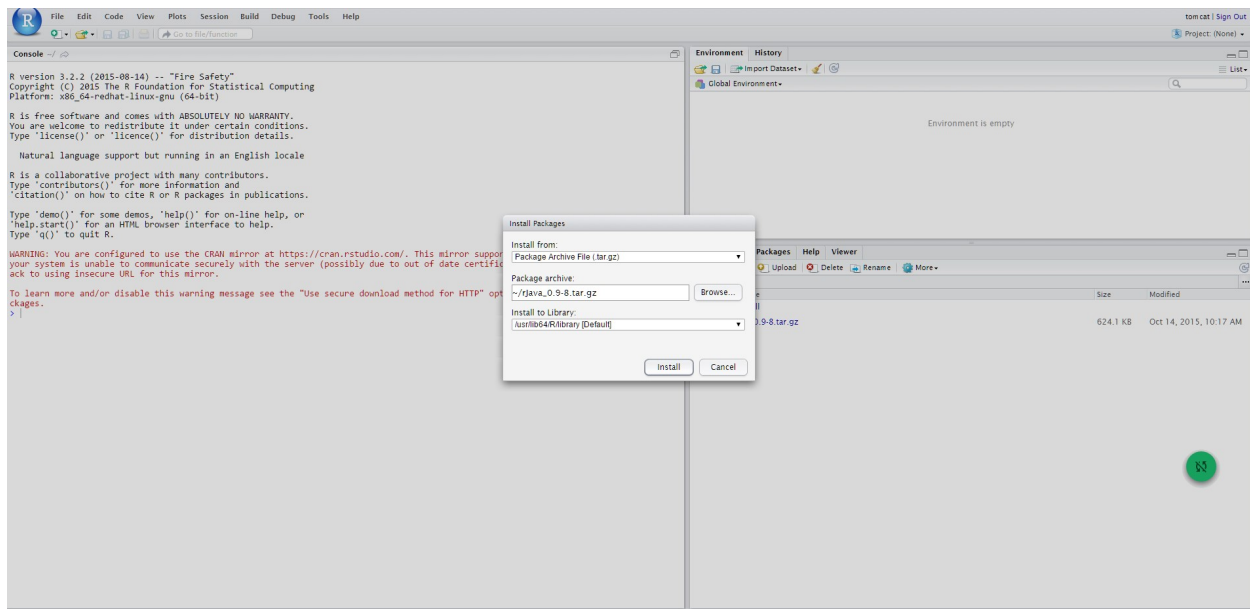
1 export R_HOME=/usr/lib64/R
2 export LD_LIBRARY_PATH=/usr/lib64/R/library/rJava/jri

```

2.8 Python Widget and DataSet

These functionalities use a Python standalone webservice, which allows to submit widgets and datasets scripts and get result from a Python environment already installed on the machine where the webservice runs. For this reason, Python environments need to be installed and configured on the same machine of Knowage server or even on a remote one,





and the Python webservice has to be running **inside that same environment**. This implies that, in order to use this functionalities, you have to install Python properly (depending on the OS) on the same machine where the service will be running. You can find all information about Python installation at <https://www.python.org>. The official supported version is Python 3.7, but other 3.x releases might work as well.

2.8.1 Install knowage-python webservice

The knowage-python package contains the source code of the webservice that has to be installed and run on the server. You can download it via pip with the command:

```
pip install knowage-python
```

or simply you can find it in the Knowage-Server github repository under the Knowage-Python folder.

You will now have to create the configuration for the webservice. This configuration will be contained inside a file called `config.xml` and placed inside the `Knowage-Python/pythonwebservice/app` folder.

The structure of the file will be as follows:

Listing 2.15: Example of a knowage-python webservice configuration file

```

1 <data>
2   <environment name="knowage-python">
3     <hmackey>foobar123</hmackey>
4     <knowageaddress>knowage.server.address.com</knowageaddress>
5     <pythonaddress>python.webservice.address.com</pythonaddress>
6     <bokehportsrange>57000-58000</bokehportsrange>
7   </environment>
8 </data>
```

- `hmackey` : the Knowage HMAC key contained in the `server.xml` file,
- `knowageaddress` : the address at which the python webservice can contact the Knowage server (the two entities can either be on the same machine or on different machines),

- `pythonaddress` : the address of the current instance of Python (it is needed by the Knowage server to contact back the engine if needed),
- `bokehportsrage` : the range of open ports upon which the service can dinamically instantiate Bokeh Servers.

2.8.2 Run knowage-python webservice

Once you have installed all the requirements, you need to get the python-webservice running. In order to do so, you can rely on a WSGI Server. If you are working on a UNIX environment, take a look at gunicorn (<https://gunicorn.org/>). The service leverages on Flask, for deployment in any other environment take a look at the official documentation (<https://flask.palletsprojects.com/en/1.1.x/deploying/#deployment>). **The entry point for the application is “Knowage-Python/pythonwebservice/knowage-python.py” and the default port is “5000”.**

Important: Webservice permissions

The knowage-python webservice must have the rights to read/write in its own folders (pythonwebservice/*).

2.9 R Widget and DataSet

As for Python, also the R functionalities leverage on a standalone webservice, this time written in R. Take a look at the official R Project documentation and find out how to get R (<https://www.r-project.org/>).

2.9.1 Install knowage-r webservice

Inside the Knowage-Server github repository, under the Knowage-R folder you can find the sources of the knowage-r webservice.

Once you have downloaded the source code, you will have to create the configuration for the webservice. This configuration will be contained inside a file called `configs.R` and placed inside the Knowage-R folder.

The configuration is indeed really simple since you only need to specify the Knowage HMAC key contained in the `server.xml` file.

In the `constants.R` file you can set the default webservice port and a whitelist of IP addresses that can contact the webservice.

2.9.2 Run knowage-r webservice

Once you have installed all the requirements, you need to get the r-webservice running. In order to do so, it's enough to run the main file “knowage-r.R” with the basic R interpreter, via the `RScript` command or an equivalent one.

Important: Webservice permissions

The knowage-r webservice must have the rights to read/write in its own folder.

2.10 Configure Knowage to enable Python/R functionalities

From the Knowage interface you can now enable the Python/R functionalities.

Go to the **Roles management** section, in the *Authorizations* tab under *Widgets* check the **Edit Python Scripts** option. Now you will be able to see the Python and R Dataset and Widget among the list of available ones.

Go to the **Configuration management** section, and create new variables of category `PYTHON_CONFIGURATION` and `R_CONFIGURATION`. The value of this variables will specify the addresses of the Python and R webservices (es. `python.webservice.address.com/domain`). Now you will be able to see the addresses of the so configured environments when creating a Dataset or a Widget.

Be aware that depending on the architecture of your solution, you might have to define two different addresses for reaching the same instance of Python.

- One address is for reaching Python from the client (browser) and will be used when creating a widget,
- One address is for reaching Python from the server (Knowage) and will be used when creating a Dataset.

2.11 DataMining Engine

The engine uses a Java/Python interface, which allows to submit scripts and get result from a Python environment already installed on the machine where the Datamining Engine runs. For this reason, Python environment need to be installed on the same machine of KnowAge server. This implies that, in order to run this engine, you have to install Python properly (depending on the OS) on the same machine where Knowage is installed. You can find all information about Python installation at <https://www.python.org>. Datamining engine only support Python 3 (the product has been tested with Python 3.4.0, but other 3.x releases are supported).

2.11.1 JPY installation

JPY is a connector that make possible a bidirectional communication between Python and Java and its components must be installed on both sides (dataminingengine Java project and Python environment). Dataminingengine project is provided with `jpy.jar` that allows the communication, but this is not sufficient, because JPY must be installed on your Python environment. To do this you have to download the JPY source files and build them by yourself on your machine (unfortunately pre-built packages are not made available yet by JPY creators). All the detailed instructions to build and install JPY on your Python environment are described on the page <http://jpy.readthedocs.org/en/stable/install.html>. During the testing phase Python 3.4 and JPY 0.8 (stable version) have been used; here the version-specific installation steps are described. You will need:

- Python 3.3 or higher (3.2 may work as well but is not tested),
- Oracle JDK 7 or higher (JDK 6 may work as well),
- Maven 3 or higher,
- Microsoft Windows SDK 7.1 or higher If you build for a 32-bit Python, make sure to also install a 32-bit JDK. Accordingly, for a 64-bit Python, you will need a 64-bit JDK.

The Python setup tools `distutils` can make use of the command-line C/C++ compilers of the free Microsoft Windows SDK. These will be used by `distutils` if the `DISTUTILS_USE_SDK` environment variable is set. The compilers are made accessible via the command-line by using the `setenv` tool of the Windows SDK. In order to install the Windows SDK execute the following steps.

- If you already use Microsoft Visual C++ 2010, make sure to uninstall the x86 and amd64 compiler redistributables first. Otherwise the installation of the Windows SDK will definitely fail. This may also be applied to higher versions of Visual C++.

- Download and install Windows SDK 7.1.
- **Download and install Windows SDK 7.1 SP1. Open the command-line and execute:**
 - "C:\\Program Files\\Microsoft SDKs\\Windows\\v7.1\\bin\\setenv" /x64 /release to prepare a build of the 64-bit version of jpy.
 - "C:\\Program Files\\Microsoft SDKs\\Windows\\v7.1\\bin\\setenv" /x86 /release to prepare a build of the 32-bit version of jpy.

Now set other environment variables:

```
1 SET DISTUTILS_USE_SDK=1
2   SET JAVA_HOME=%JDK_HOME%
3   SET PATH=%JDK_HOME%\jre\bin\server;%PATH%
```

Then, to actually build and test the jpy Python module use the following command: `python setup.py install`. To use JPY you need to replace the `jpyconfig.properties` file on your project, with the one generated by the build process that is present in your JPY built folder `jpy-master\build\lib.<SO-CPU-PYTHON_versions>`. Properties file to replace is located under `knowagedataminingengine\src\`.

Datamining engine supports the use of all Python libraries: before import a library in your script install it on your native Python environment (for example using `pip`). To use Python YOU NEED TO INSTALL the following libraries: `matplotlib`, `pandas`, `numpy`, `scipy`. You can install them using `pip` typing the following commands on your native Python console:

```
1 pip install pandas
2 pip install numpy
3 pip install scipy
4 pip install matplotlib.
```

Listing 2.16: Example of a Knowage Data Mining engine template which uses a Python script

```
1 <?xml version="1.0" encoding="ISO-8859-15"?>
2 <DATA_MINING>
3   <LANGUAGE name="Python"/>
4   <DATASETS>
5     <DATASET name="df" readType="csv" type="file" label="HairEyeColor" canUpload="true"><![CDATA[sep=',']]>
6     </DATASET>
7   </DATASETS>
8   <SCRIPTS>
9     <SCRIPT name="test01" mode="auto" datasets="df" label="HairEyeColor" libraries="csv,os,pandas,numpy">
10       <![CDATA[ print(df.ix[0,0]) y=df.ix[0,0] ]]>
11     </SCRIPT>
12   </SCRIPTS>
13   <COMMANDS>
14     <COMMAND name="testcommand" scriptName="test01" label="test01" mode=" auto">
15       <OUTPUTS>
16         <OUTPUT type="text" name="first_element" value="y" function="" mode="manual" label="first_
17         ↪element"/>
18       </OUTPUTS>
19     </COMMAND>
20   </COMMANDS>
21 </DATA_MINING>
```

Note that the `LANGUAGE` tag is used to specify the language to use: `name=Python` and `name=R` are supported. If the `LANGUAGE` tag is not present or name is not specified correctly, the default language is set to `R`.

2.12 CAS installation

CAS is an application that implements a Single-Sign-On (SSO) mechanism. It is a good practise in production environments to install it and configure it so to have secure access to the Knowage server applications. CAS expects the use of the HTTPS protocol.

2.12.1 Deploy of the CAS application

Carry out the following steps:

- shut down the server if running,
- deploy CAS application,
- for Tomcat: unzip the `cas.war` file inside the `TOMCAT_HOME/webapps/cas`
- for JBoss: copy the `cas.war` file inside the `JBOSS_HOME/standalone/deployments`
- edit `/cas/WEB-INF/classes/cas_spagobi.properties` inserting the connection parameters for the meta-data database of Knowage, as following

Listing 2.17: Connection parameters for Knowage metadata db.

```

1  spagobi.datasource.driver=<driver JDBC>
2  spagobi.datasource.url=<URL JDBC>
3  spagobi.datasource.user=<user_name>
4  spagobi.datasource.pwd=<password> encrypt.password=true

```

For further details please refer to the official documents available on CAS website <https://www.apereo.org/projects/cas>.

2.12.2 HTTPS certificate

Since CAS application requires the use of the HTTPS protocol, the user must own an SSL certificate: it can be released a Certification Authority (CA) or it can be self-signed (it is recommended the use of the `keytool` utility -<http://docs.oracle.com/javase/7/docs/technotes/tools/solaris/keytool.html>- available in the JDK).

In case the certificate self-signed, it must be inserted in the CA repository of the JDK (usually such a repository is located in the `JDK_HOME/jre/lib/security`) or in an ad-hoc repository, called `truststore`, and conveniently configured in the application server in use. It is sufficient to set the two Java properties `Djavax.net.ssl.trustStore=<truststore path>` and `Djavax.net.ssl.trustStorePassword=<truststore password>`

We suggest to refer to the Java documents for more details. In the following we will restrict on give some useful commands of the `keytool` utility if the user intends to install a self-signed certificate:

- generate a copy of the public/private key-pair into a repository (keystore) called `keystore.jks`, as below:

Listing 2.18: keystore.jks creation.

```

1  $JAVA_HOME/bin/keytool -genkeypair -keystore keystore.jks -storepass <keystore_
    password> -alias <certificate alias> -keyalg RSA -keysize 2048 -validity 5000 -dname CN=
    <server name that hosts Knowage >, OU=<organization unit>, O=<organization name>, L=<locality_
    name>, ST=<state name>, C=<country>

```

- export a certificate in a `cert.crt` file, as suggested below:
- set the certificate inside the CA certificates of the JDK to make it accessible (the user will be asked the CA certificates password, the default one is *changeit*)

Listing 2.19: Importing the certificate into JDK CA repository.

```

1      $JAVA_HOME/bin/keytool -import -trustcacerts -alias <alias del certificado> -file cert.
    ↪ crt -keystore
2      $JAVA_HOME/jre/lib/security/cacerts

```

2.12.3 Configuration of the HTTPS protocol for Tomcat

To enable the HTTPS protocol it is necessary to operate according to these steps:

- copy the keystore which contains the pair public/private keys (keystore.jks) inside the TOMCAT_HOME/conf;
- edit the TOMCAT_HOME/conf/server.xml file, comment the HTTP connector on 8080 port and uncomment the HTTPS connector on 8443 port and configure it as below:

Listing 2.20: Export of the certificate.

```

1      <Connector acceptCount="100"
2              maxHttpHeaderSize="8192"
3              clientAuth="false"
4              debug="0"
5              disableUploadTimeout="true"
6              enableLookups="false"
7              SSLEnabled="true"
8              keystoreFile="conf/keystore.jks"
9              keystorePass="<keystore password>"
10             maxSpareThreads="75"
11             maxThreads="150"
12             minSpareThreads="25"
13             port="8443"
14             scheme="https"
15             secure="true"
16             sslProtocol="TLS"
17     />

```

2.12.4 Knowage configuration

Once the CAS has been installed, it is necessary to modify the Knowage configuration. The user must edit some values of the SBI_CONFIG table using the administrator interface

Listing 2.21: Values of the SBI_CONFIG table to change.

```

1      SPAGOBI_SSO.ACTIVE:
2      set valueCheck to true
3
4      CAS_SSO.VALIDATE-USER.URL:
5      set valueCheck to https://<URL of the CAS application>/cas
6
7      CAS_SSO.VALIDATE-USER.SERVICE:
8      set valueCheck to https://<URL of the Knowage server >:8443/knowage/proxyCallback
9
10     SPAGOBI_SSO.SECURITY_LOGOUT_URL:
11     set valueCheck to https://<URL of the CAS application>/cas/logout

```

Then set the sso_class environment variable as below:

```

1      <Environment name="sso_class" type="java.lang.String" value="it.eng.spagobi.services.cas.
    ↪ CasSsoService3NoProxy"/>

```

This variable is located:

- Tomcat: in the TOMCAT_HOME/conf/server.xml
- JBoss: in the JBOSS_HOME/standalone/configuration/standalone.xml

Edit all knowage\WEB-INF\web.xml to activate CAS filters.

Listing 2.22: Setting the CAS filters for sso_class variable.

```

1 <filter>
2   <filter-name>CAS Authentication Filter</filter-name>
3   <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
4   <init-param>
5     <param-name>casServerLoginUrl</param-name>
6     <param-value>https://<nome del server CAS>/cas/login</param-value>
7   </init-param>
8   <init-param>
9     <param-name>serverName</param-name>
10    <param-value><dominio di knowage, incluso il protocollo e la porta, se non standard></param-value>
11  </init-param>
12 </filter>
13
14 <filter>
15   <filter-name>CAS Validation Filter</filter-name>
16   <filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilter</filter-class>
17   <init-param>
18     <param-name>casServerUrlPrefix</param-name>
19     <param-value>https://<nome del server CAS>/cas/</param-value>
20   </init-param>
21   <init-param>
22     <param-name>serverName</param-name>
23     <param-value><dominio di Knowage Server, incluso il protocollo e la porta, se non standard></param-
24     ↪value>
25   </init-param>
26   <init-param>
27     <param-name>proxyReceptorUrl</param-name>
28     <param-value>/proxyCallback</param-value>
29   </init-param>
30
31 [Nelle web application knowageXXXEngine presente anche questo parametro:
32
33 <init-param> <param-name>proxyCallbackUrl</param-name>
34 <param-value>
35   <dominio di knowage Server, incluso il protocollo e la porta, se non standard>/< knowageXXXEngine>/
36   ↪proxyCallback </param-value>
37 </init-param>]
38
39 </filter>
40
41 <filter>
42   <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
43   <filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
44 </filter>...
45
46 <filter-mapping>
47   <filter-name>CAS Authentication Filter</filter-name>
48   <url-pattern>/servlet/*</url-pattern>
49 </filter-mapping>
50
51 <filter-mapping>
52   <filter-name>CAS Validation Filter</filter-name>
53   <url-pattern>/servlet/*</url-pattern>
54 </filter-mapping>
55 <filter-mapping>
56   <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
57   <url-pattern>/servlet/*</url-pattern>

```

(continues on next page)

(continued from previous page)

```

58 </filter-mapping>
59
60 [Nelle web application knowageXXEngine presente anche questo mapping:
61 <filter-mapping>
62 <filter-name>CAS Validation Filter</filter-name>
63 <url-pattern>/proxyCallback</url-pattern>
64 </filter-mapping>]
```

All `web.xml` files have CAS filters already configured, but they are commented. The user must uncomment them, looking for the strings `START-CAS`, `END-CAS` and adjust the URL as the code above reports.

2.13 Advanced configuration

In this chapter we will describe all the advanced configuration parameters of Knowage.

2.13.1 Thread manager

For Tomcat: the configuration of the pool of thread is available inside the `TOMCAT_HOME/conf/server.xml`. Refer to Code below.

Listing 2.23: Configuration of the pool of thread for Tomcat.

```

1 <Resource auth="Container" factory="de.myfoo.commonj.work.FooWorkManagerFactory"
2   maxThreads="5"
3   minThreads="1"
4   queueLength="10"
5   maxDaemons="10"
6   name="wm/SpagoWorkManager"
7   type="commonj.work.WorkManager"/>
```

For JBoss: the configuration of the pool of thread is available inside the `JBOSS_HOME/standalone/configuration/s`. Refer to Code below.

Listing 2.24: Configuration of the pool of thread for JBoss.

```

1 <object-factory name="java:global/SpagoWorkManager" module="de.myfoo.commonj"
2   class="de.myfoo.commonj.work.MyFooWorkManagerFactory">
3   <environment>
4     <property name="maxThreads" value="5"/>
5     <property name="minThreads" value="1"/>
6     <property name="queueLength" value="10"/>
7     <property name="maxDaemons" value="10"/>
8   </environment>
9 </object-factory>
```

In both cases, the meaning of the configuration parameters is the following:

- `minThreads`: the minimum number of threads in the thread pool. Default: 2;
- `maxThreads`: the maximum number of threads in the thread pool. Default: 10;
- `queueLength`: the number of work items that can be queued - 0 means no queuing. Default: 10;
- `maxDaemons`: the maximum number of daemon threads to allow for this work manager. Default: 10.

2.13.2 Cache parameters

First of all, the user must configure the distributed cache. This helps to coordinate the parallel access to the distributed cache, guaranteeing a thread-safe access. It is necessary to configure the `hazelcast.xml` file (available in the `knowage/WEB-INF/classes/`) typing in the `"member"` tag the IP address or hostname of the machine on which Knowage is installed (for example `<member> 192.168.29.43</member>`). In case of multi-node configuration, it is obviously important to report all cluster members. This operation must be carried out on every node. Furthermore, it is possible to implement a finer tuning of the cache behaviour, changing the Knowage configuration. The user must edit some values of the `SBI_CONFIG` table using the specific administrator interface.

- **SPAGOBI.CACHE.NAMEPREFIX:** It configures the prefix of temporary table in the cache (Default : `"sbi-cache"`)
- **SPAGOBI.CACHE.SPACE_AVAILABLE:** It resizes cache dimension (bytes) (Default : 1024)
- **SPAGOBI.CACHE.LIMIT_FOR_CLEAN:** It configures the maximum cache section (in percentage) that can be cleaned at runtime when the cache has not enough space to store a dataset. (Default : 50)
- **SPAGOBI.CACHE.SCHEDULING_FULL_CLEAN:** It schedules the recurring operation of complete cleaning of the cache. This periodic cleaning delete all dataset in the cache, without considering further parameters. At the end of the cleaning, the cache is empty. The allowable values are: `EVERY_10_MINS`, `EVERY_15_MINS`, `EVERY_20_MINS`, `EVERY_30_MINS`, `HOURLY`, `DAILY`, `WEEKLY`, `MONTHLY`, `YEARLY`. Any value other than those listed above does not enable periodic cleaning. (Default: DAILY)
- **SPAGOBI.CACHE.DS_LAST_ACCESS_TTL:** It configures the Time To Live of a dataset inside the cache. This parameter defines the minimum TTL (in seconds) so to guarantee that a dataset remains in cache. A too-high value can lead the cache to breakdown (in this case, there is no way to insert new datasets), while a too-low value can lead to situations when there are no certainties of the stability of the dataset in the cache. (Default 600)
- **SPAGOBI.CACHE.DATABASE_SCHEMA:** Name of the schema on which the tables are created. Such schema is defined by the datasource when it is set as Write-Default. Generally it is not necessary to configure this parameter since it is calculated at runtime. (default `<empty>`)
- **SPAGOBI.CACHE.LIMIT_FOR_STORE:** It configures the ratio (in percentage) between the dimension of the cache and the maximum dimension of a dataset in the cache. If the dimension of the dataset which the user intends to persist is bigger than the configured percentage, the system blocks the that persistence attempt. (Default : 10)
- **SPAGOBI.CACHE.CREATE_AND_PERSIST_TABLE.TIMEOUT:** It represents the maximum time (in seconds) to create temporary table for the dataset. (Default : 120)
- **SPAGOBI.WORKMANAGER.SQLDBCACHE.TIMEOUT:** It represents the maximum waiting time (in milliseconds) of an asynchronous work. (Default: 180000)
- **SPAGOBI.CACHE.HAZELCAST.TIMEOUT :** It represents the maximum time (in seconds) to get a distributed lock. (Default 120)
- **SPAGOBI.CACHE.HAZELCAST.LEASETIME:** It represents the maximum time (in seconds) for releasing a distributed lock already got. (Default :240)

2.13.3 Logging

Knowage uses the component Log4J to create the log applications. Each web application has its own file inside the folder `/knowageXXXX/WEB-INF/classes/log4j.properties`. The content of this file change accordingly to the settings: the **appenders** allows to modify the level of the log. As an example, in the following code block, we analyze the log file of Knowage. In the first part we can set the generation mechanism of the log file, while in the second one the level of tracing.

Listing 2.25: Logg appender.

```

1 log4j.rootLogger=ERROR, SpagoBI
2
3 # SpagoBI Appender
4 log4j.appender.SpagoBI=org.apache.log4j.RollingFileAppender
5 log4j.appender.SpagoBI.File=${catalina.base}/logs/knowage.log
6 log4j.appender.SpagoBI.MaxFileSize=10000KB
7 log4j.appender.SpagoBI.MaxBackupIndex=0
8 log4j.appender.SpagoBI.layout=org.apache.log4j.PatternLayout
9 log4j.appender.SpagoBI.layout.ConversionPattern=[%t] %d{DATE} %5p %c.%M:%L - %m %n
10
11 log4j.appender.SpagoBI.append=false
12
13 log4j.appender.Quartz=org.apache.log4j.RollingFileAppender
14 log4j.appender.Quartz.File=${catalina.base}/logs/Quartz.log
15 log4j.appender.Quartz.MaxFileSize=10000KB
16 log4j.appender.Quartz.MaxBackupIndex=10
17 log4j.appender.Quartz.layout=org.apache.log4j.PatternLayout
18 log4j.appender.Quartz.layout.ConversionPattern= [%t] %d{DATE} %5p %c.%M:%L - %m %n
19
20 log4j.appender.SpagoBI_Audit=org.apache.log4j.FileAppender
21 log4j.appender.SpagoBI_Audit.File=${catalina.base}/logs/knowage_[1]\_OperatorTrace.log
22
23 log4j.appender.SpagoBI_Audit.layout=org.apache.log4j.PatternLayout
24 log4j.appender.SpagoBI_Audit.layout.ConversionPattern=%m%n
25
26 log4j.appender.CONSOLE = org.apache.log4j.ConsoleAppender
27 log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
28 log4j.appender.CONSOLE.layout.ConversionPattern=%c.%M: %m%n #
29
30
31 log4j.logger.Spago=ERROR, SpagoBI log4j.additivity.Spago=false
32
33 log4j.logger.it.eng.spagobi=ERROR, SpagoBI, CONSOLE
34 log4j.additivity.it.eng.spagobi=false
35
36 log4j.logger.it.eng.spagobi.commons.utilities.messages=ERROR, SpagoBI
37 log4j.logger.it.eng.spagobi.commons.utilities.urls.WebUrlBuilder=ERROR, SpagoBI
38 log4j.logger.org.quartz=ERROR, Quartz, CONSOLE
39 log4j.logger.org.hibernate=ERROR, SpagoBI
40
41 log4j.logger.audit=INFO, SpagoBI_Audit log4j.additivity.audit=false

```

If the user wishes to enable the tracing of the information to **DEBUG** level it is enough to modify the following line

```
1 log4j.logger.it.eng.spagobi=ERROR, SpagoBI, CONSOLE
```

in

```
1 log4j.logger.it.eng.spagobi=DEBUG, SpagoBI, CONSOLE
```

For further details we refer to the official Log4J documents.

2.13.4 Mail server

Knowage uses in some situations the mail server to send emails. The configuration of this feature can be done right straight through the Knowage GUI, after accessing with administrator credentials.

Selecting the category MAIL the user gets the list of parameters to configure for the following profiles:

- trustedStore;

- scheduler, used by the scheduler to send a report by mail;
- user, used directly by the user when he intends to send a report by mail;
- kpi_alarm, used by the alarm component to send mails.

Etichetta	Nome	Descrizione	Attivo	Tipo	Categoria
internal.security.encrypt.password	encrypt password	Enable the password encryption	No	NUM	SECURITY
changePwdmodule.len_min	Password Len Min	Minimum length	No	NUM	SECURITY
changePwdmodule.special_char	Special char	Special chars	No	STRING	SECURITY
changePwdmodule.upper_char	Upper char	Minimum a char must be in upper ...	No	STRING	SECURITY
changePwdmodule.lower_char	Lower char	Minimum a char must be in lower ...	No	STRING	SECURITY
changePwdmodule.number	Number	Minimum a char must be a number	No	STRING	SECURITY
changePwdmodule.alphabetical	Alphabetical	Minimum a char must be a letter	No	STRING	SECURITY
changePwdmodule.change	Change from last	The new pwd must be different fro...	No	STRING	SECURITY
changePwd.change_first	Change at first login	The pwd must be changed at first ...	No	STRING	SECURITY
changePwd.disactivation_time	Disactivation time	Number of months before disactiv...	No	NUM	SECURITY
changePwd.expired_time	Expired time	Number of days to the expiration	No	NUM	SECURITY
SPAGOB.I.SPAGOB.I.MODE.mode	SPAGOB.I mode	Enable the WebApplication or Por...	No	STRING	GENERIC_CONFIGURATION
SPAGOB.I.HOME.BANNER.view	show the banner	banner	No	STRING	GENERIC_CONFIGURATION
SPAGOB.I.HOME.FOOTER.view	show the footer	footer	No	STRING	GENERIC_CONFIGURATION
SPAGOB.I.MENU.pathTracked	pathTracked	pathTracked	No	STRING	GENERIC_CONFIGURATION
SPAGOB.I.LOOKUP.numberRows	SPAGOB.I LOOKUP numberRows	The number of rows showed in e...	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.RESOURCE_PATH_JNDI...	RESOURCE PATH JNDI NAME	The name of the JNDI variable tha...	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.SPAGOB.I.HOST_JNDI	SPAGOB.I HOST JNDI	HOST JNDI	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.SPAGOB.I.ADAPTERHTT...	ADAPTERHTTP URL	ADAPTERHTTP URL	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.TEMPLATE_MAX_SIZE	TEMPLATE MAX SIZE	TEMPLATE MAX SIZE	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.SPAGOB.I.CONTEXT	SPAGOB.I CONTEXT	SPAGOB.I CONTEXT	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.SESSION_EXPIRED_U...	SESSION EXPIRED URL	SESSION EXPIRED URL	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.DATASET.maxResult	DATASET maxResult	maxResult	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.SESSION_PARAMETER...	SESSION PARAMETERS MANA...	enabled	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.DATASET_FILE_MAX_...	DATASET FILE MAX SIZE	Max size for a file used as a dataset	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.APL.DATASET.MAX_RO...	DATASET MAX ROWS NUMBER ...	Max number of rows returned from...	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.EXECUTION.PARAME...	Parameter state persistence enab...	if true the default value for each p...	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.EXECUTION.PARAME...	Parameter state persistence scope	if equals SESSION the parameter ...	Si	STRING	GENERIC_CONFIGURATION
SPAGOB.I.EXECUTION.PARAME...	Parameter memento persistence e...	if true the last N values selected fr...	Si	STRING	GENERIC_CONFIGURATION

Fig. 2.3: Mail server configuration.

Moreover, each profile has the following values:

- smtphost: the smtp server,
- Smtpport: the port in use,
- from: the address to which the mail will be associated,
- user: the user of the server connection,
- password: user's password,
- security: the user must choose between NONE, SSL and STARTTLS.

2.13.5 Maximum file size

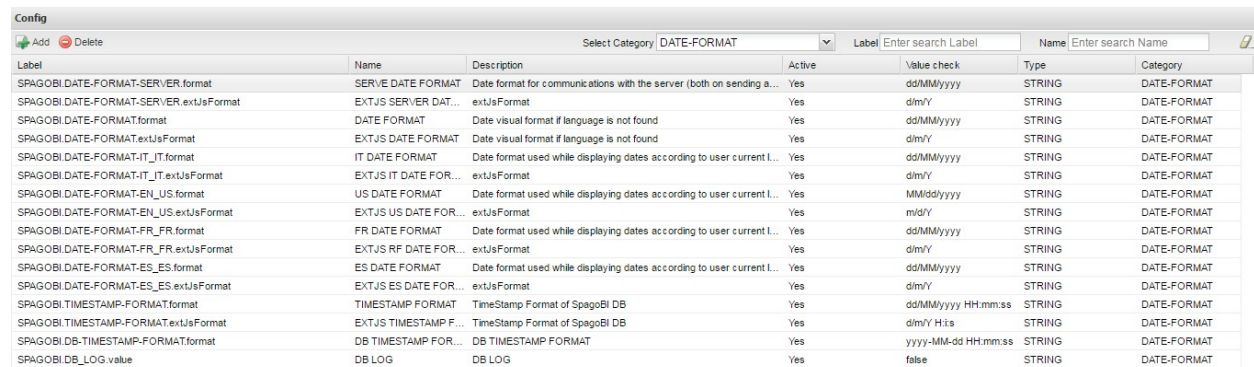
For security reasons, Knowage has a series of parameters which manage the maximum file size that can be loaded on the server through the web GUI. To modify those parameters, it is required to enter the Knowage server application as administrator and access the "server settings" section and then "configuration management". The parameters at issue are the following:

- **SPAGOB.I.TEMPLATE_MAX_SIZE** : TEMPLATE MAX SIZE: it is the maximum template dimension of an analytical document, expressed in bytes; the default value is 5MB;
- **SPAGOB.I.DATASET_FILE_MAX_SIZE** : DATASET FILE MAX SIZE: it is the maximum dimension of a file used as a dataset, expressed in bytes; the default value is 10MB;
- **SPAGOB.I.DOCUMENTS.MAX_PREVIEW_IMAGE_SIZE** : Max preview image size: it is the maximum dimension of an image used as document preview (in the document browser, for instance), expressed in bytes; the default is 1MB;

- **IMAGE_GALLERY.MAX_IMAGE_SIZE_KB** : Max image size in Kb: it is the maximum size of the images that can be used in a cockpit creation; the default is 1MB;

2.13.6 Date format

Knowage allows the user to visualize the date time in a format that depends on the selected language. To change the visualization of such formats, the user must enter Knowage as administrator and access the “Server Settings” section and, consequently, the “Configuration management”. Then finally select “DATE-FORMAT”.



Label	Name	Description	Active	Value check	Type	Category
SPAGOBI.DATE-FORMAT-SERVER.format	SERVE DATE FORMAT	Date format for communications with the server (both on sending a...	Yes	dd/MM/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-SERVER.extJsFormat	EXTJS SERVER DAT...	extJsFormat	Yes	d/m/Y	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT.format	DATE FORMAT	Date visual format if language is not found	Yes	dd/MM/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT.extJsFormat	EXTJS DATE FORMAT	Date visual format if language is not found	Yes	d/m/Y	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-IT_IT.format	IT DATE FORMAT	Date format used while displaying dates according to user current L...	Yes	dd/MM/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-IT_IT.extJsFormat	EXTJS IT DATE FOR...	extJsFormat	Yes	d/m/Y	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-EN_US.format	US DATE FORMAT	Date format used while displaying dates according to user current L...	Yes	MM/dd/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-EN_US.extJsFormat	EXTJS US DATE FOR...	extJsFormat	Yes	m/d/Y	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-FR_FR.format	FR DATE FORMAT	Date format used while displaying dates according to user current L...	Yes	dd/MM/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-FR_FR.extJsFormat	EXTJS RF DATE FOR...	extJsFormat	Yes	d/m/Y	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-ES_ES.format	ES DATE FORMAT	Date format used while displaying dates according to user current L...	Yes	dd/MM/yyyy	STRING	DATE-FORMAT
SPAGOBI.DATE-FORMAT-ES_ES.extJsFormat	EXTJS ES DATE FOR...	extJsFormat	Yes	d/m/Y	STRING	DATE-FORMAT
SPAGOBI.TIMESTAMP-FORMAT.format	TIMESTAMP FORMAT	TimeStamp Format of SpagoBI DB	Yes	dd/MM/yyyy HH:mm:ss	STRING	DATE-FORMAT
SPAGOBI.TIMESTAMP-FORMAT.extJsFormat	EXTJS TIMESTAMP F...	TimeStamp Format of SpagoBI DB	Yes	d/m/Y H:is	STRING	DATE-FORMAT
SPAGOBI.DB-TIMESTAMP-FORMAT.format	DB TIMESTAMP FOR...	DB TIMESTAMP FORMAT	Yes	yyyy-MM-dd HH:mm:ss	STRING	DATE-FORMAT
SPAGOBI.DB_LOG.value	DB LOG	DB LOG	Yes	false	STRING	DATE-FORMAT

Fig. 2.4: Date format configuration.

For each available language there are two parameters:

- **SPAGOBI.DATE-FORMAT-<lingua>_<nazione>.format**: it rules the back-end role;
- **SPAGOBI.DATE-FORMAT-<lingua>_<nazione>.extJsFormat**: it rules the front-end role.

We suggest to set the parameters in compliance with each other, according to a local data.

The parameters **SPAGOBI.DATE-FORMAT-SERVER.format** and **SPAGOBI.DATE-FORMAT-SERVER.extJsFormat** control the link between back-end and front-end. The adjustment of these parameters do not affect the web GUI.

2.13.7 Language

Knowage manages the multi-language. The list of all languages is manageable from the “Server Settings” section. Go to “Configuration management” and select the LANGUAGE_SUPPORTED category. Here there are two properties:

- **SPAGOBI.LANGUAGE_SUPPORTED.LANGUAGES** :the list of all supported languages underneath this formalism are: [it,IT],[en,US],[fr,FR],[es,ES];
- **SPAGOBI.LANGUAGE_SUPPORTED.LANGUAGE.default**: the default value is [en,US].

2.13.8 LDAP security connectors

Knowage provides integration with a LDAP server for authentication purposes.

Remark. Be sure that the Knowage users have been taken under LDAP census. The LDAP security connectors check the user that is accessing Knowage, but the user must be already defined as a Knowage user. Therefore, the users must coexist in both authentication systems (LDAP and Knowage).

Knowage ships with two LDAP security connectors:

- **LdapSecurityServiceSupplier**: a pure LDAP connector that authenticates every user using the LDAP server,
- **ProfiledLdapSecurityServiceSupplier**: a mixed LDAP connector that can authenticate some users using the LDAP server and other users using the internal Knowage authentication mechanism.

LdapSecurityServiceSupplier relies only on a LDAP configuration file, instead ProfiledLdapSecurityServiceSupplier checks also the Knowage user profile attribute **auth_mode**. If the user profile attribute **auth_mode** is defined and its value equals to **internal** for the logging user, then Knowage will use its internal authentication mechanism, otherwise it will try an authentication via LDAP.

Warning: The only way to maintain access to Knowage for **users not mapped onto LDAP** is to:

- define the user profile attribute **auth_mode**,
- set **auth_mode = internal** for every user not mapped onto LDAP,
- use the connector **ProfiledLdapSecurityServiceSupplier** (see below).

In order to setup any LDAP security connector, prepare a .properties file that includes the LDAP configuration:

- **INITIAL_CONTEXT_FACTORY**: initial context factory Java class,
- **PROVIDER_URL**: LDAP server IP,
- **SECURITY_AUTHENTICATION**: authentication type,
- **DN_PREFIX**: prefix that will be prepended to the user name to create the DN (distinguished name) of logging user,
- **DN_POSTFIX**: postfix that will be appended to the user name to create the DN (distinguished name) of logging user;

Important: The final concatenation **DN_PREFIX + <Knowage user ID> + DN_POSTFIX** must be equal to the **DN (distinguished name)** of the user as defined in LDAP server. Please check DN examples at <https://ldapwiki.com/wiki/DN%20Syntax>.

An example of LDAP configuration is the file `ldap_authorizations.properties`, available in the project `knowageldapsecurityprovider`.

Then define a custom JVM property `ldap.config`, setting its value to the path of LDAP configuration file.

In a Unix-like environment using Apache Tomcat you can add a custom JVM property to variable `JAVA_OPTS` in a `setenv.sh` file under `bin` folder:

```
export JAVA_OPTS="${JAVA_OPTS} -Dldap.config=/opt/tomcat/resources/ldap.properties"
```

In a Windows environment using Apache Tomcat you can add a custom JVM property to variable `JAVA_OPTS` in a `setenv.bat` file under `bin` folder:

```
set JAVA_OPTS="%JAVA_OPTS% -Dldap.config=C:/Tomcat/resources/ldap.properties"
```

Below there is an example of the `ldap.properties` file configuration for the **profiled** LDAP connector:

```
INITIAL_CONTEXT_FACTORY      = com.sun.jndi.ldap.LdapCtxFactory
PROVIDER_URL                 = ldaps://XXX.XXX.XXX.XXX:389
SECURITY_AUTHENTICATION      = simple
DN_PREFIX                    = CN=
DN_POSTFIX                   = ,ou=IT staff,o="Example, Inc",c=US
SEARCH_USER_BEFORE           = true
```

(continues on next page)

(continued from previous page)

```
SEARCH_USER_BEFORE_USER      =  
SEARCH_USER_BEFORE_PSW      =  
SEARCH_USER_BEFORE_FILTER    = (&((objectclass=person))(uid=%s))
```

Set `SEARCH_USER_BEFORE` key as *true*, if you want to looking for the complete distinguish name before checking authentication. Otherwise set it to *false*.

The `SEARCH_USER_BEFORE_USER` and `SEARCH_USER_BEFORE_PSW` keys are credentials to authenticate to LDAP server; if the first one is set, the second one will be considered also. *These parameters are used only if anonymous bind is not allowed for LDAP server. For this reason they are optional and can be empty.*

The `SEARCH_USER_BEFORE_FILTER` key is the filter used to retrieve the user on the LDAP server; Knowage uses the *username* as a parameter to find it. **Pay attention that %s placeholder must present.**

Important: Restart your application server in order to load the custom JVM property.

The final step is to set the LDAP security connector as follow:

- access Knowage as administrator,
- browse to **Configuration Management** via the main menu,
- set the value of config **SPAGOBI.SECURITY.USER-PROFILE-FACTORY-CLASS.className** to `it.eng.spagobi.security.LdapSecurityServiceSupplier` or `it.eng.spagobi.security.ProfiledLdapSecurityServiceSupplier`,
- save,
- log out of Knowage.

Warning: To recover the default authentication mechanism please revert manually the config **SPAGOBI.SECURITY.USER-PROFILE-FACTORY-CLASS.className** to its default value `it.eng.spagobi.security.InternalSecurityServiceSupplierImpl` using a database client.

Knowage is now ready to authenticate the users via LDAP credentials.

2.13.9 Google Sign-In integration

Knowage provides integration with Google Sign-In for authentication purposes, i.e. users can login into Knowage using their Google accounts.

Remark. Google Sign-In is exploited only for authentication, not for authorization. The authorization part (that consists of defining roles and profile attributes for users) is still in charge of Knowage: this means that, before an user can enter into Knowage with his Google account, the admin user has to define that user inside the regular Knowage users catalogue, keeping in mind that the user id must match the email of his Google account (the provided password will not be considered when user will log in). In case the user is not defined inside Knowage users list, the authentication will fail.

In order to enable Google Sign-In authentication, please follow these steps:

- start Knowage with default configuration;
- enter Knowage as admin and define users who will be allowed to enter Knowage (use their email address as user id);
- browse to **Configuration management** via the main menu;

- set the value of configuration parameter **SPAGOBI.SECURITY.USER-PROFILE-FACTORY-CLASS.className** to `it.eng.spagobi.security.GoogleSecurityServiceSupplier`, then save;
- stop Knowage application;
- follow Google Sign-In documentation in order to configure your Knowage application and to get the OAuth client ID;
- create a text file wherever you want and paste the client ID inside it, for example: create file `TOMCAT_HOME/conf/google.signin.config.properties` with this content:

```
client_id=<your client ID>.apps.googleusercontent.com
```

- add the `google.signin.config` Java System property that specifies the location of this properties file: in a Unix-like environment, edit `TOMCAT_HOME/bin/setenv.sh` and add

```
export JAVA_OPTS="${JAVA_OPTS} -Dgoogle.signin.config=<your Tomcat home folder>/conf/google.signin.config.  
↪properties"
```

instead, in a Windows environment using Apache Tomcat, edit `TOMCAT_HOME/bin/setenv.bat`:

```
set JAVA_OPTS="%JAVA_OPTS% -Dgoogle.signin.config=<your Tomcat home folder>/conf/google.signin.config.  
↪properties"
```

- start Knowage application.

When users will connect into Knowage, the login page will display the Google Sing-In button:

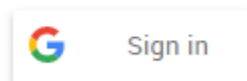


Fig. 2.5: Advanced configuration - Google Sing-In button.

When clicking on that button, Google Sing-In authentication applies; if successful, users will be redirected into their home page inside Knowage. In case the account is not recognized by Knowage (i.e. the provided email address does not match any existing Knowage user id), he will get an error message:

2.13.10 Password constraints settings

User password constraints can be set configuring parameters below:

- **changepwdmodule.len_min**: minimum number of character for the password;
- **changepwdmodule.special_char**: set of allowed special characters;



AUTHENTICATION FAILED. PLEASE CHECK IF YOU ARE TO ALLOWED TO ENTER THIS APPLICATION.

Fig. 2.6: Advanced configuration - Authentication error.

- **changepwdmodule.upper_char**: if active, the password must contain at least one of the uppercase characters set in the value;
- **changepwdmodule.lower_char**: if active, the password must contain at least one of the lowercase characters set in the value;
- **changepwdmodule.number**: if active, the password must contain at least one of the digit set in the value;
- **changepwdmodule.alphabetical**: if active, the password must contain at least one alphabetical set in the value;
- **changepwdmodule.change**: if true, new password must be different from the latest;
- **changepwd.change_first**: if true, password must be changed at first login;
- **changepwd.disactivation_time**: number of months before deactivation;
- **changepwd.expired_time**: number of days for the expiration.




















Label	Name	Value Check	Category	Active	
changepwdmodule.len_min	Password Len Min	8	SECURITY	false	 
changepwdmodule.special_ch...	Special char	_!-#\$	SECURITY	false	 
changepwdmodule.upper_char	Upper char	ABCDEFGHIJKLMNOPQRSTUVWXYZ...	SECURITY	false	 
changepwdmodule.lower_char	Lower char	abcdefghijklmnopqrstuvwxyz	SECURITY	false	 
changepwdmodule.number	Number	0123456789	SECURITY	false	 
changepwdmodule.alphabetical	Alphabetical	abcdefghijklmnopqrstuvwxyzAB...	SECURITY	false	 
changepwdmodule.change	Change from last		SECURITY	false	 
changepwd.change_first	Change at first login	true	SECURITY	true	 
changepwd.disactivation_time	Disactivation time	6	SECURITY	true	 
changepwd.expired_time	Expired time	90	SECURITY	true	 

Fig. 2.7: Advanced configuration - password constraints settings.

By default, all above configurations are disabled.

2.13.11 Login security settings

Login security configurations can be set filling fields below:

- **internal.security.login.checkForMaxFailedLoginAttempts**: if active and set to true, users will only be able to access Knowage if they have not reached the maximum number of failed login attempts;
- **internal.security.login.maxFailedLoginAttempts**: the maximum number of failed login attempts.





Label	Name	Value Check	Category	Active	
internal.security.login.checkFo...	Max failed login attempts filter	true	SECURITY	true	 
internal.security.login.maxFail...	Max failed login attempts	10	SECURITY	true	 

Fig. 2.8: Advanced configuration - login security settings.

2.13.12 Resource export folder cleaning settings

Resource export folder cleaning configurations can be set filling fields below:

- **KNOWAGE.RESOURCE.EXPORT.FOLDER.CLEANING_PERCENTAGE**: if active, the cleaning procedure will delete the files contained in the export resource folder leaving this percentage of free space (0 - 100). Default 30;
- **KNOWAGE.RESOURCE.EXPORT.FOLDER.MAX_FOLDER_SIZE**: if active, cleaning procedure will start only if the resource export folder will reach this size (byte). Default 10737418240.

2.13.13 Import / Export

2.13.13.1 Users

Specific configurations for users import procedure:

- **IMPORTEXPORTE.USER.DEFAULT_PASSWORD**: password set for all users imported by the import procedure.

3.1 Knowage at a glance

3.1.1 Discovering Knowage

Knowage is the business intelligence suite developed and managed by Engineering Group. Knowage is *flexible*, since it is based on a modular architecture and open standards in order to facilitates its customization and integration according to users' needs. It also provides a *comprehensive* set of analytical features and capabilities ranging from traditional reporting and charting tools, to more advanced analytics.

Important: Enterprise Edition only

KnowageER and KnowageSI, as submodules of Knowage Enterprise Edition, also supports **multi-tenancy** (i.e. a single Knowage instance serving multiple organizations, called tenants). In a multi-tenancy architecture, each tenant owns and manages his own users, documents, configuration and parameters, which are completely independent from those owned by other tenants.

3.1.2 Modules

Knowage Server The core of the suite, which includes all analytical features. This is the reference environment for the end user and the administrator;

Knowage Meta The environment supporting the creation and management of metadata models. This module is conceived for BI developers.

Knowage Studio The environment that allows BI developers to configure analysis and related analytical features that are then released and made available to the end user on the Server.

Knowage SDK A set of APIs compliant with the SOA architecture, allowing external applications to interact with the Knowage Server and its metadata.

Knowage Applications A set of analytical models ready to use in specific business domains.

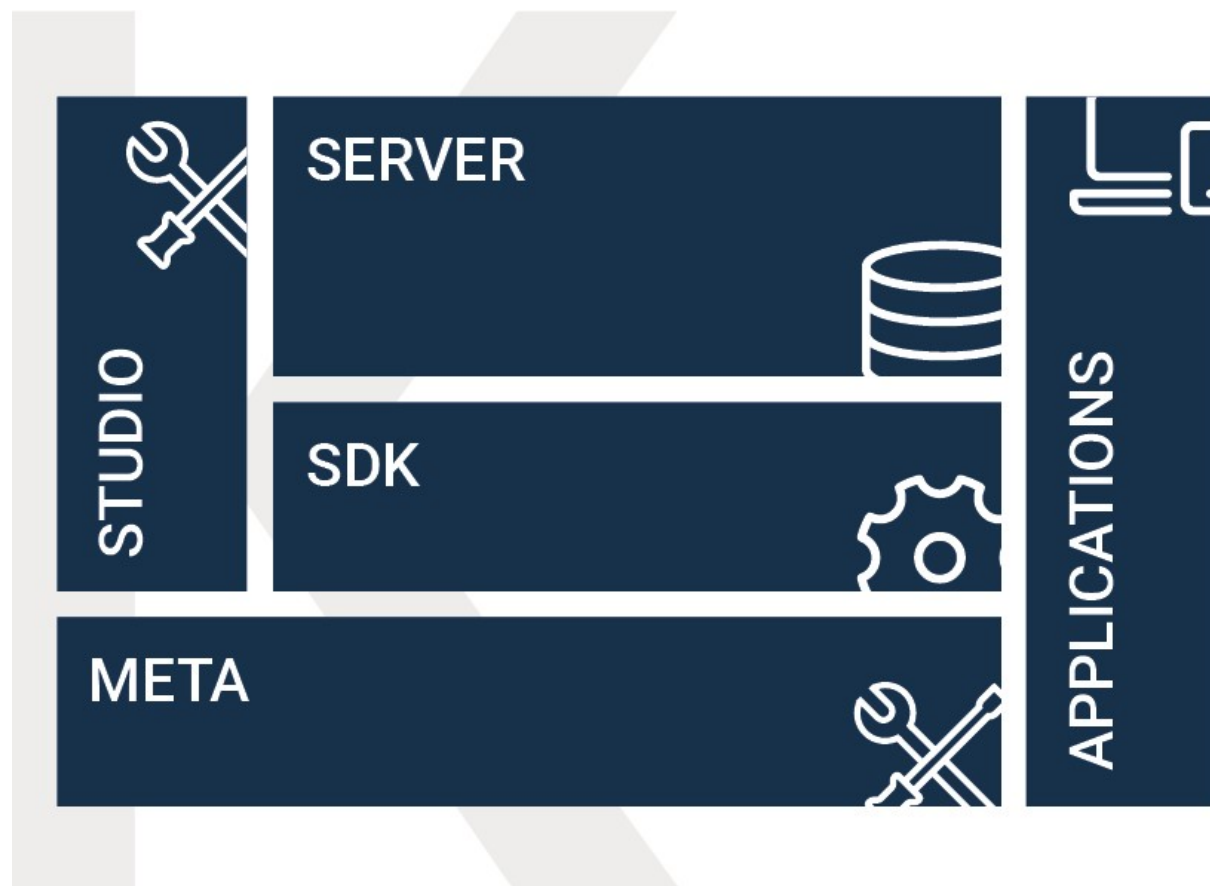


Fig. 3.1: Knowage modules.

Here we focus on Knowage Server considering the administrator perspective.

It is the main module of the suite, which provides, as we will see, the whole analytical power of the product and all administrative functionalities.

It represents an enterprise level solution for BI, supporting the whole project life-cycle, managing security and guaranteeing scalability, clustering and high availability architectures. Moreover, it is the main reference for all potential users and usages; it leads the development trend in terms of features, services and delivery models.

3.1.3 Layers

Knowage Server architecture is functionally layered on three main levels.

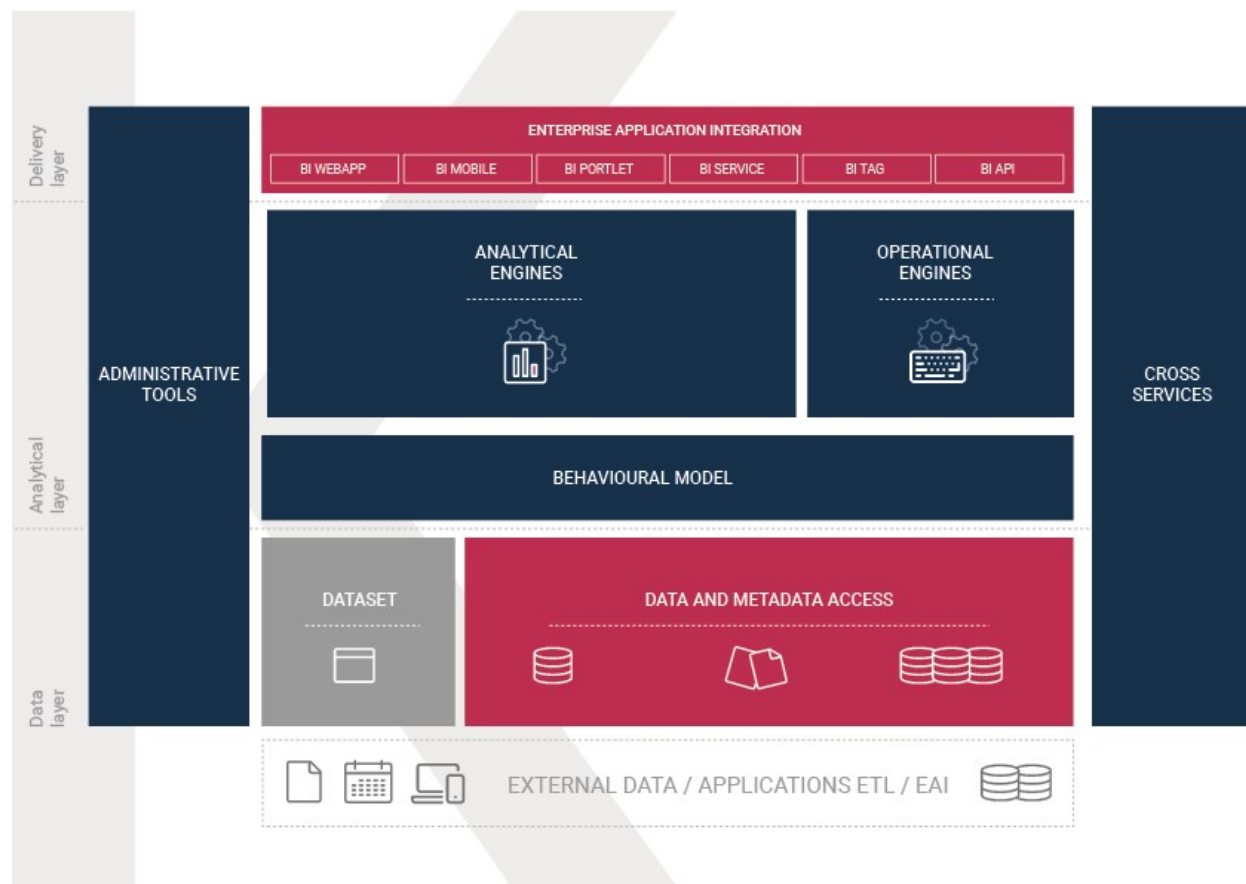


Fig. 3.2: Knowage Server architecture structure.

Delivery layer It manages all possible usages of the Server by end users or from external applications.

Analytical layer It provides all analytical functionalities of the product.

Data layer It regulates data loading through many access strategies.

Every layer of the functional architecture is composed of different application modules.

3.1.3.1 Delivery layer

The delivery layer covers all publication requirements. It can be accessed by third-party applications, and it offers end users all features and services needed to perform their BI analysis.

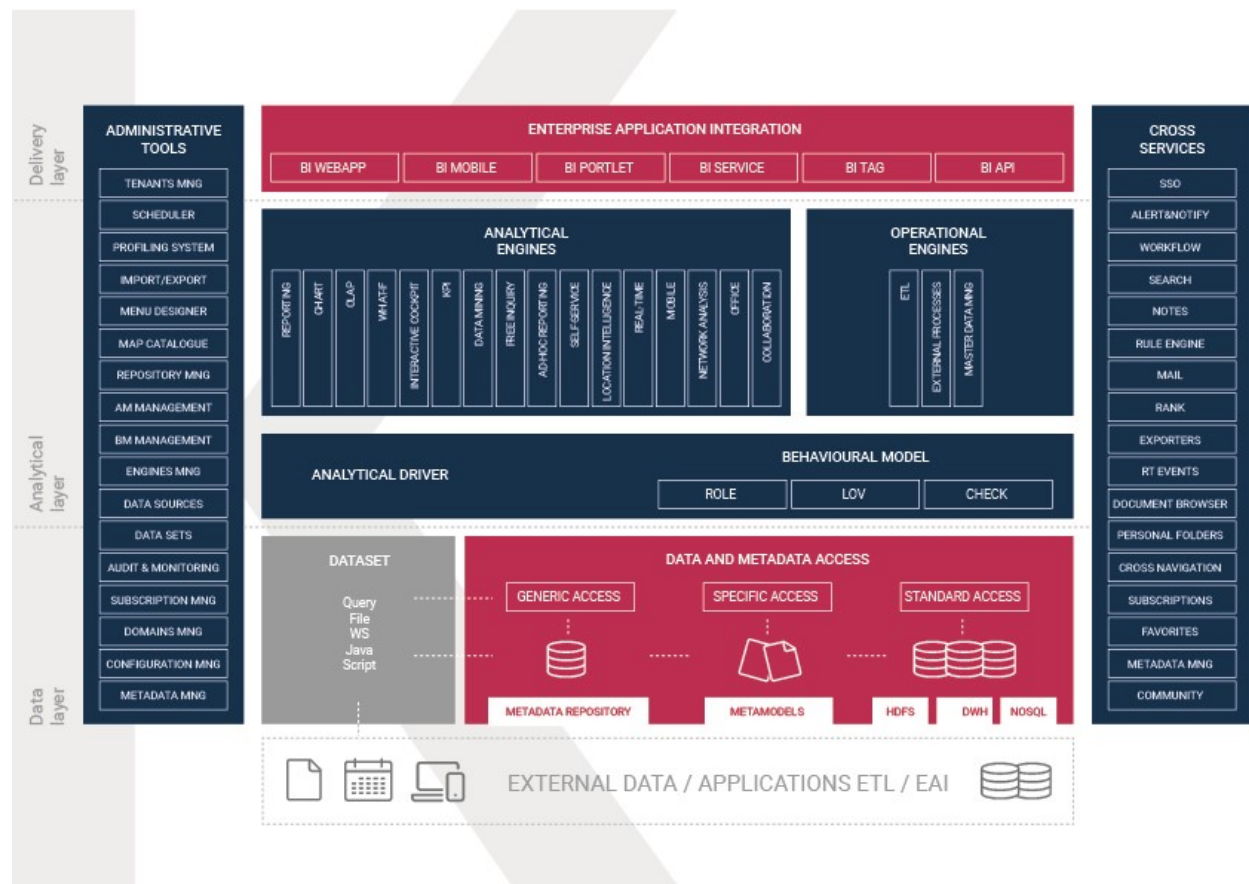


Fig. 3.3: Knowage Server architecture detail.

It can be accessed in different ways:

BI Webapp It is the default use mode. Knowage suite provides a web application, working on a standard application server. A customizable web application is provided, working on a standard application server (e.g. Tomcat, JBoss, WAS). The administrator can define the layout and specific views for each end user type.

BI Mobile Thanks to the interaction between Knowage Server and the remote client interface, users' reports, charts or cockpits can also be accessed and displayed on mobile devices, such as tablets and smartphones.

BI Service Web services allowing Knowage components to interact with external applications or to collect the results of static documents (report, image of a chart, etc.).

BI Tag Tag libraries allowing you to encapsulate a dynamic document (OLAP, GEO, etc.) into a different context.

BI API For the integration of enterprise applications behind or without the end user GUI.

3.1.3.2 Analytical layer

The analytical layer is the core of the Server. It provides all analytical features and capabilities, in a secure and role-based access mode. Its main components are:

Analytical Engines covering all analytical requirements, it provides different tools for each type of analysis (e.g. reports, charts, cockpits), in order to ensure high flexibility and end users satisfaction.

Operational Engines to interact with OLTP systems by means of ETL or processes, and manage basic BI registries such as master data or lookup domains;

Behavioural Model which regulates the visibility over documents and data, according to end users' roles.

Offering multiple solutions for the same analytical requirement and/or multiple instances for the same engine, Knowage logic and architecture provide various benefits, such as: limited workload on each engine, guaranteeing high performances; openness to improve or extend the suite and its capabilities, minimizing the impact on existing environments; high flexibility and modularity; high scalability, with minimum economic, infrastructural and application level impact.

3.1.3.3 Data layer

The data layer allows data and metadata storage and usage. BI data is often located in a data warehouse, whose design is out of the BI product scope and strictly related to the specific customer's world. Most of Knowage products offer a specific ETL tool allowing to load data at this level, covering the whole BI stack.

Knowage can directly access the data warehouse through JDBC connections (for instance, using SQL queries) or, on a higher level, it can use a specific access strategy based on metamodels, built through Knowage Meta.

As described in the next chapters, Knowage can also access less traditional data sources, like Big Data and NoSQL data sources.

All Knowage metadata are stored in a private repository hosted on a generic RDBMS and accessed by means of a generic description based on Hibernate technology. Knowage metadata contains technical information, business metadata and metamodels registry.

3.1.4 What you can do with Knowage

This section focuses on Knowage analytical and operational functionalities, administration tools and cross services.

It is important to point out that Knowage adopts an evolutionary approach, allowing you to use and adapt the different features provided by the suite according to your specific needs, and adapt them over time. The Server reflects this

strategy, guaranteeing security and consistency, thanks to the independence of the behavioural model that regulates visibility over documents and data.

Moreover, Knowage has a distributed logic and handles more instances of a same engine. This allows the workload distribution on several servers, ensuring the linear system scalability.

3.1.4.1 Analytical and operational functionalities

Knowage server provides a wide range of analytical functionalities, covered by the different products of the suite.

Concerning the operative level, Knowage Server works with:

- **ETL**, not only for the continuous loading of source data into the DWH, but even for the internal movement of data, high-level consolidations or returning of the produced information to the operational systems.
- **External processes**, for a bidirectional interaction with operational systems and external ones.
- **Master data**, to manually manage domain data.

3.1.4.2 Administrative tools and cross services

Besides its analytical, delivery and data access capabilities, Knowage Server provides all the administration tool needed to handle your Knowage instance, as well as several cross-product services to make its features even more powerful.

The **administrative tools** support developers, testers and administrators in their daily work, providing various functionalities, such as: scheduler, profiling system, import/export capabilities, menu designer, map catalogue, management of repository, analytical model, behavioural model and engines, configuration of data sources and data sets, audit & monitoring analysis, subscriptions, management of value domains, configuration settings and metadata, management of user data, hierarchies editor and community management.

The **cross services** include the common features of the product, shared by all analytical engines and documents. They are: single sign on, alert and notification, workflow, search engine, collaborative tools, sending e-mails, ranking, multiformat exporter, RT events, document browser, personal folders, cross navigation, subscription service, hot link, metadata view.

3.2 User Interface

This chapter focuses on Knowage user interface, providing detailed information on the Main Menu.

3.2.1 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.

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.

Knowage main menu is divided in three submenus: the user menu, the BI functionalities menu and the administrator one. The left panel contains the user menu, which is collapsed, and the BI functionalities menu, while the administrator menu is on the right panel.

The user menu (see below) is identified with the first icon in Table above and a label containing your user name. Opening the user menu you have the following extra buttons:



Fig. 3.4: Home page

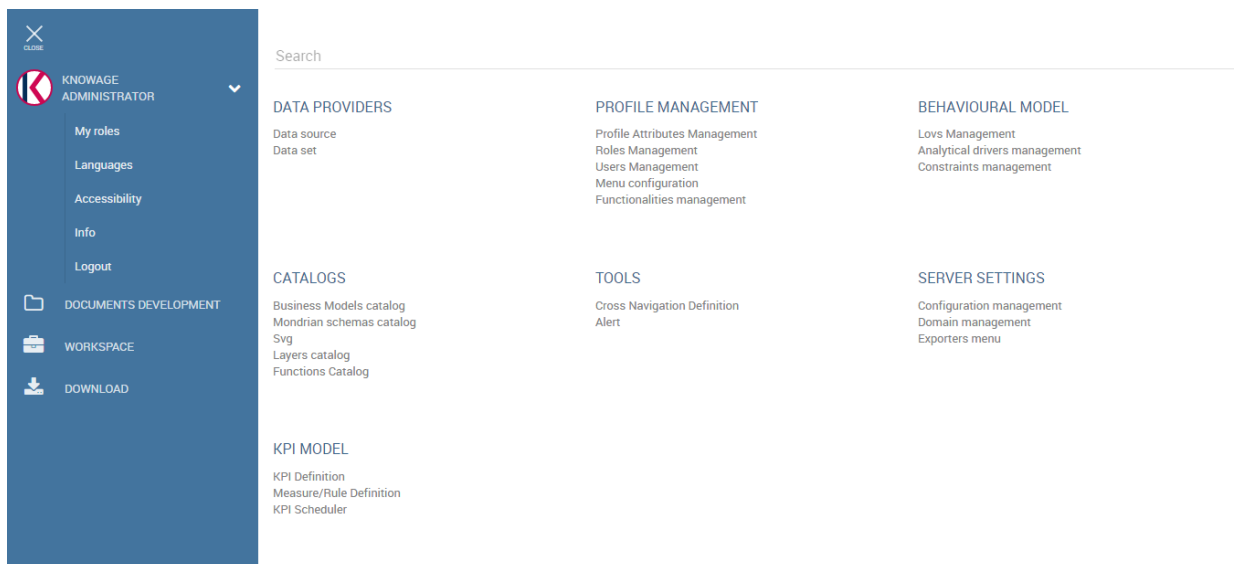







Fig. 3.5: Menu

- **My roles** If your user is associated with more than one role, Knowage requests you to specify the default one. You can select it when executing a document, or right after authentication by clicking on this icon and choosing a default role.
- **Languages** To select the language of Knowage environment.
- **Accessibility** To enable or disable accessibility mode.
- **Info** To view the details of current Knowage version.
- **Logout** To go back to log in page.

The BI functionalities menu consists in a set of icons associated with basic features:

- **Documents development** This is a standard functionality of Knowage Server. It enables you to create the analytical document as well as access and execute them.
- **Workspace** From this section you can access the **Models** and create datasets and federations.

Table 3.1: Menu components - BI Functionalities Menu.

Icon	Name	Description
	My Account	Show or update user info.
	Documents development	Document creation and acces to the archive folders.
	Workspace	Inquiry your business models, navigate and create your datasets.
	Download	Open a dialog that show the available async download.
	News	Open a dialog that show the news.

Important: **My Account** is visible only when SPAGOBI.SECURITY.MY_ACCOUNT_MENU configuration is enabled.

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 Analytical 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 analytical 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 constraints 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 analytical 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.

3.2.1.1 My Account menu

My account button, if enabled, allows user to access Modify profile page. Here you can modify your information and also permanently delete your account.

3.2.1.1.1 Modify

With this feature user can modify their own data such as Name, Surname and Email address. Password and Confirm password fields are mandatory so they must be filled before applying changes clicking on “MODIFY”.

Attention: If user fills *Password* and *Confirm password* fields with a valid password that is NOT the current password (used to log in), the password will be updated with the new value.

3.2.1.1.2 Delete account

With this feature user can delete his account clicking on *DELETE ACCOUNT* button. Then a message confirms the correct cancellation and the login page is displayed.

Attention: Account cancellation is irreversible. User will no longer be able to log into Knowage.

3.2.1.1.3 Password change

If user wants to change his password, he must fill *Password* and *Confirm Password* fields and the click on *MODIFY* button. A confirmation message will be displayed if the update has been successful.

3.3 Configure data sources

To let all the BI tools work properly you need to configure DB connection. There are two different options available for the configuration **JNDI** (recommended) and **JDBC**.

3.3.1 Connect to your data

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:

- connections retrieved 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) and thus are the preferred ones.
- direct JDBC connections, which are directly managed by Knowage;

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

Modify profile

Name *	Surname *
Knowage	User
Username *	Email address
biuser	knowageuser@knowage.com
Password *	Confirm Password *

MODIFY

DELETE ACCOUNT

Fig. 3.6: My account - Modify profile

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 the figures above) 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:

Table 3.2: Certified Data Sources

Certified Data Sources	
Oracle	11, 12
MySQL	5.7, 8
PostgreSQL	8.2, 9.1, 12.3
Maria DB	10.1, 10.2, 10.3
Teradata	15.10.0.7
Vertica	9.0.1-0
Cloudera	5.8.9
Apache Hive 1	1.1.0
Apache Hive 2	2.3.2
Apache Impala	2.6.0
Apache Spark SQL	2.3.0
Apache Cassandra	2.1.3
Mongo DB	3.2.9
Orient DB	3.0.2

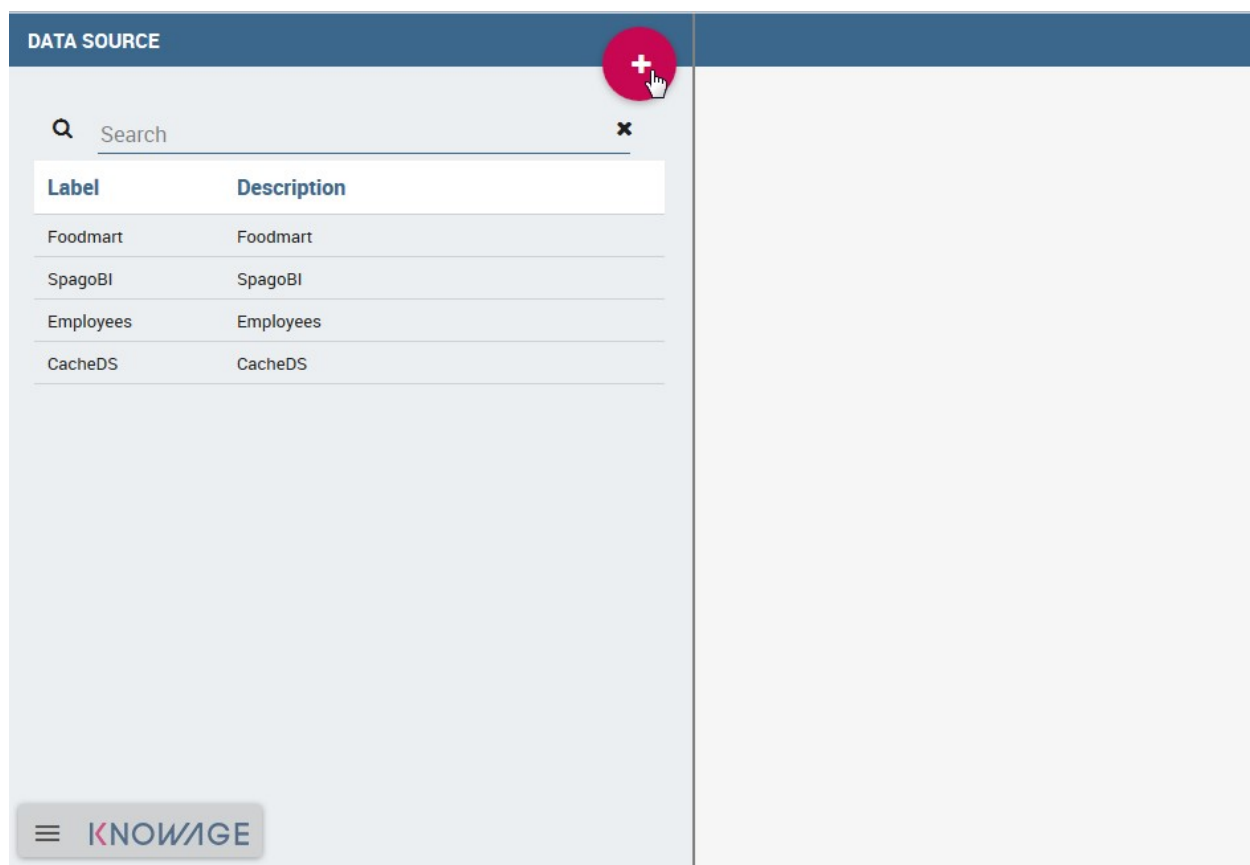


Fig. 3.7: Add a new data source

TEST SAVE CLOSE

Label

This is required

Description

Dialect

This is required

Multischema: ☐

Read only:

☐ Read only

☒ Read and write

Write Default: ☐

Type:

☒ JDBC

☐ JNDI

URL

This is required

User

Password

Driver

This is required

Fig. 3.8: Data source details.

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

Type The available options are

- 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!

3.3.2 Big Data and NoSQL

In this section we describe how you can connect Knowage to different Big Data data sources.

Important: Enterprise Edition only

Please note that these connections are available for products KnowageBD and KnowagePM only.

3.3.2.1 Hive

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem. It provides an SQL-like language called HiveQL with schema on read and transparently converts queries to map/reduce, Apache Tez and Spark. All three execution engines can run in Hadoop YARN.

Every distribution of Hadoop provides its JDBC driver for Hive. We suggest you to use or the Apache one or the one specific of your distribution. In general the JDBC driver for Hive is composed by different jars, and so you should deploy the JDBC driver with all dependencies in your application server. If you are creating a model you should create a new *Data Source Connection* and import the JDBC driver and all the dependencies.

For example suppose you want to connect to Hive using Apache driver you should include these libraries (according to your Hive version) shown in Figure below.

```
hadoop-core-0.19.0.jar
hive_metastore.jar
hive-common-0.14.0.jar
hive-exec-0.14.0.jar
hive-jdbc-0.14.0.jar
hive-metastore-0.14.0.jar
hive-service-0.14.0.jar
httpclient-4.2.5.jar
httpcore-4.2.5.jar
TCLIServiceClient.jar
libthrift-0.9.0.jar
commons-logging-1.1.3.jar
slf4j-api-1.6.1.jar
slf4j-log4j12-1.6.1.jar
log4j-1.2.16.jar
```

Fig. 3.9: Libraries to include in the apache driver.

If you forget to add one or more libraries, you will likely get a `NoClassDefFoundError` or `ClassNotFoundException`.

The parameters for the Hive connection are:

- **Dialect:** Hive QL;
- **Driver Class:** `org.apache.hive.jdbc.HiveDriver` (if you are not using some specific driver of some distribution. In this case search in the documentation of the distribution);
- **Connection URL:** `jdbc:hive2://<host1>:<port1>,<host2>:<port2>/dbName;sess_var_list?hive_conf_list#hive_var_list`.

Here `<host1>:<port1>,<host2>:<port2>` is a server instance or a comma separated list of server instances to connect to (if dynamic service discovery is enabled). If empty, the embedded server will be used.

A simple example of connection URL is `jdbc:hive2://192.168.0.125:10000`.

3.3.2.2 Spark SQL

Spark SQL reuses the Hive front end and metastore, giving you full compatibility with existing Hive data, queries and UDFs. Simply install it alongside Hive. For the installation of Spark we suggest you to look at the spark website <http://spark.apache.org/>. To create a connection to the Spark SQL Apache Thrift server you should use the same JDBC driver of Hive.

- **Driver Class:** `org.apache.hive.jdbc.HiveDriver` (if you are not using some specific driver of some distro. In this case search in the documentation of the distro);
- **Connection URL:** `jdbc:hive2://<host1>:<port1>,<host2>:<port2>/dbName;sess_var_list?hive_conf_list#hive_var_list`.

Look at the Hive section for the details about parameters. The port in this case is not the port of Hive but the one of Spark SQL thrift server (usually 10001).

3.3.2.3 Impala

Impala (currently an Apache Incubator project) is the open source, analytic MPP database for Apache Hadoop. To create a connection to Impala you should download the jdbc driver from the Cloudera web site and deploy it, with all dependencies, on the application server. The definition of the URL can be different between versions of the driver, please check on the Cloudera web site.

Example parameters for Impala connection are:

- **Dialect:** Hive SQL;
- **Driver Class:** `com.cloudera.impala.jdbc4.Driver`;
- **Connection URL:** `jdbc:\impala://dn03:21050/default`.

3.3.2.4 MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. MongoDB obviates the need for an Object Relational Mapping (ORM) to facilitate development.

MongoDB is different from the other dbs Knowage can handle, because it doesn't provide a JDBC driver, but a Java connector. The MongoDB Java driver (at this moment version 3.5.0 is included) is already included inside Knowage so isn't required to download and add it to the application server.

Example parameters for the connection are:

- **Dialect:** MongoDB;
- **Driver Class:** `mongo`;
- **Connection URL:** `“mongodb://localhost:27017/foodamart”` (please don't include user and password in this URL).

Also please pay attention that the user must have the correct privileges to access the specified database. So for example on MongoDB you can create a user with this command on the Mongo shell:

Listing 3.1: User creation.

```

1 db.createUser(
2   {
3     user: "user",
4     pwd: "user",
5     roles: [ { role: "readWrite", db: "foodmart" } ]
6   }
7 )

```

Then you must create a role that is able to run functions (this is the way used by Knowage to run the code wrote in the MongoDB's dataset definition) and assign it to the user:

Listing 3.2: Role assignment .

```

1 use admin
2 db.createRole( { role: "executeFunctions", privileges: [ { resource: { anyResource: true }, actions: [
3   ↪ "anyAction" ] } ], roles: [ ] } )
4 use foodmart
5 db.grantRolesToUser("user", [ { role: "executeFunctions", db: "admin" } ])

```

See also this useful links: - (<https://docs.mongodb.com/manual/tutorial/enable-authentication/>)
- (<https://www.claudiokuenzler.com/blog/555/allow-mongodb-user-execute-command-eval-mongodb-3.x#.W59wiaYzaUI>)

3.3.2.5 Cassandra

Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients.

There are different ways to connect Knowage to Cassandra.

If you are using Datastax Enterprise you can use Spark SQL connector and query Cassandra using pseudo standard SQL (https://github.com/datastax/spark-cassandra-connector/blob/master/doc/2_loading.md)

Another solution is to download the Apache JDBC Driver and query Cassandra using the language CQL. Also in this case the JDBC driver is composed by different jars, and so you should deploy the JDBC driver with all dependencies in your application server.

An example of Cassandra Apache driver (with dependencies) is:

- `apache-cassandra-clientutil-1.2.6.jar`
- `apache-cassandra-thrift-1.2.6.jar`
- `cassandra-all-1.2.9.jar`
- `cassandra-jdbc-2.1.1.jar`
- `guava-15.0.jar`
- `jackson-core-asl-1.9.2.jar`
- `jackson-mapper-asl-1.9.2.jar`
- `libthrift-0.7.0.jar`
- `log4j-1.2.16.jar`
- `sfl4j-api-1.6.1.jar`
- `sfl4j-log4j12-1.6.1.jar`

Example parameters for the connection are:

- **Dialect:** `Cassandra;`
- **Driver Class:** `org.apache.cassandra.cql.jdbc.CassandraDriver;`
- **Connection URL:** `jdbc:cassandra://193.109.207.65:9160/foodmart.`

Unless you are using Spark SQL to read from Cassandra, the definition of a business model over Cassandra data using Knowage Meta will be available in the next releases.

3.4 Behavioural Model

An important topic to explore before starting a new project is the creation and managing the so-called *behavioural model*.

It regulates the visibility on documents and data according to the roles and profiles of the end users. It offers many advantages in a BI project, including: reducing the required number of analytical documents to be developed and maintained; coding visibility rules once only and apply them to several documents, each one with its own analytical logics; ensuring a uniform growth of the project over time; guaranteeing the respect of the visibility rules over time, with no limitation on the number of engines and analytical documents that can be added over time.

The behavioural model is based on four main concepts:

- *user profile*, defining the user's roles and attributes;
- *repository rights*, defining the users' rights in terms of document accessibility;
- *analytical drivers*, defining which data of a document can be shown to the user;
- *presentation environment* settings, defining how the user can reach and run his own documents.

In other words, the behavioural model mainly answers the following questions:

- *WHO* uses the business intelligence solution (user profile);
- *WHAT* is visible to users, in terms of documents and data (repository rights and analytical drivers);
- *HOW* users work with their documents (analytical drivers and presentation environment settings).

3.4.1 Roles, users and attributes

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.

Table 3.3: Knowage Role Types.

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

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.

Pre-defined roles are summarized in the Table 5.1. The first four roles are technical roles, while the last one, the user, is the actual end user. Each role type has a default user associated to it. Other users can be created and associated to a role type.

When a user logs in into Knowage, his profile is automatically loaded. The full name is visible by clicking on the info button at the bottom left corner of the page.

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

Hint:

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 below, he can select a default role that will be kept valid until he logs out.



Fig. 3.10: User roles in Knowage.

The steps to create a behavioural 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** 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**: 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.

In addition to the profile attributes created by administrator, by default Knowage provides the following profile attributes:

- **user_id**: set with the user unique identifier;
- **user_roles**: set with user roles selected from the ROLES tab in Users Management menu;
- **TENANT_ID**: set with the tenant unique identifier;
- **user_session_roles**: set like *user_roles* attribute, if no default role is set. Set with default role selected, otherwise.
- **language**: set with the language selected by the user

Fig. 3.11: 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.

Hint:

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 types already described. A code and a description can be added too, as shown below.

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

The **Business Models**, **Data sets** and **KPI Categories** tabs are 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 KnowageSI, 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.

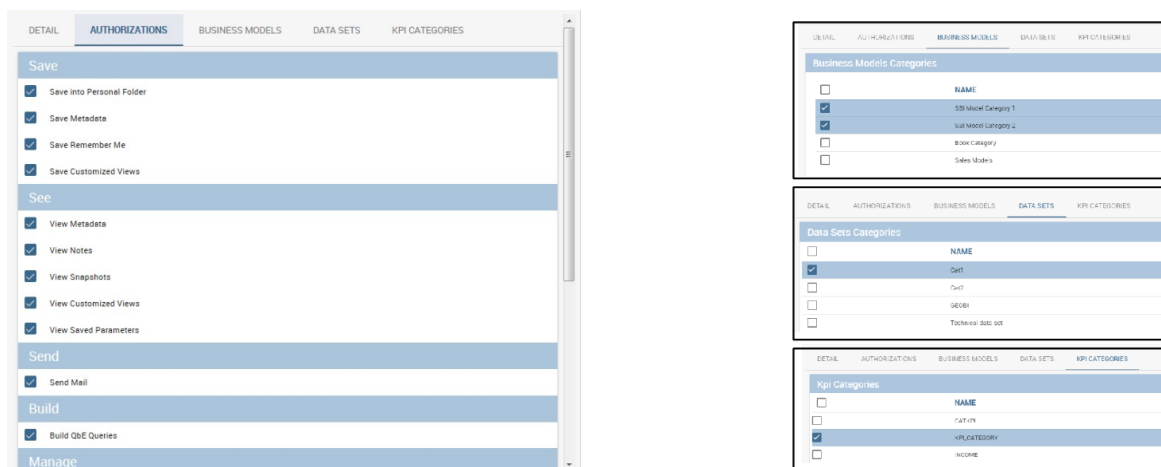


Fig. 3.12: Roles Management.

3.4.1.1 User Management

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.

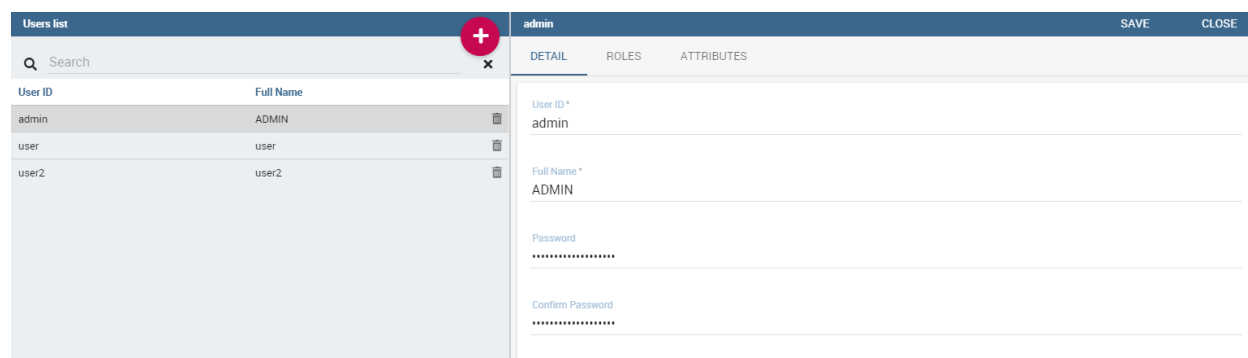


Fig. 3.13: Users Management.

3.4.1.1.1 User unlock

If user reaches the maximum number of failed login attempts (editable in advanced configurations), it will be blocked by Knowage and access will be denied. By accessing Knowage with an user having user management privileges, the blocked user will be displayed with a red warning sign and it will be possible to unlock it using the “unlock user” button. After that user will be able to log in using the latest credentials set.

3.4.1.1.2 Roles settings

Clicking on the ROLES tab you have to select one or more roles to associate with the user. After that, if more than one role is associated to the user, you can choose the default role by selecting it from the combo box on the top of the page.

Default role is optional: if you don't select a default role, at login time all roles available for the user will be loaded. If you select a role, at login time it will be the session role selected.

Users list + x

Search

User ID	Full Name
admin	ADMIN
blocked_user	blocked_user

blocked_user SAVE CLOSE

DETAIL **ROLES** **ATTRIBUTES**

THIS USER REACHED THE MAXIMUM NUMBER OF FAILED LOGIN ATTEMPTS AND THEREFORE HE WAS LOCKED. DO YOU WANT TO UNLOCK HIM? UNLOCK USER

User ID *
blocked_user

Full Name *
blocked_user

Password

Confirm Password

Fig. 3.14: Users Management - Roles settings example

bitest SAVE CLOSE

DETAIL **ROLES** **ATTRIBUTES**

Default role ▼

Select a role if you want to set it as the default. If no role is selected, it will be possible to switch between roles from the Knowage Administration menu

<input type="checkbox"/>	Name	Value
<input type="checkbox"/>	/test_importexport/admin	/test_importexport/admin
<input checked="" type="checkbox"/>	admin	admin
<input type="checkbox"/>	demo	demo
<input type="checkbox"/>	dev	dev
<input checked="" type="checkbox"/>	kte_dev	kte_dev
<input type="checkbox"/>	kte_end_user	End user role
<input type="checkbox"/>	modeladmin	modeladmin

Fig. 3.15: Users Management - Roles settings example

In the example above, for the user “*bitest*” you can choose from “admin” and “*kte_dev*” as default role.

3.5 Analytical Model

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

3.5.1 Repository structure and rights

Knowage adopts a virtual 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.

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.

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;
- **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.

Warning:

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.

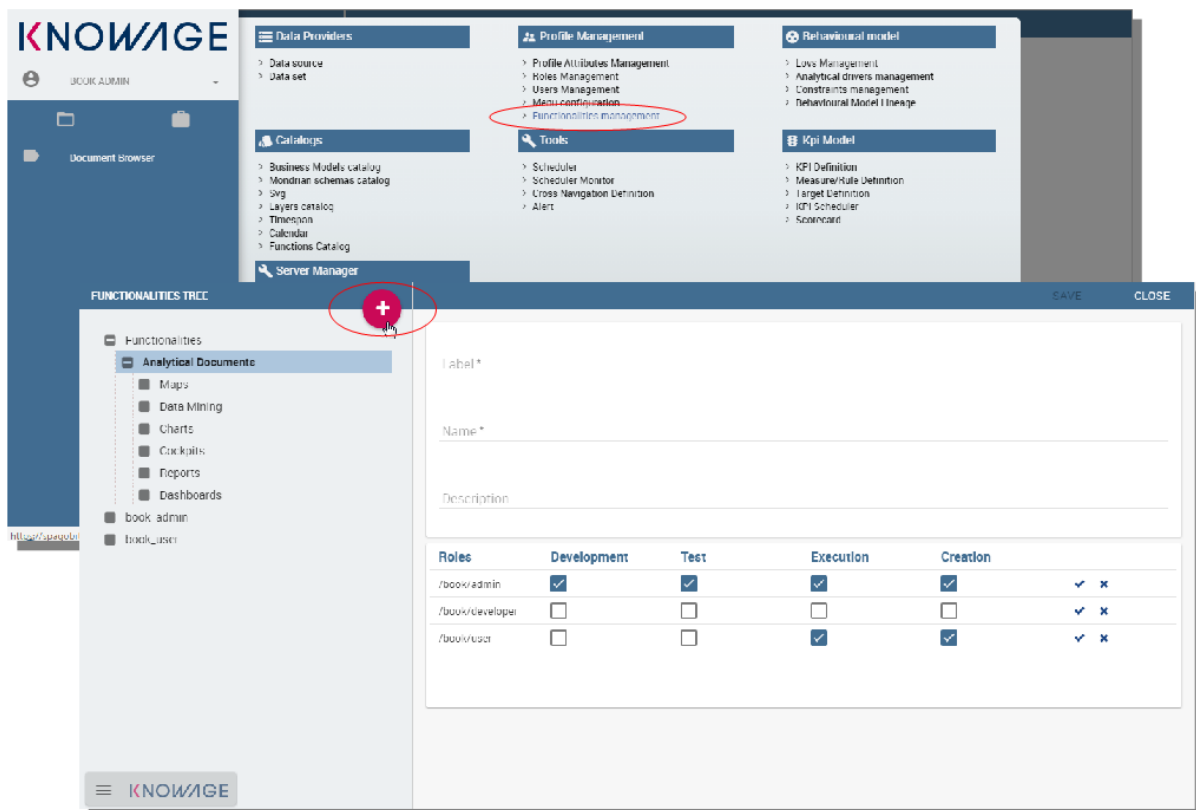
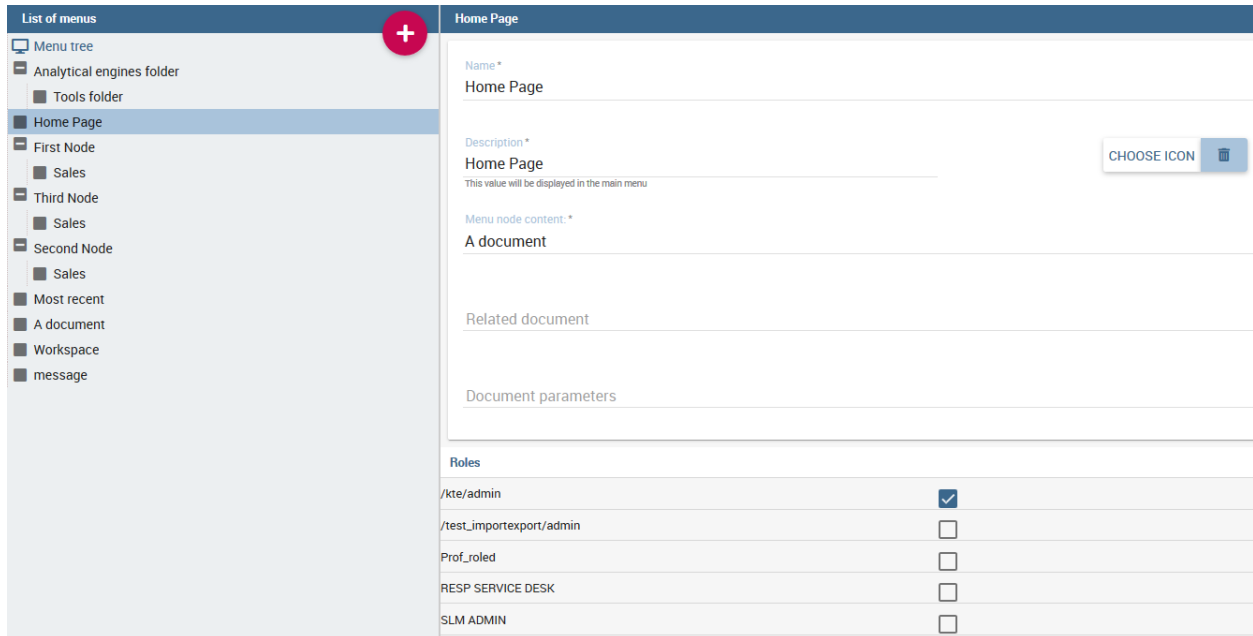


Fig. 3.16: Create a new folder and assign permissions.

3.5.2 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 is sufficient to click on the “Plus” of the Menu Configuration page to add a new folder to the **Menu Tree**. When selecting one node of the tree and clicking on the “Plus” icon, the user can add a new folder as a child of the former one.



The screenshot displays the 'Menu Configuration' interface. On the left, a 'List of menus' sidebar shows a tree structure with nodes like 'Menu tree', 'Analytical engines folder', 'Tools folder', 'Home Page', 'First Node', 'Third Node', 'Second Node', 'Most recent', 'A document', 'Workspace', and 'message'. A red '+' icon is visible above the 'Home Page' node. The main area on the right is titled 'Home Page' and contains a form for configuring this menu item. The form includes fields for 'Name *' (set to 'Home Page'), 'Description *' (set to 'Home Page' with a note 'This value will be displayed in the main menu'), 'Menu node content. *' (set to 'A document'), and 'Related document'. There is a 'CHOOSE ICON' button and a trash icon. Below the form is a 'Roles' table with checkboxes for various roles.

Roles	
/kte/admin	<input checked="" type="checkbox"/>
/test_importexport/admin	<input type="checkbox"/>
Prof_rolod	<input type="checkbox"/>
RESP SERVICE DESK	<input type="checkbox"/>
SLM ADMIN	<input type="checkbox"/>

Fig. 3.17: Menu tree actions.

In general you can:

- define a name: the name is a mandatory field and has to be unique;
- define a description: the description is a mandatory field and it is displayed in the main menu;
- choose an icon to associate to the menu and to be shown in the main menu;
- 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.

3.5.2.1 Setting menu icon

Creating a menu, you can choose an icon to be shown in the main menu. **This feature is available only for first level menu.**

To associate an icon to the menu, click on “*CHOOSE ICON*” and then you can:

- choose icon from the predefined icons included in Knowage;

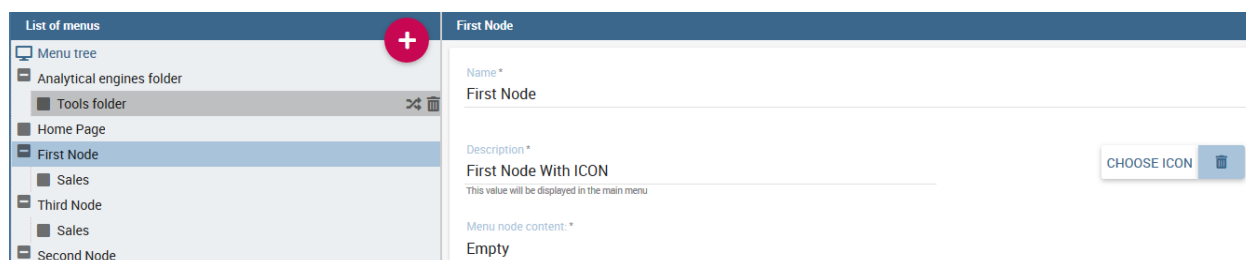


Fig. 3.18: Menu choose icon feature

- upload a new file containing a custom icon (*File type allowed are .ico, .svg, .png and the maximum size allowed is 50 KB*).

To remove the menu icon, click on the trash basket.

Dialog shown in figure below allows you to select an icon or click on “Browse...” to choose the image on your pc and upload it. After choose is completed, click on “*CHOOSE*” button to apply the changes. If no errors occurred, you’ll come back to menu configuration and see the icon you choose. If you want to filter predefined icons, you can start writing in the text field on the top of the dialog.

If you decide to choose a predefined icon, select it click on “CHOOSE” to apply the changes.

If you decide to upload a new image, click on “*Browse...*” and select your file. After that, the icon will be displayed as in the image below. Click on “*CHOOSE*” and apply the changes.

There are four types of menu item content: empty, document, static page and functionality.

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 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.

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 below, runs a URL address.

3.6 Operational engines

3.6.1 ETL

Knowage allow the upload of data from source systems according to a common ETL logic, as well as the monitoring of data flows continuously feeding the data warehouse. To this end, Knowage provides the ETL **Knowage Talend Engine**.

Important: Enterprise Edition only

Please note that in the Enterprise Edition, **KnowageTalendEngine** is shipped with KnowageBD and KnowageSI only.

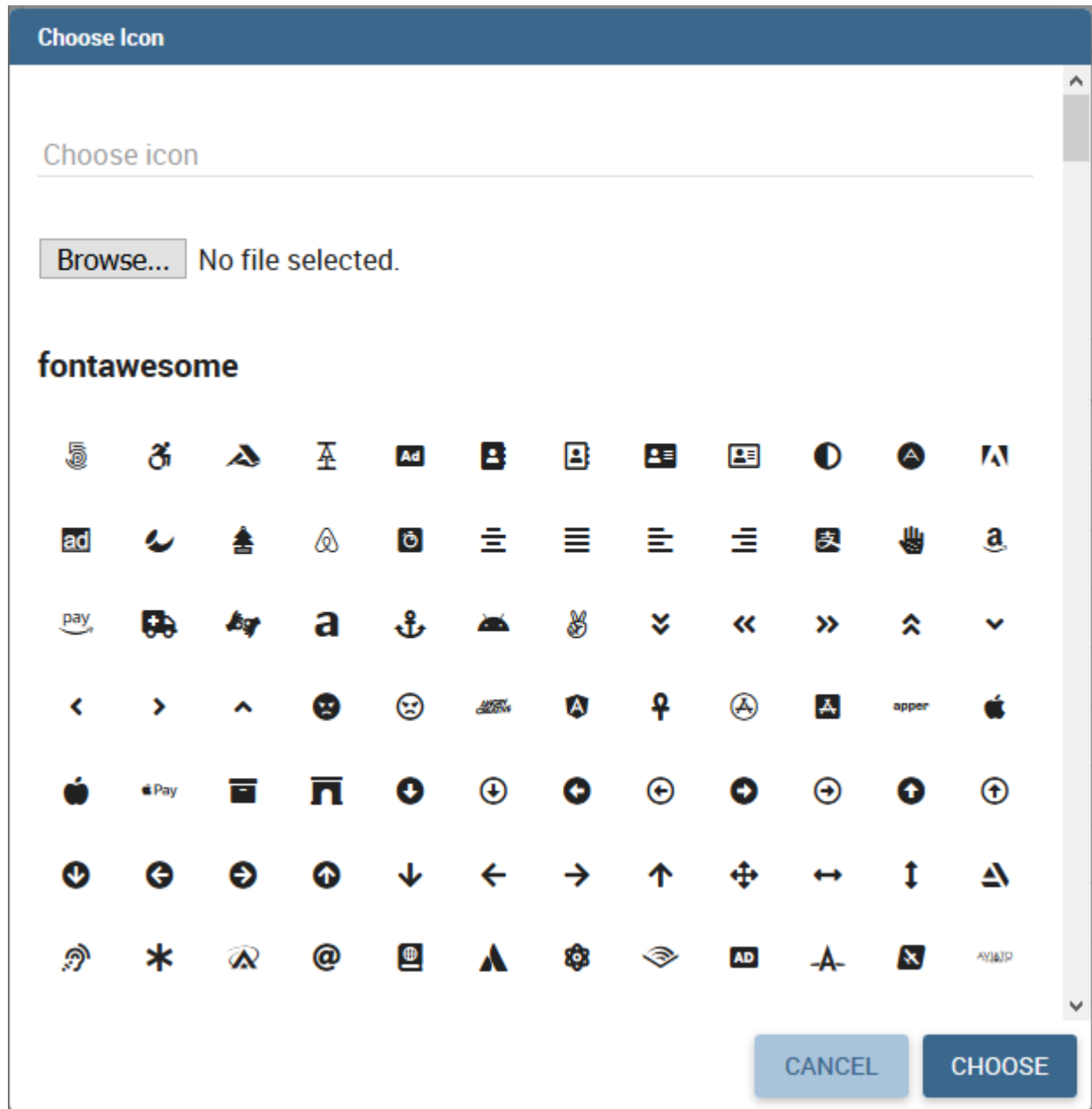


Fig. 3.19: Menu choose icon dialog

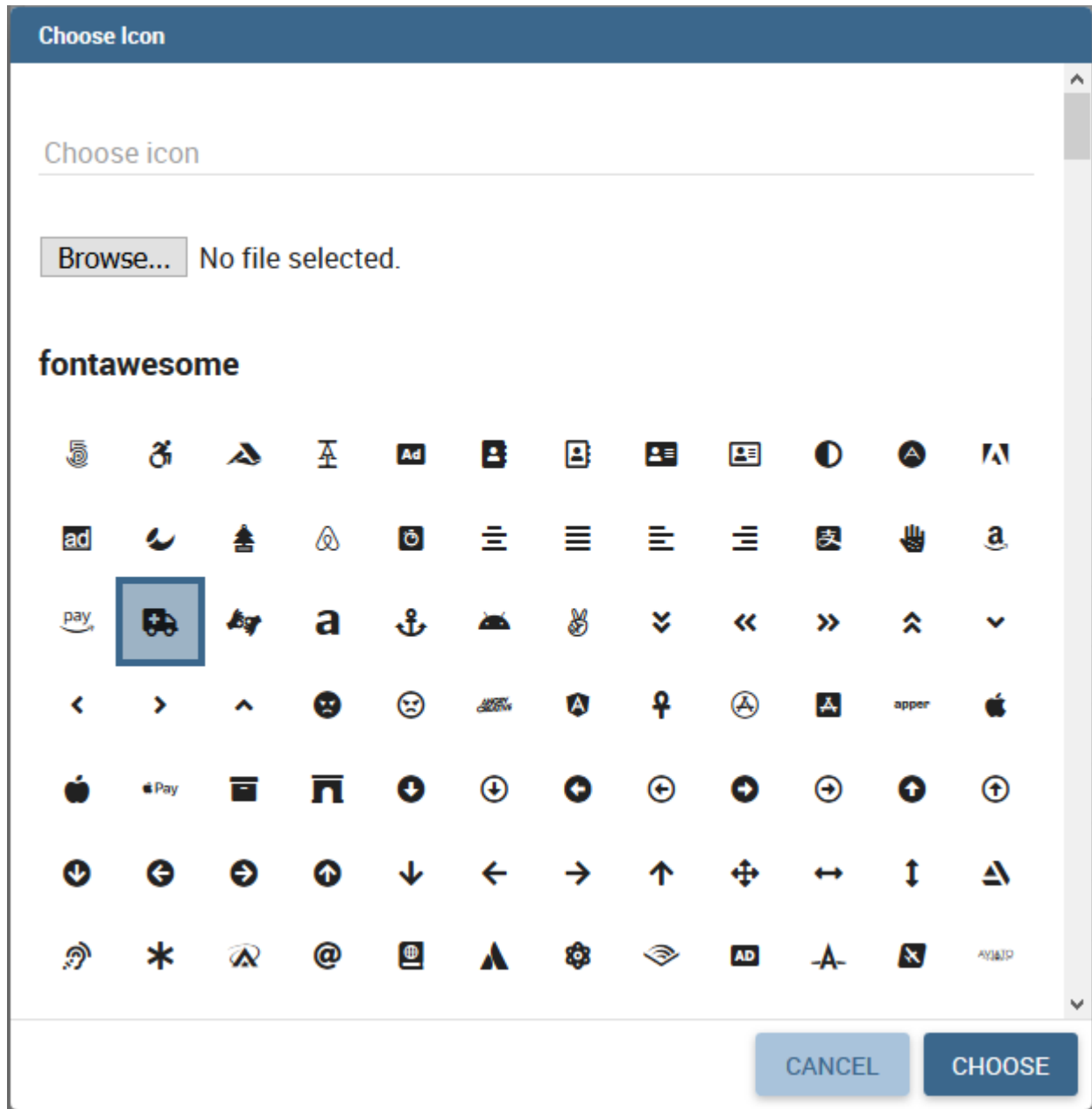


Fig. 3.20: Menu choose selected icon

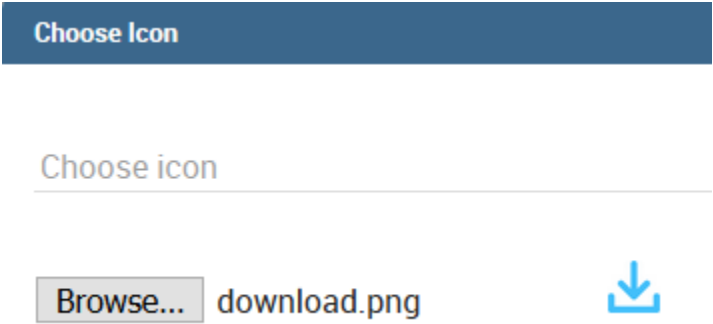


Fig. 3.21: Menu choose uploaded icon

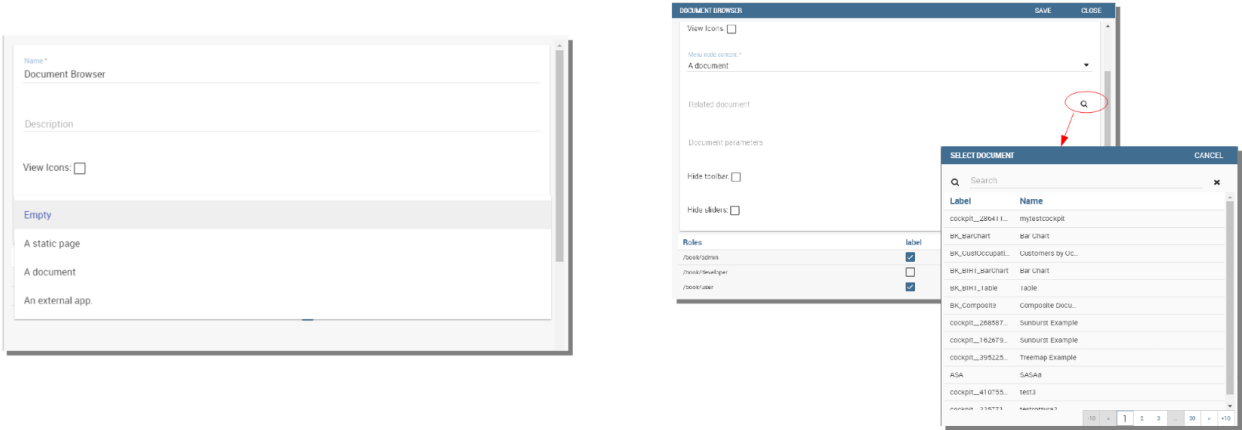


Fig. 3.22: **Empty** (left) and **document** (right) content type.

DOCUMENT BROWSER
SAVE
CLOSE

View Icons: ☐

Menu node content: *

An external app.

Application's url

Roles	label
/book/admin	<input checked="" type="checkbox"/>
/book/developer	<input type="checkbox"/>
/book/user	<input checked="" type="checkbox"/>

Fig. 3.23: External application content type.

3.6.1.1 KnowageTalendEngine

Knowage Talend Engine integrates the open source tool Talent Open Studio (TOS). Talend Open Studio (TOS) is a graphical designer for defining ETL flows. Each designed flow produces a self-alone Java or Perl package. TOS is based on Eclipse and offers a complete environment including test, debug and support for documentation.

The integration between Talend and Knowage is twofold. TOS includes Knowage as a possible execution target for its job, while Knowage implements the standard context interface for communicating with Talend. Jobs can be directly executed from Knowage web interface or possibly scheduled.

Furthermore, the analytical model for monitoring ETL flows can be successfully applied to the analysis of audit and monitoring data generated by a running job. Note that this is not a standard functionality of Knowage suite, but it can be easily realized within a project with Knowage. To create an ETL document, you should perform the following steps:

- Job design (on Talend);
- Job deploy;
- Template building;
- Analytical document building;
- Job execution.

In the remainder of the section, we discuss in detail all steps by providing examples.

3.6.1.2 Job design

The job is designed directly using Talend.

Designing an ETL job requires to select the proper components from Talend tool palette and connect them to obtain the logic of the ETL flow. Talend will map to appropriate metadata both the structure of any RDBMS and the structure of any possible flow (e.g., TXT, XLS, XML) acting as input or output in the designed ETL.

To design the ETL, several tools are available: from interaction with most RDBMS engines (both proprietary and open source) to support for different file formats; from logging and auditing features to support for several communication and transport protocols (FTP, POP, code, mail); from ETL process control management to data transformation functionalities.

Talend also supports data quality management. Furthermore, it enables the execution of external processes and can interact with other applications, e.g., open source CRM applications, OLAP and reporting tools.

The tMap tool allows the association of sources to targets according to defined rules. The main input is the source table in the data warehouse, while secondary (or lookup) inputs are dimensions to be linked to data. The output (target) is the data structure used for aggregation.

It is also possible to design parametric ETL jobs. We will see how to manage them in the next steps.

Once you have designed the ETL job, you should deploy it on Knowage Server. First of all, configure connections properties to Knowage Server. Select **Preferences > Talend > Import/export** from within Talend. Then set connection options as described below.

Table 3.4: Connection Settings.

Parameter	Value
Engine name	KnowageTalendEngine
Short description	Logical name that will be used by Talend
Host	Host name or IP address of the connection URL to Knowage
Port	Port of the connection URL to Knowage
Host	Host name or IP address of the connection URL to Knowage
Password	Password of the user that will perform the deploy

Once you have set the connection, you can right click on a job and select **Deploy on Knowage**. This will produce the Java code implementing the ETL and make a copy of the corresponding jar file at `\\resources\\talend\\RuntimeRepository\\java\\Talend project name of Knowage Server`. It is possible to deploy multiple jobs at the same time. Exported code is consistent and self-standing. It may include libraries referenced by ETL operators and default values of job parameters, for each context defined in the job. On its side, Knowage manages Talend jobs from an internal repository at `resources/talend/RuntimeRepository`, under the installation root folder.

3.6.1.3 Template building

As with any other Knowage document, you need to define a template for the ETL document that you wish to create. The ETL template has a very simple structure, as shown in the example below:

Listing 3.3: ETL template.

```

1 <etl>
2   <job    project="Foodmart"
3         jobName="sales_by_month_country_product_familiy"
4         context="Default"
5         version="0.1"
6         language="java"
7   />
8 </etl>

```

Where the tag job includes all the following configuration attributes:

- project is the name of the Talend project
- jobName is the label assigned to the job in Talends repository.
- context is the name of the context grouping all job parameters. Typically it is the standard context, denoted with the name **Default**.
- version is the job version
- language is the chosen language for code generation. The two possible options are: Java and Perl.

Values in the template must be consistent with those defined in Talend, in order to ensure the proper execution of the ETL document on Knowage Server.

3.6.1.4 Creating the analytical document

Once we have created the template, we can create a new analytical document.

Before starting to create the document, it is recommended to check whether the engine is properly installed and configured. In case the engine is not visible in the Engine Configuration list (**Data Providers > Engine Management**), you should check that the web application is active by invoking the URL `http://myhost:myport/KnowageTalendEngine`.

Now you can create the analytical document on the Server, following the standard procedure. The template for this document is the one we have just created. If the job has parameters, they should be associated to the corresponding analytical drivers, as usually. In other words, you have to create an analytical driver for each context variable defined in the Talend job.

3.6.1.5 Job execution

A Talend job can be executed directly from the web interface of Knowage Server and of course from a Talend client. To execute the job on Knowage, click on the document icon in the document browser, like with any other analytical document. The execution page will show a message to inform that the process was started.

3.6.1.6 Job scheduling

Most often it is useful to schedule the execution of ETL jobs instead of directly running them. You can rely on Knowage scheduling functionality to plan the execution of Talend jobs. While defining a scheduled execution, you can set a notification option which will send an email to a set of recipients or a mailing list once the job has completed its execution. To enable this option, check the flag **Send Mail**.

3.6.2 External processes

Knowage support the execution of processes that are external to its own activity. When analyzing data, for example through the real time console, it may be useful to perform activities such as sending notification emails or taking actions on the components of the monitored system (e.g., business processes, network nodes).

These products provide the KnowageProcessEngine, which supports the execution and management of external processes.

With the term *process* we refer to a Java instruction, however complex it may be. Processes can be executed in background or via the interface of the Console Engine. It is also possible to schedule their start and stop.

To enable the management of an external process, the following steps are required:

- Create a Java class defining the execution logic;

- If needed, create a Java class defining the logic of the process, i.e., which tasks the process is supposed to perform (optional);
- Create a template that will be associated to the Knowage document;
- Create the Knowage CommonJ analytical document;

In the following sections, we provide details about both class and template creation, and document creation.

3.6.2.1 Class definition

First of all, the developer should write a Java class that defines the desired logics for processing start and stop. In particular, this class must extend one of these two classes of the engine:

KnowageWork In this case the class to be defined only needs to reimplement the `run()` method. This class is the base case: the logic of the external process will be contained in the `run()` method.

CmdExecWork

In this case, the class to be defined must implement the method `execCommand()`. The logic of the external process can be delegated to an external class, which will be invoked by the `execCommand()` method. To stop the process, the developer is in charge of checking programmatically whether the process is still running, using the method `isRunning()`, or not.

Listing 3.4: Class template

```

1 package it.eng.spagobi.job;
2
3 import java.util.Iterator;
4 import it.eng.spagobi.engines.commonj.process.SpagoBIWork;
5
6 public class CommandJob extends SpagoBIWork {
7     @Override
8     public boolean isDaemon() {
9         return true;
10    }
11
12    @Override
13    public void release() {
14        System.out.println("Release!!");
15        super.release();
16    }
17
18    @Override public void run() {
19        super.run();
20        System.out.println("Job started! ");
21        java.util.Map parameters = getSbiParameters();
22        for (Iterator iterator = parameters.keySet().iterator(); iterator.hasNext();) {
23            String type = (String) iterator.next();
24            Object o = parameters.get(type);
25            System.out.println("Parameter " + type + " value" + o.toString());
26        }
27        for(int i = 0; i < 50 && isRunning(); i++) {
28            System.out.println("job is running!");
29            try {
30                Thread.sleep(2000);
31            } catch (InterruptedException e) {
32                e.printStackTrace();
33            }
34        }
35        System.out.println("Job finished!");
36    }
37 }

```

Note that the class `CmdExecWork` extends `SpagoBIWork` by providing additional methods. To better understand the difference between the two options, let us have a look at some code snippets. Here you can see a class implemented as an extension of `SpagoBIWork`:

Note also that we only implement the `run()` method, embedding the logic of the process in it. Below you can see an example extension of `CmdExecWork`, called `CommandJob`:

Listing 3.5: Example extension of `CmdExecWork`.

```

1 package it.eng.spagobi.job;
2 import it.eng.spagobi.engines.commonj.process.CmdExecWork;
3 import java.io.IOException;
4 public class CommandJob extends CmdExecWork{
5     public boolean isDaemon() {
6         return true;}
7     public void release() {
8         super.release();}
9     public void run() {
10        super.run();
11        if(isRunning()){
12            try {
13                execCommand();
14            } catch (InterruptedException e) {
15            } catch (IOException e) {}}}}

```

Note that this class implements the `execCommand()` method and uses the `isRunning()` method. No logic is directly embedded in this class. Therefore, we also define an external class, called `ProcessTest`, which contains the actual logic (in our example printing the content of a file):

Listing 3.6: `ProcessTest`

```

1 package it.eng.test;
2 import java.io.FileNotFoundException;
3 import java.io.FileOutputStream;
4 import java.io.PrintStream;
5 public class ProcessTest {
6     public static void main(String[] args) {
7         FileOutputStream file=null;
8         try {
9             file = new FileOutputStream("C:/file.txt");
10        } catch (FileNotFoundException e) {
11            // TODO Auto-generated catch block
12            e.printStackTrace();}
13        PrintStream output = new PrintStream(file);
14        while (true){
15            output.println("New row");
16            output.flush();
17            try {
18                Thread.currentThread().sleep(5000l);
19            } catch (InterruptedException e) {
20                // TODO Auto-generated catch block
21                e.printStackTrace();
22            } output.close();}}

```

Now that classes are ready, we pack them in .jar file containing all classes and their paths. Then we copy the jar file under the resource folder of Knowage at `RESOURCE_PATH]/commonj/ CommonjRepository/[JAR_NAME`. In the next section we will explain how to define the template, based on the class definition chosen above.

3.6.2.2 Template definition

As with any other Knowage document, we need to define a template for an external process document. The example below shows a template that corresponds to the classes `CommandJob` and `ProcessTest` defined in the examples above.

Let us note that this template corresponds to the option of implementing an extension of `CmdExecWork`.

Listing 3.7: Template Definition

```

1 <COMMONJ>
2   <WORK workName='JobTest' className='it.eng.spagobi.job.CommandJob'>
3     <PARAMETERS>
4       <PARAMETER name='cmd' value='C:/Programmi/Java/jdk1.5.0_16/bin/java' />
5       <PARAMETER name='classpath'
6         value='C:/resources/commonj/CommonjRepository/JobTest/process.jar' />
7       <PARAMETER name='cmd_par' value='it.eng.test.ProcessTest' />
8       <PARAMETER name='sbi_analytical_driver' value='update' />
9       <PARAMETER name='sbi_analytical_driver' value='level' />
10    </PARAMETERS>
11  </WORK>
12 </COMMONJ>

```

Where:

- `<COMMONJ>` is the main tag and includes all the document.
- The tag `<WORK>` specifies the process. In particular:
 - `workName` is the id of the process
 - `className` contains the name of the class implementing the process (as defined above).
- The tag `<PARAMETERS>` contains all parameters. Each `<PARAMETER>` tag includes a parameter. Some of them are mandatory

Table 3.5: CommonJ document template parameters.

Parameter	Value
cmd	Specifies the java command that will be launched, with its complete path
classpath	Specifies the classpath containing the jar file. This path will be added to the classpath for the process to run correctly.
cmd_par	Optional. In case it is defined, its value contains the Java class that will be launched instead of the job (i.e., the extension of <code>CmdWorkExec</code> or <code>KnowageWork</code>).
sbi_analytical_driver	Optional and repeatable. Each line with this attribute defines an analytical driver that should be associated with the process.

The class `CmdExecWork` (and its extensions) allows the execution of the command specified in the template. In particular, the template above would produce the following command at runtime:

Listing 3.8: Runtime command line

```

1 C:/Programmi/Java/jdk1.5.0_16/bin/java 'it.eng.test.ProcessTest'
2 update=<val> level=<val>

```

3.7 Scheduler

Knowage scheduler allows to schedule the execution of one or more analytical documents published on the Server. Documents executed by the scheduler can then be distributed along different dispatching channels. In the following we describe how to create an activity, schedule it and dispatch the results.

3.7.1 Create an Activity

In order to define a new scheduled activity the administrator must specify which documents compose the activity and how to execute them. The list of all scheduled activities can be seen selecting **Tools > Scheduler**. To create a new activity click on the “Plus” icon at the top of the page in the left area. In Figure below you can see the main scheduler page and the new activity GUI.

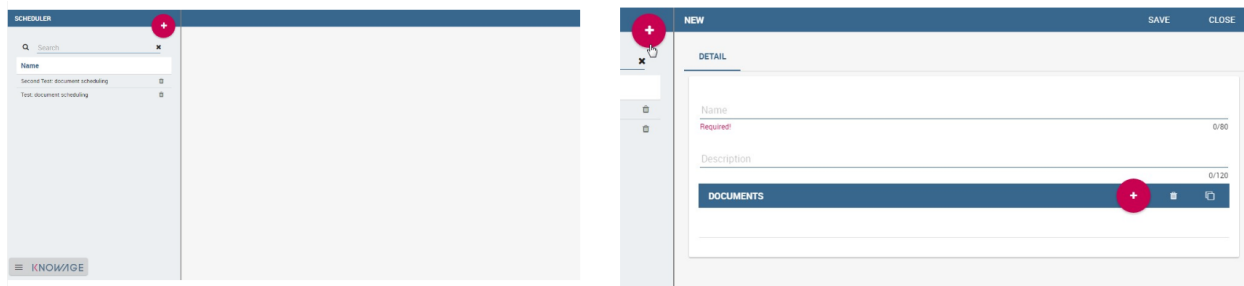


Fig. 3.24: Left: scheduler main page. Right: New activity GUI

Give a name and a description to the new activity. Then select the documents that compose it by clicking on the “Plus” icon and selecting them from the pop up wizard, see Figure below.

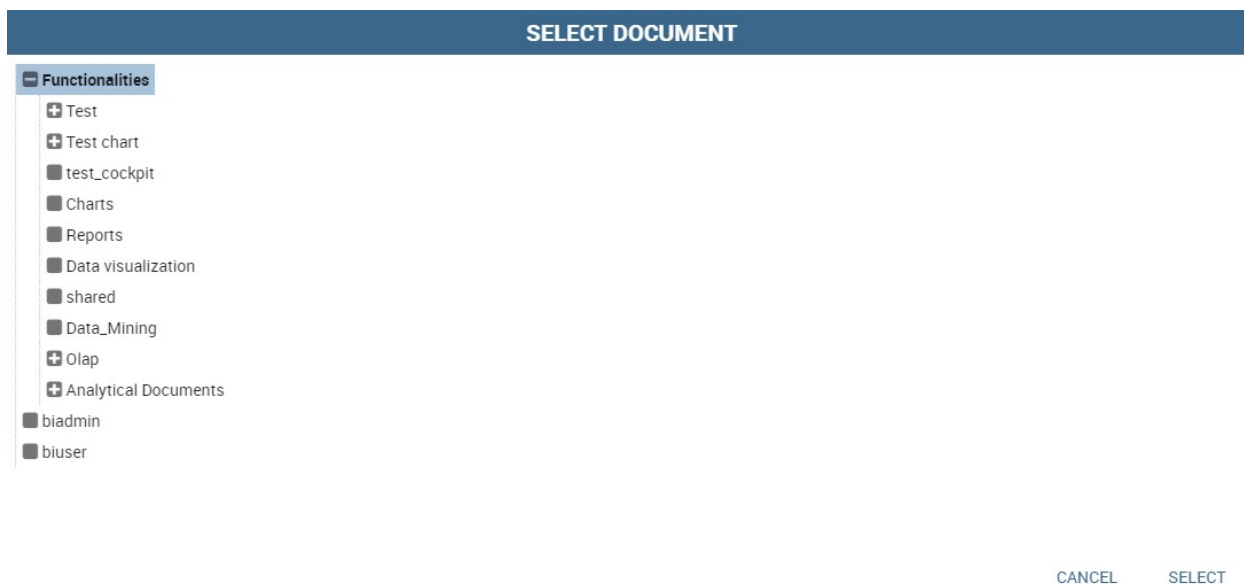



Fig. 3.25: Adding a document to an activity.

Now you need to specify how the scheduler must handle the analytical drivers of each selected document having parameters.

There are two possibilities:

- selecting a value from those available for the specific analytical driver at definition time;
- executing the report one time for each possible values of the analytical driver.

A scheduled activity can be composed by more than one report. It is also possible to add the same report to a scheduled activity more times. You can use the icon  to easily duplicate a document. Once all desired documents have been

NEW
SAVE
CLOSE

DETAIL

Document scheduling test
24/120

DOCUMENTS
+

TESTBIRT

tree

How do you want to assign values to the parameter?
Assign fixed values now

Role
/ knowage /admin

Values

Don't iterate for each value

outputType

How do you want to assign values to the parameter?
Assign fixed values now

Fig. 3.26: Manage parameters.

added and the management configuration of their parameters has been set up, save the activity by clicking on the save button. The new activity is shown in the list and can be modified or deleted using intended specifically icons.

You can manage your activity at any time just clicking on the name of the scheduling item (left side of the window) and all its features will be displayed aside (right half part of the window).

To see and modify the list of the schedules associated to an activity, click on the “Plus” icon you find in the scheduling area in the bottom right side. Similarly, click on the same button to associate schedules to newly created activities.

Scheduling details opens (Figure below).

Fig. 3.27: Scheduling detail panel.

It is composed by two areas: **Detail** and **Documents management**. The details tab let you set all properties of you scheduling. Indeed here you decide its name, description and timing. A scheduling can be made in a chosen date and time if you choose **Single Execution** as type. Otherwise it can be realized for fixed periods or start on a fixed event.

You can decide it, by choosing the scheduling type.

Available scheduling type are:

- Single Execution;
- Customized Execution;
- Event Execution.

A **Single Execution** is a fixed execution for a date and a time which happens only once. A **Customized Execution** repeats your scheduling periodically according to your settings. The **Event Execution** starts the scheduling when an event happens.

You can choose among these event types:

- REST service,
- JMS Queue,
- ContextBroker message,
- Dataset verification.

If you choose **Dataset verification**, you have to select a right structured dataset. It has to give as results only true or false. Then set the frequency in seconds. This is the frequency the dataset will be verified. For example if you set it on 10 seconds it means that each 10 seconds the dataset is executed. If the result of its execution is true, the scheduling is triggered otherwise it isn't.

Once you are done, switch to the **Document Management** tab.

Fig. 3.28: Document management.

Here you can find the dispatch configurations, that can be different for all the documents that compose the scheduled activity. All documents that compose the activity have their own dispatch configuration and the same document can be distributed along multiple dispatch channels. You can switch among the documents included in your activity by clicking on their name in the upper toolbar. There are many different possible dispatch channels that can be used to distribute the results of the execution of a scheduled activity:

- Save as snapshot,
- Save as file,
- Save as document,
- Send to Java class,
- Send mail

In the following sections we explain them in detail.

3.7.1.1 Save as snapshot

The executed document can be saved as snapshots in cyclic buffers of configurable size. For example, it is possible to store in the buffer the last 12 snapshots (the **History Length** field) of one report, scheduled to be executed one per month, in order to have a one-year long history.

The list of all snapshots contained in the buffer can be accessed from the **Show scheduled executions** contained in the **Shortcuts** menu. You can find it in the document toolbar at the top right corner. Each snapshot can be opened or deleted from this panel. These steps are shown in the following figure. A snapshot contains data queried from the database at the moment of its execution performed by the scheduler.

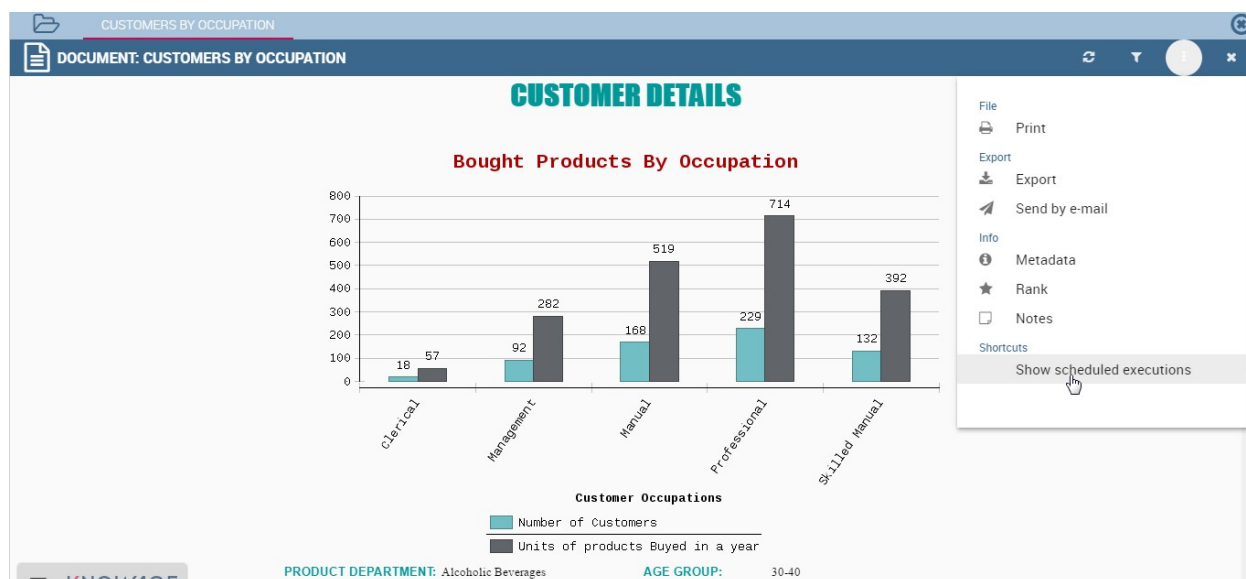


Fig. 3.29: Steps to open saved snapshots

3.7.1.2 Save as file

The executed document can be saved as file on the filesystem in the path /knowage-<version>/resources (if no destination folder is specified). Otherwise, you can create the relative path of this subfolder by writing your subfolder name. For instance, if you write “MyFirstScheduler” as file name and “Schedulation” as destination folder, after the schedulation execution a subfolder Schedulation containing the file “MyFirstScheduler” is created in /knowage-<version>/resources. If the subfolder Schedulation already exist your file is added to this subfolder. You can have a look at the form in Figure below.

The screenshot shows the 'SCHEDULATION DETAIL' form. The 'DETAIL' tab is active, and the 'DOCUMENTS MANAGEMENT' section is expanded. Under 'TestBirt', there are two options: 'SAVE AS SNAPSHOT' (unchecked) and 'SAVE AS FILE' (checked). Below these options are two text input fields: 'File name:' and 'Destination folder:', both with a 0/100 character limit. At the bottom left, there is a checkbox for 'Save as Zip file' which is currently unchecked. At the bottom right, there are 'CANCEL' and 'SAVE' buttons.

Fig. 3.30: Save as File form.

If you prefer to generate a .zip file containing the scheduled documents, you can check the dedicated mark.

3.7.1.3 Save as document

The executed document can be saved as an **Ad hoc reporting** document in the Knowage functionality tree. The document execution will be saved in the specified folder and will be visible to all users that can access that particular folder. For those documents whose execution is iterated over a parameter value, it is also possible to use the value of the parameter to decide to which folder the document shall be dispatched. To do so, define a mapping dataset composed of two columns:

- the first containing a specific parameter value;
- the second containing the label of the folder where the document shall be dispatched when the document is executed with the corresponding parameter value.

Once you have defined the mapping dataset, you can use it in the configuration settings of the document dispatcher. Like in the previous case, the scheduler will execute the report one time for each possible value of the parameter. This time, however, execution results will be dispatched in different folders, according to the mapping defined in the dataset.

3.7.1.4 Send to Java class

The executed document can be sent to a Java class implementing a custom dispatch logic. The custom class must extend the abstract class `JavaClassDestination` that implements the method `execute`. This method is called by the scheduler after document execution. Below an example of Java class.

Listing 3.9: Java Class Code Example.

```

1 package it.eng.spagobi.tools;
2 import it.eng.spagobi.analitichalmodel.document.bo.BIObjct
3 public abstract class JavaClassDestination
4 implements IJavaClassDestination {
5     BIObjct biObj=null;
6     byte[] documentByte=null;
7     public abstract void execute();
8     public byte[] getDocumentByte() {
9         return documentByte;
10    } public void setDocumentByte(byte[] documentByte) {
11        this.documentByte = documentByte;
12    }
13    public BIObjct getBiObj() {
14        return biObj;
15    }
16    public void setBiObj(BIObjct biObj) {
17        this.biObj = biObj;
18    }
19 }
```

The method `getDocumentByte` can be used to get the executed document, while the method `getBiObj` can be used to get all metadata related to the executed document. The following code snippet shows an example of a possible extension of class `JavaClassDestination`.

Listing 3.10: JavaClassDestination example.

```

1 public class FileDestination extends JavaClassDestination {
2     public static final String OUTPUT_FILE_DIR = "D:\\ScheduledRpts\\";
3     public static final String OUTPUT_FILE_NAME = "output.dat";
4     private static transient Logger logger = Logger.getLogger(FileDestination.class);
5     public void execute() {
6         File outputDir;
7         File outputFile;
8         OutputStream out;
```

(continues on next page)

(continued from previous page)

```

9  byte[] content = this.getDocumentByte();
10 String outputFileName;
11 logger.debug("IN");
12 outputFile = null;
13 out = null;
14 try {
15     outputFileName = getFileName();
16     logger.debug("Output dir [" + OUTPUT_FILE_DIR + "]");
17     logger.debug("Output filename [" + outputFileName + "]");
18     outputDir = new File(OUTPUT_FILE_DIR);
19     outputFile = new File(outputDir, outputFileName);
20     if(!outputDir.exists()) {
21         logger.debug("Creating output dir [" + OUTPUT_FILE_DIR + "] ...");
22         if(outputDir.mkdirs()) {
23             logger.debug("Output dir [" + OUTPUT_FILE_DIR + "] succesfully created");
24         } else {
25             throw new SpagoBIRuntimeException( "Impossible to create outputd dir
26 [" + OUTPUT_FILE_DIR + "]");
27         }
28     } else {
29         if(!outputDir.isDirectory()) {
30             throw new SpagoBIRuntimeException( "Outputd dir [" + OUTPUT_FILE_DIR + "]
31 is not a valid directory");
32         }
33     }
34     try {
35         out = new BufferedOutputStream( new FileOutputStream(outputFile));
36     } catch (FileNotFoundException e) {
37         throw new SpagoBIRuntimeException(
38 "Impossible to open a byte stream to file
39 [" + outputFile.getName() + "]", e);
40     } try {
41         out.write(content);
42     } catch (IOException e) {
43         throw new SpagoBIRuntimeException( "Impossible to write on file
44 [" + outputFile.getName() + "]", e);
45     }
46     } catch(Throwable t) {
47         throw new SpagoBIRuntimeException( "An unexpected error occurs while saving
48 document" + " to file [" + outputFile.getName() + "]", t);
49     } finally {
50         if(out != null) {
51             try {
52                 out.flush(); out.close();
53             } catch (IOException e) {
54                 throw new SpagoBIRuntimeException( "Impossible to properly close file
55 [" + outputFile.getName() + "]", e);
56             }
57         }
58         logger.debug("OUT");
59     }
60 }
61 private String getFileName() {
62     String filename = "";
63     BIOject analyticalDoc;
64     List analyticalDrivers;
65     BIOjectParameter analyticalDriver;
66     String extension = ".pdf";
67     analyticalDoc = getBiObj();
68     analyticalDrivers = analyticalDoc.getBiObjectParameters();
69     for(int i = 0; i < analyticalDrivers.size(); i++) {
70         analyticalDriver = (BIOjectParameter)analyticalDrivers.get(i);
71         String parameterUrlName = analyticalDriver.getParameterUrlName();
72         List values = analyticalDriver.getParameterValues();
73         if(!parameterUrlName.equalsIgnoreCase("outputType")){
74             filename += values.get(0);

```

(continues on next page)

(continued from previous page)

```

75 } else {
76 extension = "" + values.get(0);
77 }
78 }
79 filename = filename.replaceAll("[^a-zA-Z0-9]", "_");
80 filename += "." + extension;
81 return filename;
82 }
83 }

```

The class `FileDestination` copies the executed documents to the local filesystem in a folder named `D:\textbackslashScheduledRpts`. The name of the report file is generated concatenating all the parameter values used by the scheduler during execution. Once implemented and properly compiled, the Java class must be exposed to the classpath of Knowage web application. For example, you can pack the compiled class into a .jar file, copy it into the lib folder of Knowage web application and restart the server. As a last step, it is necessary to assign the fully qualified name of the new class, e.g., `it.eng.spagobi.tools.DestinationFile`, to the configuration property classpath.

3.7.1.5 Send mail

Important: Enterprise Edition only

This feature is available only with KnowageER and KnowageSI, submodules of Knowage Enterprise Edition

The executed document can be sent to one or more mail recipients. The list of mail addresses to be used to forward the executed document can be defined in three different ways:

- statically;
- dynamically, using a mapping dataset;
- dynamically, using a script.

In Figure below you can have a look at the mail form. In the following we will focus on each typology, clicking on the info icon you get detailed information.

3.7.1.5.1 Static list

If you want to choose a static list, check the option **Fixed list of recipients** and fill the configuration property **Mail to** with the list of desired mail addresses separated by a comma. An mail for each executed document will be sent to all the mail addresses contained in the list.

3.7.1.5.2 Dynamic list with mapping dataset

In this case, you have to define a two-column dataset:

- the first containing a specific parameter value;
- the second containing each mail address the executed document should be dispatched to.

You can see an example of dataset in the following Figure.

Example of mapping dataset for dynamic distribution list

Basically, when the parameter has a given value, the document will be sent to the corresponding email address. Once you have defined the mapping dataset, you can use it in the configuration settings of the document dispatcher. With

SCHEDULATION DETAIL

✓
SEND MAIL

☒ Fixed list of recipients i

Mail to: 0/100

☐ Use a DataSet as recipients' list i

☐ Use an expression i

☐ Send unique mail for all scheduled document (use this tab mail settings)

☐ Send zipped file?

☐ Include report name (and timestamp) in mail subject.

CANCEL
SAVE

Fig. 3.31: Sending mail form.

	parameter_value	mail_address	
1	President	namesurname@gmail.com	
2	VP Country Manager	name1surname1@gmail.com	
3	VP Information System	name1surname1@gmail.com	
4	VP Human Resources	name1surname1@gmail.com	
5	VP France	name1surname1@gmail.com	
6	HQ Information System	name2surname2@gmail.com	
7	HQ Marketing	name2surname2@gmail.com	
8	HQ Human Resources	name2surname2@gmail.com	
9	HQ Finance and Account	name2surname2@gmail.com	

this configuration, the scheduler will execute the report one time for each possible value of the parameter **Position**, then dispatching the results to different recipients. Specifically, all execution results passing a value of the **Position** parameter to the report starting with VP will be sent to `name1surname1@gmail.com`, the ones starting with HQ will be sent to `name2surname2@gmail.com` and the ones starting with President will be sent to `namesurname@gmail.com`.

3.7.1.5.3 Dynamic List with script

Check the option **Use an expression** and assign a value to the configuration property **Expression** with a parameter-dependent expression like the following:

```
$P{dealer}@eng.it
```

Here dealer is a document parameter label (`$P{dealer}` will be replaced by the parameter value of the scheduled execution).

3.7.2 Scheduling panel

To conclude our overview on the scheduler features, save your settings and go back to the main scheduler page.

Here you can select one of the available scheduled activities to explore details.

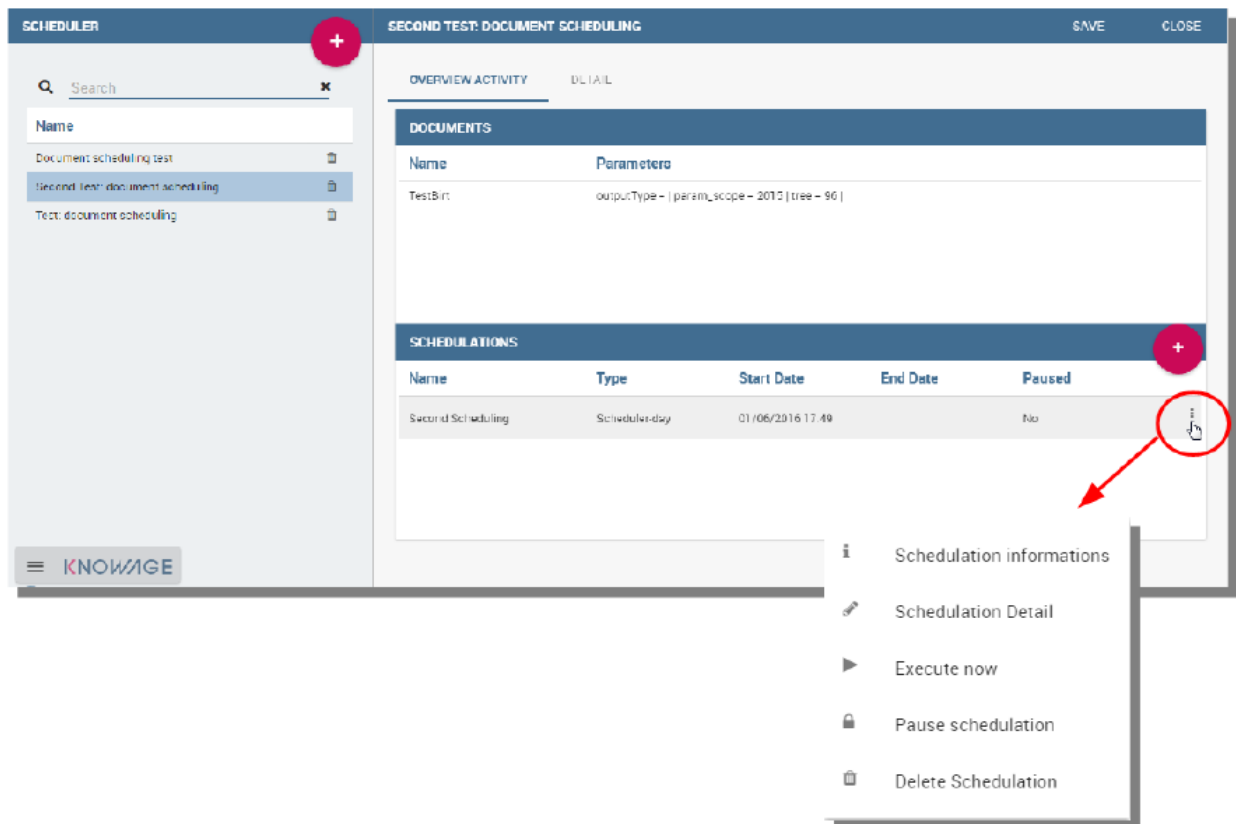


Fig. 3.32: Exploring the detailed of a scheduled activity.

A general overview of the selected scheduling is given in the right side of the page. You can inspect two tabs: **Overview activity** and **Detail**. In the Overview activity tab the main details of the scheduling are displayed summed

up. Namely it is showed the documents involved, the related parameters and their eventually default values, what kind of scheduling has been chosen (Single Execution, Customized Execution or Event Execution), the start date and so on. Note that at the end of the row you have the possibilities to explore more details by clicking on the “three dots” icon.

Here you find the following information:

- **Scheduling informations**, it give some extra information about your schedulation concerning sending emails
- **Scheduling detail**, it opens the scheduling configuration and let you change them.

SCHEDULATION INFORMATIONS

TESTBIRT

Mail to:

Attached zip name:

Mail Subject:

Contained File Name:

Mail Text:

CLOSE

Fig. 3.33: Scheduling information pop up example

- **Execute now**, by clicking it you immediately start the execution of your schedulation.
- **Pause schedulation**, it lets you pause your schedulation.
- **Resume schedulation**, it appears after having paused a schedulation, it enables you to resume it.
- **Delete Scheduling**, it lets you delete a schedulation.

In the **Detail** tab you can analyze the settings on document, that is which parameters are associated to it and how to manage them.

3.7.3 Scheduler Monitor

You can monitor the whole scheduling situation by entering the **Scheduler Monitor** item from the Knowage Menu. This feature allows you to check which schedulations are active in a certain future time interval and, eventually, to be redirected to the schedulation area in order to modify the selected schedulation.

3.8 Server Manager

Important: Enterprise Edition only

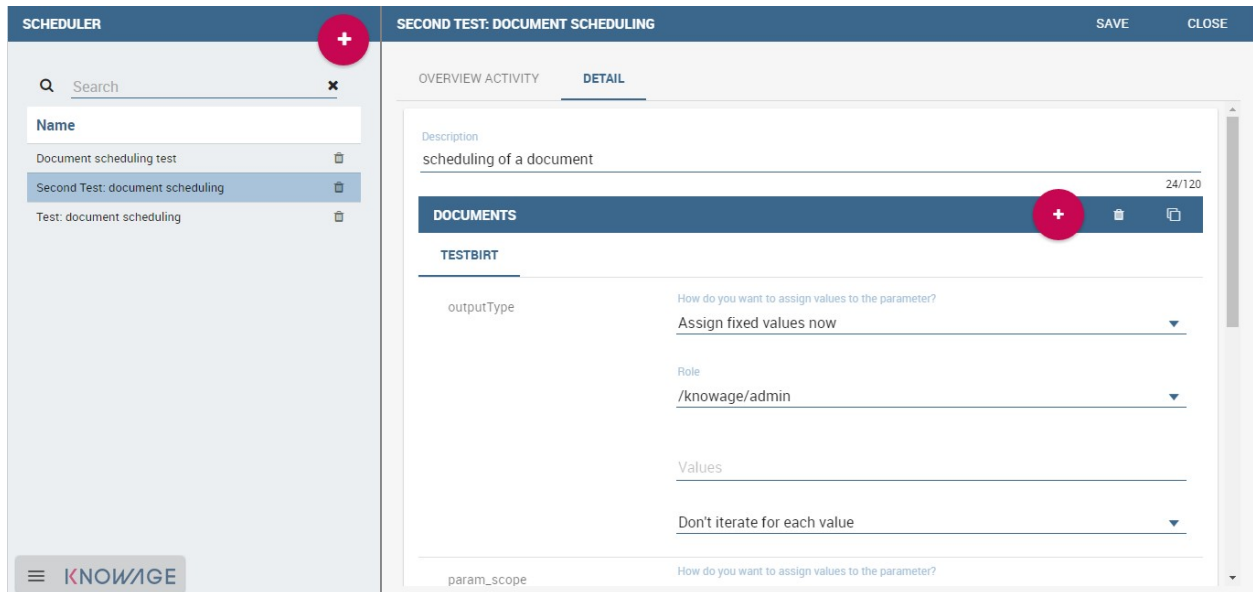


Fig. 3.34: Scheduling detail tab

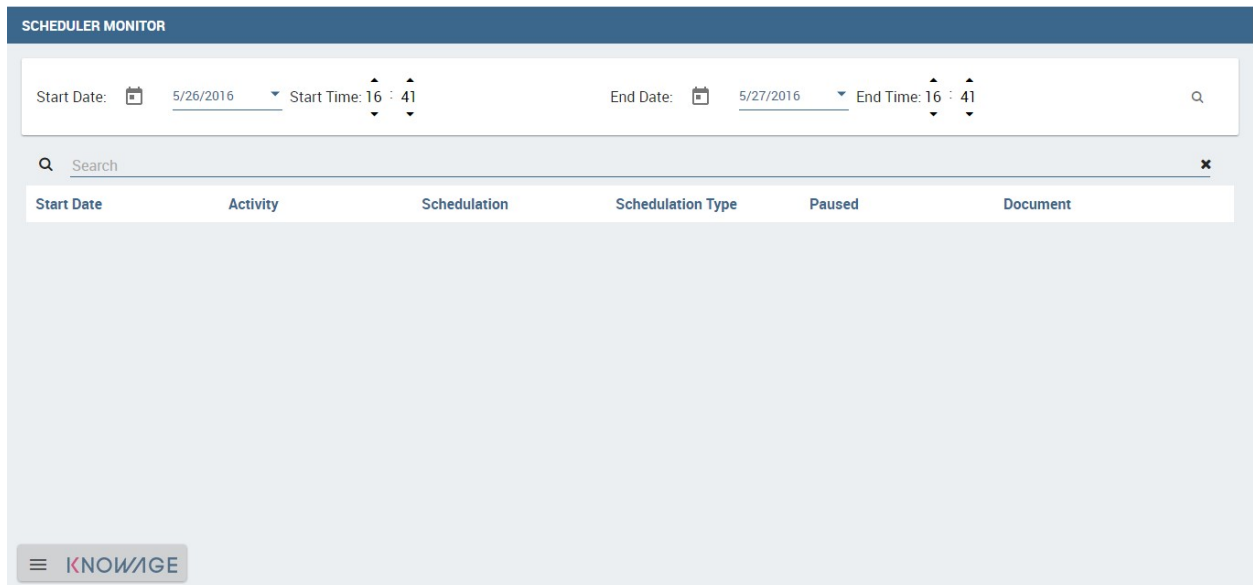


Fig. 3.35: Scheduling detail tab

All the functionalities shipped within the Server Manager are available only with Knowage Enterprise Edition

In the **Server Manager** menu panel you find some management functionalities.

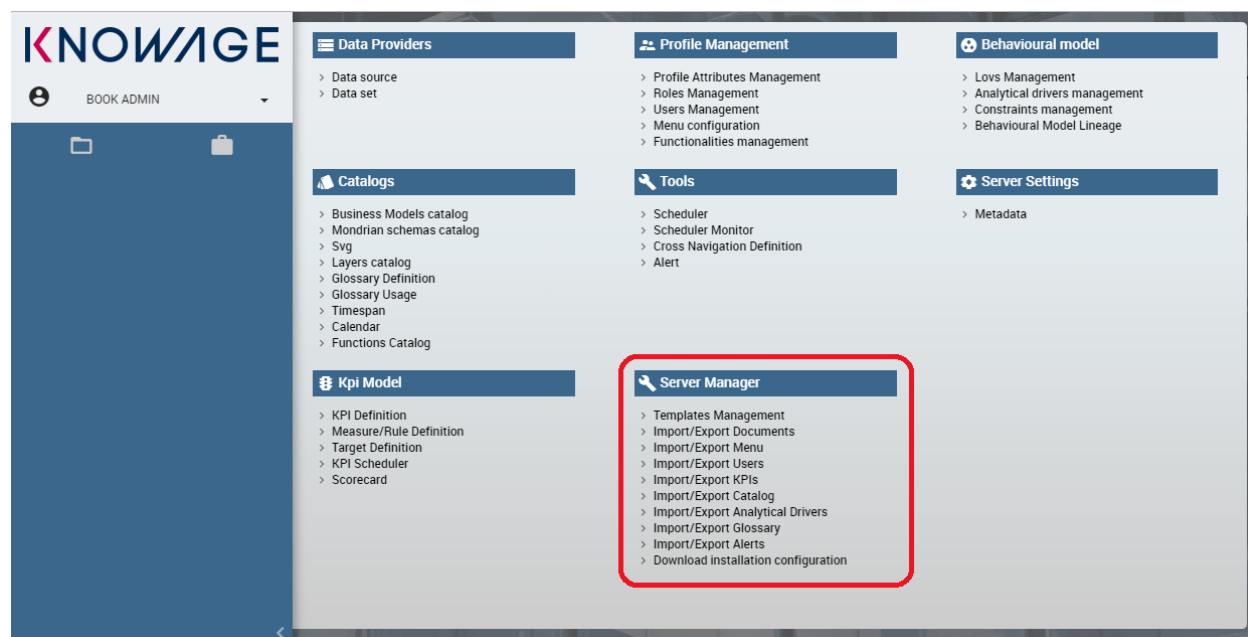


Fig. 3.36: Server Manager Panel

Those about **Import/Export** let you export some configurations or elements from one installation to another. This can be helpful for example in managing a test and a production areas. We are going to give the full description of these functionalities in the following.

3.8.1 Tenants Management

We start this section underlining that only those users who have the superadmin role can use this functionality. **Tenants Management** is available only for users who possess the Knowage Enterprise Reporting (ER) license. A **tenant** is generally a user who can or cannot employ specific product types or access some (or all) datasources inside the same environment. Then, this functionality allows you to create new tenants or manage old ones.

In the image above, on the left you have the list of existing tenants. On the top of such list it is available the **Search** box to help users to browse the tenants. When clicking on the “Plus” icon you can create a new tenant. A form opens on the right area. Insert a **Name** and a **Theme**. Then change tabs to set product types access and select which datasources are achievable.

Note that, in a single-tenant environment admin and superadmin coincides. In a multitenants environment (developed then through the Server Manager functionality), only *one* user has the superadmin role for each tenant, while there can be one or more users with admin role. In particular compared to the admin case, the superadmin has the possibility to manage the multi-tenancy. Moreover, he is the only one who can configure the JNDI datasources and access the cache configuration (through the cache manager menu item).

3.8.2 Template Management

Each Knowage document is associated to a *template*. The template defines the standard layout of a document, including specific information on its appearance and the way contents should be displayed. Templates can be encoded by hand or

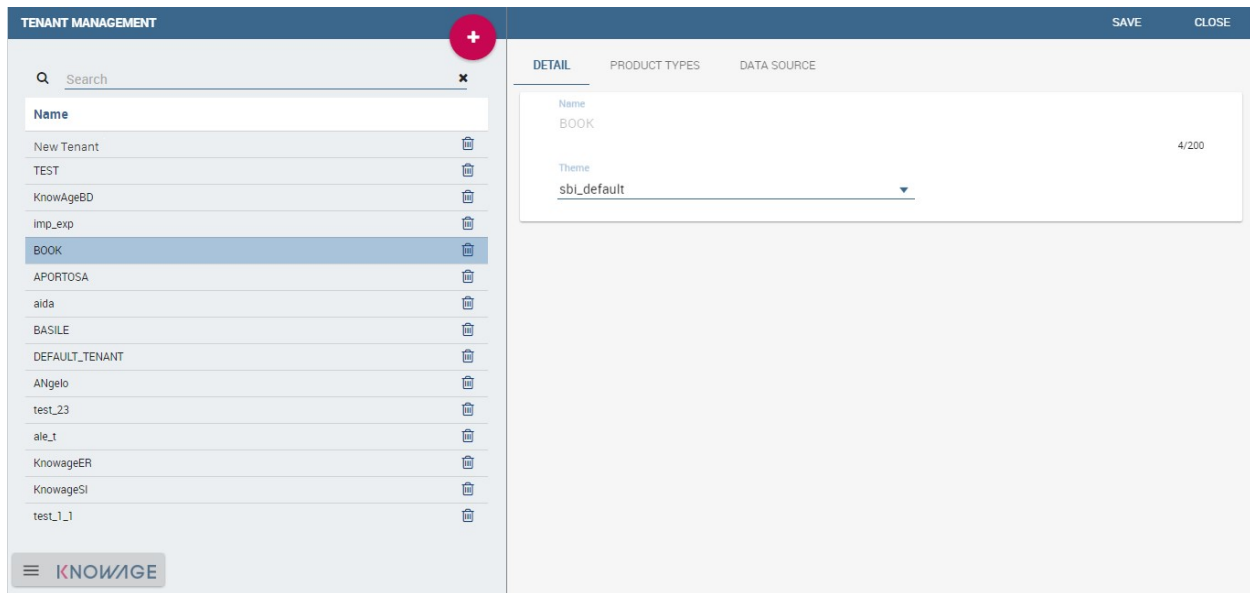


Fig. 3.37: Tenants Management window.

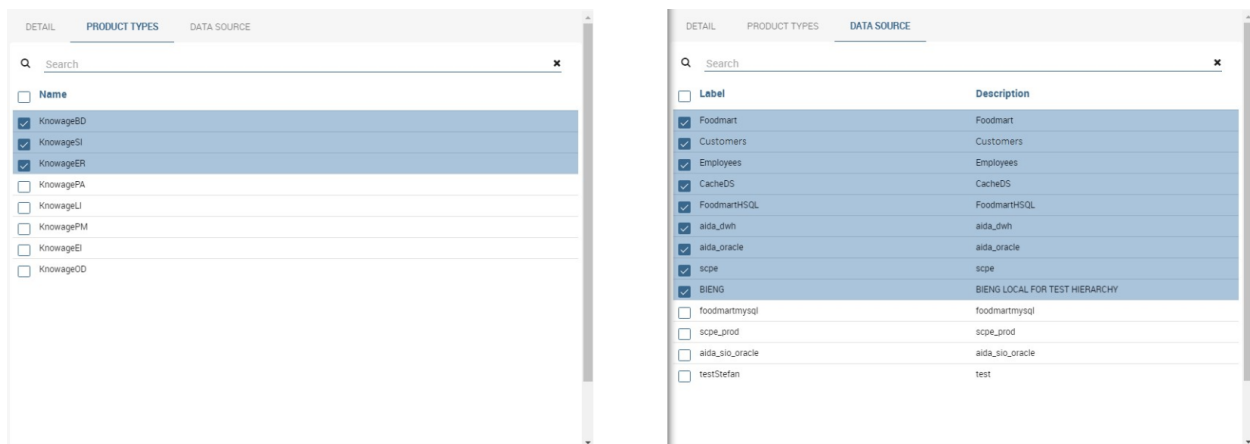


Fig. 3.38: Product types tab (Left) Datasources tab (Right).

using Knowage Studio designers, when available. For each analytical document the history of templates is maintained. Old templates can be restored if needed. A new version is saved at each deployment, either manual or from Knowage Studio.

The **Template Management** let you choose a starting date before which delete the templates versions. This could be very useful because it allows the administrator to clean the environment and save space in Knowage metadata database once a document life cycle is completed.

First of all you are asked to insert a date by clicking on the calendar icon. Then click the magnifier icon and select the documents you are interested in. The list displayed contains only documents created before the selected date. Clicking the trash icon you delete the template of the selected documents which were uploaded before the chosen date. If all the templates of a document precede the chosen date, the last template uploaded will be kept, so that no document is deleted accidentally. We sum up the steps described in Figure below.

3.8.3 Import/Export

These options are about Import\Export of Documents, Menu, Users, KPIs and Catalogs. Let's focus on each of these features.

3.8.3.1 Documents

This feature let you create and download a .zip of whole or a part of the documents existing in your Knowage installation. In this way you can upload it in another installation or keep it as backup.

When you import, all the "objects" associated to those documents (such as datasets, lovs, drivers, roles and folders) are created. Instead users, menu configurations, KPI, catalog, glossary and alert are not exported with this tool.

Let's have a look on the steps to create the .zip.

Below we show the export editor.

First of all choose the name to give to your exportation (i.e. if you choose MyFirstExport, you will create the MyFirstExport.zip).

Then select which documents do you want to export. You can browse the folder by clicking the folder icon. Choose the elements or folders you want to include by marking the related checkbox. A check in a parent folder will automatically select/deselect all its children folders/leaves.

When you have chosen a name and selected some documents, the export icon change colour from grey to pink. This means all elements are set to start exporting.

Before going on decide if you want to export **Olap customized View** and/or **Scheduled documents** and/or **BIRT Translation** and/or **Schedule configurations** and/or **Document only in the selected functionality**.

- **Olap customized View** Checking this property the export will include all the customized views saved into the chosen OLAP documents. You can find the Customized View going on the OLAP Document Menu and clicking on *Show OLAP custom View*. See the figure below:
- **Scheduled documents** Checking this property the export will include all the scheduled execution saved into the chosen documents. You can find the scheduled execution going on the Document Menu and clicking on *Show Scheduled Execution*. See the figure below:
- **BIRT Translation** Checking this property the export will include all the translation added into 'Localization' functionalities of the BIRT templates.
- **Schedule configurations** Checking this property the export will include all the schedulation associated to the chosen documents. At the end of the import you must have the schedulation saved into Scheduler section under the Tools area.

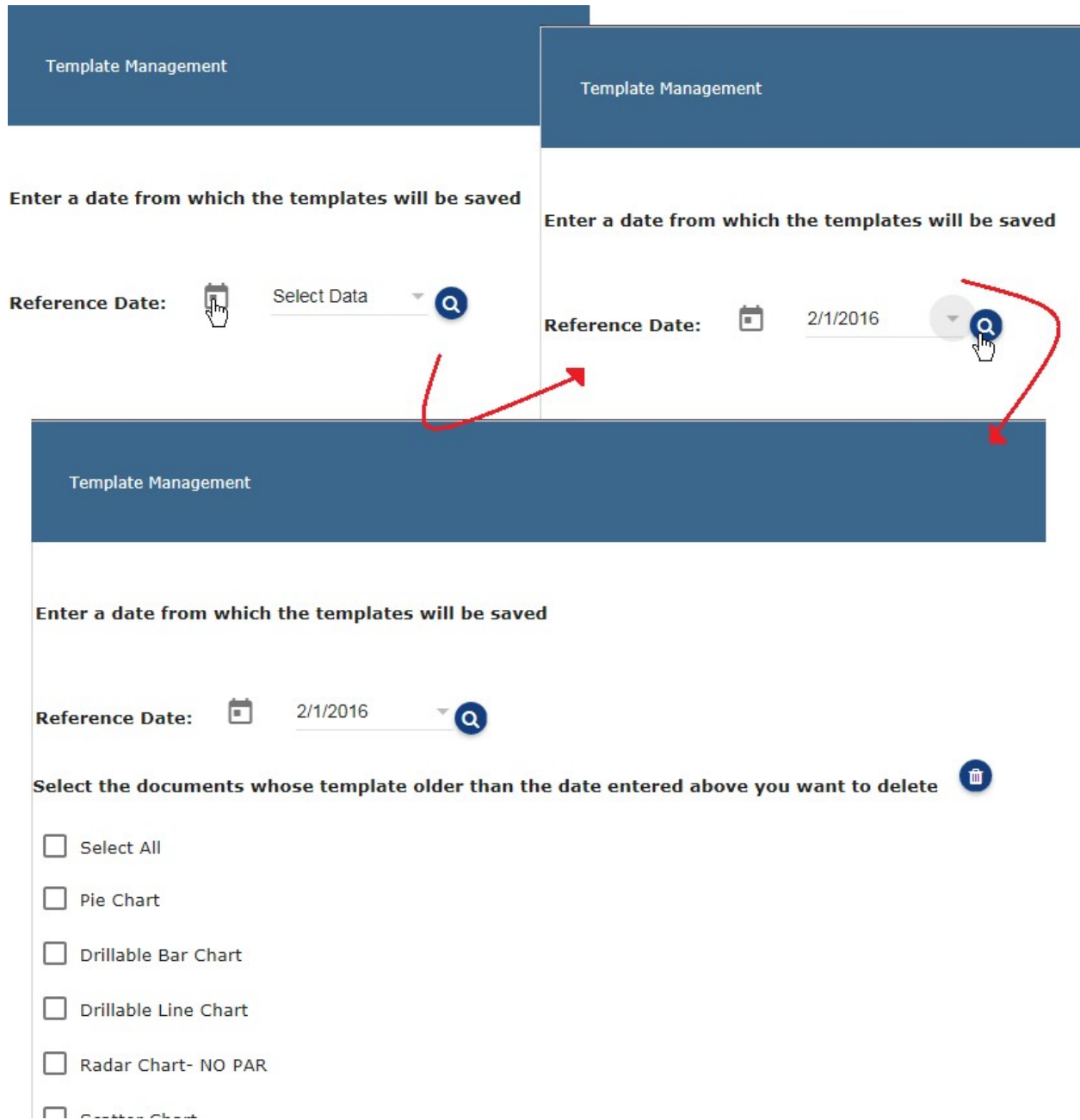


Fig. 3.39: Deleting templates

Import/Export Documents

EXPORT IMPORT

Name Export *

☐ Export Olap customized View
 ☐ Export Scheduled documents
 ☐ Export BIRT translations
 ☐ Export Schedule configurations
 ☐ Export the document only in the selected functionality

Filter Documents After Date:
 Filter by Status: ☐ Development ☐ Test ☐ Released

☐ Functionalities
 ☐ Analytical Engines
 ☐ Cockpit
 ☐ Report
 ☐ Olap
 ☐ CROSS_OLAP_REPORT (REL)

Fig. 3.40: Document Export

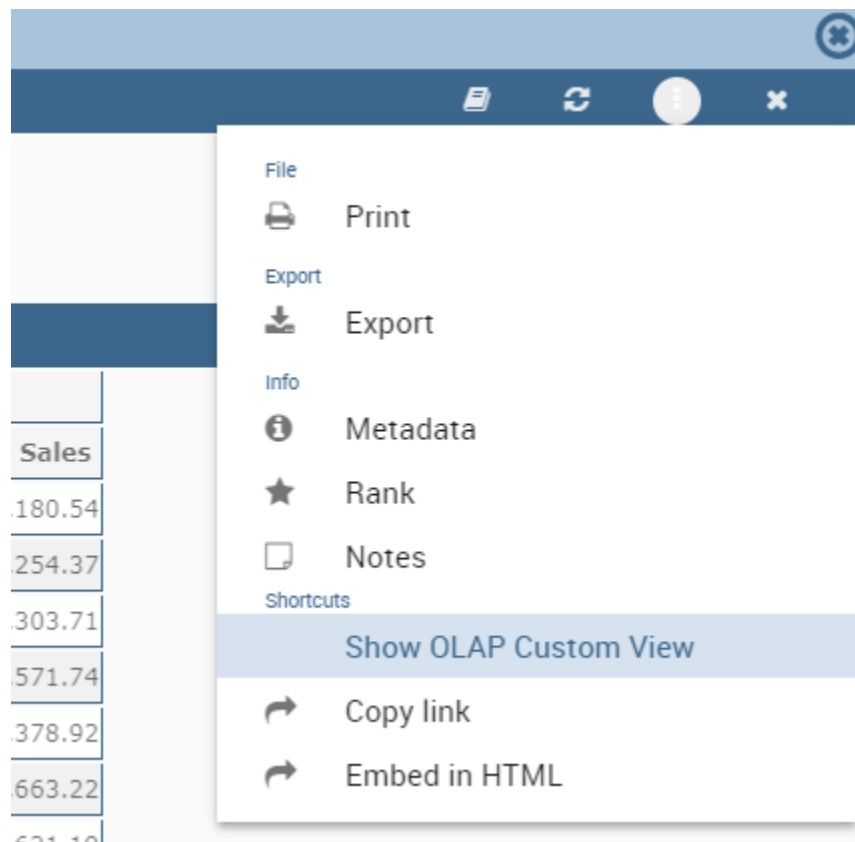


Fig. 3.41: Olap customized view

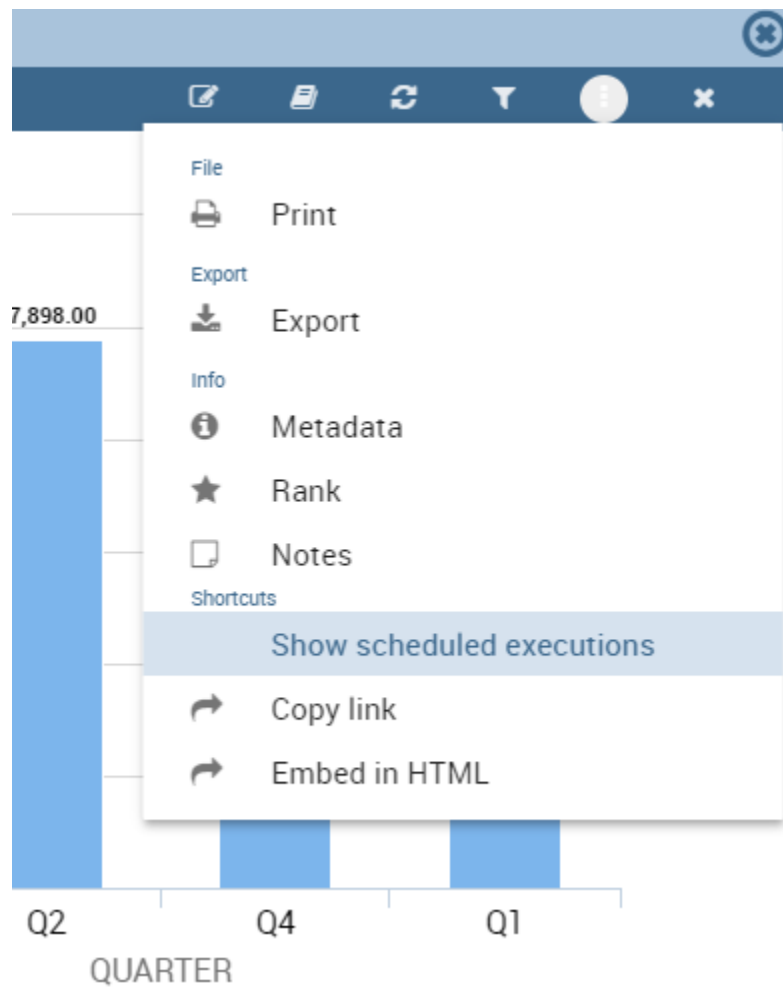


Fig. 3.42: Scheduled documents

- **Export the document only in the selected functionality** Checking this property the export will include documents only if they are inside a selected functionality.

Now you are ready to click on the export icon to generate and download the .zip. Suppose you want to upload MyFirstExport.zip in another installation. Log in it and move to **Server Manager > Import\Export Documents** area. Switch to the **Import** tab and click on **Browse** to accede your personal folders. In Figure below we show the document import interface.

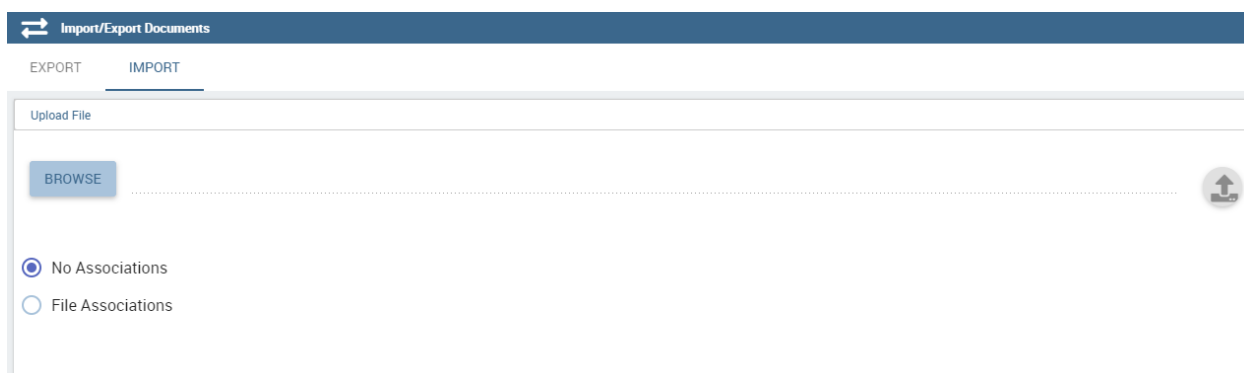


Fig. 3.43: Document Import

Choose the .zip obtained from the **Export** phase and click on the import icon. Few steps guide you through importation. You are asked to map from source to target: Roles, Engines, Datasources and Metadata.



Fig. 3.44: Document Import Wizard

If a role doesn't map any of the existing in the target one, it will be created. **Please pay attention before starting the import because target metadata will be overwritten:** documents, lov, driver, etc. which has the same label of the exported ones will have metadata overwritten at the end of import procedure. After you have carefully checked, you can start import procedure by clicking on "**START IMPORT**"

Missing licenses If one or more license is not valid or missing, during export procedure you'll be informed with an alert. Related documents cannot and won't be exported.

Licenses are checked also during import procedure and, if one or more are missing or not valid, related documents will not be imported. These documents will be listed in the last phase before the beginning of the import procedure and for each one the invalid or missing license will be shown.

3.8.3.2 Menu

This feature let you export the menu structure.

To start the export you need only to insert the Export name. Once inserted the name, the export icon changes color from grey to pink to let you understand all mandatory fields to start the export were filled. Click on this icon and the related .zip is downloaded.

Import/Export Documents

EXPORT
IMPORT

Name Export *
export

☐ Export Olap customized View
☐ Export Scheduled documents
☐ Export BIRT translations
☐ Export Schedule configurations
☐ Export the document only in the selected functionality

Filter Documents After Date:
Filter by Status:
☐ Development
☐ Test
☐ Released

WARNING! ONE OR MORE DOCUMENTS / FUNCTIONALITIES CANNOT BE EXPORTED DUE TO MISSING OR INVALID LICENSES

☐ Functionalities
☒ Functionality
☐ Functionality2
☐ Functionality3 [NOT EXPORTABLE]
☐ DocFunctionality3 (REL) [NOT EXPORTABLE]

Fig. 3.45: Document Export Missing licenses

Import/Export Documents

EXPORT
IMPORT

Upload File
Exported roles
Exported engines
Exported data sources
Metadata

START IMPORT

Documents that NOT will be imported

WARNING! ONE OR MORE DOCUMENTS CANNOT BE IMPORTED DUE TO MISSING OR INVALID LICENSES

Cockpit Cockpit	knowagecockpitengine Cockpit Engine Cockpit Engine
leaf leaf	knowagecockpitengine Cockpit Engine Cockpit Engine

Source
Target

Documents that will be imported

lka lka	lka lka
-------------------	-------------------

Fig. 3.46: Document Import Missing licenses

Import / Export Menu

EXPORT IMPORT

Name Export *
menu_export

SELECT ALL DESELECT ALL

Current database

- Menu1
 - Menu2
 - Menu3
 - Menu4
 - Menu5
 - Menu6

Fig. 3.47: Menu Export

To upload it in another installation, accede to the **Import\Export** Menu area and switch to the tab **import**. Here click on **Browse** to search in your folders the .zip previously created, see the following Figure.

Import / Export Menu

EXPORT IMPORT

Import

BROWSE new_menu.zip

Fig. 3.48: Menu Import

After clicking on “*START IMPORT*” menu items which match with existing ones will be override by the exported ones and menu items which don’t match with the existing ones will be added. You are asked to map roles from source to target. If a role doesn’t map any of the existing in the target one, it will be created.

Import / Export Menu

EXPORT IMPORT

Import Exported menu Exported roles

Source Target

START IMPORT

Fig. 3.49: Menu Import Wizard

In the first step of import procedure you’ll see source-target comparison. In particular source tree contains menu that are in the package you are importing and target tree contains the menu already present in the target system.

3.8.3.3 Users

In this area you can export the users from an installation to another, see the following Figure.

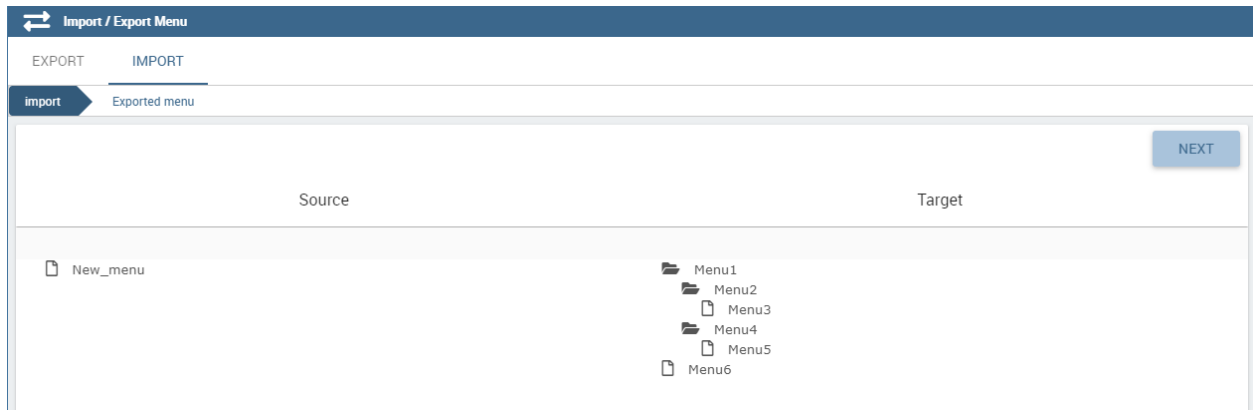


Fig. 3.50: Menu Import Wizard Tree comparison

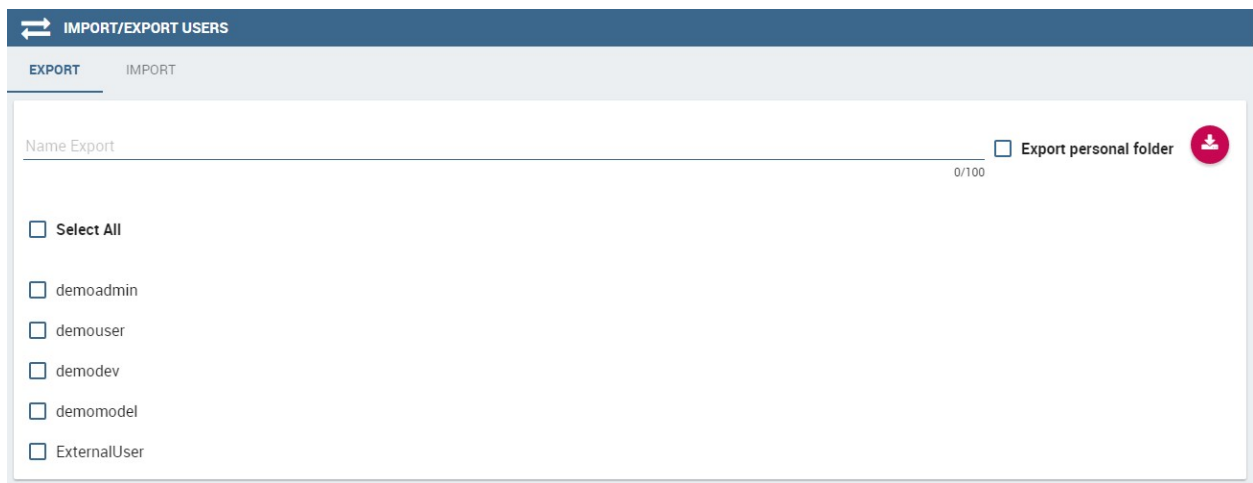


Fig. 3.51: User Export

To generate the .zip you have to mark the user to include in the export and insert an export name. Save the export in the folders of your pc and move to the other installation. You have the chance to include the personal folder of the chosen users in the Export. Put a mark in the **Export Personal folder** checkbox and choose if you want to include snapshots and subviews too.

To import the .zip in another installation, log in and open the **Server Manager > Import\Export Users**, switching to **Import** area. Here click on **Browse** to choose the .zip created by exportation. Then click on the import icon. The users contained in your file are uploaded and Catalogs displayed in the left side of the screen. Choose among the users displayed the one you want to import, mark them and click on the arrow to move them in the other side. Now click on **Start import** button and your users are successfully created in this installation too. Keep attention in marking personal folder checkbox if you want that personal folders are imported. In Figure below you can see **User Import** interface.

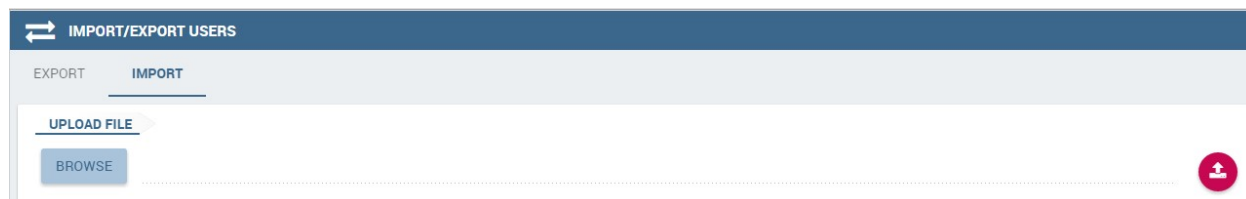


Fig. 3.52: User Import

Important: All users involved into import procedure will have his password changed with the value set in advanced configuration.

3.8.3.4 Catalogs

In this area you can export the different catalogs (such as datasets catalogs, business models catalogs and so on) from one installation to another, see the following Figure.



Fig. 3.53: Catalogs Export

To generate the .zip you have to mark the elements to include in the export and insert an export name. Save the export somewhere in your local system and move to the other installation. You have the chance to include the personal folder of the chosen users in the Export. Put a mark in the **Export Personal folder** checkbox and choose if you want to include snapshots and subviews too.

To import the .zip in another instance, log in and open the **Server Manager > Import\ Export Catalogs**, switching to **Import** area. Here click **Browse** to choose the .zip created through exportation. Uploading the file, the available exported catalogs are displayed in the bottom area. Selecting a catalogs (for instance, the **Dataset** one), all the catalogs

exported elements are displayed in the left side of the screen. Choose the ones that you want to import, decide if you want to override or to just add the missing ones and then click **Start import**. Your catalogs are successfully created in this environment. In Figure below you can see **User Import** interface.

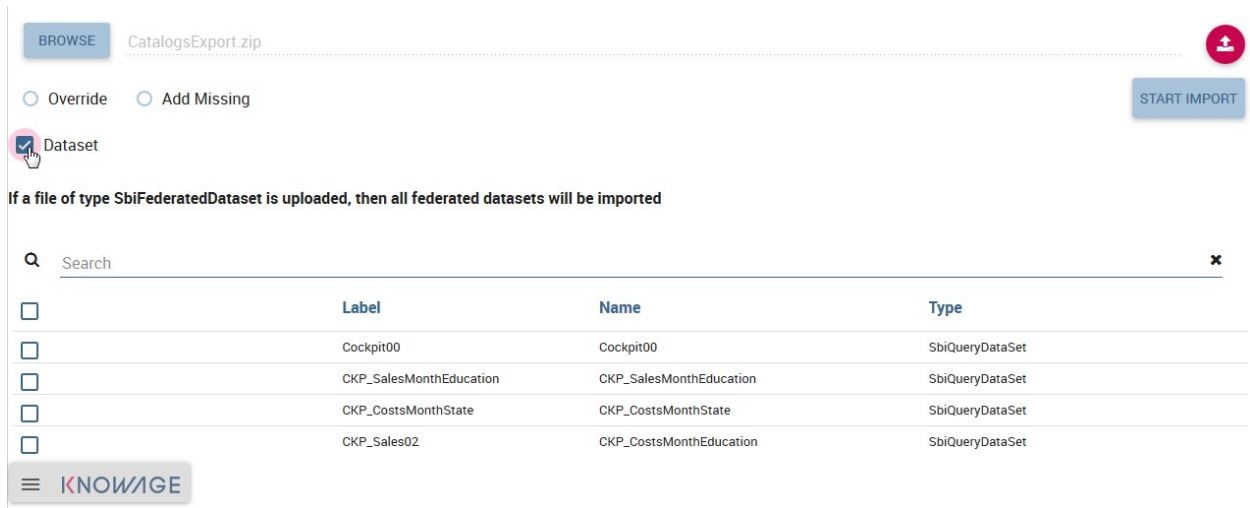


Fig. 3.54: Catalogs Import

3.8.3.5 KPIs

In this section we describe how to manage the import/export of KPIs between two tenants.

The user must enter Knowage as administrator of source tenant and click on **Import/Export KPIs** from Server Manager menu panel.

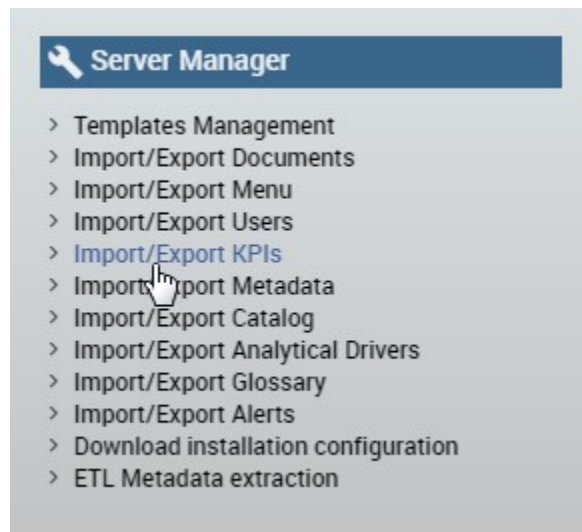


Fig. 3.55: KPIs Import/Export from menu

The page contains the **Export** and the **Import** tab, where the user can select the KPIs for the export/import respectively.

Let's start from the export feature. The user must check the KPIs for the export using the tab interface. He/she can add some more functionalities to the export action, namely:

Fig. 3.56: KPIs Import window

- to include targets,
- to include those scorecards related to the selected KPIs,
- to include schedulations.

Finally click on the red download button to get a zipped folder that will be used to conclude the export.

Fig. 3.57: Start export button

Once the .zip file is downloaded, the user has to switch tenant (the one on which he/she wants to do the import). As admin of the destination tenant, enter the Import/Export KPIs functionality and move to the Import tab.

The user must therefore browse the personal folder to catch the zipped folder and click on the red upload button just aside, as shown in the following figure.

Referring to the following image, the user has to specify if:

- to overwrite the existing KPIs and their related formulas

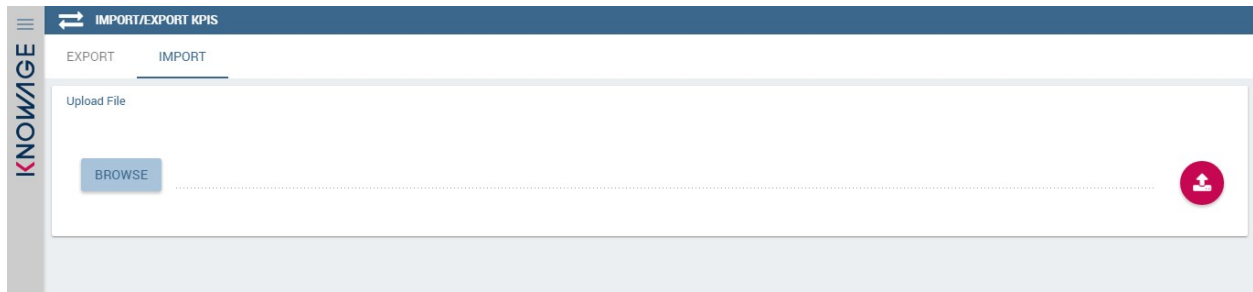


Fig. 3.58: Import tab

- to import targets,
- to import scorecards,
- to import schedulations.

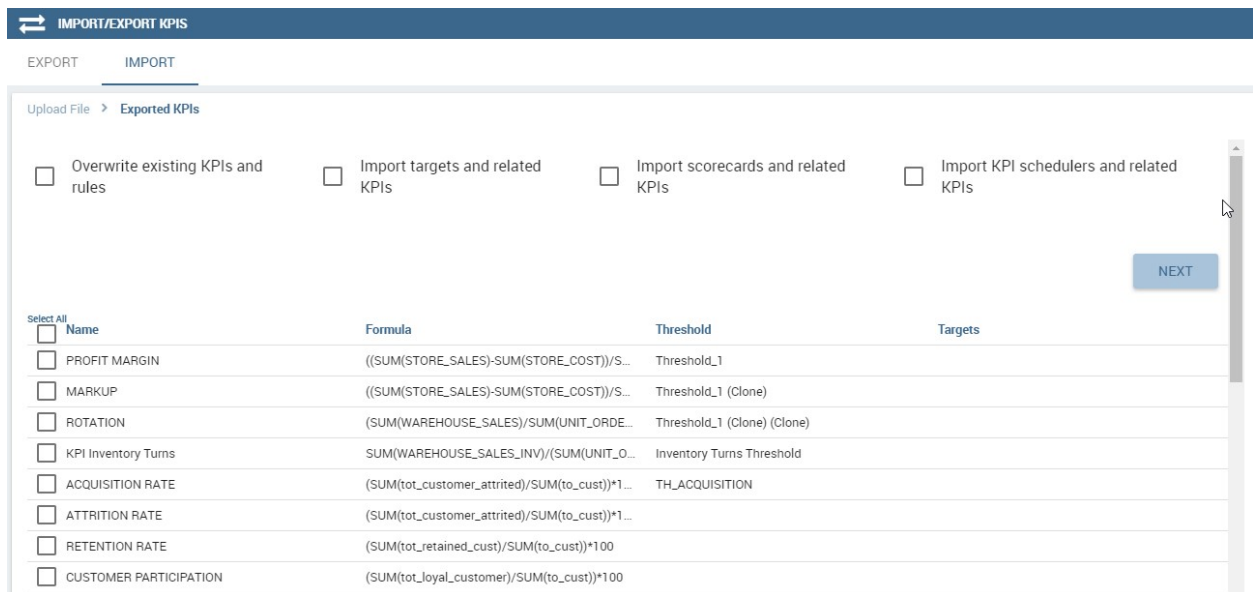


Fig. 3.59: Import KPIS settings

Once the import is started, the GUI leads the user to finalize the import procedure. In particular, the user is asked to map data sources correctly (Figure below).

The process ends successfully when the wizard shows up as following.

3.8.3.6 Analytical Drivers

This option allows to import/export the analytical drivers and their related LOV.

As shown in Figure above, the window contains the Export and the Import tab. Use the Export tab to download the .zip file to be used in the import process.

To produce such a file, the user has to log in as administrator of the source tenant. Then he has to assign a name to the export, check the analytical drivers of interest and click on the red download button available at the top right corner of the page. Note that it is possible to slim down the research of the analytical drivers by filtering on their data of creation.

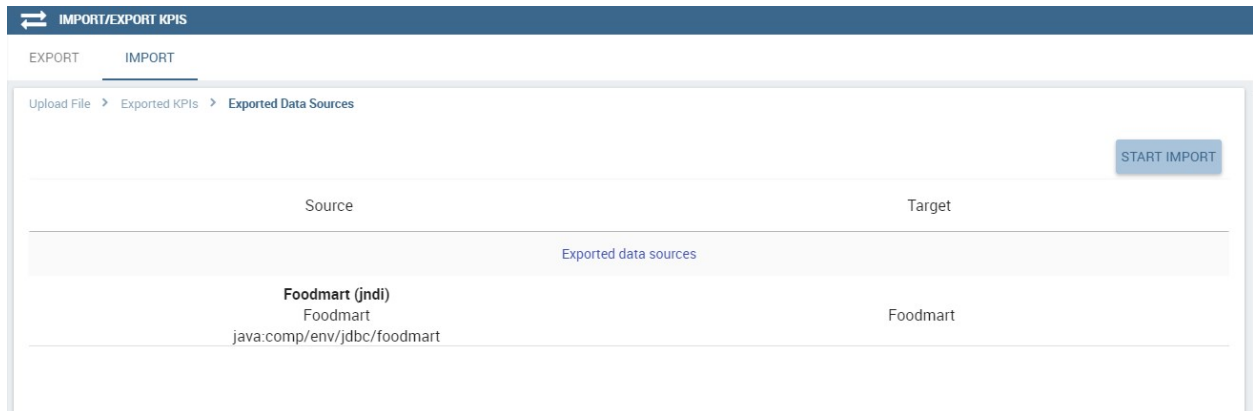


Fig. 3.60: Mapping data sources

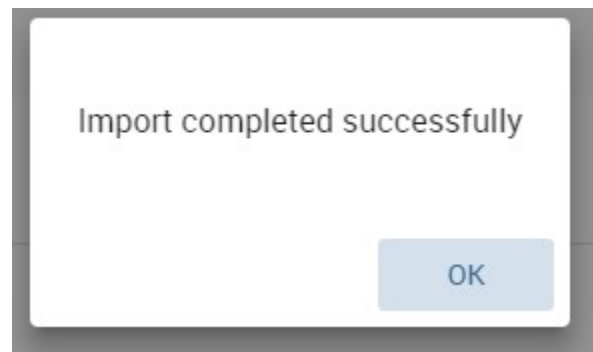


Fig. 3.61: Import KPIS ended successfully

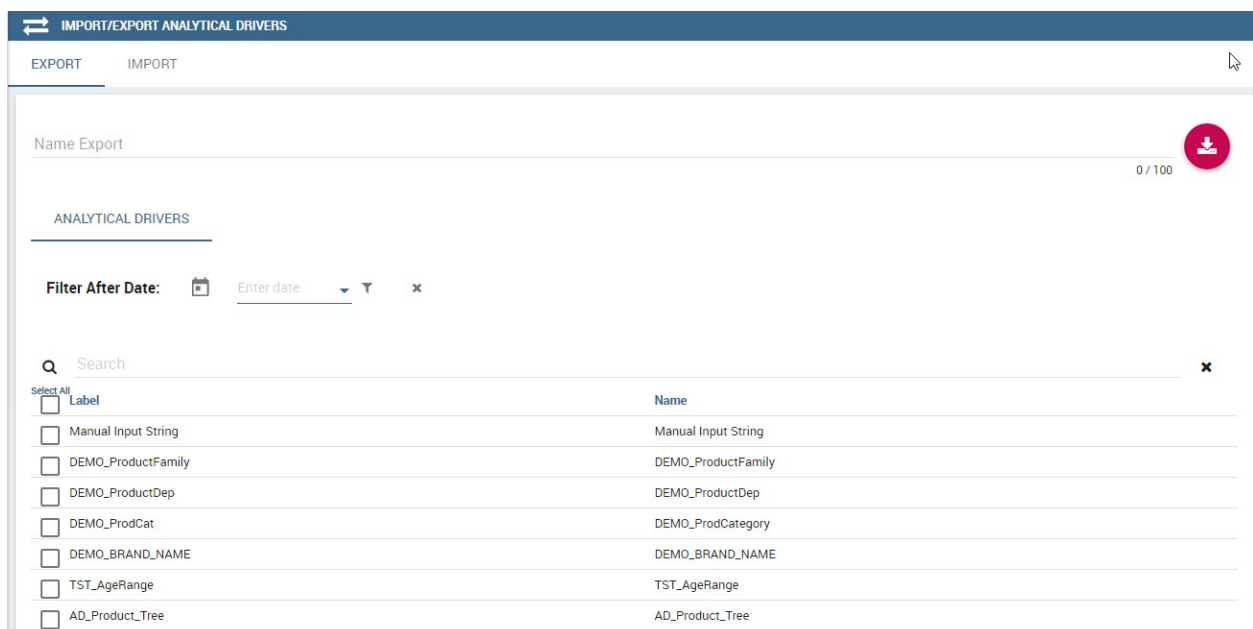


Fig. 3.62: Import/Export of analytical drivers

Switch tenant and log in as administrator. Use the Import tab to upload the zipped folder and finalize the import.

Use the GUI to upload the zipped folder, to specify if to overwrite on the existing analytical drivers or add missing. Then click on next and continue by mapping roles among tenants and data sources.

IMPORT/EXPORT ANALYTICAL DRIVERS

EXPORT IMPORT

Upload File

BROWSE dr.zip

☒ Override ☐ Add Missing

NEXT >

Search

Select All

	Name	Type
<input checked="" type="checkbox"/>	DEMO_ProductDep	

Fig. 3.63: Import of analytical drivers

IMPORT/EXPORT ANALYTICAL DRIVERS

EXPORT IMPORT

Upload File > Exported Roles

NEXT >

Source	Target
/demo/admin	
/demo/user	

Exported roles

Fig. 3.64: Import of analytical drivers

The process ends with a message containing the information about the import.

3.8.3.7 Glossary

The export/import of glossary allows the user to align glossaries among tenants.

There are the two tabs of Export and Import. The user is asked to select the glossaries to export and to type a name that will be assigned to the zipped folder. The user can help himself/herself by using the filter on data (of creation of the glossary).

Once the user has got the zipped folder he/she must switch tenant and enter as its admin. Then select the import tab from the Export/Import main window.

The user must use the arrows to indicate the glossaries he/she wants to import in the target tenant. No further information are needed to end the process. Then the user has to enter the target tenant as administrator and use the import tab to finalize the import.

IMPORT/EXPORT ANALYTICAL DRIVERS

EXPORT **IMPORT**

Upload File > Exported Roles > Exported Data Sources

START IMPORT

Source	Target
Exported data sources	
Foodmart (jndi) Foodmart java:comp/env/jdbc/foodmart	Foodmart

Fig. 3.65: Import of analytical drivers

IMPORT EXPORT GLOSSARY

EXPORT **IMPORT**

Name Export *
Glossary

8 / 100

Filter Glossary After Date:

☐ Select All

☒ MARKET ANALYSIS

Fig. 3.66: Export/Import of glossaries window

IMPORT EXPORT GLOSSARY

EXPORT **IMPORT**

BROWSE Glossary.zip

☐ Override ☒ Add Missing

START IMPORT

Source

Search

Select All ☐

MARKET ANALYSIS ☐

Target

Search

Select All ☐

Navigation buttons: >, <, >>, <<

Fig. 3.67: Import of glossaries

3.8.3.8 Catalog

This functionality allows to Export/Import the following elements

- Data sets,
- Business models,
- Mondrian catalogs,
- Layers,
- SVG files.

The steps to perform the Export/Import are equal to those seen in the previous sections. Namely, the user has to enter the **Import/Export catalog** menu item from Server Manager menu panel. The window will contain the Import and Export tabs. The export tab is used to produce the zip folder to be imported in the tenant of interest. Note that the user can apply a temporal filter to help him/her to look up elements in the list.

Fig. 3.68: Import of catalog

The import requires the zipped folder to be uploaded, to check the elements to import, to map roles among tenants and to map datasources.

3.9 Server Settings

In this chapter we describe all functionalities available in Server Settings panel of the Administrator Menu shown below.

Similar editors give you access to configurations and domains. We are going to provide an example of both cases to let you understand how their management works. A complete overview of metadata creation, editing and management conclude this chapter.

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

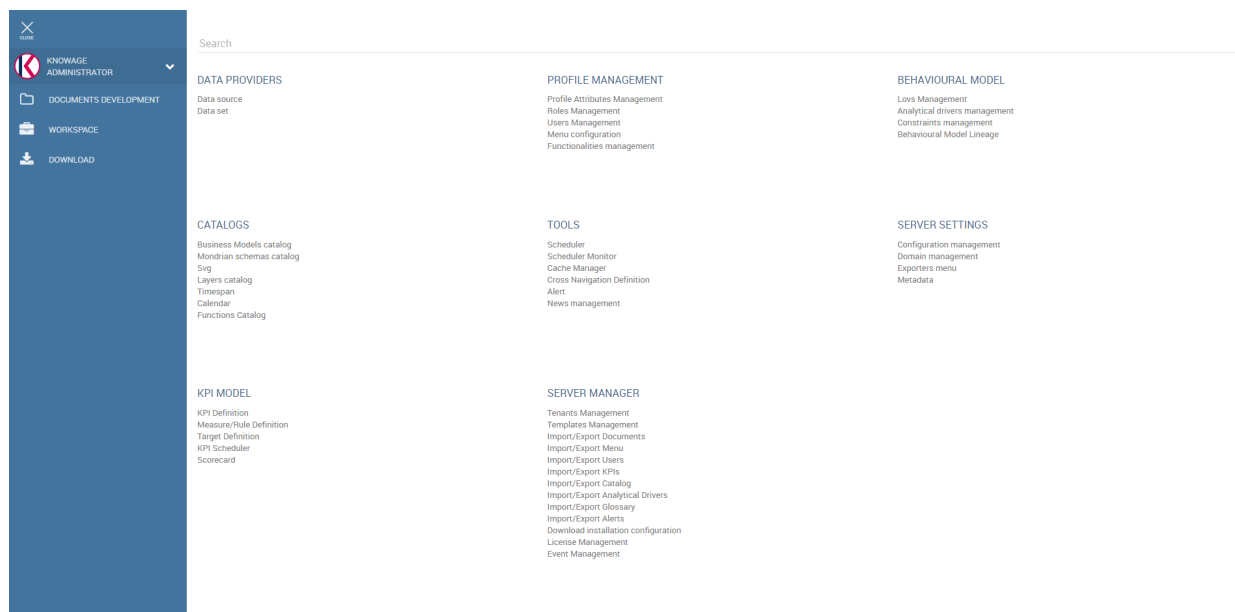


Fig. 3.69: Server Settings Panel.

CONFIGURATION MANAGEMENT			
<div> <div>Q</div> <div>DEFAULT</div> <div>X</div> </div>			
label	name	valueCheck	category
SPAGOBI.SECURITY.DEFAULT_USER	DEFAULT_USER	biadmin	SECURITY
SPAGOBI.THEMES.THEMES	THEMES	sbi_default	THEMES
SPAGOBI.THEMES.THEME.default	THEME default	sbi_default	THEMES
SPAGOBI.THEMES.THEME.sbi_default.view_name	THEME view_name	default	THEMES
SPAGOBI.THEMES.THEME.sbi_default.ext_theme	THEME ext_theme	xtheme-gray.css	THEMES
SPAGOBI.THEMES.THEME.sbi_default.name	default THEME name	sbi_default	THEMES
SPAGOBI.LANGUAGE.SUPPORTED.LANGUAGE.d	default	en,US	LANGUAGE_SUPPORTED
SCRIPT_LANGUAGE_DEFAULT	SCRIPT LANGUAGE DEFAULT	groovy	SCRIPT_LANGUAGE

Fig. 3.70: 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.

Warning: Warning

Knowage Server may take up to 1 minute to apply the configuration changes.

3.9.2 Domain Management

By clicking on **Domains Management** item menu, you can manage categories. In the figure below 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**.

DOMAIN MANAGEMENT						
<div> <div>Q</div> <div>Search</div> <div>X</div> </div>						
valueCd	valueName	domainCode	domainName	valueDescription		
QUERY	sbidomains.nm.query	INPUT_TYPE	Input mode and values	sbidomains.ds.query		
SCRIPT	sbidomains.nm.script	INPUT_TYPE	Input mode and values	sbidomains.ds.script		
FIX_LOV	sbidomains.nm.fix_lov	INPUT_TYPE	Input mode and values	sbidomains.ds.fix_lov		
JAVA_CLASS	sbidomains.nm.java_class	INPUT_TYPE	Input mode and values	sbidomains.ds.java_class		
DATASET	sbidomains.nm.dataset	INPUT_TYPE	Input mode and values	sbidomains.ds.dataset		
REPORT	sbidomains.nm.report	BIOBJ_TYPE	BI Object types	sbidomains.ds.report		
OLAP	sbidomains.nm.olap	BIOBJ_TYPE	BI Object types	sbidomains.ds.olap		
DATA_MINING	sbidomains.nm.data_mining	BIOBJ_TYPE	BI Object types	sbidomains.ds.data_mining		

Fig. 3.71: Domain management editor.

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

<div> <div>Q</div> <div>CATEGORY_TYPE</div> <div>X</div> </div>						
valueCd	valueName	domainCode	domainName	valueDescription		
Cat1	Cat1	CATEGORY_TYPE	Category Type	Cat1		
Cat2	Cat2	CATEGORY_TYPE	Category Type	Cat2		
GEOBI	GEOBI	CATEGORY_TYPE	Category Type	GEOBI		
TECH_DS	Technical data set	CATEGORY_TYPE	Category type	Technical data set		

Fig. 3.72: 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 below. Fill the columns as follow:

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

The form is titled 'NEW' and has 'SAVE' and 'CLOSE' buttons at the top right. It contains the following fields:

- Value code: Costs
- Value name: Costs
- Domain code: CATEGORY_TYPE
- Domain name: Costs Datasets
- Value description: Costs Datasets

Fig. 3.73: Form to be filled to create a new category.

- **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.

3.9.3 Exporters Menu

Exporters Menu is the new feature of Knowage. This feature allows administrator to configure the exporters of the documents. When you open this catalogue, on the left side you can see the list of created exporters and if you want to see the details of one of them or to change it, you can do that by selecting it in the list. You can add a new exporter by clicking on a plus icon. See the figure below.

The interface is titled 'Exporters' and has a search bar at the top left. It contains a table of exporters and a form to add a new one.

Engine name	Exporter
Jasper Report Engine	PDF
Jasper Report Engine	XLS
Jasper Report Engine	RTF
Jasper Report Engine	JPG
Birt Report Engine	PDF
Birt Report Engine	XLS
Birt Report Engine	XLSX
Birt Report Engine	RTF
Birt Report Engine	DOC
Birt Report Engine	DOCX
Birt Report Engine	PPTX
Birt Report Engine	CSV
Birt Report Engine	XML
Birt Report Engine	JPG
Birt Report Engine	TXT
Birt Report Engine	PPT
Qbe Engine	XLS
Qbe Engine	XLSX
Qbe Engine	CSV
Chart Engine	JPG
Chart Engine	PDF
Cockpit Engine	XLS
Cockpit Engine	XLSX
Cockpit Engine	PDF

The form to add a new exporter has the following fields:

- LINK THE EXPORTER TYPE AVAILABLE TO THE SELECTED ENGINE
- Select engine * (This is required)
- Select exporter * (This is required)

Fig. 3.74: Adding new exporter.

On the right side of catalogue you have two drop down menus where you can choose engine and exporter to link it with the engine. After you selected engine and exporter you can save it by clicking on a save button in the upper right corner. In the figure below you can see the example of creating new exporter.

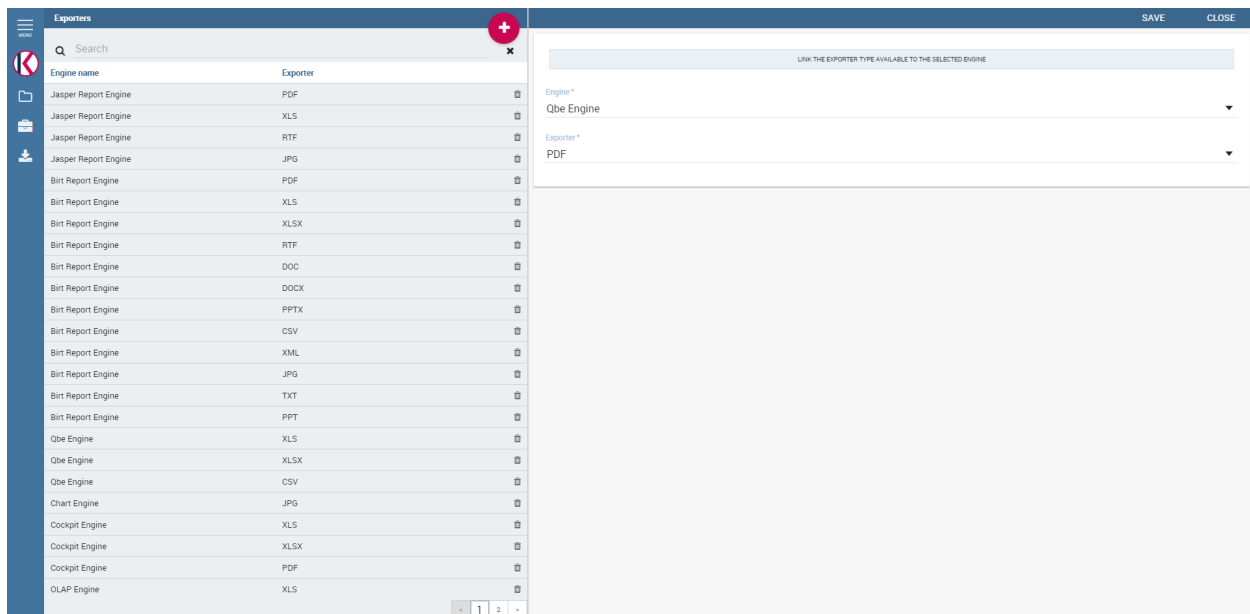


Fig. 3.75: Creating new exporter.

You also have possibility to change existing exporter and delete it.

3.9.4 Metadata

Knowage offers the possibility to define metadata categories and then give them a value for each analytical document and for each subobject.

In the metadata page, shown below, you can see the list of existing metadata. Here you can also define a new metadata using the dedicated button.

Metadata list						
<div> ⏪ ⏩ The value of the column <input type="text" value="Label"/> as a <input type="text" value="string"/> starts with <input type="text"/> Filter All ⏪ ⏩ </div>						
Label ▲	Name ▲	Description ▲	Type ▲	Used by # Docs ▲	Used by # Customized view ▲	
Business Questions	Business Questions	Business Questions	LONG_TEXT	1	0	
Creator	Creator	Creator	SHORT_TEXT	2	1	
Description	Description	Description	SHORT_TEXT	1	0	
KeyWords	KeyWords	KeyWords	SHORT_TEXT	2	1	
Language	Language	Language	SHORT_TEXT	3	1	
Long Description	Long Description	Long Description	LONG_TEXT	2	1	
Objective	Objective	Objective	LONG_TEXT	2	1	
<div> page 1 of 1 ⏪ ⏩ 1 ⏪ ⏩ </div>						

Fig. 3.76: List of existing metadata.

You define a new metadata by giving it a **Label**, a **Name**, a **Description** and a **Type**. The **Label** is a unique identifier, the **Name** is what will be shown to the end user and the **Type** can be either SHORT TEXT or LONG TEXT.

We recall that metadata visibility is one of the authorization you can set while creating roles. Only the users associated

to roles which have this authorization will view metadata. In addition, in order to edit metadata the user roles need to have another authorization called **Save Metadata**.

4.1 Introduction

This document is intended to provide a quick introduction to Knowage end user interface. It can guide you through your first approach to the suite, describing the different graphical elements and the general end user features common to all Knowage products.

No prerequisite are needed. We suggest to follow the elements description and try them while reading.

4.2 User Interface

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.

4.2.1 Preliminary information

The administrator will provide you a username and a password to log in Knowage environment.

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 visibility criteria on documents and data, according to their business profile.

This is what is called *Behavioural Model*. It allows to:

- reduce the required number of analytical documents,
- code only once the behavioural and visibility rules on data,
- guarantee the uniform growth of the project over time,
- guarantee the respect of the visibility rules over time, with no limit on the number analytical documents that can be added.

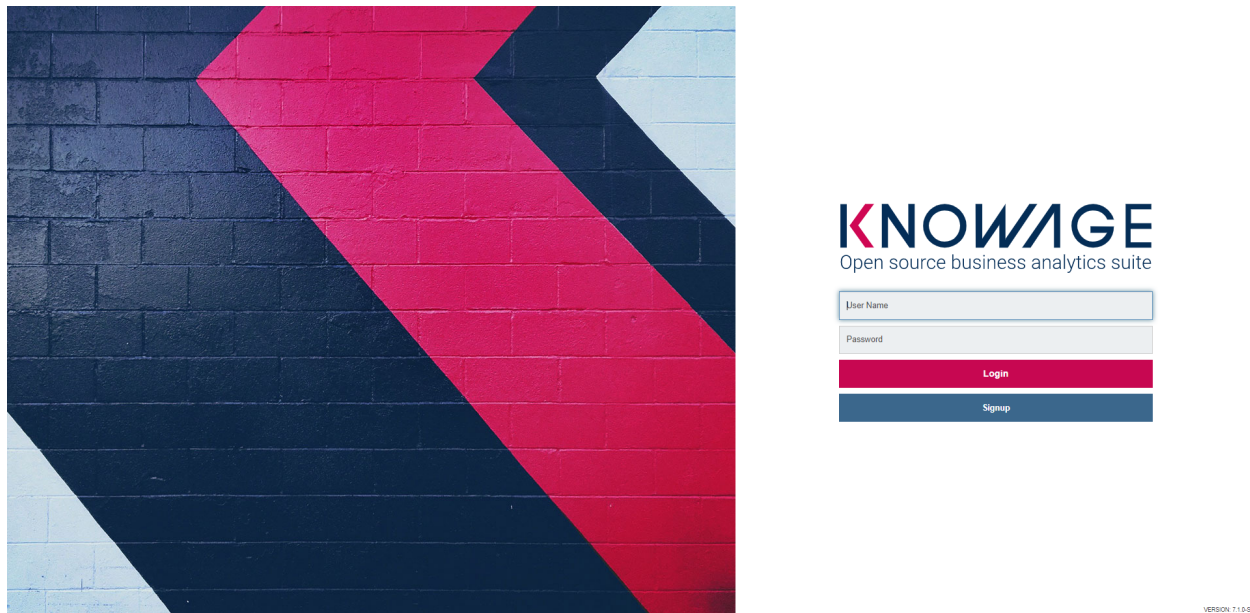


Fig. 4.1: Login page

All the analytical documents are strictly related to the behavioural model. In fact, the behavioural model guides the behaviour of the analytical documents according to the user's role, managing the visibility of documents and data.

Every role belongs to a predefined *role type*. The available role types are listed and described below.

Table 4.1: Role types

Role Type	Description
General administrator.	Manages all Knowage functionalities.
Model administrator.	Manages the behavioural model and its associated functionalities.
Developer.	Creates and modifies datasets and documents.
Test user.	Tests analytical documents.
End user.	Executes documents visible to him and creates ad-hoc reporting analysis.

From now on we suppose the reader has a role of type “End user”. If some grants are optional for this role we will state it.

4.2.2 Main menu

The 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: click it to open the menu.

Main menu consists in a set of icons associated with basic features. It is divided in two submenus: the general menu, which is collapsed, and the BI functionalities menu. It is important to underline that not all the components of the menu are mandatory, but they may be configured by the administrator according to user's needs.



You can open general menu clicking on  and then on **username** drop down menu.



Fig. 4.2: Home page

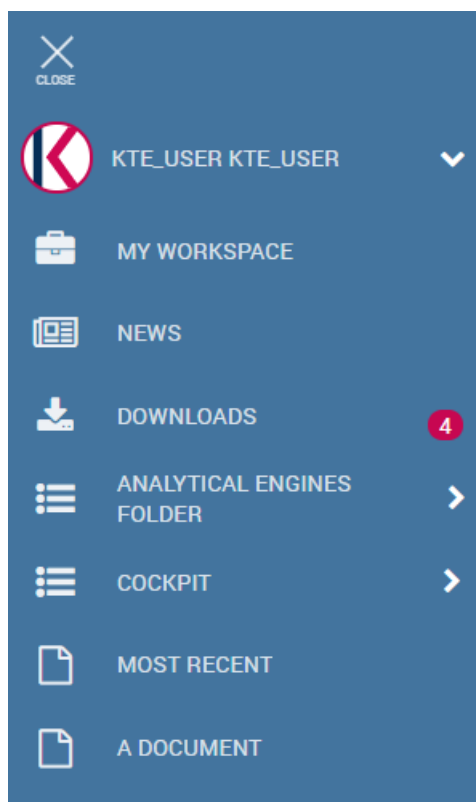


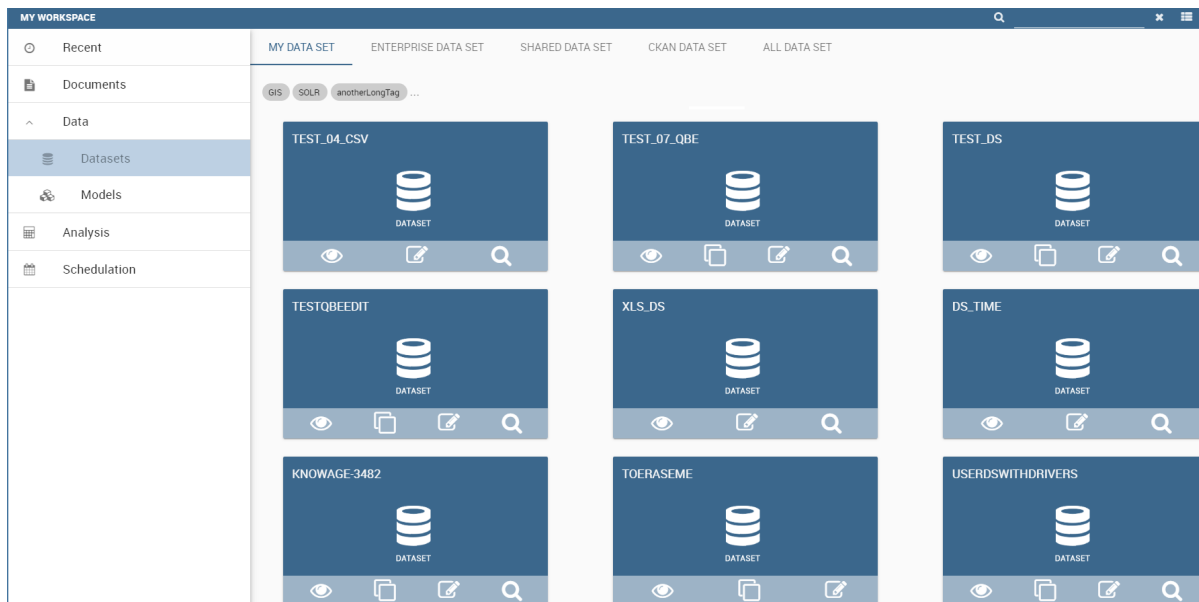
Fig. 4.3: General and BI functionalities menu.

Table 4.2: Menu components - General menu

Name	Description
My roles	Select the authentication role (available if you are associated to more than one role).
Languages	Language options.
Accessibility	Enable accessibility
Info	Infos about Knowage version.
Log Out	Log out.

Document Browser This is a standard functionality of Knowage Server. It enables you to access.

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** one you can access all the self-service BI features. Once entered this section, your datasets appear divided into four categories:



Functions catalog Lets the user enter the data mining functions a technical user has previously developed.

Glossary definition The user can define the proper glossaries and related words useful for his own analysis.

Calendar Allows the user to specify the festivity days of a certain time frame.

The **General menu** is identified with the first icon in General Menu 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.

To conclude the overview of this area we describe a not mandatory element. When configured by the Knowage administrator, you can have quick links to a static page, a document, a folder or an external application, like for

instance a web page. These links are displayed below the BI functionalities menu, as shown below.

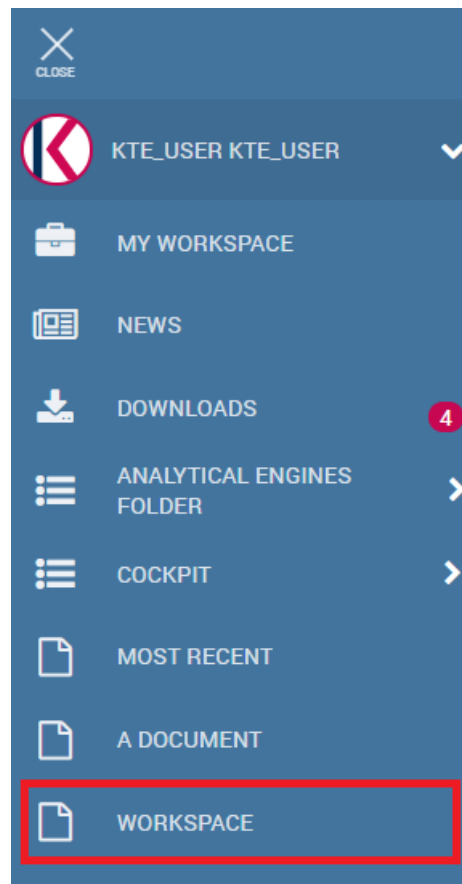


Fig. 4.4: Quick links


You can minimize the main menu by clicking the right arrow at its bottom right corner. This way the main menu is replaced by the menu button, so that you can close/open it 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.

4.2.3 Document Browser overview

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

By default, the page is divided in two parts, as shown in Figure above: 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.

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 below).

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.

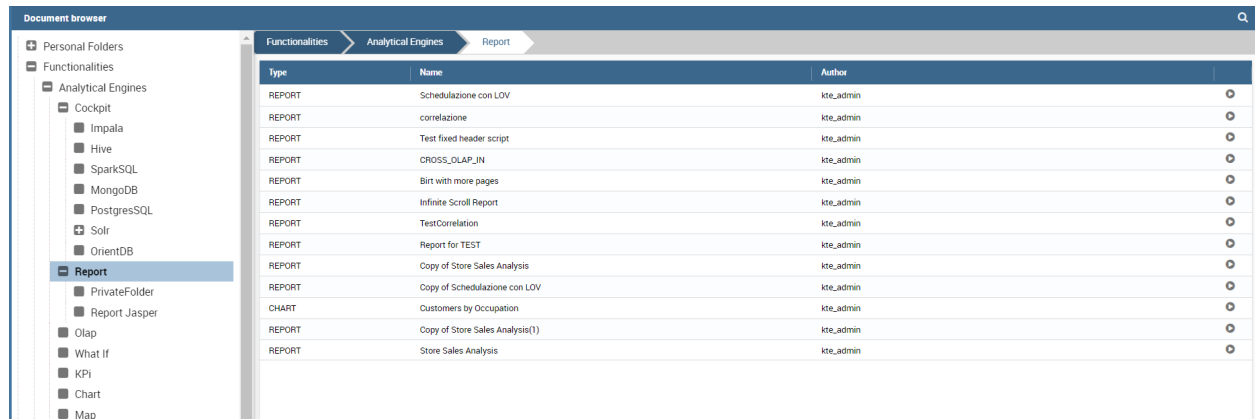


Fig. 4.5: Document Browser

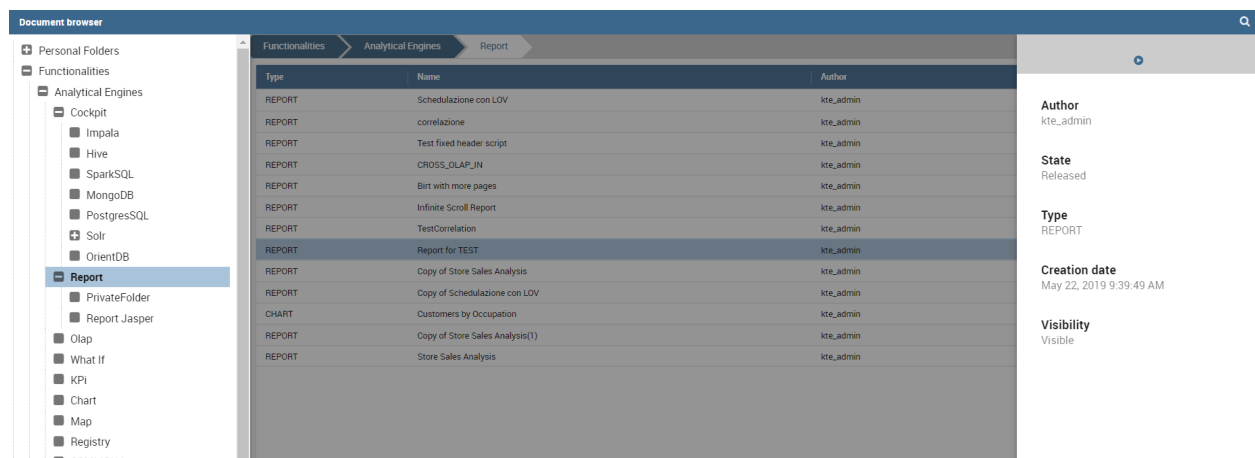


Fig. 4.6: Documents details expanded.

4.3 Document Execution

In this chapter 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.



Fig. 4.7: Back to Document Browser

4.3.1 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.

Mandatory parameters are shown in bold together with an asterisk on the right, while optional parameters are normal shaped.


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.

4.3.2 Document Toolbar

All documents inside Knowage environment share the same toolbar with different features. 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.

The  refreshes the document.

The  opens the parameters panel and it is visible only if there are parameters associated to the document.

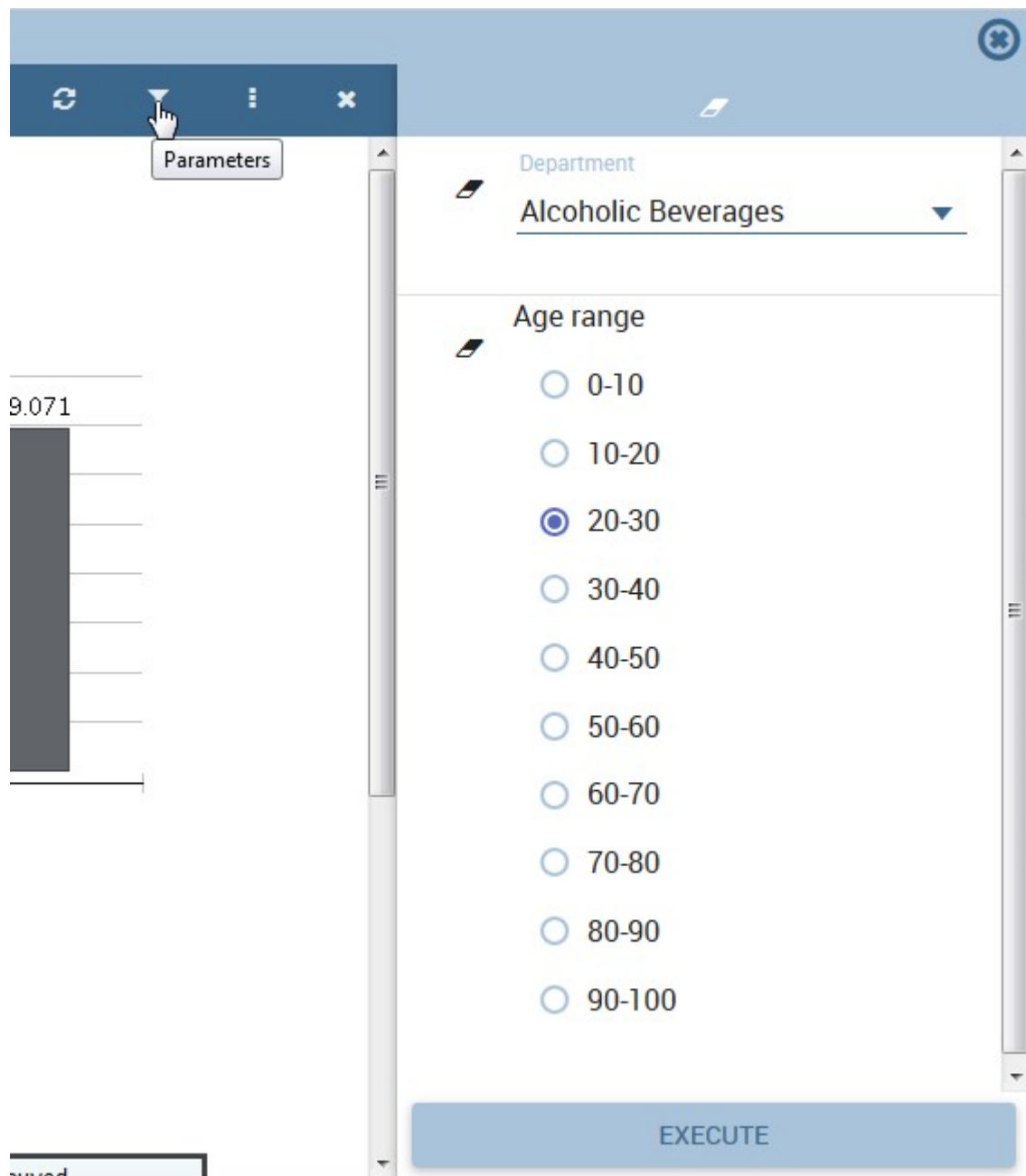



Fig. 4.8: Parameters panel



Fig. 4.9: Document Toolbar

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

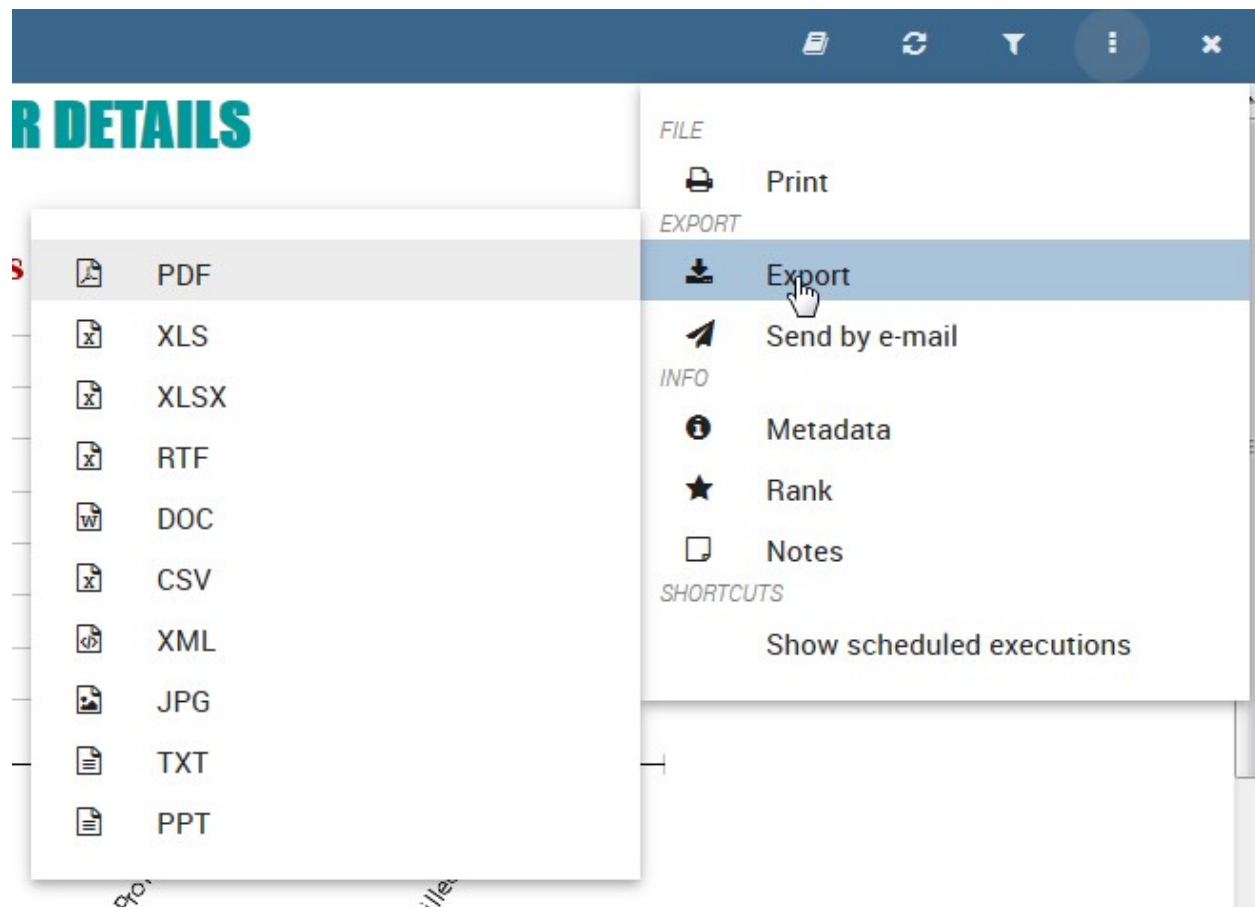


Fig. 4.10: Toolbar contextual menu

4.3.2.1 Exporters

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

Clicking **Export** in the document toolbar you will see the available formats for the current document. Select one and check the exported document.

4.3.2.2 Business and structural metadata

Knowage allows the definition of business metadata to describe an object, in our case a document. Business metadata, unlike technical metadata used in Knowage Meta to build the metamodel, are business information associated to the document intended to help users to understand, access and classify it. As such, they have been mainly conceived for the end user understanding.

There are three types of business metadata, some of them are editable while others can only be read. In particular, general metadata are read-only, while short and long text metadata are editable. General metadata contain basic information about the document, which cannot be altered because they are related to the structure of the document (e.g., type, engine, label). They provide useful and synthetic information on the document. Short and long text

metadata should be used to add relevant business information: all the allowed users will see this information, which will help them understand the purpose and context of that document.

In general, metadata should be edited by users with adequate expertise and authority to do so. Therefore, it is possible for the administrator to assign the right to edit and save metadata only to some users. The right is not specific to a profile, but it is part of the authorizations that can be granted to any role. This applies to bookmarks as well.

Metadata can be accessed from the toolbar clicking the corresponding item in the contextual menu as shown in the following figure.

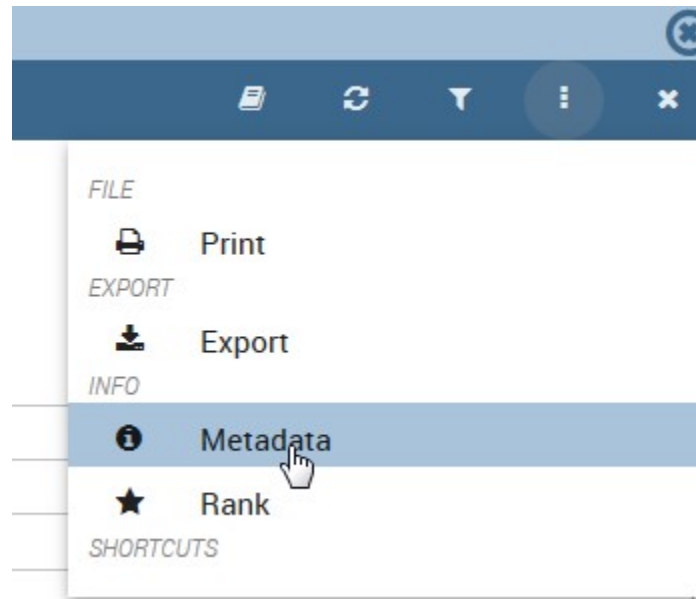


Fig. 4.11: View document metadata

As stated before, in order to see and/or edit metadata the user roles need some grants. Knowage administrator manages this authorizations. If you have editing metadata authorization, you will be able to see to change them. If you want to edit short metadata just click in the value area and write what you prefer. If you want to edit long metadata just click in the value area and an HTML editor will appear.

When you are satisfied with what you wrote just click on the **SAVE** button. We provide the following example.

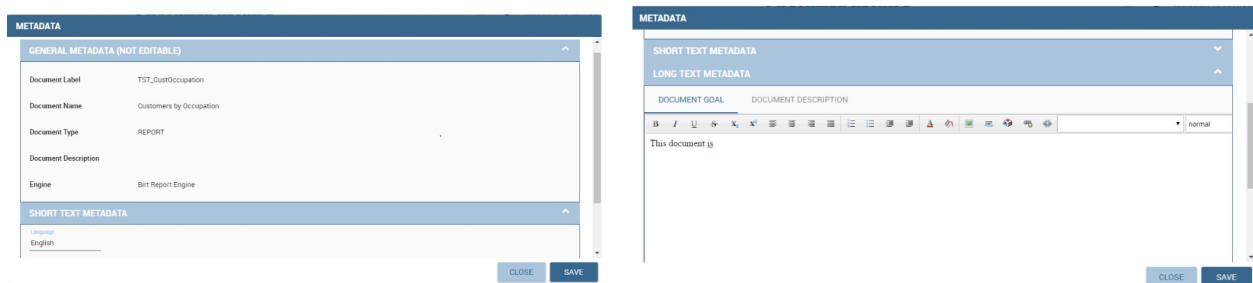


Fig. 4.12: Left: General and short metadata, Right: editing long metadata

4.3.2.3 Notes

Knowage offers a simple collaborative tool to share notes and comments on documents, allowing users to share information and receive feedback. This may be useful, for example, to limit the number of exchanged e-mails: notes can

be stored online and are accessible to all users sharing the same access rights. Each user can add a single note to a single document.

Click on **Notes** in the contextual menu of the document toolbar. In this way a text editor opens where you can type your comment. Make it private or public (i.e., accessible to users with the same rights as you) by selecting the preference in the appropriate box. Click **Save** to confirm. All public comments from all users, as well as your private notes, will be shown the next time you open this window. If you want to edit or delete a note, click on the Rate document corresponding symbols in the **Note list** tab. You can also export the note in PDF or RTF format.

4.3.2.4 Rate document

The aim of this functionality is to acquire explicit quality data that can be further used. Specifically, it allows the administrator to identify anomalies about the use of an analytical document by end users. Rating a document means assigning it a value from the end user perspective. This brings additional information with respect to traditional audit and monitoring data, which can track the number of executions but cannot interpret users motivations and feelings.

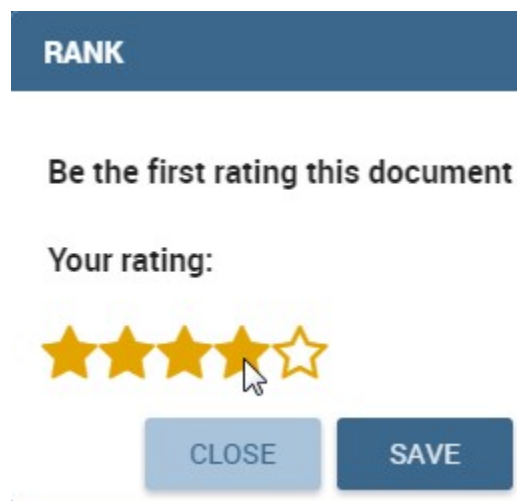


Fig. 4.13: Rating a document

To rate documents, click on **Rank** in the contextual menu of the document toolbar. Choose the document ranking in the pop-up window and click **Save**. The administrator can exploit the result of votes, in order to evaluate and improve the quality of the document.

4.3.2.5 Scheduled Execution

The administrator has the possibility to set schedulation for documents. These schedulations can be accessed by **Show Scheduled executions**.

4.4 User registration

Important: Signup is available only when `SPAGOBI.SECURITY.ACTIVE_SIGNUP_FUNCTIONALITY` configuration is enabled.

From the login page, new users can register their self to Knowage clicking on **Signup** button:

NOTES

NOTE
NOTES LIST

B
I
U
S
x₂
x²

Verdana
normal
Paragraph

My first note.

Private
CLOSE
SAVE

Fig. 4.14: Share notes and comments



KNOWAGE
Open source business analytics suite

User Name

Password

Login

Signup

VERSION 7.1.0.5

Fig. 4.15: Login page

That bring the user to the registration page:

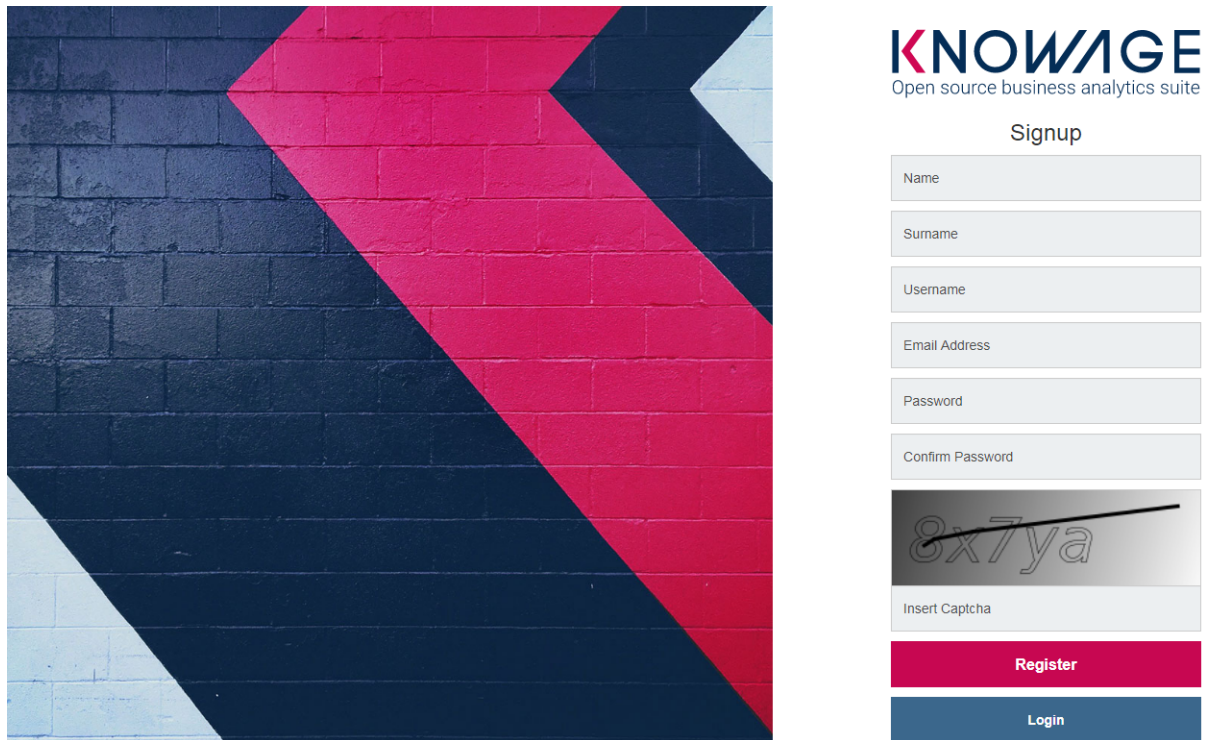
The image shows a user registration page for Knowage. On the left is a decorative background image of a brick wall with large geometric shapes in dark blue, red, and light blue. On the right is the registration form. At the top right is the Knowage logo with the tagline 'Open source business analytics suite'. Below the logo is the heading 'Signup'. The form consists of several input fields: 'Name', 'Surname', 'Username', 'Email Address', 'Password', and 'Confirm Password'. Below these is a CAPTCHA image showing the text '8x7ya' with a diagonal line through it. Under the CAPTCHA is a label 'Insert Captcha'. At the bottom of the form are two buttons: a red 'Register' button and a blue 'Login' button.

Fig. 4.16: User registration page

Where the user can insert his personal data, his username and his password (password constraints can be set using advanced configuration). After the registration the user will receive the email that let him verify himself. After email confirmation, account will be enabled and user will access Knowage.

5.1 Basic Data Access

A dataset is a way to read data from different sources and represents the portion of data used by various documents. Inside Knowage it plays a crucial role to develop analytical documents such as the interactive dashboards, reports and map documents. In particular, it is the main way to let the unskilled user make his/her own analysis. In fact he/she can create his/her own dataset uploading an XLS or a CSV file or use a dataset already defined. Knowage offers also the chance to download open data from WEB thanks to CKAN integration. Moreover it is possible to create more complete dataset from different sources through the dataset federation functionality.

We highlight that this chapter describes basic but also advanced features which are available only in KnowageBD and KnowageSI products.

Important: Enterprise Edition

If you purchased Knowage EE, the advanced features explained in this chapter are available only in KnowageBD and KnowageSI products

Let us suppose to enter, with end user credentials, the data management area clicking on the **Workspace** icon from BI functionalities menu as shown in figure below and the **Data** section of the window.

Afterward you have the subsections: **Dataset** and **Models**. Select **Models** to explore the models and the **Dataset Federation** area. Please note that the **Dataset Federation** functionality is available only in KnowageBD and KnowageSI.

5.1.1 Dataset

Into the “Dataset” area we find all the datasets classified according to their types. The datasets are categorised as follows:

- **My dataset:** datasets created by the end user that made the access, uploading a CSV or XLS file or creating a query on a business model using the Qbe interface;

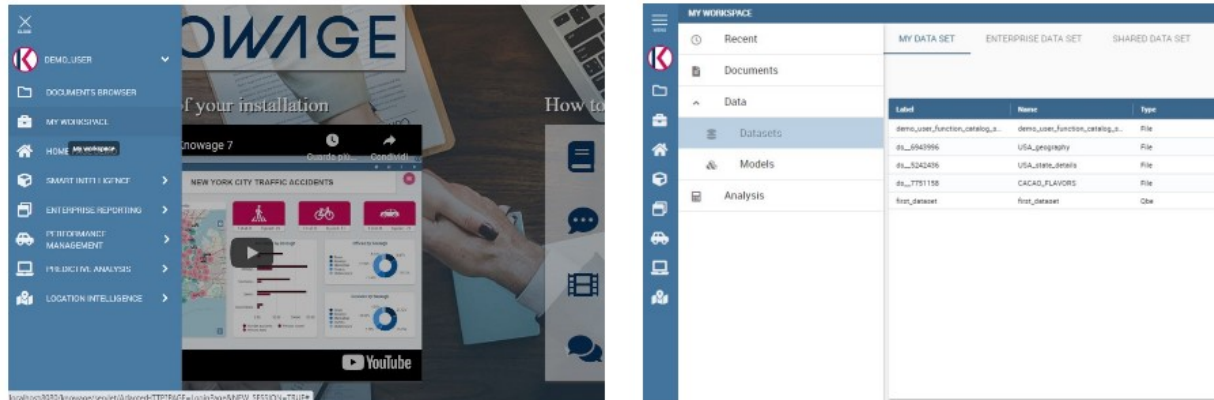


Fig. 5.1: Access to **My Data** area

- **Enterprise dataset:** certified datasets, namely datasets created by the technical/experts users and shared with the end user.
- **Shared dataset:** datasets created and shared by other end users (with the same role);
- **CKAN dataset:** in this area it is possible to download public datasets and visualize then in the CKAN datasets area;
- **All dataset:** in this folder all previous datasets are stored, namely all datasets contained in the classes just described.

5.1.1.1 My dataset

In this area the end user can create datasets uploading his/her own files.

Click **Create Dataset** to open the dataset wizard which guides you through the dataset creation. It is possible to choose between XLS or CSV file as in the following figure.

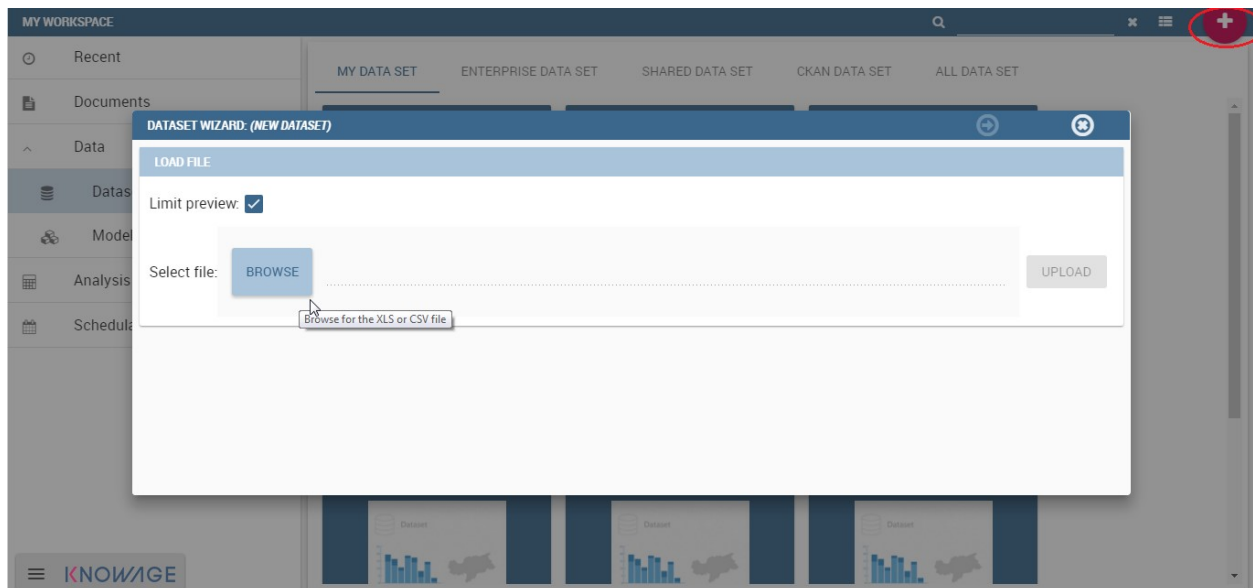


Fig. 5.2: Dataset creation.

In the example shown in the next figure, we upload an XLS file.

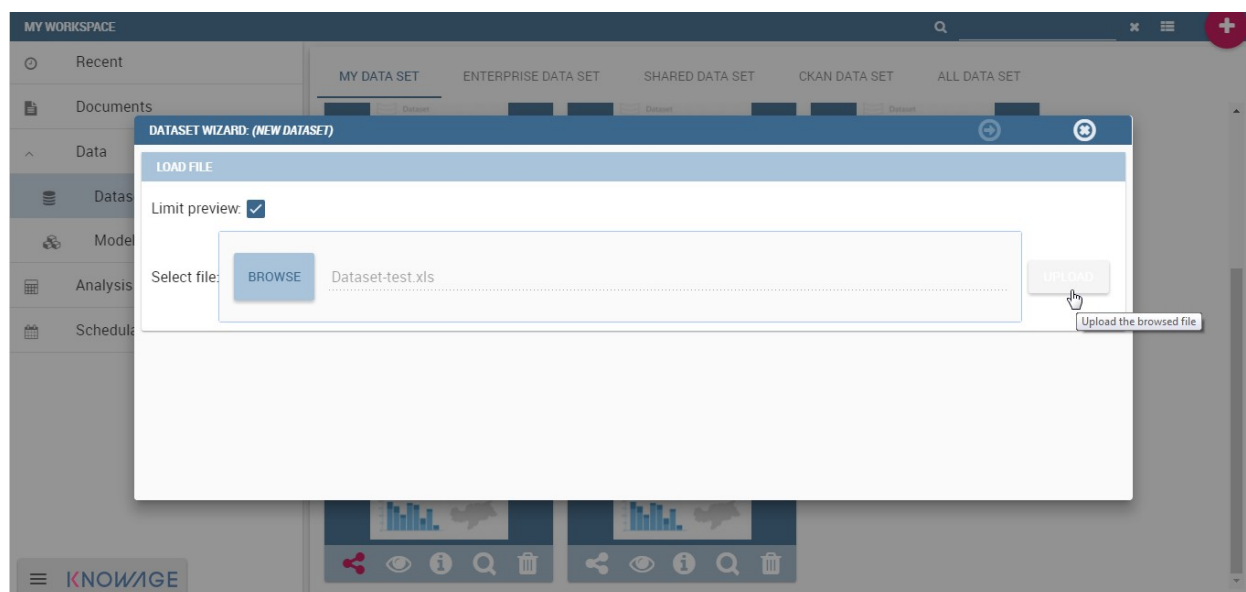


Fig. 5.3: Uploading XLS for dataset.

The wizard, shown below, 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.

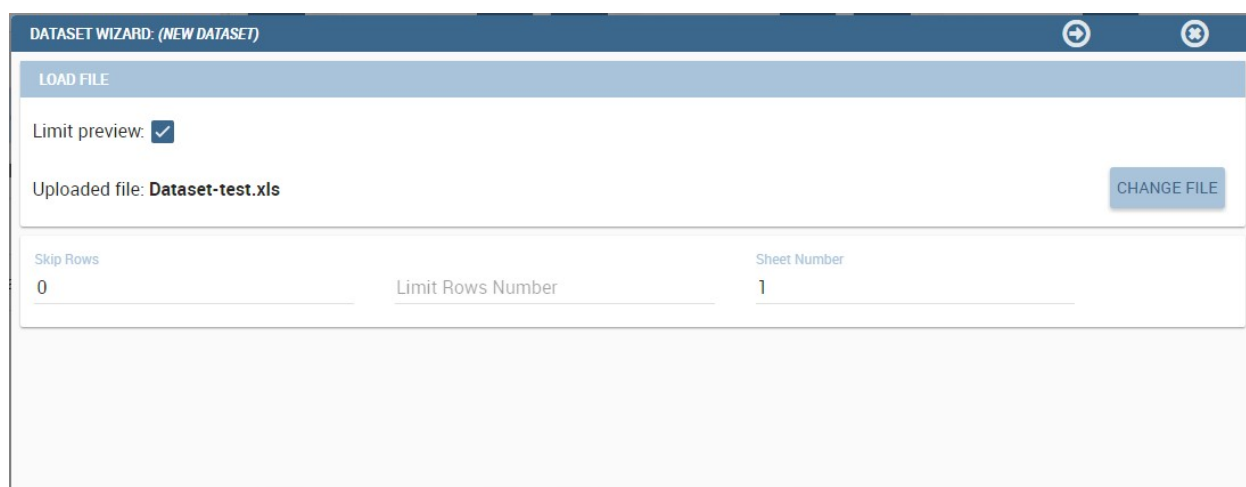


Fig. 5.4: Configuration features.

Once the file has been uploaded, it is important to check and define the metadata (measure or attribute) for each column. To switch a measure to an **attribute** (or viceversa), click on **Value** column of the interested row field as shown below.

Just few steps before saving the dataset:

- Check the data preview in order to verify the accuracy of data;
- enable or disable the persistence of dataset. Thanks to this functionality the server creates a snapshot of the extracted data in order to avoid to reload the dataset each time that the user revokes it;
- finally, name and save the dataset as shown below.

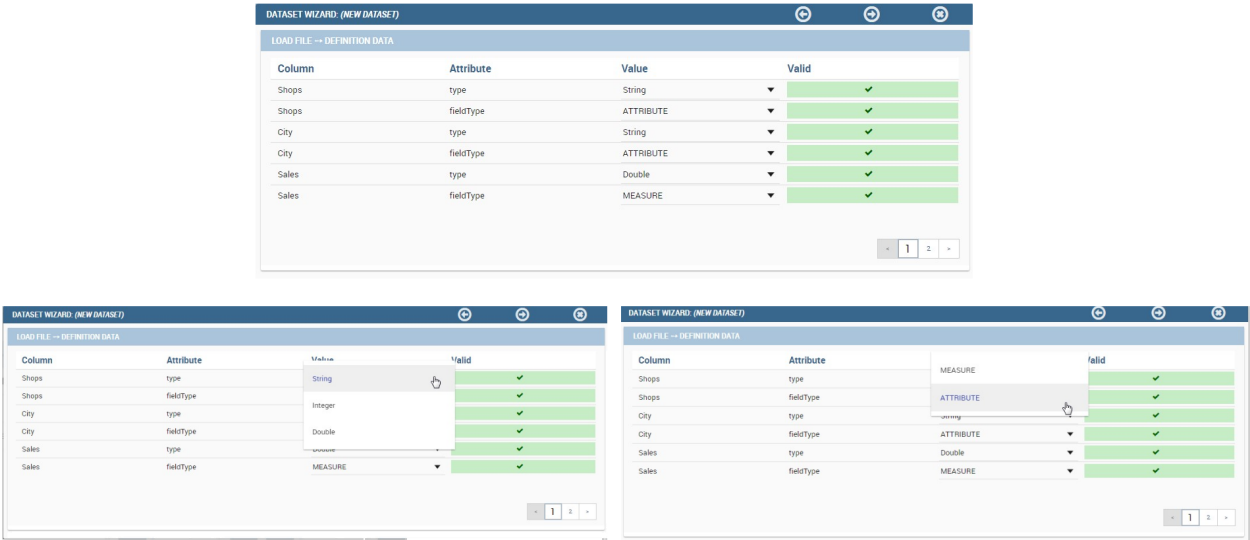


Fig. 5.5: Change metadata.

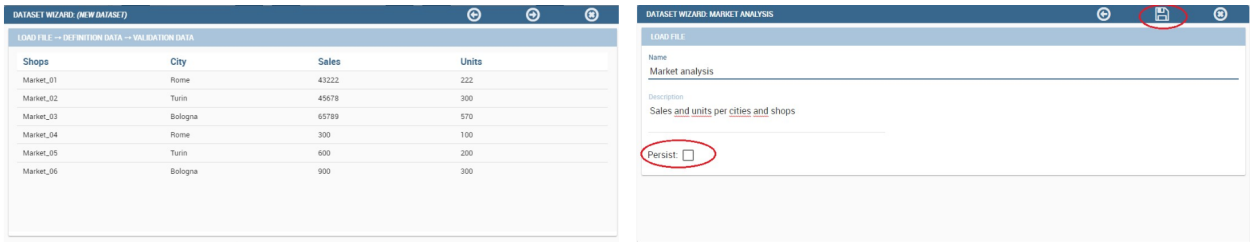


Fig. 5.6: Saving dataset.

As we discussed previously, all created datasets are available under **My dataset** area. Here it is possible to share/unshare them by clicking on the **share** icon (have a look at the next figure). 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 exact role.

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

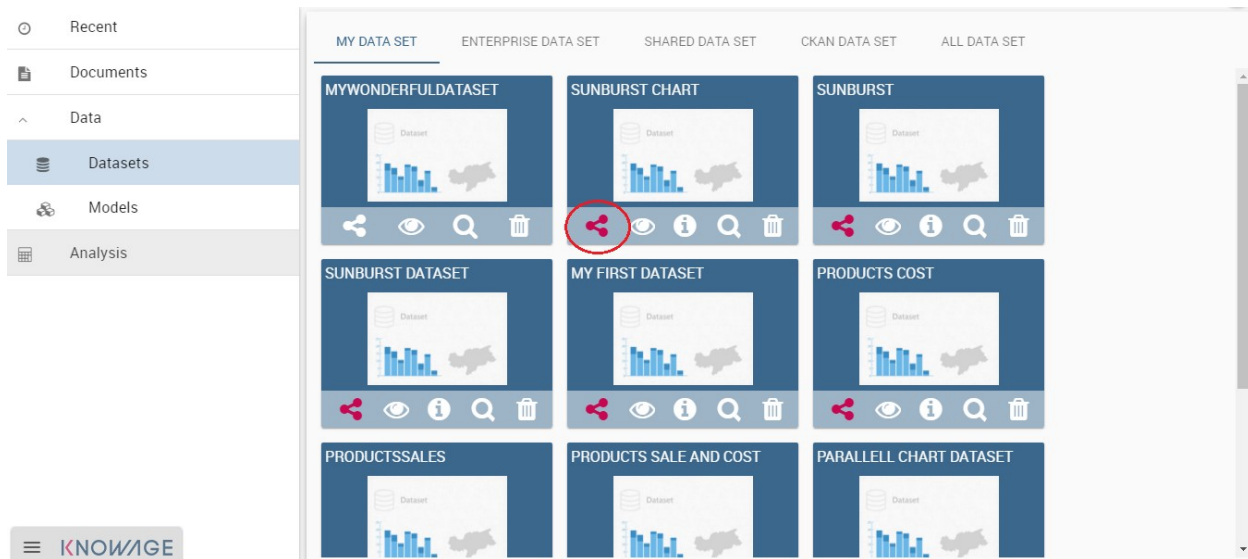


Fig. 5.7: Share a dataset.

5.1.1.2 CKAN integration

Thanks to CKAN integration it is easy to have access to datasets published in the World Wide Web (e.g. datahub.io, data.gov, data.lab.fiware.org, dati.gov.it, and more). Indeed, CKAN is the world’s leading open-source data portal platform. It is a powerful data management system that makes data accessible by providing tools to streamline publishing, sharing, finding and using data. CKAN is addressed to data publishers (national and regional governments, companies and organizations) who want to make their data open and available. So you can search and handle open data in a self-service manner.

Warning: CKAN Datasets

CKAN datasets can be divided into four main categories: “Public”, “Organization private”, “Acquired”, “User private”. Remember to download and use only the datasets having a **Public** category.

5.1.1.2.1 CKAN datasets access method

To start using CKAN datasets inside Knowage suite, go to the **CKAN Dataset** tab in the “Dataset” subsection of “Data” section under “My Workspace”. As shown in figure below, the user has to choose the repository from the combobox and then click on the repository name to access it.

A preview of datasets stored in the chosen repository will be shown.

At this point data are not usable yet, but he/she can start to handle them as we will show in the following sections. The datasets are shown with their name and description. By moving the cursor over a dataset, a list of available actions will appear. Clicking on the **Info** button, a set of information from the original CKAN resource and about the dataset

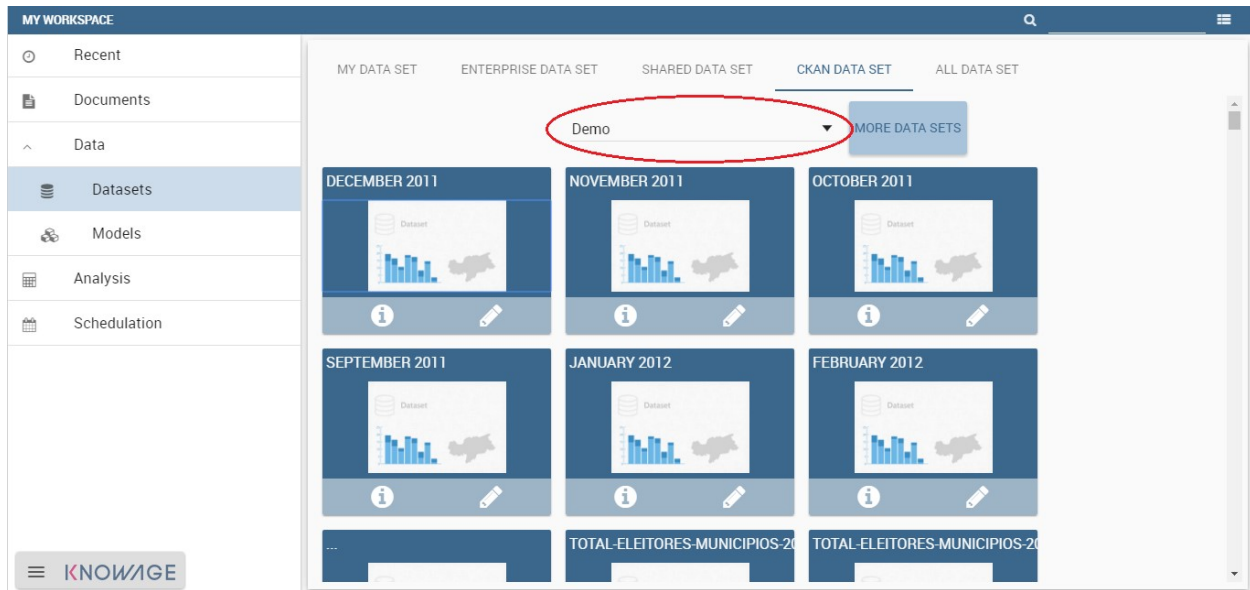


Fig. 5.8: CKAN Repositories.

status (e.g. visibility, last modification date) will be displayed by Knowage, as in the following figure. To use one of them the user has to import metadata information and then analyse the dataset on demand.

5.1.1.2.2 Export dataset

Note that once the dataset has been created, the user may find useful to get an excel from it. Knowage has designed a specific button to fulfil this need that the user can find exploring the detail panel of the dataset, as reported below.

5.1.1.2.3 Save and handle dataset

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

As a second step the user has 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 downloaded datasets. In particular use the eye-shaped icon to refresh the dataset or change metadata by repeating the download process and the magnifier icon to inquire it through the QbE interface.

5.1.2 Models

Here there are the models that a technical user has built for the end use that can query it using the QbE interface and create his/her own dataset from them.

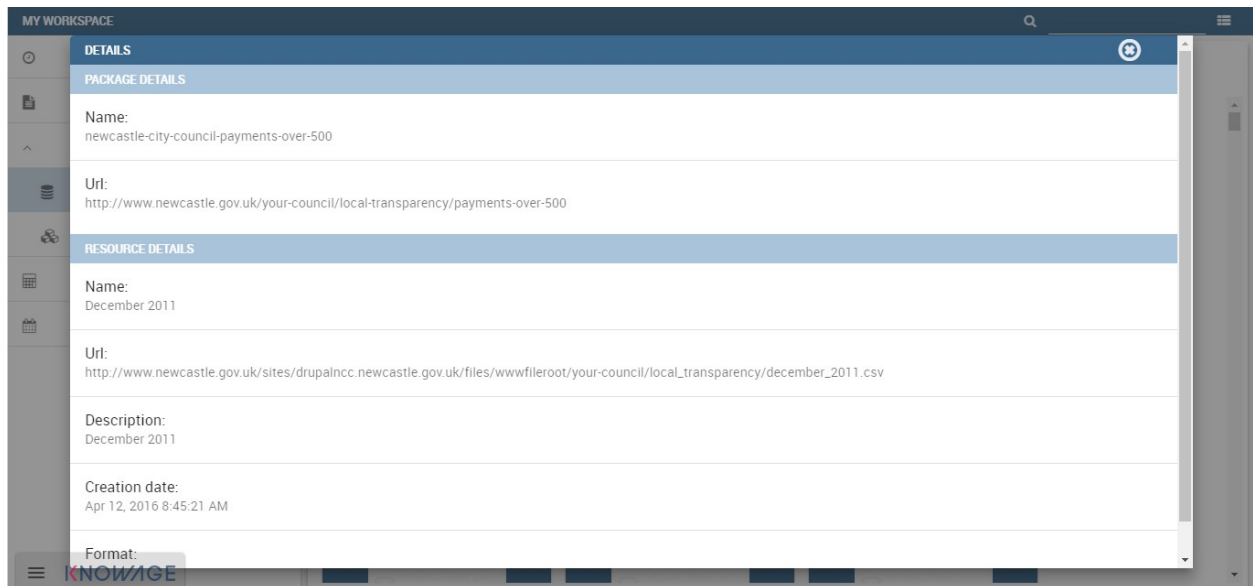


Fig. 5.9: CKAN dataset details.

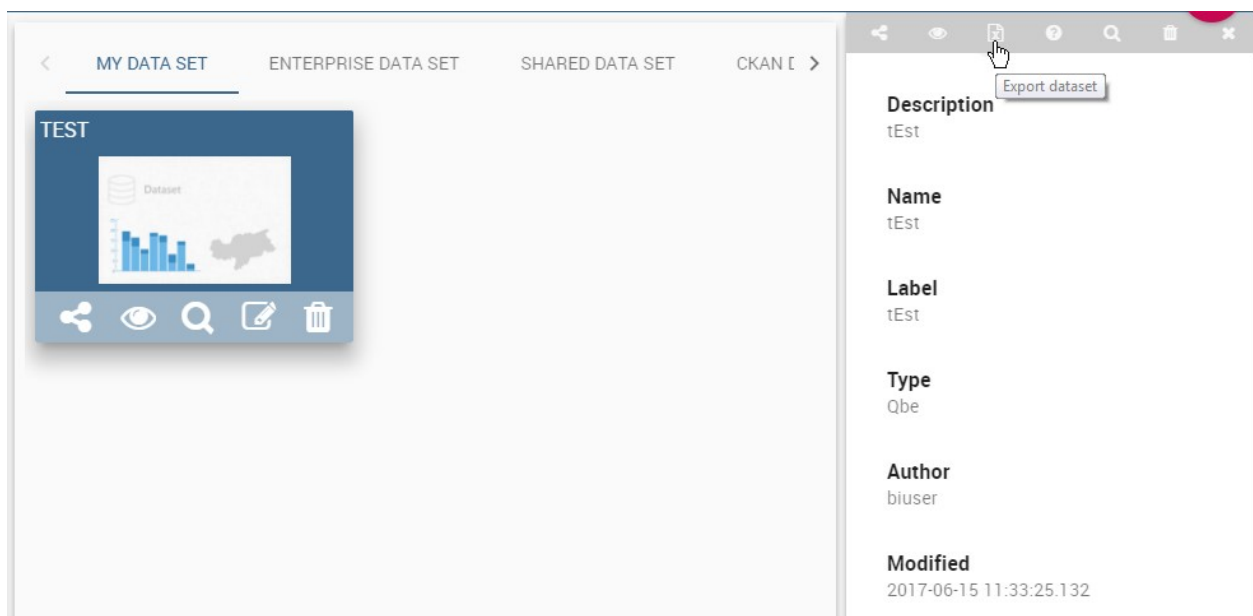


Fig. 5.10: Export dataset.

5.1.3 Dataset federation

Dataset federation is a functionality available only in KnowageBD and KnowageSI. Thanks to the Data federation functionality, you can create a new dataset combining two or more datasets according to your role permissions. Let us give you an example. Suppose you have stored in a database your products information (i.e. sales, costs, promotions ecc.) and you find as open data the customers feedbacks on these products. If you create datasets on these Dataset federation 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 available datasets and choose the ones you want to federate. 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 the following figure we choose Product.

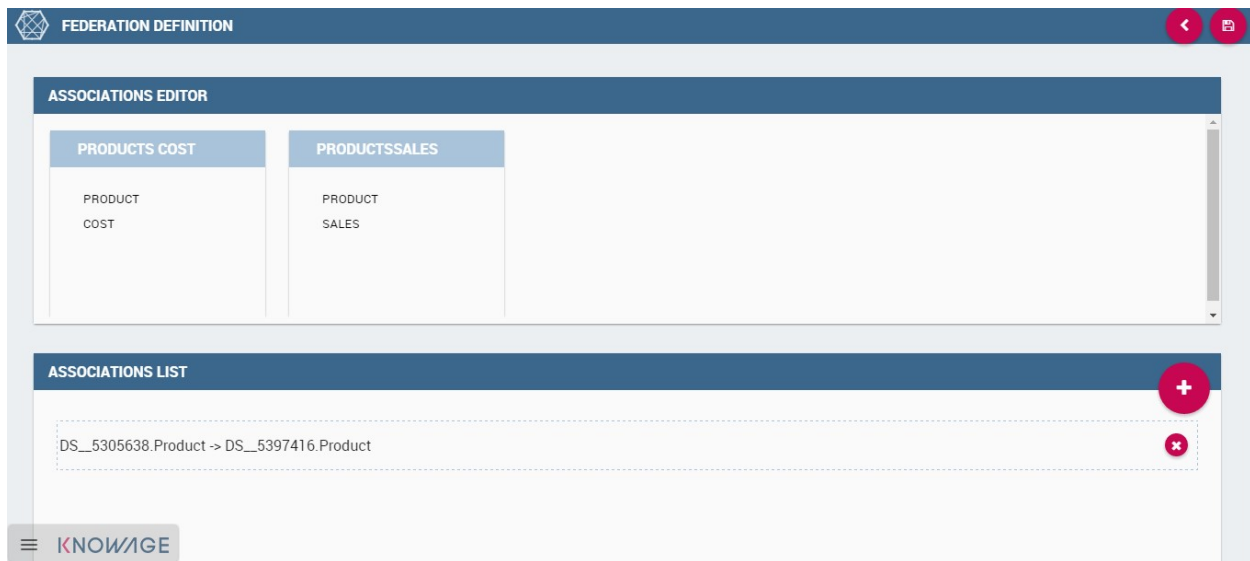


Fig. 5.11: Federated dataset details.

Once saved, The new federation has been created in **Federation definition** and you can find it 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.

5.2 Advanced Data Access

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 below. This area offers 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 identical. For this reason in this chapter we are going to describe only the dataset creation and management.

5.2.1 My first dataset

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

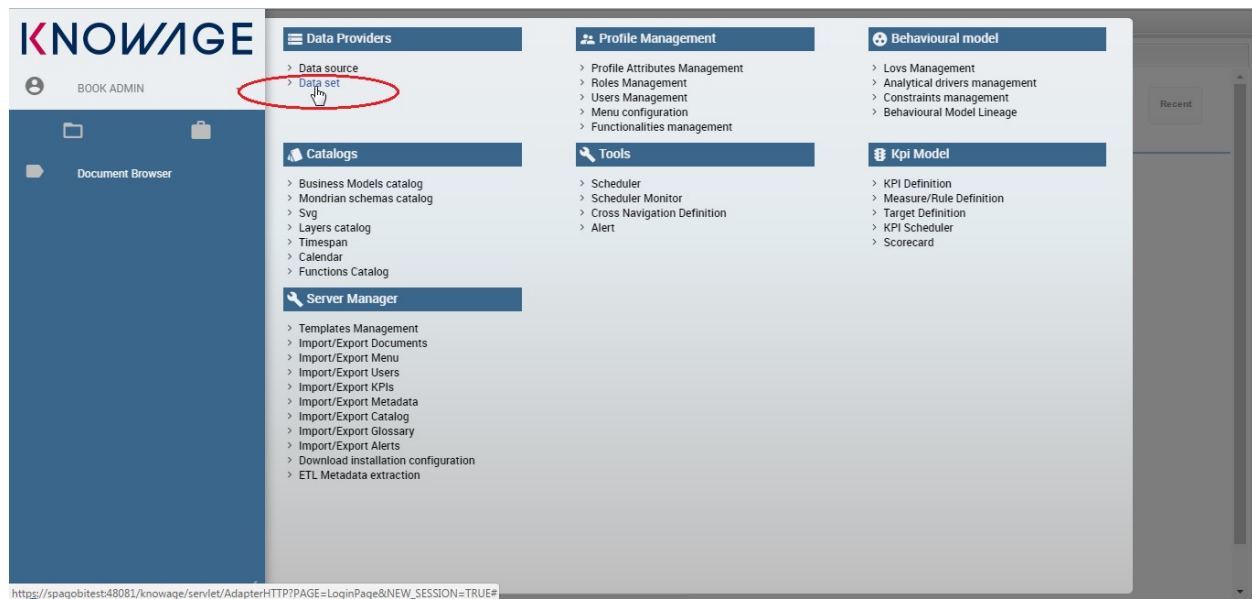


Fig. 5.12: Access data set creation area.

A dataset acts as a data provider for analytical documents that's 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,
- Custom,
- Flat,
- Ckan,
- Federated,
- REST,
- Big Data,
- Solr,
- Python/R.

All types of dataset share some common operations, while others are specific to each of them. The process for defining a dataset inside Knowage follows:


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.

5.2.1.1 New dataset creation

The dataset graphical editor is divided into two areas: the left side 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.

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 below). There is also a new feature, data set tags. These tags are used for filtering list of data sets and they are placed in the upper left corner of data set catalogue above list of data sets as you can see in the figure below.

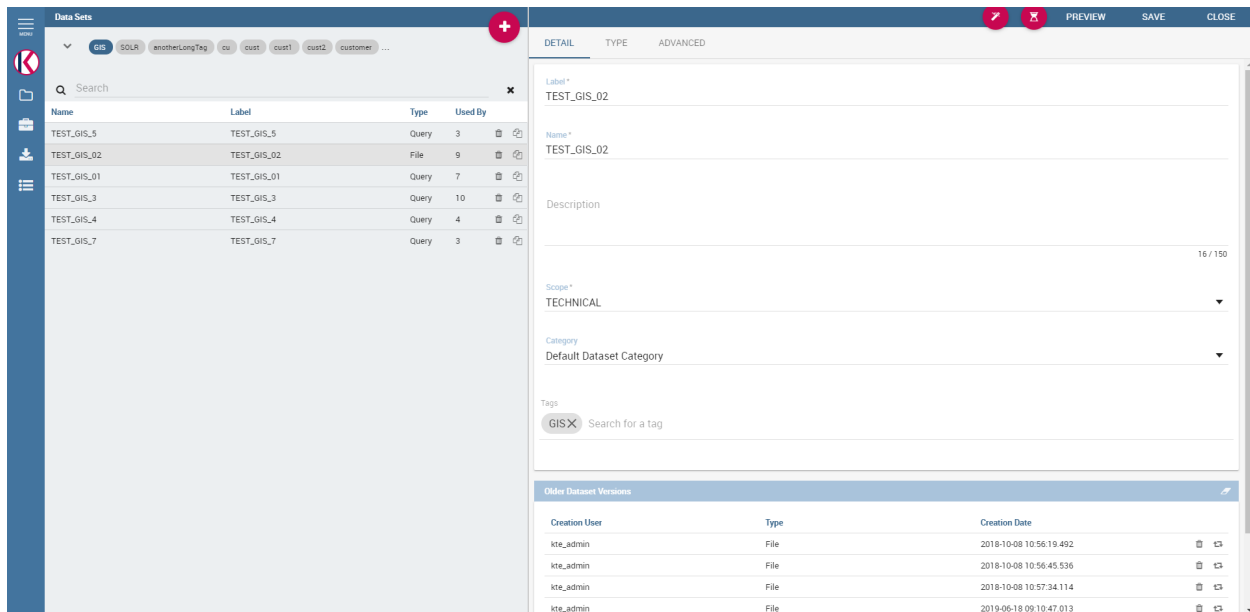


Fig. 5.13: Dataset Panel.

By clicking on some of the tags, user is filtering data set list to show only data sets with that tag. In the detail part of data set, user can define tag for that data set by writing the name of the tag in the input field Tags. By clicking on Save button, user saves the new tag and he can see it in the data set tags list.

In the lower part you can see a versioning system for the dataset: Knowage supports dataset versioning, as shown in figure below, therefore, each time you edit and save a dataset, the older version is archived and is still accessible from the lower part of the detail panel.

PREVIEWSAVECLOSE

DETAILTYPEADVANCED

Label *

TEST_GIS_02

Name *

TEST_GIS_02

Description

16 / 150

Scope *

TECHNICAL

Category

Default Dataset Category

Tags

GISX Search for a tag

Older Dataset Versions









Creation User	Type	Creation Date	
kte_admin	File	2018-10-08 10:56:19.492	 
kte_admin	File	2018-10-08 10:56:45.536	 
kte_admin	File	2018-10-08 10:57:34.114	 
kte_admin	File	2019-06-18 09:10:47.013	 

Fig. 5.14: The dataset versioning.

The Scope lets you choose between two options, whose combination allows the definition of fine-grained purpose datasets. In Table below all details of possible matching are provided.

Table 5.1: Scope options

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.

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: here you have to 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. An example is shown below.

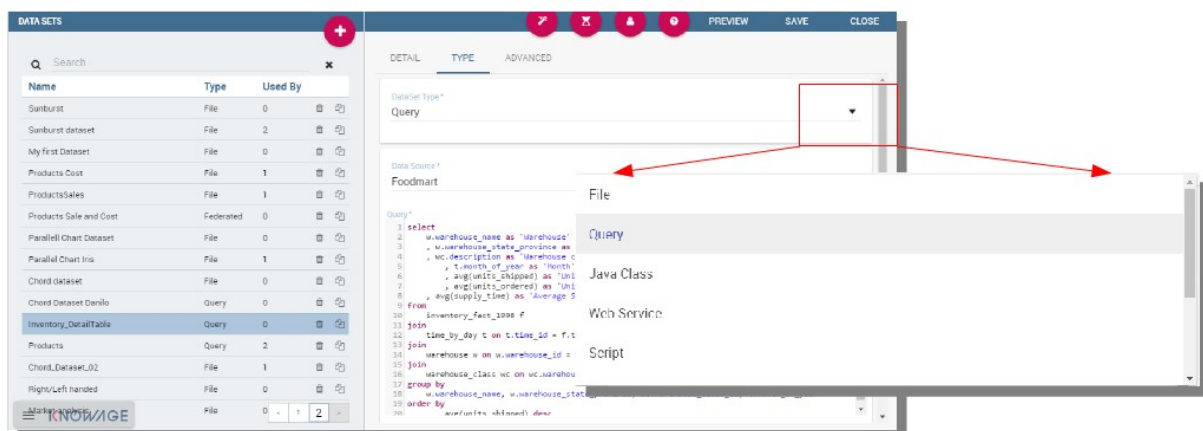




Fig. 5.15: The dataset type definition.

In the **Advanced** tab, shown in figure below, 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.

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

Let us describe more deeply each type of dataset.

DETAIL TYPE **ADVANCED**

Transformation Type: PIVOT_TRANSFORMER

Persist: ☐

Fig. 5.16: The dataset transformation tab.

5.2.1.2 File Dataset

A dataset of type File, see the following figure, 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.

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

5.2.1.3 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. Look at SQL query example.

Listing 5.1: SQL query example

```

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

```

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 next figure) and the script editor will open. 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 Query dataset's script example 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.

Listing 5.2: Query dataset's script example

```

1 if( parameters.get('year') == 1997 ) {
2   query = query.replace("FROM sales_fact_1998", "FROM sales_fact_1997");

```

(continues on next page)

The screenshot shows a configuration window for a 'File' dataset. At the top, there is a dark blue header bar with icons for edit, refresh, user, and help, followed by buttons for 'PREVIEW', 'SAVE', and 'CLOSE'. Below the header, there are three tabs: 'DETAIL', 'TYPE' (which is selected), and 'ADVANCED'. The 'TYPE' tab contains a 'DataSet Type *' dropdown menu with 'File' selected. Below this, there is a 'Select file:' label, a 'BROWSE' button, a text input field, and an 'UPLOAD' button.

Fig. 5.17: File Dataset.

The screenshot shows a 'Script editing for dataset' window. On the left, there is a 'DATA SETS' panel with a search bar and a table of datasets. The 'Products' dataset is selected. The main panel has a dark blue header bar with icons for edit, refresh, user, and help, followed by buttons for 'PREVIEW', 'SAVE', and 'CLOSE'. Below the header, there are three tabs: 'DETAIL', 'TYPE' (which is selected), and 'ADVANCED'. The 'TYPE' tab contains a 'Data Source *' dropdown menu with 'Foodmart' selected. Below this, there is a 'Query *' section with a text area containing a SQL query. At the bottom of the text area, there is a red circle around an 'EDIT SCRIPT' button.

Name	Type	Used By
Sunburst	File	0
Sunburst dataset	File	2
My first Dataset	File	0
Products Cost	File	1
ProductsSales	File	1
Products Sale and Cost	Federated	0
Parallel Chart Dataset	File	0
Parallel Chart Iris	File	1
Chord dataset	File	0
Chord Dataset Danilo	Query	0
Inventory_DetailTable	Query	0
Products	Query	2
Chord_Dataset_02	File	1
Right/Left handed	File	0
Market analysis	File	0

```
1 select * from product p
2 join product_class pc on p.product_class_id = pc.product_class_id
```

Fig. 5.18: Script editing for dataset.

(continued from previous page)

```

3 } else {
4   query = query; // do nothing
5 }

```

5.2.1.4 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\ KnowageWEB-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:

```

1 <ROWS>
2   <ROW value="value1" .../>
3   <ROW value="value2" .../>
4   ...
5 </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.

5.2.1.5 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.

```

1 <ROWS>
2   <ROW value="value1" .../>
3   <ROW value="value2" .../>
4   ...
5 </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.

5.2.1.6 QbE

Important: Enterprise Edition

If you purchased Knowage EE, this feature 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 below.

The screenshot displays the Knowage QbE Dataset interface. On the left, a table lists datasets with columns 'Name', 'Type', and 'Used By'. The 'Units_ProductFamily' dataset is highlighted. Below this table is a 'SHOW QBE QUERY' button. On the right, a 'DETAIL' panel shows the 'DataSet Type' as 'Qbe', 'Data Source' as 'Foodmart', and 'Datamart' as 'Qbe'. Below this are 'VIEW QBE QUERY' and 'OPEN QBE' buttons. A 'PARAMETERS' table is also visible, with columns 'Name', 'Type', 'Default Value', and 'Multivalued'. At the bottom, a 'Query Filter' panel shows a list of filters and a 'Query Editor' with a JSON query structure.

Name	Type	Used By
MyWonderfulDataset	Qbe	0
Cockpit00	Query	0
Cockpit_Parametric	Query	0
CKP_SalesMonthEducation	Query	10
CKP_CostsMonthEducation	Query	2
CKP_CostsMonthState	Query	10
BDH	Query	0
GIS_LIA_IAT3_I2	Query	1
GIS_DS_HSOL	Query	0
test	Query	0
chart_sample	Query	1
Units_ProductFamily	Query	0

Name	Type	Default Value	Multivalued
Units_ProductFamily	Query		

```

{
  "catalogue": {
    "queries": [
      {
        "id": "q1",
        "name": "query-q1",
        "description": "query-q1",
        "distinct": false,
        "selectedExpression": false,
        "graph": {
          "fields": [
            {
              "expression": {},
              "having": {},
              "subqueries": [
                {
                  "subquery": {}
                }
              ],
              "relations": {}
            }
          ]
        }
      }
    ]
  }
}

```

Fig. 5.19: QbE Dataset.

5.2.1.7 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()`.

5.2.1.7.1 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 below.

5.2.1.7.2 Ckan

Important: Enterprise Edition

If you purchased Knowage EE, this feature 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:

- **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. Allowed 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;

The screenshot shows the Knowage interface for configuring a Flat Dataset. The 'PREVIEW' button is highlighted with a red box and an arrow pointing to the 'Table name' field. The 'Table name' field contains 'product'. The 'Data Source' is 'Foodmart'. A table of product data is displayed below.

product_class_id	product_id	brand_name	product_name	SKU	SRP	gross_weight	net_weight	recyclable_package	low_fat	units_per_case	cases_per
30	1	Washington	Washington Berry Juice	90748583674	2.9500	3.39	6.39	false	false	30	14
52	2	Washington	Washington Mango Drink	90616087498	0.7400	7.40	4.40	false	true	18	8
52	3	Washington	Washington Strawberry Drink	50427771925	0.3200	13.1	11.1	true	true	17	13
19	4	Washington	Washington Cream Soda	61412155717	3.6100	10.0	9.0	true	false	26	10
19	5	Washington	Washington Diet Soda	85551191433	2.1000	5.56	4.65	true	false	7	10
19	6	Washington	Washington Cola	29831612750	1.1000	15.8	13.8	false	false	14	10
19	7	Washington	Washington Diet Cola	30191444754	2.8100	18.0	17.0	true	false	11	7
30	8	Washington	Washington Orange Juice	09770532250	2.9500	3.97	6.97	true	false	27	7
30	9	Washington	Washington Cranberry Juice	41815180474	2.4000	7.14	5.14	false	false	34	7
30	10	Washington	Washington Apple Juice	22114064362	1.4200	3.10	7.10	true	false	28	14
52	11	Washington	Washington Apple Drink	11071288725	3.3100	20.0	19.0	false	true	3	10
35	12	Jeffers	Jeffers Oatmeal	40031098860	1.5400	3.0	6.80	true	true	14	5
33	13	Jeffers	Jeffers Corn Puffs	13229069505	2.9000	10.1	7.39	false	false	20	10
35	14	Jeffers	Jeffers Wheat Puffs	02940813038	1.9300	21.6	20.6	true	true	33	10
35	15	Jeffers	Jeffers Cereals	26378546923	2.2500	21.0	20.2	false	true	14	9

Fig. 5.20: Flat Dataset.

- **XLS numbers**: it is the number of sheets 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.

5.2.1.7.3 Federated

Important: Enterprise Edition

If you purchased Knowage EE, this feature 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 **Federation Definitions**.

5.2.1.7.4 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:

Listing 5.3: Request body code

```

1 { "entities": [ {
2   "isPattern": "true",
3   "id": ".*",
4   "type": "Meter"
5 } ]
6 }
```

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

Listing 5.4: RJSON response code

```

1 {
2   "contextResponses": [
3     {
4       "contextElement": {
5         "id": "pros6_Meter",
6         "type": "Meter",
7         "isPattern": "false",
8         "attributes": [
9           {
10            "name": "atTime",
11            "type": "timestamp",
```

(continues on next page)

(continued from previous page)

```

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",
54                 "type": "double",
55                 "value": "1.8"
56             },
57             {
58                 "name": "prosumerId",
59                 "type": "string",
60                 "value": "pros5"
61             },
62             {
63                 "name": "unitOfMeasurement",
64                 "type": "string",
65                 "value": "kW"
66             },
67             {
68                 "name": "upstreamActivePower",
69                 "type": "double",
70                 "value": "0"
71             }
72         ]
73     },
74     "statusCode": {
75         "reasonPhrase": "OK",
76         "code": "200"
77     }

```

(continues on next page)

(continued from previous page)

```
78     }
79     ]
80 }
```

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:

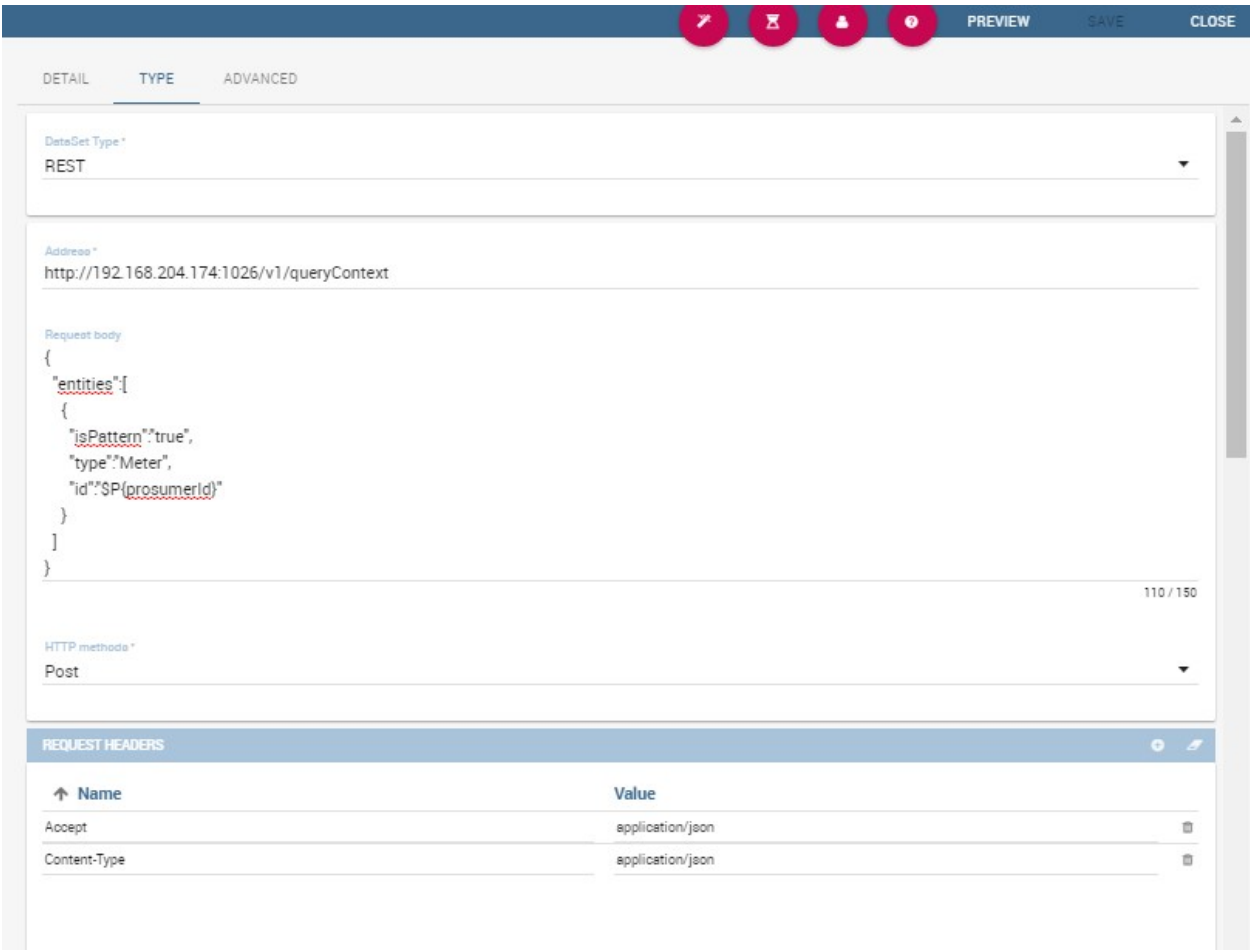


Fig. 5.21: REST dataset interface.

We specified

- the URL of the REST service;
- 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 specifications 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, you can skip the definition of the JSONPath attributes, since the JSON structure is presumed to be fixed as in the following example:

Listing 5.5: Use directly JSON attributes

```

1 {
2   "contextResponses": [
3     {
4       "prosumerId": "pros1",
5       "downstreamActivePower": 3.1,
6       "upstreamActivePower": 0.0
7     }, {
8       "prosumerId": "pros2",
9       "downstreamActivePower": 0.5,
10      "upstreamActivePower": 2.4
11     }
12   ]
13 }
```

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[:sub: `*\`]` and the dataset result will look like:

Table 5.2: Dataset result

prosumerId	downstreamActivePower	upstreamActivePower
pros1	3.1	0.0
pros2	0.5	2.4

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:

Listing 5.6: Request body for prosumerId parameter

```

1 {
2   "entities": [
3     {
4       "isPattern": "true",
5       "type": "Meter",
6       "id": "$P{prosumerId}"
7     }
8   ]
9 }
```

5.2.1.7.5 Python/R

The Python/R dataset enables users to create a dataset by writing a Python or R script that directly retrieves data. The developer of the dataset is free to write code which has to produce a **dataframe** variable as output (for Python scripts we refer to pandas dataframes). This variable will contain data that Knowage will later convert into its own format.

The screenshot shows the 'Python/R' dataset configuration interface. It includes a top navigation bar with icons for edit, timer, user, and help, and buttons for PREVIEW, SAVE, and CLOSE. The main interface has three tabs: DETAIL, TYPE (selected), and ADVANCED. The configuration fields are as follows:

- DataSet Type:** A dropdown menu set to 'Python'.
- Python environment:** A dropdown menu set to 'python.virtualenv.1.backend.url'.
- Dataframe variable name:** A text field containing 'df', with a red arrow pointing to it.
- Script:** A text area containing the following Python code:


```
1 import pandas as pd
2 df = pd.DataFrame({'animal': ['falcon', 'dog', 'spider', 'fish'], 'num_legs': [2, 4, 8, 0], 'num_wings': [2, 0, 0, 0], 'num_specimen_seen': [10, 2, 1, 0]})
```

 A red arrow points to the first line of the script.

Fig. 5.22: Python/R dataset interface.

As shown in the picture in the field **Dataframe variable name** the developer has to specify the name of the variable in which the final output of the script is stored in the form of a dataframe.

In the field **Python environment** the user can select a working environment among the available ones defined in the **Configuration Management** section.

Inside the scripts it is possible to use parameters by the usual syntax `$P{}`.

5.2.1.7.6 Big Data - NoSQL

Important: Enterprise Edition

If you purchased Knowage EE, this feature is available only in KnowageBD and KnowagePM

Knowage provides the possibility to define Big Data dataset as well as Big Data datasources. 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 according 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

Listing 5.7: Request body for prosumerId parameter

```
1 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 column 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 below.

Table 5.3: Dataset output

a	b
2	3

- 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 instance, let's consider the query `sbiDatasetfixedResult = [{a:2, b:3},{a:2, c:3}]` the dataset is

Table 5.4: Dataset output

a	b	c
2	3	
2		3

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:

```
1 var retVal= "LIST_DOCUMENTS_QUERY";
```

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

```
1 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.

Note: MongoDB Document size

Remember that MongoDB has a limit of maximum 16MB for the returned document (BSON), so pay attention to that when creating your dataset. For more information check this link: <https://docs.mongodb.com/manual/reference/limits/>

5.2.1.7.7 Solr Dataset

A dataset of type Solr, see the following figure, reads data from the popular Search Engine Solr. To define a **Solr Dataset** select the Solr type, then choose between Document or Facets type.

The screenshot shows a configuration form for a Solr dataset. It includes fields for 'DataSet Type' (set to Solr), 'Address' (http://10.0.0.17:8983/solr/), 'Collection' (foodmart_test_01_p), and 'Query' (*:*). Below these fields are radio buttons for 'Documents' (selected) and 'Facets'. At the bottom, there is a section for 'Documents' with a 'Field list (comma-separated values)' containing the text: the_date,quarter,store_cost,product_family,store_id,unit_sales.

Fig. 5.23: Solr Dataset, Document type selected.

The **Query** field is the Solr query using the Solr standard query syntax. The **Collection** field is the **core**, in Solr, the term core is used to refer to a single index and associated transaction log and configuration files (including the solrconfig.xml and Schema files, among others). Your Solr installation can have multiple cores if needed, which allows you to index data with different structures in the same server, and maintain more control over how your data is presented to different audiences. In SolrCloud mode you will be more familiar with the term collection. Behind the scenes a collection consists of one or more cores.

Documents

According to the Solr official documentation, Solr's basic unit of information is a document, which is a set of data that describes something. A recipe document would contain the ingredients, the instructions, the preparation time, the cooking time, the tools needed, and so on. A document about a person, for example, might contain the person's name, biography, favorite color, and shoe size. A document about a book could contain the title, author, year of publication, number of pages, and so on.

In the Solr universe, documents are composed of fields (these fields can be put into section **document field list**), which are more specific pieces of information. Shoe size could be a field. First name and last name could be fields. If you have chosen the type "Documents", you can add the document fields to the list below called "Documents".

Request header, if there is the need, it is possible to customize the request header of the post http request, adding optional parameters.

Filter Query parameter

Name	Value
annoemissione	SP{anno}

Parameters

Name	Type	Default Value	Multivalue
anno	Number	2015,2014	<input checked="" type="checkbox"/>

Fig. 5.24: Solr Dataset, Optional fields for filtering parameters.

Solr dataset can also use Profile Attributes. The syntax to include attributes into the dataset text is `${attribute_name}`. Profile attributes can be single-value or multivalue.

The **filter query parameter** is the Solr fq parameter and defines a query that can be used to restrict the superset of documents that can be returned, without influencing score. It can be very useful for speeding up complex queries, since the queries specified with fq are cached independently of the main query. These parameters can be used in combo with **document parameters** using the `P{ }` notation like the example picture shows.

Fields Mapping

It is important to set field types correctly in order to use a Solr dataset without problems. A field type defines the analysis that will occur on a field when documents are indexed or queries are sent to the index.

A field type definition can include four types of information:

- The name of the field type (mandatory).
- An implementation class name (mandatory). If the field type is a number and it has decimals it must be set as `pdouble` (not `int` or `string`!).
- If the field type is `TextField`, a description of the field analysis for the field type.
- Field type properties - depending on the implementation class, some properties may be mandatory.

Example: `<field name="REG_T_MP" type="pdouble" indexed="true" required="false" stored="true" multiValued="false"/>`

Faceting

Faceting is the arrangement of search results into categories based on indexed terms. If you choose Facets you can add the Facet Query. This parameter allows you to specify an arbitrary query in the Lucene default syntax to generate a facet count. The Facet Field is the facet.field parameter and identifies a field that should be treated as a facet. It iterates over each Term in the field and generate a facet count using that Term as the constraint. This parameter can be specified multiple times in a query to select multiple facet fields. The Facet Prefix is the facet.prefix parameter limits the terms on which to facet to those starting with the given string prefix. This does not limit the query in any way, only the facets that would be returned in response to the query.

☐ Documents
 ☒ Facets

Facets

Facet Query ?

Facet Field (comma-separated values)

Facet Prefix

5.2.2 Parameters and profile attributes

All dataset types except **File** and **CKAN** allow you to add parameters. This means that results can be customized according to the value of one or more parameters at execution time. Parameters can be managed from the **Type** tab. Two operations are needed to add a parameter to the dataset:

1. insert the parameter in the actual text of the dataset;
2. create the parameter in the parameters list below the editor area.

The syntax to add a parameter in the dataset code text is `$P{parameter_name}`. At dataset execution time, the parameter will be replaced by its actual value.

Warning: Attention to parameters' names!

If the dataset is used by a Knowage document, then the document parameters' URL must match the parameter name set in the dataset **Type** tab, in order for the dataset to be passed correctly.

Any parameter added to your dataset must be added to the parameters list, too. To add a parameter in the list, click the **Add** button. A new row will be created in the list: double click the name and edit the parameter values. There are three different types of parameters. For each of them the placeholder will be replaced according to a different pattern, as follows:

- **String:** the parameter value will be surrounded with single quotes if not already present.
- **Number:** the parameter value is treated as a number, with no quotes; an exception is thrown if the value passed is not a number.

- **Raw:** the parameter value is treated as a string containing a set of values; single quotes are removed from the containing string, not from the single strings composing it.
- **Generic:** the parameter is simply passed as it is, with no further processing.

In SQL query example with parameters an example is provided, where `MediaType` is a string parameter.

Listing 5.8: SQL query example with parameters

```

1 SELECT s.customer_id as CUSTOMER
2 , sum(s.store_sales) as SALES
3 , c.yearly_income as INCOME
4 , p.media_type as MEDIA
5 FROM sales_fact_1998 s, customer c, promotion p
6 WHERE
7 s.customer_id=c.customer_id and s.promotion_id=p.promotion_id and
8 p.media_type in (${MediaType})
9 GROUP BY
10 s.customer_id,
11 c.yearly_income,
12 p.media_type

```

Datasets of type Query and Script can also use *profile attributes*. Differently from parameters, profile attributes do not need to be explicitly added to the parameter list since they have been defined elsewhere. Clicking the **Available Profile Attribute** button you can see all profile attributes defined in the behavioural model and choose the one(s) you wish to insert in the dataset query/script text, as shown below.

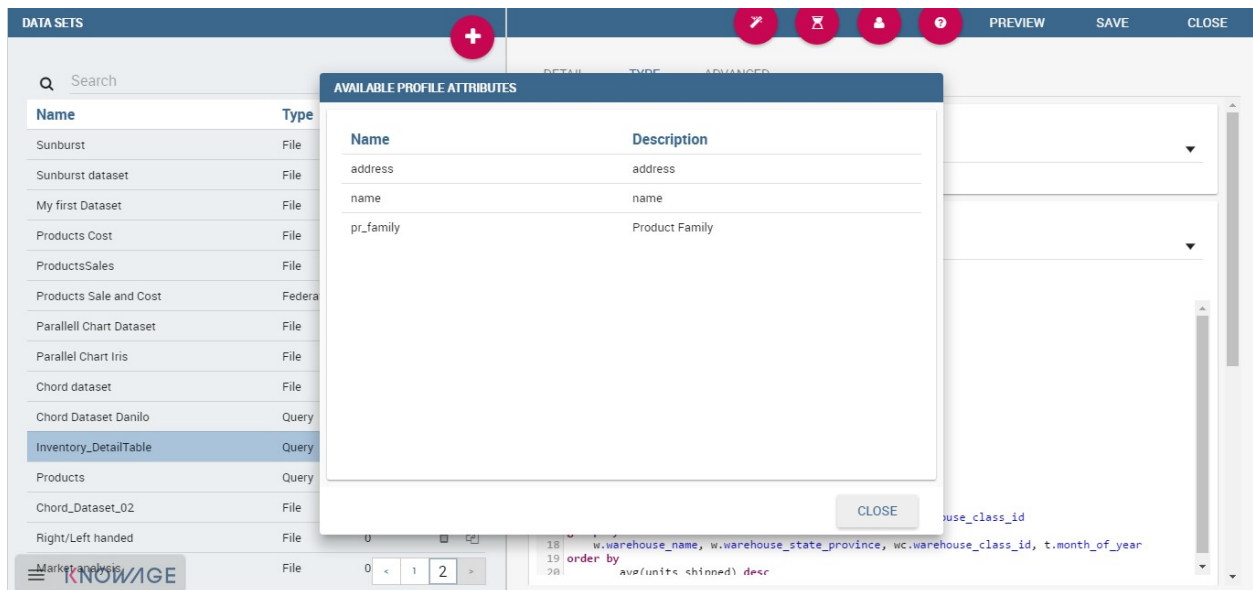


Fig. 5.25: Profile Attributes assignment.

The syntax to include attributes into the dataset text is `${attribute_name}`. Profile attributes can be single-value or multivalued.

Note: User profile attributes

Each Knowage user is assigned a profile with attributes. The user profile is part of the more general behavioural model, which allows tailored visibility and permissions on Knowage documents and functionalities.

5.2.3 Further operations on a dataset

5.2.3.1 Script option

As we reported in Section ‘Query Dataset’, the script option can be very useful when the user wants to create a very dynamic query. Dealing with parameters, if the query syntax is not handled properly, the missing of one parameter value may compromise the dataset execution itself. In particular, it can be convenient to use a script to manage the assignment of null or empty values to parameters in those cases when the user wants the filters not to be applied.

Knowage query dataset are endowed of a specific area to insert the script syntax. Clicking on the “Script” button we reported in section Query Dataset’, the interface opens a wizard containing two tabs: the script tab is the one opened by default. Here the user is asked to select the language he/she’s intended to use.

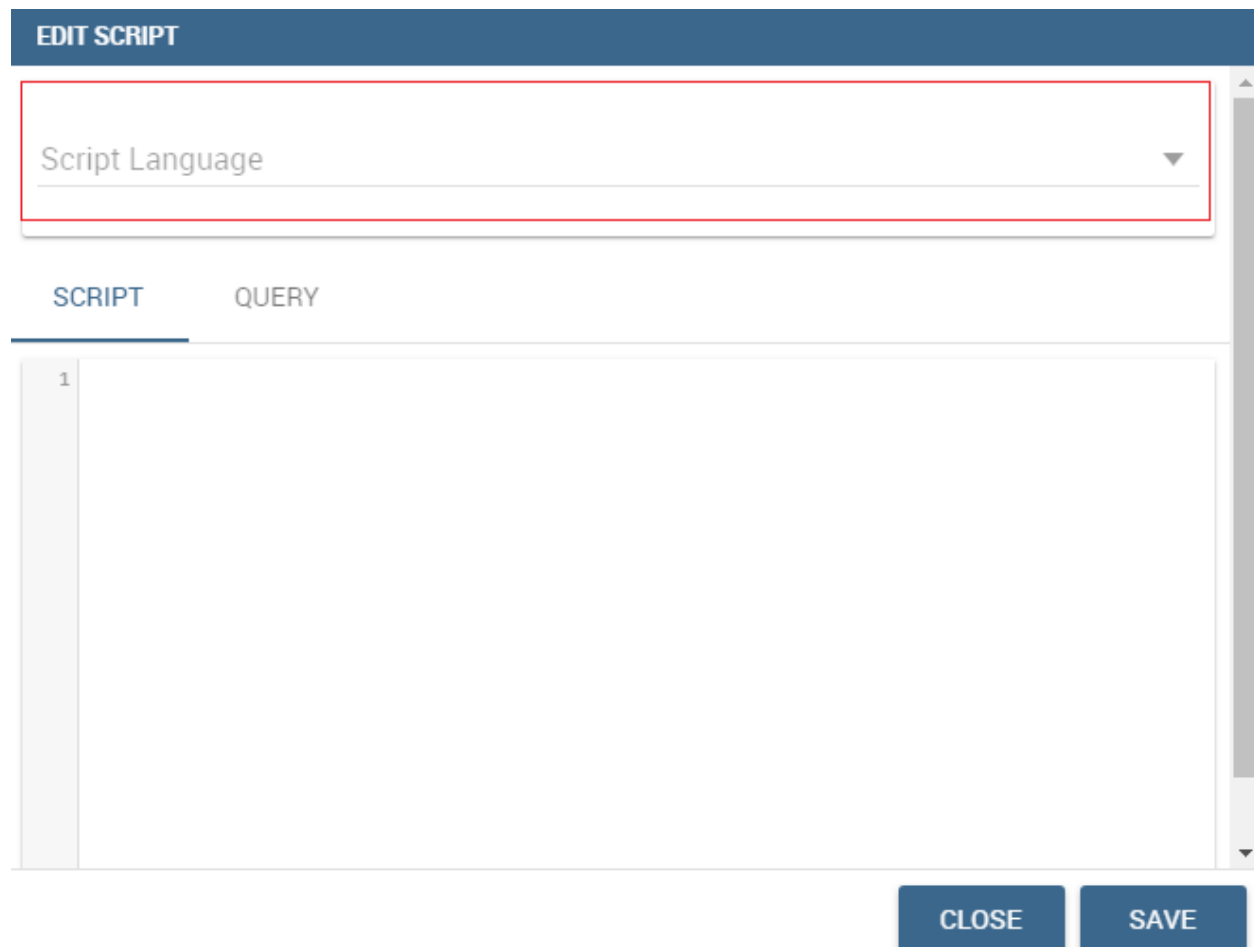


Fig. 5.26: Editing script.

Typically, scripts are configured to load placeholders with a slice of SQL code. Referring to the following pictures, we show an example of JavaScript (JS) code usage. Moving to the “Query” tab the user has to insert a placeholder where he/she’s expecting a certain clause to be added. The query will then look like the one shown below.

Moving to the “Script” tab instead, the user has to declare how the server has to manage the placeholder. The following picture shows a JS block code where the user first initializes a variable as empty: if certain conditions, on one or more parameters, are satisfied, the variable is assigned an SQL code string. Then, the JS method “.replace” will substitute the placeholder with the content the variable.

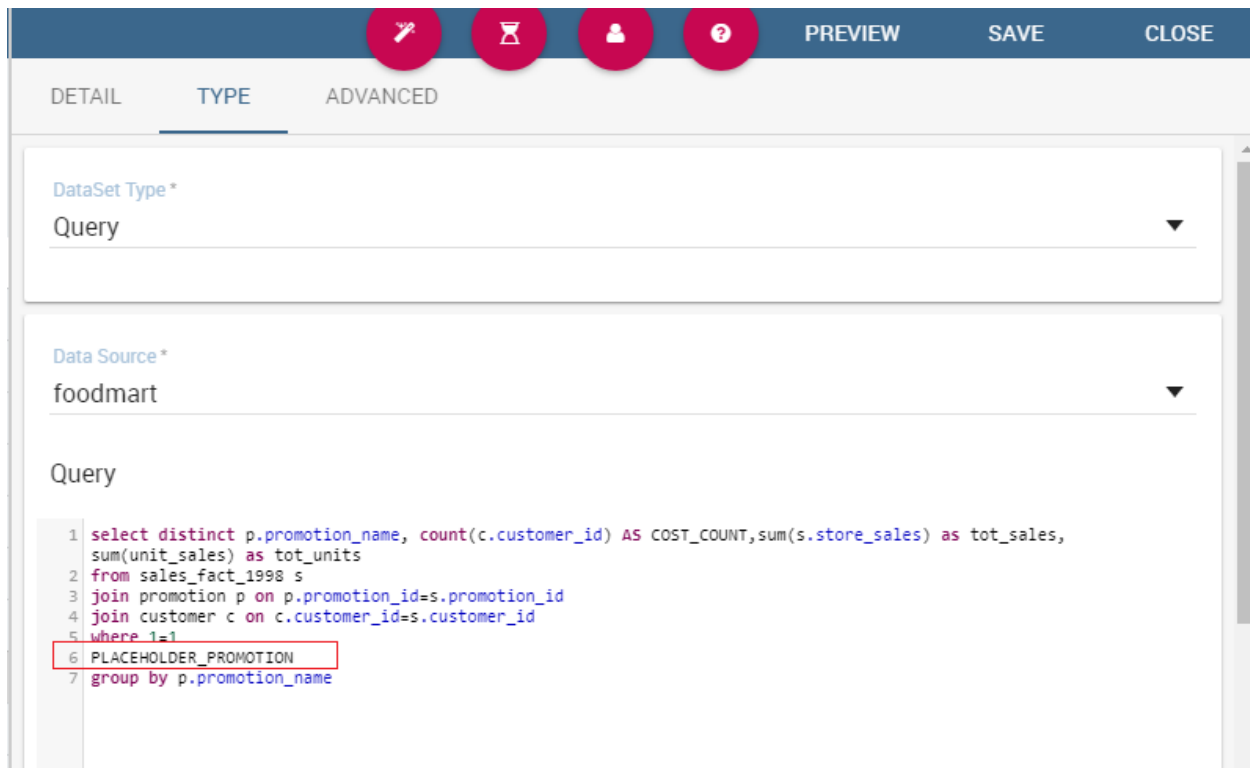


Fig. 5.27: Setting placeholder using script.

To sum up, the example reveals that if the parameter is assigned a null or empty value, conditions are not satisfied and the placeholder is substituted with an empty space (therefore nothing is added to the “where” clause). Otherwise, the SQL code is inserted into the “where” clause and the dataset is accordingly filtered.

We stress that it is not necessary to use any concatenation or JS method to recall for parameters’ values. It is enough to use the syntax `$P{par_name}` as well as seen when configuring a plain parametric dataset.

5.2.3.2 Transformations

In some cases it is useful to perform transformations on the results of a dataset, to obtain data in the desired format. The most common operation is the pivot transformation, which allows the switch between rows and columns of the dataset results. Knowage supports this operation on any type of dataset.

To set a pivot transformation, select **Pivot Transformer** in the drop down menu of the **Transformation** tab. Then set the following fields:

- **Name of Category Column to be Pivoted.** Here you should write the name of the dataset column whose values will be mapped onto columns after pivoting.
- **Name of Value Column to be Pivoted.** Here you should write the name of the result set column, whose values should become values of the previous columns (category columns).
- **Name of the Column not to be Pivoted.** Here you should write the name of those columns that should not be altered during the transformation.
- In case you wish to add a number to category columns (e.g., `1_name_of_column`), you should check the option **Automatic Columns numeration**.

An example of usage is available in figure below, showing the result set of the dataset.

EDIT SCRIPT

Language

Javascript

SCRIPT

QUERY

```
1 var prom_name = " ";
2
3 if (parameters.get('par_prom_name')!=null && parameters.get('par_prom_name')!="") {
4   prom_name = " and p.promotion_name in (${P{par_prom_name}}) ";
5 }
6 query = query.replace("PLACEHOLDER_PROMOTION", prom_name );
7
8
```

CLOSE

SAVE

Fig. 5.28: Editing script.

The image shows the configuration and preview of a Pivot Transformation in Knowage. The configuration window is titled 'ADVANCED' and shows the following settings:

- Transformation Type: **PIVOT_TRANSFORMER** (checked)
- Name of Category Column to be Pivoted: **MEMBER_CARD**
- Name of Value Column to be Pivoted: **UNI_SALES**
- Name of the Column not to be Pivoted: **OCCUPATION**

Two dataset preview windows are shown below the configuration window, illustrating the input and output data.

Left Dataset Preview (Input):

MEMBER_CARD	OCCUPATION	NUM	UNI_SALES
Bronze	Clerical	5586	18831.5505
Bronze	Management	39402	131748.9718
Bronze	Manual	43225	144411.0977
Bronze	Professional	89563	299125.2599
Bronze	Skilled Manual	48741	163198.1508
Golden	Clerical	1095	3656.2101
Golden	Management	9252	32110.6860
Golden	Manual	10007	35141.5418
Golden	Professional	19389	66208.7108
Golden	Skilled Manual	10227	35292.5362
Normal	Clerical	177	586.4950
Normal	Management	2572	8729.8460
Normal	Manual	38150	127030.5780
Normal	Professional	6759	22540.3288
Normal	Skilled Manual	44591	150067.6726

Right Dataset Preview (Output):

OCCUPATION	NUM	Bronze	Golden	Normal	Silver
Clerical	5506	18831.5505	0	0	0
Management	39402	131748.9718	0	0	0
Manual	43225	144411.0977	0	0	0
Professional	89563	299125.2599	0	0	0
Skilled Manual	48741	163198.1508	0	0	0
Clerical	1095	0	3656.2101	0	0
Management	9252	0	32110.6860	0	0
Manual	10007	0	35141.5418	0	0
Professional	19389	0	66208.7108	0	0
Skilled Manual	10227	0	35292.5362	0	0
Clerical	177	0	0	586.4950	0
Management	2572	0	0	8729.8460	0
Manual	38150	0	0	127030.5780	0
Professional	6759	0	0	22540.3288	0
Skilled Manual	44591	0	0	150067.6726	0
Clerical	5506	0	0	0	1725.0694
Management	39402	0	0	0	31223.3566

Fig. 5.29: Pivot transformation.

5.2.3.3 Dataset persistence

The **Advanced** tab is used to make a dataset persistent, 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. In order to force recalculation of the dataset, you should execute dataset preview again. This will store the newly generated data on the database table.

Once marked the dataset as persistent, you are asked to insert a table name. This is the table where data are stored and then retrieved.

Important: Enterprise Edition only

With KnowageBD, KnowageER and KnowageSI products you can also decide to schedule the persistence operation: this means that the data stored in the table will be update with 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.

5.2.3.4 Preview

Before actually using the dataset in a document, it is a good practice to test it. Clicking the **Preview** button within the **Preview** tab, you can see a preview of the result set, see the following figure. This allows the developer to check any anomaly or possible error in the dataset definition, before using it.

The figure shows two side-by-side screenshots from the Knowage application. The left screenshot, titled 'DATASET PREVIEW', displays a table with 27 rows of customer data. The right screenshot, titled 'SET PARAMETER VALUES', shows a form for setting parameter values, with a 'Name' field containing 'par_gender' and a 'Value' field with a placeholder text 'If not set, parameter will have default value'.

customer_id	account_num	lname	fname	mi	address1	address2	address3	address4	city	state_province	postal_code	country	customer_req
1	87463034688	Noumer	Shel	A	2433 Bailey Road				Tlaxiaco	Oaxaca	15058	Mexico	30
3	87473737600	Derry	Jeanne		7640 First Ave				Issaquah	WA	77980	USA	21
5	87634014179	Gutierrez	Maya		8668 Via Semesta				Novato	CA	94955	USA	42
6	87637703849	Demetrie	Robert	F	1619 Oldham Court				Lynwood	WA	90762	USA	75
7	87623117880	Kanagaki	Patricia		2860 O Mt. Hood Drive				Tlaxiaco	Oaxaca	11343	Mexico	30
12	87675237222	Crawk	Jewel	C	1792 Bellevue Rd.				Chula Vista	CA	92020	USA	13
14	87692202810	Rutledge	Bryan	K	3075 Avista Drive				Lincoln Acres	CA	92546	USA	10
16	87603205908	Planck	Peggy	M	4864 San Carlos				Camacho	Zacatecas	77787	Mexico	27
19	87605814652	Collins	Dianne		551 Rammer Dr				Oakland	CA	94606	USA	37
20	87653507000	Baker	Beverly		591 Menewood Drive				Spring Valley	CA	95762	USA	15
22	87675561200	Borgos	Laurie		1873 Lyon Circle				Bellingham	WA	78588	USA	78
23	87676304800	Wynn	Shawna	K	9174 Norris Drive Ave.				La Jolla	CA	92039	USA	33
24	87687718700	Myllis	Jacqueline		6318 Maribel Dr				Santa Fe	DF	90737	Mexico	109
25	87686740150	Gortley	Lili	N	7814 Millburn Dr				Hawley	BC	88710	Canada	53
27	87701010180	Maccaluso	Charles	B	5678 Midway Court				Anacortes	WA	98052	USA	79

Fig. 5.30: Dataset preview (left) and parameters prompt window (right).

If some parameters have been set, a window with their list will be shown: their values must be entered by double clicking on the set to string, just write the value you want to assign in the preview: quotes will be added automatically. On the other hand, if the type is raw or generic but you want to input text, then remember to add quotes to the test value.

5.3 Behavioural Model

An important topic to face before starting a new project is how to create and manage the so-called *behavioural model*.

The *behavioural model* regulates the visibility on documents and data according to the roles and profiles of the end users.

It mainly answers the following questions:

- WHO uses the business intelligence solution (user profile);
- WHAT is visible to users, in terms of documents and data (repository rights and analytical drivers);
- HOW users work with their documents (analytical drivers and presentation environment settings).

The creation and the management of the behavioural model is in charge of Knowage Administrator. However when it has been designed and built, it has to be shared with developers as well. Indeed in developing phase you have to be aware of the visibility hierarchy. You need these information to set document options correctly.

5.3.1 Roles, users and attributes

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.

You will not have grants to create new roles or users, but you are asked to match them during document profilation phases. In the following we are going to describe the elements needed for adding parameters. This elements involves profilation too. To conclude we will see how to manage accessibility while creating a document.

5.3.2 Analytical drivers

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.

The Figure above represents a report with two parameters:

- the Department, 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*.



Fig. 5.31: Parametric Report.

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 below.

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.

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. Next figure 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.

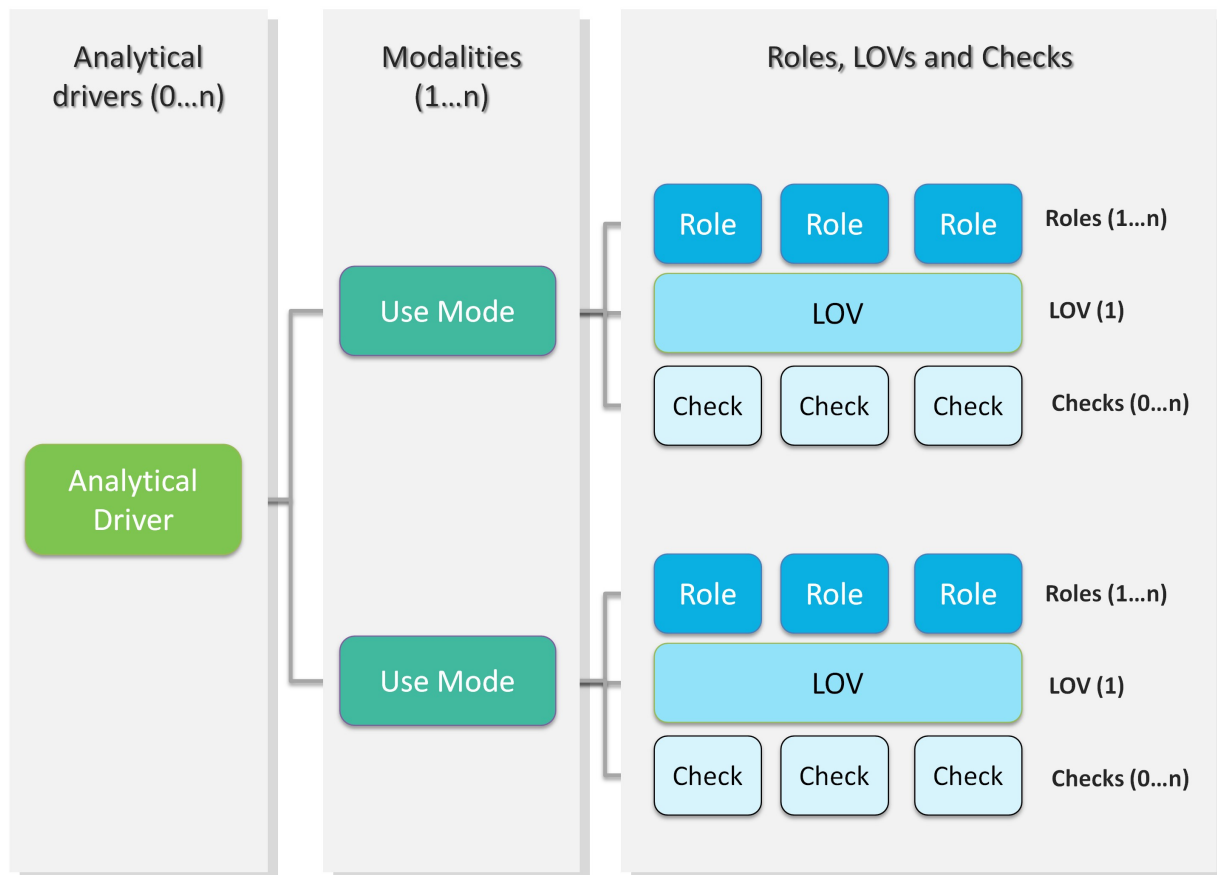


Fig. 5.32: Analytical driver schema.

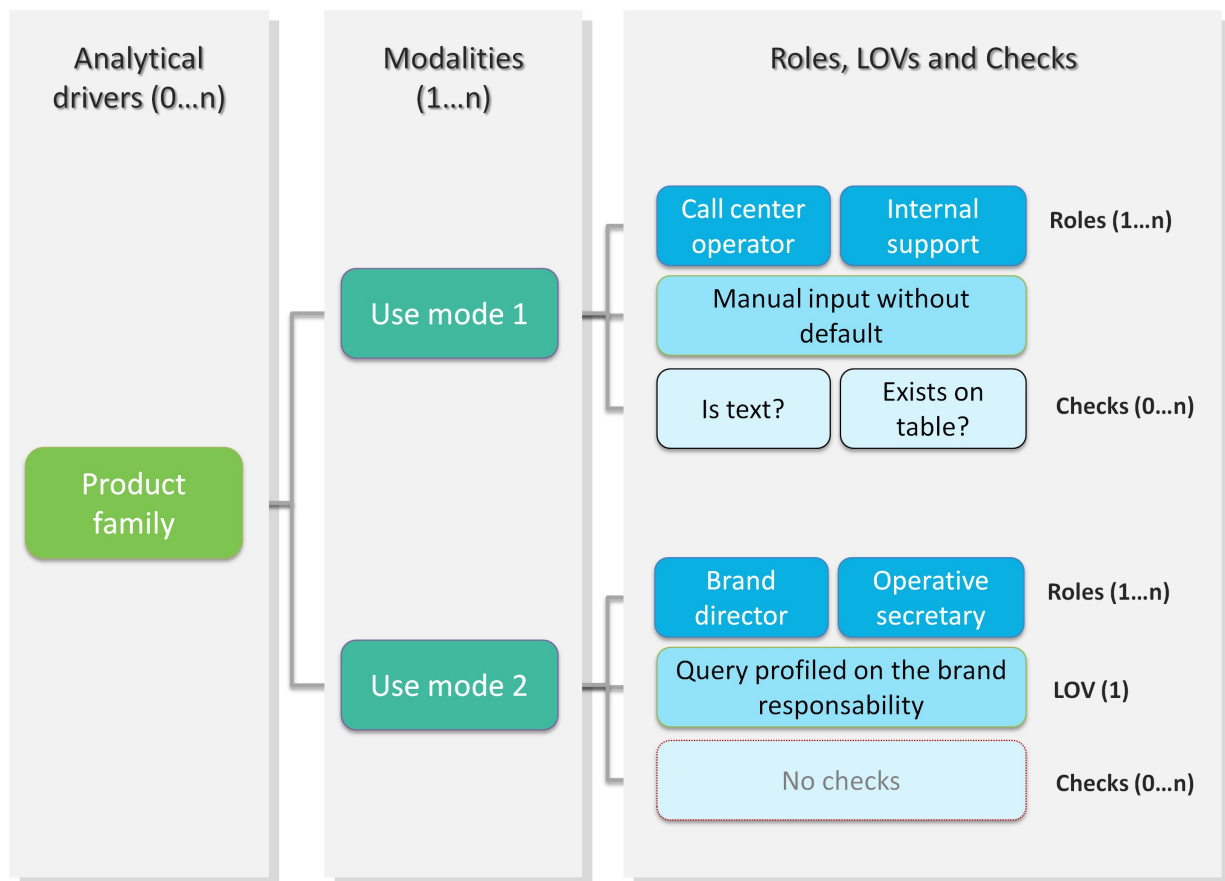


Fig. 5.33: Analytical driver schema - Example.

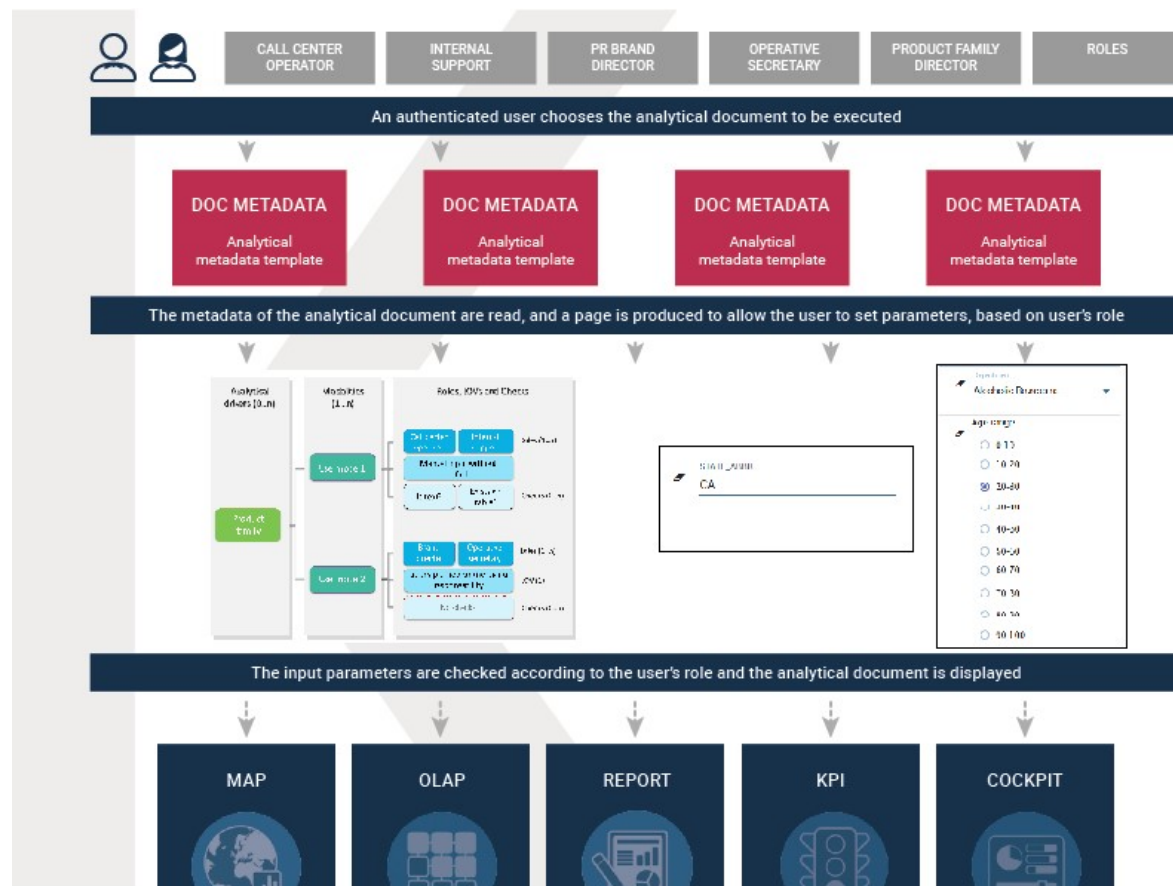


Fig. 5.34: Overall process.

5.3.2.1 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 LOV example retrieves id, name and food family for a product.


Listing 5.9: LOV example


```
1 {195, High Top Almonds, Food};
2 {522, Tell Tale Walnuts, Food};
3 {844, Very Good Soda, Drink};
```

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.

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 above. 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.

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. The two figures below exhibit an example. Columns that are not visible can be used for correlating drivers.

Note: 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. Read more at *Analytical document* chapter.

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 next figure) 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.

Note: Create a LOV for the default value of an analytical driver of type Manual Input Date

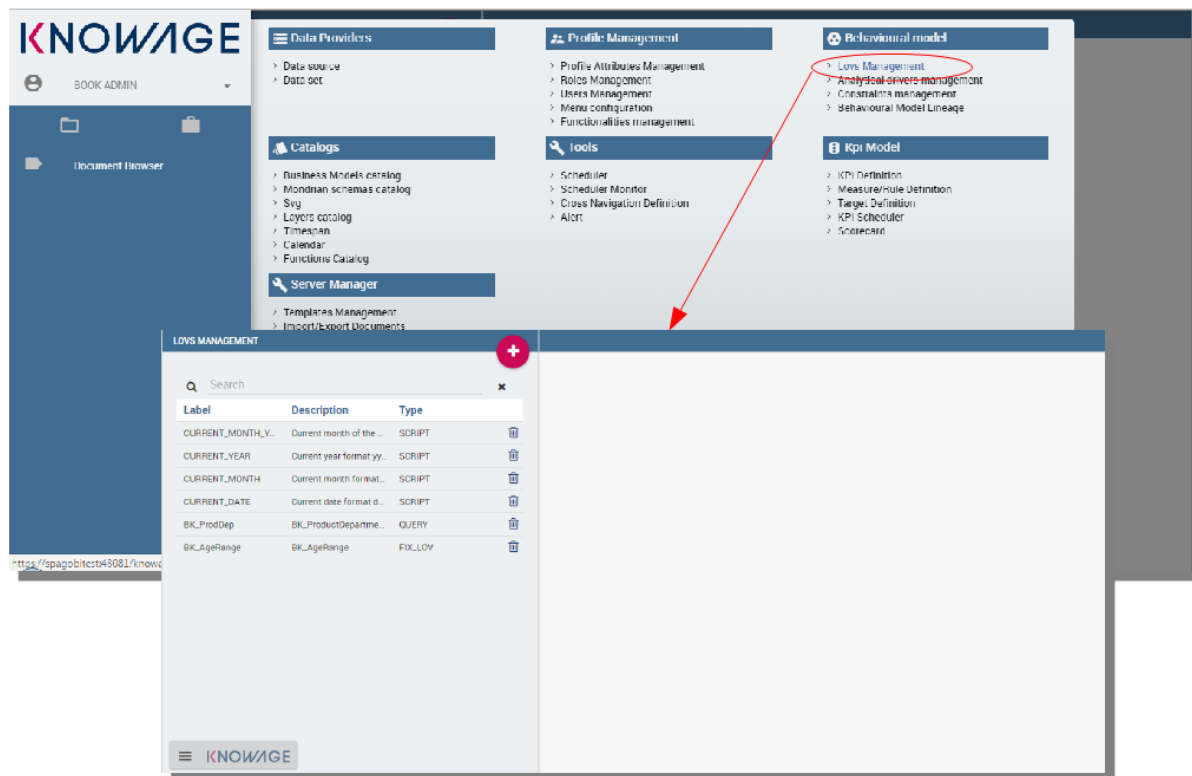


Fig. 5.35: LOV list.

The figure displays three overlapping screenshots of the LOV (List of Values) Creation interface in Knowage.

Top-Left Window (BK_AGE RANGE):

- Buttons: PREVIEW, TEST, SAVE, CLOSE
- Label*: BK_AgeRange
- Name*: BK_AgeRange
- Description: BK_AgeRange
- LOV Type*: Fixed list of values
- WIZARD FOR LOV:
 - Value: [Empty field]
 - Description: [Empty field]
 - SAVE button
 - Table with 2 columns: Value, Description

Value	Description
0-10	0-10
10-20	10-20
20-30	20-30
30-40	30-40

Middle Window (BK_PRODUCT DEPT):

- Buttons: PREVIEW, TEST, SAVE, CLOSE
- Label*: BK_ProductDept
- Name*: BK_ProductDepartment
- Description: BK_ProductDepartment
- LOV Type*: Query statement
- WIZARD QUERY:
 - Label*: [Empty field]
 - Description: [Empty field]
 - Query definition:


```
select distinct product_desc from product
```

Bottom-Right Window (CURRENT DATE):

- Buttons: PREVIEW, TEST, SAVE, CLOSE
- Label*: CURRENT DATE
- Name*: Current date
- Description: Current date format dd/mm/yyyy
- LOV Type*: Script to load values
- WIZARD QUERY:
 - Language*: [Empty field]
 - Script:


```

1
2
3 Date now = new Date();
4 var day = new Date();
5 var month = new Date();
6 if (day > 31) day = 31;
7 var month = new Date();
8 if (month > 12) month = 12;
9 var year = new Date();
10 let dayStr = day.toString().padStart(2, '0');
11 let monthStr = month.toString().padStart(2, '0');
12 let yearStr = year.toString().padStart(4, '0');
13 return dayStr + '/' + monthStr + '/' + yearStr;
14
          
```

Fig. 5.36: LOV Creation interface.

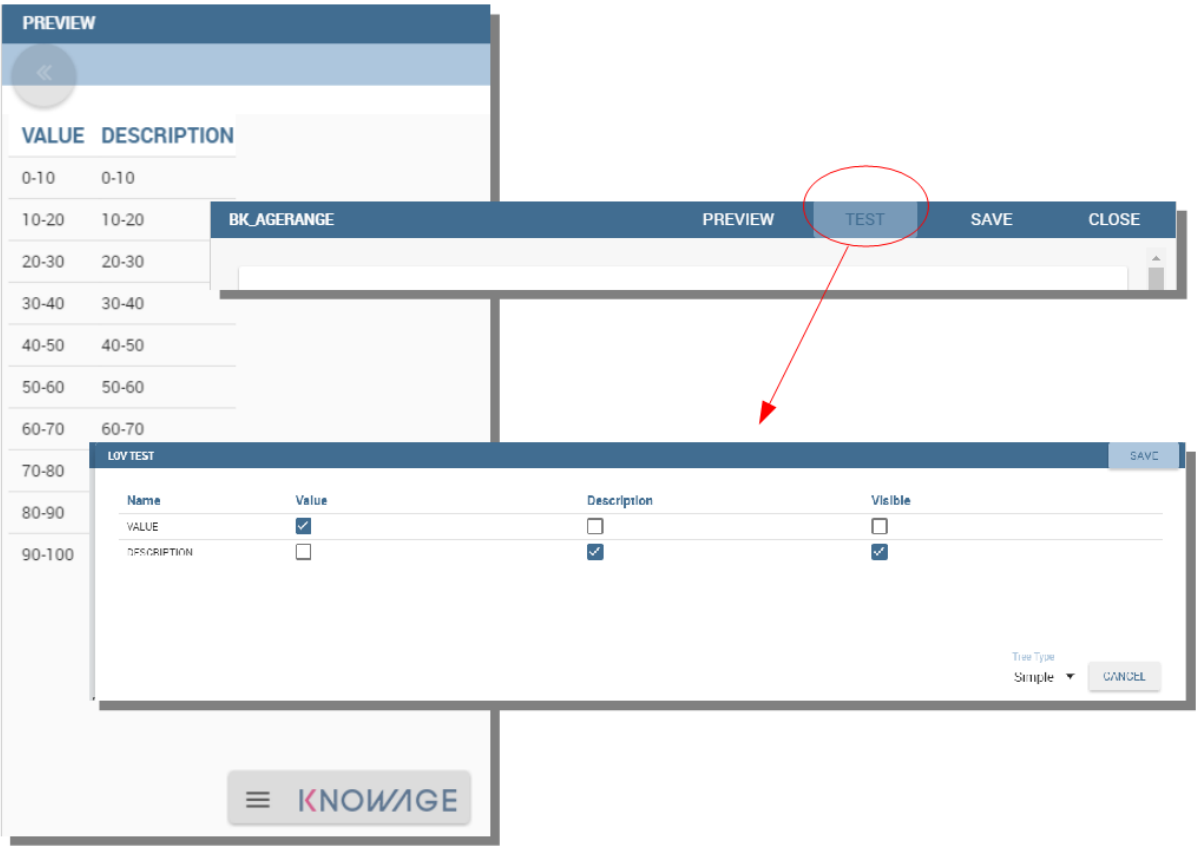


Fig. 5.37: Preview and Test of the LOV.

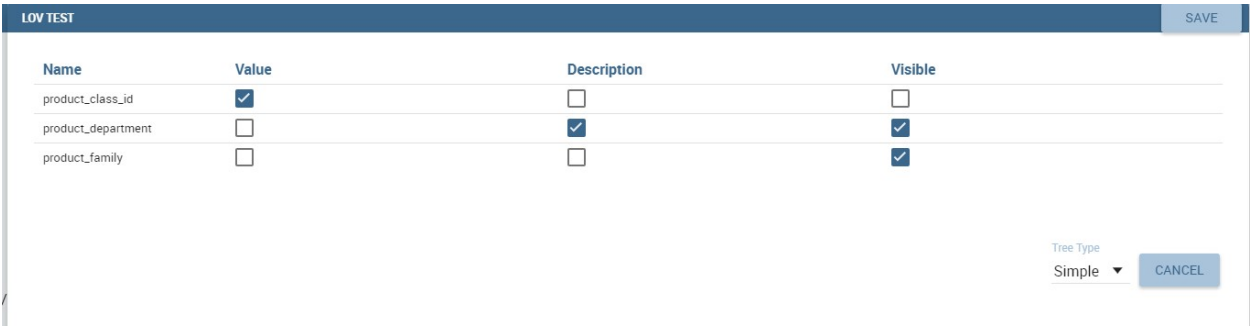


Fig. 5.38: Preview and Test of the LOV.

LOW TEST

SAVE

Name	Value	Description	Visible
VALUE	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
DESCRIPTION	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

tree type

Simple

CANCEL

Simple

Tree

Tree selectable inner nodes

Fig. 5.39: Hierarchical LOV definition.

This note is useful when using an analytical driver of type Date with an input of type Manual. In the case you want to use a particular date as default value for that driver, you have to use this syntax for the LOV: select '2017-09-10#yyyy-MM-dd' as fixed_date. Instead of the fixed date 2017-09-10 you can also use as default date the today date for example; in this case you can use a query of this type: select concat(to_date(now()), '#yyyy-MM-dd') as today. The most important thing is to concat to the default date you want to use the string #yyyy-MM-dd.

Note: Create a LOV for the default value of an analytical driver with a hierarchical LOV

In case you want to add a default value to an analytical driver with an input of type hierarchical LOV you need to use another hierarchical LOV with the default values desired. If the analytical driver LOV is of type *Tree* then the default LOV need to be of type *Tree* too. The LOV need to have values for the leaf level only. Otherwise, if the analytical driver LOV is of type *Tree selectable inner nodes* the default LOV need to be of the same type. The default LOV may have values for one of the level used in the hierarchical LOV. For example, suppose you have an analytical driver with a hierarchical LOV having levels Product Family > Product Category > Product Department. If the hierarchical LOV is of type *Tree* then in the default LOV you need to insert one or more values for the level Product Department. Your default LOV have one level, the Product Department. In case the LOV is of type *Tree selectable inner nodes* you can choose one of the three levels. Your default LOV have one level between Product Family, Product Category or Product Department.

5.3.2.2 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 Query LOV example:

Listing 5.10: Query LOV example

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

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 standard syntax `${profile_attribute_name}`, as shown in Parametric query.

Listing 5.11: Parametric query

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

Then, at LOV execution time, for the user food manager the query becomes as shown in Runtime placeholder substitute and hence the corresponding LOV will return only the brand names related to the Food product family.

Listing 5.12: Runtime placeholder substitute

```

1 SELECT DISTINCT PRODUCT_FAMILY, BRAND_NAME
2 FROM PRODUCT
3 WHERE C.PRODUCT_FAMILY = 'Food'

```

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: Standard profile attributes

There are some standard profile attributes always available that don't need to be defined for each user. These profile attributes are:

- `user_id` contains the user id of the logged in user
- `user_roles` contains the current user's roles, joined as a SQL IN clause fashion, for example: 'general_management','human_resources_management'
- `TENANT_ID` contains the tenant to which the user belongs

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



Fig. 5.40: Assistance in retrieving syntax and profile attributes.

5.3.2.3 Creating a validation rule

Knowage supports the validation of the document's input parameters via validation rules. 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.

Knowage default checks are:

- **Alphanumeric**: it checks if the parameter is alphanumeric;

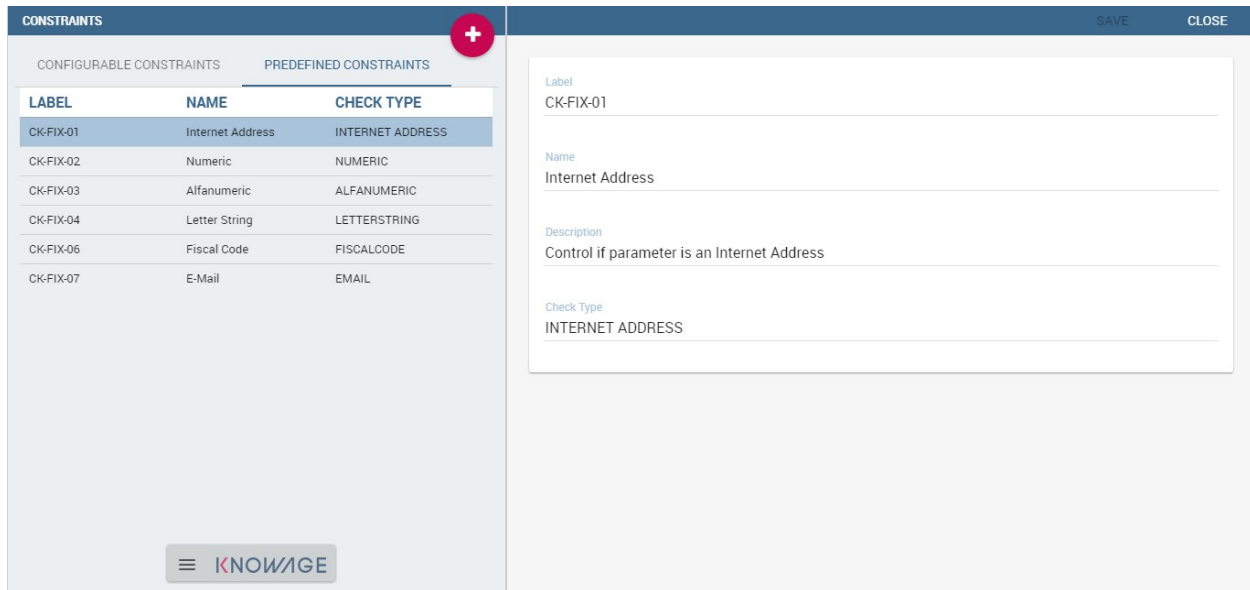



Fig. 5.41: Constraints Management.

- **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 customized 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.

5.3.2.4 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 menu item from the list and they will appear in the half right side, as shown in the

figure above. 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.

The screenshot displays the 'ANALYTICAL DRIVERS LIST' on the left and the 'BK_ProdDep' details on the right. The list table contains two entries: 'BK_ProdDep' and 'BK_AgeRange', both of type 'STRING'. The details panel for 'BK_ProdDep' shows the following configuration:

Field	Value
Label *	BK_ProdDep
Name *	BK_ProductDepartment
Description	BK_ProductDepartment
Type	String (selected)
Functional	<input checked="" type="checkbox"/>
Temporal	<input type="checkbox"/>

Fig. 5.42: Analytical Driver Management.

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 the figure below. 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 button, enabled as soon as the form is filled in, will save the driver and let the section below appear.

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 below.

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 displayed and selectable when executing the document. In fact the user can choose among:

- **Select from list:** all admissible values will be displayed directly within the drivers panel;
- **Select from popup window:** user will be able to select between admissible values by a lookup table displayed within a popup window;
- **Select from tree:** conceived for hierarchical LOVs, lets the users navigate the values in a hierarchical way;
- **Select from combobox:** the driver 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 supplied in next sections 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).

In case of Manual Input Date the user can specify a maximum value driven by a LOV:

NEW USEMODE
SAVE
CLOSE

ANALYTICAL DRIVERS DETAILS

Label *
This is required

Name *
This is required

Description

Type

☒ Date
☐ Number
☐ String
☐ Date Range

☒ Functional

☐ Temporal

Fig. 5.43: Driver creation.

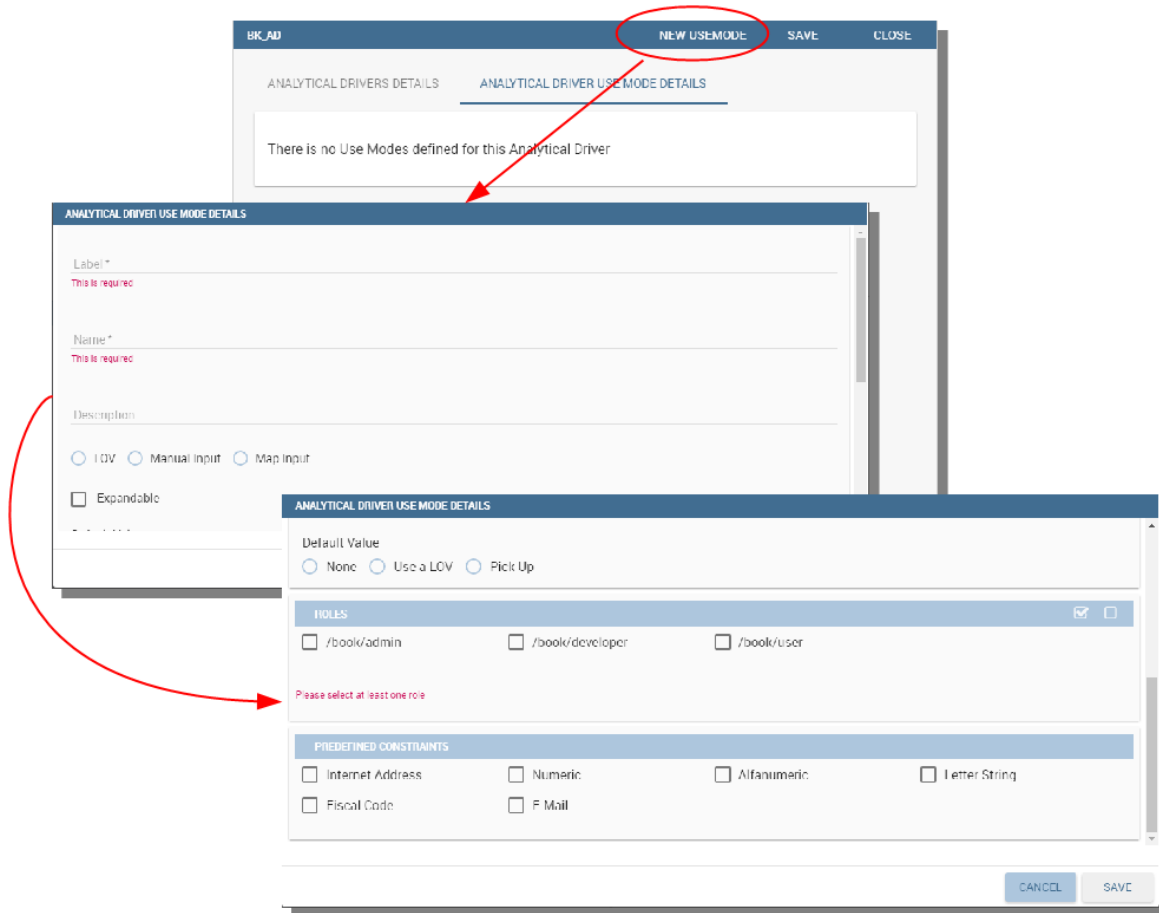


Fig. 5.44: Detail panel of LOV creation, second step.

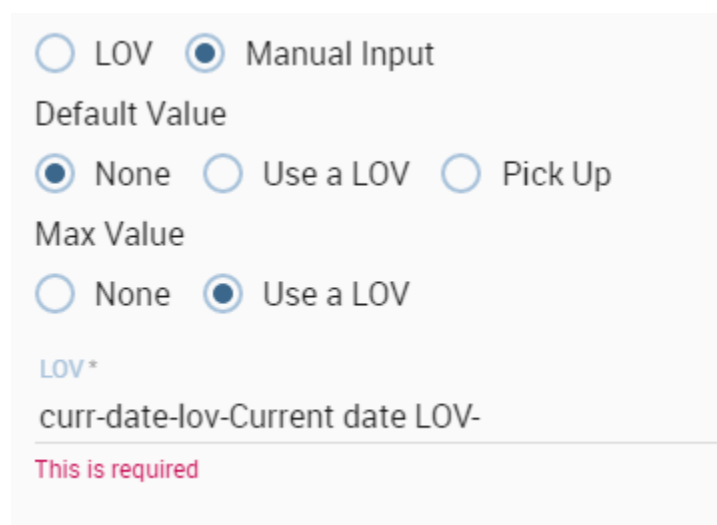


Fig. 5.45: Detail panel of LOV creation, specification of a maximum value.

During execution of a document, the date picker will be limited by that value:

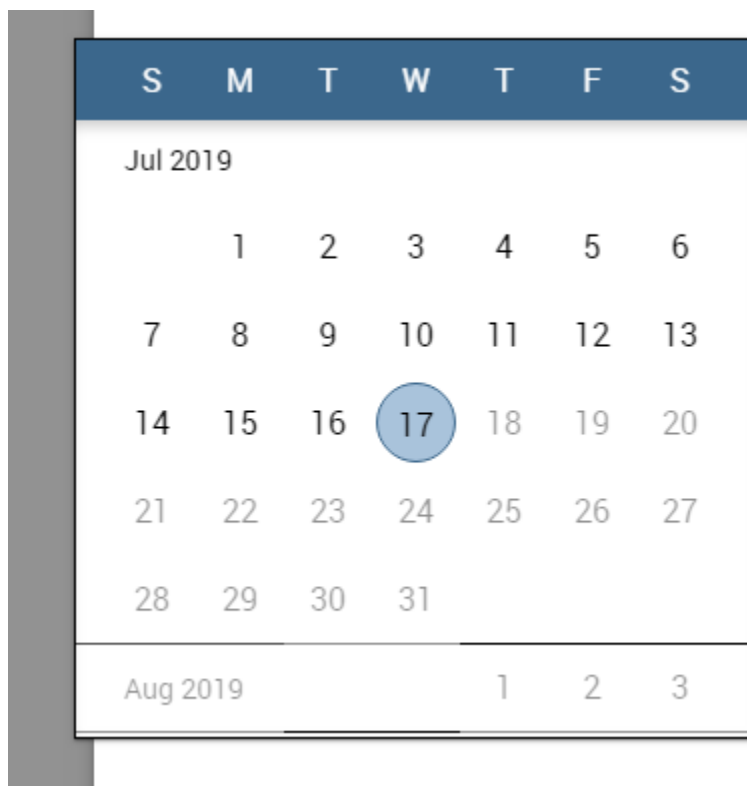


Fig. 5.46: Detail of a date picker for a date parameter with maximum value specified.

Note: Analytical driver of type Manual Input Date with a default value and/or max value

In the case you want to use an analytical driver of type Manual Input Date with a particular date as default value and/or a maximum value, you have to use a particular syntax for the LOVs query. See the note *Create a LOV for the default value of an analytical driver of type Manual Input Date* in the section *Creating a List Of Value* for more details.

Note: Analytical driver with hierarchical LOV and default LOV

In the case you want to use an analytical driver with a hierarchical LOV and a default LOV the latter need to be hierarchical too. For more details see *Create a LOV for the default value of an analytical driver with a hierarchical LOV* note in the section *Creating a List Of Value*.

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 use mode** 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.

5.3.2.5 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 the following figure, 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.

ANALYTICAL DRIVER USE MODE DETAILS

Label *
spazial_driver_USA

Name *
spazial_driver_USA

Description

☐ LOV ☒ Map Input ☐ Manual Input

Select layer *
UsaStates-UsaStates-File

Insert layer property
STATE_ABBR

CANCEL SAVE

Fig. 5.47: Spatial analytical driver settings.

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

5.3.2.6 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 the first figure of this chapter 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 family.

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.

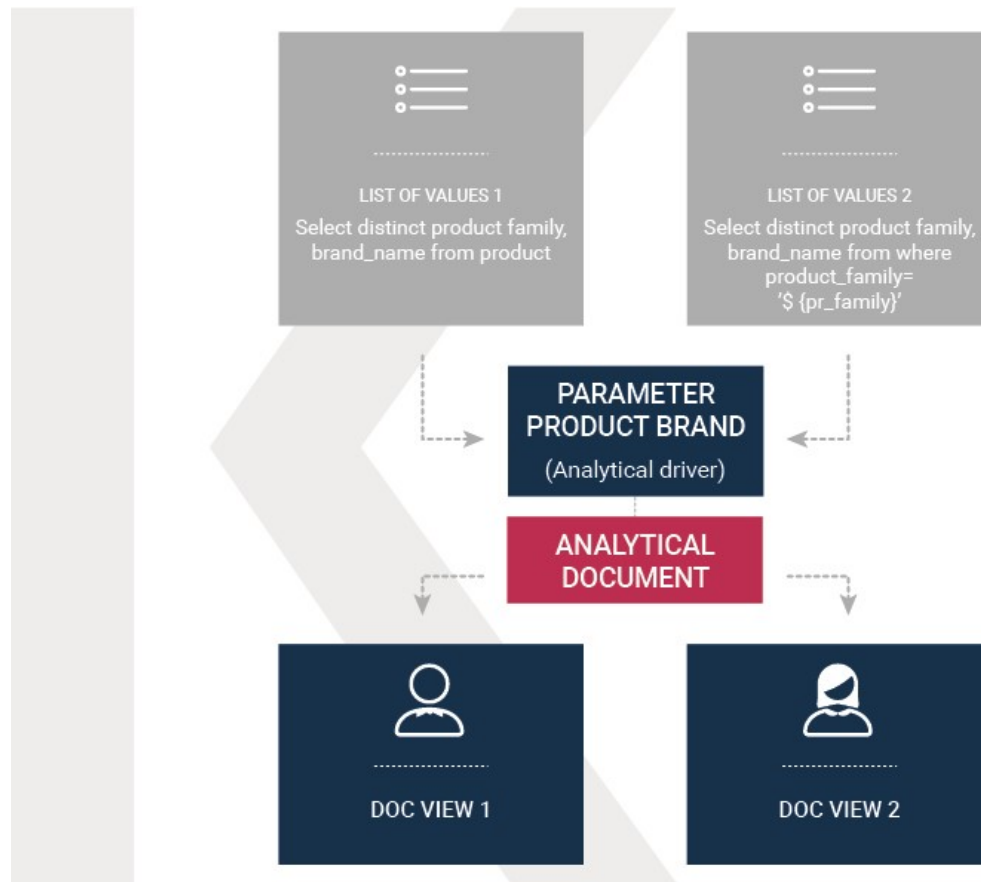


Fig. 5.48: Behavioural Model Schema.

5.3.2.7 Behavioural Model Lineage

It is possible to show a summary of the links between the LOVs, the analytical driver and the documents by selecting **Behavioural Model > Behavioural Model Lineage**.

The entire list of available LOVs, analytical driver and documents appears, as shown in figure below.

By selecting one LOV or Analytical Driver or Documents the other will refresh showing only the elements associated with the selection done. To come back to the original situation click the refresh button on the top right corner.

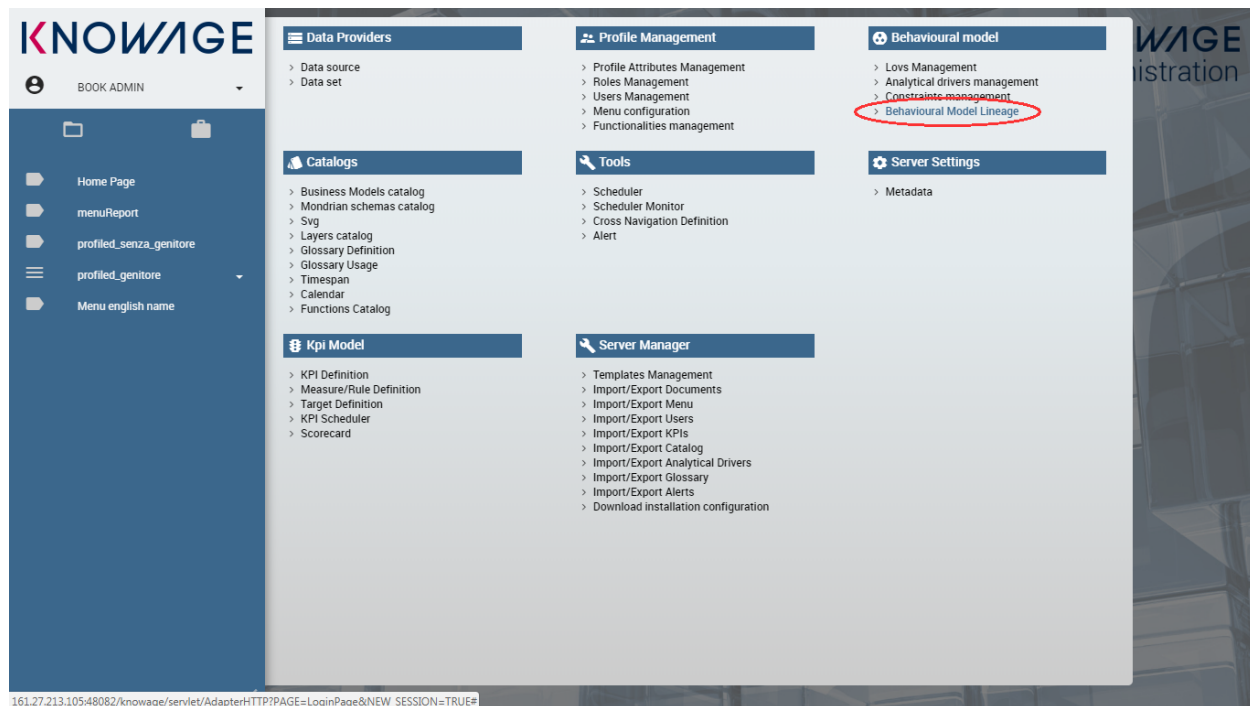


Fig. 5.49: Behavioural Model Lineage.

BEHAVIOURAL MODEL LINEAGE					
LOVS		ANALITICAL DRIVERS		DOCUMENTS	
Q Search	×	Q Search	×	Q Search	×
LABEL	NAME	LABEL	NAME	LABEL	NAME
CURRENT_MONTH_YEAR	Current month of the year	manual	manual	BAR Stacked	BAR Stacked
CURRENT_YEAR	Current year	store_country	store_country	BAR_CHART_PARAM	Grafico a Barre con parametro
CURRENT_MONTH	Current month	Tree inner node	Tree inner node	BAR_CHRT_001	Bar chart grouping/splitting functions
CURRENT_DATE	Current date	CORR_PRD_CAT	CORR_PRD_CAT	BAR_CHRT_002	Bar chart ordering function
Tree	Tree	City	City	BAR_CHRT_003	Bar chart ordering function date case
output_type	output_type	CORR_PRD_FAM	CORR_PRD_FAM	BAR_CHRT_004	Bar chart date function
impala_hier	impala_hier	CORR_PRD_SUBCAT	CORR_PRD_SUBCAT	BAR_CHRT_005	Bar chart 3D example
Tree2	Tree2	store_state	store_state	BIRT_MORE_PAGES	Birt with more pages
TEST_SAVE	TEST_SAVE	Output_Type	Output_Type	Bar_drill_test	Bar_drill_test
CORR_PRD_CLASS	CORR_PRD_CLASS	Tree	Tree	BigReport	BigReport
CORR_PRD_CAT	CORR_PRD_CAT	Country	Country	CHRT_CHORD_001	Chord chart
CORR_PRD_SUBCAT	CORR_PRD_SUBCAT	DatasetLOV_AD	DatasetLOV_AD	CHRT_GAUGE_001	Gauge Chart
store_country	store_country	PROMOTION_NAME	PROMOTION_NAME	CHRT_HTMAP_001	Heatmap Chart
store_state	store_state	manual_number	manual_number	CHRT_LINE	Line chart

Fig. 5.50: List of LOVs, analytical driver and documents.

5.4 Analytical Document

The *analytical model* is the core of Knowage Server and covers the whole range of analytical needs, providing many solutions for each analytical area, like reports, charts, OLAP documents, KPIs and so on.

The main element of the analytical model is the so called *analytical document*, a word used to group under a common concept all different types of documents that can be developed with Knowage (report, chart, cockpit, etc.) when performing a BI analysis.

In this chapter we describe step by step how to create a new analytical document. There exist many different document types, each with its own peculiarities. Here we provide a generic overview on common features, will focus on available types peculiarities in each dedicated part.

5.4.1 Main concepts

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. Templates can be encoded by hand or using Knowage Studio designers, when available. For each analytical document the history of templates is maintained. Old templates can be restored if needed. A new version is saved at each deployment, either manual or from Knowage Studio.

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.

5.4.1.1 Document types

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;
- attribution of business metadata;
- scheduled execution;
- collection of auditing and monitoring data;

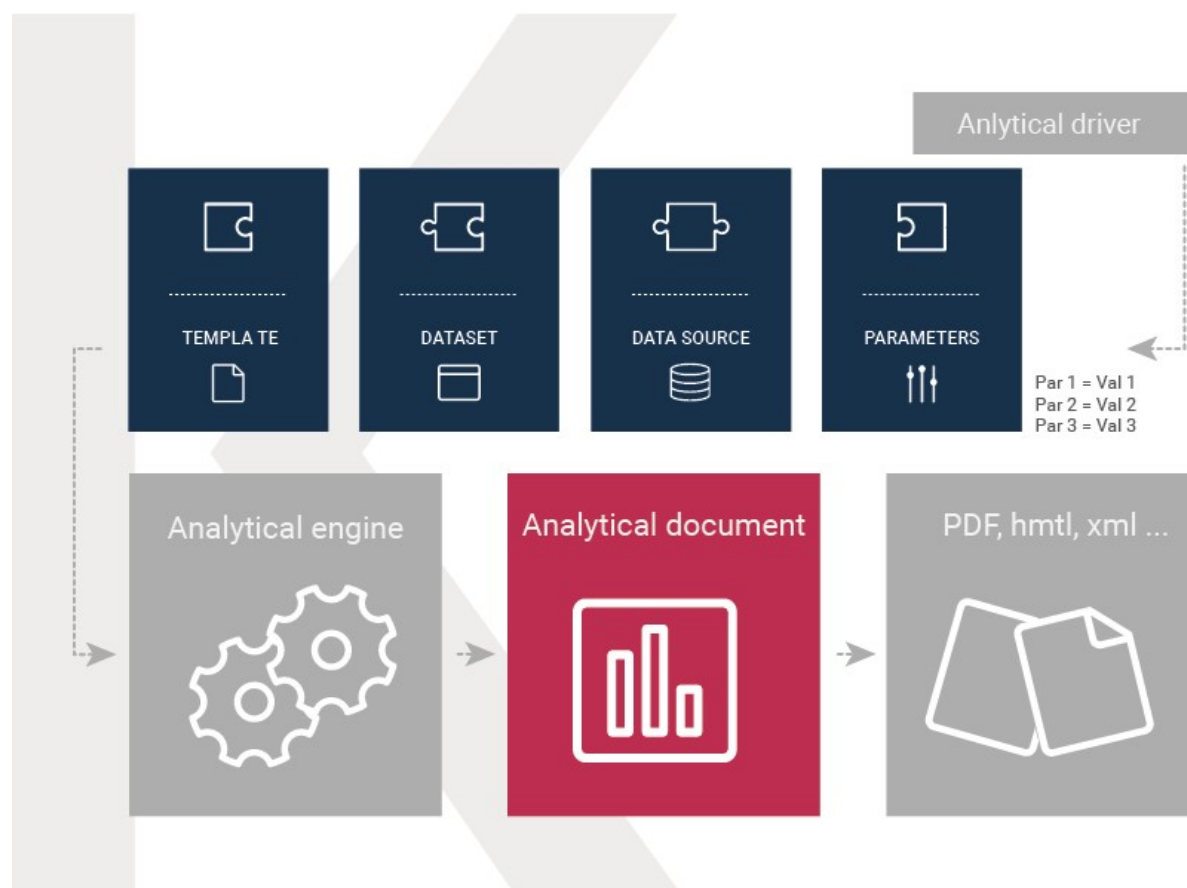


Fig. 5.51: Components of Knowage analytical document.

- adding end user notes;
- adding bookmarks;
- end user evaluation;
- sending the document by email;
- on-line or off-line (scheduled) execution.

This means that the above mentioned features are also inherited by every new engine that is developed or integrated into Knowage.

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

5.4.2 Register an analytical document

There are two different ways to create a new document on Knowage Server. The first option involves Knowage Studio: within it you can simply click on **Deploy** and the document window will open with pre-filled options. Please note that Knowage Studio can be used to create Birt or Dashboard document only.

Note: Deploy a document from Knowage Studio

Knowage Studio is the tool that allows to design and upload documents 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. This is the most general way since the Studio designer is not yet available for all documents types.

5.4.2.1 Analytical documents on Server

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

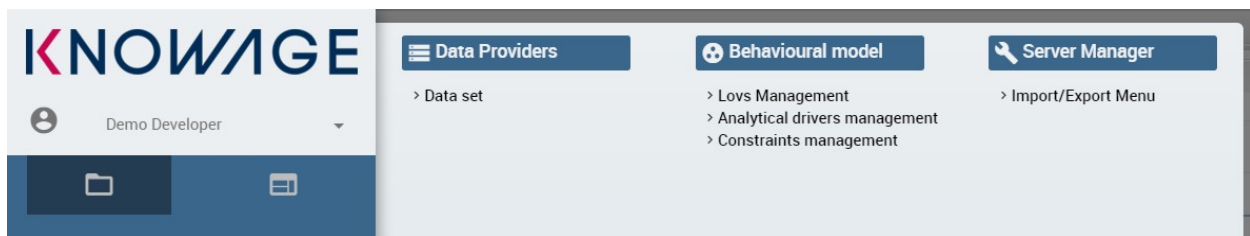



Fig. 5.52: Documents Development button.

By default the page is divided into two parts, as shown in figure below: 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 below.

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.

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

-  execute the document;

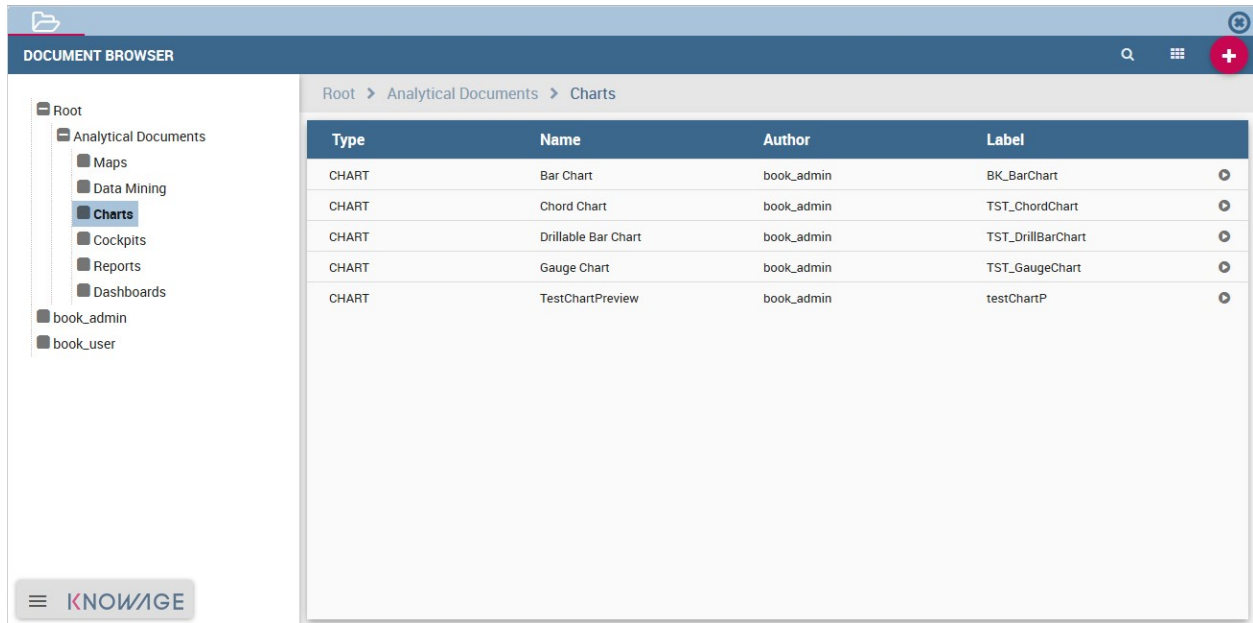


Fig. 5.53: Documents Development section.

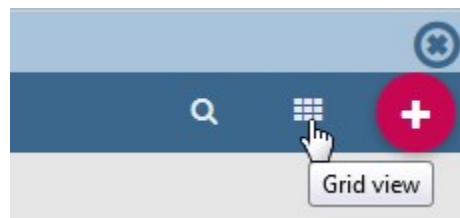


Fig. 5.54: Changing documents view.

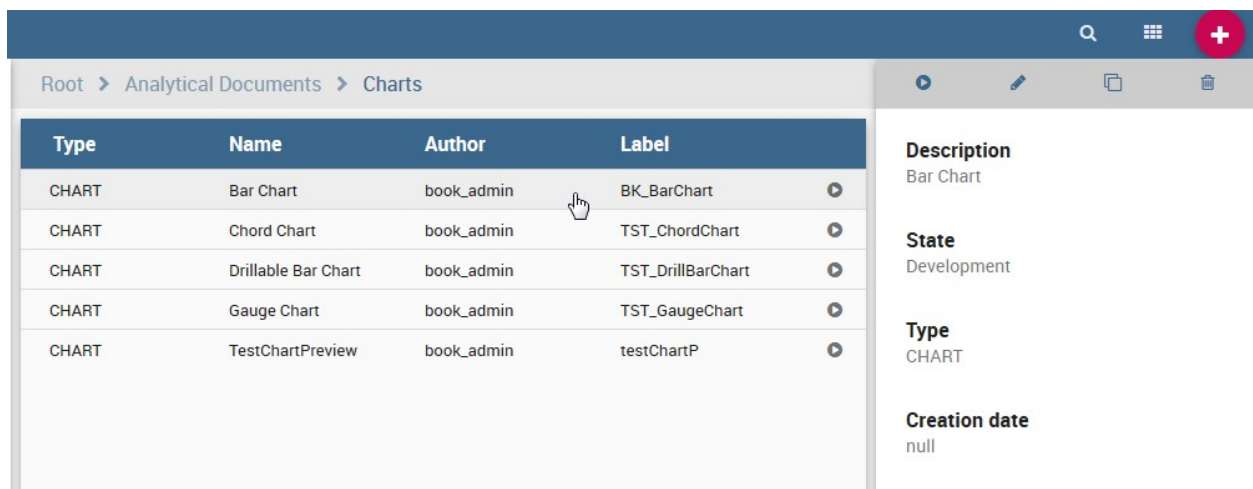





Fig. 5.55: Side panel.

-  access document details;
-  clone the item;
-  erase the document.

The figure below 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 see 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.

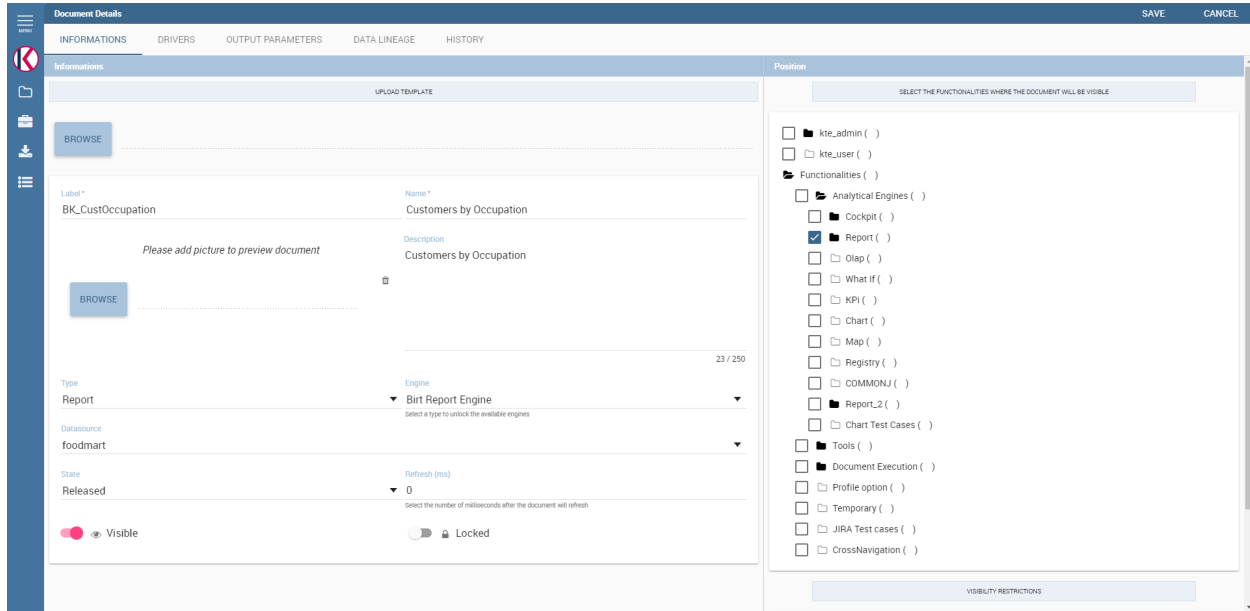


Fig. 5.56: Detail panel of Knowage analytical document.

In order to create a new document you need to click on the red plus button in the top right corner of the **Document Development** page. The different types of documents that you can create are: **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 above 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 below).

Now you have to select the dataset and/or data source that will feed your document with data. In this example you can see the pop-up window with datasets to select on the following figure.

Both should have already been defined in the corresponding sections for Knowage to show them in the available options of the menus. You can select the data source from the drop down menu. And select the dataset from the pop-up window and click save.

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 button save at the top right corner of the window.

Document Details [SAVE] [CANCEL]

INFORMATIONS

Informations

Position

UPLOAD TEMPLATE

BROWSE

Label *
Book_Example

Name *
Book_Example

Description
Book_Example

Please add picture to preview document

BROWSE

Type *
Chart

Engine *
Chart Engine

Dataset *

State *
Required

Refresh (ms)
0

Select the number of milliseconds after the document will refresh

Visible

Locked

SELECT THE FUNCTIONALITIES WHERE THE DOCUMENT WILL BE VISIBLE

☐ kte_admin ()

☐ kte_user ()

☒ Functionalities ()

VISIBILITY RESTRICTIONS

Defined Restrictions

Select Attribute

* Enter Value

Fig. 5.57: Select Type and Engine for a new document.

Select Dataset

Search

Label	Name	Description	Author	Scope
test_543	test_543		kte_admin	USER
test_54	test_54		kte_admin	USER
TestDavid2	TestDavid2		kte_user	USER
TestForPreview	TestForPreview	TestForPreview	kte_admin	USER
IMPALA_TEST_02	IMPALA_TEST_02	IMPALA_TEST_02	kte_admin	ENTERPRISE
cancelam9	cancelam9		kte_admin	USER
TEST_DATE_DRILL	TEST_DATE_DRILL	TEST_DATE_DRILL	kte_admin	ENTERPRISE
CHRT_HEATMAP	CHRT_HEATMAP		kte_admin	ENTERPRISE
IMPALA_TEST_03_P	IMPALA_TEST_03_P	IMPALA_TEST_03_P	kte_admin	ENTERPRISE
Prova elezioni senato	Prova elezioni senato	Prova elezioni senato	kte_admin	ENTERPRISE
CHRT_BAR_GROUP	CHRT_BAR_GROUP		kte_admin	ENTERPRISE
TEST_08	TEST_08		kte_admin	ENTERPRISE
Query2701	Query2701	Query2701	kte_admin	USER
TEST_SCALE	TEST_SCALE		kte_admin	ENTERPRISE
TEST_QBE_DATA2	TEST_QBE_DATA2	TEST_QBE_DATA2	kte_admin	ENTERPRISE
KNOWAGE-3730	KNOWAGE-3730	Dataset to test issue KNOWAGE-3730	kte_admin	ENTERPRISE
ds_6556626	FILE_XLS	FILE_XLS	kte_admin	USER
mare_turnover	mare_turnover	mare_turnover	kte_admin	USER
CHRT_BAR	CHRT_BAR		kte_admin	ENTERPRISE
TEST_06	TEST_06	TEST_06	angelo.admin	ENTERPRISE
Elezioni_storico	Elezioni_storico		kte_admin	USER
Prova elezioni	Prova elezioni	Prova elezioni	kte_admin	ENTERPRISE
more than 2			kte_admin	USER
ds_6572350	indicator_1 15-24 unemploy	indicator_1 15-24 unemploy	kte_user	USER
PSTG_SQL_TEST_1	PSTG_SQL_TEST_1	PSTG_SQL_TEST_1	kte_admin	ENTERPRISE

1 to 25 of 310 < > Page 1 of 13 >

[CANCEL] [SAVE]

Fig. 5.58: Selecting a dataset for the document.

5.4.2.1.1 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.

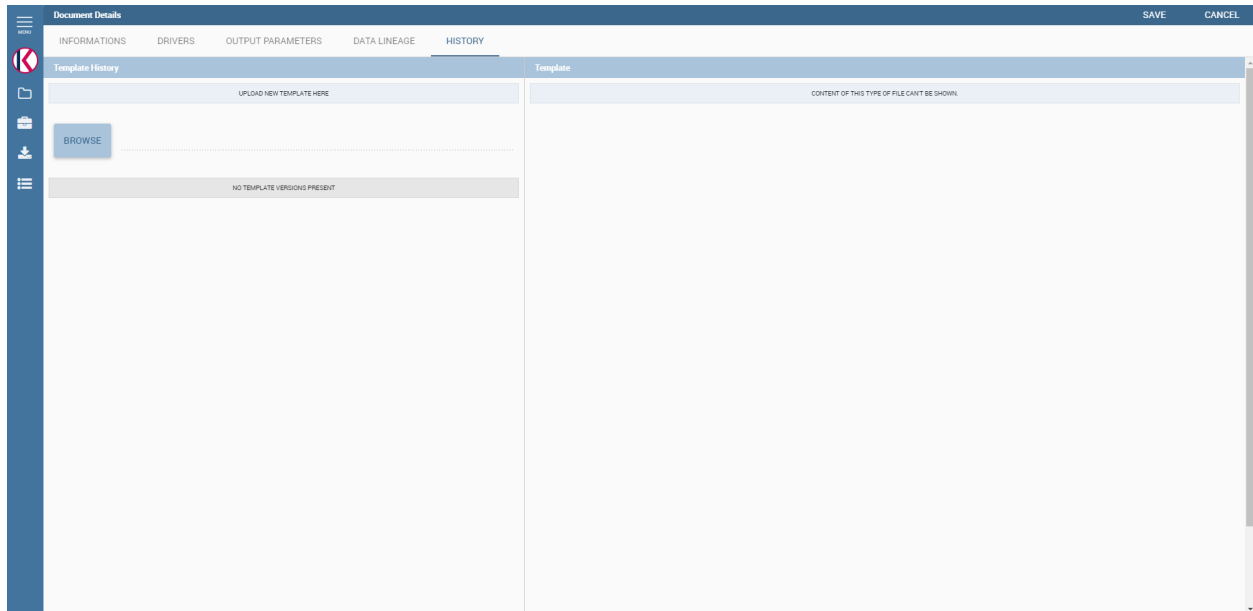
The main reason for this is that the state of the document has an impact on its accessibility. As discussed in the behavioural model, Knowage defines role types (administrator, developer, tester, user). States are compatible with the corresponding role type. Administrators can change the state of documents 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.

Important: Enterprise Edition only

In KnowageER you may also decide to temporary “lock” a document while he is working with it: it is enough to set the **Lock by user** item. This prevent other developers from modifying the same document you are working on.

5.4.2.1.2 Template Versioning

When you register a document on the Server, you need to associate a template to it. Click on tab **History** and then on button **Browse** to choose a template from your local file system and upload it.



You may have edited the template by hand or using the Studio designer. Clearly you will not have to upload the template if you are using the automatic deploy from the Studio.

Knowage Server supports versioning of uploaded templates, as shown below. To view them, click on tab **History**. All templates are saved with their date and name, and can be easily uploaded or deleted. To upload a template, click on button **Browse** to choose a template from your local file system and upload it: the new template will be uploaded. Using the same list you can download or delete a template.

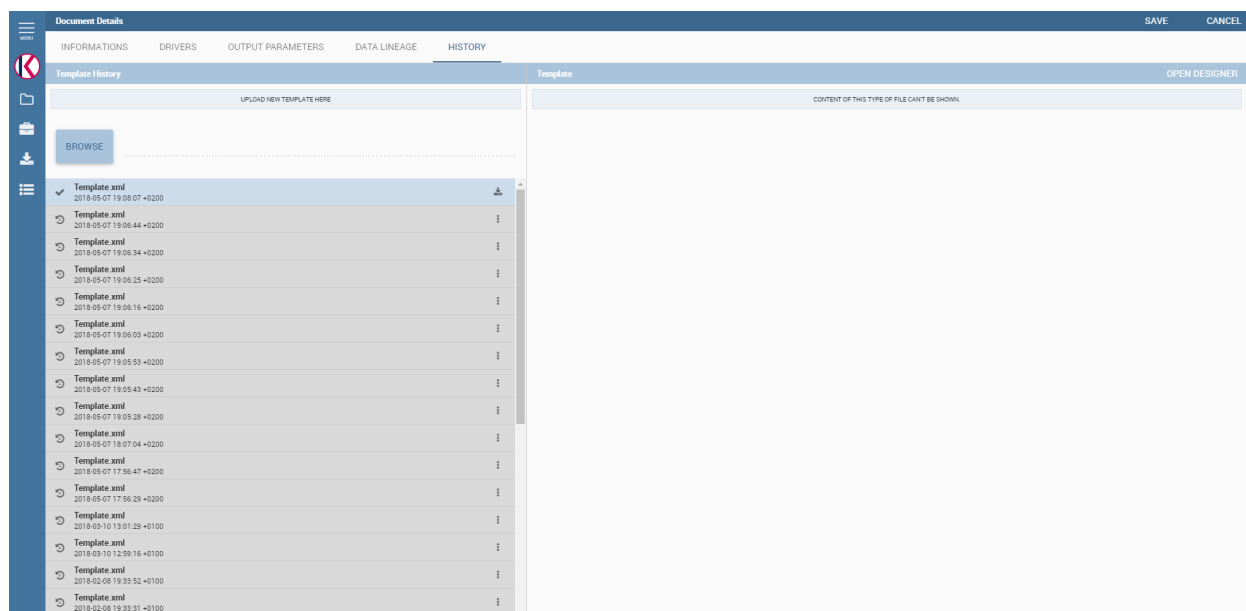


Fig. 5.59: Template versioning for analytical documents.

5.4.2.1.3 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.

Warning: Repository structure and rights

The **Functionalities tree** is Knowage document repository. It 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). On the right, 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.

5.4.3 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

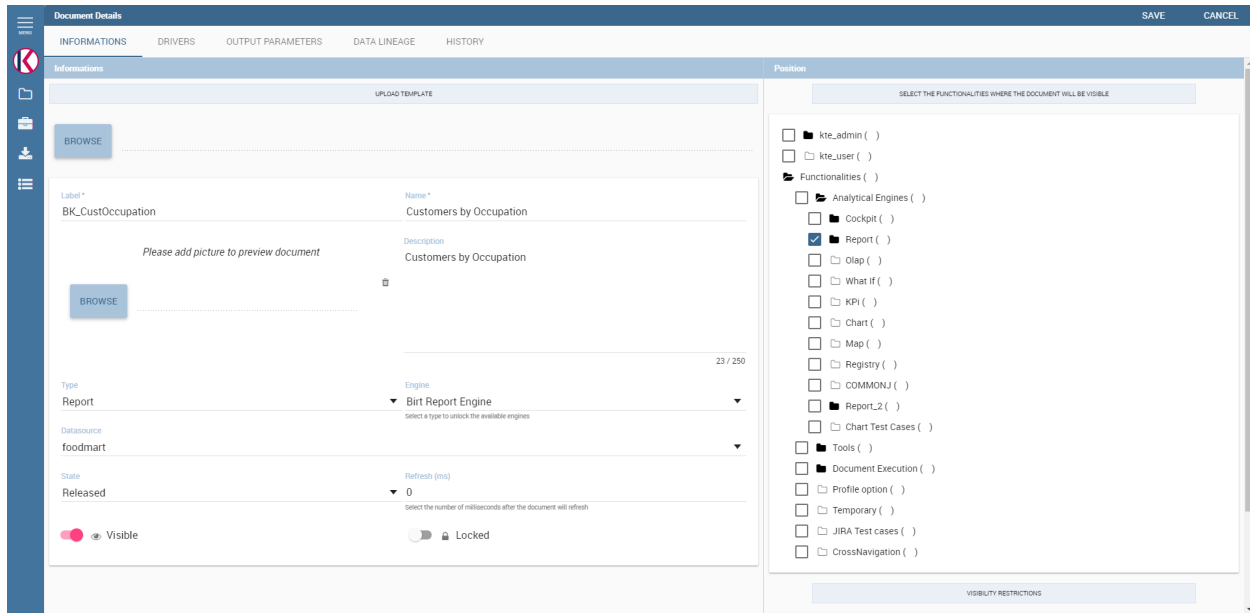


Fig. 5.60: Functionality Tree, document saving settings.

fine-grained level, beyond simple repository-based policies.

This can be done by editing conditions in the **Visibility restrictions** section on the right side of **Information** panel. To add a new condition pick a profile attribute from the drop down menu and assign it a value. 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.

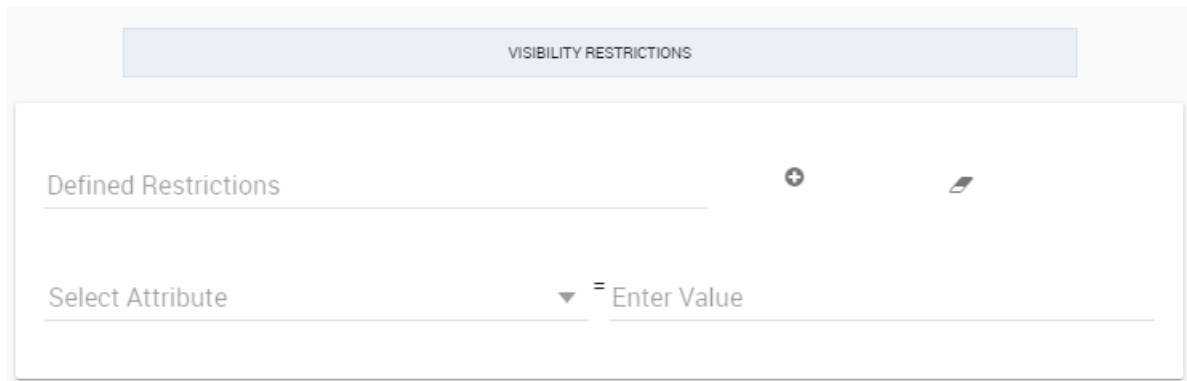


Fig. 5.61: Visibility restrictions.

5.4.4 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 click on the tab **Drivers** and then on a **Add** button, see the next figure.

Fig. 5.62: Association with analytical driver panel.

Choose a name for the title of driver. Then choose analytical driver from drop-down menu that you wish to associate to the document.

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 required 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 required 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.

After you have completed the definition of a parameter you can save it by clicking on main **Save** button in the upper right corner. To add further parameters, click on the **Add** button. Repeat the same procedure as many times you want. At this point you may wish to change the order of parameters (i.e., how they are presented to the user). To do so, click on the arrow in the list of drivers.

In the following we will see some special operations that can be performed on drivers associated to a document.

5.4.4.1 Associating a Spatial driver

As just seen, to filter on data visualization a user needs to associate an analytical driver using the “Document analytical driver details” area. As well as for the other driver you can use this interface to associate a spatial driver to the document. The procedure is right the same. When launching the document and opening the filter panel you will find



the filter just set. Click on the icon to open the map and select the geometric object (the State in the example in figure below) according to the chosen layer and property.

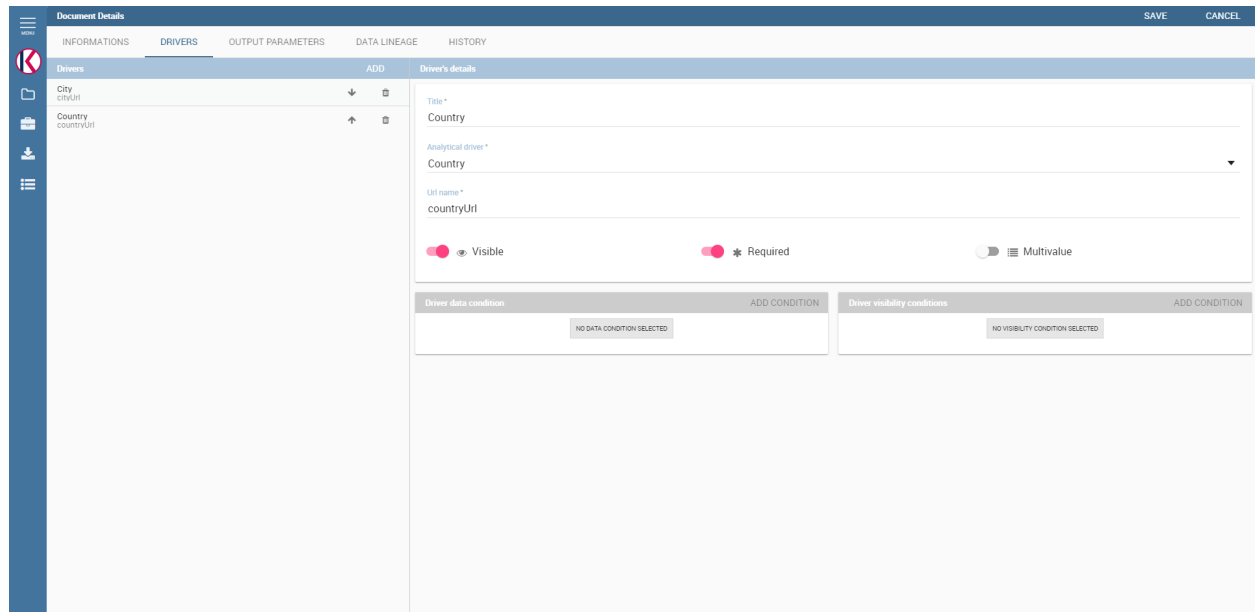


Fig. 5.63: Association with analytical driver panel.

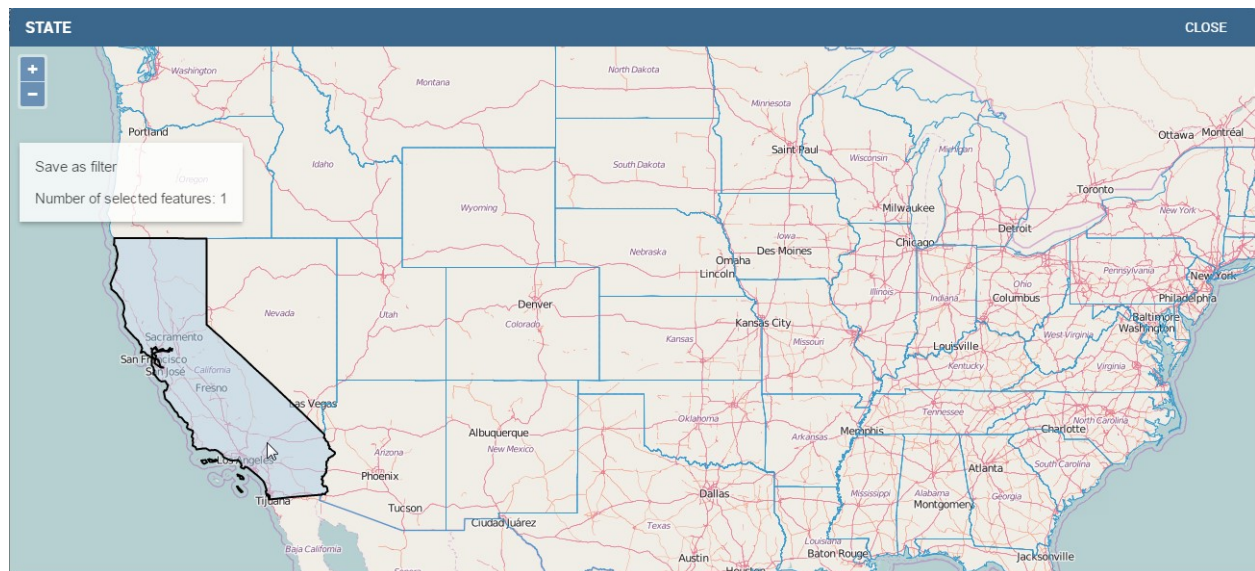


Fig. 5.64: Map filtering selection.

Click on “Close” button to confirm your selection. An example of the output is shown in the following figure.

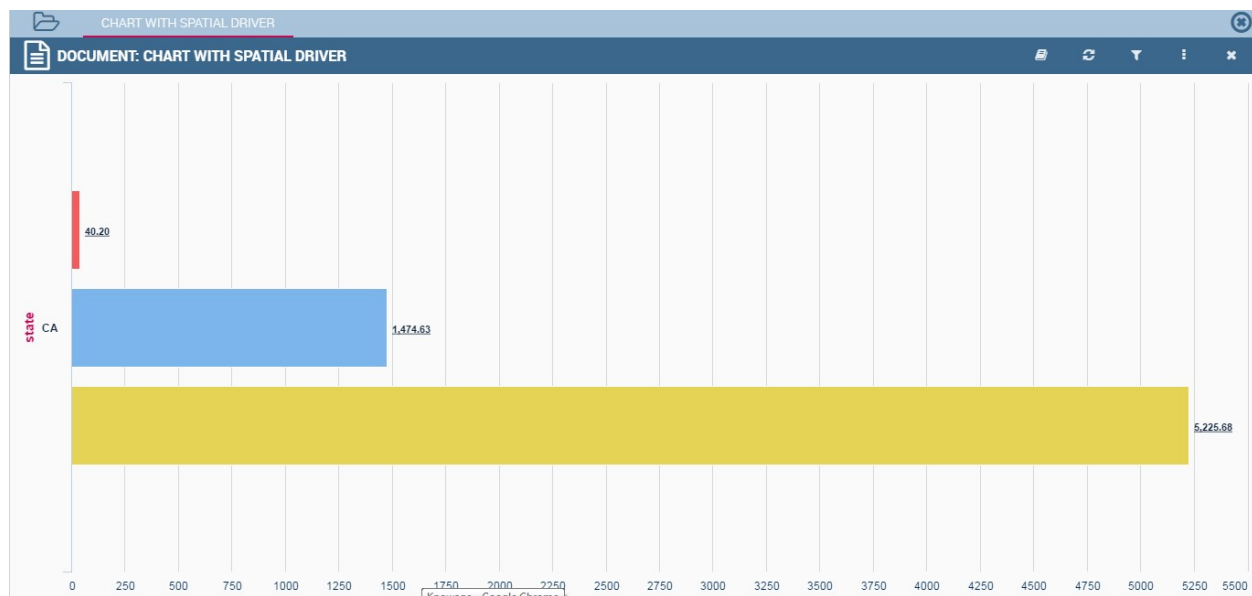


Fig. 5.65: Example of document run with a spatial filter.

5.4.4.2 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.

To set the correlation, select child parameter which will show you the details of that particular driver and then click on the **Add condition** button to open pop-up window for defining data correlation.

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.

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

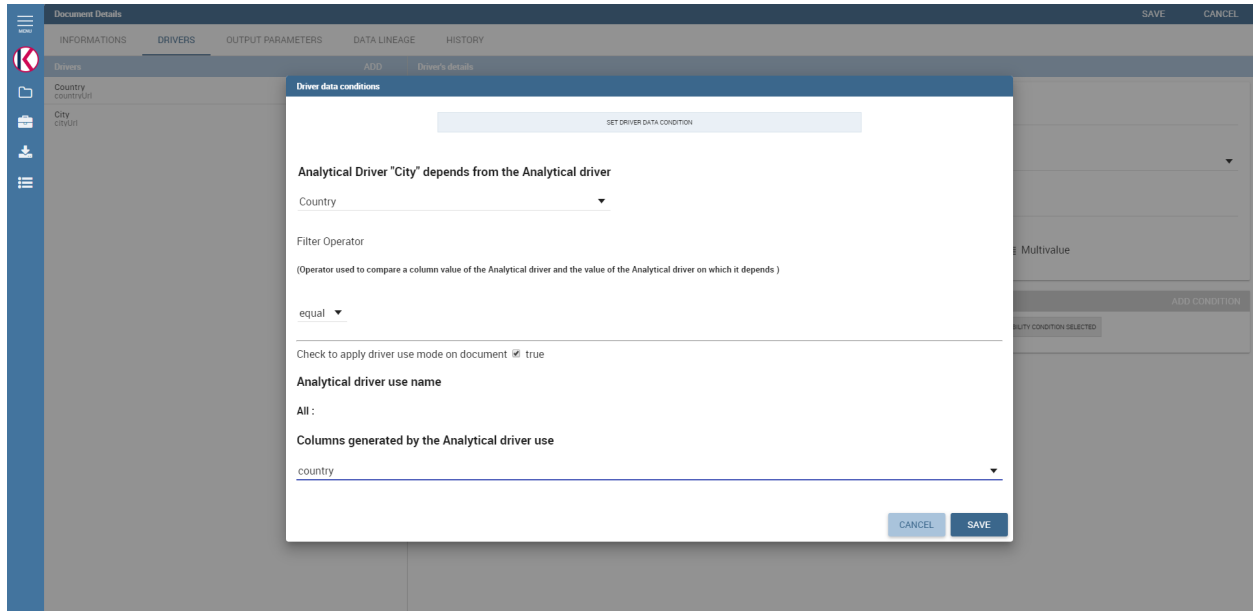


Fig. 5.66: Definition of the correlation.

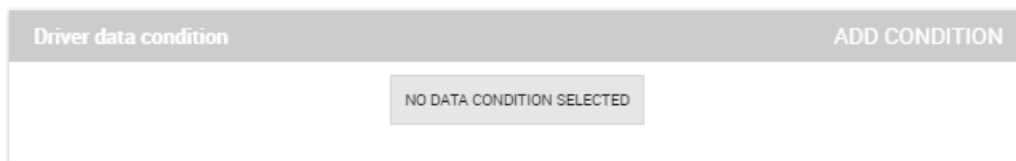


Fig. 5.67: Adding data correlation.

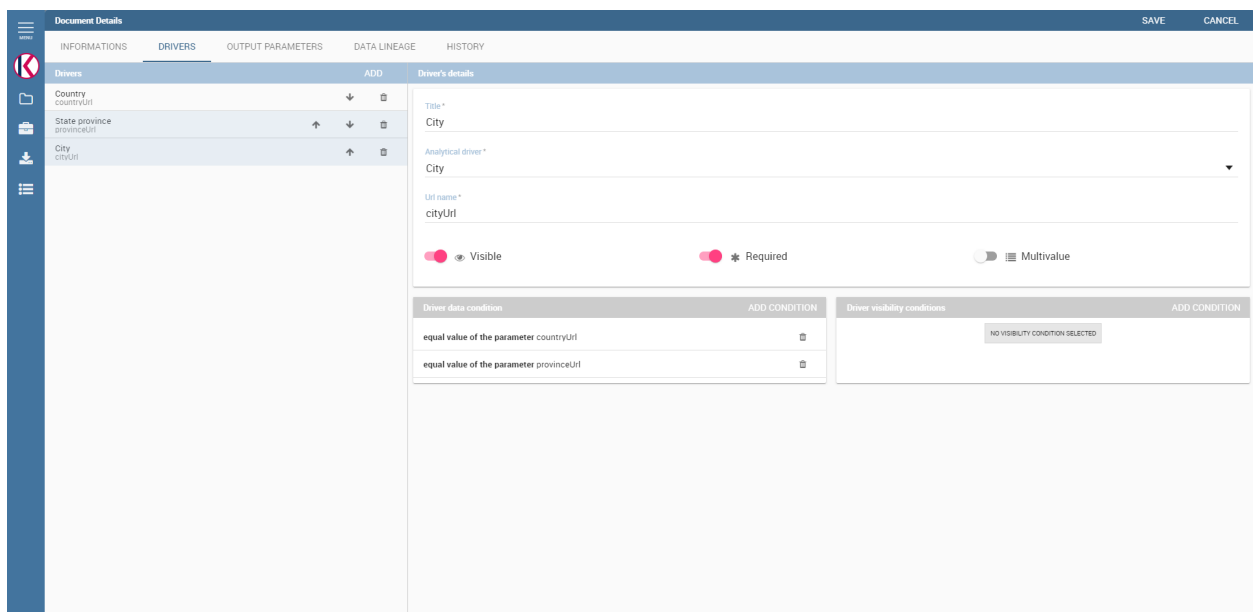


Fig. 5.68: Multiple correlations.

5.4.4.3 Correlation through LOV and drivers

In previous sections we saw how to set correlation through the GUI available in the document detail panel, but there is also the possibility to get the same result using the link between LOV and analytical drivers. More in depth, the user must have previously configured a driver that runs values that can be used in the “where” clause of a SQL query. Then the user must set a query-type LOV using the syntax

We stress that the `AD_name` is the name of the driver the administrator is trying to reach. Syntax for setting correlation through LOV configuration is:

Listing 5.13: Syntax for setting correlation through LOV configuration

```
$P{AD_name}
```

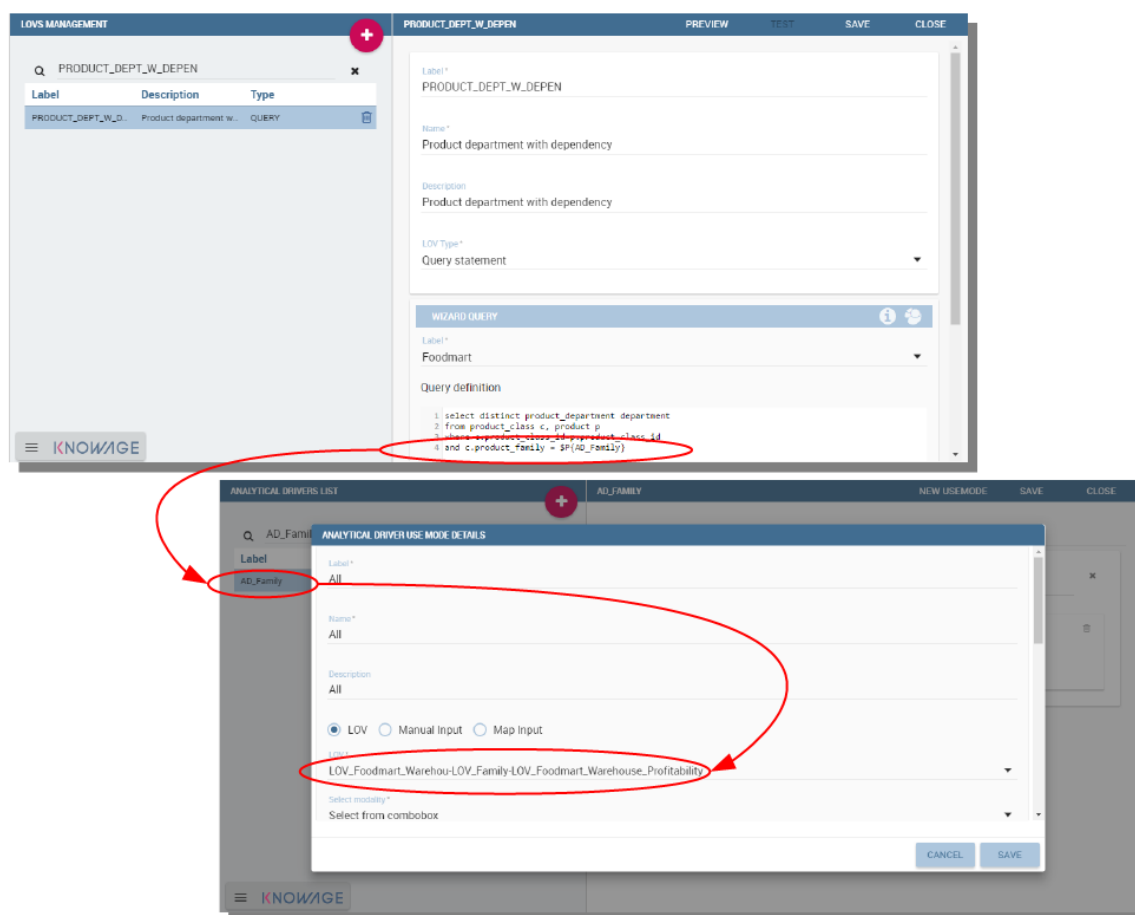


Fig. 5.69: Correlation passing driver values to LOV query .

As a result, at document execution, as soon as the user pick up a value from the “free” parameter, the other one is filtered and will show only the value related to the previous selection, as shown in Figure below.

5.4.4.4 Controlled visibility

Another type of relation between parameters is supported by Knowage. It is possible to define values of a parent parameter that force 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.

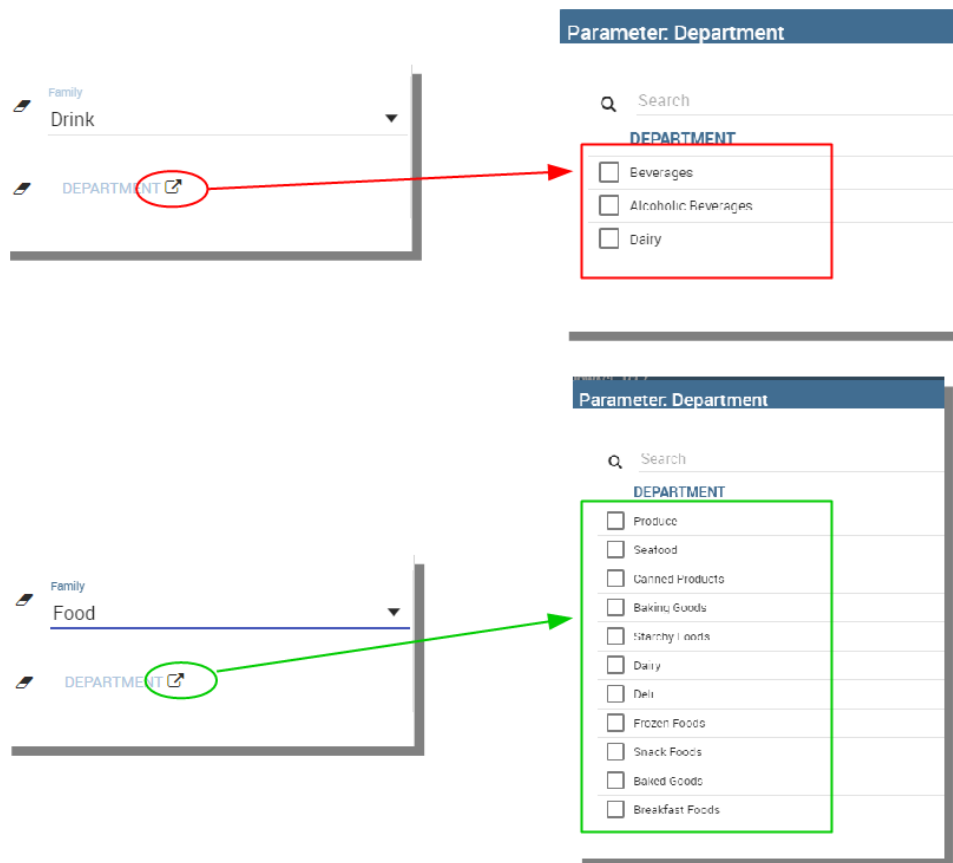


Fig. 5.70: Filtering with correlation.

To set a visibility expression, click on the **Add condition** button on the **Driver visibility conditions** card.



Fig. 5.71: Adding visual correlation

In the graphical editor you can define visibility rules similarly to correlation ones, as shown in figure below.

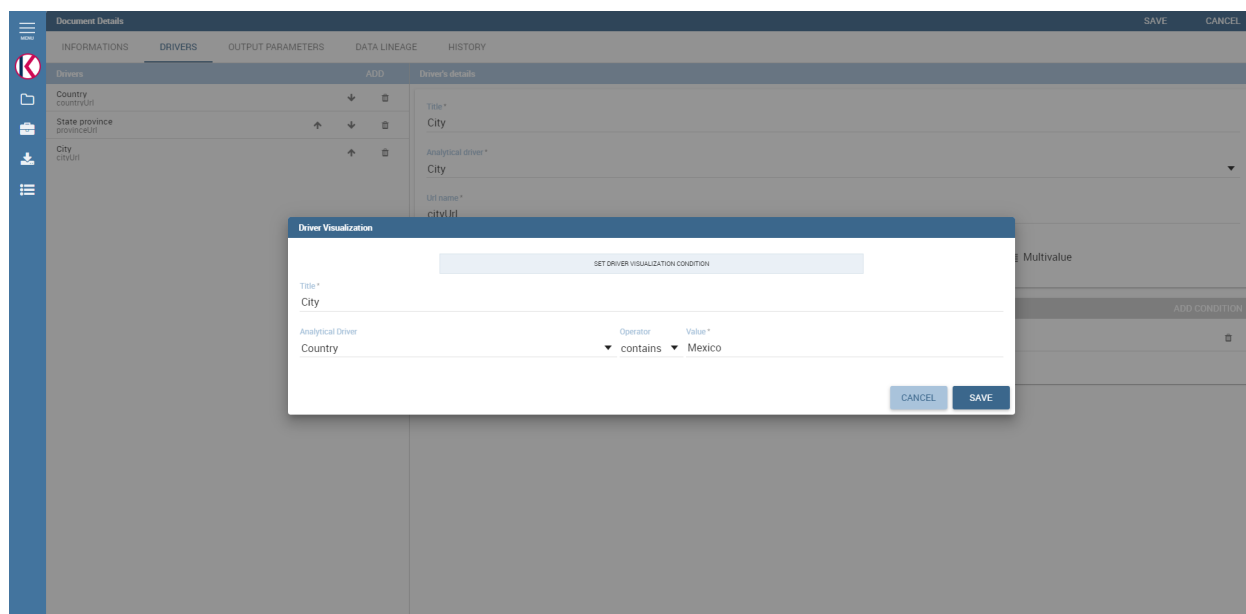


Fig. 5.72: Visibility expressions.

5.4.5 Cross Navigation

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 crossnavigation 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 necessarily coincide with all the filters applied to the document. Please refer to Chapter of Behavioural model 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 outputparameters** button 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.

5.4.5.1 Declaration of the output parameters

Enter the **Document details** of the document of interest. Then click on **Output parameters** tab and then on the button **Add** for adding new 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 (see next figure), it is possible to choose the format in which your date has been coded. The default value is related to the location defined in (**Menu > Languages**).

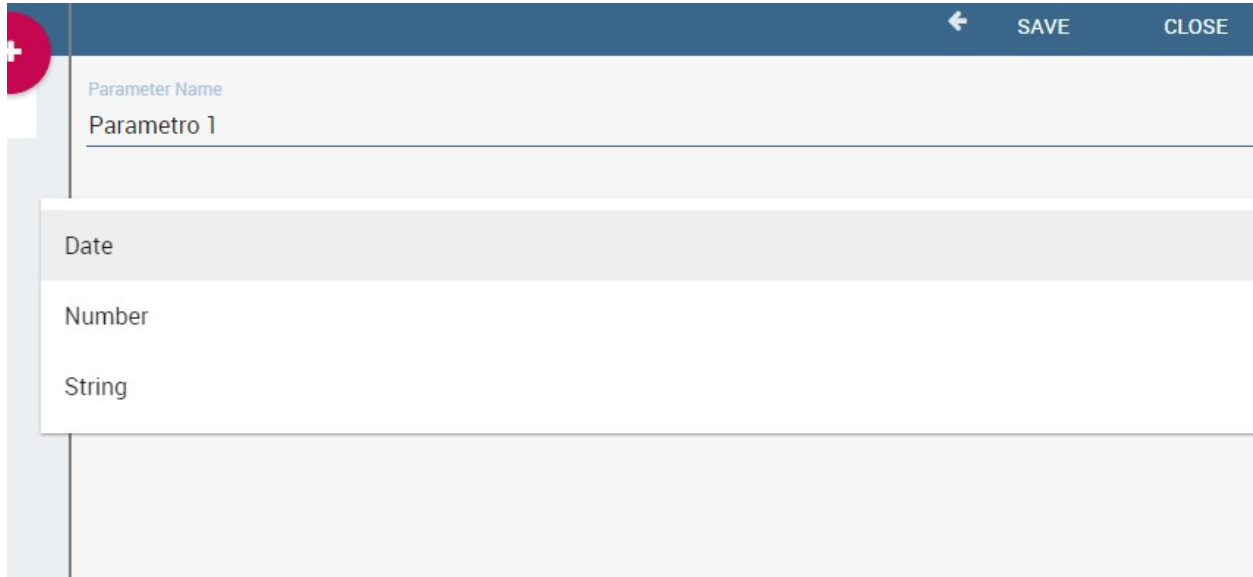
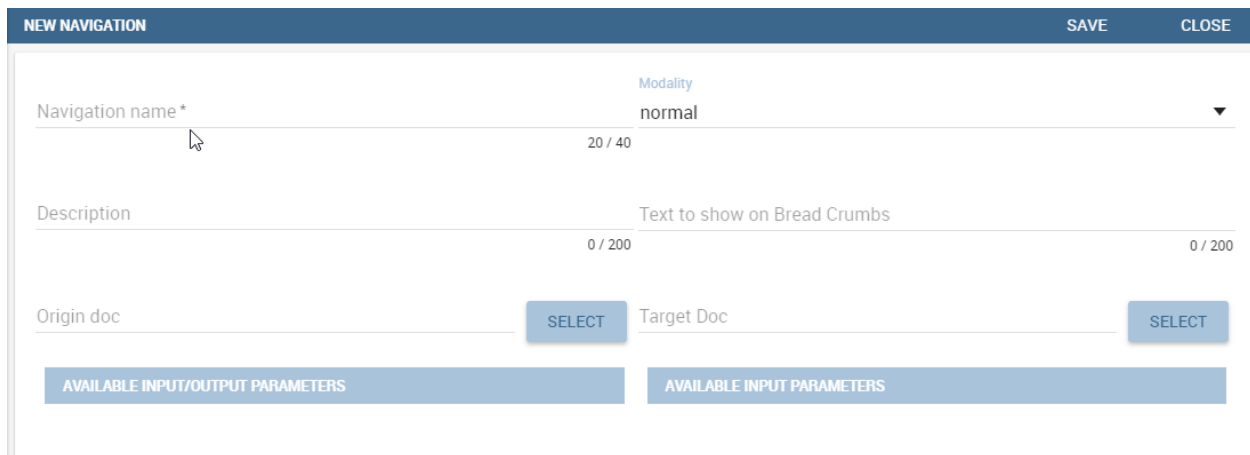


Fig. 5.73: Setting an output parameter.

5.4.5.2 Cross navigation definition

Finally you need to select the **Cross Navigation Definition** item from the menu to configure the cross navigation. The figure below shows the cross navigation definition window.



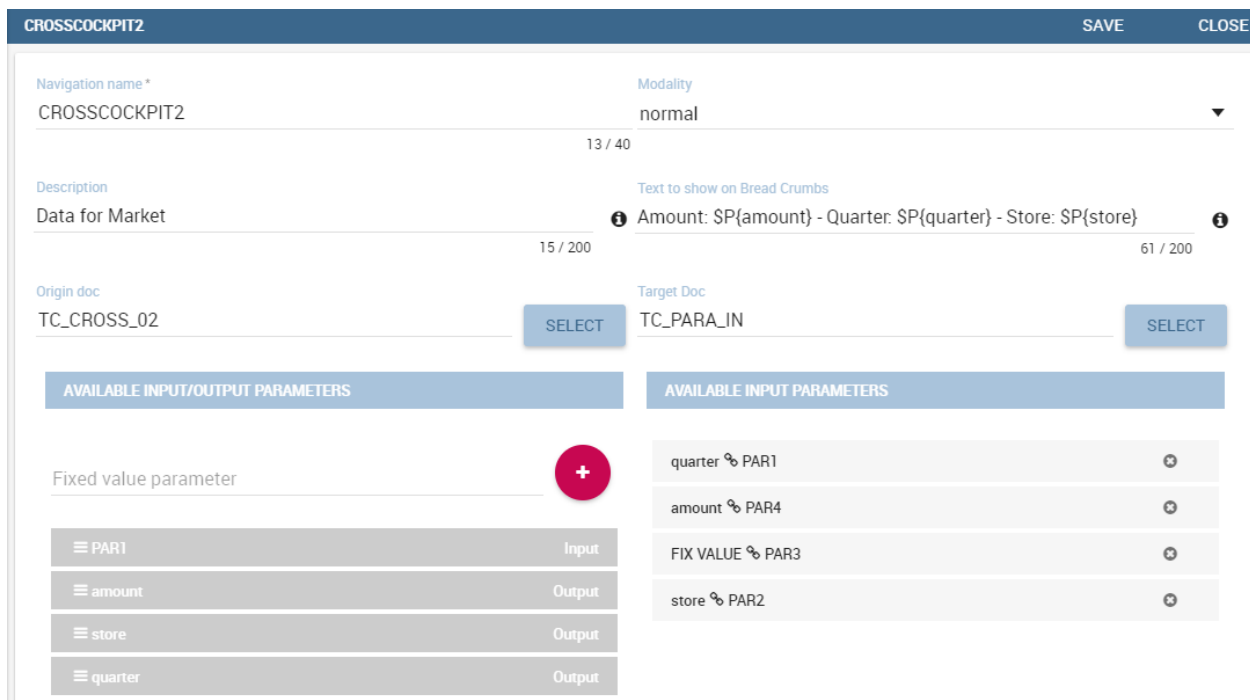
The form is titled "NEW NAVIGATION" and has "SAVE" and "CLOSE" buttons in the top right. It contains the following fields:

- Navigation name ***: A text input field with a character count of "20 / 40".
- Modality**: A dropdown menu currently set to "normal".
- Description**: A text input field with a character count of "0 / 200".
- Text to show on Bread Crumbs**: A text input field with a character count of "0 / 200".
- Origin doc**: A text input field with a "SELECT" button next to it.
- Target Doc**: A text input field with a "SELECT" button next to it.
- Below the "Origin doc" and "Target Doc" fields are two blue buttons labeled "AVAILABLE INPUT/OUTPUT PARAMETERS".

Fig. 5.74: 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.

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



The form is titled "CROSSCOCKPIT2" and has "SAVE" and "CLOSE" buttons in the top right. It contains the following fields and sections:

- Navigation name ***: A text input field containing "CROSSCOCKPIT2" with a character count of "13 / 40".
- Modality**: A dropdown menu currently set to "normal".
- Description**: A text input field containing "Data for Market" with a character count of "15 / 200".
- Text to show on Bread Crumbs**: A text input field containing "Amount: SP{amount} - Quarter: SP{quarter} - Store: SP{store}" with a character count of "61 / 200".
- Origin doc**: A text input field containing "TC_CROSS_02" with a "SELECT" button next to it.
- Target Doc**: A text input field containing "TC_PARA_IN" with a "SELECT" button next to it.
- Below the "Origin doc" and "Target Doc" fields are two blue buttons labeled "AVAILABLE INPUT/OUTPUT PARAMETERS".
- Below the "AVAILABLE INPUT/OUTPUT PARAMETERS" buttons is a section titled "Fixed value parameter" with a red circular button containing a white plus sign (+).
- Below the "Fixed value parameter" section is a table with four rows:

Parameter	Type
PAR1	Input
amount	Output
store	Output
quarter	Output
- Below the table is another section titled "AVAILABLE INPUT PARAMETERS" with four rows:

quarter	PAR1	✖
amount	PAR4	✖
FIX VALUE	PAR3	✖
store	PAR2	✖

Fig. 5.75: Setting the cross navigation through the tree item.

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

To assign fixed values to target parameters it is necessary to edit first the box labelled **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.

CROSSCOCKPIT2

SAVE

CLOSE

Navigation name*

CROSSCOCKPIT2

Modality

normal

13 / 40

Description

Data for Market

15 / 200

Text to show on Bread Crumbs

Amount: \$P{amount} - Quarter: \$P{quarter} - Store: \$P{store}

61 / 200

Origin doc

TC_CROSS_02

SELECT

Target Doc

TC_PARA_IN

SELECT

AVAILABLE INPUT/OUTPUT PARAMETERS

AVAILABLE INPUT PARAMETERS

Fixed value parameter

+

PAR1

Input

amount

Output

store

Output

quarter

Output

quarter

PAR1

*

amount

PAR4

*

FIX VALUE

PAR3

*

store

PAR2

*

Fig. 5.76: Relating parameters.

store

PAR2

*

Fig. 5.77: Association between parameters.

As you know, it is possible to define multiple cross navigation starting by the same document. In this case the system will show a popup window to choose the one that you want execute. It is possible set a specific description for each cross navigation so that will be easy to recognize the right navigation definition to use. In the same way it is possible set the text of the bread crumb and personalize it. For both attributes it is possible show parameters (of input or output type) values through the syntax `$P{parameter_name}`. Just parameters of source documents are available.

Description	Text to show on Bread Crumbs
Data for Quarter <code>\$P{quarter}</code>	i Amount: <code>\$P{amount}</code> - Quarter: <code>\$P{quarter}</code> - Store: <code>\$P{store}</code> i
28 / 200	61 / 200

Fig. 5.78: Example of parametric description and breadcrumb text

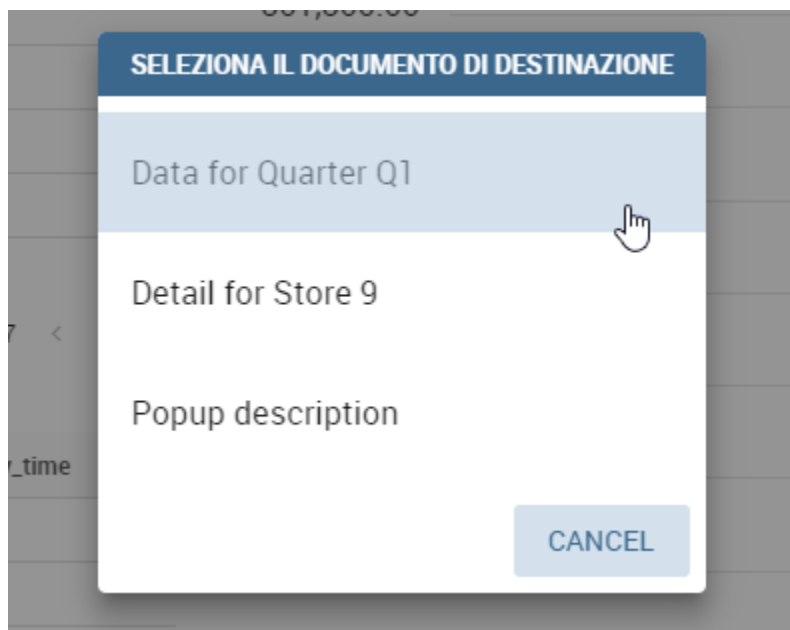


Fig. 5.79: Example of popup selection for more cross navigation definition (with params)

5.5 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
- Line
- Pie
- Sunburst
- Wordcloud
- Treemap

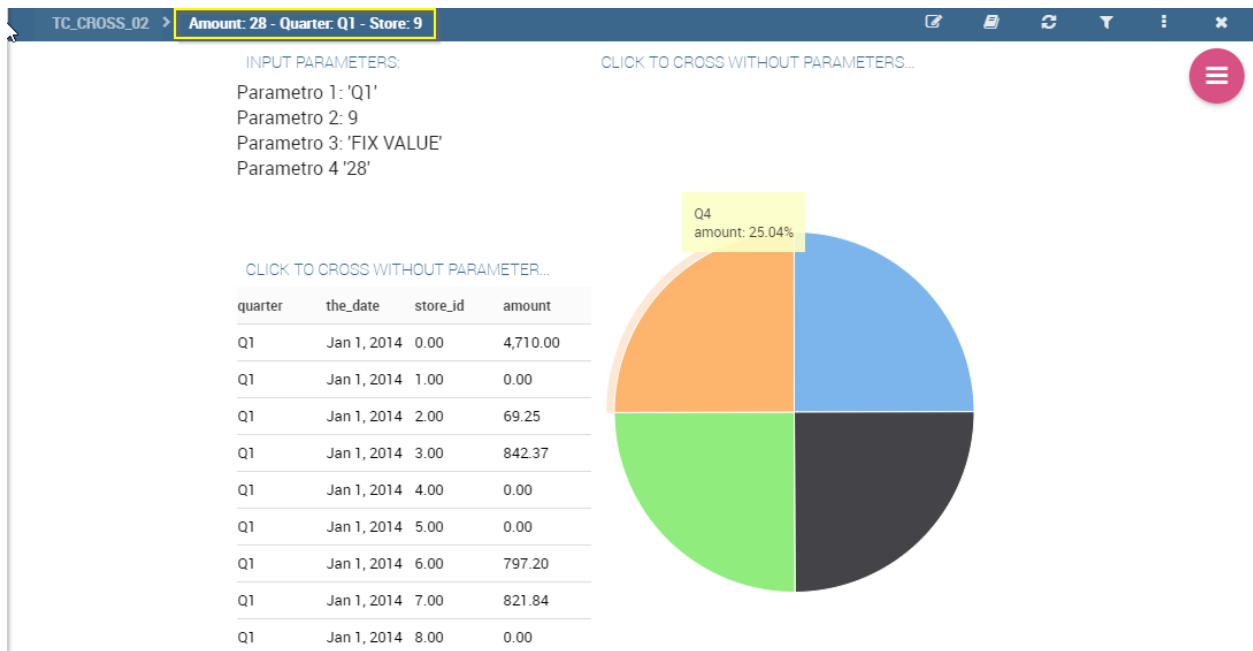


Fig. 5.80: Example of breadcrumb with params

- Parallel
- Radar
- Scatter
- Heatmap
- Chord
- Gauge

5.5.1 My first 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 **Cockpit**.

Important: Enterprise Edition only

Please note that this operation is available only in KnowageBD and KnowageSI. Using the KnowagePM license, only a technical user can create charts document, as explained in **Stand alone charts** chapter.

Once opened, the cockpit interface is an empty page with a toolbar containing different options, the second of which is the **Add chart** feature.

Note: Cockpit

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

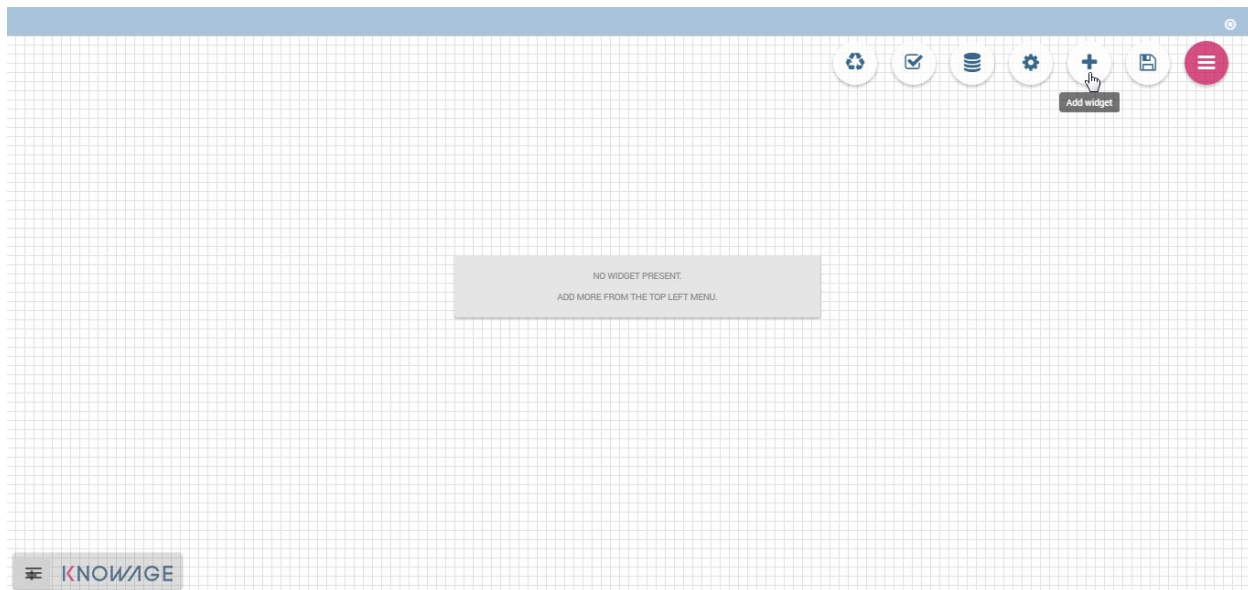


Fig. 5.81: Add a chart to a cockpit.

Clicking on the **Add Chart** 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 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 below.

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

Clicking on the **Configuration** page you will find eight different blocks as you can see in figure below.

In detail these 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**
- **Custom Colors**

These eight blocks are common to all chart types; anyway, some chart types may have additional blocks.

The **Advanced** tab contains extra features, usually exploited by an expert user. Here the user can see all settable properties associated to the chart: it reflects the property tabs that an expert user should manually edit to generate a json template.

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

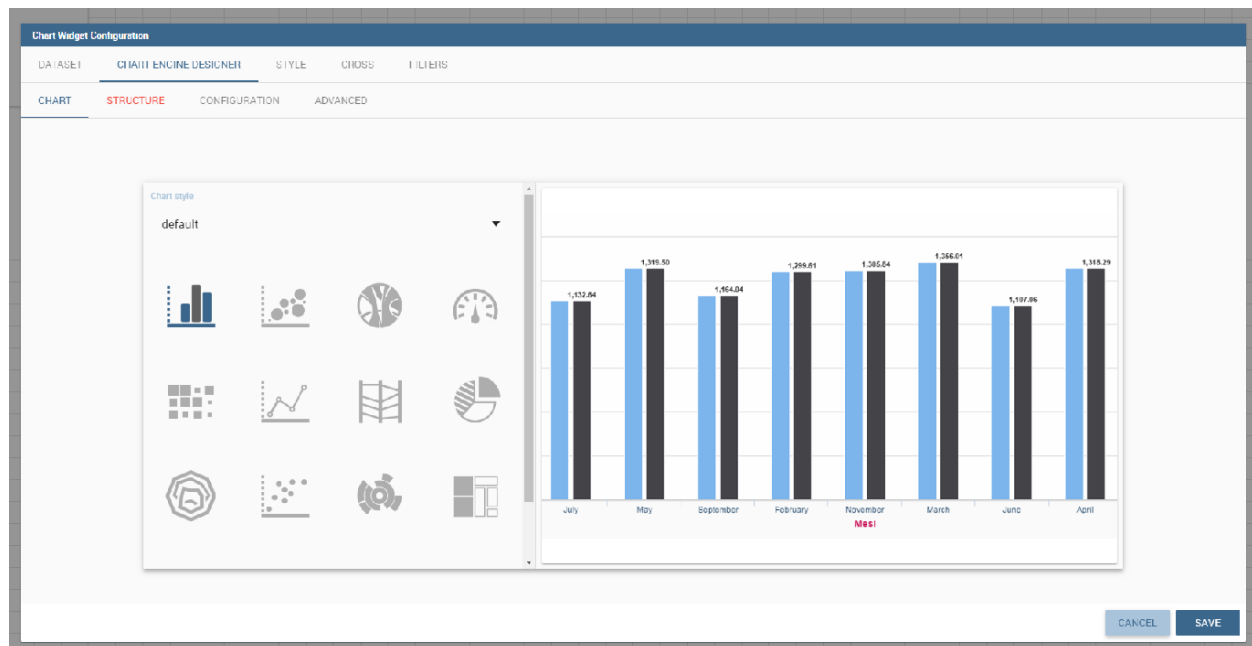


Fig. 5.82: Chart editor.

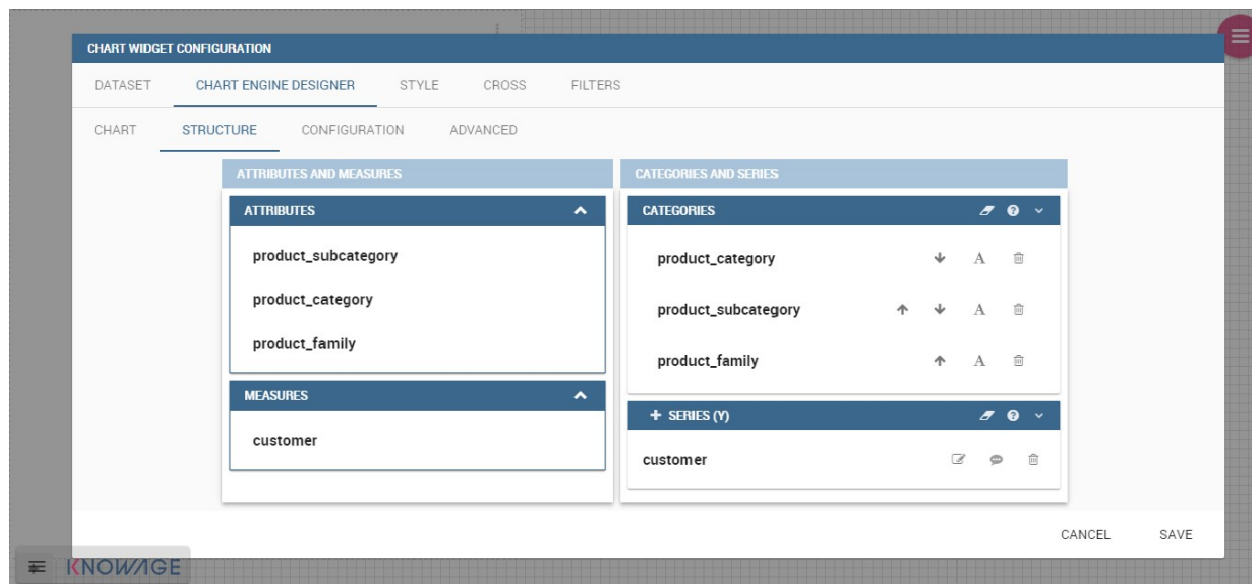


Fig. 5.83: Chart structure.

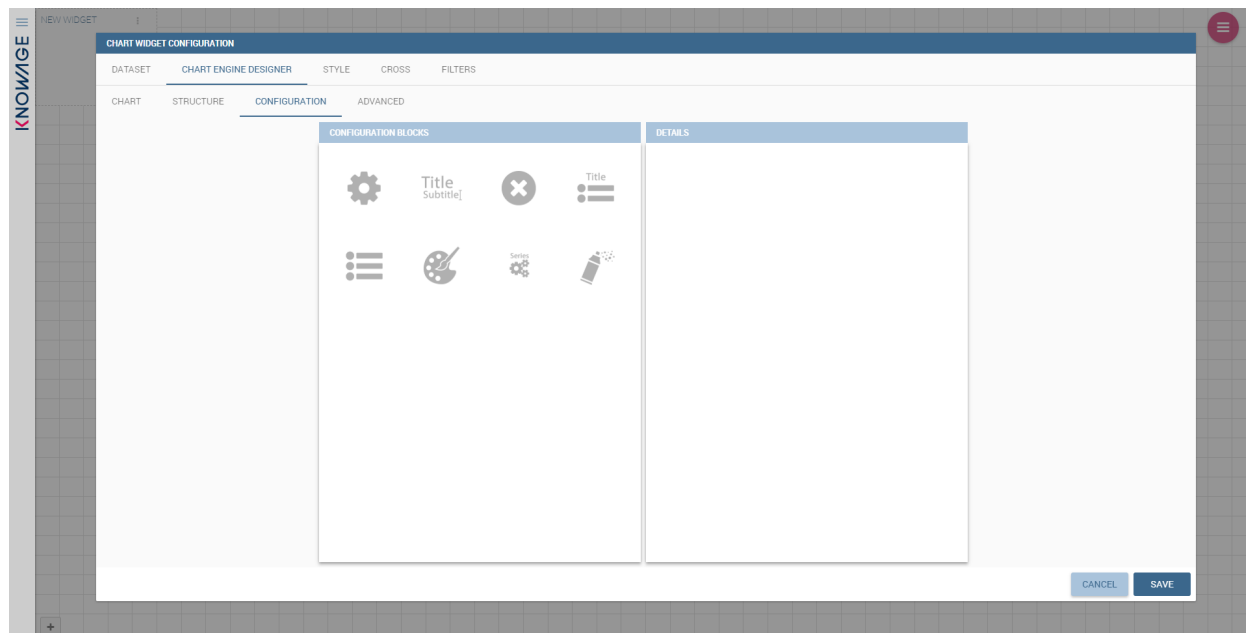


Fig. 5.84: Chart configuration.

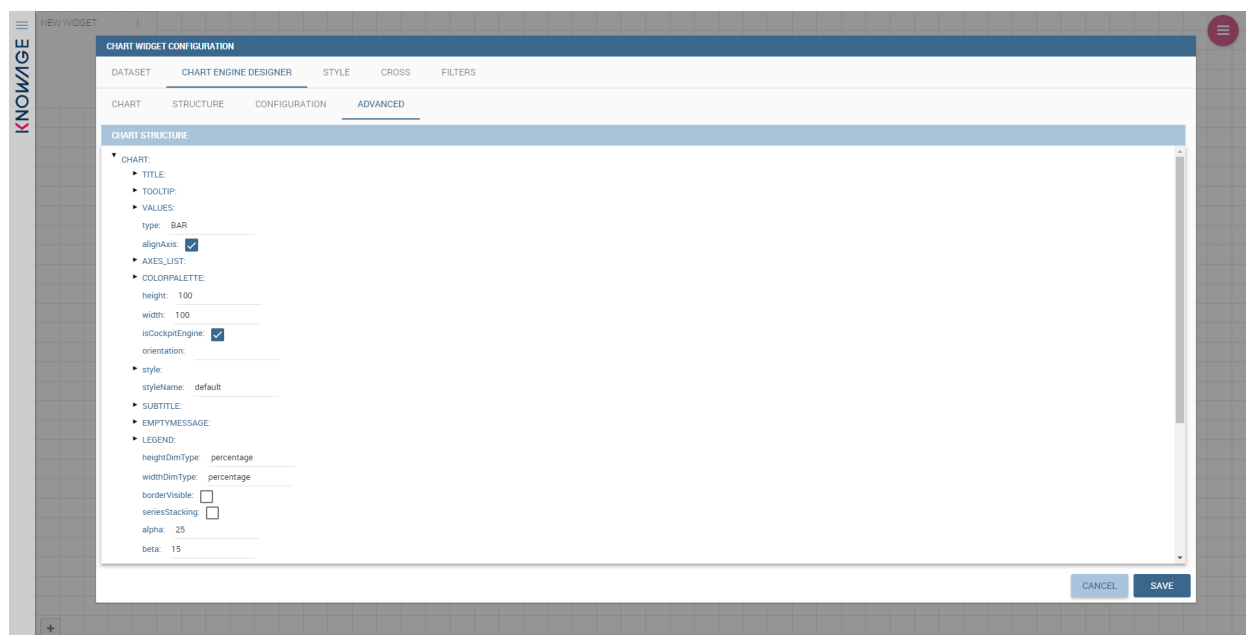


Fig. 5.85: Chart Advanced Features.

5.5.1.1 Structure

The “Structure” tab of the designer is the core of the Chart development. Here it is possible and mandatory to choose the measures and the attributes. When selected, 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.

Warning: Chart type changemens may cause broke down

Before creating any chart, it is convenient to be sure of what kind of chart you want to develop. We stress that the user can change the chart type afterwards, but at the expense of a loss of just defined settings.

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 choose 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. The Figure below will show in detail the three buttons above explained:

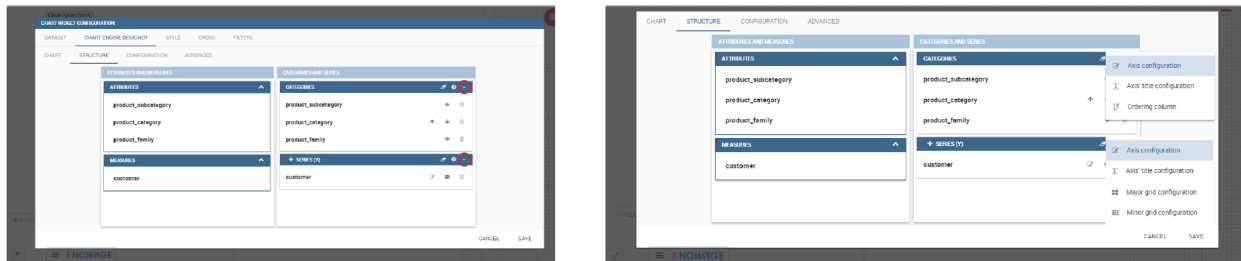


Fig. 5.86: From left to right: (a) Generic configuration axis (the specific arrow). (b) Generic configuration axis.

5.5.1.2 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, associate a specific color to a particular serie or category, 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.

In particular, in the 6.3 version, it has been introduced a new configuration option: the Custom Color.

With this new option it is possible to assign a specific color to a particular category and/or serie or to a particular value of a category and/or serie. Look at the following figure for an example.

To add a custom color simply write the category/serie value or name, select a color with the color piker and then click on the plus button. In the figure example it is assigned a color for each value of the ‘QUARTER’ category.

Indeed, the options available in this tab change according to the chart selected enabling different configurations. See Chart types in detail for a detailed description of the specific options of each chart.

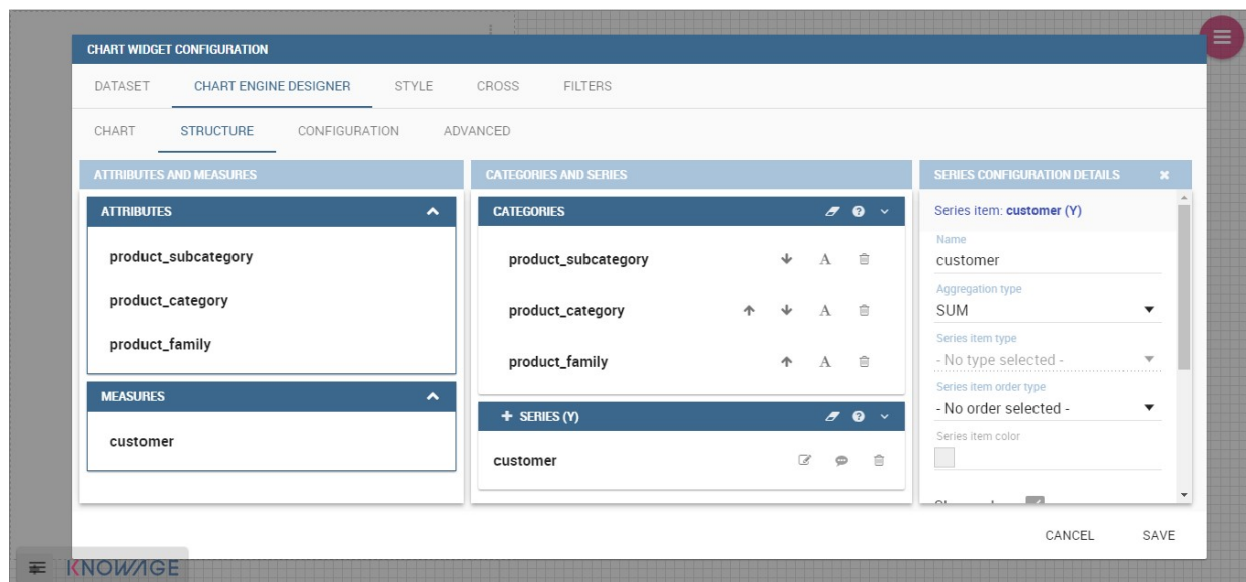


Fig. 5.87: Series style configuration.

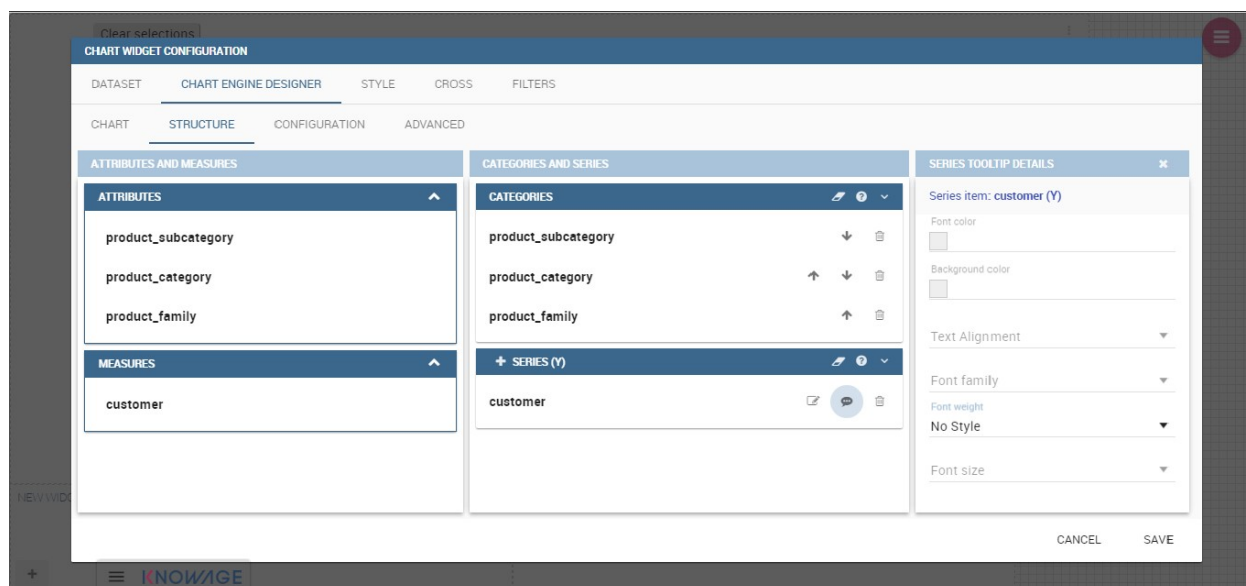


Fig. 5.88: Series tooltip details.

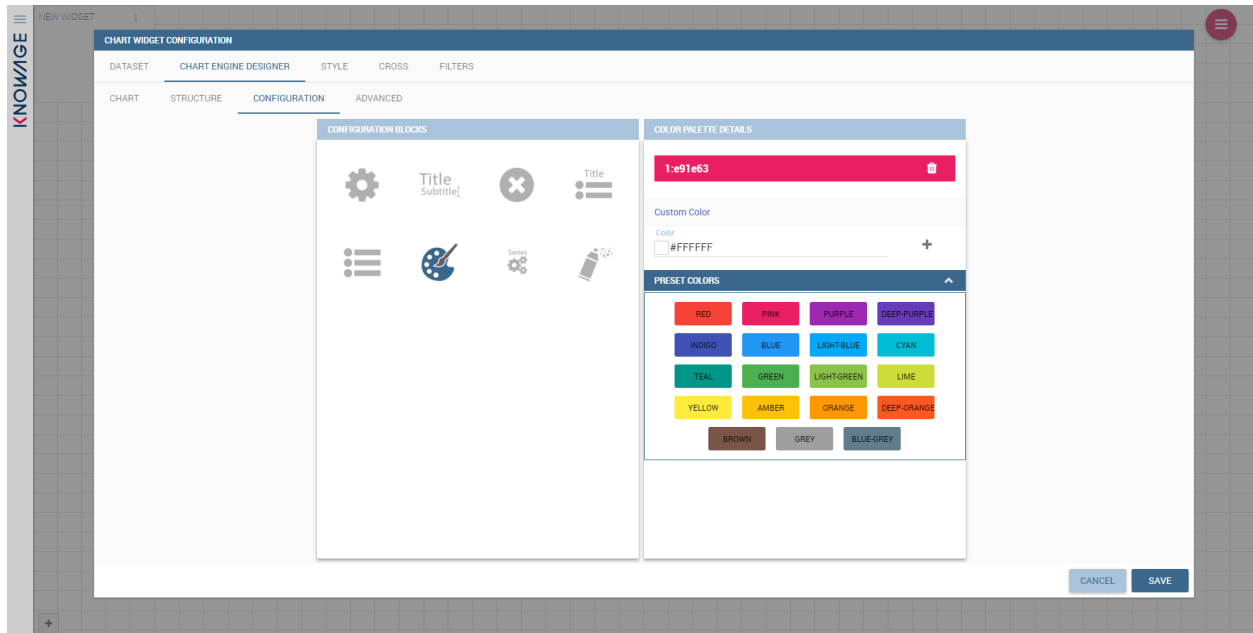


Fig. 5.89: Color box editing.

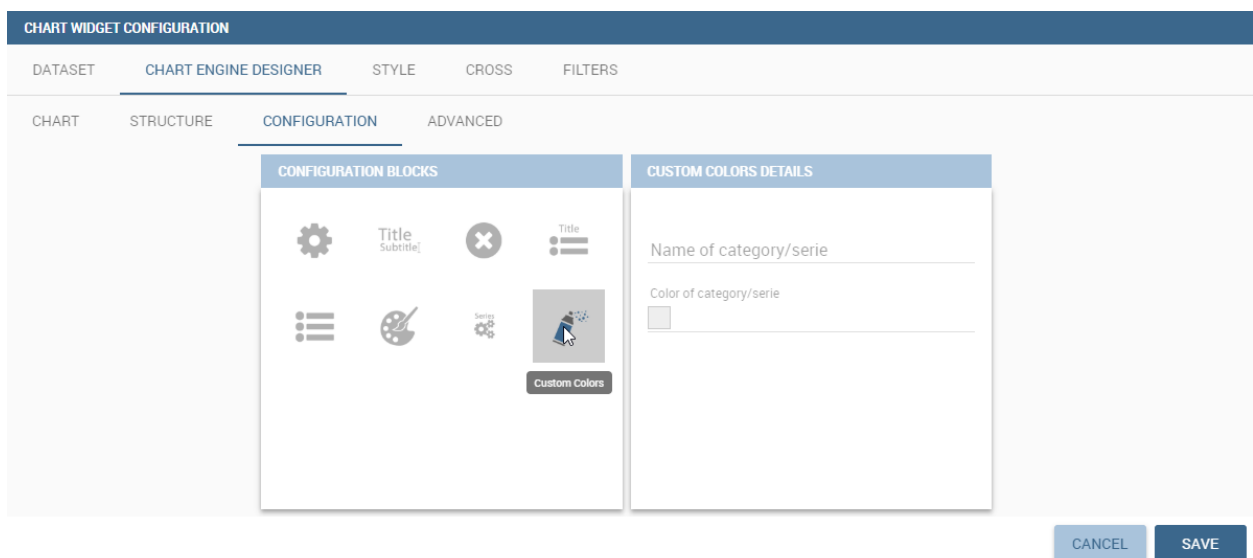


Fig. 5.90: Custom Colors details.

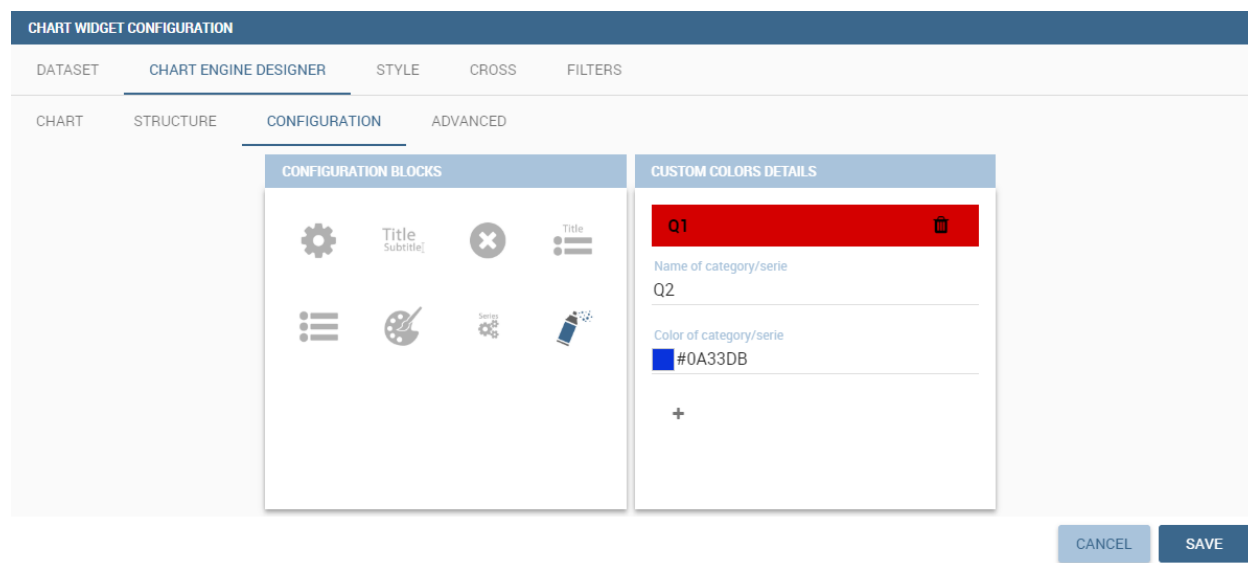


Fig. 5.91: Custom Colors example.

5.5.1.3 Advanced options

The **Advanced** tab contains some advanced options to more customize the chart. Here it is possible, for example, to set the tooltip options, the widget dimensions, if the chart is stacking or not, the grouping type.

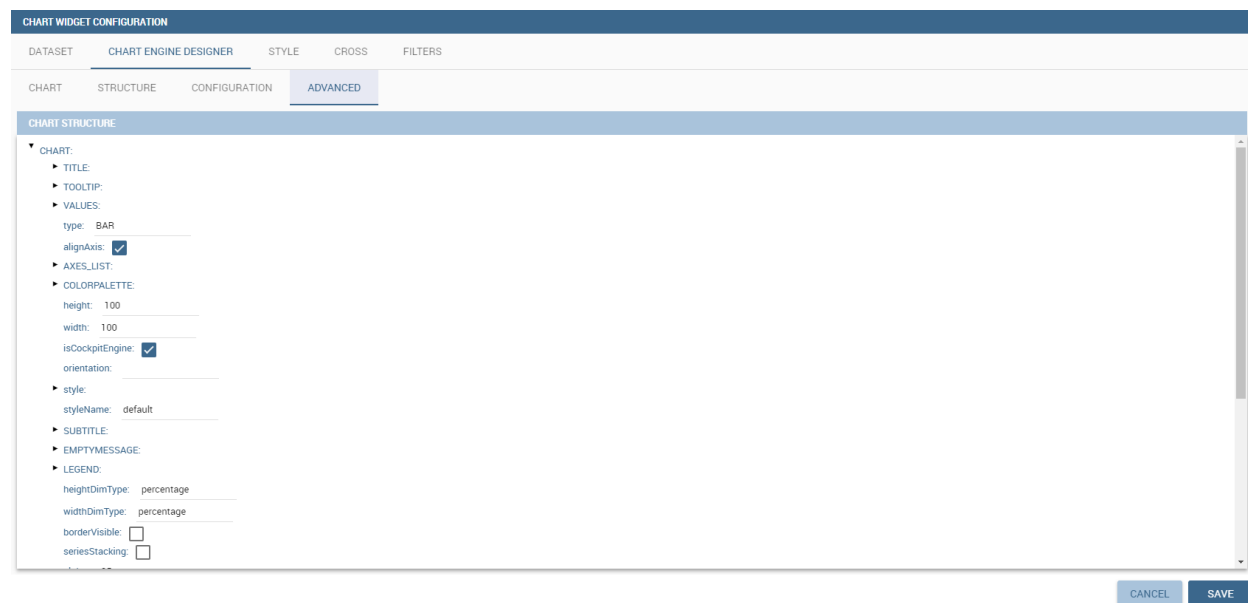


Fig. 5.92: Advanced tab.

Down here are listed some of the most useful and new options.

The **dataLabels** option can be found under the path VALUES -> SERIE -> 0 or another serie -> dataLabels. The option is available only for measures. Here it is possible to set the labels style such as the color, font family or font weight.

The **TOOLTIP** option allows to set the width and the radius of the tooltip's border.



Fig. 5.93: dataLabels option.

The **plotBands** and **plotLines** options can be found under the path **AXES_LIST** -> **AXIS** -> 0 or another serie. With these options is possible to plot respectively bands and lines on the chart with fixed values and to set their style, like the line width and the line type or the band color.

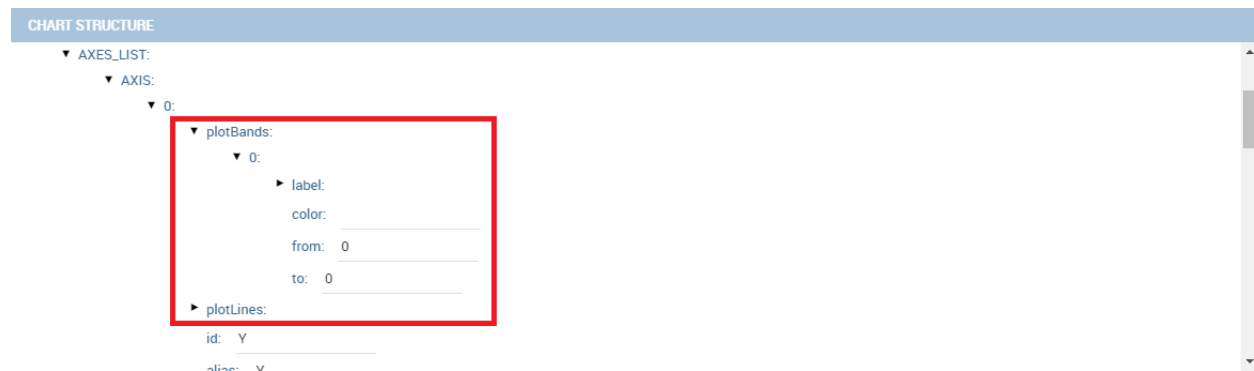


Fig. 5.94: plotBands option.

The **min** and **max** options are under the path **AXES_LIST** -> **AXIS** -> 0 or another serie. They are available only for series and allow to set the maximum and minimum axis value for the selected sere's axis.

5.5.2 Chart types in detail

This section describes the different types of chart and how to create them within the **Chart Engine** of Knowage.

5.5.2.1 Traditional charts

Knowage allows you to create the so-called traditional charts like bar, line, pie, radar and scatter chart in a fancy way.

Each chart type is built on a specific dataset. Despite all, there are some general rules that can be applied to those “simplier” and common charts. The minimum requirement is to define/have a dataset with at least one attribute column and one measure column. Then you can select the type of chart you want to use from the **Chart** section; meanwhile using the **Structure** section you can fill in the category box with one or more attributes (typically these will be place in the X-axis) and in the series box with one or more measures (typically placed as Y-axis' values). Refer to *Chart Structure* figure as example.

Once you have selected the attributes and measures you can edit the series style and axis style configurations as explained in *My first Chart*. Then go to **Configuration** to set the chart dimension, the title, the legend and to choose



Fig. 5.95: min and max options.

how to associate colors to series.

Some charts are endowed with datetime and grouping functions. In particular, it is possible to enable the grouping/splitting functions to **Bar** and **Line** charts.

The user can reach those functions just clicking on the “little arrow” located at the right end of category bar.

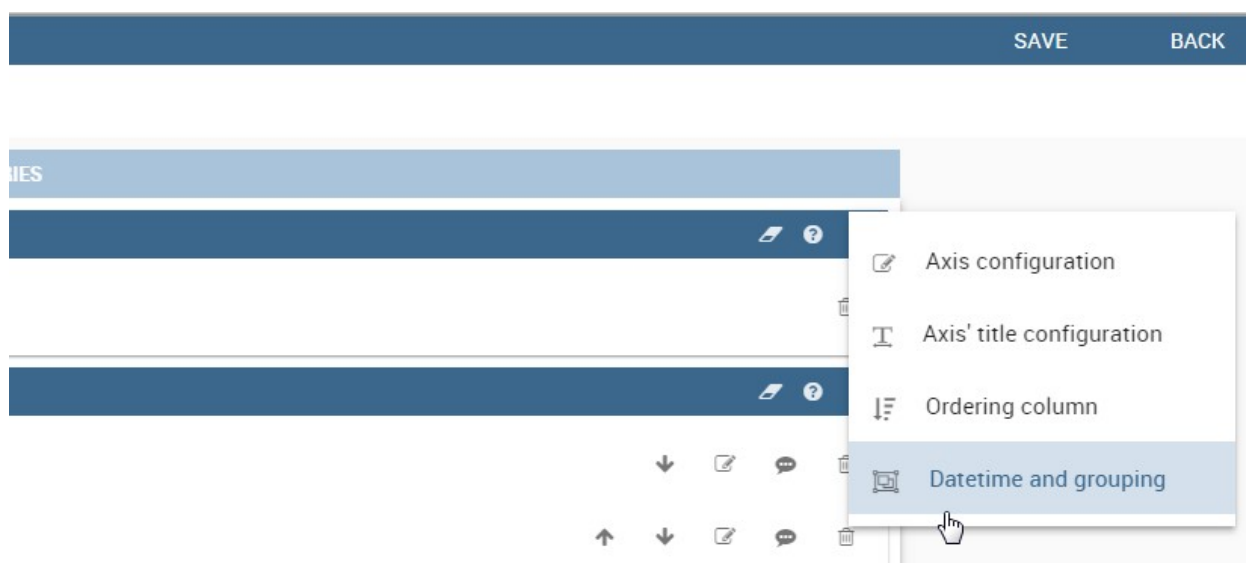


Fig. 5.96: Datetime and grouping function.

The grouping functions can be implemented only through specific categories and series configurations. As shown in figure below, the grouping function cannot be applied with just one attribute as category. To allow the function to be applied, the user must define two attributes as category fields.

As well, the user can use the splitting functions to divide one series over the second one or over the second category.

To split the first series over the second one, remember that it is necessary to choose only one attribute as category field and two measures as series values. The following figure shows an example.

Meanwhile to split a measure over second category it is mandatory to choose exactly two attributes as category field and only one measure as series value, as shown in figure below.

Futhermore, in the occurance the chart uses one datetime attribute as category field, the user can improve visualization applying the datetime function to custom date format.

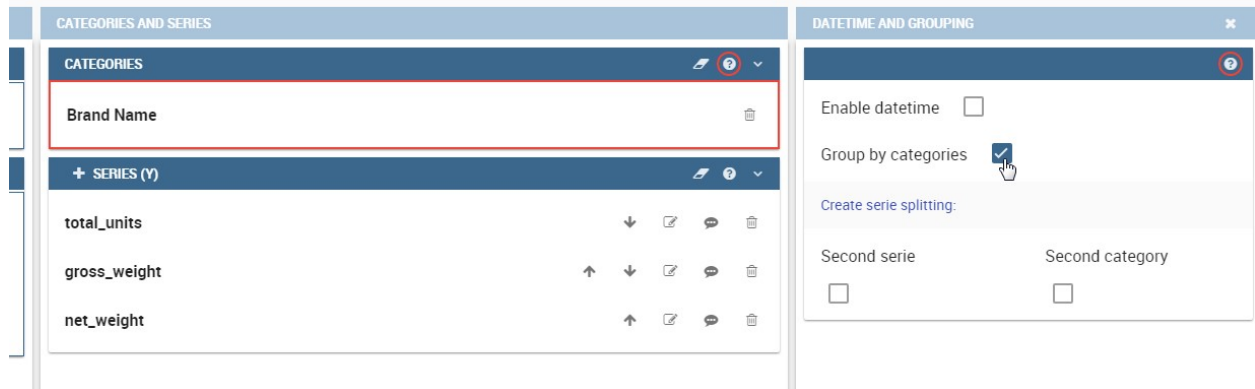


Fig. 5.97: Error alarm when enabling the grouping function.

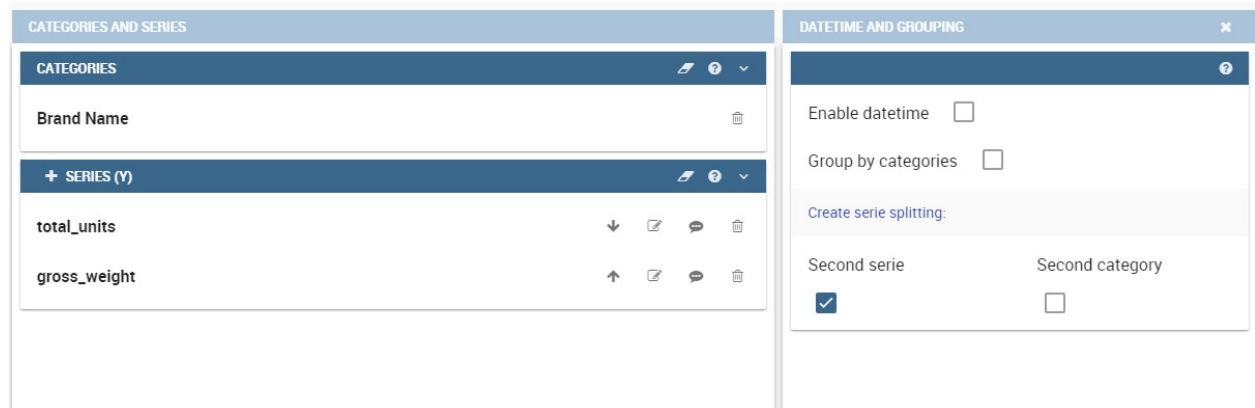


Fig. 5.98: Split over second series.

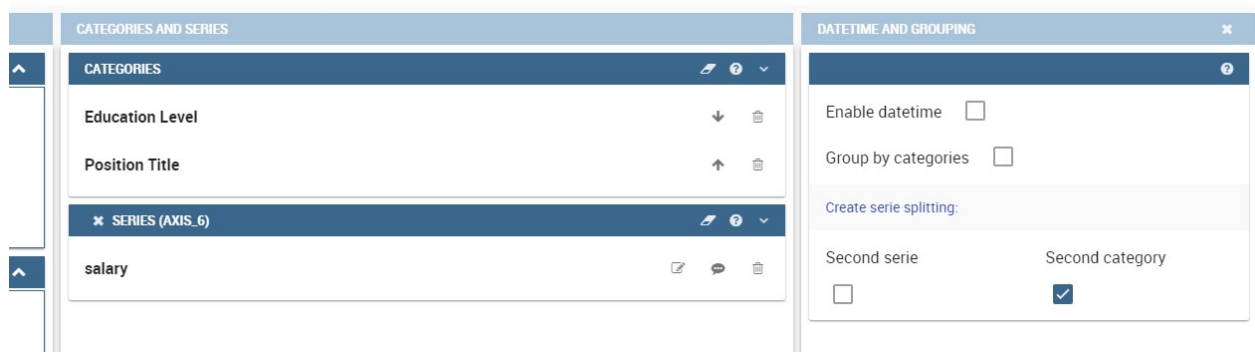


Fig. 5.99: Split over second category.

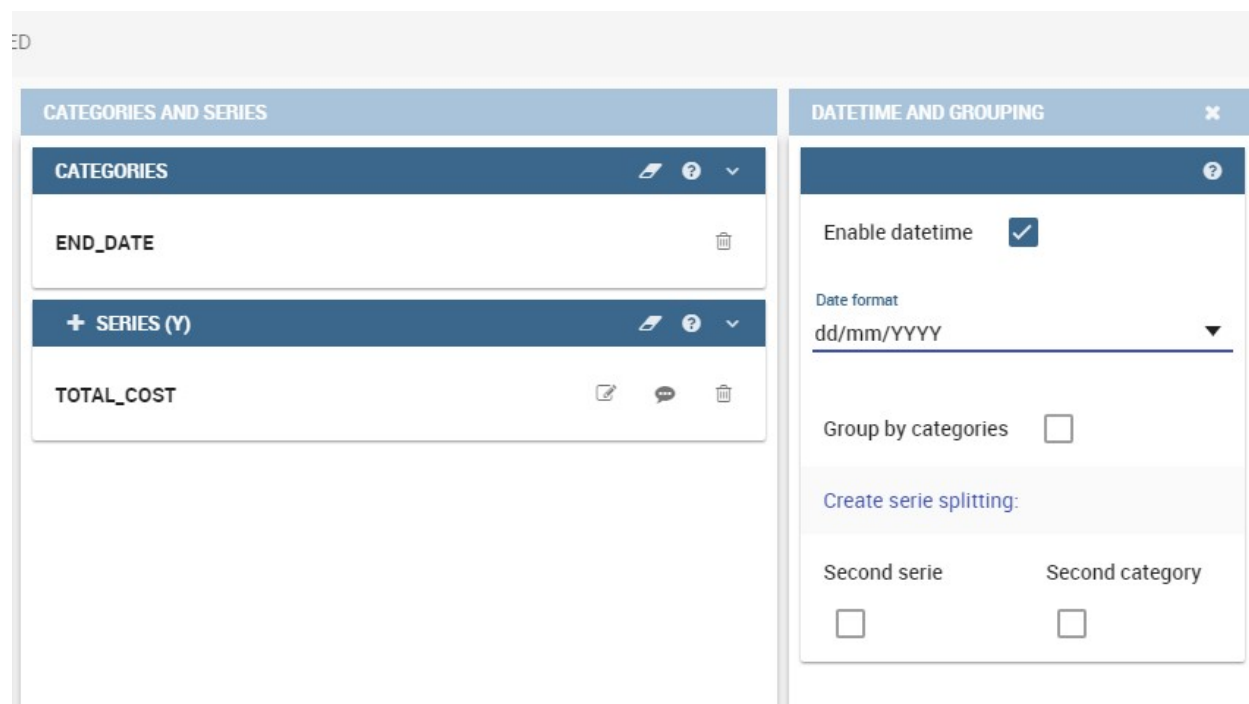


Fig. 5.100: Datetime function usage.

For bar and line chart you can add more then one container for adding series in **Structure** section. In that case you will have in your chart more then one axis for series. In **Advanced** section you can specify to align these axis to 0 (zero) value. It is check box **alignAxis** where checked means that axes will be aligned to 0, and unchecked means that they will not be aligned.

For pie chart inside **Advanced** section you can set configuration for your tooltip: to show/hide absolute value and/or percentage. Inside **tooltip** property of serie object you can find properties **showAbsValueTooltip** and **showPercentageTooltip**.

5.5.2.2 Scatter chart

A scatter chart is a graphical representation of scattering phenomenon of data. It is useful when the user wants to underlight the density of data upon certain spots to the detriment of readability of single points. If you select a scatter chart in the **Configuration** section you will have Ticks and Lables Details instead of Advanced Series Configuration. Be carefull to fill in the **Scatter configuration** with the **Zoom type**, as showed below.

You must check if you want that the values in the Y-axis start (or end) in the first (last) tick or in the first (last) value of the dataset and if you want that the last label of the category axis should be showed.

5.5.2.3 Sunburst chart

The sunburst chart is a graph with a radial layout which depicts the hierarchical structure of data displaying a set of concentric rings. The circle in the center represents the root nodes, with the hierarchy moving outward from the center. The slices in the external rings are children of the slice in the inner circle which means they lie within the angular sweep of the inner circle. The area of each slice corresponds to the value of the node. Even if sunburst charts are not efficient space-wise, they enable the user to represent hierarchies in a more immediate and fascinating way.

To create a sunburst chart in Knowage you just have to select a dataset with at least two attribute columns describing

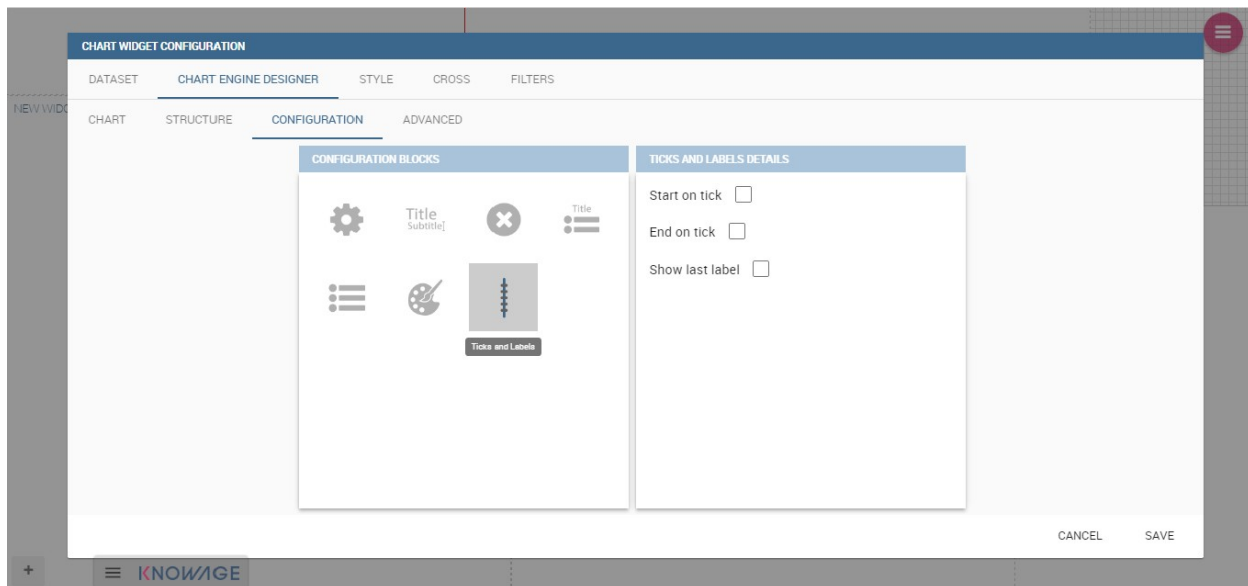


Fig. 5.101: Scatter Chart, ticks and labels details.

the hierarchy and at least a measure column that indicates the width of the slices. An example of dataset for the sunburst chart is showed in Table below.

Table 5.5: Example of dataset for the sunburst chart.

CATEGORY	SUBCATEGORY	UNIT
Baking Goods	Cooking Oil	349
Baking Goods	Sauces	109
Baking Goods	Spices	290
Baking Goods	Sugar	205
Bathroom Products	Conditioner	64
Bathroom Products	Mouthwash	159
Bathroom Products	Shampoo	254
Bathroom Products	Toilet Brushes	92
Bathroom Products	Toothbrushes	94

Once you selected the dataset and the type of chart, choose at least two attributes in the X-axis panel and a measure in the Y-axis panel as showed in the following figure.

Then click on **Configuration**. As you can see the features are not exactly the same as traditional chart. We give some tips on most important sunburst settings.

Using the **Generic** button you can set the **opacity** on mouse movement and choose how to display the measure values: absolute, percentage or both. These two features allow the visualization of data just moving the mouse over the slice: the slice is highlighted and values are shown in the center of the ring while the root-node path for the node selected is displayed on the left bottom corner of the page. Opacity and Breadcrumb configuration are available only on Community Edition. The tooltip is a mandatory field since it shows the value of the selected slice. Therefore be sure to have filled it before saving by using the **Explanation detail** panel. On Community Edition you have option custom the root-node path, clicking on the **Sequence** icon and choose position, label tail size and text style. **Sequence** option is not available on Enterprise edition, it is deprecated. Figure below sums up the three features.

In Figure below you find the sunburst obtained with data of [Table 5.5](#).

Inside **Advanced** section you can set value for scale that will increase/decrease your chart. You need to set numeric

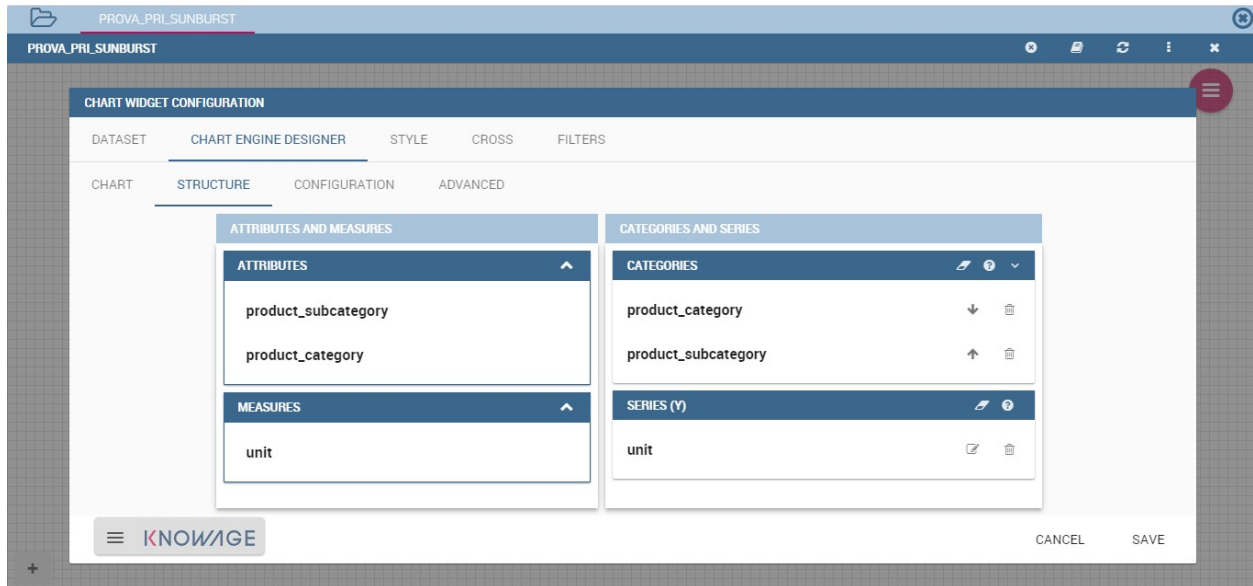


Fig. 5.102: Sunburst configuration.

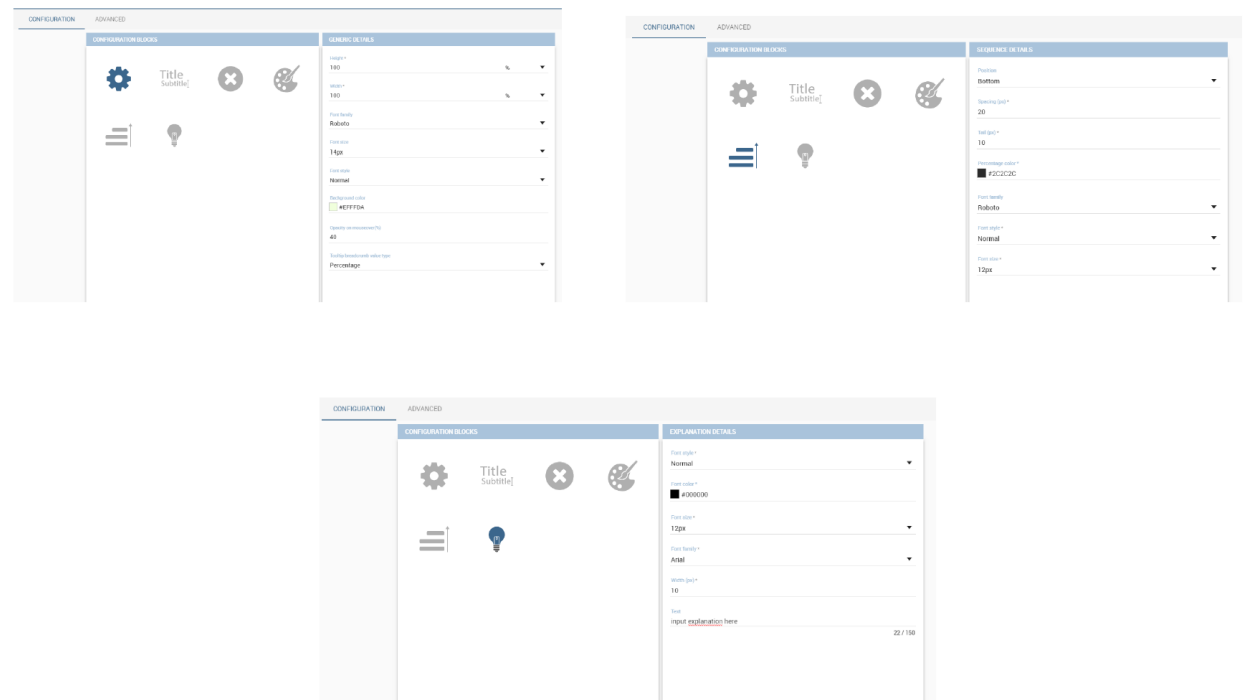


Fig. 5.103: Generic, Sequence and Explanation configuration

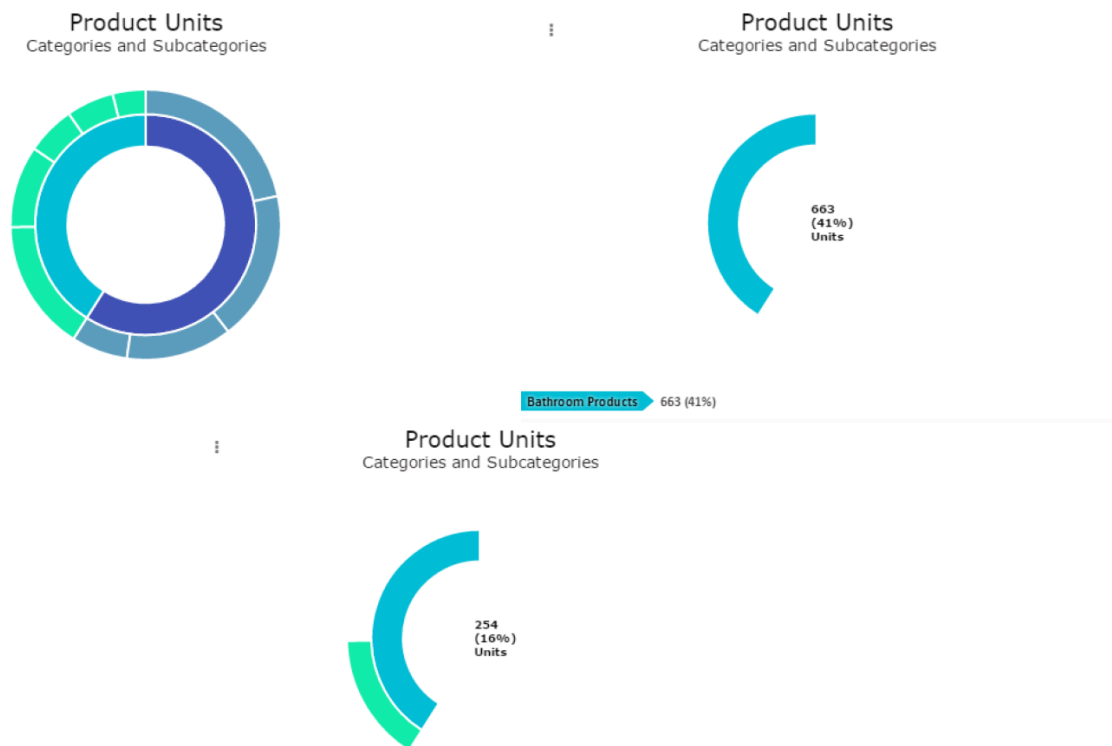


Fig. 5.104: From left to right: (a) Sunburst. (b) Sunburst category.(c) Sunburst subcategory.

value for property **scale**.

5.5.2.4 Wordcloud chart

The wordcloud chart is a graphic to visualize text data. The dimension of the words and colors depend on a specified weight or on the frequency of each word.

The dataset to create a wordcloud should have at least a column with attributes and only one column with numerical data which represents the weight to assign to each attribute. Choose one attribute as category field (the wordcloud accept only one attribute in the category box) and a measure as series field.

Switch to the **Configuration** section to set the generic configuration of the chart and to custom fields of the **Word settings details**. Here the user can decide if to resize the words accordingly to the measure retrieved in the dataset (**Series** option) or accordingly to the frequency of the attributes in the dataset (**Occurrences** option). Moreover it is possible to set the maximum number of words that you want to display, the padding between the words, the word layout and whether or not you want to prevent overlap of the words as showed in Figure below.

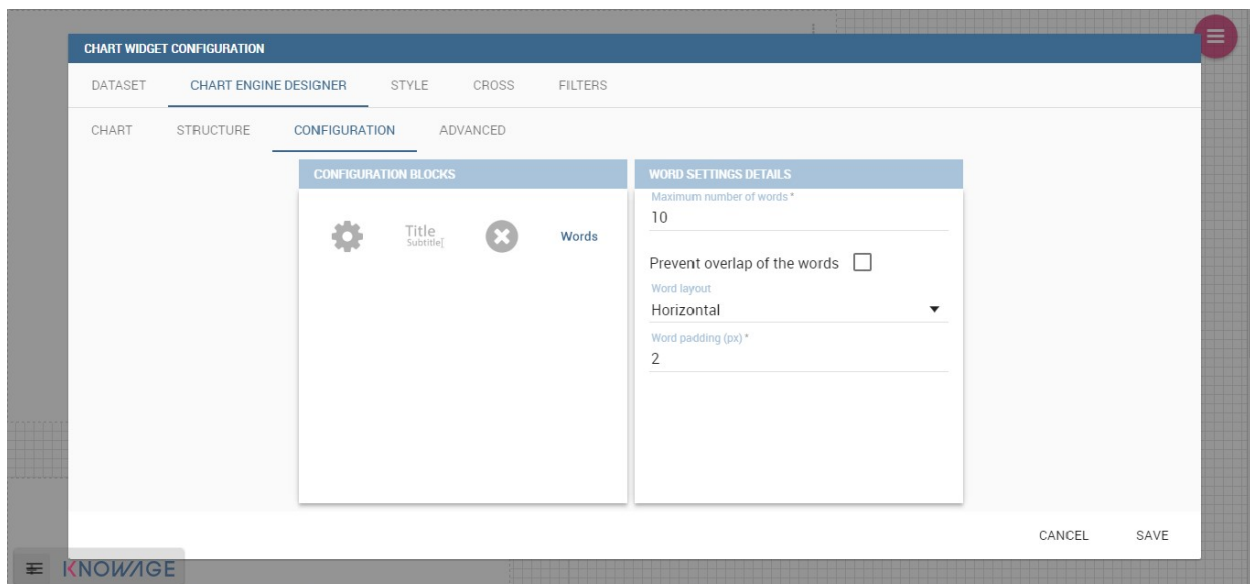


Fig. 5.105: Wordcloud chart specific configuration.

5.5.2.5 Treemap chart

The treemap is a graphical representation of hierarchical data, which are displayed as nested rectangles. Each branch of the tree is given by a rectangle, which is tiled with smaller rectangles representing sub-branches. The area of the rectangles is proportional to a measure specified by a numerical attribute. The treemap is useful to display a large amount of hierarchical data in a small space.

To create a treemap chart you have to select a dataset as the one described for the sunburst chart in the Parallel chart.

Once you have selected the dataset, choose the treemap chart type in the designer and then at least two attributes into the X-axis panel. The order of the attributes in the X-axis panel must reflect the order of the attributes in the hierarchy starting from the root to the top.

Finally you can set generic configurations and colors palette in the **Configuration** tab and advanced configurations in **Advanced editor** tab.

In Figure below we show the Treemap resulting with data of our example

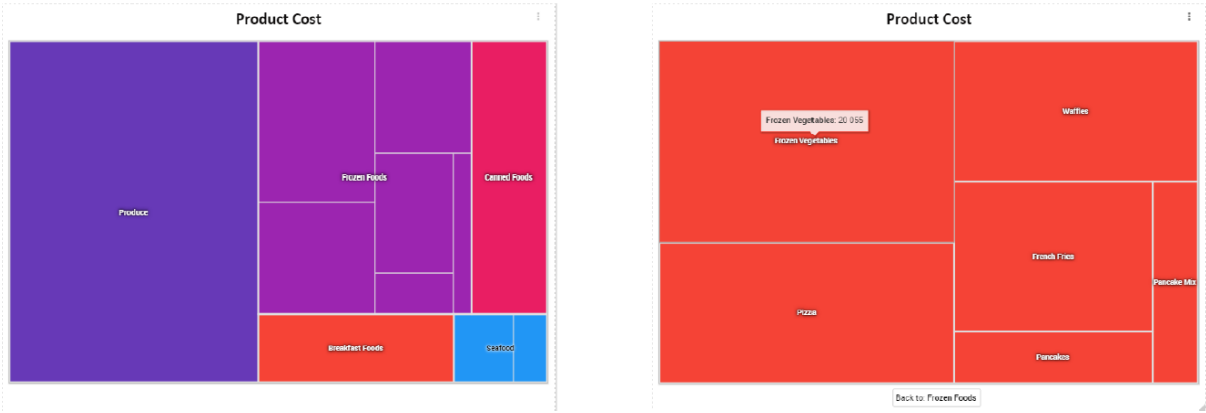


Fig. 5.106: From left to right: (a) Treemap. (b) Treemap sub-branches.

5.5.2.6 Parallel chart

The parallel chart is a way to visualize high-dimensional geometry and multivarious data. The axes of a multidimensional space are represented by parallel lines, usually equally spaced-out, and a point of the space is represented by a broken line with vertices on the parallel axes. The position of the vertex on an axis correspond to the coordinate of the point in that axis.

To create a parallel chart select a dataset with at least one attribute and two columns with numerical values. You can find an interesting example of dataset in the next table where we display some of its rows.

Table 5.6: Example of dataset for the parallel chart.

ID	sepal_length	sepal_width	petal_length	petal_width	class
36	5.0	3.2	1.2	0.2	Iris-setosa
37	5.5	3.5	1.3	0.2	Iris-setosa
38	4.9	3.1	1.5	0.1	Iris-setosa
39	4.4	3.0	1.3	0.2	Iris-setosa
40	5.1	3.4	1.5	0.2	Iris-setosa
41	5.0	3.5	1.3	0.3	Iris-setosa
42	4.5	2.3	1.3	0.3	Iris-setosa
43	4.4	3.2	1.3	0.2	Iris-setosa
44	5.0	3.5	1.6	0.6	Iris-setosa
45	5.1	3.8	1.9	0.4	Iris-setosa
66	6.7	3.1	4.4	1.4	Iris-versicolor
67	5.6	3.0	4.5	1.5	Iris-versicolor
68	5.8	2.7	4.1	1.0	Iris-versicolor
69	6.2	2.2	4.5	1.5	Iris-versicolor
70	5.6	2.5	3.9	1.1	Iris-versicolor
71	5.9	3.2	4.8	1.8	Iris-versicolor
101	6.3	3.3	6.0	2.5	Iris-virginica
102	5.8	2.7	5.1	1.9	Iris-virginica
103	7.1	3.0	5.9	2.1	Iris-virginica
104	6.3	2.9	5.6	1.8	Iris-virginica
105	6.5	3.0	5.8	2.2	Iris-virginica
106	7.6	3.0	6.6	2.1	Iris-virginica
107	4.9	2.5	4.5	1.7	Iris-virginica
108	7.3	2.9	6.3	1.8	Iris-virginica

In this example three different classes of iris are studied. Combining the values of some sepal and petal width or length, we are able to find out which class we are looking at. In Figure below (a part) you can find the parallel chart made with the suggested dataset. While in next figure (b part) it is easy to see, thanks to selection, that all iris with petal length between 2,5 and 5,2 cm and petal width 0,9 and 1,5 cm belong to the iris-versicolor class.

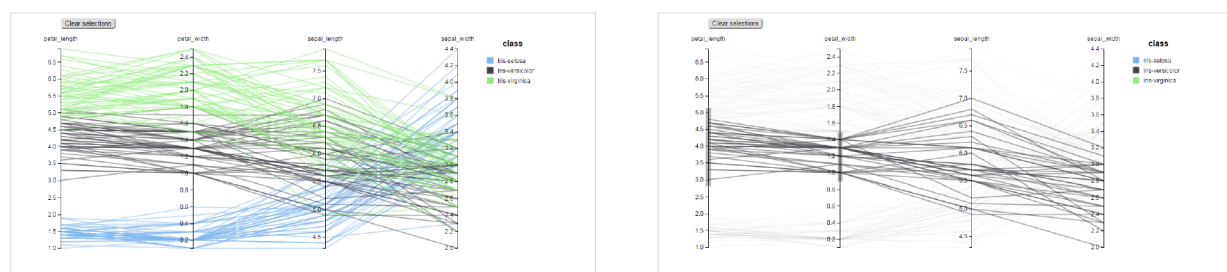


Fig. 5.107: From left to right: (a) Parallel. (b) Parallel chart selection.

Therefore, select **parallel** as chart type using the designer interface, then choose one or more attributes in the X-axis panel and one or more measures in the Y-axis panel.

On the **Configuration** tab you can set the generic configuration for the chart and you must fill the **Series as filter column** filed under "Limit configuration". Under "Tooltip configuration" there is new property available - **Maximum number of records to show tooltip**. It is used to limit showing tooltip in case there are lot of records returned from dataset, which make chart more readable.

5.5.2.7 Heatmap chart

Heatmap chart uses a chromatic Cartesian coordinate system to represent a measure trend. Each point of the Cartesian system is identified by a couple of attributes. Note that one attribute must be a datetime one. Meanwhile, each couple corresponds to a measure that serves to highlight the spot with a certain color according to the chosen gradient. Figure below gives an example of how an heatmap chart looks like inside Knowage.

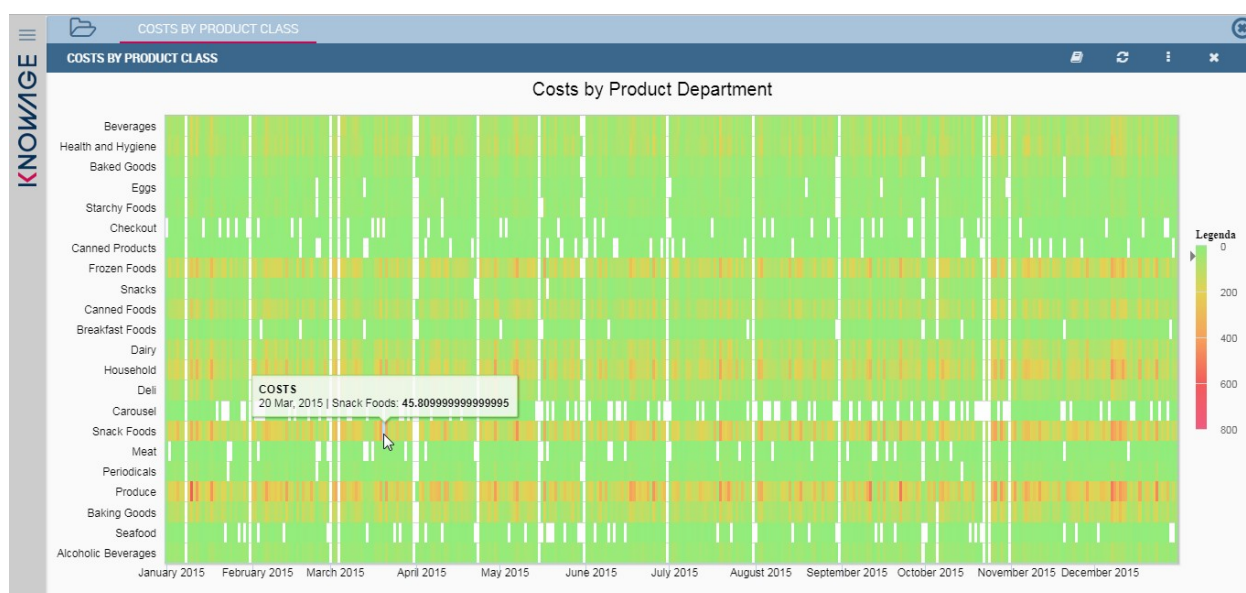


Fig. 5.108: Heatmap example.

Before configuring a heatmap chart, be sure that your dataset returns at least two attributes, one of which **must** be a datetime one, and (at least) one measure. Once entered the chart designer, choose the “Heatmap” type and move to the “Structure” tab. Use the datetime attribute and an other attribute as category fields and one measure as series fields. Figure below shows an example.

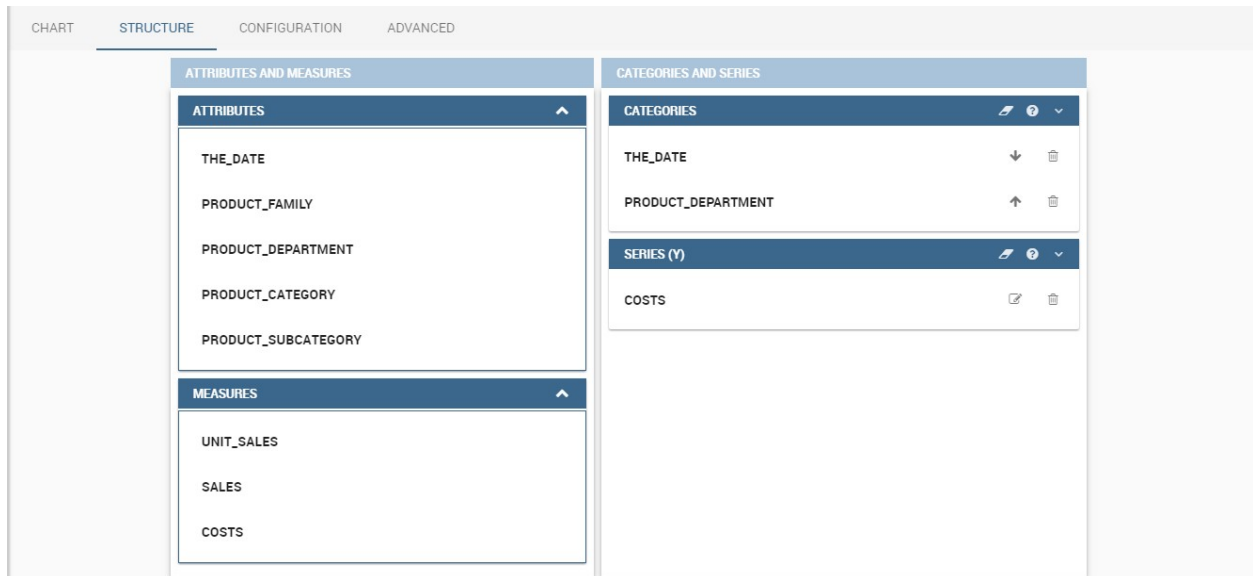


Fig. 5.109: Configuring the attributes and the series for the heatmap chart.

Note that for series axis it is possible to specify the values’ range by assigning a minimum and the maximum value, as shown in figure below. Otherwise, the engine will automatically link the axis scale to dataset results set.

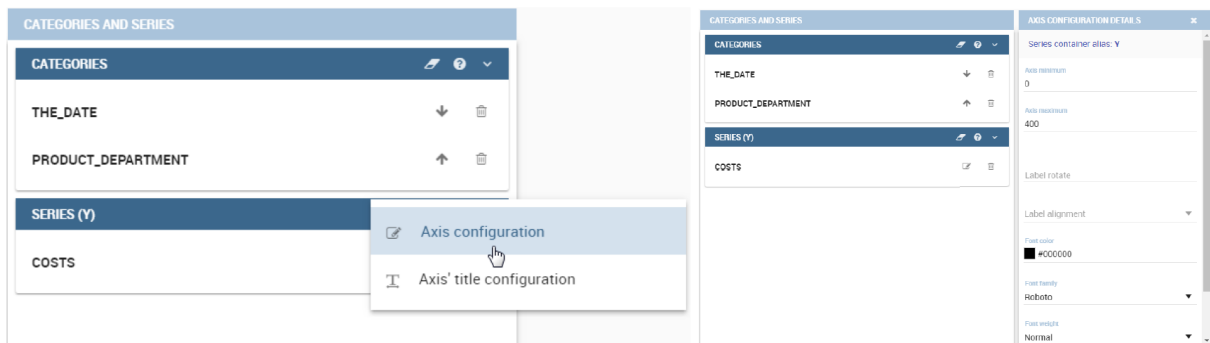


Fig. 5.110: Configure min and max values for series.

The next step is to move to **Configuration** tab and select the **Color palette** icon. Here (figure below) the user has to define the chromatic scale which will be associated to the measure values. The panel will demand the user to insert the first, the last color and the number of bands that will constitute the color scale.

The engine will create a progressive color scale as shown in the left image of figure below. To custom the scale the user can use the Preset colors and use the arrow to move up and down Heatmap chart the added color or the user can increase the number of steps and then some intermediate color to leave more contrast between them.

Remember to edit both **Legend** and **Tooltip** configuration in the **Tooltip details** panel to improve the readability of the chart.

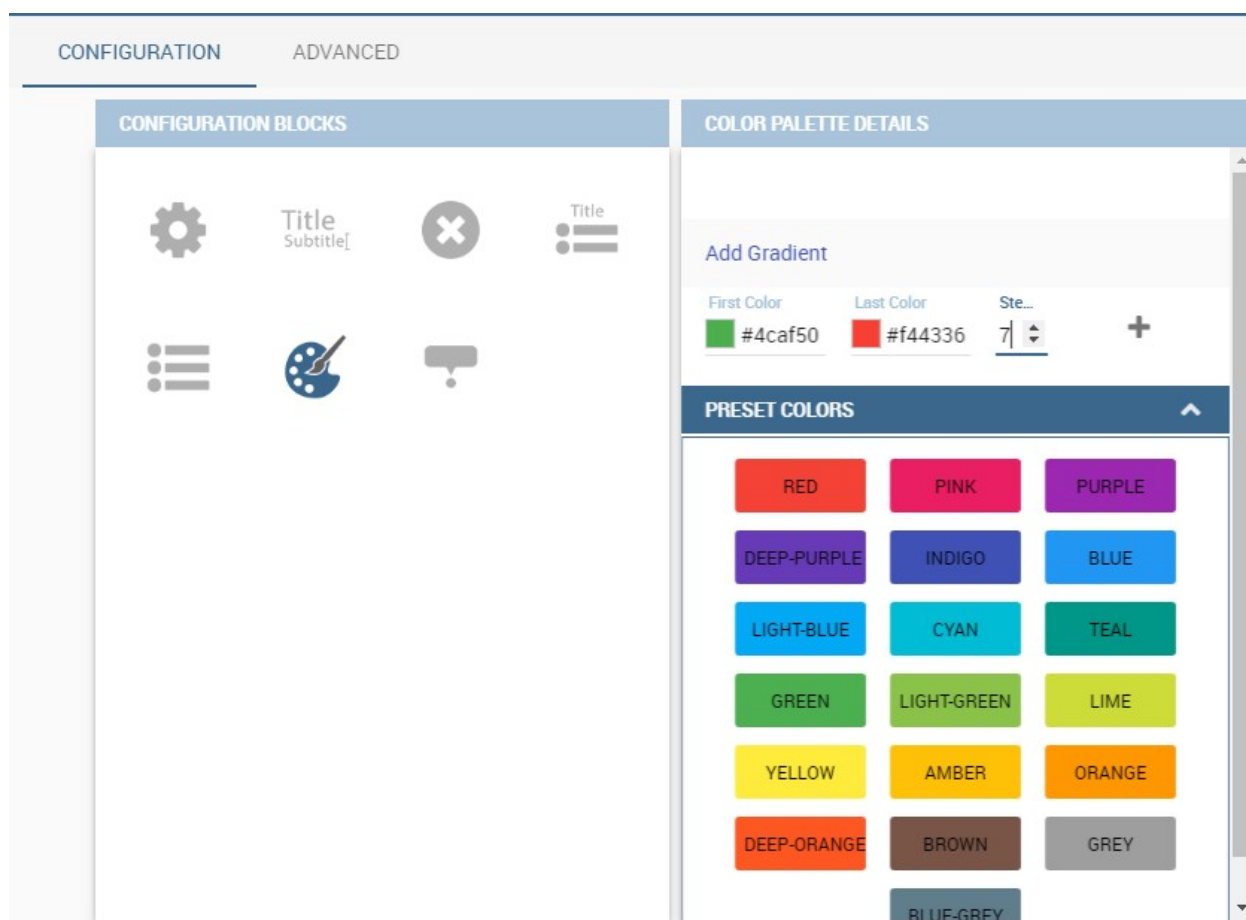


Fig. 5.111: Add gradient panel.

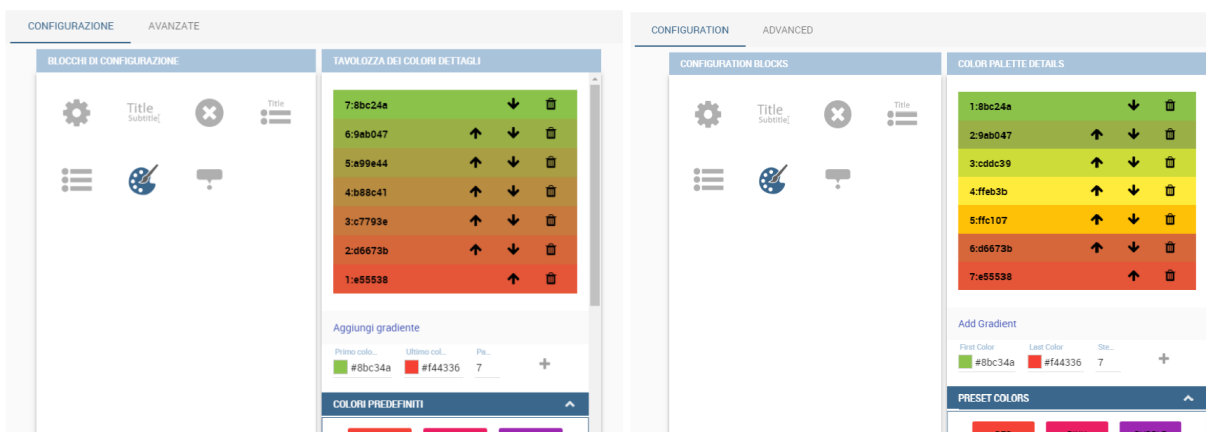


Fig. 5.112: Custom color scale.

5.5.2.8 Chord chart

Chord diagram is a graph which allows to show relationship between entities and between data in a matrix. The entities can belong to an unique category while the arc be non-oriented or belong to two different categories. In this latter case, they have directed arcs. The data are arranged radially with arcs that represent the connection between points. The width of the arc connecting two points depends on the weight assigned to the edge connecting these two points. This graphic is usefull when you want to represent a large number of data in a small space.

The chord diagram requires a dataset that have a column with numerical values. These represent the weight of the arc connecting two points. It also must have two columns with the entries for the entities to be connected in the diagram. These two columns must have the same set of values so that the engine can understand the relation between all the entities. If there is not a relation between two entities the weight of the arc is zero. Note that when you create a directed chord diagram with two different categories, all the relations between entities of the same category have a zero weight.

An example of dataset for the chord chart is represented in Table below:

Table 5.7: Example of dataset for the chord chart.

CUSTOMER_ CITY	STORE_ CITY	VALUE
Beaverton	Portland	4609.0000
Lake Oswego	Portland	4201.0000
Milwaukie	Portland	5736.0000
Oregon City	Portland	3052.0000
Portland	Portland	3984.0000
W. Linn	Portland	3684.0000
Albany	Salem	5544.0000
Corvallis	Salem	8542.0000
Lebanon	Salem	8015.0000
Salem	Salem	6910.0000
Woodburn	Salem	6335.0000
Albany	Albany	0.0000
Beaverton	Beaverton	0.0000
Corvallis	Corvallis	0.0000
Lake Oswego	Lake Oswego	0.0000
Lebanon	Lebanon	0.0000
Milwaukie	Milwaukie	0.0000
Oregon City	Oregon City	0.0000
Portland	Portland	0.0000
Salem	Salem	0.0000
W. Linn	W. Linn	0.0000

Once you have selected the dataset open the designer and select chord chart type. Then choose the two entities in the X-axis panel and the value in the Y-axis panel as showed in figure below. Now you are ready to customize the chart setting the generic configuration and the palette on **Configuration**.

5.5.2.9 Gauge chart

Gauge chart uses needles to show information as a dial reading. It allows to visualize data in a way that resembles a real-life speedometer needle. The value of the needle is read on a colored data scale. Colors are used to provide additional performance context (typically green for good and red for bad). This chart type usually is used in dashboards to show key performance indicators or any measure having reference values.

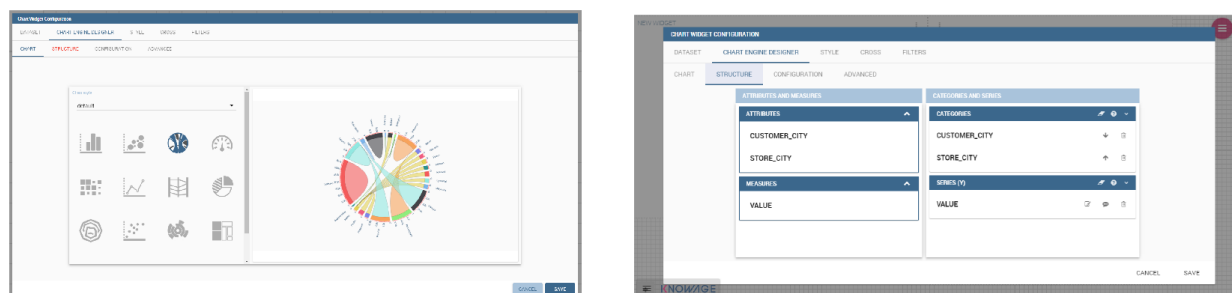


Fig. 5.113: Chord configuration.

For gauge chart you should have only series items, the one that gives you values for the chart. So, the defined dataset to be used should provide numerical data for the Y-axis for the gauge chart. After selecting the dataset go to the designer and select **gauge** in chart type combobox. Then choose one or more measure on the Y-axis panel on the **Structure**. Moreover you must not forget to provide all data needed for the **Axis style configuration** of the Y-axis.

When you finished to set all the mandatory and optional parameters and configurations in the **Structure** tab you can select the **Configuration** tab and set the generic configuration of the chart.

5.5.2.10 Bubble chart

A bubble chart requires three dimensions of data; the x-value and y-value to position the bubble along the value axes and a third value for its volume, z-value. It is a generalization of the scatter plot, replacing the dots with bubbles.

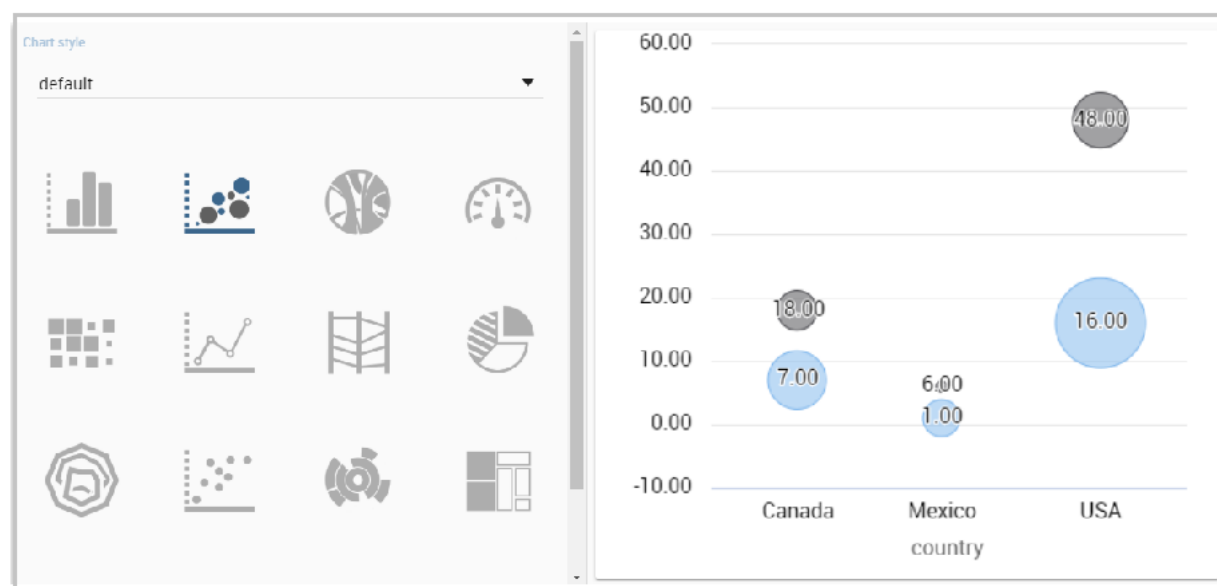


Fig. 5.114: Bubble chart.

Inside X,Y,Z containers, user can put only **measure values**. Inside Categories container user can put **attributes** that he wants to see in the **tooltip**.

Inside **Advanced** tab, user can find configuration for **plotband** and **plotline** of xaxis and yaxis.

User also has option for **splitting serie** by one of two categories. Split option has limit on **maximum** two categories. User can set which category will be used for coloring bubbles. And also can show/hide values that are inside bubbles.

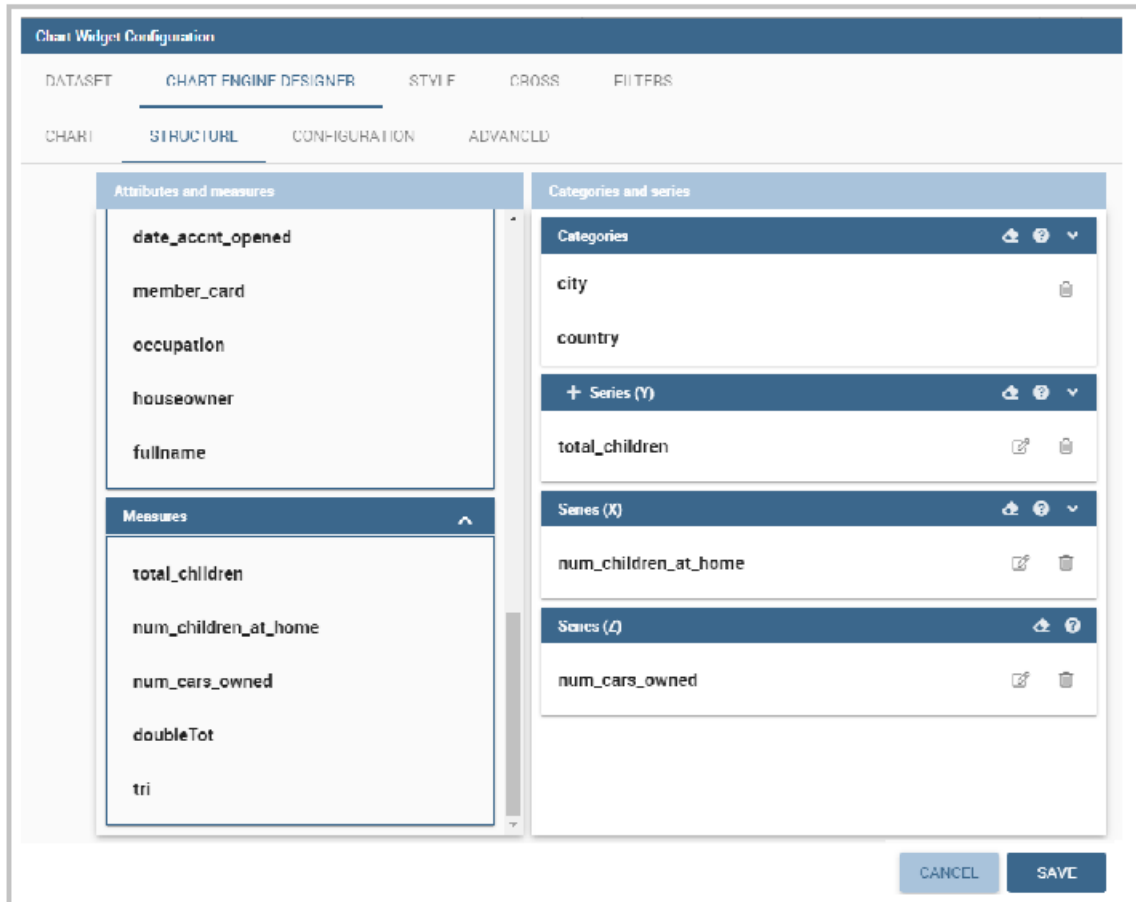


Fig. 5.115: Bubble configuration.

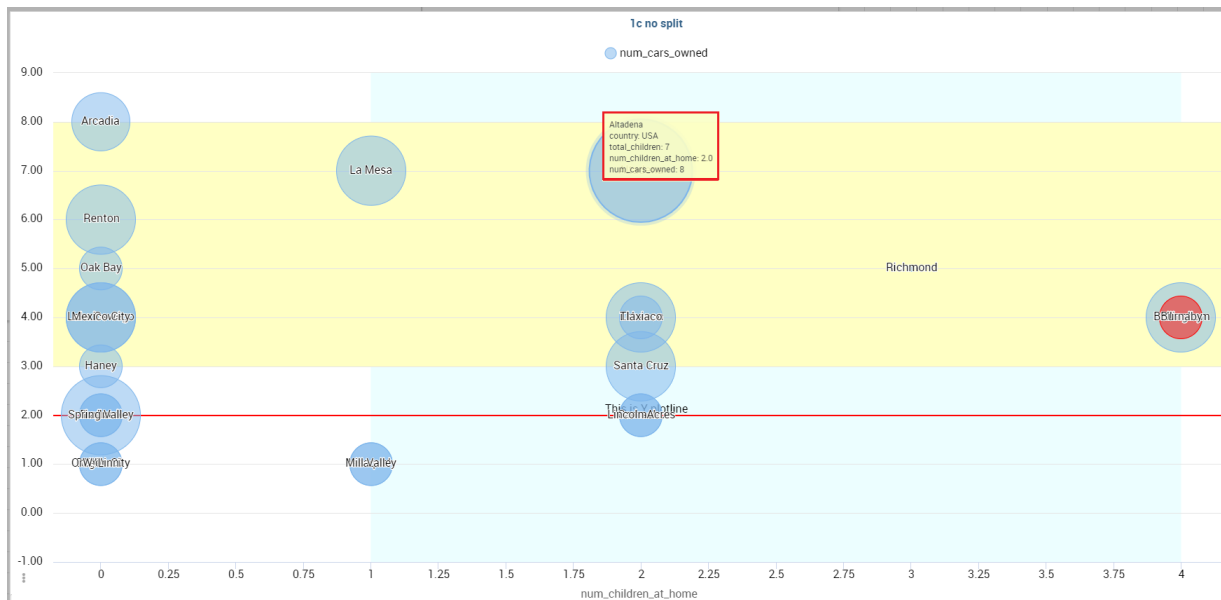


Fig. 5.116: Bubble tooltip.

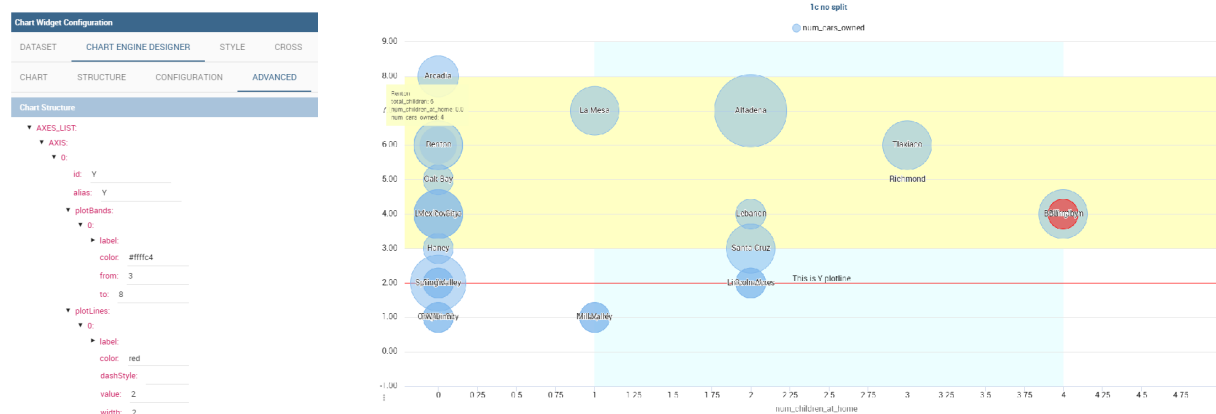


Fig. 5.117: Bubble plotband and plotline configuration.

Categories and series

Categories	Ordering column	Splitting
city	Ordering column	Splitting
country		

+ Series (Y)

total_children

Series (X)

num_children_at_home

Series (Z)

num_cars_owned

Serie splitting

Create serie splitting:

Split ☒

Colored category: country

Series to use on second category options:

Show category value ☒

Fig. 5.118: Bubble split configuration.



Fig. 5.119: Bubble split examples.

Difference between **One category - split disabled** and **Two categories - split disabled** is what is present in the tooltip.

5.5.3 A short comment on chart drill down

Knowage **Chart Engine** allows you to drill down into categories. This means that the user can explore the details of each category as many times as configured. Indeed, to let the chart admits the drill down, it is necessary first that the chart in place allows it. Secondly the user must have dragged and dropped multiple attributes into the category axis in the **Configuration** tab. The order of the attributes in the X-axis panel determines the sequence in which the categories are going to be showed. When executing the chart the label of the category is linkable and it is possible to click on the label to drill down.

The chart that enables the drill down are:

- Bar Chart
- Line Chart
- Pie Chart
- Treemap

To give an idea of the outcome, we take as instance the Bar Chart drill down. In the following example, the selected categories are four and called: `product_family`, `product_department`, `product_category` and `product_subcategory`. Once we open the document, we get as shown below:

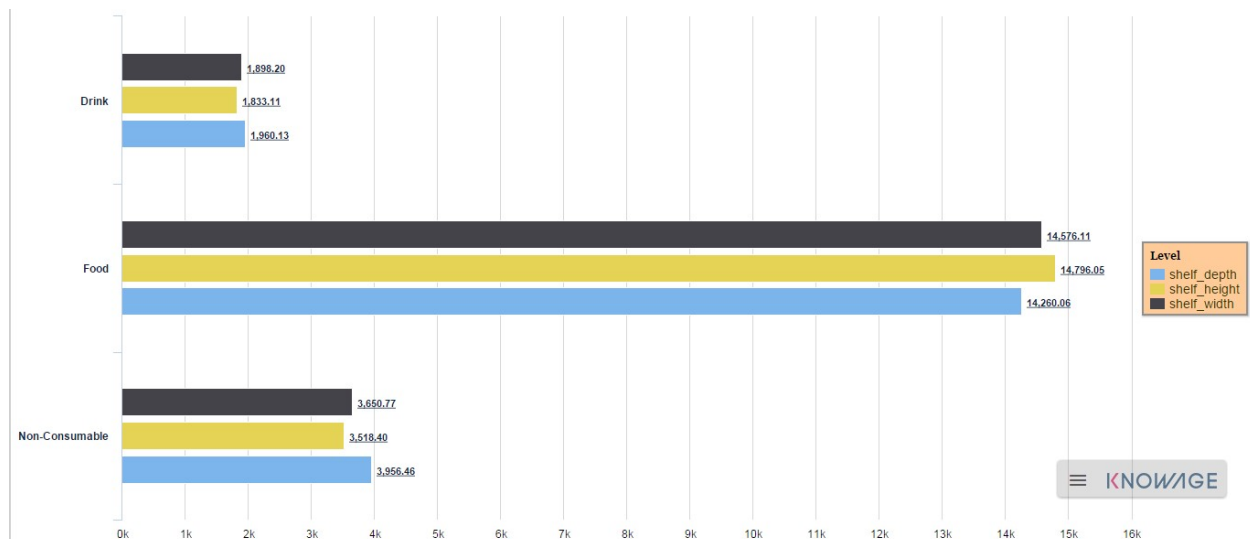


Fig. 5.120: Drillable Bar Chart

When selecting `shelf_depth` measure of the Food category one gets (see next figure):

Once again, we can select `Frozen` food subcategory and drill to a second sub-level as below:

And so on to the fourth subcategory. Selecting the “Back to: ...” icon available at the right corner of the graphic, the user can get back to the previous level. This efficient feature allows the user to have a deep insight of the analysis and draw important conclusions from it.

5.5.4 Stand alone charts

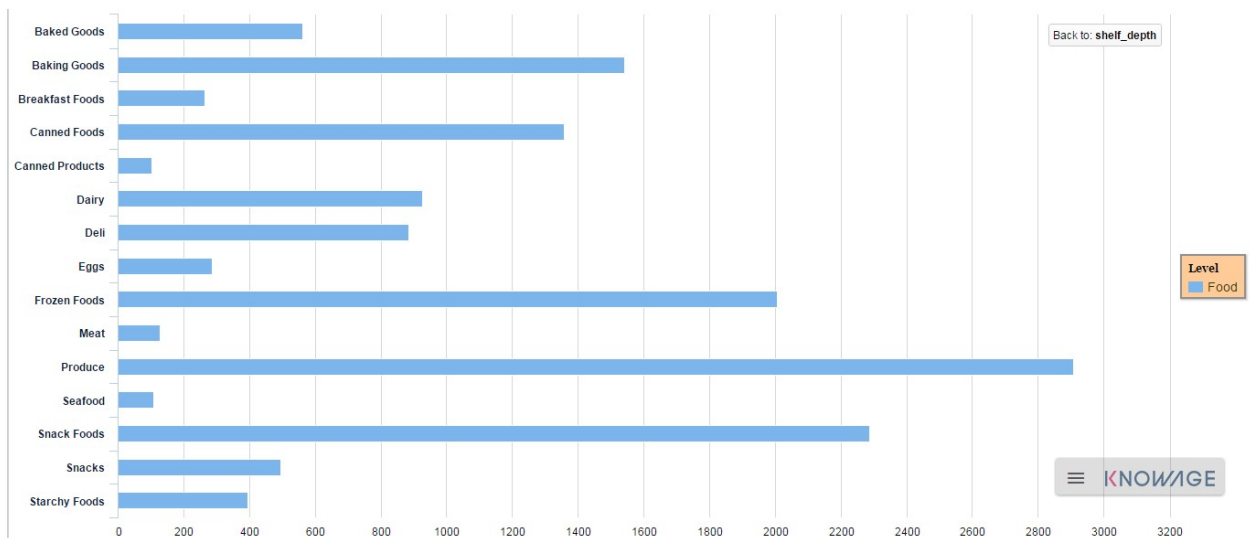


Fig. 5.121: Drillable Bar Chart: first drill

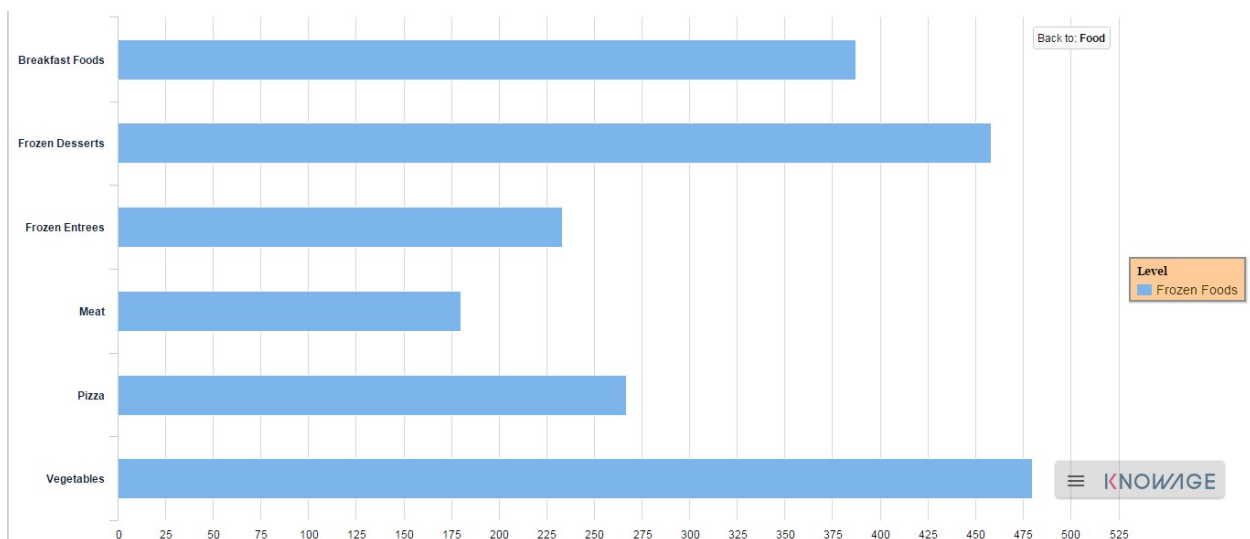


Fig. 5.122: Drillable Bar Chart: second drill

Warning: This functionality is deprecated

Design of stand alone charts is **deprecated** and may be removed in future releases, therefore we invite users to exploit the cockpit designer for charts instead.

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 below.

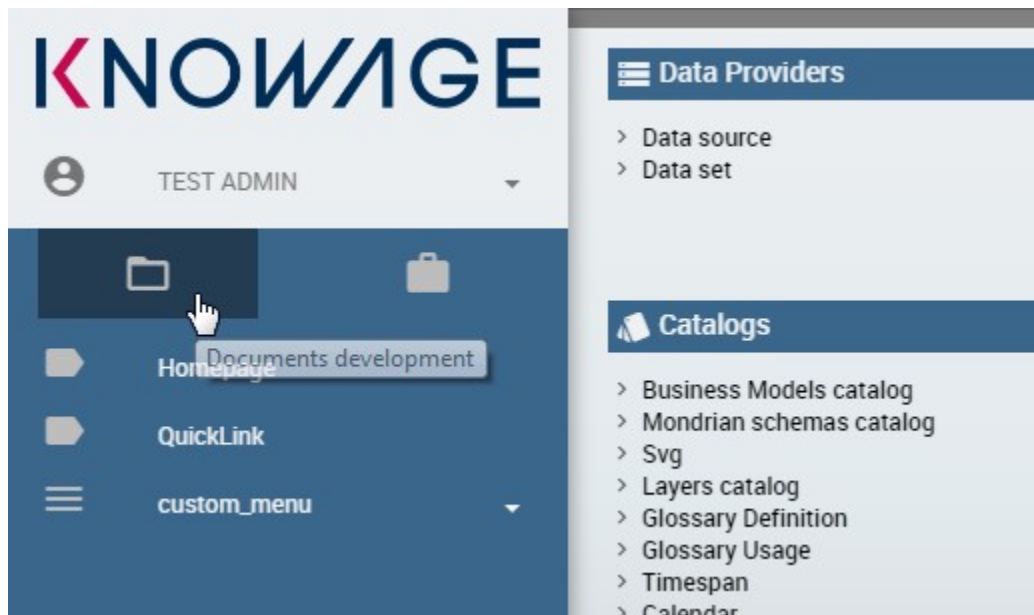


Fig. 5.123: Documents Development.

Then click on the “Plus” icon of the **Create Document** feature and select **Generic Document**.

You will be asked to fill in the form. We give an example in the following figure.

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 next figure) and finally save.

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

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.

5.5.5 Cross Navigation

When you develop a standalone chart it is possible to add a cross navigation path to it. This means that, once the chart is launched, its elements becomes clickable and it redirects the user to a second document.

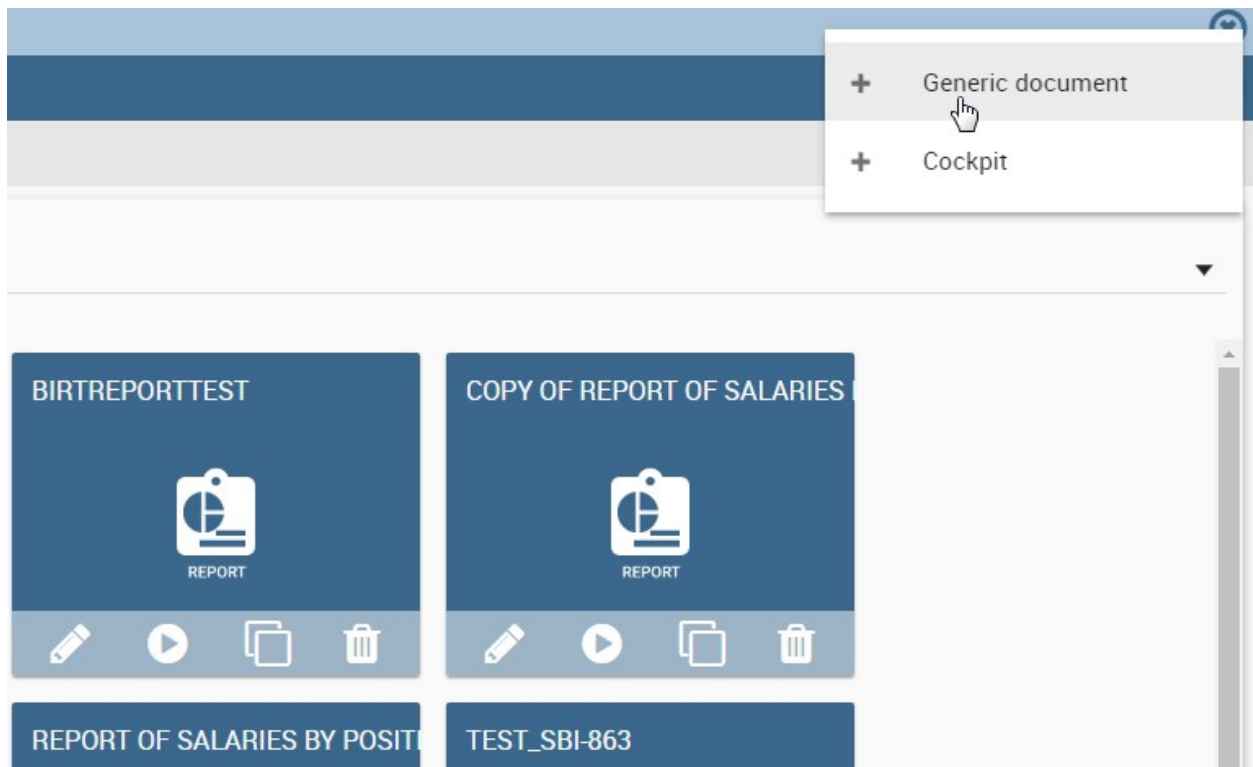


Fig. 5.124: Create a new document.

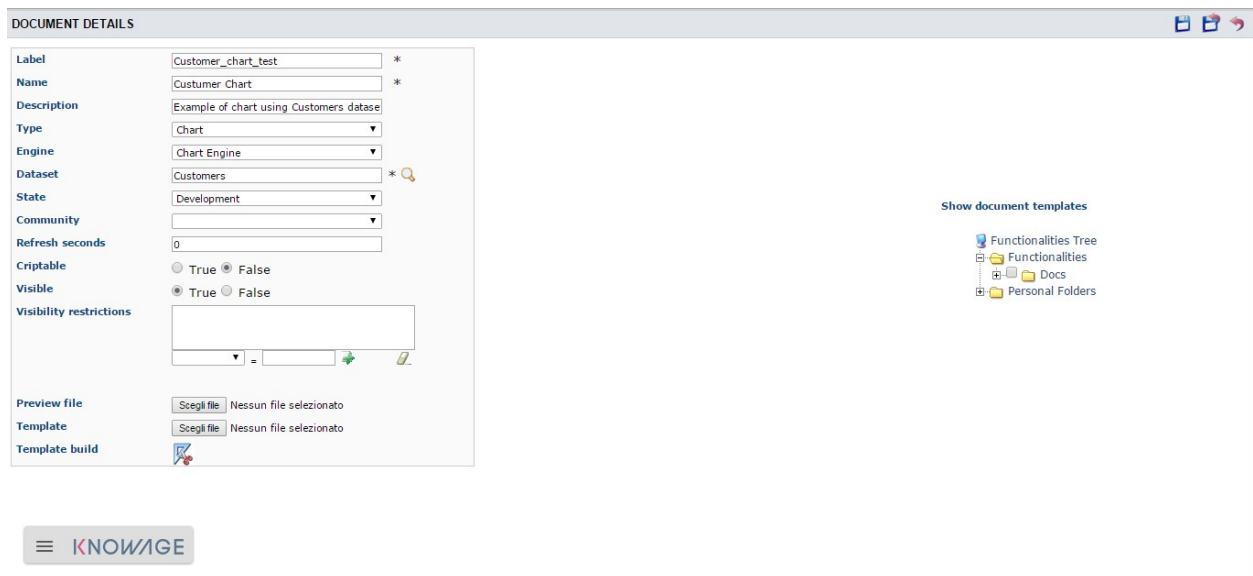


Fig. 5.125: Document Details.

Show document templates

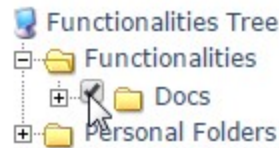


Fig. 5.126: Select the folder in which you want your chart to be saved.

For charts documents outputs parameters are automatically generated during the creation of the document. Therefore you can define cross navigation in the default way, as explained in Cross Navigation.

5.6 Cockpit

Knowage allow end users to *self-build interactive cockpits* through an intuitive and interactive interface, with a few clicks and simple drag and drop. This allows you to compose your analytical documents with multiple widgets and define associations among them, so that clicking on one widget data are automatically updated in other widgets.

It enables *data mash-up* to integrate enterprise data and externally sourced data.

Cockpit documents can be created and executed both by technical users and end users and are part of Knowage ad-hoc reporting system. A key aspect is that different widget can rely on different datasets and hence on different data sources. The only requirement needed to define associations between two or more datasets is the presence in each of them of one or more columns containing the same data.

Warning: Section structure exception


Since there are no differences between the cockpit interface reached by a final user and the one reached by a technical user, the cockpit designer is described in one unique My first Cockpit for both those kind of users. By the way, when necessary we will highlight how the same functionality can be exploited accordingly to the user's role.

5.6.1 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.

Important: Reaching the cockpit designer

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

Label	<input type="text" value="Customer_chart_test"/>	*
Name	<input type="text" value="Customer Chart"/>	*
Description	<input type="text" value="Example of chart using Customers datase"/>	
Type	<input type="text" value="Chart"/>	▼
Engine	<input type="text" value="Chart Engine"/>	▼
Dataset	<input type="text" value="Customers"/>	* 🔍
State	<input type="text" value="Development"/>	▼
Community	<input type="text"/>	▼
Refresh seconds	<input type="text" value="0"/>	
Criptable	<input type="radio"/> True <input checked="" type="radio"/> False	
Visible	<input checked="" type="radio"/> True <input type="radio"/> False	
Visibility restrictions	<div><input type="text"/> ▼ = <input type="text"/> ➕ 📱</div>	
Preview file	<input type="button" value="Scegli file"/> Nessun file selezionato	
Template	<input type="button" value="Scegli file"/> Nessun file selezionato	
Template build	<div> <input type="button" value="Generate a new template"/></div>	


 **KNOWAGE**

Fig. 5.127: Template build.

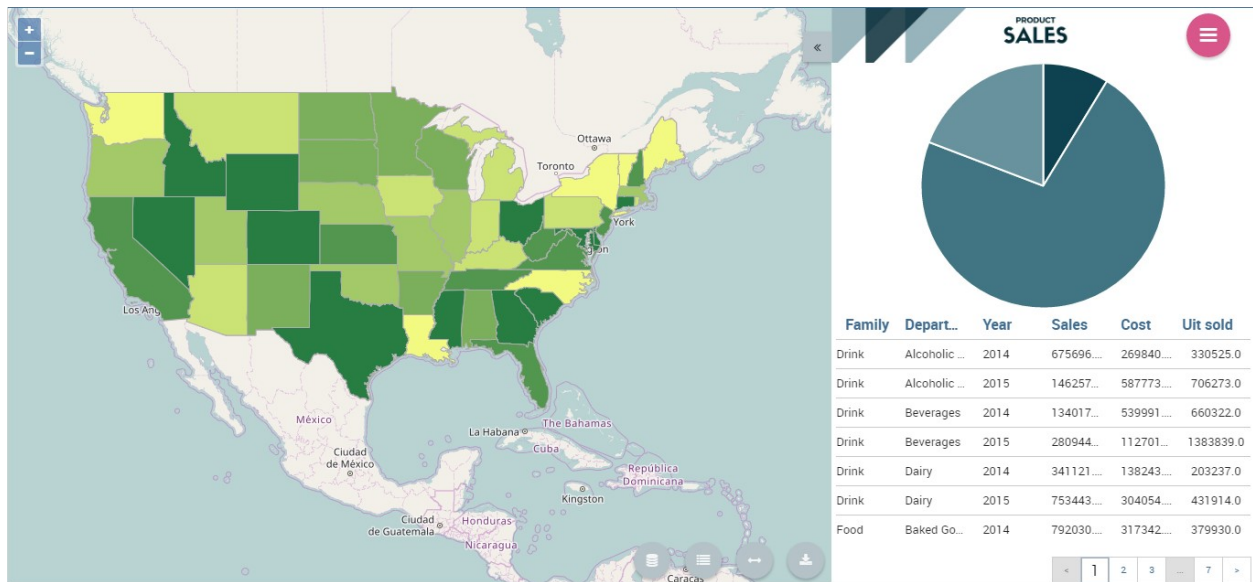









Fig. 5.128: Cockpit document example.

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 below.

Table 5.8: Cockpit editor toolbar.

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.

By clicking the button **Add Widget** you can add a widget containing a **Text**, an **Image**, a **Chart**, a **Table**, a **Cross table**, a **Document**, the **Active selections** or the **Selector** to your cockpit, as shown below.

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

5.6.1.1 Text widget

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

On the “Text editor” tab you can type the desired text in center panel and customize it. Using the dataset tab it is possible to associate dataset values to the text and read it real time at each execution. Move to the dataset tab to add a dataset to the widget. Then, going back to the Text editor tab, the user will find the dataset columns on the right side, as well as a set of functions to eventually apply to the fields. We summed up main steps in the Figure below. To add a function to a measure first select the desired function and then the field of numeric type.

It’s also possible to add variables values if at least one of them is defined. During the execution of the widget the value will be displayed based on the current variable value.

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. Finally, the “Filters” tab can be used to extract limited output from the dataset. We put details off to the table widget subsection.

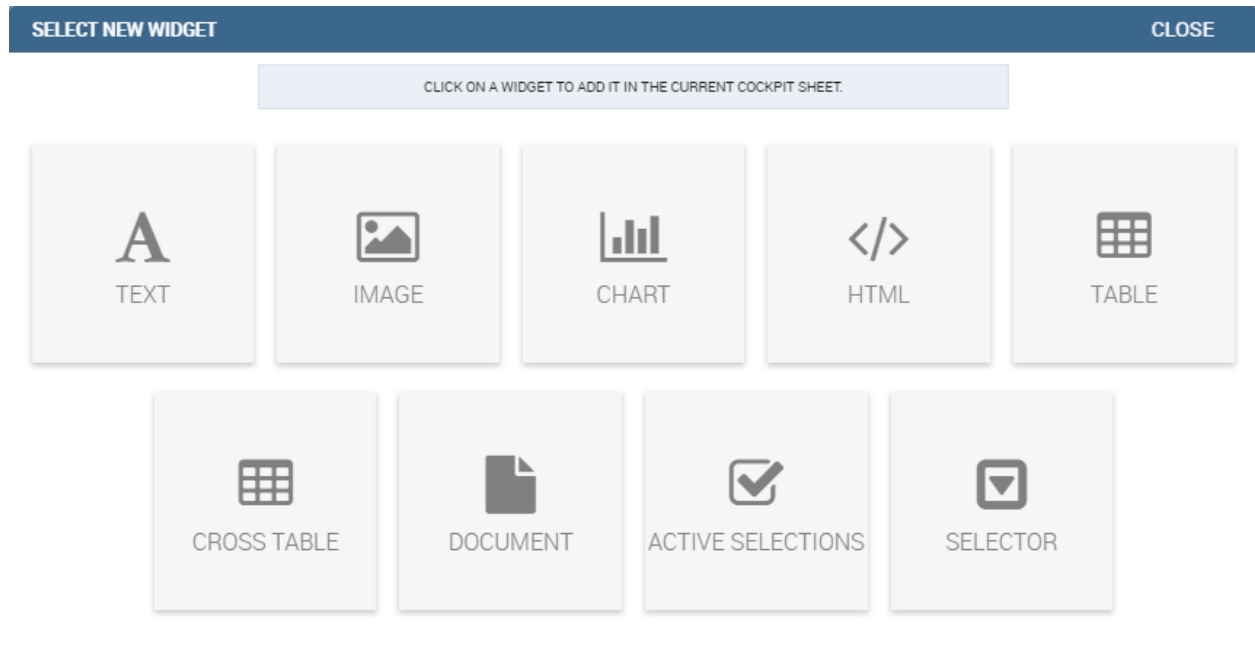


Fig. 5.129: Widget Type.

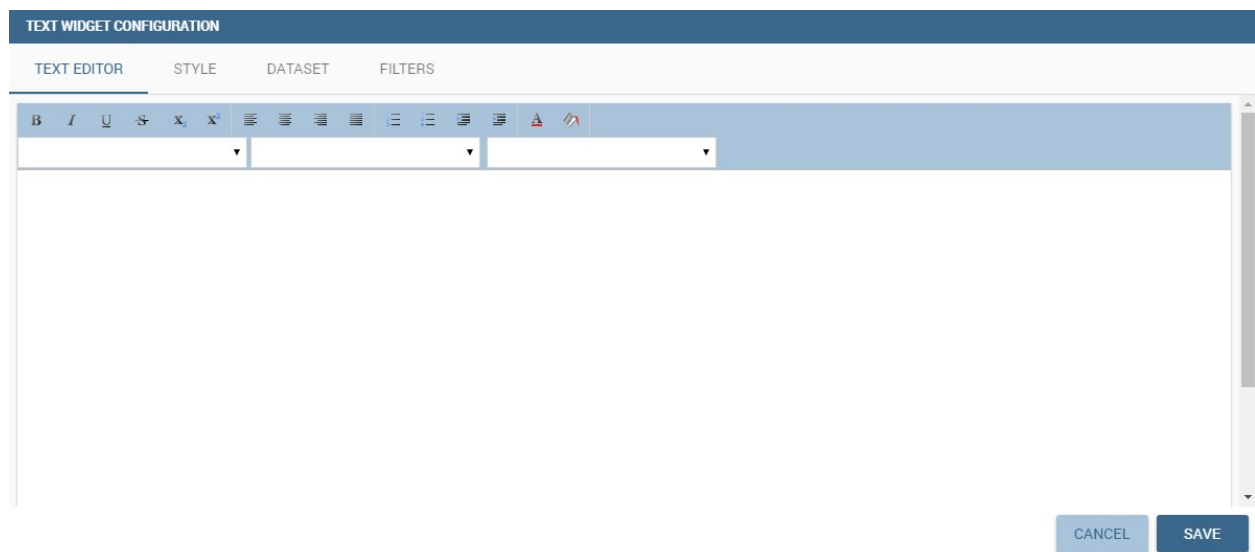


Fig. 5.130: Text editor of text widget configuration.

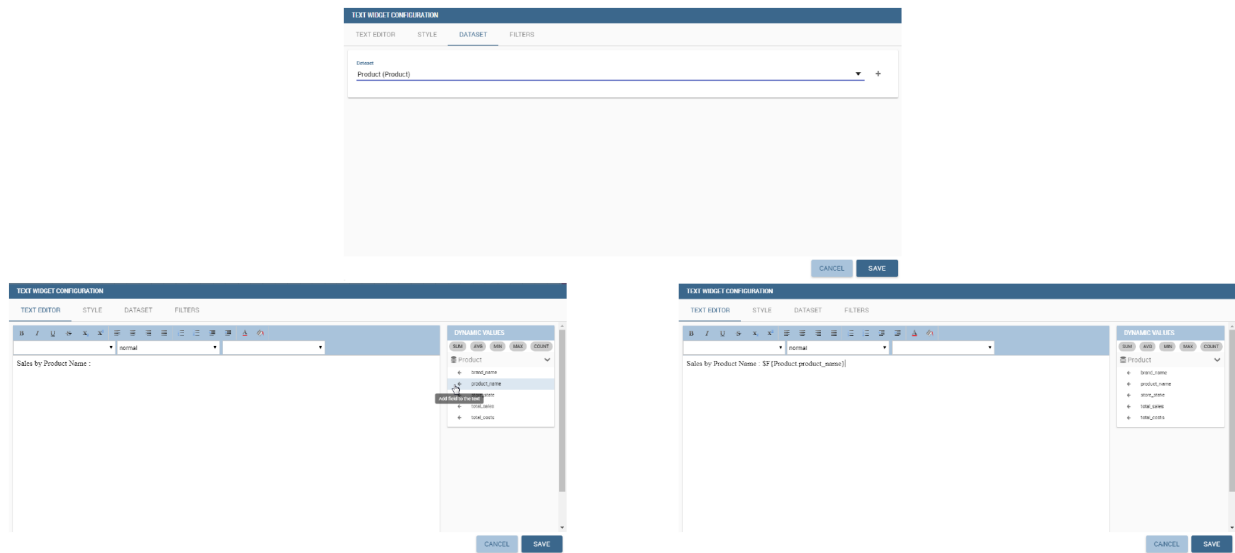


Fig. 5.131: Editing a dynamic text.

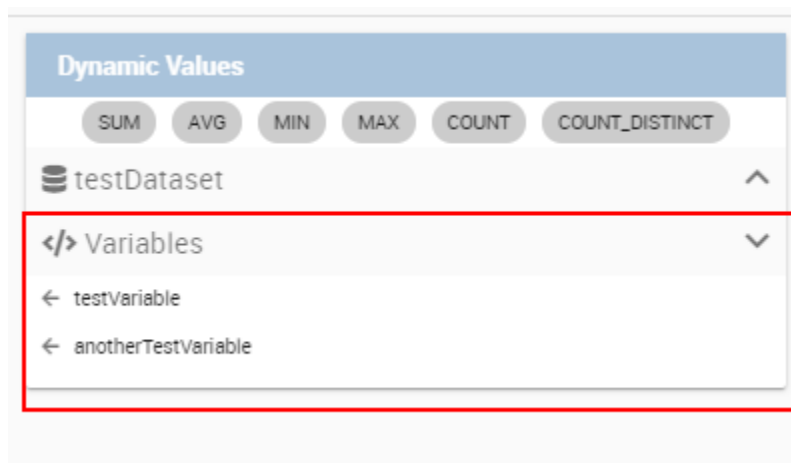


Fig. 5.132: Variables in text widget.

5.6.1.2 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 following figure.



Fig. 5.133: 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 below.

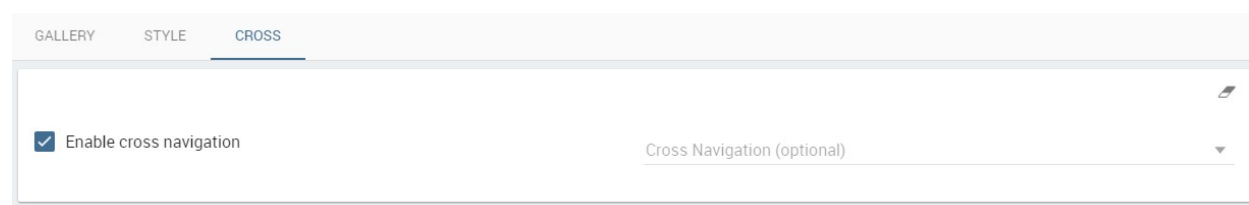


Fig. 5.134: Cross tab of Image Widget Configuration.

Warning: 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.

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 image widget cross navigation will be shown a pop up (refer to figure below) with the list of cross navigation definition that exist for this cockpit.

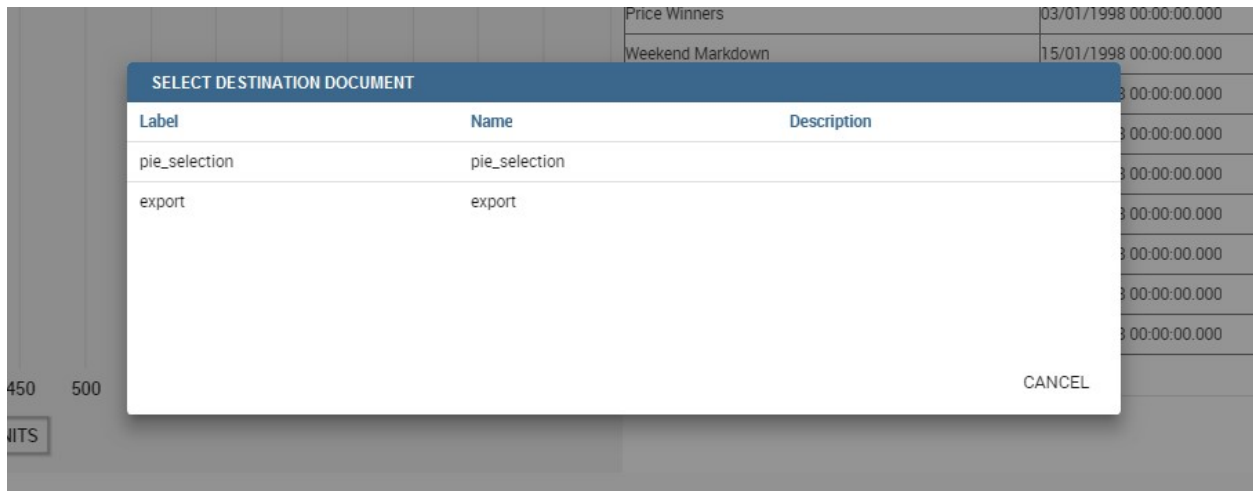


Fig. 5.135: Cross navigation multiple choices.

5.6.1.3 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 Chart chapter.

We recall that also for chart widget it is possible to set cross navigation on elements.

Warning: Cross navigation only for technical users

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

As shown in next figure, it is mandatory to enable the cross navigation feature by using the dedicate tab of chart editor GUI. It is mandatory to choose the column element to be passed to the destination document and associate it to the right output parameter (previously added to the document using the detail interface).

The cross navigation name can be left empty. In case multiple cross navigation definitions have been configured for the document, a pop up will be displayed, letting the user to choose which destination to reach (exactly as we saw earlier for Image widget in the last figure of that paragraph).

In addition, if the navigation expects other parameters to be passed, use the bottom part of the page to add the additional parameters. Figurebelow shows an example.

It is also possible using Solr datasets and calculated fields with charts.

Dataset fields editing and configuration is possible using the first tab inside widget edit mode, in this way, for example, users can modify aggregations for measures:

In this way it is possible adding new calculated fields on chart by clicking on “Add Calculated field”, the standard calculated field editing mode will appear.

Adding new calculated fields is easy as using other measure fields, using Chart Engine Designer structure tab:

5.6.1.4 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 below. In details, it is mandatory to select a dataset using the

CHART WIDGET CONFIGURATION

DATASET CHART ENGINE DESIGNER STYLE CROSS FILTERS

☒ Enable cross navigation

Column CATEGORY_VALUE

Output parameter out_par

Cross Navigation (optional)

Output parameters list

CANCEL SAVE

Fig. 5.136: Cross navigation for chart widget.

CHART WIDGET CONFIGURATION

DATASET CHART ENGINE DESIGNER STYLE CROSS FILTERS

☒ Enable cross navigation

Column CATEGORY_VALUE

Output parameter out_par

Cross Navigation (optional)

Output parameters list

☒ out_par2

Type Dynamic

Column SERIE_VALUE

CANCEL SAVE

Fig. 5.137: Add all output parameters involved in the cross navigation.

Chart Widget Configuration				
DATASET	CHART ENGINE DESIGNER	STYLE	CROSS	FILTERS
TABLE COLUMNS			ADD CALCULATED FIELD	
fname	fname	ATTRIBUTE	string	
mi	mi	ATTRIBUTE	string	
address1	address1	ATTRIBUTE	string	
address2	address2	ATTRIBUTE	string	
address3	address3	ATTRIBUTE	string	
address4	address4	ATTRIBUTE	string	
city	city	ATTRIBUTE	string	
state_province	state_province	ATTRIBUTE	string	
postal_code	postal_code	ATTRIBUTE	string	
country	country	ATTRIBUTE	string	
customer_region_id	customer_region_id	ATTRIBUTE	integer	
phone1	phone1	ATTRIBUTE	string	
phone2	phone2	ATTRIBUTE	string	
birthdate	birthdate	ATTRIBUTE	date	
marital_status	marital_status	ATTRIBUTE	string	
yearly_income	yearly_income	ATTRIBUTE	string	
gender	gender	ATTRIBUTE	string	
total_children	total_children	MEASURE	integer	<input checked="" type="checkbox"/> SUM
num_children_at_home	num_children_at_home	MEASURE	integer	<input checked="" type="checkbox"/> SUM
education	education	ATTRIBUTE	string	
date_acct_opened	date_acct_opened	ATTRIBUTE	date	
member_card	member_card	ATTRIBUTE	string	
			<div>CANCEL</div> <div>SAVE</div>	

Chart Widget Configuration				
DATASET	CHART ENGINE DESIGNER	STYLE	CROSS	FILTERS
TABLE COLUMNS			ADD CALCULATED FIELD	
fname	fname	ATTRIBUTE	string	
mi	mi	ATTRIBUTE	string	
address1	address1	ATTRIBUTE	string	
address2	address2	ATTRIBUTE	string	
address3	address3	ATTRIBUTE	string	
address4	address4	ATTRIBUTE	string	
city	city	ATTRIBUTE	string	
state_province	state_province	ATTRIBUTE	string	
postal_code	postal_code	ATTRIBUTE	string	
country	country	ATTRIBUTE	string	
customer_region_id	customer_region_id	ATTRIBUTE	integer	
phone1	phone1	ATTRIBUTE	string	
phone2	phone2	ATTRIBUTE	string	
birthdate	birthdate	ATTRIBUTE	date	
marital_status	marital_status	ATTRIBUTE	string	
yearly_income	yearly_income	ATTRIBUTE	string	
gender	gender	ATTRIBUTE	string	
total_children	total_children	MEASURE	integer	<input checked="" type="checkbox"/> SUM
num_children_at_home	num_children_at_home	MEASURE	integer	<input checked="" type="checkbox"/> SUM
education	education	ATTRIBUTE	string	
date_acct_opened	date_acct_opened	ATTRIBUTE	date	
member_card	member_card	ATTRIBUTE	string	

Calculated Field

total_children

total_children (SUM)

num_children_at_home

num_children_at_home (SUM)

num_cars_owned

num_cars_owned (SUM)

Alias *

aggregation

column totals

Validate

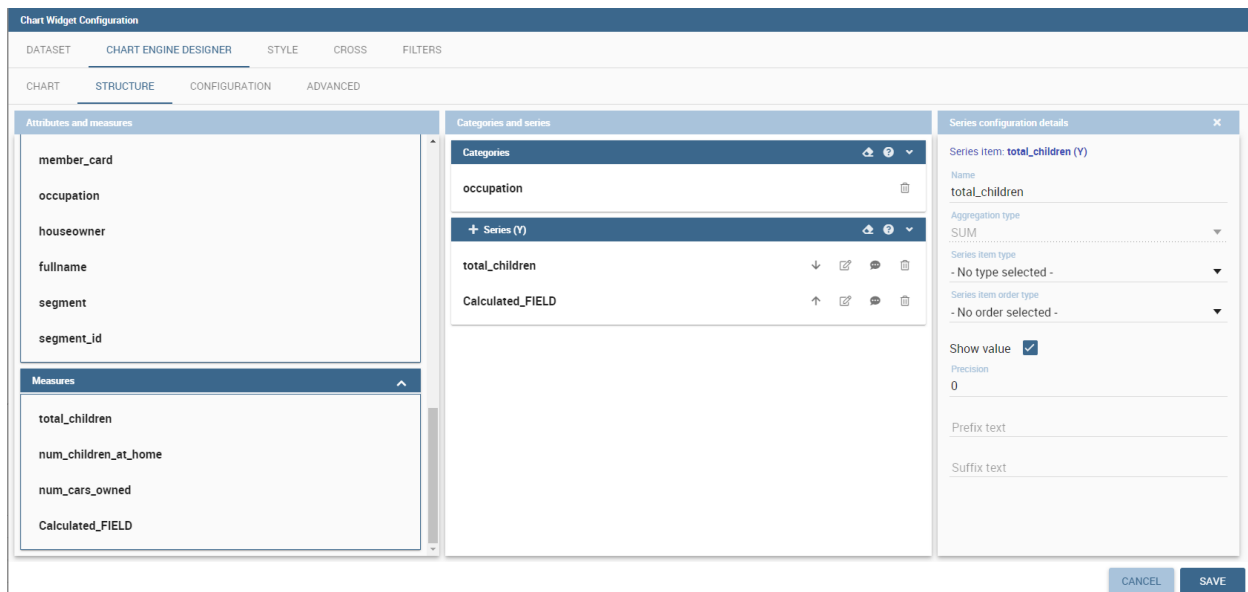
Apply on dataset

CANCEL

SAVE

CANCEL

SAVE



combobox (only if at least one dataset has 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.

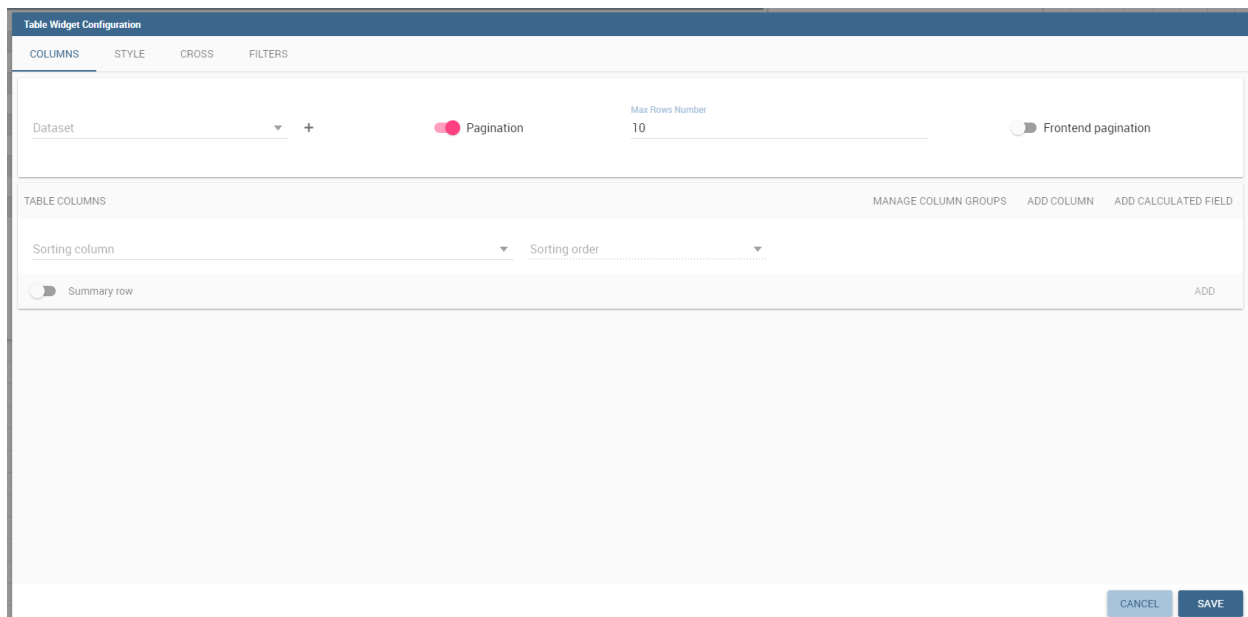


Fig. 5.138: Table configuration window.

In fact, the column area is divided into two parts: on the upper part there are the buttons to add a new column or a calculated field. Also the new functionality of column grouping is available here. In the lower section, as soon as the 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 dedicated comboboxes.

When a dataset is added to a table widget, all of its columns are listed below. If the user doesn't wish to show some of them, he can use the delete button available at the end of each column row, as shown below.

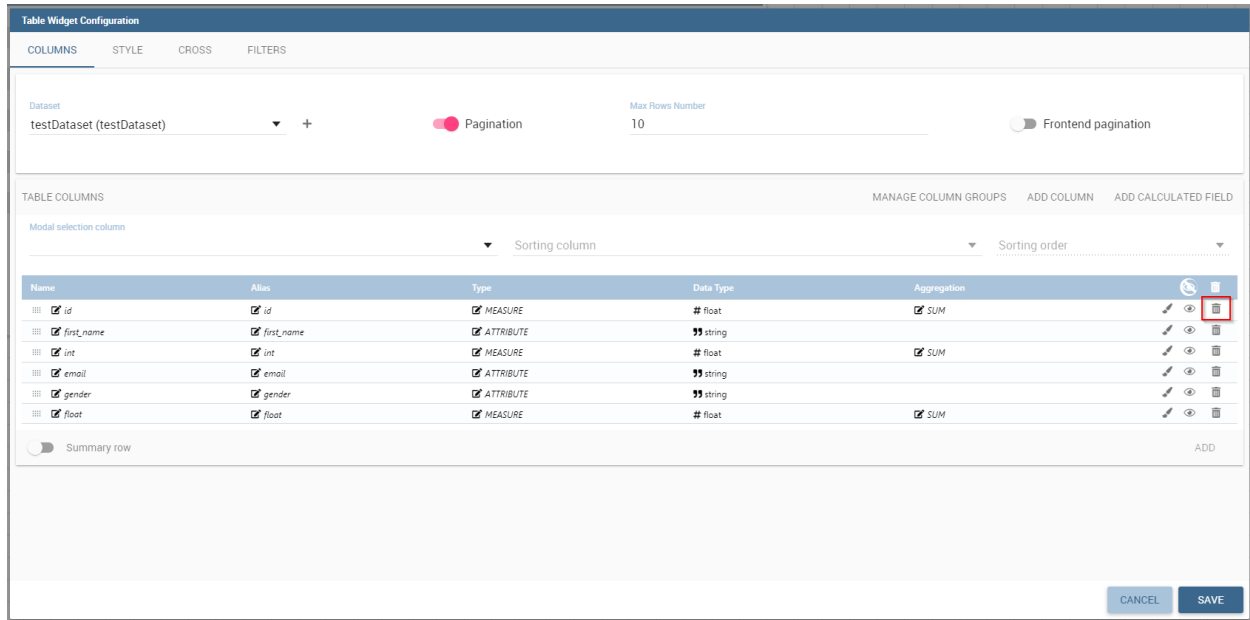


Fig. 5.139: Delete a column.

In case of accidental cancellation or new table requirements, it is possible to re-add 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 below.

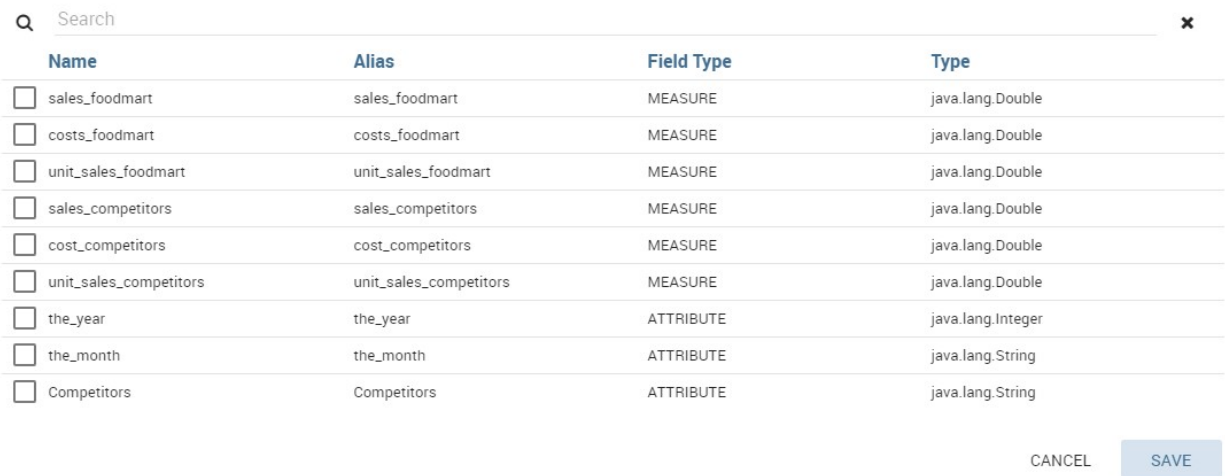


Fig. 5.140: Add a new column.

Manage Columns Groups will open a menu to add or remove columns groups and to set their style. A column group is a container for more than one column that will show a common header between them.

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 your calculated field in the dedicated area at

testGroup					
id	first_name	int	email	gender	float
766.00	Abigail	7,721.00	awyk9@pen.io	Female	41.35
626.00	Adah	6,358.00	apierreponthd@sun.com	Female	91.35
270.00	Adams	7,206.00	afincham7h@nature.com	Male	4.48
68.00	Adara	1,030.00	agibling1v@cnn.com	Female	61.65
408.00	Addy	8,793.00	asartonbb@alibaba.com	Male	84.19
868.00	Adeline	4,657.00	astacko3@latimes.com	Female	10.28
728.00	Adiana	7,889.00	abalchenk7@ed.gov	Female	17.40
7.00	Adora	609.00	ablabe6@ft.com	Female	26.00
958.00	Adrienne	8,805.00	adrezzerql@mtv.com	Female	35.54
669.00	Adrienne	2,544.00	aleadleyik@whitehouse.gov	Female	50.82

Fig. 5.141: Example column group.

the top corner of the wizard. Then you can choose from the left sidebar list the fields that you want to use in your formula. You can also use arithmetical functions or use the functions available in the menu (**aggregations**, **column totals**, **variables**). If you prefer you can create or modify the expression manually directly in the editable panel. When you are satisfied with your expression you can click on validate to check your formula syntax or save button and your calculated field appears in the field list. We provide an example in the following figure.

Fig. 5.142: Add a calculated field.

You can choose between dataset-level behaviour or table-level (default), that is the calculation modality that will involve table fields or dataset columns to get the result. If you choose dataset-level behaviour the calculation result will be done at dataset columns level only and the aggregation will involve the entire result of the calculation.

If Dataset is of type Solr, the columns displayed on the right panel are dataset columns fields and the calculated field formula elaboration is calculated on the fly.

If variables are set for the present cockpit, the variable menu button will appear, making it possible to add variable values in the calculated field expression.

Variables menu

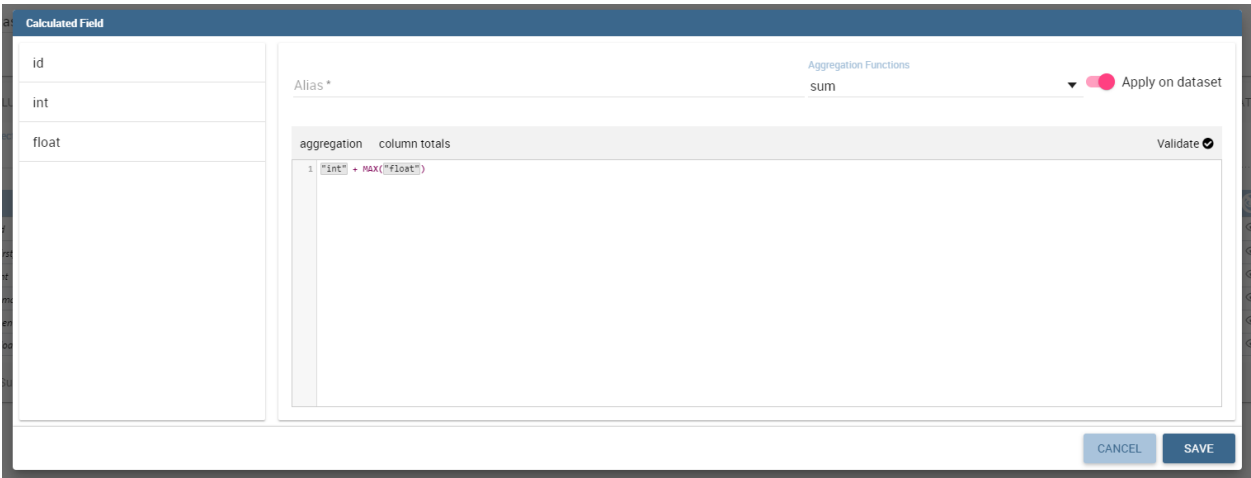
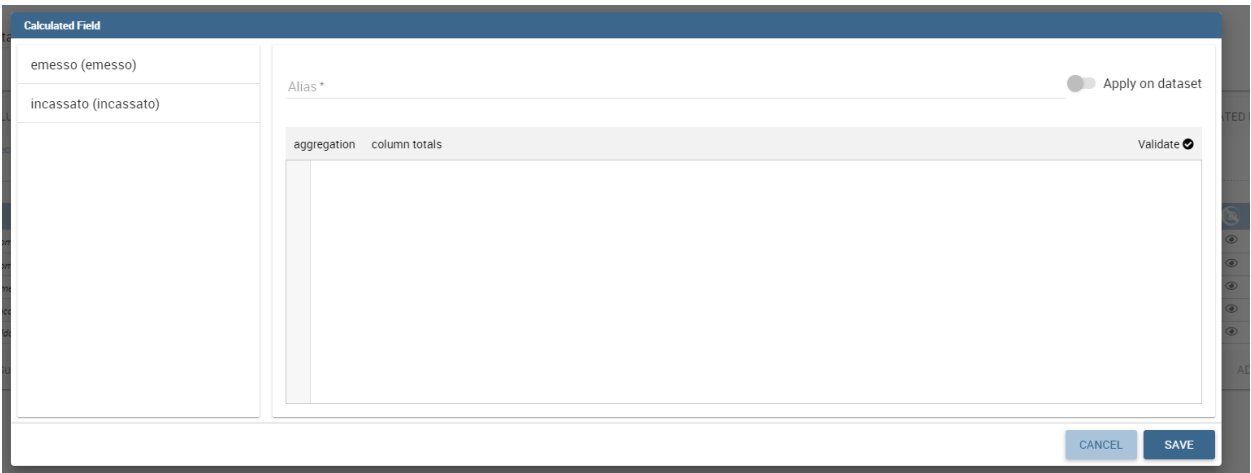
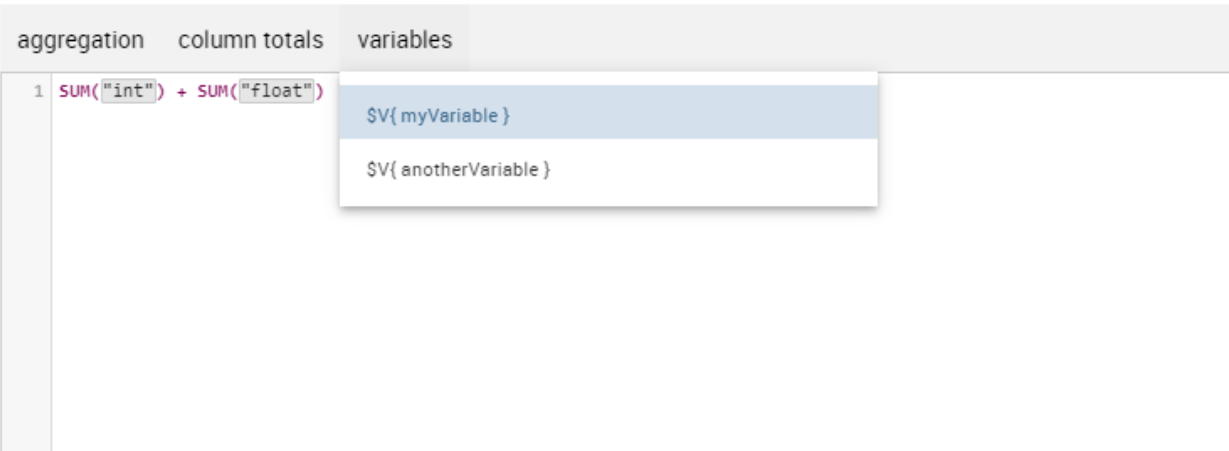


Fig. 5.143: Dataset-level behaviour



Alias *



In the bottom section of the window, you can see the table fields (with their aggregation type) listed and you also can sort columns displayed in the table by dragging them up or down, insert a column alias and customize it by adding font and style configurations using the brush shaped icon, as you can see from figure below. Here you can find configuration features to adjust the column size, max cell characters, hide column or column header options, and the row spanning toggler.

Columns Settings

If you hide the column (from this view or from the column list), the column will not be visible, but will still be used for aggregation purposes. If you enable the **row span feature**, all the same visible values of the column will be collapsed in one, to avoid repetitions.

If the column is a measure, more functionalities will become available:

- **Inline chart mode:** you can choose the visualization type of the measure, and if you choose chart and maximum/minimum values, a chart will appear in the view to represent the cell measure.
- **Thresholds:** you can choose to set some thresholds that will trigger font color, background color or will show icons if the chosen condition is satisfied.
- **Format:** you can choose the prefix, suffix and the precision (i.e. 9.8 m/s). Please be aware that the data will be formatted following the locale of the user. Otherwise you can choose to treat it as string.

For all the columns, if at least one variable is set, the variables condition box will appear. Depending on the variable usage it will be possible to set a dynamic header or to hide the column conditionally.

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*.

If a column group has been set another option will become available in order to set the optional group belonging of the column.

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

- In the **Summary** section you can show the total of the column and customize it by typing the summary name and using font and style configurations. Refer to Figure below.

TABLE COLUMNS

MANAGE COLUMN GROUPSADD COLUMNADD CALCULATED FIELD

Modal selection column

Sorting column

Sorting order

Group	Name	Alias	Type	Data Type	Aggregation	
<input checked="" type="checkbox"/> testGroup	<input checked="" type="checkbox"/> id	<input checked="" type="checkbox"/> id	<input checked="" type="checkbox"/> MEASURE	# float	<input checked="" type="checkbox"/> SUM	
<input checked="" type="checkbox"/> testGroup	<input checked="" type="checkbox"/> first_name	<input checked="" type="checkbox"/> first_name	<input checked="" type="checkbox"/> ATTRIBUTE	string		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> int	<input checked="" type="checkbox"/> int	<input checked="" type="checkbox"/> MEASURE	# float	<input checked="" type="checkbox"/> SUM	
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> email	<input checked="" type="checkbox"/> email	<input checked="" type="checkbox"/> ATTRIBUTE	string		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> gender	<input checked="" type="checkbox"/> gender	<input checked="" type="checkbox"/> ATTRIBUTE	string		
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> float	<input checked="" type="checkbox"/> float	<input checked="" type="checkbox"/> MEASURE	# float	<input checked="" type="checkbox"/> SUM	

☐ Summary row

ADD

Fig. 5.144: Column settings.

TABLE WIDGET CONFIGURATION

COLUMNSSTYLECROSSFILTERS

☐ SUMMARY

Summary Text

Font-family

Font Style

Font Size

Font-weight

Font-color

Background-color

Fig. 5.145: 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 below.

Rows

☐ Multiselectable

☐ Auto-adapt Rows height

Rows Height

☐ Enable rows summary

Rows summary text

Fig. 5.146: 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.
- In the **Header Style** section you find the different options of style for the table header. Refer to Figure below.
- 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 below.
- In the **Borders** section you can add a border to the widget and customize it by using the colors, thickness and style. Refer to the following figure.
- In the **Other Options** section you can add the shadows in the widget, you can set the background color of the widget and it is possible to disable or enable the screenshot option for that particular widget. Refer to the following figure.

244

Chapter 5. Functionalities

Grid

☐ Show Grids

Grids-width

Grids-color

☐ Alternate rows

Even-rows color

Odd-rows color

Fig. 5.147: Grid section of the Style tab.

HEADER STYLE

Font-family

Font Size

Font-weight

Font-color

Background-color

Cell horizontal alignment

Fig. 5.148: Header style section of the Style tab.

TITLES

Title text

Horizontal alignment

Font Family

Font Size

Font Weight

Header Height (px)

Title Color

Title Background Color

Fig. 5.149: Titles section of the Style tab.

BORDERS

Borders Style

Borders Thickness

Borders Color

Border radius top left

Border radius top right

Border radius bottom left

Border radius bottom right

Fig. 5.150: Borders section of the Style tab.

OTHER OPTIONS

☐ Shadows

Shadows Size

Background color

☒ Enable Screenshots

Fig. 5.151: Other Options 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).

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. the following figure shows an example of how to set the limit rows or a conditions on dataset columns.

TABLE WIDGET CONFIGURATION

COLUMNS STYLE CROSS **FILTERS**

Max Rows Number

☒ Limit rows 200

SELECT THE FILTER OPERATOR AND THE VALUE FOR THE COLUMN. REMEMBER TO USE THE CORRECT DATABASE SINTAX FOR THE VALUE. IE: %LIKE%

Add new filter: +

Dataset	Column	Column	Value	Value
TEST_01	THE_DATE	=	1998-01-05 00:00:00	

CANCEL SAVE

Fig. 5.152: Filters tab of the table widget configuration.

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

5.6.1.5 Cross Table widget

Similar configurations are available also for the Cross Table widget. In this data visualization option, you still have the tabs: **Dataset** tab, **Configuration** tab, the **Style** tab and the **Filters** tab as you can see below.

CrossTab Widget Configuration

DATASET CONFIGURATION STYLE CROSS FILTERS

Dataset

CANCEL SAVE

Fig. 5.153: 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 below. Remember to set column and row fields as attributes, while measure fields as numbers.

Once the columns, rows and measures have been selected the style of each column can be set by clicking on the cog settings icon. A popup will open with different options for the selected columnn. See figure below.

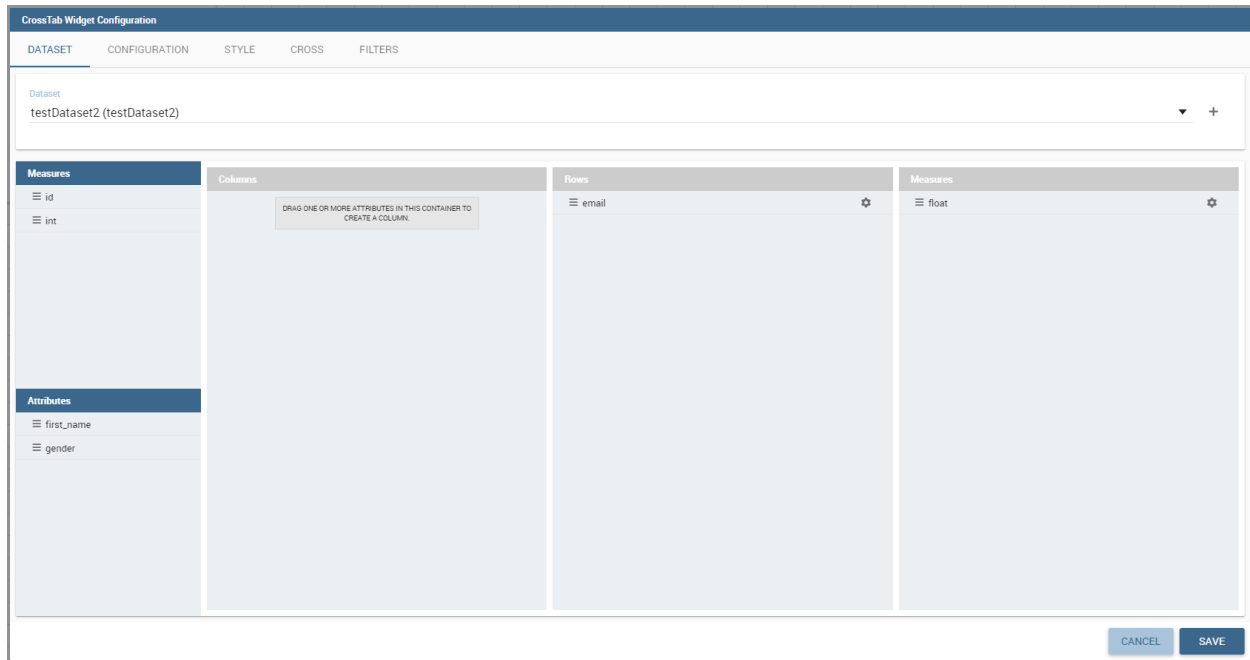


Fig. 5.154: Selecting columns, rows and measures of the crosstab.

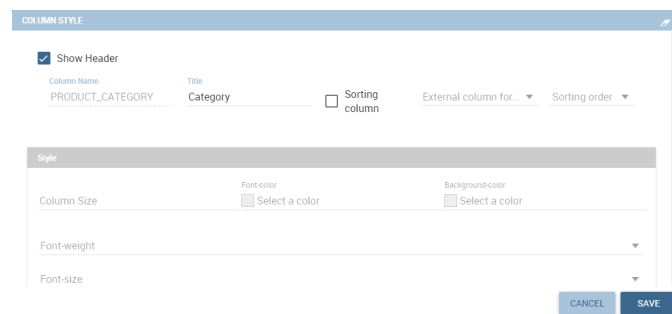


Fig. 5.155: Column style popup.

It is possible to sort the crosstab according to the values of the selected column or, alternatively, according to columns not visible in the crosstab. It can also be set the style of the column, such as the font size, the font weight or the cell alignment. There is also the possibility to specify the size of the column in pixels (you can also use percent values but it is better to use pixels).

In case the selected column is of type measure, there is a dialog to configure the behaviour of that field:

Icon	Condition	Value

Fig. 5.156: Measure configuration dialog.

A particular option for a measure is **Exclude from Total and SubTotal**: that checkbox excludes the measure from the sums of Total and SubTotal fields making the relative table cells empty.

Icon	Condition	Value
🔴	<	10
🟠	none	
🟡	none	
🟢	none	

Background thresholds			
Condition	Threshold *	Condition	Background-color
>	1000	none	rgb(219, 252, 230)
<	50	none	rgb(255, 228, 228)
Condition			

Fig. 5.157: Measure column style.

As figure above shows, you can also manage threshold. It is possible to associate a particular icon or a specific background color to a particular measure's value or range.

Once the dataset has been properly configured, you can proceed to the "Configuration" tab.

The latter is made up of three sections: **General**, **On rows** and **On columns**, as Figure below shows.

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 below exhibit an example.

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

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

CrossTab Widget Configuration

DATASET CONFIGURATION STYLE CROSS FILTERS

General

Max Cells Number

Measures on ☒ Columns ☐ Rows

Percentage calculated on ☐ Columns ☐ Rows ☒ no

On rows

☒ show totals Labels for Totals Total ☒ show sub-totals Labels for Subtotals SubTotal

On columns

☐ show totals Labels for Totals Total ☐ show sub-totals Labels for Subtotals SubTotal

CANCEL SAVE

Fig. 5.158: Configuration tab interface.

↑ 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	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	1,276,429.00
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

Fig. 5.159: 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 ↑ the_month														
↑ gender ↑ product family		January	February	March	April	May	June	July	August	September	October	November	December	
	Sales	Sales	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	

Fig. 5.160: Computing totals and/or subtotals on columns.

- **Crosstab General Options** where the rows' height, the general font and font size can be set; in particular, the layout combo determines how the columns resize themselves in respect of the contained value;

Fig. 5.161: General style options for crosstab.

- **Crosstab Headers Font Options** where you can configure the header style settings as color, background, font, etc.

Fig. 5.162: Crosstab Headers Font Options for crosstab.

- **Measures Font Options** where you can configure several style options for measures, such as color, background, font size, etc.

Fig. 5.163: 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.
- In the **Measures Headers** section you can configure different style option for measure headers, such as color, background, font size, etc.
- In the **Total** section you can set color and background of totals (if any).
- In the **Subtotal** section you can set color and background of subtotals (if any).
- In the **Titles** section you can add titles to widget and customize them using different styles.
- In the **Borders** section you can add borders to widgets and customize them using different styles.
- In the **Other Options** section you can add a shadow to widget layout and indicate its measure, color the widget background at convenience and it is possible to disable or enable the screenshot option for that particular widget.

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.

In addition to other cross-navigations, for cross table widget it is possible to set as a dynamic value the name of the selected measure column or the selected category. The choice is available from the combobox.

GRID			
<input type="checkbox"/> Show Grids	Borders Style	Borders Thickness	Grids-color
<input type="checkbox"/> Alternate rows	Even-rows color	Odd-rows color	
	Select a color	Select a color	

Fig. 5.164: Grid Options for crosstab.

MEASURES HEADERS		
Font Size <small>px, rem or % measure units are available</small>	Font Family	Font Weight
Text Decoration	Color	Background
	Select a color	Select a color

Fig. 5.165: Measures Headers Option for crosstab.

TOTALS	
Color	Background
Select a color	Select a color

Fig. 5.166: Color settings for Totals.

SUBTOTALS	
Color	Background
Select a color	Select a color

Fig. 5.167: Color settings for Subtotals.

TITLES		
Title text	Horizontal alignment	Font Family
Font Size <small>px, rem or % measure units are available</small>	Font Style	Font Weight
Header Height <small>Height in px of the space available to title</small>	Title Color	Title Background Color
	Select a color	Select a color

Fig. 5.168: Title settings.



Fig. 5.169: Border settings.

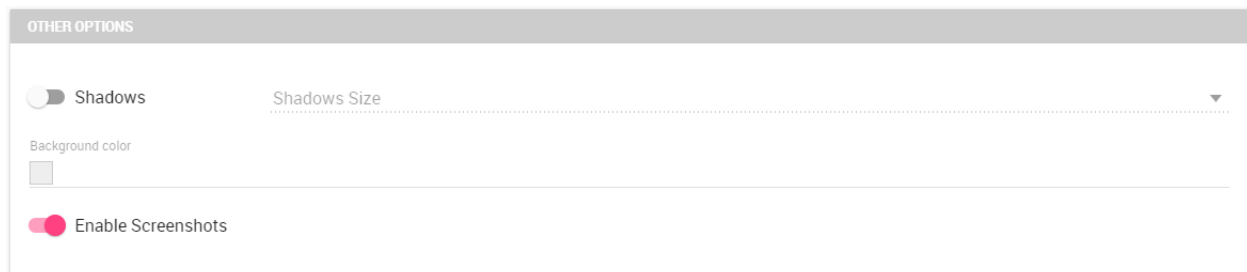


Fig. 5.170: Other Options for crosstab.

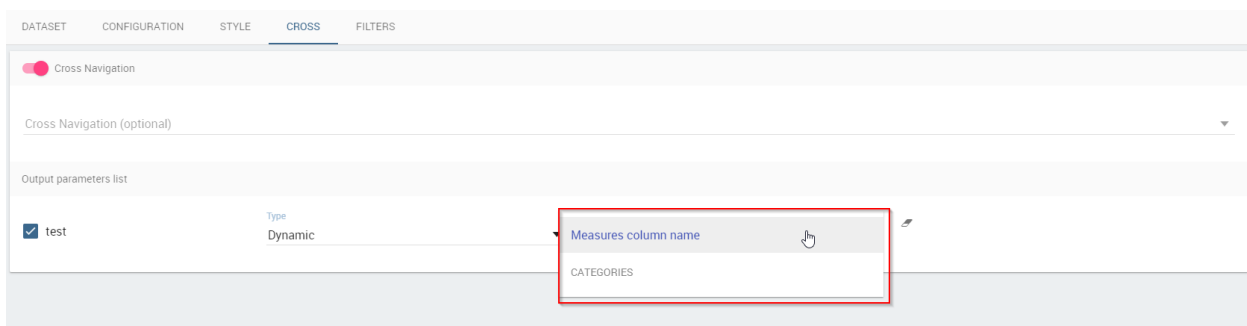


Fig. 5.171: Cross navigation for cross table widget.

5.6.1.6 Document section

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

Use the Data configuration button to add a document source to the cockpit. Click on the “Plus” icon on the right half of the page to choose among all available documents.

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

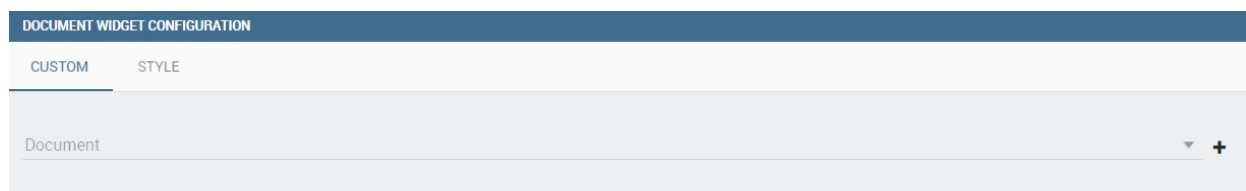


Fig. 5.172: Custom tab of the Document widget.

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

5.6.1.7 Selection widget

This widget is related to the association concept so in this subsection we give information on how to add and custom the **Selection Widget** into the cockpit area and its functioning, while we refer to the dedicated Document section for details on how to set (global) associations.

To enable the Selection widget, which means the possibility to have all associations listed and accessible on a widget, the user must open the “Selection” feature through the “Add widget” functionality and configure the demanded options. Figure below shows the “Selection widget configuration” interface.

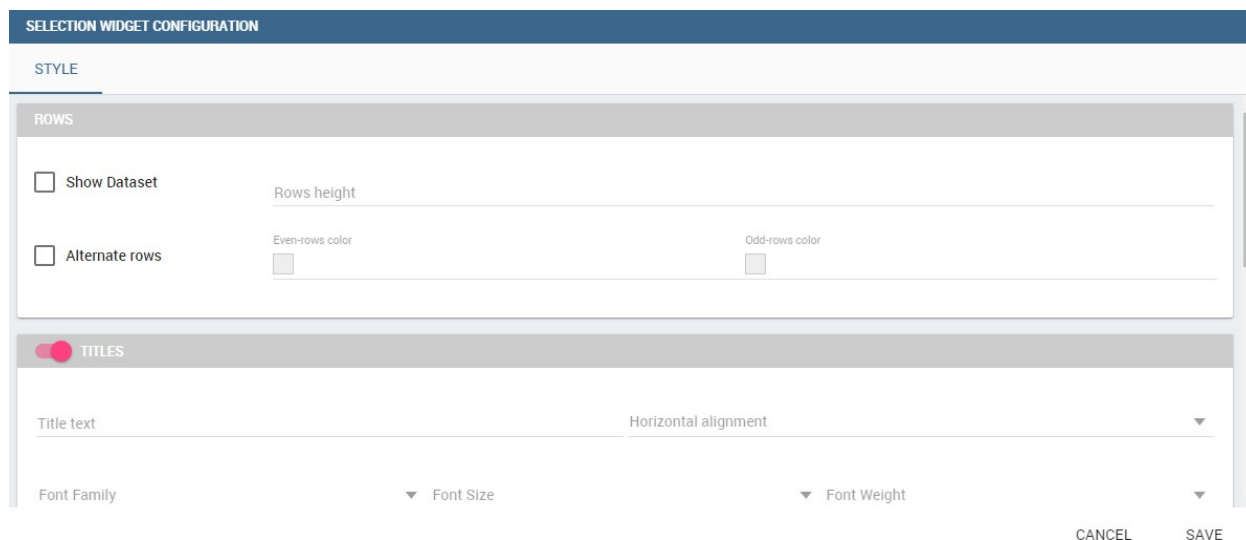


Fig. 5.173: Selection widget configuration.

The Selection Widget will display the elements selected by the user. Figure below shows an example.

qua... the_date	sto... product_family	unit_sales	store_cost
Q1 1998-01-29 00:00:00.0	13 Drink	46	44.8398
Q2 1998-04-30 00:00:00.0	13 Non,Consumable	88	66.1926
Q2 1998-05-07 00:00:00.0	13 Drink	55	37.5161
Q4 1998-11-16 00:00:00.0	18 Non,Consumable	21	21.5687
Q4 1998-10-08 00:00:00.0	17 Food	407	348.9055
Q1 1998-01-07 00:00:00.0	19 Food	371	327.3894
Q2 1998-04-08 00:00:00.0	16 Non,Consumable	98	73.0691
Q2 1998-04-14 00:00:00.0	16 Food	311	272.3359
Q2 1998-06-02 00:00:00.0	16 Drink	31	20.612

Fig. 5.174: Selection widget outlook.

If global associations have been set, clicking on table, cross table or chart elements will update all corresponding widgets. Otherwise, only the widget on which selection has been made Selector Widget will be updated. In both cases the Selection widget will display the highlighted attribute values.

5.6.1.8 Selector Widget

The **Selector Widget** is useful when an end user (a user with a USER role type) wants to add a parameter to the document.

Note: Selector widget

A technical user can use the association with an Analytical Driver to filter on cockpit.

Fig. 5.175: Selector widget outlook.

In detail, use the **Columns** tab to select the dataset and the dataset column on which you want to apply the filter. Then custom the **Select modality** options; for instance, choose between single or multivalue or to use a list or a combobox. Note that for the list option you can further choose among “vertical”, “horizontal” or “grid”. You can also decide to add a default value, chosen from main column’s first item, main column’s last item or to simply assign a static value. Finally, by clicking on the Wrap Text option it is possible to wrap the text shown in the selector; this option is useful when the categories to choose from are sting of long dimensions.

In the case of the selector of type list “grid” it is also possible to set the grid columns width.

Fig. 5.176: Grid columns width.

Move to the **Style** tab to set the widget style in terms of: label, titles, borders, shadows and background color. Figure below shows a customization example.

Fig. 5.177: Selector widget configuration.

Finally use the **Filters** tab to handle pagination or filter on a dataset column.

The Selector widget works effectively as a ready-to-use filter panel.

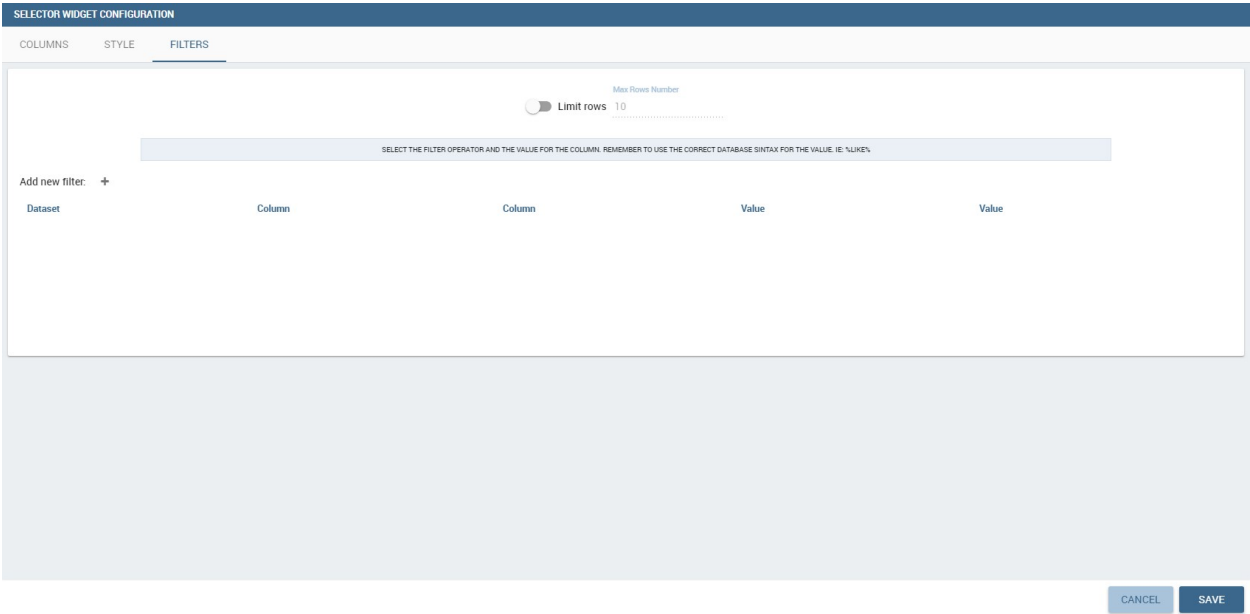


Fig. 5.178: Selector filters.

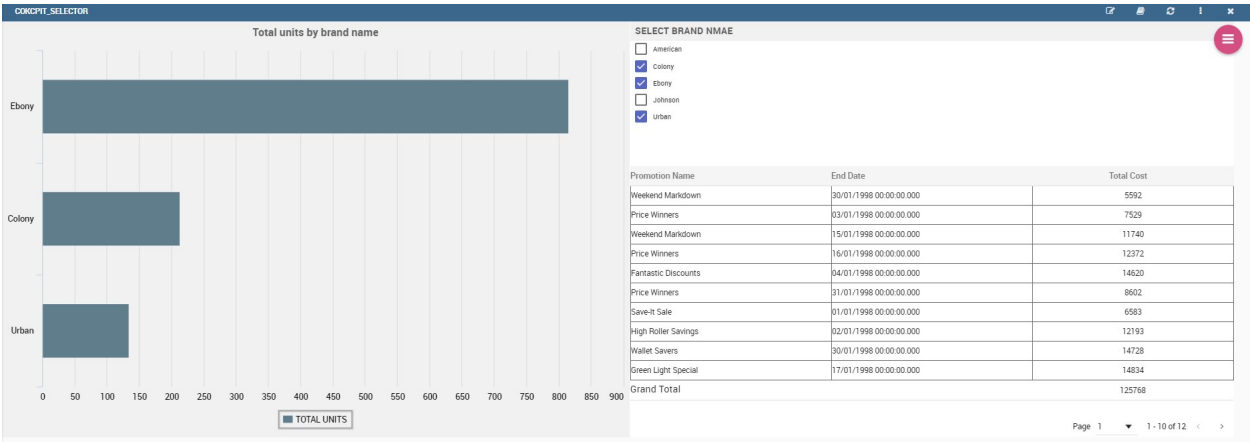


Fig. 5.179: Selector widget execution example.

5.6.1.9 HTML Widget

The HTML widget allows to add customized HTML and CSS code to add very custom dynamic elements to the cockpit. This widget supports all HTML5 standard tags and CSS3 properties.

Warning: For security reasons no custom Javascript code can be added to html tags. Every tag considered dangerous will be deleted on save by the filter.

The Edit section of the widget is composed by five tabs: the dataset, HTML editor, style, cross and filters. In the editor tab is possible to add the code that will be shown in the widget. Clicking on the top expander section in the tab, the one named “CSS” also the CSS editor will be available.

Important: A CSS property will be extended to all the classes in the cockpit with the same name, to apply the property only to the current widget use the id prefix shown in the info panel of the CSS editor

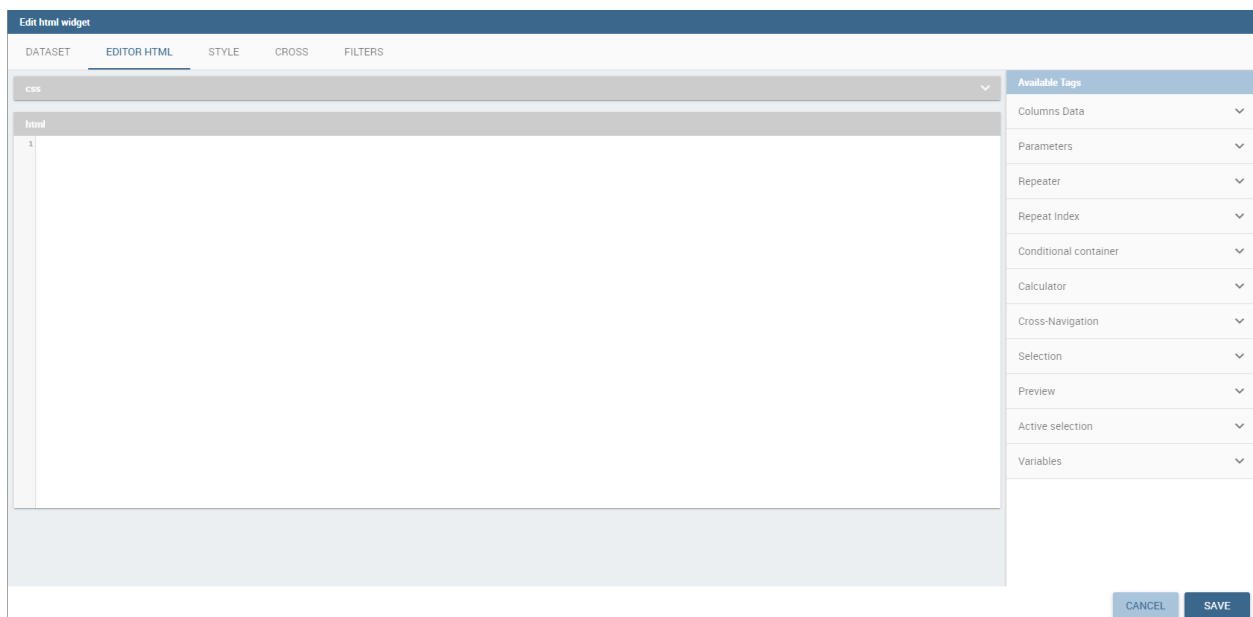




Fig. 5.180: HTML widget editor

In the right side of the editor is possible to take available tags to copy inside the code, those tags will be explained in details in the following paragraphs. Is not possible to add custom JavaScript code inside the html editor, so the available tags are the tools to make the widget dynamic and to use the dataset data.

The Dataset tab allows the user to select a dataset to make the Widget dynamic and to bind it to dataset data. After choosing a dataset the list of available columns will be show. Those names will be useful inside the dynamic tags. Here it is also possible to order the dataset according to a column and to select the ordering type (ascending or descending).

By clicking on the icon  of a specific column the dataset will be ordered by that column by default by ascending order. In order to select the descending ordering type you have to click another time on the icon (the icon will be now like this ).

Available Tags

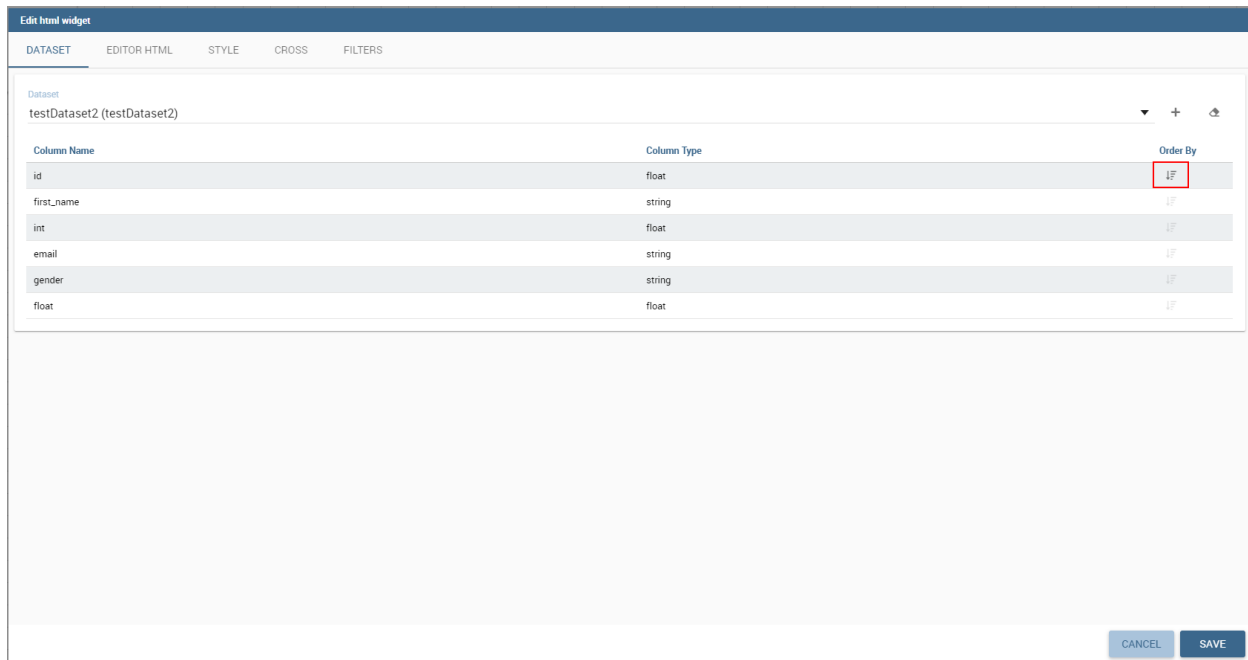


Fig. 5.181: Dataset selection

[kn-column='COLUMN-NAME' row='COLUMN-ROW-NUMBER' aggregation='COLUMN-AGGREGATION' precision='COLUMN-DECIMALS']

The **kn-column** tag is the main dynamic HTML Widget tool, it allows to select a column name from the selected dataset and to print its value. The value of the kn-column attribute should be the name of the column value you want to read in execution.

The **row** attribute is optional and is a number type attribute. If no row is selected the first row column value will be shown.

The **aggregation** attribute is optional and is a string type attribute. If inserted the value shown will be the aggregation of all column rows values. The available aggregations are: AVG|MIN|MAX|SUM|COUNT_DISTINCT|COUNT|DISTINCT COUNT.

The **precision** attribute is optional and is a number type attribute. If added and if the result value is a number, the decimal precision will be forced to the selected one.

[kn-parameter='PARAMETER-NAME']

The **kn-parameter** tag is the tool to show a dataset parameter inside the widget execution. The value of the kn-parameter attribute should be the name of the set attribute.

[kn-calc=(CODE-TO-EVALUATE) precision='VALUE-PRECISION']

The **kn-calc** tag is the tool to calculate expressions between different values on widget execution. Everything inside the brackets will be evaluated after the other tags substitution, so will be possible to use other tags inside.

The **precision** attribute is optional and is a number type attribute. If added and if the result value is a number, the decimal precision will be forced to the selected one.

<div kn-repeat="true" limit="LIMIT-NUMBER"> ... REPEATED-CONTENT ... </div>

The **kn-repeat** attribute is available to every HTML5 tag, and is a tool to repeat the element for every row of the selected dataset.

This attribute is naturally linked to `kn-column` tag. If inside a `kn-column` tag without a `row` attribute is present, the `kn-repeat` will show the column value for every row of the dataset.

Inside a `kn-repeat` is possible to use the specific tag `[kn-repeat-index]`, that will print the index of the repeated column row.

The **limit** attribute is optional and is a number type attribute. If added the number of row repeated will be limited to the selected number.

```
<div kn-if="CODE-TO-EVALUATE"> ... </div>
```

The `kn-if` attribute is available to every HTML5 tag and is a way to conditionally show or hide an element based on some other value. The attribute content will be evaluated after the other tags substitution, so will be possible to use other tags inside. If the evaluation returns true the tag will be shown, otherwise it will be deleted from the execution.

```
<div kn-cross> ... </div>
```

The `kn-cross` attribute is available to every HTML5 tag and is a way to make the element interactive on click. This attribute generates an on click event on the element to open the cross navigation set. If there is no cross navigation set this tag will not work.

```
<div kn-preview="DATASET-TO-SHOW"> ... </div>
```

The `kn-preview` attribute is available to every HTML5 tag and is a way to make the element interactive on click. This attribute generates an on click event on the element to open the dataset preview dialog. The attribute value will be the *dataset label* of the dataset that you want to open. If a dataset is not specified the cockpit will use the one set for the widget. If no dataset has been set and the attribute has no value this tag will not work.

```
<div kn-selection-column="COLUMN-NAME" kn-selection-value="COLUMN-VALUE"> ... </div>
```

The `kn-selection-column` attribute is available to every HTML5 tag and is a way to make the element interactive on click. This attributes generates an on click event on the element to set the chosen column and value in the cockpit selections. The default will use as a selection the first row value for the column.

The **kn-selection-value** attribute is optional and will add a specific value to the column selection.

```
[kn-variable='VARIABLE-NAME' key='VARIABLE-KEY']
```

The `kn-variable` tag is the tool to read the runtime value of one of the defined variables. It will change depending on the current value and can be used inside `kn-if` and `kn-calc`.

The **key** attribute is optional and will select a specific key from the variable object if the variable is “Dataset” type, returning a specific value instead of a complete dataset.

Banned Tags

In order to avoid Cross-site scripting and other vulnerabilities, some tags are automatically removed by the system when saving the cockpit:

- `<button></button>`
- `<object></object>`
- `<script></script>`

If the tag is needed for some specific behaviour (i.e. the button default hover), please replicate it with CSS using a different allowed tag.

Warning: Whitelist Base paths to external resources (images, videos, anchors, CSS files and inline frames) must be declared within `TOMCAT_HOME/resources/services-whitelist.xml` XML file inside Knowage Server, otherwise those external links will be removed by the system. This whitelist file contains safe and trusted websites, to restrict end users of providing unsafe links or unwanted web material. Knowage Server administrator can create or edit it (directly on the file system) to add trusted web sites. Here below you can see an example of

`services-whitelist.xml` file; as you can see, its structure is quite easy: `baseurl` attributes refer to external services, `relativepath` must be used for Knowage Server internal resources instead:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <WHITELIST>
3   <service baseurl="https://www.youtube.com" />
4   <service baseurl="https://player.vimeo.com" />
5   <service baseurl="https://vimeo.com" />
6   <service baseurl="https://media.giphy.com" />
7   <service baseurl="https://giphy.com" />
8   <service baseurl="https://flic.kr" />
9   <service relativepath="/knowage/themes/" />
10  <service relativepath="/knowage/icons/" />
11  <service relativepath="/knowage/restful-services/1.0/images/" />
12 </WHITELIST>

```

Like other widgets the “Style” tab and the “Filters” tab are available in order to set the general style options for the widget and to filter the results displayed in the HTML widget.

5.6.1.10 Map Widget

The Map Widget is useful when a user needs to visualize data related to a geographic position. The widget supports multiple layers, one for every dataset added to widget configuration, and one data field for every layer: the user can switch on-the-fly between all data available on the layer.

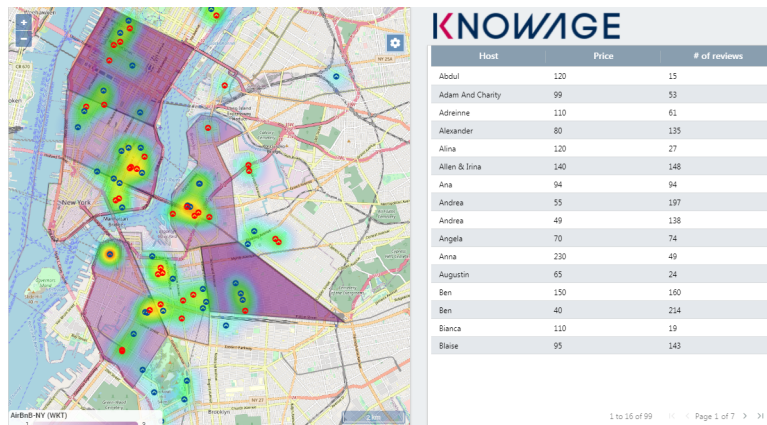




Fig. 5.182: Map widget.

In Map Widget configuration a user can add and remove layers, set the format of the spatial attribute to use and specify the attributes to display on map and on the detail popup:

Every dataset with a spatial attribute is eligible to become a layer in map widget. Only one layer of the widget can be susceptible to user selection: that layer will be the only one with **Target** slide set to on. For each layer a user can also specify its default visibility with **Default visible** slide. Enabling **Static** switch on a layer make it visible and non

clickable, useful when a user wants a fixed background layer with dynamic data from a dataset. With buttons  and  the user can set the metadata and the layer style respectively.

In layer’s metadata, the user can add calculated fields (more on that later) and set the spatial attribute of the dataset that will be used to display a markers on the map. Actually, many spatial attribute types are supported:

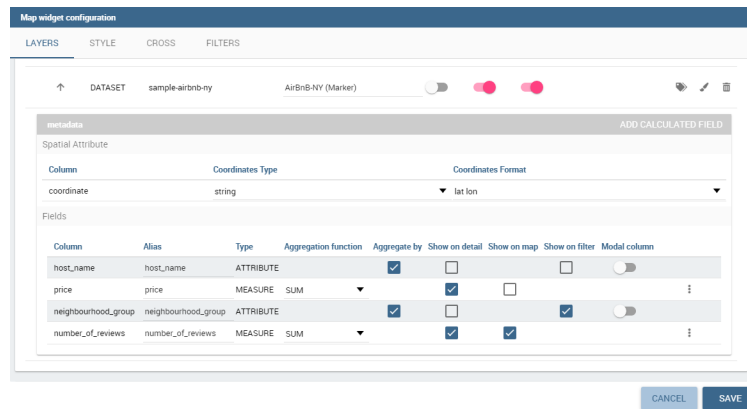


Fig. 5.183: Map widget configuration.

- String format: where the value specify two decimal numbers representing latitude and longitude separated by a space;
- JSON: where the value is a text string in [GeoJSON](#) format;
- WKT: where the value is a text string in [Well-known Text](#) format;

Important: Geographic coordinates format

For every format above user have to specify what is the format of geographic coordinate: user have to specify if latitude comes first or vice versa.

Every field of the dataset, except for the spatial one, can have a custom alias to show on map widget: just double click the label to edit it. A user can also specify if a field have to be shown on detail popup.

For measures a user could specify the aggregation function, if it has to be shown on detail popup and if it has to be shown on map: at least one field has to be shown on map.

For attributes a user could specify if it has to be shown on detail popup or if it has to be show as a filter: in that case, the attribute will be available in the control panel with its all distinct values to let the user to have an immediate evidence of which markers have the selected value for the measure

The 3-dots-menu on the right of each column of the dataset contains additional functionalities: for measures, for example, there is the possibility to specify thresholds.

The threshold menu open a dialog where the user can customize marker color by value range: that's very useful when a user wants to immediately identify a marker by it's value.

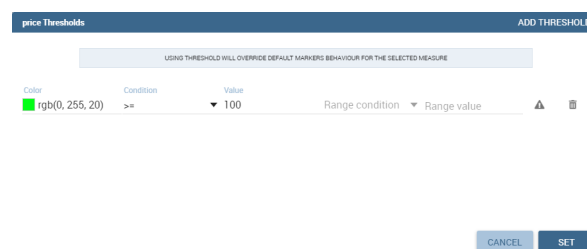


Fig. 5.184: Threshold dialog.

For all the attributes that are filters, a user could select the relative value from the control panel:

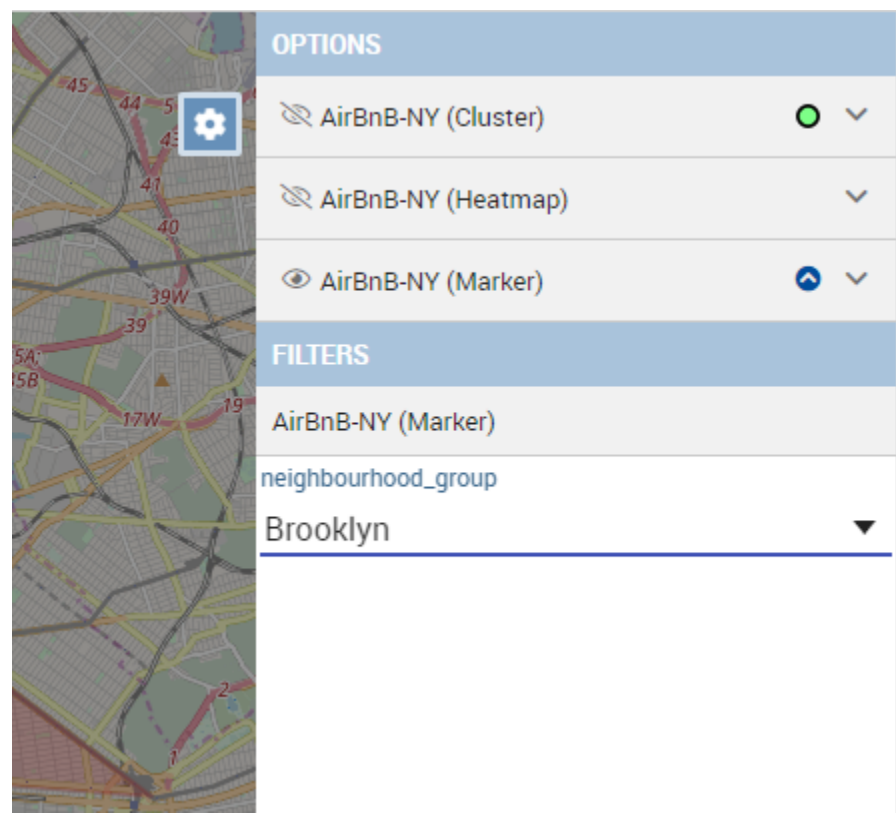


Fig. 5.185: Filter selection.

As said, Map widget supports calculated fields, a way for a user to calculate additional data to show on map or to display into popup detail:

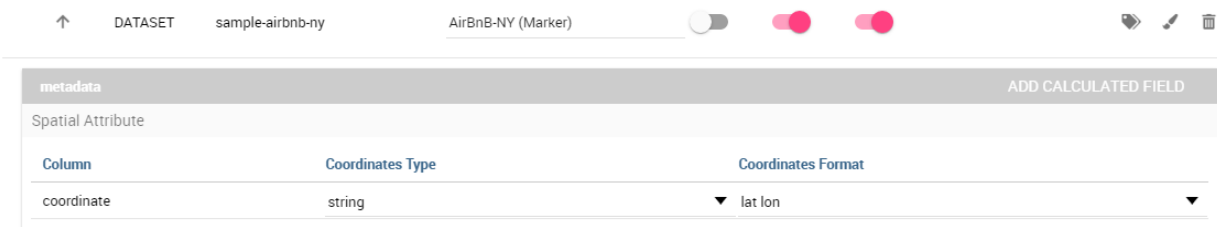


Fig. 5.186: Add calculated field button in layer's metadata.

From the calculated field's dialog a user can combine measures and operations to add more data to the layer. The user can use a SQL-like syntax to create a statement that describe the new calculated field:

The newly calculated field added by the user is shown as a measure in layer's dataset: from the 3-dots menu on the right of the field a user can update or delete the calculated field.

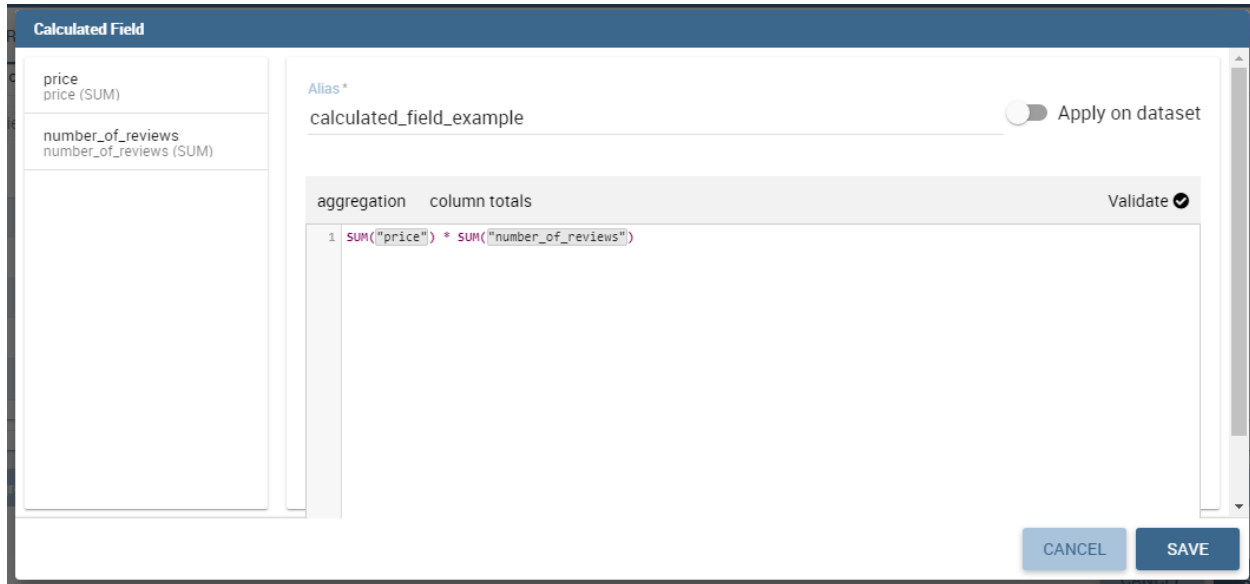


Fig. 5.187: Calculated Field's dialog.

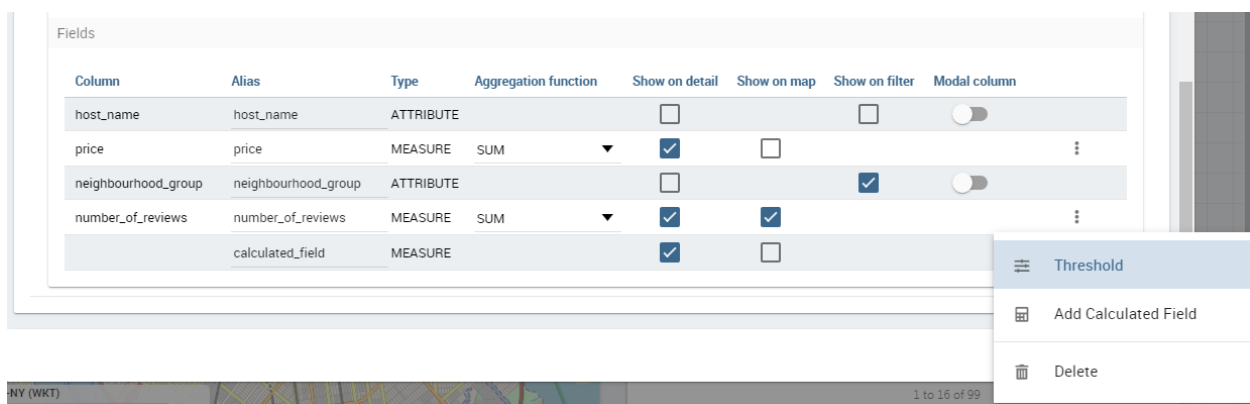


Fig. 5.188: The 3-dots menu on calculated field.

For every layer, a user can specify the way the data will be displayed on map: the user can choose between a markers, cluster, heatmaps and choroplet.

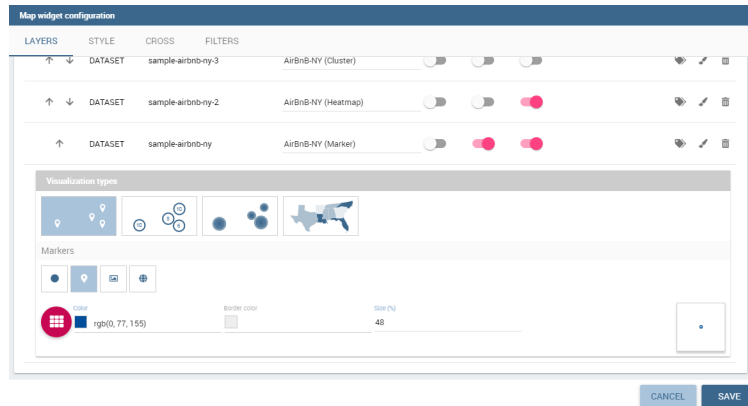


Fig. 5.189: Style configuration for every layer.

For marker there are multiple choices between a user can select. The first one is the standard marker, where a user can select only the marker color:

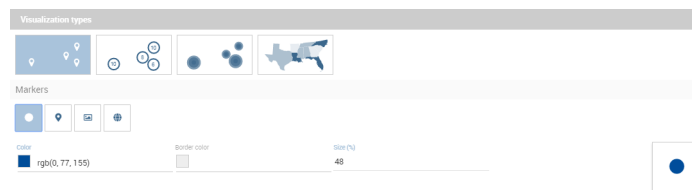


Fig. 5.190: Standard marker configuration.

The second possibility is to use a custom color and custom scale with a custom marker, for example and icon available in Font Awesome catalog:

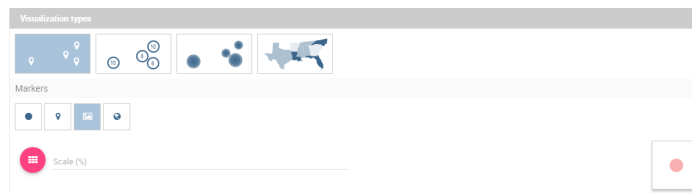


Fig. 5.191: Custom marker configuration.

A user can also use an image from Knowage media as a marker:

Finally a user can use an image from external URL as a marker:

Cluster visualization renders circles of different size where every circle aggregating positions by relative values. A user can zoom in to disaggregate the cluster until he see the single data. For this type of visualization, a user can set size and color of the circle and the size and the color of the font used to display the aggregated value:

When heatmap is selected, a user can display values by areas colored by a color range from green to red where the values are respectively lower and higher. Setting the radius and the blur, a user can specify the scale of the areas and the scale of the blur around it:

The choroplet visualization allows a user to precisely associate values to areas, very useful when spatial attribute specify a geometry instead of a single point. The classes method specify the subdivision algorithm and the classes

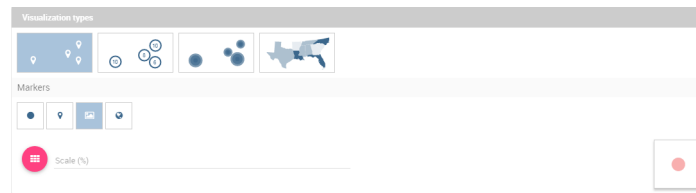


Fig. 5.192: Marker from Knowage images.

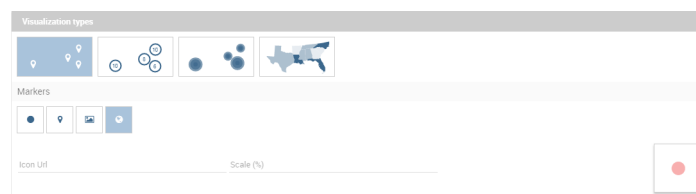


Fig. 5.193: Marker from Knowage images.

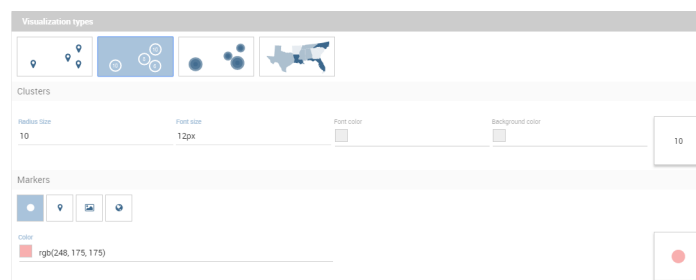


Fig. 5.194: Cluster configuration.

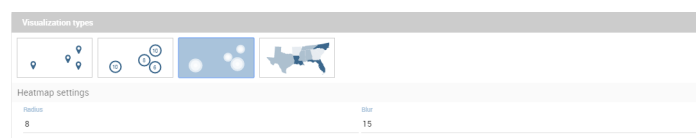


Fig. 5.195: Heatmap configuration.

number specify how many subdivision to make; the colors specify the start and the end of the range color that will follow the same range of the values:

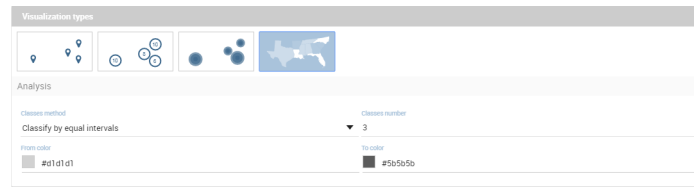


Fig. 5.196: Choropleth configuration.

5.6.1.11 Discovery Widget

Q

datanotifica

^

October 25, 2018 2:00 AM

13482

October 26, 2018 2:00 AM

12885

September 15, 2017 2:00 AM

12724

October 18, 2018 2:00 AM

12476

February 21, 2018 1:00 AM

10472

October 22, 2018 2:00 AM

10267

November 20, 2018 1:00 AM

9306

February 19, 2018 1:00 AM

9243

September 5, 2017 2:00 AM

8884

February 22, 2018 1:00 AM

8797

denominazionelista

^

FERRARA_ING_FISCALI_CDS_65_20102017

85262

TARI_CARICO_2016_I_LOTTO

81067

SANZIONI CODICE DELLA STRADA (CDS)

66519

FERRARA_ING_FISCALI_CDS_CO8BBL

61348

TARI2017_PARZIALI LOTTO II

61275

Accertamenti Tares (file corretto)

53251

TARI 2014 I LOTTO OMESSI VERSAMENTI

52616

TARI_CARICO_2016_I_LOTTO_02

52600

aggregazio...

annoemiss...

categoriaemissione

annull...

codiceb...

codic...

com...

comu...

conso...

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

CONSEGTI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

COMGIACI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

COMGIACI

CIRC001

IMOLA

Si

NUOVO CIRCOI

2015

Standard

0

E289

COMGIACI

CIRC001

IMOLA

Si

1 to 10 of 216284

< < Page 1 of 216284 > >

The Discovery Widget is used to easily use and navigate into a Solr Dataset using facets aggregation and a table results. In order to make searches, aggregations using facets and so on, after selecting the Solr dataset it is possible to choose the fields that should be shown as the result. The table result can also be configured to show a limited set of fields, please open edit mode:

Settings

The settings tab contains the management of the 3 elements that compose a directive:

- **Data table:** enabled by default is the grid containing data. You can choose the number of item per page.
- **Facets:** if enabled the sidepanel with the facets will appear. It is also possible to configure facets options:
 - *enable selection on facets*, if enabled a user click on the facets will throw a cockpit selection instead of just filtering the table.
 - *closed by default*, if enabled the facets will be visible as closed groups by default.
 - *facets column width*, this setting allows to choose the dimension of the facets column in px, rem or percentage values.

Column	Alias	Type	Style	Show Column	Show Facet	Enable text search
aggregazione	aggregazione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
annoemissione	annoemissione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
annullato	annullato	MEASURE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
categoriaemissione	categoriaemissione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
codicebelfiore	codicebelfiore	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
codicecausalenotifica	codicecausalenotifica	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
commissa	commissa	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
comune	comune	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
consolidato	consolidato	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
datadocumenti	datadocumenti	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
dataemissione	dataemissione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
datanotifica	datanotifica	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
datapostalizzazione	datapostalizzazione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
denominazioneentrata	denominazioneentrata	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
denominazioneuscita	denominazioneuscita	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
discaricato	discaricato	MEASURE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
emissione	emissione	MEASURED		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

– *facets max number*, this setting allows to choose the maximum number of facets visible for every field.

- Text search: if enabled a searchbar will appear at the top of the widget. It is possible to set a default search for widget initialization.

Enable Data Table

Pagination Items Number
10

Facets

☒ Enable Selection on Facets ☐ Closed by default

Facets Column Width
px, rem or % measure units are available

Facets max number
10

Text search

☐ Default Text search

Type

Important The options “show column” and “show facets” are only frontend side. They don’t affect the real backend Solr query, discovery widget will search for every field even though they are frontend omitted.

Facets column ordering

It is possible to change the facets column ordering, for example if there is the need to move up a field.

As shown in this example, “aggregazione” should be shown upper, just go to the edit widget section:

annoemissione 	
2018	795855
2017	644009
2016	418385
2015	216086
2014	71076
2106	6381
2019	3881
aggregazione 	
MODENA	505576
NUOVO CIRCONDARIO IMOLESE	367450
FERRARA	273149
CASALECCHIO DI RENO	212909
FORLI'	190713
RAVENNA	157556
RIMINI	156257
CESENA	133426
CESENATICO	80068



Edit Discovery widget						
DATASET SETTINGS STYLE CROSS FILTERS						
Dataset						
emissioni_rer_sor (emissioni_rer_sor)						
Columns						
Column	Alias	Type	Style	Show Column	Show Facet	Enable text search
aggregazione	aggregazione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
categoriaemissione	categoriaemissione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
annoemissione	annoemissione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
annullato	annullato	MEASURE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
codicebelfiore	codicebelfiore	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

And change the columns order dragging the field to the right position.

aggregazione	
MODENA	505576
NUOVO CIRCONDARIO IMOLESE	367450
FERRARA	273149
CASALECCHIO DI RENO	212909
FORLI'	190713
RAVENNA	157556
RIMINI	156257
CESENA	133426
CESENATICO	80068
FAENZA	51491
annoemissione	
2018	795855
2017	644009
2016	418385
2015	216086
2014	71076
2013	6381

Changing Date Format for discovery table date columns

It is also possible to change the format used to show date columns inside discovery table: In order to do that, click on style for date columns fields in edit mode

Columns						
Column	Alias	Type	Style	Show Column <input checked="" type="checkbox"/>	Show Facet <input checked="" type="checkbox"/>	Enable text search <input checked="" type="checkbox"/>
datanotifica	datanotifica	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
datapostalizzazione	datapostalizzazione	ATTRIBUTE		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

And change the “date format” property

5.6.1.12 Python/R Widget

The Python/R widgets allow to directly embed Python or R scripts inside the cockpit in order to create advanced custom analytics.

In the editor tab it is possible to add the script that will be sent to the execution engine.

Before writing the code it is necessary to specify the **type** of the output produced by the script. Knowage has support for three different output types:

- Image

COLUMN STYLE

Font-color

Select a color

Background-color

Select a color

Font-family

▼ Font Size

px, rem or % measure units are available

Font-weight

▼ Font Style

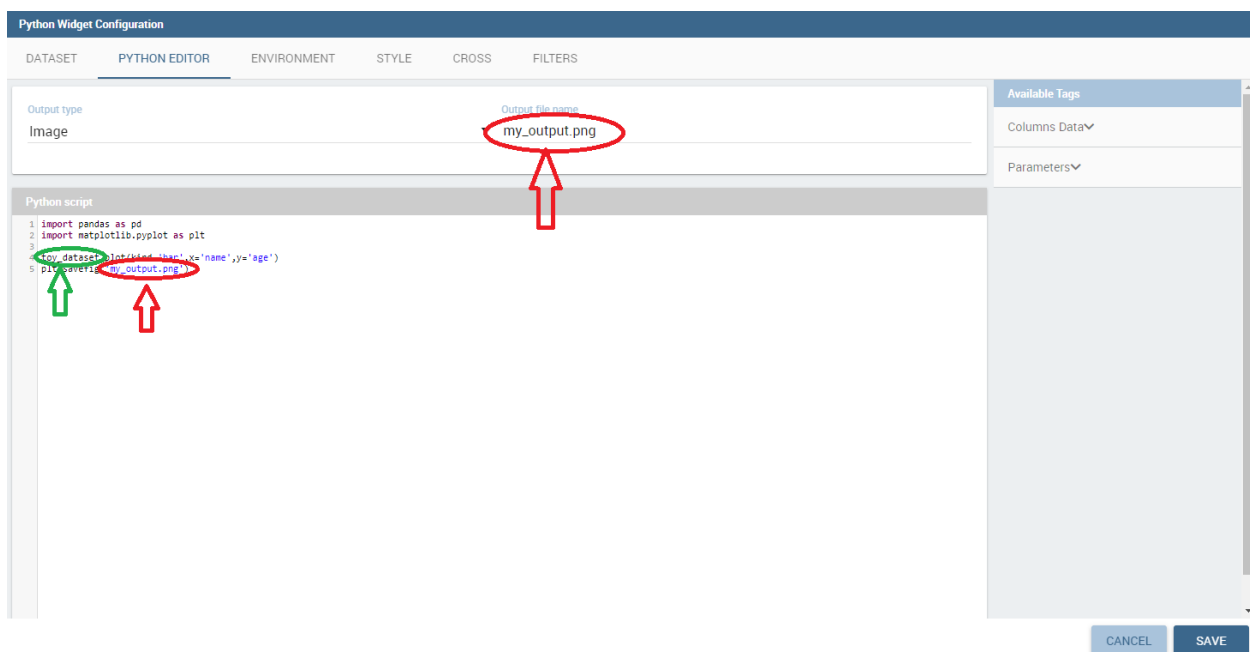
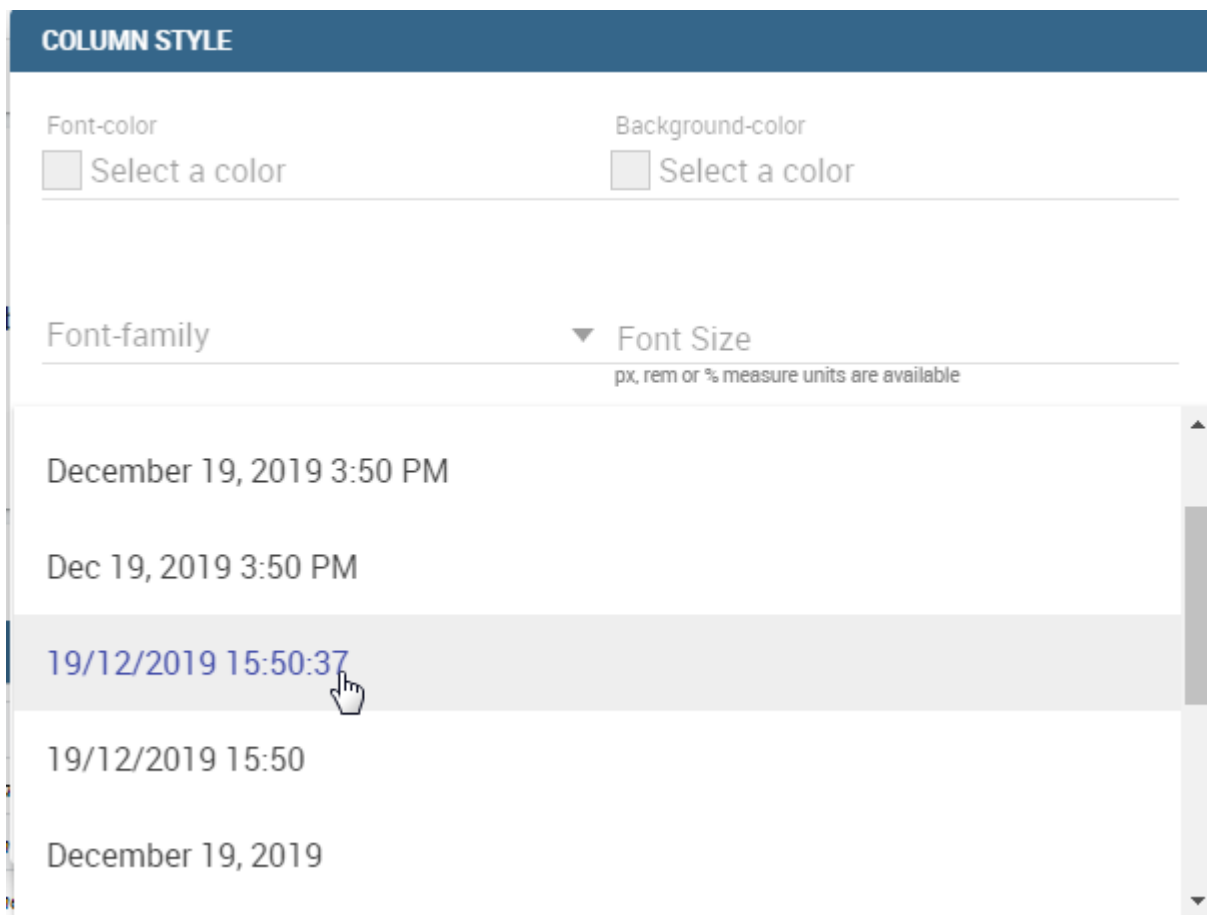
▼ Cell horizontal align... ▼

Date Format

19/12/2019 15:50:37 ▼

CANCEL

SAVE



- HTML
- Bokeh application (only for Python)

If **Image** or **HTML** is chosen as output type, it is also necessary to specify the name of the file in which the script will save its output.

The Dataset tab allows the user to select a dataset that will be accessible directly from the code. After choosing a dataset the list of available columns will be shown. Here it is also possible to order the dataset according to a column and to select the ordering type (ascending or descending).

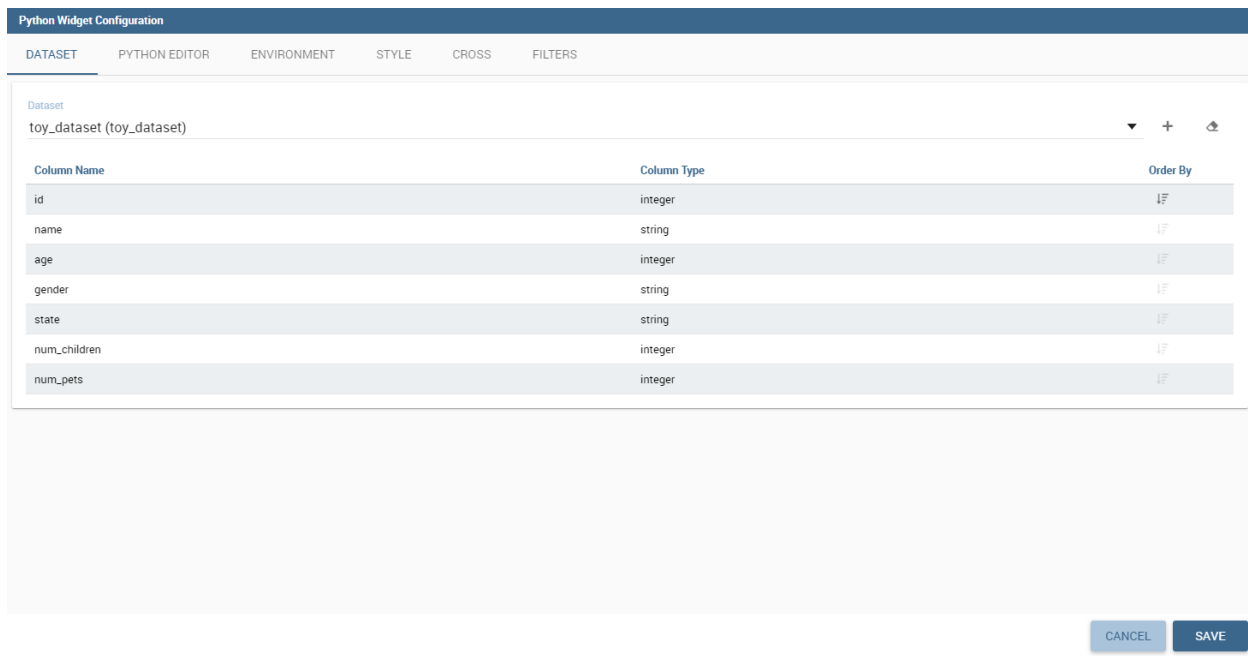


Fig. 5.197: Dataset selection

Once a dataset has been chosen, it will be possible to access it directly from the code via a **dataframe** variable. This variable will have the same name of the dataset label.

The Environment tab allows the user to choose among a list of available Python/R environments previously defined inside the **Configuration Management**. To support this kind of choice a list of available libraries is displayed for each selected environment.

Inside Python and R scripts it is possible to access analytical drivers by the usual placeholder syntax $\$P\{\}$.

Warning: This widget is sensible to associative logic, meaning that the widget is updated every time that an association is changed, but it DOES NOT trigger associative logic itself.

5.6.1.13 Custom Chart Widget

The Custom Chart allows the user to directly embed html,css and js code using a supported external chart library and integrating with Knowage data and interactions using custom API.

This widget will be available only if the *create custom chart widget* option is set for the specified role.

The Edit section of the widget is composed by five tabs: dataset, editor, style, cross and filters.

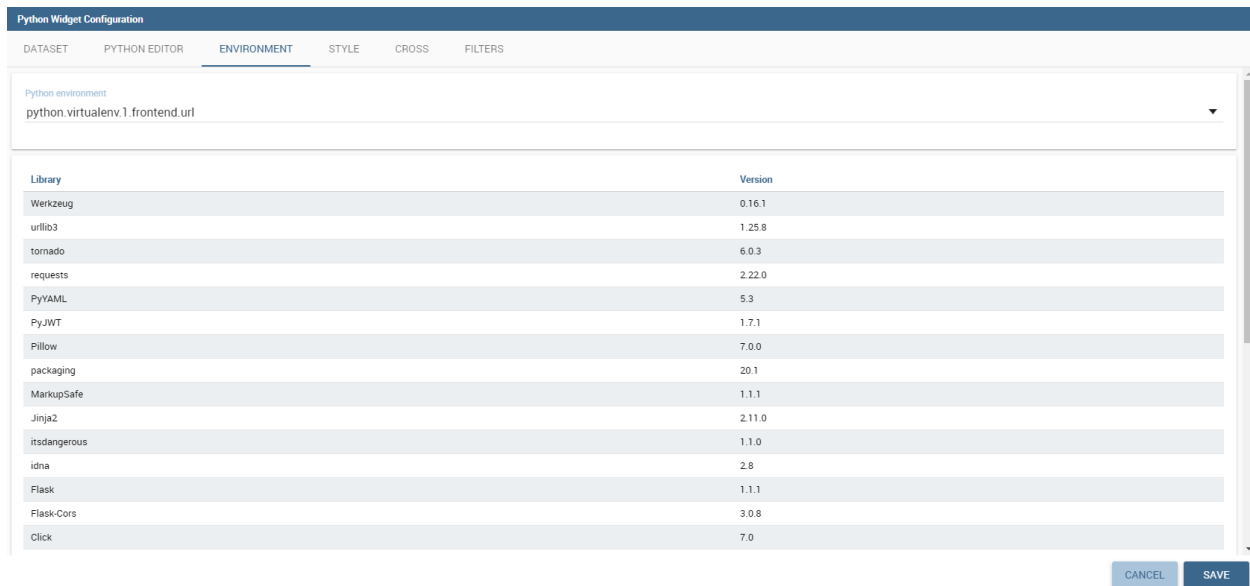


Fig. 5.198: Environment selection

The **dataset tab** allows to select a specific dataset to use as a referral for the API. Once the dataset has been selected a table with the columns list will appear below. In the table will be possible to change column alias, the column aggregation for measures and delete columns interacting with the selected column line. Clicking on *add column* or *add calculated field* buttons on top a popup will appear allowing to choose one of the dataset column to add or to insert the calculated field formula.

The **Editor tab** allows to insert custom code and it's splitted into three components: CSS, HTML, JavaScript.

The CSS component allows to insert css classes that will be used by the HTML code of the widget. It's also possible to use *@import* command if the referred url is inside the whitelist.

The HTML component allows to insert HTML tags in order to create a structure to host the custom chart and additional structural informations.

The JavaScript component is the code section, and allows to insert the custom chart code, custom Javascript code and the API usage.

To use the api the keyword is **datastore**. datastore is object that contains the actual data; it has methods to iterate over results and get all values, as the Java counterpart, plus some other methods as the following:

getDataArray

returns: *data array*

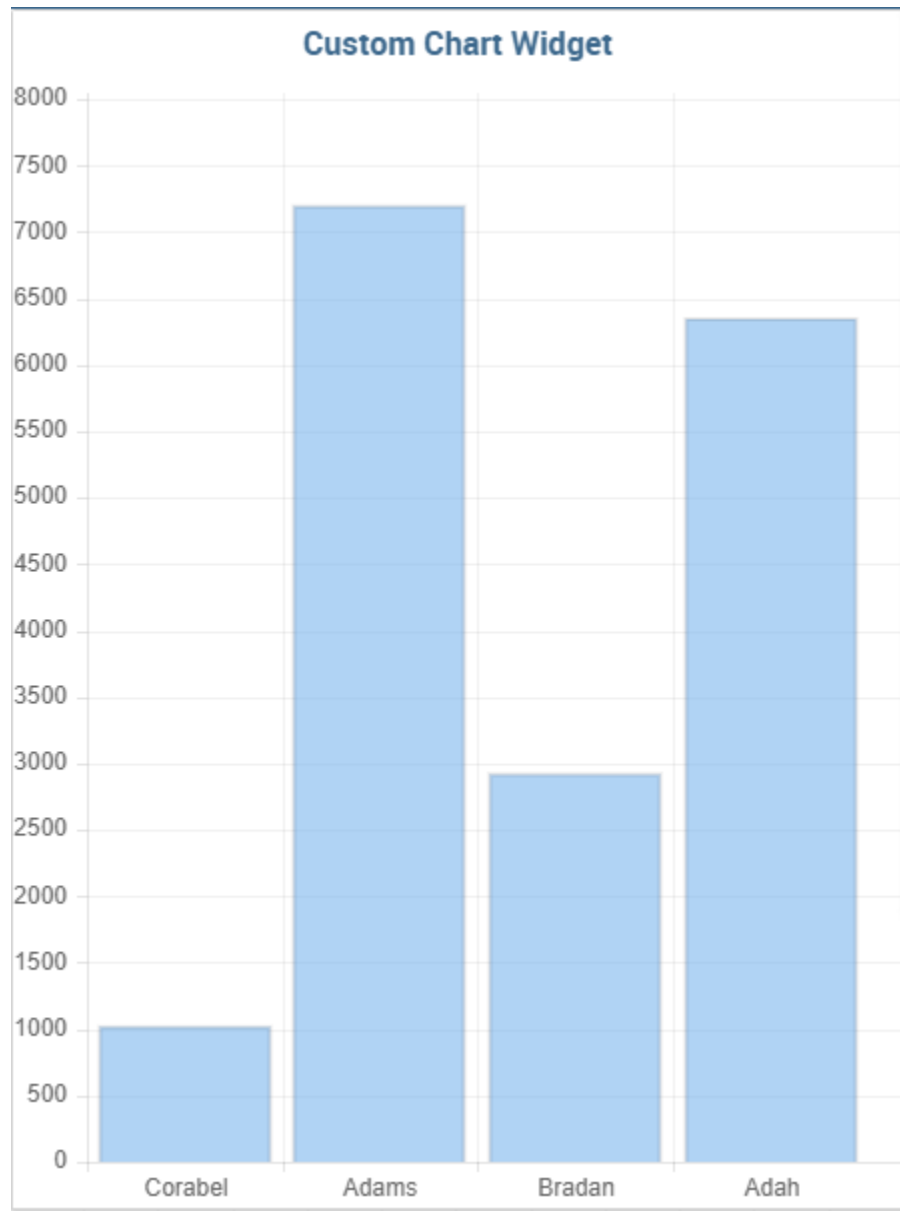
params: *custom user function*

example:

```

1  datastore.getDataArray(function(record){
2      return {
3          name: record.city,
4          y: record.num_children_at_home
5      }
6  })

```



Edit custom chart widget

DATASETEDITORSTYLECROSSFILTERS

Dataset

testDataset2 (testDataset2)

ADD COLUMNADD CALCULATED FIELD

Name	Alias	Type	Data Type	Aggregation	
id	<input checked="" type="checkbox"/> id	MEASURE	# float	<input checked="" type="checkbox"/> SUM	
first_name	<input checked="" type="checkbox"/> first_name	ATTRIBUTE	string		
int	<input checked="" type="checkbox"/> int	MEASURE	# float	<input checked="" type="checkbox"/> SUM	
email	<input checked="" type="checkbox"/> email	ATTRIBUTE	string		
gender	<input checked="" type="checkbox"/> gender	ATTRIBUTE	string		
float	<input checked="" type="checkbox"/> float	MEASURE	# float	<input checked="" type="checkbox"/> SUM	

CANCELSAVE

Edit custom chart widget

DATASETEditorSTYLECROSSFILTERS

css

html

JavaScript

```
1 var canvas = document.getElementById('myChart');
2 var ctx = document.getElementById('myChart').getContext('2d');
3 var templabels = datastore.hierarchy({'levels':['first_name'],'measures':['int':'MAX']}).getChild(0).name;
4 var labels = datastore.filter({'first_name':'Ada'}).getColumn('first_name');
5
6 labels[0] = templabels;
7
8 var data = {
9   labels: labels,
10  datasets: [
11    {
12      label: 'w of votes',
13      data: datastore.filter({'first_name':'Ada'}).getColumn('int'),
14      fillColor: 'rgba(124,121,236,0.6)'
15    }
16  ]
17 };
18 var options = {
19   scales: [
20     {
21       yAxes: [
22         {
23           ticks: {
24             beginAtZero: true
25           }
26         }
27       ]
28     }
29   ]
30 };
31 var myChart = new Chart(ctx).Bar(data,options);
32
33 canvas.onclick = function(evt){
34   var point = myChart.getBarsAtEventCross(evt)[0];
35   datastore.clickManager('first_name',point.label)
36 }
```

CANCELSAVE

result:

```
1  [  
2      {  
3          name: 'New York',  
4          y: 5  
5      },  
6      {  
7          name: 'Boston',  
8          y: 3  
9      }  
10 ]
```

getRecords

returns: array of objects; each object has nameOfDsColumn: value

params: no params

example:

```
1  datastore.getRecords()
```

result:

```
1  [  
2      {  
3          city: 'New York',  
4          total_children: 5,  
5          country: 'USA'  
6      },  
7      {  
8          name: 'Boston',  
9          total_children: 3,  
10         country: 'USA'  
11     }  
12 ]  
13
```

getColumn

returns: array of values for one dataset column

params: dataset's column name

example:

```
1  datastore.getColumn('country')
```

result:

```
1 ['USA', 'Mexico', 'Canada']
```

getSeriesAndData

returns: array of series with data for each series

params: serie/measure name, custom user function

example:

```
1 datastore.getSeriesAndData('PRODUCT_FAMILY',function(record){
2   return {
3     y: record.UNIT_SALES,
4     name: record.QUARTER
5   }
6 })
```

result:

```
1 [
2   {
3     name: 'Drink',
4     data: [
5       {
6         y: 5000,
7         name: 'Q1'
8       },
9       {
10        y: 7000,
11        name: 'Q2'
12      }
13     ]
14   },
15   {
16     name: 'Food',
17     data: [
18       {
19         y: 6000,
20         name: 'Q1'
21       },
22       {
23         y: 4000,
24         name: 'Q2'
25       },
26       {
27         y: 3000,
28         name: 'Q3'
29       }
30     ]
31   }
32 ]
```

sort - angular sort service (sorting is executed on the client side)

returns: datastore sorted by dataset's column/s

params: dataset's column name

optional: sort type object {column:'asc/desc'}

example1:

```
1 datastore.sort('STORE_ID') //by default, it is asc
2 OR:
3 datastore.sort({'STORE_ID':'asc'})
```

filter - angular filter service (filtering is executed on the client side)

returns: datastore filtered by some value for dataset's column/s

params: object that contains dataset's columns names for properties -> value to be filtered

example:

```
1 datastore.filter({'QUARTER':'Q1','STORE_ID':'1'})
```

hierarchy

returns: hierarchy object with its functions and tree

params: object that contains property levels -> array of dataset's columns names

optional: same object with optional property measures -> object that contains dataset's columns names for properties
-> aggregation function (sum, min, max)

example:

```
1 var hierarchy = datastore.hierarchy({'levels':['QUARTER','PRODUCT_FAMILY'],'measures': {'UNIT_SALES':'SUM'}})
```

result:

```
1 [
2   {
3     "name": "Q1",
4     "children": [
5       {
6         "name": "Non-Consumable",
7         "children": [],
8         "UNIT_SALES": 7.4571
9       },
10      {
11        "name": "Food",
12        "children": [],
13        "UNIT_SALES": 12
14      }
15    ],
16    "UNIT_SALES": 19.4571
17  },
18  {
19    "name": "Q2",
20    "children": [
21    ]
22  }
23 ]
```

(continues on next page)

(continued from previous page)

```

22         "name": "Non-Consumable",
23         "children": [],
24         "UNIT_SALES": 9.9429
25     },
26     {
27         "name": "Food",
28         "children": [],
29         "UNIT_SALES": 7.2
30     }
31 ],
32 "UNIT_SALES": 17.1429
33 }
34 ]

```

getChild

returns: node of hierarchy (node is Node object)

params: index of child in hierarchy

example:

```
1 hierarchy.getChild(0)
```

result:

```

1 {
2     "name": "Q1",
3     "children": [
4         {
5             "name": "Non-Consumable",
6             "children": [],
7             "UNIT_SALES": 7.4571
8         },
9         {
10            "name": "Food",
11            "children": [],
12            "UNIT_SALES": 12
13        }
14    ],
15    "UNIT_SALES": 19.4571
16 }

```

getLevel

returns: array of nodes of hierarchy on specific level

params: index of level in hierarchy

example:

```
1 hierarchy.getLevel(0)
```

result:

```

1  [
2      {
3          "name": "Q1",
4          "children": [
5              {
6                  "name": "Non-Consumable",
7                  "children": [],
8                  "UNIT_SALES": 7.4571
9              },
10             {
11                 "name": "Food",
12                 "children": [],
13                 "UNIT_SALES": 12
14             }
15         ],
16         "UNIT_SALES": 19.4571
17     },
18     {
19         "name": "Q2",
20         "children": [
21             {
22                 "name": "Non-Consumable",
23                 "children": [],
24                 "UNIT_SALES": 9.9429
25             },
26             {
27                 "name": "Food",
28                 "children": [],
29                 "UNIT_SALES": 7.2
30             }
31         ],
32         "UNIT_SALES": 17.1429
33     }
34 ]

```

node is an instance of Node object. It has convenient functions to explore the node:

```

1  var node = hierarchy.getChild(0)

```

result:

```

1  {
2      "name": "Q1",
3      "children": [
4          {
5              "name": "Non-Consumable",
6              "children": [],
7              "UNIT_SALES": 7.4571
8          },
9          {
10             "name": "Food",
11             "children": [],
12             "UNIT_SALES": 12
13         }
14     ],
15     "UNIT_SALES": 19.4571
16 }

```

getValue

returns: a measure's value for a specific hierarchy's child(node)

params: dataset's measures's name

example:

```
node.getValue('UNIT_SALES')
```

result: 19.4571

getChild

returns: a specific node's child

params: index of nodes's child

example:

```
node.getChild(0)
```

result:

```
{
  "name": "Non-Consumable",
  "children": [],
  "UNIT_SALES": 7.4571
}
```

getParent

returns: a node parent of specific child

params: no params

example:

```
node.getChild(0).getParent()
```

result:

```
{
  "name": "Q1",
  "children": [
    {
      "name": "Non-Consumable",
      "children": [],
      "sales": 7.4571
    }
  ]
}
```

(continues on next page)

(continued from previous page)

```
8         },
9         {
10             "name": "Food",
11             "children": [],
12             "sales": 12
13         }
14     ],
15     "sales": 19.4571
16 }
```

getChildren

returns: an array of node's children

params: no params

example:

```
1 node.getChildren()
```

result:

```
1 [
2     {
3         "name": "Non-Consumable",
4         "children": [],
5         "sales": 7.4571
6     },
7     {
8         "name": "Food",
9         "children": [],
10        "sales": 12
11    }
12 ]
```

getSiblings

returns: an array of node siblings to a specific child

params: no params

example:

```
1 node.getChild(0).getSiblings()
```

result:

```
1 [
2     {
3         "name": "Non-Consumable",
4         "children": [],
```

(continues on next page)

(continued from previous page)

```

5         "sales": 7.4571
6     },
7     {
8         "name": "Food",
9         "children": [],
10        "sales": 12
11    }
12 ]

```

It is also possible to interact with the other cockpit widgets, to do so it's possible to use the **clickManager**:

```
1 datastore.clickManager(columnName, columnValue);
```

This method can be added everywhere the code is managing a click event, and will notify Knowage about the interaction.

The default case (if no cross-navigation or preview-navigation is set) will throw a selection with the dataset column name and column value set in the method.

If a cross-navigation or a preview has been set in the cross tab, those will have priority on the selection and will throw the set interaction. The dynamic values used will be the ones set in the method arguments.

Important: Chart libraries

As a default Knowage supports natively Chart.js (version 1.0.2) for the Community edition and Highcharts.js (version 7.1.1) for the Enterprise Edition. In CE and EE, Knowage supports d3.js library (version 3.5.5). It is possible also to include other libraries adding the CDN script tag in the html Editor. Be aware that url not set in the whitelist will be deleted on save.

Warning: Whitelist For security reasons no dangerous Javascript code can be added to html tags. Every tag considered dangerous will be deleted on save by the filter. Base paths to external resources (images, videos, anchors, CSS files and inline frames) must be declared within TOMCAT_HOME/resources/services-whitelist.xml XML file inside Knowage Server, otherwise those external links will be removed by the system. This whitelist file contains safe and trusted websites, to restrict end users of providing unsafe links or unwanted web material. Knowage Server administrator can create or edit it (directly on the file system) to add trusted web sites. Here below you can see an example of services-whitelist.xml file; as you can see, its structure is quite easy: `baseUrl` attributes refer to external services, `relativepath` must be used for Knowage Server internal resources instead:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <WHITELIST>
3   <service baseUrl="https://www.youtube.com" />
4   <service baseUrl="https://player.vimeo.com" />
5   <service baseUrl="https://vimeo.com" />
6   <service baseUrl="https://media.giphy.com" />
7   <service baseUrl="https://giphy.com" />
8   <service baseUrl="https://flic.kr" />
9   <service relativepath="/knowage/themes/" />
10  <service relativepath="/knowage/icons/" />
11  <service relativepath="/knowage/restful-services/1.0/images/" />
12 </WHITELIST>

```

Like other widgets the “Cross”, “Style”, and the “Filters” tab are available in order to set the general style options for the widget and to filter the results displayed in the HTML widget.

5.6.1.14 Cross Navigation

Warning: Cross tab is only for technical users

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

All widgets (except selector and active selections) have the Cross tab available, that allows the user to interact with the widget in different ways: - setting a Cross-navigation between different documents - setting a Preview of a specific dataset to popup - opening an external link The interactions are mutually exclusive, so just one type can be chosen for every widget.

Cross-navigation

The cross-navigation gives the possibility to connect two documents clicking on a widget as a starting point, opening the second one as arrival and showing the breadcrumbs on top.

Fig. 5.199: Table widget cross-navigation configuration.

To enable one first of all is necessary to set a cross navigation inside the “cross-navigation definition” functionality. There is possible to set the starting/arrival point and to set the connection between different output/input parameters.

After this passage you will have the possibility to enable the cross-navigation in the tab and choose the user interaction that will start the navigation, if more than one are available.

The most complex example is the table widget cross-navigation, because it allows 3 different interactions:

- *Click on the whole row*, where the interaction will start clicking on any row
- *Click on a single column*, where the editor will choose a specific column that will start the interaction (the user will see the column values underlined)
- *Click on an icon*, where the editor will choose an icon positioned to the right side that will start the interaction.

Once the interaction has been chosen you will be able to select the cross-navigation created before. If you created more than one related to that document you will be able to choose between them. If you leave this field blank the user will be able to choose that himself.

Preview

Fig. 5.200: Table widget preview configuration.

The preview configuration is very similar to the cross-navigation one. First of all you will need to enable the navigation using the switch button. After you will need to choose an interaction type and a target dataset. the selected dataset

will be opened in a popup window but, if you check the “Direct download” property you will be able to get the dataset preview directly in the download list.

country string	city string	color string	currency string	gender string	company string
France	Marseille	Turquoise	Euro	Female	Trantow Group
France	Calais	Goldenrod	Euro	Female	Sawayn LLC
Russia	Glinka	Green	Ruble	Male	Hintz, Streich and Rippin
Russia	Uritsk	Crimson	Ruble	Male	Crooks, Hintz and Klehn
France	Montluçon	Red	Euro	Female	Yundt, Sawayn and Morar
France	Paris 02	Purple	Euro	Male	Hodkiewicz Inc
Russia	Porkhov	Purple	Ruble	Female	Jacobs, Wunsch and Sauer
Russia	Taldan	Blue	Ruble	Male	Schaden, Shields and Toy
Russia	Vodstroy	Green	Ruble	Female	Zemlak-Reynolds
Russia	Novoanninskiy	Crimson	Ruble	Female	Bins-Langworth
Russia	Bobrov	Khaki	Ruble	Male	Rice LLC
Russia	Kukuhtan	Purple	Ruble	Female	Oberbrunner-Cole
Russia	Vostryakovo	Violet	Ruble	Female	Botsford Group
France	Marseille	Fuscia	Euro	Male	Stark, Mann and Ortiz
Russia	Malinovskiy	Green	Ruble	Female	Littel and Sons

Fig. 5.201: Preview example.

Cross and Preview Parameters

Output parameters list

	Type	Value
<input checked="" type="checkbox"/> parameter2	Static	42
<input type="checkbox"/> parameter1		

Fig. 5.202: Parameters example.

Both cross-navigation and preview navigation share the parameters management. If one or more output parameters are available, you will find the list below the navigation page. You can choose which values to use clicking on the checkbox at the right side of the name, than you will have the possibility to choose between different modes to get the value:

- *Static*, entering a static value
- *Dynamic*, passing the value of the column at the selected row (or passing the column name if *Selected Column Name* is chosen)
- *Selection*, passing the current value of the selection for the specified dataset and column. Warning: the selection is not triggered clicking on the navigation, so to pass this value the selection must already be present.

Once the user will click on the widget, those parameters will be evaluated before the navigation.

Link

The link configuration is very similar to the cross-navigation one too. First of all you will need to enable the link navigation using the switch button. After you will need to choose an interaction type and a base url. The base url will be the url opened by the user click. You can also decide the type of link between the opening of a new page or the document replace opening in the same page.

Link Parameters

Link configuration interface showing the following details:

- Interaction Type:** Enable click on a single column
- Column:** first_name (Optional column to use as model)
- Base url:** http://www.test.com (The url of the page that will be opened)
- Link type:** Open in new page
- Url Parameters:**

Parameter key	Parameter type
test	JSON
- JSON Example:**

```

1 {
2   "parameters": {
3     "variable": $V(myvariable),
4     "fieldvalue": $F(myfield),
5     "parametervalue": $P(parameter)
6   }
7 }

```

Fig. 5.203: Link configuration with JSON parameter example.

The link parameters are different from the previous because they will be used to make the baseUrl selected more precise. The resulting example url will be something similar: `http://www.knowage-suite.com?*=parameter1*=value&*=parameter2*=value2`

To do so you will have to create and name different parameters, the selected name will be the one used in the url construction. You can choose how to get the parameter value with the following modes:

- *Static*, entering a static value
- *Dynamic*, passing the value of the column at the selected row (or passing the column name if *Selected Column Name* is chosen)
- *Selection*, passing the current value of the selection for the specified dataset and column. Warning: the selection is not triggered clicking on the navigation, so to pass this value the selection must already be present.
- *Analytical Driver*, passing a page parameter (input driver)
- *JSON*, passing an escaped JSON inside the url parameter. You can use the editor to create the desired JSON and use the placeholder to set the value at runtime. The placeholders are the usuals `$F{field}` for the fields, `$P{parameter}` for the parameters, `$V{variable}` for the variables.
- *JWT*, passing the JWT token of the user

5.6.1.15 Widget properties

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



Fig. 5.204: Widget properties.

Here the user can:

- move the widget in the cockpit area at convenience;
- modify its dimension;
- 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.





When executing the cockpit in visualization mode, the user has also some more options for widgets. For all widget the user can use the icon  to expand the widget to all page and use the icon  to reduce it again. There are also two new widget options: using the icon  it is possible to capture the screenshot of the widget and clicking on the icon  the data plotted on a chart or displayed in a table or crosstab are exported in an excel file.

Chart widget are endowed with an additional option that allows the user to change the chart type, as you can see in Figure below.

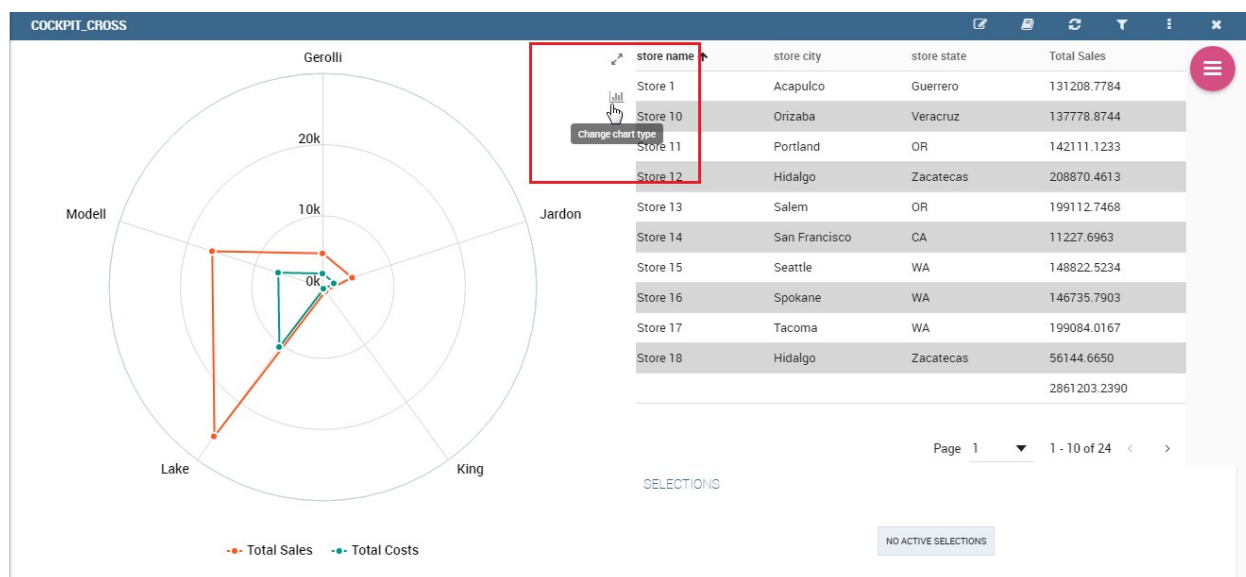


Fig. 5.205: Change chart type button.

Referring to figure below, the available chart types are: parallel, scatter, wordcloud, line, radar, bar and pie.

Pay attention though to the fact that when grouping functions have been used, the change chart type may not report the same level of aggregation. In fact, not all type of chart allows the grouping function. Refer to Chart types in detail to read more about each chart type configuration. Pay also attention when a two-series chart is changed with a single-series one. For instance the parallel chart works only when (at least) two series have been set, while the wordcloud works with only one series.

5.6.2 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 below opens. This contains the **General Settings** tab and the **Widget Style** tab.

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 or a background image and its size. In particular, in order to add a background image for the sheets, firstly you have to add the image to the catalogue of the image widget and then copy the link of the image. It is also possible to decide to enable the menu and the widgets functionalities when the document runs in display mode or to disable the screenshot functionality for every widgets.

The second tab (Figure below allows to configure some style options of the cockpit, like borders, shadows, titles and background color.

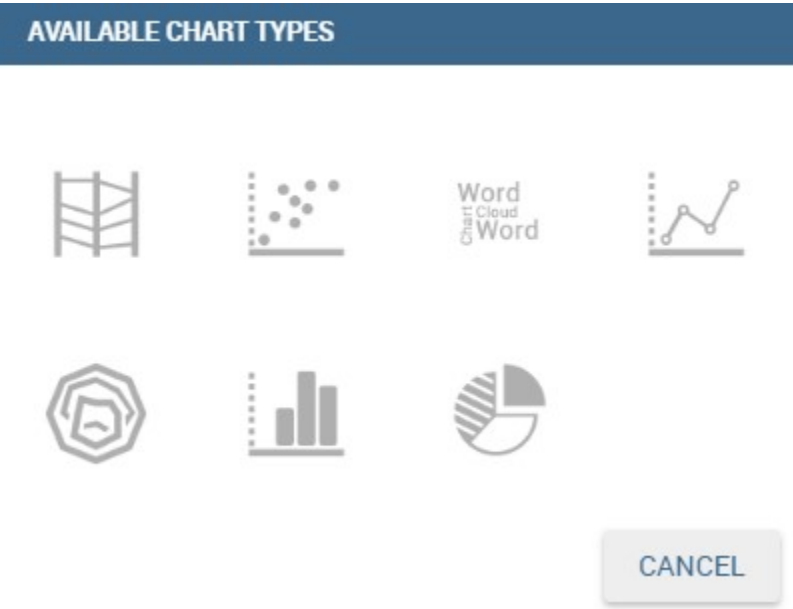


Fig. 5.206: Available chart types.

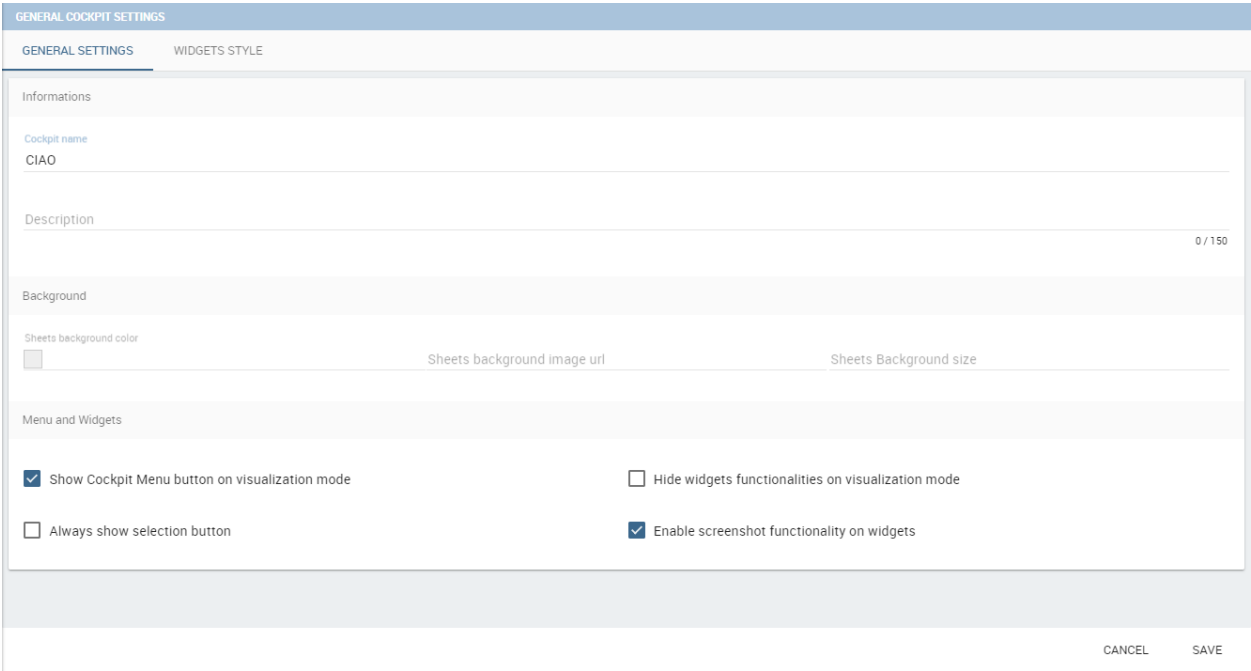


Fig. 5.207: General configuration window.

GENERAL COCKPIT SETTINGS

GENERAL SETTINGS **WIDGETS STYLE**

TITLES

Title text Horizontal alignment Font Family

Font Size Font Style Font Weight

px, rem or % measure units are available

Header Height Title Color Title Background Color

Height in px of the space available to title Select a color Select a color

BORDERS

Borders Style Borders Thickness Borders Color

Border radius top left Border radius top right Border radius bottom left Border radius bottom right

OTHER OPTIONS

☐ Shadows Shadows Size

Background color

CANCEL SAVE

Fig. 5.208: Widget style tab.

The third tab allows overriding the common css properties for the whole cockpit. The editor will highlight possible syntax errors.

5.6.3 Data configuration

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

5.6.3.1 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.

5.6.3.1.1 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 below. You can type it manually or use the look up just aside the parameter line.

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

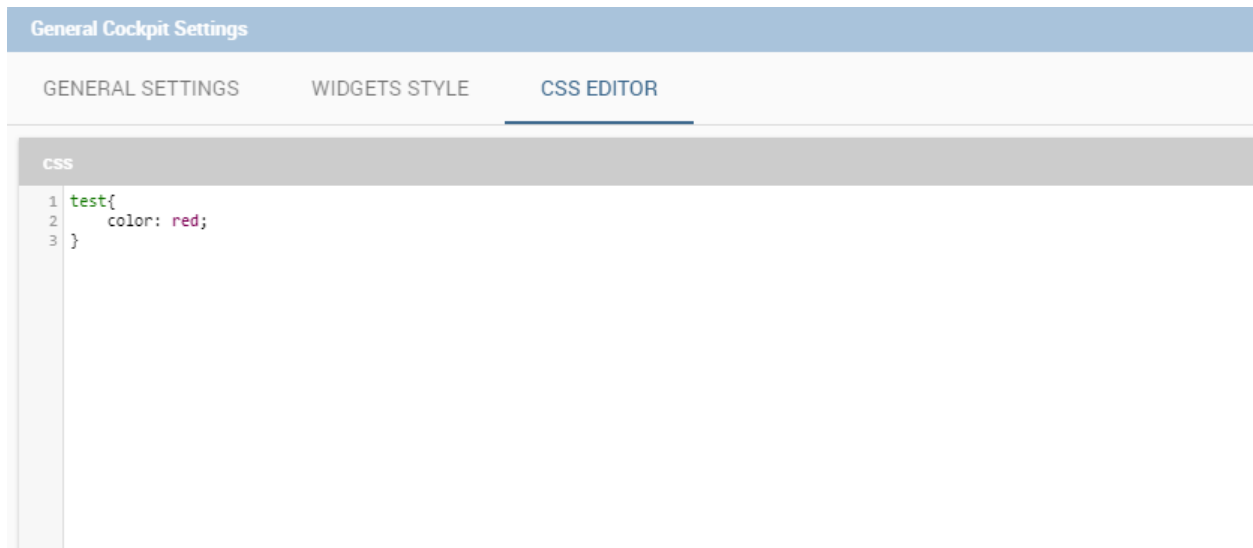


Fig. 5.209: Css editor tab.

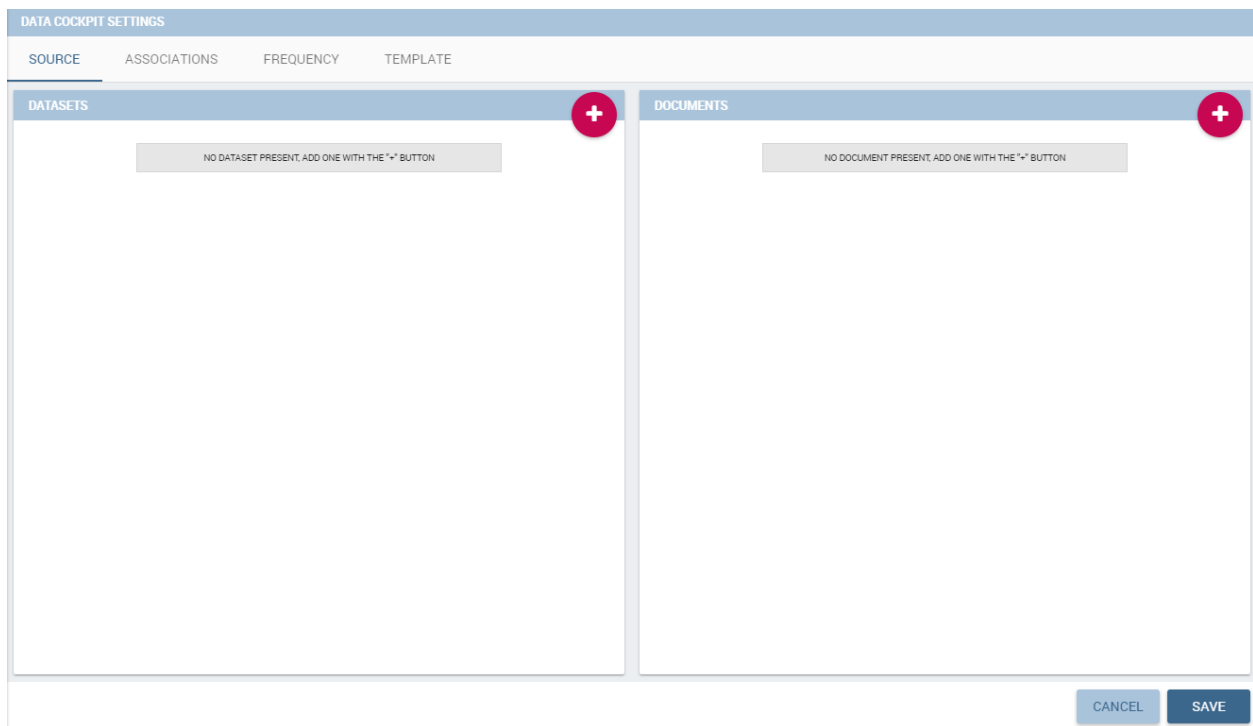


Fig. 5.210: Data configuration window.

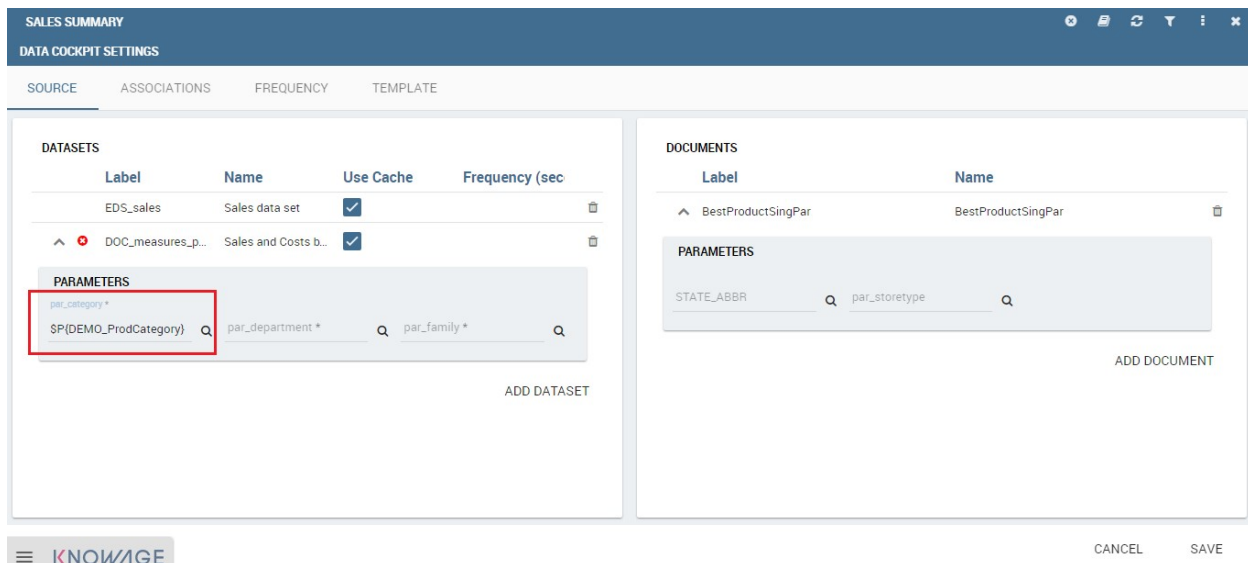


Fig. 5.211: Dataset management.

widgets. In the occurrence of Associations parametric documents, parameter boxes are shown below. Note that it is mandatory to link them to analytical drivers (previously hooked to the document) or be assigned a fixed (default) value.

5.6.3.2 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.

The following figure 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 rely on multiple columns the user must add them one by one.

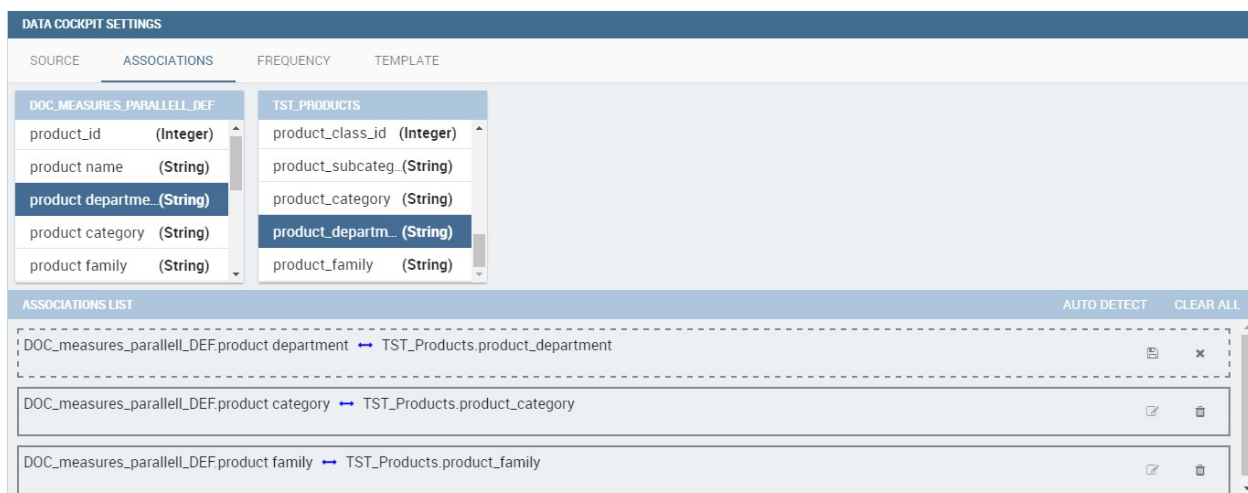


Fig. 5.212: Associations between dataset columns.

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

The screenshot shows the 'DATA COCKPIT SETTINGS' window with the 'ASSOCIATIONS' tab selected. It displays two lists of parameters: 'DOC_MEASURES_PARALLEL_DEF' and 'TST_PRODUCTS'. The 'ASSOCIATIONS LIST' at the bottom shows three associations: 'DOC_measures_parallel_DEF.SP{par_family}' linked to 'TST_Products.product_family', 'DOC_measures_parallel_DEF.SP{par_category}' linked to 'TST_Products.product_category', and 'DOC_measures_parallel_DEF.SP{par_department}' linked to 'TST_Products.product_department'.

Fig. 5.213: Associations between dataset column and dataset parameter.

Another example is supplied in Figure below. Here the association is performed between a dataset Frequency column and document parameter.

The screenshot shows the 'PRODUCT ANALYSIS' window with the 'DATA COCKPIT SETTINGS' tab selected. It displays two lists of parameters: 'DS_BASE_FOR_GIS' and 'MAP_PAR'. The 'ASSOCIATIONS LIST' at the bottom shows one association: 'MAP_PAR.SP{par_family}' linked to 'DS_BASE_FOR_GIS.product_family'.

Fig. 5.214: 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 on data update.

5.6.3.3 Indexes

If you want that cockpit loading be faster, you can create indexes on cached data. This feature is available only for cached dataset.

If you want to create an index on a column you have to choose that column by clicking. The name of the column will appears in the Indexes List section of the page. If you want to confirm your choose, click on the save icon. If you want

to cancel it, click on the cross icon. After saving an index you'll see in the list surrounded by a continuous border.

Data Cockpit Settings

SOURCE ASSOCIATIONS **INDEXES** FREQUENCY TEMPLATE

ALL_CUSTOMERS		SPARKSQL_STORE	
customer_id	(Integer)	store_id	(Long)
account_num	(Long)	store_type	(String)
lname	(String)	region_id	(Long)
fname	(String)	sales_region_id	(Long)
mi	(String)	store_name	(String)
address1	(String)	store_number	(Long)
address2	(String)	store_street_address	(String)
address3	(String)	store_city	(String)

Indexes list ADD INDEXES ON ASSOCIATIONS CLEAR ALL

SPARKSQL_STORE.store_name		
ALL_CUSTOMERS.customer_id		

CANCEL SAVE

Fig. 5.215: Indexes settings example

For example, in the figure above index on the column “customer_id” of ALL_CUSTOMERS dataset is already saved. “store_name” column of “SPARKSQL_STORE” dataset is selected. If you want to create an index on it, you have to save it.

5.6.3.4 Frequency

The Frequency tab defines a scheduling over dataset involved in the associations. An example is supplied in the next figure. This means that associations are activated automatically and data are reloaded according to this feature. In particular, groups of realtime datasets that compose one or more associations can have different update frequencies. We stress that, in order to secure the right document execution, the group frequency do not affect the other ones and each group is reloaded at different times. In addition, realtime dataset that are not involved in any association can have their own frequency.

DATA COCKPIT SETTINGS

SOURCE ASSOCIATIONS **FREQUENCY** TEMPLATE

DOC_measures_parallel DEF

TST_Products

Frequency

5

Fig. 5.216: Frequency settings example.

5.6.3.5 Variables

In this tab the user can define the variables that will be available inside the cockpit.

Every variable needs a name that will be used to call it and get the value, a type and a value. The available types are the following:

Data Cockpit Settings		
SOURCE	ASSOCIATIONS	INDEXES
FREQUENCY	VARIABLES	TEMPLATE
Variables definition		
Variable Name	Variable Type	Value
testVariable	Static	100
Variable Name	Variable Type	Attribute
DynamicValue	Profile	Country

Fig. 5.217: Variables tab

- Static: a static number or string.
- Dataset: the value of a selected dataset column. If a column is not selected a set of variables key/value will be created using the first two columns of the selected dataset.
- Profile: a set of profile attributes available. (ie. the username)
- Driver: the value of a selected analytical driver.

The variables will be available inside the widgets with the `$V{variablename}` format.

5.6.3.6 Template

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

5.6.4 Selections

Adding the **Selections** to your widgets, namely the possibility to reload all widget data according to selection made through the click on a specific item of the cockpit (cell value, chart bar, etc.). Moreover, thanks to this functionality the user can reproduce the drill down feature that we introduced in Chapter of Chart. You can check which selections are active on your cockpit at anytime thanks to the **Selection** functionality. In Section 7.1 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 below. The “Delete” button is available just aside each row to let the user to remove that specific selections. Click on the “Cancel” button to exit the window.

5.6.5 Clear cache

The **Clear cache** button lets you realign 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.

5.6.6 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



Fig. 5.218: Template example.

SELECTIONS LIST			
Dataset	Column Name	Values	
ds__5305638	Product	Sugar	
ds__2560679	City	Turin	
			<div>CANCEL</div> <div>SAVE</div>

Fig. 5.219: Selection window.

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.

5.6.7 Multisheet functionality

Cockpit allows to manage data visualization splitting it in two or more sheets. In each layer the user can find and employ the features shown above. Indeed, it is possible to perform a new analysis (as highlighted in Figure below) selecting different datasets and gadgets.

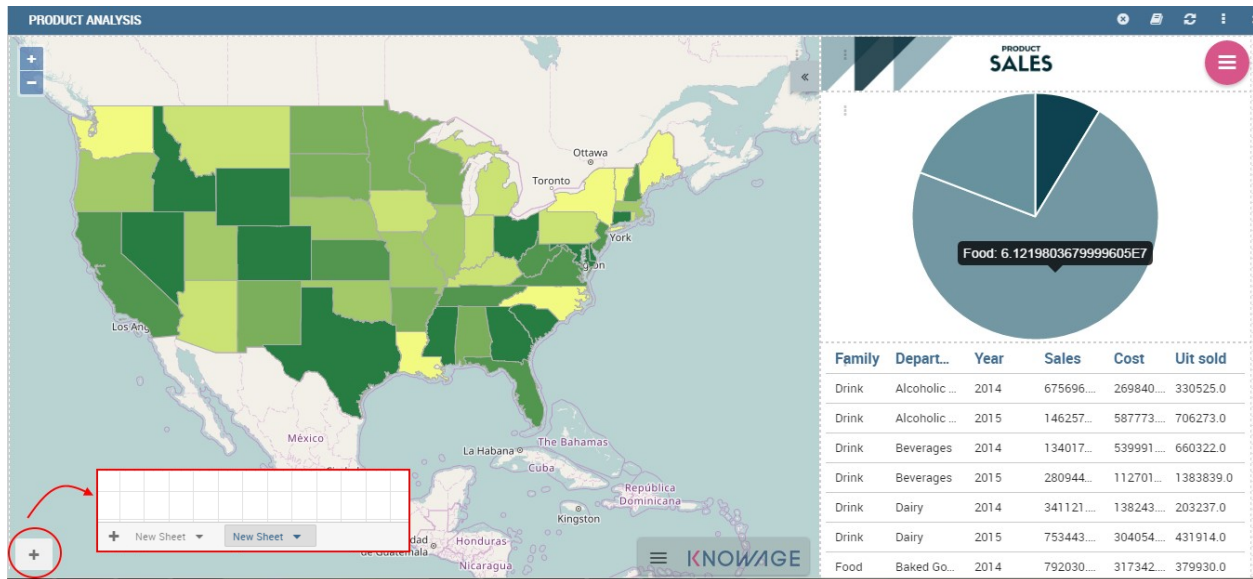


Fig. 5.220: Multisheet cockpit example.

A user can take advantage of the “move widget” functionality we saw in My first Cockpit to bring widget from one sheet to another.

Furthermore it is possible, but not mandatory, to set associations between datasets underlying different sheets. The multisheet functionality is particularly useful to focus the analysis in a single spot and have a general overview over it in few clicks at the same time.

5.6.8 Export cockpit

Cockpit document allows to export data into csv file without executing document. This is very useful when you produce data using a heavy query. This option is available if your document has parameters. When you start execution of your document, you will get opened filter panel so you can fill values. To start export, you should click on drop down menu, next to execute button, as on image below.

After process is finished, you will get notification on **download** icon  and you can find your file in **Download manager** section.

Clicking on download icon, **Download manager** will open, and you will be able to download zip file that contains csv file/files, depends of how many widgets (chart or table) you have in your document.

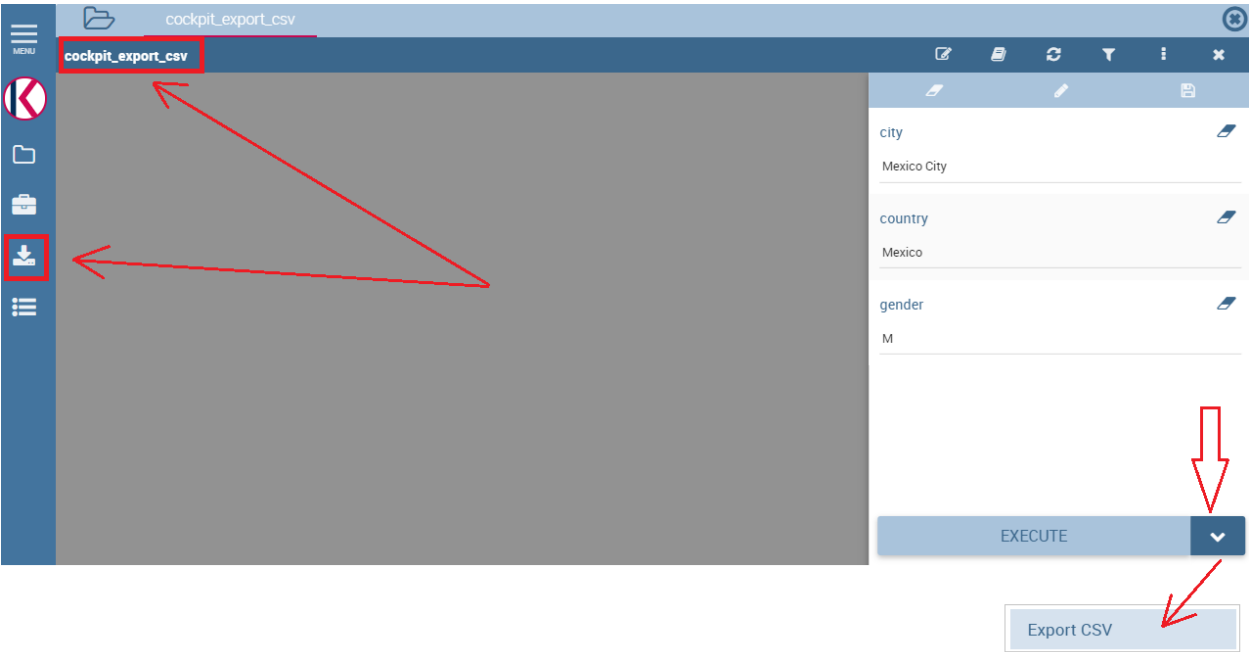


Fig. 5.221: Export cockpit into csv.

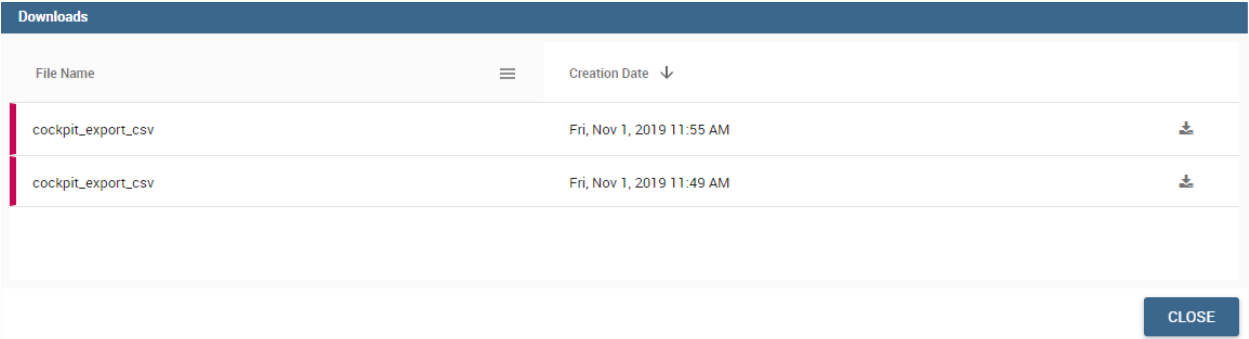


Fig. 5.222: Download manager section.

5.7 Free Inquiry

This detailed user guide is dedicated to the Qbe (acronym of Query By Example), a Free Inquiry instrument which empowers users with easy and free access to information via graphical interfaces.

Free Inquiry indicates the modus operandi of analysts and operational users that are usually seeking for business analysis that are not limited to pre-arranged lists of results. This method has a medium level of difficulty since it requires an adequate knowledge of data management and a structured organization of work.

QbE is the tool that lets you develop your free inquiry through an entirely graphical modality. Moreover, you can execute the query, check the results, export them and save the query for further use.

The material will be divided in two main sections. The first is dedicated to build queries in the Knowage Server environment, supposing that an expert user has already created a suitable business model to analyse. In the second part, we will provide the user for the principal steps to build a proper business model through the Qbe designer available in Knowage Meta.

5.7.1 My first Query By Example

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.

Let's suppose that an administrator has built a business model and, consequently, released it on Knowage Server. This permits the user to access the model, query the available entities and save results as a dataset, usable later in other Knowage documents, such as cockpits.

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

5.7.1.1 Query design and execution

To open the QbE editor, access the **Models** section, available in the end user's **Workspace**. Then, simply click on the model icon to reach the QbE graphical interface.

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

As shown in Figure 9.1 the window of the QbE editor contains the **Query designer**. In next sections we explain in detail all the areas of the **Query Designer**, the **Datamart Schema** tab, the query editor and a hidden tab dedicated to the management of queries, subqueries and parameters catalogue.

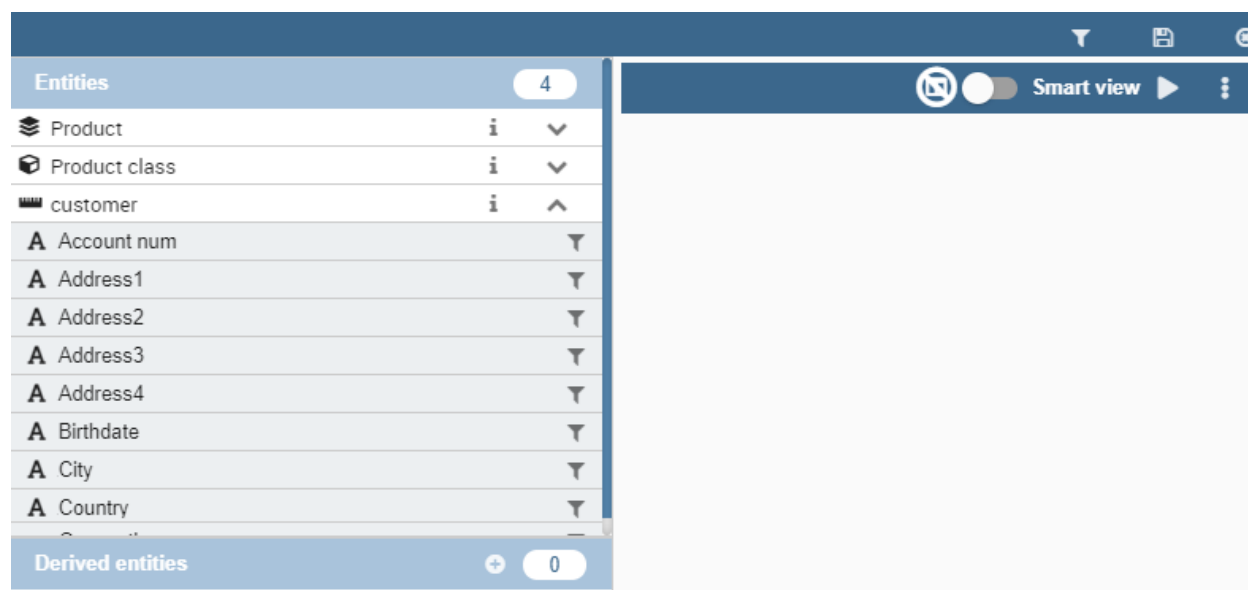


Fig. 5.223: QbE editor.

5.7.1.1.1 Datamart Schema

Starting from the left side: The upper Panel shows the searchable logical schema and the list of entities that can be queried to generate the query. Entities are represented in a tree structure, with user-defined names. Field can be added in the query (right area) clicking on it. The lower Panel shows list of created subqueries in a tree structure where children are fields of subquery

There are few types of entities: *facts*, represented by a cube symbol.(i.e., the Sales entity), *dimensions*, represented by a four-arrows symbol (i.e., the Product entity), *geographical dimension*, represented by *earth* icon.

Each single entity is composed of a title, some attributes or measures and relationships with other entities. Relations are available clicking on *i* icon of one entity. In particular, by exploding the content of an entity (i.e. Sales as in figure above), you may encounter the following elements:

- **measure:** it refers to fields associated with numeric data (e.g. UNIT SALES);
- **attribute:** it refers to fields that can be associated to a category (e.g. PRODUCT ID);
- **relation:** it refers to relationships or connections between two entities (e.g. relationship between the sales and the product dimension).

There are two available views: smart and advance. When qbe is opened by default user will see smart view. Clicking on one field of one entity that field will be added in the query and user will see executed query in immediately.

User can switch to advance view clicking on Smart view in top right corner. User can continue adding fields in the query but without seeing result.

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

5.7.1.1.2 Calculated fields management

You can create new calculated fields inside a query. In advance view there will be available option for creating **calculated fields**. Clicking on **Calculator** icon **Calculated field wizard** will open.

To build a calculated field, you shall define:

Sales				
Relation name	Source fields	Target entity	Target fields	Target fields
time	TIME_ID	Time	TIME_ID	INNER
product	PRODUCT_ID	Product	PRODUCT_ID	INNER
store	STORE_ID	Store	STORE_ID	INNER
customer	CUSTOMER_ID	Customer	CUSTOMER_ID	INNER
promo	PROMOTION_ID	Promotion	PROMOTION_ID	INNER

OK

Fig. 5.224: Relations of one entity.

Entities4

Producti

Product classi

customeri

Account num

Address1

Address2

Address3

Address4

Birthdate

City

Country

Derived entities+0

Smart view

Country

Canada

Mexico

USA

Fig. 5.225: Smart view.

Entities4

Producti

Product classi

customeri

Account num

Address1

Address2

Address3

Address4

Birthdate

City

Country

Derived entities+0

Smart view

Entity	Field	Alias	Group	Order	Aggregation	Show field
customer	Country	Country	<input checked="" type="checkbox"/>	NONE		<input checked="" type="checkbox"/>
customer	City	City	<input checked="" type="checkbox"/>	NONE		<input checked="" type="checkbox"/>

Fig. 5.226: Advance view

- **Name;**
- **Type:** string, number or date;
- **Nature:** measure or attribute;
- **Formula:** you can click on the fields included on the left and build the formula.

An example is provided below.

Fig. 5.227: Calculated field wizard.

5.7.1.1.3 Query Editor

Central panel provides a query editor. As we already mentioned, there are two available views: smart and advance.

Elements from the datamart schema on the left can be added into the query editor. To remove a field from the query editor, just click on the x icon, or if advance view is opened, clicking on three dots icon, **Delete field** is available.

5.7.1.1.4 Selected Fields

Central panel contains the list of columns to be returned by the query. To add a new field in this section, just click on a field in the schema panel tree.

This panel is structured as a table: columns contain the fields selected from the datamart schema. Every column has **gear icon** on the top (smart view), that is providing applicable functions as shown below, plus icon for deleting field from query and for ordering.

In the advance view, every row include applicable functions as shown below, plus clicking on **three dots icon** there are few more functions.

With available functions it is possible to:

- **Alias:** define aliases for fields: those aliases are shown as column headers in the result table; it is editable
- **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;

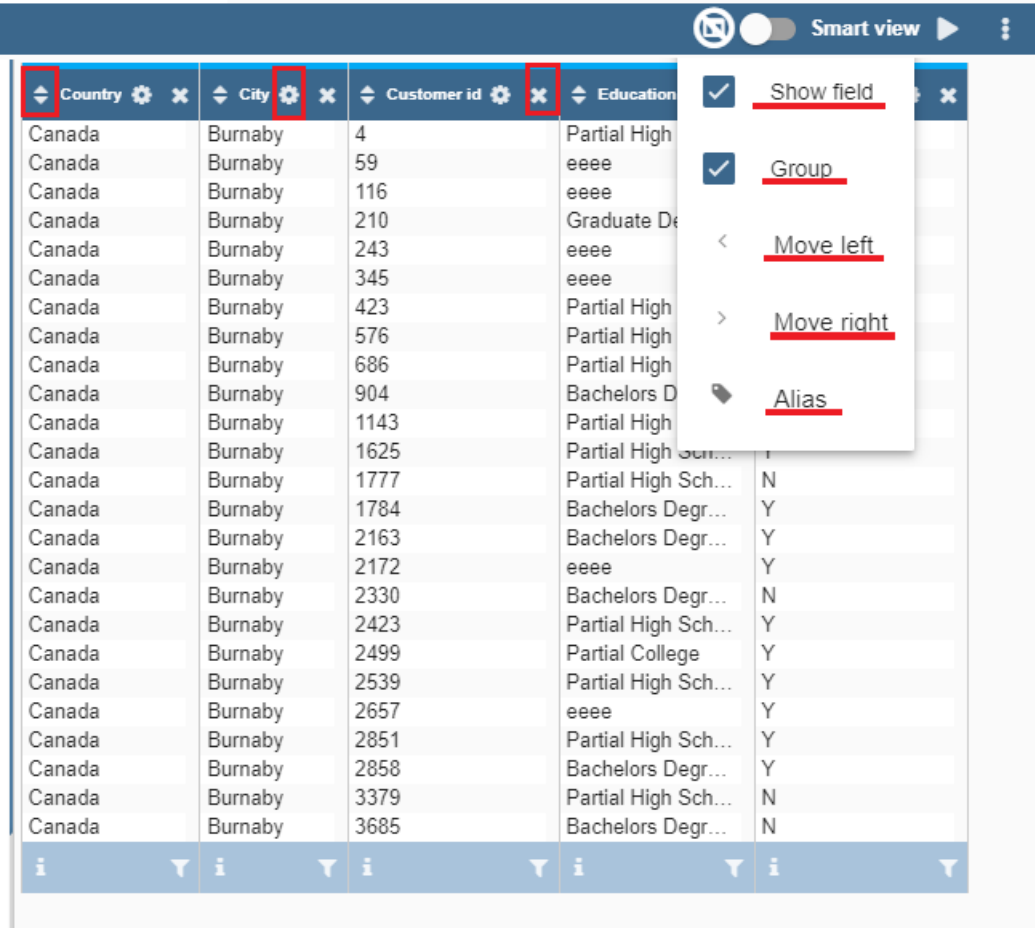


Fig. 5.228: Select fields interface in smart view.

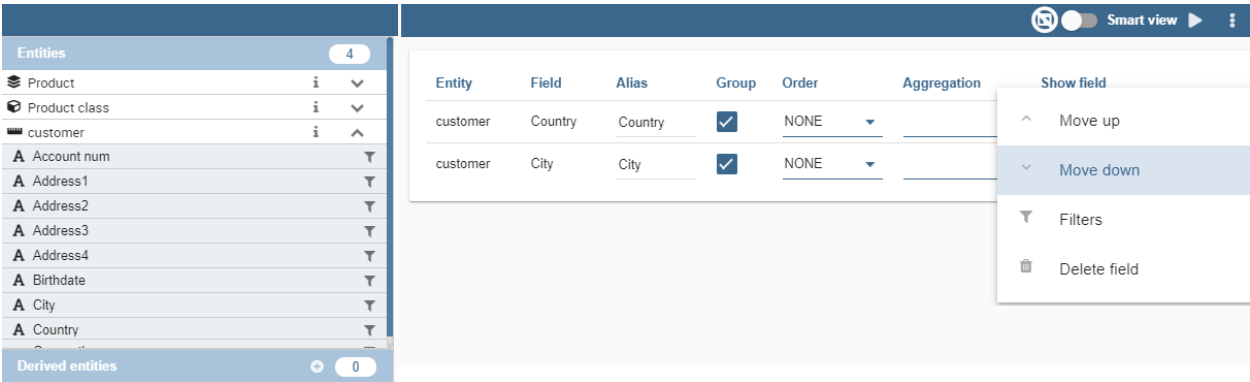


Fig. 5.229: Select fields interface in advance view.

- **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);
- **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;

It is possible to edit alias, clicking on gear icon (smart view) and on **Alias item**. In advance view, alias can be changed clicking on cell of alias column.

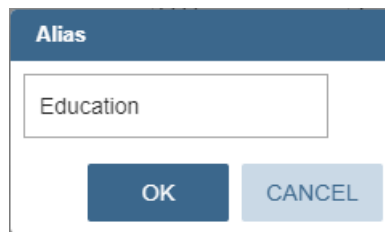


Fig. 5.230: Change alias.

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 added in the query editor, without an aggregation function defined, otherwise you will get an SQL exception. The possible grouping functions are shown in the following figure.

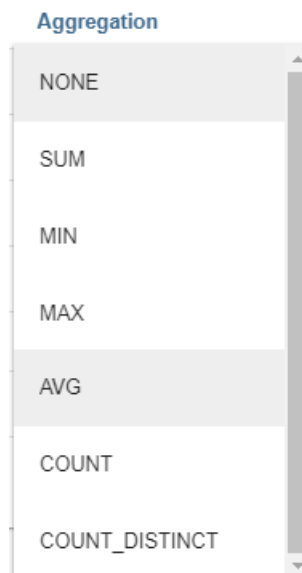


Fig. 5.231: Aggregation functions.

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

Moreover, multiple relationships may occur among entities. A typical example concerns dates. Suppose you have two relationships between the **Order** fact table and the **Time** dimension table: the first links the *order_date* column of the first table to the *time_id* column of the latter, while the second relationship joins the *shipping_date* column to the *time_id* column.

In this case, when dragging fields from both the **Order** entity and the **Time** entity you may want to specify which relationship will join the two tables: for instance, you may want to know the total number of orders according to the ordering month, the shipping month or for both. In all these situations, you can set the relationship to be used by clicking the **Relationships wizard** button at the top right corner of the panel. A pop up window opens where you can define the path to be used. Please refer to Multiple relationships section for all details regarding the disambiguation of relationships.

The toolbar about query editor sub-section has a toolbar contains additional functionalities summarized in Table below.

Table 5.9: Select fields toolbar options

Button	Description
Join definitions	
SQL	Shows SQL generated by the graphical interface
Discard Repetitions	Remove duplicated rows from results, if any
P	Add parameters
Calculator	Add calculated fields
Three gears	Open advanced filters panel
Eye	Show/hide hidden fields
Smart View	Switch between smart and advance view
Play	Preview query
Three dots	Option to choose between Deleting all fields from query and Export query into csv/xls/xlsx

5.7.1.1.5 Filters

The **Filters** panel allows you to define filter criteria (WHERE clause). Filters are structured as a table: here rows contain filters, while columns represent the elements of the filter. Filters panel can be opened in three ways: - In smart view clicking on **Filter icon** on the field in entity - In smart view clicking on **filter icon** on the already added field in the query - In advance view clicking on three dots and **Filters item**

Adding new filter is possible clicking on + icon.

Removing the filter is possible clicking on **eraser** icon.



Filters are expressions of type:

Left operand + Operator + Right operand.

Structure of Filters panel is:

- the **Field**, **Condition**, **Target** columns allow you to define filters according to the syntax defined above.
- the **Target type** column define the types of right operand: manual, value of the field, another entity, parameter, subquery;

With target type **manual** you should fill input **target** with value that you want to be right operand.

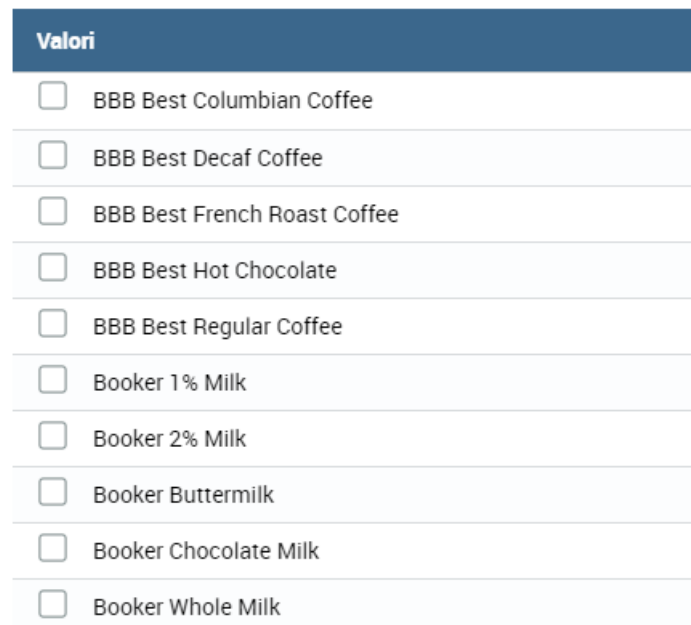
With target type **value of the field** lookup function is activated to facilitate selection of values. You are able to choose values for right operand. If you are choosing two values, you should set condition to be **between**, **not between**, **in** or **not in**. If you are choosing more than two values, you should set condition to be **in** or **not in**.



Filters

Field	Condition	Target type	Target
Customer : COUNTRY	Equals to	Manual set value	Mexico manual

Fig. 5.232: Manual target type



Valori

- ☐ BBB Best Colombian Coffee
- ☐ BBB Best Decaf Coffee
- ☐ BBB Best French Roast Coffee
- ☐ BBB Best Hot Chocolate
- ☐ BBB Best Regular Coffee
- ☐ Booker 1% Milk
- ☐ Booker 2% Milk
- ☐ Booker Buttermilk
- ☐ Booker Chocolate Milk
- ☐ Booker Whole Milk

1 to 10 of 146 |< < Page 1 of 15 > >|

Fig. 5.233: Filter lookup for right operand selection.

With target type **another entity** you will get option to choose field from another entity for your right operand.

The screenshot shows a 'Filters' interface with a table-like structure. The first row has columns: 'Field' (with value 'Customer : CITY'), 'Condition' (with value 'Equals to'), 'Target type' (with value 'Another entity'), and 'Target'. A dropdown menu is open for 'Target type', showing a list of fields from the 'STORE' entity: 'REGION ID', 'STORE NAME', 'STORE NUMBER', 'STORE CITY' (which is highlighted in blue), and 'STORE STATE'. To the right of this dropdown, another dropdown menu is open, showing a list of entity names: 'CUSTOMER', 'SALES', 'PRODUCT', 'STORE' (which is highlighted in grey), and 'PRODUCT CLASS'. A red '+' button is visible in the top right corner of the filters area.

About target type **subquery** and **parameter** there will be more words later.

Important: Enterprise Edition only

Filtering data with fields type of date/time/timestamp using calendar/time/calendar is available only for Enterprise Edition.

If you have SI license file, you will get the chance to filter your data with fields type of date/time/timestamp using calendar/time/calendar + time option. This depends of what is data type of you field, and this is coming form metamodel creation phase. When creating your metamodel, you can set data type of to your field.

Note that more complex combinations of filters can be defined using the Advance filter wizard, which you ca find selecting the **Three gears** icon.

In the following table the possible types of filters in the QbE are summarized. The use of subqueries in filters is explained later in *Advanced QbE functionalities* paragraph.

Table 5.10: Possible combinations of filters in the QbE.

Filter type	Left operand	Operator	Right operand	Example
Basic	Entity.attr ibute	Any	value	Prod.family = 'Food'
Basic	Entity.attr ibute	Any	Entity.attr ibute	Sales.sales > Sales.cost
Parametric	Entity.attr ibute	Any	[parameter]	Prod.family = [p_family]
Dynamic	Entity.attr ibute	Any	prompt	Prod.family = ?
Value list from subquery	Entity.attr ibute	In /not in	subquery	Sales.custo mer in subquery
Single value from subquery	subquery	< = >	value	Subquery > 0

5.7.1.1.6 Query Preview

While you are in smart view you can see preview of you query. While you are in advance view, and you are satisfied with your query or if you want to check the results, you can see the returned data by clicking the **Play** button located

Attribute's Detail

Name

THE DATE

Description

Structural

Aggregation Type

COUNT

The preferred aggregation type for the give column (COUNT, SUM, AVG, MAX, MIN)

Roles NOT enabled

Roles NOT enabled to view this column

Physical table

TIME

The original physical table of this column

Profile Attribute Filter Type

EQUALS TO

The type of filter to use with profile attributes

Data Type

TIMESTAMP

The data type of the given column (

Profile attribute

A profile attribute used to filter

Physical column

city

The original physical column of this

Custom function

Custom DB function to apply to column

DOUBLE

DATE

TIMESTAMP

TIME

NUMBER

DELETE

CANCEL

SAVE

Fig. 5.234: Metamodel creation.

Filters

Field

Time : THE DATE

Condition

Equals to

Target type

Manual set value

11/01/2019

Filters

Field

Time : THE DATE

Condition

Between

Target type

Manual set value

11/01/2019

11/01/2019

Fig. 5.235: Filters creation on date data type of the field.

The figure shows two screenshots of the 'Filters' creation interface for a time data type field. Both screenshots have a blue header bar with the text 'Filters' and a red circular button with a white plus sign on the right.

The top screenshot shows a single filter rule. It has three columns: 'Field', 'Condition', and 'Target type'. The 'Field' column contains 'Time : THE DATE'. The 'Condition' column has a dropdown menu with 'Equals to' selected. The 'Target type' column has a dropdown menu with 'Manual set value' selected. To the right of these columns are two time input fields, each with up and down arrows, containing '10' and '39' respectively, followed by a small trash icon.

The bottom screenshot shows the same interface but with two filter rules. The first rule is identical to the one in the top screenshot. The second rule is added below it, also with 'Time : THE DATE' as the field, 'Between' as the condition, and 'Manual set value' as the target type. It has two time input fields, each with up and down arrows, containing '10' and '39' respectively, followed by a small trash icon.

Fig. 5.236: Filters creation on time data type of the field.

The figure shows two screenshots of the 'Metamodel' creation interface for a timestamp data type field. Both screenshots have a blue header bar with the text 'Filters' and a red circular button with a white plus sign on the right.

The top screenshot shows a single filter rule. It has three columns: 'Field', 'Condition', and 'Target type'. The 'Field' column contains 'Time : THE DATE'. The 'Condition' column has a dropdown menu with 'Equals to' selected. The 'Target type' column has a dropdown menu with 'Manual set value' selected. To the right of these columns is a date input field with a calendar icon and the text '11/01/2019', followed by two time input fields, each with up and down arrows, containing '10' and '38' respectively, followed by a small trash icon.

The bottom screenshot shows the same interface but with two filter rules. The first rule is identical to the one in the top screenshot. The second rule is added below it, also with 'Time : THE DATE' as the field, 'Between' as the condition, and 'Manual set value' as the target type. It has a date input field with a calendar icon and the text '11/01/2019', followed by two time input fields, each with up and down arrows, containing '10' and '39' respectively, followed by a small trash icon.

Fig. 5.237: Metamodel creation, timestamp data type of the field.

in the top right corner of the panel. From there, you can go back to the **Designer** 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 details, split in three tabs:

- **Generic**, in this tab you set basic information for your dataset like its **Label**, **Name**, **Description** and **Scope**.
- **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.

5.7.1.2 Advanced QbE functionalities

In this section we focus on advanced features, which can be comfortably managed by more expert users.


5.7.1.2.1 Spatial fields usage

Important: Enterprise Edition only

Spatial dimension is available only for Enterprise Edition with LI licence.

The Qbe engine supports spatial queries through a set of operators (that return true or false) or a set of functions (these usually return a measure). This feature is although available only when the Location Intelligence (LI) license is possessed and when data are stored in Oracle 12c database. It also fundamental that the Business Model has to be tagged as geographical model. You can refer to Meta Web Section to have details on how to set the geographical option using Knowage Meta.

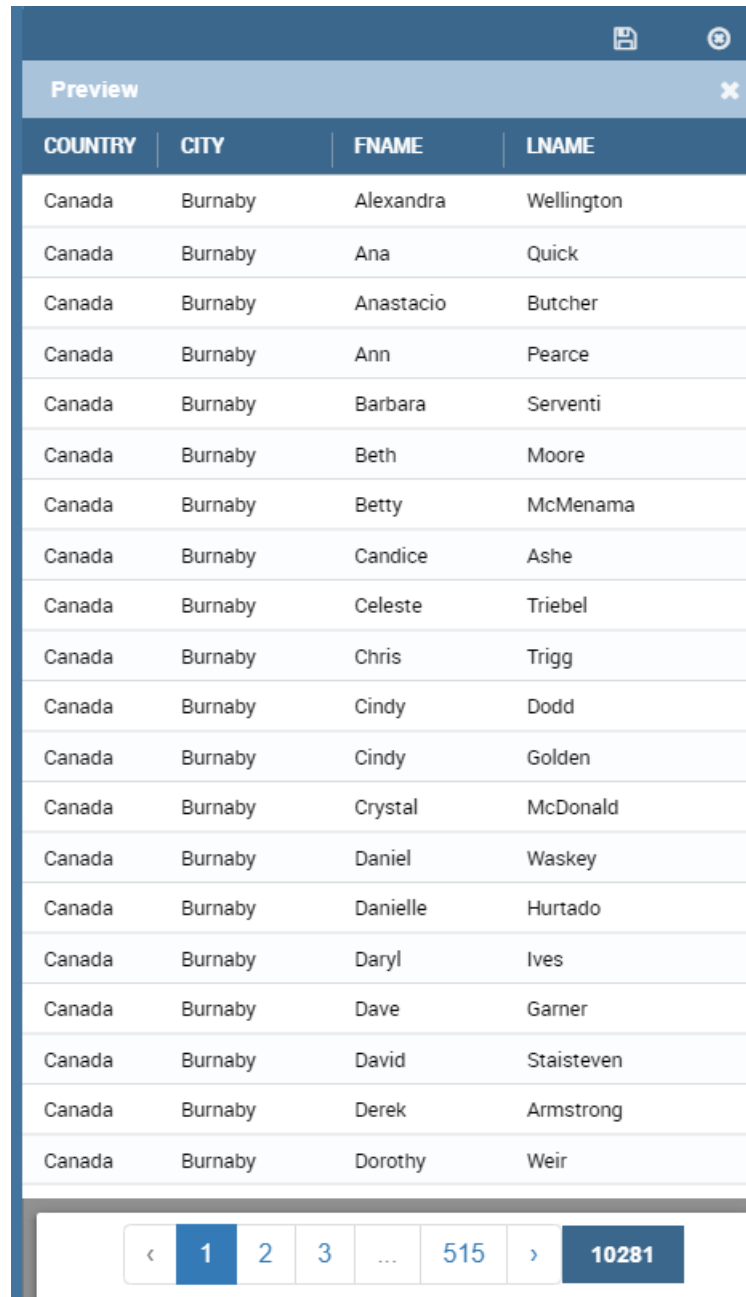
We suppose that we have a BM with geographical dimensions enabled (by a technical user). In this case the dimensions

which has spatial fields are marked with the compass icon . Once the spatial dimension is expanded the fields are listed. Here there is no tracking symbol to distinguish between geographical attributes and the “normal” one. Therefore it is very important that the user is previously informed of which fields has geometrical properties.

After a first selection of fields, it is possible to add calculated fields. Click on the **Calculator** option available on the query editor area as shown by the blue arrow in figure below. Note that a wizard opens: you can use this editor to insert a new field obtained through a finite sequence of operation on the selected fields. The circles of the next figure underline that the fields on which you can operate are the one previously selected by a simple click on the field.

In addition note that the **Items** panel provides all the applicable functions sorted by categories:

- aggregation functions,
- string functions
- time functions,



The image shows a 'Preview' window with a table of employee data. The window has a dark blue header bar with a save icon and a close button. The table has four columns: COUNTRY, CITY, FNAME, and LNAME. The data is as follows:

COUNTRY	CITY	FNAME	LNAME
Canada	Burnaby	Alexandra	Wellington
Canada	Burnaby	Ana	Quick
Canada	Burnaby	Anastacio	Butcher
Canada	Burnaby	Ann	Pearce
Canada	Burnaby	Barbara	Serventi
Canada	Burnaby	Beth	Moore
Canada	Burnaby	Betty	McMenama
Canada	Burnaby	Candice	Ashe
Canada	Burnaby	Celeste	Triebel
Canada	Burnaby	Chris	Trigg
Canada	Burnaby	Cindy	Dodd
Canada	Burnaby	Cindy	Golden
Canada	Burnaby	Crystal	McDonald
Canada	Burnaby	Daniel	Waskey
Canada	Burnaby	Danielle	Hurtado
Canada	Burnaby	Daryl	Ives
Canada	Burnaby	Dave	Garner
Canada	Burnaby	David	Staisteven
Canada	Burnaby	Derek	Armstrong
Canada	Burnaby	Dorothy	Weir

At the bottom of the window is a pagination bar with the following elements: a left arrow, a box containing '1' (highlighted), a box containing '2', a box containing '3', an ellipsis '...', a box containing '515', a right arrow, and a box containing '10281'.

Fig. 5.238: Preview wizard.

Saving Qbe Dataset

DETAIL PERSISTENCE METADATA

Label *

Name *

Description 0 / 150

Exportable on HDFS: ☐

Scope *

CANCEL SAVE

Fig. 5.239: Save qbe dataset.

Entity	Field	Alias	Group	Order	Aggregation	Show field
Aereoporto Arrivi	Codice IATA	Codice IATA	<input type="checkbox"/>	NONE	COUNT	<input checked="" type="checkbox"/>

Entities 14

- Partenze
- Arrivi
- Compagnia
- Tipologia Aeromobile
- Flight status ACDM
- Tempo giornaliero
- Tempo orario
- Aereoporto Arrivi
- c Inizio validità
- c Fine validità
- c Codice ICAO
- c Codice IATA

Fig. 5.240: QbE spatial dimensions.

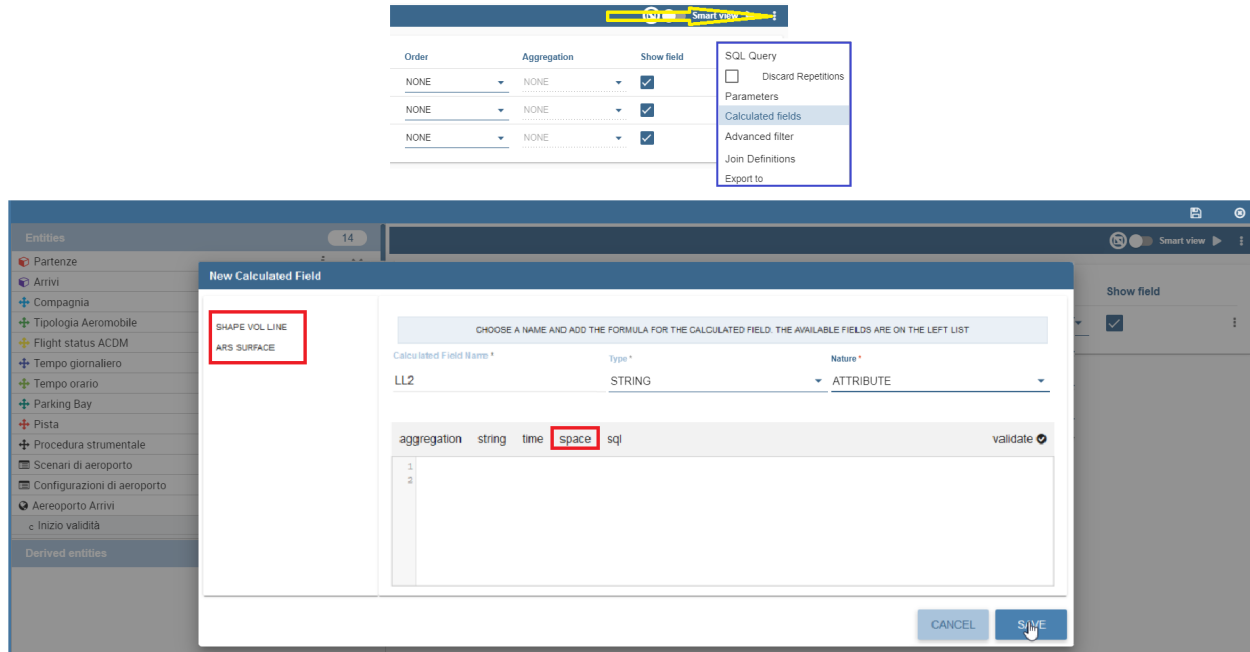


Fig. 5.241: Calculated field wizard with spatial filters.

- spatial functions,
- sql functions,
- custom function (if they are registered).

Warning: Take into account the Oracle function definition

It is important to refer to Oracle Documentation to know the arguments, in terms of type and number, of each function to assure the right functioning and do not occur in errors while running the Qbe document.

The latter are available only in the presence of a geographical Business Model and *must* be properly applied to spatial attributes or measures. Figure below shows the list of the available spatial functions while next table helps you to use them properly, supplying the corresponding Oracle function name and a link to grab more specific information about usage, number of arguments, type and output.

Table 5.11: Link to Oracle spatial functions.

Function Name	Oracle Function
distance	SDO_GEOM.SDO_DISTANCE
dimension	GET_DIMS
centroid	SDO_GEOM.SDO_CENTROID
geometrytype	GET_GTYPE
length_spa	SDO_GEOM.SDO_LENGTH
relate	SDO_GEOM.RELATE
intersection	SDO_GEOM.INTERSECTION

To apply one function click on the function name and the “Operands selection window” wizard opens. Figure below shows an example for the function “Distance”. Fill in all boxes since all fields are mandatory.

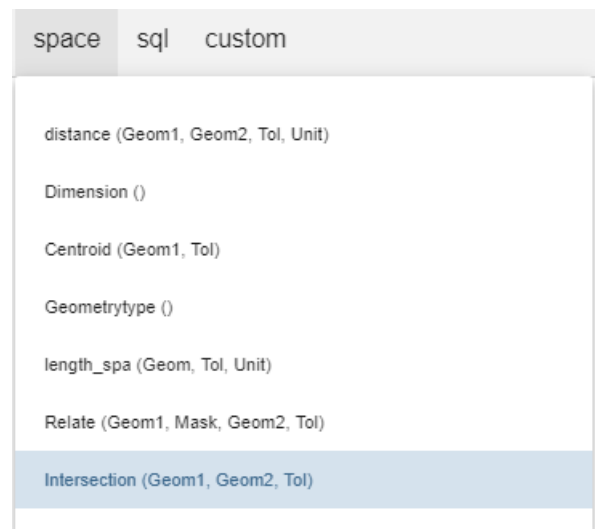


Fig. 5.242: Spatial function list.

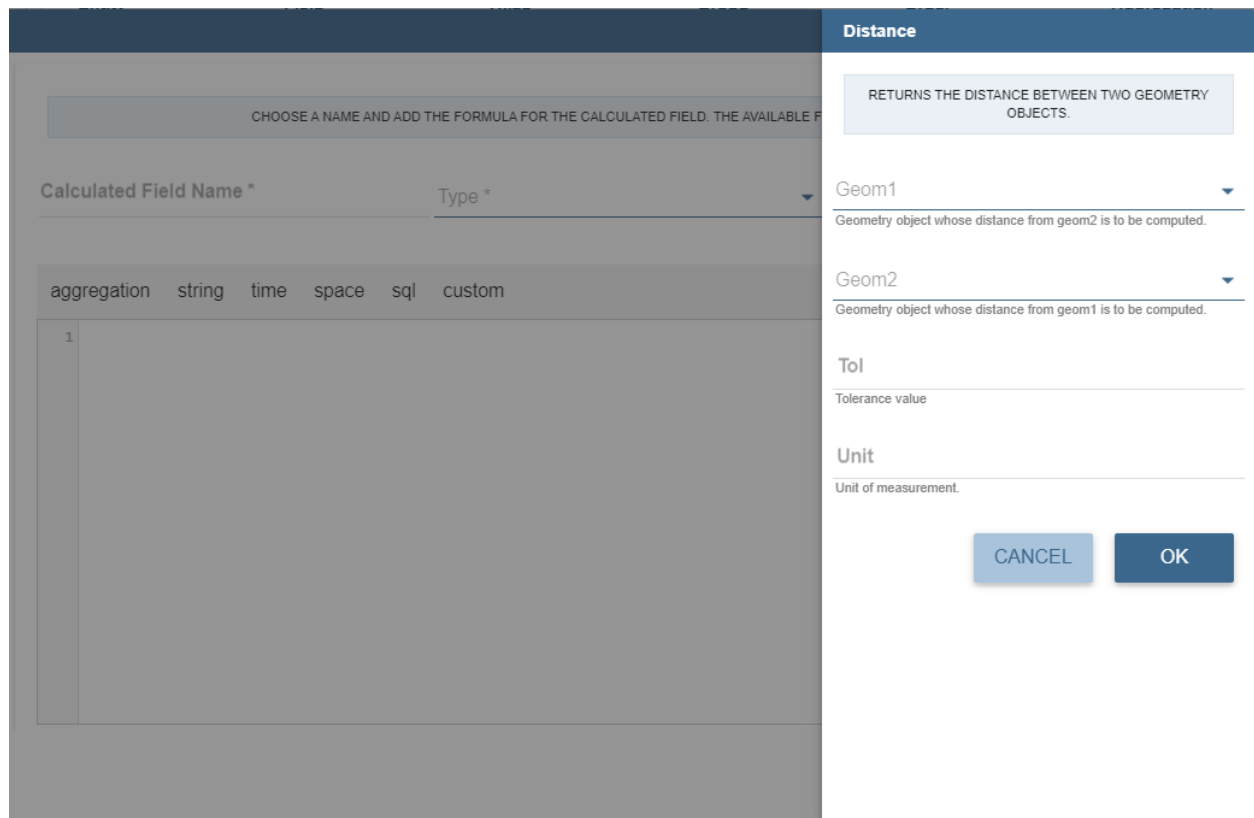


Fig. 5.243: Operands selection window.

Finally you can use spatial function to add a calculated field, as shown below.

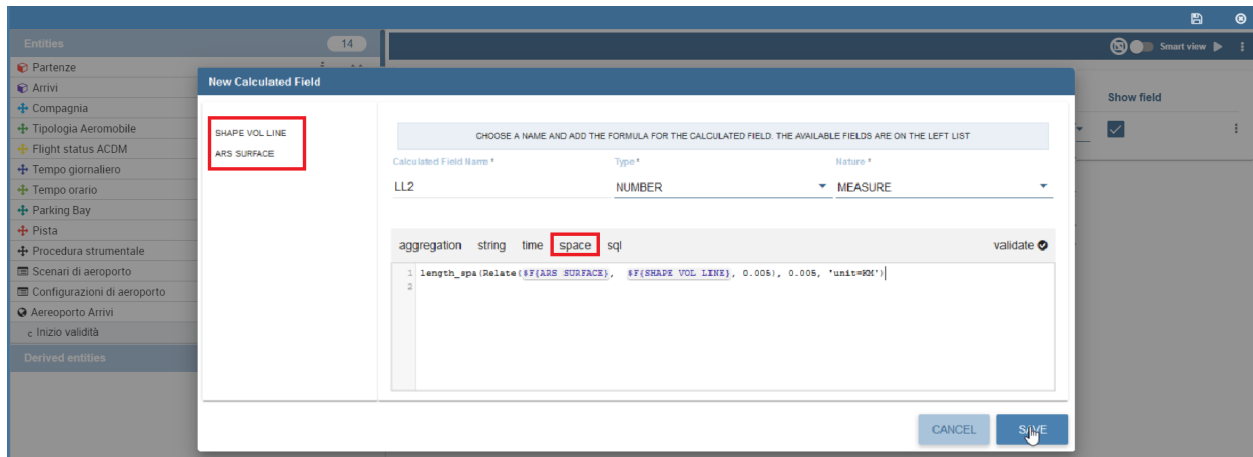


Fig. 5.244: Example of added calculated field using a spatial function.

As well as calculated fields it is possible to filter on spatial fields using specific geometric operators. Once again we report in Figure below the available geometric operator (you can find them scrolling the panel to the bottom) and report the link to the Oracle web pages in the next table.

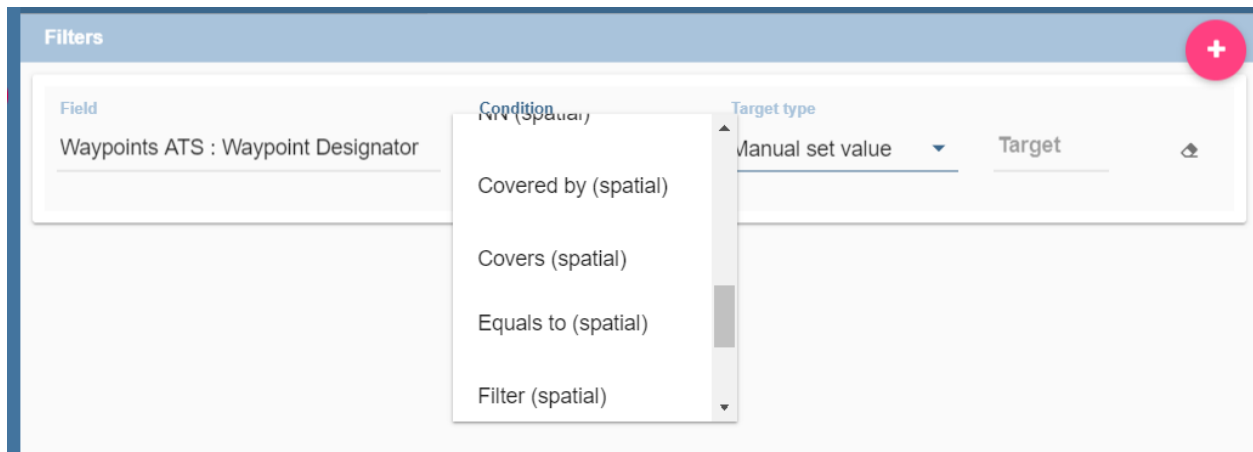


Fig. 5.245: Spatial filters.

See the table below:

Table 5.12: Link to Oracle filter functions.

Function Name	Oracle Function
touches	SDO_TOUCH
filter	SDO_FILTER
contains	SDO_CONTAINS
covered by	SDO_COVEREDBY
inside	SDO_INSIDE
covers	SDO_COVERS
overlaps	SDO_OVERLAPS
equals to	SDO_EQUAL
intersects	SDO_ANYINTERACT
nn	SDO_NN

5.7.1.2.2 Time functions for creating calculated fields

Important: Enterprise Edition only

Time functions are available only for Enterprise Edition with SI licence.

If you have SI licence, in the qbe calculated field wizard there are available time functions.

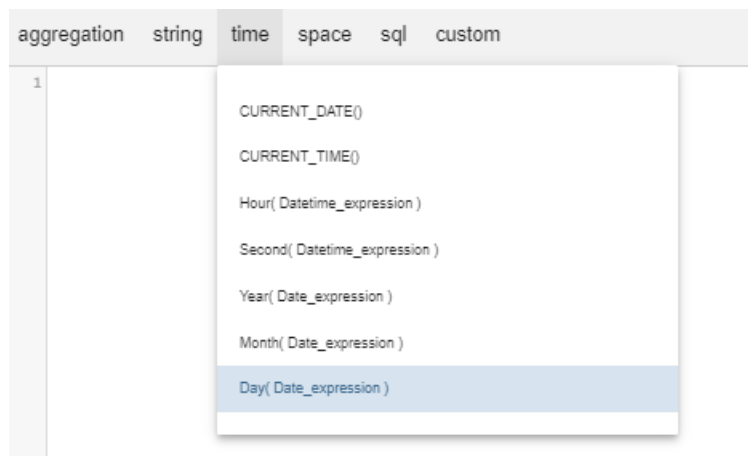


Fig. 5.246: Time functions.

See the table below:

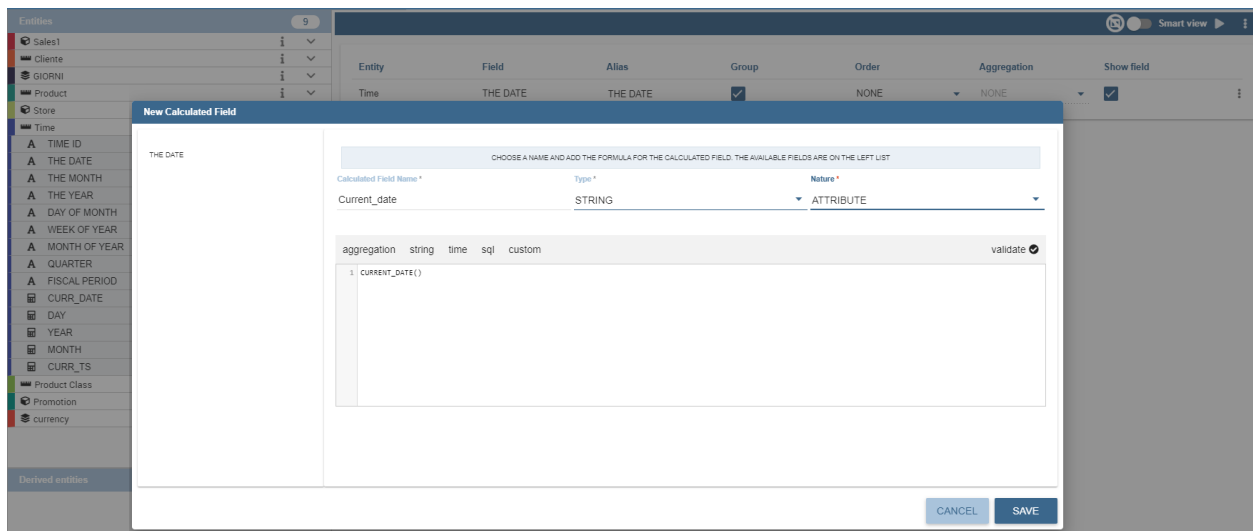


Fig. 5.247: Creating calculated field with function current_date().

In the picture below, you can see list of all created calculated fields:

In the next picture you can see result of you query:

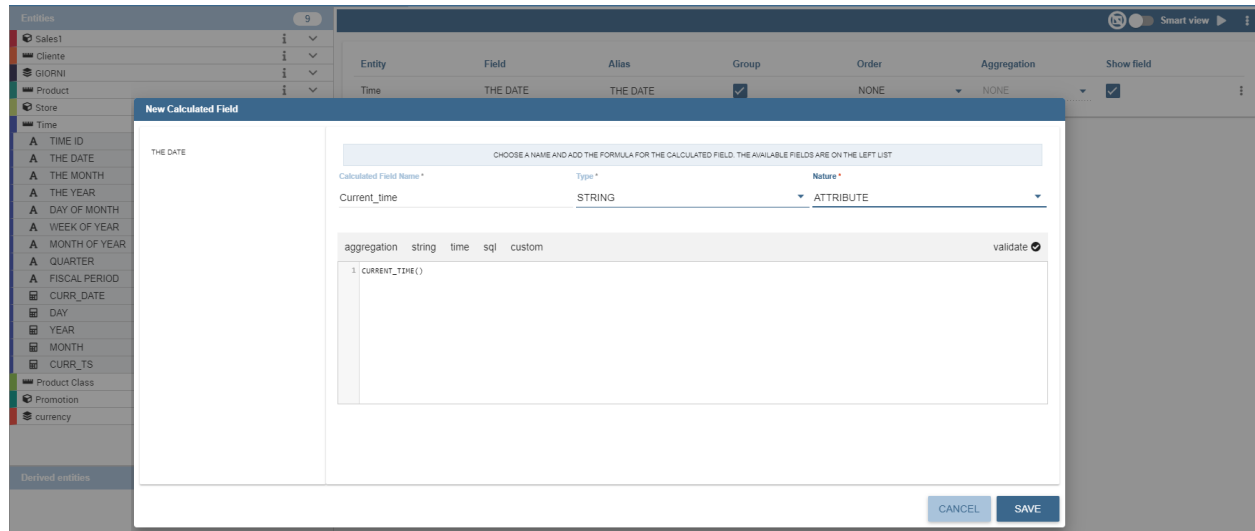


Fig. 5.248: Creating calculated field with function current_Time().

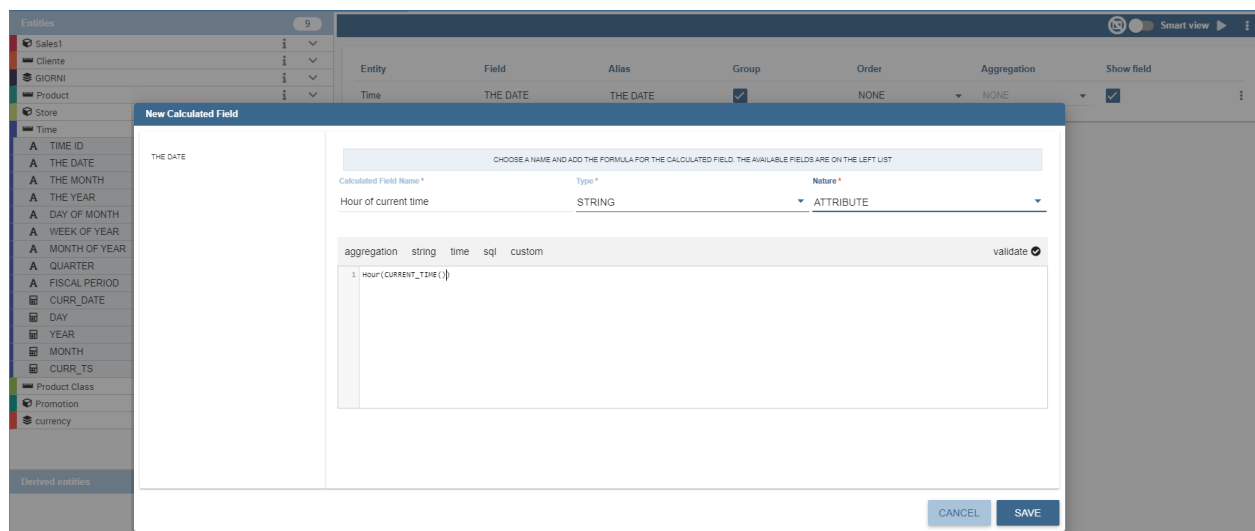


Fig. 5.249: Creating calculated field with function hour(date).

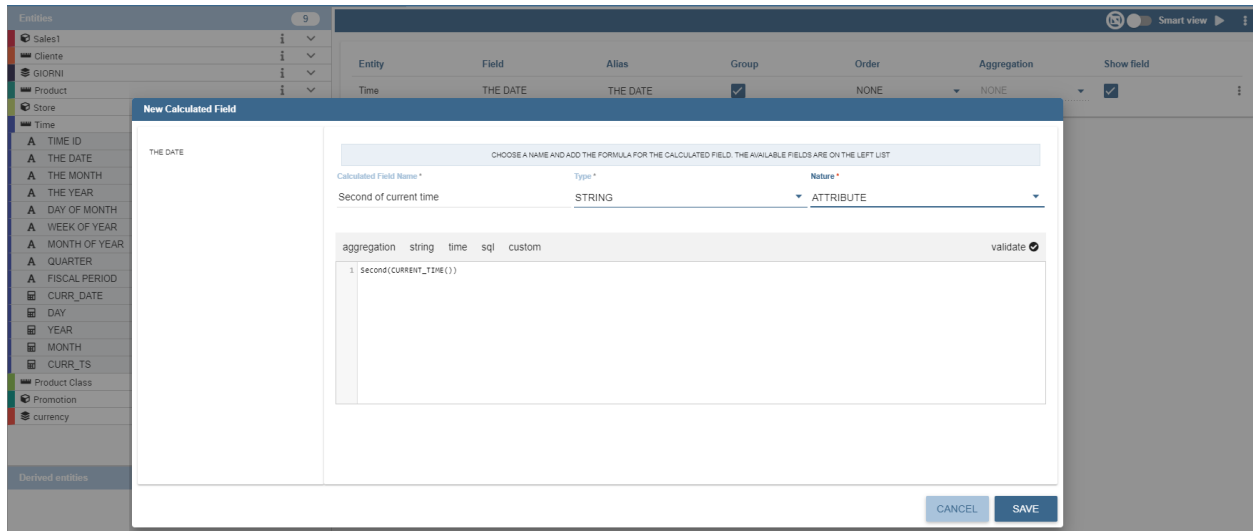


Fig. 5.250: Creating calculated field with function second(date).

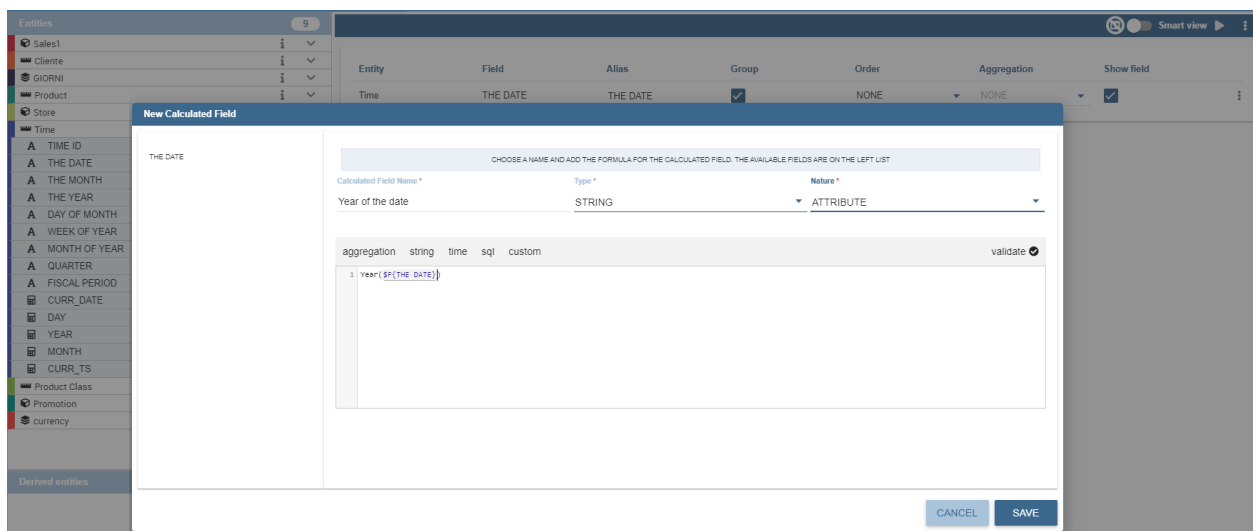


Fig. 5.251: Creating calculated field with function year(date).

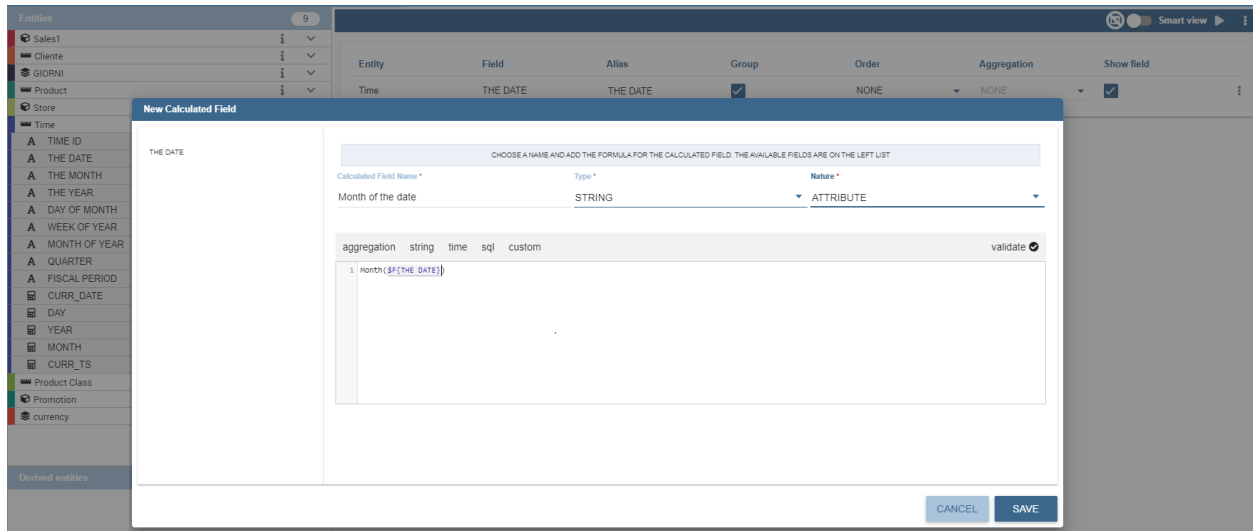


Fig. 5.252: Creating calculated field with function month(date).

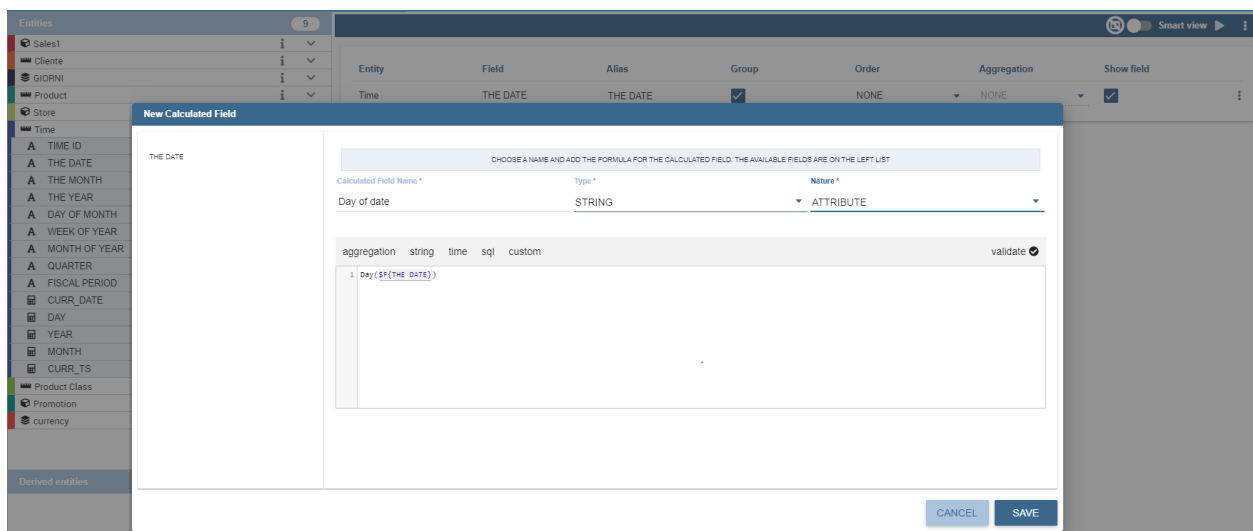


Fig. 5.253: Creating calculated field with function day(date).

Smart view						
Entity	Field	Alias	Group	Order	Aggregation	Show field
Time	THE DATE	THE DATE	<input checked="" type="checkbox"/>	NONE		<input checked="" type="checkbox"/>
Current date	Current date	Current date	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Current time	Current time	Current time	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Hour of current time	Hour of current time	Hour of current time	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Second of current time	Second of current time	Second of current time	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Year of the date	Year of the date	Year of the date	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Month of the date	Month of the date	Month of the date	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>
Day of the date	Day of the date	Day of the date	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>


Fig. 5.254: List of created calculated fields.

Preview							
THE DATE	Current date	Current time	Hour of current time	Second of current time	Year of the date	Month of the date	Day of the date
01/01/1997	01/11/2019	11:43:05	11	5	1997	1	1
02/01/1997	01/11/2019	11:43:05	11	5	1997	1	2
03/01/1997	01/11/2019	11:43:05	11	5	1997	1	3
04/01/1997	01/11/2019	11:43:05	11	5	1997	1	4
05/01/1997	01/11/2019	11:43:05	11	5	1997	1	5
06/01/1997	01/11/2019	11:43:05	11	5	1997	1	6
07/01/1997	01/11/2019	11:43:05	11	5	1997	1	7
08/01/1997	01/11/2019	11:43:05	11	5	1997	1	8
09/01/1997	01/11/2019	11:43:05	11	5	1997	1	9
10/01/1997	01/11/2019	11:43:05	11	5	1997	1	10
11/01/1997	01/11/2019	11:43:05	11	5	1997	1	11
12/01/1997	01/11/2019	11:43:05	11	5	1997	1	12
13/01/1997	01/11/2019	11:43:05	11	5	1997	1	13
14/01/1997	01/11/2019	11:43:05	11	5	1997	1	14
15/01/1997	01/11/2019	11:43:05	11	5	1997	1	15
16/01/1997	01/11/2019	11:43:05	11	5	1997	1	16
17/01/1997	01/11/2019	11:43:05	11	5	1997	1	17
18/01/1997	01/11/2019	11:43:05	11	5	1997	1	18
19/01/1997	01/11/2019	11:43:05	11	5	1997	1	19
20/01/1997	01/11/2019	11:43:05	11	5	1997	1	20

Fig. 5.255: Result of the query.

5.7.1.2.2.1 Subqueries

The **QbE Engine** also supports the definition and usage of subqueries similarly to the SQL language. As a result, you can define a subquery and use it within a filter in association to the in/not in operator, as shown in Figure below. To

create a new subquery, which can be used as a filter inside the main query, click on  button, on the left part, in **Derived entities** toolbar. In the main view you will see that you are able to add fields in subquery.

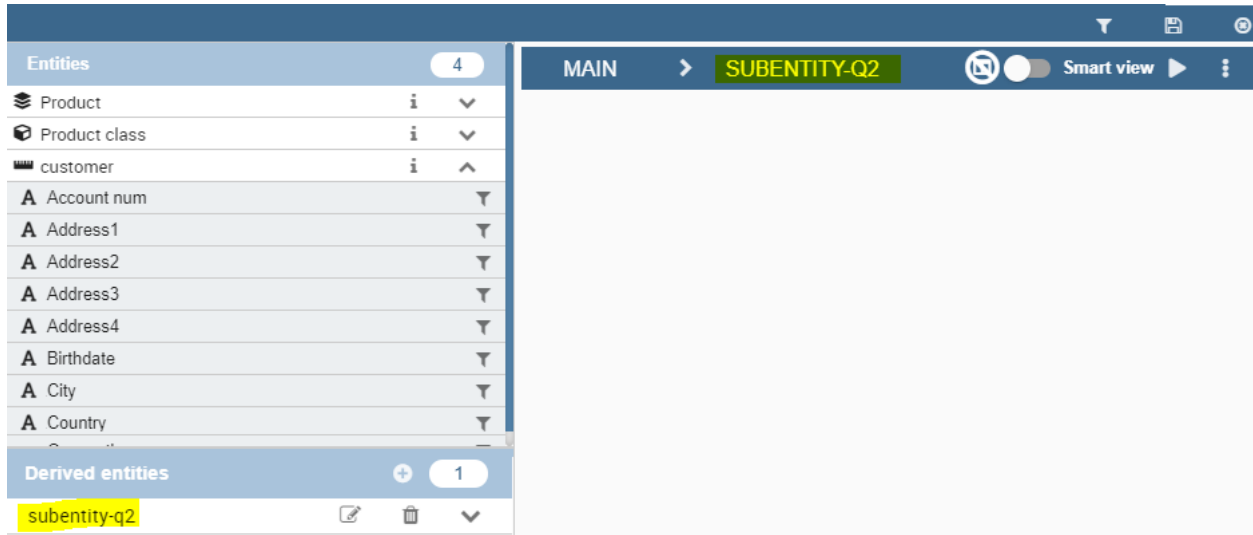


Fig. 5.256: QbE subquery view.


You can easily return to main query clicking on **MAIN** button in the query editor toolbar.

To use the sub-query inside the main query, simply choose from target type **Subquery option**, from **Target** choose subquery that you want and set the type of condition (**IN** or **NOT IN**). Now the subquery is used to provide values within the filter, in a similar way to SQL subqueries.



Fig. 5.257: QbE query: use of a subquery in a filter.

5.7.1.2.2.2 Parameters

The **QbE Engine** also supports the definition and usage of parameters that can be used to filter the data using qbe filter. To create a new parameter, which can be used as a filter inside the main query, click on  button, in the main query toolbar.





Parameters				
Name	Type	Default value	Multivalue	
city	String		<input checked="" type="checkbox"/>	 
country	String	Mexico	<input type="checkbox"/>	
profit	Number		<input type="checkbox"/>	

Fig. 5.258: QBE parameter view.

To use the parameter inside the main query, simply choose from target type **Parameter option** and from **Target** choose parameter that you want. Now the parameter is used to provide values within the filter.

Field	Condition	Target type
Customer : COUNTRY	Equals to	Parameter

city
country
profit

Fig. 5.259: QbE query: use of a parameter in a filter.

5.8 Meta Web

In this chapter 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.

5.8.1 Metamodel creation

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 next figure, 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 available, 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).

Name *

MyModel

Description

MyModel

Category *

Inventory

Data Source *

Foodmart

META WEB

Enable Meta Web

Lock model

Fig. 5.260: Setting the metamodel basic information.

After you have compiled these information, you can use the browse button to upload a model developed through an external (and specific) tool or you can click on “Save” on the top right corner of the screen and use the Meta Web engine to build it through Knowage interface. 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.

5.8.2 Association with analytical drivers

The new feature in meta web are analytical drivers and their connection to meta model via filters. In this section we will show how to practically define this association.

To add a new parameter, you can click on the tab **Drivers** and then on a **Add** button, see the next figure.

Choose a name for the title of driver. Then choose analytical driver from drop-down menu that you wish to associate to the meta model.

Once you have selected the driver, you should write the **exact URL** of the corresponding parameter. Then set if the driver is multivalue or not. By default, drivers are set to be mandatory so you don't have the option to change that like you have for analytical document.

After you have completed the definition of a parameter you can save it by clicking on main **Save** button in the upper right corner. To add further parameters, click on the **Add** button. Repeat the same procedure as many times you want.

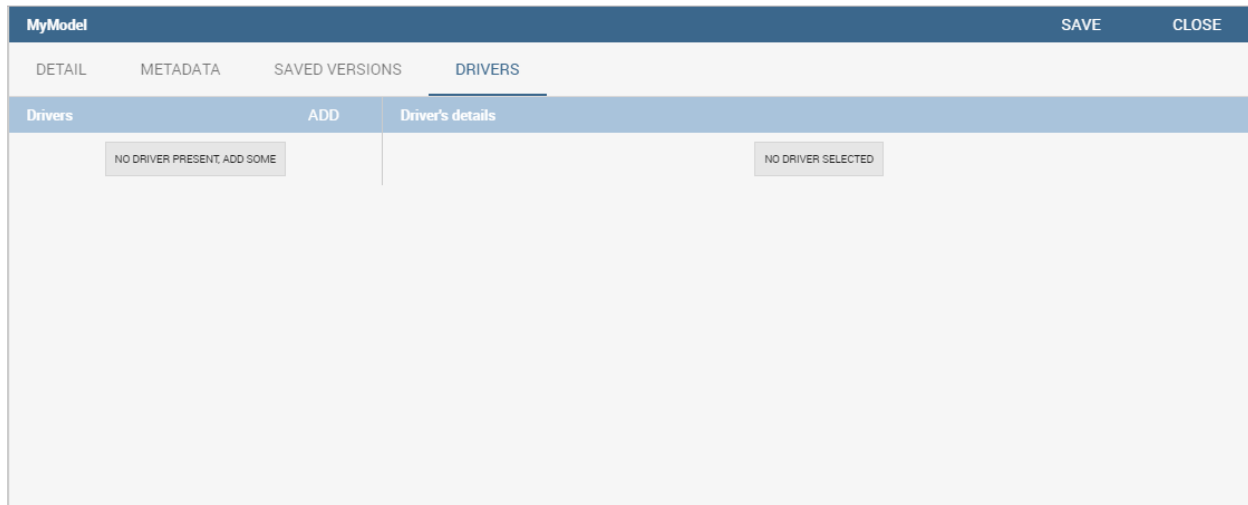


Fig. 5.261: Association with analytical driver panel.

At this point you may wish to change the order of parameters (i.e., how they are presented to the user). To do so, click on the arrow in the list of drivers.

In the following we will see some special operations that can be performed on drivers associated to a meta model.

5.8.2.1 Correlation between parameters

In the context of a meta model, 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.

To set the correlation, select child parameter which will show you the details of that particular driver and then click on the **Add condition** button to open pop-up window for defining data correlation.

In general, to configure a correlation within a meta model 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.

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.

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

MyModel

SAVE

CLOSE

DETAIL

METADATA

SAVED VERSIONS

DRIVERS

Drivers

ADD

Country
countryUrl

City
cityUrl

Country

Analytical driver *

Country

Url name *

countryUrl

Multivalue

Driver data condition

ADD CONDITION

NO DATA CONDITION SELECTED

Fig. 5.262: Association with analytical driver panel.

Driver data condition

ADD CONDITION

NO DATA CONDITION SELECTED

Fig. 5.263: Adding data correlation.

Driver data conditions

SET DRIVER DATA CONDITION

Analytical Driver "City" depends from the Analytical driver

Country ▼

Filter Operator

(Operator used to compare a column value of the Analytical driver and the value of the Analytical driver on which it depends)

equal ▼

Check to apply driver use mode on document ☒ true

Analytical driver use name

All :

Columns generated by the Analytical driver use

country ▼

CANCEL

SAVE

Fig. 5.264: Definition of the correlation.

MyModel

SAVE

CLOSE

DETAIL

METADATA

SAVED VERSIONS

DRIVERS

Drivers

ADD

Country countryUrl	↓	🗑
State province provinceUrl	↑ ↓	🗑
City cityUrl	↑	🗑

Driver's details

Title *

City

Analytical driver *

City ▼

Url name *

cityUrl

☐ Multivalue

Driver data condition

ADD CONDITION

equal value of the parameter countryUrl	🗑
equal value of the parameter provinceUrl	🗑

Fig. 5.265: Multiple correlations.

5.8.2.2 Correlation through LOV and drivers

In previous sections we saw how to set correlation through the GUI available in meta model panel, but there is also the possibility to get the same result using the link between LOV and analytical drivers. More in depth, the user must have previously configured a driver that runs values that can be used in the “where” clause of a SQL query. Then the user must set a query-type LOV using the syntax

We stress that the AD_name is the name of the driver the administrator is trying to reach. Syntax for setting correlation through LOV configuration is:

Listing 5.14: Syntax for setting correlation through LOV configuration

```
$P{AD_name}
```

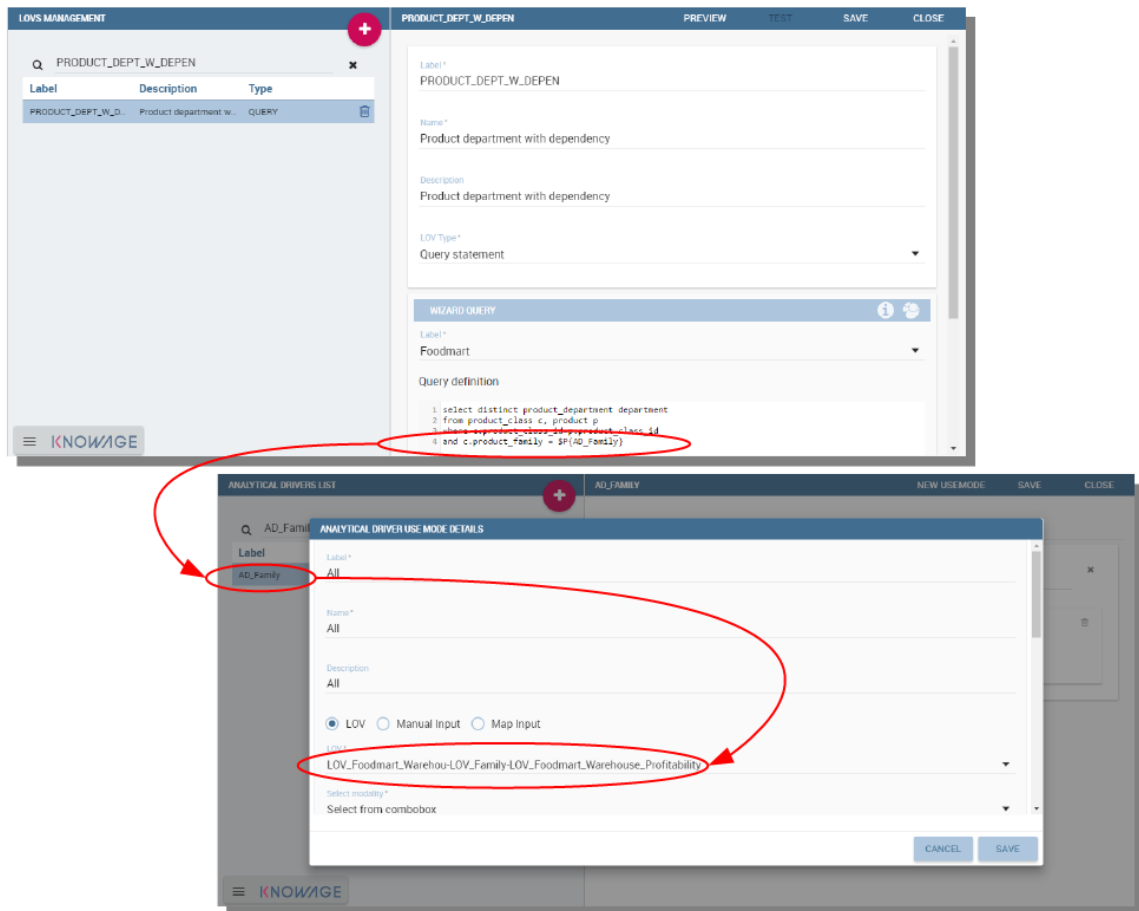


Fig. 5.266: Correlation passing driver values to LOV query .

As a result, when opening meta model, as soon as the user pick up a value from the “free” parameter, the other one is filtered and it will show only the value related to the previous selection, as shown in Figure below.

5.8.2.3 Create an empty model

The first time you enter the Meta Web, the interface (see Figure 10.2) will show you the available tables extracted from the selected data source.

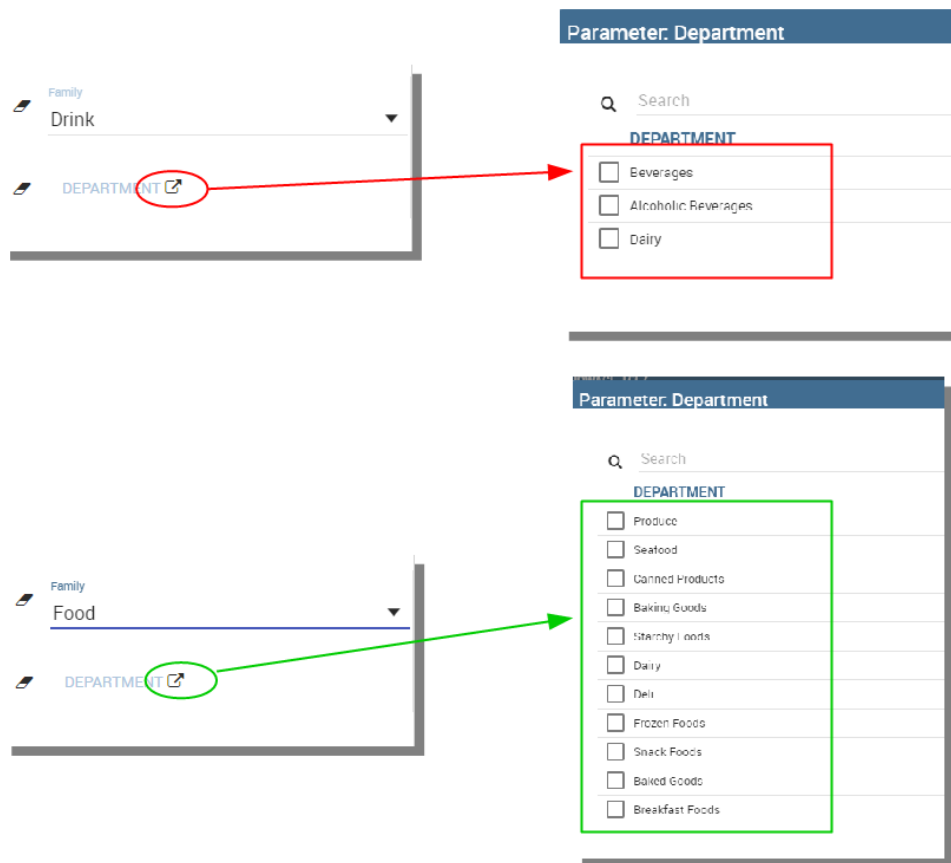


Fig. 5.267: Filtering with correlation.

FOODMARTTEST			CLOSE	CONTINUE
Q Search				
Table Name	Physical Model	Business Model		
account	<input type="checkbox"/>	<input type="checkbox"/>		
agg_c_10_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_c_14_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_c_special_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_g_ms_pcat_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_l_03_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_l_04_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_l_05_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_lc_100_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_ll_01_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
agg_pl_01_sales_fact_1997	<input type="checkbox"/>	<input type="checkbox"/>		
category	<input type="checkbox"/>	<input type="checkbox"/>		
currency	<input type="checkbox"/>	<input type="checkbox"/>		
cust_to_dept	<input type="checkbox"/>	<input type="checkbox"/>		
cust_to_dept_agg	<input type="checkbox"/>	<input type="checkbox"/>		
<input type="checkbox"/> Select all Physical Model <input type="checkbox"/> Select all Business Model				

Fig. 5.268: 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 information. 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.

5.8.2.4 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.

The “Physical Model” tab contains the tables that the user has checked earlier. On the left side of the interface you will see a tree like structure with the list of tables imported in the Physical Model (see figure below).

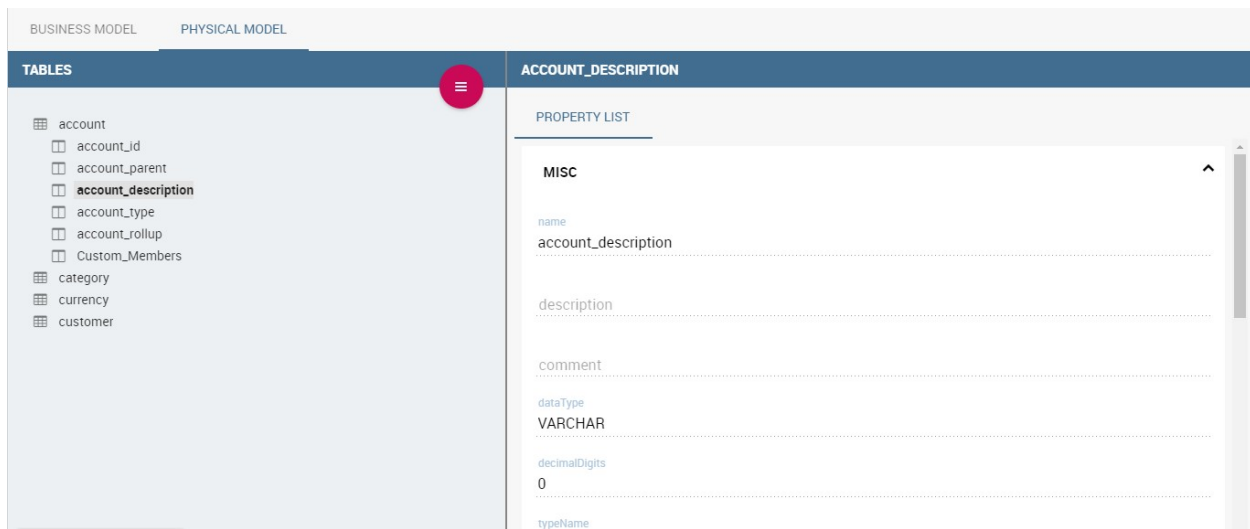


Fig. 5.269: Physical Model Tab.

The “hamburger-like” icon lets the user to update the Physical Model at any time. Referring to the figure below, selecting the “Update Physical Model” option the user can refresh the model metadata.

As shown below, the interface shows if tables have been added or deleted to the datasource and lets the user to add tables to the Physical Model.

Each table of Physical Model brings the information read from data base. Selecting each table, the interface shows on the right the list of its properties (**Property List** tab) and its foreign keys (**Foreign Keys** tab). Clicking on the icon on the left of each Physical Table, it is possible to expand the corresponding node. Highlight each column name to see (on the right side of the screen) a list of properties, like data type or length.

The Business Model tab, shown below, allows the user to custom the model in terms of column name, type, visibility, format, etc.

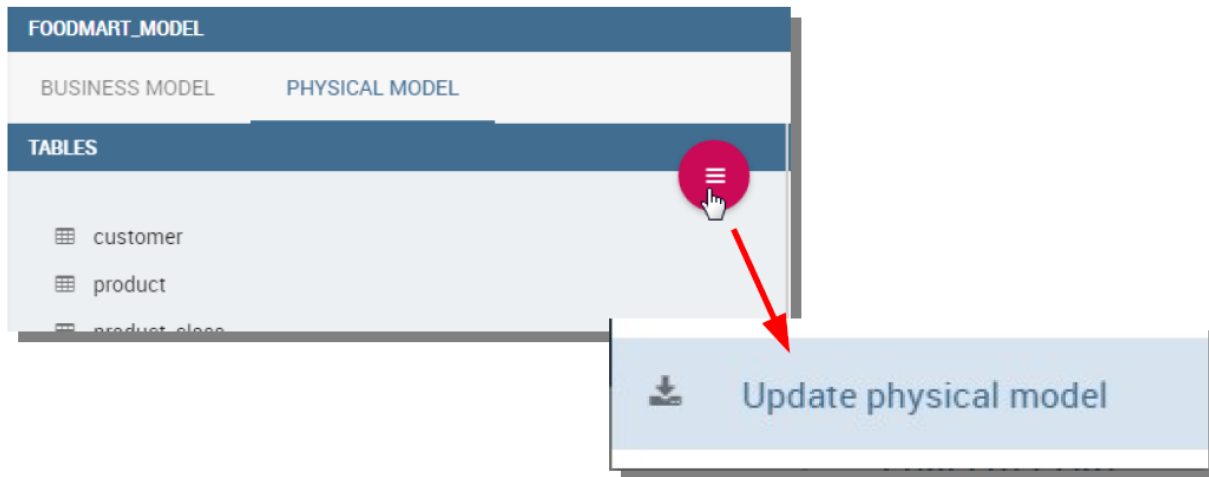


Fig. 5.270: Update the physical model.

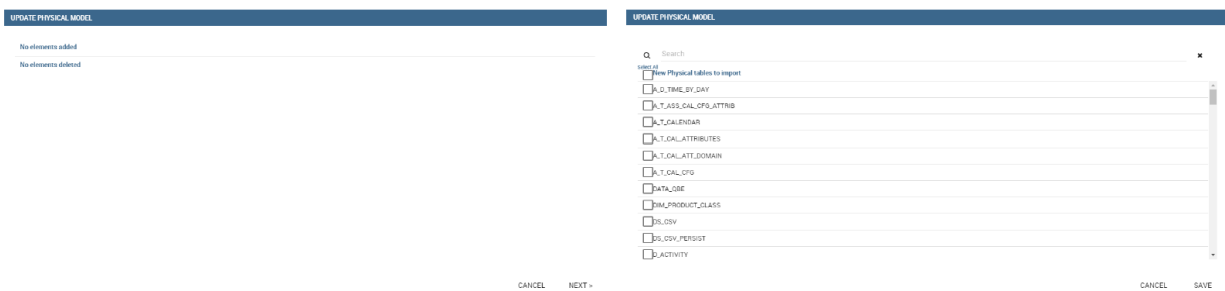


Fig. 5.271: Update the physical model.

BUSINESS MODEL		PHYSICAL MODEL								
Business Class/Business View		Customer						DELETE		
Business Class		PROPERTY LIST		ATTRIBUTES		CALCULATED FIELD		INBOUND	OUTBOUND	FILTER
Customer 29 Attributes										
Sales 11 Attributes										
Product 14 Attributes										
Store 24 Attributes										
Time 14 Attributes										
Product Class 5 Attributes										
Promotion 7 Attributes										

In this view, you see all the Business Class created at the first initialization. As well, the Business Classes are represented in a tree structure on the left side of the page. Clicking on each business class name, generic information are reported in the five tabs available on the right side of the page (Figure below).

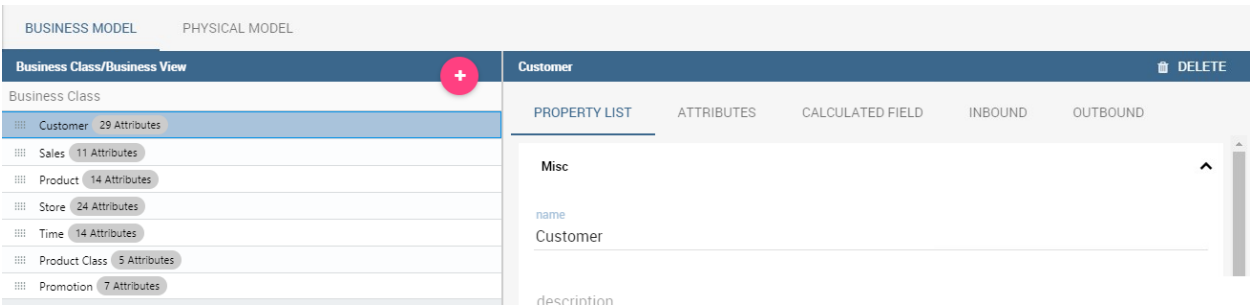
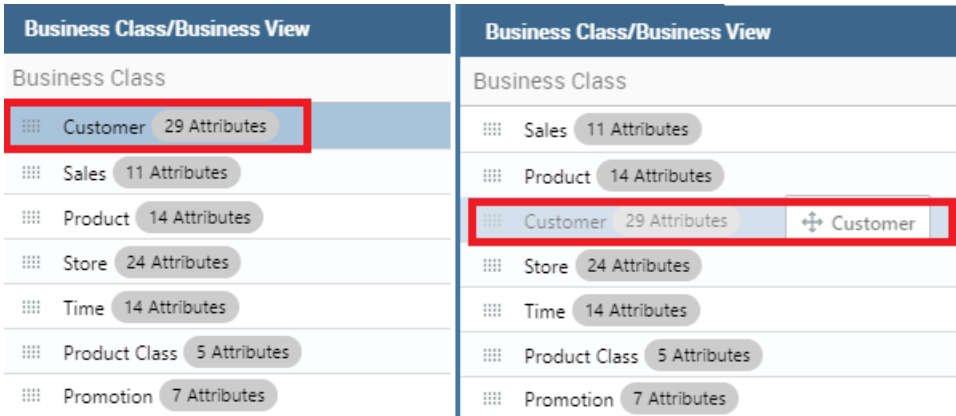


Fig. 5.273: Exploring Business Class properties.

Change the order of the business classes

If you want to change order of the business classes, you can do it using drag and drop functionality.



Using the **Property List** tab, the user can custom the business class name, type a description and see the corresponding physical table name. Here the user can also choose to hide the business class setting its visibility to false. Furthermore, when specifying the business class type, the user activates some peculiar functions that can be used in the QbE interface we described in Section 9.1. For instance, selecting the geographic dimension, the user will be able to apply the spatial functions to the dimension fields available in the QbE interface.

The **Attributes** tab lets the user to define which columns to be used as primary keys and which are effectively functional for the Business Class (not to be confused with the visibility condition). Note that, for instance, it is not possible to disable the “In Use” option when the field has already been set as foreign key.

The **Calculated field** tab is used to configure computed measures or attributes. Click on the dedicated button, as shown below, to create a new field. Use the list of functions to retrieve right function syntax and the list of fields on the left to correctly type the fields name.

The **Inbound** and **Outbound** tabs are described in the following section.

5.8.2.5 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);

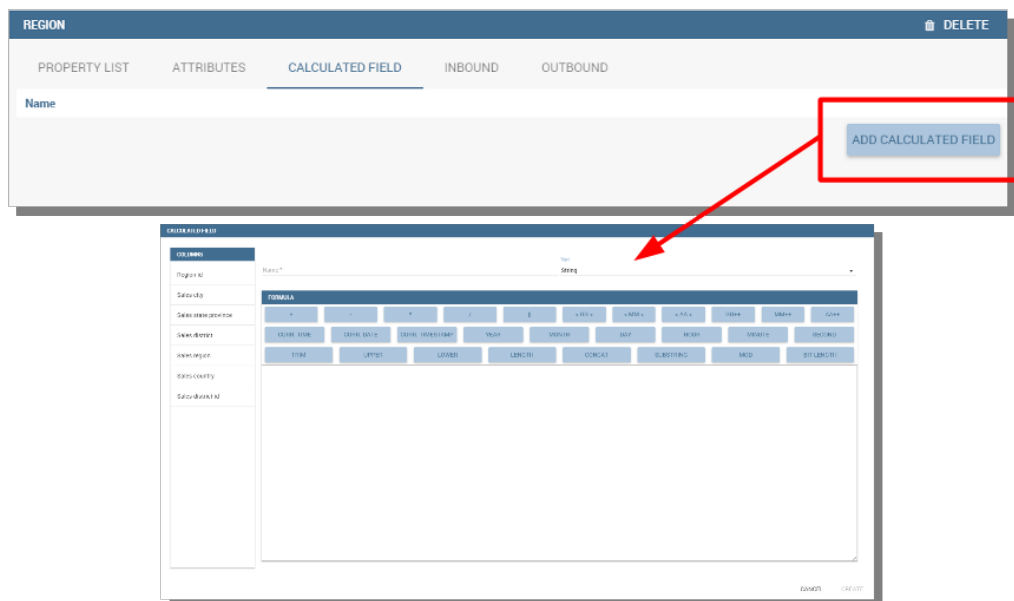


Fig. 5.274: Add calculated fields.

- **Outbound:** relationships that have the selected Business Class as a source (so the starts from).

The two relationships differ then for the direction of the bounds between tables that they establish.

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.

In Figure above 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” to save.

We stress that the cardinality of the outbound relationship can be of two types:

- 1 to N,
- 1 to N*.

Use the second type of cardinality when the type of cardinality can be optional.

As well, the cardinality of the inbound relationship can be of two types:

- N to 1,
- N* to 1.

Use the second type of cardinality when the type of cardinality can be optional.

OUTBOUND

Insert a Business Relationship Name *

MyNewRelationship

Cardinality

1 to N

Source Business Class

Customer

Target Business Class

Account

SOURCE ATTRIBUTES	TARGET ATTRIBUTES
Customer id	Account id Customer id
Account num	Account parent
Lname	Account description
Fname	Account type

CANCEL

CREATE

Fig. 5.275: Setting the outbound relationship.

5.8.2.6 SQL Filter

There is a new feature that is added in meta web. It is SQL Filter which we can define in Filter tab in meta web as you can see in the figure below. SQL Filter is used for applying already defined drivers in query.

Customer

DELETE

PROPERTY LIST

ATTRIBUTES

CALCULATED FIELD

INBOUND

OUTBOUND

FILTER

SQL expression

SQL filter is expression that is added in the end of query as part of where clause. The right syntax for sql filter is: column_name = \$P{url_name_of_the_driver}. For example: city = \$P{cityUrl}. If you want to add more than one filter, you can connect them with an operator (AND, OR...) as you can see in an example in figure below.

If you want to add filter for multivalue driver the right syntax is this: column_name IN (\$P{url_name_of_the_driver}). For example: city IN (\$P{cityUrl}).

5.8.2.7 Create a new business class

In the “Business Model” tab, the sandwich icon lets the user add other Business Classes (from the tables of the Physical Model) or a Business View (a combination of more tables with a predefined join path).

When clicking on the icon, as shown in Figure above), and selecting “New Business Class”, a new dialog asks to the users to:

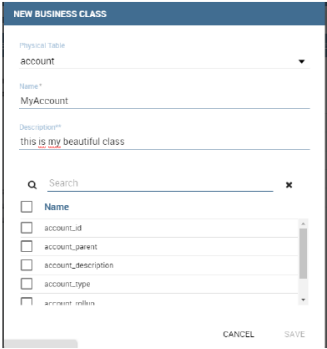
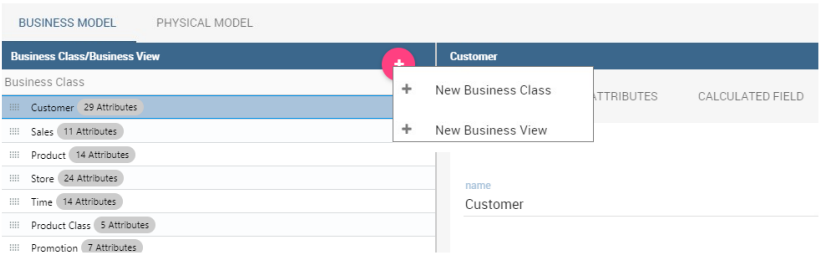
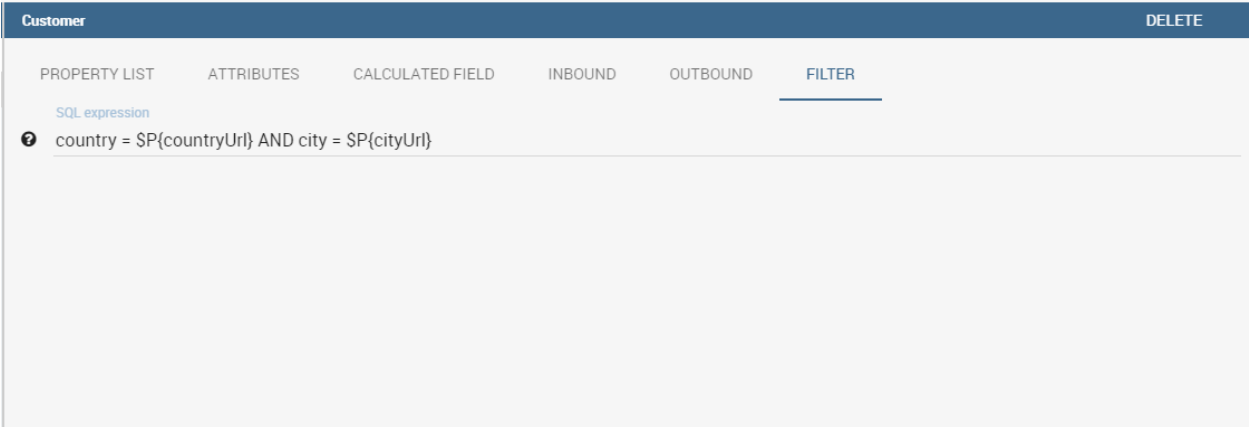


Fig. 5.276: Create a new business class.

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

Then click on save to add the business class.

As well, when clicking on “New Business View”, as reported in Figure below the user is asked to select two or more tables from the available ones and insert a description for this new business view.

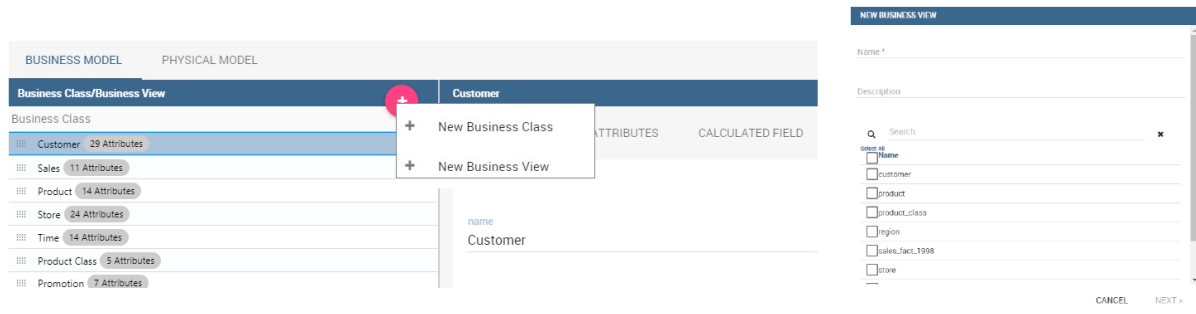


Fig. 5.277: Create a new business view.

Then, moving to the next step, the user must join tables through specific columns, typically the tables' foreign keys. Figure below shows an example.

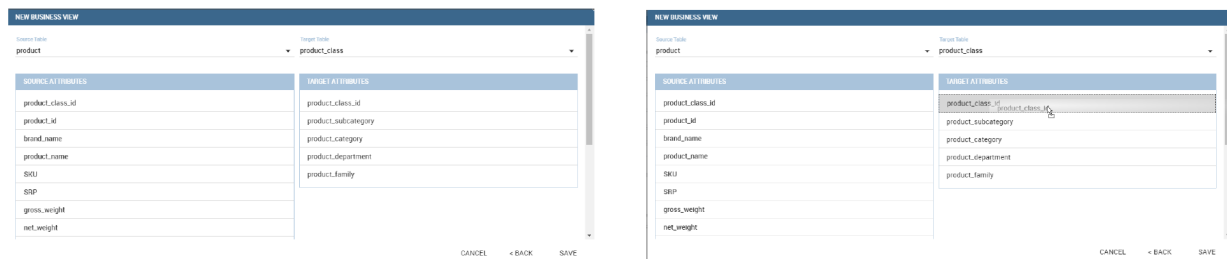


Fig. 5.278: Create a new business view.

For each business view, the interface reports the same property tabs we saw for each business class. In addition, the user finds the **Join relationships** tab and the **Physical table** tab, as highlighted in the following figure. The “Join relationships” tab shows the join clauses set to create the business view while the “Physical Table” tab recalls the physical table names.

5.8.2.8 Table property list

Scrolling the table “Property list” tab, the user finds the **Type** menu item. Expanding the related combobox the user can custom the table type among the ones available and listed below.

5.8.2.9 Column property list

As well, the user can employ each field property list (see next figure) to both inspect the object and custom it.

The **Structural** area covers an important role for the field properties. Here the user can set:

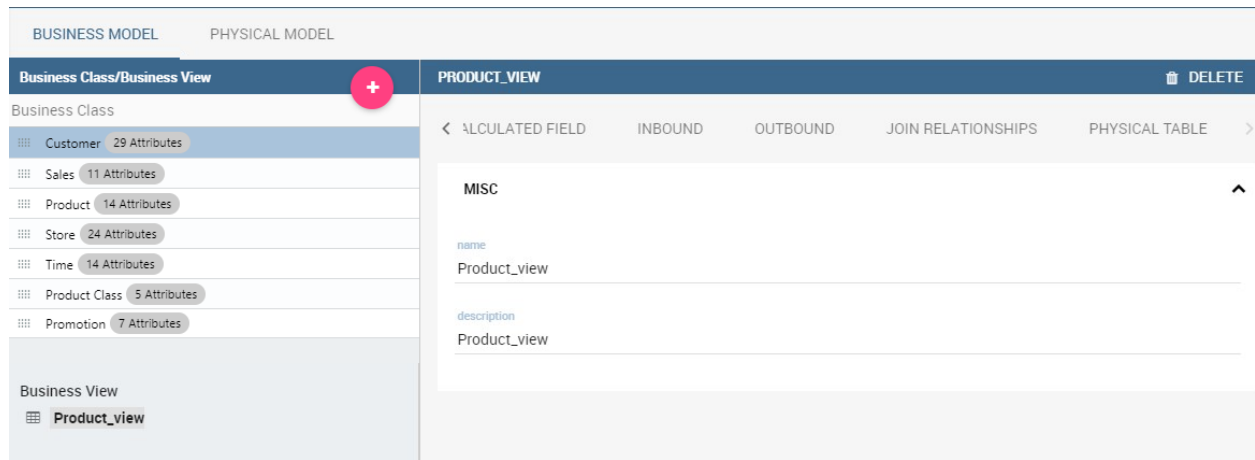


Fig. 5.279: Additional property tabs for business view.

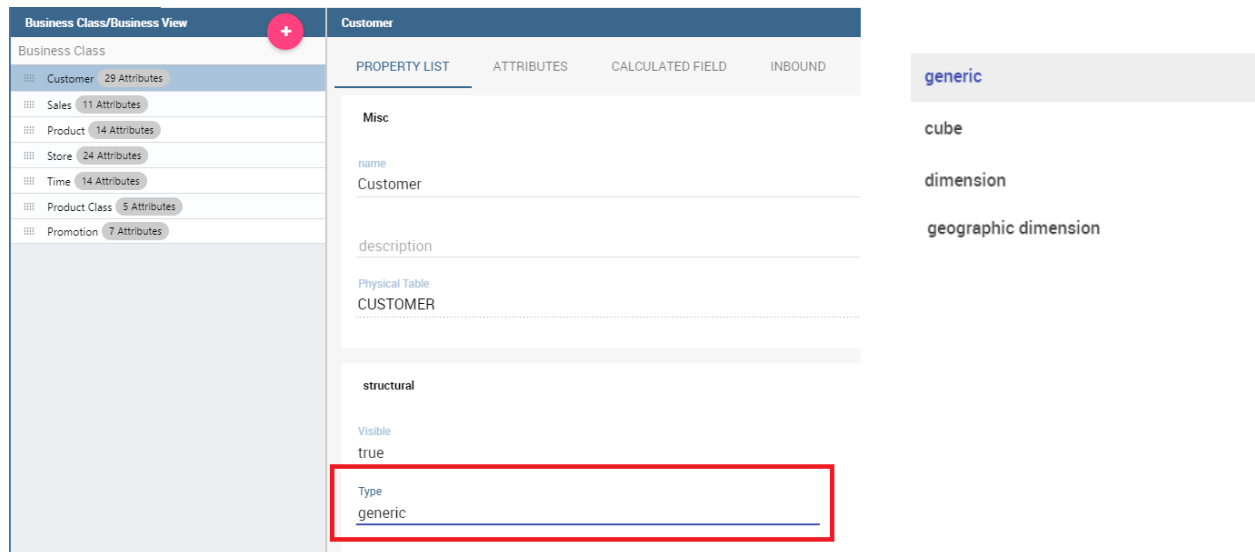


Fig. 5.280: Table property list.

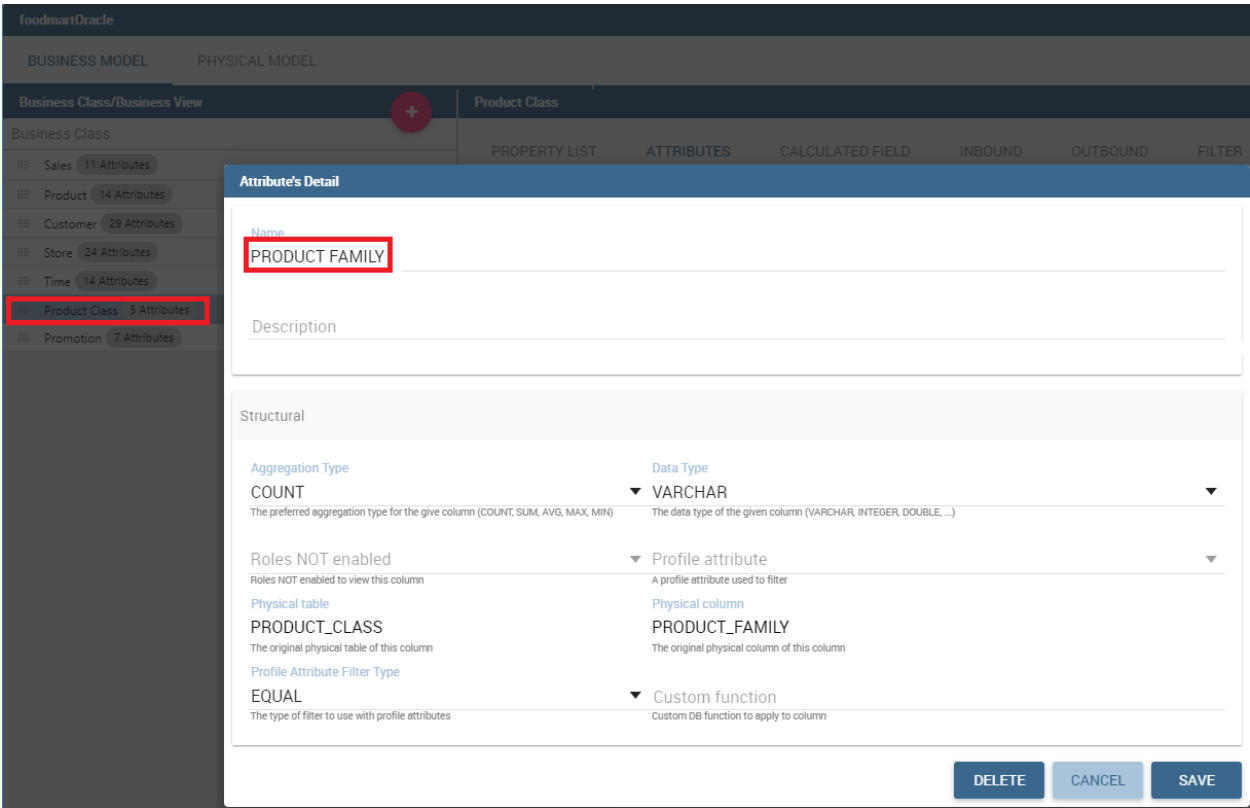


Fig. 5.281: Column property list.

- **Visibility** over the field,
- **Type**, among measure, attribute, calendar, temporal_id, the_date and hour_id,
- **Aggregation type** for measure field type,
- **Format string**, to custom the format of the string for measure field type,
- **Profile attribute**, to filter the field (and then the table records) by the user profile attributes (note that the combobox lists the available profile attributes),
- **Profile attribute filter type**, to define the filter operator among “equals to”, “in”, “like”,
- **Data type**, to indicate the field data type.

In the **Behavioural Model** area, the user can assign the field’s visibility permission to specific roles.

In the **Physical** area, recalls the physical table and field name from which the field have been take.

Add new column into business class

If you did not choose all columns from physical table, when you were creating new business class, you can do it easily. Click on business class in which you want to add new column. Open Attributes tab and click on above column name.

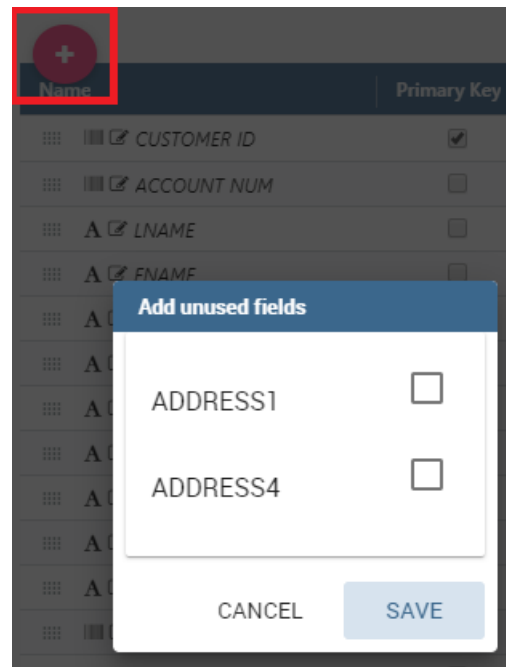


Fig. 5.282: Remove existing column from business class

If you want to remove column from business table, you need to click on business class from which you want to remove column. Open Attributes tab and click on three dots of column you want to delete. It will open details panel. Click on delete button.

If you want to change order of columns, you can do it using drag and drop functionality.

5.8.2.10 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:

Customer

DELETE

PROPERTY LIST

ATTRIBUTES

CALCULATED FIELD

INBOUND

OUTBOUND

FILTER

Attribute's Detail

Name

ADDRESS2

Description

Structural

Aggregation Type

COUNT

The preferred aggregation type for the give column (COUNT, SUM, AVG, MAX, MIN)

Data Type

VARCHAR

The data type of the given column (VARCHAR, INTEGER, DOUBLE, ...)

Roles NOT enabled

Roles NOT enabled to view this column

Physical table

CUSTOMER

The original physical table of this column

Profile Attribute Filter Type

EQUAL

The type of filter to use with profile attributes

Profile attribute

A profile attribute used to filter

Physical column

ADDRESS2

The original physical column of this column

Custom function

Custom DB function to apply to column

DELETE

CANCEL

SAVE

Fig. 5.283: Change the order of the columns

Customer						DELETE
PROPERTY LIST						
ATTRIBUTES						
CALCULATED FIELD						
INBOUND						
OUTBOUND						
FILTER						
Name	Primary Key	visibility	Type	Description		
CUSTOMER ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ACCOUNT NUM	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
LNAME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
FNAME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ADDRESS2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ADDRESS3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
MI	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
CITY	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		

Customer						DELETE
PROPERTY LIST						
ATTRIBUTES						
CALCULATED FIELD						
INBOUND						
OUTBOUND						
FILTER						
Name	Primary Key	visibility	Type	Description		
CUSTOMER ID	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ACCOUNT NUM	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
LNAME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
MI	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
FNAME	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ADDRESS2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
ADDRESS3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
CITY	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		
STATE PROVINCE	<input type="checkbox"/>	<input checked="" type="checkbox"/>	attribute	null		

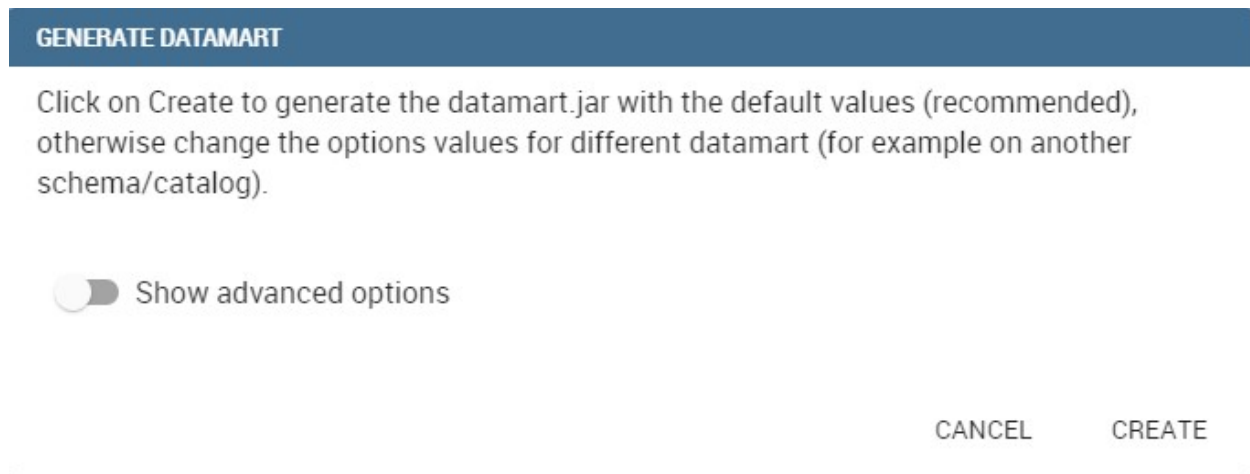


Fig. 5.284: Generate datamart dialog.

If you just press “Create” the generation of the datamart begins otherwise clicking on the switch “Show Advanced options” (see figure below) the user can modify model name, change the schema or the catalogue of the database used to query the metamodel. This option is useful when the user wishes to build the model on a source schema and produce the datamart on a different one. Furthermore, the user can check the **Generate for registry** box. In this instance, the generated datamart will be used as a registry (but will not be exploited as a QbE). The **Include source code** produces a “file.jar” containing both the compiled code (.class) and the source files (.java), useful for the debugging process.

When the datamart is generated it will be possible to query the metamodel accessing it in the Workspace interface.

5.8.2.11 Additional functions for business model

In this section, we briefly describe the generic available options for business model development. Referring to figure below, the user first finds the **Lock Model**: if enabled, only the user who developed the model can modify it.

Once the model has been saved, some more options are enabled. In fact, the user can make advantage of the **Metadata** section. Clicking the **Import metadata** button, the metadata information related to the business classes (their composition, properties, etc.) are stored into the (metadata) Knowage database. Those information can then be visualized via specific document (developed for the data lineage context).

Finally the **Saved versions** section the user keeps trace of model changes over time. Furthermore it is possible to restore old versions by checking the active column. Selecting the “three-dots” icon the user can download the jar file or the model itself or delete the version. Figure below shows an example.

5.9 Registry

A Registry document allows users to write, cancel and modify items of a datamart. Knowage allows users to implement a Registry document through the **Qbe Engine**. By the way it has a different graphical interface compared to the Qbe one. An example is given in the following figure. In next chapters we will see how to navigate a Registry document (*Registry features* paragraph) and how to create a new one (*Registry development* paragraph).

5.9.1 Registry features

The execution of a Registry document opens a plain table: the records are shown in rows and they can be browsed using the pagination available at the bottom of the window. We underline that it is possible to edit each item of the

GENERATE DATAMART

Click on Create to generate the datamart.jar with the default values (recommended), otherwise change the options values for different datamart (for example on another schema/catalog).

Show advanced options

Model Name

Foodmart_model

Schema Name

Catalog Name

foodmart

☐ Generate for registry

☐ Include source code

CANCEL
CREATE

Fig. 5.285: Generate datamart dialog: advanced options.

Foodmart_model
SAVE
CLOSE

DETAIL
METADATA
SAVED VERSIONS
DRIVERS

Name *

Foodmart_model

Description

Foodmart_model

Category *

Default Model Category ▼

Data Source *

Foodmart ▼

Upload File: BROWSE

☐ Enable Meta Web

☐ Lock model

Fig. 5.286: Business model lock.

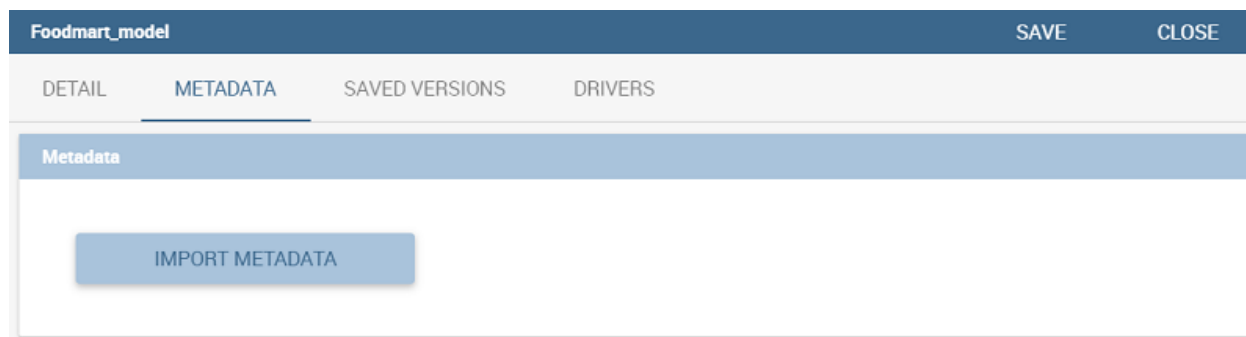


Fig. 5.287: Importing metadata.

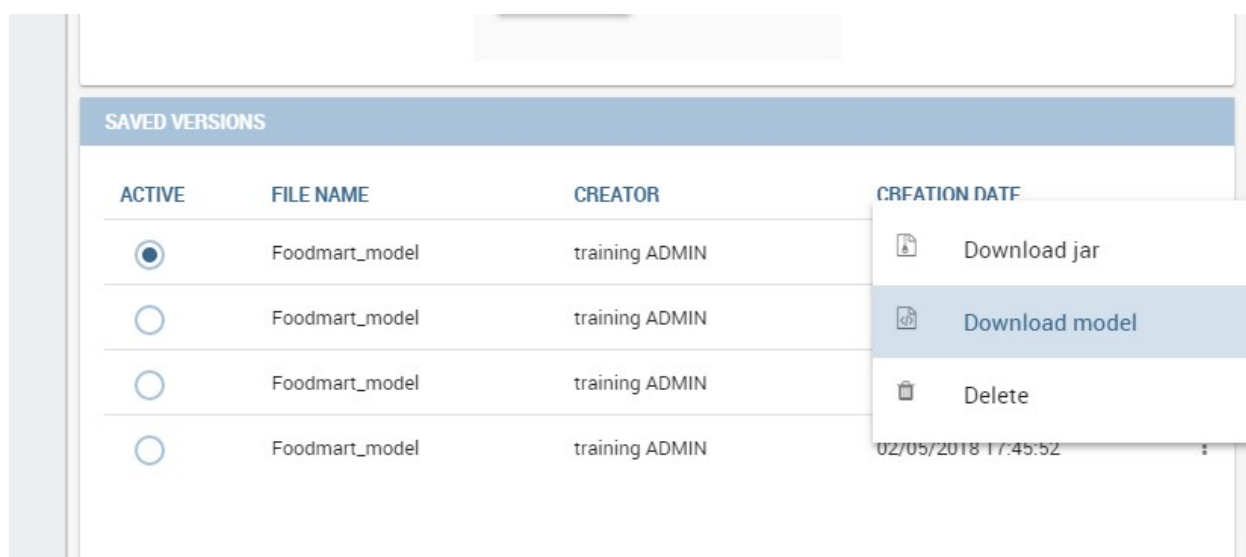


Fig. 5.288: Saved version functionalities.


Testlink - Registry sample											
Registry Document											
Filters											
Store type Sales city											
FILTER											
	store name	store type	store city	store state	has florist	has coffee bar	has video store	first opened date	store square foot	Foreign key - city	
1	HQhjdjsa	My store type 2	Beverly Hills	CA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	09/05/2019	5,432	Newton	<input type="checkbox"/>
2	Store 11	æiou??	Portland	OR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	09/16/1976	20,319	Portland	<input type="checkbox"/>
3	Store 124	Supermarket	Acapulco	Guerrero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/08/1982	23,593	Acapulco	<input type="checkbox"/>
4	Store 124	Deluxe Supermarket	Hidalgo	Zacatecas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/24/1968	30,584	Hidalgo	<input type="checkbox"/>
5	Store 13	Deluxe Supermarket	Salem	OR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	04/01/1957	27,694	Salem	<input type="checkbox"/>
6	Store 15	Supermarket	Seattle	WA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	07/23/1969	21,215	Seattle	<input type="checkbox"/>
7	Store 16	Supermarket	Spokane	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	08/23/1974	30,268	Spokane	<input type="checkbox"/>
8	Store 17	Deluxe Supermarket	Tacoma	WA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	05/30/1970	33,858	Tacoma	<input type="checkbox"/>
9	Store 18	Mid-Size Grocery	Hidalgo	Zacatecas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	06/28/1969	38,382	Hidalgo	<input type="checkbox"/>
10	Store 19	Deluxe Supermarket	Vancouver	BC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/27/1977	23,112	Vancouver	<input type="checkbox"/>
11	Store 2	Small Grocery	Bellingham	WA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	04/02/1970	28,206	Bellingham	<input type="checkbox"/>
12	Store 20	Mid-Size Grocery	Victoria	BC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	02/05/1980	34,452	Victoria	<input type="checkbox"/>
13	Store 21	Deluxe Supermarket	San Andres	DF	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	02/07/1986		San Andres	<input type="checkbox"/>
14	Store 22	Small Grocery	Walla Walla	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/23/1951		Walla Walla	<input type="checkbox"/>
15	Store 23	Mid-Size Grocery	Yakima	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	07/16/1977		Yakima	<input type="checkbox"/>

Fig. 5.289: Example of Registry document.

table if inside template that column is set to be editable. Just click on a cell you may wish to rearrange and type a string or a numeric value accordingly. Some examples are highlighted below.


	store name	store type	store city	store state	has florist	has coffee bar	has video store	first opened date	store square foot	Foreign key - city	
1	HQhjdjsa	My store type 2	Beverly Hills	CA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	09/05/2019	5,432	Newton	
2	Store 11	æiôu??	Portland	OR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	09/16/1976	20,319	Portland	
3	Store 11	Supermarket	Acapulco	Guerrero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/08/1982	23,593	Acapulco	
4	Store 124	Deluxe Supermarket	Hidalgo	Zacatecas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/24/1968	30,584	Hidalgo	
5	Store 13	Deluxe Supermarket	Salem	OR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	04/01/1957	27,694	Salem	
6	Store 15	Deluxe Supermarket	Hidalgo	Zacatecas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/25/1968			
7	Store 16	Store 13 changed	Deluxe Supermarket	Salem	OR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	04/13/1957	27694		
8	Store 17	Store 144	Small Grocery	San Francisco	CA	<input type="checkbox"/>	<input type="checkbox"/>	11/24/1957			
9	Store 18	Store 15	Supermarket	Seattle	WA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	07/24/1969			
10	Store 19	Store 16	Supermarket	Spokane	WA	<input type="checkbox"/>	<input type="checkbox"/>	08/23/1974	30,268	Spokane	
11	Store 2	Store 17	Deluxe Supermarket	Tacoma	WA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	05/30/1970	33,858	Tacoma	
12	Store 20	Store 18	Mid-Size Grocery	Hidalgo	Zacatecas	<input type="checkbox"/>	<input type="checkbox"/>	06/28/1969	38,382	Hidalgo	
13	Store 21	Store 19	Deluxe Supermarket	Vancouver	BC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/27/1977	23,112	Vancouver	
14	Store 22										
15	Store 23	Mid-Size Grocery	Yakima	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	07/16/1977		Yakima	


Fig. 5.290: Editing table cells.



Moreover, you can add new rows from scratch selecting the “Add row” button  in the header of last column. Insert in each cell the corresponding value: string, number or date. “Add row” button will be available if inside template there is a configuration: `<CONFIGURATION name=”enableAddRecords” value=”true”/>`

	store name	store type	store city	store state	has florist	has coffee bar	has video store	first opened date	store square foot	Foreign key - city	
1	HQhjdjsa	My store type 2	Beverly Hills	CA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	09/05/2019	5,432	Newton	
2	Store 11	æiôu??	Portland	OR	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	09/16/1976	20,319	Portland	
3	Store 11	Supermarket	Acapulco	Guerrero	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/08/1982	23,593	Acapulco	
4	Store 124	Deluxe Supermarket	Hidalgo	Zacatecas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/24/1968	30,584	Hidalgo	
5	Store 13	Deluxe Supermarket	Salem	OR	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	04/01/1957	27,694	Salem	
6	Store 15	Supermarket	Seattle	WA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	07/23/1969	21,215	Seattle	
7	Store 16	Supermarket	Spokane	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	08/23/1974	30,268	Spokane	
8	Store 17	Deluxe Supermarket	Tacoma	WA	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	05/30/1970	33,858	Tacoma	
9	Store 18	Mid-Size Grocery	Hidalgo	Zacatecas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	06/28/1969	38,382	Hidalgo	
10	Store 19	Deluxe Supermarket	Vancouver	BC	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/27/1977	23,112	Vancouver	
11	Store 2	Small Grocery	Bellingham	WA	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	04/02/1970	28,206	Bellingham	
12	Store 20	Mid-Size Grocery	Victoria	BC	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	02/05/1980	34,452	Victoria	
13	Store 21	Deluxe Supermarket	San Andres	DF	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	02/07/1986		San Andres	
14	Store 22	Small Grocery	Walla Walla	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/23/1951		Walla Walla	
15	Store 23	Mid-Size Grocery	Yakima	WA	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	07/16/1977		Yakima	

Fig. 5.291: Adding a new row.

Vice versa, you can delete one or more rows using the “Trash” icon  in the last column. “Trash” button will be available if inside template there is a configuration: `<CONFIGURATION name=”enableDeleteRecords” value=”true”/>`

It is important to click on the “Save” button  to store the adjustments done in the datamart. “Save” button is available in Functionality bar, above table.

Furthermore you can use filters, if implemented, available in the Functionality bar. Click on the “Filter” icon  to run the functionality. Otherwise, click on the “Cancel” icon  to clear the boxes off.

Note that, since records are displayed in a plain table, it is available a combobox (see figure below) which allows the user to visualize all fields related to the record of the previous cell and then change from one to another to get all data.

Filters

Store type

Sales city

▼

FILTER

Fig. 5.292: Functionality bar.

Testlink - Registry simple										
Registry Document										
Filters										
Store type										
Sales city										
FILTER										
	store name	store type	store city	store state	has florist	has coffee bar	has video store	first opened date	store square foot	Foreign key - city
3	Store 11	Mid-Size Grocery	terro	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/08/1982	23,593	Acapulco
4	Store 124		itecas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	03/24/1968	30,584	Hidalgo
5	Store 13	Gourmet Supermarket		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	04/01/1957	27,694	Salem
6	Store 15			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	07/23/1969	21,215	Seattle
7	Store 16	Deluxe Supermarket		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	08/23/1974	30,268	Spokane
8	Store 17			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	05/30/1970	33,858	Tacoma
9	Store 18		itecas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	06/28/1969	38,382	Hidalgo
14	Store 22			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	01/23/1951		Walla Walla
15	Store 23	Mid-Size Grocery	Yakima	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	07/16/1977		Yakima

Fig. 5.293: Select one field from a combobox.

5.9.1.1 JPivot Registry characteristics

It is possible to implement also a JPivot Registry document. The graphical features are very similar to the ones exposed in *Registry development* paragraph. An example is given below.

Testlink - Registry pivot table						
Registry Document						
Filters						
Store type						
FILTER						
store country	store state	store city	store type	store name	first opened date	store square foot
			jeioui汉字	jeioui汉字	08/16/1983	20
			Deluxe Supermarket	tretr	03/01/2020	33
			Gourmet Supermarket	twet	01/31/2020	30
			Mid-Size Grocery	TestStoreName	05/18/2020	45
			My store type 2	My Store2		0
			Supermarke	Mmm	02/03/2019	0
				Store for test	09/05/2019	5,432
				Store 8		0
			Supermarket	9999	09/05/2019	0
				Store for test	04/16/2020	5,432
Canada	BC	Vancouver	Deluxe Supermarket	Store 19	03/27/1977	23,112
		Victoria	Mid-Size Grocery	Store 20	02/05/1980	34,452
Mexico	DF	Mexico City	Mid-Size Grocery	Store 9	03/18/1955	36,509
		San Andres	Deluxe Supermarket	Store 21	02/07/1986	0

Fig. 5.294: Example of Jpivot Registry document.

In this case the table shows columns organized in a hierarchical way and a grouping function is implemented. From the left to the right the columns contain fields at different detail levels. The last column in our example in the figure above contains numeric data. Such a field is grouped at the “country” level. The grouping level depends on the configurations made on template building.

In the JPivot instance it is not allowed to add, modify or cancel rows. Furthermore, it is not permitted to edit cells which contain string items while the numeric ones are still changeable. If implemented, filters are still available.

5.9.2 Registry development

To create a Registry document there must be available a datamart schema on Knowage Server. Then you must edit an XML template. The latter is very similar to the one produced under the Qbe development but in this case you must add an appropriate tag. Indeed, if the template file has the **<REGISTRY>** tag the engine shows data in registry modality; namely it shows a table whose rows are manageable by end users by adding, editing or deleting them.

Here we exhibit a possible syntax for a Registry document.

Listing 5.15: Example (a) of template for Registry.

```

1  <?xml version="1.0" encoding="windows-1250"?>
2  <QBE>
3      <DATAMART name="RegFoodmartModel" />
4      <REGISTRY>
5          <ENTITY name="it.eng.knowage.meta.regfoodmartmodel.Product">
6              <FILTERS>
7                  <FILTER title="Class" field="product_subcategory" presentation="COMBO" />
8                  <FILTER title="Product name" field="product_name" presentation="COMBO" />
9              </FILTERS>
10             <COLUMNS>
11                 <COLUMN field="product_id" unsigned="true" visible="false" editable="false
12                 ↪ " format="####" />
13                 <COLUMN field="product_name" title="Product name" size="200"
14                 editor="MANUAL" sorter="ASC"/>
15                 <COLUMN field="product_subcategory" title="Class" size="200"
16                 ↪ product_class" />
17                 <COLUMN field="SKU" title="SKU" size="200" editor="MANUAL" />
18                 <COLUMN field="gross_weight" title="Gross weight" size="200" editor="MANUAL
19                 ↪ " />
20                 <COLUMN field="net_weight" title="Net weight" size="200" editor="MANUAL" />
21             </COLUMNS>
22             <CONFIGURATIONS>
23                 <CONFIGURATION name="enableDeleteRecords" value="true"/>
24                 <CONFIGURATION name="enableAddRecords" value="true"/>
25                 <CONFIGURATION name="isPkAutoLoad" value="true"/>
26             </CONFIGURATIONS>
27         </ENTITY>
28     </REGISTRY>
29 </QBE>

```

In particular, we give some details for each tag and main attributes.

- **ENTITY**: the entity name as in the model;
- **FILTERS**: possibility to define filters by specifying the title, the field (among shown columns) and the type among COMBO, MANUAL or DRIVER: in this last case user has also to specify the analytical driver that take this filter's value;
- **COLUMNS**: columns list specifying:
 - **field name**: the reference to the field identifier into the model,
 - **title**: the title of the column shown (optional),
 - **visible**: the visibility of the column (Optional, default true),
 - **editable**: the editability of the column (Optional, default true),
 - **color and format for numbers**: optional,
 - **editor**: the editor. Default type is free-text for simple column (not fk values), but for date is possible show the picker through the type PICKER. The format option specify the format date,

- **subEntity**: If the column is a reference key user can specify the subentity referred and the foreign key name; in this case the field shown will be of the entity referred and will be shown as combo if editable,
- **dependsFrom**: if the column content is logically correlated to other registry's column is possible specify this logic through this parameter. DependsFrom identifies the field name on which it depends (Optional),
- **dependsFromEntity**: usable only with dependsFrom parameter. It defines a different entity to resolve the correlation (Optional),
- **orderBy**: is used in case of foreign key, the combo box is ordered by the column here indicated, by default is the column extracted (Optional).
- **infoColumn**: if true ignore the column when inserting or updating the record (Optional).

We stress that it is mandatory to point at one datamart table using a column with a numeric key. The code line is highlighted in figure below. While, if not elsewhere specified, a descriptive column will be displayed by default.

Listing 5.16: Pointing at a numerical column.

```
1 <COLUMNS>
2 <COLUMN field="store_id" visible="false" editable="false" />
```

Still referring to the code above, we underline that the “product_subcategory” field is used as a subcategory. It belongs in fact to another table. In this case it is enough to add the attributes: subEntity=”rel_product_class_id_in_product_class” foreignKey=”rel_product_class_id_in_product_class”.

5.9.2.1 JPivot Registry instance

The Registry instance allows to develop also a Jpivot table. See the last figure (above) to have an idea while the syntax example is given in the next code:

Listing 5.17: Example (b) of template code for Registry.

```
1 <QBE>
2 <DATAMART name="foodmart" />
3 <REGISTRY pagination = "false" summaryColor="#00AAAA">
4 <ENTITY name="it.eng.knowage.meta.foodmart.Store">
5 <FILTERS>
6 <FILTER title="Store Type" field="store_type" presentation="COMBO" />
7 </FILTERS>
8 <COLUMNS>
9 <COLUMN field="store_id" visible="false" editable="false" />
10 <COLUMN field="store_country" title="store country" visible="true"
11 type="merge" editable="false" sorter="ASC" summaryFunction="sum" />
12 <COLUMN field="store_state" title="store state" visible="true"
13 type="merge" editable="false" sorter="ASC" />
14 <COLUMN field="store_city" title="store city" visible="true"
15 type="merge" editable="false" sorter="ASC" />
16 <COLUMN field="store_type" title="store type" type="merge" sorter="ASC" />
17 <COLUMN field="store_number" title="Number" size="150"
18 editable="true" format="#####" color="#f9f9f8" type="measure"/>
19 </COLUMNS>
20 <CONFIGURATIONS>
21 <CONFIGURATION name="enableDeleteRecords" value="true"/>
22 <CONFIGURATION name="enableAddRecords" value="true"/>
23 </CONFIGURATIONS>
24 </ENTITY>
25 </REGISTRY>
26 </QBE>
```

Note that to activate the JPivot modality it is important to add the attribute type=”merge” and have at least one numeric field. Furthermore the selected column fields must be hierarchically structured.

5.10 Create a Birt report

Reports represent the most popular type of business analysis. Indeed, they allow the visualization of data in a structured way and accordingly to predefined formats. The main characteristics of a report are:

- combination of numerical data (tables, lists, cross tables), charts and images;
- static and pixel-perfect layout;
- multi-page and multi-format output (PDF, HTML, DOC, etc.);
- organization of data by groups and sections;
- universal usage (summary, detail, analytical or operational);
- being suitable for off-line production and distribution (scheduled execution);
- ease of use.

For these reasons reports usually have a pervasive level of usage: they are used by developers to perform both synthetic and detailed analysis, having a particularly low level of difficulty.

BIRT, acronym for **Business Intelligence and Reporting Tools**, is an open source technology platform used to create data visualizations and reports. In Figure below you can see an example of BIRT report.

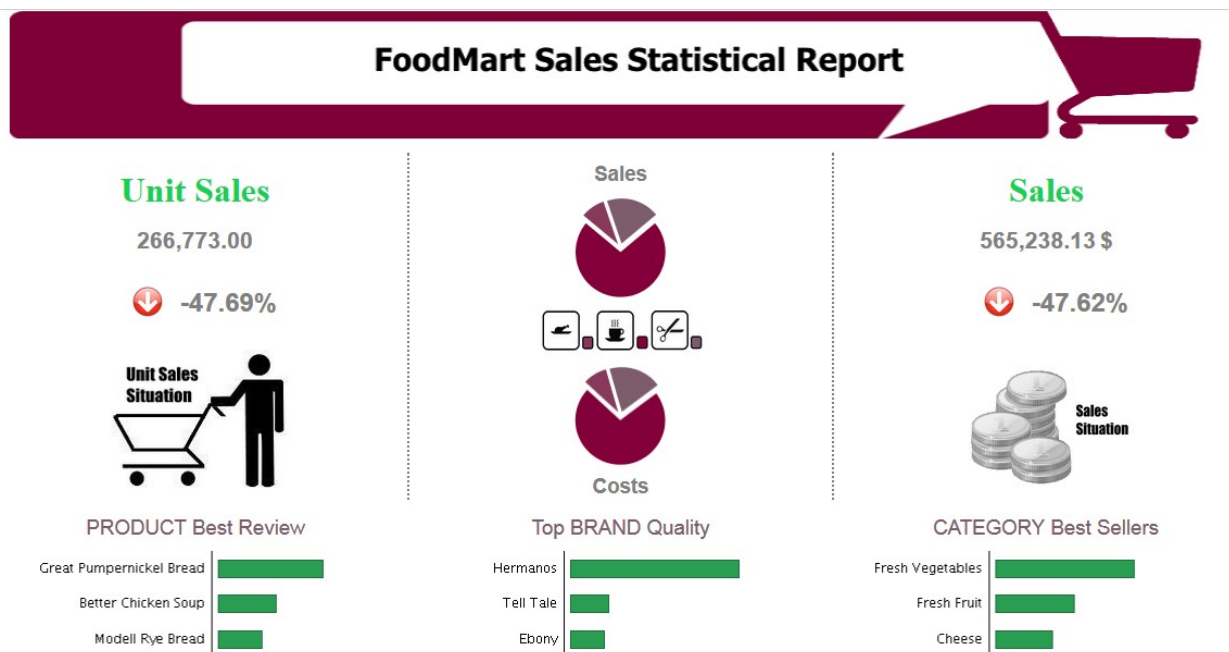


Fig. 5.295: Example of a BIRT report.

5.10.1 Developing a BIRT report

First of all you have to download the Birt report designer from here [Download Eclipse](#).

Here you can find some information about how install and use Birt report designer: [How install and use birt](#)

After that you can use the designer to create a new report.

- 1) create a new **Report Project**

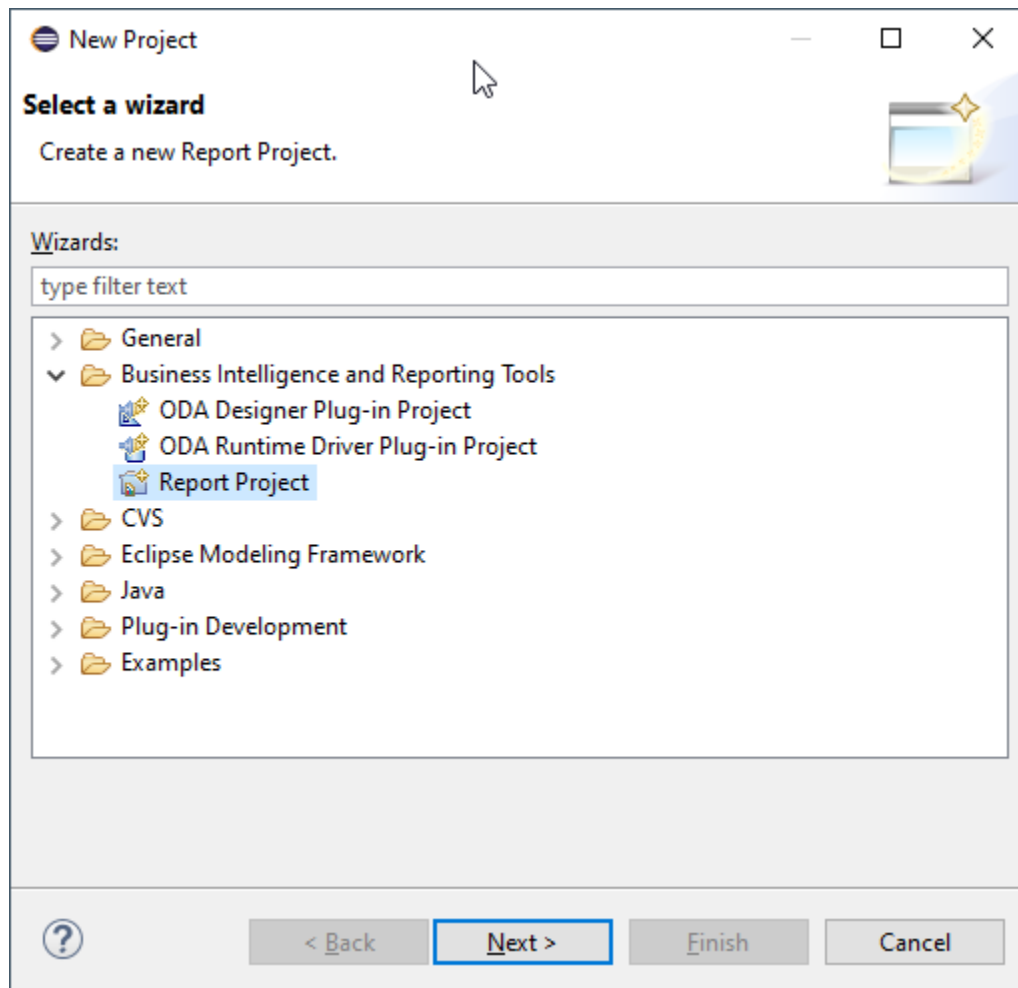


Fig. 5.296: Create a new Report Project.

2) create a new **Report**

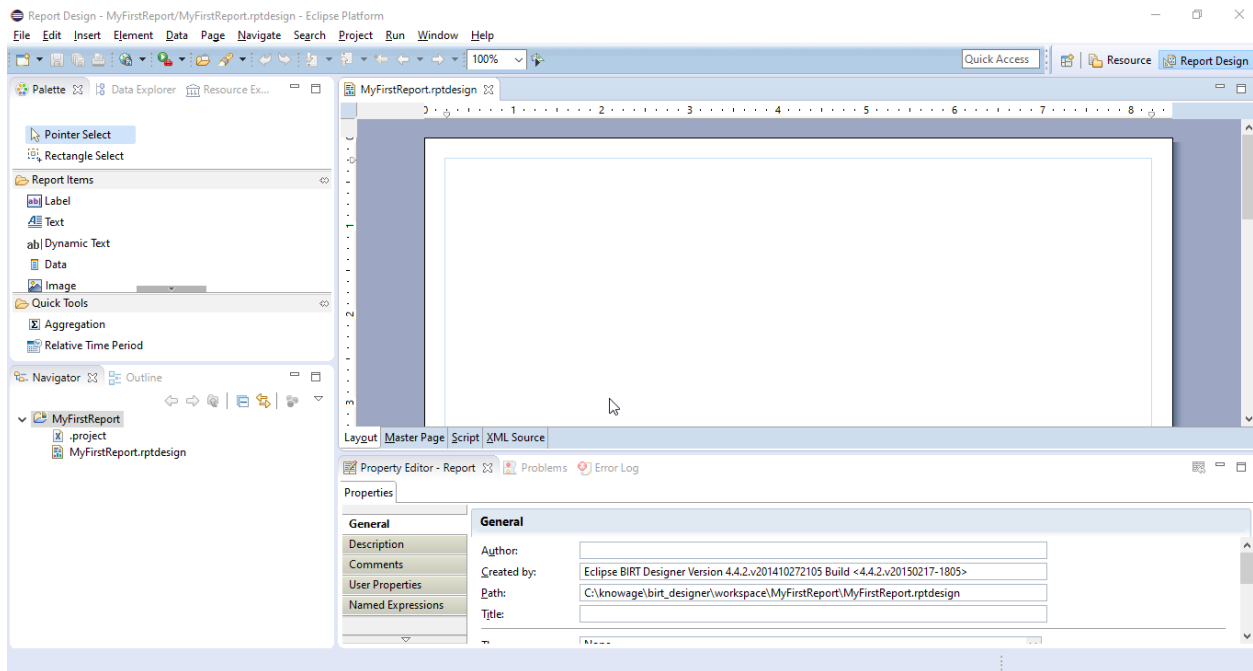


Fig. 5.297: Create a new Report.

3) Configure the data source, create the data set and design the report

Using the preview function you can test the report.

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

- **.rptdesign**: document template for reports that use the BIRT engine.

4) Deploy on KNOWAGE server using the KNOWAGE document browser you can install the report in the Server and test it.

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 birt designer to connect to the database and let the report to be executed locally. Make sure that the database set in the Server share the same schema of that defined in the designer.

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 below.

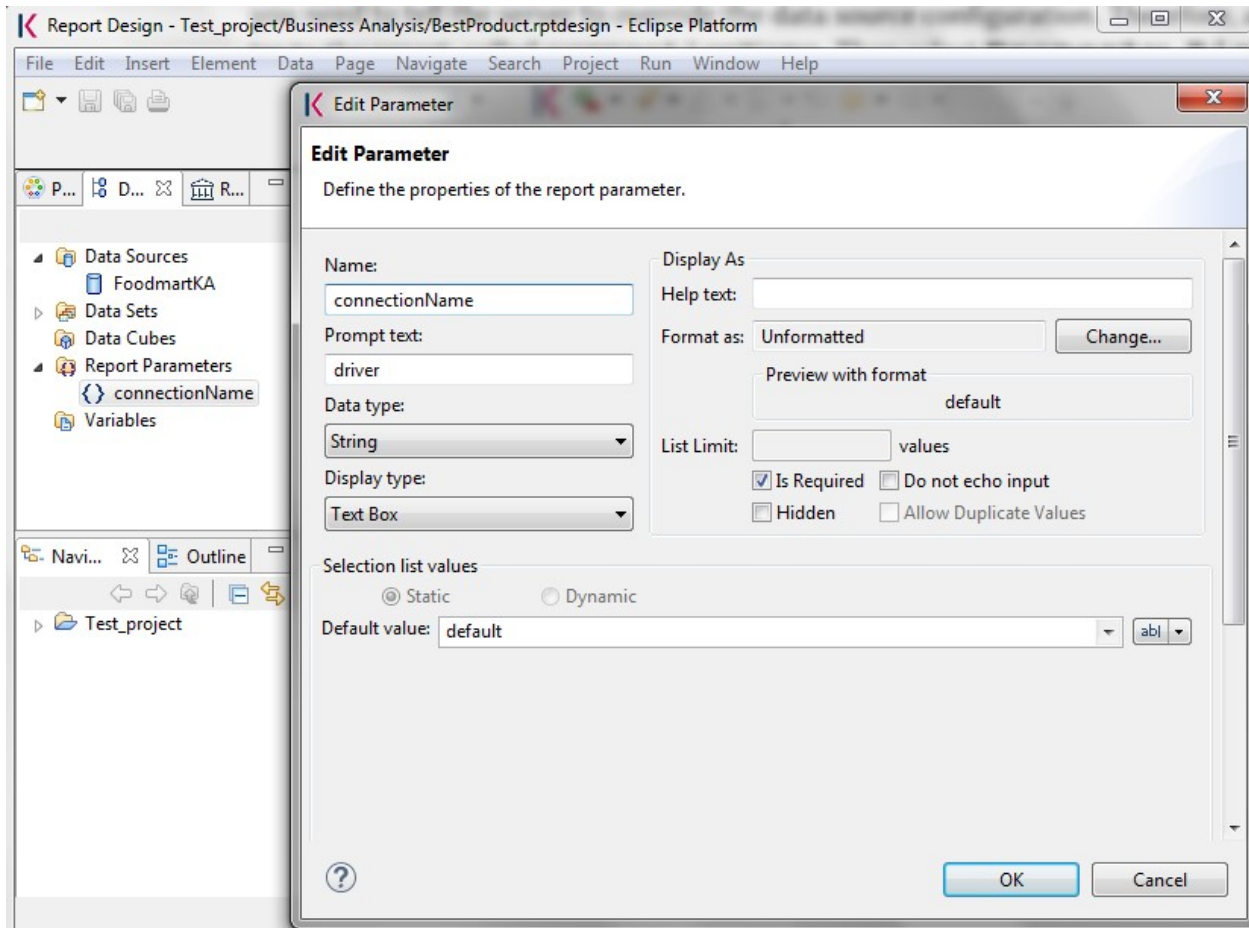


Fig. 5.299: Adding connectionName Parameter.

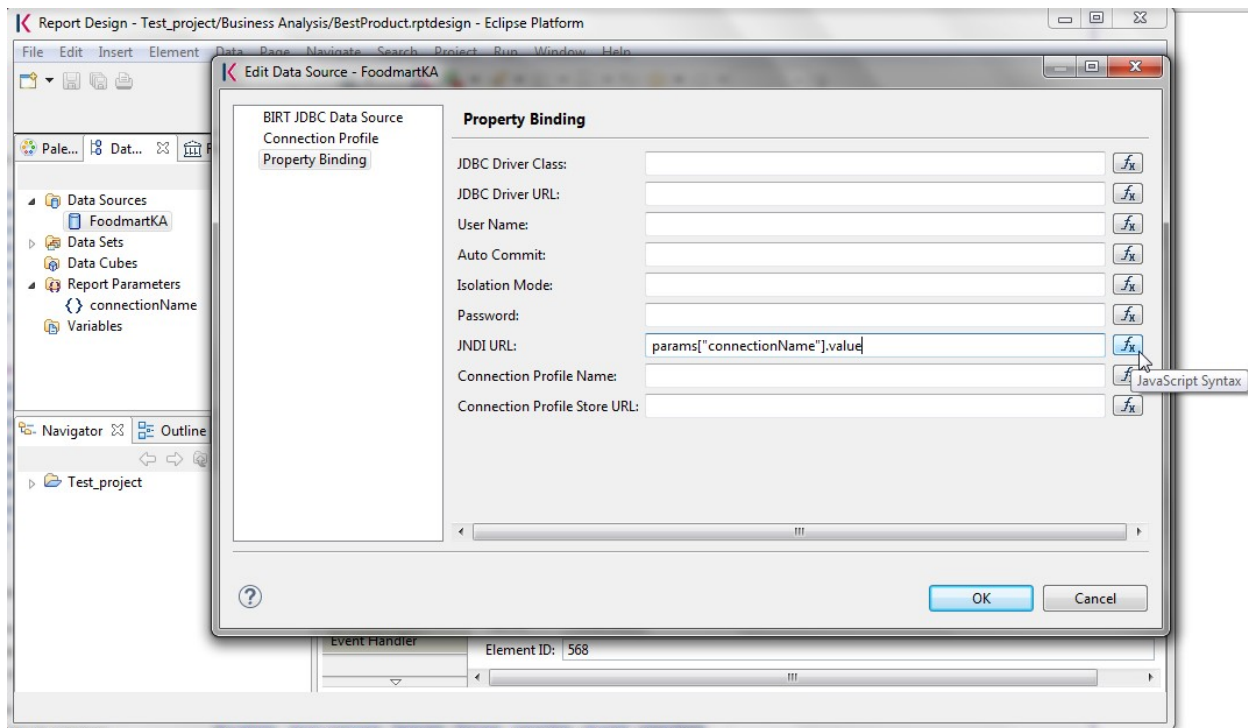


Fig. 5.300: Setting the connectionName parameter in the Data Source editor

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.

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

Note: Deploy on Knowage Server

To deploy the report you have to upload the template file directly in the document detail page.

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.

Note: BIRT Designer

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

5.10.1.1 Adding parameters to reports

Most times reports show data analysis that depend on variable parameters, such as time, place, type. Birt designer allows 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.

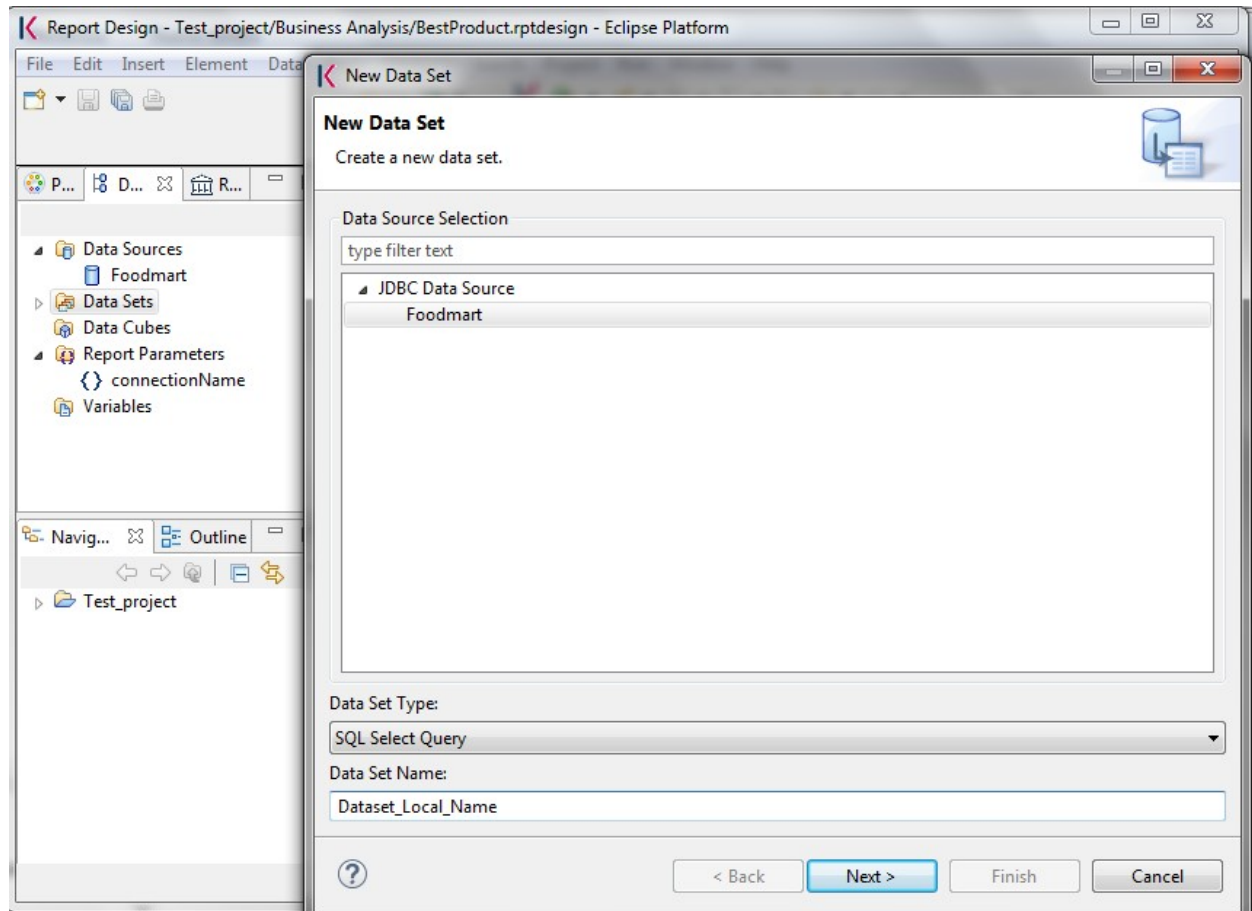


Fig. 5.301: Dataset definition.

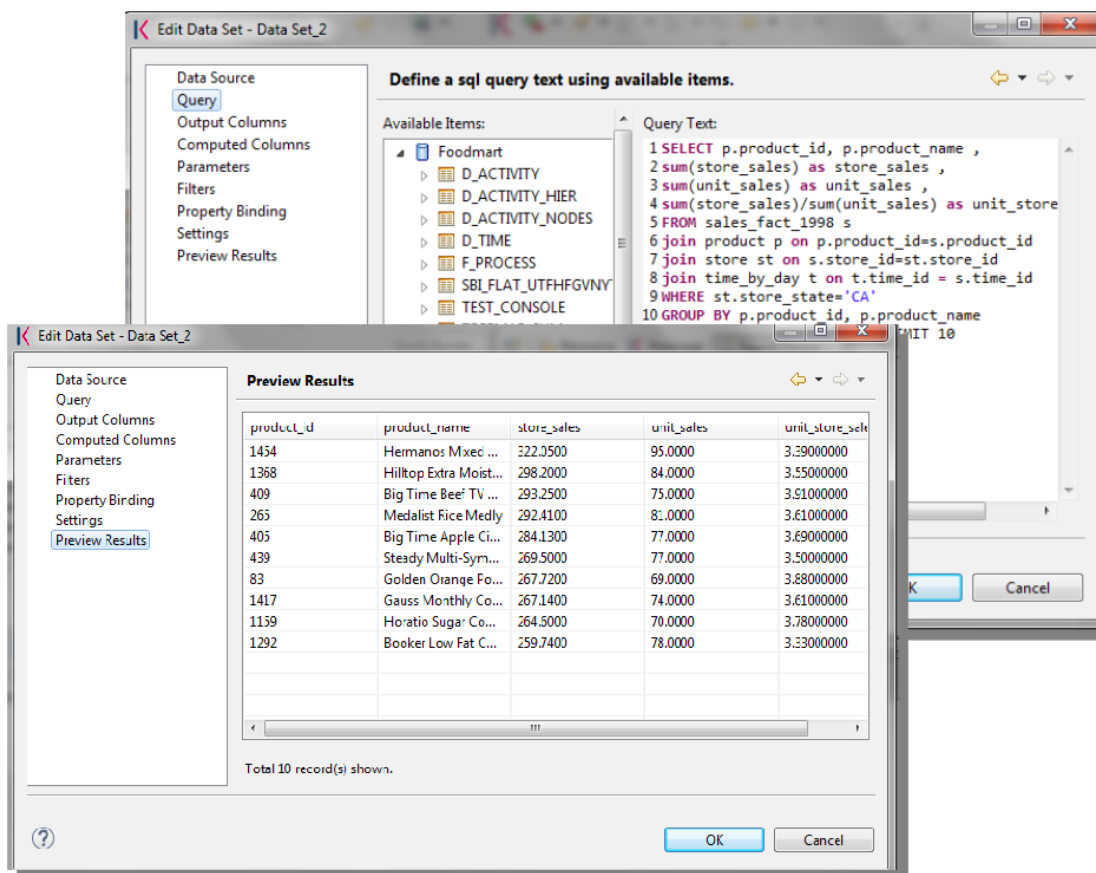


Fig. 5.302: Dataset editor, with preview.

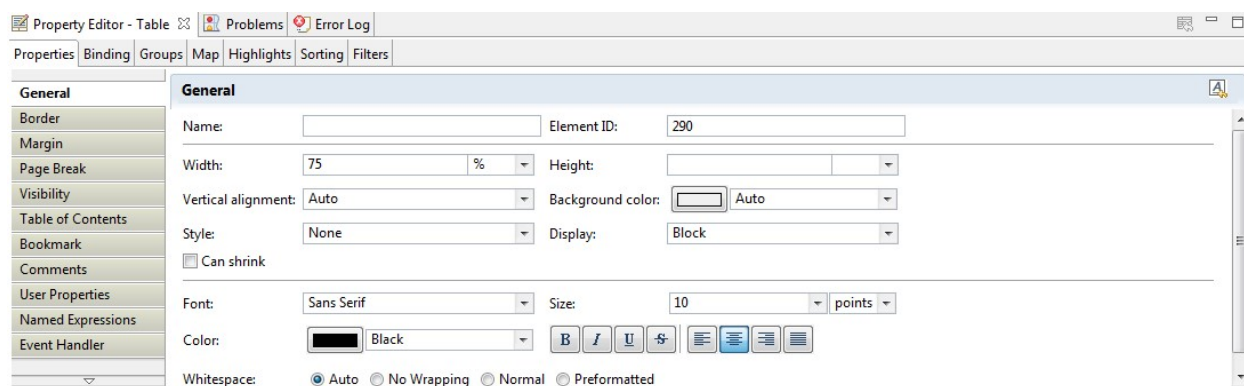


Fig. 5.303: BIRT Property Editor.

Warning: 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.

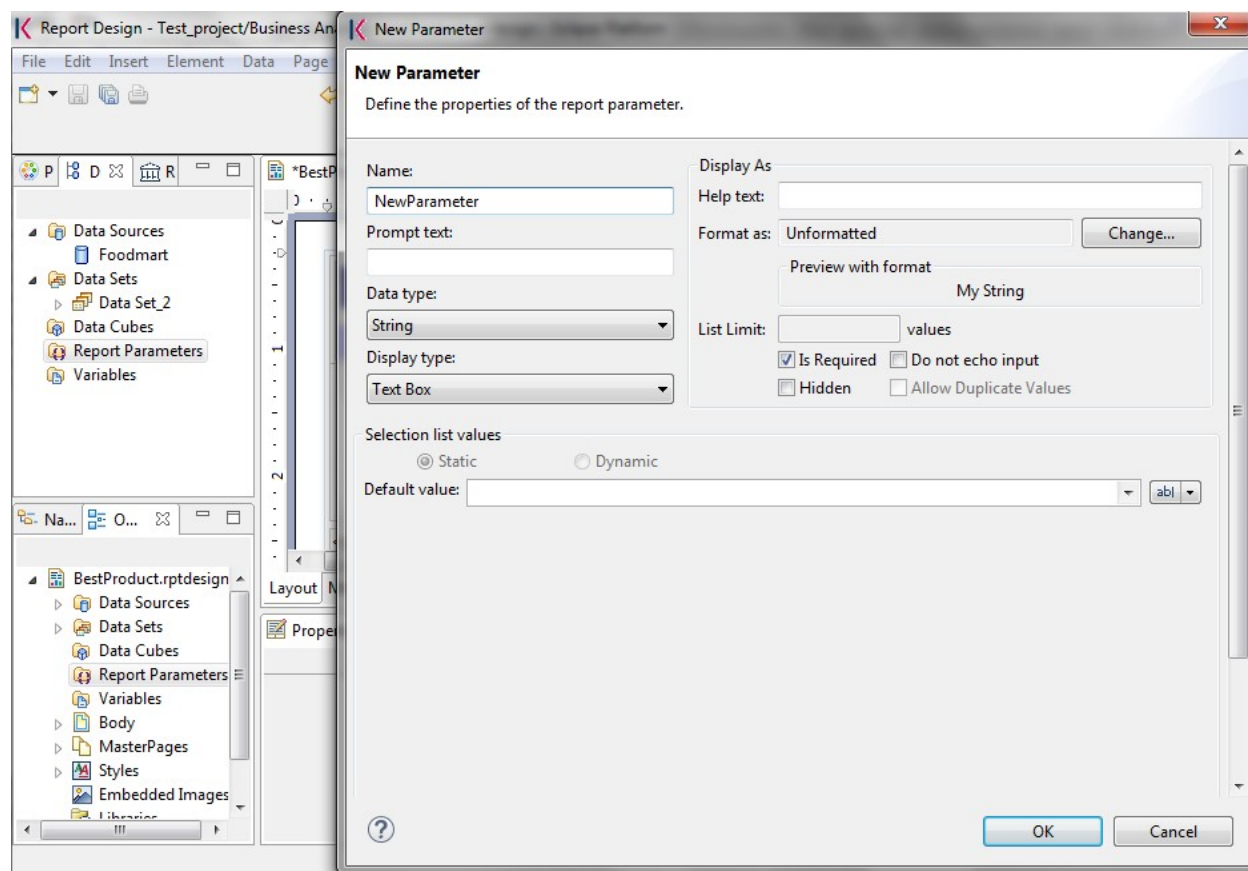


Fig. 5.304: Creation of a new parameter in a BIRT report.

Note that you must set a link for each question mark as shown below, even if the same parameter occurs multiple times in the same query.

Warning: Transfer reports from Birt designer to Server and vice versa

Any valid BIRT template can be directly uploaded in Knowage Server using the web interface for document management.

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

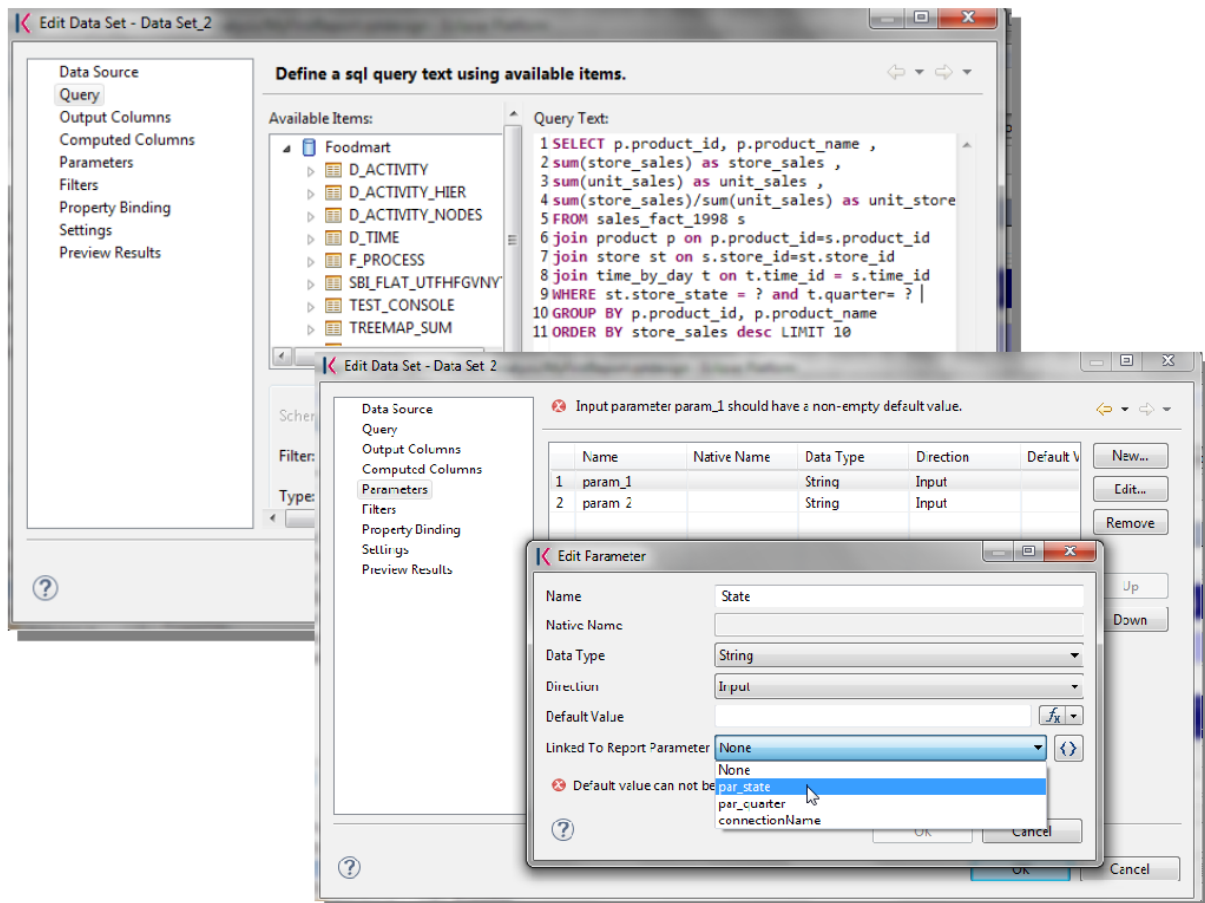


Fig. 5.305: Insert parameters into the dataset definition.

Listing 5.18: Parameters syntax

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

5.10.2 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 crossnavigation 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 another birt report, to a dashboard, a geo or any other analytical 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 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 designer. 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 below.

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 Cross Navigation syntax we give an idea of how the syntax should be like:

Listing 5.19: Cross Navigation syntax.

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

Warning: Type the right cross navigation name

It is important to underline that the “Cross_Navigation_Name” of Cross Navigation syntax is the cross navigation name related to the document and set using the “Cross Navigation Definition” feature we described in *Analytical Document* Chapter, *Cross Navigation* Section.

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 *Analytical Document* Chapter, *Cross Navigation* Section).

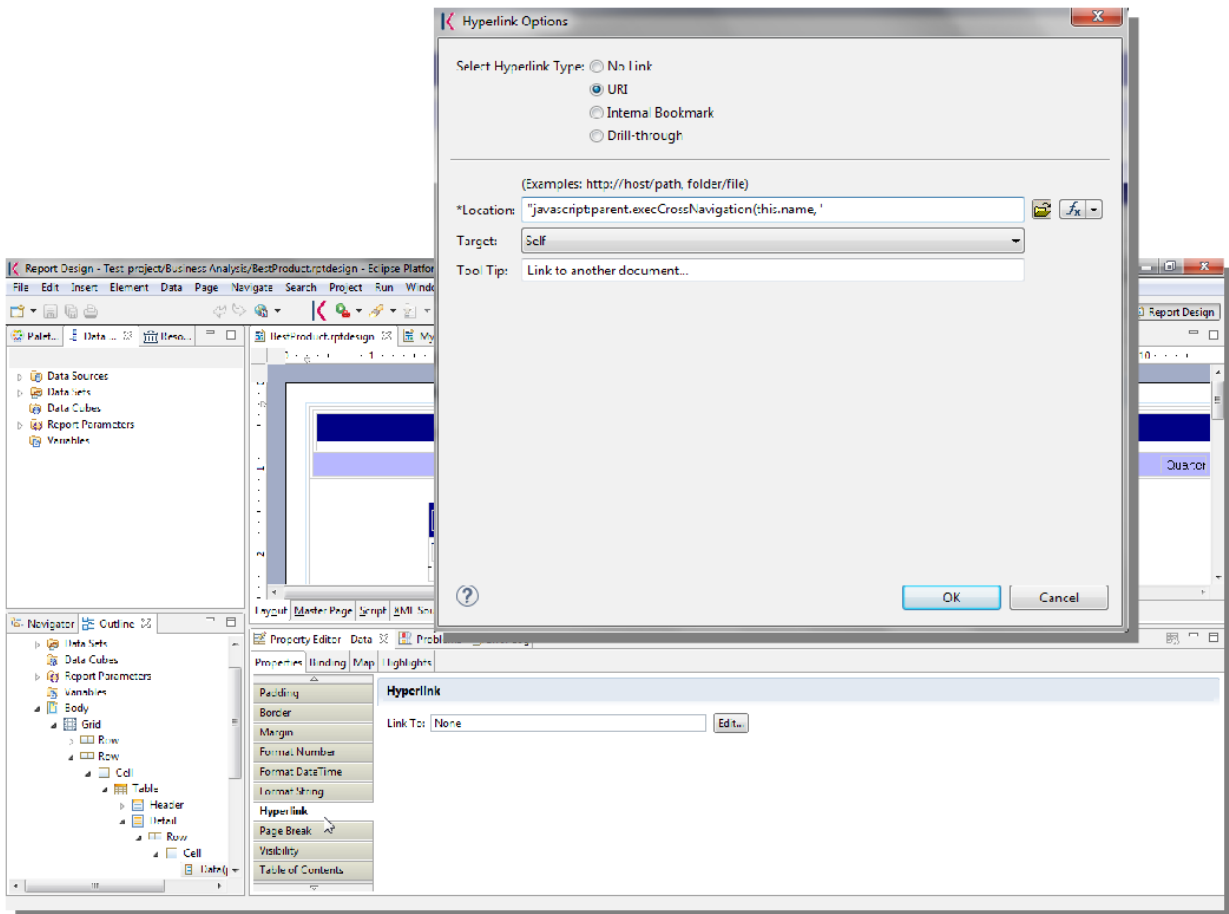


Fig. 5.306: Hyperlink editor.

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 below, and report parameters automatically.

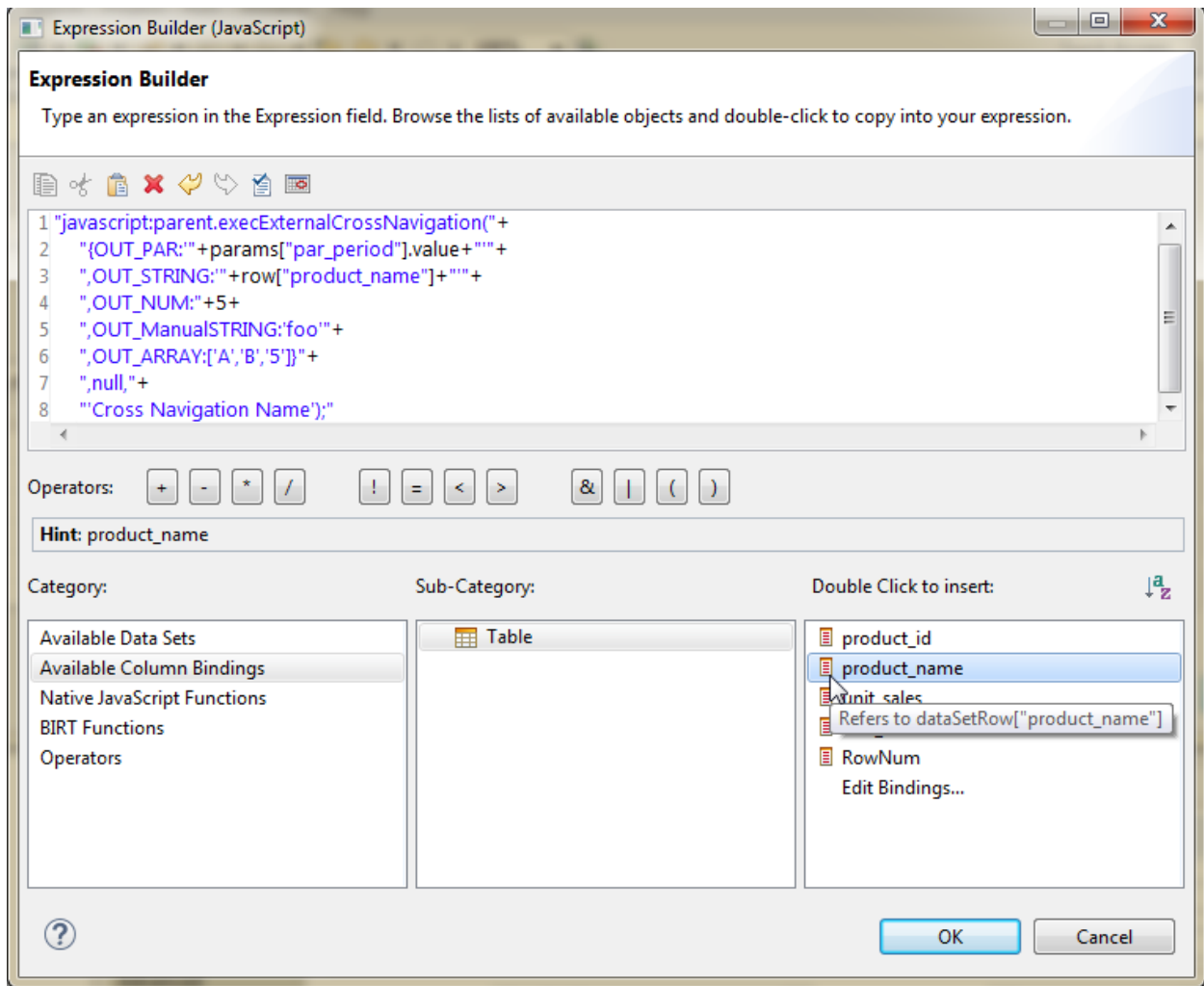


Fig. 5.307: Column bindings.

To manage multi-value parameters is enough to list all values between brackets separating them with commas, as reported in the code above. 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 *Cross Navigation* syntax code the “Cross_Navigation_Name” is left empty as in the code below, 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.

Listing 5.20: Cross Navigation syntax

```

1  "javascript:parent.execExternalCrossNavigation("+
2  "{OUT_PAR:''+params["par_period"].value+''"+
3  ",OUT_STRING:''+string_text+''"+
4  ",OUT_NUM:''+numberX+
5  ",OUT_ManualSTRING:'foo'+
6  ",OUT_ARRAY:['A','B','5']}"
7  ",null,"+
8  "''');"

```

5.11 Jasper reporting

Jasper is a stand-alone reporting tool developed by the Jaspersoft Community. An example of Jasper report is represented in the next figure. Jasper Report Engine manages Jasper report templates inside Knowage Server. A report template for Jasper is a text file with .jrxml extension that can be manually modified by very expert users by editing XML code. Otherwise, iReport, a graphical template designer, is provided for all developers who want to easily design a report for this engine.

Salary of VP Country Manager

Salary VP Country Manager						
	Name		Gender	Store Name	City	State
40000.00	Derrick Whelply	M	HQ	Alameda	CA	
	Michael Spence	M	HQ	Alameda	CA	
35000.00	Laurie Borges	F	HQ	Alameda	CA	
	Maya Gutierrez	F	HQ	Alameda	CA	
	Pedro Castillo	M	HQ	Alameda	CA	
30000.00	Beverly Baker	F	HQ	Alameda	CA	

Fig. 5.308: Example of a Jasper report.

Please note that this engine is available only in KnowageER.

5.11.1 Document definition*

Unlike the BIRT designer, iReport is not integrated in Eclipse. Therefore, before starting developing the report, you must download it from the website <http://community.jaspersoft.com/project/ireport-designer/releases>.

Note: iReport download

Download the iReport designer at <http://community.jaspersoft.com/project/ireport-designer/releases>.
Knowage support any kind of version.

Then you will have to set the path to the iReport designer in Knowage Studio. Open Window > Preferences > iReport Editor Configuration and set the correct path to the ireport.exe file.

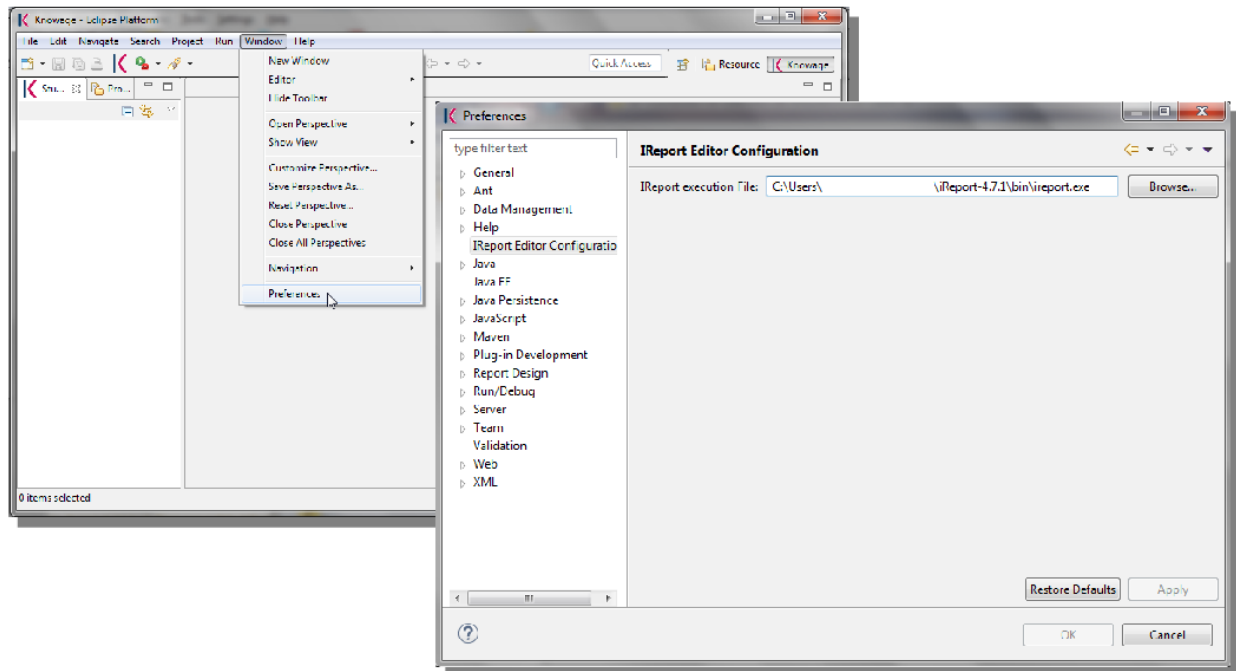


Fig. 5.309: iReport Configuration in Knowage Studio.


Similarly to the case of a BIRT report, the design and deployment of a Jasper report with Knowage Studio consists of 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 Jasper report, right-click on the **Business Analysis** folder and select **Report > Report with Jasper**. This will open an editor where you can choose a name for your document. The new document will be created under the Business Analysis folder.

Double click on the report to open the editor: the iReport editor will be launched inside Knowage Studio. Now you are ready to start to develop your report.

The next steps consist in the creation of a datasource and of a dataset. As described in section “Dataset definition”, Knowage Studio allows the development of analytical documents using either internal or external datasets. In this example we will show how to create a report with an internal dataset.

Click on the small icon  of the menu bar. The data source creation editor will open.

Select a JDBC data source, set the appropriate parameters and give the data source a name. Figure below shows the wizards that present in this procedure.

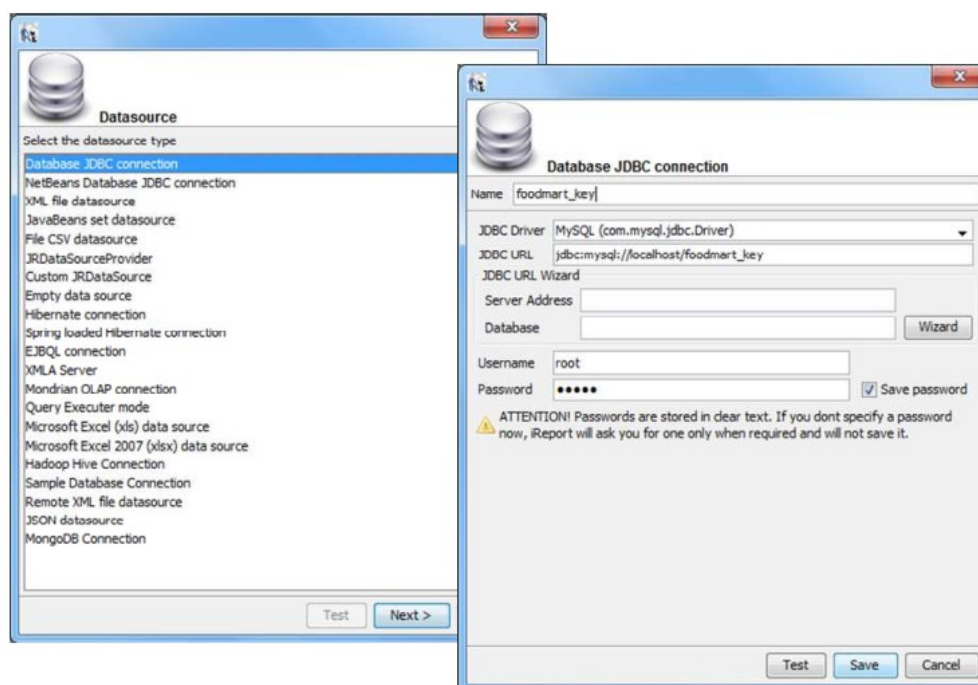



Fig. 5.310: Creation of a JDBC data source in a Jasper report.

Once you have defined the data source, create your dataset. Note that Jasper only allows one main dataset. By the way, secondary datasets can be added if needed. Right-click on the report item and select **Add dataset**. The dataset editor will guide you through the dataset definition. You can either manually edit the SQL query or use the design query tool provided by iReport.

The tree located in the left part of the window shows the elements of the report. On the right, the **Palette** shows all graphical elements you can add. You can choose to see your report within the **Designer**, to inspect the **XML** code or

to look at the **Preview** of your report. If you click on the  icon you can edit and preview your query. As you can see, the iReport designer allows the creation of complex reports, with different graphical elements such as cross tabs, charts, images and different text areas. We see now shortly how to design a very simple report, i.e. a report containing a table showing data from the defined dataset.

To create a table, click on the element in the Palette and use the wizard. Figure below exhibits the Jasper editor interface.

To insert dataset columns into the report, use the syntax showed in following syntax to insert dataset columns.

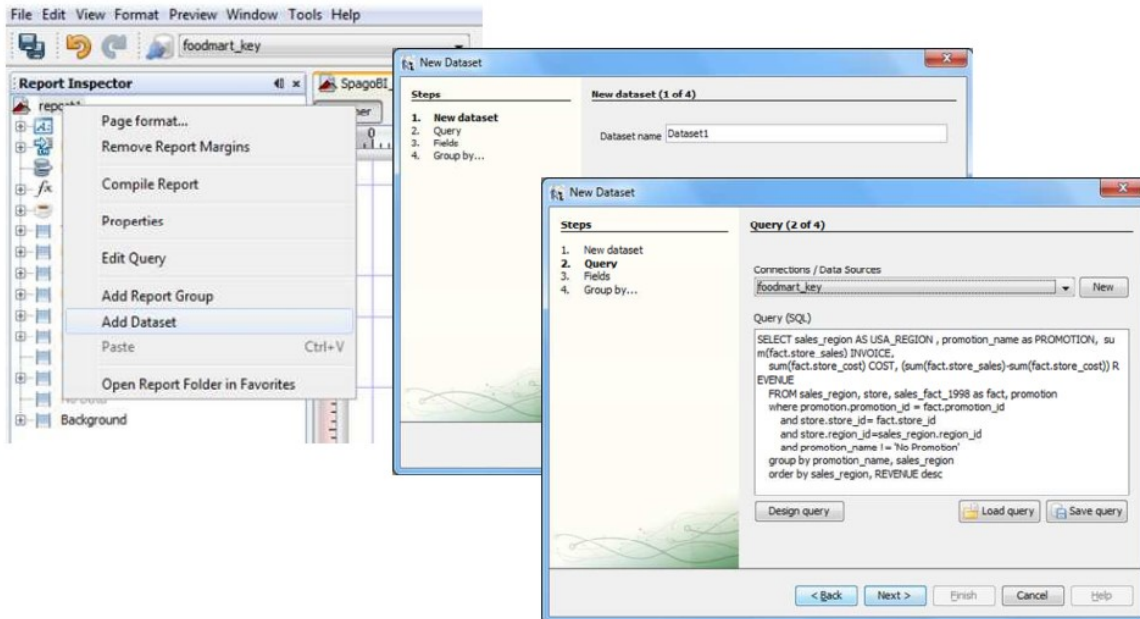


Fig. 5.311: Data Set definition.

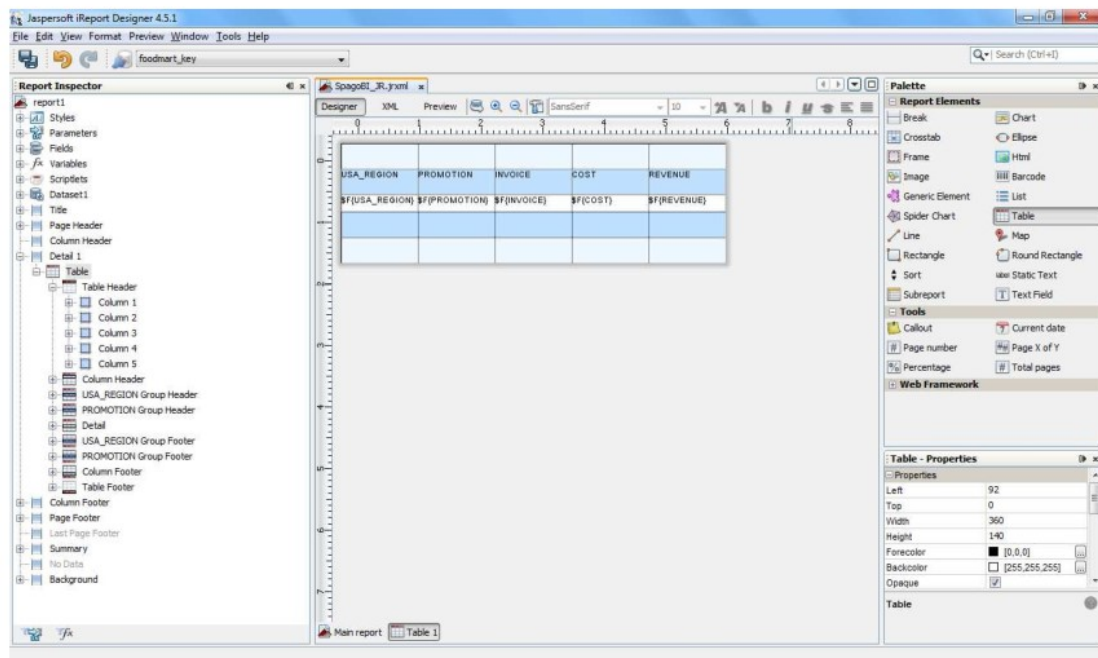


Fig. 5.312: iReport graphical editor.

Listing 5.21: Syntax to insert dataset columns.

```
$F{name_of_dataset_column}
```

Once the document has been developed, technical users can use Knowage Studio deployment service to easily register the report with its template on Knowage Server. Alternatively, any valid Jasper template (developed with or without Knowage Studio) can be directly uploaded to Knowage Server using the web interface for the document management.

This section does not provide any further detail about graphical development since it focuses on specific aspects of Knowage Jasper Report Engine. All Jasper standard functionalities work with Jasper Report Engine. For a full overview of Jasper reporting tool and a detailed developer guide, please refer to the official documentation at <http://community.jaspersoft.com/>.

5.12 Location Intelligence

Location intelligence is based on the idea that geographical spaces are a particular analytical dimension in the BI domain. It is based on:

- the geographical representation of data,
- interaction with GIS systems,
- spatial data,
- spatial operators.

Location Intelligence usually guarantees:

- an immediate perception of a phenomena distribution over a geographical area,
- interactivity,
- multivariate analysis,
- temporal snapshots.

Location Intelligence is becoming widely used, mostly thanks to the emergence of location services such as Google Maps. This domain is very easy to use for all kinds of users, usually analysts and operational profiles. By contrast, its management is not as easy, especially if it implies an internal management of the geographical data base.

5.12.1 Basic concepts

The term Location Intelligence refers to all those processes, technologies, applications and practices capable to join spatial data with business data, in order to gain critical insights, to better support decisional processes and to optimize business activities.

At the technological level, this correlation is the result of the integration between the software systems that manage these two heterogeneous types of data: geographic information systems (GIS), which manage spatial data, and Business Intelligence systems (BI), which manage business data. This integration gives rise to new technological tools supporting decision-making processes, and the analysis on those business data that are directly or indirectly related to a geographic dimension.

Location Intelligence applications significantly improve the quality of users' analysis based on a geographic dimension. Indeed, a Data Warehouse (DWH) almost always included such information. By representing the geographic distribution of one or more business measures on interactive thematic maps, users can quickly identify patterns, trends or critical areas, with an effectiveness that would be unfeasible using traditional analytical tools.

5.12.2 More on GIS and Spatial Data*

5.12.2.1 Spatial Data

The term *spatial data* refers to any kind of information that can be placed in a real or virtual geometric space. In particular, if the spatial data is located in a real geometric space — which is a geometric space that models the real space — it can be defined as *geo-referenced data*.

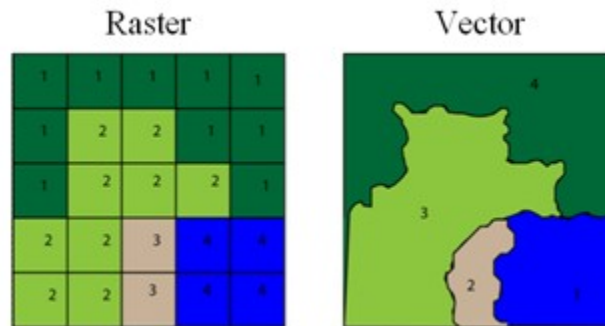


Fig. 5.313: A base layer in raster and vector format.

Spatial data are represented through graphical objects called maps. Maps are a portrayal of geographic information as a digital image file suitable for display on a computer screen.

According to the *Open Geospatial Consortium* (OGC) definition, a map is made of overlapping *layers*: a *base layer* in raster format (e.g. satellite photo) is integrated with other layers (*overlays*) in vector format. Each overlay is made of homogeneous spatial information, which models a same category of objects, called *features*.

A feature is called *geographic feature* when the constituting objects are abstractions of real-world physical objects and can be located univocally within a reference coordinate system, according to their relative position.

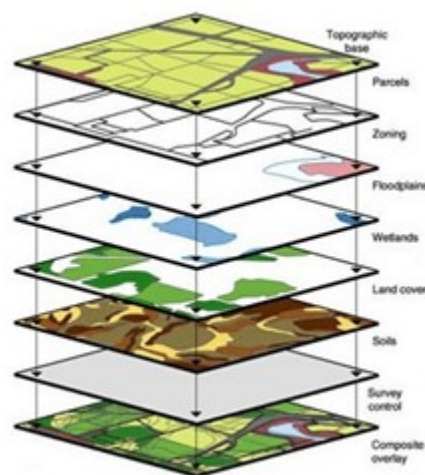


Fig. 5.314: Overlapping layer.

A feature includes:

- a set of attributes that describes its geometry (vector encoding). Geometric attributes must describe its relative shape and position in an unambiguous way, so that the feature can be properly drawn and located on the map, according to the other features of the layers.

- a set of generic attributes related to the particular type of physical object to be modeled. Generic attributes are not defined: they vary according to the type of abstraction that users want to give to each real-world physical object.

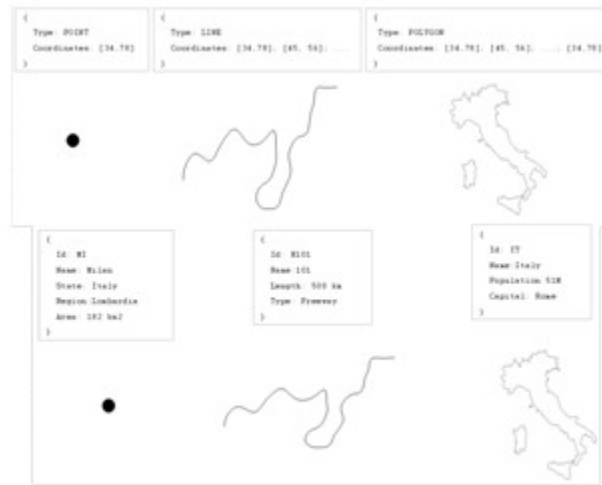


Fig. 5.315: Examples of feature.

There is a wide range of standards that can be used for the vector encoding of spatial data (e.g. GeoJSON, GML, Shape File, etc.). Most geographic information systems can perform the needed conversions among various encodings.

5.12.2.2 GIS

Geographic Information Systems (GIS) provide a set of software tools designed to capture, store, extract, transform and display spatial data [2]. Therefore, the term GIS refers a set of sole technological components that manage the spatial data during its whole life cycle, starting from the capture of the data up to its representation and re-distribution.

From a logical point of view, the key functionalities of a GIS do not differ from those of a BI system. Both systems are characterized by some specific components supporting the effective storage of data, some others supporting their manipulation, their re-distribution or their visualization. On the other hand, the implementation of these functionalities deeply differs between GIS and BI systems, since they deal with two different types of data (alphanumeric and spatial data).



Fig. 5.316: Definition of GIS, BI, spatial data and business data.

Unlike the market of BI suites, the market of GIS is characterized by a wide spread of open standards, adopted by all main vendors, which regulate the interaction among the various components of the system at all architectural levels.

Note: Open Geospatial Consortium (OGC)

The most important International organization for standardization in the GIS domain is the Open Geospatial Consortium (OGC), involving 370 commercial, governmental, non-profit and research organizations. Read more at www.opengeospatial.org.

As for the integration between GIS and BI systems, the OGC has defined two main standards supporting the re-distribution of the spatial data:

- the *Web Map Service* (WMS). It describes the interface of services that allow to generate maps in a dynamic way, using the spatial data contained in a GIS.
- the *Web Feature Service* (WFS). It describes the interface of services that allow to query a GIS, in order to get the geographic features in a format that allows their transformation and/or spatial analysis (e.g. GML, GeoJson, etc.).

Note: WMS and WFS standards for spatial data distribution

Full documentation about the WMS and WFS standards can be found at www.opengeospatial.org/standards/wms and www.opengeospatial.org/standards/wfs.

Knowage suite offers an engine supporting the Location Intelligence analytical area, the **GEOReport Engine**, generating thematic maps.

5.12.3 Analytical document execution

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

In Figure below 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.

Click on the arrow on the top right to open the Location Intelligence 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:

- **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 below.

Now you can add extra layers on the default one. Switch to the **layer** tab of the Location Intelligence 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 below.

In our example we upload some waypoints, you can see the results obtained in next figure.

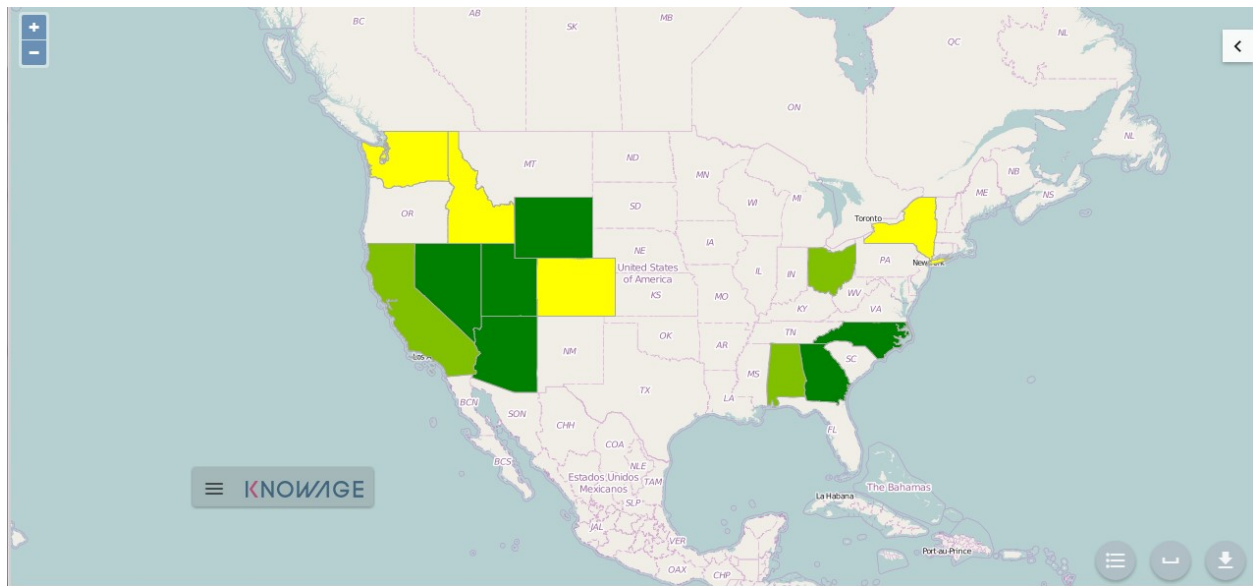


Fig. 5.317: Example of GIS document. USA sales per store

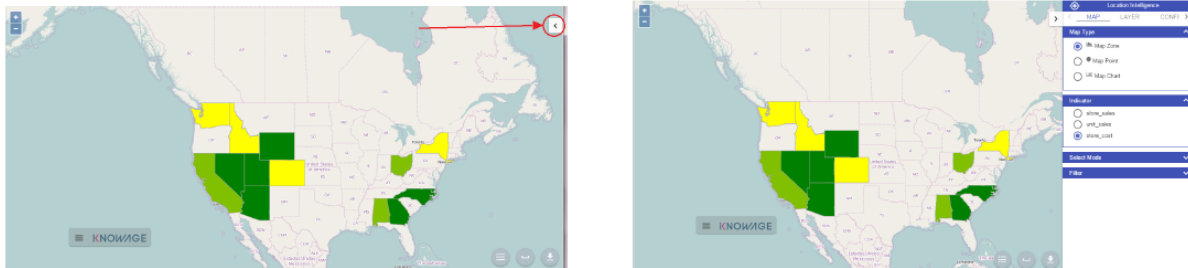


Fig. 5.318: Arrow button (left) Location Intelligence options panel (right) .

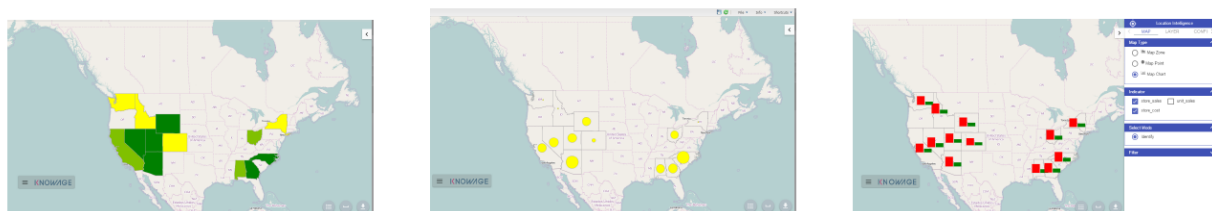


Fig. 5.319: Map Zone (left) Map Point (center) and Map Chart (right).

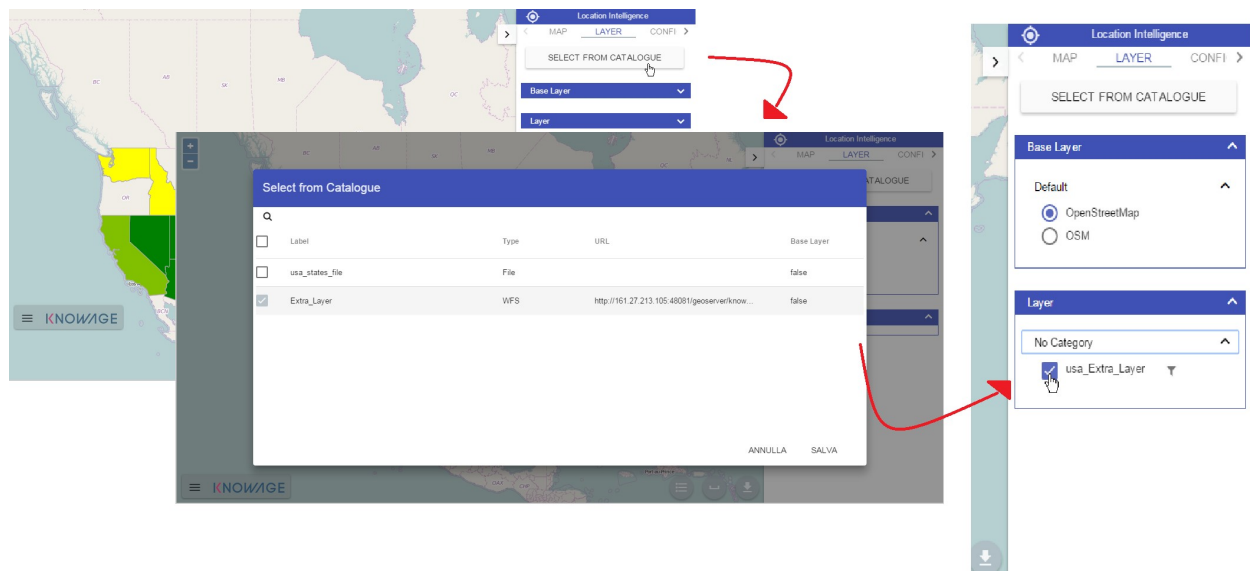


Fig. 5.320: Steps for layer adding

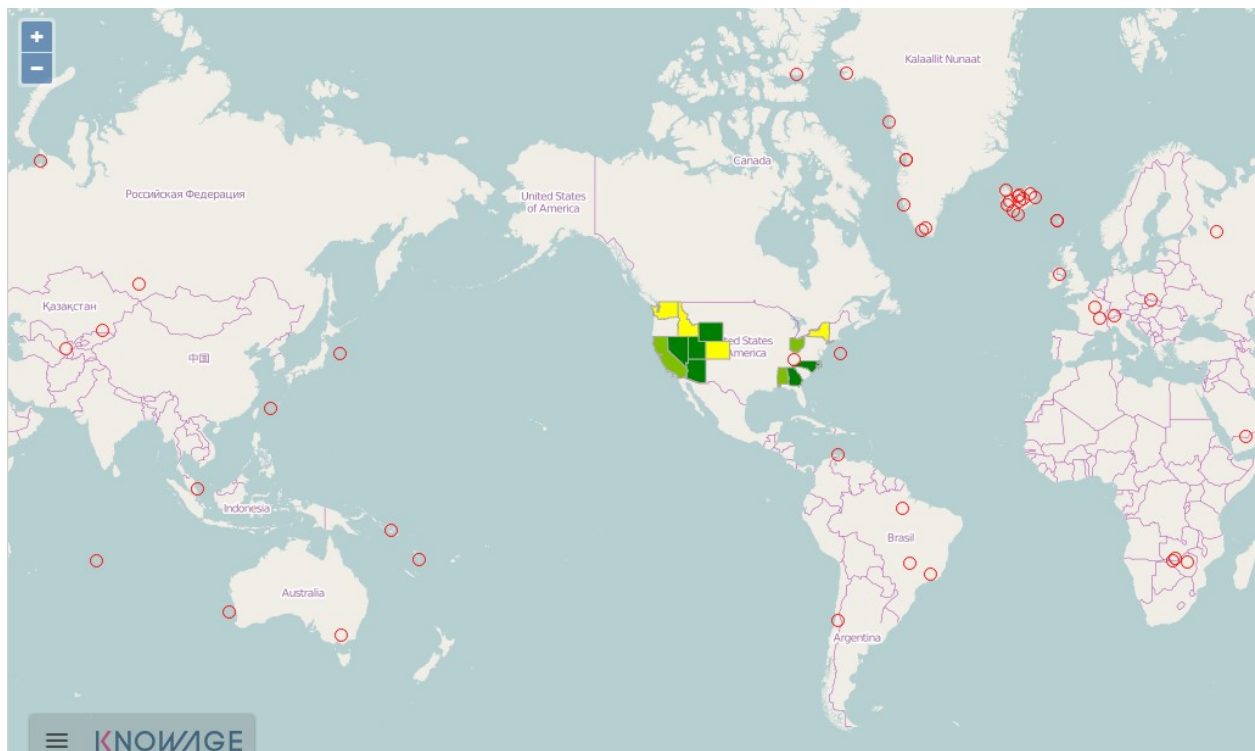


Fig. 5.321: Map with two layers

Now let's focus on **Configuration** tab of Location Intelligence 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 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 below.

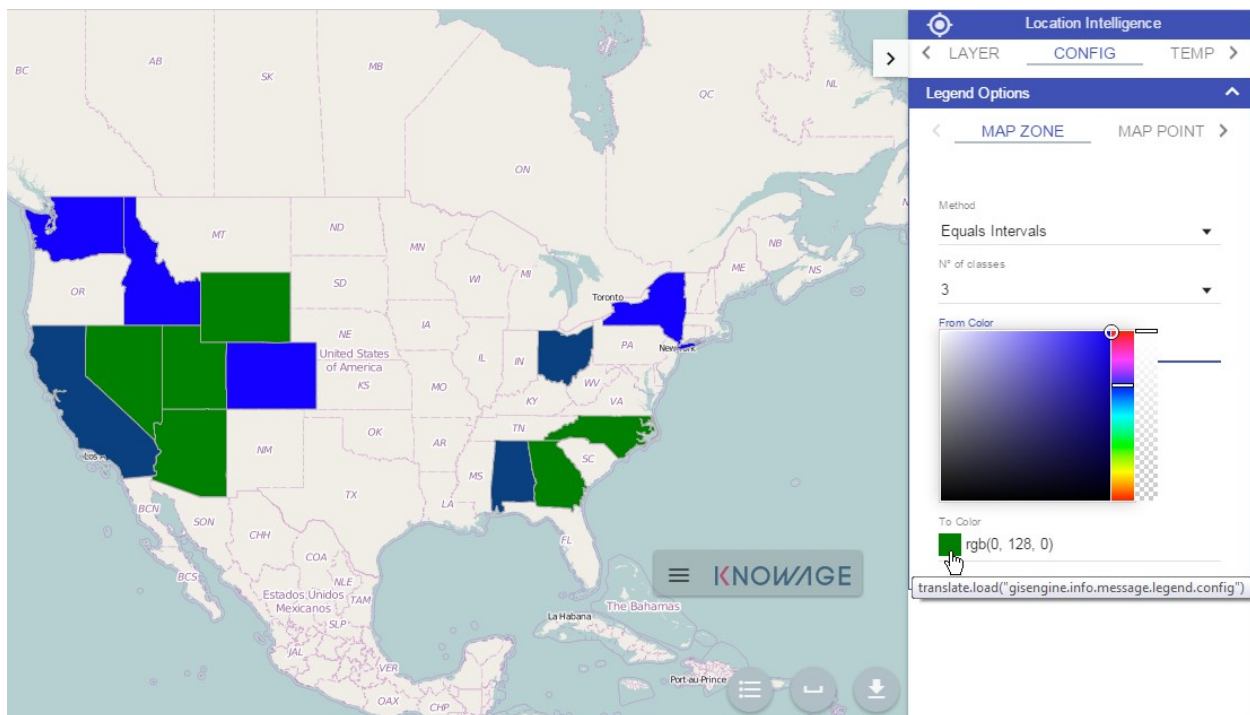


Fig. 5.322: Map Zone extra configurations

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 buttons located at the bottom right corner, you can see them underlined in the following figure. From the left to the right this buttons 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.

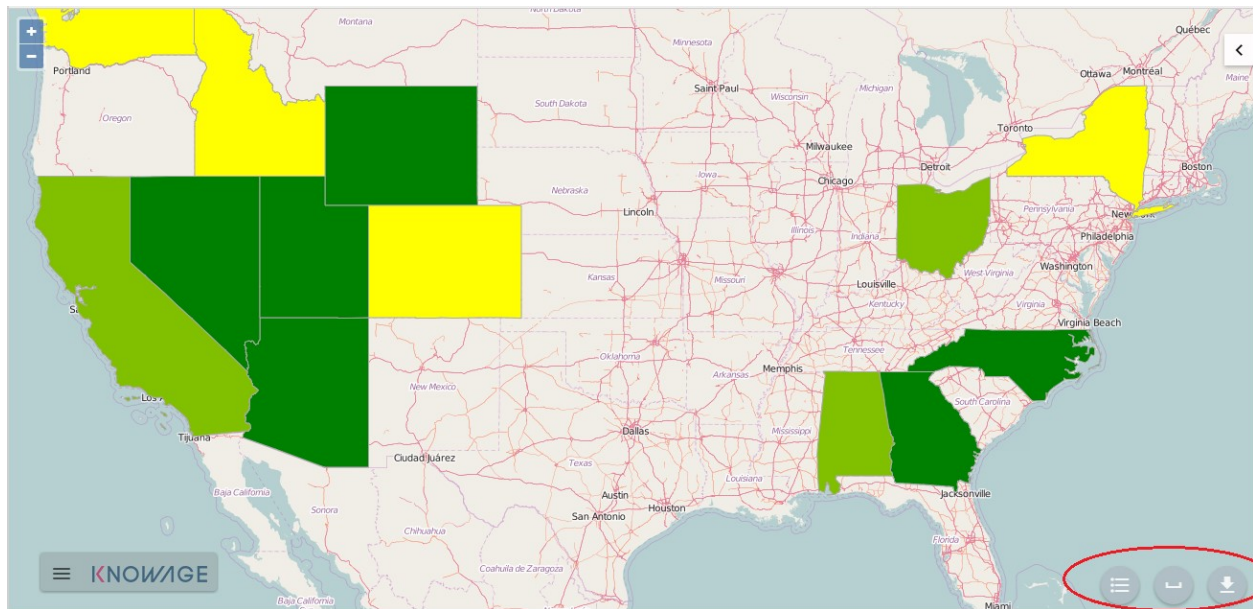


Fig. 5.323: From the left to the right: Legend, Measure and Export bottom.

5.12.3.1 Extra functionalities

Let's come back to Location Layer main tab and 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 button to perform the cross navigation with the active selection.

5.12.4 Template building with GIS designer

GIS engine document templates can now be built using GIS designer. Designer is available from administrator document detail page (for this part refer to Section 15.8) 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**.

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

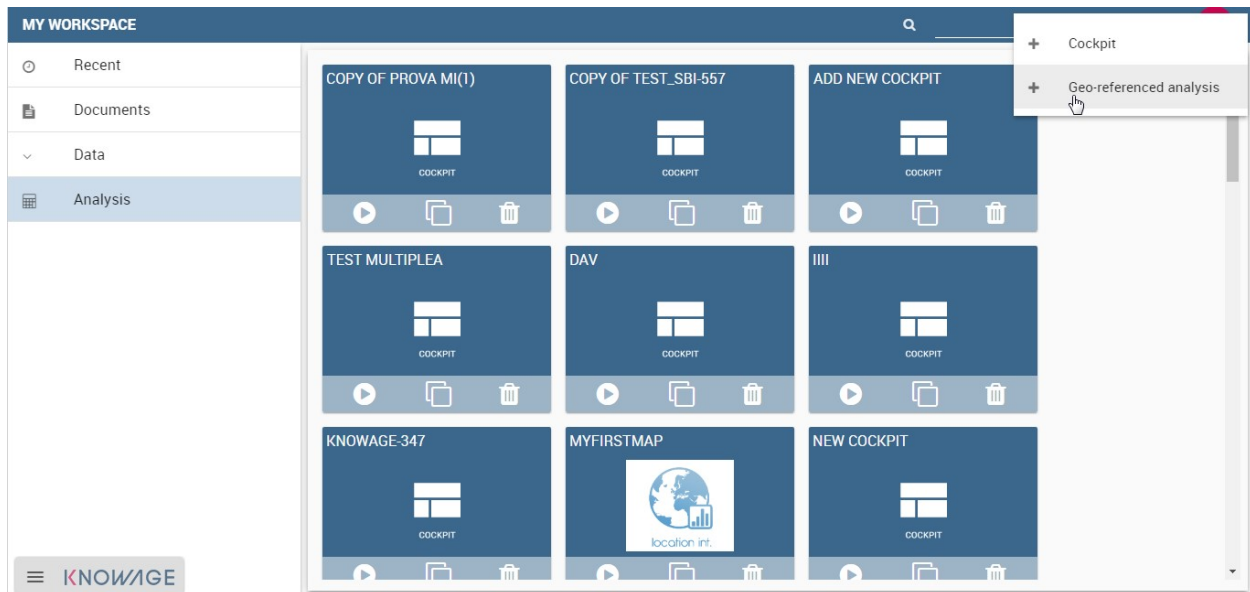


Fig. 5.324: Start a new Geo-referenced analysis.

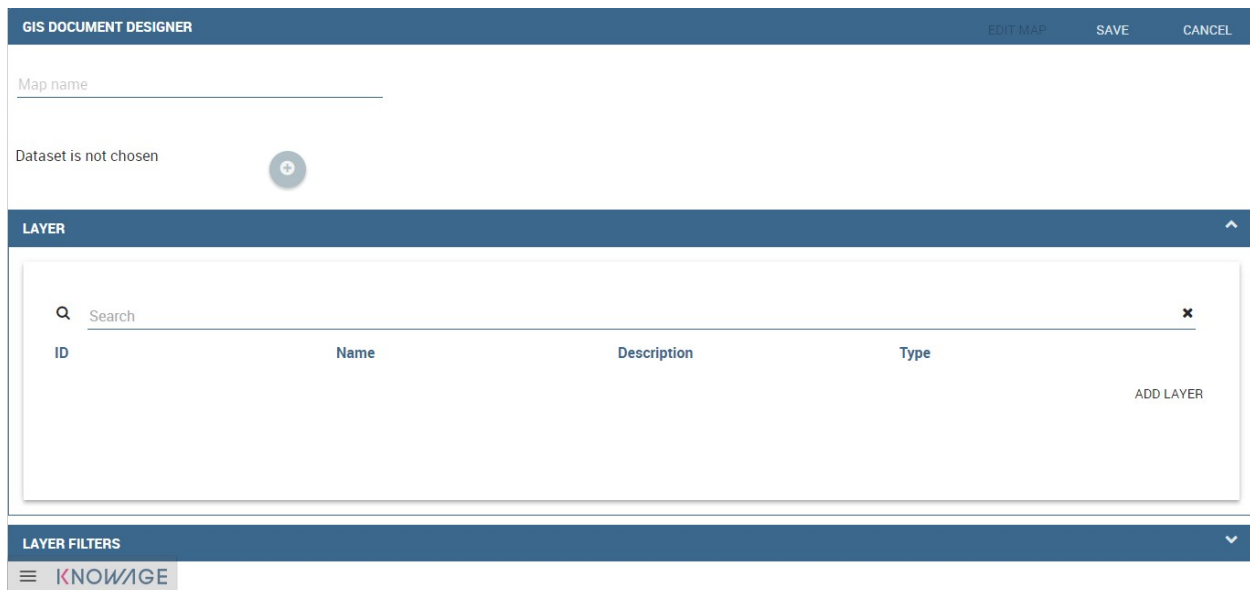


Fig. 5.325: GIS document designer window.

5.12.5 Designer sections

5.12.5.1 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 (next figure) opens a pop up with a list of all available layer catalogs. Selecting one item from the list and clicking save the selected item will be chosen for template.



Fig. 5.326: Target layer definition.

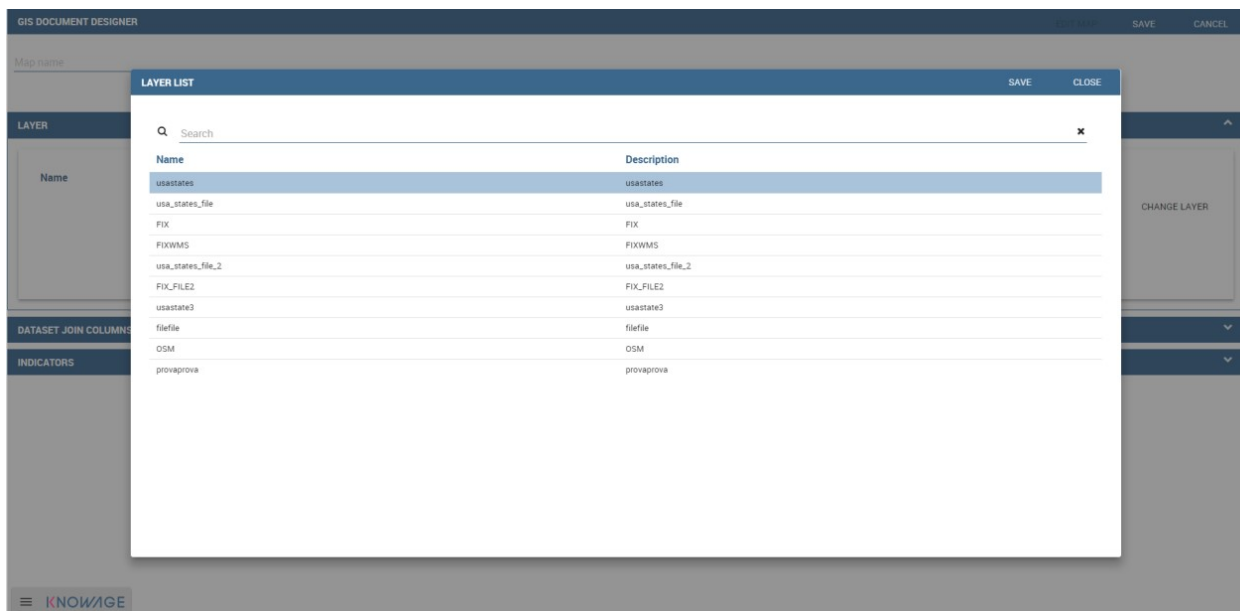


Fig. 5.327: List of available layer catalogs.

In case when there is no dataset multiple layers can be selected below.

5.12.5.2 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 figure below 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.

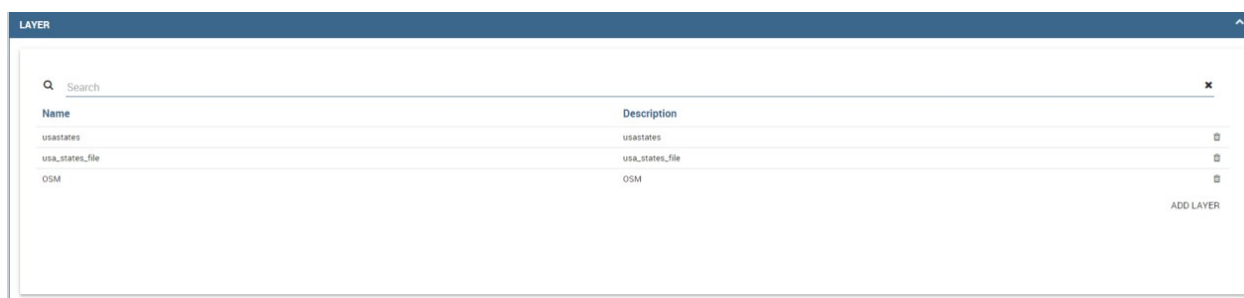


Fig. 5.328: Multiple selection of available layers.

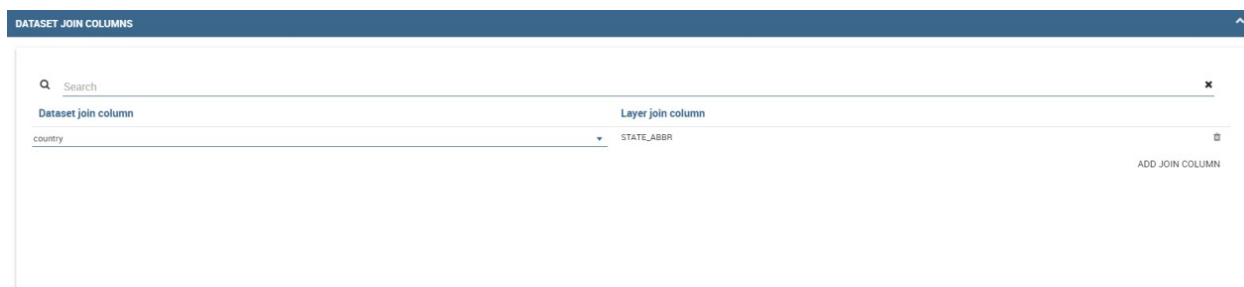


Fig. 5.329: Dataset join columns interface.

5.12.5.3 Indicators

Measures definition is configurable by adding indicators. The interface is shown below. 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 field 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.



Fig. 5.330: Indicators interface.

5.12.5.4 Filters

Using the filtering dedicated area, as shown in figure below, 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.



Fig. 5.331: Filters interface.

5.12.5.5 Map menu configuration

Through the **Map menu configuration** panel the user can decides to enable or disable some available functions and features, like the legend, the distance calculator and so on. See next figure to have a glimpse at the available items.

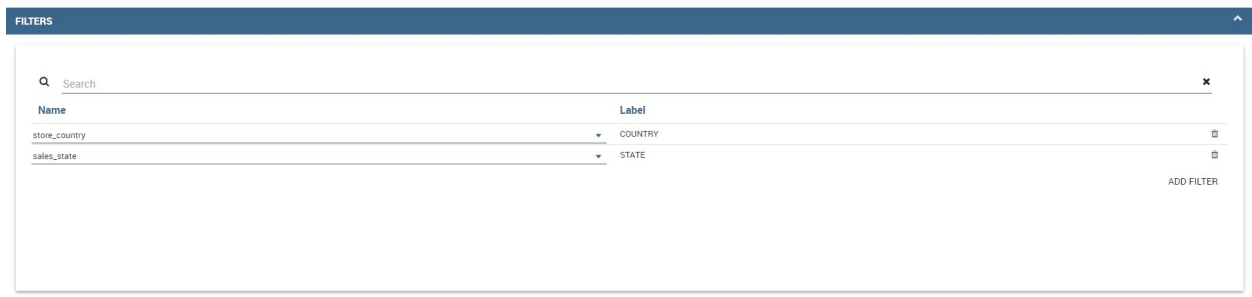


Fig. 5.332: Map menu configuration.

5.12.5.6 Layer filters

Here, as you can see from figure below, you define which target layer attributes can be used to filter the geometry. This section is only present when a dataset has been selected. Add filters button opens pop up where you can choose all available filters of the selected layers. Figure below gives an example.



Fig. 5.333: Layer filters interface.

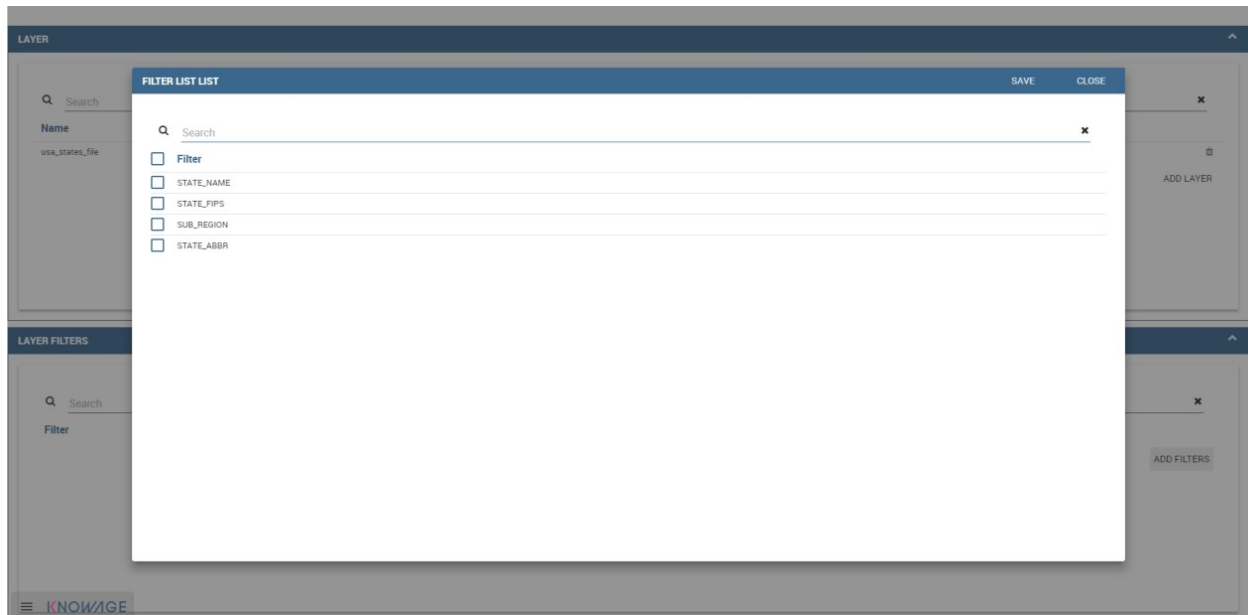


Fig. 5.334: List of available filters.

5.12.5.7 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 the following figure. When the template is saved successfully 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 below. In edit mode you are able to save all custom setting made on map.

5.12.6 GEOReport Engine*

The **GEOReport Engine** implements a *bridge integration* architecture.

Generally speaking, a bridge integration involves both the BI and the GIS systems, still keeping them completely separated. The integration between spatial data and business data is performed by a dedicated application that acts as a *bridge* between the GIS and the BI suite. This application extracts the spatial data from the GIS system and the business data from the BI suite, to answer the users' requests. Afterwards, it joins them and provides the desired results.

In particular, the **GEOReport Engine** extracts spatial data from an external GIS system and join them dynamically with the business data extracted from the Data Ware House, in order to produce a thematic map, according to the user's request. In other words, it acts as a *bridge* between the two systems, which can consequently be kept totally decoupled.

The thematic map is composed of different overlapping layers that can be uploaded from various GIS engines at the same time. Among them just one layer is used to produce the effective thematization of the map: this is called *target layer*.

You can manage your layers inside the **Layers Catalogue**.

Here you can upload the following layer types:

- File;

SAVE NEW MAP
SAVE
CLOSE

Name
geo

Description

Fig. 5.335: interface for name and description of new geo document for end user.

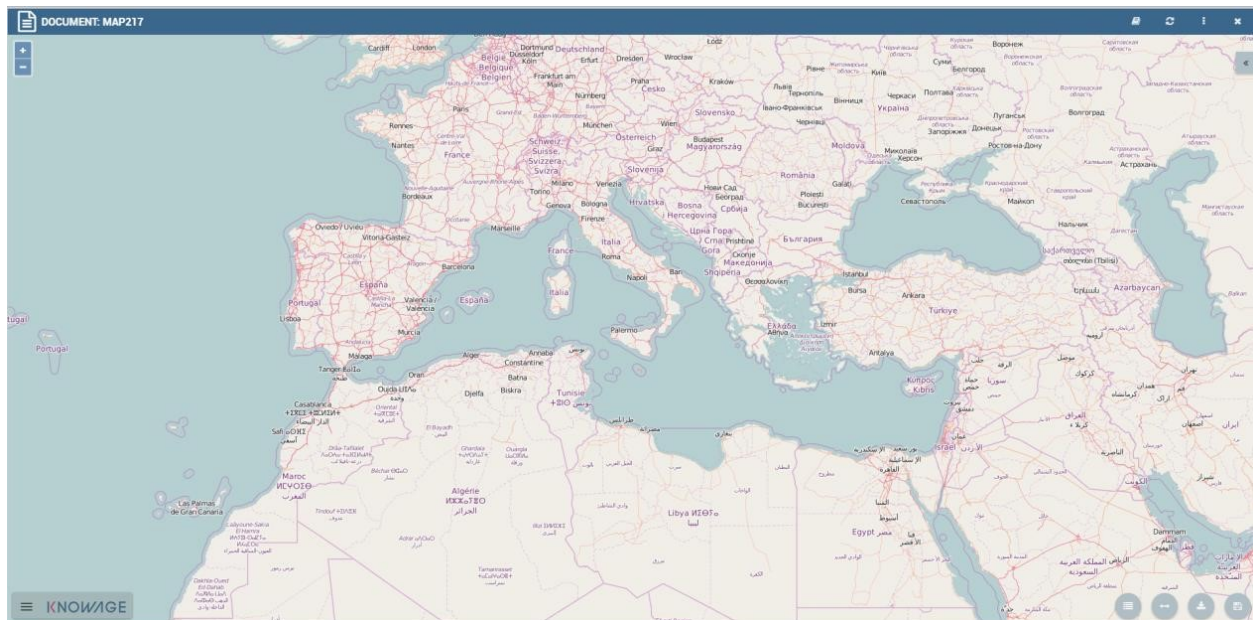


Fig. 5.336: Map in edit mode with save template available.

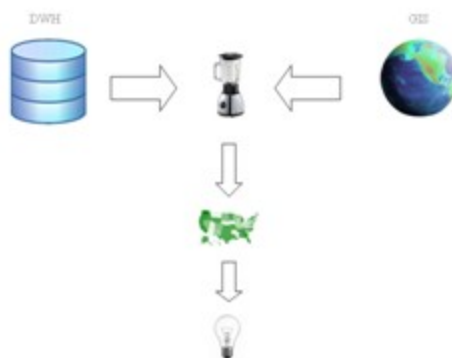


Fig. 5.337: Bridge integration architecture of the **GEOReport Engine**.

- WFS;
- WMS;
- TMS;
- Google;
- OSM.

Create a new layer clicking on the dedicated plus icon. On the right side you are asked to fill few settings before saving the new layer. Among these settings the firsts are equals for all types of layers. Once you choose the layer type, instead, some fields may change. This happens to manage all layers types from the same interface. For example if you choose **File** as type you have the possibility to choose your own .json file and upload it. After having done this, the path where your file is been uploaded is shown among the setting.

If you chose **WFS** or **WMS** you are asked to insert a specific url.

At the bottom part of layer configuration you can manage the layer visibility. Mark the role you want to give visibility privileges on this layer. If none is marked, the layer is visible to all role by default.

Once you have set all layer configuration you can switch to filter setting. Click on the tab you can find in the upper part of the screen, see the following figure.

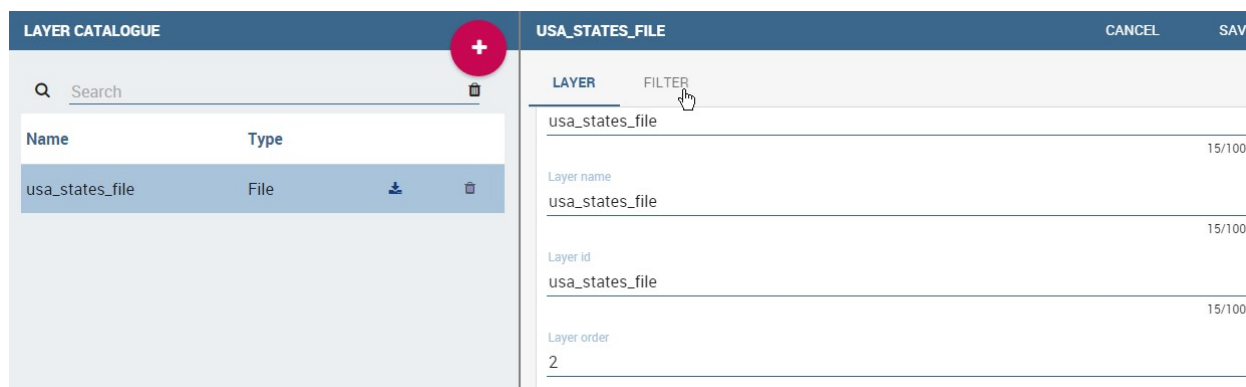


Fig. 5.338: Filter tab

Here you can choose which filters will be active during visualization phase. Choose among the properties of your layer, the available ones are only the string type.

Now you need to have a well-configured dataset to work with the base layer. The dataset has to contain one column matching a property field as type and contents otherwise you will not be able to correctly visualize your data on the map.

For example you can use a query dataset, connected to the foodmart data source, whose SQL query is shown in Code15.1.

Listing 5.22: GeoJSON file except.

```

1 SELECT r.region_id as region_id
2     , s.store_country
3     , r.sales_state as sales_state
4     , r.sales_region
5     , s.store_city
6     , sum(f.store_sales) + (CAST(RAND() \*60 AS UNSIGNED) + 1) store_sales
7     , avg (f.unit_sales)+(CAST(RAND() \* 60 AS UNSIGNED) + 1) unit_sales
8     , sum(f. store_cost) store_cost
9 FROM sales_fact_1998 f
10     , store s
11     , time_by_day t
12     , sales_region r
13 WHERE s.store_id=f.store_id
14 AND f.time_id=t.time_id
15 AND s.region_id = r.region_id
16 AND STORE_COUNTRY = 'USA'
17 GROUP BY region_id, s.store_country,r.sales_state, r.sales_region, s.store_city

```

Create and save the dataset you want to use and go on preparing the document template.

5.12.7 Template building with GIS designer for technical user*

When creating new location intelligence document using GIS engine basic template can be build using GIS designer interface. For administrator designer opens from document detail page clicking on build template button (refer to next figure). When the designer is opened the interface for basic template build is different depending on if the dataset is chosen for the document or not.

We have already described the Gis Designer when it is accessed by a final user. Since the difference relies only in how the designer is launched we will not repeat the component part and recall to *Designer section* paragraph for getting details. By the way we highlight that there is a last slight difference when defining a filter on layers. In fact, using the administrator interface, if the document has analytical driver parameters, you can also choose one of the available parameters to filter the geometry, as shown below. It is not mandatory to choose layer filters so you can also save the template without any filter selected.

When the list of selected layers is changed the filter list will be empty so you have to select filter list after filling the layer list, this is the way designer keeps consistency between layers and corresponding filters (see next figure).

5.12.8 Cross navigation definition*

It is possible to enable cross navigation from a map document to other Knowage documents. This means that, for instance, clicking on the state of Texas will open a new detail documents with additional information relative to the selected state.

You need to define the output parameters as described in Section *Cross Navigation of Analytical Document* Chapter. The possible parameters that can be handled by the GIS documents are the attribute names of the geometries of layers.

Once you have created a new Cross Navigation in the Cross Navigation Definition menu in Tools section, it is possible to navigate from the GIS document to a target document. There is still a little step to do to activate the cross navigation.

DOCUMENT DETAILS

Label	gis_des_ex *
Name	gis designer example *
Description	
Type	Location Intelligence ▼
Engine	Gis Engine ▼
Data Source	▼
Dataset	usa_state * 🔍
State	Development ▼
Community	▼
Refresh seconds	0
Criptable	<input type="radio"/> True <input checked="" type="radio"/> False
Visible	<input checked="" type="radio"/> True <input type="radio"/> False
Visibility restrictions	<div></div> <div>▼ = <div></div> ➕ 📱</div>
Preview file	<div>Scegli file</div> Nessun file selezionato
Manage output parameters	🔗
Link Document	📄
Locked by user	<input type="radio"/> True <input checked="" type="radio"/> False
Template	<div>Scegli file</div> Nessun file selezionato
Template build	🔗

≡ KNOWAGE

Fig. 5.339: Gis designer accessible from the template build.

LAYER FILTERS

<div>🔍 Search</div> <div>Filter</div> <div>STATE_FIPS</div> <div>SUB_REGION</div> <div>ADD FILTERS</div>	<div>🔍 Search</div> <div>Driver parameter</div> <div>one</div> <div>ADD FILTERS</div>
--	---

≡ KNOWAGE

Fig. 5.340: Layer filters interface with analytical drivers.

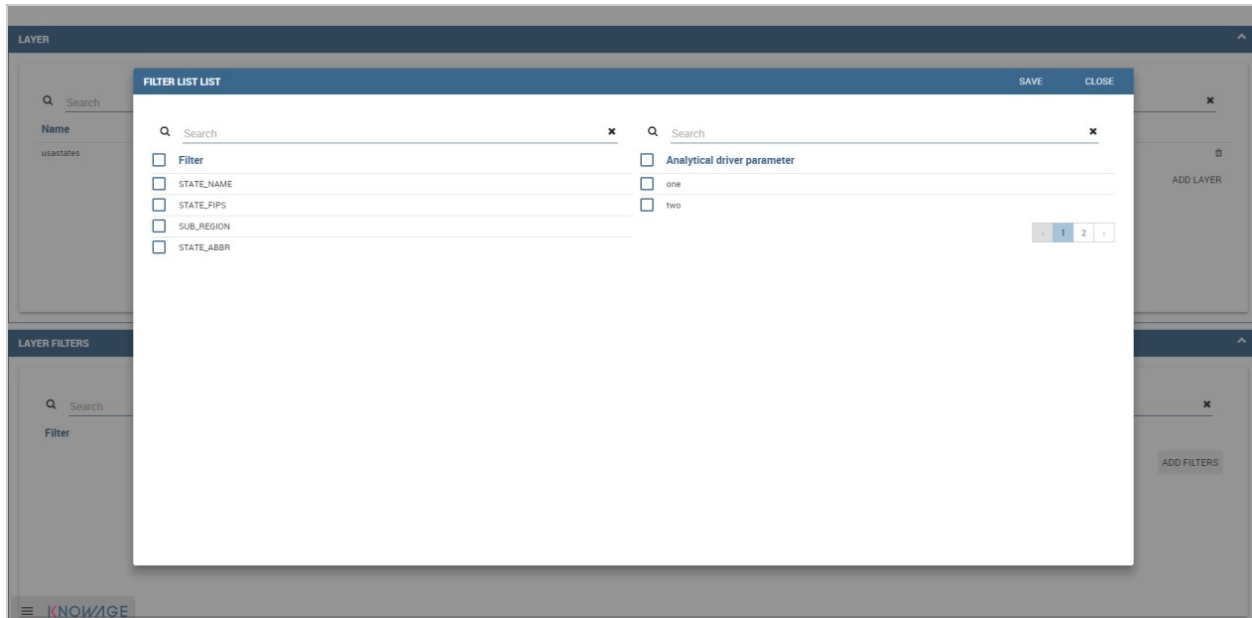


Fig. 5.341: List of available filters with list of analytical drivers.

Open the **layer** tab of the Location Intelligence options panel and click on cross navigation select mode. Now the cross navigation is activated and if you click, for example, on one of the state it will compare the above popup.

By clicking on the play button the target document will open.

5.13 SVG document visualization

In this chapter we will suppose that a technical user has created an SVG document and that the final user is enabled to visualize and use it. He can choose which KPI to show on the “map” and analyze its values, then could drill down into sub-members or other Knowage documents to view other details and other analysis.

The images in Figure above shows how is possible to change KPI analysis and drill towards other SVG documents of the same hierarchy.

5.13.1 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 through an SVG the USA map. At first we can show data at the “Regions” level and then through 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 the two figures below.

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.

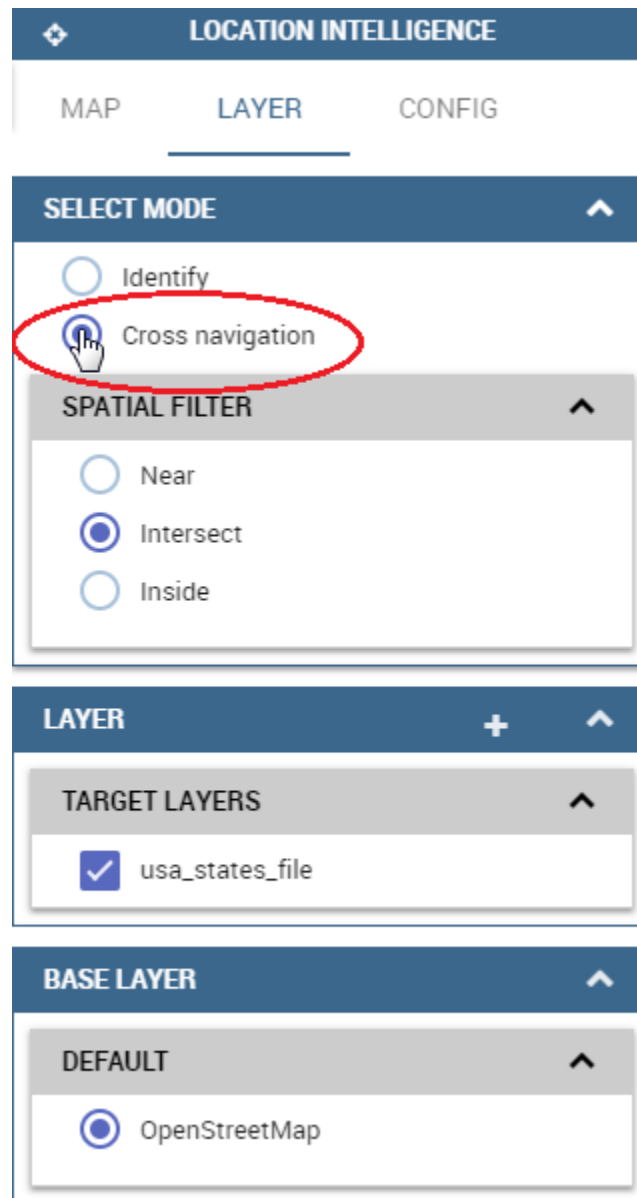


Fig. 5.342: Cross navigation option.

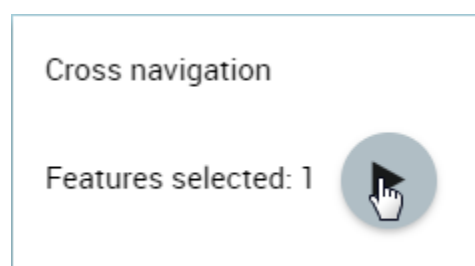


Fig. 5.343: Cross navigation popup.

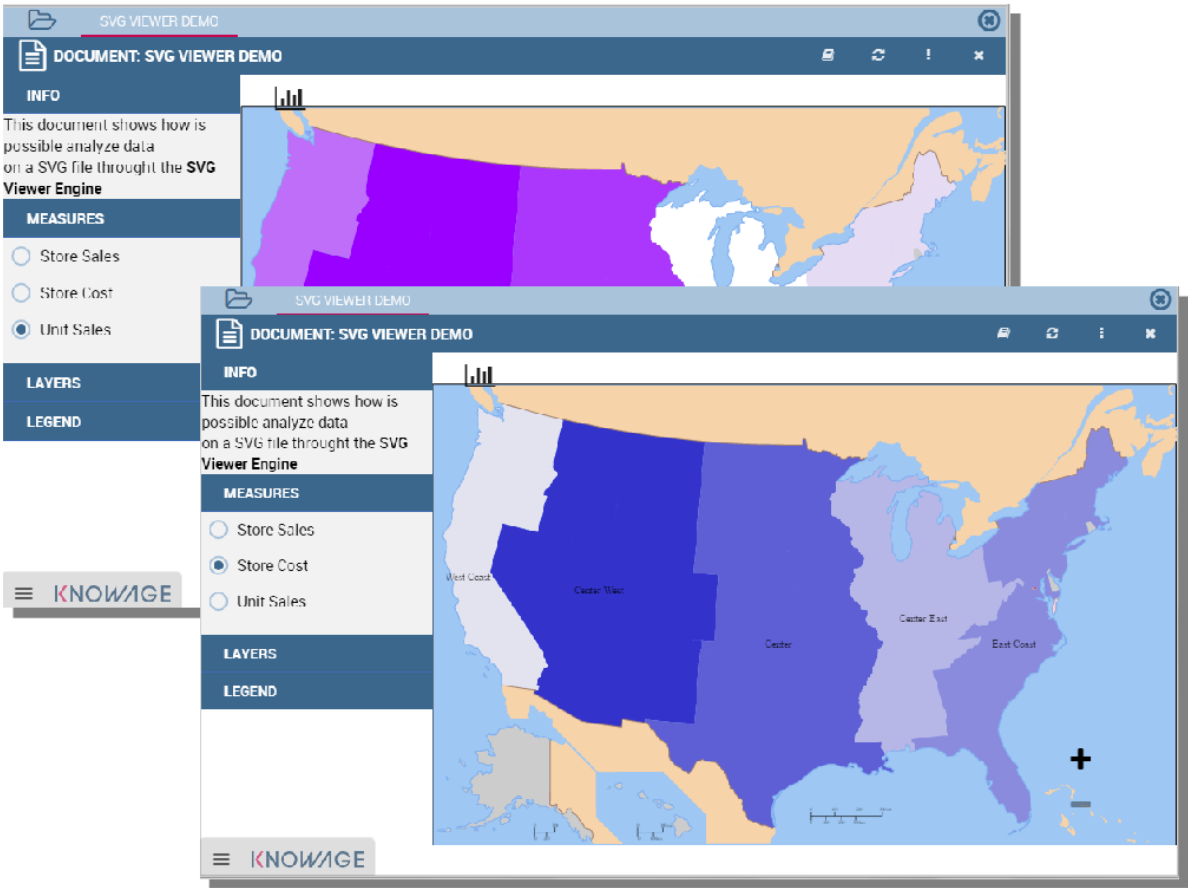


Fig. 5.344: SVG document visualization example.

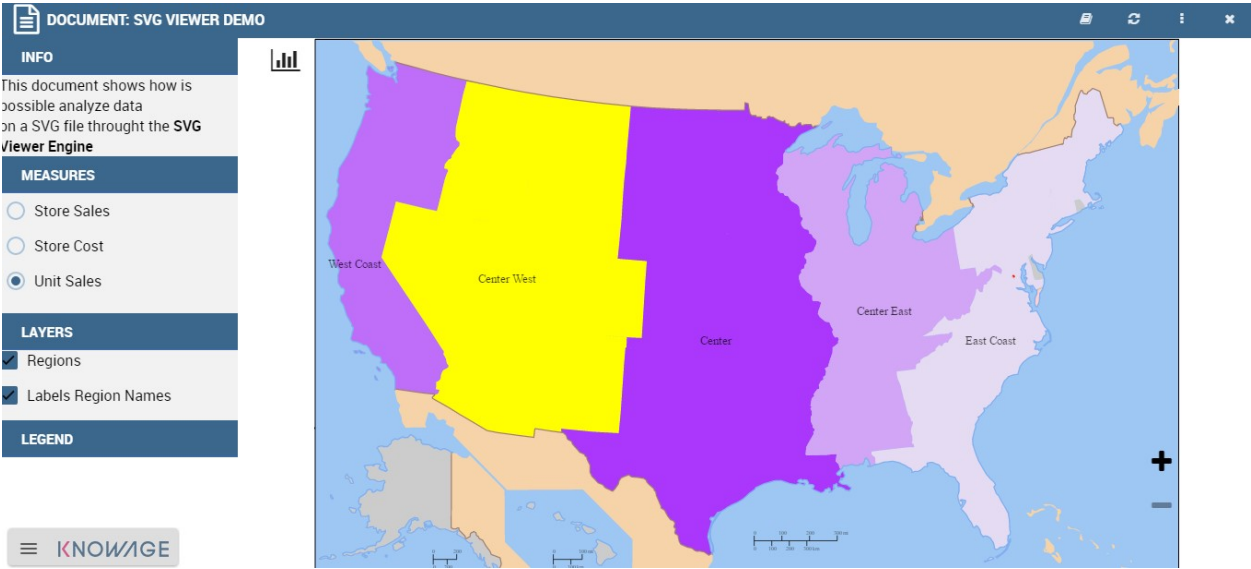


Fig. 5.345: SVG document example at the USA Regions level.

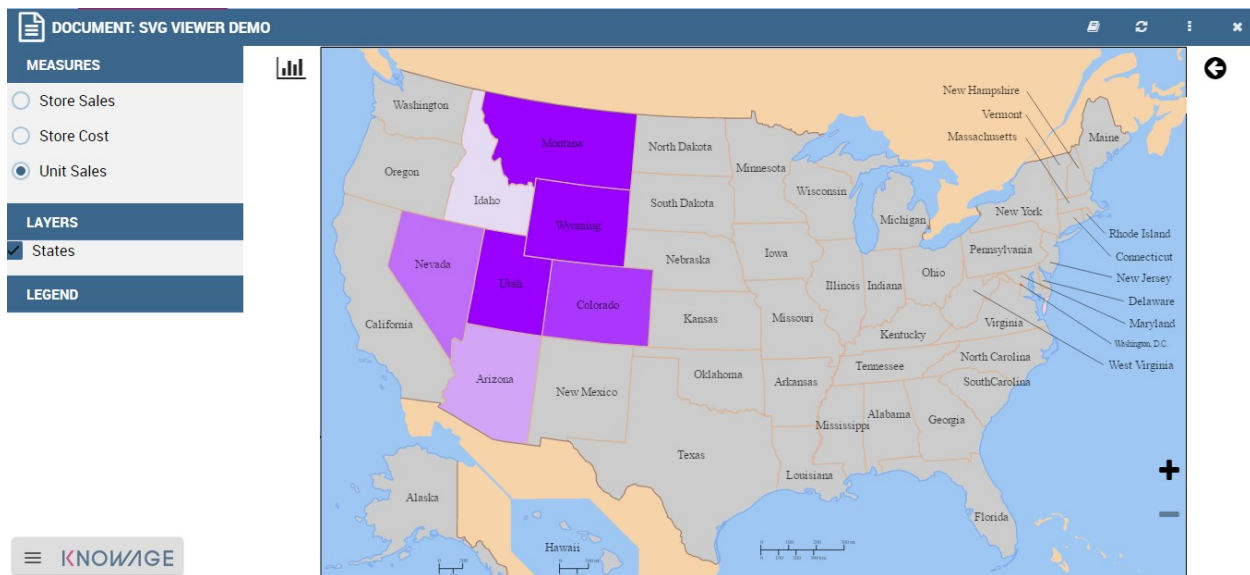


Fig. 5.346: SVG document example at the States level after the selection of the “Center West” Region.

5.13.1.1 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.

5.13.1.1.1 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), concepts (like the item production steps) and so on.

5.13.1.1.2 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 through 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.

SVG DETAIL

Name

Map USA Regions *

Description

Map USA Regions

Hierarchy

USA

Level

1

Member

regions

Template

Scegli file

Nessun file selezionato

Download Svg...

FEATURE DETAIL

regions

centroidi_regions

State_borders

Country_borders

Labels_Region_Names

Labels_State_Names

New...

Name

regions

Description

Type

Fig. 5.347: Entering the hierarchy details.

As you can see in the figure above, 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 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 the figure above 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 below:

SVG DETAIL

Name

Map USA States *

Description

Map USA States

Hierarchy

USA

Level

2

Member

states

Template

Scegli file

Nessun file selezionato

Download Svg...

FEATURE DETAIL

State_borders

Country_borders

Labels_State_Names

Frames

State_borders_old

Ocean_borders

Great_Lakes_borders

New...

Name

State_borders

Description

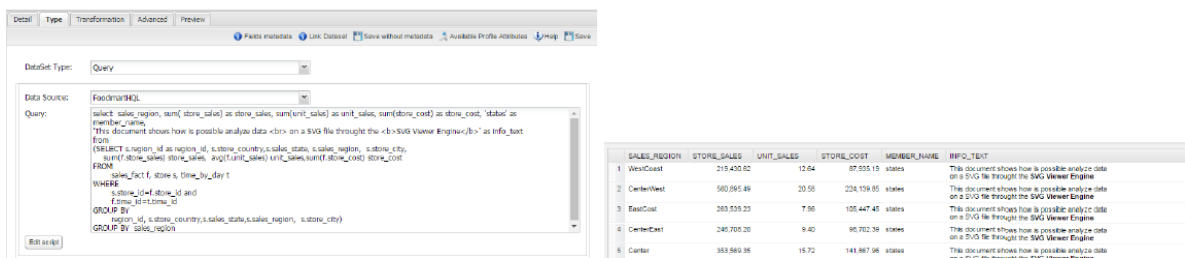
Type

Fig. 5.348: 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.

5.13.1.1.3 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. You can refer to Chapter 3 of this manual to know how to create datasets. Here in the following figure a dataset of our example:



	SALES_REGION	STORE_SALES	UNIT_SALES	STORE_COST	MEMBER_NAME	INFO_TEXT
1	WestCoast	215,430.02	12.04	87,335.13	states	This document shows how is possible analyze data on a SVG file through the SVG Viewer Engine
2	CenterWest	580,895.49	20.58	228,139.85	states	This document shows how is possible analyze data on a SVG file through the SVG Viewer Engine
3	EastCoast	383,530.23	7.86	105,447.45	states	This document shows how is possible analyze data on a SVG file through the SVG Viewer Engine
4	CenterEast	240,700.20	9.40	95,702.39	states	This document shows how is possible analyze data on a SVG file through the SVG Viewer Engine
5	Center	353,989.35	15.72	141,887.96	states	This document shows how is possible analyze data on a SVG file through the SVG Viewer Engine

Fig. 5.349: Left. Dataset behind the SVG document. Right. Dataset preview.

5.13.1.1.4 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 below.

Listing 5.23: Example of SVG code for template file.

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <MAP>
3    <DATAMART_PROVIDER>
4    <HIERARCHY name="USA">
5      <MEMBER name="regions" measure_dataset="ds_regions" level="1">
6      <MEMBER name="states" measure_dataset="ds_states" level="2">
7    <HIERARCHY>
8    <DATAMART_PROVIDER>
9  </MAP>

```

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 the meaning of the main tag in the next table *Recap of properties and function*.

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 below:

Listing 5.24: Example of SVG code for template file.

```

1 <MEMBER name ="regions" measure_dataset = "ds_regions" level="1" >
2   <METADATA>
3   <LAYERS>
4   <MEASURES default_kpi="UNIT_SALES">
5   </MEMBER>

```

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.

Listing 5.25: Example of SVG code for template file.

```

1   <METADATA>
2     <COLUMN TYPE="geoid" column_id="sales_region" />
3     <COLUMN TYPE="measure" column_id="store_sales" />
4     <COLUMN TYPE="measure" column_id="store_costs" />
5     <COLUMN TYPE="measure" column_id="unit_sales" />
6     <COLUMN TYPE="drillid" column_id="member_name" />
7     <COLUMN TYPE="info" column_id="info_text" />

```

Once again we give some details on metadata in next table.

- **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 below and could be activated or disactivated directly by an action of the the final user. At least one layer must be defined.



Fig. 5.350: Available layers set by a technical user.

Table 5.13: Recap of properties and function.

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 possible get its value dynamically through 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.
COLUMN	TYPE	Mandatory. The type of the specific column. Possible values are: <ul style="list-style-type: none"> • 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 through
5.13. SVG document visualization		387

- **MEASURES** Measures are all the business values (KPI) that the user want to monitor through 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 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 the following figure. Is possible 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.

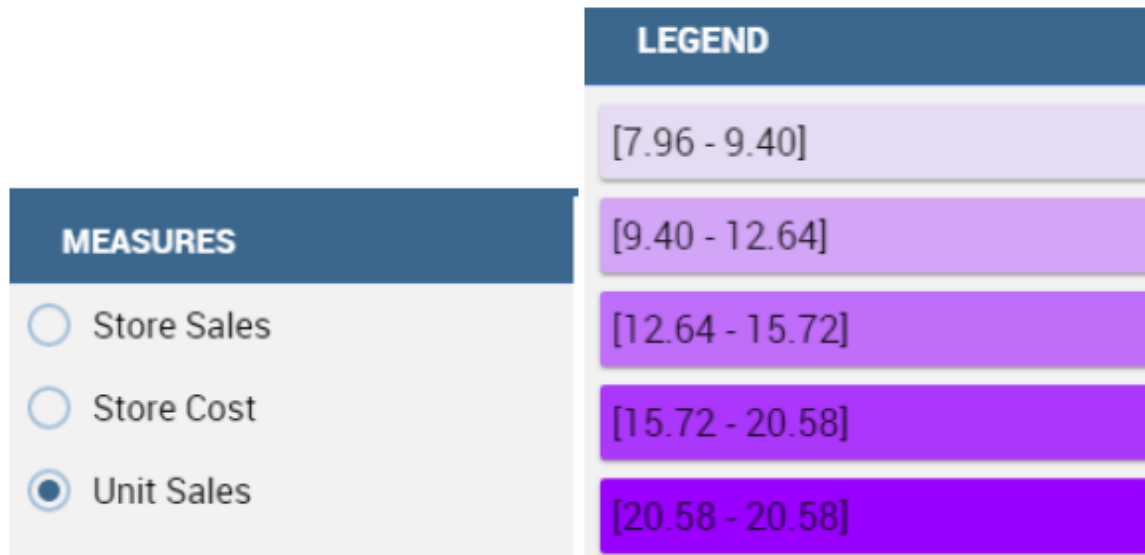


Fig. 5.351: Left. Measure panel. Right. Legend panel.

Listing 5.26: Code for setting the KPI into SVG document.

```

1  <MEASURES default_kpi="UNIT_SALES">
2    <KPI column_id="STORE_SALES" description="Store Sales" >
3      <THRESHOLDS type="quantile" lb_value="0" ub_value="none" >
4        <PARAM name="GROUPS_NUMBER" value="5" />
5      </THRESHOLDS>
6      <COLOURS type="grad" outbound_colour="#FFFFFF" null_values_color="#CCCCCC" >
7        <PARAM name="BASE_COLOR" value="#009900" />
8        <!--<PARAM name="opacity" value="0.5" />--> </COLOURS>
9      </KPI>
10     <KPI column_id="STORE_COST" description="Store Cost" >
11     <KPI column_id="UNIT_SALES" description="Unit Sales" >
12  </MEASURE>

```

We report the next table for further details on THRESHOLDS and COLOURS tag. This table includes the heuristics supporting value interval partition into a finite number of subintervals (type attribute of the THRESHOLDS tag).

While the following table 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 definition, you can create it into Knowage. Remember that it must be a “Location Intelligence”

document type with the engine “SVG Viewer Engine”.

Table 5.14: Recap of layer tag properties and function.

Tag	Property	Note
MEASURES	default_kpi	Mandatory. Defines the default kpi or the kpi that we want enable at the beginning, when we start the document execution. Its value must exist into the METADATA section as measure type.
KPI	column_id	Mandatory. The column_id property the measure that you are defining. Its value must exist into the METADATA section as measure 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: <ul style="list-style-type: none"> - 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 necessary to define correctly the thematization. Its value depends by the threshold type. This attribute 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: <ul style="list-style-type: none"> • 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.
390		

5.13.1.1.5 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:

Listing 5.27: Code for enabling external cross navigation.

```
1 <MEMBER name="states" level="2"
2   measure_dataset="ds_states"
3   enableExternalCross="true">
```

With this setting, you are able to create a “Cross Navigation Definition” with the standard Knowage functionality, where for default you’ll find the element_id as output parameter as shown in figure below. It means that the identifier of the area selected is able to be passed. Other default output parameters are **Hierarchy**, **Member** and **Level**.

Navigation name *

Origin doc

SVG Viewer Demo SELECT

AVAILABLE INPUT/OUTPUT PARAMETERS

Fixed value parameter +

≡ ELEMENT_ID	Output
--------------	--------

Fig. 5.352: Using the Cross Navigation definition to link to external documents.

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.

5.14 Data Mining

Knowage supports advanced data analysis allowing you to extract knowledge from large volumes of data, to improve your decision-making and business strategies. In particular, Knowage **Data Mining Engine** integrates R and Python scripting capabilities.

R is a programming language and software environment for statistical computing and graphics. The R language is widely used among statisticians and data miners for developing advanced algorithms and data analysis tools. Polls and

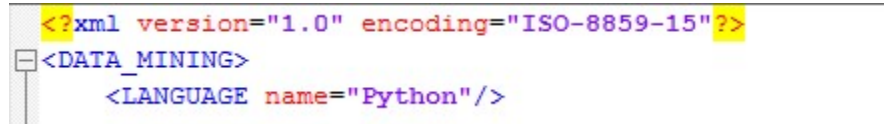
surveys show that R's popularity is continually increasing. As well, Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible and it is intended to be a highly readable language.

Thanks to Knowage **Data Mining Engine**, it is possible to execute multiple R and Python scripts in an interactive way and visualise several outputs, including the powerful R graphics. Another important thing to notice is that it allows users to perform statistical or data mining analysis on different files or Knowage datasets.

The data scientists can thus integrate its own algorithm within Knowage and deliver their output to the end user, together with new advanced visualization options useful to discover meaningful insights hidden in the data.

5.14.1 Data Mining document interface

Data Mining can be implemented using R or Python language as we just said. The starting point for developing a data mining document is to write down a template which consists of an XML file. We refer to My first data datamining document for a more detailed description of the template features. We disclose here that to communicate the engine which language you are going to use you must add the tag **LANGUAGE** as shown in Code syntax to recall the variable input



```
<?xml version="1.0" encoding="ISO-8859-15"?>
<DATA_MINING>
  <LANGUAGE name="Python"/>
```

Fig. 5.353: Setting the language for data mining.

For both R and Python languages, the engine yields different kind of outputs. The outputs can be of type:

- text,
- dataset,
- images,
- html file (only when using the R language).

Once again the outputs are set at template level. In the following we will describe how to browse and visualise each type of output. The Data Mining document execution is very simple in all cases. Once configured the document, the execution starts clicking on the document's icon from Knowage Document Browser interface as shown below.

We suppose to open a document with text outputs. At the top of the page you find the command box. When the document is launched, it is automatically executed with the default command (which is the one set to **auto** in the template file) and you get the corresponding outputs displayed in different tabs. If you wish to change the command to get other outputs you can use the dedicated combobox as emphasized in the following figure.

We say in advance that the command are set in the template file. As you can check in next figure the template must contain the **COMMAND TAGS**. In the example each command (two in this case) is made up of instructions to be executed in order to compute one or more measures.

The combobox will then reflect the command settings inside the template. Therefore it is important that the final user is told the meaning of each using "speaking" names or furnishing information with a documentation.

To switch to a different command, use the combobox and click on the target command. The simple click activates the execution.

The computed outputs are visualized in the half bottom of the page: each output populates a tab.

If the command refers to a script that needs one or more datasets, than these datasets are displayed between the command combobox and the outputs tabs. You can use the combobox which list the Knowage dataset matched to the document through the template. Otherwise you can upload a file for Knowage datasets that cannot be changed from

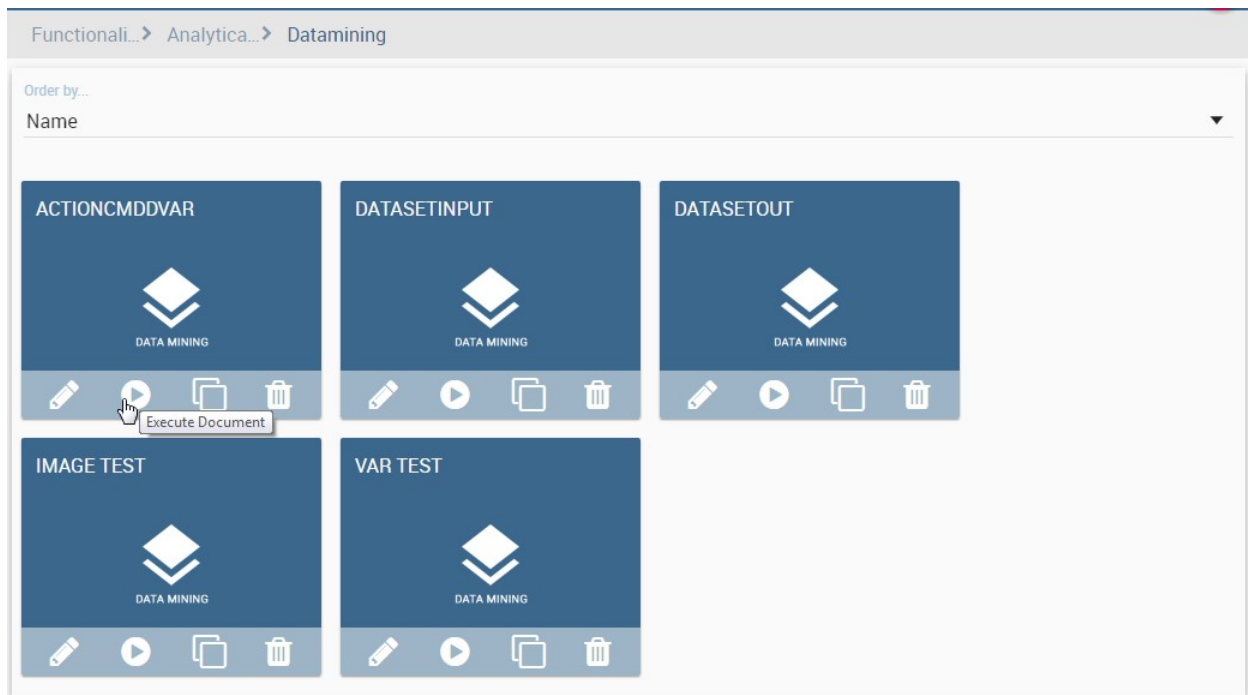



Fig. 5.354: Document browsing on server.

the GUI. Clicking on the **FILE** button it is possible to upload the file to replace the default dataset. Remember to click on the  to start the upload.

Then a **Run script** button will appear to re-execute the document as indicated below.

As well as for other analysis tools, data mining allows to filter data through parameters. In this case the setting of the parameters can be done in the script tag or the command tag. The main parts of the code for the configuration of parameters are highlighted in figure below.

We will not go further into details of R or Python code so we leave the user to take care about the issue. Generally you can use two kind of parameters (when defined by developers): output parameters and command parameters. These may be required for changing factors or more generally fields (string or numbers) inside the script or in the output functions.

When dealing with output parameters, you can update the corresponding values by filling the input box appearing in the respective output tab panel. Once you re-run the document, the modifications are applied to a single output, the one which the parameter is associated to.

On the other hand, when dealing with command parameters, you can update the variable value by clicking the double gear icon (circled in the following figure) displayed next to the command name.

The button updates the interface and opens a box (see figure below) where you can change a convalidate the modification. The variable value is passed to the whole command and hence it updates the variables of all command outputs.

In case the commands produce image outputs the interface is essentially the same as the text output case. So you can change commands, dataset and set parameter values. The output tabs will though display data through graphics. An example is given below.

Also in the dataset output case there are not considerable changes in the window organization. A Data Mining document with dataset output transform a query over a data source or a plain data container into a dataset on Knowage

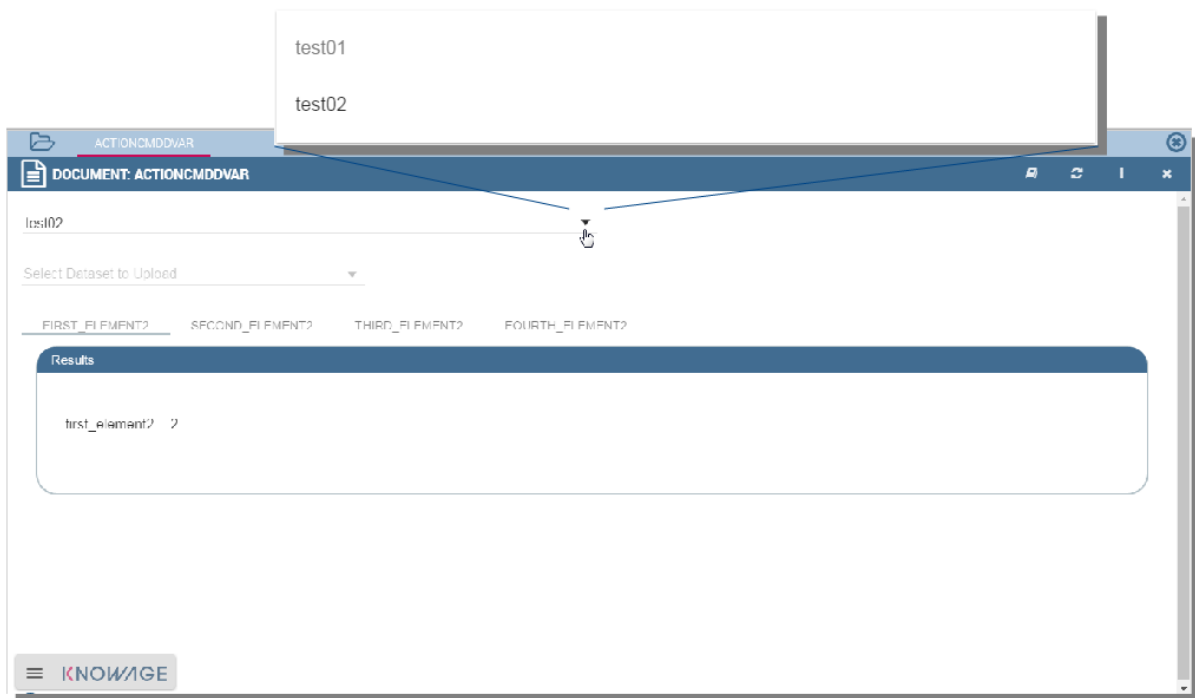


Fig. 5.355: Selecting the command to be applied.

```
<COMMANDS>
  <COMMAND name="testcommand" scriptName="test01" label="test01" mode="auto" action="inita()">
    <OUTPUTS>
      <OUTPUT type="text" name="first_element" value="" function="fib(3)" mode="manual" label="first_element"/>
      <OUTPUT type="text" name="second_element" value="" function="fib(4)" mode="manual" label="second_element"/>
      <OUTPUT type="text" name="third_element" value="4" function="fib" mode="manual" label="third_element"/> <!-- problema-->
      <OUTPUT type="text" name="fourth_element" value="" function="loga()" mode="manual" label="fourth_element"/>
    </OUTPUTS>
  </COMMAND>
  <COMMAND name="testcommand2" scriptName="test02" label="test02" mode="" action="inita()">
    <OUTPUTS>
      <OUTPUT type="text" name="first_element2" value="" function="fib(3)" mode="manual" label="first_element2"/>
      <OUTPUT type="text" name="second_element2" value="" function="fib(4)" mode="manual" label="second_element2"/>
      <OUTPUT type="text" name="third_element2" value="4" function="fib" mode="manual" label="third_element2"/> <!-- problema-->
      <OUTPUT type="text" name="fourth_element2" value="" function="loga()" mode="manual" label="fourth_element2"/>
    </OUTPUTS>
  </COMMAND>
</COMMANDS>
```

Fig. 5.356: The command tags.

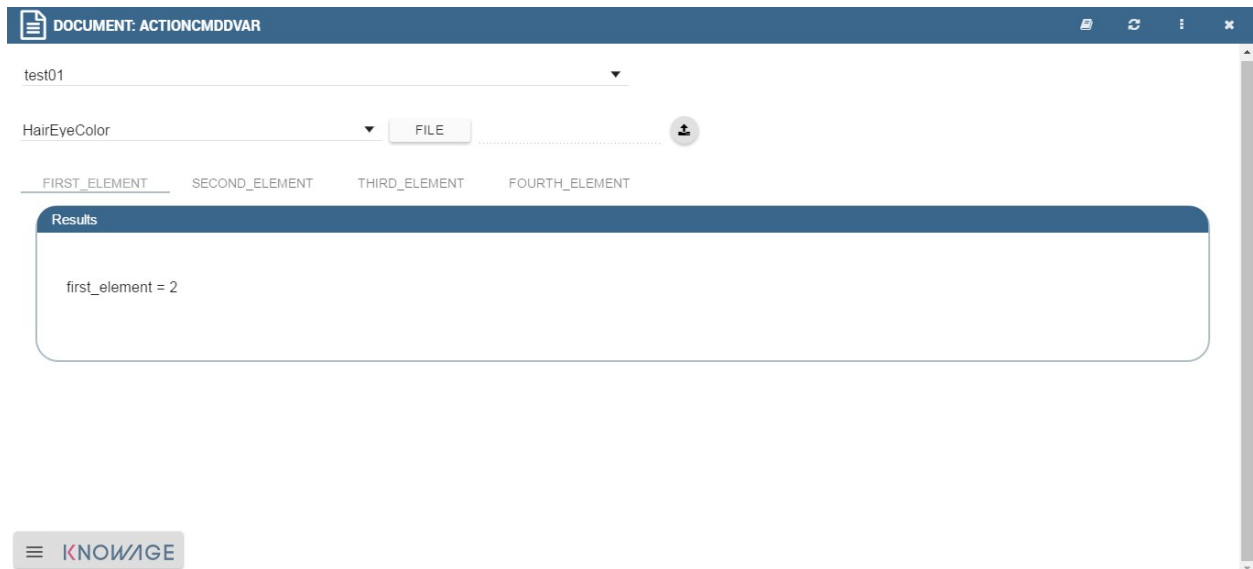


Fig. 5.357: Changing dataset related to a command.

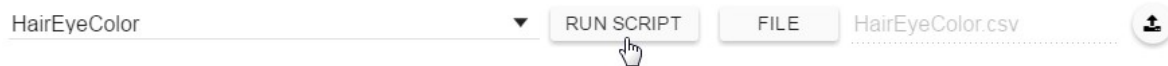


Fig. 5.358: Execution button after the upload.

```

var=$P{var}
    ]]>
</SCRIPT>
</SCRIPTS>
<COMMANDS>
  <COMMAND name="testcommand" scriptName="test01" label="test01" mode="auto" action="inita()">
    <VARIABLES>
      <VARIABLE name="var" default="3"/>
    </VARIABLES>
    <OUTPUTS>
      ....

```

Fig. 5.359: Setting parameters.

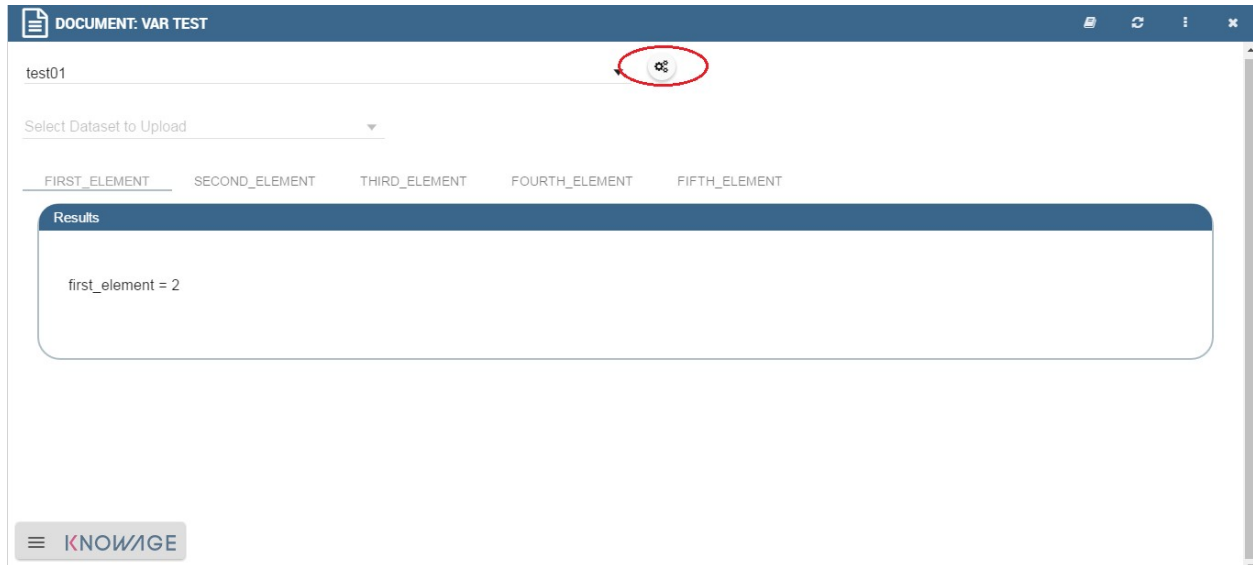


Fig. 5.360: Click on the double gear button to change the .

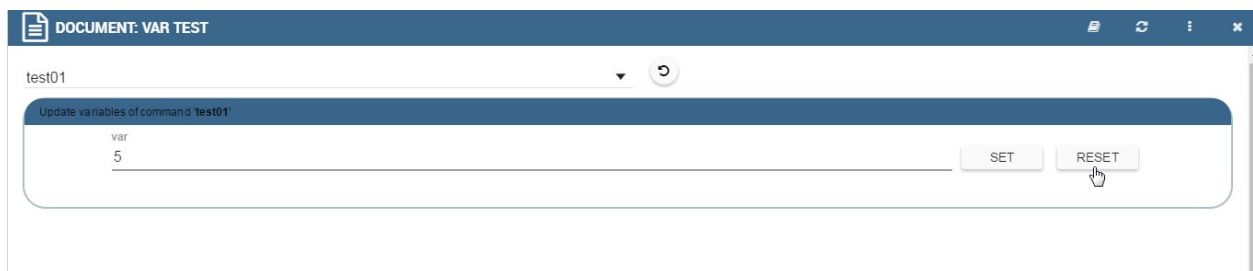


Fig. 5.361: Changing the parameter values.

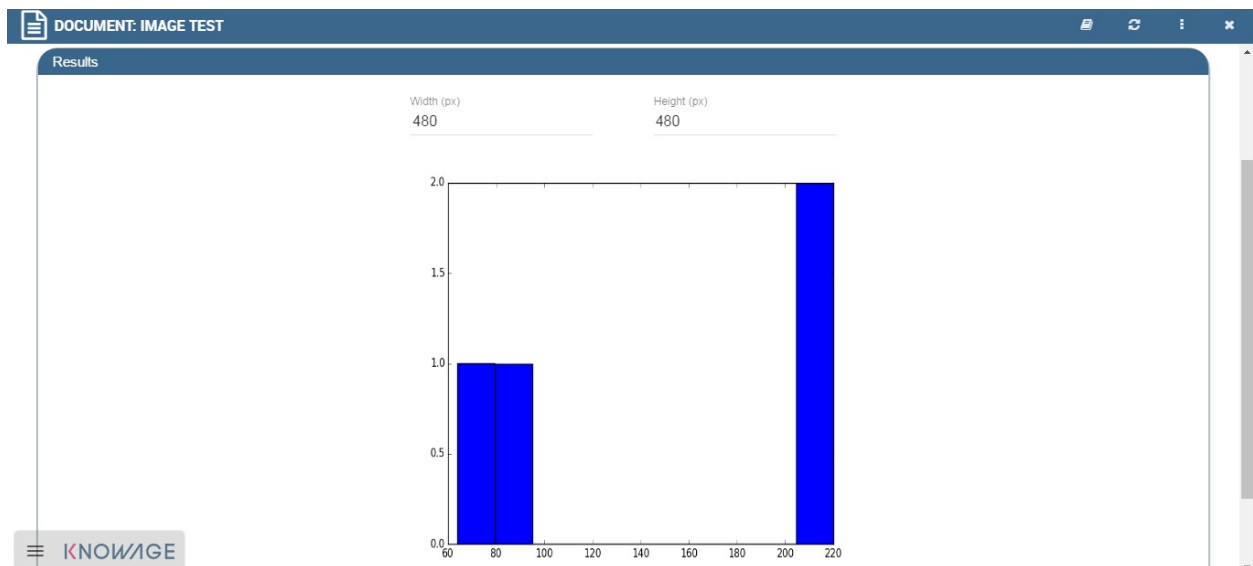


Fig. 5.362: Image outputs.

Server. For instance, this kind of output comes to be really useful when the user needs to convert a .xlsx or .csv file into a dataset on Server. The output tab will accordingly shows a message stating the name of the dataset as stored in Knowage Server, under Data Provider » data set menu item. The following figure gives an example.

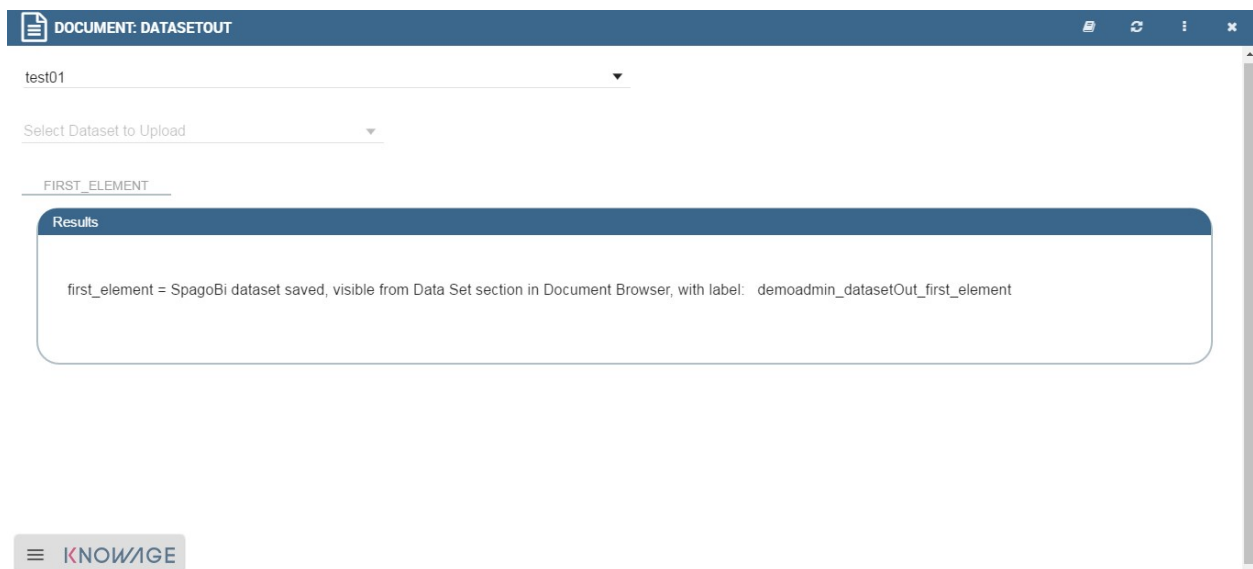


Fig. 5.363: Dataset outputs.

Only when using R language, the outputs can be set to html type. In this case, the document execution will provoke the opening of a web page containing the results requested through the command instructions.

5.14.2 Functions Catalog

The Data Mining can also be managed through the **Functions** framework. In this section we will see how to explore and handle this part, while in Create a new function in Function Catalog we will see how to create a new function.

First click on the **Functions Catalog** from the Knowage main page as shown below.

You will enter a page like the one shown in figure below.

The actions that a user can perform depend on the user's role. However, independently from the user's role, once entered the feature all functions are shown by default. Referring to the figure above, one has the page made up of:

- **categories:** these are set by an administrator user and are used to classify the functions accordingly to their definition and goals. Moreover they're of help in browsing the functions; only the admin user can add and/or modify categories.
- **tags:** they are used to easily sharpen the research and easily recall the functions that are tagged with that word; once again only the admin user can add and/or modify tags;
- **list of functions** (if there are any): these are visible and explorable by any kind of user. Anyway only an admin user can add and/or modify them.

Hint: Add or modify the categories

The admin can add a new category using the Domain management available on Knowage Server under the Server Settings section. To know more about this section, please refer to Section "Server settings" of the General Administration Manual.

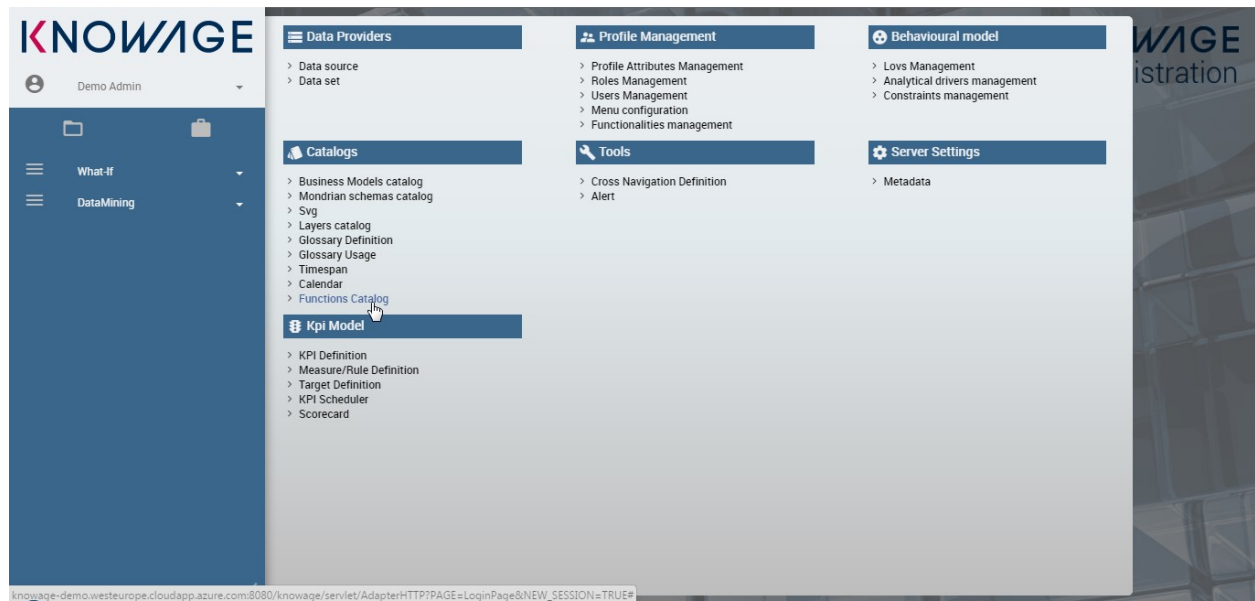


Fig. 5.364: Functions Catalog from Knowage menu.

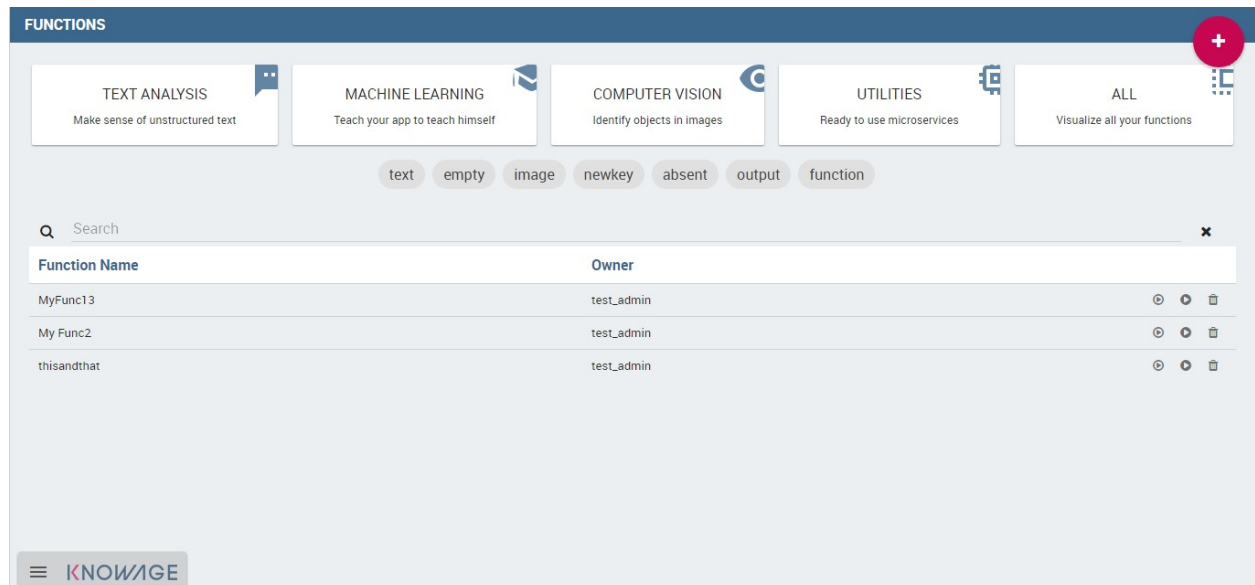


Fig. 5.365: Functions Catalog interface.

The categories for functions depends on an admin user. Taking *Functions Catalog interface* figure as an example, we have:

1. **Text Analysis:** make sense of unstructured text,
2. **Machine Learning:** teach your app to teach himself,
3. **Computer Vision:** identify objects in images,
4. **Utilities:** ready to use microservices,
5. **All:** visualizes all your functions; this is the only category that cannot be changed or removed.

To facilitate the comprehension we created some functions to be examined. We recall here that one can look for a function in different ways: using the categories or the tags or using the Functions Catalog “Search” box available at the top of the functions list as highlighted below.

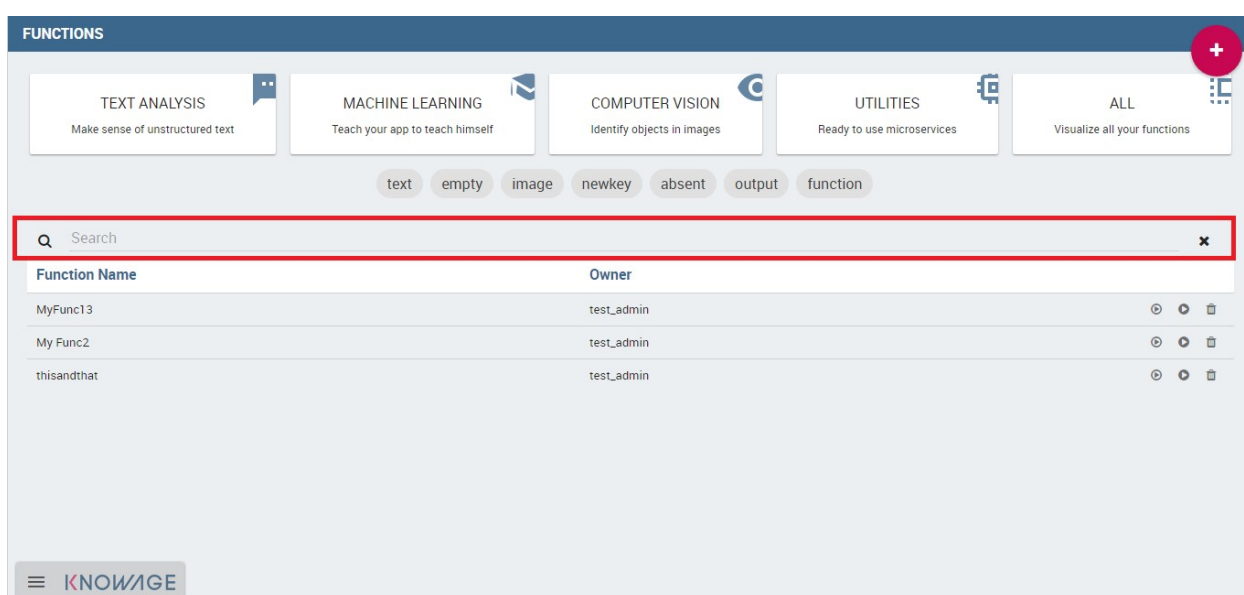





Fig. 5.366: Search box to look for a function.

We suppose here to select one category, which means to click on the category box, in order to be able to analyse the functions belonging to it.

Note that the underlined part in figure below contains a list of tags. These help to focus on the subjects and therefore functions associated to that category. Vice versa when all functions are shown, all tags are shown as well and they can be used to pick up functions related to that subject.

A function can be executed using the icon  which launches a demo (i.e. the function with default values) or using the icon  which launches the computation after the insertion of new values for data. Use the icon  for deleting the function. Only the an admin user can use the three options, while the final user can use only the “execution” button.

To create a new function an admin user must click on the “Plus” icon available at the right top corner of the page. The action opens the interface shown below. Here you have four tabs that we describe shortly in the following subsections.

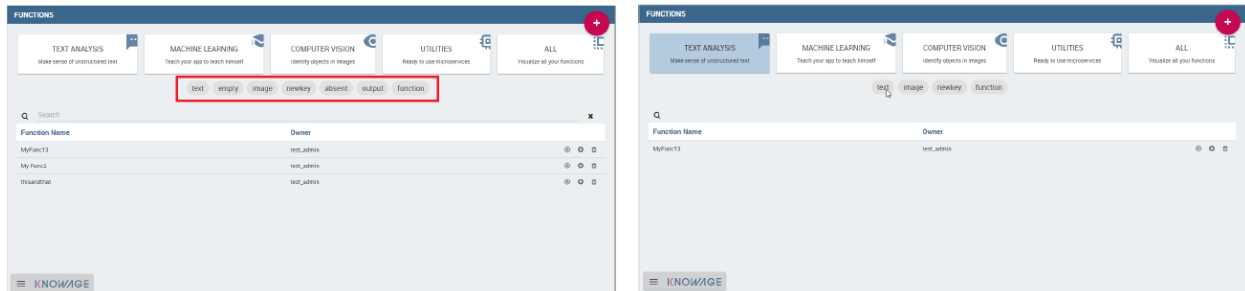


Fig. 5.367: Using tags and categories to look for functions.

The image shows a screenshot of the Knowage Functions form for creating a new function. The form has tabs for 'GENERAL', 'INPUT', 'SCRIPT', and 'OUTPUT'. The 'GENERAL' tab is active, showing fields for 'Function Name', 'Label', 'Owner', 'Type', 'Keywords', and 'Description'. The 'Owner' field is filled with 'test_admin'. The 'Description' field has a rich text editor toolbar.

Fig. 5.368: Creating a new function.

5.14.2.1 The General tab*

In this tab the user gives the general information about the function as the figure above shows. The admin user must type: the *name* of the function, the *label* with which it is identified uniquely (remember to use only numbers or letters and do not leave spaces between them). The *keywords* are where tags are defined. Finally the *Description* is where the user can insert a text or images to be shown when the function outputs are visualized.

5.14.2.2 The Input tab*

As shown in the following figure, the function admits three kind of input: the *dataset*, the *variables* and the *files* one.

Fig. 5.369: Input tab.

In the “Dataset” instance the function takes values from a Knowage dataset. It can be chosen from the combobox available in the dedicated area. Note that the combobox shows the labels of the datasets. It is also possible to ask for the preview so the user can check if the values suit the wished requests.

Fig. 5.370: The dataset input of the function settings.

In the “Variable” case, the user must insert one or more variables and match them with values using the dedicated area.

Input Variables		ADD INPUT VARIABLE
Variable name	Variable value	
var1	1	✕
Variable name	Variable value	
var2	2	✕
Variable name	Variable value	
...	✕

Fig. 5.371: The variable input of the function settings.

In the “File” case, the user is asked to browse folders and upload the wished document remembering to give an alias to it. Files as videos, images, etc are all supported by the functionalities.

Input Files		ADD INPUT FILE
File alias	Select a file	
File example	<div>BROWSE</div>	Right-Left_Hanc Loaded Right-Left_Handed.xls

Fig. 5.372: The file input of the function settings.

5.14.2.3 The Script tab*

The script tab is where an expert user defines the function through the usage of datamining languages R or Python, as shown in Figure below, or calling for an external link. In particular, it is possible to choose between the two options **Local** and **Remote**.

SAVECLOSE

GENERALINPUTSCRIPTOUTPUT

Local

Remote

Language

Python

Script

1

KNOWAGE

Fig. 5.373: The script tab.

We suppose we have chosen the “Local” modality and that we selected a dataset in the previous input tab. In this case the dataset is transformed into an R dataframe that can be recalled while editing the script using the same name of the dataset label. The following figure shows an example.

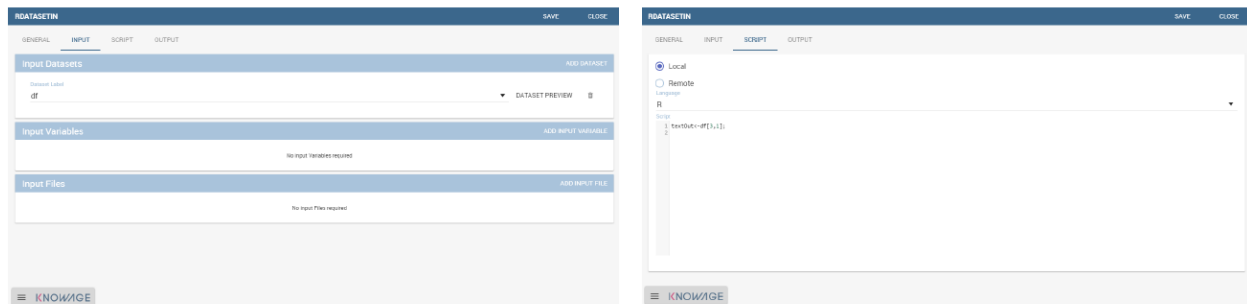


Fig. 5.374: Using the dataset dataframe generated by the software to edit the R script.

Note that if the function takes variables or files as input you can recall them through their name (as specified in the input tab). In particular, refer to Code syntax to recall the variable input in the variable instance, while for the file case remember that the alias will contain the file path.

Listing 5.28: Code syntax to recall the variable input

```
$P{variable_name}
```

We suppose now to have chosen a dataset and the local modality but to want to use the Python language (see next figure). In this case the dataset is saved and read by the script as a dataframe of the pandas libraries: <http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html>

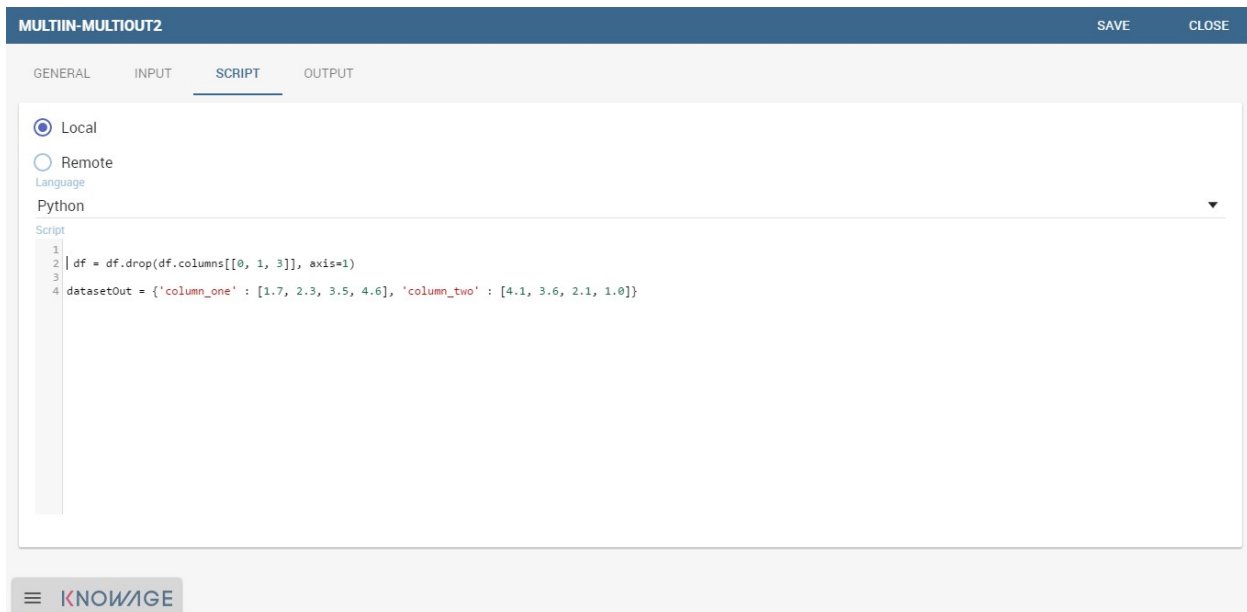


Fig. 5.375: Using the dataset dataframe of the pandas libraries generated by the software to edit the Python script.

The “Remote” instance is used for external services and when the user wants to use a language which is not supported by Knowage server. When selecting this modality the user is asked to insert an URL calling for an external web site that supports and runs the requested language.

Technically, remote functions are recorded in the catalog list. The input data of those functions are specified by the local Knowage request and the code is not stored inside Knowage. On the contrary it is located at the address specified by the URL.

The screenshot shows the 'REMOTE' configuration window in Knowage. It has a top bar with 'SAVE' and 'CLOSE' buttons. Below is a tabbed interface with 'GENERAL', 'INPUT', 'SCRIPT', and 'OUTPUT' tabs. The 'INPUT' tab is selected and contains three main sections:

- Input Datasets:** A text field labeled 'Dataset Label' containing 'biadmin_datasetOutput_first_element'. To the right is a 'DATASET PREVIEW' button and a trash icon. An 'ADD DATASET' button is in the top right corner of this section.
- Input Variables:** A table with two columns: 'Variable name' and 'Variable value'. The first row has 'a' and '4'. To the right is a trash icon. An 'ADD INPUT VARIABLE' button is in the top right corner.
- Input Files:** A text field labeled 'File alias' containing 'f1'. To its right is a 'Select a file' label, a 'BROWSE' button, and a text field showing 'Loaded dfReduced.csv'. A trash icon is on the far right. An 'ADD INPUT FILE' button is in the top right corner.

At the bottom left, there is a 'KNOWAGE' logo.

Fig. 5.376: Input definition for remote function.

To define a remote function you have to perform the steps seen above, therefore to specify label, name, inputs and outputs. Figure below shows an example.

When opening the Script tab, select the Remote Radio button. The action will create a remote address and the editor where to insert the code will not be available and the user will have only the chance to specify the URL where the code is placed.

The function that you are defining must be a REST service, in particular of POST type, and it will receive the input data in the JSON format with the syntax showed in JSON format for remote function.

Listing 5.29: JSON format for remote function

```

1  [
2    {
3      "type": "variablesIn",
4      "items":
5        {
6          "demoVarName1": "3",
7          "demoVarName2": "3"
8        }
9    },
10   {
11     "type": "datasetsIn",
12     "items":
13       {
14         "demoDsName1": "df1",
15         "demoDsName2": "df2"
16       }
17   },
18   {
19     "type": "filesIn",
20     "items":
21       {
22         "demoFileAlias1":
23           {
24             "filename": filename1,
25             "base64": ...

```

(continues on next page)

Fig. 5.377: Remote function definition.

(continued from previous page)

```

26     },
27     "demoFileAlias2":
28     {
29         "filename:filename2",
30         base64 ...
31     }
32 }
33 ]
34 ]

```

When the call runs successfully, the remote function must answer with a JSON element like the one exhibited in Code below.

Listing 5.30: JSON answer of a remote function

```

1  {
2  "resultType":"Image",
3  "result": ".image content in base64.",
4  "resultName": "res"
5  },
6  {
7  "resultType": "Dataset",
8  "result": "outDatasetLabel",
9  "resultName": "datasetName"
10 },
11 { "resultType": "File", "result":
12 {
13 "filesize": "54836", --optional
14 "filetype": "image/jpeg", --optional
15 "filename": "chart.jpg", --optional
16 "base64": ".file content in base64." },
17 "resultName": "fileToBeSave"
18 }

```

If an error occur the function must returns the lines as shown in JSON format for remote function.

Listing 5.31: JSON answer of a remote function

```

1 {
2   "service": "",
3   "errors": [
4     {
5       "message": "Here the error message."
6     }
7   ]
8 }

```

5.14.2.4 The Output tab*

Finally it is important to specify what kind of outputs the function will produce. Using the “Output” tab shown below, you can choose between:

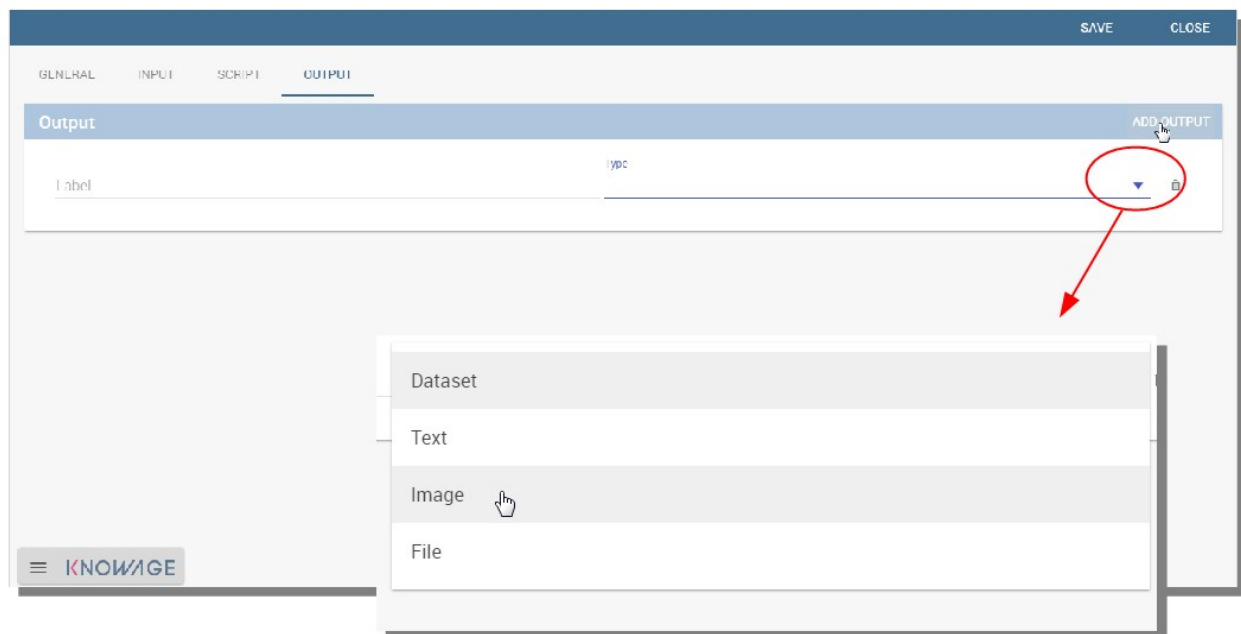


Fig. 5.378: Choosing the output type in the function definition.

- **Dataset:** the function will return a set of records as the Knowage dataset way;
- **Image:** the function will return one or more graphics showing the results through bar or pie charts or other kind of visual tools;
- **Text:** the function will return a window containing some text;
- **File:** the function will return a file.

It is possible to define more than one output for the same function. As an example, in the following figure you can see the execution of the demo for a function called “Heart diseases”. The latter was set to have two outputs, one is of type “Dataset” and the other of type “Image”. The execution opens then a window with two tabs. The first tab contains the Dataset type output, which is translated visually with a table. While the second tab contains the Image output namely a set of graphics as configured to.

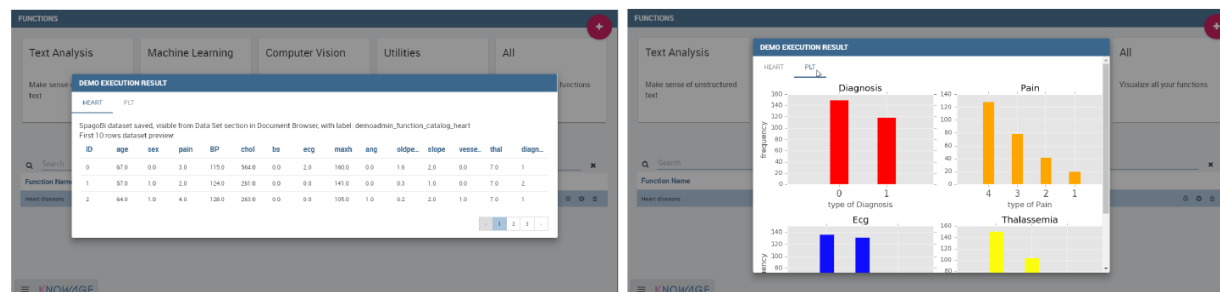


Fig. 5.379: Execution of demo for a function.

Clicking on the second execution icon you be asked to insert the new value and run the function after filling all boxes in. Figure below shows the window opening when one asks for inserting new data values.

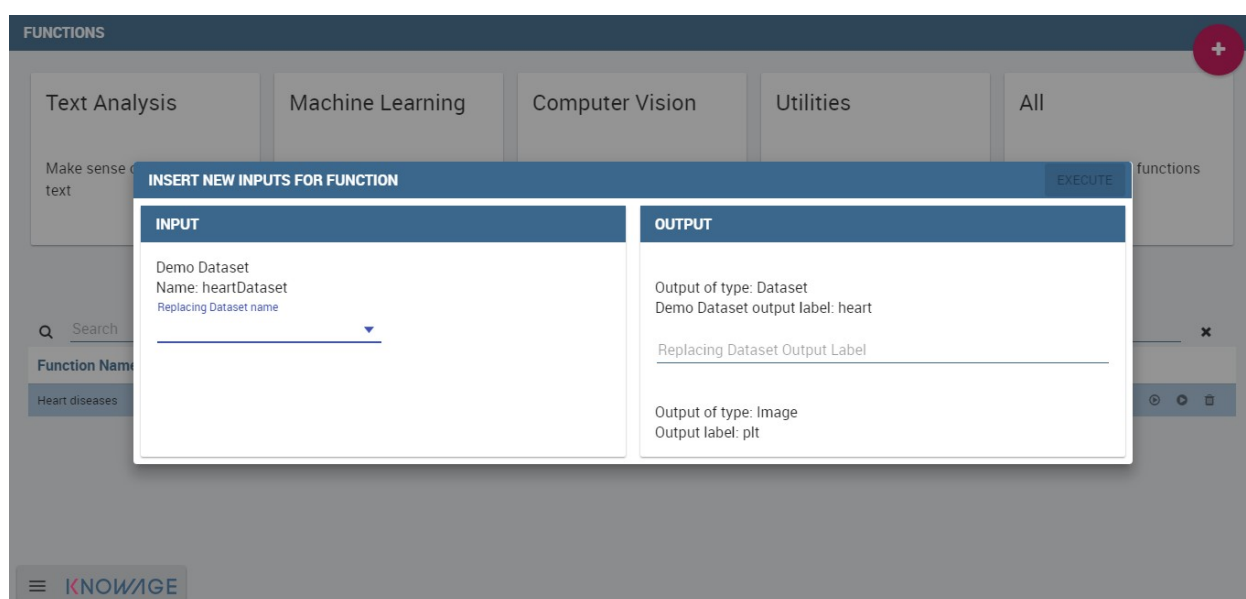


Fig. 5.380: Inserting new data values for function.

Finally clicking on the function name as shown below you can enter function configuration details and modify them.

As well as for the input case, the script can recall the output elements. We need to distinguish between the R and the Python language. Note that, in the dataset case, the user needs to name the output as reported in the script body. The two figures below show an example.

When using Python the datasetOut variable is a “pandas” dataframe while, when using R it is a dataframe. Then it is important in fact to consider the objects’ structure (input and output type must match).

When the script runs using a certain output dataset Knowage server produces a dataset whose name and label is label <User_Name> functionsCatalog <label specified in the Output tab>.

As an example the function produces a dataframe whose label and name are biadmin_functionsCatalog_datasetOut.

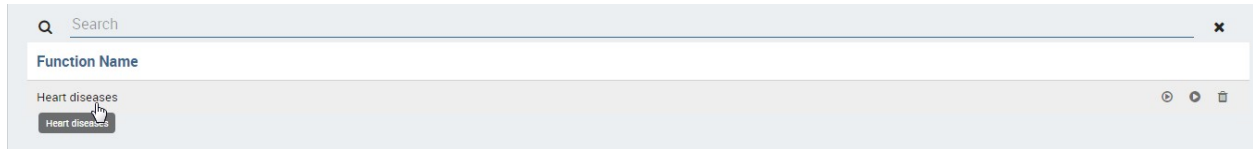


Fig. 5.381: Clicking on function name to modify it.

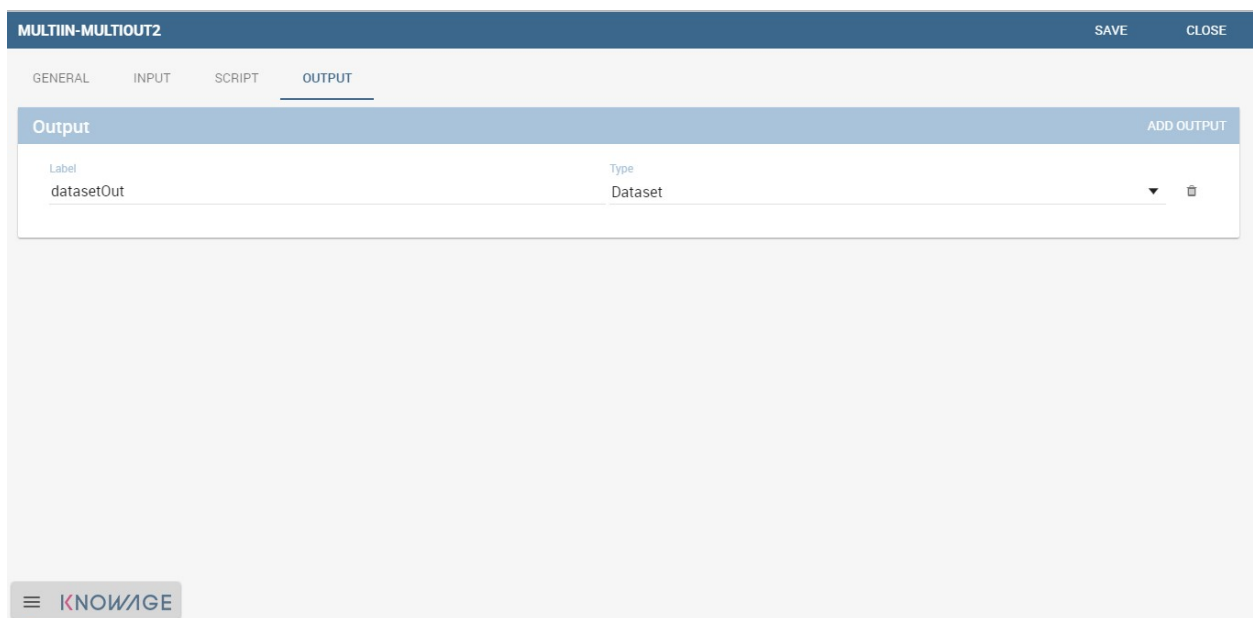


Fig. 5.382: Defining the output example.

5.14.3 Engine description

The **Data Mining Engine** is thought and implemented in order to supply KnowAge with data mining capabilities, but it also enhances OS R and Python with several distinguishing features.

Basically the integration is done through the rJava R package, JRI library and the JPY library for Python. R/Python scripts are written inside the data mining documents template and evaluated server side once the document is executed. Nevertheless, developers can combine several scripts, each one with its own outputs.

The leading component is the command object, that holds the activation of one script, together with its outputs. There can be many commands, but the one which is executed at document start up, is the one with mode set to auto, whilst all of the others will be executed once user clicks on the corresponding element.

5.14.3.1 Data

Each script can run on two kinds of datasets. The first one is the file type, that means users can upload their files, interacting with the GUI. Note that the extension of the file, as well as how to read it (comma separated or tab separated, is header present or not,...) can be specified in the proper DATASET tag in document's template. The other type of dataset is the Knowage Parametrization and customization dataset. This feature allows to use inside your R scripts data retrieved from many kind of sources, without ad-hoc R packages utilization.

Moreover this is very useful for Big Data data sources (Hive, Impala, Hbase, NoSQL databases as MongoDB, OrientDB, Cassandra etc..), because not all of them are connectable from R, and if they are, in most cases, R must be installed on their cluster.

Therefore this is the first powerful feature Knowage adds to R. Each Knowage dataset is readable as a CSV file, so developers must apply the proper functions.

5.14.3.2 Parametrization and customization

Another characteristic is that the data mining document is customizable both with Knowage Analytical driver and GUI variables. The first choice, enables developers to use the behavioural model to change the results of the Knowage datasets used by the scripts. These parameters don't modify other data mining document's parts. On the other hand, setting GUI variables can be useful to change the outputs of the scripts, but they don't affect the resultsets of the datasets.

As explained at the beginning of the chapter, there are two kind of variables: output variable and command variable. Once the user runs again the document by saving the value of the variable, the value is passed to the output function. On the other hand, by choosing the second option the whole script belonging to the command will benefit of the variable update.

5.14.3.3 Outputs

Knowage data mining document can perform a set of scripts and visualize them according to the associated predefined outputs, that can be images as well as text. This combination of results, that can be modified on-the fly using variables and shared across the network through Knowage web application, can exploit R workloads. Indeed Knowage can provide role's privileges to the document's access or execution.

5.14.4 My first data mining document*

Create a new generic document and select **Data Mining** as Type and **Data-Mining Engine** as Engine. Define label, name and description and associate the correct datasource. The next step is the definition of the template.

The template of a data mining document is a simple XML file that enables the developer to configure properly the document behaviour. Look at Linux and Tomcat example.

Listing 5.32: Linux and Tomcat example

```

1  <?xml version="1.0" encoding="ISO-8859-15"?>
2  <DATA_MINING>
3  <DATASETS>
4      <DATASET name="fileDs" readType="table" type="file" label="Dataset_Label_01 " canUpload="true
5  <"/>
6      <![CDATA[ ...read_options...]]>
7  </DATASET>
8  </DATASETS>
9
10 <SCRIPTS>
11 <SCRIPT name="Script_Name_01" datasets="fileDs" label="Script_Label_01">
12 <![CDATA[.... action_to_call<-function(x){ ... }]]>
13 </SCRIPT>
14 <SCRIPT name="Script_Name_02" datasets="fileDs" label="Script_Label_01">
15 <![CDATA[... z1<-'$P{var1}' ... ]]>
16 </SCRIPT>
17 <SCRIPT name="Script_Name_03" label="Script_Name_03">
18 <![CDATA[... z2<-'$P{var2}' ... ]]>
19 </SCRIPT>
20 </SCRIPTS>
21
22 <COMMANDS>
23 <COMMAND name="command1" scriptName="Script_Name_01" label=" Command_Label_01" mode="auto">
24 <OUTPUTS>
25 <OUTPUT type="image" name="a" value="x" function="plot" mode="auto" label="Output_Label_01"/
26 <OUTPUT type="image" name="c" value="z,k" function="biplot" mode=
27 "manual" label="Output_Label_02"/>
28 <OUTPUT type="text" name="d" value="y
29 mode="manual" label=" Output_Label_03"/>
30 </OUTPUTS>
31 </COMMAND>
32 <COMMAND name="command2" scriptName="Script_Name_02" label=" Command_Label_01" mode="manual" action=
33 "function1(x)">
34 <VARIABLES>
35 <VARIABLE name="var1" default="valuevar1"/>
36 </VARIABLES>
37 <OUTPUTS>
38 <OUTPUT type="text" name="c" value="z" function="function2(y,z)" mode=" manual" label=
39 "Output_Label_01"/>
40 </OUTPUTS>
41 </COMMAND>
42 <COMMAND name="command3" scriptName="Script_Name_03" label=" Command_Label_03" mode="manual" action=
43 "action_to_call">
44 <OUTPUTS>
45 <OUTPUT type="text" name="e" value="z2" mode="manual" label=" Output_Label_01">
46 <VARIABLES>
47 <VARIABLE name="var2" default="valuevar2"/> </VARIABLES>
48 </OUTPUT>
49 <OUTPUT type="image" name="f" value="" function="rectf(z)" mode="auto" label="Output_Label_02
50 </OUTPUTS>
51 </COMMAND>
52 </COMMANDS>
53 </DATA_MINING>

```

As you can see in the example, there are six basic tags:

- **COMMANDS**: the leading objects. They call a script execution and can have multiple outputs. They enable interactive document execution where only command in mode="auto" is executed automatically. The mode="manual" requires the user's click.
- **OUTPUTS**: to define which results have to be shown. They work with the Images. Text is the string representation of the script result, while *Image* is the chart generated by R. There are also predefined functions (histogram, plot, biplot) or developer's functions that generate the output recalled by function.
- **SCRIPTS**: they contain the R script (including objects definitions, pre-processing and functions). There can be many scripts depending on commands. The main function execution can be recalled (if needed) by the action

attribute. The main script is executed once. Outputs will look for the objects in the user's workspace.

- **DATASETS:** the data used by the scripts. They are executed at the beginning of the document's execution so that data.frames can be used further by every script. There are two dataset types:
 - file: csv, delim, text, etc., manually loaded by the end user at document execution time;
 - Knowage datasets: defined by label in document's template, whose resultset is converted in CSV. They can use analytical drivers.
- **PARAMETERS:** they corresponds to Knowage analytical drivers and can influence the behaviour of the Knowage dataset. They cannot be applied to other components.
- **VARIABLES:** are required for changing factors or more generally parameters (strings or numbers) inside the script (referenced by a command) or the output functions.

Once the template has been edited it can be upload on Knowage server to create a usable Data Mining document. Enter then Knowage document browser and click on the “Plus” icon. The insert all mandatory fields as label, name, engine and datasource. Then you must upload the template file clicking on the icon available at the bottom of the form, highlighted below.



Fig. 5.383: Creating a new function.

5.14.5 Create a new function in the Function Catalog

To create a new Function you must click on the “Plus” icon available at the right top side of the page. The action will provoke the opening of the window in figure below made up of 4 tabs.

- **General:** here you have to set the Function name, the label which identifies the function univocally, the name of the user who creates the function, the type to which the function belongs to and a brief description of the function usage.
- **Input:**



Use the icon  to insert a new dataset or a new variable. And use the icon  to delete the insertion. Choose a dataset from the combobox and use the “Preview button” to check the outcome. While for the variables you must specify the variable name and value. An example is give in figure below.



SAVECLOSE

GENERALINPUTSCRIPTOUTPUT

Function Name

Label

Owner
demoadmin

Type

Description

B I U X₂ X₃ X₄ X₅ X₆ X₇ X₈ X₉ X₁₀ X₁₁ X₁₂ X₁₃ X₁₄ X₁₅ X₁₆ X₁₇ X₁₈ X₁₉ X₂₀ X₂₁ X₂₂ X₂₃ X₂₄ X₂₅ X₂₆ X₂₇ X₂₈ X₂₉ X₃₀ X₃₁ X₃₂ X₃₃ X₃₄ X₃₅ X₃₆ X₃₇ X₃₈ X₃₉ X₄₀ X₄₁ X₄₂ X₄₃ X₄₄ X₄₅ X₄₆ X₄₇ X₄₈ X₄₉ X₅₀ X₅₁ X₅₂ X₅₃ X₅₄ X₅₅ X₅₆ X₅₇ X₅₈ X₅₉ X₆₀ X₆₁ X₆₂ X₆₃ X₆₄ X₆₅ X₆₆ X₆₇ X₆₈ X₆₉ X₇₀ X₇₁ X₇₂ X₇₃ X₇₄ X₇₅ X₇₆ X₇₇ X₇₈ X₇₉ X₈₀ X₈₁ X₈₂ X₈₃ X₈₄ X₈₅ X₈₆ X₈₇ X₈₈ X₈₉ X₉₀ X₉₁ X₉₂ X₉₃ X₉₄ X₉₅ X₉₆ X₉₇ X₉₈ X₉₉ X₁₀₀ X₁₀₁ X₁₀₂ X₁₀₃ X₁₀₄ X₁₀₅ X₁₀₆ X₁₀₇ X₁₀₈ X₁₀₉ X₁₁₀ X₁₁₁ X₁₁₂ X₁₁₃ X₁₁₄ X₁₁₅ X₁₁₆ X₁₁₇ X₁₁₈ X₁₁₉ X₁₂₀ X₁₂₁ X₁₂₂ X₁₂₃ X₁₂₄ X₁₂₅ X₁₂₆ X₁₂₇ X₁₂₈ X₁₂₉ X₁₃₀ X₁₃₁ X₁₃₂ X₁₃₃ X₁₃₄ X₁₃₅ X₁₃₆ X₁₃₇ X₁₃₈ X₁₃₉ X₁₄₀ X₁₄₁ X₁₄₂ X₁₄₃ X₁₄₄ X₁₄₅ X₁₄₆ X₁₄₇ X₁₄₈ X₁₄₉ X₁₅₀ X₁₅₁ X₁₅₂ X₁₅₃ X₁₅₄ X₁₅₅ X₁₅₆ X₁₅₇ X₁₅₈ X₁₅₉ X₁₆₀ X₁₆₁ X₁₆₂ X₁₆₃ X₁₆₄ X₁₆₅ X₁₆₆ X₁₆₇ X₁₆₈ X₁₆₉ X₁₇₀ X₁₇₁ X₁₇₂ X₁₇₃ X₁₇₄ X₁₇₅ X₁₇₆ X₁₇₇ X₁₇₈ X₁₇₉ X₁₈₀ X₁₈₁ X₁₈₂ X₁₈₃ X₁₈₄ X₁₈₅ X₁₈₆ X₁₈₇ X₁₈₈ X₁₈₉ X₁₉₀ X₁₉₁ X₁₉₂ X₁₉₃ X₁₉₄ X₁₉₅ X₁₉₆ X₁₉₇ X₁₉₈ X₁₉₉ X₂₀₀ X₂₀₁ X₂₀₂ X₂₀₃ X₂₀₄ X₂₀₅ X₂₀₆ X₂₀₇ X₂₀₈ X₂₀₉ X₂₁₀ X₂₁₁ X₂₁₂ X₂₁₃ X₂₁₄ X₂₁₅ X₂₁₆ X₂₁₇ X₂₁₈ X₂₁₉ X₂₂₀ X₂₂₁ X₂₂₂ X₂₂₃ X₂₂₄ X₂₂₅ X₂₂₆ X₂₂₇ X₂₂₈ X₂₂₉ X₂₃₀ X₂₃₁ X₂₃₂ X₂₃₃ X₂₃₄ X₂₃₅ X₂₃₆ X₂₃₇ X₂₃₈ X₂₃₉ X₂₄₀ X₂₄₁ X₂₄₂ X₂₄₃ X₂₄₄ X₂₄₅ X₂₄₆ X₂₄₇ X₂₄₈ X₂₄₉ X₂₅₀ X₂₅₁ X₂₅₂ X₂₅₃ X₂₅₄ X₂₅₅ X₂₅₆ X₂₅₇ X₂₅₈ X₂₅₉ X₂₆₀ X₂₆₁ X₂₆₂ X₂₆₃ X₂₆₄ X₂₆₅ X₂₆₆ X₂₆₇ X₂₆₈ X₂₆₉ X₂₇₀ X₂₇₁ X₂₇₂ X₂₇₃ X₂₇₄ X₂₇₅ X₂₇₆ X₂₇₇ X₂₇₈ X₂₇₉ X₂₈₀ X₂₈₁ X₂₈₂ X₂₈₃ X₂₈₄ X₂₈₅ X₂₈₆ X₂₈₇ X₂₈₈ X₂₈₉ X₂₉₀ X₂₉₁ X₂₉₂ X₂₉₃ X₂₉₄ X₂₉₅ X₂₉₆ X₂₉₇ X₂₉₈ X₂₉₉ X₃₀₀ X₃₀₁ X₃₀₂ X₃₀₃ X₃₀₄ X₃₀₅ X₃₀₆ X₃₀₇ X₃₀₈ X₃₀₉ X₃₁₀ X₃₁₁ X₃₁₂ X₃₁₃ X₃₁₄ X₃₁₅ X₃₁₆ X₃₁₇ X₃₁₈ X₃₁₉ X₃₂₀ X₃₂₁ X₃₂₂ X₃₂₃ X₃₂₄ X₃₂₅ X₃₂₆ X₃₂₇ X₃₂₈ X₃₂₉ X₃₃₀ X₃₃₁ X₃₃₂ X₃₃₃ X₃₃₄ X₃₃₅ X₃₃₆ X₃₃₇ X₃₃₈ X₃₃₉ X₃₄₀ X₃₄₁ X₃₄₂ X₃₄₃ X₃₄₄ X₃₄₅ X₃₄₆ X₃₄₇ X₃₄₈ X₃₄₉ X₃₅₀ X₃₅₁ X₃₅₂ X₃₅₃ X₃₅₄ X₃₅₅ X₃₅₆ X₃₅₇ X₃₅₈ X₃₅₉ X₃₆₀ X₃₆₁ X₃₆₂ X₃₆₃ X₃₆₄ X₃₆₅ X₃₆₆ X₃₆₇ X₃₆₈ X₃₆₉ X₃₇₀ X₃₇₁ X₃₇₂ X₃₇₃ X₃₇₄ X₃₇₅ X₃₇₆ X₃₇₇ X₃₇₈ X₃₇₉ X₃₈₀ X₃₈₁ X₃₈₂ X₃₈₃ X₃₈₄ X₃₈₅ X₃₈₆ X₃₈₇ X₃₈₈ X₃₈₉ X₃₉₀ X₃₉₁ X₃₉₂ X₃₉₃ X₃₉₄ X₃₉₅ X₃₉₆ X₃₉₇ X₃₉₈ X₃₉₉ X₄₀₀ X₄₀₁ X₄₀₂ X₄₀₃ X₄₀₄ X₄₀₅ X₄₀₆ X₄₀₇ X₄₀₈ X₄₀₉ X₄₁₀ X₄₁₁ X₄₁₂ X₄₁₃ X₄₁₄ X₄₁₅ X₄₁₆ X₄₁₇ X₄₁₈ X₄₁₉ X₄₂₀ X₄₂₁ X₄₂₂ X₄₂₃ X₄₂₄ X₄₂₅ X₄₂₆ X₄₂₇ X₄₂₈ X₄₂₉ X₄₃₀ X₄₃₁ X₄₃₂ X₄₃₃ X₄₃₄ X₄₃₅ X₄₃₆ X₄₃₇ X₄₃₈ X₄₃₉ X₄₄₀ X₄₄₁ X₄₄₂ X₄₄₃ X₄₄₄ X₄₄₅ X₄₄₆ X₄₄₇ X₄₄₈ X₄₄₉ X₄₅₀ X₄₅₁ X₄₅₂ X₄₅₃ X₄₅₄ X₄₅₅ X₄₅₆ X₄₅₇ X₄₅₈ X₄₅₉ X₄₆₀ X₄₆₁ X₄₆₂ X₄₆₃ X₄₆₄ X₄₆₅ X₄₆₆ X₄₆₇ X₄₆₈ X₄₆₉ X₄₇₀ X₄₇₁ X₄₇₂ X₄₇₃ X₄₇₄ X₄₇₅ X₄₇₆ X₄₇₇ X₄₇₈ X₄₇₉ X₄₈₀ X₄₈₁ X₄₈₂ X₄₈₃ X₄₈₄ X₄₈₅ X₄₈₆ X₄₈₇ X₄₈₈ X₄₈₉ X₄₉₀ X₄₉₁ X₄₉₂ X₄₉₃ X₄₉₄ X₄₉₅ X₄₉₆ X₄₉₇ X₄₉₈ X₄₉₉ X₅₀₀ X₅₀₁ X₅₀₂ X₅₀₃ X₅₀₄ X₅₀₅ X₅₀₆ X₅₀₇ X₅₀₈ X₅₀₉ X₅₁₀ X₅₁₁ X₅₁₂ X₅₁₃ X₅₁₄ X₅₁₅ X₅₁₆ X₅₁₇ X₅₁₈ X₅₁₉ X₅₂₀ X₅₂₁ X₅₂₂ X₅₂₃ X₅₂₄ X₅₂₅ X₅₂₆ X₅₂₇ X₅₂₈ X₅₂₉ X₅₃₀ X₅₃₁ X₅₃₂ X₅₃₃ X₅₃₄ X₅₃₅ X₅₃₆ X₅₃₇ X₅₃₈ X₅₃₉ X₅₄₀ X₅₄₁ X₅₄₂ X₅₄₃ X₅₄₄ X₅₄₅ X₅₄₆ X₅₄₇ X₅₄₈ X₅₄₉ X₅₅₀ X₅₅₁ X₅₅₂ X₅₅₃ X₅₅₄ X₅₅₅ X₅₅₆ X₅₅₇ X₅₅₈ X₅₅₉ X₅₆₀ X₅₆₁ X₅₆₂ X₅₆₃ X₅₆₄ X₅₆₅ X₅₆₆ X₅₆₇ X₅₆₈ X₅₆₉ X₅₇₀ X₅₇₁ X₅₇₂ X₅₇₃ X₅₇₄ X₅₇₅ X₅₇₆ X₅₇₇ X₅₇₈ X₅₇₉ X₅₈₀ X₅₈₁ X₅₈₂ X₅₈₃ X₅₈₄ X₅₈₅ X₅₈₆ X₅₈₇ X₅₈₈ X₅₈₉ X₅₉₀ X₅₉₁ X₅₉₂ X₅₉₃ X₅₉₄ X₅₉₅ X₅₉₆ X₅₉₇ X₅₉₈ X₅₉₉ X₆₀₀ X₆₀₁ X₆₀₂ X₆₀₃ X₆₀₄ X₆₀₅ X₆₀₆ X₆₀₇ X₆₀₈ X₆₀₉ X₆₁₀ X₆₁₁ X₆₁₂ X₆₁₃ X₆₁₄ X₆₁₅ X₆₁₆ X₆₁₇ X₆₁₈ X₆₁₉ X₆₂₀ X₆₂₁ X₆₂₂ X₆₂₃ X₆₂₄ X₆₂₅ X₆₂₆ X₆₂₇ X₆₂₈ X₆₂₉ X₆₃₀ X₆₃₁ X₆₃₂ X₆₃₃ X₆₃₄ X₆₃₅ X₆₃₆ X₆₃₇ X₆₃₈ X₆₃₉ X₆₄₀ X₆₄₁ X₆₄₂ X₆₄₃ X₆₄₄ X₆₄₅ X₆₄₆ X₆₄₇ X₆₄₈ X₆₄₉ X₆₅₀ X₆₅₁ X₆₅₂ X₆₅₃ X₆₅₄ X₆₅₅ X₆₅₆ X₆₅₇ X₆₅₈ X₆₅₉ X₆₆₀ X₆₆₁ X₆₆₂ X₆₆₃ X₆₆₄ X₆₆₅ X₆₆₆ X₆₆₇ X₆₆₈ X₆₆₉ X₆₇₀ X₆₇₁ X₆₇₂ X₆₇₃ X₆₇₄ X₆₇₅ X₆₇₆ X₆₇₇ X₆₇₈ X₆₇₉ X₆₈₀ X₆₈₁ X₆₈₂ X₆₈₃ X₆₈₄ X₆₈₅ X₆₈₆ X₆₈₇ X₆₈₈ X₆₈₉ X₆₉₀ X₆₉₁ X₆₉₂ X₆₉₃ X₆₉₄ X₆₉₅ X₆₉₆ X₆₉₇ X₆₉₈ X₆₉₉ X₇₀₀ X₇₀₁ X₇₀₂ X₇₀₃ X₇₀₄ X₇₀₅ X₇₀₆ X₇₀₇ X₇₀₈ X₇₀₉ X₇₁₀ X₇₁₁ X₇₁₂ X₇₁₃ X₇₁₄ X₇₁₅ X₇₁₆ X₇₁₇ X₇₁₈ X₇₁₉ X₇₂₀ X₇₂₁ X₇₂₂ X₇₂₃ X₇₂₄ X₇₂₅ X₇₂₆ X₇₂₇ X₇₂₈ X₇₂₉ X₇₃₀ X₇₃₁ X₇₃₂ X₇₃₃ X₇₃₄ X₇₃₅ X₇₃₆ X₇₃₇ X₇₃₈ X₇₃₉ X₇₄₀ X₇₄₁ X₇₄₂ X₇₄₃ X₇₄₄ X₇₄₅ X₇₄₆ X₇₄₇ X₇₄₈ X₇₄₉ X₇₅₀ X₇₅₁ X₇₅₂ X₇₅₃ X₇₅₄ X₇₅₅ X₇₅₆ X₇₅₇ X₇₅₈ X₇₅₉ X₇₆₀ X₇₆₁ X₇₆₂ X₇₆₃ X₇₆₄ X₇₆₅ X₇₆₆ X₇₆₇ X₇₆₈ X₇₆₉ X₇₇₀ X₇₇₁ X₇₇₂ X₇₇₃ X₇₇₄ X₇₇₅ X₇₇₆ X₇₇₇ X₇₇₈ X₇₇₉ X₇₈₀ X₇₈₁ X₇₈₂ X₇₈₃ X₇₈₄ X₇₈₅ X₇₈₆ X₇₈₇ X₇₈₈ X₇₈₉ X₇₉₀ X₇₉₁ X₇₉₂ X₇₉₃ X₇₉₄ X₇₉₅ X₇₉₆ X₇₉₇ X₇₉₈ X₇₉₉ X₈₀₀ X₈₀₁ X₈₀₂ X₈₀₃ X₈₀₄ X₈₀₅ X₈₀₆ X₈₀₇ X₈₀₈ X₈₀₉ X₈₁₀ X₈₁₁ X₈₁₂ X₈₁₃ X₈₁₄ X₈₁₅ X₈₁₆ X₈₁₇ X₈₁₈ X₈₁₉ X₈₂₀ X₈₂₁ X₈₂₂ X₈₂₃ X₈₂₄ X₈₂₅ X₈₂₆ X₈₂₇ X₈₂₈ X₈₂₉ X₈₃₀ X₈₃₁ X₈₃₂ X₈₃₃ X₈₃₄ X₈₃₅ X₈₃₆ X₈₃₇ X₈₃₈ X₈₃₉ X₈₄₀ X₈₄₁ X₈₄₂ X₈₄₃ X₈₄₄ X₈₄₅ X₈₄₆ X₈₄₇ X₈₄₈ X₈₄₉ X₈₅₀ X₈₅₁ X₈₅₂ X₈₅₃ X₈₅₄ X₈₅₅ X₈₅₆ X₈₅₇ X₈₅₈ X₈₅₉ X₈₆₀ X₈₆₁ X₈₆₂ X₈₆₃ X₈₆₄ X₈₆₅ X₈₆₆ X₈₆₇ X₈₆₈ X₈₆₉ X₈₇₀ X₈₇₁ X₈₇₂ X₈₇₃ X₈₇₄ X₈₇₅ X₈₇₆ X₈₇₇ X₈₇₈ X₈₇₉ X₈₈₀ X₈₈₁ X₈₈₂ X₈₈₃ X₈₈₄ X₈₈₅ X₈₈₆ X₈₈₇ X₈₈₈ X₈₈₉ X₈₉₀ X₈₉₁ X₈₉₂ X₈₉₃ X₈₉₄ X₈₉₅ X₈₉₆ X₈₉₇ X₈₉₈ X₈₉₉ X₉₀₀ X₉₀₁ X₉₀₂ X₉₀₃ X₉₀₄ X₉₀₅ X₉₀₆ X₉₀₇ X₉₀₈ X₉₀₉ X₉₁₀ X₉₁₁ X₉₁₂ X₉₁₃ X₉₁₄ X₉₁₅ X₉₁₆ X₉₁₇ X₉₁₈ X₉₁₉ X₉₂₀ X₉₂₁ X₉₂₂ X₉₂₃ X₉₂₄ X₉₂₅ X₉₂₆ X₉₂₇ X₉₂₈ X₉₂₉ X₉₃₀ X₉₃₁ X₉₃₂ X₉₃₃ X₉₃₄ X₉₃₅ X₉₃₆ X₉₃₇ X₉₃₈ X₉₃₉ X₉₄₀ X₉₄₁ X₉₄₂ X₉₄₃ X₉₄₄ X₉₄₅ X₉₄₆ X₉₄₇ X₉₄₈ X₉₄₉ X₉₅₀ X₉₅₁ X₉₅₂ X₉₅₃ X₉₅₄ X₉₅₅ X₉₅₆ X₉₅₇ X₉₅₈ X₉₅₉ X₉₆₀ X₉₆₁ X₉₆₂ X₉₆₃ X₉₆₄ X₉₆₅ X₉₆₆ X₉₆₇ X₉₆₈ X₉₆₉ X₉₇₀ X₉₇₁ X₉₇₂ X₉₇₃ X₉₇₄ X₉₇₅ X₉₇₆ X₉₇₇ X₉₇₈ X₉₇₉ X₉₈₀ X₉₈₁ X₉₈₂ X₉₈₃ X₉₈₄ X₉₈₅ X₉₈₆ X₉₈₇ X₉₈₈ X₉₈₉ X₉₉₀ X₉₉₁ X₉₉₂ X₉₉₃ X₉₉₄ X₉₉₅ X₉₉₆ X₉₉₇ X₉₉₈ X₉₉₉ X₁₀₀₀ X₁₀₀₁ X₁₀₀₂ X₁₀₀₃ X₁₀₀₄ X₁₀₀₅ X₁₀₀₆ X₁₀₀₇ X₁₀₀₈ X₁₀₀₉ X₁₀₁₀ X₁₀₁₁ X₁₀₁₂ X₁₀₁₃ X₁₀₁₄ X₁₀₁₅ X₁₀₁₆ X₁₀₁₇ X₁₀₁₈ X₁₀₁₉ X₁₀₂₀ X₁₀₂₁ X₁₀₂₂ X₁₀₂₃ X₁₀₂₄ X₁₀₂₅ X₁₀₂₆ X₁₀₂₇ X₁₀₂₈ X₁₀₂₉ X₁₀₃₀ X₁₀₃₁ X₁₀₃₂ X₁₀₃₃ X₁₀₃₄ X₁₀₃₅ X₁₀₃₆ X₁₀₃₇ X₁₀₃₈ X₁₀₃₉ X₁₀₄₀ X₁₀₄₁ X₁₀₄₂ X₁₀₄₃ X₁₀₄₄ X₁₀₄₅ X₁₀₄₆ X₁₀₄₇ X₁₀₄₈ X₁₀₄₉ X₁₀₅₀ X₁₀₅₁ X₁₀₅₂ X₁₀₅₃ X₁₀₅₄ X₁₀₅₅ X₁₀₅₆ X₁₀₅₇ X₁₀₅₈ X₁₀₅₉ X₁₀₆₀ X₁₀₆₁ X₁₀₆₂ X₁₀₆₃ X₁₀₆₄ X₁₀₆₅ X₁₀₆₆ X₁₀₆₇ X₁₀₆₈ X₁₀₆₉ X₁₀₇₀ X₁₀₇₁ X₁₀₇₂ X₁₀₇₃ X₁₀₇₄ X₁₀₇₅ X₁₀₇₆ X₁₀₇₇ X₁₀₇₈ X₁₀₇₉ X₁₀₈₀ X₁₀₈₁ X₁₀₈₂ X₁₀₈₃ X₁₀₈₄ X₁₀₈₅ X₁₀₈₆ X₁₀₈₇ X₁₀₈₈ X₁₀₈₉ X₁₀₉₀ X₁₀₉₁ X₁₀₉₂ X₁₀₉₃ X₁₀₉₄ X₁₀₉₅ X₁₀₉₆ X₁₀₉₇ X₁₀₉₈ X₁₀₉₉ X₁₁₀₀ X₁₁₀₁ X₁₁₀₂ X₁₁₀₃ X₁₁₀₄ X₁₁₀₅ X₁₁₀₆ X₁₁₀₇ X₁₁₀₈ X₁₁₀₉ X₁₁₁₀ X₁₁₁₁ X₁₁₁₂ X₁₁₁₃ X₁₁₁₄ X₁₁₁₅ X₁₁₁₆ X₁₁₁₇ X₁₁₁₈ X₁₁₁₉ X₁₁₂₀ X₁₁₂₁ X₁₁₂₂ X₁₁₂₃ X₁₁₂₄ X₁₁₂₅ X₁₁₂₆ X₁₁₂₇ X₁₁₂₈ X₁₁₂₉ X₁₁₃₀ X₁₁₃₁ X₁₁₃₂ X₁₁₃₃ X₁₁₃₄ X₁₁₃₅ X₁₁₃₆ X₁₁₃₇ X₁₁₃₈ X₁₁₃₉ X₁₁₄₀ X₁₁₄₁ X₁₁₄₂ X₁₁₄₃ X₁₁₄₄ X₁₁₄₅ X₁₁₄₆ X₁₁₄₇ X₁₁₄₈ X₁₁₄₉ X₁₁₅₀ X₁₁₅₁ X₁₁₅₂ X₁₁₅₃ X₁₁₅₄ X₁₁₅₅ X₁₁₅₆ X₁₁₅₇ X₁₁₅₈ X₁₁₅₉ X₁₁₆₀ X₁₁₆₁ X₁₁₆₂ X₁₁₆₃ X₁₁₆₄ X₁₁₆₅ X₁₁₆₆ X₁₁₆₇ X₁₁₆₈ X₁₁₆₉ X₁₁₇₀ X₁₁₇₁ X₁₁₇₂ X₁₁₇₃ X₁₁₇₄ X₁₁₇₅ X₁₁₇₆ X₁₁₇₇ X₁₁₇₈ X₁₁₇₉ X₁₁₈₀ X₁₁₈₁ X₁₁₈₂ X₁₁₈₃ X₁₁₈₄ X₁₁₈₅ X₁₁₈₆ X₁₁₈₇ X₁₁₈₈ X₁₁₈₉ X₁₁₉₀ X₁₁₉₁ X₁₁₉₂ X₁₁₉₃ X₁₁₉₄ X₁₁₉₅ X₁₁₉₆ X₁₁₉₇ X₁₁₉₈ X₁₁₉₉ X₁₂₀₀ X₁₂₀₁ X₁₂₀₂ X₁₂₀₃ X₁₂₀₄ X₁₂₀₅ X₁₂₀₆ X₁₂₀₇ X₁₂₀₈ X₁₂₀₉ X₁₂₁₀ X₁₂₁₁ X₁₂₁₂ X₁₂₁₃ X₁₂₁₄ X₁₂₁₅ X₁₂₁₆

- **Script:** here is where the user is required to have knowledge of R or Python language. Figure below shows an example.

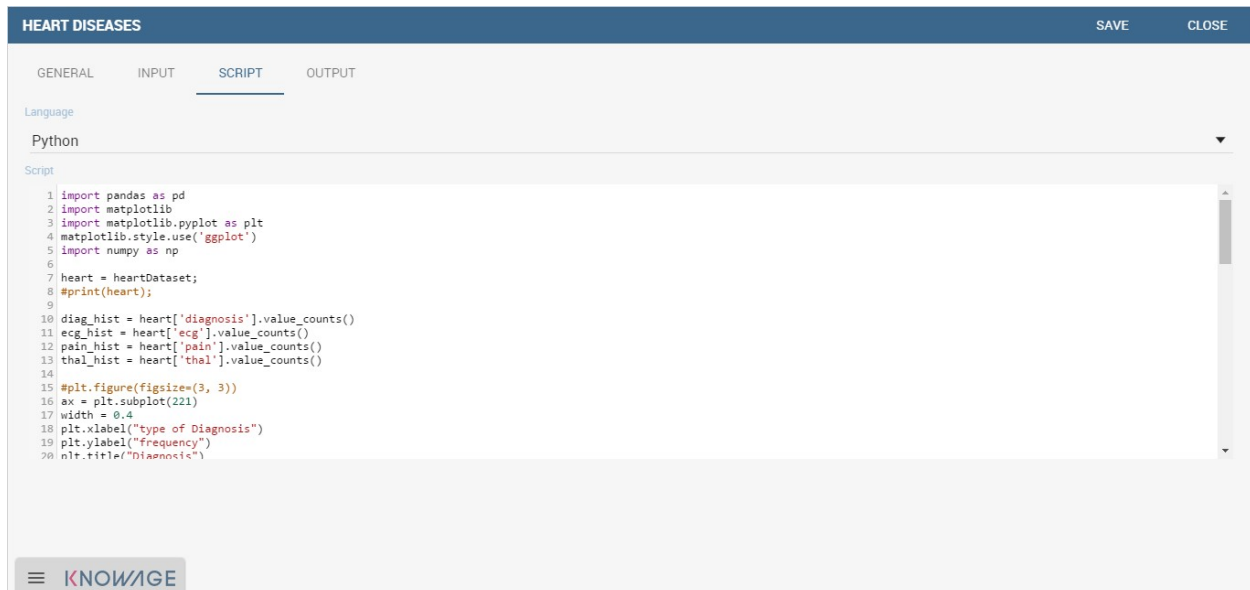




Fig. 5.387: Typing Python script

- **Output:** referring to the following figure, in the Output tab you have to choose how the output should be visualized. Still use the icon  to insert a new output and the icon  to delete the items. Then insert the output name and once again you can choose among “Text”, “Image”, “Dataset” for both Python and R.

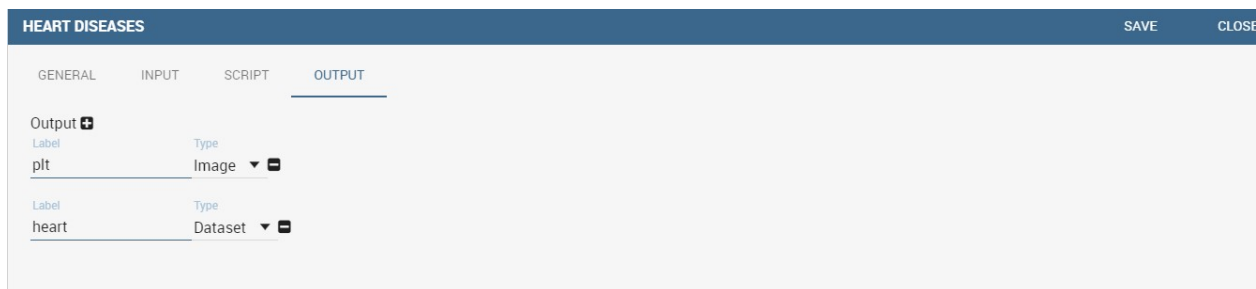


Fig. 5.388: Choosing how to visualise outputs.

Then save and you are ready to use the function.

5.15 Multidimensional Analysis

OLAP tools enable users to analyse multidimensional data interactively from multiple perspectives. OLAP consists of basic analytical operations: slice and dice, roll up, drill down and pivot.

5.15.1 OLAP user manual step by step

We start our lecture on the OLAP engine by analysing an existing OLAP document. Open the document browser folder of the Knowage suite as in figure below and launch an OLAP document.

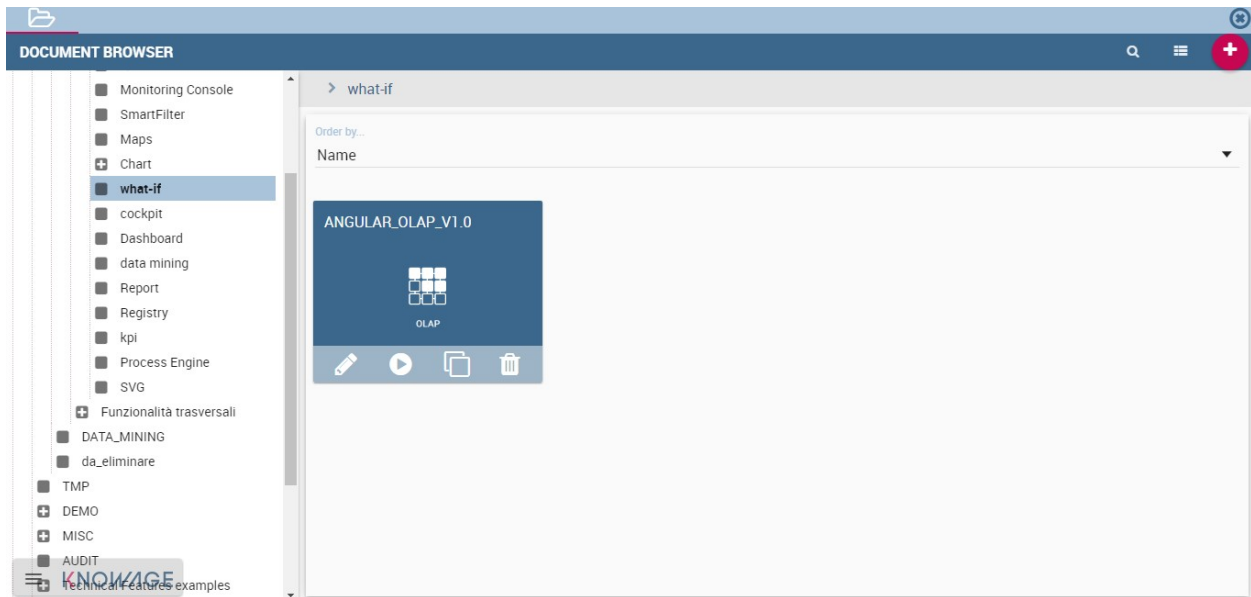


Fig. 5.389: Browse the documents and select an OLAP document.

Here an example.

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

5.15.1.1 The filter panel

Once the document is executed, the central area of the window contains the table whose measures are aggregated on dimensions. At the top of this area, panels are available to configure filters on attributes. We see in the following figure that the filter panel is made up of **Filter cards**. Here you can find the cube dimensions and their hierarchies as defined in the OLAP schema by the developer.

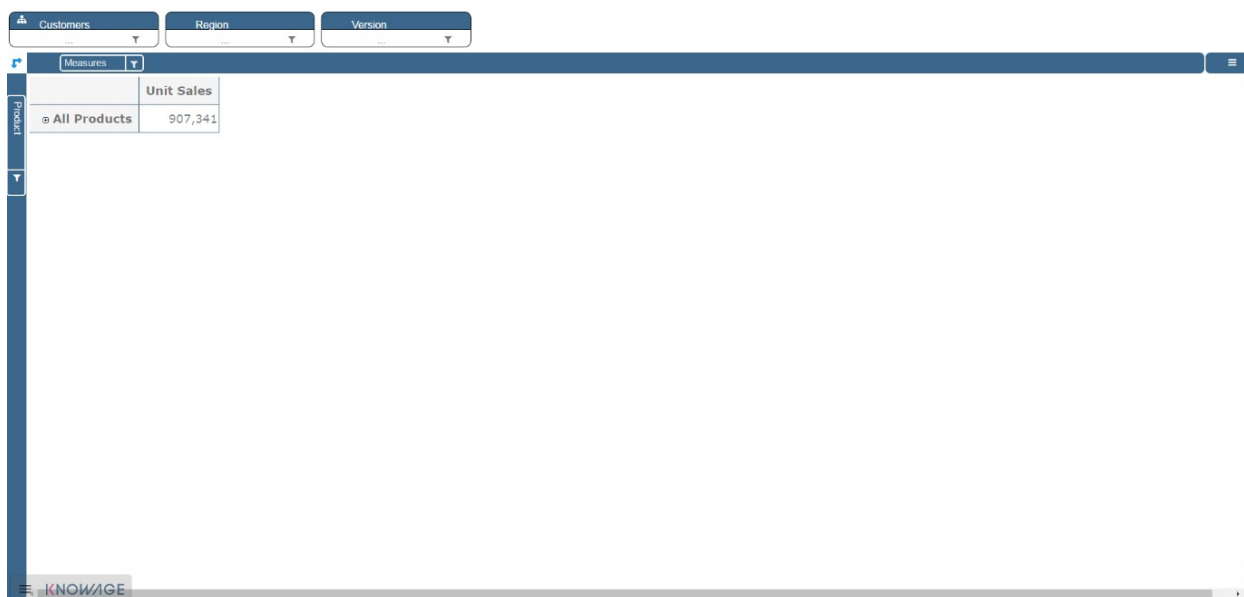
5.15.1.2 The filter cards

Filter cards can be placed on the filter panel or on column axis. You can switch their position dragging and dropping them from one place to the other.

Filter cards are used to:

- inform the user about available dimensions defined in OLAP schema,
- inform the user about dimension's name,
- perform slices,
- Add the dimensions to the cube visualization,
- place hierarchies in different axes,
- filter visible members.

Considering the next figure, we can see that a filter card is made up of:



The screenshot shows the Knowage OLAP interface. At the top, there are three filter panels: 'Customers', 'Region', and 'Version'. Below these is a 'Measures' panel with a dropdown menu. The main area displays a table with the following data:

	Unit Sales
All Products	907,341

The interface also includes a 'Product' sidebar on the left and a 'KNOWAGE' logo at the bottom left.

Fig. 5.390: Exploring an existing OLAP document.



The screenshot shows the filter panel with three filter cards: 'Customers', 'Region', and 'Version'. Each card has a dropdown arrow on the right side.

Fig. 5.391: The filter panel.

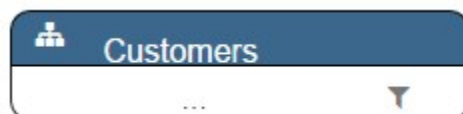


Fig. 5.392: The filter card inside the filter panel.

- an icon for opening dimension chooser dialog (a),
- a dimension name (b),
- icon slicer (c)+(e),
- choosed filter name (d),



Fig. 5.393: Features of a filter card.

5.15.1.3 Axes panel

In the panel axes you can:

- drag and drop one or more dimensions,
- organise the dimensions visualization,
- swap axes.

Referring to the following figure, the axes panel consists of the following items:

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

5.15.1.4 Pivot table

The Pivot table is the central part of the OLAP page. In figure below is shown an example.

Pivot table is used to:

- show data based on MDX query sent from the interface,
- drill down/up hierarchies' dimensions,
- drill through,
- show properties of a particular member,
- sort data,

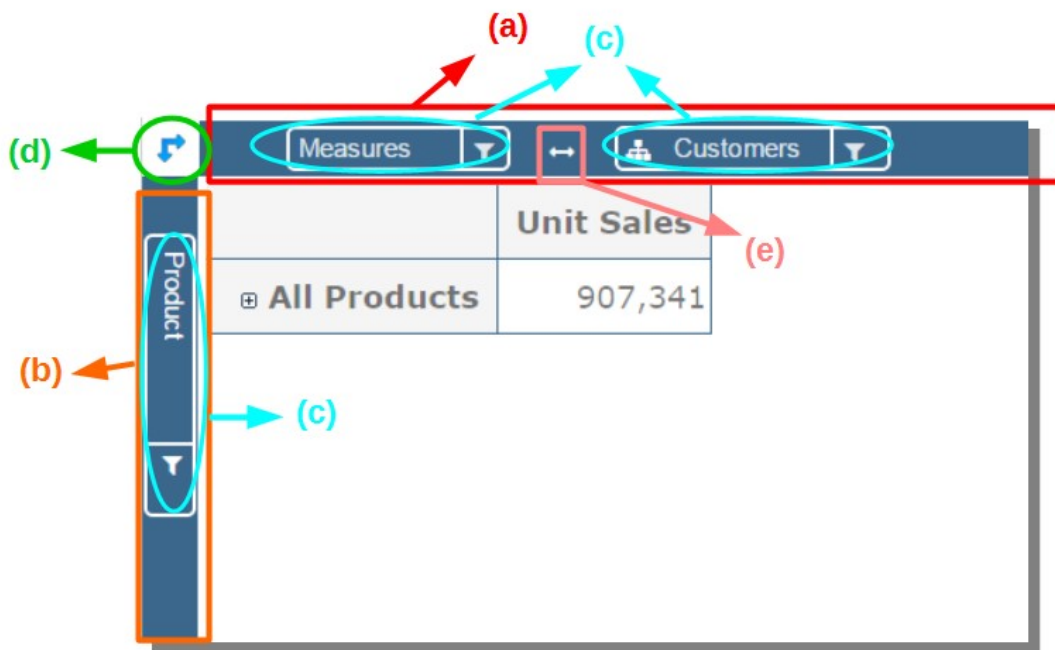


Fig. 5.394: Axes panel features.

	Unit Sales														
	All Regions	Canada West	Central West è	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

Fig. 5.395: Pivot table.

- show calculated fields,
- perform cross navigation to other documents.

Referring to next figure, Pivot table consists of:

- dimensions involved in the analysis (a),
- cells with data (b),
- icons for drill down and drill up (c),
- icons for sorting (only if enabled by the developer) (d),
- icons for showing properties (only if enabled and configured by the developer) (e),
- links for cross navigation (only if enabled and configured by the developer) (f).

	Unit Sales	Unit Sales	Unit Sales	Unit Sales
☐ All Regions	☐ Canada West	☐ Vancouver	☐ Store 19	
☐ All Products	907,341	81,698	64,858	64,858
☐ Drink	88,089	7,707	6,014	6,014
Dairy	14,397	1,301	1,018	1,018

Annotations: (a) points to the dimension headers; (b) points to the data cells; (c) points to the drill down/up icons; (d) points to the sorting icon; (e) points to the property icon; (f) points to the cross navigation link.

Fig. 5.396: Pivot table features.

5.15.1.5 Side bar

You can open the side bar by clicking on the icon positioned on the top right side of the page (see next figure). Side bar will be shown on the right side (see *Side bar* figure).

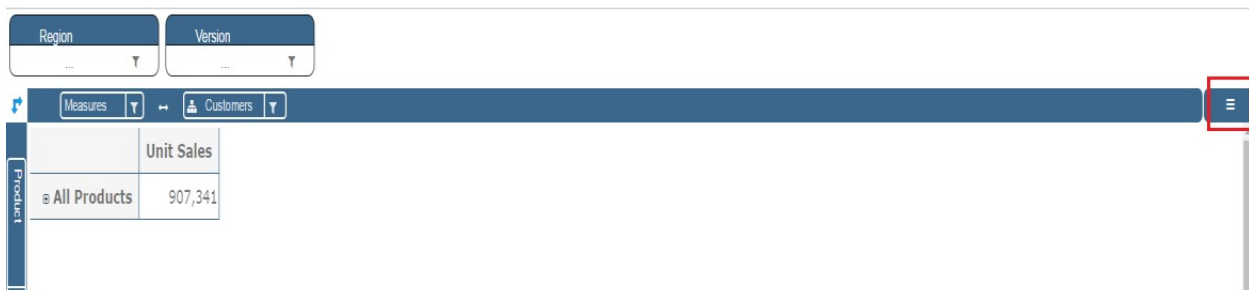


Fig. 5.397: Open the side bar.

Side bar is used to:

- choose between different data representations,

- choose between different drill types,
- call dialogs and functionalities that effect the pivot table,
- get additional data based on loaded model.

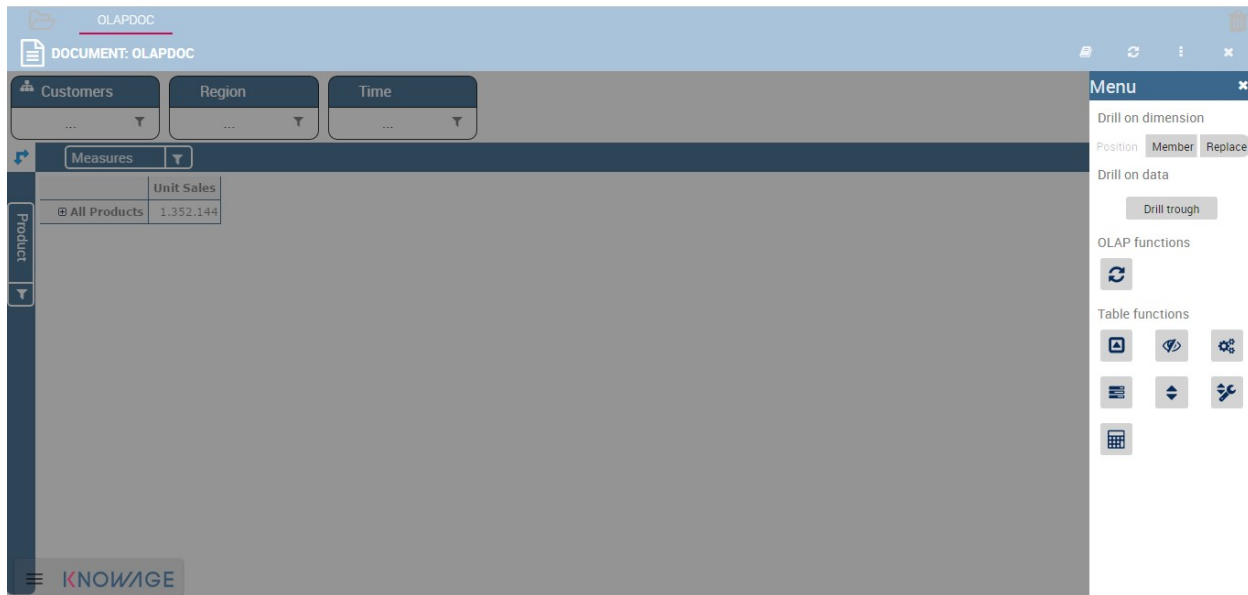


Fig. 5.398: Side bar.

The side bar shows the **Menu**. This area let you customize the Olap layout. As highlighted in the figure below, the Menu is divided in three subsections:

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

We start introducing the interface and leave the description to the next *Functionalities* paragraph. In particular, referring to next figure, drill types consists of:

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

Meanwhile, referring to the following figure, the OLAP functions consist of:

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

Referring to figure below, table functions consist of:

- show parent members (a),
- sorting settings (b),
- save customized view (c),

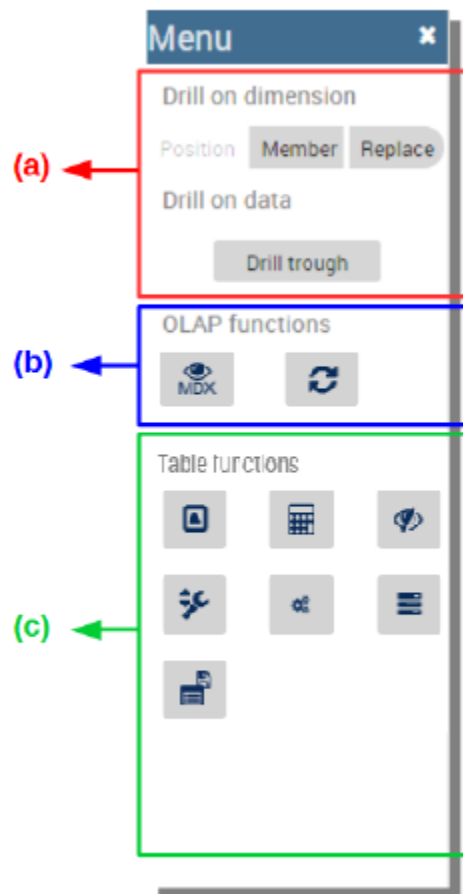


Fig. 5.399: Side bar Menu.

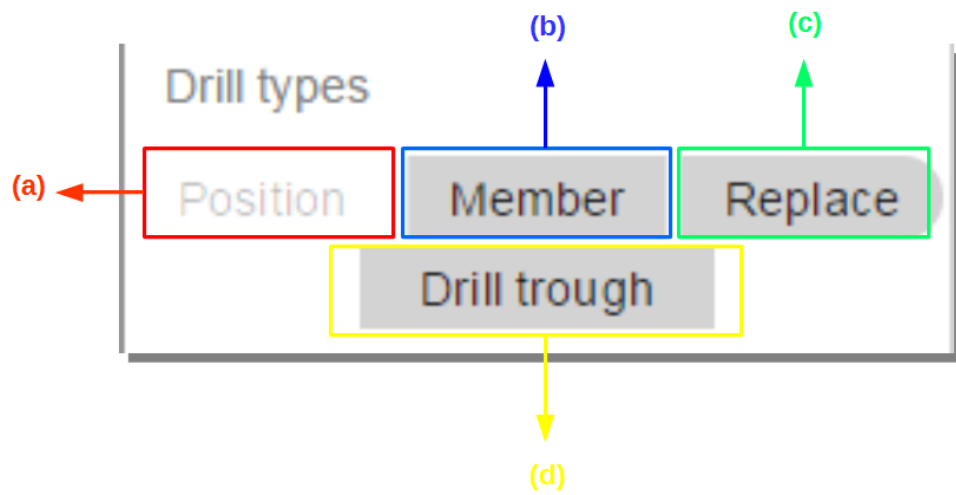


Fig. 5.400: Drill types.

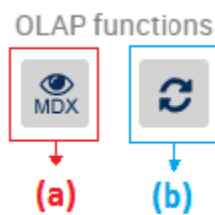


Fig. 5.401: OLAP functions.

- show properties (d),
- suppress empty rows/columns (e),
- hide spans (f),
- calculated field wizard (g).

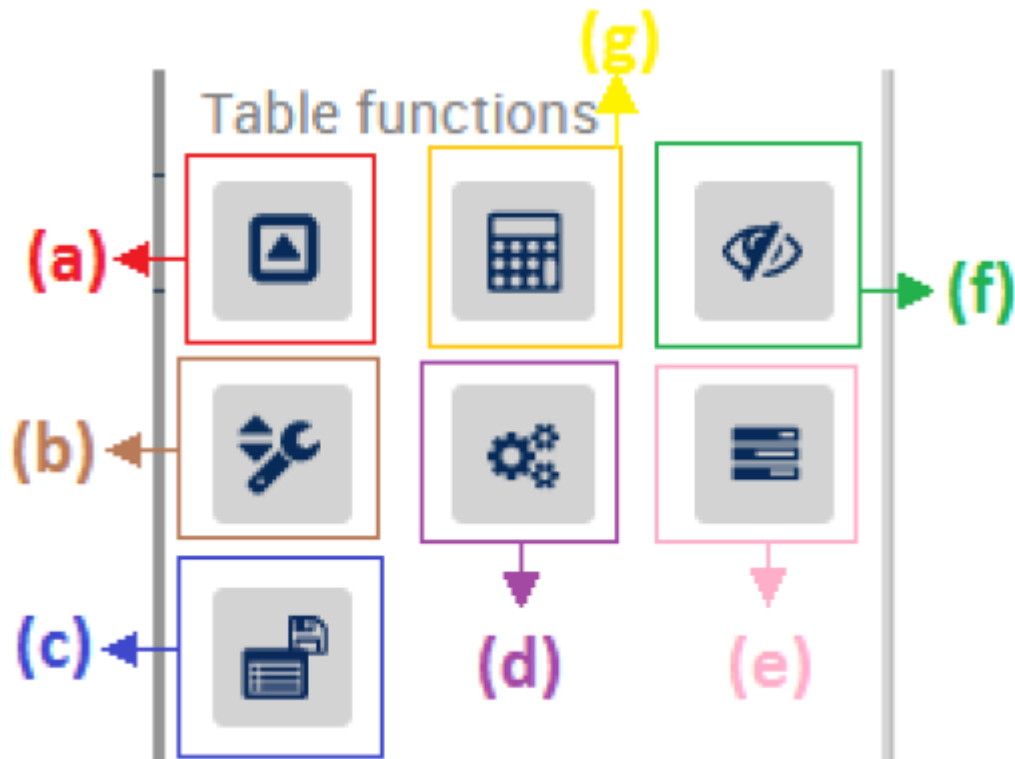


Fig. 5.402: Table functions.

Referring to next figure, what it consists of:

- lock/unlock model (a),
- delete versions (b),
- select an algorithm (c),
- output wizard (d),
- save as new version (e),
- undo (f),
- export excel for edit (g).



Fig. 5.403: Table functions.

5.15.2 Functionalities

5.15.2.1 Placing hierarchies on axes

As we already told, the user can easily move a dimension from the filter bar to the axis or viceversa dragging and dropping it to the desired place.

Let us suppose we want to move a dimension from the filter panel to the columns axis. The steps are summarized in figure below

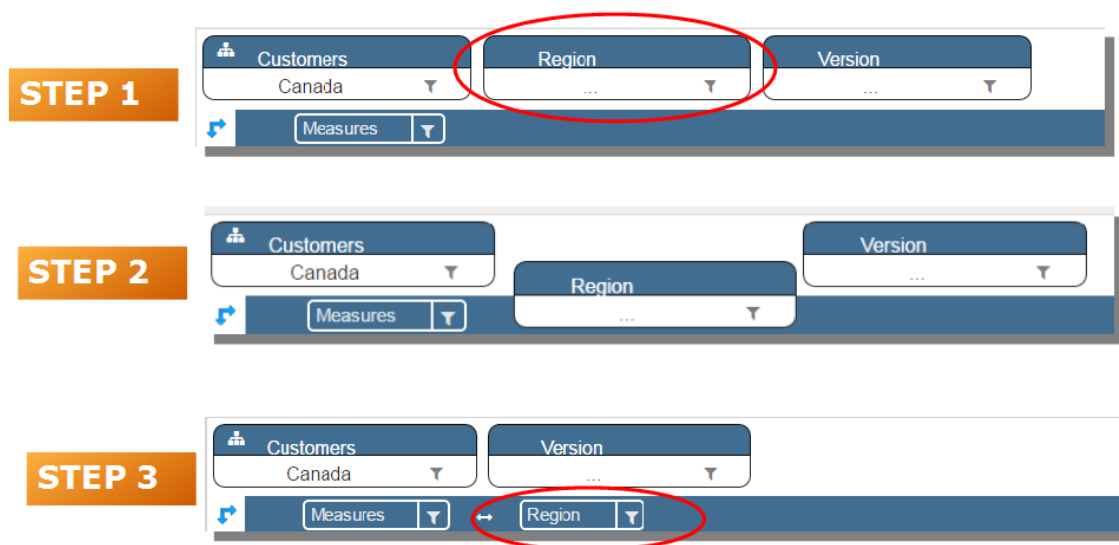



Fig. 5.404: Move a hierarchy to the columns axis.

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

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

5.15.2.2 Swapping axes

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

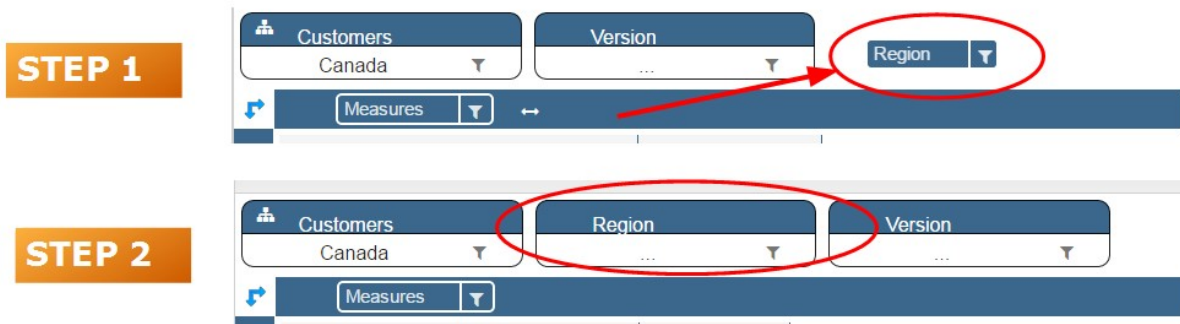


Fig. 5.405: Move a dimension from the columns axis to the filter panel.

Measures	
Product	Unit Sales
⊕ All Products	907,341

Product	
Measures	⊕ All Products
Unit Sales	907,341

Fig. 5.406: Swap axes.

5.15.2.3 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 below.



Fig. 5.407: Hierarchies icon.

A pop up will be displayed. The following figure shows its characteristics. The window will present:

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

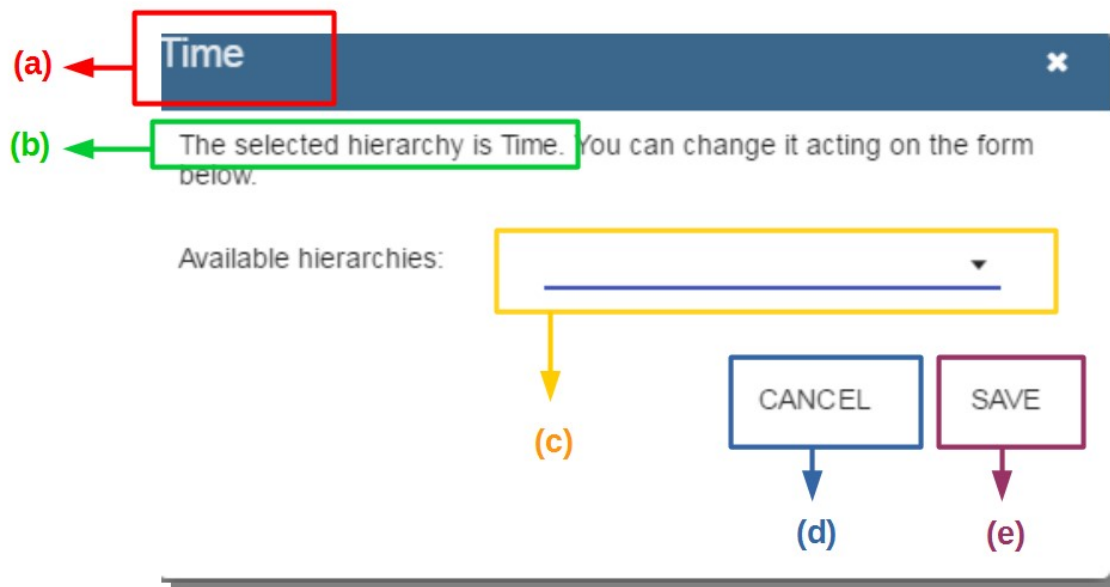


Fig. 5.408: Hierarchies dialog pop up.

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

If the user re-opens the dialog window, he/she sees the selected hierarchies and has the chance to change it if needed to, as shown below.

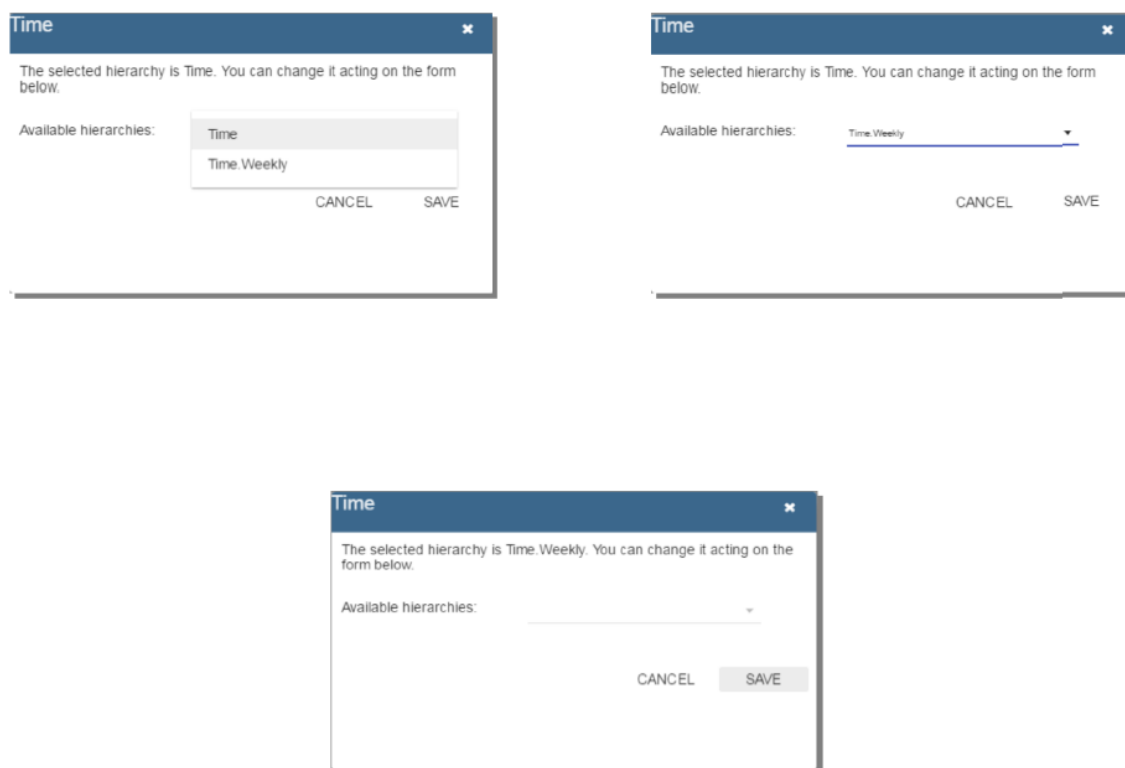


Fig. 5.409: Changing the hierarchies.

We give an example of the output when the hierarchy “Time” is selected in first next figure and hierarchy “Time Weekly” in the second next figure.

	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

Fig. 5.410: Time hierarchy: the table shows days in the month.

5.15.2.4 Slicing

The slicing operation consists in the analysis of a subset of a multi-dimensional array corresponding to a single value for one or more members of the dimensions. In order to perform this operation you need to drag and drop the dimension of interest in the axis panel. Then clicking on the filter icon choose the new single focus and apply it. Once concluded these steps the cube will show only the selected level of the dimension, while the others have been sliced out.

The following figure shows the slicer option panel which consists of:

- a dimension name (a),
- a search input field (b),
- a search button (c),
- a show/hide siblings checkbox (d),

	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

Fig. 5.411: Time Weekly hierachy: table shows weeks in the month.

- a member tree (e),
- a selected member icon (f),
- a highlighted member (result of searching) (g),
- a save and a cancel buttons (h).

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

1. by 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 (Fig. 5.412 (d))needs to be checked;
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.

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

5.15.2.5 Filtering

To filter dimension members in a pivot table, the user should click on a button (see Fig. 5.393) located on the right side of dimension's filter card placed in the filter area.

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 below. After selecting a member, the user should click on the save button. The pivot table will display the changements. Otherwise click on the cancel button to discard changes.

5.15.2.6 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. There are three drill types. In the following we give some details on them.

1. **Position:** this is the default drill type. 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 below.
2. **Member:** if the user wants to perform drill operation not only on one member per time but on all members of the same name and level at the same time it is needed to select member drill type. See below.
3. **Replace:** This option lets the user replace the parent member with his child member during drill down operation. To drill up the user should click on the arrow icon next to the dimension name on which to perform operation. See figure below.

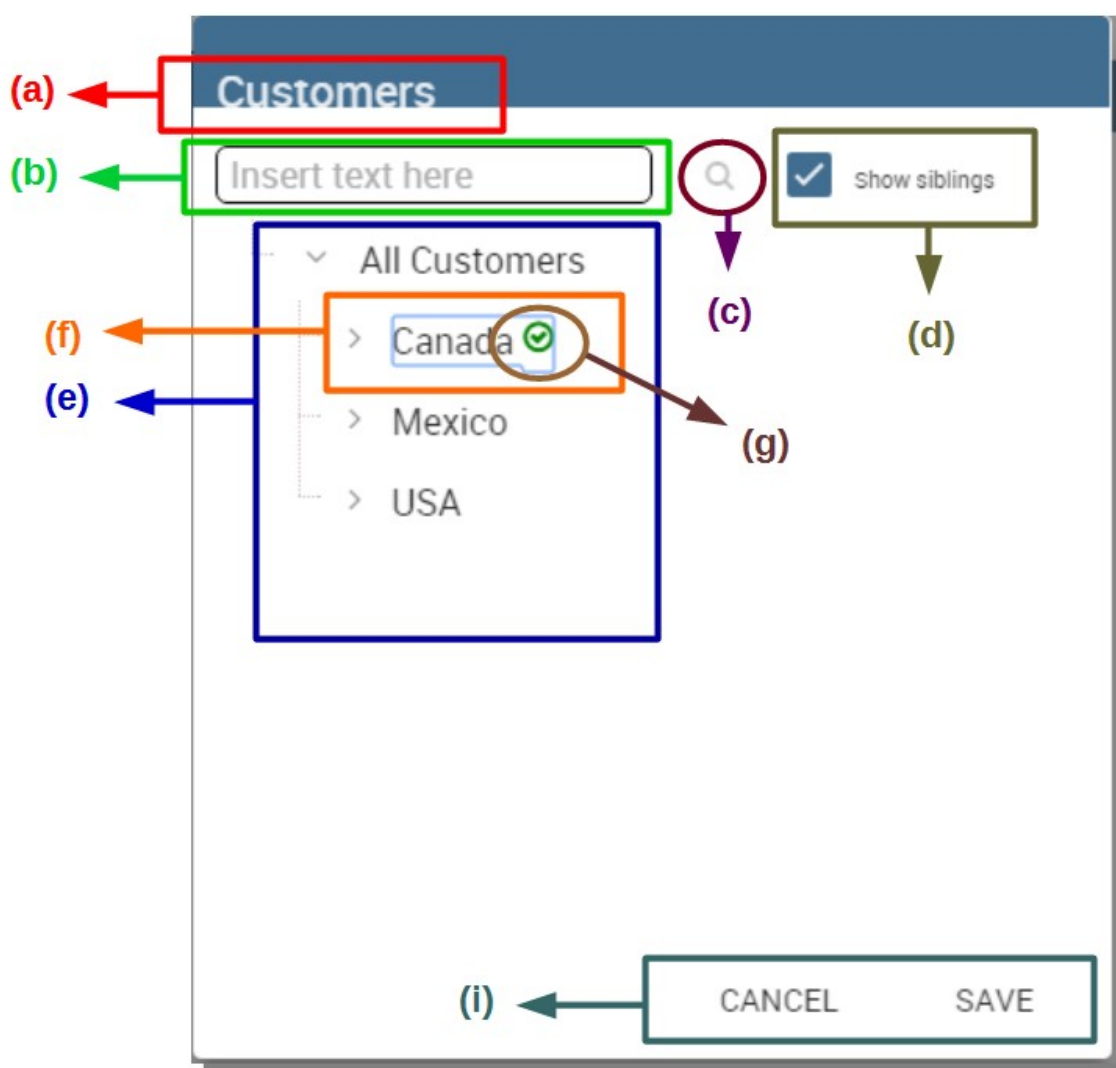


Fig. 5.412: Dialog for slicer choosing.

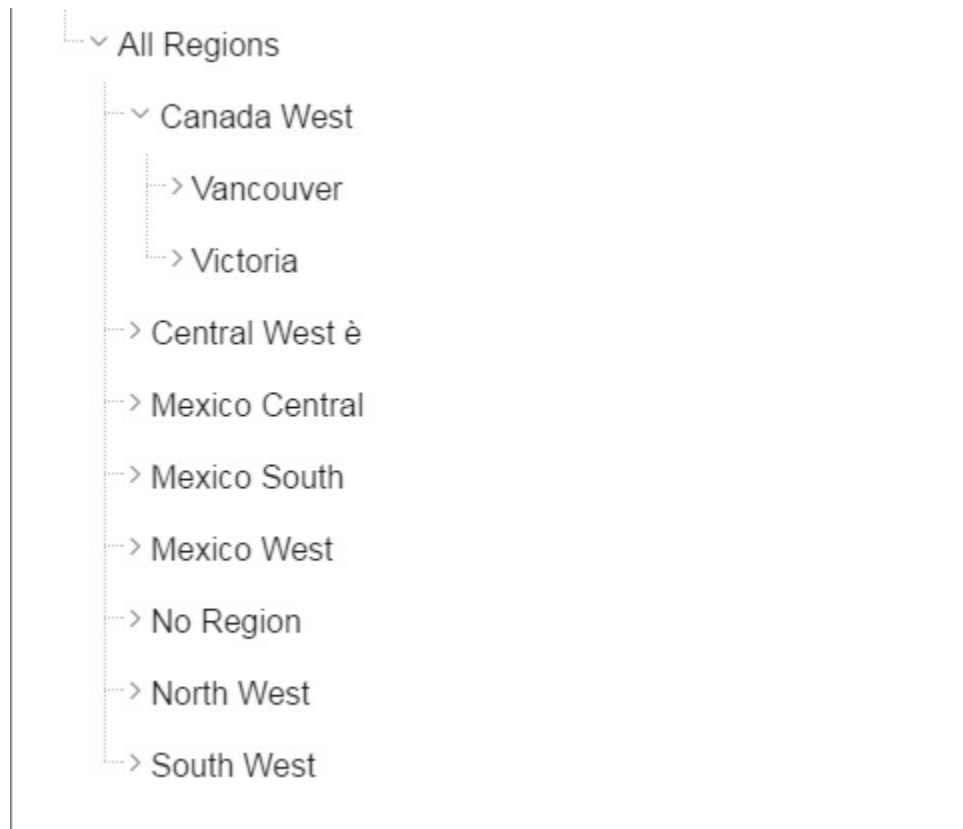


Fig. 5.413: Browsing the member tree.

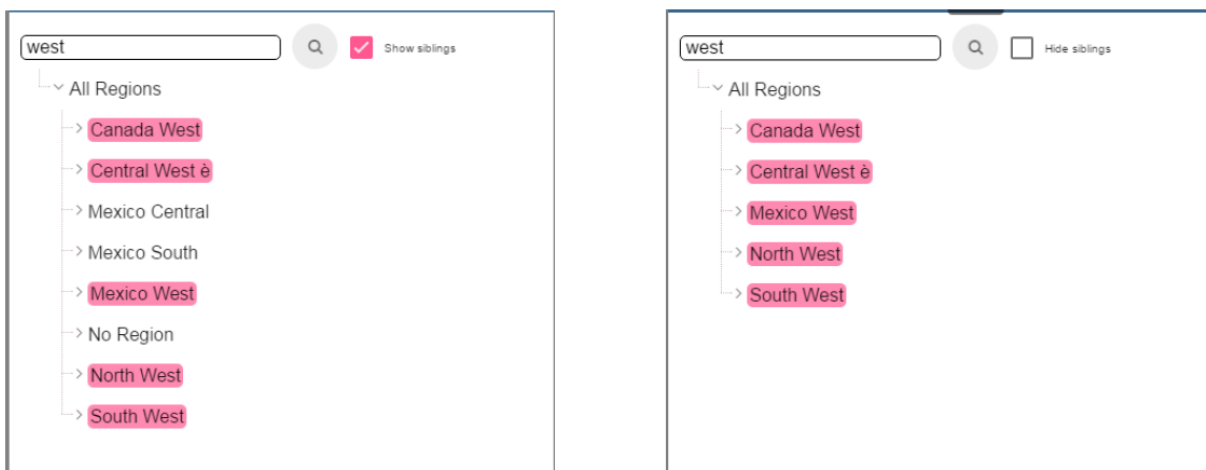


Fig. 5.414: Using the research box.

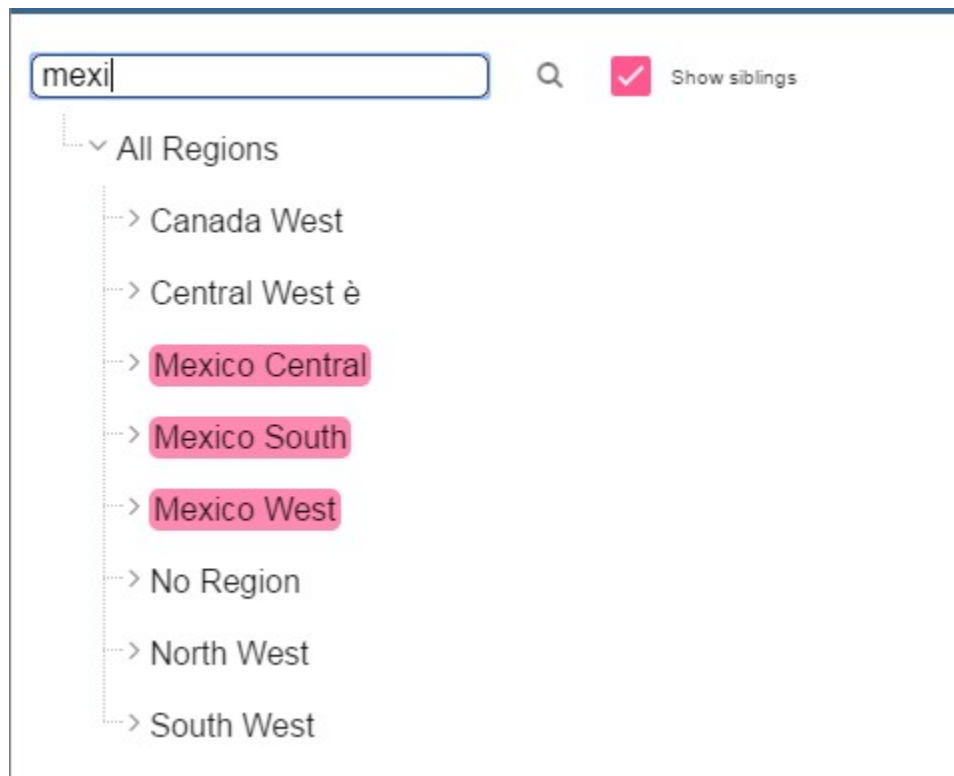


Fig. 5.415: Using the research box after a first investigation.

5.15.2.7 Drill through

To perform drill through operation the user needs first to select a cell, as in the following figure, on which to perform operations. Then clicking on the button for a drill through in the side bar, a dialog will open with results (this pop up could take some time to open).


In particular, referring to the next figure, drill though dialog consists of:

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

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,

Region

 ☒ Show siblings

▼

☒ All Regions

>

☐ Canada West

>

☐ Central West è

>

☐ Mexico Central

>

☐ Mexico South

>

☐ Mexico West

>

☐ No Region

>

☐ North West

>

☐ South West

CANCEL

SAVE

Fig. 5.416: Filter dialog.

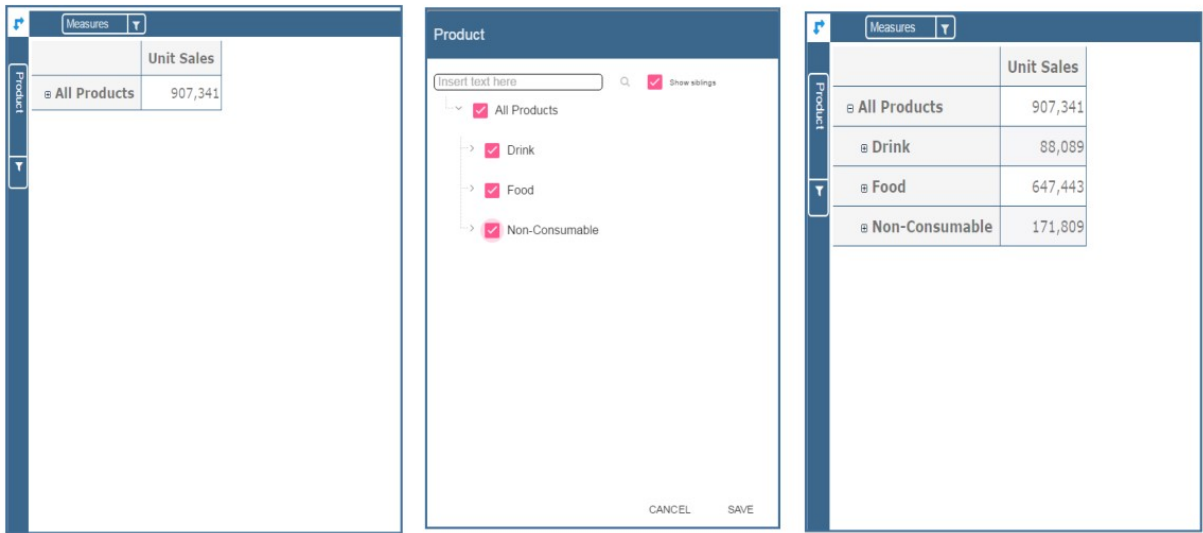


Fig. 5.417: Filter effects on pivot table.

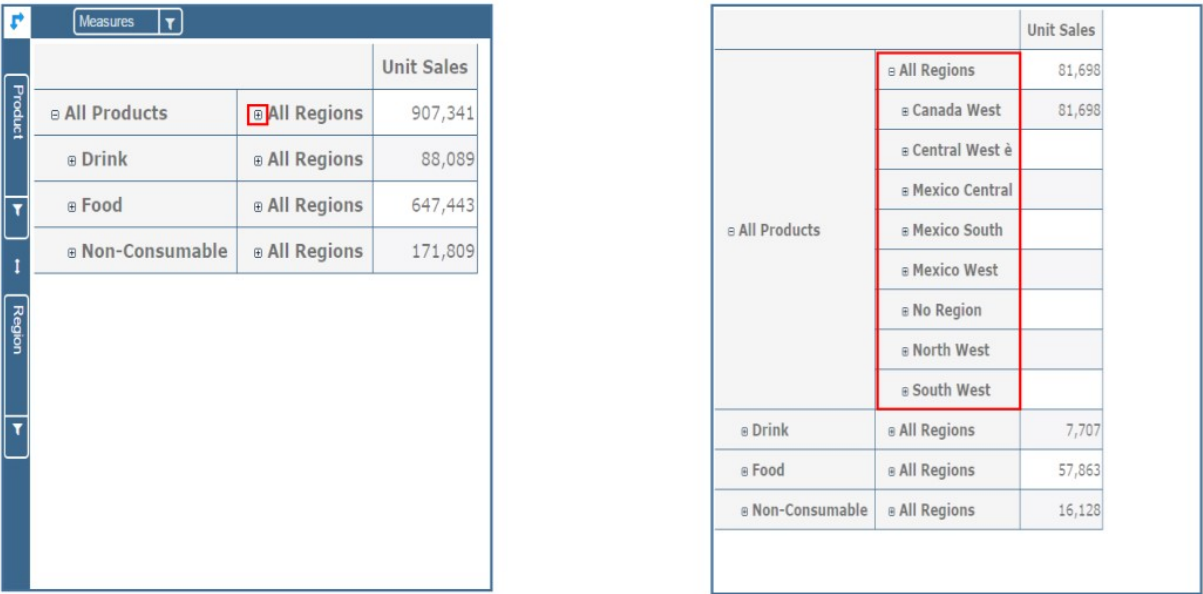


Fig. 5.418: “Position” drill down.

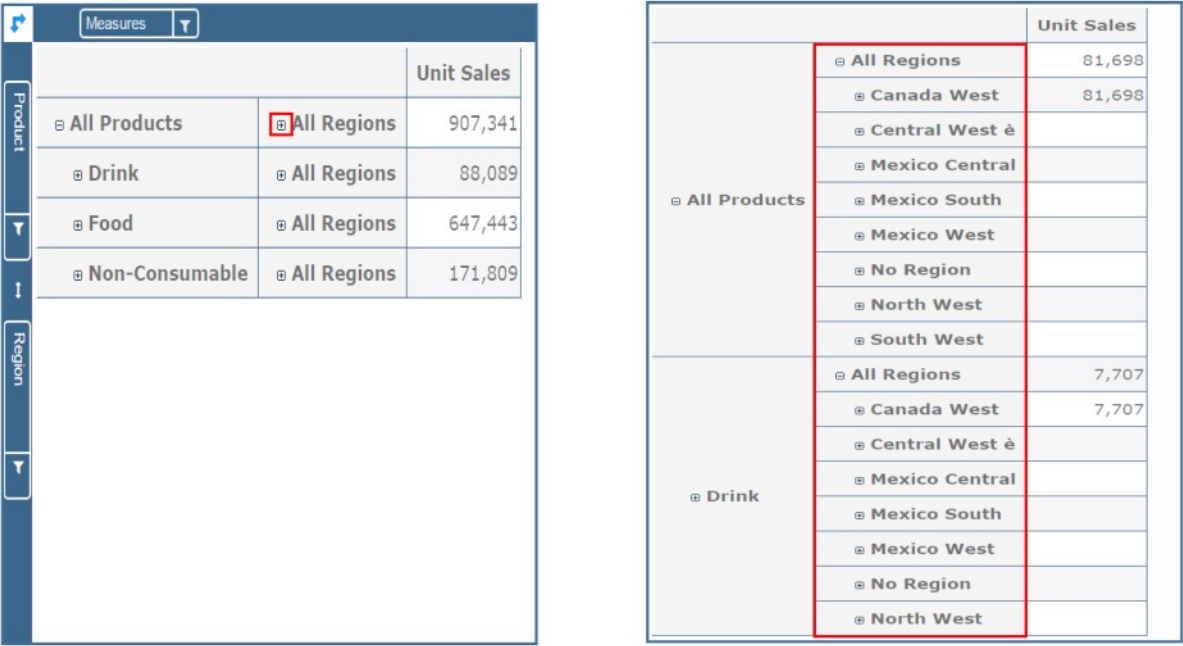


Fig. 5.419: “Member” drill down.

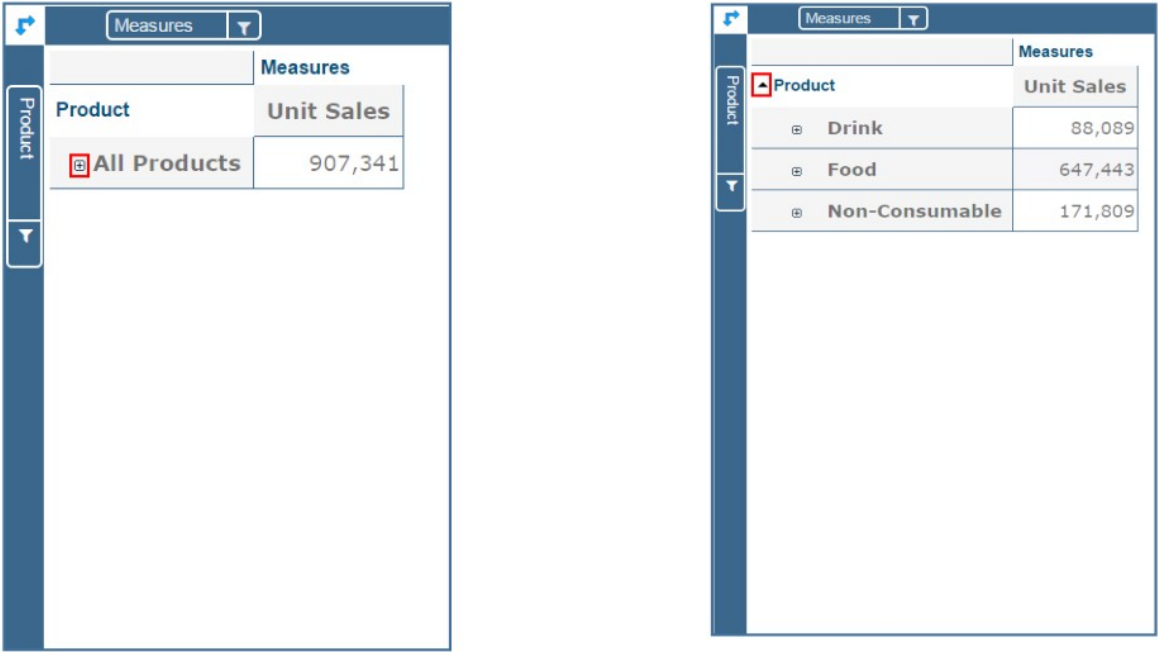
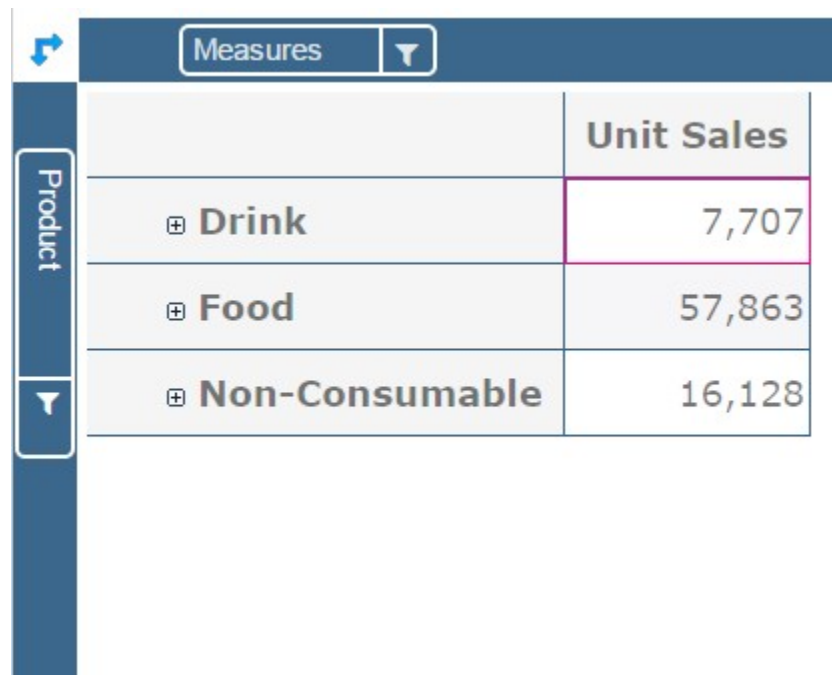


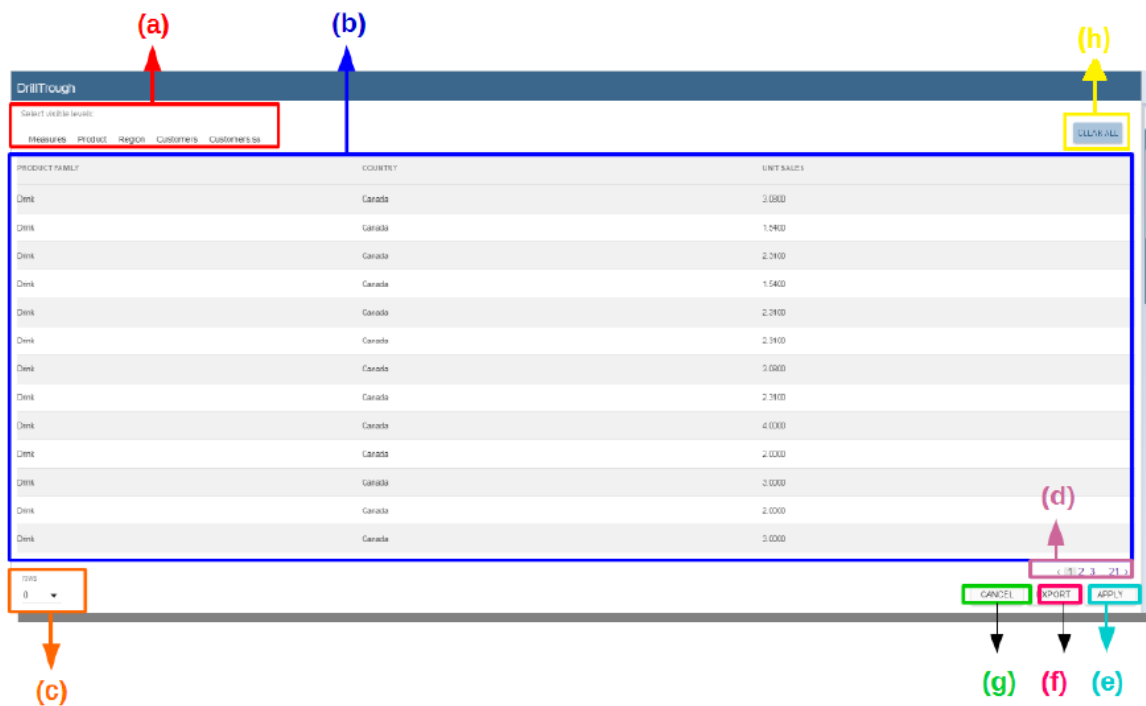
Fig. 5.420: “Replace” drill down.



The screenshot shows a software interface with a 'Measures' dropdown set to 'Unit Sales' and a 'Product' filter. Below, a table lists product families and their corresponding unit sales.

	Unit Sales
+ Drink	7,707
+ Food	57,863
+ Non-Consumable	16,128

Fig. 5.421: Drill thorough option.



The screenshot shows the 'DrillThorough' window with a table of data. Annotations (a) through (h) point to specific UI elements:

- (a) Points to the 'Measures' dropdown menu.
- (b) Points to the 'Product' dropdown menu.
- (c) Points to the 'Data' dropdown menu.
- (d) Points to the 'Export' button.
- (e) Points to the 'Apply' button.
- (f) Points to the 'Cancel' button.
- (g) Points to the 'Clear All' button.
- (h) Points to the 'Clear All' button.

PRODUCT FAMILY	COUNTRY	UNIT SALES
Drink	Canada	2.0800
Drink	Canada	1.0400
Drink	Canada	2.0800
Drink	Canada	1.5400
Drink	Canada	2.0800
Drink	Canada	2.0800
Drink	Canada	3.0800
Drink	Canada	2.0800
Drink	Canada	4.0000
Drink	Canada	2.0000
Drink	Canada	2.0000
Drink	Canada	2.0000
Drink	Canada	3.0000

Fig. 5.422: Drill thorough window.

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 above, (c)). Finally, loaded data can be exported in csv format by clicking on the “Export” button.

5.15.2.8 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.

5.15.2.9 Showing MDX

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

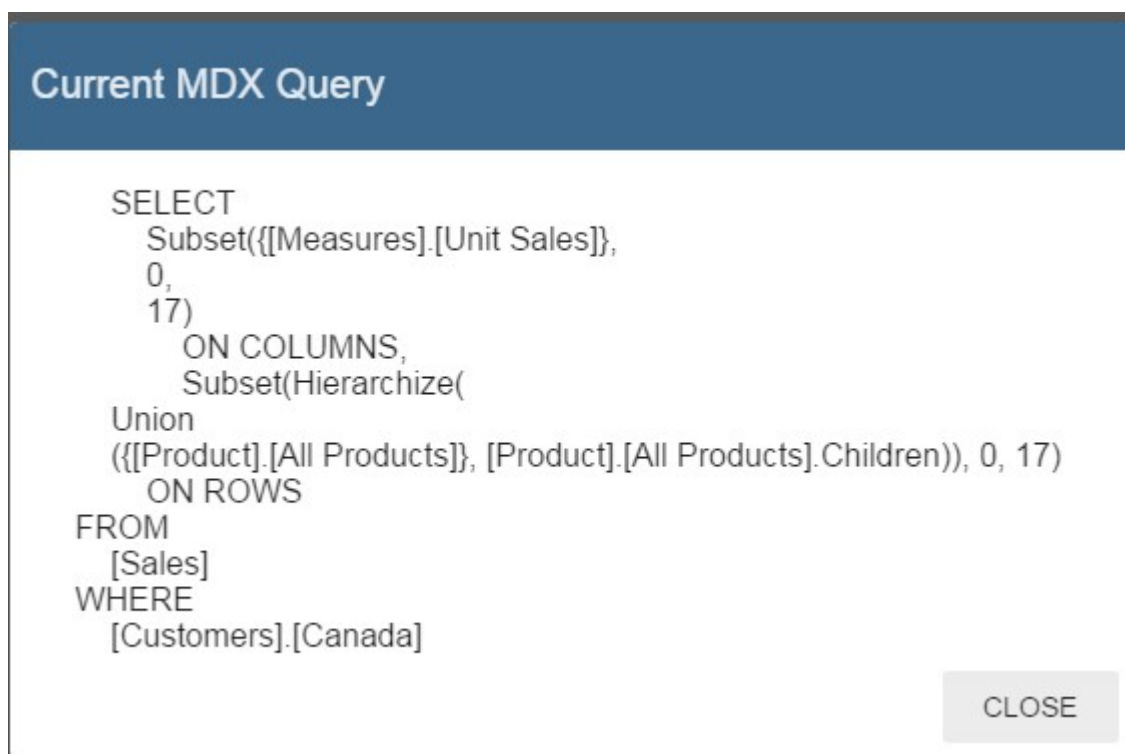


Fig. 5.423: Showing MDX query example.

5.15.2.10 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,
- click on the save button.

Result of the MDX query “should” appear in pivot table as in figure below. In fact, the user is responsible for entering *valid* MDX query.

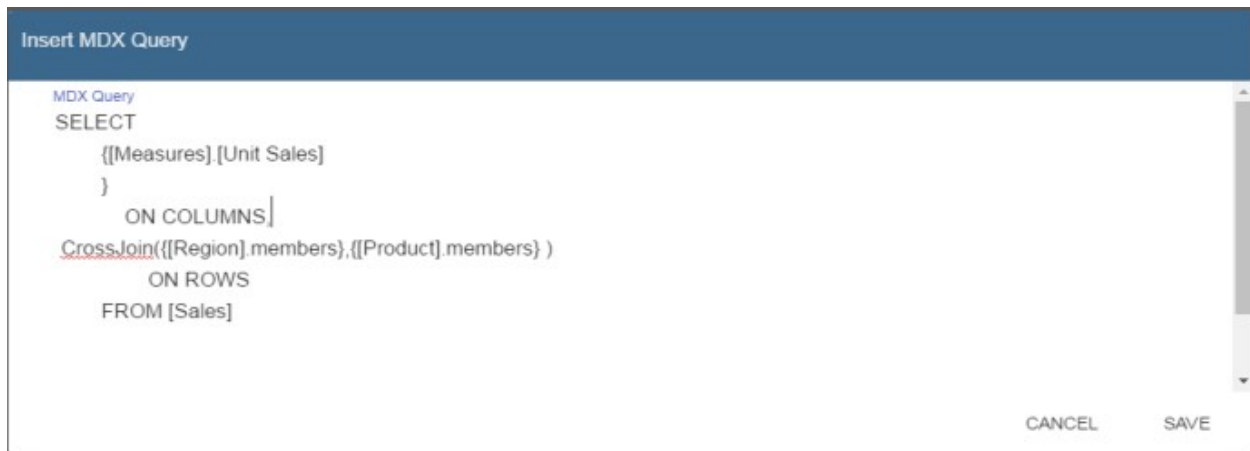


Fig. 5.424: Sending MDX query example.

5.15.2.11 Showing parent members

If a user wants to see additional information 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 the following two figures.

5.15.2.12 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 below.

5.15.2.13 Showing properties

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

1. as part of pivot table 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;
2. in a pop up as compact properties. To enable compact properties user should click on enable compact properties button in the side bar. In this way in all the cells of members Suppressing empty columns/rows which has property set, a table icon appears. This icon lets the property pop up opens. Figure below shows an example.

5.15.2.14 Suppressing empty columns/rows

To hide the empty rows and/or columns, if any, from pivot table the user can click on the "Suppress empty rows/columns" button in the side bar panel. An example is given in Figure below.

5.15.2.15 Sorting

To enable member ordering 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" hierarchy or members that are crossjoined with them, as shown below.

		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

Fig. 5.425: Sending MDX query example.

	Unit Sales								
	☐ All Regions	☐ Canada West	☐ Central West ☐	☐ 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

Fig. 5.426: 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

Fig. 5.427: Pivot table after the parent members selection.

		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

Fig. 5.428: Hide/show spans.

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

Fig. 5.429: Show properties.


	Unit Sales	Properties	
▣ All Regions	907,341	name	value
▣ Canada West	81,698	Store Type	Deluxe Supermarket
▣ Vancouver	64,858	Store Manager	Ruth
▣ Vancouver	64,858	Store Sqft	23112
 Store 19	64,858	Grocery Sqft	16418
▣ Victoria	16,840	Frozen Sqft	4016
▣ Central West è	3,588	Meat Sqft	2678
▣ Mexico Central	250,110	Has coffee bar	true
▣ Mexico South	65,669	Street address	6644 Sudance Drive
▣ Mexico West	44,870		
▣ No Region			
▣ North West	330,781		
▣ South West	130,626		

Fig. 5.430: Show properties summarized in a pop up.

	Unit Sales		Unit Sales
▣ All Regions	907,341	▣ All Regions	907,341
▣ Canada West	81,698	▣ Canada West	81,698
▣ Central West è	3,588	▣ Central West è	3,588
▣ Mexico Central	250,110	▣ Mexico Central	250,110
▣ Mexico South	65,669	▣ Mexico South	65,669
▣ Mexico West	44,870	▣ Mexico West	44,870
▣ No Region		▣ North West	330,781
▣ No District		▣ South West	130,626
▣ None			
▣ North West	330,781		
▣ South West	130,626		

Fig. 5.431: Suppressing empty columns/rows.

	Unit Sales			
	☐ All Products	☐ Drink	☐ Food	☐ Non-Consumable
☐ All Regions	907,341	88,089	647,443	171,809
☐ No Region				
☐ Central West ☐	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

Fig. 5.432: Member sorting.

To sort members the user needs to click on the sorting command **☐ All Products**, available next to each member of the pivot table. Note that the sorting criteria is ascending at first execution. If the user clicks on the sorting icon, criteria will change to descending and the result will be shown in pivot table.

To remove the sorting, 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 the following figure, dialog sorting settings consists of:

- sorting modes (a),
- no sorting (by default) (b),
- basic (c),
- breaking (d),
- count (e),
- a number input field for count mode definition (f),
- a save button (g).

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.

5.15.2.16 Calculated members and sets

Firstly we stress that to enable **Calculated fields** in your Olap document a proper button tag is needed in your Olap template. Such a tag is `<BUTTON_CC visible="true"/>`.

Once enabled, to create a calculated member/set the user should:

1. select a member of the pivot table, as in figure above, which will be the parent of the calculated member,
2. click on the “calculated field” button in the side bar panel: a “Select function” dialog will appear. The latter consists of (refer to next figure):
 - a name input field (a),
 - an aggregation functions tab (b),
 - an arithmetic functions tab (c),

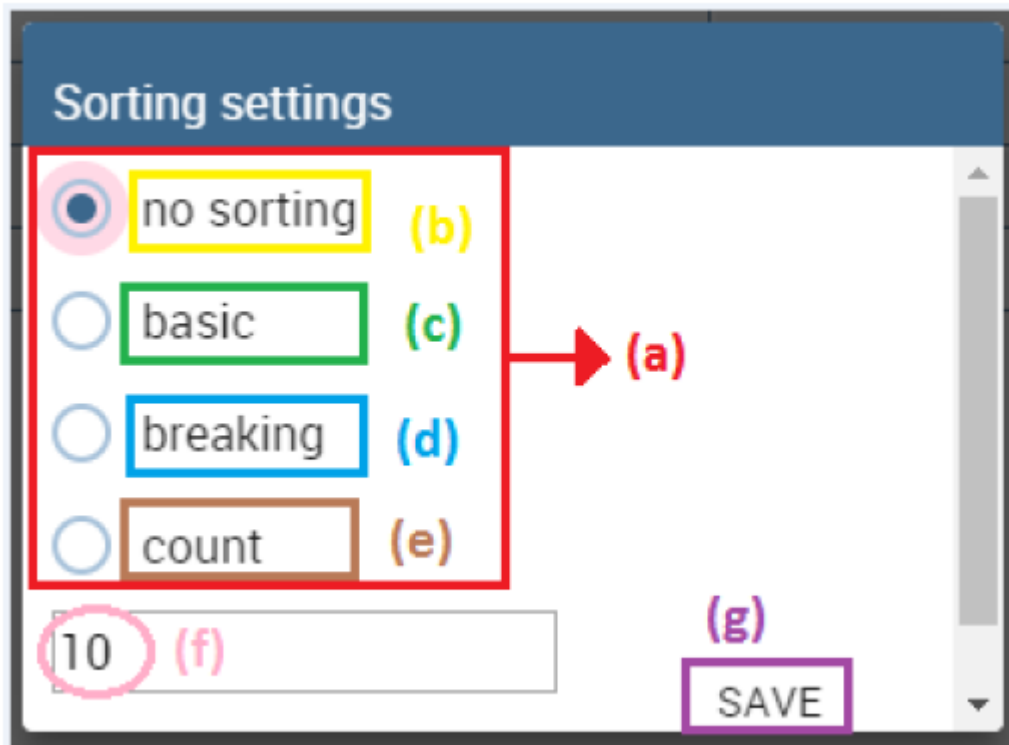


Fig. 5.433: Sorting settings window.

	Unit Sales
⊖ All Products	907,341
⊕ Drink	88,089
⊕ Food	647,443
⊕ Non-Consumable	171,809

Fig. 5.434: Calculated member.

- a temporal functions tab (d),
- a custom functions tab (e),
- a recent functions tab (f),
- an available functions list (g),
- ok and cancel buttons (h).

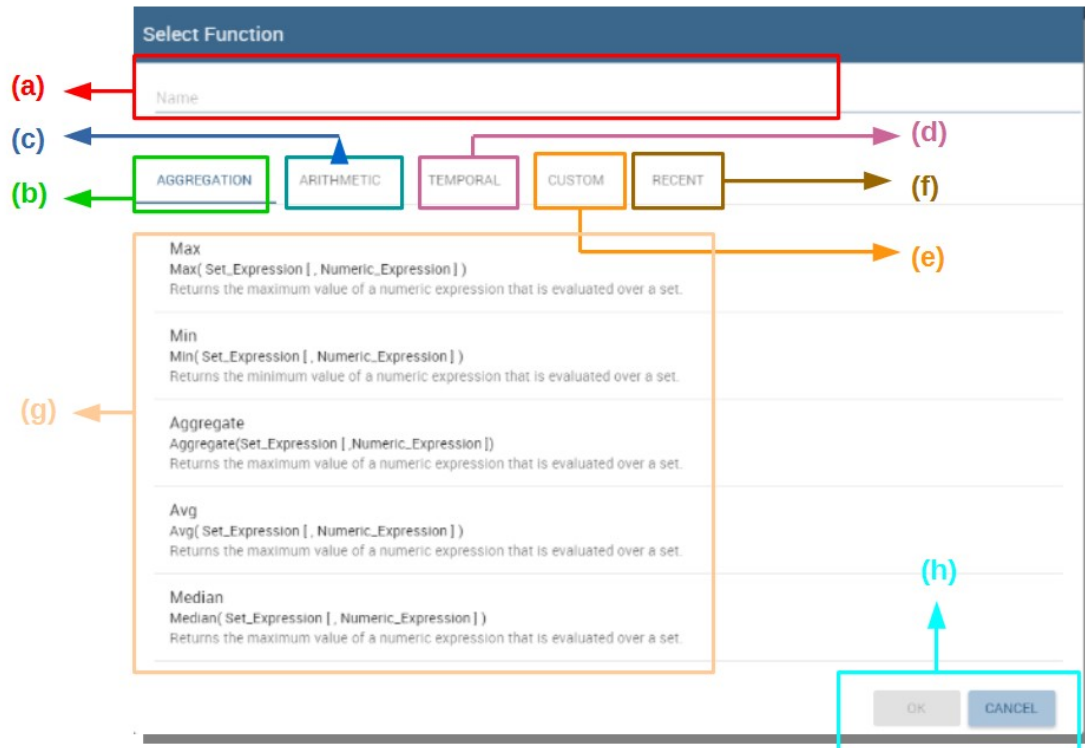


Fig. 5.435: Select function dialog.

The function definition used to create calculated members are read from the formula.xml file, located at: ROOT/resources/yourTennant/Olap folder. Functions are divided by few different tabs. In particular, **Tab Recent** contains calculated members and calculated sets created by user and saved in cookies. If there are no sets/members stored in the cookies, that tab will be empty. **Tab Custom** is where to define custom functions. These functions can be used to make really complex operations that are not part of predefined MDX functions. There you can use combination of few functions together or use operators for complex mathematical calculations. They are also defined in formulas.xml. If a specific tab doesn't contain any formula, it will not be displayed. The "Name" field is mandatory, indeed the creation of a function without a name is forbidden. In **Recent tab**, the "Name" field is hidden. the figure below provides an example of edited formula in the formulas.xml file.

3. Select a function and enter a calculated member/set name and click on "Ok". A dialog for arguments definition will show up, as shown in the following figure. This is made up of the following elements:

- selected function name (a),

```

<formula >
  <name>Max</name>
  <syntax>Max( Set_Expression [ , Numeric_Expression ] )</syntax>
  <body>Max(set,number)</body>
  <argument>
    <name>set</name>
    <expected_value>Set_Expression</expected_value>
    <default_value></default_value>
    <argument_description>A valid Multidimensional Expressions (MDX) expression that returns a set</argument_description>
  </argument>
  <argument>
    <name>number</name>
    <expected_value>Numeric_Expression</expected_value>
    <default_value>[Measures].currentMember</default_value>
    <argument_description>
      A valid numeric expression that is typically a Multidimensional Expressions (MDX) expression of cell coordinates that return a number.
    </argument_description>
  </argument>
  <output>Number</output>
  <description>
    Returns the maximum value of a numeric expression that is evaluated over a set.
  </description>
  <type>aggregation</type>
</formula>

```

Fig. 5.436: Example of one formula inside of formulas.xml.

- function description (b),
- text input fields for argument expression (c),
- expected MDX expression return type (d),
- argument's MDX expression description (e),
- open saved button (f),
- select from table button (g),
- ok and cancel buttons (h).

In particular, to input MDX expression argument, the user has three options, listed in the following.

1. Type it manually (for advance users).
2. Select members from the pivot table: to select a members that are going to be included in a set, the user should (see next figure):
 - click on select from table button,
 - click on members in a pivot table,
 - click ok in dialog to finish selection.

The expression of selected members will be imported in text input fields for argument expression as figure below shows.

3. Import expression from saved calculated members or sets. To import calculated member/set, the user should:
 - Click on open saved button. Then the dialog of saved calculated members/sets will appear (next figure) and it consists of:
 - a list of saved calculated members and sets,
 - a calculated member/set name,
 - calculated member/set return type is shown by round icon.
 - Click on calculated member/set. The expression of saved calculated member/set will be imported in text input fields for argument expression, as highlighted below.
 - After filling all the arguments of function, clicking on OK button will:
 - add calculated member in a pivot table,
 - save calculated set and it will be available for creation of other calculated member and sets.

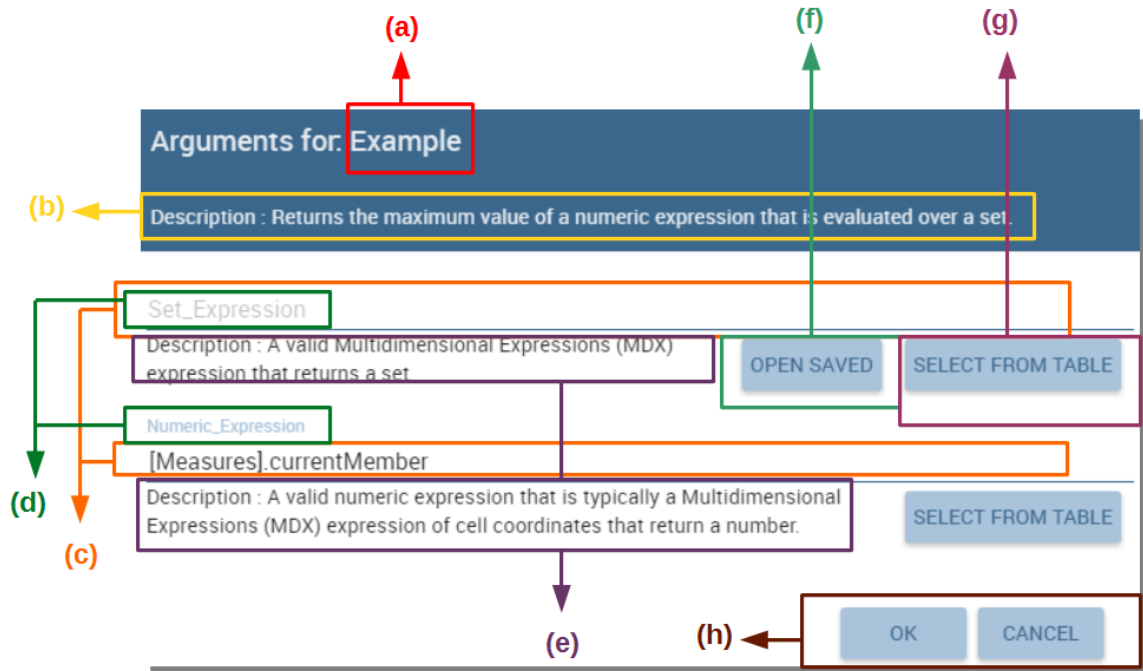


Fig. 5.437: Argument definition dialog.

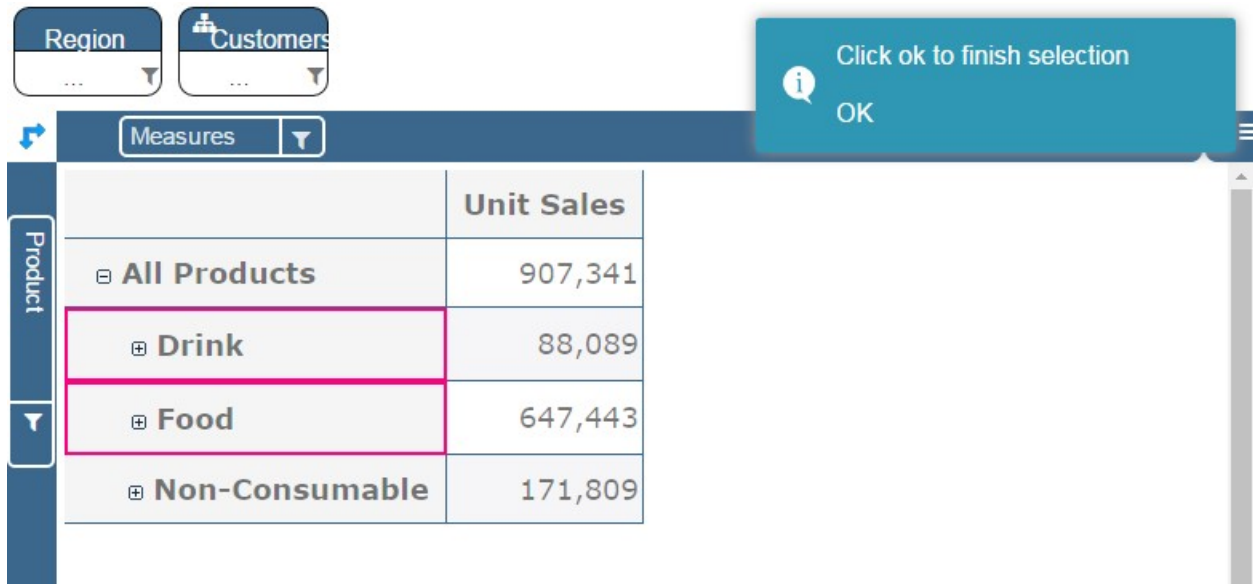


Fig. 5.438: Selecting members.

Arguments for: max

Description : Returns the maximum value of a numeric expression that is evaluated over a set.

Set Expression

[[Product].[Drink],[Product].[Food]]

Description : A valid Multidimensional Expressions (MDX) expression that returns a set

OPEN SAVEDSELECT FROM TABLE

Numeric Expression
Description : A valid numeric expression that is typically a Multidimensional Expressions (MDX) expression of cell coordinates that return a number.

OPEN EDITOROKCANCEL

Fig. 5.439: Expression of the selected members.

Arguments for: max

Description : Returns the maximum value of a numeric expression that is evaluated over a set.

Set Expression

Max(LastPeriods(2,[Time].[1997].[Q4]),Measures.currentNumber)

Description : A valid Multidimensional Expressions (MDX) expression that returns a set

OPEN SAVEDSELECT FROM TABLE

Numeric Expression
Description : A valid numeric expression that is typically a Multidimensional Expressions (MDX) expression of cell coordinates that return a number.

OPEN EDITOROKCANCEL

Fig. 5.440: Saved sets dialog.

Open saved

Mmax

CANCEL

Fig. 5.441: Expression of the saved/calculated member/set.

In tab “Recent”, opening the “Select function” dialog the user can find a list of saved calculated member and sets which can be edited or deleted. Editing is done by clicking on one of them.

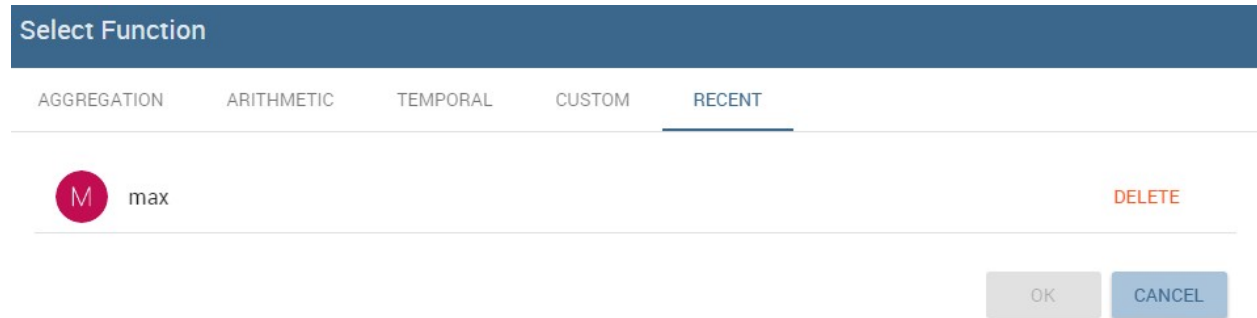


Fig. 5.442: Edit a calculated member.

Deleting is done by Delete button as shown in figure above.

5.15.3 Creation of an OLAP document*

Multidimensional analysis allows the hierarchical inquiry of numerical measures over predefined dimensions. In Cockpit we explained how the user can monitor data on different detail levels and from different perspectives. Here we want to go into details of how a technical user can create an OLAP document. We recall that the main characteristics of OLAP documents are:

- the need for a specific data structure (logical or physical);
- analysis based on dimensions, hierarchies and measures;
- interactive analysis;
- freedom to re-orient analysis;
- different levels of data analysis, through synthetic and detailed views;
- drill-down, slice and dice, drill-through operations.

Considering these items, we will describe the steps to develop an OLAP document.

5.15.3.1 About the engine

Knowage performs OLAP documents by relying on the **OLAP engine**. This engine integrates Mondrian OLAP server and two different cube navigation clients to provide multi-dimensional analysis. In general, Mondrian is a Relational Online Analytical Processing (ROLAP) tool that provides the back-end support for the engine. OLAP structures, such as cubes, dimensions and attributes, are mapped directly onto tables and columns of the data warehouse. This way, Mondrian builds an OLAP cube in cache that can be accessed by client applications. The Knowage OLAP engine provides the front-end tool to interact with Mondrian servers and shows the results via the typical OLAP functionalities, like drill down, slicing and dicing on a multi-dimensional table. Furthermore, it can also interact with XMLA servers. This frontend translates user’s navigation actions into MDX queries on the multi-dimensional cube, and show query results on the table he is navigating.

5.15.3.2 Development of an OLAP document

The creation of an OLAP analytical document requires the following steps:

- schema modelling;

- catalogue configuration;
- OLAP cube template building;
- analytical document creation.

5.15.3.2.1 Schema modelling

The very first step for a multi-dimensional analysis is to identify essential information describing the process/event under analysis and to consider how it is stored and organized in the database. On the basis of these two elements, a mapping process should be performed to create the multi-dimensional model.

Hint: From the relational to the multi-dimensional model

The logical structure of the database has an impact on the mapping approach to be adopted when creating the multidimensional model, as well as on query performances.

If the structure of the relational schema complies with multi-dimensional logics, it will be easier to map the entities of the physical model onto the metadata used in Mondrian schemas. Otherwise, if the structure is highly normalized and scarcely dimensional, the mapping process will probably require to force and approximate the model to obtain a multi-dimensional model. As said above, 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 XMLbased files, called Mondrian schemas. Mondrian schemas are treated by Knowage as resources and organized into catalogues. Hereafter, an example of Mondrian schema in Mondrian schema example:

Listing 5.33: Mondrian schema example

```

1  <?xml version="1.0"?>
2  <Schema name="FoodMart">
3      <!-- Shared dimensions -->
4      <Dimension name="Customers">
5          <Hierarchy hasAll="true" allMemberName="All Customers"
6              primaryKey="customer_id">
7              <Table name="customer"/>
8              <Level name="Country" column="country" uniqueMembers="true"/>
9              <Level name="State Province" column="state_province"
10                  uniqueMembers="true"/>
11
12              <Level name="City" column="city" uniqueMembers="false"/>
13
14          </Hierarchy> ...
15      </Dimension> ...
16
17      <!-- Cubes -->
18      <Cube name="Sales">
19
20          <Table name="sales_fact_1998"/>
21
22          <DimensionUsage name="Customers" source="Customers"
23              foreignKey="customer_id" /> ...
24
25          <!-- Private dimensions -->
26
27          <Dimension name="Promotion Media" foreignKey="promotion_id">
28
29              <Hierarchy hasAll="true" allMemberName="All Media"
30                  primaryKey="promotion_id">
31                  <Table name="promotion"/>
32                  <Level name="Media Type" column="media_type" uniqueMembers="true"/>
33

```

(continues on next page)

(continued from previous page)

```

34         </Hierarchy>
35
36     </Dimension> ...
37
38     <!-- basic measures-->
39
40     <Measure name="Unit Sales" column="unit_sales" aggregator="sum"
41         formatString="#,###.00"/>
42
43     <Measure name="Store Cost" column="store_cost" aggregator="sum"
44         formatString="#,###.00"/>
45
46     <Measure name="Store Sales" column="store_sales" aggregator="sum"
47         formatString="#,###.00"/>
48     ...
49
50     <!-- derived measures-->
51
52     <CalculatedMember name="Profit" dimension="Measures">
53         <Formula>
54             [Measures].[Store Sales] - [Measures].[Store Cost]
55         </Formula>
56
57         <CalculatedMemberProperty name="format_string" value="$#,##0.00"/>
58     </CalculatedMember>
59
60 </Cube>
61 ...
62 </Schema>

```

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.

Note: Mondrian

For a detailed explanation of Mondrian schemas, please refer to the documentation available at the official project webpage: <http://mondrian.pentaho.com/>.

5.15.3.2.1 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 your 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.

5.15.3.2.2 OLAP template building

Once the cube has been created, you need to build a template which maps the cube to the analytical document. To accomplish this goal the user must manually edit the template. The template is an XML file telling Knowage OLAP engine how to navigate the OLAP cube and has a structure like the one represented in next code:

Listing 5.34: Mapping template example

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <olap>
3      <!-- schema configuration -->
4      <cube reference="FoodMart"/>
5
6      <!-- query configuration -->
7      <MDXquery>
8          SELECT {[Measures].[Unit Sales]} ON COLUMNS
9              , {[Region].[All Regions]} ON ROWS
10             FROM [Sales]
11             WHERE [Product].[All Products].[${family}]
12             <parameter name="family" as="family"/>
13      </MDXquery>
14
15      <MDXMondrianQuery>
16          SELECT {[Measures].[Unit Sales]} ON COLUMNS
17              , {[Region].[All Regions]} ON ROWS
18             FROM [Sales]
19             WHERE [Product].[All Products].[Drink]
20      </MDXMondrianQuery>
21
22      <!-- toolbar configuration -->
23      <TOOLBAR>
24          <BUTTON_MDX visible="true" menu="false" />
25          <BUTTON_FATHER_MEMBERS visible="true" menu="false"/>
26          <BUTTON_HIDE_SPANS visible="true" menu="false"/>
27          <BUTTON_SHOW_PROPERTIES visible="true" menu="false"/>
28          <BUTTON_HIDE_EMPTY visible="true" menu="false" />
29          <BUTTON_FLUSH_CACHE visible="true" menu="false" />
30          <BUTTON_SAVE visible="true" menu="false" />
31          <BUTTON_SAVE_NEW visible="true" menu="false" />
32          <BUTTON_EXPORT_OUTPUT visible="true" menu="false" />
33      </TOOLBAR>
34
35  </olap>

```

An explanation of different sections of Mapping template example follows.

- The CUBE section sets the Mondrian schema. It should reference the exact name of the schema, as registered in the catalogue on the Server.
- The MDXMondrianQuery section contains the original MDX query defining the starting view (columns and rows) of the OLAP document.
- The MDX section contains a variation of the original MDX query, as used by the Knowage Engine. This version includes parameters (if any). The name of the parameter will allow Knowage to link the analytical driver associated to the document via the parameter (on the Server).
- The TOOLBAR section is used to configure visibility options for the toolbar in the OLAP document. The exact meaning and functionalities of each toolbar button are explained in next sections. A more complete list of the available options is shown in Menu configurable options:

Listing 5.35: Menu configurable options

```

1  <BUTTON_DRILL_THROUGH visible="true"/>
2  <BUTTON_MDX visible="true"/>
3  <BUTTON_EDIT_MDX visible="true"/>
4  <BUTTON_FATHER_MEMBERS visible="true"/>
5  <BUTTON_CC visible="true"/>
6  <BUTTON_HIDE_SPANS visible="true"/>
7  <BUTTON_SORTING_SETTINGS visible="true"/>
8  <BUTTON_SORTING visible="true" />

```

(continues on next page)

(continued from previous page)

```

9 <BUTTON_SHOW_PROPERTIES visible="true"/>
10 <BUTTON_HIDE_EMPTY visible="true"/>
11 <BUTTON_FLUSH_CACHE visible="true"/>
12 <BUTTON_SAVE visible="true"/>
13 <BUTTON_SAVE_NEW visible="true"/>
14 <BUTTON_UNDO visible="true"/>
15 <BUTTON_VERSION_MANAGER visible="true"/>
16 <BUTTON_EXPORT_OUTPUT visible="false"/>

```

5.15.3.2.3 Creating the analytical document

Once you have the template ready you can create the OLAP document on Knowage Server.

To create a new OLAP document, click on the “create a new document” button 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**.

Type a name, a functionality, load the XML template and save. You will see the document in the functionality (folder) you selected, displayed with the typical cube icon as shown below.

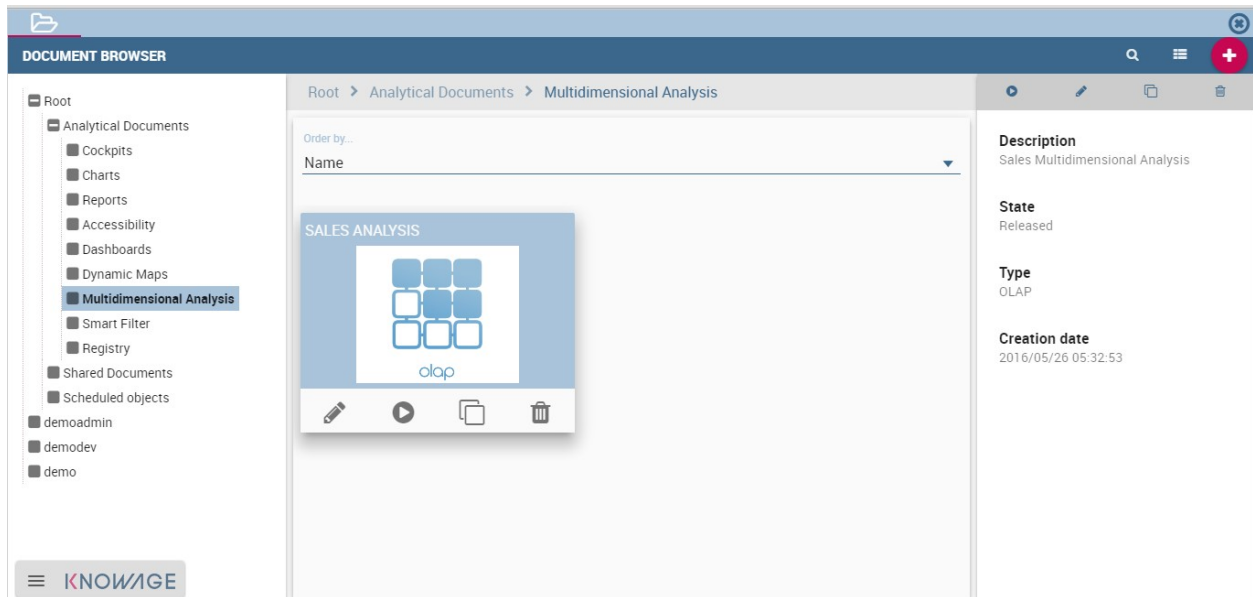


Fig. 5.443: OLAP document on server.

5.15.3.3 OLAP Designer*

Knowage Server is also endowed of an efficient OLAP designer which avoid the user to edit manually the XML-based template that we discussed on in Development of an OLAP document. 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. It is mandatory that the chosen Mondrian schema has no parameters applied.

Warning: Mondrian schema for OLAP designer

The Mondrian schema must not be filtered thorough any parameter or profile attribute.

The page as the one in figure below will open.

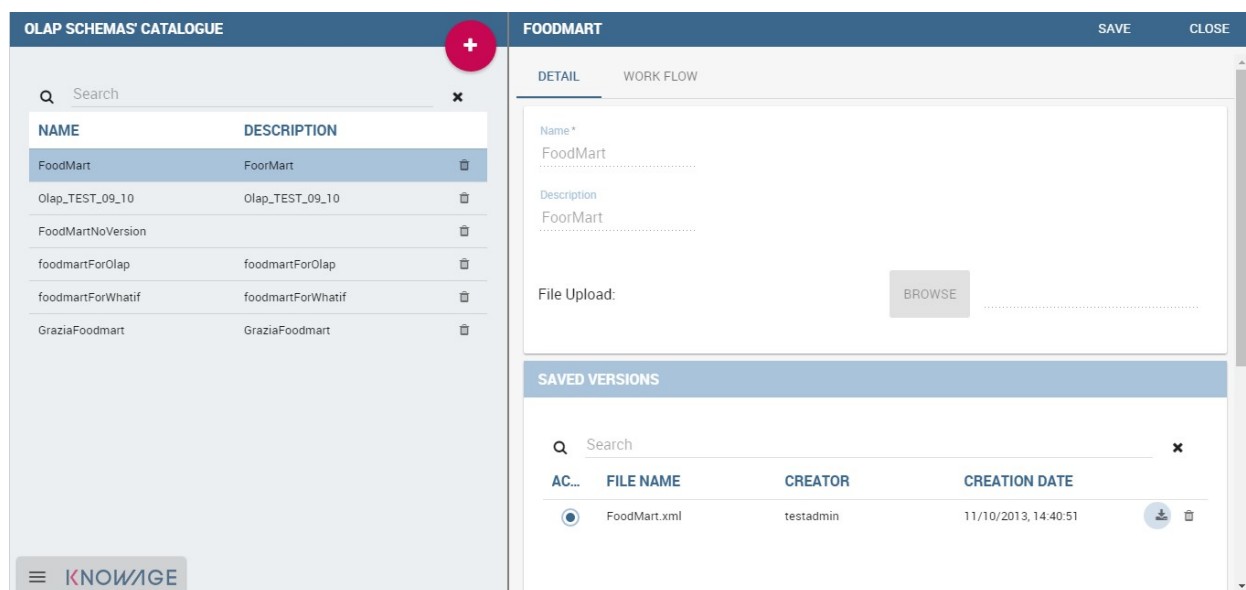



Fig. 5.444: Schema Mondrian from catalog.





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. Next figure sums up these three steps. Following the example just given below you will enter a page like that of the second figure below.

Once entered the page the user can freely set the fields as filter panels or as filter cards, according to requirements. Refer to *Functionalities* Chapter to review the terminology. Make your selection and you can already save the template as shown below.

You can notice that the side panel contains some features (see next figure):

-  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 *Functionalities* 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

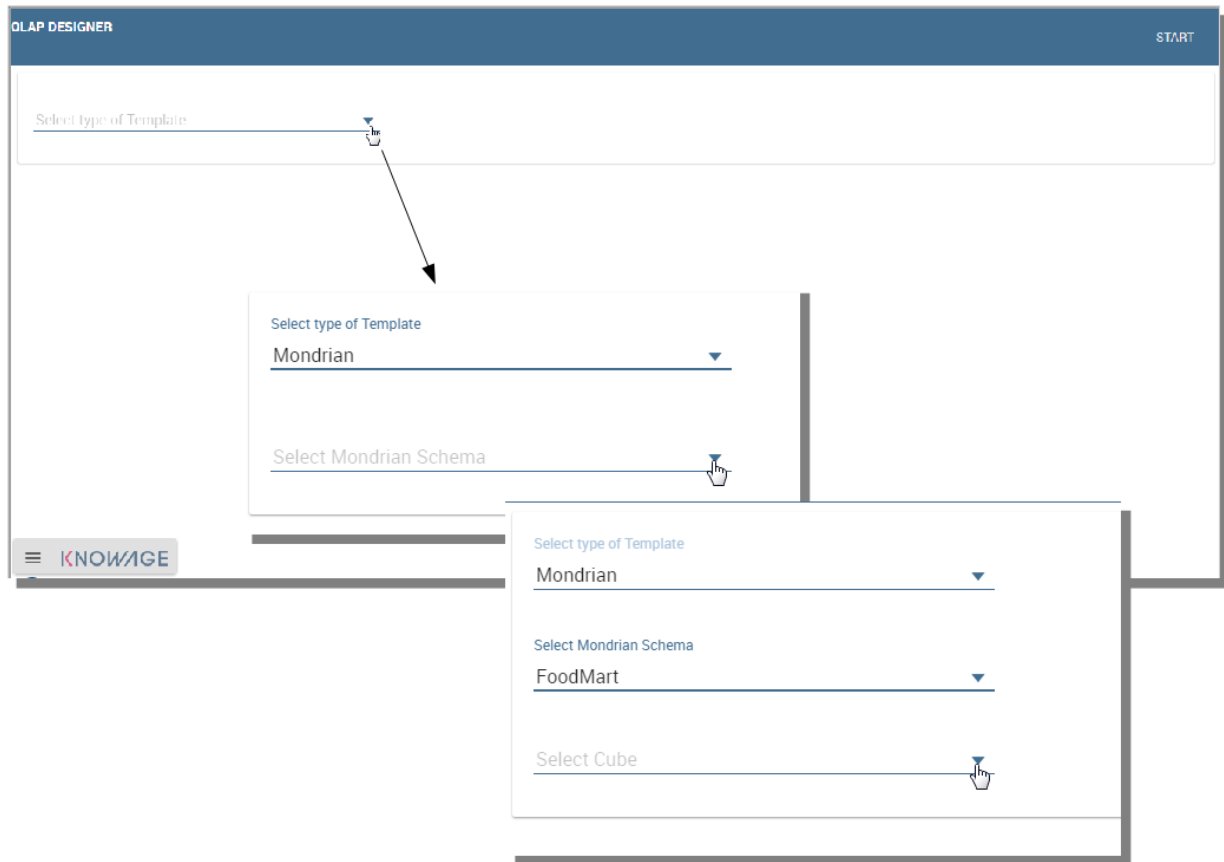


Fig. 5.445: OLAP core configuration.

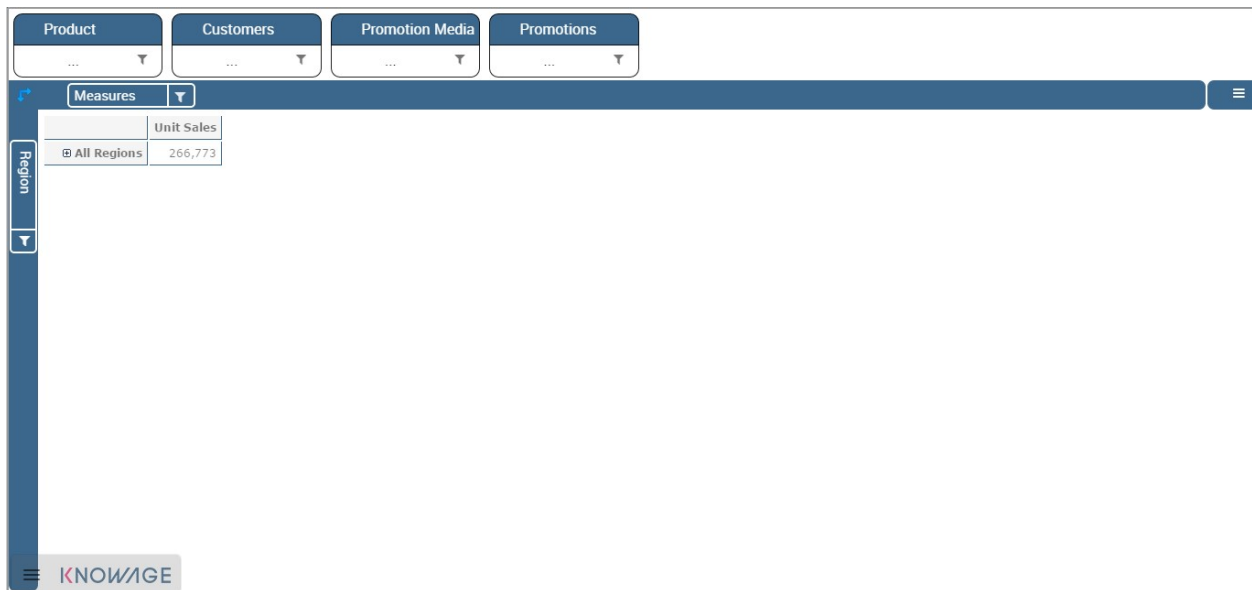


Fig. 5.446: Defining OLAP template.

Region	Product	Promotions	Measures	Unit Sales	Store Cost	Store Sales	Customer Count
All Media	All Customers		Unit Sales	266,773	225,627.23	565,238.13	5,582
Bulk Mail	All Customers		Unit Sales	4,320	3,740.95	9,349.07	333
Cash Register Handout	All Customers		Unit Sales	6,697	5,715.67	14,321.33	482
Daily Paper	All Customers		Unit Sales	7,738	6,559.23	16,479.81	528
Daily Paper, Radio	All Customers		Unit Sales	6,891	5,668.77	14,169.42	499
Daily Paper, Radio, TV	All Customers		Unit Sales	9,513	8,055.22	20,173.97	687
In-Store Coupon	All Customers		Unit Sales	3,798	3,263.11	8,162.46	290
No Media	All Customers		Unit Sales	195,448	165,214.85	414,026.92	5,044
Product Attachment	All Customers		Unit Sales	7,544	6,306.24	15,898.25	532
Radio	All Customers		Unit Sales	2,454	2,087.51	5,213.61	186
Street Handout	All Customers		Unit Sales	5,753	4,856.54	12,192.90	381
Sunday Paper	All Customers		Unit Sales	4,339	3,673.86	9,092.89	307
Sunday Paper, Radio	All Customers		Unit Sales	5,945	5,027.31	12,551.96	422
Sunday Paper, Radio, TV	All Customers		Unit Sales	2,726	2,341.58	5,819.33	196
TV	All Customers		Unit Sales	3,607	3,116.40	7,786.21	274

Fig. 5.447: Defining OLAP template.

lets the end-user the permission to edit and modify. Referring to the following figure, 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.

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 user must access the "Cross Navigation Definition" functionalities available in Knowage Server. Here, referring to *Cross Navigation* section of *Analytical document* chapter, you will use the parameter just set as output parameter.

As shown in figure below, 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 Fig. 5.448 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 available at all. For the other two options the instructions are right the same as the What-if engine.

5.15.3.3.1 Profiled access

As for any other analytical document, Knowage provides filtered access to data via its behavioural model. The behavioural model is a very important concept in Knowage. For a full understanding of its meaning and functionalities, please refer to Behavioural Model.

Knowage offers the possibility to regulate data visibility based on user profiles. Data visibility can be profiled at the level of the OLAP cube, namely the cube itself is filtered and all queries over that cube share the same data visibility criteria.



Fig. 5.448: Side panel features for the OLAP Designer.



Fig. 5.449: Wizard to configure the scenario.

To set the filter, which is based on the attribute (or attributes) in the user's profile, the technical user has to type the Mondrian schema. We report Cube level profilation example as a reference guide. Note that data profiling is performed on the cube directly since the filter acts on the data retrieval logics of the Mondrian Server. So the user can only see the data that have been got back by the server according to the filter.

Listing 5.36: Cube level profilation example.

```

1  <?xml version="1.0"?>
2  <Schema name="FoodMartProfiled">
3  ...
4  <Cube name="Sales_profiled"> <Table name="sales_fact_1998"/>
5  ...
6  <!-- profiled dimension -->
7  <Dimension name="Product" foreignKey="product_id">
8    <Hierarchy hasAll="true" allMemberName="All Products" primaryKey="product_id">
9      <View alias="Product">
10        <SQL dialect="generic">
11          SELECT pc.product_family as product_family, p.product_id as
12            product_id,
13            p.product_name as product_name,
14            p.brand_name as brand_name, pc.product_subcategory as
15            product_subcategory, pc.product_category as product_category,
16            pc.product_department as product_department
17          FROM product as p
18          JOIN product_class as pc ON p.product_class_id = pc.
19            product_class_id
20          WHERE and pc.product_family = '${family}'
21        </SQL>
22      </View>
23
24      <Level name="Product Family" column="product_family"
25        uniqueMembers="false" />
26      <Level name="Product Department" column="product_department"
27        uniqueMembers="false"/>
28      <Level name="Product Category" column="product_category"
29        uniqueMembers=" false"/>
30      <Level name="Product Subcategory" column="product_subcategory"
31        uniqueMembers="false"/>
32      <Level name="Brand Name" column="brand_name"
33        uniqueMembers="false"/>
34      <Level name="Product Name" column="product_name"
35        uniqueMembers="true"/>
36    </Hierarchy>
37  </Dimension>
38 </Cube>
39 ...
40 </Schema>

```

In the above example, the filter is implemented within the SQL query that defines the dimension using the usual syntax

Region **Customers** **Promotions**

Measures

		Unit Sales	Store Cost	Store Sales	Sales Count	Customer Count
⊞ All Media	⊞ All Products	266,773	225,627.23	565,238.13	86,849	5,582
Bulk Mail	⊞ All Products	4,320	3,740.95	9,349.07	1,404	333
Cash Register Handout	⊞ All Products	6,607	5,715.67	14,321.33	2,180	482
Daily Paper	⊞ All Products	7,738	6,558.23	16,479.81	2,488	528
Daily Paper, Radio	⊞ All Products	6,891	5,668.77	14,169.42	2,238	499
Daily Paper, Radio, TV	⊞ All Products	9,513	8,055.22	20,173.97	3,070	687
In-Store Coupon	⊞ All Products	3,708	3,263.11	8,162.46	1,292	290
No Media	⊞ All Products	195,448	165,214.85	414,026.92	63,668	5,044
Product Attachment	⊞ All Products	7,544	6,306.24	15,898.25	2,439	532
Radio	⊞ All Products	2,454	2,087.31	5,213.61	798	186
Street Handout	⊞ All Products	5,753	4,856.54	12,192.90	1,821	381
Sunday Paper	⊞ All Products	4,339	3,673.86	9,092.80	1,411	307
Sunday Paper, Radio	⊞ All Products	5,945	5,027.31	12,551.96	1,935	422
Sunday Paper, Radio, TV	⊞ All Products	2,726	2,341.58	5,819.33	934	196
TV	⊞ All Products	3,607	3,116.40	7,786.21	1,171	274

Cross Navigation Definition

Select the type of cross navigation below

Select type of Cross Navigation

From Member

NEXT CANCEL

Click ok to finish selection

OK

Cross Navigation Definition

Member Name

[Promotion Media].[Media Type]

SELECT FROM TABLE

Parameter Name

media_type

FINISH CANCEL

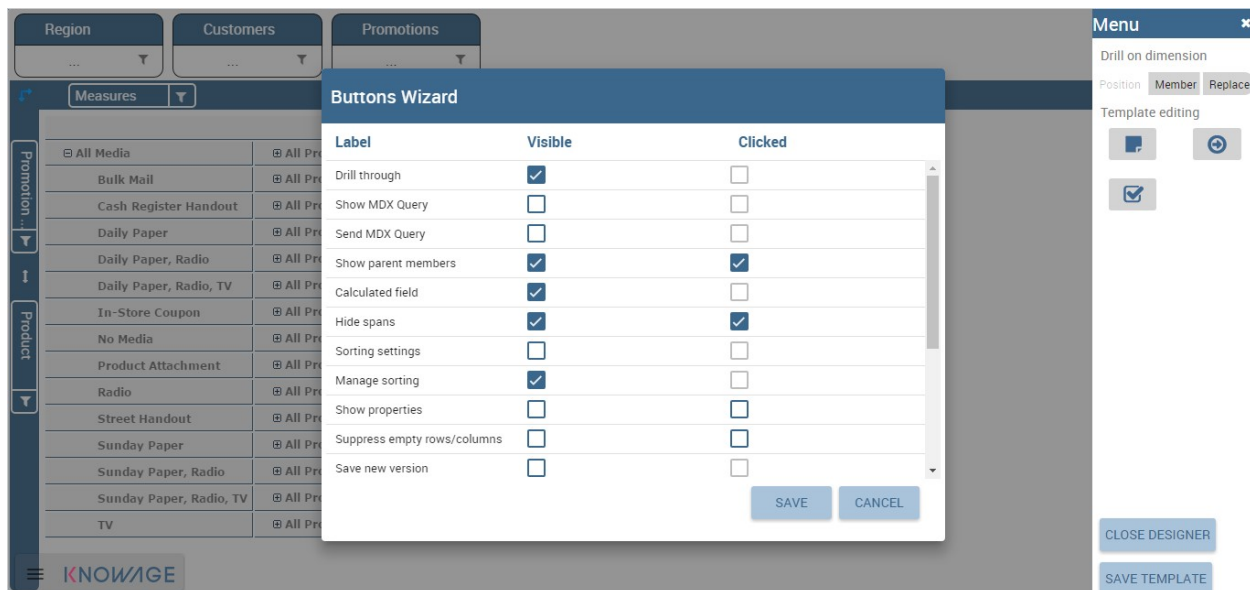


Fig. 5.451: Wizard to configure the scenario.

“pr.product_family = ‘\${family}’”.

The value of the “family” user profile attribute will replace the \${family} placeholder in the dimension definition.

You can filter more than one dimensions/cubes and use more profile attributes. The engine substitutes into the query the exact value of the attribute; in case of a multi value attribute to insert in an SQL-IN clause you will have to give the attribute a value like 'value1', 'value2', and insert into the query a condition like “and pc.product_family IN (\${family})”.

Once the OLAP document has been created using the template designer the user can insert parameters to profile the document. To set parameters the user has to download the Mondrian schema and edit it; modify the dimension(s) (that will update according to the value parameter(s)) inserting an SQL query which presents the parametric filtering clause.

Hint: Filter through the interface

Note that for the OLAP instance, it has not proper sense to talk about “general” parameters. In this case we only deal with profile attributes while all the filtering issue is performed through the interface, using the filter panel.

5.15.3.4 Cross Navigation

The cross navigation must be implemented at template level but also at analytical document level. The latter has been already wildy described in Cross Navigation . In the following we will see the first case. Observe that both procedures are mandatory.

For OLAP documents it is possible to enable the cross navigation on members or on cells and we will give more details on these two cases in the following.

Generally speaking, the user must modify the template file to configure the cross navigation in order to declare the output parameters of the document. We remember that the output parameters definition is discussed in *Cross Navigation* section of *Analytical document* chapter of this manual.

5.15.3.4.1 Cross navigation on members

To activate the cross navigation on a member means that the user can click on a member of a dimension to be sent and visualize a target document. The first type of navigation can be set by editing the OLAP query template. In the first case you need to add a section called “clickable” inside the MDX query tag. In fact,

- the attribute value is equal to the hierarchy level containing the member(s) that shall be clickable;
- the element represents the parameter that will be passed to the destination document. The name attribute is the URI of the parameter that will be passed to the target document. The value 0 represents the currently selected member, as a convention: this value will be assigned to the parameter whose URI is null.

Figure below gives an example. Note that you can recognize that the cross navigation is activated when elements are shown blue highlighted and underlined.

		Unit Sales
Region	⊖ All Products	1.352.144
	⊕ <u>Drink</u>	124.372
	⊕ <u>Food</u>	969.623
	⊕ <u>Non-Consumable</u>	257.148

Fig. 5.452: Cross navigation on member.

If you open the template file you will read instructions similar to the ones reported in Syntax used to set cross navigation.

Listing 5.37: Syntax used to set cross navigation.

```

1 <MDXquery>
2   select {[Measures].[Unit Sales]} ON COLUMNS,
3   {[([Region].[All Regions], [Product].[All Products])}] ON ROWS from
4   [Sales_V]
5   <clickable uniqueName="[Product].[Product Family]" >
6     <clickParameter name="family" value="{0}"/>
7   </clickable>
8 </MDXquery>

```

5.15.3.4.2 Cross navigation from a cell of the pivot table

This case is similar to the one-dimension drill except that in this case values of all dimensions can be passed to the target document. In other words, the whole dimensional context of a cell can be passed. Now let us suppose the user wishes to click on a cell and pass to the target document the value of the level family of product dimension and year

of time dimension. It should create two parameters one for family where dimension is product, hierarchy is product, level is product family and one for year parameter where dimension is time, hierarchy is time and level is year. Let us see what happens when user clicks on a cell. Depending on the selected cell, the analytical driver family of the target document will have a different value: it will be the name of the context member (of the selected cell) of the “Product” dimension, i.e. the [Product] hierarchy, at [Product].[ProductFamily] level. Look at the following Table for some examples:

Table 5.15: Context member on product dimension

Context member on Product dimension	“Family” analytical driver value
[Product].[All Products]	[no value: it will be prompted to the user]
[Product].[All Products].[Food]	Food
[Product].[All Products].[Drink]	Drink
[Product].[All Products].[Non-Consumable]	Non-Consumable
[Product].[All Products].[Food].[Snacks]	Food
[Product].[All Products].[Food].[Snacks].[Candy]	Food

Let us have a look at the template. Syntax used to set cross navigation shows how to use the cross navigation tag:

Listing 5.38: Syntax used to set cross navigation.

```

1  <CROSS_NAVIGATION>
2  <PARAMETERS>
3    <PARAMETER name="family" dimension="Product" hierarchy="[Product]" level="[Product].[Product_
↩Family]" />
4    <PARAMETER name="year" dimension="Time" hierarchy="[Time]" level="[Time].[Year]" />
5  </PARAMETERS>
6  </CROSS_NAVIGATION>

```

A green arrow will be visible in the toolbar to show that cross navigation is enabled. When user clicks on that icon in each cell a green arrow will be displayed in each cell. User can click on that icon to start cross navigation from a cell.

5.16 What-if analysis

The **What-if** analysis is the capability to make hypothesis and see how these impacts on the business. In practise user can perform What-if analysis using an extension of the OLAP tool. The process of What-if is composed in three phases:

- data preparation,
- workflow definition,
- hypothesis definition.

We start then focusing on this last phase.

5.16.1 Interface details

The workflow has an impact on data visualization. A user can understand the state of the workflow looking at the What-if section of the sidebar. There are three possibilities described in the following.

- The user can perform the What-if analysis: in this case the What-If section contains the buttons to edit values; see figure below to check those buttons.
- The schema is locked by another user. In this case a message appears with the name of the active user in the workflow as shown below.



Fig. 5.453: What-If buttons inside the sidebar.



Fig. 5.454: The What-If is used by another user.

- The workflow is finished.

We briefly recall the functionality of the main buttons:

- Unlock model: it changes the state of the workflow in order to move control to next user.
- Save: it persists modification in the database.
- Save as new version: it persists modification in the database in a new version.
- Undo: it undoes last modification.
- Delete versions: it opens a wizard user can use to delete versions.
- Output wizard: it allows user to export the edit cube in two different formats, table and csv in the specific.

5.16.2 Meta-language description

We saw that the What-If engine allows the final user to change data values. Here we see how it is possible to modify a cell value through a formula, unconditionally from the aggregation level of the cell. The formula must be written using a particular language called **meta-language** that is described below. Firstly the available arithmetic symbols are: $+ - * / () \%$.

The computation $100 + 10\%$ is a simple example of usage of the operation $\%$. Note that the formula can start with “ \equiv ”, but this is not mandatory.

To activate the editing of a measure cell that is not shown in the OLAP you must first click on the filter icon of the measure filter card and check the target measure. Then select the version you want to use and change values of figure below shows where are available these objects in the interface.

Then double-click on the target measure cell and a box will appear allowing you to insert a formula. Type the computation syntax and click on the $f(x)$ icon to convalidate it or cancel it, as shown below.

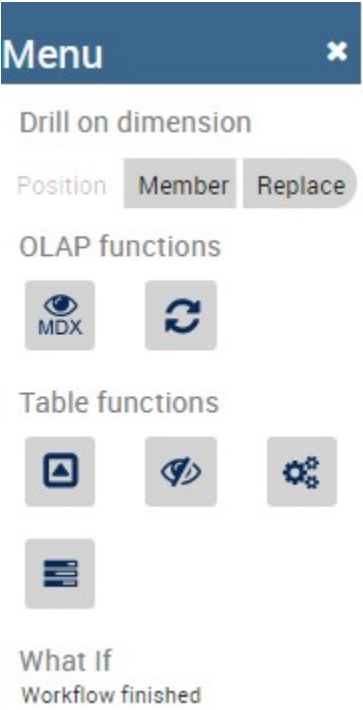


Fig. 5.455: The What-If is finished.

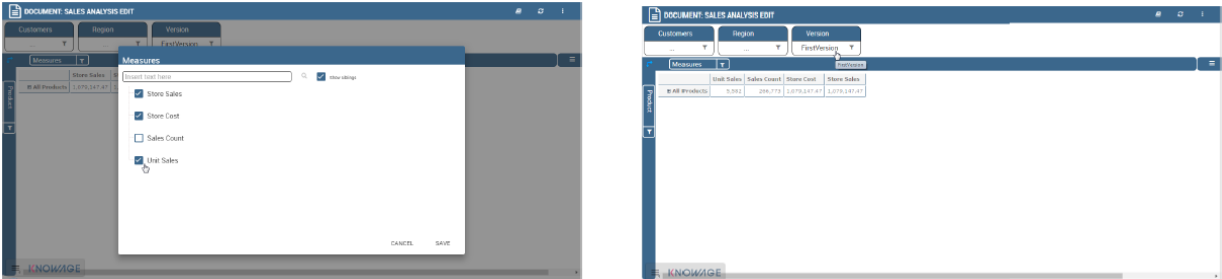


Fig. 5.456: Checking measures and selecting version.

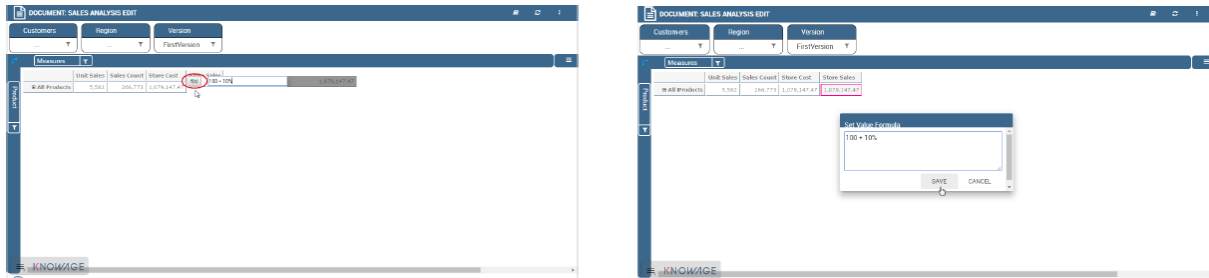


Fig. 5.457: Inserting formula and its convalidation.

We stress that you can also refer to members that are not included in the tuple represented by the cell that is going to be modified. Let's see some examples. For example suppose the cell refers to the following tuple reported in Code below:

Listing 5.39: Product.Deli

```
1 [Measures].[Store Sales], [Product].[Food].[Deli], [Version].[0],
2 [Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

You can refer to the tuple in the next code with just Product.Eggs and at the same time to the tuple in the second code below with just Product.Eggs; Measures.Unit Sales

Listing 5.40: Product.Eggs

```
1 [Measures].[Store Sales], [Product].[Food].[Eggs], [Version].[0],
2 [Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

Listing 5.41: Product.Eggs; Measures.Unit Sales

```
1 [Measures].[Unit Sales], [Product].[Food].[Eggs], [Version].[0],
2 [Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

Note that if you create a formula on a cell and you want to move it along a dimension (for example the cell refers to member Time.2016 and you want to get value for Time.2017) you have to refer to a member of same level. So for example you can get value of the cell for Time.2017, but not for Time.2017.May.

The syntax is as the one shown in Referring to different members or, in case you are using another hierarchy, as in the second code below where you can concatenate different members with “;”.

Listing 5.42: Referring to different members.

```
1 <dimension's name>.<member's name>or[<dimension's name>].<member's name>]
```

Listing 5.43: Referring to different members of another hierarchy.

```
1 <dimension's name>.<hierarchy's name>.<member's name>or[<dimension's name>].< hierarchy's name>].<member
  ↳'s name>]
```

You can also refer to members that are on the same level but they are not sibling members: suppose that, for example, the cell's tuple is as in Code below:

Listing 5.44: Example of cell's tuple.

```
1 [Measures].[Store Sales], [Product].[Food].[Deli], [Version].[0],
2 [Region].[Mexico Central], [Customers].[All Customers], [Customers].[All Customers]
```

Note that you can refer to the tuple

Listing 5.45: Example of cell's tuple.

```
1 [Measures].[Store Sales], [Product].[Drink].[Alcoholic Beverages],
2 [Version].[0], [Region].[Mexico Central], [Customers].[All Customers],
3 [Customers].[All Customers]
```

just with:

Listing 5.46: Shorten syntax code.

```
1 [Product].[Drink.Alcoholic Beverages]
```

Another example from Code below

Listing 5.47: Example of cell's tuple.

```
1 [Measures].[Store Sales], [Product].[Food].[Deli].[Meat],
2 [Version].[0], [Region].[Mexico Central], [Customers].[All Customers],
```

to Code below

Listing 5.48: Example of cell's tuple.

```
1 [Measures].[Store Sales], [Product].[Drink].[Alcoholic Beverages].[Beer and Wine], [Version].[0],
2 [Region].[Mexico Central], [Customers].[AllCustomers], [Customers].[All Customers]
```

is as in the following code

Listing 5.49: Used expression.

```
1 [Product].[Drink.Alcoholic Beverages.Beer and Wine]
```

Note that the last part of the expression is the portion of the path to the target member that differs from the path of the cell's member. Some other examples:

Listing 5.50: Further example.

```
1 [Product].[Food]
```

5.16.3 What-if analysis implementation

In this chapter we will deal with some technical features of the What-If analysis that can be handled only by expert users.

5.16.3.1 Workflow description*

When you perform a what-if analysis the schema is shared in order to be used as a data source. Therefore each time a document linked to a schema can be edited only by one user per time. This behaviour is managed by the Workflow of the schema. The administrator can configure a workflow opening the details of the model in OLAP schema catalogue, selecting the schema and going on the workflow tab available on the top of the right sided area. The tab is red circled below.

Referring to the next figure, the interface for the definition of the workflow is composed of a double list where

- the **available users** area contains all the users,
- the **workflow** area contains the sequence of users for the workflow.

When an administrator clicks on the user in the list “available users” the user will be added in the workflow as shown in Figure 10.3.

Administrator can move the users in the sequence or remove them clicking on the “action buttons”. When the workflow is defined, the administrator can start it clicking on the button start. To start a workflow means to enable the first user of the sequence to apply the what-if on that schema. When a workflow is started it can not be edited by anyone else

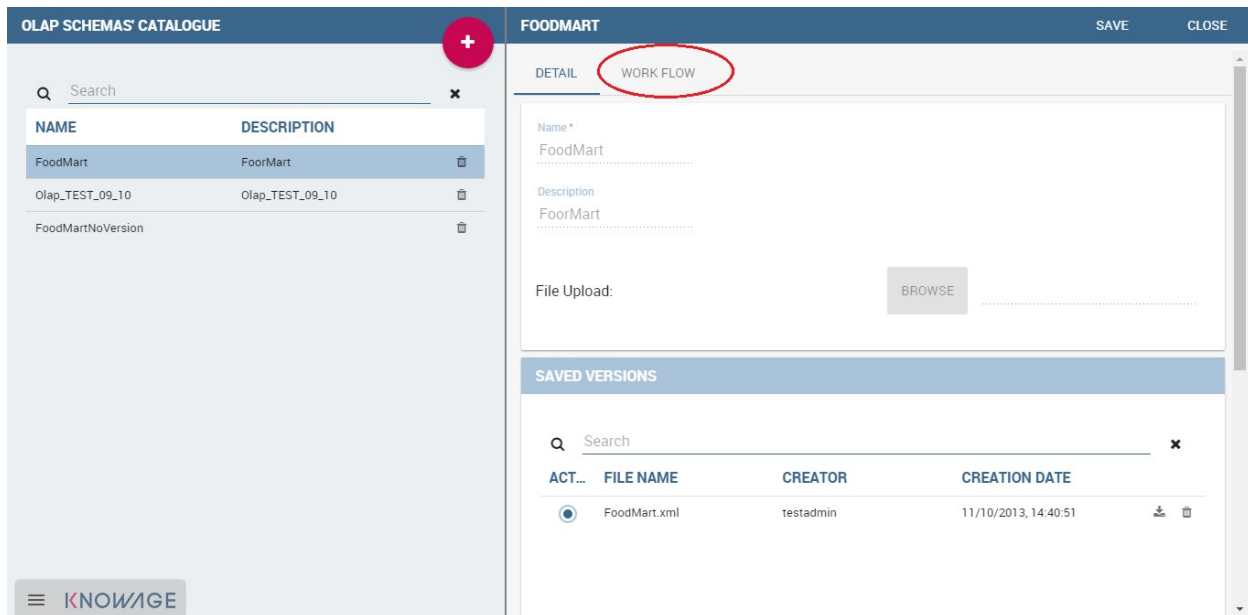


Fig. 5.458: Workflow tab.

and an icon appears in the row of actual active user so that the administrators can monitor the state of the schema. An example is provided by Figure 10.4

5.16.3.2 Schema definition*

As we foresaid, the What-If analysis requires some modification on the database. The first step is to create a new table in the database to store the named version of the modified data. The user will then change the values of the cube; it is then mandatory to create a new table with a structure similar to the analysed cube and a new table (wbversion) that will contain the versioning of the definitions set in the analysis.

Therefore the structure of the new fact table should contain:

- all the foreign keys to the dimensions (all the ones visible in the cube),
- all the editable measures,
- a new numeric column that is a foreign key referencing the version table.

In Figure below there is an example where the cube is sales_fact_1998 and the new table is sales_fact_1998_virtual.

The sales_fact_1998_virtual table should be initialized with the same data contained in sales_fact_1998 plus 0 as version; the wbversion table should be initialized with one record with wbversion = 0 and a name plus a description for the “original values”.

5.16.3.3 Changes in the mondrian schema*

Now you should map the new tables in the mondrian schema. In order to merge the fact table and the table with the editable measure we create a virtual cube. A virtual cube is a special cube where the values are the result of the join of other cubes. In our case the join keys are the dimensions. The actions to be performed in the mondrian schema are listed right below.

- To create a new “Version” dimension as in Changing the Mondrian Schema.

DETAIL
WORK FLOW

AVAILABLE USERS

Q
Search
x

Username	Name
test_admin	TEST ADMIN
test_dev	test_dev
test_user	TEST USER
test_user2	test_user2
test_workspace	test_workspace

USERS WORK FLOW

Q
Search
x

Username	Name
----------	------

Fig. 5.459: Workflow tab interface.

DETAIL

WORK FLOW

AVAILABLE USERS

Q

Search

X

Username	Name
test_user	TEST USER
test_user2	test_user2

USERS WORK FLOW

Q

Search

X

Username	Name			
test_dev	test_dev	↑	↓	×
test_admin	TEST ADMIN	↑	↓	×
test_workspace	test_workspace	↑	↓	×

Fig. 5.460: Selecting users for workflows.

AVAILABLE USERS		USERS WORK FLOW	
<div> <div>Q</div> <div>Search</div> <div>×</div> </div>		<div> <div>Q</div> <div>Search</div> <div>×</div> </div>	
Username	Name	Username	Name
test_user	TEST USER	test_dev	test_dev ✓
test_user2	test_user2	test_admin	TEST ADMIN
		test_workspace	test_workspace

Fig. 5.461: Selecting users for workflows.

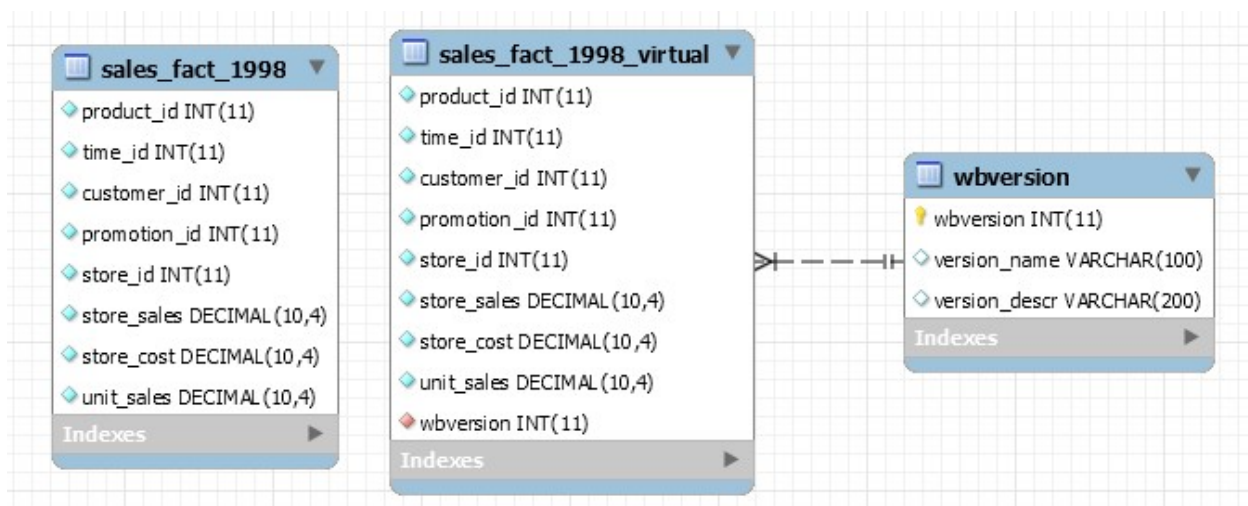


Fig. 5.462: Cube and new virtual table example.

Listing 5.51: Changing the Mondrian Schema.

```

1 <Dimension name="Version">
2   <Hierarchy hasAll="false" primaryKey="wbversion"
3     defaultMember="[Version].[0]" >
4     <Table name="wbversion"/>
5     <Level name="Version" column="wbversion" uniqueMembers="true"
6       captionColumn="version_name"/>
7   </Hierarchy>
8 </Dimension>

```

- To create the mapping of the editable cube (in our example the table sales_fact_1998_virtual) as shown in Code Creating the mapping of the editable cube.

Listing 5.52: Creating the mapping of the editable cube.

```

1 <Cube name="Sales_Edit">
2   <Table name="sales_fact_1998_virtual"/>
3   <DimensionUsage name="Product" source="Product"
4     foreignKey="product_id" />
5   <DimensionUsage name="Region" source="Region"
6     foreignKey="store_id"/>
7   <DimensionUsage name="Customers" source="Customers" foreignKey="customer_id"/>
8   <DimensionUsage name="Version" source="Version"
9     foreignKey="wbversion"/>
10  <Measure name="Store Sales" column="store_sales" aggregator="sum"
11    formatString="#,###.00"/>
12 </Cube>

```

The name of the cube ("Sales_Edit") is the value of the edit Cube attribute of the tag scenario in the template. Note that the name of the dimension Version must be exactly "Version"!!

- To create the virtual cube that will contain the mapping of the columns as in Code below.

Listing 5.53: Creating the virtual cube.

```

1 <VirtualCube name="Sales_V">
2   <CubeUsages>
3     <CubeUsage cubeName="Sales_Edit" ignoreUnrelatedDimensions="true"/>
4     <CubeUsage cubeName="Sales" ignoreUnrelatedDimensions="true"/>
5   </CubeUsages>
6
7   <VirtualCubeDimension cubeName="Sales" name="Customers"/>
8   <VirtualCubeDimension cubeName="Sales" name="Product"/>
9   <VirtualCubeDimension cubeName="Sales" name="Region"/>
10  <VirtualCubeDimension cubeName="Sales_Edit" name="Customers"/>
11  <VirtualCubeDimension cubeName="Sales_Edit" name="Product"/>
12  <VirtualCubeDimension cubeName="Sales_Edit" name="Region"/>
13  <VirtualCubeDimension cubeName="Sales_Edit" name="Version"/>
14  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Unit Sales Original]" visible="false"/>
15  <VirtualCubeMeasure cubeName="Sales" name="[Measures].[Sales Count Original]" visible="false"/>
16  <VirtualCubeMeasure cubeName="Sales_Edit" name="[Measures].[Store Sales]" visible="true"/>
17  <VirtualCubeMeasure cubeName="Sales_Edit" name="[Measures].[Store Cost]" visible="true"/>
18
19  <CalculatedMember name="Sales Count" dimension="Measures">
20    <Formula>VALIDMEASURE([Measures].[Sales Count Original])</Formula>
21  </CalculatedMember>
22
23  <CalculatedMember name="Unit Sales" dimension="Measures">
24    <Formula>VALIDMEASURE([Measures].[Unit Sales Original])</Formula>
25  </CalculatedMember>
26 </VirtualCube>

```

Specifically, in the virtual cube you should specify:

- the list of cubes to be joined (CubeUsages);
- the list of the dimensions of the cube (as you can see it contains all the common dimensions, plus the Version that belongs only to the editable cube);
- the list of the measures. You can perceive that there is a calculated member for the measure Sales Count Original (Sales Count Original is the name of a measure in the Sales cube). This is a trick for the not editable measures. This type of measure lives only in the DWH cube and not in the editable cube. This is due to the fact that the engine doesn't know how to give a value for these measures for the different values of the Version dimension (remember that only the editable cube has the Version dimension). The calculated field solves this problem propagating the same version of the not editable (and versionable) measure for all the version.

Now all the MDX queries can be performed in the virtual cube.

5.17 Glossary and data lineage

The **Glossary** functionality offers a way to find documents by browsing an index page.

5.17.1 Glossary management

The Glossary management is split in two sections. Once logged in, the user can find the two menu items: **Glossary Definition** and **Glossary Usage**, as showed below.

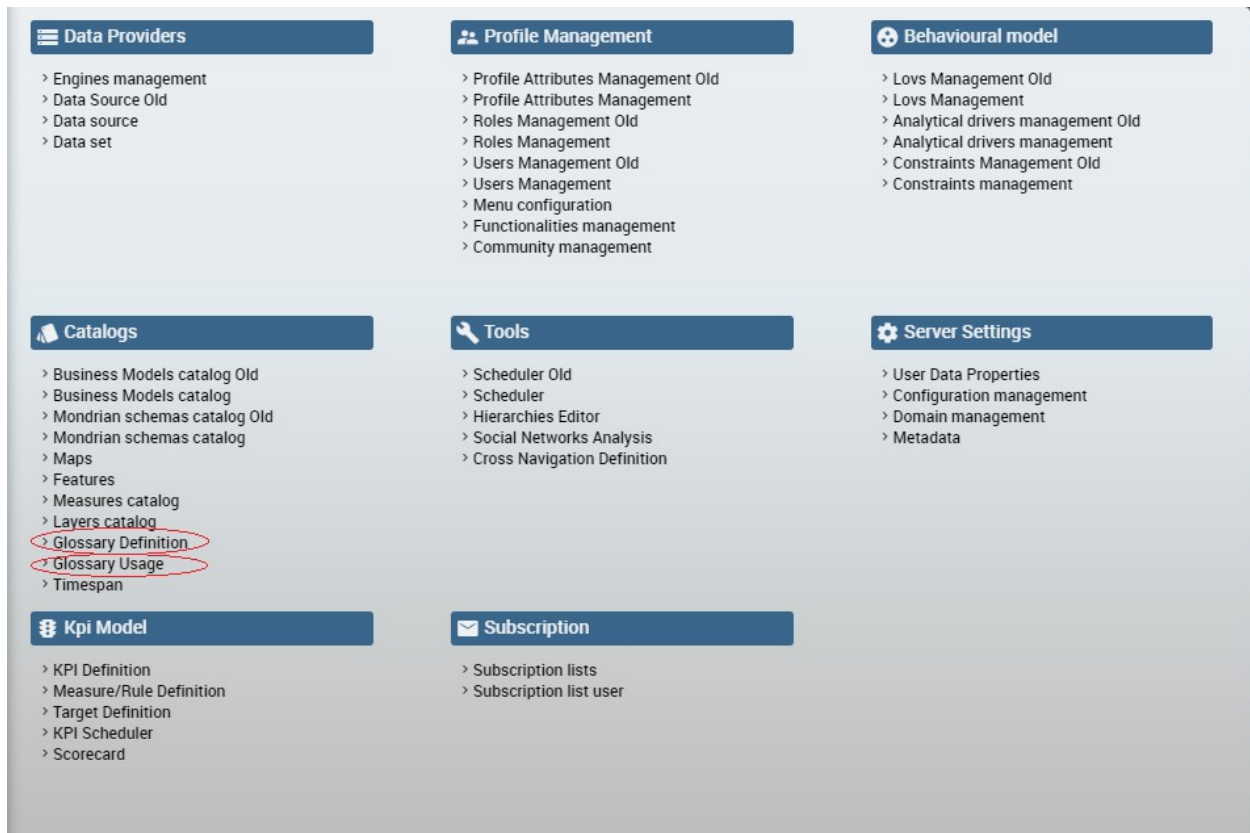


Fig. 5.463: Glossary menu items.

To create a new glossary, click on the Glossary Definition menu item that you can find under the **Catalogs** section of the Knowage interface. As shown in the figure below the page contains two areas:

- **Word:** here there is a list of terms. The latter are used as labels to attach to analytical objects as datasets or documents in order to link those objects to the glossary;
- **Glossary:** it is intended a hierarchical structure made up of “Words”.

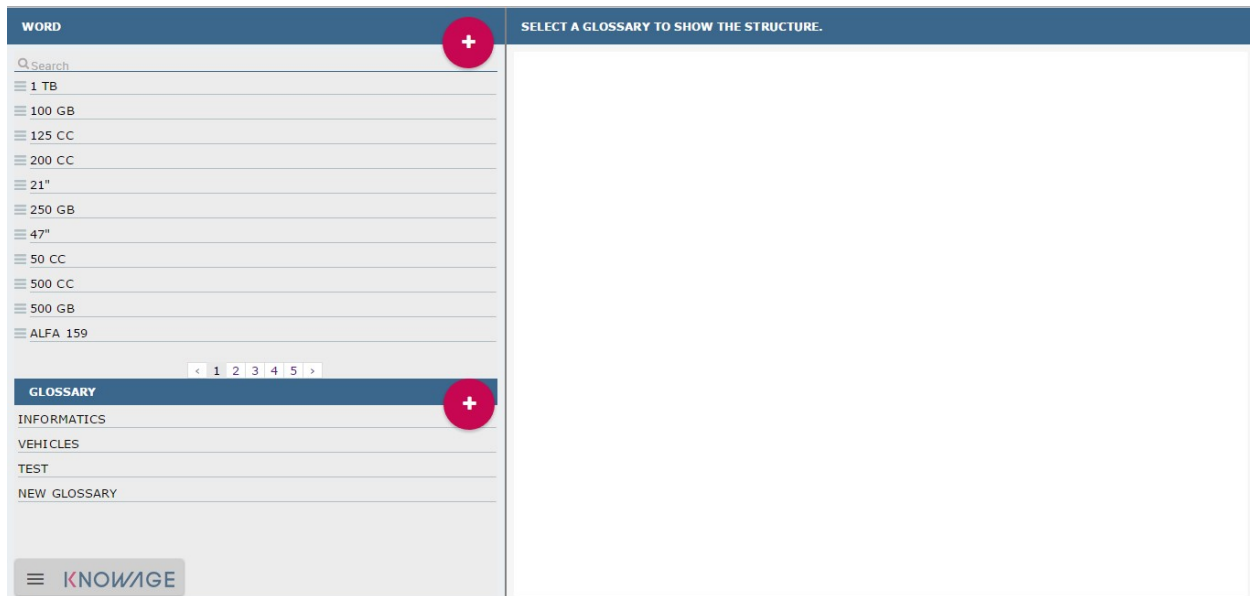


Fig. 5.464: Glossary definition window.

In the following we give some more technical information about these two sections.

In the “Word” area are listed, if any, the words created in a previous moment. To explore the detail of each of them, the user just has to right click on it. A panel containing three features will be shown, as figure below highlights .

When exploring the detail a wizard will pop up showing the following characteristics:

- description,
- state,
- category,
- a formula description,
- a list of links to other words,
- a list of attributes which can be added a value to.

The same panel can be used to modify or delete the word.

To add a new word, click on the “Plus” icon available in the right up corner of the “Word” area as the figure below shows. A format will be open in the right half part of the screen. Insert “Name” and “Description”, which are all mandatory fields and add additional details by necessity. Then click on the Save button. Observe that it is possible to look a “Word” up using the dedicated filter available at the top of the Words list. Type a string in the box and the research will start automatically. Remember to cancel the string from the box to get back to the entire list.

In the “Glossary” area are listed, if any, all glossaries created in a previous moment. To explore an existing glossary the user must simply click on the item. Figure below shows an example. Here the hierarchical structure of the glossary is underlined. To add a new glossary click on the “Plus” icon at the right top corner of the designated area.

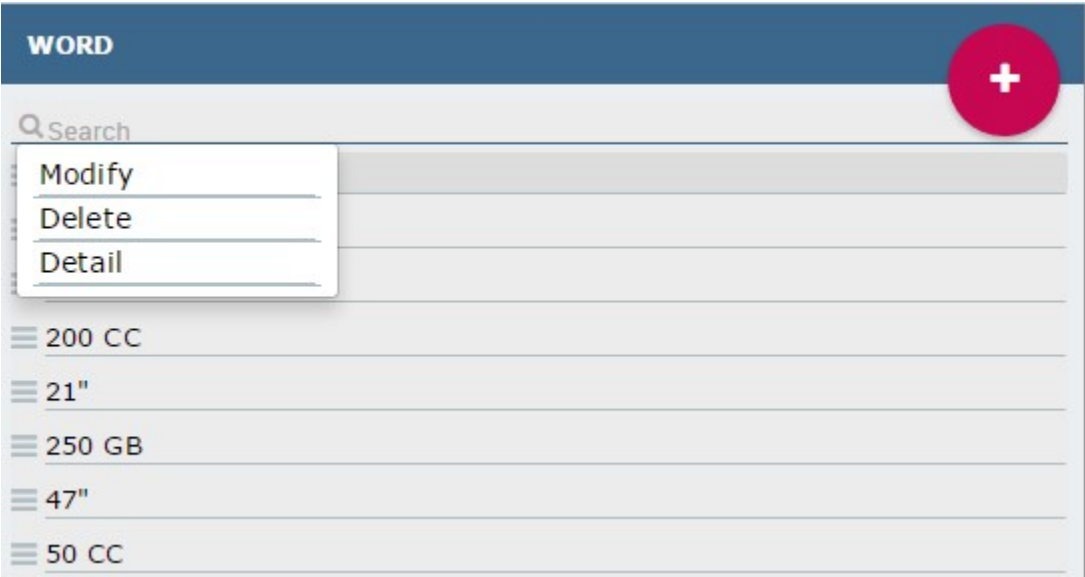


Fig. 5.465: Exploring an existing word.

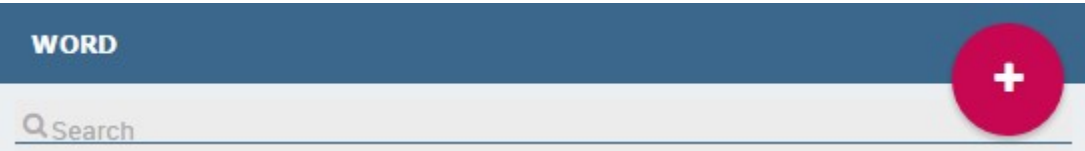


Fig. 5.466: Add a new word.

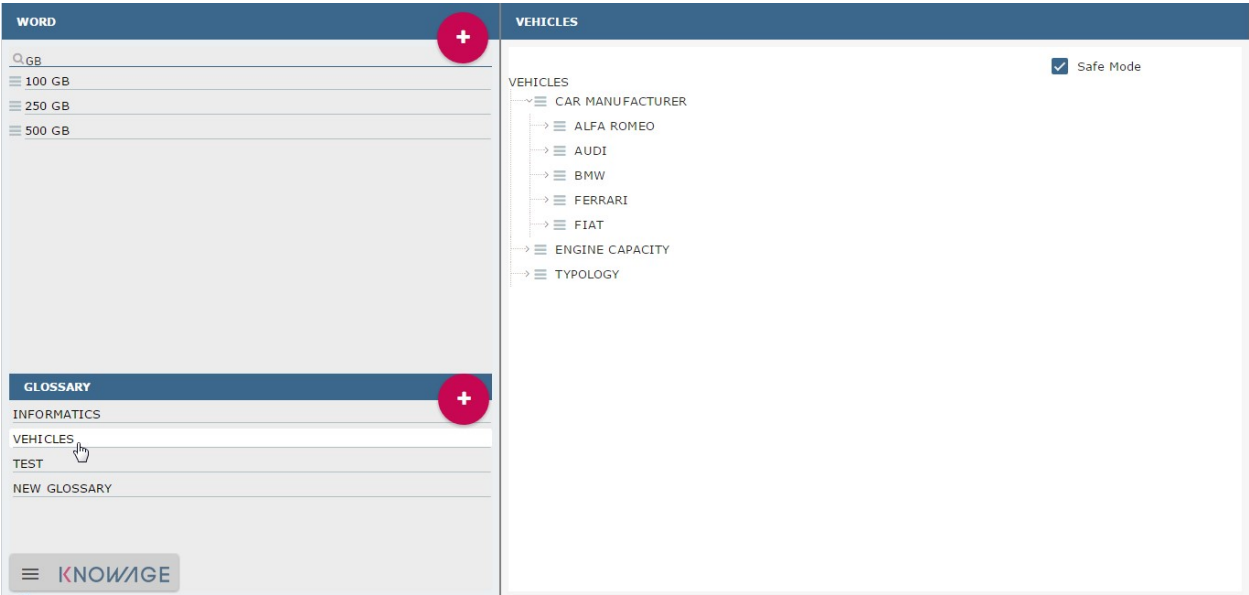


Fig. 5.467: Exploring a glossary from the menu.

Right clicking on the glossary label as shown in the following figure (right side) the user can add a new child. The “New Node” wizard will open. It is mandatory to give a Name to the node while it is recommended to add a Code and a Description. Once the user has set the nodes, it is possible to add children or words to each of them.

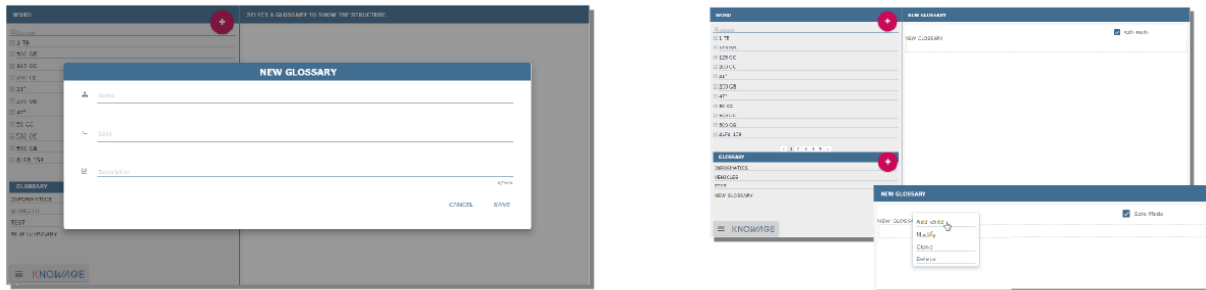


Fig. 5.468: (Left) New glossary wizard. (Right) Add a new child to the glossary.

In particular, if one right clicks on the node name, as in the next figure, a panel will be opened. It allows the user to add one (or more) child or word to the node. In both cases, the user must fill in the mandatory fields. We remark that if the user chooses to add a word through the panel item, the word will be created from scratch and added to the Word list after saving it. To add an existing word the user has to drag and drop the word from the list to the node. Notice that at the right top corner of the designated area a **Safe Mode** button is available. Select it if you want to assure that it cannot be modified by a user with no administrator permissions.

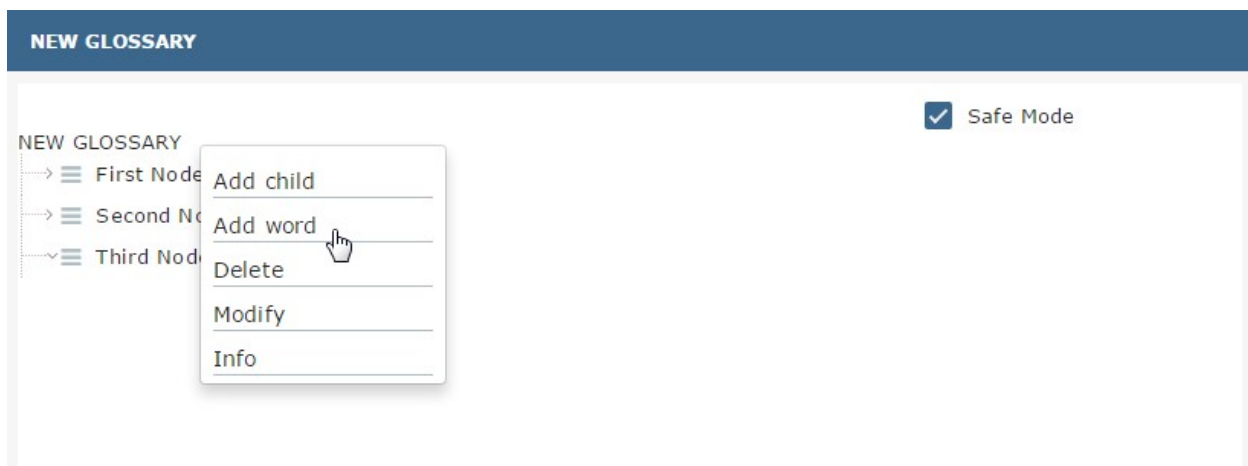


Fig. 5.469: Add items to the node(s).

Complete the tree structure of the glossary. Use the panel features of each node or of the glossary itself (remember to right click on the items to get such a panel) to add, modify, inspect or delete elements.

5.17.2 Glossary Usage

This functionality is profiled accordingly to the user role and it includes features that allow to

- visualize the glossary,
- visualize the associations,
- manage the associations between the glossary and the documents,

- manage the associations between the glossary and the datasets.

Selecting **Glossary Usage** from the Catalogs contextual menu, the user encounters the page showed below. Here four tabs are available: **Glossary**, **Navigation**, **Document Management** and **Dataset Management**.

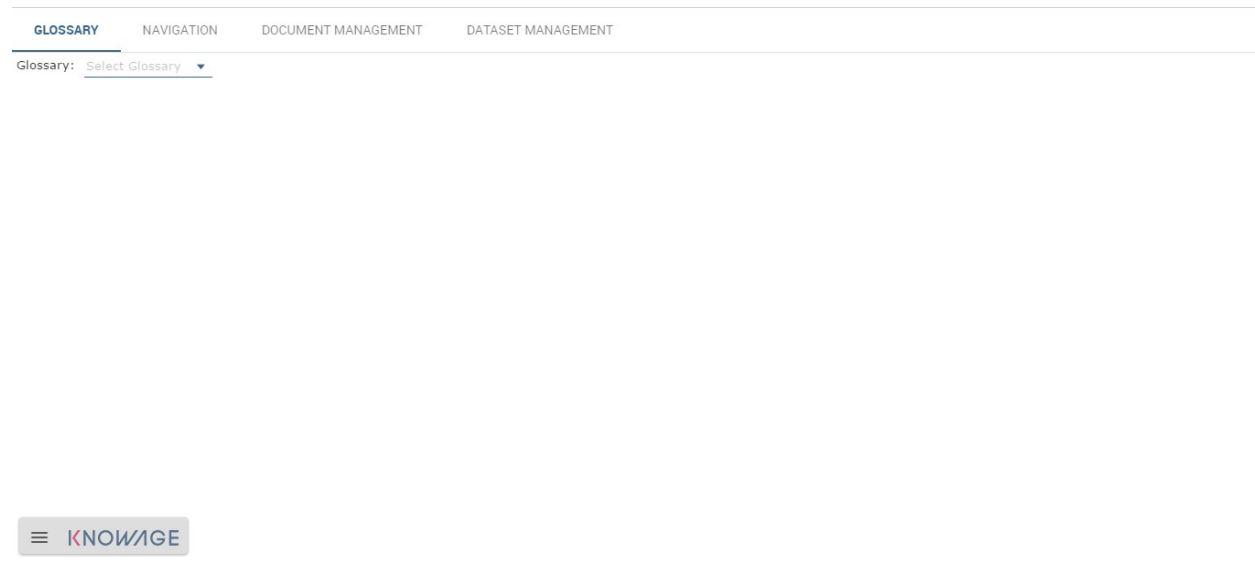



Fig. 5.470: Glossary Usage graphic interface.

The Glossary tab provides the possibility to visualize the existing glossaries. Select a glossary from the combobox available in this page to inspect its elements. Use the icon with a circled “i” to visualize the details of the related element, as shown below. Note that it is enabled the possibility to look a word up using the configured research box.

The navigation paths can be explored in the second tab. This window has an associative logic which facilitates to browse the associations. In other terms, here it is possible to check the relations between documents or datasets and words of a glossary. An example is given in the following figure.


To use this functionality, select a glossary using the designated combobox available at the top of the “word” column. The window will show all words associated to that glossary. Selecting one of those words a list of documents will be displayed in the area in the middle of the page. Use the circled i icon to inspect the document details and in addition to run it. In fact the “Run” button is available at the right bottom corner of the detail panel, as shown below.

The filters chosen by the user can be removed through the filter red icon or by selecting the **Clear Filter** button  located at the right top corner of the word list.

Note that it is possible to inspect the details of each element using the specific icon.

The Document management tab is the place where to set the associations between the analytical documents and the words of a glossary. This functionality is profiled through the authorization **Manage Glossary Technical**.

The page is made up of three columns: the “documents” one on the left, the “word” in the middle and the “glossary” on the right. To associate a word to a document or see which words are related to it the user must select a document from the list of the left sided column. Then it is mandatory to select a glossary from the combobox available on the right sided column. Finally drag and drop words from the glossary tree to the “word” column in the middle of the page. Note that the user must drag and drop the word at the beginning of the list: when a light blu box with dotted borders

appears it is possible to end the action. To deselect the choice the user can click on the icon  aside each word. This procedure is recap by figure below.

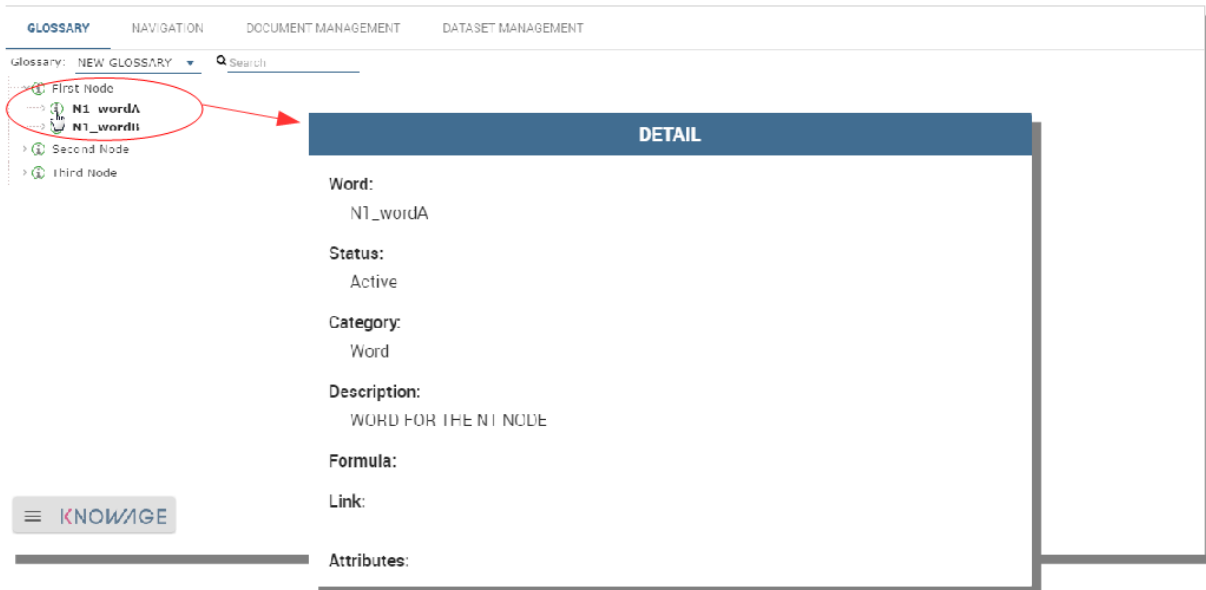


Fig. 5.471: Visualization of glossary details.

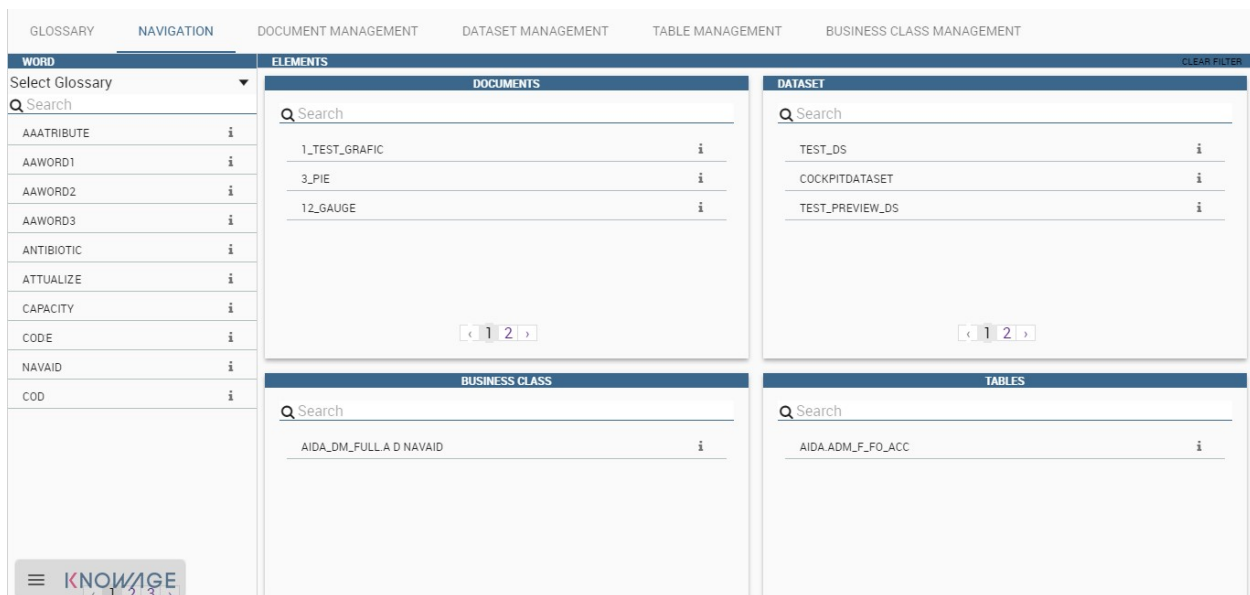


Fig. 5.472: Navigation tab window.

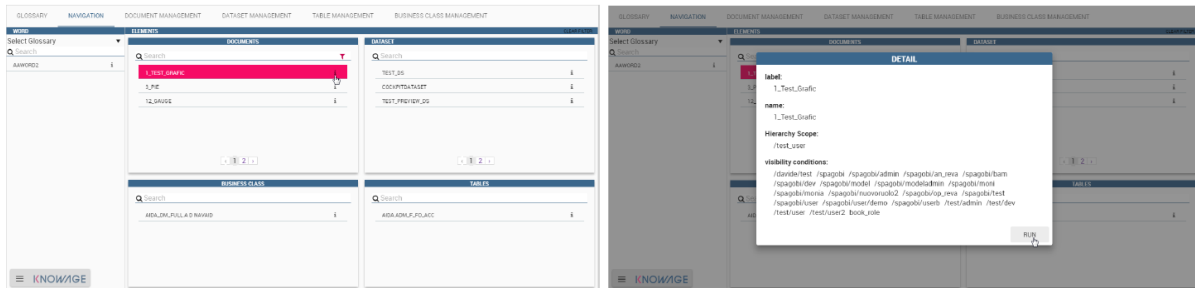


Fig. 5.473: Execution documents by means of the glossary.

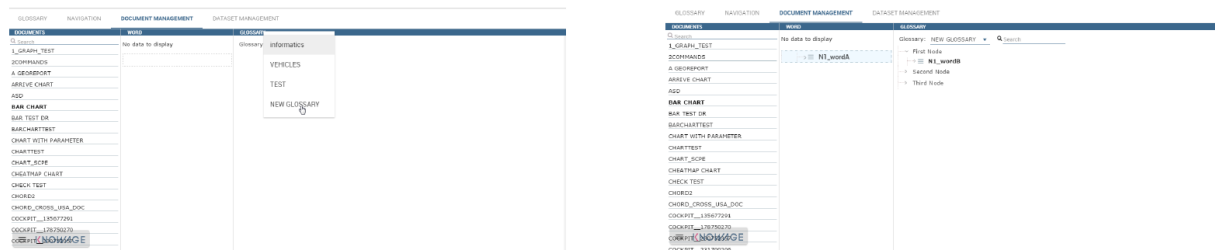


Fig. 5.474: Managing the association with a document: (Left) Select the document. (Right) Associate one (or more) word(s).

If one gets back to the navigation tab and select the glossary used in the previous step, it is possible to check the association just set.

Equally, the Dataset management feature allows the user to set the associations between datasets and glossaries. The next figure shows an example. The window is splitted in four areas: **Dataset**, **Dataset/Word**, **Column/Word** and **Glossary**. First the user must select a dataset on the left area. The chosen dataset is highlighted and its fields appear in the Column/Word area. Now, the user select a glossary using the combobox on the right side area. Finally the user can drag and drop words from the glossary tree to the dataset or the single fields of the dataset.

Once the datasets or the documents are linked to the glossaries, the user can enter the Glossary Usage menu item to browse easily the elements inside the Knowage suite.

5.17.3 Help Online functionality

The user can inspect the association of a specific analytical element (dataset, document or model) by using the **Help Online** functionality. The latter can be reached:

- from the Document Browser,
- from the toolbar of each document, once launched,
- from every dataset,
- from every entity of the Qbe model,
- from Birt reports,
- from the cockpit.

As an example, we show in figure below the graphic interface the user will encounter once he/she has launched a document and wishes to use the Help Online functionality.

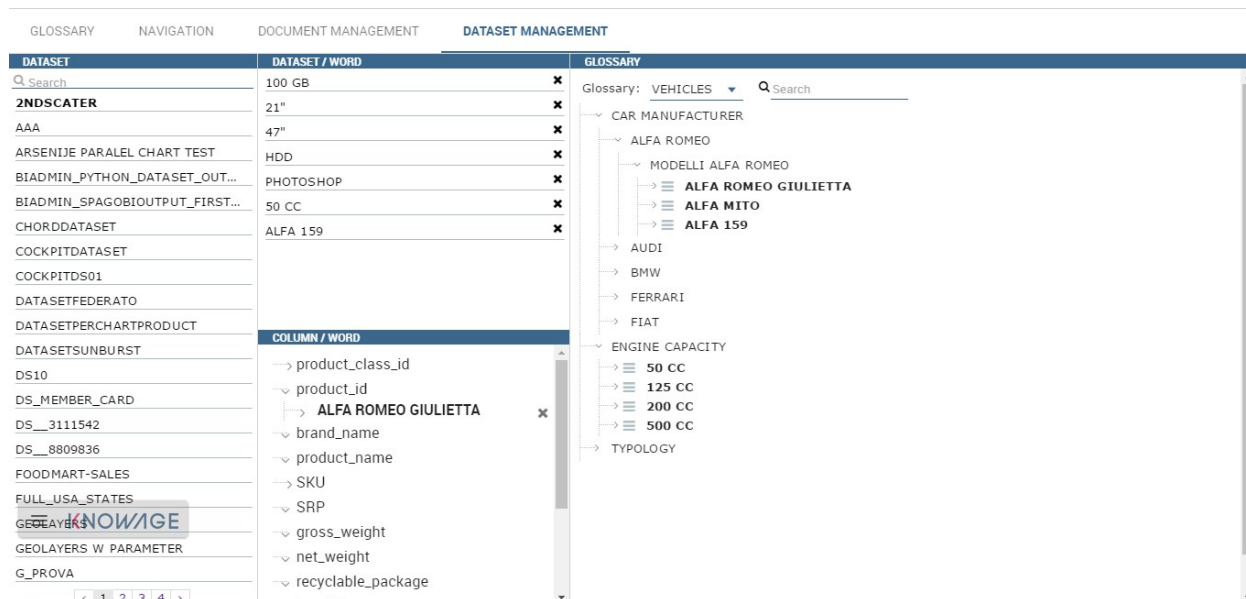


Fig. 5.475: Dataset management tab.

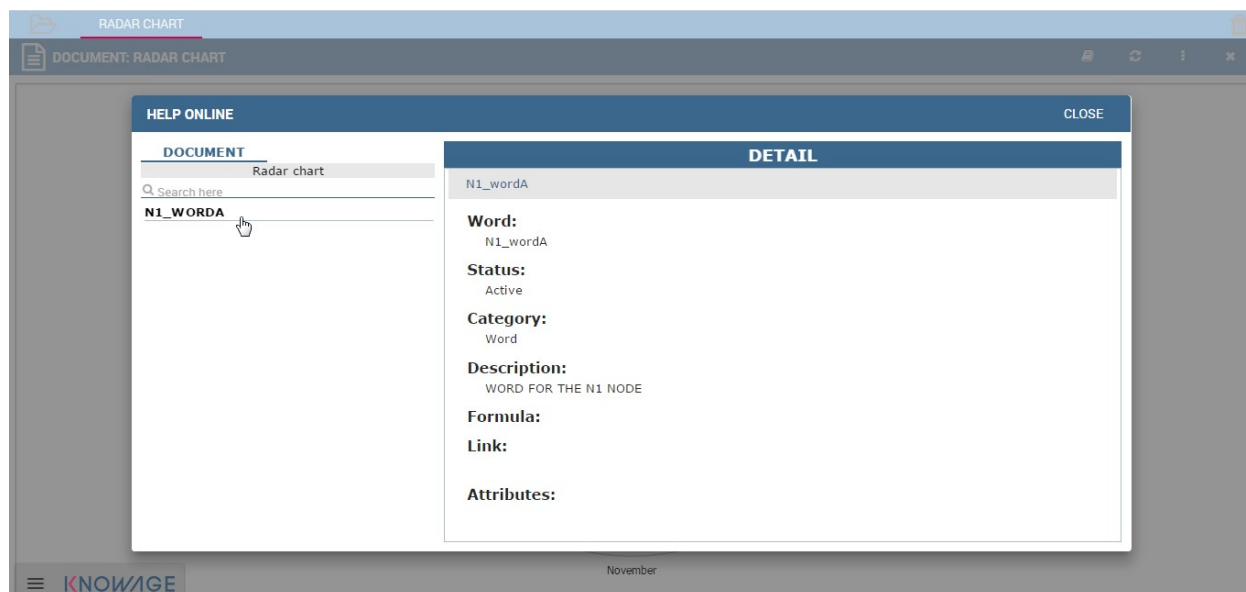


Fig. 5.476: Help Online wizard.

5.18 Key Performance Indicator

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 the next figure).

5.18.1 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 implementation.

- **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 previous step and configure thresholds.
- **Target:** it is possible to monitor the value of one or more KPIs comparing it to an additional fixed value.
- **Scheduling:** 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 below.

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 CTRL key and the space bar opens a contextual menu containing the available datasource columns and the database keywords. Refer to the following figure to have an example.

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 scheduling 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 below shows.

Generally the rule such a query can return one or more measures and possibly an attribute. An example is given below.

A typology (measure, attribute and temporal attribute) and a category can be assigned to each field returned by the query using the **Metadata** tab as highlighted in the next figure. The typology is required to associate a type to each

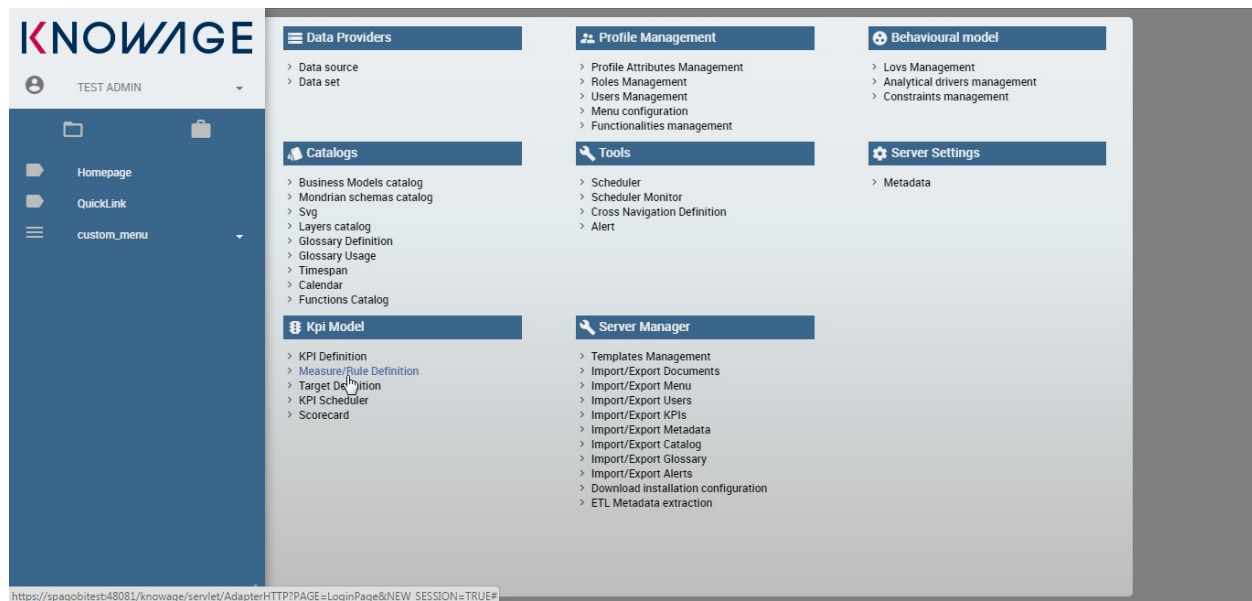


Fig. 5.477: Measure/Rule Definition menu item.

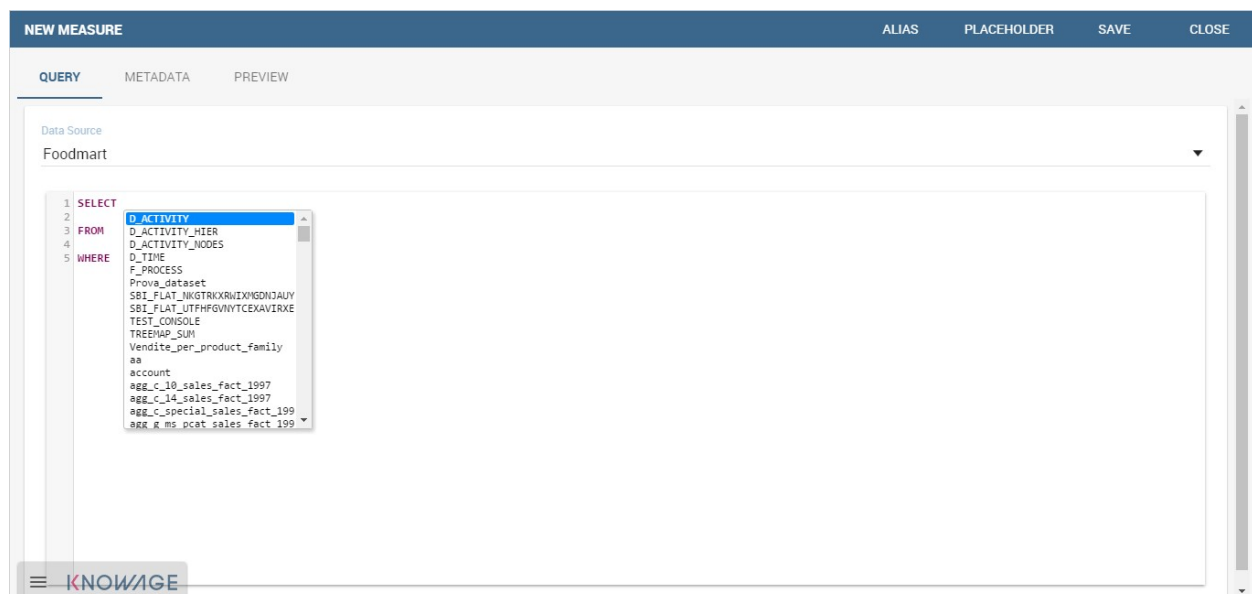


Fig. 5.478: Editing the query when defining a KPI.

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

```

Fig. 5.479: Setting placeholder in query definition.

PRODUCT_SALES

ALIAS PLACEHOLDER SAVE CLOSE

QUERY METADATA PREVIEW

Data Source

Foodmart

```

1 | SELECT
2 | p.product_name, s.store_sales, s.store_cost, s.unit_sales
3 | FROM
4 | sales_fact_1998 s join product p on p.product_id = s.product_id
5 |

```

Fig. 5.480: Query definition.

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.

Fig. 5.481: Metadata settings.

We say in advance that, it is important to distinguish these metadata categories from the required field “Category” that occurs while saving the KPI definition (see next figure).

Fig. 5.482: Category assigned when saving a KPI definition.

In 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 below).

Warning: Do not mistake metadata category with the KPI category

The category defined in the metadata tab of the “Measure definition” functionally are not the same categories selected in the tab area of the “Roles management” functionality (see the figure above). The first are used to classify the metadata while the second are needed for the profiling issue.

As we told, a proper categorization exists for the aggregations of type temporal. In fact, when associating “temporal

DETAIL	AUTHORIZATIONS	BUSINESS MODELS	DATA SETS	KPI CATEGORIES
Kpi Categories				
NAME				
<input type="checkbox"/>	PROFIT			
<input type="checkbox"/>	PRODUCT			
<input type="checkbox"/>	CUSTOMER			
<input type="checkbox"/>	SALES			
<input type="checkbox"/>	INVENTORY			
<input type="checkbox"/>	EMPLOYEE			

Fig. 5.483: KPI category.

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 the following figure. 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 result set.

Let’s now examine extra features available on the right top corner. There you can find the following tab:

- **Alias:** you can see the aliases defined in other rules; note that only the aliases of those columns 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.
- **Placeholder:** here you can check the existing placeholders. These are set in the query you’re editing or in other ones.
- **Save:** to save the query and other settings just configured.
- **Close:** to exit the rule configuration window.

KPI definition. Select the **KPI definition** item from the contextual menu of the main page of Knowage, as shown in figure below. Click on the “Plus” icon to configure a new KPI.

The window opens a first tag, entitled **Formula** (see figure below), where you must type the formula to enable calculations.

Press CTRL key and space bar simultaneously to access all measures defined in the rules, as shown below.

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 below.

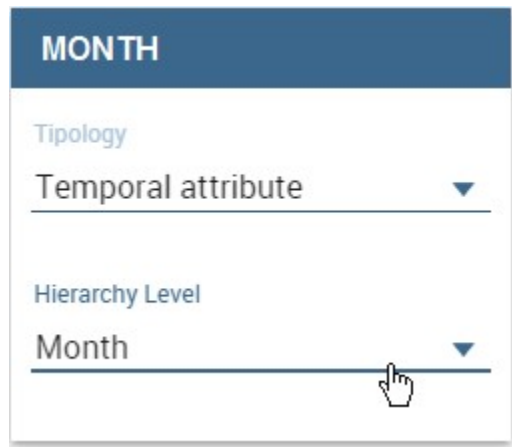


Fig. 5.484: Hierarchy level for temporal attributes.

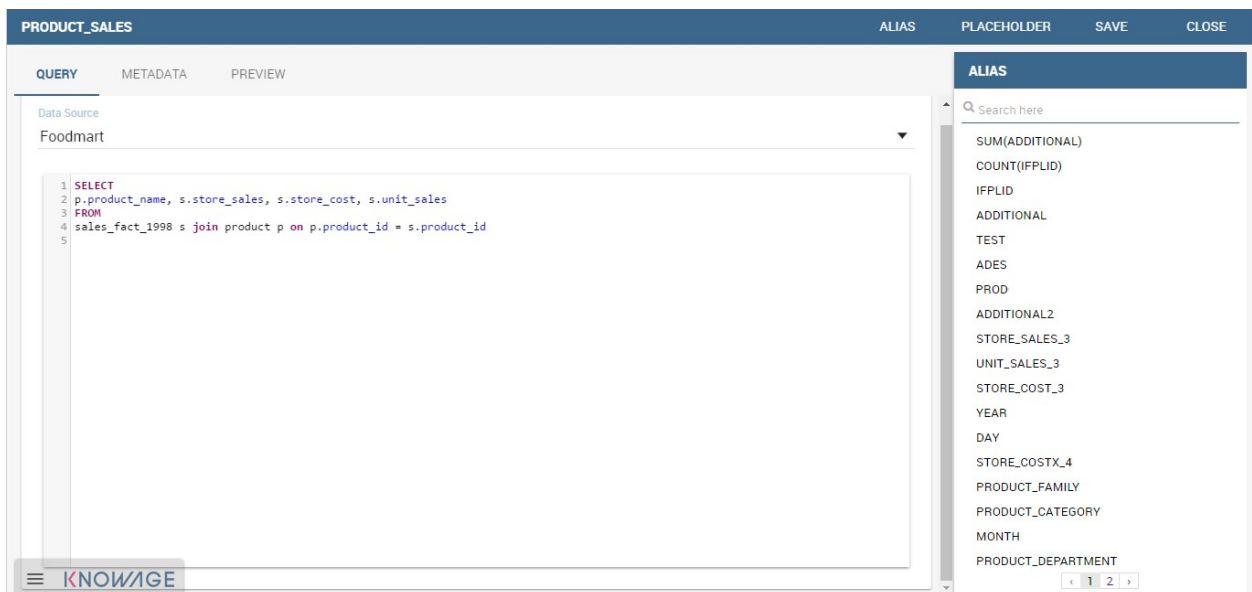


Fig. 5.485: Checking aliases.



Fig. 5.486: Setting placeholders in a query.

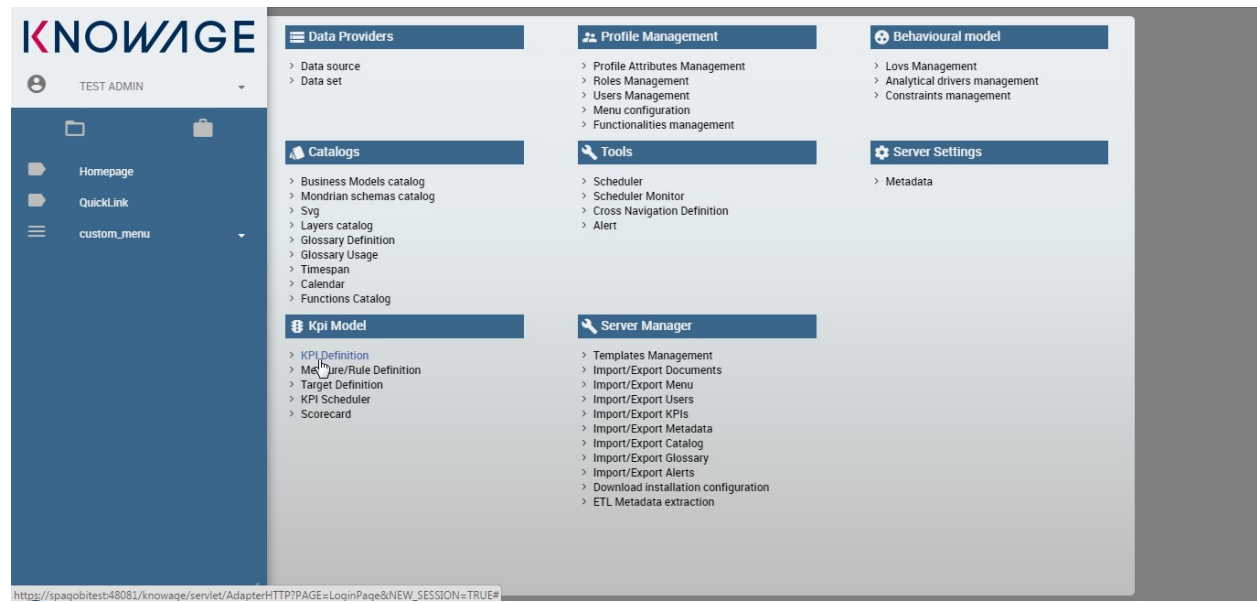


Fig. 5.487: Configure a new KPI.

NEW KPI

ALIAS SAVE CLOSE

FORMULA CARDINALITY THRESHOLD

1

KNOWAGE

Fig. 5.488: Formula definition tab.

NEW KPI

ALIAS SAVE CLOSE

FORMULA* CARDINALITY

1

CUMMITLE
ADDITIONAL5
ADDITIONAL6
ADDITIONAL1
ADDITIONAL3
DANIELE
DANIELE2
COMPUTEDTAKEOFF
store_cost
store_sales
unit_sales
STORE_COST_4
STORE_SALES_4
UNIT_SALES_4
SALES_BY_BREND
STORE_SALES_BY_BRE
testCost1

KNOWAGE

Fig. 5.489: Press ctrl and space to get measures.

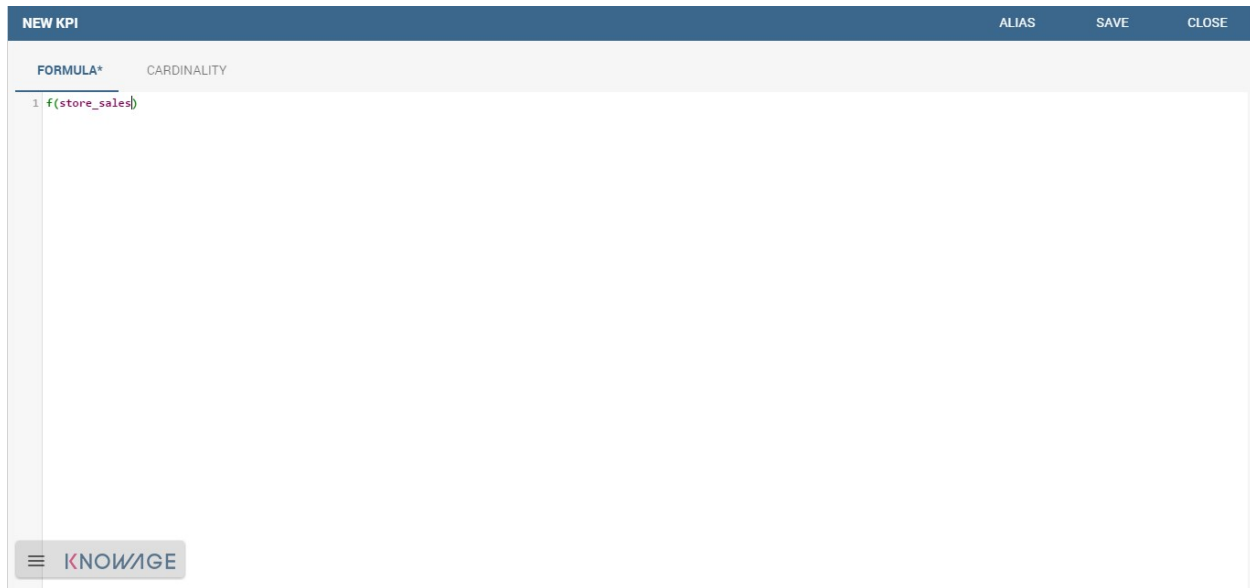


Fig. 5.490: Formula syntax.

Clicking on the $f()$ the interface opens a pop up where you can select which function apply to the measure, see figure below. Once the selection is made the formula will be autofilled with the proper syntax and you can go on editing it.

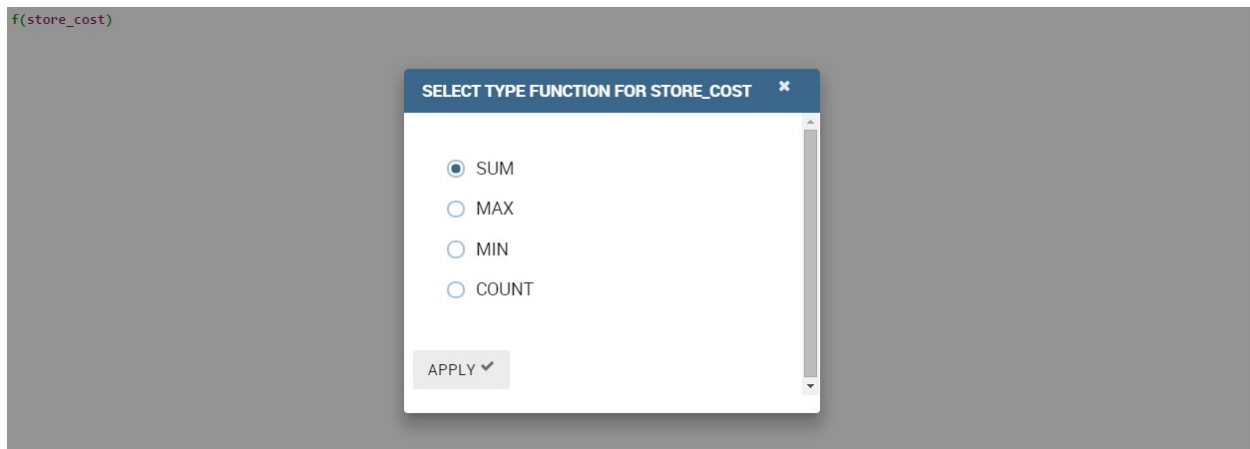


Fig. 5.491: Available functions.

Once a complete formula (an example is given in figure below) 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 below, 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.

Limit values can be set using the Threshold tab (Figure below). 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.

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**

EXAMPLE

ALIASSAVECLOSE

FORMULACARDINALITYTHRESHOLD

1 (Σ (store_sales) - Σ (store_cost)) / COUNT(unit_sales)

≡ KNOWAGE

Fig. 5.492: Complete formula example.

NEW KPI

ALIASSAVECLOSE

FORMULA*CARDINALITYTHRESHOLD

(Σ (store_sales) - Σ (store_cost)) / COUNT(unit_sales)

	store_sales	store_cost	unit_sales
product_name	✓	✓	✓

≡ KNOWAGE

Fig. 5.493: Cardinality settings example.

EXAMPLE ALIAS SAVE CLOSE

FORMULA CARDINALITY THRESHOLD

Name: StoreSales Threshold Description: 0 / 1000

Type: Range THRESHOLD LIST

	Position	label	Min	Include Min	Max	Include Max	Severity	Color
↑ ↓	1		0	<input checked="" type="checkbox"/>	50	<input checked="" type="checkbox"/>	Low	■ #00FF29
↑ ↓	2		50	<input type="checkbox"/>	100	<input checked="" type="checkbox"/>	Medium	■ #FF5C00
↑ ↓	3		100	<input type="checkbox"/>	150	<input checked="" type="checkbox"/>	High	■ #FF0000
↑ ↓	4		150	<input type="checkbox"/>	200	<input checked="" type="checkbox"/>	urgent	■ #030008

ADD NEW THRESHOLD ITEM

≡ **KNOWAGE**

Fig. 5.494: Setting thresholds.

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.

Warning: “Standard” colors for thresholds

We'll 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 next figure. 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 permissions 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.

Target. This step is not mandatory. Enter the **Target Definition** menu item as shown below.

Clicking on the “Plus” icon you can add a new target (Figure below).

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 possible 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 below.

In the KPI visualization phase, a red bold thick will be displayed on the indicated value (see next figure).

Note that once targets are set, the window in Figure 7.20 gets populated with a list. Note that here the category serves as description and only to order the records.

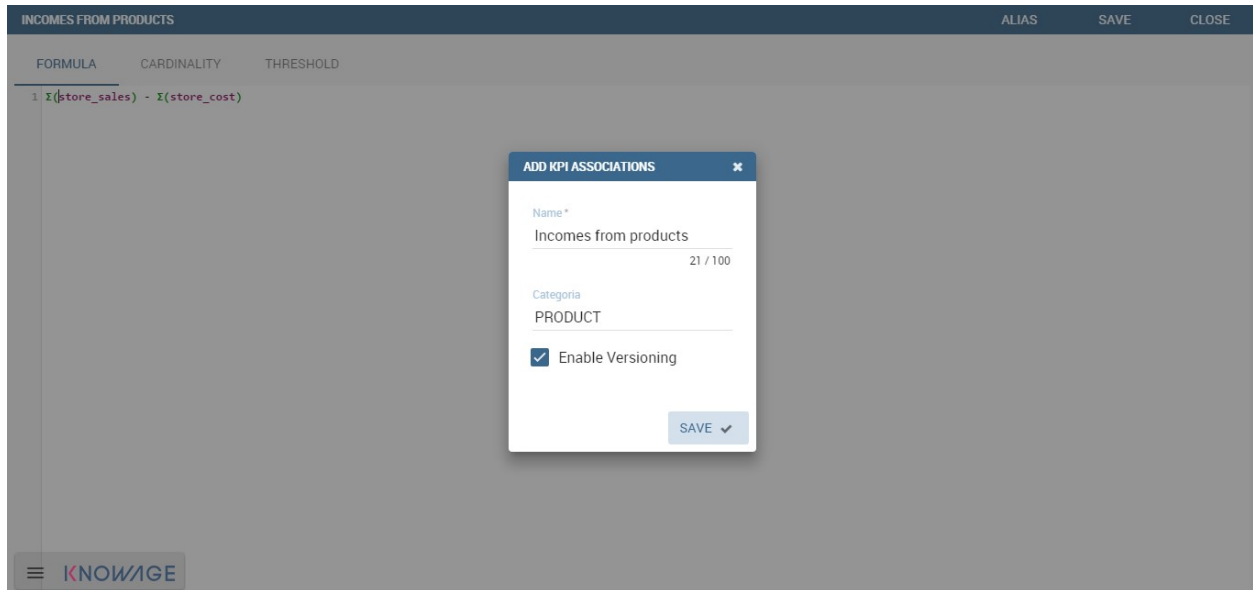


Fig. 5.495: Save the KPI definition and set category.

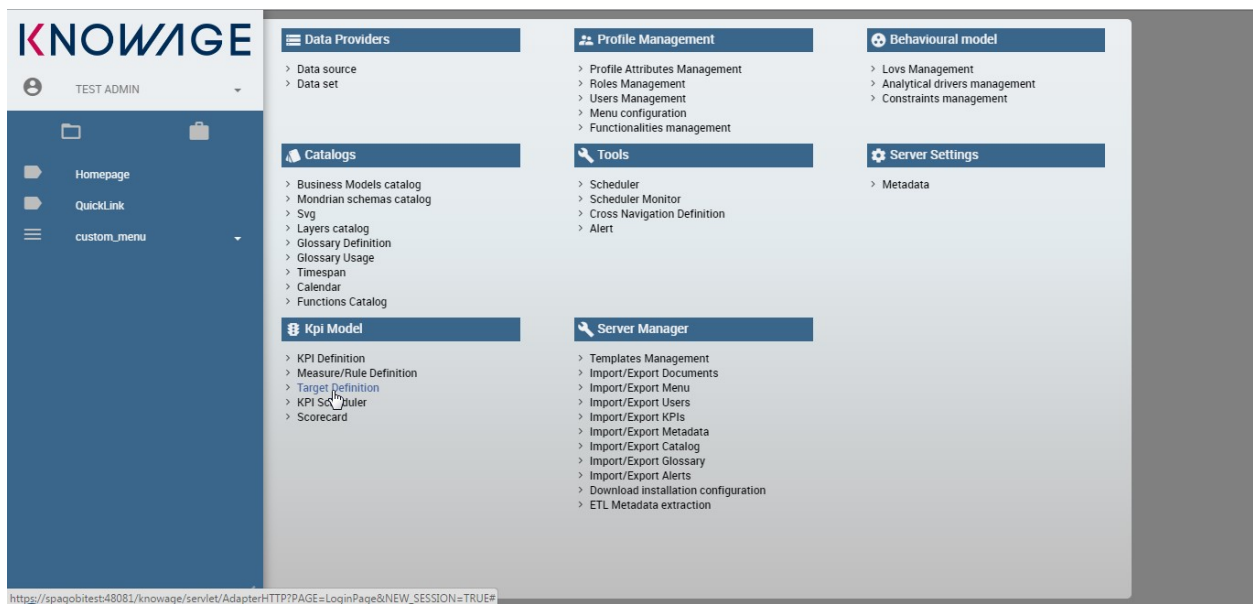


Fig. 5.496: Target Definition menu item.

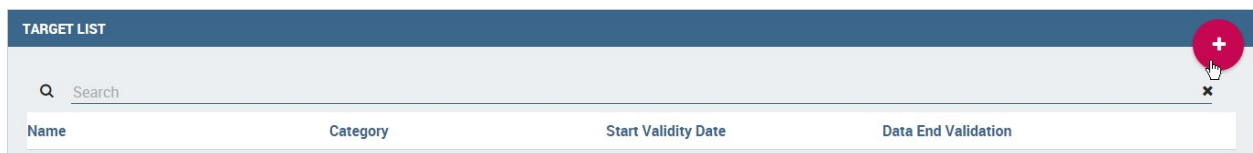


Fig. 5.497: Add a new target.

TARGET INCOME

SAVE

CLOSE

Name

Target Income

13/100

Start Validity Date

5/1/2016

End Validity Date

5/31/2016

APPLY TARGET ON KPI

KPI name	Value
Income	9500000

ADD KPI ASSOCIATION

Fig. 5.498: KPI target association.

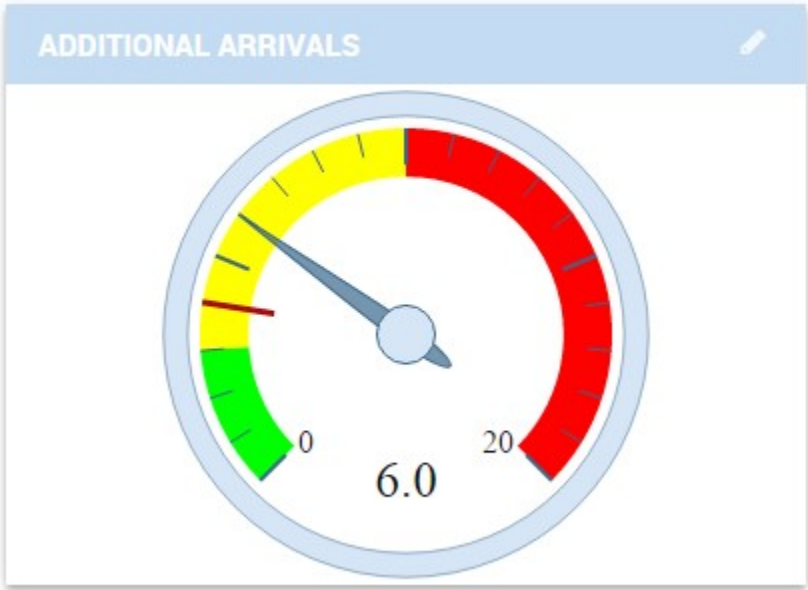


Fig. 5.499: Target mark in KPI scale of values.

Scheduling. 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 below.

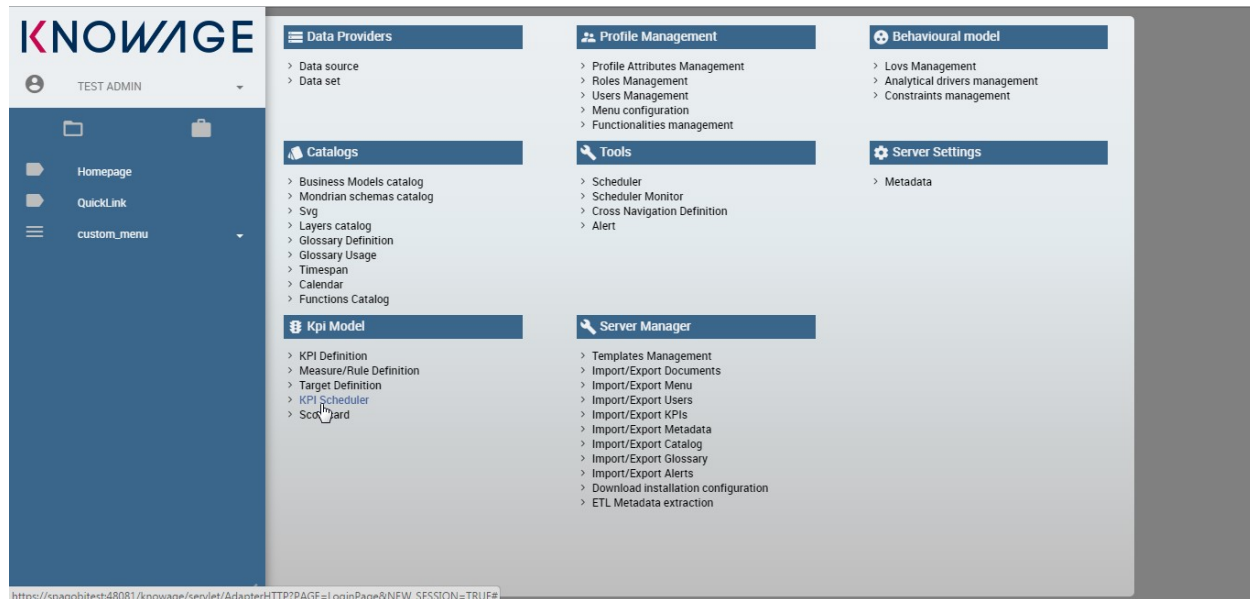


Fig. 5.500: KPI Scheduler menu item.

As for the other interfaces it is enough to click on the “Plus” icon to create a new scheduling. The new scheduling window presents several tabs.

- **KPI:** it is possible to associate one or more KPI to the scheduling clicking on “Add KPI Association”.

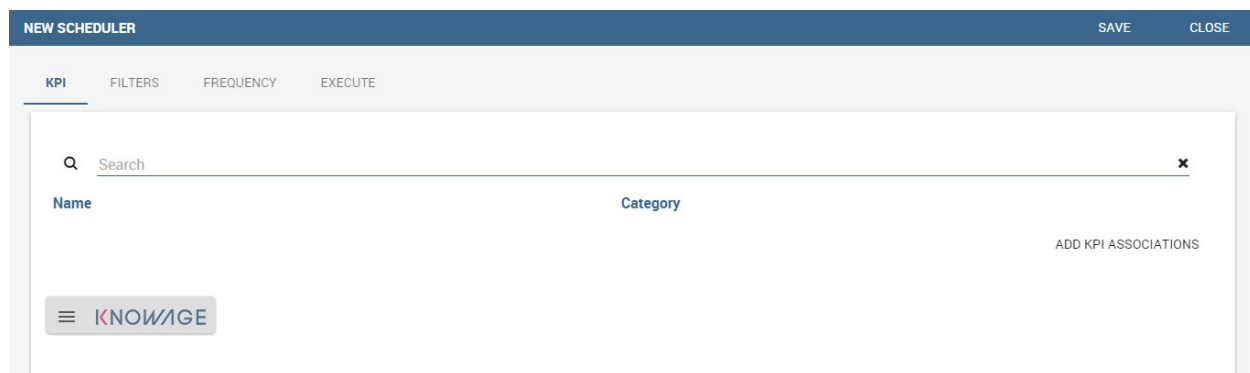


Fig. 5.501: KPI tab window.

- **Filters:** here you assign values to the filters (if configured) associated to the scheduling. Note that it is possible 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.
- **Frequency:** here is the place where the scheduling time interval (start and end date) can be set together with its frequency.
- **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

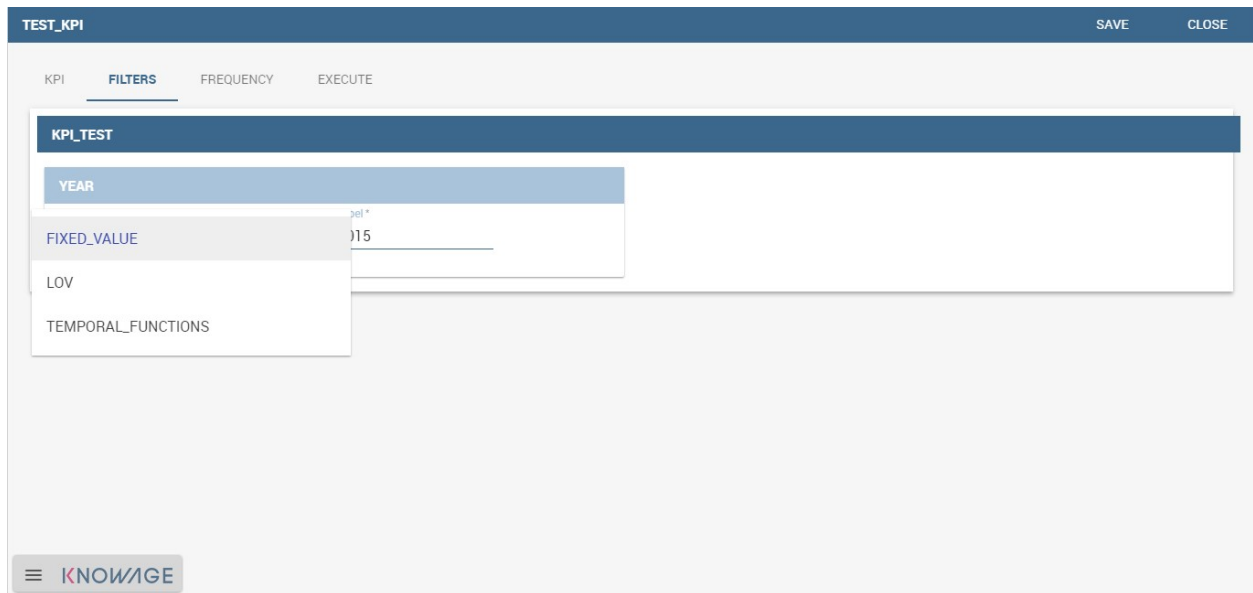


Fig. 5.502: Filters options.

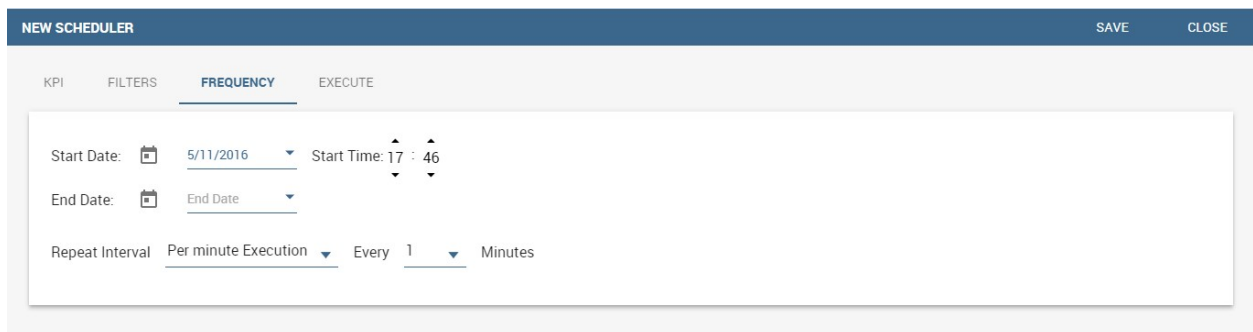


Fig. 5.503: Frequency tab window.

and all error log text files are available right beneath. While selecting **Delete and insert** the previous data are deleted.

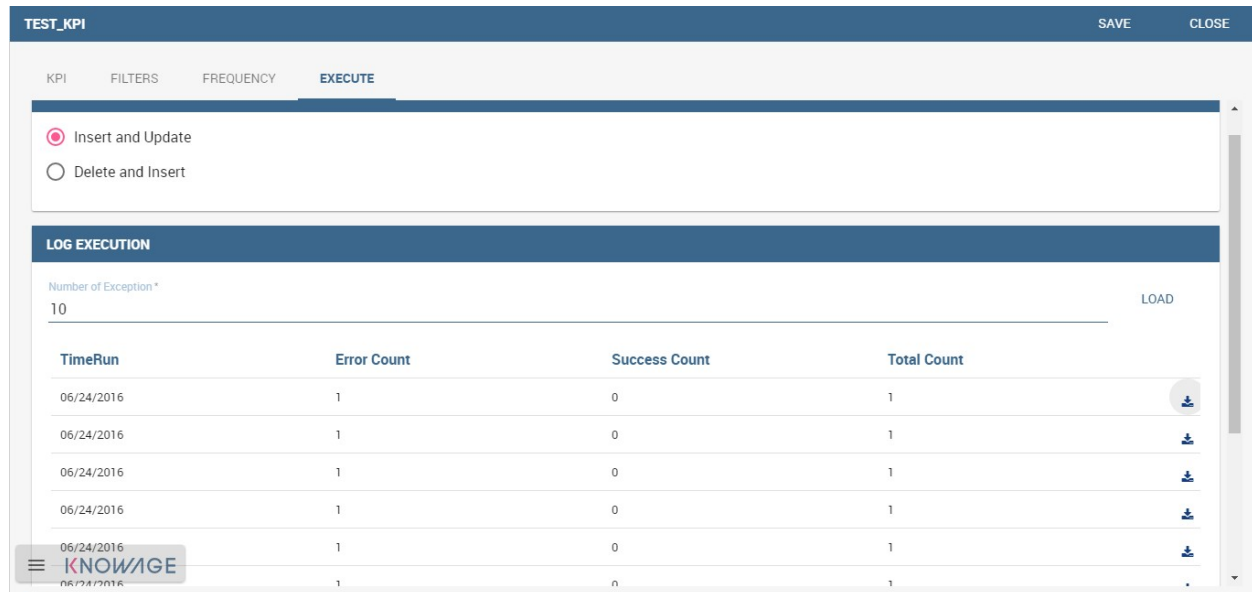


Fig. 5.504: Execute tab window.

In Figure below we sum up the example case we have referred to since now.

Once the schedulation is completed click on the “Save” button. Remember to give a name to the schedulation as in the following figure.

5.18.2 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 below). Then we save.

Click on the **Template build** icon to develop the template. Here you can choose between KPI and Scorecard (refer to Scorecard Chapter for details on the Scorecard option). In the KPI case it is possible to choose between the two following type of document.

- **List:** with this option it is possible to add several KPI that will be shown in the same page with a default visualization.
- **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 or KPI Card; 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 in figure below you can 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 below). Note that the “View as” field is were you can decide if the widget will be a Speedometer or a KPI Card.

Moreover, you can set the other properties of the KPI document using the **Options** and the **Style** areas (Figure below).

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 (following figure) the user is invited to indicate the level of aggregation choosing

NEW SCHEDULER
SAVE
CLOSE

KPI
FILTERS
FREQUENCY
EXECUTE

Q
Search
x

Name	Category
Income	INCOME

NEW SCHEDULER
SAVE
CLOSE

KPI
FILTERS
FREQUENCY
EXECUTE

Start Date:
5/11/2016
Start Time:
18 : 1

End Date:
5/11/2016
End Time:
18 : 4

Repeat Interval
Per minute Execution
Every
1
Minutes

NEW SCHEDULER
SAVE
CLOSE

KPI
FILTERS
FREQUENCY
EXECUTE

EXECUTION TYPE

☒ Insert and Update
☐ Delete and Insert

Fig. 5.505: Overview of the KPI case.

INCOME
SAVE
CLOSE

KPI
FILTERS
FREQUENCY
EXECUTE

EXECUTION TYPE

☒ Insert and Update
☐ Delete and Insert

SAVE SCHEDULER
x

Name
Income
6/40

SAVE ✓

Fig. 5.506: Creation of a KPI Document.

DOCUMENT DETAILS

Label: *

Name: *

Description:

Type:

Engine:

State:

Community:

Refresh seconds:

Criptable: ☐ True ☒ False

Visible: ☒ True ☐ False

Visibility restrictions:

Preview file: Nessun file selezionato

Locked by user: ☐ True ☒ False

Template: Nessun file selezionato

Template build:

Show document templates

Functionalities Tree

- Functionalities
 - TEST
 - TMP
 - DEMO
 - MISC
 - AUDIT
 - Technical Features examples
- Personal Folders

Fig. 5.507: Overview of the KPI case.

KPI DOCUMENT DESIGNER SAVE

☒ KPI ☐ Scorecard

TYPE OF DOCUMENT

☐ List ☒ Widget

KPI LIST

Q Search ×

Name	Category	View as	Range Min Value	Range Max Value	Prefix / Suffix label	Show label as
kpi_test	TEST_KPI	Speedometer	0	300	Prefix / Suffix label	<input checked="" type="checkbox"/> Suffix ✕

ADD KPI ASSOCIATIONS

KNOWAGE

Fig. 5.508: Setting the KPI associations using the dedicated area.



Fig. 5.509: Areas of the Template Build for KPI.

among “day”, “week”, “month”, “quarter”, “year”.

Finally in the “Style” area the user can customize the size of the widget, the font, the color and size of texts.

Then save and run the document. Examples are shown in the last three figures below.

In case the document contains KPIs that involves grouping functions upon some attributes, it is possible to filter data returned on those attributes. To easily retrieve the attributes on which measures are grouped, it is sufficient to check the fields listed in the “Cardinality” tab of the KPI definition. WE recall it in the picture below.

Then to use them to filter the document, first add the proper analytical drivers. Refer to Section 5.4 to get more information about how to associate an analytical driver to a document (and therefore to a KPI document). Then it is mandatory that the URL of the analytical driver *must* coincide with the *attribute aliases* on which you have defined the grouping.

5.19 Scorecard

The **Scorecard** feature, available in Knowage suite as highlighted in the following figure, allows to supervise different KPIs at the same time. This option gives an exclusive complete overview of the KPIs situation when the user is not interested in a single threshold check. This tool is in fact useful when concern is addressed to monitoring the overcoming of two or more critical KPI values.

5.19.1 Scorecard development

A scorecard is structured in hierarchical levels. Shortly, there is a first level called **Perspective** composed of KPIs grouped on targets. Otherwise, **Targets** are assigned a threshold depending on the KPIs they are composed of. In the following we will describe in detail a scorecard configuration. When clicking on the Scorecard menu item the window of figure below opens. Here the implemented scorecards are listed and can be explored once selected and edited.

The “Plus” icon available at the right top corner of the page opens a new window where to set a new scorecard, as shown below. Assign a name and click on **Add perspective** (Figure below).

A perspective allows you to organise the monitoring over targets.

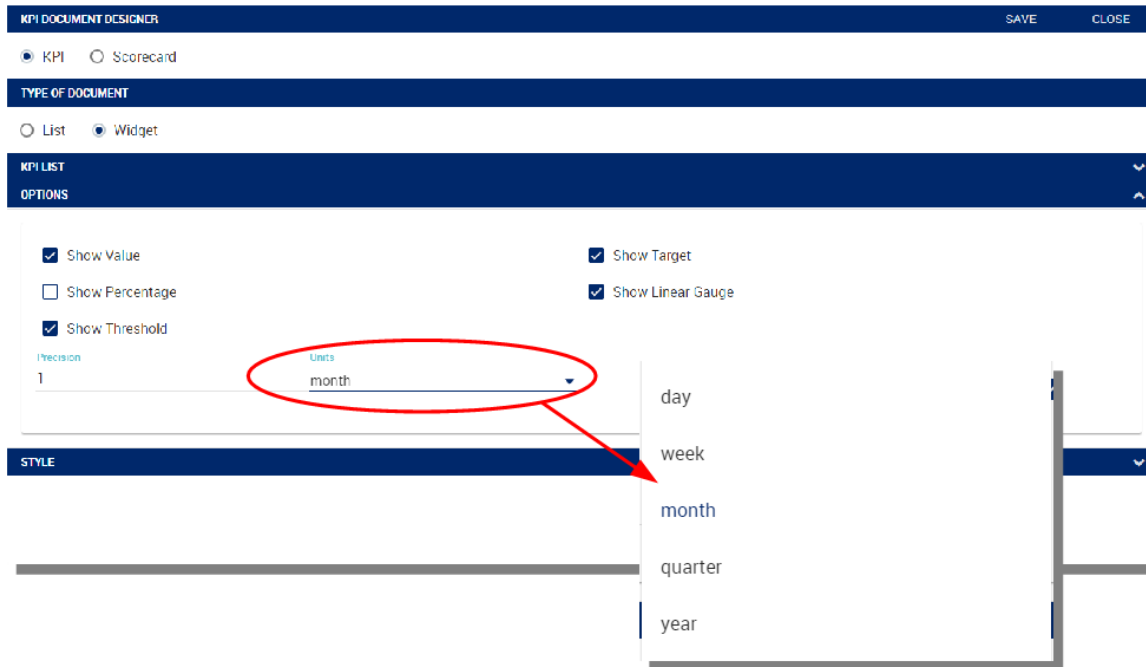


Fig. 5.510: Choose the time granularity.



Fig. 5.511: Style settings.



Fig. 5.512: Cardinality settings example.

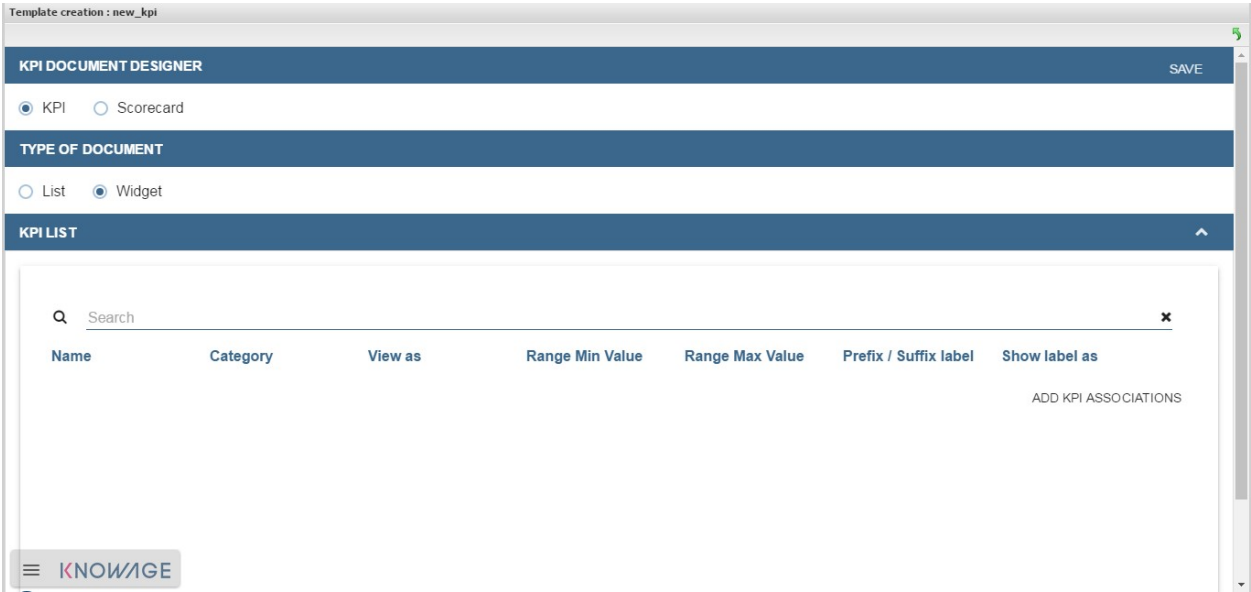


Fig. 5.513: KPI association.

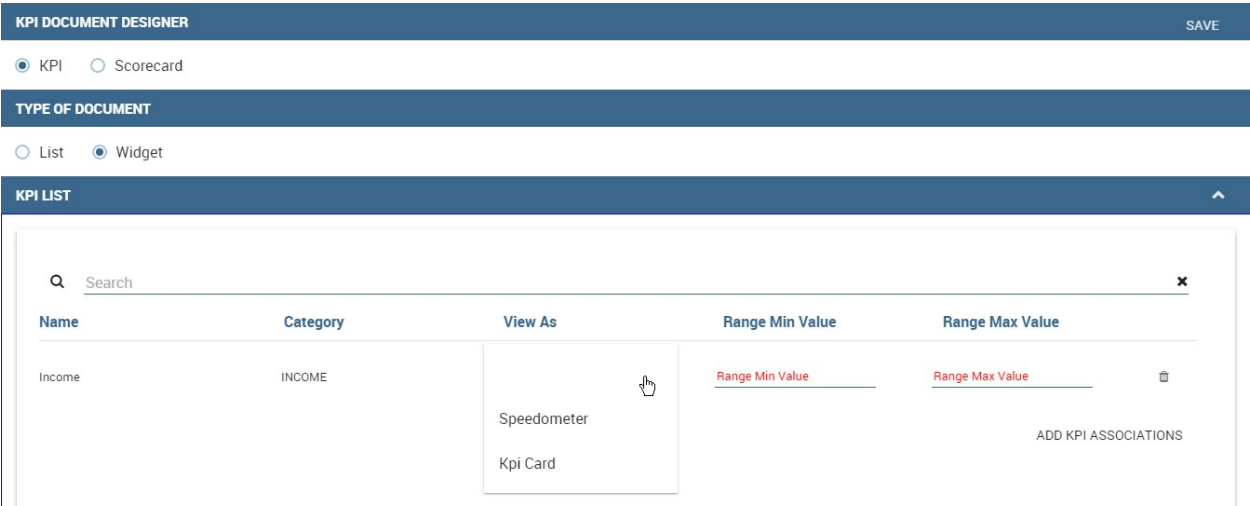


Fig. 5.514: Widget document.

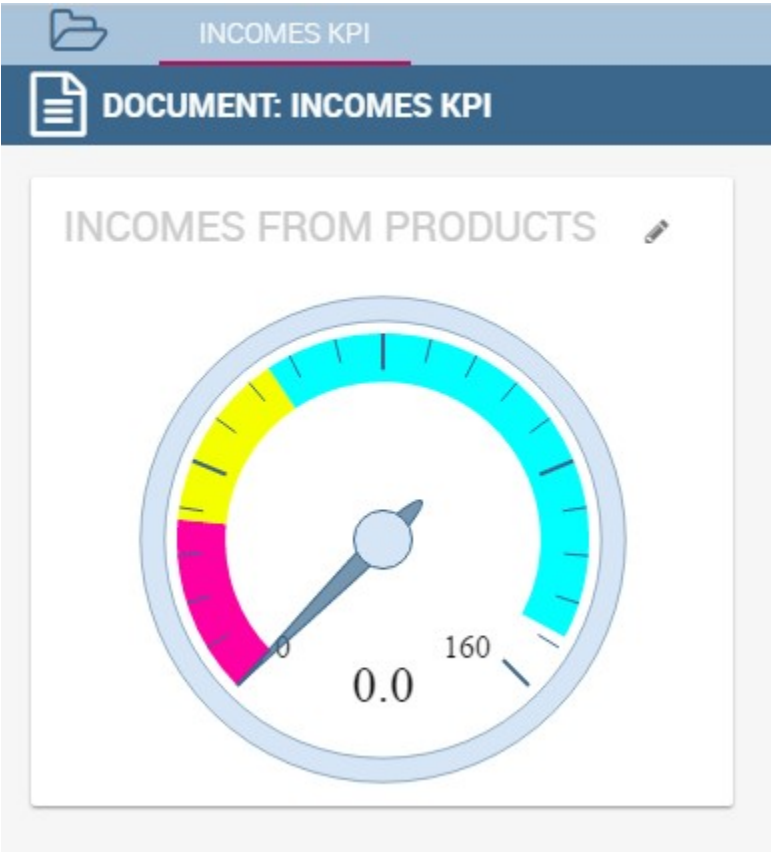


Fig. 5.515: KPI Speedometer.

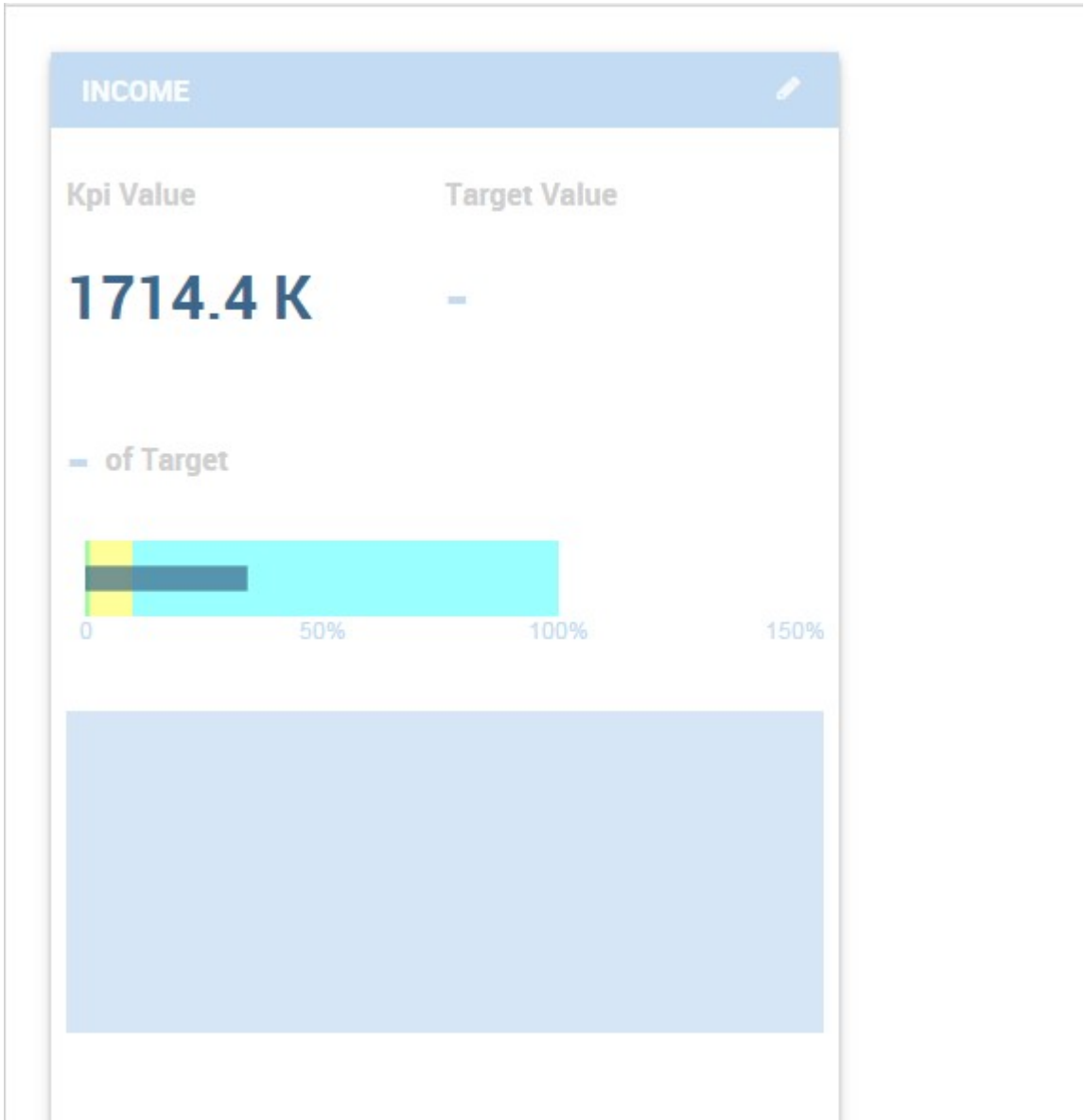


Fig. 5.516: KPI Card.

Severity	Name	Value	Thresholds e target	Trend
<div></div> LOW	Income	1714387.4	<div></div>	<div></div>

Fig. 5.517: KPI List.

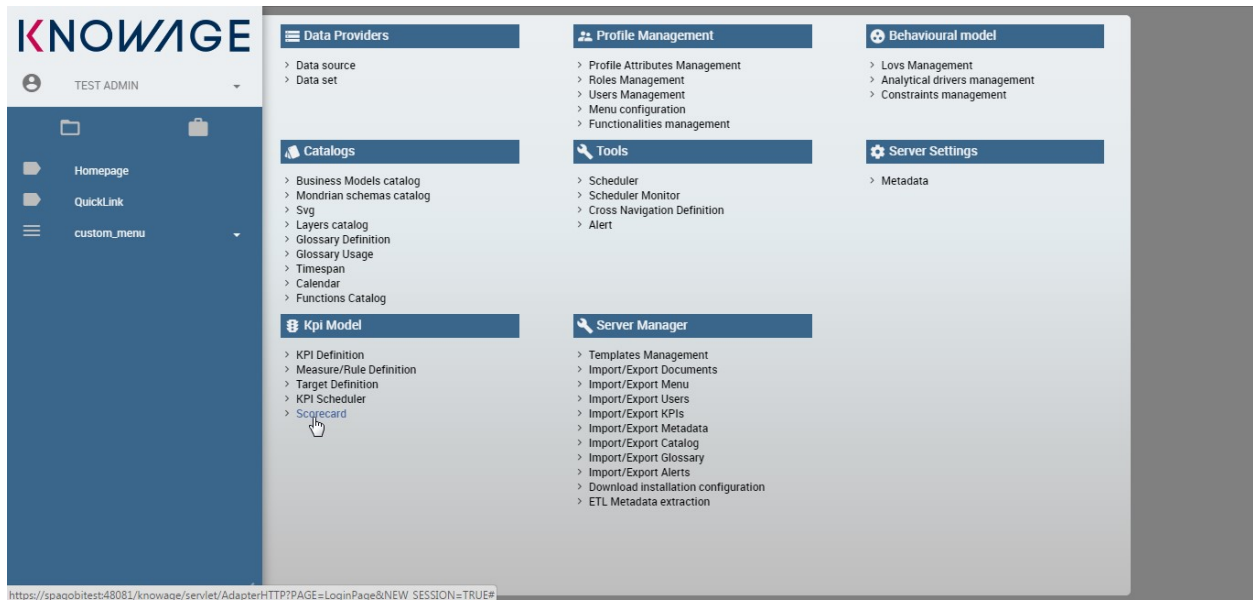


Fig. 5.518: Scorecard from the contextual menu.

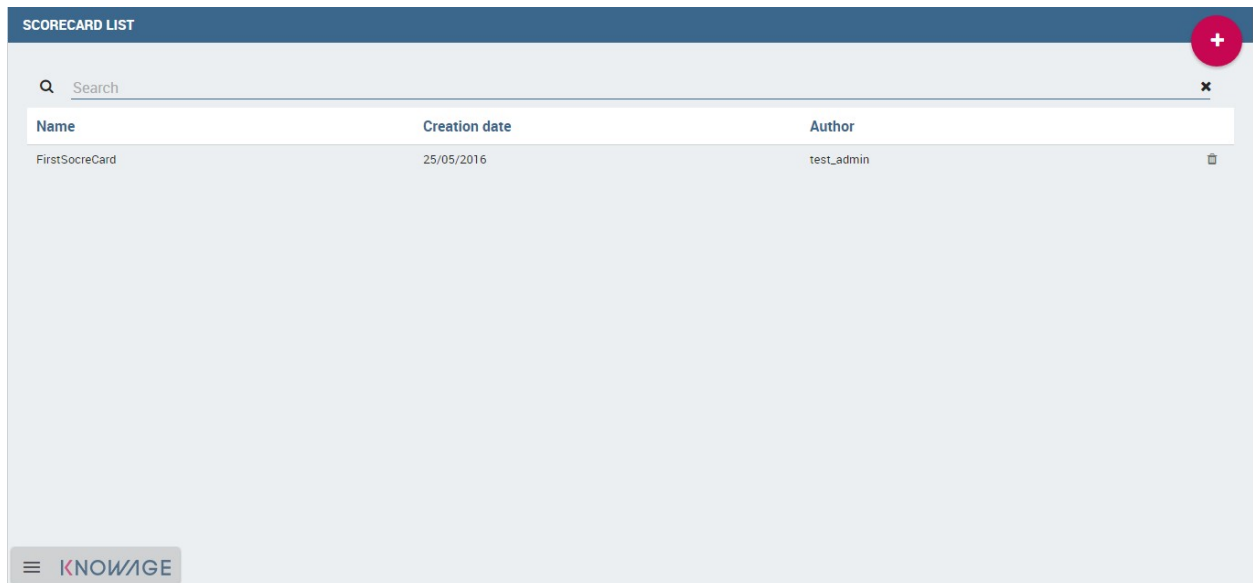


Fig. 5.519: Scorecard window.

The screenshot shows a web form titled "FIRSTSCORECARD" with "SAVE" and "CLOSE" buttons in the top right. Below the title bar, there is a "Scorecard" section with a "Scorecard Name" field containing the text "FirstScoreCard". Below this is a "PERSPECTIVE LIST" table with an "ADD PERSPECTIVE" button in the top right corner. The table is currently empty. At the bottom left, there is a "KNOWAGE" logo.

Fig. 5.520: Defining a new scorecard.

The screenshot shows a web form titled "NEW SCORECARD" with "SAVE" and "CLOSE" buttons in the top right. Below the title bar, there is a "Scorecard Name" field. Below this is a "PERSPECTIVE LIST" table with an "ADD PERSPECTIVE" button in the top right corner. A dialog box is open over the table, titled "New Perspective". It contains an "Evaluation Criterion" field with the text "Policy 'Majority'" and an "ADD TARGET" button. A mouse cursor is pointing at the "ADD TARGET" button. At the bottom left, there is a "KNOWAGE" logo.

Fig. 5.521: Add perspective to the scorecard.

An example is given in the following figure.

Fig. 5.522: Perspective list example.

In fact, each perspective manages one or more targets accordingly to the user's requirements. A target consists of one or more KPIs and it is assigned a threshold color according to the chosen **Evaluation criterion**. In fact, if one selects:

- **Policy "Majority"** the target gets the threshold of the KPI threshold that numerically exceeds the others,
- **Policy "Majority with Priority"** the target gets the threshold of a specific KPI,
- **Policy "Priority"** the target gets the majority threshold of the KPIs in case the primary stated KPI returns the lower threshold, namely the "green" one, while it gets the threshold of a primary stated KPI in case the latter returns the other thresholds, namely the "yellow" or the "red" one.

Warning: Thresholds of selected KPIs must have the right colors

Note that the scorecard shows the right colors accordingly with the selected policy only if the KPIs which compose the targets have **no filters** and **standard colors** (see Section 7.1, Step 2 for definitions) to highlight the threshold.

Warning: "Standard" colors for thresholds

When the targets contain parametric KPIs the target/perspective evaluation cannot be completed for value absence. Therefore the warning lights turn grey. The right visualization of the scorecard must be implemented through a scorecard document. Check Section 8.2 to have more details on how to develop a scorecard document.

An example is showed below.

The same choice is available at the perspective level (refer to next figure), that is:

- **Policy "Majority"** the perspective gets the threshold of the target threshold that numerically exceeds the others,
- **Policy "Majority with Priority"** the perspective gets the threshold of a specific target,

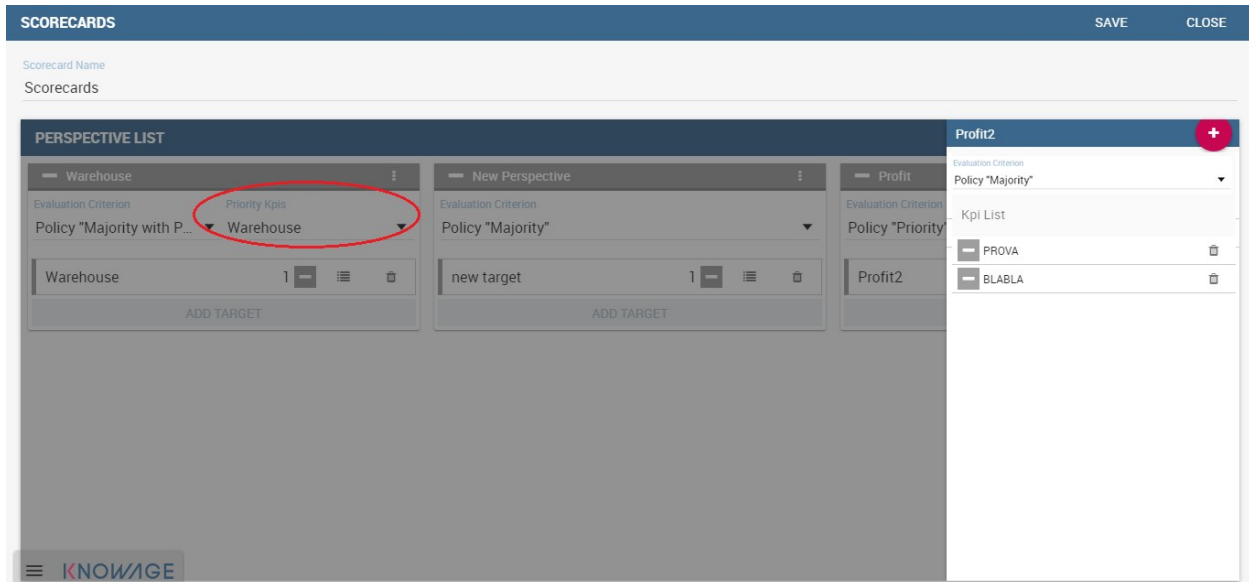


Fig. 5.523: Select the KPI with priority.

- **Policy “Priority”** the perspective gets the majority threshold of the targets in case the primary stated target returns the lower threshold, namely the “green” one, while it gets the threshold of a primary stated target in case the latter returns the other thresholds, namely the “yellow” or the “red” one.

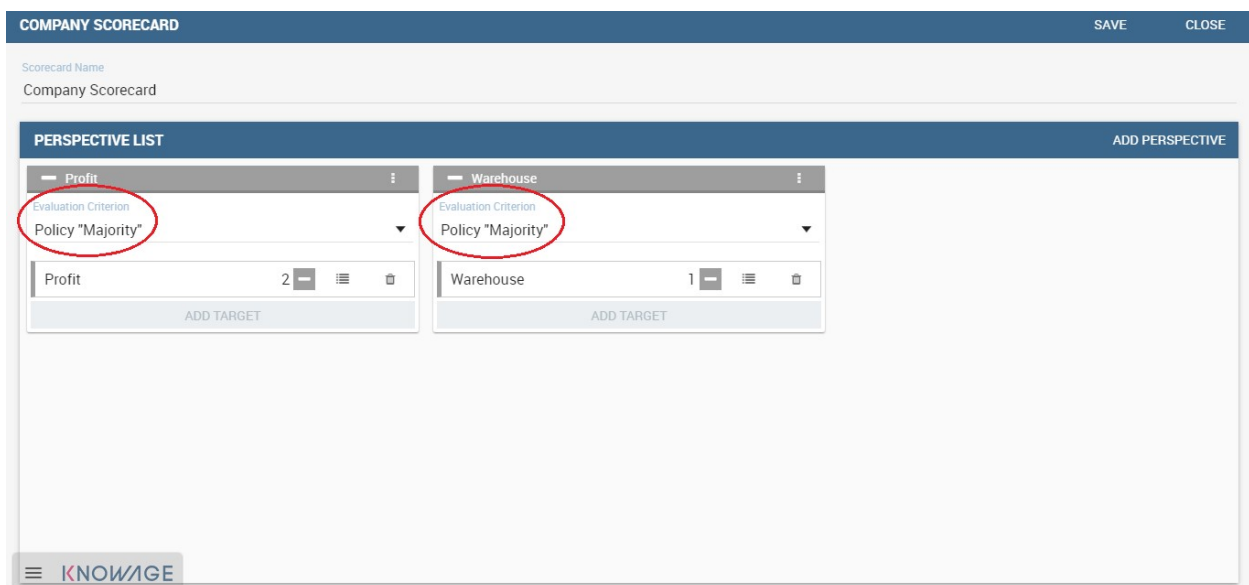


Fig. 5.524: Perspective policy.

Remember to save once perspectives and targets have been set.

5.19.2 Creation of a Scorecard document

Once saved it is possible to develop a scorecard document which can be easily consulted by (authorized) end users. To create a scorecard document click on the “Plus” icon available in the document browser and then “Generic document”

from the panel as shown below. Here fill

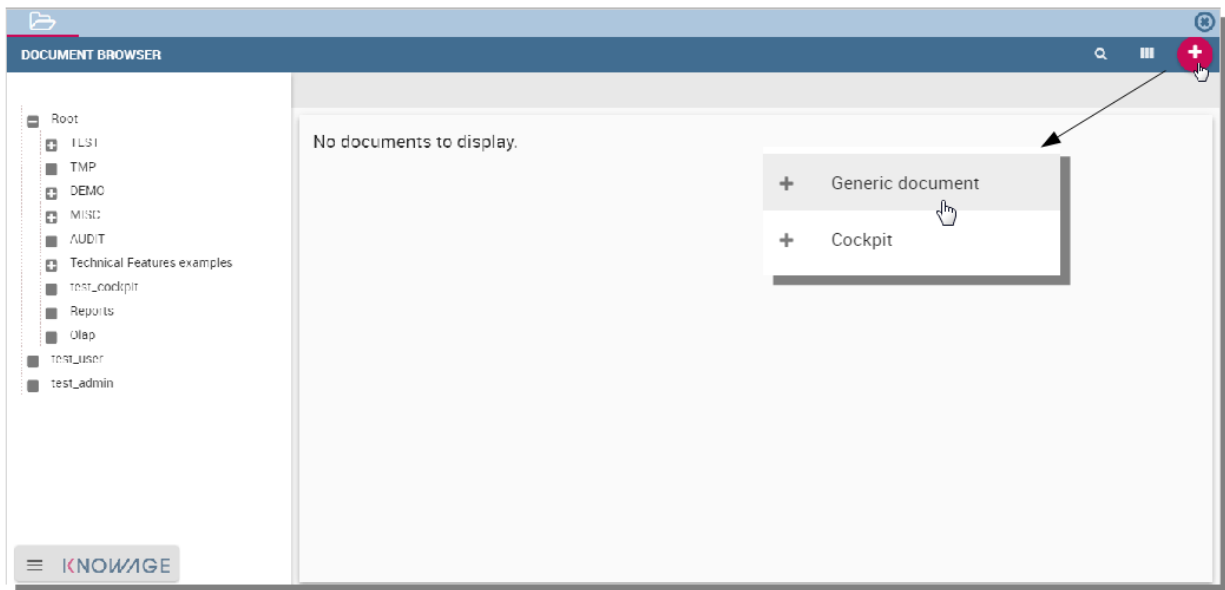


Fig. 5.525: Create a generic document from document browser.

in mandatory fields (marked with an asterisk) and select the KPI type and **KPI engine**. Then open the “Template build”. Here select the “Scorecard” option as in figure below9 and consequently Creation of a Scorecard document choose an existing scorecard from the list. Make the desired customizations and save.

Figure below gives an example of the scorecard document interface. The arrows point out the perspectives’ achievement of the goals or, on the contrary, the missing of the targets. As well the achievement/failure of the single targets is pinpointed by the arrow signals close to each target.

Note that it is possible to check the policy used for each perspective. In fact, by clicking on one of them a wizard opens showing the policy adopted and the goal got by ach KPI.

5.20 Alert

The **Alert** functionality available under the **Tool** section of Knowage main menu, as shown in Figure below, allows you to implement a control over events. Technically it consists of monitoring the possible overcoming of fixed thresholds and the consequent signal of anomalies by means of an email or by launching an ETL process. In next sections we will see in details how to configure the alarms using the Alert feature.

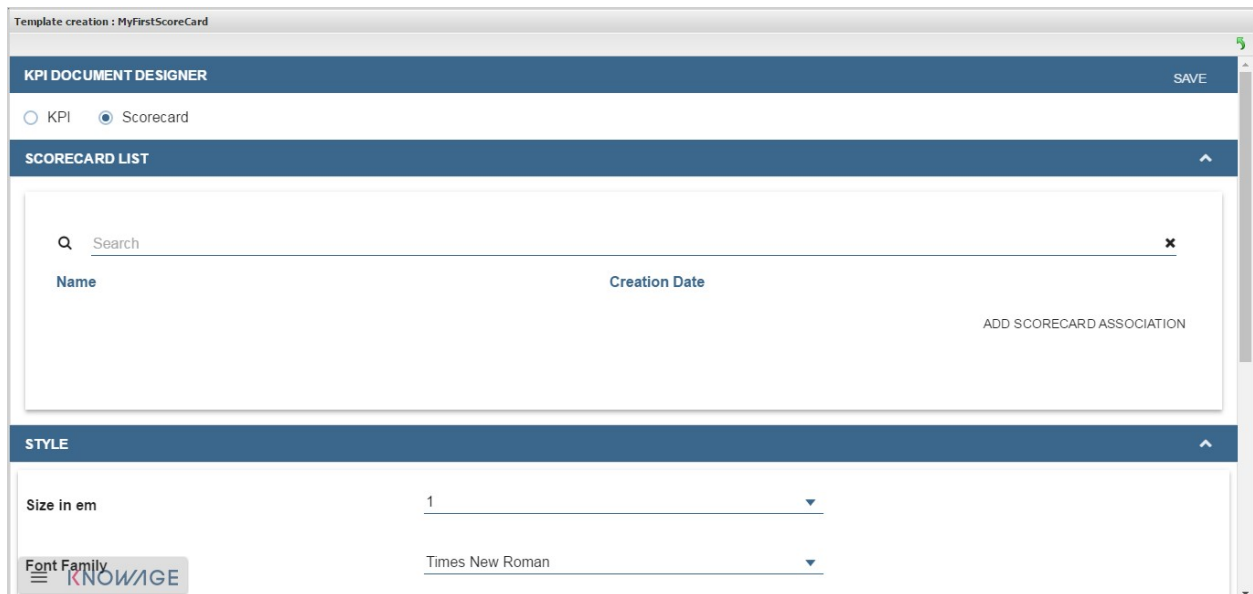


Fig. 5.526: Template creation window.

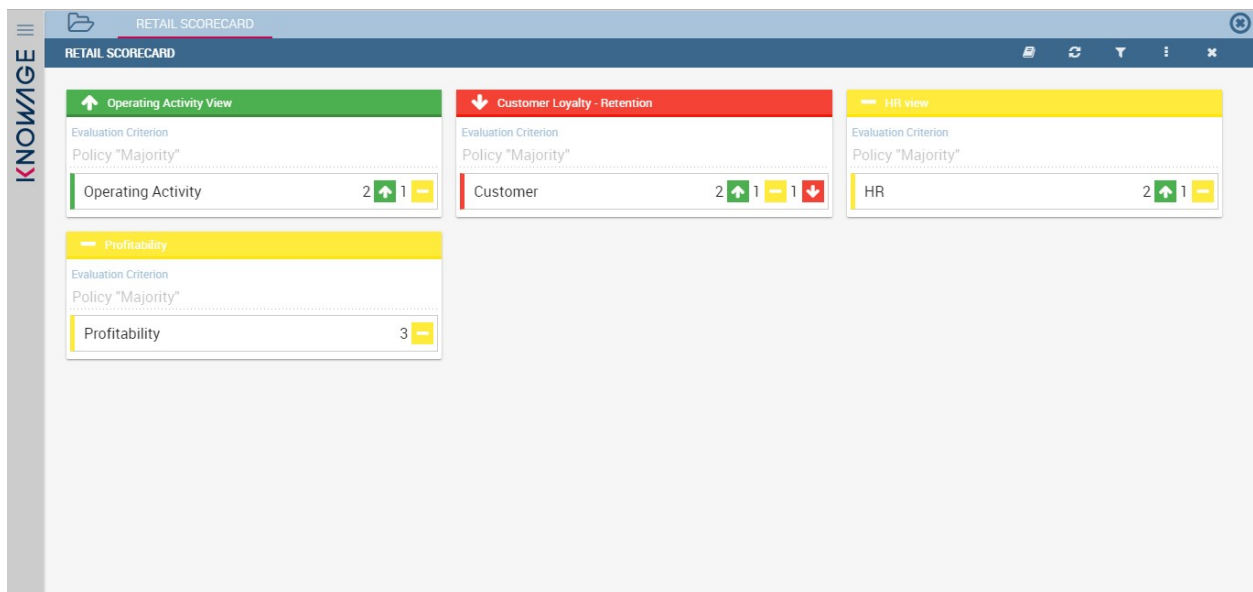


Fig. 5.527: Scorecard document interface.

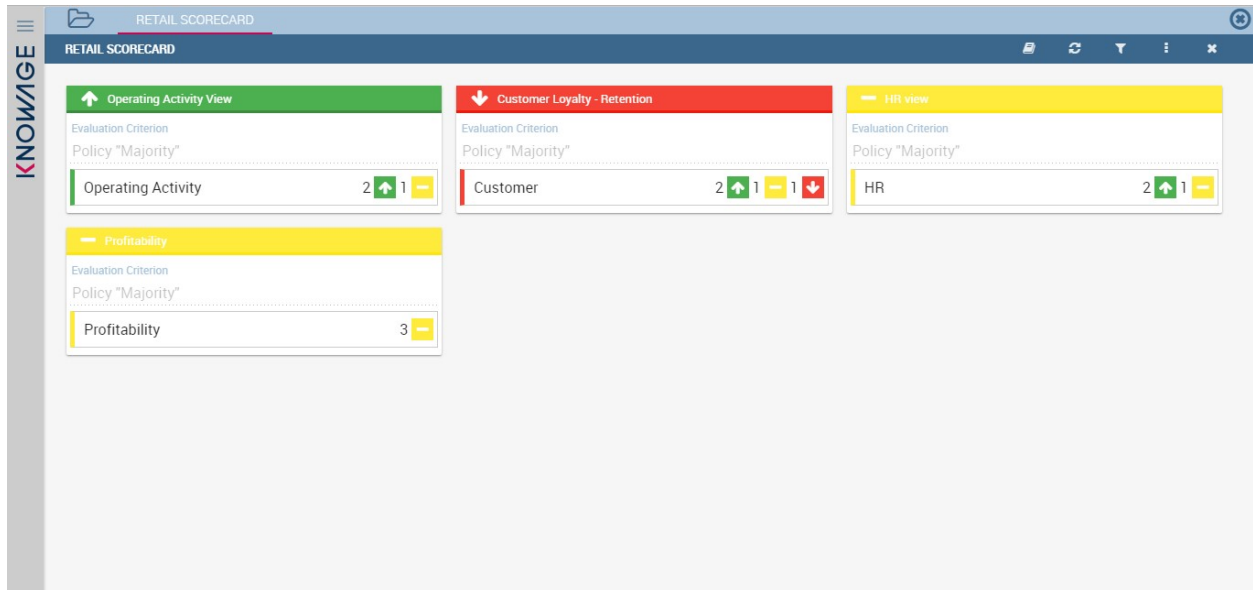


Fig. 5.528: Scorecard document interface.

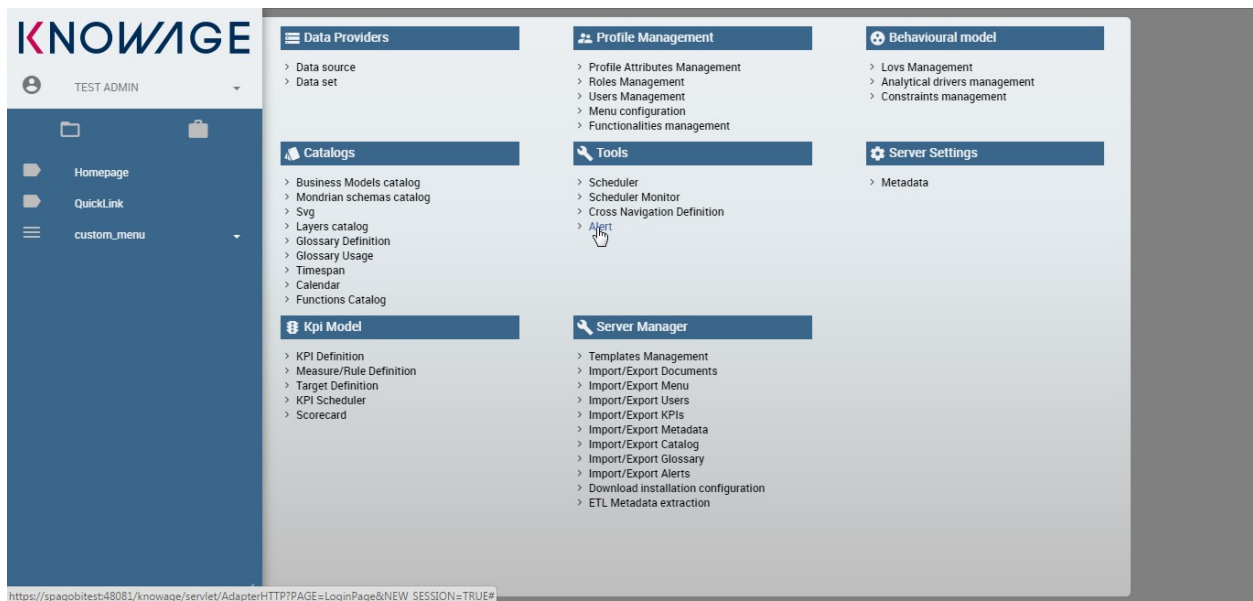


Fig. 5.529: Alert functionality from contextual menu.

5.20.1 Alert implementation

The Alert definition window (refer to the figure above) displays the list of existing alerts and a search box at the top of the page to facilitate to browse over the items. In the right top corner it is available the “Plus” icon to configure a new alert.

Clicking on the “Plus” icon the page in the following figure opens.

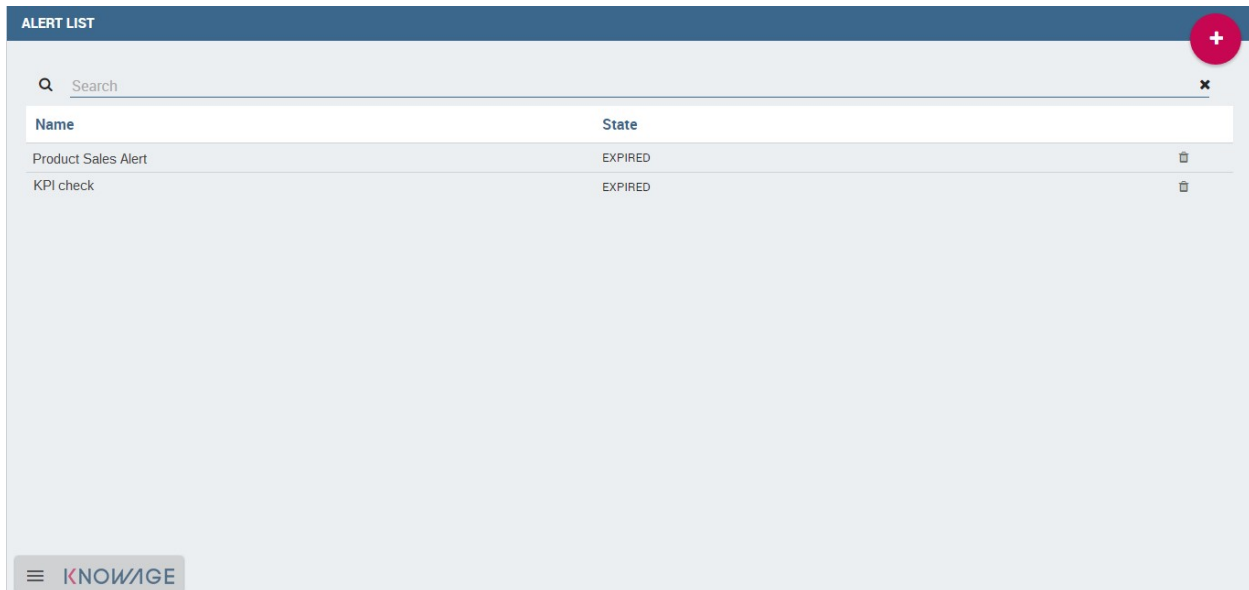


Fig. 5.530: Alert definition window.

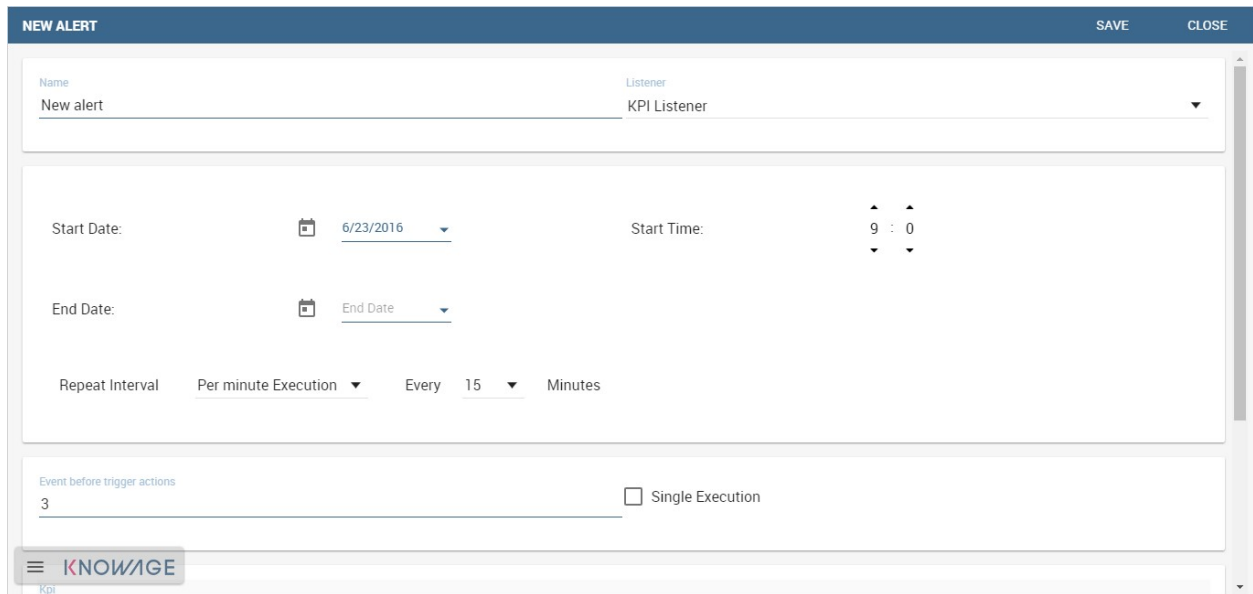
Give a name and select the **KPI listener** from the combobox. Then indicate the start date, the start time and eventually the Alert implementation end date. Specify the time interval and how many times it must be repeated. At this point choose between **Single Execution** and **Event before trigger actions**: in the first case the alert must signal the anomaly just once its occurrence while in the other case it must send the alarms when the irregularity occurs as many times as specified. Note indeed that the box is editable and it must contain a number which indicates the number of irregularities that has to be detected before the alert starts.

At the bottom of the page associate the KPI of interest and select the action by clicking on the **Add action** button. Here you can choose between **Send mail** or **Execute a document**.

If the “Send mail” is chosen, the user is asked to insert the thresholds that should be monitored: in case the latter are overcome the functionality sends the email to the indicated subjects. Remember to insert all mandatory fields (name, thresholds and subjects of the mail) and, in particular, to select the user to which the email is sent. Note that in the mail subjects box you can type both mail addresses or usernames (for example “user_admin”), helped by the automatic insertion text as shown in the next figure. In fact, Knowage server will pick out the mail address from the profile attribute associated to that user. Therefore, we recommend to set the profile attribute previously. In particular, remember that the profile attribute *must* be Alert implementation called “email”.

Furthermore, note that the mail body can contain a plain text message or a table showing the registered KPI values which activated the alert. To enable table visualization option you must insert the placeholder `*TABLE_VALUE*` in the mail body as shown in Figure below.

If the “Execute a document” option is selected, the alert will launch an ETL process when the chosen thresholds are overcome. Namely, the functionality will launch a process to execute batch actions as the writing of proper tables of a DWH, creation of a file, the call to a log service. Note that the ETL process must exist as a document in Knowage server. An example is given in the following figure.



NEW ALERT SAVE CLOSE

Name: Listener:

Start Date: Start Time:

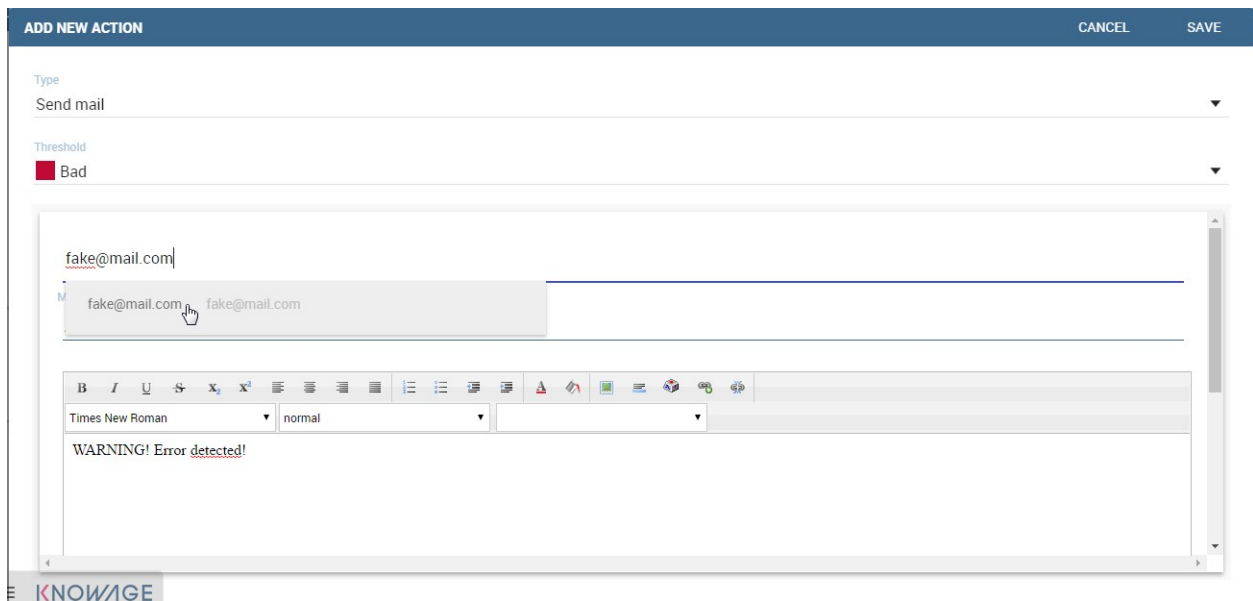
End Date:

Repeat Interval: Every Minutes

Event before trigger actions: ☐ Single Execution

KNOWAGE

Fig. 5.531: Alert settings.



ADD NEW ACTION CANCEL SAVE

Type:

Threshold:

B I U

WARNING! Error detected!

KNOWAGE

Fig. 5.532: Send email configuration.

Fig. 5.533: Send email containing a table with detected values.

Save both the action and the alert configuration settings to store your alert. Remember that it is possible to schedule more than one monitoring over the same alert definition.

5.21 News management

The **news management** is an helpful functionality to keep the end user constantly informed on any changements or updates on the Knowage platform itself. In this section we will describe the main steps to insert a latest piece of news.

5.21.1 How to publish news

To put news in place is mandatory to have administration priviledges. An admin user can insert the news entering the “news management” functionality from Knowage main menu, as the following picture highlights:

Consequently, the user is redirected to the News management list. The page is divided into two areas. On the left side, all previously inserted news are listed. When clicking on one of them, its details are shown on the right side.

To edit the latest news instead, the user must click on the red plus icon button available at the center of the page. When clicking on the icon, Knowage server opens the interface shown in the following image:

Here the user is asked to:

- insert the news title,
- insert the news description,
- insert the text of the news,
- specify the expiration date, i.e. the date until the news can be visible from the end user,
- to check/unckeck a flag to enable/disable the piece of news,
- the list of the roles that have the permission on reading the news.

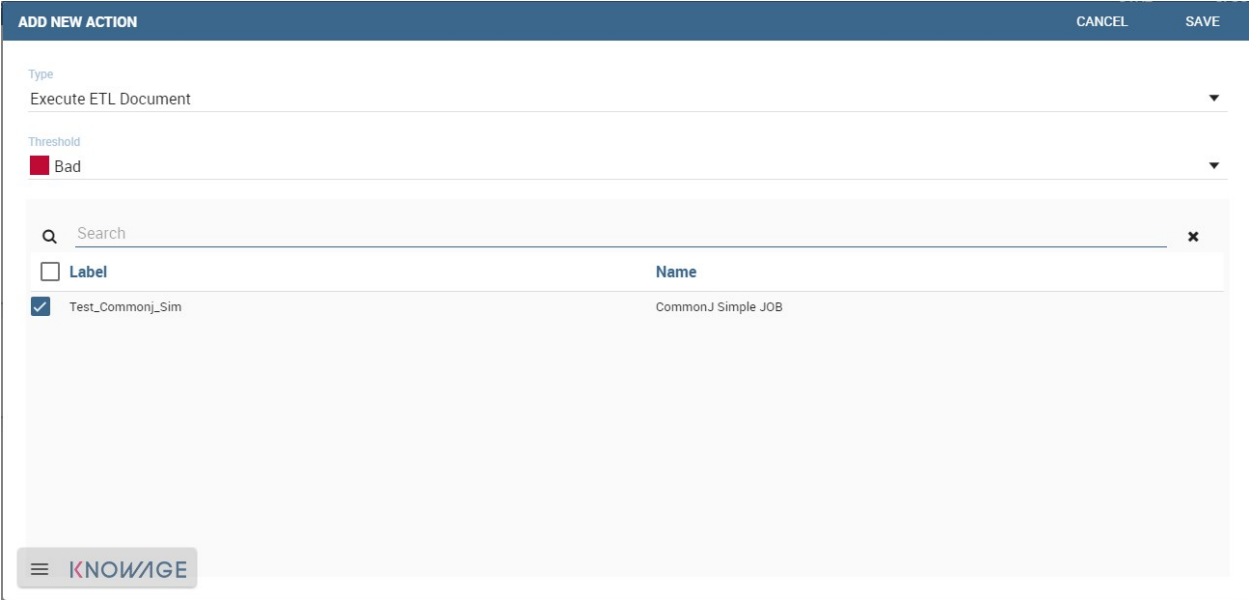


Fig. 5.534: Execute document configuration.

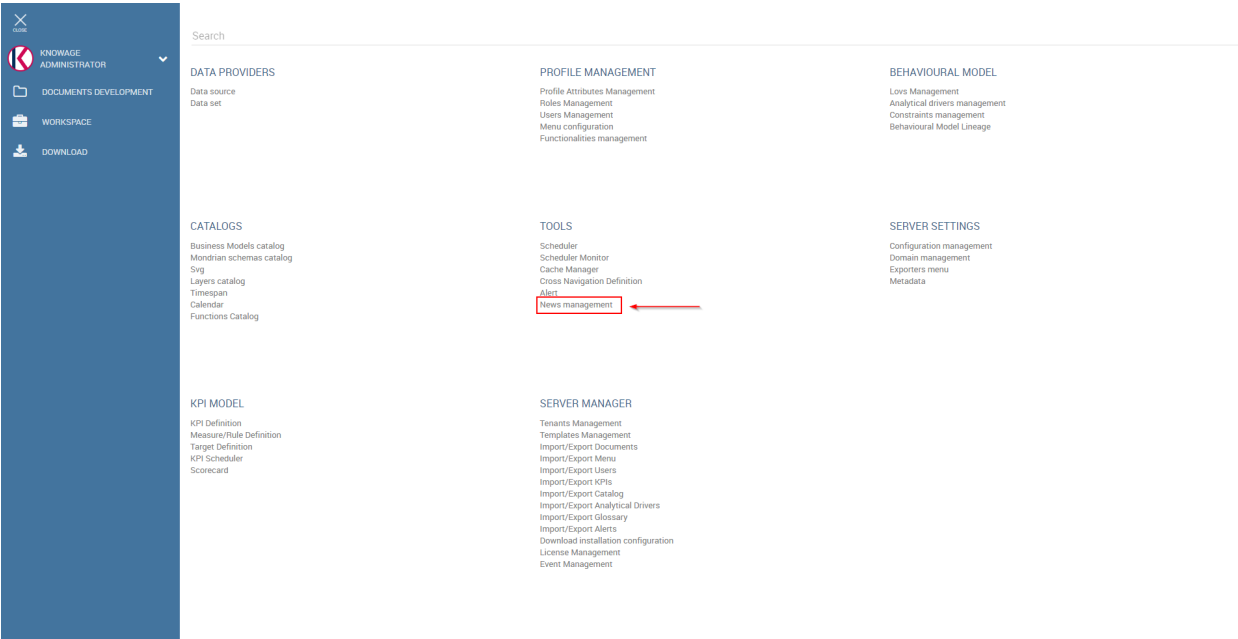


Fig. 5.535: News management menu item.

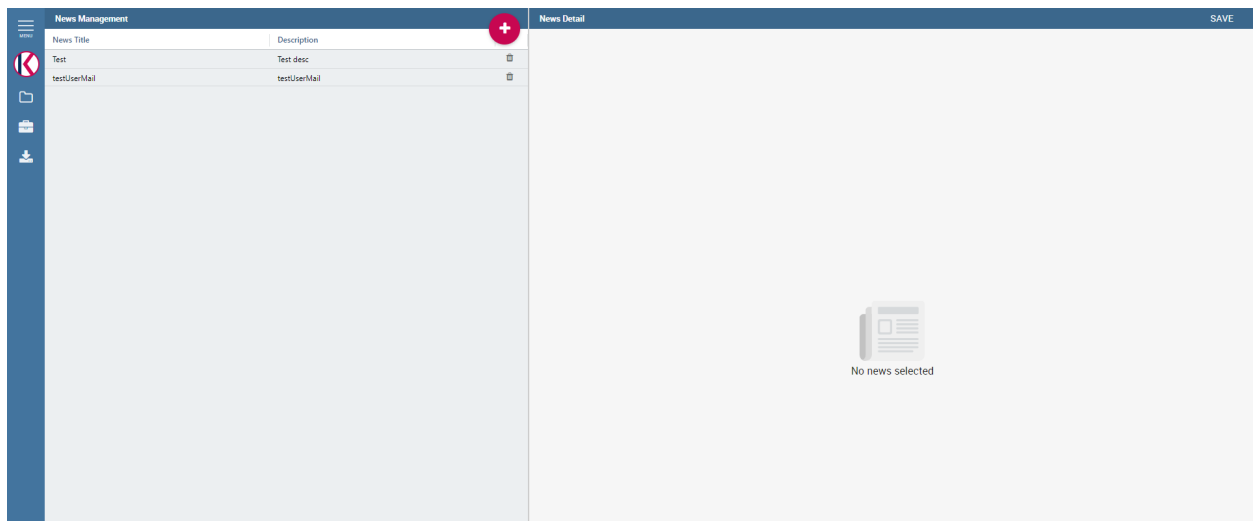


Fig. 5.536: Access news management list.

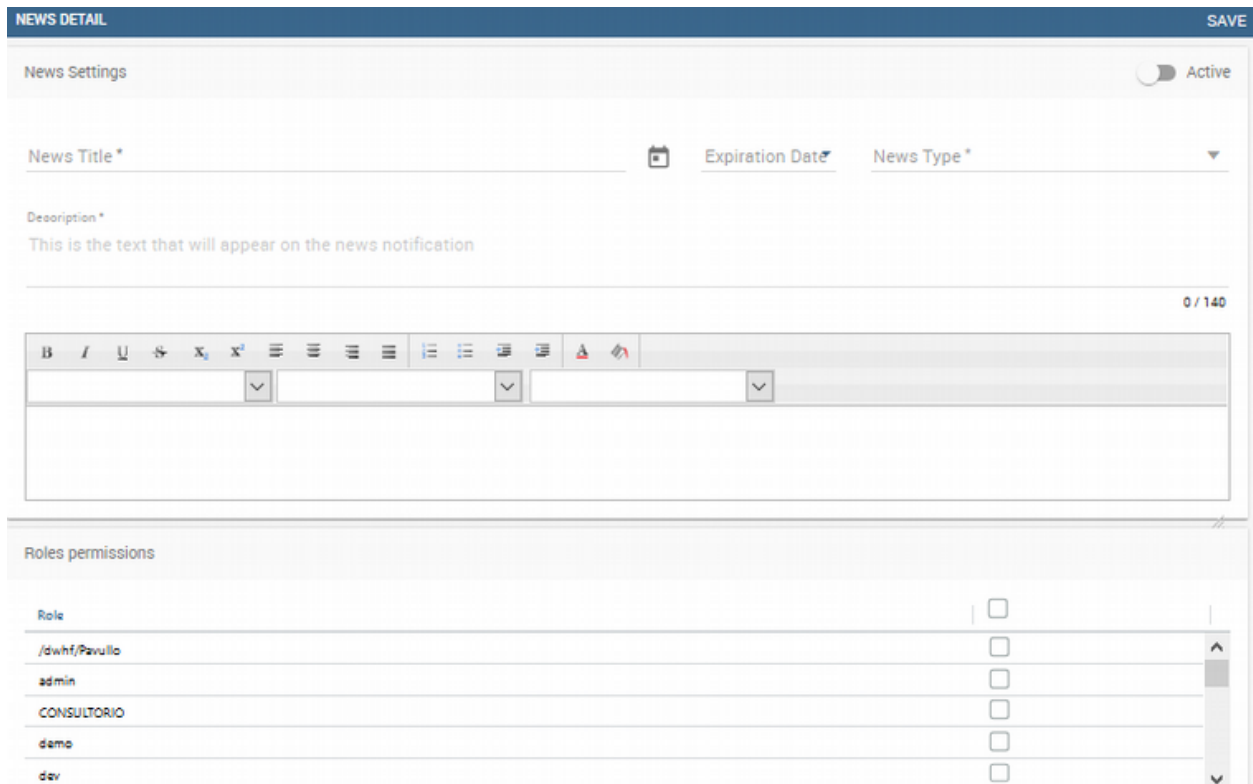


Fig. 5.537: Access news management list.

Once the “news details” form is filled in all its mandatory parts, the user can save the piece of news using the “save” button, available on top right corner of the page.

5.21.2 How the end user can read the news

When some news are added, the **news icon** reports the presence of recent news thanks to a red spot symbol that highlights also the number of latest unread news.

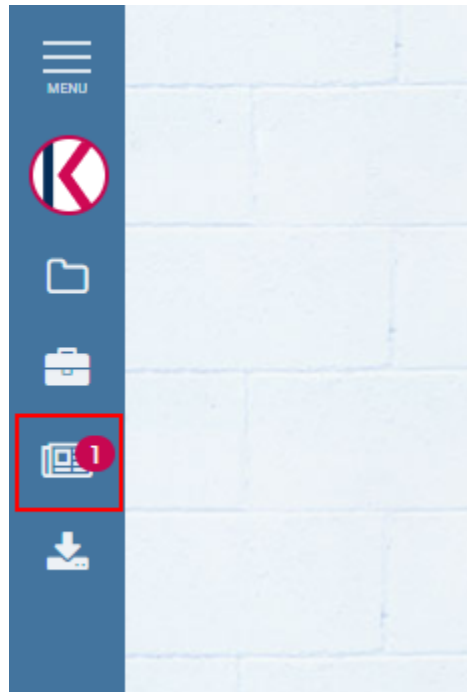


Fig. 5.538: News notification.

The news icon is always available under the Knowage menu panel, while the red icon stays still the user has some unread news. Clicking on the icon, the list of recent unread news appears.

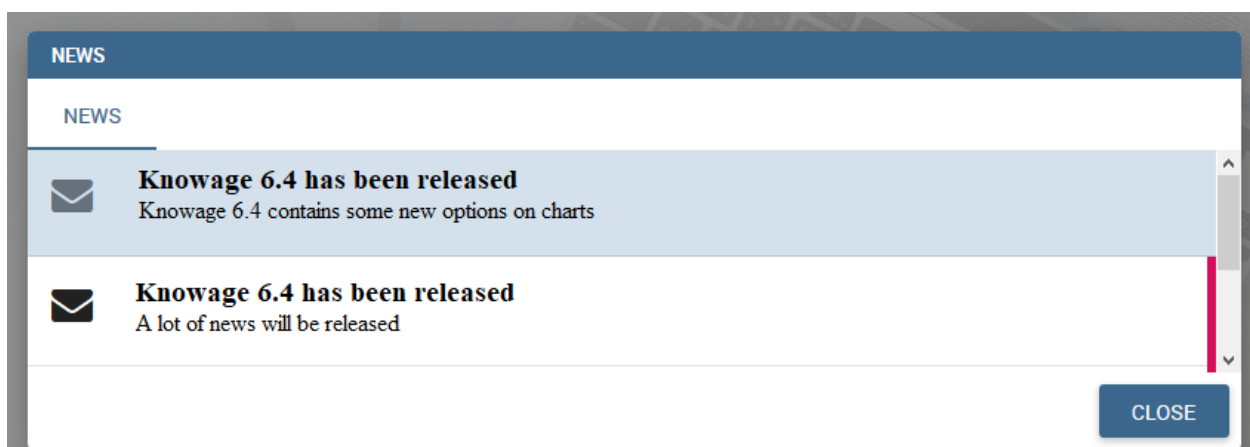


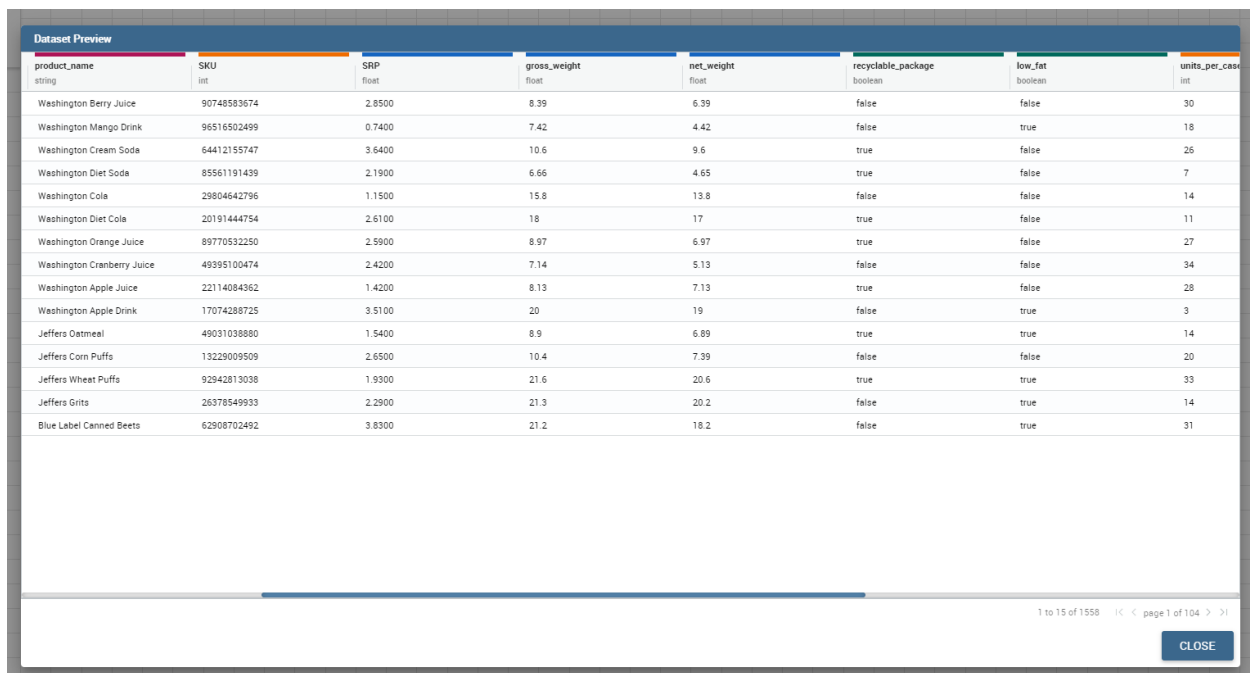
Fig. 5.539: News notification pop up.

The user can access news information, simply selecting the news item itself from the list. Finally the user can mark it as read.

5.22 Dataset Preview

The **dataset preview** is a functionality present in workspace, in cockpit and in BIRT report that will manage to get a preview of the selected dataset and parameter.

5.22.1 Dataset preview structure



product_name	SKU	SRP	gross_weight	net_weight	recyclable_package	low_fat	units_per_case
string	int	float	float	float	boolean	boolean	int
Washington Berry Juice	9074883674	2.8500	8.39	6.39	false	false	30
Washington Mango Drink	95516502499	0.7400	7.42	4.42	false	true	18
Washington Cream Soda	64412155747	3.6400	10.6	9.6	true	false	26
Washington Diet Soda	85561191439	2.1900	6.66	4.65	true	false	7
Washington Cola	29804642796	1.1500	15.8	13.8	false	false	14
Washington Diet Cola	20191444754	2.6100	18	17	true	false	11
Washington Orange Juice	89770532250	2.5900	8.97	6.97	true	false	27
Washington Cranberry Juice	493951100474	2.4200	7.14	5.13	false	false	34
Washington Apple Juice	22114084362	1.4200	8.13	7.13	true	false	28
Washington Apple Drink	17074288725	3.5100	20	19	false	true	3
Jeffers Oatmeal	49031038880	1.5400	8.9	6.89	true	true	14
Jeffers Corn Puffs	13229009509	2.6500	10.4	7.39	false	false	20
Jeffers Wheat Puffs	92942813038	1.9300	21.6	20.6	true	true	33
Jeffers Grits	26378549933	2.2900	21.3	20.2	false	true	14
Blue Label Canned Beets	62908702492	3.8300	21.2	18.2	false	true	31

1 to 15 of 1558 < > page 1 of 104 > >>

CLOSE

Fig. 5.540: Dataset preview example screenshot.

As you can see in the example, in the dataset preview you can see the dataset columns highlighted with different colors depending on the data type. Same data type columns will share the same color.

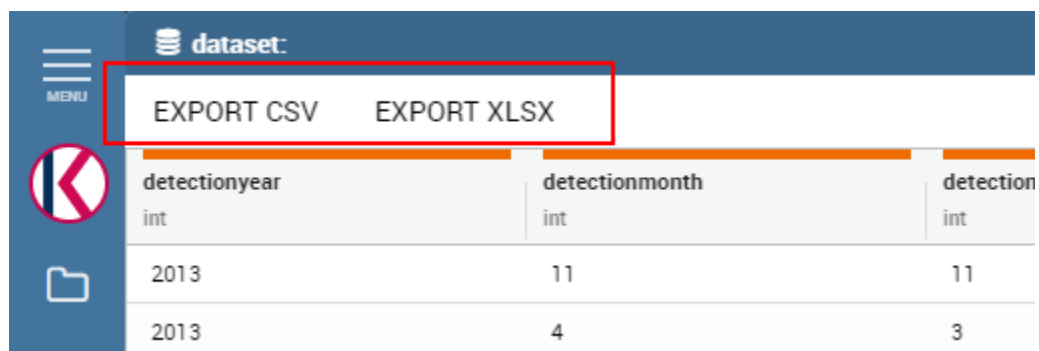
For every column you will be able to read the column name and the column type inside the column header. At the bottom a pagination will appear if needed.

For some specific dataset the export bar will also be available. If the selected dataset supports this functionality a bar will appear at the top of the dialog. Clicking the buttons inside you will be able to download the preview in the selected format.

5.22.2 Dataset preview export

Clicking on the export button, or if the default for that preview is set to download, an exportation process will start in background.

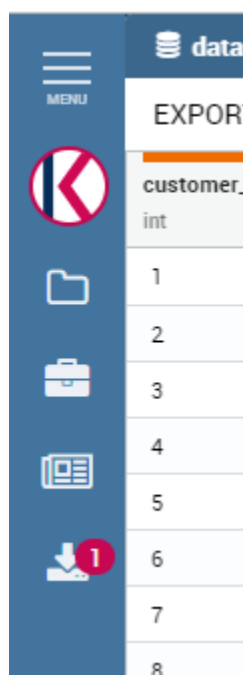
After the process is over a notification badge will appear on the *downloads* menu icon.



The image shows a user interface for a dataset. At the top, there is a blue header bar with the text 'dataset:'. Below this, there is a white bar with two buttons: 'EXPORT CSV' and 'EXPORT XLSX'. These buttons are highlighted with a red rectangular box. Below the buttons, there is a table with three columns: 'detectionyear', 'detectionmonth', and 'detection'. The first column has a data type of 'int' and contains the values '2013' and '2013'. The second column has a data type of 'int' and contains the values '11' and '4'. The third column has a data type of 'int' and contains the values '11' and '3'.

detectionyear int	detectionmonth int	detection int
2013	11	11
2013	4	3

Fig. 5.541: Dataset export bar.



The image shows a vertical sidebar on the left side of a user interface. The sidebar has a blue background and contains several icons: a menu icon, a 'K' logo, a folder icon, a briefcase icon, a document icon, and a download icon. The download icon is highlighted with a red circle and a white number '1'. To the right of the sidebar, there is a table with a header row labeled 'customer.' and a data type of 'int'. The table contains a list of numbers from 1 to 8.

customer. int
1
2
3
4
5
6
7
8

Fig. 5.542: Download notification badge.

5.22.3 Inside workspace

Inside workspace you can access the dataset preview clicking on the preview button like in the example below.

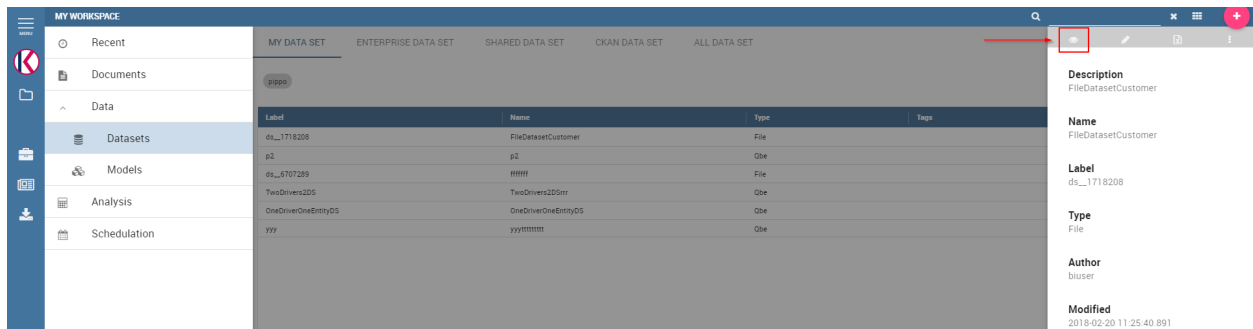


Fig. 5.543: How to access workspace preview.

5.22.4 Inside cockpit

Inside cockpit you can find the dataset preview settings in the following widget configuration:

- Table widget
- Chart widget
- Html widget

Opening the widget settings the *cross* tab will be available, with the cross-navigation and preview functionalities.

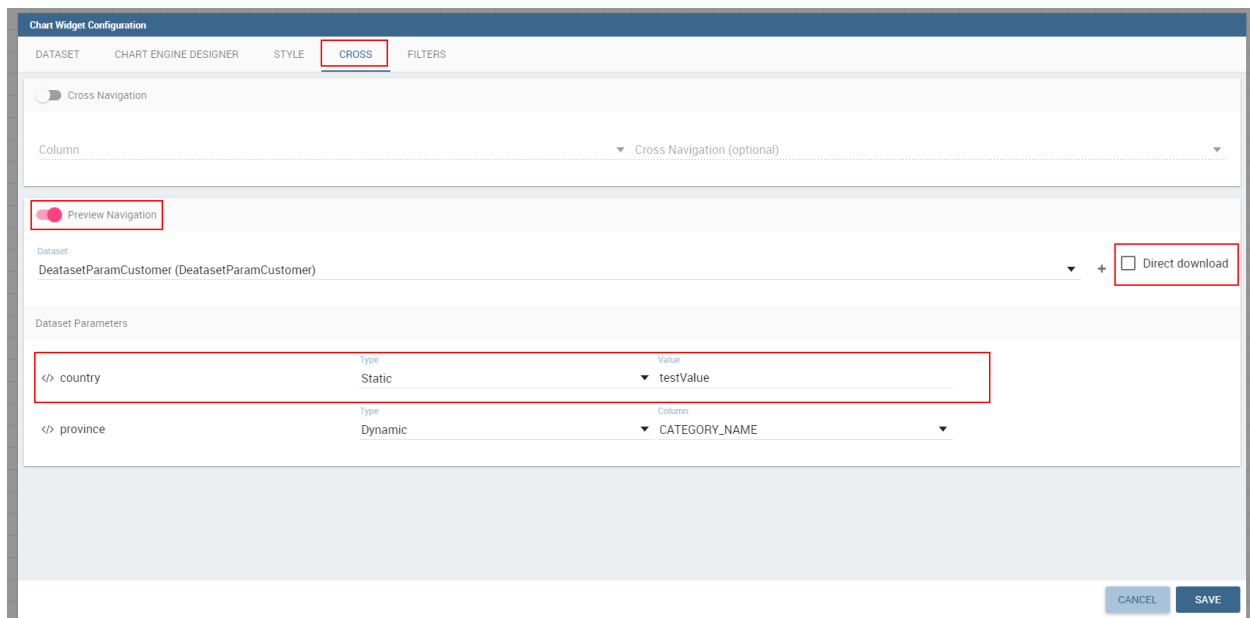


Fig. 5.544: Preview navigation configuration.

Enabling the preview toggle you will have the possibility to choose which dataset will be show in preview.

If some dataset parameters are set, those will be visibile in the list below. It's possible to set the parameter's value in different ways depending on the selected widget.

Selecting *Direct download* checkbox will allow the user to export directly the preview result on interaction. The download will be available as a common preview export inside downloads menu.